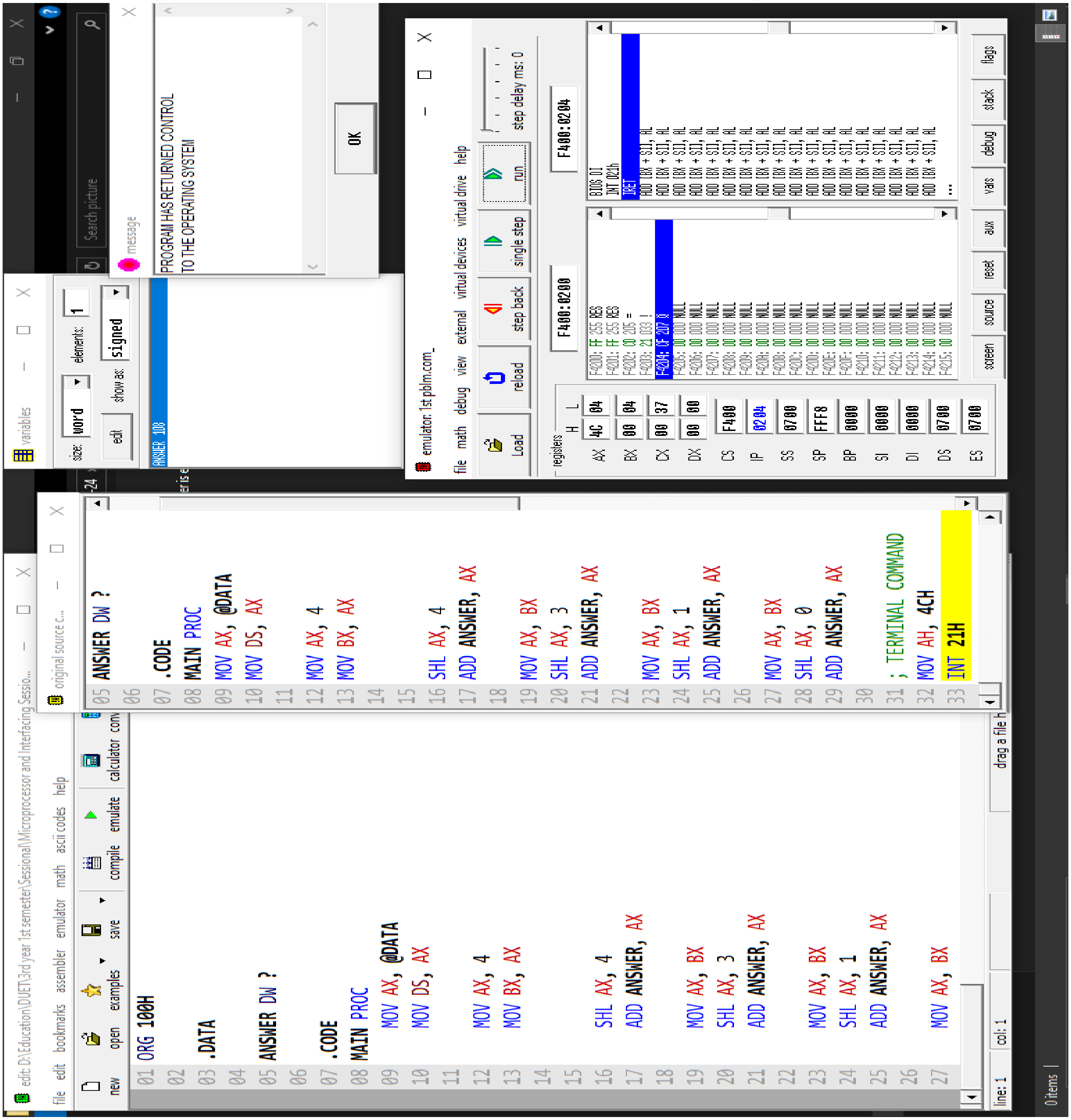
1. Write a program to multiply AX by 27 using only Shift and Add instructions. You should not use the MUL instruction.

