Al-Azhar UNIVERSITY

Faculty of Engineering

Computers and Systems Engineering Department

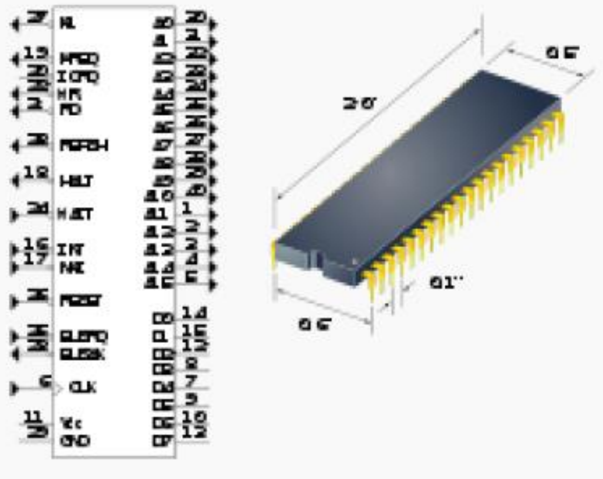
Z80 Processor

Objectives of the Experiment:-

- Describe Z-80 structure.
- Explain how Z-80 run programs.
- Explain simple examples to show Z-80 registers and memory pages.

INTRODUCTION

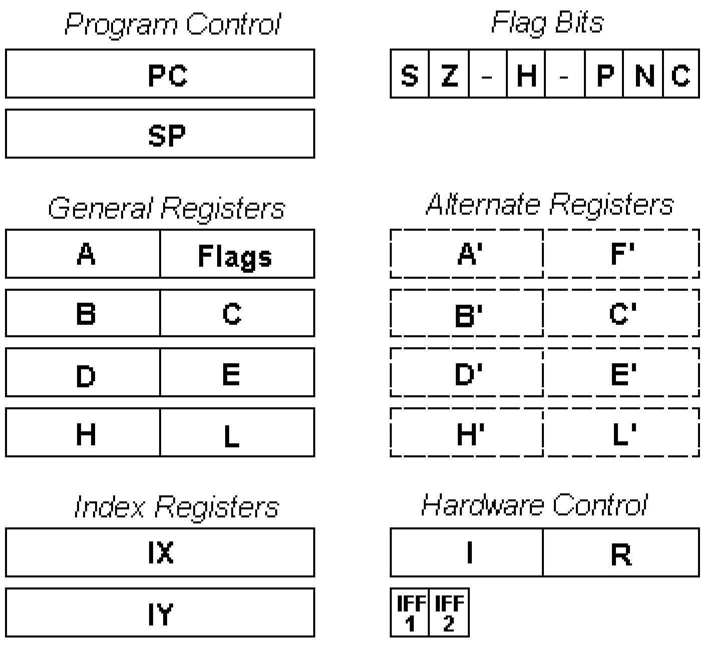
The **Z80** (US: *"zee-eighty"*, UK: *"zed-eighty"*) is an 8-bit microprocessor designed by Zilog and sold from July 1976 onwards. It was widely used both in desktop and embedded computer designs as well as for military purposes. The Z80 and its derivatives and clones made up one of the most commonly used CPU families of all time, and, along with the MOS Technology 6502 family, dominated the 8-bit microcomputer market from the late 1970s to the mid-1980s.



The Z80's original DIL40 chip package pinout

Z80 Processor Registers

All programming of the Z80 revolves around its internal registers. These registers are used to control the flow of the program and to operate on data items. There are 16-bit registers inside the processor that address the program bytes in memory and there are 8-bit registers into which data bytes are loaded and then operated on. These 8-bit registers can also be combined in pairs to form 16-bit addresses and these addresses can be used to index data structures in memory that span several bytes.



Z80 Registers

Instruction Set

The closer we get to the core of a computer, the more unfamiliar things may appear. In fact, we are getting closer and closer to the atoms of the computer: the 1s and 0s! This is the language of the machine. But if it's 1s and 0s, how do we actually communicate with the processor? Systems programmers use **assembly language**, which is a type of programming language.

