**Logo, company name

Description automatically generated**

**JK Lakshmipat University, Jaipur**

**Institute of Engineering and Technology (IET)**

**CS1117 Computer Organization and System**

**Lab File**

SUBMITTED BY

**Akshat singh**

22018

SUBMITTED TO

**Mr. Divanshu Jain**

**EXPERIMENT – 1**

**OBJECTIVE**

Write a program for 16-bit addition on 8086 Emulator.

**THEORY**

**The code used is in RED**

**The registers are in GREEN**

**The statements are in Blue**

**The variables are in Purple**

**The DATA SEGMENT and values are in ORANGE**

# We start with DATA SEGMENT here we write variables, values and instructions which we want to execute.

**DW = Define Word**

**X = variable**

**Z = variable**

**SUM = Addition of the values**

**Carry = To carry the carry which is 1**

**DATA ENDS = The data table ended**

**--------------------------------EXPLAINING DATA SEGMENT-------------------**

1. **We take (X) as a variable, and we take (DW) as we are allocating space for our value (0FFAAH).**

**; - We are doing the same with (Z) taking it as a variable and (DW) for space for our value (0AAFAH)**

1. **Now we type (SUM DW?). To add both the values which are stored in (DW).**

**; - But we also use ‘?’ argument because we don’t know what the value of (SUM DW) YET is. We use it because the data can be anything and if expect to read the data, we must write ‘?’**

1. **Now there will be a carry after the (SUM DW ?) instruction and we will use (CARRY DB 00H) Function to relocate 00H into DB thus saving the carry 1.**
2. **(DATA ENDS) it’s a basic instruction to End the (DATA SEGMENT)**

**---------------------------------------- CODE SEGMENT-------------------------------**

1. **CODE SEGMENT : - We are using the ASSUME argument to tell the assembler which segment register we are going to use to access a segment**

**; - By writing (ASSUME CS:CODE, DS : DATA) we are specifying (CS) as CODE , And DS as DATA. CS is code segment and DS is data segment.**

1. **START :**

**; - This is a simple argument to tell the assembler that the code starts from here.**

**-------------------------------------CODE TO ADD---------------------------------------**

1. **We start by writing the (MOV) instruction to move the (,DATA) into (AX).**

**Which is the primary accumulator, and it is used for arithmetic instructions. = (MOV AX,DATA)**

1. **Next, we use (MOV) again to move the DATA which is now in AX to (DS) = (MOV DS,AX)**

**; - As DS reserves the number of bytes in the memory space. Hence the DATA is now stored in DS)**

1. **Now we can start adding the variables we specified earlier X, Z. We are going to use AX register as AX is used to store data over 255 values and our 16-bit number is more than 255 so we need to use the AX register. = (MOV AX,X) and (ADD AX,Z)**
2. **Now we use MOV instruction to move the value of X into AX (MOV AX,X)**
3. **Now we are going to use ADD instruction to add the value of AX with Z and as X’s value is stored in AX. We are going to write (ADD AX,Z)**
4. **Now to get the CARRY we must JNC SKIP and INC CARRY Otherwise our carry ‘1’ will not be added (We can also only use INC CARRY to get our carry)**
5. **Next, we’ll use the SKIP : MOV to skip the next MOV argument as we are going to use the (SUM , AX) to add both the variables X, Z**
6. **But now we are not going to MOV (AX) but rather we are going to use the (AH) because the value is now much greater than the variables X,Y**

**; - But we are also going to use the 4CH argument otherwise the command (MOV AH,) will not work we use 4CH to store hexadecimal value.**

1. **Next, we are going to have to use the command INT 21H otherwise our program will not work. As this command is a function dispatcher. It will execute the commands basically.**
2. **Now at last we end our code with CODE ENDS AND END START command. To tell the assembler to stop executing the commands.**