> Recall that shifting left *n* bits multiplies the operand by 2^n.
>
> If the multiplier is not an absolute power of 2,
> then express the multiplier as a sum of terms which are absolute powers of 2.
>
> For example, multiply AX by 7. $(7 = 4 + 2 + 1 = 2^2 + 2^1 + 1)$
>
> Answer = AX shifted left by 2 + AX shifted left by 1 + AX.
>
> **Note:** Only the original value of AX is used in each operation above.



edit: D:\Education\DUET\3rd year 1st semester\Sessional\Microprocessor and Interfacing Sessio...
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator conv
line: 1 col: 1
drag a file h
original source c...
variables
message
emulator: 1st pblm.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
Search picture
0 items

```
ORG 100H

.DATA

ANSWER DW ?

.CODE
MAIN PROC
MOV AX, @DATA
MOV DS, AX

MOV AX, 4
MOV BX, AX


SHL AX, 4
ADD ANSWER, AX

MOV AX, BX
SHL AX, 3
ADD ANSWER, AX

MOV AX, BX
SHL AX, 1
ADD ANSWER, AX

MOV AX, BX
```

```
ANSWER DW ?

.CODE
MAIN PROC
MOV AX, @DATA
MOV DS, AX

MOV AX, 4
MOV BX, AX


SHL AX, 4
ADD ANSWER, AX

MOV AX, BX
SHL AX, 3
ADD ANSWER, AX

MOV AX, BX
SHL AX, 1
ADD ANSWER, AX

MOV AX, BX
SHL AX, 0
ADD ANSWER, AX

; TERMINAL COMMAND
MOV AH, 4CH
INT 21H
```

size: word elements: 1 show as: signed

ANSWER 108

PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM

OK

registers

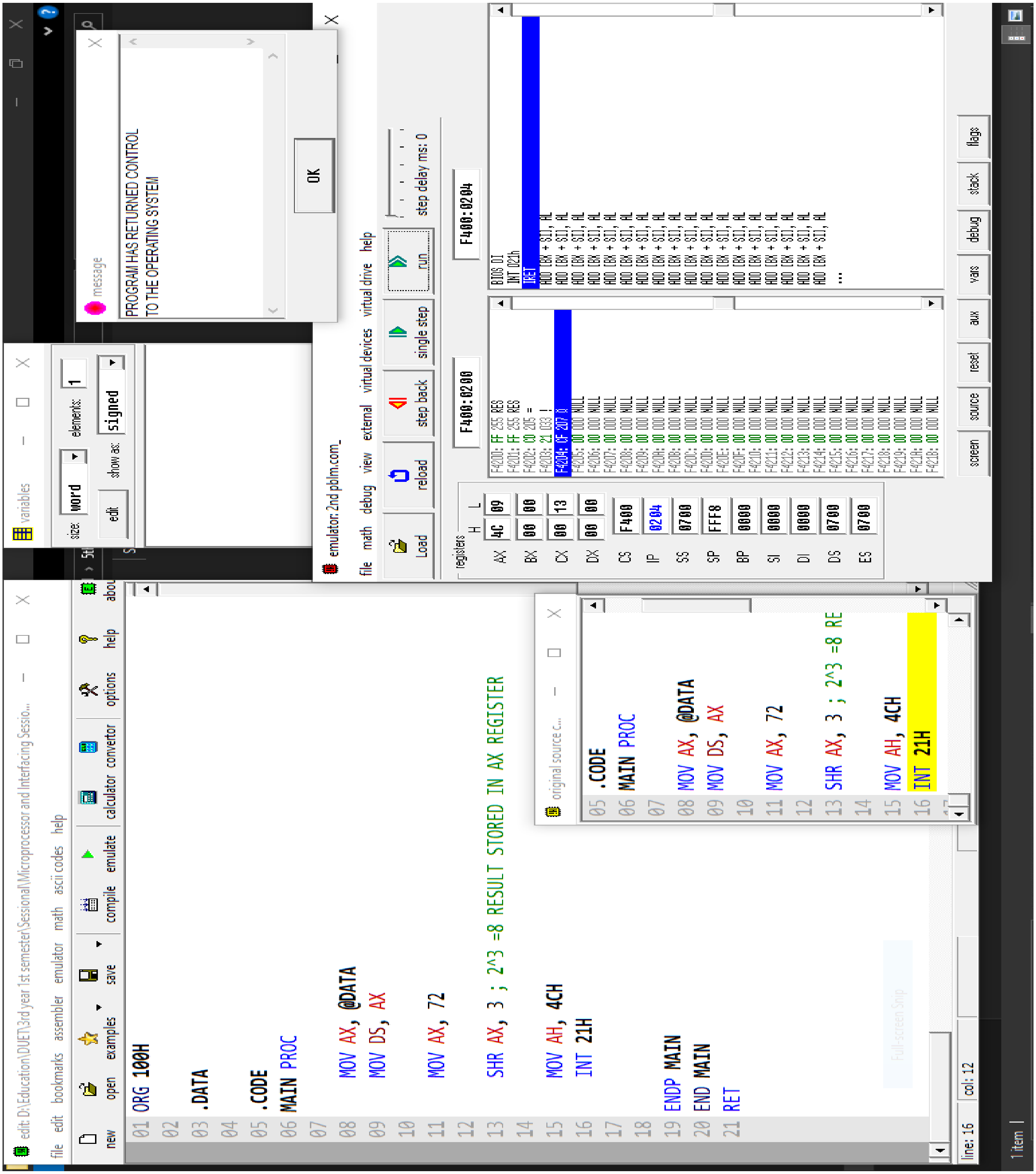
	H	L
AX	4C	04
BX	00	04
CX	00	37
DX	00	00
CS	F400	
IP	0204	
SS	0700	
SP	FFF8	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
F400:0200

