Assignment -1

OC

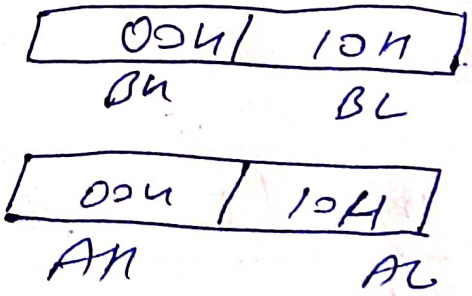
YASH GUPTA
SX020001029

Q-1 Explain the workflow of the following instruction MOV AX, BX

Ans

- The MOV instruction is the most important command in the 8086 because it moves data from one location to another.
- MOV AX, BX is a register addressing mode,
- Operand value is present in register BX.
- The instruction MOV AX, BX transfer the data of BX to AX.
- for eg MOV BX, 0010H

MOV AX, BX



Q-2

Explain the execution of following & which flags will be set?

(a) MOV DH, 3 : This is an immediate addressing mode 3H will be stored in DH.

NEG DH : 2's of complement is applied for value at DH and is stored there only.

Carry flag, Sign flag, Auxillary flag are set after this instruction

NOTE : NEG DH, 2 → error

SUB DH, 1 : Value stored at DH will be decreased by 1 and stored in DH

DH = DH - 1

Sign flag, parity flag

SUB DH, 73H ; The hexa decimal value at DH is subtracted by hexa decimal 73 and stored in DH

DH = DH - 73

Sign flag.

ADD DH, 79H ; The value at DH is added with 79H and stored in DH

DH = DH + 79

Carry & Auxillary

ADD DH, 02H ; The value at DH is added with 02H and stored in DH

DH = DH + 2

No flags

CMP DH, 0F AH ; The value at DH is compared with 0F AH

Carry flag, parity flag and auxillary flag

ADD DH, 03H ; The value at DH is added to 3 and stored in DH

No flags

(b) MOV AX, OFEH ; AX → OFEH (Immediate addressing mode)

INC AX ; AX → AX + 0001 H

(Parity flag)

MOV BX, 0 ; BX → 0000H (Immediate addressing)

ADD BX, AX ; BX → BX + AX

Parity flag

NEG BX ; BX → 2's complement of value at BX

Carry, Sign, auxillary flag

SUB AX, BX ; AX → AX - BX

Carry flag

NEG AX ; ABX → 2's complement of value at AX

Carry, Sign, auxillary flag

NEG BX ; BX → 2's complement of value at BX

SBB AX ; subtract BX from AX along with borrow included

Sign, auxillary

DEC AX ; AX → AX - 1

Sign flag

ADC BX, BX ; BX → BX + BX + Carry

auxillary

SUB AX, BX ; AX → AX - BX

Sign, auxillary .

NEG BX ; BX → 2's complement of BX

Carry, Sign, auxillary

NEG AX ; AX → 2's complement of AX

Carry, Auxillary

(c) MOV CL, 1H ; CL → 1H (immediate)

