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. 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.

**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 accessable. Other Segments are not needed to be initialized, Only assuming is enough.

**Next Line – MOV AL,NUM1  
ADD AL,NUM2  
MOV RESULT,AL**

The above three 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 memory, 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,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.