National University of Computer and Emerging Sciences

Lab Manual

Computer Organization and Assembly Language

Lab 04

Instructor Hazoor Ahmad

Class CS3

Sections A, D, H, K

Semester Fall 2022

Fast School of Computing

FAST-NU, Lahore, Pakistan

Objectives

- How to interpret the different types of jumps
- How to use the different types of registers and how to manipulate them in assembly language
- How to perform arithmetic operations with registers and conditional jumps •
How to use the debugger for viewing the available registers and their function

Contents

Note for all questions: You can make as many memory variables as you need

ACTIVITY 1:

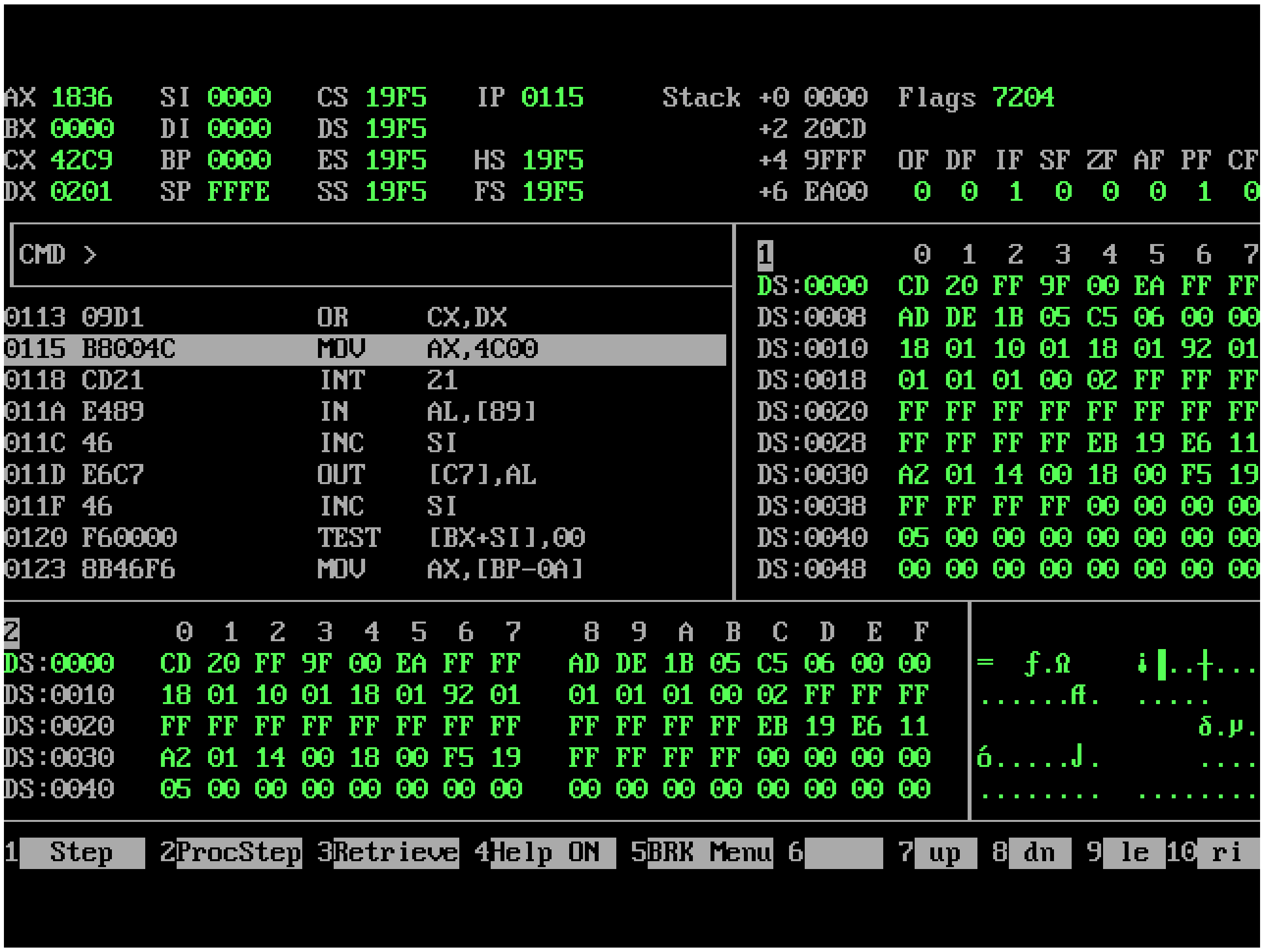
Initialize ���� with last 4 digits of your roll number (for example, if your roll number is 16L-1105 then ���� should be initialized with 1105).
Once initialized, write a program to swap every pair of bits in the AX register as shown in **Table** below:

AX	**Contents of AX (Your Roll #)**			
Befor e	**0000**	**0100**	**0101**	**0001**
After	**0000**	**1000**	**1010**	**0010**

