MICROPROCESSOR AND INTERFACING

LAB MANUAL: 6

Execute the following tasks

Task 1:

What will be the hexadecimal values of DX, AX, and the Carry flag after the following instructions execute?

mov ax,1234h

mov bx,100h

mul bx

SOURCE CODE:

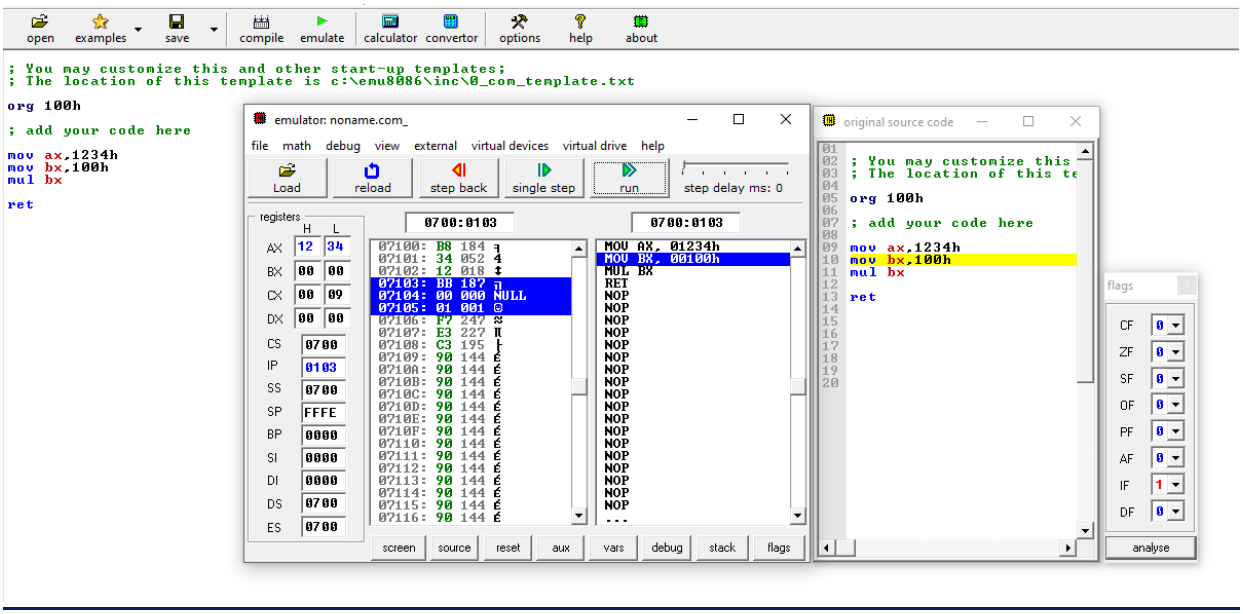
```
mov ax,1234h
mov bx,100h
mul bx
```

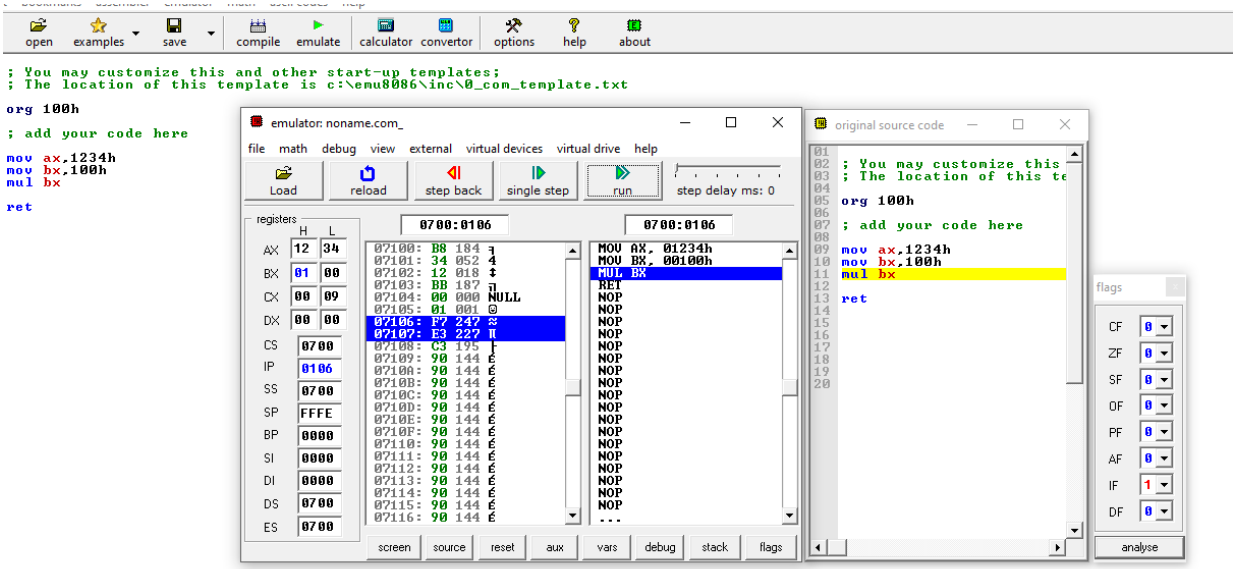
EXPALANATION:

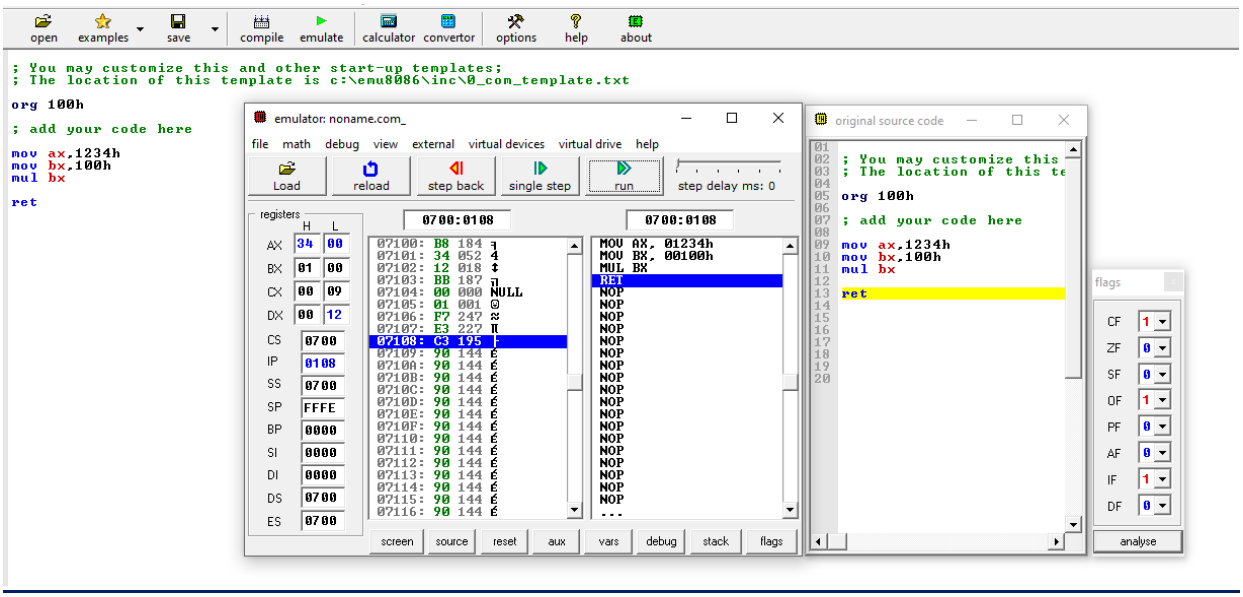
1234h will be copied to the accumulator register.

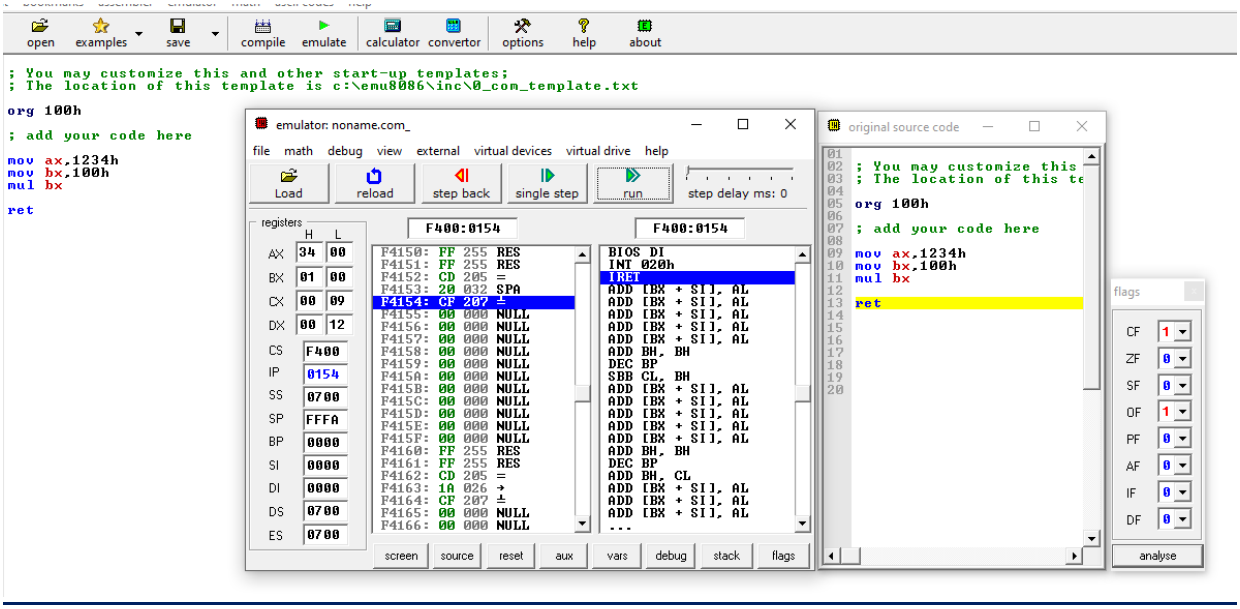
100h will be copied to the accumulator register and previous content will be removed.

Content of AX will be multiplied with the content of BX which has nothing in it or zero and result will be stored in AX. The result is 0 because anything multiplied by 0 is 0.


open
examples
save
compile
emulate
calculator
convertor
options
help
about
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt
org 100h
; add your code here
mov ax,1234h
mov bx,100h
mul bx
ret
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
0700:0103
0700:0103
MOV AX, 01234h
MOV BX, 00100h
MUL BX
RET
screen
source
reset
aux
vars
debug
stack
flags
original source code
analyse
CF
ZF
SF
OF
PF
AF
IF
DF


open
examples
save
compile
emulate
calculator
convertor
options
help
about
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt
org 100h
; add your code here
mov ax,1234h
mov bx,100h
mul bx
ret
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
0700:0106
0700:0106
MOV AX, 01234h
MOV BX, 00100h
MUL BX
RET
screen
source
reset
aux
vars
debug
stack
flags
original source code
analyse
CF
ZF
SF
OF
PF
AF
IF
DF





