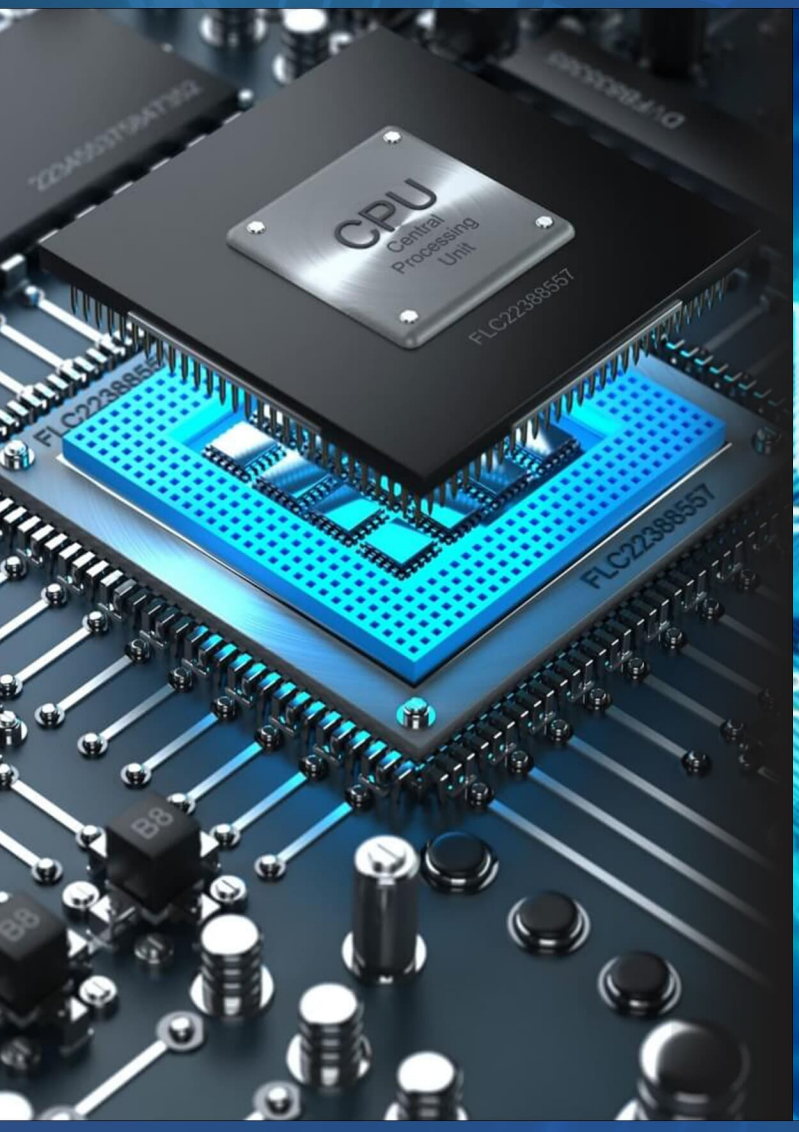
ស្ថាបត្យកម្មកុំព្យូទ័រ និង IOT



រៀបរៀងដោយ
អ៊ុក ប៉ុល្លីវ៉ាន់
២០២១

CAMBODIA
RESEARCH CREATIVITY INNOVATION

គាំទ្រថវិការលើការរៀបរៀង និងបោះពុម្ពដោយ
មូលនិធិស្រាវជ្រាវ កំនិតច្នៃប្រឌិត និងនវានុវត្តន៍

ស្ថាបត្យកម្មកុំព្យូទ័រ និង IoT

រៀបរៀងដោយ
អ៊ុក ប៉ុល្លីវ៉ាន់
២០២១

គាំទ្រថវិការលើការរៀបរៀង និងបង្ហោះដោយ
មូលនិធិស្រាវជ្រាវ គំនិតច្នៃប្រឌិត និងនវានុវត្តន៍

សាកលវិទ្យាល័យភូមិន្ទភ្នំពេញ
មហាវិទ្យាល័យវិទ្យាសាស្ត្រ
ដេប៉ាតឺម៉ង់ព័ត៌មានវិទ្យា

គណៈកម្មការនិពន្ធ

១. លោក **អ៊ុក ប៉ុល្លីវ៉ាន់** សាស្ត្រាចារ្យដេប៉ាតឺម៉ង់ព័ត៌មានវិទ្យា

គណៈកម្មការត្រួតពិនិត្យ

១. លោក **ប៉ុក លក្សមុនី** អនុប្រធានដេប៉ាតឺម៉ង់ព័ត៌មានវិទ្យា	ប្រធាន
២. លោក **ជា សំណាង** អនុប្រធានដេប៉ាតឺម៉ង់ថ្នាក់ឆ្នាំសិក្សាមូលដ្ឋាន	អនុប្រធាន
៣. លោក **ជឹម ប៊ុនធឿន** សាស្ត្រាចារ្យដេប៉ាតឺម៉ង់ព័ត៌មានវិទ្យា	សមាជិក
៤. អ្នកស្រី **ស្វីយ សុវត្ថារី** សាស្ត្រាចារ្យដេប៉ាតឺម៉ង់ព័ត៌មានវិទ្យា	សមាជិក

បុព្វកថា

ដំណើរអភិវឌ្ឍនៃព្រះរាជាណាចក្រកម្ពុជានៅក្នុងយុគសម័យទំនើបនេះ ជាមេរៀនដ៏ជោគជ័យបំផុតមួយ ដែលចាប់ឫសគល់ចេញពីការបញ្ចប់របបប្រល័យពូជសាសន៍ ការបញ្ចប់សង្គ្រាម ការផ្សះផ្សារជាតិ ការកសាងមូលដ្ឋានរឹងមាំនៃសន្តិភាពនិងស្ថេរភាព និងការអភិវឌ្ឍសេដ្ឋកិច្ច។ នៅក្រោយពេលដែលសន្តិភាពត្រូវបានកើតឡើងដោយបរិបូរណ៍នៅឆ្នាំ១៩៩៨ កម្ពុជាទទួលកំណើនសេដ្ឋកិច្ចខ្ពស់ គឺប្រមាណ៨% ក្នុងមួយឆ្នាំ។ លើសពីនេះទៀត អត្រានៃភាពក្រីក្រត្រូវបានកាត់បន្ថយពីប្រមាណ៥៣% នៅឆ្នាំ២០០៤ មកនៅទាបជាង១០% នៅឆ្នាំ២០១៩។ ដំណើរនៃការអភិវឌ្ឍជាតិជាសកម្មភាពដែលបន្តទៅមុខជាប់ជានិច្ច ហើយគោលនយោបាយថ្មីៗដែលមានលក្ខណៈអន្តរវិស័យគ្របដណ្តប់ក៏កំពុងលេចរូបរាងឡើង ដើម្បីតម្រង់ទិសកម្ពុជាឆ្ពោះទៅកាន់ប្រទេសមានប្រាក់ចំណូលមធ្យមកម្រិតខ្ពស់នៅឆ្នាំ២០៣០ និងឈានឡើងជាប្រទេសមានប្រាក់ចំណូលខ្ពស់ នៅឆ្នាំ២០៥០។ ការប្រែប្រួលឆាប់រហ័សនៃនិម្មាបនកម្មពិភពលោកនិងតំបន់ រួមទាំងទំនាក់ទំនងភូមិសាស្ត្រនយោបាយ បានផ្តល់កាលានុវត្តភាពសម្រាប់ការអភិវឌ្ឍឧស្សាហកម្មនៅកម្ពុជា ដែលត្រូវបានរាជរដ្ឋាភិបាលចាត់ទុកជាមូលដ្ឋានគ្រឹះនៃកំណើនសេដ្ឋកិច្ចកម្ពុជា។ រាជរដ្ឋាភិបាលកម្ពុជាបាន និងកំពុងបន្តពង្រឹងនិងអភិវឌ្ឍវិស័យអប់រំឆ្ពោះទៅរកការស្រាវជ្រាវនិងនវានុវត្តន៍ ដើម្បីពង្រឹងសមត្ថភាពនិងជំនាញរបស់ធនធានមនុស្សនៅកម្ពុជា ឱ្យស្របទៅនឹងបរិបទថ្មីនៃការអភិវឌ្ឍ ជាពិសេសការពង្រឹងសហគ្រិនភាពក្នុងការរៀបចំម៉ូដែលធុរកិច្ចថ្មីៗ។ ដើម្បីចាប់យកកាលានុវត្តភាពពីបដិវត្តន៍ឧស្សាហកម្មទី៤ និងសេដ្ឋកិច្ចឌីជីថលដែលកំពុងផុសផុលឡើង ប្រព័ន្ធអេកូឡូហ្ស៊ីដែលបង្កលក្ខណៈអំណោយផលដល់ការបង្កើតថ្មី នវានុវត្តន៍ ការស្រាវជ្រាវ និងអភិវឌ្ឍន៍ត្រូវតែមានការកែលម្អ។

បណ្តាប្រទេសនៅទ្វីបអាស៊ីកំពុងនាំមុខក្នុងការវិនិយោគលើការស្រាវជ្រាវនិងអភិវឌ្ឍ ដោយមានភាគហ៊ុនប្រមាណ៤៤% នៃការវិនិយោគទាំងមូលរបស់ពិភពលោក។ ប្រទេសចិនកំពុងបន្តកសាងហេដ្ឋារចនាសម្ព័ន្ធនៃការវិនិយោគលើការស្រាវជ្រាវនិងអភិវឌ្ឍ ក៏ដូចជាសមត្ថភាពមនុស្ស។ ផ្ទុយទៅវិញ ប្រទេសនៅទ្វីបអាមេរិកខាងត្បូងនិងអាហ្វ្រិក កំពុងស្ថិតនៅឆ្ងាយពីការវិនិយោគនេះ ហើយជាលទ្ធផលប្រទេសទាំងនោះក៏ពុំមានកំណើនសេដ្ឋកិច្ចគួរឱ្យកត់សម្គាល់ដែរ។ ទុនវិនិយោគសរុបលើការស្រាវជ្រាវនិងអភិវឌ្ឍរបស់ប្រទេសនៅទ្វីបអាមេរិកខាងត្បូងនិងអាហ្វ្រិក មានប្រមាណ៥%នៃការវិនិយោគទាំងមូលរបស់ពិភពលោក ក្នុងពេលដែលតំបន់ទាំង២នេះមានប្រជាជនប្រមាណ២០%នៃប្រជាជនពិភពលោក។ ប្រទេសចំនួន៦ដែលមានលំដាប់ខ្ពស់ជាងគេនៅក្នុងការវិនិយោគលើការស្រាវជ្រាវនិងអភិវឌ្ឍ រួមមានសហរដ្ឋអាមេរិក ចិន ជប៉ុន អាល្លឺម៉ង់ ឥណ្ឌា និងកូរ៉េខាងត្បូង ដែលស្មើនឹងប្រមាណ៧០%នៃទុនវិនិយោគសរុបរបស់ពិភពលោក។

តើចំណេះដឹង ផលិតផល និងសេវាកម្មថ្មីទាំងនេះកើតឡើងពីអ្វី? ហើយកើតឡើងដោយរបៀបណា? ព្រះរាជាណាចក្រកម្ពុជាកំពុងតែកសាងមូលដ្ឋានសម្រាប់ការត្រៀមខ្លួនទទួល និងប្រកួតប្រជែងក្នុងយុគសម័យបដិវត្តឧស្សាហកម្មទី៤ នៅក្នុងសេដ្ឋកិច្ចដែលផ្អែកលើពុទ្ធិ ហើយដែលប្រការនេះ

ចាំបាច់តម្រូវឱ្យពលរដ្ឋកម្ពុជា ត្រូវក្លាយខ្លួនជាពលរដ្ឋឌីជីថល ពលរដ្ឋសកល និងពលរដ្ឋដែលប្រកបដោយការទទួលខុសត្រូវ ដែលមានសមត្ថភាពក្នុងការផលិត ចែកចាយ និងប្រើប្រាស់ពុទ្ធិដើម្បីទទួលមនុញ្ញផល និងរួមចំណែកក្នុងកំណើន។ ធនាគារពិភពលោកបានធ្វើការកត់សម្គាល់តាំងពីឆ្នាំ ២០០២នូវបម្លាស់ប្តូរនៃមូលដ្ឋានសេដ្ឋកិច្ច ពីសេដ្ឋកិច្ចដែលពឹងផ្អែកលើកម្លាំងពលកម្ម និងធនធានអតិកម្ម (Labour and Resource Based Economy) ទៅកាន់សេដ្ឋកិច្ចដែលពឹងផ្អែកលើពុទ្ធិ (Knowledge Based-Economy) ដែលក្នុងន័យនេះ ពុទ្ធិគឺជាគន្លឹះនៃការអភិវឌ្ឍ។ អាស្រ័យហេតុនេះ នៅលើគន្លងដែលកម្ពុជាកំពុងធ្វើដំណើរឆ្ពោះទៅកាន់សេដ្ឋកិច្ចឌីជីថល សង្គមកម្ពុជាត្រូវតែមានសមត្ថភាពក្នុងការផលិត ជ្រើសរើស បន្សាំ បង្កើតមុខរបរ និងប្រើប្រាស់ពុទ្ធិ ដើម្បីរក្សានិរន្តរភាពនៃកំណើន និងកែលម្អជីវភាពរស់នៅ។ សមត្ថភាពទាំងនេះ អាចកើតឡើងនៅពេលពលរដ្ឋកម្ពុជាមានឱកាសក្នុងការទទួលបានបទពិសោធន៍ពីការស្រាវជ្រាវ ការបណ្តុះគំនិតច្នៃប្រឌិត និងការស្វែងរកនវានុវត្តន៍។

កំណែទម្រង់វិស័យអប់រំ គឺជាការត្រួសត្រាយមាគ៌ាសម្រាប់ដំណើរឆ្ពោះទៅកាន់សង្គមប្រកបដោយពុទ្ធិ និងប្រជាពលរដ្ឋប្រកបដោយភាពរស់រវើក។ តាមរយៈមូលដ្ឋានអប់រំ សង្គមប្រកបដោយពុទ្ធិនឹងប្រមូលផ្តុំ បង្កើត និងចែករំលែក ទៅកាន់សមាជិកក្នុងសង្គមនូវសម្បទាអប់រំ ពិសេសគឺពុទ្ធិសម្បទាក្នុងបុព្វហេតុនៃមនុស្សជាតិនិងឧត្តមប្រយោជន៍នៃប្រទេស។ សង្គមប្រកបដោយពុទ្ធិ គឺពុំគ្រាន់តែជាសង្គមដែលសម្បូរព័ត៌មានប៉ុណ្ណោះទេ តែជាសង្គមដែលប្រជាពលរដ្ឋអាចធ្វើបរិវត្តកម្មពីព័ត៌មានទៅជាមូលធនប្រកបដោយប្រសិទ្ធភាព។ ការរីកចម្រើនទៅមុខដ៏លំដាប់នៃបច្ចេកវិទ្យានិងតំណភ្ជាប់ បានពង្រីកព្រំដែននៃការចូលទៅកាន់ និងការទទួលបានព័ត៌មានជាសកល ហើយដែលក្នុងន័យនេះ ការអប់រំនឹងបន្តវិវត្តទៅមុខនិងមានការផ្លាស់ប្តូរ។ សង្គមមួយដែលមានអំណាន និងរបាប់ជាបុរេលក្ខខណ្ឌនៃជីវភាពប្រចាំថ្ងៃនៃប្រជាពលរដ្ឋ ពេលនោះបំណិននៃអំណាន និពន្ធ និងការគណនាលេខនព្វន្ត គឺជាចលករនៃការរៀនរបស់សិស្ស។ ធាតុដ៏ចម្បងមួយដែលស្ថិតនៅក្នុងការកសាងសង្គមដែលប្រកបដោយពុទ្ធិគឺសៀវភៅសិក្សា ហើយការរៀបរៀង និពន្ធ និងកែលម្អសៀវភៅសិក្សាជាប្រចាំ គឺជានវានុវត្តន៍នៃវិស័យអប់រំដែលនាំទៅរកការសិក្សាពេញមួយជីវិត ការអភិវឌ្ឍសម្បទាអប់រំ និងការចែករំលែកចំណេះដឹង។ មូលដ្ឋានអប់រំ ជាពិសេសគឺគ្រឹះស្ថានឧត្តមសិក្សាត្រូវមានតួនាទីដែលប្រកបដោយការឆ្លើយតប ចំពោះតម្រូវការខាងលើនេះ។ សាស្ត្រាចារ្យ អ្នកស្រាវជ្រាវ និងបុគ្គលិកអប់រំត្រូវបន្តសិក្សាជាប់ជានិច្ច តាមរយៈការរៀបរៀង និពន្ធ និងកែលម្អសៀវភៅសិក្សា ហើយដែលសៀវភៅសិក្សាទាំងនេះនឹងក្លាយជាស្ពាននៃទំនាក់ទំនងរវាងនវានុវត្តន៍នៃបច្ចេកវិទ្យា និងការរៀននិងបង្រៀននៅក្នុងថ្នាក់រៀន។

សង្គមដែលប្រកបពុទ្ធិ ក៏ជាសង្គមដែលបណ្តុះឱ្យមានរចនាសម្ព័ន្ធទន់នៃសេដ្ឋកិច្ចដែលពឹងផ្អែកលើពុទ្ធិដែរ។ ឧទាហរណ៍ជាក់ស្តែងនៃបែបផែននេះរួមមាន Silicon Valley នៃសហរដ្ឋអាមេរិក សួនឧស្សាហកម្មវិស្វកម្មអាកាសយានយន្តនិងយានយន្តនៅទីក្រុង Munich ប្រទេសអាល្លឺម៉ង់ តំបន់ជីវបច្ចេកវិទ្យានៅក្រុង Hyderabad ប្រទេសឥណ្ឌា តំបន់ផលិតគ្រឿងអេឡិចត្រូនិកនិងសារគមនាគមន៍ឌីជីថលនៅទីក្រុង Seoul ប្រទេសកូរេខាងត្បូង ក៏ដូចជាសួនឧស្សាហកម្មថាមពល និងឥន្ធនគីមីសាស្ត្រនៃប្រទេសប្រេស៊ីល ហើយក៏នៅមានទីក្រុងនៃប្រទេសជាច្រើនទៀតនៅលើពិភពលោក។ លក្ខណៈសម្បត្តិ

នៃទីក្រុងទាំងនេះគឺការប្រើប្រាស់និន្នាការនៃការអភិវឌ្ឍដែលជំរុញ និងតម្រង់ទិសដោយចំណេះដឹង ហើយដែលចំណេះដឹងទាំងនោះកើតចេញជាដំបូងពីការវិនិយោគទៅលើគ្រឹះស្ថានឧត្តមសិក្សា ស្ថាប័នស្រាវជ្រាវ មជ្ឈមណ្ឌលឧត្តមភាពនៃជំនាញជាន់ខ្ពស់ ការប្រកួតប្រជែងដោយគុណាធិបតេយ្យ និងជាពិសេសគឺការបណ្តុះវប្បធម៌អំណាននិងនិពន្ធសៀវភៅ។ ល្បឿននៃការរីកចម្រើនផ្នែកពុទ្ធិ និងបច្ចេកវិទ្យាកំពុងមានសន្ទុះលឿនជាងអ្វីដែលសិស្ស និងនិស្សិតអាចទទួលបានពីគ្រូនៅគ្រឹះស្ថានសិក្សា ដែលធ្វើឱ្យគោលដៅនៃការអប់រំនៅពេលបច្ចុប្បន្ននេះ មានការប្រឈមខ្លាំងជាងពេលណាទាំងអស់។ ឧទាហរណ៍ ក្នុងមួយឆ្នាំ មានសៀវភៅជាង២,២លានចំណងជើង ត្រូវបានសរសេរនិងបោះពុម្ព ដែលក្នុងនោះ ប្រទេសចិនមាន៤៤០ពាន់ ចំណែកឯសហរដ្ឋអាមេរិកមាន៣០៥ពាន់ និងប្រទេសរុស្ស៊ីមាន១២០ពាន់ចំណងជើង។

ខណៈពេលដែលបច្ចេកវិទ្យាកំពុងរីកចម្រើនជារៀងរាល់ថ្ងៃ មធ្យោបាយសម្រាប់អំណានក៏មានច្រើនជម្រើសសម្រាប់សិស្ស-និស្សិត និងសាធារណៈជន រួមមានការអានសៀវភៅ ការអានលើឧបករណ៍អេឡិចត្រូនិក ការអានដោយប្រើទូរសព្ទវៃឆ្លាត និងការអានលើកុំព្យូទ័រ ដែលសុទ្ធសឹងជាមធ្យោបាយសំខាន់ៗដែលនាំអ្នកអានទាំងឡាយឱ្យសម្រេចគោលបំណងអានរបស់ខ្លួន។ ម្យ៉ាងវិញទៀត អំណានដោយប្រើមធ្យោបាយបច្ចេកវិទ្យាទំនើប ចំណាយពេលតិច ងាយស្រួលអាន និងជួយដល់បរិស្ថានមួយកម្រិតទៀត។ នាពេលបច្ចុប្បន្ន សិស្ស-និស្សិត និងសាធារណៈជនកម្ពុជាដែលស្រឡាញ់អំណានកំពុងតែប្រើប្រាស់មធ្យោបាយអំណានទាំងនេះ។ បើយើងក្រឡេកមើលទៅប្រទេសជឿនលឿន ទោះបីជាបច្ចេកវិទ្យារីកចម្រើនខ្លាំងយ៉ាងណា អំណានតាមរយៈសៀវភៅនៅតែមានសន្ទុះដដែល។ ម្យ៉ាងវិញទៀត បច្ចេកវិទ្យាអានបែបទំនើបតាមរយៈឧបករណ៍ទំនើប អាស្រ័យលើលទ្ធភាពនៃធនធានអប់រំឌីជីថល និងមាតិកាឌីជីថលគ្រប់គ្រាន់ដែលបានផលិត និងបង្ហោះចែកចាយសម្រាប់អំណាន។

ក្នុងបរិបទកម្ពុជា ជាពិសេសក្នុងបរិការណ៍នៃការផ្ទុះរីករាលដាលនៃជំងឺកូវីដ-១៩ ក្រសួងអប់រំ យុវជន និងកីឡា បានជំរុញឱ្យមានបរិវត្តកម្មឌីជីថលនៅក្នុងអេកូស៊ីស្តែមនៃការអប់រំ ជាពិសេសការអប់រំតាមប្រព័ន្ធអេឡិចត្រូនិកនិងការអប់រំពីចម្ងាយ ដើម្បីលើកកម្ពស់អំណាន តាមរយៈការផលិតមាតិកាឌីជីថលដែលមានភាពចំរុះ ការកសាងសមត្ថភាពផ្នែកតំណភ្ជាប់និងវេទិកាឌីជីថល ការពង្រីកវិសាលភាពនៃមជ្ឈមណ្ឌលទិន្នន័យ និងការលើកកម្ពស់គុណភាពនៃការផលិតធនធានអប់រំឌីជីថល គួបផ្សំជាមួយការចែកសន្លឹកកិច្ចការឱ្យសិស្សយកទៅរៀននៅផ្ទះ និងការចុះទៅជួបជាមួយសិស្សជាបណ្តុំនៅតាមសហគមន៍។ ក្នុងន័យលើកកម្ពស់អំណាន និងភាពសម្បូរបែបនៃធនធានសៀវភៅសិក្សា ឱ្យកាន់តែមានប្រសិទ្ធភាពនិងភាពសក្តិសិទ្ធិ និងផ្តល់ឱកាសអំណានកាន់តែច្រើនថែមទៀតដល់សិស្សានុសិស្ស និស្សិត និងសាធារណៈជន ក្រសួងអប់រំ យុវជន និងកីឡាលើកទឹកចិត្តនូវចំណុចមួយចំនួនដូចខាងក្រោម៖

១. សាស្ត្រាចារ្យ អ្នកស្រាវជ្រាវ និងបុគ្គលិកអប់រំ សូមបន្តនិងបង្កើនការបោះពុម្ពស្នាដៃបន្ថែមទៀត ដើម្បីធ្វើឱ្យធនធានសម្រាប់អំណានកាន់តែសម្បូរបែប ជាពិសេសធនធានអំណានជាខេមរភាសា

២. គ្រឹះស្ថានឧត្តមសិក្សា សូមផ្តល់លទ្ធភាពគ្រប់បែបយ៉ាង ដើម្បីឱ្យបុគ្គលិកអប់រំគ្រប់លំដាប់ថ្នាក់ និងនិស្សិតគ្រប់កម្រិតសិក្សាអាចចូលរួមអាន និងសិក្សាស្រាវជ្រាវតាមគ្រប់លទ្ធភាព ជាមួយធនធានអំណាន ជាពិសេសការរៀបចំឱ្យមានពេលវេលាសម្រាប់សហសិក្សា និងអំណានក្នុងបណ្ណាល័យ

៣. សាស្ត្រាចារ្យតាមមុខវិជ្ជា និងអ្នកស្រាវជ្រាវតាមជំនាញឬវិស័យ ត្រូវរៀបចំដំណើរការរៀនបង្រៀន និងស្រាវជ្រាវដែលមានដាក់បញ្ចូលកិច្ចការស្វ័យសិក្សា សហសិក្សា ឬការស្រាវជ្រាវបណ្ណាល័យដែលតម្រូវឱ្យនិស្សិត ត្រូវអាននិងស្រាវជ្រាវជាមួយធនធានអំណាន

៤. គ្រឹះស្ថានឧត្តមសិក្សា និងមជ្ឈមណ្ឌលស្រាវជ្រាវ ត្រូវខិតខំឱ្យអស់លទ្ធភាពក្នុងការបង្កើតបណ្ណាល័យ មជ្ឈមណ្ឌលរក្សាឯកសារ ឬមជ្ឈមណ្ឌលអប់រំឌីជីថលជាដើម ដើម្បីឱ្យបុគ្គលិកអប់រំគ្រប់លំដាប់ថ្នាក់និងនិស្សិតគ្រប់កម្រិតសិក្សាអាចទទួលបាន និងស្វែងរកប្រភពសម្រាប់អំណានកាន់តែសម្បូរបែប និងមានភាពបត់បែន ឆ្លើយតបតាមតម្រូវការអ្នកអាន

៥. និស្សិតគ្រប់កម្រិតសិក្សាត្រូវខិតខំនិងចំណាយពេលវេលាដើម្បីអាន និងចាត់ទុកវប្បធម៌និងអកប្បកិរិយាអំណានជាផ្នែកមួយ នៃពេលវេលានិងភាពស៊ីវិល័យនៃជីវិតប្រចាំថ្ងៃ

៦. បងប្អូនជនរួមជាតិ ដែលជាមាតាបិតា ឬអ្នកអាណាព្យាបាល សូមជួយជំរុញនិងបង្កលក្ខណៈកាន់តែច្រើនថែមទៀត ជាពិសេសការលៃលកចំណាយនៅក្នុងគ្រួសារសម្រាប់ការទិញសម្ភារៈសិក្សា សៀវភៅអាន និងឧបករណ៍សម្រាប់អំណានដល់កូនៗ ដែលចាត់ទុកជាការវិនិយោគមួយដ៏សំខាន់ សម្រាប់ បង្កើនចំណេះដឹង និងអនាគតរបស់ពួកគេ។

ដោយមានការគាំទ្រពីក្រសួងសេដ្ឋកិច្ច និងហិរញ្ញវត្ថុ នៅឆ្នាំ២០២០ ក្រសួងអប់រំ យុវជន និងកីឡា បានបង្កើតមូលនិធិស្រាវជ្រាវ គំនិតច្នៃប្រឌិត និងនវានុវត្តន៍ ដែលហៅកាត់ថា “មូលនិធិ ស.គ.ន.” និងហៅជាភាសាអង់គ្លេសថា The Research Creativity and Innovation Fund ដែលហៅកាត់ជាភាសាអង់គ្លេសថា “RCI Fund”។ គោលដៅចម្បងនៃមូលនិធិនេះ គឺរួមចំណែកលើកកម្ពស់វប្បធម៌នៃការស្រាវជ្រាវ បំផុសគំនិតច្នៃប្រឌិត និងជំរុញការធ្វើនវានុវត្ត ដើម្បីជាប្រយោជន៍ដល់វិស័យអប់រំ យុវជន និងកីឡា ដែលឆ្លើយតបទៅនឹងទីផ្សារពលកម្ម និងសាកលភាវូបនីយកម្ម។ មូលនិធិ ស.គ.ន. បានសម្រេចកំណត់ប្រធានបទ ជាអាទិភាពសម្រាប់ការគាំទ្រដោយមូលនិធិចំនួន៣ រួមមានឌីជីថលនីយកម្មសម្រាប់បដិវត្តឧស្សាហកម្ម៤.0 (Digitalization for IR.4.0) ការស្រាវជ្រាវអនុវត្តលើវិស័យកសិកម្ម (Applied Agricultural Research) និងការស្រាវជ្រាវគរុកោសល្យសតវត្សទី២១ (21st Century Pedagogy Research)។

ដោយមានការធ្វើអាទិភាពវូបនីយកម្មទៅលើទិសដៅ នៃការប្រើប្រាស់ថវិកាមូលនិធិសម្រាប់ឆ្នាំ ២០២០ ក្រសួងសេដ្ឋកិច្ច និងហិរញ្ញវត្ថុ និងក្រសួងអប់រំ យុវជន និងកីឡា បានផ្តល់ការគាំទ្រដល់ការ**រៀបរៀង និពន្ធ និងកែលម្អ សៀវភៅសិក្សា (Text book) ដែលនឹងត្រូវប្រើប្រាស់នៅកម្រិតឧត្តមសិក្សា**។ គោលបំណងនៃការរៀបរៀង និពន្ធ និងកែលម្អ សៀវភៅសិក្សានៅកម្រិតឧត្តមសិក្សា គឺដើម្បីបង្កើនបរិមាណ លើកកម្ពស់គុណភាព និងពង្រីកសមធម៌នៃធនធានសិក្សាជាខេមរភាសា ជូនដល់និស្សិត

ដែលកំពុងបន្តការសិក្សា និងត្រៀមខ្លួនធ្វើការស្រាវជ្រាវនៅកម្រិតឧត្តមសិក្សា។ លើសពីនេះទៀត ការរៀបរៀង និពន្ធ និងកែលម្អសៀវភៅសិក្សានៅកម្រិតឧត្តមសិក្សា មានគោលដៅដូចខាងក្រោម ៖

១. ឆ្លើយតបជាបន្ទាន់ចំពោះការខ្វះខាតធនធានសិក្សា ដែលជាតម្រូវការសិក្សារបស់និស្សិតនៅកម្រិតឧត្តមសិក្សា

២. លើកកម្ពស់ទំនើបភាវូបនីយកម្ម និងឧត្តមានុវត្តន៍នៃការរៀននិងបង្រៀន និងការស្រាវជ្រាវនៅលើមុខវិជ្ជា កម្មវិធីសិក្សា ឬមុខជំនាញជាក់លាក់

៣. បង្កើនភាពស៊ីជម្រៅក្នុងការកសាងវិជ្ជាជីវៈនិងបទពិសោធន៍សម្រាប់ហានៈសាស្ត្រាចារ្យ និងអ្នកស្រាវជ្រាវ

៤. រួមចំណែកដល់ការកសាងភាពជាសហគមន៍វិជ្ជាជីវៈ ការចែករំលែកបទពិសោធន៍ និងវប្បធម៌នៃការរៀបរៀង និពន្ធ និងកែលម្អសៀវភៅសិក្សានៅកម្រិតឧត្តមសិក្សា។

ក្រសួងអប់រំ យុវជន និងកីឡា បានវាយតម្លៃខ្ពស់ចំពោះការបោះជំហានប្រកបដោយមនសិការវិជ្ជាជីវៈនៃគ្រឹះស្ថានឧត្តមសិក្សា និងបុគ្គលិកអប់រំទាំងអស់ ក្នុងការរៀបចំ រៀបរៀង និពន្ធ និងកែលម្អ សៀវភៅសិក្សា ដើម្បីបង្កើនបរិមាណ លើកកម្ពស់គុណភាព និងពង្រឹងសមធម៌នៃធនធានសិក្សាជាខេមរភាសា ជូននិស្សិតដែលកំពុងបន្តការសិក្សា និងត្រៀមខ្លួនធ្វើការស្រាវជ្រាវនៅកម្រិតឧត្តមសិក្សា។ សៀវភៅសិក្សាជាផ្នែកមួយនៃការទទួលស្គាល់គុណភាពអប់រំនៃគ្រឹះស្ថានឧត្តមសិក្សា និងជាធនធានសិក្សាដែលជាមូលដ្ឋានមួយដ៏សំខាន់ ក្នុងការគាំទ្រដល់ការបង្រៀន និងរៀន ហើយត្រូវមានបរិមាណគ្រប់គ្រាន់ ឆ្លើយតបទៅនឹងកម្មវិធីអប់រំ និងតម្រូវការសិក្សាស្រាវជ្រាវ។ ជាគោលការណ៍ គ្រឹះស្ថានឧត្តមសិក្សាទាំងអស់ ត្រូវមានសៀវភៅសិក្សាដែលប្រើជាគោលសម្រាប់មុខវិជ្ជានីមួយៗ។ ចំនួនសៀវភៅសិក្សាដែលគ្រប់គ្រាន់សម្រាប់ការស្រាវជ្រាវ និងការសិក្សារបស់និស្សិត ត្រូវមានយ៉ាងតិចមួយចំណងជើងក្នុងមួយមុខវិជ្ជា ហើយត្រូវតម្កល់យ៉ាងតិច២ច្បាប់នៅក្នុងបណ្ណាល័យ ឬអាចរកបានតាមប្រព័ន្ធអេឡិចត្រូនិក។ ក្រសួងអប់រំ យុវជន និងកីឡា លើកទឹកចិត្តបន្ថែមទៀតជូនដល់គ្រឹះស្ថានឧត្តមសិក្សារដ្ឋ និងឯកជនដែលបានស្នើសុំថវិកាមូលនិធិ ស.គ.ន រួច សូមចូលរួមបន្ថែមទៀតដើម្បីបង្កើនចំនួនចំណងជើងសៀវភៅ។ ចំណែកគ្រឹះស្ថានឧត្តមសិក្សារដ្ឋ និងឯកជនដែលពុំទាន់បានដាក់ពាក្យស្នើសុំថវិកាមូលនិធិ ដើម្បីរៀបរៀង និពន្ធ និងកែលម្អ សៀវភៅសិក្សានៅកម្រិតឧត្តមសិក្សា សូមរួសរាន់ចូលរួមដើម្បីជាគុណប្រយោជន៍ដល់តម្រូវការដ៏ទទូចនិងថ្លៃថ្លានៃនិស្សិតកម្ពុជាក្នុងការសិក្សា និងស្រាវជ្រាវនៅកម្រិតឧត្តមសិក្សា។

សេចក្តីបញ្ជាក់
នៃមូលនិធិស្រាវជ្រាវ គំនិតច្នៃប្រឌិត និងនវានុវត្តន៍

សៀវភៅសិក្សានេះជាលទ្ធផលនៃការស្នើសុំអនុវត្តថវិកាមូលនិធិស្រាវជ្រាវ គំនិតច្នៃប្រឌិត និងនវានុវត្តន៍ ក្នុងគម្រោង**រៀបរៀង និពន្ធ និងកែលម្អសៀវភៅសិក្សា ដែលនឹងត្រូវប្រើប្រាស់នៅកម្រិតឧត្តមសិក្សា។** សៀវភៅសិក្សានេះ ត្រូវបានរៀបរៀង និពន្ធ ឬកែលម្អដោយមានការធានាអះអាងថាជាស្នាដៃរបស់អ្នកនិពន្ធផ្ទាល់ និងបានឆ្លងកាត់ត្រួតពិនិត្យ ផ្តល់យោបល់ និងវាយតម្លៃដោយក្រុមប្រឹក្សាអប់រំ ក្រុមប្រឹក្សាស្រាវជ្រាវ ឬក្រុមប្រឹក្សាដែលមានតម្លៃស្មើនៃគ្រឹះស្ថានឧត្តមសិក្សា និងតាមរយៈកិច្ចសន្យាដែលបានធ្វើឡើង និងដែលបានតម្កល់ទុកនៅមូលនិធិស្រាវជ្រាវ គំនិតច្នៃប្រឌិត និងនវានុវត្តន៍។ ពាល់ខ្លឹមសារ ការបកស្រាយ ឬរូបភាព ដែលមាននៅក្នុងសៀវភៅនេះ គឺជាជំហរនិងទស្សនៈផ្ទាល់របស់អ្នកនិពន្ធ ហើយពុំឆ្លុះបញ្ចាំង ឬជាតំណាងដល់មូលនិធិស្រាវជ្រាវ គំនិតច្នៃប្រឌិត និងនវានុវត្តន៍ នៃក្រសួងអប់រំ យុវជន និងកីឡាឡើយ។

អារម្ភកថា

សៀវភៅនេះសម្រាប់បង្រៀនដល់អ្នកសិក្សាដែលរៀនផ្នែកព័ត៌មានវិទ្យា វិស្វកម្ម និងអេឡិចត្រូនិច។ វាជាមុខវិជ្ជាមួយយ៉ាងសំខាន់សម្រាប់រចនាកុំព្យូទ័រ, ទូរសព្ទ, ឧបករណ៍វៃឆ្លាត, រ៉ូបូត និង ការសរសេរកម្មវិធីដើម្បីអោយឧបករណ៍ទាំងនេះដំណើរការបាន។ ហើយវាក៏ជាមូលដ្ឋានគ្រឹះដ៏សំខាន់សម្រាប់រៀនបន្តទៅមុខវិជ្ជារ៉ូបូតទិក, AI, Machine Leaning និង Deep Leaning ដែលជាបច្ចេកវិទ្យាទំនើបប្រើក្នុងពេលបច្ចុប្បន្ននេះ។ ការរៀនមុខវិជ្ជានេះតម្រូវអោយអ្នកសិក្សាចេះប្រើប្រាស់ធុងប្រព័ន្ធរបស់កុំព្យូទ័រនិង Arduino ផ្សំជាមួយនឹងឧបករណ៍អេឡិត្រូនិចផ្សេងទៀត រួមទាំងការសរសេរកម្មវិធីដើម្បីធ្វើការតេស្តជាក់ស្តែងមុននឹងដំឡើងវាអោយទៅជាឧបករណ៍ម៉ាស៊ីនអេឡិចត្រូនិចមួយ។ ចេះពីការប្រើប្រាស់ប្រព័ន្ធ IoT ដើម្បីអោយអ្នកសិក្សាយល់ដឹងពីការតបណ្តាញ និង ធ្វើការតភ្ជាប់អ៊ិនតឺណិតនូវឧបករណ៍ដែលបានដំឡើងរួច និងអាចធ្វើការបញ្ជាពីចំងាយដោយប្រើកុំព្យូទ័រ, ទូរសព្ទ និង ឧបករណ៍បញ្ជាផ្សេងៗមួយចំនួនទៀត។

សៀវភៅនេះវាសមល្មមឲ្យអ្នកសិក្សាស្ថាបត្យកម្មកុំព្យូទ័រ និង IoT ទាំងឡាយមានមូលដ្ឋានគ្រឹះរឹងមាំក្នុងការសិក្សាបន្ត ឬស្រាវជ្រាវបន្ថែម ឬក្នុងការប្រកបការងារផ្សេងៗក្នុងវិជ្ជាជីវៈរបស់ខ្លួនផងដែរ។

ខ្ញុំបាទសង្ឃឹមថា សៀវភៅនេះនិងអាចបំពេញនូវតម្រូវការរបស់អ្នកសិក្សាហើយខ្ញុំបាទក៏រងចាំទទួលការទិទៀនកែលំអពីសំណាក់អ្នកសិក្សាទាំងឡាយ ដើម្បីអោយការរៀបរៀងក្រោយៗទៀតអោយកាន់តែល្អប្រសើរឡើង។

សេចក្តីថ្លែងអំណរគុណ

ខ្ញុំបាទឈ្មោះ **អ៊ុក ប៉ុល្លីវ៉ាន់** ជាសាស្ត្រាចារ្យដេប៉ាតឺម៉ង់ព័ត៌មានវិទ្យា នៃមហាវិទ្យាល័យវិទ្យាសាស្ត្រ នៃសាកលវិទ្យាល័យភូមិន្ទភ្នំពេញ និងជាអ្នករៀបរៀងសៀវភៅសិក្សាដែលមានចំណងជើងថា "ស្តាបត្យកម្ម កុំព្យូទ័រ និង IoT"។ ខ្ញុំបាទសូមសម្តែងនូវគារវកិច្ចដ៏ខ្ពង់ខ្ពស់ និងសូមគោរពថ្លែងអំណរគុណនិងដឹងគុណយ៉ាងជ្រាលជ្រៅចំពោះ៖

- ក្រសួងសេដ្ឋកិច្ច និង ហិរញ្ញវត្ថុ ដែលបានផ្តល់ថវិកាមកក្រសួងអប់រំ ដើម្បីបង្កើតជាមូលនិធិសម្រាប់គាំទ្រដល់ការសរសេរសៀវភៅសិក្សាសម្រាប់កម្រិតឧត្តមសិក្សារបស់ខ្ញុំបាទនេះ។

- ក្រសួងអប់រំ យុវជន និង កីឡា ដែលមាន ឯកឧត្តមបណ្ឌិតសភាចារ្យរដ្ឋមន្ត្រី ហង់ជួន ណារ៉ុន ជាប្រមុខដែលលោកបានដឹកនាំបង្កើតមូលនិធិស្រាវជ្រាវគំនិតឆ្នៃប្រឌិត និង នវានុវត្តន៍ឡើងដើម្បីគាំទ្រទាំងស្រុងដល់ដំណើរការសរសេរសៀវភៅនេះជាចំណែកមួយក្នុងបេសកកម្មកំណែទម្រង់វិស័យអប់រំឧត្តមសិក្សារបស់លោក។

- ឯកឧត្តមបណ្ឌិត សាន វឌ្ឍនា អនុរដ្ឋលេខាធិការក្រសួងអប់រំ យុវជន និង កីឡា ដែលក្នុងនាមលោកជាតំណាងមូលនិធិ លោកបានយកអស់កម្លាំងការចិត្ត ជួយចាត់ចែងនិងជ្រោមជ្រែងយ៉ាងពេញទំហឹង ព្រមទាំងផ្តល់ជាអនុសាសន៍និង ដំណោះស្រាយនានា ដើម្បីជំនះនូវាល់ឧបសគ្គដែលកើតមានក្នុងដំណើរការសរសេរសៀវភៅនេះ។

- ឯកឧត្តមបណ្ឌិត ជេត ជាលី សាកលវិទ្យាធិការនៃសាកលវិទ្យាល័យភូមិន្ទភ្នំពេញ ដែលបានយកអស់កម្លាំងកាយចិត្ត ព្រមទាំងពេលវេលាដ៏មមាញឹកនិង មានតម្លៃរបស់លោកក្នុងការជួយលើកទឹកចិត្តជួយខ្នះខ្នែងដោះស្រាយ និង ផ្តល់អនុសាសន៍ដ៏ត្រឹមត្រូវ ក៏ដូចជាបង្កលក្ខណៈងាយស្រួលនានា ទាំងផ្នែករដ្ឋបាល និង ទាំងផ្នែកបច្ចេកទេស ដើម្បីឲ្យដំណើរការរៀបរៀងសៀវភៅនេះបានប្រព្រឹត្តទៅដោយរលូនរហូតចប់ចុងចប់ដើម។

- លោក លួន ពេជ្របូរី និង អស់លោក-លោកស្រី ជាក្រុមការងារសម្របសម្រួលទាំងអស់របស់មូលនិធិស្រាវជ្រាវ គំនិតឆ្នៃប្រឌិត និងនវានុវត្តន៍ ដែលបានយកអស់កម្លាំងកាយចិត្ត ជួយឈឺឆ្អាលក្នុងការដោះស្រាយបញ្ហាប្រឈមនានាយ៉ាងសស្រាក់សស្រាំជាទីបំផុត ដោយមិនគ្នូញត្អែរ និងមិនខ្លាចនឿយហត់ ដើម្បីឲ្យការសរសេរសៀវភៅនេះប្រព្រឹត្តទៅដោយរលូន។

- ឯកឧត្តមបណ្ឌិត ស្ទ្រីវ ថាវិទ្ធ អ្នកសម្របសម្រួលសាលាក្រោយឧត្តម និង លោកបណ្ឌិត សុខ សិរី អនុប្រធានការិយាល័យស្រាវជ្រាវ ដែលតែងជួយឈឺឆ្អាល ជួយសម្របសម្រួល និងផ្តល់សេវាសម្រាប់ការប្រជុំបច្ចេកទេសនានាតាមអនឡាញតាំងពីដើមរៀងមក។

- អស់លោក-លោកស្រី ជាសមាជិក-សមាជិកាគណៈកម្មការអភិវឌ្ឍសៀវភៅសិក្សា ក៏ដូចជាគណៈកម្មការត្រួតពិនិត្យសៀវភៅសិក្សា របស់សាកលវិទ្យាល័យភូមិន្ទភ្នំពេញ ដែលបានជួយផ្តល់ធាតុចូលដល់សៀវភៅនេះឲ្យកាន់តែមានគុណភាពប្រសើរថែមទៀត។

- ជាចុងក្រោយ និងជាពិសេស លោក ធូ វេណូ ប្រធានការិយាល័យស្រាវជ្រាវនៃសាកលវិទ្យាល័យភូមិន្ទភ្នំពេញ ដែលតែងជួយផ្តល់ធាតុចូលចំពោះរាល់ឯកសារការងារ ទាំងផ្នែករដ្ឋបាលនិងទាំងផ្នែកបច្ចេកទេស លោកថែមទាំងជួយសម្រួលរាល់បញ្ហាប្រឈមនានាយ៉ាងជិតដល់បំផុត ដោយមិនប្រកាន់ពេលវេលា និងដោយចិត្តជ្រះថ្លារហូតការសរសេរសៀវភៅនេះទទួលបានជោគជ័យជាស្តាពរ។

មាតិកា

បុព្វកថា i
សេចក្តីបញ្ជាក់ vi
អារម្ភកថា vii
សេចក្តីថ្លែងអំណរគុណ viii
មេរៀនទី១ ការតាងទិន្នន័យ និងមូលដ្ឋានគ្រឹះពិជគណិតប៊ូលីន 1
1. ការតាងទិន្នន័យ 1
1.1. ប្រព័ន្ធគោលពីរ 1
1.2. ប្រព័ន្ធគោលដប់ប្រាំមួយ 3
2. ASCII (American Standard Code for Information Interchange) code 8
3. ពិជគណិតប៊ូលីន (Boolean Algebra) 11
3.1. លក្ខណៈគ្រឹះរបស់ប៊ូលីន 11
3.2. តក្កវិទ្យា Gate 12
4. លក្ខណៈនៃតក្កវិទ្យា 14
5. គោលការណ៍ទ្វេភាពនៅក្នុងពិជគណិតប៊ូលីន (The Principle of duality) 15
6. ទ្រឹស្តីបទនៃពិជគណិតប៊ូលី (Theorem of Boomlean Algebra) 15
7. អនុគមន៍ប៊ូលីន 16
7.1. និយមន័យ 16
7.2. ការបង្រួមនៃអនុគមន៍ (Minization of Boolean Function) 17
7.3. ធាតុបំពេញនៃអនុគមន៍ (Complement of a function) 17
7.4. Minterms និង Maxterms (Minterms and Maxterms) 18
7.5. ផលបូកនៃផលគុណ (Sum_of_Products) 19
7.6. ផលគុណនៃផលបូក (Products_of_Sum) 20
8. ប្រមាណវិធីនព្វន្ធរបស់កំព្យូទ័រ (COMPUTER ARITHMETIC) 20
8.1. Flip_Plop(FF) 20
8.2. ការតាងចំនួនគត់ (Integer Representation) 21
8.3. ចំនួនទសភាគ (Floating_Point) 22
9. ឧបករណ៍អេឡិចត្រូនិច 25
9.1. Led 25
9.2. Resistor 25
9.3. Transistor 25
9.4. Diode 25
9.5. កាប៉ាស៊ីទ័រ 26
9.6. Breadboard 26
9.7. Switch 26
9.8. LCD 16X2 27

9.9. Keypad 27
9.10. Relay 27
9.11. I2C LCD 1602 28
9.12. L298N Motor Driver 28
9.13. Motor 28
9.14. Ultrasonic Sensor 28
9.15. TCRT5000 Sensor 28
សំណួរ និង លំហាត់ 30
មេរៀនទី ២ ការណែនាំអោយស្គាល់ ស្ថាបត្យកម្មកុំព្យូទ័រ 32
1. កុំព្យូទ័រស្ថាបត្យកម្ម(Computer Architecture) 32
2. និយមន័យ Microprocessor (Definition Of Microprocessor) 34
2.1. ការប្រើប្រាស់ Microprocessor 35
2.2. ការអភិវឌ្ឍរបស់ Microprocessor 35
3. CISC និង RISC 39
4. ភាពខុសគ្នារវាង CISC និង RISC 40
5. CPU មានទំនាក់ទំនងជាមួយ RAM និង ROM 40
6. ភាពខុសគ្នារវាង RAM និង ROM 41
សំណួរ និង លំហាត់ 42
មេរៀនទី ៣ X86 Microprocessor **និងពាក្យបញ្ជា** 43
1. 8086 MICROPROCESSOR 43
1.1. General – Purpose Register (AX, BX, CX, DX) 45
1.2. Flags Register 46
1.3. INDEX Register 49
1.4. Instruction Pointer 50
1.5. CS Register 50
1.6. DS Register 51
1.7. ES Register 51
1.8. SS Register 51
2. សំនុំពាក្យបញ្ជា 8086(Instruction Set of 8086) 51
2.1. ប្រមាណវិធីនព្វន្ធ និងតក្កវិទ្យា 52
2.2. ពាក្យបញ្ជាតក្កវិទ្យា 56
2.3. Data Transfer 57
2.4. Stack Operation 62
2.5. ASCII –BCD Conversion 65
2.6. Bit Shift 73

មេរៀនទី ៤ អាសយដ្ឋានម៉ូត ម៉ាស៊ីនកូដ និងកម្មវិធី Debug 75
1. អាសយដ្ឋានម៉ូត(Addressing Mode) 75
1.1. Register AM 75
1.2. Immediate AM 75
1.3. direct AM 76
1.4. Register Indirect AM 77
1.5. Based Relative AM 78
1.6. Indexed Relative AM 78
1.7. Based Indexed Relative AM 79
2. ការរកតំលៃម៉ាស៊ីនកូដ(Find Machine Language) 80
2.1. MOV(Move) Instruction 80
2.2. ADD(Addition) Instruction 82
2.3. DEC(Decrement) Instruction 83
2.4. INC(Increment) Instruction 83
2.5. JNZ(Jump Not Zero) Instruction 84
3. កម្មវិធី Debug 84
3.1. ចូលក្នុងកម្មវិធី Debug 84
3.2. បញ្ជា q(Quit command) 84
3.3. បញ្ជា R (Register command) 84
3.4. បញ្ជា A (Assemble Command) 84
3.5. G (Go Command) 85
3.6. បញ្ជា U(Unassemble Command) 85
3.7. D(Dump) 85
3.8. E(Enter) 86
3.9. T (Trace) 87
3.10. Examining / Altering The Flag Register in Debug 89
3.11. L (Load) 89
3.12. N (Name) 89
4. ការធ្វើការងារខាងក្នុងនៃ 8086(Internal Working of 8086 /8088) 90
5. មុខងារនៃជើង Pin 8086 91
មេរៀនទី ៥ កម្មវិធី X86Asm IDE **និង ការប្រើប្រាស់បញ្ជា** Interrupts 95
1. ការដំឡើង X86Asm IDE 95
2. កម្មវិធីដំបូង (First Program) 97
3. Build និង Run 98
4. ភាសា Assembly ប្រើក្នុងកម្មវិធី X86Asm 98
4.1. ប្រភេទទិន្នន័យ(Data Type) 99

4.2. Invoke macro 100
4.3. ការប្រើបញ្ជា if 101
4.4. ការប្រើបញ្ជា if...else 101
4.5. ការប្រើបញ្ជា if...elseif...else 102
4.6. While Loop 104
4.7. អារេ(Array) 104
4.8. Pointer 105
4.9. JNZ(Jump Not Zero) Instruction 107
4.10. ពាក្យបញ្ជា CMP 108
4.11. JA, JB, JE Instruction 108
4.12. ពាក្យបញ្ជា IN and OUT 109
4.13. ការតេស្តអំពូល Led ក្នុង Emu8086 109
4.14. កម្មវិធី Proteus 109
4.15. ការតេសឧបករណ៍អេឡិចត្រូនិច 110
4.16. SHL and SHR Instruction 114
4.17. ការបង្កើត GUI 115
5. ការប្រើប្រាស់បញ្ជា Interrupts(INT) 119
5.1. BIOS INT 10H និង DOS INT 21H 119
5.2. Keyboard Features 126
5.3. ការប្រើប្រាស់ Macros 133
5.4. FIELS FEATURES 135
5.5. ការប្រើ BIOS និង DOS INTERUPPT ជាមួយនឹងភាសា C 142
5.6. លក្ខណៈនៃការប្រើ MOUSE 144
5.7. DISK STORAGE: 13H DISK FUNCTION 152

មេរៀនទី ៦ ស៊ុតីយូ ៣២ និង ៦៤ ប៊ីត 153
1. ៨០៣៨៦ Register 153
1.1. EAX(Accumulator Register) 154
1.2. Base Register (EBX) 154
1.3. Count Register (ECX) 154
1.4. Data Register(EDX) 155
1.5. Segment Register 155
1.6. Index Register 156
1.7. Flags Register 157
1.8. Pointer Register(EIP, ESP,EBP) 158
2. Addressing Mode 158

3. Math Coprocessor 159
3.1. Signed Integers 159
3.2. លេខគោលពីរក្នុងគោលដប់(Binary_Code Decimal) 160
3.3. ចំនួនទសភាគ(Floating-Point) 160
3.4. ការបំលែងគោលដប់ទៅចំនួនទសភាគ(Floating point) 160
3.5. ការបំលែងពីចំនួនទសភាគ ទៅគោលដប់ 161
4. សំនុំពាក្យបញ្ជា (INSTRUCTION SET) និង FP register 161
4.1. Data Transfer Instructions 161
4.2. Arithmetic Instructions 162
4.3. ពាក្យបញ្ជា FMUL 163
4.4. ពាក្យបញ្ជា FSQRT 164
5. Core 2 Duo, Core i3,i5,i7,i9 and Xeon Microprocessor 165
5.1. ស្ថាបត្យកម្ម(Architecture) 165
5.2. ការប្រើប្រាស់ Register 166
5.3. ទំហំ Data Storage 167
5.4. សំនុំពាក្យបញ្ជា (Instruction Sets) 168
6. x86-64 Assembly 169
6.1. កម្មវិធីដំបូងក្នុងឈីប Corei7 ឬ Corei9 169
6.2. កម្មវិធីបូកលេខពីរចំនួន 170
6.3. ការប្រើប្រាស់ទម្រង់ Procedure 171
7. Addressing Mode 172
7.1. Address និងតម្លៃ 172
7.2. SIMD Instruction 173
8. Operating System 174
9. Calling Conventions 174
សំនួរ និង ចំហាត់ 177
មេរៀនទី ៧ Arduino Microcontroller 179
1. ស្ថាបត្យកម្ម Arduino 179
2. Arduino Board 180
2.1. Arduino Uno R3 180
2.2. Arduino Mega 2560 181
2.3. Arduino Nano 182
2.4. Esp8266 និង NODEMCU 183
3. IDE (integrated development environment) 184
3.1. Arduino IDE 184
3.2. Python IDE 185

3.3. ATMEL STUDIO IDE 185
3.4. UnoArduSim IDE 185
4. រចនាសម្ព័ន្ធ Arduino 186
4.1. អនុគមន៍ setup( ) 186
4.2. អនុគមន៍ loop( ) 186
5. កម្មវិធីក្នុង Edit/View 187
6. ការប្រើប្រាស់សៀរៀលម៉ូនីទ័រ 188
6.1. អនុគមន៍ Serial.begin( ) 188
6.2. Serial.print( ) និង Serial.println( ) 188
6.3. ការប្រើប្រាស់ Arduino Serial Input 190
7. អនុគមន៍ PinMode( ) និង digitalWrite( ) 190
7.1. pinMode ( ) 190
7.2. digitalWrite ( ) 191
8. ការតេសកូដដោយប្រើអំពូល LED 192
8.1. ប្រើកម្មវិធី UnoArduSim 192
8.2. ប្រើកម្មវិធី Proteus 192
9. ប្រើកម្មវិធី Arduino IDE 194
9.1. ឧបករណ៍បញ្ចេញបញ្ចូល 194
9.2. Keypad 196
9.3. IR Infrared Sensor 198
9.4. Motor Driver..... 200
9.5. Buzzer 201
9.6. សេនស័រសីតុណ្ហភាព 202
9.7. Gas Sensor 203
សំណួរ និង លំហាត់ 206
មេរៀនទី ៨ ភាសា Arduino 207
1. Compiler..... 207
2. សញ្ញាប្រមាណវិធី(Operator) 207
3. អថេរ(Variable) 208
3.1. ការប្រើអថេរ 208
3.2. ប្រភេទទិន្នន័យ 208
4. ការដំឡើង Led ចំនួនពីរ 211
4.1. UnoArduSim 211
4.2. ប្រើកម្មវិធី Proteus 212
5. អនុគមន៍(Function) 212
5.1. រចនាសម្ព័ន្ធនៃអនុគមន៍ 213

5.2. analogRead( ) 215
5.3. analogWrite( ) 215
6. ការប្រើបញ្ជា Arduino IF 216
7. ការប្រើ Arduino if ...else statement 218
8. Arduino Loops 221
8.1. ការប្រើប្រាស់ for loop 221
8.2. ការប្រើប្រាស់ while loop 223
9. Arduino Array 225
10. Arduino Class 228
សំណួរ និងលំហាត់ 231
មេរៀនទី ៩ គម្រោងគំរូ 232
1. ឧបករណ៍បញ្ជាភ្លើងដោយឥតប្រើខ្សែប្រើ Bluetooth 232
1.1. ទៅជា BCL(Bluetooth controlled lamp)? 232
1.2. ឧបករណ៍សម្រាប់ធ្វើ Bluetooth control lamp 232
1.3. របៀបតភ្ជាប់ Hardware 233
1.4. ការប្រើ app សម្រាប់ Bluetooth ក្នុងទូរស័ព្ទ 234
1.5. កូដសម្រាប់បញ្ជា Bluetooth control lamp 234
1.6. ការតេសគម្រោងនេះក្នុង Proteus 235
2. Simple Calculator 235
2.1. I2C LCD 1602 ជាអ្វី? 236
2.2. របៀបភ្ជាប់ LCD with I2C to Arduino 236
2.3. Source Code 236
3. ការបង្កើត Radio Controller Car (RC CAR) 239
3.1. តើអ្វីទៅជា RC Car? 239
3.2. គ្រឿងសម្រាប់ធ្វើ RC CAR 239
3.3. របៀបតភ្ជាប់ Hardware 240
3.4. ការប្រើ app សម្រាប់បញ្ជាឡាន ក្នុងទូរស័ព្ទ 241
3.5. កូដសម្រាប់បញ្ជា RC Car (Coding) 241
4. ប្រព័ន្ធស្រោចទឹក និងគ្រប់គ្រងធារាសាស្ត្រស្វ័យប្រវត្តិ 243
4.1. ឧបករណ៍ការដំឡើង 243
4.2. ការតភ្ជាប់ 244
4.3. CODING 244
5. ឌីជីថល CLOCK 244
5.1. ឧបករណ៍ការដំឡើង 245
5.2. ការប្រើ DS3231 ជាមួយនឹង TIME1637 245
5.3. CODING ការប្រើ DS3231 ជាមួយនឹង TIME1637 245

5.4. ការដំឡើង DS3231 ជាមួយនឹង LCD 247
5.5. កម្មវិធីដំណើរការ DS3231 ជាមួយនឹង LCD 247
6. SIM800L ការប្រើស្វ័យប្រវត្តិកម្មក្នុងផ្ទះ 250
6.1. ការដំឡើងឧបករណ៍ 251
6.2. ការតភ្ជាប់ 251
6.3. ជំហានការតេស្ត Sim800L និងប្រើទូរសព្ទ 251
6.4. ការប្រើប្រាស់ AT Command 251
6.5. កម្មវិធីគំរូខាងក្រោមនេះអាចតេស្តជាមួយនឹងផ្នែកវីងជាក់ស្តែង 252
6.6. ឬជាមួយនឹងកម្មវិធី Proteus 253
សំណួរ និង លំហាត់ 256
មេរៀនទី ១០ APP INVENTOR 257
1. របៀបបង្កើតគម្រោង(Project) 257
2. របៀបធ្វើតេស្ត និងដំឡើងកម្មវិធីជំនួយ 258
2.1. ដំឡើង aiStarter 258
2.2. ដំឡើង AI Companion 258
2.3. របៀបរចនា 259
3. របៀបបង្កើតកម្មវិធីដំបូង 263
3.1. របៀបរចនានៅលើ Designer 263
3.2. រចនាសម្ព័ន្ធ 263
3.3. លក្ខណៈបច្ចេកទេស 264
3.4. របៀបបង្កើតប្លុកកូដនៅលើ Block Editor 265
4. របៀបបង្កើតកម្មវិធីបូកលេខពីរចំនួន 267
5. របៀបបង្កើតកម្មវិធីធ្វើប្រមាណវិធី Array 268
6. របៀបបង្កើតកម្មវិធីធ្វើប្រមាណវិធីស្វ័តនៅលើ Interface 269
7. របៀបបង្កើតកម្មវិធីធ្វើការបិទ ឬបើកអំពូល 270
8. បង្កើតឧបករណ៍ប្រើសំលេង 271
9. របៀបវេចខ្ចប់ជា APK 271
10. របៀបបង្កើតកម្មវិធីធ្វើការបញ្ជាទៅលើ RC car 272
10.1. ការបង្កើត Interface 272
10.2. ការប្រើប្លុកកូដ 272
សំណួរ និង លំហាត់ 275
មេរៀនទី ១១ Internet of Thing 276
1. From M2M to IOT 276
2. ស្ថាបត្យកម្មអាយអូធី (IoT Architecture) 276
2.1. ជំហានទី ១: Sensors និង Actuators 277
2.2. ជំហានទី ២: Internet Gateways និង Data Acquisition Systems 277

2.3. ជំហានទី ៣: Pre-processing(Analytics at the Edge) 278
2.4. ជំហានទី ៤: In-depth Analysis in the Cloud or Data Center 278
3. IoT Ecosystem 278
4. IoT protocols 279
5. MQTT 279
6. NodeMCU ESP8266 280
6.1. ការប្រើប្រាស់ GPIO(General Purpose input and output) pin 280
6.2. ការប្រើប្រាស់ Led មួយជាមួយកម្មវិធី Arduino IDE 280
6.3. ការប្រើ OLED with ESP8266 NodeMCU 281
6.4. ការតភ្ជាប់ OLED with ESP8266 NodeMCU 282
7. ការប្រើសេនស័រ DHT11 ជាមួយនឹង NodeMCU ប្រើជា Web Server 284
7.1. ផ្នែករឹង 284
7.2. កម្មវិធី 284
7.3. OUTPUT 286
7.4. ភាសា HTML 287
8. ការគ្រប់គ្រងអំពូលតាមអ៊ិនធើណិត(Lamp Controller by Internet) 288
8.1. សមាសភាគផ្សំ 288
8.2. ការប្រើប្រាស់ឧបករណ៍ និងតភ្ជាប់ 288
8.3. ការតភ្ជាប់ Relay Module និងអំពូល 289
8.4. Internet Server 289
8.5. ការដំឡើងកម្មវិធីទូរស័ព្ទ 289
8.6. ដំណើរការកូដ 291
9. ការប្រើប្រាស់ អអBlynk 292
9.1. ហេតុអ្វីត្រូវប្រើ Blynk? 292
9.2. របៀបបង្កើត GUI ដើម្បីគ្រប់គ្រងចំនួន ៤ អំពូល 292
10. Solar Remote Control Lamp 297
10.1. គ្រឿងផ្សំ 297
10.2. ការតភ្ជាប់ឧបករណ៍ 297
10.3. របៀបបង្កើត APP 298
11. IoT ប្រើប្រាស់ FireBase និង NodeMCU 298
11.1. Why Choose Firebase? 299
11.2. ការបង្កើត app 299
11.3. Programming NodeMCU 299
សំណួរ និង លំហាត់ 203
មេរៀនទី ១២ ភាសា Micropython **ជាមួយនឹង** ESP32 **និង** ESP8266 305
1. កម្មវិធីសម្រាប់សរសេរភាសាPython 305

2. អនុគមន៍ Print 306
3. Compiling និង Interpreting 306
4. អថេរ(Variable) 307
4.1. ការប្រើអថេរ 307
4.2. ប្រភេទទិន្នន័យ 307
5. អនុគមន៍ និង Type Conversion 310
5.1. អនុគមន៍ 310
5.2. ការបង្កើតអនុគមន៍ 310
5.3. Type Conversation 311

6. ការប្រើ If Statement 311
7. ការប្រើ Arduino if...else statement 312
8. Loops 315
8.1. ការប្រើប្រាស់ for loop 315
8.2. ការប្រើប្រាស់ while loop 316
9. អនុគមន៍(Function) 316
9.1. ឈ្មោះអនុគមន៍ 316
9.2. Return Type 316
9.3. Function Body 316
10. Arduino Array និង Class 317
10.1. Array 317
10.2. Class 317
11. ភាសា Micropython 320
11.1. និយមន័យ MicroPython 321
11.2. កម្មវិធីក្នុង MicroPython ដំបូង 321
11.3. ការភ្ជាប់ NodeMCU ទៅនឹងកុំព្យូទ័រ 321
11.4. ការសរសេរកូដ 321
11.5. MicroPython – Boards support 322
11.6. NodeMCU ESP8266 322
11.7. ការប្រើប្រាស់ GPIO(General Purpose input and output) pin 323
11.8. ការប្រើប្រាស់ Led មួយជាមួយកម្មវិធី Arduino និងភាសា Python 323
11.9. ការប្រើ OLED with ESP8266 NodeMCU 324
11.10. ការតភ្ជាប់ OLED with ESP8266 NodeMCU 324
សំណួរ និង លំហាត់ 328
សេចក្តីសង្ខេប 329
ឯកសារពិគ្រោះ 331

មេរៀនទី ១ ការតាងទិន្នន័យ និង មូលដ្ឋានគ្រឹះពីជគណិតប៊ូលីន

ការសិក្សាអំពីប្រព័ន្ធកុំព្យូទ័រ(Computer System) ដែលមានផ្នែករឹង (Hardware) និងផ្នែកទន់ (Software) របស់ស៊ីភីយូ(CPU)ដែលមាននៅក្នុងមេរៀននេះ។ ជាដំបូងត្រូវចេះពីការប្រើប្រព័ន្ធបាប់, ការបំលែងពីគោលដប់ទៅប្រព័ន្ធគោលពីរ ឬគោលដប់ប្រាំមួយ និងស្គាល់ពីការធ្វើប្រមាណវិធីរំកិលលេខប្រព័ន្ធគោលពីរ។ ការប្រើប្រាស់កូដជាស្តែន និង ASCII កូដ ដែលតំណាងឱ្យអក្សរ, លេខ និងនិមិត្តសញ្ញាសម្រាប់ប្រើប្រាស់នៅក្នុង Keyboard និង អង្គចងចាំ(RAM) សម្រាប់យកវាទៅធ្វើប្រមាណវិធី និងសរសេរកូដនៅក្នុងភាសា Assembly, Arduino និង Python។ ចំពោះប្រមាណវិធីតក្កវិទ្យា និង Gate ក៏រឹតតែសំខាន់ទៅទៀតដោយយកទៅកសាងជាស្មៀគ្វីនៃផ្នែករឹងកុំព្យូទ័រ, ឧបករណ៍អេឡិចត្រូនិច និងឧបករណ៍ឆ្លាតវៃផ្សេងៗទៀតមុននឹងបង្កើតឧបករណ៍ជាក់លាក់មួយ។

1. ការតាងទិន្នន័យ

1.2. ប្រព័ន្ធគោលពីរ

នៅក្នុងប្រព័ន្ធគោលពីរគឺជាប្រព័ន្ធមួយមានដែលមានប្រើលេខតែពីរគត់គឺលេខ 1 និង 0 ហើយមានលេខ ២ ជាគោលនៃប្រព័ន្ធ។ លេខធំបំផុតនៃប្រព័ន្ធនេះគឺលេខ ១ ដែលតូចបន្ទាប់ពីលេខគោល។ កុំព្យូទ័រប្រើប្រព័ន្ធគោលពីរព្រោះដូចជាស៊ីភីយូ, អង្គចងចាំ(RAM) និងឧបករណ៍បញ្ចេញឬបញ្ចូលផ្សេងៗមួយចំនួនទៀតដោយសៀគ្វីរបស់វាបង្កើតពីត្រង់ស៊ីស្ទ័រ (Transistor) ដែលប្រើជាកុងតាក់បើក(Switch ON) តំណាងឱ្យលេខ 1 និង កុងតាក់បិត (Switch OFF) តំណាងឱ្យលេខ 0។ 0 និង ១ ច្រើនផ្សំចូលបង្កើតបានជាទិន្នន័យ និងពាក្យបញ្ជាសម្រាប់ប្រើប្រាស់នៅក្នុងកុំព្យូទ័រ។

ឧទាហរណ៍: ពេលចុចអក្សរ A ធំនៅលើ Keyboard គិតជា ASCII : 0100 0001b មានន័យប្រើលេខក្នុងប្រព័ន្ធគោលពីរដោយតួលេខនិមួយៗហៅថាប៊ីត(Bit) និង ប្រើឧបករណ៍ជាក់ស្តែងគឺប្រើចំនួន 8 ត្រង់ស៊ីស្ទ័រដើម្បីតាងឱ្យលេខនេះនៅក្នុងអង្គចងចាំ។ ការធ្វើប្រមាណវិធីក្នុងស៊ីភីយូ និងសរសេរកូដនៅក្នុងភាសាដំបូង Assembly ត្រូវគិតពីខ្នាតដូចខាងក្រោមនេះ៖

- bit =1 or bit = 0
- nibble = 4 bits
- byte = 8 bits
- Word = 16 bits = 2 bytes
- Double word = 32 bits = 4 bytes
- Quad Word = 64 bits = 8 bytes

- ចំពោះលេខ 32 bits

Low Bit ហៅថា LSB (bit 0)

High Bit ហៅថា MSB (bit 31)

16 bits

Low Bit ហៅថា LSB (bit 0)

High Bit ហៅថា MSB (bit 15)

- ចំពោះលេខ 8 bitsLow Bit ហៅថា LSB (bit0) និង High Bit ហៅថា MSB

1.1.1. ការបូកលេខប្រព័ន្ធគោលពីរ

ការបូកលេខក្នុងប្រព័ន្ធគោលពីរគឺ មានលក្ខណៈប្រហាក់ប្រហែលទៅនឹងការបូកលេខនៅក្នុងប្រព័ន្ធគោលដប់ដែរ។ ហើយដើម្បីបូកនេះត្រូវចេះពីការប្រើប្រាស់តារាងគ្រឹះដូចខាងក្រោមដែលមានបួនករណីសំខាន់ៗ

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 មានគ្រាទុកមួយ ឬ 1 + 1 = 10

ចំណាំ 0 + 0 + 0 = 0

1 + 1 + 1 = 1 មានគ្រាទុកមួយ 1

☺. **ឧទាហរណ៍:** បូកលេខ Binary: (a).1011 + 0011

(b).10110001 +11001111

	1011	10000011
+	0011	01101110
	1110	11110001

1.1.2. ការដកលេខប្រព័ន្ធគោលពីរ

ការដកលេខក្នុងប្រព័ន្ធគោលពីរក៏ប្រហាក់ប្រហែល នឹងការដកលេខក្នុងប្រព័ន្ធគោលដប់ដែរ។ ដោយខុសគ្នាត្រង់ពេលដក ១ ពីលេខ ០ ឬ ១ សងនឹង ០ ចាំបាច់ត្រូវខ្ចីលេខ ១ ពីលេខលំដាប់ខ្ពស់បន្ទាប់ហើយលេខនោះត្រូវបានកាត់បន្ថយដោយ ១ (ឬ ១ ត្រូវបានបន្ថែមទៅប៊ីតបន្ទាប់ subtrahend) ហើយនៅសល់គឺ ១។ តារាងខាងក្រោមប្រើសម្រាប់ធ្វើប្រមាណវិធីដកលេខ៖

0 – 0 = 0	-	0	1
1 – 0 = 1	0	0	1
1 – 1 = 0	1	1	0
0 – 1 = 1 ខ្ចីលេខ 1			

☺. **ឧទាហរណ៍:** ធ្វើប្រមាណវិធីដកលេខ Binary ខាងក្រោម៖

	1010	10011110 10000011	= 16 bits = 1 word
-	0011	01100101 00010111	= 16 bits = 1 word
	01 1 1	00111001 01101100	= 16 bits = 1 word

1.1.3. ការគុណលេខប្រព័ន្ធគោលពីរ

តារាងមេគុណបង្ហាញដូចខាងក្រោមជាមូលដ្ឋានគ្រឹះប្រើសម្រាប់ធ្វើប្រមាណវិធីគុណ៖

0 * 0 = 0	0 * 1 = 0
1 * 0 = 0	1 * 1 = 1

☺. **ឧទាហរណ៍:** គុណលេខ Binary 1011 * 0011 :

ចម្លើយ

	1100 multiplicand
	*101 multiplicator
+	1100
	0000
	1100
	111100 product

1.1.4. ការចែកលេខប្រព័ន្ធគោលពីរ

នៅក្នុងគោលដប់ការចែកលេខជាមួយនឹងលេខ ០ គឺគ្មានន័យ។ដូចនេះនៅក្នុងការចែកលេខនៅក្នុងប្រព័ន្ធគោលពីរក៏ដូចគ្នាដែរ។ ដូច្នេះនៅក្នុងប្រមាណវិធីនេះមានតែពីរករណីគឺ៖

0 / 1 = 0 និង 1 / 1 = 1

1.1.5. ការបំលែងពីប្រព័ន្ធគោលដប់ទៅគោលពីរ

មានវិធីជាច្រើនយ៉ាងក្នុងការបំលែងពីប្រព័ន្ធគោលដប់ទៅប្រព័ន្ធគោលពីរ។ វិធីដែលមានជាបន្តបន្ទាប់ខាងក្រោមប្រើសម្រាប់ធ្វើការបំលែងពីប្រព័ន្ធគោលដប់ទៅប្រព័ន្ធគោលពីរ៖

របៀបទី ១: របៀបបំលែងពីប្រព័ន្ធគោលដប់ទៅគោលពីរប្រើប្រមាណវិធីចែក

1. ធ្វើប្រមាណវិធីចែកដោយយកលេខប្រព័ន្ធគោលដប់ ចែកនឹងលេខ 2
2. ធ្វើប្រមាណវិធីដូចនៅជំហានទី១ ជាបន្តបន្ទាប់រហូតដល់លទ្ធផលចែកស្មើនឹងលេខ 0 ទើបឈប់ធ្វើប្រមាណវិធីចែក
3. កំណត់លេខលទ្ធផលបានមកពីការរាប់សំណល់នៃប្រមាណវិធីចែកដោយគិតសំណល់ពីខាងក្រោមទៅខាងលើវិញ។

របៀបទី ២: របៀបបំលែងពីប្រព័ន្ធគោលដប់ទៅគោលពីរប្រើប្រមាណវិធីដក

1. ការធ្វើប្រមាណវិធីដកដោយយកលេខក្នុងប្រព័ន្ធគោលដប់ដកនឹងលេខ 2^n ដោយ 2^n ជាចំនួនលេខនៅក្នុងប្រព័ន្ធគោលដប់មានតម្លៃស្មើនឹងតួដែលឱ្យ ឬតូចជាងតួដែលឱ្យបន្ទាប់។
2. ក្នុងករណី 2 ស្វ័យគុណណាមានប្រមាណវិធីដកប្រមាណវិធីនោះកំណត់លេខ1 ហើយរាល់ 2 ស្វ័យគុណណាគ្មានប្រមាណវិធីដកនោះកំណត់លេខ 0។
3. កំណត់បានលេខគោលពីរដោយរាប់លេខ 1 ឬ លេខ 0 ពីស្វ័យគុណធំទៅស្វ័យគុណតូច

ចូរធ្វើការបំលែងលេខគោលដប់ ១៥៦ ទៅគោលពីរតាមរបៀបទី ២

128	64	32	16	8	4	2	1
1	0	0	1	1	1	0	0

156 – 128 = 28 ពេល 156 ដកនឹង 2^7=128 គិត 1

28 – 16 = 12 ពេល 28 ដកនឹង 2^4=16 គិត 1

12 – 8 = 4 ពេល 12 ដកនឹង 2^2=4 គិត 1

4 – 4 = 0 ពេល 4 ដកនឹង 2^2=4 គិត 1។ ដូច្នេះលេខ 156 = 10011100b

1.1.6. ការបំលែងពីប្រព័ន្ធគោលពីរទៅគោលដប់

ដើម្បីបំលែងលេខពីប្រព័ន្ធគោលពីរ ទៅគោលដប់ត្រូវអនុវត្តតាមរូបមន្តខាងក្រោម៖

$Dec.=b_0.2^0 + b_1.2^1 + b_3.2^3+...+b_n.2^n$

ដោយ b_0, b_1, …, bn ជាតួនៃលេខប្រព័ន្ធគោលពីរដោយរាប់ពីខាងស្តាំទៅខាងឆ្វេង

☺.**ឧទាហរណ៍**: បំលែងលេខគោលពីរខាងក្រោមទៅជាលេខគោលដប់

a. 10001b ចម្លើយគឺ 11001b = $1x2^0 +0x2^1 +0x2^2 + 0x2^3 + 1x2^4 =17$
b. 111010b= $0x2^0 +1x2^1 +0x2^2 + 1x2^3 + 1x2^4+1x2^5 =$
c. 0110011b=

1.2. ប្រព័ន្ធគោលដប់ប្រាំមួយ

ប្រព័ន្ធគោលដប់ប្រាំមួយគឺជាប្រព័ន្ធមួយដែលមានប្រើលេខចំនួន 16 គួគឺ មានពីលេខ 0, 1, 2, 3, 4 ,5 ,6 ,7 ,8 ,9 ,A ,B ,C ,D ,E និង F ហើយប្រើលេខ 16 ជាគោលនៃប្រព័ន្ធ។

វាត្រូវបានប្រើសម្រាប់តាងឱ្យលេខប្រព័ន្ធគោលពីរសម្រាប់ធ្វើប្រមាណវិធីផ្សេងៗ នៅក្នុងមេរៀនកុំព្យូទ័រ ស្ថាបត្យកម្ម, ប្រើប្រាស់នៅក្នុងភាសា Assembly និង ភាសាម៉ាស៊ីនកូដដែលប្រើនៅក្នុងអង្គចងចាំ និងឱ្យ

ស៊ីកីយូដំណើរការ។ ព្រោះការប្រើប្រព័ន្ធនេះវាខ្លី និងឱ្យយើងងាយយល់ជាជាងការប្រើប្រព័ន្ធគោលពីរ ពិបាកក្នុងការធ្វើប្រមាណវិធី និង ការចងចាំ។ នៅក្នុងកម្មវិធីសិប្បនិម្មិតមួយចំនួនធំប្រើប្រព័ន្ធនេះ ដូចជា Proteus ឬ Simulator x86 ដើម្បីប្រាប់ស៊ីកីយូឱ្យដំណើរការ។

ប្រើអក្សរ "H" នៅពីខាងក្រោយលេខ និង 0x នៅពីខាងមុខលេខសំគាល់ថាលេខនោះជាលេខក្នុងប្រព័ន្ធគោលដប់ប្រាំមួយពេលប្រើវាក្នុងភាសាផ្សេងៗ។

☺.**ឧទាហរណ៍:**

1. 0x87AE ដោយ 0x តំណាងឱ្យលេខប្រព័ន្ធគោលដប់ប្រាំមួយ និង លេខនៅខាងក្រោយវាគឺ87AE ជាលេខដែលត្រូវប្រើប្រាស់សម្រាប់ធ្វើប្រមាណវិធី និងសរសេរកូដក្នុងភាសា Assembly និងភាសា python។

2. 23acH ដោយអក្សរ "H" គឺ មានអត្ថន័យដូចការសរសេរ 0x ដែរនិងជាលេខ 23ac ជាលេខនៅក្នុងប្រព័ន្ធដប់ប្រាំមួយ។

3. ACD123H លេខក្នុងប្រព័ន្ធគោលដប់ប្រាំមួយអាចជាអក្សរធំ ឬ អក្សរតូចក៏បាន

4. លក្ខណៈប្រព័ន្ធគោល ១៦គឺ មួយតួលេខស្មើនឹង បួនតួ(4 bits) នៃលេខគោល ២។

☺.**ឧទាហរណ៍:** 14AFh = 00010100 10101111b និង នៅក្នុងភាសា Assembly បើយើងសរសេរ៖

Number1 dd 10000h ; Number1 = 10000 បញ្ជាក់ថាលេខប្រព័ន្ធគោលដប់ប្រាំមួយ

Number2 dd 10000 ; Number2 = 10000 បញ្ជាក់ថាលេខប្រព័ន្ធគោលដប់

Number3 dd 10000b ; Number1 = 10000 បញ្ជាក់ថាលេខប្រព័ន្ធគោលពីរ

1.2.1. ការបូកលេខប្រព័ន្ធគោលដប់ប្រាំមួយ

ការបូកលេខនៅក្នុងប្រព័ន្ធគោលដប់ប្រាំមួយគឺត្រូវអនុវត្តតាមតារាងលេខគ្រឹះដូចខាងក្រោម ដោយយកលេខនៅជួរឈរនិមួយៗបូកជាមួយនឹងជួរដេកនិមួយៗហើយលទ្ធផលបង្ហាញតាមជួរដេកតាមលំដាប់លំដោយ៖

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B

D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

☺. **ឧទាហរណ៍:** បូកលេខក្នុងប្រព័ន្ធគោលដប់ប្រាំមួយដូចខាងក្រោម៖

a. 35AH + 93FH
b. BCD12H + 89FEDH
c. 5612345678234H + ABCD12444434H
d. 34566AEDDFFH + 2345DDD67H

ចម្លើយ

a. 35AH
+ 93FH
C99H

b. BCD12H
+ 89FEDH
146CFFH

d. 34566AEDDFFH
+ 2345DDD67H
568AC1CBB5FH

c . 5612345678234H
+ ABCD12444434H
60CF057ABC668H

1.2.2. ការដកលេខប្រព័ន្ធគោលដប់ប្រាំមួយ

ចំពោះការធ្វើប្រមាណវិធីដកលេខត្រូវមើលតាមរបៀបដកលេខតាមតារាងគ្រឹះ ដោយលេខជា Minuend Subtrahend តាមជួរឈរ ដកនឹង លេខ Subtrahend តាមជួរដេក និង លទ្ធផលជាក់តាមជួរដេករៀងដូចដែលមាននៅក្នុងតារាងដូចខាងក្រោម៖

Subtrahend

-		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
M	0	0	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1
I	1	1	0	F	E	D	C	B	A	9	8	7	6	5	4	3	2
n	2	2	1	0	F	E	D	C	B	A	9	8	7	6	5	4	3
u	3	3	2	1	0	F	E	D	C	B	A	9	8	7	6	5	4
e	4	4	3	2	1	0	F	E	D	C	B	A	9	8	7	6	5
n	5	5	4	3	2	1	0	F	E	D	C	B	A	9	8	7	6
d	6	6	5	4	3	2	1	0	F	E	D	C	B	A	9	8	7
	7	7	6	5	4	3	2	1	0	F	E	D	C	B	A	9	8
	8	8	7	6	5	4	3	2	1	0	F	E	D	C	B	A	9
	9	9	8	7	6	5	4	3	2	1	0	F	E	D	C	B	A
	A	A	9	8	7	6	5	4	3	2	1	0	F	E	D	C	B
	C	B	A	9	8	7	6	5	4	3	2	1	0	F	E	D	C
	C	C	B	A	9	8	7	6	5	4	3	2	1	0	F	E	D
	D	D	C	B	A	9	8	7	6	5	4	3	2	1	0	F	E
	E	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	F
	F	F	E	E	C	B	A	9	8	7	6	5	4	3	2	1	0

☺.**ឧទាហរណ៍:** ចូរដកលេខ Hexadecimal តាមករណីដូចខាងក្រោម៖

1. 95AH – 43FH
2. 8AC8H – 567AH
3. F123456H – 129AFEBDH

<u>ចម្លើយ</u>

a. 95AH
- 43FH
51BH

b.8AC8H
- 567AH
E142H

c. F1231456H
-129AFEBDH

1.2.3. ការគុណលេខប្រព័ន្ធគោលដប់ប្រាំមួយ

ការធ្វើប្រមាណវិធីគុណនៅក្នុងប្រព័ន្ធគោល ១៦ គឺ ធ្វើដូចនៅក្នុងប្រព័ន្ធគោល ១០ ដែរ។ដោយមុននឹងគុណលេខយើងត្រូវចេះពីមេលេខពី ២ ដល់លេខ ៩។ ចំណែកឯការគុណលេខក្នុងគោល ១៦ ត្រូវមើលតាមតារាងមេគុណខាងក្រោមដែលជួយឱ្យយើងអាចធ្វើការគុណលេខបានក្នុងប្រព័ន្ធនេះ៖

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	1A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	AB
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	86	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

☺. **ឧទាហរណ៍:** ការគុណលេខ Hexadecimal ខាងក្រោម៖

a. 5AH * 3FH = 1626H

b. 31DH * 19FH = 50C03H

1.2.4. ការចែកលេខប្រព័ន្ធគោលដប់ប្រាំមួយ

ការធ្វើប្រមាណវិធីចែកលេខនៅក្នុងប្រព័ន្ធគោល ១៦ គឺធ្វើដូចការចែកលេខនៅក្នុងប្រព័ន្ធគោលដប់ដែរ។ ដោយការចែកលេខដាមួយនឹង ០ គឺ គ្មានន័យ។ ហើយការចែកនៅក្នុងប្រព័ន្ធគោល ១៦ ត្រូវចេះពីរបៀបគុណលេខនិងចែកដូចបានរៀបរាប់នៅខាងលើ។

1.2.5. ការបំលែងពីប្រព័ន្ធគោល ១០ ទៅប្រព័ន្ធគោល ១៦

ដើម្បីបំលែងពីប្រព័ន្ធគោល ១០ ទៅប្រព័ន្ធគោល ១៦ ត្រូវអនុវត្តតាមវិធីចែកដូចដែលមាននៅក្នុងជំហានបន្តបន្ទាប់ខាងក្រោមនេះ៖

1. យកលេខគោល ១០ ចែកនឹង លេខ 16 ជាគោលនៃប្រព័ន្ធ
2. ធ្វើប្រមាណវិធីដូចជំហានទី១ ជាបន្តបន្ទាប់រហូតផលចែកស្មើ 0 ទើបឈប់
3. លទ្ធផលបានមកពីធ្វើការរាប់សំណល់នៃប្រមាណវិធីចែកដោយគិតសំណល់ពីខាងក្រោមទៅខាងលើ និងសំណល់អាចមានពី 0 ដល់ F។

ចម្លើយ: ដូច្នេះលេខប្រព័ន្ធគោលដប់ 10767 = 2A0Fh

1.2.6. ការបំលែងពីប្រព័ន្ធគោល ១៦ ទៅប្រព័ន្ធគោល ១០

ដើម្បីធ្វើការបំលែងលេខពីប្រព័ន្ធគោល ១៦ ទៅគោល ២ ត្រូវអនុវត្តតាមរូបមន្តខាងក្រោម៖

$d_0. 16^0 + d_1.16^1 + d_2. 16^2 + d_3.16^3 + \ldots + b_n16^n = Dec$

- ដោយ d_0 , d_1 ,..., dn ជាតួនៃលេខគោល១៦ដោយរាប់ពីខាងស្តាំទៅខាងឆ្វេង
- **ឧទាហរណ៍:** ចូរធ្វើការបំលែងលេខគោល ១៦ គឺលេខ 2A0Fh ទៅប្រព័ន្ធគោលដប់

  15 * 16^0 + 0 * 16^1 + 10 * 16^2 + 2 * 16^3 = **10767 ដោយលេខ ១០៧៦៧ ជាលេខក្នុងប្រព័ន្ធគោលដប់**
- **ចំណាំ: A = 10 , B=11, C=12 , D=13, E =14 និង F=15 ពីគោលពីរទៅគោល ១៦**

1.2.7. ការបំលែងពីប្រព័ន្ធ២ទៅប្រព័ន្ធគោល ១៦ និង ត្រលប់មកវិញ

ដើម្បីធ្វើប្រមាណវិធីបំលែងគោល ២ ទៅប្រព័ន្ធគោល ១៦ និង ដើម្បីបំលែងគោល ១៦ ទៅលេខគោល២ វិញត្រូវអនុវត្តតាមតារាងគ្រឹះដូចខាងក្រោមហើយ ការធ្វើបំលែងនេះមានសារៈសំខាន់ក្នុង ការធ្វើប្រមាណវិធីតក្តវិទ្យា, រកតម្លៃកូដ Ascii ដែលផ្ទុកក្នុង Memory, រកតម្លៃម៉ាស៊ីនកូដនៃពាក្យបញ្ជារបស់ CPU និមួយៗនិងតម្លៃនៃ Hex File ៖

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B

12	1100	C
13	1101	D
14	1110	E
15	1111	F

☺. **ឧទាហរណ៍:** ចូរធ្វើការបំលែងលេខពីប្រព័ន្ធគោល ២ ទៅគោល ១៦ និង ត្រលប់មកវិញតាមប្រមាណវិធីដូចខាងក្រោម៖

1. AB12H = 10101011 00010010b
2. 1010110010000010b = AC82h

ចំណាំ: 2 *តួលេខ Hexa ស្មើ* 1 *byte នៃលេខ Binary*, **3DH =0011 1101= 1 byte = 8 bits**

៤ តួលេខ Hexa ស្មើ ២ *byte នៃលេខ Binary* ឬ ស្មើនឹង word

8 តួលេខ Hexa ស្មើ 4 byte នៃលេខ Binary ឬ ស្មើនឹង double word

2. ASCII (American Standard Code for Information Interchange) code

ការតាងទិន្នន័យសម្រាប់ប្រើប្រាស់នៅក្នុងអង្គចងចាំ(Memory) និង គ្រប់ការធ្វើប្រមាណវិធីទាំងអស់នៅក្នុងប្រព័ន្ធកុំព្យូទ័រគឺ ប្រើលេខគោលពីរគឺ 1 និង 0។ ដូច្នេះលេខ 1 ច្រើន និង 0 ច្រើនផ្សំចូលគ្នាបង្កើតបានទំរង់ ៨ ប៊ីតសម្រាប់តាងឱ្យអក្សរធំ, អក្សរតូច, លេខ និងនិមិ្តតសញ្ញាផ្សេងៗទៀត។

☺.**ឧទាហរណ៍**: បើយើងសរសេរ "abcDF123*" នៅក្នុងកម្មវិធីមួយនោះពេលផ្ទុកនៅក្នុង Memory ហើយបើគិតជាប្រព័ន្ធគោលដប់ប្រាំមួយគឺ 61H, 62H, 63H, 44H, 45H, 31h, 32h, 33h, 2Ah = 9 bytes ឬស្មើ ៩ ទីតាំងនៃ Memory។

កូដប្រើនៅលើ **Keyboard និង Memory** លេខកូដសម្រាប់តំណាងឱ្យអក្សរ លេខ និមិ្តតសញ្ញាផ្សេងៗ ប្រើប្រាស់នៅលើ Keyboard ហៅថាស្កែនកូដ(Scan Code) និង លេខកូដប្រើប្រាស់នៅក្នុង Memory(RAM) ហៅថា ASCII កូដ។ ឧទាហរណ៍ពេលយើងចុចអក្សរ A នៅលើ Keyboard គិតជាកូដស្មើនឹង 1Eh គិតក្នុងប្រព័ន្ធគោលដប់ប្រាំមួយហៅថាស្កែនកូដ និង កូដនេះពេលផ្ទុកនៅក្នុងអង្គចងចាំគិតជាកូដ 41h វិញហៅថា ASCII កូដ។ ហើយបើយើងចុច Shift+ Key A គិតជាស្កែនកូដនៅដដែលតែពេលផ្ទុកក្នុងគិតជាលេខ 41h វិញ។ ចូរមើលតារាងខាងក្រោមសម្រាប់ប្រើប្រាស់កូដពេលចុច Key, Shift, Alt និង Ctrl:

LEETTES	NOMAL		SHIFT		CTRL	ALT		
a and *A*	*1E*	61	1E	*41*	1E	01	1E	00
b and B	30	62	30	42	30	02	30	00
c and C	2E	63	2E	43	2E	03	2E	00
d and D	20	64	20	44	20	04	20	00
e and E	12	65	12	45	12	05	15	00
f and F	21	66	21	46	21	06	12	00
g and G	22	67	22	47	22	07	22	00
h and H	23	68	23	48	23	08	23	00
i and I	17	69	17	49	17	09	17	00
j and J	24	6A	24	4A	24	0A	24	00
k and K	25	6B	25	4B	25	0B	25	00

l and L	26	6C	26	4C	26	0C	26	00
m and M	32	6D	32	4D	27	0D	32	00
n and N	31	6E	31	4E	31	0E	31	00
o and O	18	6F	18	4F	18	0F	18	00
p and P	19	70	19	50	19	10	19	00
q and Q	10	71	10	51	10	11	10	00
r and R	13	72	13	52	13	12	13	00
s and S	1F	73	1F	53	1F	13	1F	00
t and T	14	74	14	54	14	14	14	00
u and U	16	75	16	55	16	15	16	00
v and V	2F	76	2F	56	2F	16	2F	00
w and W	11	77	11	57	11	17	11	00
x and X	2D	78	2D	58	2D	18	2D	00
y and Y	15	79	15	59	15	19	15	00
z and Z	2C	7A	2C	5C	2C	1A	2C	00
Spacebar	39	20	39	20	39	20	39	20
FUNCTION								
KEYS	NOMAL		SHIFT		CTRL		ALT	
F1	3B	00	54	00	5E	00	68	00
F2	3C	00	55	00	5F	00	69	00
F3	3D	00	56	00	60	00	6A	00
F4	3E	00	57	00	61	00	6B	00
F5	3F	00	58	00	62	00	6D	00
F6	40	00	59	00	63	00	6E	00
F7	41	00	5A	00	64	00	6E	00
F8	42	00	5B	00	65	00	6F	00
F9	43	00	5C	00	66	00	70	00
F10	44	00	5D	00	67	00	71	00
F11	85	00	87	00	89	00	8B	00
F12	86	00	88	00	8A	00	8C	00
KEYPAD	NOMAL		SHIFT		CTRL		ALT	
Ins and 0	52	00	52	30	92	00		
End and 1	4F	00	4F	31	75	00	00	01
DnArrow and 2	50	00	50	32	91	00	00	02
PgDn and 3	51	00	51	33	76	00	00	03
LtArrow and 4	4B	00	4B	34	73	00	00	04

5(keypad)	4C	00	4C	35	8F	00	00	05
RtArrow and 6	4D	00	4D	36	74	00	00	06
Home and 7	47	00	47	37	77	00	00	07
UpArrow and 8	48	00	48	38	8D	00	00	08
Pgup and 9	49	00	49	39	84	00	00	09
+(gry key)	4E	2B	4E	2B	90	00	4E	00
-(gry key)	4A	2D	4A	2D	8E	00	4A	00
Del and	53	00	53	2E	93	00		
*(gry key)	37	2A	37	2A	96	00	37	00
TOP ROW	NORMAL		SHIFT		CTRL		ALT	
`and~	29	60	29	7E			29	00
1 and !	02	31	02	21			78	00
2 and @	03	32	03	40	03	00	79	00
3 and #	04	33	04	23			7A	00
4 and $	05	34	05	24			7B	00
5 and %	06	35	06	25			7C	00
6 and ^	07	36	07	5E	07	1E	7D	00
7 and &	08	37	08	26			7E	00
8 and *	09	38	09	2A			7F	00
9 and (	0A	39	0A	38			80	00
0 and)	0B	30	0B	29			81	00
- and _	0C	2D	0C	5F	0C	1F	82	00
= and +	0D	3D	0D	2B			83	00
OPERRATION KEYS	NORMAL		SHIFT		CTRL		ALT	
Esc	01	1B	01	1B	01	1B	01	00
Backspace	0E	08	0E	08	0E	7E	0E	00
Tab	0F	09	0F	94	00	A5	A5	00
Enter	1C	0D	1C	0D	1C	0A	1C	00
PUNCTUATION	NOMAL		SHIFT		CTRL		ALT	
[and {	1A	5B	1A	7B	1A	1B	1A	00
] and }	1B	5D	1B	7D	1B	1D	1B	00
; and :	27	3B	27	3A			27	00
` and ~	28	27	28	22			28	00
\ and |	2B	5C	2B	7C	2B	1C	2B	00
, and <	33	2C	33	3C			33	00

. and >	34	2E	34	3E			34	00
/ and ?	35	2F	35	3F			35	00
KEY	NORMAL		SHIFT		CTRL		ALT	
Slash(/)	E0	2F	E0	2F	95	00	A4	00
Enter	E0	0D	E0	0D	E0	0A	A6	00
Home	47	E0	47	E0	77	E0	97	00
End	4F	E0	4F	E0	75	E0	9F	00
PageUp	49	E0	49	E0	84	E0	99	00
PageDown	51	E0	51	E0	76	E0	A1	00
DownArrow	50	E0	50	E0	91	E0	A0	00
LeftArrow	4B	E0	4B	E0	73	E0	9B	00
RightArrow	4D	E0	4D	E0	74	E0	9D	00
UpArrow	48	E0	48	E0	8D	E0	98	00
Ins	52	E0	52	E0	92	E0	A2	00
Del	53	E0	53	E0	93	E0	A3	00
CapLock	3A		Shift(Right)			36		

3. ពិជគណិតប៊ូលីន (Boolean Algebra)

ពិជគណិតប៊ូលីនជាប្រព័ន្ធមួយប្រើតួលេខតែពីរគត់គឺលេខ ១ និងលេខ ០ និងត្រូវបានគេយកទ្រឹស្តី និង ប្រមាណវិធីរបស់វាទៅបង្កើតជាស្វៀគ្វីសម្រាប់ធ្វើការផ្ទុកទិន្នន័យក្នុងអង្គចងចាំ(Memory) និងធ្វើប្រមាណវិធី តក្កវិទ្យា និង ប្រមាណលេខគណិតនៅក្នុងកុំព្យូទ័រ។

3.1. លក្ខណៈគ្រឹះរបស់ប៊ូលីន

3.1.1. ការបូកលេខតក្កវិទ្យា(Logical Addition)

នៅក្នុងពិជគណិតប៊ូលីនគេប្រើសញ្ញា "+" តំណាងឱ្យការបូកលេខតក្កវិទ្យា។ លទ្ធផលស្មើ ១ កាលណាការបញ្ចូលតម្លៃយ៉ាងតិចស្មើនឹង ១។ ហើយលទ្ធផលស្មើនឹង ០ វិញនៅពេលបញ្ចូលតម្លៃ ទាំងពីរស្មើ ០។ កន្សោម A + B = C ដោយ A , B ជាអថេរមានតម្លៃលេខពីរគឺ ០ និង ១ និង ពាក្យ សម្រាប់បញ្ចូលតម្លៃនិង Output សម្រាប់បញ្ចេញតម្លៃ។

Inputs		Output
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

☺ **ឧទាហរណ៍:** សន្មត់ A = 10100011 , B=11110111 ដោយលេខនៅអថេរ A និង B មួយៗមានទំហំ ៨ ប៊ីត។

A = 1010 0011 ជាតម្លៃបញ្ចូលទី ១

+B = 1111 0111 ជាតម្លៃបញ្ចូលទី ២

C = 1111 0111 ជាតម្លៃលទ្ធផល

3.1.2. ការគុណលេខតក្កវិទ្យា (Logical Multiplication)

នៅក្នុងពិជគណិតប៊ូលីនគេប្រើសញ្ញា "." ជាផលគុណតក្កវិទ្យា។ លទ្ធផលស្មើ ១ កាលណាការបញ្ចូលតម្លៃទាំងពីរស្មើ ១។ ហើយលទ្ធផលស្មើនឹង ០ ក្នុងករណីក្រៅពីនេះ។

កន្សោម A . B = C ដោយ A , B ជាអថេរមានតម្លៃលេខពីរគឺលេខ ០ និង ១

Inputs		Output
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

☺ **ឧទាហរណ៍:** A = 1101 0001 ជាតម្លៃបញ្ចូលទី ១

B = 0011 0111 ជាតម្លៃបញ្ចូលទី ២

C = 0001 0001 ជាតម្លៃលទ្ធផល

3.1.3. ជាតុបំពេញ (Complements)

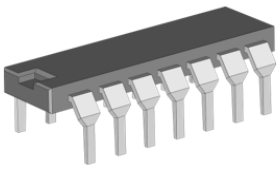
សញ្ញា "-" ឬអក្សរ "c" ប្រើសម្រាប់ជាតុបំពេញ ដោយនៅក្នុងតក្កវិទ្យាប្រើ ¬។ ប្រមាណវិធីជាតុបំពេញ(Complements)គឺ ជាការប្តូរលេខពី ១ ទៅ ០ និង ០ ទៅ ១។

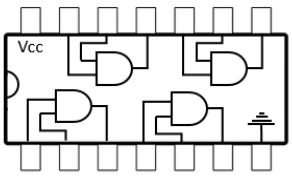
A	A^c
0	1
1	0

☺ **ឧទាហរណ៍:** A =1010001 ចម្លើយ: A^c = 0101110

3.2. **តក្កវិទ្យា** Gate

ប្រមាណវិធីទាំងអស់នៅក្នុងកុំព្យូទ័រ និងឧបករណ៍អេឡិចត្រូនិចគឺ អនុវត្តដោយន័យផ្សំចូលគ្នាទៅតាមនិមិត្តសញ្ញាដែលតភ្ជាប់គ្នាដោយកំណត់បានជាប្តកសៀកគ្វីជាច្រើនដែលហៅថាតក្កវិទ្យា។ ម៉្យាងទៀតវាជាអេឡិចត្រូនិចសៀកគ្វីដែលប្រតិបត្តិឱ្យបានពីការបញ្ចូល(Input)ច្រើនបញ្ចូលគ្នាដើម្បីឱ្យបានលទ្ធផល(Outputs) មួយ។

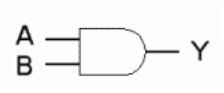




ចំពោះប្រភេទ IC, Microchip, Microprocessor និង Microcontroller សៀគ្វីរបស់វាបង្កើតពីដោយប្រភេទផ្សេងៗនៃ Logic Gate ដើម្បីធ្វើប្រមាណវិធីតក្កវិទ្យា (OR, AND, NOT) និង លេខនព្វន្ត និងរួមទាំងការផ្ទុកទិន្នន័យ។ មានប្រភេទ Gate ជាច្រើនដែលយើងត្រូវសិក្សាដែលជាគ្រឹះសម្រាប់កសាងជាសៀគ្វីដើម្បីធ្វើប្រមាណវិធីផ្សេងៗដែលកើតចេញពីកន្សោមមួយ។ ហើយវាគ្រាន់តែតំណាងឱ្យត្រង់ស៊ីស្ទ័រ ដែលអាចធ្វើការបិទ បើកចរន្ត និងអាចប្រើជា ON និង OFF នៅក្នុងសៀគ្វីមួយ។

3.2.1. AND GATE

វាគឺជាការរួមគ្នាដែលបានមកពីសញ្ញាប្រមាណវិធីផលគុណតក្កវិទ្យានោះគឺជាអេឡិចត្រូនិចសៀគ្វីដែលមានលទ្ធផល(Output) ស្មើ ១ បើសិនការបញ្ចូលតម្លៃទាំងអស់មានតម្លៃស្មើ ១។ ករណីក្រៅពីនេះលទ្ធផលស្មើ ០។

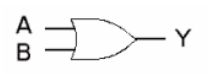


$$Y = A \cdot B$$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

3.2.2. OR GATE

វាគឺជា ការរួមគ្នាដែលបានមកពីសញ្ញាប្រមាណវិធីផលបូកតក្កវិទ្យានោះគឺជាអេឡិចសៀគ្វីដែលមានលទ្ធផល(Output) ស្មើ ១ កាលណាការបញ្ចូលតម្លៃមានតម្លៃស្មើ ១ យ៉ាងតិចនៅក្នុងនោះ។ ចំពោះករណីក្រៅពីនេះគឺលទ្ធផលស្មើ ០។ មានន័យថា $0 \text{ or } 1 = 1, 1 \text{ or } 0 = 1, 1 \text{ or } 1 = 1$ ហើយចំពោះ $0 \text{ or } 0 = 0$។



$$Y = A + B$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

3.2.3. NOT GATE

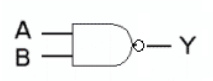
វាគឺបានមកពីប្រមាណវិធីធាតុបំពេញ(Complements)។ នោះគឺជាអេឡិចត្រូនិចសៀគ្វីដែលមានលទ្ធផលខាងក្រៅបានមកពីការផ្ទុយនឹងតម្លៃបញ្ចូល។ វាអាចហៅថា Inverter។ នៅក្នុងគណិតតក្កវិទ្យាប្រមាណនេះហៅថាឈ្នាប់មិន ដែលតាងដោយនិមិត្តសញ្ញា " – " ។

Truth table		NOT Gage Symbol	
Inputs	Outputs	A	NOT A
A	A^c	1	0
0	1	0	1

3.2.4. NAND , NOR និង Exclusive NOR gate

3.2.4.1. NAND gate

NAND gate គឺជាធាតុបំពេញនៃ AND gate។ លទ្ធផលស្មើ ១ ពេលបញ្ចូលតម្លៃស្មើ 0 យ៉ាងតិចនៅក្នុងនោះ។ លទ្ធផលស្មើ 0 ក្នុងករណីតែមួយគត់នៅពេលដែលបញ្ចូលទាំងអស់ស្មើ ១។ សញ្ញា "↑" ប្រើតំណាងឱ្យប្រមាណវិធី NAN gate។ $A \uparrow B = (A.B)^C$ ។ មានន័យថា 0 or 1 =1 , 1 or 0 = 1 , 1 or 1= 0 ហើយចំពោះ 0 or 0 =1

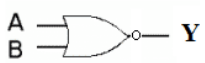


$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

3.2.4.2. NOR gate

គឺជាធាតុបំពេញនៃ OR gate។ លទ្ធផលស្មើ 0 ពេលបញ្ចូលតម្លៃស្មើ ១ យ៉ាងតិចនៅក្នុងនោះ។ លទ្ធផលស្មើ ១ ក្នុងករណីតែមួយគត់នៅពេលដែលបញ្ចូល ទាំងអស់ស្មើ 0 សញ្ញា " ↓ " ប្រើតំណាងឱ្យប្រមាណវិធី NAN gate។ $A \downarrow B = (A+B)$:



$Y = \overline{A+B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

3.2.4.3. Exclusive NOR gate

XNOR gate គឺជាធាតុបំពេញនៃ Exclusive_OR gate។ លទ្ធផលស្មើ ១ ពេលបញ្ចូលតម្លៃស្មើទាំងអស់ស្មើ 0 ឬ ទាំងអស់ ១។ លទ្ធផលស្មើ 0 នៅពេលបញ្ចូលតម្លៃយ៉ាងតិចស្មើ ១ សញ្ញា "." ប្រើតំណាងឱ្យប្រមាណវិធី XNOR។

$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

4. លក្ខណៈនៃតក្កវិទ្យា

លក្ខណៈនៃតក្កវិទ្យាប្រើសម្រាប់យកទៅដោះស្រាយទ្រឹស្តីបទ និង ប្រមាណវិធីផ្សេងៗ។

លក្ខណៈទី១

a. $x + 0 = x$

b. $x.1 = x$

លក្ខណៈទី២

a. $x + 1 = 1$

b. $x.0 = 0$

លក្ខណៈទី៤: លក្ខណៈផ្តុំ

a. $x + (y + z) = (x + y) + z$

b. $x.(y.z) = (x.y).z$

លក្ខណៈទី៥: លក្ខណៈបំបែក

a. $x. (x + y) = x.y + x.z$

b. $x + y.z = (x + y).(x + z)$

លក្ខណៈទី៣: លក្ខណៈត្រលប់

a. $x + y = y + x$

b. $x.y = yx$

<u>សម្រាយបញ្ជាក់</u>

$x + y.z = x.1 + y.z$

$= x.(1+z)+y.z$

$= x+ x.z +y.z$

$= x.1 + x.z + y.z$

$= x.(1 +y) + x.z + y.z$

$= x + x.y + x.z + y.z$

$= x.x + x.y + x.z + y.z$

$= x.(x + y) + z.(x+y)$

$= (x+ y).(x +z)$

5. គោលការណ៍ទ្វេភាពនៅក្នុងពិជគណិតប៊ូលីន (The Principle of duality)

នៅក្នុងពិជគណិតប៊ូលីនមាន គោលការណ៍ទ្វេភាពច្បាស់លាស់រវាងប្រមាណវិធីបូកជាមួយនឹងប្រមាណវិធីគុណ និងពីលេខ ១ ទៅលេខ ០ ឬ ពីលេខ ០ ទៅលេខ ១ វិញ។ មានន័យថាភាពជាគូបានមកពីការប្តូរសញ្ញា + ទៅសញ្ញា . និង ពីលេខ ១ ទៅលេខ ០ ឬ ពីលេខ ០ ទៅលេខ ១ វិញ។ ដូច្នេះបើមានលក្ខណៈ ឬ ទ្រឹស្តីបទពិសេសមួយបានដោះស្រាយនោះភាពជាគូរបស់វាទាញបាន ដោយស្វ័យប្រវត្តិ ដោយមិនបាច់ដោះស្រាយ ដោយគ្រាន់តែប្តូរសញ្ញា + ទៅ សញ្ញា . ឬ ពីប្តូរពីលេខ ១ ទៅ ០ និង ពីលេខ ០ ទៅលេខ ១ វិញ។

☺ **ឧទាហរណ៍:**

a. $x.(y + z) = x.y + x.z$ និង b. $x + y.z=(x + y). (x+z)$

	Column1	Column2	Column3
Row1	1+1=1	1+0=0+1=1	0+0=0
Row2	0.0=0	0.1=1.0=0	1.1=1

☺ **ឧទាហរណ៍:** ចូររកគូនៃកន្សោមខាងក្រោម៖

$\overline{A} + A.B + 1$ និង $x + 1 =1$

6. ទ្រឹស្តីបទនៃពិជគណិតប៊ូលី (Theorem of Boomlean Algebra)

ដើម្បីបង្កើតទ្រឹស្តីបទ យើងត្រូវមានវត្ថុមុខគេបង្អស់គឺសញ្ញាជើម ស្វ័យសត្យ និងលក្ខណៈ។

- **ទ្រឹស្តីបទទី១**

a. $x + x = x$

សម្រាយបញ្ជាក់ទ្រឹស្តីបទទី១. a.

$x + x = (x+ x) 1$

b. $x . x = x$

សម្រាយបញ្ជាក់ទ្រឹស្តីបទទី១.b.

$x.x = (x.x) + 0$

$= (x + x).(x + x^c)$ $= (x.x) + x.x^c$

$= x + x.x^c$ $= x + 0$

$= x + 0$ $= x$

$= x$

- **ទ្រឹស្តីបទទី២: a.** $x + x.y = x$ b. $x.(x + y) = x$

  សម្រាយបញ្ជាក់ទ្រឹស្តីបទទី២ .a. សម្រាយបញ្ជាក់ទ្រឹស្តីបទទី២ .b.

  $x + x.y = x.(1 + y)$ $x.(x + y) = x$

  $= x.1$ $x.x + x.y = x$

  $= x$ $x + x.y = x$

  ដូចនេះ $x + x.y = x$ $x.(1 + y) = x$

- **ទ្រឹស្តីបទទី៣: a.** $x + x^c.y = x + y$ b. $x.(x^c + y) = x.y$

  សម្រាយបញ្ជាក់ទ្រឹស្តីបទទី3. a. សម្រាយបញ្ជាក់ទ្រឹស្តីបទទី3. b.

  a. $x + x^c.y = (x + x^c).(x + y)$ $x.(x^c + y) = x.y$

  $= 1.(x + y)$ $x.x^c + x.y = x.y$

  $= x + y$ $x.y = x.y$

- **ទ្រឹស្តីបទទី ៤**

  $X^{2c} = x$

  $X^{3c} = x^c$

- **ទ្រឹស្តីបទទី ៥** (DeMorgan's)

  a. $(x + y)^c = x^c + y^c$

  b. $(x.y)^c = x^c + y^c$

  ទ្រឹស្តីបទទី5(a) និង (b) គឺ សំខាន់ និងមានប្រយោជន៍។ បន្ថែមពីលើនេះទៀតមានរហូតដល់ n អថេរដែលមានរៀងដូចខាងក្រោម៖

  $(x1 + x2 + \ldots + xn)^c = x1^c . x2^c \ldots .. xn^c$

  $(x1 . x2 \ldots . xn)^c = x1^c + x2^c + \ldots + xn^c$

7. អនុគមន៍ប៊ូលីន

7.1. និយមន័យ: អនុគមន៍ប៊ូលីនមានរៀងជាកន្សោមមួយដែលកើតចេញពីប្រមាណវិធី ទ្វេភាគទាំងពីរគឺ ប្រមាណវិធីបូក និង គុណ ព្រមទាំងប្រមាណវិធីធាតុបំពេញ វង់ក្រចក និង សញ្ញាស្មើ។

☺ **ឧទាហរណ៍**: $F(x,y,z) = x + y^c.z$ អានថា F ជាអនុគមន៍ x, y, z ស្មើនឹង $x + y^c.z$ រឺ អាចសរសេរ: $F(x,y,z) = x + y^c.z$។

ដើម្បីអោយតម្លៃទៅលើអថេរនិមួយៗនៃអនុគមន៍ នោះតម្លៃអថេរអោចមានតំលៃ 0 និង ១។ អនុគមន៍ប៊ូលីនអាចបង្ហាញផងដែរនៅក្នុងតារាងភាពពិត ដោយចំនួនជួរដេកនៃតារាងនិមួយៗ ស្មើនឹង 2^n (n ជាចំនួនអថេរ) ដោយជួរដេកនៃតារាងនិមួយៗរាប់ពី ១ ទៅ 2^n ។

Input			Outputs		
x	y	z	y^c	$y^c.z$	$F = x + y^c.z$
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	0	1

7.2. ការបង្រួមនៃអនុគមន៍ (Minization of Boolean Function)

ដើម្បីបង្រួមអនុគមន៍ត្រូវប្រើលក្ខណៈ, ទ្រឹស្តីបទ និងរបៀបមួយចំនួនទៀត ដែលបានសិក្សារួច។ ការបង្រួមនៃអនុគមន៍ងាយស្រួលក្នុងការដោះស្រាយជាមួយ Logic Gate។

☺ **ឧទាហរណ៍:** ចូរបង្រួមកន្សោមប៊ូលីន អោយទៅជាចំនួន Literal មួយ។

a. $X^c.y^c.z + x^c.y.z + x.z$

b. $x.y + x^c.z + y.z$

c. $(x + y)..(x^c + z).(y+z)$

សំរាយបញ្ជាក់

a. $X^c.y^c.z + x^c.y.z + x.y^c = x^c.z(y^c + y) + x.y^c$

$= x^c.z.1+x.y^c$

$=x^c.z+ x.y^c$

ដូច្នេះ $x^c.y^c.z + x^c.y.z + x.y^c = x^c.z+ x.y^c$

b. $x.y + x^c.z + y.z = x.y + x^c.z+y.z.1$

$= x.y + x^c + y.z.(x^c + x)$

$=x.y.(1+ z) + x^c.z(1 +y)$

$=x.y + x^c.z$

ដូច្នេះ $x^c.y^c.z + x^c.y.z + x.y^c= x.y + x^c.z$

c. $(x + y)..(x^c + z).(y +z)$ ជាគូរនៃ b

7.3. ធាតុបំពេញនៃអនុគមន៍ (Complement of a function)

ធាតុបំពេញនៃអនុគមន៍ F គឺ F^c បានមកពីលេខ 1 ទៅលេខ 0 ឬ ពី 0 ទៅ លេខ 1 វិញ។

☺ **ឧទាហរណ៍១:** តារាងភាពពិតចំពោះអនុគមន៍ $F=x.y^c+ x^c.z$ និងធាតុបំពេញរបស់វាគឺ F^c

x	y	z	x^c	y^c	$x.y^c$	$x^c.z$	$F=x.y^c+x^c.z$	F^c
0	0	0	1	1	0	0	0	1
0	0	1	1	1	0	1	1	0
0	1	0	1	0	0	0	0	1
0	1	1	1	0	0	1	1	0
1	0	0	0	1	1	0	1	0
1	0	1	0	1	1	0	1	0
1	1	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	1

ឧទាហរណ៍ ២: រកជាតុបំពេញនៃអនុគមន៍ខាងក្រោមដោយប្រើទ្រឹស្តីបទ De Morgan

a. $F1=x^c.y.z^c + x^c.y^c.z$

b. $F2=x.(x^c.y^c + y.z)$

សម្រាយបញ្ជាក់

(a). $F1^c = (x^c.y.z^c+x^c.y^c.z)^c$

$= (x^c.y.z^c)^c.(x^c.y^c.z^c)$

$= (x+y^c+z).(x+y+z^c)$

ដូច្នេះ $F1=x^c.y.z^c + x^c.y^c.z$ ជាតុបំពេញរបស់វាគឺ $F1^c=(x+y^c+z).(x+y+z^c)$

7.4. Minterms និង Maxterms (Minterms and Maxterms)

7.4.1. Minterms: គឺជាក្រុម (Term) មួយកើតចេញពីប្រមាណវិធីគុណដែលមានអថេរជាមួយអថេរនិមួយៗ ជា Prime ហើយបើសិនជាប់ទាក់ជាមួយប្រព័ន្ធគោលពីរនោះ Prime 0 និង Unprime គឺ 1។ មាននិយថាកន្សោម Minterms និមួយៗ កើតឡើងពី Unprime ដោយប្រើប្រមាណវិធីគុណ។

7.4.2. Maxterms: គឺជាក្រុមមួយដែលផ្ទុយពីកន្សោម Minterm។

Variable			Minterms		Maxterms	
X	y	z	Term	Designation	Term	Designation
0	0	0	$x^c.y^c.z^c$	m1	x+y+z	M1
0	0	1	$x^c.y^c.z$	m2	$x+y+z^c$	M2
0	1	0	$x^c.y.z^c$	m3	$x+y^c+z$	M3
0	1	1	$x^c.y.z$	m4	$x.y^c+z^c$	M4
1	0	0	$x.y^c.z^c$	m5	$x^c+y.z$	M5
1	0	1	$x.y^c.z$	m6	x^c+y+z^c	M6
1	1	0	$x.y.z^c$	m7	x^c+y^c+z	M7
1	1	1	x.y.z	m8	$x^c+y^c+z^c$	M8

នេះគឺជានិមិ្តតសញ្ញារបស់ Minterms និង Maxterms ដែលបង្ហាញនៅក្នុង Truth Table ហើយមានទំរង់ជា mj និង Mj ដោយ mj ជាក្រុម Minterm Designation និង Mj ជា ក្រុម Maxterms Designation។

7.5. ផលបូកនៃផលគុណ (Sum_of_Products)

កន្សោមផលបូកនៃផលគុណគឺជា ក្រុមផលគុណជាច្រើននៃតក្កវិទ្យាបន្ថែមប្រមាណវិធីបូកជាមួយគ្នា

☺ **ឧទាហរណ៍ ២:** កន្សោម $x^c.y.z^c + x^c.y^c.z$ ជាផលបូកនៃផលគុណ ដើម្បីរកកន្សោមអនុគមន៍ប៊ូលីន ក្នុងទំរង់ផលបូកនៃផលគុណត្រូវអនុវត្តតាមជំហានដូចខាងក្រោម៖

1. សង់តារាងភាពពិតដើម្បីរកតំលៃអនុគមន៍ប៊ូលីន
2. ទំរង់ Minterms ដែលបានមកពីតំលៃ Boolean Funtion ពីជំហានទី 1 ដែលមាន តំលៃស្មើ 1។
3. ធ្វើអោយបាននូវកន្សោមជាផលបូកនៃ Minterms ទាំងអស់នៃជំហានទី ២

☺ **ឧទាហរណ៍ ២:** ពន្លាតកន្សោម $F= x.y^c + x^c.z$ ជាផលបូកនៃផលគុណ

1. តារាងភាពពិត

x	y	z	x^c	y^c	$x.y^c$	$x^c.z$	F
0	0	0	1	1	0	0	0
0	0	1	1	1	0	1	1
0	1	0	1	0	0	0	0
0	1	1	1	0	0	1	1
1	0	0	0	1	1	0	1
1	0	1	0	1	1	0	1
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0

2. ទំរង់ Minterms និមួយៗ ដែលមានតំលៃស្មើ 1 គឺ

   $001 = x^c.y^c.z$

   $011 = x^c.y.z$

   $100 = x.y^c.z^c$

   $101 = x.y^c.z$

3. កន្សោមផលបូកនៃ Minterms ទាំងអស់គឺ

   $F = x^c.y^c.z + x^c.y.z + x.y^c.z^c + x.y^c.z$

   ☛ **ចំណាំ:** វិធីខាងលើយើងអាចធ្វើតាមរបៀបដូចខាងក្រោម៖

   $F= x.y^c + x^c.z$ មានបីអថេរ x, y, z, Term1 $x.y^c$ ខ្វះអថេរ z នោះយើងថែម $z+z^c$

   $x.y^c(z + z^c) = x.y^c.z + x.y^c.z^c$

   Term2 $x^c.y$ ខ្វះអថេរ y នោះយើងថែម $y + y^c$, $x^c.y(y + y^c)= x^c.y.z + x^c.y^c + z$

ដូច្នេះ $F=x^c.y^c.z + x^c.y.z + x.y^c.z^c + x.y^c$

ឬ F= m5 + m4 + m3 + m1

7.6. ផលគុណនៃផលបូក (Products_of_Sum)

កន្សោមផលគុណនៃផលបូកគឺ ជាក្រុម(Term)ផលបូកជាច្រើននៃតក្កវិទ្យាបន្ថែមប្រមាណវិធីជាមួយគ្នា។

ឧទាហរណ៍: កន្សោម $(x+y^c).(x+y^c)$ ជាផលបូកនៃផលគុណ

ដើម្បីរកកន្សោមអនុគមន៍ប៊ូលីននៃផលគុណត្រូវអនុវត្តតាមជំហាន ដូចខាងក្រោមនេះ៖

1. សង់តារាងភាពពិតដើម្បីរកតំលៃអនុគមន៍ប៊ូលីន
2. ទំរង់ Maxterms ដែលបានមកពីតំលៃ Boolean Funtion ពីជំហានទី 1 ដែលមានតំលៃស្មើ 0
3. ធ្វើអោយបាននូវកន្សោមជាផលគុណនៃ Maxterms ទាំងអស់នៃជំហានទី២

☺ **ឧទាហរណ៍ ២:** ពន្លាតកន្សោម $F= x.y^c + x^c.z$ ជាផលគុណនៃផលបូក ទំរង់ Maxterm និមួយៗដែលមានតំលៃស្មើ 0 គឺ:

$000 = x + y + z$

$010 = x + y^c + z$

$110 = x^c + y^c + z$

$111 = x^c + y^c + z$

ដូច្នេះ $F = (x+y+z).(x+y^c+z).(x^c+y^c+z)(x^c+y^c+z^c)$

ឬ $F= M0 + M2 + M6 + M7$

8. ប្រមាណវិធីឥព្វន្ធរបស់កំព្យូទ័រ (COMPUTER ARITHMETIC)

ALU គឺជាឧបករណ៍មួយនៅក្នុងស៊ីភីយូមានតួនាទីអាចធ្វើការគណនាប្រមាណវិធីលេខនព្វន្ត(បូក, ដក, គុណ និង ចែក) និង ប្រមាណវិធីតក្កវិទ្យា(បូក, គុណ និង ធាតុបំពេញ)។ ចំណែក Control Unit គឺជាឧបករណ៍ដ៏សំខាន់ជាងគេក្នុងស៊ីភីយូដែរ ដែលបញ្ជាឱ្យ ALU គណនានូវអ្វីតាមពាក្យបញ្ជាដែលបានបញ្ចូលមកមានដូចជា Add, Sub, Mul, Div និងប្រមាណវិធីតក្ក Or, And, Not និងធ្វើប្រមាណវិធីរំកិលលេខប្រព័ន្ធគោលពីរខាងឆ្វេងទៅខាងស្តាំ។ វាធ្វើការបែងចែកទៅតាមលំដាប់លំដោយនៃសញ្ញាបញ្ចូល។ វាមានទំនាក់ទំនងជាមួយនឹងសមាសភាគផ្សេងៗដូច Register សម្រាប់ធ្វើការទទួលយក និងបញ្ចេញទិន្នន័យ។ វាក៏អ្នកធ្វើនូវប្រមាណវិធី Flagsដែលទាក់ទងនឹងដំណើរការលក្ខខណ្ឌ និងរង្វិលជុំ។

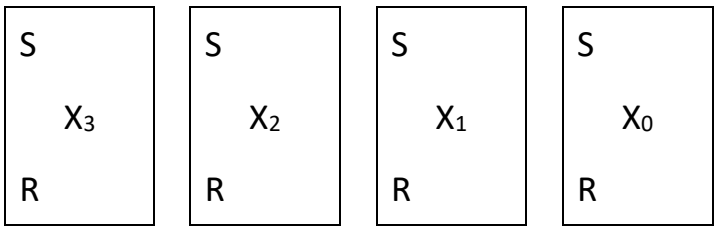
8.1. Flip_Plop(FF): គឺជាតក្កសៀគ្វី(Logic Circuits) ដែលផ្ទុកនូវព័ត៌មានគិតជាប៊ីតដោយមាន Logic1 និង Logic0។ ពីរ ឬ ច្រើន FF_s តភ្ជាប់ជាមួយគ្នាអាចផ្ទុកព័ត៌មានជា ៨ ប៊ីត ឬជា១៦ ប៊ីត ឬលើសពីនេះ។ FF មួយមានពីរ ឬច្រើន Inputs និងមានតែពីរ Outputs។ ហើយ Outputs ទាំងពីរជានិច្ចជាកាលផ្ទុយនឹងគ្នា។

ដ្យាក្រាមតក្កវិទ្យា				និម្មិតសញ្ញាតក្កវិទ្យា
R	S	X	X^c	Remark
0	0	NC	NC	គ្មានការផ្លាស់ប្តូរពី State មុន
0	1	1	0	Set ; Set ក្នុងករណី S=1 , R=0
1	0	0	1	Reset ក្នុងករណី S=0; R=1
1	1	0	0	Next state unpredictable

តាមតារាងខាងលើជាឧទាហរណ៍ងាយមួយរបស់ FF។ វាមានព័ត៌មានផ្តល់មកឱ្យវិញពី Two Outputs (X និង X^c) ទៅ Two inputs (R និង S)។ នេះជាពត៌មានផ្តល់ដោយ FF ជាមួយ Memory តាមរូបខាងលើ FF មានការបញ្ចូលតម្លៃពីរហើយកំណត់បានលទ្ធផលពីគ្រឺ X និង X^c ដែលជាផ្ទុយនឹងគ្នា។ ជាធម្មតាលទ្ធផលរបស់ FF ចាត់ទុកជា Output leed (ការគិតដោយឡែក) នៃលទ្ធផល X (output) លុះត្រាលទ្ធផល X^c (output) ជាប្រភេទដោយឡែក។ ដូច្នេះបើ State(0 slate, 1 state)មួយ, លទ្ធផលនៃ FF តាមរូបខាងលើជា high (1) វាយល់ X ជា logic 1 និង X^c ជា low(0) វាយល់ថាជា logic 0 ។

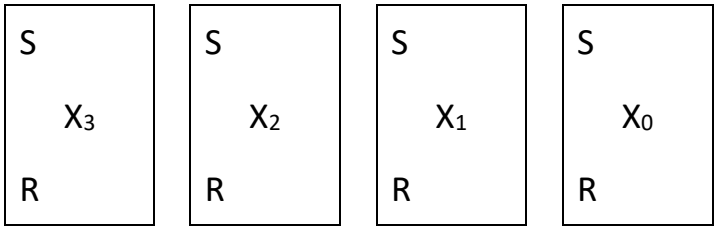
8.2. **ការតោងចំនួនគត់** (Integer Representation)

ចំនួនលេខប្រើក្នុងឌីជីថលម៉ាស៊ីន(digital machine) មានលេខតំណាង ដោយប្រើប្រាស់ឧបករណ៍ផ្ទុកហៅថា FF។ វាមានលេខផ្ទាល់ខ្លួនជាច្រើនតំណាងឱ្យឧបករណ៍ផ្ទុកលេខគោលពីរ(binary storage device) គឺជាប្រព័ន្ធមួយតំណាងឱ្យចំនួនគត់។



(a) តំណាងអោយចំនួនគត់

តម្លៃនៅក្នុង register = $X_3.2^3 + X_2.2^2 + X_3.2^1 + X_2.2^0$



b) Sign_plus _ Magnitude system

តាមរូបខាងលើបង្ហាញពី FF បួន X_3, X_2, X_1 និង X_2 ប្រើសម្រាប់ផ្ទុកចំនួនលេខ។ ការសរសេរតម្លៃឬបញ្ចេញលទ្ធផលរបស់ FF ដែលឱ្យវាគ្រាន់តែជាទំរង់ចំនួនគត់។ ដូច្នេះ $X_3 = 1$, $X_1 = 0$, $X_0 = 0$ ស្មើ 1100 រឺស្មើ 12 (decimal)។

វាមានសារៈសំខាន់ណាស់សម្រាប់តំណាងឱ្យលេខវិជ្ជមាន និង ចំនួនអវិជ្ជមានដូច្នេះការបន្ថែម bit ជាតំរូវការចាំដើម្បីឱ្យចំនួនវិជ្ជមាន និង អវិជ្ជមានហៅជា sign bit។ នេះគឺជាភាពទូទៅជំនួសខាងឆ្វេង (Left) នៃទំហំ bit។ ក្នុងរូប (b) X_4 ជា sign bit និង X_3, X_2, X_1, និង X_0 ជាទំហំ bit។ លេខ 0 នៅក្នុង X_4 មានន័យថាជាចំនួនវិជ្ជមាន និង លេខ ១ ក្នុង X_4 មានន័យថាជាចំនួនអវិជ្ជមាន។ ដូច្នេះ $X_4 = 0$, $X_3 = 1$, $X_2 = 1$, $X_1 = 0$ និង $X_0 = 1$ ឱ្យជាចំនួនវិជ្ជមាន 1101 ឬ 13 decimal, និង $X_4 = 1$, $X_3 = 1$, $X_2 = 1$, $X_1 = 0$ និង $X_0 = 1$ឱ្យជាចំនួនអវិជ្ជមាន 1101 ឬ - ១៣ decimal។ ប្រព័ន្ធនេះហៅថាប្រព័ន្ធគោលពីរតាងឱ្យចំនួនគត់ ឬ signed – magnitude binary integer system។ បើ register មួយមាន FF ប្រាំបីនោះ signed – integer binary system មាន 7 magnitude ឬ integer និងមាន sign bit តែមួយគត់។

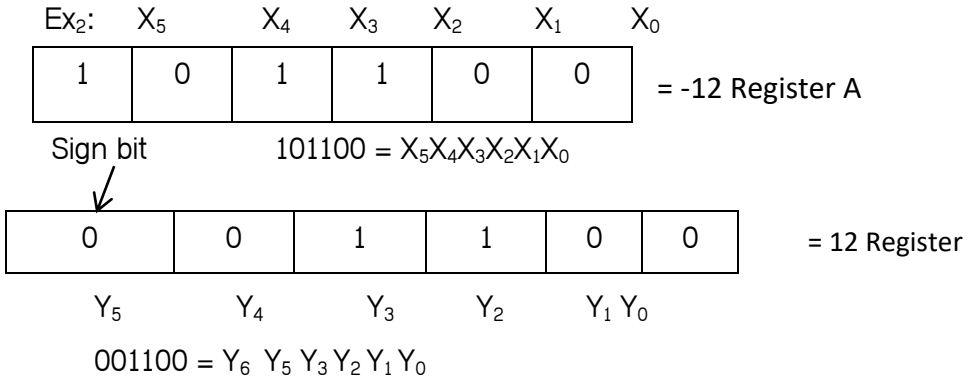
ឧទាហរណ៍$_1$: 1011 = 1.2 គណនាជា decimal តាម Iutege repse – sentation

តាមរូបមន្ត $X_0.2^3 + X_2.2^2 + X_1.2^1 + X_0.2^0$

$=1.2^2 + 0.2^2 + 1.2^1 + 1.2^0$

$=8 + 2 + 1$

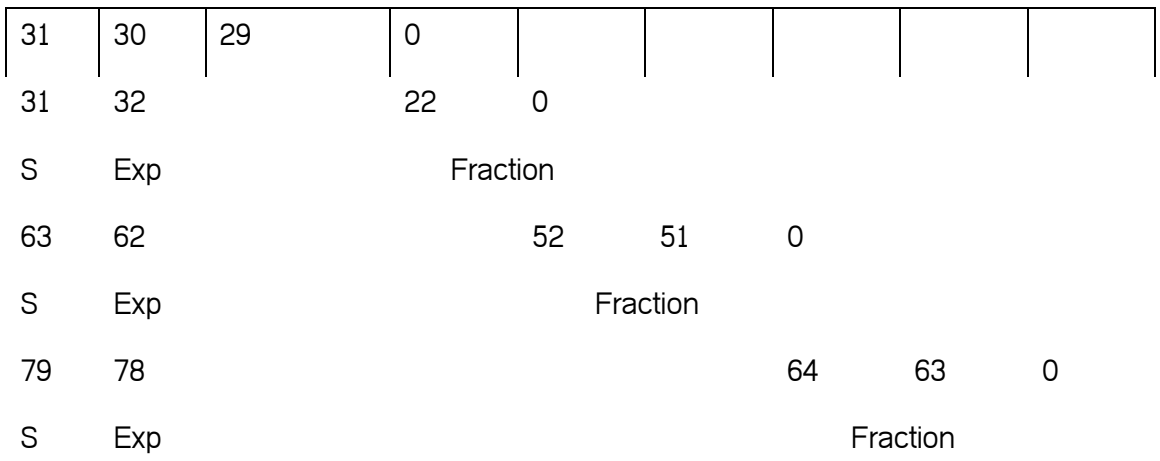
$=11$



Positive Number			Negative Number		
Decimal	Binary	Hexadecimal	Decimal	Binary	Hexadecimal
0	0000	1	8	1000	8
1	0001	2	9	1001	9
2	0010	3	10	1010	A
3	0011	4	11	1011	B
4	0100	5	12	1100	C
5	0101	6	13	1101	D
6	0110	7	14	1110	E
7	0111	8	15	1111	G

8.3. ចំនួនឥសនាគ (Floating_Point)

លេខជាចំនួនទសភាគអាចហៅម្យ៉ាងទៀតថាជាចំនួនពិត។ លេខជាទសភាគចែកជាបីផ្នែកគឺ: Sign_ bit, biased exponent, និង Significand bits។ គ្រួសារនៃ Arithmetic Coprocessor ទទួលស្គាល់ប្រភេទទិន្នន័យនេះមានបីករណីគឺ short (32 bits), long (64 bits), និង temporary (80 bits) គួរកត់សំគាល់ដែលថាទំរង់ short ហៅថា single-precision, ទំរង់ long ហៅថា double precision និងទំរង់ Temporary ហៅថា extended precision។ បើក្នុងភាសា Assemble យើងសរសេរ x real4 100.34 មានន័យថាប្រើ single-precision និង y real8 1968.23 ប្រើ double precision។



8.3.1. ការបំលែងពីប្រព័ន្ធគោលដប់ទៅចំនួនទសភាគ

ការធ្វើប្រមាណវិធីបំលែងលេខគោលដប់ទៅជាទំរង់ទសភាគ ត្រូវអនុវត្តតាមជំហានជាបន្តបន្ទាប់ដូចខាងក្រោម៖

1. បំលែងលេខគោលដប់ទៅគោលពីរសិនមុខនឹងធ្វើការបំលែង
2. Normalize binary number មានន័យថាកំណត់ x.xxxxx... គុណនឹង 2^6
3. គណនា Biased exponent
4. ផ្ទុកលេខឱ្យទៅជាទំរង់ Floatin-point

☞ **ឧទាហរណ៍:** 4 ជំហានបន្តបន្ទាប់ខាងក្រោមនេះជាការបំលែងលេខ Dec=100.25 ទៅជាទំរង់ Floating_point ដោយអនុវត្តន៍តាមទំរង់ Single_precision:

Step	Result
1.	100.25 = 1100100.01
2.	1100100.01=1.10010001x 2^6
3.	110 + 01111111 = 10000101
4.	Sign = 0 ;Exponent = 10000101
	Significand = 10010001000000000000000
	Hexaa =42C8800H

8.3.2. ការបំលែងពីចំនួនទសភាគទៅប្រព័ន្ធគោលដប់

ដើម្បីបំលែងលេខទសភាគទៅជាទំរង់គោលដប់ ត្រូវអនុវត្តតាមជំហានជាបន្តបន្ទាប់ដូចខាងក្រោម៖

1. បំបែក Sign_bit, biased exponent, និង Significand
2. បំលែង Biased exponent ទៅជា True exponent =Exp – Bias
3. សរសេរលេខ normalize binary number
4. បំលែងវាទៅជា de_normlize binary number
5. បំលែង de_normlize binary number ទៅជា Decimal number

☞ **ឧទាហរណ៍:** ចូរបំលែងលេខជា single_precision floating_point ខាងក្រោមនេះ ទៅជាលេខ Decimal:

a. 11000001110010010000000000000000
b. 11000001111100000000000000000000

Step	Result
1.	Sign = 1; Exponent = 10000011
	Significand = 10010000000000000000
2.	100 = 10000011 - 01111111
3.	1.1001001 x 2^4
4.	11001.001
5.	-25.125

8.3.3. ការរក 2s Complement

ការ 2s Complement ទាក់ទងនឹងប្រមាណវិធីជាច្រើនដូចប្រមាណវិធីលេខគិតសញ្ញា (Signed Number) ឬប្រមាណវិធីដកតាម Microprocessor។ ហើយដើម្បីរក 2s complement ត្រូវអនុវត្តតាមជំហានបន្តបន្ទាប់ដូចខាងក្រោមនេះ៖

Step1: ប្តូរលេខ Binary ពី 1 ទៅ 0

Step2: យកលទ្ធផលជំហានទី១ ថែម ១

ឧទាហរណ៍: រក 2's complement នៃលេខ 12

12=0000 1100(bits)

1111 0011

Add 1

1111 010

8.3.4. Binary Half_Adder

បែបបទសំខាន់ប្រើក្នុងធាតុលេខគណិត (Arithmetic element)ក្នុង ALU នៃស៊ីភីយូ គឺប្រើជា haff_adder ដោយមានការបញ្ចូលតម្លៃពីរនិងធ្វើការបញ្ចេញលទ្ធផលមួយ។ មុខងារចំបងរបស់ haff_adder ប្រើសម្រាប់ធ្វើប្រមាណវិធីបូកពីរចំនួនក្នុងលេខគោលពីរឱ្យលទ្ធផលជា S(Sum) មួយនិង C(carry) មួយតាមប្រមាណវិធីបូក។

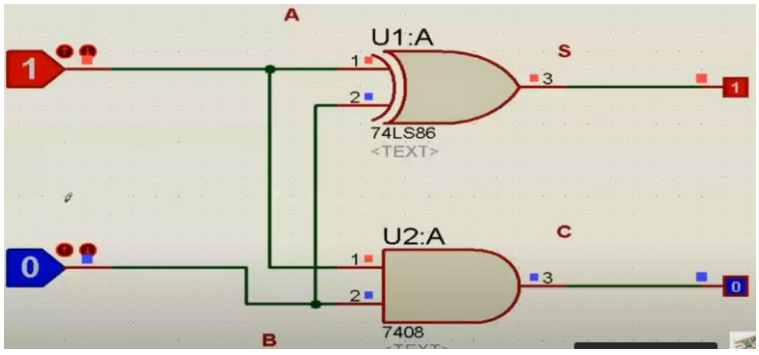
ខាងក្រោមជាតារាងភាពពិតដោយមាន A, B និង output, តាងដោយ S និង C។

Input		Outputs	
A	B	S	C
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

តាមតារាងភាពពិតគឺគិតតាមផលបូកនៃផលគុណយើងបានកន្សោម S និង C ដូចខាងក្រោម៖

$S = A^C.B + A.B^C$

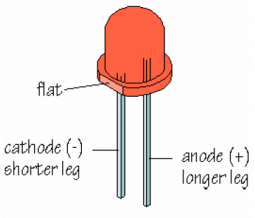
C= A. B ចំពោះសៀគ្វីខាងក្រោមតំណាងឱ្យកន្សោម S និង C គឺ

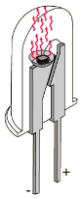


9. ឧបករណ៍អេឡិចត្រូនិច

9.1. Led

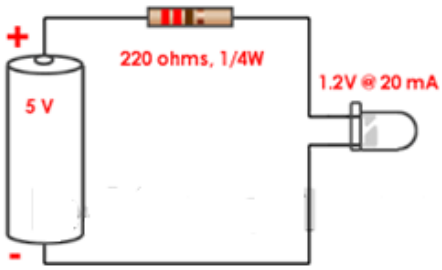
LED គឺជាអំពូលមួយប្រភេទដែលផ្សំពី semiconductors និង electroluminescence សម្រាប់បង្កើតនូវពន្លឺ។ វាមានជើងពីរគឺ Cathode(-) និង anode(+) ។

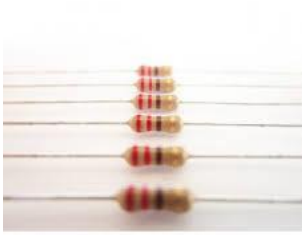




9.2. Resistor

រ៉េស៊ីស័រគឺជាគ្រឿងអេឡិចត្រូនិចមានតួនាទីក្នុងការប្រែចរន្តអគ្គិសនី ឬ កំណត់តម្លៃចរន្តតង់ស្យុង ដើម្បីផ្តល់ទៅឱ្យគ្រឿងទទួលផ្សេងៗតាមតម្រូវការ។ គេប្រើវានៅក្នុងសៀគ្វីជាច្រើនដូចជា ម៉ាញ៉ូទូទេស្សន៍, ទូរស័ព្ទ, ម៉ាស៊ីនកុំព្យូទ័រ និងឧបករណ៍ដទៃច្រើនទៀត។



