Fall Semester 2021-2022

Microprocessor and Interfacing

Lab Report

Experiment No: 1 Task No: 1

Course Code: CSE2006

Slot: L7+L8

Submitted By: Alokam Nikhitha

Reg. Numb: 19BCE2555

Submitted To: Dr. Abdul Majed KK

Programs involving Arithmetic and Data Transfer Operations

Aim:

A. Addition:

1) Write a program to add two 8-bit numbers and store the results in AX.
2) Write a program to add two 16-bit numbers and store the results in BX.
3) Write a program to add two 16-bit numbers stored in the memory location [1000H], [1002H] and store its final results in the memory location [1004H] in BX.
4) Write a program to add a data byte located at offset 0500H in 2000H segment to another data byte available to 0600H in the same segment and store the result at 0700H in the same segment.

B. Subtraction:

1) Write a program to subtract two 8-bit numbers and store the results in AX.
2) Write a program to subtract two 16-bit numbers and store the results in BX.
3) Write a program to subtract two 16-bit numbers stored in the memory location [1000H], [1002H] and store its final results in the memory location [1004H] in BX.
4) Write a program to subtract a data byte located at offset 0500H in the 2000H segment to another data byte available at 0600H in the same segment and store the result at 0700H in the same segment.

C. Data Transfer:

1) Write a program code to write the first five even numbers 0,2,4,6,8 at locations shown in the memory diagram.

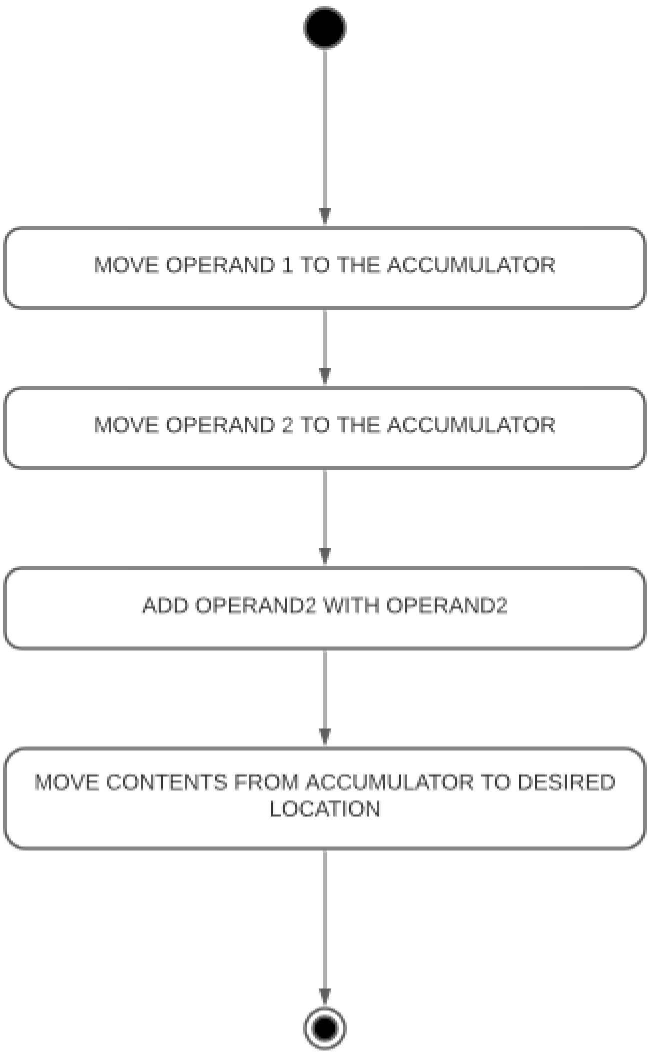
Tool Used: emu8086 simulator

A. Addition

Algorithm:

- Move the first value from the given memory location to accumulator register (AX).
- Move the second value from the given memory location to accumulator register (BX).
- Add the base register 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:



i) 8-bit addition:

Design and Calculations:

For 8-bit we assign values to AL and AH

we are adding the numbers 05H and 03H. The expected final result is 08H.

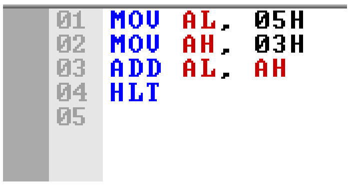
Program Code:

MOV AL, 05H

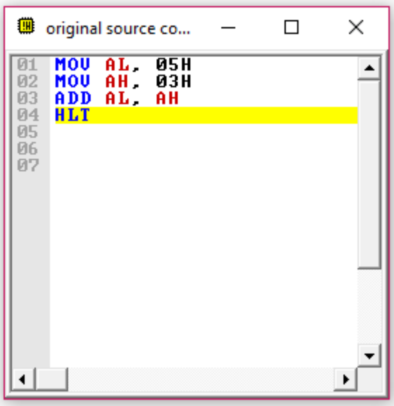
MOV AH, 03H

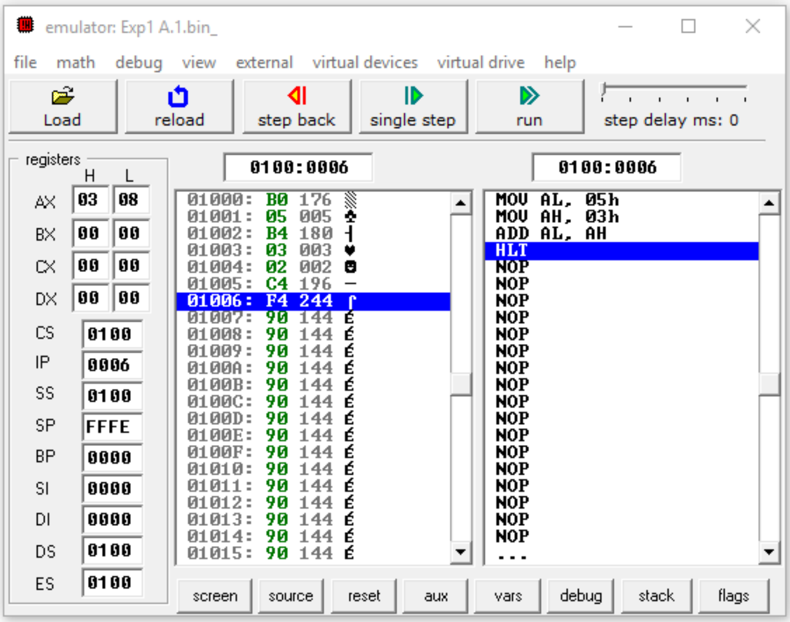
ADD AL, AH

HLT



Output:





Result and Inference:

- **The AL register initially had our value 05H and the AH register has 03H.**
- **The final result obtained is 08H.**
- **The final result is stored in AX register.**

ii) 16-bit addition:

Design and Calculations:

For 16-bit addition, we assign values to AX and BX.In the code we are adding the numbers 3333H and 4444H. The expected final result is 7777H.

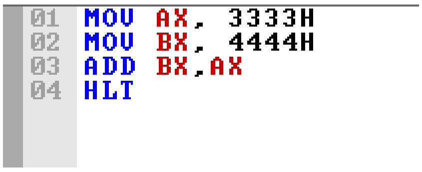
3333

+ 4444

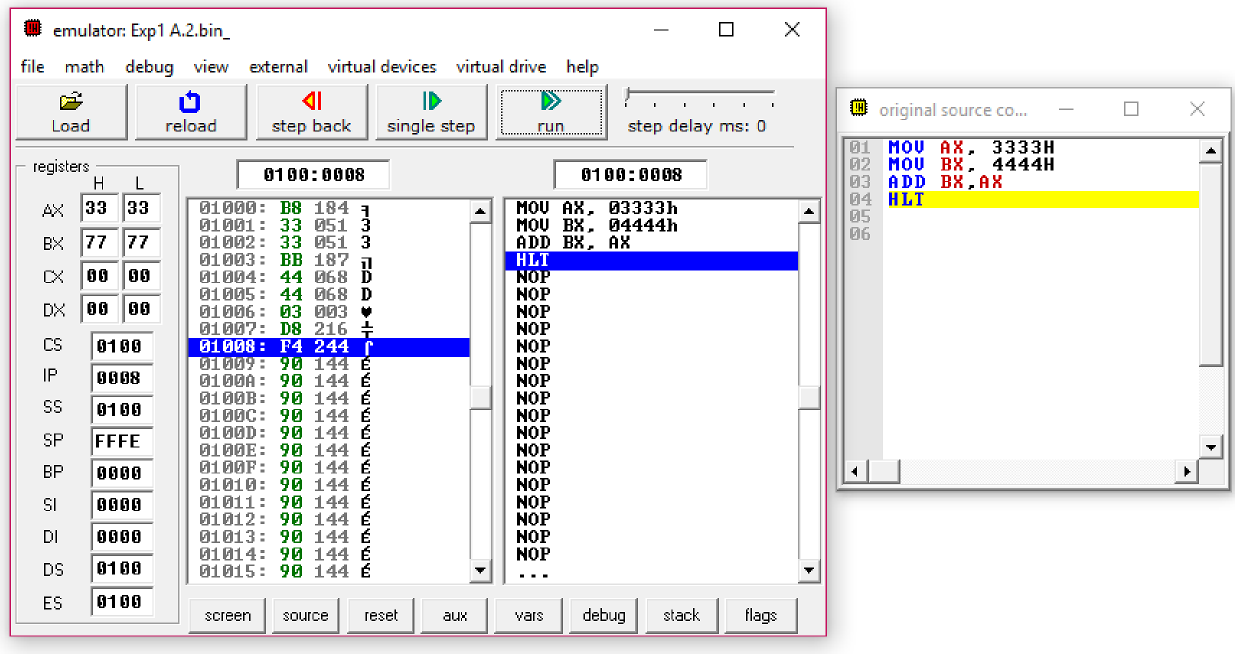
7777

Program Code:

```
MOV AX, 3333H
MOV BX, 4444H
ADD AX, BX
HLT
```



Output:



Result and Inference:

- The AX register initially had value 3333H and BX register has 4444H.
- The final result obtained is 7777H.
- The final result is stored in BX register.

iii) 16-bit addition with memory access:

Design and Calculations:

For 16-bit addition with memory access , The values from given memory location are moved to accumulator and base register. These values are then added to store the result first in accumulator and then the result can be moved to desired location.

We collect data from the memory location at 1000H and 1002H. The value in 1000H is 3333H and at the location 1002H is 4444H. The added result is 7777 and is stored in the memory location 1004H

```
   3333
 + 4444
________
   7777
```

Program Code:

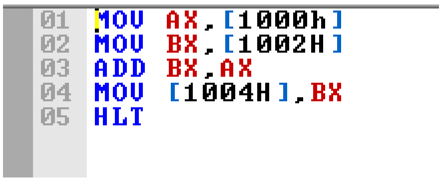
MOV AX, [1000H]

MOV BX, [1002H]

