Tutorial 1

Aarati Mahato
BEIT 4th Sem Sec. 'B'
Roll No. '1'

PAGE NO.:
DATE:

Microprocessor Tutorial

Assignment on 8085 microprocessor programming:

Q.1) WAP to find the smallest number among 10 numbers stored from memory location 4050H.

Ans:-

```
      LXI H, 4050H
      MVI C, 09H
L2:   CMP M
      JC L1
      MOV A,M
L1:   INX H
      DCR C
      JNZ L2
      MOV M,A
      HLT
```

Q.2) WAP to change the D5 bit of ten numbers stored from memory location 4000H if they are greater than 80H.

Ans:-

```
      LXI H, 4000H      ; Starting memory location pointer
      MVI C, 0AH        ; Counter for 10 data
L2:   MOV A,M
      CPI 81H
      JC L1
      XRI 20H           ; 00100000 Complements bit D5
      MOV M,A
L1:   INX H
      DCR C
      JNZ L2
```

PAGE NO.: _____
DATE:

Q.3) WAP to swap the D3 and D5 bits of 10 numbers stored from memory location 5000H if any number is greater than 70H and less than A0H otherwise set the D3 bit and reset the D5 bit.

Ans:-

```
        LXI H, 5000H
        MOV A,M
        CPI 71H
        JC  L1
        CPI A0H
        JNC L1
// Swapping bit D3 and D5
        ANI 20H
        RRC
        RRC
        MOV D,A
        MOV A,M
        ANI 08H
        RLC
        RLC
        MOV C,A
        MOV A,M
        ANI D7H
        ORA C
        ORA D
        JMP LAST
// Setting D3 and resetting D5
L1:     ORI 08H  ; Sets D3
        ORI DFH  ; Resets D5
LAST:   MOV M,A
        HLT
```

PAGE NO.: _____
DATE:

Q.4) WAP to transfer the 8 bit number from location 4000H to 5000H if D5 bit is 1 and D3 bit is 0 otherwise transfer data by changing D2 and D6 bit.

Ans:-

```
~~LXI B, 5000H~~
LXI H, 4000H ~~→ MVI D, 0A~~
MOV A,M
RLC
RLC
RLC          ; D5 = Cy
JC L1
MOV A,M
XRI 44H
STA 5000H
HLT
```

```
LXI H, 4000H
MOV A,M
RLC
RLC
RLC          ; D5 = Cy
JNC L1
RLC
RLC          ; D3 = Cy
MOV A,M
JNC L2
L1: MOV A,M
XRI 44H
L2: STA 5000H
HLT
```

PAGE NO.:
DATE:

Q.5) WAP a program to copy and convert the lower case ASCII code to upper case ASCII code from memory location 3050H to 4050H if any. Otherwise copy the code as they are. Assume that they are 50 data in the source memory.

→ Solution:

```
        LXI H, 3050H  ; source
        MVI B, 4050H  ; destination
        MVI D, 32H

Loop:   MOV A,M       ; A ← [HL]
        CPI 61H       ; Compare with ASCII code
                        of a = 97.
        JC TRANSFER   ; data less than 61h
        CPI 7BH       ; Compare with 123
        JNC TRANSFER  ; data more than equal to 7
        SUI 20H       ; convert to uppercase.
TRANSFER: STAX B
        INX H
        INX B
        DCR D
        JNZ Loop
        HLT
```

Q.6) WAP to transfer data from one table to another table if the number one in the data Ps greater than 4 otherwise copy 00H to next table.

→ Ans:-

```
        LXI H, 8000H
        LXI B, 9000H
        MVI D, 0AH
```

```
Loop: MOV A,M
      CPI 4H          ; Compare with 4H
      JNC Transfer
      MOV A,00H       ; Put 00h in accumulator
Transfer: STAX B
      INX H
      INR B
      DCR D
      JNZ Loop
      HLT
```

Q.7) WAP to add 10 data stored from memory location 3050H and stored 16 bit result at the end of table.

→ Ans:-

```
      MVI A,00H  ; clear accumulator for lower
                   byte sum
      MVI B,00H  ; clear register B for higher byte
                   sum.
      MVI C,0AH  ; set up C as a counter
      LXI H,3050H ; set up HL as memory pointer
UP:   ADD M
      JNC Down       ; If no carry, don't increase reg.B
      INR B          ; If carry, save carry bit point
                       to next memory location
Down: INX H
      DCR C
      JNZ UP
      MOV M,A        ; transfer sum (lower byte), to memory
      INX H
      MOV M,B        ; transfer higher byte sum
      HLT
```

PAGE NO.: _____
DATE:

Q.8. WAP to add all the positive number stored from memory location 3050 to 3059 and stored 16 bit result at the end of the table.

→ Ans:-

```
LXI H, 3050H  ; setup HL as memory pointer
MVI D, 00H    ; for lower byte sum
MVI B, 00H    ; for higher byte sum
MVI C, 0AH    ; for counter

UP: MOV A, M
    RAL           ; Rotate left with carry
    JC Dowon      ; Negative number
    RAR           ; Obtain original value
    ADD D
    MOV D, A
    JNC Dowon
    INR B
Dowon: INX H
    DCR C
    JNZ UP
    MOV A, D
    MOV M, A      ; Save lower byte sum
    INX H
    MOV M, B      ; Save higher byte sum.
    HLT
```

Q.9) There are two tables of the data stored at memory location 3050H and 3060H, each of which have 10 data. WAP to store data from first table to third table starting from 3070

PAGE NO.: _____
DATE:

if corresponding data in first table is greater than second table else store FBH in the third table.

