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**

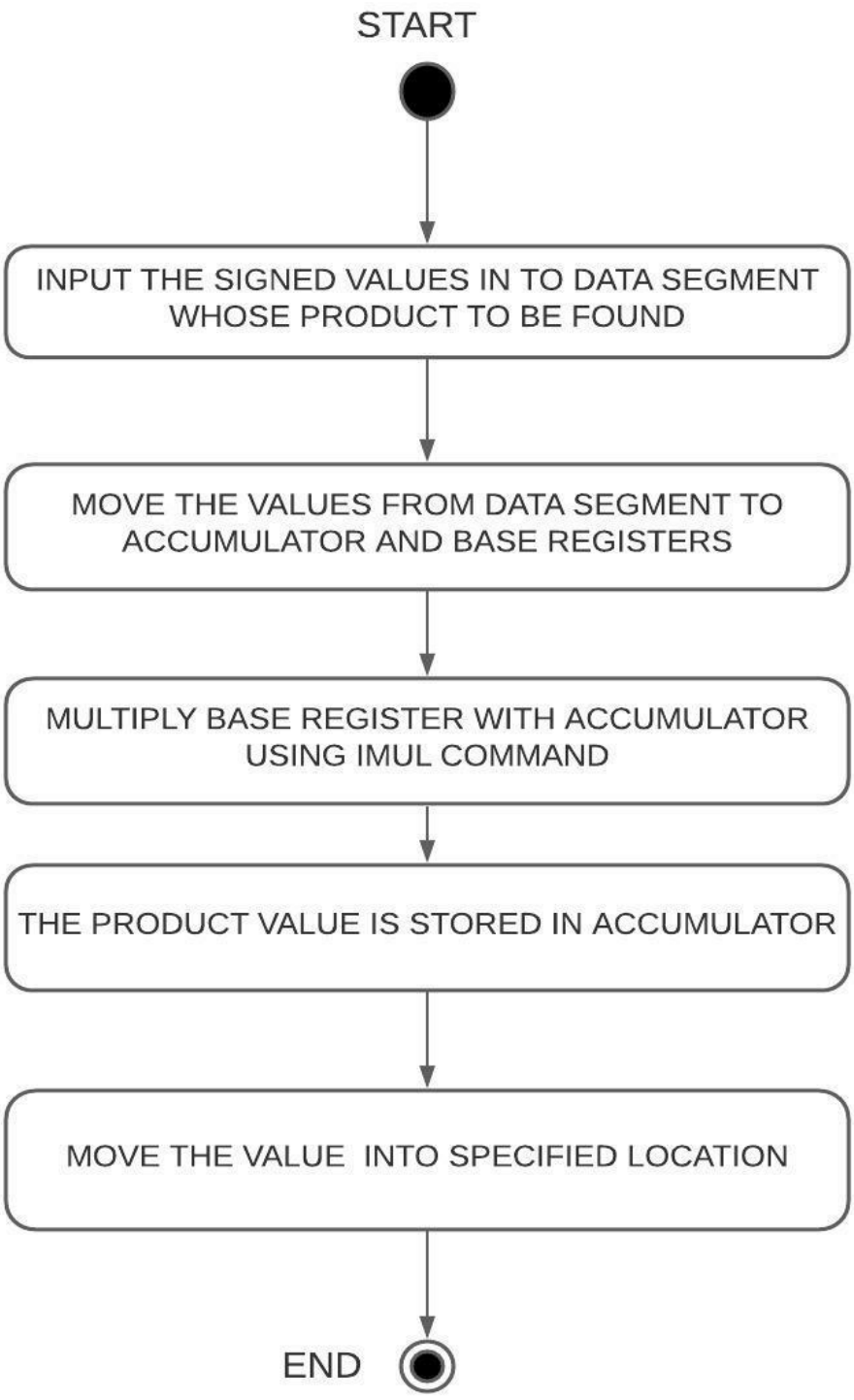
Tool Used: emu8086 simulator

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.**

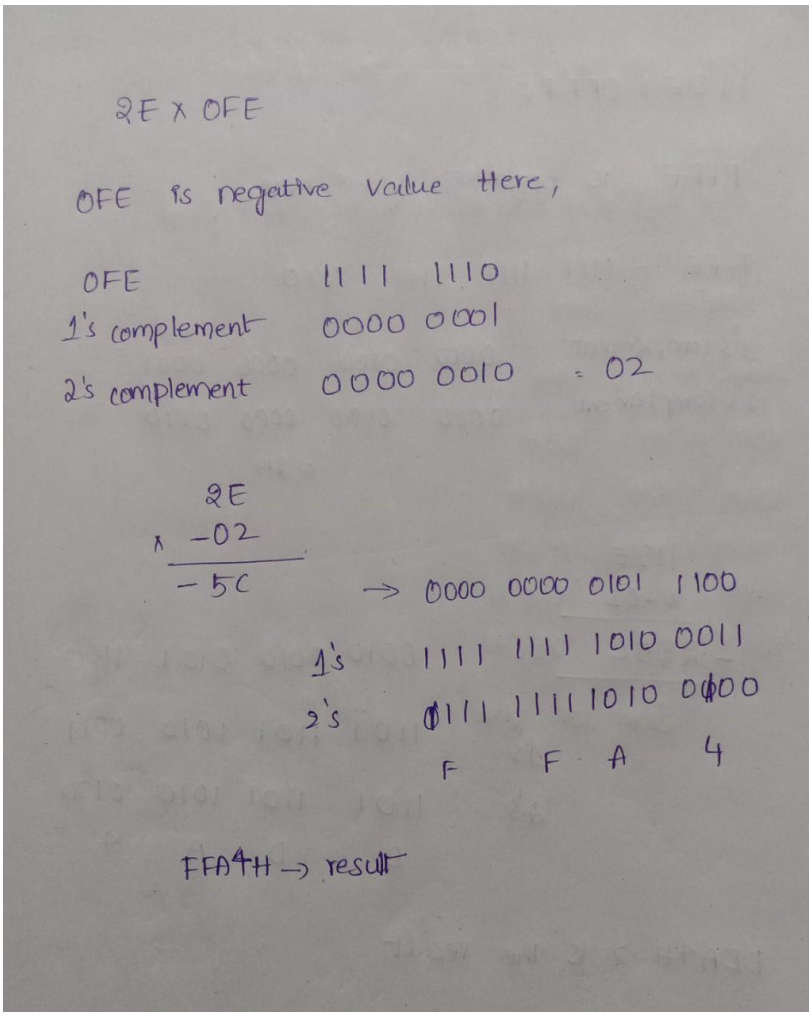
Flow Chart:


START
INPUT THE SIGNED VALUES IN TO DATA SEGMENT WHOSE PRODUCT TO BE FOUND
MOVE THE VALUES FROM DATA SEGMENT TO ACCUMULATOR AND BASE REGISTERS
MULTIPLY BASE REGISTER WITH ACCUMULATOR USING IMUL COMMAND
THE PRODUCT VALUE IS STORED IN ACCUMULATOR
MOVE THE VALUE INTO SPECIFIED LOCATION
END

