EXPERIMENT 05
Multiplication and Division Instructions

Objective

- Understand Multiplication and Division instructions
- Working of ASCII adjust instruction

Theory

Multiplication Instruction

There are two instructions for multiplication, MUL (Multiplication) and IMUL (Integer Multiplication) Instruction.

Instruction format are:

```
MUL   Source      ; Unsigned multiplication
IMUL  Source      ; Integer multiplication for Signed number
```

The source may be a register or memory but not a constant. Also the source value and the multiplying value should be of same magnitude.

Byte Form Multiplication

In the byte form of multiplication **one number should present in source and other in AL register**. The result may be of 16-bit (1 word) which is stored in AX register.

Word Form Multiplication

In the word form of multiplication one number is present in source and other is in AX register. The result may be of 32-bit (Double word) so most significant 16-bit goes in DX and least significant 16-bit goes to AX register.

AAM (ASCII Adjust for Multiplication) Instruction

The AAM instruction converts the result of two BCD digit multiplications into unpacked BCD format. This instruction convert the result into BCD whose value is lower then 100_D or 64_H, this is because the result is store in AX register in the unpacked form that are AL and AH so largest value is 99_D which store as AH = 09 and AL = 09 . The operation for the instruction is the result which is store in AL register is divided by 10_D and the quotient store in AL register and while remainder store in AH register.

E.g. MUL Instruction

```
MOV   AL, 05H     ; Move value in AL which is source
MOV   BL, 07H     ; Move value in BL which is source
MUL   BL          ; AX = 05H x 07H = 23H
AAM               ; AX = 0305H (5 x 7 = 35)
```

```
TITLE EXAMPLE_PROG_MUL
.MODEL SMALL
.STACK 100H
.DATA
MESSAGE1 DB 0AH, 0DH, "ENTER DIGIT 1:$"
MESSAGE2 DB 0AH, 0DH, "ENTER DIGIT 2:$"
RESULT   DB 0AH, 0DH, "RESULT IS:$"
.CODE
START:
    MOV AX, @DATA
    MOV DS, AX
    MOV DX, OFFSET MESSAGE1
    MOV AH, 09H
    INT 21H
    MOV AH, 01H
    INT 21H
    SUB AL, 30H; ASCCI REMOVED
    MOV BL, AL ; STORING VALUE 1

    MOV DX, OFFSET MESSAGE2
    MOV AH, 09H
    INT 21H
    MOV AH, 01H
    INT 21H

    SUB AL, 30H; ASCII REMOVED
    MOV BH, AL ; STORING VALUE 2
    ;-------------------------------------------------------
    ; FOR MULTIPLICATION ONE VALUE SHOULD BE IN AL REGISTER
    ;-------------------------------------------------------
    MOV AL, BL ; MOV VALUE1 TO AL AGAIN OTHER VALUE IS STORED IN BH
    MOV AH, 00H; CLEARING AH BEFORE MULTIPLICATION

    MUL BH ; ---->> AL*BH, RESULT IS IN AX
    AAM        ; ASCII ADJUST FOR MULTIPLICATION, AH HAS THE FIRST VALUE, AL HAS THE SECOND VALUE
    MOV BX, AX ; STORING RESULT FOR FUTURE USE

    MOV DX, OFFSET RESULT
    MOV AH, 09H
    INT 21H

    MOV DL, BH   ; FIRST VALUE DISPLAY
    ADD DL, 30H  ; ADDING ASCII
    MOV AH, 02H
    INT 21H

    MOV DL, BL   ; SECOND VALUE DISPLAY
    ADD DL, 30H  ; ADDING ASCII
    MOV AH, 02H
    INT 21H

    MOV AH, 4CH
    INT 21H
END START
```

Division Instruction

Likewise multiplication instruction there are two division instruction which are DIV (Division) and IDIV (Integer Division) Instruction.

Instruction format are:

```
DIV   Divisor      ; Unsigned Division
IDIV  Divisor      ; Integer Division for Signed number
```

The source may be a register or memory but not a constant as in multiplication instruction. If dividend is of 16-bit then divisor is of 8-bit also if dividend is of 32-bit then divisor is of 16-bit. For signed division, the remainder has the same sign as the sign of dividend.

