Microprocessor and Assembly Language Lab

Lab Material 6 for CSE 312 (M&AL Lab)

Dr. Shah Murtaza Rashid Al Masud
Associate Prof.
Dept. of CSE, UAP

MUL and DIV Operations

MUL Instruction

The MUL (unsigned multiply) instruction multiplies an 8, 16, or 32 bit operand by either AL, AX, or EAX. The instruction formats are:

MUL *r/m8*
MUL *r/m16*
MUL *r/m32*

The single operand is the multiplier. The following table shows the default multiplicand and product, depending on the size of the multiplier:

MUL Instruction

Multiplicand	Multiplier	Product
AL	*r/m8*	AX
AX	*r/m16*	DX:AX
EAX	*r/m32*	EDX:EAX

The register(s) holding the product are **twice the size of the multiplicand and multiplier**, guaranteeing that overflow will never occur.

MUL Instruction

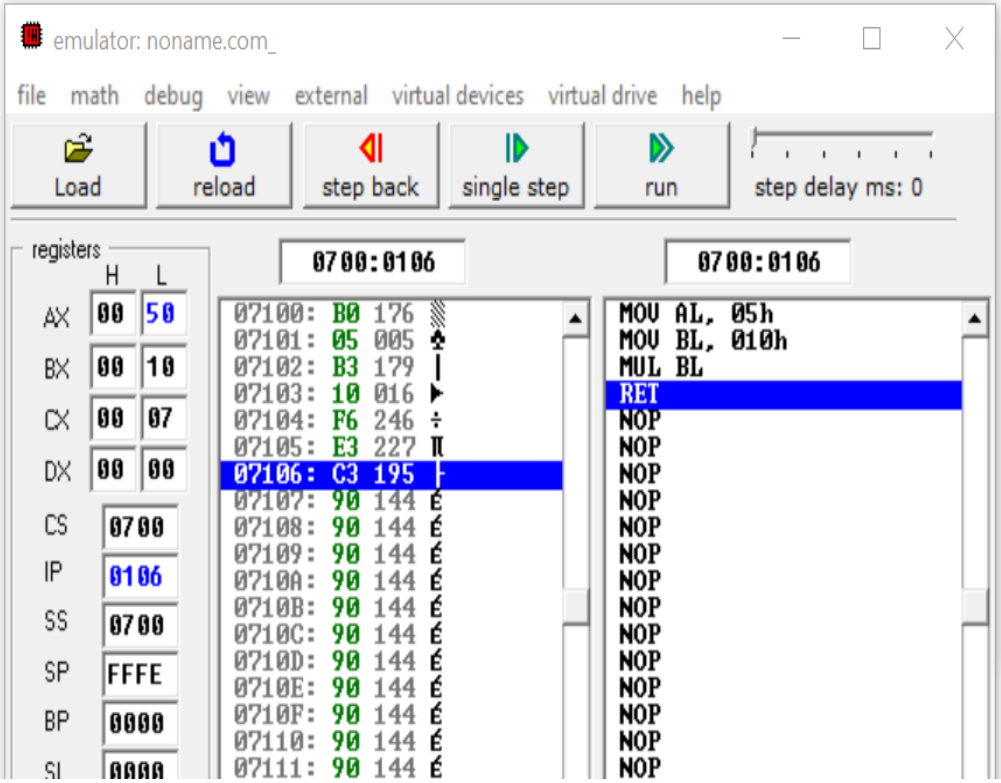
Example 1: The following statements perform 8-bit unsigned multiplication (5 * l0h), producing 50h in AX:

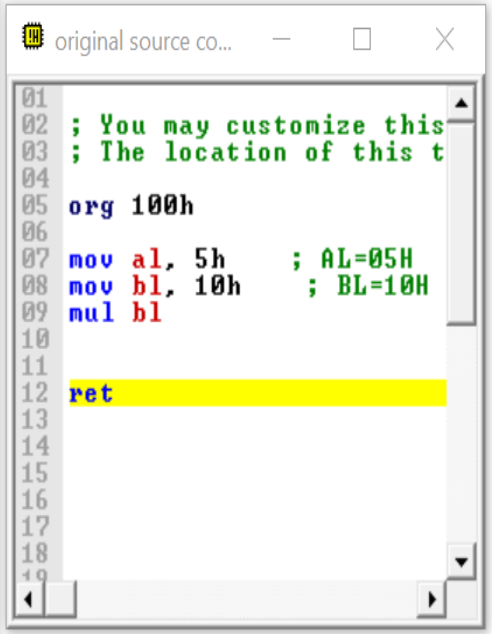
```
mov al,5h     ; AL=05H
mov bl,l0h    ; BL=10H
mul bl                ; AL*BL=5H*10H=50H→AX  CF = 0
```

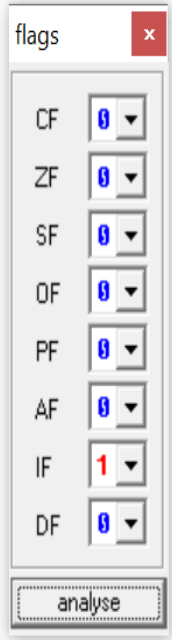
The Carry flag is clear because AH (the upper half of the product) equals zero.

MUL Instruction

```
org 100h

mov al, 5h  ; AL=05H
mov bl, 10h ; BL=10H
mul bl

ret
```

