CSE331L_3 – Print and I/O

In this Assembly Language Programming, A single program is divided into four Segments which are -

1. Data Segment,
2. Code Segment,
3. Stack Segment, and
4. Extra Segment.

Print: Hello World in Assembly Language

```
DATA SEGMENT
     MESSAGE DB "HELLO WORLD!!!$"
ENDS
CODE SEGMENT
    ASSUME DS:DATA CS:CODE
START:
      MOV AX,DATA
      MOV DS,AX
      LEA DX,MESSAGE
      MOV AH,9
      INT 21H
      MOV AH,4CH
      INT 21H
ENDS
END START
```

Now, from these one is compulsory i.e. Code Segment if at all you don't need variable(s) for your program. if you need variable(s) for your program you will need two Segments i.e. Code Segment and Data Segment.

First Line – DATA SEGMENT
DATA SEGMENT is the starting point of the Data Segment in a Program and DATA is the name given to this segment and SEGMENT is the keyword for defining Segments, where we can declare our variables.

Next Line – MESSAGE DB "HELLO WORLD!!!$"
MESSAGE is the variable name given to a Data Type (Size) that is DB. DB stands for Define Byte and is of One byte (8 bits). In Assembly language programs, variables are defined by Data Size not its Type. Character need One Byte so to store Character or String we need DB only that don't mean DB can't hold number or numerical value. The string is given in double quotes. $ is used as NULL character in C programming, so that compiler can understand where to STOP.

Next Line – DATA ENDS
DATA ENDS is the End point of the Data Segment in a Program. We can write just ENDS But to differentiate the end of which segment it is of which we have to write the same name given to the Data Segment.

Next Line – CODE SEGMENT
CODE SEGMENT is the starting point of the Code Segment in a Program and CODE is the name given to this segment and SEGMENT is the keyword for defining Segments, where we can write the coding of the program.

Next Line – ASSUME DS:DATA CS:CODE
In this Assembly Language Programming, there are Different Registers present for Different Purpose So we have to assume DATA is the name given to Data Segment register and CODE is the name given to Code Segment register (SS,ES are used in the same way as CS,DS)

Next Line – START:
START is the label used to show the starting point of the code which is written in the Code Segment. : is used to define a label as in C programming.

Next Line – MOV AX,DATA
MOV DS,AX

After Assuming DATA and CODE Segment, still it is compulsory to initialize Data Segment to DS register. MOV is a keyword to move the second element into the first element. But we cannot move DATA Directly to DS due to MOV commands restriction, hence we move DATA to AX and then from AX to DS. AX is the first and most important register in the ALU unit. This part is also called INITIALIZATION OF DATA SEGMENT and It is important so that the Data elements or variables in the DATA Segment are made accessible. Other Segments are not needed to be initialized, only assuming is inhale.

Next Line – LEA DX,MESSAGE
MOV AH,9
INT 21H

The above three-line code is used to print the string inside the MESSAGE variable. LEA stands for Load Effective Address which is used to assign Address of variable to DX register (The same can be written like this also MOV DX,OFFSET MESSAGE both mean the same). To do input and output in Assembly Language we use Interrupts. Standard Input and Standard Output related Interrupts are found in INT 21H which is also called as DOS interrupt. It works with the value of AH register, If the Value is 9 or 9h or 9H (all means the same), That means PRINT the string whos Address is Loaded in DX register.

Next Line – MOV AH,4CH
INT 21H

The above two-line code is used to exit to dos or exit to operating system. Standard Input and Standard Output related Interrupts are found in INT 21H which is also called as DOS interrupt. It works with the value of AH register, If the Value is 4ch, that means Return to Operating System or DOS which is the End of the program.

Next Line – CODE ENDS

CODE ENDS is the End point of the Code Segment in a Program. We can write just ENDS But to differentiate the end of which segment it is of which we have to write the same name given to the Code Segment.

Last Line – END START

END START is the end of the label used to show the ending point of the code which is written in the Code Segment.

Execution of program explanation – Hello World

First save the program with HelloWorld.asm filename. No Space is allowed in the name of the Program File and extension as .asm (dot asm because it's an Assembly language program). The written Program has to be complied and Run by clicking on the RUN button on the top. The Program with No Errors will only run and could show you the desired output. Just see the screenshots below.