Task 2:

What will be the hexadecimal values of DX and AX after the following instructions execute?

mov dx,0087h

mov ax,6000h

mov bx,100h

div bx

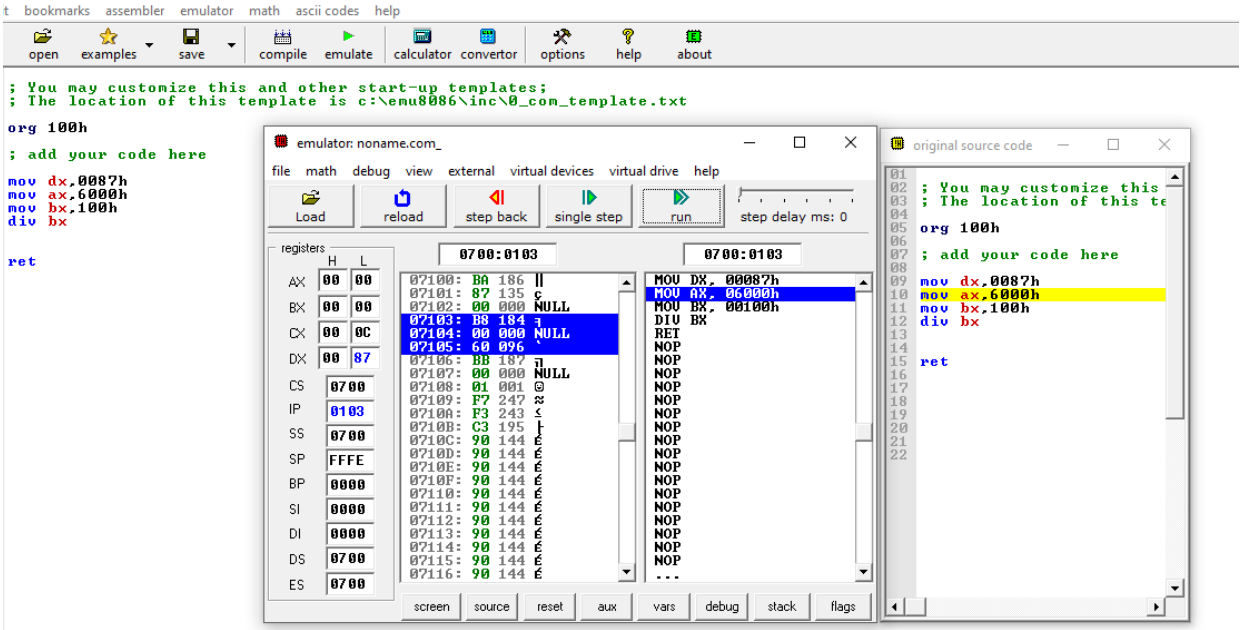
SOURCE CODE:

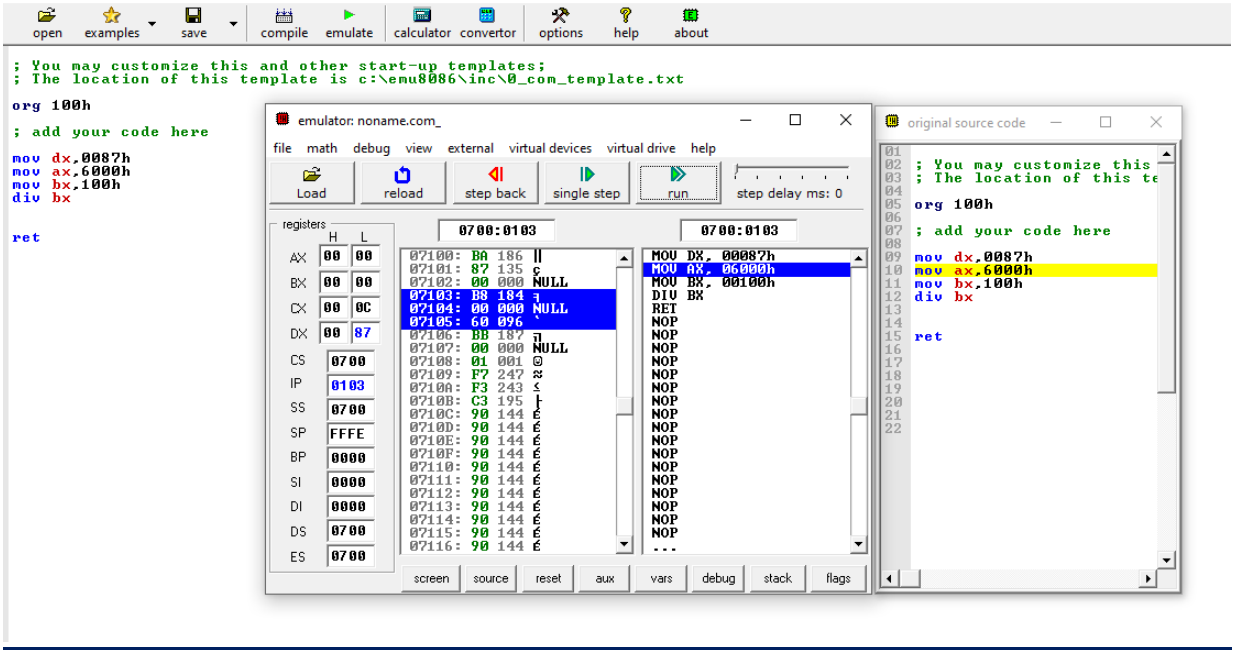
```
mov dx,0087h
mov ax,6000h
mov bx,100h
div bx
```

