North South University

Department of Electrical & Computer Engineering

Course Assignment(ASM Codes)

Submitted By:

Name: Fahima Noor
ID: 1912052042

Course: Microprocessor Interfacing and Embedded System
Course Code: CSE 331
Section: 03
Date Of Submission: 20 April 2021

Submitted To:
Rishad Arfin (Rsf)

Fahima Noor
1912052042

1.

```
L1: NOP
L2: MOV AX, F000H  ; loading the last address.
L3: MOV DS, AX
L4: MOV BX, FFFFH
L5: MOV AL,BYTE PTR DS:[BX] ; loading data in AL

L6: MOV CX, 0000H  : loading the first address
L7: MOV DS, CX
L8: MOV BX, 0000H
L9: MOV AH, BYTE PTR DS:[BX] ; loading data in AH
L10: ADD AL, AH      ; adding AL with AH
L11: MOV CX, F000H   ; Loading the last even
                       address
L12: MOV DS, CX
L13; MOV BX, FFFEH
L14; MOV BYTE PTR DS:[BX],AL ; storing in the last
                               address.
L15: HLT ; END
```

Fahima Noor
19120520 42

2.

```
L1: NOP
L2: IN AL, 00H  ; Read from fixed port first address
L3: MOV DL, AL
L4: IN AL, 01H  ; Read from fixed port second address
L5: ADD AL, DL
L6: MOV CX, F000H  ; Loading the last address of 8086
L7: MOV DS, CX
L8: MOV BX, FFFFH
L9: MOV BYTE PTR DS:[BX], AL  ; storing in the last memory address
L10: HLT  ; END
```

3.

```
L1: NOP
L2: MOV CX, F000H  ; loading last address
L3: MOV DS, CX
L4: MOV Bx, FFFFH
L5: MOV AL, BYTE PTR DS:[BX]  ; loading data in AL
L6: MOV CX, 0000H  ; loading first address
L7: MOV DS, CX
L8: MOV Bx, 0000H
```

```
L9:  MOV AH, BYTE PTR DS:[BX] ; loading data in AH
L10: ADD AL, AH
L11: OUT FEH, AL ; storing in last even memory location of fixed port address
L12: HLT ; END
```

4.

```
L1:  NOP
L2:  MOV DX, 0000H ; first address of variable port
L3:  IN AL, DX ; read from the address
L4:  MOV BL, AL ; store data from AL to BL
L5:  MOV DX, FFFFH ; last address of variable port
L6:  IN AL, DX ; read from the address
L7:  ADD AL, BL
L8:  OUT FEH, AL ; store in the last even address of fixed port.
L9:  HLT ; END
```

Fahima Noor
1912052042

5.

```
L1: NOP
L2: MOV CX, 0000H
L3: MOV DS, CX
L4: MOV BX, 0000H
L5: MOV AL, BYTE PTR DS:[BX]
L6: MOV BX, 0001H
L7: MOV AH, BYTE PTR DS:[BX]
L8: ADD AL, AH
L9: MOV BX, 0002H
L10: MOV AH, BYTE PTR DS:[BX]
L11: ADD AL, AH
L12: MOV DX, 3200H ; variable port address
L13: OUT DX, AL ; store the result in variable port address
L14: HLT ; END
```

Fahima Noor
1912052042
Pg-05

6.

```
L1: NOP
L2: MOV CX, FDOOH   ; Loading the last address
L3: MOV DS, CX
L4: MOV BX, FFFFH
L5: MOV AL, BYTE PTR DS:[BS] ; loading data in AL
L6: MOV CX, 0000H   ; loading the first address
L7: MOV DS, CX
L8: MOV BX, 0000H
L9: MOV AH, BYTE PTR DS:[BX] ; loading data in AH
L10: ADD AL, AH
L11: CMP AL, 05
L12: JG LOAD   ; will go to load label if AL greater than 5
L13: JMP BREAK
```

```
L14: LOAD:
L15: MOV CX, F000H ; Loading the last even memory location
L16: MOV DS, CX
L17: MOV BX, FFFEH
L18: MOV BYTE PTR DS:[BX], AL ; Storing in the last even memory
L19: BREAK
L20: HLT ; END
```

