Fall Semester 2021-2022

Microprocessor and Interfacing

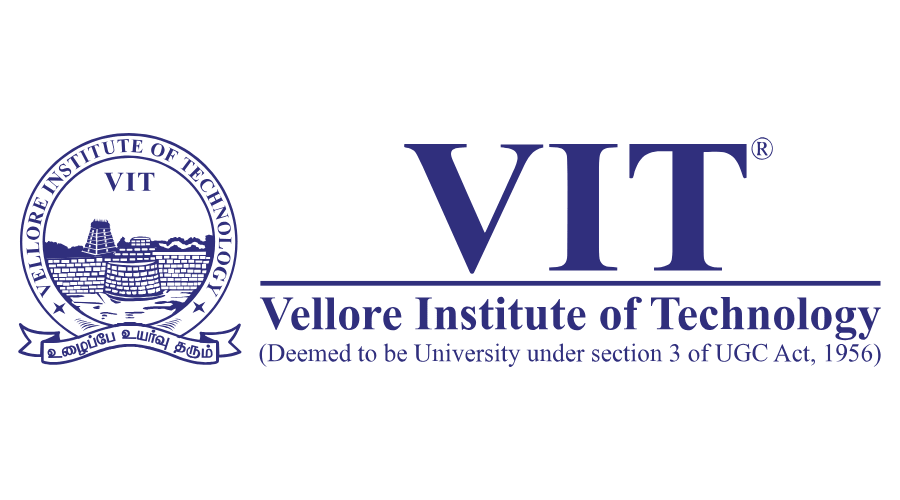
Lab Report

Digital Assignment-2

Experiment No: 3 Task No: 2

Course Code: CSE2006

Slot: L7+L8



Submitted By: Alokam Nikhitha

Reg. Numb: 19BCE2555

Submitted To: Dr. Abdul Majed KK

**EXPERIMENT 3:**

**Programs involving Arithmetic Operation of Signed Numbers**

**Aim:**

**A. 16 Bit multiplication for signed numbers:**

1. **Write an Assembly Language Programme (ALP) to multiply 8 bit signed numbers.**
2. **Write an Assembly Language Programme (ALP) to multiply 16 bit signed numbers.**

**B.16 Bit Division for signed numbers:**

1. **Write an Assembly Language Programme (ALP) to divide 16 bit by 8 bit signed numbers.**
2. **Write an Assembly Language Programme (ALP) to divide 32 bit by 16 bit signed numbers.**

**C.Sum of N numbers:**

1. **Write a program to find the sum of N numbers**

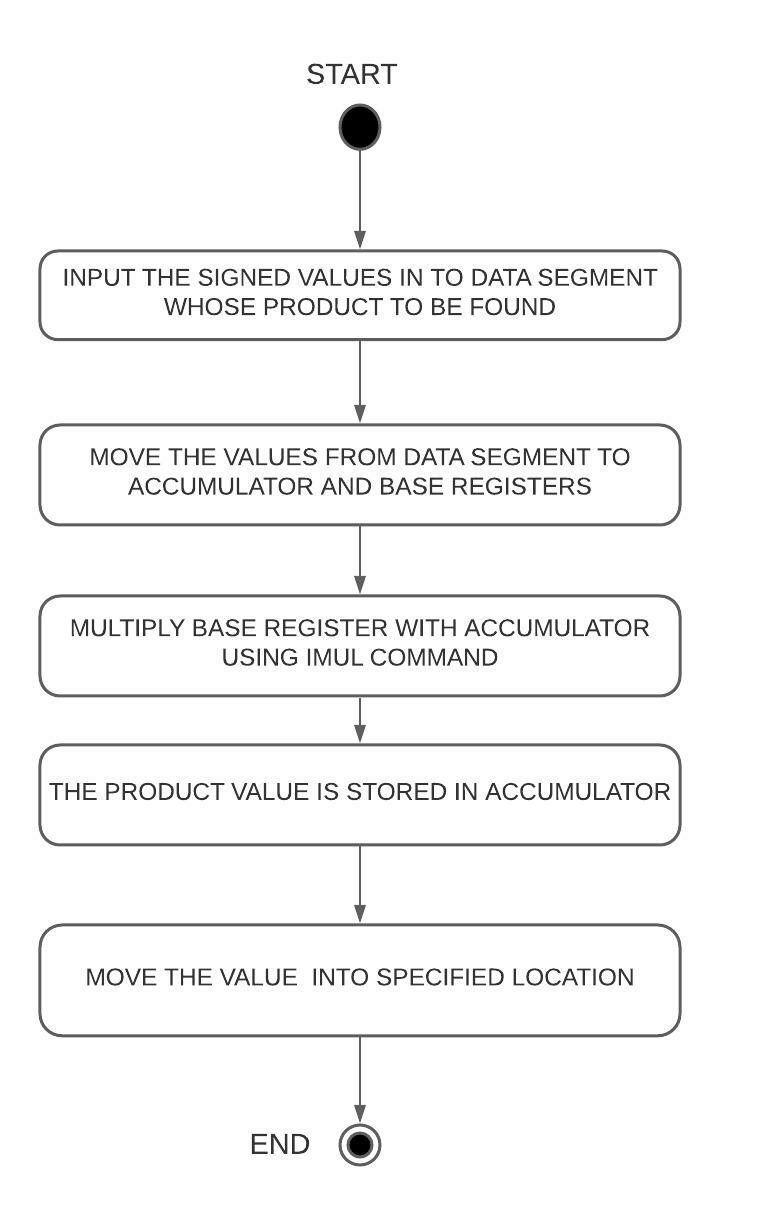
**Tool Used: emu8086 simulator**

**A. Multiplication for signed numbers**

**Algorithm:**

* **Input the values whose product is to be found into the Data Register**
* **Move the first value to accumulator register (AX).**
* **Move the second value(Signed value) to Base register (BX).**
* **Multiply the base register (IMUL command) with the accumulator.**
* **The above step will store the updated value in accumulator itself.**
* **Move the contents of accumulator to desired memory location.**
* **Halt the overall process.**

