~~In this~~

CSE331L_3 - Print and I/O.

In this assembly language programming. A single program is divided into four segments which are -

I. Data segment
II. Code segment
III. Stack segment
IV. Extra segment.

Print Helloworld 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 all you dont need variables for your program. if you needs variable for your program you will need two segments. i.e. Code segment and Data segments.

First line: DATA SEGMENT
DATA SEGMENT is the starting point of the Data segment in a program and ~~segment~~ SEGMENT is the keyword for defining segments. where we can declare our variables.

Next line: MESSAGE DB "HELLO WORLD!!$"
MESSAGE is a variable name given to a Data Type (size) that is the DB. DB stands for define byte and is of One byte (8 bits). In Assembly language programs, variables are defined as data size not its type. Character need oneByte so to store Characters 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. $ sign 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~~ 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 SEGMENTS
CODE SEGMENTS is the starting point of the Code segments in a program and CODE is the name given to this segment and SEGMENT is the keyword

for defining segments, where we can write coding of the program.

Next line - ASSUME DS: DATA CS: CODE

In the Assembly programming language, 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 the Data Segment to DS register. MOV is a keyword

to move the second element into first element But we cannot move DATA Directly to DS due to MOV commands restriction, hence we move DATA to AX and then AX to DS. AX is the most important register in the ALU unit. This part is also called INITIALIZATION of DATA SEGMENT and it is so important that the DATA elements or variables in the DATA Segment are made accessible. Other segments are not need to be initialized, only asuming is inhale.

Next line- LEA DX, MESSAGE
MOV AH, 9
INT 21H

The abore 3 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. To do input and output in Assembly we use interrupts. Standart Input and output related Interrupts are found in 21H which is also called DOS interrupt. It works with the value of AH registers. If the value of is

Thats 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 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 which is written in the code segment.

Execution of program explaination- Hello World.
First save the program with Hello_World.asm file name. No space is allowed in the name of the program file and extension as .asm. The written program has to be compiled and run by clicking for the RUN button on the top. The program with No errors could be run and show you the desire output:

Note:- In assembly Language programming we have com-format and EXE formal. We are Learning in exe format only which simple then com format to understand and write. We can write the program in lower to upper case but i prepare Upper case.

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
MOV AH, 9      ; load and display the string
INT 21H
MOV AH, 2
MOV DL, 0DH    ; carrage return.
MOV 21H
```

```
MOV DL, 0AH
INT 21H        ; line feed.

LEA DX, STRING_2
MOV AH, 9
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
    MOV AH, 9                ;load and display prompt_1
    INT 21H

    MOV AH, 1
    INT 21H                  ; read character.

    MOV BL, AL               ;save the given char into BL
    LEA DX, PROMPT_2
    MOV AH, 9                ; load and display PROMPT_2
    INT 21H

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

    MOV AH, 4CH
    INT 21H
  MAIN ENDP
END MAIN.
```

ASSEMBLY EXAMPLE 3 - READ A STRING FROM USER AND DISPLAY THIS STRING IN NEW LINE.

```
. MODEL SMAH
. STACK 100H
  .CODE
  0 MAIN PROC
      MOV AH,1               ;Read character
      INT 21H
```

```
MOV BL, AL      ; save input into BL
MOV AH, 2
MOV DL, ODH     ; carrage return
MOV 21H

MOV DL, OAH     ; line feed
INT 21H

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

MOV AH, 4CH
INT 21H         ; return control to DOS
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
   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
        JN2 INPUT
        ; MOV AL, SYM
        MOV [SI], '$'

OUTPUT : LEA DX, OUTPUT_M
         mov AH, 9
         INT 21H
         MOV DL, 0AH
         MOV AH, 02H
         INT 21H
         MOV DX, OFFSET. STRING2
         MOV AH, 09H
         INT 21H
         mov AH, 4CH
         INT 21H
MAIN  END P
END  MAIN.
```

Assembly example 5- Printing string using MOV instruction

```
.MODEL
; .STACK
.DATA
   MSG1 DB 'KI!!! Kemon bage :D $'

.CODE
   MOV AX, @DATA
   MOV DS, AX
   MOV dx , OFFSET MSG1.
   MOV ah, 09h
   int 21 H
   mov ah, 4ch
   int 21ff
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
     MOV DS, AX
     LEA DX, PROMPT
     MOV AH, 9
```

```
int 21H
mov cx, 10
MOV AH, 2
mov DL, 48
@ LOOP:
    INT 21H
    INC DL
    DEC CX
    JNZ @loop
  MOV AH, 4CH
  INT 21H
MAIN ENDP
END MAIN.
```

Assembly example 2 - Sum of two integers.

```
.MODEL SMALL
.STACK 100H
.DATA
PROMPT1  DB \'Enter the first digit: $\'
PROMPT_2 DB \'Enter the second digit: $\'
PROMPT_3 DB \'Sum of the 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
MOV VALUE1, AL
MOV AH, 2              ; return carriage
MOV DL, 0DH
INT 21H
MOV DL, 0AH            ; line feed
INT 21H
LEA DX, PROMPT_2       ; load and display prompt_2
MOV AH, 9              ; read a character
INT 21H
MOV AH, 1
INT 21H
SUB AL, 30H            ; save second digit
MOV VALUE_2, AL
MOV AH, 2              ; carriage return
MOV DL, 0DH
INT 21H                ; line feed
MOV DX, PROMPT_3
MOV AH, 9              ; load and display prompt3
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
```