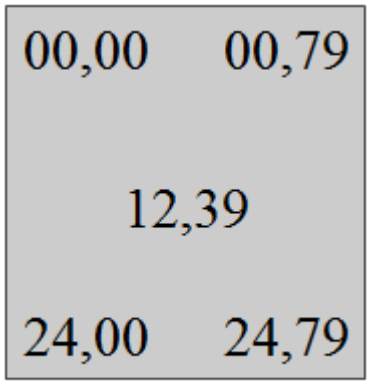
CS222- Lab 6

Assembly language Programming

The goal of this is to familiarize the students with 8086 assembly language features.

INT 10 /INT 21

- **INT 10H subroutines are used to communicate with the computer's screen video.**



Cursor Locations

- INT 10H Function 06
  - AL = number of lines to scroll (with AL=00, window will be cleared)
  - BH = attribute of blank rows
  - CH, CL = upper row, left column
  - DH, DL = lower row, right column

- INT 10H function 02; setting the cursor to a specific location
  - Function AH = 02 will change the position of the cursor to any location.
  - The desired cursor location is in DH = row, DL = column

P1:
Write a program that clears the screen and sets the cursor at the center of the screen

```
; clearing the screen
        MOV AX, 0600H     ;scroll the entire page
        MOV BH, 07        ; normal attribute (white on black)
        MOV CX, 0000      ; upper left
            MOV DX,184FH      ; lower right
            INT 10H
;setting the cursor at the center
        MOV AH,02  ; set cursor option
        MOV BH, 00 ; page 0
```

```
MOV DL, 39 ;
MOV DH, 12          ;
INT 10H
```

- **INT 10H function 03; get current cursor position**

```
MOV AH, 03
MOV BH, 00
INT 10H
```

Test Different examples
Ans here:

```
.model small
.stack 64
.code
; clearing the screen
start:  MOV AX, 0600H     ;scroll the entire page
        MOV BH, 07     ; normal attribute (white on black)
        MOV CX, 0000       ; upper left
      MOV DX,184FH    ; lower right
      INT 10H
;setting the cursor at the center
        MOV AH,02  ; set cursor option
        MOV BH, 00 ; page 0
        MOV DL, 00 ;
        MOV DH, 00     ;
        INT 10H
        end start
        .exit
        .end

;1
.model small
.stack 64
.code
; clearing the screen
start:  MOV AX, 0600H     ;scroll the entire page
        MOV BH, 07     ; normal attribute (white on black)
        MOV CX, 0000       ; upper left
      MOV DX,184FH    ; lower right
      INT 10H
;setting the cursor at the center
        MOV AH,02  ; set cursor option
        MOV BH, 00 ; page 0
        MOV DL, 79 ;
        MOV DH, 00     ;
        INT 10H
        end start
```

```
        .exit
        .end
;2
.model small
.stack 64
.code
; clearing the screen
start:  MOV AX, 0600H     ;scroll the entire page
        MOV BH, 07     ; normal attribute (white on black)
        MOV CX, 0000       ; upper left
      MOV DX,184FH    ; lower right
      INT 10H
;setting the cursor at the center
        MOV AH,02  ; set cursor option
        MOV BH, 00 ; page 0
        MOV DL, 39 ;
        MOV DH, 12     ;
        INT 10H
        end start
        .exit
        .end
;3
.model small
.stack 64
.code
; clearing the screen
start:  MOV AX, 0600H     ;scroll the entire page
        MOV BH, 07     ; normal attribute (white on black)
        MOV CX, 0000       ; upper left
      MOV DX,184FH    ; lower right
      INT 10H
;setting the cursor at the center
        MOV AH,02  ; set cursor option
        MOV BH, 00 ; page 0
        MOV DL, 00 ;
        MOV DH, 24     ;
        INT 10H
        end start
        .exit
        .end

4;4
.model small
.stack 64
.code
; clearing the screen
start:  MOV AX, 0600H     ;scroll the entire page
        MOV BH, 07     ; normal attribute (white on black)
        MOV CX, 0000       ; upper left
      MOV DX,184FH    ; lower right
      INT 10H
;setting the cursor at the center
```

```
        MOV AH,02  ; set cursor option
        MOV BH, 00 ; page 0
        MOV DL, 79 ;
        MOV DH, 24     ;
        INT 10H
        end start
        .exit
        .end

;5
.model small
.stack 64
.code
; clearing the screen
start:  MOV AX, 0600H      ;scroll the entire page
        MOV BH, 07      ; normal attribute (white on black)
        MOV CX, 0000        ; upper left
      MOV DX,184FH    ; lower right
      INT 10H
;setting the cursor at the center
        MOV AH,02  ; set cursor option
        MOV BH, 00 ; page 0
        MOV DL, 39 ;
        MOV DH, 12     ;
        INT 10H
        end start
        .exit
        .end
```