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)

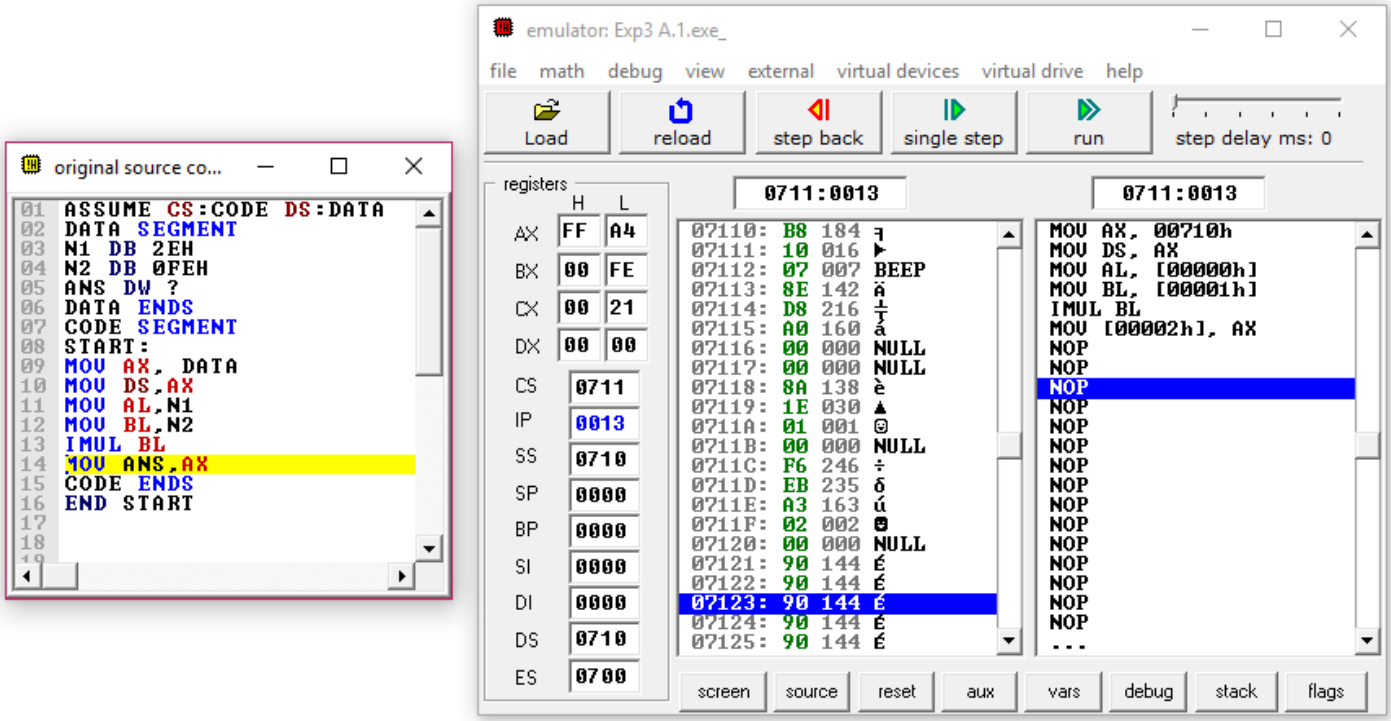


Program Code:

```
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
```

```
01  ASSUME CS:CODE DS:DATA
02  DATA SEGMENT
03       N1 DB 2EH
04       N2 DB 0FEH
05       ANS DW ?
06  DATA ENDS
07  CODE SEGMENT
08       START:
09       MOV AX, DATA
10       MOV DS,AX
11       MOV AL,N1
12       MOV BL,N2
13       IMUL BL
14       MOV ANS,AX
15       CODE ENDS
16  END START
17
```

Output:



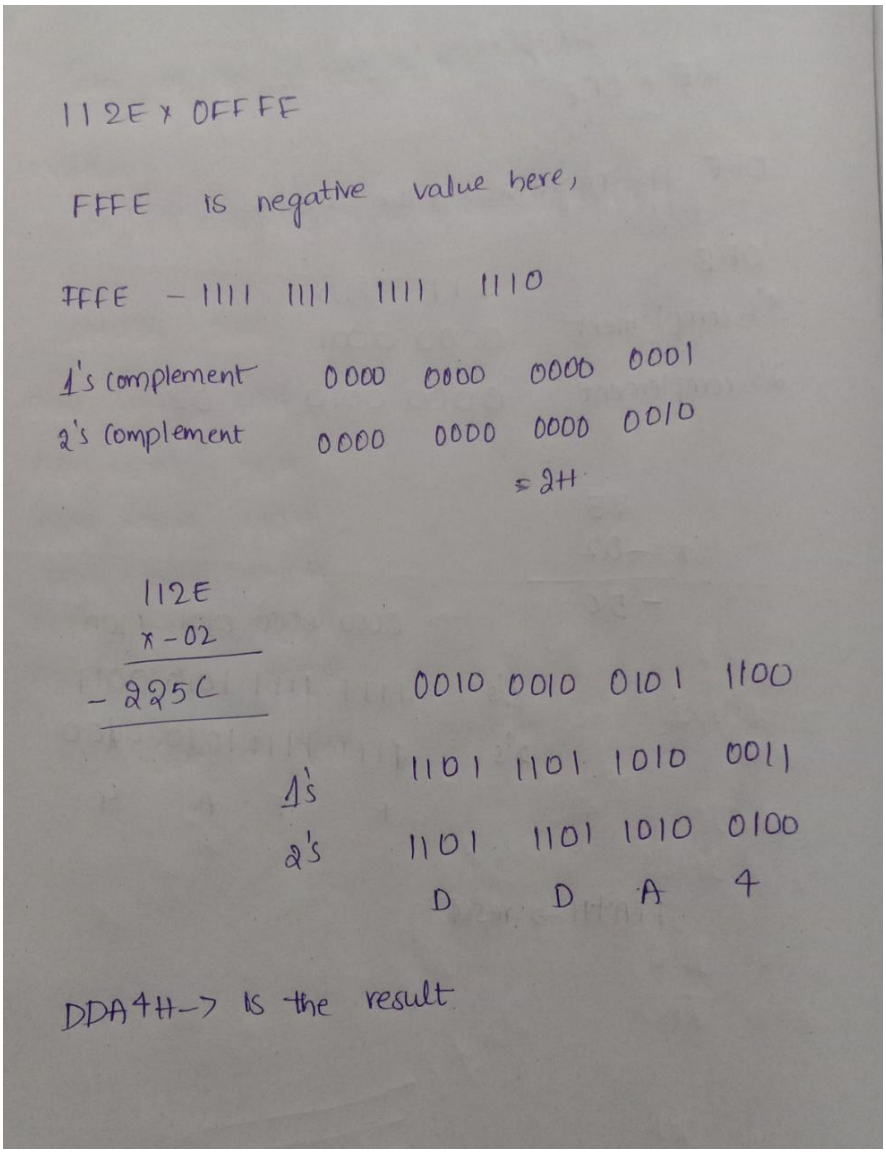
Result and Inference:

-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)

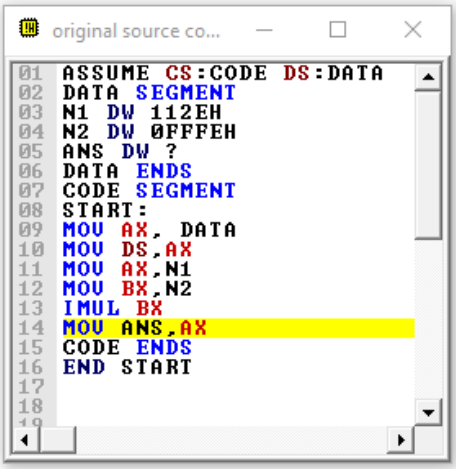


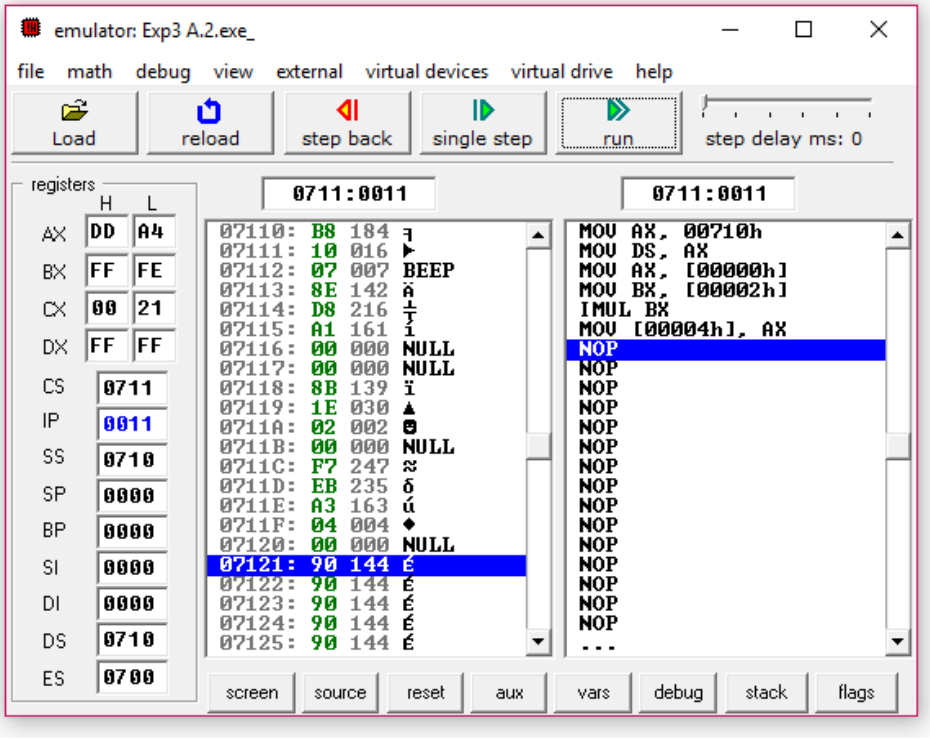
Program Code:

```
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
```

```
01 ASSUME CS:CODE DS:DATA
02 DATA SEGMENT
03     N1 DW 112EH
04     N2 DW 0FFFEH
05     ANS DW ?
06 DATA ENDS
07 CODE SEGMENT
08     START:
09     MOV AX, DATA
10     MOV DS,AX
11     MOV AX,N1
12     MOV BX,N2
13     IMUL BX
14     MOV ANS,AX
15     CODE ENDS
16 END START
17
```

Output:





Result and Inference:

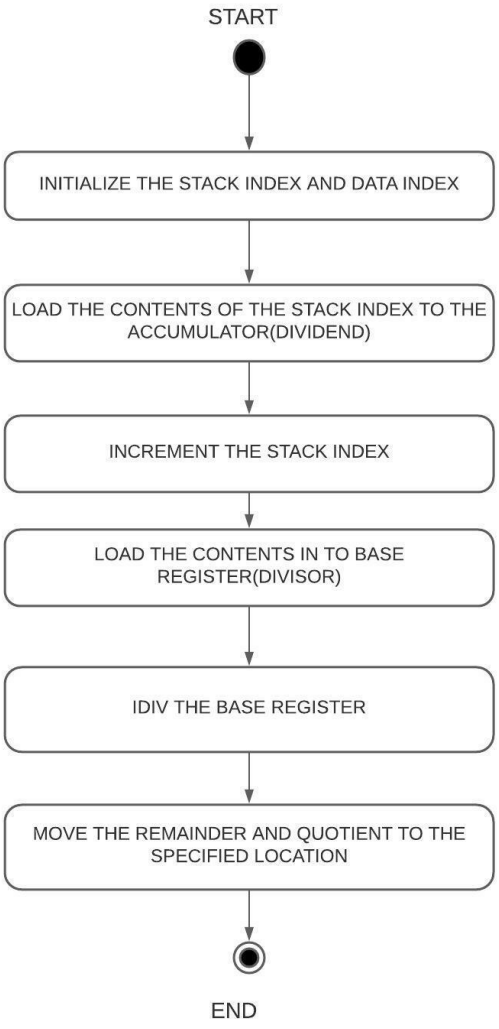
-The Result obtained is same as the calculated value(DDA4h) in Accumulator

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:


START
INITIALIZE THE STACK INDEX AND DATA INDEX
LOAD THE CONTENTS OF THE STACK INDEX TO THE ACCUMULATOR(DIVIDEND)
INCREMENT THE STACK INDEX
LOAD THE CONTENTS IN TO BASE REGISTER(DIVISOR)
IDIV THE BASE REGISTER
MOVE THE REMAINDER AND QUOTIENT TO THE SPECIFIED LOCATION
END

