Capital University of Science and Technology

Department of Computer Science

CS2523 – Computer Organization and Assembly Language

ASSIGNMENT NO. 6: Effective Address Calculation, Jumps/Branching, Relative Addressing, Bit Manipulation

CLO: 1. Define concepts in the design of microprocessor as state machine and designing its data path and its controller. [C1- Remembering]

CLO: 3. Implement assembly programs of intermediate complexity using the intel 8088 architecture. The student should also be able to convert intermediate complexity program in high level language into assembly code. [C3- Applying]

Semester: Summer 22 **Max Marks:** 10

Instructor: Ms. Tayyaba Zaheer

Assigned Date: September 14, 2022 **Due Date:** September 17, 2022

Name: **Reg. No.**

Guidelines:

You are required to submit the **screenshots of code and output of the program (where required) and concepts in your own words i.e. must be hand written** in the assignment file (word or pdf – pictures attached must be readable and in portrait mode) as **courseCode_studentReg#_studenName** via Microsoft Teams.

Important Note:

1) Must not copy from other students, so do it all yourself.
2) Assignment should be hand written.

Description:

Emu8086 is an 8086-microprocessor emulator and disassembler. Emu8086 permits to assemble, emulate and debug 8086 programs (16bit/DOS).

Tasks: [Hint: you can take help from lectures]

Task#1: Effective Address Calculation: **(03 marks)**

Question: What is the effective address generated by each of the following instruction?

Initially BX=0x0100, label=0x0234, [label]=0x0010, and SI=0x00E1

(Offset in part a is in decimal)

a) mov ax, [bx+40]

Solution:

Effective Address = bx + 40

= 0100 + 0028

= 0128

b) mov ax, [bx+label]

Solution:

Effective Address = bx + label

= 0100 + 0234

= 0334

c) mov ax, [bx+si]

Solution:

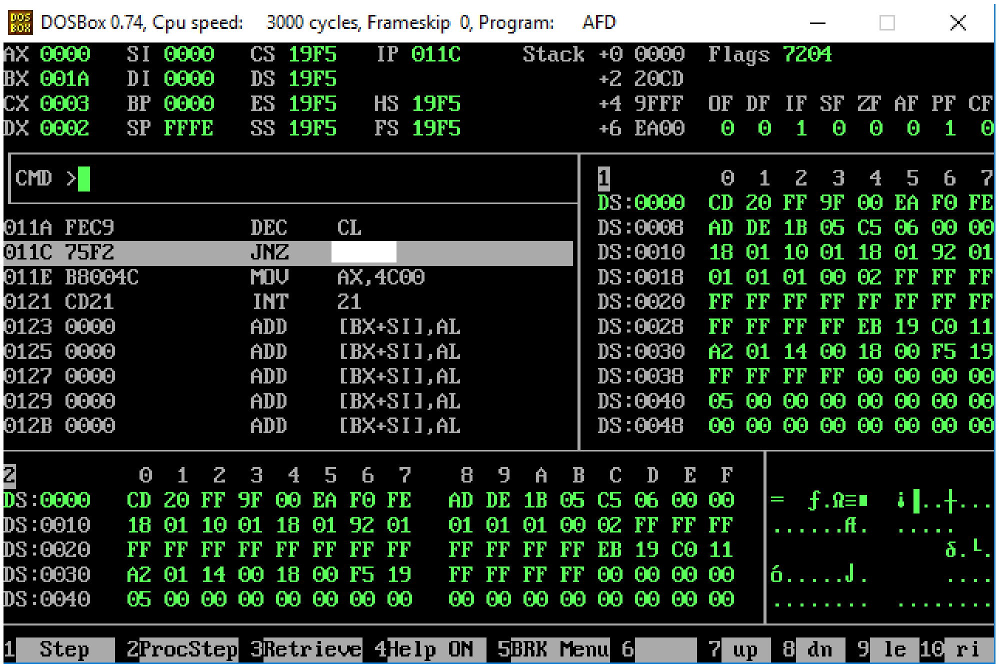
Effective Address = bx + si

= 0100 + 00E1

= 01E1

Task#2: Conditional, Unconditional Jumps and Relative Addressing: **(04 marks)**

Question: Analyze the given Relative Address/Jump Address and explain the reason behind the value 0110 i.e. JNZ 0110 [You can consult the helping material provided with the assignment i.e. "conditionalJumpsCodes" and "RelativeAddressing"]