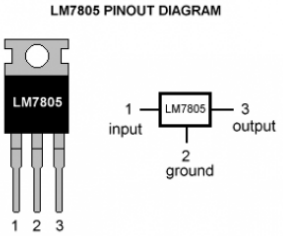


តាមរូបខាងលើការប្រើថ្ម (Battery) ជាប្រភពផ្តល់ចរន្តអគ្គិសនីសម្រាប់ឱ្យភ្លើង ៥ វុល ទៅឱ្យ Led។ តែដោយសារ LED ប្រើភ្លើងតែ 1.2 វុល ដូច្នេះត្រូវប្រើរ៉េស៊ីស័រ ២២0 ohms មួយដើម្បីប្រែចរន្តឱ្យនៅសល់តែ 1.2 វុល។ បើអត់ប្រើវាភ្លើងចេញពីប្រភពមានភ្លើង ៥ វុលធ្វើឱ្យអំពូល Led ឆេះ ឬ ខូច។

9.3. Transistor

ត្រង់ស៊ីស្ទ័រ(Transistor)គឺជាគ្រឿងអេឡិចត្រូនិចមួយប្រភេទ ដែលផលិតឡើងពីសារធាតុ Semiconducto ហើយគេប្រើសម្រាប់ត្រួតពិនិត្យ និងបញ្ជាចរន្តនៅក្នុងសៀគ្វីអគ្គិសនីដែលជាទូទៅគេប្រើវាកុងតាក់ភ្ជាប់ និងកុងតាក់ផ្តាច់ចរន្ត និងពង្រីករលកសញ្ញាផងដែរ។ វាអាចប្រើជា Switch ON(1) និង Swith OFF(0) ដូច្នេះគេយក ១ ច្រើន និង 0 ច្រើនផ្សំចូលគ្នាបង្កើតជាទិន្នន័យ និងពាក្យបញ្ជាសម្រាប់ប្រើប្រាស់នៅក្នុងកុំព្យូទ័រ និងឧបករណ៍អេឡិចត្រូនិចផ្សេងៗដទៃជាច្រើនទៀត។

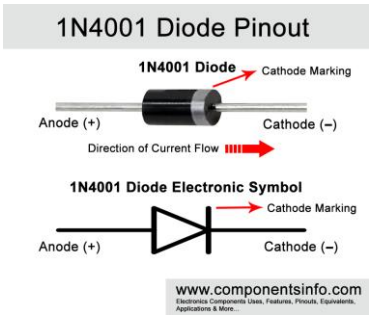
LM7805 ជាប្រភេទត្រង់ស៊ីស្ទ័រសម្រាប់បំលែងចរន្តអគ្គិសនីពី ១២ វុល ឬលើសពីនេះទៅជាចរន្ត ៥ វុល វាមានជើងបីដូចបង្ហាញតាមរូបខាងក្រោម៖



9.4. Diode

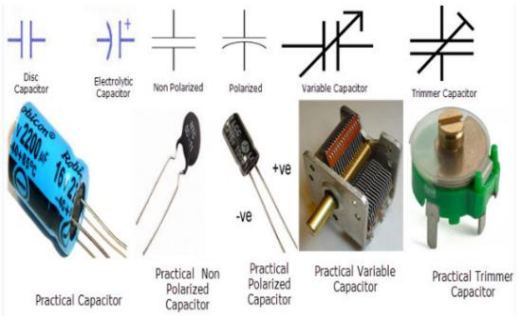
ឌីយ៉ូតគឺគ្រឿងអេឡិចត្រូនិចមួយប្រភេទដែលផលិតពីសារធាតុស៊ីលីខុន (Silicon) ឬ Germanium គេប្រើវាសម្រាប់កែទិសដៅចរន្តឱ្យរត់ទៅតាមទិសដៅកំណត់មួយ។ វាមានតួនាទីកែ

ទិសដៅចរន្តឆ្លាស់ទៅជាចរន្តជាប់ធ្វើជាភ្លើងសញ្ញាបញ្ជាក់ពីទង្វើការឧបករណ៍ ឬគ្រឿងទទួលកំពុងធ្វើការការពារគ្រឿងទទួលមិនឱ្យឆេះនៅពេលផ្តល់ប្រភពខុសប៉ូល... ជាដើម។



9.5. **កាប៉ាស៊ីទ័រ**

កាប៉ាស៊ីទ័រគឺគ្រឿងអេឡិចត្រូនិចមួយប្រភេទដែលប្រើសម្រាប់ផ្ទុក ឬផ្ទេរចរន្តអគ្គិសនី។ វាមានតួនាទីច្រើនយ៉ាងដូចជាច្រោះរលកសំលេងការពារកុំឱ្យចរន្ត DC រត់ឆ្លងកាត់សៀគ្វីធ្វើឱ្យរលកសញ្ញាមានលំនឹងបង្កើតលំយោលករ...ជាដើម។



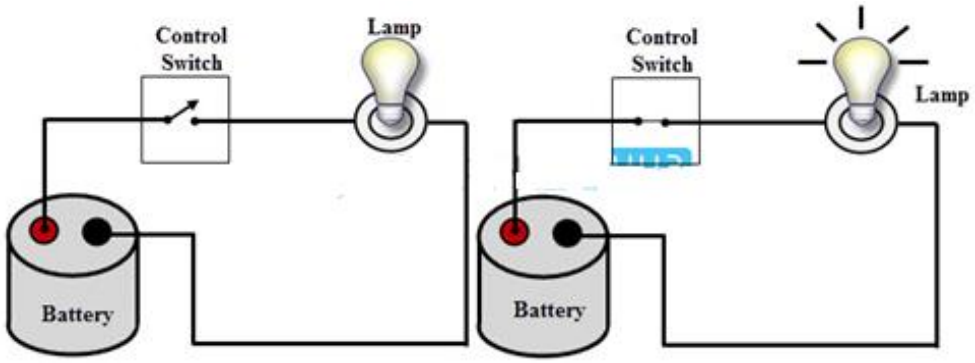
9.6. Breadboard

វាគឺជាក្តារឆ្លាស្ថិចរាងចតុកោណដែលមានបណ្តុំប្រហោងតូចៗជាច្រើននៅក្នុងនោះ។ រន្ធទាំងនេះអនុញ្ញាតឱ្យយើងបញ្ចូលសមាសធាតុអេឡិចត្រូនិចបានយ៉ាងងាយស្រួលអាចបង្កើត និងសាកល្បងសៀគ្វី។ ដូចជាអេឡិចត្រូនិកតាមរូបខាងក្រោម ដែលមានថាមពលថ្មកុងតាក់ឧបករណ៍ទប់និងអំពូល LED (អំពូលបញ្ចេញពន្លឺ)។ ដើម្បីស្វែងយល់បន្ថែមអំពីគ្រឿងអេឡិចត្រូនិកនីមួយៗសូមមើលអេឡិចត្រូនិចផ្សេងៗដែលបានរៀនរួចទៀត។

9.7. Switch

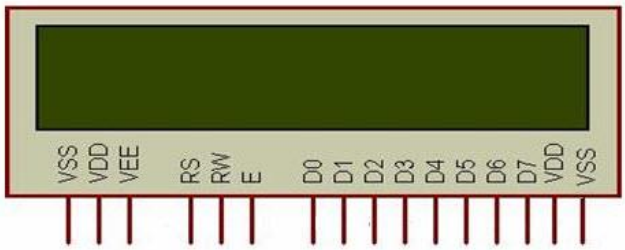
កុងតាក់ គឺជាឧបករណ៍ដែលត្រូវបានរចនាឡើងដើម្បីរំខានចរន្តចរន្តនៅក្នុងសៀគ្វី។ និយាយសាមញ្ញពាក្យប្តូរអាចបង្កើតឬបំបែកសៀគ្វីអគ្គិសនី។ ពាល់កម្មវិធីអគ្គិសនីនិងអេឡិចត្រូនិចប្រើយ៉ាងហោចណាស់កុងតាក់មួយដើម្បីដំណើរការប្រតិបត្តិការនិងបិទឧបករណ៍។

ដូច្នេះកុងតាក់ គឺជាផ្នែកមួយនៃប្រព័ន្ធបញ្ជាហើយបើគ្មានវាទេប្រតិបត្តិការត្រួតពិនិត្យមិនអាចត្រូវបានសម្រេចទេ។ កុងតាក់មួយអាចដំណើរការមុខងារពីរគឺបិទបើក (បិទទំនាក់ទំនង) ឬបិទទាំងស្រុង (ដោយបើកទំនាក់ទំនងរបស់វា)។ នៅពេលទំនាក់ទំនងរបស់កុងតាក់ត្រូវបានបិទកុងតាក់បង្កើតជាផ្លូវបិទសម្រាប់ចរន្តដើម្បីហូរ ដូច្នេះបន្ទុកផ្ទុកថាមពលពីប្រភព។ នៅពេលទំនាក់ទំនងនៃកុងតាក់បើកគ្មានថាមពលនឹងត្រូវបានប្រើប្រាស់ ដោយបន្ទុកដូចបង្ហាញក្នុងរូបភាពខាងក្រោមទេ។



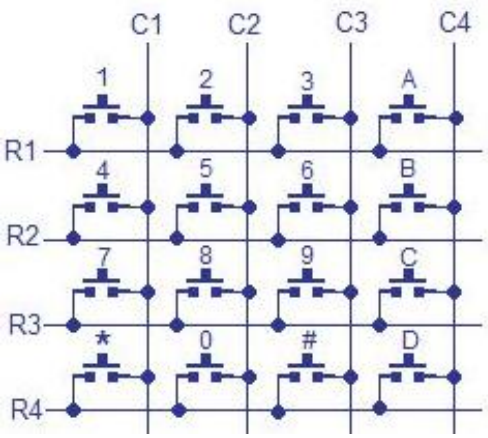
9.8. LCD 16X2

LCD (Liquid Crystal Display)គឺជាឧបករណ៍បញ្ចេញនូវទិន្នន័យ និង ព័ត៌មាននៅលើអេក្រង់នៃ LCD។ វាមាន Rows ចំនួន 2 (0 – 1)មាន Columns ចំនួន 16 (0 – 15)មាន Pins ចំនួន 16។



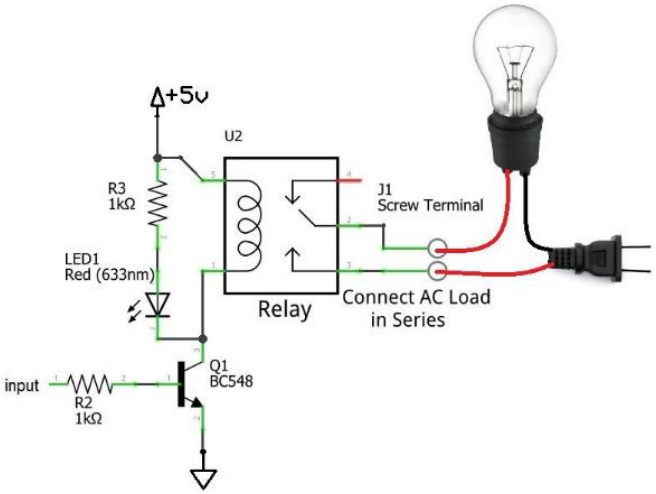
9.9. Keypad

Keypad គឺជាបន្ទះសៀគ្វីមួយដែលផ្គុំឡើងពីប៊ូតុង ជា (Button) ច្រើន។ វាជាប្រភេទ Input Device ដែលគេប្រើសម្រាប់ធ្វើការបញ្ចូលតម្លៃលេខ និង អក្សរពី Keypad ទៅកាន់ Arduino Board។



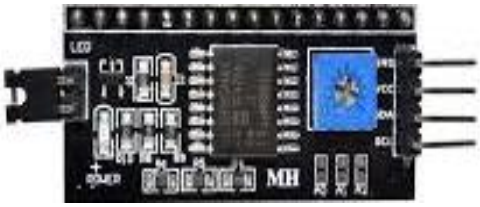
9.10. Relay

Relay ជាឧបករណ៍អគ្គិសនីមួយដែលមានតួនាទីបញ្ជូនបន្តនូវចរន្តអគ្គិសនី។ វាត្រូវបានប្រើសម្រាប់បញ្ជូនបន្តនូវចរន្តអគ្គិសនី 220V ឬក្រោមនេះទៅកាន់ឧបករណ៍ផ្សេងទៀត ដោយវាអាចបើកឬបិទចរន្តអគ្គិសនីអាស្រ័យតាមការបញ្ជាពី Arduino board។ យើងត្រូវការ Relay ចំនួន៨សម្រាប់គម្រោងនេះ។



9.11. I2C LCD 1602

ប្រើសម្រាប់ភ្ជាប់ពី LCD ទៅកាន់ Arduino Board។ វាមាន 16 Pins សម្រាប់ភ្ជាប់ទៅកាន់ LCD និង មាន 4 Pins (GND, VCC, SDA, SCL) សម្រាប់ភ្ជាប់ទៅកាន់ Arduino។



9.12. L298N Motor Driver

គឺជា driver ដែលអាចឱ្យយើងគ្រប់គ្រងល្បឿន និង ទិសដៅរបស់ម៉ូទ័រទាំងពីរ។ នៅលើម៉ូទ័រ driver នេះគឺមានជើងសម្រាប់ដោតម៉ូទ័រចំនួនពីរហើយវាមានជើងបីទៀតដែលមានឈ្មោះ GND, 5v និងប្រលោះសម្រាប់យកភ្លើងពី Battery(9v-12v)។

9.13. Motor

គឺជាប្រភេទ Motor ដែរមានតង់ស្យុង12V សម្រាប់បង្វិលកង់ឱ្យរ៉ូបូតទៅទៅមុខ ទៅក្រោយ ទៅឆ្វេង និង ទៅស្តាំ។

9.14. Ultrasonic Sensor

គឺជាឧបករណ៍អេឡិចត្រូនិកដែលវាស់ចម្ងាយនៃវត្ថុគោលដៅដោយបញ្ចេញរលកសំឡេង ultrasonic ហើយបំលែងសំឡេងដែលបានឆ្លុះបញ្ចាំងទៅជាសញ្ញាអគ្គិសនី។

9.15. TCRT5000 Sensor

ឧបករណ៍នេះប្រើសម្រាប់ចាប់ពណ៌ និងចំងាយ។ ម៉ូឌុលឧបករណ៍ចាប់សញ្ញានេះមាននៅលើឧបករណ៍បញ្ជូន IR និងឧបករណ៍ទទួល IR។ ឧបករណ៍បញ្ជូនបន្ត IR បញ្ចេញសញ្ញា IR ជាបន្តបន្ទាប់ដែលបន្ទាប់មកត្រូវបានឆ្លុះបញ្ចាំងដោយឧបសគ្គហើយ បន្ទាប់មកត្រូវបានរកឃើញដោយ

អ្នកទទួល IR។ TCRT5000 ម៉ូឌុលឧបករណ៍ឆ្លុះបញ្ចាំងអ៊ីនហ្វ្រារ៉េដត្រូវបានប្រើជាញឹកញាប់ក្នុងរ៉ូបូត Line Follower Robot ប្រើជាចង្កូតពីព្រោះម៉ូឌុលនេះអាចដឹងបានប្រសិនបើផ្ទៃខាងលើមានពណ៌សឬខ្មៅ។

សំណួរ និង លំហាត់

1. ចូរធ្វើប្រមាណវិធីលេខប្រព័ន្ធគោល ២ដូចខាងក្រោម៖
   a. 11110111 + 1011
   b. 11001011 – 11011011
   c. 10011 * 1011
   d. 11001100 / 1100
2. ចូរធ្វើប្រមាណវិធីលេខក្នុងប្រព័ន្ធគោល ១៦ ខាងក្រោម៖
   a. 35FAH + 9A3FH
   b. BCDCH - 8359H
   c. AB1H * 4AH
   d. 1234H * ACH
3. ចូរធ្វើការបំលែងពីប្រព័ន្ធគោល ២ ទៅប្រព័ន្ធគោល ១៦
   a. 1010110001
   b. 1110101
4. តើលេខខាងក្រោមមួយណាលេខ Hexa, Binary ឬ លេខ Decimal?
   a. 0x123d d. 345
   b. 1236 e. 100100b
   c. 2344H f. ABDH
5. ចូរធ្វើប្រមាណវិធីរក 1′ complement ក្នុងប្រមាណវិធី 16 bits:
   a. 10000111 01001010
   b. 34h
6. ចូររកប្រមាណវិធីរក Logical
   c. 1001 and 1110
   d. 1001110 or 10001111
   e. Not 10000000 1001
7. ចូរប្រមាណវិធីរក 2′ s complement
   a. 10000111 01001010
   b. 34h (byte)
   c. 134AH
8. ចូរបំលែងលេខ Hexadecimal ខាងក្រោមទៅជាលេខ Decimal System។
   a. 16AH
   b. D9FEH
   c. 34DAH

9. ចូរធ្វើការបំលែងលេខ Binary ខាងក្រោមទៅជាលេខ Decimal System។
   a. 1001100001
   b. 10001111111111
10. ចូរធ្វើការបំលែងលេខ Binary ខាងក្រោមទៅជាលេខ Decimal System។
   c. 1001100001
   d. 10001111111111
11. ការបំលែងលេខ Dec=97.55 ទៅជាទំរង់ទសភាគដោយអនុវត្តន៍តាមទំរង់ Single_precision
12. ចូរបំលែងលេខតួអក្សរខាងក្រោមទៅជាលេខ Scan និង ASCII កូដ
   a. Soure = " Computer Architecture And IoT "
   b. Data = " 123456789Abcd@#;"
13. ចូរពន្យល់នៅចំណុចតាមរូបខាងក្រោម (a), (b), (c), (d) និង (e)

(a)

3		0	0	1	1
+9		+1	0	0	1
12		1	1	0	0

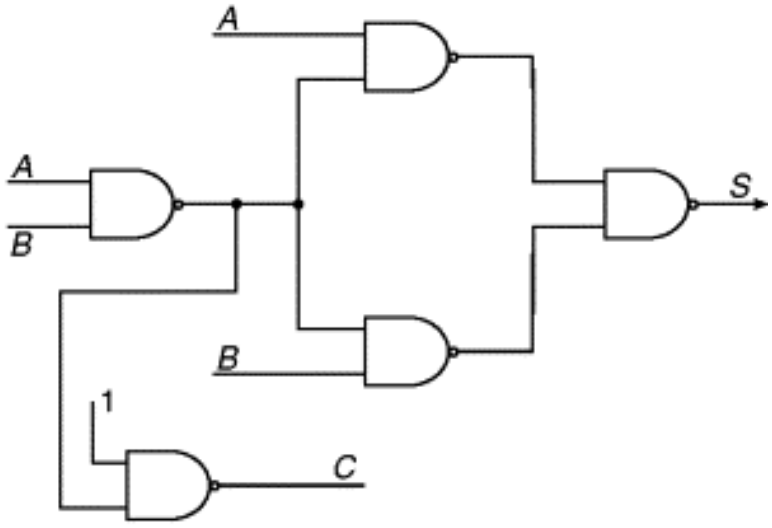
(b)

A	0		0	1	1
B	+0		+1	+0	+1
	0		1	1	0 (carry 1)

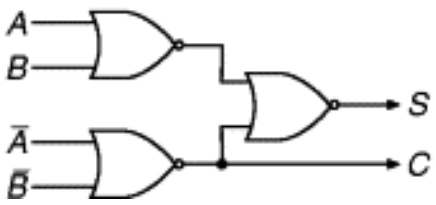
(c)

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(d)

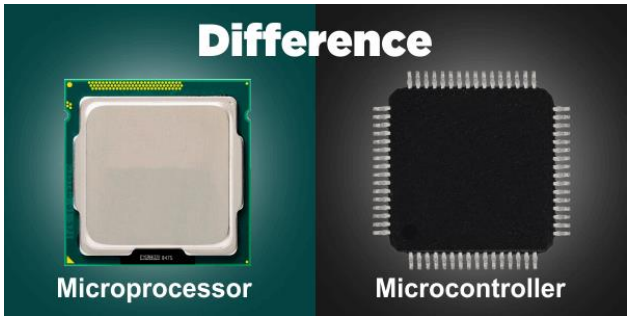


(e)



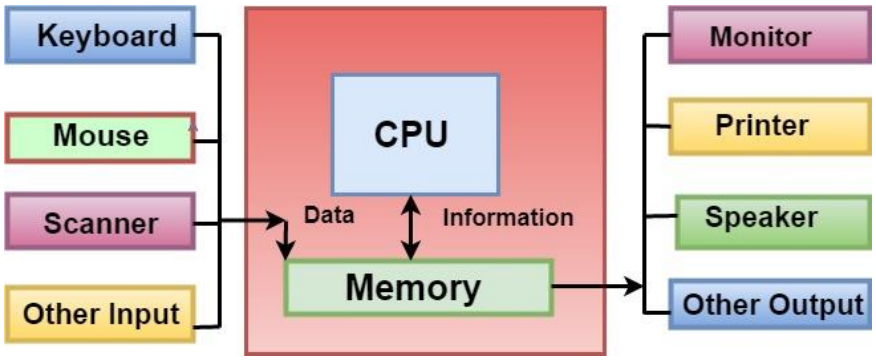
មេរៀនទី ២ ការណែនាំឱ្យស្គាល់ ស្ថាបត្យកម្មកុំព្យូទ័រ

កុំព្យូទ័រស្ថាបត្យកម្ម(Computer Architecture)គឺជាមុខវិជ្ជាមួយសិក្សាយ៉ាងលំអិតទៅលើផ្នែកស្ថាបត្យកម្ម នៃ Microprocessor និង Microcontroller ដែលវាទាំងពីរជាខួរក្បាលរបស់ម៉ាស៊ីនកុំព្យូទ័រ,ទូរស័ព្ទដៃ, រ៉ូប៉ូតទិក, ឧបករណ៍វៃឆ្លាត(Smart phone) និងឧបករណ៍អេឡិចត្រូនិចមួយចំនួនទៀតដែលអាចដំណើរការទៅបានដោយសារពាក្យបញ្ជា (Instructions) ដែលមាននៅក្នុងវាជាអ្នកប្រាប់។ ហើយសំនុំពាក្យបញ្ជានេះវាកម្មវិធីដែលយើងបានសរសេរដោយប្រើភាសា Assembly, Arudino និង Python។ នៅក្នុងមេរៀននេះក៏សិក្សាអំពីស្ថាបត្យកម្មរបស់អង្គចងចាំ និងឧបករណ៍បញ្ចូល ឬបញ្ចេញផងដែរ។ ការសិក្សាសំខាន់អំពីឧបករណ៍ទាំងអស់ដែលបានរៀបរាប់ខាងលើជាប់ទាក់ទងនឹងស្ថាបត្យកម្ម(Architecture), ការសរសេរកូដ(Programming), ការបង្កើតទាំងផ្នែករឹង(Hardwares) និងផ្នែកទន់(Softwares)។



ការសិក្សាលើឧបករណ៍មានដូចជាស៊ីភីយូ(CPU), អង្គចងចាំ(Memory), BUS, ឧបករណ៍ផ្ទុកឬបញ្ចេញព័ត៌មាន (I/O), ផត(PORT) និងឌីជីថលអេឡិចត្រូនិចដើម្បីធ្សាំរាបង្កើតបានឧបករណ៍មួយ ឬច្រើន និងធ្វើឱ្យវាអាចដំណើរការទៅបានដោយសារផ្នែកទន់។ អ្នកប្រើប្រាស់ ឬអ្នកសរសេរកម្មវិធីអាចបញ្ចូលនូវទិន្នន័យ និងពាក្យបញ្ជាដោយផ្ទាល់ទៅក្នុងស៊ីភីយូ(CPU) ដើម្បីឱ្យវាអាចធ្វើការប្រតិបត្តិកូដដែលមាននៅក្នុងឈីប(Chip) ដូចរៀបរាប់ខាងលើដើម្បីឱ្យវាធ្វើការបញ្ជាទៅលើឧបករណ៍ដែលបានតភ្ជាប់ឱ្យដំណើរការ និងចេញជាលទ្ធផលណាមួយដែលចង់បាន។ រូបខាងក្រោមបង្ហាញពីប្រព័ន្ធកុំព្យូទ័រមួយ ដែលមានផ្នែករឹងមានដូចឧបករណ៍បញ្ចូល ប្រតិបត្តិ និងបញ្ចេញលទ្ធផល។

- ឧបករណ៍បញ្ចូល: Keyboard, Mouse, Scanner, Microphone
- ឧបករណ៍ប្រតិបត្តិ: Microprocessor
- បញ្ចេញលទ្ធផល



1. **កុំព្យូទ័រស្ថាបត្យកម្ម**(Computer Architecture)

Computer Architecture ឬយើងហៅថាកុំព្យូទ័រស្ថាបត្យកម្មវាធ្សាំដោយពាក្យពីរគឺ:

- ស្ថាបត្យកម្ម(**Architecture**): គឺជាការរចនាកុំព្យូទ័រ រ៉ូបូត ឧបករណ៍សិប្បនិម្មិត(AI) និងឧបករណ៍អេឡិចត្រូនិចមួយចំនួនទៀតមានដូចជា ទូរស័ព្ទដៃ ឬ អ៊ីនធឺណិតនៃការគិត(IoT) ឱ្យទៅជាប្រព័ន្ធមួយដែលមានផ្នែកវឹង និងផ្នែកទន់ ទៅតាមគោលបំណងណាមួយជាក់លាក់។

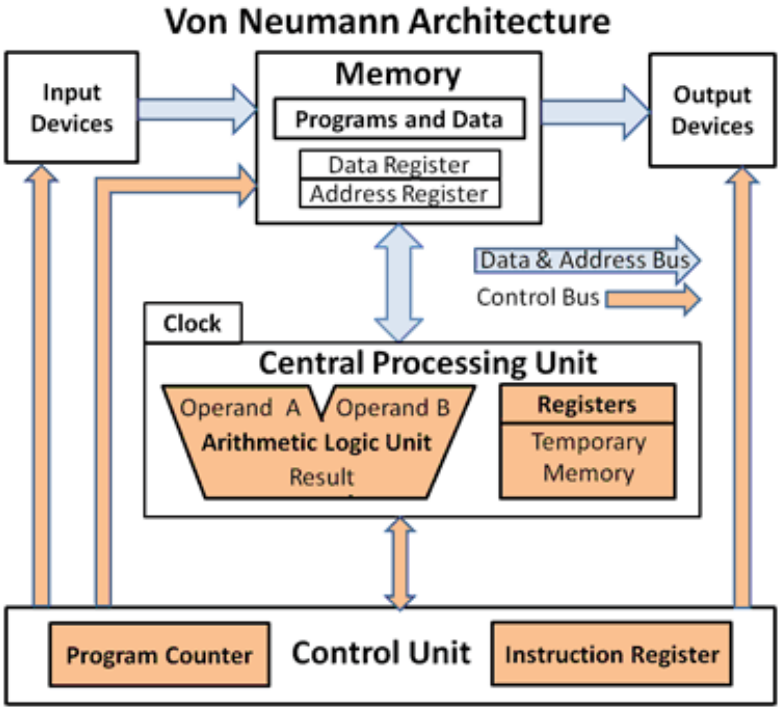
- កុំព្យូទ័រ(**Computer**) គឺជាម៉ាស៊ីនឧបករណ៍អេឡិចត្រូនិចដែលអាចបញ្ចូល, ដំណើរការ, បញ្ចេញលទ្ធផល និងអាចរក្សាទុកទិន្នន័យ។

- ការបញ្ចូល(Input)ទិន្នន័យទៅឱ្យកុំព្យូទ័រ។ ការបញ្ចូលអាចផ្តល់ដោយមនុស្សម្នាក់ដោយប្រើ Keyboard ឬ ឧបករណ៍ផ្សេងៗទៀត។ ឧទាហរណ៍ការបញ្ចូលទិន្នន័យសម្រាប់ធ្វើការគណនា និងពាក្យបញ្ជាក្នុងកម្មវិធីមួយសម្រាប់ដំណើការឱ្យចេញលទ្ធផលមួយ។ ការដំណើរការទិន្នន័យឱ្យទៅជាព័ត៌មាន ដូចជាការធ្វើប្រមាណបូក ដក គុណ ចែក, ដំណើរការកម្មវិធី, ការផ្លាស់ប្តូរទិន្នន័យពីកន្លែងមួយទៅកន្លែងមួយទៀតទៅតាមពាក្យបញ្ជារបស់អ្នកប្រើប្រាស់។ ការដំណើរការទាំងអស់នេះគឺធ្វើដោយស៊ីភីយូរបស់កុំព្យូទ័រ។ ការបង្ហាញព័ត៌មាន(Output) គឺជាលទ្ធផលសម្រេចមួយទៅឱ្យអ្នកប្រើប្រាស់ដោយប្រើអេក្រង់ម៉ូនីទ័រ ឬ ព្រីនធ័រជាដើម។ ចំណែកការផ្ទុកទិន្នន័យ និង ពាក្យបញ្ជាសម្រាប់ប្រើទៅអនាគតដោយប្រើឌីស។

ស្ថាបត្យកម្មកុំព្យូទ័រគឺជាសំណុំនៃវិធាន និងវិធីសាស្ត្រដែលពិពណ៌នាអំពីឈ្មោះ, មុខងារ, ការរៀបចំ និងការអនុវត្តប្រព័ន្ធកុំព្យូទ័រ រ៉ូបូតទិច និងឧបករណ៍សិប្បនិម្មិត ដើម្បីឱ្យមានទំនាក់ទំនងគ្នារវាងផ្នែកវឹង និងផ្នែកទន់។ *ឬអាចនិយាយដោយខ្លីថា គឺជាការចនាកុំព្យូទ័រ រ៉ូបូតឧបករណ៍ឆ្លាតវៃ ឬ ទូរសព្ទដៃអាចដំណើរការទៅបានដោយមានផ្នែកទន់ជាអ្នកប្រាប់ឱ្យផ្នែកវឹង។ ស្ថាបត្យកម្មនៃកុំព្យូទ័រគឺជាប់ពាក់ព័ន្ធទៅនឹងការសិក្សាសំនុំពាក្យបញ្ជាស្ថាបត្យកម្ម(instruction set architecture) , ប្លុកដ្យាក្រាមនៃស៊ីភីយូ និង ការសរសេរកាសា Assembly ដោយប្រើពាក្យបញ្ជា និងផ្នែកវឹងដោយផ្ទាល់ក្នុងស៊ីភីយូមួយ។*

ការសិក្សានៅក្នុងមុខវិជ្ជានេះគឺទាក់ទងសំខាន់ការដំឡើងកុំព្យូទ័រ, រ៉ូបូត(Robot), ឧបករណ៍វៃឆ្លាត និងឧបករណ៍ផ្សេងៗទៀតដោយខ្លួនឯង និងធ្វើនូវអ្វីដែលមានស្រាប់ រួមមានទាំងការសរសេរកម្មវិធីដើម្បីឱ្យឧបករណ៍ទាំងនេះអាចប្រតិបត្តិការទៅតាមការបញ្ជារបស់មនុស្ស ដោយប្រើកម្មវិធីនៅលើកុំព្យូទ័រ ទូរស័ព្ទ និងឧបករណ៍ដទៃមួយចំនួនទៀត។

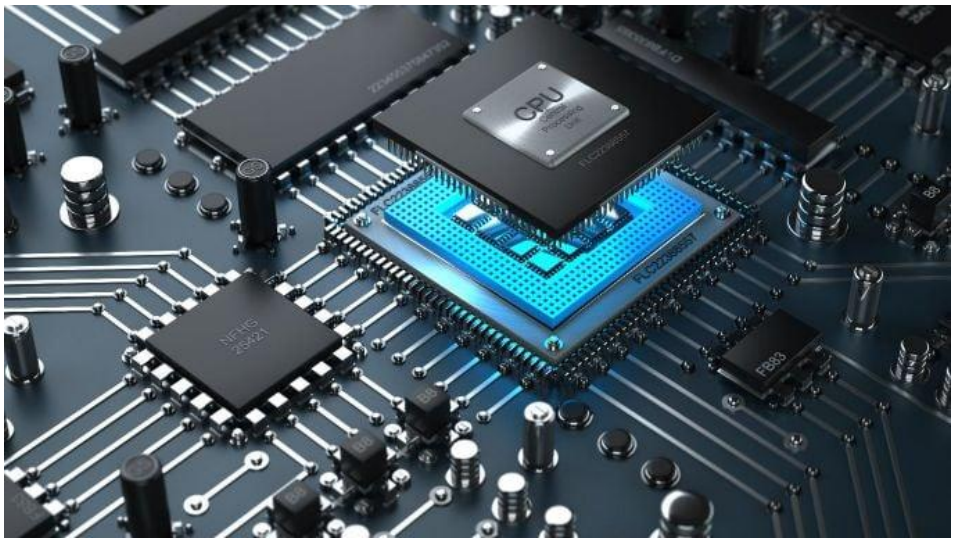
រូបខាងក្រោមបង្ហាញពីស្ថាបត្យកម្មកុំព្យូទ័រ:



2. **និយមន័យ** Microprocessor (Definition Of Microprocessor)

Microprocessor គឺជាផ្នែកយ៉ាងសំខាន់មួយនៃស្ថាបត្យកម្មកុំព្យូទ័រដោយគ្មានវាអ្វីៗនឹងមិនអាចដំណើរការបានឡើយនៅលើកុំព្យូទ័រមួយ។ វាប្រើប្រាស់នូវប្រព័ន្ធតក្កវិទ្យាដូចដែលបានប្រើនៅក្នុងស៊ីភីយូនៃកុំព្យូទ័រ។ វាបង្កើតឡើងពីបច្ចេកវិទ្យានៃការបង្រួមសៀគ្វី នោះយើងអាចនិយាយបានថាវាជាផ្នែកប្រតិបត្តិ ឬជាអ្នកបញ្ជានៅក្នុងទំរង់នៃសៀគ្វីមួយ។ វាដូចទៅនឹងស៊ីភីយូត្រង់ថា Microprocessor មានសៀគ្វីឌីជីថល សម្រាប់ធ្វើការគ្រប់គ្រង, ដំណើរការទិន្នន័យ និងធ្វើការគណនាទិន្នន័យតាមរយៈពាក្យបញ្ជាជាអ្នកប្រាប់ និងជាអ្នកត្រួតពិនិត្យ។ ម្យ៉ាងវិញទៀតស៊ីភីយូបង្កើតពី IC (Integrated circuit) ចាប់ពីប្រភេទ Large-Scale ឬ Very Large-scale Integrated circuit (VLSC), ពេលនោះស៊ីភីយូត្រូវបានគេហៅឈ្មោះថ្មីមួយទៀតហៅថា Microprocessor ព្រោះវាត្រូវបានបង្រួម CU, ALU និង Register និងឧបករណ៍ផ្សេងៗ ចូលទៅក្នុងបន្ទះឈីបតែមួយ។

អាចនិយាយម្យ៉ាងទៀតថា Microprocessor គឺជាការបង្រួម CU, ALU, Register និងសមាសភាគអេឡិចត្រូនិចមួយចំនួនទៀតដូចជា Cache Memory ឬ Math Coprocessor ចូលទៅក្នុងឈីបតែមួយ ។



វាត្រូវបង្កើតសម្រាប់ប្រើជាមួយនឹង RAM, ROM, I/O port, Timer Clock។ល។ ដើម្បីដំណើរកម្មវិធី និងធ្វើប្រមាណវិធីផ្សេងៗ តែឧបករណ៍ទាំងអស់នៅទីតាំងផ្សេងៗ ពីគ្នា។

មុខងារសំខាន់របស់ Microprocessor គឺ៖

- ធ្វើការគណនាប្រមាណវិធីនព្វន្ត, តក្កវិទ្យា និង ការសំរេចចិត្ត
- បញ្ជាឱ្យធ្វើការបំលាស់ទីទិន្នន័យពី ផ្នែកមួយទៅផ្នែកមួយទៀតនៃកំព្យូទ័រ
- ធ្វើការបកប្រែពាក្យបញ្ជា និងធ្វើការប្រតិបត្តិ
- ទទួលខុសត្រូវក្នុងការបញ្ចេញទិន្នន័យ និង ទទួលផ្ទុកទិន្នន័យនៅលើឌីស ឬ អង្គចងចាំ

2.1. **ការប្រើប្រាស់** Microprocessor

ការប្រើប្រាស់ Microprocessor (CPU) សម្រាប់ធ្វើការត្រួតពិនិត្យ និង ដំណើរការទៅលើឧបករណ៍ផ្សេងៗមានដូចជា៖

- សមាសភាគកុំព្យូទ័រមាន: Scanner, ម៉ាស៊ីនព្រីន, ក្តារចុច, អង្គប្រព័ន្ធ (CPU នៅក្នុង System Unit), Monitor និងឧបករណ៍ទៀតមួយចំនួនទៀតរបស់កុំព្យូទ័រ។
- ឧបករណ៍អេឡិចត្រូនិចប្រើប្រាស់ក្នុងផ្ទះ: វិទ្យុ(Radio), ទូរទស្សន៍ (Television), ម៉ាស៊ីនចាក ថាសចំរៀង, ម៉ាស៊ីនបោកខោអាវ ឬ ម៉ាស៊ីនត្រជាក់។ល។
- ឧបករណ៍ស្វ័យប្រវត្តប្រើក្នុងការិយាល័យ: ម៉ាស៊ីនគិតលេខ,ទូរស័ព្ទ, ម៉ាស៊ីនទូរសារ និង ម៉ាស៊ីន ថតចំលង។
- រោងចក្រស្វ័យប្រវត្តិ: គ្រឿងយន្តស្វ័យប្រវត្តិ និង មនុស្សយន្ត។
- សាលពិសោធន៍ស្វ័យប្រវត្តិ ដូចជាម៉ាស៊ីនពិសោធន៍ស្វ័យប្រវត្តិសម្រាប់វិភាគរូបមន្តផ្សេងៗ ដូចជា វិភាគគីមីនុយក្លេអ៊ែក្នុងរូបវិទ្យា។
- Raspberry pi និង AI robots

2.2. **ការអភិវឌ្ឍន៍របស់** Microprocessor

Intel 4004 ជា Microprocessor ដំបូងដែលបានបង្កើតនៅថ្ងៃទី 15 ខែ វិច្ឆិកា ឆ្នាំ 1971 ប្រើជាមួយនឹង**កុំព្យូទ័រនៅជំនាន់ទី ៣ និង**មានសមាសភាពសំខាន់ៗដូចខាងក្រោម៖

- ទំហំខ្សែបញ្ជូនទិន្នន័យនៅក្នុង (Internal Data **Bus**): **4 bits**
- ទំហំខ្សែបញ្ជូនទិន្នន័យនៅក្នុងខាងក្រៅ (External Data Bus**): 8 bits**
- ទំហំខ្សែអាស័យដ្ឋាន (Address Bus)**: 9 bits និង** Physical Memory: **512** Bytes
- MHz : **0.1 ~ 0,5 ;** ចំនួន transistor: **2300**

✧ **Internal Data Bus:** គឺជាផ្លូវសម្រាប់ធ្វើការនាំទិន្នន័យ និងពាក្យបញ្ជាដែលមានទីតាំងនៅក្នុង ស៊ីភីយូ។ វាមានទំហំកាន់តែធំនោះមានន័យថាមានខ្សែភ្លើងកាន់តែច្រើនធ្វើឱ្យការនាំទិន្នន័យបានច្រើនក្នុងពេលតែមួយជាមធ្យមវាមានទំហំចាប់ពី 8 ប៊ីត (មាន៨ ខ្សែភ្លើង) ទៅ 64 ប៊ីត (មាន៦៤ ខ្សែភ្លើង)។ វាគឺជាផ្លូវទៅមកបានមានន័យថាអាចប្រើផ្លូវនេះដើម្បីទទួល និងបញ្ជូនទិន្នន័យ។ កំលាំងនៃការប្រតិបត្តិកុំព្យូទ័រអាស្រ័យទៅនឹងទំហំ Buses, ដោយ 8-bit data bus អាចនាំទិន្នន័យបានម្តង ៨ ប៊ីតក្នុងពេលតែមួយ។ ១៦ប៊ីតអាចនាំទិន្នន័យម្តងចំនួន 2 bytes ឬលើសពីនេះអាចនាំបានកាន់តែច្រើន។

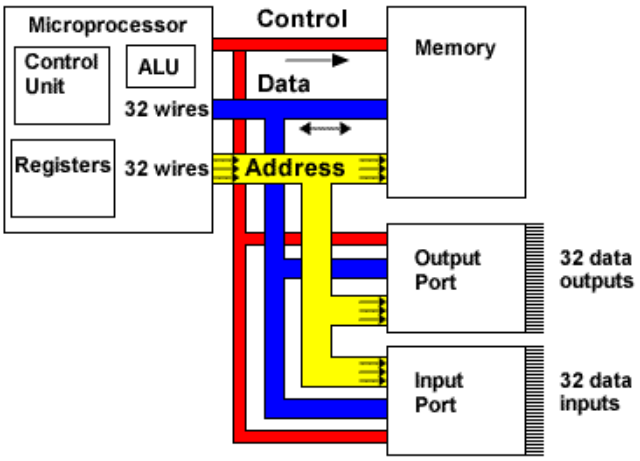
✧ **External Data Bus:** គឺជាផ្លូវសម្រាប់នាំទិន្នន័យ និង ពាក្យបញ្ជាដែលមានទីតាំងនៅផ្នែកខាងក្រៅស៊ីភីយូ។ ដូចជា Bus ភ្ជាប់ពីអង្គចងចាំទៅស៊ីភីយូ ឬពីអង្គចងចាំទៅឌីសជាដើម។

Data Bus ភ្ជាប់ពីស៊ីភីយូទៅកាន់អង្គចងចាំ និង ឧបករណ៍ផ្សេងៗដែលស្ថិតនៅក្រៅនោះហៅថា External Data Bus។ ដូចជាខ្សែបន្ទះសម្រាប់ធ្វើការចំលងទិន្នន័យចេញពី Mainboard ទៅកាន់ Hard disk ក៏ជាប្រភេទ External Data Bus ដែរ។

✧ **Address Bus:** គឺជាផ្លូវសម្រាប់នាំលេខអាសយដ្ឋានរបស់ទិន្នន័យ និង ពាក្យបញ្ជា ដែលស្ថិតក្នុងអង្គចងចាំ។ វាមានចំនួនប៊ីតកាន់ច្រើនធ្វើឱ្យចំនួនទីតាំងកាន់តែច្រើនឬទំហំអង្គចងចាំកាន់តែធំ។ វាមានទិសដៅតែមួយមានន័យថាស៊ីភីយូអាចប្រើផ្លូវនេះ ដើម្បីនាំអាសយដ្ឋាន ចេញក្នុងខ្លួនវាទៅកាន់ RAM, ROM និងឧបករណ៍ផ្សេងៗទៀត។

	Address	Location	Byte
0h=0	**0000b**	**01000001=A**	-Start Address =0000b គឺជា
1h	0001b	01000010=B	អាសយដ្ឋានចាប់ផ្តើមនៃ Memory
2h	0010b		-មួយទីតាំងអាចផ្ទុកបានមួយ Byte
3h	0011b		-បើ Memory មានលេខ Address
4h	0100b		ចំនួន១៦ ប៊ីតនោះចំនួនទីតាំងស្មើ
5h	0101b		នឹង ១៦ byte
6h	0110	a=01100001b	-ពេលអក្សរ A ត្រូវផ្ទុកត្រូវការមួយ
7h	0111		ទីតាំង
8h	1000	9=00111001b	
9h	1001		
Ah	1010		
Bh	1011		
Ch	1100		
Dh	1101		-អាសយដ្ឋានចុងក្រោយនៃអង្គចង
Eh	1110		ចាំហៅថា End Address
Fh=15	1111b		End Addres =1111=10000

✧ **Control Bus:** គឺជាផ្លូវសម្រាប់នាំពាក្យបញ្ជាពីស៊ីយូ(CU) ទៅកាន់ RAM ឬឧបករណ៍ផ្សេងៗ។ ពាក្យបញ្ជាមានពីគឺ Read/ Write Signal និង Command Signal។ Read/ Write Signal ជាព័ត៌មានសម្រាប់ឱ្យ RAM, Register បញ្ចេញ និង បញ្ចូលពត៌មាន។ Command Signal ជាព័ត៌មានសម្រាប់ឱ្យ ALU ធ្វើប្រមាណវិធីមួយជាក់លាក់ផងដែរ។ វាគឺជាផ្លូវសម្រាប់អោយកុំព្យូទ័រដែលប្រើដោយស៊ីភីយូដើម្បីទំនាក់ទំនងជាមួយឧបករណ៍ដែលមាននៅក្នុងកុំព្យូទ័រ។ នេះកើតឡើងតាមរយៈការតភ្ជាប់រាងកាយដូចជាខ្សែកាបឬសៀគ្វីដែលបានបោះពុម្ពហើយ។



✧ **Signal:** ជាព័ត៌មាន ឬក៏ជាសញ្ញាមួយប្រើសម្រាប់ឱ្យឧបករណ៍មួយដឹងអ្វីដែលត្រូវធ្វើ, ដូចជាដំណើរការផ្ទុកព័ត៌មាន ក្នុង **RAM** ហៅថា Write។ ពេលយកព័ត៌មានចេញពី **RAM** ហៅថា **Read**។

✧ **Physical Memory**: គឺជាប្រភេទ Memory ដែលគេផលិតពីរោងចក្រមក។ វាមាននាទីផ្ទុកតម្លៃទិន្នន័យ និង ពាក្យបញ្ជាប្រើឱ្យស៊ីភីយូពេលធ្វើប្រមាណវិធីមួយ។ រូបមន្តខាងក្រោមនេះប្រើសម្រាប់រកទំហំរបស់ Physical Memory:

$$PM(RAM) = 2^x \text{ Bytes}$$

2 ជាលេខប្រព័ន្ធគោលពីរដែលមានលេខ ១ ឬ ០

x ជាទំហំនៃលេខ Address Bus គិតថា Bits

☺.**ឧទាហរណ៍**: a. គេមាន **Address Bus = 8 bits**, ចូរគណនា **Physical Memory**?

b. គេមាន **Address bus = 20 bits,** ចូរគណនា **Physical Memory**?

ចម្លើយ

Physical Memory = 2^x Bytes

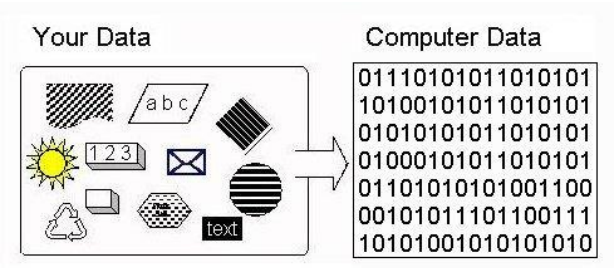
Physical Memory = 2^8 Bytes

Physical Memory = 256 Bytes

ដូច្នេះ**Physical Memory =** 256 Bytes

✧ **Transistor**: គឺជាកុងតាក់បិទឬបើកនៅក្នុងឈីបមួយ។ ពេលកុងតាក់បើកតាងដោយលេខ **១** និង កុងតាងដោយលេខ ០។ លេខ ០ និង ១ ច្រើនផ្សំចូលគ្នាបង្កើតជាទិន្នន័យ និងពាក្យបញ្ជាសម្រាប់ប្រើប្រាស់នៅក្នុងអង្គចងចាំនៃកុំព្យូទ័រ។

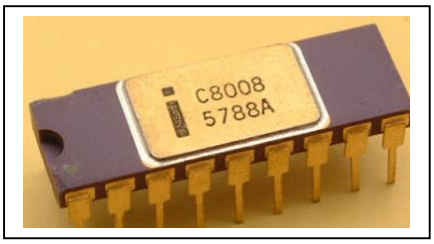
Hexa	Binary	Ram
2AH	00101010	1 Location =1 byte = 8 transitor



✧ **MHz** (Megaherz): គឺជាខ្នាតល្បឿនស៊ីភីយូប្រើដើម្បីវាស់ល្បឿនបញ្ជូនទិន្នន័យរបស់ឧបករណ៍អេឡិចត្រូនិករួមមានធាតុលេ, Bus និងនាឡិកាខាងក្នុងរបស់កុំព្យូទ័រ។ ល្បឿនគិតត្រង់ ១ លាន Cycle ស្មើនឹង

1MHz ក្នុងមួយនាទីដែលធៀបទៅនឹងល្បឿនប្រេកង់របស់ទ្រនិចនាឡិកា។ ៨០៨៦ មានល្បឿន MHz នោះមាន Cycle ស្មើ ៦លាន Cycle។

Intel 8008: គឺជាប្រភេទស៊ីភីយូត្រូវគេបង្កើតនៅឆ្នាំថ្ងៃទី ១ ខែមេសា ឆ្នាំ 1972 ប្រើជាមួយកុំព្យូទ័រលើតុដំបូងឈ្មោះ Altair 8800 ដោយមានសមាសភាពសំខាន់ៗដូចខាងក្រោម៖



- Internal Data Bus: 8 bits
- External Data Bus: 8 bits
- Address Bus: 14 bits
- Physical Memory: 16 KB
- Number of Transistor: 3000

តារាងខាងក្រោមនេះបង្ហាញពីឈ្មោះ Microprocessor សំខាន់ៗ ដែលបានវិវត្តន៍ពី Intel 8080 ទៅ Pentium IV

a.1974s Processor

Product	8080	8085	8086	8088	80286
Date	1/4/1974	12/03/1976	8/6/1978	1/6/1979	1/2/1982
Internal data bus	8	8	16	16	16
External Data Bus	8	8	16	8	16
Address Bus	16	16	20	20	24
Physical Memory	64KB	64KB	1MB	1MB	16MB
Number Transistors	4500	6500	29,000	29,000	130,000
Clock Speed(Mhz)	2~3	3~8	5~10	5~8	6~16
Data Type	8	8	8,16	8,16	8,16

b. 1985s Processor

Product	80386 DX	80386 SX	80486 DX	80486 SX	Pentium
Date	17/10/1985	16/6/1988	10/4/1989	10/4/1992	22/3/1993
Internal data bus	32	16	32	32	32
External Data Bus	32	16	32	32	32
Address Bus	32	24	32	32	32
Physical Memory	4GB	16KB	4GB	4GB	4GB
Transistors	275,000	275,000	1.2 million	1.2 million	3.1 million
Clock Speed	16~33	16~33	25~60	25~50	60~133
Data Type	8,16,32	8,16,32	8,16,32	8,16,32	8,16,32

c. 1995s Processor

Product	Pentium Pro	Pentium II	Pentium III	Pentium IV
Date	01/11/1995	07/05/1997	26/02/1999	26/02/2000
IDB	32	32	32	32
EDB	64	64	64	64
Address Bus	36	36	36	36
PM	64GB	64GB	64GB	64GB
Transistors	5.5 million	7.5million	9.5~28 million	42 million
Clock speed	133~233	233~450	450~1000	1.00~1.5GHz
Data Type	8,16,32	8,16,32	8,16,32	8,16,32

d. 2001 Processor

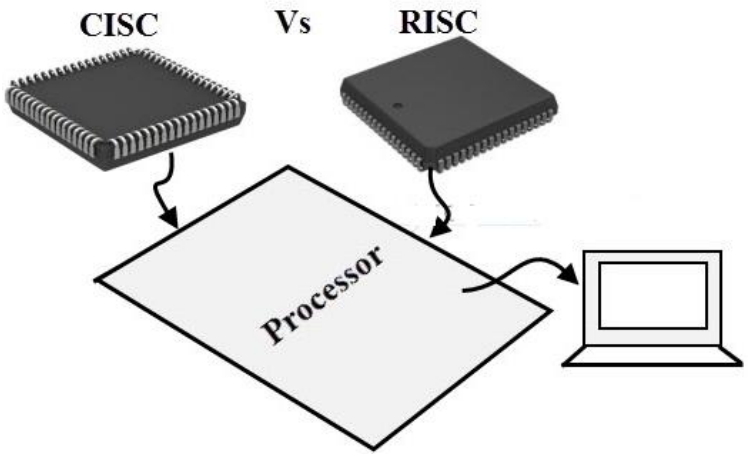
Product	Itanium M	Pentium 4E	Pentium D	Intel Cores
Date	03/2003	02/2004	26/02/2005	27/07/2006
IDB	32 bits	32 bits	64bits	64bits
Cache Memory	L1: 64KB L2: 1MB	L1: L2: 1MB	L1 L2:1 *2MB	L1 L2:4MB
System Bus	400 MHz	533MHz	533MHz	1333~1666
Transistors	77million	125 millions	230 millions	291 millions
Clock Speed	900 MHz~1.7 GHz	233~450	2.8~3.4 GHz	2.00~3.00GHz

ស៊ីភីយូបច្ចុប្បន្ននេះមានឈ្មោះ Core i5, Core i7, core i9, ឬមានឈ្មោះផ្សេងៗទៀតក្រៅពីនេះ ដែលបានអភិវឌ្ឍន៍ទៅតាមសម័យកាល និង មានល្បឿនគិតជា GH_z និងអង្គចងចាំគឺ RAM មានទំហំកាន់តែធំ។ សូមមើលលំអិតអំពីវានៅមេរៀនទី ៣។

3. CISC និង RISC

នៅដើមឆ្នាំ ១៩៨០ គ្រប់ស៊ីភីយូទាំងអស់គឺជាប្រភេទ Single-Chip ឬ ជាប្រភេទ CISC(Complex Instruction Set Computer) ត្រូវបានរៀបចំដោយមានច្បាប់ត្រឹមត្រូវ។ CISC សំដៅទៅលើស៊ីភីយូដែលមានរាប់រយពាក្យបញ្ជា (Instructions) និងរៀបចំឡើងសម្រាប់សម្រេចគ្រប់ស្ថានភាពទាំងអស់។ ដើម្បីរៀបចំស៊ីភីយូឱ្យមានពាក្យបញ្ជាច្រើននេះត្រូវប្រើប្រាស់មិនត្រឹមតែរាប់សែនត្រង់ស៊ីទ័រប៉ុណ្ណោះទេ ហើយថែមទាំងមានការស្មុគស្មាញប្រើពេលវេលាយូរ និង មានតម្លៃថ្លៃទៀតផង។ គ្រួសារស៊ីភីយូ X86 គឺប្រើប្រាស់ច្បាប់ CISC។ នៅដើមឆ្នាំ 1980, គេបានបង្កើតស៊ីភីយូថ្មីមួយប្រភេទទៀតបានចងក្រងជាច្បាប់មួយហៅថា RISC (Reduced Instruction Set Computer) ត្រូវបានវិវត្តន៍។ អ្នកគ្រាំទ្ររបស់ RISC បានជជែកមិនត្រូវប្រើពាក្យបញ្ជាច្រើនដូចដែលប្រើនៅក្នុង

CISC ទេ ហើយត្រូវបង្រួមពីពាក្យបញ្ជារាប់រយទៅ 40 Instruction ឬលើសពីនេះបន្តិចបន្តួច។ ដូច្នេះនៅដើមឆ្នាំ 1970 ត្រូវបានបង្កើតជា Single-chip RISC បានដំណើរការដោយក្រុមអ្នកស្រាវជ្រាវមួយនៅសាកលវិទ្យាល័យ California។ សព្វថ្ងៃនេះ RISC ត្រូវបានរៀបចំដោយមានច្បាប់ត្រឹមត្រូវដោយ មិនឆ្លងកាត់ពិសោធន៍រយៈយូរ។ រហូតមកដល់ឆ្នាំ ១៩៨០ មានក្រុមហ៊ុនជាច្រើនបានផលិតស៊ីភីយូដោយប្រើទ្រឹស្តី RISC នេះដើម្បីបង្កើតបានគ្រួសារ CPU ARM និង AVR ដែលសម្រាប់ប្រើប្រាស់នៅលើឧបករណ៍តូចៗមានដូចជាទូរសព្ទដៃ រ៉ូបូត និងឧបករណ៍អេឡិចត្រូនិចមួយចំនួនទៀត។



4. តារាងខុសគ្នារវាង CISC និង RISC

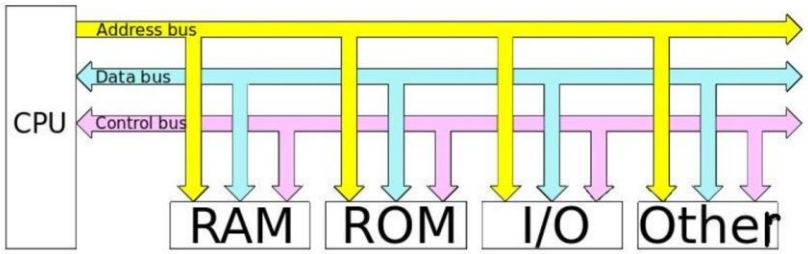
- សង្កត់ធ្ងន់ទៅលើ Hardware - ស៊ីភីយូមាន Instruction រាប់រយ - ប្រើប្រាស់ត្រង់ស៊ីស្ទ័ររាប់លាន - របៀបនៃការដោះស្រាយមានច្រើន - ការប្រើប្រាស់គឺទូលំទូលាយ - ប្រើពេលយូរឬយឺត និងមានតម្លៃថ្លៃ - ដំណើរការនៃការប្រមូលផ្តុំ Instruction គឺមានភាពលំបាក។	- សង្កត់ធ្ងន់ទៅលើ software - ស៊ីភីយូមាន Instruction តិចក្រោម ៤0 - ប្រើការត្រង់ស៊ីស្ទ័រតិច - របៀបនៃការដោះស្រាយតិច - ភាពស្មុគស្មាញនៅក្នុង Compiler - លឿន និងតម្លៃថោកជាង - ដំណើរការនៃការប្រមូលផ្តុំ Instruction គឺមានភាពងាយស្រួល។

5. CPU មានទំនាក់ទំនងជាមួយ RAM និង ROM

ចំពោះស៊ីភីយូពេលដំណើរការព័ត៌មាន ឬទិន្នន័យ និងពាក្យបញ្ជាត្រូវទាញយកពីក្នុង RAM ឬ ROM។ មុខងាររបស់ ROM នៅក្នុងកុំព្យូទ័រគឺផ្តល់ពត៌មានជាក់លាក់ និង អចិន្ត្រៃយ៍។ ពត៌មានទាំងនេះមានដូចជាតារាងសម្រាប់ធ្វើឱ្យអក្សរដិតនៅលើម៉ូនីទ័រ កម្មវិធីចាប់ផ្តើម(Start Up Program)ចាំបាច់ដែលទាក់ទងទៅដំណើរ Boot និង Restart ការកុំព្យូទ័រ។ ពេលបើកកុំព្យូទ័រកម្មវិធីចាប់ផ្តើមធ្វើការតេស្តឧបករណ៍ដែលបានដំឡើងនៅពេលបើកកុំព្យូទ័រ, រកចំនួនសរុបរបស់ RAM និងបើកកម្មវិធីប្រព័ន្ធប្រតិបត្តិការម៉ាស៊ីនដែលយើងមើលឃើញពត៌មាននៅលើម៉ូនីទ័រ។ ក្នុង ROM ក៏មានពាក្យបញ្ជាសម្រាប់ធ្វើការបញ្ជូល និងបញ្ចេញព័ត៌មានពីឧបករណ៍មួយទៅឧបករណ៍មួយទៀតផងដែរ។

ចំណែក RAM វិញគឺប្រើផ្ទុកព័ត៌មានមិនអចិន្ត្រៃយ៍ និង ព័ត៌មាននៅក្នុងវាអាចផ្លាស់ដោយអ្នកប្រើប្រាស់បាន។ វាអាចផ្ទុកកម្មវិធីប្រព័ន្ធប្រតិបត្តិការកុំព្យូទ័រ កម្មវិធីប្រើប្រាស់ និងអត្ថបទនៅពេលអ្នកប្រើប្រាស់កំពុងប្រើកុំព្យូទ័រ។

កម្មវិធីត្រូវបានផ្ទុកនៅក្នុង RAM ពេលយើងកំពុងប្រើកម្មវិធីនោះសម្រាប់ធ្វើការបញ្ជាទៅលើពាក្យបញ្ជាផ្សេងៗ ដែលត្រូវធ្វើសម្រាប់ប្រាប់ទៅស៊ីភីយូធ្វើឱ្យប្រតិបត្តិប្រមាណវិធីមួយទៅតាមបញ្ជានោះ។ CPU មិនអាចដំណើរការព័ត៌មានដោយ ផ្ទាល់ជាមួយ Disk បានទេព្រោះវាដំណើរការយឺតណាស់។ RAM និង ROM ហៅថា Primary Memory និង Disk ហៅថា Secondary Memory។



6. **តារាងខុសគ្នារវាង** RAM **និង** ROM

RAM	Differences	ROM
❖ stores during and after processing	**Data and Program**	❖ stored by manufacturer
❖ stores information temporarily	**Content**	❖ stores information permanently
❖ very fast but uses a lot of power	**Processing Time**	❖ fast but uses very little power
❖ volatility	**Volatile**	❖ non-volatile

សំណួរ និង លំហាត់

1. ដូចម្តេចដែលហៅថា Computer Architecture? តើការសិក្សាមុខវិជ្ជានេះដើម្បីធ្វើអ្វីខ្លះ?
2. ដូចម្តេចដែលហៅថា Microprocessor? ចូររៀបរាប់ពីមុខងាររបស់វា។
3. ចូរពន្យល់ពីភាពខុសគ្នារវាង CISC និង RISC Computer?
4. ចូរនិយាយពីឧបករណ៍គ្រឹះដែលផលិតបានជា CPU ចាប់ពីឆ្នាំ 1942 រហូតដល់បច្ចុប្បន្ននេះ?
5. ចូរប្រាប់ឈ្មោះឧបករណ៍ណាខ្លះដែលប្រើ Microprocessor?
6. រកទំហំ Physical Memorey បើ CPU និមួយៗមាន Address Bus ដូចខាងក្រោម
    a. 20 bit Address Bus (in MB)
    b. 32 bit Address Bus (in MB and GB)
    c. 64 bit Address Bus (in GB and TB)
7. ចូរប្រៀបធៀបលក្ខណៈខុសគ្នារវាង 80286 និង 80386?
8. តើកុំព្យូទ័រមួយអាចដំណើរការទៅបានអាស្រ័យទៅនឹងឧបករណ៍អ្វីខ្លះ? ចូរឱ្យឈ្មោះឧបករណ៍ទាំងអស់នោះ។
9. តើហេតុអ្វីបានជា CPU ប្រើប្រព័ន្ធគោលពីរ និងតក្កវិទ្យា? ចូរបញ្ជាក់ពីមូលហេតុ។
10. តើ 8080, Pentium D, Pentium Pro និង Corei7 មានប៉ុន្មាន Bit Microprocessor? ហើយលេខនិមួយ ប្រើជាមួយ Physical Memory ទំហំប៉ុន្មាន?
11. តើ Inter Core i9 មានប៉ុន្មាន bit Microprocessor? ហើយលេខ Address ចាប់ផ្តើម និង Address ចុងក្រោយពីណាទៅណាគិតលេខប្រព័ន្ធគោល ១៦?
12. ចូរគូសរូប Physical Memory តាងឱ្យ 8086, 8085 Microprocessor, ដោយ Start Address និង End Address ព្រមទាំងគណនាទំហំរបស់វាផង។

13.ចូរពន្យល់ធាតុនិមួយៗនៃស៊ីភីយូ Core i9 តាមរូបខាងក្រោមឱ្យបានក្បោះក្បាយ:

Operating System	Microsoft Window 10(64 –bit)
Model	Generic
Motherboard	ASRock Z490M Pro4
Topology	1 Processor, 10 cores, 20 Threads
Identifier	GenuineIntel Family 6 Model 165 Stepping 4
Base Frequency	3.70 Gz
Memory Size	32678 MB
Type	8, 16, 32, 64 bits

មេរៀនទី ៣ X86 Microprocessor និងពាក្យបញ្ជា

X86 គឺជាតំណាងឱ្យស៊េរីនៃឈ្មោះស៊ីភីយូ(CPU)ដែលត្រូវបានបង្កើតឡើងចាប់តាំងពីឆ្នាំ ១៩៧៨ រហូតដល់ឆ្នាំនាពេលបច្ចុប្បន្ននេះដោយចាប់ផ្តើមពី Intel 8086 ជាស៊ីភីយូគ្រឹះដំបូងដែលគេយកវាបង្កើតបាន 80186, 80286, 80386, 80486, 80586 ហៅថា Pentium, Pintium2, 3, 4, Core i3, Core i5, Core i7, Core i9 និងឈ្មោះផ្សេងៗទៀតដែលផលិតដោយក្រុមហ៊ុន Intel និងក្រុមហ៊ុនផ្សេងទៀតដែរដូចជាក្រុមហ៊ុន AMD ឬ Motorola 68000 ជាដើម។ នេះគឺជាស្ថាបត្យកម្មដែលត្រូវបានប្រើនៅក្នុងកុំព្យូទ័រលើតុ និងកុំព្យូទ័រយួរដៃភាគច្រើន។ ការប្រើប្រាស់ម៉ាស៊ីនមេ(Servers) នៅសតវត្សរ៍ទី ២១ ក៏ប្រើស៊ីភីយូនៃគ្រួសារ x86 និង x64 bits ដែរ។ ស៊ីភីយូ ៦៤ ប៊ីតគឺជាម៉ាស៊ីនមួយដែលមានទំហំពាក្យ ៦៤ ប៊ីតតម្រូវការសម្រាប់ការចងចាំនិងកម្មវិធីដែលពឹងផ្អែកខ្លាំងលើទិន្នន័យ ដូចជាកម្មវិធីឌីហ្សាញជំនួយផ្នែកកុំព្យូទ័រ(ប្រព័ន្ធCAD)ប្រព័ន្ធគ្រប់គ្រង ទិន្នន័យកម្មវិធីបច្ចេកទេសនិងវិទ្យាសាស្ត្រនិងម៉ាស៊ីនមេដែលមានដំណើរការខ្ពស់។ ស្ថាបត្យកម្មកុំព្យូទ័រ ៦៤ ប៊ីតផ្តល់នូវប្រសិទ្ធភាពខ្ពស់ជាងស្ថាបត្យកម្ម ៣២ ប៊ីតដោយដោះស្រាយព័ត៌មានទ្វេដងនៃ Clock Cycle។

1. 8086 MICROPROCESSOR

8086 / 8088 Microprocessor ចែករចេញជាផ្នែកសំខាន់ៗគឺ Bus Interface Unit (BIU), Execute Unit (EU), និង Instruction Queue ដែលធ្វើឱ្យវាអាចដំណើរការពាក្យបញ្ជា(Instructions) និង ដំណើរការប្រមាណវិធីផ្សេងៗ ដែលកើតចេញពីប្រូក្រាមមួយ។

តួនាទីសំខាន់របស់ Execute Unit ប្រតិបត្តិពាក្យបញ្ជា, ដែលបានមកពី BIU ជាអ្នកផ្តល់ពាក្យបញ្ជានេះនិងទិន្នន័យទៅឱ្យវា។ EU មាន ALU, CU, Index Register, Pointer Register និង General – Purpose Register ដែលផ្នែករឹងទាំងនេះអាចធ្វើការប្រតិបត្តិពាក្យបញ្ជា, ធ្វើប្រមាណវិធីនព្វន្ធ និងតក្កវិទ្យា និង ផ្ទុកទិន្នន័យ និង លេខអាសយដ្ឋាន(Address)។

BIU មាន Instruction Queue និង Segment Registers។ មុខងាររបស់ Instruction Queue ផ្ទុកពាក្យបញ្ជាដែលត្រូវប្រតិបត្តិ និងមានទំហំ 4 bytes(8088), ឬ 6 bytes(8086)។ ចំណែក Segment Register ផ្ទុកលេខ Address ចាប់ផ្តើមនៃផ្នែ Segment និមួយៗ។ មុខងារដ៏សំខាន់របស់ BIU គឺ ទទួលយកពាក្យបញ្ជាពី Memory និងផ្លាស់វាទៅផ្ទុកក្នុង Instruction Queue។ ក្នុងពេលកំពុងប្រតិបត្តិរបស់ 8086/8088 គឺវាប្រតិបត្តិ Instruction 1, Instruction 2 ត្រូវបានផ្ទុកនៅក្នុង Instruction Queue និងធ្វើតាមវិធីនេះជាបន្តបន្ទាប់រហូតដល់ចប់ Instrutions។

8085 Register គឺជា Internal Memory មួយដែលមានទីតាំងស្ថិតនៅក្នុង CPU មានតួនាទី ផ្ទុក Code Instruction, Data និងលេខ Address ពេលដែល CPU ដំណើរការ Instruction មួយ។ ខាងក្រោមនេះជា Register របស់ 8085

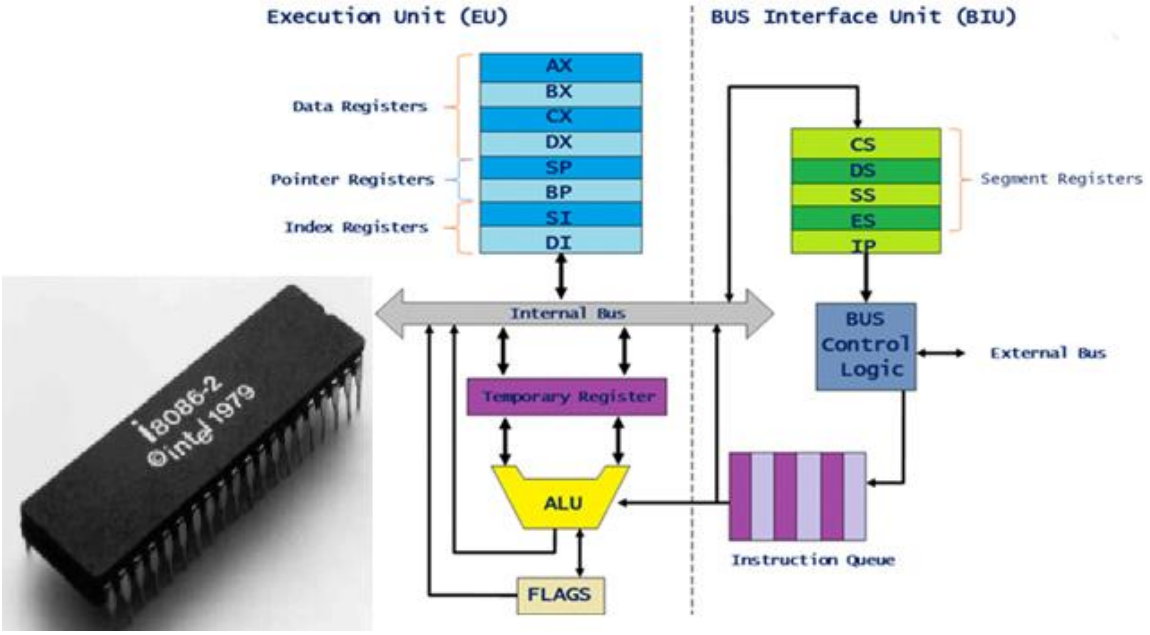
- SP (Stack Pointer)
- Mar (Memory Address)
- Temp 1 and Temp 2
- A (Accumulator)
- B (Based Register)
- C (Count Registser)
- D (Data Register)
- E (Externded Register)
- H (High Register)
- L (Low Register)
- Status Register

MAR (Memory Address Register)គឺជា Register មួយ មានតួនាទីផ្ទុកលេខ Address ដែលមកពី PC និង បញ្ចូលលេខ Address នេះទៅលើ Address Bus។ MAR ជា Register មានទំហំ bytes។

A (Accumlator) Register មានតួនាទីផ្ទុកទៅ Data និង លទ្ធផលក្រោយ ALU គណនារួចវាអាចបញ្ចេញ ឬ បញ្ចូលទិន្នន័យទៅអោយ Memory ឬALU។ A Register មានទំហំ 1 byte។ នៅក្នុង 8bit-Microprocessor មាន A, B, C, D, E, H, និង L ជា General-Purpose Register ដែលRegister និមួយៗមានទំហំ ១byte(8bit Registers)។ វាគឺជា Register ដែលប្រើជាងគេនៅក្នុង Microprocessor។ ការប្រើ B, C, D, E, H និង L register មានតួនាទីដូចទៅនឹង A register ដែរ BC, DE, និង HL ផ្សំគ្នាបង្កើតបាន 16 bit Registers។ ចំណែកឯ Status Register គឺជា Register មួយប្រើសម្រាប់ផ្ទុកចំនួន Flages ទាំងអស់ដែលបានមកពីប្រមាណវិធីមួយ រឺជាប់ទាក់ទងទៅនឹងការប្រើ Flage instruction។ Status Flags មានទំហំ 1byte និង មានទំរង់ដូចខាងក្រោម៖

8086 / **8088** មាន Register ច្រើនជាង **8085**។ Internal Register គឺមានចាប់ 8 ប៊ីតទៅ 16 ប៊ីត, ដូចជា AL, BL, CL, DL, AH, BH, CH, DH ជា 8bit Register ដោយអក្សរ "L" តាងឱ្យ Low Register និងអក្សរ "H" តាងឱ្យ Hight Register។ **AX**, **BX**, **CX**, **DX**, **SI, DI**, **SP**, **BP**, IP គឺជា 16 bit Register, ដោយអក្សរ "X" នៅខាងក្រោយតាងឱ្យ 16 bit។ ការស្គាល់ឈ្មោះ, ទំហំ និង មុខងារជាការសំខាន់សម្រាប់យកវាទៅប្រើសរសេរកូដក្នុងភាសា Assembly។

រូបខាងក្រោមនេះបង្ហាញពីប្លុកដ្យាក្រាមរបស់ស៊ីភីយូ 8086/ 8088:



ប្លុកដ្យាក្រាមរបស់ Microprocessor គឺបង្ហាញពីសំនុំផ្នែករឹងនៅក្នុងស៊ីភីយូមួយ រួមមានដូចជា Register, ALU, CU, និងឧបករណ៍ដទៃទៀតដែលមានទំនាក់ទំនងជាមួយនឹងគ្នា។ ប្លុកដ្យាក្រាមគឺធ្វើឱ្យយើងអំពីស្ថាបត្យកម្មរបស់ស៊ីភីយូមួយ។ ចំណែកគំរូកម្មវិធី ឬហៅថាកម្មវិធីម៉ូដែល(Programming Model) គឺបង្ហាញពីសំនុំ Register ទាំងអស់ដល់ឱ្យអ្នកប្រើប្រាស់ ឬ អ្នកសរសេរកម្មវិធីដើម្បីស្គាល់ពីឈ្មោះ ទំហំ និងចេះប្រើ Registers ទាំងនេះផ្ទុកទិន្នន័យឬពាក្យបញ្ជានៅពេលសរសេរកម្មវិធីក្នុងភាសា Debug ឬ Assembly នៅពេលយើងធ្វើការជាមួយវានោះត្រូវប្រើប្លុកដ្យាក្រាម និង Programming Model ដែលគេបង្កើតពីរោងចក្រមក ហើយយើងមិនអាចកែប្រែវាបានឡើយ។

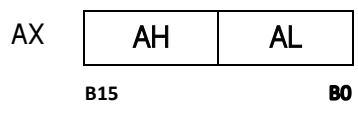
- **Control Logic** ជាផ្នែកមួយដ៏សំខាន់បំផុតរបស់ស៊ីភីយូជាទទួលខុសត្រូវក្នុងការរៀបចំបកប្រែពាក្យបញ្ជាដើម្បីធ្វើការបញ្ជា ដឹកនាំ និងកុងត្រូលលើ Memory, ឧបករណ៍ I/O, ALU ឬ Register ឱ្យដំណើរការ។ វាក៏មានតួនាទីបញ្ចូលសញ្ញាបញ្ជាធ្វើការត្រួតពិនិត្យដំណើរការទិន្នន័យដែលធ្វើការបំលាស់ពីទីពីឧបករណ៍មួយទៅឧបករណ៍មួយទៀត និង មានទំនាក់ទំនងទៅនឹង Flags ដើម្បីឱ្យវាអាចដំណើរការលក្ខខណ្ឌ និងរង្វិលជុំនៃការធ្វើប្រមាណវិធីមួយ។
- **Arithmetic Logic Unit**(ALU) ជាផ្នែកទី២ នៃស៊ីភីយូប្រើសម្រាប់ធ្វើប្រមាណវិធីនព្វន្ធ (+, -, *, /) តក្ក (ឈ្នាប់ឬ, នឹង, មិន) និងប្រតិបត្តិការប្រមាណវិធីផ្សេងទៀតដូចជាការប្រៀបធៀប រំកិលលេខគោលពីរ ឬ រង្វិលជុំជាដើម។ ក្រោយពេលវាធ្វើប្រមាណរួចមាន លទ្ធផលពីរ គឺលទ្ធផលនៃប្រមាណវិធី និង Flags(CF, DF, TF, SF, IF, ZF, OP)។
- **Register** គឺជាផ្នែកទី៣ដែលជាអង្គចងចាំមួយដែលមានទីតាំងស្ថិតនៅក្នុងស៊ីភីយូប្រើសម្រាប់ផ្ទុកពាក្យបញ្ជា, ទិន្នន័យ និងលេខអាសយដ្ឋាន ពេលដែលស៊ីភីយូប្រតិបត្តិកម្មវិធី ឬធ្វើប្រមាណវិធីមួយ។ ខាងក្រោមនេះជាឈ្មោះ Register របស់ស៊ីភីយូ 8085 ដែលជាស៊ីភីយូនេះប្រើមុខស៊ីភីយូ 8086 ហើយ Register និមួយៗមានទំហំតែ ៨ ប៊ីត ។

CPU 8085	CPU 8086
▪ SP (Stack Pointer)	▪ SP (Stack Pointer)
▪ Mar (Memory Address)	▪ IP (Intruction Pointer)
▪ Temp 1 and Temp 2	▪ SI(Source),DI(Destination)
▪ A (Accumulator)	▪ AX(Accumulator)
▪ B (Based Register)	▪ BX(Base Register)
▪ C (Count Register)	▪ DX(Data Register
▪ D (Data Register)	▪ CX(Count Register)
▪ E (Externded Register)	▪ CS(Code Segment)
▪ H (High Register)	▪ DS(Data Segment)
▪ L (Low Register)	▪ ES(Extra Segment), SS
▪ Status Register	▪ Flags Register

1.1. General-Purpose Register (AX, BX, CX, DX)

1.1.1. AX (Accumulator Register Extended)

AX Register បានការអភិវឌ្ឍមកពី Register របស់ស៊ីភីយូ A Register នៃ8085 ប្រើប្រាស់សម្រាប់ប្រមាណវិធីដែលជាប់ទាក់ទងទៅនឹងបញ្ចូល ឬបញ្ចេញទិន្នន័យ និង ប្រមាណវិធីនព្វន្ធជាច្រើនទៀត។ ចំពោះប្រមាណវិធីចែកលេខ, គុណលេខ និង ធ្វើការបកប្រែពាក្យបញ្ជានោះត្រូវប្រើ AX ជាចាំបាច់។ ពាក្យបញ្ជាខ្លះដែលបង្កើតពីកូដមានប្រសិទ្ធភាពក៏ជាប់ទាក់ទងប្រើ AX។ វាផ្សំពី AH (Accumulator High) = 1byte និង AL (Accumlator Low)= 1Byte និង វាមានទំហំស្មើនឹង 2 bytes។



1.1.2. Base Register (Bx)

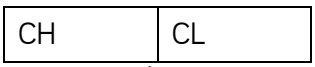
BX គឺជាប្រភេទ Register មួយទៀតដែលផ្សំពី BH និង BL ដោយវាទាំងពីរមួយៗមានទំហំមួយបៃ។ ចំណែក BX មានទំហំ 2 បៃ។ មុខងារដ៏សំខាន់របស់វាគឺ៖

- ផ្ទុកតម្លៃទិន្នន័យនិង លទ្ធផលនៃការធ្វើប្រមាណវិធីនព្វន្ត(+, -, *, /) និងតក្កវិទ្យា (មានឈ្មោះ and, or, not)
- ឆ្ពោះទៅកាន់លេខអាសយដ្ឋាន (**Offset Address**) នៃទិន្នន័យដែលមាននៅក្នុងផ្នែក Data Segment

BX = 16 bits , BH = 8 bits And BL = 8 bits

1.1.3. Count Register (CX)

CX គឺជា Register ផ្សំពី CH និង CL ហើយមួយៗមានទំហំ ១ Byte ដែលបង្កើតបាន Cx ដែលមានទំហំ ២ Byte។

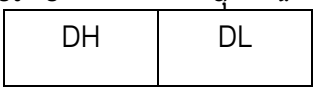


CX = 16 bits =2 bytes, CH = 8 bits និង CL = 8 bits

- ផ្ទុកតម្លៃទិន្នន័យបានដូចការប្រើប្រាស់ AX ដែរ
- ផ្ទុកចំនួនវង្វិលជុំដែលត្រូវធ្វើប្រមាណវិធីដែលមាន n ចំនួន
- ផ្ទុកចំនួនដងនៃការប្រមាណវិធី Shift Bits(SHL, SHR)

1.1.4. Data Register(DX)

DX ប្រភេទគឺជា Register ទីបួនដែលផ្សំពី DL និង DH។ ដោយ DH = 1 Byte, DL = 1 Bytes និង DX គឺមានទំហំ 2 Bytes។ មុខងារសំខាន់របស់ DX គឺអាចផ្ទុកទិន្នន័យនិងលទ្ធផលបានដូច AX ហើយចំពោះប្រមាណវិធីគុណលេខ និង ចែកនោះជាប់ទាក់ទងលេខដែលមានទំហំធំជាង Register នោះត្រូវប្រើវា។ DL អាចផ្ទុកចំនួនវង្វិលជុំ ដូចប្រើ CX។



DX = 16 bits , DH = 8 bits and DL = 8

1.2. Flags Register

វាគឺជា Register ដែលមានទំហំ 2 bytes និងប្រើវាសម្រាប់ផ្ទុក Flags ទាំងអស់ដែលបានមកប្រមាណវិធីមួយ ឬ កើតចេញពីការប្រតិបត្តិកម្មវិធីមួយ។ ស៊ីភីយូ 8086 មាន Flags ទាំងអស់ចំនួនប្រាំបួនដូចដែលបានបង្ហាញនៅក្នុងរូបខាងក្រោម៖

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R	R	R	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

ការប្រើ Flags ជាការសំខាន់ណាស់ក្នុងប្រមាណវិធីលំក្ខខណ្ឌ និង ប្រមាណវិធីវង្វិលជុំ។ គ្រប់ Jump instruction ត្រូវបានប្រើជាមួយ Flags ដើម្បីឆ្ពោះទៅកាន់ Target Address នៃទិន្នន័យនៅក្នុងភាសា Assembly។ ជាពិសេសដើម្បីប្រើ Flags ក្នុងការសរសេរកម្មវិធី និងត្រូវចេះកំណត់ Flags ទាំងនេះឱ្យបានច្បាស់លាស់ដោយមើលការធ្វើប្រមាណវិធីគំរូជាបន្តបន្ទាប់ដូចខាងក្រោម។

- U= Unfine
- R= Reserve
- DF = Direction Flag
- DF=Trace Flag
- TF=Trace Flag
- SF = Sign Flag
- AF = Auxilary Flag
- CF = Carry Flag

1.2.1. ការកំណត់ Carry Flag

- CF=1 នៅពេលធ្វើប្រមាណវិធីមួយកើតមានត្រាទុកត្រង់ខ្ទង់ប៊ីតទី៧ចំពោះប្រមាណវិធី ៨ ប៊ីត ខ្ទង់ប៊ីតទី15 ចំពោះប្រមាណវិធី16 ប៊ីត និងខ្ទង់ប៊ីតទី31 ចំពោះប្រមាណវិធី 32 ប៊ីត។
- CF=0 នៅពេលធ្វើប្រមាណវិធីមួយគ្មានត្រាទុកត្រង់ខ្ទង់ប៊ីតទី7 ចំពោះប្រមាណវិធី8 ប៊ីត, ខ្ទង់ប៊ីតទី15 ចំពោះប្រមាណវិធី១៦ប៊ីត, និងខ្ទង់ប៊ីតទី31 ចំពោះប្រមាណវិធី៣២ប៊ីត។

1.2.2. Auxilary Carry Flag (AF)

- AF = 1 នៅពេលធ្វើប្រមាណវិធីមួយកើតមានត្រាទុកត្រង់ខ្ទង់ប៊ីតទី3 ចំពោះប្រមាណវិធី 8 ប៊ីត។
- AF = 0 នៅពេលធ្វើប្រមាណវិធីមួយគ្មានត្រាទុកត្រង់ខ្ទង់ Bits ទី3 ចំពោះប្រមាណវិធី 8 bits។

1.2.3. Parity Flag (PF)

- បន្ទាប់ពីធ្វើប្រមាណវិធីមួយ PF មានតម្លៃស្មើលេខ0 ឬ ១ពិនិត្យទៅលើលេខលទ្ធផលនៃប្រមាណវិធីត្រង់ Lower byte ជាចំនួនគូដែលបានមកពីការរាប់លេខ1 នោះ PF=1។ ឧទាហរណ៍លេខគោលពីរ 1101 0001 មានលេខ១ចំនួន៤ ជាចំនួនគូ។
- បន្ទាប់ពីធ្វើប្រមាណវិធីមួយ PF ពិនិត្យទៅលើលេខលទ្ធផលត្រង់ Lower byte ជាចំនួនសេសដែលបានមកពីការរាប់លេខ1 នោះ PF មិនកំណត់។ ឧទាហរណ៍លេខគោលពីរ 1101 0011 មានលេខ១ចំនួន5 ជាចំនួនសេស(Odd) នោះ PF=0។

1.2.4. Zero Flag(ZF)

- ZF = 1 នៅពេលលេខលទ្ធផលនៃប្រមាណវិធីនព្វន្ធ ឬតក្កវិទ្យាមានតម្លៃស្មើ 0
- ZF = 0 នៅពេលលេខលទ្ធផលនៃប្រមាណវិធីនព្វន្ធមានតម្លៃខុសពី 0។

1.2.5. Sign Flag(SF)

- SF=1 នៅពេលលេខលទ្ធផលនៃប្រមាណវិធីនព្វន្ធ ឬតក្កវិទ្យាមានលេខត្រង់ MSB ស្មើនឹង 1។ នៅពេល SF=1, បញ្ជាក់លទ្ធផលនោះជាចំនួនអវិជ្ជមាន។
- SF=0 នៅពេលលទ្ធផលនៃប្រមាណវិធីនព្វន្ធ ឬតក្កវិទ្យាមានលេខត្រង់ MSB ស្មើនឹង 0។ នៅពេល SF=0, បញ្ជាក់លទ្ធផលនោះជាចំនួនវិជ្ជមាន។

☞ **ឧទាហរណ៍ 1**: រក Flags តាមប្រមាណវិធី 39H + 2FH:

		ចម្លើយ	
	39H	00111001	57
ADD	2FH	00101111	47
	68H	01101000	104

CF = 0 ដោយគ្មានត្រាទុកត្រង់ខ្ទង់ B7 (ខ្ទង់ Bit7 គឺរាប់ពីខាងស្តាំទៅខាងឆ្វេង)

PF = 0 ដោយការរាប់លេខ1 ក្នុងលទ្ធផលជាចំនួនសេស(ដោយមានលេខ១ ចំនួន៣)

AF = 1 ដោយមានត្រាទុកត្រង់ខ្ទង់ B3

ZF = 0 ដោយលទ្ធផលនៃប្រមាណវិធីមិនស្មើ0(មានតែករណីមួយគត់គឺលទ្ធផលនៃប្រមាណវិធីស្មើសូន្យទាំងអស់ 00000000 ទើប ZF=1 ក្រៅពីនឹងគឺ ZF=1)

SF = 0 ដោយលេខត្រង់ MSB=0 ឬ ខ្ទង់ Bits ទី7 ស្មើ0

☞ **ឧទាហរណ៍2:** រក Flags តាមប្រមាណវិធី 9CH + 64H ខាងក្រោម៖

<u>ចម្លើយ</u>

	9CH	10011100	156
ADD	64H	01100100	100
	00H	00000000	256

CF = 1 ពេលធ្វើប្រមាណវិធីដោយមានត្រាទុក ត្រង់ខ្ទង់ B7

PF = 1 ដោយការរាប់លេខ 1′s ក្នុងលទ្ធផលជាចំនួនគូ

AF = 1 ពេលធ្វើប្រមាណវិធីដោយគ្មានត្រាទុកត្រង់ខ្ទង់ B3

ZF = 1 ដោយលទ្ធផលនៃប្រមាណវិធីស្មើ0

SF = 0 ដោយលេខនៅត្រង់ MSB = 0

☞**ឧទាហរណ៍ 3:** រក Flags តាមផ្នែកនៃកម្មវិធីដូចខាងក្រោម៖

MOV AX , 34F5H

MOV DX , 95EBH

ADD AX , DX

<u>ចម្លើយ</u>

	34F5H	00110100	11110101	13557
ADD	95EBH	10010101	11101011	38379
	CAF0H	11001010	11100000	51936

CF = 0 ពេលធ្វើប្រមាណវិធីដោយគ្មានត្រាទុកត្រង់ខ្ទង់ B7

PF = 0, ដោយការរាប់លេខ 1′s ក្នុងលទ្ធផល ជាចំនួនសេស

AF = 1 ធ្វើប្រមាណវិធីគ្មានត្រាទុកត្រង់ខ្ទង់ B3 ; ចំពោះ ZF = 0 ដោយលទ្ធផលនៃប្រមាណវិធីមិនស្មើ0 និង SF = 1 ដោយលេខត្រង់ MSB = 1 ; OF = 0 ខ្ទង់ប៊ីទី១៤ និង១៥ គ្មានត្រាទុក។ ក្នុងករណីកំណត់ OF មើលត្រាទុកពីរខ្ទង់ បើមានត្រាទុកទាំងពីរ និងគ្មានត្រាទុក ទាំងពីរនោះ OF=0 ផ្ទុយពីនឹងមានន័យថាមានត្រាទុកមួយ និងអត់មានមួយ OF=0។

☞ **ឧទាហរណ៍ 4:** រក Flags តាមផ្នែកនៃកម្មវិធីខាងក្រោម៖

MOV BX , AAAAH ; BX ផ្ទុកតម្លៃ AAAAH ឬមានន័យថា Assign តម្លៃចូល Bx

MOV DX , 5556H ; ផ្ទេរតម្លៃ 5556H ចូល Dx ឬ Assign តម្លៃចូល Dx

ADD BX , DX ; BX = BX + DX

<u>ចម្លើយ</u>

	AAAAH	10101010	10101010	49690
ADD	5556H	01010101	01010110	21846
	0000H	00000000	00000000	71536

CF = 1 ពេលធ្វើប្រមាណវិធីដោយមានត្រាទុកត្រង់ខ្ទង់ B7

PF = 1 ការរាប់លេខ 1′s ក្នុងលទ្ធផលជាចំនួនសេស

AF = 1 ដោយគ្មានត្រាទុក ត្រង់ខ្ទង់ B3; ZF = 1 ដោយលទ្ធផលនៃប្រមាណវិធីស្មើ0

SF = 0 ដោយលេខត្រង់ MSB = 0 ; OF = 0 មើលការរៀបរាប់នៅឧទាហរណ៍ទី៣

- Trap Flag (SF)

-TF = 1 ពេលអ្នកសរសេរកម្មវិធី ប្រើបញ្ជា T ក្នុងកម្មវិធី Debug Pro ឬ Assembly

-TF = 0 ពេលអ្នកសរសេរកម្មវិធី មិនប្រើ T Command ឬ Trance Command។

🕮 **ចំណាំ:** T ឬ Trace Command ជាបញ្ជាឱ្យកម្មវិធី Debug ឬ C ប្រតិបត្តិម្តងមួយ Statement។

- Interrupt Flag (IF)

IF គឺប្រើជាមួយនឹង Interrupt Instruction។ ពេល IF = 0ស៊ីភីយូទទួលស្គាល់សំណើរ Interrupt ពីគ្រឿងកុំព្យូទ័រ និង IF =1 ស៊ីភីយូមិនទទួលស្គាល់សំណើរ Interrupt ពីគ្រឿងកុំព្យូទ័រ (peripherals)។

- Direction Flag(DF)
  - DF=1 នៅពេលអ្នកសរសេរកម្មវិធីប្រើពាក្យបញ្ជា STD នៅក្នុងកម្មវិធី។
  - DF=0 នៅពេលអ្នកសរសេរកម្មវិធីប្រើពាក្យបញ្ជា CLD នៅក្នុងកម្មវិធី។

🕮 **ចំណាំ:**

-STD ជា Instruction កំណត់ឱ្យ DF = 1 ដើម្បីឱ្យ CPU អាចយកតួអក្សរពីស្តាំទៅខាងឆ្វេង ឬ ពីខាងក្រោមទៅខាងលើ។

-CLD ជា Instruction កំណត់ឱ្យ DF = 0 ដើម្បីឱ្យ CPU អាចយកតួអក្សរពីខាងឆ្វេងទៅខាងស្តាំ ឬ ពីខាងលើទៅខាងក្រោម។

- Overflow Flag (OF)

-OF = 1 ពេលធ្វើប្រមាណវិធីមួយមានត្រាទុកនៅត្រង់ខ្ទង់ប៊ីតB6, ប៉ុន្តែគ្មានត្រាទុក ត្រង់ខ្ទង់ប៊ីត B7 ឬ គ្មានខ្ទង់ទី6។ ប៉ុន្តែមានត្រាទុកត្រង់ប៊ីត B7 ចំពោះប្រមាណវិធី8 bits។ ចំពោះប្រមាណវិធី 16 bit គឺមើលត្រាទុក ត្រង់ប៊ីត ទី14 និង bit ទី15។

-0F = 0 ពេលប្រមាណវិធីមួយមានត្រាទុក ខ្ទង់ bit 6 និង ខ្ទង់bit 7 ឬគ្មានត្រាទុក ត្រង់ខ្ទង់ bit 6 និងខ្ទង់ bit 7។ ចំពោះប្រមាណវិធី 16 bit គឺមើលត្រាទុក ត្រង់ bit ទី 14 និង bit ទី15

☺**ឧទាហរណ៍:** បង្ហាញពី Flags បន្ទាប់ពីប្រតិបត្តិប្រមាណវិធីបូក Hexa ខាងក្រោម៖

Hexa	Binary	Dec
28H	00101000	40
+ 2FH	+ 00101111	47
57H	01010111	87

1.3. INDEX Register

1.3.1. Destination Index Register (DI)

DI ជា Register មួយដែលមានទំហំ 2 bytes និងមិនអាចចែកទីតាំងដូច AX, BX,DX និង CX ទេ។ ពោលគឺវាគ្មាន Register តូចៗទេ និង មុខងាររបស់ DI គឺ៖

- ការផ្ទុកតម្លៃទិន្នន័យ និង លទ្ធផលនៃការធ្វើប្រមាណវិធីតក្ក និងលេខនព្វន្តមួយ។

ប្រើវាសម្រាប់ឆ្ពោះទៅកាន់លេខអាសយដ្ឋាននៃទិន្នន័យ ដែលត្រូវដំណើរការដោយវាប្រើដូច BX register ដែរ។

- ប្រើជា Pointer Register សម្រាប់ឆ្ពោះទៅកាន់លេខអាសយដ្ឋានរបស់ទិន្នន័យ ជាតួអក្សរជាប្រភេទ String នៅក្នុងផ្នែ Extra Data Segment។

DI [ ] DI = 2 bytes = 1 word

1.3.2. Source Index Register (SI)

SI ជា Register មួយដែលមានទំហំ 2 bytes មានមុខងារសម្រាប់៖

- ផ្ទុកនូវតម្លៃទិន្នន័យ និង លទ្ធផលនៃការធ្វើប្រមាណវិធីមួយ
- ឆ្ពោះទៅកាន់ Affest Address ប្រើដូច DI ដែរ។
- ឆ្ពោះទៅកាន់អាសយដ្ឋាន(Affest Address)នៃទិន្នន័យជាប្រភេទតួអក្សរ String នៅក្នុងផ្នែ Data Segment

1.4. Instruction Pointer

IP គឺជា Register សំខាន់បំផុតនៅក្នុងស៊ីភីយូ និងមានទំហំ ១៦ ប៊ីត។ វាត្រូវបានប្រើសម្រាប់ផ្ទុកលេខអាសយដ្ឋាននៃពាក្យបញ្ជាដែលត្រូវប្រតិបត្តិទីមួយ ទីពីរ និងបន្តបន្ទាប់នៅពេលស៊ីភីយូដំណើរការកម្មវិធីដែលមានក្នុងអង្គចងចាំ(RAM)។ វាក៏ប្រើជាមួយនឹង CS register ដើម្បីរក Physical Address ដែលត្រូវប្រតិបត្តិ។

1.5. CS Register

CS Register ប្រើសម្រាប់ឆ្ពោះទៅកាន់អាសយដ្ឋានចាប់ផ្តើម(Start Address)នៃផ្នែកូដ Segment ដែលមានទំហំ 64 KB ដោយប្រើវាផ្ទុកនូវលេខអាសយដ្ឋាននេះ។ ដោយ CS Register មានទំហំ 16 ប៊ីត ដូច្នេះវាអាចផ្ទុកអាសយដ្ឋានចាប់ផ្តើមបានតែ 16 ប៊ីតដែរ។ ដូច្នេះ CS Register អាចផ្ទុក Phyiscal Address បាននោះត្រូវលុប 0H ខាងចុងមួយរបស់អាសយដ្ឋានចាប់ផ្តើមនៃ ៨០៨៦ ដោយដែនកំណត់ពី 00000H ដល់ FFFFFh(២០ប៊ីត)។ នៅពេលស៊ីភីយូប្រតិបត្តិពាក្យមួយដើម្បីធ្វើការហៅពាក្យបញ្ជាពី RAM ចូលក្នុងខ្លួនវានោះត្រូវប្រើ Physical Addess។ ដោយធ្វើប្រមាណវិធីដោយយកតម្លៃនៅក្នុង CS Resgister គុណនឹងលេខក្នុងគោដប់ប្រាំមួយគឺលេខ10H រួចបូកតម្លៃ Offset Address ក្នុង IP បង្កើតបានជាអាសយដ្ឋាននៃពាក្យបញ្ជាដែលមានរូបមន្ត៖

PA = CS * 10H + IP	IP = Start addrees of code

ឧទាហរណ៍: ពេលសរសេរ Mov al, 50h ដោយសន្មត់ CS=0700H និង IP = 0100H

1. គណនា Physical Address នៃពាក្យបញ្ជា(Mov)ដែលប្រតិបត្តិ
2. គណនា Physical Address ទាបបំផុតនៃផ្នែ Code Segment
3. គណនា Physical Address ខ្ពស់បំផុត នៃផ្នែ Code Segment
4. ចូរបង្ហាញ Logical Address នៃពាក្យបញ្ជា(Instruction)
5. ចូរបង្ហាញ Offset Address នៃពាក្យបញ្ជា

ចម្លើយ

1. PA = CS*10H +IP = 0700H+ 0100H , ដូច្នេះ PA=07100H
2. គណនា PA ទាបបំផុតនៃផ្នែ CS= 0700H + 0000H = 07000H
3. គណនា PA ខ្ពស់បំផុតនៃផ្នែ CS = 0700H + FFFFH=16FFFh
4. LA = 0700: 0100H
5. Offfest address នៃពាក្យបញ្ជា Mov គឺ IP = 0100H

1.6. DS Register

DS Register មានតួនាទីឆ្ពោះទៅកាន់លេខអាសយដ្ឋានចាប់ផ្តើមរបស់ផ្នែ Data Segment រួចវាផ្ទុកលេខអាសយដ្ឋាននេះ។ តម្លៃអាសយដ្ឋាននៅក្នុង DS Register ធ្វើប្រមាណវិធីគុណនឹងលេខគោលដប់ប្រាំមួយគឺ 10H រួចបូកនឹងតម្លៃ OA(Offset Address) នៅក្នុង BX, SI, DI ឬលេខ OA ដោយផ្ទាល់ បង្កើតបានជា PA នៃទិន្នន័យដែលត្រូវដំណើរការ។ ខាងក្រោមនេះគឺរូបមន្តដើម្បីរក Physical Address របស់ទិន្នន័យ។

1.7. ES Register

ES Register មានតួនាទីឆ្ពោះទៅកាន់លេខអាសយដ្ឋានចាប់ផ្តើមរបស់ផ្នែ Extra Data Segment ។ តម្លៃអាសយដ្ឋាននៅក្នុង ES Register ធ្វើប្រមាណវិធីគុណនឹងលេខគោលដប់ប្រាំមួយគឺ10H ហើយបូកនឹងតម្លៃ OA នៅក្នុង DI បង្កើតបានជា PA នៃទិន្នន័យដែលត្រូវដំណើរការក្នុងផ្នែ Extra Data Segment។ នេះគឺរូបមន្តដើម្បីរក Physical Address របស់ Data ក្នុងផ្នែ Extra Data Segment

$$PA=ES * 10H + DI$$

1.8. SS Register

គឺជា Register មាននាទីឆ្ពោះទៅកាន់លេខអាសយដ្ឋាន ចាប់ផ្តើមរបស់ផ្នែ Stack Segment រួចវាផ្ទុកលេខអាសយដ្ឋាននេះ។ តម្លៃលេខអាសយដ្ឋាននៅក្នុង SS Register ធ្វើប្រមាណវិធីគុណនឹងលេខគោលដប់ប្រាំមួយគឺ 10H បូកនឹងតម្លៃ OA នៅក្នុង SP បង្កើតបានជា PA នៃទិន្នន័យដែលត្រូវដំណើរការក្នុងផ្នែ Stack Segment។

$$PA = SS * 10 + SP (BP)$$

2. **សំនុំពាក្យបញ្ជា 8086**(Instruction Set of 8086)

ពាក្យបញ្ជារបស់ស៊ីភីយូ 8086 គឺជាពាក្យដែលមានទំរង់ជាប្រព័ន្ធគោលពីរ។ នៅពេលពាក្យបញ្ជានេះត្រូវបានបកប្រែ នោះវាប្រាប់ទៅស៊ីភីយូឱ្យធ្វើកិច្ចការជាក់លាក់ណាមួយ។ ពាក្យបញ្ជាជាពាក្យកាត់ដែលមានពីរ ឬបីអក្សរ ឬហៅថាជា Mnemonic។ វាគឺជាពាក្យគន្លឹះប្រើសម្រាប់ការដំណើរការអ្វីមួយ និងត្រូវបានបញ្ចូលទៅក្នុងកូដ file ធ្វើប្រមាណវិធីនៃ assembler របស់ពាក្យបញ្ជានៃកម្មវិធីមួយ។

ទំរង់ទូទៅពាក្យបញ្ជារបស់ស៊ីភីយូចែកចេញជាពីរផ្នែកដូចបង្ហាញតាមករណីដូចខាងក្រោម៖

Opcode Operand ឬ **Opcode Address**

- ✧ **Opcode** ប្រាប់ទៅស៊ីយូ(CU) ឱ្យដំណើរការទិន្នន័យ និងធ្វើប្រមាណវិធីផ្សេងៗ ដូចជាប្រមាណវិធីនព្វន្ធ, តក្កវិទ្យា ឬធ្វើការបំលាស់ទីទិន្នន័យពីកន្លែងមួយទៅកន្លែងមួយផ្សេងទៀត។
- ✧ **Operand** គឺជាទិន្នន័យដែលត្រូវប្រតិបត្តិដោយ Opcode។
- ✧ **Address** ប្រាប់ទៅស៊ីភីយូឱ្យរកអាសយដ្ឋាន នៃ Operand ដែលត្រូវការយកមកប្រតិបត្តិ។

ពាក្យបញ្ជារបស់ស៊ីភីយូចែកចេញជាផ្នែកៗដូចខាងក្រោម៖

◆ Arithmetic And Logic	◆ Comparison
◆ Data Transfer	◆ Flag Operation
◆ Stack Operation	◆ Input / Output
◆ ASCII – BCD conversion	◆ Looping
◆ Bit Shift	◆ Signed Numbers , String

2.1. ប្រមាណវិធីនព្វន្ធ និងតក្កវិទ្យា

2.1.1. MOV (MOVE) Instruction

MOV គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់ចំលងទិន្នន័យពី Memory(RAM) ទៅផ្ទុកនៅក្នុង Register និងបញ្ច្រាសមកវិញពី Register ទៅផ្ទុកក្នុង Memory។ វាអាចធ្វើការចំលងទិន្នន័យពី Register មួយទៅ Register មួយទៅដែលមានទីតាំងនៅក្នុងស៊ីភីយូ។

✍ **Syntax : MOV dest , source ;** យកតម្លៃពីទីតាំង Source ទៅ Destination

- ✧ Source អាចប្រើ: ឈ្មោះ Register និង Memory(ប្រើអថេរ x, y,...z), Register និង Register, ឈ្មោះ Register និងតម្លៃផ្ទាល់។ តែមិនអាចប្រើ Memory និង Memory បានទេ។
- ✧ dest អាចប្រើ: ឈ្មោះ Register និង Memory, Register និង Register, ឈ្មោះ Register និងលេខអាសយដ្ឋាន។

ចំណាំ: នៅពេលប្រើពាក្យបញ្ជា MOV ត្រូវកំណត់ទីតាំង dest និង source ត្រូវមានទំហំដូចគ្នា, បើ dest = 8 bits, នោះ source = 8 bits និង បើ dest =16 bits នោះ source= 16 bit ដែរ។

☺**ឧទាហរណ៍**: ប្រើ MOV Instruction ជាមួយ Register និងប្រាប់ពីមុខងាររបស់ Register។

1. **MOV AX , 1234H** ; ចំលងតម្លៃ 1234H ចូលក្នុង AX register ឬនិយាយខ្លី ***កំណត់តម្លៃ ឱ្យ AX = 1234H***។ នៅក្នុងកម្មវិធី X86Asm IDE យើងត្រូវប្រើអនុគមន៍ PrintHex Ax សម្រាប់ធ្វើការបង្ហាញតម្លៃពីក្នុង AX ទៅអេក្រង់បញ្ចេញតម្លៃ។
2. MOV CL , AL ; ចំលងតម្លៃពីក្នុង AL ចូលកាន់ CL register ; ឬកំណត់CL =AL
3. MOV CX , DX ; ចំលងតម្លៃពីក្នុង DX ទៅដាក់ក្នុង CX register

ចំណាំ: - ការប្រើសញ្ញា ; នៅខាងចុងសរសេរពាក្យបញ្ជាវាជា comment ដូចដែលយើង ប្រើ សញ្ញា // នៅក្នុងភាសា C។

- នៅក្នុងភាសា Assembly យើងត្រូវយល់ដឹងពីការប្រើបញ្ជានេះសំខាន់មុខគេ

2.1.2. ADD (Addition) Instruction

ADD គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់ប្រាប់ទៅ **ALU** ឱ្យធ្វើប្រមាណវិធីបូកពីចំនួន ដោយទិន្នន័យទី១ត្រូវហៅចេញដោយស្វ័យប្រវត្តិពី Registser(AX,BX,CX,DX,SI,SI) និង ទិន្នន័យទី២ ហៅចេញពី Register/Memory(ប្រើអថេរ x, y, z,...) និងលទ្ធផលក្រោយពី ALU គណនារួចផ្ទុកនៅក្នុង Register ឬ Memory ដែលជាទីតាំង Destination។

✍**Syntax : ADD dest , sou** ; មានន័យថា sou = sou + dest

ឧទាហរណ៍: ការប្រើពាក្យបញ្ជា ADD ជាមួយនឹងការប្រើ Register ឬ Memory សូមមើលការសរសេរពាក្យបញ្ជាគំរូដូចខាងក្រោមនេះ៖

1. **ADD AL , CL** , គឺជាការបូកលេខពីរចំនួនដោយយកតម្លៃទី១ពីក្នុង AL បូកនឹងតម្លៃទី២ក្នុង BL និងលទ្ធផលផ្ទុកនៅក្នុង AL។ ឬនិយាយខ្លី **AL = AL + CL**
2. **ADD AX ,1234h** ; AX = AX + 1234h ដោយ AX ជាឈ្មោះ Register ទំហំ១៦ប៊ីត
3. **ADD DX , x** ; DX =DX +x : ដោយ ADD ជាឈ្មោះពាក្យបញ្ជា ,DX ជាឈ្មោះ Register និង x ជាអថេរ។ នៅក្នុងភាសានេះប្រកាសអថេរ ១៦ប៊ីត x dw 1234h
4. **ADD sum , DI** ; មានន័យថា sum = sum + DI

សំគាល់: ការសរសេរឈ្មោះពាក្យបញ្ជាអាចប្រើអក្សរតូច, ធំទាំងអស់ និងតូចធំក៏បាន។ add sum , DI ឬ Add sum, di។ ចំពោះការប្រើប្រាស់ឈ្មោះ Register ក៏ដូចគ្នាដែរ។ បើនៅទីតាំង Dest មានទំហំ ៨ ប៊ីត និង source ក៏ត្រូវមានទំហំ ៨ ប៊ីតដែរ។

2.1.3. Sub (Subtraction) Instruction

SUB គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់ប្រាប់ទៅ ALU ឱ្យធ្វើប្រមាណវិធីពីដកលេខពីរចំនួន។ ការដកលេខដោយយក Data1 ហៅចេញពី Register និង Data2 ហៅចេញពី Register/Memory លទ្ធផលក្រោយពី ALU គណនារួចផ្ទុកនៅក្នុង Register ឬ Memory ដែលជាទីតាំង Destination។

✍**Syntax : Sub dest , sou** ; sou = sou – dest ;

ការប្រើ **Sub** Instruction ជាមួយនឹងការប្រើ Register និង Memory សូមមើលការរៀបរាប់ដូចខាងក្រោម៖

- Sub AL , CL ; យកតម្លៃក្នុង AL ដកនឹង CL និងផ្ទុកលទ្ធផលក្នុង AL
- Sub AX ,1234h ; យកតម្លៃក្នុង AX ដកនឹង 1234h និងផ្ទុកលទ្ធផលក្នុង AX
- Sub r , DI ; r = r- DI

2.1.4. DEC (Decrement) Instruction

DEC ជាពាក្យបញ្ជាមួយសម្រាប់ធ្វើការបន្ថែមតម្លៃលេខនៅក្នុង Register ឬ Memory ម្តងមួយដូចដែលយើង ប្រើ i=i+1 នៅក្នុងភាសា C ដែរ។

✍**Syntax : DEC Register**

MOV CX , 5 ; CX =5

DEC CX ; CX = CX – 1 = 4

DEC CX ; CX = CX – 1 = 3 យើងអាចសរសេរ SUB CX , 2 ស្មើនឹងសរសេរការសរសេរពាក្យបញ្ជា DEC ចំនួនពីរដង។

2.1.5. INC (Increment) Instruction

INC ជាពាក្យបញ្ជាសម្រាប់ធ្វើការបន្ថែមលេខនៅក្នុង Register ម្តងមួយ។

⊗ **Syntax :** INC Register

MOV BX , 1230H ; កំណត់ BX = 1230H ឬយកតម្លៃ1230h ចូលក្នុង Bx

INC BX ; BX = BX + 1 = 1231H ឬសរសេរ Add bx, 1

INC BX ; BX = BX + 1 ឬយើងសរសេរ ADD BX , 1

2.1.7. MUL Instruction

Mul ជាបញ្ជាមួយប្រើសម្រាប់ធ្វើប្រមាណវិធីការគុណលេខនៅក្នុងស៊ីភីយូ។ ដើម្បីធ្វើគុណលេខចែកជាបួនករណីដូចតារាងគ្រឹះដូចខាងក្រោមចំពោះស្ថាបត្យកម្មប្រើ ៣២ ប៊ីត។

Mul dest ; dest = dest * source

Multiplication	Operand 1	Operand 2	Result
Byte x Byte	AL	Register or Memory	AX
Word x Word	AX	Register or Memory	DX:AX
Byte x Word	AL=byte AH = 0	Register or Memory	DX:AX
DwordxDword	Eax	Register or Memory	EAX:EBX

ឧទាហរណ៍ទី១: ចូរមើលកូដគំរូខាងក្រោមដើម្បីការគុណលេខ x(byte) = 10 គុណនឹង y(byte)=20 និងលទ្ធផលផ្ទុកនៅក្នុង Ax Register។

```
.Data
        x    Db    10   ; Declarses x( byte )=10 ;in C : int char x=10
        y    Db    20   ; Declarses y( byte )=20;in C : int char y=20
.Code
start:
        mov al, x  ; al =x មានន័យថា al ផ្ទុកតម្លៃ x គិតជាតម្លៃគឺ al = 10
        mul y ; ax = al x y =x  * y ការគុណលេខ Byte*Byte ជាច់ខាតតម្លៃxផ្ទុកក្នុងal
        PrintDec ax ; បង្ហាញតម្លៃគោលដប់ពីក្នុង Ax register ទៅ Debug Window
Ret      ; Return to operation System
end start ; បញ្ចប់កម្មវិធី ដោយមាន start: ជាចំនុចចាប់ផ្តើមកម្មវិធី
```

ចំណាំ: ការប្រកាសប្រភេទអថេរដែលត្រូវផ្ទុកលេខជាចំនួនគត់(Unsigned Number)នោះត្រូវប្រើ db, dw, dd។ តាមឧទាហរណ៍ខាងលើពេលសរសេរ **x db 10** ;មានន័យថាប្រកាសអថេរ x = 10។ យើងអាចដឹងទំហំអថេរ x មើលការប្រើ data type :db មានន័យថា Define byte។ ចំពោះការប្រើ dw(define word) សូមមើលឧទាហរណ៍ខាងក្រោម៖

ឧទាហរណ៍ទី២: គុណលេខ x(word) =1234h គុណនឹង y(word)=5678h និងលទ្ធផលផ្ទុកនៅក្នុង DX:AX

```
.Data
   x   Dw      1000 ; declares x  as word and x=1000
   y   Dw      2000 ; declares x as word and x=2000
.Code
start:
    mov ax, x          ; move value x to ax register
    mul y              ; dx:ax = ax x y =
    PrintDec ax        ;displays the value in ax in decimal
    PrintDec dx ;displays the value in dx in decimal
Ret
end start
```

ឧទាហរណ៍ទី៣: កូដខាងក្រោមសម្រាប់ធ្វើប្រមាណវិធីគុណលេខប្រើទំហំ ៣២ប៊ីតដោយ x(dword) =10 គុណនឹង y(dword)=1000 :

```
.Data
   x  Dd  1000          ; declares x double word and x=1000
   y  Dd  2000          ; declares x double word and x=2000
.Code
start:
    mov eax, x  ; move value x to eax register
    mul y               ; edx : eax = eax x y
    PrintDec eax         ;displays the value in eax in decimal
    PrintDec edx        ;displays the value in edx in decimal
Ret                     ;Return to OS
end start               ;end program
```

2.1.7. Div instruction

Div ជាពាក្យបញ្ជាប្រើសម្រាប់ធ្វើប្រមាណវិធីការចែកលេខ និងការធ្វើប្រមាណវិធីនេះចែកជាបួនករណីដូចតារាងខាងក្រោម៖ **ក្នុងភាសា C ប្រើសញ្ញា / ។ ឧទាហរណ៍ result = x / y =Qutient , result = x%y =remainder។**

Div Register or Memory ; Ex :25 / 2

Division	Dividend=25	Divisor=2	Quotient	Remaider
Byte / Byte	AL=B, AH=0	Mem or Reg	AL	AH
Word / Word	AX=W, DX=0	Mem or Reg	AX	DX
Word / Byte	AX = Word	Mem or Reg	AL	AH
DWord / Word	DX:AX=DWord	Mem or Reg	AX	DX
Dword / Dword	Eax	Mem or Reg	Eax	Edx

```
.Data
        num     Db      92      ;declares num double word and num=92
        two     Db      2       ;declares two double word and two=2
.Code
start:
        mov al, x       ; move the value x to al
        div y           ;  al = quotient and AH =remainder
        PrinDec al      ; displays the value in eax in decimal
        PrintDec ah     ; displays the value in edx in decimal
getkey
Ret
end start
```

2.2. ពាក្យបញ្ជាតក្កវិទ្យា

2.2.1. And instruction

AND **គឺ**ជាពាក្យបញ្ជាមួយប្រើសម្រាប់ធ្វើប្រមាណវិធីតក្កវិទ្យាឈ្នាប់នឹងអាស្រ័យលើទិន្នន័យទាំងពីរចំនួន និងលទ្ធផលផ្ទុកនៅទីតាំង Destination ដោយអាចប្រើជា Memory ឬ Register និង Source operand អាចប្រើជា Immediate, Memory, Register។ **ក្នុងភាសា C ប្រើសញ្ញា && .ឧទាហរណ៍ result = x &&y**

Format : AND Dest , Source ; Dest = Dest AND Source

ឧទាហរណ៍ធ្វើប្រមាណវិធី 1001 1101b And 10001001b=1000 1001b និងកូដ:

```
.data
        x       db      10011101b ; ប្រកាសអថេរ x ផ្ទុកលេខ10011101b
        y       db      10001001b ;ប្រកាសអថេរ y ផ្ទុកលេខ10001001b
        r       db      00000000b
.code
start:
        mov ax, x       ; ax = x , Ax ផ្ទុកតម្លៃ x
        and ax, y       ; ax and y ធ្វើប្រមាណវិធីតក្កវិទ្យាឈ្នាប់នឹង
        PrintBin ax     ; បង្ហាញលទ្ធផលនៅលើ Debug Windows
Ret                     ; Return to Os
end start               ; end the program
```

2.2.2. Or instruction

OR ជាពាក្យបញ្ជាប្រើសម្រាប់ធ្វើប្រមាណវិធីតក្កវិទ្យាឈ្នាប់ឬ អាស្រ័យលើតម្លៃលេខទាំងពីរ និងលទ្ធផលផ្ទុកនៅទីតាំង Dest ដោយទីតាំងនេះអាចប្រើជា Memory ឬ Register និងតម្លៃលេខ Source អាចប្រើជា Immediate, Memory, Register។ **ក្នុងភាសា C ប្រើសញ្ញា || ។ ឧទាហរណ៍ result = x || y ។**

កម្មវិធីខាងក្រោមបញ្ជាក់ពីការប្រើពាក្យបញ្ជា Or ដោយកម្មវិធីដូចបញ្ជា And ដែរ

```
.data
x       db      10011101b ; ប្រកាសអថេរ x ផ្ទុកលេខ10011101b
y       db      10001001b ;ប្រកាសអថេរ  y ផ្ទុកលេខ10001001b
r       db      00000000b
.code
start:
        mov ax, x       ; ax = x , មានន័យថាប្រើ Ax ផ្ទុកតម្លៃ x
        or ax, y        ; ax or y ធ្វើប្រមាណវិធីតក្កវិទ្យាឈ្នាប់ឬ( Logical or )
        PrintBin ax     ; បង្ហាញលទ្ធផលនៅលើអេក្រង់ Debug Windows
Ret                     ; Return to Os
end start               ; end the program
```

Format : **Or** Dest , Source ; Dest = Dest Or Source

2.2.3. Not instruction ; ក្នុងភាសា C ប្រើសញ្ញា !

NOT ជាបញ្ជាប្រើសម្រាប់ធ្វើប្រមាណវិធីតក្កវិទ្យាឈ្នាប់មិន។ ការធ្វើប្រមាណវិធីរបស់វាគឺប្តូរពីលេខ ១ ទៅ ០ និងពីលេខ ០ មកលេខ ១ វិញ។

Format : **NOt** dest

Data 1001 0001b not data 0110 1110

```
.data
        x  db 10101100b ; ប្រកាសអថេរ x ផ្ទុកលេខ10101100
.code
Start:  Mov al, x ; al ផ្ទុកតម្លៃ x ដោយ al =x
        Not al ; ធ្វើប្រមាណវិធីឈ្នាប់មិន
        PrintHex al ; បង្ហាញតម្លៃពីក្នុង al ទៅលើអេក្រង់លទ្ធផល
Ret
Ent start
```

2.3. Data Transfer

2.3.1. CLD (Clear Direction Flags)

CLD ជាពាក្យបញ្ជាប្រើសម្រាប់ឱ្យ DF = 0។ ក្នុងការប្រើ String Instruction បើ DF = 0 , Pointer Register បន្ថែម Address រាល់ការប្រតិបត្តិ Instruction និមួយៗ។ បើ DF = 1 , Pointer Register បន្ថយ Address រាល់ការប្រតិបត្តិ Instruction និមួយៗ។

Syntax : CLD និងការប្រើប្រាស់Flags គឺ DF

2.3.2. STD (Set Direction Flags)

STD ជា Instruction ប្រើឱ្យ DF = 1។ ក្នុងការប្រើ String Instruction បើ DF = 1, នោះ Pointer Register បន្ថយ Address រាល់ការប្រតិបត្តិ Instruction និមួយៗ។ បើ DF = 1, Pointer Register បន្ថែម Address រាល់ការប្រតិបត្តិ Instruction និមួយៗ។

Syntax : STD និងការប្រើប្រាស់Flags គឺ **DF**

2.3.3. LEA (Load Effective Address)

បញ្ជានេះផ្ទុកនូវ Effective Address របស់ Operand ដោយផ្ទាល់ទៅក្នុងទីតាំង Destination ដោយប្រើ 16 bits Register។

Syntax : LEA dest. , sou.

Flags : Unchange

😐ឧទាហរណ៍: វិភាគផ្នែកកម្មវិធីដូចខាងក្រោម៖

```
ORG  100H ; កំណត់លេខ Address ដំបូងនៃកូដដែលសរសេរក្នុង Emu8086
.data ; Keyword នេះសម្រាប់បញ្ចូលទិន្នន័យឱ្យ Assembler
 DATA DB 23H,48H,40H,80H,90H,39H; The First address of DATA is 0102H
.code ;Keyword នេះសម្រាប់បញ្ចូលពាក្យបញ្ជាឱ្យ Assembler
        LEA DI , DATA ; Load effective address of DATA to DI register .
```

```
        LEA SI , DATA + 4 ; Load effective address of DATA +4 to SI register
        MOV AL , [SI] ; move the value at address that SI point to , to AL reg.
ret
```

☺**ឧទាហរណ៍:** ចូរវិភាគផ្នែក និងមើលគំរូនៃកម្មវិធី Assembly ដូចខាងក្រោមដើម្បីស្វែងយល់ពីការប្រើប្រាស់ពាក្យបញ្ជា LED៖

```
org 100h
.MODEL SMALL
.DATA
        DATA1 DW 2000H ; ប្រកាសអថេរ DATA1( 16 bit )= 2000h
        DATA2 DW 3000H; ប្រកាសអថេរ DATA2( 16 bit )= 3000h
.CODE
.STARTUP
        LEA SI, DATA1 ;ផ្ទុក Effect Address ទៅក្នុង SI Register
        MOV DI, OFFSET DATA2 ; move the offset address of DATA2
        MOV BX , [SI] ; ផ្ទេរតម្លៃនៅអាសយដ្ឋានដែល SI ចង្អុលទៅចូលក្នុង Bx register
        MOV CX , [DI]  ; ផ្ទេរតម្លៃនៅអាសយដ្ឋានដែល DI ចង្អុលទៅចូលក្នុង CX register
        MOV [SI] , BX ; ផ្ទេរតម្លៃពីក្នុង BX Register ទៅអាសយដ្ឋានដែល SI ចង្អុលទៅ
        MOV [DI] , CX ; ផ្ទេរតម្លៃពីក្នុង CX Register ទៅអាសយដ្ឋានដែល DI ចង្អុលទៅ
.EXIT
```

2.3.4. LODSB (Load String Byte)

ធ្វើការចំលងមួយបៃពីទីតាំងដើមទៅចូលក្នុង AL ដែលចង្អុលបង្ហាញដោយ DS:SI។ បើសិន DF = 0, នោះ SI នឹងបង្កើនលេខ Address ដើម្បីបន្តទៅកាន់ទីតាំងបន្ទាប់។ បើ DF = 01, នោះ SI នឹងបន្ថយលេខ Address ដើម្បីបន្តទៅកាន់ទីតាំងបន្ទាប់តាមលំដាប់ថយ

Syntax : LODSB

Flags : Unchanged

☺**ឧទាហរណ៍:** គេមាន Sourece = "ABCDEF" ។ ចូរសរសេរកម្មវិធីដើម្បីធ្វើការផ្ទេរទិន្នន័យពីផ្នែទីតាំង Source ទៅកាន់ផ្នែ Destination ដោយប្រើ LODSB instruction។

```
org 100h ; កំណត់លេខអាសយដ្ឋានឱ្យ IP Register
.data ; ប្រើសម្រាប់បញ្ចូលទិន្នន័យ
        a1 db "Hello Rupp" ; ប្រកាសអថេរ a1 ផ្ទុក String byte
.code  ; ប្រើសម្រាប់បញ្ចូលកូដ
        LEA SI, a1 ; បញ្ចូលលេខអាសយដ្ឋានដំបូងនៃអក្សរ H ចូល SI
        MOV CX, 10 ; cx =10 ដែលលេខ ១0 ជាចំនួនវង្វិលជុំ
        MOV AH, 0Eh ; AH = OE ដែលលេខ 0Eh ជាលេខអនុគមន៍បង្ហាញតួអក្សរ
        m: LODSB
            INT 10h ; បង្ហាញតម្លៃទៅក្នុង AL ទៅអេក្រង់បញ្ចេញលទ្ធផល
```

LOOP m

RET

2.3.5. LODSW (Load String Word)

ចំលង String Word ពី ទៅក្នុង AX Register ដែលចង្អុលបង្ហាញដោយ DS:SI។ បើ DF = 0, SI នឹងបង្កើនលេខ Address ដើម្បីបន្តទៅកាន់ទីតាំងបន្ទាប់។ បើ DF = 01, SI នឹងបន្ថយលេខ Address ដើម្បីបន្តទៅកាន់ទីតាំងបន្ទាប់តាមលំដាប់ថយ។

Syntax : LODSW

Flags : Unchanged

😑**ឧទាហរណ៍**: ចូរសរសេរកម្មវិធី Assembly ដើម្បីផ្ទេរតួអក្សរនៅ Data:"Computer Architecter "ទៅទីតាំង Destination។

2.3.6. XCHG (Exchange)

XCHG គឺជាពាក្យបញ្ជាធ្វើការផ្លាស់ប្តូរធាតុនៅក្នុង Registers ពីរ ឬរវាង Register និង Memory វិញ។

Syntax : XCHG dest , sour

😑**ឧទាហរណ៍**: កម្មវិធីខាងក្រោមប្រើភាសា Asm ដើម្បីធ្វើការប្តូរតម្លៃពី AX=5070H ទៅ DX = 1112H

org 100h ; កំណត់លេខអាសយដ្ឋានចាប់ផ្តើមនៃ Instruction ដំបូង

MOV AX, 5070H ; ផ្ទេរតម្លៃ Immedeate:5070H ចូលក្នុង AX register។

MOV BX, 1112H ; ផ្ទេរតម្លៃ Immedeate:5070H ចូលក្នុង AX register។

XCHG AX , BX ; ជាការប្តូរតម្លៃរវាង AX register និង BX Register។

RET

💻 **OUTPUT** : AX = 1112H , BX = 5070H

2.3.7. LDS (Load Data Segment)

ពាក្យបញ្ជា LDS ធ្វើការផ្ទុក Double word memory ចូលទៅក្នុង 16-bits Register ផ្ទុក Word 1 និង DS ផ្ទុក word 2។

Syntax : LDS dest , sour

Flags :

😑**ឧទាហរណ៍**: រកលទ្ធផលតាមកម្មវិធីគំរូដូចខាងក្រោម៖

ORG 100h **; កំណត់លេខអាសយដ្ឋានចាប់ផ្តើមនៃ Instruction ដំបូង**

LDS AX, m ; Ax ផ្ទុកតម្លៃ m ដោយ Ax =1234h

RET ; ត្រលប់ទៅកាន់ Os ដើម្បីធ្វើការបញ្ចប់ការប្រត្តិបត្តិ Instruction

m DW 1234h ; ប្រកាសអថេរ m ផ្ទុកតម្លៃ 1234h

DW 5678h ; ប្រកាសតម្លៃលេខអាសយដ្ឋានចាប់ផ្តើមដាក់ក្នុង DS=5678h

2.3.8. LES , LSS Instruction

ពាក្យបញ្ជា LES ធ្វើការផ្ទុក Double word memory ចូលក្នុង 16-bits Register ផ្ទុក Word 1 និង ES ផ្ទុក word 2។ ចំណែក LSS instruction ផ្ទុក Double word memory ចូលក្នុង 16-bits Register ផ្ទុក Word 1 និង DS ផ្ទុក word 2។

Syntax : LES dest , sou

LSS dest , sou

☺**ឧទាហរណ៍**: រកលទ្ធផលតាមកម្មវិធីគំរូដូចខាងក្រោម៖

ORG 100h **; កំណត់លេខអាសយដ្ឋានចាប់ផ្តើមនៃ Instruction ដំបូង**

LES AX, m ; Ax ផ្ទុកតម្លៃ m , Ax =1234h

RET ; ត្រលប់ទៅកាន់ Os ដើម្បីធ្វើការបញ្ចប់ការប្រត្តិបត្តិ Instruction

m DW 1234h ; ប្រកាសអថេរ m ផ្ទុកតម្លៃ 1234h

DW 5678h ; ប្រកាសតម្លៃលេខអាសយដ្ឋានចាប់ផ្តើមជាក់ក្នុង ES=5678h

☺**ឧទាហរណ៍**: រក Output តាម Program ខាងក្រោម៖

ORG 100h **; កំណត់លេខអាសយដ្ឋានចាប់ផ្តើមនៃ Instruction ដំបូង**

LSS AX, m ; Ax ផ្ទុកតម្លៃ m , Ax =1234h

RET; ត្រលប់ទៅកាន់ Os ដើម្បីធ្វើការបញ្ចប់ការប្រត្តិបត្តិ Instruction

m DW 1234h ; ប្រកាសអថេរ m ផ្ទុកតម្លៃ 1234h

DW 5678h ; ប្រកាសតម្លៃលេខអាសយដ្ឋានចាប់ផ្តើមជាក់ក្នុង SS=5678h

2.3.9. MOVSB Instruction

ចំលងតម្លៃជាលេខ ឬជា String Data ជា Byte ពី DS:[SI] ទៅកាន់ ES:[DI] , បើ DF = 0 នោះ SI =SI +1 និង DI = DI + 1។ បើ DF = 0 នោះ SI = SI -1 និង DI = DI -1

Syntax : REP MOVSB

2.3.10. MOVSBW Instruction

ចំពោះពាក្យបញ្ជានេះចំលងតម្លៃលេខ និងទិន្នន័យជា String Data ជា WORD ពី DS:[SI] ទៅ ES:[DI] , បើ DF = 0 ហើយលេខអាសយដ្ឋានកើនម្តងពីរគឺ SI =SI +2 និង DI = DI + 2។ តែបើ DF = 0 នោះលេខអាសយដ្ឋានថយម្តងពីរគឺSI = SI -2និងDI = DI +2

Syntax : REP MOVSW

2.3.11. REP instruction

ពាក្យបញ្ជា REP ប្រើសម្រាប់ឱ្យបញ្ជា MOVSB ឬ MOVSW ធ្វើការប្រតិបត្តិឡើងវិញ រហូតដល់ CX ស្មើសូន្យទើបឈប់។

Syntax : REP MOVSB/MOVSW

☺**ឧទាហរណ៍**: រកចម្លើយតាមកម្មវិធីដូចខាងក្រោម៖

CLD ; Clear DF ឱ្យស្មើ 0	REP MOSB ;ប្រតិបត្តិរហូតដល់ Cx=0
LEA SI, A1 ; ផ្ទុកអាសយដ្ឋាន A1 ចូលក្នុង SI	RED
LEA DI, A2; ផ្ទុកអាសយដ្ឋាន A2 ចូលក្នុង DI	A1 DB 1, 2, 3, 4, 5,6
MOV CX, 6; Cx ផ្ទុកតម្លៃ ៦ , CX =6	A2 DB 6 DUP(0)

2.3.12. XLAT Instruction

គឺជាពាក្យបញ្ជាប្រើសម្រាប់ធ្វើការបំលែងលេខជា Byte ទៅជាទំរង់ផ្សេង។ ដូចជាលេខកូដ ASCII ឬ EBCDIC។ ការធ្វើប្រមាណវិធីរបស់ពាក្យបញ្ជា XLAT គឺយកតម្លៃនៅលេខអាសយដ្ឋានស្មើ BX + AL ចូលទៅក្នុង AL។

Syntax : XLAT No operand

☺**ឧទាហរណ៍:** បង្ហាញលទ្ធផលតាមកម្មវិធីដូចខាងក្រោមនេះ៖

```
org 100h
mov dl, 7ch ; ផ្ទេរតម្លៃ Immedeate:7CH ចូលក្នុង DL register។
lea bx, table  ; ផ្ទុកលេខ Address: "table " ចូលក្នុង AL register។
mov al, dl  ; ចំលងតម្លៃពីក្នុង DL register ទៅដាក់ក្នុង AL register។
shr al, 4    ; ធ្វើប្រមាណវិធី SHR ចំនួន 4 ដង
xlat         ; គឺយកតម្លៃនៅ Address = BX + AL ចូលទៅក្នុង AL។
mov ah, 0eh
int 10h  ; Int10 និង Function:0eH ប្រើសម្រាប់បង្ហាញមួយតួអក្សរនៅលើអេក្រង់
mov al, dl ; ចំលងតម្លៃពីក្នុង DL register ទៅដាក់ក្នុងAL register។
and al, 0fh  ; ធ្វើប្រមាណវិធី Logical AND
xlat  ; គឺយកតម្លៃនៅ Address = BX + AL ចូលទៅក្នុង AL។
mov ah, 0eh
int 10h ; Int10 និង Function:0eH ប្រើសម្រាប់បង្ហាញមួយតួអក្សរនៅលើអេក្រង់
mov ah, 0
int 16h
ret
table db '0123456789abcdef'
```

2.3.13. LAHF and SAHF instruction

ផ្ទុកទិន្នន័យចូលទៅក្នុង AH នូវតម្លៃនៅក្នុង Flags Register ត្រង់ Low។ **SAHF** ផ្ទេរតម្លៃពីក្នុង AH ចូលក្នុង Flags Register វិញ។

AH bit : 7 6 5 4 3 2 1 0

[SF] [ZF] [0] [AF] [0] [PF] [1] [CF]

bits 1, 3, 5 are reserved.

Syntax : **LAHF ; SAHF**

☺**ឧទាហរណ៍**: ចូររកចម្លើយតាមកម្មវិធីដូចបានបង្ហាញនៅខាងក្រោម៖

```
org 100h          |     add al,bl
mov al, 0x9c      |     lahf
mov bl, 0x64      |     ret
```

2.3.14. IN និង OUT instruction

ពាក្យបញ្ជា IN ប្រើសម្រាប់ធ្វើការផ្ទេរតម្លៃពី I/O port ចូលទៅកាន់ក្នុង AL និង AX Register។ ចំណែកពាក្យបញ្ជា OUT សម្រាប់ធ្វើការផ្ទេរតម្លៃពីក្នុង AL , AX ទៅកាន់ I/O PORT វិញ។ PORT គឺជាកន្លែងតភ្ជាប់ទំនាក់ទំនងរវាងស៊ីភីយូ និងឧបករណ៍ខាងក្រៅដែលអនុញ្ញាតឱ្យឧបករណ៍ទទួលនិងបញ្ជូនទិន្នន័យទៅវិញទៅមកបាន។ វាជាកន្លែងផ្លាស់ប្តូរទិន្នន័យរវាងឧបករណ៍ពីរ ឬច្រើន។

Syntax : IN AL(AX) , PORT NUMBER

OUT PORT NUMBER ,AL(AX)

😐**ឧទាហរណ៍**: IN AL, P8 ; ផ្ទេរតម្លៃពី port p8 ចូលទៅកាន់ AL

OUT p8 , AL ; ផ្ទេរតម្លៃពីក្នុង AL Register ទៅកាន់ Port p8

អនុវត្ត៖ ចូរបញ្ចូលតម្លៃ byte គឺ 0A7h ទៅកាន់ port 110:

```
mov al, 0A7h ; al = 0A7h ,Al ផ្ទុកតម្លៃ 0A7h ឬ AL Assign តម្លៃ 0A7h
out 110, al     ; Port 110 = AL , Port 110 ផ្ទុកតម្លៃក្នុង Al
mov ax , 0 ; reset register.
in al, 110 ; read byte from port 110 into AL:
```

ចំណាំ: ការប្រើពាក្យបញ្ជាទាំងពីរនេះគឺខុសគ្នាត្រង់ពាក្យបញ្ជា IN យកតម្លៃពី PORT ទៅចូលក្នុងស៊ីភីយូ និង OUT យកតម្លៃពីក្នុងស៊ីភីយូទៅកាន់ PORT វិញ។

2.4. Stack Operation

2.4.1. Push Instruction

បញ្ជា Push គឺធ្វើការផ្ទេរតម្លៃទិន្នន័យពីក្នុង Register(ax, bx, cx, dx, si, di) ទៅកាន់ផ្នែ Stack Segment។ ការផ្ទុកតម្លៃពីក្នុង Register ទៅកាន់ផ្នែនេះត្រូវតំរៀបពីលេខអាសយដ្ឋានពីធំទៅតូច។ ដោយយកអាសយដ្ឋានមានទិន្នន័យធ្វើប្រមាណវិធីដកនឹង ១ បានអាសយដ្ឋាននៃទីតាំងដែលត្រូវផ្ទុកទិន្នន័យ និង ធ្វើតាមវិធីនេះរហូតដល់អស់វា។

Note: ចំពោះ 80186 ឡើង SP=SP-2 និង SS:[SP]=Operands

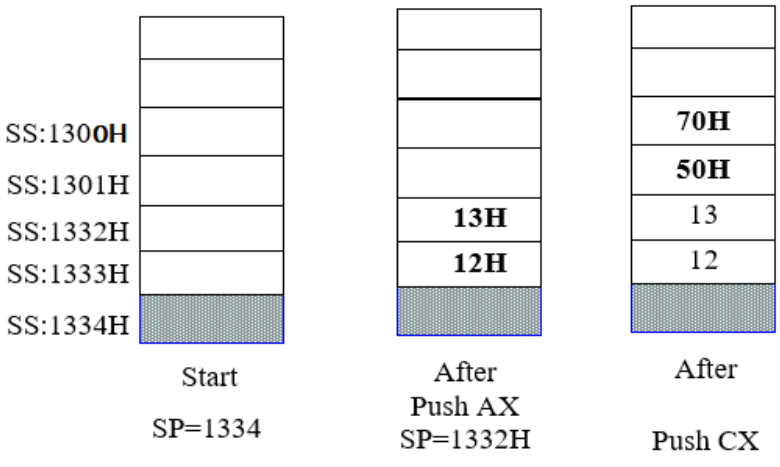
Sytax:	Statement	Machine Code
Push Register		

😐**ឧទាហរណ៍**: ចូរធ្វើប្រមាណវិធីផ្ទេរតម្លៃពីក្នុង AX =1213H និង CX = 5070H ទៅក្នុងផ្នែ Stack Segment ដោយប្រើ SP=1334H និង SS = 0700H។

1. **ចូរធ្វើការ**គូររូបបញ្ជាក់ពីការផ្ទេរតម្លៃពី Register ទៅកាន់ផ្នែ Stack Segment។
2. ចូរប្រើកម្មវិធី Debug ដើម្បីផ្ទេរតម្លៃដូចក្នុងសំនួរ a.។
3. ចូរប្រើកម្មវិធី Emu8086 ដើម្បីផ្ទេរតម្លៃដូចក្នុងសំនួរ a.។

ចម្លើយ

1. គូររូបបញ្ជាក់ពីការផ្ទេរតម្លៃពីក្នុង Register ទៅកាន់ផ្នែ Stack Segment



2. Debug Program

```
MOV AX, 1213H ; Move the value of 1213H to AX register
MOV  CX , 5070H ; Move the value : 5070H to CX register.
MOV DX, 0700H ; Move the value 0700H to DX register.
MOV SS,DX ; Move the value of DX register to SS Register.
MOV SP, 1334H ; Move the immediate 1334H to SP register
PUSH AX ; Transfer data in AX register to the Stack point to by SP register
PUSH CX
INT 3
```

0700:1330 70 50 13 12 ; Store data in memory DS:1330H=70H, DS:1331H=50H

3. កម្មវិធី Emu8086

OGR 100H ; កំណត់អាសយដ្ឋាននៃពាក្យបញ្ជាដំបូង

```
MOV DX, 0700H ; ផ្ទេរតម្លៃ Immedeate:0700H ចូលក្នុង Dx register។
MOV SS, DX  ; ផ្ទេរតម្លៃពីក្នុង Ax register ចូលក្នុង Dx register។
MOV AX, 1213H ; ផ្ទេរតម្លៃ Immedeate:1213H ចូលក្នុង Ax register។
MOV  CX , 5070H  ; ផ្ទេរតម្លៃ Immedeate:5070H ចូលក្នុង Cx register។
MOV SP, 1334H ;  ផ្ទេរតម្លៃ Immedeate:1334H ចូលក្នុង SP register។
PUSH AX ; ចំលងតម្លៃពីក្នុង AX register ទៅកាន់ផ្ទៃ Stack Segments
PUSH CX
```

RET ; ត្រលប់ទៅកាន់ OS ដើម្បីធ្វើការបញ្ចប់ការប្រតិបត្តិពាក្យបញ្ជា

OUTPUT :*0700:1332 12 13* និង *0700:1330 50 70*

☺Practice: គេមានផ្នែកនៃកម្មវិធីនៅក្នុង emu8086 IDE ដូចខាងក្រោម៖

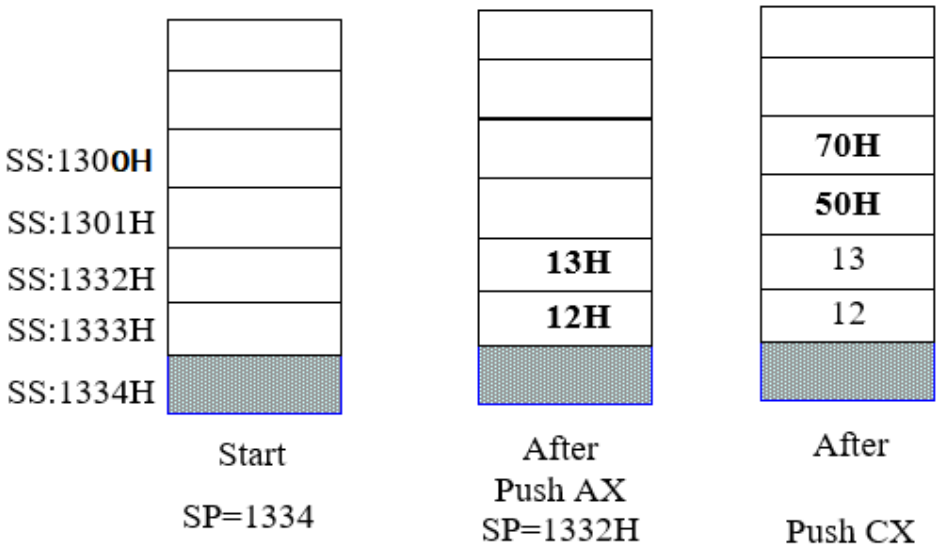
```
Org 100h
.data
    arrays db 50h, 60h ;ប្រកាសអថេរ arrays ផ្ទុកតម្លៃ 50h និង 60h
.code
    MOV BX , offset arrays ; ផ្ទេរតម្លៃលេខអាសយដ្ឋាន arrays ចូលក្នុង Bx register
    MOV  SP, 1334h ;ផ្ទេរតម្លៃ Immedeate:1334H ចូលក្នុង SP register។
    PUSH [BX] ; ផ្ទេរតម្លៃនៅអាសយដ្ឋានដែល BX  ចង្អុលទៅចូល Stack Segment
```

a. ចូរពន្យល់ PUSH [BX] ឱ្យបានក្បោះក្បាយ។
b. ចូរបង្ហាញទិន្នន័យនៅក្នុងផ្ទៃ Stack Segments បន្ទាប់ពីប្រតិបត្តិ PUSH [BX] ?

