Assignment 00

Name: Ridwanul Haque
ID: 1721144042
Section: 02

Course: CSE 331

Submission date: 09-04-2020

Submitted To: Rishad Arfin (RSF)

1.

```
org 100h
.MODEL SMALL
.STACK 100H
L0: NOP          ; NO OPERATION
L1: MOV AX,0000H ; set segment to 0000H
    MOV DS,AX
    MOV BX,0000H ; set offset to 0000H
    MOV CL,BYTE PTR DS:[BX] ; read data from first
                              memory location
L2: MOV AX,0F000H
    MOV DS,AX
    MOV BX,0FFFFH
    MOV CH,BYTE PTR DS:[BX] ; read data from last
                              memory location
L3: ADD CL,CH    ; adding
L4: MOV AX,0F0000H
    MOV DS,AX
    MOV BX,0FFFEH
    MOV BYTE PTR DS:[BX],CL ; store added data
                              in last even
                              memory location
L5: HLT          ; Halt
```

```
org 100H
2. L1: NOP
   L2: IN AL,00H
       ~~MOV AL,~~
       MOV CL,AL    ; Read data from first memory location
                      of fix port
       IN AL,01H
       MOV CH,AL    ; Read data from second memory location
                      of fix port
   L3: ADD CL,CH
   L4: MOV AX,0F000H
       MOV DS,AX
       MOV BX,0FFFFH
       MOV BYTE PTR DS:[BX],CL ; load result in
                                 last memory location
   L5: HLT
```

3.

```
org 100h

L0: NOP

L1: MOV AX,0F000H
    MOV DS, AX
    MOV BX, 0FFFFH
    MOV CL, BYTE PTR DS:[BX]   ; load data from last
                               memory location

L2: MOV AX, 0000H
    MOV DS, AX
    MOV BX, 0000H
    MOV CH, BYTE PTR DS:[BX]   ; load data from
                               first memory location

L3: ADD CL,CH

L4: MOV AL,CL
    OUT 0FEH,AL    ; save added value in last
                   even memory address of
                   FIX port.

L6: HLT
```

4. org 100 H

L0: NOP

L1: MOV DX, OFFFFH ; read from last location of variable port

IN AL, DX

MOV CL, AL

L2: MOV DX, 0000H

IN AL, DX

MOV CH, AL

L3: ADD CL, CH

MOV AL, CL

L4: OUT OFEH, AL

Q.5] org 100H

```
L0: NOP
L1: MOV AX, 0000 H   ; set segment to 0000
    MOV DS, AX
    MOV BX, 0000 H   ; set offset to 0000
    MOV DL, BYTE PTR DS:[BX] ; read data from 00000H
L2: MOV BX, 0001 H
    ADD DL, BYTE PTR DS:[BX]  ; read data from 00001H
L3: MOV BX, 0002H
    ADD DL: BYTE PTR DS:[BX] ; read data from 00002H
L4: MOV AL, DL
    MOV DX, 0FOFFH ; Move data to FOFFH of
    OUT DX, AL      ; variable port address
L5: HLT
ret
```

6] org 100H

```
L0: NOP
L1: MOV AX, 0F000H
    MOV DS, AX
    MOV BX, OFFFFH
    MOV DL, BYTE PTR DS:[BX] ; last memory location
L2: MOV AX, 0000H
    MOV DS, AX
    MOV BX, 0000H
    MOV DH, BYTE PTR DS:[BX] ; read from first location
L3: ADD DL, DH
L4: CMP DL, 5    ; compare result with 5
    JLE L6       ; Jump if less
L5: MOV AX, 0F000H
    MOV DS, AX
    MOV BX, OFFFEH
    MOV BYTE PTR DS:[BX], DL ; last even memory
                             location
L6: HLT
ret
```

7.1 org 100h :

```
L0: NOP
L1: MOV AX, 0F000H
    MOV DS, AX
    MOV BX, 0FFFFH
    MOV DL, BYTE PTR DS:[BX] ; read from last memory address

L2: MOV AX, 0000H
    MOV DS, AX
    MOV BX, 0000H
    MOV DH, BYTE PTR DS:[BX] ; read from memory address

L3: SUB DL, DH    ; substract

L4: JS L5         ; Jump if sign is ~~positive~~ ~~negative (result)~~ negative
    MOV AL, DL
    MOV DX, 0FFFFH
    OUT DX, AL ; positive result in variable port

L5: MOV AL, DL
    OUT 00H, AL ; negative result in fixed port address

L6: HLT

ret
```

