



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

CSE2006

Microprocessor and Interfacing

Lab Assignment 1

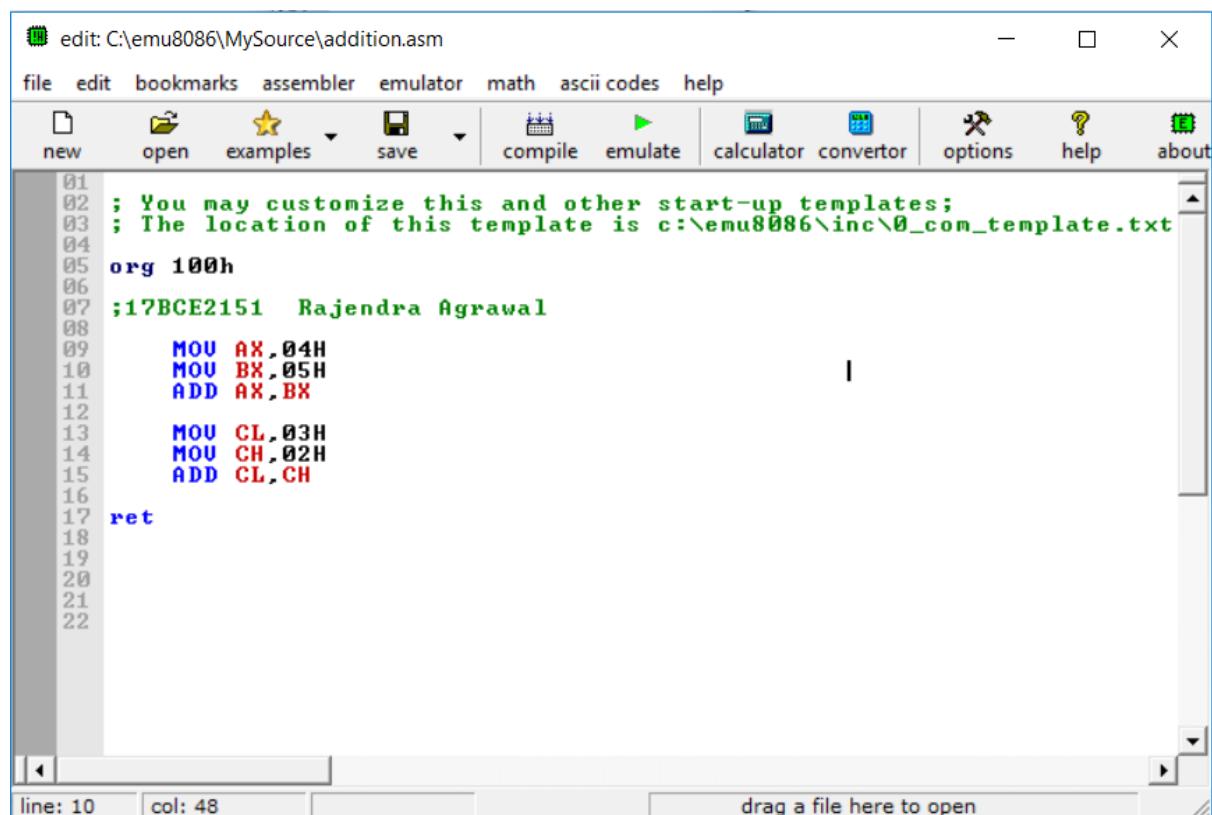
Slot: L39+L40

Name: Rajendra Agrawal
Reg. No: 17BCE2151

Question: Write an 8086 Assembly Language Program to perform basic Arithmetic operations in both 8bit and 16bit mode of computation.

1. Addition:

Code Snippet:



The screenshot shows the emu8086 assembly editor interface. The title bar reads "edit: C:\emu8086\MySource\addition.asm". The menu bar includes file, edit, bookmarks, assembler, emulator, math, ascii codes, and help. The toolbar contains icons for new, open, examples, save, compile, emulate, calculator, convertor, options, help, and about. The main code window displays the following assembly code:

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agrawal
07
08 MOU AX,04H
09 MOU BX,05H
10 ADD AX,BX
11
12 MOU CL,03H
13 MOU CH,02H
14 ADD CL,CH
15
16 ret
17
18
19
20
21
22
```

The status bar at the bottom shows "line: 10 col: 48" and "drag a file here to open".

Step by Step Execution:

Step 1: Load the program.

The screenshot shows a debugger interface with two main panes. The left pane displays the assembly code and registers. The assembly code window shows the following instructions:

```
01 ; You may customize this
02 ; The location of this t
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agr
07
08 MOU AX, 00004h
09 MOU BX, 00005h
10 ADD AX, BX
11 ADD AX,BX
12
13 MOU CL, 03h
14 MOU CH, 02h
15 ADD CL, CH
16
17 ret
18
19
20
21
22
23
24
```

The registers window shows the following values:

	H	L
AX	00	00
BX	00	00
CX	00	0F
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The right pane shows the original source code:

```
; You may customize this
; The location of this t
org 100h
;17BCE2151 Rajendra Agr
MOU AX, 00004h
MOU BX, 00005h
ADD AX, BX
ADD AX,BX
MOU CL, 03h
MOU CH, 02h
ADD CL, CH
ret
```

Step 2: Moving 04H into AX for word mode operation.

The screenshot shows the debugger interface again. The registers pane now shows AX = 00 04. The assembly code pane shows the same instructions as before, but the first instruction has been modified:

```
01 ; You may customize this
02 ; The location of this t
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agr
07
08 MOU AX, 04H
09 MOU BX, 05H
10 ADD AX, BX
11 ADD AX,BX
12
13 MOU CL, 03h
14 MOU CH, 02h
15 ADD CL, CH
16
17 ret
18
19
20
21
22
23
24
```

Step 3: Moving 05H into BX.

The screenshot shows a debugger interface with two main windows. The left window displays the assembly code and registers. The right window shows the original source code. The assembly code window has the following details:

Register	H	L
AX	00	04
BX	00	05
CX	00	0F
DX	00	00
CS	0700	
IP	0106	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The assembly code pane shows the following instructions:

```
01: MOU AX, 00004h
02: MOU BX, 00005h
03: ADD AX, BX
04: MOU CL, 03h
05: MOU CH, 02h
06: ADD CL, CH
07: RET
08: NOP
09: NOP
10: NOP
11: ADD AX, BX
12: MOU CL, 03H
13: MOU CH, 02H
14: ADD CL, CH
15: ret
16:
17:
18:
19:
20:
21:
22:
23:
24:
```

Step 4: Adding AX and BX and store result in AX.

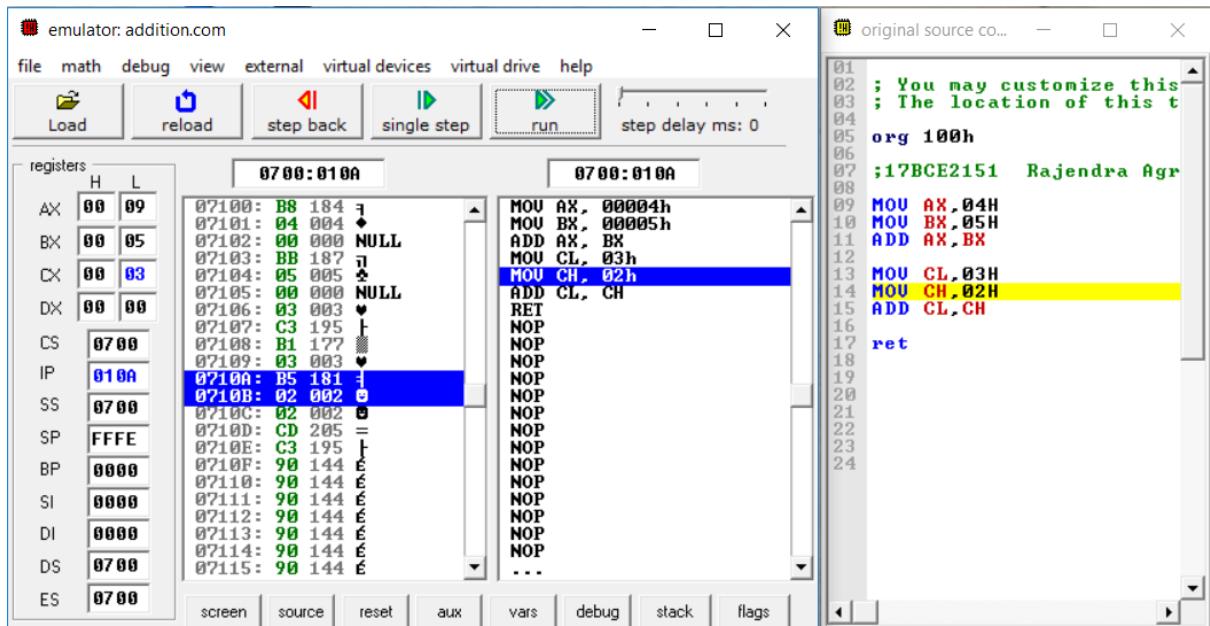
The screenshot shows a debugger interface with two main windows. The left window displays the assembly code and registers. The right window shows the original source code. The assembly code window has the following details:

Register	H	L
AX	00	09
BX	00	05
CX	00	0F
DX	00	00
CS	0700	
IP	0108	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

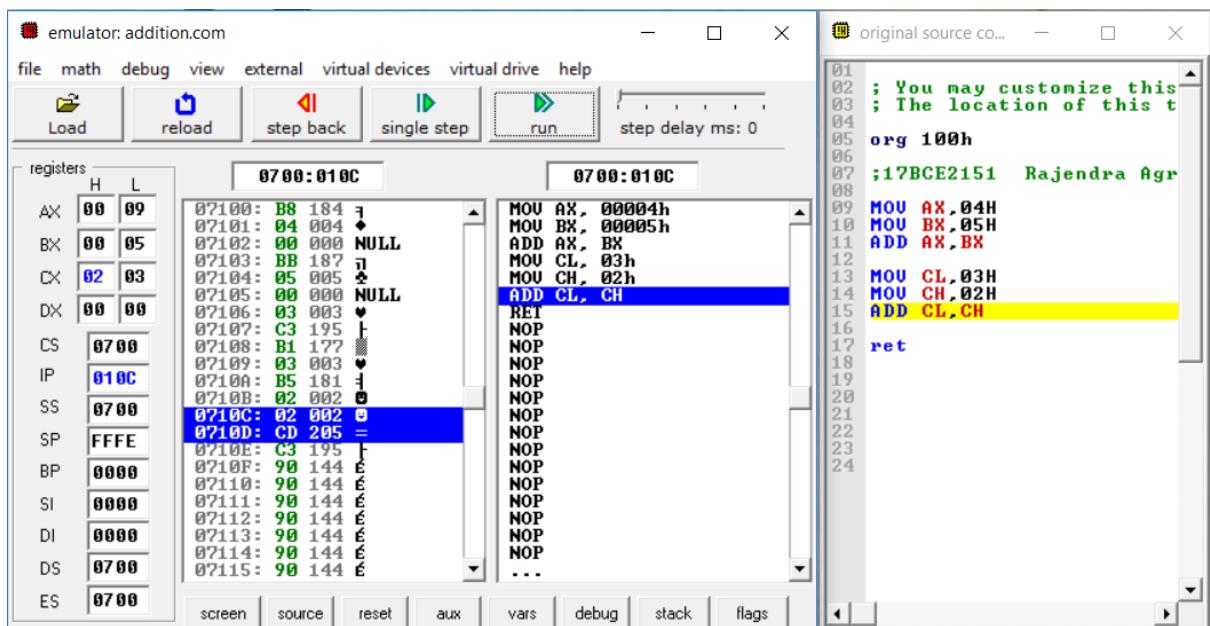
The assembly code pane shows the following instructions:

```
01: MOU AX, 00004h
02: MOU BX, 00005h
03: ADD AX, BX
04: MOU CL, 03h
05: MOU CH, 02h
06: ADD CL, CH
07: RET
08: NOP
09: NOP
10: NOP
11: ADD AX, BX
12: MOU CL, 03H
13: MOU CH, 02H
14: ADD CL, CH
15: ret
16:
17:
18:
19:
20:
21:
22:
23:
24:
```

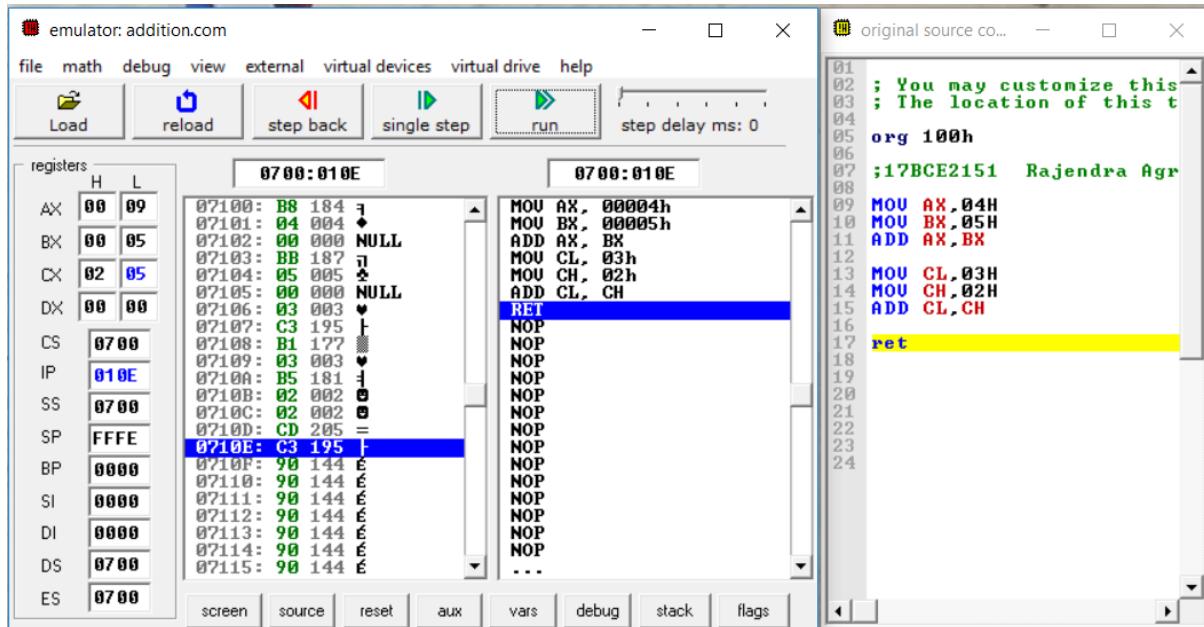
Step 5: Moving 03H into CL for byte mode operation.



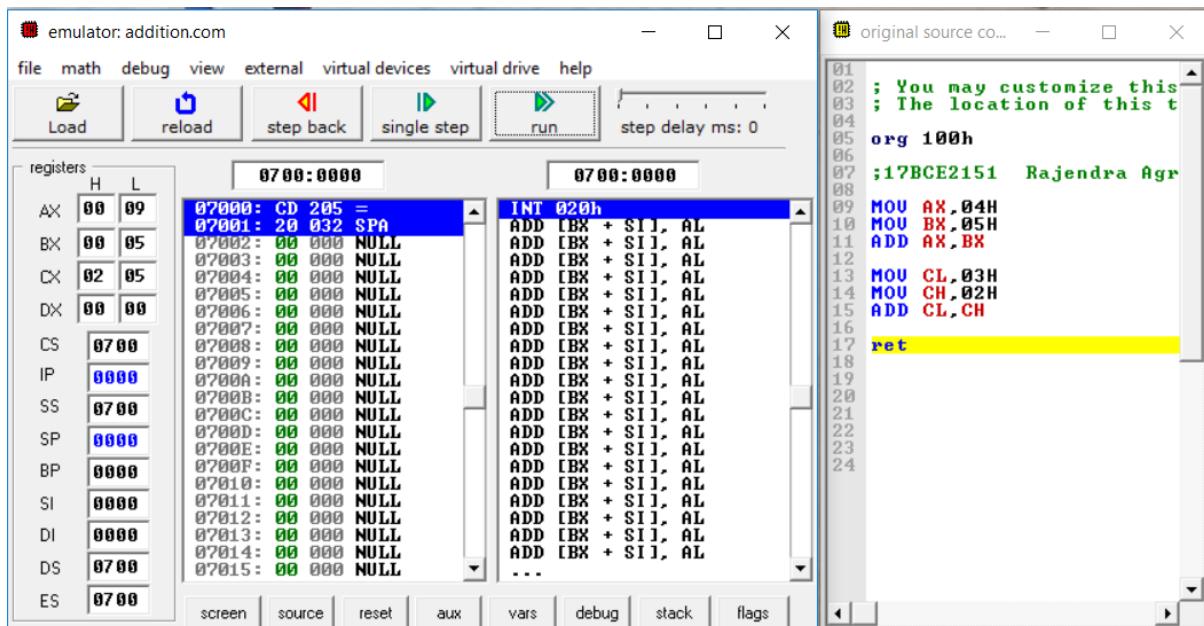
Step 6: Moving 02H into CH.



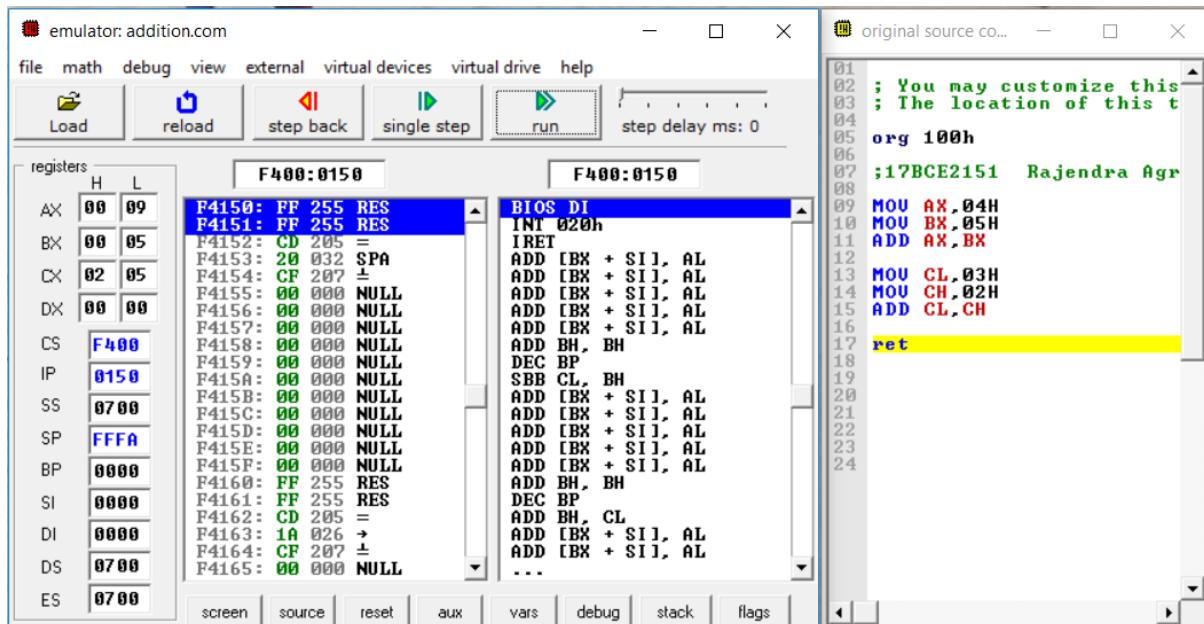
Step 7: Add CL and CH, and store result into CL.