→ Ans:-

```
    LXI H, 3050H   ; memory pointer for first table
    LXI B, 3060H   ; memory pointer for second table
    LXI D, 3070H   ; memory pointer for third table
    LXI SP, 3000H  ; Stack Pointer
    MVI A, 0AH     ; Counter

Loop: PUSH  PSW
      LDAX  B          ; [A] ← [[BC]]
      CMP   M          ; [A] ← [A] - [[HL]]
      JC    L1         ; Data of 1st table greater
      MOV   A, FBH     ; [A] ← FBH
L1:   STAX  D          ; Store data in third table.
      INX H
      INX B
      INX D
      POP PSW
      DCR A            ; Decrease counter
      JNZ Loop
      HLT
```

Q.10) There are two tables of the data stored at memory location 3050H and 3060H, each of which have 10 data. Check ~~Data~~ whether any two corresponding data are equal. If they are equal turn on bit D_7

PAGE NO.: _____
DATE:

Ans:-

```
LXI H, 3050H
LXI B, 3060H
MVI C, 0AH    ; counter for 10 data

Loop: LDAX B
      CMP M    ; Compare with corresponding data
      JNZ L1   ; Data not equal
      ORI 80H  ; Turn on bit D7
L1 :  INX H
      INX B
      DCR C
      JNZ Loop
      HLT
```

Q.11) WAP to transfer 10 data with even parity from table starting from 3050H ~~or~~ to table 3060H else transfer the data by setting D_7 bit and resetting D_3 bit.

→ Ans:-

```
LXI H, 3050H
LXI B, 3060H
MVI C, 0AH    ; Counter for 10 data

MOV A, M
JPE L1        ; Jump on even parity
; Setting bit D7 and resetting bit D3.
```

Q.12) WAP to convert 10 binary numbers to BCD stored from memory location 40AAH and store to the memory location 41AAH if the result less than 100.

→ Ans:-

```
        LXI H, 40AAH
        LXI D, 41AAH
        PUSH D          ; Save contents of DE pair
NEXT:
        MOV A,M
        MVI B,64H
        CALL BIN-BCD    ; Call subroutine to convert
                          binary to BCD
        POP D
        CMP CPI 64H
        JNC Last
        POP D
        STAX D
Last:   INX H
        INX D
        PUSH D
        MVI A, B4H
        CMP L
        JNZ Next
        HLT

BIN-BCD:
        MVI B, 0AH
        MVI C, 00H
L1:     INR C
        SUB B ; subtract each time 0AH from
                accumulator data
```

```
        INC  L1
        DCR  C
        ADD  B
        MOV  D,A
        MOV  A,C
        RLC
        RLC
        RLC
        ADD  D ;  add two nibbles to get final
                  BCD number
        RET
```

Q.13) WAP to convert 10 BCD stored from memory location 3070H to 3080H to binary numbers.

→ Ans:-

```
        LXI  H, 3070H
        MVI  C, 0AH  ;  counter for 10 data
Loop:   MOV  A,M
        CALL BCD-BIN
        ~~START~~
        MOV  M,A
        INX  H
        ~~DCR~~ DCR C
        JNZ  Loop
        HLT

BCD-BIN:
        MVI  E, 0AH
        MOV  B, A
        ANI  0FH; ——→ BCD1
        MOV  C, A; ——→ BCD2
```

```
MOV A,B
ANI F0H ; -> BCD 2
RRC
RRC
RRC
RRC
MOV D,A
MVI A,00H
L1: ADD D
DCR E
JNZ L1
MOV D,A
MOV A,C
ADD D ; Binary = BCD2 x 10 + BCD1
RET
```

Q.14) Data is stored from 4050H. Insert 5 data at 4055H taking from 4040H but do not loss the previous content.

→ Ans:-

```
LXI B,4050H ; address where another data to be stored
LXI H,4040H ; source memory address
LXI D,4060H
PUSH D ; Save content of DE pair
L1: LDAX B
STAX D
MOV A,M
STAX B
INX B
```

```
        INX  D
        INX  H
        MVI  A,05H
        CMP  E
        JNZ  L1
        POP  D
L2:     LDAX D
        STAX B
        INX  B
        INX  D
        MVI  A,05H
        CMP  A,05H
        JNZ  L2
        HLT
```

; Save the contents of 4056 to 405AH to location 4060H to 4064H and also store the data from 4040H to 4056H to 405AH.

; Save the contents of 4060H - 4064H to memory location 405BH to 405FH

Q.15) Data is stored from 4050H to 4060H copy to another desired location in reverse order.

→ Ans:-

```
        LXI  H, 4060H   ; Source memory location pointer
        LXI  B, 4080H   ; Destination memory location pointer.
        MVI  D, 11H
Loop:
        MOV  A, M
        STAX B
        DCX  H
        INX  B
        DCR  D
        JNZ  Loop
        HLT
```

PAGE NO.:
DATE:

Q.16) In above program sort the data in ascending order and store to the next table starting from 4070H.