81 org 100h

```
L0: NOP
L1: MOV CX, 0000H
    MOV DS, CX
    MOV BX, 0000H
    MOV DL, BYTE PTR DS:[BX]

L2: MOV CX, 0000H
    MOV DS, CX
    MOV BX, 0001H
    MOV DH, BYTE PTR DS:[BX]

L3: CMP DL, DH      ; compare
L4: JA L5           ; Jump to L5 is if greater
    MOV AL, DH
    OUT 00H, AL     ; store in fix port
    JMP L6          ; Jump to DOS
L5: MOV AL, DL
    OUT 00H, AL     ; store in fix port
L6: HLT
ret
```

9| .MODEL SMALL
.STACK 100H

.DATA
P DB 'Number is Positive$'
N DB 'Number is Negative$'

.CODE
MAIN PROC
MOV CX, 0000H
MOV DS, CX
MOV BX, 0000H
MOV AL, BYTE PTR DS:[BX] ; set the value to 00000H

TEST AL, 80H ; CHECK MSB
JNZ L0 ; Jump if ZF zero
MOV AX, @DATA
MOV DS, AX ; initialize string in Data segment
LEA DX, P ; Display P
MOV AH, 9 ; Displaying Built in function
INT 21H ; interrupt
JMP end ; Jump to end label

L0:
MOV AX, @DATA
MOV DS, AX
LEA DX, N ; Display string N
MOV AH, 9
INT 21H

end:
MOV AH, 4CH ; Built in function to return DOS
INT 21H
MAIN ENDP
END MAIN

10| .STACK 100H

.DATA

odd DB 'The number is odd$' ; string save

even DB 'The number is even$'

.CODE

MAIN PROC

MOV CX, 0000H

MOV DS, CX

MOV BX, 0000H

MOV CL, BYTE PTR DS:[BX] ; read CL from 00000H

TEST CL, 01H ; check LSB

JNZ ODD ; Jump if 1 or not zero

MOV AX, @DATA ; initialize Data to segment

MOV DS, AX

LEA DX, even ; print even in prompt

INT 21H ; interrupt

JMP END ; Jump to END

ODD:

MOV AX, @DATA

MOV DS, AX

LEA DX, ~~ODD~~ odd

MOV AH, 9

INT 21H

END:

MOV AH, 4CH

INT 21H

MAIN ENDP

END MAIN

11 .MODEL SMALL

```
.STACK 100H
.DATA
    msg1 DB 'The number is palindrom $'
    msg2 DB 'The number is not palindrom $'
.CODE
MAIN PROC
    MOV CX, 0000H
    MOV DS, CX
    MOV BX, 0000H
    MOV AL, BYTE PTR DS:[BX] ; load data from 00000H
    MOV DL, AL
    MOV CX, 8    ; loop count
L0: SHL AL, 1    ; shift left and msb in CF
    RCR BL, 1    ;Rotate right through carry in BL
    LOOP L0      ; Back to L0
    CMP DL, BL
    JNZ L1       ; Jump if not zero to L1
    MOV AX, @DATA
    MOV DS, AX   ; Initialize Data to segment
    LEA DX, msg1 ; Print massage to prompt
    MOV AH, 9
    INT 21H
    JMP L2
L1: MOV AX, @DATA
    MOV DS, AX
    LEA DX, msg2 ; Print massage to prompt
    INT 21H
L2: MOV AH, 4CH  ; return to DOS
    INT 21H
MAIN ENDP        ; End main process
END MAIN         ; MAIN END
```

12.
```
.MODEL SMALL
.STACK 100H

.CODE
MAIN PROC
    MOV CX, 0F000H
    MOV DS, CX
    MOV BX, 0FFFFH
    MOV BL, BYTE PTR DS:[BX] ; Read from last memory address
    MOV DL, 00H              ; set DL to 00000000
    MOV CX, 8                ; loop count
L1: ROL BL, 1                ; ROL Left 1 by 1 ; MSB in CF
    JNC L2                   ; carry if 0
    INC DL
L2: LOOP L1

    MOV AH, 4CH              ; return to DOS
    INT 21H
MAIN ENDP
END MAIN
```

13)

```
.MODEL SMALL
.STACK 100H

.CODE
MAIN PROC
    MOV CX,0F000H
    MOV DS,CX
    MOV BX,FFFFH
    MOV BL,BYTE PTR DS:[BX] ; read from last location

    MOV DL,00H ; clear DL
    MOV CX,8   ; LOOP count

L1: ROL BL,1   ; Rotate Left 1 bit
    JC L2      ; Jump if not 0
    INC DL

L2:
    LOOP L1

    MOV AH,4CH ; Return to DOS
    INT 21H

MAIN ENDP
END MAIN
```

14.
```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC

    SHL AL,4        ; multipling by 2,4,8,16 by shifting left
    MOV AH,4CH
    INT 21H
MAIN ENDP
END MAIN
```

15|

```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
    MOV
    SAR AL,4        ; deving by 2,4,6,16
    MOV AH,4CH
    INT 21H
MAIN ENDP
END MAIN
```