Note :- In this Assembly Language Programming, we have Com format and EXE format. We are Learning in EXE format only which simple then COM format to understand and Write. We can write the program in lower or upper case, but i prepare Upper Case. (this program is executed on emu8086 emulator software)

Now Try This –

```
DATA SEGMENT

     MESSAGE DB "HELLO WORLD$"

START:
      MOV AX,DATA
      MOV DS,AX
      LEA DX,MESSAGE
      MOV AH,9
      INT 21H
      MOV AH,4CH
      INT 21H

END START
```

Assembly Example 1 – Print 2 strings

```
.MODEL SMALL
.STACK 100H

.DATA
   STRING_1  DB  'I hate CSE331$'
   STRING_2  DB  'But I Love Kacchi!!!$'

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

    LEA DX, STRING_1            ; load & display the STRING_1
    MOV AH, 9
    INT 21H

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

    MOV DL, 0AH                 ; line feed
    INT 21H

    LEA DX, STRING_2            ; load & display the STRING_2
    MOV AH, 9
    INT 21H

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

  MAIN ENDP
END MAIN
```

Assembly Example 2 – Read a String and Print it

```
.MODEL SMALL
.STACK 100H

.DATA
   MSG_1  EQU  'Enter the character : $'
   MSG_2  EQU  0DH,0AH,'The given character is : $'

   PROMPT_1  DB  MSG_1
   PROMPT_2  DB  MSG_2

.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

    MOV AH, 1                   ; read a character
    INT 21H

    MOV BL, AL                  ; save the given character into BL

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

    MOV AH, 2                   ; display the character
    MOV DL, BL
    INT 21H


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

Assembly Example *3* – Read a string from user and display this string in a new line.

```
.MODEL SMALL
 .STACK 100H

 .CODE
   MAIN PROC
     MOV AH, 1                  ; read a character
     INT 21H

     MOV BL, AL                 ; save input character into BL

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

     MOV DL, 0AH                ; line feed
     INT 21H

     MOV AH, 2                  ; display the character stored in BL
     MOV DL, BL
     INT 21H

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

   MAIN ENDP
END MAIN
```

Assembly Example 4 – Read a string with gaps and print it.

```
.MODEL SMALL

.STACK 64

.DATA

        STRING DB ?

        SYM DB '$'

        INPUT_M DB 0ah,0dh,0AH,0DH,'Enter the Input',0DH,0AH,'$'

        OUTPUT_M DB 0ah,0dh,0AH,0DH, 'The output is',0DH,0AH,'$'

.CODE

MAIN PROC

        MOV AX,@DATA

        MOV DS,AX

        MOV DX,OFFSET INPUT_M    ; lea dx,input_m

        MOV AH,09

        INT 21H

        LEA SI,STRING

INPUT:  MOV AH,01

        INT 21H

        MOV [SI],AL


        INC SI

        CMP AL,0DH

        JNZ INPUT

       ; MOV AL,SYM

        MOV [SI],'$'
```

```
OUTPUT: LEA DX, OUTPUT_M             ; load and display PROMPT_2

        MOV AH, 9

        INT 21H

        MOV DL,0AH

        MOV AH,02H

        INT 21H

        MOV DX,OFFSET STRING

        MOV AH,09H

        INT 21H

        MOV AH,4CH

        INT 21H

MAIN ENDP

END MAIN
```

Assembly Example 5 – Printing string using MOV instruction

```
.Model Small
;.STACK

.DATA
      MSG1 DB 'KI!!! Kemon lage :D $'

.CODE

   MOV AX,@DATA
   MOV DS,AX

   MOV dx,OFFSET MSG1   ; LEA dx,msg1
   mov ah,09h
   int 21h

   mov ah,4ch
   int 21h

END
```

Assembly Example 6 – Print Digit from 0 – 9

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

```
.DATA
   PROMPT  DB  \'The counting from 0 to 9 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 CX, 10                  ; initialize CX

    MOV AH, 2                   ; set output function
    MOV DL, 48                  ; set DL with 0

    @LOOP:                      ; loop label
      INT 21H                   ; print character

      INC DL                    ; increment DL to next ASCII character
      DEC CX                    ; decrement CX
    JNZ @LOOP                   ; jump to label @LOOP if CX is 0

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

Assembly Example 7 – Sum of two integers

```
.MODEL SMALL
 .STACK 100H

 .DATA
    PROMPT_1  DB  \'Enter the First digit : $\'
    PROMPT_2  DB  \'Enter the Second digit : $\'
    PROMPT_3  DB  \'Sum of First and Second digit : $\'

    VALUE_1   DB  ?
    VALUE_2   DB  ?

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

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

     MOV AH, 1                  ; read a character
     INT 21H
```

```
    SUB AL, 30H                 ; save First digit in VALUE_1 in ASCII code
    MOV VALUE_1,AL

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

    MOV DL, 0AH                 ; line feed
    INT 21H

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

    MOV AH, 1                   ; read a character
    INT 21H

    SUB AL, 30H                 ; save Second digit in VALUE_2 in ASCII code
    MOV VALUE_2,AL

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

    MOV DL, 0AH                 ; line feed
    INT 21H

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

    MOV AL, VALUE_1             ; add First and Second digit
    ADD AL, VALUE_2

    ADD AL, 30H                 ; convert ASCII to DECIMAL code

    MOV AH, 2                   ; display the character
    MOV DL, AL
    INT 21H

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