→ Ans :-

```
MVI D, 09H
LXI H, 4050H
LXI B, 4070H
MVI D, 09H
MV C, 09H
MOV A, M
INX H
CMP M
JC L1
MOV B, M
```

```
LXI B, 4070H ; Destination
MVI D, 09H
L3: LXI H, 4050H
MVI E, 09H
L2: MOV A, M
INX H
CMP M
JC L1
STAX B
INX H
INX B
L1: DCR E
JNZ L2
DCR D
JNZ L3
HLT
```

PAGE NO.: _____
DATE:

Q.17) WAP to multiply two 8 bit numbers by successive addition.

→ Ans:-

```
LDA  2000H    ; Get the number
MOV  E, A     ; save to register E
MVI  D, 00H   ; Get the first number in D
                register
LDA  2001H    ; Get the number
MOV  C, A     ; Initialize counter
LXI  H, 0000H ; Result = 0
BACK: DAD  D  ; Result = result + first numb
      DCR  C
      JNZ  BACK
      SHLD 2800H ; store result
      HLT.
```

Q.18) WAP to multiply two 8 bit number by shift left and add (multiplication by shifting).

→ Ans:-

```
LDA  8000H ; Get first number
MOV  E, A
MVI  D, 00H
LDA  8001H ; Get second number
MOV  C, A  ; Initialize counter
LXI  H, 0000H
RAL
JNC  SKIP
DAD  D
SKIP: DCR  C
      JNZ  MULT
```

PAGE NO.:
DATE:

Ans:-

```
        LXI H, 2200H
        MOV E, M        ; Get multiplicand
        MVI D, 00H
        INX H
        MOV A, M        ; Get multiplier
        LXI H, 0000H    ; Product = 0
        MVI B, A##      ; Initialize counter
MULT:
        DAD H
        RAL
        JNC SKIP
        DAD D
SKIP:
        DCR B
        JNZ MULT
        SHLD 2300H
        HLT
```

Q. 19) WAP to implement hex up counter.

→ Ans:-

```
        MVI B, FFH
NEXT:   DCR B           - 4T
        MVI C, COUNT    - 7T
DELAY:  DCR C
        JNZ DELAY
        MOV A, B        - 4T
        OUT PORT#       - 10T
        JMP NEXT        - 10T
```

$T_0 = 35T$

The register C is the time delay register which is loaded by a value COUNT to produce a time delay of 1 ms.

To find the value of COUNT:-

$$T_D = T_L + T_o$$

where

T_D = Time Delay

T_L = Time delay inside loop

T_o = Time delay outside loop

The Delay loop includes two instructions DCR C (4T) & JNZ (10 T)

So,

$$T_L = 14 * \text{clock period} * \text{count}$$
$$= 14 * 0.5 \times 10^{-6} \times \text{count}$$
$$\Rightarrow 7 \times 10^{-6} \times \text{count}$$

Delay outside loop.

T_o = 35 * clock period

$\Rightarrow 17.5\,\mu s$

So,

$$1\,ms = (17.5 + 7 * \text{Count})\,\mu s$$

$\therefore$ Count = $(140)_{10}$

PAGE NO.: _____
DATE:

Q.20) WAP to find the factorial of a numbers.

→ Ans:-

```
        MVI  D, 04H    ; 4 is the number
                       ; whose factorial is desired
        MOV  B,D
        DCR  D
Loop2:  MOV  E,D
        XRA  A         ; [A] ← [A] ∀ [A]
                       ; clearing accumulator to store result
Loop:   ADD  B
        DCR  E
        JNZ  Loop
        MOV  B,A       ; result is in reg. A or B
        DCR  D
        JNZ  Loop2
        HLT.
```

Q.21) WAP to find the square root of a numbers.

→ Ans:-

```
        LDA  4200H     ; Get the number
        MOV  B,A
        MVI  C,02H     ; divisor
        CALL DIV       ; Call subroutine to get intial value(X)
REP:    MOV  E,D
        MOV  A,B       ; Get dividend (Y) in A
        MOV  C,D       ; Get divisor(X) in C.
        CALL DIV       ; Call subroutine to get initial
                       value (Y/X) in D reg.
        MOV  A,D
        ADD  E         ; Get (Y/X + X)
        MVI  C, 02H    ; Divisor in C reg.
```

```
CALL DIV  ; Call subroutine to get (Y/X+X)/2
MOV A,E
CMP D     ; Compare X with Accumulator conte
JNZ REP   ; If [A] is not equal to X, then repea
STA 4201H
HLT
```

Subroutine:-

```
DIV :
      MVI D,00H; clear D for quotient
NEXT: SUB C ; Subtract divisor from dividend
      INR D
      CMP C ; Repeat subtraction until divisor is
      JNC NEXT                    less than divident
      RET
```

Q. 22) WAP to generate Fibonacci Series.

→ Ans:-

```
      LXI H, 9000H
      MVI D,COUNT ; Initialize counter for no. of data in
      MVI A, 00H  ; Load first number
      MOV B, A
      MOV M, A    ; Store first number
      INX H
      MVI A, 01H  ; Load second number
      MOV C, A    ; Initialize C to store the curre
                    number
      MOV M,A     ; Store second number
      INX H
Loop: MOV A,B
```

```
ADD C   ; Add two numbers
MOV B, C  ; Make current number the previous no.
MOV C, A
MOV M, A
INX H
DCR D     of count
JNZ Loop  ; If value is not zero, repeat the
            process
HLT.
```

Q. 23) WAP to disassemble (separate nibbles) a data stored in memory location 2050H, store the result ~~at~~ 2060H and 2070H.