16)

```
.MODEL SMALL
.STACK 100H
.DATA
    msg1 DB 'Greater than 9 $'
    msg2 DB 'Less than 9 $'
.CODE
MAIN PROC
    SHL AL, 4       ; Take lower nibble to higher
    SHR AL, 4       ; again take in lower
    CMP AL, 9
    JG L1           ; Jump if greater to L1
    MOV AX, @DATA   ; initializing Data Segment
    MOV DS, AX
    MOV AH, 9
    LEA DX, msg2
    INT 21H         ; interrupt to Display
    JMP L2
L1: MOV AX, @DATA
    MOV DS, AX
    MOV AH, 9       ; string output
    LEA DX, msg1    ; Display msg1
    INT 21H
L2: MOV AH, 4CH
    MOV
    INT 21H

MAIN ENDP
END MAIN
```

17.
```
.MODEL SMALL
.STACK 100H
.DATA
    msg1 DB 'Greater than 9$'        ; load string
    msg2 DB 'Not greater than 9 $'
.CODE
  MAIN PROC
    MOV
    SHR CH,4    ; shift right 4 times
    CMP CH,9
                ; cmp CH with 9
    JG L1
                ; jump if greater than 9
    MOV AX, @DATA
    MOV DS, AX   MOV AH,9   ; string to prompt
    LEA DX, msg2   ; Display Massage
    INT 21H
    JMP END

    L1: MOV AX,@DATA
        MOV DS,AX   MOV AH,9
        LEA DX, msg1
        INT 21H

    END: MOV AH,4CH
         INT 21H

  MAIN ENDP      ; end main Process
  END MAIN
```

18)
```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
    OR BL, 01H      ; set the lsb
    OR BL, 80H      ; set the msb
    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

19)
```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
    AND CL, 0FEH    ; clear lsb
    AND CL, 07FH    ; clear msb
    MOV AH, 4CH
    INT 21H
    MAIN ENDP
    END MAIN
```

20)
```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
    XOR BL, 80H     ; change MSB by XOR
    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

21)
```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
    XOR AL,0AAH   ; changing evenly bits by XOR with AAH
    MOV AH,4CH
    INT 21H
MAIN ENDP
END MAIN
```

22)
```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
    XOR AL, 55H   ; changing oddly bit by XOR with 55H
    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

23)
```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
    MOV CX, 0000H
    MOV DS, CX
    MOV BX, 0000H
    MOV AL, BYTE PTR DS:[BX] ; load from first location
    MOV DL, AL
    MOV CX, 8 ; loop count
    L0: SHL AL, 1   ; get msb in CF
        RCR BL, 1   ; save reversed number in BL
        LOOP L0     ; LOOP L0
    MOV AH, 4CH
    INT 21H
MAIN ENDP
```

24| .MODEL SMALL
.STACK 100H
.CODE
MAIN PROC

XOR CL,CL ; clear cl by XOR with CL
MOV AH,4CH
INT 21H
MAIN ENDP
END MAIN

25| .MODEL SMALL
.STACL 100H
.CODE
MAIN PROC

AND AL,0AAH ; set evenly bit to 0.
MOV AH,4CH
INT 21H
MAIN ENDP
END MAIN

26| .MODEL SMALL
.STACK 100H
.CODE
MAIN PROC

OR CL,~~0AH~~ 0AAH ; set odd index by OR
MOV AH,4CH
INT 21H
MAIN ENDP
END MAIN

27| .MODEL SMALL
.STACK 100H
.CODE
MAIN PROC

```
    ROL AL,4        ; rotate by 4 to exchang nibble & lower to higher
    MOV AH,4CH
    INT 21H
MAIN ENDP
END MAIN
```

~~25~~ .

28| .MODEL SMALL
.STACK 100 H

.DATA
msg1 DB 'EQUAL $'
msg2 DB 'NOT EQUAL $'

.CODE
MAIN PROC
MOV DL, AL ; copping AL in DL
SHL AL, 4 ; taking lower nibble to higher
SHR AL, 4 ; again taking higher nibble to lower
SHR DL, 4 ; taking higher nibble to lower of DL
JNE L1 ; Jump not equal
MOV ~~AX~~ AX, @DATA ; initializing Data segment
MOV DS, AX
MOV AH, 9
LEA DX, msg1 ; Display to prompt
INT 21H
JMP END ; Jump to END stage
L1: MOV AX, @DATA
MOV DS, AX
MOV AH, 9
LEA DX, msg2
INT 21H
END:
MOV AH, 4CH
INT 21H
MAIN ENDP
END MAIN

29)