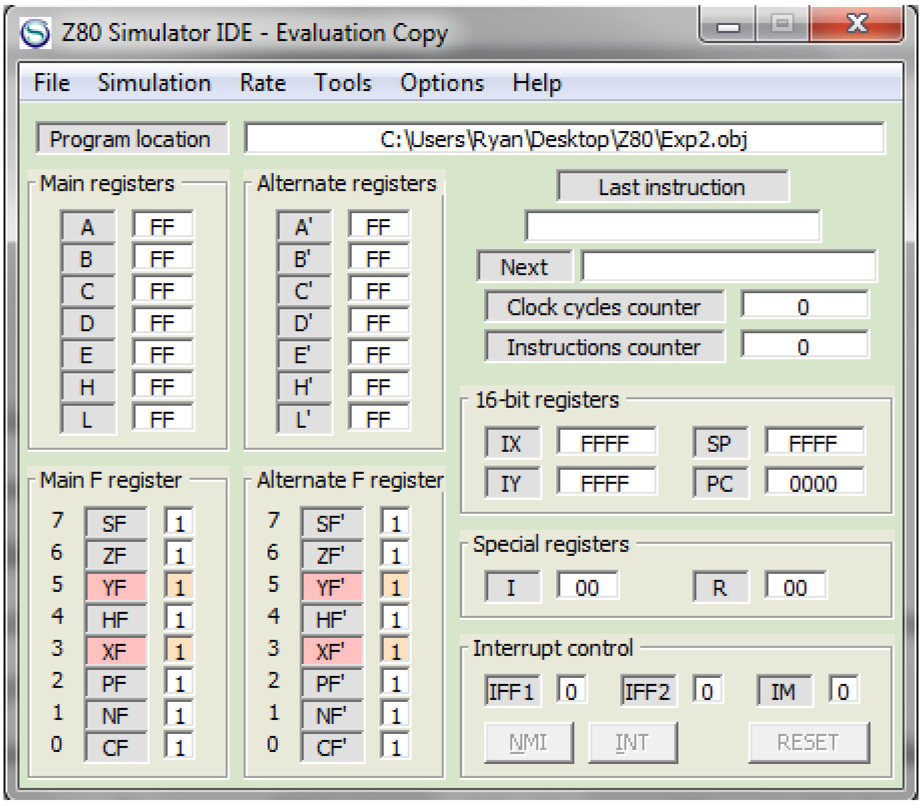
An **instruction set** (used in what is called **ISA**, or Instruction Set Architecture) is code that the computer processor (CPU) can understand. It contains instructions or tasks that control the movement of bits and bytes within the processor.

EXAMPLE 1
Create a text file with name memfill and change the extension from .txt to .asm and write down the bellow assembly code. This assembler routine fills the memory range FF00H - FFFFH with values FFH - 00H respectively.

```
        LD A,0FFH ;initial value in register A
        LD BC,0FF00H ;initial value in register pair BC
L1:     LD (BC),A ;load value in A to the memory location addressed by BC
        INC BC ;increment BC
        DEC A ;decrement A
        JP NZ,L1  ;loop until value in A is zero
        LD (BC),A ;load value 00H to memory location FFFFH
        HALT ;halt cpu
        .END
```

Program Execution

- Start Z80 Simulator IDE.


Z80 Simulator IDE - Evaluation Copy

File Simulation Rate Tools Options Help

Program location: C:\Users\Ryan\Desktop\Z80\Exp2.obj

Main registers

A	FF
B	FF
C	FF
D	FF
E	FF
H	FF
L	FF

Alternate registers

A'	FF
B'	FF
C'	FF
D'	FF
E'	FF
H'	FF
L'	FF

Last instruction

Next

Clock cycles counter: 0

Instructions counter: 0

16-bit registers

IX	FFFF	SP	FFFF
IY	FFFF	PC	0000

Special registers

I	00	R	00

Main F register

7	SF	1
6	ZF	1
5	YF	1
4	HF	1
3	XF	1
2	PF	1
1	NF	1
0	CF	1

Alternate F register

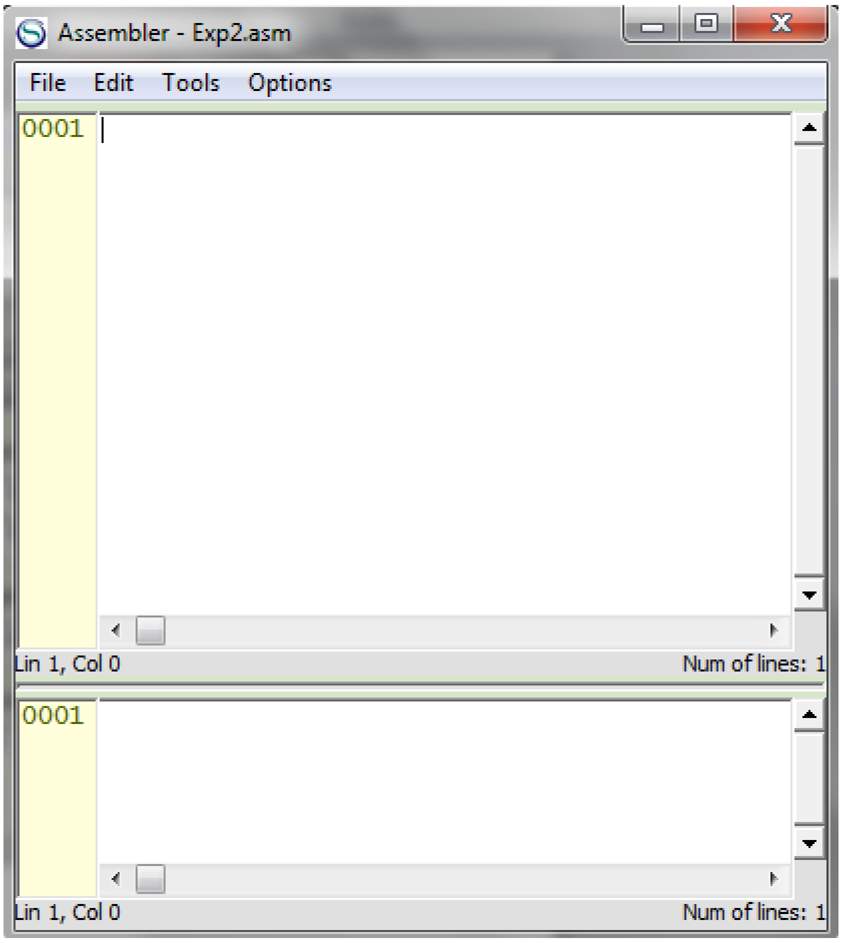
7	SF'	1
6	ZF'	1
5	YF'	1
4	HF'	1
3	XF'	1
2	PF'	1
1	NF'	1
0	CF'	1