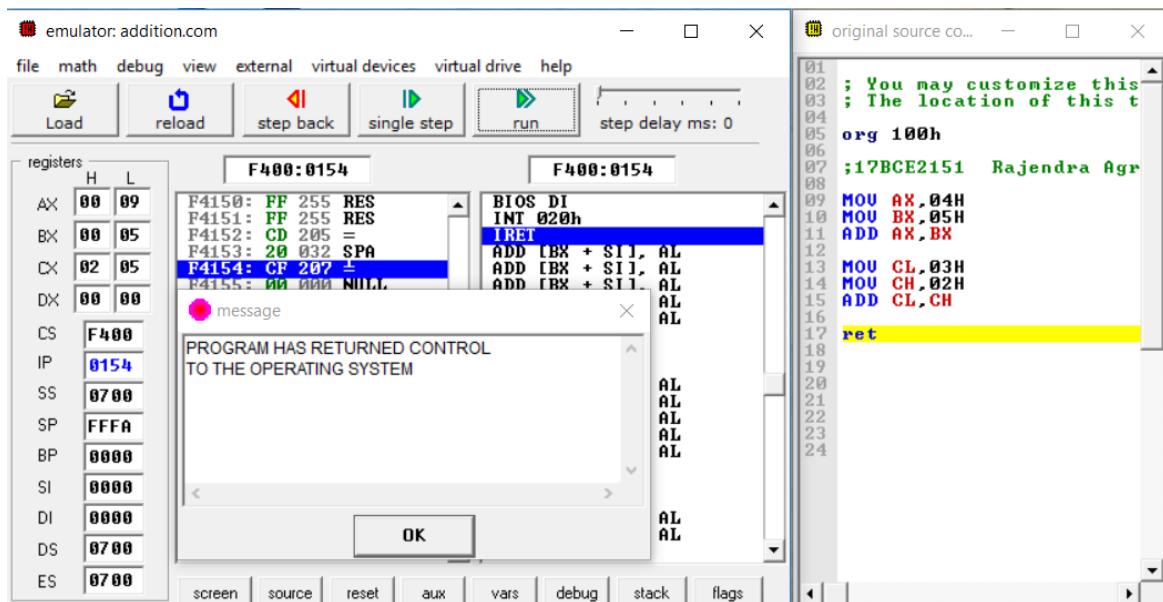
Step 8: Executing return.



Step 9: Executing Return.



Step 10: Execution complete.



2. Subtraction:

Code Snippet:

The screenshot shows the emu8086 assembler interface with the file 'subtraction.asm' open. The code is as follows:

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agrawal
07
08 MOU AX,09H
09 MOU BX,03H
10 SUB AX,BX
11
12 MOU CL,03H
13 MOU CH,02H
14 SUB CL,CH
15
16 ret
17
18
19
20
21
22
```

