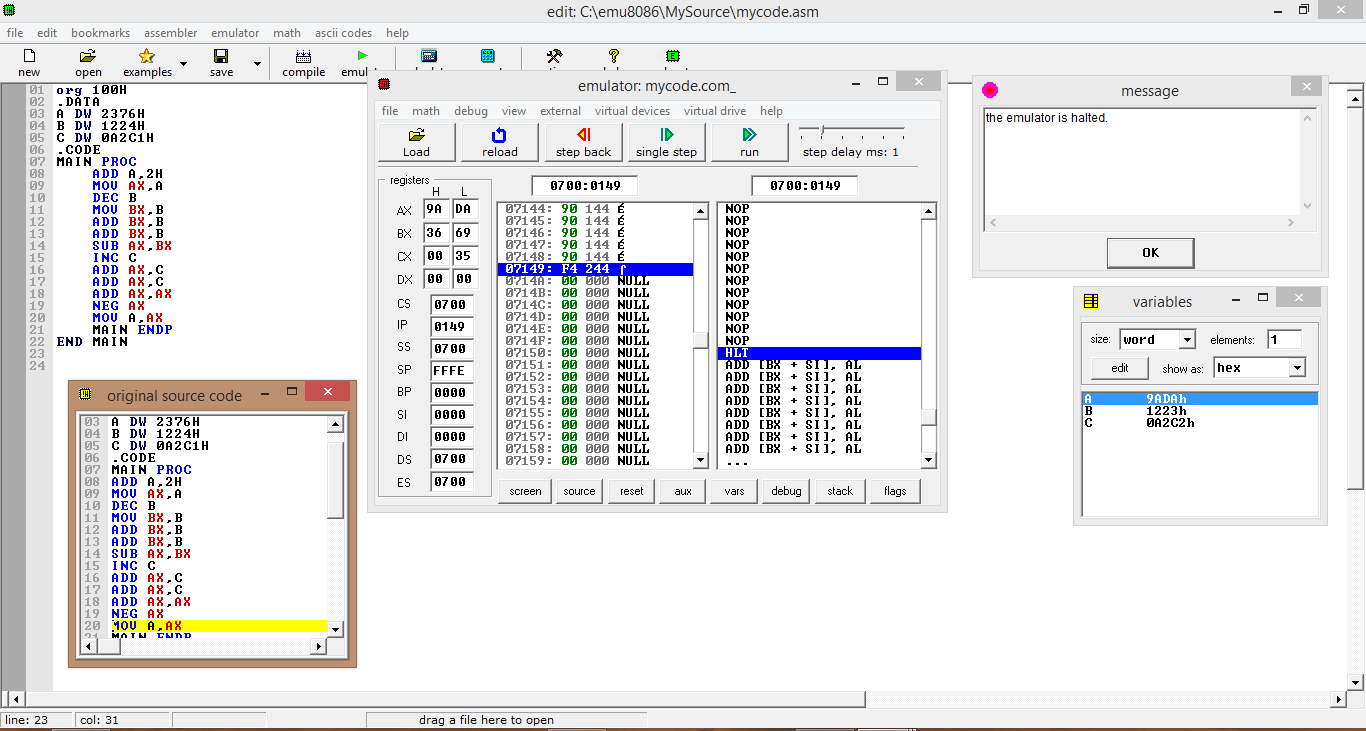
**Class Assignment:**

The following high-level language is to be translated into assembly language,

A= -2((A+2)-3(B-1)+2(C+1))

Where A, B and C are variables.