Interrupt control

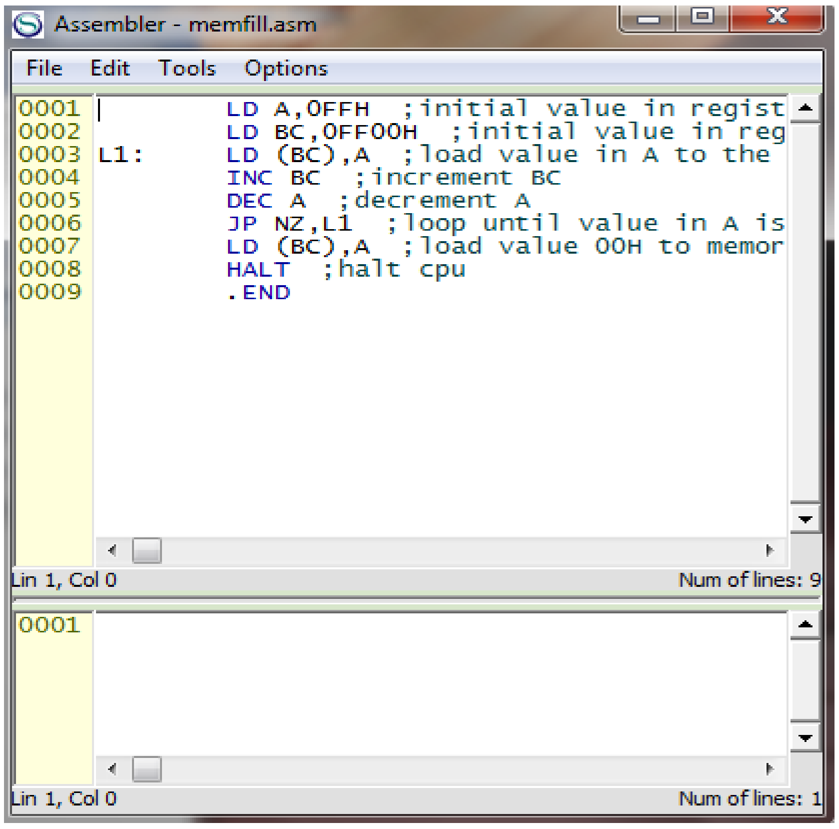
IFF1 0 IFF2 0 IM 0

NMI INT RESET

Click on Tools\Assembler.