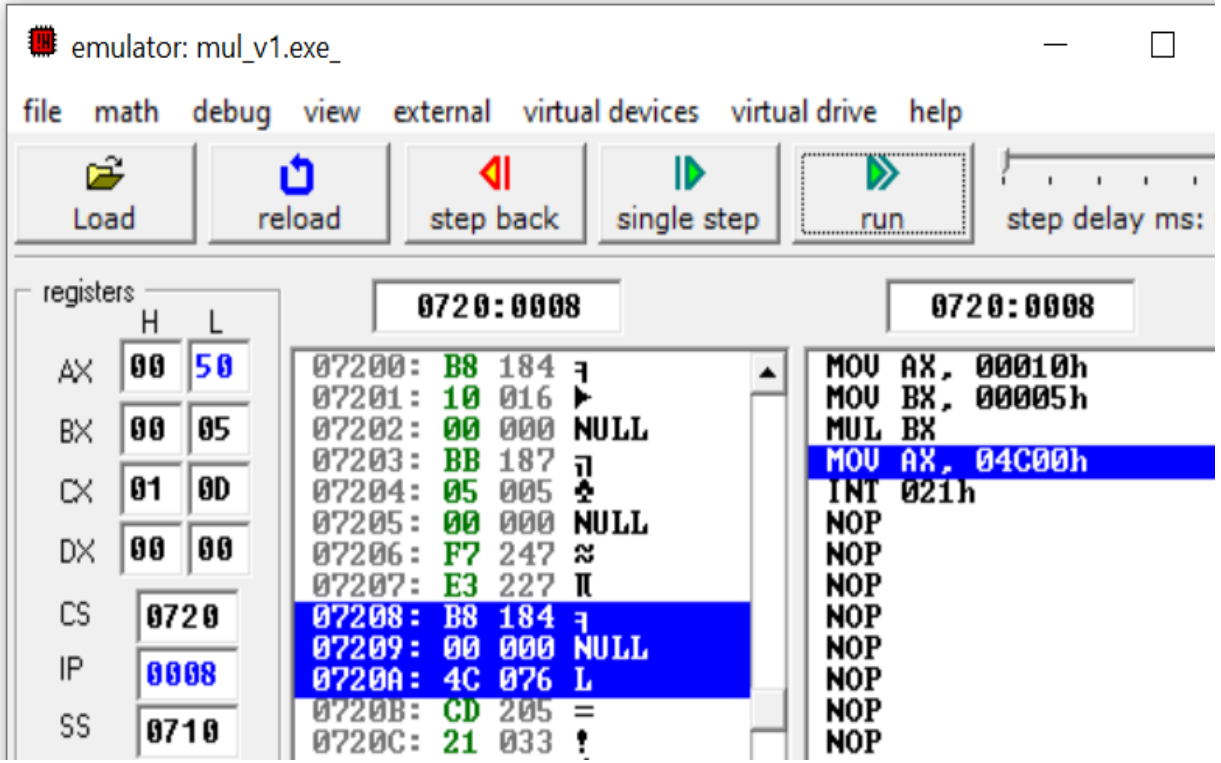






MUL Instruction

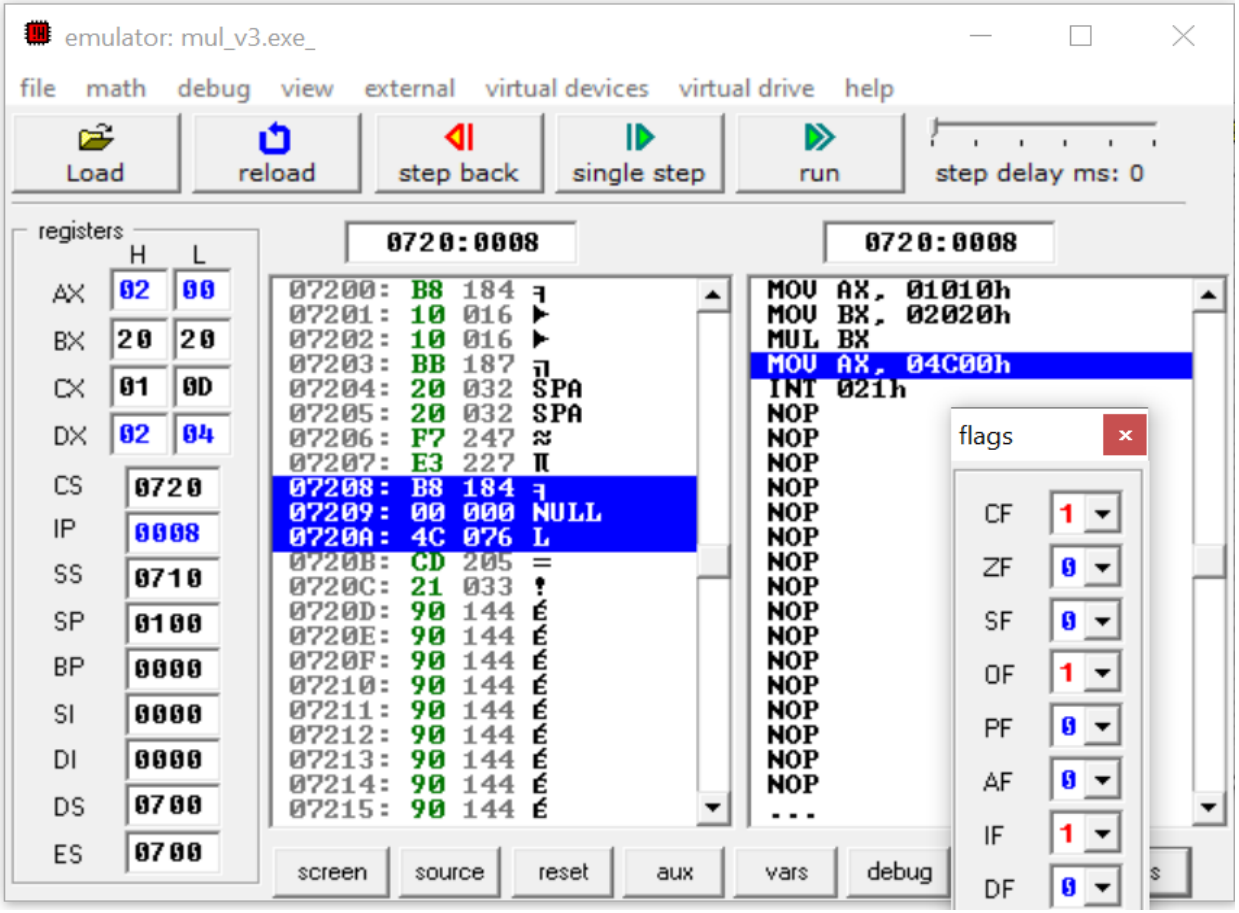
```
.MODEL SMALL
.STACK 100H
.DATA

.CODE SEGMENT
MAIN PROC


        MOV AX,10H
        MOV BX,05H
        MUL BX


        MOV AX,4C00H
        INT 21H

      MAIN ENDP
END MAIN
```



