|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FAKULTI TEKNOLOGI KEJURUTERAAN**  **ELEKTRIK DAN ELEKTRONIK**  **UNIVERSITI TEKNIKAL MALAYSIA MELAKA** | | | | | |
| **COMPUTER ORGANIZATION AND ARCHITECHTURE** | | | | | |
| BEEC 2373 | | | SEMESTER 2 | SESI 2019/2020 | |
| LAB 1: INTRODUCTION TO 8086 EMULATOR | | | | | |
| **NO.** | **STUDENTS' NAME** | | | | **MATRIC. NO.** |
| **1.** | **AHMAD IRFAN BIN HARMAN** | | | | **B081910068** |
| **2.** | **CHE IZZATI AYUNI BINTI CHE ROHIM** | | | | **B081910021** |
| **3.** |  | | | |  |
| **PROGRAMME** | | **1BEEC** | | | |
| **SECTION / GROUP** | | **S1/1** | | | |
| **DATE** | | **18 FEBRUARY 2020** | | | |
| **NAME OF INSTRUCTOR(S)** | | 1. **EN. NOOR MOHD ARIFF BIN BRAHIN** | | | |
|  | | | |
| **EXAMINER’S COMMENT(S)** | | | | **TOTAL MARKS** | |

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. No. | Date | Author(s) | Description |
| 1.0 | 12 FEB 2020 | 1. Noor Mohd Ariff 2. Ahmad Nizamudin | 1. Update to new UTeM logo 2. Update faculty's name 3. Change "course" to "programme" 4. Remove verification stamp |
|  |  |  |  |
|  |  |  |  |

## OBJECTIVES

* To write an 8086 assembly language programming to execute basic devices.
* To become familiar in using simulation software for 8086.
* To implement a simple application based on simulation.

## EQUIPMENT/COMPONENTS

1. Personal computer installed with 8086 Software.

## SYNOPSIS & THEORY

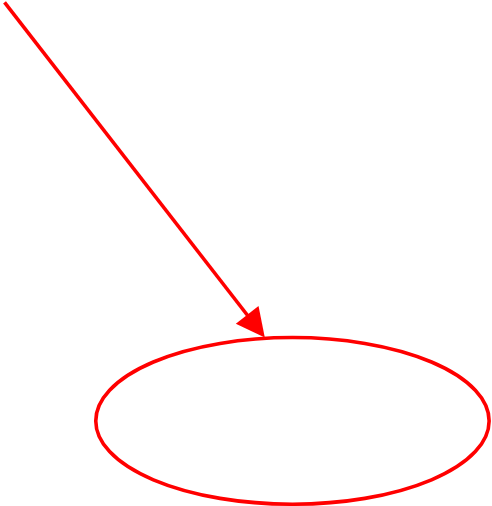
In this experiment, students will get an introduction to the 8086 assembly language program development process. Through a simple example provided below, students will learn how to use edit, assemble and simulate programs using the emu8086 suite.

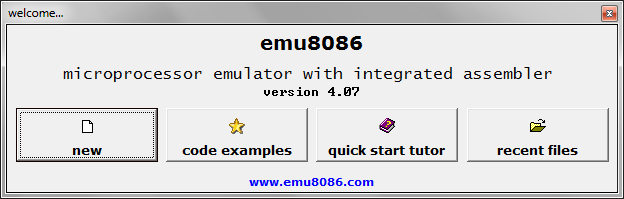
The emu8086 cross-assembler and simulator allows you to write a 8086 assembly language program, assemble it into 8086 binary code on a PC, and then execute this code as if you were running it on a real 8086 microprocessor. The simulator allows you to execute a single instruction at a time, and to observe the state of simulated registers after the execution of each instruction.

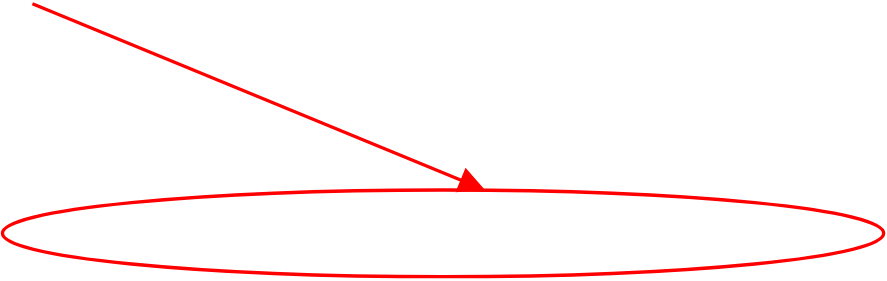
1. **PROCEDURE**

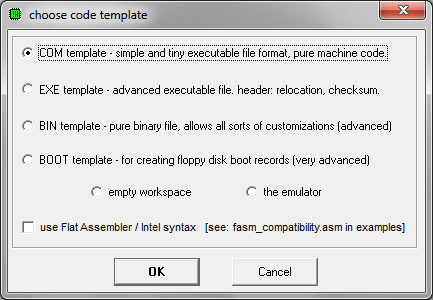
## Introduction To The Emulator8086 Assembly Language Tool

1. To begin a session, run the program emu8086.exe. This is a text editor for 8086 assembly language programs. When you run emu8086 you see the following dialog box. Select ‘new’ tab.