2.4.2. Pop Instruction

Pop គឺជាពាក្យបញ្ជាសម្រាប់ធ្វើការផ្ទេរតម្លៃពីក្នុងផ្ទៃ Stack Segment ទៅកាន់ Register វិញ។ ការផ្ទេរតម្លៃពីផ្ទៃ Stack Segment ទៅកាន់ Register ដោយគិតពី Address តូចទៅធំ ដោយបន្ថែមម្តងមួយ។ SP register ចង្អុលទៅអាសយដ្ឋានណា,អាសយដ្ឋាននោះគឺជា Address ដំបូងដែលត្រូវយកទិន្នន័យចេញ។

ឧទាហរណ៍: ចូរសរសេរកម្មវិធីដើម្បីធ្វើការផ្ទេរតម្លៃពីផ្ទៃ Stack Segment តាមរូបក្នុងចំទៅ AX និង CX ដោយមាន SP=1330H និង SS = 1400H។



Debug Program

E SS: 1330 70 50 13 12 ; SS:1330H=70H ,SS:1331H=50H , SS:1332H =13,...

MOV SP, 1330H ; SP ផ្ទុកតម្លៃ 1330H

MOV DX , 1400H ; ផ្ទេរតម្លៃ Immedeate:1400H ចូលក្នុង DX register

MOV SS , DX ; ផ្ទេរតម្លៃពីក្នុង Dx register ចូលក្នុង SS register

POP AX ; ផ្ទេរតម្លៃពីផ្ទៃ Stack ទៅកាន់ AX register ចង្អុលដោយ SP register

POP CX ; ផ្ទេរតម្លៃពីផ្ទៃ Stack ទៅកាន់ CX register ចង្អុលដោយ SP+2 register

INT 3

OUTPUT : DX = 5070H , CX = 1213H

Practice: គេមានកម្មវិធីដូចខាងក្រោម៖

- E 1300 50 , 60
- MOV BX , 1200
- MOV SP, 1300
- POP [BX]

a. តើ POP [BX] មាន AM ឈ្មោះអ្វី?ពន្យល់វាឱ្យបានក្បោះក្បាយ។

b. ចូរធ្វើការបង្ហាញ Data នៅក្នុងផ្ទៃ Stack Segment បន្ទាប់ពីប្រតិបត្តិ POP [BX]?

2.4.3. RET instruction

ប្រើសម្រាប់ត្រលប់ទៅកាន់ទំរង់ការដើមមួយដោយចូលទៅកាន់ near ឬ far call វិញ ឬប្រើសម្រាប់ត្រលប់ទៅកាន់ប្រព័ន្ធប្រតិបត្តិវិញ។

Sytax : RET

```
ORG 100h  ; for COM file.
        CALL p1 ; ហៅ Proceture p1 មកប្រតិបត្តិ
```

```
ADD AX, 1 ; AX = AX +1 គឺជាការបូកលេខរវាងតម្លៃនៅក្នុង AX  +1
RET       ; return to OS.
p1 PROC near     ; procedure declaration.
MOV AX, 1234h
RET     ; return to caller.
p1 ENDP
```

2.4.4. PUSHA Instruction

ប្រើសម្រាប់ធ្វើការផ្ទេរតម្លៃពីក្នុង AX, BX, CX, DX, SI, DI, SP Register ចូលទៅកាន់ផ្នែ Stack Segment។

Syntax : PUSHA

2.4.5. POPA Instruction

ផ្ទេរតម្លៃពីក្នុងផ្នែ Stack Segment ចូលទៅកាន់ AX, BX, CX, DX, SI, DI, SP

Syntax : POPA

Syntax : CLD

Flags : DF

2.4.6. PUSHF Instruction

ផ្ទុកតម្លៃពីរក្នុង Flags Register ទៅកាន់ផ្នែ Stack Segment។

```
ORG 100H
MOV SP, 1334H       ;ផ្ទេរតម្លៃ ១៣៣៤ ចូលក្នុង SP Register
MOV AL, 0X9C ;ផ្ទេរតម្លៃ 0x9C ចូលក្នុង AL Register
MOV BL, 0X64 ;ផ្ទេរតម្លៃ 0x64 ចូលក្នុង BL Register
ADD AL, BL     ; AL = AL + BL
PUSHF          ;
RET
```

💻 **OUTPUT** : SS:1332H =0257H

ចំណាំ: ចំពោះពាក្យបញ្ជា POPA, PUSHA និង Pusha ប្រើជាមួយឈ្មោះស៊ីភីយូ (CPU), ចាប់ពី 80186 ឡើងទៅ។

2.5. ASCII –BCD Conversion

ប្រមាណវិធីនព្វន្ធ BCD និងការប្រើប្រាស់លេខ **ASCII** ក្នុងអង្គចងចាំ, ការសរសេរកូដ និងការធ្វើប្រមាណវិធីផ្សេងៗជាការសំខាន់ក្នុងការសរសេរភាសា **Assembly** និងធ្វើប្រមាណវិធីដោយផ្ទាល់ជាមួយនឹងស៊ីភីយូ។ តារាងខាងក្រោមបង្ហាញពីលេខគ្រឹះសម្រាប់ធ្វើការបំលែងគោលពីរទៅគោលដប់បានលឿនបំផុត។

2.5.1. ប្រមាណវិធីនព្វន្ធ BCD

លេខប្រព័ន្ធគោលពីរដែលមានទំហំ៤ ប៊ីតតំណាងឱ្យលេខពី ០ ទៅ ៩ ហៅថា BCD ឬ ហៅថា Binary Code Decimal ។

Decimal	BCD	Hexadecimal
0	0000	0H
1	0001	1H
2	0010	2H
3	0011	3H
4	0100	4H
5	0101	5H
6	0110	6H
7	0111	7H
8	1000	8H
9	1001	9H

លេខ BCD ប្រើប្រាស់នៅក្នុងប្រព័ន្ធកុំព្យូទ័រមានពីរប្រភេទគឺលេខ Unpacked BCD និង Packed BCD។

- **Unpacked BCD: គឺជា**លេខគោលដប់មួយដោយមួយតួស្មើនឹង ៤ ប៊ីតលេខគោលពីរ ដោយកំណត់បានជា Low 4 bit ជាលេខ BCD និង Upper 4 bit មានតម្លៃស្មើនឹងលេខ0

☺ឧទាហរណ៍ Dec Unpacked BCD

7 07

- **Packed BCD:** វាជាលេខក្នុងប្រព័ន្ធ BCD (Decimal)។

☺ឧទាហរណ៍: 1. 10010010 ជាលេខ Packed BCD ព្រោះ 1001 ជាលេខ BCD និង 0010 ក៏ជាលេខ BCD ដែរ។

2. តើលេខ 11000111 ជាប្រភេទ Packed BCD ឬ Unpacked BCD? គឺមិនមែនជាលេខ Packed BCD ទេ។ ពីព្រោះលេខ 1100b មិនមែនជាលេខ BCD។ ដោយលេខ BCD មានលេខពីសូន្យដល់លេខប្រាំបួន ហើយលេខគោលពីរ 1100b = CH។

2.5.2. លេខ ASCII

នៅក្នុងការប្រើប្រាស់លេខគោលដប់ជាប្រភេទ ASCII តំណាងឱ្យលេខពីសូន្យដល់លេខប្រាំបួន។ ហើយពេលដែលយើងចុច Key ឬ ប្រើតម្លៃលេខសូន្យនៅក្នុងសញ្ញា " " គឺ "0" មានតម្លៃគោលពីរស្មើនឹង 011 0000(30H) គឺសម្រាប់ផ្តល់ទៅឱ្យកុំព្យូទ័រ។ លេខគោលពីរបន្តបន្ទាប់ដូចបានបង្ហាញក្នុងតារាងគ្រឹះខាងក្រោមគឺបង្ហាញលេខកូដមានតម្លៃសមមូលទៅលេខគោលដប់ប្រាំមួយ និងគោលដប់៖

Key	ASCII	
	HEX	BIN
0	30	011 0000
1	31	011 0001
2	32	011 0010
3	33	011 0011
4	34	011 0100
5	35	011 0101
6	36	011 0110
7	37	011 0111
8	38	011 1000
9	39	011 1001

☺**ឧទាហរណ៍**: ប្រើ Unpacked, Packed BCD និង ASCII ដើម្បីតាងឱ្យលេខគោលដប់៖

a. 12 b. 3456

a. លេខ 12 =0102h ជាលេខ Unpacked ; 12 Packed BCD ; 3132h លេខ ASCIII

b. 3456 =03040506h ជាលេខ Unpacked ; 3456 Packed BCD ; 33343536h លេខក្នុងកូដ ASCII

2.5.3. ការបំលែងពីលេខ ASCII ទៅ BCD

ការដំណើរការទិន្នន័យលេខជាប្រភេទ BCD នៅក្នុងកុំព្យូទ័របាន នោះលេខជា ASCII កូដដែលផ្តល់តាមរយៈ Keyboard នោះត្រូវបំលែងវាទៅជាលេខ BCD។ ហើយដើម្បីបំលែងពី ASCII ទៅ Packed BCD ឬ Unpacked BCD ជាមុនសិន។

2.5.4. ការបំលែងពី ASCII ទៅ Unpacked BCD

ដើម្បីធ្វើការបំលែងពីលេខ ASCII ទៅលេខ Unpacked BCD, អ្នកប្រើប្រាស់ត្រូវកំណត់លេខ "011" ត្រង់ Higher 4 bits នៃលេខ ASCII។ ដើម្បីធ្វើបានដូចនេះរាល់លេខនិមួយៗរបស់ ASCII ត្រូវធ្វើប្រមាណវិធី AND ជាមួយតម្លៃ 00001111(0FH)។ ឧទាហរណ៍ខាងក្រោមគឺជាការបំលែងលេខ ASCII ទៅជា Unpacked BCD:

☞**ឧទាហរណ៍: ចូរធ្វើការវិភាគ** និង រកលទ្ធផលពីកម្មវិធីខាងក្រោម៖

```
.data
        Ascii db "3456" ;ប្រកាសអថេរ ASCII ផ្ទុកលេខ 31h,32h,33h,...
        unpacke_BCD dd 0;ប្រកាសអថេរ unpacke_BCD ផ្ទុកលេខ 0
.code
start:
```

```
mov ecx, 2 ; cx = 2 សម្រាប់ការធ្វើប្រមាណវិធីចំនួនពីរដង
mov ebx, offset Ascii ; bx = Ascii ដែលជាលេខអាសយដ្ឋាននៃទិន្នន័យ
lea edi, unpacke_BCD ; di = unpacke_BCD
back:
    mov ax, [ebx] ; Bx ចាប់យកតម្លៃ២ Bytes ពី Memory ចូល AX
    and ax, 0F0Fh ; ax = ax and 0X0F0Fh
    xchg ah, al ; ធ្វើការផ្លាស់ប្តូរតម្លៃរវាង al ទៅ ah
    mov [edi+2], ax ; ផ្ទេរតម្លៃលទ្ធផលពីក្នុង Ax ទៅ [di+2]
    add ebx, 2  ;ធ្វើបង្កើន Address ក្នុង ebx ម្តង ២
    sub edi, 2 ; ធ្វើបង្កើនលេខ Address ក្នុង edi ម្តង ២
    dec ecx ; ធ្វើការបន្ថយលេខក្នុងម្តងមួយ
    jnz back ; ឆ្ពោះទៅកាន់ back
printf( "Unp_BCD=%lx \n",unpacke_BCD );
getkey  ; ធ្វើការទប់អេក្រង់បញ្ចេញលទ្ធផល
ret  ; ត្រលប់ទៅកាន់ OS ធ្វើការបញ្ចប់ការប្រតិបត្តិពាក្យបញ្ជា
end start ; បញ្ចប់កម្មវិធី
```

2.5.5. ការបំលែងពី ASCII ទៅ packed BCD

ក្នុងការធ្វើការបំលែងលេខ ASCII ទៅលេខ Packed BCD តាមពីរជំហាន៖

1. ត្រូវបំលែងជា Unpacked BCD
2. បន្ទាប់មកផ្សំវាឱ្យទៅជា Packed BCK ដោយធ្វើតាមប្រមាណវិធីខាងក្រោម៖
    - ផ្ទេរ Data ចូលក្នុង Register
    - ធ្វើប្រមាណវិធី AND រវាងតម្លៃក្នុង Register ជាមួយ 0F0FH
    - ធ្វើប្រមាណវិធី XCHG រវាងតម្លៃក្នុង AH និង AL
    - ធ្វើប្រមាណវិធី SHL ក្នុង AH ចំនួន ៤ ដង
    - ធ្វើប្រមាណវិធី OR រវាងតម្លៃក្នុង AH និង AL

ឧទាហរណ៍ខាងក្រោមបញ្ជាក់ពីការបំលែងពី ASCII ទៅជា Packed BCD:

Key	ASCII	PACKED BCD
4	34	0110100
7	37	0110111

ចូរមើល និងធ្វើការវិភាគកម្មវិធីដូចខាងក្រោម៖

```
.data
    Ascii db "47" ; ប្រកាសអថេរ Ascii ផ្ទុកលេខ 34h, 37h
    Pack_BCD db 0 ;ប្រកាសអថេរ Pack_BCD ផ្ទុកលេខ 0
.code
Start:
```

```
mov ebx ,offset Ascii        or al,ah
mov ax, [ebx]                mov Pack_BCD, al
and ax, 0f0fh                printf( "Unp_BCD=%x \n",Pack_BCD )
xchg ah, al                  getkey( )
mov cl, 4                    ret
shl ah, cl                   end start
```

2.5.6. ការបំលែងពី Packed BCD ទៅ ASCII

ក្នុងការធ្វើប្រមាណវិធីបំលែងពីលេខ Packed BCD ទៅលេខកូដ ASCII នោះត្រូវអនុវត្តតាមជំហានតាមលំដាប់លំដោយដូចខាងក្រោម៖

- ត្រូវបំលែងទៅ Packed BCD ទៅ Unpacked BCD
- ផ្ទេរ Data ចូលក្នុង AL Register
- ផ្ទេរតម្លៃពី AL to AH
- **ធ្វើប្រមាណវិធី** SHR ក្នុង AH ចំនួន ៤ ដង
- ធ្វើប្រមាណវិធី AND រវាងតម្លៃក្នុង AX ជាមួយតម្លៃ OF0FH

2. យកលេខ Unpacked BCD នេះ OR ជាមួយតម្លៃ 30H។

ឧទាហរណ៍ខាងក្រោមបញ្ជាក់ពីការបំលែងពី Packed BCD ទៅជា ASCII:

Packed BCD	Unpacked BCD
29	02 and 09
10	00000010 and 00001001

ចូរធ្វើការវិភាគកម្មវិធីដូចខាងក្រោមដើម្បីធ្វើការបំលែងពី Packed BCD ទៅ ASCII:

```
.data
        Pack_BCD db 0x47 ;ប្រកាសអថេរ Pack_BCD ផ្ទុកតម្លៃ 0x47
         Ascii dw 0; ប្រកាសអថេរ Ascii ផ្ទុកតម្លៃ 0
.code
start:
        lea ebx,Pack_BCD     ; ផ្ទុក Effective Address ទៅ bx
        mov al, [ebx]        ; AL ផ្ទុកតម្លៃដែលបានមកពី Bx ចង្អុលទៅ
        mov ah, al           ; បញ្ចូលតម្លៃ AL ទៅ AH ដើម្បីបាន AH = AL
        mov cl, 4            ; បញ្ចូលតម្លៃ 4 ទៅ AH ដើម្បីបាន CL = AL
        shr ah, cl           ; ធ្វើប្រមាណវិធីរំកិលលេខគោលពីរចំនួន ៤ ដង
        and ax, 0x0f0f ; ធ្វើប្រមាណវិធីតក្ក And ax = ax and 0x0f0f
        or ax, 0x3030  ; ធ្វើប្រមាណវិធីតក្កax = ax or 0x3030
        mov Ascii, ax ; បញ្ចូលតម្លៃ AX ទៅ Ascii ដើម្បីបាន Ascii = ax
printf( "Unp_BCD=%x \n",Ascii ) ;បង្ហាញតម្លៃនៅលើអេក្រង់បញ្ចេញតម្លៃ
getkey( ); Pause Screen ហើយពេលចុច Key ផ្សេងៗដើម្បីបិទវា
ret
```

```
end start
```

2.5.7. BCD Addition

DAA គឺជាពាក្យបញ្ជាប្រើបន្ទាប់ពីប្រមាណវិធីបូក BCD ដើម្បីបំលែងលទ្ធផលទៅជា BCD ត្រឹមត្រូវ។ វាបន្ថែមលេខ ៦ ទៅលើ Lower 4 bits នៃ AL បើ LN > 9 ឬ បើ AF = 1។ វាបន្ថែមលេខ ៦ ទៅលើ Upper 4 bits នៃ AL បើ UN > 9 ឬ បើ CF = 1។

ទំរង់ទូទៅ: DAA ;

Flags : SF , ZF , AF , PF, CF , OF

☞**ឧទាហរណ៍:** ចូរវិភាគ និងរកលទ្ធផលតាមកម្មវិធីដូចខាងក្រោម៖

```
.Data
   x      Db      78h ; ប្រកាសអថេរ x ដើម្បីផ្ទុកតម្លៃ 78h
   y      Db      68h; ប្រកាសអថេរ y ដើម្បីផ្ទុកតម្លៃ 68h
.code
Start:
        MOV  AL, x ; ផ្ទេរតម្លៃ x ចូលក្នុង AL register។
        ADD AL,  y ; ធ្វើការបូកលេខរវាងតម្លៃនៅក្នុង AL ជាមួយនឹងតម្លៃក្នុង y
        DAA ; យកលទ្ធផលនៅក្នុង AL កែឱ្យទៅជាលេខ BCD ត្រឹមត្រូវ។
        .if Carry? ; បើសិន Carry ស្មើនឹង ១ អនុវត្ត Mov ah, 1
            mov ah, 1
         .endif
        PrintHex ax
Ret
End start
```

☞**ឧទាហរណ៍ 2:** រកលទ្ធផលពីកម្មវិធីខាងក្រោម៖

```
        MOV  AL, 97H; ផ្ទេរតម្លៃ Immediate :97H ចូលក្នុង AL register។
        ADD AL, 38; ជាការបូកលេខរវាងតម្លៃនៅក្នុង AL ជាមួយនឹង 38H
        DAA ; យកលទ្ធផលនៅក្នុង AL កែឱ្យទៅជាលេខ BCD ត្រឹមត្រូវ។
```

2.5.8. BCD Subtraction

DAS (Decimal Adjust after Subtraction) គឺប្រើបន្ទាប់ពីប្រមាណវិធីដកលេខ BCD ដើម្បីធ្វើការបំលែងលទ្ធផលទៅជា BCD ត្រឹមត្រូវ។ វាដកលេខ៦ ទៅលើ Lower 4 bits នៃ AL បើ LN > 9 ឬ បើ AF = 1។ វាដកលេខ ៦ ទៅលើ Upper 4 bits នៃ AL បើ UN > 9 ឬ បើ CF = 1 ។

Format: DAS ;

Flags: SF, ZF, AF, PF, CF, OF

☞**ឧទាហរណ៍១:** រកលទ្ធផលពីកម្មវិធីខាងក្រោម៖

```
        MOV  AL, 97H; ផ្ទេរតម្លៃ Immediate :97H ចូលក្នុង AL register។
```

SUB AL, 38H ; ជាការដកលេខរវាងតម្លៃនៅក្នុង AL ជាមួយនឹង 38H

DAS ; យកលទ្ធផលនៅក្នុង AL កែឱ្យទៅជាលេខ BCD ត្រឹមត្រូវ។

MOV [450] , AL ; ; ផ្ទេរតម្លៃពីក្នុង AL register ទៅកាន់លេខ Addr:450

☞**ឧទាហរណ៍២:** រកលទ្ធផលពីកម្មវិធីខាងក្រោម៖

MOV AL, 97H ; ផ្ទេរតម្លៃ Immediate :97H ចូលក្នុង AL register។

SUB AL, 38H; ជាការដកលេខរវាងតម្លៃនៅក្នុង AL ជាមួយនឹង 38H

DAS; យកលទ្ធផលនៅក្នុង AL កែឱ្យទៅជាលេខ BCD ត្រឹមត្រូវ។

MOV [450] , AL ; ; ផ្ទេរតម្លៃពីក្នុង AL register ទៅកាន់លេខ Addr:450

MOV [451] , AL

2.5.9. AAA(ASCII Adjust After Addition)

AAA នេះគឺប្រើបន្ទាប់ពីពាក្យបញ្ជា ADD ដែលបានបូកលេខពីរ ASCII។ វាមានតួនាទីបំលែងលេខលទ្ធផលដែលបានមកពី ADD ទៅជាលេខ Unpacked BCD។ ការប្រតិបត្តិរបស់វាពិនិត្យទៅលើតម្លៃ LN(Lower Nibble) នៃ AL។ បើ LN មានតម្លៃពី A→F ឬ AF = 1, AAA ធ្វើការថែម 0110(6H) ទៅលើ AL និងបន្ថែមលេខ 1h ទៅលើ AH បន្ទាប់មកលុបតម្លៃក្នុង HN របស់ AL។ ដើម្បីលទ្ធផលជា ASCII ត្រូវប្រើពាក្យបញ្ជា OR ជាមួយនឹងតម្លៃ 30H។

Format: AAA ;

Flags: SF, ZF, AF, PF, CF, OF

☞**ឧទាហរណ៍1:** រកលទ្ធផលពីកម្មវិធីខាងក្រោម៖

MOV AL, 37H; ផ្ទេរតម្លៃ Immediate :37H ចូលក្នុង AL register។

ADD AL, 39H; ជាការបូកលេខរវាងតម្លៃនៅក្នុង AL ជាមួយនឹង 31H

AAA; បំលែងលេខលទ្ធផលដែលបានមកពី ADD ទៅជា Unpacked BCD

OR AX , 3030h ;

MOV [450] , AX

☞**ឧទាហរណ៍2:** រកលទ្ធផលពី Program ខាងក្រោម៖

MOV AL, 37H ; ផ្ទេរតម្លៃ Immediate :37H ចូលក្នុង AL register។

ADD AL, 38

AAA បំលែងលេខលទ្ធផលដែលបានមកពី ADD ទៅជា Unpacked BCD

OR AX , 3030

MOV [450] , AL

INT 3

2.5.10. AAS (ASCII Adjust After Subtraction)

AAS នេះគឺប្រើបន្ទាប់ពីពាក្យបញ្ជា SUB ដែលបានដកលេខពីរ ASCII។ វាមានតួនាទីបំលែងលេខលទ្ធផលដែលបានមកពី SUB ទៅជា Unpacked BCD ការប្រតិបត្តិរបស់វាពិនិត្យទៅលើតម្លៃ Lower Nibble នៃ AL។ បើ LN មានតម្លៃពី A→F ឬ AF = 1, AAA ដក 0110 ពី AL, និងដកលេខ 1 ពី AH និង លុបតម្លៃក្នុង AH។ ដើម្បីលទ្ធផលជា ASCII ត្រូវ OR ជាមួយតម្លៃ 30H។

Format: AAS ;

Flags: SF, ZF, AF, PF, CF, OF

☞**ឧទាហរណ៍:** រកលទ្ធផលពី Program ខាងក្រោម៖

```
MOV  AL, 39 ; ផ្ទេរតម្លៃ Immediate :39 ចូលក្នុង AL register។
SUB AL, 34 ; ជាការដកលេខរវាងតម្លៃនៅក្នុង AL ជាមួយនឹង 34H
AAS; បំលែងលេខលទ្ធផលដែលបានមកពី SUB ទៅជា Unpacked BCD
MOV AL, 30
MOV [450] , AL
INT 3
```

2.5.10. AAM (ASCII Adjust After Multiplication)

AAM ពាក្យបញ្ជាប្រើបន្ទាប់ពី MUL ដែលបានគុណលេខពី Unpacked BCD ដែល Input Data ជា ASCII។ វាធ្វើការបំលែងលទ្ធផលក្នុង AL ជាលេខ Hexa ទៅជា Unpacked BCD, ដោយយកលេខក្នុង AL ចែកនឹង 0AH (បើគិតជា Dec ត្រូវយកលេខក្នុង AL បំលែងជា Dec រួចនឹង ១0), បាន Quotient ផ្ទុកក្នុង AH និង Remainder ផ្ទុកក្នុង AL។

Flags: SF, ZF, AF, PF, CF, OF

☞**ឧទាហរណ៍1:** រកលទ្ធផលពី Program ខាងក្រោម៖

```
MOV  AL, "9" ; ផ្ទេរតម្លៃ Immediate :37H ចូលក្នុង AL register។
AND  AL, 0Fh ; ធ្វើប្រមាណវិធីឈ្នាប់ឬតម្លៃក្នុង DL ជាមួយនឹង OFh
MOV  DL, "7" ; ផ្ទេរតម្លៃ Immediate :37H ចូលក្នុង DL register។
AND  DL, OFh ; ធ្វើប្រមាណវិធីឈ្នាប់នឹងតម្លៃក្នុង DL ជាមួយនឹង OFh
MUL DL ; ការគុណលេខតម្លៃក្នុង AL ជាមួយនឹង DL
AAM     ; ធ្វើការបំលែងលទ្ធផលក្នុង AL ជាលេខ Hexa
OR AX , 3030H
```

☞**ឧទាហរណ៍2:** **a.** សរសេរកម្មវិធី Assembly ខាងក្រោមដើម្បីគុណលេខ 38hx38H:

b. រកលទ្ធផលពីកម្មវិធីក្នុង a.

2.5.12. AAD (ASCII Adjust Before Division)

AAD គឺជាពាក្យបញ្ជាប្រើមុនពាក្យបញ្ជា DIV ដើម្បីបំលែងលេខតួចែក(Dividend) ទៅជា Unpacked BCD។ វាធ្វើការបំលែងលេខ Dividend ដោយយកលេខក្នុង AH គុណនឹង 10(A), រួចយក Product បូកតម្លៃក្នុង AI។

Format: AAD ;

Flags: SF, ZF, PF, OF

☞**ឧទាហរណ៍:** រកលទ្ធផលពីកម្មវិធីខាងក្រោម៖

```
MOV  AX, 3238 ; កំណត់ AX ផ្ទុកតម្លៃ 3238
MOV BL , 37    ;  កំណត់ BL ផ្ទុកតម្លៃ 37
AND BL , 0Fh   ; BL = BL and OFH
AND AX , 0F0Fh ; AX = AX and OF0Fh
```

```
AAD     ;ធ្វើការបំលែងលេខលទ្ធផលឱ្យទៅជាលទ្ធផលត្រឹមត្រូវ
DIV BL ; AL = AL/ BL
OR  AX , 3030
INT 3
```

2.6. Bit Shift

2.6.1. ROR(Rotate right)

ROR គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់រំកិលលេខគោលពីរពីខាងឆ្វេងទៅខាងស្តាំ, ដោយ Bits ត្រង់ LSB ត្រូវទៅបំពេញនៅត្រង់ Bits MSB និង ក្លាយទៅជា CF។

Syntax: ROR Reg , CL

Reg អាចប្រើទំហំ 8 bits ឬ 16 bits

- CL ផ្ទុកចំនួនដងនៃ Rotate Right

☞**ឧទាហរណ៍:**

```
MOV AL , 36H ; AL = 0011 0110
ROR  AL , 1     ; AL = 0001 1011  CF=0
ROR  AL , 1     ; AL = 1000 1101  CF=1
ROR  AL , 1     ; AL = 1100 0110  CF=1
```

☞**ឧទាហរណ៍**: ចូរសរសេរកម្មវិធីមួយរាប់លេខ 1 នៅក្នុង Byte មួយ។

```
From the data segment
        DATA1 DB 97H
        COUNT DB ?
From the code segment
        SUB BL,BL
        MOV DL,8
        MOV AL,DATA1
Againt: ROL AL,1
        NC NEXT
        INC BL
Next:   DEC DL
JNZ Againt
        MOV COUNT ,BL
```

☞**ឧទាហរណ៍**: ចូរសរសេរកម្មវិធីមួយរាប់លេខ 1 នៅក្នុង word មួយ។

2.6.2. ROL(Rote Left)

ROL គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់ធ្វើការរំកិលលេខគោលពីរ ពីស្តាំទៅខាងឆ្វេង, ដោយ Bits ត្រង់ MSB ត្រូវទៅបំពេញនៅត្រង់ Bits LSB និង ក្លាយទៅជា CF។

Syntax: ROL Reg , CL

Reg អាចប្រើ 8bits ឬ 16 bits និង CL ផ្ទុកចំនួនដងនៃ Rotate Right

☞**ឧទាហរណ៍:**

```
MOV AL , 72H ; AL = 0111 0010
ROL  AL , 1     ; AL =  1110 0100  CF=0
ROL  AL , 1     ; AL =  1100 1001  CF=1
ROL  AL , 1     ; AL =  1001 0011  CF=1
```

2.6.3. RCR(Rotate Right Through Carry)

RCR គឺជា Instructions មួយប្រើសម្រាប់រំកិលលេខគោលពីរពីខាងឆ្វេងទៅខាងស្តាំ, យកតម្លៃលេខ CF ទៅបំពេញនៅត្រង់ Bits MSB និង Bits ត្រង់ LSB ក្លាយទៅជា CF។

Syntax: RCR Reg , CL

Reg អាចប្រើ 8bits ឬ 16 bits និងប្រើ CL ផ្ទុកចំនួនដងនៃ Rotate Right

☞**ឧទាហរណ៍:**

```
CLC ; CF = 0
MOV AL , 26H ; AL = 0010 0110
RCR  AL , 1     ; AL = 0001 0011  CF=0
RCR AL , 1     ; AL =  0000  1001  CF=1
RCR AL , 1
RCR AL , 1
RCR AL , 1
```

2.6.4. RCL(Rotate Left Through Carry)

RCL គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់ធ្វើការរំកិលលេខគោលពីរ ពីខាងស្តាំទៅខាងឆ្វេង, យកតម្លៃរបស់ CF ទៅបំពេញនៅត្រង់ Bits LSB និង Bits ត្រង់ MSB ក្លាយទៅជា CF

Syntax: RCR Reg , CL

Reg អាចប្រើ 8bits ឬ 16 bits និងប្រើ CL ផ្ទុកចំនួនដងនៃ Rotate Right

☞**ឧទាហរណ៍**:

```
STC ; CF = 1
MOV BL , 15H ; AL = 0010 0100
RCL  BL , 1     ; AL = 0010 1011  CF=0
RCL BL , 1     ; AL =  0101  0110  CF=0
Or :
STC ; CF = 1
MOV BL , 15H ; AL = 0010 0100
MOV CL , 2
RCL BL , CL
```

មេរៀនទី ៤ អាសយដ្ឋានម៉ូត ម៉ាស៊ីនភូដ និងកម្មវិធី Debug

1. អាសយដ្ឋានម៉ូត (Addressing Mode)

ស៊ីភីយូអាចដំណើរការយកទិន្នន័យ(operands)ផ្សេងៗ ដោយបានមកពីការផ្ទេរពី៖

- Memory(ប្រើអថេរដូចជា x, y,z,...) ទៅកាន់ Register(AX, BX, CX, DX, SI, DI)
- Register(AX, BX, CX, DX, SI, DI) ទៅកាន់ Register
- Register ទៅកាន់ Memory វិញ

  វិធីដំណើរការទិន្នន័យពីទៅវិញទៅមករវាង RAM ទៅកាន់ CPU និងពី Register មួយទៅកាន់ Register មួយវិធីបែបនេះហៅថា Addressing Mode(AM)។ ចំនួន AM គឺគេបានកំណត់នៅពេលបង្កើត Microprocessor និងមិនអាចផ្លាស់ប្តូរបានដោយអ្នកសរសេរកម្មវិធី ឬអ្នកប្រើប្រាស់បានឡើយ។ វាចែកចេញជា ៧ ប្រភេទដូចខាងក្រោម៖

- Register
- Immediate
- direct
- Register Indirect
- Based Relative
- Indexed Relative
- Based Indexed Relative

1.1. Register AM

ការប្រើប្រាស់ម៉ូតនេះជាប់ទាក់ទងទៅនឹងរំកិល(Move), ធ្វើប្រមាណវិធីបូក ឬប្រមាណវិធីផ្សេងៗដោយទិន្នន័យប្រើឈ្មោះ Register និង Register។ Memory មិនដំណើរការនៅពេលម៉ូតនេះប្រតិបត្តិ។ ដូច្នេះធ្វើឱ្យវាដំណើរការបានលឿនទៅតាមផ្នែកនិមួយៗរបស់វា។

ឧទាហរណ៍ខាងក្រោម ជា Register Addressing Mode:

(a). MOV BX , DX ; Copy the contents of DX into BX

(b). MOV ES , AX ; Copy the contents of AX into ES

(c). ADD AL , BL ; Add the contents of AL to contents of BL

(d). ADD SI , DI ;យកតម្លៃក្នុង SI បូកនឹង DI និងផ្ទុកលទ្ធផលក្នុង SI

✡**ចំណាំ: ចំពោះ** Statement ខាងក្រោមបង្ហាញជា Register AM ដែលមិនអាចប្រើបាន

a. MOV AX , DL ; ការប្រើបញ្ជា Mov ត្រូវឈ្មោះRegister នៅទីតាំងទាំងពីរមានទំហំប៉ុនគ្នា
b. ADD DI , AL ;ការប្រើបញ្ជា ADD ត្រូវឈ្មោះRegister នៅទីតាំងទាំងពីរមានទំហំប៉ុនគ្នា
c. MOV SS , IP ; ដោយ IP register ប្រើសម្រាប់ឆ្ពោះទៅកាន់លេខអាសយដ្ឋាននៃពាក្យបញ្ជា

☺**ឧទាហរណ៍**: ១.បើសិនតម្លៃនៅក្នុង AX មានតម្លៃ 1000H និង BL មានតម្លៃ 80H។ តើលទ្ធផលនៃការនៅក្នុង AX register ស្មើនឹងតម្លៃប៉ុន្មាន? បន្ទាប់ពីប្រតិបត្តិបញ្ជា: MOV AL , BL?

២.សរសេរកូដចេញពីសំនួរ ១.

1.2. Immediate AM

ការប្រើប្រាស់នូវម៉ូតនេះតម្លៃទិន្នន័យនៅទីតាំងដើមគឺជាតម្លៃថេរ ហើយតម្លៃនេះគឺជាតម្លៃផ្ទាល់របស់ពាក្យបញ្ជា។ ក្នុងពេលប្រតិបត្តិពាក្យបញ្ជាតម្លៃថេរបំលែងទៅជាលេខគោលពីររួមជាមួយនឹង Opcode។ វាអាចប្រើសម្រាប់ផ្ទុកពត៌មានពី Memory ទៅកាន់ Register ផ្សេងៗ លើកលែងតែ Segment Register និង Flag Register។

ឧទាហរណ៍ខាងក្រោមជា Register Addressing Mode:

(a). MOV AX , 2550 ; Move 2550 into AX.

(b). MOV CX , 0625 ; Move the decimal values 625 into CX.

(c). ADD AL , 40H ; add the content of al with imm:40H.

✡**ចំណាំ:** ដើម្បីផ្ទេរតម្លៃចូលទៅក្នុង Segment Register នោះត្រូវអនុវត្តតាមជំហានទី១ និងទី២ ដូចខាងក្រោមនេះ៖

1. ផ្ទេរតម្លៃចូលក្នុង AX, BX, CX, DX (General_purpose register)

2. ផ្ទេរតម្លៃពី General_purpose register ទៅ Segment register

☺**ឧទាហរណ៍:** គេមានតម្លៃនៅក្នុង AX=1234H, ផ្ទេរតម្លៃនេះចូលក្នុង DS , ES register

mov ax, 1234H ; ផ្ទេរតម្លៃ 1234h ចូលក្នុង AX register

mov ds, ax ; ផ្ទេរតម្លៃពីក្នុង DS ចូលក្នុង DS register

mov es, ax ; ផ្ទេរតម្លៃពីក្នុង ax ចូលក្នុង ES register

1.3. direct AM

នៅក្នុងម៉ូតនេះតម្លៃថេរគឺស្ថិតនៅក្នុងទីតាំង Memory និងប្រើលេខអាសយដ្ឋានដោយផ្ទាល់ដើម្បីហៅធ្វើការហៅតម្លៃលេខនោះមកប្រតិបត្តិនៅពេលដំណើរការពាក្យបញ្ជាមួយ, នៅក្នុងម៉ូតនេះ លេខអាសយដ្ឋានផ្សំឡើងពី DS និង BX បង្កើតបានជា Physical Address។

ឧទាហរណ៍ខាងក្រោម ជា Register Addressing Mode:

(a). MOV [7000] , AX : Move the contents of AX into DS:7000 and DS:7001.

(b). MOV DL , [2400] : Move the contents of DS:2400 into CX.

(c). ADD [5000] , DL : Move the contents of DL into DS:5000.

គួរកត់សំគាល់ដែរថានៅក្នុង Immediate addressing Mode, តម្លៃ Operand ផ្តល់ទៅឱ្យ Instruction ដោយភ្លាមៗ។ ចំណែកឯនៅក្នុង Direct AM, តាមរយៈ Address នៃ Operand គឺផ្តល់ទៅឱ្យ Instruction។

☺**ឧទាហរណ៍:** រក Physical Address នៃទីតាំង Memory និងជាតុរបស់វាបន្ទាប់ពីប្រតិបត្តិពាក្យ បញ្ជា ដូចខាងក្រោម និង ប្រើកម្មវិធីធ្វើការតេសសំនួរនេះ?

MOV AX , 8899H

MOV [3518H] , AX

សន្មត់ DS=1512H

<u>ចម្លើយ</u>

ក្នុងពេលស៊ីកីយូប្រតិបត្តិ MOV AX , 8899H នោះបាន AX មានតម្លៃ 8899H នឹងបន្ទាប់ពីប្រតិបត្តិ MOV [3518h] , AX , គឺផ្ទេរតម្លៃក្នុង AX ទៅ DS:3518h=99H និង DS:3519h=88H។

☺**ឧទាហរណ៍**: ចូរមើលកូដធ្វើម្រមាណវិធីបូកលេខ៣២ប៊ីត Hex: Data1 = 12345678H + Data2 = ABCF8765 ដូចខាងក្រោម និងបង្ហាញ Addressing Mode តាមពាក្យបញ្ជានិមួយៗ៖

```
Org 100h
.data
    data1 dd   12345678h ; ប្រកាសអថេរ data1  ផ្ទុកលេខ 12345678h
    data2 dd 0ABCF8765h          ; ប្រកាសអថេរ data2  ផ្ទុកលេខ0ABCF8765h
```

```
        mem dd 0                    ; ប្រកាសអថេរ mem  ផ្ទុកលេខ 0
.code
start:
        mov ax, word ptr data1 ; ax= 5678h ដោយលេខនេះជា L-Word នៃ data1
        mov bx, word ptr data2  ; bx= 8765h ដោយលេខនេះជា L-Word នៃ data2
        add ax , bx ; ax= ax + bx =5678h + 8765h =ddddh
        mov word ptr mem , ax ; ds:memory: dd  dd
        mov cx, word ptr data1+2 ; cx= 1234h ដោយលេខនេះជា H-Word នៃ data1
        mov dx, word ptr data2+2  ; xx= ABCDh ដោយលេខនេះជា H-Word នៃ data2
        adc dx , cx ;dx = dx + cx = 1234h + ABCDh =BE03h
        mov word ptr mem+2, dx      ;DS:mem = 03 BE
ret
```

➢ ការផ្ទុកទិន្នន័យនៃ data1 db 0x12345678 និង data2 db 0xABCF8765 គឺតំរៀបពីខាងស្តាំទៅខាងឆ្វេងដោយ Logical Addresss: **0700:0102** : 78 56 34 12 65 87 CF AB ; សន្មត់ DS=0700 និង oftset Memory = 0102h

➢ Instruction : mov ax, word ptr data1 ; ពេល Compile ទៅជាទំរង់ mov ax, word ptr [0102H] ; មានន័យថាជាការចាប់យកតម្លៃនៅ Address data1 គឺ 0102H និង 0103H ចូលក្នុង ax regiser ; ជាលទ្ធផលពេល Run : mov ax , word ptr data1 ; ax= **5678**H

1.4. Register Indirect AM

នៅក្នុងម៉ូតនេះ, អាសយដ្ឋាននៃទីតាំង Memory ដែល operand ស្ថិតនៅគឺត្រូវប្រើ Register មួយ។ Register ទាំងនេះគឺមាន BX, DI, SI ប្រើសម្រាប់ឆ្ពោះទៅកាន់អាសយដ្ឋានរបស់ទិន្នន័យ ដើម្បីរក Physical address នៅក្នុងម៉ូតនេះត្រូវប្រើ Register DS និង Pointer Register។

(a). MOV AL , [BX] ; Move DS:BX into AL

(b). MOV CL , [SI] ; Move DS:SI into CL .

(c). ADD [DI] , DX ; Move the contents of DX into DS:DI and DS:DI+1

(d). MOV AX , [SI]

គួរកត់សំគាល់ដែរថា BX នៅខាងក្រៅ[ ], កម្មវិធីបកប្រែដោយ MOV យកធាតុនៃ BX ចូលទៅក្នុង AL (នោះកម្មវិធីផ្តល់នូវ error ព្រោះប្រភពដើមនិងគោលដៅ មានទំហំខុសគ្នា)។ តែសរសេរ BX នៅក្នុង[ ]មានន័យថាឆ្ពោះទៅកាន់ Offset address ដើម្បីកំណត់យក Content នៅក្នុងទីតាំង Memory។

☺**ឧទាហរណ៍:** សន្មត់ DS=1060H, SI= 2498H, BX = 2220H និង AX=1AADH រក Physical Address នៃទីតាំង Memory និង Contents របស់វាបន្ទាប់ពីប្រតិបត្តិ Instruction ខាងក្រោម៖

MOV AL , [BX] និង MOV [SI] , AX

✍**ឧទាហរណ៍:** សរសេរកូដដើម្បីយក Data ពីក្នុង Memory ពី LA=0700:0100 តាមរូបខាងក្រោមទៅទីតាំង Destination ដែលមាន Logical Address: 0700:0100 **41 42 43 44 45 46 47 48**- 00 00 0000

1.5. Based Relative AM

នៅក្នុងម៉ូតនេះ, ប្រើ Register BX និង BP ផ្សំជាមួយតម្លៃ Displace_ment , បង្កើតបានជា Effective Address។ នៅក្នុង Mode នេះដើម្បីរក PA នៃ Data ត្រូវប្រើ DS ជាមួយ BX និង SS ជាមួយនឹង BP។

ឧទាហរណ៍: ខាងក្រោមជា Based Relative Addressing Mode ប្រើ BX Register:

```
MOV CX , [BX] + 10 ; Move DS:BX+5 into CX .
                   ;PA=DS(SBL)+BX+5
```

- MOV ជា Instruction របស់ CPU
- BX ឬ CX ហៅថា Register
- 5 ជាតម្លៃ Displacement
- BX + 5 ហៅថា Effective address
- Effictive Address ជាទីតាំងរបស់ Data ដែលត្រូវដំណើរការ

ការជ្រើសរើសកូដគឺ MOV CX , [BX] + 10 អាចសរសេរ MOV CX , 10[BX]។ គួរកត់សំគាល់ដែរថា Low address contents នឹងផ្ទេរទៅក្នុង CL និង High address contents ផ្ទេរទៅក្នុង CH

ឧទាហរណ៍ខាងក្រោម ជា Based Relative Addressing Mode ប្រើ BX Register:

```
MOV AL , [BP] + 5 ; PA=SS(SBL) + BP + 5
```

✡ចំណាំ: 1. add al, vector[bx] ; Register Indirect AM

2. mov filename[bx+2], 0

✍**ឧទាហរណ៍:** ចូរសរសេរកូដដើម្បីយក Data ពីក្នុង Memory ពី LA=0700:0107 តាមរូបខាងក្រោមទៅទីតាំង Destination ដែលមាន Logical Address:0700:107: **41 42 43 44 45 46 47 48**- 00 00 0000 00 00 00 00

1.6. Indexed Relative AM

ការប្រើម៉ូតទី៦នេះគឺដូចទៅនឹងម៉ូតនៅចំនុចទី៥ ដែរ, គ្រាន់តែការប្រើ Register ខុសគ្នា, ដោយនៅក្នុងម៉ូតនេះប្រើ DI និង SI Register ជា Offset Address។

ឧទាហរណ៍ខាងក្រោមជា Indexed Relative Addressing Mode:

```
MOV DX , [SI] + 5 ; Move DS:SI + 5 and DS: SI + 6 into DX
                  ; PA = DS(SBL) + SI + 5 and PA = DS(SBL) + SI + 6
MOV CL , [DI] + 20 ;
```

😐**ឧទាហរណ៍:** សន្មត់ DS = 4500 , SS = 2000 , BX = 2100 , SI = 1486 , DI = 9500 , BP = 7814 និង AX = 2512 ។

ចូរបង្ហាញពី PA ដែលតម្លៃក្នុង AX ត្រូវផ្ទុកនៅក្នុង Memory បន្ទាប់ពីប្រតិបត្តិពាក្យបញ្ជានីមួយៗដូចខាងក្រោមនេះ៖

a. MOV [BX] + 20 , AX
b. MOV [DI] + 4 , AX
c. MOV 10[SI] , AX
d. MOV 12[BP] , AX

ចម្លើយ

PA = Segment Register (SBL) + Offset register + displacement

a. DS:BX + 20 location 47120=(12) and 47121 = (25)

ឧទាហរណ៍: ប្រូក្រាមខាងក្រោមគឺជាការបូកពីរ Array:

```
org 100h
jmp start
vec1 db 1, 2, 5, 6
vec2 db 3, 5, 6, 1
vec3 db ?, ?, ?, ?
start:
    lea si, vec1 ; Load Effective Address into si register
    lea bx, vec2 ; Load Effective Address into bx register
    lea di, vec3; Load Effective Address into di register
    mov cx, 4
sum:
   mov al, [si]+3
   add al, [bx]+3 ; al = al + value at address that si point to
   mov [di]+3, al
   dec si
   dec bx
   dec di
   dec cx
   jnz sum
ret
```

1.7. Based Indexed Relative AM

ការផ្សំគ្នារវាងម៉ូត Based និង Indexed Relative បង្កើតបានជាម៉ូតមួយឈ្មោះ Based Indexed Relative។ នៅក្នុងម៉ូតនេះការប្រើ Register ជា Based register ជាមួយនឹង Indexed Register។

ឧទាហរណ៍ខាងក្រោម ជា Based Indexed Relative Addressing Mode:

```
MOV CL , [BX][DI] + 20 ; ចំលងតម្លៃនៅ DS:BX + DI + 20 ចូលទៅកាន់ក្នុង CL
                       ; PA = DS(SHB) + BX + DI + 20
MOV CH , [BX][SI] +10 ; ចំលងតម្លៃនៅ DS:BX + DI + 10 ចូលទៅកាន់ក្នុង CH
MOV AH , [BP][DI] + 4 ; ចំលងតម្លៃនៅ DS:BP + DI + 4 ចូលទៅកាន់ក្នុង AH
MOV DX , [BP][SI] + 12 ; ចំលងតម្លៃនៅ DS:BP + SI + 12 ចូលទៅកាន់ក្នុង DX
MOV [BX][DI]+12 , AL  ; ចំលងតម្លៃពីក្នុង AL ទៅកាន់អាសយដ្ឋាន DS:BX+DI+12
```

ពាក្យបញ្ជាខាងលើអាចប្រើបាននៅក្នុងកម្មវិធីតែប៉ុណ្ណោះ។ តែចំពោះករណីខាងក្រោមគឺមិនអាចប្រើនៅក្នុងកម្មវិធីបានទេ៖

MOV AH , [BP + SI + 29] ORMOV AH , [SI + BP + 29]

2. **ការរកតម្លៃម៉ាស៊ីនកូដ**(Find Machine Language)

ម៉ាស៊ីនកូដគឺជាសំនុំពាក្យបញ្ជាដែលមានទំរង់ប្រព័ន្ធគោលពីរ និងអាចប្រតិបត្តិដោយស៊ីភីយូដោយផ្ទាល់។ នៅពេលកម្មវិធី(កូដ)មួយត្រូវបានប្រតិបត្តិដោយស៊ីភីយូបាន លុះត្រាកម្មវិធីនោះមានទំរង់ជាម៉ាស៊ីនកូដសិនដោយប្រើកម្មវិធីការបំលែង ឬយើងហៅថា Compile។ នៅក្នុងកម្មវិធី Emu8086 ចុច Mouse លើពាក្យ **compile** ធ្វើការបំលែងកូដម្តងទាំងមូល។ ចំណែកយើងប្រើពាក្យ **emulate** ធ្វើការប្រតិបត្តិកូដម្តងមួយពាក្យបញ្ជា។

2.1. MOV(Move) Instruction

-ករណី **Immediate AM**

MOV reg , imm	1011 ww rrr	data	data if op=1

☞**ចំណាំ:** - R(Register) , PM(Physical Memory) , w (word)rrr(register)

តារាងខាងក្រោមគឺជាលេខកូដប្រើសម្រាប់រកតម្លៃ w និង rrr ធ្វើប្រមាណវិធី R/M:

Mod = 11			Mod = 00
8-bit(w=0)	**16-bit(w=1)**	**Mmm**	**EA**
AL	AX	**000**	**[SP] + [SI]**
CL	CX	**001**	**[SP] + [DI]**
DL	DX	**010**	**[BP] + [SI]**
BL	BX	**011**	**[BP] + [DI]**
AH	SP	**100**	**[SI]**
CH	BP	**101**	**[DI]**
DH	SI	**110**	**[OA]**
BH	DI	**111**	**[BX]**

☺**ឧទាហរណ៍:** ចូរសរសេរ Statement និងបង្ហាញពី Machine Code ដើម្បីផ្ទៀរ:

1. Immediate :57H ទៅ AL Register។

   MOV AL, 57H = ? ? (រកតម្លៃ MC)

   MOV AL , 57h = 1011 ww rrr data ជារូបមន្តរកតម្លៃម៉ាស៊ីនកូដ ដូច្នេះ

   ww = 0 ដោយប្រើឈ្មោះ AL Register ដែលមានទំហំ 8 bits និងចំពោះតម្លៃ

   rrr = AL = 000 សូមមើលតារាងកូដខាងលើ និង data =57h

   MOV AL , 57h = **1011 0000 57H**

   MOV AL, 57H = B057H (2 bytes)

2. Immediate:123AH ទៅ DX Register។

   MOV DX , 123AH = ? ? (រកតម្លៃ MC)

   MOV DX , 123Ah = 1011 ww rrr l-data h-data

   rrr = DX =010 ជាតម្លៃកូដនៃអក្សរ DX

   w = 1 ប្រើទំហំ Register 16 bits

   l_data = 3AH , h_data = 12H

   MOV DX , 123AH = 1011 1010 3A 12H

   MOV DX , 123AH = BA3A12H (3 bytes)

b. ករណីប្រើម៉ូត **Register, Direct, Indirect** AM, Based Relative AM, Indexed Relative AM និង Based Indexed Relative AM ប្រើ

100010 dw	oo rrr mmm	disp

- d = 1 , ការផ្ទេរតម្លៃចូលក្នុង Register (Transfer data to register) ក្រៅពីនឹងd=0។
- oo (modifier bits) គឺជាប់ទាក់ទងប្រើ mod បើទីតាំងទាំងពីរប្រើ Register នោះ mod= 11 , ប្រើ memory វិញនោះ mod =00 ,...
- rrr is register ; rrr មើលការប្រើទំហំ Register បើប្រើ Register 8 bits នោះ rrr=0 តែបើការប្រើ Register ១៦ ប៊ីតវិញនោះ rrr =1
- mmm is register or memory ករណីនេះជាប់ទាក់ទងនឹងការប្រើឈ្មោះ Register ឬ memory

តារាងខាងក្រោមគឺជាកូដតាងឱ្យ Statement មួយជា Based Indexed Relative AM ឬ ប្រើជាមួយ Segment Register:

mod 01		mod 11
Function	Mmm	Segment Register
[SP] + [SI] + disp8	000	ES
[SP] + [DI] + disp8	001	CS
[BP] + [SI] + dips8	010	SS
[BP] + [DI] + disp8	011	DS
[SI] + disp8	100	FS
[DI] + disp8	101	GS
[BP] + disp8	110	
[BX] + disp8	111	

ឧទាហរណ៍: ចូរប្រើរូបមន្តខាងលើដើម្បីរកតម្លៃម៉ាស៊ីនកូដតាមតារាងខាងក្រោមខាងក្រោយ ដោយគេបានដោះស្រាយរួចហើយ៖

Format	Example	Machine Code
	MOV DL , BL	8A D3
MOV reg , reg	MOV CX , DX	8B CA
	MOV BH , CH	8A FD
	MOV DATA , DL	88 16 02 01
MOV mem , reg	MOV NUMB, DH	88 36 06 01
	MOV [DI] +10, CL	88 4D 0A
	MOV [BX] , SI	89 37
	MOV CX, COUNT	8B 0E 03 01
MOV reg , mem	MOV CL , [BX] +10	8A 4F 0A
	MOV BL , TEMP[DI]	8A 9D 05 01
	MOV DX , [BX][SI]+9	8B 50 09

- Address របស់ Data គឺ 0102H
- Address របស់ NUMB គឺ 0106H
- Address របស់ COUNT គឺ 0103H
- Address របស់ TEMP គឺ 0105H

របៀបបន្តបន្ទាប់ខាងក្រោមគឺជាវិធីរកតម្លៃម៉ាស៊ីនកូដនៃពាក្យបញ្ជា :Mov DL, BL

Mov DL, BL = **100010 dw oo rrr mmm disp = 1000 1010 1101 0011b**

= 8AD3h

- d=1 ព្រោះផ្ទេរតម្លៃចូលក្នុង Register និង w = 0 ដោយនៅទីតាំងទាំងពីរប្រើ ៨ ប៊ីត Register
- oo =11 ដោយទីតាំងទាំងពីរប្រើឈ្មោះ Register គឺ DL និង AL
- rrr = dl = 010 យកតម្លៃលេខកូដ Register នៅទីតាំង Destination
- mmm = bl =011 យកតម្លៃលេខកូដ Register នៅទីតាំង source
- disp គឺអត់មានប្រើនៅក្នុងពាក្យបញ្ជាខាងលើ

ចូរសាកល្បងរកតម្លៃម៉ាស៊ីនកូដនៃពាក្យបញ្ជា MOV CX , DX = 8BCAh

= 10001011 11001010 b

c. ករណីប្រើ AX និង AL :

101000dw	disp

✡ចំណាំ: d = 0 , ពេលផ្ទេរតម្លៃចូល AX , ឬ AL register

☺**ឧទាហរណ៍:** ចូររកតម្លៃ MC តាមតារាងខាងក្រោម៖

Format	Example	Machine Code
	MOV DATA, AL	*A2 02 01*
MOV mem , acc	MOV NUM , AX	*A3 03 01*
	MOV AL, DATA	*A0 02 01*
MOV acc , mmm	MOV AX , NUM	*A1 03 01*

d. ករណីប្រើ Segment Register

1000 11 d0 oo sss mmm disp

☺**ឧទាហរណ៍:** ចូររកតម្លៃ MC តាមតារាងខាងក្រោម៖

Format	Example	Machine Code
MOV reg , reg	MOV SS , AX	8E D0H
	MOV DS , DX	8E C2H

2.2. ADD(Addition) Instruction

- ករណីប្រើជា Register ឬ Memory AM

000000 dw	oo rrr mmm

☺**ឧទាហរណ៍:** ចូររកតម្លៃ MC តាមតារាងខាងក្រោម៖

Format	Example	Machine Code
	ADD DL , BL	02 D3
ADD reg , reg	ADD CX , DX	03 CA
	ADD BX , SI	03 DE
	ADD DATA , DL	00 16 02 01
ADD mem , reg	ADD NUMB, DH	00 36 03 01
	ADD [DI] +10, CL	00 4D 0A
	ADD CX, COUNT	03 0E 04 01
ADD reg , mem	ADD CL , [BX] +10]	02 4F 0A
	MOV DX , [BX][SI]+9	8B 50 09

- Address របស់ Data គឺ 0102H
- Address របស់ NUMB គឺ 0103H
- Address របស់ COUNT គឺ 0104H

- ករណីប្រើជា Immediate AM ឬ Accumulator register

100000 sw	oo 000 mmm	data

Format	Example	Machine Code
	ADD DATA , 34H	80 06 02 01 34
ADD mem , imm	ADD LIST , "A"	80 06 03 01 41
	ADD [DI], 2120H	81 05 02
	ADD CX , 3	83 C1 03
ADD reg , imm	ADD DL, 60H	80 C2 60
	ADD DX , 1200H	81C2 0012H
	ADD AX, 1234H	05 34 12
ADD acc , imm	ADD AL , 1AH	04 1A
	ADD AH, 44H	80 C4 44

2.3. DEC(Decrement) Instruction

☹**Syntax**: Statement Machine Code

DEC Reg.

1111111w

2.4. INC(Increment) Instruction

☹**Syntax**: Statement Machine Code

DEC Reg.

1111111op	000 ooo rrr

2.5. *JNZ(Jump Not Zero) Instruction*

☹**Syntax**: Statement Machine Code

JNZ Target Addr. | 0111010 |

3. **កម្មវិធី Debug**

វាជាកម្មវិធីមួយមានស្រាប់នៅក្នុងប្រព័ន្ធប្រតិបត្តិការ MS–DOS ដោយស្វ័យប្រវត្តិ។ កម្មវិធីនេះប្រើសម្រាប់មើលឈ្មោះ Register របស់ 8086 /8088 រួមនឹងធាតុរបស់វានៅពេលដំណើរការកម្មវិធីមួយ។ វាក៏អាចឱ្យអ្នកសរសេរកម្មវិធីសរសេរកូដមួយដោយប្រើ Register និងពាក្យបញ្ជារបស់ស៊ីភីយូដោយផ្ទាល់ដើម្បីបានជាឃ្លាបញ្ជាផ្សេងៗ ដែលបង្កើតបានជាកូដ(code)សម្រាប់ប្រតិបត្តិនៅក្នុងឈីបនេះ។ កម្មវិធីនេះក៏អាចឱ្យអ្នកសរសេរកម្មវិធីស្គាល់ពីស៊ីភីយូប្រតិបត្តិកូដមួយក្នុងទំរង់ជាម៉ាស៊ីនកូដ និងស្គាល់ពីការផ្ទុក Opcode, Operand ទៅក្នុងទីតាំង Memory មួយដោយប្រើលេខអាសយដ្ឋានដោយផ្ទាល់ និងដឹងពីការធ្វើប្រមាណវិធីជាក់ស្តែងក្នុងខ្លួនវា។ ជាពិសេសប្រើនៅក្នុងការតេស្ត ៨០៨៦ Traner Kit សម្រាប់ដំណើរការ ជាមួយនឹងឧបករណ៍ជាក់ស្តែងដូចជា LED, KEYPAD, LCD, MOTOR ឬ Sensor ជាដើម។

3.1. **ចូលក្នុងកម្មវិធី Debug**

DOSBOX	Emu8086
1.ចូលកម្មវិធី DOSBox	1. ចូលកម្មវិធី Emu8086
2.Z:\ >**mount c c:** ចុច Enter	2. ចុច Mouse លើ emulate
3.Z:\>**C:** ↵	3. ចុច Mouse លើ debug

3.2. **បញ្ជា q**(Quit command)

"q" មានតួនាទីប្រើសម្រាប់ចាកចេញពីកម្មវិធី Debug។

- q ↵ C:\> exit ⟶ Windows

3.3. **បញ្ជា R** (Register command)

R ជាបញ្ជាមួយប្រើសម្រាប់មើលឈ្មោះ Registers ទាំងអស់របស់ 8086 /8088 និង ធាតុផ្សេងៗ ដែលមាននៅ ក្នុង Register នោះ។

☞ ចូរវាយពាក្យ R និងបង្ហាញអំពីសំនុំ Register ទាំងអស់របស់ស៊ីភីយូ ៨០៨៦

- R .↵

AX = 0000 BX = 0000 CX = 0000 DX = 0000 SP = FFEE BP = 0000 DI = 0000 SI = 0000 DS = 1900 ES = 1900 SS = 1900 CS = 1900 IP=0100 NV UP EI PL NZ NA PO NC : ដោយ NV (No Overflow flag), UP, EI (Enable), PL (Plus),NZ (No Zero)

3.4. **បញ្ជា A** (Assemble Command):

'A' គឺជាបញ្ជាមួយប្រើសម្រាប់បញ្ចូលកូដជាភាសា Assembly ដើម្បីដំណើរការដោយផ្ទាល់ ក្នុងស៊ីភីយូ ៨០៨៦ ។

☞ Practice

-A ↵

1900:0100 ⟶ កម្មវិធី Debug ចូលដោយស្វ័យប្រវត្តិទៅអាសយដ្ឋានឈ្មោះ Logical Address ដូចខាងក្រោមនេះ៖

1900:0100 MOV AX , 1234

1900:0103 MOV BX , ABCD

1900:0106 ADD AX , BX

1900:0109 INT 3 រួចចុច Enter ពីរដងដើម្បីចាកចេញការសរសេរនេះទៅសញ្ញា - **ទើបយើងអាចប្រើបញ្ជា Run .**

3.5. G (Go Command)

'G' ជាពាក្យបញ្ជា(Instruction)ប្រើសម្រាប់ដំណើរការកម្មវិធីទាំងមូលដើម្បីឱ្យចេញលទ្ធផលនៅក្នុង ៨០៨៦ Register។

☞ **Practice**

- A ↵

1900 : 0100 MOV AL , 12

1900 : 0102 ADD AL , 40

1900 : 0104 INT 3

1900 : 0105

- G ↵

AX = 0052 BX = 0000 CX = 00000 DX = 0000 , ...

3.6. បញ្ជា U(Unassemble Command):

'U' ជាពាក្យបញ្ជាមួយប្រើសម្រាប់មើលនូវពាក្យបញ្ជាស៊ីកីយូនិមួយៗដែលសមមូលនឹងតម្លៃនៃម៉ាស៊ីនកូដ(Machine Code)។

☺Syntax: **U Start Addr. End Addr.**

☞ Practice

-U 100 0104 ↵

1900: 0100 B012 MOV AL,12

1900: 0104 0430 ADD AL, 30

1900: 0105 CC INT 3

Machine Code Assemble Pro

3.7. D(Dump): គឺជាបញ្ជាមួយរបស់កម្មវិធី Debug ដើម្បីបង្ហាញទិន្នន័យពីក្នុង Memory ។

☺**Syntax: D Start Addr. End Addr.**

☞Practice:

-E DS:1200 50 60 70 30 4F 9C

-D DS:**1200 1205** សម្រាប់មើលទិន្នន័យពី Address:1200H ដល់ 1205H ដោយទិន្នន័យនៅ៖

DS:1200 = 50H	DS:1203 = 60H
DS:1201 = 60H	DS:1204 = 50H
DS:1202 = 50H	DS:1205= 59

3.8. E(Enter): គឺជាបញ្ជាប្រើសម្រាប់ធ្វើការបញ្ចូលតម្លៃពី Keyboard នៅពេលដំណើរការកូដ។

☺**Syntax:** E OA Data List

✍**ឧទាហរណ៍:** ប្រើកម្មវិធី Debug ដើម្បីធ្វើប្រមាណវិធីបូក Hex : 1F+12+15+45

a. ការប្រើ AL, CL, BL, AH ផ្ទុកចំនួនទី 1,2,3,4 រៀងគ្នានិងលទ្ធផលផ្ទុកក្នុង AL register

b. ការប្រើ Loop ដើម្បីដំណើរការប្រមាណវិធីខាងលើ

a. ការប្រើ AL, CL, BL, AH Register

```
1900: 0100  MOV AL, 1F     ; ផ្ទេរតម្លៃ 1F ចូល AL ដើម្បីឱ្យបាន AL =1F
1900: 0102  MOV CL,12        ; ផ្ទេរតម្លៃ 12 ចូល CL ដើម្បីឱ្យបាន CL =12
1900: 0104  MOV BL,15        ; ផ្ទេរតម្លៃ 15 ចូល BL ដើម្បីឱ្យបាន BL =1F
1900: 0106  MOV AH,45        ; ផ្ទេរតម្លៃ  45 ចូល AH ដើម្បីឱ្យបាន AH =45
1900: 0108  ADD AL, CL     ; AL = AL + BL
1900: 010A ADD BL, AH        ; BL = BL +AH
1900: 010C  ADD  AL, BL      ; AL = AL + BL
1900:10E INT 3
1900:10F
-G ←┘
AX = 458B
```

b. ការប្រើ Loop

☞**របៀបទី១**

```
1900:100   MOV CX , 0004 ; CX=0004
1900:103   MOV BX , 0200 ;Bx=0200
1900:106   MOV AL,0  ;AL= 0
1900:108   ADD AL,[BX]        ;AL = AL + តម្លៃលេខនៅអាសយដ្ឋានដែលBX ចង្អុលទៅ
1900:10A  INC BX              ;Bx = Bx + 1
1900:10B  DEC  CX             ;CX= CX -1
1900:10C  JNZ 108
1900:10E  INT  3
1900:10F
-   e 200 1F 12  15  45   ←┘
-   d 200  203  ←┘
1900 : 200 1F  12  15  45
```

- g ←┘ ក្រោយពេលវាយបញ្ជានេះបានលទ្ធផលគឺ AX = 458B

តារាងខាងក្រោមបង្ហាញពីទំរង់ដំណើរការកម្មវិធីទាំងមូលចាប់ពីរង្វិលជុំដំបូងរហូតដល់ចុងក្រោយ៖

Loop	CX	2F	BX	AL	+	[BX]	AL
1	4	0	0200	0	+	1F	1F
	3	0	0201				
2	2	0	0202	1F	+	12	31
3	1	0	0203	31	+	15	46
4	0	1	0204	46	+	45	8B

☞ **របៀបទី២**

1900:100 MOV CX , 0003 ; កំណត់តម្លៃរង្វិលជុំ 3 ដងអោយ CX

1900:103 MOV BX , 0200 ; កំណត់តម្លៃលេខ Address 0200 ដងអោយ BX

1900:106 MOV AL , 1F ; កំណត់តម្លៃរង្វិលជុំ 1F ដងអោយ AL

1900:108 ADD AL , [BX] ; យកតម្លៃនៅលេខអាសយដ្ឋានដែលBX ឆ្ពោះទៅកាន់ចូល AL

1900:10A INC BX ; បន្ថែមលេខ ១ ក្នុង Bx ដោយ Bx = Bx + 1

1900:10B DEC CX ;បន្ថយលេខ ១ ក្នុង Cx ដោយ Bx = Bx + 1

1900:10C JNZ 108

1900:10E INT 3

1900:10F

- e 200 1F 12 15 45 ↵
- g រួចចុច Enter ដើម្បីបានលទ្ធផលគឺ Ax = 458B

3.9. T (Trace): គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់ឱ្យកម្មវិធីប្រតិបត្តិម្តងមួយStatement។

-A ↵

CS: 100 MOV AL , 9CH; AL = 9Ch

MOV DH , 64H; DH = 64h

ADD AL , DH ; AL = AL + DH

INT 3; បញ្ចប់ការប្រតិបត្តិពាក្យបញ្ជា

-T ↵ **បន្ទាប់មកបានលទ្ធផលដូចខាងក្រោម៖**

AX = 009C BX = 0000 CX = 0000 DX = 0000 SP = FFEE BP = 0000 DI = 0000 SI = 0000 DS = 1900 ES = 1900 SS = 1900 CS = 1900 IP=0100 NV UP DI PL NZ NA PO NC

-T ↵

AX = 0000 BX = 0000 CX = 0000 DX = 6400 SP = FFEE BP = 0000 DI = 0000 SI = 0000 DS = 1900 ES = 1900 SS = 1900 CS = 1900 IP=0100 NV UP DI PL NZ NA PO NC

-T ↵

AX = 009C BX = 0000 CX = 0000 DX = 0000 SP = FFEE BP = 0000 DI = 0000 SI = 0000 DS = 1900 ES = 1900 SS = 1900 CS = 1900 IP=0100 NV UP DI PL ZR AC PE CY

3.10. Examining / Altering The Flag Register in Debug

-R F គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់មើលសញ្ញាសំគាល់ទាំងអស់នៅក្នុងកម្មវិធី Debug។

ការអនុវត្តន៍៖

-R F enter

NV UP DI PL NZ NA PO NC

Table : Codes for the Flag Register

Flag	Code when Set(=1)	Code when Set(0)
OF Overflow flag	OV (overflow)	NV (no overflow)
DF Direction flag	DN (down)	UP (up)
IF Interrupt flag	EI (enable interrupt)	DI (Disable interrupt)
SF Sing Flag	NG (negative)	PL (plus)
ZF Zero flag	ZR (zero)	NZ (no zero)
AF auxiliary flag	AC (Auxiliary carry)	NA (no Auxiliary carry)
PF Parity flag	PE (parity even)	PO (parity odd)
CF Cary flag	CY (carry)	NC (no carry)

W (Write)

W គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់រក្សាទុកទិន្នន័យនៅលើ Hard Disk

W <start Address> <drive number><starting sector><sector>

3.11. L (Load)

L គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់ហៅទិន្នន័យនៅលើឌីសទៅដាក់នៅក្នុងផ្នែ Memory។

L <start Address> <drive number><starting sector><sector>

3.12. N (Name)

N គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់ហៅបញ្ចូលឈ្មោះ File ទៅក្នុងកម្មវិធី Debug។

-N <filename>

-L

ឧទាហរណ៍: ការបញ្ជូនកម្មវិធី Assembly ទៅក្នុងកម្មវិធីកែតំរូ

(a). ការបញ្ជូនឈ្មោះនិងពាក្យបញ្ជា

-N A:\PROGRAMS\Prog1.exe

-L

(b). Loading on entering Debug program

A:>Debug A:\PROGRAMS\Prog1.exe

ឧទាហរណ៍: ការរក្សាទុកនិងបញ្ជូនកូដ

1900:100 MOV CX , 0005 ;កំណត់តម្លៃរង្វិលជុំ5ដងអោយ CX

1900:103 MOV BX , 0200 ;កំណត់តម្លៃលេខអាសយដ្ឋាន ២០០អោយ BX

1900:106 MOV AL , 0 ;កំណត់តម្លៃលេខ 0 អោយ AL

1900:108 ADD AL , [BX] ; យកតម្លៃនៅលេខអាសយដ្ឋានដែលBX ឆ្ពោះទៅកាន់ចូល AL

1900:10A INC BX ; បន្ថែមលេខ ១ ក្នុង Bx ដោយ Bx = Bx + 1

1900:10B DEC CX; បន្ថយលេខ ១ ក្នុង Cx ដោយ Bx = Bx + 1

1900:10C JNZ 108

1900:10E INT 3

1900:10F

- **R CX ពេលប្រើពាក្យបញ្ជានេះរួចយើងបានលទ្ធផលគឺ CX 0000**
- **R រួចចុច Enter Key បានលទ្ធផលដូចខាងក្រោម៖**

  AX = 009C BX = 0000 **CX = 0012** DX = 0000 SP = FFEE BP = 0000 DI = 0000 SI = 0000 DS = 1900 ES = 1900 SS = 1900 CS = 1900 IP=0100 NV UP DI PL NZ NA PO NC
- **N** C:\exam.com
- **W**

  Writing 0012 bytes
- **Q**

  C:>Debug
- **N C:>exam.com**
- **L**
- **R**

  AX = 009C BX = 0000 CX = 0000 DX = 0000 SP = FFEE BP = 0000 DI = 0000 SI = 0000 DS = 1900 ES = 1900 SS = 1900 CS = 1900 IP=0100 NV UP DI PL NZ NA PO NC
- **U CS:100 10E**

  1900:100 B90500 MOV CX , 0005 ;CX=0005

  1900:103 BB0002 MOV BX , 0200 ;BX=0200

  1900:106 B000 MOV AL , 0 ; AL=0

  1900:108 0207 ADD AL , [BX] ; Al = AL + data at address that Bx point to

  1900:10A 43 INC BX ; Bx = Bx + 1

  1900:10B 49 DEC CX ; Cx = Cx + 1

1900:10C 75FA JNZ 108

1900:10E A20502 INT 3

1900:10F CC

- A

1900: 0100 MOV AX, C1AD

1900: 0100 MOV AX , 6170

1900: 0100 ADD AX , BX

1900: 0100 INT 3

G ↵

CY, PE , NZ , NA

- CY (Carry)
- PE (Parity Even)
- NZ (No Zero)
- NA (No Axilary)

MOV AX, 1250 ; Ax = 1250

MOV CX, 7115 ; CX = 7115

ADD AX, CX ; Ax = Ax +BX

ឧទាហរណ៍: ចូរបង្ហាញលទ្ធផលនិងសញ្ញាសំគាល់តាមកម្មវិធីខាងក្រោម៖

១. ប្រើក្នុងកម្មវិធី Debug

២. ប្រើក្នុងកម្មវិធី X86Asm-Program

ដំណោះស្រាយ

១.Debug Program	២.ក្នុង X86Asm-Program
MOV AX,1250	MOV AX,1250H
MOV CX, 7115	MOV CX,7115H
ADD AX , CX	ADD AX , CX
INT 3	ret ការប្រើកម្មវិធីទាំងពីរខុសគ្នាត្រង់ចំនុចនេះ

4. ការធ្វើការងារខាងក្នុងនៃ 8086(Internal Working of 8086 /8088)

ចូរនិយាយពីស៊ីកីយូ 8086 (8088)ធ្វើការប្រតិបត្តិម៉ាស៊ីនកូដនៃកម្មវិធីដូចខាងក្រោម៖

Assembly Program		Machine Code	Emu8086 Program
MOV AL , 12h	=	B012H	MOV AL, 12H
ADD AL, 30h	=	0403H	ADD AL, 30H

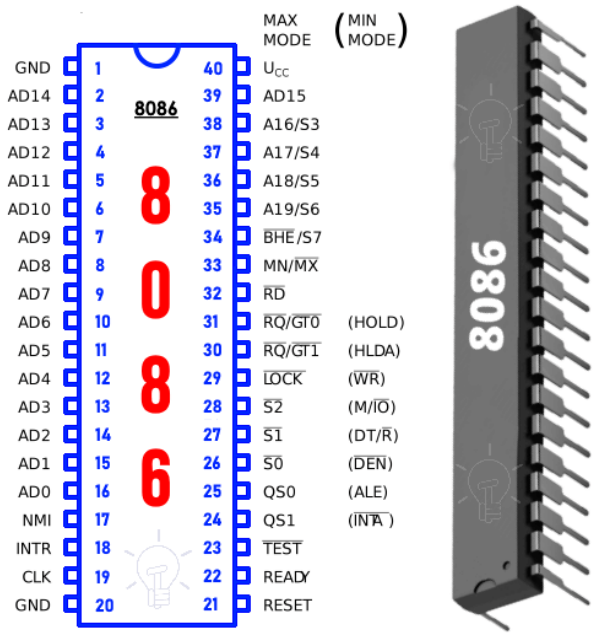
ស៊ីកីយូ8086 ឬ 8088 ដំណើរការម៉ាស៊ីនកូដនៃកម្មវិធីខាងលើតាមដំហានដូចខាងក្រោម៖

១. ជាដំបូង CPU ត្រូវរកអាសយដ្ឋានជា PA(Physical Address)របស់**B0**ដោយយកតម្លៃចាប់ផ្តើមនៅក្នុង **CS** និង **OA** នៅក្នុង **IP** បូកបញ្ចូលតម្លៃទាំង២។ ដើម្បីធ្វើបានដូចនេះនោះត្រូវយកលេខអាសយដ្ឋានក្នុង **CS** ធ្វើប្រមាណវិធី Shift bit Left 4 ដង រួចបូកតម្លៃ OA នៅក្នុង IP បង្កើតបានជា PA របស់ B0។ បន្ទាប់មកបញ្ជូនលេខអាសយដ្ឋាននេះតាមរយៈ Address Bus ទៅកាន់ Memory ដើម្បីឱ្យ Memory រកតម្លៃកូដ Instrution នៅ Address ដែលបានបញ្ជូនមក និងបន្ទាប់មក CU បញ្ជូន Read Signal តាមរយៈ Control Bus ដើម្បីបំលាស់ទីតម្លៃកូដ **B0** តាមរយៈ Data Bus ទៅផ្ទុកនៅក្នុង Instrution Queue និងបន្ទាប់មកបញ្ចូលកូដនេះទៅឱ្យ Decoder នៃCU ធ្វើការបកប្រែ។ តាមរយៈ:កូដនេះស៊ីភីយូអាចយកតម្លៃ 12h នៅទីតាំង OA(Offset Address)បន្ទាប់ដើម្បីយកតម្លៃនេះពី Memory ទៅផ្ទុកនៅ AL Register។ បន្ទាប់ពីប្រតិបត្តិ Code នេះចប់ IP=102h ដែលជា OFFSET ADDRESS នៃ Code Intruction បន្ទាប់ទៀត។

២. អនុវត្តន៍ដូចជំហានទី១, CPU អាចយកកូដ 04 ទៅបកប្រែ បន្តទៀត និងតាមរយៈ:កូដ នេះ CPU អាចយកតម្លៃមួយនៅ OA=103 និងតម្លៃមួយទៀតនៅក្នុង AL Register ទៅឱ្យ ALU ធ្វើប្រមាណវិធីបូកនិងលទ្ធផលក្រោយពេលគណនាផ្ទុកនៅក្នុង AL Register វិញ Flags ទាំងអស់ផ្ទុកនៅក្នុង Flag Register។ បន្ទាប់ពីដំណើរការ Code 04 ចប់ IP=104។

៣. អនុវត្តន៍ដូចជំហានទី១ ដដែល CPU អាចប្រែ Code cc បន្ទាប់ទៀតនិងតាមរយៈ Code នេះធ្វើឱ្យ CPU អាចបញ្ឈប់ដំណើរការកម្មវិធីទាំងមូល។

5. **មុខងារនៃជើង Pin 8086**



ជើង Pin និមួយៗរបស់ ៨០៨៦ គឺប្រើប្រាស់សម្រាប់តភ្ជាប់ទៅឧបករណ៍ផ្សេងៗទៀតមានដូចជា RAM, ROM, Hard Disk, I/O, PORT,...សម្រាប់បញ្ជូនទិន្នន័យ និងពាក្យបញ្ជារវាងឧបករណ៍ទាំងនេះ ខាងក្រោមនេះគឺជាមុខងារជើង Pin របស់ 8086 Microprocessor :

- V_{cc}: ជាជើង Pin សម្រាប់ផ្តល់ចរន្ត +5V ទៅឱ្យស៊ីភីយូ និង GND គឺជើង Pin ផ្តល់ធ្វើការតភ្ជាប់ទៅប៉ូលដក។ គួរកត់សំគាល់ដែរថា 8086/8088 មានជើង GNDPin ចំនួនពីរ

- CLK: ផ្តល់គ្រឹះ Timing Signal ទៅឱ្យស៊ីកីយូ។ Clock signal អាចមានជា Duty cycle មួយនៃ 33% ដោយផ្តល់ Proper internal timing ចំពោះ 8086/8088។
- AD_7-AD_0: គឺជាផ្លូវសម្រាប់ធ្វើការនាំលេខអាសយដ្ឋាន ឬ ទិន្នន័យ(Address/Data bus)។ វាផ្សំឡើងពី Bus ច្រើនចូលគ្នា និងមានទំហំ ៨ ប៊ីត នៃ Memory address ឬ I/O port។ក្នុងករណីប្រើ Bus ជាលេខ Adddress លុះត្រាតែតម្លៃ ALE មានតម្លៃតក្កស្មើនឹង ១។ តែបើ ALE មានតម្លៃតក្កវិទ្យាស្មើនឹង 0 វិញនោះ AD_7-AD_0 ប្រជា Data Bus វិញ។
- MN/$\overline{MX}$(Minimum/maximum mode pin): Pin នេះប្រើសម្រាប់ធ្វើការជ្រើសរើសម៉ូត Minimum ឬ Maximum ចំពោះប្រមាណវិធីក្នុង Microprocessor។ បើ Minimum mode មានការជ្រើសរើសនោះ MN/MX pin អាចភ្ជាប់ដោយផ្ទាល់ទៅជើង Pin 5.0V។
- $\overline{BHE}$ / S_7: Bus high enable pin គឺប្រើសម្រាប់ 8086 ធ្វើឱ្យមាន Enable Most- significant bit data bus (D_{15}-D_8) ក្នុងពេល Read ឬ Write ក្នុងករណីបើ S7 ជានិច្ចកាលតម្លៃតក្កវិទ្យាស្មើ 1
- Minimum Mode Pins: Minimum Mode Pins នៃ 8086/8088 ត្រូវបានភ្ជាប់ដោយ MN/ $\overline{MN}$ pin ដោយផ្ទាល់ពី +ទៅ-5.0 V ដោយឆ្លងកាត់ Pull up resister។
- IO/M or M/IO: សម្រាប់ជ្រើសរើស Memory Or I/O។ Pin នេះបង្ហាញនូវអ្វីដែល Microprocessor address bus មាន Memory address or I/O port address។
- WR: Pinនេះប្រើញឹកញាប់ដែលបង្ហាញពី Output data ទៅ Memory or I/O devices។ ក្នុងរយៈពេលដែល $\overline{WR}$ មានតក្ក 0, data bus មានតម្លៃសម្រាប់ Memory or I/O devices។ Pin នេះជា float ទៅ high – impedance រយៈមាន hold acknowledge។
- $\overline{INTA}$: Interrupt acknowledge signal គឺមានការឆ្លើយតបទៅ INIR input pin a។ Pin នេះជាធម្មតាប្រើសម្រាប់ភ្ជាប់ទៅ Gate របស់ Interrupt vector number ដែលមាននៅលើ Data bus សម្រាប់ឆ្លើយតបទៅ Interrupt request។
- ALE (Address latch enable): ជើងPin បង្ហាញអំពី address/data bus មានពត៌មានស្តីអំពីលេខអាសយដ្ឋាន។ ហើយលេខអាសយដ្ឋាននេះអាចជា Memory address ឬជាលេខជា I/O port។ គួរកត់សំគាល់ដែរថា ALE signal មិនមាន Float ក្នុងពេលមាន Hold acknowledge។
- DTR / $\overline{R}$: Data transmit / receive signal បង្ហាញពី Microprocessor data bus បញ្ជូន (DT/R = 1) or ទទួល ($\overline{DT}$/R=0) data។ នេះប្រើសម្រាប់ធ្វើឱ្យ មាន internal data bus buffer។
- DEN: Data bus enable មានសកម្មភាពលើ External data bus buffers។
- HOLD: Hold input សំនើរដោយផ្ទាល់ទៅ Direct memory access (DMA) បើ Hold signal មាន Logic 1, Microprocessor ឈប់ប្រតិបត្តិ Software និង ផ្លាស់ Address, Data, និង Control bus នៅត្រង់ High –impedance state។ បើ Hold signal មាន Logic 0, Microprocessor ឈប់ប្រតិបត្តិ Software ធម្មតា។
- HLDA: Hold Acknowledge បង្ហាញពី 8086 microprocessor មានបញ្ជូល Hold state។

- $\overline{SSO}$: $\overline{SSO}$ status line គឺមានតម្លៃស្មើនឹង S_0 pin នៅក្នុង Maximum mode operation នៃ Microprocessor។ Signal នេះគឺផ្សំជាមួយ IO/R និង DT/R សម្រាប់បកប្រែ Function នៃ Current bus cycle។
- Maximum Mode Pins: ធ្វើការសំរេចទៅតាមលំដាប់លំដោយក្នុង Maximum mode សម្រាប់ប្រើជាមួយ External Microprocessor, ភ្ជាប់ទៅ MN/MN Pin to ground
- $\overline{S_2}$, $\overline{S_1}$, and $\overline{S_0}$: Status bit បង្ហាញពីមុខងារនៃ Current bus cycle។ Signal នេះគឺបកប្រែជាធម្មតាដោយ 8288 bus controller។
- $\overline{RQ}$ / $\overline{GTI}$ and $\overline{RQ/GT0}$: Request/grant pins សំណើរដោយផ្ទាល់ទៅ DMA ក្នុងពេល Maximum mode មានប្រមាណវិធី។ Line នេះទាំងពីរជា bi-directional និង ប្រើសម្រាប់ជាសំណើរ និង Grant DMA operation។
- $\overline{LOCK}$: Look output គឺប្រើសម្រាប់ Lock peripherals off នៃ System។ Pin នេះមានសកម្មភាពដោយប្រើ Lock : Prefix នៅលើ Instruction។
- QS_1 and QS_0: Queue status bits បង្ហាញពី Status នៃ Internal Instruction queue, Pin នេះគឺផ្តល់សម្រាប់ដំណើរការដោយ numeric coprocessor (8087)។
- A_{15}-A_0: គឺជា address bus ផ្តល់នូវ Upper-half memory address bits ដែលតំណាងឱ្យ bus cycle ទាំងស្រុង។ Address នេះភ្ជាប់ទៅ High-impedance ក្នុងពេលមាន Hold acknowledge
- AD_{15}-AD_8: គឺជាខ្សែ Addresss/Data bus ផ្សំឡើង Address data bus ច្រើនចូលគ្នានៅលើ 8086។ ខ្សែនេះមាន address bits A_{15}-A_8 នៅពេល ALE មានតម្លៃ Logic 1 Pin ហើយ Data bus ភ្ជាប់ទៅ D_{15}-D_8 នោះ Pin ចូលទៅនេះជា High-impedance state ក្នុងពេល Hold acknowledge កើតឡើង។ គ្រប់ពេល Microprocessor បញ្ជូនលេខ Address ចេញ ALE មាន Logic 1 (Set High) បង្ហាញពីពត៌មាននៅលើ Pin AD_{15}-AD_0 គឺជា Address (A_{15}-A_0)។ ពេល Data ត្រូវបញ្ជូន ឬ បញ្ចូល ALE មាន Logic 0 (Set Low) បង្ហាញពីពត៌មាននៅលើ Pin AD_{15}-AD_0 គឺជា Data Bus។
- A_{19}/S_6-A_{16}/ S_3: Address /data bus គឺមានច្រើនសម្រាប់ផ្តល់ Address signals A_{19}-A_{16} និងមាន Status bits S_6-S_3។ Pin នេះផ្តល់ជា High-impedance state ក្នុងពេលដំណើរការ Hold acknowledge។ Status bits S_6 ជានិច្ចមានទំរង់ Logic 0, bit S_5 បង្ហាញ Condition នៃ IF flags bits និង S_4 និង S_3 បង្ហាញពី Segment មួយកំពុងដំនើរការក្នុងពេលកើតមាន Bus cycle។
- $\overline{RD}$: រាល់ពេល Read signal មានតក្កស្មើនឹង 0 នោះ data bus គឺទទួលយក Data ពី Memory ឬឧបករណ៍ I/O។ Pin នេះជា Float ទៅ high-impedance ក្នុងពេលកើតមាន Hold acknowledge។
- READY: Input នេះគឺធ្វើការត្រួតពិនិត្យបញ្ចូល wait states ទៅក្នុង timing នៃ Microprocessor។ បើសិន READY pin គឺផ្លាស់ទៅជា Logic 0 level, នោះធ្វើឱ្យ Microprocessor ចូលទៅក្នុង Waits state និង មាន idle។ បើសិន READY pin គឺផ្លាស់ទៅជា Logic 1 level។ វាគ្មានអនុវត្តនៅប្រមាណវិធីនៃ Microprocessor។

- INTR: I/nterrupts Request គឺប្រើសម្រាប់ Request Hardware interrupt។ បើ INTR មាន Held high ពេល IF=1។
- $\overline{\text{TEST}}$: TES pint គឺមាន Input ដែលធ្វើការ Test ដោយ WAIT instruction។ បើ TEST ជា Logic 0, នោះ Wait instruction មានមុខងារជា NOP។ បើ TEST ជា Logic 1, នោះ Wait instruction រង់ចាំសម្រាប់ TEST ក្លាយទៅជា Logic 0។ Pin នេះគឺភ្ជាប់ទៅ 8087 numeric coprocessor។
- NMI: Non-Makable Interrupt input មានមុខងារប្រហាក់ប្រហែល INTR លើកលែងតែ NMI interrupt មិនមានការត្រួតពិនិត្យមើល IF flags bit មាន Logic 1។
- RESET: Reset input បណ្តាលឱ្យ Microprocessor ធ្វើឱ្យមានធ្វើការឡើងវិញដោយខ្លួនឯងក្នុងករណីបើមាន Pin នេះមាន Held high ឬមានបញ្ហា។ នៅពេលដែល 8086/8088

មេរៀនទី ៥ កម្មវិធី X86Asm IDE និង ការប្រើប្រាស់បញ្ជា Interrupts

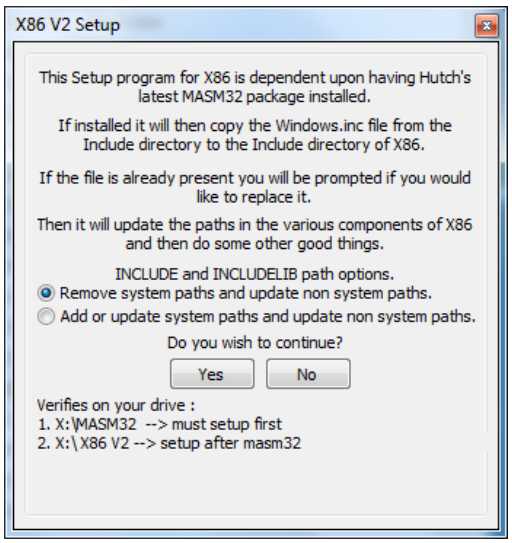
X86Asm ជាកម្មវិធីសម្រាប់សរសេរកូដភាសា Assembly មួយដោយប្រើទំហំ 8, 16 និង 32_bit Registers និងប្រើពាក្យបញ្ជារបស់ស៊ីភីយូចាប់ពីលេខ 8086 រហូតដល់ស៊ីភីយូបច្ចុប្បន្ននេះ។ វាអាចបញ្ចូលឈ្មោះអនុគមន៍របស់ C/C++ និង API ដែលជួយសម្រួលក្នុងការសរសេរកូដបានលឿន និងមានភាពងាយស្រួល។ ជាងនេះទៅទៀតការសរសេរកូដនេះប្រើសម្រាប់បង្កើតគម្រោងថ្មីៗដោយប្រើជា Console ដោយផ្ទាល់ និងអាចប្រើ Resource Editor ដែលមានទំរង់ដូចការប្រើកម្មវិធី Visual Basic ដៃក្នុងការបង្កើតDialog ឬ Windows បានលឿនបំផុត និងមិនសូវស្មុគស្មាញ។ នៅក្នុងការសរសេរកូដ Asm ដំបូងយល់ដឹងពីការប្រើប្រាស់ស្តាបត្យកម្ម និងការប្រើសំនុំពាក្យបញ្ជាដោយផ្ទាល់របស់ស៊ីភីយូ, អាចសរសេរកម្មវិធីផ្សេងៗដោះស្រាយចំណោទ ឬបញ្ហា និងជាពិសេសដឹងពីប្រមាណវិធី និង ការប្រត្តិបត្តិរបស់ស៊ីភីយូដោយផ្ទាល់។

1. ការដំឡើង X86Asm IDE

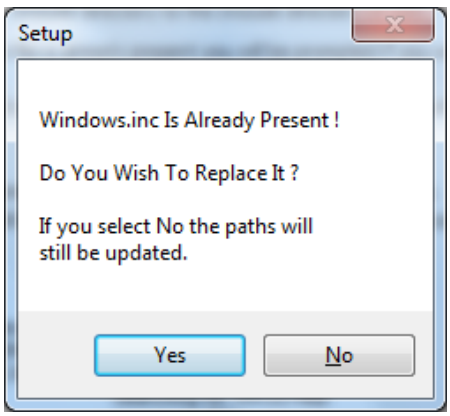
មុននឹងអាចដំណើរការកម្មវិធីទៅបាននោះ យើងត្រូវដំឡើងកម្មវិធីនេះជាមុនសិនរួមជាមួយនឹងកម្មវិធីMasm32 ដែលផ្តល់អនុគមន៍ និងCompiler។ ការដំឡើងកម្មវិធីទាំងពីរនេះត្រូវធ្វើតាមដំហានដូចខាងក្រោមនេះ៖

1.1. ចំលង Folder **X86V3** និង **Masm32** ទៅដាក់ក្នុង Drive C(បានតែ Drive C) ដោយធ្វើយ៉ាងណាឱ្យកម្មវិធីទាំងពីរនេះនៅ Drive C ជាមួយគ្នាតែនៅទីតាំងខុសគ្នា។ ហើយមិនត្រូវមាន Sub_Folder ពេលចុច Mouse ពីរលើ Folder ខាងលើដោយឃើញមាន X86V3 ឬ Masm32 ពីរដងនោះកម្មវិធីនេះក៏មិនអាចដំណើរការបានដែរ។

1.2. ចូលទៅក្នុងថត **X86V3** រួចជ្រើសរើសយកពាក្យ **setup.exe** រួច Double Click លើវាបន្ទាប់មកវាបង្ហាញ Dialog Box បន្តបន្ទាប់ដូចខាងក្រោម៖



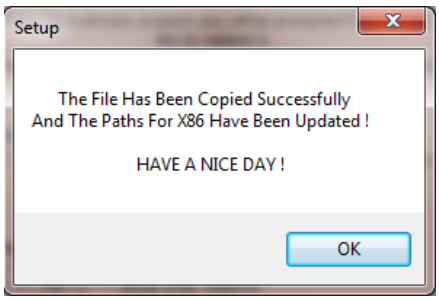
(រូបទី ១)



(រូបទី 2)

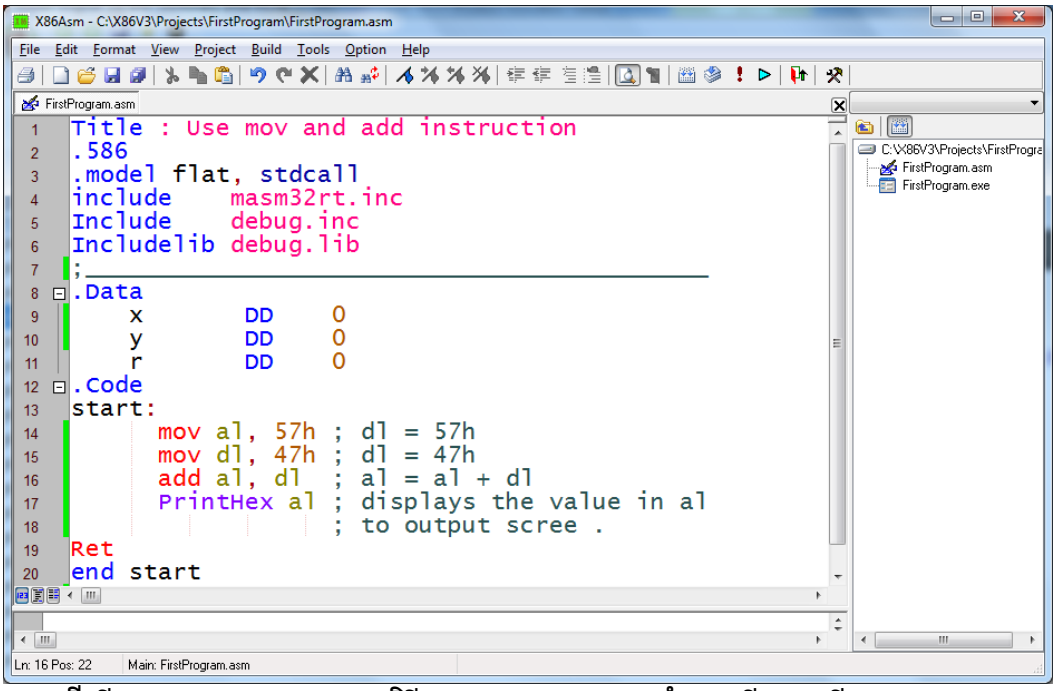
1.3. ចុច Mouse លើពាក្យ Yes រូបទី១បន្ទាប់មកវាបង្ហាញ Dialog Box រូបទី២ រួចចុច Yes

1.4. រង់ចាំរហូតដល់វាបង្ហាញ Dialog Box ដូចខាងក្រោម៖



1.5. ចុចលើពាក្យ **OK** តាមរូបខាងលើជាការបញ្ចប់ការដំឡើងកម្មវិធី X86Asm

1.6. ចូលទៅក្នុង **X86V3** Folder, រួចចុចលើពាក្យ X86.exe ដើម្បីចូលទៅក្នុងកម្មវិធី X86Asm ដូចបង្ហាញក្នុងរូបខាងក្រោម៖

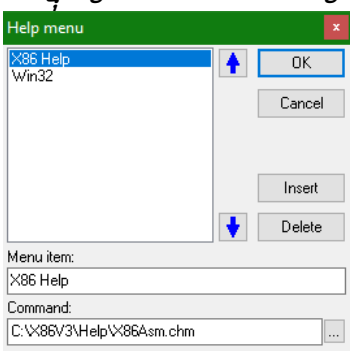


1.7. ដើម្បីសរសេរកូដ Asm ក្នុងកម្មវិធី X86Asm ទៅបានជាដំបូងយើងត្រូវមើល X86Asm Help ដែលមាននៅក្នុង Help Menu។ ហើយពេលយើងចូលក្នុងកម្មវិធីដំបូងគឺគ្មាន X86Asm Help ទេ ដូច្នេះដើម្បីដំឡើងវាជាការសំខាន់បំផុតសម្រាប់ការងារសរសេរកម្មវិធីដំបូងនៅលើ Editor នេះ៖

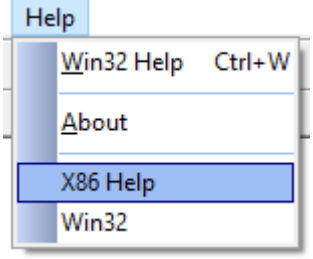
*១. ត្រូវចុចលើ **Option** ជ្រើសរើសយក **Help** menu និងបន្ទាប់បង្ហាញនូវ Dialog Box ដូចខាងក្រោមនៅដំហានទី៣ រូបខាងស្តាំ*

*២. នៅក្នុងប្រអប់ **Menu item:** វាយសេចក្តី X86 Help*

*៣. ក្នុងប្រអប់ **Command** ជ្រើសរើស C:\X86V3\Help\X86Asm.chm*



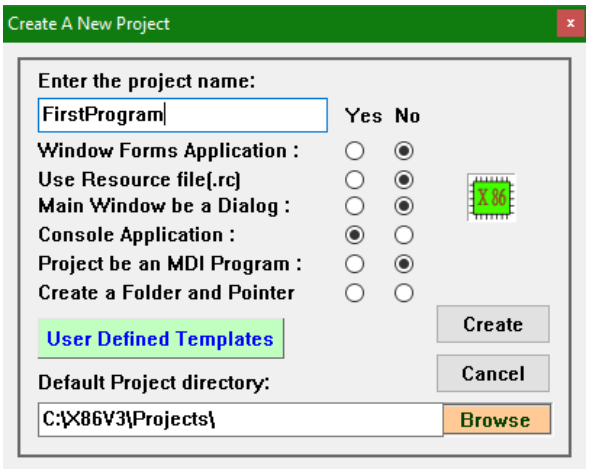
(៣)

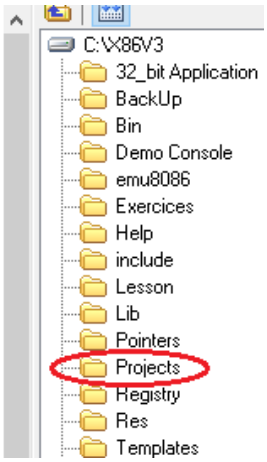


2. **កម្មវិធីដំបូង** (First Program)

ដើម្បីសរសេរកូដ Asm ដំបូងក្នុងកម្មវិធី X86Asm នោះត្រូវអនុវត្តតាមជំហានបន្តបន្ទាប់ដូចខាងក្រោម៖

1. ចុច **Ctrl + SpaceBar** ឬចុចលើ Add New Projects ពីក្នុង **Project** Menu
2. វាយឈ្មោះ File ដែលចង់បង្កើតតាមរូបខាងក្រោម និងចុច Mouse លើពាក្យ **Create** ហើយឈ្មោះ File ដែលទើបនឹងបានបង្កើតកម្មវិធីផ្ទុកវាស្វ័យប្រវត្តិទៅក្នុងថត **Projects** នៃ File Browser សូមមើលរូបខាងក្រោមខាងស្តាំ
3. ចុចលើថត Projects ដែលនៅខាងស្តាំនៃកម្មវិធី និងរកឈ្មោះដែលយើងបានបង្កើតដែលមានឈ្មោះ FirstProgram.asm





4. ការសរសេរកូដឧទាហរណ៍ដំបូងក្នុងកម្មវិធីគំរូដូចបានបង្ហាញខាងក្រោម ដោយគ្រាន់តែសរសេរឃ្លាបញ្ជា **print "Assembly Programming is fun"** និងប្រើអនុគមន៍ **getkey** និង ឃ្លាបញ្ជាក្រៅពីនឹងគឺមានស្រាប់ទាំងអស់។

```
Title : Testing First program in X86Asm
.586
.model flat, stdcall
include    masm32rt.inc
Include    debug.inc
Includelib debug.lib
;_____________________________________________
.Data ;បញ្ចូលទិន្នន័យដោយផ្ទាល់ទៅកម្មវិធី
.Code ; បញ្ចូលឃ្លាបញ្ជា និងពាក្យបញ្ជា
start:
    print "Assembly Programming is fun"
    getkey
Ret
end start
;====================================
```

Title : ប្រើសម្រាប់ដាក់ចំណងជើងឱ្យកូដបញ្ជាក់ពីប្រមាណវិធីដែលត្រូវធ្វើ

.586 ; ប្រាប់ Assembler ប្រើពាក្យបញ្ជាស៊ីភីយូ Pentium Processor ដោយប្រើ បានតែ ៣២ ប៊ីត និងអាចប្រើស៊ីភីយូ .386, .486,.586,….

.model flat, stdcall ; ប្រាប់ពីអង្គចងចាំម៉ូឌុលដែលត្រូវប្រើគឺជា Flat stdcall

include masm32rt.inc ; បញ្ចូល masm32rt.inc ដើម្បីប្រើអនុគមន៍ផ្សេងៗ

Include debug.inc ; បញ្ចូល debug.inc ប្រើអនុគមន៍ក្នុងកម្មវិធី debug

Includelib debug.lib ; បញ្ចូល Library ដើម្បីអនុគមន៍ Debug ទៅបាន

start: ; **ជា**ចំនុចចាប់ផ្តើមនៃការសរសេរកូដដែលជាអាសយដ្ឋានដំបូងនៃពាក្យបញ្ជា

print "Assembly Programming is fun" ;បង្ហាញឃ្លាក្នុងសញ្ញា “”

getkey ; ទប់អេក្រង់បង្ហាញរហូតលុះត្រាចុច Key ណាមួយបានបាត់វា

Ret ; ត្រលប់កាន់ Os ដើម្បីធ្វើការបញ្ឈប់ការប្រតិបត្តិពាក្យបញ្ជា

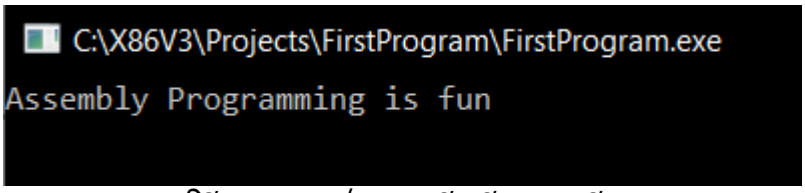
end start; ចំនុចចាប់ចុងបញ្ចប់នៃការសរសេរកូដ

ចូរមើល Help បន្ថែមដោយគ្រាន់ចុច Mouse លើ **Help Menu** , ជ្រើសរើសយក **X86 Help** , បន្ទាប់យក **Projects** និង **FirstProgram** ដើម្បីឱ្យកាន់តែច្បាស់ក្នុងអំពីការសរសេរកូដ Asm ដំបូង។

3. Build និង Run

1. ចុច Ctrl+B ឬ ចុចលើរូប ដែលមាននៅ Toolbar ដើម្បី build កម្មវិធី

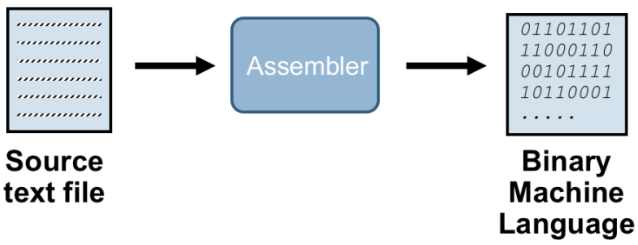
2. ចុច Ctrl + R ឬចុចលើរូប ដើម្បីធ្វើការដំណើរការ(Run)កម្មវិធីដើម្បីបានលទ្ធផលនៅលើអេក្រង់បញ្ចេញលទ្ធផល(Output Screen):



*ក្នុងករណីពេល RUN កម្មវិធីគ្មានអេក្រង់ខាងលើយើងចុចលើ **Option** Menu បន្ទាប់មកចុចលើ **Console** និង **OK**។*

4. ភាសា Assembly ប្រើក្នុងកម្មវិធី X86Asm

ភាសា Assembly(Asm) គឺជាភាសាកំរិតទាបមួយដោយប្រើពាក្យបញ្ជា, Register, RAM និង អនុគមន៍ និង Method សម្រាប់សរសេរកូដរបស់ស៊ីភីយូដោយផ្ទាល់។ វាអាចសរសេរកម្មវិធីដោះស្រាយចំណោទឬបញ្ហាដូចក្នុងភាសា C,បង្កើត GUI, driver, Owner OS, រ៉ូបូតទិក និង ឧបករណ៍រ៉ែឆ្លាត។ វាក៏ជាមូលដ្ឋានគ្រឹះសម្រាប់រៀនភាសា Asm ផ្សេងៗទៀតដែលប្រើជាមួយស៊ីភីយូ 8051 ឬ 8085 និងភាសាថ្នាក់ខ្ពស់ដូចជា Arduino ឬ Python ដែលមានរៀននៅក្នុងសៀវភៅនេះផងដែរ។ អាចប្រើវាបង្កើតកម្មវិធីទាក់ទងនឹងដំណើរការផ្នែករឹងដូចដែលយើងបានធ្វើនៅក្នុងកម្មវិធី Emu8086 ដោយយើងដឹងពីការប្រើប្រាស់ I/O, Sensor, Actuator, Memory ជាមួយនឹង Microprocessor និង Microcontroller ផងដែរ។



4.1. ប្រភេទទិន្នន័យ(Data Type)

វាជាប្រភេទទិន្នន័យសម្រាប់សម្គាល់នៃប្រភេទអថេរ ដែលត្រូវផ្ទុកលេខជាចំនួនគត់, ទសភាគ, តួអក្សរ ឬ តួអក្សរ និងទំហំលេខដែលត្រូវផ្ទុកនៅក្នុង Memory។

- លេខជា Unsigned Number(លេខអត់គិតសញ្ញា) យើងប្រើ db, dw, dd, dq
- លេខជា Signed number(លេខគិតសញ្ញា+ និង-) ប្រើ sbyte, sword, sdword
- ហើយចំពោះលេខទសភាគប្រើ Real4, Real8, Real10

C For DOS	Asm Program		
	Usigned Number	Signed Number	Floating Point
char	db(define Byte=8 bits)	sbyte	Real4
Int	dw (define Word=16 bits	sword	Real8
long int	dd (define Double word=32 bits)	sdword	Real10
Double long int	Dq (define Quard word =64 bits)	sq	

ឧទាហរណ៍១: កម្មវិធីគំរូខាងក្រោមគឺការបញ្ចូលតម្លៃ x និង y ដោយផ្ទាល់នៅក្នុងកម្មវិធីនៅក្រោម .data និងធ្វើការបង្ហាញនៅលើអេក្រង់ Output ដោយប្រើ ១៦ ប៊ីត register។

```
.data
        x  dw  200     ; declares x word and x=34h
        y  sdword -1000; declares y dword and y=-1000
.code
start:
        PrintDec x ;displays the value x on output screen
        PrintDec y ;displays the value x on output screen
Ret
end start
;Output Screen :        x = 200; Y = -1000
```

ឧទាហរណ៍២: ***កម្មវិធី***ខាងក្រោមគឺការបញ្ចូលតម្លៃ x និង y ដោយផ្ទាល់នៅក្នុងកម្មវិធី និងធ្វើប្រមាណវិធីដកលេខ និងផ្ទុកលទ្ធផលនៅក្នុង ax :

```
.data
        x  dw  200 ; declares x word and x=200
        y  dw  100 ; declares y dword and y=100
.code
start:
        mov ax, x ;move the value x into ax
        sub ax, y ;ax = ax - y
        PrintDec ax ; displays the value in ax to output screen
Ret
end start
; Output Screen : ax = 100
```

4.2. Invoke macro

វាប្រើសម្រាប់ហៅឈ្មោះអនុគមន៍ និង Method មួយចំនួន រួមនឹងធាតុរបស់វាមកប្រើប្រាស់ក្នុងការសរសេរកូដនៅក្នុងកម្មវិធី X86Asm បានលឿន និងមានភាពងាយស្រួល។ បើសិនជាយើងបញ្ចូល និងបញ្ចេញទិន្នន័យជាជាងការប្រើប្រាស់ពាក្យបញ្ជាច្រើនធ្វើឱ្យយើងពិបាកយល់។

Syntax:

INVOKE FunctionName [argumentList] ; argumentList;ចំពោះការប្រើ argumentList អាចប្រើជា៖

- តម្លៃលេខ និង កន្សោមចំនួនគត់
- ឈ្មោះអថេរ និង Register
- address និង ADDR expression

invoke crt_scanf, chr$("%d"), addr number ;ហៅអនុគមន៍ឈ្មោះ crt_scanf ដើម្បីបញ្ចូលជាចំនួនគត់ និងផ្ទុកវានៅអថេរ number

- អនុគមន៍ crt_scanf មានមុខងារដូចយើងប្រើ scanf ក្នុងភាសា C ដែរ
- Chr$ ជាបញ្ជាឱ្យបញ្ចូល String និងបញ្ចប់ដោយសញ្ញា $ ដែលបញ្ជាក់ពីការបញ្ចប់នៃការបញ្ចូលតួអក្សរ និងបំលែងជាចំនួនគត់ក្នុងប្រព័ន្ធគោល ១០ ពេលប្រើ %d។
- លេខដែលបានបញ្ចូលក្រោយពេលបំលែងហើយត្រូវផ្ទុកអាសយដ្ឋាន number មានន័យថាអថេរ number ផ្ទុកតម្លៃដែលបានបញ្ចូល។ សូមមើលឧទាហរណ៍
- *Input : 1234 ពី Keybaord ប្រើ chr$ ដោយគិតជា String :31323334h$ ពេលផ្ទុកក្នុង Memory និងប្រើ %d ដើម្បីបំលែងទៅជា 1234 វិញដូចនឹងលេខដែលយើងបានវាយបញ្ចូលនិងចុងក្រោយបានអថេរ number =1234។*

ឧទាហរណ៍: កម្មវិធីខាងក្រោមគឺការបញ្ចូលតម្លៃ x និង y ពី Keyboard ពេលដំណើរកម្មវិធី រួចធ្វើការបូកលេខ ដោយប្រើ ១៦ bit register ដោយប្រើ CPU ឈ្មោះ ៨០៨៦:

```
.data
        x  dw  0 ; declares x word and x=34h
        y  dw  0 ; declares x word and x=34h
.code
start:
        print "Enter a x:" ;  ពេលដំណើរការចេញឃ្លា enter a x: នៅលើអេក្រង់
        invoke crt_scanf, chr$( "%d" ), addr x ;  បញ្ចូលលេខចំនួនគត់ និងផ្ទុកនៅ x
        mov si, x ; move the value x into si
        print "Enter a y:" ;enter a y :
        invoke crt_scanf, chr$( "%d" ), addr y; បញ្ចូលលេខចំនួនគត់ និងផ្ទុកនៅ y
        add si, y ; si = si +y
        PrintDec si ;displays the value in si to output screen
        Ret ; Return to os to stop execute instruction
end start
; Output Screen:
```

;Enter a x: 1000

;Enter a y: 2000

SI= 3000

4.3. **ការប្រើបញ្ជា if**

if ប្រើសម្រាប់ត្រួតពិនិត្យលក្ខខណ្ឌនៃចំណោទមានមួយលក្ខខណ្ឌ និងកន្សោមលក្ខខណ្ឌត្រូវប្រើឈ្មោះអថេរ និង Register ដែលមានទំហំ 32 bit។ បើលក្ខខណ្ឌពិតអនុវត្ត StatementA។ ហើយបើលក្ខខណ្ឌមិនពិតអនុវត្តបញ្ជានៅខាងក្រោម .endif នោះគឺ StatementB។

.if condition

StatementA ; មាន statement1,statement2,...statementn

.endif ; បញ្ចប់ការប្រើ if condition

StatementB **;** មាន statement1,statement2,...statement

ចំណាំ:លក្ខខណ្ឌអាចប្រើឈ្មោះ Register និង Register; Register និង Memory ;

;Register or Memory និង immediate value ;

;តែជៀសវាងការប្រៀបធៀប Memory និង Memory

ឧទាហរណ៍: កម្មវិធីគំរូខាងក្រោមធ្វើការប្រៀបធៀបតម្លៃ x=5 < 10

```
.Data
   x      DD      5 ; ប្រកាសអថេរ x ផ្ទុកតម្លៃ 5 ដើម្បីបាន x =5
.Code
start:
     .if ( x < 10 )
        printf( "Five is now less than ten, that's a big surprise" )
     .endif
 Getkey
Ret
End start
```

4.4. **ការប្រើបញ្ជា if...else**

ការប្រើ if...else ប្រើសម្រាប់ត្រួតពិនិត្យនៃចំណោទមានលក្ខខណ្ឌពីរដាច់ពីរគ្នា។ បើលក្ខខណ្ឌក្នុងបញ្ជា if ពិតអនុវត្ត StatementA និង បើលក្ខខណ្ឌមិនពិតអនុវត្ត StatemenB

Syntax :

```
.if condition
        StatementA
.else
        StatementB
.endif
```

ឧទាហរណ៍: កម្មវិធីគំរូខាងក្រោមរកតម្លៃដាច់ខាតនៃចំនួនមួយ។ បើបញ្ចូល num=-25 ប្រើកន្សោមគណនាគឺ num = -num * (-1) ដើម្បីបានលទ្ធផល num=25។ ក្នុងភាសា Assembly ត្រូវប្រើពាក្យបញ្ជា Mul ដើម្បីធ្វើប្រមាណវិធីគុណ៖

```
.Data
    num    sword     0 ;ប្រកាសអថេរ num ជា Sign number  និង num =0
    sign   sword -1 ;ប្រកាសអថេរ Sign ជា Sign number  និង sign =-1
.Code
start:
   printf("Enter a num:") ;  ធ្វើការបង្ហាញចេញឃ្លា enter a num: នៅលើអេក្រង់
   invoke crt_scanf,chr$("%d"), addr num
   .if num<0; បើ num=-5 នោះ Assembler ដំណើរការ mov ax, num
       mov ax, num ; បើ num =-25 នោះ Ax =-25
        mul sign ; ax=ax*sign = -25  * -1 =25
        printf("The absolute %d is %d", num,ax)
   .else
      printf("The absolute %d is %d",num,num)
   .endif
   getkey
Ret
end start
```

ចំណាំ: អនុគមន៍ getkey ប្រើសម្រាប់ទប់ Output Screen និងពេលចុច Key ផ្សេងៗ ទៀតវាដើម្បីបិទវា។

; បន្ទាប់ពីដំណើរការកូដខាងលើ និងបញ្ចូលតម្លៃត្រង់ Enter a num :Output Screen

Enter a num = 25 *The absolute 25 is 25*	*Enter a num = -25* *The absolute -25 is 25*

4.5. **ការប្រើប្រាស់** if...elseif...else

ការប្រើ if...elseif...else នៅក្នុងកម្មវិធី X86Asm សម្រាប់ត្រួតពិនិត្យលក្ខខណ្ឌនៃចំណោទដែលមានច្រើនលើសពីរ។

```
.if condition1
        Statement1
.elseif condition2
        Statement2
...
.else
        Statementn
.endif
```

ឧទាហរណ៍: កូដគំរូខាងក្រោមបង្កើតម៉ាស៊ីនគិតលេខមួយដែលមានទំរង់៖

```
Enter x:
Select(+,-,*,/):
```

```
Enter y:
.Data
   x            DD       0 ;declares x as dword and x =0
   y            DD       0 ;declares y as dword and y =0
   sign DD       0 ;declares sign as dword and sign =0
   r     dq   0
.Code
start:
     print "Enter x :" ; displays Enter x :
     invoke crt_scanf, chr$("%d"), addr x ; input the value of x as integer
     print "Select(+,-,*,/):" ; Select the operator + , - ,  *, /
     invoke crt_scanf, chr$("%s"), addr sign ; input the value of sign as integer
     print "Enter y :" ; displays Enter y :
     invoke crt_scanf, chr$("%d"), addr y
     .if (sign=="+")
                  mov eax, x ; sets eax = x
                  add eax, y ; eax = eax + y
                  printf("The result of adds two number is %d", eax)
     .elseif(sign=="-")
                  mov eax, x  ; sets eax = x
                  sub eax, y   ; eax = eax -y
                  printf("The result of subtracts two number is %d", eax)
     .elseif(sign=="*")
           mov eax, 0 ; clear eax = 0
            mov eax, x ; sets eax = x
            Mul y        ; edx:eax = eax * y
            mov dword ptr r, eax ; move the value in eax to location r0 and r1
            mov dword ptr r+2, edx ; move the value in edx to cation r2 and r3
            printf("The result of multiple two number is %d", eax)
     .else
          printf("The operator is wrong")
     .endif
     getkey
Ret
end start
```

4.6. While Loop

គឺជារង្វិលជុំ(Loop)ប្រើសម្រាប់អនុវត្តនូវពាក្យបញ្ជាស៊ីភីយូ (CPU statements) ច្រើនដងដើម្បីធ្វើប្រមាណវិធីដែលមាន n ចំនួន។ ដោយដំណើរការរបស់វាត្រូវពិនិត្យលក្ខខណ្ឌជាមុនសិន។ បើលក្ខខណ្ឌពិតវាអនុវត្ត statements(s) របស់វា។

.while(condition)

statement(s)

.endw

Condition :អាចប្រៀបធៀប :

- Register vs Register : ax > bx
- Register vs Memory : ax <= x
- Register vs Immediate value : CX!=0
- Memory vs Immediate value : n <=5

ឧទាហរណ៍: កូដគំរូខាងក្រោមប្រើរង្វិលជុំ While គណនា sum =1+2+3+…+n

```
.Data
   sum     dw    0 ; declares sum=0
   n       Dw      0 ; n=0
.Code
start:
     print "Enter a n:" ; បង្ហាញឃ្លា Enter a n:
     Invoke crt_scanf, chr$( "%d" ), addr n ; បញ្ចូលតម្លៃ n
     mov dx, 1 ; dx=i=1 គឺជាតួដំបូងដែលត្រូវធ្វើប្រមាណវិធីបូក
     .while( dx<=n ) ; បើតម្លៃក្នុង dx <= n ប្រតិបត្តិបញ្ជា
                  add sum, dx ; sum=sum+dx
                  inc dx      ; dx=dx+1
     .endw
     Invoke crt_printf, chr$( "sum=%d" ), sum ; ឬអាចប្រើprint( "sum=%d",sum )
     getkey
Ret       ;return to os
end start
```

4.7. អារេ(Array)

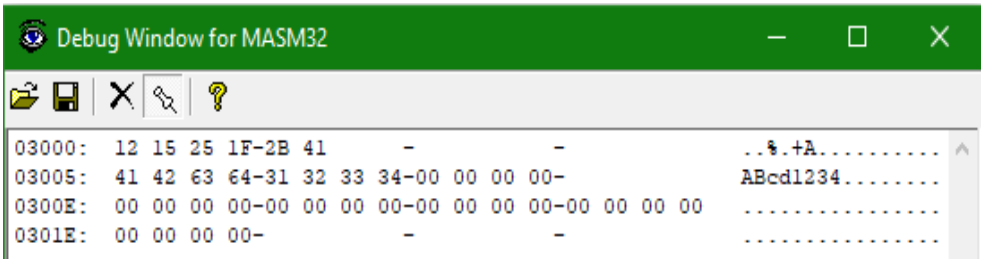
អារេគឺជាសំនុំធាតុ ឬវត្ថុដែលមានប្រភេទទិន្នន័យដូចគ្នា និងមានឈ្មោះរួមគ្នា។ ការប្រកាសអារេត្រូវចាប់ផ្តើមដោយឈ្មោះអថេរ, ប្រភេទទិន្នន័យ និងសំនុំទិន្នន័យ។ ម៉្យាងទៀតអាចជាឈ្មោះអថេរ, ប្រភេទទិន្នន័យ និងជំនួយធាតុដែលត្រូវប្រើ។

Variables *data type* list data

ឧទាហរណ៍ប្រកាសឈ្មោះ arrays Db 12h, 15h, 25h, 1Fh, 2Bh ;មានន័យថាប្រកាសឈ្មោះ arrays(byte) ហើយលេខនិមួយៗគិតជា Byte និង arrays =12h, 15h, 25h, 1Fh, 2Bh។ ហើយឈ្មោះ arrays ពេល Compile ទៅលេខអាសយដ្ឋាន 403000h ផ្ទុកលេខ 12h, 403001h =15h, 403002h =25h, 403003 = 1FH, 403004h = 2Bh។

ឧទាហរណ៍១: កម្មវិធីគម្រូខាងក្រោមបង្ហាញពីការប្រកាសអារេព្រមទាំងធាតុរបស់វា៖

```
.Data
    arrays    Db 12h, 15h, 25h, 1Fh, 2Bh ; ប្រកាសអថេរ arrays ប្រភេទ Byte
     sou db "ABcd1234", 0; ប្រកាសអថេរ sou ប្រភេទ Byte
     list    DB     20 dup( 0 ) ; ប្រកាសអថេរ List ផ្ទុកលេខ 00h( 1Byte ) ចំនួន ២0
Location
.Code
start:
        DbgDump offset array , 6 ; មើលនៅ address arry ចំនួន ៦ ទីតាំង
        DbgDump offset sou , 6 ; មើលនៅ sou ចំនួន ៦ ទីតាំង
        DbgDump offset list , 20 ; មើលទិន្នន័យនៅ address arrays ចំនួន ៦
Location
Ret
End start
```



4.8. Pointer

Pointer គឺប្រើសម្រាប់ចាប់យកទិន្នន័យដែលមាន n ចំនួនពីក្នុង Memory មកធ្វើប្រមាណវិធីដោយប្រើ Register ដើម្បីផ្ទុកលេខអាសយដ្ឋានទិន្នន័យមានដូចជា SI, DI, BX ចំពោះតែ ៨0៨៦ ស៊ីភីយូ។ តែចំពោះស៊ីភីយូចាប់ពីលេខ ៨0៣៨៦ ឡើងត្រូវប្រើ ៣២ ប៊ីត Register មានដូចជា Eax, Ebx, Ecx, Edx, Esi, Edi,...។ ចំពោះ Register ដែលត្រូវប្រើជា Pointer ត្រូវសរសេរជាក់ក្នុង [ ]

ឧទាហរណ៍១: កូដខាងក្រោមប្រើ 16 bit register ចាប់ពី CPU : 80386, 80486, 80586, Pentium Pro, and Pentium 4 Microprocessor។ សូមមើលមេរៀនការប្រើឈ្មោះ Register 32 bit:

80386 Microprocessor and later

```
.Data
   data    Db     12h, 15h, 25h, 1Fh, 2Bh
   sum     Db     0
```

8086 Microprocessor

```
. org 100h
.data
    data    Db     12h, 15h, 25h, 1Fh, 2Bh
```

```
.Code
start:
   mov ecx, lengthof data  ;ecx = 5
   mov ebx, offset data
   .while( ecx!=0 )
        mov al, [ebx]     ;al = 12h
        add sum, al       ;sum =sum +al
        add ebx,1;ebx=ebx +1
       sub ecx,1   ;ecx =ecx – 1
.endw
PrintHex sum
Getkey
Ret
End start
```

```
   sum     Db     0
.code
   mov cx, 5
   mov bx, offset data
back:
   mov al, [bx]
   add sum, al
   add bx,1
   sub cx,1   ;
   jnz back
Ret
```

ចំណាំ: *ត្រង់ Jnz back យើងប្រើពាក្យបញ្ជា loop back ដែលមានមុខងារដូចគ្នា។*

;Output : sum =96 **DS:0107 =96**

ឧទាហរណ៍២: ចូរមើលកម្មវិធីរកតម្លៃតូចបំផុតនៃ n ចំនួនដោយប្រើស៊ីភីយូ :80386, 80486, 80586(Pentium), Pentium Pro, and Pentium 4 Microprocessor។ សូមមើលមេរៀនការប្រើឈ្មោះ Register 32 bit:

```
.Data
   n      DD      0
   list   DD      20 dup( 0 )
   small  dd   0
.Code
start:
     print "This is a program to find th ",13, 10
     printf( "Enter n:" )
     invoke crt_scanf,chr$( "%d" ), addr n
     ;input n number from keyboard
     mov esi, 0
     .while( n!=0 )
        printf( "Enter num[%d]:", esi )
        invoke crt_scanf, chr$( "%d" ), addr list[esi*4]
              inc esi
              dec n
```

```
        .endw
        ;find the smallest of n number in memory
        mov n, esi
        mov esi, 0
        mov eax, list[esi]
        .while(n!=0)
                .if(eax<list[esi*4])
                        mov eax, list[esi*4] ;eax = smallest number
                .endif
                inc esi
                dec n
        .endw
        mov small, eax
        print "---------------------------",13,10
        printf("The smallest is %d\n",small)
        print "_____________________________________"
        getkey
    Ret
    end start
```

4.9. JNZ(Jump Not Zero) **Instruction**

JNZ គឺជាពាក្យបញ្ជាមួយប្រើសម្រាប់ឆ្ពោះទៅកាន់គោលដៅដើម(Target Address) ឬជា Label ដូចយើងប្រើនៅក្នុងភាសា C។ ទំរង់ទូទៅខាងក្រោមប្រើ back:។

- JNZ ឆ្ពោះទៅកាន់គោលដៅដើមបើសិនជាតម្លៃក្នុង Cx!=0 ឬ(ZF=0)។ តម្លៃក្នុង CX មានតម្លៃខុសពីសូន្យនាំឱ្យបានZF = 0។
- JNZ មិនឆ្ពោះទៅកាន់គោលដៅដើមបើសិនជាតម្លៃក្នុង Cx=0 ឬ(ZF= 1)

☹ Syntax: JNZ អាសយដ្ឋានដើម(Target Address)

ទំរង់ទូទៅនៃការពាក្យបញ្ជា JNZ គឺត្រូវប្រើជាមួយ CX, ECX, RCX ដើម្បីដំណើរការរង្វិលជុំ (Loop) ដែលមានទំរង់ទូទៅដូចខាងក្រោមដោយប្រើ back ជា Target Address:

```
        MOV CX , count ; set loop repeat count
back:   …   …          ; 1st instruction of loop
        …   …          ; 2nd instruction of loop
         .   . .
        …   …          ;nth instruction of loop
        DEC CX         ;decrement by 1
        JNZ back       ;repeat from again if cx!=0
        …   …          ; frist instruction executed
                       ; after loop is complete
```

4.10. ពាក្យបញ្ជា CMP

CMP ជាពាក្យបញ្ជាប្រើសម្រាប់ធ្វើការប្រៀបធៀបតម្លៃលេខពីរចំនួន ដើម្បីបានលទ្ធផលដូចករណីខាងក្រោម៖

Result	CF	ZF
Dest > Sou	0	0
Dest = Sou	0	0
Dest < Sou	1	0

Syntax : CMP Dest , Sou

```
//use if statement
.if ( Dest > Sou )
        statementA
.endif
```

ស្មើ EMU8086 :

```
CMP dest , sou
JA back
```

4.11. JA, JB, JE Instruction

- JA ប្រើបន្ទាប់ពី CMP ដើម្បីកំណត់យកកន្សោម Dest > Sou មានន័យថាបើលទ្ធផលក្នុងករណី Dest > Sou ឬ CF = 0 និង ZF = 0 នោះ JA ឆ្ពោះទៅកាន់អាសយដ្ឋានដើម។ ករណីក្រៅពីនេះ JA មិនឆ្ពោះទៅអាសយដ្ឋានដើម។
- JE ប្រើបន្ទាប់ពីពាក្យបញ្ជា CMP ដើម្បីកំណត់យកកន្សោម Dest = Sou មានន័យថាបើលទ្ធផលក្នុងករណី Dest = Sou ឬ CF = 0 and ZF = 1 នោះ JE ឆ្ពោះទៅអាសយដ្ឋានដើម។ ករណីក្រៅពីនេះ JB មិនឆ្ពោះទៅ Target Address។
- JB ប្រើបន្ទាប់ពីពាក្យបញ្ជា CMP ដើម្បីកំណត់យក Dest < Sou មានន័យថាបើលទ្ធផលក្នុងករណី Dest < Sou ឬ CF = 1 and ZF = 0 នោះ JB ឆ្ពោះទៅអាសយដ្ឋានដើម។ ករណីក្រៅពីនេះ JB មិនឆ្ពោះទៅទេ។
- JZ ប្រើឆ្ពោះទៅកាន់ Label បើតម្លៃក្នុង CX =0 ឬ ZF =1

ឧទាហរណ៍: ចូរធ្វើការវិភាគប្រូក្រាមខាងក្រោមរក Max នៃពីរចំនួន ដើម្បីរកចម្លើយ ក្នុង Register និង Memory រួចប្រាប់ពីមុខងាររបស់ Register និមួយៗដែលបានប្រើនៅក្នុងប្រូក្រាម?