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
```

```
F400:0204

BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

2. 2. Write a program to divide AX by 8 using Shift instructions. You should not use the DIV instruction. Assume AX is a multiple of 8.

Recall that shifting right *n* bits divides the operand by 2^n.



edit: D:\Education\DUET\3rd year 1st semester\Sessional\Microprocessor and Interfacing Sessio...
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

```
ORG 100H

 .DATA

 .CODE
MAIN PROC

 MOV AX, @DATA
 MOV DS, AX

 MOV AX, 72

 SHR AX, 3 ; 2^3 =8 RESULT STORED IN AX REGISTER

 MOV AH, 4CH
 INT 21H


ENDP MAIN
END MAIN
RET
```

line: 16 col: 12
variables
size: word
elements: 1
show as: signed
edit

message

PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM

OK
emulator: 2nd pblm.com_
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	09
BX	00	00
CX	00	13
DX	00	00
CS	F400	
IP	0204	
SS	0700	
SP	FFF8	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

F400:0200

```
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
F4216: 00 000 NULL
F4217: 00 000 NULL
F4218: 00 000 NULL
F4219: 00 000 NULL
F421A: 00 000 NULL
F421B: 00 000 NULL
```

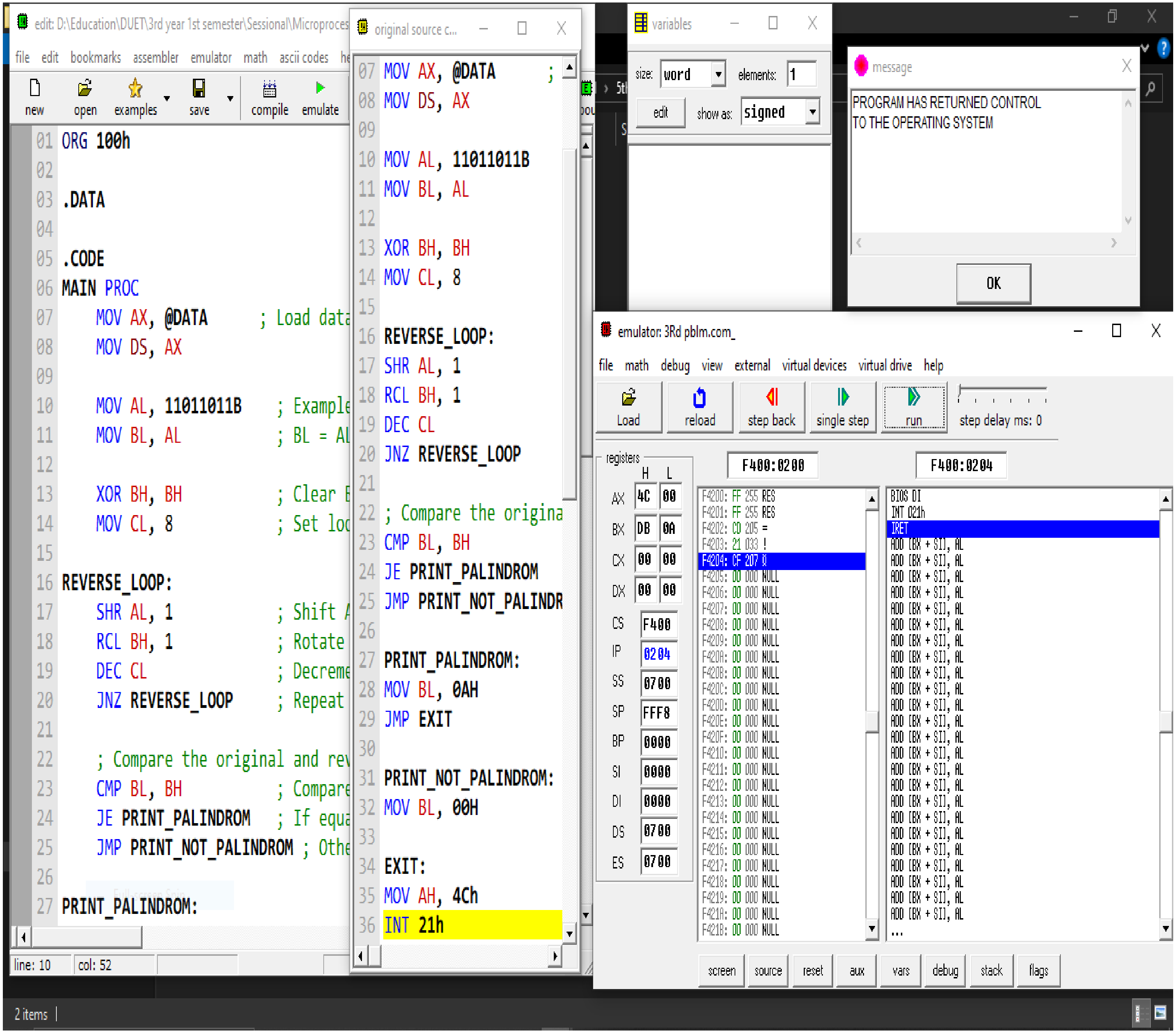
F400:0204

```
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

