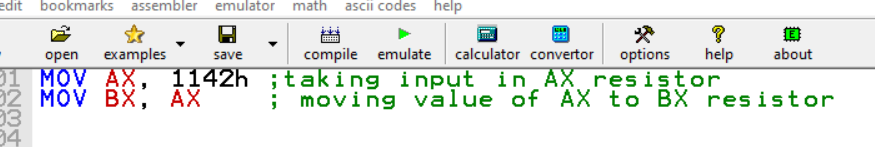
**Task 1:**

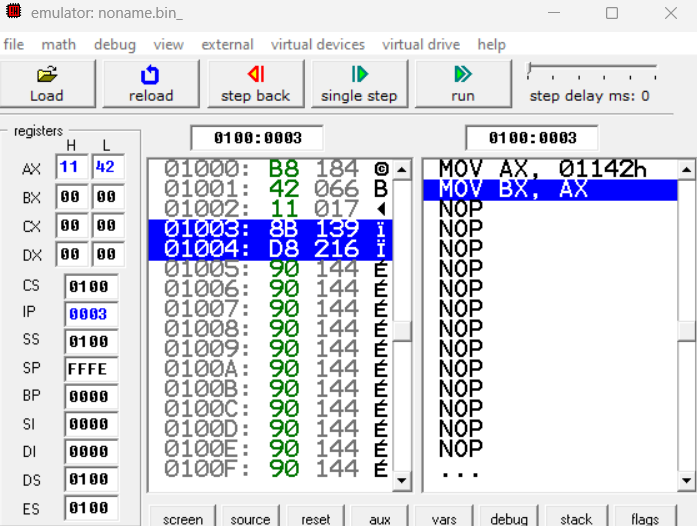
Code:

MOV AX, 1142h ;taking input in AX resistor

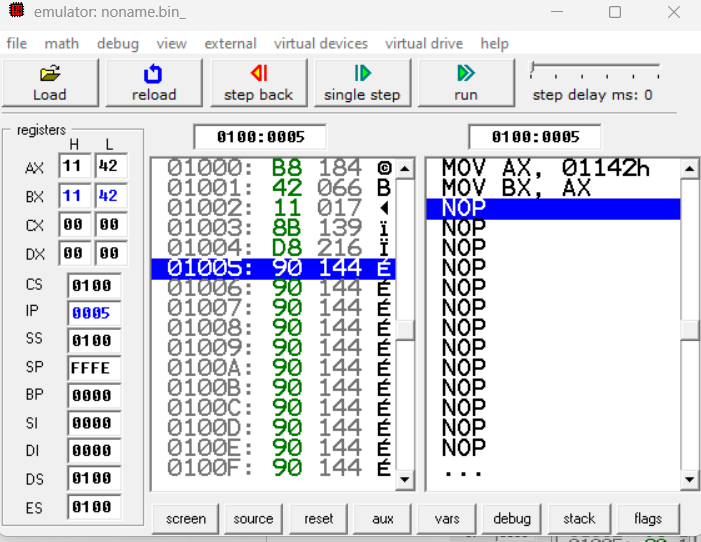
MOV BX, AX ; moving value of AX to BX resistor



Output:

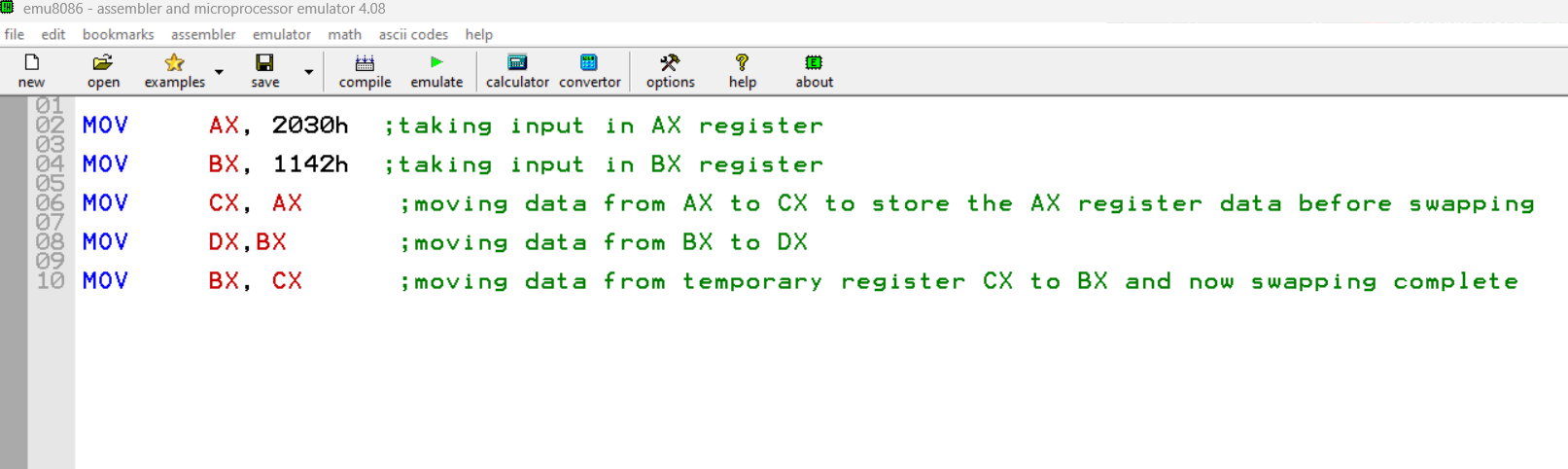


Moving input from AX to BX =>

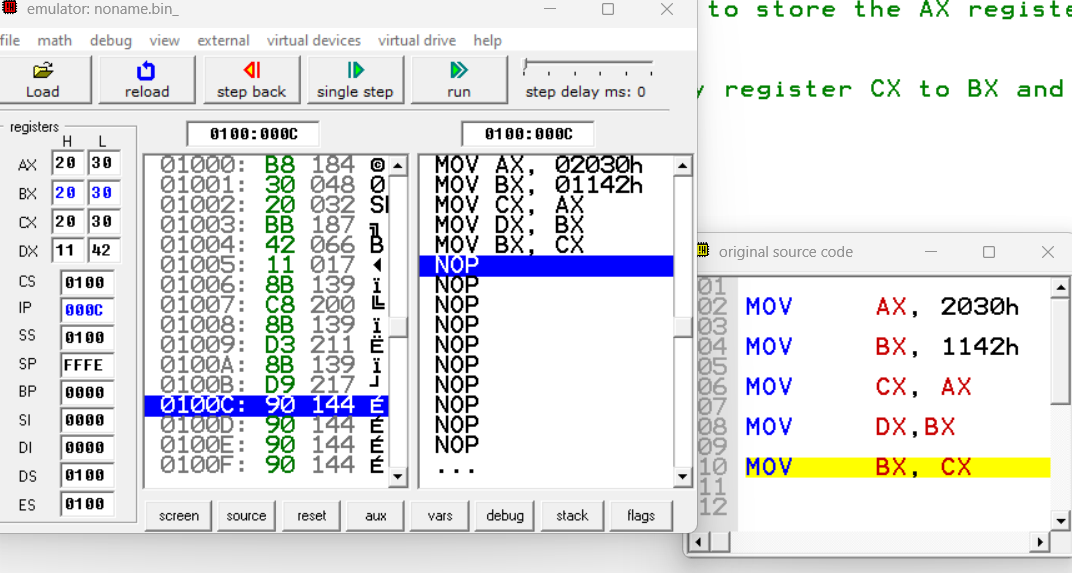


**Task 2:**

code:



output:



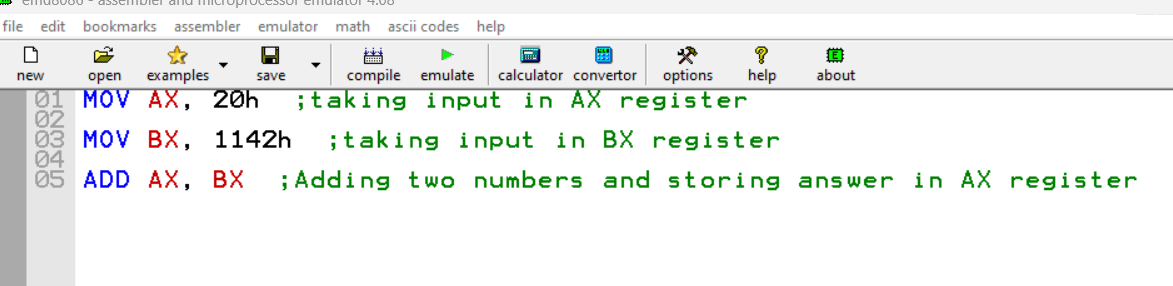
**Task 3:**

Code:

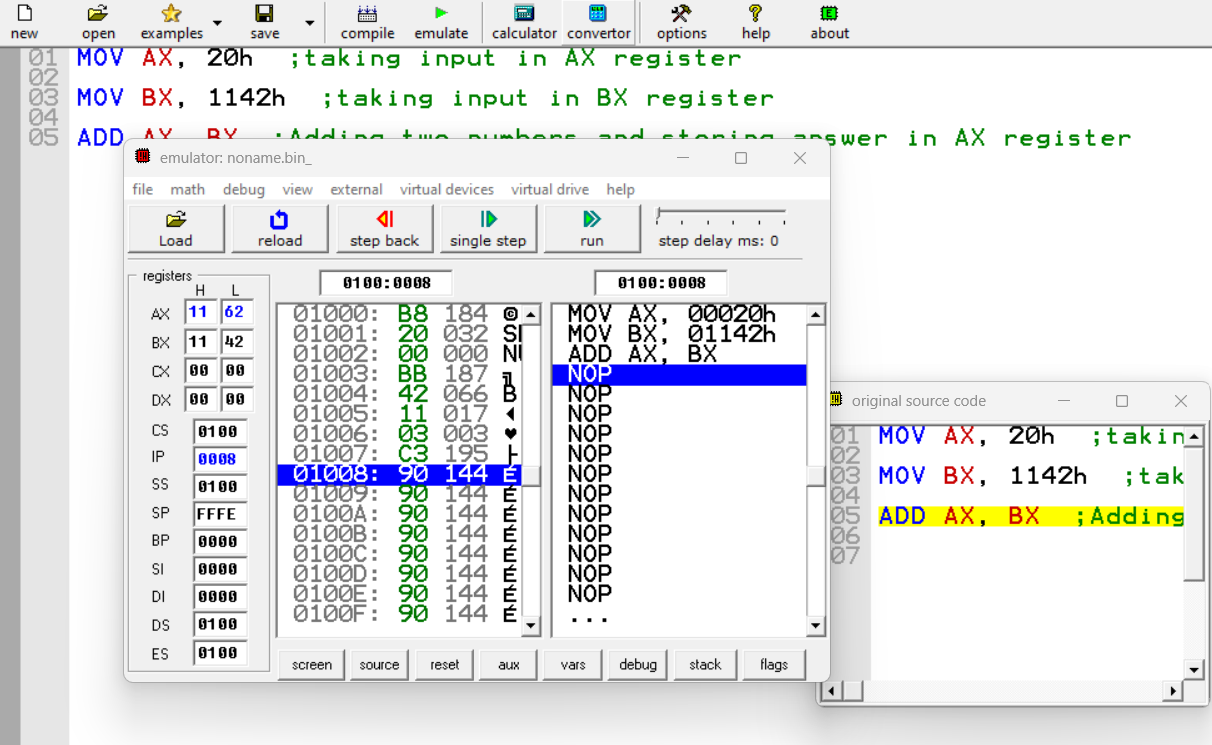
MOV AX, 20h ;taking input in AX register