ប្រើបញ្ជា If...else	ប្រើពាក្យបញ្ជា CMP, JA, JB, JE

.Data
x Dw 0
y Dw 0
m Dw 0
.Code
start:
print " Enter a x:"
invoke crt_scanf, chr$("%d"), addr x
print " Enter a y:"
invoke crt_scanf, chr$("%d"), addr y
mov ax, x ; eax =x
.if (ax > y)
mov m, ax ;m =eax=x
	.Data
	x DD 0
	y DD 0
	m D 0
	.Code
	start:
	print " Enter a x:"
	invoke crt_scanf, chr$("%d"), addr x
	print " Enter a y:"
	invoke crt_scanf, chr$("%d"), addr y
	mov eax, x
	cmp eax, y ; compares eax with y
	ja max_eax ; បើ eax > y ទៅmax_eax

```
    .else
        mov dx, y ;edx=y
         mov m, dx ;m =edx=y
    .endif
        printf( "max=%d",m )
getkey
Ret
end start
```

```
        mov edx,  y
        mov m, edx
        jmp print_max
    max_eax:
         mov m, eax
    print_max:
    invoke crt_printf,chr$( "max=%d" ),
```

Enter a x :60
Enter a y :50
Max = 60

Enter a x :50
Enter a y :60
Max = 60

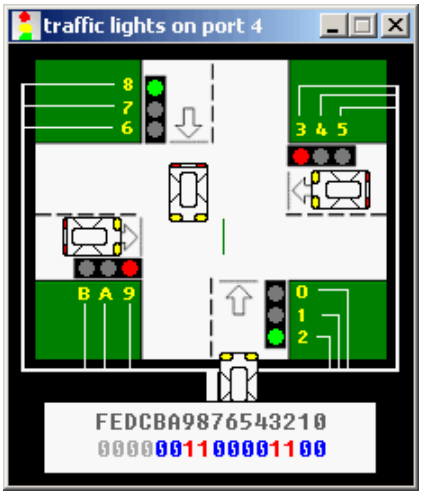
4.12. ពាក្យបញ្ជា IN និង OUT

IN ជាពាក្យបញ្ជាសម្រាប់ធ្វើការផ្ទេរទិន្នន័យពី I/O PORT ទៅកាន់ AL និង AX Register។ ពីចំណែក OUT ពីធ្វើការផ្ទេរតម្លៃពី AL ឬ AX Register ទៅកាន់ I/O port វិញ។ I/O PORT គឺជាកន្លែងសម្រាប់ធ្វើការភ្ជាប់ទៅទៅកាន់ឧបករណ៍អេឡិចត្រូនិចដើម្បីបញ្ចូលឬបញ្ចេញទិន្នន័យ។

Syntax: IN AL(AX) , PORT NUMBER និង OUT PORT NUMBER ,AL(AX)

4.13. ការតេស្តអំពូល Led ក្នុង Emu8086

ការតេស្តអំពូលដំបូងនៅក្នុងកម្មវិធី Emu8086 នោះយើងអាចប្រើឧទាហរណ៍ឈ្មោះ traffic light ដោយប្រើលេខ 04h ជាឈ្មោះ Port ឬហៅថា port 4។ ហើយយើងអាចប្រើ port 4 ក្នុងការសរសេរកូដ និងប្រើ Bits ពី 0 ដល់ Bh។ វាជាមូលដ្ឋានគ្រឹះដំបូងក្នុងការធ្វើការតេស្តជាមួយនឹងសមាសភាគអេឡិចត្រូនិចមានដូចជា Led, Button, LCD, Motor, Sensor និងឧបករណ៍ជាច្រើនទៀត សម្រាប់ការងារបន្តធ្វើគម្រោងជាច្រើននៅក្នុង Arduino និង Raspberry Board។




```
#start=Traffic_Lights.exe#
name "traffic"
.data
;                       FEDCBA9876543210
        led_red dw 0000_0000_0000_0001b
.code
        Mov ax, led_red
        Out 4, ax
ret
```

4.14. កម្មវិធី Proteus

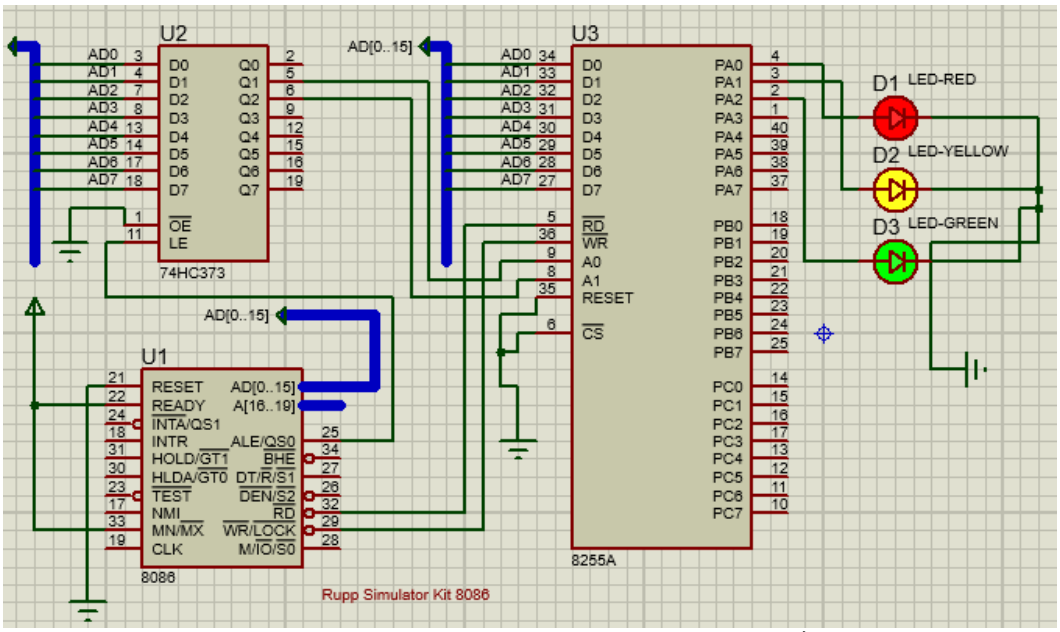
គឺជាកម្មវិធីសិប្បនិម្មិតមួយដែលអនុញ្ញាតឱ្យយើងធ្វើការរចនា និងគូរសៀគ្វីអេឡិចត្រូនិចដើម្បីធ្វើការតេស្តទៅលើឧបករណ៍ផ្សេងៗ ដូចជា LED, BUTTON, SENSOR, MOTOR, 7 SEGMENT, LCDៗលៗ ជាមួយនឹងស៊ីភីយូប្រភេទ Microprocessor និង Microcontroller។

យើងអាចប្រើកម្មវិធីនេះធ្វើនូវគម្រោងផ្សេងៗ ដូចបញ្ជាអំពូលដោយប្រើ Bluetooth, Internet, IOT, Robotics, និង AI Robot ចាប់ពីមេរៀនទី៤ ឡើងទៅ និងធ្វើគម្រោងមួយចំនួនទៀតដែលមាននៅក្នុងសៀវភៅ **ROBOTICES&AI** ជាសៀវភៅទី២សម្រាប់ប្រើនៅឆមាសទី២ សុទ្ធតែប្រើកម្មវិធីនេះទាំងអស់ដើម្បីធ្វើការតេស្តផ្នែករឹង និងផ្នែកទន់មុខនឹងដំឡើងឧបករណ៍ជាក់ស្តែង។

4.15. ការតេស្តឧបករណ៍អេឡិចត្រូនិច

4.15.1. ការប្រើ LED មួយនិងកម្មវិធីរបស់វា

ដើម្បីធ្វើការតេស្តសមាសភាគអេឡិចជាមួយនឹងស៊ីភីយូ ៨០៨៦ នោះត្រូវយល់ឱ្យច្បាស់អំពីការប្រើជើង Pin និងមុខងារនៃ ៨០៨៦ ដូចបានរៀបរាប់នៅចំណុចខាងលើ។ ចំពោះស៊ីភីយូ ៨០៨៦ វាមិនអាចដំណើរការដោយខ្លួនឯងបានទេនោះត្រូវការឈីបជំនួយមួយចំនួនធំមានដូចជាឈីប 74CH373 និង8255A។



DATA SEGMENT ; ប្រកាសប្រើផ្ទៃ data segment សម្រាប់បញ្ចូលទិន្នន័យ

PORTA EQU 00H ; កំណត់ PORTA=00h បើកជើង PORTA ជា Output

PORTB EQU 02H ; កំណត់PORTB = 02H

PORTC EQU 04H ; កំណត់ឈ្មោះ PORTC ឱ្យស្មើនឹងតម្លៃកូដ 04H

PORT_CON EQU 06H ;កំណត់PORT ទាំងបីជា Input និង Output

DATA ENDS

CODE SEGMENT ; ប្រកាសប្រើផ្ទៃ code segment សម្រាប់បញ្ចូលកូដ

MOV AX, DATA ; Ax = DATA ដែលជាលេខអាសយដ្ឋានផ្ទៃ Data Segment

MOV DS, AX ; DS = AX ដែលជាលេខអាសយដ្ឋានផ្ទៃ Data Segment

ORG 0000H ; កំណត់ប្រើលេខអាសយដ្ឋាននៃកូដដំបូង

START:

; កំណត់ជើង Pin នៃ Prot A, Port B និង Port C ជា Input និង Output

MOV DX, PORT_CON

```
        MOV AL, 10000000B ; port C (output), port A (output) in mode 0 and
        ;PORT B (INPUT) in mode 0
        OUT DX, AL  ; DX = PORT_CON
 ;JMP XX
XX:   ;Turn OFF 3 LED
        MOV AL, 00000000b ; AL = 0000 0000b
        MOV DX, PORTA       ; DX =PORTA បើកជើងពីននៃPORTA ជា Output
       OUT DX, AL ; PORTA =0000 0000b បញ្ចេញតម្លៃទៅពីន PA0=0,PA1=0,
       PA2=0;   បើពីនទាំងពីរស្មើសូន្យគឺបិទអំពូលទាំងបីព្រោះពីនទាំងនេះតភ្ជាប់ទៅ
       LED
        MOV CX, 5000 ; Delay ចំនួន 5000 Millisecon ស្មើ ៥ Second
loopy1: loop loopy1 ; បន្តយតម្លៃក្នុង CX ឆ្ពោះទៅកាន់ loopy1 បើខុសពីសូន្យ
        MOV AL, 00000111b ; Al =00000111b ធ្វើការ ON LED បី
        MOV DX, PORTA     ; DX =PORTA បើកជើង PinនៃPORTA ជា Output
        OUT DX,AL          ; PORTA =00000111b ជើងពីន PA0=1,PA1=1,PA2=1
        ; បើពីន PA0=1,PA1=1,PA2=1អំពូលទាំងបីភ្លឺព្រោះពីនទាំងនេះតភ្ជាប់ទៅ LED
        MOV CX, 5000; Delay ; Delay ចំនួន 5000 Millisecon ស្មើ ៥ Second
loopy2:loop loopy2
JMP XX
CODE ENDS
END
```

- ការប្រើ **LED** និងកម្មវិធីគំរូខាងលើវាជាគ្រឹះដំបូងដ៏សំខាន់បំផុតសម្រាប់ធ្វើការងារបន្តបន្ទាប់ដែលមាននៅក្នុងសៀវភៅនេះ។

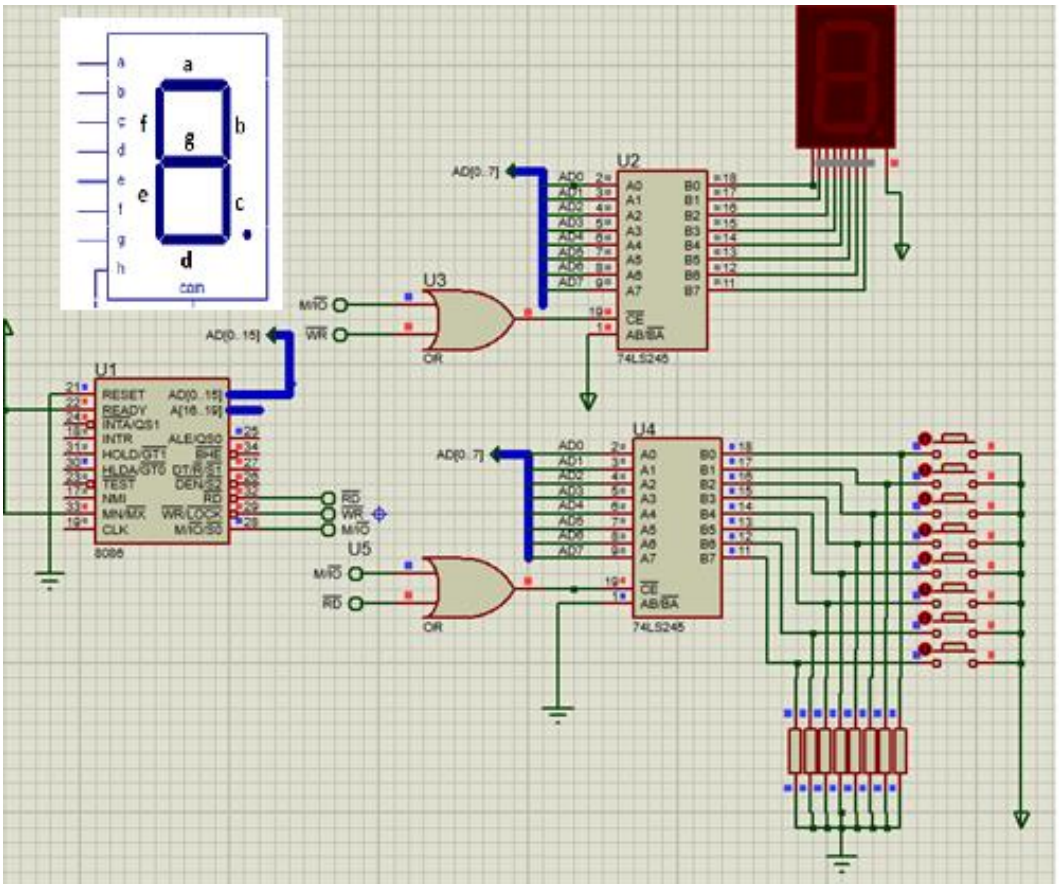
- ឈីប 74CH372 ប្រើសម្រាប់ផ្ទុកទិន្នន័យ និងពាក្យបញ្ជាបណ្តោះអាសន្នរវាងដែលបានមកពីស៊ីភីយូ 8086 ឬលេខលើសពីនេះ។ ហើយចំពោះ 74LS254 មានមុខងារដូចឈីប 8255A។ រូបខាងក្រោម គឺប្រើ PORTB(B7-B0) សម្រាប់ធ្វើការបញ្ចេញឬបញ្ចូលតម្លៃ និងប្រើជាមួយសៀគ្វីតក្ត។

- បន្ទះឈីប 8255A នេះប្រើសម្រាប់បញ្ចេញ និងបញ្ចូលទិន្នន័យរវាង 8086 និង ឧបករណ៍ I/O Port វាត្រូវបានគេរចនាឡើងសម្រាប់ប្រើជាមួយសមត្ថភាពស៊ីភីយូ ៨ប៊ីត ១៦ប៊ីត និងខ្ពស់ជាង។ វាមានបណ្តាញបញ្ចូល/បញ្ចេញទិន្នផលចំនួន២៤ ដែលអាចនឹងត្រូវបានរៀបចំជាលក្ខណៈជាពីរក្រុមដែលមានដប់ពីរខ្សែចូលគ្នា, ឬបីក្រុមដែលមានប្រាំបីខ្សែជាច់ៗពីគ្នា។ Port I/O មានចំនួនពីរក្រុមត្រូវបានដាក់ឈ្មោះជាក្រុម A និងក្រុម B។ ដូច្នេះក្រុម A មាន PortA ទំហំ ៨ប៊ីត និងរួមជាមួយនឹង Port C(PC7-PC4) ៤ប៊ីត។ ចំណែកក្រុម B មាន Port B ទំហំ ៨ប៊ីត និង Port C(PC3-PC0) ដោយទំហំ ៤ប៊ីត។ វាជាជើង Pin សម្រាប់ឱ្យអ្នកសរសេរកូដយកវាមកប្រើដើម្បីបញ្ចេញឬបញ្ចូលទិន្នន័យ រួមនឹង PortA។ ដោយ Port A សម្គាល់ដោយនិមិត្តសញ្ញា PA0-PA7 ខណៈពេលដែល

Port C ត្រូវបានសម្គាល់ជា PC4-PC7។ ស្រដៀងគ្នានេះដែរ Port B ៨ ប៊ីតដែលមានខ្សែ PB0-PB7។ ដោយផត(Port)ពី D0 ដល់ D7 ជា Bi-direction Bus អាចបញ្ចេញឬបញ្ចូលទិន្នន័យរវាងពីស៊ីភីយូ ៨០៨៦ និងឈីប 74CH373។

4.15.2. Segement និង Button

7 segment គឺជាឧបករណ៍បញ្ចេញព័ត៌មានអេឡិចត្រូនិចមួយដោយវាមួយអាចបញ្ចេញលេខគោលដប់ និងប្រើវាក្នុង digital clocks, electronic meters, basic calculators, និងឧបករណ៍អេឡិត្រូនិចផ្សេងៗទៀត។ ប្រើ Rupp Simulator Kit ដើម្បីបានរូបដូចខាងក្រោម៖




```
CODE    SEGMENT PUBLIC 'CODE'
        ASSUME CS:CODE
START:
    xor al,al ; al = al xor al = 0000 0000b
    mov al, 00000001h  ; al = 0000 0001h
loopf:
back:
     IN AL, 66H; បញ្ចូល PORT  66H ទៅកាន់ AL
```

```
        CMP AL, 00h       ; ធ្វើការប្រៀបធៀប AL= 66h និង ០០ ។ពេលចុចមិន  ;ចុច
        Key តម្លៃ AL =00h=00h  JZ  ទៅ back និងចុច key តម្លៃក្នុង   AL=01h=00h
        JZ  ទៅ back មិនទៅ back
        JZ back
        ; បើបានចុច Key អនុវត្តពាក្យបញ្ជាខាងក្រោម
        NOT AL                    ; AL =FEh
        OUT 66H, AL               ;
        JMP loopf                 ; Forever
    CODE    ENDS
        END START
```

ចូរធ្វើការកែប្រែកម្មវិធីខាងលើដោយធ្វើការតេស្តដោយចុចប៊ូតុងម្តងឱ្យ Led ភ្លឺ និងចុចម្តងទៀតឱ្យ Led បិទ

4.15.3. Motor បើកកម្មវិធី Emu8086 ចុចលើ example និងជ្រើសរើស stepper motor រួចធ្វើការតេស្តវាក្នុងកម្មវិធីនេះ និងធ្វើការដំឡើងវានៅក្នុងកម្មវិធី Proteus។

```
#start=stepper_motor.exe# ;  to run devices to in program
name "stepper"
#make_bin#
steps_before_direction_change = 20h ; 32 ( decimal )
jmp start
```

```
; bin data for clock-wise
; half-step rotation:
datcw    db 0000_0110b
         db 0000_0100b
         db 0000_0011b
        db 0000_0010b
; bin data for counter-clock-wise
; half-step rotation:
datccw   db 0000_0011b
         db 0000_0001b
         db 0000_0110b
          db 0000_0010b
; bin data for clock-wise
; full-step rotation:
datcw_fs db 0000_0001b
         db 0000_0011b
```

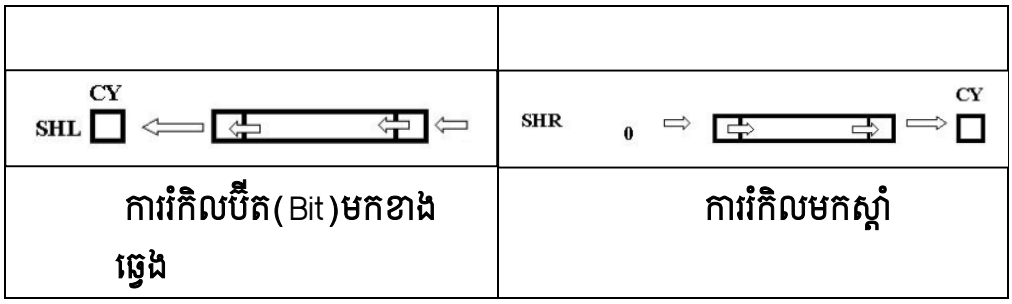
```
clock-wise half-step.
mov si, 0
mov cx, 0 ; step counter
next_step:
; motor sets top bit when it's ready
to accept new command
wait:   in al, 7
        test al, 10000000b
        jz wait
mov al, [bx][si]
out 7, al
inc si
cmp si, 4
jb next_step
mov si, 0
inc cx
```

```
            db 0000_0110b
            db 0000_0000b
; bin data for counter-clock-wise
; full-step rotation:
  datccw_fs db 0000_0100b
                      db 0000_0110b
                        db 0000_0011b
                        db 0000_0000b
    start:
    mov bx, offset datcw ; start from
    cmp cx,
    steps_before_direction_change
    jb  next_step
    mov cx, 0
    add bx, 4 ; next bin data
    cmp bx, offset datccw_fs
    jbe next_step
    mov bx, offset datcw ; return to
    clock-wise half-step.
jmp next_step
```

4.16. SHL និង SHR Instruction

4.16.1. ការរំកិលប៊ីត(Bit)មកខាងឆ្វេង

SHL ជាបញ្ជាមានតួនាទីធ្វើការរំកិលលេខគោលពីរពីខាងស្តាំទៅខាងឆ្វេង។ គ្រប់ការរំកិលលេខដោយប៊ីតនៅត្រង់ MSB ត្រូវផ្ទុកនៅត្រង់ CF និងប៊ីតត្រង់ LSB បំពេញដោយលេខ 0 និងចំនួនប្រមាណវិធីដែលត្រូវរំកិលពីខាងឆ្វេងខាងស្តាំតាមតម្លៃក្នុង CL។

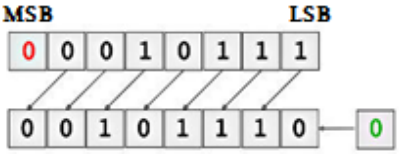


ការរំកិលប៊ីត(Bit)មកខាងឆ្វេង	ការរំកិលមកស្តាំ

ទំរង់ទូទៅ: SHL dest, CL; cl store number of times to shift

Flags: CF = 0, OF = 0, SF, ZF, PF

ឧទាហរណ៍១: ចូរធ្វើប្រមាណវិធីរំកិលលេខ 00010111 ដោយរំកិលចំនួនមួយដង៖



ឧទាហរណ៍២: ចូរធ្វើប្រមាណវិធីតាមកូដខាងក្រោម៖

```
.Data
        x   db 0A7H ; declares x stores 0A7h
.Code
start:
    mov al, x ; sets al stores the value of x
    mov cl, 3 ; cl = 3 is number to perform shift bit
    shr al, cl   ;shift bit the value in al by cl
```

```
    PrintHex al   ; displays the value in al to output screen
  Ret
  end start
  ; Output Screen : AL =0001 0100b=14h
```

4.16.2. ការរំកិលមកស្តាំ

SHR ជាបញ្ជាមានតួនាទីធ្វើការរំកិលលេខគោលពីរពីខាងឆ្វេងទៅខាងស្តាំ។ គ្រប់ការរំកិលលេខដោយបីតនៅត្រង់ LSB ត្រូវផ្ទុកនៅត្រង់ CF និងបីតនៅត្រង់ MSB បំពេញដោយលេខ 0 និងចំនួនប្រមាណវិធីដែលត្រូវរំកិលពីខាងឆ្វេងខាងស្តាំតាមចំនួនក្នុង CL។

ទំរង់ទូទៅ: **SHR** dest, CL

SHR dest, CL; cl store number of times to shift

Flags: CF = 0, OF = 0, SF, ZF, PF

4.17. ការបង្កើត GUI

គឺជាកម្មវិធីមួយដែលជួយឱ្យអ្នកប្រើប្រាស់មានទំនាក់ទំនងជាមួយទៅនឹងកុំព្យូទ័រ។ នៅក្នុងប្រូក្រាម X86Asm គឺផ្តល់នូវ GUI សម្រាប់បង្កើតទំរង់ផ្សេងៗនៃកម្មវិធីសម្រាប់ប្រើប្រាស់ដែលមានភាពងាយស្រួល និង លឿនបំផុត។ ហើយវាជាមូលដ្ឋានគ្រឹះដំបូងក្នុងការបង្កើត Interface សម្រាប់កម្មវិធីផ្សេងៗទៀតដូចជាក្នុងភាសា Python ឬ App Inventor ជាដើម។

ដើម្បីធ្វើការបង្កើតកម្មវិធី GUI ក្នុងប្រូក្រាម X86Asm នោះត្រូវអនុវត្តតាមជំហានដូចខាងក្រោមនេះ

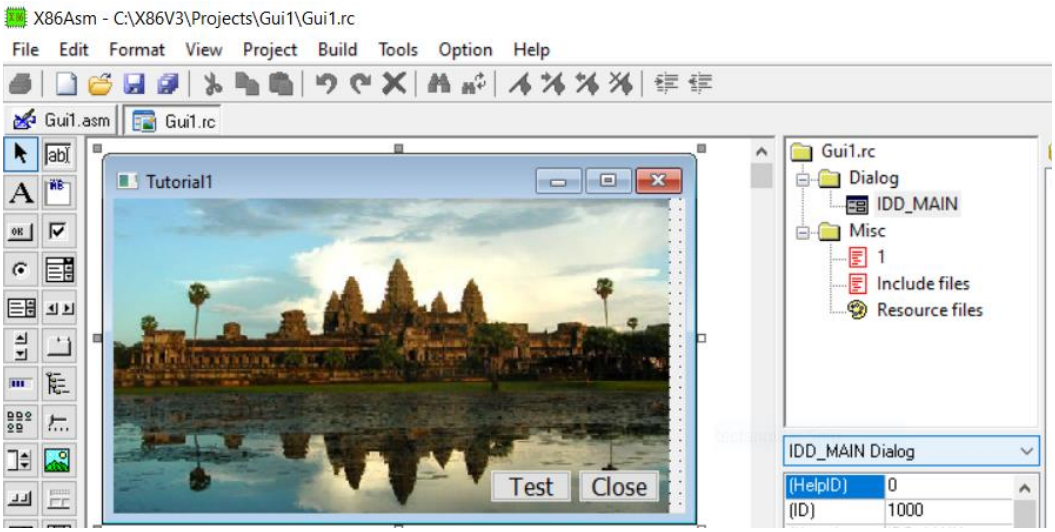
១. ត្រូវចុចលើ Project និងជ្រើសយក Add new projects បន្ទាប់មកដាក់ឈ្មោះ GUI1

២. ចុចលើប៊ូតុង User Defined Templates និងជ្រើសយក Tutorial1 រួចចុចលើ Return បន្ទាប់ចុចលើប៊ូតុងនេះរួចហើយបានកូដ និង main window ដោយស្វ័យប្រវត្តិដូចខាងក្រោម៖

```
Title : Program Use MssageBox and Sendmessage function
.386
.model flat, stdcall  ;32 bit memory model ; ប្រើ Memory model
option casemap :none  ;case sensitive  ; ប្រើប្រភេទ Case sensitive
include Tutorial1.inc     ; បញ្ចូល Library ឈ្មោះ Tutorial1 ដើម្បីចេញ GUI ដំបូង
include rsrc.inc          ; បញ្ចូល Library ឈ្មោះ rsrc សម្រាប់ការ Resource  Editor
.data
    tiltle DB" Test Function ", 0 ; ប្រើប្រាស់ អថេរ tiltle ដាក់ចំណង់ជើង Window
    txt    DB" This is a button1", 0 ;ប្រើប្រាស់ អថេរ txt ដាក់ Text នៅលើប៊ូតុង
.code
start:
    invoke  GetModuleHandle,NULL ; ចាប់តម្លៃអាសយដ្ឋានម៉ូឌុលកម្មវិធីចូល Eax
    mov     hInstance, eax   ; hInstace ផ្ទុកអាសយដ្ឋានម៉ូឌុលពីក្នុង Eax register
    invoke  InitCommonControls ; ចាប់ផ្តើមឱ្យធ្វើការកំណត់Controls ផ្សេងៗ
```

```
    ;ឃ្លាបញ្ជាខាងក្រោមធ្វើការបង្ហាញផ្ទៃ Main Window និងហៅ DlgProc ដំណើរការ
    ;ដែលនៅក្នុង Procedure នេះកូដបញ្ជាទៅ Widgets សម្រាប់ធ្វើប្រមាណវិធីផ្សេងៗ។
    invoke  DialogBoxParam,hInstance,IDD_MAIN,NULL,addr DlgProc,NULL
    invoke   ExitProcess, 0 ; ពេលចុច exit ឬចុចលើសញ្ញាខ្វែងចេញពីកម្មវិធី GUI
DlgProc  proc NhWnd:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM
    .if uMsg==WM_INITDIALOG
    .elseif uMsg==WM_COMMAND ; លក្ខខណ្ឌនេះពិតពេលធ្វើការចុចលើប៊ូតុងណា; ;
        មួយសូមមើលការពន្យល់ត្រង់ចំណាំខាងក្រោម៖
       Mov ecx, wParam ; ecx =1002 ជា ID  ពេលចុចលើប៊ូតុងលើ [Test]
       movzx eax, cx      ; កំណត់ eax = cx សម្រាប់ប្រើនឹងលក្ខខណ្ឌខាងក្រោម
       shr ecx, 16   ;ធ្វើឱ្យតម្លៃក្នុង ecx =0 ដើម្បីផ្ទៀងផ្ទាត់ ecx==BN_CLICKED
       .if ecx==BN_CLICKED ; ពិតមកដល់ចំណុចនេះពេលចុចលើប៊ូតុងមួយ
        .if eax==IDC_BTN1 ; បើចុចលើប៊ូតុងតេស្ត [Test] វាពិតលក្ខខណ្ឌនេះ
        ; បង្កើត Title, Text និង Button នៅលើ Message Box
                InvokeMessageBox, hWnd, Addr txt, Addr tiltle, MB_OK
        .elseifeax==IDC_BTN2 ; បើចុចលើប៊ូតុង [Close] វាពិតលក្ខខណ្ឌនេះ
        ;ធ្វើការបិទ Dialog Box
         invoke  SendMessage,hWnd,WM_CLOSE,NULL,NULL
         endif
      .endif
    .elseif  uMsg==WM_CLOSE
         invoke EndDialog, hWnd,0 ; ការបិទ Main Window
    .else
         mov  eax, FALSE
         ret
    .endif
    mov   eax,TRUE
    ret
DlgProc endp
end start
```

៣. ធ្វើការចុចលើឈ្មោះ GUI1.rc ដែលមានកន្ទុយ .rc និង ចុចលើ IDD_MAIN និងបន្ទាប់មកបានផ្ទៃកម្មវិធី GUI ដូចរូបខាងក្រោម៖



៤. មុខនឹងធ្វើការ **Build** និង **Run** ត្រូវកំណត់ឈ្មោះ Head File ឱ្យដូចនឹងឈ្មោះ file ដែលយើងបានបង្កើត។ ត្រលប់ទៅកូដចុចលើ Tab: **GUI1.asm** រួចប្តូរត្រង់ include Tutorial1.inc ទៅ include **GUI1.inc** និងត្រលប់ទៅ Main Window ដោយចុចលើ Tab: GUI1.rc

៥. ចុងក្រោយចុចលើ **Project** និងជ្រើសរើសយកពាក្យ **Export names រួចចុចលើ Save**

៦. ចុចលើ Build និង Run និងបានកម្មវិធីដូចរូបខាងលើ

ចំណាំ: ១. ចំពោះការប្រើប្រាស់កូដ Window Message ដែលកំណត់ប្រើដោយ Masm32 គឺ WM_COMMAND = 0x0111, BN_CLICKED=0, WM_CLOSE=0x0010, និង WM_CREATE =0x0001 និងករណីផ្សេងៗទៀតសូមមើលតារាងរបស់វា។

២. សូមបង្កើត GUI បន្តបន្ទាប់ទៀតដោយខ្លួនឯងដោយមើលពី Tutorial2 ដល់ Tutorial 10 និង Tutorial ផ្សេងៗទៀតដែលខ្ញុំបានបង្កើតនៅក្នុងប៊ូតុង **User Defined Templetes** ដើម្បីធ្វើការតេស្ត Widgets ដូចជា Edit Box, Radio, Combo Box,..។

4.17.1. អនុគមន៍ MessageBox : ជាអនុគមន៍បង្កើត ធ្វើការបង្ហាញ និងធ្វើប្រមាណវិធីនៅលើ Message Box។

InvokeMessageBox, hWnd, lpText, lpCaption, uType

hWnd : handle of owner window ផ្ទុកលេខអាសយដ្ឋានតំណាងឱ្យ Window

lpText : address of text in message box ផ្ទុកលេខអាសយដ្ឋានតំណាងឱ្យ Text

lpCaption : address of title of message box លេខ Address នៃចំណងជើង

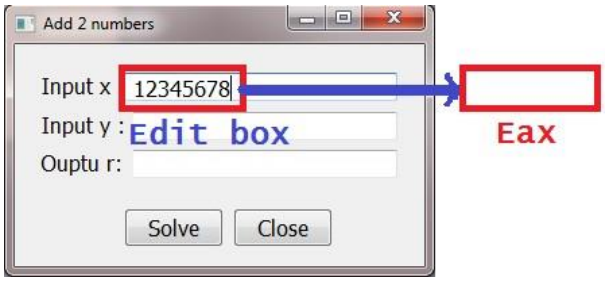
uType : style of message box ; Style នៃ Message Box ដូចជា OK ឬ Cancel

ឧទាហរណ៍: កម្មវិធីគំរូខាងក្រោមប្រើអនុគមន៍ MessageBox និង Send message ដើម្បីធ្វើការបង្ហាញ និងជាគំរូដំណើរសម្រាប់ដំណើរការធ្វើ GUI ក្នុងភាសា Assembly។

4.17.2. SendMessage: ជាអនុគមន៍ធ្វើការបញ្ជូន Message មួយទៅឱ្យ Window មួយ ឬច្រើន

- InvokeMessageBox, hWnd, Msg, wParan, lParam
- hWnd : handle of owner window
- Msg : Message to send
- wParam : first message parameter
- lParam : second message parameter

4.17.3. GetDlgItemInt: ជាអនុគមន៍មួយប្រើសម្រាប់ចាប់យកតម្លៃពីក្នុង Widget មួយទៅដាក់ក្នុង Eax register។



InvokeGetDlgItemInt, hDlg, nIDDlgItem , 0 , bSigned

- hDlg : handle to dialog box
- nIDDlgItem : control identifier
- bSigned : specifies whether value is signed or unsigned

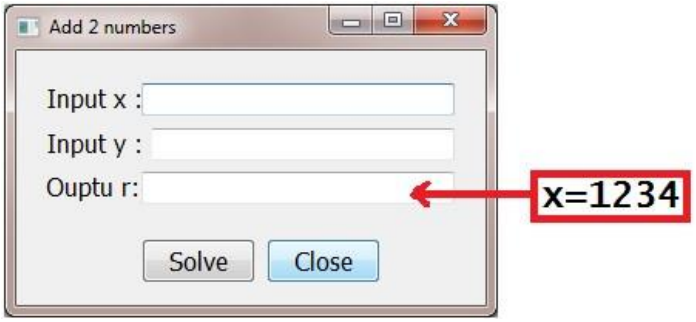
៣ . dwtoa function ប្រើសម្រាប់បំលែងតម្លៃលេខចំនួនគត់ទៅជាលេខ ACII។

eax=1234 dw(dword) to a(ascii):31323334h

Invokedwtoa, dwValue, lpBuffer

- dwValue : The DWORD value to convert
- lpBuffer : The address of the buffer to put the converted DWORD into.

4.17.4. SetDlgItemText: ជាអនុគមន៍សម្រាប់កំណត់យកតម្លៃពីក្នុងអថេរមួយទៅដាក់ក្នុង Widget(Control) វិញ។



Syntax:

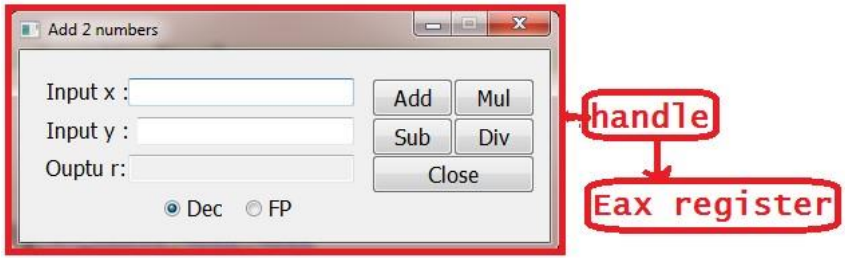
InvokeSetDlgItemText, hDlg, nIDDlgItem , lpString

hDlg : handle to dialog box

nIDDlgItem : control identifier

lpstring : text to set

4.17.5. GetDlgItem: ជាអនុគមន៍ប្រើសម្រាប់ចាប់យកតម្លៃ Handle (ជាអថេរមួយតំណាងឱ្យ Dialog box)ទៅដាក់នៅក្នុង Eax register។



Syntax:

InvokeGetDlgItemText, hDlg, nIDDlgItem

hDlg : handle to dialog box

nIDDlgItem : control identifier

5. ការប្រើប្រាស់បញ្ជា Interrup(INT)

INT គឺត្រូវប្រើជាឈ្មោះ និងលេខបង្កើតជាបញ្ជាមួយសម្រាប់ធ្វើអ្វីមួយជាក់លាក់ ដោយប្រើជាមួយនឹងលេខអនុគមន៍។ វាប្រើប្រាស់សម្រាប់សរសេរប្រូក្រាមមានលក្ខណៈងាយស្រួល និងបានលឿន។ វាជំនួយក្នុងឱ្យការសរសេរកូដមួយដើម្បីធ្វើការបង្ហាញតួអក្សរមួយឬច្រើននៅលើអេក្រង់ដោយគ្រាន់តែហៅ INT មកប្រើ និងអាចធ្វើការងារផ្សេងៗទៀតដូចជាប្រើវាជាមួយអនុគមន៍ដែលអាចធ្វើការជាមួយឌីស Drive ឬ ជាមួយឧបករណ៍បញ្ចេញឬបញ្ចូល(IO) ដូចជា Mouse ឬ Keyboard ជាដើម។ យើងអាចហៅផ្នែកទន់ Interrupts បង្កើតជាផ្នែកប្រូក្រាមសម្រាប់ print និង scan ដែលបានប្រើនៅក្នុងកម្មវិធី X86Asm និង C ប្រូក្រាមដែរ។ ***នោះមានន័យថាលេខ INT រួមនឹងលេខអនុគមន៍បង្កើតជាផ្នែកនៃប្រូក្រាមមួយហើយត្រូវបានប្រើជាមួយពាក្យបញ្ជា (Instructions) និង Register របស់ស៊ីភីយូសម្រាប់ធ្វើប្រមាណវិធីជាមួយផ្នែករឹងនិងផ្នែកទន់របស់កុំព្យូទ័រ (Computer)។***

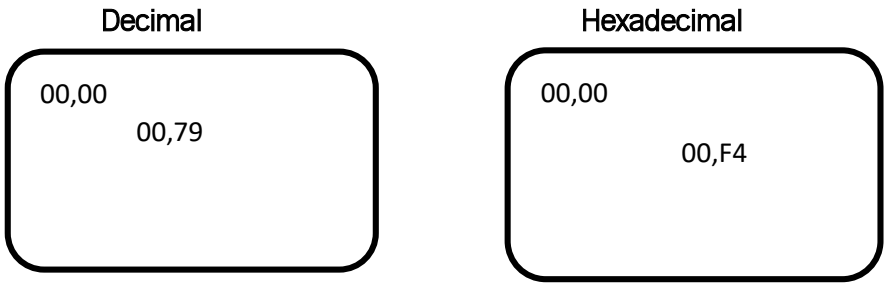
5.1. BIOS INT 10H និង DOS INT 21H

មានប្លុកកូដដែលមានសារៈសំខាន់ខ្លះមាននៅក្នុង BIOS និង មាននៅក្នុង DOS ដែលអាចប្រើទៅបានសម្រាប់ឱ្យអ្នកប្រើប្រាស់ប្រើពាក្យបញ្ជា INT។ ការប្រើបញ្ជានេះគឺមានទំរង់ទូទៅដូចខាងក្រោម៖

Format INT XX ; xx ប្រើជាលេខគោលដប់ប្រាំមួយដោយមានដែនកំណត់ពី 00H ទៅ FFH នោះវាមានសរុបចំនួន ២៥៥ INT។ នៅក្នុងគ្រួសារ x86 ហើយ INT និមួយៗ មានមុខងារខុសៗគ្នា។ ដូចជាការប្រើ INT 10H និង INT 21 H គឺបញ្ជូលតម្លៃត្រូវប្រើ AH Register ហើយតម្លៃនៃអនុគមន៍គឺមានពី 00H ទៅ 255H ដែរ។

5.1.1. ការសំអាតអេក្រង់ដោយប្រើ Int 10H និង Function 06H

ការប្រើ Int 10H និង Function 06H ធ្វើការសំអាតអេក្រង់ដែលមានតម្លៃក្នុង Register ជាមុនសិនមុននឹងប្រើ INT 10H ដើម្បីធ្វើការហៅធាតុ: AH=00, AL=00, BH=07, CX=0000, DH = 24 និង DL= 79 មកប្រតិបត្តិ។ ខាងក្រោមនេះជាកម្មវិធីក្នុងការប្រើ Int 10H , Function 06 :




```
MOV AH,06 , AH=06 to select scroll function
MOV AL,00 ; AH = 00  the entire page
MOV BH,07 ; BH = 07  for normal attribute
MOV CH,00 ; CH = 00 row value of start point
MOV CL,00 ; CL =  column value of start point
MOV DH,24 ; DH = 24 row value of end point
```

MOV DL,79 ; DL = 79 column of end point

INT 10 H ; Invoke interrupt ហៅជាតុផ្សេងៗនៅក្នុងអនុគមន៍មកប្រតិបត្តិ

គួរកត់សំគាល់ដែរថានៅក្នុងកម្មវិធី Debug ការបញ្ចូលតម្លៃដោយផ្ទាល់ទាំងអស់ត្រូវគិតជាលេខក្នុងប្រព័ន្ធគោលដប់ប្រាំមួយដោយប្រើលេខគោលដប់(0ទៅ9) និងអក្សរ(AទៅF)។

5.1.2. ការកំណត់ទីតាំង Cursor នៅលើអេក្រង់ដោយប្រើ Int 10H និង Function 02H

ការប្រើ INT 10H និង Function 02H សម្រាប់កំណត់ទីតាំង Cursor នៅទីតាំងផ្សេងៗទៅលើអេក្រង់ដែលមានតម្លៃនៅក្នុង Register DX ផ្ទុកចំនួនជួរដេក(Rows) និងជួរឈរ(Columns)។

🕮 ទំរង់ទូទៅ

AH = Value of function	MOV AH, VALUE OF FUNCTION
BH = Page 00	MOV BH , Page 00
DL = Column	MOV DL , COL
DH = Row	MOV DH, ROW

☞ឧទាហរណ៍: សរសេរកូដដើម្បីកំណត់ទីតាំង Cursor នៅ row = 15 = 0FH និង column = 25H

MOV AH, 02 ; ផ្ទេរតម្លៃ Function :02H ចូលទៅក្នុង AH register

MOV BH, 00 ; ផ្ទេរតម្លៃ Page :00H ចូលទៅក្នុង BH register

MOV DL, 25 ; ផ្ទេរតម្លៃ Column :02H ចូលទៅក្នុង DL register

MOV DH, 15 ; ផ្ទេរតម្លៃ Column :02H ចូលទៅក្នុង DL register

INT 10 H ; Invoke int 10H ហៅជាតុដែលមាននៅក្នុងអនុគមន៍មកប្រតិបត្តិ

☞ឧទាហរណ៍: ចូរបង្កើត CLS Command ដោយប្រើកម្មវិធី Debug ឬ C។

5.1.3. ការបង្ហាញទិន្នន័យជា String ដោយប្រើ Int 10H , Function 09H

ចំពោះការប្រើ INT 21H និង Function 09 អាចស្កែនសំនុំទិន្នន័យ ASCIIទៅកាន់ម៉ូនីទ័រ។ ដើម្បីធ្វើបានដូចនេះ Register ត្រូវមានតម្លៃ AH ស្មើតម្លៃអនុគមន៍ និង DX ផ្ទុកអាសយដ្ឋានរបស់ទិន្នន័យ ASCII។ វាធ្វើការបង្ហាញតួអក្សរទិន្នន័យដែលចង្អុលដោយ DX ហូតដល់ជួបសញ្ញា $ ទើបធ្វើការឈប់ប្រតិបត្តិ។

🕮 ទំរង់ទូទៅនៃការប្រើអនុគមន៍នេះ

AH = Value of function	*MOV AH , VAL*
DX = offset address of data	MOV DX , OFFSET ADDRESS OF DATA
INT 21H	INT 21H

☞ឧទាហរណ៍: ដើម្បីធ្វើការបង្ហាញព័ត៌មានប្រយោគ " The earth is but one country",″$″ នោះត្រូវបង្កើតឃ្លាបញ្ជាដូចខាងក្រោម៖

Data_ASCII DB " The earth is but one country",″$″ ; ប្រកាសតួអក្សរជា String

MOV AH , 09 ; ផ្ទេរតម្លៃ Function :02H ចូលទៅក្នុង AH register។

MOV DX , OFFSET Data_ASCII ; ផ្ទុកលេខ Address នៃអថេរចូលទៅក្នុង DX

Output : **The earth is but one country**

5.1.4. ការបង្ហាញមួយតួអក្សរនៅលើអេក្រង់

ការប្រើ INT 21H និង Function 02H ប្រើសម្រាប់បង្ហាញមួយតួអក្សរនៅលើម៉ូនីទ័រ។

Input : AH = 02 H DL = Single Character (ASCII)	Output : No Output

☞**ឧទាហរណ៍**: កម្មវិធីគំរូខាងក្រោមដើម្បីបង្ហាញអក្សរ "Z" នៅលើ Monitor។

```
MOV AH , 02 ; ផ្ទេរតម្លៃ Function :02H ចូលទៅក្នុង AH register
MOV DL , 5A ; ផ្ទេរតម្លៃ Character :5AH ចូលទៅក្នុង DL register
INT 21  Invoke int 21H ហៅធាតុដែលមាននៅក្នុងអនុគមន៍មកប្រតិបត្តិ
```

5.1.5. ការបង្ហាញលេខគោលដប់

INT 21H , Function 06H ប្រើសម្រាប់បង្ហាញលេខគោលដប់នៅលើ Monitor

- Input : AH = 06 H | MOV AH, VALUE OF FUNCTION
  DL = Decimal (ASCII) MOV | DL, DEC
- Output : No Output

☞**ឧទាហរណ៍**: កម្មវិធីគំរូខាងក្រោមនេះបង្ហាញលេខ Dec=98 នៅលើអេក្រង់បញ្ចេញលទ្ធផល៖

```
MOV AH , 06; ផ្ទេរតម្លៃ Function :02H ចូលទៅក្នុង AH register
MOV DL , 39; ផ្ទេរតម្លៃ Function :39H ចូលទៅក្នុង DL register
INT 21 ;  INVOKE INT 21H
MOV AH, 06 ; ផ្ទេរតម្លៃ Function :06H ចូលទៅក្នុង AH register
MOV DL , 38; ផ្ទេរតម្លៃ Function :38H ចូលទៅក្នុង DL register
INT 21H ; Invoke int 21H ហៅធាតុដែលមាននៅក្នុងអនុគមន៍មកប្រតិបត្តិ
INT 3 ; បញ្ចប់ការប្រតិបត្តិនៅក្នុងកម្មវិធី Debug
```

☞**ឧទាហរណ៍**: ចូរសរសេរកូដដើម្បីបូកលេខគោលដប់45 + 34 ហើយបង្ហាញលទ្ធផលនៅលើអេក្រង់ (a).ប្រើអនុគមន៍ PrintDec ក្នុងX86Asm និង(b). ប្រើ Int 21H និង Function 06H។

a. ប្រើប្រាស់ Editor X86Asm

```
.data
        Result  dw 0 ; ការប្រកាសអថេរឈ្មោះ Result ដើម្បីធ្វើការផ្ទុកតម្លៃសូន្យ
.code
Start:
      mov ax, 45h ; ផ្ទេរតម្លៃ Immediate :34 ចូលទៅក្នុង Ax register។
      add ax, 34h ; Ax = AX + 34h យកតម្លៃក្នុង AX register បូកនឹង 34h
       PrintDec ax  ; ធ្វើការបង្ហាញលេខគោលដប់នៅអេក្រង់បញ្ចេញលទ្ធផល
ret
end start
```

;ធ្វើការចុចលើ Build និង Run បានលទ្ធផលនៅលើអេក្រង់បញ្ចេញលទ្ធផលគឺ ax=0058

b. ប្រើ INT 21H និង 6H(ប្រើ Editor Emu 8086)

```
org 100h ;
.data
    data dw 0
.code
;adds two bytes
    mov ax, 0x45 ; ax =
    add ax, 0x 34
 ;converting dec to ascii
    mov ah,al
    mov cl, 4
    shr ah,cl ; shift bit left the value in ah
    and al, 0x0f  ; al = al and 0x0f
    or ax,0x3030 ; ax = ax or 0x3030
    mov data, ax ; data stores the value
    mov ah , 6 ; ah = 6
    mov dl,byte ptr data+1
    int 0x21 ; invoke int 0x21 and Fuc:06
    mov ah , 6 ; ah =6
    mov dl,byte ptr data
     int 0x21
ret
```

5.1.6. ការផ្លាស់ប្តូរ Video Mode

Int 10H និង AH=00h ប្រើប្រាស់សម្រាប់ធ្វើការផ្លាស់ប្តូរ Video Mode នៅលើផ្ទៃអេក្រង់។

AL = Video Mode

0 : 40 x 25 Text Mode , Monochrome ប្រើទំហំ៤0 គុណនឹង ២៥

1 : 40 x 25 Text Mode , Color // Micro Assemble used 13

2 : 80 x 25 Text Mode , Monochrome ប្រើទំហំ ៨0 គុណនឹង ២៥

3 : 80 x 25 Text Mode , Color ប្រើទំហំ៨0 គុណនឹង ២៥ ជាប្រភេទពណ៌

4 : 320 x 200 4- Color graphic ជាប្រភេទក្រាភិចពណ៌ទំហំ 320 x 200

6 : 640 x 200 2- Color graphicជាប្រភេទក្រាភិចពណ៌ទំហំ 640 x 200

7 : Internal Mode ; ជាម៉ូតខាងក្នុងBIOS និង DOS

13h - graphical mode. 40x25. 256 colors. 320x200 pixels. 1 page.

☞**ឧទាហរណ៍**: កម្មវិធីគំរូខាងក្រោមកំណត់ Video Mode 320 x 200 Resolution។

MOV AH , 00 ; ផ្ទេរតម្លៃ Function :00H ចូលទៅក្នុង AH register

MOV AL , 4 ; ផ្ទេរតម្លៃ Immediate :4H ចូលទៅក្នុង AL register។

INT 10 H ; Invoke int 10H ហៅធាតុដែលមាននៅក្នុងអនុគមន៍មកប្រតិបត្តិ

5.1.7. INT 10H , Function 21H: ប្រើសម្រាប់បង្ហាញមួយ ១ តួអក្សរដែលមានច្រើនតួនៅក្នុង CX, ពណ៌នៅអក្សរនិងនៅលើផ្ទៃអេក្រង់ដែលផ្ទុកនៅក្នុង BL។

DEC	HEXA	BIN	COLOR
0	0	0000	Black
1	1	0001	Blue
2	2	0010	Green
3	3	0011	Cyan
4	4	0100	Red
5	5	0101	Magenta

6	6	0110	Brown
7	7	0111	Light Gray
8	8	1000	Dark Gray
9	9	1001	Light Blue
10	A	1010	Light Green
11	B	1011	Light Cyan
12	C	1100	Light Red
13	D	1101	Light Magenta
14	E	1110	Yellow
15	F	1111	White

😑**ឧទាហរណ៍:** សរសេរប្រូក្រាមដើម្បី៖

1. ផ្លាស់ប្តូរ video mode (320 x 200 text mode)
2. ការបង្ហាញអក្សរ "D" ឱ្យបាន 200H នៅទីតាំងជាមួយ Attribute black និងមាន white blinking.
3. ចូរធ្វើការកែកូដក្នុងកម្មវិធី Debug ទៅជាកូដនៅក្នុងកម្មវិធី emu8086

```
MOV  AH , 00 00 ; ផ្ទេរតម្លៃ Function :00H ចូលទៅក្នុង AH register
MOV  BH , 00 ; ផ្ទេរតម្លៃ 00H ចូលទៅក្នុង BH register
MOV  AL , 44  ; ផ្ទេរតម្លៃ 00H ចូលទៅក្នុង BH register
MOV  CX , 200 ;ផ្ទេរតម្លៃ 00H ចូលទៅក្នុង BH register
MOV  BL , F0  ; ផ្ទេរតម្លៃ F0H ចូលទៅក្នុង BL register។
INT 21H Invoke int 21H ហៅជាតុដែលមាននៅក្នុងអនុគមន៍មកប្រតិបត្តិ
INT 3
```

5.1.8. Int 10h និងFunction 0CH: ធ្វើការផ្លាស់ប្តូរពណ៌សម្រាប់ Pixel មួយ។

AL = pixel color	MOV AL, pixel color
CX = column	MOV CX ,column
DX = row	MOV DX , row

✍**ឧទាហរណ៍:**

```
mov al, 13h    ; ផ្ទុកតម្លៃ pixel:13H ចូលទៅក្នុង AL register
mov ah, 0      ; ផ្ទេរតម្លៃ 00H ចូលទៅក្នុង AH register
int 10h        ; set graphics video mode.
mov al, 1100b ;ផ្ទេរតម្លៃ 1100H ចូលទៅក្នុង Al register
mov cx, 10     ; ផ្ទេរតម្លៃ 10 ចូលទៅក្នុង CX register
mov dx, 20     ; ផ្ទេរតម្លៃ 20H ចូលទៅក្នុង dx register
mov ah, 0ch    ; ផ្ទេរតម្លៃ 0cH ចូលទៅក្នុង AH register
int 10h        ; set pixel.
mov ah, 00
int 0x16
```

ឧទាហរណ៍: ចូរសរសេរកូដដើម្បីគូររូបចតុកោណកែងមួយដែលមាន Width(បណ្តោយ) = 100, កំពស់(Height) = 50 និងមាន row = 16, Col = 13។

5.1.9. Int 10H , Function 09H: បង្ហាញ Character ASIII និង ដាក់ពណ៌របស់វានៅទីតាំងណាមួយនៃការកំណត់ Cursor។

AL = Character ASII to display	MOV AL , ASCII
BH = Page Number	MOV BH , Page Number
BL = Attribute	MOV BL , Atribute
CX = number of time to write character ASII	

5.1.10. Int 21 H , Function 2A H: ប្រើសម្រាប់បង្ហាញអំពី Current Date ពីក្នុង System:

```
Input :   AH  =  2AH
Output :  AL = Day of week ( 0 = Sunday , 1 = Monday ,... )
          CX = Year            ;  mov cx, year
          DH = Month           ;  mov dh , month
          DL = Day of month    ;  mov dl , day of month
```

5.1.11. Int 21 H , Function 2BH: ធ្វើការកំណត់ Current Date ពីក្នុង System:

```
Input :   AH  =  2AH
          AL = Day of week ( 0 = Sunday , 1 = Monday ,... )
          CX = Year            ; mov cx, year
          DH = Month           ; mov dh, moth
          DL = Day of month    ; mov dl , day of month
```

5.1.12. Int 21 H , Function 2CH: ប្រើសម្រាប់បង្ហាញ Current Time ពីក្នុង System:

CH = hour (0...23)	MOV CH , hour ; CH = ម៉ោង
CL = minute (0...59)	MOV CL , minute ; CL = នាទី
DH = second (0..59)	MOV DH , second ; DH =វិនាទី
DL = hundredth of second(0..99)	MOV DL , hundredth of second

ឧទាហរណ៍: ចូរសរសេរកម្មវិធីដើម្បីបង្ហាញថ្ងៃ និងម៉ោងដែលមានទំរង់ដូចខាងក្រោម៖

Week Day : Friday 20-06-2009

Time : 04:40:45

5.1.13. Int 10H , Function 01H: ធ្វើការកែប្រែទំហំ Cursor រវាង Top និង Button Scan

- 0 : 14 សម្រាប់ VGA	Input : AH = 01 H
- 0 : 13 MDA និង EGA	CH = Start Scan Line
- 0 : 7 CGA	CL = End Scan Line
លេខ 0 គឺជា Top	Int 10 H
លេខ 14 , 13 និង 7 គឺជា Button	Output : No operation

ឧទាហរណ៍: កម្មវិធីគំរូខាងក្រោមធ្វើការបិទ ឬ បើក Cursor ដែលហៅថាBlinking Text Cursor

```
; hide blinking text cursor:
mov ch, 32 ; ផ្ទុកតម្លៃ 32H ចូលទៅក្នុង CH register
mov ah, 1 ; ផ្ទុកតម្លៃ Function :01H ចូលទៅក្នុង AH register
int 10h
; show standard blinking text cursor:
mov ch, 6 ; ផ្ទុកតម្លៃ 08H ចូលទៅក្នុង CH register
mov cl, 7 ;  ផ្ទុកតម្លៃ 07H ចូលទៅក្នុង CL register
mov ah, 1 ; ផ្ទុកតម្លៃ Function :01H ចូលទៅក្នុង AH register
int 10h
; show box-shaped blinking text cursor:
mov ch, 0 ; ផ្ទុកតម្លៃ 00H ចូលទៅក្នុង CH register
mov cl, 7 ; ផ្ទុកតម្លៃ 07H ចូលទៅក្នុង CH register
mov ah, 1 ; ផ្ទុកតម្លៃ Function :01H ចូលទៅក្នុង AH register
int 10h
```

5.1.14. Int 10H , Function 00H: ប្រើសម្រាប់កំណត់ផ្ទៃ Windows នៅលើ Screen

AL = Page Number	MOV AL , Page number
BH = Color Effect	MOV BH , Color effect
CX = Starting Row : Column	MOV CX, Row:Col

5.1.15. INT 10h និង AH = 13h: ធ្វើការបង្ហាញតួអក្សរនៅលើអេក្រង់ម៉ូនីទ័រ។

AL = write mode:

bit 0: update cursor after writing;

bit 1: string contains attributes.

BH = page number.

BL = attribute if string contains only characters (bit 1 of AL is zero).

CX = number of characters in string (attributes are not counted).

DL,DH = column, row at which to start writing.

ES:BP points to string to be printed.

ឧទាហរណ៍: ប្រូក្រាមគំរូខាងក្រោមធ្វើការបង្ហាញតួអក្សរនៅលើអេក្រង់ម៉ូនីទ័រ

```
mov al, 1; ផ្ទុកតម្លៃ 01H ចូលទៅក្នុង AL register
mov bh, 0 ; ផ្ទុកតម្លៃ 00H ចូលទៅក្នុង BH register
mov bl, 0011_1011b
mov cx, msg1end - offset msg1 ; calculate message size.
mov dl, 10 ; ផ្ទុកលេខ 10H ចូលទៅក្នុង AH register
```

```
mov dh, 7 ; ផ្ទុកលេខ 07H ចូលទៅក្នុង AH register
push cs ; ផ្ទុកលេខពីក្នុង CS register ទៅកាន់ផ្នែ Stack
pop es ; ផ្ទុកលេខពីក្នុង Stack ទៅកាន់ផ្នែ  CS register
mov bp, offset msg1
mov ah, 13h  ផ្ទុកលេខ 13H ចូលទៅក្នុង AL register
int 10h
jmp msg1end
msg1 db " hello, world! " ;declares string "hello word"
msg1end
```

5.1.16. INT 21h និង AH=5 នាំតួអក្សរមួយទៅឱ្យ Printer រួចធ្វើការបោះពុម្ពវា។ ដើម្បីធ្វើការងារនេះ អក្សរនេះត្រូវកំណត់ឱ្យ DL=character to print, after execution AL=DL.

ឧទាហរណ៍:

```
mov ah , 5 ;  ផ្ទេរលេខ 05H ចូលទៅក្នុង AH register
mov dl , 'a' ; ផ្ទេរលេខ ASCII=41H ចូលទៅក្នុង DL register
int 21h
```

5.2. Keyboard Features

5.2.1. Read a Character

Int 16H និង Function 00H ធ្វើការបញ្ចូលតួអក្សរទៅផ្ទុកនៅក្នុង AL ជាតម្លៃលេខកូដASCII និងតម្លៃក្នុង AH ជាស្កែនកូដ។

AH = value of function ជាអនុគមន៍ប្រើសម្រាប់ចាប់តលៃជាក់ក្នុង AL ពេលចុច Key

int 0x16 ;ធ្វើការចាប់ស្កែនកូដ និង ASCII។ សូមមើលតារាងកូដនៅក្នុងមេរៀនទី១

ឧទាហរណ៍: MOV AH, 00 ; ជាអនុគមន៍ SCAN តម្លៃ។ សូមមើលតម្លៃកូដជា

INT 0X16 ; ហៅអនុគមន៍ AH = 00 ដំណើរការចាប់តម្លៃចូល AL

- ពេលចុច ESC key នោះនាំឱ្យតម្លៃក្នុង AH =01h និង AL =1BH
- ពេលចុច Enter key តម្លៃក្នុង AH=1Dh និង AL =0DH
- ពេលចុច Down arrow នោះនាំឱ្យតម្លៃក្នុង AH =50h និង AL =00H
- ពេលចុច Up arrow key តម្លៃក្នុង AH =… និង AL = 48H
- ពេលចុច a នោះនាំឱ្យតម្លៃក្នុង AH =…. និង AL =48H

```
org  100h
; print welcome message:
mov dx, offset msg
mov ah, 9
int 21h
xor bx, bx ; zero bx register.
wait:  mov ah, 0   ; wait for any key....
'Esc' to stop...", 0Dh,0Ah, "$"
msg2 db 0Dh ,0Ah, "keypresses: $"
    print_ax proc ;names of procedure
    cmp ax, 0 ;compares ax with 0
     mov al, '0' ; al =30h
     mov ah, 0eh ; ah= 0eh
     int 10h ; invoke int 10h
```

```
int 16h
cmp al, 27  ; if key is 'esc' then exit.
je stop
mov ah, 0eh ; print it.
int 10h
inc bx ; increase bx on every key press.
jmp wait
; print result message:
      stop:  mov dx, offset msg2
      mov ah, 9
      int 21h
       mov ax, bx
       call print_ax
       ; wait for any key press:
       mov ah, 0
       int 16h
       ret ; exit to operating system.
       msg db "I'll count all your
       keypresses. press
      pop ax ;gets the value from stack
      ret
print_ax_r:
      pusha ; push the value to stack
      mov dx, 0 ; dx = 0
      cmp ax, 0 ;compares ax, 0
      je pn_done ; if dest = sou
      mov bx, 10 ; bx =10
      div bx    ;ax = ax / bx
      call print_ax_r
      mov ax, dx
      add al, 30h
      mov ah, 0eh
      int 10h
      jmp pn_done
      pn_done:
      popa
     ret
endp
```

5.2.2. Display a character

Int 10H និង Function 0EH ប្រើសម្រាប់យកតួអក្សរតែមួយពីក្នុង AL register ទៅបង្ហាញនៅលើអេក្រង់បញ្ចេញលទ្ធផល។

AH = value of function

int 0x10

ឧទាហរណ៍:

mov al, 'a' ; ចំលងតម្លៃ 41H ចូលទៅក្នុង AL register

mov ah, 0eh ; ចំលងតម្លៃ 0eH ចូលទៅក្នុង AH register

int 10h ; ហៅតម្លៃក្នុង AL register ទៅបង្ហាញនៅលើអេក្រង់បញ្ចេញលទ្ធផល

5.2.3. Determine Whether a Character

Int 16H និង Function 01H អនុញ្ញាតឱ្យប្រើ Key ចំនួន៨៣ key ឬលើសពីនេះនៅលើ Keyboard និងធ្វើការផ្ទុកវានៅក្នុងអង្គចងចាំ Keyboard(Buffer Keyboard)។

AH = value of function គឺប្រើតម្លៃអនុគមន៍លេខ 01 ដើម្បីធ្វើការប្រើ Key នៅលើ Keypad

int 0x16

➢ សំគាល់

ZF = 1 if keystroke is not available. បើ keystroke មិនអាចធ្វើទៅបាន

ZF = 0 if keystroke available. បើ keystroke ធ្វើទៅបានតាមការកំណត់

AH = BIOS scan code. ប្រើលេខជា scan code ព្រោះ Keyboard ប្រើលេខនេះ

AL = ASCII character.

(if a keystroke is present, it is not removed from the keyboard buffer).

ឧទាហរណ៍:

```
org     100h
; print a welcome message:
mov dx, offset msg
mov ah, 9
int 21h
;============================
; eternal loop to get
; and print keys:
wait_for_key:
; check for keystroke in
; keyboard buffer:
 mov     ah, 1
 int     16h
 jz      wait_for_key
; get keystroke from keyboard:
mov     ah, 0
int     16h
; print the key:
mov     ah, 0eh
int     10h
; press 'esc' to exit:
cmp     al, 1bh
jz      exit
jmp     wait_for_key
exit:
ret

      msg  db  "Type anything...",  0Dh, 0Ah
           db "[Enter] - carriage return.", 0Dh,0Ah
           db "[Ctrl]+[Enter] - line feed.", 0Dh , 0Ah
           db "You may hear a beep", 0Dh , 0Ah
           db "    when buffer is overflown.", 0Dh,0Ah
           db "Press Esc to exit.", 0Dh,0Ah , "$"
```

5.2.4. Inputting A Single Character of data from Keyboard

INT 21H , Function 08H ប្រើសម្រាប់បញ្ចូលមួយតួអក្សរពី Keyboard ទៅផ្ទុកនៅក្នុង AL

- Input : AH = 08 H
- Output : AL = Character ASCII

```
mov ah, 08h
mov char , al
```

ឧទាហរណ៍: សរសេរកូដដើម្បីបញ្ចូលអក្សរ "A" ទៅក្នុង AL Register។

☞ **EMU8086 Program**

```
ORG 100h
      MOV  AH , 08
INT  21
RET
```

☞ **Practice**

Press **A**

💻**OUTPUT**

AL = 41H

5.2.5. ការបញ្ចូល String ពី KEYBOARD

INT 21H និង Function 0AH ប្រើសម្រាប់នាំតួអក្សរពី Keyboard ទៅផ្ទុកនៅក្នុង Memory

- Input : AH = 0A H

  DX = Offset Address of data គឺជាអាសយដ្ឋានចាប់ផ្តើមនៃទិន្នន័យ

- Output : No

ឧទាហរណ៍: ចូរសរសេរកូដដើម្បីបញ្ចូលតួអក្សរ "abcd" ទៅផ្ទុកនៅ OA =0120H នៃ Memory

☞EMU8086 Program

```
ORG 100H ;កំណត់លេខអាសយដ្ឋានដំបូងឱ្យកម្មវិធីដោយចាប់ផ្តើមពីលេខ 100ទៅ
.DATA ; ជា Directive មួយប្រើសម្រាប់ឱ្យយើងប្រកាសអថេរ
      BUFFER  DB 300, ?, 300 DUP(" ");ប្រកាសអថេរ BUFFER ទទេរចំនួន ៣០០
.CODE; ជា Directive មួយប្រើសម្រាប់ឱ្យយើងបញ្ចូលកូដ
   MOV  AH , 0AH ;ចំលងលេខ Function :02H ចូលទៅក្នុង AH register។
   MOV  DX , BUFFER ; ចំលងលេខ Address "BUFFER ចូលទៅក្នុង AH   ;register
   INT 21 H ; ហៅជាតុនៃអនុគមន៍មកធ្វើការប្រតិបត្តិ
RET
```

- abcd ↵

DS: 1200 0A **04** ***61 62 63 64*** **លេខនិមួយៗជា** ***ASCII*** **កូដនៃអក្សរ** ***a, b, c, d***

1201H = 04, Dos requires sizes of the buffer

1200H = 0AH Keyboard was activated

1202H – 1205 : ASCII Dataa

☞**សំគាល់**: ពេលប្រើ INT 0x21 និង Func: 0AH យើងបានតម្លៃជា Byte និង Data:

Bytes	data

ឧទាហរណ៍: កម្មវិធីគំរូខាងក្រោមប្រើ INT 16 , Fuction 00 រួចប្រើ KEY: Down Arrow, Up Arrow, ESC, 1. 2 និង Enter រួមនឹង INT ផ្សេងៗទៀត ដើម្បីនាំ Highlight លើ Menu:

```
TITLE  EXAMPLE_SELMU111( EXE ) Select item from menu
include "emu8086.inc"
;.....................................................
   .MODEL      SMALL;Model Tiny
   .STACK 64
;.....................................................
   .DATA
;.....................................................
TOPROW        EQU 00
BOTROW        EQU 04
```

```
LEFCOL          EQU 20
COL     DB 00
ROW     DB 00
COUNT DB  ?
ATTRIB DB  ?
NINTEEN DB 19
MENU  DB  0C9H, 17  DUP( 0CDH ), 0BBH
          DB  0BAH, '1.Incomes        ', 0BAH
          DB  0BAH, '2.Expenses       ',0BAH
          DB  0BAH, 'Press esc to exit', 0BAH
          DB  0C8H, 17  DUP ( 0CDH ), 0BCH
PROMPT DB 09,09 ,"Press 1 to select menu 1",13,10
            DB 09,09, "Press 2 to select menu 2",13,10
            DB 09,09, "Press 3 to select menu 3","$"
define_scan_num
define_print_string
define_print_num
define_print_num_uns
;=================================================
   .CODE
          MAIN PROC     FAR
                    MOV AX , @data
                    MOV DS , AX
                    MOV ES , AX
                 CALL Q10CLEAR
                    MOV ROW,BOTROW + 3
                    MOV CL,00
                    CALL Q20CURS
                    CALL DISP_STR
          A10LOOP:
                    CALL B10MENU
                    MOV COL , LEFCOL + 1
                    CALL Q20CURS
                    MOV ROW , TOPROW + 1
                    MOV ATTRIB , 1FH
                    CALL D10DISP
;.....................................................
;Set menu line to normal / highlight
;.....................................................
D10DISP PROC          NEAR
          MOV AL , row
          MUL NINTEEN
          LEA SI , MENU +1
          ADD SI , AX
          MOV COUNT , 17
D20:
          CALL Q20CURS
          MOV AH,09H
          MOV COL,  LEFCOL + 1
          CALL Q20CURS
;Hide text cursor
          mov ch, 32
```

```
        CALL C10INPUT
        CMP AL , 0DH
        JE  A10LOOP
        MOV AX,4C00H
        INT 21H
MAIN ENDP
;=========================
;            Display full menu
;-----------------------------------------
B10MENU PROC        near
        MOV ROW , TOPROW + 1
B20:
        MOV COL , LEFCOL + 1
        CALL Q20CURS
        MOV AH , 09H
        MOV AL , 0DBH
        MOV BH , 00
        MOV BL , 04H
        MOV CX,19
        INT 10H
        INC ROW
        CMP ROW , BOTROW +2
        JNE B20
        MOV ROW , TOPROW
        LEA SI , MENU
        MOV ATTRIB , 71H
B30:
        MOV COL , LEFCOL
        MOV COUNT , 19
B40:
        CALL Q20CURS
        MOV AH , 09H
        MOV AL, [SI]
        MOV BH , 00
        MOV BL , ATTRIB
        MOV CX , 01
        mov ah, 1
        int 10h
        RET
D10DISP ENDP
;----------------------------------------
;       Accept input for request
;.....................................
C10INPUT    PROC  near
        MOV AH , 00H
        INT 16H
        CMP    AH,50H    ;    AH    =50H
downarrow key
        JE C20
        CMP AH , 48H ; AH = 48H uparrow
key
        JE C30
        CMP AL,0X31   ; AL=31H PRESS
        JE case1
        CMP AL , 0DH
        JE C90
        CMP AL , 1BH
        JE C90
C20:MOV ATTRIB , 71H
        CALL D10DISP
        INC ROW
        CMP ROW , BOTROW-1
        JBE C40
        MOV ROW , TOPROW+ 1
        JMP C40
C30:
        MOV ATTRIB , 71H
        CALL D10DISP
        DEC ROW
        CMP ROW , TOPROW + 1
        JAE C40
        MOV ROW , BOTROW - 1
```

```
                INT 10H
                INC COL
                INC SI
                DEC COUNT
                JNZ B40
                INC ROW
                CMP ROW , BOTROW + 1
                JNE B30
                RET
        B10MENU ENDP
        MOV AL, [SI]
                MOV BH,00
                MOV BL , ATTRIB
                MOV CX, 01
                INT 10H
                INC COL
                INC SI
                DEC COUNT
                JNZ D20
C40:
        CALL Q20CURS
        MOV ATTRIB , 1FH
        CALL D10DISP
        JMP C10INPUT
C90:CALL Q10CLEAR
   RET
C10INPUT ENDP
case1 proc
; Show text cursor
        MOV AH,1
        MOV CH,6
        MOV CL,7
        INT 0X10
        MOV ROW,3
        MOV COL,20
        call Q20CURS
        CALL Q10CLEAR
print "Enter rent house( byte )= " call print_string ;and
call    scan_num
mov al,cl
printn ""
gotoxy 20,5
print "Enter Salary( byte )= " call print_string ;and
 call    scan_num
    printn ""
    add cl,al
    gotoxy 20,7
 print "Total= "
    mov     ax, cx
    call    print_num
    mov ah,06
    mov dl,"$"
    int 0x21
      MOV AH,00
      MOV BH , 60H
      MOV CX, 0000
      MOV DX, 309EH
      INT 10H
  RET
  Q10CLEAR ENDP
  ;.....
  ;       Set cursor col : row
  ;.....
  Q20CURS PROC      NEAR
        MOV AH , 02H
```

```
        INT 0X16
        mov row,BOTROW+2
        mov col,00
        call Q20CURS
        call Q10CLEAR
      CALL DISP_STR
      JMP A10LOOP
case1 endp
;.....
;     Clear screen
;.....
Q10CLEAR   PROC NEAR
      MOV AX , 0600H
      MOV BH , 00
      MOV DH, ROW
      MOV DL , COL
      INT 10H
      RET
 Q20CURS ENDP
DISP_STR   PROC
 NEAR
    MOV AH,09
      LEA DX,PROMPT
      INT 0X21
      RET
 DISP_STR ENDP
      END MAIN
```

5.3. ការប្រើប្រាស់ Macros

វាមានប្លុកកូដជាច្រើននៅក្នុងភាសា Assembly ដែលជាសំនុំពាក្យបញ្ជាសម្រាប់ធ្វើអ្វីមួយជាក់លាក់ ហើយដែលត្រូវប្រើក្នុងទំរង់ដដែលៗ។ ឧទាហរណ៍ INT 21H និង Function 09H មានមុខងារបង្ហាញទិន្នន័យជាតួអក្សរនៅលើអេក្រង់។ ដូច្នេះដើម្បីកាត់បន្ថយពេលវេលាក្នុងការសរសេរកម្មវិធីដដែលៗ នោះត្រូវប្រើ Macro។ ហើយការប្រើវាយើងបានប្រើប្រាស់រួចមកហើយក្នុងការបង្កើតកម្មវិធី GUI។

វាអនុញ្ញាតឱ្យអ្នកសរសេរការងារមួយដោយប្រើសំនុំនៃកូដដើម្បីធ្វើអ្វីមួយ។ តែម្តងគត់ហើយអាចហៅវាមកប្រើនៅពេលត្រូវការ។ ខាងក្រោមនេះជាទំរង់ទូទៅនៃការសរសេរ Macro:

```
name MACRO [parametes,...] ; ដាក់ឈ្មោះដែលចង់បង្កើត , Directive និងប៉ារ៉ាម៉ែត្រវា
    LOCAL LABEL1 ,LABEL2,... ; កំណត់ទីតាំងជា Local Lebel1, 2, 3, …, n
        INSTRUCTIONS      ; ជាសំនុំកូដសម្រាប់ធ្វើប្រមាណវិធីជាក់លាក់មួយ
ENDM ជា Directive មួយប្រើសម្រាប់ធ្វើការបញ្ចប់ការសរសេរ MACRO
```

☺.**ឧទាហរណ៍**: កម្មវិធីគំរូខាងក្រោមប្រើ

```
    MyMacro    MACRO
      LOCAL label1, label2
      CMP  AX, 2 ; compares ax = 2 with2
      JE label1 ;if( ax = 2 )
      CMP  AX, 3 ;compares ax = 3with 3
      JE label2 ; if ax =3
    label1:
      INC  AX ; incresement by 1 in ax
abel2:
   ADD  AX, 2 ; ax = ax +2
ENDM ;end of macro
ORG 100h
      MyMacro
      MyMacro
RET
```

ការប្រើ Macros ក្នុងគោលបំណងចងក្រងផ្នែកនៃប្រូក្រាមដាក់នៅក្នុង Header file ដែលបានកំណត់នូវផ្តល់តម្លៃជាលេខ ឬ ពីអថេរមួយទៅឱ្យប៉ារ៉ាមែត្រសម្រាប់ប្រើប្រាស់ក្នុងប្រូក្រាម និងប្រើជាឈ្មោះ Locals បានច្រើនទៀតក្នុង Macro មួយ។ ទំរង់ទូទៅខាងក្រោមសម្រាប់បង្កើតឈ្មោះ Functions នៅក្នុង Header File មួយ៖

```
; register to stack
regsave macro
push ax
push bx
push cx
push dx
push di
push si
push bp
pushf
endm
```

```
;stack to register
        regresto macro
        popf
        pop bp
        pop si
        pop di
        pop dx
        pop cx
        pop bx
        pop ax
```

ឧទាហរណ៍: កម្មវិធីគំរូខាងក្រោមបង្កើតអនុគមន៍ cls( ) , print( ) ,print.hex.byte( ) ដោយប្រើ Head File ឈ្មោះ include "sysnum.inc" នៅក្នុង Microprocessor Assembler។ ការបង្កើតអនុគមន៍ចងទៅជាប្តកកូដមួយងាយស្រួលប្រើ និងមានទំរង់ខ្លីៗ។

```
cls macro
    mov ax,0600h
    mov bh,07
    mov cx,0000
    mov dx,0x184f
    int 10h
endm
print macro string
    mov ah,09
    mov dx,offset string
    int 21h
endm
print.hex.byte macro
    ;if cf=1
     jnc next
     mov ah,6
     mov dl,0x31
        lea si,mem
        mov dl, [si]
        mov al, dl
        shr al, 4    ; leave high part only.
        xlat         ; get hex digit.
        mov ah, 0eh  ; teletype sub-function.
        int 10h
        mov al, dl
        and al, 0fh  ; leave low part only.
        xlat         ; get hex digit.
        mov ah, 0eh  ; teletype sub-function.
        int 10h
        mov [si],al
        ; wait for any key press:
        mov ah, 0
        int 16h
        ret
```

int 0x21 next:	table db '0123456789ABCDEF' endm

ការបង្កើតអនុគមន៍ cls , print ,print.hex.byte យើងអាចយកវាទៅប្រើក្នុងប្រូក្រាម X86Asm និង Emu8086 ដូចដែលយើងប្រើអនុគមន៍ PrintDec( ), PrintHex( ), printf( ), scanf( ) ជាដើម។

5.4. FIELS FEATURES

ទិន្នន័យផ្ទុកនៅក្នុងឌីសមានដូចជា Hard disk និង Diskette ហើយមានឈ្មោះផ្សេងៗពីគ្នាជាទំរង់នៃ File មួយ។ ទិន្នន័យផ្ទុកនៅក្នុងឌីសជាស៊េរីនៃ Bytes ដែលបានតំរៀបតាមលំដាប់លំដោយដែលកម្មវិធីជាអ្នករៀបចំ។

byte A	byte1	byte2	byte3	...	byten

- floppy disk វាជា Second Storage មាន sides ចំនួនពីរ និងមាន 2 heads
- side និមួយៗ មាន 80 cylinders ដោយមានលេខគោលដប់ពី **0..79**
- cylinder និមួយៗមានចំនួនទាំងអស់ 18 sectors ពីលេខផ្នែកទីមួយដល់ផ្នែកទី ១៨
- sector និមួយៗ **512** bytes។ ដូច្នេះទំហំសរុបគឺ 2 x 80 x 18 x 512 = **1,474,560** bytes

នៅក្នុងចំនុចនេះគឺប្រើ INT 21H និង ប្រើលេខ Function តាមតារាងដូចខាងក្រោមនេះសម្រាប់៖

3CH : Create File បង្កើត file 3DH :Open File បើក File 3EH : Close File បិទ File 3FH : Read Record អាន File	40H : Write Record ; សរសេរ File 42H : Move File Pointer AH= 39h Create Directory **បង្កើត** Directory AH= 3Ah : Remove Directory **លុប** Directory

គួរកត់សំគាល់ដែរថាការប្រើ Files Handles គឺប្រើជាមួយប្រភេទជា ASCII ដែលអាចកំណត់បានទីតាំង DRIVE, DIRECTORY PATH, និង FILENAMES សម្រាប់ផ្ទុកធាតុរបស់ Files មួយ។

ឧទាហរណ៍1:

```
PATHNAME1 DB "C:\VOTHEARNEAR.TXT", 00H
```

ឧទាហរណ៍2:

```
PATHNAME1 DB "C:\MYFOLDER\VOTHEARNY.TXT", 00H
```

5.4.1. **INT 21h** និង **AH= 3Ch**: ធ្វើការបង្កើតឈ្មោះ Files ស្ថិតនៅទីតាំងកំណត់មួយ។ ទំរង់ខាងក្រោមនេះជាគ្រឹះសម្រាប់បង្កើត File មួយ។

CX = file attributes:
mov cx, 0 ; normal - no attributes.
mov cx, 1 ; read-only.
mov cx, 2 ; hidden.
mov cx, 4 ; system
	mov cx, 7 ; hidden, system and read-only!
	mov cx, 16 ; archive
	DS:DX -> ASCIIZ filename.
	returns:

☺**ឧទាហរណ៍**: កម្មវិធីគម្រោងខាងក្រោមធ្វើការបង្កើតឈ្មោះ Files ស្ថិតនៅទីតាំងកំណត់មួយ។

```
org 100h
        jmp code:
        filename db "c:\myfile.txt",
0
        handle dw  ?
        code:
                mov ah, 3ch
                mov cx, 0
```

```
mov dx, offset filename
        mov ah, 3ch
        int 21h
        jc err
        mov handle, ax
        err:
        ; ....
        ret
```

5.4.2. INT 21h / AH= 40h: សរសេរ File មួយសម្រាប់ធ្វើការបញ្ចូលទិន្នន័យ។

BX = file handle.

CX = number of bytes to write.

DS:DX -> data to write.

return:

CF clear if successful; AX = number of bytes actually written.

CF set on error; AX = error code.

5.4.3. INT 21h / AH= 3Eh: ពេលយើងបានបញ្ចប់ការសរសេរ File មួយនោះត្រូវបិទវា។

Entry: BX = file handle

Return:

CF clear if successful, AX destroyed.

CF set on error, AX = error code (06h).

☺**ឧទាហរណ៍**: ខាងក្រោមនេះជាប្រូក្រាមមួយប្រើសម្រាប់សរសេរអត្ថបទមាន ១០០តួចូលទៅក្នុង

```
C:\em8086\vdrive\c\ vothearny.txt
 ;PROGRAM : TEST FILES
;WRITE BYTE :OUK POLYVANN
;DATE :20-06-2009
org  100h
jmp start
s1       db 100, ?, 100 dup(' ')
; file handle:
```

```
        xor cx, cx
    mov cl, s1[1]
    int 21h
    jc  error
; close file

        mov bx, handle
        mov ah, 3eh
```

```
handle   dw   0
; set segment registers to code:
start:  mov ax, cs
        mov ds, ax
        mov es, ax
jmp m1
msg1    db  "enter any text: $"
m1:     mov dx, offset msg1
        mov ah, 9
        int 21h
; input a string:
        mov dx, offset s1
        mov ah, 0ah
        int 21h
jmp m2
pathname        db    "d:\emu8086\vdrive\c\
 vothearny.txt",00
m2:
; create new file:
        mov cx, 0
        mov ah, 3ch
        mov dx, offset pathname
        int 21h
        jc  error
        mov handle, ax
; write buffer to file:
        mov ah, 40h
        mov bx, handle
        mov dx, offset s1+2
```

```
        int 21h
        jc error
        jmp next
  ; print message complete:
  next:  jmp m5
 msg5      db   0Dh,0Ah,"save complete",
0Dh,0Ah,'$'
        m5:     mov dx, offset msg5
                mov ah, 9
                int 21h
        jmp ok:
  error:  jmp m6
msg6    db  0Dh,0Ah,"The select disk drive
in not in use",0Dh,0Ah
            db    0Dh,0Ah,"Check to make
  sure a disk is inserted",0Dh,0Ah,'$'
        m6:     mov dx, offset msg6
                mov ah, 9
                int 21h
  ok:
  jmp m4
  msg4          db    0Dh,0Ah,"press any
  key...",0Dh,0Ah,'$'
  m4:     mov dx, offset msg4
        mov ah, 9
        int 21h
        mov ah, 0
        int 16h
        mov ah, 4ch
        int 21h
```

5.4.4. INT 21h / AH= 3Dh: ប្រើសម្រាប់ធ្វើការបើក File ដែលមានស្រាប់។

Entry:	Return:

AL = access and sharing modes:
mov al, 0 ; read
mov al, 1 ; write
mov al, 2 ; read/write
DS:DX -> ASCIZ filename.
	CF clear if successful, AX = file handle.
	CF set on error AX = error code.
	note 1: file pointer is set to start of file.
	note 2: file must exist.

ឧទាហរណ៍: ប្រូក្រាមខាងក្រោមនេះជាប្រូក្រាមមួយប្រើសម្រាប់បើកសេចក្តីរបស់ Files ក្នុង C:\ em8086\ vdrive\c\ vothearny.txt

```
org  100h
jmp start
s1        db 200, ?, 200 dup(' ')
t2        db 100, ?, 100 dup(' ')
filename  db 30, ?, 30 dup(' ')
; file handle:
handle   dw   0
path    db "c:\emu8086\vdrive\c\vothearny.txt",00
start:
```

```
; open existing file:
        mov al, 0     ; read.
        mov ah, 3dh
        mov dx, offset path
        int 21h
        jc  error
        mov handle, ax
; read bytes from file
        mov ah, 3fh
        mov bx, handle
        xor cx, cx
        mov cl, t2[0]
        mov dx, offset t2[2]
        mov ah, 3eh
        int 21h
        jc error
; set '$' to the end of the buffer:
        xor bx, bx
        mov bl, t2[1]        ; get actual size.
        mov t2[bx+2], '$'
; print the buffer:
        mov dx, offset t2[2] ; skip 2 control bytes.
        mov ah, 9
error:  jmp m5
        msg5    db  0Dh,0Ah,"error...",0Dh,0Ah,'$'
        m5:     mov dx, offset msg5
```

```
        int 21h
        jc error
        mov t2[1], al
        int 21h
jmp m3
        msg3  db  13,10,'Open
complete',13,10,"$"
        m3:mov ah,09
        mov dx,offset msg3
        int 0x21
        jmp ok
; print error message:
; close file
        mov bx, handle
```

```
        mov ah, 9
         int 21h
ok:
jmp m4
msg4  db  0Dh,0Ah,"press any key...",0Dh,0Ah,'$'
m4:     mov dx, offset msg4
        mov ah, 9
        int 21h
        mov ah, 0
        int 16h
; exit to the operating system:
        mov ah, 4ch
        int 21h
```

5.4.5. **INT 21h / AH= 42h**: កំណត់ទីតាំងឱ្យឈ្មោះ File ថ្មីៗ (current file position)

AL = origin of move: 0 - start of file. 1 - current file position. 2 - end of file.

BX = file handle.

CX:DX = offset from origin of new file position.

Return:

CF clear if successful, DX:AX = new file position in bytes from start of file.

CF set on error, AX = error code.

5.4.6. **INT 21h / AH= 3Fh:** អាន **file** មួយតាមចំនួន Bytes ដែលមាននៅក្នុង CX

BX = file handle ;CX = number of bytes to read. ;DS:DX ->buffer for data.

Return:

CF is clear if successful - AX = number of bytes actually read; 0 if at EOF (end of file) before call.

CF is set on error AX = error code.

☺ ឧទាហរណ៍:

```
org 100h
mov ah, 3ch
mov cx, 0
mov dx, offset filename
mov ah, 3ch
int 21h  ; create file...
mov handle, ax
mov bx, handle
mov dx, offset data
mov cx, data_size
mov ah, 40h
int 21h ; write to file...
mov al, 0
mov bx, handle
mov cx, 0
mov dx, 7
mov ah, 42h
int 21h ; seek...
mov bx, handle
mov dx, offset buffer
mov cx, 4
mov ah, 3fh
int 21h ; read from file...
mov bx, handle
mov ah, 3eh
int 21h ; close file...
ret
```

5.4.7. **INT 21h និង AH= 39h**: ប្រើសម្រាប់បង្កើតឈ្មោះ Directory ឬ Folder។

DS:DX -> ASCIZ pathname; zero terminated string, for

Return: CF clear if successful AX destroyed. CF set on error AX = error code.

Note: all directories in the given path must exist except the last one.

✍ ឧទាហរណ៍:

```
org 100h ; កំណត់លេខអាសយដ្ឋានដំបូងឱ្យកម្មវិធី ដោយឱ្យ IP =100h
mov dx, offset filepath ; កំណត់លេខលេខ offset addres នៃ File ចូល Dx
mov ah, 39h ;al = 39h ជាលេខអនុគមន៍បង្កើតឈ្មោះ Directory ឬ Folder
int 21h ; ធ្វើការប្រតិបត្តិបង្កើត Directory
ret
    filepath DB "C:\mydir", 0 ; path to be created.
end
```

5.4.8. **INT 21h និង AH= 3Ah**: លុប directory

Entry: DS:DX -> ASCIZ pathname of directory to be removed.

Return:

CF is clear if successful, AX destroyed CF is set on error AX = error code.

Notes: directory must be empty (there should be no files inside of it).

ឧទាហរណ៍:

```
org 100h;កំណត់លេខអាសយដ្ឋានដំបូងឱ្យកម្មវិធី ដោយឱ្យ IP=100h
mov dx, offset filepath; ; កំណត់លេខលេខ offset addres នៃ File ចូល Dx
mov ah, 3Ah ; ah=3Ah ជាលេខអនុគមន៍លុបឈ្មោះ Directory ឬ Folder
int 21h; ធ្វើការប្រតិបត្តិលុប Directory
ret
    filepath DB "C:\mydir", 0          ; path to be created.
end
```

5.4.9. **INT 21h និង AH= 3Bh**: ប្រើសម្រាប់កំណត់ current directory.

Entry: DS:DX -> ASCIZ pathname to become current directory (max 64 បៃ)

Return:

Carry Flag is clear if successful, AX destroyed.

Carry Flag is set on error AX = error code.

Notes: even if new directory name includes a drive letter, the default drive is not changed,

only the current directory on that drive.

5.4.10. **INT 21h** / **AH= 19h**: ចាប់យក current default drive.

Return: AL = drive (0=A:, 1=B:, etc)

5.4.11. **INT 21h / AH= 56h**: ប្រើសម្រាប់ប្តូរឈ្មោះ File ឬ Directory មួយដែលមានរួច ហើយឱ្យទៅជាឈ្មោះថ្មី។

Entry:

DS:DX -> ASCIZ filename of existing file.

ES:DI -> ASCIZ new filename.

Return:

CF clear if successful.

CF set on error, AX = error code.

Note: allows move between directories on same logical drive only; open files should not be renamed!

ឧទាហរណ៍:

```
OLDSTR DB "D:\OLDPATH\OLDNAME",00H
NEWSTR DB "D:\NEWPATH\NEWNAME, 00H
...
```

```
MOV AH, 56H
LEA DX, OLDSTR
LEA DI, NEWSTR
INT 21H
```

5.4.12. **INT 21h / AH= 41h**: លុប file (unlink).

Entry:

DS:DX -> ASCIZ filename (no wildcards, but see notes).

return:

CF clear if successful, AX destroyed. AL is the drive of deleted file (undocumented).

CF set on error AX = error code.

Note: DOS does not erase the file's data; it merely becomes inaccessible because the FAT chain for the file is cleared deleting a file which is currently open may lead to file system corruption

ឧទាហរណ៍:

```
ASCstr          db 'd:\sarin.txt',00h
...
mov ah, 41h
lea dx , ASCstr
int 0x21
```

5.5. ការប្រើ BIOS និង DOS INTERRUPT ជាមួយនឹងភាសា C

យើងអាចបញ្ចូល INT របស់ BIOS និង DOS និងអនុគមន៍របស់វានៅក្នុងភាសា C / C++ ដោយមិនបានលំអិតពីចំណេះដឹងនៃការប្រើភាសា 80x86 Assembly ក៏បាន។ ប៉ុន្តែយើងអាចសរសេរកម្មវិធី ដោយប្រើអនុគមន៍ DOS ដោយហៅ INT 21H និង INT BIOS មកប្រើ , ដោយមានការជួយពី Compilers នៅក្នុងទំរង់ int86 និងអនុគមន៍ intdos។ អនុគមន៍ int86 គឺប្រើសម្រាប់ហៅទំរង់ផ្សេងៗនៃ PC's interrupts, និងអនុគមន៍ intdos មកប្រើជាមួយនឹង INT 21H DOS។

ដើម្បីប្រើការប្រើបញ្ជា int86 , យើងអាចកំណត់ឱ្យ Registers មានតម្លៃដែលចង់បាន ហើយទើបហៅ int86 មកប្រើ។ ហើយការប្រើវាយើងអាចដំណើរការ 8086 Registers។ នៅក្នុងករណីនេះ int8086 គឺប្រើដូចពាក្យបញ្ជា "INT #" នៅក្នុងភាសា 8086 Assembly ដែរ។ ហើយដើម្បីដំណើរការ 80x86 registers, យើងត្រូវប្រើរចនាសម្ព័ន្ធ union REGS ទើប C compilers យល់ និងស្គាល់បាន។

union REGS regin , ragout ;

- regin និង ragout គឺជាអថេរប្រើជាមួយនឹងអក្សរ x ឬ h និងឈ្មោះ Register នៃស៊ីភីយូ ៨០៨៦
- union REGS អនុញ្ញាតឱ្យ 80x86 registers ដំណើរការនៅក្នុងទំរង់ 8-bit និង 16-bit។

16-bit		8-bits	
C language	Assembly language	C language	Assembly language
regin.x.ax	AX	regin.h.al	AL
		regin.h.ah	AH
regin.x.bx	BX	regin.h.bl	BL
		regin.h.bh	BH
regin.x.cx	CX	regin.h.cl	CL
		regin.h.ch	CH
regin.x.dx	DX	regin.h.dl	DL
		regin.h.dh	DH
regin.x.si	SI		
regin.x.di	DI		
regin.x.cy	CY		

- ប្រើអក្សរ x តាងឱ្យទំហំ 16 – bit register មានដូច ax, bx, cx, dx ,si, di
- ប្រើអក្សរ h តាងឱ្យទំហំ 8 – bit register(al, ah, bl, bh, cl, ch, dl, dh)
- ចំពោះការប្រើអថេរ regin យើងត្រូវប្រកាស union REGS regin សម្រាប់ផ្ទុកតម្លៃ
- ចំពោះការប្រើ regout យើងត្រូវប្រកាស union REGS regout សម្រាប់បញ្ចេញតម្លៃ

➢ **ឧទាហរណ៍:** កូដគំរូខាងក្រោមគឺជាការប្រៀបធៀបរវាងភាសា C និង Assembly:

```
/* C Language                    Assembly Language */
union REGS regin , regout;
regin.h.ah = 0x25;                          mov ah, 0x25
regin.x.ax = 0x4567                         mov ax, 0x4567
regin.x.si = 0x1290                         mov si, 0x1290
int86( int# ,&regin,&regout );              int #
```

➢ **ឧទាហរណ៍**: ប្រើពាក្យបញ្ជា int86 ដើម្បីសំអាត់ធ្វើការអេក្រង់។ ដើម្បីធ្វើការងារនេះយើងត្រូវប្រើINT 10H និង Fuction :06H

```
main( ){
        union REGS regin ,regout; ប្រកាស regin និង regout សម្រាប់បញ្ចូលឬបញ្ចេញតម្លៃ
        regin.x.ax = 0x0600; កំណត់ឱ្យ ax ផ្ទុកតម្លៃ 0x0600 ដោយ AH =06 ជាអនុគមន៍
        regin.x.bx = 0x07 ; កំណត់ឱ្យ bx ផ្ទុកតម្លៃ  0x07
        regin.x.cx = 00;  កំណត់ឱ្យ cx ផ្ទុកតម្លៃ 00 ជាចំណុចចាប់ផ្តើមពីនៃការសំអាត
        regin.x.dx = 0x184F ; កំណត់ឱ្យ dx ផ្ទុកតម្លៃ 0x184F ជាតម្លៃចុងក្រោយនៃការសំអាត
        int86( 0x10,&regin,&regout ) ;}
```

- **ឧទាហរណ៍**: ការប្រើ៣ក្យបញ្ជា int86 សម្រាប់ប្រើប្រមាណវិធីផ្សេងៗដូចមានការរៀបរាប់ខាងក្រោម៖
    1. រក្សាទីតាំង cursor នៅចំណុចកំណត់មួយនៅលើម៉ូនីទ័រ
    2. កំណត់ទីតាំង cursor នៅជួរដេក row 12 និងជួរឈរ column 8
    3. ធ្វើការបង្ហាញឃ្លា "Hello World " នៅលើអេក្រង់ ដោយប្រើអនុគមន៍ printf
- **លំហាត់ទី១:** ចូរប្រើ INT 21H និង Func:2AH ដើម្បីបង្ហាញ date ជាទំរង់ dd-mm-yy
    ១. ការប្រើ intdos function
    ២. ការប្រើ int86 function
    ៣. ការប្រើ asm function
- **លំហាត់ទី២**: ប្រើ៣ក្យបញ្ជា int86 ជាមួយនឹង INT 21H សម្រាប់រកទំហំសរុបនៃ Memory ដែលបានប្រើនៅលើកុំព្យូទ័រ។
- .**លំហាត់ទី៣**: ប្រើ៣ក្យបញ្ជា int86 ជាមួយនឹង INT 33H សម្រាប់ធ្វើការជ្រើស Menu របស់កម្មវិធីមួយដែលបានសរសេរនៅក្នុងភាសា C

5.6. **លក្ខណៈនៃការប្រើ** MOUSE

ការប្រើបញ្ជា Int 33H និងលេខអនុគមន៍របស់ Mouse សម្រាប់ដំណើរការជ្រើសរើស (select) ចុចម្តង(click) និងចុចពីរដង(double click)ដោយមានលេខអនុគមន៍ដូចដែលបានរៀបរាប់ដូចខាងក្រោមនេះដោយប្រើជាមួយនឹង INT 33H:

00H	ជាតម្លៃលេខនៃអនុគមន៍សម្រាប់ឱ្យ Mouse ធ្វើការចាប់ផ្តើមដំបូង
01H	ធ្វើការបង្ហាញរូបរៀងនៃ mouse pointer នៅលើម៉ូនីទ័រ
02H	ធ្វើការលាក់ ឬបំបាត់ mouse pointer នៅលើម៉ូនីទ័រ
03H	ចាប់តម្លៃពេលធ្វើការចុច button status និងឆ្លោះទៅកាន់ទីតាំងកំណត់មួយ។
04H	ធ្វើការចង្អុលទៅកាន់ទីតាំងមួយ(pointer location)
05H	ប្រើប៊ូតុង និងចុចលើព័ត៌មាន (Get button – press information)
06H	ប្រើប៊ូតុង និងចុចលើព័ត៌មាន (Get button – release information)
07H	កំណត់ដែនកំណត់ផ្តេកសម្រាប់ Pointer(Set horizontal limits for pointer)
08H	កំណត់ដែនកំណត់បញ្ឈរសម្រាប់ Pointer(Set vertical limits for pointer)
09H	កំណត់ប្រភេទ Pointer ក្រាហ្វិក(Set graphics pointer type)
0AH	កំណត់ប្រភេទ Text ក្រាហ្វិក (Set text pointer type)
0BH	Read mouse-motion counters
0CH	Install interrupt handler for mouse events
0DH	Turn on light pen emulation
0EH	Turn off light pen emulation
0FH	Set mickey –to – pixel ratio

Mouse គឺប្រើជ្រើសរើស អនុវត្ត និងសម្រាប់ចង្អុលទៅទីតាំងមួយដែលចង់បាន ហើយវាអាចដំណើរការទៅបានត្រូវមានផ្នែកទន់ដែលហៅ Driver។ ខាងក្រោមនេះការប្រើអនុគមន៍ Mouse ជាមួយនឹង Int 33H សម្រាប់ធ្វើការងារផ្សេងគ្នា។

5.6.1. **Function00H**: Initialize the mouse

នេះគឺជាបញ្ជាទីមួយដែលធ្វើឱ្យប្រូក្រាមមួយអាចប្រើប្រាស់ Mouse មួយទៅបាន និងអាចប្រតិបត្តិតែម្តង។ ដើម្បីបានបញ្ជានេះត្រូវសរសេរឃ្លាបញ្ជា Mov ah, 00h។

5.6.2. **Function01H:** Display the mouse pointer

បន្ទាប់ពីបានប្រើលេខអនុគមន៍ 00H, នោះត្រូវប្រើអនុគមន៍លេខ 01H ដើម្បីធ្វើការបង្ហាញ Mouse Pointer នៅលើអេក្រង់។ នោះត្រូវសរសេរ Mov ax, 0001h និង INT 0x33h

ឧទាហរណ៍: ប្រូក្រាមគំរូខាងក្រោមបង្ហាញពីការប្រើលេខអនុគមន៍ 00H និង 01H រួមជាមួយបញ្ជា Int 33H សម្រាប់ធ្វើការចាប់ផ្តើម និងបង្ហាញ Mouse Pointer នៅលើអេក្រង់ៈ

☞ **ការប្រើប្រាស់** ***keyword : asm***

```
void main( ){
        asm{ mov ax,00 ; កំណត់តម្លៃ 0000H ចូលទៅក្នុង AX register
             int 33H ; ធ្វើការដំណើរការឱ្យស្គាល់ Mouse ដំបូង
            mov ax, 01H ; កំណត់តម្លៃ 01H ចូលទៅក្នុង AX register
            int 33H } ; ធ្វើការបង្ហាញ Mouse Pointer នៅលើអេក្រង់
```

☞ **ការប្រើប្រាស់** *int86*

```
void main( ){
        union REGS regin, regout;
        regin.x.ax = 0; កំណត់តម្លៃ 0000H ចូលទៅក្នុង AX register
        int86( int 0x33 , &regin , &regout ) ; ប្រើ int 33h និង &regin , &regout
        regin.x.ax =0 ; កំណត់តម្លៃ 0000H ចូលទៅក្នុង AX register
        int86( int 0x33 ,&regin , &regout ) };; ប្រើ int 33h និង &regin , &regout
```

5.6.3. **Function05H: ធ្វើចាប់តម្លៃពេលចុចប៊ូតុង**(Get button-release information)

លេខអនុគមន៍ Ax = 05h និង Int33h ប្រើសម្រាប់ចុច Mouse button, ដោយកំណត់នៅក្នុង BX ជាមួយលេខ 0= left, 1 = right, 2 =center.

```
MOV AX , 05H ; កំណត់តម្លៃ 0005H ចូលទៅក្នុង AX register
MOV BX , button number ; កំណត់តម្លៃ button number ចូលទៅក្នុង AX register
INT 33H ; ធ្វើការដំណើរការបញ្ជូលតំលៃប៊ូតុងចូលទៅកាន់ BX
```

5.6.4. **Function06H: ធ្វើចាប់តម្លៃពេលចុចប៊ូតុង**(Get button-release information)

ដើម្បីប្រើលេខអនុគមន៍នេះប្រើសម្រាប់ធ្វើការចុច Mouse button, ដោយកំណត់នៅក្នុង BX ជាមួយនឹងលេខ 0 = left, 1 = right, 2 =center។

```
MOV AX , 06H ; កំណត់តម្លៃ 0006H ចូលទៅក្នុង AX register
MOV BX , button number ; ; កំណត់តម្លៃ button number ចូលទៅក្នុង AX register
INT 33H ;ធ្វើការដំណើរការបញ្ជូលតំលៃប៊ូតុងចូលទៅកាន់ BX
```

5.6.5. **Function 02H: Concal the Mouse Pointer**

ប្រើប្រាស់សម្រាប់លុបបំបាត់ Mouse Pointer នៅចាកចេញពីដំណើរការកម្មវិធីមួយ។

MOV AX, 02H ; កំណត់តម្លៃ 0002H ចូលទៅក្នុង AX register

INT 33H ;ធ្វើការដំណើរការលុបបំបាត់ Mouse Pointer នៅពេលចាកចេញពីកម្មវិធី

5.6.6. Function 04H: Set Pointer Location

ប្រើ Function នេះសម្រាប់កំណត់ទីតាំង Mouse Pointer នៅលើ Screen ដោយមាន Code ដូចខាងក្រោម៖

MOV AX , 04H; កំណត់តម្លៃ 0004H ចូលទៅក្នុង AX register

MOV CX , horizontal-location; កំណត់តម្លៃ horizontal ចូលទៅក្នុង CX register

MOV DX , vertical-location; កំណត់តម្លៃ vertical ចូលទៅក្នុង DX register

INT 33H ;ធ្វើការដំណើរកំណត់នៅទីតាំងទៅតាមតម្លៃក្នុង CX និង DX

ឧទាហរណ៍:

```
void main( ){
        union REGS regin, regout; ប្រកាសអថេរ
        regin.x.ax =4;
        regin.x.cx=24
        regin.x.dx = 16
        int86( int 0x33 ,&regin , &regout ) }
```

5.6.7. Function 03H: Get button Status and Location

Function នេះប្រើសម្រាប់ចុច Buttons ហើយយកតម្លៃនេះត្រូវផ្ទុកនៅក្នុង BX Register និងចាប់យកតម្លៃនៃ Mouse Pointer ស្ថិតនៅលើ Screen មួយទៅ ផ្ទុកនៅក្នុង CX Register និង DX Register

☞ *Input*

- BX = Status Button
  - Bit 0 = No presssed
  - Bit 1 = Left
  - Bit 2 = Right
  - Bit 3 = Center

☞ *Output*

- CX = Horizontal (x) coordinate
- DX = Vertical (y) coordinate

EX:

```
void mousepos( int &xpos , int &ypos , int &click ){
    regin.x.ax = 3;
    int86( int 0x33 ,&regin , &regout )
    click = out.x.bx ;
    xpos = regout.x. cx;
    ypos = regout.x.dx }
```

5.6.8. **Function 07H: Set Horizontal Limits for pointer**

យើងអាចធ្វើប្រមាណវិធីដើម្បីកំណត់តម្លៃ Minimum និង Maximum horizontal ចំពោះ Pointer។

```
MOV AX , 07H
MOV CX , min-loc ; Minimum location
MOV DX , max-loc ; Maximum location
INT 33H
```

5.6.9. **Function 08H: Set Vertical Limits for pointer**

យើងអាចធ្វើប្រមាណវិធីដើម្បីកំណត់តម្លៃ Minimum និង Maximum Vertical ចំពោះ Pointer។

```
MOV AX , 07H
MOV CX , min-loc ; Minimum location
MOV DX , max-loc ; Maximum location
INT 33H
```

EX: void mouseminmax( ){

```
int x1=100;
int x2 =400;
int y1 =200;
int y2 = 400;
regin.x.ax = 7;
regin.x.cx = x1 ;
regin.x.dx = x2 ;
int86( int 0x33 ,&regin , &regout )
regin.x.ax = 8;
regin.x.cx = y1 ;
regin.x.dx = y2 ;
int86( int 0x33 ,&regin , &regout )}
```

ឧទាហរណ៍: ប្រូក្រាមខាងក្រោមនេះគឺប្រើ int 0x33 និង Function:00H, 01H ,03 ដើម្បីតេសទៅលើ Menu ដែលមានទំរង់ដូចខាងក្រោម៖

```
;GRAPHIC USING MOUSE POINTER
;PROGRAM: STUDENTS PRACTICES
;File names: EXAMPLE_TEST_MOUSE.ASM
;WRITE BY :OUK POLYVANN
;DATE :12-10-2009
include "8086io.inc" ;ប្រើ Header File ឈ្មោះ 8086io.inc
Org 100h ; កំណត់លេខ Address ចាប់ផ្តើមនៃកូដ
.DATA
```

```
ascii db "00000000"
MENU  DB 0C9H,15 DUP( 0CDH ),0BBH
      DB 0BAH,'  ARITHMETIC   ',0BAH
      DB 0C8H,15 DUP( 0CDH ),0BCH
TOP_ROW EQU 10  ;TOP ROW
BOT_ROW EQU 12 ;BOTTON ROW
LEF_COL EQU 20 ;LEFT COL
RIG_COL EQU 36 ;RIGHT COL
MENU_COLOR_N  EQU 0001_1111B; BLUE BACKGROUND
MENU_COLOR_C  EQU 1001_1111B; LIGHT BLUE BACKGROUN
          BUTTON_COUNT    EQU 17
          X_ROW   EQU  20
          X_COL   EQU  10
          Y_ROW   EQU  20
          Y_COL   EQU  20
          ROW DB 00
          COL DB 00
          ROW_END DB 24
          COL_END DB 79
          MOUSE_POS_X DW 00
          MOUSE_POS_Y DW 00
          MOUSE_BUTTON DW 00
          MOUSE_POINT DB 00
          MENU_STATUS   DB   00;   MENU
          STATUS
          ;00=    NOT    DISPLAYED    ;01=
          DISPLAYED
          EIGHT   DW 08
          TEN     DW 10
          COUNT DB 17 ;COLUMN COUNT
          ATTRIB DB 00H
          PAGENO DB 00H
          .CODE
  MAIN    PROC NEAR
     CLS(  )
     CURSORON
     MOV AL, [SI]
     INT 10H
     MOV AH, ROW
     CMP AH, ROW_END
     JNE CONT_DRAW
     MOV AH, COL
     CMP AH, COL_END
     JE STOP_DRAW
     CONT_DRAW:
DEC DL
    JNZ SKIP_UPDATE_ROW
        MOV DL, COUNT
        ADD ROW, 1
        SUB COL, DL
        CALL SETCUR
    SKIP_UPDATE_ROW:
    ADD COL, 1
    INC SI
JMP BACK
STOP_DRAW:
    RET
DRAW ENDP
;   CHECK MOUSE POINT
GET_MOUSE_POINT PROC NEAR
;CHECK MENU STATUS
;IF MENU IS DIPLAYED, IGNORE BUTTON
```

```
    print  13,5,1111_1101b,"DISPLAY  AND
GET MOUSE POINTER "
    print 10,20,1111_1001b,"x:"
    print 20,20,1111_1100b,"y:"
    print 32,20,0000_1010b," Press Esc to
Exit "

    MOV ATTRIB, MENU_COLOR_N
    CALL DRAW_BUTTON
    CALL MOUSE
MAIN    ENDP
;    GET AND SHOW MOUSE POINTER
MOUSE   PROC NEAR
;INITIALIZE MOUSE
    MOV AX,0
    INT 33H
;SHOW MOUSE POINTER
    MOV AX,1
    INT 33H
MOUSE_STATUS:
;GET CHARACTER
    MOV AH, 6
    MOV DL, 255
    INT 21H
    CMP AL, 1BH
    JE EXIT_PROGRAM
;GET MOUSE STATUS
    MOV AX,3
    INT 33H
    MOV MOUSE_BUTTON, BX
;Convert xbinary to xascii
    MOV DI, DX
    MOV AX, CX
    MOV DX, 00
    DIV EIGHT
    MOV CX, AX ;cx=xascii
;IF MENU IS NOT DISPLAYED, IGNORE
MENU
CMP MENU_STATUS, 0
JNE SKIP_BUTTON_CHK
;CHECK IF THE MOUSE
    CMP MOUSE_POS_X, LEF_COL
    JB NOT_BUTTON
     CMP MOUSE_POS_X, RIG_COL
      JA NOT_BUTTON
        CMP MOUSE_POS_Y, TOP_ROW
        JB NOT_BUTTON
          CMPMOUSE_POS_Y, BOT_ROW
            JA NOT_BUTTON
                MOV MOUSE_POINT, 01
                  JMP MOUSE_POINT_SET
    NOT_BUTTON:
       MOV MOUSE_POINT, 00
    JMP MOUSE_POINT_SET
MOUSE_POINT_SET:
SKIP_BUTTON_CHK:
RET
GET_MOUSE_POINT ENDP
EXIT_PROGRAM PROC NEAR
    CLS(  )
    MOV AX,4C00H
    INT 21H
EXIT_PROGRAM ENDP
;   SET CURSOR POSITION
SETCUR PROC NEAR
    PUSHA
    MOV AH, 02
    MOV BH, PAGENO
    MOV DH, ROW
    MOV DL, COL
    INT 10H
    POPA
```

```
    MOV AX, DI
    MOV DX, 00
    DIV EIGHT
    MOV DX, AX ;dx=yascii
;CHECK IF THE MOUSE MOVE
    CMP CX, MOUSE_POS_X
    JNE MOUSE_MOVE
        CMP DX, MOUSE_POS_Y
        JNE MOUSE_MOVE
            JMP MOUSE_STILL
;IF MOUSE MOVE, STORE
    MOUSE_MOVE:
        MOV MOUSE_POS_X, CX
        MOV MOUSE_POS_Y, DX
;DISPLAY MOUSE POSITION
        MOV ROW, X_ROW
        MOV COL, X_COL+3
        CALL SETCUR
        CALL CLEAR_XY
        MOV CX, MOUSE_POS_X
        MOV AX, 00
        CALL PRINTDEC
        MOV ROW, Y_ROW
        MOV COL, Y_COL+3
        CALL SETCUR
        CALL CLEAR_XY
        MOV CX, MOUSE_POS_Y
        MOV AX, 00
        CALL PRINTDEC
        CALL GET_MOUSE_POINT
;CHECK MENU STATUS
;IF MENU IS DIPLAYED,
    CMP MENU_STATUS, 00
    JNE IGNORE_BUTTON
;IF MOUSE ON BUTTON,
    CMP MOUSE_POINT, 01
    RET
SETCUR ENDP
;   PRINT DECIMAL IN AX:CX
PRINTDEC PROC NEAR
    CMP AX, 00
    JNE PRINTDEC_LOOP
    CMP CX, 00
    JNE PRINTDEC_LOOP
    MOV DL, 30H
    mov ah, 06h
    int 21h
    JMP LEAVE_PRINTDEC
PRINTDEC_LOOP:
    pusha
;clear remainder in dx
    mov dx, 0
; divide high byte
    div TEN  ; ax=quotient, dx=remainder
; mov quotient of high byte into di
    mov BX, ax
    mov ax, cx ; mov low byte into ax
    cmp ax, 00
    JNE PRINTDEC_CONT
    CMP DX, 00
    JNE PRINTDEC_CONT
        JMP PRINTDEC_BREAK_LOOP
    PRINTDEC_CONT:
; divide low byte
    div TEN ; ax=(dx:ax)/10; dx=remainder
; mov quotient of low byte into cx
    mov cx, ax
;mov quotient of high byte into ax
;for next division
    mov ax, BX
call PRINTDEC_LOOP
;Display on screen
```

```
    JNE NOT_ON_BUTTON
        MOVATTRIB, MENU_COLOR_C
        CALL DRAW_BUTTON
    JMP ON_BUTTON
;IF MOUSE NOT ON BUTTON,
    NOT_ON_BUTTON:
    MOV ATTRIB, MENU_COLOR_N
    CALL DRAW_BUTTON
ON_BUTTON:
IGNORE_BUTTON:
MOUSE_STILL:
JMP MOUSE_STATUS
MOUSE ENDP
;DRAW BUTTON
DRAW_BUTTON PROC NEAR
    PUSHA
    MOV COUNT, BUTTON_COUNT
    MOV ROW, TOP_ROW
    MOV COL, LEF_COL
    MOV ROW_END, BOT_ROW
    MOV COL_END, RIG_COL
    LEA SI, MENU
    CALL DRAW
    POPA
    RET
DRAW_BUTTON ENDP
;   DRAW (SI)
DRAW PROC NEAR
    MOV DL, COUNT
BACK:
    CALL SETCUR
    MOV AH, 09
    MOV BH, PAGENO
    MOV CX, 1
    MOV BL, ATTRIB
    or dl, 30h
    mov ah, 06h
    int 21h
    PRINTDEC_BREAK_LOOP:
        popa
LEAVE_PRINTDEC:
    RET
PRINTDEC ENDP
;-----------------------------------------------
;   CLEAR X,Y NUMBER
;-----------------------------------------------
CLEAR_XY PROC NEAR
    PUSHA
    MOV AL, 20H
    MOV CX, 6
    MOV AH, 0AH
    INT 10H
    POPA
    RET
CLEAR_XY ENDP
;-----------------------------------------------
;   PRINT TEXT
;SET TEXT OFFSET IN DX BEFORE CALL
;-----------------------------------------------
PRINTTXT PROC NEAR
    PUSHA
    CALL SETCUR
    MOV AH, 9
    INT 21H
    POPA
    RET
PRINTTXT END
ret
```

5.7. DISK STORAGE: 13H DISK FUNCTION

5.7.1. **INT 13h** និង **AH = 02h** ប្រើសម្រាប់ធ្វើការបញ្ចូលទិន្នន័យពីឌីសសិចទ័រ(disk sectors) ទៅកាន់អង្គចងចាំ Memory។

5.7.2. **INT 13h** និង **AH = 03h** ប្រើសម្រាប់ Write write disk sectors.

AL = number of sectors to read/write (must be nonzero)

CH = cylinder number (0..79).

CL = sector number (1..18).

DH = head number (0..1).

DL = drive number (0..3 , for the emulator it depends on quantity of FLOPPY_ files).

ES:BX points to data buffer.

return:

CF set on error.

CF clear if successful.

AH = status (0 - if successful).

AL = number of sectors transferred.

Note: each sector has **512** bytes

5.7.3. **INT 12h** - get memory size.*return:*

AX = kilobytes of contiguous memory starting at absolute address 00000h, this call returns the contents of the word at 0040h:0013h.

5.7.4. **INT 13h** / **AH = 00h** - reset disk syste

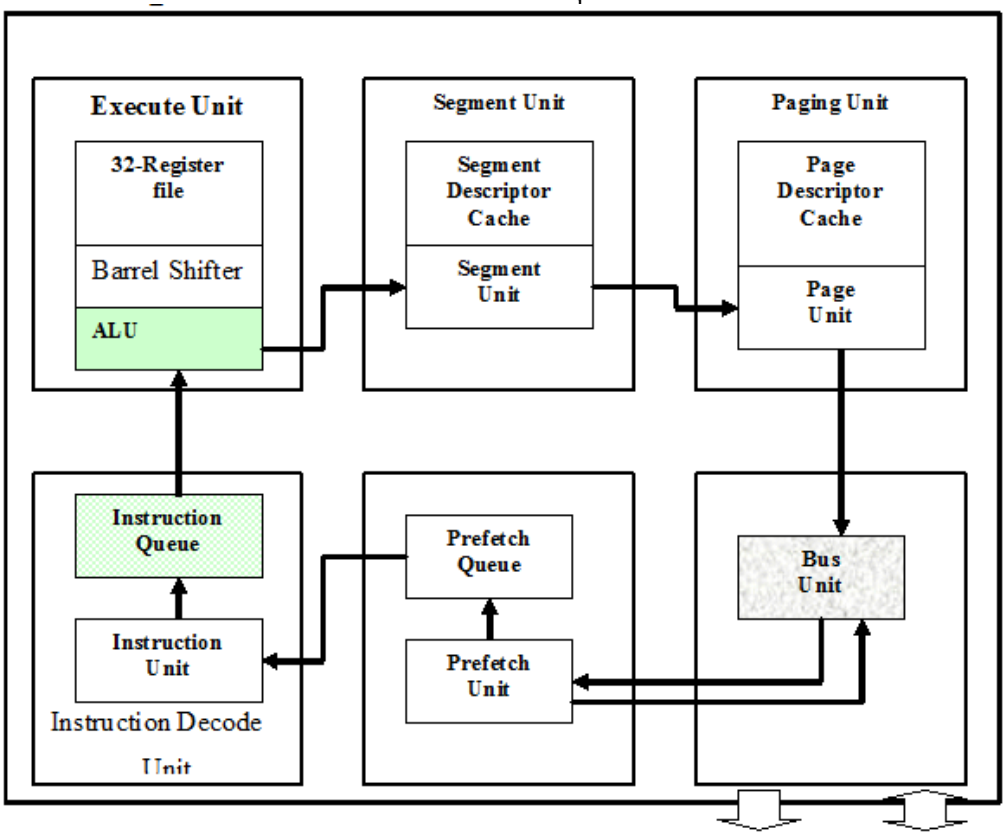
មេរៀនទី ៦ ស៊ីភីយូ ៣២ និង ៦៤ ប៊ីត

គំរូកម្មវិធី(Programming Model)ចាប់ពី 8086 រហូតដល់ Pentium Pro គឺគេគិតទៅលើផ្នែកកម្មវិធីអាចមើលឃើញ(Program Visible) ដែលបណ្តាលឱ្យ Registers របស់វាអាចប្រើបាននៅក្នុងកម្មវិធី និងកំណត់ប្រើដោយពាក្យបញ្ជាដែលអាចដំណើរការប្រមាណវិធីផ្សេងៗឬដំណើរការអ្វីមួយ។ ជាងនេះទៅទៀតចាប់ពីស៊ីភីយូ 80286 ឡើងទៅមានកម្មវិធីអាចមើលឃើញប្រើសម្រាប់ត្រួតពិនិត្យ និងប្រតិបត្តិអង្គចងចាំ(Memory)ហៅថាប្រព័ន្ធការពារអង្គចងចាំ(Protected Memory System)។

ចំពោះ 80386 Microprocessor គឺជាគ្រួសារនៃ X86 និង បានវិវត្តន៍មកពី 80286 ដោយបានកែប្រែសមាសភាគសំខាន់ដូចខាងក្រោម៖

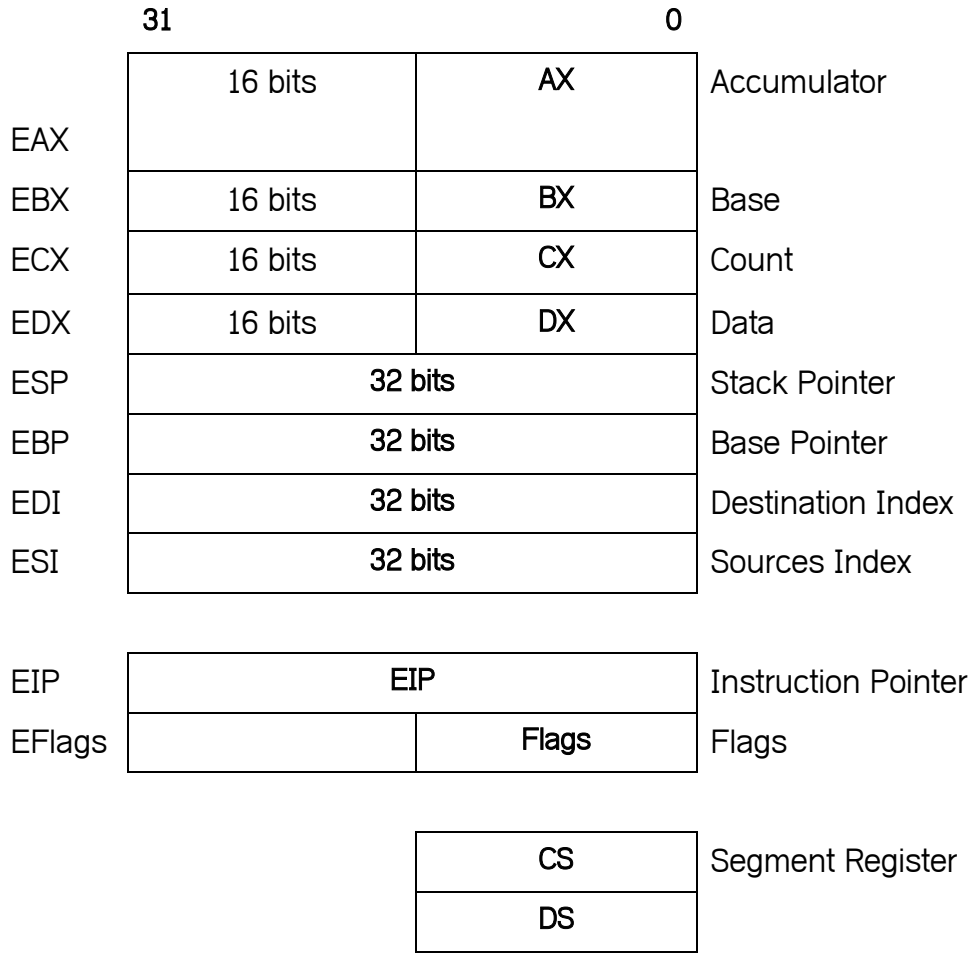
- Data Bus បានពង្រីកពី 16 bits ទៅ 32 bits ទាំង I/O Data bus និង External Data Bus
- Register ទាំងអស់បានធ្វើការអភិវឌ្ឍទៅជា 32ប៊ីត ដែលធ្វើឱ្យវាជា 32-bit Microprocessor
- Address Bus បានពង្រីកទៅ 32ប៊ីតធ្វើឱ្យទំហំ Physical Memory ឡើងដល់ទំហំ 4 GB
- កាលណាពង្រីកទំហំ Address Bus នោះទំហំ Physical Memory ត្រូវបានកើនឡើងរហូតដល់ 4 Gigabytes ឬ ១៦ GB។

8086, 8088 និង 80286 មានស្ថាបត្យកម្មខាងក្នុង 16 ប៊ីតឬ ប្រើ 16 –bit Register។ ចំណែក 80386, 80486, Pentium និង Pentium Pro ប្រើស្ថាបត្យកម្មខាងក្នុង 32ប៊ីតឬមាន 32–bits Register។



1. ៨០៣៨៦ Register

Register ដែលប្រើសម្រាប់ធ្វើប្រមាណវិធីទូទៅនៃ 80386 គឺប្រើទំហំ 32-bit registers មានដូចជា EAX, EBX, ECX, EDX, EBP, ESP, ESI, និង EDI។ Registers ទាំងនេះគឺប្រើទាក់ទងការប្រតិបត្តិពាក្យបញ្ជា ធ្វើប្រមាណវិធីនព្វន្ធ និងតក្កវិទ្យាដូចស៊ីភីយូ ៨០៨៦ ដែរ។ វាក៏ប្រើសម្រាប់ធ្វើការរកលេខអាសយដ្ឋាននៃពាក្យបញ្ជាដែលត្រូវយកពីអង្គចងចាំដោយប្រើ EIP។



1.1.EAX(Accumulator Register)

EAX អភិវឌ្ឍមកពី AX Register របស់ 8086 និង មាននាទីផ្ទុកទិន្នន័យ និង លទ្ធផលក្រោយពី ALU គណនារួច។ EAX ផ្សំពី AX និង 16 ប៊ីតបង្កើតបានទំហំស្មើនឹង 4 bytes EAX អាចប្រើជា Pointer Register ដូចការប្រើ BX របស់ 8086 ដែរ។ EXA =32 bits, AX =16 bits, AL = 8 bits និង AH = 8

EAX	16 bits	AX

1.2. Base Register (EBX)

Base Register ផ្សំពី BX និង 16bit បង្កើតបានទំហំ 4 បៃ។ មុខងារសំខាន់របស់ EBX គឺ៖

- ផ្ទុកតម្លៃទិន្នន័យ ឬ លទ្ធផលបានដូច EAX
- ឆ្ពោះទៅកាន់ Offset Address នៃទិន្នន័យក្នុងផ្នែ Data Segment

☞**ឧទាហរណ៍**: ចូរបង្ហាញតម្លៃក្នុង Register បន្ទាប់ពីប្រតិបត្តិ Instruction និមួយៗ ខាងក្រោម

```
MOV EBX, 7050H    ; កំណត់តម្លៃ 7050h ចូលក្នុង Ebx
MOV EBX, 1234759H ; កំណត់តម្លៃ 1234759H ចូលក្នុង Ebx
```

1.3. Count Register (ECX)

ECX ជា Count Register ផ្សំពី CX និង 16bit និងមានទំហំ 4 បៃ។

16 bits	CX

មុខងារសំខាន់របស់ ECX គឺ៖

- ផ្ទុកតម្លៃ Data បានដូច EAX

- ផ្ទុកចំនួន Loop ដែលត្រូវធ្វើប្រមាណវិធី
- ផ្ទុកចំនួនដងដែលត្រូវធ្វើប្រមាណវិធី SHL , SHR
- ប្រើជា Pointer Register

1.4. Data Register(EDX)

EDX គឺជា Register ដែលផ្សំពី DX និង 16 ប៊ីត។ វាមានទំហំ៤ បៃ។ មុខងារសំខាន់របស់ EDX គឺអាចផ្ទុក Data និង លទ្ធផលបានដូច AX និង EDX អាចប្រើ Pointer Register។ ខាងក្រោមគឺកូដប្រើប្រាស់មួយនឹង 8086 និង 80386:

8086	80386
.data data1 dd 50EF8765H data2 dd 12345678H resullt db 0 .code Start: mov ax,word ptr data1 mov bx, word ptr data2 add ax,bx mov dx, word ptr data1+2 mov cx,word ptr data2+2 adc dx,cx mov word ptr result, ax mov word ptr result+2 ,dx ret end start	.data data1 dd 50EF8765H data2 dd 12345678H result dd 0 .code Start: mov ecx , data1 mov edx , data2 add ecx , edx mov result, ecx ret end start

1.5. Segement Register

Segment Register ចែកជា 6 ប្រភេទគឺ

- DS Register
- ES Register
- CS Register
- SS Register
- FS Register
- GS Register

8086 Segments Register មាន:

Data Segment Register

Code Segment Register

Extra Data Segment Register

Segment Register ជាប្រភេទ Register មានទំហំ 16 bits ហើយប្រើដូច Segment Register របស់ 8086 ក្នុងការប្រើម៉ូដជា Real Mode Memory។ នៅក្នុង Protect Mode Memory ខុសពី Real Mode memory ដោយ FS និង GS regsiter វិតន៍ពី ES register។

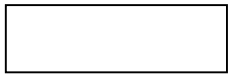
Segment	Offset	Special Purpose
CS	EIP	Instruction Address
SS	ESP and EBP	Stack Address
DS	EAX EBX ECX EDX ESI EDI and 16 or a 32 bit number	Data Adress
ES	EDI for String instructions	String destination address
FS	No default	General address
GS	No default	General address

EX: ចូរវេក Physical Adress និង Logical Addres:CS:EIP ?

1.6. Index Register

1.6.1. Source Index Register(ESI)

Source Index Register គឺជា Register ដែលមានទំហំ 4 បៃ និងប្រើសម្រាប់ផ្ទុកទិន្នន័យ និងលទ្ធផលនៃការធ្វើប្រមាណវិធីមួយដូចការប្រើ EAX ដែរ។ ESI អាចប្រើជា Pointer Register សម្រាប់ឆ្ពោះទៅកាន់ក៏មានមុខងារផ្សេង Affective Address នៃទិន្នន័យជាតួអក្សរ String នៅទីតាំង Source ក្នុងករណីប្រើ String Instruction។



☞**ឧទាហរណ៍:** ចូរបង្ហាញលទ្ធផលពីកម្មវិធីគំរូខាងក្រោម៖

```
MOV ESI , 456FAA12H ; កំណត់តម្លៃ 456FAA12H ចូលទៅកាន់ ESI
MOV ECX, 13444230H  ; កំណត់តម្លៃ 13444230H  ចូលទៅកាន់ EDI
ADD  ESI , ECX   ;  ធ្វើប្រមាណវិធីបូកតម្លៃក្នុង ESI ជាមួយនឹង ECX និងផ្ទុកលទ្ធផល ក្នុង ESI
MOV mem , ESI ; The address of mem is 0x0012FF88
```

1.6.2. **Destination Index Register**(EDI)

EDI គឺជា Register ដែលមានទំហំ 4 បៃ និងប្រើសម្រាប់ផ្ទុកទិន្នន័យ និងលទ្ធផលនៃការធ្វើប្រមាណវិធីមួយ ដូចការប្រើ EAX ដែរ។ EDI អាចប្រើជា Pointer Register សម្រាប់ឆ្ពោះទៅកាន់ Affective Address នៃទិន្នន័យជាតួអក្សរជា String នៅក្នុងទីតាំង Destination ក្នុងករណីប្រើជា String Instruction។

☞**ឧទាហរណ៍១**: កម្មវិធីខាងក្រោមប្រើ ESI និង EDI ផ្ទុកទិន្នន័យ និងលទ្ធផលនៃការបូកលេខពីរចំនួន

```
.data
        num_1 sdword 0ABCD8765h ;
        num_2 sdword 123456789h ;
.code
        Start:
                mov     esi , num_1 ; move the value of num_1 into esi
```

```
        mov     edi , num_2 ;  move the value of num_1 into edi
        sub     esi , edi ; subtract esi and edi and result store in esi
        mov     result , esi ; move the value of esi to address of result
    printf ( " result=", result );
    ret
    end start
```

1.7. Flags Register

គឺជា Register ផ្ទុក Flags ដែលបានមកពីប្រមាណវិធីមួយ ឬ កើតចេញពីប្រូក្រាមមួយ។ រូបខាងក្រោមនេះគឺជា Flags Register ចាប់ពី 80286 ឡើង

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		IOP1	IOP0	OF	DF	IF	TF	SF	ZF		AF		PF		CF
										ID	VIP	VIF	AC	VM	RF
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

16Byte

- U= Unfine
- R= Reserve
- DF= Direction Flag
- DF=Trace Flag
- TF=Trace Flag
- SF = Sign Flag
- AF = Auxilary Flag
- CF = Carry **Flag**
- IOPL(I/0 privilege level)
- NT(nested task)
- RF(resume)
- VM(vitual mode)
- AC(alignment check)
- VIP(virtual interrupt pending)
- VIF(virtual interrupt flags)
- ID(identification)

8086មានទី Flags :

Direction Flag
Trace Flag
Sign Flag
Auxilary Flag
Cary Flag
Overflow Flag
Interrupt Flag
Parity Flag
Zero Flag

1.8. Pointer Register(EIP, ESP, EBP)

1.8.1. **EIP**: ជា Register មានទំហំ 32 bit និង មានមុខងារឆ្ពោះទៅ Address របស់ Instruction ដែលត្រូវប្រតិបត្តិនៅក្នុងផ្នែ Code Segment។

1.8.2. **ESP:** ជា Register មានទំហំ 32 bit និង មានមុខងារឆ្ពោះទៅ Address របស់ Data ដែលត្រូវដំណើរការនៅក្នុងផ្នែ Stack Segment។

1.8.3. **EBP:** ជា Register មានទំហំ 32 bit និង មានមុខងារឆ្ពោះទៅ Address របស់ Data ដែលត្រូវដំណើរការនៅក្នុងផ្នែ Stack Segment។ EBP អាចប្រើសម្រាប់ផ្ទុក Data ឬ Result នៃការធ្វើប្រមាណវិធីមួយ។

2. Addressing Mode

តារាងគ្រឹះខាងក្រោម Addressing Modes ប្រើជាមួយនឹង Microprocessor 80386

Addressing Mode	Operand	Default Segment
Register	Register	None
Immediate	Data	None
Direct	Offset Address	DS
Register Indirect	[BX]	DS
	[BP]	DS
	[DI]	DS
	[EAX]	DS
	[EBX]	DS
	[ECX]	DS
	[EDX]	DS
	[ESI]	DS
	[EDI]	DS
Based Relative	[BX] + Disp	DS
	[BP] + Disp	SS
	[EAX] + Disp	DS
	[EBX] + Disp	DS
	[ECX] + Disp	DS
	[EDX] + Disp	DS
	[EBP] + Disp	SS
Indexed Relative	[DI] + Disp	DS
	[SI] + Disp	SS
	[EDI] + Disp	DS
	[ESI] + Disp	DS
Based Indexed Relative	[R1][R2]+ Disp	If BP is used , segment is SS

	R1 And R2 are any of above	Otherwise ,DS is the Segment
Scaled-indeded	Is available only in the 80386 througgh Pentium Pro Microprocessor . The second register of a pair of registers is modified by the sccale factor of 2X , 4X , or 8X to generate the operand memory address . Example : MOV EDX,[EAX+4*EBX]	

3. Math Coprocessor

ប្រភេទស៊ីភីយូប្រជា Math Coprocessors រួមមាន 8087, 80287, 80387SX, 80387DX, និង 80486SX ដែលអាចប្រើជាមួយ Microprocessor និមួយៗ។ ចំពោះ 80486 DX ឡើងទៅ Math Coprocessor គឺបានគេកសាងនៅក្នុងស៊ីភីយូតែម្តង។

8087 ជាផ្នែកមួយនៃ Math Coprocessor ដែលអាចធ្វើប្រមាណវិធីនព្វន្ទទសភាគ(floating point), រកឬសសមីការ, Partial tangent, partial arctangent, និង Logarithms។ ការប្រើ Data Type រួមមាន 16, 32 និង 64 bits Signed Integer ; 18 digit BCD data, និង 32, 64 និង 80 bits floating-point។

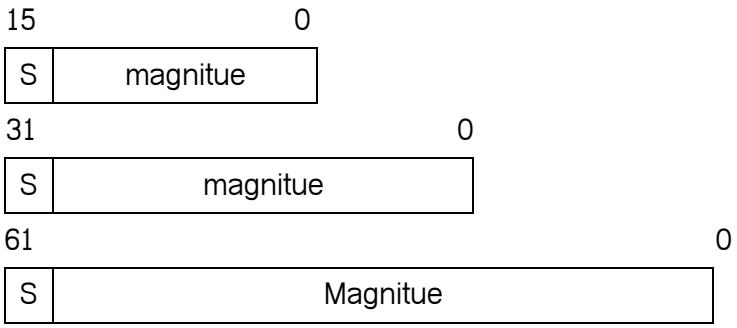
Math Coprocessor គឺជាឈីបមួយមានមុខងារធ្វើប្រមាណវិធី៖

3.1. Signed Integers

ចំនួនគត់គិតសញ្ញា(Signed number)ប្រើជាមួយ Coprocessor ដែលជាគ្រឹះសម្រាប់ធ្វើប្រមាណវិធីក្នុងប្រភេទលេខនេះ។ ហើយដើម្បីធ្វើប្រមាណវិធីវាត្រូវធ្វើតាម 2′s complement។ ពេលប្រើជាមួយ Arithmetic Coprocessor, Signed Integer មាន 16 bits (word), 32 bits (short integer), និង 64 bits (Long Integer)។

☞**ឧទាហរណ៍**: ចូរធ្វើប្រមាណវិធី Signed Number ដើម្បីរកផ្ទៀងផ្ទាត់ចម្លើយខាងក្រោម៖

```
0002              DATA1 DW    +1  ;16 bits integer
FFDE              DATA2 DW    -34 ; 16 bits integer
000004D2          DATA3 DD    +1234; short integer
FFFFFF9C          DATA4 DD    -100; short interger
000000000005BA0   DATA5 DQ    +23456 ; long integer
FFFFFFFFFFFFFF86  DATA6 DQ    -122 ; long interger
```



3.2. **លេខគោលពីរក្នុងគោលដប់** (Binary_Code Decimal)

ទំរង់លេខនេះតម្រូវឱ្យមាន 80 ប៊ីតនៃ Memory។ លេខនិមួយៗ គឺផ្ទុកជា ១៨ កញ្ចប់តួចំនួនគត់នៅក្នុងប្រាំបួនបៃនៃ Memory។ ហើយចំពោះមួយបៃមាន ៤ តួលេខនៃលេខគោលដប់ប្រាំមួយ។ ចំនួនដប់បៃមានតែ Sign bit មួយចំពោះលេខ 18 digits signed BCD។ គួរកត់សំគាល់ដែរថាលេខវិជ្ជមាន និង អវិជ្ជមានគឺស្ថិតនៅក្នុងទំរង់ពិតមួយ និង ប្រើ complement form។

79 0

Sign Byte	D17	D16	D17	D15	...	D3	D2	D1	D0

☞**ឧទាហរណ៍**: ចូរធ្វើប្រមាណវិធីដើម្បីធ្វើឱ្យផ្ទាត់ចម្លើយខាងក្រោម៖

0000000000000000200 DATA1 DT 200

0000000000000000010 DATA2 DT -10

0000000000000010020 DATA3 DT 10020

3.3. **ចំនួនទសភាគ** (Floating-Point)

លេខជាចំនួនទសភាគអាចហៅថាជាចំនួនពិត (Real Number)។ វាចែកជាបីផ្នែកគឺ: Sign – bit, biased exponent, និង Significand bits។ គ្រួសារនៃ Arithmetic Coprocessor ទទួលស្គាល់បីប្រភេទនៃលេខ Floating _point : short (32 bits), long (64 bits), និង temporary (80 bits)។ គួរកត់សំគាល់ដែរថាទំរង់ short ហៅថា single-precision, ទំរង់ long ហៅថា double precision និង ទំរង់ Temporary ហៅថា extended precision។

31 30 22 0

S	Exp	Fraction

63 62 52 51 0

S	Exp	Fraction

79 78 64 63 0

S	Exp	Fraction

3.4. **ការបំលែងគោលដប់ទៅ ចំនួនទសភាគ** (Floating point)

ដើម្បីបំលែងលេខគោលដប់ទៅទំរង់ជាចំនួនទសភាគត្រូវអនុវត្តតាមជំហានជាបន្តបន្ទាប់ដូចខាងក្រោម៖

1. បំលែងលេខគោលដប់ ទៅលេខគោលពីរដោយធ្វើប្រមាណវិធីចែកលេខ ឬតាមវិធីដកលេខ
2. ធ្វើឱ្យលេខគោលពីរមានលក្ខណៈធម្មតា (Normalize binary number)
3. គណនាបៀសអេចប៉ូណង់គិតតាមស្វ័យគុណ ២ ស្វ័យគុណ x (Biased exponent)
4. ផ្ទុកលេខឱ្យទៅជាទំរង់ទៅជាចំនួនទសភាគ (Floating-point)

☞**ឧទាហរណ៍**: 4 ជំហានបន្តបន្ទាប់ខាងក្រោមនេះជាការបំលែងលេខ Dec=100.25 ទៅជាទំរង់ Floating_point ដោយអនុវត្តន៍តាមទំរង់ Single_precision:

Step Result

1. 100.25 = 1100100.01 បំលែងលេខគោលដប់ ទៅលេខគោលពីរ
2. 1**100100**.01=1.10010001x **2^6 កាត់យកក្បៀស 1. និង 2^6 រាប់លេខ 100100**

3. 110 + 01111111 = 10000101 ដោយលេខ 01111111 ជាលេខថេរដែលត្រូវបូក
4. Sign = 0

   Exponent = 10000101

   Significand = 10010001000000000000000

   Hexaa =42C88000H

3.5. ការបំលែងពីចំនួនទសភាគ ទៅគោលដប់

ដើម្បីធ្វើការបំលែងលេខទសភាគទៅជាទំរង់គោលដប់ត្រូវអនុវត្តតាមជំហានជាបន្តបន្ទាប់ដូចខាងក្រោម៖

5. បំបែក Sign_bit, biased exponent, និង Significand
6. បំលែង Biased exponent ទៅជា True exponent = Exp - Bias
7. សរសេរលេខ normalize binary number
8. បំលែងវាទៅជា de_normalize binary number
9. បំលែង de_normalize binary number ទៅជា Decimal number

☞**ឧទាហរណ៍**: ចូរបំលែងលេខជា single_precision floating_point ខាងក្រោមនេះទៅជាលេខ Decimal:

a. 11000001110010010000000000000000
b. 11000001111100000000000000000000

Step	Result
1.	Sign = 1 Exponent = 10000011 Significant = 10010010000000000000000
2.	100 = 10000011 - 01111111
3.	1.1001001×2^4
4.	11001.001
5.	-25.125

4. សំនុំពាក្យបញ្ជា (INSTRUCTION SET) និង FP register

4.1. Data Transfer Instructions

មានគ្រឹះបីសំខាន់ក្នុងការបំលាស់ទី Data: Floationg_point, Signed Integer និង BCD

- **Floating_point Data Transfers**: មានប្រភេទពាក្យបញ្ជាបួនគឺ FLD (Load Real), FST (Store real), FSTP (store real and pop) និង FXCH (exchange)។ Inst មានពាក្យបញ្ជាថ្មីថែមលើស៊ីភីយូ Pentium Pro ហៅថា FCMOV ប្រើជាមួយ Flaoting_point condition។

- ពាក្យបញ្ជា **FLD**: ចំលងទិន្នន័យជាចំនួនទសភាគពីក្នុងអថេរ (Memroy) ទៅកាន់ផ្នែ Top stack។ វាផ្ទុកនៅ Top stack ហើយបន្ថែម Stack pointer ដោយលេខ 1។

- ពាក្យបញ្ជា **FST**: ធ្វើការចំលងទិន្នន័យជាចំនួនទសភាគពីក្នុង Top stack ទៅកាន់អថេរវិញ(Memory)។

- ពាក្យបញ្ជា **FSTP**: ចំលងទិន្នន័យពី Top stack ទៅ Memory ឬ Coprocessor ផ្សេងៗ រួចហើយផ្ទេរទិន្នន័យចេញពី Top Stack។
- ពាក្យបញ្ជា FXCH: ធ្វើផ្លាស់ប្តូរតម្លៃរវាង Top stack

▪ **Integer Data Transfer Instruction**: Coprocessor ស្គាល់ពាក្យបញ្ជាសម្រាប់ធ្វើការផ្លាស់ប្តូរទិន្នន័យជាប្រភេទ Integer Data: FILD (load integer), FIST (store integer), និង FISTP (strore integer and pop) ហើយពាក្យបញ្ជានិមួយៗមានមុខងាររៀងដូច FLD, FST និង FSTP instruction។

▪ **BCD Data Transfer Instruction:** ប្រើ Instructions ពីរគឺ FBLD និង FBSTP ហើយមានមុខងារដូច FLD និង FST instruction។

▪ **FPU** Register គឺប្រើទំហំ ៨០ ប៊ីតដែលមានដូចរូបខាងក្រោម៖

ដាក់ FPU Register

4.2. Arithmetic Instructions

4.2.1. FADD instruction: ប្រើសម្រាប់ធ្វើប្រមាណវិធីបូកក្នុង Coprocessor

Mode	form	Example
Stack	ST(1) , ST	FAAD ; ST(1) = ST(1) + ST
Register	ST , ST(n)	FAAD ST , ST(2) ;ST = ST + ST(2)
	ST(n),ST	FAAD ST(2), ST ;ST(2) = ST(2) + ST
Register	ST(n),ST	FADDP ST(3) , ST ; ST(3) =ST(3) + ST
Memory	Operand	FADD DATA2

☞**ឧទាហរណ៍**: ប្រូក្រាមខាងក្រោមប្រើសម្រាប់ធ្វើការបូកលេខពី Floating_point

```
.Data
        x    real4 100.50 ; ប្រកាសអថេរ x ទំហំ ៣២ប៊ីតផ្ទុកលេខ 100.50
        y    real4 200.25  ; ប្រកាសអថេរ y ទំហំ ៣២ប៊ីតផ្ទុកលេខ 200.25
        r    real8 0.0     ; ប្រកាសអថេរ r ទំហំ 64 ប៊ីតផ្ទុកលេខ 0
.Code
start:
        Fld x ; st(0)=x មានន័យថាចំលងតម្លៃពីអថេរ x ទៅ st(0)  វាជា Top Register
        Fld y ; st(0)=y ;ចំលងតម្លៃពីអថេរ y ទៅ st(0) ហើយតម្លៃ x នៅSt(1) វិញ st(1)=x
        Fadd st(0), st(1) ;st(0) =st(0)+st(1) ធ្វើប្រមាណវិធីបូកលេខ st(0) បូកនឹង st(1)
        Fst r ; ផ្ទេរតម្លៃពីក្នុង st(0) ទៅអថេរ r (load the value from st(0) to r)
        printf("sum=%.2f",r) ; ធ្វើការបង្ហាញលទ្ធផលនៅលើអេក្រង់បញ្ចេញតម្លៃ
        getkey
Ret
end start
```

4.3. ពាក្យបញ្ជា FMUL: ប្រើសម្រាប់ធ្វើប្រមាណវិធីគុណលេខទសភាគ។ ចំពោះពាក្យបញ្ជា MUL ប្រើកន្លងមកសម្រាប់គុណលេខចំនួនគត់។

format	Example
FMUL	FMUL DATA
FMULP	FMULP ST , ST(2)
FIMUL	FMUL ST(2),ST

☞**ឧទាហរណ៍១**: ប្រូក្រាមគំរូខាងក្រោមប្រើសម្រាប់ធ្វើការគណនាផ្ទៃរង្វង់៖

```
.DATA
        RAD     real4   91.67; ប្រកាសអថេរ RAD ទំហំ ៣២ប៊ីតផ្ទុកលេខ 91.67
        AREA    real8   0.0; ប្រកាសអថេរ AREA ទំហំ 64 ប៊ីតផ្ទុកលេខ 00.00
.CODE
start:
        FLD RAD ;ចំលងតម្លៃពីអថេរ RAD  ទៅ st(0)
        FMUL ST(0),ST(0) ; ST(0) = ST(0) គុណនឹង ST(0); ST(0) =RAD  *RAD
        FLDPI ; ST(0) ; ផ្ទេរតម្លៃ  π (3.14159.. ) ទៅកាន់ ST(0)
        FMUL ST(0), ST(1) ; ST(0)= ST(0)  * ST(1) = π* RAD * RAD
        FSTP AREA ; ផ្ទេរតម្លៃពីក្នុង st(0) ទៅអថេរ AREA
        printf( "The area circle is %f",AREA )
        getkey
ret
end start
```

☞**ឧទាហរណ៍២**: ប្រូក្រាមគំរូខាងក្រោមប្រើសម្រាប់ធ្វើការគណនាឬសសមីការដឺក្រេទី១:

```
.Data
        a Real4 0.0; ប្រកាសអថេរ a ទំហំ ៣២ប៊ីតផ្ទុកលេខ 00.00
        b Real4 0.0; ប្រកាសអថេរ b ទំហំ ៣២ប៊ីតផ្ទុកលេខ 00.00
        x Real8 0.0; ប្រកាសអថេរ x ទំហំ ៣២ប៊ីតផ្ទុកលេខ 00.00
.Code
start:
;       invoke ClearScreen, 0
        Invoke crt_printf, chr$( "Enter a:" ) ; បង្ហាញឃ្លា Enter a:
        Invoke crt_scanf, chr$( "%f" ), Addr a
        Invoke crt_printf, chr$( "Enter b:" ) ; បង្ហាញឃ្លា Enter b:
        Invoke crt_scanf, chr$( "%f" ), Addr b
        .If ( a == 0 && b == 0 )
                print " Equation is many root", 13, 10
        .ElseIf ( a == 0 && b != 0 )
```

```
        print " Equation is not root", 13, 10
    .Else
        Fld a ; ចំលងតម្លៃពីអថេរ a  ទៅ st(0)=a
        Fld b ;ចំលងតម្លៃពីអថេរ b  ទៅ st(0)=b និង st(1) = a
        Fdiv St(0), St(1) ; st(0) = st(0) / st(1) =b/a
        Fchs  ; ធ្វើការផ្លាស់ប្តូរតម្លៃពី st(0) ទៅ st(1)
        Fst x ; ផ្ទេរតម្លៃពីក្នុង st(0) ទៅអថេរ x
        print "x:" ; ធ្វើការបង្ហាញឃ្លា x: នៅលើអេក្រង់បញ្ចេញលទ្ធផល
        Invoke crt_printf, chr$("%g"), x ; បញ្ចេញតម្លៃលទ្ធផលពីអថេរ x
    .EndIf
    getkey
    Ret
end start
```

4.4. **ពាក្យបញ្ជា FSQRT: ប្រើសម្រាប់រកឫសការេនៃ ST(0) (Square Root)**

```
        FSQRT RSQRT
```

☞**ឧទាហរណ៍**: ការគណនាឫសសមីការដឺក្រេទី២ដោយ $Root = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$- បើ Delta > 0។ ចូរបង្ហាញតម្លៃក្នុង R1 និង R2 ក្រោយកម្មវិធីដំណើរការចប់។ កម្មវិធីគំរូនេះគ្រាន់តែអោយវិភាគនៃការប្រើពាក្យបញ្ជាក្នុង FPU ប៉ុណ្ណោះ។

```
.data
TWO     DD 2.00 ; ប្រកាសអថេរ TWO ផ្ទុកលេខ 02.00
FOUR    DD 4.00; ប្រកាសអថេរ FOUR ផ្ទុកលេខ 04.00
A1      DD 1.00; ប្រកាសអថេរ A1 ផ្ទុកលេខ 01.00
B1      DD 0.00; ប្រកាសអថេរ B1 ផ្ទុកលេខ 00.00
C1      DD -9.0; ប្រកាសអថេរ C1 ផ្ទុកលេខ -09.00
R1      DD ? ; ប្រកាសអថេរ R1 ផ្ទុកលេខ 00.00
R2      DD ? ; ប្រកាសអថេរ R2 ផ្ទុកលេខ 00.00
.CODE
start:
    FINIT ; ; initialise 80387 co-processor
    FST   R1 ; ចំលងតម្លៃពី Register  st(0) ទៅ R1, ដោយ R1=00.00
    FST   R2 ; ចំលងតម្លៃពីពី Register st(0) ទៅ R2
    FLD   TWO ; ចំលងតម្លៃពីអថេរ TWO ទៅ st(0),ដោយ st(0)=2,st(1)=0
    FMUL  A1  ;  st(0) = st (0) គុណនឹង A1
    FLD   FOUR; ចំលងតម្លៃពីអថេរ FOUR ទៅ st(0)
    FMUL  A1 ;   st(0)=st(0)  * A1 នាំអោយបាន st(0)=4, st(1)=2, st(2)=0
```

```
        FMUL  C1 ;  គុណលេខ st(0) នឹង C1 និងផ្ទុកលទ្ធផលក្នុង St(0)
        FLD   B1 ; ចំលងតម្លៃពីអថេរ B1 ទៅ st(0)
        FMUL  B1 ;  គុណលេខ st(0) នឹង B1 និងផ្ទុកលទ្ធផលក្នុង St(0)
        FSUB ;
        FTST ; Compare ST(0) with 0.0.
        FSTS R1 ;
        SAHF
        JZ ROOTS1
        FCOMPP
    JMP ROOTS2
        ROOTS1:
            FLD B1
            FSUB ST , ST(1)
            FDIV R1
            FLD B1
            FADD
            FDIVR
            FSTR R2
        ROOTS2:
        ret
        end
```