→ Ans:-

```
LXI H, 2050H
MOV B, M
MOV A, B
ANI 0FH ; mask the upper nibble
STAX 2060H; store the lower nibble
MOV A, B
ANI 0F  ; bring upper nibble to lower
            nibble
RRC
RRC
RRC
RRC
STAX 2070H ; storing upper nibble
HLT
```

Q.24) WAP to combine least significat bit of memory locations 2040H & 2041H and store at 2042H.

→ Ans:-

```
      LDA  2040H
      RRC
      JNC  L1
      MVI  B, 00H
      JMP  LAST
L1:   MVI  B, 01H
LAST: LDA  2041H
      RRC
      JNC  L2
      MVI  C, 00H
      JMP  FAST
L2:   MVI  C, 00H
FAST: MOV  A, B
      ADD  C
      STA  2042H
      HLT
```

PAGE NO.:
DATE:

Q.25) WAP to find the sum of 10 BCD numbers stored sto in memory location starting from 2050H and store the result at the end of table.

→ Ans:-

```
    LXI H, 2050H
    MVI C, 05H
    MVI D, 00H ; carry
    MVI E, 00H ; sum
Loop:
    MOV A, M
    ADD E
    DAA
    MOV E, A
    JNC SKIP
    INR D
    MOV A, D
    DAA
    MOV D, A
SKIP:
    INX H
    DCR C
    JNZ LOOP
    MOV A, E
    STA 205AH
    MOV A, D
    STA 205BH
    HLT.
```

Q.26) WAP to count number of negative, positive and zero elements among 50 data stored in memory location starting from 2050H and store the count at the end of the table.

Ans:-

```
        LXI H, 2050H
        MOV B, 32H  ; for counter
        INX H
        MVI C, 00H  ; for +ve number
        MVI D, 00H  ; for -ve number
        MVI E, 00H  ; for zero number
UP:     MOV A, M
        ORI 00H
        JNZ Docon
        INR E
        JMP Next
Docon:  RAL
        JNC Docon1
        INR D
        JMP Next
Docon1: INR C
Next:   INX H
        DCR B
        JNZ UP
        MOV M, C
        INX H
        MOV M, D
        INX H
        MOV M, E
        HLT
```

Q.27) WAP to convert the ASCII number stored at 2040H to hexadecimal code and store the result at 2041H.

→ Solution:-

```
LDA  2040H
CPI  39H
JC   L1          ; If no. less than 39H, subtract 30H otherwise subtract 37H
SUI  07H         ; [A] ← [A] - 07H
L1:
SUI  30H         ; [A] ← [A] - 30H
STA  2041H
HLT
```

Q.28) WAP to convert a hexadecimal number stored at 2040H into ASCII code and store the result at 2041H.

→ Ans:-

```
LDA  2040H
CPI  0AH
JC   L1
ADI  07H
L1:
ADI  30H
STA  2041H
STA
```

Q.29) 10 16 bit data are stored in memory location starting from 2040H. however the higher order byte of all the data are same. WAP to transfer the lower order bytes to another memory location starting

from 3040H discarding the higher order byte

→ Solution:-

```
MOV C, 0AH ; Counter for 10 data
LXI H, 2040H ; Source memory location point
LXI B, 3040H ; Destination " " "
Loop: MOV A, M
      STAX B
      INX H
      INX H
      INX B
      DCR C
      JNZ Loop
      HLT
```

Q.30> 10 bytes of data are stored in memory location starting from 2050H, which includes some blanks (bytes with zero value) WAP to eliminate the blanks from those data.

→ Solution:-

```
      MVI D, 0AH
      MVI E, 00H
      LXI H, 2050H
      LXI B, 3050H
L2:
      MOV A, M
      CPI 01H
      JC L1
      MOV B, A
      INR E
```

```
L1:   INX H
      INX B
      DCR D
      JNZ L2
      LXI H, 2050H
      LXI B, 3050H
L3:   MOV M, B
      DCR E
      JNZ L3
      HLT
```

PAGE NO.:
DATE:

Q.36) There are two tables T_1 and T_2 containing 10 data bytes in each. WAP to calculate $[T_1 - T_2]$ store the result in third table T_3.

→ Ans:-

```
        LXI B, 4000H    ; table T1
        LXI H, 4010H    ; table T2
        LXI D, 4020H    ; table T3
UP:     LDAX B
        SUB M
        STAX D
        INX B
        INX H
        INX D
        MOV A, C
        CPI 0AH
        JNZ UP
        HLT
```

Q.35) A set of three packed BCD numbers are stored in memory location starting from D050H. The seven segment code of digits 0 to 9 for a common cathode LED are stored in memory location starting from D070H. WAP to and subroutines UNPACK and LEDCODE to unpack BCD numbers and select appropriate seven segment code for each digit.

→ Solution:-

```
        LXI SP, STACK
        LXI H, D050H
        MVI D, 03H
        CALL UNPACK
        HLT
```

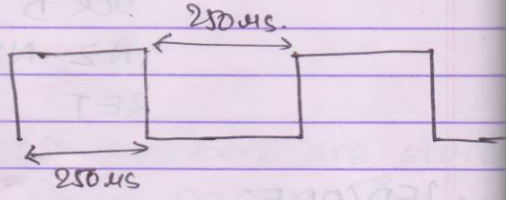
```
UNPACK :
        LXI B, BUFFER
NXTBCD: MOV A,M
        ANI F0H
        RRC
        RRC
        RRC
        RRC
        CALL LEDCODE
        INX B
        MOV A,M
        ANI 0FH
        CALL LEDCODE
        INX B
        INX H
        DCR D
        JNZ NXTBCD
        RET

LEDCODE:
        PUSH H
        LXI H, CODE
        ADD L
        MOV L,A
        MOV A,M
        STAX B
        POP H
        RET

CODE:   3F      ; Digit 0
        06      ; Digit 1
```

```
5B      ; Digit 2
4F      ; Digit 3
66      ; Digit 4
6D      ; Digit 5
7D      ; Digit 6
07      ; Digit 7
7F      ; Digit 8
6F      ; Digit 9
00      ; Invalid digit
```

Q.3) WAP to generate a square wave with the period of 500 microseconds. Assume the system clock period is 325ns and use D_0 to output the square wave.

→ Solution :-

ON and OFF period is 250 µs.




```
        MVI D,
Loop:   MVI A,00H
        OUT D0H
        CALL DELAY
        MVI A,01H
        OUT D0H
        CALL DELAY
        JMP LOOP

DELAY:
        MVI B, n (FFH) ; 7T
L2:     LXI H, y (FFFFH) ; 10x
L1:     DCX H           ; 6xy
        JNZ L1          → 10x(y-1)x n
                        → 7x n
        DCR B           ; 4x n
        JNZ L2          → 10x(n-1)
                        → 7x1.
```

Calculation:-

$7 + 10x + 6xy + 10xy - 10x + 7x + 4x + 10x - 10 + 7$

or, $(21x + 16xy + 4)T = 250\mu s$

or, $(21x + 16xy + 4) \times 0.5\mu s = 250\mu s$

or, $21x + 16xy + 4 = 500$

let $x = 255 = FFH$

then, $y = FFFFH$.

Q.31) A set of eight data bytes are stored in memory location starting from D070H. WAP to add two bytes at a time and store the sum in the same memory location; lower order sum replacing first byte and carry replacing the second byte. If carry is not generated, clear the memory location of second byte.

→ Ans:-

```
        LXI  H, D070H
        MVI  C, 08H
UP:
        MOV  A, M
        INX  H
        ADD  M
        DCX  H
        MOV  M, A      ; Replacing first byte with
        INX  H                lower order sum
        JNC  Down
        MVI  M, 01H
        JMP  Next
Down:
        MVI  M, 00H    ; clearing second byte.
```

Next:

```
INX H
DCR C
JNZ UP
HLT.
```

Q.32) A set of eight data bytes are stored in memory location starting from D070H. WAP to subtract two bytes at a time and store the result in the same memory location i.e. D070 in a sequential manner.

→ solution:-

```
LXI H, D070H
MVI C, 08H
MOV A, M
INX H
SUB M
DCX H
MOV M, A
INX H
```

```
MVI C, 09H
LXI H, D070H
L1: MOV A, M
INX H
SUB M
DCX H
MOV M, A
INX H
DCR C
JNZ L1
HLT
```

Q-33) WAP to count from 0 to 9 with a one second delay between each count. After 9, the counter should again reset to 0 and start again. Use any register pair to generate delay. Assume the clock frequency of microcomputer is 1MHz.

→ Solution:-

```
start: MVI  B,00H
Display: OUT  PORT #
         LXI  H, A2C2H        } 10T states = T0
Loop:    DCX  H
         MOV  A,L
         ORA  H               } TL = 24T states
         JNZ  Loop
         INR  B
         MOV  A,B
         CPI  0AH             } T0
         JNZ  Display
         JZ   start
         HLT
```

Delay Calculation:

Loop Delay T_L = 24T states x T x Count

1 second = 24 x 1 x 10^{-6} x Count

Count = $\frac{1}{24 \times 10^{-6}}$ = 41666 = A2C2H

Programming with 8086.

Q.2) WAP to convert the characters of a string entered by the user into uppercase if the character is lowercase and vice versa.