```
.CODE
MAIN PROC

MOV AX, @DATA
MOV DS, AX

MOV AX, 72

SHR AX, 3 ; 2^3 =8 RE

MOV AH, 4CH
INT 21H
```

3. Write a program to check if a byte is a Palindrome. [Hint: Use Rotate instructions]. If the byte is a Palindrome, then move AAh into BL. Otherwise move 00h in BL.

> A Palindrome looks the same when seen from the left or the right.
>
> For example, 11011011 is a Palindrome but 11010011 is not a Palindrome



edit: D:\Education\DUET\3rd year 1st semester\Sessional\Microproces
file edit bookmarks assembler emulator math ascii codes he
new open examples save compile emulate
line: 10 col: 52
2 items

```
ORG 100h

.DATA

.CODE
MAIN PROC
    MOV AX, @DATA      ; Load data
    MOV DS, AX

    MOV AL, 11011011B    ; Example
    MOV BL, AL           ; BL = AL

    XOR BH, BH           ; Clear B
    MOV CL, 8            ; Set loo

REVERSE_LOOP:
    SHR AL, 1            ; Shift A
    RCL BH, 1            ; Rotate
    DEC CL               ; Decreme
    JNZ REVERSE_LOOP     ; Repeat

    ; Compare the original and rev
    CMP BL, BH           ; Compare
    JE PRINT_PALINDROM   ; If equa
    JMP PRINT_NOT_PALINDROM ; Othe