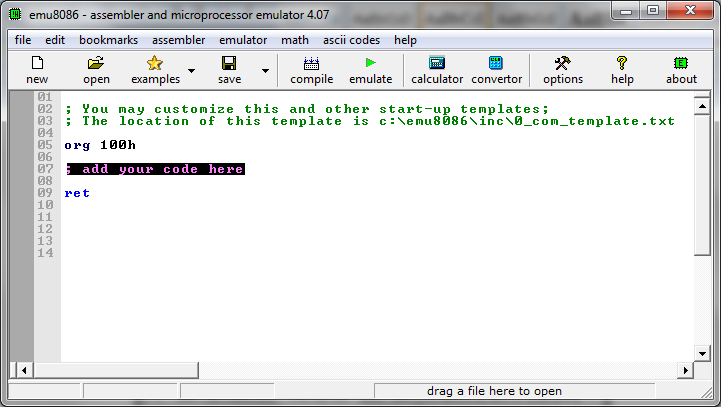




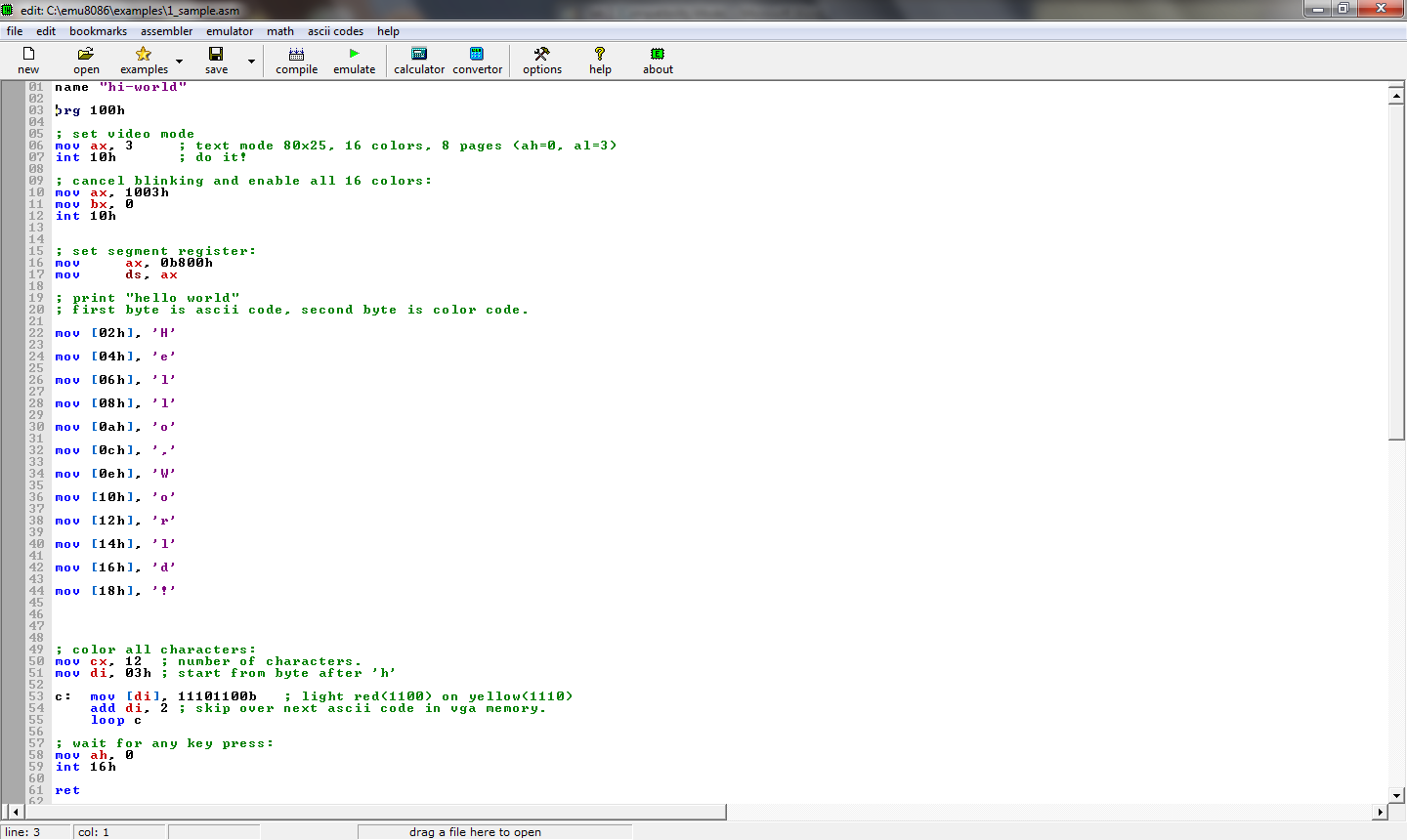
1. Then choose code template.