ADD AX, BX

MOV [1004H], AX

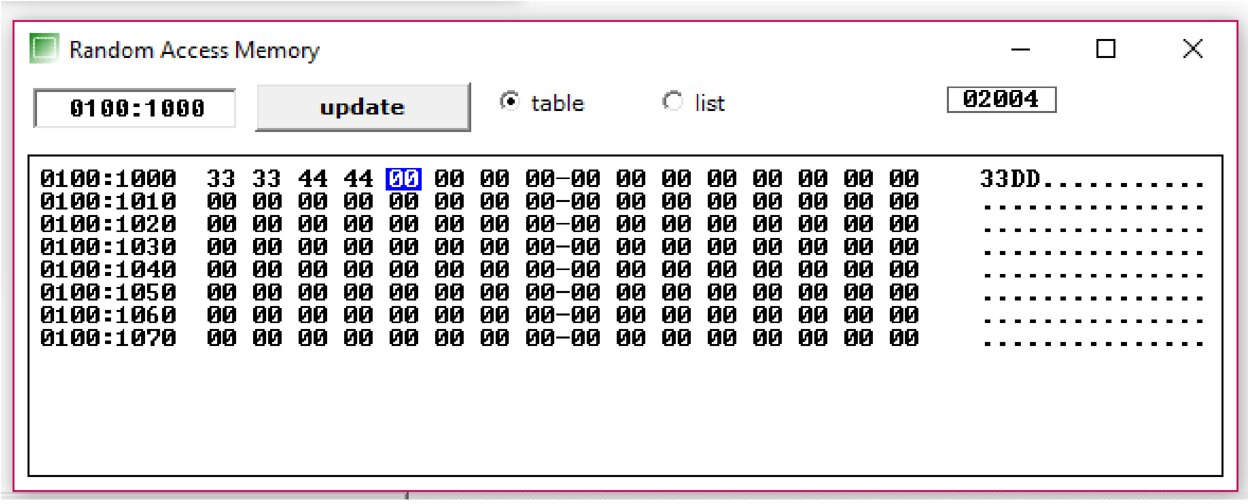
HLT



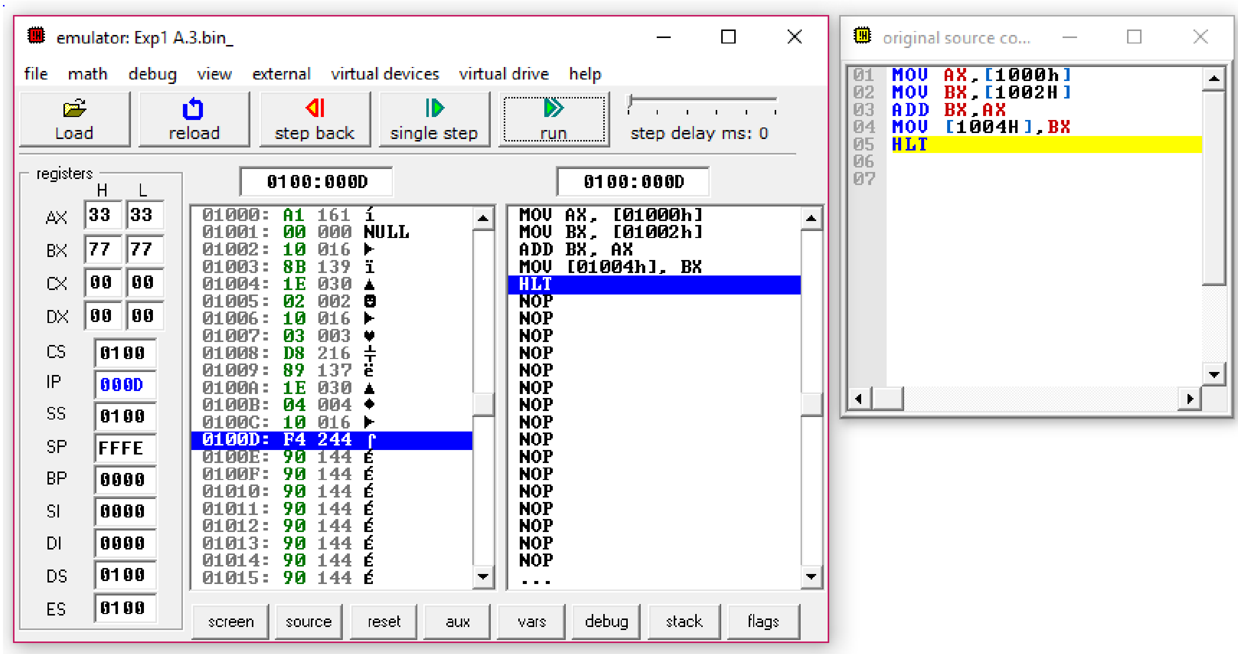
```
MOV AX,[1000h]
MOV BX,[1002H]
ADD BX,AX
MOV [1004H],BX
HLT
```

Output:

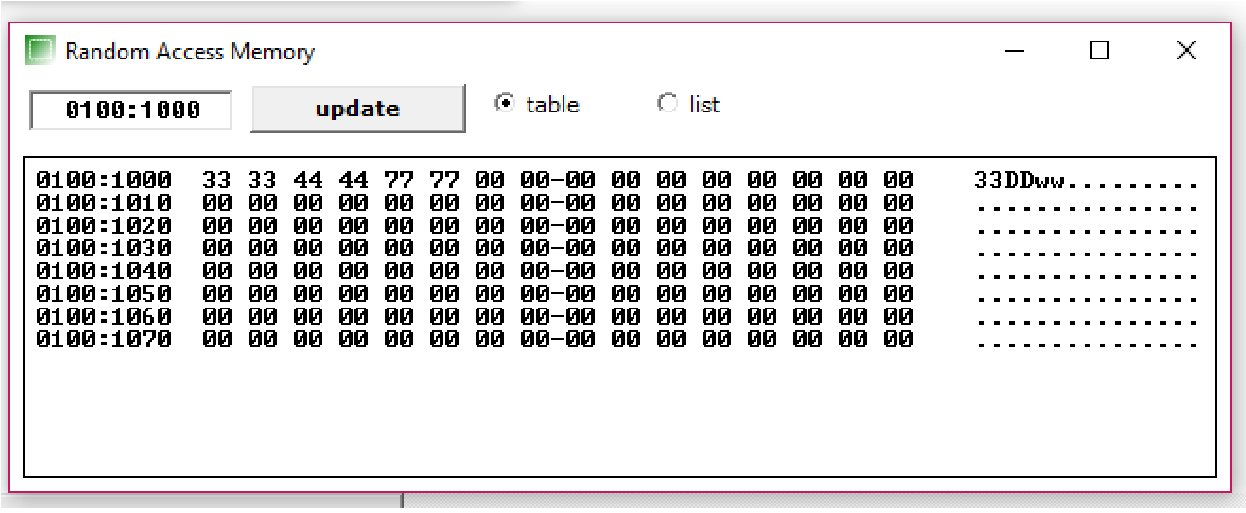
Memory before:



Simulation:



Memory After:



Result and Inference:

- The AX register assigned with data from the location of 1000H which contains 3333H.
- The BX register assigned with data from the location of 1002H which had the value of 4444H.
- The final result obtained is 7777H.
- The final result is stored in AX register and the value is moved to the location 1004H as seen in memory after snippet.

iv) 16-bit addition with physical memory access:

Design and Calculations:

For physical memory access we need physical address of the given location. Then we store the first virtual address in accumulator and shift it to data segment register for external memory reference. The values of segment offset are then updated in accumulator and base register. These registers are then added and the

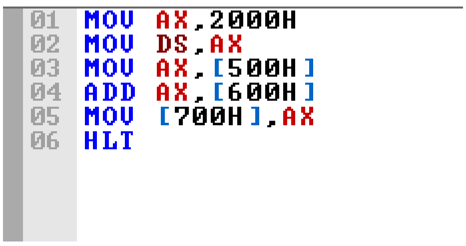
result is stored in accumulator itself. Then the content of accumulator is moved to the specified memory location.

In the code we collect data from memory location at 2000H and offset 500H and 600H. The value in 500H is 3333H and at the location 600H is 4444H. The added result is 7777 and is stored in the memory location 700H

```
  3333
+ 4444
-----------------
  7777
```

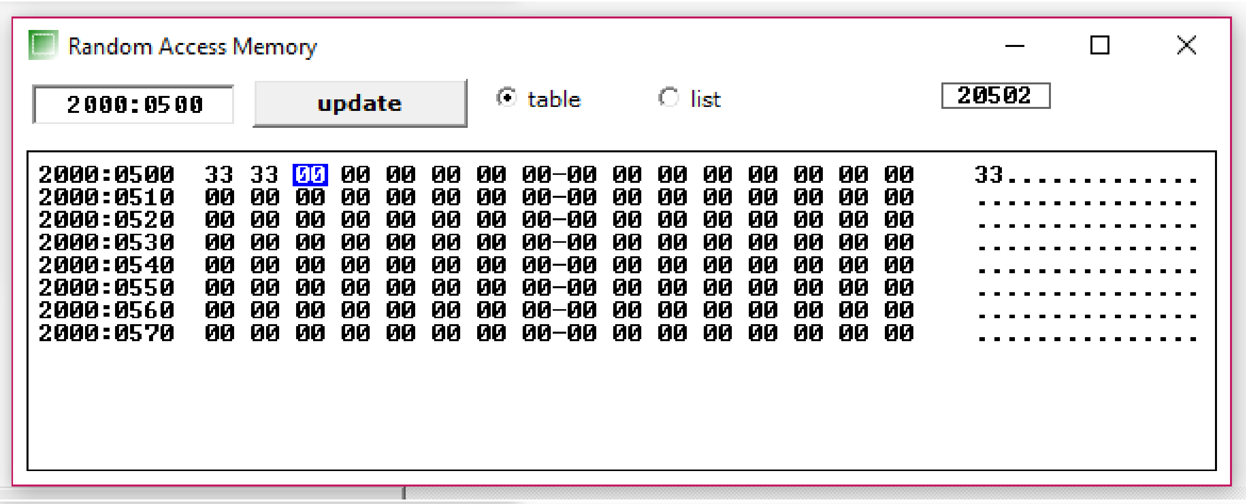
Program Code:

```
MOV AX, 2000H
MOV DS, AX
MOV AX, [500H]
MOV BX, [600H]
ADD AX, BX
MOV [700H], AX
HLT
```

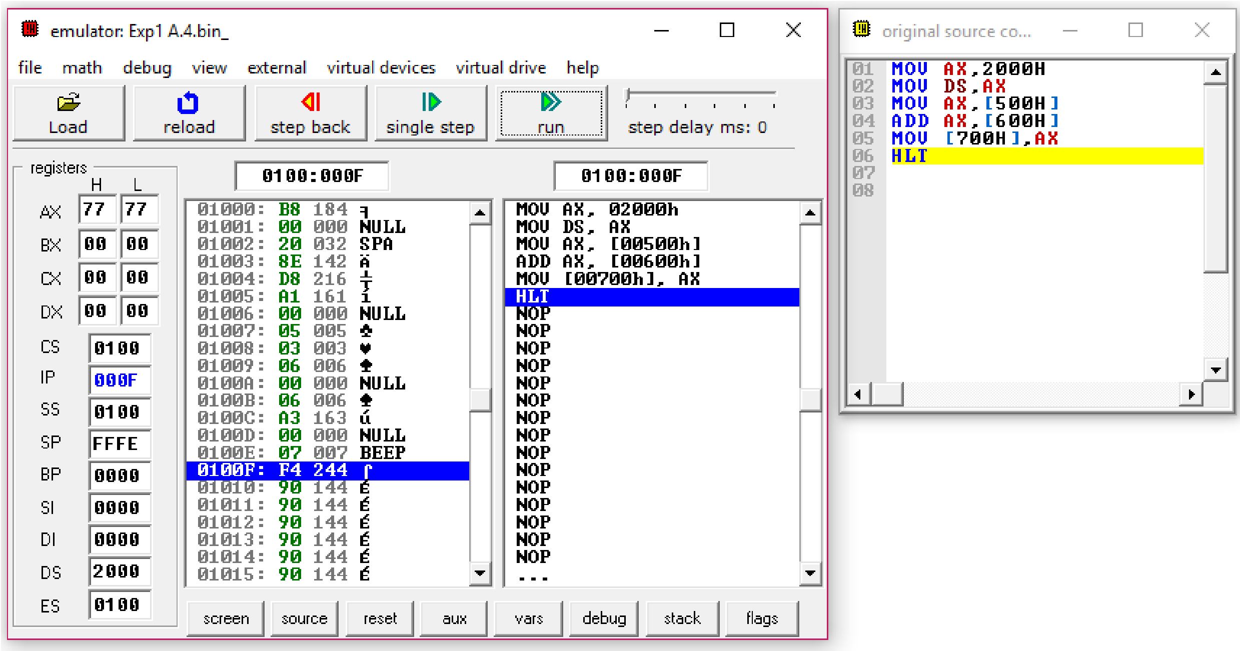


Output:

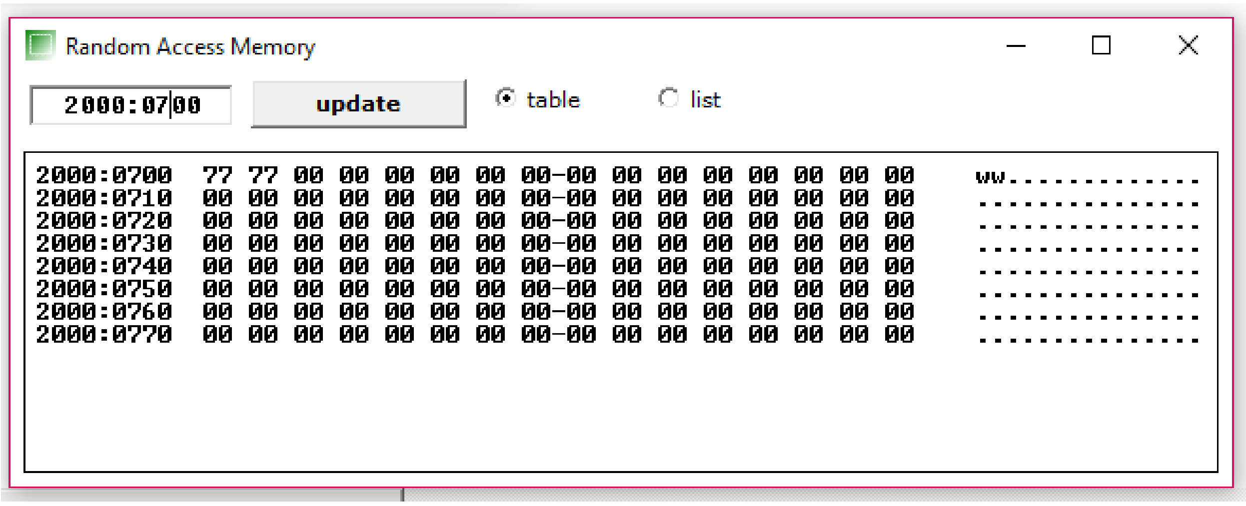
Memory before:



Simulation:



Memory After:



Result and Inference:

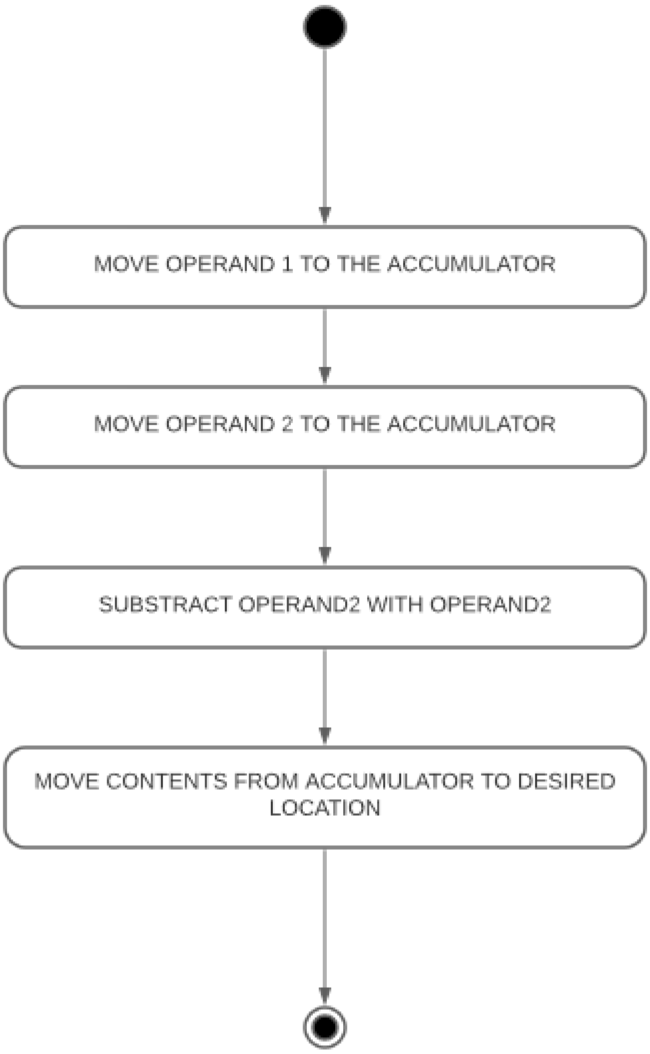
- **The AX register initially has value from location 2000H block and 500H segment offset. The value loaded at AX was 3333.**
- **The BX register initially has value from location 2000H block and 600H segment offset which had the value of 4444.**
- **The final result obtained is 7777H.**
- **The final result is stored in AX register and the value is moved to the location at 2000H block and offset 700H as seen in memory after snippet.**

B. Subtraction

Algorithm:

1) **Move the first value from the given memory location to accumulator register (AX).**
2) **Move the second value from the given memory location to base register (BX).**
3) **Subtract the base register with the accumulator.**
4) **The substracted value is stored in accumulator**
5) **Move the contents of accumulator to desired memory location.**
6) **Halt the process.**

Flow Chart:


MOVE OPERAND 1 TO THE ACCUMULATOR
MOVE OPERAND 2 TO THE ACCUMULATOR
SUBSTRACT OPERAND2 WITH OPERAND2
MOVE CONTENTS FROM ACCUMULATOR TO DESIRED LOCATION

i) 8-bit subtraction:

Design and Calculations:

For 8-bit subtraction we can perform the action in accumulator register itself. The accumulator register is divided into two 8-bit registers termed as AL and AH in order to support 8-bit addition.

In the code we are subtracting the numbers 05H and 03H. The expected final result is 02H.

Program Code:

MOV AL, 05H

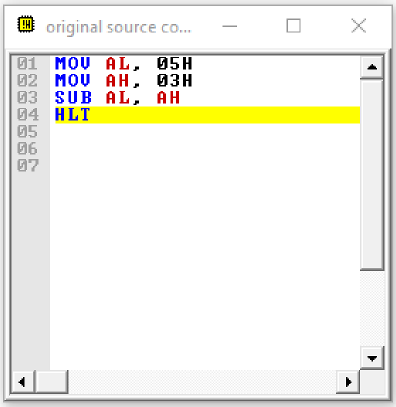
MOV AH, 03H

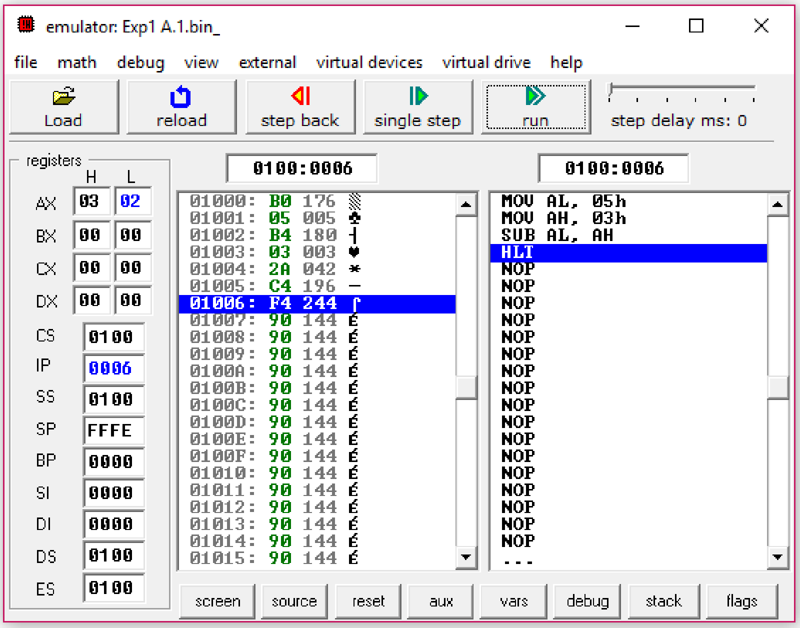
SUB AL, AH

HLT

```
MOV AL, 05H
MOV AH, 03H
SUB AL, AH
HLT
```

Output:





Result and Inference:

- **The AL register initially had our value 05H and the AH register has 03H.**
- **The final result obtained is 02H.**
- **The final result is stored in AX register.**

ii) 16-bit subtraction:

Design and Calculations:

For 16-bit subtraction we need to use the base register and the accumulator register together. This is contrary to how we did 8-bit subtraction by only using accumulator register.

In the code we are subtracting the numbers 3232H and 6767H. The expected final result is 3535H.

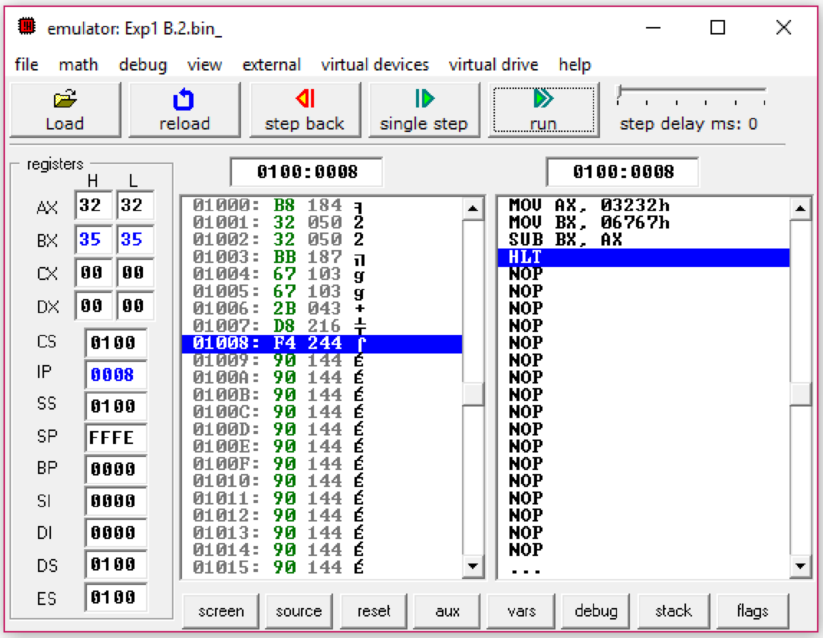
```
  6767
- 3232
----------------
  3535
```

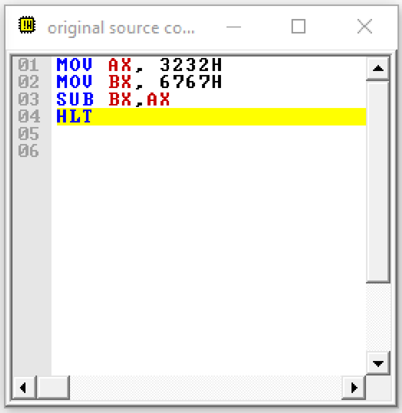
Program Code:

```
MOV AX, 3232H
MOV BX, 6767H
SUB AX, BX
HLT
```

```
MOV AX, 3232H
MOV BX, 6767H
SUB BX,AX
HLT
```

Output:





Result and Inference:

- The AX register initially had our value 3232H and the BX register has 6767H.
- The final result obtained is 3535H.
- The final result is stored in AX register.

iii) 16-bit subtraction with memory access:

Design and Calculations:

For 16-bit subtraction with memory access we need to use the base register and the accumulator register together. The values from desired memory location are moved to accumulator and base register. These values are then added to store the result first in accumulator and then the result can be moved to desired location.

In the code we are accessing the memory location at 1000H and 1002H. The value in 1000H is H and at the

location 1002H is H. The subtracted result is 1010H and is stored in the memory location 1004H

9054

- 7644

1A10

Program Code:

MOV AX, [1000H]

MOV BX, [1002H]

SUB AX, BX

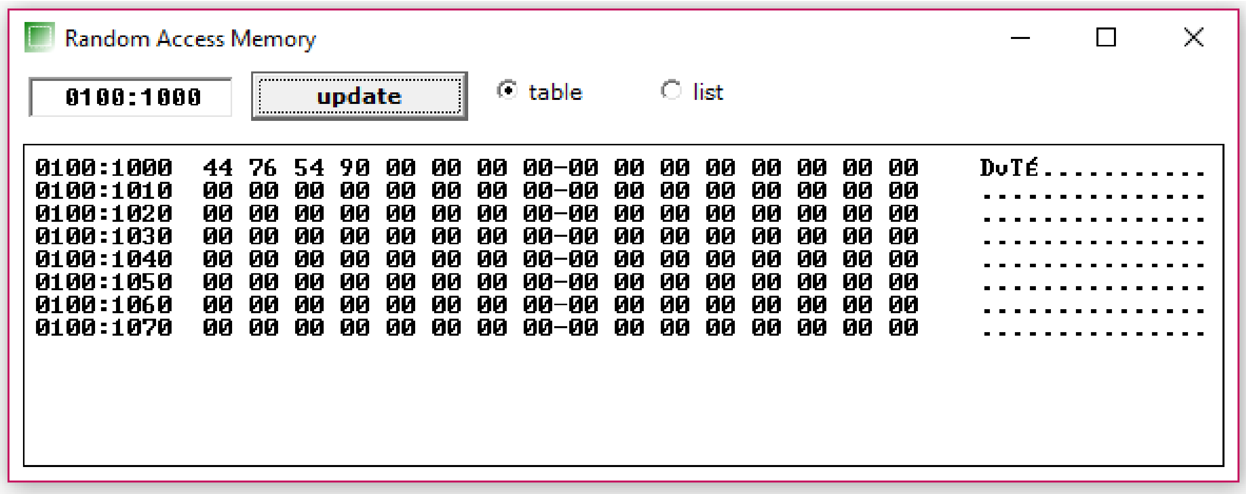
MOV [1004H], AX

HLT

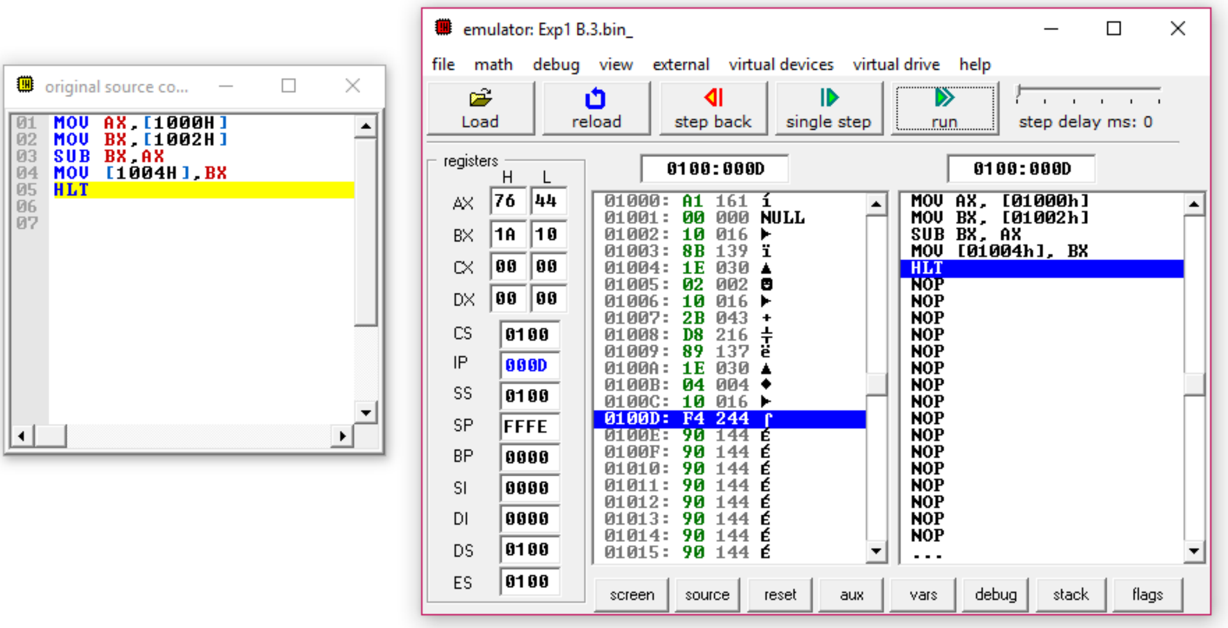
```
MOV AX,[1000H]
MOV BX,[1002H]
SUB BX,AX
MOV [1004H],BX
HLT
```

Output:

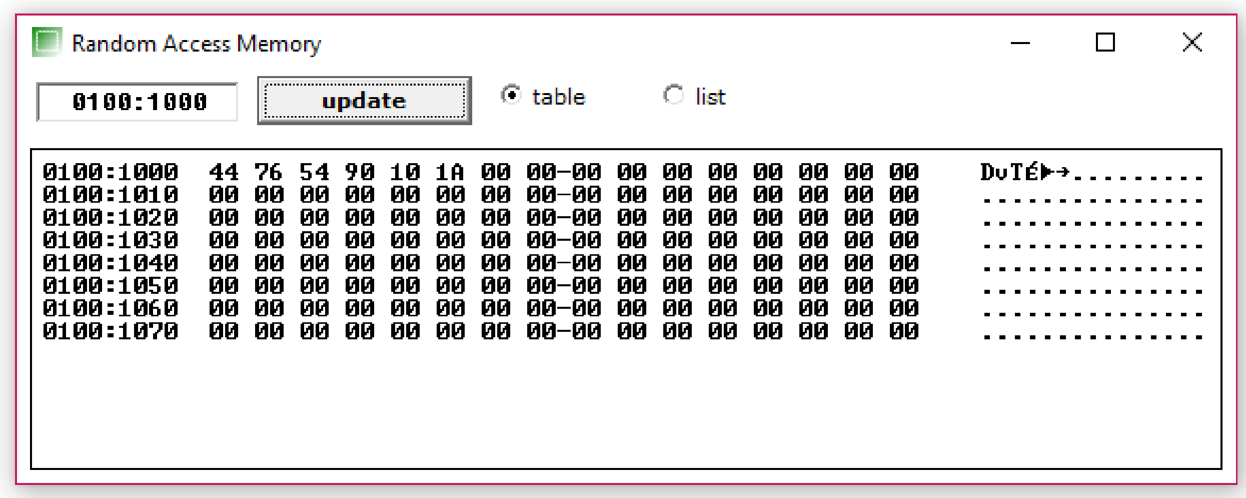
Memory before:



Simulation:



Memory After:



Result and Inference:

- The AX register initially, accessed the location of 1000H which had the value 7644H.
- The BX register had accessed the location of 1002H which had the value of 9054H.
- The final result obtained is 1A10H.
- The final result is stored in AX register and the value is moved to the location 1004H as seen in memory after snippet.

iv) 16-bit subtraction with physical memory access:

Design and Calculations:

For physical memory access we need to first find physical address of the given location. Then we store the first virtual address in accumulator and shift it to data segment register for external memory reference.

The values of segment offset are then updated in accumulator and base register. These registers are then subtracted and the result is stored in accumulator itself. Then the content of accumulator is moved to the specified memory location.

In the code we are accessing the memory location at 2000H and offset 500H and 600H. The value in 500H is 9999H and at the location 600H is 4433H. The result is 5566H and is stored in the memory location 700H

```
   9999
 + 4433
-----------------
   5566
```

Program Code:

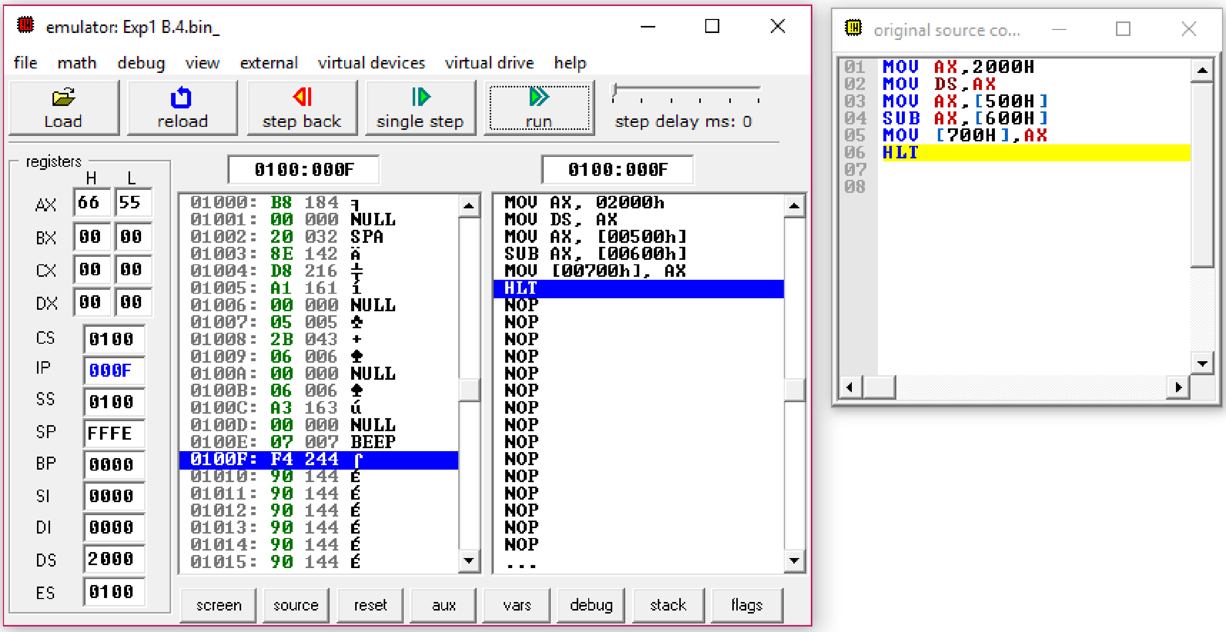
```
MOV AX, 2000H
MOV DS, AX
MOV AX, [500H]
MOV BX, [600H]
SUB AX, BX
MOV [700H], AX
HLT
```

```
MOV AX,2000H
MOV DS,AX
MOV AX,[500H]
SUB AX,[600H]
MOV [700H],AX
HLT
```

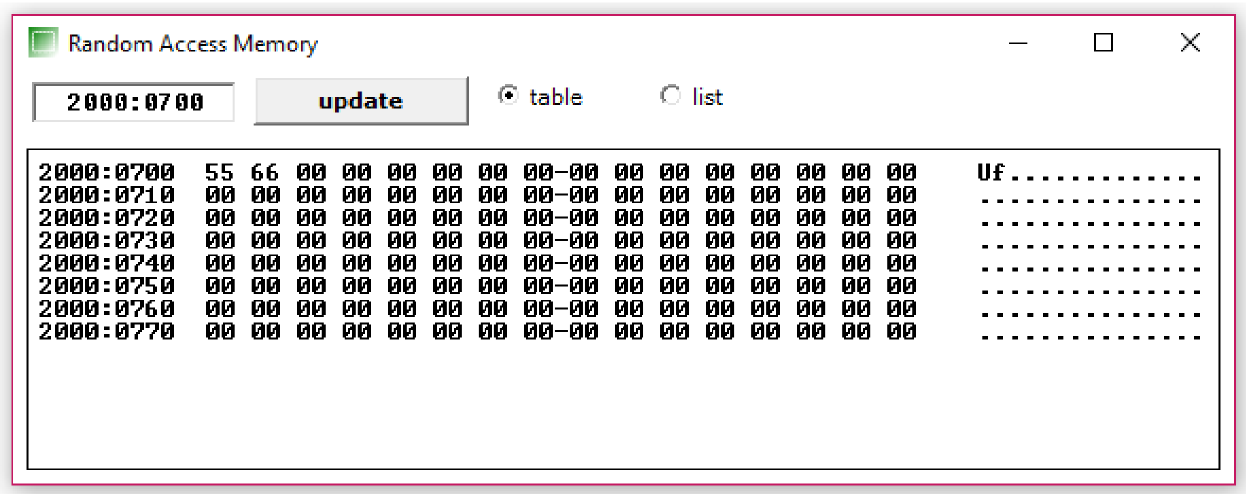
Output:

Memory before:

Simulation:



Memory After:



Result and Inference:

- The AX register initially, accessed the location at 2000H block and 500H segment offset. The value loaded at AX was 9999.
- The BX register had accessed the location of 2000H block and 600H segment offset which had the value of 4433.
- The final result obtained is 5566H.

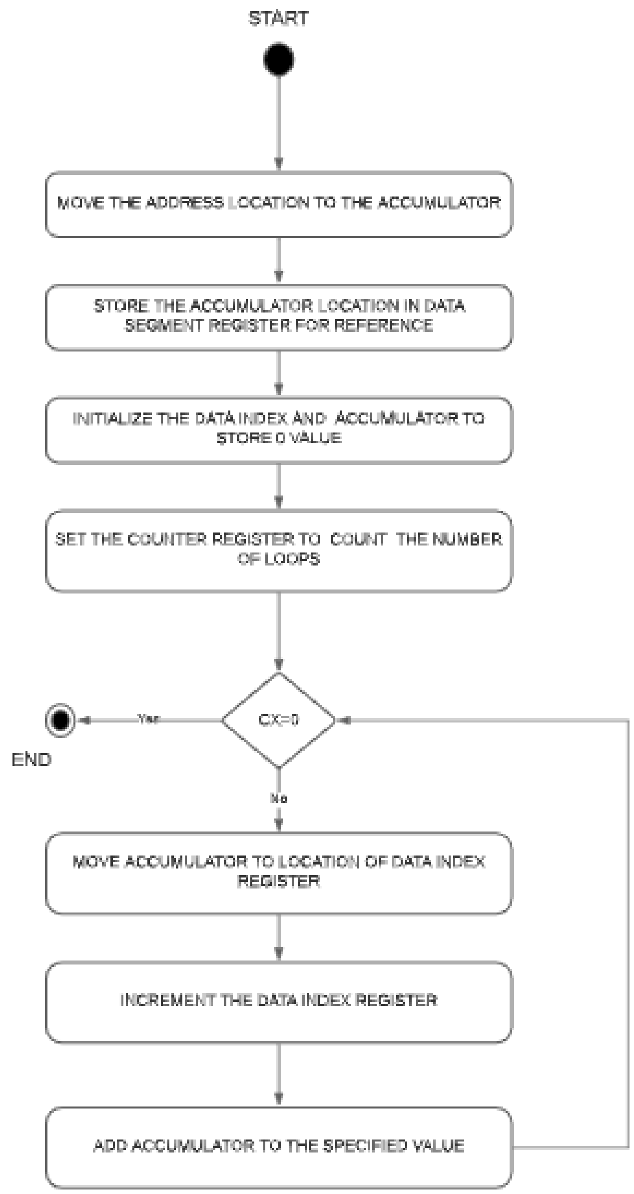
- **The final result is stored in AX register and the value is moved to the location at 2000H block and offset 700H as seen in memory after snippet.**

C.Data-Flow

Algorithm:

1) **Move segment address to accumulator.**
2) **Store accumulator reference in data segment register.**
3) **Reference the Data index register and accumulator register to base location.**
4) **Reference the lower counter register to location where the values are to be added, lower counter register is sufficient because the values to be added are 8-bit.**
5) **Create a label (say ABC) which reference the data index register.**
6) **Increment the data index register by one.**
7) **Add the accumulator with required value (02H our case).**
8) **Goto step 6.**

Flow Chart:


START
MOVE THE ADDRESS LOCATION TO THE ACCUMULATOR
STORE THE ACCUMULATOR LOCATION IN DATA SEGMENT REGISTER FOR REFERENCE
INITIALIZE THE DATA INDEX AND ACCUMULATOR TO STORE 0 VALUE
SET THE COUNTER REGISTER TO COUNT THE NUMBER OF LOOPS
CX=0
Yes
END
No
MOVE ACCUMULATOR TO LOCATION OF DATA INDEX REGISTER
INCREMENT THE DATA INDEX REGISTER
ADD ACCUMULATOR TO THE SPECIFIED VALUE

Design and Calculations:

Here we are moving the starting location to data index register via accumulator. This is to initialize the looping address. Then we initialize the values of accumulator and data index register to 0. The data index register is to point to next location and the accumulator is used to store the value that is to be updated. We then initialize the counter register to 5 that is the number of times we need to run the loop. We then create a label to indicate the start of loop and move the contents of accumulator to location at which data index register in pointing, this implies that 0 will be stored at 3000H location. Then we increment the data index register. Thus, the data index register now points to 3001H location. Also, we increment the accumulator by 02H to now have the value 2. Similarly, we run the loop 5 times to store the value 0, 2, 4, 6 and 8.