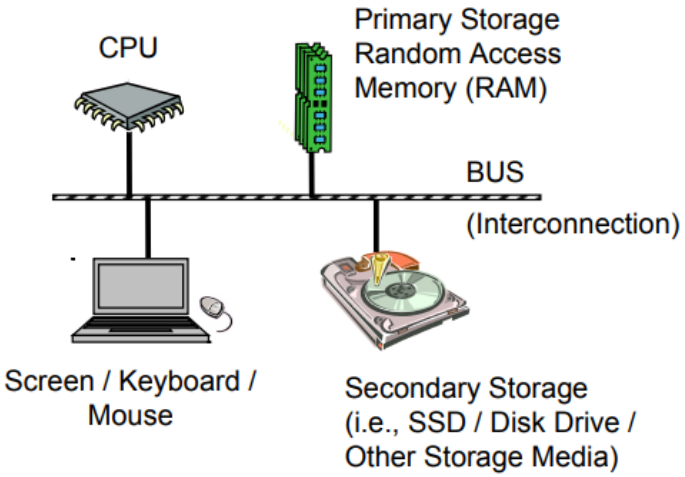
5. Core 2 Duo, Core i3, i5, i7, i9 and Xeon Microprocessor

អស់រយៈពេលជាច្រើនឆ្នាំមកហើយអ្នកសរសេរកម្មវិធីកុំព្យូទ័របានប្រើ x86 ដើម្បីសរសេរកូដអនុវត្តការងារសំខាន់ៗទាក់ទងផ្នែករឹង និងផ្នែកទន់។ ទោះជាយ៉ាងណាក៏ដោយកុំព្យូទ័រ 32 ប៊ីតកំពុងត្រូវបានជំនួសដោយកុំព្យូទ័រ 64 ប៊ីតហើយភាសា Assembly ក៏បានផ្លាស់ប្តូរពី ៣២ ប៊ីតទៅ ៦៤ ប៊ីត ដែរ។ x64 គឺជាឈ្មោះទូទៅសម្រាប់ផ្នែកបន្ថែម ៦៤ ប៊ីតចំពោះស្ថាបត្យកម្មសំណុំប៊ីតរបស់ Intel និង AMD's 32-bit x86។ AMD បានណែនាំកំណែទី១ នៃ x64 ដែលដំបូងហៅថា x86-64 ហើយក្រោយមកប្តូរឈ្មោះទៅជា AMD64។ ក្រុមហ៊ុន Intel បានដាក់ឈ្មោះការអនុវត្តរបស់ពួកគេដោយ IA-32e និងបន្ទាប់មក EMT64។

ក្រុមហ៊ុនផលិត Microprocessor ម៉ាក Intel Corei7 ឬ Corei9 ពួកគេត្រូវបានណែនាំអោយប្រើនៅក្នុងខែឧសភាឆ្នាំ ២០១៧ សម្រាប់បន្ទះឈីប LGA ២០៦៦ ដែលត្រូវបានគេស្គាល់ថាជាប្រព័ន្ធដំណើរការ Intel Core X-series។ ជាមួយនឹងចំនួន Core កំរិតខ្ពស់របស់ពួកគេប្រើធន់នឹងថាមពលកំដៅព្រមទាំងដំណើរការមានលឿនបំផុត។ ពួកគេត្រូវបានបង្កើតឡើងសម្រាប់ប្រើប្រាស់ជាមួយនឹងកុំព្យូទ័រទំនើបពេលបច្ចុប្បន្ននេះ។ ការកំណែទូរស័ព្ទចល័តដែលមានមូលដ្ឋានលើរន្ធ BGA 1440 ស្តង់ដារត្រូវបានចេញផ្សាយនៅឆ្នាំ ២០១៨ ដែលមាន Cores ចំនួន ៦ ដែលមានអាយុកាលខ្ពស់និង Cash Memory ទំហំ ១២ មេហ្គាបៃផងដែរ។

5.1. ស្ថាបត្យកម្ម (Architecture)

សមាសធាតុមូលដ្ឋាននៃកុំព្យូទ័ររួមមានអង្គភាពដំណើរការមានស៊ីភីយូ,អង្គចងចាំ (RAM) ឧបករណ៍ផ្ទុកទីពីរ(Second Storage) និង ឧបករណ៍បញ្ចូល/បញ្ចេញទិន្នផល។ ហើយឧបករណ៍ទាំងអស់អាចធ្វើការតភ្ជាប់គ្នាទៅបានត្រូវប្រើ Bus សម្រាប់នាំទិន្នន័យ និងពាក្យបញ្ជារវាងឧបករណ៍ទាំងនេះ។ ស្ថាបត្យកម្មគឺសំដៅ Von Neumann Architecture4 ដែលបង្ហាញអំពីប្លង់ផ្នែករឹងដែលនៅក្នុងនោះមានក្នុងស៊ីភីយូមួយ ដើម្បីអោយអ្នកការសរសេរកម្មវិធីយកវាមកប្រើដើម្បីសរសេរកូដដោយផ្ទាល់ប្រើផ្នែករឹង និងផ្នែកទន់ដែលមានស្រាប់។ កម្មវិធី និងទិន្នន័យជាធម្មតាត្រូវបានរក្សាទុកនៅលើឧបករណ៍ផ្ទុកក្នុង Hard Disk។ នៅពេលដែលកម្មវិធីមួយត្រូវបានប្រតិបត្តិវាត្រូវតែចម្លង ពីឧបករណ៍ផ្ទុកទីពីរទៅដាក់ក្នុងអង្គចងចាំចម្បងឬ អង្គចងចាំមេ (RAM) ទើបស៊ីភីយូអាចដំណើរការកម្មវិធីទាំងនោះទៅបាន។

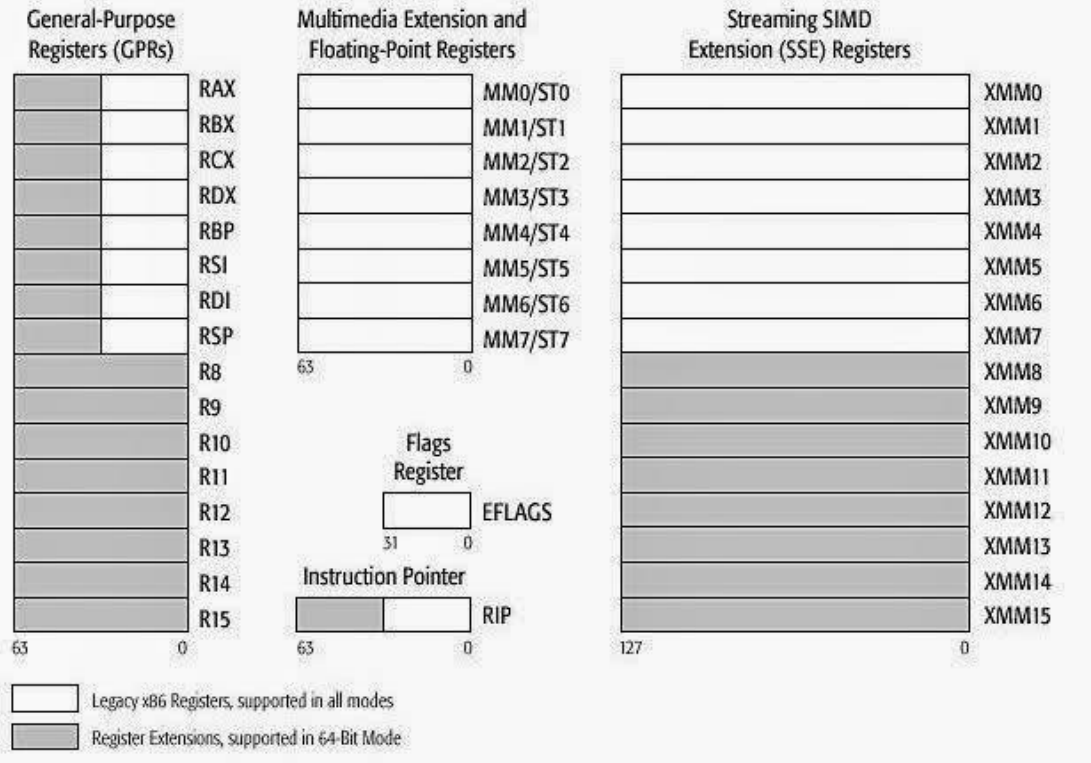


5.2. ការប្រើប្រាស់ Register

ចាប់តាំងពីមានការប្រើ 64 ប៊ីត Register អនុញ្ញាតឱ្យប្រើប្រាស់ទំហំ និងទីតាំងមានច្រើនប្រភេទកំណត់បៃជា 8 ប៊ីត, Word ជា 16 ប៊ីត, Double Word ជា 32 ប៊ីត, quadword ជា 64 ប៊ីត និង Double quadword ជា 128 ប៊ីត។ ក្រុមហ៊ុន Intel រក្សាទុកបៃ "endian តិចតួច" មានន័យថាបៃសំខាន់ទាបត្រូវបានរក្សាទុកនៅក្នុងអាស័យដ្ឋានមេម៉ូរីទាបដើម្បីសរសេរកូដក្នុងភាសា Assembly នៅក្នុង Core i3,5,7,9 ឬលេខលើសពីនេះ ត្រូវដឹងពីទំហំ និងមុខងារ Register, ពាក្យបញ្ជា (Instruction) និងរចនាសម្ព័ន្ធនៃការប្រើប្រាស់ CPU 32 bit ជាមុនសិន។ ជាពិសេសត្រូវចេះរចនាសម្ព័ន្ធរបស់ស៊ីភីយូ ៨០៨៦ ដែលជាមូលដ្ឋានគ្រឹះដ៏សំខាន់បំផុតសម្រាប់រៀនពីស្ថាបត្យកម្មនិងភាសា Asm នៃស៊ីភីយូដទៃទៀត។ រូបភាពខាងក្រោមបង្ហាញឈ្មោះ Register ដែលមានទំហំ ៦៤ bit ជាប្រភេទប្រើប្រាស់ទូទៅសម្រាប់គ្រប់ការងារទាំងអស់ដែលមានចំនួន៨ គឺ RAX, RBX, RCX, RDX, RBP, RSI, RDI, និង RSP។ បន្ទាប់មកទៀតគឺ Register មានឈ្មោះ R8-R15 ដែលមានការប្រើអក្សរ R ជំបូងជំនួសដោយអក្សរ E នៅលើ Register ចំនួនប្រាំបីដំបូងថែមលើ Register 64 ប៊ីតខាងលើ។ ស្រដៀងគ្នានេះដែរសម្រាប់ RAX, RBX, RCX, និង RDX គឺប្រើដូច Eax, Ebx, Ecx, និង Edx ដែរ។ ដោយ Rax មាន Eax និង 32 bit។

Register	Accumulator		
64-bit	RAX		
32-bit		EAX	
16-bit			AX
8-bit		AH	AL

- ចំពោះ Register ពី R8 ដល់ R15 អាចប្រើបានតាមរបៀបស្រដៀងគ្នានេះដែរ៖ R8 (qword), R8D (dword), R8W (word), R8B (Byte)។
- RIP 64 ប៊ីតជា Pointer Register ចង្អុលទៅកាន់ពាក្យបញ្ជាបន្តបន្ទាប់ដែលត្រូវយកមកប្រតិបត្តិនិងគាំទ្រពីម៉ូតដែលមេម៉ូរីទំហំ 64 ប៊ីត។ ប្លង់អាសយដ្ឋាន Memory នៅក្នុងប្រព័ន្ធប្រតិបត្តិការបច្ចុប្បន្នត្រូវបានគ្របដណ្តប់សម្រាប់អភិវឌ្ឍន៍ នៅស៊ីភីយូក្រោយៗទៀត។
- Register ចាប់ពី MM0/ST0 ដល់ MM7/ST7 ដោយមួយៗមានទំហំ ៦៤ ប៊ីត និងប្រើទាក់ទងនឹងប្រមាណវិធីទសភាគដូចប្រើក្នុងឈីប ៨០៨៧។
- Register XMM មានចំនួនទាំងអស់ដប់ប្រាំមួយ Register ដែលអនុញ្ញាតឱ្យប្រតិបត្តិការស្របគ្នាលើតម្លៃជាក់លាក់ទ្វេឬពីរដង។ ពាក្យខ្លះក៏ធ្វើការផងដែរលើបណ្តុំបៃ, Word, Dword និងលេខគត់។ ការណែនាំទាំងនេះហៅថាការបន្ថែមស៊ីមកាតអេសឌី XMM0 ដល់ XMM15 មួយៗទំហំ ១២៨ ប៊ីត។



5.3. ទំហំ Data Storage

ស្ថាបត្យកម្ម x86-64 អាចប្រើប្រាស់សំណុំជាក់លាក់នៃទំហំជាតុសម្រាប់ទិន្នន័យដែលមានមូលដ្ឋានទាំងអស់គ្រឹះមកពី x86-32 bits។ តារាងខាងក្រោមបង្ហាញអំពីការប្រើប្រភេទទិន្នន័យដែលអាចប្រើនៅក្នុងភាសាAssembly ដែលអាចធ្វើប្រមាណវិធីជាចំនួនគត់៖

Storage Size	Size(bit)	Size(bit)	Data Type
Byte	8-bit	1 byte	db
Word	16-bit	2 bytes	Dw
Double-word	32-bit	4 bytes	dd
Quadword	64-bit	8 bytes	dq
Double quadword	128-bit	16 bytes	ddq

5.4. **សំនុំពាក្យបញ្ជា** (Instruction Sets)

ការប្រើប្រាស់ពាក្យបញ្ជា (Instructions) គឺធ្វើដូចប្រើប្រាស់នៅក្នុងស៊ីភីយូ 32 bit ដែរ មានដូចជាការប្រើ Mov, Add, Sub, Mul, Div, And, Or, Not និងមានការថែមឈ្មោះពាក្យបញ្ជាជាច្រើនទៀតនៅក្នុងស៊ីភីយូ ទើបបង្កើតថ្មីទៅតាមការអភិវឌ្ឍន៍ នៃសម័យកាលនិមួយៗ។ តារាងគ្រឹះខាងក្រោមបង្ហាញពីឈ្មោះពាក្យខ្លះដែលអាចប្រើជា Corei7 ឬ Corei9:

Opcode	Meaning	Opcode	Meaning
MOV	Move to/from/between memory and registers	AND/OR/XOR/NOT	Bitwise operations
CMOV *	Various conditional moves	SHR/SAR	Shift right logical/arithmetic
XCHG	Exchange	SHL/SAL	Shift left logical/arithmetic
BSWAP	Byte swap	ROR/ROL	Rotate right/left
PUSH/POP	Stack usage	RCR/RCL	Rotate right/left through carry bit
ADD/ADC	Add/with carry	BT/BTS/BTR	Bit test/and set/and reset
SUB/SBC	Subtract/with carry	JMP	Unconditional jump

MUL/IMUL	Multiply/unsigned	JE/JNE/JC/JNC/J *	Jump if equal/not equal/carry/not carry/ many others
DIV/IDIV	Divide/unsigned	LOOP/LOOPE/LOOPNE	Loop with ECX
INC/DEC	Increment/Decrement	CALL/RET	Call subroutine/return
NEG	Negate	NOP	No operation
CMP	Compare	CPUID	CPU information

6. x86-64 Assembly

កុំព្យូទ័រ PC នីមួយៗមាន microprocessor ដែលគ្រប់គ្រងលើប្រមាណវិធីនព្វន្ធ តក្កវិទ្យា និងការកុងត្រូលកុំព្យូទ័រអោយដំណើរការ។ ក្រុមគ្រួសារនៃស៊ីភីយូដំណើរការនីមួយៗមានពាក្យបញ្ជា (Instruction) ផ្ទាល់ខ្លួនសម្រាប់ដោះស្រាយប្រតិបត្តិការផ្សេងៗ ដូចជាទទួលបានព័ត៌មានបញ្ចូលពី keyboard បង្ហាញព័ត៌មាននៅលើអេក្រង់ និងបំពេញការងារផ្សេងៗ។ សំណុំនៃពាក្យបញ្ជាទាំងនេះត្រូវបានគេហៅថាភាសាស៊ីភីយូ ឬភាសាម៉ាស៊ីន។ ខ្លះក្បាលយល់តែភាសាម៉ាស៊ីនដែលមានទំរង់លេខ ១ និងលេខ០។ ទោះយ៉ាងណាក៏ដោយភាសាម៉ាស៊ីនមាននៅមានភាពស្រពិចស្រពិល និងស្មុគស្មាញសម្រាប់ប្រើក្នុងការអភិវឌ្ឍន៍ផ្នែកទន់។ ដូច្នេះភាសាកំរិតទាបត្រូវបានរចនាឡើងសម្រាប់ក្រុមគ្រួសារស៊ីភីយូទៅតាមដំណាក់កាលនិមួយៗ ដោយ ៨០៨០ ជាគ្រឹះដំបូងសម្រាប់បង្កើតជាគ្រួសារស៊ីភីយូបច្ចុប្បន្នដូចជា Corei7 ឬ Corei9 ជាដើម។

ការយល់ដឹងអំពីភាសា Assembly ធ្វើឱ្យយើងដឹងអំពីៈ

- របៀបដែលកម្មវិធីមានទំនាក់ទំនងជាមួយប្រព័ន្ធប្រតិបត្តិការប្រព័ន្ធដំណើរការ និង BIOS
- អំពីទិន្នន័យត្រូវបានតំណាងនៅក្នុងអង្គចងចាំជាមួយនឹង ឧបករណ៍បញ្ចេញឬបញ្ចូល(IO)
- ដឹងពីខ្លះក្បាលដំណើរការប្រមាណវិធី និងពាក្យបញ្ជា(Instruction)ក្នុងអង្គចងចាំ
- របៀបដែលពាក្យបញ្ជាចូលទៅដំណើរការនូវទិន្នន័យ និងកម្មវិធីប្រើឧបករណ៍ខាងក្រៅ
- ដឹងពីពាក្យបញ្ជាដំណើរការជាក់លាក់លើផ្នែករឹងរបស់វាដែលជាគំនិតអោយបង្កើតផ្នែករឹង
- អាចសរសេរកម្មវិធីដោះស្រាយចំណោទ ឬបញ្ជាដូចភាសាកំរិតខ្ពស់

6.1. កម្មវិធីដំបូងក្នុងឈីប Corei7 ឬ Corei9

កម្មវិធីគំរូខាងក្រោមដំបូងដើម្បីធ្វើការបង្ហាញឃ្លា 'Wellcome RUPP!' នៅលើអេក្រង់បញ្ចេញលទ្ធផល(Output Screen) ដោយកម្មវិធីសរសេរកូដឈ្មោះ SASM IDE: %include "io64.inc" ; បញ្ចូលឈ្មោះ Library ប្រើ ៦៤ ប៊ីត

Section .data ;ផ្នែកបញ្ចូលទិន្នន័យអោយកម្មវិធីប្រើដូច .data ក្នុងកម្មវិធី x86Asm ដែរ

msg db ' Wellcome RUPP!', 0

section .textផ្នែកបញ្ចូលពាក្យបញ្ជាអោយកម្មវិធី ប្រើដូច .code ក្នុង x86Asm ដែល

global CMAIN ; ប្រកាសឈ្មោះអោយកម្មវិធីសម្រាប់ដំណើរការពាក្យបញ្ជាផ្សេងៗ

CMAIN:

mov rbp, rsp ; កំណត់អោយ rbp ស្មើសូន្យដើម្បីធ្វើការដំណើរការកម្មវិធី Debug

PRINT_STRING msg ; បង្ហាញឃ្លា Wellcome RUPP! នៅអេក្រង់

NEWLINE ; ធ្វើការចុះបន្ទាត់បន្ទាប់ពីបង្ហាញឃ្លារួច

xor rax, rax ; កំណត់អោយ rax ស្មើសូន្យ

ret ;ត្រលប់ទៅកាន់ OS ដើម្បីធ្វើការបញ្ឈប់ការប្រតិបត្តិ

6.2. កម្មវិធីបូកលេខពីរចំនួន

កម្មវិធី Assembly គំរូខាងក្រោមបង្ហាញពីការធ្វើប្រមាណវិធីនព្វន្ធបូកពីរចំនួន r=x+y ដោយតម្លៃទាំងពីរបញ្ចូលផ្ទាល់ក្នុងកូដតែម្តង និងប្រើ NASM Compiler:

```
%include "io64.inc"

section .data ;

    x dq 100000 ; ប្រកាសអថេរ x ទំហំ ៦៤ ប៊ីតផ្ទុកលេខ 100000

    y dq 200000; ប្រកាសអថេរ y ទំហំ ៦៤ ប៊ីតផ្ទុកលេខ 200000;

    r dq 0 ;  ប្រកាសអថេរ r ទំហំ ៦៤ ប៊ីតផ្ទុកលេខ 0

section .text

global CMAIN

CMAIN:

    mov rbp, rsp; កំណត់អោយ rbp=0 ដើម្បីធ្វើការដំណើរការកម្មវិធី Debug

    mov rax, 0    ; កំណត់អោយ rax ស្មើសូន្យ ; rax =0

    mov rax, qword[x] ; កំណត់អោយ rax ផ្ទុកតម្លៃរបស់ x ,ដោយ

    rax=100000

    add rax, qword[y] ; rax = ax បូកនឹងតម្លៃរបស់ y=200000

    mov qword [r], rax ; ចំលងតម្លៃក្នុង rax ទៅកាន់អថេរ r

    PRINT_STRING "The result of x and y is " ;បង្ហាញឃ្លានៅលើអេក្រង់

    PRINT_DEC 8 , r ;ធ្វើការបង្ហាញលទ្ធផលនៅលើអេក្រង់ពីអថេរ r

xor rax, rax

ret
```

6.3. ការប្រើប្រាស់ឧទ្រម្ភ Procedure

កម្មវិធីគំរូខាងក្រោមប្រើភាសា C++ ដើម្បីចាប់តម្លៃមួយពីភាសា Assembly ដោយប្រើកម្មវិធីសរសេរកូដឈ្មោះ Microsoft Visual Studio 2019 ។ កម្មវិធីនេះត្រូវបានគេនិយមប្រើដើម្បីសរសេរកូដដោយផ្ទាល់ជាមួយ នឹងស្ថាបត្យកម្មរបស់ស៊ីភីយូដែលប្រើបច្ចុប្បន្ននេះ។ ដោយយើងអាចមើលមេរៀន និង Videos ដែលមានការសរសេរភាសា Assembly ដោយលក្ខខណ្ឌ, រង្វិលជុំ(Loop), អារេ, Pointer Register, ប្រើបញ្ជា INT និងបញ្ជាជាច្រើនទៀតដូចដែលយើងបានប្រើភាសា Assembly ក្នុង X86Asm ដែរ។សូមលឺញទៅកាន់ https://www.youtube.com/c/WhatsACreel/videos ដើម្បីធ្វើការស្រាវជ្រាវបន្ថែម។

១. ចូលកម្មវិធី Microsoft Visual Studio 2019

២. ចុចលើ Create a New Project និងបន្ទាប់ចុចលើ Empty Project រួចចុចលើ Next

៣. ត្រង់ប្រអប់ Project name វាយឈ្មោះគម្រោងដោយដាក់ឈ្មោះ asmTu1(យើងអាចដាក់ឈ្មោះអ្វីក៏បាន) តែដំបូងដើម្បីងាយក្នុងការពន្យល់ខ្ញុំដាក់ឈ្មោះនេះ និងបន្ទាប់មកចុចលើ create

៤. ចុច mouse ស្តាំលើ asmTu1 និងជ្រើសរើសយក Build Dependencies បន្ទាប់មកជ្រើសយករួចយក Build Customizations...

៥. ចុចលើគូស Tick ត្រង់ masm(.targets, props) និងចុចលើ Ok

៦. ចុច mouse ស្តាំលើឈ្មោះ asmTu1 និងបន្ទាប់មកចុចលើ Add និងយក New Items... និងជ្រើសយក C++ file(.ccp)

៧. វាយឈ្មោះ main.cpp ត្រង់ប្រអប់ Name: និងចុចលើ Add បន្ទាប់មកសរសេរកូដដូចខាងក្រោម

```
#include <iostream> //បញ្ចូលឈ្មោះ Library ប្រើប្រើរចនាសម្ព័ន្ធនៃភាសា C++
using namespace std; // ប្រើ namespace named ឈ្មោះ std សម្រាប់ប្រើ thing
extern "C" int SomeFunction( ); //extern "C" ប្រើសម្រាប់ប្រកាសអនុគមន៍
int main( )
{
        cout << "The result is: " << SomeFunction( ) << endl;
        return 0;
}
```

៨. ចុច mouse ស្តាំលើ Source Files និងជ្រើសរើសយក Add និងយក New Items... ... និងជ្រើសយក C++ file(.ccp)

៩. វាយឈ្មោះassembly.asm ត្រង់ប្រអប់ Name: និងចុចលើ Add បន្ទាប់មកសរសេរកូដដូចខាងក្រោម:

```
.code
SomeFunction proc ; ការប្រើទំរង់ Procedurec ក្នុងភាសា Assembly
        mov rax, 12345678h ; កំណត់តម្លៃ rax =12345678h
        mov rbx, 10000000h ; កំណត់តម្លៃ rbx =10000000h
        add rax, rbx ; rax = rax + rbx
```

```
ret
SomeFunction endp ; បញ្ចប់ទំរង់ Procedure
End
```

10. ជ្រើសយក **x64** ក្នុង Solutions Platforms និងចុចលើ debug menu និងជ្រើសយក Start Without Debugging ឬ ចុច Ctrl + F5

ចំណាំ: យើងអាចដំណើរការកម្មវិធី Debug ដើម្បីមើលពីការប្រតិបត្តិរបស់ស៊ីភីយូដោយផ្ទាល់ដោយចុច F5 និងចុចលើ Debug Menu និង ចុចលើ Windows ជ្រើសរើស Register និងចុចលើ Step Over ដើម្បីដំណើរការម្តងមួយ Statement។ កម្មវិធី Visual Studio 2019 អាចអោយយើងសរសេរកម្មវិធីដោយប្រើ Assembly សម័យទំនើបនេះ។

7. Addressing Mode

មុនពេលប្រើពាក្យបញ្ជាអោយបានត្រឹមត្រូវនោះ ត្រូវមានមូលដ្ឋានគ្រឹះមួយចំនួនយើងត្រូវយល់អំពីរបៀបដោះស្រាយដែលជាមធ្យោបាយដែលពាក្យបញ្ជាអាចចូលទៅកាន់ការ Register ឬការចងចាំ។ ខាងក្រោមនេះគឺជារបៀបដោះស្រាយរួមជាមួយឧទាហរណ៍ៈ

- Immediate: តម្លៃគឺផ្ទុកនៅក្នុងពាក្យបញ្ជា ADD EAX, 14 ; add 14 into 32-bit EAX
- Register to register គឺប្រើប្រាស់Register និង Register :ADD R8L, AL
- Indirect: ការប្រើប្រាស់នេះប្រើ 8, 16, or 32 bit displacement, general purpose registers សម្រាប់ base និង index, និង scale of 1, 2, 4, or 8 ដើម្បីគុណជាមួយនឹង index ។ បច្ចេកទេសទាំងនេះក៏អាចត្រូវបានជាក់បញ្ចូលជាមួយផ្នែក FS: ឬ GS:នោះត្រូវសរសេរពាក្យបញ្ជា MOV R8W, 1234[8*RAX+RCX] ; មានន័យថាចំលងទិន្នន័យជា word នៅ address 8*RAX+RCX+1234 ចូលក្នុង R8W។ មានវិធីស្របច្បាប់ជាច្រើនដើម្បីសរសេររឿងនេះ។ ខាងក្រោមនេះជាគំរូដោយ:

```
Mov Ecx, dword ptr table[RBX]RDI[]
Mov Ecx, dword ptr table[RDI]RBX]
Mov Ecx, dword ptr table[RBX+RDI[]
Mov Ecx, dword ptr [tableRBX+RDI[]
```

ចំណាំ: dword ptr ប្រាប់ទៅ assembler ពីរបៀបចូលទៅ encode MOV instruction.

7.1. Address និងតម្លៃ

វិធីតែមួយគត់ដើម្បីចូលប្រើអង្គចងចាំគឺមានតង្កៀប ([])។ មានន័យថាដើម្បីយកតម្លៃពីអថេរមួយចូល Register នោះត្រូវកំណត់ byte, word, dword, qword ទៅតាមទំហំអថេរ និងអថេរត្រូវក្នុងតង្កៀប។ តែបើអត់តង្កៀបយកលេខ Address ចូល Register ខុសពី Masm32។

```
mov rax, qword [var1] ; value of var1 in rax ;យកតម្លៃ var1 ចូល rax
mov rax, var1 ; address of var1 in rax ; យកលេខអាសយដ្ឋានរបស់ var1 ចូល rax
```

7.2. SIMD Instruction

ចំពោះពាក្យបញ្ជា Single Instruction Multiple Data (SIMD)ប្រតិបត្តិពាក្យបញ្ជាតែមួយលើបំណែកទិន្នន័យជាច្រើនស្របគ្នា និងជាការប្រើប្រាស់ធម្មតា។ ពាក្យបញ្ជា MMX និង SSE (ប្រើ Register MMX និង XMM រៀងៗខ្លួន) គាំទ្រប្រតិបត្តិការ SIMD ដែលអនុវត្តពាក្យបញ្ជាលើទិន្នន័យរហូតដល់ ៨ បំណែកស្របគ្នា។ ឧទាហរណ៍ប្រាំបីបៃអាចត្រូវបានបន្ថែមទៅប្រាំបីបៃនៅក្នុងពាក្យបញ្ជាមួយដោយប្រើ MMX។ Register ចាប់ពី MM0/ST0 ដល់ MM/ST7 គឺជា Floating-Point Register ឬហៅម៉្យាងទៀតថា FPR0-FPR7 ប្រើសម្រាប់ប្រមាណវិធីទសភាគដូចប្រើក្នុងឈីប ៨០៨៧ ផងដែរ

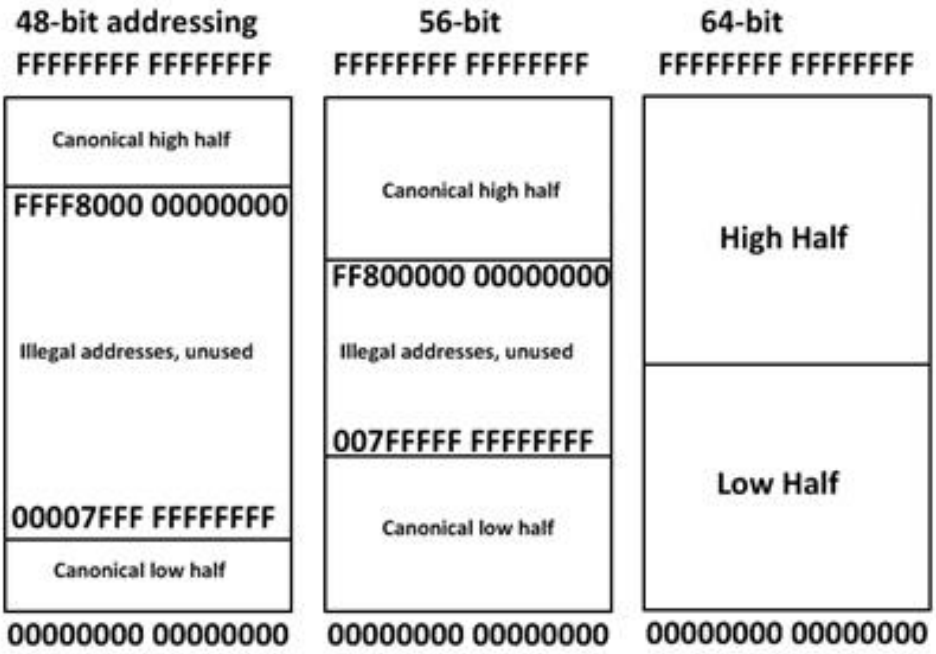
echnology	Register size/type	Item type	Items in Parallel
MMX	64 MMX	Integer	8, 4, 2, 1
SSE	64 MMX	Integer	8,4,2,1
SSE	128 XMM	Float	4
SSE2/SSE3/SSSE3...	64 MMX	Integer	2,1
SSE2/SSE3/SSSE3...	128 XMM	Float	2
SSE2/SSE3/SSSE3...	128 XMM		

```
section .data
        vec1 real4 1.0, 2.0, 3.0, 4.0
        vec2 real4 5.0, 6.0 ,7.0, 8.0
section .text
global CMAIN
CMAIN:
        ;write your code here
        movups xmm0, xmmword ptr [vec1]
        movups xmm1, xmmword ptr [vec1]
        addps xmm0, xmm1
xor rax, rax
ret
```

8. Operating System

ប្រព័ន្ធ ៦៤ ប៊ីតអនុញ្ញាតឱ្យដោះស្រាយទិន្នន័យពី២ទៅ៦៤បៃនៃទំហំទិន្នន័យនៅក្នុងទ្រឹស្តី ប៉ុន្តែមិនមានបន្ទះឈីបបច្ចុប្បន្នអនុញ្ញាតឱ្យចូលប្រើ ១៦exabytesទាំងអស់(១៨.៤៤៦.៧៤៤.០៧៣.៧០៩.៥៥១.៦១៦ បៃ)។ ឧទាហរណ៍ស្ថាបត្យកម្ម AMD ប្រើតែ ៤៨ ប៊ីតទាបនៃអាស័យដ្ឋានហើយប៊ីតពី ៤៨ ដល់ ៦៣ ត្រូវតែជាច្បាប់ចម្លងប៊ីត ៤៧ ឬខួរក្បាលដំណើរការលើកលែង។ ដូច្នេះអាស័យដ្ឋានមានចាប់ពី 0 ដល់ 00007FFF`FFFFFFFF និងពី FFFF8000`00000000 តាមរយៈ FFFFFFFF`FFFFFF សម្រាប់ ២៥៥ តេរ៉ាបៃ (២៨១.៤៧៤.៩៧៦.៧១០.៦៥៦ បៃ) នៃអាស័យដ្ឋាននិម្មិតដែលអាចប្រើបាន។ គុណវិបត្តិមួយទៀតគឺថាការដោះស្រាយអង្គចងចាំទាំងអស់ ៦៤ ប៊ីតតម្រូវឱ្យមានតារាងទំព័រដោច្រើនសម្រាប់ប្រព័ន្ធផ្ទុកទិន្នន័យដោយប្រើអង្គចងចាំដែលមានតម្លៃសម្រាប់ប្រព័ន្ធដែលមានតិចជាង ១៦ ប៊ីត។ ចំណាំទាំងនេះគឺជាអាស័យដ្ឋាននិម្មិតមិនមែនអាសយដ្ឋានពិតទេ។

ជាលទ្ធផលប្រព័ន្ធប្រតិបត្តិការជាច្រើនប្រើពាក់កណ្តាលខ្ពស់ នៃទំហំនេះសម្រាប់ប្រព័ន្ធប្រតិបត្តិការដោយចាប់ផ្តើមពីផ្នែកខាងលើនិងថយចុះចំណែកកម្មវិធីអ្នកប្រើប្រើពាក់កណ្តាលទាបចាប់ផ្តើមពីបាតឡើងហើយឡើងលើ។ ជំនាន់រឹនដូបច្ចុប្បន្នប្រើ៤៨ប៊ីតនៃអាស័យដ្ឋាន (១៦តេរ៉ាប៊ី = ១៧.៥៩២.១៨៦.០៤៤,៤១៦ បៃ)។ អាស័យដ្ឋានលទ្ធផល ត្រូវបានបង្ហាញនៅក្នុងរូបភាពទី២ អាស័យដ្ឋានលទ្ធផលមិនសំខាន់សម្រាប់កម្មវិធីអ្នកប្រើប្រាស់ទេ ពីព្រោះអាស័យដ្ឋានត្រូវបានកំណត់ដោយប្រព័ន្ធប្រតិបត្តិការ ប៉ុន្តែភាពខុសគ្នារវាងអាស័យដ្ឋានអ្នកប្រើប្រាស់និងអាស័យដ្ឋានខឺណែលមានប្រយោជន៍សំរាប់បំបាត់កំហុស។



9. Calling Conventions

នៅក្នុងផ្នែកនេះពិពណ៌នាអំពីដំណើរការស្តង់ដារ និង Convention ដែលមុខងារអ្នកហៅ(Caller)ដើម្បីហៅចូលអនុគមន៍ផ្សេងទៀតក្នុងកូដ x៦៤។ វាធ្វើការហៅ x64 ទូទៅដែលជាអនុរបស់ Microsoft ៦៤ សម្រាប់ការហៅនូវអនុគមន៍ក្នុងភាសា C មកប្រើ(សូមមើល MSDN, Chen, និង Pietrek)។ ចំណែក Application Binary Interface (ABI) កាសម្រាប់ការហៅតម្លៃដើមទៅអោយ Register ចំនួន។ ឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍នៅក្នុងភាសា C++ ហៅអនុគមន៍ funct1 ទៅអោយ Register : RCX, RDX, R8, R9 និង Push តម្លៃបន្ទាប់ចូល Stack.

```
func1(int a, int b, int c, int d, int e, int f);
// a in RCX, b in RDX, c in R8, d in R9, f then e pushed on stack
// main.cpp
        #include <iostream> ; //បញ្ចូលឈ្មោះ Library ប្រើប្រើរចនាសម្ព័ន្ធនៃភាសា C++
        using namespace std; // ប្រើ namespace named ឈ្មោះ std សម្រាប់ប្រើ thing
        extern "C" int SomeFunction( int a, int b ); //ប្រកាសអនុគមន៍មានប៉ារ៉ាម៉ែត្រ a, b
        int main( ){
                SomeFunction( 100, 200 ) ;
                return 0;}
    ;assembly.asm
        .code
        SomeFunction proc ;ប្រើសម្រាប់ទទួលយកតម្លៃពីអនុគមន៍ក្នុងភាសា C ចូល rcx,
        rdx
        ret
        SomeFunction endp
        End
```

ចម្លើយ: ក្រោយពីដំណើរការកូដនេះយើងបាន rax =100, និង rdx = 200

• RCX, RDX, R8, R9 ត្រូវបានប្រើសម្រាប់អាគុយម៉ង់ចំនួនគត់និងជាតុនៃ Ponter លំដាប់នោះពីឆ្វេងទៅស្តាំ។

• XMM0, 1, 2, និង 3 ត្រូវបានប្រើសម្រាប់អាគុយម៉ង់ចំណុច Pointer

• អាគុយម៉ង់បន្ថែមត្រូវបានបញ្ចេញឬបញ្ចូលនៅតំបន់ Stack ពីខាងឆ្វេងទៅស្តាំ និង ប៉ារ៉ាម៉ែត្រដែលមានប្រវែងតិចជាង ៦៤ ប៊ីតមិនត្រូវបានពង្រីកសូន្យទេ។

• វាគឺជាការទទួលខុសត្រូវរបស់អ្នកហៅបម្រុងទុក ៣២ បៃនៃ"shadow space" (សម្រាប់រក្សាទុក RCX, RDX, R8, និង R9 ប្រសិនបើចាំបាច់) មុនពេលហៅមុខងារនេះ។

• វាគឺជាការទទួលខុសត្រូវរបស់អ្នកហៅតំបន់ក្នុងការសម្អាតដង់បន្ទាប់ពីការហៅ និងធ្វើការត្រឡប់តម្លៃចំនួនគត់ (ស្រដៀងទៅនឹង x86) ត្រូវបានត្រឡប់មកវិញនៅក្នុង RAX ប្រសិនបើ ៦៤ប៊ីត ឬតិចជាងនេះ។

• តម្លៃត្រឡប់ទសភាគបានត្រឡប់ជា XMM0 និងតម្លៃត្រឡប់ធំជាង (រចនាសម្ព័ន្ធ) មានកន្លែងទំនេរនៅលើ Stack ដោយហៅចូលហើយ RCX បន្ទាប់មកមាន Pointer ចង្អុលទៅកន្លែងត្រឡប់នៅពេលការលែលត្រូវបានហៅ។ ចុះបញ្ជីការប្រើប្រាស់ប៉ារ៉ាម៉ែត្រចំនួនគត់ត្រូវបានរុញមួយទៅខាងស្តាំ។ RAX ត្រឡប់អាស័យដ្ឋាននេះទៅអ្នកហៅ។

•Stackត្រូវបានតម្រឹម ១៦ បៃ "ហៅ" ជំរុញឱ្យតម្លៃត្រឡប់ ៨ បៃដូច្នេះអនុគមន៍ non-leaf ដែលមិនមែនសងអស់ត្រូវកែតម្រូវ Stack ដោយតម្លៃទម្រង់ ១៦n+៨ នៅ។

• RAX, RCX, RDX, R8, R9, R10, និង R11 Register ត្រូវបានគេចាត់ទុកថា non-leaf នឹងបង្កជាហេតុហើយត្រូវចាត់ទុកថាត្រូវបានបំផ្លាញនៅពេលហៅមុខងារ។

• RBX, RBP, RDI, RSI, R12, R14, R14, និង R15 ត្រូវតែរក្សាទុកនៅក្នុងមុខងារណាមួយដោយប្រើពួកវា។

• សូមកត់សម្គាល់ថាគ្មានអនុសញ្ញាហៅសម្រាប់ការចុះឈ្មោះអណ្តែតទឹក (ហើយដូច្នេះ MMX)។

• ព័ត៌មានលម្អិតបន្ថែម (វ៉ារ៉ាហ្គាស, ការលើកលែងចំពោះការដោះស្រាយការដង់គ្នា) គឺនៅគេហទំព័រ Microsoft ។

```
func2( float a, double b, float c, double d, float e, float f );
// a in XMM0, b in XMM1, c in XMM2, d in XMM3, f then e pushed on stack
```

សំណួរ និង លំហាត់

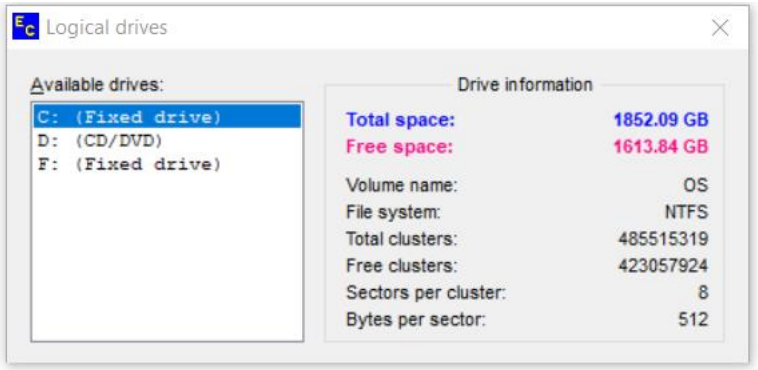
1. ដូចម្តេចដែលហៅថា(a) x86bits? (b)x64bits ? ចូរប្រៀបធៀបលក្ខណៈខុសគ្នាទាំងពីរនេះ?
2. តើ Intel Core9 មានប៉ុន្មាន bit Microprocessor? វាបង្កើតនៅឆ្នាំណា? ប្រើជាមួយនឹង Register ទំហំប៉ុន្មាន? និងប្រើជាមួយ Memory ទំហំប៉ុន្មាន GB?
3. ចូរសរសេរឈ្មោះ ៣២ប៊ីត Register ឱ្យបាន១០និងរៀបរាប់ពីមុខងាររបស់វាដោយសង្ខេប។
4. ចូរសរសេរឈ្មោះ ៦៤ប៊ីត Register ឱ្យបាន១០ និងរៀបរាប់ពីមុខងាររបស់វាដោយសង្ខេប។
5. ចូររាប់ឈ្មោះ Instructions ឱ្យបានចំនួន២០? និងរៀបរាប់ពីមុខងារនិមួយៗរបស់វា ដោយសង្ខេប ចាប់ពីស៊ីភីយូ ៨០៨៦ ដល់ Corei9 ដែលប្រើនាពេលបច្ចុប្បន្ននេះ។
6. តើគេប្រើភាសា Assembly សម្រាប់ធ្វើអ្វីការងារអ្វីខ្លះ? ចូររៀបរាប់អោយបាន ៦ ចំណុច?
7. តើកម្មវិធី X86Asm IDE ប្រើសម្រាប់ធ្វើអ្វី ?វាប្រើសម្រាប់កូដស៊ីភីយូពីឈ្មោះណាដល់ឈ្មោះណា?
8. ចូររៀបរាប់ឈ្មោះទិន្នន័យន័យប្រើជាមួយប្រមាណវិធីលេខ Signed Number, Unsigned Number និង FPU ពីស៊ីភីយូចាប់ពី ៨០៨៦ ដល់បច្ចុប្បន្ន? ចូរអោយឧទាហរណ៍បញ្ជាក់ពីករណីមួយៗ
9. តើ Ponter Instruction និង Pointer Register ខុសគ្នាយ៉ាងដូចម្តេច?
10. តើកម្មវិធី (a) Emu8086, និង (b) Proteus ប្រើសម្រាប់ធ្វើអ្វី?
11. តើការប្រើលេខអនុគមន៍ និងលេខ Interrups ប្រើសម្រាប់ធ្វើអ្វី? ចូរអោយឧទាហរណ៍បញ្ជាក់?
12. សរសេរកម្មវិធីរក Max ពីរចំនួន និង n ចំនួនដោយប្រើ x86-32 និង x86-64 Assembly?
13. សរសេរកម្មវិធីរក sum=1+2+3+...+n ពីចំនួនដោយប្រើ x86-32 និង x86-64 Assembly?
14. សរសេរកម្មវិធីធ្វើប្រមាណវិធីបូកអារ x dw 1000, 2000, 3000, 4000, , 6000 ដោយប្រើ x86-32 និង x86-64 Assembly?
15. ចូរពន្យល់ពាក្យបញ្ជា R, G, A, , FDIV, ក្នុងកម្មវិធី Debug?
16. ចូរពន្យល់ពាក្យបញ្ជា FPU និងលើកឧទាហរណ៍មួយដោះសមីកាដឺក្រេទី២
17. ចូរនិយាយពីដំហាននៃស៊ីភីយូប្រតិបត្តិកូដខាងក្រោម:

    Mov al, 12h

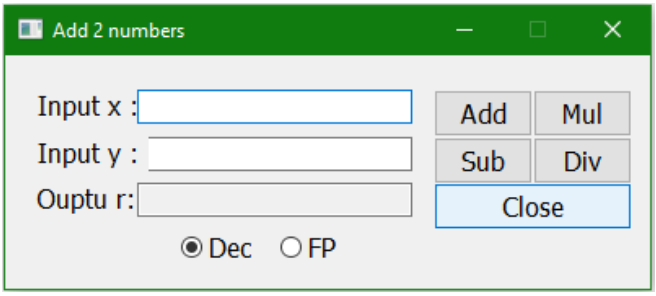
    Mov bl, 23h

    Add al, bl
18. ចូរសរសេរកម្មវិធីបង្កើត Owner OS ដើម្បីបិទម៉ាស៊ីនកុំព្យូទ័រតាម GUI App ខាងក្រោម:

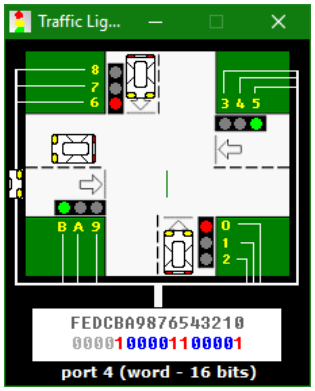
19. ចូរសរសេរកម្មវិធីទាក់ទង Driver នៃកុំព្យូទ័រតាម GUI App ខាងក្រោម:



20. សរសេរកម្មវិធីបង្កើត GUI ដូចខាងក្រោម:



21. សរសេរកម្មវិធីដើម្បីធ្វើភ្លើងចរាចរតាមរូបខាងក្រោម:



មេរៀនទី ៧ Arduino Microcontroller

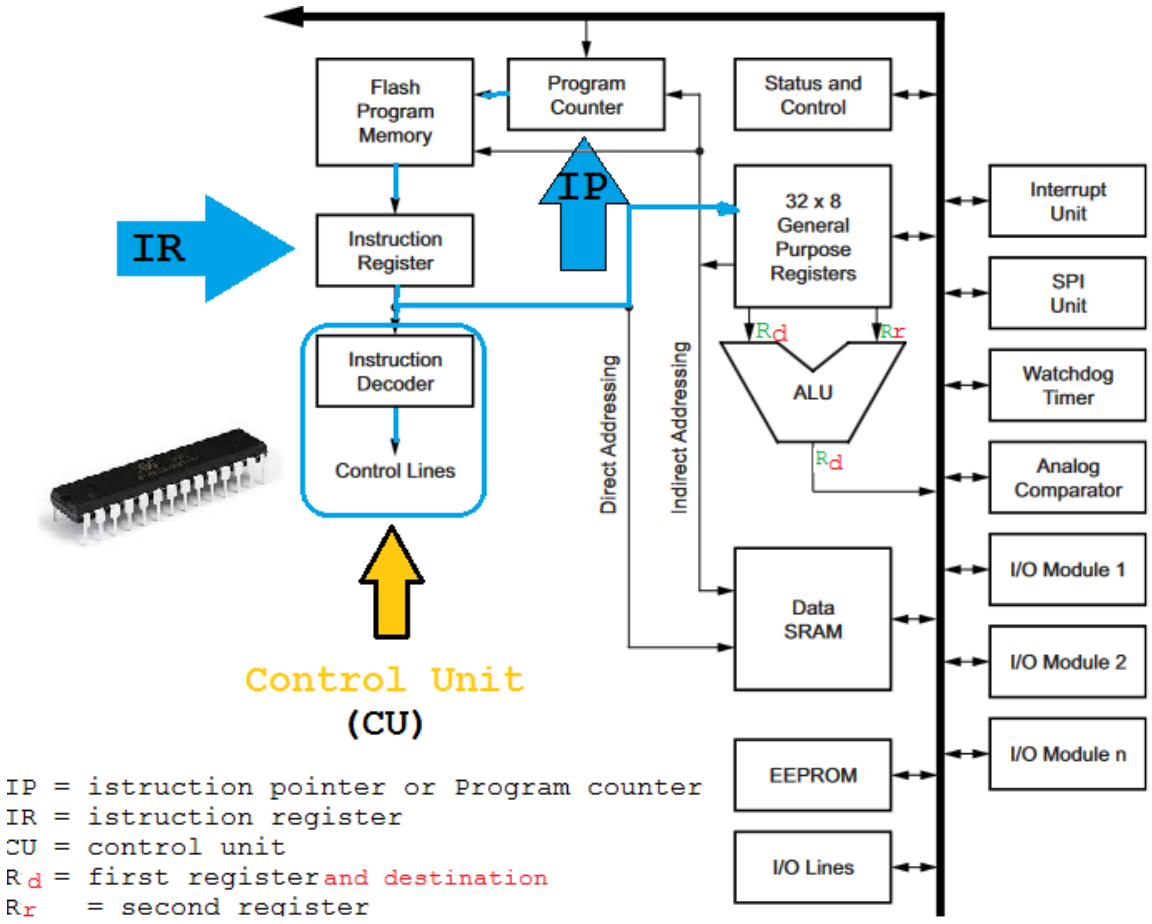
Microcontrollersជាឧបករណ៍អេឡិចត្រូវបានយើងប្រើប្រាស់វាជាច្រើននៅក្នុងជីវភាពប្រចាំថ្ងៃទោះបីពេលខ្លះយើងមិនស្គាល់វាក៏ដោយ។ វាគឺជាប្រភេទ សៀគ្វីអាយស៊ី(IC) ឬហៅម្យ៉ាងទៀតថាបន្ទះឈីប ដែលត្រូវបានប្រើជាញឹកញាប់ នៅក្នុងឧបករណ៍អេឡិចត្រូនិចប្រើប្រាស់នៅក្នុងផ្ទះ, សាលារៀន, ទីកន្លែងសាធារណៈ ឬ រោងចក្រ ដែលបានបញ្ចូលវាប្រើនៅក្នុងសៀគ្វីទូរស័ព្ទ, ទូរទស្សន៍, ម៉ាស៊ីនថតចំលង, រ៉ូបូត, ម៉ាស៊ីនផលិតរថយន្ត។ល។ យើងក៏បានសិក្សារួចមកហើយអំពី Microprocessors ដូចជា Intel 8086, 80286, 80386, 80486, Pentium I,II,II,IV, Corei3, Corei5, Core i7, Core i9 ទាំងអស់នេះជាស៊ីភីយូដែលជាខួរក្បាលរបស់កុំព្យូទ័រ។ វាទាំងអស់អាចដំណើរការទៅបានដោយសារមានពាក្យបញ្ជា សម្រាប់ធ្វើការគណនាលេខ និង បញ្ជូន/បញ្ចេញទិន្នន័យទៅទៅឌីស, ROM, RAM ឬ ឧបករណ៍ IO(Input/Output)។ ឧបករណ៍ត្រួតពិនិត្យ Microcontroller គឺជាប្រភេទនៃ Microprocessor ដែរ គ្រាន់តែវាមុខងាររផ្សេងៗពីគ្នាដូចរៀបរាប់ខាងក្រោមនេះ។

- ជាកុំព្យូទ័រតូចមួយនៅលើបន្ទះឈីប។ វាមានអង្គចងចាំហើយអាចដាក់កម្មវិធីដើម្បីធ្វើការគណនាទទួលការបញ្ចូល និងបង្កើតលទ្ធផល។
- អាចបញ្ចូលកូដទៅក្នុងឈីបនោះបានដែលយើងបានសរសេរនៅលើកុំព្យូទ័រ ដើម្បីប្រាប់វាឱ្យធ្វើអ្វីមួយដែលចងបាន។
- វាជាប្រភេទ built in "I/O" (input/output)។ ដូច្នេះវាអាច read នូវតម្លៃឌីជីថល 0 និង ១និងតម្លៃ analog មានពីតម្លៃលេខ 0 ដល់ ១០២៤។ វាអាចតភ្ជាប់ដោយផ្ទាល់ជាមួយនឹងសមាសភាគអេឡិចត្រូនិចមានដូចជា switches, buttons, LCD displays, LEDS, Digital Electronic, Relays, Sensor, Keypad, camera និង serial ports។
- ជាទូទៅវាប្រើប្រាស់សម្រាប់ត្រួតពិនិត្យទៅលើឧបករណ៍តូចៗដូចបានរៀបរាប់ខាងលើ ផ្ទុយពីការប្រើ microprocessors សម្រាប់ប្រើប្រាស់ជាមួយនឹងកុំព្យូទ័រដែលគ្រប់គ្រងដោយផ្នែកទន់ដែលមានស្រាប់។
- ត្រូវបានប្រើជាញឹកញាប់នៅក្នុងឧបករណ៍ចល័ត ដែលដំណើរការប្រើថ្មឬអាគុយដូចម៉ាស៊ីនថតឌីជីថល។ ដូច្នេះពួកវាច្រើនតែមានថាមពលទាបជាមួយនឹងការប្រើប្រាស់ចរន្តតូច។
- បើប្រៀបធៀប microprocessor នៅក្នុង PC ប្រើ Memory គិត GB ដូចជា DDR4 Ram មានទំហំ 8 ឬ ១៦ GB ឬលើសពីនេះ។ ចំណែកឯ SRAM ប្រើក្នុង microcontroller អាចមានទំហំពី 1 ដល់ 64 KB
- កម្មវិធីអាចផ្ទុកនៅក្នុង EPROM ឬ EEPROM ជាប្រភេទ volatile (កម្មវិធីមិនបាត់នៅពេលឧបករណ៍ត្រូវបានបិទ) អង្គចងចាំដែលអាចត្រូវបានលុបចោលបានជាបន្តបន្ទាប់និងសរសេរឡើងវិញ។ ប្រភេទនៃ Microcontroller មានដូចជា 8051, AVR, PIC, RPi,….

1. ស្ថាបត្យកម្ម Arduino

គឺស្ថាបត្យកម្មផ្នែករឹង(Hardware) និងវាអាចដំណើរការទៅបានត្រូវមានកម្មវិធី ឬហៅថាកម្មវិធី Arduino និង ទិន្នន័យនៃកម្មវិធីត្រូវផ្ទុកនៅក្នុងអង្គចងចាំដោយឡែកៗពីគ្នា។ វាមានអង្គចងចាំចំនួន២ ប្រភេទសម្រាប់ធ្វើការចងចាំកម្មវិធី និងអង្គចងចាំសម្រាប់ទិន្នន័យ។ ម្ល៉ោះហើយទិន្នន័យត្រូវបានរក្សាទុកនៅក្នុងអង្គចងចាំ SRAM និងកូដត្រូវបានរក្សាទុកនៅក្នុងប្រភេទ Flash Memory។ ចំពោះទិន្នន័យជាអចិន្ត្រៃយ៍វិញនោះត្រូវផ្ទុកនៅក្នុង EEROM។

វាជាប្រភេទ Microcontroller ដែលមាន CU, ALU, Register, I/O port, RAM, ROM, EEROM និង Clock នៅក្នុង Chip(IC) តែមួយ។



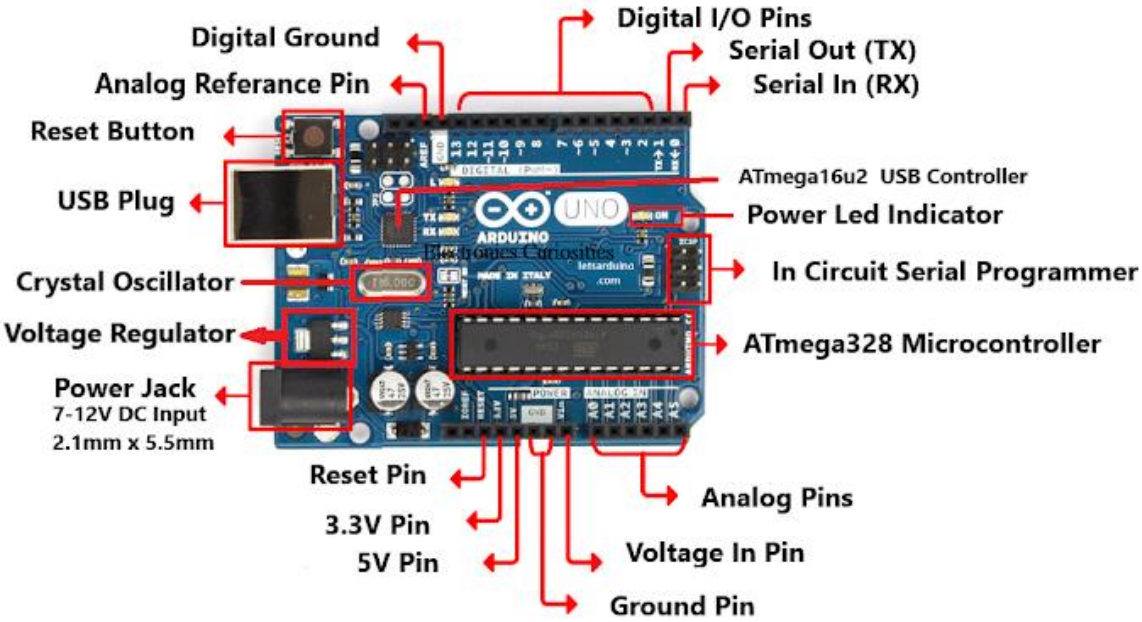
2. Arduino Board

វាជាបន្ទះដំរអេឡិចត្រូនិច ឬហៅថាជាធុងប្រព័ន្ធ (System Unit) មួយប្រើសម្រាប់ធ្វើការដំណើរការនូវសមាសភាគអេឡិចត្រូនិចផ្សេងៗ ដែលបានតភ្ជាប់ជាមួយនឹងវា។ សមាសភាគអេឡិចត្រូនិចរួមមានដូចជា Keyboard, Keypad, LCD, button, Sensor ឬឧបករណ៍បញ្ចេញនិងបញ្ចូល(Input&Oupt) ។ ចំពោះការតភ្ជាប់ពីកុំព្យូទ័រទៅ Arduino Board ដោយប្រើខ្សែ USB ដើម្បីបញ្ចូលកូដ និងទិន្នន័យទៅវិញទៅមកបានរវាងឧបករណ៍ទាំងពីរនេះ។ ចំពោះប៉ត(Board)ដែលត្រូវប្រើមានច្រើនប្រភេទ និងអាចជ្រើសរើសយកមកប្រើទៅតាមការធ្វើគម្រោង និង តំរូវការអ្នកប្រើប្រាស់។

2.1. Arduino Uno R3 គឺជាប្រភេទ mainboard មួយដែលមានមូលដ្ឋានគ្រឹះលើស៊ីភីយូដែលមានឈ្មោះ ATmega328P ។ វាមានតួនាទីផ្ទុកកូដ និងទិន្នន័យដែលបានបញ្ចូលពីកុំព្យូទ័រសម្រាប់ធ្វើការប្រតិបត្តិដើម្បីបញ្ជាទៅលើឧបករណ៍ផ្សេងៗ ដែលបានភ្ជាប់ជាមួយនឹងវាដើម្បីឱ្យដំណើរការទៅបាន។ ខាងក្រោមនេះជាសមាសភាគផ្សំសំខាន់ៗដែលប្រើប្រាស់នៅលើបន្ទះ Arduino Uno:

- CPU ឈ្មោះ ATMEGA 328P បង្កើតដោយក្រុម ATMEL និងមានល្បឿន 16 Mhz quartz crystal
- វាមាន 14 Digital input/output pin មានលេខពី ០ ដល់លេខ ១៣ និងប្រើសម្រាប់បញ្ចូលនិងបញ្ចេញតម្លៃ ០(OFF) និង ១(ON)។ វាជាកន្លែងមួយបញ្ចេញ និងបញ្ចូលទិន្នន័យ។ ហើយម៉្យាងទៀតអាចប្រើ Pin ទាំងនេះជាប្រភេទ Pwm បញ្ចេញឬបញ្ចូលតម្លៃលើសពី១។
- 6 analog inputs Pin មានពីអក្សរ A0 ដល់ A6 អាចប្រើបានពីរករណីទី១ដូច Digital Pin និងទី២ប្រើសម្រាប់បញ្ចូលតម្លៃ Analog ពី ០ ទៅ ១០២១ ឬ០ ដល់ ២៥៦ អាស្រ័យទៅតាមប្រភេទនៃការប្រើប្រាស់ឧបករណ៍ជាក់ស្តែង។

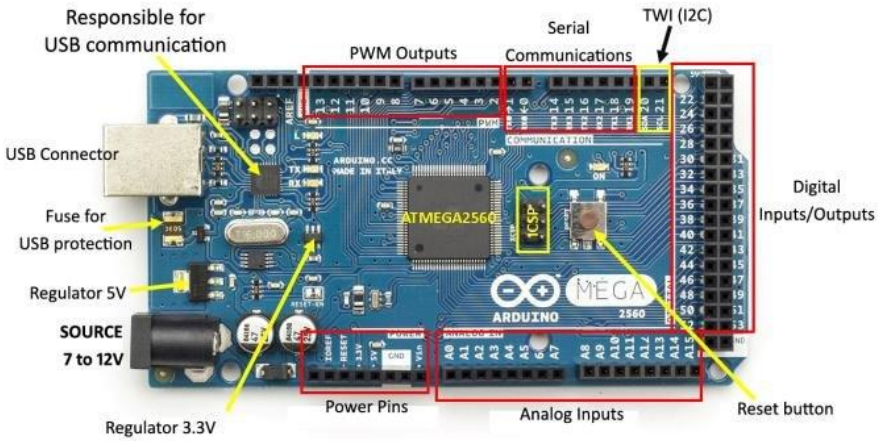
- USB Port សម្រាប់ភ្ជាប់ USB cable ពីកុំព្យូទ័រទៅកាន់ប៉ត Arduino
- មាន power jack មួយសម្រាប់ភ្ជាប់ភ្លើង ១២ វុល ឬក្រោមនេះ
- មាន ICSP header មួយប្រើជាមួយនឹង USBasp Base ឬ CP2102 Programmer សម្រាប់បញ្ចូលកូដដោយមិនចាំបាច់ប្រើ UPB Port។
- មានកុងតាក់សម្រាប់កំណត់ reset មួយ ពេលប៉តមានបញ្ហា និងមានអំពូល LED មួយប្រើជាមួយនឹង Pin13 និង SRAM មានទំហំ 2 KB រួមនឹង Serial Buffer ទំហំ ៦៤ ប៊ីត
- **Flash Memory** =32 Kb , EEPRO :1024 byte = **1** KB



2.2. Arduino Mega 2560

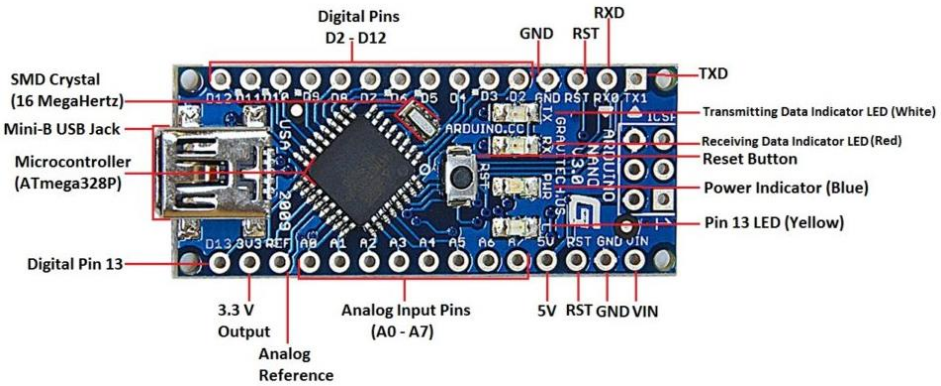
គឺជា Arduino Board មានបន្ទះសៀគ្វីទំហំធំជាងប៉ត Arduino និង មាន Digital និង Analog Pin ច្រើនជាង និងទំហំអង្គចងចាំកាន់តែធំ។ វាមានមុខងារប្រហាក់ប្រហែលនឹងការប្រើដូច Arduino UNO ដែរ។ រូបខាងក្រោមគឺជា Arduino Mega 2560:

- CPU ឈ្មោះ ATMEGA 2560 បង្កើតដោយក្រុម ATMEL និងមានឈ្បឿន 16 Mhz quartz crystal
- 54 Digital input/output pin មានលេខពី 0 ដល់លេខ ៥៣ និងប្រើសម្រាប់បញ្ចូលនិង បញ្ចេញតម្លៃ 0(OFF) និង ១(ON)។ បើជើងពីនទាំងនេះមានតម្លៃលេខ១ ន័យក្នុងចរន្តអគ្គិសនីមានភ្លើង ៥ វុល។ បើជើងពីនទាំងនេះមានតម្លៃលេខ0 ន័យក្នុងចរន្តអគ្គិសនីគឺគ្មានមានភ្លើង។
- 16 analog inputs មានពីអក្សរ A0 ដល់ A16 អាចប្រើបានពីរករណីទី១ដូច Digital Pin និងទី២ប្រើសម្រាប់បញ្ចូលតម្លៃ **Analog** ពី 0 ទៅ ១០២១ ឬ0 ដល់ ២៥៦។
- USB Connection សម្រាប់ភ្ជាប់ USB cable ពីកុំព្យូទ័រទៅកាន់ប៉ត Arduino
- មាន power jack មួយសម្រាប់ភ្ជាប់ភ្លើងចន្លោះពី ៧ ទៅ ១២ វុលឬលើសពីនេះ
- មាន ICSP header ប្រើជាមួយនឹង USBasp ឬ CP2102 សម្រាប់បញ្ចូលកូដ
- កុងតាក់សម្រាប់កំណត់ reset ពេលមានបញ្ហា
- SRam = 8 KB ,Flash Memor = 256 Kb និង EEPRO : 4 KB



2.3. Arduino Nano

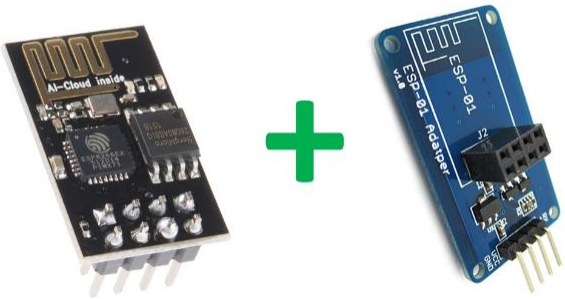
វាគឺជាប្រភេទ Microcontroller មួយមានបន្ទះសៀគ្វីទំហំតូចដាងគេ និងមានមូលដ្ឋានគ្រឹះលើស៊ីភីយូ ATmega328 (Arduino Nano 3. x)។ វាមានមុខងារប្រហាក់ប្រហែលទៅនឹង Arduino Uno R3 និងអត់មាន DC Power jack ហើយដំណើរការជាមួយខ្សែ Mini-B USB ខ្សែធម្មតា។



Microcontroller	ATmega328P
Architecture	AVR
Operating Voltage	5 Volts
Flash Memory	32 KB of which 2 KB used by Bootloader
SRAM	2KB
Clock Speed	16 MHz
Analog I/O Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 milliAmps
Input Voltage	(7-12) Volts

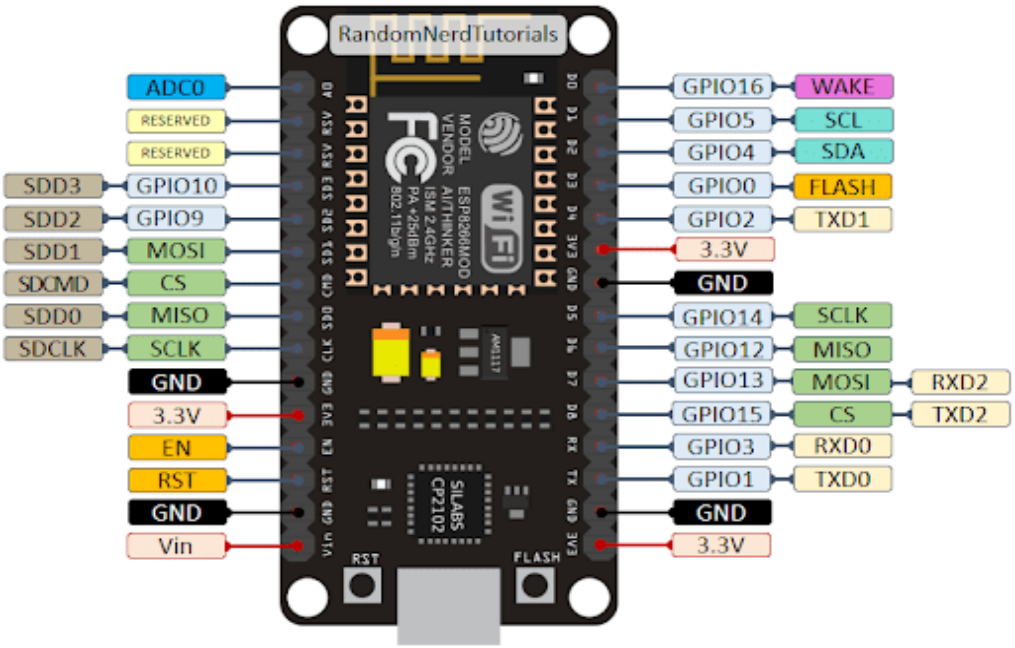
2.4. ESP8266 និង NodeMCU

2.4.1. **ESP8266** គឺជាឧបករណ៍ដែលបង្កើតឡើងពីប្រព័ន្ធ Soc ដែលមានភ្ជាប់ TCP/IP ដែលអនុញ្ញាតឲ្យ microcontroller ដែលភ្ជាប់នឹងវាមានសមត្ថភាពប្រើប្រាស់បណ្តាញ Wifi បាន។ វាក៏អាចមានសមត្ថភាពទំនាក់ទំនងកម្មវិធី ឬ បិទ ប្រព័ន្ធ WIFI ទាំងមូលបានផងដែរ។ វាក៏មានមុខងារអាចបង្កើតនូវ Soft_AP(Access point) សម្រាប់ឲ្យយើងអាចភ្ជាប់ពីឧបករណ៍ផ្សេងបញ្ជាមកកាន់ Microcontroller ដោយមិន ត្រូវការ Internet ដូចជា Bluetooth ផងដែរ។ វាក៏អាចប្រើដូច Arduino Uno ដែរ។



2.4.2. NodeMCU ESP 8266

វាជាប្រភេទ Open source ដោយប្រើកម្មវិធីបង្កប់ដែលមានឈ្មោះថា Firmware និង ការអភិវឌ្ឍន៍ត្រូវបានកំណត់នូវគោលដៅជាពិសេសសម្រាប់កម្មវិធី ដែលមានមូលដ្ឋានលើការប្រើប្រាស់ IoT។ វារួមបញ្ចូលទាំងកម្មវិធីបង្កប់ដែលដំណើរការលើប្រព័ន្ធ ESP8266 Wifi_SoC ពីប្រព័ន្ធ Espressif និងផ្នែករឹងដែលផ្អែកលើ ម៉ូឌុល ESP-12។ NodeMCU 8266 ប្រើ CPU ឈ្មោះ **ESP8266**(LX106) និងប្រព័ន្ធប្រតិបត្តិការ XTOS ដែលប្រភេទ Microcontrollers និងមានមុខងារប្រើដូច Arduion Uno ឬ Arduino Nano ដែរ។



NodeMCU ESP8266 Specifications & Features

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- Small Sized module to fit smartly inside your IoT project

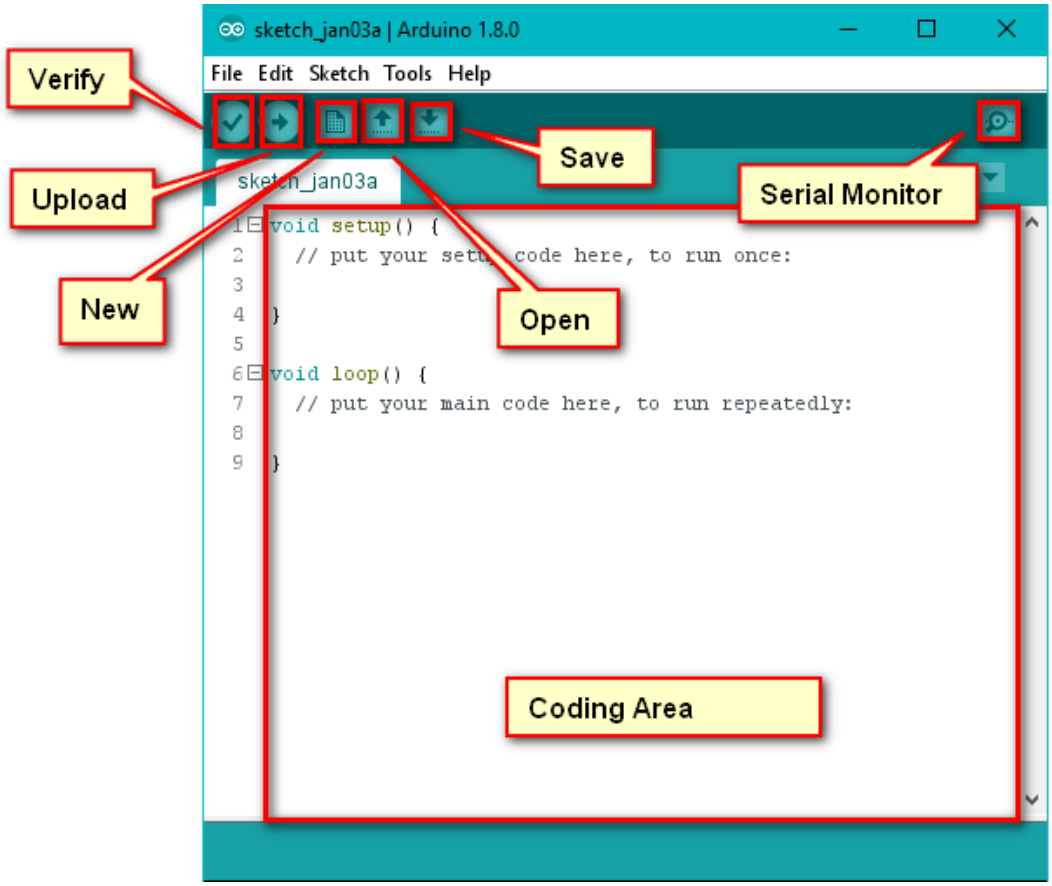
▪ Operating Voltage: 3.3V ▪ Input Voltage: 7-12V ▪ Digital I/O Pins (DIO): 16 ▪ Analog Input Pins (ADC): 1 ▪ UARTs: 1	▪ SPIs: 1 ▪ I2Cs: 1 ▪ Flash Memory: 4 MB ▪ SRAM: 64 KB ▪ Clock Speed: 80 MHz ▪ PCB Antenna

3. IDE(integrated development environment)

IDE ជាកម្មវិធីមួយសម្រាប់ឱ្យអ្នកប្រើប្រាស់សរសេរកូដ ឬបង្កើតកម្មវិធី Arduino ដើម្បីធ្វើការបញ្ជាទៅលើឧបករណ៍អេឡិចត្រូនិច, ឧបករណ៍វៃឆ្លាត និងរ៉ូបូត។ ដោយនៅលើកម្មវិធីនេះយើងអាចសរសេរភាសា Arduino និង Assembly និងបញ្ចូលទៅក្នុងឈីប ATMEGA328P ដែលមាននៅលើ Arduino Board។ វាក៏ជាកម្មវិធីសម្រាប់ឱ្យអ្នកសរសេរកម្មវិធី សរសេរកូដដើម្បីដោះស្រាយចំណោទ ឬបញ្ហាមួយដែរដូចដែលយើងប្រើនៅក្នុង C/C++, C#, Java ឬ Python ជាដើម។

3.1. Arduino IDE

IDE មានន័យថា Integrated Development Environment (**IDE**) គឺជាកម្មវិធី ដែលប្រើបានជាមួយនឹង Windows, macOS, Linux ដោយប្រើភាសា Arduino និង Assembly។ វាប្រើសម្រាប់សរសេរកូដទាក់ទងនឹងកម្មវិធីបញ្ជាឧបករណ៍ឌីជីថល, ឧបករណ៍វៃឆ្លាត និងរ៉ូបូត។ វាអាចប្រើជាមួយនឹងជាមួយនឹងប៉ុតជាច្រើនដូច Arduino Uno, Arduino Mega 2560, ESPO 32, NANO, Clone Board,។ល។



Verity : ប្រើផ្ទៀងផ្ទាត់ ឬប្រើជា Compile សម្រាប់ធ្វើការបំលែង Source code

upload : ប្រើសម្រាប់ Compile និងupload sketch ទៅ Arduino board.

new: ប្រើសម្រាប់បង្កើតផ្ទៃ Editor ថ្មី

Open: ប្រើសម្រាប់បើកកូដ ថ្មី

Save: ប្រើសម្រាប់រក្សាទុក sketch.

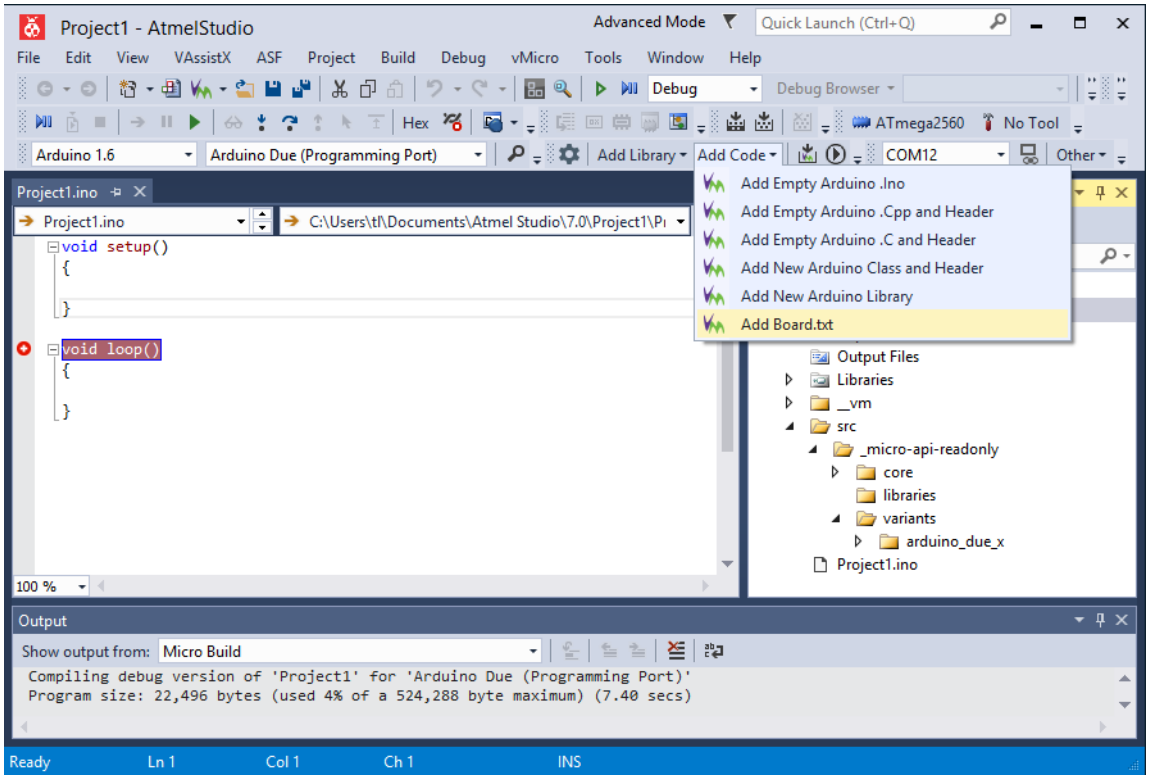
Serial Monitor: បើក serial monitor ដើម្បីផ្ញើទិន្នន័យពីកុំព្យូទ័រទៅ Arduino Board

3.2. Python IDE

Python IDE ប្រើសម្រាប់សរសេរភាសា Python ហើយមានច្រើនប្រភេទ ដែលគេប្រើនាពេលបច្ចុប្បន្ននេះ ដូចជា IDLE, ជាច្រើនប្រភេទ Thonny, PyChar, Visual Studion Code, Jupyter,...។ Editor ទាំងនេះអនុញ្ញាតឱ្យអ្នកសរសេរកូដធ្វើការសាកល្បង, ប្រើ Debug សម្រាប់មើលពីដំណើរការនៃការប្រតិបត្តិកូដ និងអាចសរសេរកម្មវិធីដើម្បីបញ្ជាក់ ឧបករណ៍អេឡិចត្រូនិច ឬរ៉ូបូតដូចនឹង ភាសា Arduino ដែរ។

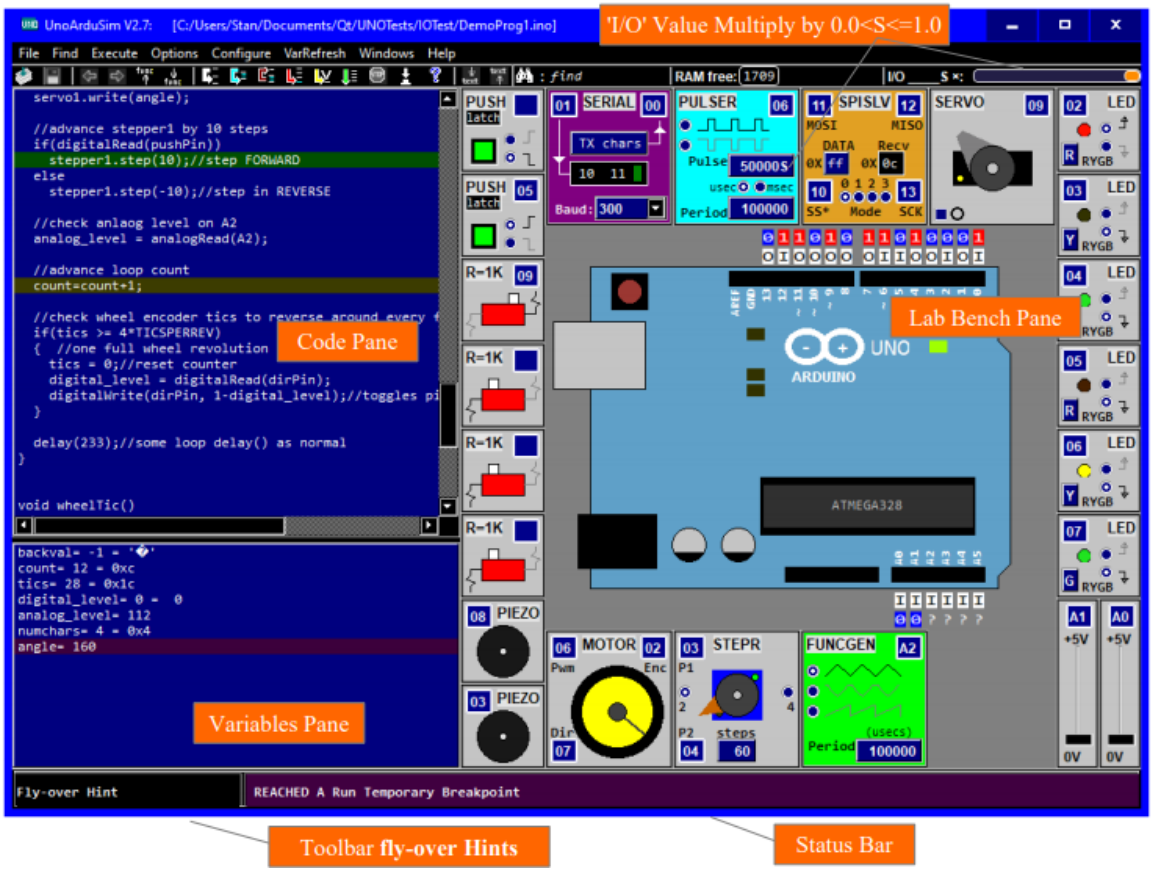
3.3. ATMEL STUDIO IDE

ប្រើសម្រាប់ប្រើភាសា Assembly ឬ C/C++ និងអាចសរសេរកម្មវិធីដើម្បីបញ្ជា ឧបករណ៍អេឡិចត្រូនិច ឬរ៉ូបូតដូចនឹង ភាសា Arduino ដែរ។



3.4. UnoArduSim IDE

វាជាប្រភេទកម្មវិធីកុំព្យូទ័រដែលអនុញ្ញាតឱ្យយើងធ្វើការសរសេរកូដដោយប្រើភាសា Arduino ដើម្បីដោះស្រាយចំណោទឬបញ្ហាផ្សេងៗ ដូចនៅក្នុងភាសា Assembly ឬ C ដែរនិងធ្វើការកុងត្រូល និងបញ្ជាទៅលើឧបករណ៍សិប្បនិម្មិតមួយចំនួនធំដូចដែលបានរៀបរាប់នៅចំនុចខាងក្រោម យើងអាចសរសេរកម្មវិធីនិងកែសម្រួលដោយប្រើ៖



- បើក Edit/View ដោយគ្រាន់តែចុចពីរដងនៅផ្ទៃកូដផេន(Code Pane) ដើម្បីធ្វើការសរសេរកម្មវិធីផ្សេងៗ។ និងត្រលប់ទៅកូដផេនវិញចុចលើ Accept ដើម្បីធ្វើការដំណើរការកូដនៅលើផ្ទៃនេះដោយប្រើបញ្ជា Run ដើម្បីធ្វើការបញ្ជាទៅលើឧបករណ៍ផ្សេងៗ

- យើងដំណើរការម្តងមួយជំហានៗដើម្បីពិដំណើរការកូដតាមបញ្ជានិមួយៗដោយចុចលើ Step-Into , បញ្ឈប់ដំណើរការកម្មវិធីដោយចុចលើ stop ឬវិសេតកូដនៅពេលកូដមានបញ្ហាកើតឡើងវិញចុចលើ Reset ។

- UnoArduSim ផ្តល់នូវឧបករណ៍សិប្បនិម្មិតដែលសំខាន់ជាងគេគឺ Arduino Uno និង Arduino Mega2560 និង ឧបករណ៍បញ្ចូលឬបញ្ចេញ IO និង Interface ជាប្រភេទច្រើនប្រភេទទៀត អនុញ្ញាតឱ្យអ្នកប្រើប្រាស់ជ្រើសរើសមកប្រើ។ ឧបករណ៍ទាំងនេះមានដូចជាម៉ូទ័ DC, ម៉ូតូ, Button, stepper- ម៉ូទ័រ, និងម៉ូទ័រ servo, ការបង្ហាញ TFT, I2C, ឌីជីថល, ម៉ាស៊ីនបង្កើតរលកអាណាឡូក, ឧបករណ៍បំពងសំឡេង piezo, អំពូល LED និង Interface មួយចំនួនទៀត។ រូបខាងក្រោមជាផ្ទៃផេនសម្រាប់ធ្វើការសរសេរកូដ។

4. **រចនាសម្ព័ន្ធ Arduino**

មូលដ្ឋានគ្រឹះក្នុងការបង្កើតអនុគមន៍មួយត្រូវគិតទៅលើបីចំនុច។ ចំនុចទី១ ឈ្មោះអនុគមន៍និមួយៗត្រូវមានឈ្មោះពិសេសទៅតាមបញ្ហាងាយក្នុងការចំណាំ ស្រួលប្រើ និងបញ្ចប់ដោយសញ្ញា ( ) ដើម្បីបញ្ជាក់ថាឈ្មោះនៅពីក្រោយសញ្ញានេះជាឈ្មោះអនុគមន៍។ ចំនុចទី២ អនុគមន៍ត្រូវមានការឲ្យតម្លៃ ឬគ្មានឲ្យតម្លៃអាស្រ័យលើការប្រើ Return Type។ ហើយចំនុចទី៣ជារូបរាង នៃអនុគមន៍ដែលត្រូវប្រើសញ្ញា { } ដែលយើងអាចសរសេរពាក្យបញ្ជា មួយឬច្រើននៅក្នុងសញ្ញានេះ។

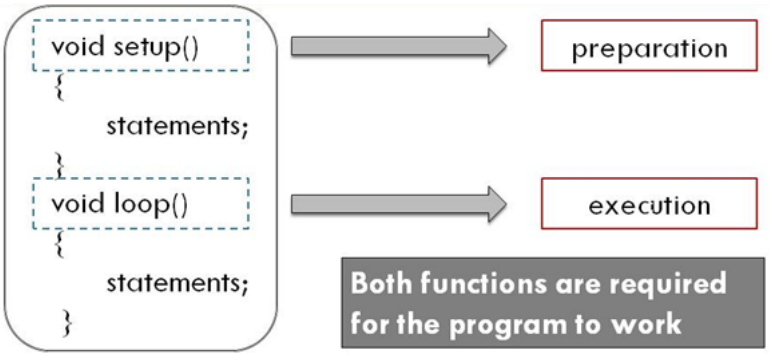
ដើម្បីសរសេរកូដក្នុង UnoArdSim IDE ត្រូវចេះអំពីរចនាសម្ព័ន្ធនៃភាសា Assembly ឬ Arduino ដែលបានរៀននៅមេរៀនទីមុខ និងមេរៀនបន្ទាប់។ ហើយការសរសេរកម្មវិធីដំបូងត្រូវស្គាល់អនុគមន៍សំខាន់ៗចំនួនពីគឺ setup( ) និង loop( ) និងអនុគមន៍បន្តបន្ទាប់ទៀតដែលមាននៅក្នុងមេរៀននេះសម្រាប់ធ្វើការ តេស្តទៅលើឧបករណ៍បញ្ចេញឬបញ្ចូលមួយចំនួន។

4.1. **អនុគមន៍** setup( )

ប្រើសម្រាប់សរសេរបញ្ជា (Instructions) សម្រាប់ដោះស្រាយចំណោទដូចដែលយើងបានសរសេរនៅក្នុងភាសា Assembly, ការកំណត់តម្លៃដំបូង, កំណត់ជើង Pin នៃប៉ត Arduino និង ជាប់ទាក់ទងនឹងការដំឡើងផ្នែករឹងណាមួយ នៃសមាសភាគអេឡិចត្រូនិចផ្សេងៗ។ នៅក្នុងអនុគមន៍នេះយើងប្រើ Serial Monitor, LCD, TFT និងឧបករណ៍ដទៃទៀត ជាអេក្រង់បញ្ចេញ និងបញ្ចូលតម្លៃរបស់កូដមួយ។ វាប្រតិបត្តិការតែម្តងនៅពេលដំណើរការ(Run)កូដ ឬ ចុច Reset Key នៅលើប៉ត Arduino។

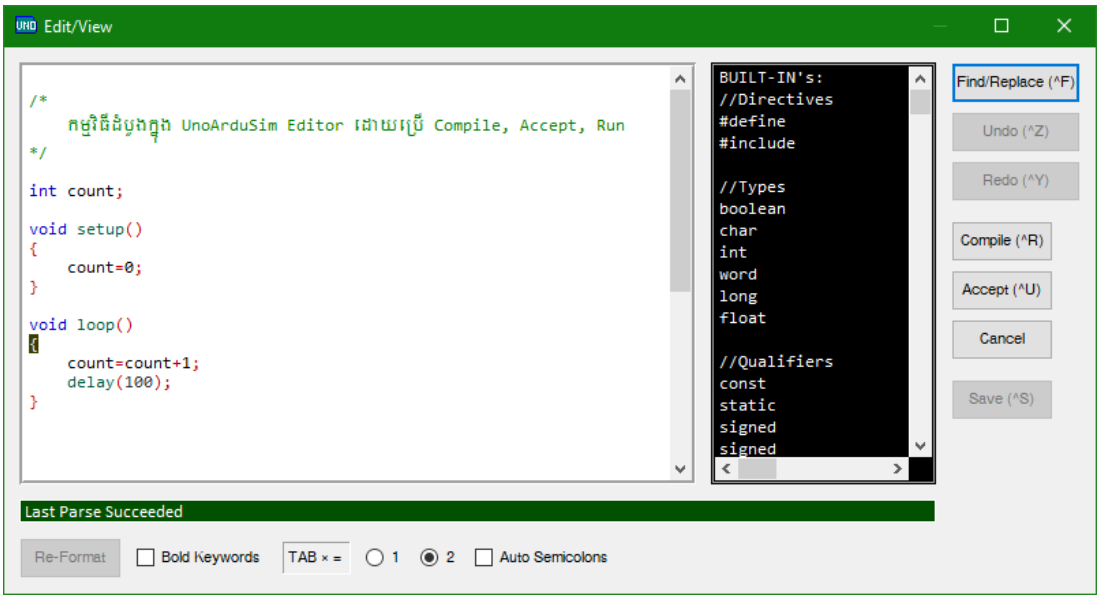
4.2. **អនុគមន៍** loop( )

សម្រាប់សរសេរសំនុំពាក្យបញ្ជា ឬហៅថា main code ដើម្បីធ្វើការបញ្ជាទៅលើផ្នែករឹង ឬដោះស្រាយបញ្ហាណាមួយត្រូវសរសេរវាក្នុងអនុគមន៍នេះ។ វាប្រតិបត្តិបញ្ជាមួយឬច្រើនម្តងហើយម្តងទៀតជារៀងរហូតគ្មានឈប់ ដូចដែលយើងប្រើបាន while(1) ក្នុងភាសា C ដែរ។ ដើម្បីសរសេរកម្មវិធី Arduino ទៅបានត្រូវមានអនុគមន៍ពីរខាងលើ និង អាចបង្ហាញគំរូដូចខាងក្រោម៖



5. **កម្មវិធីប្រើ** Edit/View

យើងអាចសរសេរ រួមទាំងកែសម្រួលកម្មវិធីដោយប្រើអេក្រង់ដូចបានបង្ហាញតាមរូបខាងក្រោមដោយគ្រាន់តែចុចពីរដងនៅផ្នែកូដផេន(Code Pane)។



- **Compiter:** ប្រើសម្រាប់បំលែងកម្មវិធីឱ្យទៅជាម៉ាស៊ីនកូដ
- **Accept:** ប្រើសម្រាប់បញ្ចូលកូដទៅក្នុងកូដផេន(Code Pane)។ ក្នុងករណីបើយើងប្រើប៉ត Arduino ពិតប្រាកដមានន័យថាបញ្ចូលកូដទៅក្នុងផ្ទុក Flash Memory ដែលមានទំហំ ៣២ KB នៃ ស៊ីភីយូ ATMEGA328។ ពេលកម្មវិធីនៅក្នុងកូដផេនទើបយើងអាចដំណើរការបាន ដោយចុចលើ ។

6. ការប្រើប្រាស់សៀរៀលម៉ូនីទ័រ

សៀរៀល**ម៉ូនីទ័រ**ត្រូវបានប្រើសម្រាប់ធ្វើការការប្រាស្រ័យទាក់ទងរវាងប៉ត Arduino និងកុំព្យូទ័រឬឧបករណ៍ផ្សេងទៀត។ គ្រប់ Arduino ប៉តទាំងអស់មានយ៉ាងហោចណាស់ផតសៀរៀលមួយ (ត្រូវបានគេស្គាល់ថាជា UART ឬ USART): សៀរៀល។ វាប្រាស្រ័យទាក់ទងនៅលើជើងពីនឌីជីថល ០ (RX) និង ១ (TX) ក៏ដូចជាជាមួយកុំព្យូទ័រតាមរយៈយូអេសប៊ី។ ដូច្នេះប្រសិនបើអ្នកប្រើមុខងារទាំងនេះអ្នកក៏មិនអាចប្រើពីន ០ និង ១ សម្រាប់បញ្ចូលឬចេញឌីជីថលបានដែរ។

សៀរៀលម៉ូនីទ័រ (Serial Monitor) ជាផ្ទៃអេក្រង់បញ្ចេញលទ្ធផល ឬហៅម្យ៉ាងទៀតថាជា Output Screen ដូចដែលយើងប្រើនៅក្នុងភាសា Assembly ឬ C ដែរ។ វាមុខងារសម្រាប់បង្ហាញនូវលទ្ធផល និងព័ត៌មានផ្សេងៗ ពេលកំពុងដំណើរកូដនៅក្នុង Arduino IDE។ ដើម្បីប្រើប្រាស់វាបាន ត្រូវអនុវត្តតាមជំហានបន្តបន្ទាប់ដូចខាងក្រោម៖

1. ត្រូវភ្ជាប់ Arduino Board ជាមួយនឹងកុំព្យូទ័រសិនដោយប្រើខ្សែ USB។ ហើយត្រង់ Baud Rate ត្រូវជ្រើសកំណត់តម្លៃឱ្យដូចទៅនឹងកូដរបស់យើង ឬសមមាត្រនឹងប៉តដែលប្រើ។

2. Serial.println ជាអនុគមន៍ឬបញ្ជាមួយប្រើសម្រាប់ធ្វើការបង្ហាញតម្លៃលេខ ឬ String នៅលើសៀរៀល។ វាផ្សំពី serial Object ដែលគេបានបង្កើតជាស្រេចនៅក្នុង Arduino IDE និងអនុគមន៍ println សម្រាប់បង្ហាញព័ត៌មាននៅលើសៀរៀលនិងធ្វើការចុះបន្ទាត់ដោយខណ្ឌពីគ្នាដោយសញ្ញា . ។

3. Baud Rate: គឺជាល្បឿននៃការបញ្ជូនទិន្នន័យរវាងកុំព្យូទ័រ និងប៉ត Arduino ដោយមានការបើក Port និងកំណត់ល្បឿនបញ្ជូនទិន្នន័យ ៩៦០០ bits ក្នុងវិនាទី។

ចំណែកឯការប្រើសៀលរៀលម៉ូនីទ័រនៅក្នុង UnoArduSim IDE។ វាមានប្រអប់ TX សម្រាប់បញ្ចូលទិន្នន័យ និងធ្វើការផ្ញើដោយចុចលើពាក្យ TOSEND ដើម្បីយកទិន្នន័យទៅផ្ទុកក្នុងអង្គចងចាំក្នុងឈីប ATMEGA328P នៃបន្ទះ Arduino។ អេក្រង់ RX មានផ្ទៃពណ៌ខ្មៅសម្រាប់បង្ហាញលទ្ធផលពេលកម្មវិធីមួយបានដំណើរការរួច។

6.1. អនុគមន៍ Serial.begin( )

បង្កើតការទំនាក់ទំនងសៀរៀលរវាងបន្ទះ Arduino របស់អ្នកនិងឧបករណ៍ផ្សេងៗទៀត។ ការប្រើប្រាស់ទូទៅបំផុត នៃការទំនាក់ទំនងសៀរៀលដែលអ្នកនឹងបង្កើតរវាងប៉ត Arduino និងកុំព្យូទ័ររបស់អ្នកតាមរយៈខ្សែយូអេសប៊ី។

នៅពេលដែលអ្នកបានបង្កើតទំនាក់ទំនងសៀរៀលរវាងឧបករណ៍ពីរវាអនុញ្ញាតឱ្យឧបករណ៍ទាំងពីរអាចទំនាក់ទំនងដោយប្រើកម្មវិធីសៀរៀល។

6.2. Serial.print( ) និង Serial.println( )

វាមុខងារធ្វើការបង្ហាញទិន្នន័យទៅសៀរៀលម៉ូនីទ័រជាអត្ថបទ ASCII ដែលមនុស្សអាចអានបាន។ ពាក្យបញ្ជានេះអាចមានទម្រង់ជាច្រើន។ លេខត្រូវបានបោះពុម្ពដោយប្រើតួអក្សរ ASCII សម្រាប់ខ្ទង់នីមួយៗ ហើយសរសេរវានៅក្នុង " "។

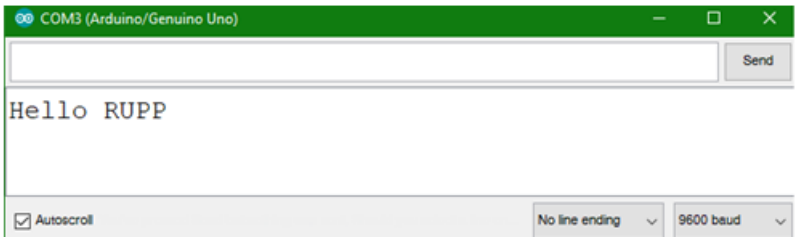
Serial.print(val)

Serial.print(val, format)

➧ឧទាហរណ៍ទី១: កម្មវិធីគំរូខាងក្រោម ធ្វើការបង្ហាញឃ្លា Hello RUPP នៅសៀរៀល Monitor

```
void setup( ) {
        Serial.begin( 9600 );//បើក usb port និងកំណត់អត្រាទិន្នន័យគិតជា9600ប៊ីត
        ក្នុងមួយវិនាទីសម្រាប់ការបញ្ជូលទិន្នន័យឱ្យសៀរៀល
        Serial.println( "Hello RUPP" );//បង្ហាញតួអក្សរនៅលើសៀរៀល Monitor
}
void loop( ) {}
```

Serial Monitor Output



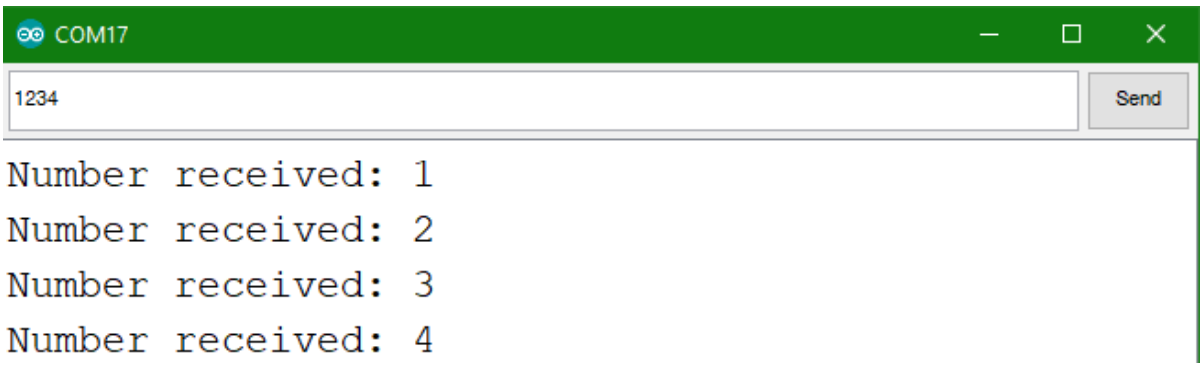
➧ឧទាហរណ៍ទី២: កម្មវិធីខាងក្រោមជាការបូកលេខពីរចំនួនដោយតម្លៃពីបញ្ចូលនៅលើ Serial Monitor:

```
int x; បង្កើតអញ្ញាតិ x សម្រាប់បញ្ចូលតម្លៃចំនួនគត់
int y; បង្កើតអញ្ញាតិ y សម្រាប់បញ្ចូលតម្លៃចំនួនគត់
int r; បង្កើតអញ្ញាតិ r សម្រាប់បញ្ចូលតម្លៃចំនួនគត់
void setup( ) {
  // put your setup code here, to run once:
   Serial.begin( 9600 ); //បើក usb port និង កំណត់អត្រាទិន្នន័យគិត9600ប៊ីតក្នុងមួយ
   ;វិនាទី (បញ្ជូន)ដែលជាល្បឿនបញ្ជូលទិន្នន័យរវាងឧបករណ៍ពីរ
   Serial.println( "Sum of two number" );  សម្រាប់បង្ហាញតួអក្សរលើserial monitor
   Serial.print( "Enter x=" );បង្ហាញពាក្យ Enter  ទៅក្នុងសឺរៀល
  while( Serial.available( )==0 ){ } រងចាំការចុចឃីហើយផ្ញើតម្លៃទៅរក្សាទុកBuffer
      x=Serial.parseInt( ); ចាប់យកតម្លៃជាចំនួនគត់ហើយផ្ទេរទៅ X
      Serial.print( "Enter y=" );
   while( Serial.available( )==0 ){ }
      x=Serial.parseInt( );
      r=x+y; ការគណនាតម្លៃផលបូកពីរចំនួន;
     Serial.println( r ); បង្ហាញតម្លៃ r ពេលដែលបូករួចហើយ }
void loop( ){}
```

6.3. **ការប្រើប្រាស់** Arduino Serial Input

ការប្រើប្រាស់អនុគមន៍ Serial.read( ) សម្រាប់នាំទិន្នន័យពីកុំព្យូទ័រឬឧបករណ៍ផ្សេងទៅឱ្យប៉ត Arduino ដោយការទាក់ទងគ្នាប្រើសៀរៀលម៉ូនិទ័រ ដោយអាចបញ្ជូនទិន្នន័យម្តងបានមួយតួនៃលេខគោលពីរឬហៅថាម្តងបានមួយប៊ីត។

```
void setup( ) {
  Serial.begin(9600);
}
char read_byte = 0;
void loop( ) {
    if (Serial.available( )>0){//if data is send បើទិន្នន័យត្រូវបានផ្ញើ
    read_byte = Serial.read( ); //ចាប់យកតួរអក្សរដែលបានបញ្ជូលផ្ញើកក្នុងអថេរ read_byte
    // check if a number was received
    if ( (read_byte >= '0') && (read_byte <= '9') ) {
            Serial.print("Number received: ");//បង្ហាញឃ្លា Number received:
            Serial.println(read_byte);//បង្ហាញតម្លៃនៅអថេរ read_bute
    }
    else {
            Serial.println("Not a number."); ");//បង្ហាញឃ្លា Not a number.");
    }
  }
}
```



7. **អនុគមន៍** PinMode( ) **និង** digitalWrite( )

អនុគមន៍ទាំងពីរនេះមានសារៈសំខាន់ណាស់ក្នុងការធ្វើការតេស្តកូដ និងយល់ពីការប្រើ Arduino IDE និង LED មួយឬច្រើន និង ដឹងពីរបៀបបញ្ជូលកូដទៅប៉ត។ វាជាគ្រឹះដ៏សំខាន់ពេលយើងសរសេរកូដដំបូង និងជាគំរូមួយសម្រាប់ឱ្យយើងសរសេរកូដបន្តបន្ទាប់ទៀតដើម្បីតេសឧបករណ៍ជាច្រើនទៀត។

7.1. pinMode ( ) ជាអនុគមន៍នេះសម្រាប់បើកគន្លឹះជើង pin ជាចេញ និងចូល។ សូមបញ្ជាក់ផងដែរថាជើងពីងមួយអាចចេញលេខ១ឬ0 ក្នុងប្រើវាជាឌីជីថលពីន។ ដូច្នេះដើម្បីបើកគន្លឹះចេញឬចូលត្រូវប្រើអនុគមន៍នេះ។

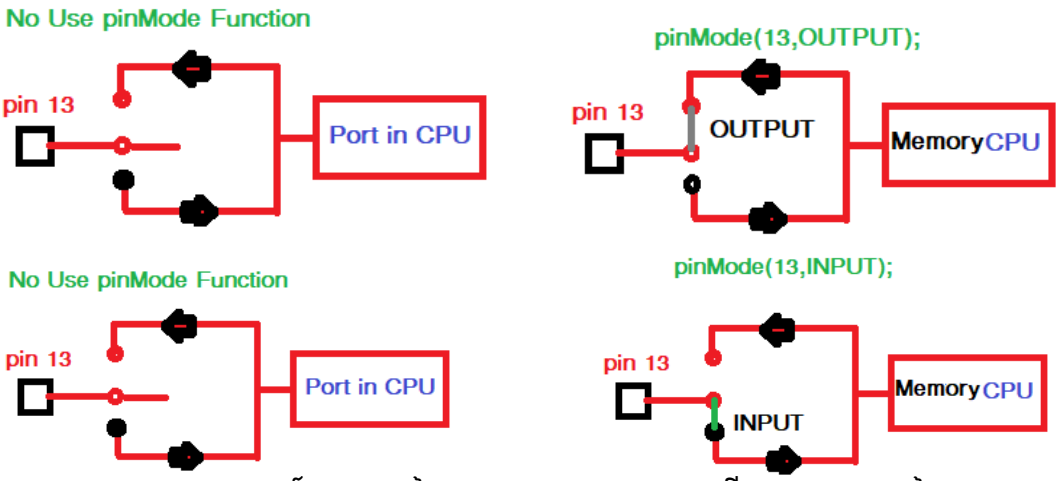
pinMode(pin, mode)

pin ប្រើ Digital Pin បើប៉ត Arduino Uno អាចប្រើលេខ 0 ដល់ ១៣។

mode : អាចប្រើ OUTPUT or INPUT

ឧទាហរណ៍: pindMode(13, OUTPUT); មាននន័យបើកគន្លឹះ Pin 13 ជាទិសដៅចេញ។ ដូចជាយើងចង់ចេញពីបន្ទប់ក្នុងផ្ទះនោះទី១ត្រូវបើកទ្វារសិន និងទី២ទើបដើរចេញបាន។ ដូច្នេះការ

សរសេរបញ្ជានេះគឺគ្រាន់តែបើកជើង Pin ប៉ុណ្ណោះ មិនមែនមានន័យថានាំទិន្នន័យចេញទេ។ តែក្នុងករណីនាំលេខ ១ ឬ ០ ចេញទៅឱ្យជីថលពីនត្រូវសរសេរ digitalWrite (13, 1)មានន័យនាំលេខ១ចេញទៅ Pin 13 ឬន័យក្នុងអគ្គិសនីជើងនេះមានភ្លើង ៥ វុលៗ សរុបមកដើម្បីនាំលេខ១ ឬ០ចេញយើងត្រូវប្រើបញ្ជាទាំងពីរដូចបានរៀបរាប់ខាងលើដាច់ខាតគឺទី១ បើកជើងពីនសិន ដោយសរសេរ pindMode(13, OUTPUT); នៅក្នុងអនុគមន៍ void setup( ); និងទី២ ទើបអាចនាំលេខ០ ឬ១បានចេញដោយសរសេរ digitalWrite (13, 1) នៅក្នុងអនុគមន៍ void loop( ) ដើម្បីឱ្យអំពូលបិទឬបើកតាមការកំណត់វិនាទី នៅក្នុងអនុគមន៍ delay( );។ ហើយវាជាមូលដ្ឋានសំខាន់ដំបូងសម្រាប់ការងារបន្តបន្ទាប់ទៀត។



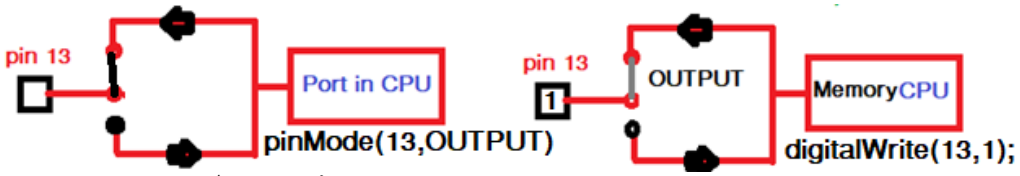
7.2. digitalWrite ( ) ជាអនុគមន៍បញ្ចេញតម្លៃ ១(ON) ឬ ០(OFF) ពីក្នុង Memory នៃ Arduino Board (ATME328) ទៅជើង Digital Pin។ បើ

digitalWrite(pin, value)

HIGH = ON = 1 = 3-5V

LOW = OFF= 0 = No Power

ឧទាហរណ៍: digitalWrite(13, 1); មានន័យថាបញ្ចេញលេខ ១ ទៅ Pin13។ ហើយក្នុងអគ្គិសនីធ្វើឱ្យ Pin 13 មានចរន្ត 5v។



digitalRead ( )ជាអនុគមន៍បញ្ជូលតម្លៃ ១(ON) ឬ ០(OFF) ពីក្នុង Digital Pin ទៅ Memory នៃ Arduino Board (ATME328) ទៅជើង Digital Pin។ បើ

DigitalRead(pin)

HIGH = ON = 1 = 3-5V ;LOW = OFF= 0 = No Power

ឧទាហរណ៍: buttonState = digitaRead(buttonPin); មានន័យថាបញ្ជូលលេខ ១ ឬ០ ពេលចុច ប៊ូតុង ទៅ Pin13។

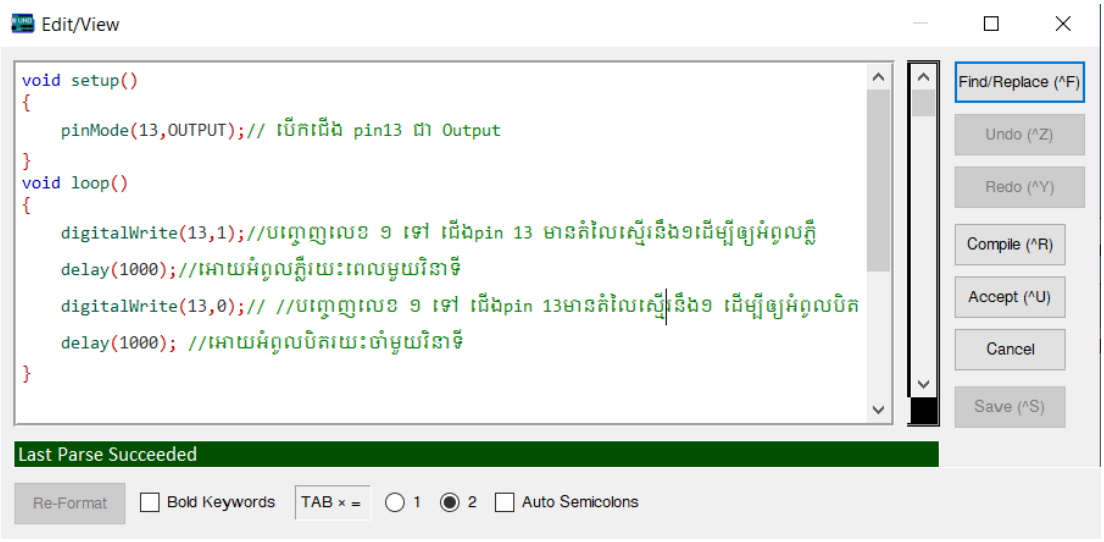
តែករណីបើកគន្លឹះចូលយើងត្រូវសរសេរ **pindMode(13, INPUT)**;មានន័យកំណត់ជើងគន្លឹះ Pin 13 ជាទិសដៅចូល។ ពីព្រោះ Pin 13 មានតែមួយវាអាចនាំលេខ ១ ឬ ០ ចេញ និង ចូលសូមមើលរូបខាងក្រោម។ តែក្នុងករណីនាំលេខ ១ ឬ ០ ចេញត្រូវសរសេរ

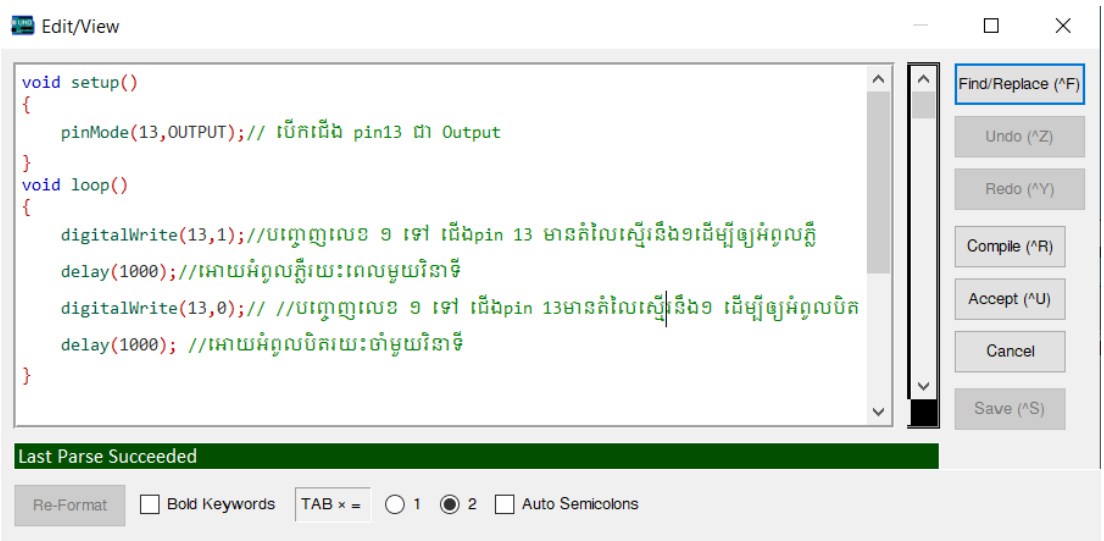
8. ការតេសកូដដោយប្រើអំពូល LED

8.1. ប្រើកម្មវិធី UnoArduSim

កូដដំបូងនេះធ្វើការបញ្ជាបិទឬបើកអំពូល Led មួយឬហៅថា Blinking Led ដោយឱ្យអំពូលភ្លឺមួយរយៈ ១ វិនាទី និង បិទមួយវិនាទី ត្រូវអនុវត្តតាមជំហាននីមួយៗ ដូចខាងក្រោម៖

- **ជំហានទី១:** ចុចMouseពីរដងនៅផ្ទៃកូដផេន និងសរសេរកូដតាមគំរូដូចខាងក្រោម





- **ជំហានទី២:** ចុចលើបញ្ជា Compiler នៅខាងស្តាំដើម្បីធ្វើការបំលែងកូដទៅម៉ាស៊ីនកូដ
- **ជំហានទី៣:** ចុចលើបញ្ជា Accept ដើម្បីបញ្ចូលកូដទៅផ្ទុកក្នុង Program File នៃ ATMEGA328P ដែលមាននៅលើ Arduino Board
- **ជំហានទី៤:** ចុចលើ (Run) ប្រតិបត្តិកូដដើម្បីបានលទ្ធផលអំពូលបិទឬបើករយៈពេលក្នុងពេលមួយវិនាទី

8.2. ប្រើកម្មវិធី Proteus

8.2.1. ការដំឡើងផ្នែករឹង

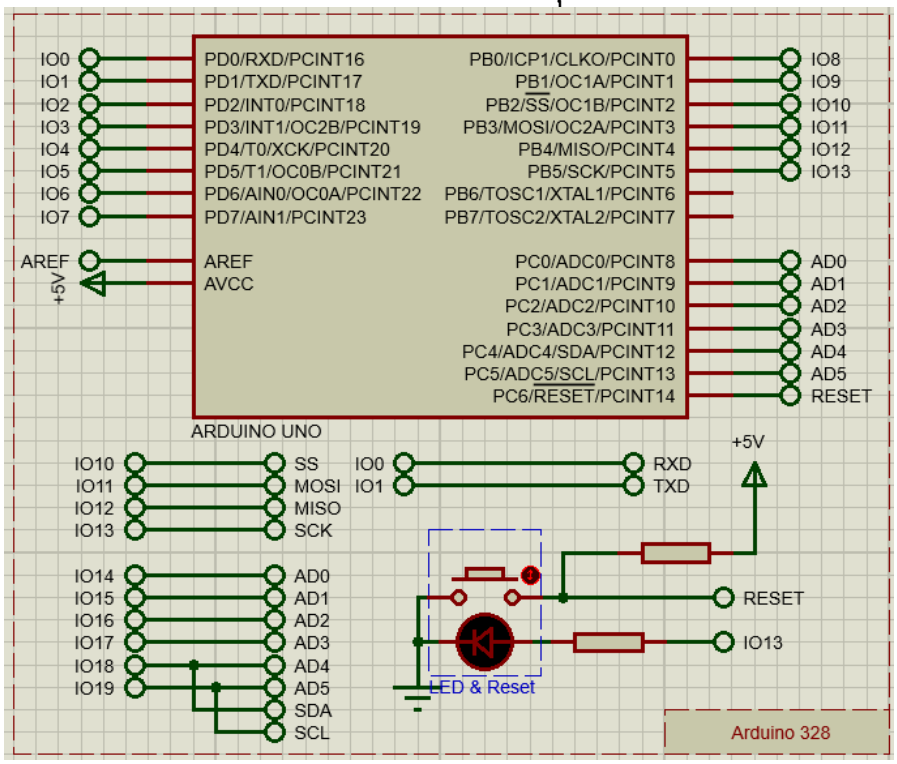
- **ជំហានទី ១:** ចូលក្នុងកម្មវិធី Proteus និងបន្ទាប់ជ្រើសរើសយក New Protes

- **ជំហានទី ២:** ដាក់ឈ្មោះ Project ក្នុងប្រអប់ name

- **ជំហានទី ៣:** ចុចលើ Next ចំនួនបីដង

- **ជំហានទី ៤:** ជ្រើសរើសយក Create Firmware Project និងត្រង់ Family ជ្រើសយក Arduino

- **ជំហានទី ៥:** ចុច Finish និងបានផ្ទៃដូចខាងក្រោមក្នុង Tab:Schematic Capture :



ជំហានទី ៥: ចំលងកូដខាងក្រោម

```
void setup( )
{
        pinMode( 13,OUTPUT );// បើកគន្លឹះ pin13 ជាទិសដៅចេញ
}
void loop( )
{
        digitalWrite( 13,1 );//បញ្ចេញលេខ ១ ទៅជើងpin 13 មានតម្លៃស្មើនឹង១
        ដើម្បីឲ្យអំពូលភ្លឺ
        delay( 1000 );//ឱ្យអំពូលភ្លឺរយៈពេលមួយវិនាទី
        digitalWrite( 13,0 );// //បញ្ចេញលេខ ១ ទៅ ជើងpin 13មានតម្លៃស្មើនឹង១
        ដើម្បីឲ្យអំពូលបិទ
        delay( 1000 ); //ឱ្យអំពូលបិទរយៈចាំមួយវិនាទី}
```

8.2.2. ការប្រើ Arduino Library

ក្នុងករណីយើងចង់ប្រើរូបដែលមាននៅក្នុងកម្មវិធី UnoArduSim នោះយើងត្រូវអនុវត្តតាមជំហានដូចខាងក្រោម៖

ជំហានទី១: ទាញឬចំលងយក Library ពីទីតាំងណាមួយសម្រាប់ប៉ត Arduino

ជំហានទី២: ចុចលើ Drive c:\ រួចជ្រើសរើស Program File(x86)

ជំហានទី៣: ចុចលើ Labcenter Electronis

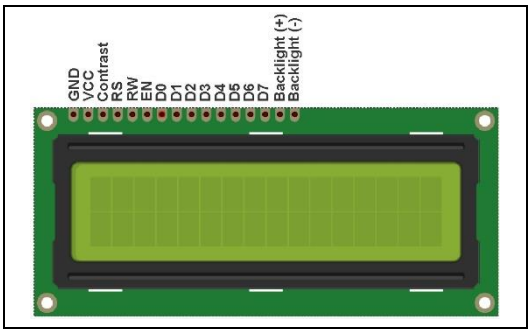
ជំហានទី៤: ចុចលើ Proteus 8 professional និងជ្រើសយក DATA និងបន្ទាប់មកចុច Mouse លើ Library រួចចំលង ARDUINO.LIB និង ARDUINO.IDX

សូមមើលការប្រើប៉តនេះកម្មវិធី Proteus នៅក្នុងឧទាហរណ៍ជាបន្តបន្ទាប់ខាងក្រោមៈ

9. ប្រើកម្មវិធី Arduino IDE

ក្នុងករណីយើងមានប៉ត Arduino ជាក់ស្តែងយើងសរសេរកូដដូចបានរៀបរាប់ខាងលើជាក់ក្នុង Arduino IDE និងជ្រើសរើស Serial Port អាចជា COM4, COM4, COM1,...និងបន្ទាប់មកឈ្មោះ Board ដែលបានប្រើជាក់ស្តែងដូចជា Arduino/Genuino Uno, Arduino Nano ឬ Arduino/Genuino Mega or Mega2560។

9.1. ឧបករណ៍បញ្ចេញបញ្ចូល



LCD (Liquid Crystal Display) គឺជាឧបករណ៍សម្រាប់បង្ហាញតម្លៃលេខ, សញ្ញា និង តួអក្សរ មកលើអេក្រង់ LCD នៅពេលអ្នកប្រើប្រាស់ប្រើវាធ្វើអ្វីមួយ។ វាមានជួរដេកចំនួន 2(0 -1), ជួរឈរ ចំនួន 16(0-15), មាន Pins ចំនួន 16។ នៅក្នុង LCD មាន Register ចំនួនពីរគឺ Command Register និង data register.