The code performs a subtraction operation where AX is subtracted from BX, and the result is stored in AX. It then performs another subtraction operation where CL is subtracted from CH, and the result is stored in CL.

Step by Step Execution:

Step 1: Load the program.

The screenshot shows the emu8086 emulator interface with the file 'subtraction.com_' loaded. The registers window shows the initial values of the registers. The assembly window shows the assembly code being executed step-by-step. The code is identical to the one in the previous screenshot.

Register	H	L
AX	00	00
BX	00	00
CX	00	0F
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

Registers window:

	0700:0100	0700:0100
07100: B8 184	MOU AX, 00009h	
07101: 09 009 TAB	MOU BX, 00003h	
07102: 00 000 NULL	SUB AX, BX	
07103: BB 187	MOU CL, 03h	
07104: 03 003	MOU CH, 02h	
07105: 00 000 NULL	SUB CL, CH	
07106: 2B 043	RET	
07107: C3 195	NOP	
07108: B1 177	NOP	
07109: 03 003	NOP	
0710A: B5 181	NOP	
0710B: 02 002	NOP	
0710C: 2A 042	NOP	
0710D: CD 205	NOP	
0710E: C3 195	NOP	
0710F: 90 144	NOP	
07110: 90 144	NOP	
07111: 90 144	NOP	
07112: 90 144	NOP	
07113: 90 144	NOP	
07114: 90 144	NOP	
07115: 90 144	NOP	

Assembly window:

```
01 ; You may customize this
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agrawal
07
08 MOU AX,09H
09 MOU BX,03H
10 SUB AX,BX
11
12 MOU CL,03H
13 MOU CH,02H
14 SUB CL,CH
15
16 ret
17
18
19
20
21
22
```

Step 2: Move 09H into AX for word mode operation.