7.

```
L1: NOP
L2: MOV CX, F000H ; loading the last address
L3: MOV DS, CX
L4: MOV BX, FFFFH
L5: MOV AL, BYTE PTR DS:[BX] ; Loading data in AL
L6: MOV CX, 0000H
L7: MOV DS, CX
L8: MOV BX, 0000H
L9: MOV AH, BYTE PTR DS:[BX] ; Loading data in AH
L10: SUB AL, AH
```

Fahima Noor
19120052042


```
L11: CMP  AL, 0
L12: JG   POSITIVE
L13: OUT  00H, AL  ; Value of AL is written in 00H
                     (direct addressing for fixed port)
L14: JMP BREAK
L15: POSITIVE:
L16: MOV DX, FFFFH ; DX = FFFFH, store last location
                     of variable port addre
L17: OUT DX, AL    ; Value of AL is written FFFFH
                     (indirect addressing)
L18: BREAK:
L19: HLT ; END
```

Fahima Noor
19120052042

8.

```
L1:  NOP
L2:  MOV CX, 0000H
L3:  MOV DS, CX
L4:  MOV BX, 0000H
L5:  MOV AL, BYTE PTR DS:[BX] ; Loading data in AL
L6:  MOV BX, 0001H  ; loading second address
L7:  MOV AH, BYTE PTR DS:[BX] ; loading data in AH
L8:  CMP AL, AH
L9:  JG FIRST_ADDRESS ; go to this label if AL is
                        greater
L10: JL SECOND_ADDRESS ; go to this label if AH is
                         greater
L11: FIRST_ADDRESS:
L12: OUT 00H, AL  ; Value of AL is written in 00H
                    address
L13: JMP BREAK
L14: SECOND ADDRESS:
L15: MOV AL, AH ; Port address is only accessible with
                  AL/AX. So data in AH need to be
                  stored in AL
L16: OUT 00H, AL
L17: BREAK  ; Value of AL is written in 00H
L18: HLT    ; END
```

Fahima Noor
1912052042

9.

```
L1: NOP
L2: MOV AL, 1000000b ; suppose data is 10000000b
L3: CMP AL, 0
L4: JE ZERO ; if AL=0 then would jump to zero label
L5: JG POSITIVE ; if AL= +ve, jump to positive label
L6: JL NEGATIVE ; if AL=-ve, jump to negative label
L7: ZERO:
    ; instructions under this label will be executed
      if AL=0
L8: JMP BREAK
L9: POSITIVE:
    ; instructions under this label will be executed
      if AL = +ve
L10: JMP BREAK
L11: NEGATIVE:
    ; instructions under this label will be
      executed if AL=-ve
L12: BREAK:
L13: HLT ; END
```

10

```
L1: NOP
L2: MOV CL, 00000011b ; suppose 3 is taken as data
L3: MOV AL, 00000001b ; only odd number have LSB=1.
L4: AND CL, AL
L5: CMP CL, 1 ; check to see CL=1 or not
L6: JE ODD
L7: JNE EVEN

L8: ODD:
    ; intructions under this label will be executed if data is odd
L9: JMP BREAK
L10: EVEN:
    ; instructions under this label will be executed if data is even.
L11: BREAK:
L12: HLT ; END
```

11.

```
L1 : NOP
L2 : MOV AL, 11011011b ; suppose AL reads this data
L3 : MOV AH, AL ; string copied so that this string can
                  be reversered and compared with
                  original.
L4 : MOV CL, 0    ; counter
L5 : REVERSE :
L6 : CMP CL, 8
L7 : JE  EXIT_REVERSE
L8 : SHL AL, 1    ; Left shift bit
L9 : RCR BL, 1    ; right rotate bits with carry
L10 : INC CL      ; increment counter
L11 : JMP REVERSE
L12 : EXIT_REVERSE :
L13 : MOV AL, BL ; move the reversed pattern
L14 : CMP AH, AL
L15 : JE  PALINDROME
L16 : JNE NOT_PALINDROME
L17 : PALINDROME:
      ; executes this label if string palindrome
L18 : JMP BREAK
L19 : NOT_PALINDROME
      ; executes this label if string is not palindrome.
L20 : BREAK:
L21 : HLT
```

