CSE331L_5 – Flow Control Instructions

1. Write an ASM code to input a number and print whether the number is positive, negative of zero

```
.MODEL SMALL
 .STACK 100H

 .DATA
    PROMPT  DB  'Enter the digit : $'
    MSG     DB  'The entered digit is : $'

 .CODE
   MAIN PROC
     MOV AX, @DATA                ; initialize DS
     MOV DS, AX

     LEA DX, PROMPT               ; load and print PROMPT
     MOV AH, 9
     INT 21H

     MOV AH, 1                    ; read a character
     INT 21H

     MOV BL, AL                   ; save the input character into BL

     MOV AH, 2                    ; carriage return
     MOV DL, 0DH
     INT 21H

     MOV DL, 0AH                  ; line feed
     INT 21H

     LEA DX, MSG                  ; load and print MSG
     MOV AH, 9
     INT 21H

     CMP BL, 30H                  ; compare input digit and 0

     JL @NEGATIVE                 ; jump to label @NEGATIVE if digit<0
     JZ @ZERO                     ; jump to label @ZERO if digit=0
     JG @POSITIVE                 ; jump to label @POSITIVE if digit>0

     @NEGATIVE:                   ; jump label
       MOV DL, 'N'
       JMP @DISPLAY               ; jump to label @DISPLAY

     @ZERO:                       ; jump label
       MOV DL, 'Z'
       JMP @DISPLAY               ; jump to label @DISPLAY

     @POSITIVE:                   ; jump label
       MOV DL, 'P'
       JMP @DISPLAY               ; jump to label @DISPLAY
       @DISPLAY:                    ; jump label
       MOV AH, 2                  ; print the character
       INT 21H

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

2. Write an ASM code to Input an array of 10 size and print it.

```
.MODEL SMALL
 .STACK 100H

 .DATA
    PROMPT_1  DB  \'Enter the Array elements :\',0DH,0AH,\'$\'
    PROMPT_2  DB  \'The Array elements are : $\'

    ARRAY   DW  10 DUP(0)

 .CODE
   MAIN PROC
     MOV AX, @DATA                ; initialize DS
     MOV DS, AX

     MOV BX, 10                   ; set BX=10

     LEA DX, PROMPT_1             ; load and display the string PROMPT_1
     MOV AH, 9
     INT 21H

     LEA SI, ARRAY                ; set SI=offset address of ARRAY

     CALL READ_ARRAY              ; call the procedure READ_ARRAY

     LEA DX, PROMPT_2             ; load and display the string PROMPT_2
     MOV AH, 9
     INT 21H

     LEA SI, ARRAY                ; set SI=offset address of ARRAY

     CALL PRINT_ARRAY             ; call the procedure PRINT_ARRAY

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


 ;-------------------------  Procedure Definitions  ------------------------;



 ;-----------------------------  READ_ARRAY  -------------------------------;


 READ_ARRAY PROC
```

```
  ; this procedure will read the elements for an array
  ; input : SI=offset address of the array
  ;       : BX=size of the array
  ; output : none

  PUSH AX                        ; push AX onto the STACK
  PUSH CX                        ; push CX onto the STACK
  PUSH DX                        ; push DX onto the STACK

  MOV CX, BX                     ; set CX=BX

  @READ_ARRAY:                   ; loop label
    CALL INDEC                   ; call the procedure INDEC

    MOV [SI], AX                 ; set [SI]=AX
    ADD SI, 2                    ; set SI=SI+2

    MOV DL, 0AH                  ; line feed
    MOV AH, 2                    ; set output function
    INT 21H                      ; print a character
  LOOP @READ_ARRAY               ; jump to label @READ_ARRAY while CX!=0

  POP DX                         ; pop a value from STACK into DX
  POP CX                         ; pop a value from STACK into CX
  POP AX                         ; pop a value from STACK into AX

  RET                            ; return control to the calling procedure
READ_ARRAY ENDP


;-----------------------------  PRINT_ARRAY  -----------------------------;


PRINT_ARRAY PROC
  ; this procedure will print the elements of a given array
  ; input : SI=offset address of the array
  ;       : BX=size of the array
  ; output : none

  PUSH AX                        ; push AX onto the STACK
  PUSH CX                        ; push CX onto the STACK
  PUSH DX                        ; push DX onto the STACK