First operand is multiplicand and the second one is multiplier. Result is stored in AX

MUL Instruction

```
.MODEL SMALL
.STACK 100H
.DATA

.CODE SEGMENT
MAIN PROC


        MOV AX,1010H
        MOV BX,2020H

        MUL BX


        MOV AX,4C00H
        INT 21H

      MAIN ENDP
END MAIN
```



First operand is multiplicand and the second one is multiplier. Result is stored in AX

MUL Instruction

Example 2: The following statements perform 16-bit unsigned multiplication (0010h * 2000h) producing 00020000h in DX:AX:

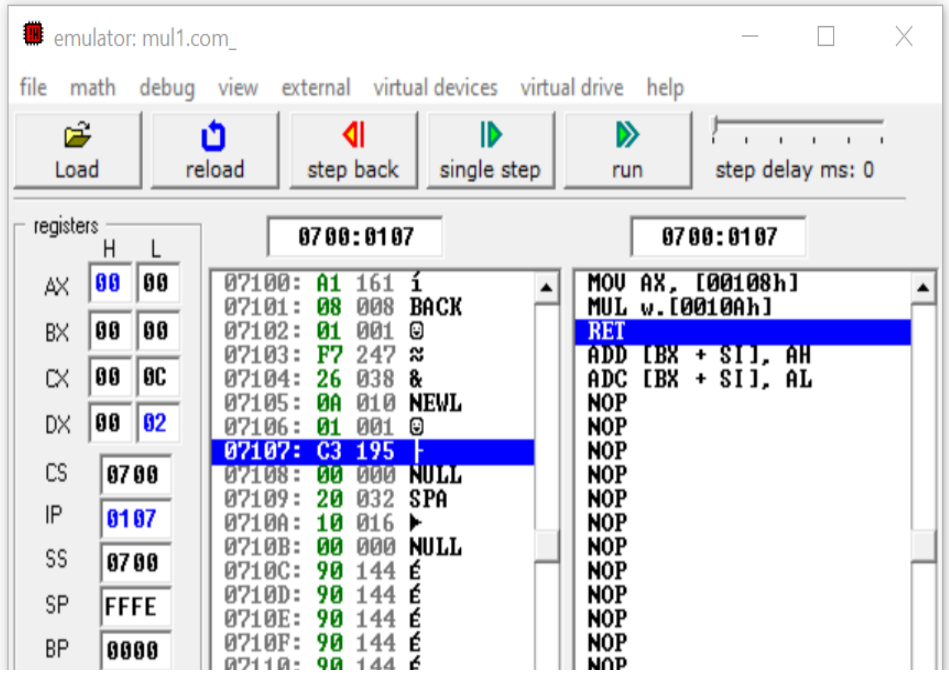
```
. data
val1 WORD 2000h
val2 WORD 0010h
.code
mov ax, val1 ; AX=2000H
mul val2       ;
```

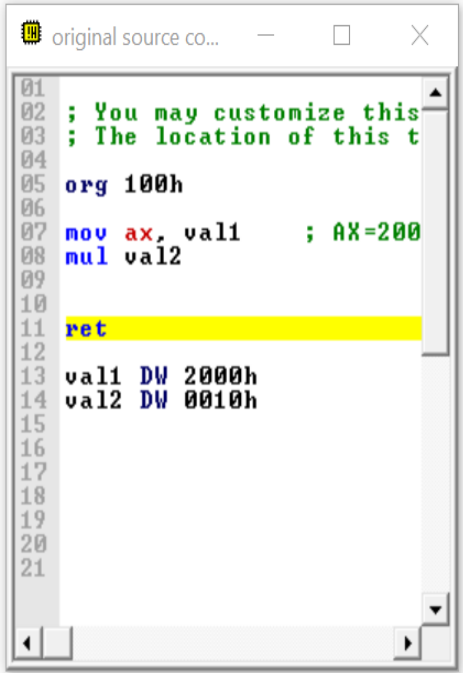
AX*val2=2000H*0010H=00020000h→DX:AX CF = 1

The Carry flag is set because DX is not equal to zero.