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

Divison of 16 bit by 8 bit Number

F336 / 75

F336 is Negative Number

1111 0011 0011 0110

1's 0000 1100 1100 1001

2's 0000 1100 1100 1010

0 C C A

= -CCA

75) -CCA (1B

75

57A

507

73

Divisor = -1B 0001 1011

1's 1110 0100

2's 1110 0101 -E5

Remainder = -73 0111 0011

1's 1000 1100

2's 1000 1101 -8D

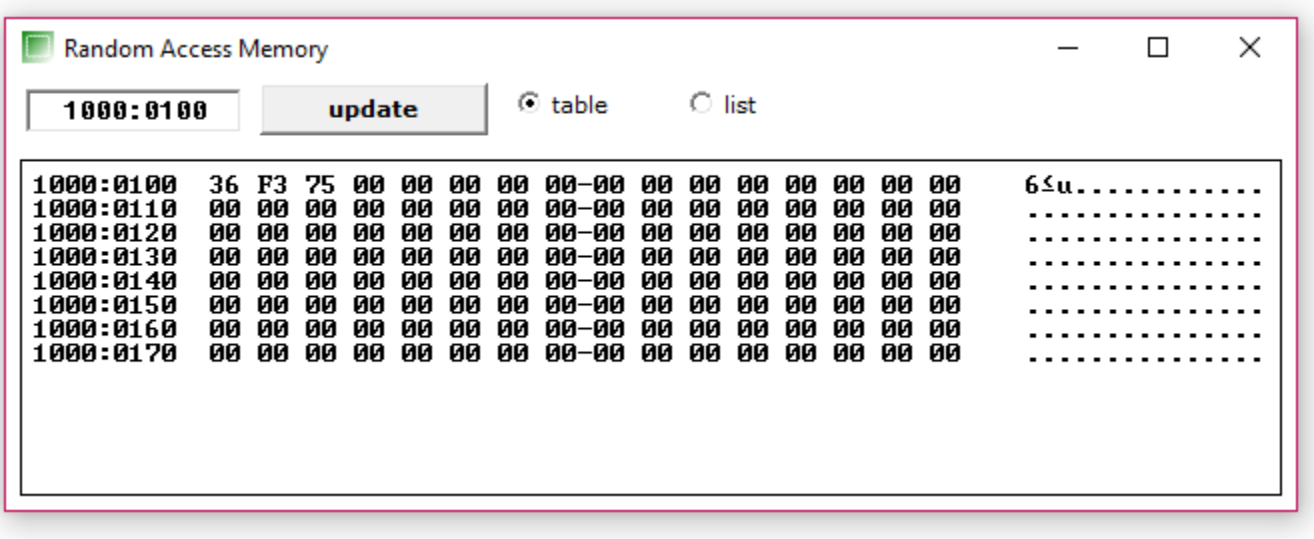
Divisor - E5 Remainder - 8D

Program Code:

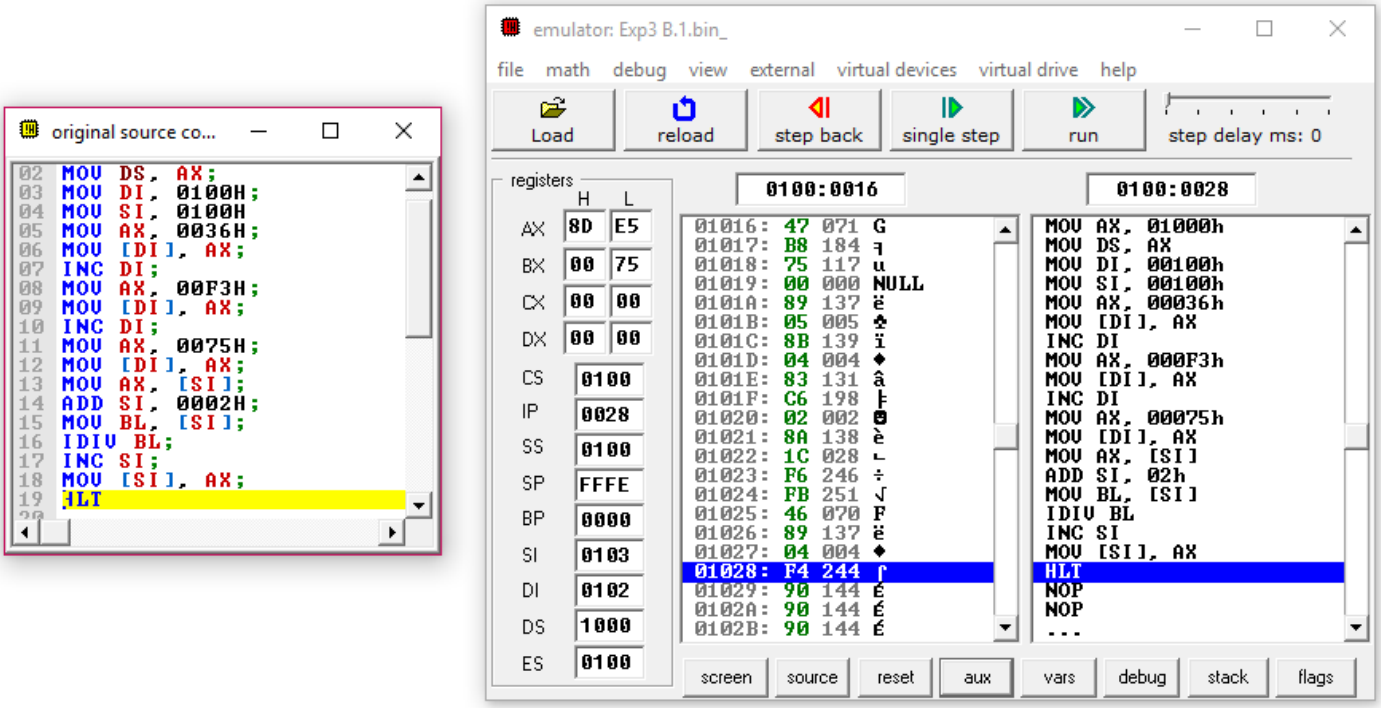
```
MOV AX, 1000H;
MOV DS, AX;
MOV DI, 0100H;
MOV SI, 0100H
MOV AX, 0036H;
MOV [DI], AX;
INC DI;
MOV AX, 00F3H;
MOV [DI], AX;
INC DI;
MOV AX, 0075H;
MOV [DI], AX;
MOV AX, [SI];
ADD SI, 0002H;
MOV BL, [SI];
IDIV BL;
INC SI;
MOV [SI], AX;
HLT
```

```
01 MOV AX, 1000H;
02 MOV DS, AX;
03 MOV DI, 0100H;
04 MOV SI, 0100H
05 MOV AX, 0036H;
06 MOV [DI], AX;
07 INC DI;
08 MOV AX, 00F3H;
09 MOV [DI], AX;
10 INC DI;
11 MOV AX, 0075H;
12 MOV [DI], AX;
13 MOV AX, [SI];
14 ADD SI, 0002H;
15 MOV BL, [SI];
16 IDIV BL;
17 INC SI;
18 MOV [SI], AX;
19 HLT
```