- CL = 05H
- Store 00 in 3000H + 0000H (DS + DI)
- DI = 0001H and AX = 0002H
- CL = 04H
- Store 02 in 3000H + 0001H (DS + DI)
- DI = 0002H and AX = 0004H
- CL = 03H
- Store 04 in 3000H + 0002H (DS + DI)
- DI = 0003H and AX = 0006H
- CL = 02H
- Store 06 in 3000H + 0003H (DS + DI)
- DI = 0004H and AX = 0008H
- CL = 01H

- **Store 08 in 3000H + 0004H (DS + DI)**
- **DI = 0005h and AX = 00AH**
- **Cl = 00H**

Since CL==0 Halt the process

Program Code:

MOV AX,

3000H MOV

DS, AX MOV

DI, 0000H

MOV AX,

0000H MOV

CL, 05H

ABC: MOV [DI],

AX INC DI

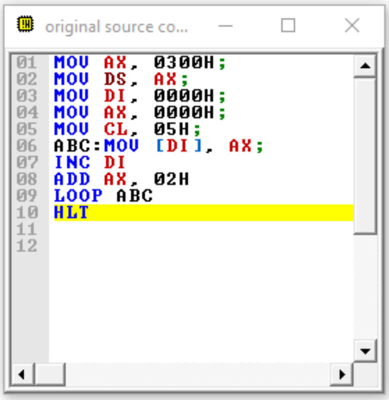
ADD AX,

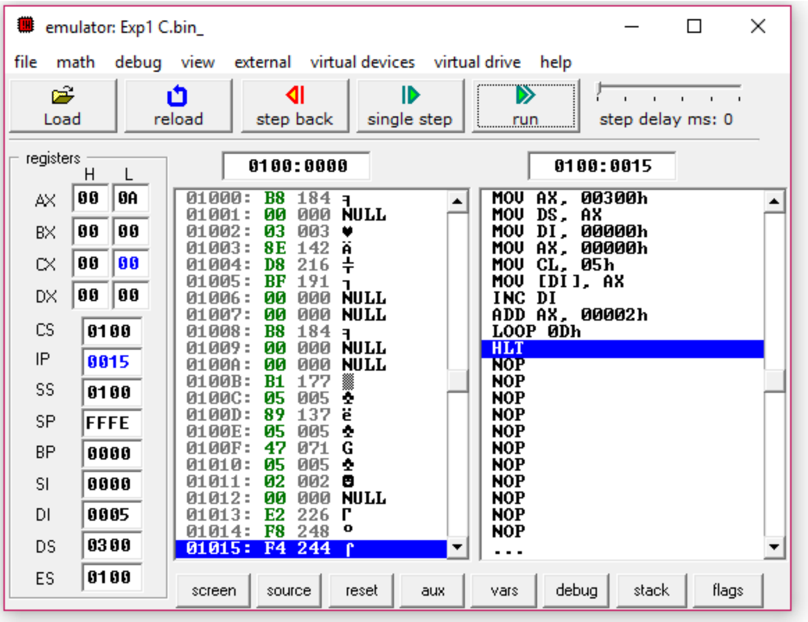
02H LOOP

ABC HLT

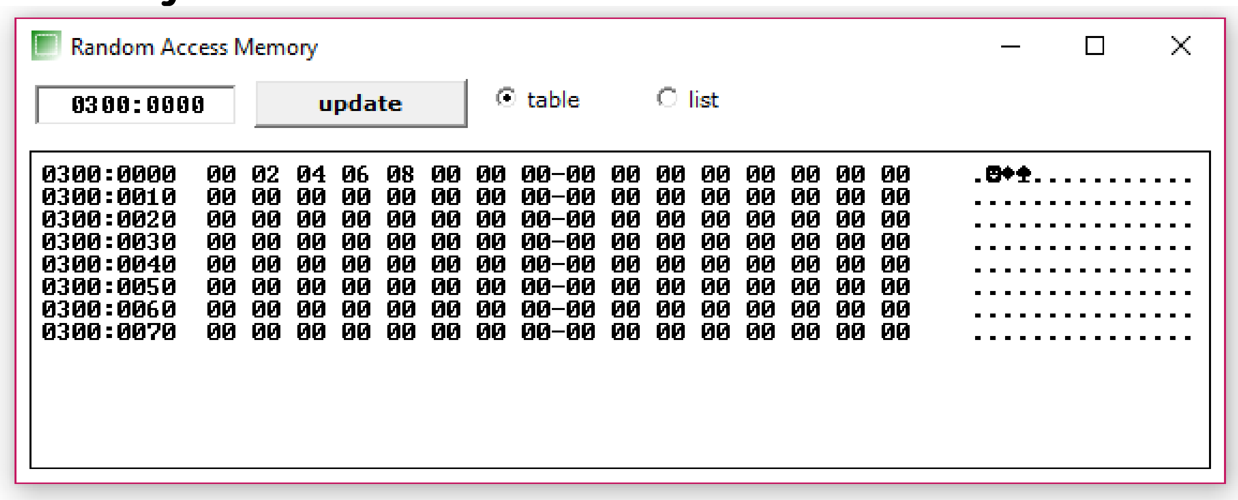
```
MOV AX, 0300H;
MOV DS, AX;
MOV DI, 0000H;
MOV AX, 0000H;
MOV CL, 05H;
ABC:MOV [DI], AX;
INC DI
ADD AX, 02H
LOOP ABC
HLT
```

Simulation:





Memory After:



Result and Inference:

- **The DS register initially stores the 3000H location for reference and the accumulator and data index are storing 0000H.**
- **Also, CL initially stores 05H.**
- **After first loop runs the value 0H is updated at 3000H location and the AX, DI now holds 02H and 01H respectively.**
- **Similarly, the loop runs 5 times.**
- **The location of 3000H, 3001H, 3002H, 3003H and 3004H are stored with values 0,2,4,6 and 8 respectively.**
- **When CL=00H the loop exits.**
- **The value in AX currently is 10H and in DI is 05H.**

Task 2 Date: 09/Aug/2021

Programs involving Multiplication and Division Operations

Aim:

A. Multiplication:

1) Write an Assembly Language Program (ALP) to multiple two numbers of 16-bit data. The input data must load to the location given below and the output product should be stored as per the memory location given below.

Input		Output	
Memory Address	Content	Memory Address	Content
1100	1A	1104	
1101	EF	1105	
1102	50	1106	
1103	CD	1107	

2) Write an Assembly Language Program (ALP) to multiple two numbers of 16-digit data based on your roll number.

Input		Output	
Memory Address	Content	Memory Address	Content
2100	1A	2104	
2101	EF	2105	
2102	50	2106	
2103	CD	2107	

B. Division:

1) Write an Assembly Level Program (ALP) to divide 32-bit data by 16-bit data. The input data must load to the location given below, the output quotient and remainder should be stored as per the memory location given below.

Input		Output	
Memory Add	Content	Memory Add	Content
3100	0A	3106	
3101	58	3107	
3102	C2	3108	
3103	71	3109	
3104	F2		
3105	F6		

2) Write an Assembly Language Program (ALP) to divide 32-bit data by 16-bit data based on your roll number.

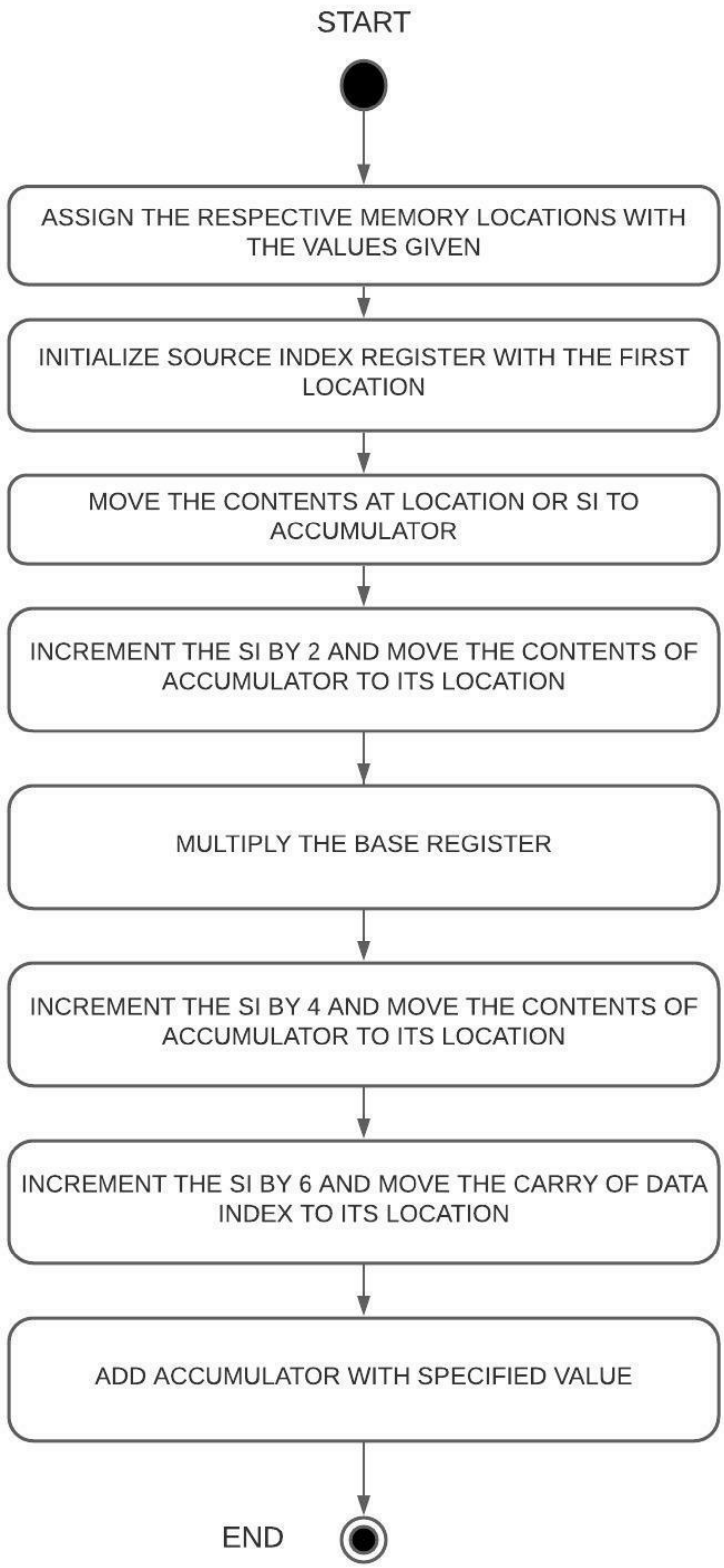
Tool Used: emu8086 simulator

A. Multiplication

Algorithm:

1) Move all the values in the given memory locations.
2) Move the smallest memory location to SI register for reference.
3) Move data at location SI to accumulator register (AX).
4) Increment the SI value by 2 to access the next memory location.
5) Move the data to base register (BX).
6) Multiply the base register. This will multiply BX with AX.
7) The result of above operation is stored in accumulator itself.
8) Increment the SI by 4 and move to the given location.
9) Increment the SI by 6 and move the carry of Data register to the new memory location of SI.

Flow Chart:


START
ASSIGN THE RESPECTIVE MEMORY LOCATIONS WITH THE VALUES GIVEN
INITIALIZE SOURCE INDEX REGISTER WITH THE FIRST LOCATION
MOVE THE CONTENTS AT LOCATION OR SI TO ACCUMULATOR
INCREMENT THE SI BY 2 AND MOVE THE CONTENTS OF ACCUMULATOR TO ITS LOCATION
MULTIPLY THE BASE REGISTER
INCREMENT THE SI BY 4 AND MOVE THE CONTENTS OF ACCUMULATOR TO ITS LOCATION
INCREMENT THE SI BY 6 AND MOVE THE CARRY OF DATA INDEX TO ITS LOCATION
ADD ACCUMULATOR WITH SPECIFIED VALUE
END

i) Specified value multiplication:

Design and Calculations:

The source index register is used as reference for the location to point at and stores the memory location of 1100H. We then store the values mentioned to the specific locations of 1100H, 1101H, 1102H, 1103H. We then move the data from memory to accumulator.

AX: EF1A from 1100H

BX: CD50 from 1101H

We then multiply the base register which stores the result in accumulator and the carry in data register. Hence, we move the contents of accumulator and data register to the specified location of 1104H and 1106H.

```
A1)     EF1A
      x CD50
      ------
        0000
       4AB82
      C2452
     B3538
    --------
    BFC28A20
    --------
   result = BFC28A20
```


Program Code:

MOV [1100H], 1AH

MOV [1101H], EFH

MOV [1102H], 50H

MOV [1103H], CDH

MOV SI, 1100H

MOV AX, [SI]

MOV BX, [SI+2]