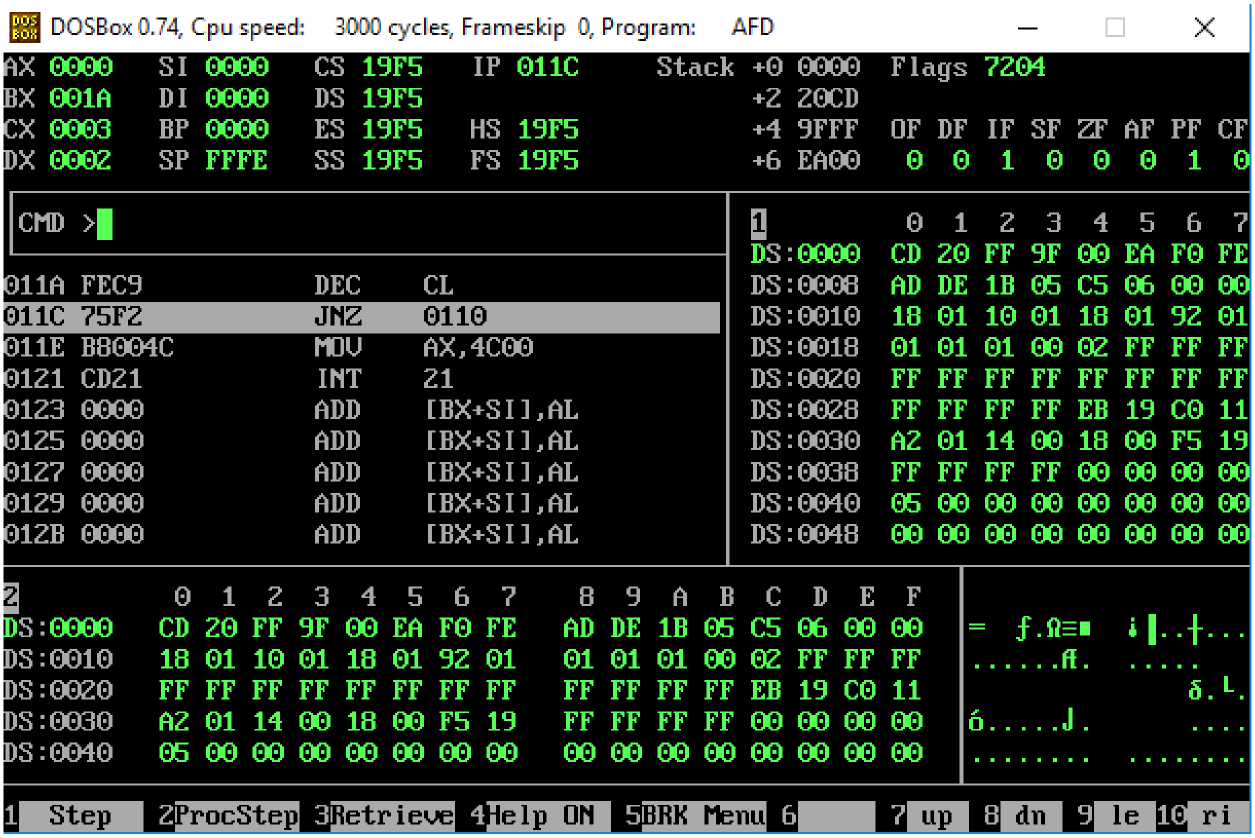

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

```
AX 0000   SI 0000   CS 19F5   IP 011C      Stack +0 0000  Flags 7204
BX 001A   DI 0000   DS 19F5                      +2 20CD
CX 0003   BP 0000   ES 19F5   HS 19F5            +4 9FFF   OF DF IF SF ZF AF PF CF
DX 0002   SP FFFE   SS 19F5   FS 19F5            +6 EA00    0  0  1  0  0  0  1  0

 CMD >                                          1        0  1  2  3  4  5  6  7
                                                DS:0000  CD 20 FF 9F 00 EA F0 FE
011A FEC9           DEC     CL                  DS:0008  AD DE 1B 05 C5 06 00 00
011C 75F2           JNZ                         DS:0010  18 01 10 01 18 01 92 01
011E B8004C         MOV     AX,4C00             DS:0018  01 01 01 00 02 FF FF FF
0121 CD21           INT     21                  DS:0020  FF FF FF FF FF FF FF FF
0123 0000           ADD     [BX+SI],AL          DS:0028  FF FF FF FF EB 19 C0 11
0125 0000           ADD     [BX+SI],AL          DS:0030  A2 01 14 00 18 00 F5 19
0127 0000           ADD     [BX+SI],AL          DS:0038  FF FF FF FF 00 00 00 00
0129 0000           ADD     [BX+SI],AL          DS:0040  05 00 00 00 00 00 00 00
012B 0000           ADD     [BX+SI],AL          DS:0048  00 00 00 00 00 00 00 00

2           0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
DS:0000    CD 20 FF 9F 00 EA F0 FE  AD DE 1B 05 C5 06 00 00   =  ƒ.Ω≡▪  ¡▌..┼...
DS:0010    18 01 10 01 18 01 92 01  01 01 01 00 02 FF FF FF   ......Æ.  .....
DS:0020    FF FF FF FF FF FF FF FF  FF FF FF FF EB 19 C0 11                δ.└.
DS:0030    A2 01 14 00 18 00 F5 19  FF FF FF FF 00 00 00 00   ó.....J.      ....
DS:0040    05 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........  ........

1 Step  2ProcStep 3Retrieve 4Help ON 5BRK Menu 6      7 up 8 dn 9 le 10 ri
```



```
DOSBox 0.74, Cpu speed:    3000 cycles, Frameskip  0, Program:    AFD
AX 0000   SI 0000   CS 19F5   IP 011C      Stack +0 0000  Flags 7204
BX 001A   DI 0000   DS 19F5                      +2 20CD
CX 0003   BP 0000   ES 19F5   HS 19F5            +4 9FFF  OF DF IF SF ZF AF PF CF
DX 0002   SP FFFE   SS 19F5   FS 19F5            +6 EA00   0  0  1  0  0  0  1  0

 CMD >                                         1          0  1  2  3  4  5  6  7
                                               DS:0000   CD 20 FF 9F 00 EA F0 FE
011A FEC9          DEC    CL                   DS:0008   AD DE 1B 05 C5 06 00 00
011C 75F2          JNZ    0110                 DS:0010   18 01 10 01 18 01 92 01
011E B8004C        MOV    AX,4C00              DS:0018   01 01 01 00 02 FF FF FF
0121 CD21          INT    21                   DS:0020   FF FF FF FF FF FF FF FF
0123 0000          ADD    [BX+SI],AL           DS:0028   FF FF FF FF EB 19 C0 11