MUL Instruction

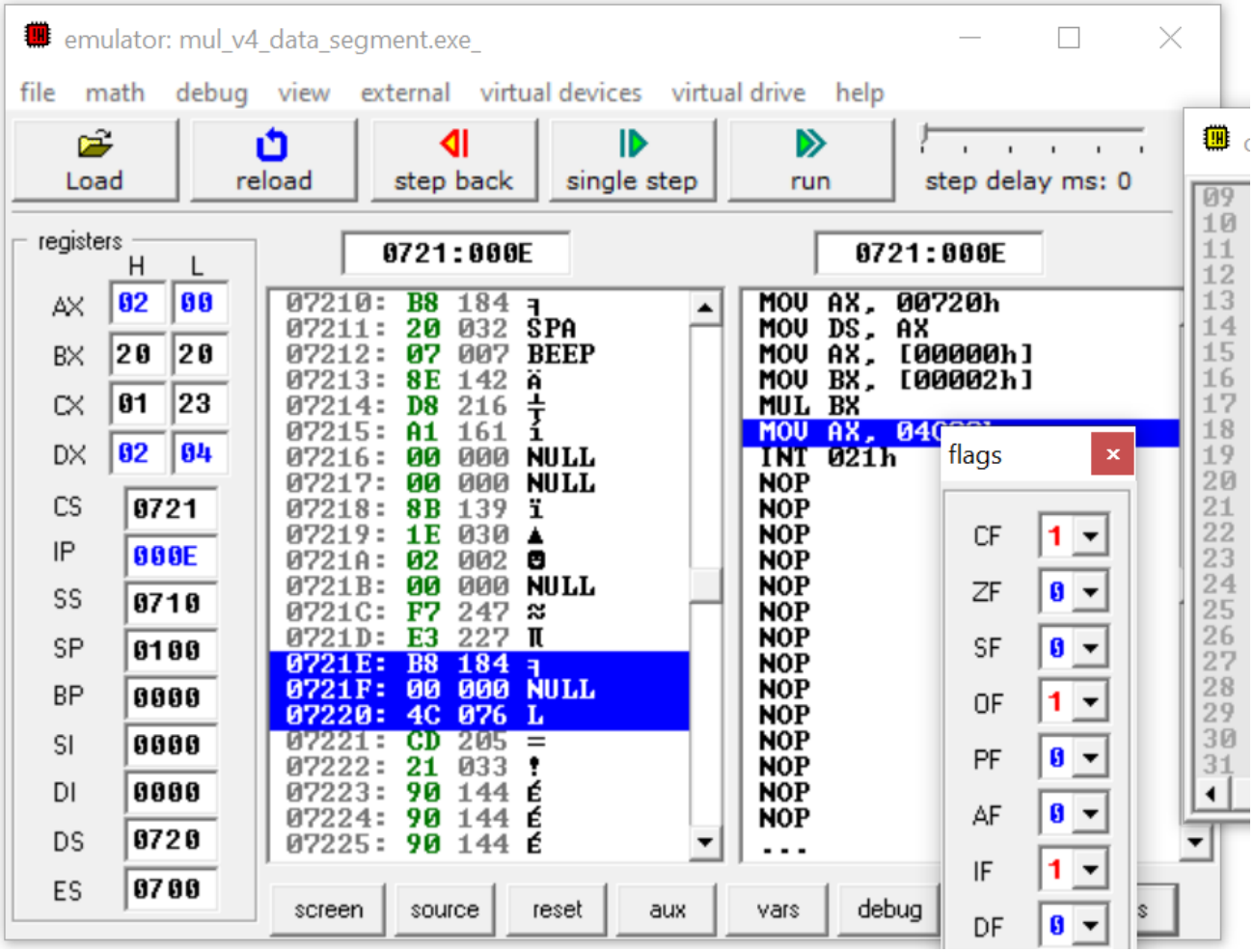
```
org 100h

mov ax, val1     ; AX=2000H
mul val2


ret

val1 DW 2000h
val2 DW 0010h
```





MUL Instruction (MOV CX, 0000H)

```
.MODEL SMALL
.STACK 100H
.DATA

val1 DW 1010h
val2 DW 2020h

.CODE SEGMENT
MAIN PROC


        MOV AX, @DATA
        MOV DS, AX

        MOV AX,val1
        MOV BX,val2

        MUL BX


        MOV AX,4C00H
        INT 21H

      MAIN ENDP
END MAIN
```



MUL Instruction

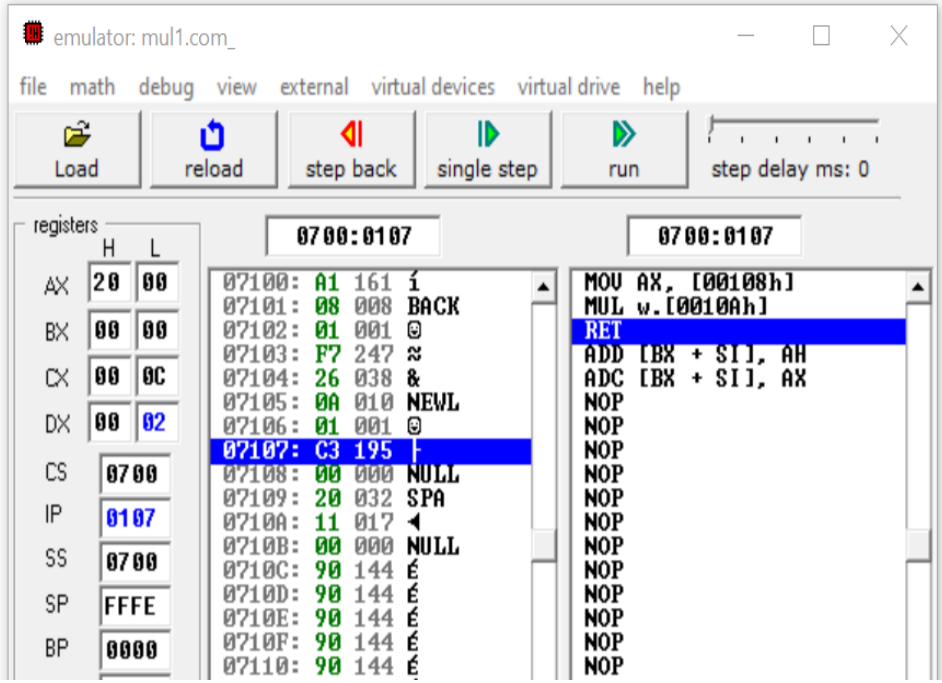
Example 2: The following statements perform 16-bit unsigned multiplication (0011h * 2000h) producing 00022000h in DX:AX:

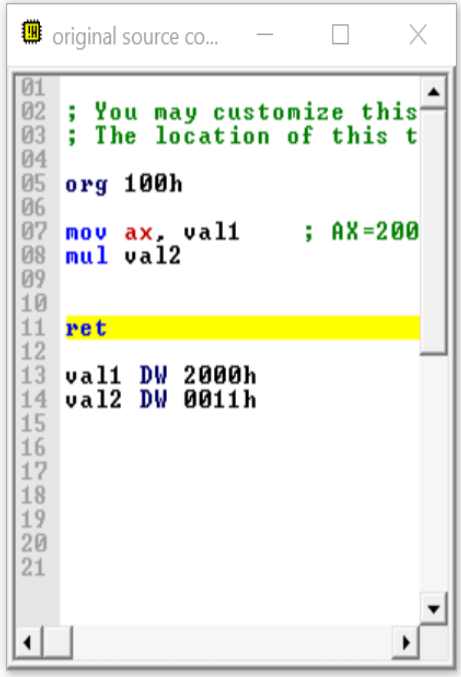
```
. data
val1 WORD 2000h
val2 WORD 0011h
.code
mov ax, vall      ; AX=2000H
mul val2          ; AX*val2=2000H*0011H=00021000h→DX:AX
CF = 1
```

The Carry flag is set because DX is not equal to zero.