→

```
title program to convert string into uppercase
    if it is lowercase and vice versa

dosseg
 .model small
 .stack 64h
 .data
        paralst   label   byte
        maxlen    db      50
        actlen    db      ?
        str1      db      50 dup ('$')
        str2      db      13,10, '$'

 .code
    main proc
            mov   ax, @data
            mov   ds, ax
    ; keyboard  string input
            mov   ah, 0ah
            mov   ax, offset paralst
            int   21h
    ; cursor set
            mov   ah, 09h
            lea   dx, str2
            int   21h
```

```
; case conversion
    mov dl, actlen
    mov ch, 0 ; cx = actlen
    mov si, 0 ; si = 0
    mov al, str1[si]
l1: cmp al, 'z'
    ja l2
    cmp str1[si], 'a'
    jb capital
    sub str1[si], 32
    jmp l2
capital:
    cmp str1[si], 'Z'
    ja l2
    cmp str1[si], 'A'
    jb l2
    add str1[si], 32
l2:
    inc si
    loop l1
; keyboard display of string
    mov ah, 09h
    lea dx, str1
    int 21h

    mov ax, 4c00h
    int 21h
main endp
end main
```

Q.4) Count the number of vowels in a string and display it.

Ans:-

```
title program to count numbers of vowels in a
      string and display it

dosseg
    .model small
    .stack 64h
    .data
        paralst     label    byte
        maxlen      db       50
        actlen      db       ?
        str         db       50 dup('$')
        str2        db       13,10,'$'
        str3        db       'aeiouAEIOU'
    .code
        main proc
            mov ax, @data
            mov ds, ax
        ; keyboard string input
            mov ah, 0ah
            mov ax, offset paralst
            int 21h
        ; cursor set
            mov ah, 09h
            lea dx, str2
            int 21h
        ; counting numbers of vowels
            mov bl, 0   ; counter to count vowel
            mov dl, actlen
```

```
        mov dh, 0        ; dx = 00actlen
        mov si, 0
        mov di, 0
l5:     mov al, str[si]
        mov cx, 10       ; aeiouAEIOU = 10
l3:     cmp al, str3[di]
        je l1
        inc di
        loop l3
        jmp l4
l1:     inc bl
l4:     inc si
        dec dx
        jnz l5
; displaying vowel count

        mov ah, 02h
        mov dl, bl
        int 21h

        mov ax, 4c00h
        int 21h
main    endp
end     main
```

```
mov al, bl
mov dx, 0
mov cx, 10
mov bx, 00
l2: div cx
add dx, 30h
push dx
inc bx
mov dx, 0000h
cmp ax, 0
ja l2
mov cx, bx
mov ah, 02h
l3: pop dx
int 21h
loop l3
```

```
mov ax, 4
int 21h
main en
end main
```

PAGE NO.:
DATE:

Q.5) Reverse the ~~string~~ string, entered by the user.

→
```
title program to reverse a string
dosseg
    .model small
    .stack 64h
    .data
        maxlen    db   50
        actlen    db   ?
        str       db   50 dup ('$')
        str2      db   13, 10, '$'

    .code
        main proc
            mov ax, @data
            mov ds, ax
    ; keyboard string input
            mov ah, 0ah
            mov ax, offset maxlen
            int 21h
    ; cursor set
            mov ah, 09h
            lea dx, str2
            int 21h
    ; reverse string
            mov cl, actlen
            shr cl, 01
            mov ch, 0
            mov si, 0
            mov bl, actlen
            sub bl, 1
            mov bh, 0
```

```
13:  mov al, str[si]
     mov dl, str[bx]
     mov str[si], dl
     mov str[bx], al
     dec bx
     inc si
     loop 13
  ; display string (reversed)
       mov ah, 09h
       lea dx, str
       int 21h

       mov ax, 4c00h
       int 21h
       main endp
end main.
```

(Q.6) Test whether the entered string is "EEC Sane eemc.edu.np" or not. If yes display "Valid Use else display "Invalid User".

→

```
title progsam to check user

dosseg
    .model small
    .stack 64h
    .data
        maxlen   db   50
        actlen   db   ?
        str      db   50 dup('$')
        str2     db   13,10,'$'
        str3     db   'EEC Sanepa.eemc.edu.np'
        str4     db   'Valid User','$'
```

```
str5    db   'Invalid User', '$'
```

PAGE NO.:
DATE:

```
.code
        mov ax, @data
        mov ds, ax
; keyboard string input
        mov ah, 0ah
        mov ax, offset maxlen
        int 21h
; clear screen
        mov ah, 06h
        mov al, 00h
        mov bh, 70h
        mov cx, 0000h
        mov dx, 184fh
        int 10h
; cursor set
        mov ah, 09h
        lea dx, str2
        int 21h
; Comparision
        mov al, actlen
        cmp al, 22
        jne dispnotvalid
        mov cx, 22
        mov si, 0
        mov di, 0
l1:     mov al, str1[si]
        cmp al, str3[di]
        jne dispnotvalid
        inc si
        inc di
        loop l1
```

```
; display valid user
        mov ah, 09h
        lea dx, str4
        int 21h
        jmp last
; display not valid user
dispnotvalid: mov ah, 09h
        lea dx, str5
        int 21h
last :  mov ax, 4c00h
        int 21h
        main endp
    end main.
```

Q.9) Check whether the user entered string is pallindrome or not.

