1800411120
Md Farhan Ghmam

CSE 4503 (Microprocessor)

24 June 2021

Ans. to Q.no. 2 (a)

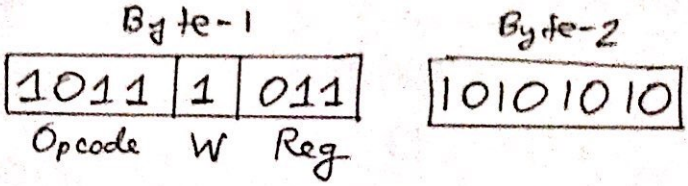
```
ORG 100h
START:  MOV AH, 1       ; set value of AH for input
        INT 21H         ; calling interrupt
        MOV BL, AL

        MOV AH, 2       ; set value of AH for output
        MOV DL, 0AH     ; for new line (line feed)
        INT 21H

        MOV DL, 0DH     ; for carriage return
        INT 21H

        MOV AL, BL
        ADD AL, 1       ; incrementing (i.e. next character)
        MOV DL, AL
        INT 21H         ; We don't need to move AH, 2
                        ; because AH already has 2
RET
```

Ans. to Q.no. 2(b)

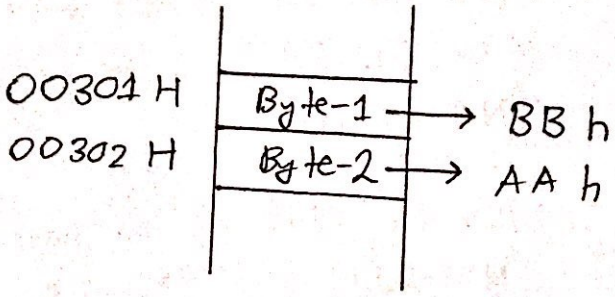
(i)

MOV BX, 10101010B

The machine code is



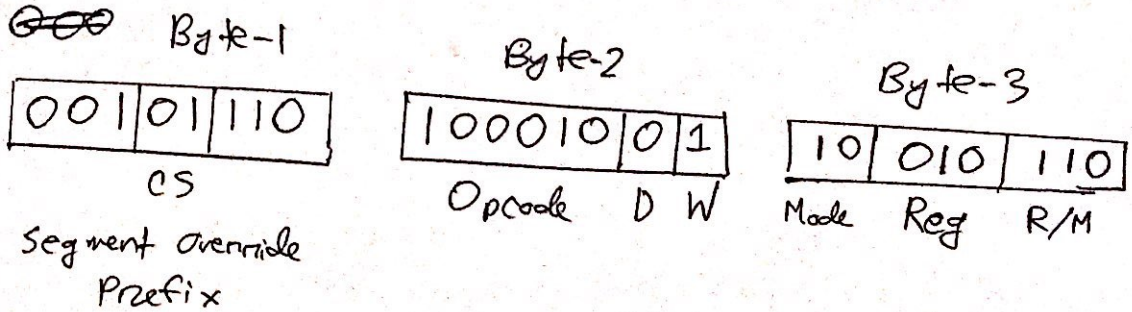
We need two memory locations to store these.



(ii)

MOV CS: [BP+1234H], DX

The machine code is



Byte-4

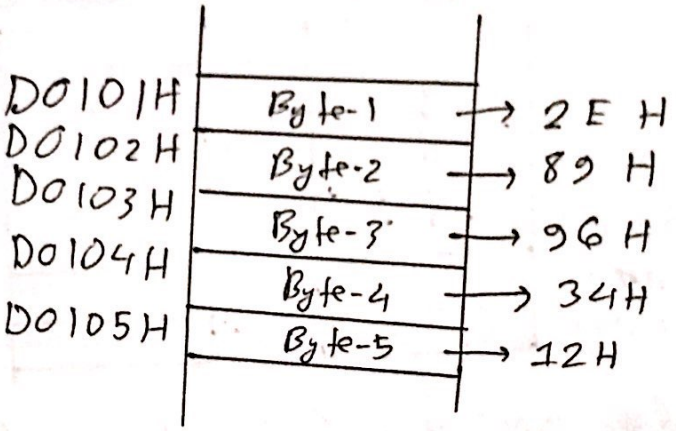
0011 0100

34H

Byte-5

0001 0010

12H



(5 Locations)

(iii)

IN AL, 192.