P2:

- **INT 21H is provided by DOS to be invoked to perform extremely useful functions.**
- **INT 21H Option 09: Outputting a string of data to the monitor**
  - **INT 21H can be used to send a set of ASCII data to the monitor.**
  - **AH = 09; DX = offset address of the ASCII data to be displayed.**
  - **INT 21H option 09 will display the ASCII data string pointed to by DX until it encounters the dollar sign "$".**

A11 DB‘ India is my country’,’$’

```
lea DX, msg
     MOV AH,09
     INT 21H
```

Test Different examples

Ans here:

```
.model small
.stack 64

.data
msg  DB "India is my  country $"
msg1 DB 10,13,"Hi, i successfully printed this in a new line $"
.code
start:  mov ax, @data
        mov ds, ax
        mov dx, offset msg
        mov AH, 09
          INT 21H
        mov dx, offset msg1
        mov AH, 09
          INT 21H
        MOV AH, 04CH
        INT 21H
        end start
        .end
```

P3:

- **INT 21H Option 02**: Outputting a single character to the monitor
  - DL is loaded with the character first

```
MOV AH,02
Movdl,'j'
      INT 21H
```

Test Different examples

Ans here:

```
.model small
.stack 64
.code
start:  MOV AH,02
        Mov dl,'h'
        INT 21H
        mov ah,4ch
        int 21h
        end start
        .end
```

P4:

Study the following ALP for:

- Clear the screen
- Set the cursor to the center
- Display the message “This is a test of the display routine”

```
        .MODEL SMALL
        .STACK 64
;--------------
        .DATA
MESSAGE DB      'This is a test of the display routine','$'
;--------------
        .CODE
MAIN      PROC   FAR
        MOV    AX,@DATA
        MOV    DS,AX
        CALL   CLEAR          ;CLEAR THE SCREEN
        CALL   CURSOR         ;SET CURSOR POSITION
        CALL   DISPLAY        ;DISPLAY MESSAGE
        MOV    AH,4CH
        INT    21H            ;GO BACK TO DOS
       MAIN      ENDP
;THIS SUBROUTINE CLEARS THE SCREEN
    CLEAR PROC
        MOV    AX,0600H          ;SCROLL SCREEN FUNCTION
        MOV    BH,07             ;NORMAL ATTRIBUTE
        MOV    CX,0000           ;SCROLL FROM ROW=00,COL=00
        MOV    DX,184FH          ;TO ROW=18H,COL=4FH
        INT    10H             ;INVOKE INTERRUPT TO CLEAR SCREEN
        RET
       CLEAR  ENDP
;THIS SUBROUTINE SETS THE CURSOR AT THE CENTER OF THE SCREEN
      CURSOR   PROC
        MOV    AH,02           ;SET CURSOR FUNCTION
        MOV    BH,00           ;PAGE 00
        MOV    DH,12           ;CENTER ROW
        MOV    DL,39           ;CENTER COLUMN
        INT    10H          ;INVOKE INTERRUPT TO SET CURSOR POSITION
        RET
          CURSOR   ENDP
;THIS SUBROUTINE DISPLAYS A STRING ON THE SCREEN
        DISPLAY     PROC
        MOV    AH,09             ;DISPLAY FUNCTION
        MOV    DX,OFFSET MESSAGE    ;DX POINTS TO OUTPUT BUFFER
        INT    21H             ;INVOKE INTERRUPT TO DISPLAY STRING
        RET
          DISPLAY     ENDP
        END    MAIN
```

Your task:

Write ALP to performs the following, (1) clears the screen, (2) sets the cursor at the beginning ;of the third line from the top of the screen, (3) accepts the message "IBM perSonal ;COmputer" from the keyboard, (4) converts lowercase letters of the message to uppercase, ; (5) displays the converted ;results on the next line.