MOV BX, 1142h ;taking input in BX register

ADD AX, BX ;Adding two numbers and storing answer in AX register



output:



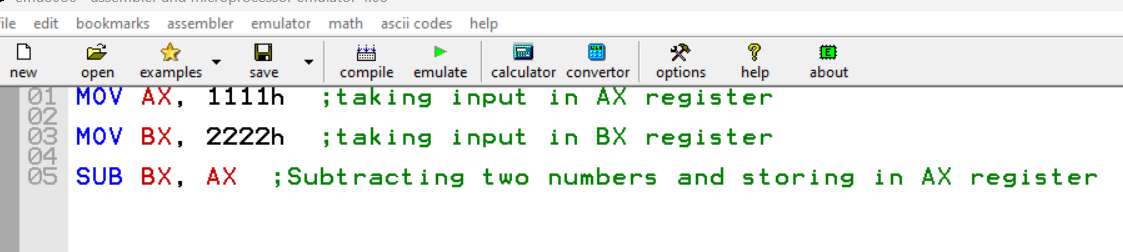
**Task 4:**

Code:

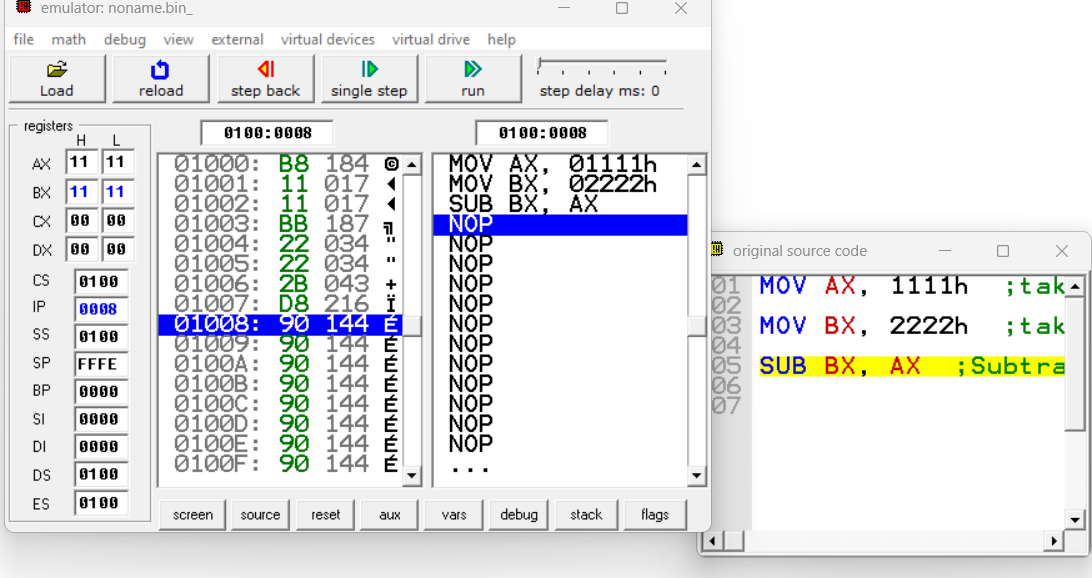
MOV AX, 1111h ;taking input in AX register

MOV BX, 2222h ;taking input in BX register

SUB BX, AX ;Subtracting two numbers and storing in AX register



output:



**Q.A. :**  when we subtract larger number from the smaller one in 8086 microprocessor, an overflow will happen. Because when the result of a subtraction operation in the 8086 microprocessor is too large to be represented in the given number of bits, the overflow flag (OF) is set to indicate that overflow has occurred.