MUL BX

MOV [SI+4], AX

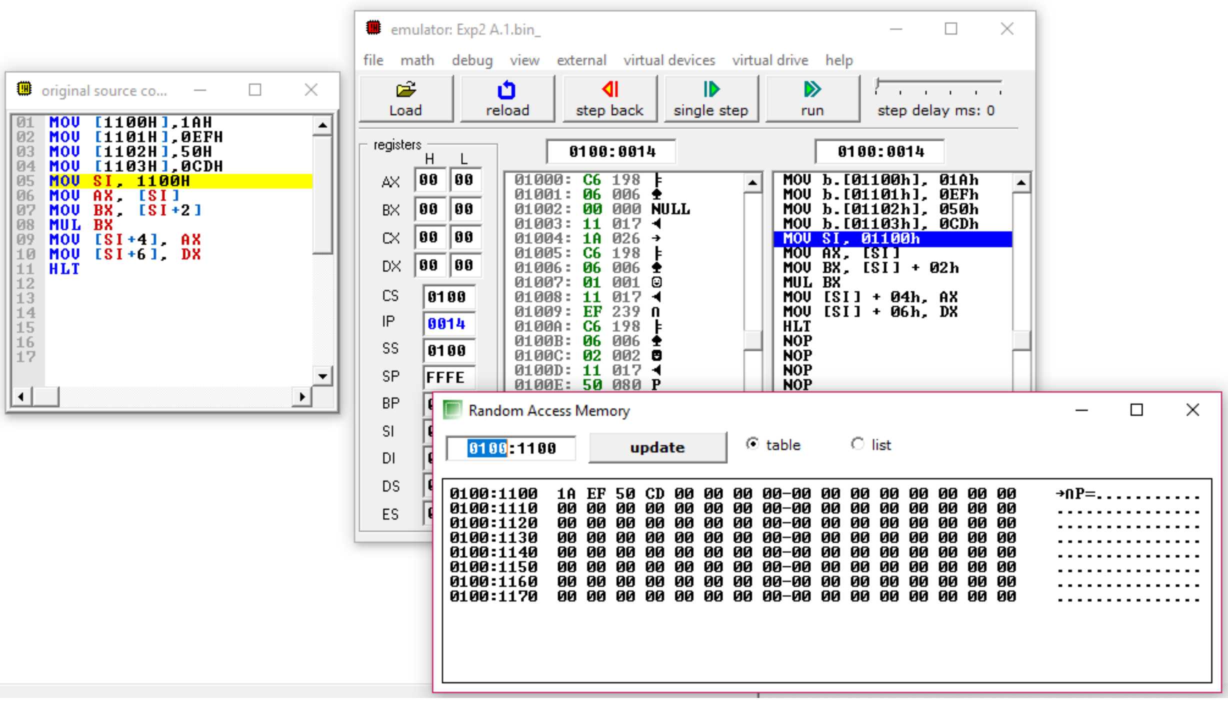
MOV [SI+6], DX

HLT

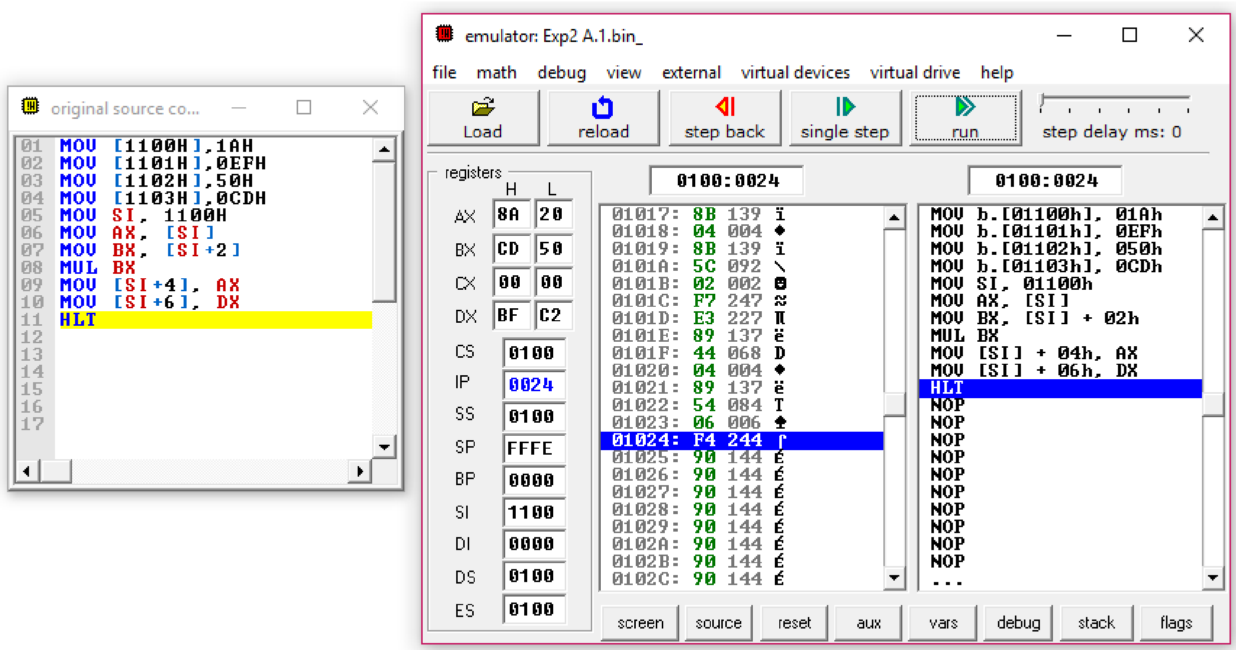
```
MOV [1100H],1AH
MOV [1101H],0EFH
MOV [1102H],50H
MOV [1103H],0CDH
MOV SI, 1100H
MOV AX, [SI]
MOV BX, [SI+2]
MUL BX
MOV [SI+4], AX
MOV [SI+6], DX
HLT
```

Output:

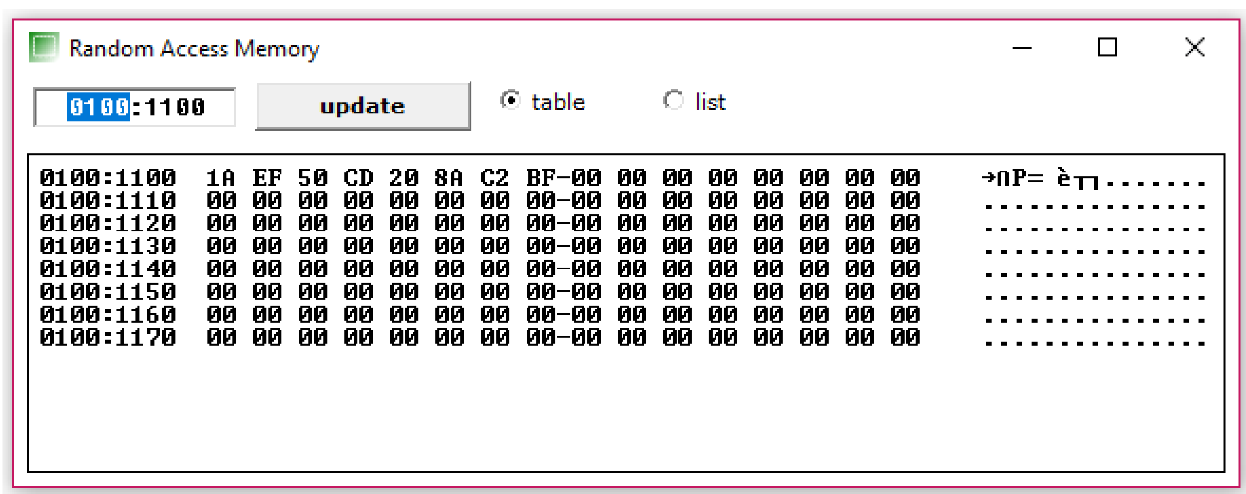
Memory after loading the values:



Simulation:



Memory After:



Result and Inference:

- The accumulator initially had the value EF1A.
- The base register initially had the value CD50.
- The result BFC28A20 is stored in the memory locations of 1104H and 1106H.
- Data at the location are:
  1104H: 20H
  1105H: 8AH
  1106H: C2H
  1107H: BFH
- Hence the result is BFC28A20 as expected.

ii) Registration number based multiplication:

Design and Calculations:

Here we are going to need Source index register, accumulator, base register and the data register. The source index register is used as reference for the location to point at and stores the memory location of 2100H. We then store the values mentioned to the specific locations of 2100H, 2101H, 2102H, 2103H. We then move the data from given memory location to accumulator

My registration number is 19BCE2555 and hence-

AX: 9BCE from 2100H

BX: 2555 from 2101H

We then multiply the base register which stores the result in accumulator and the carry in data register. Hence, we move the contents of accumulator and data register to the specified location of 2104H and 2106H.

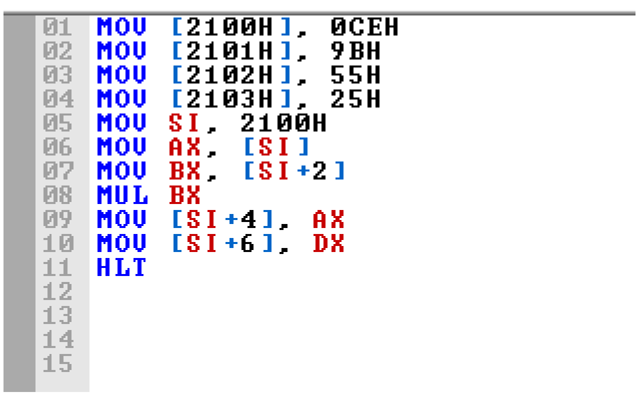
A2)

```
        9BCE
      x 2555
     ---------
       30B06
      30B06
     30B06
    1379C
   ----------
    16B88166
   ----------
```

result : 16B88166

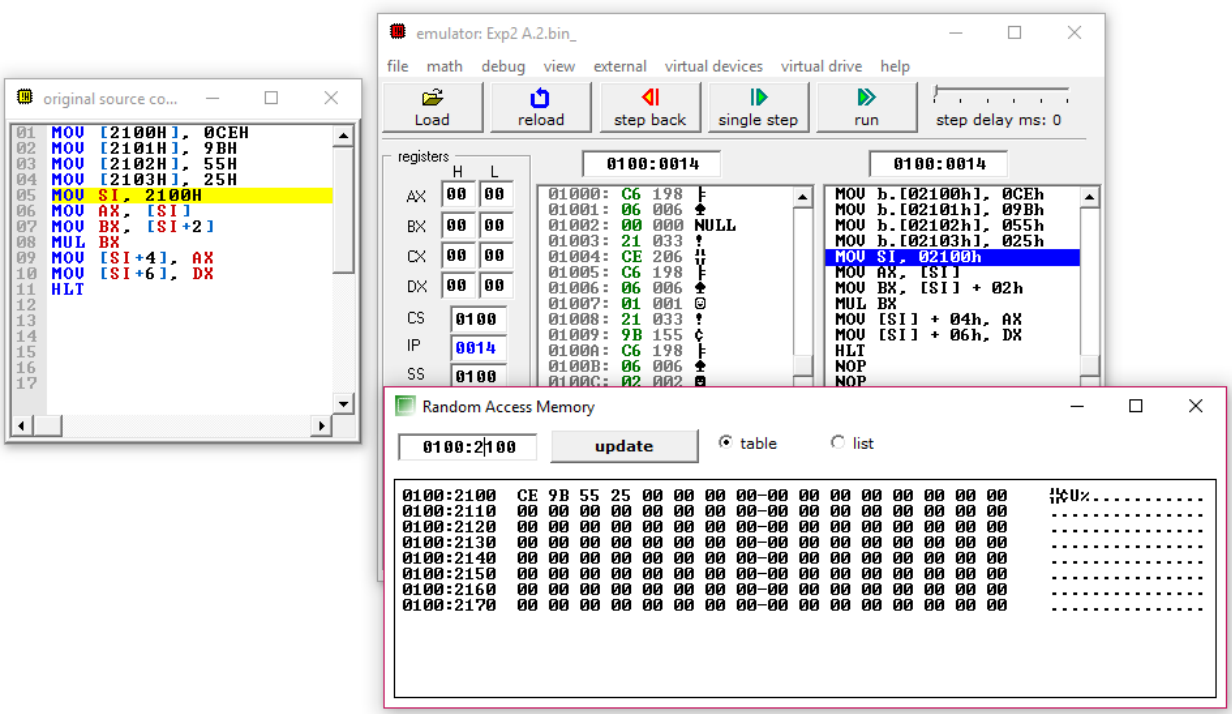
Program Code:

```
MOV [2100H], 0CEH
MOV [2101H], 9BH
MOV [2102H], 55H
MOV [2103H], 25H
MOV SI, 2100H
MOV AX, [SI]
MOV BX, [SI+2]
MUL BX
MOV [SI+4], AX
MOV [SI+6], DX
HLT
```

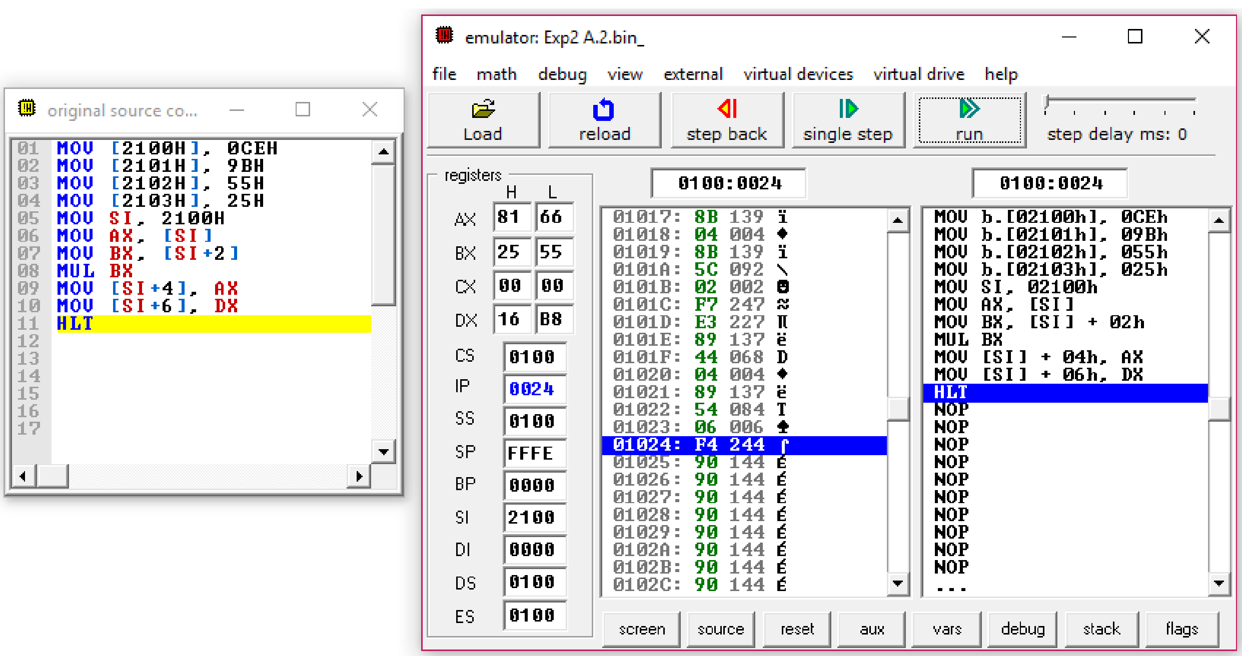


Output:

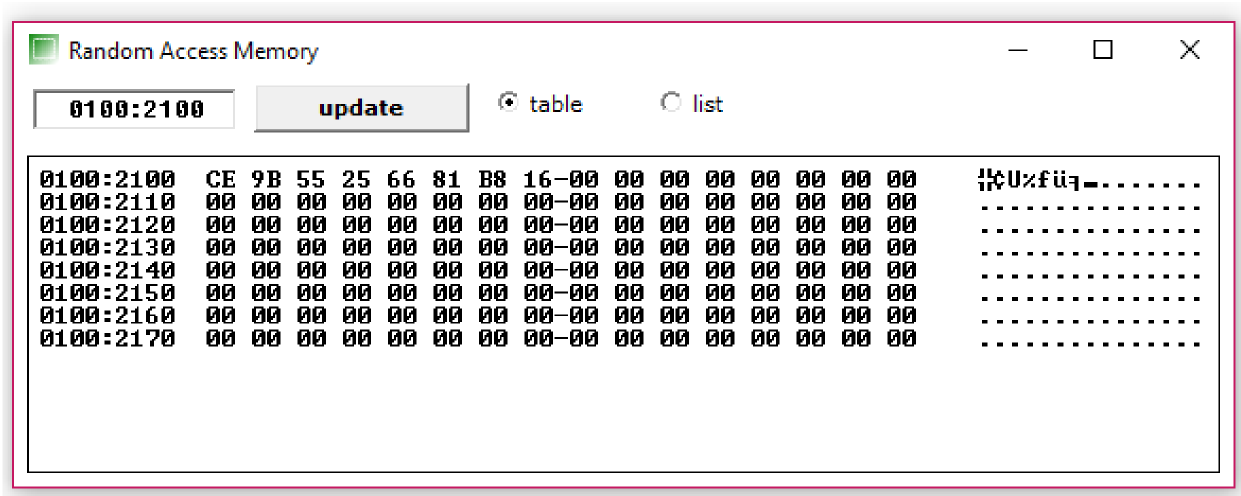
Memory after loading the values:



Simulation:



Memory After:



Result and Inference:

- **The accumulator initially had 9BCE.**
- **The base register initially had 2555.**
- **The expected result of 16B88166 is stored in the memory locations of 2104H and 2106H.**
- **Data at the location are:**
  2104H: 66
  2105H: 81
  2106H: B8
  2107H: 16
- **Hence the result is 16B88166 as expected.**

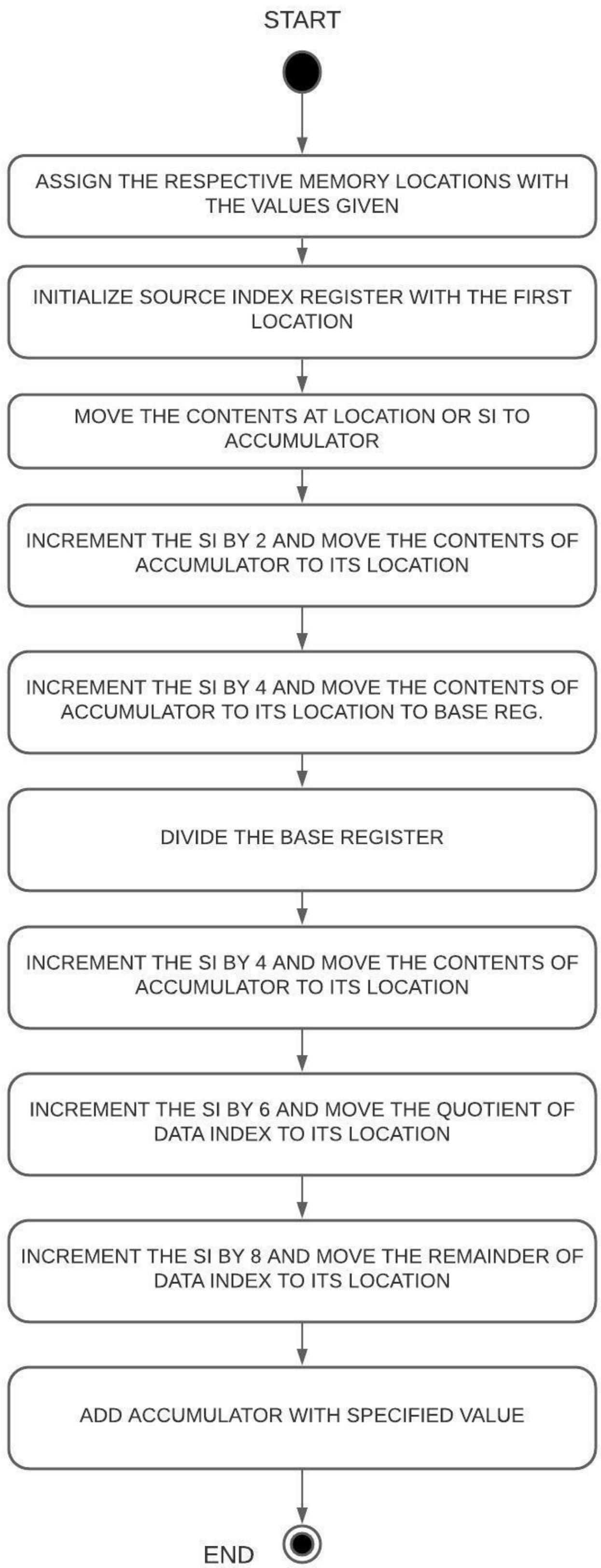
B. Division

Algorithm:

1) Move all the values in the specified memory locations.
2) Move the starting memory to SI register for reference.
3) Move the contents at location SI to accumulator register (AX).
4) Increment the SI value by 2 in order to point to the next memory location.
5) Increment the SI value by 4 and move the next contents to base register (BX).
6) Divide the base register. This will store the quotient in AX.
7) Move the contents of accumulator register to specified memory location

8) The remainder is stored in Data register and we move it to the specified location

Flow Chart:


START
ASSIGN THE RESPECTIVE MEMORY LOCATIONS WITH THE VALUES GIVEN
INITIALIZE SOURCE INDEX REGISTER WITH THE FIRST LOCATION
MOVE THE CONTENTS AT LOCATION OR SI TO ACCUMULATOR
INCREMENT THE SI BY 2 AND MOVE THE CONTENTS OF ACCUMULATOR TO ITS LOCATION
INCREMENT THE SI BY 4 AND MOVE THE CONTENTS OF ACCUMULATOR TO ITS LOCATION TO BASE REG.
DIVIDE THE BASE REGISTER
INCREMENT THE SI BY 4 AND MOVE THE CONTENTS OF ACCUMULATOR TO ITS LOCATION
INCREMENT THE SI BY 6 AND MOVE THE QUOTIENT OF DATA INDEX TO ITS LOCATION
INCREMENT THE SI BY 8 AND MOVE THE REMAINDER OF DATA INDEX TO ITS LOCATION
ADD ACCUMULATOR WITH SPECIFIED VALUE
END

i) Specified value division:

Design and Calculations:

Here we are going to need Source index register, accumulator, base register and the data register. The source index register is used as reference for the location to point at and stores the memory location of 3100H. We then store the values mentioned to the specific locations of 3100H, 3101H, 3102H, 3103H, 3104H, 3105H. We then move the data to accumulator

Hence values are-

Value of AX: 580A

Value of DX: 71C2

Value of BX: F6F2

We then divide the base register which stores the quotient in accumulator and the remainder in data register. Hence, we move the contents of accumulator and data register to the specified location of 3106H and 3108H.

B1)

F6F2) 71C2580A (75EE
6C09E
5B878
4D2BA
E5BF0
D813C
DAA4A
D813C
$\boxed{290E}$

Quotient : 75EE

Remainder: 290E

Program Code:

```
MOV [3100H], 0AH
MOV [3101H], 58H
MOV [3102H], 0C2H
MOV [3103H], 71H
MOV [3104H], 0F2H
MOV [3105H], 0F6H
MOV SI, 3100H
```

MOV AX, [SI]

MOV DX, [SI+2]

MOV BX, [SI+4]

DIV BX

MOV [SI+6], AX

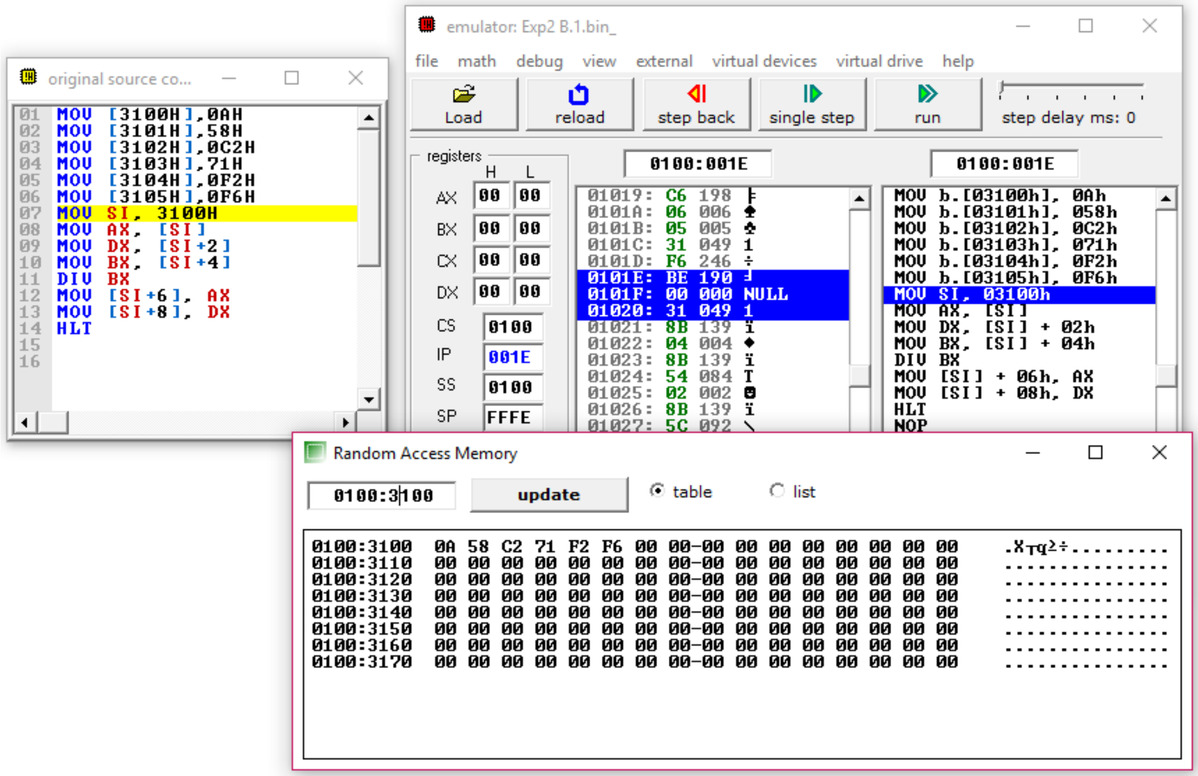
MOV [SI+8], DX

HLT

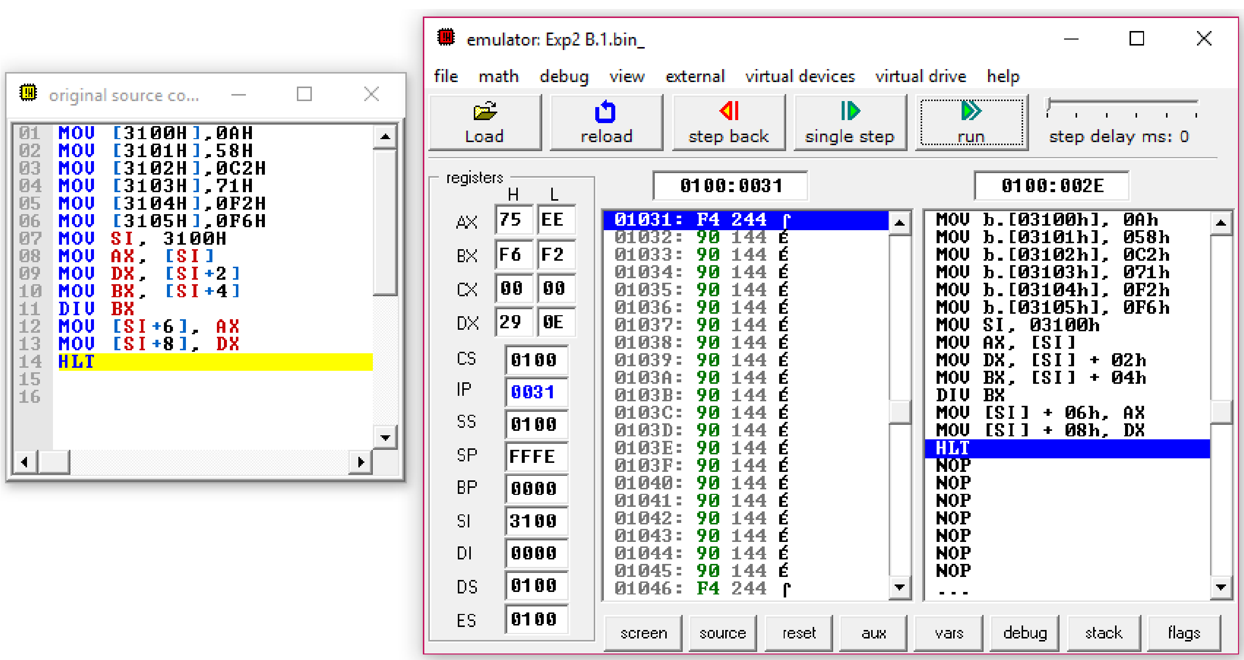
```
MOV [3100H],0AH
MOV [3101H],58H
MOV [3102H],0C2H
MOV [3103H],71H
MOV [3104H],0F2H
MOV [3105H],0F6H
MOV SI, 3100H
MOV AX, [SI]
MOV DX, [SI+2]
MOV BX, [SI+4]
DIV BX
MOV [SI+6], AX
MOV [SI+8], DX
HLT
```