**Code and Result:**

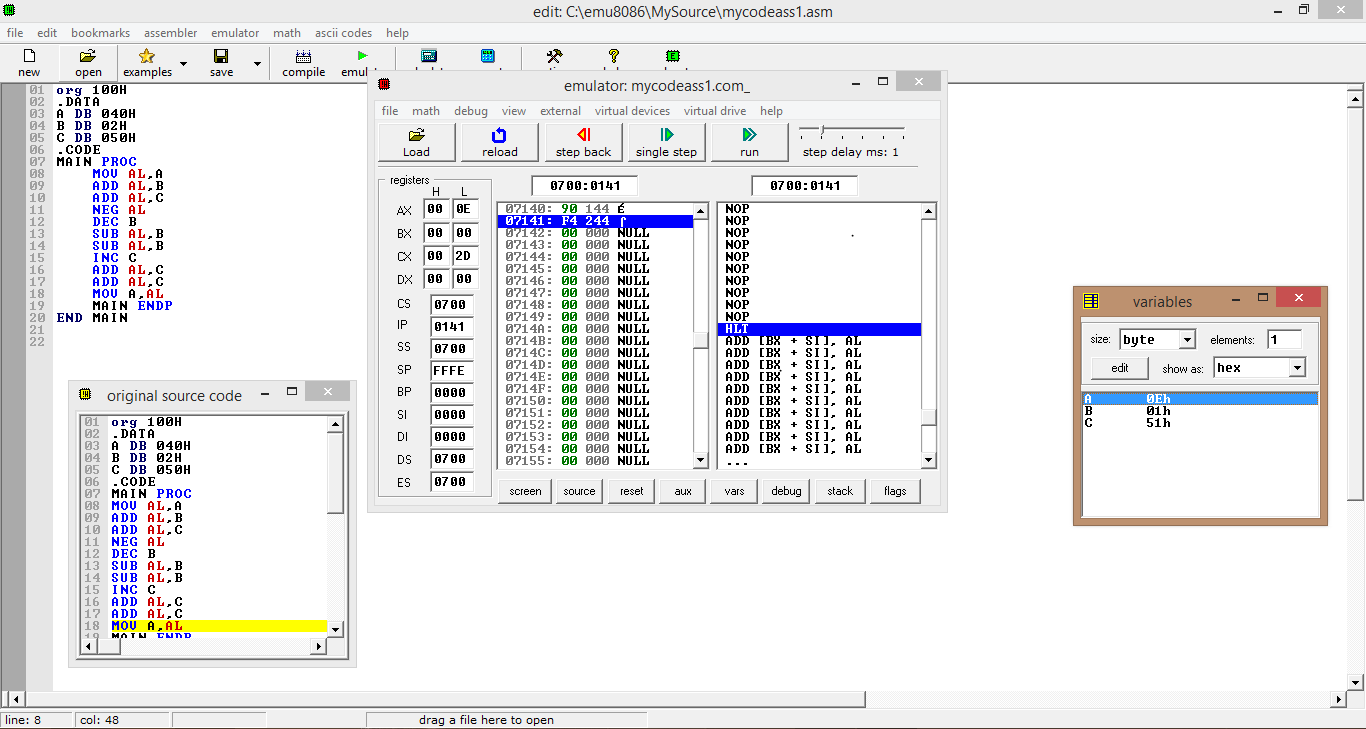


**Home Assignment:**

The following high-level language is to be translated into assembly language,

A= -(A+B+C)-2(B-1)+2(C+1))

Where A, B and C are variables.