Fahima Noor
1912052042

12.

```
L1: NOP
L2: MOV CX, F000H    ; loading the last address
L3: MOV DS, CX
L4: MOV BX, FFFFH
L5: MOV BL, BYTE PTR DS:[BX] ; Loading data in BL
L6: MOV DL, BL ; copies data in BL into DL
L7: MOV AL, 00000001b ; needed for AND operation
L8: MOV CX, 8 ; loop counter
L9: MOV AH, 0 ; will keep track of number of 1s
L10: COMPARE:
L11: AND DL, AL ; Only LSB can be 0 or 1, other bits
                  will always result in zero
L12: CMP DL, 0 ; any value present mean 1 was
                 present as its bit for that particular
                 bit location in BL register
L13: JG COUNT_ONES
L14: SHIFT_DATA:
L15: SHR BL, 1 ; logical shift right
L16: MOV DL, BL
L17: LOOP COMPARE
L18: JMP BREAK
```

```
L19: COUNT_ONES:
L20: INC AH ; AH will be increment everytime 1
             is found in BL register.
L21: JMP SHIFT_DATA
L22: BREAK:
L23: HLT ; END
```

13.

```
L1 : NOP
L2 : MOV CX, F000H ; loading last address
L3 : MOV DS, CX
L4 : MOV BX, FFFFH
L5 : MOV BL, BYTE PTR DS:[BX] ; Loading data in BL
L6 : MOV DL, BL ; DL now has same data as BL
L7 : MOV AL, 00000001b
L8 :
     MOV CX, 8 ; Counter for loop
L9 : MOV AH, 0 ; counts number of 1s
L10: COMPARE
L11: AND DL, AL ; LSB only has the chance of
                  being 1
L12: CMP DL, 0 ; any value present in DL means
L13: JG COUNT_ONES 1 was present.
```

Fahima Noor
1912052042


```
L14 : SHIFT_DATA :
L15 : SHR BL, 1   ; shift bits to right
L16 : MOV DL, BL  ; move data BL into DL as well
L17 : LOOP COMPARE

L18 : COUNT_ZEROS:
L19 : MOV AL, 08
L20 : SUB AL, AH ; the number of zeros in bl
                   register will be stored in AL
L21 : JMP BREAK

L22 : COUNT_ONES:
L23 : INC AH      ; increaments to indicate the
L24 : JMP SHIFT_DATA  presence of 1 in data

L25 : BREAK:
L26 : HLT ; END
```

Fahima Noor
1912052042
Pg-15

14

```
L1: NOP
L2: MOV AL, 04  ; suppose data is 01005
L3: SAL AL, 1   ; shift arithematic Left the data in AL by
                  1 bit. Equivalent to Multiply by 2
L4: SAL AL,1    ; Multiply by 4 (2^2)
L5: SAL AL, 1 ; Multiply by 8 (2^3)
L6: SAL AL, 1 ; Multiply by 16 (2^4)
L7: HLT ; END
```

15.

```
L1: NOP
L2: MOV AX, 32
L3: SHR AX, 1   ; divide by 2
L4: SHR AX, 1   ; divide by 4 (2^2)
L5: SHR AX, 1   ; divide by 8 (2^3)
L6: SHR AX,1    ; divide by 16 (2^4)
L7: HLT ; END
```

Fahima Noor
19120520 42
Pg-16

16.

```
L1:  NOP
L2:  MOL BL, 00001111b
L3:  AND AL,BL ; removes the higher nibble in AL
L4:  CMP AL, 09
L5:  JG GREATER
L6:  JL LOWER
L7:  JMP BREAK
L8:  GREATER:
       ; will exeute instructions under this label
         if lower nibble of AL greater than 9
L9:  JMP BREAK
L10: LOWER:
       ; will execute instructions under this label
         if lower nibble of AL is lesser than 9
L11: BREAK:
L12:  HLT ; END
```

17.

```
L1:  NOP
L2:  SHR CH, 4  ; Shifts the higher nibble to
                  right (in the lower nibble)
L3:  CMP CH, 9
L4:  JG GREATER
L5:  JL LOWER
L6:  JMP BREAK

L7:  GREATER
       ; instructions under this label would be
         executed if higher nibble greater than 9
L8:  JMP BREAK
L9:  LOWER
       ; executed if higher nibble less than 9

L10: BREAK:
L11:   HLT ; END
```

