**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 – 3**

**OBJECTIVE**

**Write an ALP to add a series of 5 bytes stored in the memory from locations  20,000H to 20,004H. Store the result immediately after the series.**

**THEORY**

**The code used is in RED**

**The registers are in BLUE**

**The statements are in GREEN**

**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 = Define Byte**

**A = variable**

**B = variable**

**C = Variable**

**D = Variable**

**E = 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 (A) as a variable, and we take (DB) as we are allocating space for our value (0F9H).**

**; - We are doing the same with (B) taking it as a variable and (DB) for space for our value (0A0H)**

**; - We are doing the same with (C) taking it as a variable and (DB) for space for our value (0F0H)**

**; - We are doing the same with (D) taking it as a variable and (DB) for space for our value (0B3H)**

**; - We are doing the same with (E) taking it as a variable and (DB) for space for our value (0C1H)**

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

**; - But we also use ‘?’ argument because we don’t know what the value of (SUM DB) 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 DB ?) 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 A,B,C,D,E. We are going to use DL,BH,AH,CH,CL registers as these are used to store data under 255 values and our 8-bit number is less than 255 so we can use any of the registers specified earlier. = {(MOV DL,A), (ADD BH,B), (MOV AH,C), (MOV CH,D), (MOV CL,E)}**
2. **Now we use MOV instruction to move the value of A into DL (MOV DL,A)**

**;- We are going to do the same will all the values we have**

**WE use MOV instruction to move the value of variables into (DL,BH,AH,CH,CL)**

**{(MOV BH,B),(MOV AH,C),(MOV CH,C),(MOV CL)}**

1. **Now I’ll be MOVING the values stored in different registers earlier**

**By using the codes**

**(MOV [2000H],DL), (MOV [2001H],BH) ,** (**MOV [2002H],DH),(MOV [2003H],CH),(MOV [2004H],CL)**

1. **Now we use the CL register and move the value 05 into it to tell the LOOP function to LOOP FIVE times. (MOV CL,05)**
2. **Now we use the SI register to point to the address we want to start our loop from which in this case is 2000H (MOV SI,2000H)**
3. **Now to save our carry we are going to move the value 00H into the AX register to save the carry which will become 1. (MOV AX,00H)**
4. **Now we are going add the loop by moving the value of SI into BL**

**:- As the SI register stores the value which we pointed to earlier in line 7 and BL register will be able to start the loop from our value 2000H which was stored in SI. (ADDLOOP:MOV BL,[SI])**

1. **Now we use the AL register to store the data from BL so that we can later SUM all the values stored in BL from our 9th line. (ADD AL,BL)**
2. **Now we use JNC NEXT to tell the program to Jump to the next argument if there is no carry. (JNC NEXT)**
3. **Now we are going to tell the program to increment to the carry to 1 into AH register (INC AH) with NEXT:INC SI we are also incrementing the SI and our carry will come to SI from last line.**
4. **Now we are also going to decrement the CL by 1 to stop the loop after 4 times. (DEC CL)**
5. **Now we are telling the program to add the loop if there is no zero to save us the carry’s generated (JNZ ADDLOOP)**
6. **Now are telling the program to store the final value of AL into SUM**