**Code and result**:

**1**

**Class Assignment:**

**Program 1: Condition of Status Flags:**

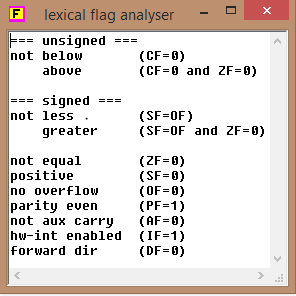
**Code:**

CODE SEGMENT

ASSUME CS:CODE, DS:CODE

MOV AX,22B7H

MOV BX,5A27H

 MOV CX,84C5H

MOV DX,0000H

ADD AX,BX

OR AX,BX

XOR AX,CX

NOT AX

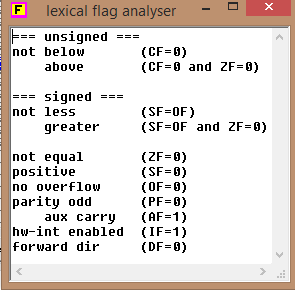
TEST CX,DX

AND CX,AX

HLT

CODE ENDS

END



**Program 2:**

**Code:**

CODE SEGMENT

ASSUME CS:CODE, DS:CODE

MOV AX,0FFFH

MOV CX,0FFFH

ADD AX,CX

HLT

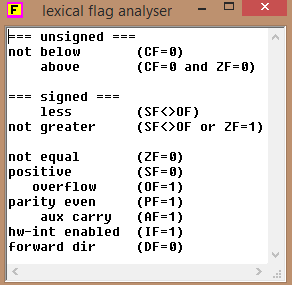
CODE ENDS

END

**Program 3:**

**Code: Condition of Status Flags:**

CODE SEGMENT

 ASSUME CS:CODE, DS:CODE

MOV AX,8000H

MOV BX,0001H

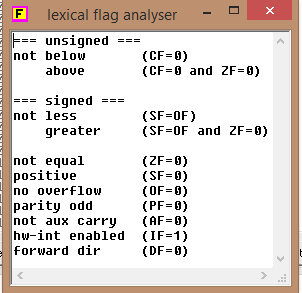
SUB AX,BX

HLT

CODE ENDS

END

**Program 4:**

**Code:**

CODE SEGMENT

ASSUME CS:CODE, DS:CODE

MOV AL,80H

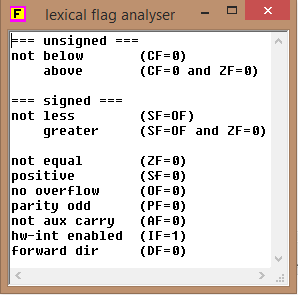
MOV BL,80H

ADD AL,BL

HLT

CODE ENDS

END



**Program 5:**

**Code:**

CODE SEGMENT

ASSUME CS:CODE, DS:CODE

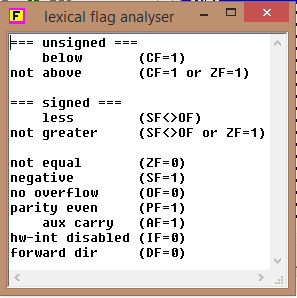
MOV AL,0FFH

INC AL

HLT

CODE ENDS

END



**Program 4:**

**Code:**

ORG 100H

.CODE

MAIN PROC

MOV AX,4000H

ADD AX,AX

SUB AX,0FFFH

NEG AX

INC AX

MOV AH,4CH

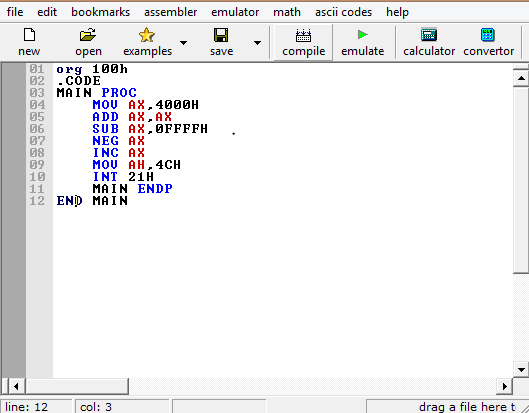
INT 21H

MAIN ENDP

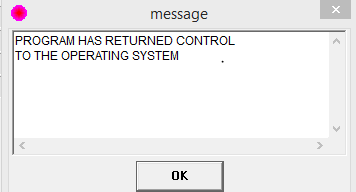
END MAIN

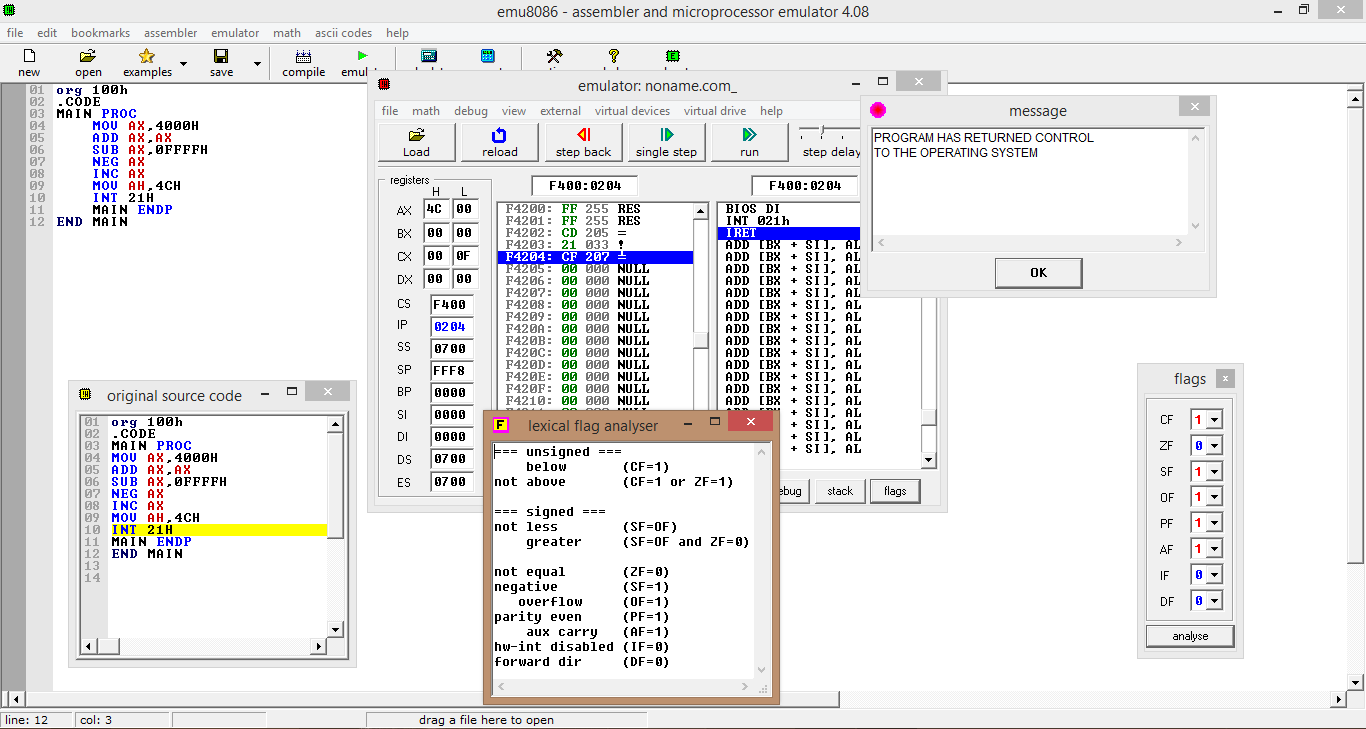
**Home Assignment:**

Run the following code on single step mode and write down the values of registers and status flags