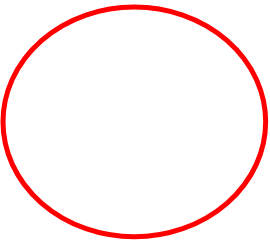


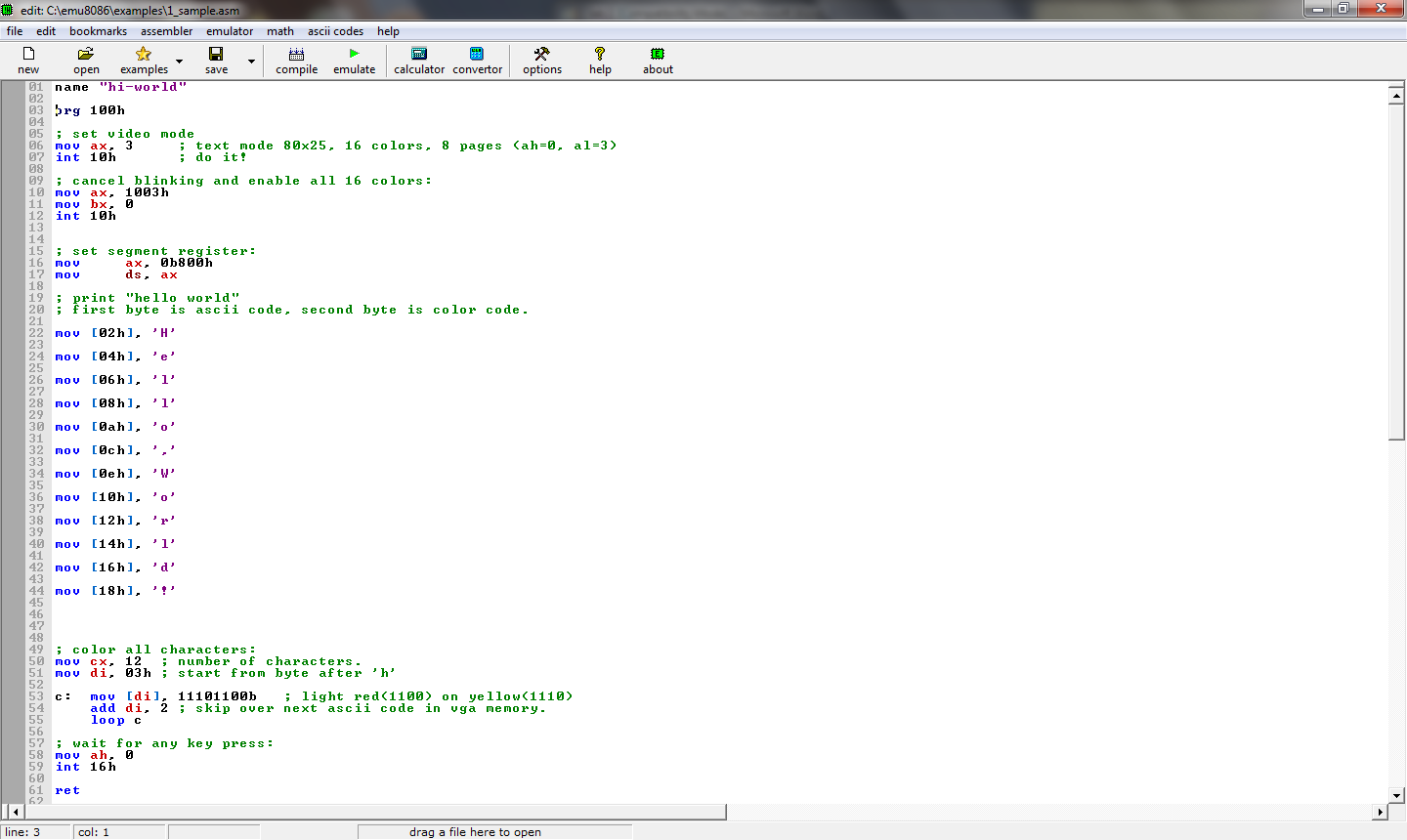
1. This is the defaults code. You can erase this code and start again or you can use it. Note that the provided code gives you the beginning and end of a program.



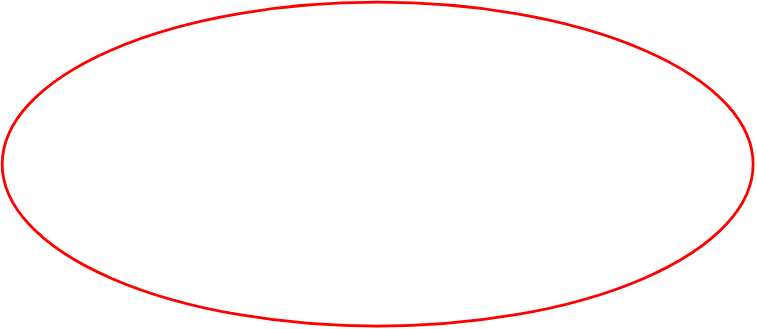
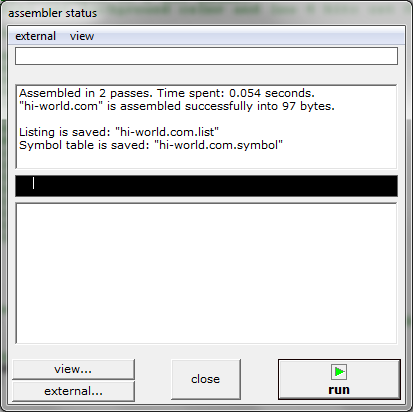
1. Consider compiling the example program to display ‘Hello World’ below.