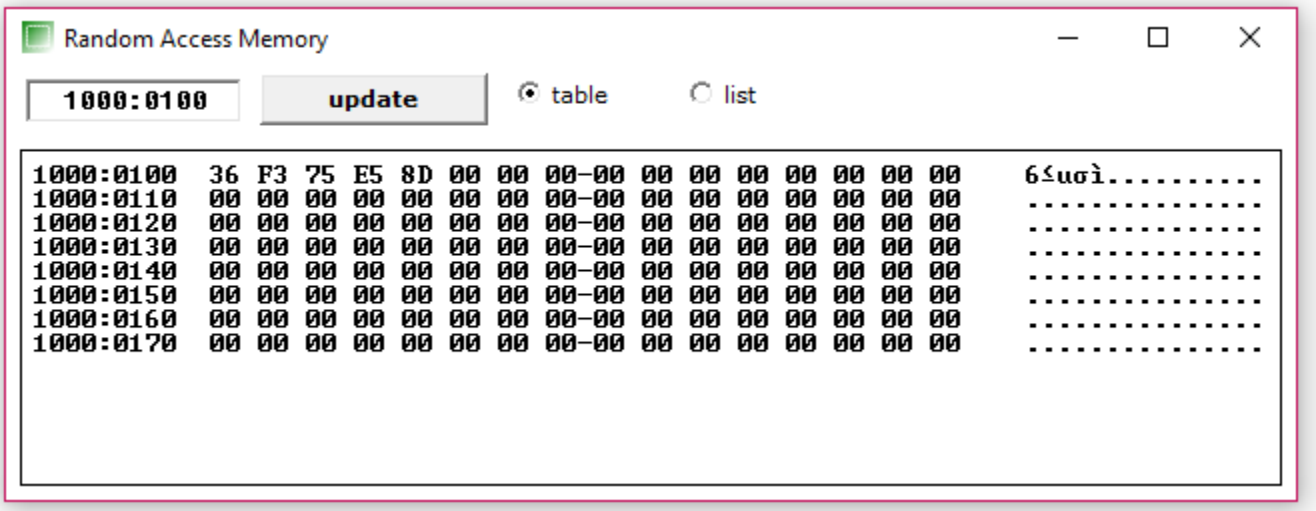
Memory before Divison :



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:

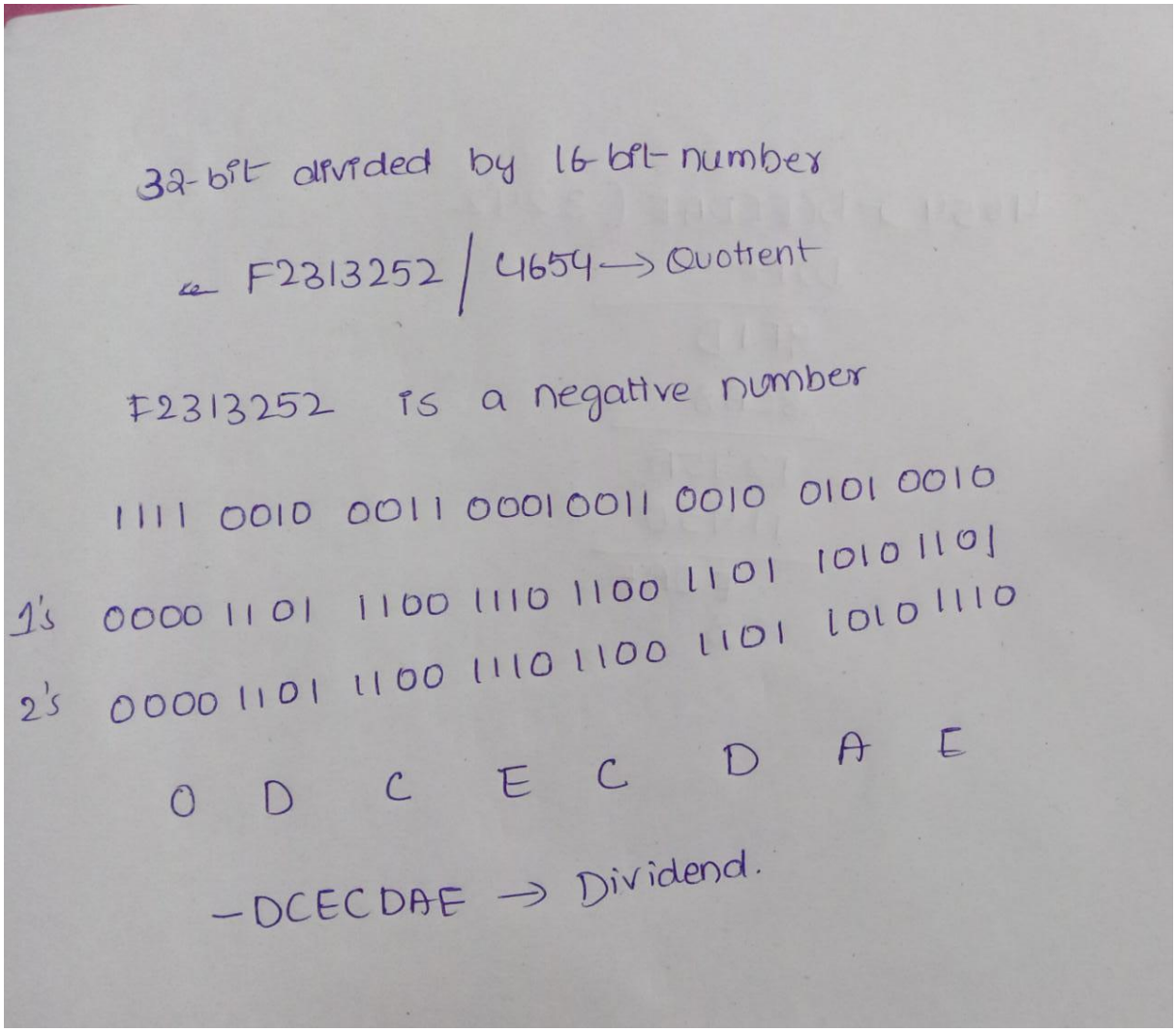
Initialize the Data Segment(2000h) stack index(SI)[0100h] and data index(DI)[0100h] to point at the location and move the Dividend(F2313252h)and divisor(4654h) to Memory by incrementing the SI value .

