Lab # 02
Introduction to Assembly Language Tools

Introduction to Assembly Language Tools

Software tools are used for editing, assembling, linking, and debugging assembly language programming. You will need an assembler, a linker, a debugger, and an editor. These tools are briefly explained below.

Assembler

An **assembler** is a program that converts **source-code** programs written in **assembly language** into **object** files in machine language. Popular assemblers have emerged over the years for the Intel family of processors. These include MASM (Macro Assembler from Microsoft), TASM (Turbo Assembler from Borland), NASM (Netwide Assembler for both Windows and Linux), and GNU assembler distributed by the free software foundation.

Linker

A **linker** is a program that combines your program's **object file** created by the assembler with other object files and **link libraries**, and produces a single **executable program**.

Debugger

A **debugger** is a program that allows you to trace the execution of a program and examine the content of registers and memory.

Editor

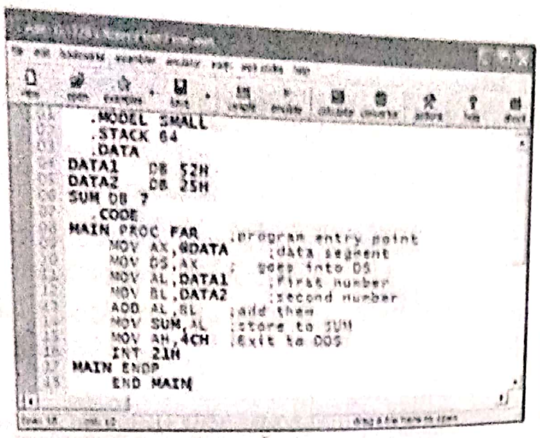
You need a text editor to create assembly language source files. You can use NotePad , or any other editor that produces plain ASCII text files.

Emu8086

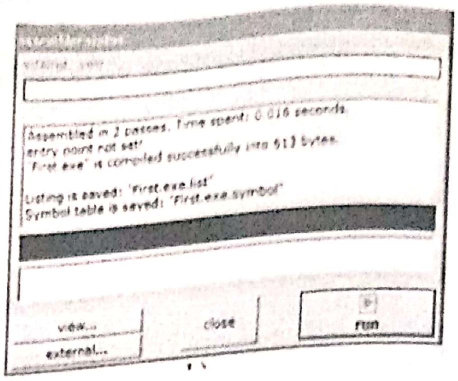
Emu8086 combines an advanced source editor, assembler, disassemble and software emulator (Virtual PC) with debugger. It compiles the source code and executes it on emulator step by step.
Visual interface is very easy to work with. You can watch registers, flags and memory while your program executes.
Arithmetic & Logical Unit (ALU) shows the internal work of the central processor unit (CPU).
Emulator runs programs on a Virtual PC, this completely blocks your program from accessing real hardware, such as hard-drives and memory, since your assembly code runs on a virtual machine, this makes debugging much easier. 8086 machine code is fully compatible with all next generations of Intel's microprocessors, including Pentium II and Pentium 4. This makes 8086 code very portable, since it runs both on ancient and on the modern computer systems. Another advantage of 8086 instruction set is that it is much smaller, and thus easier to learn.

EMU8086 Source Editor

The source editor of EMU86 is a special purpose editor which identifies the 8086 mnemonics, hexadecimal numbers and labels by different colors as seen in Figure 1.



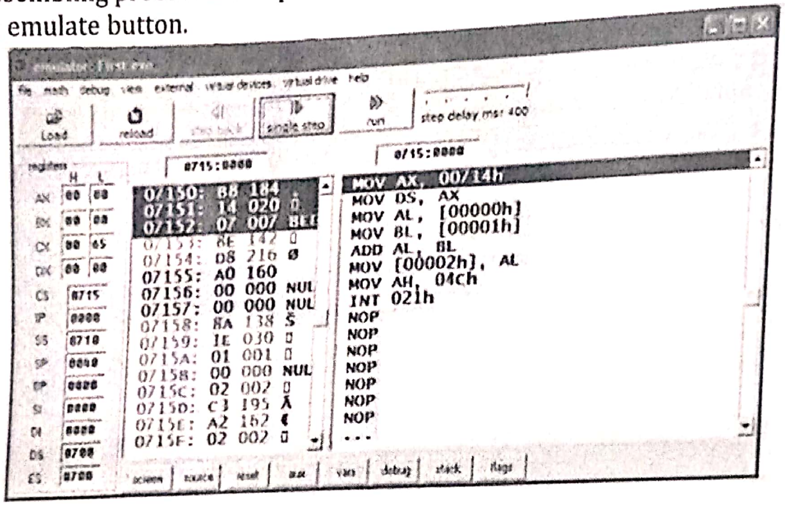
```
.MODEL SMALL
.STACK 64
.DATA
DATA1   DB 52H
DATA2   DB 25H
SUM DB ?
 .CODE
MAIN PROC FAR       ;program entry point
    MOV AX,@DATA        ;data segment
    MOV DS,AX       ;  goes into DS
    MOV AL,DATA1        ;First number
    MOV BL,DATA2        ;second number
    ADD AL,BL      ;add them
    MOV SUM,AL     ;store to SUM
    MOV AH,4CH     ;Exit to DOS
    INT 21H
MAIN ENDP
    END MAIN
```



```
Assembled in 2 passes. Time spent: 0.016 seconds
entry point not set!
 "First.exe" is compiled successfully into 513 bytes.

Listing is saved: "First.exe.list"
Symbol table is saved: "First.exe.symbol"
```