  MOV CX, BX                     ; set CX=BX

  @PRINT_ARRAY:                  ; loop label
    MOV AX, [SI]                 ; set AX=AX+[SI]

    CALL OUTDEC                  ; call the procedure OUTDEC

    MOV AH, 2                    ; set output function
    MOV DL, 20H                  ; set DL=20H
    INT 21H                      ; print a character

    ADD SI, 2                    ; set SI=SI+2
  LOOP @PRINT_ARRAY              ; jump to label @PRINT_ARRAY while CX!=0

  POP DX                         ; pop a value from STACK into DX
  POP CX                         ; pop a value from STACK into CX
  POP AX                         ; pop a value from STACK into AX

  RET                            ; return control to the calling procedure
PRINT_ARRAY ENDP


;-------------------------------  INDEC  -----------------------------------;


INDEC PROC
  ; this procedure will read a number in decimal form
  ; input : none
  ; output : store binary number in AX

  PUSH BX                        ; push BX onto the STACK
  PUSH CX                        ; push CX onto the STACK
  PUSH DX                        ; push DX onto the STACK

  JMP @READ                      ; jump to label @READ

  @SKIP_BACKSPACE:               ; jump label
  MOV AH, 2                      ; set output function
  MOV DL, 20H                    ; set DL=\' \'
  INT 21H                        ; print a character

  @READ:                         ; jump label
  XOR BX, BX                     ; clear BX
  XOR CX, CX                     ; clear CX
  XOR DX, DX                     ; clear DX

  MOV AH, 1                      ; set input function
  INT 21H                        ; read a character

  CMP AL, \"-\"                    ; compare AL with \"-\"
  JE @MINUS                      ; jump to label @MINUS if AL=\"-\"

  CMP AL, \"+\"                    ; compare AL with \"+\"
  JE @PLUS                       ; jump to label @PLUS if AL=\"+\"

  JMP @SKIP_INPUT                ; jump to label @SKIP_INPUT

  @MINUS:                        ; jump label
  MOV CH, 1                      ; set CH=1
  INC CL                         ; set CL=CL+1
  JMP @INPUT                     ; jump to label @INPUT

  @PLUS:                         ; jump label
  MOV CH, 2                      ; set CH=2
  INC CL                         ; set CL=CL+1

  @INPUT:                        ; jump label
    MOV AH, 1                    ; set input function
    INT 21H                      ; read a character

    @SKIP_INPUT:                 ; jump label

    CMP AL, 0DH                  ; compare AL with CR
    JE @END_INPUT                ; jump to label @END_INPUT

    CMP AL, 8H                   ; compare AL with 8H
    JNE @NOT_BACKSPACE            ; jump to label @NOT_BACKSPACE if AL!=8

    CMP CH, 0                     ; compare CH with 0
    JNE @CHECK_REMOVE_MINUS       ; jump to label @CHECK_REMOVE_MINUS if CH!=0

    CMP CL, 0                     ; compare CL with 0
    JE @SKIP_BACKSPACE            ; jump to label @SKIP_BACKSPACE if CL=0
    JMP @MOVE_BACK                ; jump to label @MOVE_BACK

    @CHECK_REMOVE_MINUS:          ; jump label

    CMP CH, 1                     ; compare CH with 1
    JNE @CHECK_REMOVE_PLUS        ; jump to label @CHECK_REMOVE_PLUS if CH!=1

    CMP CL, 1                     ; compare CL with 1
    JE @REMOVE_PLUS_MINUS         ; jump to label @REMOVE_PLUS_MINUS if CL=1

    @CHECK_REMOVE_PLUS:           ; jump label

    CMP CL, 1                     ; compare CL with 1
    JE @REMOVE_PLUS_MINUS         ; jump to label @REMOVE_PLUS_MINUS if CL=1
    JMP @MOVE_BACK                ; jump to label @MOVE_BACK

    @REMOVE_PLUS_MINUS:           ; jump label
      MOV AH, 2                   ; set output function
      MOV DL, 20H                 ; set DL=\' \'
      INT 21H                     ; print a character

      MOV DL, 8H                  ; set DL=8H
      INT 21H                     ; print a character

      JMP @READ                   ; jump to label @READ

    @MOVE_BACK:                   ; jump label

    MOV AX, BX                    ; set AX=BX
    MOV BX, 10                    ; set BX=10
    DIV BX                        ; set AX=AX/BX