**Task 5:**

Code:

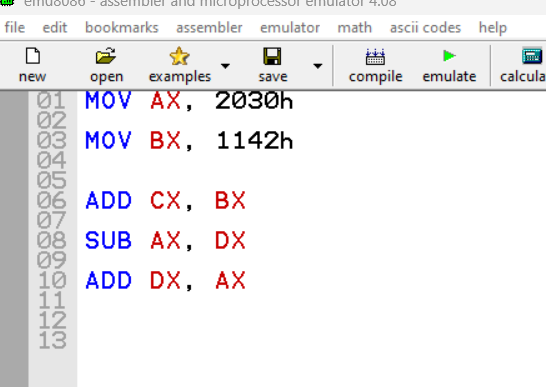
MOV AX, 2030h

MOV BX, 1142h

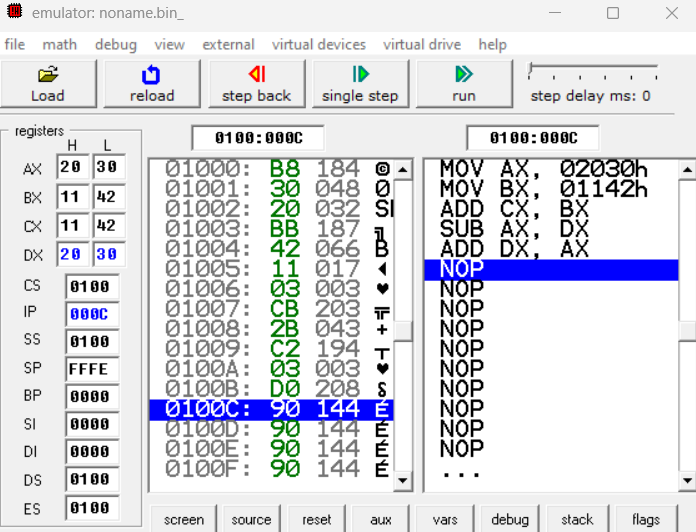
ADD CX, BX

SUB AX, DX

ADD DX, AX



Output:



**Task 6 (1):**

Code: .MODEL SMALL

.STACK 100H

.DATA

; DEFINE YOUR VARIABLES

A DW 1000h

B DW 2000h

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

;1 no. A = B - A

MOV AX, A ; Move value of A into AX

MOV BX, B ; Move value of B into BX

SUB BX, AX ; Subtract value of AX from BX

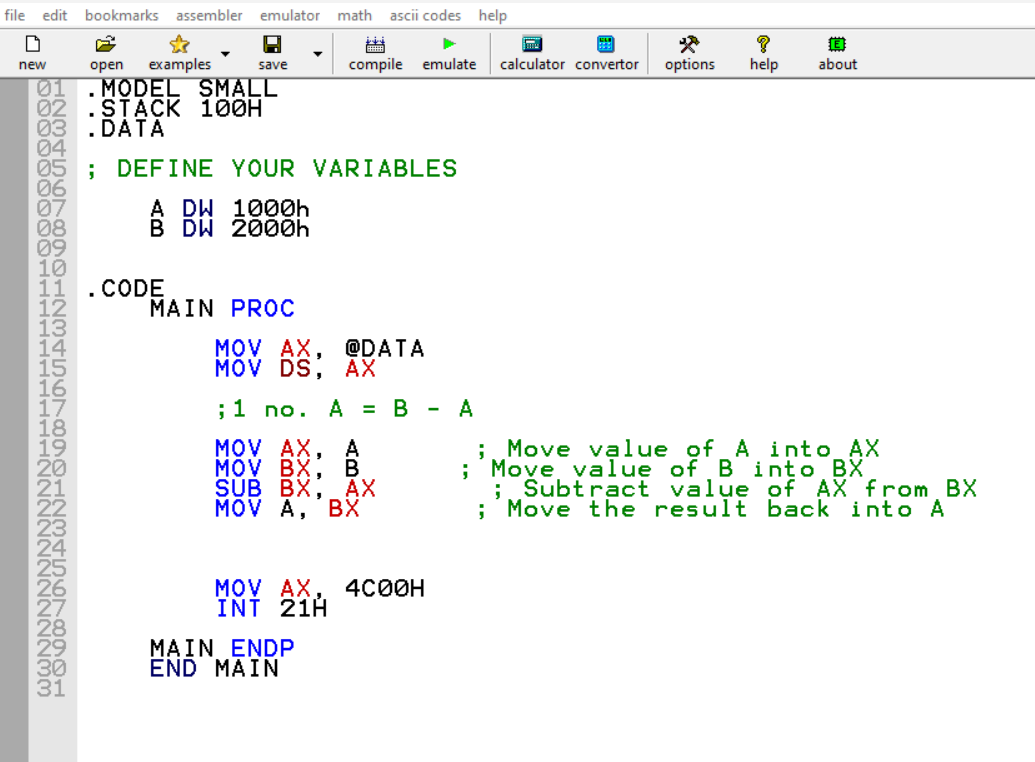
MOV A, BX ; Move the result back into A

MOV AX, 4C00H

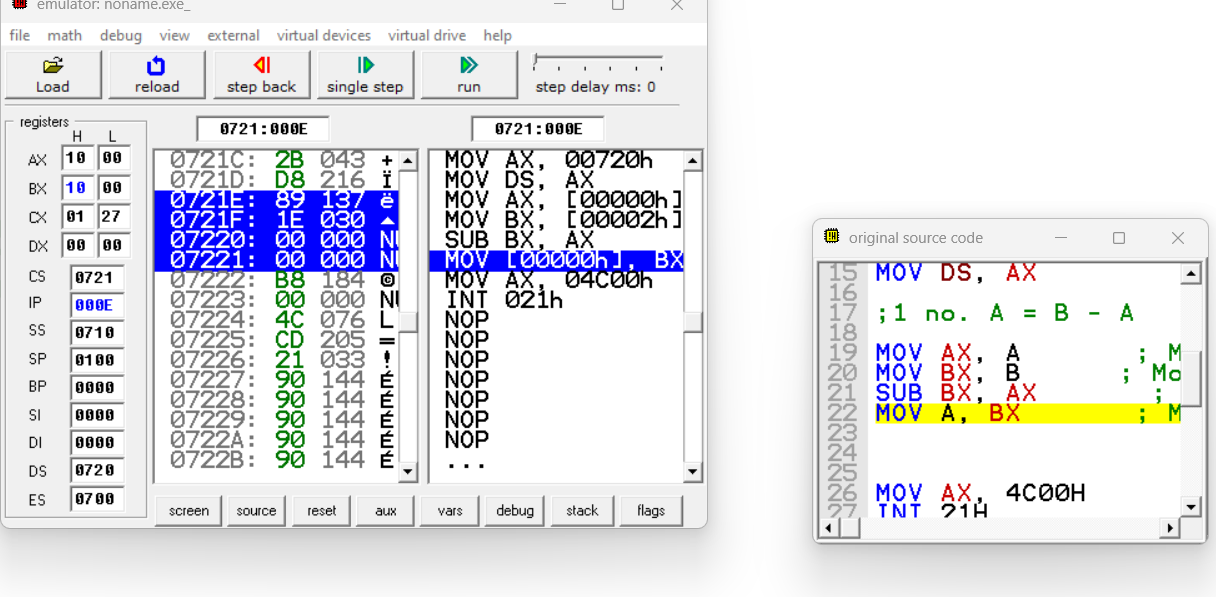
INT 21H

MAIN ENDP

END MAIN



Output:



**Task 6 (2):**

Code:

.MODEL SMALL

.STACK 100H

.DATA

; DEFINE YOUR VARIABLES

A DW 1000h

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

; 2. A = -(A + 1)