18.

```
L1   NOP
L2   MOV DL, 10000001b
L3   OR BL, DL ; sets the MSB/LSB to 1
L4   HLT ; END
```

Fahima Noor
1912052042
Pg-18

19.

```
L1 : NOP
L2: MOV DL, 10000000b
L3: XOR BL, DL ; change sign bit
L4: HLT ; END
```

20.

```
L1; NOP
L2; MOV DL, 01010101b
L3: XOR AL, DL ; change the even index bits
L4: HLT;END
```

21.

```
L1 : NOP
L2: MOV DL, 10101010b
L3:
    XOR BL, DL
L4:
    HLT ; END
```

22.

```
L1 : NOP
L2 : MOV CL,0
L3 : REVERSE
L4 : CMP CL, 8
L5 : JE EXIT-REVERSE
L6 :
    SHL AL, 1          ; left shift bit
L7 : RCR BL, 1         ; right rotate bits with carry
L8 : INC CL            ; increment counter
```

Fahima Noor
1912052042
Pg- 19

```
L9:   JMP REVERSE
L10:  EXIT_REVERSE:
L11:  MOV AL, BL  ; move the reversed data in AL
L12:  HLT ; END
```

Q3.

```
L1:  NOP
L2:  XOR CL, CL  ; clears the bit as same bit XOR operation
L3:  HLT ; END
```

24

```
L1:  NOP
L2:  MOV CL, 10101010B
L3:  AND AL, CL  ; clear even bits in AL
L4:  HLT ; END
```

25

```
L1:  NOP
L2:  MOP
L3:  MOV AL, 10101010B
L4:  OR CL, AL  ; odd index in CL will be 1
L5:  HLT ; END
```

26:

```
L1 :  NOP
L2 :  MOV  DL, AL
L3 :  SHR  DL, 4    ; right shift 4 bit
L4 :  SHL  AL, 4    ; left shift 4 bits
L5 :  OR   AL, DL   ; nibbles exchanged
L6 :  HLT  ; END
```

27.

```
L1 :  NOP
L2 :  MOV  DL, 00001111b
L3 :  AND  DL, AL   ; DL taking lower nibble only
L4 :  SHR  AL, 4    ; right shift 4 bits (higher
                      nibble is shited to lower nibble)
L5 :  CMP  DL, AL
L6 :  JE   EQUAL
L7 :  JMP  BREAK

L8 :  EQUAL :
          ; execute if AL lower nibble equal higher
            nibble.
L9 :  BREAK
L10:  HLT  ; END
```

Fahima Noor
1912052042

28

```
L1 : NOP
L2 : ADD  AL, DL

L3 : JZ  EQUAL ; jump to EQUAL LABEL IF RESULT ZERO
L4 : MOV  DH, AL   ; if result is not zero
L5 : JMP  BREAK

L6 : EQUAL :
L7 : MOV  DL, AL  ; if result is zero
L8 : BREAK :
L9 : HLT ; END
```

29

```
L1 :  NOP
L2 :  ADD  AL, DL
L3 : JP  EVEN
L4 : MOV  DH, AL
L5 : JMP  BREAK
L6 : EVEN
L7 : MOV  DL, AL
L8 : BREAK
L9 : HLT  ; END
```

30.

```
L1 : NOP
L2: MOV AX, 0001H
L3: MOV DX, 0002H
L4 : ADD AX, DX
L5: JNS CHECK_OVERFLOW ; will jump if result is unsigned (SF=0)
L6: CHECK_OVERFLOW
L7: JO UNSIGNED_OVERFLOW ; jump if OF=1
L8: MOV CX, AX
L9: JMP BREAK
L10: UNSIGNED_OVERFLOW :
L11: MOV DX, AX
L12: BREAK :
L13: HLT ; END
```