    MOV BX, AX                    ; set BX=AX

    MOV AH, 2                     ; set output function
    MOV DL, 20H                   ; set DL=\' \'
    INT 21H                       ; print a character

    MOV DL, 8H                    ; set DL=8H
    INT 21H                       ; print a character

    XOR DX, DX                    ; clear DX
    DEC CL                        ; set CL=CL-1

    JMP @INPUT                    ; jump to label @INPUT

    @NOT_BACKSPACE:               ; jump label

    INC CL                        ; set CL=CL+1

    CMP AL, 30H                   ; compare AL with 0
    JL @ERROR                     ; jump to label @ERROR if AL<0
    CMP AL, 39H                  ; compare AL with 9
    JG @ERROR                    ; jump to label @ERROR if AL>9

    AND AX, 000FH                ; convert ascii to decimal code

    PUSH AX                      ; push AX onto the STACK

    MOV AX, 10                   ; set AX=10
    MUL BX                       ; set AX=AX*BX
    MOV BX, AX                   ; set BX=AX

    POP AX                       ; pop a value from STACK into AX

    ADD BX, AX                   ; set BX=AX+BX
    JS @ERROR                    ; jump to label @ERROR if SF=1
  JMP @INPUT                     ; jump to label @INPUT

  @ERROR:                        ; jump label

  MOV AH, 2                      ; set output function
  MOV DL, 7H                     ; set DL=7H
  INT 21H                        ; print a character

  XOR CH, CH                     ; clear CH

  @CLEAR:                        ; jump label
    MOV DL, 8H                   ; set DL=8H
    INT 21H                      ; print a character

    MOV DL, 20H                  ; set DL=\' \'
    INT 21H                      ; print a character

    MOV DL, 8H                   ; set DL=8H
    INT 21H                      ; print a character
  LOOP @CLEAR                    ; jump to label @CLEAR if CX!=0

  JMP @READ                      ; jump to label @READ

  @END_INPUT:                    ; jump label

  CMP CH, 1                      ; compare CH with 1
  JNE @EXIT                      ; jump to label @EXIT if CH!=1
  NEG BX                         ; negate BX

  @EXIT:                         ; jump label

  MOV AX, BX                     ; set AX=BX

  POP DX                         ; pop a value from STACK into DX
  POP CX                         ; pop a value from STACK into CX
  POP BX                         ; pop a value from STACK into BX

  RET                            ; return control to the calling procedure
INDEC ENDP


;--------------------------------  OUTDEC  --------------------------------;


OUTDEC PROC
```

```
   ; this procedure will display a decimal number
   ; input : AX
   ; output : none

   PUSH BX                        ; push BX onto the STACK
   PUSH CX                        ; push CX onto the STACK
   PUSH DX                        ; push DX onto the STACK

   CMP AX, 0                      ; compare AX with 0
   JGE @START                     ; jump to label @START if AX>=0

   PUSH AX                        ; push AX onto the STACK

   MOV AH, 2                      ; set output function
   MOV DL, \"-\"                    ; set DL=\'-\'
   INT 21H                        ; print the character

   POP AX                         ; pop a value from STACK into AX

   NEG AX                         ; take 2\'s complement of AX

   @START:                        ; jump label

   XOR CX, CX                     ; clear CX
   MOV BX, 10                     ; set BX=10

   @OUTPUT:                       ; loop label
     XOR DX, DX                   ; clear DX
     DIV BX                       ; divide AX by BX
     PUSH DX                      ; push DX onto the STACK
     INC CX                       ; increment CX
     OR AX, AX                    ; take OR of Ax with AX
   JNE @OUTPUT                    ; jump to label @OUTPUT if ZF=0

   MOV AH, 2                      ; set output function

   @DISPLAY:                      ; loop label
     POP DX                       ; pop a value from STACK to DX
     OR DL, 30H                   ; convert decimal to ascii code
     INT 21H                      ; print a character
   LOOP @DISPLAY                  ; jump to label @DISPLAY if CX!=0

   POP DX                         ; pop a value from STACK into DX
   POP CX                         ; pop a value from STACK into CX
   POP BX                         ; pop a value from STACK into BX

   RET                            ; return control to the calling procedure
 OUTDEC ENDP

 END MAIN
```

3. Write an ASM code to copy element from one array to another.

