We will need two Segments i.e. Code Segment and 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, their 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 accessable. Other Segments are not needed to be initialized, Only assuming is enhalf.

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

The above three line code is used to print String or Message present in the character Array till $  symbol which tells the compiler to stop.

Now, lets understand line by line

LEA DX,MESSAGE  in this LEA stands for LOAD EFFECTIVE ADDRESS and it loads the effective address of second element into the first element.  This same code can be interchangeably written as MOV DX, OFFSET MESSAGE where OFFSET  means effective address and MOV means move  second element into the first element.

**MOV AH,9**  
**INT 21H**

The above two line code is used to PRINT the String or Message of the address present in DX register.

Standard Input and Standard Output related Interupts 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, That means PRINT the String or Message of the address present in DX register.

**Next Line – MOV AH,1  
      INT 21H  
      MOV X,AL**

The above three line code is used to Read a Character from Console and save the value entered in variable X in its ASCII form.

Standard Input and Standard Output related Interupts are found in INT 21H which is also called as DOS interrupt. It works with the value of AH register, If the Value is 1 or 1h, That means READ a Character from Console, Echo it on screen and save the value entered in AL register.

MOV X,AL  means move value in AL register into variable X.

**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 Interupts 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.

MOV AH,4CH This same code can be interchangably written as MOV AX,4C00H where AX register is initialized with 4C00H which means 4C gets saved in AH register and 00 gets saved in AL register. different books follow different forms.

**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.

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.