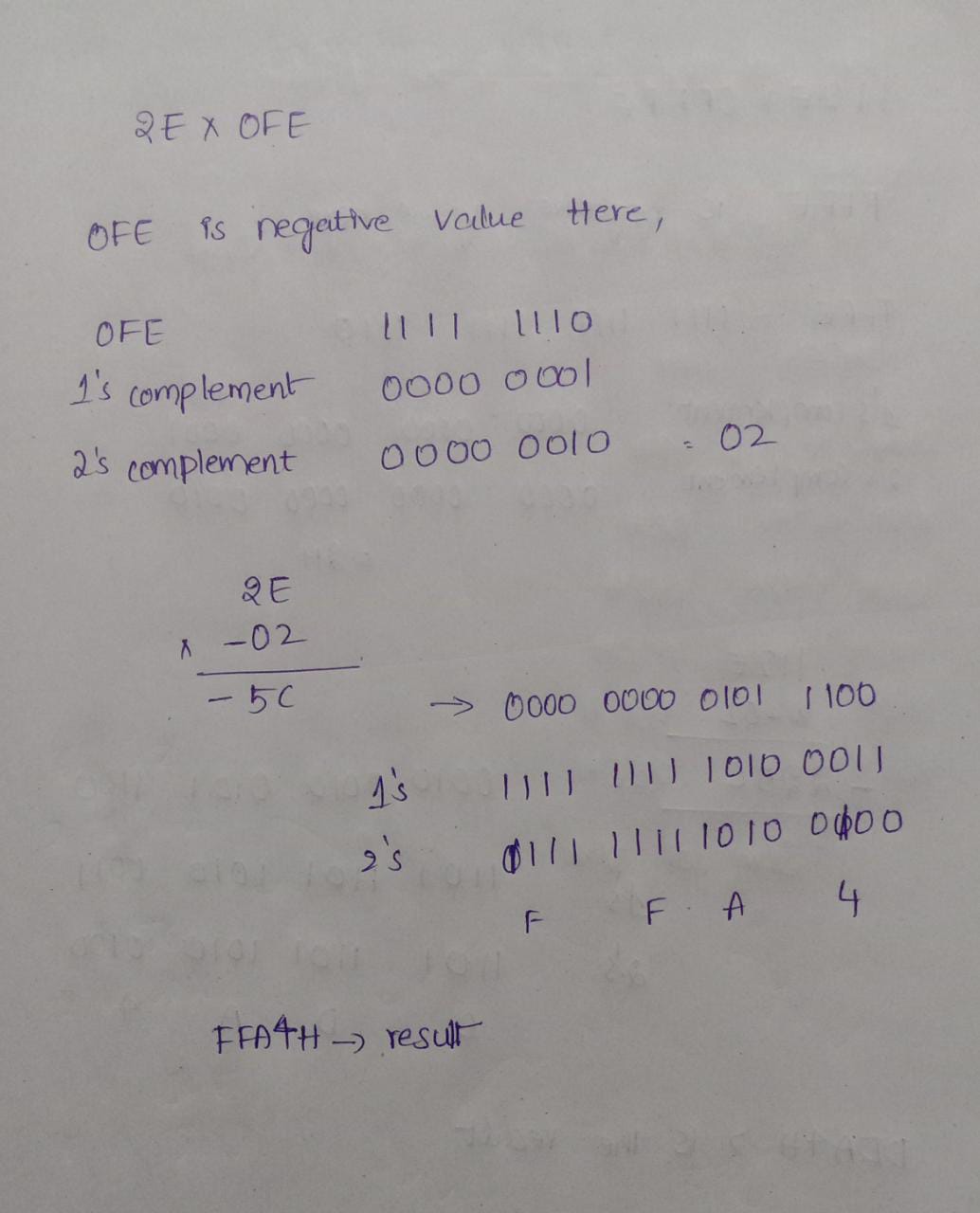
**Flow Chart:**

****

**1)Write an Assembly Language Programme (ALP) to multiply 8 bit signed numbers.**

**Design and Calculations:**

For 8-bit signed multiplication we need to use AL and BL registers from the Data Segment. We use IMUL command here to execute the signed multiplication of numbers.The data is stored in N1 and N2 and the values are moved to Accumulator(AL) and Base Registers(BL) and the Base Register(BL) is multiplied with Accumulator(AL)