—---

```
[org 0x0100]
mov ax, 1836h
mov cx, 0011001100110011b
and cx, ax
mov dx, 1100110011001100b
and dx, ax
shl cx,2
shr dx,2
```

```
or cx, dx
mov ax, [0x4c00]
int 0x21
```



```
AX 1836   SI 0000   CS 19F5   IP 0115     Stack +0 0000  Flags 7204
BX 0000   DI 0000   DS 19F5               +2 20CD
CX 42C9   BP 0000   ES 19F5   HS 19F5     +4 9FFF  OF DF IF SF ZF AF PF CF
DX 0201   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 FF FF
0113 09D1            OR      CX,DX              DS:0008  AD DE 1B 05 C5 06 00 00
0115 B8004C          MOV     AX,4C00            DS:0010  18 01 10 01 18 01 92 01
0118 CD21            INT     21                 DS:0018  01 01 01 00 02 FF FF FF
011A E489            IN      AL,[89]            DS:0020  FF FF FF FF FF FF FF FF
011C 46              INC     SI                 DS:0028  FF FF FF FF EB 19 E6 11
011D E6C7            OUT     [C7],AL            DS:0030  A2 01 14 00 18 00 F5 19
011F 46              INC     SI                 DS:0038  FF FF FF FF 00 00 00 00
0120 F60000          TEST    [BX+SI],00         DS:0040  05 00 00 00 00 00 00 00
0123 8B46F6          MOV     AX,[BP-0A]         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 FF FF   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 E6 11             δ.µ.
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
```

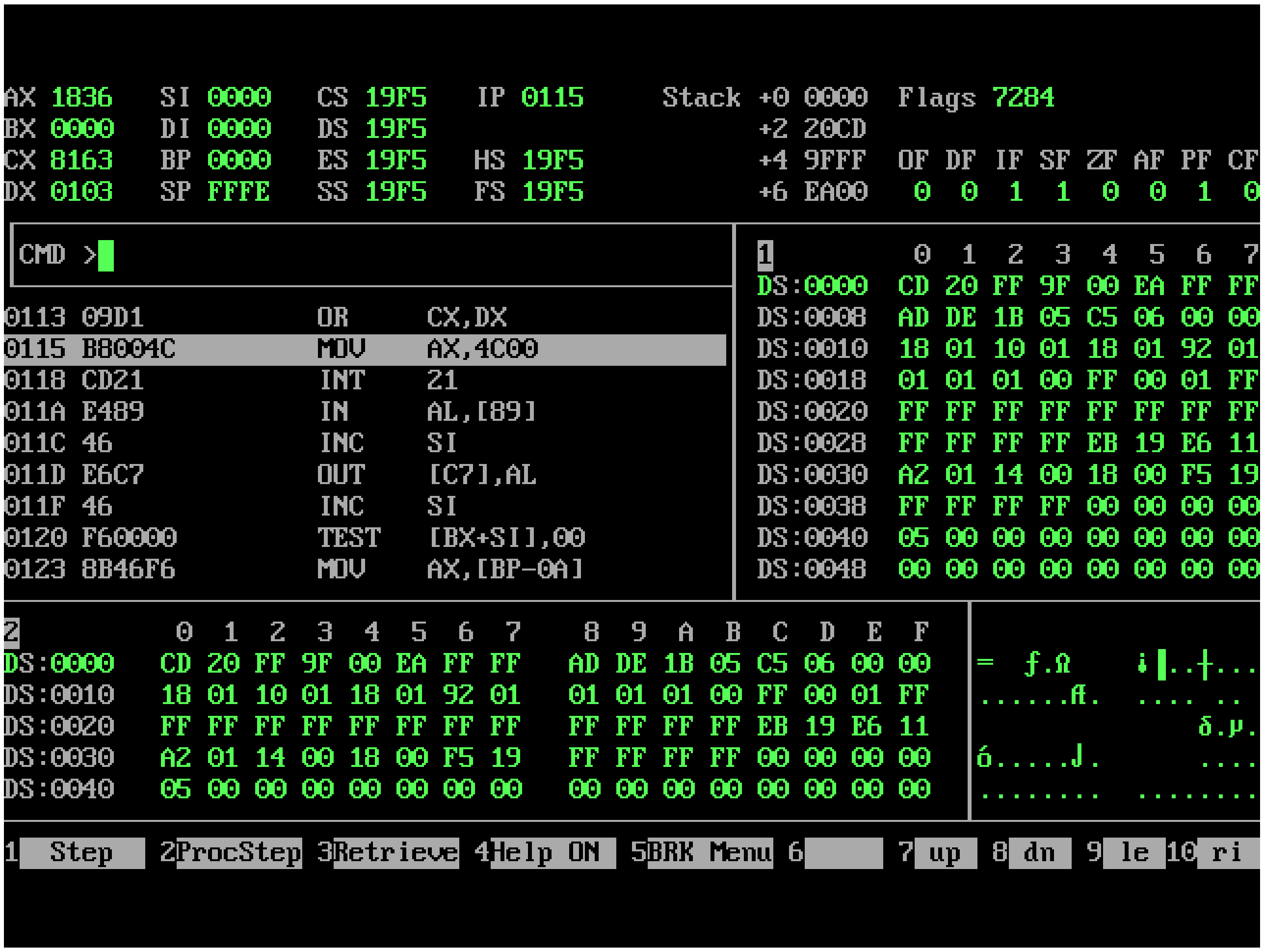
ACTIVITY 2:

Modify your program in Activity 1 to swap two bits as shown in **Table** below:

AX	Contents of AX (Your Roll #)			
Before	0000	0100	0101	0001
After	0000	0001	0101	0100

```
[org 0x0100]
mov ax, 1836h
mov cx, 0000111100001111b
and cx, ax
mov dx, 1111000011110000b
and dx, ax
```

```
shl cx,4
shr dx,4
or cx, dx
mov ax, [0x4c00]
int 0x21
```



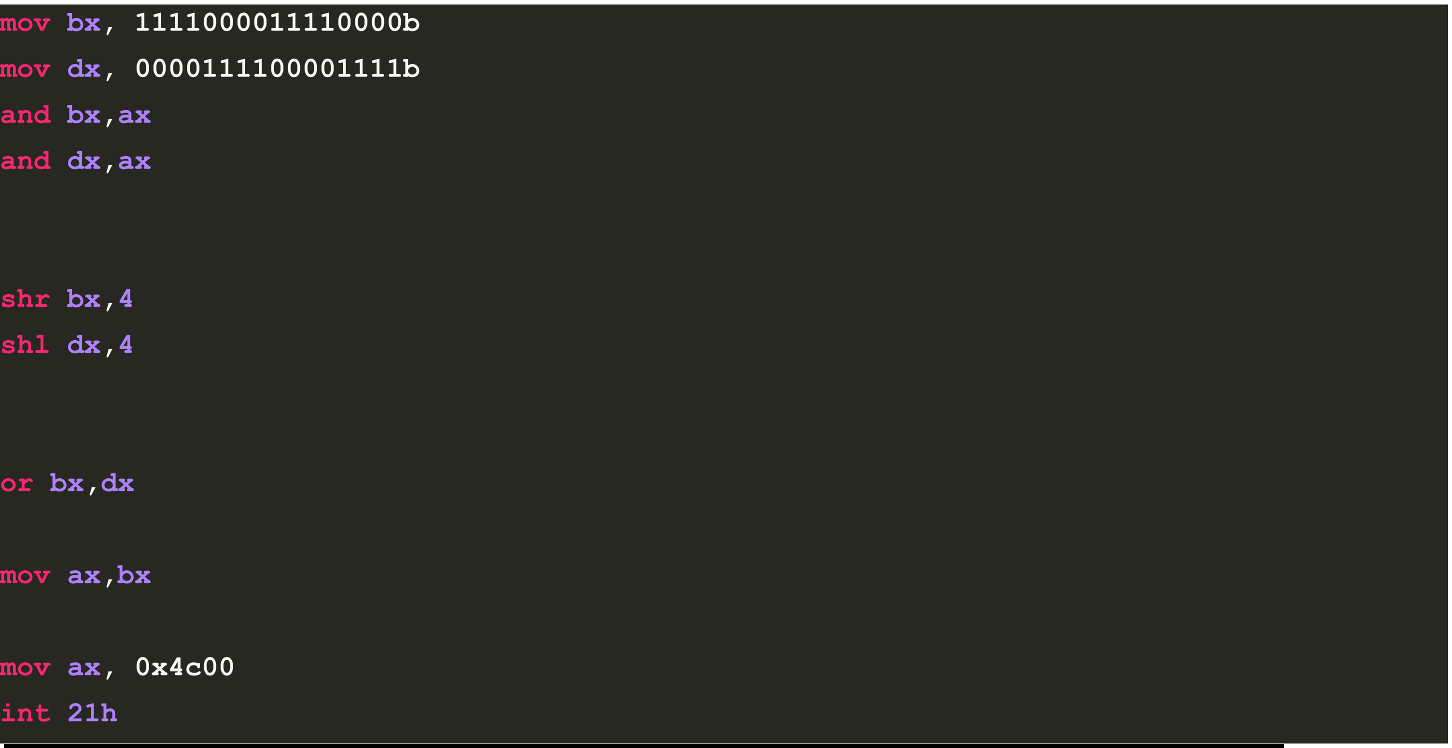
```
AX 1836   SI 0000   CS 19F5   IP 0115     Stack +0 0000  Flags 7284
BX 0000   DI 0000   DS 19F5               +2 20CD
CX 8163   BP 0000   ES 19F5   HS 19F5     +4 9FFF  OF DF IF SF ZF AF PF CF
DX 0103   SP FFFE   SS 19F5   FS 19F5     +6 EA00   0  0  1  1  0  0  1  0

 CMD >                                       1         0  1  2  3  4  5  6  7
                                             DS:0000   CD 20 FF 9F 00 EA FF FF
0113 09D1          OR      CX,DX             DS:0008   AD DE 1B 05 C5 06 00 00
0115 B8004C        MOV     AX,4C00           DS:0010   18 01 10 01 18 01 92 01
0118 CD21          INT     21                DS:0018   01 01 01 00 FF 00 01 FF
011A E489          IN      AL,[89]           DS:0020   FF FF FF FF FF FF FF FF
011C 46            INC     SI                DS:0028   FF FF FF FF EB 19 E6 11
011D E6C7          OUT     [C7],AL           DS:0030   A2 01 14 00 18 00 F5 19
011F 46            INC     SI                DS:0038   FF FF FF FF 00 00 00 00
0120 F60000        TEST    [BX+SI],00        DS:0040   05 00 00 00 00 00 00 00
0123 8B46F6        MOV     AX,[BP-0A]        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 FF FF  AD DE 1B 05 C5 06 00 00   = ƒ.Ω    ↓▌..┼...
DS:0010   18 01 10 01 18 01 92 01  01 01 01 00 FF 00 01 FF   ......Æ.  ....  ..
DS:0020   FF FF FF FF FF FF FF FF  FF FF FF FF EB 19 E6 11               δ.µ.
DS:0030   A2 01 14 00 18 00 F5 19  FF FF FF FF 00 00 00 00   ó.....⌡.      ....
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
```