PRINT_PALINDROM:
```

```
MOV AX, @DATA      ;
MOV DS, AX

MOV AL, 11011011B
MOV BL, AL

XOR BH, BH
MOV CL, 8

REVERSE_LOOP:
SHR AL, 1
RCL BH, 1
DEC CL
JNZ REVERSE_LOOP

; Compare the origina
CMP BL, BH
JE PRINT_PALINDROM
JMP PRINT_NOT_PALINDR

PRINT_PALINDROM:
MOV BL, 0AH
JMP EXIT

PRINT_NOT_PALINDROM:
MOV BL, 00H

EXIT:
MOV AH, 4Ch
INT 21h
```

size: word elements: 1
show as: signed

PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM

emulator: 3Rd pblm.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run
screen source reset aux vars debug stack flags

step delay ms: 0

registers

	H	L
AX	4C	00
BX	DB	0A
CX	00	00
DX	00	00
CS	F400	
IP	0204	
SS	0700	
SP	FFF8	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

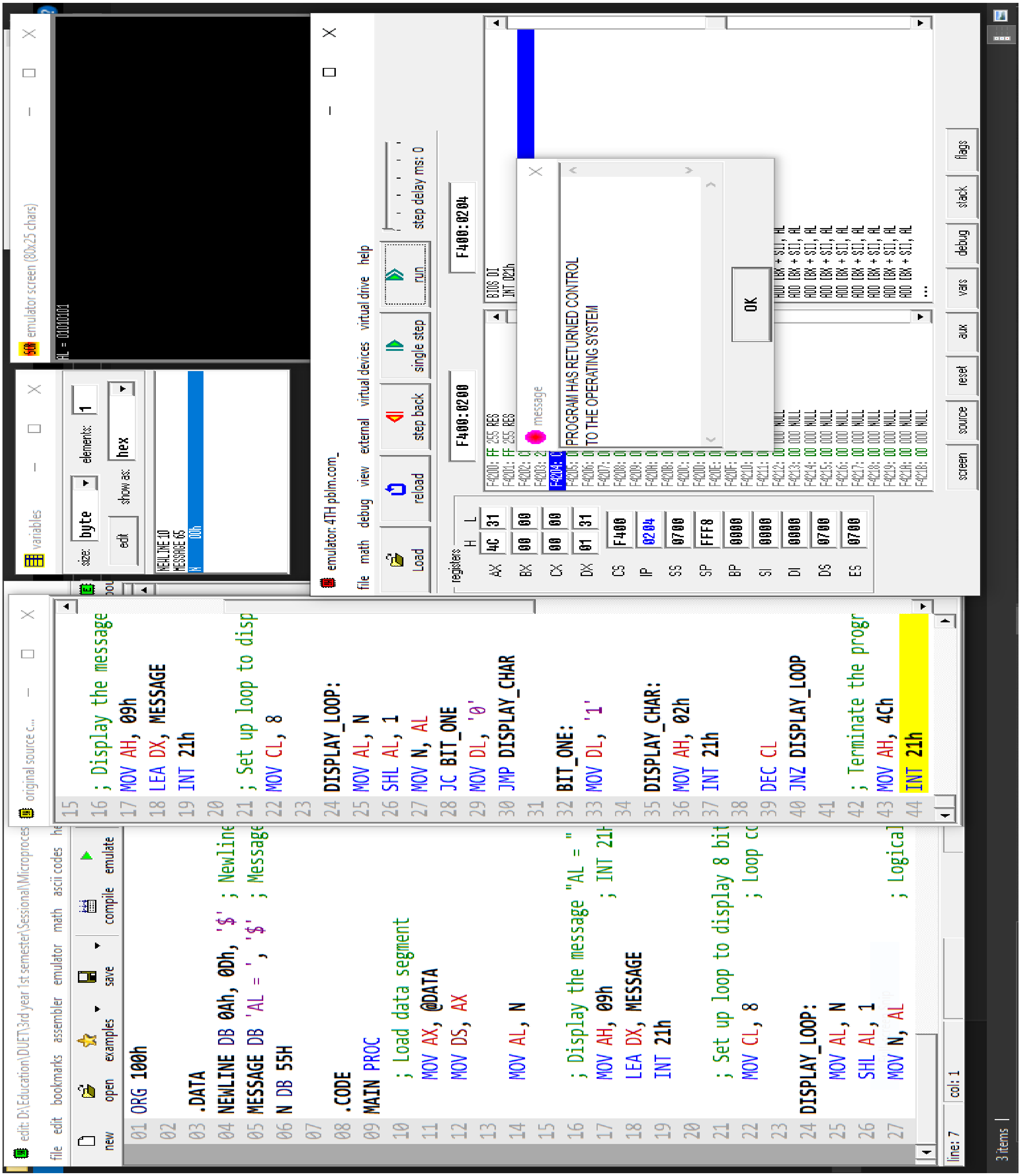
```
F400:0200
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ¤
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 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
F4216: 00 000 NULL
F4217: 00 000 NULL
F4218: 00 000 NULL
F4219: 00 000 NULL
F421A: 00 000 NULL
F421B: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

4. Write a program to display the bits of a register or memory location. Use the INT 21H interrupts to display data on the display monitor. [Hint: Use logical shift instruction to move data bit into the carry flag]

For example, if AL = 55H, then your program must display:

AL = 0 1 0 1 0 1 0 1


edit: D:\Education\DUET\3rd year 1st semester\Sessional\Microproces
file edit bookmarks assembler emulator math ascii codes he
new open examples save compile emulate

```
ORG 100h

.DATA
NEWLINE DB 0Ah, 0Dh, '$'  ; Newline
MESSAGE DB 'AL = ', '$'   ; Message
N DB 55H

.CODE
MAIN PROC
    ; Load data segment
    MOV AX, @DATA
    MOV DS, AX

    MOV AL, N

    ; Display the message "AL = "
    MOV AH, 09h             ; INT 21H
    LEA DX, MESSAGE
    INT 21h

    ; Set up loop to display 8 bit
    MOV CL, 8               ; Loop co

DISPLAY_LOOP:
    MOV AL, N
    SHL AL, 1
    MOV N, AL               ; Logical
```

line: 7
col: 1
3 items
original source c...

```
; Display the message
MOV AH, 09h
LEA DX, MESSAGE
INT 21h

; Set up loop to disp
MOV CL, 8

DISPLAY_LOOP:
MOV AL, N
SHL AL, 1
MOV N, AL
JC BIT_ONE
MOV DL, '0'
JMP DISPLAY_CHAR

BIT_ONE:
MOV DL, '1'

DISPLAY_CHAR:
MOV AH, 02h
INT 21h

DEC CL
JNZ DISPLAY_LOOP

; Terminate the progr
MOV AH, 4Ch
INT 21h
```

```
NEWLINE 10
MESSAGE 65
N       00h
```

```
AL = 01010101
```

emulator: 4TH pblm.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
F400:0200
F400:0204

	H	L
AX	4C	31
BX	00	00
CX	00	00
DX	01	31
CS	F400	
IP	0204	
SS	0700	
SP	FFF8	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

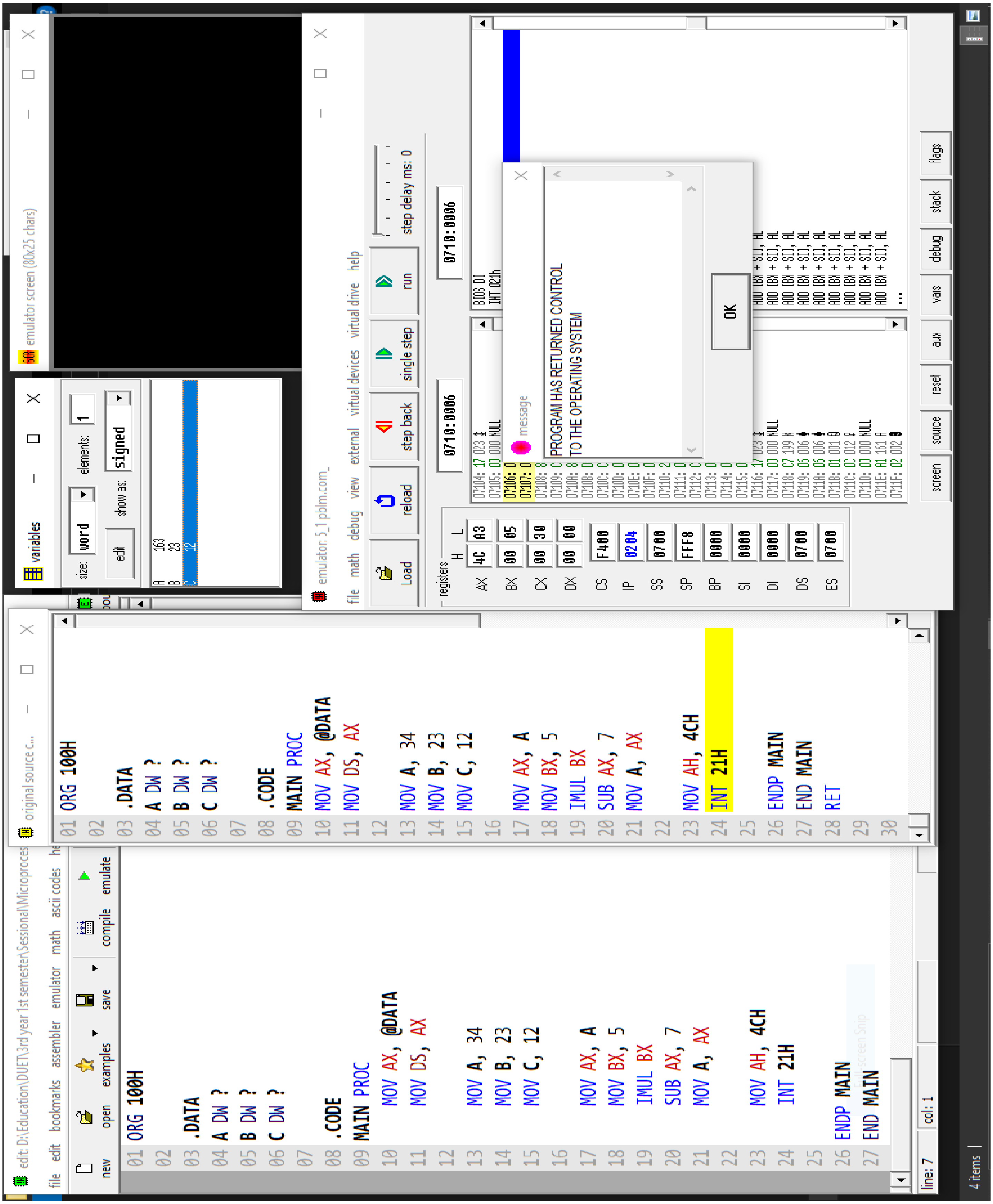
```
F4200: FF 255 RES
F4201: FF 255 RES
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
F4216: 00 000 NULL
F4217: 00 000 NULL
F4218: 00 000 NULL
F4219: 00 000 NULL
F421A: 00 000 NULL
F421B: 00 000 NULL
```

```
BIOS DI
INT 021h
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM

5. Write assembly code for each of the following high-level language assignment statements. Suppose that A, B, and C are word variables and all products will fit in 16 bits. Use IMUL for multiplication. It's not necessary to preserve the contents of variables A, B, and C.

   a. A = 5 x A – 7