ANS Here:

```
.MODEL SMALL
.STACK 64
.DATA
                    MESSAGE DB   "This is a test of the display routine $"
                    STR1 DB "ENTER STRING ->$"
                    NEWLINE DB 10,13,"$"
                    INSTR1 DB 20 DUP("$")
                    STR11 DB "STRING IS : ->$"
                    LOWERC DB "Enter String to be converted to uppercase $"

.CODE
MAIN       PROC    FAR
         MOV    AX,@DATA
         MOV    DS,AX
         CALL   CLEAR          ;CLEAR THE SCREEN
         CALL   CURSOR         ;SET CURSOR POSITION
         CALL   DISPLAY        ;DISPLAY MESSAGE
           CALL   ACCEPT

         MOV    AH,4CH
         INT    21H              ;GO BACK TO DOS
        MAIN       ENDP
;THIS SUBROUTINE CLEARS THE SCREEN
     CLEAR PROC
         MOV    AX,0600H           ;SCROLL SCREEN FUNCTION
         MOV    BH,07              ;NORMAL ATTRIBUTE
         MOV    CX,0000            ;SCROLL FROM ROW=00,COL=00
         MOV    DX,184FH           ;TO ROW=18H,COL=4FH
         INT    10H              ;INVOKE INTERRUPT TO CLEAR SCREEN
         RET
        CLEAR  ENDP
;THIS SUBROUTINE ACCEPTS THE STRING

        ACCEPT PROC
               LEA SI,INSTR1

            MOV AH,09H
            LEA DX,NEWLINE
            INT 21H

              MOV AH,09H
              LEA DX,STR1
              INT 21H

            MOV AH,0AH
            MOV DX,SI
            INT 21H

            MOV AH,09H
```

```
        LEA DX,NEWLINE
        INT 21H

          MOV AH,09H
LEA DX,STR11
INT 21H

        MOV AH,09H
        LEA DX,INSTR1+2
        INT 21H

        MOV AH,09H
        LEA DX,NEWLINE
        INT 21H

          MOV AH,09H
        LEA DX,LOWERC
        INT 21H

          MOV AH,09H
        LEA DX,NEWLINE
        INT 21H

LEA SI,INSTR1
MOV AH,01H

READ:
INT 21H
MOV BL,AL

          CMP AL,0DH
JE  DISPLAYSTRING
          CMP AL,61H
          JC CHANGE
          XOR AL,20H

CHANGE: MOV [SI],AL
INC SI
          JMP READ

DISPLAYSTRING:
          MOV AL,'$'
MOV [SI],AL

          LEA DX,INSTR1
MOV AH,09H
INT 21H

          RET
ACCEPT ENDP
```

```
;THIS SUBROUTINE SETS THE CURSOR AT THE CENTER OF THE SCREEN
      CURSOR   PROC
         MOV    AH,02          ;SET CURSOR FUNCTION
         MOV    BH,00          ;PAGE 00
         MOV    DH,03          ;CENTER ROW
         MOV    DL,00          ;CENTER COLUMN
         INT    10H          ;INVOKE INTERRUPT TO SET CURSOR POSITION
         RET
           CURSOR   ENDP
;THIS SUBROUTINE DISPLAYS A STRING ON THE SCREEN
        DISPLAY     PROC
         MOV    AH,09            ;DISPLAY FUNCTION
         MOV    DX,OFFSET MESSAGE    ;DX POINTS TO OUTPUT BUFFER
         INT    21H              ;INVOKE INTERRUPT TO DISPLAY STRING
         RET
           DISPLAY     ENDP
         END    MAIN
```

```
.model small
.stack 64
.data

a1 DB 'H', 'e', 'l', 'l', 'o'
a2 dw 111h, 222h, 333h, 444h, 555h
.code

start:  movax,@data
movds,ax
        LEA SI, a1
        MOV CX, 5
        MOV AH, 0Eh
m: LODSB
      INT 10h
      LOOP m
;Load word at DS:[SI] into AX
      LEA SI, a2
        MOV CX, 5
        REP LODSW
mov ah,4ch
int 21h
end
```

P5:

String Instructions

Load byte at DS:[SI] into AL. Update SI.
Algorithm:
AL = DS:[SI]
if DF = 0 then
SI = SI + 1
else
SI = SI - 1

Example

a1 DB 'H', 'e', 'l', 'l', 'o'

LEA SI, a1
MOV CX, 5
MOV AH, 0Eh
m: LODSB
INT 10h
LOOP m

Your task:

Load word at DS:[SI] into AX. Update SI.