MOV AX, A ; Move value of A into AX

ADD AX, 1 ; Add 1 to AX

NEG AX ; Negate the value of AX

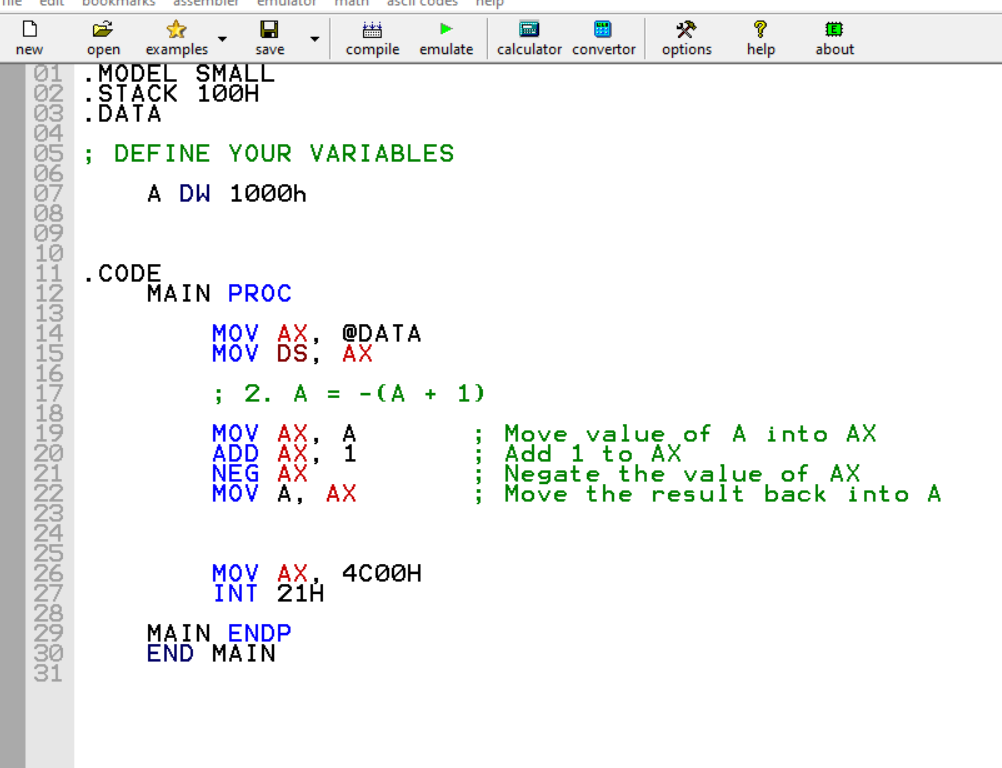
MOV A, AX ; Move the result back into A

MOV AX, 4C00H

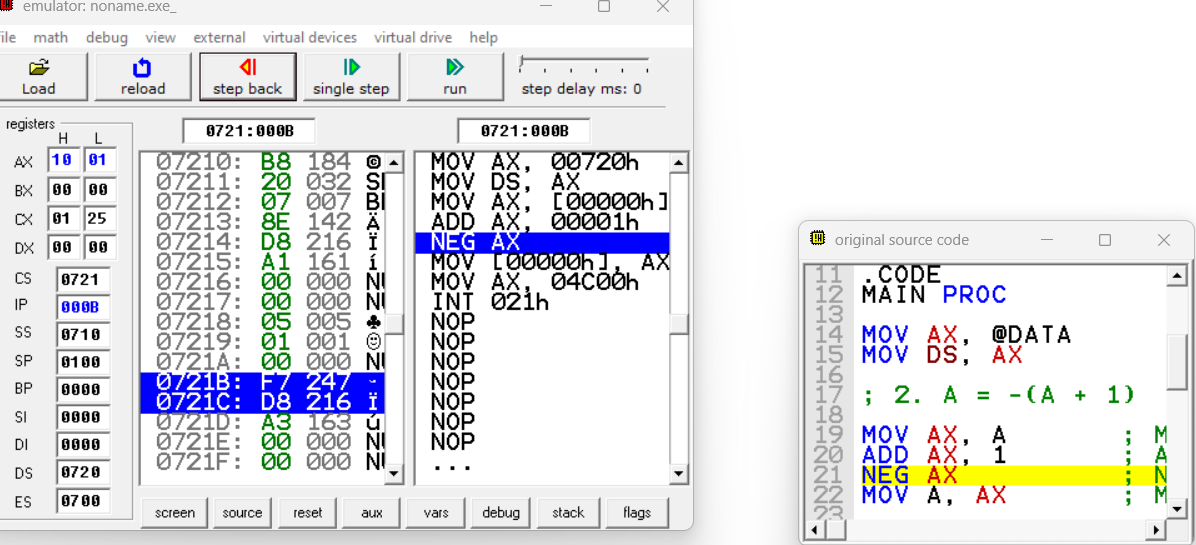
INT 21H

MAIN ENDP

END MAIN



Output:



**Task 6 (3):**

**Code:**

.MODEL SMALL

.STACK 100H

.DATA

; DEFINE YOUR VARIABLES

A DW 1000h

B DW 2000h

C DW ?

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

; 3. C = A + (B + 1); use inc

MOV AX, B ; Move value of B into AX

;ADD AX, 1 ; Add 1 to AX

INC AX ; Increment BX by 1 (same as adding 1)

