**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 – 2.2**

**OBJECTIVE**

**Write a program for 8-bit unsigned subtraction 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.

**DB = Data byte**

**X = variable**

**Z = variable**

**DIFF = Difference of the values**

**BORROW = 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 (DB) as we are allocating space for our value (0CDH).**

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

1. **Now we type (DIFF DB?). To subtract both the values which are stored in (DB).**

**; - But we also use ‘?’ argument because we do not know what the value of (DIFF DB) is YET. 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 BORROW after the (BORROW DB ?) instruction and we will use (BORROW DB 00H) Function to relocate 00H into DB thus saving the BORROW 1 .**
2. **(DATA ENDS) it is 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 SUBTRACT-----------------------------**

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 AL register as AL is used to store data under 255 values and our 8-bit number is less than 255 so we don’t need to use the AX register. = (MOV AL,X) and (ADD AL,Z)**
2. **Now we use MOV instruction to move the value of X into AL (MOV AL,X)**
3. **Now we are going to use SUB instruction to subtract the value of AL with Z and as X’s value is stored in AL. We are going to write (SUB AL,Z)**
4. **Now to get the BORROW we must JNC SKIP, and INC BORROW Otherwise our BORROW ‘1’ will not be SAVED (We can also only use INC BORROW to get our BORROW)**
5. **Next, we’ll use the SKIP : MOV to skip the next MOV argument as we are going to use the (DIFF, AL) to SUBTRACT both the variables X, Z.**
6. **But now we are not going to MOV (AL) 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 DB 0CDH**

**Z DB 0FBH**

**DIFF DB ?**

**BORROW DB 00H**

**DATA ENDS**

**CODE SEGMENT**

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

**START:**

**MOV AX,DATA**

**MOV DS,AX**

**MOV AL,X**

**SUB AL,Z**

**JNB SKIP**

**INC BORROW**

**SKIP: MOV DIFF,AL**

**MOV AH,4CH**

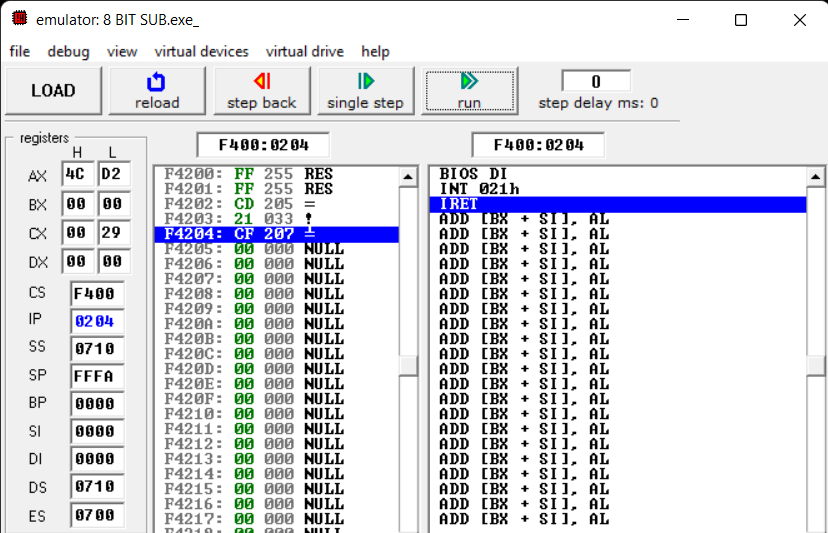
**INT 21H**

**CODE ENDS**

**END START**

**RESULTS/OUTPUT**

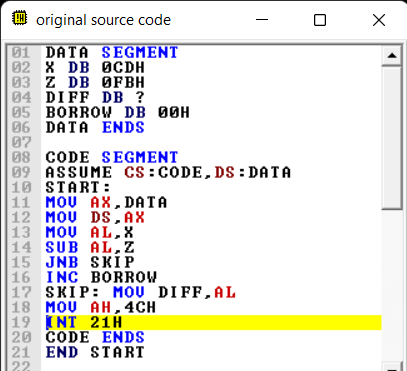
1. **EMULATOR RUNNED**

****

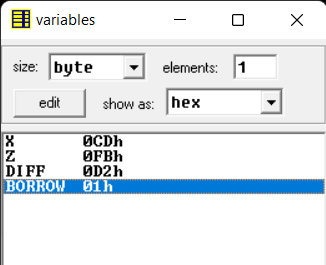
1. **FLAGS**

****

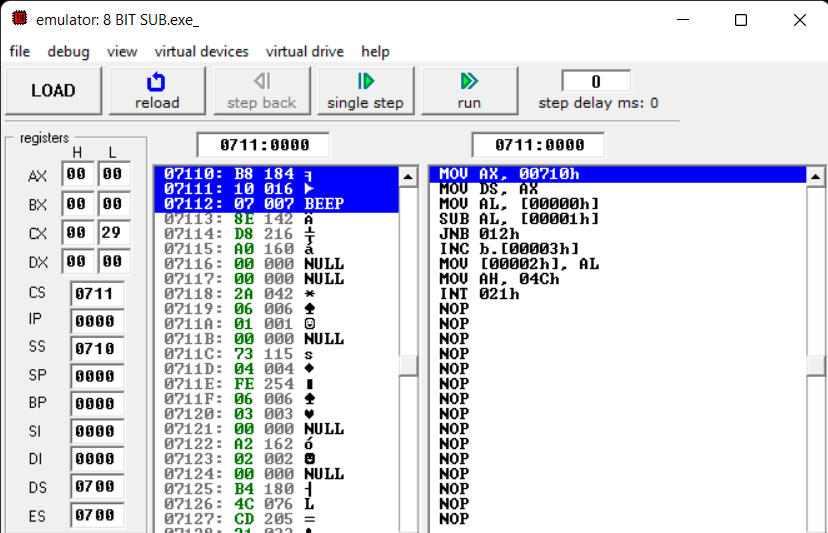
1. **SOURCE**

****

1. **VARIABLES**

****

1. **EMULATOR NOT RUNNED**

****