Byte Form Division

In the byte form of division the **16-bit dividend is placed in AX register and the divisor is in 8-bit register. After division the 8-bit quotient is in AL register while 8-bit remainder in AH register**.

Word Form Division

In the word form of division the 32-bit dividend is DX (higher bits of word) and AX (lower bits of word) registers and divisor is in 16-bit register. After division the 16-bit quotient is in AX register while 16-bit remainder in DX registers.

E.g. DIV Instruction

```
MOV   AX, 0009H    ; AX = 0009H
MOV   BL, 02H      ; BL = 02H
DIV   BL           ; AX = AX / BL (AX = 0104H where 04 in AL is quotient and 01 in
                     AH is remainder)
```

E.g. IDIV Instruction

```
MOV   AX, -0009D   ; AX = -0009D
MOV   BL, 07H      ; BL = 07H
IDIV  BL           ; AX = AX / BL (AX = FEFFH where FF in AL is quotient and FE in
                     AH is remainder)
```

AAD (ASCII adjust for division) Instruction

The AAD instruction converts the unpacked BCD into binary and store in AL or converts unpacked BCD into packed BCD and store in AL. From the AX register the AH register is multiply by 10_D or $0A_H$ then add it to AL register it also clear the AH register.

```
MOV   AX, 0302H    ; DIVIDEND (32 in unpacked form)
AAD                ; AX = 0020H
MOV   BL, 4        ; DIVISOR
DIV   BL           ; AX = 0008H (32 / 4 = 8)
```

```
TITLE EXAMPLE_PROG_DIV
.MODEL SMALL
.STACK 100H
.DATA
MESSAGE1 DB 0AH, 0DH, "ENTER DIVIDENT:$"
MESSAGE2 DB 0AH, 0DH, "ENTER DIVISOR:$"
QUOTIENT   DB 0AH, 0DH, "QUOTIENT IS:$"
REMAINDER   DB 0AH, 0DH, "REMAINDER IS:$"
.CODE
START:
    MOV AX, @DATA
    MOV DS, AX
    MOV DX, OFFSET MESSAGE1
    MOV AH, 09H
    INT 21H
    MOV AH, 01H
    INT 21H
    SUB AL, 30H; ASCCI REMOVED
    MOV BL, AL ; STORING DIVIDENT
    MOV DX, OFFSET MESSAGE2
    MOV AH, 09H
    INT 21H
    MOV AH, 01H
    INT 21H
    SUB AL, 30H; ASCII REMOVED
    MOV BH, AL ; STORING DIVISOR
    ;-------------------------------------------------------------------------
    ; FOR DIVISION DIVIDENT MUST BE IN AX REGISTER AND DIVISOR IN OTHER REGISTER
    ;-------------------------------------------------------------------------
    MOV AL, BL ; MOV DIVIDENT TO AL AGAIN DIVISOR IS STORED IN BH
    MOV AH, 00H; CLEARING AH BEFORE DIVISION
    AAD        ; AAD IS ALWAYS USED BEFORE DIVISION
    DIV BH     ; ---->> AL/BH, QUOTIENT IN AL, REMAINDER IN AH
    MOV BX, AX ; STORING RESULT FOR FUTURE USE
    MOV DX, OFFSET QUOTIENT
    MOV AH, 09H
    INT 21H
    MOV DL, BL   ; QUOTIENT DISPLAY
    ADD DL, 30H  ; ADDING ASCII
    MOV AH, 02H
    INT 21H
    MOV DX, OFFSET QUOTIENT
    MOV AH, 09H
    INT 21H
    MOV DL, BH   ; REMAINDER DISPLAY
    ADD DL, 30H  ; ADDING ASCII
    MOV AH, 02H
    INT 21H
    MOV AH, 4CH
    INT 21H
END START
```