9.1.1. សមាសភាគនៅលើ LCD

- Command Register ប្រើដើម្បីបញ្ចូលពាក្យបញ្ជាពិសេសនៅក្នុង LCD។ ពាក្យបញ្ជាគឺជាសំណុំទិន្នន័យពិសេសហើយវាត្រូវបានប្រើដើម្បីផ្តល់ពាក្យបញ្ជាផ្ទៃក្នុងឱ្យ LCD ដូចជាឱ្យអេក្រង់ច្បាស់ផ្លាស់ទៅជួរទី១, តួអក្សរទី១, កំណត់ទស្សន៍ទ្រនិច។ល។

- Data Register ប្រើសម្រាប់ផ្ទុកទិន្នន័យដែលត្រូវបង្ហាញ

- VSS ជា Ground សម្រាប់ធ្វើការតភ្ជាប់ GND LCD ទៅប៉ត Arduino ឬឧបករណ៍ផ្សេងទៀត
- VDD: គឺជាជើង VCC ប្រើសម្រាប់ផ្តត់ផ្តង់ថាមពល 5V ឱ្យ LCD
- VEE: កំណត់លើភាពផ្ទុយគ្នានៃការបង្ហាញលើអេក្រង់ lcd ដោយធ្វើការផ្លាស់ប្តូរថាមពលពីប្រភព 0V ទៅ 5V។ វាត្រូវបានប្រើសម្រាប់ការកែតម្រូវកម្រិតពន្លឺនៅលើ LCD។
- RS ត្រូវបានគេស្គាល់ Register Select ហើយប្រើជ្រើសរើស Command /data register។ ដើម្បីជ្រើសរើស Command Register នោះ RS មានតម្លៃស្មើនឹងសូន្យ។ ចំណែកការជ្រើសរើសទិន្នន័យ Data Register នោះ RS មានតម្លៃស្មើនឹងមួយ។
- RW ប្រើដើម្បី read / write ទិន្នន័យឬបញ្ជាទៅឱ្យ LCD។ នៅពេលដែល low(0v) អ្នកត្រូវបានអនុញ្ញាតឱ្យ write ទិន្នន័យទៅ LCD។ ហើយពេលដែល high(2.7v ទៅ 5v)អ្នកអាច read ទិន្នន័យពី LCD។

- E(Enable) Pin នេះត្រូវបានប្រើដើម្បីបញ្ចូលទិន្នន័យទៅ Data Register។ បញ្ជាឧបករណ៍បញ្ជា LCD ឱ្យបង្ហាញតួអក្សរនៅលើ LCD។
- D0 – D7 គឺជា pin ដែលផ្ទុកទិន្នន័យ 8 bit ឬ 4 bit ដែលបញ្ជូនទៅដើម្បីធ្វើការបង្ហាញទិន្នន័យនៅលើអេក្រង់នៃ LCD។ នៅក្នុងរបៀប 8 bit D7 អាចត្រូវបានត្រួតពិនិត្យនៅក្នុង software ដើម្បីដឹងថាតើពាក្យបញ្ជា LCD ឬប្រតិបត្តិការទិន្នន័យបានចប់ដែរឬទេ។ នៅពេលខ្សែនេះគឺ HIGH វាបង្ហាញថាប្រតិបត្តិការ LCD កំពុងដំណើរការ។ នៅក្នុងរបៀប 4 bit pin D7 ទៅ D4 អាចត្រូវបានប្រើដើម្បីសរសេរពាក្យបញ្ជាឬទិន្នន័យណាមួយទៅ LCD។

9.1.2. Header File: LiquidCrystal.h

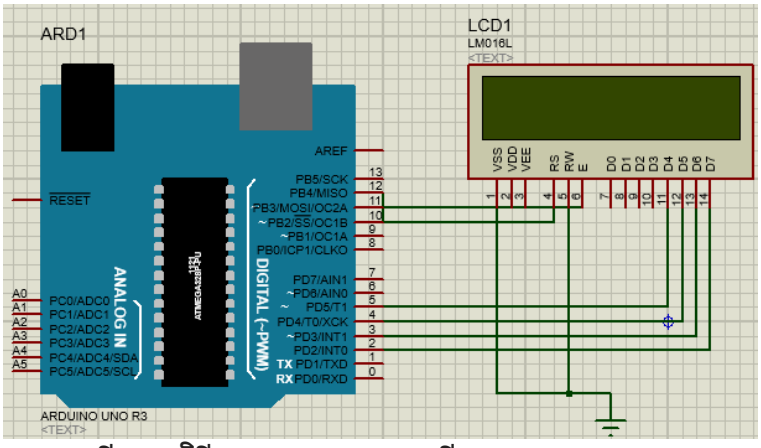
LiquidCrystal.h គឺជា Library មួយប្រើសម្រាប់កំណត់ជើង Pin Arduino ភ្ជាប់ទៅ LCD ។ ដូច្នេះដើម្បីបញ្ចូលវាត្រូវសរសេរ **#include<LiquidCrystal.h>** សម្រាប់បង្កើត Object ឈ្មោះ lcd ឬឈ្មោះផ្សេងក៏បានរួចយកវាធ្វើជាមួយអនុគមន៍ក្នុងភាសា Arduino ដើម្បីបានបានជាបញ្ជាសម្រាប់ធ្វើប្រមាណវិធីនៅលើ LCD។ ខាងក្រោមគឺជាអនុគមន៍មួយចំនួនដែលត្រូវប្រើវាក្នុងការសរសេរកូដ:

lcd.begin(cols,rows);	អនុគមន៍នេះកំណត់ទំហំlcdដែលបានប្រើដែលមានជួរឈរនិងជួរដេក
lcd.print( );	អនុគមន៍នេះប្រើសម្រាប់ធ្វើការបង្ហាញតួអក្សរនៅលើ LCD
lcd.cursor( );	សម្រាប់កំណត់ឱ្យ Cursor នៅទីតាំងផ្សេងៗនៅអេក្រង់
lcd.write( );	យើងអាចប្រើអនុគមន៍នេះសម្រាប់ write ប្រភេទនៃទិន្នន័យ នៅលើអេក្រង់ដូចពីសីតុណ្ហភាព, GPS,…
lcd.display( );	ធ្វើការបើក lcd បន្ទាប់ពីបិទដោយប្រើអនុគមន៍ noDislay( )
lcd.home( );	កំណត់ទីតាំង Cursor នៅ(0,0) ខាង Upper-left corner
lcd.blink( );	ធ្វើការបង្ហាញ blink lcd cursor
lcd.autoscroll( );	ធ្វើការ scroll Screen ដោយស្វ័យប្រវត្តិ

ចំណាំ: ការស្គាល់ Pin នៃ LCD និងអនុគមន៍មួយចំនួនសម្រាប់ធ្វើប្រមាណវិធីលើ LCD និងជាមូលដ្ឋានគ្រឹះដ៏សំខាន់សម្រាប់រៀនឧបករណ៍ផ្សេងៗមួយចំនួនទៀត។

9.1.3. កម្មវិធីលើ LCD

- Hardware: ចូររចនាផ្នែករឹងឱ្យបានដូចរូបខាងក្រោម



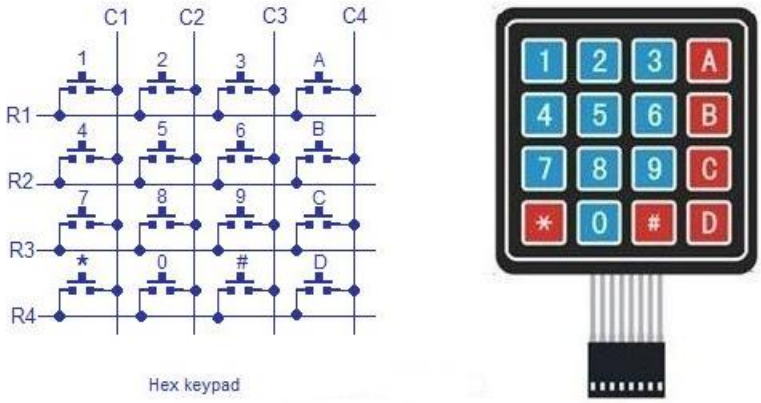
- coding: បើកកម្មវិធី Arduino រួចចុចលើ File→Example→ liquidCrystal បន្ទាប់យក HellowWorld

```
/ include the library code:
#include <LiquidCrystal.h>
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd( 12, 11, 5, 4, 3, 2 );//RS,
void setup( ) {
   lcd.begin( 16, 2 ); // set up the LCD's number of columns and rows:
   lcd.print( "hello, world!" );}  // Print a message to the LCD.
void loop( ) {
  // set the cursor to column 0, line 1
  // ( note: line 1 is the second row, since counting begins with 0 ):
  lcd.setCursor( 0, 1 );
  // print the number of seconds since reset:
  lcd.print( millis( ) / 1000 );}
```

9.2. Keypad

Keypad 4x4 គឺជាបន្ទះសៀគ្វីមួយដែលផ្គុំឡើងពីប៊ូតុងពីលេខ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 និងអក្សរ A, B, C, D, *, #. ។ ប្រើវាសម្រាប់ធ្វើការបញ្ចូលតម្លៃលេខ, អក្សរ, សញ្ញាពី Keypad ទៅកាន់ប៊ត Arduino វាជាឧបករណ៍បញ្ចូលទិន្នន័យសម្រាប់ប្រើប្រាស់នៅលើឧបករណ៍ដែលតម្រូវឱ្យមានធាតុបញ្ចូលសំខាន់ៗដូចជា៖ ម៉ាស៊ីនគិតលេខ, បញ្ជាទូរទស្សន៍, ម៉ាស៊ីន ATM, ចាក់សោទ្វារឌីជីថលជាដើម។

- មានប៊ូតុងចុចចំនួន 16 (0,1,2,3,4,5,6,7,8,9,A,B,C,D, *,#)
- មានជួរដេក(Rows)ចំនួន 4 (R1-R4)
- មានជួរឈរ(Colums)ចំនួន 4 (C1-C4) ។



នៅក្នុងកម្មវិធីខាងក្រោមយើងប្រើ Library ឈ្មោះ Keypad.h ដើម្បីបង្កើត Object ដែលមានឈ្មោះ myKeypad ឬអាចដាក់ឈ្មោះផ្សេងៗក៏បាន សម្រាប់ប្រើជាអនុគមន៍ដំណើរការនៅលើ Keypad។ វាឧបករណ៍បញ្ចូលមួយប្រើប្រាស់សម្រាប់ធ្វើគម្រោងផ្សេងៗដែលមាននៅក្នុងសៀវភៅនេះ ព្រោះថារាល់ការងារជាប់ទាក់ទងនឹងកុំព្យូទ័រស្ថាបត្យកម្មពេលរចនាត្រូវមាន:

១. ឧបករណ៍បញ្ចូលទិន្នន័យ(Input Device) មានដូចជា Keypad, Senosr

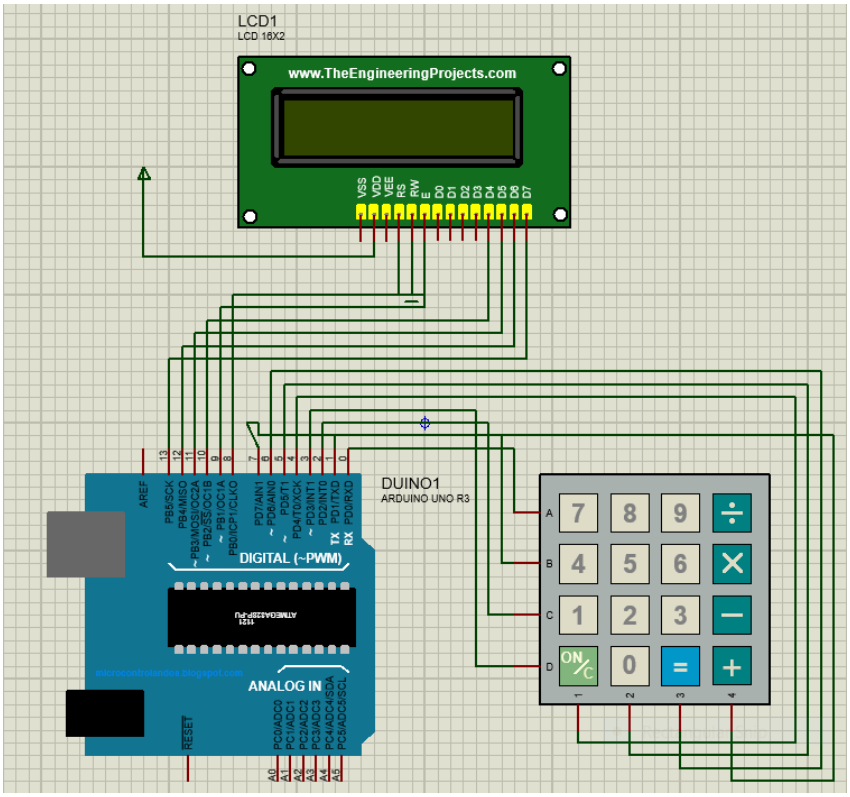
២. ឧបករណ៍ដំណើរការទិន្នន័យមានដូចប៉ត Arduino, Raspberry pi, ESP8266

៣. ឧបករណ៍ដំណើរការបញ្ចេញទិន្នន័យមានដូចជា LCD, Motor Driver

៤. ឧបករណ៍សម្រាប់រក្សាទុកនូវព័ត៌មាន

char key = mykeypad.getkey( ); //អរេរ Key សម្រាប់ទទួលតម្លៃ string ពីបញ្ចា

- Hardware: ចូររចនាផ្នែករឹងឱ្យបានដូចរូបខាងក្រោម



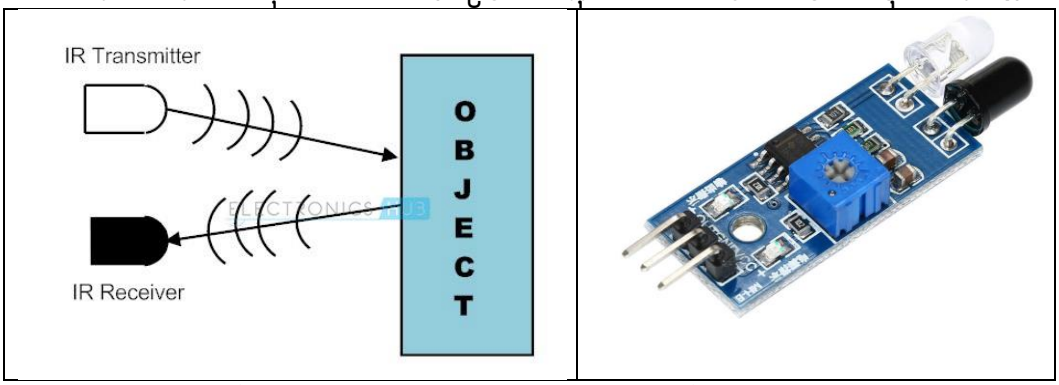
- coding: បើកកម្មវិធី Arduino រួចចុចលើ File→Example→ liquidCrystal បន្ទាប់

```
#Include<LiquidCrystal.h>  //បញ្ចូល Library របស់ LCD
#Include<Keypad.h>  //បញ្ចូលLibraryរបស់ Keypad
Liquid Crystal lcd(3,9,10,11,12,13)  //បង្កើតObject ឬ កំណត់ជើងឲ្យLCD
Const byte ROWS =  4  //តំណាងឲ្យ ROW4 របស់ Keypad
Const byte COLS = 4  //តំណាងឲ្យCOLS4 របស់ Keypad
Char keys [ROWS][COLS]= {
        {'7','8','9','/'},
        {'4','5','6','*'}
        {'1','2','3,'-'},
        {'X','0','=','+'},
        };  //កំណត់ Keys ជា String ដែលសម្រាប់ការចុចនៅលើ Keypad
byte rowpin [ROWS] = {0,1,2,3};  //កំណត់Arrayផ្ទុកសំណុំធាតុ 0,1,2,3 ដែល
ជាPin //របស់Arduinoសម្រាប់តភ្ជាប់ពីROWSរបស់ Keypad ទៅ Arduino
byte colpin [COLS] = {4,5,6,7};  //កំណត់Arrayផ្ទុកសំណុំធាតុ 4,5,6,7 ដែល
ជាPIN  //របស់Arduinoដើម្បីតភ្ជាប់ Columns របស់ Keypad ទៅ Arduino
Keypad mykeypad = Keypad(makeKeypad(Keys),rowPins, Colpin, Rows,
Cols); //បង្កើត MyKeypad  Object សម្រាប់ផ្សំជាមួយនិងអនុគមន៍ក្នុង
Keypad ដើម្បី //ធ្វើការចុចitem នៅក្នុងវង់ក្រចក
void  setup( ){
        lcd.begin(16,2); //កំណត់ទំហំ LCD ដែលមានជួរឈរ១៦និងជួរដេក២
        lcd.setCursor(2,0);  //កំណត់ទីតាំងCursor នៅជួរឈរទី២,ជួរដេក 0
         lcd.print("Hello   User");  //បង្ហាញឃ្លា Hello User នៅលើ LCD
         delay(2000);  //ពន្យាពេលរយៈ 2 វិនាទី
         lcd.clear( );  //ធ្វើការសំអាត screen LCD
}
void loop{
char key = mykeypad.getkey( ); //អានថៃ Key ផ្ទុកតម្លៃstring ពីពេលចុចKey
        If(key){
                lcd.print(key);}}
```

9.3. IR Infrared Sensor

IR ជាឧបករណ៍អ៊ីនហ្វ្រារ៉េដសេនស័រគឺជាឧបករណ៍អេឡិចត្រូនិចដែលបញ្ចេញដើម្បីដឹងអំពីទិដ្ឋភាពខ្លះនៃមជ្ឈដ្ឋានជុំវិញ។ ឧបករណ៍នេះអាចវាស់កំដៅនៃវត្ថុក៏ដូចជាវារកចលនា។

ឧបករណ៍ចាប់សញ្ញាប្រភេទទាំងនេះវាស់តែកាំរស្មីអ៊ីនហ្វ្រារ៉េដជាជាងបញ្ចេញវា ដែលហៅថា ឧបករណ៍IRអកម្ម ជាធម្មតានៅក្នុងវិសាលគមអ៊ីនហ្វ្រារ៉េដវត្តទាំងអស់បញ្ចេញទម្រង់វិទ្យុសកម្មកម្តៅ។



- ឧបករណ៍បញ្ជូនពន្លឺអ៊ីនហ្វ្រារ៉េដគឺជាឧបករណ៍បញ្ចេញពន្លឺ (LED) ដែលបញ្ចេញកាំរស្មីអ៊ីនហ្វ្រារ៉េដ។ ដូច្នេះពួកគេត្រូវបានគេហៅថា IR LED។ ទោះបីជាអំពូលអេឡិចត្រូនិច IR មើលទៅដូចជាអំពូល LED ធម្មតាក៏ដោយកាំរស្មីដែលបញ្ចេញដោយវាមិនអាចមើលឃើញដោយភ្នែកមនុស្សទេ។

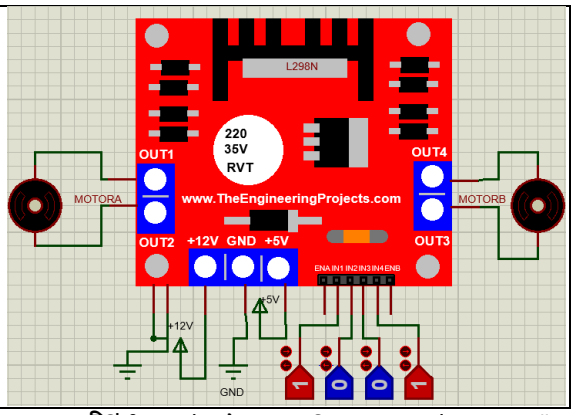
- អ្នកទទួលអ៊ីនហ្វ្រារ៉េដត្រូវបានគេហៅថាឧបករណ៍ចាប់សញ្ញាអ៊ីនហ្វ្រារ៉េដផងដែរ នៅពេលដែលពួកគេរកឃើញវិទ្យុសកម្មពីឧបករណ៍បញ្ជូន IR។ អ្នកទទួល IR មកក្នុងទម្រង់ជាម៉ាស៊ីនកូដ។ ម៉ាស៊ីនថតអ៊ីនហ្វ្រារ៉េដមានភាពខុសប្លែកពីឌីជីថលរូបថតធម្មតានៅពេលថតរូបបានរូបភាពជាក់ក្នុងអង្គចងចាំ។ IR Recdiver នៅពេលគេបានជួបតែកាំរស្មីអ៊ីនហ្វ្រារ៉េបង្កើតបានជាតុងស្យុងគិតជាវុល ហើយបំលែងទៅតម្លៃ Analog។

- **ការដំឡើងផ្នែករឹងតាមរូបខាងក្រោម ដោយ pin6 ទៅ IN1, pin5ទៅ IN2, និង pin4 ទៅ IN3, pin3ទៅ IN4**
- **កម្មវិធី**

```
int irPin = 3;  //ប្រកាសអថេរ irPin ភ្ជាប់ទៅ Pin 13
int ledPin =12;//ប្រកាសអថេរ ledPin ភ្ជាប់ទៅ Pin 12
void setup( ) {
  Serial.begin(9600); //បើក Serial port និងកំណត់ល្បឿន ៩៦០០ ក្នុងមួយវិនាទី
  pinMode(irPin, OUTPUT); //កំណត់ irPin ជាទិសដៅចេញ
  pinMode(ledPin, OUTPUT);//កំណត់ ledPin ជាទិសដៅចេញ
 }
 void loop( ) {
  int z=digitalRead(irPin); //ធ្វើការចាប់តម្លៃហើយផ្ទុកនៅក្នុង z
  if(z==1){
   digitalWrite(ledPin,1);} // បញ្ចេញតម្លៃទៅ ledPin ដើម្បីបើកអំពូល
  if(z==0){
   digitalWrite(ledPin,0); }// បញ្ចេញតម្លៃទៅ ledPin ដើម្បីបិតអំពូល
```

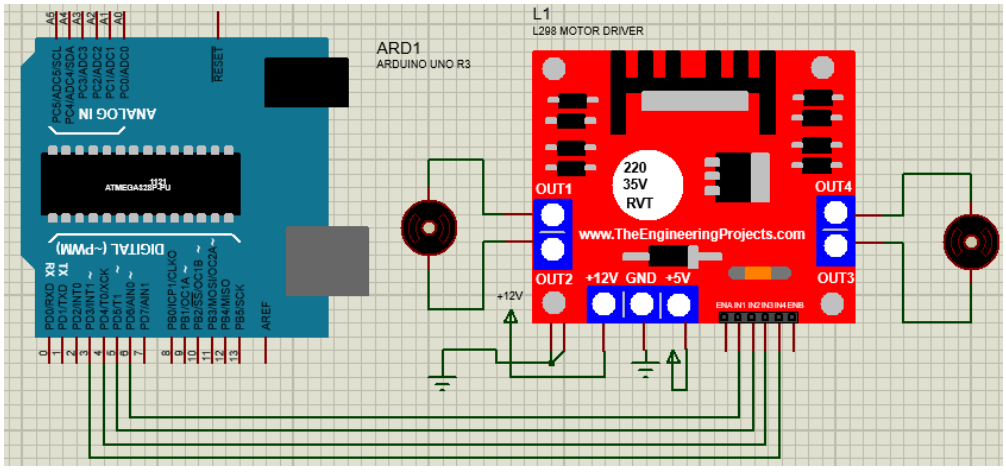
9.4. Motor Driver

វាគឺជា driver ដែលអាចឱ្យយើងគ្រប់គ្រងល្បឿន, ទិសដៅសម្រាប់ Motor ទាំងពីរ និងវាប្រកពផ្តល់ភ្លើង ៥ វុលទៅឱ្យ Arduino Board។ នៅលើ Motor driver នេះគឺមាន Pin OUT1 និង OUT2 សម្រាប់ភ្ជាប់ទៅ MotorA និង Pin OUT3 និង OUT4 សម្រាប់ភ្ជាប់ទៅ MotorB។ Power Pin មានពីរគឺ ១២វុល និង GND ប្រើសម្រាប់បញ្ចូលភ្លើងចាប់ពី ៦ វុល ទៅ ១២ វុលពី battery និងមានភ្លើងចេញ ៥ វុលមួយ។ ចំណែកឯ Pin ភ្ជាប់ទៅ Arduino Board មាន ENA, IN1, IN2, IN3, IN4 និង ENA។ ដោយ ENA Pin ឱ្យ MotorA ដំណើរការ និង IN1, IN2 ទទួលតម្លៃ 0 ឬ 1 ដើម្បីឱ្យវិលទៅមុខឬថយក្រោយ។ ចំណែក ENB Pin ឱ្យ MotorB(តាមរូបនៅខាងស្តាំ) ដំណើរការ និង IN3, IN24 ទទួលតម្លៃ 0 ឬ 1 ដើម្បី ឱ្យវិលទៅមុខឬថយក្រោយ។ ចូរមើលរូបខាងក្រោមដើម្បីយល់ពីដំណើរការនៃជើង Pin ទាំងនេះ និងធ្វើការតេសវានៅកម្មវិធី Proteus:

- ជើង 12V ភ្ជាប់ទៅប្រភពភ្លើង ១២ វុល - ជើង GND ភ្ជាប់ទៅប្រភពភ្លើង GND - 5V ជាភ្លើងចេញសម្រាប់ភ្ជាប់ទីBoard - ពេល IN1=1 និង IN2=0 ម៉ូទ័រវិលតាមទ្រនិចនាឡិកា -ពេល IN3=0 និង IN4=0 ម៉ូទ័រវិលតាមទិសដៅបញ្ច្រាសវិញ	

ឧទាហរណ៍: ខាងក្រោមគឺជាការសរសេរកម្មវិធីកំណត់តម្លៃ 0 និង 1 ដាក់នៅក្នុងប៉ត Arduino និងពេលដំណើរការវាផ្តល់តម្លៃទាំងនេះឱ្យម៉ូទ័រ Driver។ ខុសពីរូបក្នុងតារាងតម្លៃ 0 និង 1 គឺផ្តល់ដោយប្រើតម្លៃតក្កដែលមានឈ្មោះ LOGICSTSATE។

- **ការដំឡើងផ្នែករឹងតាមរូបខាងក្រោម ដោយ pin6 ទៅ IN1, pin5ទៅ IN2, និង pin4 ទៅ IN3, pin3ទៅ IN4**



- **កម្មវិធី**

```
int in1 =6; //ប្រកាសអថេរ Pin6 ភ្ជាប់ទៅ in1
int in2 =5; //ប្រកាសអថេរ Pin5 ភ្ជាប់ទៅ in2
int in3 =4; //ប្រកាសអថេរ Pin4 ភ្ជាប់ទៅ in3
```

```
int in4 =3; //ប្រកាសអថេរ Pin3 ភ្ជាប់ទៅ in4
void setup( ){
  pinMode(in1, OUTPUT); //បើកគន្លឹះ Pin1 ជាទិសដៅចេញ
  pinMode(in2, OUTPUT); //បើកគន្លឹះ Pin2 ជាទិសដៅចេញ
  pinMode(in3, OUTPUT); //បើកគន្លឹះ Pin3 ជាទិសដៅចេញ
  pinMode(in4, OUTPUT); //បើកគន្លឹះ Pin2 ជាទិសដៅចេញ}
void loop( ){
  // motor A ដែលមាន in1, in2 ត្រូវបានវិលទៅមុខ
   digitalWrite(in1,1);
   digitalWrite(in2,0);
   // motor B ដែលមាន in3, in4 ត្រូវបានវិលទៅមុខ
   digitalWrite(in3,1);
   digitalWrite(in4,0); }
}
```

9.5. Buzzer

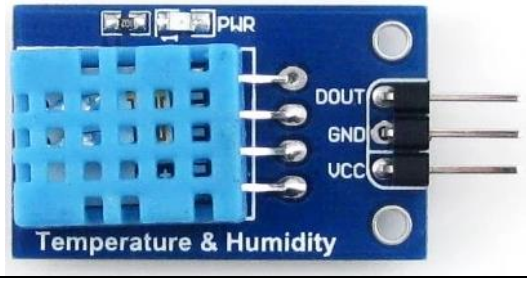
នៅក្នុងការចំនុចនេះយើងនឹងរៀនពីរបៀបប្រើឧបករណ៍បំពងសំឡេង buzzer ឬ piezo speaker ជាមួយនឹង Arduino។ ឧបករណ៍ Buzzers អាចរកបាននៅក្នុងឧបករណ៍រោទ៍កុំព្យូទ័រ, កម្មវិធីកំណត់ពេលវេលា និងការបញ្ជាក់ពីការបញ្ចូលរបស់អ្នកប្រើប្រាស់ដូចជាការចុច Mouse ឬ keystroke។ ការប្រើប្រាស់ឧបករណ៍នេះយើងប្រើអនុគមន៍ពីរគឺ tone( )និង noTone( )។

- tone( ): ធ្វើការផ្ញើ 1KHz=1000 សំលេង signal ទៅ pin 9 នៃ Arduino
- noTone( ): ធ្វើការបិតសំលេង
- ក្នុងករណីយើងគ្មានប៉ត Arduino ដោយអាចប្រើកម្មវិធី UnoArduSim ឬកម្មវិធី Proteus ជំនួសវិញ។ ឧទាហរណ៍ខាងក្រោមធ្វើការតេស Buzzer ជាមួយនឹង Arduino ដោយគ្រាន់តែភ្ជាប់pin 9 arduino ទៅជើង + នៃ buzzer និងជើងដក Buzzer ភ្ជាប់ទៅ GND។

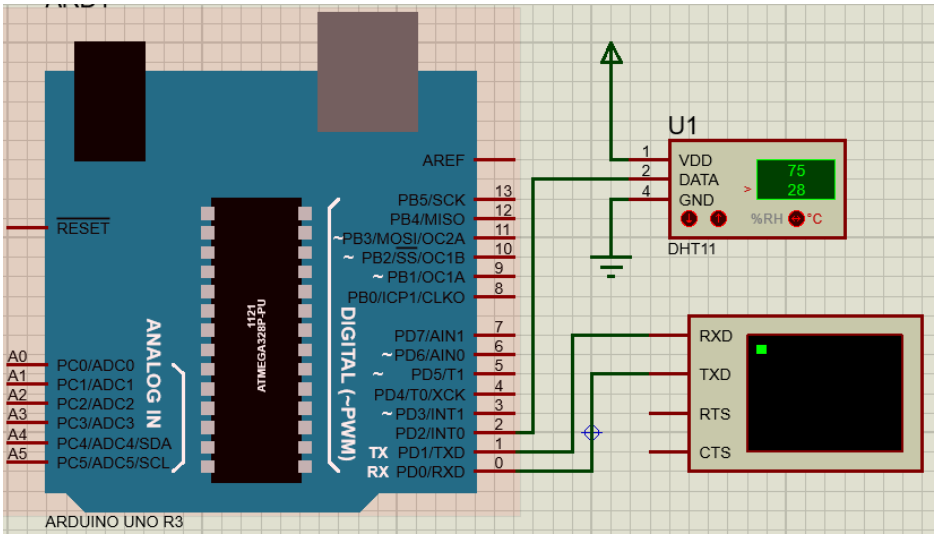
```
const int buzzer = 9; //buzzer to arduino pin 9
void setup( ){
    pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output
}
void loop( ){
    tone(buzzer, 1000); // Send 1KHz sound signal...
    delay(1000);        // ...for 1 sec
    noTone(buzzer);     // Stop sound...
    delay(1000);        // ...for 1sec
}
```

9.6. **សេនស័រសីតុណ្ហភាព**

DHT11គឺជាឧបករណ៍ចាប់សញ្ញាសីតុណ្ហភាព និងសំណើមដែលត្រូវបានប្រើជាទូទៅ។ឧបករណ៍នេះភ្ជាប់មកជាមួយអ NTC ធ្វើការវាស់សីតុណ្ហភាព និងឧបករណ៍ត្រួតពិនិត្យខ្នាតតូច ៨ ប៊ីតដើម្បីផ្តល់តម្លៃសីតុណ្ហភាពនិងសំណើមជាទិន្នន័យសៀរៀល។ ឧបករណ៍នេះក៏ត្រូវបានក្រិតតាមខ្នាតពីរោងចក្រផងដែរ ដែលជាហេតុដូច្នេះងាយស្រួលក្នុងការធ្វើអន្តរកម្មជាមួយឧបករណ៍ត្រួតពិនិត្យ Microcontrollers ផ្សេងទៀត។ ឧបករណ៍នេះអាចវាស់សីតុណ្ហភាពពី ០ អង្សាសេទៅ ៥០ អង្សាសេនិងសំណើមពី ២០% ទៅ ៩០% ជាមួយនឹងភាពត្រឹមត្រូវនៃ ១ អង្សាសេនិង± ១%។ ដូច្នេះប្រសិនបើយើងកំពុងចង់វាស់វែងនៅក្នុងជួរនេះនោះឧបករណ៍ចាប់សញ្ញានេះអាចជាជម្រើសត្រឹមត្រូវសម្រាប់ការងាររបស់យើង។

-VCC ជាជើងប្រើ ៥ វ៉ុល សម្រាប់ភ្ជាប់ទៅ ៥ វ៉ុល Arduino -GND ជាជើងប្រើ GND សម្រាប់ភ្ជាប់ទៅ GND Arduino -DOUT ជាជើងសម្រាប់បញ្ចូលទិន្នន័យ	

-ផ្នែករឹងធ្វើការតភ្ជាប់ផ្នែករឹងតាមរូបខាងក្រោម



-កម្មវិធី

```
// Example testing program DHT humidity/temperature sensors
#include "DHT.h"
#define DHTPIN 2   // Digital pin connected to the DHT sensor
// Uncomment whatever type you're using!
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);
void setup( ) {
  Serial.begin(9600);
  Serial.println(F("DHTxx test!"));
```

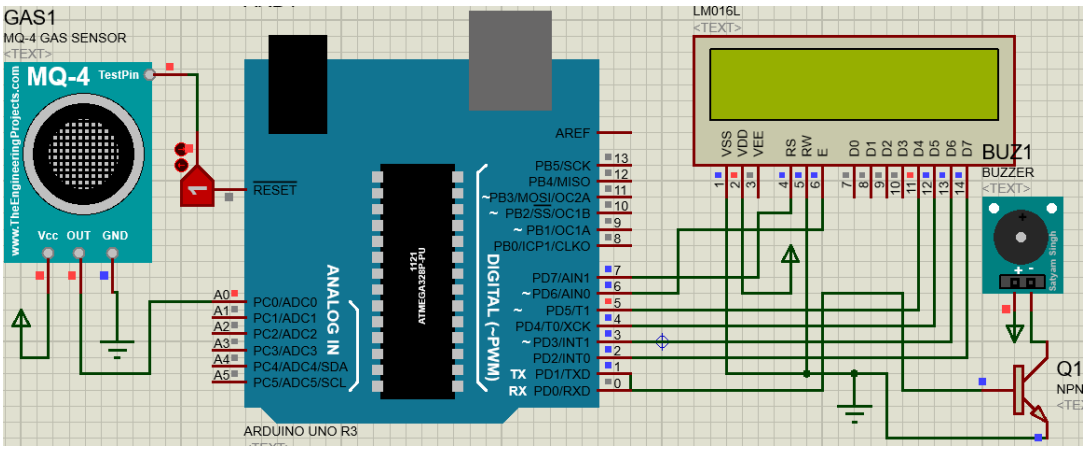
```
  dht.begin( );
}
void loop( ) {
  // Wait a few seconds between measurements.
  delay(2000);
  // Reading temper or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old'
  float h = dht.readHumidity( );
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature( );
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));
}
```

9.7. Gas Sensor

ឧបករណ៍ចាប់សញ្ញាឧស្ម័នត្រូវបានគេរកឃើញសមាសធាតុងាយនឹងបង្កជាហេតុ; នៅពេលទាក់ទងនឹងរាងកាយមនុស្សទាំងនេះអាចជាក្លិនទឹកនោម, ផ្សែង, ញើសក្លៀកឬដកដង្ហើមចេញចូល។

វាជំណើរការដូចជាច្រមុះអេឡិចត្រូនិចដែលអាចពាក់បានដែលត្រូវបានគេហៅថា "អេឡិចត្រូនិច" ។ វាមានជើងបីគឺ VCC, OUT និង GND។

- ផ្នែករឹងធ្វើការតភ្ជាប់ផ្នែករឹងតាមរូបខាងក្រោម



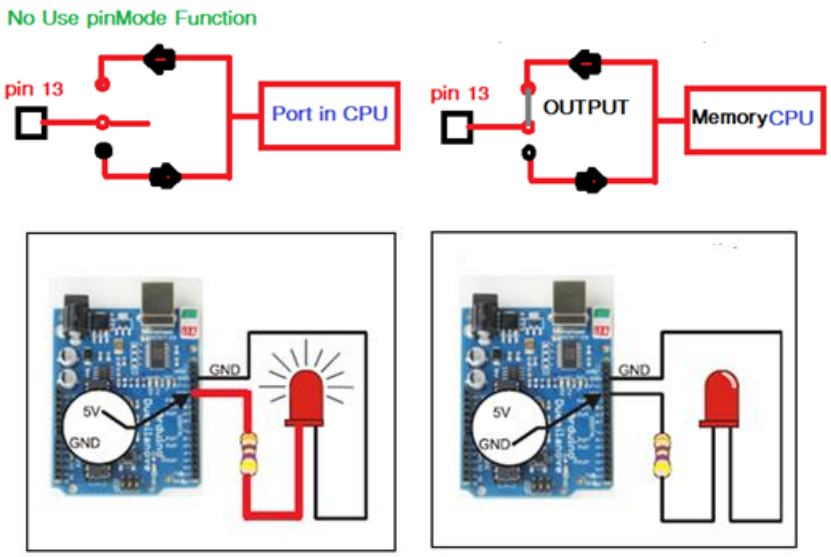
- កម្មវិធីវិធីគំរូសម្រាប់ធ្វើការចាប់សញ្ញា និងបង្ហាញនៅលើ lcd

```
#include <LiquidCrystal.h>
LiquidCrystal lcd( 7, 6, 5, 4, 3, 2 );
#define MQPin A0
#define buzzer 1

void setup( ) {
  lcd.begin( 16, 2 );
  pinMode( MQPin, INPUT_PULLUP );
  pinMode( buzzer, OUTPUT );
   lcd.setCursor( 5, 0 );
   lcd.print( "GAS" );
   lcd.setCursor( 3, 1 );
   lcd.print( "DETECTOR" );
   delay( 1000 );
   lcd.clear( );
}

void loop( ) {
int gas_value = digitalRead( MQPin );
if( gas_value==HIGH )
{
 digitalWrite( buzzer, HIGH );
```

```
  lcd.setCursor(6, 0);
  lcd.print("GAS");
   lcd.setCursor(3, 1);
  lcd.print("DETECTED");
  delay(200);
  lcd.clear( );
  delay(200) };
else{
 lcd.clear( );
 digitalWrite(buzzer, LOW);
}}
```

សំណួរ និង លំហាត់

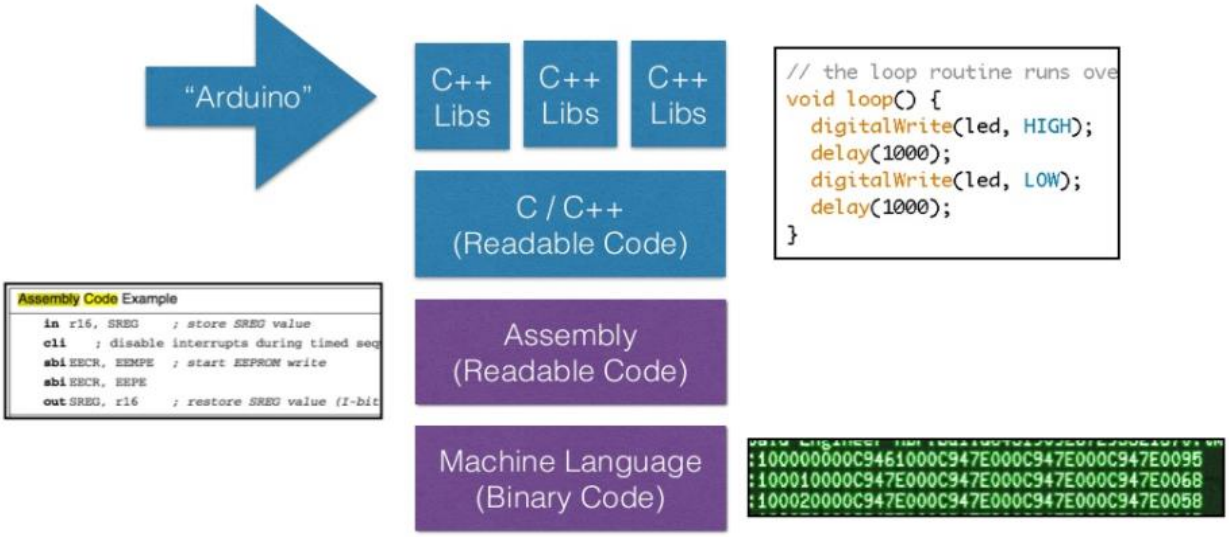
1. ដូចម្តេចដែលហៅ Microcontroller? វាមានប៉ុន្មានប្រភេទ? ឈ្មោះអ្វីខ្លះ?
2. ចូរប្រៀបធៀបលក្ខណៈខុសគ្នារវាង Microcontroller និង Microprocessor?
3. តើ Microcontroller អាចបង្កើតបានជាកុំព្យូទ័រដៃឬទេ? ចូរបញ្ជាក់ពីមូលហេតុ?
4. ដូចម្តេចដែលហៅថា Arduino IDE? តើវាមានប៉ុន្មានប្រភេទ? ឈ្មោះអ្វីខ្លះ?
5. គេប្រើ UnoArduSim IDE សម្រាប់ធ្វើអ្វី? និងដើម្បីសរសេរកូដត្រូវធ្វើដូចម្តេច?
6. ចូរនិយាយពីមុខងារនៃ Arduino Uno ប្រើក្នុង UnoArduSim? និងពន្យល់ពីតួនាទីនៃ (a). digital pin, (b).Analog pin; (c).Reset key; (d).Led Pin13; (e).(f). Usb port ដែលមាននៅលើប៉ត Arduino Uno ឱ្យបានក្បោះក្បាយ?
7. ចូរប្រៀបធៀបលក្ខណៈខុសគ្នារវាង Arduino Mega2560 និង arduino Uno?
8. ចូរនិយាយពីឈ្មោះ និងមុខងារផ្នែករឹងដែលមានក្នុង CPU ATMEGA328?
9. ចូរនិយាយពីឈ្មោះ និងរៀបរាប់ពីមុខងារនៃឧបករណ៍ I/O(lcd, i2c, led, push, switch, TFT lcd, serial monitor, dc motor, servo motor, 7 segment, dot matrix, Buzzer, ...)ដែលប្រើជាមួយនឹង UnoArduSim simulator?និងសរសេរកូដអោយឧបករណ៍និមួយៗដំណើរការ។
10. ចូរនិយាយពីមុខងារនៃអនុគមន៍ (a).setup( ), (b).loop( ), (c).pinMode( ), និង (d).digitalWrite( )?
11. តើត្រូវធ្វើដូចម្តេចដើម្បី Upload Code ពីកុំព្យូទ័រ ទៅ Arduino Board?
12. ចូរសរសេរ Statement តាមរូបខាងក្រោម៖



13. ចូរសរសេរកូដដើម្បី៖
    a. ដើម្បីចុចតម្លៃពី Keyboard និងធ្វើការបង្ហាញតម្លៃនិមួយៗនៅលើ LCD
    b. ធ្វើការបញ្ជា Motor ពីរទៅមុខពេលចុចលេខ ១ ក្នុង Serial Monitor
    c. ធ្វើការបិតអំពូល ឬបើកពេលចុចលេខ ១ ឬ0 ក្នុង Serial Monitor

មេរៀនទី ៨ ភាសា Arduino

ការសរសេរកម្មវិធី Arduino ឬហៅម៉្យាងទៀតថា Arduino Sketch នោះត្រូវប្រើរចនាសម្ព័ន្ធភាសា C/C++ និងការប្រើប្រាស់អនុគមន៍, តម្លៃ(variable និង constants) និង ស្ថាបត្យកម្មនៃ Arduino Board ដែលជាមូលដ្ឋានគ្រឹះយ៉ាងសំខាន់ដើម្បីបង្កើតកម្មវិធីទាក់ទងនឹងការដោះស្រាយចំណោទ និងធ្វើការបញ្ជា និងកុងត្រូលទៅលើផ្នែករឹងរបស់ Arduino Board និងឧបករណ៍ដទៃទៀតដែលបានតភ្ជាប់ជាមួយនឹងវាឱ្យដំណើរការទៅបាន។ ការចេះសរសេរកម្មវិធីនៅក្នុងភាសានេះ ឆាប់ចេះ យល់ និងអាចសរសេរបាននោះត្រូវចេះសរសេរកម្មវិធីក្នុងភាសា C។ យើងអាចសរសេរកូដដោយប្រើកភាសា Asm ដួលដល់កុងត្រូល និង ដំណើរការផ្នែករឹងបានលឿនបំផុត ជាជាងប្រើភាសា C។



1. Compiler

Compiler គឺជាកម្មវិធីមួយធ្វើការបកប្រែកម្មវិធីទាំងមូលទៅជាភាសាដែលម៉ាស៊ីនអាចយល់បាន។ វាកម្មវិធីចងក្រងកម្មវិធីកុំព្យូទ័រដែលបកប្រែ(ចងក្រង)កូដប្រភពដែលសរសេរជាភាសាកម្រិតភាសា Arduino និងភាសា Assembly ទៅជាសំណុំនៃពាក្យបញ្ជាជាភាសាម៉ាស៊ីនដែលអាចយល់បាន ដោយស៊ីភីយូរបស់កុំព្យូទ័រឌីជីថល។ វាគឺជាកម្មវិធីធំណាស់ដែលមានការពិនិត្យកំហុសនិងសមត្ថភាពផ្សេងទៀត។

2. **សញ្ញាប្រមាណវិធី**(Operator)

វាប្រើសម្រាប់ធ្វើប្រមាណវិធីគណិតវិទ្យាដើម្បីរកលទ្ធផលចេញពីកន្សោមគណនាមួយ និងរកលទ្ធផលពិតមិនពិតនៃកន្សោមប្រៀបធៀប។ និមិត្តសញ្ញាដែលមានបន្តបន្ទាប់ក្នុងតារាងដូចបានបង្ហាញខាងក្រោមសំខាន់ប្រើប្រាស់ដើម្បីបង្កើតកន្សោមគណិត តក្កវិទ្យា និងកន្សោមលក្ខខណ្ឌប្រើប្រាស់ក្នុងការសរសេរកូដនៅក្នុង Arduino។ សញ្ញាប្រមាណវិធី គឺជានិមិត្តសញ្ញាដែលប្រាប់អ្នកចងក្រង ដើម្បីអនុវត្តមុខងារគណិតវិទ្យា ឬតក្កវិទ្យា ជាក់លាក់។

- Arithmetic
- Comparison Operators
- Boolean Operators
- Bitwise Operators
- Compound Operators

សញ្ញា	អត្ថន័យ	ឧទាហរណ៍	លទ្ធផលពិតឬមិនពិត
==	ស្មើ	1 + 1 == 2	ពិត
!=	មិនស្មើ	3.2 != 2.5	មិនពិត
<	តូចជាង	10 < 5	មិនពិត
>	ធំជាង	10 > 5	ពិត
<=	តូចជាងឬស្មើ	126 <= 100	មិនពិត
>=	ធំជាងឬស្មើ	5.0 >= 5.0	ពិត

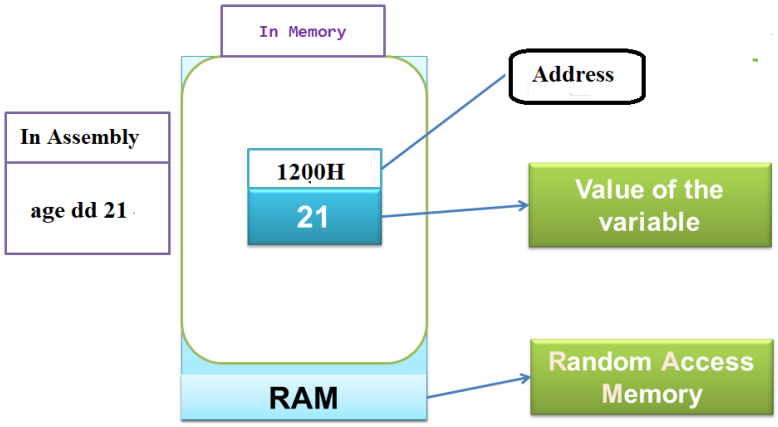
សញ្ញា	អត្ថន័យ	ឧទាហរណ៍	លទ្ធផល
+	បូក	1+1	=2
-	ដក	10-2	=8
*	គុណ	3*2	=6
/	ចែក(លទ្ធផលជាចំនួនទសភាគ)	5/2	=2.5
//	ចែក(លទ្ធផលជាចំនួនគត់)	5//2	=2
%	ចែកយកសំណល់	10%3	=1
**	ស្វ័យគុណ	2**3	=8

3. អថេរ(Variable)

3.1. ការប្រើអថេរ

អថេរប្រើប្រាស់នៅក្នុងការសរសេរកម្មវិធីដើម្បីរក្សាទុកនូវតម្លៃ សម្រាប់ធ្វើប្រមាណវិធីនព្វន្ត, តក្កវិទ្យា និងប្រមាណវិធីផ្សេងៗដទៃទៀត។ វាតាងឱ្យលេខអាសយដ្ឋានរបស់ Memory ដើម្បីផ្ទុកតម្លៃនៅពេលកម្មវិធីមួយត្រូវបាន Compile រួច។ ឧទាហរណ៍នៅក្នុងភាសា Assembly បើយើងប្រកាស:

age db 21 ; ប្រកាសអថេរ age ទំហំ 1 byte និងផ្ទុកលេខ 21។ ពេល Compiler អថេរ age ទៅជាលេខអាសយដ្ឋាន(Address)1200h និងប្រើ Memory ចំនួនមួយទីតាំងរូបខាងក្រោម៖



char x= 21 ក្នុងភាសា C ក៏មានន័យប្រហាក់ប្រហែលនឹងករណីខាងលើដែរ។

3.2. ប្រភេទទិន្នន័យ

គ្រប់អថេរទាំងអស់ត្រូវមានឈ្មោះ និង ប្រភេទទិន្នន័យ(char, int, long int)ប្រើសម្រាប់សម្គាល់នៃប្រភេទអថេរដែលត្រូវផ្ទុកលេខជាចំនួនគត់, ទសភាគ និង អក្សរ និងទំហំដែនកំណត់ដែលត្រូវផ្ទុកនៅក្នុង Memory។ តារាងខាងក្រោមបង្ហាញពីប្រភេទទិន្នន័យប្រើក្នុង Arduio IDE:

Arduino Data Types	Value Assigned	Value Ranges
boolean	8 Bit	True or False
byte	8 Bit	0 to 255
char	8 Bit	-127 to 128
unsigned char	8 Bit	0 to 255
word	16 Bit	0 to 65535
unsigned int	16 Bit	0 to 65535
int	16 Bit	-32768 to 32767
long	32 Bit	-2,147,483,648 to 2,147,483,647
float	32 Bit	-3.4028235E38 to 3.4028235E38

ឧទាហរណ៍: int ledPin = 13; ប្រើឈ្មោះអថេរ ledPin តាងឱ្យអំពូល LED មួយ ហើយភ្ជាប់ទៅជើង 13 ឬអាចនិយាយខ្លីថា ***កំណត់ LED ភ្ជាប់ទៅកាន់ Pin 13*** (sets led to pin 13)។

int buttonPin = 12; ប្រកាសឈ្មោះអថេរ buttonPin តាងឱ្យ button មួយហើយភ្ជាប់ទៅ pin 12 ។ ឬអាចនិយាយខ្លីថា ***កំណត់ button ភ្ជាប់ទៅកាន់ Pin 12*** (sets led to pin 12)។

☞**អនុវត្តទី1:** កម្មវិធី(Sketch)ខាងក្រោមបង្ហាញពីការប្រើប្រាស់អថេរផ្ទុកតម្លៃដើម្បីយកទៅដំណើរការជាមួយ Arduino Board នឹងសៀរៀលម៉ូនីទ័រដែលជា Output Screen ប្រើនៅក្នុង Arduino IDE។

```
//Declares Goble Variable
int num1 = 1000; //ប្រកាសតម្លៃអថេរ num1 =1000
int num2 = 2000; //ប្រកាសតម្លៃអថេរ num1 =1000
int sum =0 ;
void setup( ) {
    // open the serial port at 9600 bps:
     Serial.begin(9600); //បើក Serial port និងកំណត់អត្រាទិន្នន័យគិតជា9600ប៊ីតក្នុងមួយវិនាទី
   sum = num1 + num2;  //ការបូកលេខពីចំនួន
   Serial.print(num1); // បង្ហាញតម្លៃ num1 នៅលើសៀរៀលម៉ូនីទ័រ
   Serial.print("+"); // បង្ហាញសញ្ញា + នៅលើសៀរៀលម៉ូនីទ័រ
   Serial.print(num2); // បង្ហាញតម្លៃ num2 នៅលើសៀរៀលម៉ូនីទ័រ
   Serial.print("=");// បង្ហាញសញ្ញា នៅលើសៀរៀលម៉ូនីទ័រ
   Serial.print(sum);
}
void loop( ) {
}
```

Outpupt On Serial Monitor

☞អនុវត្តទី២: កម្មវិធី Arduino សម្រាប់ការអនុវត្តនេះដោយបញ្ចូលតម្លៃចំនួនទសភាគដូចបង្ហាញខាងក្រោមដើម្បីរកមធ្យមភាគនៃពិន្ទុ បីមុខវិជ្ជា Math, English និង khmer:

```
void setup( ) {
  //Declares Local Variable
     float math =6.50 ; //ប្រកាសតម្លៃអថេរ  math =6.50
     float english=7.25 ; //ប្រកាសតម្លៃអថេរ  english=7.25
     float khmer = 8.75 ; //ប្រកាសតម្លៃអថេរ  khmer = 8.75
     float average = 0;
     average = ( math+english+khmer )/3;
     Serial.begin( 9600 ); //បើក usb ផតនិងកំណត់អត្រាទិន្នន័យគិតជា9600ប៊ីតក្នុងមួយវិនាទី
     Serial.print( "Average =" );
     Serial.println( average ); // បង្ហាញតម្លៃ average នៅលើសៀរៀលម៉ូនីទ័រ

}
void loop( ) {
}
```

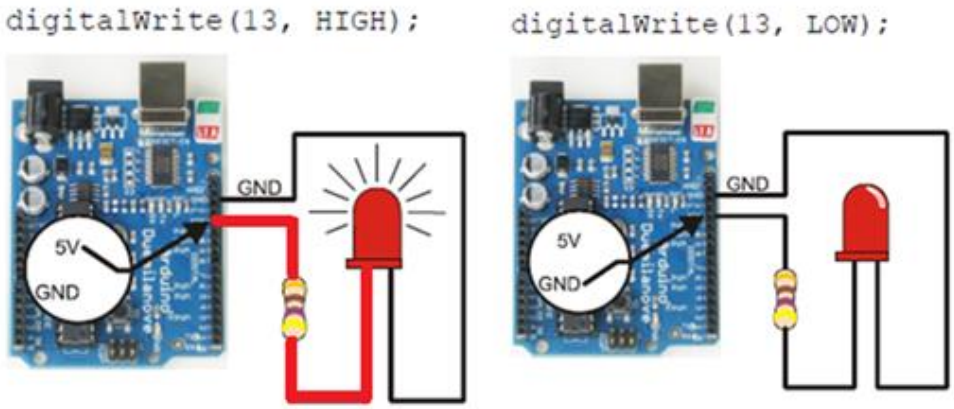
; Output Screen : Average =7.50

☞អនុវត្តទី3: កម្មវិធីគំរូខាងក្រោមធ្វើឱ្យ Led មួយបើករយៈពេល ១000 វិនាទី និងបិទរយៈពេល ១000 វិនាទី:

```
// the setup function runs once when you press reset //or power the board
//===================================================
int ledPin =13 ;//sets led to pin 13
void setup( ) {
  //initialize digital pin 13 as an output.
  pinMode( ledPin, OUTPUT );
}

//the loop function runs over and over again //forever
void loop( ) {
```

```
  digitalWrite( ledPin, HIGH );//turn the LED on
  delay( 1000 ); // wait for a second
  digitalWrite( ledPin, LOW );// turn the LED off
  delay( 1000 );// wait for a second
}
```

Output Screen



4. ការដំឡើង Led ចំនួនពីរ

ការដំឡើងអំពូល Led ចំនួនពីរប្រើជាមួយនឹងប៉ត Arduino និងសរសេរកូដឱ្យ Led ទាំងពីរបើករយៈពេល ១០០០ វិនាទី និងបិទរយៈពេល ១០០០ វិនាទីដោយប្រើកម្មវិធីសរសេរកូដ UnoArduSim និង Proteus ការងារនេះប្រហាក់ប្រហែលអនុវត្តទី ៣ ចំណុច ៣.២.។

4.1. UnoArduSim

4.1.1. ការតភ្ជាប់ឧបករណ៍

- នៅត្រង់ប្រអប់ LED ទីមួយវាយលេខ 05 មានន័យថាភ្ជាប់ Pin ទី ៥ ទៅអំពូល
- នៅត្រង់ប្រអប់ LED ទី២វាយលេខ 06 មានន័យថាភ្ជាប់ Pin ទី ៦ ទៅអំពូល

4.1.2. ការសរសេរកូដ

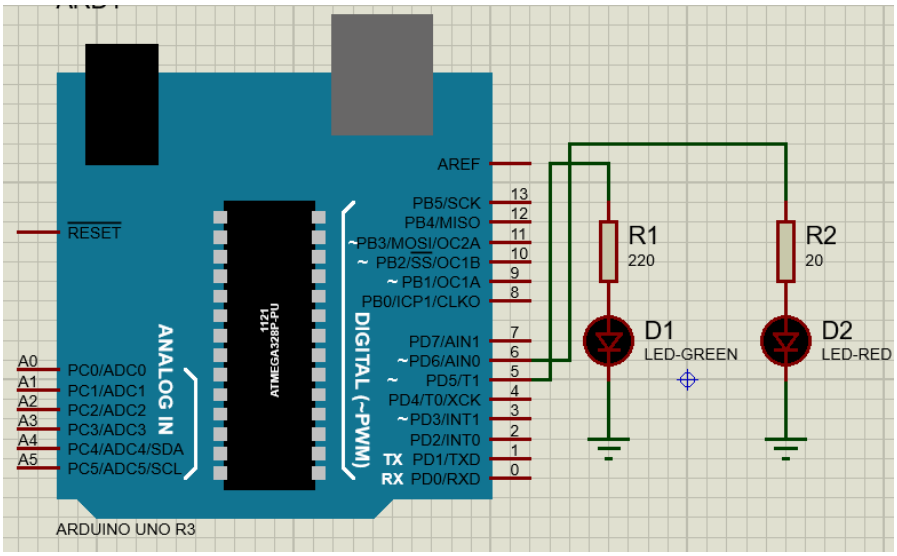
```
int ledPin1 =5 ; //sets led to pin 5
int ledPin2 =6 ; //sets led to pin 6
void setup( ) {
  //initialize digital pin 13 as an output.
   pinMode( ledPin1, OUTPUT );
   pinMode( ledPin2, OUTPUT );
}
//the loop function runs over and over again //forever
void loop( ) {
  digitalWrite( ledPin1, HIGH );//turn the LED on
    digitalWrite( ledPin2, HIGH );//turn the LED on
  delay( 1000 ); // wait for a second
```

```
    digitalWrite(ledPin1, LOW);// turn the LED off
    digitalWrite(ledPin2, LOW);// turn the LED off
    delay(1000);// wait for a second
}
```

4.2. ប្រើកម្មវិធី Proteus

4.2.1. ការតភ្ជាប់ឧបករណ៍

- មុនដំបូងយើងត្រូវបើកកម្មវិធី Proteus រួចចុច Mouse លើចុចលើសញ្ញា
- នៅក្នុងកម្មវិធីចុចលើលើអក្សរ P និងវាយពាក្យ Arduino ក្នុងប្រអប់ keywords និងជ្រើសរើសយក Arduino Uno R3 រួចហើយចុចលើ OK
- នៅក្នុងកម្មវិធីចុចលើលើអក្សរ P និងវាយពាក្យ led red ក្នុងប្រអប់ keywords និងជ្រើសរើសយក LED_RED រួចចុចលើ OK
- ចុច Mouse លើអក្សរ P និងវាយពាក្យ led green រួចជ្រើសរើសយក LED_GREEN ហើយចុច OK
- ចុច Mouse លើអក្សរ P និងវាយពាក្យ res រួចចុច OK បន្ទាប់មកធ្វើការតភ្ជាប់តាមរូបដូចខាងក្រោម៖



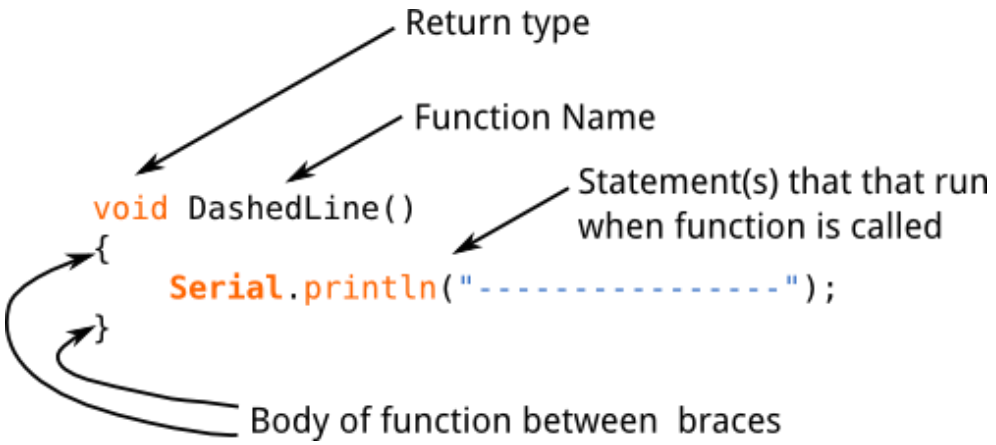
4.2.2. **ការសរសេរកូដ គឺយកតាមគំរូនៅចំណុច៤.១.២. ដាក់ក្នុងកម្មវិធី Arduino IDE និងបំលែងវាទៅ Hexa File។ ហើយនៅក្នុងកម្មវិធី Proteus រួចចុចលើរូបប៉ត Arduino និងជ្រើសយក File Program និងបើក File Hex ទៅតាមទីតាំងវាស្ថិតនៅ និងចុចលើ**

5. អនុគមន៍ (Function)

អនុគមន៍ជាបញ្ជាដែលមានឈ្មោះផ្សេងៗពីគ្នាហើយអាចហៅយកមកប្រើម្តងទៀតទៅតាមកិច្ចការដែលត្រូវធ្វើ និងវាមានស្រាប់នៅក្នុងកម្មវិធី Arduino និង Library នៃឧបករណ៍ណាមួយដែលហៅយកមកប្រើ។ យើងបានសិក្សារួចមកហើយនៅមេរៀនទី៤នូវអនុគមន៍សំខាន់ៗមានដូចជា setup( ), loop( ), digitalWrite( ) ឬ println( )។ វានិមួយៗគឺមានសំនុំនៃពាក្យបញ្ជារៀងៗខ្លួន និងមានតួនាទីខុសៗគ្នា។ នៅក្នុងជំពូកនេះយើងនឹងសិក្សាអំពីអនុគមន៍មួយចំនួនទៀតដែលស្រាប់នៅក្នុងកម្មវិធី Arduino និង អនុគមន៍បង្កើតដោយខ្លួនឯងដើម្បីធ្វើការងារមួយជាក់លាក់មួយ។

5.1. រចនាសម្ព័ន្ធនៃអនុគមន៍

មូលដ្ឋានគ្រឹះក្នុងការបង្កើតអនុគមន៍មួយត្រូវគិតទៅលើបីចំនុចជាសំខាន់។ ចំនុចទី១ឈ្មោះអនុគមន៍និមួយៗត្រូវមានឈ្មោះពិសេសទៅតាមបញ្ហាងាយក្នុងការចំណាំ និងស្រួលប្រើ រួចបញ្ចប់ដោយសញ្ញា ( ) ដើម្បីបញ្ជាក់ថាឈ្មោះនៅពីក្រោយវាជាអនុគមន៍។ ចំនុចទី២ អនុគមន៍ត្រូវមានការឲ្យតម្លៃ ឬគ្មានឲ្យតម្លៃ អាស្រ័យលើការប្រើ Return Type។ ហើយចំនុចទី៣ជារូបរាង នៃអនុគមន៍ដែលត្រូវប្រើសញ្ញា { } ដែលយើងអាចសរសេរពាក្យបញ្ជាមួយឬច្រើននៅក្នុងសញ្ញានេះ។



5.1.1. ឈ្មោះអនុគមន៍

នៅពេលយើងបង្កើតអនុគមន៍ត្រូវដាក់ឈ្មោះដូចគ្នាទៅនឹងប្រើអថេរដែរ៖

- ការបង្កើតឈ្មោះអនុគមន៍ត្រូវចាប់ផ្តើមដោយអក្សរពី A ដល់ Z; a ដល់ z; និងលេខពី 0 ដល់ ៩ ឧទាហរណ៍បង្កើតឈ្មោះ addTwoNumber( ) និងមិនត្រូវចាប់ផ្តើមដោយលេខពីលេខ 0 ដល់ ៩ នៅពីខាងមុខឈ្មោះអនុគមន៍នោះទេ។
- មិនត្រូវប្រើឈ្មោះអនុគមន៍ដែលដូចគ្នានឹងពាក្យគន្លឹះភាសា ឬឈ្មោះអនុគមន៍ដែលមានស្រាប់។
- ការបញ្ជូនតម្លៃ និងការប្រកាសប៉ារ៉ាម៉ែត្រនោះត្រូវបានបញ្ចូលវានៅក្នុងវង់ក្រចក ( )។

ឧទាហរណ៍: int myMultiplyFunction(int x, int y)

5.1.2. Return Type

ជាផ្នែកមួយនៃអនុគមន៍ដើម្បីឲ្យតម្លៃនៃលទ្ធផលមួយទៅឱ្យអនុគមន៍វិញ។ ដោយប្រើជាប្រភេទ int(integer) ឬជា String។ ក្នុងករណីយើងមិនចង់ឲ្យតម្លៃពីអនុគមន៍នោះយើងត្រូវប្រើ void ពីមុខឈ្មោះអនុគមន៍។

5.1.3. Function Body

រូបរាងនៃអនុគមន៍មួយត្រូវបានបង្កើតឡើងជាប្លុក នៃកូដសម្រាប់ដោះស្រាយបញ្ហាឬបញ្ហាណាមួយទៅជាកញ្ចប់ ឬក្រុមមួយ និងមានមុខងារជាក់លាក់មួយ ហើយត្រូវសរសេរវានៅក្នុងនៅចន្លោះដង្កៀប{ }។

☞ **អនុវត្ត១:** កម្មវិធីខាងក្រោមជាគំរូបង្កើតឈ្មោះអនុគមន៍ DashedLine( ) ហើយបានប្រើវានៅក្នុង void setup( ) សម្រាប់ហៅវាមកប្រើធ្វើការគូសបន្ទាត់។

```
void setup( ) {
      Serial.begin(9600); //បើក usb ផត និងកំណត់អត្រាទិន្នន័យគិតជា9600ប៊ីតរៀនាទី
      DashedLine( ); // ជាអនុគមន៍សម្រាប់ធ្វើការបង្ហាញ ……..
      Serial.println("| Program Menu |");
      DashedLine( ); // ជាអនុគមន៍សម្រាប់ធ្វើការបង្ហាញ Dashed Line
}
void loop( ) {
}
     void DashedLine( ) //បង្កើតអនុគមន៍មួយឈ្មោះ DashedLine ( )
{
     Serial.println("----------------");
}
```

Serial Monitor Output

ចំណាំ: អនុគមន៍ឈ្មោះ DashedLine( ) ត្រូវបានប្រើនៅក្នុង void setup( ) ដូចបានបង្ហាញនៅក្នុងកូដខាងលើ និងបន្ទាប់មកត្រូវហៅវាមកប្រើចំនួនពីរដងដើម្បីធ្វើគូសេញ្ញា "……." Dashed Line ចំនួនពីរ។

☞ អនុវត្ត២: កម្មវិធីគំរូខាងក្រោមបង្កើតអនុគមន៍សម្រាប់ឱ្យឲ្យតម្លៃទៅបារ៉ាម៉ែត្រ x និង y

```
void setup( ){
   int i = 2 ; //ប្រកាសតម្លៃអថេរ  i = 2
   int j = 3 ; //ប្រកាសតម្លៃអថេរ j = 3
   int k; //ប្រកាសតម្លៃអថេរ  k= 0
   Serial.begin(9600);
   k = myMultiplyFunction(i,  j); // k now contains 6
   Serial.print("The result multiply of two number is ");
   Serial.println(k);
   delay(500);
}
void loop( ) {
}
int myMultiplyFunction(int x, int y){
        int result;  declares the variable result to stores the valu
```

```
result = x * y;
return result ;
}
```

5.2. analogRead( ): ជាអនុគមន៍ប្រើសម្រាប់បញ្ជូលតម្លៃអាណាឡូក(Analog) អាចមានតម្លៃពី ០ ដល់ ២៥៦ ឬពី ០ ទៅ ១០២៤ ឬលើសពីនេះទៅឱ្យ Analog Pin។ ហើយតម្លៃនិមួយៗអាស្រ័យទៅនឹងការប្រើឧបករណ៍ជាក់ស្តែងដូចជា Motor ឬ Sensor។

5.2.1. ការតភ្ជាប់ឧបករណ៍

- មុនដំបូងយើងត្រូវបើកកម្មវិធី Proteus រួចចុច Mouse លើចុចលើសញ្ញា
- នៅក្នុងកម្មវិធីចុច Mouse លើអក្សរ P និងវាយពាក្យ Arduino រួចជ្រើសរើសយក Arduino Uno R3 ហើយចុច OK
- ចុច Mouse លើអក្សរ P និងវាយពាក្យ pot-hg រួចជ្រើសរើសយក POT_HGហើយចុច OK និងបន្ទាប់មកធ្វើការតភ្ជាប់វាទៅ DC និង Ground
- ចុច Mouse លើ និងជ្រើសយក VIRTUAL TERMINAL និងធ្វើការតភ្ជាប់ដោយ TX Arduino ទៅ RX VIRTUAL TERMINAL និង RX Arduino ទៅ TX VIRTUAL TERMINAL

5.2.2. ការសរសេរកូដតាមគំរូខាងក្រោម និងបំលែងវាទៅ Hexa File។ នៅក្នុងកម្មវិធី Proteus និងត្រង់ File Program បើក File Hex ទៅតាមវាស្ថិតនៅទីតាំងណាមួយ និងចុចលើ

```
// the setup routine runs once when you press reset:
void setup( ) {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);}
// the loop routine runs over and over again forever:
void loop( ) {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);    // delay in between reads for stability
}danologRead(pin)
```

5.3. analogWrite( ): បញ្ចេញតម្លៃ analog ទៅឱ្យ Anolog Pin។

analogWrite(pin) ដោយ pin: គឺលេខនៅលើ Arduino Board ដែលត្រូវបញ្ជូល

ឧទាហរណ៍ខាងក្រោមបង្ហាញពីកូដធ្វើការតេស្តតម្លៃ analog ជាមួយនឹង Motor L293D ដើម្បីកំណត់ល្បឿននៃម៉ូទ័រឱ្យវិលលឿនឬយឺតដោយប្រើកម្មវិធី Proteus។

- **ការដំឡើងផ្នែករឹង**

Arduino board L293D Motor Driver

តភ្ជាប់ Pin10 ទៅ in2 សម្រាប់បញ្ជូនតម្លៃ

តភ្ជាប់ Pin9 ទៅ in1 សម្រាប់បញ្ជូនតម្លៃ

En1 Motor driver តភ្ជាប់ទៅ DC Power រួចបន្តទៅ Vss និង Vs និង GND ទៅ Ground

OUT1 ភ្ជាប់ទៅជើងម្ខាងនៃមូទ័រនិង OUT2 ភ្ជាប់ទៅជើងម្ខាងទៀត

- **ការសរសេរកូដ**ដើម្បីធ្វើការតេស Motor ឱ្យវិលលឿនឬយឺតតាមតម្លៃក្នុងអថេរ val ដោយល្បឿនម៉ូទ័រមានចន្លោះពី ០ ទៅ ២៥៥។

```
int val =150 ; //ប្រកាសអថេរ val ផ្ទុកតម្លៃល្បឿននៃមូទ័រ
void setup( ){
}
Void loop( ){
        analogWrite(9, val); //នៅជើង Pin 9 បញ្ចេញតម្លៃអថេរ val
        analogWrite(10, 0);   //នៅជើង Pin 10 បញ្ចេញតម្លៃអថេរ 0
}
```

6. **ការប្រើបញ្ជា Arduino IF**

ការប្រើបញ្ជា if ដើម្បីពិនិត្យមើលពីការចុចបញ្ចូលតម្លៃដោយប្រើ Button, Keyboard ឬឧបករណ៍ផ្សេងទៀតដើម្បីបញ្ចូលតម្លៃទៅកាន់ Arduino Board សម្រាប់ឱ្យឧបករណ៍មួយផ្សេងទៀតធ្វើអ្វីមួយ។ ការសម្រេចចិត្តអាស្រ័យទៅលើអ្នកប្រើប្រាស់ដោយចុច Key។ ពេលចុច Key មួយត្រូវបានចុចនោះការបញ្ចូលតម្លៃទៅកាន់ Arduino Board គិតលេខ ១ គិតក្នុងប្រព័ន្ធគោលពីរ និងលែងដៃមកវិញគិតលេខ ០។

ឧទាហរណ៍ទី១: កូដខាងក្រោមបង្ហាញពីការប្រើលក្ខខណ្ឌ if ដើម្បីបង្ហាញឃ្លា I am Ardunio recived បានទទួលតម្លៃលេខគោលដប់ប្រាំមួយ និងបង្ហាញវានៅលើ Serial Monitor.

```
int  read_byte=0;  //តាងអញ្ញាត្តិread_byte=0
void setup( ){
        Serial.begin(9600); } // បើក port serial មានល្បឿនបញ្ជូនទិន្នន័យ 9600 bit/s
void loop( )
{
        if(Serial.available( )>0){//ពេលទិន្នន័យត្រូវបានផ្ញើពីសៀរៀលនោះតម្លៃធំជាង0
                read_byte=Serial.read( ); //ឱ្យតម្លៃជាអក្សរទៅសៀរៀលដើម្បីឱ្យវាអាន
                Serial.print("I am Ardunio recived : "); //បង្ហាញសារលើសឺរៀលម៉ូនីទ័រ
                Serial.println(read_byte,HEX); } //សេនតម្លៃនៃពាក្យ I am ardunio recive ជា
        តម្លៃប្រព័ន្ធគោល១៦
}
```

;ការដំណើរការ: ពេលចុចលេខ ១ និងចុច Send នោះបានតម្លៃផ្ញើលេខនេះទៅ Arduino ប៉ត ហើយលទ្ធផលគឺ **I am Ardunio recived :៣១**

ឧទាហរណ៍ទី២: កូដខាងក្រោមបង្ហាញពីការប្រើលក្ខខណ្ឌ if(Serial.available)>0 ហើយលក្ខខណ្ឌនេះពិតពេលតួអក្សរមួយត្រូវបានផ្ញើចេញពី Serial Motor នៃកុំព្យូទ័រ ទៅ Arduino Board។ បញ្ជា Serail.read( ) ប្រើសម្រាប់ចាប់តម្លៃដែលបានផ្ញើមក។ ឧទាហរណ៍ពេលចុច H នៅលើ Serial Monitor នោះអថេរ read_byte = H។

```
void setup( ) {
  Serial.begin( 9600 );//បើកឌុយ usb ដើម្បីវាដំណើរការ នឹងកំណត់ល្បឿនរបស់សៀរៀល
}
void loop( ) {
  char read_byte; ប្រកាសអថេរ read_byte សម្រាប់ផ្ទុកតម្លៃពេលបញ្ចូលពីសៀរៀល
  if ( Serial.available( ) > 0 ) //បើមានទិន្នន័យត្រូវបានផ្ញើពី serial port
  {
        read_byte = Serial.read( ); // ចាប់យកតម្លៃបានបញ្ចូលផ្ទុកនៅក្នុង read_byte
        Serial.print( "You typed: " );// ធ្វើការបង្ហាញឃ្លា You typed:
        Serial.println( read_byte ); }// ធ្វើការបង្ហាញឃ្លាតួអក្សរនៅ read_byte និងចុះបន្ទាប់
}
```

ឧទាហរណ៍2: ចូរធ្វើការតភ្ជាប់ Led មួយជាមួយនឹង Arduino Uno R3 តាមករណីដូចខាងក្រោម៖

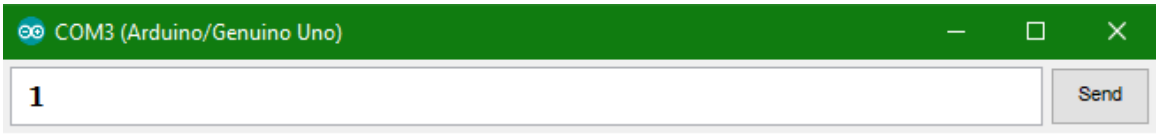
Arduino Uno	LED
GND	Pin –
Pin 13	Pin +

កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើបញ្ជា if ចំនួនពីរដងដើម្បីធ្វើការបិទ ឬបើក អំពូលប្រើជើង Pin 13 ដោយប្រើ Serial Monitor ជាកន្លែងបញ្ជា។

```
//============================================================
void setup( ) {
  Serial.begin( 9600 );
  pinMode( 13, OUTPUT );  // LED on pin 13 of UNO
}
void loop( ) {
  char read_byte;
```

```
  if (Serial.available( ) > 0){// is a character available?
    read_byte = Serial.read( );//ចាប់យកតម្លៃបានបញ្ចូលផ្ទុកនៅក្នុង read_byte
    if (read_byte == '1') { //បើសិន read_byte ស្មើលេខ ១ អនុវត្តខាងក្រោម
      digitalWrite(13, HIGH); //បញ្ចេញលេខ ១ នៅជើង Pin 13 ដើម្បីឱ្យបើកអំពូល
    }
    if (read_byte == '0'){ //បើសិន read_byte ស្មើលេខ 0 អនុវត្តខាងក្រោម
      digitalWrite(13, LOW); //បញ្ចេញលេខ 0 នៅជើង Pin 13 ដើម្បីឱ្យអំពូលបិត
    }
  }
}
```

បន្ទាប់ពីដំណើរការកូដខាងលើ និងប្រើ Serial Monitor យើងត្រូវចុចលេខ ១ និងចុច mouse លើពាក្យ Send សម្រាប់ផ្ញើលេខ ១ ទៅឱ្យ Arduino Board រួចវាដំណើរការបញ្ជា digitalWrite(13,High) ដើម្បីអោយអំពូលភ្លឺ។



នៅលើ Serial Monitor ដដែលពេលចុចលេខ 0វិញ និងចុច mouse លើពាក្យ Send ដើម្បីបិទ Led។

7. ការប្រើ Arduino if...else statement

នៅពេលប្រើបញ្ជា if សំនុំពាក្យបញ្ជានៅក្នុងខ្លួនវាធ្វើការប្រតិបត្តិការតែម្តងនៅពេលលក្ខខណ្ឌពិត។ តែបើមិនពិតរំលងទៅអនុវត្តន៍ Statement នៅក្រៅបញ្ជា if ។ តែបើយើងថែមការប្រើបញ្ជា else មួយទៀតនោះមានន័យថាប្រើបញ្ជា if...else ដែលជាប់ទាក់ទងនឹងការដោះស្រាយចំណោទ ឬបញ្ហាដែលប្រើពីរលក្ខខណ្ឌ។ បើលក្ខខណ្ឌពិតអនុវត្ត StatementA (Statement1, Statement2,..., Statementn) និងមិនពិតវាអនុវត្ត StatementB:

```
If( condinton ){
        Statement A }
Else{
        Statement B}
```

ឧទាហរណ៍ទី១: កូដខាងក្រោមបង្ហាញពីការប្រើប្រាស់បញ្ជា if...else ដោយប្រើសៀរៀលម៉ូនីទ័រ ដើម្បីជាកន្លែងបញ្ជាពេលចុចលេខ១ ក្នុងប្រអប់ TX រួចចុចTOSEND នោះអំពូលបើក និងចុចលេខ0 ក្នុងប្រអប់ TX រួចចុច TOSEND នោះអំពូលបិត។

```
void setup( ) {
  Serial.begin(9600); // បើក USB Portនិងកំណត់ល្បឿនទិន្នន័យ៩៦០០bits ក្នុង១វិនាទី
  pinMode(13,OUTPUT);// កំណត់ pin 13 ជា output
```

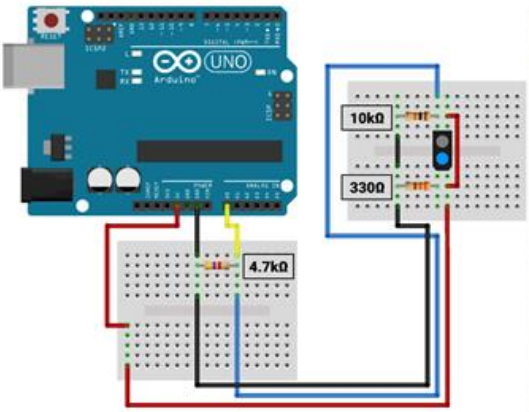
```
}
void loop( ) {
  char read_byte;  //ប្រើសម្រាប់ចាប់តម្លៃពី serial monitor
  if( Serial.available( )>0 ){ // ប្រសិនបើមានតួអក្សរពិត
        read_byte=Serial.read( );//អនុគមន៍ចាប់តម្លៃដែលបាន send ពី serial read
        if( read_byte=='1' ){     // សរសេរដើម្បីចាប់តម្លៃលេខ១
              digitalWrite( 13,HIGH ); // បង្ហាញឱ្យ pin លេខ13ភ្លឺ}
        if( read_byte=='0' ){ // សរសេរដើម្បីចាប់តម្លៃលេខ0
              digitalWrite( 13,LOW ); // បង្ហាញឱ្យ pin លេខ13 រលត់}
  }
}
```

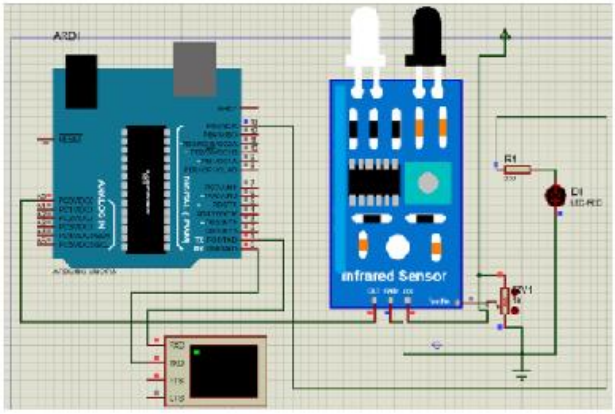
ឧទាហរណ៍ទី ២: កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើ Button មួយដើម្បីធ្វើការបញ្ជាទៅលើLed មួយឱ្យបិទ ឬបើក។ ពេលចុច Button បង្កើតតម្លៃលេខ ១ និងលែងដៃមកវិញបង្កើត បានលេខ 0:

```
const int BUTTON_PIN = 7; //ប្រកាសឲ្យ BUTTON_PIN ទទួលតម្លៃពីជើង pin លេខ 7
const int BUTTON_PIN = 7;  //ប្រកាសអថេរ BUTTON_PIN ទទួលតម្លៃពីជើង pin លេខ 7
const int LED_PIN = 3;  // ប្រកាសឲ្យអថេរ LED_PIN ទទួលតម្លៃពីជើង pin លេខ 3
// variables ដែលនឹងមានការប្រែប្រួល
int buttonState = 0; //ប្រកាសអថេរbuttonState ជាអ្នកចាំទទួលតម្លៃនៃការចុចប៊ូតុងដោយដំបូងឲ្យវាមានតម្លៃ0(មិនទាន់ចុច)
void setup( ) {
        pinMode( LED_PIN, OUTPUT );    //បើកជើង pin ទី3 ជាទិសដៅចេញ
        pinMode( BUTTON_PIN, INPUT )};  //បើកជើង pin ទី៧ ជាទិសដៅចូល
void loop( ) {
buttonState = digitalRead( BUTTON_PIN );  // ឲ្យ buttonState ចាំទទួលការការផ្លាស់ប្តូរតម្លៃពី0ទៅ១ពេលចុច
 // ឃ្លាបញ្ជាទៅកាន់អំពូលLED
if( buttonState == HIGH )   //បើសិនជា buttonStateមានតម្លៃស្មើ១
        digitalWrite( LED_PIN, HIGH )//បញ្ចេញជើងPin៣មានចរន្ត៥វុលដើម្បីឱ្យអំពូលបើក
else  // បើសិនជាbuttonStateមានតម្លៃស្មើ0
        digitalWrite( LED_PIN, LOW )};//បញ្ចេញជើងPin៣មានចរន្ត0វុលដើម្បីឱ្យអំពូលបិត
```

ឧទាហរណ៍៣: ចូរមើល**កូដគំរូ**ខាងក្រោមបង្ហាញពីការប្រើសេនស័រ IR Infrared ដើម្បីមើលតម្លៃវានៅលើ Serial Monitor ដោយប្រើកម្មវិធី Arduino ឬ Proteus។

ការតភ្ជាប់រវាងពី Sensor ជាមួយនឹង Ardunio Uno។ មុខនឹងធ្វើការតេសឧបករណ៍ទាំងនេះ សូមមើលការប្រើបញ្ជា **if...else នៅចំនុចទី៥ខាងក្រោម៖**






```
void setup( ) {
  Serial.begin( 9600 );
  delay( 500 );
  Serial.println( "Welcome to Robotics Lab\n" );
  pinMode( A0,INPUT );
  Serial.begin( 9600 );
  pinMode( 13,OUTPUT );
}
void loop( ) {
  delay( 500 );
  Serial.println( "" );
  //Serial.println( digitalRead( A0 ) );
  if( digitalRead( A0 )==0 )
  {
    digitalWrite( 13,HIGH );
    Serial.print( "Object Found " );
  }
 else{
    digitalWrite( 13,LOW );
    Serial.print( "NO Object Found " );}
}
```

ឧទាហរណ៍៤: ចូរមើលកូដគំរូខាងក្រោមបង្ហាញពីការប្រើបញ្ចូលតម្លៃID នៅលើសៀវរៀល។ បើបញ្ចូល ID ត្រូវវាបង្ហាញ **Valid TAG** នៅលើសៀវរៀលនិងអំពូលភ្លឺ។ ក្នុងករណីបើបញ្ចូល ID ខុសវាបង្ហាញ Invalid TAG នៅលើសៀវរៀលនិងអំពូលបិត។

```
int count = 0;
char c;
String id;
```

```
void setup( ) {
        Serial.begin(9600);
        pinMode(13, OUTPUT);
        Serial.println("Please scan your RFID TAG");
}
void loop( ) {
  while(Serial.available( )>0){
        c = Serial.read( );
        count++;
         id += c;
   if(count == 12)  {
         Serial.print(id);
         //break;
        if(id=="AB123456789A")      { //AB123456789B
                Serial.println("Valid TAG");
                digitalWrite(13, HIGH);
         }
         else{
                digitalWrite(13, LOW);
                Serial.println("Invalid TAG");}}
  }
  count = 0;
  id="";
  delay(500);
}
```

8. Arduino Loops

ការប្រើរង្វិលជុំដើម្បីធ្វើការប្រតិបត្តិពាក្យបញ្ជាមួយ ឬច្រើនក្នុងទំរង់ដដែលៗសម្រាប់ធ្វើប្រមាណវិធីដែលមាន n ចំនួន។ ចំពោះការប្រើរង្វិលជុំជាមួយនឹងឧបករណ៍ឌីជីថលអេឡិចត្រូនិចដើម្បីឱ្យវាដំណើរការហូតទៅតាមកូដដែលយើងបានបញ្ចូលទៅក្នុង Flash Memory នៃប៉ត Arduino ឱ្យ CPU ដំណើរការ។

8.1. ការប្រើប្រាស់ for loop

Sketch ខាងក្រោមបង្ហាញពីការប្រើប្រាស់ for loop:

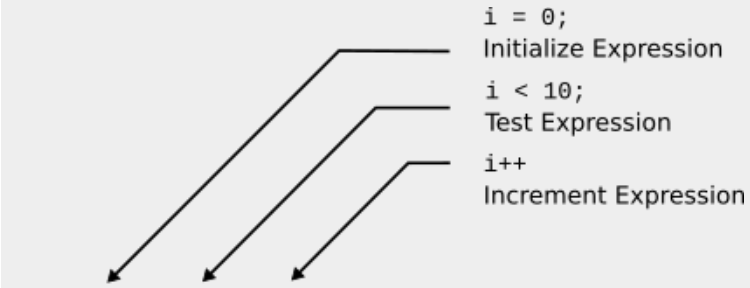
```
void setup( ) {
  int i;
  Serial.begin(9600);
  for (i = 0; i < 10; i++) {
    Serial.print("i = ");
```

```
    Serial.println(i);
  }
}
void loop( ) {
}
```

Output Screen:

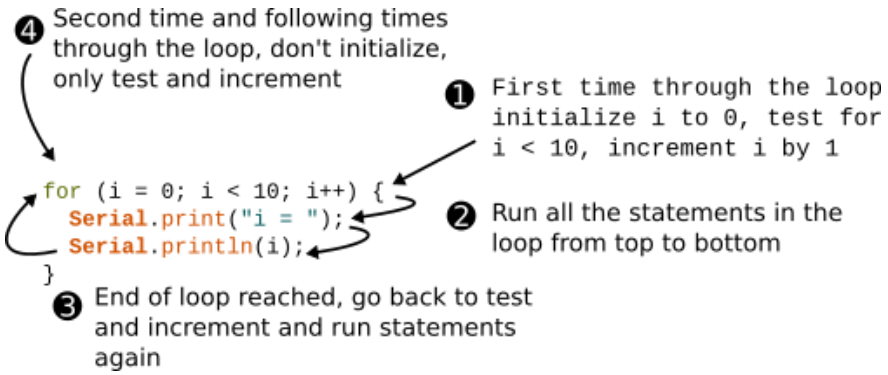
ជាក់រូប Output

តាមរូបខាងក្រោមបង្ហាញពីដំណើរការ for loop



នៅក្នុង For Loop មានកន្សោមបីសំខាន់គឺ initialize expression, Test Expression, និង Increment Expression។ កន្សោមទាំងបីចែកដាច់ពីគ្នាដោយ a semicolons (;)។

កន្សោមដំបូងគឺ initialize expression វាដំណើរការតែម្តងនៅពេល loop ចាប់ផ្តើមដំណើរការ ដើម្បីរកតម្លៃដំបូងសម្រាប់អថេរ i=0 និងប្រើវាសម្រាប់បង្កើនតម្លៃ i នៅ Loop បន្ទាប់។ តាមកូដឧទាហរណ៍ខាងក្រោម i < 10(Test Expression)គឺជាកន្សោមលក្ខខណ្ឌដើម្បីបន្ត loop ឬមិនបន្ត។ នៅពេលកន្សោមលក្ខខណ្ឌពិតធ្វើការបន្ត Loop និងកន្សោមលក្ខខណ្ឌមិនពិតចាកចេញពី Loop។ ចំណែកឯកន្សោម i++(Increment Statement) ធ្វើការបន្ថែមតម្លៃម្តងមួយឱ្យ i។



ឧទាហរណ៍: កូដខាងក្រោមធ្វើបិទ ឬបើកអំពូលចំនួន ៤

```
int timer = 200;// The higher the number, the slower the timing.
void setup( ) {
   // use a for loop to initialize each pin as an output:
   for ( int thisPin = 9; thisPin <13; thisPin++ )  {
```

```
        pinMode(thisPin, OUTPUT);      }
    }
    void loop( ) {
      // loop from the lowest pin to the highest:
       for (int thisPin = 9; thisPin <13; thisPin++) {
            digitalWrite(thisPin, HIGH);   // turn the pin on:
            delay(timer);
            digitalWrite(thisPin, LOW);     }// turn the pin off:
            // loop from the highest pin to the lowest:
            for (int thisPin = 13; thisPin > 9; thisPin--) {
            digitalWrite(thisPin, HIGH);    // turn the pin on:
            delay(timer);// turn the pin off:
            digitalWrite(thisPin, LOW); }
    }
```

8.2. ការប្រើប្រាស់ while loop

ការប្រើ while ប្រហាក់ប្រហែលនឹងការប្រើរង្វិលជុំ For ដែរ។ តែខុសគ្នាត្រង់ការប្រើរង្វិលជុំ While គឺបំបែកកន្សោមនៃរង្វិលជុំនៅដាច់ៗពីគ្នា។ វាត្រូវបានប្រើប្រាស់សម្រាប់ដំណើរការប្តូកនៃកូដនៅក្នុងខ្លួនវាម្តងហើយម្តងទៀតរហូតដល់កន្សោមលក្ខខណ្ឌមិនពិតទើបចាកចេញពី Loop។

Syntax:

```
while (condition) { //បើលក្ខខណ្ឌពីអនុវត្ត statement(s)
        // statement(s) អាច statement1, statement2,..., statementn
}
```

ឧទាហរណ៍១: កម្មវិធីខាងក្រោមគឺយកពីកូដខាងលើដែលប្រើរង្វិលជុំ For ដោយគ្រាន់តែជំនួសប្រើ While វិញ។ ហើយការរង្វិលជុំនេះយើងបានអនុវត្តច្រើនហើយនៅក្នុងភាសា Assembly។

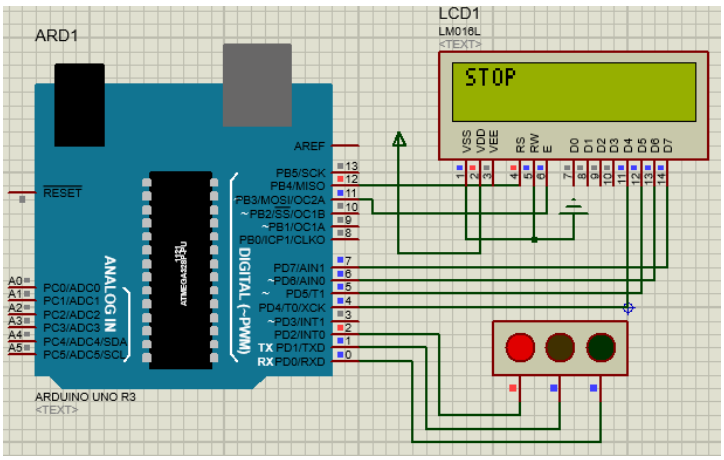
```
void setup( ) {
    int i = 0 ; // កំណត់អថេរ i ឱ្យមានតម្លៃដំបូងស្មើនឹង ១
    Serial.begin(9600);// // បើក port serial(COM1,COM2,..) ល្បឿន 9600 bps
    while (i < 10) { // បើតម្លៃ  i លេខក្រោម ១0
        Serial.print("i = ");// បង្ហាញអក្សរ នៅលើសៀវៀលម៉ូនីទ័រ
        Serial.println(i); "); // បង្ហាញតម្លៃ i នៅលើសៀវៀលម៉ូនីទ័រ
        i++;} // i = i + 1 បង្កើតតម្លៃ i ម្តងមួយ
    }
```

ឧទាហរណ៍២: ចូរសរសេរកូដដើម្បីឱ្យ:

១. LED ពណ៌ក្រហមបើក និងនៅលើអេក្រង់ LCD ចេញពាក្យ STOP

២. LED ពណ៌លឿងបើក និងនៅលើអេក្រង់ LCD ចេញពាក្យ WAIT

៣. LED ពណ៌បៃតងបើកនិងនៅលើអេក្រង់ LCD ចេញពាក្យ GO




```
#include<LiquidCrystal.h>
//GREEN = 0; YELLOW = 1; RED = 2;
const int rs = 12, en = 11, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd( rs, en, d4, d5, d6, d7 );
int i=0 ;
void setup( )
{
        Lcd.begin( 16, 2 );// set up the LCD's number of columns and rows:
        lcd.print( "" );// Print a message to the LCD
        while( i<3 ){
                pinMode( i, OUTPUT ); //Turn on LED red , yellow and green
                i++;} // Increment the valu of i
}
void loop( ){
```

no_light( );
delay(1000);
lcd.display( );
lcd.clear( );
red_light( );
delay(1000);
lcd.print("STOP");
delay(1000);
lcd.clear( );
yellow_light( );
delay(1000);
lcd.print("WAIT");
delay(1000);
	void no_light( ){
	int j=0;
	while(j<3){
	digitalWrite(i, LOW);
	j++;}
	}
	void red_light( )
	{ digitalWrite(0, LOW);
	digitalWrite(1, LOW);
	digitalWrite(2, HIGH);}
	void yellow_light( )
	{ digitalWrite(0, LOW);
	digitalWrite(1, HIGH);

```
        lcd.clear( );                    digitalWrite(2, LOW);}
        green_light( );          void green_light( )
        delay(1000);                    { digitalWrite(0, HIGH);
        lcd.clear( );                     digitalWrite(1, LOW);
        lcd.print("GO");                  digitalWrite(2, LOW);}
        delay(1000);
        lcd.clear( );            }
```

9. Arduino Array

អារេ(Array) គឺជាសំនុំធាតុ ឬ វត្ថុដែលមានប្រភេទទិន្នន័យដូចគ្នា និងមានឈ្មោះរួមគ្នាពេលផ្ទុកនៅក្នុងអង្គចងចាំ។ ការប្រកាសអារេត្រូវចាប់ផ្តើមដោយឈ្មោះអថេរ និងចំនួនធាតុ ឬជាសំនុំទិន្នន័យ។ យើងអាចកំណត់ធាតុនៃអារេជាចំនួនគត់ ដែលជាចំនួនពីរឬច្រើនធាតុដែលកើតឡើងពីមួយទៅមួយទៀត។

ចំនុចសំខាន់នៅត្រង់នេះគឺថាធាតុនីមួយៗនៅក្នុងអារេត្រូវបានដាក់ចាប់ពីធាតុមួយទៅធាតុបន្តបន្ទាប់ទៀតដោយខណ្ឌពីគ្នាដោយប្រើសញ្ញា "," ដែលអនុញ្ញាតឱ្យយើងចូលប្រើធាតុនីមួយៗជាវេនដោយប្រើរង្វិលជុំ(loop)។

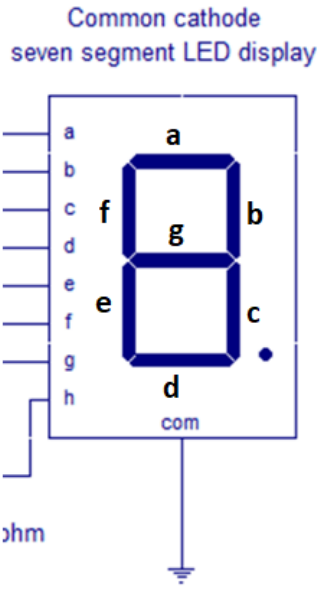
ឧទាហរណ៍១: កូដខាងក្រោមបង្ហាញពីមូលដ្ឋានគ្រឹះនៃការប្រើ Array : **void setup( ) {**

```
int my_array[5];     // an array with 5 integer elements
int I;               // used as an index into the array
Serial.begin(9600);
my_array[0] = 2;    // assign a value my_array[0]=2 to the 1st element
my_array[1] = 3;  // assign a value my_array[1]=3  to the 2nd element,
my_array[2] = 4; // assign a value my_array[2]=4  to the 3nd element,
my_array[3] = 5; // assign a value my_array[3]=5  to the 4nd element,
my_array[4] = 6;
// display each number from the array in the serial monitor window
for (i = 0; i < 5; i++) {
        Serial.println(my_array[i]);}// បង្ហាញតម្លៃពីលេខ ២ ដល់លេខ៦
}
void loop( ) {}
```

//Output Screen

ឧទាហរណ៍២: កម្មវិធីខាងក្រោមបង្ហាញការប្រើប្រាស់ 7 Segment សម្រាប់ធ្វើការបង្ហាញលេខពី ០ ដល់លេខ ៩។ តារាង និងកូដខាងក្រោមប្រើប្រាស់សម្រាប់ធ្វើការតភ្ជាប់ពីរវាង 7 Segment ទៅ Arduino ដោយមើលរូបនិងកូដខាងក្រោម។ ៧ Segment មានពីរប្រភេទគឺ Common cathode និង Common anode -ការដំឡើង 7 Segment ខាងក្រោមទៅ Arduino សូមមើលកូដខាងក្រោម៖

Common Cathode												
Decimal digit	Input lines				Output lines							Dispay
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	1	0	1	1	9




```
//ការដំឡើង Arduino Board ជាមួយនឹង  7 Segment មើលគំរូខាងក្រោម:
//                  Arduino pin: 2, 3, 4, 5, 6,7, 8
//                  7 Seg Pin:   a b c d e f g ប្រើក្នុងកម្មវិធី Proteus
//                  Number :     7 6 4 2 1 9 10 ប្រើជាមួយនឹងឧបករណ៍ជាក់ស្តែង
// ក្នុងករណីប្រើ common anode ប្រើលេខ 1 សម្រាប់ឱ្យអំពូល ON, 0 អំពូល OFF
// ក្នុងករណីប្រើ common cathode ប្រើលេខ 0  សម្រាប់ឱ្យអំពូល OFF, 1 អំពូល ON
//កម្មវិធីកំណត់ LED digit pattern បង្ហាញលេខពី0 ដល់ ៩ នៅ ៧ Segement
byte seven_seg_digits[10][7] = { { 0,0,0,0,0,0,1 },  // 0,0,0,0,0,0,1 ជាកូដចេញលេខ0
                                 { 1,0,0,1,1,1,1 },  /// 0,0,0,0,0,0,1 ជាកូដចេញលេខ 1
                                 { 0,0,1,0,0,1,0 },  // 0,0,1,0,0,1,0 ជាកូដចេញលេខ 2
                                 { 0,0,0,0,1,1,0 },  // 0,0,0,0,1,1,0 ជាកូដចេញលេខ 3
                                 { 1,0,0,1,1,0,0 },  // 1,0,0,1,1,0,0 ជាកូដចេញលេខ 4
                                 { 0,1,0,0,1,0,0 },  // 0,1,0,0,1,0,0 ជាកូដចេញលេខ 5
                                 { 0,1,0,0,0,0,0 },  // 0,1,0,0,0,0,0 ជាកូដចេញលេខ 6
                                 { 0,0,0,1,1,1,1 },  // 0,0,0,1,1,1,1 ជាកូដចេញលេខ 7
                                 { 0,0,0,0,0,0,0 },  // 0,0,0,0,0,0,0 ជាកូដចេញលេខ ៨
                                 { 0,0,0,1,1,0,0 }   // 0,0,0,1,1,0,0 ជាកូដចេញលេខ}; 9
};
/*If you were using a common anode LED you would use this section
 byte seven_seg_digits[10][7] = { / កំណត់ Arrays មាន១0ជួរដេក និង ៧ជួរឈរ
```

```
                 { 1,1,1,1,1,1,0 },  //ជាកូដចេញលេខ 0
                { 0,1,1,0,0,0,0 },  //ជាកូដចេញលេខ 1
                { 1,1,0,1,1,0,1 },  // ជាកូដចេញលេខ 2
                { 1,1,1,1,0,0,1 },  //ជាកូដចេញលេខ 3
                { 0,1,1,0,0,1,1 },  //ជាកូដចេញលេខ 4
                { 1,0,1,1,0,1,1 },  //ជាកូដចេញលេខ 5
                { 1,0,1,1,1,1,1 },  // ជាកូដចេញលេខ 6
                { 1,1,1,0,0,0,0 },  //ជាកូដចេញលេខ 7
                { 1,1,1,1,1,1,1 },  // ជាកូដចេញលេខ 8
                { 1,1,1,1,0,1,1 }   // ជាកូដចេញលេខ   };*/
void setup( )
{
   pinMode(2, OUTPUT);      // 2--->7         A
   pinMode(3, OUTPUT);      // 3--->6         B
   pinMode(4, OUTPUT);      // 4--->4         C
   pinMode(5, OUTPUT);      // 5--->2         D
   pinMode(6, OUTPUT);      // 6--->1         E
   pinMode(7, OUTPUT);      // 7--->9         F
   pinMode(8, OUTPUT);      // 8--->10        G
//    pinMode(9, OUTPUT);// 9-->RDP decimal point
//    writeDot(0);  // start with the "dot" off
}
//void writeDot(byte dot) {
//    digitalWrite(9, dot); // decimal point - pin 9 on Arduino
//}
void sevenSegWrite(byte digit) {
   byte pin = 2; // Arduino pin where the segment list starts
   for (byte segCount = 0; segCount < 7; ++segCount) {
     digitalWrite(pin, seven_seg_digits[digit][segCount]);
     ++pin;}
}
// displays digits from 0 to 9
void loop( ) {
   for (byte count = 0; count < 10; ++count) {
       delay(1000);
       sevenSegWrite(count ); }
```

```
    // delay(4000);
    }
```

10. Arduino Class

Class នៅក្នុងភាសា Assembly ឬភាសា C/C++ ជាប្លង់មេ(Blueprint) ឬពុម្ពគំរូដើមប្រើសម្រាប់បង្កើតនូវវត្ថុ (Objects) ផ្សេងៗ ដែលមានព័ត៌មានលម្អិតទាំងអស់នៃវត្ថុមួយៗ វាជាបណ្តុំនៃលក្ខណៈ ឬជា Properties និង អនុគមន៍(Function)ដែលទាំងពីរជាសមាជិកនៃ Class។ Object គឺជាឧបករណ៍ផ្សេងៗ ឬមនុស្ស សត្វដែលមាន ប្រភេទច្បាស់លាស់ណាមួយ ឬអាចនិយាយបានថាជាលក្ខណៈជាក់លាក់របស់ class។ រាល់វត្ថុទាំងអស់ត្រូវមានរូបរាង និងសកម្មភាពជាក់លាក់ច្បាស់លាស់។

```
    Class name{
    public:
        statement(s);
    }
```

ឧទាហរណ៍១: កម្មវិធីខាងក្រោមគឺបង្ហាញពីគំរូការបង្កើត Class

```
class LED_Controls {
public:
     void Begin(byte GPIO_pin) {
          pinMode(GPIO_pin, OUTPUT);// sets GPIO_pin as OUTPUT }
   void ON(byte GPIO_pin){
          digitalWrite(GPIO_pin, HIGH);// Turn ON LED}
   void OFF(byte GPIO_pin){
         digitalWrite(GPIO_pin,LOW);}
   void flash(byte GPIO_pin, int flash_speed, int flashes){
          for (int f=0; f < flashes; f = f +1){
                ON(GPIO_pin);
                delay(flash_speed);
                OFF(GPIO_pin);
                delay(flash_speed);}
   }
};
// Class      Object
LED_Controls   LED;
void setup( ) {
          LED.Begin(LED_BUILTIN);}
void loop( ) {
          LED.ON(LED_BUILTIN);
           delay(1000);
```

```
            LED.OFF(LED_BUILTIN);
             delay(1000);
            LED.flash(LED_BUILTIN,100,10);
}
```

ឧទាហរណ៍២: កូដខាងក្រោមបង្ហាញពីបង្កើត Objects នៅក្នុង Class សម្រាប់ធ្វើការចុច Key នៅលើ Keypad៖

```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
//define the symbols on the buttons of the keypads
char Keys[ROWS][COLS] =
{
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}};
    byte rowPins[ROWS] = {5,4,3,2};
    byte colPins[COLS] = {9,8,7,6};
    //initialize an instance of class NewKeypad
    Keypad customKeypad = Keypad(makeKeymap(Keys) ,rowPins, colPins,ROWS,COLS);
    void setup( )
    {
      Serial.begin(9600);
      Serial.println("Please press the keyboard:");}
    void  loop( )
    {
      char key = customKeypad.getKey( );
      if(key!=NO_KEY)
      {
            Serial.print("Key Value : ");
            Serial.println(key);}
    }
```

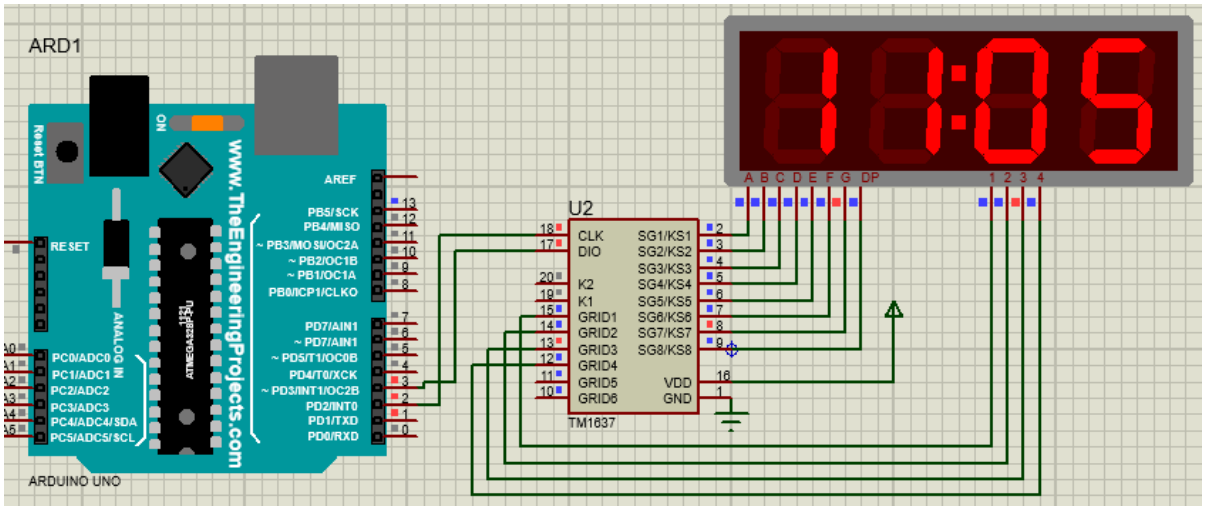
ឧទាហរណ៍៣: កូដខាងក្រោមបង្ហាញពីបង្កើត Objects នៅក្នុង Class សម្រាប់ធ្វើការបង្ហាញពាក្យ " Hello Word " នៅលើ Lcd:

```
    #include <Wire.h>
    #include <LiquidCrystal.h>
    #include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C lcd( 0x27,16,2 ); // set the LCD address to 0x3F for a 16 chars and 2 line
display
void setup( )
{
   //lcd.begin( );.
   lcd.init( ); // initialize the lcd
   // Print a message to the LCD.
   lcd.backlight( );
   lcd.setCursor( 0,0 );
   lcd.print( "I Love RUPP" );
   lcd.setCursor( 1,1 );
   lcd.print( "A3 Students" );
}
void loop( ){}
```

សំណួរ និងលំហាត់

1. ដូចម្តេចដែលហៅថាភាសា Arduino? គេប្រើវាសម្រាប់ធ្វើអ្វី?
2. ដូចម្តេចដែលហៅថា (a).Operator, (b). Data Type និង (c). Function?
3. ចូរនិយាយពីភាពខុសគ្នារវាង if statement និង if...else statement។
4. ចូរនិយាយពីភាពខុសគ្នារវាង for និង while loop។ ចូរឱ្យពីដំណើរការនៃរង្វិលជុំទាំងពីរ។
5. ដូចម្តេចដែលហៅថា (a).Array, (b). pointer? ចូរឱ្យឧទាហរណ៍ពីការប្រើ Array និង Pointer
6. ដូចម្តេចដែលហៅថា Class? គេប្រើវាសម្រាប់ធ្វើអ្វី? ចូរឱ្យឧទាហរណ៍ពីការបង្កើត Class ក្នុងកម្មវិធី រួចពន្យល់វាឱ្យបានក្បោះក្បាយ។
7. ចូរសរសេរកូដដើម្បី:
   a. ឱ្យ ៤ Led On/OFF
   b. បង្ហាញឃ្លា "Hello Rupp" និងឃ្លានេះរត់ពីខាងស្តាំទៅខាងឆ្វេងនៅលើ LCD
   c. ប្រើសេនស័រចំនួន ៥ TRC5000 ឬ IR infrared និងនៅពេល Sensor1, 2, 3, 4, 5 detect ឱ្យ Led1, 2, 3, 4, 5 ON/OFF តាមលំដាប់រៀង។ មានន័យថាពេល Sensor1, detect ឱ្យ Led1 ភ្លឺ និងមិន Detect រលត់អំពូលទាំងអស់រលត់។
   d. ចុចប៊ូតុងបួនតាមលំដាប់ឱ្យ led1, led2, led3, led4 នៅលើតេស្ត Board RUPPភ្លឺ ឬរលត់ និងឱ្យចេញលេខ 1, 2, 3, 4 នៅលើ Serial Monitor។
   e. ធ្វើអំពូលចរាចរនៅលើតេស្ត Board Traffic Light ឱ្យដូចទៅនឹងដងផ្លូវប្រើសព្វថ្ងៃ
   f. តេស្តអំពូលមួយដោយប្រើ Bluetooth
8. ចូរសរសេរកម្មវិធីដើម្បីបង្ហាញម៉ោងនៅលើ GROVE4DIGIT តាមរូបខាងក្រោម:



9. ចូលសរសេរកូដពេលចុចលេខនៅលើ Keypad ឱ្យវាដំណើរការទៅតាមលេខ អក្សរ និងសញ្ញាដែលមាននៅលើ Keypad។

មេរៀនទី ៩ គម្រោងគំរូ

នៅក្នុងជំពូកនេះយើងនឹងសិក្សាអំពីការបង្កើតឧបករណ៍អេឡិចត្រូនិក និងឧបករណ៍ឆ្លាតវៃ ដែលជាគម្រោងគំរូឱ្យមានការប្រើប្រាស់ និងការដំឡើងឧបករណ៍ជាក់ស្តែងឱ្យទៅជាឧបករណ៍មួយអាចដែលអាចប្រើបាននៅក្នុងផ្ទះ, សាលារៀន ទីសាធារណៈ ឬកន្លែងដទៃផ្សេងទៀត។ រៀនពីការបង្កើត RC car ដែលរ៉ូបូតដំបូង និងជាគ្រឹះសម្រាប់ធ្វើការបង្កើតរ៉ូបូតធំៗផ្សេងៗ ដូចជារ៉ូបូតបោសផ្ទះ, រាវមីន, កាត់ស្មៅ ឬរ៉ូបូតប្រកួតជាដើម។ ចំនុចបន្ទាប់គឺការសរសេរកូដដើម្បីបញ្ជាទៅលើឧបករណ៍ឆ្លាតវៃ និង រ៉ូបូតដែលបានបង្កើតរួចហើយឱ្យដំណើរការតាមការបញ្ជារបស់យើង និងដំណើការដោយស្វ័យប្រវត្តិ។

កម្មវិធី Proteus គឺជាកម្មវិធីកុំព្យូទ័រមួយមានលក្ខណៈជា virtual Machine ដែលជាកម្មវិធីសិប្បនិម្មិត (emulator) ដែរអោចឲ្យអ្នកប្រើប្រាស់អាចរចនាឧបករណ៍ផ្សេងៗដូចទៅនឹង ឧបករណ៍ពិតៗ ដែរ។ គេប្រើប្រាស់វាសម្រាប់ធ្វើតេសឧបករណ៍ផ្សេងៗ ដូចជា LED, Bluetooth, Arduino Board ... ជាដើម មុននិងឈានចូលទៅប្រើប្រាស់ឧបករណ៍ជាក់ស្តែង។

1. ឧបករណ៍បញ្ជាភ្លើងដោយឥតប្រើខ្សែប្រើ Bluetooth(Bluetooth controlled lamp)

ការបញ្ជាអំពូលឲ្យបើក ឬបិទ មានច្រើនវិធីណាស់។ ដូចជាការបញ្ជាដោយប្រើកុងតាក់, បញ្ជាដោយប្រើ wireless ការបញ្ជាដោយប្រើសំលេង និងវិធីផ្សេងដទៃច្រើនទៀត។ ប៉ុន្តែនៅក្នុងមេរៀននេះយើង នឹងសិក្សាស្វែងយល់អំពីការគ្រប់គ្រង ឬការបញ្ជាអំពូលភ្លើងតាមរយៈ Wireless គឺ Bluetooth (ឧបករណ៍គ្មានខ្សែ)។ មិនតែប៉ុណ្ណោះ នឹងស្វែងយល់កាន់តែច្បាស់អំពីឧបករណ៍នីមួយៗ(Components) ការតភ្ជាប់(Connection) និងការសរសេរកូដ(Code)។ វាមូលដ្ឋានគ្រឹះដ៏សំខាន់សម្រាប់ធ្វើគម្រោងផ្សេងៗទៀតដែលមាននៅក្នុងមេរៀននេះ។

1.1. ដោយ BCL(Bluetooth controlled lamp)**?**

វាគឺជាឧបករណ៍អេឡិត្រូនិចមួយ ដែលយើងអាចគ្រប់គ្រងបានពីចម្ងាយដោយធ្វើការបញ្ជាតតាមរយៈប្រើទូរស័ព្ទជាតេឡេបញ្ជា។ ពេលចុចប៊ូតុង **Turn On** នៅលើកម្មវិធីទូរសព្ទបញ្ជូនString ១ ទៅប្លឺធូស និងពេលចុចប៊ូតុង **Turn OFF** បញ្ជូន String 0 ទៅប្លឺធូស។

1.2. ឧបករណ៍សម្រាប់ធ្វើ Bluetooth control lamp

ឧបករណ៍សម្រាប់គម្រោងនេះមានដូចជា៖

1.2.1. Smart Phone: ជាទូរស័ព្ទឆ្លាតវៃដែលអាចប្រើប្រាស់នូវមុខងារមួយចំនួនដូចជា Bluetooth ឬ Wifi ជាដើម។ ប្រើវាជា Remote Control

1.2.2. Arduino Uno វាជា System Unit ដែលវាមានតួនាទីផ្ទុកកូដដែលបានបញ្ចូលពីកុំព្យូទ័រ សម្រាប់ធ្វើការបញ្ជាទៅលើឧបករណ៍ដែលបានភ្ជាប់ជាមួយនឹងវាដើម្បីឱ្យដំណើរការទៅតាមកូដដែលបានបញ្ចូលក្នុងខួរក្បាលរបស់វា។

1.2.3. Relay ជាឧបករណ៍អគ្គិសនីមានតួនាទីបញ្ជូនបន្តនូវចរន្តអគ្គិសនី។ វាត្រូវបានគេប្រើសម្រាប់បញ្ជូនបន្តនូវចរន្តអគ្គិសនី ២២០វ៉ុល ឬ ក្រោមនេះទៅកាន់ឧបករណ៍ផ្សេងៗ ដោយវាអាចបើកឬបិទនូវចរន្តអាស្រ័យលើការបញ្ជា របស់ Arduino board។

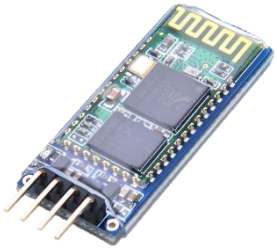
COM សម្រាប់ភ្ជាប់ទៅជើងម្ខាងនៃអំពូល និង NC ភ្ជាប់ទៅប្រភពភ្លើងសម្រាប់ធ្វើការភ្ជាប់ចរន្ត ឬ ផ្តាច់ចរន្តនិងជើងម្ខាងទៀតនៃអំពូលត្រូវភ្ជាប់ទៅកាន់ប្រភពចរន្តអគ្គិសនីដោយផ្ទាល់។

NO: (Often Beginning NO)

VCC: សម្រាប់ភ្ជាប់ទៅជើង ៥វ៉ុលនៃ Arduino

IN: សម្រាប់ភ្ជាប់ទៅជើង PIN នៃ Arduino

GND: សម្រាប់ភ្ជាប់ទៅជើង GND នៃ Arduino



1.2.4. Bluetooth: ប៊្លូធូសគឺជាស្តង់ដាបច្ចេកវិទ្យាឥតខ្សែសម្រាប់ធ្វើការផ្លាស់ប្តូរទិន្នន័យក្នុងចម្ងាយខ្លី និង ការតភ្ជាប់ឥតប្រើខ្សែ។ ទិន្នន័យត្រូវបានបញ្ជូនទូរស័ព្ទចល័តឬ កុំព្យូទ័រទៅកាន់ Bluetooth រួចទៅ Arduino Board។ មានជើង pin ចំនួន ៤ គឺ VCC, GND, TXD និង RXD ប៊្លូធូសអាចភ្ជាប់បច្ចេកវិទ្យាដែលអនុញ្ញាតឱ្យអ្នកភ្ជាប់ឧបករណ៍ជាច្រើននៅក្នុងផ្ទះរបស់អ្នកដូចជាវ៉ាយហ្វាយអ្នកអាចប្រើប៊្លូធូ ដើម្បីផ្លាស់ប្តូរឯកសារនិងធ្វើប្រតិបត្តិការអេឡិចត្រូចនិច ហើយកាសឥតខ្សែ កាសទូរស័ព្ទដៃនិង ហ្គេម កម្យាន្តអាចប្រើប៊្លូធូតាមវិធីមួយឬច្រើន។

1.2.5. Lamp: គឺជាអំពូលមួយប្រើវាសម្រាប់បំភ្លឺភ្លើងពេលយើងចុចប៊ូតុង ON ឬ OFF នៅលើកម្មវិធីដែលបានដំឡើងនៅលើទូរស័ព្ទដៃដោយប្រើប្រព័ន្ធ Android។

1.3. **របៀបតភ្ជាប់** Hardware

- **ការតភ្ជាប់ Bluetooth ទៅកាន់ Arduino uno:**
  - GND នៃ Bluetooth ភ្ជាប់ទៅ Pin GND នៃ Arduino uno
  - VCC នៃ Bluetooth ភ្ជាប់ទៅ Pin 5v នៃ Arduino uno
  - TX នៃ Bluetooth ភ្ជាប់ទៅ RX នៃ Arduino uno
  - RX នៃ Bluetooth ភ្ជាប់ទៅ TX នៃ Arduino uno

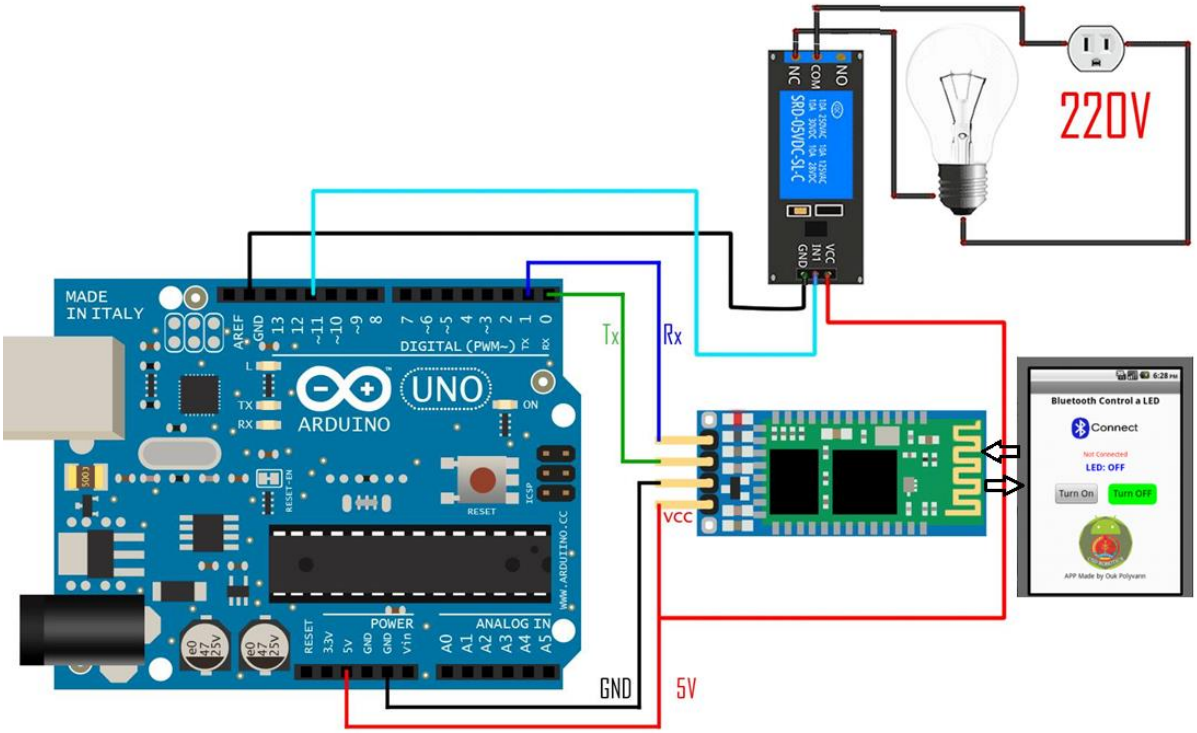
  សូមមើលរូបខាងក្រោមដើម្បីធ្វើការតភ្ជាប់រវាង Arduino Board និង Relay និងទៅ Relay
- **ការតភ្ជាប់ Relay to Arduino**
  - VCC នៃ relay តភ្ជាប់ទៅកាន់ pin 5v នៃ Arduino
  - GND នៃ relay តភ្ជាប់ទៅកាន់ pin GND Arduino
  - IN1 នៃ relay ភ្ជាប់ទៅ pin 13 Arduino
- **ការតភ្ជាប់ Relay to lamp**

  NC នៃ Relay to pin (+)lamp និង Com នៃRelay to pin (-)lamp

1.4. ការប្រើ app សម្រាប់ Bluetooth ក្នុងទូរស័ព្ទ

1. តំឡើងកម្មវិធីឈ្មោះ "Bluetooth Control a Led " ដែលឈ្មោះBTControlLED.apk ដាក់ក្នុងទូរស័ព្ទដៃដោយទាញយកវានៅទីតាំង។ សូមសរសេរកូដខាងក្រោមដាក់ក្នុងប៉តArduino សិន និងធ្វើការ Paired ប្លូធូសនៅលើទូរសព្ទជាមុនសិនមុននឹងបើកកម្មវិធីនេះដំណើរការ
2. ចុចលើប៊ូតុង Connect និងបន្ទាប់មកជ្រើសរើសលេខអាសយដ្ឋាននៃប្លូធូសដោយមានការប្តូរពាក្យពី Not Connect ទៅ Connected នោះយើងអាចធ្វើការបញ្ជាបាន។

ចំណាំ: ក្នុងការធ្វើតេស្តយើងធ្វើការភ្ជាប់ភ្លើង 5v ពី Arduino ទៅកាន់ ៥ វ៉ុលនៃ Relay សិនបានហើយ ព្រោះនៅលើ Relay មានអំពូល LED មួយដែលអាចយើងដឹងថាអំពូល ON ឬ OFF ។ ដល់តេស្តចប់សព្វគ្រប់ទើបយើងធ្វើការភ្ជាប់ភ្លើង ២២០ វ៉ុលព្រោះភ្លើងនេះមានគ្រោះថ្នាក់បើយើងមិនប្រាស់លាស់ក្នុងការតេ និងដោយយើងធ្វើការតេស្តច្រើនដងវាអាចប៉ះផ្ទាល់ជាមួយនឹងភ្លើងនេះឬតំណពេលយើងតមិនបានល្អ។



1.5. កូដសម្រាប់បញ្ជា Bluetooth control lamp

```
#include <SoftwareSerial.h>
#define lampPin 13  //កំណត់ LedPin ភ្ជាប់ទៅកាន់ជើងលេខ ១៣
char state = 0;
void setup( ) {
  pinMode( lampPin, OUTPUT ); //កំណត់មានជើងledPin ជាទិសដៅចេញ
  digitalWrite( lampPin, LOW ); //កំណត់ឱ្យ ledPin បិត
  Serial.begin( 9600 );
}
void loop( ){
    // Checks whether data is comming from the serial port
    if( Serial.available( ) > 0 ){//បើទិន្នន័យត្រូវបានផ្ញើទៅផ្ទុកក្នុង Buffer មានតម្លៃ>0
        Serial.println( "LED: ON" ); //Displays LED ON Smart Phone
```

```
        state = Serial.read( );} // Reads the data from the serial port
    if ( state == '0' ) {
        digitalWrite( lampPin, LOW ); // Turn LED OFF
        Serial.println( "LED: ON" );// Send back, to the phone, the String "LED: ON"
        state = 0;}
    else if ( state == '1' ) {
        digitalWrite( lampPin, HIGH ); // Turn LED OFF
        Serial.println( "LED: OFF" );// Send back, to the phone, the String "LED: OFF"
        state = 0; }
}
```

1.6. ការតេសតគម្រោងនេះក្នុង Proteus

នៅពេលដំណើរការគម្រោងខាងលើបើសិនគ្មានឧបករណ៍ជាក់ស្តែងទេ យើងអាចដំឡើងវាក្នុងកម្មវិធី Proteus ជំនួសវិញដោយអនុវត្តតាមជំហានបន្តបន្ទាប់ខាងក្រោម:

1. បើកកម្មវិធី Proteus រួចធ្វើការដំឡើងឧបករណ៍និមួយៗដូចរូបខាងលើក្នុងកម្មវិធីនេះ
2. ធ្វើការបើក Serial Port ក្នុងកុំព្យូទ័រដោយចុចលើរូប នៅខាងស្តាំបំផុតនៃ **Taskbar**
3. ចុច Mouse ស្តាំលើរូប និងជ្រើសរើសពាក្យ **Go to Setting**
4. នៅកម្មវិធី **Settings** កំណត់ប៊ូធូសទៅ ON ដូចរូប
5. បន្ទាប់ចុចលើ **More Bluetooth Options** និងជ្រើសយក **COM Port**
6. ចុចលើប៊ូតុង Add និង OK បន្ទាប់មកបង្ហាញ COM3 ឬ COM4 តាមឧបករណ៍
7. នៅកម្មវិធី Arduino IDE ជ្រើស Serial Port ធ្វើការជ្រើសយក COM3 ឬ COM4 ដូចជំហានទី ៦ និងធ្វើការបំលែងកូដឱ្យទៅជា Hexa File
8. ធ្វើការតភ្ជាប់រវាងប៊ូធូសរវាងទូរសព្ទដៃជាមួយនឹង កុំព្យូទ័រដោយដំបូងត្រូវ Paired ប៊ូធូសលើទូរសព្ទសិន និងជ្រើសរើសឈ្មោះប៊ូធូសកុំព្យូទ័ររួច និងមានព័ត៌មានទៅកាន់កុំព្យូទ័រ
9. ចុងក្រោយ Run ឧបករណ៍នៅកម្មវិធី Proteus ដោយចុចលើរូប និងបើកវិធីនៅលើទូរសព្ទដោយជ្រើសរើសអាសយដ្ឋានប៊ូធូស និងធ្វើការបញ្ជូនវាទៅតាមលំដាប់លំដោយ

2. Simple Calculator

គឺម៉ាស៊ីនគណនាលេខធម្មតាដែលអាចប្រើសម្រាប់ធ្វើប្រមាណវិធី បូក, ដក, គុណ និង ចែកបាន។ ប្រើវាក្នុងវិស័យអប់រំដែលអាចជួយសម្រួលដល់សិស្សានុសិស្សយល់ពីការដំឡើង និងជាពិសេសដឹងពីការសរសេរគឺកូដវាដោយមានប្រើ Class, Object និង អនុគមន៍ផ្សេងៗ។

គ្រប់គម្រោងទាំងអស់ត្រូវប្រើប៉ត Arduino គឺជាបន្ទះសៀគ្វីអេឡិចត្រូនិចមួយ ត្រូវបានប្រើជាមួយកម្មវិធី Arduino សម្រាប់សរសេរកូដ និងធ្វើការ Upload ពីកូដនេះដែលប្រើនៅលើកុំព្យូទ័រទៅក្នុង Flash Memory របស់វា។ ចំពោះការប្រើប្រាស់កម្មវិធី Proteus ក៏យើងប្រើកម្មវិធីនេះដែរ។

Keypad គឺជាបន្ទះសៀគ្វីមួយដែលផ្គុំឡើងពីប៊ូតុងជាច្រើន។ ប្រើវាសម្រាប់ធ្វើបញ្ចូលតម្លៃលេខ និងអក្សរពី Keypad ទៅកាន់ប៉ត Arduino។ វាឧបករណ៍បញ្ចូលទិន្នន័យប្រើប្រាស់នៅក្នុងការធ្វើគម្រោងនេះ ហើយយើងក៏បានប្រើវារួចហើយនៅមេរៀនទី៤។ Keypad មាន:

- មានប៊ូតុងចុចចំនួន 16 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, *, #)

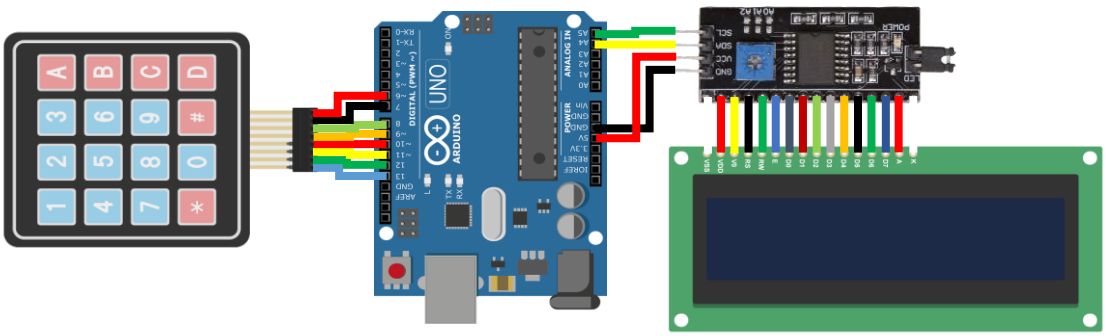
- មានជួរដេកចំនួន 4 (R1-R4) និងក្នុងកម្មវិធី Proteus មានអក្សរ A, B, C, D
- មានជួរឈរចំនួន 4 (C1-C4) និងក្នុងកម្មវិធី Proteus មានអក្សរ 1, 2, 3, 4

Arduino Board	Keypad
13 -> R1	9 -> C1
12 -> R2	8 -> C2
11 -> R3	7 -> C3
10 -> R4	6 -> C4

2.1. I2C LCD 1602 ជាអ្វី?

វាប្រើសម្រាប់ភ្ជាប់ពី LCD ទៅកាន់ប៉ត Arduino ដោយមាន 16 Pins សម្រាប់ភ្ជាប់ទៅកាន់ LCD និងមានតែ 4 Pins គឺ GND, VCC, SDA, SCL សម្រាប់ភ្ជាប់ទៅកាន់ Arduino។

2.2. របៀបតភ្ជាប់ LCD with I2C to Arduino



ឧទាហរណ៍ num1(20) + num2(30) = 50(Output)

2.3. Source Code

```
#include <LiquidCrystal.h> //បញ្ចូល Library របស់ LCD
#include <Keypad.h> //បញ្ចូល Library របស់ Keypad
LiquidCrystal lcd(0x3F, 16,2);// បង្កើត Object lcd និងកំណត់លេខអាសយដ្ឋាន និងទំហំ
const byte ROWS = 4; // តំណាងឲ្យ Rows=4 របស់ Keypad
const byte COLS = 4; // តំណាងឲ្យ Columns=4 របស់ Keypad
//អថេរ Keys ផ្ទុកអារេ string ប្រើសម្រាប់ប្រើប្រាស់លេខ និងសញ្ញានៅលើ Keypad
char keys [ROWS] [COLS] = {
  {'7', '8', '9', '/'},
  {'4', '5', '6', '*'},
  {'1', '2', '3', '-'},
  {'X', '0', '=', '+'}
};
//កំណត់អារេផ្ទុកសំនុំជាតុ 9, 8, 7, 6 ដែលជា Pin  Arduino ប៉តដើម្បីតភ្ជាប់ពីជួរដេកនៃ Keypad
byte rowPins[ROWS] = {9 ,8 ,7 ,6};
```

```
//កំណត់អារេផ្ទុកសំនុំជាតុ 13,12,11,10 ជា Pin  Arduino ប៉ុតដើម្បីតភ្ជាប់ពីជួរឈរនៃ Keypad
byte colPins[COLS] = {13, 12, 11, 10};
//បង្កើត myKeymap  Object សម្រាប់ធ្វើជាមួយនឹងអនុគមន៍ក្នុង Keypad ដើម្បីធ្វើការចុច Key  //ទៅ
តាមសំនុំជាតុនៅក្នុងវង់ក្រចក ដោយមានអនុគមន៍ makeKeymap(keys) ធ្វើឱ្យ keys        // អាច
ដំណើរការបាននៅលើ Keypad និងតាមជើង Pin Arduino ដែលមានជួរដេក ៤ និងជួរឈ //៤  និងមាន
ម៉ាទ្រីស 4 x 4
Keypad myKeypad = Keypad( makeKeymap(keys),rowPins,colPins, ROWS, COLS );
//variables declaration
boolean valOnePresent = false; // ប្រើជាលក្ខខណ្ឌប្រើចុច Key លើកទី ១
//for set when we press num1 or num2
String num1, num2; // ប្រកាសអថេរ num1 និង num2 ផ្ទុកលេខទី ១ និងទី ២
double ans; // ប្រកាសអថេរ ans សម្រាប់ផ្ទុកលទ្ធផល
char op // ប្រកាសអថេរ op សម្រាប់ធ្វើប្រមាណវិធីពេលចុច Key
void setup( ){
        lcd.begin(16,2);//ចាប់ផ្តើមបើក 16 Colums និង 2 Rowsទាំងអស់របស់ LCD
        lcd.setCursor(2,0); //បញ្ជាសម្រាប់កំណត់ទីតាំង Columnទី2  និង Row ទី 0
        lcd.print("RUPP DSC Robotic");//បង្ហាញឃ្លា RUPP DSC Robotic នៅលើ LCD
        delay(1500);
        lcd.clear( );}//Clear screen and positions the cursor in the upper-left corner.
void loop( ){
        //អថេរ key សម្រាប់ផ្ទុកតម្លៃ ពេលអនុគមន៍ myKeypad.getKeyចាប់តម្លៃ string
        char key = myKeypad.getKey( ); //press :20=3230h
        // input num1 and num2
         if (key != NO_KEY && key=='1'||key=='2'|| key=='3'||key=='4'||
        key=='5'||key=='6'|| key=='7'||key=='8'|| key=='9'||key=='0' ){
                if (valOnePresent ==false){//អនុវត្តពេលអ្នកប្រើ ចុចតម្លៃ string លើកដំបូង
                        num1 = num1 + key;//ធ្វើប្រមាណវិធីបូក string បន្តបន្ទាប់
                        int numLength = num1.length( );//រាប់ប្រវែង string ក្រោយពេលបាន//
                        ចុច    ហើយផ្ទុកក្នុងអថេរ numLength
                        lcd.setCursor(15 - numLength, 0); //lcd.setCursor(cols,rows)
                        //កំណត់ទីតាំងបង្ហាញទិន្នន័យនៅលើអក្រង់LCD ពីខាងស្តាំទៅខាងឆ្វេង
                        lcd.print(num1);}//បង្ហាញតម្លៃ String នៅលើ LCD
                else {
                //អនុវត្តពេលចុចតម្លៃ string ជាលើកទី 2 ក្រោយពេលចុច Operator +, -,  *, /
                num2 = num2 + key;
                int numLength = num2.length( );
```

```
                lcd.setCursor( 15 - numLength, 1 );
                lcd.print( num2 );}
  } else if ( valOnePresent == false && key != NO_KEY && ( key == '/' || key == '*' || key
== '-' || key == '+' ) ){
//ការអនុវត្តពេល អ្នកប្រើ ចុច Operator +,  - ,   *,  /
                if ( valOnePresent == false ){
                //ធ្វើប្រមាណវិធីម្តងទៀតនៅក្រោយពេលទិន្នន័យចាស់ ត្រូវបានលុបចោល។
                        valOnePresent = true;
                        //កំណត់ទីតាំង Cursor ដើម្បីបង្ហាញ Data
                        lcd.setCursor( 15,0 ); //operator on right corner
                        lcd.print( key );} //បង្ហាញសញ្ញាគឺ `+´, `-´, `*´, និង `/´
}
  else if (  key != NO_KEY && key == '=' ){
        //Key មានតម្លៃ និង key ស្មើនិង char :  ` =`
         if ( op == '+' ){
                ans = num1.toInt( ) + num2.toInt}
                else if ( op == '-' ){
                ans = num1.toInt( ) - num2.toInt( );}
         else if ( op == '*' ){
                ans = num1.toInt( )  * num2.toInt( );}
         else if ( op == '/' ){
                ans = num1.toInt( ) / num2.toInt( );}
         lcd.clear( );//clear data ចាស់
         lcd.setCursor( 15,0 );//កំណត់ទីតាំងរបស់ data ដែលត្រូវបង្ហាញ
         //បើក autoscoll នៅលើ LCD ដើម្បីscroll Result ក្រោយពេលគណនា ដោយautomatic
lcd.autoscroll( );
         lcd.print( ans );//បង្ហាញ ចម្លើយ ក្រោយពេលគណនារួច
lcd.noAutoscroll( );     }//បិទ autoscoll
else if ( key != NO_KEY && key == 'X' ){
         valOnePresent = false;
         num1="";
         num2="";
         op=' ';
         ans=0;
         lcd.clear( );}
}
```

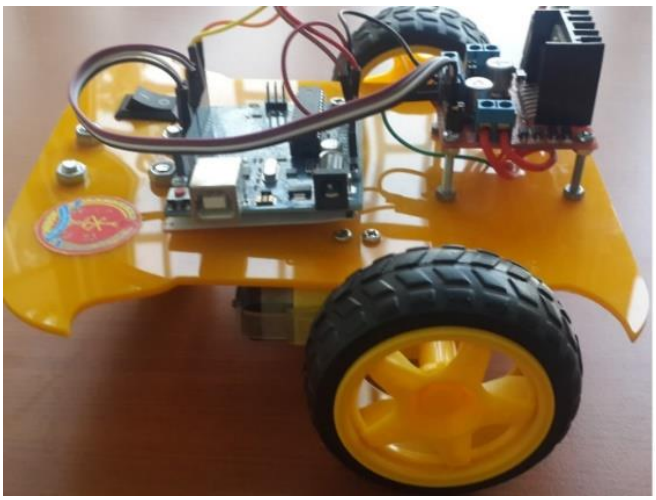
ចំណាំ: ករណីប្រើ Proteus ដោយប្តូរតម្លៃទៅតាមជើងPIN Arduino, PIN LCD និងលេខនៅ Keypad បានកូដយើងដំណើរការ។

3. ការបង្កើត Radio Controller Car (RC Car)

វាគឺជារ៉ូបូតតូចមួយដែលជាទូទៅត្រូវបានភ្ជាប់ទៅនឹងឧបករណ៍អេឡិចផ្សេងដទៃទៀត ឬ បណ្តាញតាមរយៈវិធីឥតខ្សែផ្សេងៗគ្នាដូចជា Bluetooth, Zigbee, NFC, Wi-Fi, LiFi, 3G,NRF និងឧបករណ៍ជាច្រើនទៀត។ វាជាមូលដ្ឋានគ្រឹះសម្រាប់ធ្វើការបង្កើតរ៉ូបូតផ្សេងៗដូចជារ៉ូបូតរត់លើផ្លូវ, ជៀសឧបសគ្គ, ធ្វើការប្រកួតប្រជែង និងរ៉ូបូតប្រើប្រាស់លើវិស័យផ្សេងៗដូចវិស័យកសិកម្ម, យោធា, ឧស្សាហកម្ម ជាដើម។ ចំពោះរ៉ូបូតនៃគម្រោងនេះយើងបង្កើតវាឡើងដើម្បីជាគ្រឹះដំបូងសម្រាប់ជាចំណេះដឹង ក៏ដូចជាបទពិសោធន៍ថ្មីៗ ដើម្បីបង្កើតរ៉ូបូតដែលទាក់ទងនឹងការរចនាផ្នែករឹង, ផ្នែកទន់ និងកម្មវិធីប្រើលើទូរសព្ទដៃ និងឱ្យវាអាចដំណើរការផ្សេងៗគ្នាតាមការបញ្ជា។

3.1. តើអ្វីទៅជា RC CAR?