****

**Program Code:**

**ASSUME CS:CODE DS:DATA**

**DATA SEGMENT**

**N1 DB 2EH**

**N2 DB 0FEH**

**ANS DW ?**

**DATA ENDS**

**CODE SEGMENT**

**START:**

**MOV AX, DATA**

**MOV DS,AX**

**MOV AL,N1**

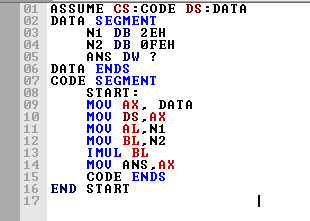
**MOV BL,N2**

**IMUL BL**

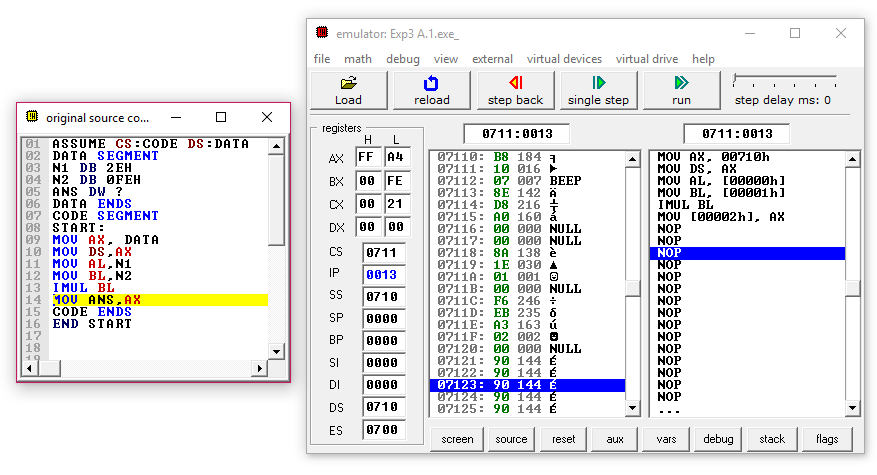
**MOV ANS,AX**

**CODE ENDS**

**END START**

****

**Output:**

****

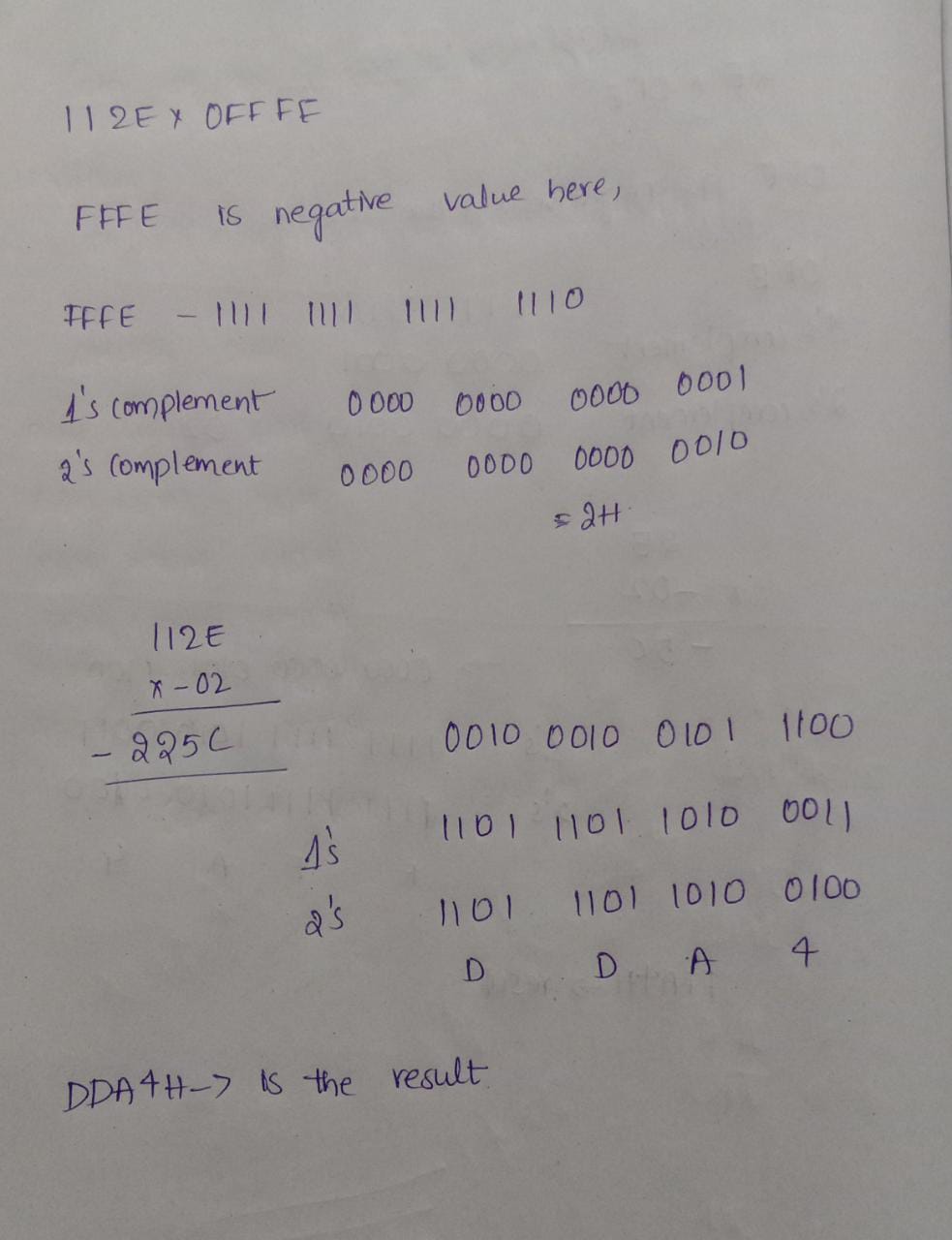
**Result and Inference:**

**-The Result FFA4h is same as we calculated and is stored in Accumulator**

**2) Write an Assembly Language Programme (ALP) to multiply 16 bit signed numbers.**

**Design and Calculations:**

For 16-bit signed multiplication we need to use AX and BX registers from the Data Segment. We use IMUL command here to execute the signed multiplication of numbers.The data is stored in N1 and N2 and the values are moved to Accumulator(AX) and Base Registers(BX) and the Base Register(BX) is multiplied with Accumulator(AX)