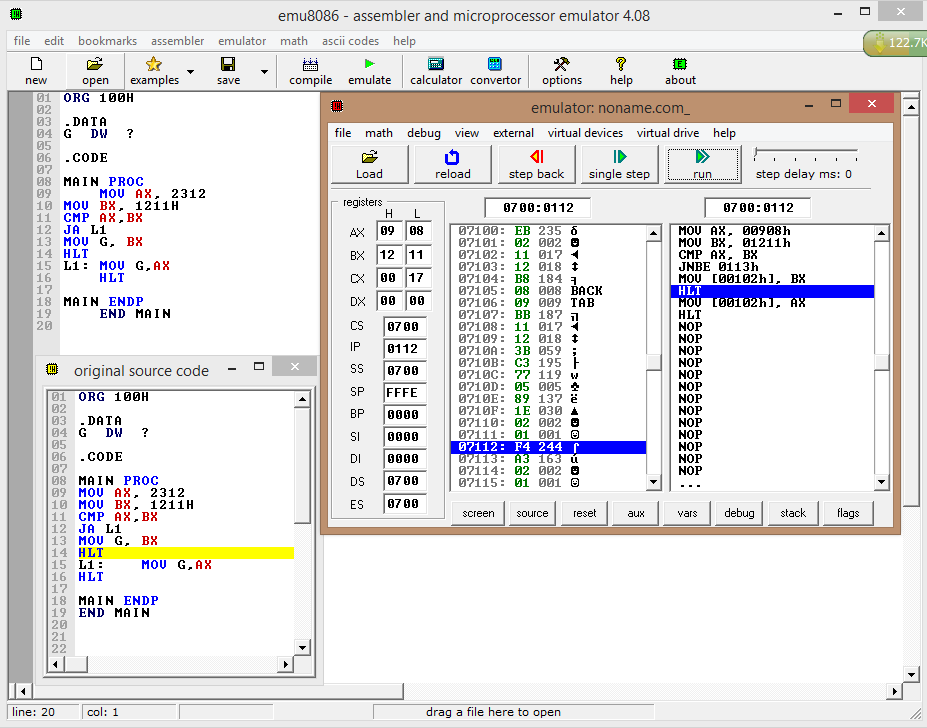


**Results:**



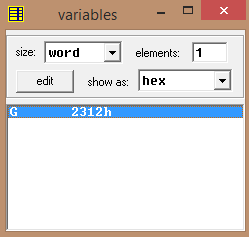


2

**Class Assignment**

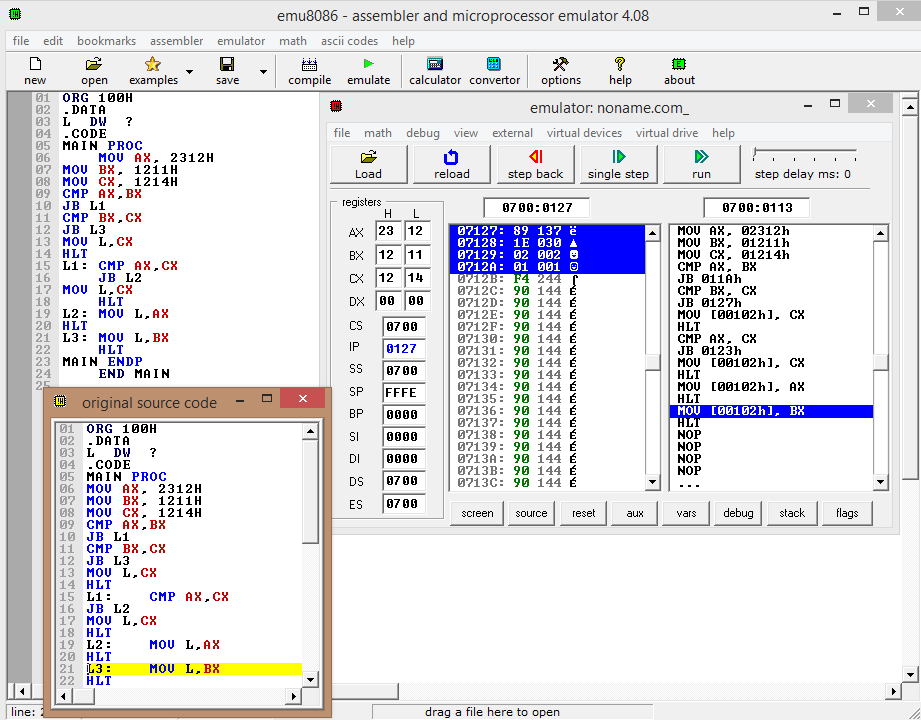
Program:1

Result:

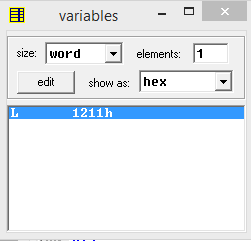


Emulator registered the Greater number on variable G.

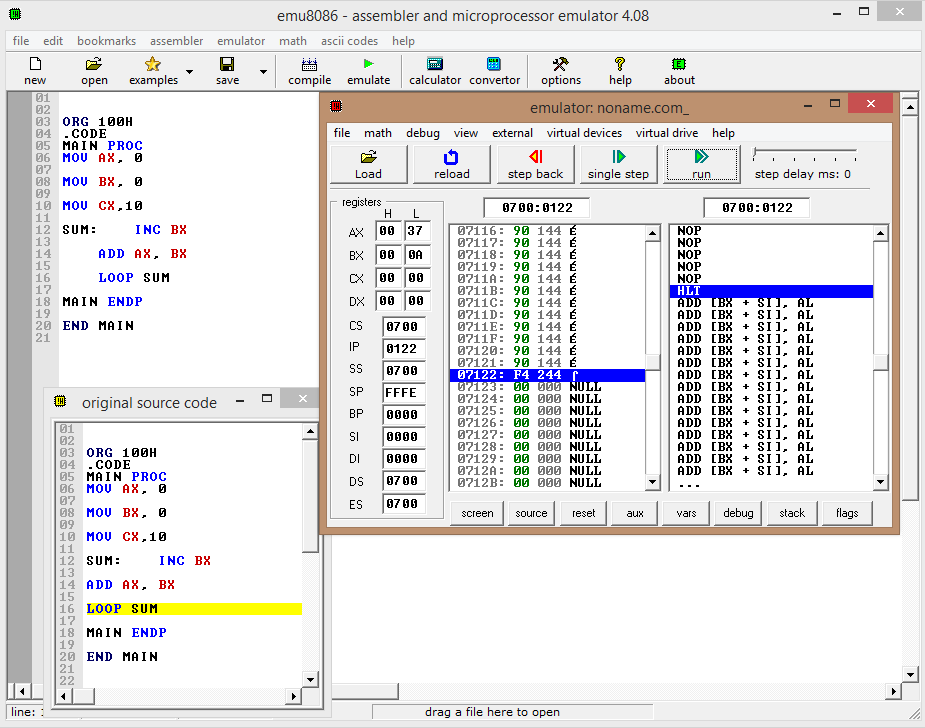
Program:2



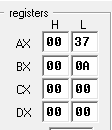
Result:



Emulator registered the lowest number on variable L.

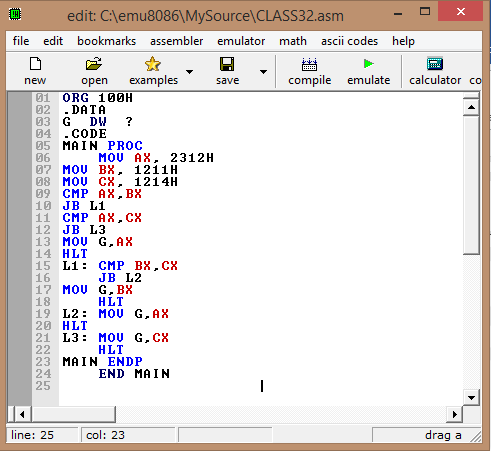
 **Program 3:**

**Results:**

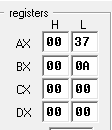


The sum of the numbers was registered on AX =37h

**Home Assignment:**

1. **Write an assembly code that determines the greatest of three numbers stored in AX, BX and CX. Put t**