RC CAR គឺជាប្រភេទរ៉ូបូតមួយដែលយើងអាច គ្រប់គ្រងវាបានពីចំងាយ ដោយធ្វើការបញ្ជាវាដោយប្រើ smart phone ឬ controller (ឧទាហរណ៍បញ្ជា តាម Bluetooth WIFI រលកវិទ្យុសកម្ម)



3.2. គ្រឿងសម្រាប់ធ្វើ RC CAR

ឧបករណ៍សម្រាប់ធ្វើដំឡើង RC CAR រួមមានដូចជា Resin Panels (បន្ទះដំរសម្រាប់ធ្វើតួឡាន), Arduino Board, Jumper wire(ខ្សែភ្លើង), Battery DC Gear Motor 12V, Wheel, Driver L298N, HC-06, Cell holder, កុងតាក់ និង កូនកង់ចង្កូតរ៉ូបូត។

3.2.1. Resin Panels គឺជាបន្ទះដំរសម្រាប់ធ្វើតួរបស់រ៉ូបូតដែលធ្វើពីដំរ។ ហើយប្រើសម្រាប់ចាប់ម៉ូទ័រនឹងសម្រាប់ជាក់គ្រឿងផ្សេងៗទៀត។

3.2.2. Arduino UNO R3 គឺជាខួរក្បាលរ៉ូបូត ដែលអាចឱ្យយើងសរសេរកូដបញ្ចូលទៅតាមអ្វីដែលយើងចង់ឱ្យវាធ្វើ។ វាមានលក្ខណៈជាបន្ទះសៀគ្វីអេឡិចត្រូនិចដែលមានជើងសម្រាប់បញ្ចូលកូដ និងជើងអេឡិចត្រូនិចដែលអាចដើរតួនាទីបានទាំងការបញ្ជាចេញនិងអានចូលមកវិញ។

3.2.3. Motor Driver L298N គឺជា driver ដែលអាចឱ្យយើងគ្រប់គ្រងល្បឿន និង ទិសដៅសម្រាប់ម៉ូទ័រទាំងពីរ។ នៅលើម៉ូទ័រdriver នេះគឺមាន Pin OUT1 និង OUT2 សម្រាប់ភ្ជាប់ទៅ MotorA និង Pin OUT3 និង OUT4 សម្រាប់ភ្ជាប់ទៅ MotorB ។ Power Pin មានពីរគឺ ១២វុល និង GND ប្រើសម្រាប់បញ្ចូលភ្លើងចាប់ពី ៦ វុល ទៅ ១២ វុលពី battery និងមានភ្លើងចេញ ៥ វុលមួយ។ ចំណែក ៦ Pin ភ្ជាប់ទៅ Arduino Board មាន ENA, IN1, IN2, IN3, IN4 និង ENA។ ដោយ ENA

Pin ឱ្យ MotorA ដំណើរការ និង IN1, IN2 ទទួលតម្លៃ 0 ឬ 1 ដើម្បីឱ្យវិលទៅមុខឬថយក្រោយ។ ចំណែក ENB Pinឱ្យ MotorB ដំណើរការ និង IN3, IN24 ទទួលតម្លៃ 0 ឬ 1 ដើម្បី ឱ្យវិលទៅមុខ ឬថយក្រោយ

3.2.4. Jumper គឺជាខ្សែភ្លើង សម្រាប់តភ្ជាប់ពីរឧបករណ៍មួយទៅឧបករណ៍មួយទៀត។ DC Gear Motor 12V គឺជាប្រភេទ Motor ដែរមានតង់ស្យុងចន្លោះពី 9V ទៅ 12V សម្រាប់បង្វិលកង់ឱ្យRobot ទៅមុខ ទៅក្រោយ ទៅឆ្វេង និង ទៅស្តាំ។

3.2.5. Battery(ថ្ម) គឺជាប្រភពថាមពលសម្រាប់ផ្តល់ទៅឱ្យ DC Motor 12V ដែលមានប៉ូលបូក និងប៉ូលដក។ វាផ្តល់តង់ស្យុងពី 9V-12V ទៅឱ្យ Motor driver L298N និង Arduino Board។

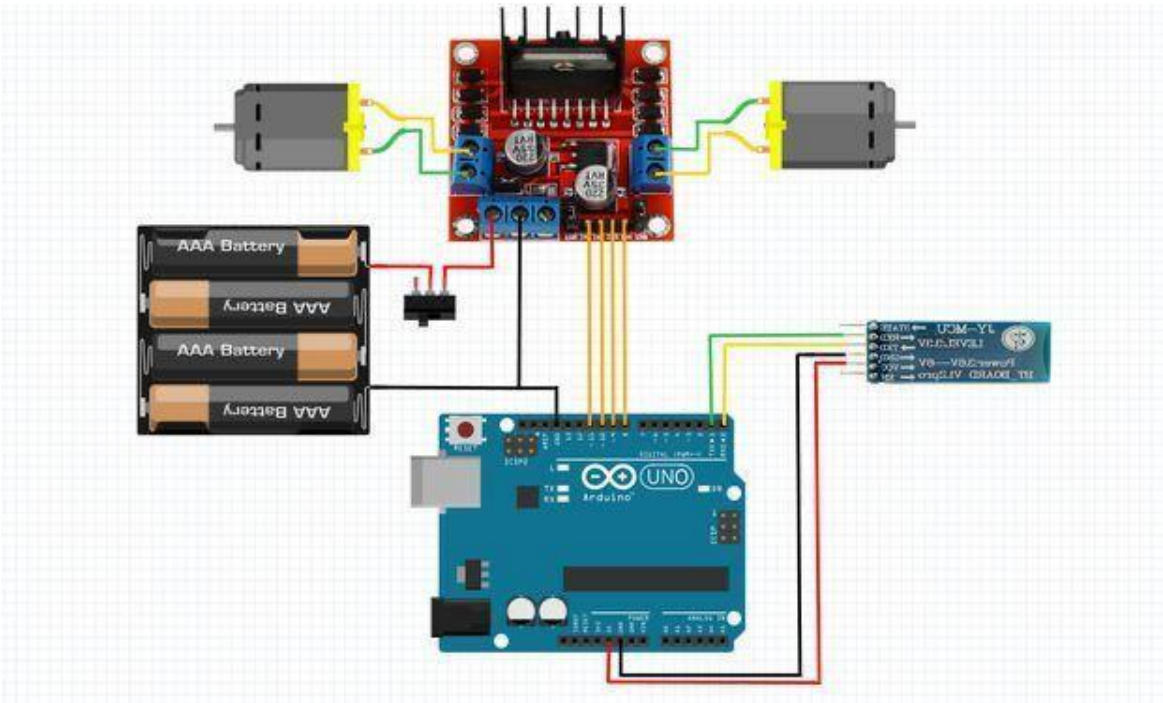
3.2.6. Bluetooth HC-06 ជាឧបករណ៍ដែលទទួលទិន្នន័យពី smartphone តាមរយៈ Bluetooth ហើយវាបញ្ជូនទៅកាន់ Arduino board ហើយ Arduino បញ្ជូន Data ទៅកាន់ Motor driver L298N ដើម្បីធ្វើការបញ្ជាទៅលើ Motors ទាំងពីរ។

3.2.7. កុងតាក់ គឺជាឧបករណ៍អគ្គិសនីសម្រាប់បិទនិងបើកភ្លើង។ យើងប្រើសម្រាប់បើកឱ្យភ្លើងចូលទៅកាន់ Motor driver និង Arduino។ នៅក្នុង RC car យើងអាចប្រើកុងតាក់ណាមួយក៏បានឱ្យតែវាសម្របទៅនឹងឡានរបស់យើង(កុងតាក់តូចល្មម)។

3.2.8. Cell holder គឺជាឧបករណ៍មួយដែលអាចដាក់ថ្មបាន។ ជាទូទៅគេច្រើនប្រើ cell holder ដែលប្រើបានថ្មពីរគ្រាប់។ ពីព្រោះថ្មពីរគ្រាប់គឺយើងអាចប្រើថាមពលវាបានសមាមាត្រទៅនឹងតម្រូវការរបស់ យើងដែលត្រូវការថ្មចន្លោះពី 9vទៅ12v។

3.2.9. កូនកង់ ចង្កូតឡានគឺជាកូនកង់តូចមួយដែលគេប្រើវាសម្រាប់ទប់ឡាន(ព្រោះយើងប្រើ Motor តែ2)ហើយវាក៏អាចបត់បែនបានទៅតាមការបញ្ជារបស់ Motor and Wheel ដែលនៅខាងក្រោយ នៃតួ RC Car។

3.3. របៀបតភ្ជាប់ Hardware



3.4. ការប្រើ app សម្រាប់បញ្ជាឡាន ក្នុងទូរស័ព្ទ

1. បញ្ជាទៅមុខប្រើតម្លៃលេខ
2. បញ្ជាទៅក្រោយប្រើតម្លៃលេខ
3. បញ្ជាទៅខាងប្រើតម្លៃលេខ
4. បញ្ជាទៅខាងស្តាំ ប្រើតម្លៃលេខ
5. បញ្ជាទៅទិសចន្លោះសញ្ញាខាងឆ្វេង និងមុខប្រើតម្លៃលេខ
6. បញ្ជាទៅទិសចន្លោះសញ្ញាខាងឆ្វេង និងក្រោមប្រើតម្លៃលេខ
7. បញ្ជាទៅទិសចន្លោះសញ្ញាខាងស្តាំ និងមុខប្រើតម្លៃលេខ
8. បញ្ជាទៅទិសចន្លោះសញ្ញាខាងស្តាំ និងក្រោមប្រើតម្លៃលេខ

3.5. កូដសម្រាប់បញ្ជា RC Car (Coding)

```
int in1 =4; //ប្រកាសអថេរតំណាងឱ្យ in1 នៃ Driver ភ្ជាប់ទៅ Pin លេខ ៤ Arduino
int in2 =5; //ប្រកាសអថេរតំណាងឱ្យ in2 ភ្ជាប់ទៅ Pin លេខ ៥ Arduino
int in3 =6; //ប្រកាសអថេរតំណាងឱ្យ in3 ភ្ជាប់ទៅ Pin លេខ ៦ Arduino
int in4 =7; //ប្រកាសអថេរតំណាងឱ្យ in4 ភ្ជាប់ទៅ Pin លេខ ៧ Arduino
int motorS1=9;  //ប្រកាសអថេរ motorS1( ENA ) ភ្ជាប់ទៅ Pin លេខ 9 Arduino
//សម្រាប់បញ្ជារល្បឿននៃ Motor A
int motorS2=10; //ប្រកាសតម្លៃ Pin motorS2( ENB )ភ្ជាប់ទៅ Pin លេខ 10 Arduino
//សម្រាប់បញ្ជារល្បឿននៃ Motor B
char Value; //ចាប់យក data ពីអនុគមន៍ Serial.Read
void setup( ){
      pinMode( in1, OUTPUT ); //កំណត់ Pin in1 ជាទិសដៅចេញ
      pinMode( in2, OUTPUT ); //កំណត់Pin in2 ជាទិសដៅចេញ
      pinMode( in3, OUTPUT ); //កំណត់Pin in3 ជាទិសដៅចេញ
      pinMode( in4, OUTPUT ); //កំណត់Pin in4 ជាទិសដៅចេញ
      Serial.begin( 9600 ); //បើក Serial Port ដែលមានល្បឿន 9600 ក្នុងមួយវិនាទី
      pinMode( motorS1, OUTPUT );
      pinMode( motorS2, OUTPUT ); }
```

```
void loop( ){
        if( Serial.available( ) > 0 ){
           Value = Serial.read( );
            char Speed ;
            if( Value >10 ){
               Speed=Value;}
        if( Value == 'O' ){
            digitalWrite( ledPin,1 );}
        if( Value == 'F' ){
            digitalWrite( ledPin,0 );}
       if( Value == 1 ) {
       // motor A ដែលមាន in1, in2 ត្រូវបាន
       //វិលទៅមុខ
            forward( );
            analogWrite( motorS1,Speed );
             analogWrite( motorS2,Speed ); }
        // motor B ដែលមាន in3, in4 ត្រូវបាន
       //វិលទៅមុខ
       if( Value == 2 ) { //Back
            back( );
            analogWrite( motorS1,Speed );
            analogWrite( motorS2,Speed );}
       if( Value == 3 )//Left{
            digitalWrite( in1,1 );
            digitalWrite( in2,0 );
            digitalWrite( in3,0 );
            digitalWrite( in4,1 );
            digitalWrite( ledPin,1 );
            analogWrite( motorS1,Speed );
            analogWrite( motorS2,Speed );}
       if( Value == 4 ) {//Right
            digitalWrite( in1,0 );
            digitalWrite( in2,1 );
            digitalWrite( in3,1 );
            digitalWrite( in4,0 );
            digitalWrite( ledPin,1 );
    digitalWrite( in4,0 );
    digitalWrite( ledPin,1 );}
if( Value == 6 ) {//forward_left
     forward( );
     analogWrite( motorS1,Speed-
     ceil( Speed/2 ) );
     analogWrite( motorS2,Speed );}
if( Value == 7 ) { //forward_right

      forward( );
      analogWrite( motorS1,Speed );
      analogWrite( motorS2,Speed-
     ceil( Speed/2 ) );
    }
if( Value == 8 ) { //back_left
      back( );
     analogWrite( motorS1,Speed-
     ceil( Speed/2 ) );
      analogWrite( motorS2,Speed );}
if( Value == 9 ) { //back_right
      back( );
      analogWrite( motorS1,Speed );
       analogWrite( motorS2,Speed-
       ceil( Speed/2 ) );}
}//End void loop
//Function
void forward( )
{
  digitalWrite( in1,1 );
  digitalWrite( in2,0 );
  digitalWrite( in3,1 );
  digitalWrite( in4,0 );
  digitalWrite( ledPin,1 );
}
void back( )
{
```

```
        analogWrite(motorS1,Speed);
        analogWrite(motorS2,Speed);}
    if(Value == 5) { //Stop
        digitalWrite(in1,0);
        digitalWrite(in2,0);
        digitalWrite(in3,0);
        digitalWrite(in1,0);
        digitalWrite(in2,1);
        digitalWrite(in3,0);
        digitalWrite(in4,1);
        digitalWrite(ledPin,1);
    }
```

4. ប្រព័ន្ធស្រោចទឹក និងគ្រប់គ្រងធារាសាស្ត្រស្វ័យប្រវត្តិ

គោលបំណងនៃការបង្កើតសម្រាប់ធ្វើការយល់ដឹងអំពីប្រព័ន្ធ AUTOMATION និងអាចធ្វើការស្រោចដំណាំនៅក្នុងផ្ទះបាន និងជាមូលដ្ឋានគ្រឹះសម្រាប់បង្កើតឧបករណ៍ដទៃផ្សេងទៀត ដើម្បីជួយដល់ការងារកសិករ។ វាជា AUTOMATION សម្រាប់ជួយដល់៖

- ផ្តល់ភាពងាយស្រួលដល់ការដាំដំណាំ
- កាត់បន្ថយការប្រើប្រាស់កម្លាំង
- កាត់បន្ថយការចំណាយផ្សេងៗ
- ជួយអភិវឌ្ឍន៍ដល់វិស័យកសិកម្មផងដែរ
- ជាSimple ក្នុងគម្រោងAutomation

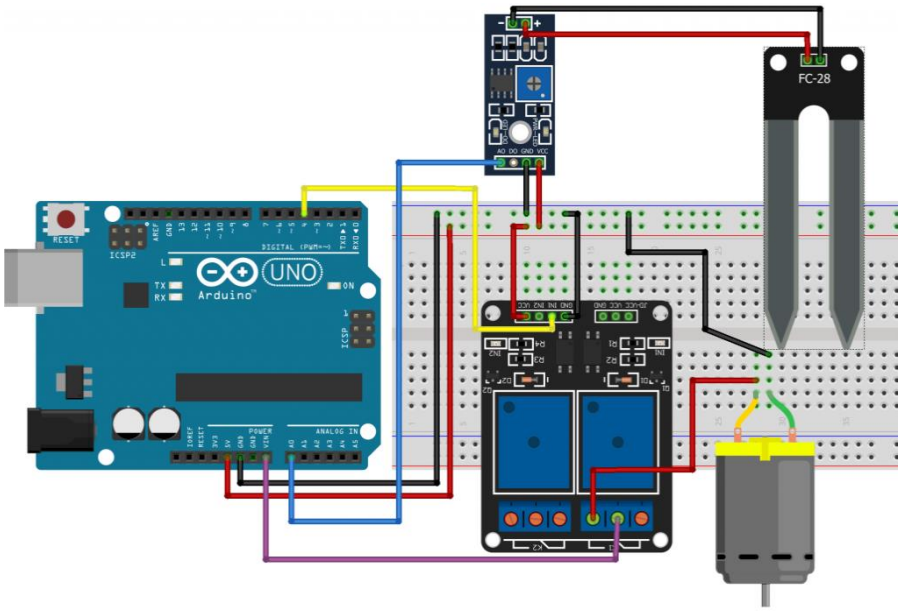
4.1. ឧបករណ៍ការដំឡើងរួមមាន:

- Arduino Board និង Relay
- **DC Motor Pump** ជាប្រភេទម៉ូទ័រមួយប្រើសម្រាប់បូម និងបញ្ចេញទឹក។

- **SOIL MOISTURE SENSOR** ជាឧបករណ៍ចាប់យកសំណើមដី និងត្រូវបានប្រើដើម្បីវាស់ កំរិតបរិមាណទឹកនៅក្នុងដី។ ជាពិសេសប្រើវាសម្រាប់អនុវត្តការពិសោធនៅក្នុងវគ្គសិក្សាដូចជាវិទ្យាសាស្ត្រដី, វិទ្យាសាស្ត្រកសិកម្ម, វិទ្យាសាស្ត្របរិស្ថាន, សាកវប្បកម្មរុក្ខសាស្ត្រ, និងជីវវិទ្យា។ វាជាឧបករណ៍បញ្ចូលតម្លៃមួយទៅឱ្យ ARDUINO BOARD ដែលមានប្រើថាមពល៖ ៣,៣ វុលឬ ៥ វុល និងសញ្ញាវ៉ុលលទ្ធផល: 0-4.2 វុល គិតជាតម្លៃ ANOLOG មានពី 0 ដល់ ១០២៤
- **Adapter 12 volt** ប្រើសម្រាប់តភ្ជាប់ភ្លើង ២២0 វុល ជាមួយនឹង Power Jack របស់ Arduino Uno ដើម្បីផ្តល់ភ្លើងចេញ ១២ វុលប្រើ Pin Vin ទៅឱ្យ DC Motor Pump និងភ្លើង ៥ វុលទៅ Relay និង Soil Moisture Senstureឯ

4.2. ការតភ្ជាប់

- ដំបូងយើងត្រូវយកខ្សែ JUMPER ដោតជើង GND របស់ ARDUINO BOARD ទៅកាន់ជើង GND របស់ RELAY, SOIL MOISTURE SENSOR ហើយនឹងប៉ូលដកនៃ DC MOTOR PUMP
- បន្ទាប់មកដោតខ្សែ JUMPER ពីជើង ៥V របស់ ARDUINO UNO ទៅកាន់ជើង VCC របស់ RELAY, SOIL MOISTURE SENSOR និងជើង NO របស់ RELAY



- បន្ទាប់មកទៀតយើងត្រូវដោតខ្សែ JUMPER ពីជើង PIN A5 របស់ ARDUINO UNO ទៅកាន់ជើង AOUT របស់ SOIL MOISTURE SENSOR
- ហើយយើងត្រូវដោតខ្សែ JUMPER ពីជើង PIN 0 ទៅកាន់ជើង IN របស់ RELAY
- ចុងក្រោយដោតខ្សែពីជើង COM របស់ RELAY ទៅកាន់ប៉ូលបូកនៃDC MOTOR PUMP។

4.3. CODING

```
int c;
    void setup( ) {
            pinMode( A0,INPUT
            pinMode( 0,OUTPUT );
            Serial.begin( 9600 );
    }
```

```
void loop( ) {
        c=analogRead( A5 );
        if( c>600 ){
            digitalWrite( 5, 1 );}
        else{
            digitalWrite( 5, 0 );}}
```

កំណត់សំគាល់: ចំពោះការធ្វើប្រព័ន្ធជារាសាស្ត្រប្រហាក់ប្រហែលគម្រោងខាងលើដែរ តែត្រូវថែម LCD និង Ultrasonic Sensor រួមនឹងឧបករណ៍ផ្សេងៗមួយចំនួនទៀត។

5. ឌីជីថល CLOCK

វាគឺជានាឡិកាឌីជីថល 24 ម៉ោងដែលមានមូលដ្ឋានលើប័ត arduino។ វាត្រូវបានបំពាក់ដោយថាមពលថ្មខាងក្រៅដោយប្រើ DS3231 ឬ DS1703 ជាមួយនឹងឧបករណ៍បញ្ចេញព័ត៌មានដូចជា Time1637, LCD និង DOT Matrix ជាដើម។ ហើយការកំណត់មានប៊ូតុងបី (ពីឆ្វេងទៅស្តាំ) សម្រាប់ការកំណត់ម៉ោង នាទី និង វិនាទី ទៅតាមពេលវេលាត្រូវបានកំណត់ជាក់លាក់មួយ។ ការបង្កើតវាឡើងដើម្បី:

- ផ្តល់ភាពងាយស្រួលក្នុងការមើលម៉ោងដែលអាចប្រើប្រាស់ដូចនាឡិកាដែលគេបានបង្កើតដែរ
- ចំណេញពេលវេលា និងមានភាពងាយស្រួលក្នុងការកំណត់ពេលវេលា
- ជាពិសេសយល់ដឹងពីការប្រើប្រាស់ផ្នែករឹង និងផ្នែកទន់

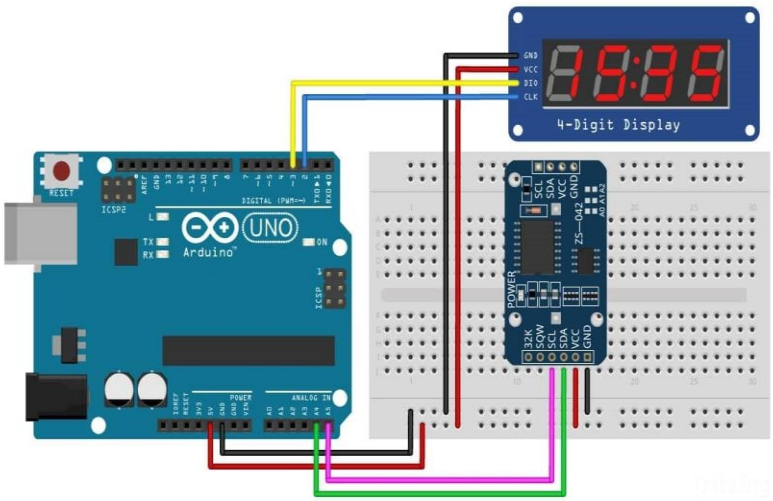
5.1. ឧបករណ៍ការដំឡើងរួមមាន

- Arduino Board, 3 Button

- TRC DS3231 គឺជាឧបករណ៍ប្រើសម្រាប់រក្សាទុកនូវម៉ោង, ថ្ងៃ ខែ ឆ្នាំ, និងសីតុណ្ហភាព។ វាអាចកំណត់ទៅតាមម៉ោងទូទៅតាមទីតាំងដែលវាកំពុងស្ថិតនៅ និងចងចាំម៉ោងបានល្អបើទោះបីជាថាមពលរបស់ឧបករណ៍ត្រូវបានកាត់ផ្តាច់។ វាមានថ្មមួយដែលមានថាមពលធ្វើឱ្យម៉ូឌុល ដំណើរការបានល្អទៅតាមពេលវេលា។

- Time1637: គឺជាឧបករណ៍អាចប្រើសម្រាប់បង្ហាញម៉ោង នាទី និងវិនាទី។

5.2. ការប្រើ DS3231 ជាមួយនឹង TIME1637

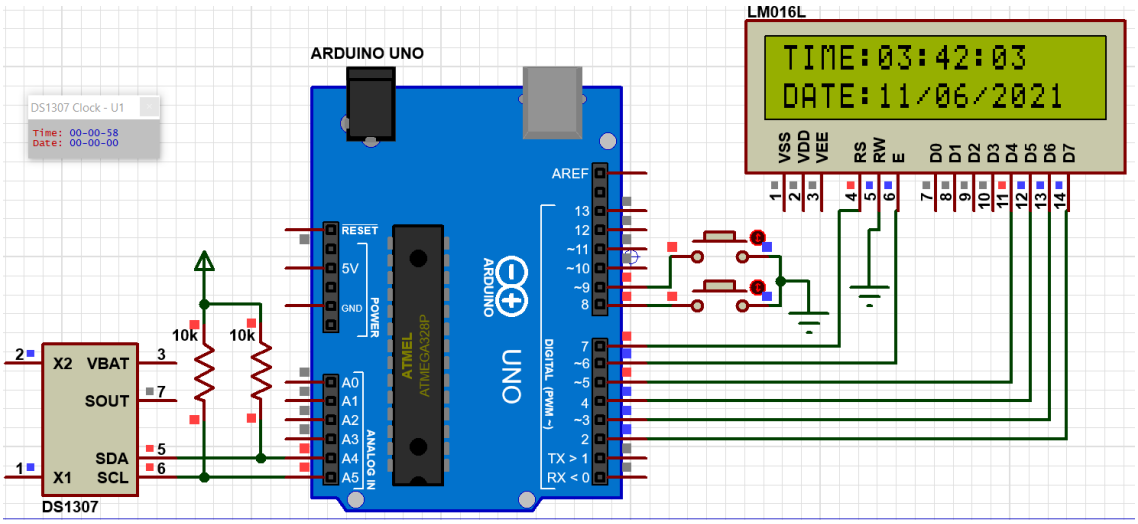


5.3. CODING ការប្រើ DS3231 ជាមួយនឹង TIME1637

```
//Project name : Digital Clock On TM1632
//Prepared by Ouk Polyvann
##include <RTClib.h> //ការប្រកាស header file របស់RTC DS3231
#include <TM1637Display.h> //ការប្រកាស header file របស់ TM1637
#define CLK 9// ប្រកាសនិងផ្តល់តម្លៃដល់ pin CLK = 9
#define DIO 8// ប្រកាសនិងផ្តល់តម្លៃដល់ pin DIO = 8
RTC_DS3231 rtc;// បង្កើត object ឈ្មោះ rtc
TM1637Display display = TM1637Display( CLK , DIO );//ប្រាប់ទៅtm1637 ឲ្យស្គាល់ជើង pin CLK
និង DIO
```

```
const unit8_t blank[] = {0x00, 0x00, 0x00, 0x00}; //ប្រកាស លេខ Address 0x00 7segment
ឲ្យtm1637
void setup( ) {
   // put your setup code here, to run once:
   Serial.begin( 9600 ); // បើក serial port កំណត់តម្លៃឈ្មឿន 9600 ទៅឲ្យ Serial port
   delay( 3000 ); // រងចាំ 3000( ms )  = 3 វិនាទី
   if( !rtc.begin( ) ) {//ត្រួតពិនិត្យថា rtc ត្រូវបានភ្ជាប់ហើយឬនូវ
      Serial.println( "Couldn't fine RTC" ); //បង្ហាញឃ្លា couldn't fine RTC លើសៀវៀល
      while( 1 ); }// loop ពិតជានិច្ច
   if( rtc.lostPower( ) ){ /* ត្រួតពិនិត្យថា rtc បាត់បង់ថាមពលឬអត់បើបាត់បង់ set សាជាថ្មី
     Serial.println( "RTC lost power, lets set the time!" ); //បង្ហាញឃ្លាលើ serial Monitor
     rtc.adjust( DateTime( 2021,1,21,3,0,0 ) );}//ផ្តល់នូវតម្លៃកាលបរិច្ឆេទនិងម៉ោងទៅឲ្យ RTC
  display.setBrightness( 0x0a ); // កំណត់ការបង្ហាញពន្លឺដល់ 4led 7segment
  display.setSegments( blank ); }// ប្តូរតម្លៃបង្ហាញទៅឲ្យ 7segment
}
void loop( ) {
   // put your main code here, to run repeatedly:
   DateTime now = rtc.now( ); //ការផ្តល់តម្លៃនៃពេលវេលាបច្ចុប្បន្នទៅឲ្យ Function DateTime
    int displaytime = ( now.hour( )  * 100 )+ now.minute( );//បង្កើតអថេរdisplaytimeសម្រាប់
   //ផ្ទុកតម្លៃរួបមន្តរបស់ពេលវេលា

   Serial.println( displaytime ); //បង្ហាញតម្លៃ  displaytime នៅលើ Serial Monitor
   int a = now.second( )%2; //បង្កើតអថេរ a សម្រាប់ផ្ទុកតម្លៃឲ្យ Blinking ដែលនៅកណ្តាល     //
   ម៉ោងនៃ 4led digital
  if( a==0 ) /* ប្រសិនបើតម្លៃរបស់ a ឬ Blinking =0 */ {
      display.setColon( false ); }// Blinking Colon មិនបង្ហាញទេ
  else /* ផ្ទុយមកវិញ */ {
      display.setColon( true ); } // Blinking Colon បង្ហាញលើ 4led digital of 7segment
   display.showNumberDec( displaytime, true );}//បង្ហាញម៉ោងដែលដំណើរការពេលបច្ចុប្បន្ន
}
```

5.4. ការដំឡើង DS3231 ជាមួយនឹង LCD សូមមើលរូបខាងក្រោម៖



5.5. កម្មវិធីដំណើរការ DS3231 ជាមួយនឹង LCD សូមមើលរូបខាងក្រោម៖

```
// Real time clock and calendar with set buttons using DS1307 and Arduino
// include LCD library code
#include <LiquidCrystal.h>
// include Wire library code (needed for I2C protocol devices)
#include <Wire.h>
// LCD module connections (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
void setup( ) {
  pinMode(8, INPUT_PULLUP); // button1 is connected to pin 8
  pinMode(9, INPUT_PULLUP); button2 is connected to pin 9
  // set up the LCD's number of columns and rows
  lcd.begin(16, 2);
  Wire.begin( );// Join i2c bus
}
char Time[]     = "TIME:  :  :  ";
char Calendar[] = "DATE:  /  /20  ";
byte i, second, minute, hour, date, month, year;
void DS1307_display( ){
  // Convert BCD to decimal
  second = (second >> 4)  * 10 + (second & 0x0F);
  minute = (minute >> 4)  * 10 + (minute & 0x0F);
  hour   = (hour >> 4)    * 10 + (hour & 0x0F);
  date   = (date >> 4)    * 10 + (date & 0x0F);
  month  = (month >> 4)   * 10 + (month & 0x0F);
```

```
  year   = ( year >> 4 )    * 10 + ( year & 0x0F );
  // End conversion
  Time[12]     = second % 10 + 48;
  Time[11]     = second / 10 + 48;
  Time[9]      = minute % 10 + 48;
  Time[8]      = minute / 10 + 48;
  Time[6]      = hour   % 10 + 48;
  Time[5]      = hour   / 10 + 48;
  Calendar[14] = year   % 10 + 48;
  Calendar[13] = year   / 10 + 48;
  Calendar[9]  = month  % 10 + 48;
  Calendar[8]  = month  / 10 + 48;
  Calendar[6]  = date   % 10 + 48;
  Calendar[5]  = date   / 10 + 48;
  lcd.setCursor( 0, 0 );
  lcd.print( Time );        // Display time
  lcd.setCursor( 0, 1 );
  lcd.print( Calendar );    // Display calendar
}
void blink_parameter( ){
  byte j = 0;
  while( j < 10 && digitalRead( 8 ) && digitalRead( 9 ) ){
    j++;
    delay( 25 );
  }
}
byte edit( byte x, byte y, byte parameter ){
  char text[3];
  while( !digitalRead( 8 ) );    // Wait until button ( pin #8 ) released
  while( true ){
    while( !digitalRead( 9 ) ){  // If button ( pin #9 ) is pressed
      parameter++;
      if( i == 0 && parameter > 23 )  // If hours > 23 ==> hours = 0
        parameter = 0;
      if( i == 1 && parameter > 59 )  // If minutes > 59 ==> minutes = 0
        parameter = 0;
```

```
      if( i == 2 && parameter > 31 )  // If date > 31 ==> date = 1
        parameter = 1;
      if( i == 3 && parameter > 12 )  // If month > 12 ==> month = 1
        parameter = 1;
      if( i == 4 && parameter > 99 )  // If year > 99 ==> year = 0
        parameter = 0;
      sprintf( text,"%02u", parameter );
      lcd.setCursor( x, y );
      lcd.print( text );
      delay( 200 );  // Wait 200ms
    }
    lcd.setCursor( x, y );
    lcd.print( "  " );             // Display two spaces
    blink_parameter( );
    sprintf( text,"%02u", parameter );
    lcd.setCursor( x, y );
    lcd.print( text );
    blink_parameter( );
    if( !digitalRead( 8 ) ){  // If button ( pin #8 ) is pressed
      i++;      // Increament 'i' for the next parameter
      return parameter;                       // Return parameter value and exit
    }
  }
}
void loop( ) {
  if( !digitalRead( 8 ) ){  // If button ( pin #8 ) is pressed
      i = 0;
      hour   = edit( 5, 0, hour );
      minute = edit( 8, 0, minute );
      date   = edit( 5, 1, date );
      month  = edit( 8, 1, month );
      year   = edit( 13, 1, year );
      // Convert decimal to BCD
      minute = ( ( minute / 10 ) << 4 ) + ( minute % 10 );
      hour = ( ( hour / 10 ) << 4 ) + ( hour % 10 );
      date = ( ( date / 10 ) << 4 ) + ( date % 10 );
```

```
    month = ((month / 10) << 4) + (month % 10);
    year = ((year / 10) << 4) + (year % 10);
    // End conversion
    // Write data to DS1307 RTC
    Wire.beginTransmission(0x68);  // Start I2C protocol with DS1307 address
    Wire.write(0);        // Send register address
    Wire.write(0);        // Reset sesonds and start oscillator
    Wire.write(minute);  // Write minute
    Wire.write(hour);      // Write hour
    Wire.write(1);         // Write day (not used)
    Wire.write(date);       // Write date
    Wire.write(month);      // Write month
    Wire.write(year);      // Write year
    Wire.endTransmission( );   // Stop transmission and release the I2C bus
    delay(200);     // Wait 200ms
  }
  Wire.beginTransmission(0x68);  // Start I2C protocol with DS1307 address
  Wire.write(0);     // Send register address
  Wire.endTransmission(false); // I2C restart
  Wire.requestFrom(0x68, 7);  // Request 7 bytes from DS1307 and release I2C bus    //at
end of reading
  second = Wire.read( );  // Read seconds from register 0
  minute = Wire.read( );   // Read minuts from register 1
  hour   = Wire.read( );     // Read hour from register 2
  Wire.read( );       // Read day from register 3 (not used)
  date   = Wire.read( );// Read date from register 4
  month  = Wire.read( );  // Read month from register 5
  year   = Wire.read( );    // Read year from register 6
  DS1307_display( );     // Diaplay time & calendar
  delay(50);     // Wait 50ms
}
```

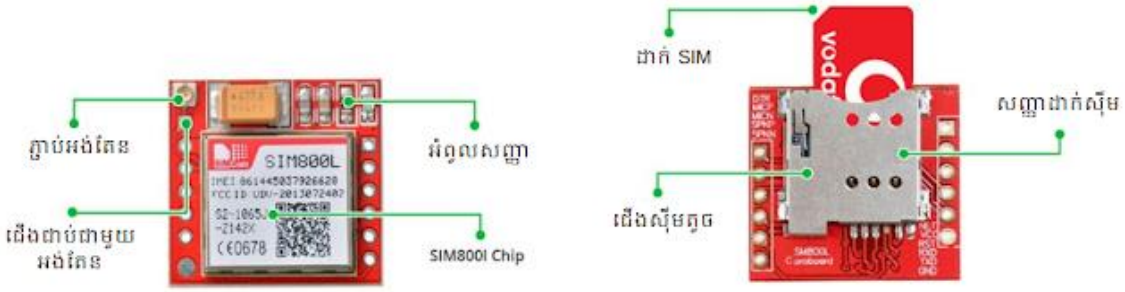
6. SIM800L ការប្រើស្វ័យប្រវត្តិកម្មក្នុងផ្ទះ(SIM800L HOME AUTOMATION)

ការប្រើ SIM800L ឬ SIM900L ដោយភ្ជាប់វាជាមួយនឹងប៉ត Arduino រវាងទូរស័ព្ទដើម្បីបញ្ជាឧបករណ៍ពីចំងាយដោយប្រើជាសារ (Message) និង Call ដើម្បីបញ្ជាបរិក្ខារពីចម្ងាយដោយអាចបិទ ឬបើកនូវឧបករណ៍ផ្សេងៗដោយប្រើលេខទូរស័ព្ទប្រភេទ Smart ឬ Cellcard។ ការបង្កើតវាឡើងដើម្បីស្វែងយល់ពីបច្ចេកវិទ្យាទំនាក់ទំនងដែលអាចយកទៅប្រើក្នុងគេហដ្ឋាន, សាលារៀន ឬទីលានសាធារណៈផ្សេងៗក្នុងជីវិតប្រចាំថ្ងៃដោយប្រើទូរសព្ទដៃជាគ្រឿងបញ្ជាដែលយើងហៅថាឧបករណ៍បញ្ជាពីចម្ងាយ។

6.1. ការដំឡើងឧបករណ៍

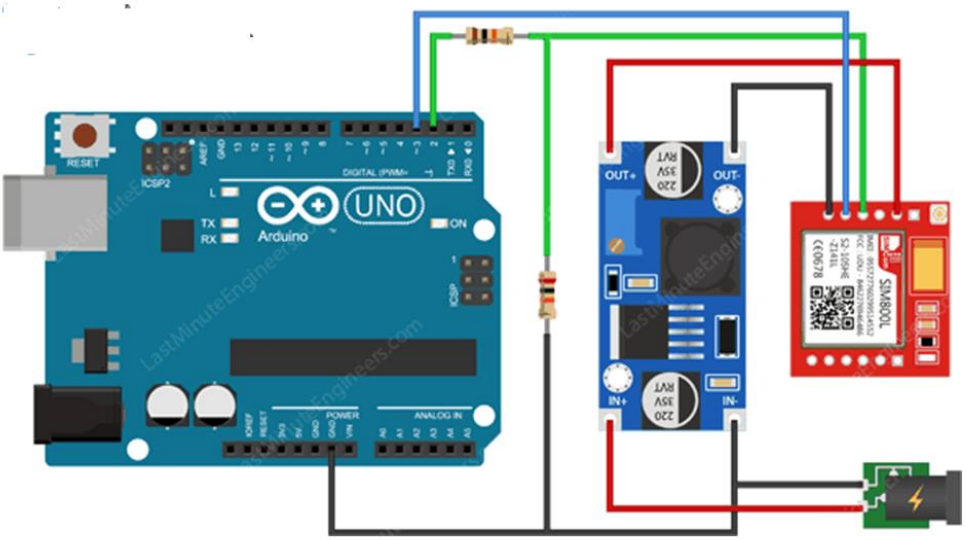
- Arduino Uno ឬ Arduino Nano

- Sim800L ជាម៉ូឌុលខ្នាតតូចដែលអនុញ្ញាតឱ្យបញ្ជូន GPRS។ ដោយអាចបញ្ជូននិងទទួលសារ និងទទួលការហៅជាសំឡេងបាន។



- Step Down សម្រាប់បញ្ចូលភ្លើង ១២ វុល និងផ្តល់ភ្លើងទៅឱ្យ GSM ម៉ូឌុលអាចនៅចន្លោះ 3.4V - 4.4V និងផ្តល់ចរន្តយ៉ាងហោចណាស់ 2A។ សូមមើលរូបខាងក្រោម។

6.2. ការតភ្ជាប់



6.3. ជំហានការតេស្ត Sim800L និងប្រើទូរសព្ទ

1. ខ្សែ GND ត្រូវយកពី BATERRY ឬពី Power Jack ដើម្បីប្រើភ្លើងចាប់ពី ៧វុលឡើងទៅ
2. បើ AT command AT+CNMI=2,2,0,0 មាន ERROR អាចមកពីការប្រើខ្សែ GND
3. នៅលើប៉តមានភ្លើងបំភ្លឺពេលវាដំណើរការបញ្ជាក់ថាស្គាល់ port
4. ប្រើភ្លើង 3.7 Volt HP battery ទៅ supply sim800L
5. នៅលើ Smart Phone វាយ

#a1 (Turn On Led1)

#a0 (Turn off Led1)

6.4. ការប្រើប្រាស់ AT Command មានដូចជា៖

- AT ប្រើតេស្តទំនាក់ទំនងរវាង Arduino និង GSM បើទាក់ទងគ្នាបានវានឹងឆ្លើយ OK។
- AT+CSQ ប្រើឲ្យ GSM មើលសេវាស៊ីមដែលវាបានប្រើ ជាតួលេខដំបូង មានតម្លៃប្រហែល 5dB ឬច្រើន ជានេះបើសេវាខ្លាំង។

- ការប្រើប្រាស់ GSM បានកាន់តែទូលំទូលាយ អាស្រ័យលើការសិក្សាលើបញ្ជា AT។ យល់ដឹងពីបញ្ជា AT បានកាន់តែច្រើនធ្វើឱ្យងាយស្រួលប្រើប្រាស់ GSM ម៉ូឌុល។

ដើម្បីផ្ញើសារ SMS បាន៖

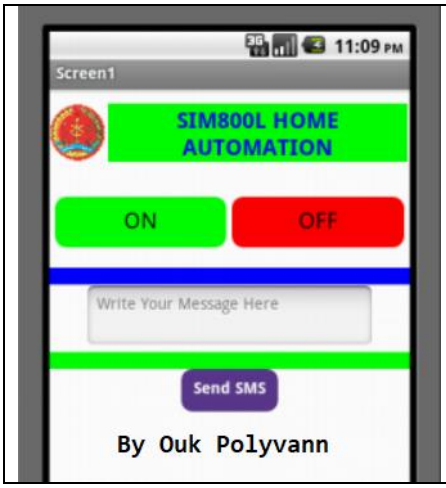
- AT+CMGF=1 ឲ្យ GSM ដំនើរការក្នុងគោលការណ៍ SMS។
- AT+CNMI=1,2,0,0,0 ឲ្យ GSM ចាំទទួលនឹងចាត់ចែងសារពីអ្នកផ្សេងត្រូវប្រើ
- AT+CMGL=\REC UNREAD\ ឲ្យ GSM Read សារមិនទាន់ Read
- AT+CMGDA=\DEL ALL\ ឲ្យ GSM លុប Messages

6.5. កម្មវិធីគំរូខាងក្រោមនេះអាចតេស្តជាមួយនឹងផ្ទៃក្រឹងជាក់ស្តែង

```
//Program :How To Use Arduino with SIM800L
//to call and send message
//By :Ouk Polyvann
#include <SoftwareSerial.h>// បញ្ចូល Library ដើម្បីបង្កើត Object
SoftwareSerial sim(2, 3);//ជាង Pin RX : TX សម្រាប់តភ្ជាប់ទៅប៉ត Arduino
String number = "85511995516"; // +855 is the country code
void setup( ) {
   Serial.begin(9600);//បើក USB port និងកំណត់ល្បឿនឱ្យ Serial port
   Serial.println("Wait few seconds...");//បង្ហាញឃ្លា Wait few seconds...")
   delay(5000);// delay រយៈពេល ៥ វិនាទី
   Serial.println("Sistem Started..."); //បង្ហាញឃ្លា Sistem Started...
   sim.begin(9600);//ចាប់ផ្តើម Sim និងកំណត់ល្បឿន Serial Port
   delay(1000); // delay រយៈពេល ១ វិនាទី
   Serial.println("Type c to make a call and s to send an SMS");
}
void loop( ) {
  if (Serial.available( ) > 0)//ប្រើទិន្នន័យត្រូវបានផ្ញើចេញ
    switch (Serial.read( )){
      case 'c':
          callNumber( );//ហៅអនុគមន៍ CallNumber( )
          break;
      case 's':
         SendMessage( );
         break; }
  if (sim.available( ) > 0)
        Serial.write(sim.read( ));}
void SendMessage( ){
   //Selects the operating mode of the GSM/GPRS modem or mobile phone
```

```
    sim.println("AT+CMGF=1");
    delay(1000);
    sim.println("AT+CMGS=\"" + number + "\"\r");
    delay(1000);
    String SMS = "Hi , How are you?";
    sim.println(SMS);
    delay(100);
    sim.println((char)26);
    delay(1000);}
void callNumber( ) {
    sim.print (F("ATD"));
    sim.print (number);
    sim.print (F(";\r\n"));}
```

ដើម្បីធ្វើការដំណើរការកូដខាងលើយើងត្រូវបើកកម្មវិធី ឬផ្ទើដូចខាងក្រោម៖



- កម្មវិធីនេះប្រើសម្រាប់ធ្វើការផ្ញើតួរអក្សរទៅឱ្យ ប៉ត Arduion ដើម្បីធ្វើការបិត ឬបើកអំពូល
- វាយ **#a1** នៅក្នុងប្រអប់ Write Your Message Here និងចុចលើប៊ូតុង Send SMS ដើម្បីធ្វើការបើក(Turn On Led1)

  វាយ **#a0** នៅក្នុងប្រអប់ Write Your Message Here និងចុចលើប៊ូតុង Send SMS ដើម្បីធ្វើការបើក(Turn On Led1)

6.6. ប្រជាមួយនឹងកម្មវិធី Proteus

```
int led2Pin = 6; //ប្រកាស led2 កំណត់ថាជើង Pin លេខ 6 នឹងភ្ជាប់ទៅកាន់ led
//បង្កើត software serial object ដើម្បីភ្ជាប់ទំនាក់ទំនងជាមួយ SIM900D
SoftwareSerial mySerial(12,13);
void setup( ){
        mySerial.begin(9600);  // បើក Serial Monitor SIM900D
        Serial.begin(9600);  //បើក Serial Monitor Arduino
        pinMode(4, OUTPUT);
        pinMode(5, OUTPUT);  //កំណត់ Digital Pin led ជា Output
        pinMode(6, OUTPUT);
        delay(100); }// ពន្យាពេលដើម្បីប្តូរក្នុងរយៈពេល 0.1 វិនាទី
```

```
void loop( ){
      if ( Serial.available( )>0 )
      switch( Serial.read( ) ) {
            case 'a':
               SendMessage( );
               break;
           case 'b':
            SendMessage1( );
               break;
            case 'c' :
                  SendMessage2( );
                  break;
            case 'd':
                  EndMessage( );
                  break;  }
if ( mySerial.available( )>0 )

                        Serial.write( mySerial.read( ) );}
```

```
void SendMessage( ){
        mySerial.println("     ");
        //កំណត់ SMS to Text Mode ឲ្យ GSM ដំណើរការក្នុងគោលការណ៍ SMS
         mySerial.println("AT+CMGF=1");
         delay(100);//ពន្យាពេលដើម្បីបួរក្នុងរយៈពេល 0.1 វិនាទី
         mySerial.println("AT+CMGS=\"+885xxxxxxxxx\"\r");
         delay(100);//ពន្យាពេលដើម្បីបួរក្នុងរយៈពេល 0.1 វិនាទី
         mySerial.println("Control LED with GSM ");
        mySerial.println("Press 'a' Lights ON");
         mySerial.println("The First Pattern");
         delay(100);//ពន្យាពេលដើម្បីបួរក្នុងរយៈពេល 0.1 វិនាទី
         digitalWrite(4, HIGH); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led ភ្លឺ
         digitalWrite(5, LOW); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led បិទ
         digitalWrite(6,LOW); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led បិទ
         delay(100);//ពន្យាពេលដើម្បីបួរក្នុងរយៈពេល 0.1 វិនាទី }
        void SendMessage1( )
        {
        mySerial.println("     ");
        //កំណត់ SMS to Text Mode ឲ្យ GSM ដំនើរការក្នុងគោលការណ៍ SMS
        mySerial.println("AT+CMGF=1");
        delay(100);//ពន្យាពេលដើម្បីបួរក្នុងរយៈពេល 0.1 វិនាទី
        mySerial.println("AT+CMGS=\"+885xxxxxxxxx\"\);
        delay(100); //ពន្យាពេលដើម្បីបួរក្នុងរយៈពេល 0.1 វិនាទី
        mySerial.println("Control LED with GSM ");
        mySerial.println("Press 'b' Lights ON");
        mySerial.println("The Second Pattern");
```

```
delay(100); //ពន្យាពេលដើម្បីប្តូរក្នុងរយៈពេល 0.1 វិនាទី
digitalWrite(4, LOW); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led បិទ
digitalWrite(5, HIGH);//កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led ភ្លឺ
digitalWrite(6,LOW); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led បិទ
delay(100); } //ពន្យាពេលដើម្បីប្តូរក្នុងរយៈពេល 0.1 វិនាទី
void SendMessage2( )
{
mySerial.println("     ");
//កំណត់ SMS to Text Mode ឲ្យ GSM ដំនើរការក្នុងគោលការណ៍ SMS
  mySerial.println("AT+CMGF=1");
 delay(100);
 mySerial.println("AT+CMGS=\"+885xxxxxxxxx\"\r");
 delay(100);//ពន្យាពេលដើម្បីប្តូរក្នុងរយៈពេល 0.1 វិនាទី
 mySerial.println("Control LED with GSM ");
 mySerial.println("Press 'c' Lights ON");
 mySerial.println("The Third Pattern");
 delay(100);//ពន្យាពេលដើម្បីប្តូរក្នុងរយៈពេល 0.1 វិនាទី
 digitalWrite(4, LOW); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led បិទ
 digitalWrite(5, LOW); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led បិទ
 digitalWrite(6,HIGH); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led ភ្លឺ
 delay(100);}//ពន្យាពេលដើម្បីប្តូរក្នុងរយៈពេល 0.1 វិនាទី
}
void EndMessage( )
 {
   mySerial.println("     ");
   mySerial.println("ATD +885xxxxxxxxx:");
   mySerial.println("Press 'd' Lights Off");
  delay(100);//ពន្យាពេលដើម្បីប្តូរក្នុងរយៈពេល 0.1 វិនាទី
  digitalWrite(4, LOW); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led បិទ
  digitalWrite(5,LOW); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led បិទ
  digitalWrite(6,LOW); //កំណត់ចង្អុលបង្ហាញយកតម្លៃ digital pin led បិទ
 delay(100);//ពន្យាពេលដើម្បីប្តូរក្នុងរយៈពេល 0.1 វិនាទី
 }
```

សំណួរ និង លំហាត់

១. ដូចម្តេចដែលហៅថាRC Car? តើវាផ្សំឡើងពី hardware អ្វីខ្លះ? ចូរៀបរាប់ពីមុខងាររបស់ឧបករណ៍នីមួយៗដោយសង្ខេប? តើ Motor driver ប្រើភ្លើងចូលប៉ុន្មានវ៉ុល? រួចពន្យល់ពីមូលហេតុដោយសង្ខេប។

២. ចូរសរសេរកូដដើម្បីបញ្ជាម៉ូទ័រទាំងពីរនៃ RC Car ទៅមុខ ក្រោយ ឆ្វេង និង ស្តាំដោយមានបង្ហាញព័ត៌មាននៅលើ LCD នៅក្នុងវិធី Proteus?

៣. ដូចម្តេចហៅថា Bluetooth Control Lamp? តើវាផ្សំពីឧបករណ៍អ្វីខ្លះ? ពន្យល់មុខងារ Bluetooth និងជើង Pin នីមួយៗរបស់វារួចធ្វើការតភ្ជាប់ទៅប៉ត Arduino។

៤. ចូរបញ្ជាក់ពីមូលហេតុបង្កើត Simple Calculator? ចូរប្រាប់ឈ្មោះឧបករណ៍ដែលបង្កើតគម្រោងនេះ? តើគម្រោងនេះជាមូលដ្ឋានសម្រាប់បង្កើតបានកុំព្យូទ័រមួយបានដែរឬទេ? ចូរបញ្ជាក់ពីមូលហេតុ? ចូរសរសេរកូដតេសវានៅក្នុងកម្មវិធី Proteus?

៥. ដូចម្តេចដែលហៅថា Automation? ចូរនិយាយពីតួនាទីឧបករណ៍ប្រើក្នុង Automatic Watering System? តើគម្រោងនេះអាចយកទៅប្រើនៅក្នុងវិស័យកសិកម្មអ្វីបាន? ចូរសរសេរកូដតេសវានៅក្នុងកម្មវិធី Proteus?

៦. ដូចម្តេចដែលហៅឌីជីថល Clock? ចូរនិយាយពីមុខងារនៃឧបករណ៍នីមួយៗដោយសង្ខេប? ចូរសរសេរកូដដើម្បីចេញលេខពី 0 ដល់លេខ ៩ នៅលើ TM1637?

៧. ដូចម្តេចដែលហៅ **SIM800L controlled 3 LED**? តើផ្សំឡើងពីhardware អ្វីខ្លះ? វាប្រើនៅក្នុងវិស័យអ្វីខ្លះ? ចូររៀបរាប់ពីមុខងារនីមួយៗរបស់វាដោយសង្ខេប។

មេរៀនទី ១០ APP INVENTOR

វាជាកម្មវិធីមួយដែលអភិវឌ្ឍដោយសាកលវិទ្យាល័យមួយឈ្មោះថា MIT(Massachusetts Institute of Technology) ហើយត្រូវបានអនុញ្ញាតឲ្យអ្នកប្រើប្រាស់ប្រើវាដើម្បីបង្កើតកម្មវិធីផ្សេងៗសម្រាប់ទូរស័ព្ទប្រភេទ Android បានដោយងាយស្រួល។ អ្នកអាចចូលប្រើប្រាស់វាបាននៅលើគេហទំព័រ។

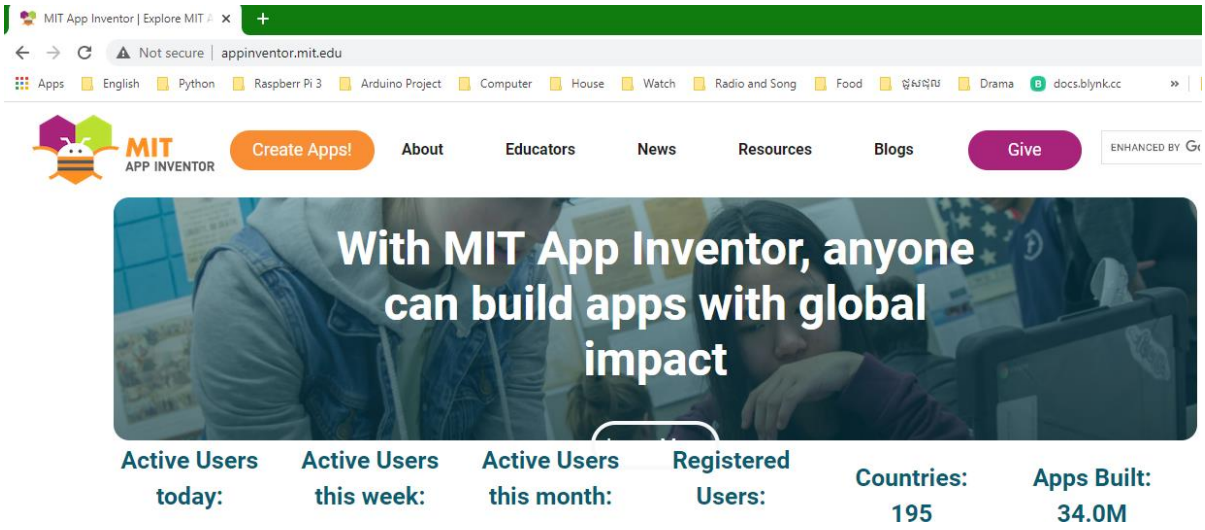
App Inventor ត្រូវបានរចនាឡើងដើម្បីផ្តល់ភាពងាយស្រួលដល់អ្នកចាប់ផ្តើមបង្កើតកម្មវិធីដំបូង ហើយវាមានលក្ខណៈពិសេសនានាដូចបានរៀបរាប់នៅចំនុចមួយចំនួនខាងក្រោម បើធៀបទៅនឹង IDE (Integrated Development Environments) ដទៃទៀតដូចជា Android Studio ឬ Eclipse ជាដើម។

- ដំណើរការនៅលើគេហទំព័រអ៊ីនធឺណិត និងមិនត្រូវការឲ្យអ្នកប្រើប្រាស់ទាញយក និងដំឡើងវាឡើយ
- ប្រើប្តូកម៉ូត(Block Modes),បណ្តុំលក្ខខណ្ឌ និងពាក្យបញ្ជាហៅថាប្តូក(Blocks) ជំនួសការសរសេរកូដ និងពុំត្រូវការឲ្យអ្នកប្រើប្រាស់ចេះភាសាសរសេរកម្មវិធីអ្វីឡើយក៏អាចប្រើវាបានដែរ។
- មាន Interfaces សាមញ្ញ និងងាយយល់ក្នុងការប្រើប្រាស់
- មានកម្មវិធីជំនួយនៅលើទូរស័ព្ទ Android ដែលអនុញ្ញាតឲ្យអ្នកប្រើប្រាស់ធ្វើការតេស្តសាកល្បងកម្មវិធីភ្លាមៗ
- ជាកម្មវិធីដ៏ប្រសើរមួយសម្រាប់អ្នកដែលចង់ចាប់ផ្តើមបង្កើតកម្មវិធីសម្រាប់ទូរស័ព្ទ Android ដោយពុំត្រូវការចំណាយពេលយូរក្នុងការរៀន។

នៅក្នុងមេរៀននេះ អ្នកនឹងស្វែងយល់អំពីរបៀបបង្កើតគម្រោង, ធ្វើតេស្ត, រចនា និងសរសេរកូដដោយប្រើប្រាស់ប្តូក ដែលជាមូលដ្ឋានក្នុងការប្រើប្រាស់ App Inventor។

1. <u>របៀបបង្កើតគម្រោង</u> (Project)

មុននឹងចាប់ផ្តើមរចនា និងបង្កើតកម្មវិធីអ្នកត្រូវស្វែងយល់ជាមុនសិនអំពីរបៀបបង្កើតគម្រោង និងរៀបចំដំឡើងកម្មវិធីជំនួយមួយចំនួនដែលអនុញ្ញាតឲ្យអ្នកធ្វើតេស្តកម្មវិធីរបស់អ្នកទៅបាន។



- គម្រោងជាគំនូរប្លង់កម្មវិធីរបស់អ្នកហើយវាផ្តុំកនូវការរៀបចំរចនា និងគ្រោងដំណើរការរបស់កម្មវិធី។ដោយសារតែ App Inventor ដំណើរការនៅលើគេហទំព័រ ដូច្នេះការចូលប្រើប្រាស់ និងបង្កើតគម្រោងមានលក្ខណៈខុសប្លែកពី IDE ដទៃ។ ដើម្បីប្រើកម្មវិធីនេះបានត្រូវអនុវត្តតាមជំហានដូចខាងក្រោម៖
- **ជំហានទី១៖** សូមចូលទៅកាន់គេហទំព័រ http://appinventor.mit.edu ហើយចុចលើប៊ូតុង "Create apps!" ឬចូលទៅកាន់គេហទំព័រ http://ai2.appinventor.mit.edu ដោយផ្ទាល់។
- ម៉្យាងទៀតយើងអាចវាយពាក្យ App Inventor 2 ក្នុង Google និងចុច Mouse លើ MIT App Inventor

- **ជំហានទី២៖** វាយបញ្ចូលនូវ Gmail របស់អ្នកតាមដែលបានស្នើសុំដើម្បី login ចូលប្រើប្រាស់ App Inventor។
- **ជំហានទី៣៖** សូមចុចលើប៊ូតុង "Start new project" និងអនុវត្តតាមរូបដូចខាងក្រោមពីជំហានទី ១ ដល់ទី ៣ ដើម្បីបង្កើត និងដាក់ឈ្មោះគម្រោងថ្មី។

ក្រោយពីអនុវត្តន៍តាមជំហានខាងលើហើយ អ្នកនឹងសម្រេចក្នុងការបង្កើតគម្រោងផ្សេងៗហើយក្នុងកម្មវិធីនេះ នឹងនាំអ្នកទៅកាន់អេក្រង់ថ្មីសម្រាប់រចនាកម្មវិធីរបស់នេះ។

2. របៀបធ្វើតេស្ត និងដំឡើងកម្មវិធីដំបូង

មុននឹងចាប់ផ្តើមរចនាកម្មវិធីរបស់អ្នក ចាំបាច់ត្រូវរៀបចំកម្មវិធីសម្រាប់ធ្វើតេស្តសាកល្បង ឬយើងហៅថា Output Screen ជាមុនសិន។ ព្រោះអ្វីដែលអ្នកឃើញនៅក្នុងផ្ទាំងរចនារបស់ App Inventor ពុំដូចពិតប្រាកដទៅនឹងអ្វីដែលអ្នកនឹងឃើញនៅក្នុងទូរស័ព្ទឡើយ។

អ្នកអាចធ្វើតេស្តកម្មវិធីរបស់អ្នកបានបីរបៀបគឺ៖

- ប្រើកម្មវិធីជំនួយ AI Companion តាមរយៈ Wifi
- ប្រើកម្មវិធីជំនួយ AI Companion តាមរយៈអេសប៊ីត(USB)
- ប្រើ Emulator នៅលើកុំព្យូទ័រដូចអ្នកប្រើកម្មវិធី UnoArduSim ឬ Proteus។ ដោយតំរូវឱ្យដំឡើងកម្មវិធី aiStarter អាចទាញយកពីទីតាំងណាមួយនៅលើ Google និងដំឡើងវា។

ដើម្បីរក្សាភាពសាមញ្ញនៅក្នុងមេរៀននេះខ្ញុំនឹងរៀបរាប់តែពីការធ្វើតេស្តដោយប្រើ AI Companion តាមរយៈ Wifi និងការធ្វើតេស្តដោយប្រើ Emulator និង AI Companion តាមរយៈ USB ដោយយើងអាចចូលទៅកាន់គេហទំព័រនេះដើម្បីទាញយកកម្មវិធីទាំងនេះ រួចធ្វើការដំឡើង។

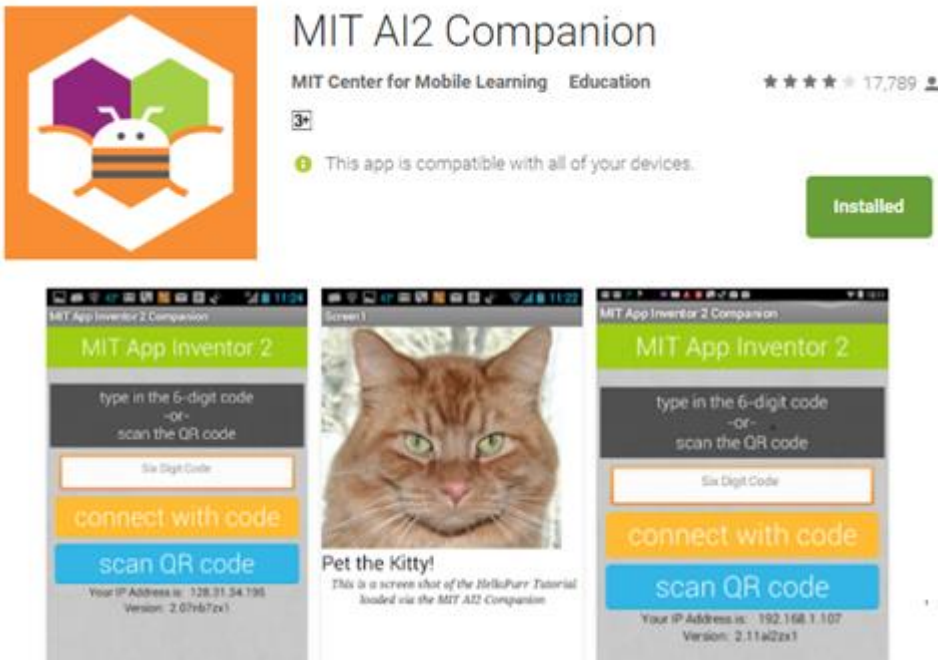
2.1. ដំឡើង aiStarter

ការប្រើប្រាស់កម្មវិធី Emulator ឬខ្សែយូអេសប៊ីតត្រូវឱ្យប្រើកម្មវិធីមួយដែលមានឈ្មោះថា aiStarter វាជាកម្មវិធីជំនួយដែលអនុញ្ញាតឱ្យមានទំនាក់ទំនងរវាង Browser និង Emulator ឬខ្សែយូអេសប៊ីនៃ App Inventor។ ពេលអ្នកបញ្ចូលកម្មវិធីតាមជំហានខាងក្រោម និងបើកវា៖

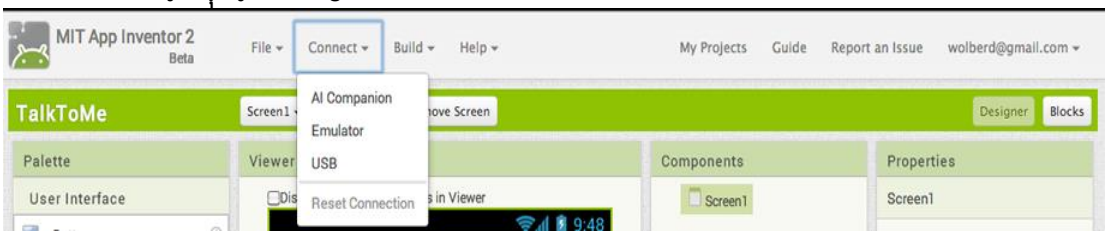
1. អាចទាញយក aiStarter ពីទីតាំងណាមួយនៅលើ Google ឬយកពីប្រភពណាមួយ
2. ធ្វើការដំឡើងកម្មវិធីដោយតាមការណែនាំដែលមាននៅលើកម្មវិធីរហូតដល់ចប់
3. បង្កើតកម្មវិធីសូមមើលនៅត្រង់ទំព័រខាងក្រោម

2.2. ដំឡើង AI Companion

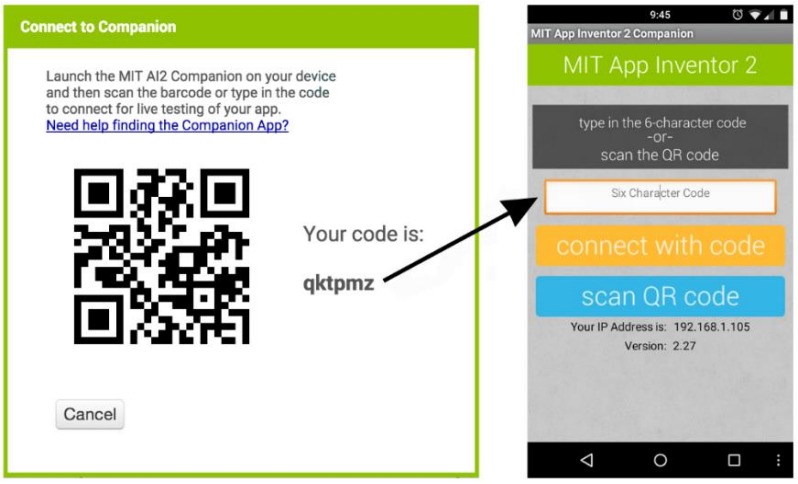
- **ជំហានទី១៖** នៅលើទូរស័ព្ទ Android របស់អ្នក សូមទាញយកកម្មវិធី MIT AI2 Companion ពី Google Play Store



- **ជំហានទី២៖** នៅក្នុងគម្រោងរបស់អ្នក សូមចុចលើប៊ូតុង Connect ហើយជ្រើសយក AI Companion ដូចក្នុងរូបខាងក្រោម។



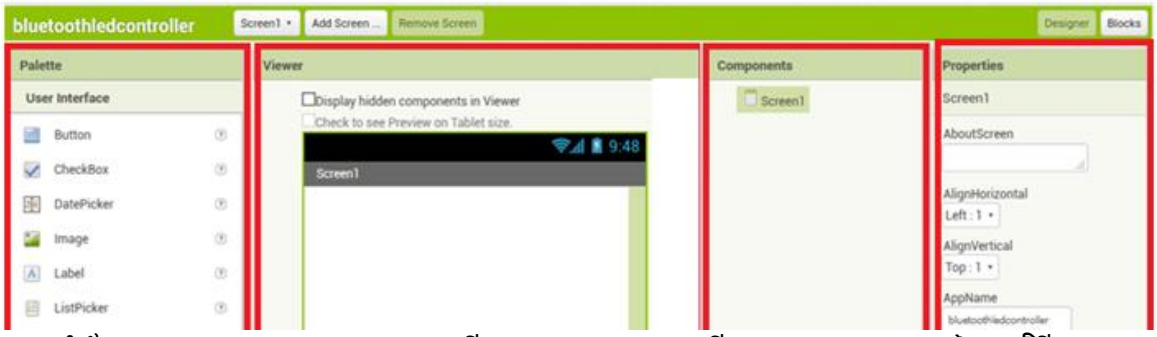
- **ជំហានទី៣៖** អ្នកនឹងឃើញផ្ទាំង QR Code មួយលេចឡើយ ហើយអ្នកត្រូវស្កេន QR Code នេះតាមរយៈកម្មវិធី MIT AI2 Companion ដែលអ្នកបានដំឡើងលើទូរស័ព្ទរបស់អ្នក។



ក្រោយពីធ្វើតាមជំហានខាងលើហើយ កម្មវិធី MIT AI2 Companion នឹងភ្ជាប់ទៅកាន់គម្រោងរបស់អ្នកនៅលើកុំព្យូទ័រ ហើយរាល់ការផ្លាស់ប្តូរដែលអ្នកធ្វើនៅលើគម្រោងនោះនឹងត្រូវបានបង្ហាញភ្លាមៗនៅលើអេក្រង់ទូរស័ព្ទរបស់អ្នក។

2.3. រមៀបរចនា

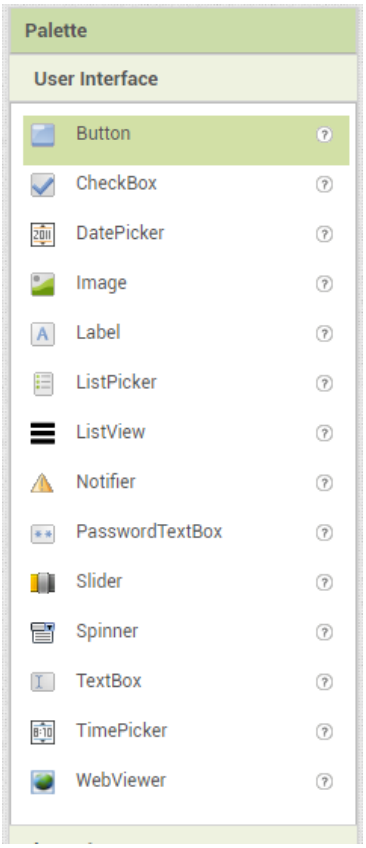
នៅពេលអ្នកបើកគម្រោង(Project) របស់អ្នកភ្លាម អ្នកនឹងឃើញគេហទំព័រដូចខាងក្រោម៖



គេហទំព័រនេះហៅថា Designer Editor ហើយវាអនុញ្ញាតឲ្យអ្នកធ្វើការរចនា Interface នៃកម្មវិធី។ នៅក្នុងគេហទំព័រនេះ មានផ្ទាំងចំនួនប្រាំដូចដែលអ្នកបានឃើញក្នុងរូបខាងលើ៖

- **Palette**៖ ជាកន្លែងសម្រាប់ឲ្យអ្នកជ្រើសរើស User Interface, Layout, Sensor, Media, Drawing Animation, Map, Social, Storage, Connectivity ដើម្បីទាញយកធាតុនិមួយៗដាក់ចូលក្នុងកម្មវិធី។
- **Viewer**៖ ជាកន្លែងសម្រាប់ឲ្យអ្នកទាញនូវសមាសភាគផ្សេងៗពី Platte ដាក់ចូល និងបង្ហាញពីទម្រង់ទូទៅនៃកម្មវិធីមួយ។
- **Components**៖ ជាផ្ទាំងបញ្ជីឈ្មោះធាតុដែលបានបង្កើតនៅក្នុង Viewer និងអាចជ្រើសរើសវាធ្វើអ្វីមួយ
- **Media**៖ ជាកន្លែងសម្រាប់ឲ្យអ្នក Upload ឯកសារដទៃដូចជា រូបភាព និងសម្លេងចូលក្នុង Project របស់អ្នកសម្រាប់ប្រើប្រាស់ក្នុងកម្មវិធីអ្នក។
- **Properties**៖ ជាកន្លែងសម្រាប់ឲ្យអ្នកកំណត់លក្ខណៈឲ្យសមាសភាគដែលបានបង្កើតនៅលើ Viewer របស់អ្នក។ តាមរូបខាងក្រោមមានផ្នែជាបន្តបន្ទាប់ដើម្បីស្វែងយល់ឱ្យកាន់តែលម្អិតអំពីផ្ទាំង Palette, Components និង Properties។

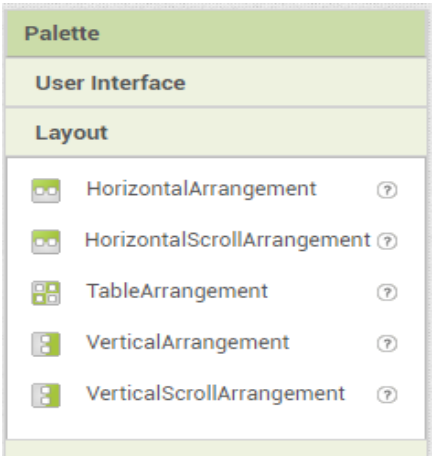
អំពី Palette



នៅក្នុងផ្ទាំង Palette មាន Components ជាច្រើនដូចជាប៊ូតុង, Bluetooth, Barcode Scanner ជាដើម ដែលត្រូវបានរៀបជាផ្នែកៗ ហើយអ្នកអាចជ្រើសរើសយកមកដាក់ក្នុងកម្មវិធីអ្នកបាន។ ប៉ុន្តែ Components ដែលអ្នកនឹងត្រូវការប្រើប្រាស់ច្រើនបំផុត គឺ Components នៅក្នុងផ្នែក User Interface និង Layout។

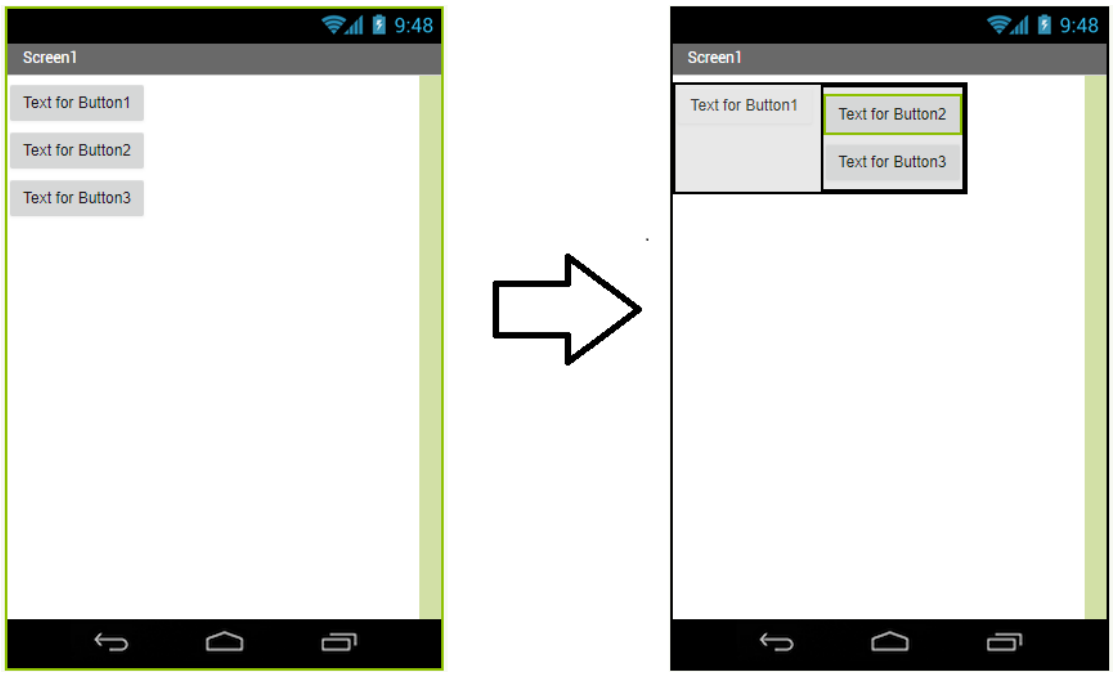
User Interface ជា ផ្នែកមួយដែលផ្ទុកនូវ Components ទាំងឡាយដែលនឹងត្រូវបង្ហាញក្នុងកម្មវិធីរបស់យើង ហើយមានអន្តរកម្មជាមួយអ្នកប្រើប្រាស់ ដូចជា Button, Checkbox, Image, Textbox ជាដើម ហៅថា Controls ឬ Budget ។

ឧទាហរណ៍បើយើងចង់បង្កើតប៊ូតុងនៅក្នុងផ្នែ Viewer យើងយក Mouse ដាក់លើ Button និចចុច Mouse អោយជាប់ រួចទាញវាដាក់ក្នុង Viewer ។ ចំពោះ Control និមួយមាននិម្មិតសញ្ញាសំគាល់ ងាយយកវាទៅប្រើក្នុងបង្កើតប្លុកកូដ ។



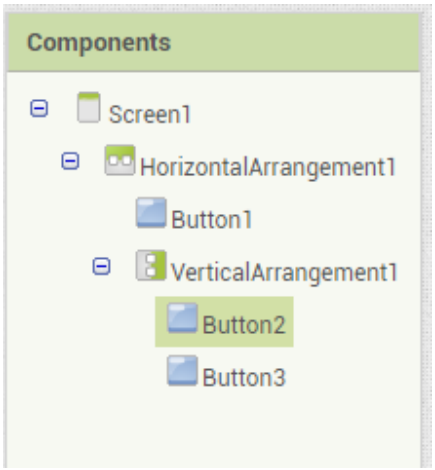
ចំណែក Layout វិញជាផ្នែកដែលផ្ទុកនូវគំរូប្លង់ដែលសម្រាប់កំណត់ការតម្រៀបនៃសមាសភាគក្នុងកម្មវិធីរបស់អ្នកដូចរូប ។ វាមានសារៈសំខាន់ព្រោះប្រសិនបើអ្នកមិនប្រើ Layout ទេនោះកម្មវិធីអ្នកនឹងតម្រៀប Componentsទាំងឡាយពីលើចុះក្រោមដោយស្វ័យប្រវត្តិ ។ ក្នុងករណីយើងអត់បាន ប្រើធាតុនៅក្នុង Layout នោះពេលបង្កើត Control មួយៗ វាចុះបន្ទាត់តាមលំដាប់លំដោយ ។ តែប្រើ យើងប្រើវាយើងអាចកំណត់ Control ទៅតាមទីតាំង ដែលយើងចង់បាន

Layout មានពណ៌ប្រផេះដោយ Default។ ប្រសិនបើអ្នកមិនចង់បង្ហាញពណ៌ អ្នកអាចកែ Properties របស់វាឈ្មោះ BackgroundColor ពី Default ទៅជា None វិញ។ គួរកត់សំគាល់ដែរថានៅក្នុងផ្នែនេះជាកន្លែងដ៏សំខាន់មួយដែលអោយយើងធ្វើការជ្រើសរើសតាមការងាររចនាតាមរដ្ឋរដែក និងជួរឈរទៅតាមបំណងរបស់យើងចង់បាន។



រូបខាងឆ្វេងអត់ប្រើ Layout និងរូបស្តាំប្រើ Horizontal Layout ដែលបានប្រើជា Vertical Layout ពីក្នុងបន្ទាប់ Button ទី១

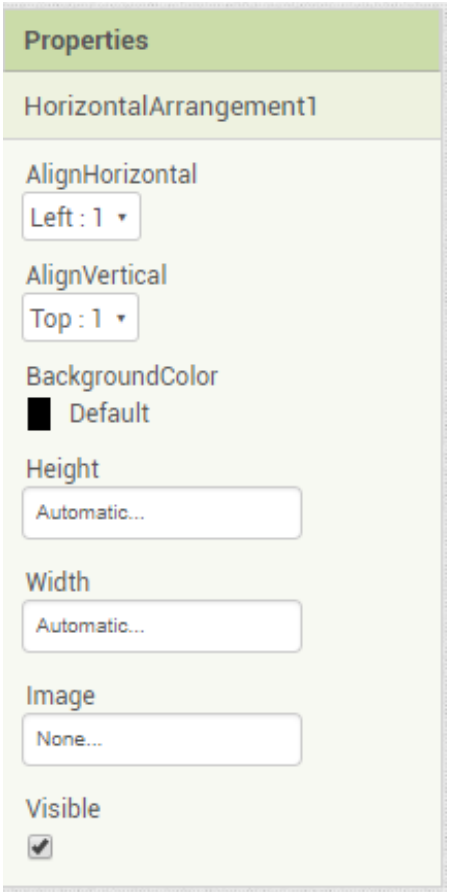
អំពី Components



ផ្ទាំង Components បង្ហាញពី Components ទាំងអស់ដែលអ្នកបានយកមកប្រើប្រាស់ក្នុងកម្មវិធីរបស់អ្នក។ ជាឧទាហរណ៍ រូបខាងឆ្វេងនេះបង្ហាញពី Components ដែលបានប្រើក្នុងការតម្រៀប Layout ខាងលើ និងទីតាំងរបស់វានីមួយៗ ដូចជា Button2 និង Button3 ស្ថិតនៅក្នុង VerticalArrangement1 ។ វាជាកន្លែងដែលអាចយើងដឹងពីអ្វីៗដែលបានរចនាកម្មវិធី និងសម្រាប់យកវាទៅបង្កើតជា ប្លុកកូដ ។

អ្នកគួរតែកែឈ្មោះរបស់ធាតុ(Component)នីមួយៗជានិច្ច ដូចជាដាក់ឈ្មោះប៊ូតុងថា btnSubmit ជាង Button1 ដើម្បីការពារការភាន់ច្រឡំនៅពេលប្រើ Blocks Editor។ ផ្នែនេះបង្ហាញនូវអ្វីដែលយើងបានរចនានូវ Interface ដែលមាន Widgets ដែលបានយកមក User Interface និងប្រើវាសម្រាប់បង្កើតប្លុកកូដ

អំពី Properties



វាបង្ហាញពីលក្ខណៈរបស់ Component ដែលអ្នកកំពុង Select ហើយអ្នកអាចកំណត់លក្ខណៈទាំងនេះតាមតម្រូវការ ។ រូបខាងឆ្វេងជា Properties របស់ Horizontal Arrangement1 ហើយអ្នកអាចកំណត់ AlignHorizontal ដែលជាទីតាំងរបស់ Child Component ថាគួរតែនៅឆ្វេង កណ្តាល ឬខាងស្តាំក្នុង Horizontal Arrangement1 ។ អ្នកក៏អាចកំណត់ពណ៌, កម្ពស់, ទទឹង និងរូបភាពជា Background ឲ្យវាបានផងដែរ។

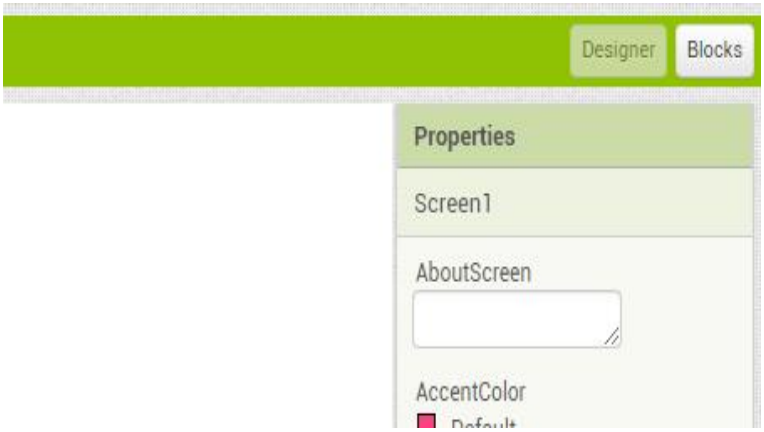
នៅក្នុងផ្ទាំងនេះយើងកំណត់លក្ខណៈអោយប៊ូតុង និង វិដ្ជកែត(Widgets) ផ្សេងៗដោយចុចMouse លើវានោះសិន ។ បន្ទាប់មកយើងអាចសរសេរអក្សរ រួចធ្វើអក្សរអោយដិត ទ្រេត និងជ្រើសរើសទំហំអក្សរនៅលើប៊ូតុងជាដើម ។ ហើយធាតុនិមួយៗនៅក្នុងផ្នែនេះអាចប្រែប្រួលទៅតាមយើងជ្រើសរើសសមាសភាគនៅក្នុងផ្ទាំង Components ។

3. របៀបបង្កើតកម្មវិធីដំបូង

3.1. របៀបរចនាលើ Designer

ប្រសិនបើអ្នកចង់បង្កើតប៊ូតុងមួយដែលមានអក្សរ "Click Me" ហើយស្ថិតនៅចំកណ្តាលអេក្រង់ទូរស័ព្ទ អ្នកត្រូវធ្វើតាមជំហានដូចខាងក្រោម៖

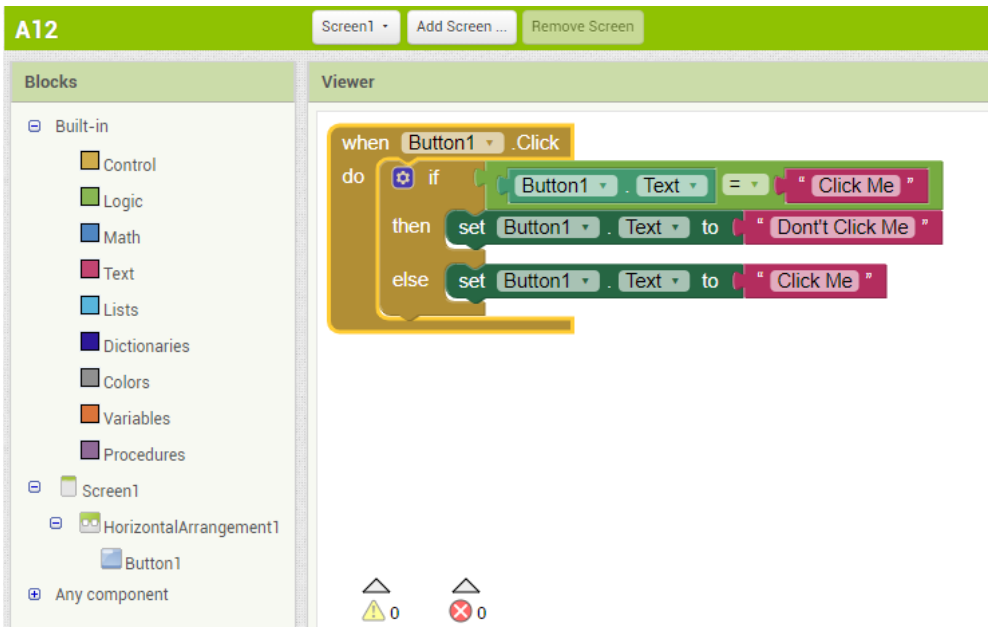
- **ជំហានទី១៖** ចុចលើ Screen1 នៅក្នុងផ្ទាំង Components ដើម្បីបង្ហាញពី Properties របស់វា។
- **ជំហានទី២៖** ក្នុងផ្ទាំង Properties របស់ Screen1 ប្តូរតម្លៃរបស់ AlignHorizontal និង AlignVertical ទៅជា Center ទាំងពីរ។ នេះនឹងធ្វើឲ្យរាល់ Components ទាំងអស់ដែល យើងដាក់បញ្ចូលបង្ហាញនៅត្រង់ចំណុចកណ្តាលអេក្រង់។
- **ជំហានទី៣៖** ទាញ Button ដែលស្ថិតនៅក្នុងផ្នែក User Interface នៃ Palette មកដាក់ក្នុងកម្មវិធីរបស់អ្នក នោះអ្នកនឹងឃើញប៊ូតុងមួយនៅកណ្តាលអេក្រង់ ហើយមានឈ្មោះថា **Button1**។
- **ជំហានទី៤៖** នៅក្នុងផ្ទាំង Properties របស់ Button1 ត្រង់ "Text" ប្តូរឈ្មោះពី Button1 ទៅឈ្មោះ "Click Me" ជំនួសវិញ។ ក្រោយពីធ្វើតាមជំហានខាងលើ អ្នកនឹងបានកម្មវិធីដែលមានទម្រង់ដូចខាងក្រោម៖



3.2. រចនាសម្ព័ន្ធ

ការបង្កើតប្លុកកូដនៅក្នុង App Inventor គឺតាមពិតមិនត្រូវការឲ្យអ្នកសរសេរអ្វីនោះឡើយ។ អ្នកគ្រាន់តែប្រើបណ្តុំលក្ខខណ្ឌ និងពាក្យបញ្ជាហៅថាប្លុកដើម្បីកំណត់នូវសកម្មភាពអ្វីមួយដែលអ្នកចង់ឲ្យដំណើរការក្នុងកម្មវិធីរបស់អ្នក។ រាល់ពេលបង្កើតនូវសមាសភាគផ្សេងៗដូចជាប៊ូតុង ឬ TexBox។ វាបង្កើតបង្កើត Block Mode និង Block Function រួមទាំងប្លុកផ្សេងៗទៀតដោយស្វ័យប្រវត្តិ។ នៅក្នុងកម្មវិធី App Inventor ក៏មាន Block សម្រាប់ប្រកាសអថេរ, ដាក់លេខ ឬអក្សរ, កន្សោមតក្កវិទ្យា, កន្សោមគណិតវិទ្យា និង ប្លុកដទៃជាច្រើនទៀតដែលហៅថា Build_in block។

ដើម្បីធ្វើដូច្នេះអ្នកត្រូវចូលទៅកាន់ Blocks Editor ដោយចុចលើប៊ូតុង Blocks ក្នុង Designer Editor រួចបង្កើតប្លុកកូដតាមរូបដូចខាងក្រោម៖



3.3. លក្ខណៈបច្ចេកទេស

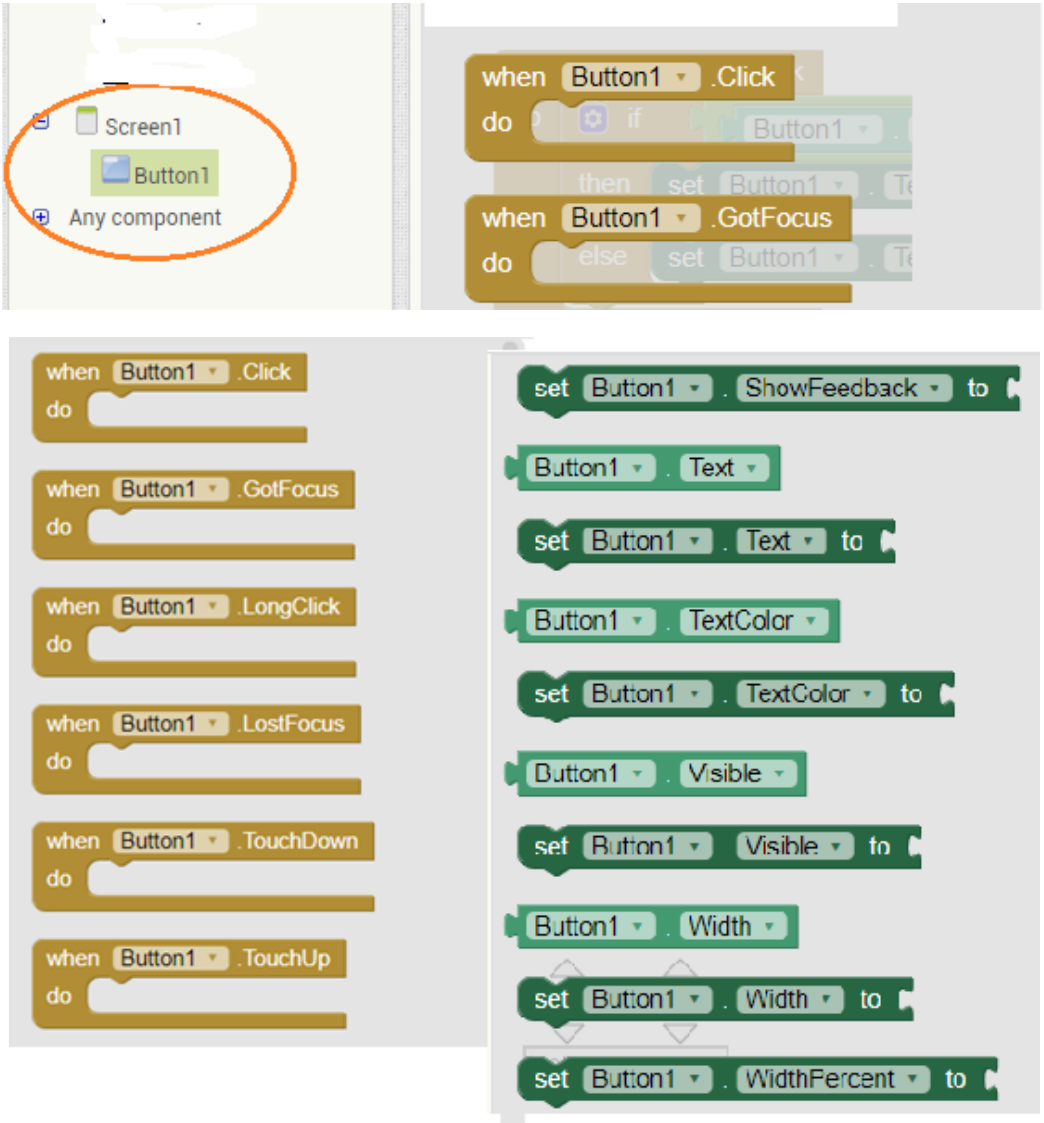
នៅក្នុង Blocks Editor មានបីផ្ទាំង៖

- **Blocks**៖ ជាកន្លែងដែលឲ្យអ្នកជ្រើសរើសនូវប្លុកផ្សេងៗមកប្រើមានប្លុក Control, Logic, Math,Text, List, Color, Variable, និង Procedure។
- **Media**៖ ជាកន្លែងបង្ហាញឯកសារដែលអ្នកបាន Upload ចូលក្នុង Project។
- **Viewer**៖ ជាកន្លែងសម្រាប់ឲ្យអ្នកទាញយក Blocks ទៅដាក់ផ្គុំគ្នាបង្កើតបានជាប្លុកកូដដើម្បីប្រើប្រាស់ Blocks Editor ឲ្យមានប្រសិទ្ធភាព ចំណុចសំខាន់ដែលអ្នកត្រូវស្វែងយល់គឺ Blocks។ Blocks ជាបណ្តុំលក្ខខណ្ឌ និងពាក្យបញ្ជាដែលយើងអាចយកមកផ្គុំគ្នាជាកន្សោមលក្ខខណ្ឌ សម្រាប់ឲ្យកម្មវិធីរបស់យើងមានសកម្មភាពឆ្លើយតបទៅនឹងអ្នកប្រើប្រាស់វិញបាន។
  Blocks ចែកចេញជាពីរប្រភេទគឺ Built-in Blocks និង Component-Specific Blocks ដែលនឹងរៀបរាប់បន្ថែមនៅផ្នែកខាងក្រោម។
  - Built-in Blocks

Built-in Blocks ជាប្រភេទប្លុកដែលអ្នកអាចប្រើប្រាស់បានជាទូទៅ។ វាចែកចេញជាប្រាំបីប្រភេទដូចក្នុងរូបខាងស្តាំ ហើយនៅពេលដែលអ្នកចុចលើវា ផ្ទាំងមួយនឹងលេចឡើងសម្រាប់ឲ្យអ្នកជ្រើសរើសប្លុកជាក់លាក់របស់ប្រភេទនោះ។ប្លុកនិមួយៗមានមុខងារខុសៗគ្នា

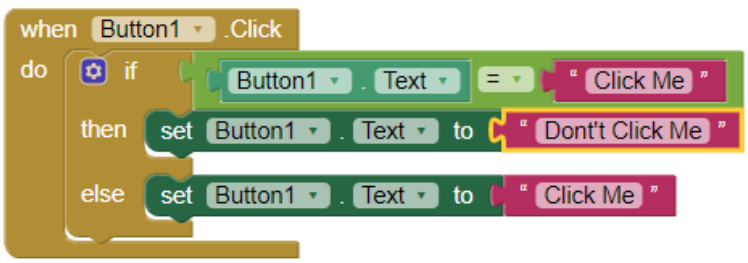
  - **Component-Specific Blocks ជា Blocks** ទាំងឡាយ ដែលត្រូវបានបង្ហាញនៅពីក្រោមផ្នែក Built-in Blocks។ វានីមួយៗតំណាងឲ្យ Controls ឬហៅ Widgets មានដូចជា Button,

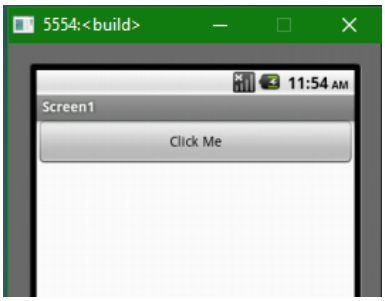
TextBox, Label,... ដែលអ្នកបានដាក់ចូលក្នុងកម្មវិធី។ ហើយអ្នកអាចប្រើវាដើម្បីជ្រើសរើសយកសកម្មភាពជាក់លាក់ទាក់ទងនឹង Widgets ទាំងនោះ។



3.4. របៀបបង្កើតប្លុកកូដនៅលើ Block Editor

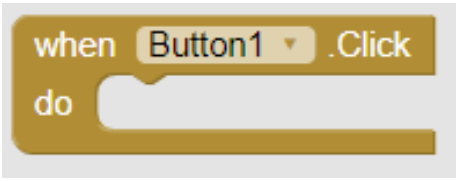
ជាទូទៅរាល់ប្លុកនិមួយៗដែលយកមកផ្គុំបានជាប្លុកកូដ ដោយបែងចែកជាច្រើនប្រភេទទៅតាមកម្មវិធីដែលយើងចង់ធ្វើ។ ខាងក្រោមគឺជាការប្រើប្លុកមួយចំនួនបង្កើតជាប្លុកកូដដើម្បីឱ្យកម្មវិធីយើងអាចប្រើប្រាស់នៅលើទូរស័ព្ទដោយចុចលើប៊ូតុង "Click Me" វាបង្ហាញពាក្យ " Don't Click" និងត្រលប់មកវិញ។



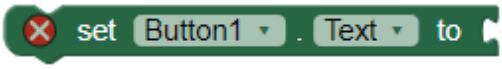


១.ព្រឹត្តការណ៍(Event)៖ ជាប្លុកម៉ូត(Block Mode)ដែលមាននាទីកំណត់ឲ្យមានសកម្មភាពនៅពេលដែលមានព្រឹត្តិការណ៍មួយកើតឡើង ដូចជាពេលអ្នកប្រើប្រាស់ចុចលើប៊ូតុងមួយ។ យើងអាចសម្គាល់

វាបាន ដោយវាមានរាងជាអក្សរ U ផ្តេក។ រូបខាងក្រោមជាប្លុកម៉ូត "when Button1.click" និងទាញមកប្រើចុច លើ Button1 និងធ្វើការជ្រើសរើសឱ្យបានដូចរូបខាងក្រោម៖



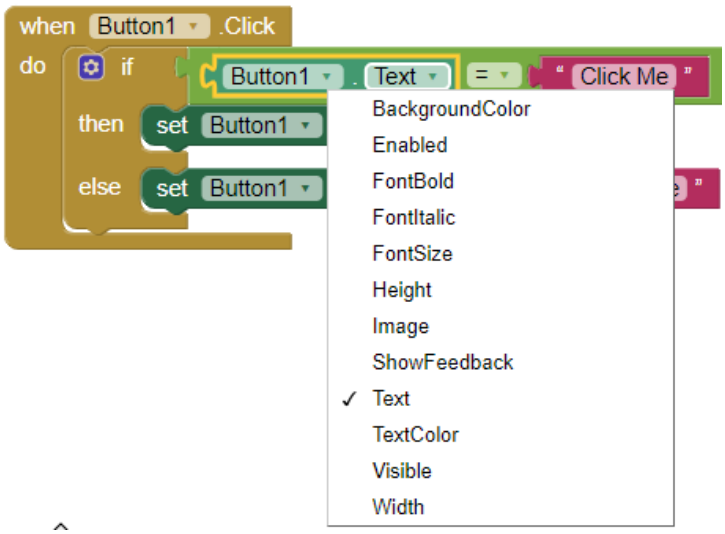
២.ប្លុកអនុគមន៍៖ ជា Block ឬជាអនុគមន៍ប្រើសម្រាប់កំណត់អំពើ, សកម្មភាពធ្វើអ្វីមួយ ឬបញ្ចេញលទ្ធផល។ យើងអាចសម្គាល់វាបានដោយខាងស្តាំរបស់វាមានចន្លោះសម្រាប់ភ្ជាប់ Block តម្លៃ និងពាក្យ set ឬ get នៅពីមុខ។



- ***៣.ប្លុកតម្លៃ៖*** ជា Block ដែលប្រាប់ពីតម្លៃ ឬអក្សរអ្វីមួយ។ យើងអាចសម្គាល់វាបានដោយខាងឆ្វេងរបស់វាមានធ្មេញសម្រាប់ភ្ជាប់ទៅនឹង Block អនុគមន៍ ហើយខាងស្តាំរបស់វាទទេ ដោយទាញវាមកប្រើពី **text** ក្នុង Block Build_in។

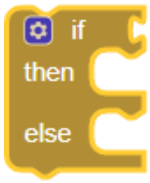
- ***៤.ប្លុកកន្សោមគត្តុ៖*** ប្លុកសម្រាប់បង្កើតកន្សោមលក្ខខណ្ឌពេលប្រើប្លុក If ដោយទាញវាមកប្រើពី **logic** ក្នុង Block Build_in:

- ***៥.ប្លុកឈ្មោះ៖*** គឺជាឈ្មោះផ្សំពីប៊ូតុង Button1 ជាមួយនឹង Text ឬជាតុផ្សេងៗទៀតដូចរូបខាងក្រោម និងប្រើវាដូចជាអថេរមួយសម្រាប់ផ្ទុកតម្លៃ និង អក្សរ។ យើងអាចទាញយកមកប្រើដោយចុចលើ Button1 និងជ្រើសរើសវា។



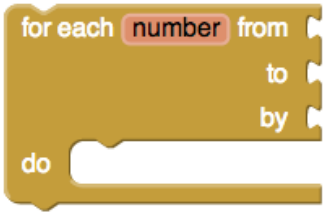
- ***ប្លុកលក្ខខណ្ឌ if៖*** គឺជាប្លុកលក្ខខណ្ឌដែលប្រើលក្ខខណ្ឌមួយបើសិនប្លុកកន្សោមរបស់ if ពិតនោះអនុវត្តប្លុកកូដនៅ then។

- ***ប្លុកលក្ខខណ្ឌ if...else៖*** គឺជាប្លុកលក្ខខណ្ឌដែលប្រើលក្ខខណ្ឌពីរដូចដែលយើងប្រើ if...else statement នៅក្នុងភាសា C ឬ Assembly ដែរ។ បើសិនប្លុកកន្សោមរបស់ if ពិតនោះអនុវត្តប្លុកកូដនៅ then និង ម៉្យាងទៀតអនុវត្តប្លុកកូដនៅ Else។



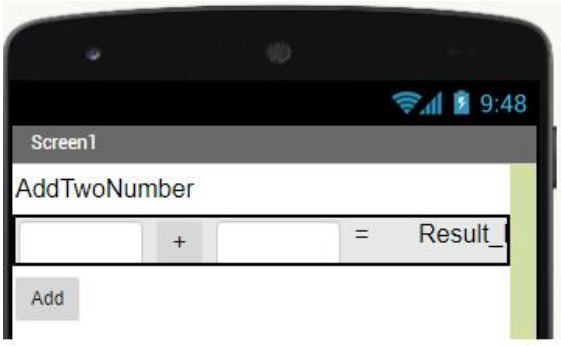
- ***ប្លុកប្រកាសអថេ៖*** ជាប្លុកមួយប្រើសម្រាប់ប្រកាសអថេរប្រើសម្រាប់ផ្ទុកតម្លៃ និងចាប់យកវាពី Variable ក្នុង Block Build_in។

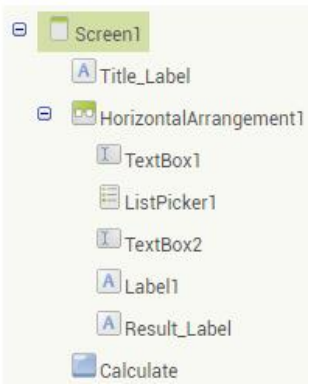
- ***ប្លុករង្វិលជុំ(Loop)៖*** ប្រើសម្រាប់ដំណើរការរង្វិលជុំដែលប្រើវានៅក្នុងចំនុចខាងក្រោម នៃមេរៀននេះ។ តាមរូបខាងក្រោមគឺជាប្លុករង្វិលជុំ និងដំណើរការប្លុកនៅក្នុងផ្នែកសម្រាប់ធ្វើប្រមាណវិធីរកតម្លៃជាលេខនីមួយៗនៅក្នុងអថេរ number ដោយចាប់ផ្តើមតម្លៃពី from និងបញ្ចប់នៅ to និងបង្កើន number តម្លៃរាល់ពេល។ ប្រើឈ្មោះអថេរ number ដែលបានផ្តល់ដើម្បីផ្ទុកទៅលើតម្លៃ។ អ្នកអាចប្តូរឈ្មោះអថេរ number ទៅជាអ្វីផ្សេងប្រសិនបើអ្នកចង់។



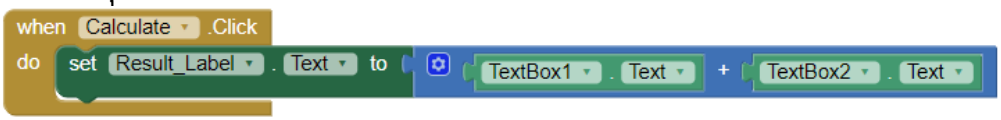
- ***ប្លុក Array និងគណិតវិទ្យា៖*** គឺប្លុកប្រកាសសំនុំធាតុនៃ Arrays និងយកវាមកប្រើពី។ ប្លុកគណិតសម្រាប់ប្រើជាលេខ ឬកន្សោមគណិតវិទ្យា៖

4. <u>របៀបបង្កើតកម្មវិធីបូកលេខពីរចំនួន</u>: ដើម្បីបង្កើតកម្មវិធីមួយបូកពីរចំនួន ត្រូវបង្កើតTextBox ពីរសម្រាប់បញ្ចូលតម្លៃ និងលទ្ធផលផ្ទុកនៅក្នុង Result_Label និង ListPicker សម្រាប់ដាក់សញ្ញា "+" និងប៊ូតុងមួយទៀតប្រើម្រាប់ចុចឱ្យបូកលេខ។ ចូរបង្កើត Interface ឱ្យបានដូចរូបរូបខាងក្រោម៖





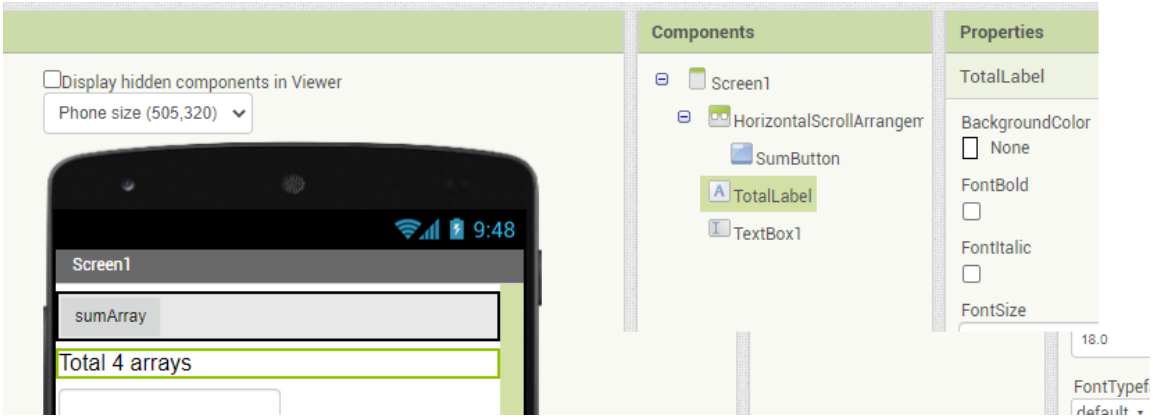
1. ចុច Mouse លើ **Screen1** និងជ្រើសយក **Label** ពី User Interface
2. ចុចលើ **Rename** នៅក្នុងផ្ទៃ Components ប្តូរឈ្មោះទៅ Title_Label
3. ចុចលើ Title_Label និងត្រង់ text នៃ Properties វាយឈ្មោះ AddTwoNumber
4. ចុចលើ **Layout** នៃ Palette និងជ្រើសយក **HorizontalArrangement1**
5. ត្រង់ User Interface ចុចលើ **TextBox** និង **ListPicker** រួចប្តូរឈ្មោះទៅជាសញ្ញា "+"
6. ត្រង់ User Interface ចុចលើ **TextBox** និង **Label** រួចប្តូរឈ្មោះទៅ "**Result_Label**"
7. ចុចលើ **Button** និង **Rename** ប្តូរឈ្មោះទៅជា "Calculate"
8. ត្រង់ **Text** នៃ Properties ដាក់ឈ្មោះ Add
9. ចូរបង្កើតប្លុកកូដតាមរូបខាងក្រោម៖



Output Screen

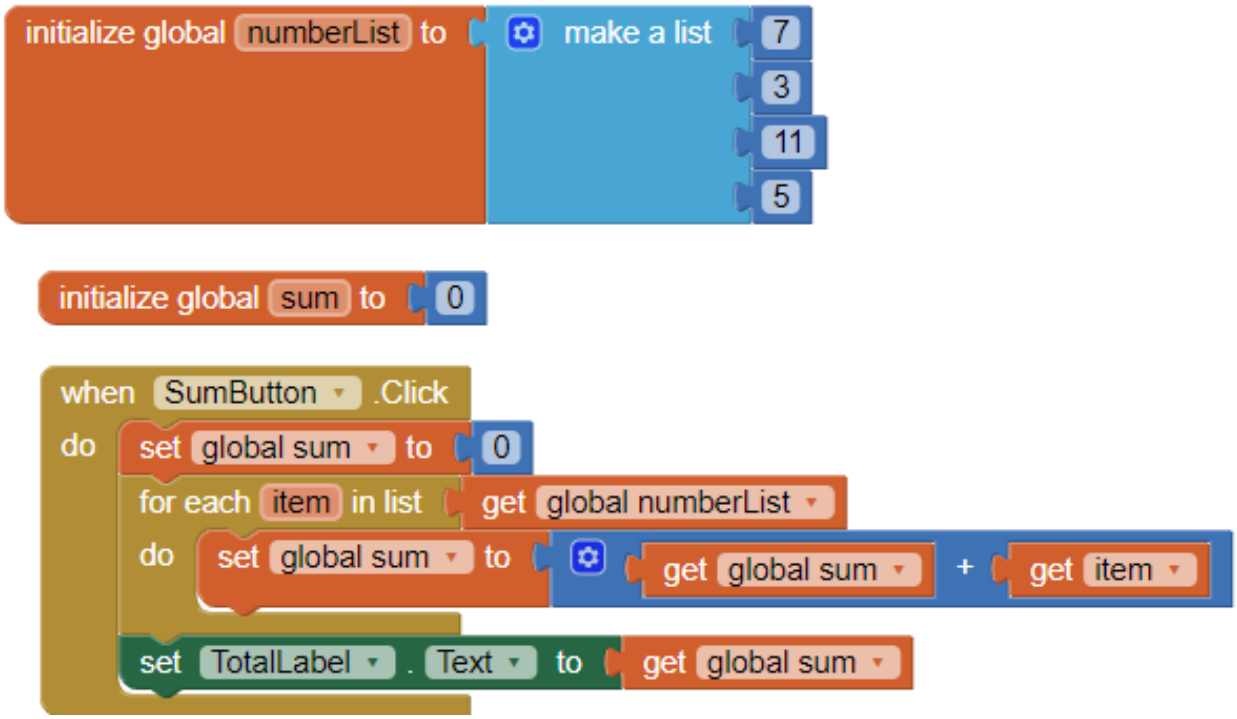
5. រៀបបង្កើតកម្មវិធីធ្វើប្រមាណវិធី Array

1. ចូរបង្កើត Interface តាមរូបដូចខាងក្រោម៖

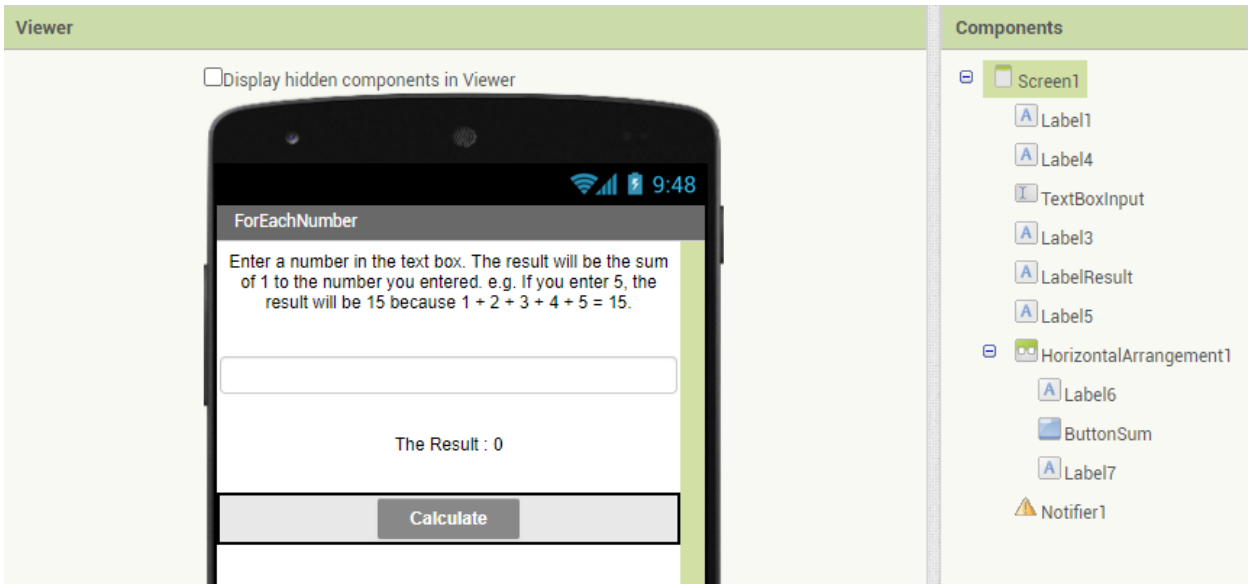


2. បង្កើតប្លុកកូដដោយអនុវត្តតាមជំហានដូចខាងក្រោម៖

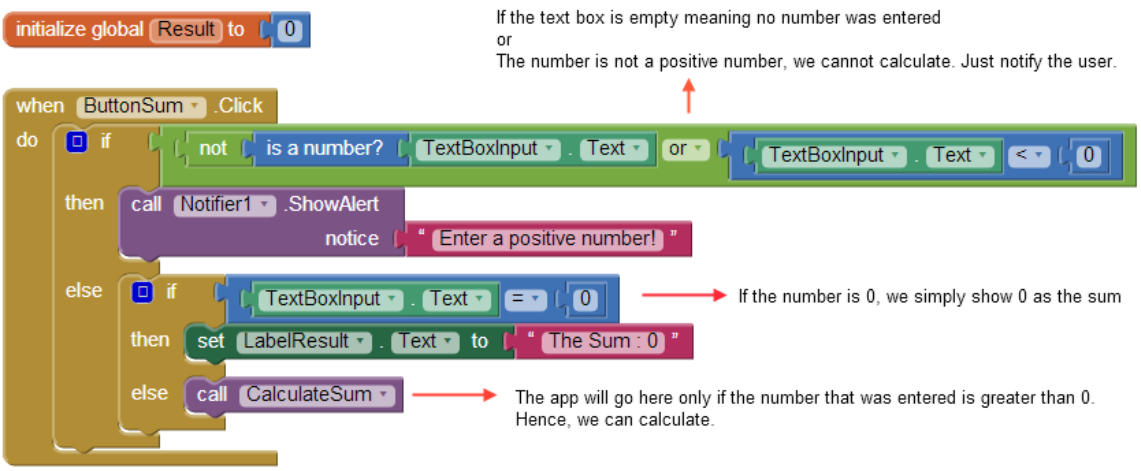
1. ប្រកាស Array ឈ្មោះ **initialize global "numberList"** និង **make a list :7, 3, 11, 5** និងក្នុងករណីចង់បន្ថែមធាតុអារេចុច Mouse ស្តាំលើសញ្ញាផ្កាយ និងទាញយក item ចូលក្នុង list
2. ប្រកាស Array ឈ្មោះ **initialize global "sum=0"**
3. ប្រើប្លុក for each "**item**" in list និងដំណើរការ Loop ទៅតាមចំនួនធាតុដែលមាននៅក្នុង List និងធាតុនិមួយៗ ដែលត្រូវធ្វើប្រមាណវិធីចាប់យកពីអារេ **numberList**
4. ដំណើរការ Loop ដំបូងគឺ "item =7 " រួចការបូកលេខ sum = sum + item=0+7 ដោយប្លុកតាមរូប ខាងក្រោម
5. និងដំណើរការរហូតដល់ "item=5" ជាធាតុចុងក្រោយនៃ Loop



6. របៀបបង្កើតកម្មវិធីធ្វើប្រមាណវិធីស្វ័តនៅលើ Interface ដូចខាងក្រោម:

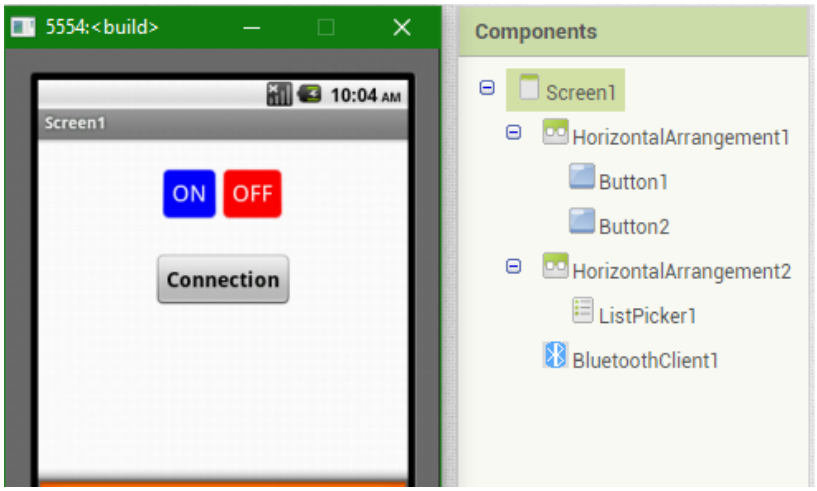


2. បង្កើតប្លុកកូដ



7. របៀបបង្កើតកម្មវិធីធ្វើការបិទ ឬបើកអំពូល

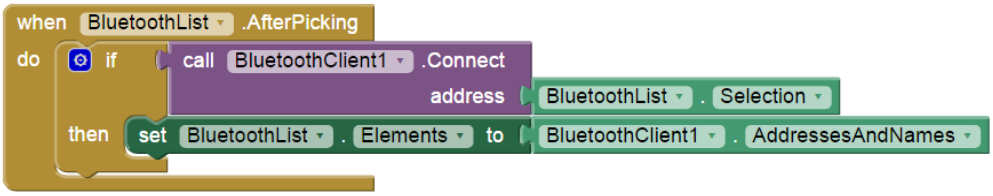
1. ចូរបង្កើត Interface ដូចខាងក្រោម៖



2. បង្កើតប្លុកកូដ

2.1. យើងនឹងចាប់ផ្តើមជាមួយនឹងប្លុក **when ListPicker2.BeforePicking**។ ដំបូងយើងនឹងបន្ថែមក្នុងប្លុក "BeforePicking" ជាមួយនឹងទៅប្លុក **'set Bluetooth Elements'** និងភ្ជាប់វាទៅ **"AddressesAndNames BluetoothClient"**។ នេះគឺជាការកំណត់ពីអំពីសំនុំឈ្មោះ និង លេខ អាសយដ្ឋានរបស់ឧបករណ៍ប្លូធូសដែលត្រូវបាន Paired រួចទៅហើយជាមួយនឹងទូរស័ព្ទរបស់យើង។ ដូច្នេះនៅពេលដែលយើងចុចលើ ListPicker គឺ " Connect " សំនុំឈ្មោះ និង លេខ Address ឧបករណ៍ប្លូធូសដែលបានភ្ជាប់ទាំងអស់នឹងបង្ហាញឡើងនៅលើទូរសព្ទដៃដើម្បីឱ្យយើងធ្វើការជ្រើសរើស។

2.2. បន្ទាប់យើងបានកំណត់នូវអ្វីដែលនឹងកើតឡើង យើងនឹងយកឬជ្រើសរើសម៉ូឌុល Bluetooth។ នៅលើប្លុក BluetoothClient យើងនឹងបន្ថែមប្លុក 'call BluetoothClient .Connect address' និងបន្ថែមប្លុក 'BluetoothList Selection' ទៅវា។ មានន័យថាទូរស័ព្ទរបស់យើងនឹងភ្ជាប់ទៅអាសយដ្ឋានប្លូធូសដែលយើងបានជ្រើសរើសពីមុនដោយបង្កើតប្លុកកូតដូចខាងក្រោម៖

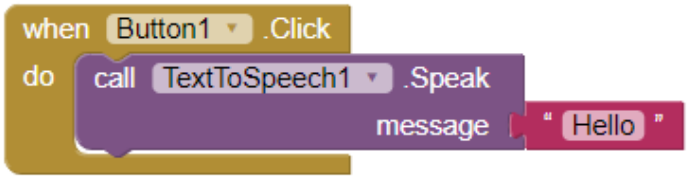


8. បង្កើតឧបករណ៍ប្រើសំលេច

ឧទាហរណ៍៖ បន្តពីឧទាហរណ៍ក្នុងផ្នែក "របៀបរចនា" ដែលយើងបានបង្កើតប៊ូតុងមួយនៅកណ្តាលអេក្រង់ទូរស័ព្ទ ប្រសិនបើអ្នកចង់ឲ្យទូរស័ព្ទអ្នកនិយាយពាក្យ "Hello" នៅពេលអ្នកចុចប៊ូតុងនោះ អ្នកត្រូវធ្វើតាមជំហានដូចខាងក្រោម៖

- **ជំហានទី១៖** ដោយសារតែមុខងារនិយាយត្រូវការ Component ឈ្មោះ TextToSpeech អ្នកត្រូវបន្ថែមវាចូលកម្មវិធីរបស់អ្នក។ វាស្ថិតនៅក្នុងផ្នែក Media របស់ផ្ទាំង Palette ក្នុង Designer Editor។
- **ជំហានទី២៖** កំណត់ព្រឹត្តិការណ៍។ ត្រឡប់មក Blocks Editor វិញ អ្នកត្រូវចុចលើ Button1 ដែលស្ថិតក្នុងផ្ទាំង Blocks ហើយជ្រើសរើសយក Block ព្រឹត្តិការណ៍ដែលសរសេរថា "when Button1.Click" មកដាក់ក្នុងផ្ទាំង Viewer។
- **ជំហានទី៣៖** កំណត់សកម្មភាព។ ដើម្បីឲ្យមានសម្លេងនិយាយ អ្នកត្រូវចុចលើ TextToSpeech1 ក្នុងផ្ទាំង Blocks ហើយជ្រើសយកប្លុកសកម្មភាពដែលសរសេរថា "call TextToSpeech1.Speak message" មកភ្ជាប់ជាមួយ Block ព្រឹត្តិការណ៍ខាងលើ។
- **ជំហានទី៤៖** កំណត់តម្លៃ។ អ្នកត្រូវប្រាប់ TextToSpeech1 អំពីពាក្យដែលត្រូវនិយាយ។ ដូច្នេះអ្នកត្រូវចុចលើ Text ដែលជា Built-in Block ហើយជ្រើសយក Block ទទេ " " មកភ្ជាប់នឹង Block សកម្មភាព។ បន្ទាប់មកអ្នកត្រូវបំពេញពាក្យថា Hello ចូលក្នុងចន្លោះនោះ។

បន្ទាប់ពីធ្វើតាមជំហានខាងលើ អ្នកនឹងបានបណ្តុំ Blocks មួយដូចខាងក្រោម



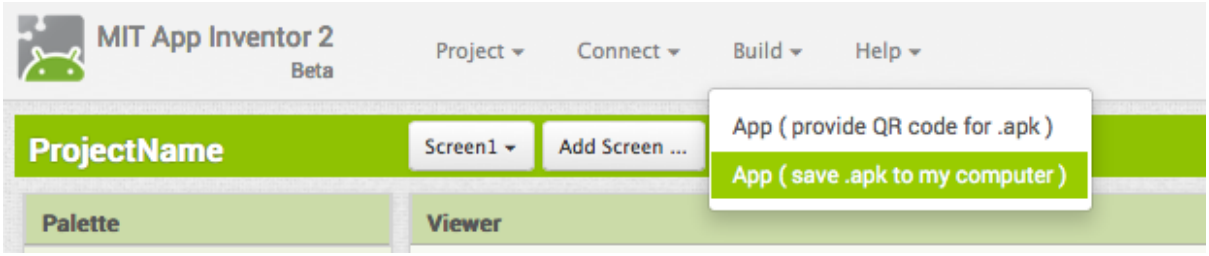
វាមានន័យថា"នៅពេលដែល Button1 ត្រូវបានចុចហៅ TextToSpeech1 ឲ្យនិយាយពាក្យ Hello"។

ក្រោយមក អ្នកអាចធ្វើការតេស្តកម្មវិធីនេះដើម្បីដឹងពីដំណើរការដែលអ្នកបានកំណត់។

អបអរសាទរ! អ្នកបានបង្កើតកម្មវិធីទីមួយរបស់អ្នកជាមួយ App Inventor ដោយជោគជ័យ!

9. រៀបចេចខ្ចប់ជា APK

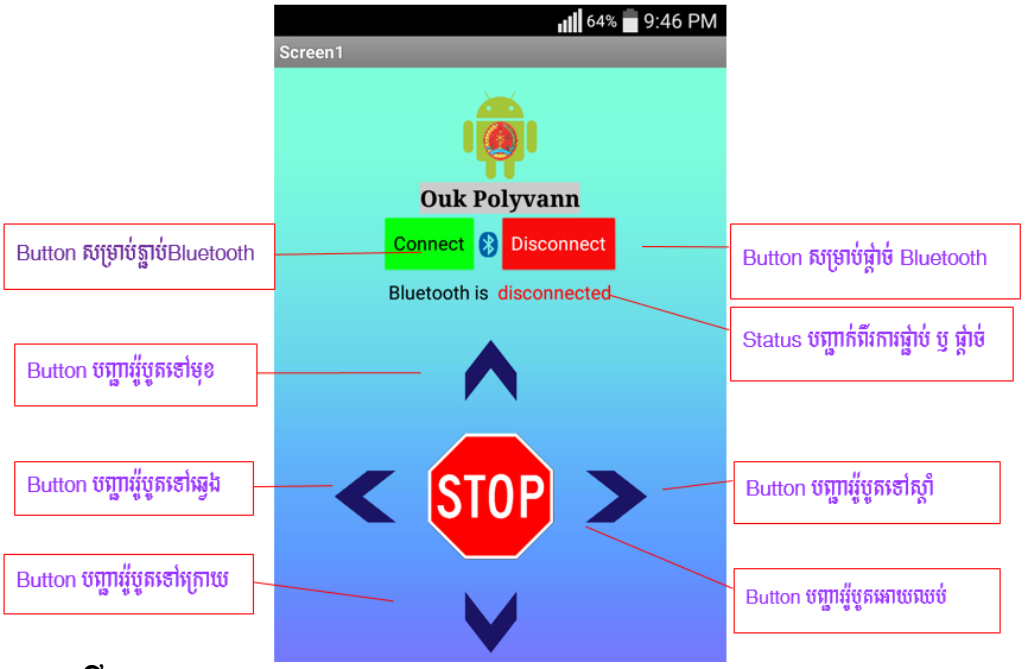
ជាចុងបញ្ចប់ ក្រោយពីអ្នកបានរចនា និងសរសេរកម្មវិធីរបស់អ្នករួចរាល់ហើយ អ្នកអាចវេចខ្ចប់គម្រោងរបស់អ្នកជាកញ្ចប់ APK ដែលអាចឲ្យអ្នកដទៃយកទៅដំឡើង និងប្រើប្រាស់បាន។ ការខ្ចប់គឺងាយស្រួលទេ ដោយអ្នកគ្រាន់តែចុចលើប៊ូតុង Build ហើយជ្រើសយក "App (save .apk to my computer)" នៅក្នុងគម្រោងរបស់អ្នក។



TIP ទូរស័ព្ទ Android ហាមមិនឲ្យអ្នកប្រើប្រាស់ដំឡើងកម្មវិធីក្រៅពី Play Store។ ដូច្នេះដើម្បីដំឡើងពី APK file អ្នកត្រូវចូលទៅកាន់ "Setting > Lock screen and Security" នៅលើទូរស័ព្ទរបស់អ្នក ហើយធីកលើ "Unknown Sources"។

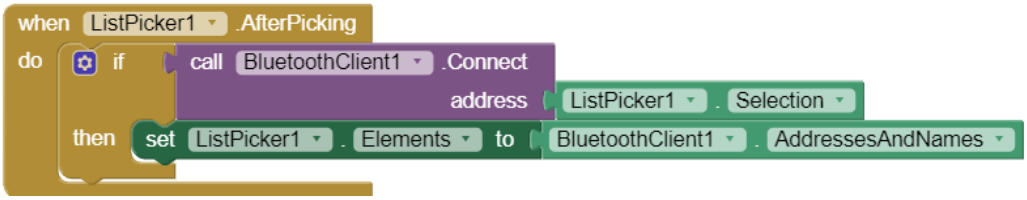
10. របៀបបង្កើតកម្មវិធីធ្វើការបញ្ជាឆៅលើ RC car

10.1. ការបង្កើត Interface ខាងក្រោមដើម្បីបញ្ជារ៉ូបូតទៅមុខ ក្រោយ ឆ្វេង និង ស្តាំ

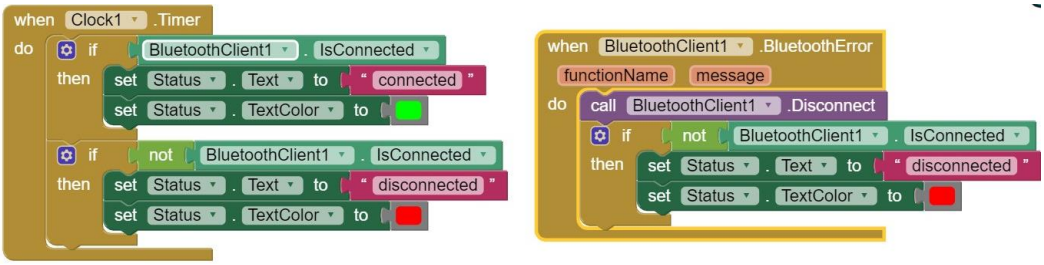


10.2. ការប្រើប្លុកកូដ

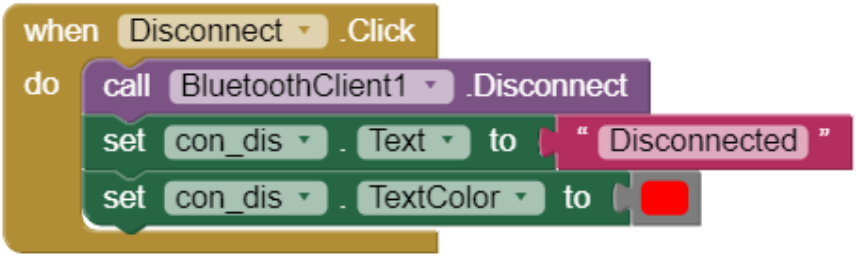
10.2.1. ប្លុកកូដខាងបង្ហាញពីការតភ្ជាប់រវាង Bluetooth នៃទូរសព្ទជាមួយនឹង Bluetooth Arduino។ នៅពេលយើងចុចលើ **Button connect** តាមរូបខាងលើវាបង្ហាញលេខអាសយដ្ឋានឱ្យជ្រើសរើសហើយចុចលើអាសយដ្ឋាននោះដើម្បីភ្ជាប់ទៅកាន់អាសយដ្ឋានរបស់ Bluetooth RC car។



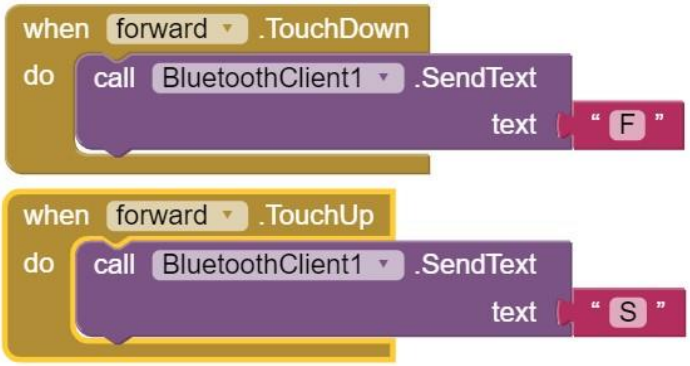
10.2.2. ប្លុកកូដខាងក្រោមបង្ហាញនៅពេលដែលចុចពាក្យ connected បានសម្រេច status របស់វានឹងបង្ហាញថា connected បើជាមិនទាន់បានភ្ជាប់នោះទេ status របស់វានឹងបង្ហាញថា disconnected។



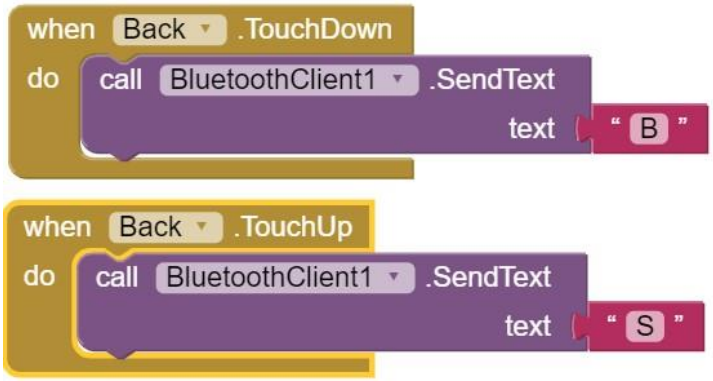
10.2.3. នៅពេលបានភ្ជាប់បានសម្រេច ហើយយើងចង់ផ្តាច់វិញយើងត្រូវចុច Button disconnect



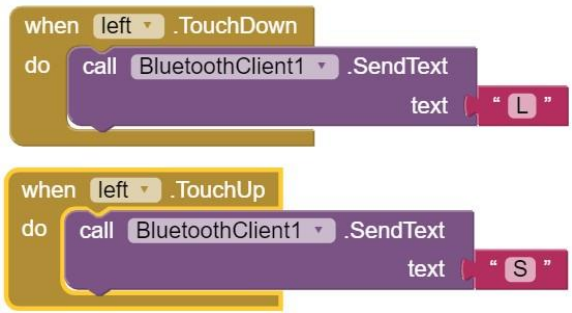
10.2.4. ប្លុកកូដខាងក្រោមសម្រាប់ដំណើរការនៅពេលយើងនៅយើងចុចនៅលើ button Forward នៅលើកម្មវិធីខាងលើនោះបញ្ជូលអក្សរ F ទៅកាន់ Bluetooth និងបន្ទាប់មកបញ្ជូលវាទៅកាន់ Arduino ប៉ុតែដើម្បីធ្វើការប្រតិបត្តិកូដឱ្យរ៉ូបូតទៅមុខ។



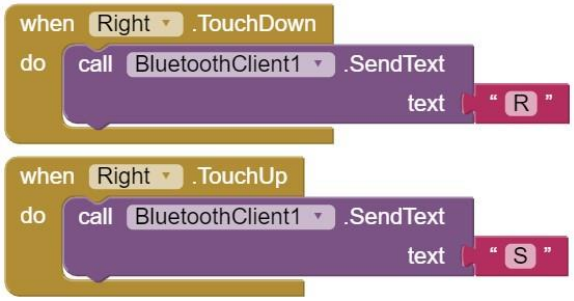
10.2.5. ប្លុកកូដខាងក្រោមបង្ហាញក៏ដូចនៅចំនុច**10.2.4.** ដោយគ្រាន់ត្រង់ដាក់អក្សរ Bនៅពេលដែល យើងចុច button Back នោះនិងបញ្ជូលតម្លៃ B តាមរយៈ Bluetooth ទៅ Arduino។



10.2.6. ប្លុកកូដខាងក្រោមបង្ហាញក៏ដូចនៅចំនុច**10.2.4.** ដោយគ្រាន់ត្រង់ដាក់អក្សរ Lនៅពេលដែលយើងចុច button Left នោះនិងបញ្ជូលតម្លៃ L តាមរយៈBluetoothទៅ Arduino។



10.2.7. ប្តូកកូដខាងក្រោមបង្ហាញក៏ដូចនៅចំនុច**១០.២.៤.** ដោយគ្រាន់ត្រង់ដាក់អក្សរ R នៅពេលដែល យើងចុច button Right នោះនិងបញ្ជូលតម្លៃ R តាមរយៈ Bluetooth ទៅ Arduino



សំណួរ និង លំហាត់

១.ដូចម្តេចដែលហៅថា App Inventor 2? គេប្រើវាសម្រាប់ធ្វើអ្វី?

២.ដូចម្តេចដែលហៅថា (a). Interface, (b).Procedure(c).OS android, (d).Palete

៣.ចូរពន្យល់ពី Controls(Widgets) និមួយដែលមានក្នុង User Interface និងលើកជាឧទាហរណ៍មកបញ្ជាក់តាមការប្រើ Controls និមួយៗ។

៤.ចូរពន្យល់ពីមុខងារធាតុនិមួយដែលមានក្នុង Properties និង Components?

៥. ចូរបង្កើត Mobile App និងមានសមត្ថភាពដូចកម្មវិធីទូរសព្ទដែលកំពុងប្រើ

a. Calculator

b. Clock

c. Calendar

៦. បង្កើត Mobile App ដើម្បី

5.១.បញ្ជារ៉ូបូតប្រភេទ RC car?

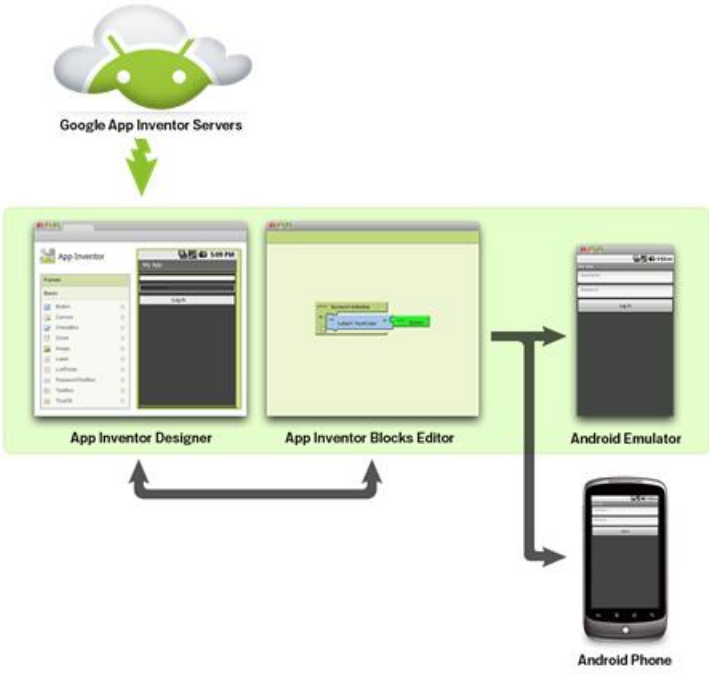
5.២.បញ្ជាគម្រោង Bluetooth Controlled Lamp?

5.៣.បញ្ជាគម្រោងស្រោចដំណាំស្វ័យប្រវត្តិ?

5.៤.បញ្ជាគម្រោងម៉ោងឌីជីថល? ដែលកំណត់ម៉ោង នាទី និងវិនាទីតាម Bluetooth?