SUB CL, 65 H ; CL → CL - 65H

Carry, Sign, parity, auxillary

NEG CL, 27H → Error

instead NEG CL ; CL → 2's complement of CL

Carry, auxillary

ADD CL, 86 H ; CL → CL + 86H

Sign flag

SBB CL, 33H , CL → CL - 33H with borrow

Sign, parity flag

(d) MOV AX, 0F2F H ; AX → F24F H

MOV BX, 0A3F5H ; BX → A3F5H

ADD BX, AX ; BX → BX + AX

Carry, Sign, Parity & Auxillary

SBB AX, 0F24E H ; AX → AX - F24E H with borrow

Zero and parity flag

MOV CX, 1000 H ; CX → 1000H

ADC CX, 0F14C H ; CX → CX + F14C

Carry flag

ADC CX, 0EEEE H ; CX → CX + EEEE H

Sign flag, parity flag, Auxillary flag

(e) MOV AX, 9800H ; AX → 9800H

SUB AX, 0FFF H ; AX → AX - FFF H

Carry, Auxillary flag

MOV CX, 5750 ; CX → 5750 H

ADC CX, 0AABBH ; CX → CX + AABB H + Carry

Carry, parity flag

MOV BX, 0EBB3 H ; BX → EBB3 H

ADC BX, 144CH ; BX → 144C H + Carry

Carry, parity, auxillary flags

MOV BX, 0002H ; BX → 2H

ADD BX, 0EEEEH ; BX → BX + EEEE H

Sign, parity, auxillary

Q.3 $Z = (X^2 + Y^2)/(X^2 - Y^2)$

X and Y are 8 bit numbers

Z is a 16 bit number

Ans

```
DATA SEGMENT
        X DB 31H
        Y DB 12H
        Z DW 0000H

DATA ENDS

CODE SEGMENT
        ASSUME CS: CODE DS: DATA

START: MOV AX, DATA; initializing the data segment
       MOV DS, AX
       MOV AL, X ; AL → 31H
       MUL AL ; AX → AL * AL
       MOV [500H], AX ; stores X^2 in 500H
                        AL in 500H
                        AH in 501H

       MOV AL, Y ; AL → 12H
       MUL AL ; AL → AL * AL &
       MOV [600H]; stores Y^2 in 600H
                   AL in 600H
                   AH in 601H
```

MOV AX, [SOH] ; moving to AX from SOH

MOV BX, [SOH] ; moving to BX from SOH

ADD AX, BX ; AX → AX + BX

MOV CX, [SOH] ; moving to CX from SOH

SUB CX, BX ; CX → CX - BX

DIV CX ; AX → (DX)(AX)/(CX)
Quotient in AX
remainder in DX

LEA BX, Z ; loading address of Z in BX

MOV [BX], AX ; storing result in Z

STOP : POP

CODE ENDS

END START

Q-9 Two numbers x1 & x2 are stored in memory. Perform following operation & store in x3.

x3 = (x1 AND 0FH) XOR(x2)

Ans

```
DATA SEGMENT
        X3  DB 00H
DATA SEGMENT
CODE SEGMENT
        ASSUME CS: CODE, DS: DATA

START
        MOV AX, 2000 ; Initializing Data register
        MOV DS, AX
        MOV [600H], 31H ; Storing 31H at 600H
                          offset value
        MOV [700H], 12H ; Storing 12H at 700H offset value
        MOV CH, [600H] ; copying value at 600H to CH
        AND CH, 0FH ; CH -> CH AND 0FH
        MOV DH, [700H]; copying value at 700H to DH
        XOR CH, DH ; CH -> CH ⊕ DH
        LEA BX, X3 ; loading offset value of
                     X3 to BX
        MOV [BX], CH ; storing the final result
                       at X3
STOP : NOP
        CODE ENDS
        END START
```

Q-5

```
DATA SEGMENT
    ARRAY1 DB 21H,25H, 26H, 26H, 00H,01H, 02H,
              03H, 09H,05H,06H
              ; declaring array 1 as Array1

    ARRAY2 DB 21H,25H,26H,00H, 01H, 02H, 03H,09H,06H
              ; declaring array 2 as Array 2

DATA ENDS

CODE SEGMENT
    ASSUME CS: CODE  DS: DATA

START: MOV AX, DATA
    MOV DS, AX
    MOV AX, 0000H ; Storing 0000H at AX

    LEA SI, ARRAY1; loading address of 1st element
                    of array1 in SI

    LEA DI, Array2; loading address of 2nd element
                    of array2 in DI.

    MOV CX, 0000AH ; Storing 000AH in CX

LABEL: MOV AL, [SI] ; moving address of SI in AL
    MOV BL, [DI] ; moving address of DI in BL

    CMP AL, BL ; Comparing value of AL, BL and
                 storing in AL

    JNZ LABEL1 ; jump if not zero to label1

    INC SI ; Increment SI
    INC DI ; Increment DI
    DEC CX ; Decrement CX
    JNZ LABEL; jump if not zero to Label
    MOV [5000H], 01H; storing 01H in 5000H
    JMP STOP
```

```
LABEL ; MOV [5000H], 00H  ; storing 00H in 5000H

STOP: NOP

CODE ENDS
   ENDS START
```