****

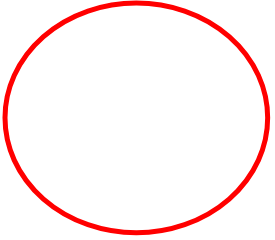
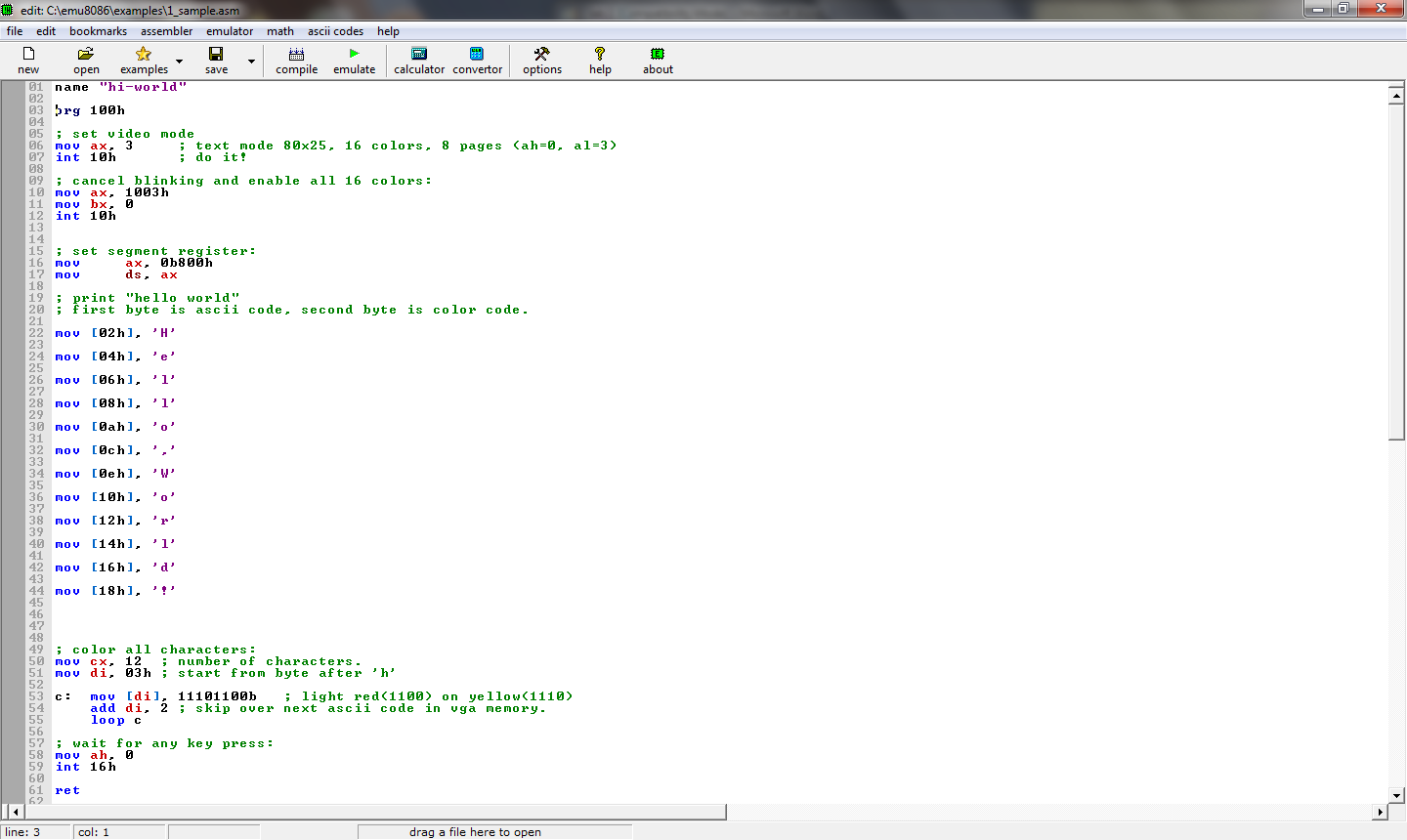
1. Click on the Assembler tab to get the ‘compile' menu or you can select the ‘compile’ icon in the tool bar. Select this. This will invoke the assembler. The code will either assemble successfully with no errors, or unsuccessfully with one or more errors. If you have an error, you have to re-edit your code and then assemble it again. You repeat this step until you have no errors. 