5.៥.បញ្ជាគម្រោង**SIM800L ការប្រើស្វ័យប្រវត្តិកម្មក្នុងផ្ទះ**

៧. ចូរពន្យល់ដំណើរការនៃការបង្កើតកម្មវិធីនៅក្នុង Inventor តាមរូបខាងក្រោម៖



មេរៀនទី ១១ Internet of Thing

IOT មកពីពាក្យពេញ (Internet Of Things)មានន័យថាការធ្វើការបញ្ជា(Remote) ទៅឧបករណ៍អេឡិចត្រូនិច និងឧបករណ៍មួយចំនួនទៀតមានដូចជាកុំព្យូទ័រ, ទូរសព្ទ, Ipad ជាដើម ដោយប្រើប្រាស់នូវបច្ចេកវិទ្យាអ៊ីនធឺណែត(Internet)។ វាពិពណ៌នាអំពីបណ្តាញនៃវត្ថុរូបធាតុ "Things" ដែលត្រូវបានដំណើរការដោយសេនស័រ, ផ្នែកទន់និងបច្ចេកវិទ្យាផ្សេងទៀតក្នុងគោលបំណងតភ្ជាប់គ្នា និងផ្លាស់ប្តូរទិន្នន័យជាមួយឧបករណ៍និងប្រព័ន្ធដទៃទៀតនៅលើអ៊ីនធឺណែត។ វាមានមុខងារពិនិត្យតាមដានបេះដូង, ឧបករណ៍បញ្ជូនជីវជីបលើសត្វកសិដ្ឋាន, រថយន្តដែលមានភ្ជាប់មកជាមួយឧបករណ៍ចាប់សញ្ញាសែនស័រ ឧបករណ៍វិភាគឌីអិនអេ, គ្រប់គ្រងប្រព័ន្ធស្វ័យប្រវត្តិកម្មក្នុងផ្ទះ, គ្រប់គ្រងពីសីតុណ្ហភាព និងឧបករណ៍ពាក់ផ្សេងទៀតដូចមាននៅក្នុងរូបខាងក្រោម៖

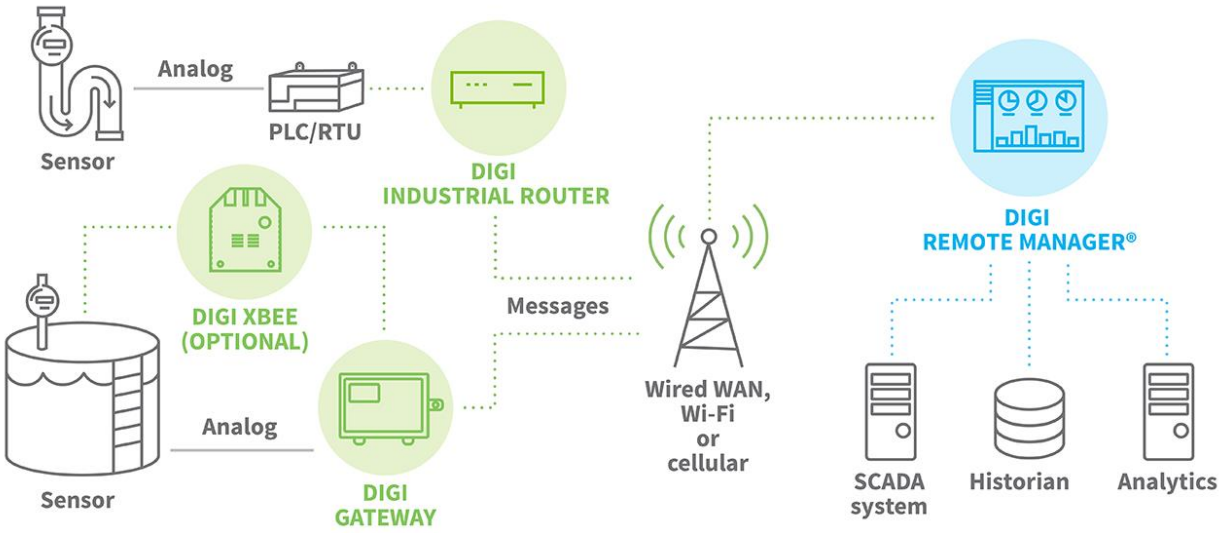
1. From M2M to IOT

ដំបូងវាជាការសំខាន់ដែលត្រូវដឹងថា IoT ជាមុនសិន វាគឺជាផ្នែករងនៃបច្ចេកវិទ្យា M2M។ អាយអូធីពាក់ព័ន្ធនឹងការទំនាក់ទំនងរវាងម៉ាស៊ីនដោយគ្មានការបញ្ចូលរបស់មនុស្សធ្វើឱ្យវានោះជាទម្រង់ នៃការប្រាស្រ័យទាក់ទង M2M។ ទោះយ៉ាងណាអាយអូធីពង្រីកថាមពលនិងសក្តានុពលនៃបច្ចេកវិទ្យា M2M តាមមធ្យោបាយថ្មី។ ភាពខុសគ្នាធំបំផុតរវាង M2M និង IoT គឺថាប្រព័ន្ធ M2M ប្រើទំនាក់ទំនងពីចំណុចមួយទៅចំណុចមួយ។ ប្រព័ន្ធ IoT ជាធម្មតាស្ថិតនៅលើឧបករណ៍របស់ខ្លួននៅក្នុងបណ្តាញណែកវើកជាសាកលដែលអនុញ្ញាតឱ្យមានសមាហរណកម្មខ្នាតធំនិងកម្មវិធីទំនើបជាងមុន។ ការធ្វើមាត្រដ្ឋានបានគឺជាភាពខុសគ្នាសំខាន់មួយទៀតរវាង M2M និងអាយតូ។ អាយអូធីត្រូវបានរចនាឡើងដើម្បីអាចធ្វើមាត្រដ្ឋានបានខ្ពស់ព្រោះថាជារឿយៗឧបករណ៍អាចត្រូវបានបញ្ចូលទៅក្នុងបណ្តាញហើយបញ្ចូលទៅក្នុងប្រព័ន្ធដែលមានស្រាប់ជាមួយនឹងការរំខានតិចតួចបំផុត។ បណ្តាញ M2M ក៏អាចមានកម្លាំងពលកម្មច្រើនជាងមុនក្នុងការតំឡើងនិងថែរក្សាពីព្រោះការភ្ជាប់ពីចំណុចមួយទៅចំណុចថ្មីត្រូវតែបង្កើតសម្រាប់ឧបករណ៍នីមួយៗ។

2. **ស្ថាបត្យកម្មអាយអូធី** (IoT Architecture)

ស្ថាបត្យកម្មប្រព័ន្ធអាយអូធីត្រូវបានពិពណ៌នាជារឿយៗថា គឺជាដំណើរការដោយមានបួនជំហានដែលទិន្នន័យហូរចេញពីឧបករណ៍ចាប់សញ្ញាភ្ជាប់ជាមួយ "វត្ថុ (thing)" តាមរយៈបណ្តាញណែតវើកហើយទីបំផុតទៅ

មជ្ឈមណ្ឌលទិន្នន័យសាជីវកម្មឬ Cloud សម្រាប់ដំណើរការវិភាគ និងផ្ទុក។ នៅក្នុងអាយអូធី " វត្ថុ" អាចជាម៉ាស៊ីនអាគារ ឬក៏មនុស្សម្នាក់។ ដំណើរការនៅក្នុងស្ថាបត្យកម្មអាយអូធីក៏បញ្ជូនទិន្នន័យតាមទិសដៅផ្សេងទៀតតាមទំរង់នៃសេចក្តីណែនាំ ឬបញ្ជាដែលប្រាប់អោយយើងធ្វើសកម្មភាពឬឧបករណ៍ដែលទាក់ទងនឹងរាងកាយផ្សេងទៀតដើម្បីចាត់វិធានការមួយចំនួនដើម្បីគ្រប់គ្រងដំណើរការលើឧបករណ៍។ Actuator ធ្វើសកម្មភាពអាចធ្វើអ្វីមួយដែលសាមញ្ញដូចជាការបើកភ្លើង ឬលទ្ធផលដូចជាការបិទបណ្តាញមានការជួបបញ្ហាផ្សេងៗ។



2.1. **ឧទាហរណ៍ទី១**: Sensors និង Actuators

ដំណើរការចាប់ផ្តើមជាមួយឧបករណ៍ចាប់សញ្ញាសែនស័រ និងឧបករណ៍ធ្វើសកម្មភា(actuators) ធ្វើការភ្ជាប់ដែលម៉ូនីទ័រ (ក្នុងករណីឧបករណ៍ចាប់សញ្ញា) ឬ កុងត្រូល(ក្នុងករណី actuators) "វត្ថុ" ឬឧបករណ៍ដំណើរ។ ឧបករណ៍ចាប់សញ្ញាចាប់យកទិន្នន័យទាក់ទងនឹងស្ថានភាពនៃដំណើរការ ឬស្ថានភាពបរិស្ថានដូចជាសីតុណ្ហភាព, សំណើម, សមាសធាតុគីមី, កម្រិតសារធាតុរាវនៅក្នុងធុងមួយ, លំហូរសារធាតុរាវនៅក្នុងបំពង់មួយ ឬល្បឿននៃបណ្តុំសន្និបាតក៏ដូចជាករណីជាច្រើនទៀត។ ក្នុងករណីខ្លះឧបករណ៍ចាប់សញ្ញាអាចរកឃើញស្ថានភាព ឬព្រឹត្តិការណ៍ដែលត្រូវការការឆ្លើយតបស្ទើរតែភ្លាមៗដើម្បីឱ្យ actuator សកម្មភាពអាចអនុវត្តសកម្មភាពដោះស្រាយក្នុងពេលវេលាជាក់ស្តែងឧទាហរណ៍ការលៃតម្រូវអត្រាលំហូរសារធាតុរាវ ឬចលនារបស់មនុស្សយន្តឧស្សាហកម្ម។ ក្នុងស្ថានភាពទាំងនេះភាពយឺតយ៉ាវទាបរវាងឧបករណ៍ចាប់សញ្ញា និងការវិភាគ/ឧបករណ៍ធ្វើឱ្យសកម្មត្រូវបានទាមទារ។ ដើម្បីជៀសវាងការពន្យាពេលនៃដំណើរវិលជុំនៃទិន្នន័យទៅម៉ាស៊ីនមេការវិភាគទិន្នន័យដើម្បីកំណត់ការបរាជ័យនិងការបញ្ជូនវត្ថុបញ្ជាទៅ "វត្ថុ" ដំណើរការសំខាន់នេះត្រូវបានអនុវត្តនៅជិតនឹងដំណើរការដែលត្រូវបានត្រួតពិនិត្យឬត្រួតពិនិត្យ។ ដំណើរការ "គែម" នេះអាចត្រូវបានអនុវត្តដោយប្រព័ន្ធនៅលើឧបករណ៍ម៉ូឌុលដូចជាម៉ូឌុលឌីជី ConnectCore®និង / ឬឌីជីថលកោសិកាឌីជីជាមួយPython។

2.2. **ឧទាហរណ៍ទី២**: Internet Gateways និង Data Acquisition Systems

ប្រព័ន្ធទទួលទិន្នន័យ (DAS) ប្រមូលទិន្នន័យពីឧបករណ៍ចាប់សញ្ញាហើយបំលែងវាពីអាណាឡូកទៅជាទ្រង់ទ្រាយឌីជីថល។ បន្ទាប់មកប្រព័ន្ធទទួលទិន្នន័យធ្វើការប្រមូលផ្តុំ និងធ្វើអោយមានទ្រង់ទ្រាយទិន្នន័យមុនពេលបញ្ជូនវាតាមច្រកអ៊ីនធឺណែតតាមរយៈ WANs សម្រាប់ដំណាក់កាលបន្ទាប់នៃដំណើរការ។

នៅពេលនេះបរិមាណទិន្នន័យគឺនៅអតិបរមារបស់វា។ បរិមាណអាចមានទំហំធំធេងជាពិសេសឧទាហរណ៍នៅក្នុងរោងចក្រ ដែលរោងចក្រឧបករណ៍ចាប់សញ្ញារាប់រយអាចប្រមូលទិន្នន័យក្នុងពេលដំណាលគ្នា។ សម្រាប់ហេតុផលនោះទិន្នន័យក៏ត្រូវបានចងក្រង និងបង្រួមទៅនឹងទំហំតូចល្អបំផុតសម្រាប់ការបញ្ជូន។

2.3. ដំណាក់កាលទី៣: Pre-processing(Analytics at the Edge)

នៅពេលដែលទិន្នន័យ IoT ត្រូវបានគេឌីជីថល និងប្រមូលផ្តុំវានឹងត្រូវការដំណើរការដើម្បីកាត់បន្ថយបរិមាណទិន្នន័យបន្ថែមទៀតមុនពេលដែលវាទៅដល់មជ្ឈមណ្ឌលទិន្នន័យឬ Cloud។ ឧបករណ៍អាចអនុវត្តការវិភាគមួយចំនួនដែលជាផ្នែកនៃដំណើរការមុន។ ម៉ាស៊ីនសិក្សាអំពីដែលអាចមានប្រយោជន៍ខ្លាំងណាស់នៅដំណាក់កាលនេះ ដើម្បីផ្តល់នូវមតិយោបល់ទៅក្នុងប្រព័ន្ធនិងកែលម្អដំណើរការជាប្រចាំដោយមិនចាំបាច់រង់ចាំការណែនាំ ដើម្បីត្រលប់មកពីមជ្ឈមណ្ឌលទិន្នន័យសាជីវកម្ម ឬ Cloudទេ។ ដំណើរការនៃប្រភេទនេះជាទូទៅនឹងប្រព្រឹត្តទៅនៅលើឧបករណ៍មួយដែលស្ថិតនៅជិតទីតាំងដែលឧបករណ៍ចាប់សញ្ញាមានទីតាំងដូចជានៅក្នុងកន្លែងដាក់ខ្សែភ្លើងនៅនឹងកន្លែង។

2.4. ដំណាក់កាលទី ៤ : In-depth Analysis in the Cloud or Data Center

នៅដំណាក់កាលទី ៤ ក្នុងដំណើរការនេះប្រព័ន្ធព័ត៌មានវិទ្យាដែលមានអនុភាពអាចត្រូវបានគេយកមកធ្វើជាអ្នកវិភាគគ្រប់គ្រងនិង រក្សាទុកទិន្នន័យ។ នេះជាធម្មតាកើតឡើងនៅក្នុងមជ្ឈមណ្ឌលទិន្នន័យសាជីវកម្មឬនៅក្នុងពពកដែលទិន្នន័យពីតំបន់ / ឧបករណ៍ចាប់សញ្ញាជាច្រើនអាចត្រូវបានបញ្ចូលគ្នាដើម្បីផ្តល់នូវរូបភាពទូលំទូលាយនៃប្រព័ន្ធ IoT ទាំងមូលនិងផ្តល់នូវការយល់ដឹងដែលអាចធ្វើទៅបានទាំងអ្នកគ្រប់គ្រងផ្នែកព័ត៌មានវិទ្យានិងអាជីវកម្ម។ ក្រុមហ៊ុនមួយអាចមានប្រតិបត្តិការនៅក្នុងភូមិសាស្ត្រផ្សេងគ្នាហើយទិន្នន័យ IoT អាចត្រូវបានវិភាគដើម្បីកំណត់និន្នាការនិងលំនាំសំខាន់ៗឬរកចំណុចខ្វះចន្លោះ។

នៅកម្រិតនេះកម្មវិធីជាក់លាក់សម្រាប់ឧស្សាហកម្ម និងឬក្រុមហ៊ុនអាចត្រូវបានប្រើដើម្បីធ្វើការវិភាគស៊ីជម្រៅ និងអនុវត្តច្បាប់អាជីវកម្មដើម្បីកំណត់ថាតើត្រូវការសកម្មភាពណាមួយដែលត្រូវធ្វើ ទិន្នន័យចូលអាចបង្ហាញពីការផ្លាស់ប្តូរដែលចង់បាន ចំពោះការកំណត់ឧបករណ៍ឬមធ្យោបាយផ្សេងទៀតដើម្បីធ្វើឱ្យដំណើរការមានប្រសិទ្ធិភាពបង្កើតជារង្វិលជុំ ដែលសម្របសម្រួលការកែលម្អថេរ។ ដំណាក់កាលទី៤ ក៏រួមបញ្ចូលទាំងការផ្ទុកនៅក្នុងឃ្លាំងទិន្នន័យទាំងសម្រាប់រក្សាទុកកំណត់ត្រានិងសម្រាប់ការវិភាគបន្តទៀត។

3. IoT Ecosystem

បណ្តាញដ៏ទូលំទូលាយនៃឧបករណ៍ និងបច្ចេកវិទ្យាដែលមានការតភ្ជាប់គ្នា និងពឹងផ្អែកគ្នាដែលត្រូវបានអនុវត្តដោយអ្នកឯកទេសឆ្ពោះទៅគោលដៅជាក់លាក់មួយ ដូចជាការបង្កើតទីក្រុងឆ្លាត។

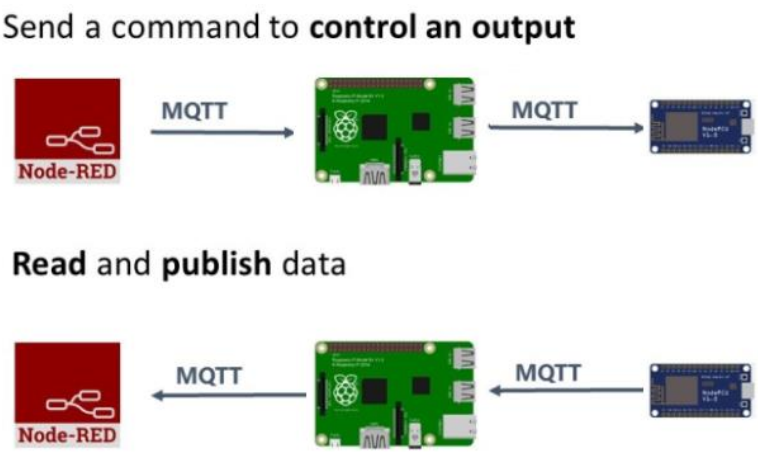
ជាក់ស្តែងមានកម្មវិធីគ្មានដែនកំណត់ចំពោះ IoT ហើយដូច្នេះយើងអាចនិយាយអំពីប្រព័ន្ធអេកូឡូស៊ី IoT ដែលគ្មានទីបញ្ចប់ ប៉ុន្តែប្រសិនបើយើងចង់ធ្វើនូវអ្វីដែលកំពុងកើតឡើងនៅក្នុងប្រព័ន្ធអេកូឡូស៊ីរហូតដល់ចំណុចសំខាន់អ្នកនឹងបង្កើតគ្រោងការណ៍សាមញ្ញមួយ គឺឧបករណ៍ប្រមូលទិន្នន័យហើយបញ្ជូនវាឆ្លងកាត់បណ្តាញទៅវេទិកា ដែលប្រមូលទិន្នន័យសម្រាប់ប្រើប្រាស់នាពេលអនាគតដោយភ្នាក់ងារ។ ដូច្នេះយើងមានសមាសធាតុសំខាន់ៗចំពោះប្រព័ន្ធអេកូអេកូះ ឧបករណ៍បណ្តាញវេទិកានិងភ្នាក់ងារ។

4. IoT protocols

ពិធីការ IoT គឺជាផ្នែកសំខាន់មួយនៃបច្ចេកវិទ្យា IoT stack - បើគ្មានពួកវាទេផ្នែករឹងនឹងមិនត្រូវបានប្រើទេ ព្រោះវាអនុញ្ញាតឱ្យវាផ្លាស់ប្តូរទិន្នន័យតាមលំដាប់លំដោយនិងមានអត្ថន័យច្បាស់លាស់។ ក្នុងចំណោមបំណែកនៃទិន្នន័យ(data)ដែលបានផ្ទេរទាំងនេះព័ត៌មានមានប្រយោជន៍អាចត្រូវបានស្រង់ចេញសម្រាប់អ្នកប្រើប្រាស់ចុងក្រោយ ហើយចំពោះការដាក់ពង្រាយទាំងមូលក្លាយជាផលចំណេញខាងសេដ្ឋកិច្ច។ ជាពិសេសទាក់ទងនឹងការគ្រប់គ្រងឧបករណ៍IoT។ នៅពេលនិយាយអំពី Internet of Things យើងតែងតែគិតអំពីទំនាក់ទំនង។ អន្តរកម្មរវាងឧបករណ៍ចាប់សញ្ញាឧបករណ៍ច្រកចេញចូលម៉ាស៊ីន មេនិងកម្មវិធីអ្នកប្រើគឺជាលក្ខណៈសំខាន់ដែលធ្វើឱ្យអ៊ិនធឺណែតនៃអ្វីៗជាអ្វី។ ប៉ុន្តែអ្វីដែលអាចឱ្យវត្ថុឆ្លាតទាំងអស់នេះអាចនិយាយ និងធ្វើអន្តរកម្មបានគឺពិធីការ IoT ដែលអាចត្រូវបានគេមើលឃើញថាជាភាសាដែលឧបករណ៍ IoT ប្រើដើម្បីទំនាក់ទំនង។

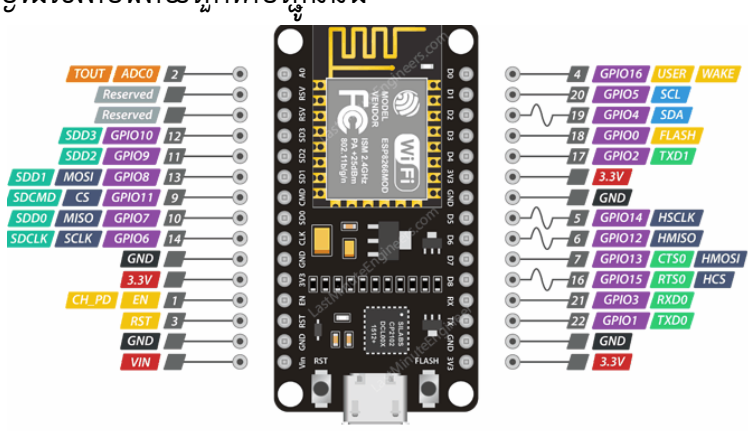
5. MQTT

MQTT គឺជាពិធីការផ្ញើសារដ៏សាមញ្ញមួយដែលត្រូវបានរចនាឡើងសម្រាប់ឧបករណ៍ដែលមានកម្រិតបញ្ជូនទាប។ ដូច្នេះវាគឺជាដំណោះស្រាយដ៏ល្អឥតខ្ចោះសម្រាប់កម្មវិធីអ៊ិនធឺណែតធីធី។ MQTT អនុញ្ញាតឱ្យអ្នកផ្ញើពាក្យបញ្ជាដើម្បីត្រួតពិនិត្យលទ្ធផលអាននិងបោះពុម្ពផ្សាយទិន្នន័យពីថ្នាំងឧបករណ៍ចាប់សញ្ញានិងអ្វីៗជាច្រើនទៀត។ ដូច្នេះវាពិតជាងាយស្រួលក្នុងការបង្កើតទំនាក់ទំនងរវាងឧបករណ៍ជាច្រើន។ ប្រសិនបើយើងចូលចិត្តស្វ័យប្រវត្តិកម្មនៅផ្ទះ ហើយអ្នកចង់បង្កើតប្រព័ន្ធស្វ័យប្រវត្តិកម្មផ្ទះពេញលេញតាមការស្វែងយល់បន្ថែម។



6. NodeMCU ESP8266

វាជាប្រភេទ Open source ដោយប្រើកម្មវិធីបង្កប់ដែលហៅថា Firmware និង ការអភិវឌ្ឍន៍ត្រូវបានកំណត់នូវ គោលដៅជាពិសេសសម្រាប់កម្មវិធីដែលមានមូលដ្ឋានលើការប្រើប្រាស់ IoT។ វាអាចបញ្ចូលកម្មវិធី បង្កប់ដែលដំណើរការលើប្រព័ន្ធ Esp8266 Wifi_SoC ពីប្រព័ន្ធ Espressif និងផ្នែករឹងដែលផ្អែកលើម៉ូឌុល ESP-12។ NodeMCU 8266 ប្រើស៊ីភីយូឈ្មោះ **ESP8266**(LX106) និងប្រព័ន្ធប្រភារ XTOS ដែលប្រភេទ Microcontrollers និងមានមុខងារប្រើដូច Arduino Uno ឬ Arduino Nano ដែរ។ ការបកប្រែពាក្យ "អ៊ិនធឺរណែតនៃអ្វីៗ" (អាយអូធី) បង្ហាញពីនិន្នាការមួយដែលឧបករណ៍បង្កប់មួយចំនួនធំប្រើសេវាកម្មទំនាក់ទំនងដែលត្រូវបានបង្កើតឡើងដោយពិធីសារអ៊ីនធឺណិត។ ភាគច្រើននៃឧបករណ៍ទាំងនេះដែលត្រូវបានគេហៅថាជាវត្ថុឆ្លាតវៃមិនត្រូវបានធ្វើប្រតិបត្តិការដោយមនុស្សដោយផ្ទាល់ទេ ប៉ុន្តែវាមានជាសមាសធាតុនៅក្នុងអគារ ឬយានយន្ត ឬត្រូវបានផ្សព្វផ្សាយនៅក្នុងបរិស្ថាន។ (ក្រុមប្រឹក្សាស្តាបត្យកម្មអ៊ិនធឺណិត) ហេដ្ឋារចនាសម្ព័ន្ធសកលសម្រាប់សង្គមព័ត៌មានដែលផ្តល់សេវាកម្មកម្រិតខ្ពស់តាមរយៈការភ្ជាប់ទំនាក់ទំនងគ្នា (រូបវន្តនិងនិម្មិត) ដោយផ្អែកលើបច្ចេកវិទ្យាព័ត៌មាននិងទំនាក់ទំនងអន្តរប្រតិបត្តិការដែលមានស្រាប់និងកំពុងវិវត្ត អ៊ិនធឺរណែត នៃឧបករណ៍កុំព្យូទ័រដែលបានបង្កប់នៅក្នុងវត្ថុប្រើប្រាស់ប្រចាំថ្ងៃដែលអាចអោយពួកគេបញ្ជូននិង



6.1. **ការប្រើប្រាស់ GPIO**(General Purpose input and output) **pin**

- NodeMCU ESP8266 ប្រើប្រាស់ GPIO Pin មានចំនួន ១៧ ដែលមានមុខងារសម្រាប់បញ្ចេញឬបញ្ចូលតម្លៃ 0 និង ១ និងពីនខ្លះទៀតអាចប្រើជា I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button។

- Power មានពីន GND ចំនួនបួន និង ពីនVcc ប្រើភ្លើង 3.3V ចំនួនពីរ

- Vin អាចផ្តល់ភ្លើង ៥ វុល និងអាចផ្តល់ថាមពលដល់ NodeMCU ទាំងមូលរវាង 7V-12V ដែលមានរហូតដល់ 1A។

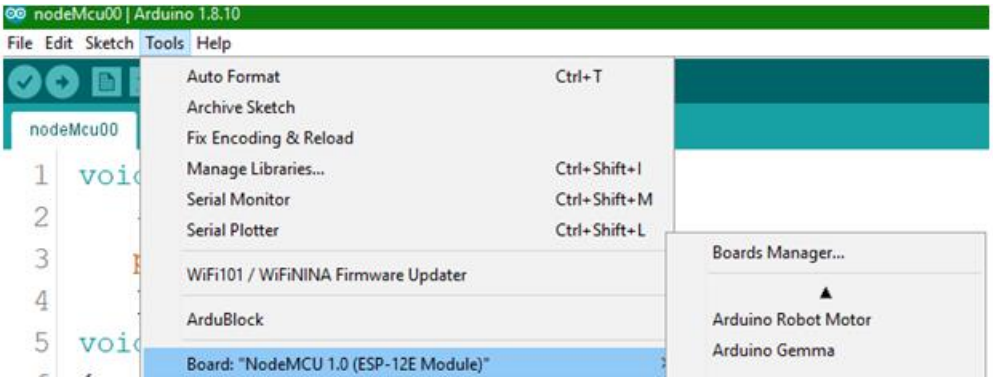
- នៅលើ Board មានជើងចាប់ពី D0 ដល់ D8 ប្រើសម្រាប់បញ្ចេញ ឬបញ្ចូលតម្លៃលេខ 0 និង ១។ ជើងពីនទាំងនេះប្រើសម្រាប់ភ្ជាប់ទៅឧបករណ៍ផ្សេងៗទៀតដូចជា LED, Sensor, LCD។ល។

6.2. **ការប្រើប្រាស់** *Led* **មួយជាមួយកម្មវិធី** *Arduino IDE*

យើងអាចប្រើអំពូលមួយនៅលើ NodeMCU ដែលមានស្រាប់សម្រាប់ការធ្វើតេស្តការងារដំបូង។ ដូចដែលបានរៀបរាប់នៅមេរៀនមុនដោយប្រើ Pin D0 ឬ Pin D4 ប្រើទៅតាមប្រភេទនៃ ប៉ត(Board)ដើម្បីឱ្យភ្លើងពណ៌ខៀវបើក ឬបិទ ដែលមាននៅលើប៉តស្រាប់។ ដើម្បីធ្វើការងារនេះចូលបើកកម្មវិធី Arduino IDE ត្រូវសរសេរកូដដូចបានបង្ហាញខាងក្រោមដោយប្រើ Pin D4។

```
void setup( )
  {
        pinMode( D4, OUTPUT ); //set D4 pin as output
  }
 void loop( )
  {
        digitalWrite( D4, HIGH );  //Turn On LED
        delay( 500 );  //delay one second
        digitalWrite( D4, LOW ); // Turn OFF LED
        delay( 500 ); } //delay one second
```

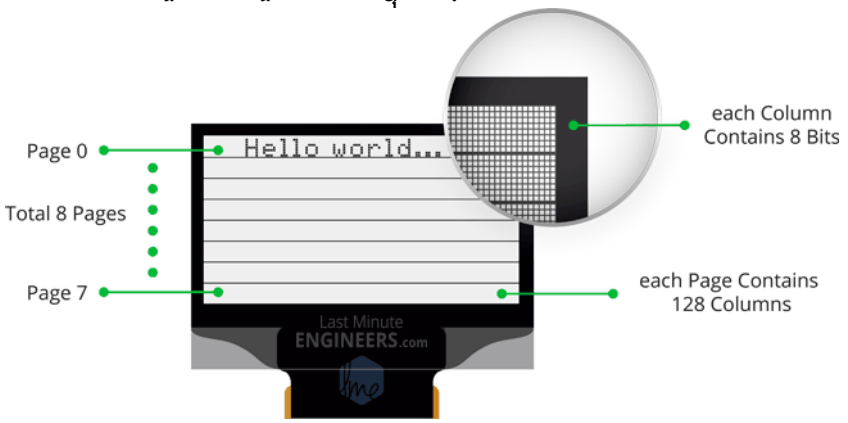
ក្រោយពីបានសរសេរកូដខាងលើរួច បន្ទាប់មកយក Mouse ចុចលើ **Tools** និងជ្រើសរើសយក Boards: **Node MCU 1.0 (ESP-12 Module)**។ សូមមើលតាមរូបខាងក្រោម និងបើមិនមានឈ្មោះ Board នេះទេយើងត្រូវបញ្ចូលវា និងជ្រើសរើស Serial Communication។



6.3. ការប្រើ OLED with ESP8266 NodeMCU

OLED ជាឧបករណ៍ធ្វើការបង្ហាញនូវព័ត៌មានដូចដែលយើងប្រើ LCD ដែរនៅមេរៀនមុន។ តែវាមានទំហំតូចជាង LCD ហើយមានជួរដេកនិង ជួរឈរច្រើនជាង។

OLED module មាន SSD1306 driver បានបង្កើតនៅក្នុងខ្លួនវាទំហំ 1KB GDDRAM។ វាសម្រាប់ធ្វើអេក្រង់ផ្ទុក bit patterm ដែលត្រូវបង្ហាញព័ត៌មាន។ មានផ្ទៃ Memory 1K ដោយបែងចែកទៅ 8 ទំព័រ (ពី 0 ទៅ 7) និង ទំព័រនិមួយៗមាន 128 columns/segments (block 0 ទៅ 127) និងអាចផ្ទុកទិន្នន័យ 8 bits (ពី0ទៅ7)មានជួរឈរនិមួយៗ អាចផ្ទុកទិន្នន័យ 8 bits (ពី0ទៅ 7)។

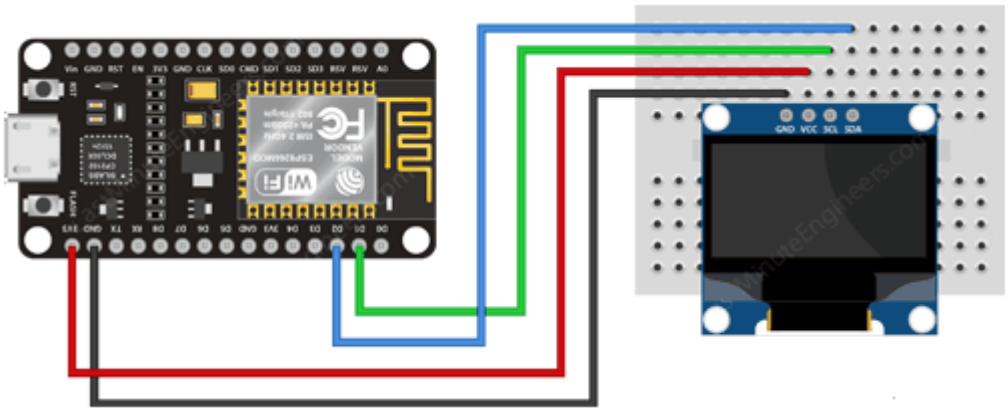


Displaying simple Text (Hello World)

```
// Display Text
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,28);
display.println("Hello world!");
display.display();
delay(2000);
```

6.4. ការតភ្ជាប់ OLED with ESP8266 NodeMCU

1. ធ្វើការតភ្ជាប់រវាង NodeMCU និង OLED តាមរូបខាងក្រោម៖



2. ចូរសរសេរកូដនៅ IDE Arduino

#include <SPI.h> ;បញ្ចូល Library ប្រើសម្រាប់ទំនាក់ទំនងជាមួយគ្រឿងកុំព្យូទ័រមួយឬច្រើន

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
Adafruit_SSD1306 display(-1);
void setup( )
{
        // initialize with the I2C addr 0x3C
        display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
        // Clear the buffer.
        display.clearDisplay( );
```

// Display Text
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,28);
	// Display ASCII Characters
	display.setCursor(0,24);
	display.setTextSize(2);
	display.write(3);

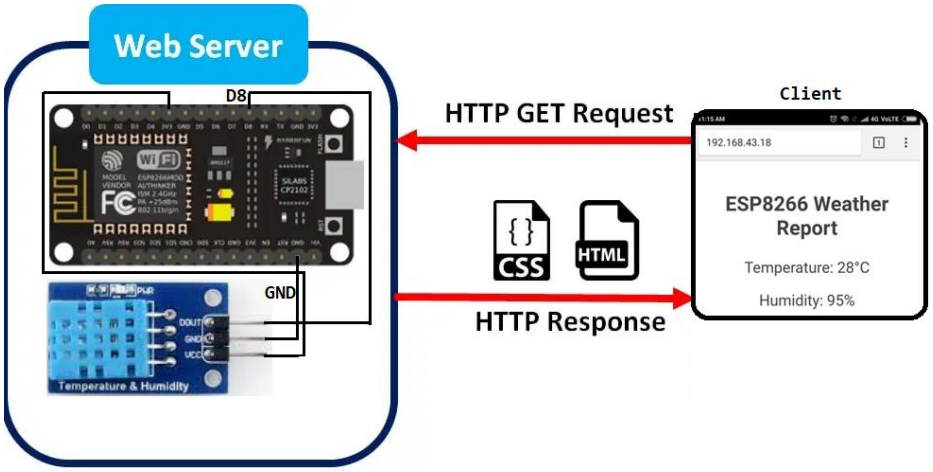
```
    display.println("Hello world!");
    display.display( );
    delay(2000);
    display.clearDisplay( );
    // Display Inverted Text
    display.setTextColor(BLACK,
    WHITE);
    display.setCursor(0,28);
    display.println("Hello world!");
    display.display( );
    delay(2000);
    display.clearDisplay( );
    // Changing Font Size
    display.setTextColor(WHITE);
    display.setCursor(0,24);
    display.setTextSize(2);
    display.println("Hello!");
    display.display( );
    delay(2000);
    display.clearDisplay( );
    // Display Numbers
    display.setTextSize(1);
    display.setCursor(0,28);
          display.println(123456789);
          display.display( );
          delay(2000);
display.clearDisplay( );
// Specifying Base For Numbers
display.setCursor(0,28);
display.print("0x")
display.print("(HEX) = ");
display.print(0xFF, DEC);
display.println("(DEC)");
display.display( );
```

```
display.display( );
delay(2000);
display.clearDisplay( );
// Scroll full screen
Display.setCursor(0,0);
display.setTextSize(1);
display.println("Full");
display.println("screen");
display.println("scrolling!");
display.display( );
display.startscrollright(0x00, 0x07);
delay(2000);
display.stopscroll( );
delay(1000);
display.startscrollleft(0x00, 0x07);
delay(2000);
display.stopscroll( );
delay(1000);
display.startscrolldiagright(0x00,
0x07);
delay(2000);
          display.startscrolldiagleft(0x00
, 0x07);
delay(2000);
display.stopscroll( );
display.clearDisplay( );
// Scroll part of the screen
display.setCursor(0,0);
display.setTextSize(1);
display.println("Scroll");
display.println("some part");
display.println("of the screen.");
display.display( );
display.startscrollright(0x00, 0x00);
}void loop( ) {}
```

7. ការប្រើសេនស័រ DHT11 ជាមួយនឹង NodeMCU ប្រើជា Web Server

នៅក្នុងគម្រោងនេះ យើងត្រូវបង្កើតឧបករណ៍មួយសម្រាប់ចាប់យកសីតុណ្ហភាពដោយប្រើឧបករណ៍ឈ្មោះ សេនស័រ DHT11 នៅក្នុងផ្ទះ និងទីកន្លែងណាមួយដើម្បីដឹងព័ត៌មានអំពីសីតុណ្ហភាព ដោយប្រើម៉ាស៊ីនមេ(Server) មួយ។ DHT11 ជាឧបករណ៍មួយប្រើប្រាស់សម្រាប់ចាប់យកទិន្នន័យជាប្រភេទសីតុណ្ហភាព និងសំណើមជាក់លាក់ទៅតាមទីតាំងដែលវាកំពុងស្ថិតនៅ។

នៅក្នុងគម្រោងនេះផងដែរយើងប្រើ ESP8266 NodeMCU ជាប្រភេទឧបករណ៍មានមុខងារដូចប៉ត Arduino សម្រាប់ធ្វើការត្រួតពិនិត្យដែលមានភាពងាយស្រួលភ្ជាប់ទៅ WiFi network និងបង្កើត Web Server មួយ។ ពីព្រោះនៅលើវាមានឧបករណ៍WiFi ស្រាប់ខុសពីប៉ត Arduino ដោយគ្មានឧបករណ៍នេះទេ។ នៅពេលយើងបានភ្ជាប់ឧបករណ៍នៃគម្រោងនេះ និងដំណើរការវាជាមួយនឹង web server ណាមួយដោយ NodeMCU រួមជាមួយនឹងឧបករណ៍ DHT11 ធ្វើការចាប់សីតុណ្ហភាព និងសំណើម រួចផ្ញើព័ត៌មានទៅឱ្យ web browser និងបន្ទាប់មកយើងអាចមើលឃើញព័ត៌មាននោះតាមរយៈកម្មវិធីនៅទូរស័ព្ទ ឬកុំព្យូទ័រដែលយើងហៅថាគេហទំព័រមួយដែលរូបបង្ហាញខាងក្រោម។



7.1. ផ្នែករឹង: ការដំឡើងផ្នែករឹងសូមមើលរូបខាងលើសំខាន់គឺប្រើជើង Pin8 នៃប៉តភ្ជាប់ទៅជើង OUT នៃ DHT11។

7.2. កម្មវិធី: កូដខាងក្រោមសរសេរឡើងបង្កើត Interface ដាក់ក្នុងកុំព្យូទ័រប្រើជា Client ដើម្បីធ្វើការទំនាក់ទំនងជាមួយនឹង Web Sever។

```
//Example Code
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include "DHT.h"
// Uncomment one of the lines below for whatever DHT sensor type you're using!
//#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT21   // DHT 21 (AM2301)
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
/*Put your SSID & Password*/
const char* ssid = "YourNetworkName";  // Enter SSID here
const char* password = "YourPassword";  //Enter Password here
```

```
ESP8266WebServer server(80);
// DHT Sensor
uint8_t DHTPin = D8;
// Initialize DHT sensor.
DHT dht(DHTPin, DHTTYPE);
float Temperature;
float Humidity;
void setup() {
    Serial.begin(115200);//Start the Serial communication to send messages
    //to the computer
    delay(100);// delay 1 second
    pinMode(DHTPin, INPUT);// sets DHTPin as Output
    dht.begin();   //
    Serial.println("Connecting to ");
    Serial.println(ssid);
    //connect to your local wi-fi network
    WiFi.begin(ssid, password);
    //check wi-fi is connected to wi-fi network
    while (WiFi.status() != WL_CONNECTED) {
    delay(1000); // delays one second
    Serial.print(".");}
    Serial.println("");Serial.println("WiFi connected..!");
    Serial.print("Got IP: ");  Serial.println(WiFi.localIP());
    server.on("/", handle_OnConnect);// Call the 'handleRoot' function when a
    //client requests URI "/"
    server.onNotFound(handle_NotFound);//
    server.begin();
    Serial.println("HTTP server started");}
void loop() {
    server.handleClient();// Listen for HTTP requests from clients
}
void handle_OnConnect() {
    Temperature = dht.readTemperature(); // Gets the values of the
    //temperature
     Humidity = dht.readHumidity();// Gets the values of the humidity
```

```
    server.send( 200, "text/html", SendHTML( Temperature,Humidity ) ); }
void handle_NotFound( ){
    server.send( 404, "text/plain", "Not found" );}// When a client requests an
    //unknown URI ( i.e. something other than "/" ), call function
    "handleNotFound"
String SendHTML( float Temperaturestat,float Humiditystat ){
    String ptr = "<!DOCTYPE html> <html>\n";
    ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-
    scale=1.0, user-scalable=no\">\n";
        ptr +="<title>ESP8266 Weather Report</title>\n";
        ptr +="<style>html { font-family: Helvetica; display: inline-block; margin:
        0px auto; text-align: center;}\n";
            ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px
            auto 30px;}\n";
            ptr +="p {font-size: 24px;color: #444444;margin-bottom: 10px;}\n";
        ptr +="</style>\n";
    ptr +="</head>\n";
    ptr +="<body>\n";
            ptr +="<div id=\"webpage\">\n";
            ptr +="<h1>ESP8266 NodeMCU Weather Report</h1>\n";
            ptr +="<p>Temperature: ";
            ptr +=( int )Temperaturestat;
            ptr +="°C</p>";
            ptr +="<p>Humidity: ";
            ptr +=( int )Humiditystat;
            ptr +="%</p>";
            ptr +="</div>\n";
    ptr +="</body>\n";
    ptr +="</html>\n";
    return ptr;}
```

7.3. OUTPUT

7.4. តាសា HTML: តំណាងឱ្យ Hyper Text Markup Language ដែលជាភាសាដែលត្រូវបានប្រើប្រាស់យ៉ាងទូលំទូលាយដើម្បីធ្វើការអភិវឌ្ឍគេហទំព័រ។ នៅក្នុងគម្រោងនេះយើងប្រើភាសានេះជាមួយនឹងភាសា Arduino ដើម្បីបង្កើតគេហទំព័រតូចមួយសម្រាប់បង្ហាញអំពីសីតុណ្ហភាព និងសំណើមដែលទាញមកពី Web Server។ កម្មវិធីខាងក្រោមសម្រាប់សរសេរក្នុងភាសា HDML ដែលជាគ្រឹះសម្រាប់ធ្វើការស្វែងយល់នូវអ្វីដែលយើងបានសរសេរកូដនៅខាងលើ ដោយប្រើអនុគមន៍ **SendHTML( )** និងអថេរ ptr+ = statement(s) ដែលបានប្រកាសក្នុងកម្មវិធី **String ptr = "<!DOCTYPE html> <html>\n";** ។

<!DOCTYPE html>// This tag defines the document type and HTML version.

<html>// This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.

<head>// This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.

<title>This is document title</title> The <title> tag is used inside the **<head>** tag to mention the document title.

</head>//end tap head

<body> This tag represents the document's body which keeps other HTML tags like <h1>, , etc.

<h1>This is a heading</h1>// This tag represents the heading.

** **Hello World! // This tag represents a paragraph.

</body>

</html>

- **String ptr = "<!DOCTYPE html> <html>\n";**//ប្រកាស <! DOCTYPE> ប្រាប់កម្មវិធី Brower ឱ្យដំណើរការកូដ HTML

- **ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";**// <meta>មានកំណត់ Viewport ដែលជាផ្ទៃ Window ដែលអាចមើលឃើញ និងជាតុ **width=device-width**:កំណត់ទទឹងទំព័រដើម្បីធ្វើតាមទទឹងអេក្រង់របស់ឧបករណ៍, **initial-scale=1.0**:កុងត្រូល Zoom Level **និង user-scalable=no** កំណត់ឱ្យគេហទំព័រមិនអាច Zoom បាន។

- Statements បន្តបន្ទាប់មើលការប្រើប្រាស់ភាសា HTML

- **ចូរធ្វើការ Update កូដខាងលើដោយមានចំុតុងពីរសម្រាប់ធ្វើការបិតឬបើកអំពូល**

8. ការគ្រប់គ្រងអំពូលតាមអ៊ុនធើណិត(Lamp Controller by Internet)

ការធ្វើគម្រោងនេះគឺមានការទាក់ទងទៅនឹងការគ្រប់គ្រងអំពូលឱ្យ បិទ ឬបើកពីទីតាំងណាក៏បានដូចជាការបញ្ជាអំពូលបិតឬបើកពីផ្ទះទៅសាលារៀនជាដើម។ គម្រោងនេះគឺគ្រាន់តែជាឧទាហរណ៍សម្រាប់បង្ហាញពីយល់ដឹង និងមានភាពងាយស្រួលនៃការប្រើប្រាស់បច្ចេកវិទ្យា IOT ប៉ុណ្ណោះ។ ជាក់ស្តែងយើងអាចធ្វើការបញ្ជាបានច្រើនទៀតដូចជា Remote Printer, Remote Router ឬ Access Point ជាដើម។

8.1. សមាសភាគដ៏ចាំបាច់ដូចជា៖

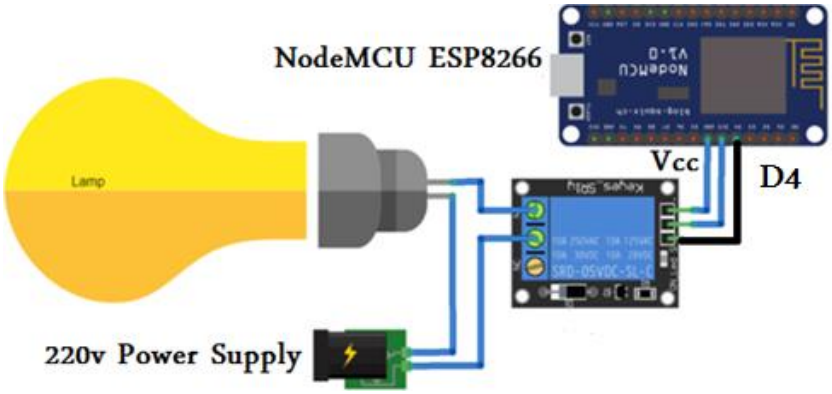
- Router ជាឧបករណ៍ **Network** មួយប្រភេទដែលអាចធ្វើការ Routing ទៅកាន់ **Internet** បាន។ វាជាផ្លូវចំលង (Default Gateway) របស់ប័ត Arduino ក្នុងការដំណើរការទៅកាន់ Internet បានតាមរយៈ **Public IP Address**។

- **Internet Cable** ជាខ្សែ **Network** សម្រាប់ភ្ជាប់ពី **Ethernet Shield** ទៅកាន់ **Router** ដើម្បីឱ្យ **Arduino** អាចធ្វើការប្រតិបត្តិទៅកាន់ **Network Resource** បាន។

8.2. ការប្រើប្រាស់ឧបករណ៍ និងតភ្ជាប់

8.3. ការតភ្ជាប់ Relay Module និងអំពូល

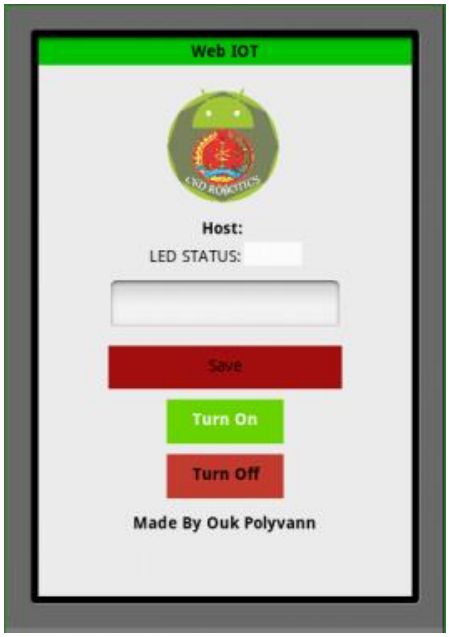
ជាឧបករណ៍អគ្គិសនីមានតួនាទីបញ្ជូនបន្តនូវចរន្តអគ្គិសនី។ វាត្រូវបានគេប្រើសម្រាប់បញ្ជូនបន្តនូវចរន្តអគ្គសនី ២២០ វ៉ុល ឬ ក្រោមនេះទៅកាន់ឧបករណ៍ផ្សេងៗ ដោយវាអាចបើកឬបិទចរន្តតាមការបញ្ជារបស់អ្នកប្រើប្រាស់។ វាប្រៀបដូចជាកុងតាក់មួយហើយទទួលការបិទ ឬបើកពីប៉ត Aduino។ វាមាន Pin បួន VCC , GND, INT1 និង INT2។



8.4. Internet Server

ដើម្បីឲ្យប៉ត Arduino អាចទទួល និងបញ្ជូនទិន្នន័យរវាងឧបករណ៍ផ្សេងៗក្នុង WAN បានយើងត្រូវការ Server ផ្ទាល់ខ្លួនមួយសម្រាប់ផ្តល់លទ្ធភាពដល់ប៉ត Arduino និង ឧបករណ៍ផ្សេងអាចស្គាល់ និងទំនាក់ទំនងគ្នាបាន។ យើងបានប្រើប្រាស់នូវ Server របស់ Teleduino (Teleduino.org) ដែលជា Server Free មួយសម្រាប់ Ethernet Shield គ្រប់ប្រភេទទំនាក់ទំនងគ្នាជាមួយឧបករណ៍ដទៃទៀតក្នុង WANបាន ឬ Server មួយណាផ្សេងទៀតក៏បាន។

8.5. ការដំឡើងកម្មវិធីចូរសិ្ព



- រកកម្មវិធីខាងស្តាំដែលឈ្មោះ Web IoT ការបង្កើត Web Server គឺជាការបង្កើតនូវគេហទំព័រមួយដោយប្រើប្រាស់ក្រោមបណ្តាញជា Internet សម្រាប់អោយអ្នកប្រើប្រាស់អាចធ្វើការគ្រប់គ្រងនិងបញ្ជាទៅកាន់ឧបករណ៍ផ្សេងៗបានពីចំងាយ
- ត្រង់ប្រអប់ LED STATUS :
- ក្រោម LED STATUS: សម្រាប់ធ្វើការផ្ទើតម្លៃ
- ប៊ូតុង Save សម្រាប់ធ្វើការរក្សាទុកនូវព័ត៌មានផ្សេងៗ
- ប៊ូតុង Turn ON សម្រាប់ធ្វើការបើកអំពូល
- ប៊ូតុង Turn Off សម្រាប់ធ្វើការបិតអំពូល

កូដខាងក្រោមបង្ហាញពីការបិទ ឬបើកអំពូលដោយប្រើ Web IOT:

```
#include <Arduino.h>  //header file សម្រាប់ប្រើប្រាស់កូដក្នុង Arduino IDE
#include <ESP8266WiFi.h>//header file ឲ្យស្គាល់ NodeMCU ESP8266 board
//assign ឈ្មោះនិងលេខសម្ងាត់ WiFi ដើម្បីបើកដំណើរការបញ្ជាតាម internet
const char * SSID = "Home317";
const char * PASSWORD = "011995516vng";
WiFiServer server(80); //កំណត់លេខ port របស់ web server ស្មើតម្លៃ ៨0
// Assign output variables to GPIO pins
const int output = 4; //ប្រកាសអញ្ញាត output ស្មើ ៤ ត្រូវជាពីន D4 ក្នុង NodeMCU

void setup( ) {
          Serial.begin(115200);    //ចាប់ផ្តើម serial monitor ក្នុងល្បឿន 115200
         //ចាប់ផ្តើមអញ្ញាត output ជា OUTPUT និងកំណត់វាស្មើ low
          pinMode(output, OUTPUT);
          digitalWrite(output, LOW);
          Serial.print("Connecting to ");//បង្ហាញ "Connecting to"
          Serial.println(SSID); //បង្ហាញឈ្មោះ WiFi
          WiFi.begin(SSID,PASSWORD); //ភ្ជាប់ Wi-Fi network ជាមួយឈ្មោះនិងលេខសម្ងាត់
          while (WiFi.status( ) != WL_CONNECTED) {
           delay(500);
           Serial.print(".");}
         // Print local IP address
         Serial.println("");
         Serial.println("WiFi connected at IP address:");//បង្ហាញ "WiFi connected at IP
         //address:"

         Serial.println(WiFi.localIP( ));
         // Start Web Server
         server.begin( );
}
void loop( ){
          //ត្រួតពិនិត្យប្រសិនបើ client ត្រូវបានភ្ជាប់
          WiFiClient client = server.available( );
          // Do nothing if server is not available
          if (!client) {
            return;
          }
```

```
        // Wait a client
        while( !client.available( ) ){}//រង់ចាំរហូតដល់ client ផ្ញើទិន្នន័យ
        // A new client is connected, get the request
        String request = client.readStringUntil('\r'); //អានបន្ទាត់ដំបូងនៃការស្នើ
        Serial.println(request);// //បង្ហាញការស្នើ
        client.flush( );//ឲ្យ client ដំណើរការ
        int value = LOW; // ប្រកាសអថេរឈ្មោះ value
        if (request.indexOf("/led/1") != -1)
        {
            digitalWrite(output, HIGH);// ឲ្យអំពូលភ្លឺ
            value = HIGH;// ឲ្យអថេរ value=1
        }
        if (request.indexOf("/led/0") != -1)
        {
            digitalWrite(output, LOW);
            value = LOW;
        }
        // Display GPIO status
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("");
        client.println("<!DOCTYPE HTML>");
            client.println("<html><head><style>*{margin:0;padding:0;}body{backgrou
            nd-color:#edededff;}</style></head>");
        if(value == HIGH) {
            client.print("ON");
        } else {
            client.print("OFF");
        }
        client.println("</html>");
        Serial.println("");
}
```

8.6. ដំណើរការកូដ: បន្ទាប់ពីត្រូវបាន Upload កូដចូលទៅក្នុង NodeMCU ESP8266 នឹងចាប់យក IP address នៃ WiFi Network ដែលយើងបានកំណត់ក្នុងកម្មវិធី។

- IP address នោះនឹងបង្ហាញនៅក្នុង Serial monitor នៃ Arduino IDE
- ចំលង IP address ជាក់ចូល Web Browser នោះវានឹងចូលទៅកាន់ app ដែលបានបង្កើតក្នុង App Inventor 2 ដើម្បីកុងត្រូលទៅលើអំពូល

- ពេលចុចលើប៊ូតុង **Turn ON** នឹងបញ្ជូនទិន្នន័យទៅកាន់ប៉ត ដើម្បីដំណើរការកូដសម្រាប់ឱ្យ **Relay** ភ្ជាប់ចរន្តដើម្បីបើកអំពូល

- ដូចគ្នានេះដែរពេលចុចលើប៊ូតុង **Turn Off** នឹងបញ្ជូនទិន្នន័យទៅកាន់ប៉ត ដើម្បីដំណើរការកូដសម្រាប់ឱ្យ **Relay** ផ្តាច់ចរន្តដើម្បីបិតអំពូល

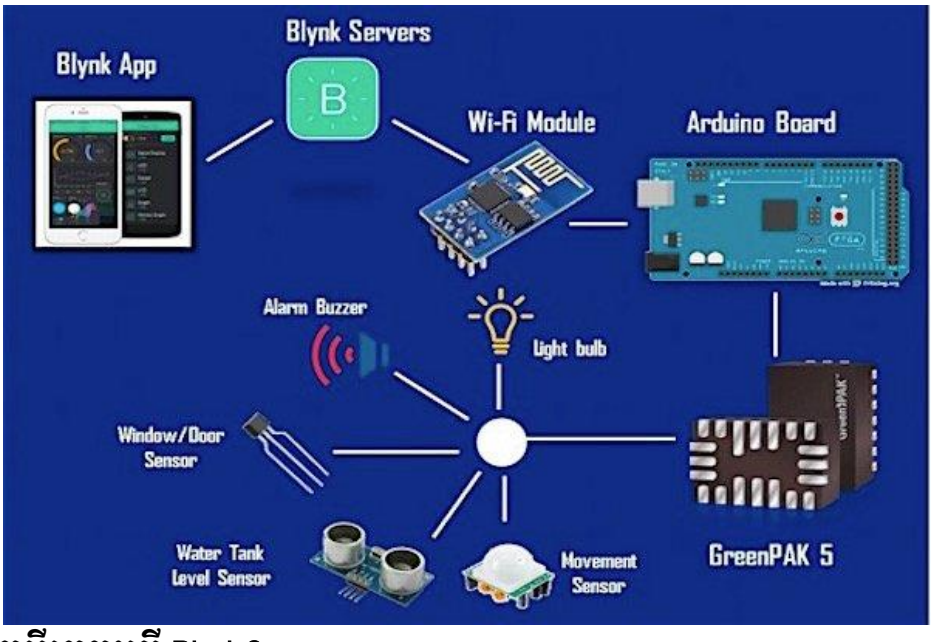
9. ការប្រើប្រាស់ Blynk

- Blynk ត្រូវបានរចនា និងបង្កើតឡើងសម្រាប់ IOT(Internet of Thing) គ្រប់គ្រងផ្នែករឹងពីចម្ងាយ និងអាចបង្ហាញទិន្នន័យរបស់ឧបករណ៍ចាប់សញ្ញា(Sensor data), អាចផ្ទុកទិន្នន័យ និងធ្វើអ្វីៗជាច្រើនទៀត។ វាទទួលខុសត្រូវរាល់ការទំនាក់ទំនងរវាងស្មាតហ្វូន(Smart Phone) និងផ្នែករឹង(Hard)។ យើងប្រើ Blynk Cloud ដើម្បីដំណើរការម៉ាស៊ីនមេ Blynk ឯកជនដែលរបស់ក្រុមមួយនៅក្រៅស្រុក។ វាជាប្រភពបើកចំហអាចគ្រប់គ្រងឧបករណ៍រាប់ពាន់យ៉ាងងាយស្រួលហើយថែមទាំងអាចត្រូវបានដាក់ដំណើរការនៅលើ Raspberry Pi ទៀតផង។

- កម្មវិធី Blynk App អនុញ្ញាតឱ្យយើងបង្កើត GUI ដ៏អស្ចារ្យសម្រាប់គម្រោងនេះដោយប្រើធាតុក្រាហ្វិក (Widgets)ផ្សេងៗដែលក្រុមហ៊ុនផ្តល់ឱ្យ។

- Blynk Library ប្រើសម្រាប់ដំណើរផ្នែករឹងដែលយើងបានបង្កើត និងបើកការទំនាក់ទំនងជាមួយម៉ាស៊ីនមេ និងដំណើរការរាល់ពាក្យបញ្ជាចូលនិងចេញ។

- មុននឹងធ្វើគម្រោងនេះសូមមើលពីដំណើរការវាតាមរូបខាងក្រោមជាការសំខាន់បំផុត



9.1. ហេតុអ្វីត្រូវប្រើ Blynk?

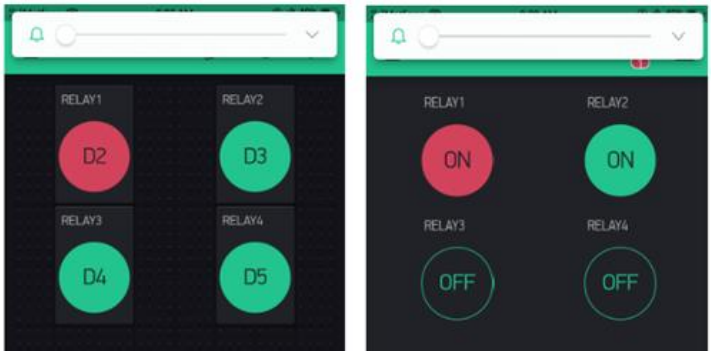
- ងាយស្រួលក្នុងការអនុវត្ត GUI និងផ្តល់នូវភាពងាយស្រួលក្នុងការបង្កើតគម្រោងថ្មីជាមួយកម្មវិធីប្រើនៅលើទូរស័ព្ទ

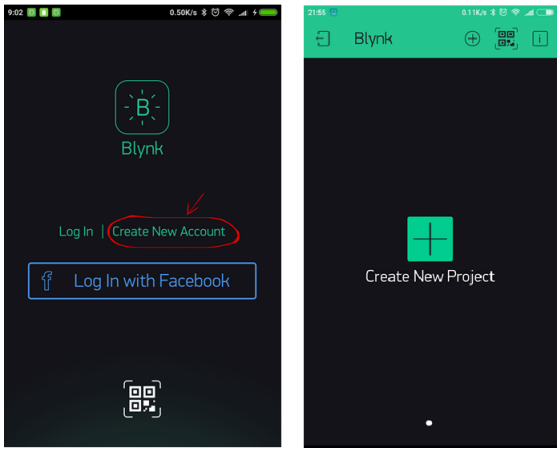
- មិនអាចប្រើប្រាស់ Local Network,ប្រើភាសា Arduino / C++និងអាចបង្កើតគម្រោង 3rd party

9.2. របៀបបង្កើត GUI ដើម្បីគ្រប់គ្រងចំនួន ៤ អំពូល

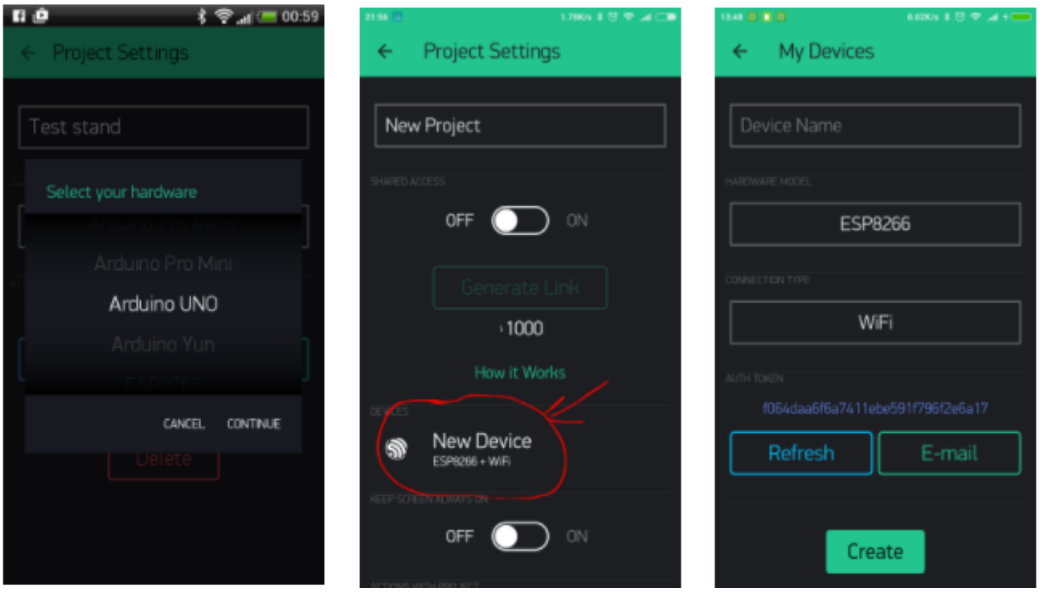
១. បន្ទាប់ពីបានទាញយក Blynk App ដាក់លើទូរសព្ទរួចយើងត្រូវបង្កើតគណនី Blynk ថ្មី។ បន្ទាប់មកចុចលើរូបកម្មវិធី Blynk ដើម្បីចូលកម្មវិធីនេះ

២. បង្កើតគម្រោងថ្មីដូចរូបខាងលើ ដោយអនុវត្តតាមជំហានបន្តបន្ទាប់ដូចខាងក្រោម៖



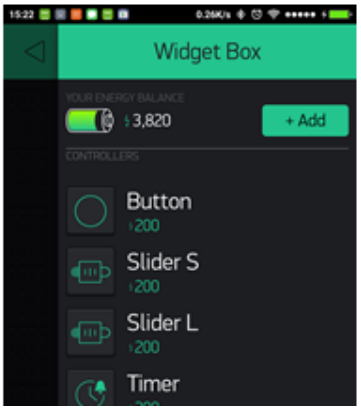


៣. វាយឈ្មោះត្រង់ Project Name និងជ្រើសរើស ESP8366 ត្រង់ CHOOSE DEVICE រួចចុចលើប៊ូតុង Create



៤. Auth Token គឺជាឧបករណ៍កំណត់អត្តសញ្ញាណពិសេសមួយដែលត្រូវការដើម្បីភ្ជាប់ផ្នែករឹងទៅនឹងស្មាតហ្វូន។ រាល់គម្រោងថ្មីដែលអ្នកបង្កើតនឹងមាន Auth Token ផ្ទាល់ខ្លួន។ យើងនឹងទទួលបានវាដោយស្វ័យប្រវត្តិនៅលើអ៊ីមែលរបស់អ្នកបន្ទាប់ពីបង្កើតគម្រោង។

៥. បង្កើត Interface ដោយចុចលើសញ្ញា និងជ្រើសរើស Widget ដែលចង់បាន។ ហើយការបង្កើតនេះយើងធ្វើដូចការប្រើ X86 Editor ឬភាសា Python ដោយប្រើ Widget យើងហៅថា Controls ឬដូចយើងបង្កើតកម្មវិធីដោយប្រើ App Inventor ដែរ។ ចូរមើលរូបខាងក្រោមដើម្បីធ្វើការរចនា។



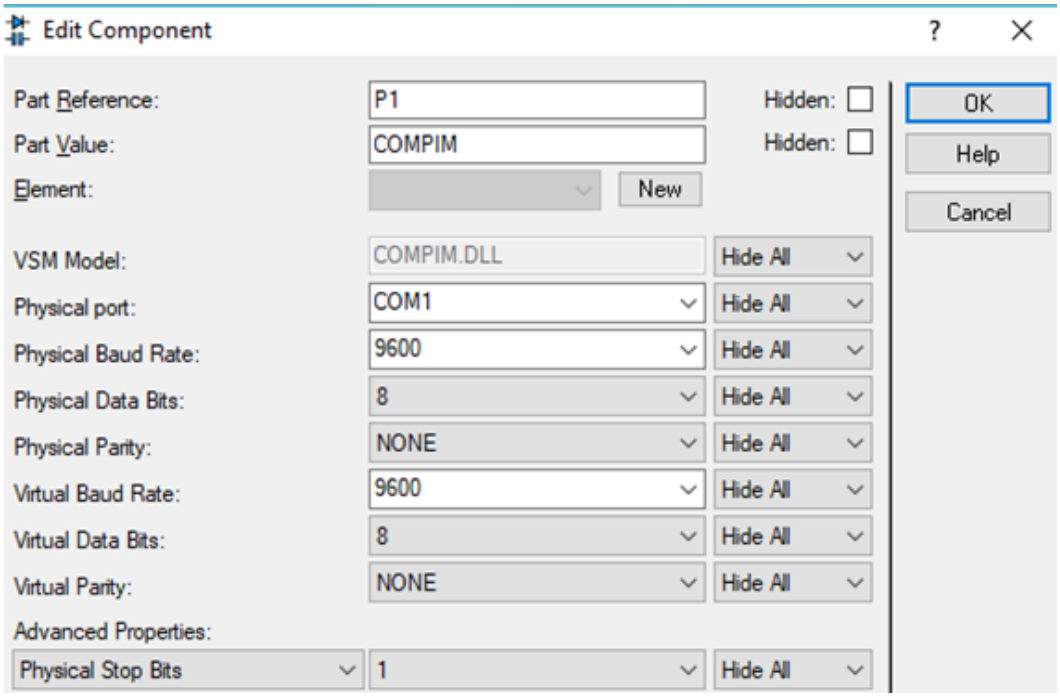
៦.ធ្វើការតភ្ជាប់ឧបករណ៍ NodeMCU ជាមួយនឹង LED មួយ និងសរសេរកូដខាងក្រោម៖

```
/*************************************************************
  Tuturial "LED control using NodeMCU and app Blynk
  Project: led( there is on button:D0 )
 *************************************************************/
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
// You should get Auth Token in the Blynk App.
// Go to the Project Settings ( nut icon ).
char auth[] = "Q0EaKQSOG0vv20xnaY4vpt0Fdu9g7yuv"; //You will get this Auth
Token in Gmail if you signin Blynk app with your Gmail ID
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Roboctic_Lab"; //Enter your WIFI Name
char pass[] = "2019@Robotic";//Enter your WIFI Password
void setup( ){
  // Debug console
  Serial.begin( 9600 );
  pinMode( D1,OUTPUT );
  pinMode( D2,OUTPUT );
  pinMode( D3,OUTPUT );
  pinMode( D4,OUTPUT );
  digitalWrite( D1,HIGH );
  digitalWrite( D2,HIGH );
  digitalWrite( D3,HIGH );
  digitalWrite( D4,HIGH );
  Blynk.begin( auth, ssid, pass );
```

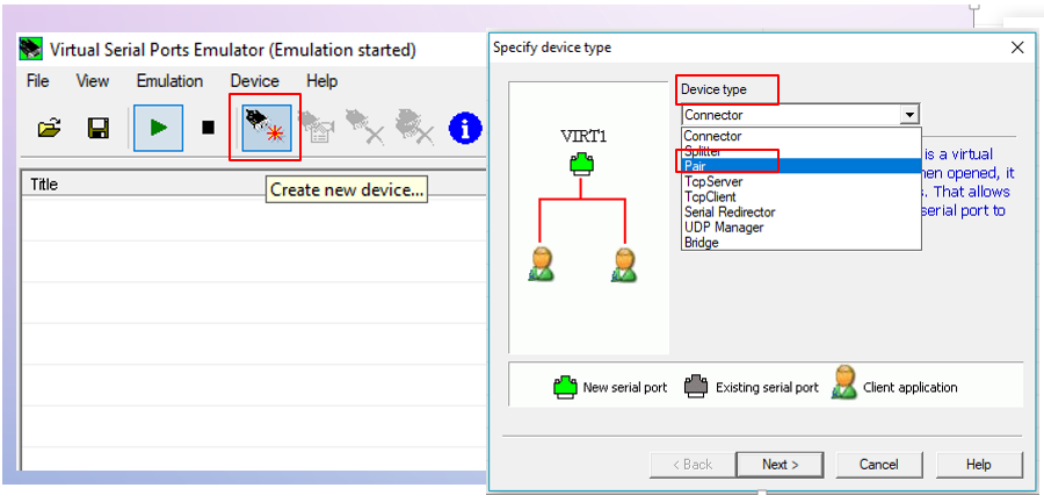
```
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 8442);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8442);
}
void loop(){
Blynk.run();}
```

7. តេស្តក្នុងកម្មវិធី Proteus: ក្នុងករណីយើងគ្មានប៉តខាងលើ យើងអាចធ្វើតេស្តវាដោយប្រើ កម្មវិធីសិប្បនិម្មិត **Proteus** តែនៅក្នុងកម្មវិធីគ្មានប៉ត **NodeMCU ESP8266 ទេ។** ដូច្នេះយើងប្រើ **Arduino Board** ជាមួយនឹង **COMPIM ជំនួសឱ្យ ESP8266។ ហើយអាចធ្វើការតភ្ជាប់ Port** ដែលអាចប្រើបានជាមួយនឹងកម្មវិធី **Proteus** យើងត្រូវទាញយក និងដំឡើងកម្មវិធីដែលមានឈ្មោះ VSPE (Virtual Serial Port Emulator)។ វាគឺជាកម្មវិធីមួយសម្រាប់តភ្ជាប់ Port ជាមួយនឹង COMPIM នៅក្នុងកម្មវិធី Proteus។ ដើម្បីឲ្យវាមានទំនាក់ទំនងគ្នាត្រូវអនុវត្តន៍តាមបីដំណាក់កាល៖

១. **ដំណាក់កាលទី១៖** នៅលើ COMPIM ចុចលើវាពីរដង បន្ទាប់មកយក Physical Port: COM1 (ឬជ្រើសយកឈ្មោះ Port ណាមួយក៏បាន) និងយក Physical Baud Rate:9600 និង Virtual Baud Rate:9600 ដូចការកំណត់នៅក្នុងកូដ។



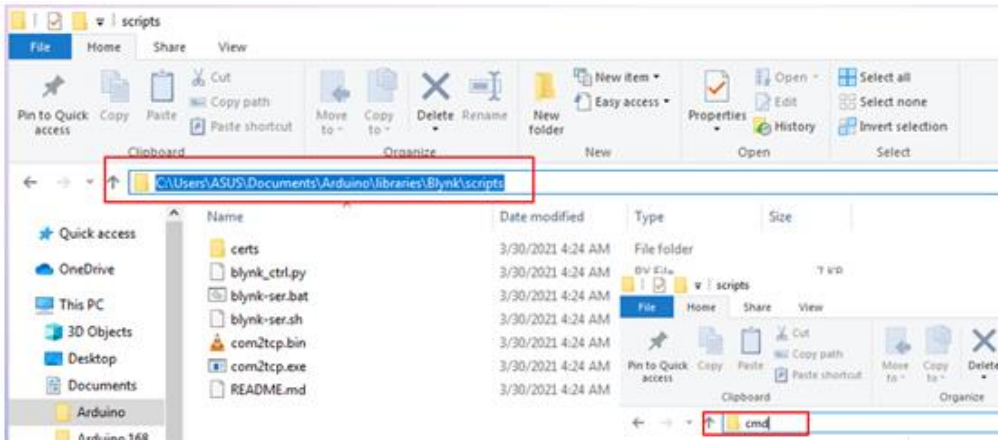
២. **ដំណាក់កាលទី២៖** នៅក្នុង VSPE បន្ទាប់ពីបានចូលក្នុងកម្មវិធីនេះហើយ ចុចលើ Create New Device បន្ទាប់មកត្រង់ Device Type រើសយក Pair ហើយចុច Next បន្ទាប់មកទៀតជ្រើសរើសឈ្មោះ Port (បើនៅក្នុង COMPIM យក COM1 នោះនៅក្នុង VSPE ត្រូវយក COM1 និង COM2 ហើយ បើនៅក្នុង COMPIM យក COM2 នោះនៅក្នុង VSPE ត្រូវយក COM2 និង COM3) និង Final ចុងក្រោយចុច Start Emulation ជាការស្រេច (បញ្ជាក់៖ មិនត្រូវបិទវាចោលទេ គ្រាន់តែ Minimize បានហើយ)។



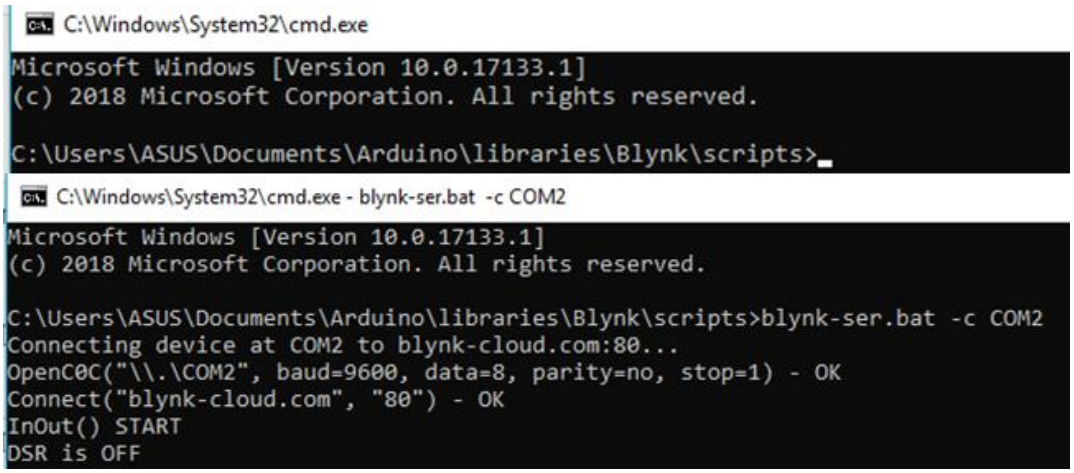
បន្ទាប់ចុចលើ Pair រួចហើយជ្រើសរើស COMx ពី Serial Virtual Port 1 និង COMy ពី Serial Virtual Port 2។ សូមបញ្ជាក់ COMx អាចជា COM1, COM2, COM3,... និង COMy គឺដូចគ្នាទៅតាម Port ប្រើប្រាស់ជាមួយនឹង Board ដែលយើងបាន Paired។ និងចុងក្រោយចុចលើ Start emulation ដើម្បីបានលទ្ធផលបាន COMx<==>COMy។

៣. ដំណាក់កាលទី៣៖

- ចូលទៅកាន់ទីតាំងរបស់ File Arduino ដូចរូបខាងក្រោម បន្ទាប់មកចំលងទុក ហើយវាយពាក្យ cmd បន្ទាប់មកចុច Enter



- ចុងក្រោយវាយបញ្ជា **blynk-ser.bat –c COMy**



10. Solar Remote Control Lamp

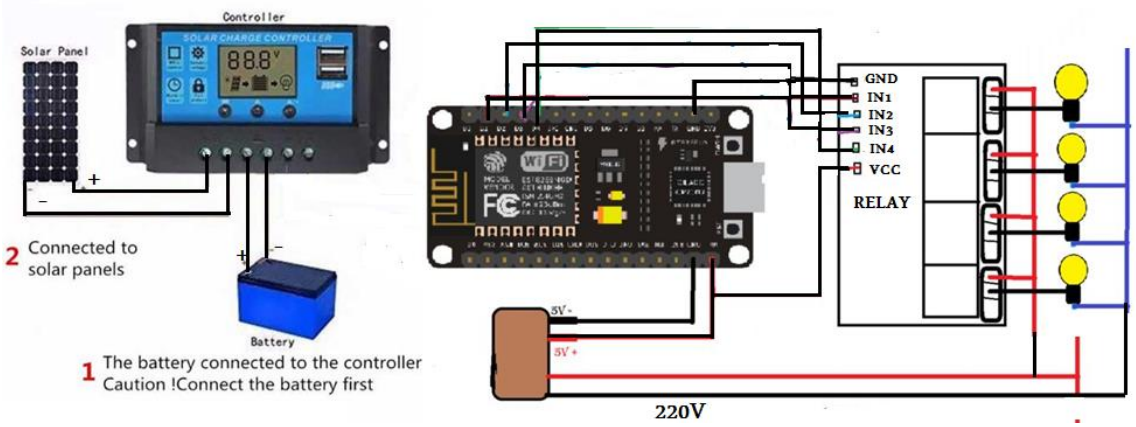
ការធ្វើគម្រោងនេះគឺមានការទាក់ទងទៅនឹងការគ្រប់គ្រងអំពូល។ ដោយអាចគ្រប់គ្រងអំពូលឱ្យ បិទ ឬបើកនូវទីតាំងណាក៏បាន។ ការគ្រប់គ្រងអំពូលនេះគ្រាន់តែជា ឧទាហរណ៍សម្រាប់បង្ហាញភាពងាយស្រួលនៃការ។ នៅក្នុងការងារនេះយើងប្លែកត្រង់ការប្រើ Charge Controller ដែលមានមុខងារគ្រប់គ្រងរវាង Solar Panel និងអាគុយ។ សារៈប្រយោជន៍ Solar Remote Control Lamp:

- ផ្តល់ធ្វើឱ្យយើងនូវភាពងាយស្រួល ចំណេញពេលវេលា និងចំណាយប្រាក់តិច
- មានភាពហ៊ាសទាន់ចិត្ត និងអាចជួយសម្រួលដល់កន្លែងដែលមិនទាន់មានអគ្គិសនីប្រើ
- អាចដាក់អំពូលនៅឆ្ងាយពីផ្ទះនិងអាចគ្រប់គ្រងវាបានយ៉ាងមានប្រសិទ្ធភាព និងឱ្យយើងមានទំនុកចិត្ត

10.1. គ្រឿងផ្សំ

- **Battery** គឺជាថាមពលអគ្គីសនីដែលផ្តល់ភ្លើងDC សម្រាប់ធ្វើការប្រើប្រាស់ជាមួយឧបករណ៍ដូចជា ទូរទស្សន៍ ម៉ាញ៉ូ និងអ្វីដែលសំខាន់ជាងនេះប្រើប្រាស់ជាមួយនឹង **Charge controller** សម្រាប់គ្រប់គ្រងចរន្តដែលទទួលបានពី **Solar panel**។
- **Charge controller** គឺជាឧបករណ៍ដែលមាននាទីគ្រប់គ្រងថាមពលអគ្គីសនីដែលទទួលបានពីការបម្លែងថាមពលព្រះអាទិត្យ ដើម្បីផ្តល់ថាមពលទៅឱ្យអំពូលមានពន្លឺ។
- **Solar Panel** គឺជាឧបករណ៍ដែលមាននាទីក្នុងការបម្លែងថាមពលព្រះអាទិត្យ ឱ្យក្លាយទៅជាថាមពលអគ្គីសនីដែលយើងអាចប្រើប្រាស់តាមតម្រូវការដែលយើងចង់បាន។

10.2. ការតភ្ជាប់ឧបករណ៍



10.3. របៀបបង្កើត APP

ចូលកម្មវិធី Blynk រួចបង្កើតកម្មវិធីតាមរូបខាងក្រោមដោយមានប៊ូតុងបួន D0, D1, D2 និងD3។ មើលពីរបៀបង្កើតកម្មវិធីនេះនៅចំណុចទី ៩ ខាងលើ

ការសរសេរកូដ

#define BLYNK_PRINT Serial	pinMode(D2,OUTPUT);
#include <ESP8266WiFi.h>	pinMode(D3,OUTPUT);
#include <BlynkSimpleEsp8266.h>	pinMode(D4,OUTPUT);
char auth[]="YourAuthToken";	digitalWrite(D1,HIGH);
char ssid[]="Robotic_Lab";	digitalWrite(D2,HIGH);
char pass[]="2019@Robotic";	digitalWrite(D3,HIGH);
void setup( )	digitalWrite(D4,HIGH);
{	Blynk.begin(auth,ssid,pass);
Serial.begin(9600);	}//**ដើម្បីឱ្យ**Solar lamp **ដំណើរការ។**
pinMode(D1,OUTPUT);	void loop( ){ Blynk.run( );}

11. IoT ប្រើប្រាស់ FireBase និង NodeMCU

11.1. Why Choose Firebase ?

Firebase ផ្តល់នូវវិធីរហ័សក្នុងការរក្សាទិន្នន័យដែលប្រមូលបាននៅកម្រិតឧបករណ៍ ហើយវាដំណើរការល្អជាមួយប្រព័ន្ធប្រតិបត្តិការ Android API ដែលគាំទ្រដោយ AndroidThings ។ អ្នកសរសេរកម្មវិធីទូរស័ព្ទ និងឧបករណ៍ជាច្រើនដែលបានប្រើប្រាស់ម៉ាស៊ីនមេ។ វាពិតជាអាចជួយបង្រួមគម្លាតនោះ និងធ្វើឱ្យវាកាន់តែងាយស្រួល។ វាគួរឱ្យចាប់អារម្មណ៍ណាស់ដែលអ្នកអភិវឌ្ឍន៍ប្រើលក្ខណៈក្រៅបណ្តាញរបស់វា។ វាផ្តល់ជាលទ្ធផលដល់អ្នកនឹងត្រូវប្រមូលទិន្នន័យក្រៅប្រព័ន្ធហើយនៅពេលមានបណ្តាញហើយបញ្ជូនព័ត៌មាននេះទៅម៉ាស៊ីនមេរបស់អ្នក។ វាមានលក្ខណៈពិសេសក្រៅបណ្តាញរបស់វាពិតជាអាចធ្វើឱ្យសាមញ្ញសម្រាប់អ្នកអភិវឌ្ឍន៍ជាច្រើន។ វាមានលក្ខណៈពិសេសរាប់បញ្ចូលទាំង Daltal Database, ការ

ផ្ទៀងផ្ទាត់ភាពត្រឹមត្រូវ, ការធ្វើសារ, ការផ្ទុក, គេហទំព័រ, បន្ទប់ពិសោធន៍និងការវិភាគ និងមូលដ្ឋានទិន្នន័យ ជាក់ស្តែងប៉ុណ្ណោះ។ នៅក្នុងការធ្វើគម្រោងនេះយើងចែកជាបីដូចខាងក្រោម៖

- ការបង្កើតគណនី Firebase
- ការបង្កើតកម្មវិធីប្រើនៅលើទូរសព្ទដោយប្រើ APP INVENTOR
- ការសរសេរកម្មវិធី Arduino

11.2. **ការបង្កើត** app

ចូរបង្កើតកម្មវិធី Android សម្រាប់គម្រោងនេះដោយប្រើកម្មវិធី MIT App Inventor ដែលយើងបានសិក្សារួចហើយនៅមេរៀនទី ៧។ សូមធ្វើការរចនា Interface ដូចខាងក្រោម៖

	- កម្មវិធីនេះសម្រាប់ធ្វើការបង្ហាញពីសីតុណ្ហភាព - Temperature : 0.00 អង្សារ និង Humidity:0.00% ដែលទាញមកពី Web Server - ប៊ូតុង ON សម្រាប់ធ្វើការបើកអំពូល - ប៊ូតុង ON សម្រាប់ធ្វើការបើកអំពូល

11.3. Programming NodeMCU

```
//FirebaseESP8266.h must be included before ESP8266WiFi.h
#include "FirebaseESP8266.h"      // Install Firebase ESP8266 library
#include <ESP8266WiFi.h>
#include <DHT.h>                  // Install DHT11 Library and Adafruit Unified
Sensor Library
#define FIREBASE_HOST "YOUR_FIREBASE_PROJECT.firebaseio.com"
//Without http:// or https:// schemes
#define FIREBASE_AUTH "YOUR_FIREBASE_DATABASE_SECRET"
#define WIFI_SSID "YOUR_WIFI_AP"
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"
#define DHTPIN 2                  // Connect Data pin of DHT to D2
int led = D5;                     // Connect LED to D5
#define DHTTYPE    DHT11
DHT dht( DHTPIN, DHTTYPE );
//Define FirebaseESP8266 data object
FirebaseData firebaseData;
```

```
FirebaseData ledData;
FirebaseJson json;
void setup( )
{
  Serial.begin( 9600 );
  dht.begin( );
  pinMode( led,OUTPUT );
  WiFi.begin( WIFI_SSID, WIFI_PASSWORD );
  Serial.print( "Connecting to Wi-Fi" );
  while ( WiFi.status( ) != WL_CONNECTED )
  {
    Serial.print( "." );
    delay( 300 );
  }
  Serial.println( );
  Serial.print( "Connected with IP: " );
  Serial.println( WiFi.localIP( ) );
  Serial.println( );
  Firebase.begin( FIREBASE_HOST, FIREBASE_AUTH );
  Firebase.reconnectWiFi( true );
}
void sensorUpdate( ){
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' ( its a very slow
sensor )
  float h = dht.readHumidity( );
  // Read temperature as Celsius ( the default )
  float t = dht.readTemperature( );
  // Read temperature as Fahrenheit ( isFahrenheit = true )
  float f = dht.readTemperature( true );
  // Check if any reads failed and exit early ( to try again ).
  if ( isnan( h ) || isnan( t ) || isnan( f ) ) {
    Serial.println( F( "Failed to read from DHT sensor!" ) );
```

```
  return;
}
Serial.print(F("Humidity: "));
Serial.print(h);
Serial.print(F("%  Temperature: "));
Serial.print(t);
Serial.print(F("C  ,"));
Serial.print(f);
Serial.println(F("F  "));
if (Firebase.setFloat(firebaseData, "/FirebaseIOT/temperature", t))
{
  Serial.println("PASSED");
  Serial.println("PATH: " + firebaseData.dataPath());
  Serial.println("TYPE: " + firebaseData.dataType());
  Serial.println("ETag: " + firebaseData.ETag());
  Serial.println("------------------------------------");
  Serial.println();}
else
{
  Serial.println("FAILED");
  Serial.println("REASON: " + firebaseData.errorReason());
  Serial.println("------------------------------------");
  Serial.println();
}
if (Firebase.setFloat(firebaseData, "/FirebaseIOT/humidity", h))
{
  Serial.println("PASSED");
  Serial.println("PATH: " + firebaseData.dataPath());
  Serial.println("TYPE: " + firebaseData.dataType());
  Serial.println("ETag: " + firebaseData.ETag());
  Serial.println("------------------------------------");
  Serial.println();
}
else
```

```
  {
    Serial.println("FAILED");
    Serial.println("REASON: " + firebaseData.errorReason());
    Serial.println("------------------------------------");
    Serial.println();
  }
}
void loop() {
  sensorUpdate();
  if (Firebase.getString(ledData, "/FirebaseIOT/led")){
    Serial.println(ledData.stringData());
    if (ledData.stringData() == "1") {
    digitalWrite(led, HIGH);
    }
  else if (ledData.stringData() == "0"){
    digitalWrite(led, LOW)};
  }
  delay(100)};
```

សំណួរ និង លំហាត់

1. ដូចម្តេចដែលហៅថា IOT? គេប្រើវាសម្រាប់ធ្វើអ្វី?
2. តើការប្រើបច្ចេកវិទ្យានេះមានសារៈសំខាន់អ្វីខ្លះចំពោះបដិវត្តឧស្សាហកម្ម 4.0ពេលបច្ចុប្បន្ននេះ?
3. ដូចម្តេចដែលហៅថា OLED? វាមាន Pin សរុបចំនួនប៉ុន្មាន? ចូរពន្យល់មុខងារនៃ Pin និមួយៗ
4. ចូរធ្វើការតភ្ជាប់រវាង NodeMCU ជាមួយនឹងអំពូល LED ៤ និងសរសេរកូដឱ្យវាដំណើរការ?
5. ដូចម្តេចដែលហៅថា (a). HTML , (b). Router, (c).WAN , (d). Network?
6. ចូរបង្កើត Interface តាមរូបខាងក្រោម៖

7. ចូរសរសេរកូដឱ្យ Interface សំនួរទី៦ ដំណើរការ Inface ខាងលើ?
8. ចូរនិយាយពីតួនាទីនៃ Interface DHT11 DHT22 w/ ESP8266 NodeMCU Using Web Server? ហើយប្រើនៅក្នុងវិស័យណាខ្លះ?
9. ចូរពន្យល់ពីកូដដែលបង្កើត Interface ខាងក្រោមឱ្យបានក្បោះក្បាយ?

10. ចូរនិយាយពីរបៀបប្រើប្រាស់នៃ Interface សំនួរទី 9.
11. ដូចម្តេចដែលហៅថាការគ្រប់គ្រងអំពូលតាមអ៊ិនធើណិត? តើវាផ្សំពីឧបករណ៍អ្វីខ្លះ? រួចនិយាយពីមុខងារនិមួយៗ របស់វា?
12. តើ Blynk គេប្រើសម្រាប់ធ្វើអ្វី? ចូរបង្កើត Interface ដើម្បីបានដូចក្នុងរូបខាងក្រោម៖

13. ចូរពន្យល់ពីដំណើរការតាមរូបខាងក្រោម៖

មេរៀនទី ១២ ភាសា Micropython ជាមួយនឹង ESP32 និង ESP8266

Python ជាភាសាកម្មវិធីកម្រិតខ្ពស់មួយដែលមានភាពពេញនិយមបំផុតសំរាប់ការសរសេរកម្មវិធីនាពេលបច្ចុប្បន្ននេះ។ វាមានភាពងាយស្រួលក្នុងការសរសេរកម្មវិធីដែលមានភាពស្មុគស្មាញបានលឿនជាជាងការប្រើភាសាដទៃទៀត។ យើងប្រើវាសម្រាប់គោលបំណងទូទៅ (General Purpose) និងមានលក្ខណៈជា Dynamic System ហើយវាគ្រប់គ្រងលើ Memory ដោយស្វ័យប្រវត្តិ។ វា support ទម្រង់ programming ជាច្រើន ដូចជា OOP, multi_pursose និងភ្ជាប់មកជាមួយ standard library មួយចំនួនផងដែរ។ Python អាចប្រើប្រាស់បានជាមួយប្រព័ន្ធប្រតិបត្តិការជាច្រើនដូចជា Windows, Linux, និង MacOS។

ការប្រើភាសា Python សម្រាប់ការងារ និងវិស័យមួយចំនួនធំដូចខាងក្រោម៖

- បង្កើតវិបសាយ
- បង្កើត App ឬ Desktop App
- ការគ្រប់គ្រងទិន្នន័យ ឬបណ្តាញផ្សេងៗ
- AI និង Machine Learning
- Automation
- ការសរសេរកម្មវិធីទាក់ទងនឹង ESP32/ ESP8266, Arduino, Raspberry
- ផ្នែកវិទ្យាសាស្ត្រដូចជាគណិតវិទ្យាសម្រាប់ដោះស្រាយនូវបញ្ហាមួយចំនួនដែលមានភាពស្មុគស្មាញអង្គការ NASA

1. កម្មវិធីសម្រាប់សរសេរភាសា Python

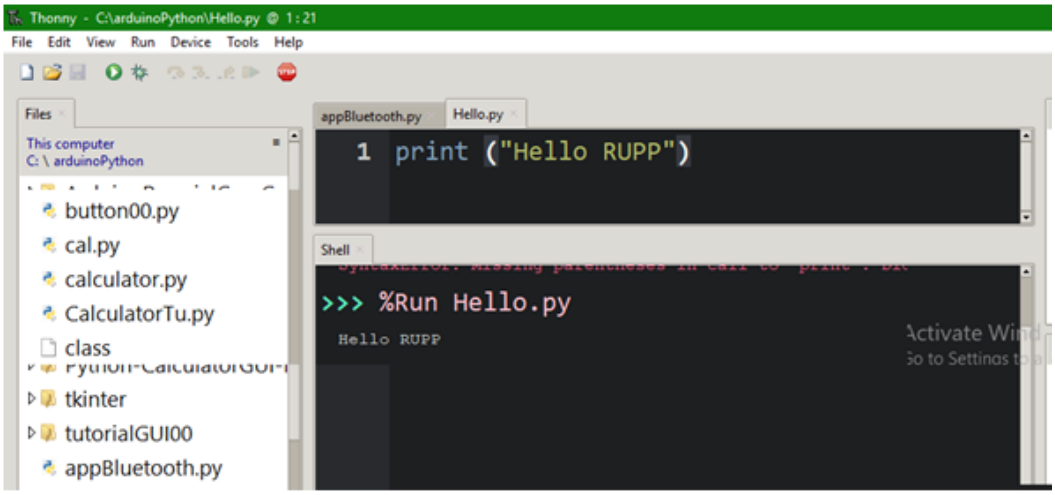
Python IDE ប្រើសម្រាប់សរសេរភាសា Python ហើយវាច្រើនប្រភេទមានដូចជា IDLE, Thonny, PyChar, Visual Studion Code, Jupyter,...។ នៅក្នុងមេរៀននេះយើងប្រើ Editor ទាំងនេះដើម្បីសរសេរកូដសម្រាប់ធ្វើការដោះស្រាយចំណោទ ឬបញ្ហាផ្សេងៗដូច C/C++។ ជាពិសេសកូដធ្វើការបញ្ជាឧបករណ៍អេឡិចត្រូនិច,ឧបករណ៍វៃឆ្លាត, រ៉ូបូត និង AI ដោយប្រើ Board Raspberry និង NodeMCU

ការដំឡើង និងបើកកម្មវិធី Thonny

១. ដំឡើងកម្មវិធី **Python** ដោយរកវានៅ https://www.python.org/ និង កម្មវិធី Thonny ដោយទាញពីទីតាំងដែលបានកំណត់នៅលើ Google

២.ចូលកម្មវិធី Python IDE ដែលចង់ប្រើ, ដូចករណីខាងក្រោម ប្រើ Thonny Editor

៣. សរសេរបញ្ជាតាមរូបខាងក្រោម និងចុចលើ run current script



2. អនុគមន៍ print( )

វាជាអនុគន៍មានមុខងារដូចប្រើ printf( ) ដែលយើងបានប្រើនៅក្នុងភាសា C ដែរ។ និងអនុគមន៍ print នៅក្នុងកម្មវិធី X86Asm ដោយយើងគ្រាន់តែ print "Hello Rupp"។ អនុគមន៍ទាំងនេះសម្រាប់បង្ហាញព័ត៌មានជាអក្សរ ឬ លេខ នៅលើ Output screen។

- **Syntax** : **print**(Expression)
- **អនុវត្ត១**:គឺជាកូដខាងក្រោមដែលបានសរសេររួចហើយនៅលើកម្មវិធីខាងលើដោយប្រើ Thonny ឬ Jupyter Editor។

  print ("Hello Rupp")
- Output Screenn

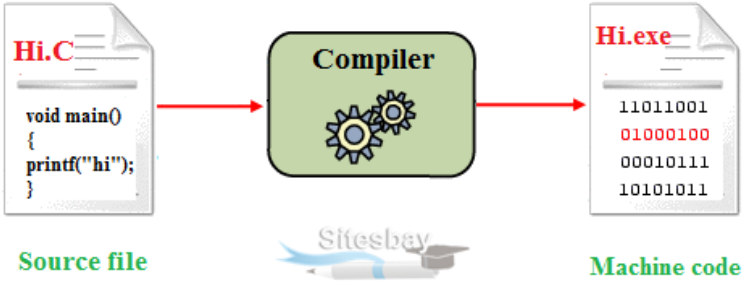
  Hello Rupp
- **អនុវត្ត២**: ចូរសាកល្បងសរសេរកូដក្នុង Editor ខាងលើដើម្បីបង្ហាញឃ្លាដូចខាងក្រោម (" ការដាក់ឈ្មោះតំណាងអោយកូដត្រូវបញ្ចប់ដោយ.py" ដោយប្រើអថេរ text):

```
In [3]: text = ("ការដាក់ឈ្មោះតំណាងអោយកូដត្រូវបញ្ចប់ដោយ.py")
        print (text)
```

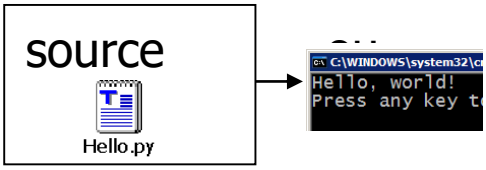
ការដាក់ឈ្មោះតំណាងអោយកូដត្រូវបញ្ចប់ដោយ.py

3. Compiling និង Interpreting

ភាសាជាច្រើនទាមទារអោយយើងបកប្រែកម្មវិធី ឬកូដរបស់យើងទៅជាទម្រង់មួយដែលម៉ាស៊ីនអាចយល់បាន។ Compiler ជាកម្មវិធីសម្រាប់ធ្វើការបំលែងកម្មវិធីទាំងមូលទៅ ឬយើងហៅថា Source Code ទៅជាភាសាម៉ាស៊ីនដែលជាសំនុំនៃ Instructions សម្រាប់ប្រាប់អោយ CPU ធ្វើការងារអ្វីមួយដែលមាននៅក្នុងកម្មវិធីនោះ ។



ចំណែកនៅក្នុងភាសា Python វិញ ប្រើ Interpreter សម្រាប់ធ្វើការបកប្រែកូដទៅជាភាសាម៉ាស៊ីនដោយផ្ទាល់តែម្តងដោយធ្វើការ Run តែម្តង។



ការប្រើសម្រាប់ធ្វើប្រមាណវិធីគណិតវិទ្យា និងតក្កវិទ្យា។ និមិ្មតសញ្ញាដែលមានបន្តបន្ទាប់ក្នុងតារាងដូចបានបង្ហាញសម្រាប់សរសេរកន្សោមគណិតគណនារកលទ្ធផលនៃចំណោទ ឬបញ្ហាមួយ។ ប្រើជាកន្សោមតក្កប្រើប្រាស់លក្ខខណ្ឌនៅបញ្ជា if និងវង្វិលជុំផ្សេងៗ។ ឧទាហរណ៍បើយើងសរសេរ if (x!= 0) ។ x!=0 ជាកន្សោមតក្ក និងមានន័យថាបើអថេរ x មានតំលៃមិនស្មើនឹងសូន្យលក្ខខណ្ឌនឹងពិត ដូចជា x =4, 3, 10,22, ….។ បើយើងសរសេរកន្សោមគណនា sum = x + y។ កន្សោមនេះជាការគណនារកលទ្ធផល Sum។

4. អថេរ(Variable)

4.1. ការប្រើអថេរ

អថេរត្រូវបានប្រើប្រាស់នៅក្នុងការសរសេរកម្មវិធី(កូដ) ដើម្បីរក្សាទុកនូវតម្លៃ សម្រាប់ធ្វើប្រមាណវិធីនព្វន្ត, តក្កវិទ្យា និងប្រមាណវិធីផ្សេងៗដទៃទៀត។ វាតាងអោយលេខអាសយដ្ឋានរបស់ Memory ដើម្បីផ្ទុកតម្លៃនៅពេលកម្មវិធីមួយត្រូវបានបំលែង។

4.2. ប្រភេទទិន្នន័យ

គ្រប់អថេរទាំងអស់ត្រូវមានឈ្មោះ និង ប្រភេទទិន្នន័យ(char, int, long int) ប្រើសម្រាប់សម្គាល់នៃប្រភេទ និងទំហំនៃអថេរដែលត្រូវផ្ទុកលេខជាចំនួនគត់, ទសភាគ និង អក្សរ ព្រមទាំងទំហំនៃអថេរដែលត្រូវផ្ទុកនៅក្នុង Memory។ តារាងខាងក្រោម បង្ហាញពីប្រភេទទិន្នន័យប្រើក្នុង Arduino IDE:

Name	Type	Description
Integers	int	Whole number such as : 3, 3000, 200
Floating Point	float	Number with floating point : 12.25, 100.75, 45. 50
String	Str	Ordered sequence of characters : "Hello", "1000"
List	list	Ordered sequence of objects :[100, "Hello", 200.5]
Dictionarys	dict	Unordered Key:{"mykey":"value","name":"ny"
Tuples	tup	Order immulable sequence of objects(10, "hell0",)
Sets	Set	Unordered collection of unique object {"a","b","c"
Boolean	bool	Logical value indication : True or False

អនុវត្តទី១: កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើប្រាស់អថេរផ្ទុកតំលៃជាប្រភេទគត់ និង String រួចធ្វើបង្ហាញលទ្ធផល:

```
x = 100 # x is of type int
name="Savath Vothearny" # name is now of type str
```

```
print ( " x=",x )
print ( name )
```

💻.លទ្ធផល

```
x = 100
Savath Vothearny
```

អនុវត្តទី២: កម្មវិធីខាងក្រោមបង្ហាញការប្រើកន្សោមគណនារកផលបូក និងផលដកពីរចំនួនដោយប្រើចំនួនគត់ និងចំនួនទសភាគ:

```
int1 = 100 ;int2 =200 ; ប្រា
flo1 =100.75 ; flo2 = 45.25 ;
sum = int1 + int2
dif = flo1 - flo2
print( int1,"+",int2,"+",sum )
print( flo1,"-",flo2,"=",dif )
```

Output Screen

```
100 + 200 + 300
100.75 - 45.25 = 55.5
```

អនុវត្តទី3: កូដគំរូខាងក្រោមប្រើអថេរប្រភេទជាប្រភេទ:

```
#List
a = [10, 5.5, 'python']
print( a )
a = [5,10,15,20,25,30,35,40]
# a[2] = 15
print( "a[2] = ", a[2] )
# a[0:3] = [5, 10, 15]
print( "a[0:3] = ", a[0:3] )
# a[5:] = [30, 35, 40]
print( "a[5:] = ", a[5:] )
```

Output Screen

```
[10, 5.5, 'python']
a[2] =  15
a[0:3] =  [5, 10, 15]
a[5:] =  [30, 35, 40]
```

Dictionary

```
dict = {1:"john","lastname":"smith", "age":25}
# prints the value where key value is 1
print( dict[1] )
```

```
        # prints the value where key value is "lastname"
        print( dict["lastname"] )
        # prints the value where key value is "age"
        print( dict["age"] )
```

Output Screen

john

smith

25

អនុវត្តទី ៤

```
# Python Program - Make Simple Calculator

print( "1. Addition" );
print( "2. Subtraction" );
print( "3. Multiplication" );
print( "4. Division" );
print( "5. Exit" );
choice = int( input( "Enter your choice: " ) );
if ( choice>=1 and choice<=4 ):
    print( "Enter two numbers: " );
    num1 = int( input( ) );
    num2 = int( input( ) );
    if choice == 1:
        res = num1 + num2;
        print( "Result = ", res );
    elif choice == 2:
        res = num1 - num2;
        print( "Result = ", res );
    elif choice == 3:
        res = num1  * num2;
        print( "Result = ", res );

    else:
        res = num1 / num2;
        print( "Result = ", res );
elif choice == 5:
    exit( );
```

```
else:
    print( "Wrong input..!!" );
```

5. អនុគមន៍ និង Type Conversion

5.1. អនុគមន៍

គឺជា subprogram ឬប្លុកកូដ នៃ main program ប្រើសម្រាប់ដោះស្រាយនូវចំណោទ ឬ បញ្ហាណាមួយ អោយទៅផ្នែកៗ និងមានតួនាទីខុសគ្នា។ ឧទាហរណ៍បើយើងសរសេរកូដបង្កើត ឧបករណ៍គិតលេខមួយ នោះយើងត្រូវបង្កើតឈ្មោះអនុគមន៍បួនមាន add_twonum; subtract_twonum; multiple_twon, និង divide_twonum ដោយអនុគមន៍និមួយៗមាន Subprogram និងមុខងាររៀងៗខ្លួន។

អនុគន៍អាចហៅមកប្រើ ម្តងហើយម្តងទៅតាមតំរូវការពេលដំណើរការ ដោយប្រើតែឈ្មោះនៃអនុគមន៍ នោះ ដោយមិនបាច់សរសេរ Subprogram នោះឡើងវិញ។ ការប្រើអនុគមន៍មានភាពងាយស្រួលក្នុងការ សរសេរកូដទៅតាមផ្នែកនិមួយៗ និងងាយស្រួលក្នុងការស្វែងរកកំហុស។

ការប្រើអនុគមន៍ក្នុងភាសានិមួយៗមានលក្ខណៈខុសគ្នាផងដែរ។ នៅក្នុងភា សា Python អនុគមន៍ ត្រូវប្រើ Keyword "def" បន្ទាប់មកដាក់មួយឈ្មោះ និង Parameter ឬ Argument ត្រូវតែដាក់នៅក្នុង វង់ក្រចក "( )" និងបញ្ចប់ដោយ " : " ។

```
def  function_name  (parameters):
⟷ function body
```

5.1. input ជា function សម្រាប់បញ្ចូលទិន្នន័យជាអក្សរពី Keyboard តែប៉ុណ្ណោះ។

```
Variable = input( prompt )
prompt ជាអក្សរត្រូវបង្ហាញមុនពេល input
name = input( "Enter your name: " )
print( "Hello", name )
```

Output Screen

```
Enter your name: Bora
Hello Bora
```

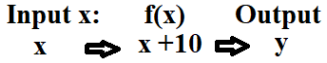
5.2. ការបង្កើតអនុគមន៍

អនុវត្តទី១

```
def add(x, y):
        return x + y
num1 = 1000
num2 = 2000
print( num1,"+",num2,"=", add( num1,num2 ) )
```

អនុវត្តទី២

```
def myFunction( a ):
    y= a+10
    return y
```



```
print("Programstarts!")
x = 3
result = myFunction(2+x)
print("Result of function call:", result)
print("Enter correct operation"
```

5.3. Type Conversation

Type conversion ជាបំលែងទិន្នន័យពីប្រភេទមួយ ទៅប្រភេទមួយទៀត។

int(expression) : បំលែងទៅជាចំនួនគត់ (integer)

float(expression) : បំលែងទៅជាចំនួនទសភាគ (float)

str(expression) : បំលែងទៅជាអក្សរ (string)

អនុវត្តទី ១

```
num = 25
fnum = float( num )
print( num, "Convert into", fnum )
```

Output Screen :

```
25 Convert into 25.0
```

អនុវត្តទី ២

```
name = input( "Enter your name: " )
year = int( input( "Enter your birth year: " ) )
print( "Hello", name )
print( "Your are", 2019-year )
```

Output Screen

```
Enter your name: Bopha
Enter your birth year: 1999
Hello Bopha
Your are 20
```

6. ការប្រើ IF statement

If condition :

```
statementA
```

If statement គឺត្រូវបានប្រើប្រាស់ធ្វើការប្រតិបត្តិប្លុកនៃកូដរបស់វា។ បើសិនលក្ខខណ្ឌ ពិតអនុវត្ត statementA និងបើលក្ខខណ្ឌមិនពិតមិនអនុវត្ត statementA។ នៅក្នុងភាសា Python ប្លុកនៃកូដរបស់វាត្រូវចុច Tab Key ហើយអាចសរសេរ Statement មួយឬច្រើនបាន ដោយមិនចាំបាច់ប្រើ { } ដូចភាសា C និងដើម្បីបង្កើតកន្សោមលក្ខខណ្ឌត្រូវមើលតាម ឧទាហរណ៍ខាងក្រោម៖

- Equals: `a == b`
- Not Equals: `a != b`

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

```
x = input( "Enter a x:" )
y = input( "Enter a y:" )
if( x>y ):
   print(  "x is greater than y" )
if ( x<y ) :
   print(  "y is greater than x" )
```

Output Screen

```
Enter a x:50
Enter a y:60
y is greater than x
>>> %Run ifsatement00.py
Enter a x:69
Enter a y:50
x is greater than y
```

7. ឧទាហរណ៍ Arduino if...else statement

```
If condinton :
        statement A
else
        statement B
```

ការដំណើរការបញ្ជា if... else ដំបូងពិនិត្យលក្ខខណ្ឌនៅក្នុងបញ្ជា if បើវាពិតអនុវត្ត statementA និងបើលក្ខខណ្ឌ មិនពិតអនុវត្ត stetementB។ ទាក់ទងនឹងចំណោទ ឬបញ្ជា ដែលប្រើពីរលក្ខខណ្ឌជាជាងយើងប្រើបញ្ជា if ច្រើនដង។

ឧទាហរណ៍: កូដខាងក្រោមបង្ហាញពីដំណើរការប្រើ if...else

```
x = input( "Enter a x:" )
y = input( "Enter a y:" )
if( x>y ):
   print(  "x is greater than y" )
else:
   print(  "y is greater than x" )
```

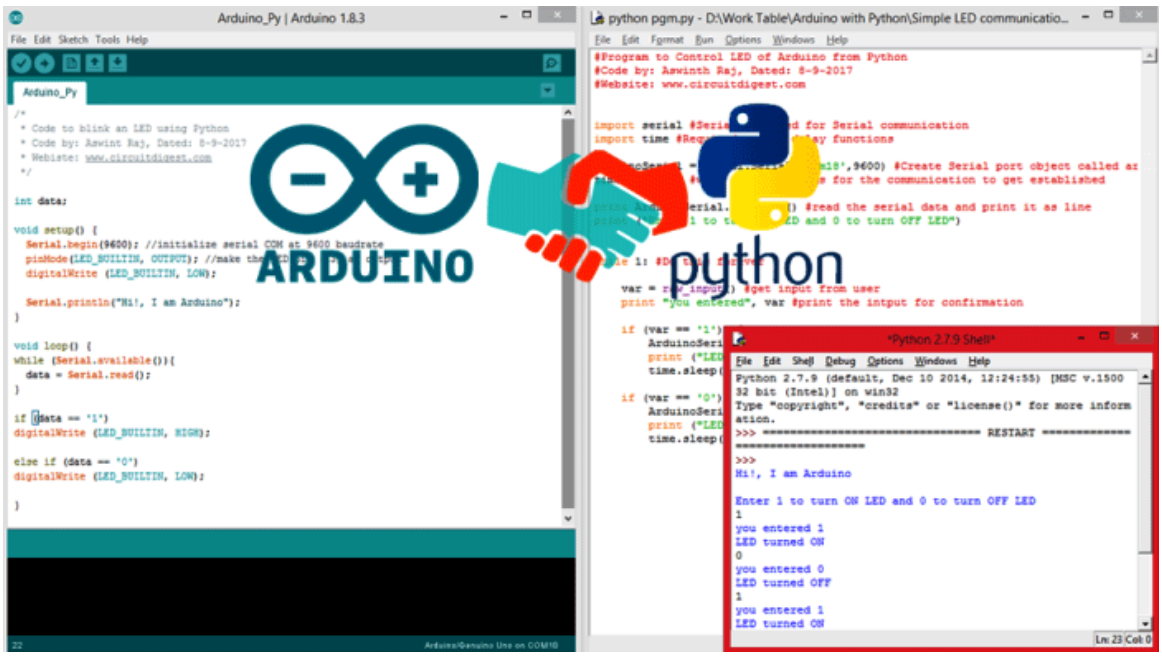
ប្រសិនបើមានលក្ខខណ្ឌច្រើន គេអាចតភ្ជាប់ជាមួយនឹង elif (else if)

```
if condition :
        statement(s)
elif condition :
        statement(s)
else :
        statements(s)
```

ឧទាហរណ៍១

```
print("1. Addition");
print("2. Subtraction");
print("3. Multiplication");
print("4. Division");
print("5. Exit");
choice = int(input("Enter your choice: "));
if (choice>=1 and choice<=4):
   print("Enter two numbers: ");
   num1 = int(input( ));
   num2 = int(input( ));
   if choice == 1:
      res = num1 + num2;
      print("Result = ", res);
   elif choice == 2:
      res = num1 - num2;
      print("Result = ", res);
   elif choice == 3:
      res = num1  * num2;
      print("Result = ", res);
   else:
      res = num1 / num2;
      print("Result = ", res);
elif choice == 5:
   exit( );
else:
   print("Wrong input..!!");
```

ឧទាហរណ៍២ : កូដខាងក្រោមការប្រើប្រាស់កម្មវិធីសរសេរក្នុងភាសា Python និងកម្មវិធី Arduino Board ដើម្បីធ្វើការបិត ឬបើក Led មួយ ។



កម្មវិធីវិធី Arduino

```
const int led=13;
int value=0;
void setup( )
{
Serial.begin(9600);
pinMode(led, OUTPUT);
digitalWrite (led, LOW);
Serial.println("Connection established...");
}
void loop( )
{
while (Serial.available( ))
{
value = Serial.read( );
}
If (value == '1')
        digitalWrite (led, HIGH);
else if (value == '0')
        digitalWrite (led, LOW);
}
```

កម្មវិធីវិធី Arduino

8. Loops

ការប្រើវង្វិលជុំ(Loop) សម្រាប់ធ្វើការប្រតិបត្តិ Instruction មួយឬច្រើនក្នុងទំរង់ ដដែលដើម្បីធ្វើប្រមាណវិធី។

8.1. ការប្រើប្រាស់ for loop

For ជាLoopមួយប្រើសម្រាប់អនុវត្តន៍ Statement ច្រើនដងទៅតាមចំនួនកំណត់ណាមួយ។

for variableName **in** groupOfValues **:**

statement(s)

variableName ជាអថេរប្រើសម្រាប់ចាប់យកតម្លៃពី groupOfValues

groupOfValues អាចត្រូវបានគេប្រើដោយអនុគមន៍ range( ) រឺ List។

range(start, stop)

range(start, stop, step)

Start ជាតម្លៃចាប់ផ្តើម

Stop ជាតម្លៃចុងបញ្ចប់

Step ជាដំហានកើន ឬ ថយ

range មានតម្លៃចាប់ពី start រហូតដល់ stop-1។ ដែលតម្លៃរបស់វាកើន ឬ ថយ ទៅតា step

```
for x in range( 1,4 ):
    print( x,"squared is",x *x )
```

Output

```
1 squared is 1
2 squared is 4
3 squared is 9
```

Initialize Expression

វាជំណើរការតែម្តងនៅពេល loop ពេលចាប់ផ្តើមដំណើរការ។ វាជាតំលៃដំបូងសម្រាប់អថេរ x=1 និងប្រើវាសម្រាប់បង្កើនតំលៃ x

Loop Condition *ពេលគឺជាកន្សោមលក្ខខ័ណ្ឌនៃ Loop ដើម្បីបន្ត loop នៅពេល កន្សោមលក្ខខ័ណ្ឌ True នៅពេល =3 និង x=4 មិនពិត ចាកចេញពី Loop ។*

អនុវត្តទី ១

```
print( "Odd Number" )
for i in range( 1,10,2 ):
    print( i )
```

```
Output:
Odd number
1
3
5
7
9
```

8.2. ការប្រើប្រាស់ while loop

ការប្រើប្រាស់វាប្រហាក់ប្រហែលនឹងការប្រើ For loop ដែរ។ តែខុសគ្នាត្រង់ការប្រើ While loop គឺបំបែកកន្សោមនៃវង្វិលជុំនៅដាច់ៗពីគ្នា។ វាត្រូវបាន ប្រើប្រាស់សម្រាប់ដំណើរការប្តូកនៃកូដ(code)នៅក្នុងខ្លួនវាម្តងហើយម្តងទៀតរហូតដល់កន្សោមលក្ខខ័ណ្ឌមិនពិតទើបចាកចេញពី Loop។

Syntax :

```
while (condition) {

 // statement(s)

}
```

9. អនុគមន៍(Function)

អនុគមន៍ជាបញ្ជាដែលមានឈ្មោះផ្សេងៗពីគ្នា ហើយអាចហៅយកមកប្រើម្តងៗម្តងទៀតទៅតាមកិច្ចការដែលត្រូវធ្វើ និងវាមានស្រាប់នៅក្នុង Arduino IDE និង Library ណាមួយនៃឧបករណ៍ដែលយកមកប្រើ។ យើងបានសិក្សារួចមកហើយនៅមេរៀនមុខនូវអនុគមន៍សំខាន់ៗមានដូចជា setup( ), loop( ), digitalWrite( ) ឬ println( ), វានិមួយៗគឺមានសំនុំនៃ Instructions រៀងៗខ្លួននិងមានតួនាទីខុសៗ។ នៅក្នុងជំពូកនេះយើងនឹងសិក្សាអំពីអនុគមន៍មួយចំនួនទៀតដែលស្រាប់នៅក្នុង Arduino IDE និង អនុគមន៍បង្កើតដោយខ្លួនឯង។

អនុគមន៍គឺជាបណ្តុំនៃ Instructions។

មូលដ្ឋានគ្រឹះក្នុងការបង្កើតអនុគមន៍មួយយើងត្រូវគិតលើបីចំនុច។ ចំនុចទី១ ឈ្មោះអនុគមន៍(Function)និមួយៗត្រូវមានឈ្មោះពិសេសទៅតាមបញ្ជាងាយក្នុង ការចំណាំ និងស្រួលប្រើ រួចបញ្ចប់ដោយសញ្ញា ( ) ដើម្បីបញ្ជាក់ឈ្មោះនៅពីក្រោយជាឈ្មោះអនុគមន៍។ ចំនុចទី ២ អនុគមន៍ត្រូវមានការត្រលប់តំលៃ ឬគ្មានត្រលប់តំលៃអាស្រ័យលើការប្រើ Return Type។ និងចំនុចទី៣ចុងរូបរាង នៃអនុគមន៍ត្រូវប្រើ សញ្ញាប { }

9.1. ឈ្មោះអនុគមន៍

នៅពេលយើងបង្កើតអនុគមន៍ត្រូវដាក់ឈ្មោះហើយការដាក់ឈ្មោះដូចគ្នាទៅនឹងប្រើអថេរដែរ៖

- ឈ្មោះវាអាចត្រូវបានបង្កើតឡើងដោយអក្សរក្រមលេខ (A ដល់ Z; a ដល់ z; 0 ដល់ ៩)។ ឧទាហរណ៍បង្កើតឈ្មោះ addTwoNumber( ) និងមិនចាប់ផ្តើមដោយលេខ ពីលេខ0 ដល់៩។
- មិនត្រូវប្រើឈ្មោះអនុគមន៍ដែលដូចគ្នានឹងពាក្យគន្លឹះភាសាឬមុខងារដែលមានស្រាប់។
- ការបញ្ជូនតម្លៃឬប៉ារ៉ាម៉ែត្រទៅអនុគមន៍នឹងត្រូវបានបញ្ចូលវានៅក្នុងវង់ក្រចក ( )។ ឧទាហរណ៍int myMultiplyFunction(int x, int y)

9.2. Return Type

ជាផ្នែកមួយនៃអនុគមន៍សម្រាប់ Return តំលៃលទ្ធផលត្រឡប់ទៅមកវិញ ដែលអាចជាប្រភេទ int ឬជា String។ ក្នុងករណីយើងមិនចង់ត្រលប់តំលៃអោយអនុគមន៍ទេយើងត្រូវ void ពីមុខឈ្មោះអនុគមន៍។

9.3. Function Body

រូបរាងនៃអនុគមន៍ត្រូវបានបង្កើតឡើងជាជាប្តូកនៃកូដ(code)សម្រាប់ដោះស្រាយបញ្ហាណាមួយទៅជាផ្នែក ឬក្រុមមួយ និងមានមុខងារផ្សេងៗពីគ្នា ដោយសរសេរវានៅក្នុងនៅចន្លោះដង្កៀប{}។

អនុវត្ត ១

Serial Monitor Output

អនុគមន៍ឈ្មោះ DashedLine( ) ត្រូវបានបង្កើតឡើងនៅក្នុងអនុគមន៍ void setup( ) ដូចបានបង្ហាញនៅក្នុងកូដខាងក្រោមហើយបន្ទាប់មកត្រូវហៅវាមកប្រើពីរដងដើម្បីធ្វើគួរសញ្ញា "----" Dashed Line ចំនួនពីរដង។

អនុវត្ត ២ អនុគមន៍ត្រលប់តំលៃ

10. Arduino Array និង Class

10.1. Array គឺជាសំនុំធាតុ រឺ វត្ថុដែលមានប្រភេទទិន្នន័យដូចគ្នា និង និងមានឈ្មោះរួមគ្នា និងទំហំដូចគ្នា ពេលផ្ទុកនៅក្នុង Memory។ ការប្រកាស Arrays ត្រូវចាប់ផ្តើមដោយឈ្មោះអថេរ និងសំនុំ Data។ យើងអាចមាន Array នៃចំនួនគត់ (ប្រភេទ int) ដែលជាចំនួនគត់ចំនួនពីរឬច្រើនដែលកើតឡើងពីមួយទៅមួយ។

ចំនុចសំខាន់នៅត្រង់នេះគឺ ថាធាតុនីមួយៗនៅក្នុងអារេត្រូវបានដាក់ដោយផ្ទាល់បន្ទាប់ពីធាតុមុនដែលអនុញ្ញាតឱ្យយើងចូលប្រើធាតុនីមួយៗជាវេនដោយប្រើរង្វិលជុំ។ ធាតុនៅក្នុងអារេមួយសំដៅទៅលើតម្លៃនីមួយៗនៅក្នុងអារេ។ ប្រសិនបើយើងមានអារេនៃចំនួនគត់បន្ទាប់មកលេខគត់នីមួយៗត្រូវបានគេសំដៅទៅលើធាតុនៃអារេ។

10.2. Class

Class នៅក្នុងភាសា C ជាប្លង់មេ(Blueprint) ឬពុម្ពគំរូដើមប្រើសម្រាប់បង្កើតនូវ Objects ផ្សេងៗដែលមានព័ត៌មានលម្អិតទាំងអស់នៃវត្ថុមួយ។ វាបណ្តុំនៃ Properties និង Method(function) ដែលទាំងពីរជាសមាជិកនៃ Class

ឧទាហរណ៍ ១: កូដខាងក្រោមបង្ហាញពីមូលដ្ឋានគ្រឹះនៃការប្រើ Array:

```
void setup( ) {
  int my_array[5]; // an array with 5 integer elements
  int i; // used as an index into the array
  Serial.begin(9600);
  my_array[0] = 2 // assign a value of 2 to the 1st element
  my_array[1] = 3 //assign a value of 2 to the 2nd  enlement
  my_array[2] = 4;
  my_array[3] = 5;
  my_array[4] = 6;
// display each number from the array in the serial monitor
  for (i = 0; i < 5; i++) {
    Serial.println(my_array[i]);
  }
}
void loop( ) {}
```

ឧទាហរណ៍ ២: ការបង្កើត Class

```
class LED_Controls {
  public:
    void Begin(byte GPIO_pin) {
```

```
    pinMode(GPIO_pin, OUTPUT);
  }
  void ON(byte GPIO_pin){
    digitalWrite(GPIO_pin, HIGH);
  }
  void OFF(byte GPIO_pin){
    digitalWrite(GPIO_pin,LOW);
  }
  void flash(byte GPIO_pin, int flash_speed, int flashes){
    for (int f=0; f < flashes; f = f +1){
      ON(GPIO_pin);
      delay(flash_speed);
      OFF(GPIO_pin);
      delay(flash_speed);
    }
  }
};
// Class Object
LED_Controls   LED;
void setup( ) {
 LED.Begin(LED_BUILTIN);
}
void loop( ) {
 LED.ON(LED_BUILTIN);
 delay(1000);
 LED.OFF(LED_BUILTIN);
 delay(1000);
 LED.flash(LED_BUILTIN,100,10);
}
```

ឧទាហរណ៍ 3: កូដខាងក្រោមបង្ហាញពីបង្កើត Objects នៅក្នុង Class សម្រាប់ធ្វើការចុច Key នៅលើ Keypad:

```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
//define the symbols on the buttons of the keypads
char Keys[ROWS][COLS] =
{
```

```
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {5,4,3,2};
byte colPins[COLS] = {9,8,7,6};
//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap( Keys ) ,rowPins, colPins,ROWS,COLS );
void setup( )
{
  Serial.begin( 9600 );
  Serial.println( "Please press the keyboard:" );
}
void  loop( )
{
  char key = customKeypad.getKey( );
  if( key!=NO_KEY )
  {
    Serial.print( "Key Value : " );
    Serial.println( key );
  }
}
```

ឧទាហរណ៍ 4: កូដខាងក្រោមបង្ហាញពីបង្កើត Objects នៅក្នុង Class សម្រាប់ធ្វើការ ដើម្បីធ្វើការបង្ហាញពាក្យ " Hello Word " នៅលើ Lcd:

```
#include <Wire.h>
#include <LiquidCrystal.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd( 0x27,16,2 ); // set the LCD address to 0x3F for a 16 chars and 2 line display
void setup( )
{
   //lcd.begin( );
   lcd.init( ); // initialize the lcd
   // Print a message to the LCD.
   lcd.backlight( );
```

```
    lcd.setCursor(0,0);
    lcd.print("I Love RUPP");
    lcd.setCursor(1,1);
    lcd.print("A3 Students");
}
void loop()
{
}
```

11. **តាសា** Micropython

នៅក្នុងមេរៀននេះជាដំបូងត្រូវស្វែងយល់ពីវិធីចាប់ផ្តើមជាមួយកម្មវិធីបង្កប់ **MicroPython** ប្រើប្រាស់ជាមួយនឹងប៉ត **ESP32 និង ESP8266**។ នៅក្នុងជំពូកនេះក៏ណែនាំអ្នកឱ្យស្គាល់ភាសា **Micropython** ដំបូងដោយត្រូវស្គាល់ពីភាពខុសគ្នារវាង **Micropython** និងភាសា **Python** ធម្មតានិងវិធីរៀបចំកម្មវិធីចូលទៅក្នុងប៉ត (**Board**)ទាំងពីរគឺ **ESP32** និង **ESP8266**។ សំខាន់ជាងនេះដឹងពីការសរសេរភាសា **Micropython** ដោយប្រើ **IDE** ផ្សេងៗ និងជាពិសេសការសរសេរកូដបញ្ជាសមាសភាគអេឡិចត្រូនិច និងឧបករណ៍វៃឆ្លាត។

11.1. **និយមន័យ** MicroPython

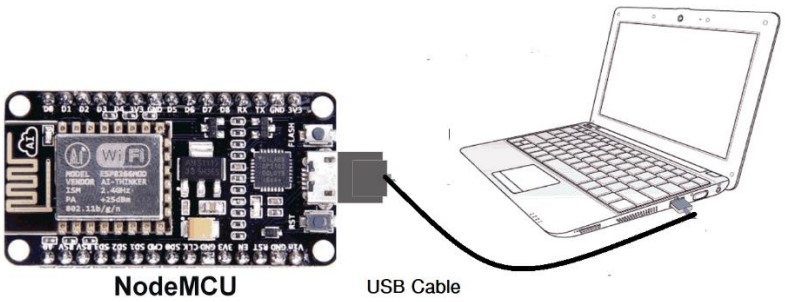
MicroPython គឺជាការអនុវត្តឡើងវិញនៃ Python 3 ដែលបានកំណត់គោលដៅសម្រាប់ធ្វើការត្រួតពិនិត្យឧបករណ៍ខ្នាតតូចនិងប្រព័ន្ធបង្កប់ឬយើងហៅថាប្រព័ន្ធ Embedded។ MicroPython គឺស្រដៀងនឹងពស់ថ្លាន់ធម្មតា។ ដូច្នេះប្រសិនបើអ្នកដឹងពីរបៀបសរសេរកម្មវិធី Python រួចហើយនោះអ្នកក៏ដឹងពីរបៀបសរសេរកម្មវិធីនៅក្នុង MicroPython ដែរ។

ក្រៅពីការលើកលែងមួយចំនួនលក្ខណៈភាសារបស់ Python ក៏មានផងដែរនៅក្នុង MicroPython។ ភាពខុសគ្នាធំបំផុតរវាង Python និង MicroPython គឺថា MicroPython ត្រូវបានរចនាឡើងដើម្បីធ្វើការក្រោមល័ក្ខខ័ណ្ឌដោយប្រើជាមួយនឹងប៉ត ESP32 ឬ ESP8266។ ដោយសារតែរឿងនោះ ហើយវាមិនមកជាមួយបណ្ណាល័យស្តង់ដារពេញលេញទេ។ វារួមបញ្ចូលទាំងសំណុំតូចនៃបណ្ណាល័យស្តង់ដារ Python។ ទោះយ៉ាងណាក៏ដោយវារួមបញ្ចូលម៉ូឌុលដើម្បីចូលប្រើផ្នែករឹងកម្រិតទាប - នេះមានន័យថាមានបណ្ណាល័យដែលអាចចូលដំណើរការបានយ៉ាងងាយស្រួល និងធ្វើអន្តរកម្មជាមួយ GPIOs។

11.2. កម្មវិធីក្នុង MicroPython ដំបូង

Python គឺជាភាសាសរសេរកម្មវិធីមួយដែលត្រូវបានគេប្រើយ៉ាងទូលំទូលាយសាមញ្ញនិងងាយរៀនបំផុត។ ដូច្នេះការលេចចេញនូវ Micropython ធ្វើឱ្យមានភាពងាយស្រួលនិងសាមញ្ញបំផុតក្នុងការរៀបចំកម្មវិធីអេឡិចត្រូនិចឌីជីថល។ ប្រសិនបើអ្នកមិនដែលរៀបចំកម្មវិធីអេឡិចត្រូនិកឌីជីថលពីមុនមកទេនោះ MicroPython គឺជាចំណុចចាប់ផ្តើមដ៏ល្អមួយ។ គោលដៅរបស់ Micropython គឺដើម្បីធ្វើឱ្យកម្មវិធីអេឡិចត្រូនិចឌីជីថលមានលក្ខណៈសាមញ្ញតាមដែលអាចធ្វើទៅបានដូច្នេះវាអាចត្រូវបានប្រើដោយនរណាម្នាក់។ បច្ចុប្បន្ន MicroPython ត្រូវបានប្រើដោយចំណង់ចំណូលចិត្តអ្នកស្រាវជ្រាវគ្រូអ្នកអប់រំនិងសូម្បីតែនៅក្នុងផលិតផលពាណិជ្ជកម្ម។ ខាងក្រោមកូដសម្រាប់ការភ្លឺបភ្លែតៗ LED នៅលើ ESP ៣២ ឬអ៊ី ESP៨២៦៦ ដែលសរសេរវានៅក្នុង Thonny IDE ៖

11.3. ការភ្ជាប់ NodeMCU ទៅនឹងកុំព្យូទ័រ: ប្រើខ្សែ USB Macro-B ជាប្រភេទខ្សែទូរសព្ទ Sumsung ឬខ្សែទូរសព្ទដទៃទៀត។ នៅលើប៉តខាងក្រោមមាន LED មួយសម្រាប់អោយយើងតេសដោយប្រើ Pin:D4 ទៅប្រើនៃ Board។



11.4. ការសរសេរកូដ

ចូលកម្មវិធី Thonny និងមុនសរសេរកូដយើងត្រូវជ្រើសរើឈ្មោះ Board : ESP32 ឬ ESP8266 ដែលបានភ្ជាប់ជាមួយនឹងកុំព្យូទ័រ។ បន្ទាប់ជ្រើសលេខ Port និងតំឡើង Firmware វិសិន ។ ក្នុងកម្មវិធី Thonny IDE ចុចលើ Tool Menu→Option→Interpreter រួចជ្រើសតាមតាមរូប ខាងក្រោមនេះ :

របៀបទី ១

```
>>> from machine import Pin #ពី machine ម៉ូឌូល បញ្ចូលប្រើ Pin Class
>>> led=Pin(2, Pin.OUT) #បង្កើត Led Object នៃ Pin Class ដោយ Pin 2 និងកំណត់ជា Output
>>> led.value(1) #កំណត់អោយអំពូលរលត់
>>> led.value(0) ) #កំណត់អោយអំពូលភ្លឺង
```

របៀបទី ២

```
from machine import Pin #ពី machine  ម៉ូឌុល បញ្ចូលប្រើ Pin Class
    from time import sleep  #ពី time ម៉ូឌុល បញ្ចូលប្រើ sleep Class
    led = Pin(2, Pin.OUT)  # create led object of Pin class on GPIO5(d1) and set as
    an output
    while True:
      led.value(not led.value()) #កំណត់អោយ LED ON/OFF
      sleep(1) # អោយ LED បិត ឬបើករយៈពេល មួយវិនាទី
```

ចំពោះការតេសដំបូងយើងប្រើ LED ដែលមានស្រាប់នៅលើ Board ដោយផ្ទាល់ហើយ Pin អាចលេខ ២ ឬលេខ ០ អាស្រ័យលើយើងប្រើប៉តដែលយើងកំពុងប្រើ។

របៀបទី 3: លេខកូដខាងលើយើងអាចសរសេរម៉្យាងទៀតតាមករណីដូចខាងក្រោម:

```
#from machine module importing Pin class
from machine import Pin
from time import sleep
#create led object of Pin class on GPIO5(d1) and set as an output
led = Pin(5, Pin.OUT)
while True:
   led.value(1) #Turn on Led
   sleep(1)     #delay 1 menute
   led.value(0) #Turn on Led
   sleep(1)     #delay 1 menute
```

- កូដទាំងខាងលើដំណើរការនៅលើកុំព្យូទ័រ ហើយកូដនេះមិនបានបញ្ចូលវានៅក Memory នៃ NodeMCU ឡើយ។ដើម្បីបញ្ចូលកូដខាងលើទៅក្នុង Memory នៃ ESP32 ឬ ESP8266 ត្រូវ អនុវត្តតាមជំហានដូចខាងក្រោម:

11.5. MicroPython – Boards support

វាជាប្រភេទ Open source ដោយប្រើកម្មវិធីបង្កប់ដែលហៅថា Firmware និង ការអភិវឌ្ឍន៍ត្រូវបានកំណត់នូវគោលដៅជាពិសេសសម្រាប់កម្មវិធីដែលមានមូលដ្ឋានលើការប្រើប្រាស់ IoT។ វារួមបញ្ចូលទាំងកម្មវិធីបង្កប់ដែលដំណើរការលើប្រព័ន្ធ Esp8266 Wifi_SoC ពីប្រព័ន្ធ Espressif និងផ្នែករឹងដែលផ្អែកលើម៉ូឌុល ESP-12។ NodeMCU 8266 ប្រើ CPU ឈ្មោះ **ESP8266**(LX106) និងប្រព័ន្ធប្រការ XTOS ដែលប្រភេទ Microcontrollers និងមានមុខងារប្រើដូច Arduion Un0 ឬ Arduino Nano ដែរ។

11.6. NodeMCU ESP8266

វាជាប្រភេទ Open source ដោយប្រើកម្មវិធីបង្កប់ដែលហៅថា Firmware និង ការអភិវឌ្ឍន៍ត្រូវបានកំណត់នូវ គោលដៅជាពិសេសសម្រាប់កម្មវិធីដែលមានមូលដ្ឋានលើការប្រើប្រាស់ Internet of think(IoT)។

វាអាចបញ្ចូលកម្មវិធី បង្កប់ដែលដំណើរការលើប្រព័ន្ធ Esp8266 Wifi_SoC ពីប្រព័ន្ធ Espressif និង ផ្នែករឹងដែលផ្អែកលើម៉ូឌុល ESP-12។ NodeMCU 8266 ប្រើ CPU ឈ្មោះ **ESP8266**(LX106) និងប្រព័ន្ធ ប្រភារ XTOS ដែលប្រភេទ Microcontrollers និងមានមុខងារប្រើដូច Arduion Uno ឬ Arduino Nano ដែរ។

11.7. ការប្រើប្រាស់ GPIO(General Purpose input and output) **pin**

- NodeMCU ESP8266 ប្រើប្រាស់ GPIO Pin មានចំនួន ១៧ ដែលមានមុខងារសម្រាប់បញ្ចេញ ឬបញ្ចូលតំលៃ 0 និង ១ និង Pin ខ្លះទៀតអាចប្រើជា I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button។

- Power មាន Pin GND ចំនួនបួន និង Pin Vcc ប្រើភ្លើង 3.3V ចំនួនពីរ

- Vin អាចផ្តល់ភ្លើង ៥ វុល និងអាចផ្តល់ថាមពលដល់ NodeMCU ទាំងមូលរវាង 7V-12V ដែលមាន រហូតដល់ 1A។

- នៅលើ Board មានជើងចាប់ពី Do ដល់ D8 ប្រើសម្រាប់បញ្ចេញ ឬបញ្ចូលតំលៃលេខ 0 និង ១។ ជើងទាំងនេះប្រើសម្រាប់ភ្ជាប់ទៅឧបករណ៍ផ្សេងៗទៀតដូចជា LED, Sensor, LCD។ល។

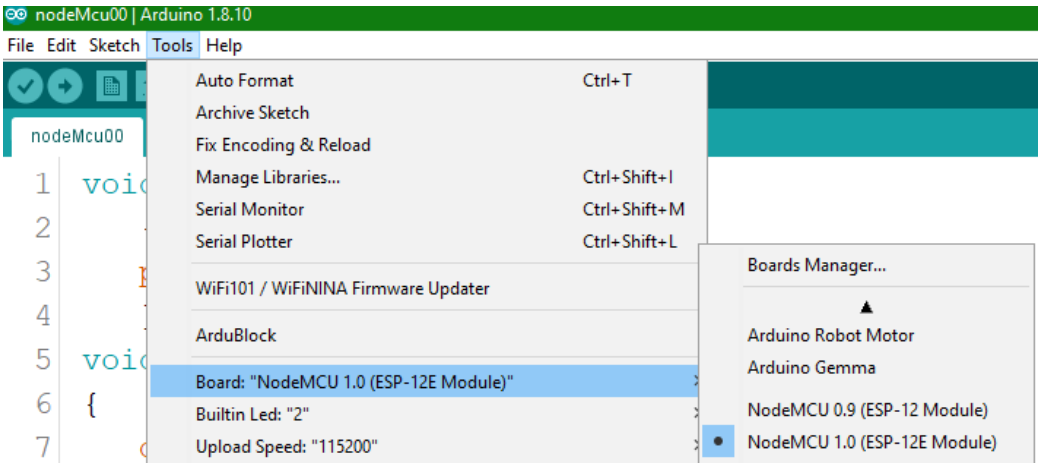
11.8. ការប្រើប្រាស់ Led មួយដាមួយកម្មវិធី Arduino *និងភាសា* Python

យើងអាចប្រើអំពូល LED មួយនៅលើNodeMCUដោយផ្ទាល់សម្រាប់ការធ្វើតេស្តការងារដំបូង។ ដូចដែលបានរៀបរាប់នៅមេរៀនមុនដោយប្រើ Pin D0 ឬ Pin D4 ប្រើទៅតាមប្រភេទនៃ Board ដើម្បីអោយភ្លើងពណ៌ខៀវបើក ឬបិទ ដែលមាននៅលើ Board ស្រាប់។ ដើម្បីធ្វើការងារនេះចូលបើកកម្មវិធី Arduino IDE ត្រូវសរសេរកូដដូចបានបង្ហាញខាងក្រោមដោយប្រើ Pin D4។

ប្រើភាសា Arduino

```
void setup( )
{
        pinMode( D4, OUTPUT );//set D4 pin as output
}
void loop( )
{
        digitalWrite( D4, HIGH );//Turn On LED
        delay( 500 );
        digitalWrite( D4, LOW ); // Turn OFF LED
        delay( 500 ) };
```

ក្រោយពីបានសរសេរកូដខាងលើរួច បន្ទាប់មកយក Mouse ចុចលើ **Tools** និងជ្រើសរើសយក Boards: **Node MCU 1.0 (ESP-12 Module) *សូមមើល***តាមរូបខាងក្រោម និង បើមិនមានឈ្មោះ Board នេះទេយើងត្រូវបញ្ចូលវា។



- ប្រើភាសា Python

```
#from machine module importing Pin class
from machine import Pin
from time import sleep
#create led object of Pin class on GPIO5(d1) and set as an output
led = Pin(5, Pin.OUT)
while True:
    led.value(1) #Turn on Led
    sleep(1)     #delay 1 menute
    led.value(0) #Turn on Led

    sleep(1)     #delay 1 menute
```

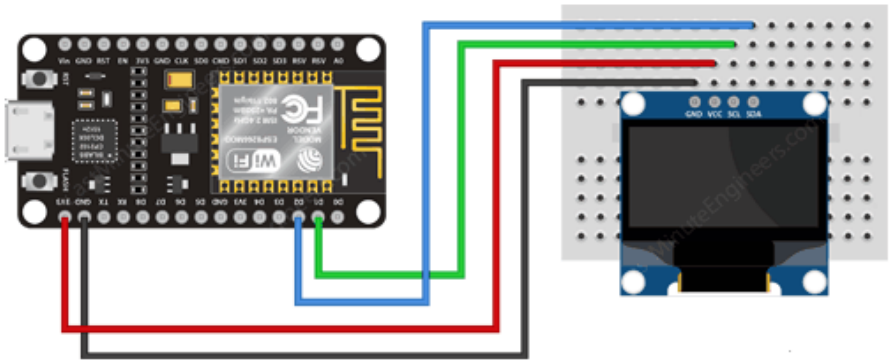
11.9. ការប្រើ OLED with ESP8266 NodeMCU

OLED ជាឧបករណ៍ធ្វើការបង្ហាញនូវព័ត៌មានដូចដែលយើងប្រើ LCD ដែរនៅមេរៀនមុន។ តែវាមានទំហំតូចជាង LCD ហើយមាន Rows និង Columns ច្រើនជាង។

OLED module មាន SSD1306 driver បានបង្កើតនៅក្នុងខ្លួនវាទំហំ 1KB GDDRAM។ វាសម្រាប់ធ្វើអេក្រង់ផ្ទុក bit patterm ដែលត្រូវបង្ហាញព័ត៌មាន។ មានផ្នែ Memory 1K ដោយបែងចែកទៅ 8 ទំព័រ (ពី 0 ទៅ 7) និង ទំព័រនិមួយៗមាន 128 columns/segments (block 0 ទៅ 127). និងមាន column និមួយៗ អាចផ្ទុក 8 bits data (ពី 0 ទៅ 7).

11.10. ការតភ្ជាប់ OLED with ESP8266 NodeMCU

1. ធ្វើការតភ្ជាប់រវាង NodeMCU និង OLED តាមរូបខាងក្រោម៖



2. ចូរសរសេរកូដនៅ IDE Arduino

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
Adafruit_SSD1306 display(-1);
void setup()
{
        // initialize with the I2C addr 0x3C
        display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
        // Clear the buffer.
        display.clearDisplay();
        // Display Text
        display.setTextSize(1);
        display.setTextColor(WHITE);
        display.setCursor(0,28);
        display.println("Hello world!");
display.display();
delay(2000);
display.clearDisplay();
// Display Inverted Text
display.setTextColor(BLACK, WHITE); // 'inverted' text
display.setCursor(0,28);
display.println("Hello world!");
display.display();
delay(2000);
display.clearDisplay();
// Changing Font Size
display.setTextColor(WHITE);
display.setCursor(0,24);
display.setTextSize(2);
display.println("Hello!");
display.display();
delay(2000);
display.clearDisplay();
// Display Numbers
```

```
display.setTextSize(1);
display.setCursor(0,28);
display.println(123456789);
display.display( );
delay(2000);
display.clearDisplay( );
display.setCursor(0,28);
display.print("0x"); display.print(0xFF, HEX);
display.print("(HEX) = ");
display.print(0xFF, DEC);
display.println("(DEC)");
display.display( );
delay(2000);
display.clearDisplay( );
// Display ASCII Characters
display.setCursor(0,24);
display.setTextSize(2);
display.write(3);
display.display( );
delay(2000);
display.clearDisplay( );
// Scroll full screen
display.setCursor(0,0);
display.setTextSize(1);
display.println("Full");
display.println("screen");
display.println("scrolling!");
display.display( );
display.startscrollright(0x00, 0x07);
delay(2000);
display.stopscroll( );
delay(1000);
display.startscrollleft(0x00, 0x07);
delay(2000);
display.stopscroll( );
delay(1000);
```

```
display.startscrolldiagright(0x00, 0x07);
delay(2000);
display.startscrolldiagleft(0x00, 0x07);
delay(2000);
display.stopscroll( );
display.clearDisplay( );
// Scroll part of the screen
display.setCursor(0,0);
display.setTextSize(1);
display.println("Scroll");
display.println("some part");
display.println("of the screen.");
display.display( );
display.startscrollright(0x00, 0x00) };
```

សំណួរ និង លំហាត់

I. 1. ដូចម្តេចដែលហៅថា(a)Python?; (b).គេប្រើវាសម្រាប់ធ្វើអ្វី?

2. ចូរនិយាយពីភាពខុសគ្នារវាង Compiling and Interpreting?

3. តើ input ជា function ប្រើសម្រាប់ធ្វើអ្វី? ចូរអោយឧទាហរណ៍ពីការប្រើអនុគមន៍នេះ?

4. ចូរពន្យល់ Statements និមួយៗខាងក្រោម៖

```
1. import RPi.GPIO as GPIO
2.GPIO.setmode(GPIO.BOARD)
3.GPIO.setup(3, GPIO.OUT)
4.GPIO.output(3, True)
```

II. ចូរសរសេរ python program ដោយប្រើប្រាស់ while និង for loop ដើម្បីគណនា៖

1) $p = 1 * 2 * 3 * ... * 10$
2) $sum = 2 + 4 + 6 + 8 + \cdots + n$
3) $sum = 1 + 3 + 5 + \cdots + n$

III. ចូរសរសេរ *python program* ដើម្បី*:*

1) ឬសនៃសមីការទី ១ និង ទី ២

2) រក Max នៃ n ចំនួ

IV. ចូរសរសេរ *python program* ដើម្បី*:*

1) បង្ហាញពាក្យ *"Hello World"* នៅ *LCD*

2) ពេលចុច $Button$ ដើម្បីអោយ LED មួយ ON/OF

សេចក្តីសន្ទេប

សៀវភៅនេះគឺតម្រូវ**ឱ្យ**អ្នកសិក្សារៀនយ៉ាងលម្អិតទៅស្តាបត្យកម្មស៊ីភីយូ(CPU) អង្គចងចាំ និង ឧបករណ៍បញ្ចេញឬបញ្ចូលរបស់ប្រព័ន្ធកុំព្យូទ័រ និង IoT។ វាផ្តល់នូវការយល់ដឹងអំពីបច្ចេកទេសពីការរចនាអំពីផ្នែករឹង និងផ្នែកទន់របស់កុំព្យូទ័រទំនើប និងបច្ចេកវិទ្យាថ្មីនាពេលបច្ចុប្បន្ននេះ។ វាផ្តោតសំខាន់ទៅលើការប្រើប្រាស់ប្រព័ន្ធគោល លេខកូដ និងស្តាបត្យកម្មប្រើប្រាស់នៅក្នុងអង្គចងចាំស៊ីភីយូ និង ភាសា **Assembly** ក្នុងគ្រួសារស៊ីភីយូ **X86 និង X64 bits** ព្រមទាំងស្តាបត្យកម្ម **Microcontrollers** និង ភាសា **Arduino** ក្នុងគ្រួសារ **AVR និង ESP32**។ អោយអ្នកសិស្សាចេះបង្កើតឧបករណ៍អេឡិចត្រូនិក, រ៉ូបូត និងឧបករណ៍ឆ្លាតវៃដែលមានការដំឡើងផ្នែករឹង និងសរសេរកូដដើម្បីអោយឧបករណ៍ទាំងនោះដំណើរការ។ ជាងនេះទៅទៀតការអោយមានចំណេះដឹងក្នុងការបង្កើត App សម្រាប់ប្រើប្រាស់នៅទូរសព្ទដោយប្រើ App Inventor ដែលផ្តល់ភាពងាយស្រួលដល់អ្នកចាប់ផ្តើមបង្កើតកម្មវិធីដំបូង។ អ្នកសិក្សាចេះប្រើប្រាស់ប្រព័ន្ធ IoT ដែលពិពណ៌នាអំពីបណ្តាញនៃវត្ថុរូបធាតុដែលត្រូវបានដំណើរការដោយសែនស័រ, ផ្នែកទន់ និងបច្ចេកវិទ្យាផ្សេងទៀតក្នុងគោលបំណងតភ្ជាប់គ្នា និងផ្លាស់ប្តូរទិន្នន័យជាមួយឧបករណ៍ និងប្រព័ន្ធដទៃទៀតនៅលើបណ្តាញអ៊ីនធឺណិត។ សៀវភៅនេះមានចំនួន៨មេរៀនត និងខ្លឹមសារសង្ខេបដូចខាងក្រោម៖

- **មេរៀនទី ១**: ការប្រើប្រាស់កូដជា Scan និង ASCII កូដ តំណាងអោយអក្សរ, លេខ និង និមិត្តសញ្ញាសម្រាប់ប្រើប្រាស់នៅក្នុង Keyboard និង Memory(RAM) សម្រាប់យកវាទៅធ្វើប្រមាណវិធី និងសរសេរកូដ។ ការធ្វើប្រមាណវិធីតក្កវិទ្យាជាមួយសមាសភាគអេឡិចត្រូនិចក៏រឹតតែសំខាន់ទៅទៀតដោយយកទៅកសាងជាស្នៀគ្វីនៃផ្នែករឹងកុំព្យូទ័រ, រ៉ូបូត និងឧបករណ៍ឆ្លាតវៃផ្សេងៗទៀតមុននឹងបង្កើតឧបករណ៍ជាក់លាក់មួយ។ យល់ពីប្រមាណវិធីនព្វន្ធ និងទសភាគក្នុងកុំព្យូទ័រ។
- **មេរៀនទី ២**: អ្នកសិក្សានឹងយល់ពីការ Microprocessor **និង** Microcontroller ដែលវាទាំងពីរជាខួរក្បាលរបស់កុំព្យូទ័រ, ទូរស័ព្ទដៃ, រ៉ូប៉ូតទឹក, ឧបករណ៍វៃឆ្លាត និងឧបករណ៍អេឡិចត្រូនិចដែលវាទាំងពីរអាចដំណើរការទៅបានដោយសារពាក្យបញ្ជាជាអ្នកប្រាប់។ ហើយសំនុំពាក្យបញ្ជានេះ វាគឺជាកូដដែលនិស្សិតបានសរសេរដោយប្រើភាសា **Assembly, Arudino** និង **Python** និងដាក់បញ្ចូលវាទៅឈីបខាងលើ។
- **មេរៀនទី ៣**: អោយនិស្សិតយល់ពី X86 គឺជាឈ្មោះតំណាងអោយស៊េរីនៃឈ្មោះ CPU ដើម្បីបង្កើតជា Main Board ដែលត្រូវបានប្រើនៅក្នុងកុំព្យូទ័រលើតុ និងកុំព្យូទ័រយួរដៃភាគច្រើន។ ព្រមទាំងកុំព្យូទ័រជា Workstation និងម៉ាស៊ីនមេ(Servers) នៅសតវត្សរ៍ទី ២១។ យល់ពីការប្រើប្រាស់សំនុំផ្នែករឹងក្នុងស៊ីភីយូ និង ពាក្យបញ្ជារបស់វាដើម្បីធ្វើប្រមាណវិធី និង សរសេរកូដដោយផ្ទាល់ដោយប្រើភាសា Assembly និង Hex File។
- **មេរៀនទី ៤**: ស៊ីភីយូអាចដំណើរការយកទិន្នន័យ(operands)ផ្សេងៗ ការរកតម្លៃ Machine code និងការប្រើប្រាស់កម្មវិធី Debug.
- **មេរៀនទី ៥**: X86Asm ជាកម្មវិធីសម្រាប់សរសេរកូដភាសា Assembly មួយដោយប្រើទំហំ 8, 16 និង 32_bit Registers និងប្រើពាក្យបញ្ជារបស់ស៊ីភីយូចាប់ពីលេខ 8086 រហូតដល់ស៊ីភីយូបច្ចុប្បន្ននេះ និងប្រើ Interrupts ដែលជាអនុគមន៍ផ្ទាល់ប្រើជាមួយនឹងភាសា Assembly។
- **មេរៀនទី ៦**: គំរូកម្មវិធី(Programming Model)ចាប់ពី 8086 រហូតដល់ Pentium Pro គឺគេគិតទៅលើផ្នែកកម្មវិធីអាចមើលឃើញ(Program Visible) ដែលបណ្តាលឱ្យ Registers របស់វាអាចប្រើបាននៅក្នុងកម្មវិធី និង កំណត់ប្រើដោយពាក្យបញ្ជាដែលអាចដំណើរការប្រមាណវិធីផ្សេងៗឬដំណើរការអ្វីមួយ។

ជាងនេះទៅទៀតចាប់ពីស៊ីភីយូ 80286 ឡើងទៅមានកម្មវិធីអាចមើលឃើញប្រើសម្រាប់ត្រួតពិនិត្យ និង ប្រតិបត្តិអង្គចងចាំ(Memory)ហៅថាប្រព័ន្ធការពារអង្គចងចាំ(Protected Memory System)។

- **មេរៀនទី ៧**: យល់ពីការប្រើប្រាស់ Microcontrollers នៅលើឧបករណ៍អេឡិចត្រូវបានយើងប្រើប្រាស់វាជាច្រើននៅក្នុងជីវភាពប្រចាំថ្ងៃ។ ពីការប្រើប្រភេទ Arduino Board ដែលជាបន្ទះសៀគ្វីសម្រាប់តភ្ជាប់ដោយផ្ទាល់ជាមួយនឹង switches, buttons, LCD displays, LEDS, Digital Electronic, Relays, Sensor, Keypad, camera និង serial ports។ ចុងក្រោយក្នុង មេរៀននេះអោយនិស្សិតយល់ពីការសរសេរកូដដំបូងនៅលើ Editor ផ្សេងៗ មានដូចជា Arduino IDE, UnoArduSim, Python IDE និង ATMEL STUDIO IDE និងដឹងពីការបញ្ចូលកូដទៅក្នុងឈីបខាងលើដែលបានសរសេរនៅលើកុំព្យូទ័រ។
- **មេរៀនទី ៨**: ការសរសេរកម្មវិធី Arduino ត្រូវប្រើភាសា C/C++ និង Assembly ដែលជាមូលដ្ឋានគ្រឹះយ៉ាងសំខាន់ដើម្បីបង្កើតកម្មវិធីធ្វើការបញ្ជា និងកុងត្រូលទៅលើផ្នែករឹងរបស់ Arduino និងឧបករណ៍ដទៃទៀតដែលបានតភ្ជាប់ជាមួយនឹងវាអោយដំណើរការទៅបាន។ ការសរសេរកម្មវិធីគឺផ្សំឡើងពីអនុគមន៍, តំលៃ(variable និង constants) និង រចនាសម្ព័ន្ធនៃ Arduino Board។
- **មេរៀនទី ៩**: នៅក្នុងជំពូកនេះយើងនឹងសិក្សាអំពីការបង្កើតឧបករណ៍អេឡិចត្រូនិក និងឧបករណ៍ឆ្លាតវៃដែលជាគម្រោងងគំរូមានការប្រើប្រាស់ និងការដំឡើងឧបករណ៍ជាក់ស្តែងអោយទៅជាឧបករណ៍មួយអាចដែលអាចប្រើបាននៅក្នុងផ្ទះ, សាលារៀន ទីសាធារណៈ ឬកន្លែងដទៃផ្សេងទៀត។ រៀនពីការបង្កើត RC car ដែលរ៉ូបូតដំបូង និង ជាគ្រឹះសម្រាប់ធ្វើការបង្កើតរ៉ូបូតធំៗផ្សេងៗ ដូចជារ៉ូបូតបោសផ្ទះ, រាវមីន, កាត់ស្មៅ ឬរ៉ូបូតប្រកួតជាដើម។ ចំនុចបន្ទាប់គឺការសរសេរកូដដើម្បីបញ្ជាទៅលើឧបករណ៍ឆ្លាតវៃ និង រ៉ូបូតដែលបានបង្កើតរួចហើយអោយដំណើរការតាមការបញ្ជារបស់យើង និងដំណើការដោយស្វ័យប្រវត្តិ។
- **មេរៀនទី ១០**: និស្សិតយល់ពីការបង្កើតកម្មវិធីដំបូងប្រើនៅលើទូរសព្ទដោយប្រើ App Inventor។ ដោយវាដំណើរការនៅលើគេហទំព័រអ៊ីនធឺណិត និងមិនត្រូវការឲ្យអ្នកប្រើប្រាស់ ទាញយក និងដំឡើងឡើយ។ ប្រើបណ្តុំលក្ខខណ្ឌ និងពាក្យបញ្ជាហៅថាប្លុក(Blocks) ជំនួសការសរសេរកូដ និងពុំត្រូវការឲ្យអ្នកប្រើប្រាស់ចេះភាសាសរសេរកម្មវិធីអ្វីឡើយ។ មាន Interfaces សាមញ្ញ និងងាយយល់មានកម្មវិធីជំនួយនៅលើទូរស័ព្ទ Android ដែលអនុញ្ញាត ឲ្យអ្នកប្រើប្រាស់ធ្វើការតេស្តសាកល្បងកម្មវិធីភ្លាមៗ និងរហ័ស។
- **មេរៀនទី ១១**: យល់ពីការប្រើ IOT សម្រាប់ធ្វើការធ្វើការបញ្ជា(Remote) ទៅឧបករណ៍ជាច្រើនដោយប្រើប្រាស់បច្ចេកវិទ្យាអ៊ីនធឺណិត ពីឧបករណ៍អេឡិចត្រូនិចនិងឧបករណ៍មួយចំនួនទៀតដោយប្រើ Laptop, Phone, Ipad ជាដើម។ វាពិពណ៌នាអំពីបណ្តាញនៃវត្ថុរូបធាតុ "Things" ដែលត្រូវបានដំណើរការដោយ Sensor, ផ្នែកទន់និងបច្ចេកវិទ្យាផ្សេងទៀតក្នុងគោលបំណងតភ្ជាប់គ្នានិងផ្លាស់ប្តូរទិន្នន័យជាមួយឧបករណ៍និងប្រព័ន្ធដទៃទៀតនៅលើអ៊ីនធឺណិត។
- **មេរៀនទី ១២: Python** ជាភាសាកម្មវិធីកម្រិតខ្ពស់មួយដែលមានភាពពេញនិយមបំផុតសំរាប់ការសរសេរកម្មវិធីនាពេលបច្ចុប្បន្ននេះ។ វាមានភាពងាយស្រួលក្នុងការសរសេរកម្មវិធីដែលមានភាពស្មុគស្មាញបានលឿនជាជាងការប្រើភាសាដទៃទៀត។ យើងប្រើវាសម្រាប់គោលបំណងទូទៅ (General Purpose) និងមានលក្ខណៈជា Dynamic System ហើយវាគ្រប់គ្រងលើ Memory ដោយស្វ័យប្រវត្តិ។ វា support ទម្រង់ programming ជាច្រើន ដូចជា OOP, multi_pursose និងភ្ជាប់មកជាមួយ standard library មួយចំនួនផងដែរ។ វាអាចប្រើប្រាស់បានជាមួយប្រព័ន្ធប្រតិបត្តិការជាច្រើនដូចជា Windows, Linux, និង MacOS។

ឯកសារពិគ្រោះ

1. Kip R. Irvine, "Assembly Language For X86" Copyright © 2011, 2007, 2003, 1999 by Pearson Education, Inc., Upper Saddle River, New Jersey 07458..

2.“Arduino in Action” by Martin Evans , JOSHUA NOBLE and JORDAN HOCHENBAUM