**CODE**

**DATA SEGMENT**

**X DW 0FFAAH**

**Z DW 0AAFAH**

**SUM DW ?**

**CARRY DB 00H**

**DATA ENDS**

**CODE SEGMENT**

**ASSUME CS:CODE, DS:DATA**

**START:**

**MOV AX,DATA**

**MOV DS,AX**

**MOV AX,X**

**ADD AX,Z**

**JNC SKIP**

**INC CARRY**

**SKIP: MOV SUM, AX**

**MOV AH,4CH**

**INT 21H**

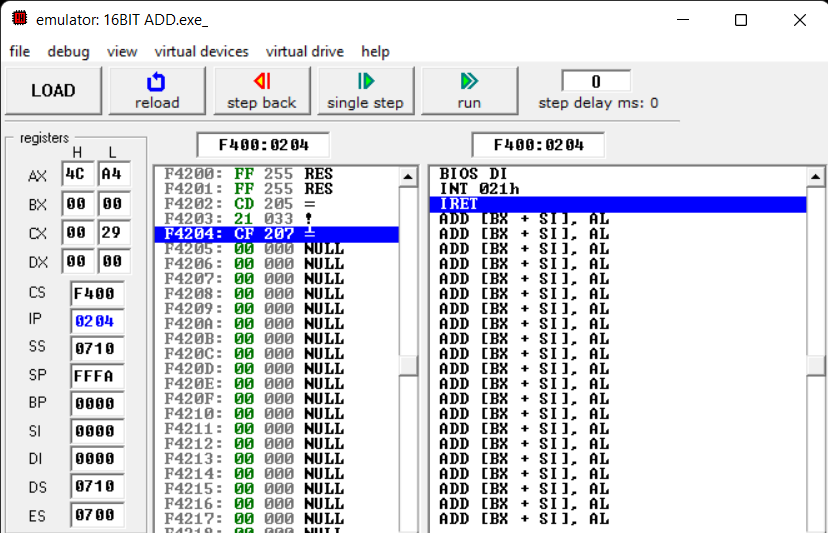
**; INT 03H**

**CODE ENDS**

**END START**

**RESULTS/OUTPUT**

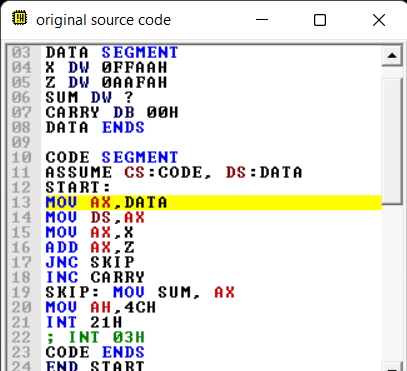
1. **EMULATOR RUNNED**

****

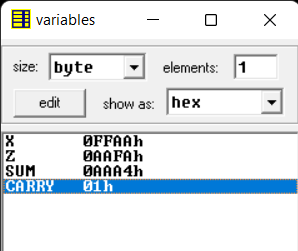
1. **FLAGS**

****

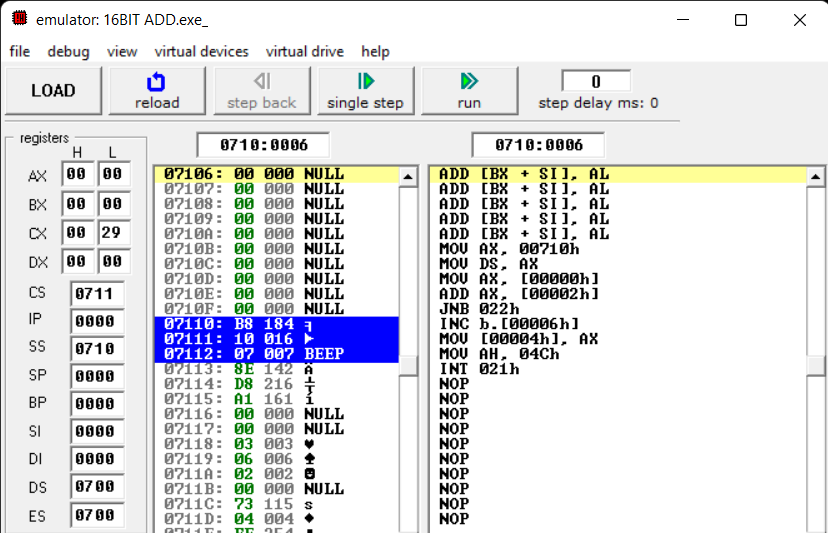
1. **SOURCE**

****

1. **VARIABLES**

****

1. **EMULATOR NOT RUNNED**

****