The screenshot shows a debugger interface with two windows. The left window displays the assembly code and registers for memory location 0700:0103. The right window shows the original source code. The assembly code at address 0700:0103 is:

```
07100: B8 184 TAB
07101: 09 009 NULL
07102: 00 000 NULL
07103: BB 187 ni
07104: 03 003 ▼
07105: 00 000 NULL
```

The register values are:

Registers	H	L
AX	00	09
BX	00	00
CX	00	0F
DX	00	00
CS	0700	
IP	0103	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The right window shows the original source code:

```
; You may customize this
; The location of this t
01
02 org 100h
03 ;17BCE2151 Rajendra Agr
04
05 MOU AX, 00009h
06 MOU BX, 00003h
07 SUB AX, BX
08
09 MOU AX, 09H
10 MOU BX, 03H
11 SUB AX, BX
12
13 MOU CL, 03h
14 MOU CH, 02h
15 SUB CL, CH
16
17 ret
18
19
20
21
22
23
24
```

Step 3: Move 03H into BX.

The screenshot shows a debugger interface with two windows. The left window displays the assembly code and registers for memory location 0700:0106. The right window shows the original source code. The assembly code at address 0700:0106 is:

```
07100: B8 184 TAB
07101: 09 009 NULL
07102: 00 000 NULL
07103: BB 187 ni
07104: 03 003 ▼
07105: 00 000 NULL
07106: 2B 043 +
07107: C3 195 ↴
```

The register values are:

Registers	H	L
AX	00	09
BX	00	03
CX	00	0F
DX	00	00
CS	0700	
IP	0106	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The right window shows the original source code:

```
; You may customize this
; The location of this t
01
02 org 100h
03 ;17BCE2151 Rajendra Agr
04
05 MOU AX, 00009h
06 MOU BX, 00003h
07 SUB AX, BX
08
09 MOU AX, 09H
10 MOU BX, 03H
11 SUB AX, BX
12
13 MOU CL, 03h
14 MOU CH, 02h
15 SUB CL, CH
16
17 ret
18
19
20
21
22
23
24
```

Step 4: Subtract AX and BX and store result in AX.

The screenshot shows a debugger interface with two main windows. The left window displays the assembly code and registers. The assembly code window shows the following instructions:

```
07100: B8 184 F TAB
07101: 09 009 TAB
07102: 00 000 NULL
07103: BB 187 I
07104: 03 003 V
07105: 00 000 NULL
07106: 2B 043 +
07107: C3 195 J
07108: B1 177 M
07109: 03 003 V
```

The registers window shows the following values:

	H	L
AX	00	06
BX	00	03
CX	00	0F
DX	00	00
CS	0700	
IP	0108	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The right window shows the original source code:

```
; You may customize this
; The location of this t
01
02 org 100h
03 ;17BCE2151 Rajendra Agr
04
05 MOU AX, 09H
06 MOU BX, 03H
07 SUB AX, BX
08
09 MOU CL, 03H
10 MOU CH, 02H
11 SUB CL, CH
12
13 ret
```

Step 5: Move 03H into CL for byte mode operation.

The screenshot shows a debugger interface with two main windows. The left window displays the assembly code and registers. The assembly code window shows the following instructions:

```
07100: B8 184 F TAB
07101: 09 009 TAB
07102: 00 000 NULL
07103: BB 187 I
07104: 03 003 V
07105: 00 000 NULL
07106: 2B 043 +
07107: C3 195 J
07108: B1 177 M
07109: 03 003 V
0710A: B5 181 F
0710B: 02 002 O
```