Q-76

```
DATA SEGMENT
     STRING  DB 'MY NAME IS'
     STRING1 DB  22 DUP(0)
DATA ENDS

CODE SEGMENT
     ASSUME CS:CODE, DS :DATA ES:DATA

START
     MOV AX, STRING ; Initialising data and ES registers
     MOV DS, AX
     MOV ES, AX
     MOV AX, 0000H ; clearing the value at AX
     LEA SI, string ; loading SI with address of
                      string
     LEA DI, string1 ; loading DI with address of
                       string1
     MOV CL, 50H ; CL → 0BH for looping
                   loop
TAG: MOV AL,[SI] ; storing string in STRING1 using
     STOSB                loop
     INC SI
     DEC CL
     JNZ TAG
```

```
MOV AL,'S' ; adding SANTA to MY NAME IS
STOSB
MOV AL,'A'
STOSB
MOV AL,'N'
STOSB
MOV AL,'T'
STOSB
MOV AL,'A'
STOSB
MOV AL,' '
STOSB
MOV AL,'C' ; adding CLAUS to MYNAME IS
STOSB                    SANTA
MOV AL,'L'
STOSB
MOV AL,'A'
STOSB
MOV AL,'U'
STOSB
MOV AL,'S'
STOSB
STOP: NOP
    CODE ENDS
    END START
```

Q-7 In the following program determine the values of X1, Y1, Z1 at the end.

```
DATA SEGMENT
    X1 DW 0302H
    Y1 DW 071SH
    Z1 DW f227H
DATA ENDS
CODE SEGMENT
    ASSUME CS: CODE, DS: DATA
START:
    MOV AX, DATA ; Initializing DATA Segment
    MOV DS, AX
    MOV AX, X1 ; AX → 0302H
    MOV BX, Y1 ; BX → 071SH
    MOV CX, Z1 ; CX → f227H
    PUSH CX ; CX is pushed to stack
    PUSH BX
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH AX
    XCHG BX, Z1 ; Exchange value of BX, Z1
    XCHG CX, AX ; Exchange value of CX, AX
    MOV Z1, AX ;
    MOV Y1, CX
    MOV X1, BX
CODE ENDS
END START
```

Ans $X1 = 0f22H$, $Y1 = 0302H$ $Z1 = 0f227H$

Q-8 See if you can spot the grammatical errors in the following instructions

(a) MOV BH, AX

- AX is 16 bit register while BH is 8 bit register.
- 16 bit data can't be stored in a 8 bit register.

(b) MOV DX, CL

- DX is 16 bit register and CL is 8 bit register.
- 8 bit data can't be stored in a 16 bit register.

(d) MOV 7632H, CX

- We can't store a value of CX register in 7632H value, But 7632H will consider as a memory location and proceeds

(c) ADD AL, 2073H

- We can't add the 8 bit data and a 16 bit data

(e) IN BL, 04H

Data can't be transfered from port its BL register. It can only be transfered to AL, AX and DX registers

Q-9 Describe the function of each assembler directives and instruction statement in the following short program

```
;PRESSURE READ PROGRAM
DATA_HERE SEGMENT
        PRESSURE DB 0; Storage for pressure
DATA HERE ENDS
PRESSURE - PORT EQU 04H; Pressure sensor
                         connected to port 04H
CORRECTION-FACTOR EQU 07H; Current factor
                           of 07H
CODE- HERE SEGMENT
        ASSUME CS: CODE-HERE, DS: DATA HERE
        MOV AX, DATA-HERE; Initialize Data segment
        MOV DS, AX
        IN AL, PRESSURE-PORT; value of pressure in
                              stored in AL register
        ADD AL, CORRECTION-FACTOR;
                value of correction factor is
                add to AL content and
                replaced
        MOV PRESSURE, AL; the value of
                          of AL register is
                          transferred to pressure
CODE- HERE ENDS
        END
```