**:- So that we can clearly check the output (MOV SUM,AL)**

1. **Now we move the carry we saved into AH into CARRY.**

**:- So that we can clearly check the carry’s we got in the output**

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.**

**----------------------------------------Theory Ends----------------------------------------**

**CODE**

**DATA SEGMENT**

**A DB 0F9H**

**B DB 0A0H**

**C DB 0F0H**

**D DB 0B3H**

**E DB 0C1H**

**SUM DB ?**

**CARRY DB 00H**

**DATA ENDS**

**CODE SEGMENT**

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

**START:**

**MOV AX,DATA**

**MOV DS,AX**

**MOV DL,A**

**MOV BH,B**

**MOV DH,C**

**MOV CH,D**

**MOV CL,E**

**MOV [2000H],DL**

**MOV [2001H],BH**

**MOV [2002H],DH**

**MOV [2003H],CH**

**MOV [2004H],CL**

**MOV CL,05**

**MOV SI,2000H**

**MOV AX,00H**

**ADDLOOP:MOV BL,[SI]**

**ADD AL,BL**

**JNC NEXT**

**INC AH**

**NEXT: INC SI**

**DEC CL**

**JNZ ADDLOOP**

**MOV SUM,AL**

**MOV CARRY,AH**

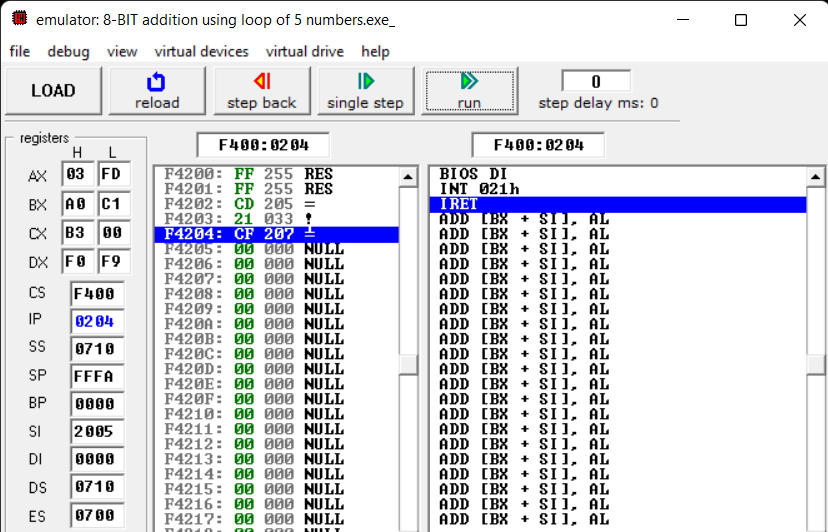
**INT 21H**

**CODE ENDS**

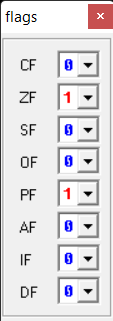
**END START**

**RESULTS/OUTPUT**

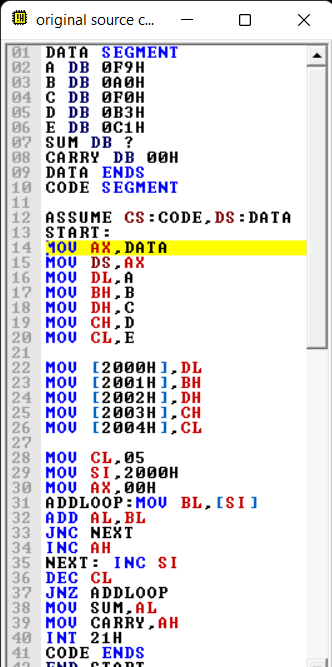
1. **EMULATOR RUNNED**

****

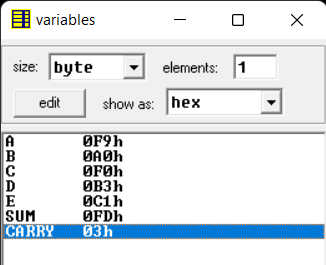
1. **FLAGS**

****

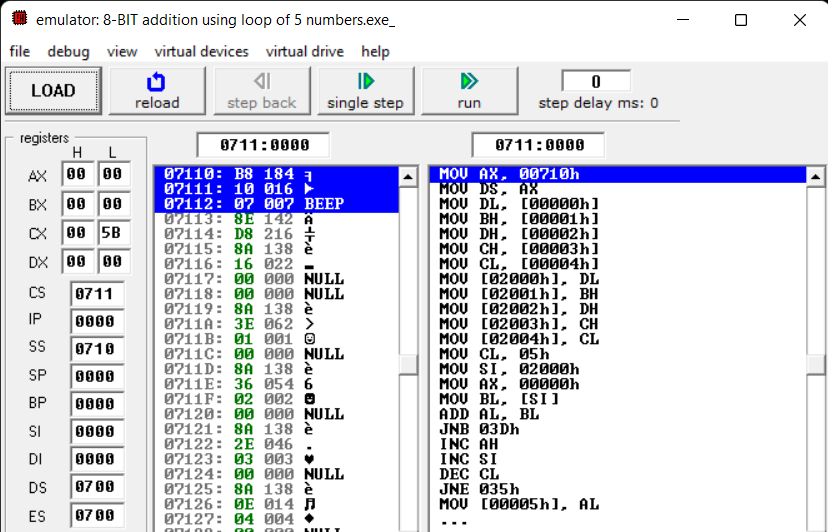
1. **SOURCE**

****

1. **VARIABLES**

****

1. **EMULATOR NOT RUNNED**

****