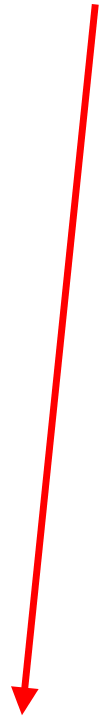
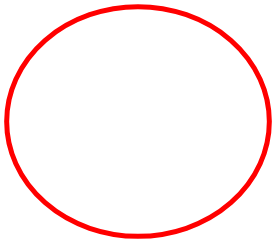
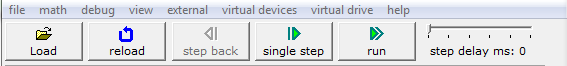
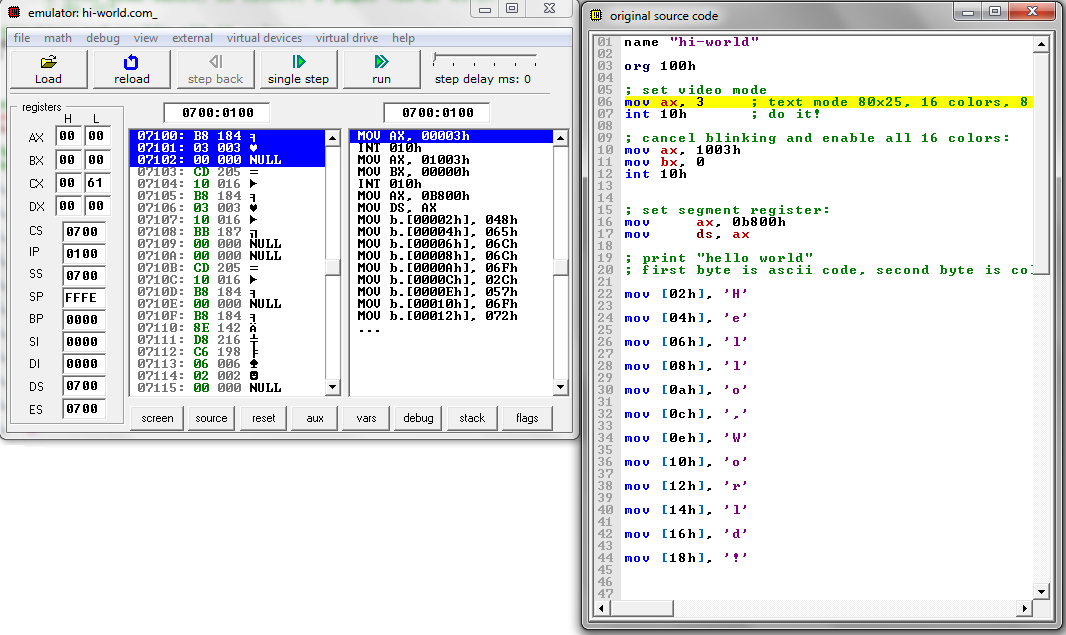


1. When you have successfully assembled the code and you need to click on '**Close**' to close the assembler status dialog box.

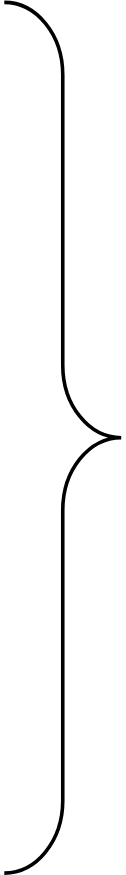
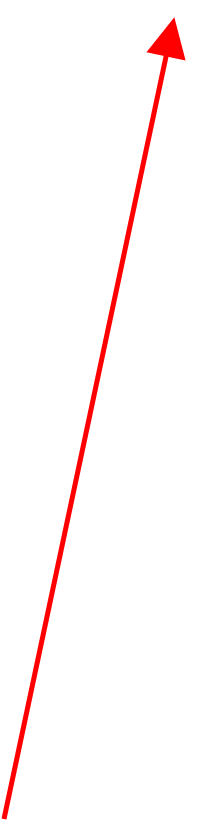
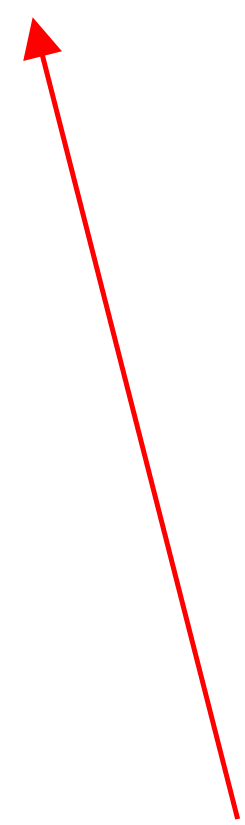
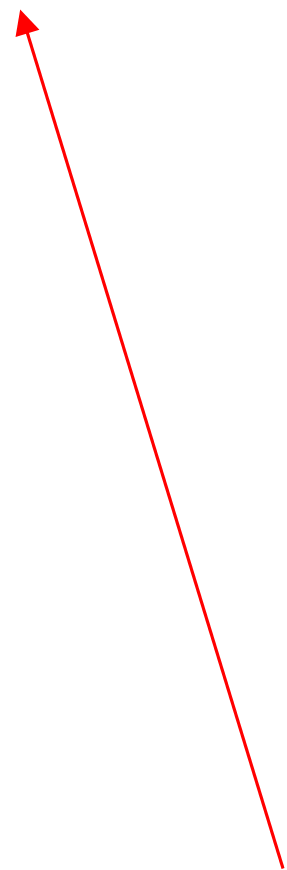
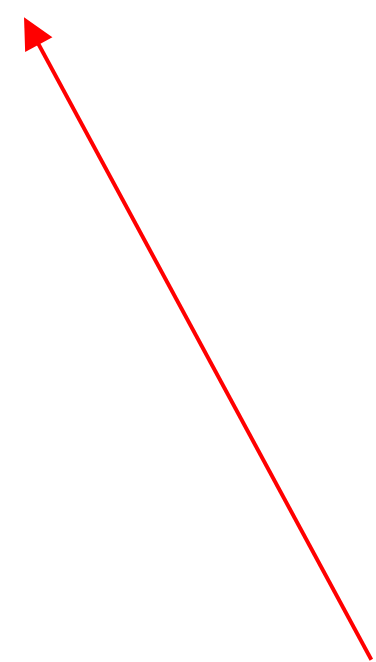
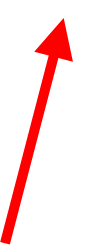
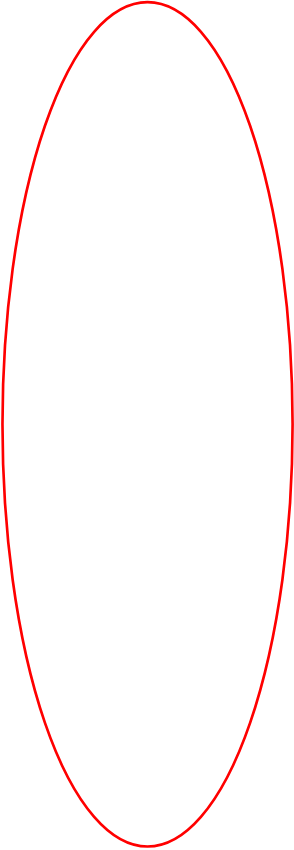
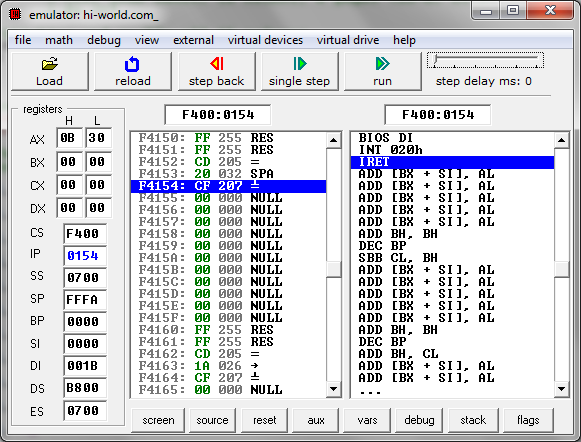