****

**Program Code:**

**ASSUME CS:CODE DS:DATA**

**DATA SEGMENT**

**N1 DW 112EH**

**N2 DW 0FFFEH**

**ANS DW ?**

**DATA ENDS**

**CODE SEGMENT**

**START:**

**MOV AX, DATA**

**MOV DS,AX**

**MOV AX,N1**

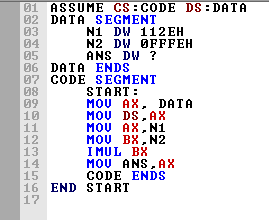
**MOV BX,N2**

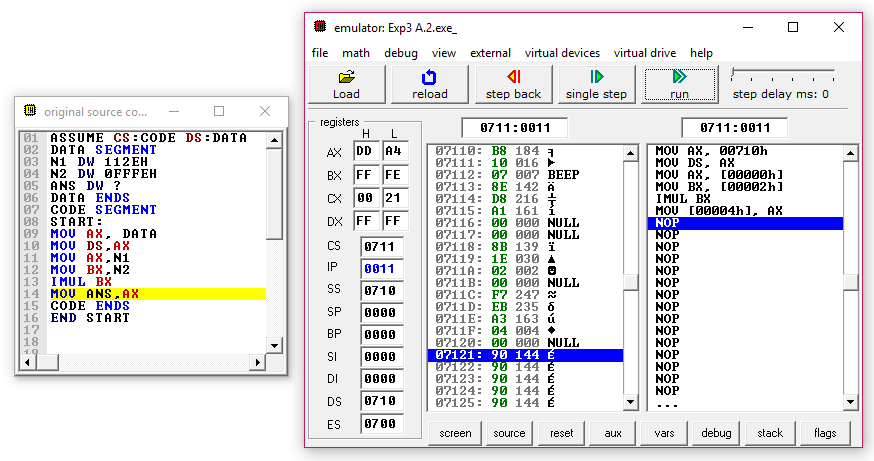
**IMUL BX**

**MOV ANS,AX**

**CODE ENDS**

**END START**

****

**Output:**

**Result and Inference:**

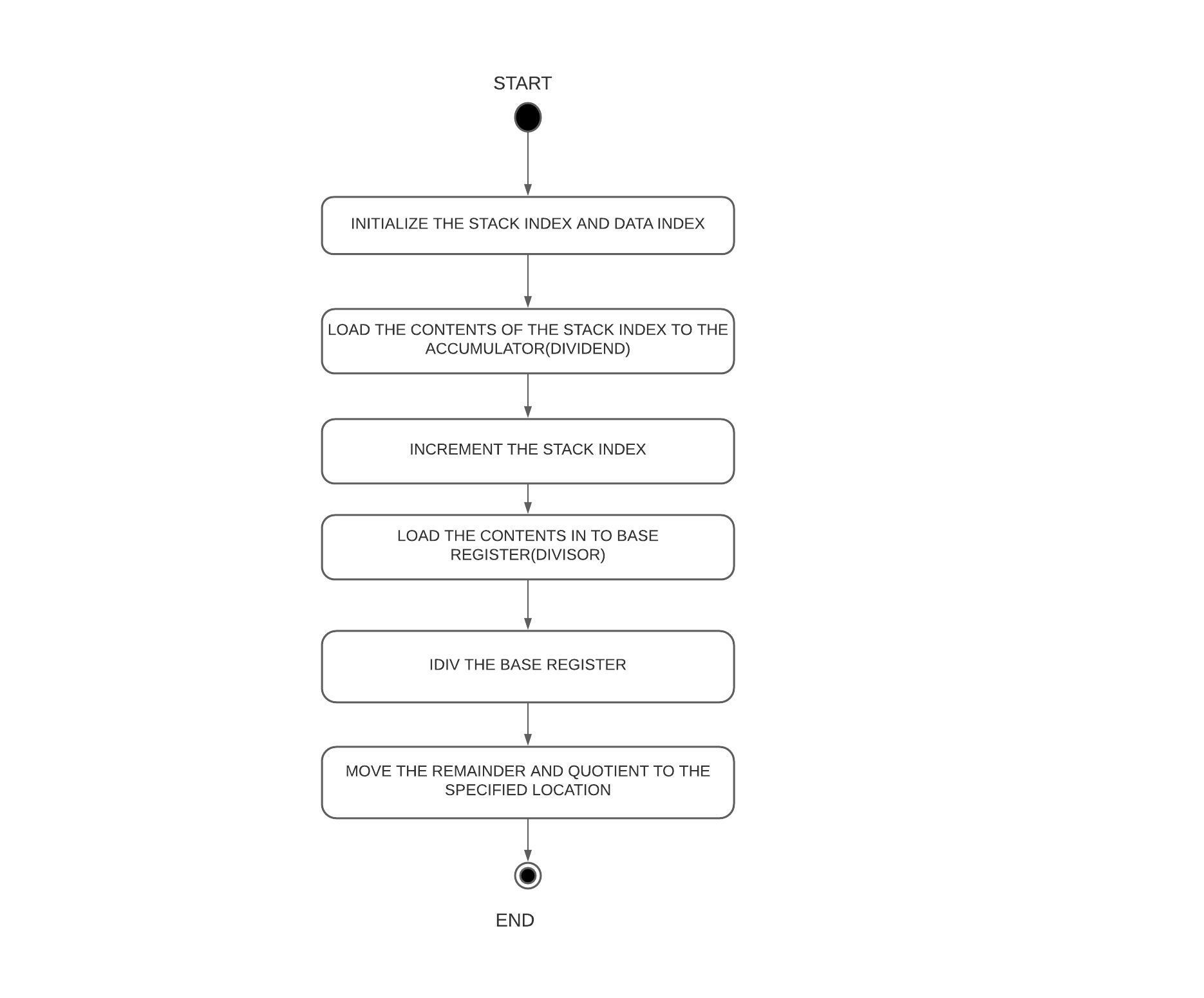
**-The Result obtained is same as the calculated value(DDA4h) in Accumulator**

**B.Division for signed numbers**

**Algorithm:**

* **Initialize the stack index(SI) and data index(DI) to point at the location where data is to be fetched from and is to be stored at.**
* **Load the values in stack index to Accumulator(Dividend)**
* **Increment stack index by 2.**
* **Load the values in stack index to accumulator.(Divisor)**
* **Divide the base register using the IDIV command.**
* **Move the Remainder and Quotient to the Memory Location by Incrementing the Stack Index**

**Flow Chart:**

****

**1)Write an Assembly Language Programme (ALP) to divide 16 bit by 8 bit signed numbers.**

**Design and Calculations:**

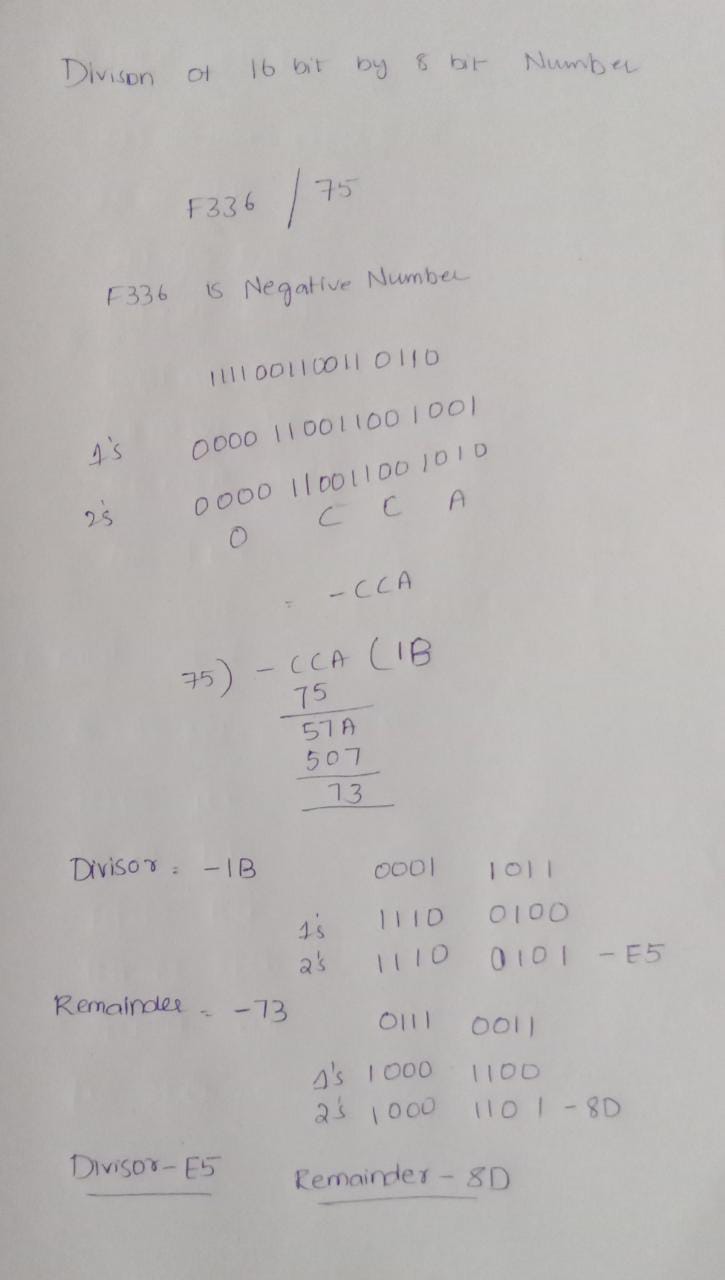
**Initialize the Data Segment(1000h) stack index(SI)[0100h] and data index(DI)[0100h] to point at the location and move the Dividend(F336h )and divisor(75h) to Memory by incrementing the SI value .**