→
```
title program to check pallindrome
dosseg
    .model small
    .stack 64h
    .data
        maxlen  db  50
        actlen  db  ?
        str     db  50 dup ['$']
        str2    db  13, 10, '$'
        str3    db  'Pallindrome', '$'
        str4    db  'Not Pallindrome, $'

    .code
        main proc
```

```
        mov ax, @data
        mov ds, ax
; keyboard string input
        mov ah, 0ah
        mov ax, offset maxlen
        int 21h
; clear screen
        mov ah, 06h
        mov al, 00h
        mov bh, 70h
        mov cx, 0
        mov dx, 184fh
        int 10h
; cursor set
        mov ah, 09h
        lea dx, str2
        int 21h
; check pallindrome
        mov cl, actlen
        shr cl, 01          ; cx = actlen/2
        mov ch, 0
        mov si, 0
        mov bl, actlen
        sub bl, 1
        mov bh, 0
l3:     mov al, str2[si]
        cmp al, str2[bx]
        jne l1
        dec bx
        inc si
        loop l3
```

```
; display pallindrome
        mov ah, 09h
        lea dx, str3
        int 21h
        jmp last
; display not pallindrome
dispnotpallindrome: mov ah, 09h
                    lea dx, str4
                    int 21h
last: mov ax, 4c00h
      int 21h
      main endp
end main
```

Q.1) Sort the 10 numbers stored in memory and display them after sorting.

→

```
title program to sort numbers and display the
dosseg
.model small
.stack 100h
.data
    arr   db 9, 8, 4, 16, 10, 2, 20, 25, 19, 27
    msg1  db 'The sorted order is: ', '$'
    tablike db ~~'  '~~ ' ', '$'
.code
    main proc
        mov ax, @data
        mov ds, ax
        mov bl, 9
    l1: mov si, 0
```

```
        mov cx, 9
l2 :    mov al, arr[si]
        inc si
        cmp al, arr[si]
        jbe skip

    ;swapping
        mov dl, arr[si]
        mov arr[si], al
        dec si
        mov arr[si], dl
        inc si
skip:   loop l2
        dec bl
        jnz l1

        lea dx, msg1
        mov ah, 09h
        int 21h

    ; for display
        mov si, offset arr   ; mov si, 0
        mov ch, 00           ; cx = 10
        mov cl, 10
no_of_times:
        ;display one space separation for each
          digit
        mov ah, 09h
        mov dx, offset tablike ; Lea dx, tablike
        int 21h
```

```
        mov ah,00
        mov bx,00
        mov al,[si]      ; ax = smallest number
        push cx
        mov cx, 10
no_of_digits:
        mov dx, 0
        div cx
        add dx, 30h      ; (dx:ax/cx = and
        push dx
        inc bx
        cmp ax,0
        ja no_of_digits
        mov ah, 02h
        mov cx, bx
popping:
        pop dx
        int 21h
        loop popping
        inc si
        pop cx
        loop no_of_times

        mov ax, 4c00h
        int 21h
main endp
  end main
```

PAGE NO.: _____
DATE:

Q.8) Enter a single digit number and find the sum upto that number from zero. Also display the result.

→ Ans:-

title program to input a number and display the sum upto that number from zero.

```
.model small
.stack 100h
.data
    input   db  "Please enter a number", '$'
    output  db  "The sum is: ", '$'
    newline db  0dh, 0ah, '$'
    sum     dw  0
.code
    main proc
        mov ax, @data
        mov ds, ax
        mov ah, 09h     ; ask for input number
        lea dx, input
        int 21h
        mov ah, 01h
        int 21h         ; keyboard input character

        mov ah, 00h
        sub al, 30h     ; find actual number by
        mov cx, ax          converting ASCII
    l1: add sum, cx     ; find sum
        loop l1
        mov ax, sum
        mov dx, 0
```

```
        mov bx,0
        mov cx,10
l2:     div cx
        add dx, 30h
        push dx
        inc bx
        mov dx,00h
        cmp ax,00h
        ja l2

        mov ah,09h
        lea dx,newline  ; print newline
        int 21h

        mov ah,09h
        lea dx,output ; print message
        int 21h

        mov cx,bx
l3:     mov ah,02h
        pop dx
        int 21h
        loop l3

        mov ax, 4c00h
        int 21h
main    endp
end main.
```

PAGE NO.: _____
DATE:

Q.3) Generate the multiplication table of a single digit user entered number.

→ Ans:-

title program to generate multiplication table of an user entered single digit number

```
.model small
.stack 100h
.data
    input db "Please enter a number", '$'
.code
    main proc
        mov ax, @data
        mov dx, ax
    ; Display the message
        mov ah, 09h
        lea dx, input
        int 21h
    ; keyboard number input
        mov ah, 07h ; console input without echo
        int 21h

        sub al, 30h ; convert to actual number
        mov bl, 01h      from ASCII
        mov bh, al
        mov cx, 10
    Back: push cx
        push bx
        mul bl      ; ax = al * bl
        mov cx, 10
        mov bx, 0
```

```
l1:  mov dx,0
     div cx      ;(dx:ax/cx = ax:dx/cx)
     add dx, 30h
     push dx
     inc bx
     cmp ax,0
     ja l1

     mov cx, bx
     mov ah,02h
l2:  pop dx
     int 21h
     loop l2

     mov ah,02h ; A blank character print
     mov dl," " ; separate two numbers
     int 21h

     pop bx
     mov al,bh
     inc bl
     pop cx
     loop Back

     mov ax, 4c00h
     int 21h
main endp
end main.
```

PAGE NO.: _____
DATE:

Q.7) Calculate and display the sum of series: $1+x^2+x^3+x^4....$ upto 10 terms.