ACTIVITY 3

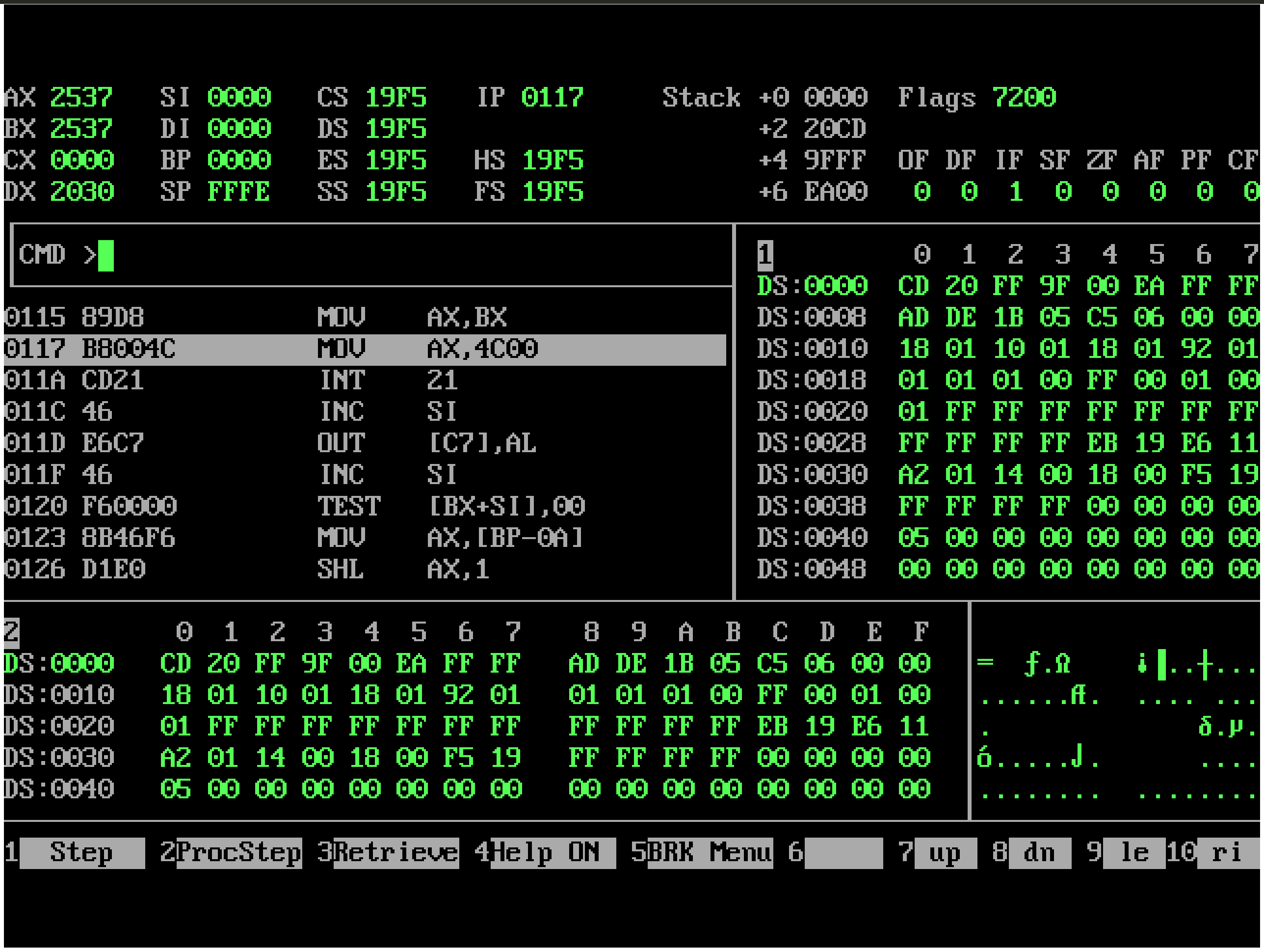
Modify your program in Activity 1 & 2 to swap two nibbles as shown in **Table** below:

AX	Contents of AX (Your Roll #)			
Befor e	0000	0100	0101	0001
After	0100	0000	0001	0101

```
[org 0x0100]

mov ax, 5273h
```



```
mov bx, 1111000011110000b
mov dx, 0000111100001111b
and bx,ax
and dx,ax


shr bx,4
shl dx,4


or bx,dx

mov ax,bx

mov ax, 0x4c00
int 21h
```



```
AX 2537   SI 0000   CS 19F5   IP 0117     Stack +0 0000  Flags 7200
BX 2537   DI 0000   DS 19F5                     +2 20CD
CX 0000   BP 0000   ES 19F5   HS 19F5           +4 9FFF  OF DF IF SF ZF AF PF CF
DX 2030   SP FFFE   SS 19F5   FS 19F5           +6 EA00   0  0  1  0  0  0  0  0

 CMD >                                          1          0  1  2  3  4  5  6  7
                                                DS:0000   CD 20 FF 9F 00 EA FF FF
0115 89D8           MOV    AX,BX                DS:0008   AD DE 1B 05 C5 06 00 00
0117 B8004C         MOV    AX,4C00              DS:0010   18 01 10 01 18 01 92 01
011A CD21           INT    21                   DS:0018   01 01 01 00 FF 00 01 00
011C 46             INC    SI                   DS:0020   01 FF FF FF FF FF FF FF
011D E6C7           OUT    [C7],AL              DS:0028   FF FF FF FF EB 19 E6 11
011F 46             INC    SI                   DS:0030   A2 01 14 00 18 00 F5 19
0120 F60000         TEST   [BX+SI],00           DS:0038   FF FF FF FF 00 00 00 00
0123 8B46F6         MOV    AX,[BP-0A]           DS:0040   05 00 00 00 00 00 00 00
0126 D1E0           SHL    AX,1                 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 FF FF   AD DE 1B 05 C5 06 00 00   = ƒ.Ω      ¡▌..┼...
DS:0010   18 01 10 01 18 01 92 01   01 01 01 00 FF 00 01 00   ......Æ.   ....  ...
DS:0020   01 FF FF FF FF FF FF FF   FF FF FF FF EB 19 E6 11   .             δ.µ.
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
```

ACTIVITY 4:

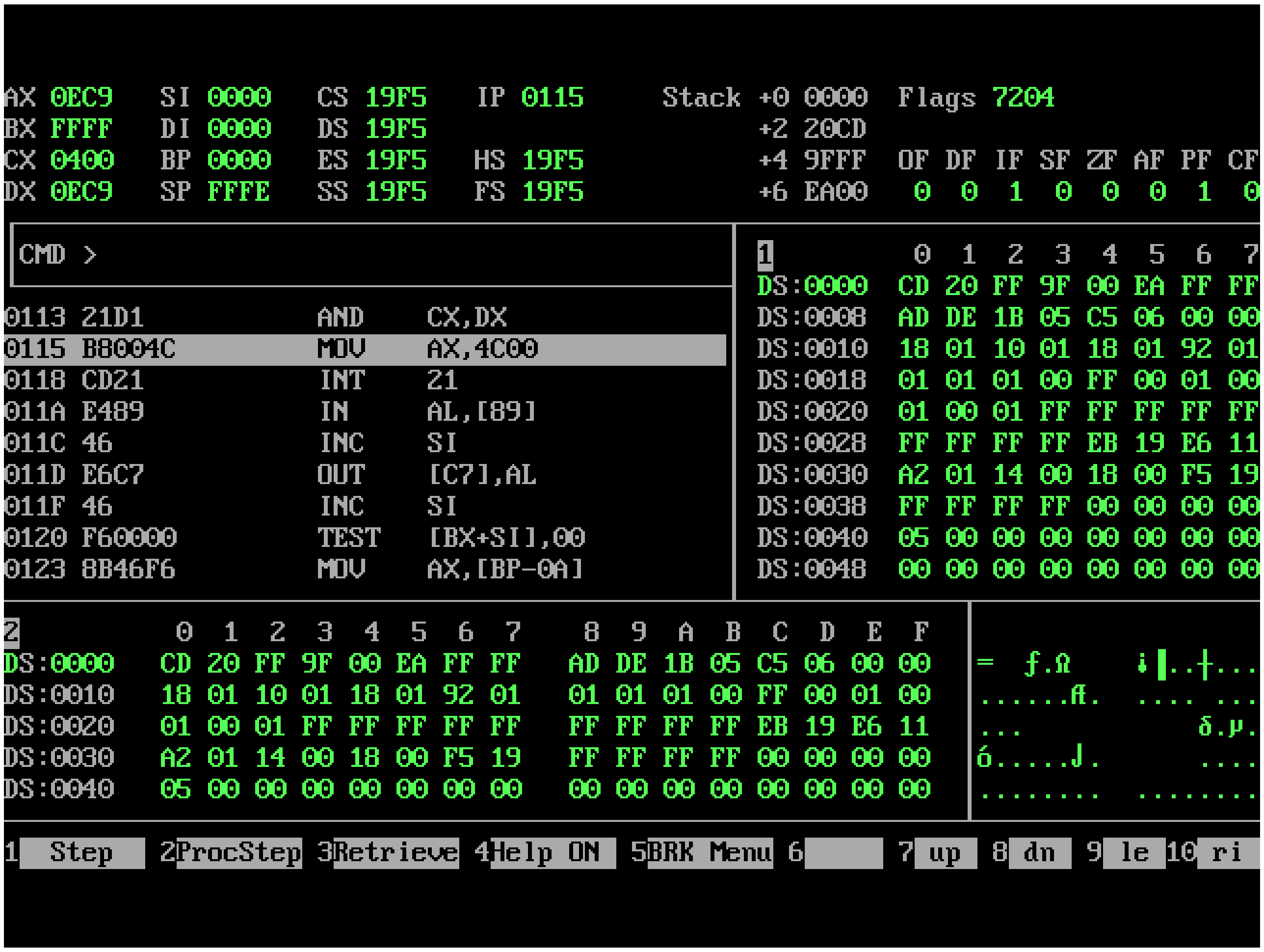
Initialize ���� with last 4 digits of your roll number (for example, if your roll number is 16L-1105 then ���� should be initialized with 1105). Store ���� in ����. Make a memory variable ��, initialize it with 0 and compute