**Load the values in stack index to Accumulator(Dividend)**

**Increment stack index by 2 and Load the values in stack index to Base Register.(Divisor)**

**Divide the base register using the IDIV command.**

**Move the Remainder and Quotient to the Memory Location by Incrementing the Stack Index**

****

**Program Code:**

**MOV AX, 1000H;**

**MOV DS, AX;**

**MOV DI, 0100H;**

**MOV SI, 0100H**

**MOV AX, 0036H;**

**MOV [DI], AX;**

**INC DI;**

**MOV AX, 00F3H;**

**MOV [DI], AX;**

**INC DI;**

**MOV AX, 0075H;**

**MOV [DI], AX;**

**MOV AX, [SI];**

**ADD SI, 0002H;**

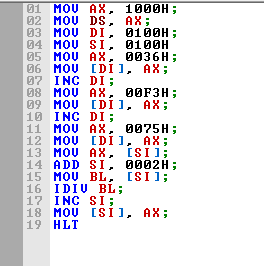
**MOV BL, [SI];**

**IDIV BL;**

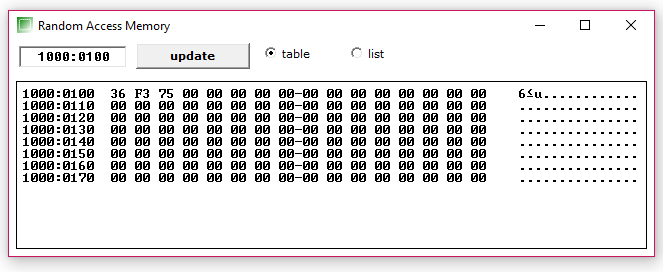