→ Solution:-

title program to find sum of series $1+x^2+x^3+x^4+....$ and display

```
dosseg
.model small
.stack 100h
.data
    input db 'Enter the value of x', '$'
    sum dw 0
.code
main proc
mov ax, @data
mov ds, ax
;display message
    mov ah, 09h
    lea dx, input
    int 21h
; single character input from keyboard.
    mov ah,07h ; console input without echo
    int 21h

    mov bl, al
    mov bx, 1
    sub al, 30H ; convert to number
    mov dl,al
    mov dh, 0 ; entered value in dx
    mov cx, 0ah ; 10 terms
    mov bx, 01H ; first number
11: push dx
    add sum, bx   [sum + 1 = 0 + 1]
```

```
pop dx
mul cx
loop L1

mov ax, sum
daa
mov sum, ax

; display:-

mov ax, sum
mov dx, 0000h
mov cx, 10
mov bx, 00
12: div cx
add dx, 30h
push dx
inc bx
mov dx, 0000h
cmp ax, 0
ja 12

mov cx, bx
mov ah, 02h
13: pop dx
int 21h
loop 13

mov ax, 4c00h
int 21h
main endp
end main
```

Q.10) Generate multiplication table of five numbers stored in memory and display in the following format:

5	10	15	20	25	30	35	40	45	50
3	6	9	12	15	18	21	24	27	30

→ Ans:-

Title program to generate multiplication table of 5 stored numbers.

```
dosseg
 .model small
 .stack 100h
 .data
      arrnum  db  5,4,3,6,7
 .code
      main proc
       mov ax, @data
        mov ds, ax
        mov ax, 0
         mov cx, 5
          mov bx, 0
total_no of table:
          mov dl, 1 ; runs 1 to 10 foreach no.
          push cx
          mov cx, 10

 label1:
          mov al, arrnum[bx]
          mul dl    ; ax = al*dl

 ; for display
```

```
push bx
push cx
push dx
mov bx,0
mov cx,10
no_of_digits:
    mov dx,0
    div cx          ; (dx:ax/cx = ax:dx
    add cx,30h
    push dx
    inc bx
    cmp ax,0
    ja no_of_digits
    mov ah,02h
    mov cx,bx
popping:
    pop dx
    int 21h
    loop popping

    mov dl,32h
    mov ah,02h
    int 21h
    pop dx ; popping respective re
             which were pushed b
             display

    pop cx
    pop bx

    int dl
    loop label1
```

```
inc bx
pop cx
mov ah, 02h
mov dl, 0ah
int 21h

mov dl, 0dh
int 21h
loop total_no. of tabel

mov ax, 4c00h
int 21h
main endp
end main.
```

Q.) (1) Generate the fibonacci series and display it (upto 9 terms).

→ Solution:-

```
title program to generate fibonacci series
dosseg
.model small
.data
.code
mov ax, @data
mov ds, ax
mov si, 0009h
mov bl, 30h
mov cl, 31h
mov dl, bl
mov ah, 02h
int 21h
mov dl, 0ah
```

```
        mov ah, 02h
        int 21h
jump:
        mov al, bl
        add al, cl
        aaa
        mov cl, bl
        mov bl, al
        mov dl, al
        mov ah, 02h
        int 21h
        mov dl, 0ah
        mov ah, 02h
        int 21h
        mov dl, 0dh
        mov ah, 02h
        int 21h
        dec si
        jnz jump

        mov ax, 4c00h
        int 21h
main endp
end main
```

Q.13) Count the number of words in a string and display the result.

→ Solution:-

```
title program to count number of words in a
    string
dosseg
.model small
.stack 64h
.data
      maxlen   db  100
      actlen   db  ?
      str      db  100dup('$')
      str1     db  'no. of words is', '$'
.code
      main  proc
            mov ax, @data
            mov ds, ax
            mov cx, 00h
            mov bx, 00h
            mov ax, 00h
            lea dx, maxlen
            mov ah, 0ah
            int 21h

            mov ch, 00h
            mov cl, actlen
            mov bx, 0100h
            cmp str[0], ' '
            jnz l1
            sub dh, 01h
```

```
l1 :    cmp  ste[bx], ''
        jnz  l3
l2 :    inc bx
        dec cx
        cmp  str[bx], ''
        jz  l2
        inc dh
        cmp dh, 0ah
        jb l3
        mov dh, 00h
        inc dl
l3 :    inc bx
        loop l1
        cmp str[bx-1], ''
        jnz l4
        sub dh, 01
        jnz l4 sub dl, 01h
        add dh, 0ah

l4 :    mov bx, dx
        mov ah, 02h
        mov dl, 0ah
        int 21h
        mov dl, 0dh
        int 21h
        lea dx, str1
        mov ah, 09h
        int 21h
```

```
mov dx, bx
add dl, 30h
mov ah, 02h
int 21h
add dh, 30h
mov dl, dh
~~mov dl, d~~
mov ah, 02h
int 21h
mov ax, 4c00h
int 21h
main endp
end main.
```