**Results:**



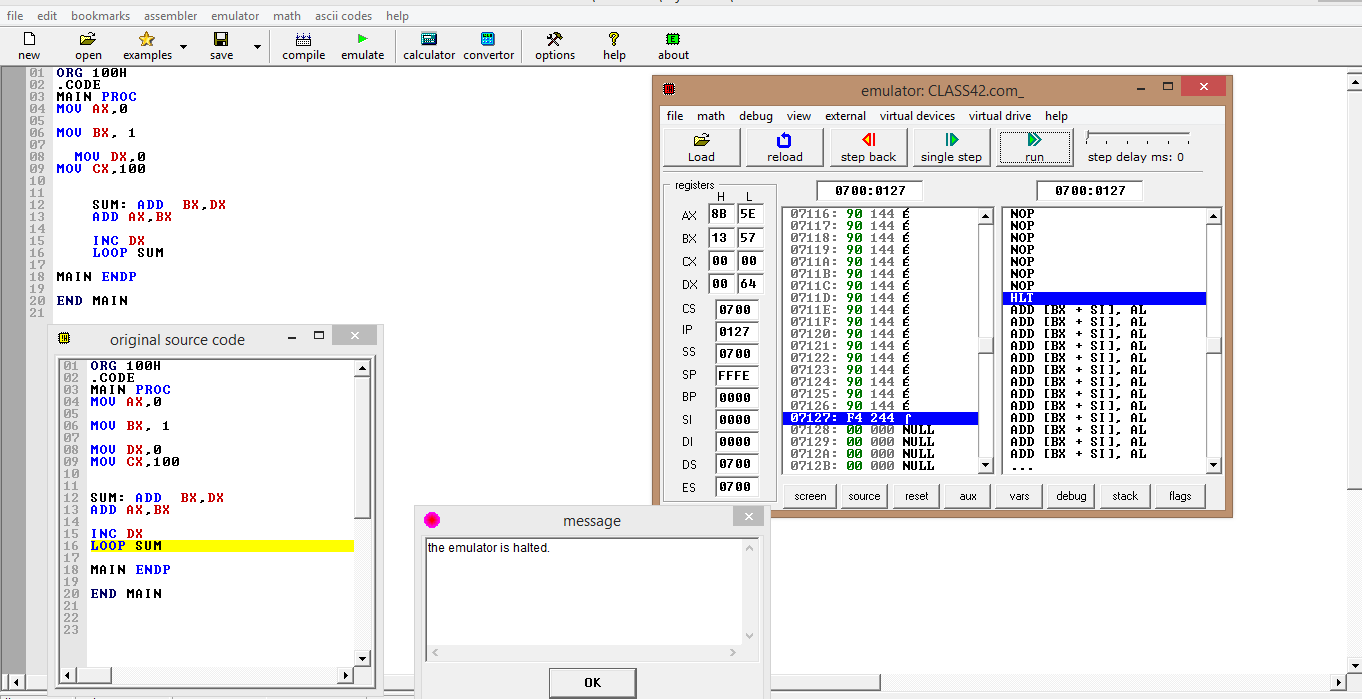
The sum of the numbers was registered on AX =37h

**Home Assignment:**

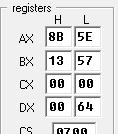
1. **Write an assembly code to find the summation of 1st 100 term of the following series:**

**1+2+4+7+11+……………………**

.



**Results:**



The sum of the numbers was registered on AX =8B5Eh

**3**

**Class Assignment:**

**(a) Program 1:**

CODE SEGMENT

MOV AL, 45h

AND AL, 7Fh; clear the sign bit of AL while leaving other bits unchanged

OR AL, 81h; set the msb and lsb while preserving the other bits.