Byte-1

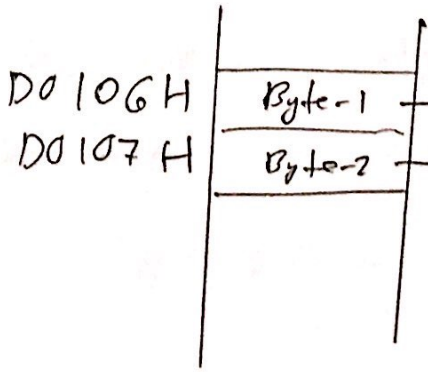
1110010	0
Opcode	W

[192 → C0h]

Byte-2

1100 0000

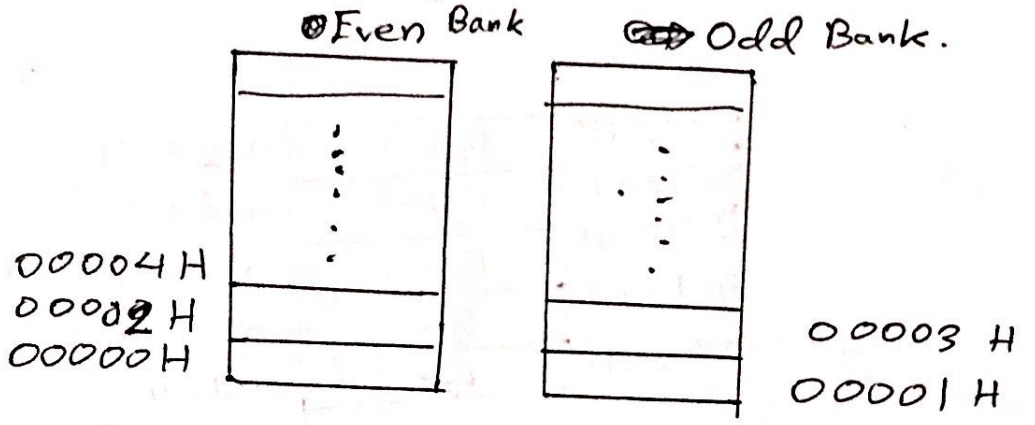
C0h



(2 Locations)

Ans. to Q.no. 2(c)

(i) In 8086 memory is partitioned where ~~odd~~ even and odd memory locations are located seperately.



The data are stored in two columns (one with odd memory locations and one with even).

While fetching data, CPU collects a single row. If location 00000H and 00001H needs to be accessed then 1 CPU cycle is enough. But to access 00001H and 00002H, CPU needs 2 cycles. (for 16 bit data)

Thus, the partitions with even addresses ~~are~~ is the even bank and with odd addresses is odd bank.

Ans. to Q.no. 2(c) (ii)

The flags of 8085 are -

(i) Sign flag - if MSB is 1 then 1 (i.e negative)
if MSB is 0, then 0 (i.e positive)

(ii) Zero flag - if value is 0, then 1.
if value is not 0, then 0.

(iii) Auxillary flag - Carry from 4th to 5th bit.
For BCD operations.

(iv) Carry flag - Carry from 8th bit, then 1.
If no carry, then 0.

(v) Parity - if even parity exists, then 1.
otherwise 0.

8086 has all these flags and more which are -

(i) Trap flag → 1 - debugging mode
0 - normal mode

(ii) Interrupt flag → CPU will receive and respond interrupts if 1.

(iii) Direction flag - DF=1 means forward memory access
DF=0 means backward memory access

(iv) Overflow Flag - if sign bit gets changed then OF is 1, else 0.

Ans to Q.no 1(a)

→ Exactly 64 kb for code segment
→ default for stack
→ size of data segment more than 64 Kb

So, we use .modelcompact (Ans.)

Ans. to Q no. 1(b)

~~After executing MOV AL, 1Ah~~

After executing all instructions,

AL value

MOV AL, 1Ah → 0001 1010

NOT AL → 1110 0101

ADD AL, A1h

```
  1 1 1 0 0 1 0 1
  1 0 1 0 0 0 0 1
---------------------
1 0 1 0 0 0 1 1 0
```

carry

So, the value of AL will be 1000 0110 B

or, 86 h

The values of flag will be

CF = 1 [There is carry 1]

PF = 0 [Odd no. of 1s]

AF = 0 [From 4th to 5th bit no carry]

ZF = 0 [value is not zero]

SF = 1 [leftmost bit is 1]

DF = 0 [processing is done forward]

Ans to Q.no. 1(c)

(i) AND and TEST

(ii) NOT and NEG

NOT does 1's complement to the ~~data, operand~~ register value

NEG does 2's complement to the register value

2's complement is 1's complement plus 1.

Ex - MOV AL, 01h

NOT AL

01h 00000001

1's complement 11 11 11 10

So, output will be FEh

MOV AL, 01h

NEG AL,

01h 00000001

1's comp 1111 1110

+ 00000001

2's comp 1111 1111

So, output is FFh

(iii) SUB and CMP

Subtraction stores the result in the destination register i.e. the first register.

CMP doesn't store the result value in register and only updates the flags.

Ex-

```
MOV AX, 1
MOV BX, 1
SUB AX, BX
```

Value of AX will be 0.

```
MOV AX, 1
MOV BX, 1
CMP AX, BX
```

Value of AX won't change, it will be 1.

Ans. to Q. no. 3(a)

```
ORG 0100h
.DATA
P DB 10000001
~~Q DB 1A2BH~~
Q DB 1A2BH
R DW 260
S DB ?
.CODE
MAIN PROC
    MOV AL,P
    MOV BL,2
    MUL AL,BL
    MOV CL,R
    MOV S,Q
    MAIN ENDP
END MAIN
RET
```

Here, it is a mistake because default type is decimal. There should be a B.

Connection: 10000001B

DB can store 8 bits only

Connection: DW 1A2BH

MUL doesn't need AL to be specified

Connection: MUL BL

R is DW which is 16 bits. We need to have same size of source and dest register

Connection: MOV CX, R

Q is ~~DB~~ DW, and S is DB. So, this can not be performed.

Connection: D MOV DX, Q.

Ans. to Q.no. 3(b)

(i)

Given, physical location 4A37B h

Segment number 40FF h

phy location = (seg no.) x 10h + offset

=) offset address = phy location - (seg no) x 10h.

segment number x 10h = 40FF x 10

= 40FF0 h

∴ Offset address is

```
  4A37B
  40FF0
 -------
  0938B
```

Ans: 938B

Ans. to Q no. 3(b) (ii)

$$\text{phy loc} = (\text{seg\_no}) \times 10h + \text{offset}$$

$$\Rightarrow \text{segment\_no.} = \frac{\text{phy-loc} - \text{offset}}{10h}$$

Physical address $4A37Bh$

Offset $- 0123Bh$

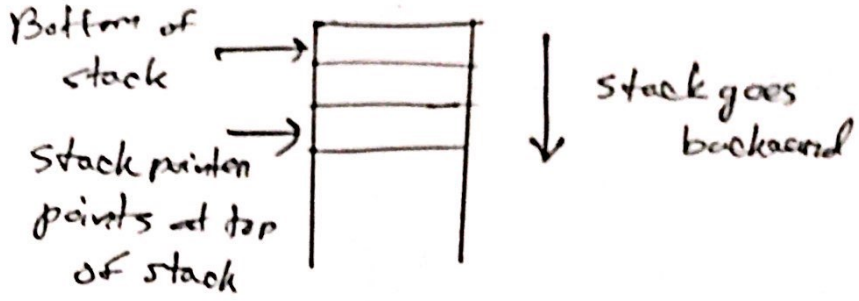
$49140h$

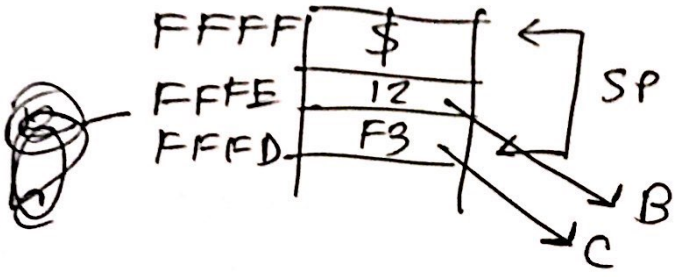
Dividing by 10, we get,

segment_no. $= 4914h$ (Ans.)

Ans. to Q no. 3(c)

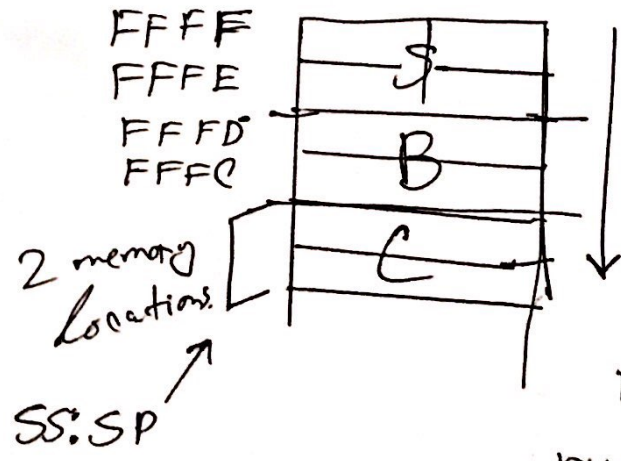
For 8085, only stack pointer is used to access stack. The memory goes backward in direction.





First ~~we push~~ stack pointer ~~points~~ points to the bottom of stack ~~FFFF~~, FFFF. Then we push B and it ~~stores points~~ stores to FFFE and then pushing C it ~~points~~ stores to FFFD.

For 8086,



The address is calculated using stack segment and stack pointer. The formula of physical address is used to do this. Each operation takes 2 memory locations.

First, SP is FFFE. If we push B, then it is FFFC.

Physical location is calculated by $SS \times 10_h + SP$,