MUL Instruction

```
org 100h

mov ax, val1     ; AX=2000H
mul val2


ret

val1 DW 2000h
val2 DW 0011h
```

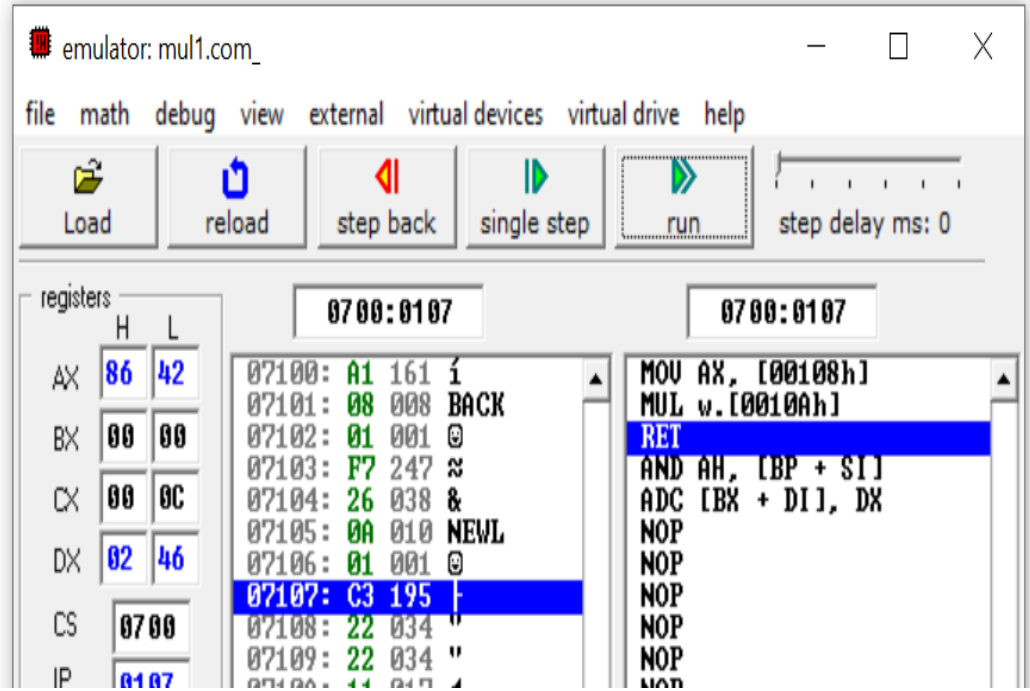


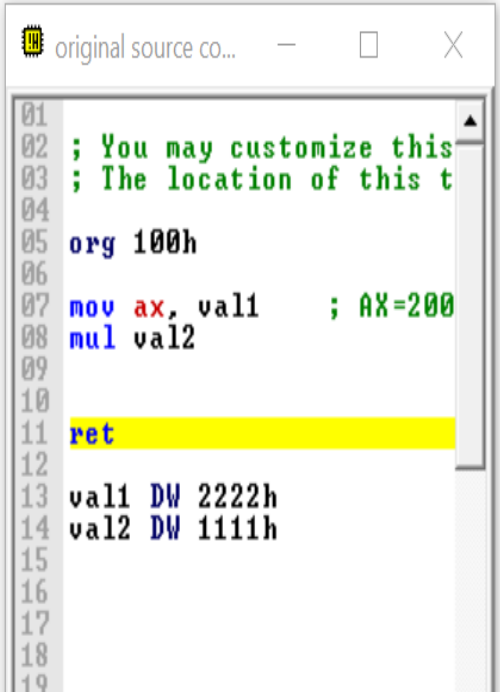


MUL Instruction

Another example with different operand

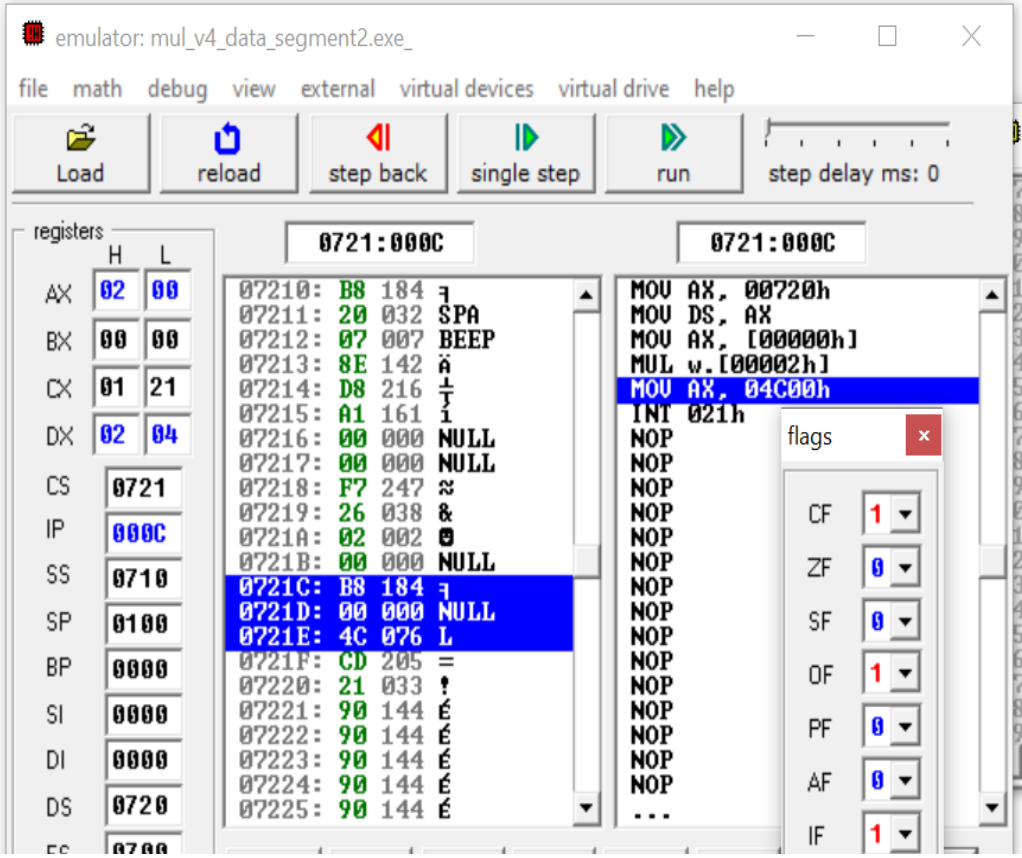
```
org 100h

mov ax, val1     ; AX=2000H
mul val2


ret

val1 DW 2222h
val2 DW 1111h
```