error message/report will be in this window

1. Then you need to click on '**emulate**' tab to invoke the assembler.
2. Then click on '**run**' tab to invoke the assembler running the code.



1. Then click on '**run**' tab to invoke the assembler running the code.



Disassembled machine code

Observe the register content each single step

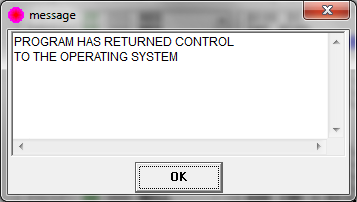
The line of the code that being execute

**Physical HEX DECIMAL ASCII**

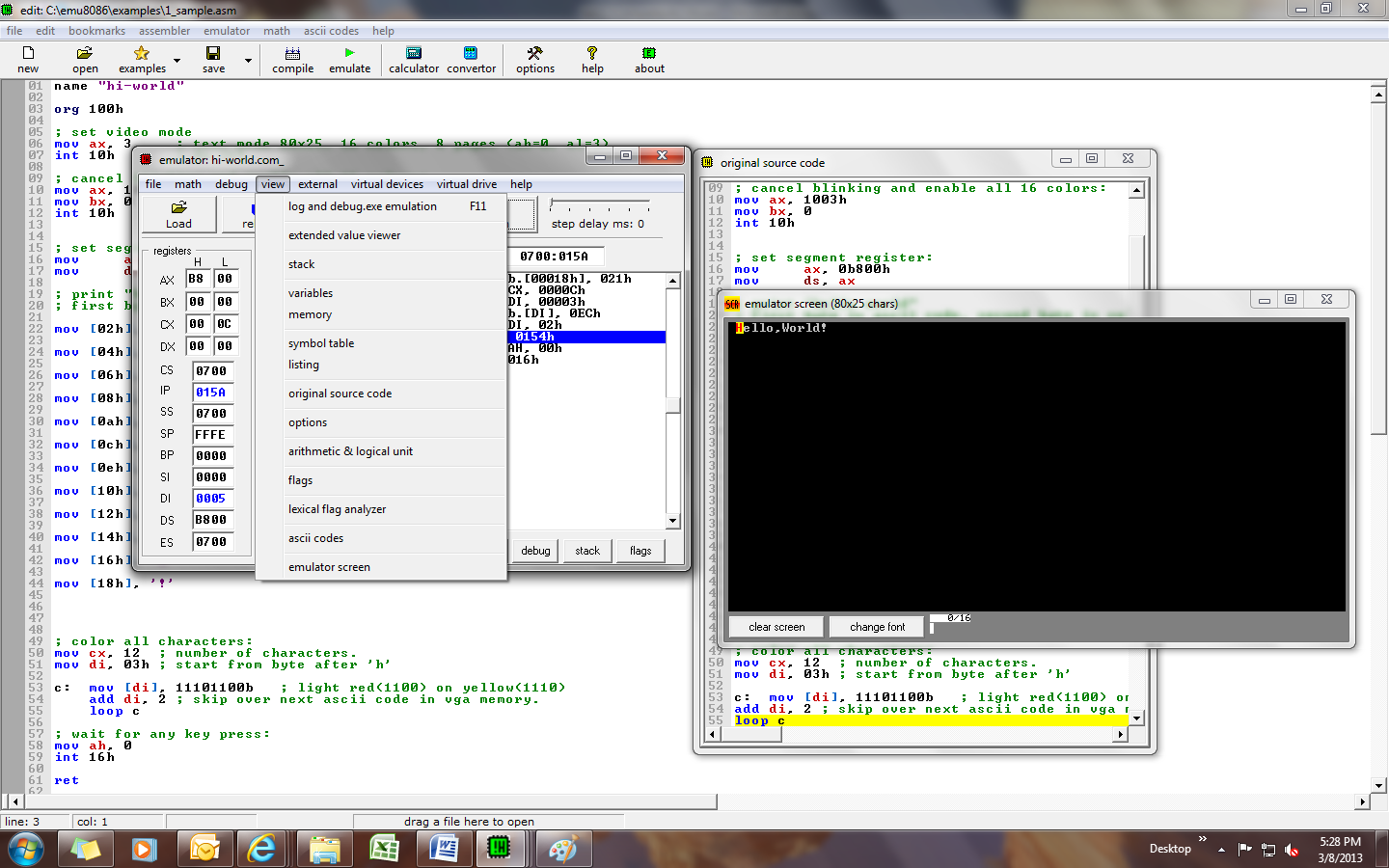
**Address :**

**The Memory List**

1. Below dialog box will pop out, once the assembler finish running the code.



1. Reload back the program. Next, click '**single step**' tab to observe the changes of the register content.
2. If you click on the ‘**view**’ tab, below pull-down list will appear and you can see several i/o window.