31

```
L1:NOP
L2:MOV AX , 0001H
L3:MOV DX, 0002H
L4:ADD AX, DX
L5: JS CHECK_OVERFLOW ; jump if SF = 1
L6: CHECK_OVERFLOW
L7: JO UNSIGNED_OVERFLOW : (jump if OF = 1)
```

```
L8:  MOV  CX , AX  ;
L9:  JMP  BREAK
L10: UNSIGNED - OVERFLOW :
L11: MOV  DX , AX  ; stores result in DX if signed
                     overflow occurs.
L12: BREAK :
L13: HLT ; END
```

32

```
L1: NOP
L2: MOV  AL , 10101111b
L3: MOV  DL , 11101111b
L4: MOV  BL , 00001111b
L5: ADD  AL , BL ; remove higher nibble of AL
L6: ADD  DL , BL ; remove higher nibble of DL
L7: ADD  AL , DL ; store addition result in AL
L8: SHL  AL , 3  ; 5th bit shifted to 8th place
L9: JS HAS-AUXILLARY CARRY ; jump if SF = 1, This
                             can only happen if
                             Auxillary Flag occurs
L10: JMP BREAK
L11: HAS-AUXILLARY-CARRY :
     ; instructions under this label will be
       executed if result have auxillary carry.
```

Fahima Noor
1912052042

L12: BREAK;

L13: HLT ; END

33:

```
L1 :  NOP
L2:   MOV CX, 0000H ; Loading the first address
L3:   MOV DS, CX
L4:   MOV BX, 0000H
L5:   MOV AL, BYTE PTR DS:[BX] ; loading data in AL
L6:   INC BX ; Loading the second address
L7:   MOV AH, BYTE DS:[BX] ; Loading data in AH
L8 :  INC BX ; Loading third address
L9:   CMP AL, AH
L10:  JL X1   ; jump if AL smaller than AH
L11:  JG X2   ; jump if AL greater than AH
L12:  X1 ;
L13:  MOV AH, BYTE PTR DS:[BX] ; Loading data from third address
L14:  CMP AL, AH
L15:  JL X3
L16:  JG X4
L17:  JMP BREAK
```

```
L18:  X2:
L19:  MOV AL, BYTE PTR DS:[BX] ; loading data from third address
L20: CMP AH, AL
L21: JL X4
L22: JG X3
L23: JMP BREAK

L24: X3:
L25: MOV DL, AH    ; stores highest value into DL
L26: JMP BREAK
L27: X4:
L28: MOV DL, AL    ; stores highest value into DL
L29: JMP BREAK
L30: BREAK:
L31: HLT ; END
```

Fahima Noor
19120520 42

34

```
L1:  NOP
L2:  MOV CX, 0000H ; loading first address
L3:  MOV DS, CX
L4:  MOV BX, 0000H
L5:  MOV AL, BYTE PTR DS:[BX] ; loading data in AL
L6:  INC BX ; loading 2nd address
L7:  MOV AH, BYTE PTR DS:[BX]
L8:  INC BX ; loading 3rd address
L9:  CMP AL, AH
L10: JL X1
L11: JG X2

L12: X1:
L13: MOV AH, BYTE PTR DS:[BX]
L14: CMP AL, AH
L15: JL X3
L16: JG X4
L17: JMP BREAK
```

Fahima Noor
1912052042


```
L18: X2 :
L19: MOV AL, BYTE PTR DS:[BX]
L20: CMP AH, AL
L21: JL X4
L22: JG X3
L23: JMP BREAK

L24: X3 :
L25: MOV DL, AL
L26: JMP BREAK

L27: X4 :
L28: MOV DL, AH
L29: BREAK
L30: HLT ; END
```

Fahima Noor
1912052042

35

```
L1:  NOP
L2:  VALUE  DB 10 DUP (?)
L3:  MOV BL, 48 ; ASCII of 0 is 48
L4:  MOV CL, 48 ; Acts as counter
L5:  LEA SI, VALUE ; uninitialized array being
                     pointed by si
L6:  LOOPING;
L7:  CMP CL, 58  ; will loop till CL reaches 58
L8:  JE BREAK
L9:  MOV AL, CL
L10: SUB AL, BL  ; difference between ASCII
                   value will produce numeric
                   value.
L11: MOV [SI], AL
L12: INC SI
L13: INC CL
L14: JMP LOOPING
L15: BREAK;
L16: HLT ;END
```