Multiplication and Division Instruction table

Function	**Instruction syntax**	**Operation type**	**Registers used in Operation**	**Registers used to store result**
Multiplication	MUL Source *for unsigned number* IMUL Source *for signed number*	Byte Format	Source may be register byte or memory byte and second byte in AL register	The 16-bits (1 word) result store in AX register
		Word Format	Source may be word register or memory word and second word in AX register	The result is of 32-bits (double word). The most significant 16-bits store in DX register while the least significant 16-bits store in AX register
Division	DIV Divisor *for unsigned number* IDIV Divisor *for signed number*	Byte Format	The 8-bits Divisor store in register byte or memory byte and 16-bits dividend store in AX register	The 8-bits (1 byte) quotient store in AL and 8-bits remainder store in AH register
		Word Format	The 16-bit Divisor store in register word or memory word and 32-bit dividend store in DX:AX registers	The 16-bit (1 word) Quotient store in AX and 16-bits remainder store in DX register

Note: For IMUL and IDIV if the sign bit for number is set then its 2`s complement will taken for calculation. And also if the sign for both the numbers are same then after calculation the result will store directly and if the sign are different then the 2`s complement of result will store.

Exercise

1) Write a program that perform all arithmetic operations.

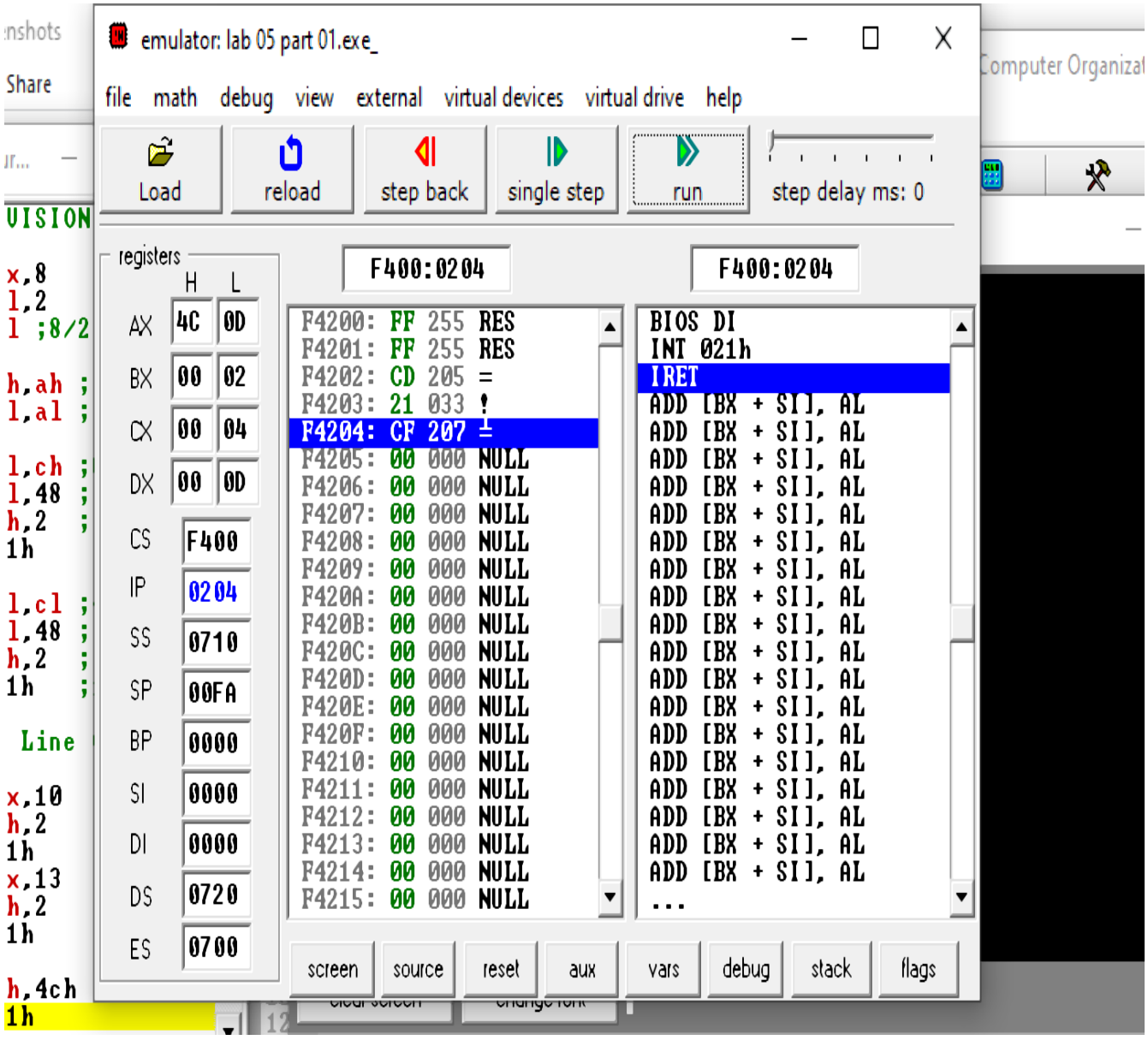
ew open examples save | compile emulate | calculator convertor | options help about