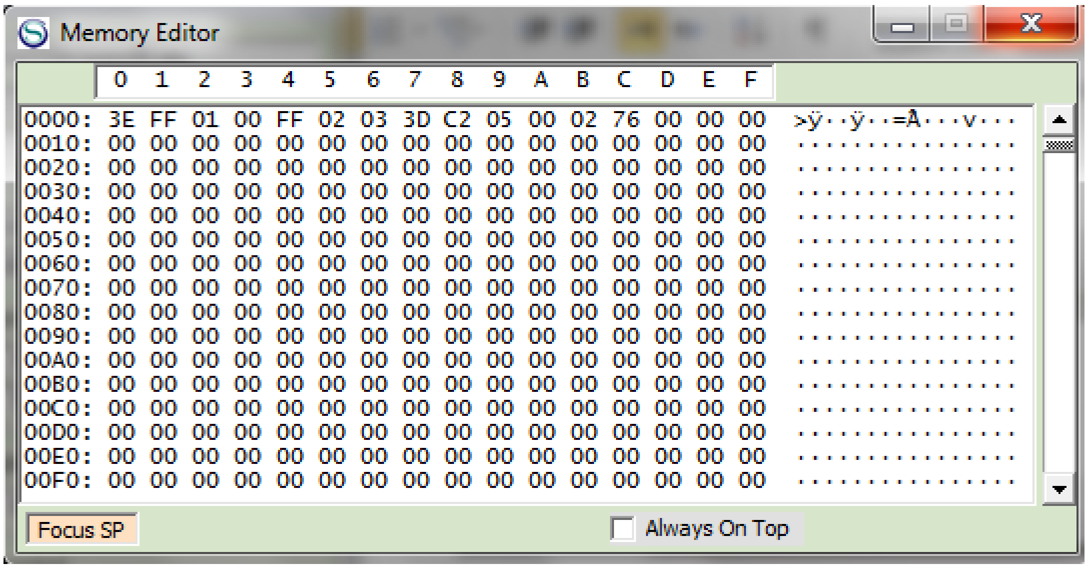


- Click on File\Open and select memfill.asm file and click on Open. The assembler source program will be displayed in the editor.



```
        LD A,0FFH   ;initial value in regist
        LD BC,0FF00H   ;initial value in reg
L1:     LD (BC),A   ;load value in A to the
        INC BC   ;increment BC
        DEC A   ;decrement A
        JP NZ,L1   ;loop until value in A is
        LD (BC),A   ;load value 00H to memor
        HALT   ;halt cpu
        .END
```

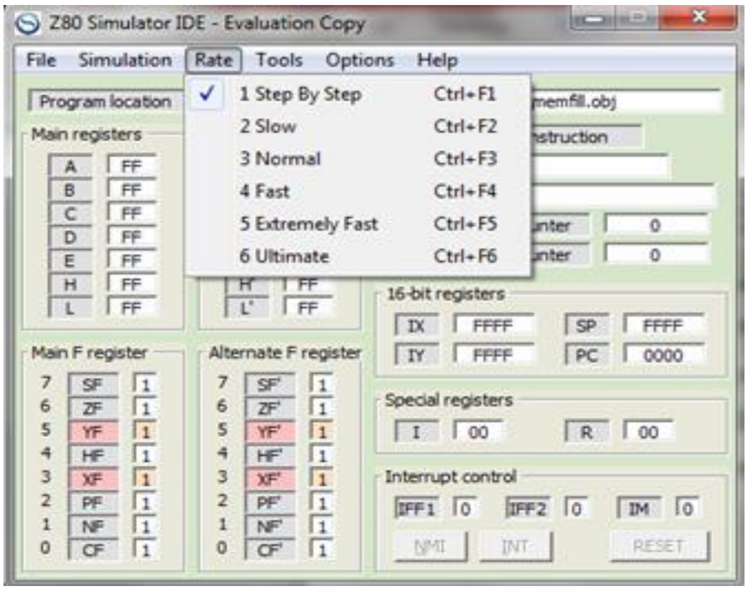
- Click on Tools\Assemble. After the operation is completed the assembler will generate two files: memfill.lst (assembler listing with assembled opcodes) and memfill.obj (binary image of assembled routine ready to be loaded into memory). The output listing file memfill.lst will be displayed.
- Click on Tools\ Assemble &Load. That will load the program file memfill.obj into Z80 Simulator IDE memory.
- Close assembler window.
- Select the option Options\Enable logging.
- Select the option Options\Refresh Memory Editor.
- Click on Tools\Memory Editor. That will open the Memory Editor window. 13 bytes of the program are in focus.



```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000: 3E FF 01 00 FF 02 03 3D C2 05 00 02 76 00 00 00   >ÿ··ÿ··=Â···v···
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
0040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
0050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
00C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
00D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
00E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
00F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ················
```

- Reposition the windows on the screen to get better view, if necessary use Always on Top option on Memory Editor window.
- Using the scroll bar select the memory range FF00-FFFF.

- You can choose the rate of execution by choosing the desired rate under Rate Tab.



EXAMPLE 2

Examine memfill2.asm file from the application folder. This is modified example 1. The value that will be used to fill the memory range FF00H - FFFFH is get from I/O port 01H. The routine is then repeated.

```
L2:     IN A,(01H) ; get value on port 01H to be used for memory fill
        LD D,0FFH ;initial value  in counter register D
        LD BC,0FF00H ;initial value in pointer register pair BC
L1:     LD (BC),A ;load value in A to the memory location addressed by BC
        INC BC ;increment pointer BC
        DEC D ;decrement
        counter                    D
        JP NZ,L1 ;loop until value in D is zero
        LD (BC),A ;fill the last    memory location FFFFH
        JP L2 ;repeat routine
        .END
```

POST-LAB

Write a program that multiplies two numbers taken from user and save result into memory address?