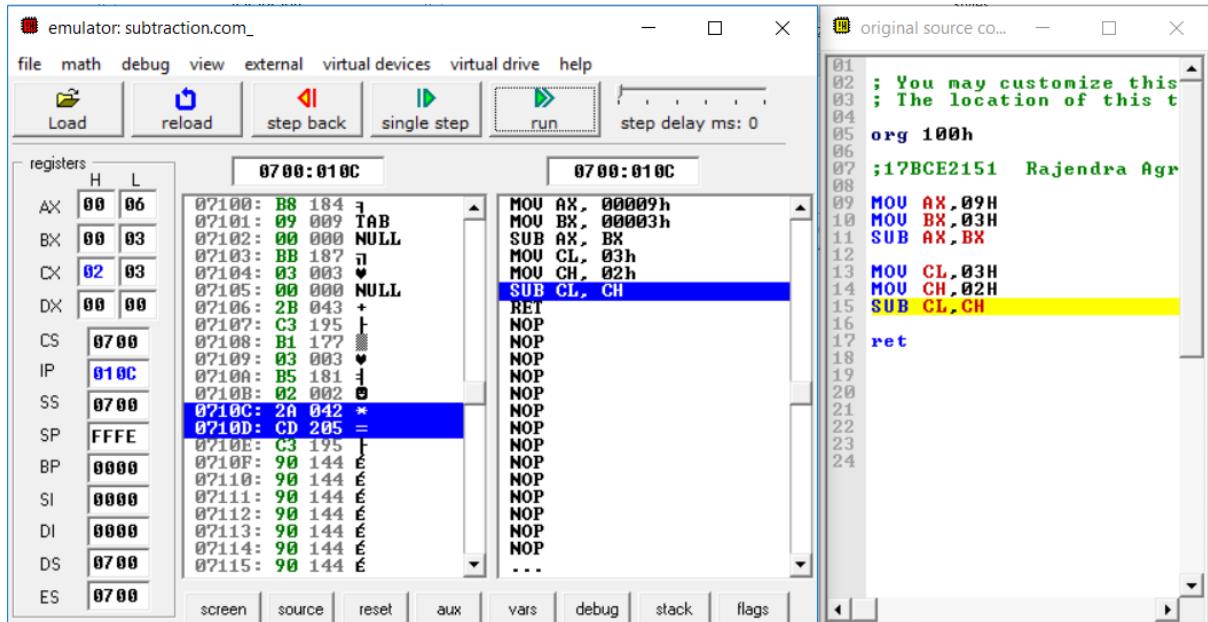
The registers window shows the following values:

	H	L
AX	00	06
BX	00	03
CX	00	03
DX	00	00
CS	0700	
IP	010A	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

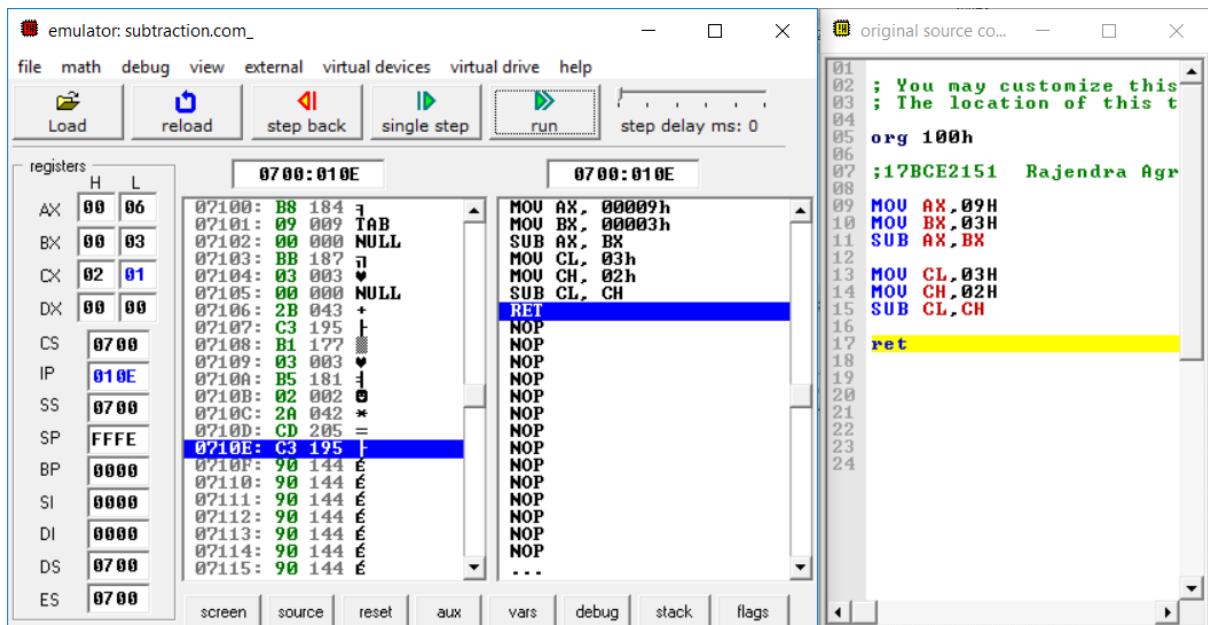
The right window shows the original source code:

```
; You may customize this
; The location of this t
01
02 org 100h
03 ;17BCE2151 Rajendra Agr
04
05 MOU AX, 09H
06 MOU BX, 03H
07 SUB AX, BX
08
09 MOU CL, 03H
10 MOU CH, 02H
11 SUB CL, CH
12
13 ret
```

Step 6: Move 02H into CH.