```
ORG 100H

.DATA
A DW ?
B DW ?
C DW ?

.CODE
MAIN PROC
MOV AX, @DATA
MOV DS, AX

MOV A, 34
MOV B, 23
MOV C, 12

MOV AX, A
MOV BX, 5
IMUL BX
SUB AX, 7
MOV A, AX

MOV AH, 4CH
INT 21H

ENDP MAIN
END MAIN
```

```
ORG 100H

.DATA
A DW ?
B DW ?
C DW ?

.CODE
MAIN PROC
MOV AX, @DATA
MOV DS, AX

MOV A, 34
MOV B, 23
MOV C, 12

MOV AX, A
MOV BX, 5
IMUL BX
SUB AX, 7
MOV A, AX

MOV AH, 4CH
INT 21H

ENDP MAIN
END MAIN
RET
```

```
A   163
B   23
C   12
```

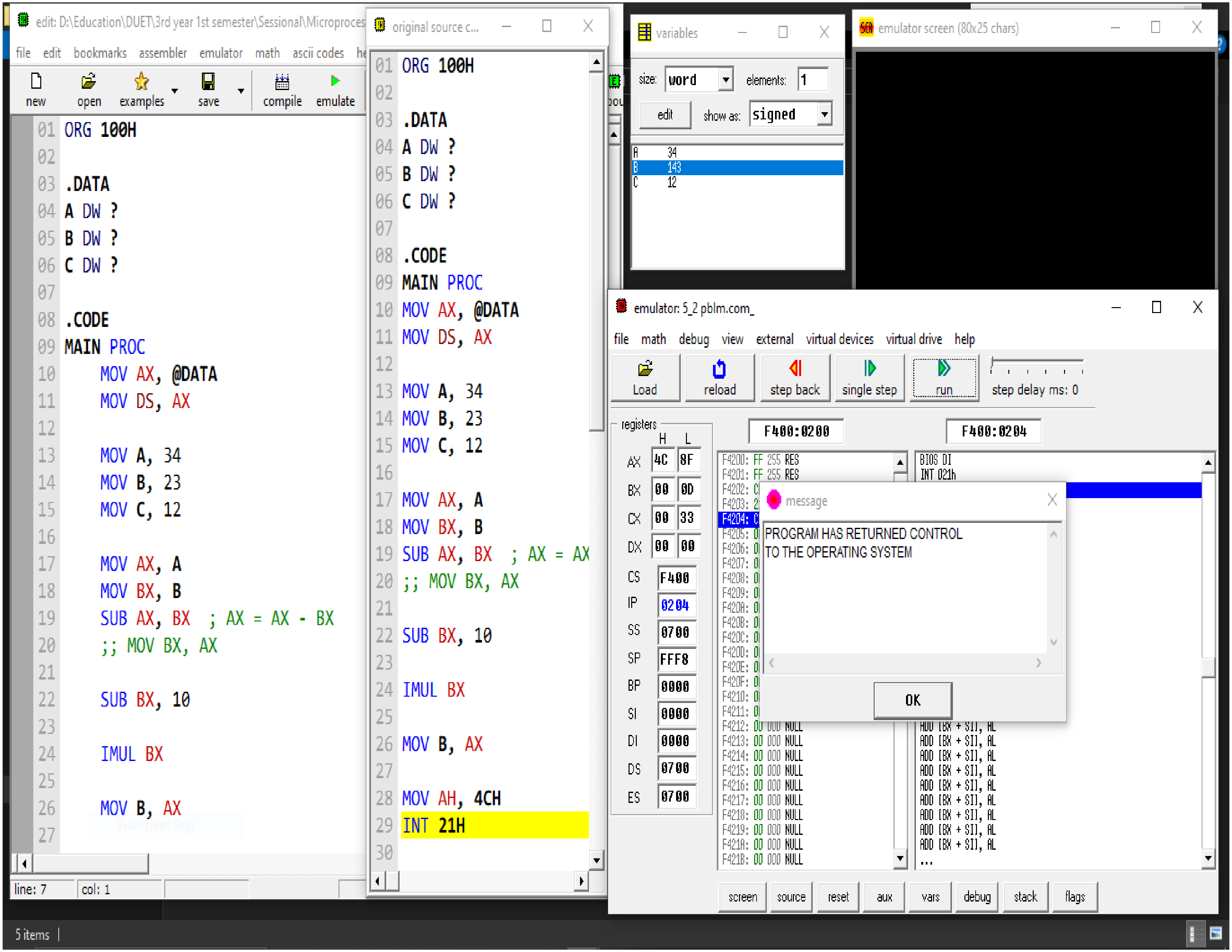
```
registers
      H  L
AX    4C A3
BX    00 05
CX    00 30
DX    00 00
CS    F400
IP    0204
SS    0700
SP    FFF8
BP    0000
SI    0000
DI    0000
DS    0700
ES    0700
```

```
0710:0006
```

```
PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM
```

edit: D:\Education\DUET\3rd year 1st semester\Sessional\Microproces
file edit bookmarks assembler emulator math ascii codes he
new open examples save compile emulate
line: 7 col: 1
original source c...
variables
size: word elements: 1
edit show as: signed
emulator screen (80x25 chars)
emulator: 5.1 pblm.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
message
OK
screen source reset aux vars debug stack flags
4 items

b. B = (A-B) x (B-10



edit: D:\Education\DUET\3rd year 1st semester\Sessional\Microproces
file edit bookmarks assembler emulator math ascii codes he
new open examples save compile emulate
original source c...
variables
emulator screen (80x25 chars)
emulator: 5_2 pblm.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run
step delay ms: 0
screen source reset aux vars debug stack flags
line: 7 col: 1
5 items

```
ORG 100H

.DATA
A DW ?
B DW ?
C DW ?

.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    MOV A, 34
    MOV B, 23
    MOV C, 12

    MOV AX, A
    MOV BX, B
    SUB AX, BX  ; AX = AX - BX
    ;; MOV BX, AX

    SUB BX, 10

    IMUL BX

    MOV B, AX
```

```
ORG 100H

.DATA
A DW ?
B DW ?
C DW ?

.CODE
MAIN PROC
MOV AX, @DATA
MOV DS, AX

MOV A, 34
MOV B, 23
MOV C, 12

MOV AX, A
MOV BX, B
SUB AX, BX  ; AX = AX
;; MOV BX, AX

SUB BX, 10

IMUL BX

MOV B, AX

MOV AH, 4CH
INT 21H

```

size: word elements: 1
show as: signed

A	34
B	143
C	12

registers

	H	L
AX	4C	8F
BX	00	0D
CX	00	33
DX	00	00
CS	F400	
IP	0204	
SS	0700	
SP	FFF8	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```
F400:0200
F4200: FF 255 RES
F4201: FF 255 RES
F4202: C
F4203: 2
F4204: C
F4205: 0
F4206: 0
F4207: 0
F4208: 0
F4209: 0
F420A: 0
F420B: 0
F420C: 0
F420D: 0
F420E: 0
F420F: 0
F4210: 0
F4211: 0
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
F4216: 00 000 NULL
F4217: 00 000 NULL
F4218: 00 000 NULL
F4219: 00 000 NULL
F421A: 00 000 NULL
F421B: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

message

PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM

OK