MOV BX, A ; Move value of A into BX

ADD BX, AX ; Add value of AX to BX

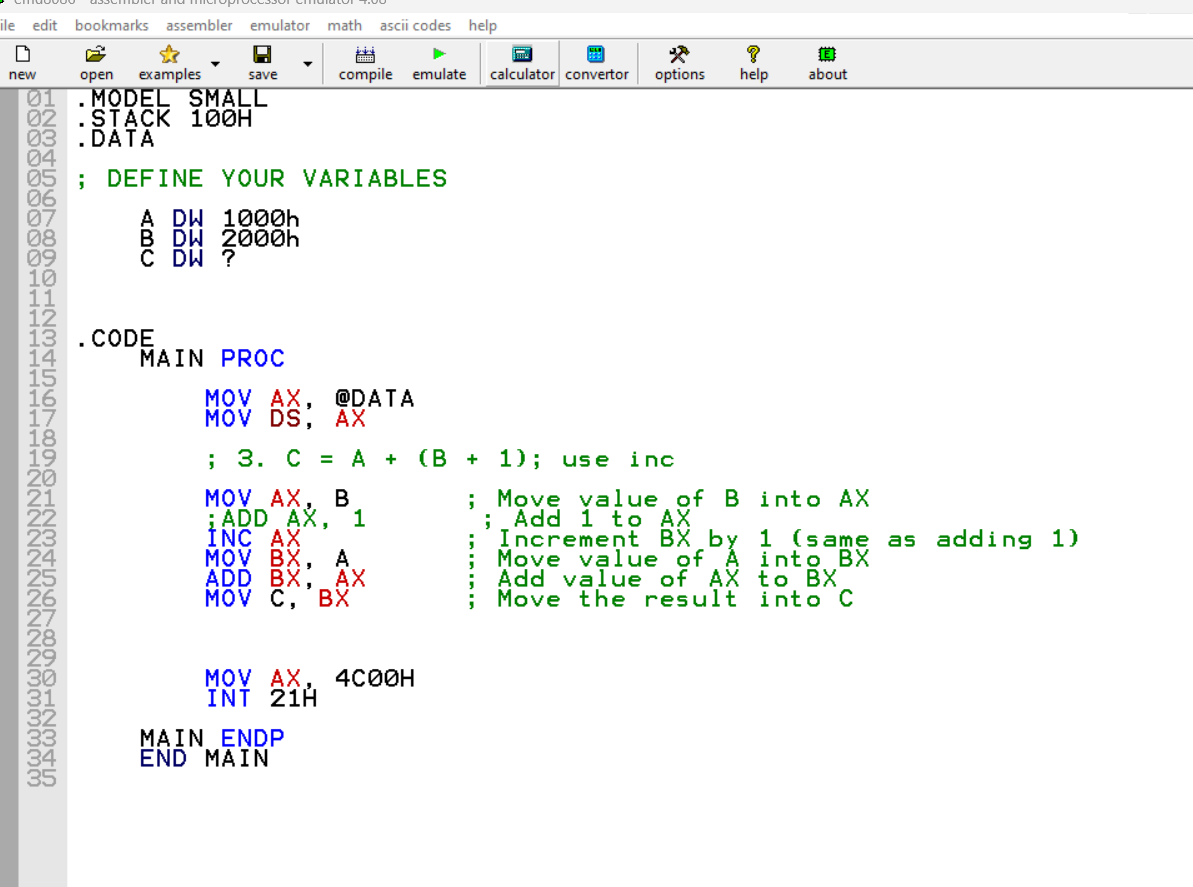
MOV C, BX ; Move the result into C

MOV AX, 4C00H

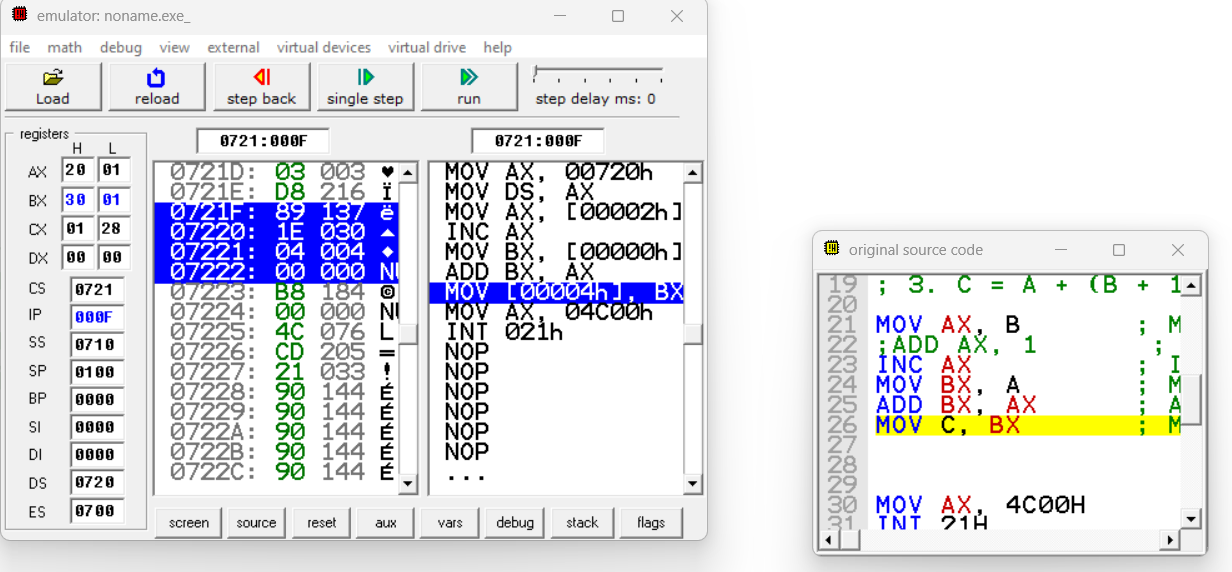
INT 21H

MAIN ENDP

END MAIN



Output:

****

**Task 7(1):**

Code:

.MODEL SMALL

.STACK 100H

.DATA

; DEFINE YOUR VARIABLES

X DW 3350h

Y DW 2456h

Z DW ?

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

; 1. X \* Y

MOV AX, X ; Move value of X into AX

MOV BX, Y ; Move value of Y into BX

MUL BX ; Multiply AX by BX (result in AX)