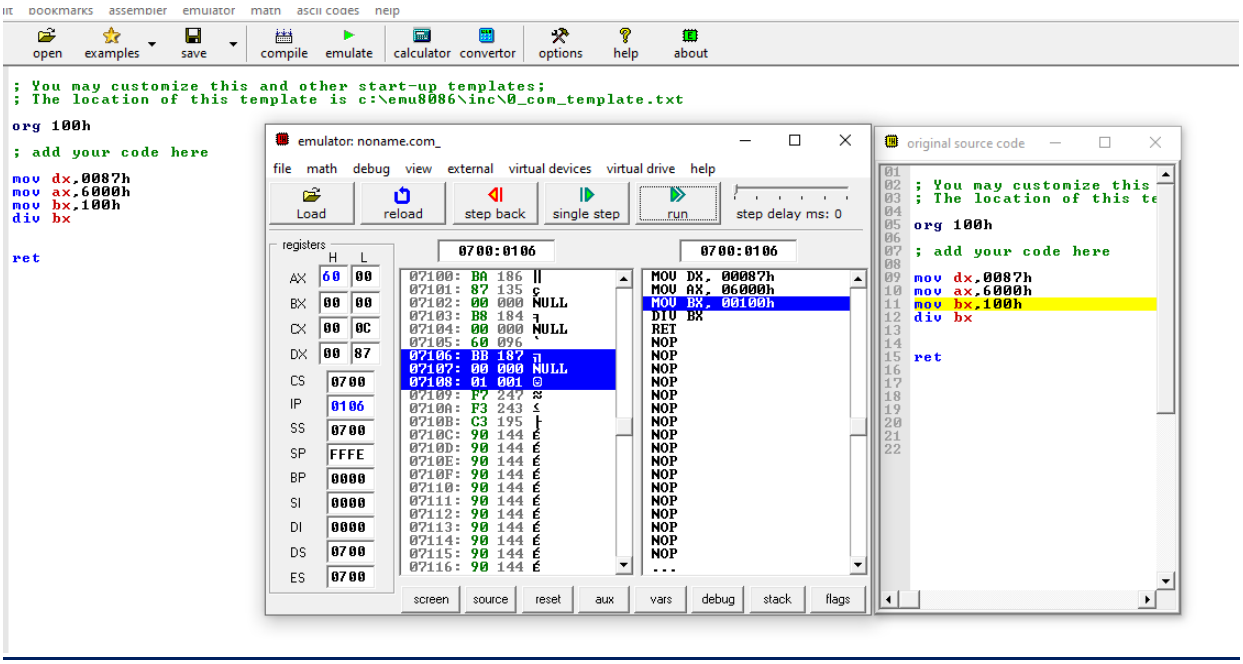
Explanation:

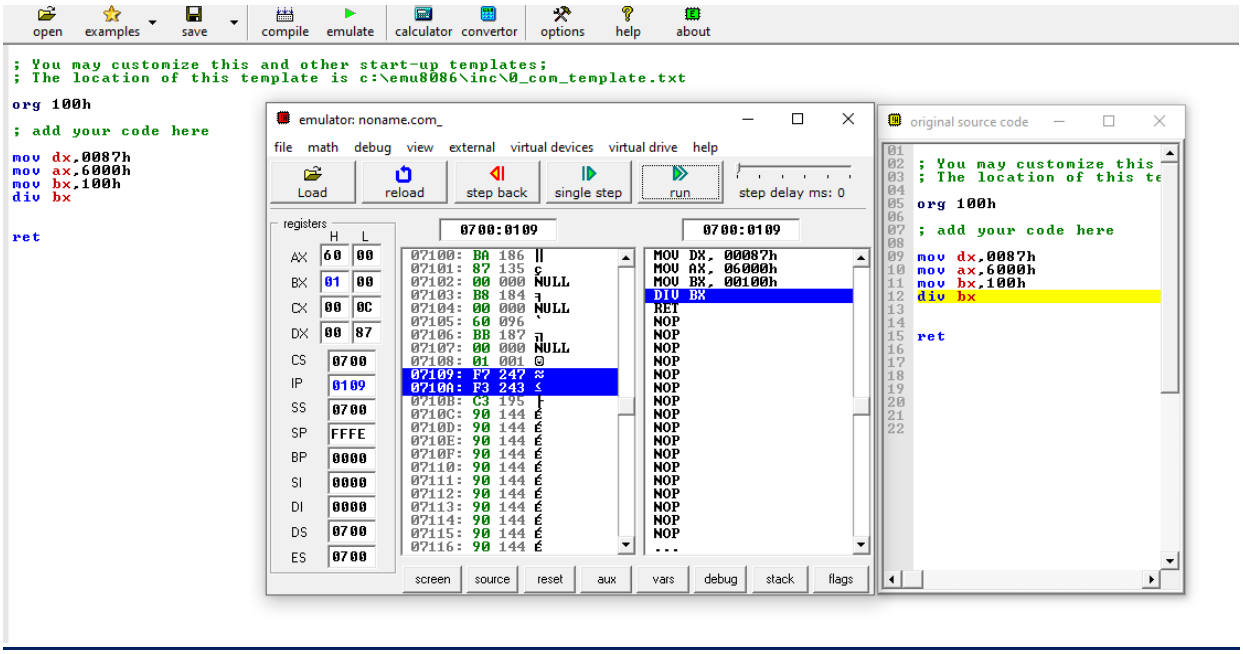
0087h will be copied to DX, 6000h will be copied to AX and 100h to BX register. Content of AX is dividend and content of BX is divisor.