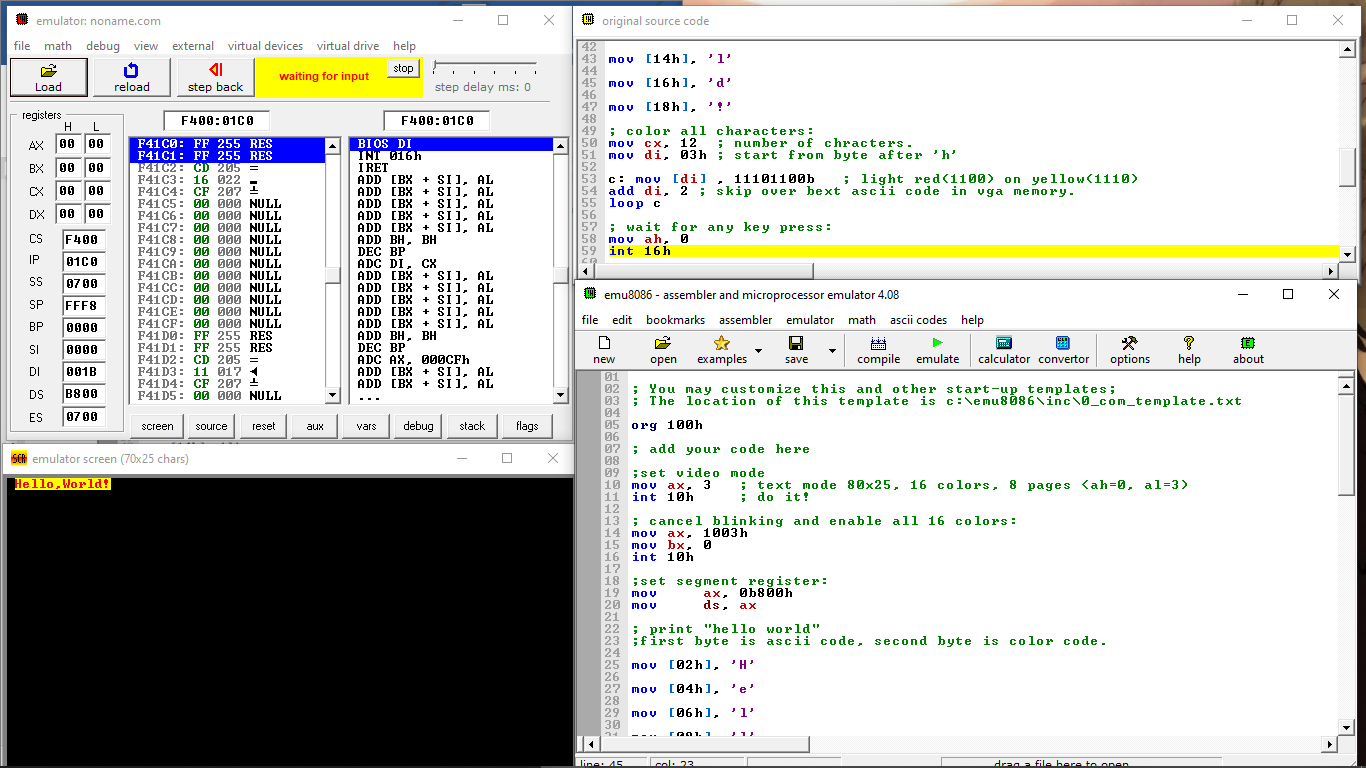


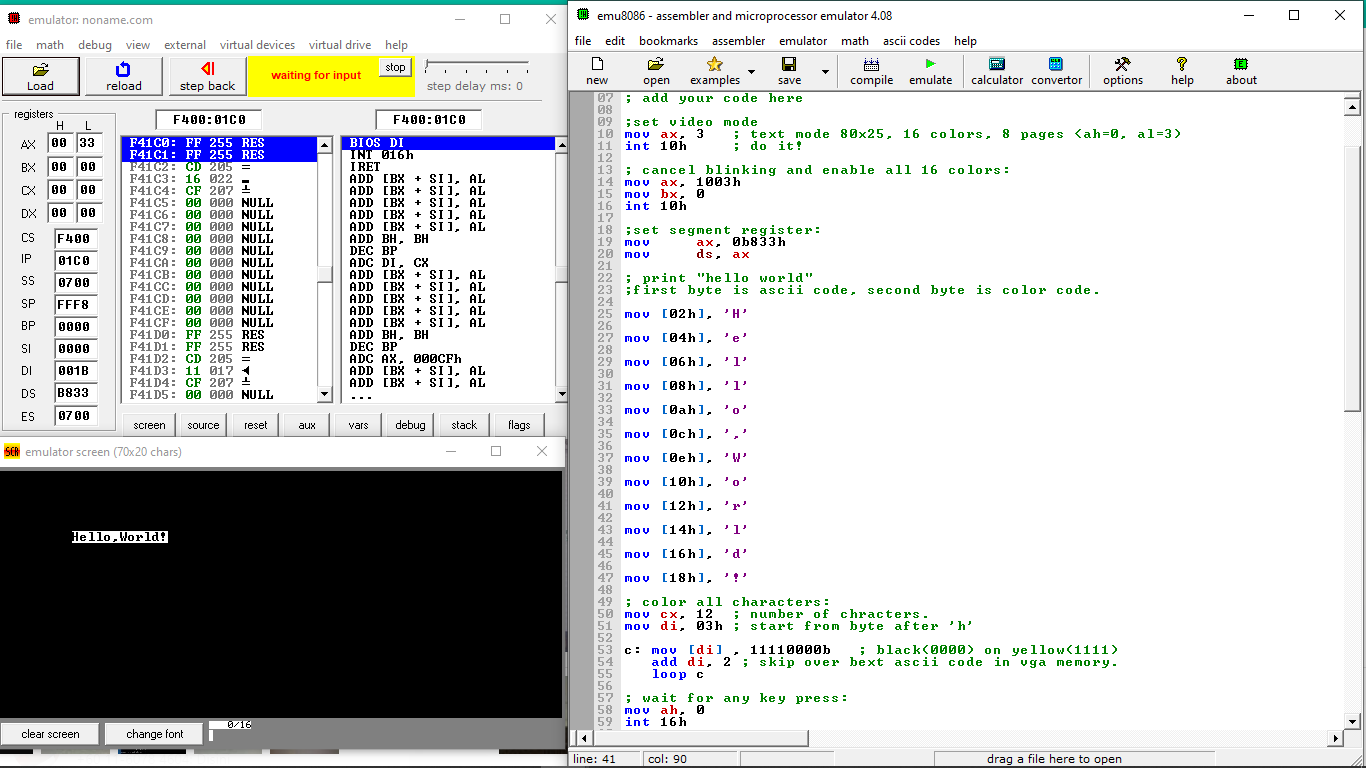
1. Observe the contents of the specified registers below **after** each instruction has been executed and fill in the following table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **AX** | | **CS** | **CX** | **IP** | **SP** | **DI** | **DS** |
|  | **H** | **L** |  |  |  |  |  |  |
| mov ax, 1003h | 10 | 03 | 0700 | 0061 | 0108 | FFFE | 0000 | 0700 |
| mov bx, 0 |  |  |  |  |  |  |  |  |
| int 10h |  |  |  |  | 0100 |  |  |  |
| mov ax, 0b800h | B8 | 00 |  |  | 0110 |  |  |  |
| mov ds, ax |  |  |  |  | 0112 |  |  | B800 |
| mov [02h], 'H' |  |  |  |  | 0117 |  |  |  |
| mov [04h], 'e' |  |  |  |  | 011C |  |  |  |
| mov [06h], 'l' |  |  |  |  | 0121 |  |  |  |
| mov [08h], 'l' |  |  |  |  | 0126 |  |  |  |
| mov [0ah], 'o’ |  |  |  |  | 012B |  |  |  |
| mov [0ch], ',' |  |  |  |  | 0130 |  |  |  |
| mov [0eh], 'W' |  |  |  |  | 0135 |  |  |  |
| mov [10h], 'o' |  |  |  |  | 013A |  |  |  |
| mov [12h], 'r' |  |  |  |  | 013F |  |  |  |
| mov [14h], 'l' |  |  |  |  | 0144 |  |  |  |
| mov [16h], 'd' |  |  |  |  | 0149 |  |  |  |
| mov [18h], '!' |  |  |  |  | 014E |  |  |  |
| mov cx, 12 |  |  |  | 000C | 0151 |  |  |  |
| mov di, 03h |  |  |  |  | 0154 |  | 0003 |  |
| c: mov [di], 11101100b |  |  |  |  | 0157 |  |  |  |
| add di, 2 |  |  |  |  | 015A |  | 0005 |  |
| loop c |  |  |  | 000B | 0154 |  |  |  |
| mov ah, 0 | 00 | 00 |  |  | 015E |  | 001B |  |
| int 16h |  |  | F400 | 0000 | 01C0 | FFF8 |  |  |
| ret |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

1. Now, edit the program above to print out different text (anything with more than 15 char) with different colours and printed at the third line.
2. Have fun ☺.

## EXPERIMENT RESULTS





## DISCUSSION

* We learn how to use edit, assemble and simulate programs using the emu8086 suite.
* Emu8086 cross-assembler and simulator allows us to write a 8086 assembly language program.
* Simulator allows us to execute a single instruction at a time and to observe the state of simulated registers after the execution of each instruction.
* Assembly level programming is very important to low-level embedded system design is used to access the processor instructions to manipulate hardware.

## CONCLUSION

In conclusion, by using the emu8086 software, we could write an 8086 assembly language programming to execute basic devices. Besides, it helps us to become familiar in using simulation software for 8086. We also can implement a simple application based on the simulation.