0125 0000          ADD    [BX+SI],AL           DS:0030   A2 01 14 00 18 00 F5 19
0127 0000          ADD    [BX+SI],AL           DS:0038   FF FF FF FF 00 00 00 00
0129 0000          ADD    [BX+SI],AL           DS:0040   05 00 00 00 00 00 00 00
012B 0000          ADD    [BX+SI],AL           DS:0048   00 00 00 00 00 00 00 00

2          0  1  2  3  4  5  6  7    8  9  A  B  C  D  E  F
DS:0000   CD 20 FF 9F 00 EA F0 FE   AD DE 1B 05 C5 06 00 00   =  ƒ.Ω≡■  ¡▌..┼...
DS:0010   18 01 10 01 18 01 92 01   01 01 01 00 02 FF FF FF   ......Æ.  .....
DS:0020   FF FF FF FF FF FF FF FF   FF FF FF FF EB 19 C0 11                 δ.└.
DS:0030   A2 01 14 00 18 00 F5 19   FF FF FF FF 00 00 00 00   ó.....J.      ....
DS:0040   05 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00   ........  ........

1 Step  2ProcStep 3Retrieve 4Help ON  5BRK Menu 6      7 up  8 dn  9 le 10 ri
```

Solution:

IP or PC = 011E

Offset = F2

Two's complement of F2 is E

Jump Address = 011E + (-000E) = 011E – 000E= 0110

Reason:

Conditional Jumps are always short. Range of short jump is -127 to 128.

+ive Offset or Forward Jump = 0000 0000 to 0111 1111 = 00H to 7FH

-ive Offset or Backward Jump=

First, you ***invert*** each **bit** of the offset byte (giving its **1's Complement**): **FF**h (1111 1111) —> **00**h (**0**000 0000) and **80**h (**1**000 0000) —> **7F**h (**0**111 1111). Following this, you simply **add 1** to each intermediate value, then make it a negative number. So, the **2's Complement** of each byte is in reality: **FF**h —> **-01**h (a -1) and **80**h —> **-80**h (a -128); these are not only *conceptually* **negative** numbers, but also *electronically,* or there couldn't be ***backward*** jumps (the CPU *knows* they are negative offsets because the first byte **EB**, tells it this is a SHORT Jump instruction where any value from 80h to FFh is treated as such).

Task#3: Bit Manipulation: **(03 marks)**

Question 1: Suppose AL contains 10011011b and CF= 0. Give the new contents of AL after each of the following instructions is executed. Assume the preceding initial condition for each part of this Question is AL contains 10011011b and CF= 0.

a) SHL AL,1

b) SHR AL, CL if CL contains 3

c) ROL AL ,1

d) SAR AL, CL if CL contains 3

e) RCR AL,CL if CL contains 2

Solution:

a) AL=00110110 CF=1

b) AL=00010011 CF=0

c) AL=00110111 CF=1

d) AL=11110011 CF=0

e) AL=10100110 CF=1

Question 2: Need to turn on Bit 4 and Bit 7 of a byte (remember that the bit on the right-hand side is Bit 0). Define the mask and logical operator accordingly and show your result.

Solution:

7	6	5	4	3	2	1	0	**Bit Position**
0	0	0	0	0	0	0	0	**Data**
1	0	0	1	0	0	0	0	**Mask**
1	0	0	1	0	0	0	0	**Logical Operator – OR Result**