```
DATA SEGMENT
A DB 1,2,3,4,5,6,7,8,9,10
B DB 10 DUP(0)
DATA ENDS
CODE SEGMENT
```

```
         ASSUME DS:DATA,CS:CODE
START:
      MOV AX,DATA
      MOV DS,AX
      MOV CL,10
      LEA BX,A
      LEA SI,B
  L1: MOV CH,BYTE PTR[BX]
      MOV BYTE PTR[SI],CH
      MOV DH,BYTE PTR[SI]
      INC BX
      INC SI
      DEC CL
      CMP CL,00
      JNZ L1
      MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

4. Write an ASM code to read a letter and print if it is Upper case or Lower case

```
.MODEL SMALL
 .STACK 100H

 .DATA
    PROMPT  DB  \'Enter the character : $\'
    MSG_1   DB  \'The input letter is : $\'
    MSG_2   DB  \'The input character is not \"y\" or \"Y\".$\'

 .CODE
   MAIN PROC
     MOV AX, @DATA                ; initialize DS
     MOV DS, AX

     LEA DX, PROMPT               ; load and print PROMPT
     MOV AH, 9
     INT 21H

     MOV AH, 1                    ; read a character
     INT 21H

     MOV BL, AL                   ; save the input character into BL

     MOV AH, 2                    ; carriage return
     MOV DL, 0DH
     INT 21H

     MOV DL, 0AH                  ; line feed
     INT 21H

     CMP BL, \"y\"                  ; compare input character and \"y\"

     JE @DISPLAY                  ; jump to label @DISPLAY if input=y

     CMP BL, \"Y\"                  ; compare input character and \"Y\"

     JE @DISPLAY                  ; jump to label @DISPLAY input=\"Y\"

     LEA DX,MSG_2                 ; load and print MSG_2
     MOV AH, 9
     INT 21H

     JMP @EXIT                    ; jump to label @EXIT

     @DISPLAY:                    ; jump label
       LEA DX,MSG_1               ; load and print MSG_1
       MOV AH, 9
       INT 21H

       MOV AH, 2                  ; print the character
       MOV DL, BL
       INT 21H
     @EXIT:                       ; jump label

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

5. Write an ASM code to read a binary number and revise it bit wise.

```
.MODEL SMALL
 .STACK 100H

 .DATA
   PROMPT_1  DB  \'Enter the binary number (max 8-bit) : $\'
   PROMPT_2  DB  0DH,0AH,\'The given binary number in reverse order is : $\'

 .CODE
   MAIN PROC
     MOV AX, @DATA                ; initialize DS
     MOV DS, AX

     LEA DX, PROMPT_1             ; load and display PROMPT_1
     MOV AH, 9
     INT 21H

     XOR BL, BL                   ; clear BL
     MOV CX, 8                    ; initialize loop counter
     MOV AH, 1                    ; set input function

     @INPUT:                      ; jump label
       INT 21H                    ; read a digit
       CMP AL, 0DH                ; compare digit with carriage return
       JE @END                    ; jump to label @END if carriage return
       AND AL, 0FH                ; convert ascii to decimal code
       SHL BL, 1                  ; rotate BX to left by 1 bit
       OR BL, AL                  ; set the LSB of BX with input
     LOOP @INPUT                  ; jump to label @INPUT

     @END:                        ; jump label

     MOV AL, BL                   ; copy BL into AL
     MOV CX, 8                    ; initialize loop counter

     @LOOP:                       ; loop label
       SHL AL, 1                  ; shift AL to left by i bit
       RCR BL, 1                  ; rotate BL right through carry
     LOOP @LOOP                    ; jump to label @LOOP

     LEA DX, PROMPT_2              ; load and display PROMPT_2
     MOV AH, 9
     INT 21H

     MOV CX, 8                     ; initialize loop counter
     MOV AH, 2                     ; set output function


      @OUTPUT:                      ; jump label
       SHL BL, 1                   ; shift left BL by 1 bit

       JNC @ZERO                   ; jump to label @ZERO if CF=0
         MOV DL, 31H               ; set DL=1
         JMP @DISPLAY              ; jump to label @DISPLAY

       @ZERO:                      ; jump label
         MOV DL, 30H               ; set DL=0

       @DISPLAY:                   ; jump label
         INT 21H                   ; display digit
     LOOP @OUTPUT                  ; jump to label @OUTPUT