�� = (��||��)&&(�� ⊙ 0��1������)

|| is bitwise OR operation, && is bitwise AND operation whereas ⊙ is bitwise XOR operation.

—---

```
[org 0x0100]


mov ax, 5380

mov cx, 1111111111111111b

mov bx, ax

xor bx, cx

or  bx, ax

mov cx, ax

xor ax, 1BCDh

mov dx,ax

and cx,dx


mov ax , 0x4c00

int 21h
```



```
AX 0EC9   SI 0000   CS 19F5   IP 0115     Stack +0 0000  Flags 7204
BX FFFF   DI 0000   DS 19F5                     +2 20CD
CX 0400   BP 0000   ES 19F5   HS 19F5           +4 9FFF  OF DF IF SF ZF AF PF CF
DX 0EC9   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 FF FF
0113 21D1          AND    CX,DX               DS:0008  AD DE 1B 05 C5 06 00 00
0115 B8004C        MOV    AX,4C00             DS:0010  18 01 10 01 18 01 92 01
0118 CD21          INT    21                  DS:0018  01 01 01 00 FF 00 01 00
011A E489          IN     AL,[89]             DS:0020  01 00 01 FF FF FF FF FF
011C 46            INC    SI                  DS:0028  FF FF FF FF EB 19 E6 11
011D E6C7          OUT    [C7],AL             DS:0030  A2 01 14 00 18 00 F5 19
011F 46            INC    SI                  DS:0038  FF FF FF FF 00 00 00 00
0120 F60000        TEST   [BX+SI],00          DS:0040  05 00 00 00 00 00 00 00
0123 8B46F6        MOV    AX,[BP-0A]          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 FF FF   AD DE 1B 05 C5 06 00 00   = ƒ.Ω   ↓▌..┼...
DS:0010   18 01 10 01 18 01 92 01   01 01 01 00 FF 00 01 00   ......Æ.  .... ...
DS:0020   01 00 01 FF FF FF FF FF   FF FF FF FF EB 19 E6 11   ...          δ.µ.
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
```

—--

ACTIVITY 5:

Initialize ���� with last 4 digits of your roll number (for example, if your roll number is 16L-1105 then ���� should be initialized with 1105). Store ���� in ����. Make a 32-bit memory variable ��, initialize it with 0 and compute