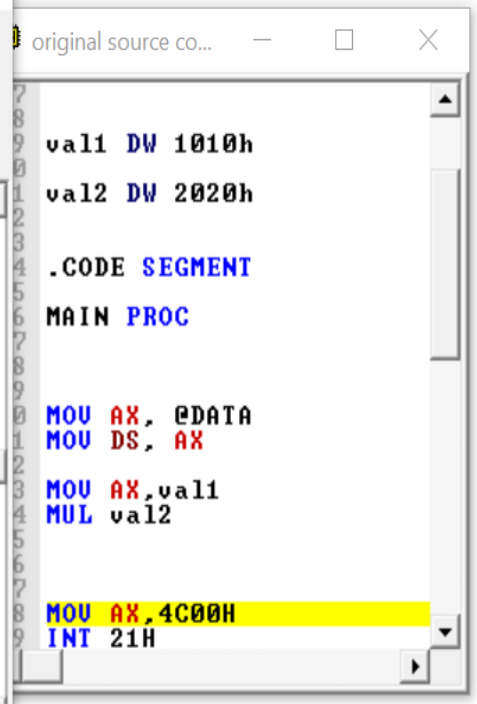




MUL Instruction

```
.MODEL SMALL
.STACK 100H
.DATA

val1 DW 1010h
val2 DW 2020h

.CODE SEGMENT
MAIN PROC


        MOV AX, @DATA
        MOV DS, AX

        MOV AX,val1
        MUL val2


        MOV AX,4C00H
        INT 21H

      MAIN ENDP
END MAIN
```





DIV Instruction

The DIV (unsigned divide) instruction performs 8-bit, 16-bit, and 32-bit division on unsigned integers. A single operand is supplied (register or memory operand), which is assumed to be the divisor. The instruction formats for DIV are:

DIV *r/m8*

DIV *r/m16*

DIV r/m32

The following table shows the relationship between the dividend, divisor, quotient, and remainder. Everything is determined by the size of the **divisor**:

DIV Instruction

The following table shows the relationship between the dividend, divisor, quotient, and remainder. Everything is determined by the size of the **divisor**:

Dividend	Divisor	Quotient	Remainder
AX	*r/m8*	AL	AH
DX:AX	*r/m16*	AX	DX
EDX:EAX	*r/m32*	EAX	EDX

DIV Instruction

***Example* 1**: The following instructions perform 8-bit unsigned division (83h / 2), producing a quotient of 41h and a remainder of 1:

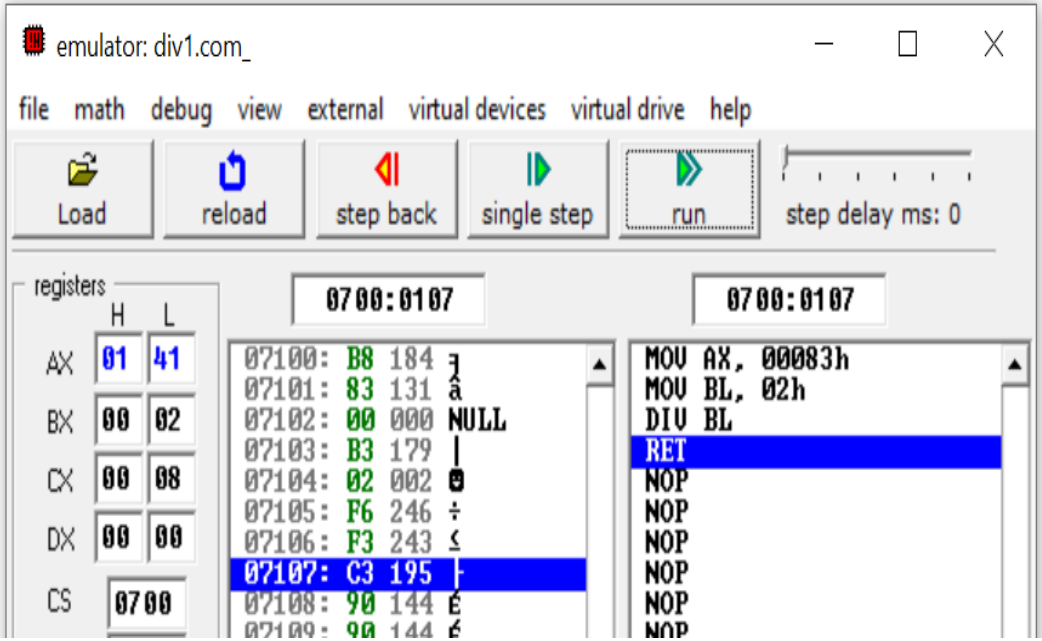
```
mov ax,0083h          ; AX=0083H
mov bl,2              ; BL=02H
div bl                ; AX/BL=0083H/02H→AL=41H, AH=01H
```

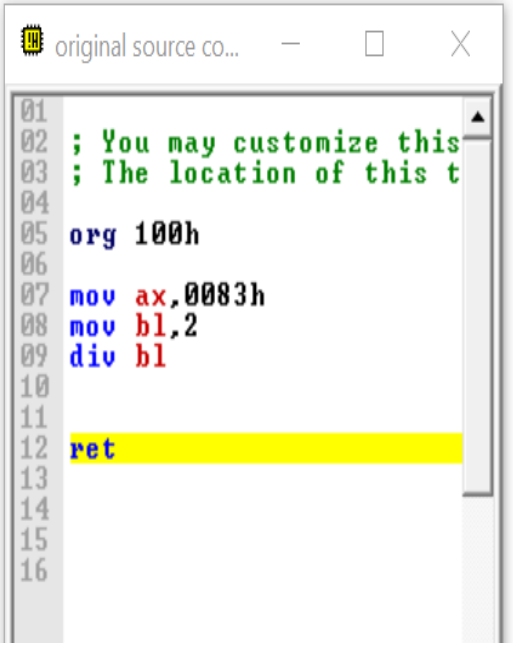
First operand is dividend and the second one is divisor.