Output:

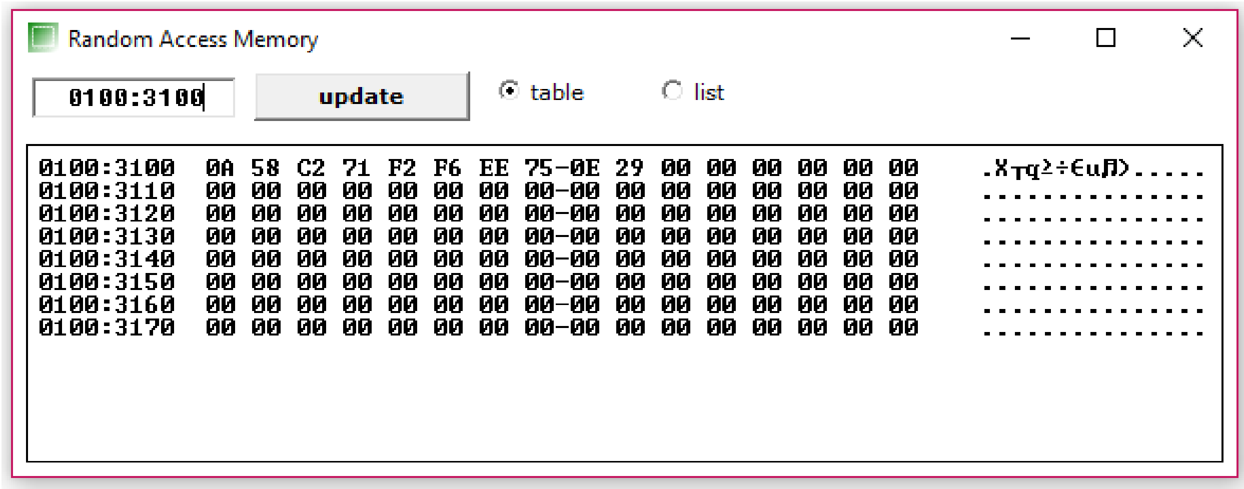
Memory after loading the values:



Simulation:



Memory After:



Result and Inference:

- **The accumulator initially had 580A.**
- **The data register initially had 71C2.**
- **The base register initially had F6F2.**
- **The expected quotient of 75EE is stored in the memory location of 3106H.**
- **The expected remainder of 290E is stored in the memory location of 3108H.**
- **Data at the location are:**
  3106H: EE
  3107H: 75
  3108H: 0E
  3109H: 29
- **Hence the quotient is 75EE and remainder is 290E as expected.**

ii) Registration number-based division:

Design and Calculations:

Here we are going to need Source index register, accumulator, base register and the data register. The source index register is used as reference for the location to point at and stores the memory location of 4100H. We then store the values mentioned to the specific locations of 4100H, 4101H, 4102H, 4103H, 4104H, 4105H. We then move data to accumulator. Hence current values are-

Value of AX: 2555

Value of DX: 9BCE

Value of BX: FFF1(Divisor)

We then divide the base register which stores the quotient in accumulator and the remainder in data register. Hence, we move the contents of accumulator and data register to the specified location of 4106H and 4108H.

B2)

FFF1) 9BCE2555 (9BD7
8FF79
BD695
AFF5B
D73A5
CFF3D
74685
6FF97
$\boxed{46EE}$

Quotient : 9BD7

Remainder: 46EE

Program Code:

MOV [4100H],55H

MOV [4101H],25H

MOV [4102H],0CEH

MOV [4103H],9BH

MOV [4104H],0F1H

MOV [4105H],0FFH

MOV SI, 4100H

MOV AX, [SI]

MOV DX, [SI+2]

MOV BX, [SI+4]

DIV BX

MOV [SI+6], AX

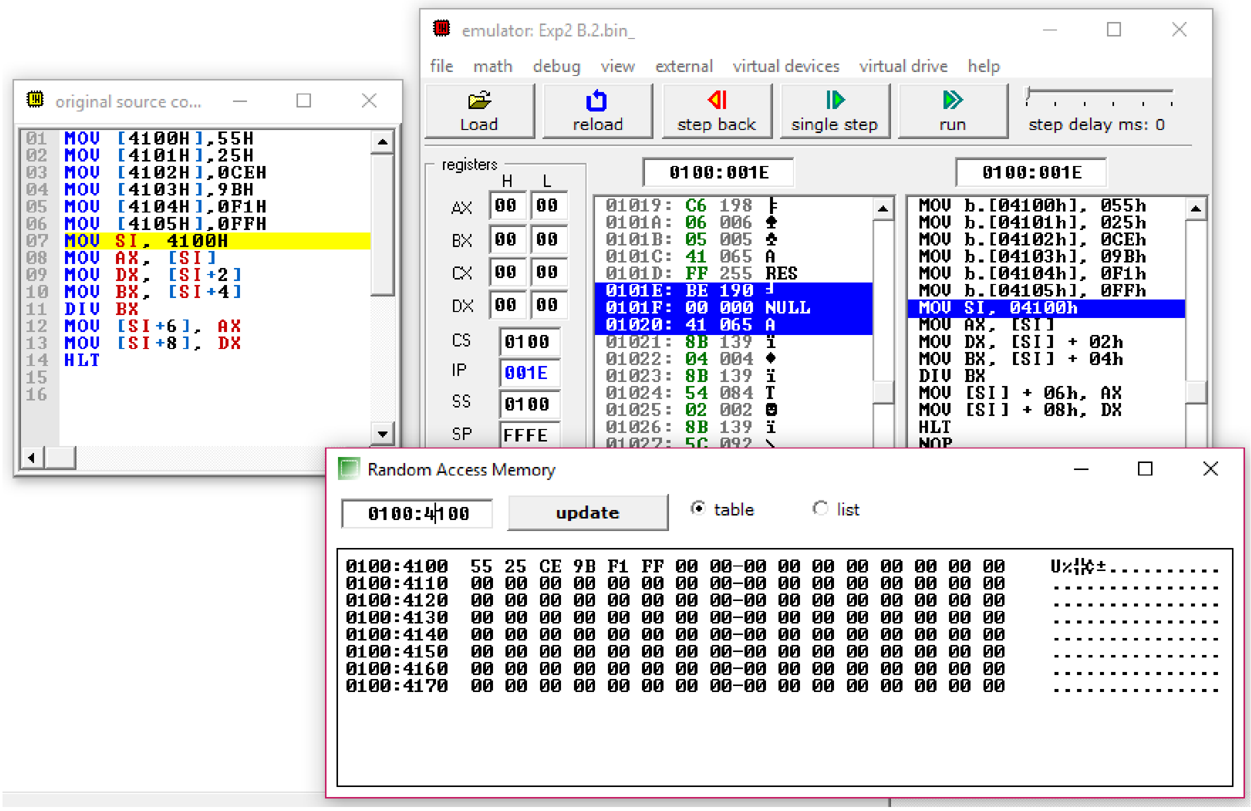
MOV [SI+8], DX

HLT

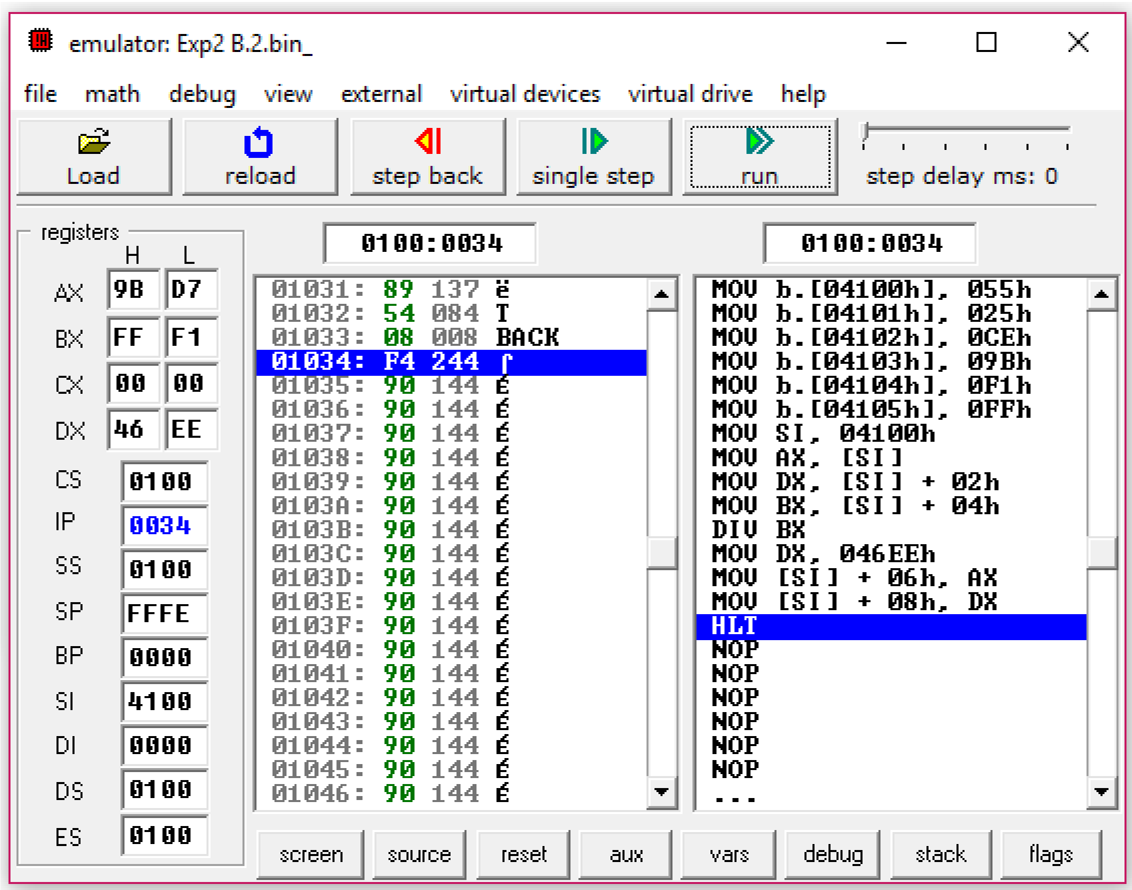
```
MOV [4100H],55H
MOV [4101H],25H
MOV [4102H],0CEH
MOV [4103H],9BH
MOV [4104H],0F1H
MOV [4105H],0FFH
MOV SI, 4100H
MOV AX, [SI]
MOV DX, [SI+2]
MOV BX, [SI+4]
DIV BX
MOV [SI+6], AX
MOV [SI+8], DX
HLT
```

Output:

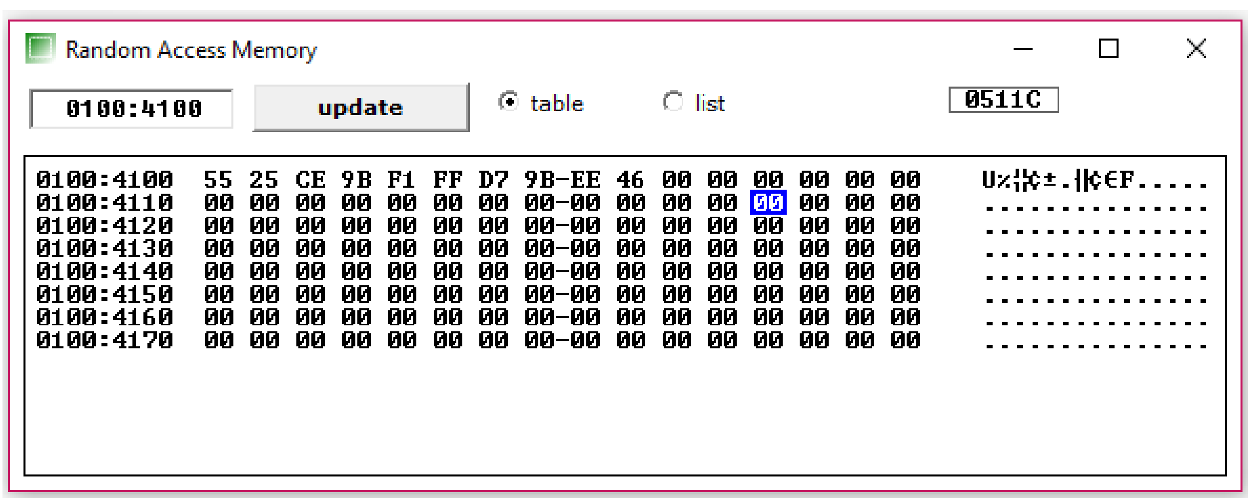
Memory after loading the values:



Simulation:



Memory After:



Result and Inference:

- **The accumulator initially had 2555.**
- **The data register initially had 9BCE.**
- **The base register initially had FFF1.**
- **The quotient 9BD7 is stored in the memory location of 4106H.**
- **The remainder 46EE is stored in the memory location of 4108H.**
- **Data at the location are:**
  4106H: D7
  4107H: 9B
  4108H: EE
  4109H: 46
- **Hence the quotient is 9BD7 and remainder is 46EE as expected.**