**INC SI;**

**MOV [SI], AX;**

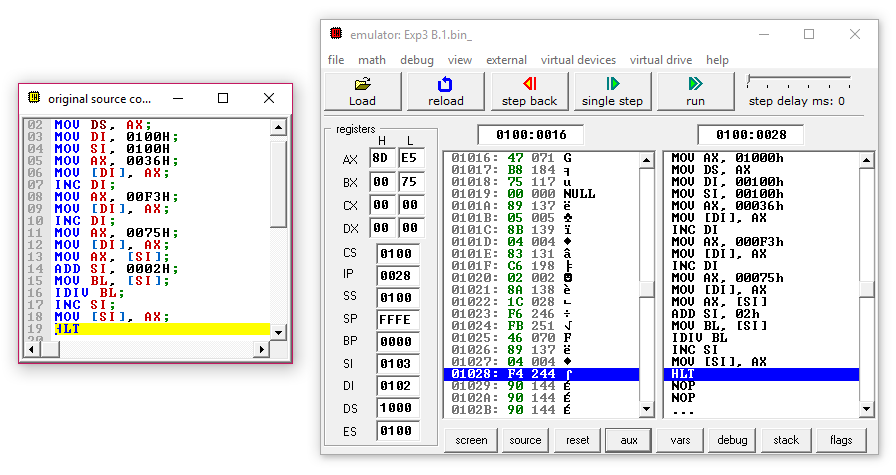
**HLT**

****

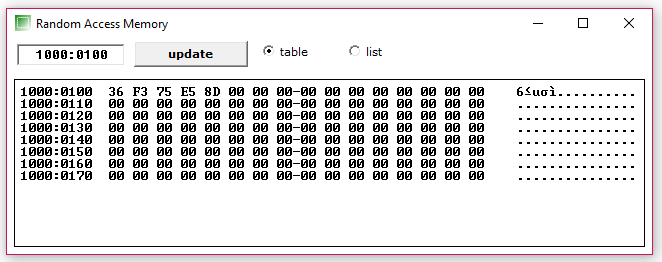
**Memory before Divison :**

****

**Output:**

****

**Memory after Divison :**

****

**Result and Inference:**

**-The Quotient E5h(-1Bh) is stored in [1000:0103]**

**-The Remainder 8Dh(-73h) is stored in[1000:0104]**

**2)Write an Assembly Language Programme (ALP) to divide 32 bit by 16 bit signed numbers.**

**Design and Calculations:**

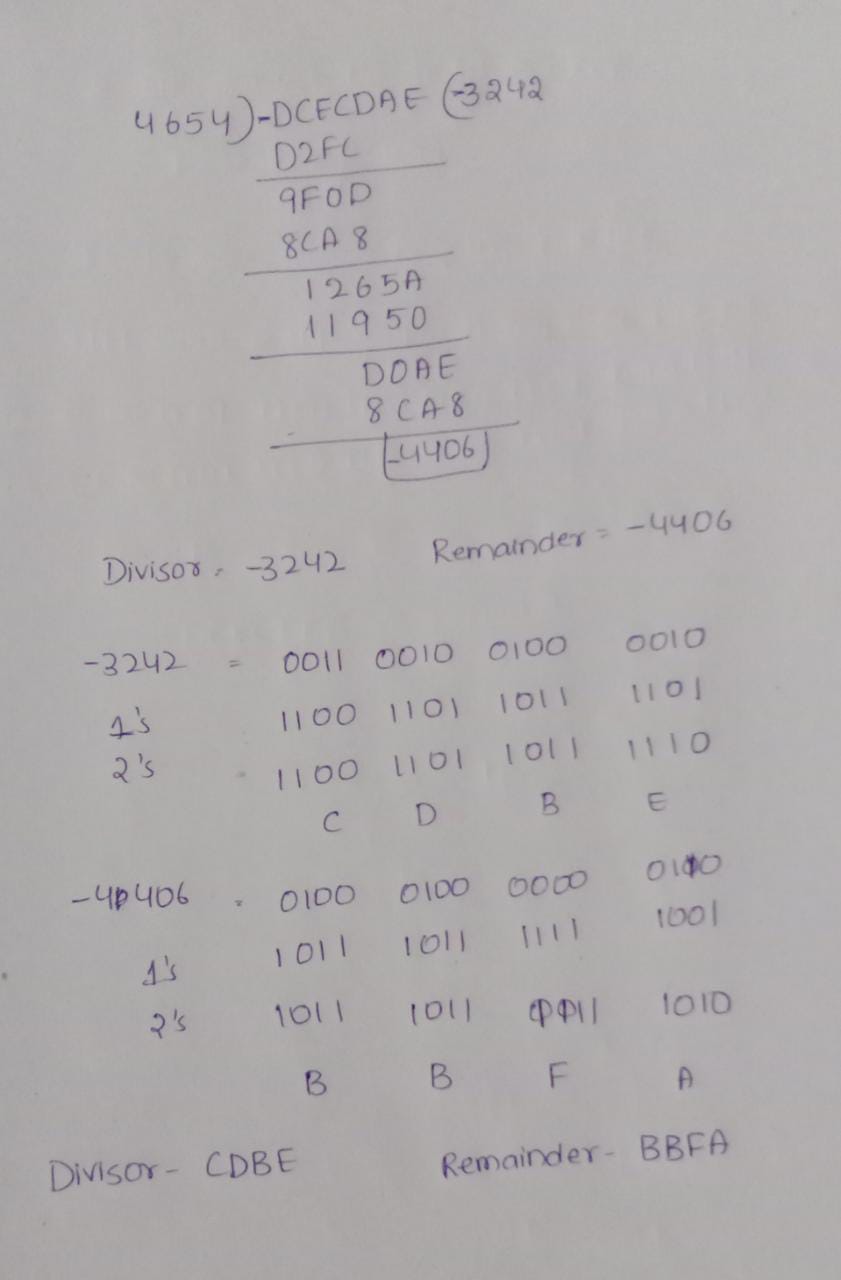
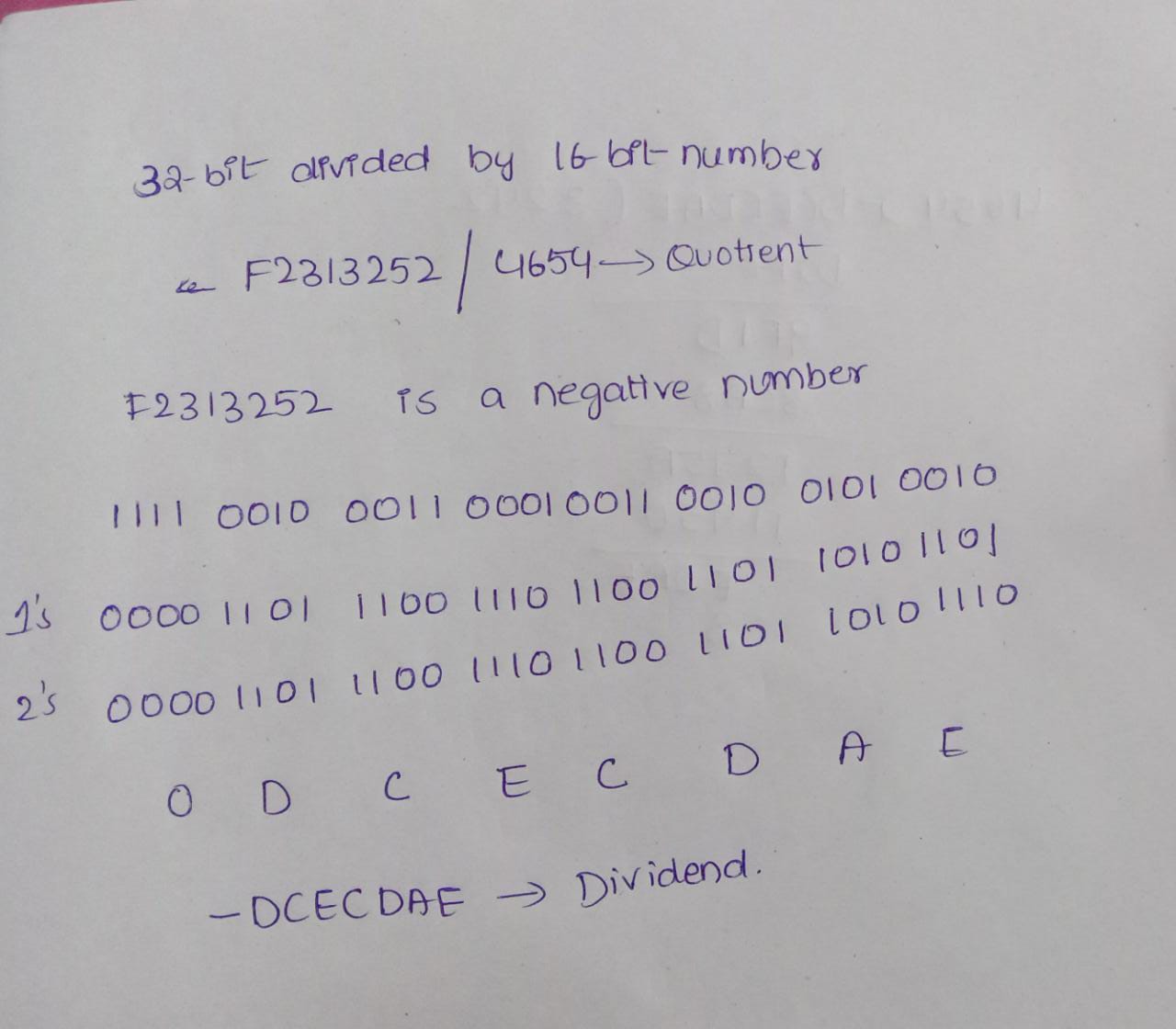
**Initialize the Data Segment(2000h) stack index(SI)[0100h] and data index(DI)[0100h] to point at the location and move the Dividend(F2313252h )and divisor(4654h) to Memory by incrementing the SI value .**

**Load the values in stack index to Accumulator and Data Register(DX)(Dividend)**

**Increment stack index by 2 and Load the values in stack index to Base Register.(Divisor)**

**Divide the base register using the IDIV command.**

**Move the Remainder and Quotient to the Memory Location by Incrementing the Stack Index**

****

**Program Code:**

**MOV AX, 2000H;**

**MOV DS, AX;**

**MOV DI, 0100H;**

**MOV SI, 0100H**

**MOV AX, 0052H;**

**MOV [DI], AX;**

**INC DI;**

**MOV AX, 0032H;**

**MOV [DI], AX;**

**INC DI;**

**MOV AX, 0031H;**

**MOV [DI], AX;**

**INC DI;**

**MOV AX, 00F2H;**

**MOV [DI], AX;**

**INC DI;**

**MOV AX, 0054H;**

**MOV [DI], AX;**

**INC DI;**

**MOV AX, 0046H;**

**MOV [DI], AX;**

**MOV AX, [SI];**

**ADD SI, 0002H;**

**MOV DX, [SI];**

**ADD SI, 0002H;**

**MOV BX, [SI];**

**IDIV BX;**

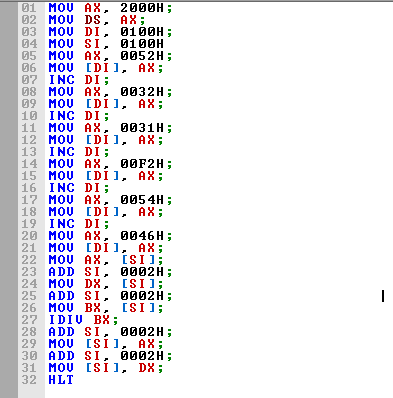
**ADD SI, 0002H;**

**MOV [SI], AX;**

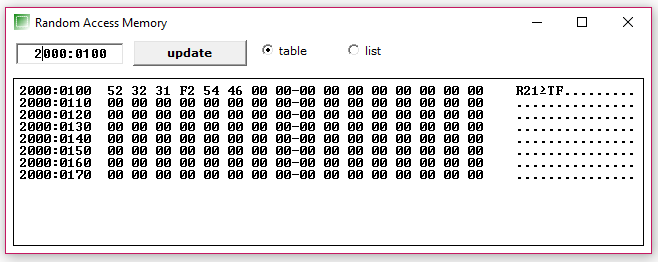
**ADD SI, 0002H;**

**MOV [SI], DX;**

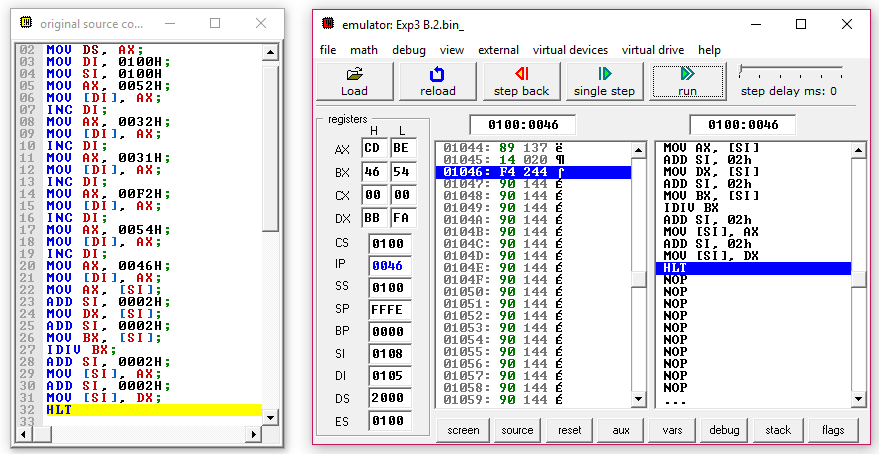
**HLT**

****

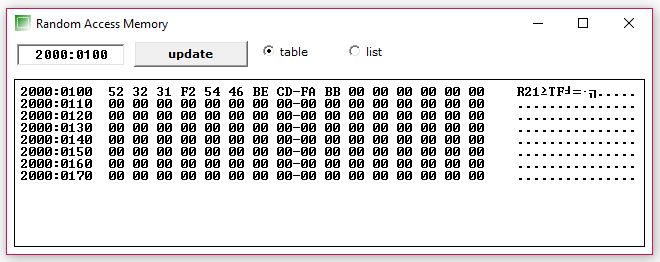
**Memory before Division:**

****

**Output:**

****

**Memory After Divison:**

****

**Result and Inference:**

**-The Quotient CDBEh(-3242h) is stored in Memory Location [2000:0107][2000:0106]**

**-The Remainder BBFAh(-4406h) is stored in Memory Location [2000:0109][2000:0108]**

**C. Sum of N numbers**

**ALGORITHM**

**-Input the Array of Values in Data Segment**

**-Input the Size of the Array in Data segment**

**-Move all the values of the Array to a specified Memory Location(SI) in Code Segment**

**-Move the size of Array from Data segment in to Counter Register**

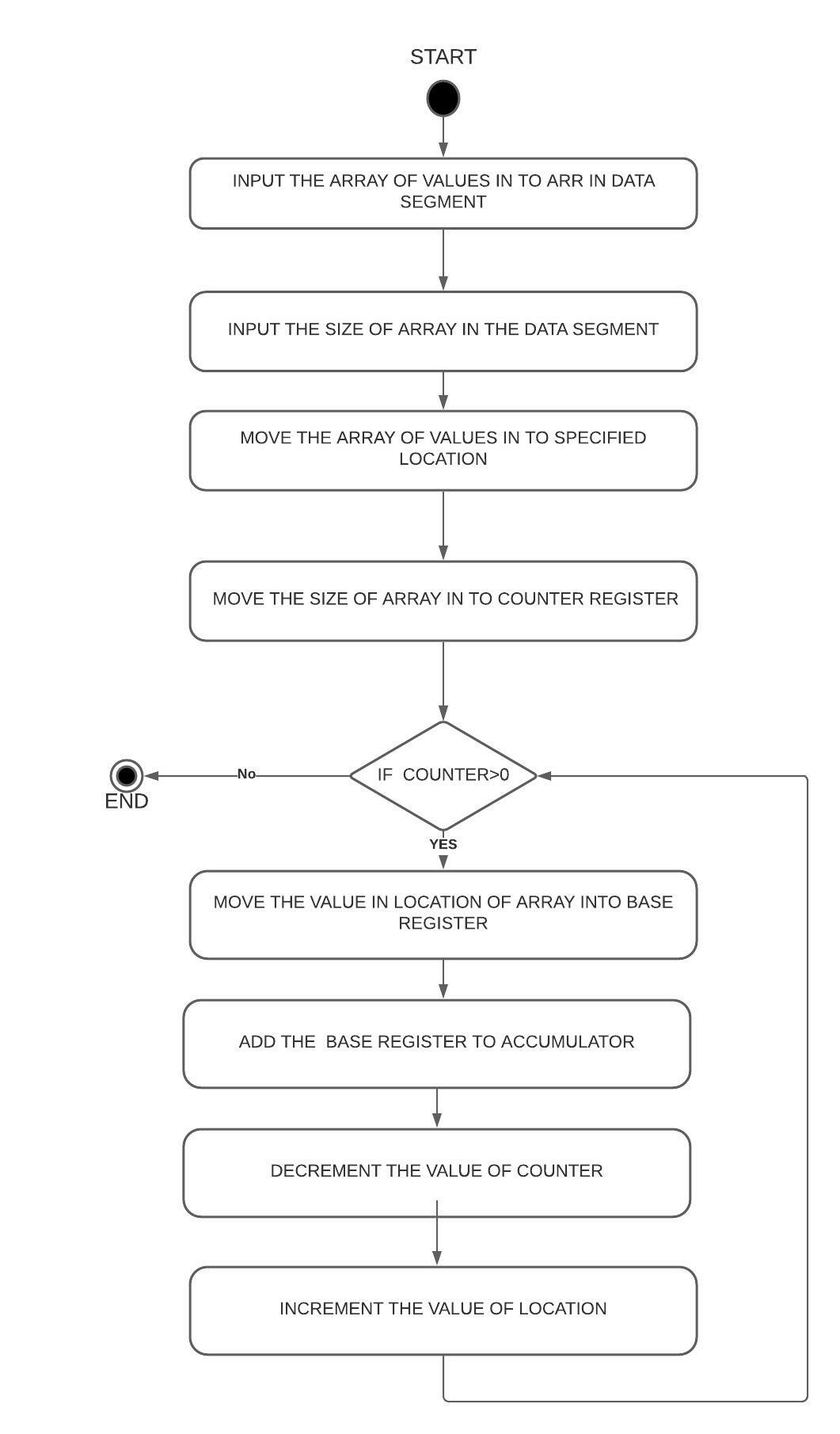
**-Now Run a loop and Add move the values in SI location of to Base Register and add that to Accumulator .Increment the SI value and**

**Decrement the Counter Register**

**-Repeat the Process until the Counter becomes 0**

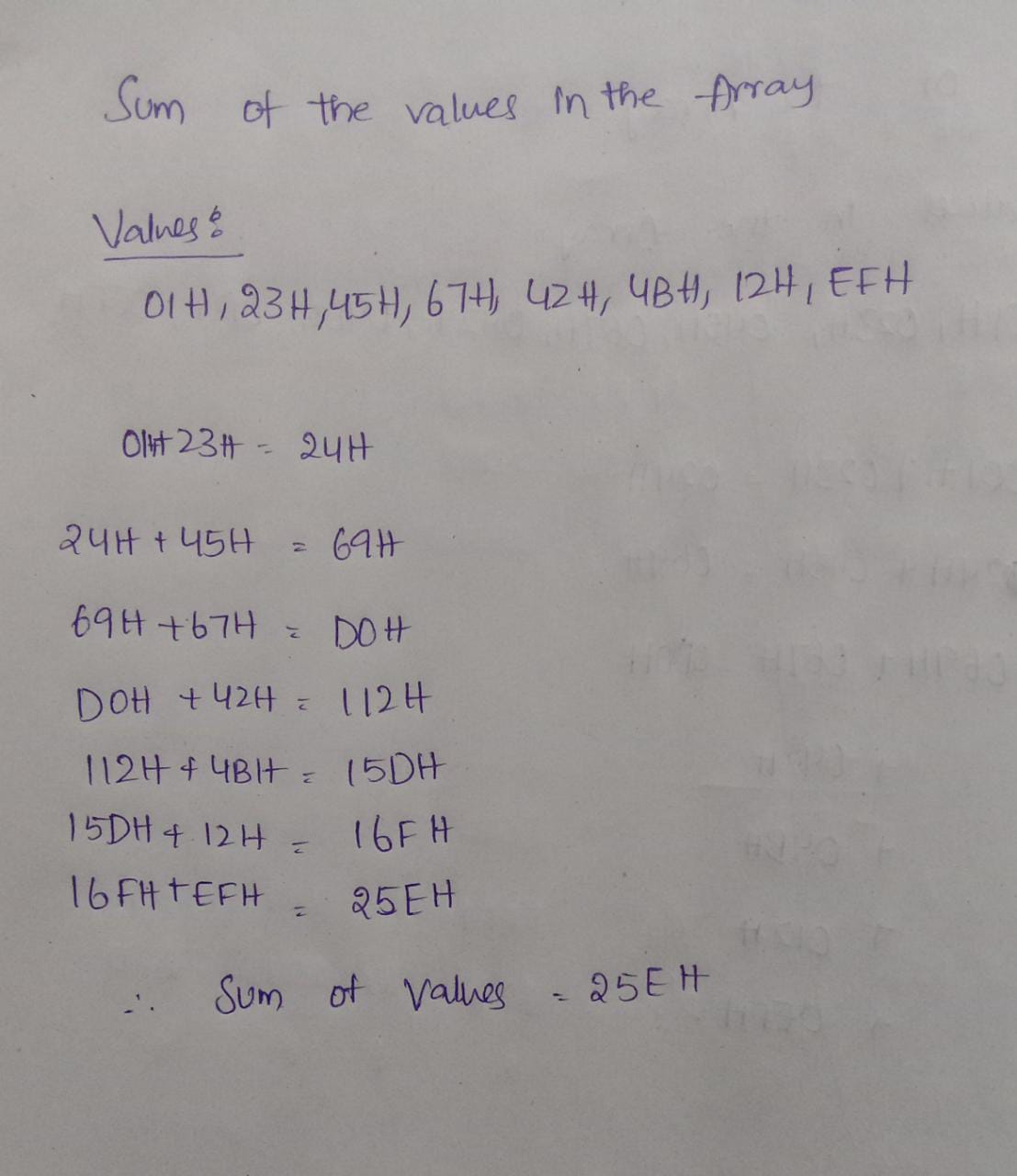
**-The Sum is stored in Accumulator.**

**FLOW CHART**

****

**Design and Calculations:**

**-Input the Array of Values(001H, 023H, 045H, 067H, 042H, 04BH, 012H, 0EFH ) in Data Segment .Input the Size of the Array(8) in Data segment.Move all the values of the Array to a specified Memory Location(SI) in Code Segment.Move the size of Array from Data segment in to Counter Register.Now Run a loop and Add move the values in SI location of to Base Register and add that to Accumulator .Increment the SI value and Decrement the Counter Register.Repeat the Process until the Counter becomes 0 .The Sum is stored in Accumulator.**

****

**Program Code:**

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

**DATA SEGMENT**

**ARR DB 001H, 023H, 045H, 067H, 042H, 04BH, 012H, 0EFH**

**N DW 08H**

**SUM DW 01 DUP (?)**

**DATA ENDS**

**CODE SEGMENT**

**START:**

**MOV AX, @DATA**

**MOV DS, AX**

**MOV CX, N**

**MOV AX, 0000H**

**MOV SI, OFFSET ARR**

**ABC:**

**MOV BL, [SI]**

**INC SI**

**ADD AX, BX**

**DEC CX**

**JNZ ABC**

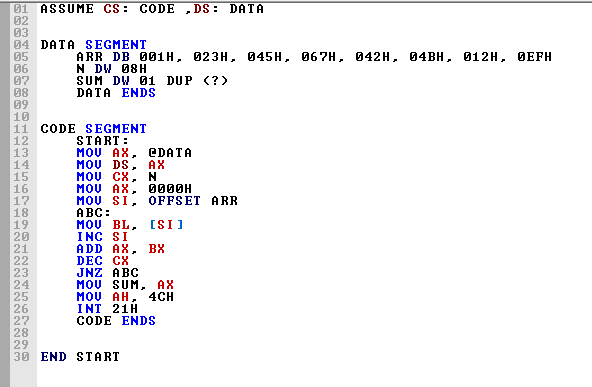
**MOV SUM, AX**

**MOV AH, 4CH**

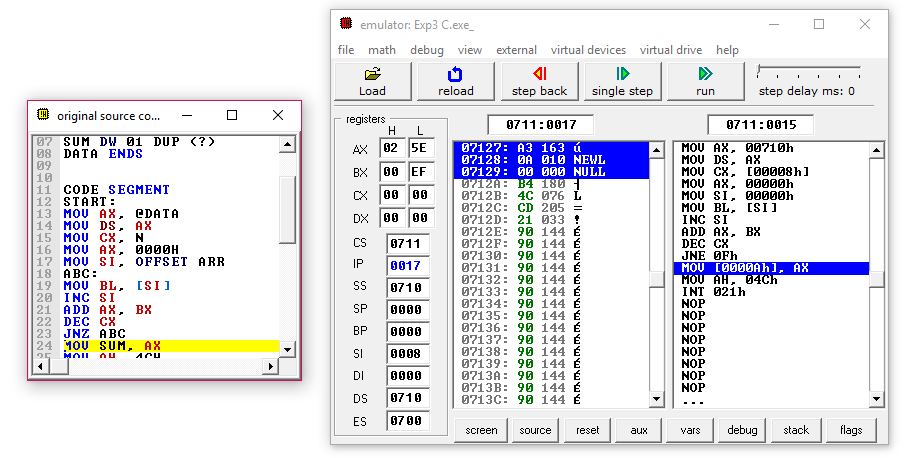
**INT 21H**

**CODE ENDS**

**END START**

****

**Output:**

****

**Result and Inference:**

**-The values of the array are loaded to Base Register by increasing the value of location.**

**-Each time the value loaded in register is added to the Accumulator**

**- The process is continued till the counter becomes 0**

**-Hence the sum is stored in Accumulator**