Quotient or Result is stored in AL and the Reminder in AH

DIV Instruction

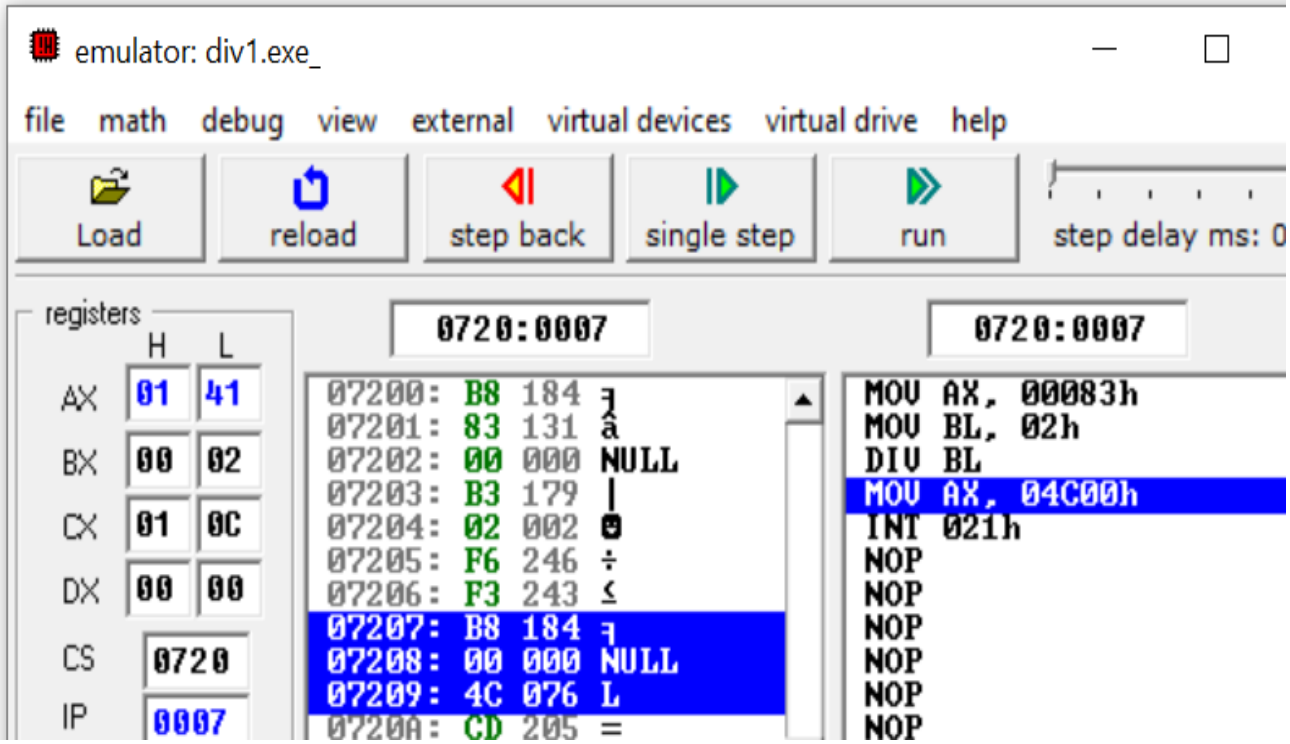
```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

mov ax,0083h
mov bl,2
div bl


ret
```





DIV Instruction

```
 .MODEL SMALL
.STACK 100H
.DATA

.CODE SEGMENT
MAIN PROC
        mov ax,0083h
        mov bl,2
        div bl

         MOV AX,4C00H
         INT 21H

       MAIN ENDP
END MAIN
```



DIV Instruction

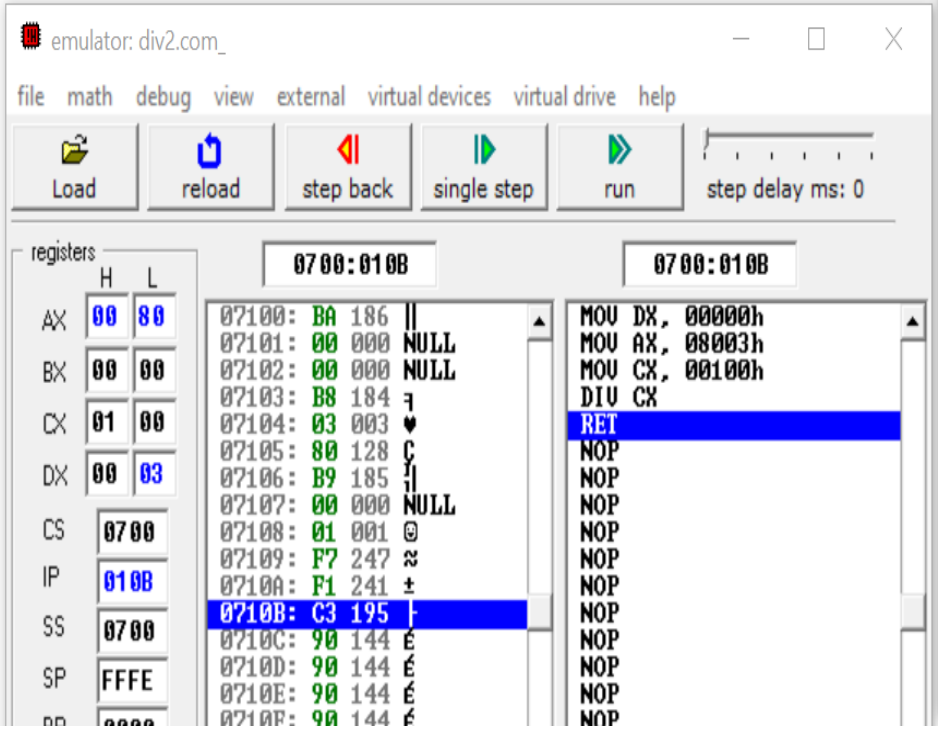
***Example* 2**: The following instructions perform 16-bit unsigned division (8003h / 100h), producing a quotient of 80h and a remainder of 3.