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

6. Write an ASM code to read a HEX number and print the binary of it.

```
.MODEL SMALL
 .STACK 100H

 .DATA
   PROMPT_1  DB  \'Enter the hexadecimal number ( max 4-digit ) : $\'
   PROMPT_2  DB  0DH,0AH,\'The equivalent 16-bit binary number is : $\'
   ILLEGAL   DB  0DH,0AH,\'Illegal hex number. Try again : $\'

   COUNT     DB  ?

 .CODE
   MAIN PROC
     MOV AX, @DATA                 ; initialize DS
     MOV DS, AX

     LEA DX, PROMPT_1              ; load and display the string PROMPT_1
     MOV AH,9
     INT 21H

     JMP @START                    ; jump to label @START_2

     @START_1:                     ; jump label
       LEA DX, ILLEGAL             ; load and display the string ILLEGAL
       MOV AH, 9
       INT 21H
```

```
@START:                      ;
  XOR BX, BX                 ; clear BX
  MOV COUNT, 30H             ; initialize loop counter

@START_2:                    ; jump label
  MOV AH, 1                  ; set input function
  INT 21H                    ; read a character

  CMP AL, 0DH                ; compare Al with CR


  JNE @SKIP                  ; jump to label @SKIP if AL!=CR

  CMP COUNT, 30H             ; compare COUNT with 0
  JBE @START_1               ; jump to label @START_1 if COUNT<=0
  JMP @END                   ; jump to label @END

  @SKIP:                     ; jump label

  CMP AL, \"A\"                ; compare AL with \"A\"
  JB @DECIMAL                ; jump to label @DECIMAL if AL<A

  CMP AL, \"F\"                ; compare AL with \"F\"
  JA @START_1                ; jump to label @START_1 if AL>F
  ADD AL, 09H                ; add 9 to AL
  JMP @OK                    ; jump to label @OK

  @DECIMAL:                  ; jump label
    CMP AL, 30H              ; compare AL with 0
    JB @START_1              ; jump to label @START_1 if AL<0

    CMP AL, 39H              ; compare AL with 9
    JA @START_1              ; jump to label @START_1 if AL>9

  @OK:                       ; jump label

  INC COUNT                  ; increment the COUNT variable

  AND AL, 0FH                ; convert the ascii into binary code

  MOV CL, 4                  ; set CL=4
  SHL AL, CL                 ; shift AL towards left by 4 positions

  MOV CX, 4                  ; set CX=4

  @LOOP_1:                   ; loop label
    SHL AL, 1                ; shift AL towards left by 1 position
    RCL BX, 1                ; rotate BX towards left by 1 position
                             ; through carry
  LOOP @LOOP_1               ; jump to label @LOOP_1 if CX!=0

 CMP COUNT, 34H              ; compare COUNT with 4
 JE @END                     ; jump to label @END if COUNT=4
 JMP @START_2                ; jump to label @START_2

@END:                        ; jump label

LEA DX, PROMPT_2             ; load and display the string PROMPT_2
MOV AH, 9
INT 21H
```

```
     MOV CX, 16                   ; set CX=16
     MOV AH, 2                    ; set output function

     @LOOP_2:                     ; loop label
       SHL BX, 1                  ; shift BX towards left by 1 position
       JC @ONE                    ; jump to label @ONE if CF=1
       MOV DL, 30H                ; set DL=0
       JMP @DISPLAY               ; jump to label @DISPLAY



       @ONE:                      ; jump label
         MOV DL, 31H              ; set DL=1

       @DISPLAY:                  ; jump label
         INT 21H                  ; display a character
     LOOP @LOOP_2                 ; jump to label @LOOP_2 if CX!=0

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

7. Write an ASM code to read a binary number (8-digit) and print the sum.

```
.MODEL SMALL
 .STACK 100H

 .DATA
   PROMPT_1  DB  0DH,0AH,\'Enter the first binary number ( max 8-digits ) : $\'
   PROMPT_2  DB  0DH,0AH,\'Enter the second binary number ( max 8-digits ) : $\'
   PROMPT_3  DB  0DH,0AH,\'The SUM of given binary numbers in binary form is : $\'
   ILLEGAL   DB  0DH,0AH,\'Illegal character. Try again.$\'

 .CODE
   MAIN PROC
     MOV AX, @DATA                ; initialize DS
     MOV DS, AX

     JMP @START_2                 ; jump to label @START_2

     @START_1:                    ; jump label
       LEA DX, ILLEGAL            ; load and display the string ILLEGAL
       MOV AH, 9
       INT 21H

     @START_2:                    ; jump label
       XOR BX, BX                 ; clear BX

       LEA DX, PROMPT_1           ; load and display the string PROMPT_1
       MOV AH, 9
       INT 21H

       MOV CX, 8                  ; initialize loop counter
       MOV AH, 1                  ; set input function

       @LOOP_1:                   ; loop label
         INT 21H                  ; read a character

         CMP AL, 0DH              ; compare AL with CR
```

```
      JNE @SKIP_1              ; jump to label @SKIP_1 if AL!=0DH

      CMP CX, 8                ; compare CX with 8
      JE @START_1              ; jump to label @START_1 if CX=8
      JMP @EXIT_LOOP_1         ; jump to label @EXIT_LOOP_1

      @SKIP_1:                 ; jump label
        AND AL, 0FH            ; convert ascii into decimal code
        SHL BL, 1              ; shift BL towards left by 1 position

       OR BL, AL              ; set the LSB of BL with LASB of AL
    LOOP @LOOP_1               ; jump to label @LOOP_1 if CX!=0

    @EXIT_LOOP_1:              ; jump label

    LEA DX, PROMPT_2           ; load and display the string PROMPT_2
    MOV AH, 9
    INT 21H

    MOV CX, 8                  ; initialize loop counter
    MOV AH, 1                  ; set input function

    @LOOP_2:                   ; loop label
      INT 21H                  ; read a character

      CMP AL, 0DH              ; compare AL with CR
      JNE @SKIP_2              ; jump to label @SKIP_2 if AL!=0DH

      CMP CX, 8                ; compare CX with 8
      JE @START_2              ; jump to label @START_2 if CX=8
      JMP @EXIT_LOOP_2         ; jump to label @EXIT_LOOP_2

      @SKIP_2:                 ; jump label
        AND AL, 0FH            ; convert ascii into decimal code
        SHL BH, 1              ; shift BH towards left by 1 position
        OR BH, AL              ; set the LSB of BH with LASB of AL
    LOOP @LOOP_2               ; jump to label @LOOP_2 if CX!=0

    @EXIT_LOOP_2:              ; jump label

    LEA DX, PROMPT_3           ; load and display the string PROMPT_3
    MOV AH, 9
    INT 21H

    ADD BL, BH                 ; add BL and BH
    JNC @SKIP                  ; jump to label @SKIP if CF=1
      MOV AH, 2                ; print the digit 1 i.e. carry
      MOV DL, 31H
      INT 21H

    @SKIP:                     ; jump label

    MOV CX, 8                  ; initialize loop counter
    MOV AH, 2                  ; set output function

    @LOOP_3:                   ; loop label
      SHL BL, 1                ; shift BL towards left by 1 position
      JC @ONE                  ; jump to label @ONE if CF=1
      MOV DL, 30H              ; set DL=0
      JMP @DISPLAY             ; jump to label @DISPLAY
```

```
         @ONE:                    ; jump label
           MOV DL, 31H            ; set DL=1

         @DISPLAY:                ; jump label
           INT 21H                ; print the character
       LOOP @LOOP_3               ; jump to label @LOOP_3 if CX!=0

     MOV AH, 4CH                  ; return control to DOS
     INT 21H

    MAIN ENDP
 END MAIN
```

8. Write an ASM code to check a palindrome string.

```
Data Segment
  str1 db \'MADAM\',\'$\'
  strlen1 dw $-str1
  strrev db 20 dup(\' \')
  str_palin db \'String is Palindrome.\',\'$\'
  str_not_palin db \'String is not Palindrome.\',\'$\'
Data Ends

Code Segment
  Assume cs:code, ds:data

  Begin:
    mov ax, data
    mov ds, ax
    mov es, ax
    mov cx, strlen1
    add cx, -2

    lea si, str1
    lea di, strrev

    add si, strlen1
    add si, -2
    L1:
       mov al, [si]
       mov [di], al
       dec si
       inc di
       loop L1
       mov al, [si]
       mov [di], al
       inc di
       mov dl, \'$\'
       mov [di], dl
       mov cx, strlen1

    Palin_Check:
       lea si, str1
       lea di, strrev
       repe cmpsb
       jne Not_Palin

    Palin:
       mov ah, 09h
       lea dx, str_palin
       int 21h
       jmp Exit