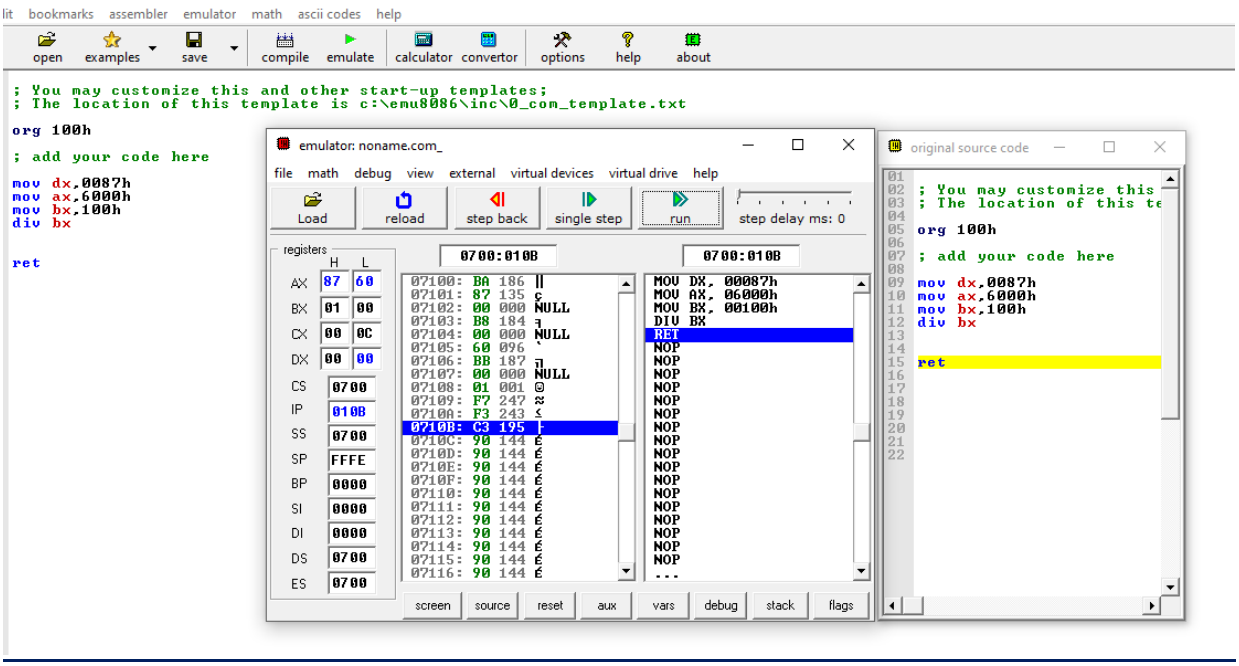
The result after division: Remainder will be stored in DX and Quotient in AX register




open
examples
save
compile
emulate
calculator
convertor
options
help
about
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt
org 100h
; add your code here
mov dx,0087h
mov ax,6000h
mov bx,100h
div bx
ret
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
0700:0103
0700:0103
AX 00 00
BX 00 00
CX 00 0C
DX 00 87
CS 0700
IP 0103
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
07100: BA 186
07101: 87 135
07102: 00 000 NULL
07103: B8 184
07104: 00 000 NULL
07105: 60 096
07106: BB 187
07107: 00 000 NULL
07108: 01 001
07109: F7 247
0710A: F3 243
0710B: C3 195
0710C: 90 144
0710D: 90 144
0710E: 90 144
0710F: 90 144
07110: 90 144
07111: 90 144
07112: 90 144
07113: 90 144
07114: 90 144
07115: 90 144
07116: 90 144
MOV DX, 00087h
MOV AX, 06000h
MOV BX, 00100h
DIV BX
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen
source
reset
aux
vars
debug
stack
flags
original source code
; You may customize this
; The location of this te
org 100h
; add your code here
mov dx,0087h
mov ax,6000h
mov bx,100h
div bx
ret


open
examples
save
compile
emulate
calculator
convertor
options
help
about
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt
org 100h
; add your code here
mov dx,0087h
mov ax,6000h
mov bx,100h
div bx
ret
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
0700:0106
0700:0106
AX 60 00
BX 00 00
CX 00 0C
DX 00 87
CS 0700
IP 0106
SS 0700
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
07100: BA 186
07101: 87 135
07102: 00 000 NULL
07103: B8 184
07104: 00 000 NULL
07105: 60 096
07106: BB 187
07107: 00 000 NULL
07108: 01 001
07109: F7 247
0710A: F3 243
0710B: C3 195
0710C: 90 144
0710D: 90 144
0710E: 90 144
0710F: 90 144
07110: 90 144
07111: 90 144
07112: 90 144
07113: 90 144
07114: 90 144
07115: 90 144
07116: 90 144
MOV DX, 00087h
MOV AX, 06000h
MOV BX, 00100h
DIV BX
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen
source
reset
aux
vars
debug
stack
flags
original source code
; You may customize this
; The location of this te
org 100h
; add your code here
mov dx,0087h
mov ax,6000h
mov bx,100h
div bx
ret





Task 3:

Take two binary numbers such as11111111

11111111

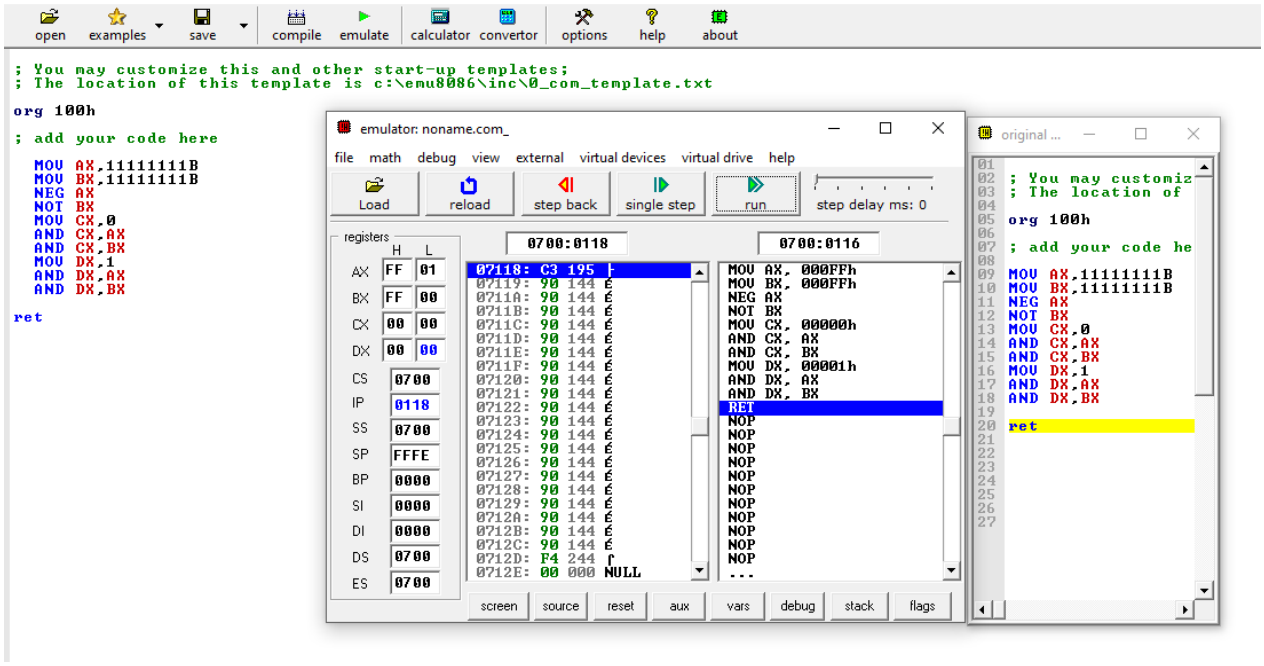
Perform two's complement of the first number and save it to one of the register.

Then, reverse the bits of second binary number and save it to the other register.

1. **Take the value 0 in third register and perform the AND operation with both of the results computed above.**
2. **Take the value 1 in third register and perform the AND operation with both of the results computed above.**

SOURCE CODE:

```
MOV AX,11111111B
MOV BX,11111111B
NEG AX
NOT BX
MOV CX,0
AND CX,AX
AND CX,BX
MOV DX,1
AND DX,AX
AND DX,BX
```

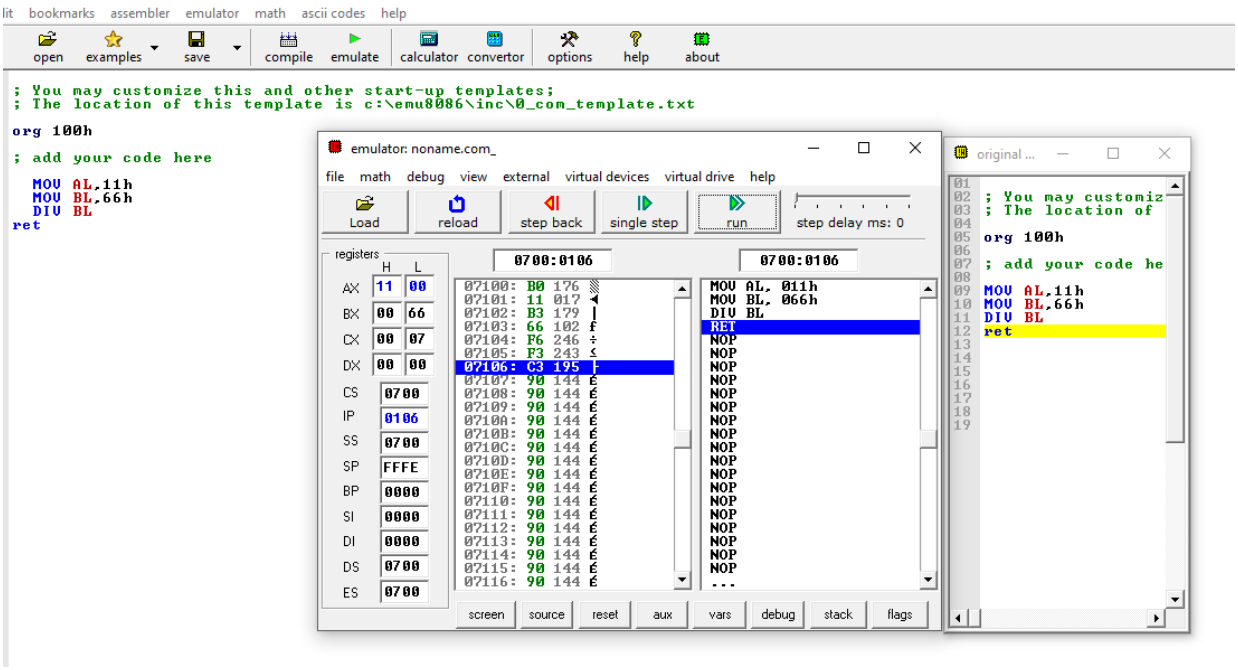


Task 4:

Division of 8 bit numbers using immediate addressing mode?