Your Answer Here:

```
.model small
 .stack 64
.data a1 DB 's', 't', 'a', 'r', 't'
 a2 dw 233h, 322h, 343h, 454h, 555h
 .code
 start: mov ax,@data
 mov ds,ax
LEA SI, a1
 MOV CX, 5
 MOV AH, 0Eh
 m: LODSB
 INT 10h
LOOP m ;Load word at DS:[SI] into AX
```

```
LEA SI, a2
 MOV CX, 5
 REP LODSW
 mov ah,4ch
 int 21h
end start
```

P6:

Copy byte at DS:[SI] to ES:[DI]. Update SI and DI.

Algorithm:
ES:[DI] = DS:[SI]
if DF = 0 then
SI = SI + 1
DI = DI + 1
else
SI = SI - 1
DI = DI - 1

Example

```
a1 DB 1,2,3,4,5
a2 DB 5 DUP(0)

LEA SI, a1
LEA DI, a2
MOV CX, 5
REP MOVSB
```

Your task

Copy **word** at DS:[SI] to ES:[DI]. Update SI and DI.

Answer Here:

```
.model small
.stack 64

.data
a1 dw 233h, 322h, 343h, 454h, 555h
a2 dw 323h, 233h, 400h, 500h, 444h

.code
start : mov  ax,@data
        mov  ds,ax
        LEA SI, a1
        LEA DI, a2
        MOV CX, 5
        REP MOVSW

        mov ah, 4ch
        int 21h
end start
.end
```

P7:

PUSH l general purpose registers DI, SI, BP to stack and pop BX, DX,CX, AX from the stack.

```
POP DI
POP SI
POP BP

POP BX
POP DX
```

POP CX
POP AX

Your task

Push all general purpose registers AX, CX, DX, BX, SP,BP, SI, DI in the stack

Pop to different register and verify the operation.
Answer Here:

```
.model small
.stack 64
.data
.code
start:
        MOV AX, 0001H

        MOV BX, 0002H

        MOV CX, 0003H

        MOV DX, 0004H

        MOV SP, 0005H

        MOV BP, 0006H

        MOV SI, 0007H

        MOV DI, 0008H

        PUSH AX

        PUSH CX

        PUSH DX
```

```
        PUSH BX

        PUSH SP

        PUSH BP

        PUSH SI

        PUSH DI

        POP SI

        POP DI

        POP SP

        POP BP

        POP AX

        POP CX

        POP DX

        POP BX

        INT 21H

end start
.end
```

P8:

Compare String: Study the following sample program

```
.model small
.stack 64
.data

  STR1 DB "ENTER FIRST STRING HERE ->$"
  STR2 DB "ENTER SECOND STRING HERE ->$"
  STR11 DB "FIRST STRING : ->$"
  STR22 DB "SECOND STRING: ->$"

  INSTR1 DB 20 DUP("$")
  INSTR2 DB 20 DUP("$")
  NEWLINE DB 10,13,"$"
  N DB ?
  S DB ?
  MSG1 DB "BOTH STRING ARE SAME$"
  MSG2 DB "BOTH STRING ARE DIFFERENT$"
.code


START:  MOV AX,@DATA
        MOV DS,AX

        LEA SI,INSTR1
        LEA DI,INSTR2

;GET STRING
        MOV AH,09H
        LEA DX,STR1
        INT 21H

        MOV AH,0AH
        MOV DX,SI
        INT 21H


        MOV AH,09H
```

```
        LEA DX,NEWLINE
        INT 21H

        MOV AH,09H
        LEA DX,STR2
        INT 21H

        MOV AH,0AH
        MOV DX,DI
        INT 21H


        MOV AH,09H
        LEA DX,NEWLINE
        INT 21H


;PRINT THE STRING

        MOV AH,09H
        LEA DX,STR11
        INT 21H

        MOV AH,09H
        LEA DX,INSTR1+2
        INT 21H

        MOV AH,09H
        LEA DX,NEWLINE
        INT 21H

        MOV AH,09H
        LEA DX,STR22
        INT 21H

        MOV AH,09H
        LEA DX,INSTR2+2
        INT 21H
```

```
        MOV AH,09H
        LEA DX,NEWLINE
        INT 21H

;STRING COMPARISION
        MOV BX,00

        MOV BL,INSTR1+1
        MOV BH,INSTR2+1

        CMP BL,BH
        JNE L1

        ADD SI,2
        ADD DI,2

      L2:MOV BL,BYTE PTR[SI]
        CMP BYTE PTR[DI],BL
        JNE L1
        INC SI
        INC DI
        CMP BYTE PTR[DI],"$"
        JNE L2

        MOV AH,09H
        LEA DX,MSG1
        INT 21H

        JMP L5

      L1:MOV AH,09H
        LEA DX,MSG2
        INT 21H



     L5:
```

```
        MOV AH,09H
        LEA DX,NEWLINE
        INT 21H

        MOV AH,4CH
        INT 21H

END START
```