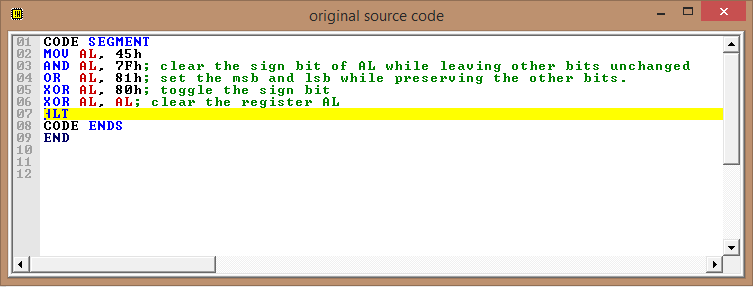
XOR AL, 80h; toggle the sign bit

XOR AL, AL; clear the register AL

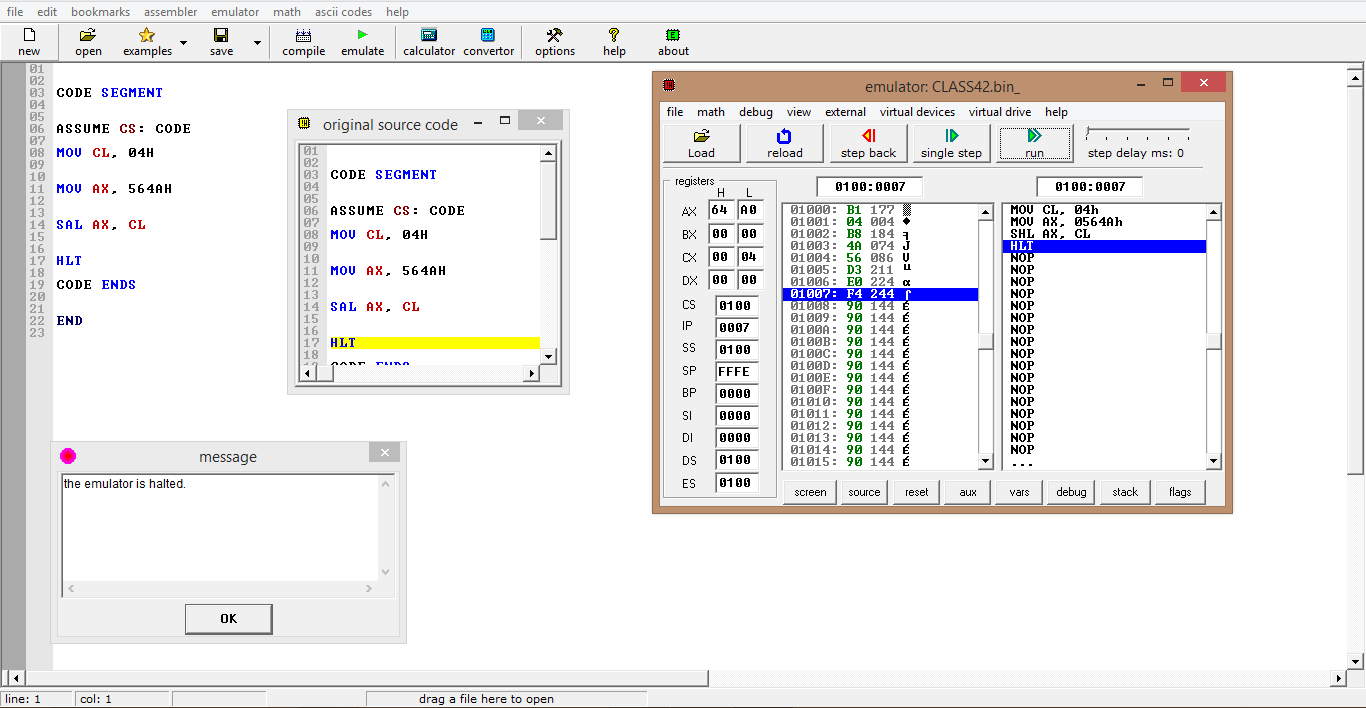
HLT

CODE ENDS

END



**Result:** The program executed various logical operations.



**(b) Program 2:**

CODE SEGMENT

ASSUME CS: CODE

MOV CL, 04H

MOV AX, 564AH

SAL AX, CL

HLT

CODE ENDS

END

**Result:**

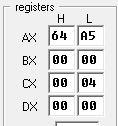
Value of AX before run= 564A= 0101 0110 0100 1010

Left shift by 4=

Value of AX before run= 64A0= 0110 0100 1010 0000

So, The program Shifted the value by 4 bit

So, The program Shifted the value by 4 bit

**(c) Program 3: Result:**

CODE SEGMENT

ASSUME CS: CODE

MOV CL, 04H

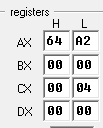
MOV AX, 564AH

ROL AX, CL

HLT

CODE ENDS

END

 **(d) Program 4: Result:**

CODE SEGMENT

ASSUME CS: CODE

MOV CL, 04H

MOV AX, 564AH

RCL AX, CL

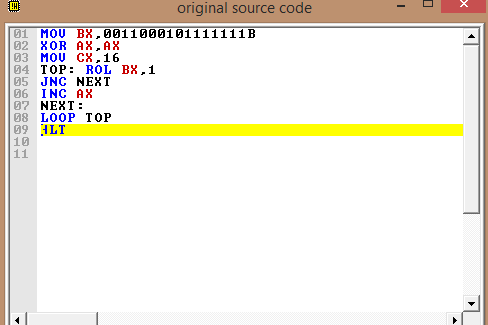
HLT

CODE ENDS

END

**Home Assignment:**

1. **Write some code to count the number of 1 bit in BX, without changing BX. Put the answer in AX.**

**Code:**

MOV BX,0011000101111111B

XOR AX,AX

MOV CX,16

TOP: ROL BX,1

JNC NEXT

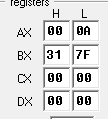
INC AX

NEXT:

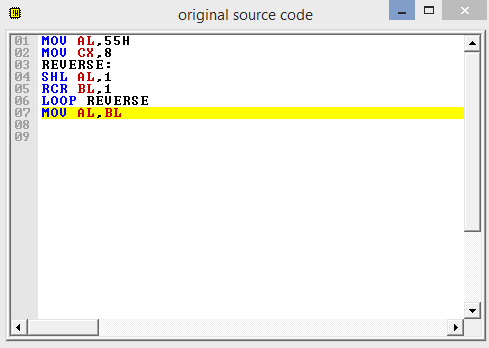
LOOP TOP

HLT

**Result:**

 Number of 1, AX=Ah= 10

BX=317F= 0011 0001 0111 1111= Unchanged

1. **Write some code to reverse the bit pattern of the number stored in AX.**

**Code:**

MOV BX,0011000101111111B

XOR AX,AX

MOV CX,16

TOP: ROL BX,1

JNC NEXT

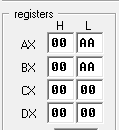
INC AX

NEXT:

LOOP TOP

HLT

**Result:**

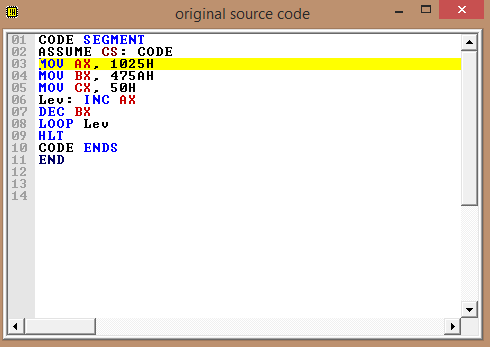


Before execution AX=55h = 01010101

After execution AX=AAh= 10101010

So, the bit pattern was reversed

4

**Class Assignment:**

**(a) Program 1:**

CODE SEGMENT

ASSUME CS: CODE

MOV AX, 1025H

MOV BX, 475AH

MOV CX, 50H

Lev: INC AX

DEC BX

LOOP Lev

HLT

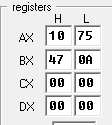
CODE ENDS

END

**Observation:**

The loop kept executing until CX=0

**Results:**

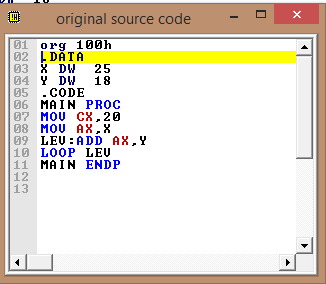


50h times increment of AX =1075h

50h times decrement of BX =470Ah

**Home Assignment:**

1. **Suppose x = 25 and y = 18. Add y with x for 20 times**

**Code:**

org 100h

.DATA

X DW 25

Y DW 18

.CODE

MAIN PROC

MOV CX,20

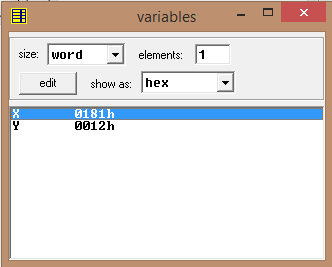
MOV AX,X

LEV:ADD AX,Y

LOOP LEV

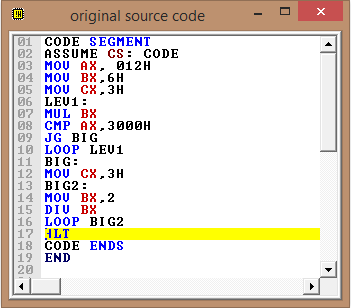
MAIN ENDP

**Result:**



X+20Y= 181H

1. **Multiply 12 by 6 until result is below 3000H. If result is greater than this, divide the result by 2 for 3 times.** **Find Least Common Multiplier of 12H and 25H.**

**Code:**

CODE SEGMENT

ASSUME CS: CODE

MOV AX, 012H

MOV BX,6H

MOV CX,3H

LEV1:

MUL BX

CMP AX,3000H

JG BIG

LOOP LEV1

BIG:

MOV CX,3H

BIG2:

MOV BX,2

DIV BX

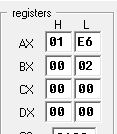
LOOP BIG2

HLT

CODE ENDS

END

**Result:**



1. **Find Least Common Multiplier of 12H and 25H.**

**Code:**

org 100H

.DATA

A DW 012H

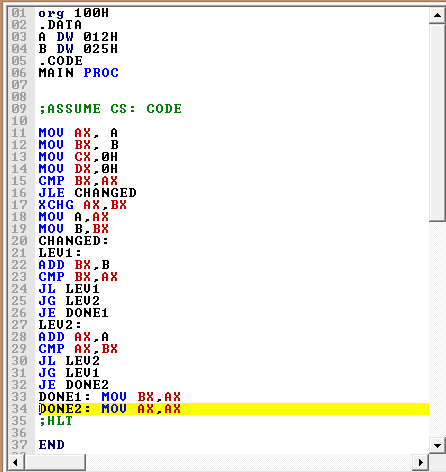
B DW 025H

.CODE

MAIN PROC

;ASSUME CS: CODE

MOV AX, A

MOV BX, B

MOV CX,0H

MOV DX,0H

CMP BX,AX

JLE CHANGED

XCHG AX,BX

MOV A,AX

MOV B,BX

CHANGED:

LEV1:

ADD BX,B

CMP BX,AX

JL LEV1

JG LEV2

JE DONE1

LEV2:

ADD AX,A

CMP AX,BX

JL LEV2

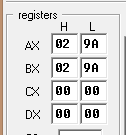
JG LEV1

JE DONE2

DONE1: MOV BX,AX

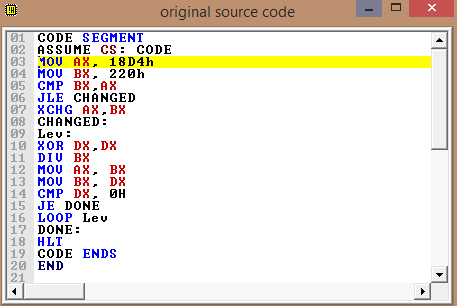
DONE2: MOV AX,AX

END

**Result:**

LCM of 12H and 25H is 29AH

**4. Find GCD of 18D4H and 220H.**

**Code:**

CODE SEGMENT

ASSUME CS: CODE

MOV AX, 18D4h

MOV BX, 220h

CMP BX,AX

JLE CHANGED

XCHG AX,BX

CHANGED:

Lev:

XOR DX,DX

DIV BX

MOV AX, BX

MOV BX, DX

CMP DX, 0H

JE DONE

LOOP Lev

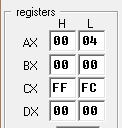
DONE:

HLT

CODE ENDS

END

**Result:**



GCD of 18D4H and 220H is 4H

**5**

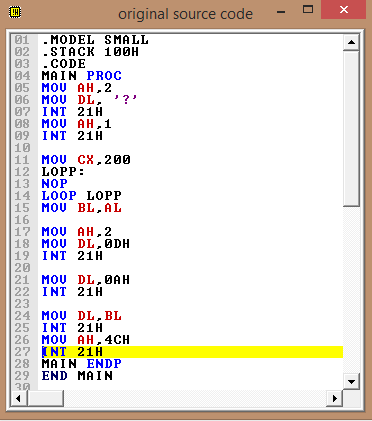
**Class Assignment:**

1. **Program 1: Read a character from the keyboard and then display it after 2 seconds.**

**CODE:**

.MODEL SMALL

.STACK 100H

.CODE

MAIN PROC

MOV AH,2

MOV DL, '?'