SOURCE CODE:

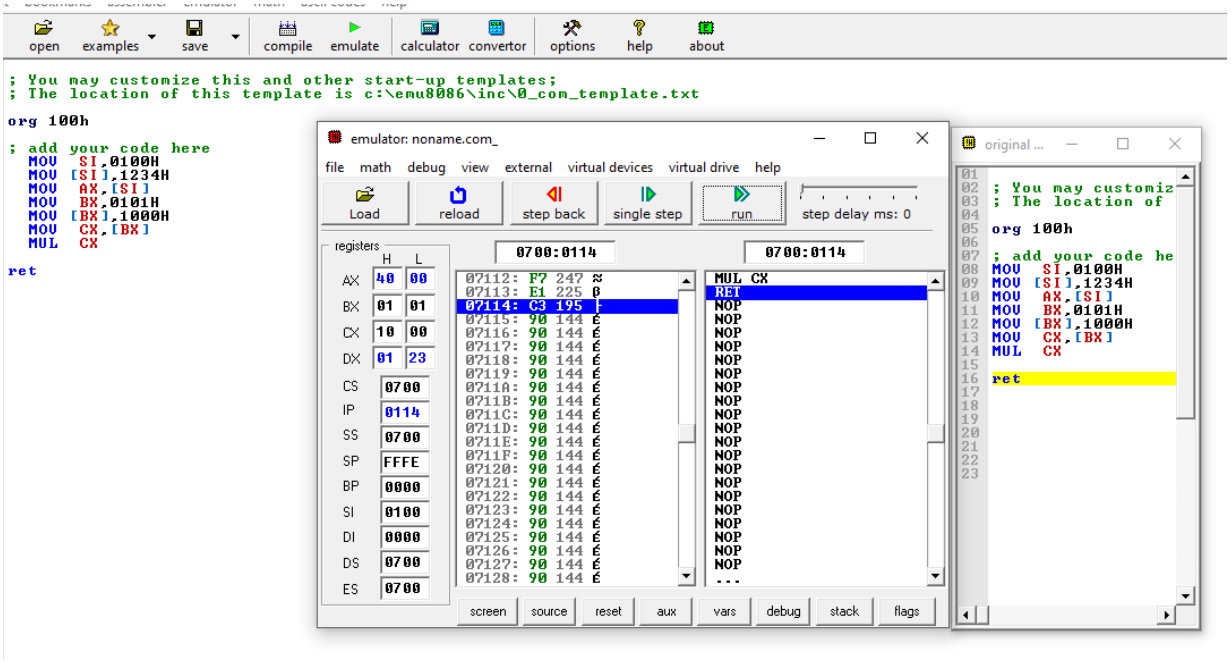
```
MOV AL,11h
MOV BL,66h
DIV BL
```



Write a code to perform multiplication on 16 bit numbers in consecutive memory locations?

SOURCE CODE:

```
MOV  SI, 0100H
MOV [SI], 1234H
MOV  AX, [SI]
MOV  BX, 0101H
MOV [BX], 1000H
MOV  CX, [BX]
MUL  CX
```



Write a code to add 16 bit numbers and find the average of numbers?

SOURCE CODE:

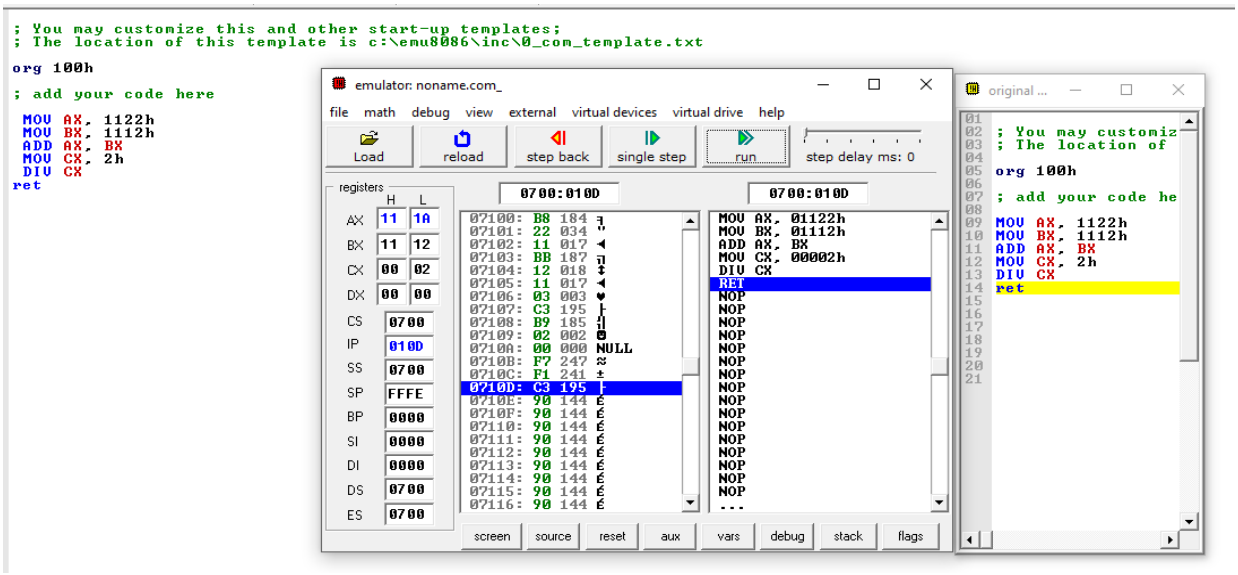
MOV AX, 1122h

MOV BX, 1112h

ADD AX, BX

MOV CX, 2h

DIV CX



Task 5:

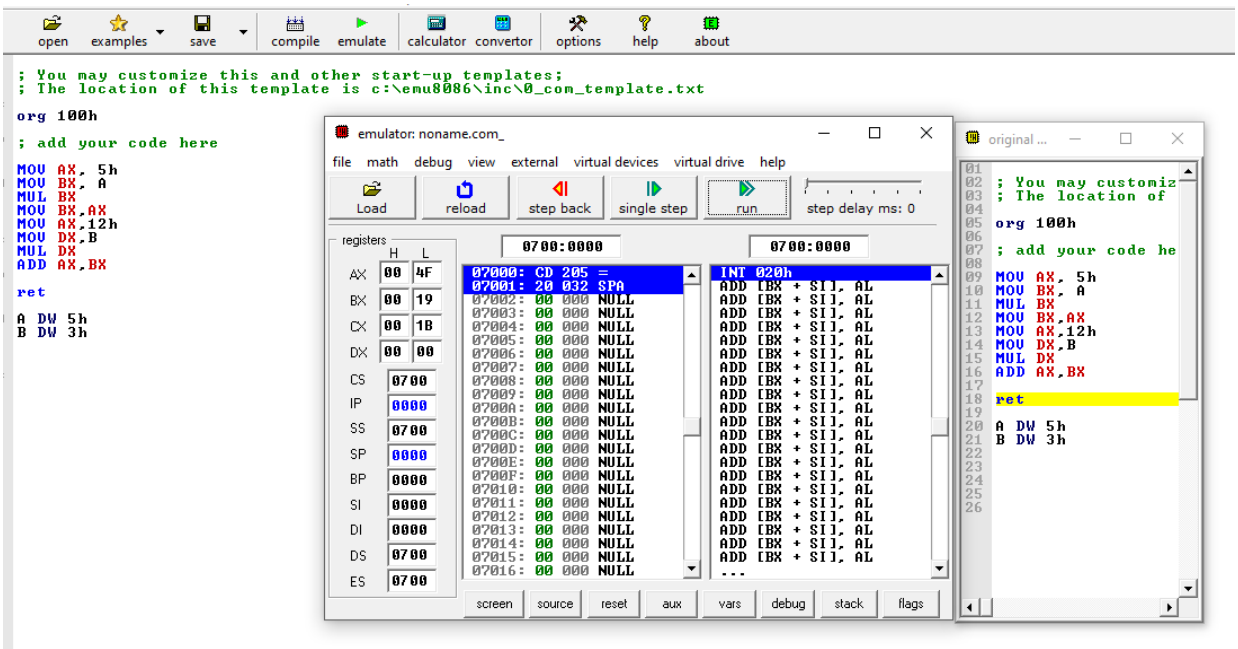
Translate the high-level language assignment statement: A=5×A+12×B
Let A and B be word variables, and suppose there is no overflow.

SOURCE CODE:

```
MOV AX, 5h
MOV BX, A
MUL BX
MOV BX,AX
MOV AX,12h
MOV DX,B
MUL DX
ADD AX,BX

ret

A DW 5h
B DW 3h
```

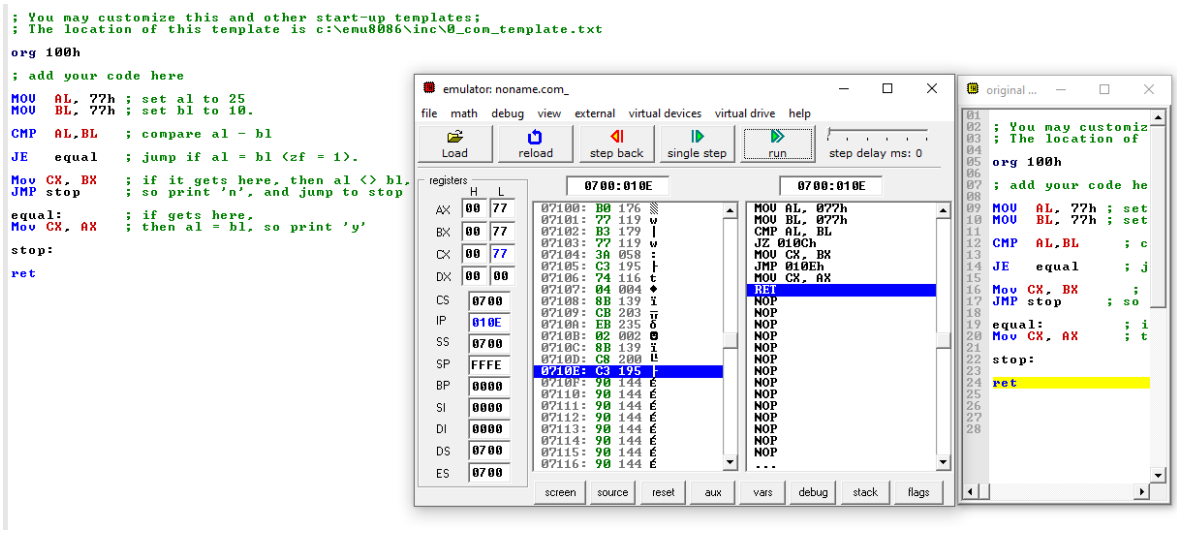


Task 6:

Try the example no 7 with different numbers for AL and BL, open flags by clicking on flags button, use single step and see what happens

SOURCE CODE:

```
MOV  AL, 77h ; set al to 25
MOV  BL, 77h ; set bl to 10.
CMP  AL,BL   ; compare al - bl
JE   equal   ; jump if al = bl (zf = 1).
Mov CX, BX   ; if it gets here, then al <> bl,
JMP stop     ; so print 'n', and jump to stop
equal:       ; if gets here,
Mov CX, AX   ; then al = bl, so print 'y'
stop:
ret
```



----------THE END----------