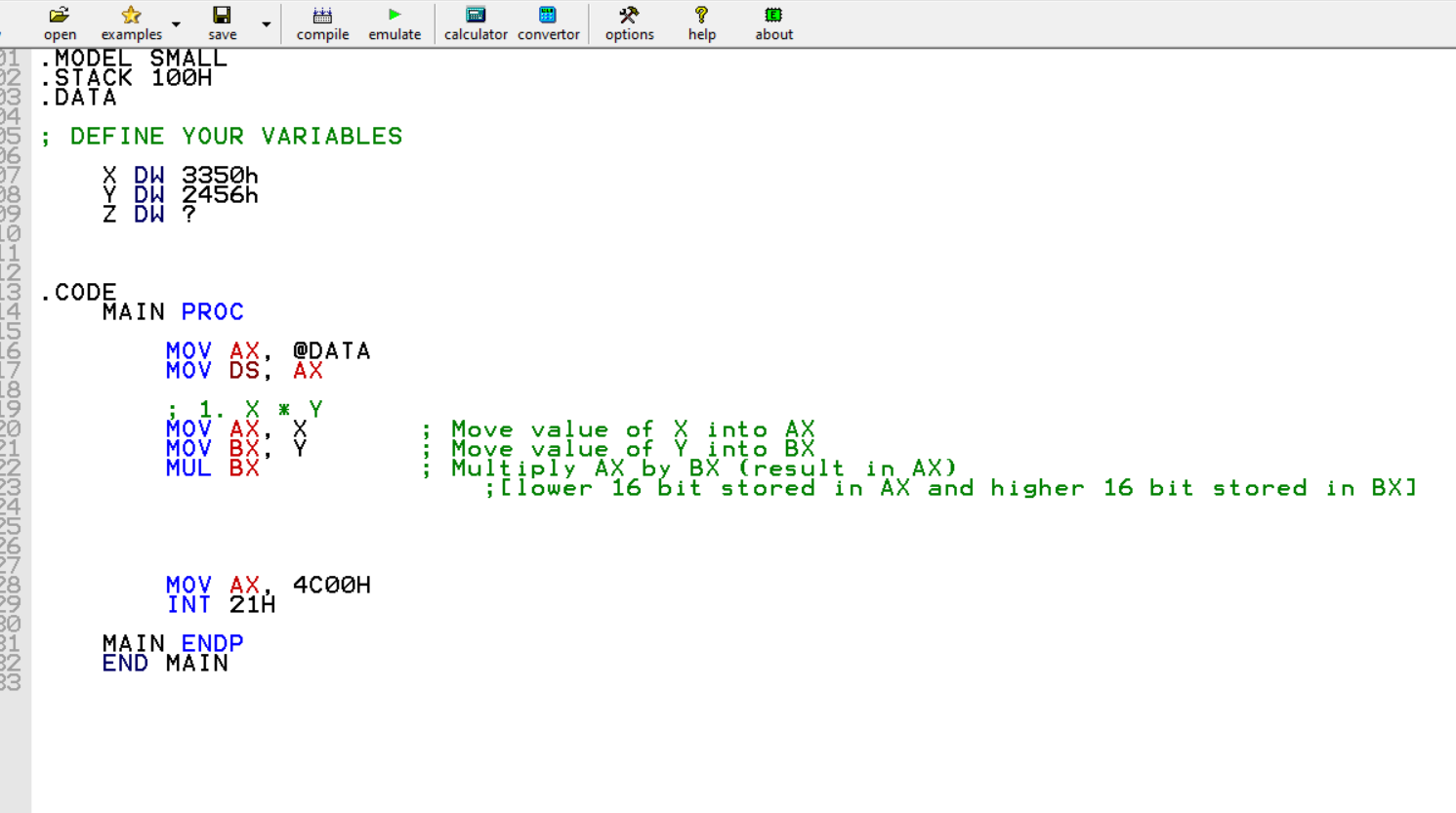
;[lower 16 bit stored in AX and higher 16 bit stored in BX]

MOV AX, 4C00H

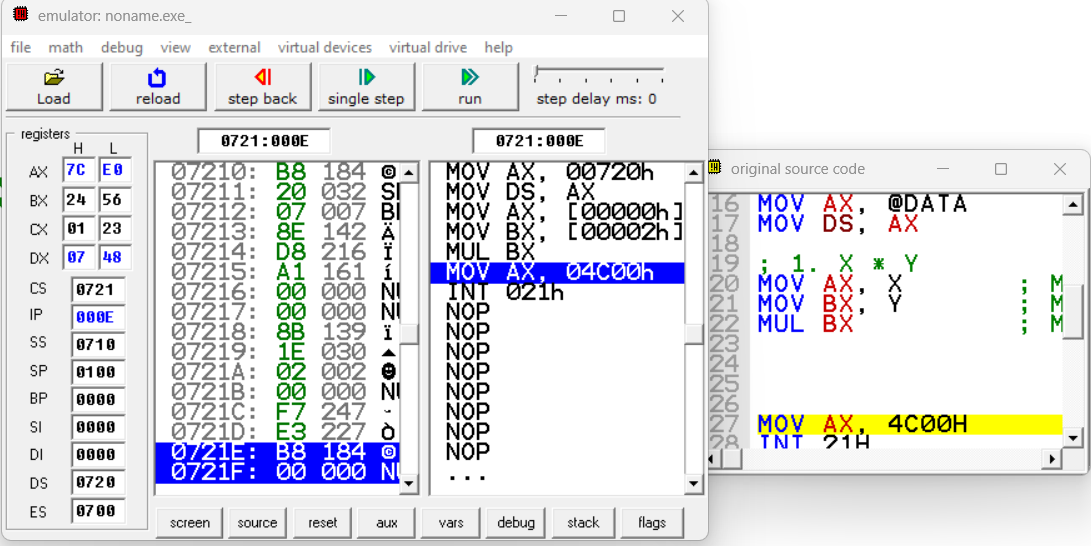
INT 21H

MAIN ENDP

END MAIN



Output:



**Task 7(2):**

Code:

.MODEL SMALL

.STACK 100H

.DATA

; DEFINE YOUR VARIABLES

X DW 3350h

Y DW 2456h

Z DW ?

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

; 2. X / Y

MOV AX, X ; Move value of X into AX

MOV BX, Y ; Move value of Y into BX

XOR DX, DX ; Clear DX to prepare for division

DIV BX ; Divide AX by BX (quotient in AX, remainder in DX)

MOV AX, 4C00H

INT 21H

MAIN ENDP

END MAIN

Output:

