**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,MSG1        
      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,MSG1  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 interchangably written as MOV DX, OFFSET MSG1 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  
      SUB AL,30H  
      MOV NUM1,AL**

The above Four line code is used to Read a Character from Console and save the value entered in variable NUM1 in its BCD form. This can be done by subtracting 30H i.e. SUB AL,30H. The value coming from Console is Basically in ASCII form. eg. When you enter 5 we see 35H,So by subtracting 30H we get back to value as 5.

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.

SUB AL,30H means subtracting 30H from AL.

MOV NUM1,AL  means move value in AL register into variable NUM1.

**Next Line – LEA DX,MSG2  
      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 i.e. for MSG2.  
**Next Line – MOV AH,1  
      INT 21H  
      SUB AL,30H  
      MOV NUM2,AL**

The above Four line code is used to Read a Character from Console and save the value entered in variable NUM2 in its BCD form.

**Next Line – ADD AL,NUM1**

The above line code is used to add the two variables and save the result in another variable.

As we know the programs work only with the instructions in the**instruction set**. Instruction **ADD** is used to add to numbers in the following permutations above. **REG** stands for **Registers** (Eg.  AX, BX, CX, DX ). **memory** stands for **Variable** or **Address**. **immediate** stands for **Numbers** or **Values.** Let us understand the meanings of the above permutations.

First permutation :- **REG**, **memory**means Register can be added with memory.

Second permutation :- **memory**, **REG**means memory can be added with Register.

Third permutation :- **REG, REG**means Register can be added with Register.

Fourth permutation :- **memory**, **immediate**means memory can be added with immediate.

Fifth permutation :- **REG**, **immediate**means Register can be added with immediate.

Note :- In the permutations above it will work only in the order mentioned above and not by interchanging the first to second and second to first.

Now, we have understood part of it to add to number we can write ADD NUM1, NUM2, But there is no permutation for **ADD m emory, memory**, Hence we have to send one number to AL or AX depending on DB or DW. AX Register is called Accumalator. and is used for holding the result of Addition in it After Addition. Now we are taking DB, So we have t0 instruction **MOV** **AL,NUM1** move NUM1 variable value to AL Register.  After moving NUM1 to AL, We can Add REG to memory, So we have **ADD** **AL,NUM2** or (We can Add memory to REG , So we have **ADD NUM2,AL**) Both are allowed as per permutations so use one from two. Now, the Resultant Value is saved in Accumalator AL for DB and AX for DW, So move Resultant value to RESULT variable by instruction **MOV RESULT,AL**

**Next Line – MOV AH,0  
     AAA**

 The first line is just to clear the unwanted garbage value present in AH  register as the AH register is going to be used later. The above two line code is used to corrects result in AH and AL after addition when working with BCD values . AAA means ASCII Adjust after Addition. **AAA** Instruction has NO Operands or values attached to it. If we are adding two BCD numbers the Result is saved in AL register in HEXadecimal form. What AAA exactly does is it converts the result into BCD form and **first** digit is saved in AH register and **second**digit is saved in AL register.

**Next Line – ADD AH,30H  
      ADD AL,30H**

Since the Result of Multiplication is in AH and AL register in BCD form After using AAM. Now we want to print the result on screen or console. the BCD form value will not show us the Result But will print the Coresponding Ascii Codes of the number, Hence By adding 30H to BCD will Convert it to ASCII code which will print the digit (number) on screen.

**Next Line – MOV BX,AX**

Since the AH register is used again and again. We cannot lose the result in AH and AL register, So to save the to be printed value in BH and BL register. By moving AX to BX.

**Next Line – LEA DX,MSG3  
      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 i.e. for MSG3.

**Next Line – MOV AH,2  
      MOV DL,BH  
      INT 21H**

The above Three line code is used to Write a Character on Console present in BH 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 2 or 2h, That means WRITE a Character on Console present in DL register hence the value to be printed is moved to DL register. Here we are printing BH register.

**Next Line – MOV AH,2  
      MOV DL,BL  
      INT 21H**

The above Three line code is used to Write a Character on Console present in BL 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 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.

**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.