```
.MODEL SMALL
.STACK 100H

.CODE
MAIN PROC

    MOV AX, 1010H  ; taking 1010H as example
    MOV BL, AL
    MOV BH, AH
    ADD BL,BH      ; Adding BL, BH
    JNZ L1         ; Jump if not zero
    MOV DL,BL      ; copping in DL
    JMP END        ; Jump to END

    L1: MOV DH,BL  ; copping in DH
    END: MOV AH,4CH
         INT 21H
MAIN ENDP
END MAIN
```

30]
```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC

    MOV CX, 1010H
    MOV BL, CL
    MOV BH, CH
    ADD BL, BH

    JNP ODD    ; Jump if not parity
    MOV DL, BL ; copping in DL
    JMP END
    ODD: MOV DH, BL
    END: MOV AH, 4CH
         INT 21H
MAIN ENDP
END MAIN
```

31]
```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC

    ADD CX, BX
    JMP END
    JC L1      ; Jump if carry
    MOV CX, BX ; copping in CX
    JMP END
    L1: MOV DX, BX   ;copping in DX
    END: MOV AH, 4CH
         INT 21H
MAIN ENDP
END MAIN
```

32| .MODEL SMALL
.STACK 100H
.CODE
MAIN PROC

ADD BX, CX
JO L1 ; if sign overflow Jump to L1
MOV CX, BX
JMP END
L1 : MOV DX, BX
END: MOV AH, 4CH
INT 21H
MAIN ENDP
END MAIN

33] .MODEL SMALL
.STACK 100H
.DATA
msg1 DB 'Auxilary Carry $'
msg2 DB 'No Auxilary Carry $'
.CODE
MAIN PROC

```
MOV DL,AL
MOV CL,AH
SHL DL,4        ; Shifting Left CL by 4
SHL CL,4        ; Shifting Left DL by 4
ADD CL,DL
JC L1           ; Jump if carry exist
MOV AX,@DATA
MOV DS,AX
MOV AH,9
LEA DX,msg2 ; Displaying Massage
INT 21H
JMP END         ; Jump to END
L1: MOV AX,@DATA
    MOV DS,AX
    MOV AH,9
    LEA DX,msg1 ; Displaying Masage
    INT 21H
END: MOV AH,4CH
     INT 21H
MAIN ENDP
END MAIN
```

34]

```
.MODEL SMALL
.STACK 100H
.CODE
MAIN PROC
    MOV CX,0000H
    MOV DS, CX
    MOV BX, 0000H
    MOV AL, BYTE PTR DS:[BX] ; load data from 00000H
    MOV BX, 0001H
    MOV AH, BYTE PTR DS:[BX] ; load data from
    MOV BX, 0002H
    MOV CL, BYTE PTR DS:[BX]

    CMP AL, AH
    JG Greater1  ; Jump if greater first one.
    CMP AH, CL
    JG Greater2  ; Jump if second one is greater
    MOV DL, CL   ; if third is greater copping
    JMP END
    Greater1: CMP AL, CL
              JG Greater3 : first is greater
              MOV DL, CL  ; third is greater
              JMP END
    Greater3: MOV DL, AL
              JMP END

    Greater2: MOV DL, AH

    END:      MOV AH, 4CH
              INT 21H
MAIN ENDP
END MAIN
```

35)

```
.MODEL SMALL
.STACK 100H

.CODE
MAIN PROC
    MOV CX, 0000H
    MOV DS, CX
    MOV BX, 0000H
    MOV AL, BYTE PTR DS:[BX]
    MOV BX, 0001H
    MOV AH, BYTE PTR DS:[BX]
    MOV BX, 0002H
    MOV CL, ~~0002H~~ BYTE PTR DS:[BX]

    CMP AL, AH
    JL  Smaller1  ; Jump if less first one
    CMP AH, CL
    JL  Smallest2 ; Jump if less ~~first one~~ second one
    MOV DL, CL
    JMP END
Smaller1: CMP AL, CL
          JL Smallest3 ; first is smallest
          MOV DL, CL
          JMP END

Smallest3: MOV DL, AL
           JMP END     ; second is smallest

Smallest2: MOV DL, AH

END: .MOV AH, 4CH
     INT 21H

END MAIND
END MAIN
```

3. 36]
```
.MODEL SMALL
.STACK 100H
.Data
    P DB 'Enter a number between 0-9: $'
    C DB ?
.CODE
MAIN PROC
    MOV AX,@DATA
    MOV DS,AX
    MOV AH,9    ; Output to prompt
    LEA DX, P
    INT 21H

    MOV AH,1    ; take character input
    INT 21H
    MOV C,AL
    AND C,0FH   ; converts to numerical value

    MOV AH,2
    MOV DL,0AH
    INT 21H
    MOV DL,0DH
    INT 21H     ; line break input

    MOV AH,2
    MOV DL,C
    INT 21H

    MOV AH,4CH
    INT 21H
MAIN ENDP
END MAIN
```