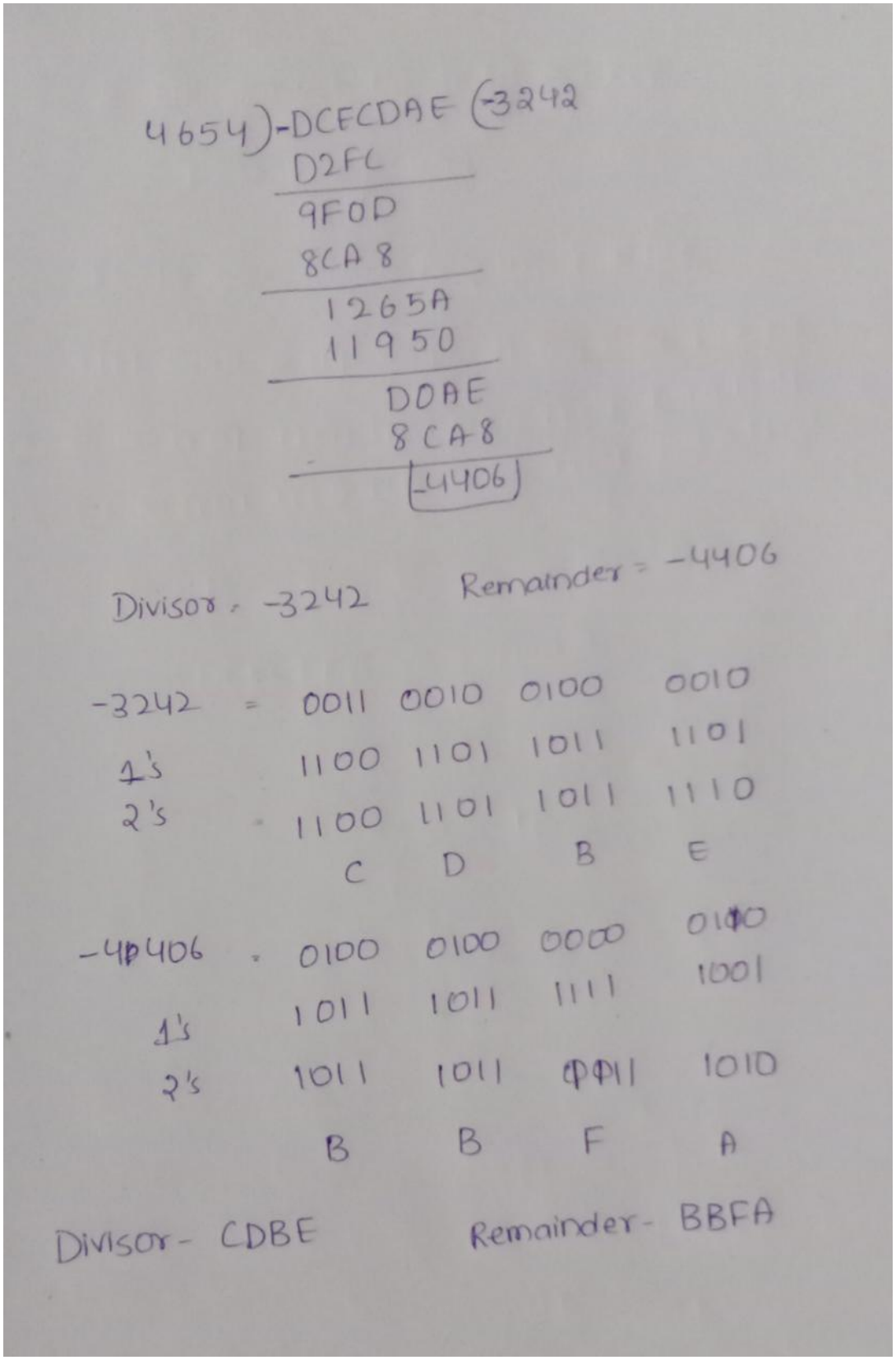
Load the values in stack index to Accumulator and Data Register(DX)(Dividend)

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


32-bit divided by 16-bit number
← F2313252 / 4654 → Quotient
F2313252 is a negative number
1111 0010 0011 0001 0011 0010 0101 0010
1's 0000 1101 1100 1110 1100 1101 1010 1101
2's 0000 1101 1100 1110 1100 1101 1010 1110
0 D C E C D A E
-DCECDAE → Dividend.


4654)-DCFCDAE (-3242
D2FC
9F0D
8CA8
1265A
11950
D0AE
8CA8
-4406
Divisor: -3242 Remainder = -4406
-3242 = 0011 0010 0100 0010
1's 1100 1101 1011 1101
2's 1100 1101 1011 1110
C D B E
-4406 = 0100 0100 0000 0110
1011 1011 1111 1001
1's
2's 1011 1011 1111 1010
B B F A
Divisor - CDBE Remainder - BBFA

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;

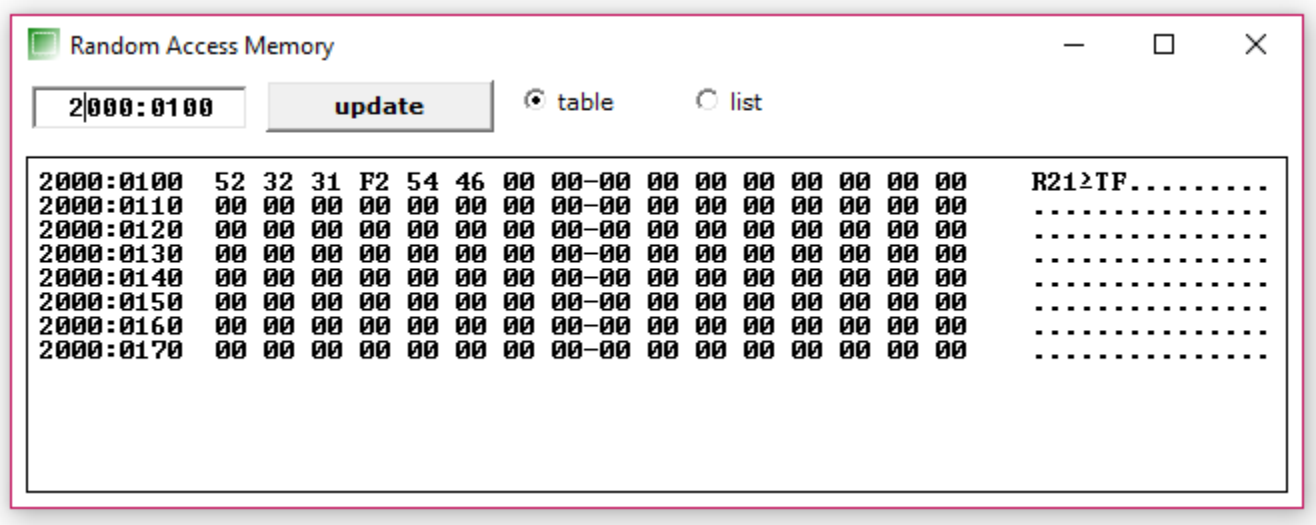
ADD SI, 0002H;

MOV [SI], DX;