INT 21H

MOV AH,1

INT 21H

MOV CX,200

LOPP:

NOP

LOOP LOPP

MOV BL,AL

MOV AH,2

MOV DL,0DH

INT 21H

MOV DL,0AH

INT 21H

MOV DL,BL

INT 21H

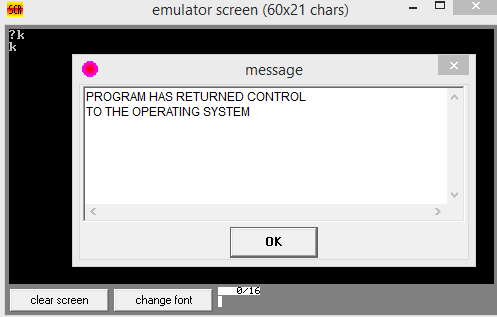
MOV AH,4CH

INT 21H

MAIN ENDP

END MAIN

**Results:**

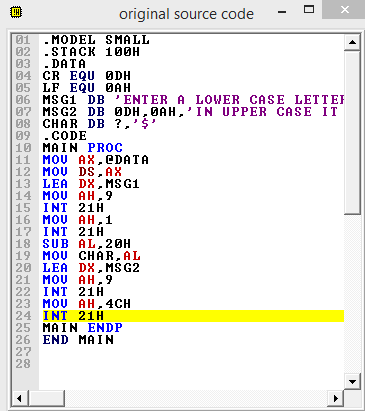


**b) Program 2: Read a lowercase character from the keyboard and then display it in uppercase.**

**CODE:**

.MODEL SMALL

.STACK 100H

.DATA

CR EQU 0DH

LF EQU 0AH

MSG1 DB 'ENTER A LOWER CASE LETTER: $'

MSG2 DB 0DH,0AH,'IN UPPER CASE IT IS: '

CHAR DB ?,'$'

.CODE

MAIN PROC

MOV AX,@DATA

MOV DS,AX

LEA DX,MSG1

MOV AH,9

INT 21H

MOV AH,1

INT 21H

SUB AL,20H

MOV CHAR,AL

LEA DX,MSG2

MOV AH,9

INT 21H

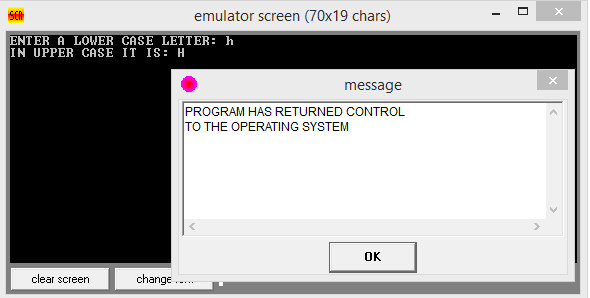
MOV AH,4CH

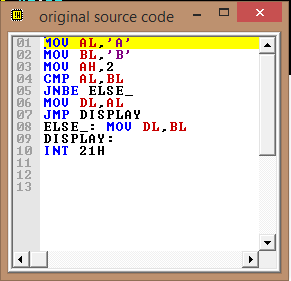
INT 21H

MAIN ENDP

END MAIN

**Results:**



**Home Assignment:**

1. **AL and BL contains extended ASCII characters, display the one that comes first in the character sequence.**

**CODE:**

MOV AL,'A'

MOV BL,'B'

MOV AH,2

CMP AL,BL

JNBE ELSE\_

MOV DL,AL

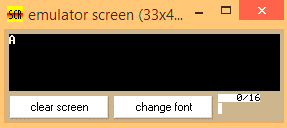
JMP DISPLAY

ELSE\_: MOV DL,BL

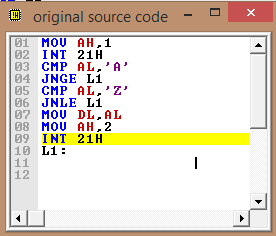
DISPLAY:

INT 21H

**Results:**



1. **Read a character, and if it is an uppercase letter, display it.**



**Code:**

MOV AH,1

INT 21H

CMP AL,'A'

JNGE L1

CMP AL,'Z'

JNLE L1

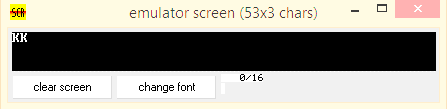
MOV DL,AL

MOV AH,2

INT 21H

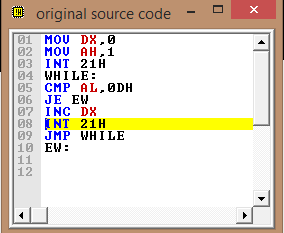
L1:

**Result:**



1. **Write some code to count the number of characters in a line.**

**Code:**



MOV DX,0

MOV AH,1

INT 21H

WHILE:

CMP AL,0DH

JE EW

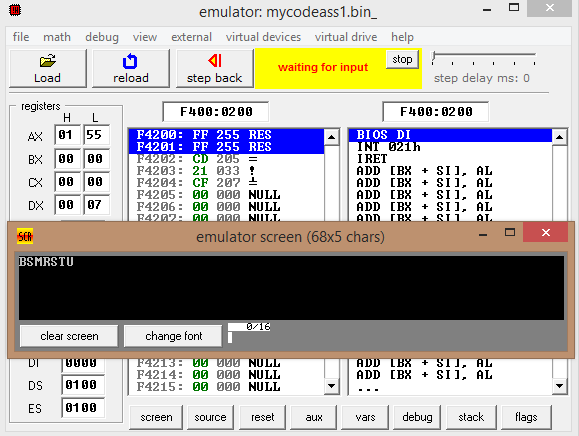
INC DX

INT 21H

JMP WHILE

EW:

**Result:**



Number of characters in the line is = DX=7