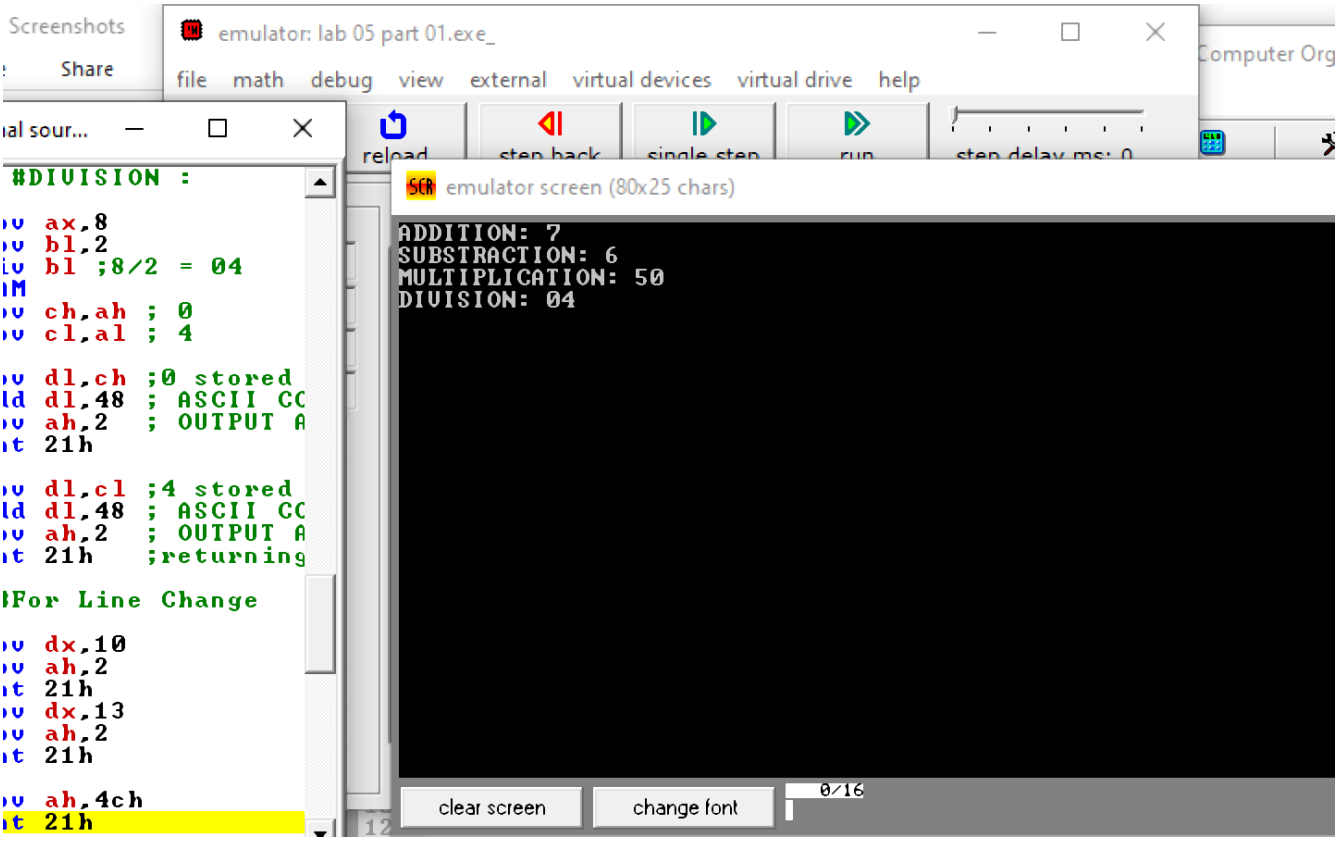
```
001 .model small
002 .stack
003 .data
004 A db 'ADDITION : $'
005 B db 'SUBSTRACTION : $'
006 C db 'MULTIPLICATION : $'
007 D db 'DIVISION : $'
008 .code
009     main proc
010             ;#Display FUNCTION NAME :
011         mov ax,@data
012     mov ds,ax
013     mov dx,offset A
014     mov ah,9
015     int 21h
016
017
018         ; #ADDITION :
019
020             mov bl,5 ;var1 = 1 --> 7
021           mov cl,2 ;var2 = 2
022
023
024
025           add bl,cl       ; 1+2=3 ; var2 = var2+var1
026           add bl, 48
027
028           mov dl,bl
029           mov ah,2
030           int 21h
031
032           ;#For Line Change
033
034      mov dx,10
035     mov ah,2
036     int 21h
```

```
037     mov dx,13
038     mov ah,2
039     int 21h
040
041             ;#Display FUNCTION NAME :
042         mov ax,@data
043     mov ds,ax
044     mov dx,offset B
045     mov ah,9
046     int 21h
047
048         ; #SUBSTRACTION :
049
050            mov bl,8 ;var1 = 1 --> 7
051            mov cl,2 ;var2 = 2
052
053
054
055            sub bl,cl      ; 1+2=3 ; var2 = var2+var1
056            add bl, 48
057
058            mov dl,bl
059            mov ah,2
060            int 21h
061
062                ;#For Line Change
063
064      mov dx,10
065     mov ah,2
066     int 21h
067     mov dx,13
068     mov ah,2
069     int 21h
070
071      ;#Display FUNCTION NAME :
072         mov ax,@data
```