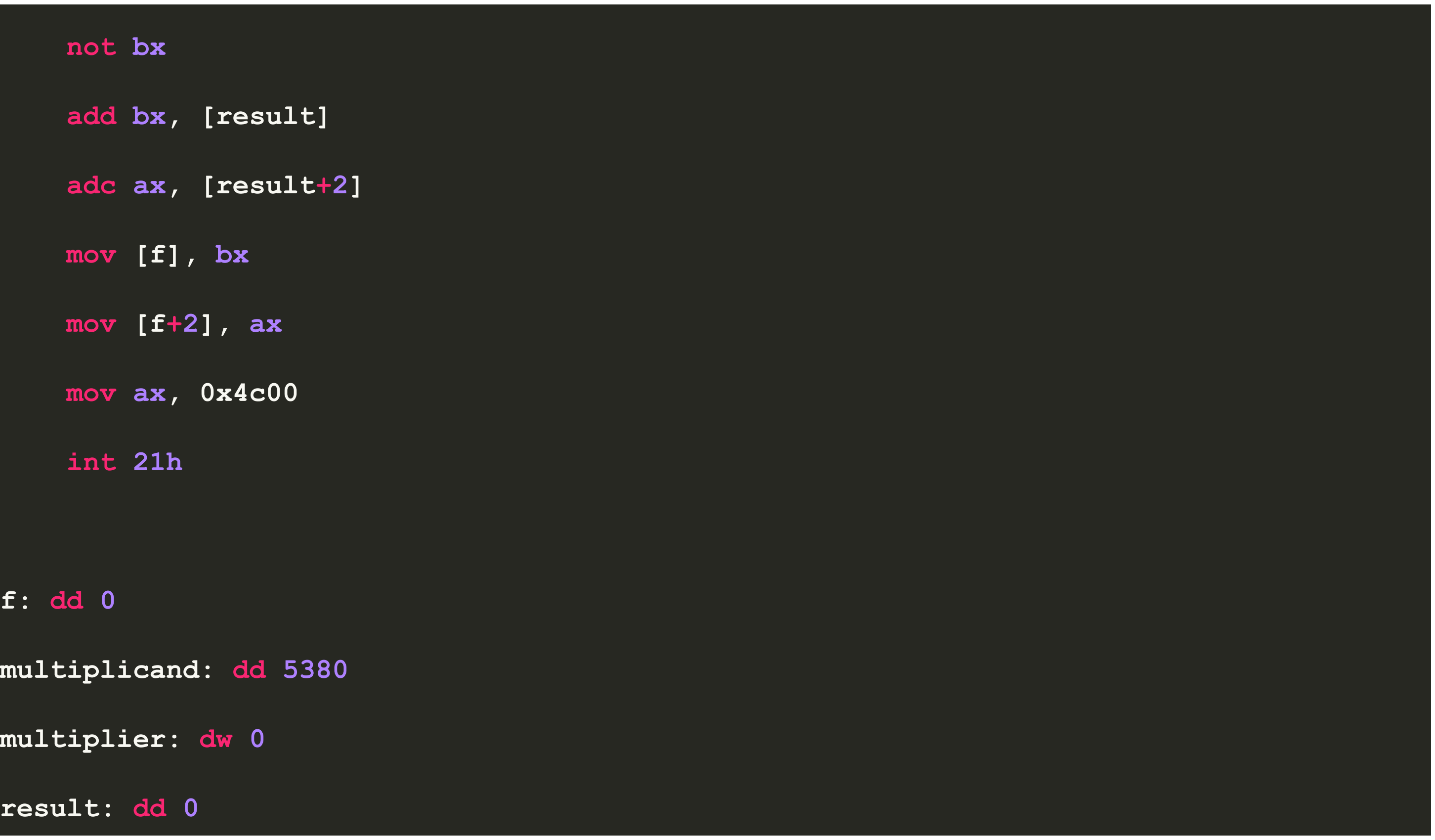
$$�� = (�� \times ��) + \{��, ��\}$$

$\times$ is **Multiplication** operation, $+$ is **Addition** operation whereas $\{��, ��\}$ **concatenates** 16-bit **A** and **B** to form **32-bit** number.