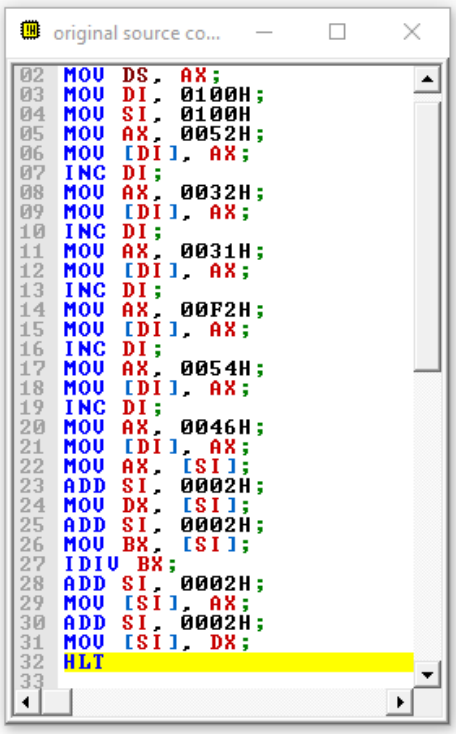
HLT

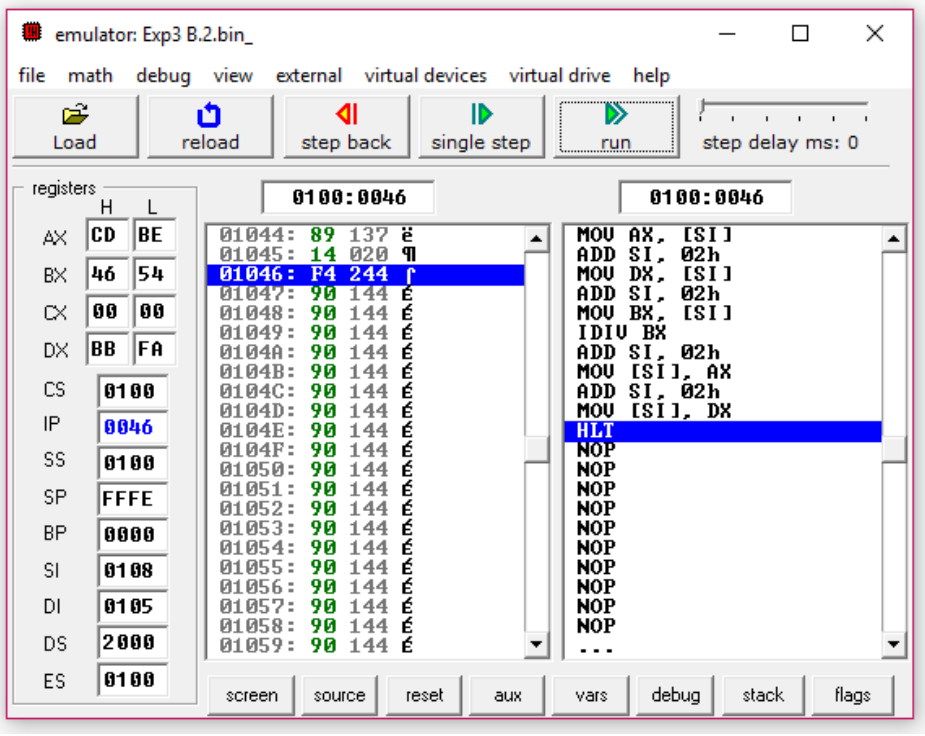
```
01 MOV AX, 2000H;
02 MOV DS, AX;
03 MOV DI, 0100H;
04 MOV SI, 0100H
05 MOV AX, 0052H;
06 MOV [DI], AX;
07 INC DI;
08 MOV AX, 0032H;
09 MOV [DI], AX;
10 INC DI;
11 MOV AX, 0031H;
12 MOV [DI], AX;
13 INC DI;
14 MOV AX, 00F2H;
15 MOV [DI], AX;
16 INC DI;
17 MOV AX, 0054H;
18 MOV [DI], AX;
19 INC DI;
20 MOV AX, 0046H;
21 MOV [DI], AX;
22 MOV AX, [SI];
23 ADD SI, 0002H;
24 MOV DX, [SI];
25 ADD SI, 0002H;
26 MOV BX, [SI];
27 IDIV BX;
28 ADD SI, 0002H;
29 MOV [SI], AX;
30 ADD SI, 0002H;
31 MOV [SI], DX;
32 HLT
```

Memory before Division:

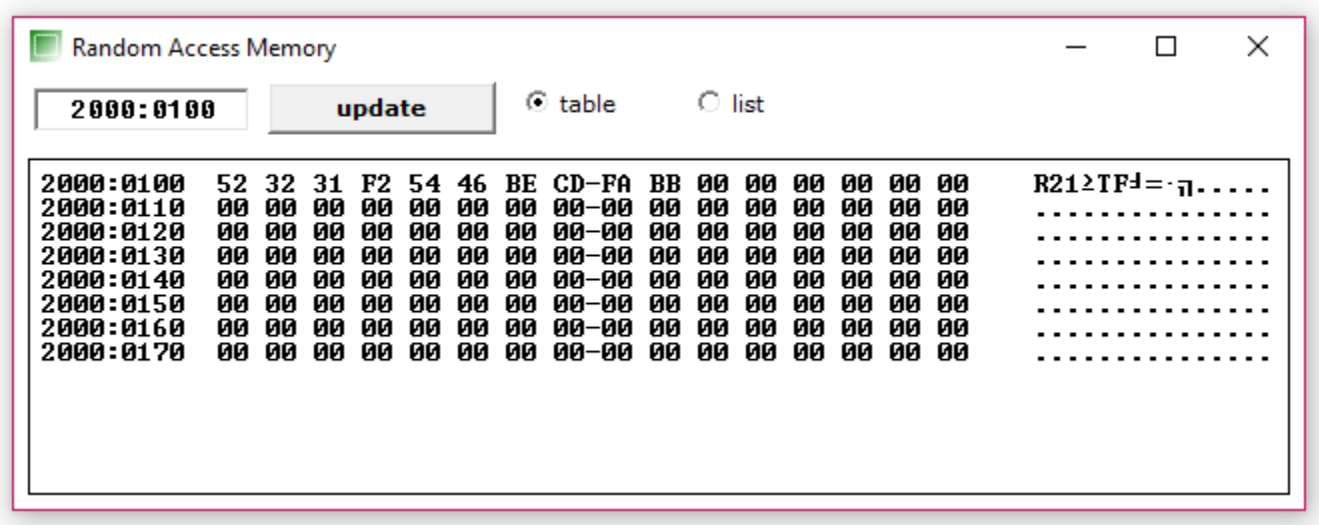


Output:





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

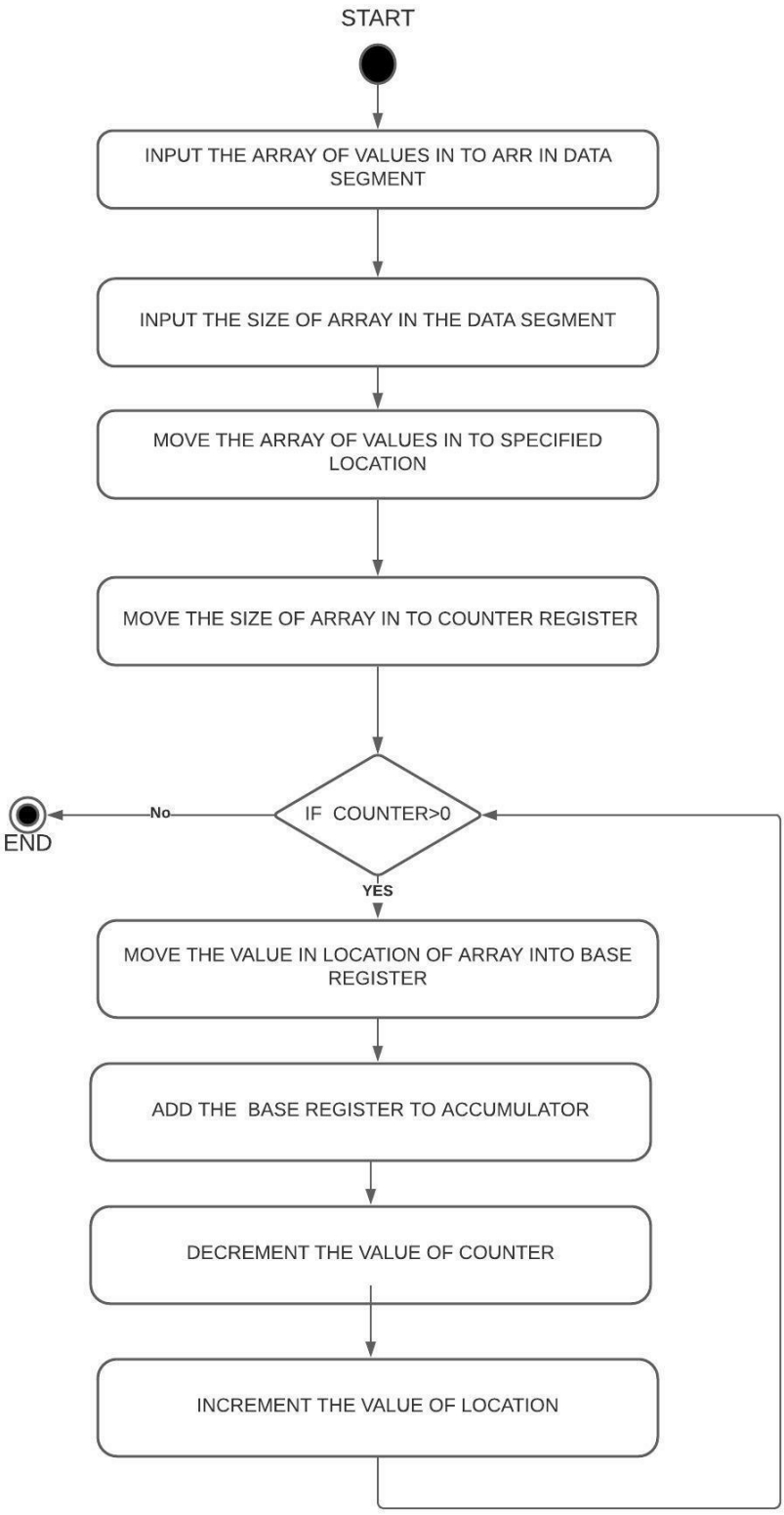
-Now Run a loop and Add move the values in SI location of to Base Register and add that to Accumulator .Increment the SI value and

Decrement the Counter Register

-Repeat the Process until the Counter becomes 0

-The Sum is stored in Accumulator.

FLOW CHART


START
INPUT THE ARRAY OF VALUES IN TO ARR IN DATA SEGMENT
INPUT THE SIZE OF ARRAY IN THE DATA SEGMENT
MOVE THE ARRAY OF VALUES IN TO SPECIFIED LOCATION
MOVE THE SIZE OF ARRAY IN TO COUNTER REGISTER
IF COUNTER>0
No
END
YES
MOVE THE VALUE IN LOCATION OF ARRAY INTO BASE REGISTER
ADD THE BASE REGISTER TO ACCUMULATOR
DECREMENT THE VALUE OF COUNTER
INCREMENT THE VALUE OF LOCATION

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.

Sum of the values in the Array

Values:

01H, 23H, 45H, 67H, 42H, 4BH, 12H, EFH

01H + 23H = 24H

24H + 45H = 69H

69H + 67H = D0H

D0H + 42H = 112H

112H + 4BH = 15DH

15DH + 12H = 16FH

16FH + EFH = 25EH

∴ Sum of values = 25EH

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

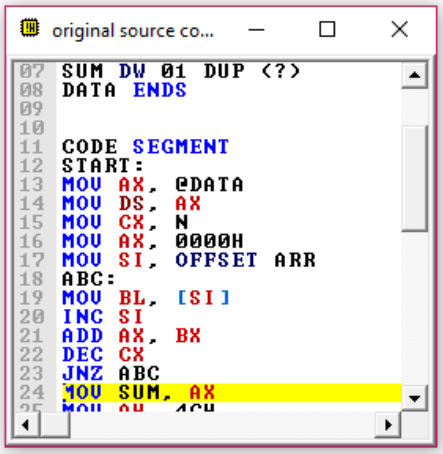
INT 21H

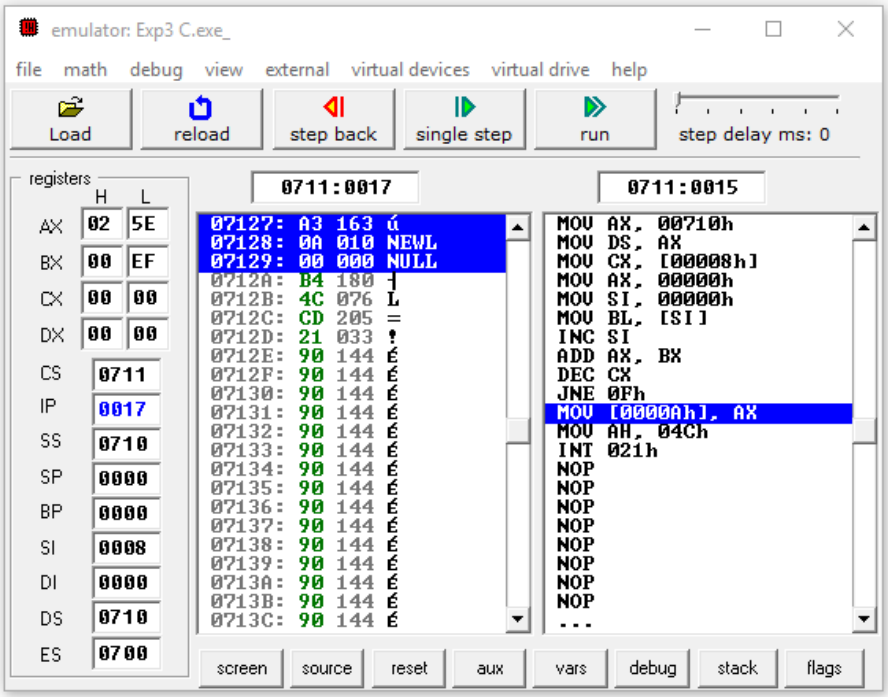
CODE ENDS

END START

```
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