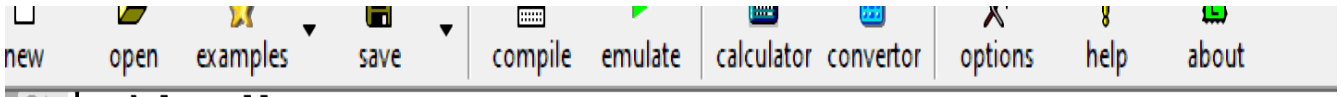
```
073     mov ds,ax
074     mov dx,offset C
075     mov ah,9
076     int 21h
077
078         ; #MULTIPLICATION :
079
080                 mov al,5
081     mov bl,10
082     mul bl ;5*10 = 50
083     AAM
084     mov ch,ah ; 5
085     mov cl,al ; 0
086
087     mov dl,ch ;5 stored
088     add dl,48 ; ASCII CONVERSION
089     mov ah,2  ; OUTPUT ASCII CONVERTED VALUE
090     int 21h
091
092        mov dl,cl ;0 stored
093     add dl,48 ; ASCII CONVERSION
094     mov ah,2  ; OUTPUT ASCII CONVERTED VALUE
095     int 21h   ;returning value
096
097          ;#For Line Change
098
099      mov dx,10
100     mov ah,2
101     int 21h
102     mov dx,13
103     mov ah,2
104     int 21h
105
106        ;#Display FUNCTION NAME :
107         mov ax,@data
108     mov ds,ax
109     mov dx,offset D
```

```
107     mov dx,offset b
110     mov ah,9
111     int 21h
112
113    ; #DIVISION :
114
115         mov ax,8
116     mov bl,2
117     div bl ;8/2 = 04
118     AAM
119     mov ch,ah ; 0
120     mov cl,al ; 4
121
122     mov dl,ch ;0 stored
123     add dl,48 ; ASCII CONVERSION
124     mov ah,2  ; OUTPUT ASCII CONVERTED VALUE
125     int 21h
126
127        mov dl,cl ;4 stored
128     add dl,48 ; ASCII CONVERSION
129     mov ah,2  ; OUTPUT ASCII CONVERTED VALUE
130     int 21h   ;returning value
131
132     ;#For Line Change
133
134      mov dx,10
135     mov ah,2
136     int 21h
137     mov dx,13
138     mov ah,2
139     int 21h
140
141        mov ah,4ch
142     int 21h
143
144
145 endp
146 end main
```


emulator: lab 05 part 01.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 4C 0D
BX 00 02
CX 00 04
DX 00 0D
CS F400
IP 0204
SS 0710
SP 00FA
BP 0000
SI 0000
DI 0000
DS 0720
ES 0700
F400:0204
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 Ï
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
...
screen source reset aux vars debug stack flags

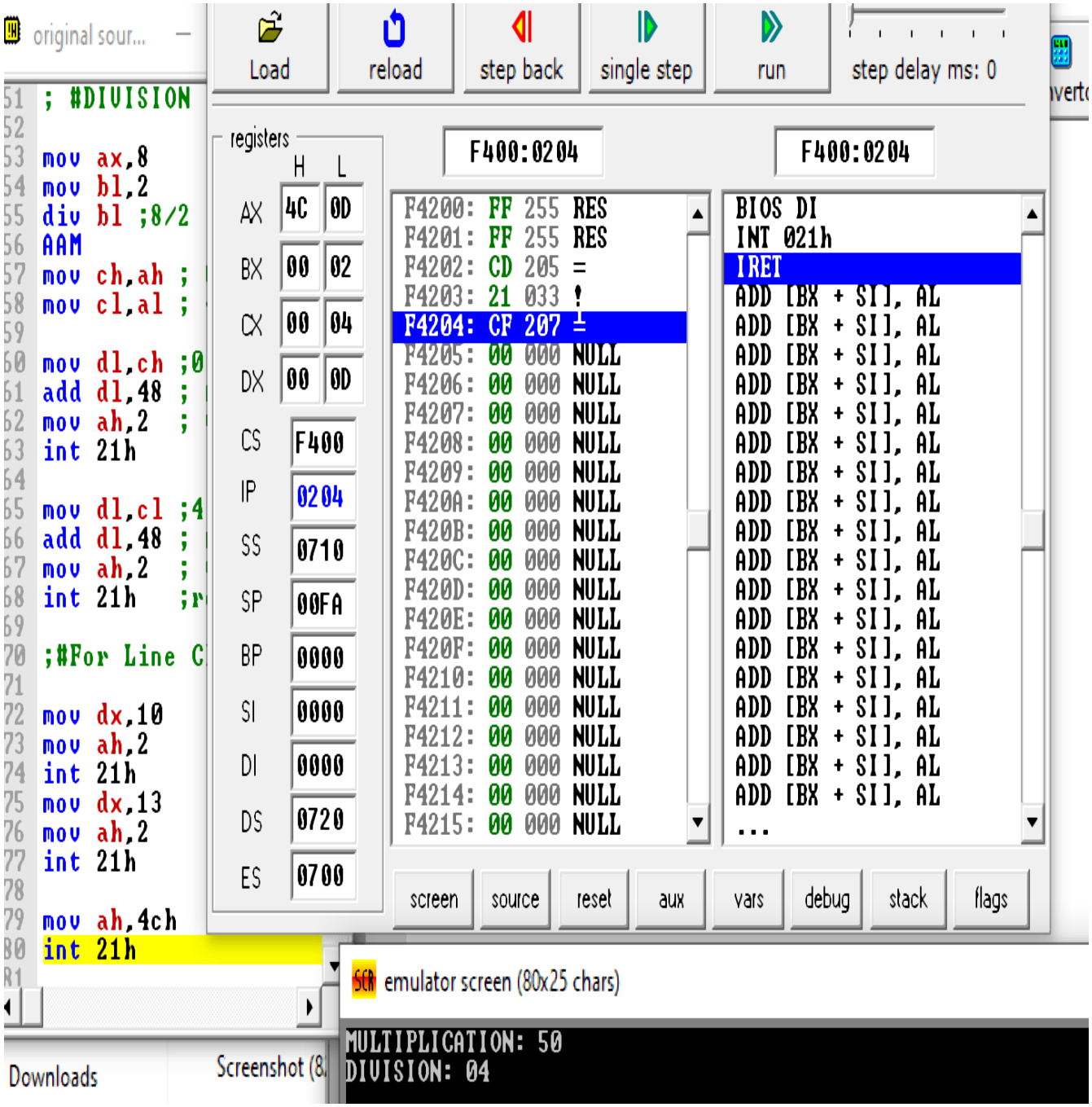


2) Write a program for multiplication and division




```
.model small
.stack
.data
C db 'MULTIPLICATION : $'
D db 'DIVISION : $'
.code
    main proc

            ;#Display FUNCTION NAME :
        mov ax,@data
    mov ds,ax
    mov dx,offset C
    mov ah,9
    int 21h

        ; #MULTIPLICATION :

                mov al,5
    mov bl,10
    mul bl ;5*10 = 50
    AAM
    mov ch,ah ; 5
    mov cl,al ; 0

    mov dl,ch ;5 stored
    add dl,48 ; ASCII CONVERSION
    mov ah,2  ; OUTPUT ASCII CONVERTED VALUE
    int 21h

       mov dl,cl ;0 stored
    add dl,48 ; ASCII CONVERSION
    mov ah,2  ; OUTPUT ASCII CONVERTED VALUE
    int 21h   ;returning value

         ;#For Line Change
```

```
     mov dx,10
    mov ah,2
    int 21h
    mov dx,13
    mov ah,2
    int 21h

       ;#Display FUNCTION NAME :
        mov ax,@data
    mov ds,ax
    mov dx,offset D
    mov ah,9
    int 21h

   ; #DIVISION :

        mov ax,8
    mov bl,2
    div bl ;8/2 = 04
    AAM
    mov ch,ah ; 0
    mov cl,al ; 4

    mov dl,ch ;0 stored
    add dl,48 ; ASCII CONVERSION
    mov ah,2  ; OUTPUT ASCII CONVERTED VALUE
    int 21h

       mov dl,cl ;4 stored
    add dl,48 ; ASCII CONVERSION
    mov ah,2  ; OUTPUT ASCII CONVERTED VALUE
    int 21h   ;returning value

    ;#For Line Change

     mov dx,10
    mov ah,2
    int 21h
      mov dx,10
     mov ah,2
     int 21h
     mov dx,13
     mov ah,2
     int 21h

        mov ah,4ch
     int 21h


endp
end main
```


original sour...
; #DIVISION
mov ax,8
mov bl,2
div bl ;8/2
AAM
mov ch,ah ;
mov cl,al ;
mov dl,ch ;0
add dl,48 ;
mov ah,2 ;
int 21h
mov dl,cl ;4
add dl,48 ;
mov ah,2 ;
int 21h ;r
;#For Line C
mov dx,10
mov ah,2
int 21h
mov dx,13
mov ah,2
int 21h
mov ah,4ch
int 21h
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 4C 0D
BX 00 02
CX 00 04
DX 00 0D
CS F400
IP 0204
SS 0710
SP 00FA
BP 0000
SI 0000
DI 0000
DS 0720
ES 0700
F400:0204
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ╧
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
...
screen
source
reset
aux
vars
debug
stack
flags
emulator screen (80x25 chars)
MULTIPLICATION: 50
DIVISION: 04
Downloads
Screenshot (8