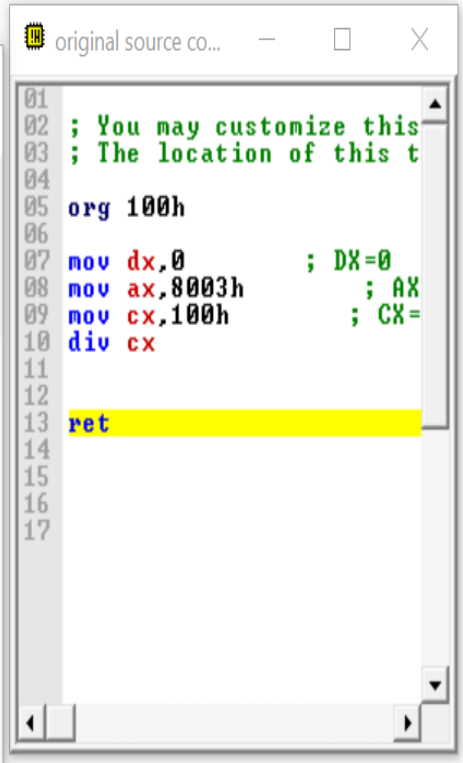
DX contains the high part of the dividend, so it must be cleared before the DIV instruction executes:

```
mov dx,0          ; DX=0
mov ax,8003h      ; AX=8003H
mov cx,100h       ; CX=100H
div cx            ; AX=0800H DX=0003H
```

DIV Instruction

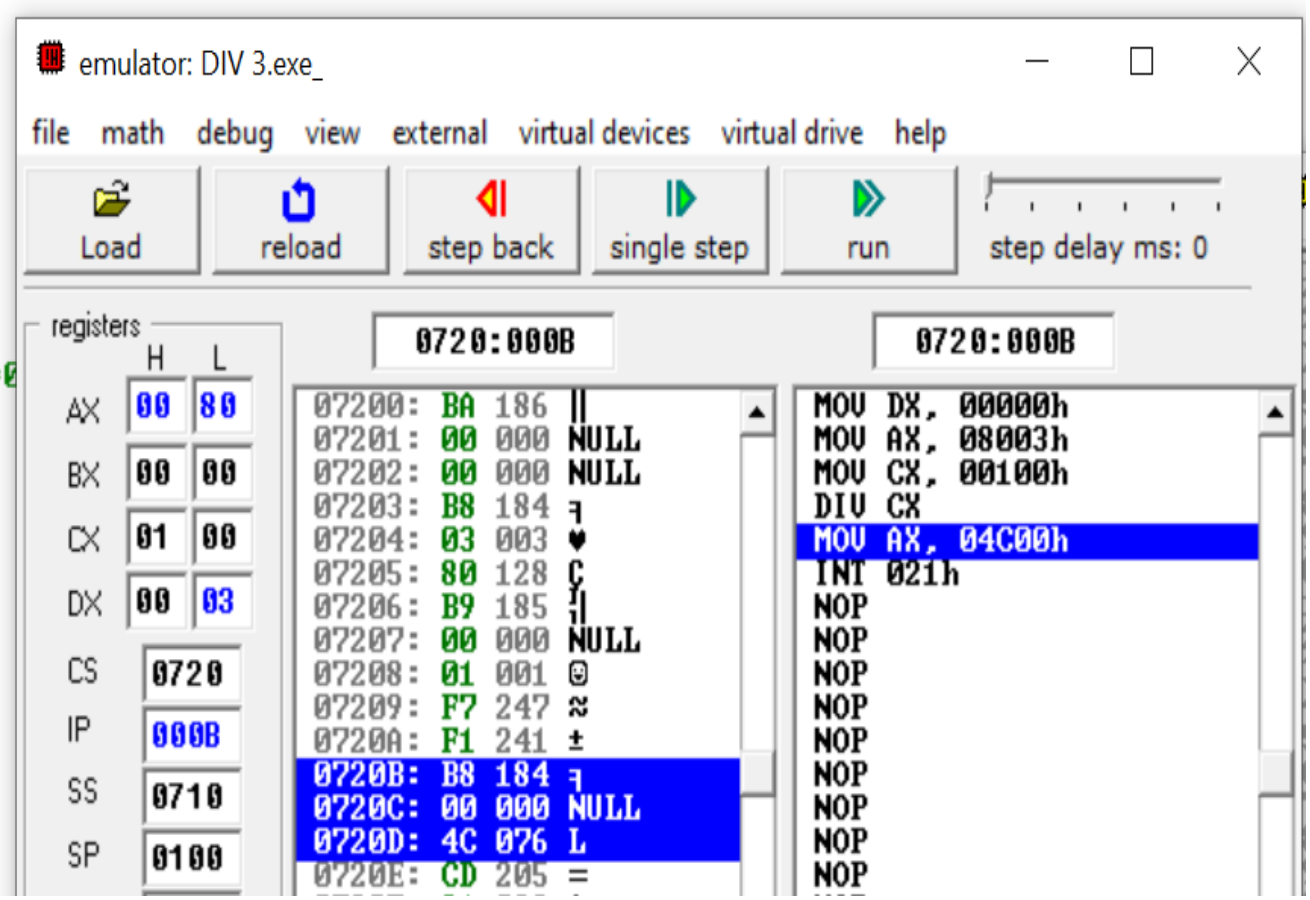
```
org 100h

mov dx,0          ; DX=0
mov ax,8003h        ; AX=8003H
mov cx,100h       ; CX=100H
div cx


ret
```





DIV Instruction

```
 .MODEL SMALL
.STACK 100H
.DATA

.CODE SEGMENT
MAIN PROC

        mov dx,0        ; DX=0
        mov ax,8003h      ;
        mov cx,100h       ;
        div cx            ;


         MOV AX,4C00H
         INT 21H


        MAIN ENDP
END MAIN
```



First operand is dividend and the second one is divisor.

Quotient or Result is stored in AX and the Reminder in DX