Figure:1

The compile button on the taskbar starts assembling and linking of the source file. A report window is opened after the assembling process is completed. Figure 2 shows the emulator of 8086 which gets opened by clicking on emulate button.



```
     H  L
AX  00 00
BX  00 00
CX  00 65
DX  00 00
CS  0715
IP  0000
SS  0710
SP  0040
BP  0000
SI  0000
DI  0000
DS  0700
ES  0700
```

```
0715:0000
07150: B8 184 ©
07151: 14 020 ¶
07152: 07 007 BEL
07153: 8E 142 Ä
07154: D8 216 Ø
07155: A0 160
07156: 00 000 NUL
07157: 00 000 NUL
07158: 8A 138 Š
07159: 1E 030 ▲
0715A: 01 001 ☺
0715B: 00 000 NUL
0715C: 02 002 ☻
0715D: C3 195 Ã
0715E: A2 162 ¢
0715F: 02 002 ☻
```

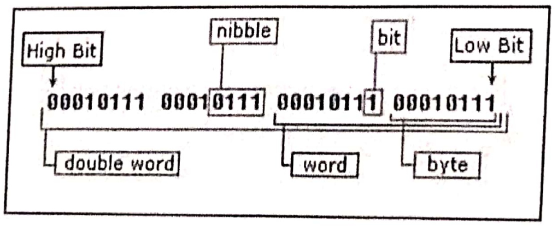
```
0715:0000
MOV AX, 00714h
MOV DS, AX
MOV AL, [00000h]
MOV BL, [00001h]
ADD AL, BL
MOV [00002h], AL
MOV AH, 04Ch
INT 021h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Figure 2

Data Representation in EMU8086

Binary System

Computers are not as smart as humans are (or not yet), it's easy to make an electronic machine with two states: **on** and **off**, or **1** and **0**. Computers use binary system, binary system uses 2 digits: **0, 1**And thus the base is **2**. Each digit in a binary number is called a **BIT**, 4 bits form a **NIBBLE**, 8 bits form a **BYTE**, two bytes form **WORD**, two words form a **DOUBLE WORD** (rarely used)



In emu8086, There is a convention to add "b" in the end of a binary number, this way we can determine that 101b is a binary number with decimal value of 5.

4 2 1
1 0 1 b

Decimal System

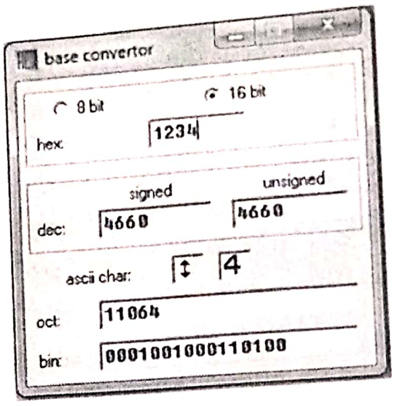
Most people today use decimal representation to count. In the decimal system there are 10 digits:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9. These digits can represent any value, for example: **754**.

Hexadecimal System

Hexadecimal System uses 16 digits:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F and thus the base is 16. Hexadecimal numbers are compact and easy to read. There is a convention to add "h" in the end of a hexadecimal number.

Base Converter

Emulator 8086 provides base converter facility which can convert any number (signed/unsigned) into octal, hexadecimal, decimal and binary number systems.



hex: 1234
dec: signed 4660, unsigned 4660
ascii char: 4
oct: 11064
bin: 0001001000110100

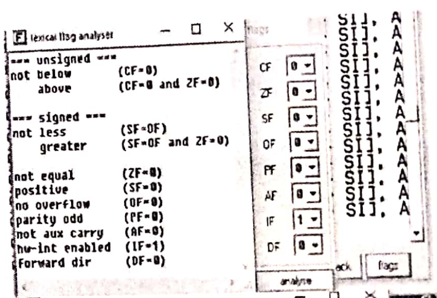
Example: Use Emu8086 to make the following calculations:

10100101b = ? d
1234h = ? d
39d = ? h

1. Start *Emu8086* by selecting its icon from the start menu, or by running Emu8086.exe
3. Choose "Math" and specify "Base Convertor" in emu8086.
4. Enter one of the numbers like in the Figure .

Emulator 8086 Environment

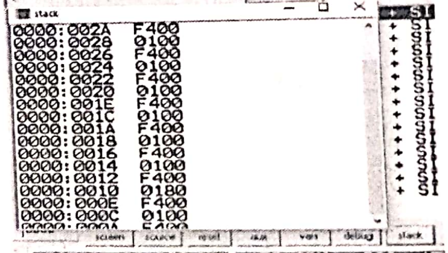
- **Flags** : By pressing "flags" in the bottom-right corner of the window(figure 3)you can observe flag values at the execution of instruction written in a program ,



```
=== unsigned ===
not below        (CF=0)
    above        (CF=0 and ZF=0)

=== signed ===
not less         (SF=OF)
    greater      (SF=OF and ZF=0)

not equal        (ZF=0)
positive         (SF=0)
no overflow      (OF=0)
parity odd       (PF=0)
not aux carry    (AF=0)
hw-int enabled   (IF=1)
Forward dir      (DF=0)
```

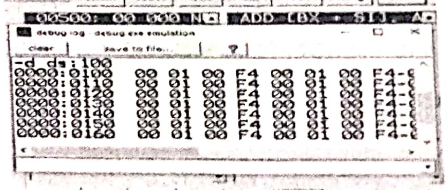
CF 0, ZF 0, SF 0, OF 0, PF 0, AF 0, IF 1, DF 0

- **Stack:** stack segment window is popup by pressing "stack". Stack is directly affected when call and return instructions used in the program.



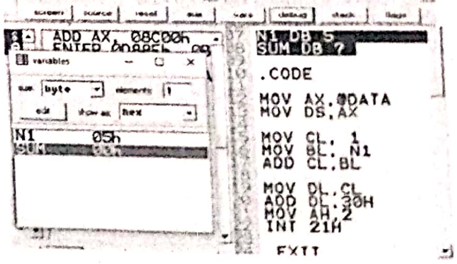
```
0000:002A   F400
0000:0028   0100
0000:0026   F400
0000:0024   0100
0000:0022   F400
0000:0020   0100
0000:001E   F400
0000:001C   0100
0000:001A   F400
0000:0018   0100
0000:0016   F400
0000:0014   0100
0000:0012   F400
0000:0010   0180
0000:000E   F400
0000:000C   0100
```

- **Debug**: emulator86 also provide debug environment where you can view and also edit the contents of memory segments , register values through debug commands.



```
-d ds:100
0000:0100   00 01 00 F4 00 01 00 F4-
0000:0110   00 01 00 F4 00 01 00 F4-
0000:0120   00 01 00 F4 00 01 00 F4-
0000:0130   00 01 00 F4 00 01 00 F4-
0000:0140   00 01 00 F4 00 01 00 F4-
0000:0150   00 01 00 F4 00 01 00 F4-
0000:0160   00 01 00 F4 00 01 00 F4-
```

- **Variables:** Variable/constants declare in the program can also be viewed through "vars" .

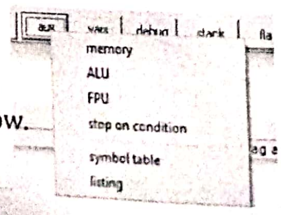


size: byte, elements: 1, show as: hex

```
N1        05h
SUM       07h
```

```
ADD AX, 08C00h
ENTER 0088Fh, 00

N1 DB 5
SUM DB ?

.CODE

MOV AX,@DATA
MOV DS,AX

MOV CL, 1
MOV BL, N1
ADD CL,BL

MOV DL,CL
ADD DL,30H
MOV AH,2
INT 21H

EXIT
```

- **Aux:** you can view various options(listing ,memory, ALU and FPU) under this tab as mention in the figure below.



EXERCISE:

1. Name four software tools used for assembly language programming
   - Assembler
   - Linker
   - Text Editor
   - Debugger

2. Convert the following numbers using Base Converter: Do all the calculations manually and compare with the results obtained using base converter.

a) 10110011b (binary to hexadecimal)

Solution:

Answer using base converter : (B3)h

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

Pairs of four for binary to hexadecimal conversion:

0011 = 3

1011 = 11 = B

Answer: (B3)h

b) 455d (decimal to binary)

Solution:

Answer using base converter : (111000111)b

2	455	
2	227	1
2	113	1
2	56	1
2	28	0
2	14	0
2	7	0
2	3	1
2	1	1

Answer: (111000111)b

c) 2AFh (hexadecimal to binary)

Answer using base converter : (1010101111)b

2 = 0010

A = 10 = 1010

F = 15 = 1111

Answer: (001010101111)b

d) 0A1h (hexadecimal to binary)

Answer using base converter : (10100001)b

0 = 0000

A = 1010

1 = 0001

Answer: (000010100001)b

3. Write each of the following 8-bit Signed Binary Integers in Decimal:

a) 11011100 = 220

b) 10010001 = 145

c) 10001111 = 143

d) 10000000 = 128

27/11/25