Modify the program to report the position ofthe difference.

Your Answer Here:

```
.model small
.stack 64
.data

  STR1 DB "ENTER FIRST STRING HERE ->$"
  STR2 DB "ENTER SECOND STRING HERE ->$"
  STR11 DB "FIRST STRING : ->$"
  STR22 DB "SECOND STRING: ->$"

  INSTR1 DB 20 DUP("$")
  INSTR2 DB 20 DUP("$")
  NEWLINE DB 10,13,"$"
  N DB ?
  S DB ?
  MSG1 DB "BOTH STRING ARE SAME$"
  MSG2 DB "BOTH STRING ARE DIFFERENT AT POSITION $"
.code


START:  MOV AX,@DATA
        MOV DS,AX
        MOV CX, 0

        LEA SI,INSTR1
        LEA DI,INSTR2

;GET STRING
        MOV AH,09H
        LEA DX,STR1
        INT 21H
```

```
    MOV AH,0AH
    MOV DX,SI
    INT 21H

    MOV AH,09H
    LEA DX,NEWLINE
    INT 21H

    MOV AH,09H
    LEA DX,STR2
    INT 21H

    MOV AH,0AH
    MOV DX,DI
    INT 21H

    MOV AH,09H
    LEA DX,NEWLINE
    INT 21H


;PRINT THE STRING

    MOV AH,09H
    LEA DX,STR11
    INT 21H

    MOV AH,09H
    LEA DX,INSTR1+2
    INT 21H

    MOV AH,09H
    LEA DX,NEWLINE
    INT 21H

    MOV AH,09H
    LEA DX,STR22
    INT 21H

    MOV AH,09H
    LEA DX,INSTR2+2
    INT 21H

    MOV AH,09H
    LEA DX,NEWLINE
    INT 21H

;STRING COMPARISION
    MOV BX,00

    MOV BL,INSTR1+1
```

```
    MOV BH,INSTR2+1

    CMP BL,BH
    JNE L1

    ADD SI,2
    ADD DI,2

  L2:MOV BL,BYTE PTR[SI]
    CMP BYTE PTR[DI],BL
    JNE L1
    INC CX
    INC SI
    INC DI
    CMP BYTE PTR[DI],"$"
    JNE L2

    MOV AH,09H
    LEA DX,MSG1
    INT 21H

    JMP L5

  L1:
      MOV AH,09H
    LEA DX,MSG2
    INT 21H
      MOV AH,02H
      MOV BL, CX
    MOV DX, BL
    ADD DX, "0"
    INT 21H

  L5:
    MOV AH,09H
    LEA DX,NEWLINE
    INT 21H

    MOV AH,4CH
    INT 21H

END START
```

P9:

Store byte in AL into ES:[DI]. Update DI.

Example:

.data

a1 DW 5 dup(0)

LEA DI, a1
MOV AL, 12h
MOV CX, 5
REP STOSB

Write an ALO t Store word in AX into ES:[DI]. Update DI.

YourAns Here

Submission :
Submit single doc/pdf file with above answers. Course work submission through cs322.iitp@gmail.com with subject: YourrollNo_Lab4. **Due on** 14th September 2018, 5PM**.**