    Not_Palin:
       mov ah, 09h
       lea dx, str_not_palin
       int 21h

    Exit:

       mov ax, 4c00h
       int 21h
Code Ends
End Begin
```

9. Write an ASM code to read a binary number and print the factorial of the binary number (MUL instruction)

```
.MODEL SMALL
 .STACK 100H

 .DATA
   PROMPT_1   DB  \'Enter a Positive Binary number (max. 1000) : $\'
   PROMPT_2   DB  0DH,0AH,\'The Factorial of the given number is : $\'
   ILLEGAL    DB  0DH,0AH,\'Illegal character. Try again : $\'

 .CODE
   MAIN PROC
     MOV AX, @DATA                ; initialize DS
     MOV DS, AX

     LEA DX, PROMPT_1             ; load and display the string PROMPT_1
     MOV AH, 9
     INT 21H

     CALL BINARY_INPUT            ; call the procedure BINARY_INPUT

     CALL FACTORIAL               ; call the procedure FACTORIAL

     LEA DX, PROMPT_2             ; load and display the string PROMPT_2
     MOV AH, 9
     INT 21H

     CALL BINARY_OUTPUT           ; call the procedure BINARY_OUTPUT

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


 ;-------------------------  Procedure Definitions  ------------------------;



 ;----------------------------  BINARY_INPUT  ------------------------------;


 BINARY_INPUT PROC
```

```
  ; this procedure will read a number in binary form

  ; input : none
  ; output : store binary number in BL
  ; uses : MAIN

  JMP @START                     ; jump to label @START

  @ERROR:                        ; jump label

  LEA DX, ILLEGAL                ; load and display the string ILLEGAL
  MOV AH, 9
  INT 21H

  @START:                        ; jump label

  MOV CX, 4                      ; initialize loop counter
  XOR BX, BX                     ; clear BX
  MOV AH, 1                      ; set input function

  @INPUT:                        ; loop label
    INT 21H                      ; read a digit

    CMP AL, 0DH                  ; compare input and CR
    JE @END                      ; jump to label @END if input is CR

    CMP AL, 30H                  ; compare AL with 0
    JL @ERROR                    ; jump to label @ERROR if AL<0

    CMP AL, 31H                  ; compare AL with 1
    JG @ERROR                    ; jump to label @ERROR if AL>1

    AND AL, 0FH                  ; convert ascii to decimal code
    SHL BL, 1                    ; shift BL by 1 position towards left
    OR  BL, AL                   ; place the input decimal digit in BL
  LOOP @INPUT                    ; jump to label @INPUT if CX!=0

  @END:                          ; jump label

  RET                            ; return control to the calling procedure
BINARY_INPUT ENDP


;----------------------------  BINARY_OUTPUT  -----------------------------;


BINARY_OUTPUT PROC
  ; this procedure will display a number in binary form
  ; input : BX
  ; output : none
  ; uses : MAIN

  MOV CX, 16                     ; initialize loop counter
  MOV AH, 2                      ; set output function

  @OUTPUT:                       ; loop label
    SHL BX, 1                    ; shift BX by 1 position towards left
    JC @ONE                      ; jump to label @ONE if CF=1
    MOV DL, 30H                  ; move 0 to DL
    JMP @DISPLAY                 ; jump tp label @DISPLAY
    @ONE:                        ; jump label


     MOV DL, 31H                ; move 1 to DL

    @DISPLAY:                    ; jump label
      INT 21H                    ; display a digit
  LOOP @OUTPUT                   ; jump to label @OUTPUT if CX!=0

  RET                            ; return control to the calling procedure
BINARY_OUTPUT ENDP


;-----------------------------  FACTORIAL  --------------------------------;


FACTORIAL PROC
  ; this procedure will compute the factorial of a given number
  ; input : BL
  ; output : store the factorial of the number in BX
  ; uses : MAIN

  MOV AX, 1                      ; set AX=1

  XOR CX, CX                     ; clear CX
  MOV CX, BX                     ; set CX=BX

  @LOOP:                         ; loop label
    MUL CX                       ; multiply CX with AL i.e. AX=AL*CX
  LOOP @LOOP                     ; jump to label @LOOP if CX!=0

  MOV BX, AX                     ; set BX=AX

  RET                            ; return control to the calling procedure
FACTORIAL ENDP

END MAIN
```