ASSEMBLER DIRECTIVES USED:

DB - Data Byte - Refers as byte data
This DB reserves a byte in the program memory

EQU - Defining assembler constant
associates a constant value to the name of a constant

END - End of program
It shows that there are no instructions after it

Q-1> Write the 8086 instructions which will perform the indicated operation.

a. copy AL to BL
MOV BL, AL

b. Load 43H in CL
MOV CL, 43H

c. Increment the contents of CX by 1.
INC CX

d. Add 07 to DL
ADD DL, 07H

d Copy SP to BP
MOV BP, SP

f multiply AL time BL

MUL BL

g Copy Ax to a memory location at offset 245A4 in the data segment

mov [245AH], Ax ; AL will be at 245A4
AH will be at 245B4

h. Reserved SP by 1

DEC SP

i. Rotate the most significant bit of AL into the least significant bit position.

ROL AL, 1

j. copy DL to a memory location whose offset is in Bx

mov [Bx], DL

k. Mask the most significant bit of AX to a 1, but do not affect the other bits

OR AX, 8000H

l. Invert the lower 4 bits of BL, but do not affect the other bits

XOR, BL, 0FH

Q-11 Actual location (address) of the memory for the following command

MOV AX, SOURCE [BX][SI]

= 10 * DS + SOURCE + [BX] + [SI]

Q-12 What will happen if you execute the following command in 8086.

(a) PUSH (SOURCE)

This instruction pushes the contents of SOURCE and SOURCE+1 to the stack. The stack pointer is decremented by 2.

(b) CMP AX, BX

It will compare AX & BX by doing Substraction

	Cf	Zf	Sf
AX > BX	0	0	0
AX = BX	0	1	0
AX < BX	1	0	1

(c) NEG AL

2's complement of AL is calculated and replaced.

(d) TEST AX, BX

Test instruction is used to add operands & update the flags without changing the values in either of the registers

Q-13 an assembly language program which converts the fahrenheit temperature to celsius using the following relation: $C = (F-32)5/9$

```
DATA SEGMENT
    F DB 29 ; consider fahrenheit = 41°F
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE DS:DATA
START: MOV AX, DATA : Initialising Data segment
    MOV DS, AX
    MOV AX, 0000H ; clearing AX register
    MOV AL, F ; copying F value to AL
```

```
SUB AL, 20H ; AL → AL - 20H = 09H
MOV BH, 05H ; BH → 05H
MUL BH ; AX → BH * AL = 45H
MOV CH, 09 ; CH → 09H
DIV CH ; AL → quotient = 05H
MOV F, AL ; copying AL to F
```

```
STOP : NOP
CODE ENDS
END START
```

Q-11 See if you can find any errors in the following instructions or groups of instructions

a. CNTDOWN : MOV BL, -72H
DEC BL
JNZ CNTDOWN

This will make an infinite loop

b. ADD CX, AL
addition can't be performed b/w 8-bit & 16bit registers

c JMP BL
There is no such instruction.
We have to use label instead of registers.

d. JNZ [BX]

There is no such instruction we have to use label instead of a location.

Q-15 Compute the average of 4 bytes stored in a array in memory.

```
DATA SEGMENT
     NUMS DB 01H, 0AH, 05H, 08H
     AVG  DB 1 DUP(?)
DATA ENDS
CODE SEGMENT CS:CODE, DS:DATA
START: MOV AX, DATA ; Initializing the data segment
       MOV DS, AX
       MOV AX, 0000H
       MOV CL, 04H ; taking CL as counter in loop
       LEA BX, NUMS ; loading BX with NUMS starting address
HERE:  ADD AL, [BX] ; Implementing loop and adding all elements and storing in AL
       INC BX
       DEC CL
       JNZ HERE ; if CL=0 loop will stop
```

```
MOV  CL, 04H ; CL -> 04H
MOV DIV CL ; Quoted of AX/CL is
                in AL
LEA BX, AVG ; load BX with
                AVG address
mov [BX], AL ; storing average of 4
                number in memory

START : NOP
        CODE ENDS
        END START
```