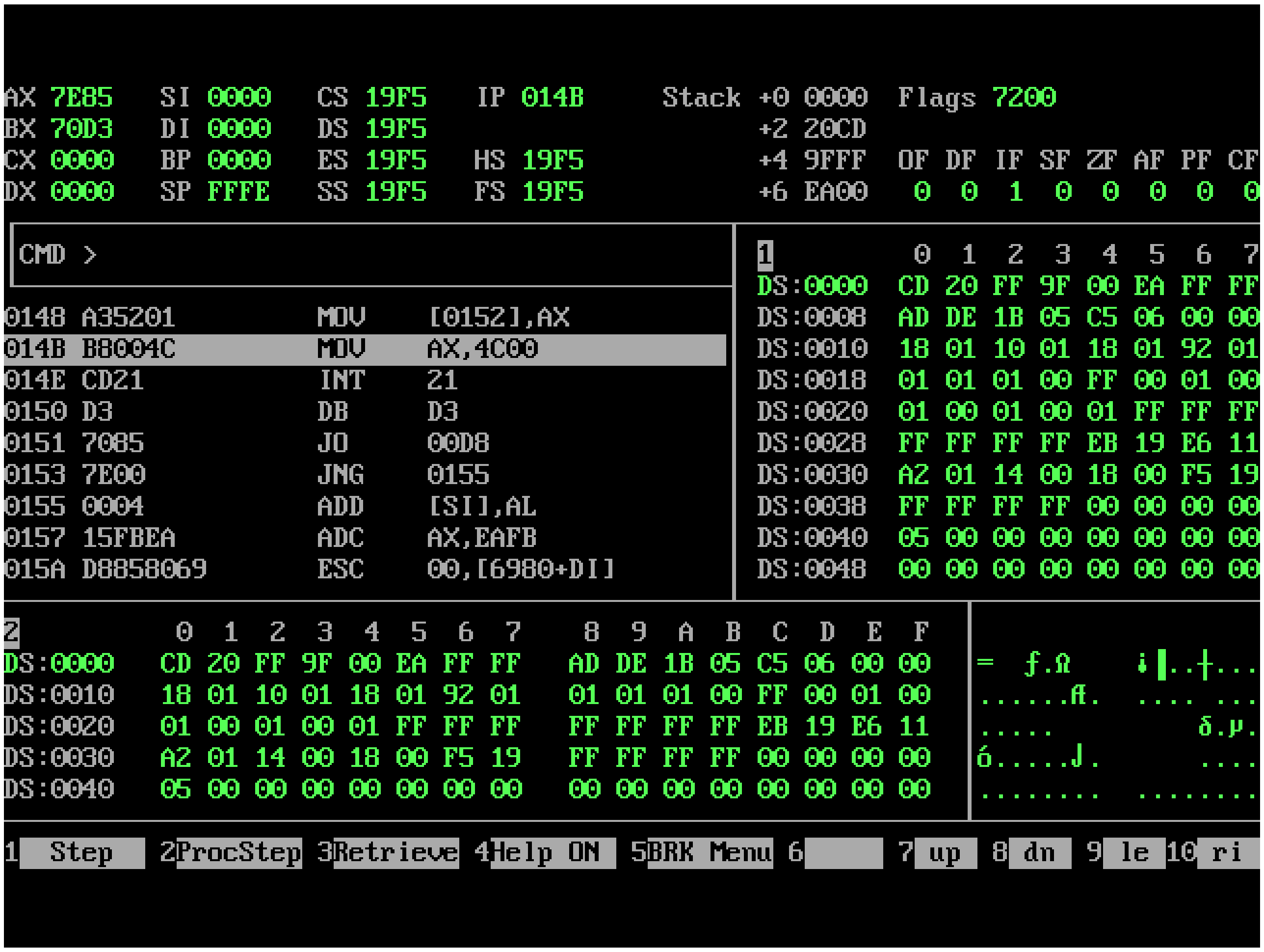
—--

```
 [org 0x0100]

mov ax,5380

mov bx, ax
```

```
not bx
mov word [multiplicand], ax
mov word [multiplier], bx

mov cl, 16
mov bx, 1

checkbit:
    test bx, [multiplier]
    jz skip


    mov ax, [multiplicand]
    add [result], ax
    mov ax, [multiplicand + 2]
    adc [result + 2], ax

    skip:
        shl word [multiplicand], 1
        rcl word [multiplicand + 2], 1
        shl bx , 1

dec cl
jnz checkbit

mov ax, 5380
mov bx, ax
```



```
    not bx

    add bx, [result]

    adc ax, [result+2]

    mov [f], bx

    mov [f+2], ax

    mov ax, 0x4c00

    int 21h


f: dd 0

multiplicand: dd 5380

multiplier: dw 0

result: dd 0
```



```
AX 7E85   SI 0000   CS 19F5   IP 014B     Stack +0 0000  Flags 7200
BX 70D3   DI 0000   DS 19F5                     +2 20CD
CX 0000   BP 0000   ES 19F5   HS 19F5           +4 9FFF  OF DF IF SF ZF AF PF CF
DX 0000   SP FFFE   SS 19F5   FS 19F5           +6 EA00   0  0  1  0  0  0  0  0

 CMD >                                       1         0  1  2  3  4  5  6  7
                                             DS:0000   CD 20 FF 9F 00 EA FF FF
0148 A35201          MOV     [0152],AX       DS:0008   AD DE 1B 05 C5 06 00 00
014B B8004C          MOV     AX,4C00         DS:0010   18 01 10 01 18 01 92 01
014E CD21            INT     21              DS:0018   01 01 01 00 FF 00 01 00
0150 D3              DB      D3              DS:0020   01 00 01 00 01 FF FF FF
0151 7085            JO      00D8            DS:0028   FF FF FF FF EB 19 E6 11
0153 7E00            JNG     0155            DS:0030   A2 01 14 00 18 00 F5 19
0155 0004            ADD     [SI],AL         DS:0038   FF FF FF FF 00 00 00 00
0157 15FBEA          ADC     AX,EAFB         DS:0040   05 00 00 00 00 00 00 00
015A D8858069        ESC     00,[6980+DI]    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 FF FF   AD DE 1B 05 C5 06 00 00   =  ƒ.Ω    ¡▌..┼...
DS:0010   18 01 10 01 18 01 92 01   01 01 01 00 FF 00 01 00   ......Æ.  .... ...
DS:0020   01 00 01 00 01 FF FF FF   FF FF FF FF EB 19 E6 11   .....         δ.µ.
DS:0030   A2 01 14 00 18 00 F5 19   FF FF FF FF 00 00 00 00   ó.....╜.      ....
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
```

—--

ACTIVITY 6:

Differentiate between Near, Far and Short Jumps. Write your own assembly language programs and demonstrate how these jumps have been taken.

REFERENCES

- "http://www.dosbox.com/download.php?main=1
- http://sourceforge.net/projects/nasm
- http://www.nasm.us/
  - http://www.programmersheaven.com/download/21643/download.aspx (AFD)