Worksheet 2

Objectives: This worksheet provides introduction about Assembly and how to run assembly code using 8086 microprocessor.

Submission: Write the answers for all activities into a Microsoft Word document and save using your IT Number. Upload the document into the link provided in the courseweb before the deadline.

EMU8086 software is an emulator designed to emulate the 8086 microprocessor. This software is used to write assembly programs, compile and run on an emulated 8086 microprocessor. Refer the prereading material available in the courseweb and complete this practical sheet.

Setting up

1. Start emu8086
2. Click open button and select the "adding.asm" file to open-source code. (Download it from courseweb first).
3. Read the script. The comments (shown in green), these explain what each line of code does.
4. Run the program using the "emulate" button. The script will be compiled and then run on the emulator. The program adds number 1 (which is given as 5) and number 2 (which is given as 10). The addition of the numbers (15 in this case) is output in the command window in binary.
5. Reload the program by clicking on the "reload" button.
6. Click on the "flags" button at the bottom right of the emulator window to display the flag bits.
7. Go through the program line by line using "single step" button and observe what each line of code does by observing the registers (shown on the left side in the emulator window) and the flag register (shown on the right side of the emulator window).

Activity

1. Write down the values of the registers **BX**, **DX**, and **ZF** just after running the line *zero: int 21h* at the given counter value (**CX** value).
   Observe how the output (**DX**) changes, the value printed onto the display is the value stored in **DX**. **ZF** is used to determine the value that should be printed, observe how this change with each **CX**.

   Hint :-
   - Run the "adding.asm" source code.
   - Check the CX, BX, DX and ZF values in the beginning on the emulator.
   - Click "Single step" button on the emulator until you get CX value as 00 06.
   - Check the original source code, once you get CX value as 00 06.

- Now you are in line *print: move ah, 2*.
- Move line *zero: int 21h* by clicking Single step button again.
- Check the BX, DX, and ZF (Click on flag button) values and write down.
- Check other values by changing the CX value.

CX	BX	DX	ZF
00 06	**00 3C**	**00 30**	**1**
00 03			
00 02			

2. Observe the registers **CS** and **IP**. **CS** register contains the segment number of the next instruction and the **IP** contains the offset. These two registers in conjunction tells the processor which instruction should be executed next. Write down the values of **CS** and **IP** just after running each of the given *instructions* at the given **CX** value.

The *actual address* of the instruction can be obtained by adding **IP** to **CS**. Fill in the *actual address*. The *actual address* is where the instruction is actually located in the memory.

Hint :-

- Run the "adding.asm" source code.
- Check the CX, IP, and CS values in the beginning on the emulator.
- Click "Single step" button on the emulator until you get CX value as 00 23.
- Check the original source code, once you get CX value as 00 23.
- Now you are in line *mov al, 5*.
- Move line *mov al, 10* by clicking Single step button again.
- Check the IP, and CS values and write down.
- Check other values by changing the CX value and instructions.

Instruction	CX	IP	CS	Actual address
mov bl, 10	00 23	**0102**	**0700**	**07102**
add bl, al	00 23			
jz zero	00 07			
zero: int 21h	00 07			

Worksheet 2

Print Characters on the command prompt Screen

Setting up

1. Start emu8086.
2. Open the "Hey.asm" file.
3. Read the script. The comments (shown in green), these explain what each line of code does. The program prints the given characters one by one.
4. Run the program using the "emulate" button. The script will be compiled and then emulator will start.
5. "Run" button can be used to run the whole script. "Single step" button can be used to execute the code line by line. "Step back" button can be used to go back to the last executed line. "Reload" button can be used to reload the program.
6. Go through the program line by line using "single step" button and observe what each line of code does by observing the registers (shown on the left side in the emulator window).

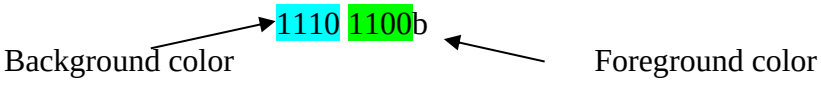
Activity

1. The given program prints "Hey" onto the command prompt character by character. Edit the code to print "Hello World!" word. Only **even number addresses** can be used to hold Characters.

   (**NOTE:** *make sure the is an empty byte between the character addresses, as the byte following the character address holds the character attributes, which are the color for the background and foreground.*)

   E.g.,
   mov [00h], 'H' – Even number address
   mov [02h], 'e'
   mov [04h], 'l'
   …
   mov [14h], 'd'
   mov [16h], '!'

2. Character attributes can be added to change the color of the foreground (character color) and background color. The byte following the character byte is used to set the character attributes. **Odd number addresses** can be used to hold character colors. The high 4 bits of the character attribute byte is used to set the background color and the low 4 bits are used to set the foreground colors. Add the character attribute byte as 11101100b to set the background to yellow and foreground to light red. (**NOTE:** *the color codes are given at the top of the code.*)



E.g.,

```
mov [00h], 'H'
mov [01h], 11101100b  - Odd number address

mov [02h], 'e'
mov [03h], 11101100b

…

mov [14h], 'd'
mov [15h], 11101100b

mov [16h], '!'
mov [17h], 11101100b
```

Try changing the character attribute byte to set different foreground and background colors.

3. The positions of characters can be simply changed by changing the memory address of the character (the number within the square brackets [ ]). Change the memory addresses to position the "Hello World!" at the **middle** of the screen. (**NOTE:** *useable memory address starts from [00h] and ends at [9Eh].*)

Hint:-

- Convert 00 and 9E hexadecimal values to decimal values.
- To find middle of the screen memory address, add decimal values of the start and end values and divide into 2.
- [(start decimal value + end decimal value) / 2]
- If you got odd number as the result, get the next even number as starting middle memory address.
- Convert decimal value to hexadecimal.
- Hold character by character starting from middle memory address and print "Hello World!" at the middle of the screen.