Step 7: Subtract CL and CH and store result into CL.



Step 8: Executing Return.

The screenshot shows the emulator interface with two windows. The left window displays the assembly code and registers. The right window shows the source code. The assembly code window has two panes: the left pane shows memory starting at address 07000, and the right pane shows memory starting at address 07000. The source code window shows assembly language code with line numbers 01 to 24. The instruction at line 17 is highlighted in yellow: `ret`.

```

; You may customize this
; The location of this t
01
02
03
04
05
06
07 ;17BCE2151 Rajendra Agr
08
09 MOU AX,09H
10 MOU BX,03H
11 SUB AX,BX
12
13 MOU CL,03H
14 MOU CH,02H
15 SUB CL,CH
16
17 ret
18
19
20
21
22
23
24

```

Step 9: Executing Return.

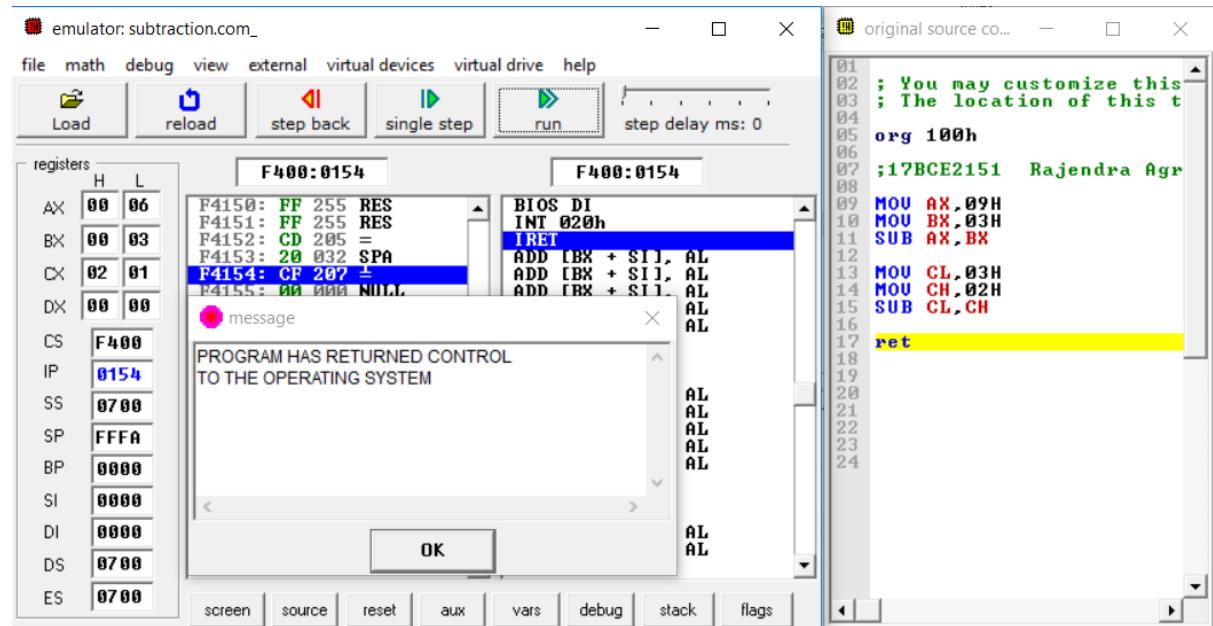
The screenshot shows the emulator interface with two windows. The left window displays the assembly code and registers. The right window shows the source code. The assembly code window has two panes: the left pane shows memory starting at address F4000, and the right pane shows memory starting at address F4000. The source code window shows assembly language code with line numbers 01 to 24. The instruction at line 17 is highlighted in yellow: `ret`.

```

; You may customize this
; The location of this t
01
02
03
04
05
06
07 ;17BCE2151 Rajendra Agr
08
09 MOU AX,09H
10 MOU BX,03H
11 SUB AX,BX
12
13 MOU CL,03H
14 MOU CH,02H
15 SUB CL,CH
16
17 ret
18
19
20
21
22
23
24

```

Step 10: Execution Complete.



3. Multiplication:

Code Snippet:

The screenshot shows the emu8086 assembler editor interface. The title bar reads "edit: C:\emu8086\MySource\multiplication.asm". The menu bar includes file, edit, bookmarks, assembler, emulator, math, ascii codes, and help. The toolbar has icons for new, open, examples, save, compile, emulate, calculator, converter, options, help, and about. The main window displays the following assembly code:

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agrawal
07
08 MOU AX,09H
09 MOU BX,03H
10 MUL BX
11
12 MOU AL,03H
13 MOU AH,02H
14 MUL AH
15
16 ret
17
18
19
20
21
22
```

The status bar at the bottom shows "line: 16 col: 5" and "drag a file here to open".

Step by Step Execution:

Step 1: Load the program.

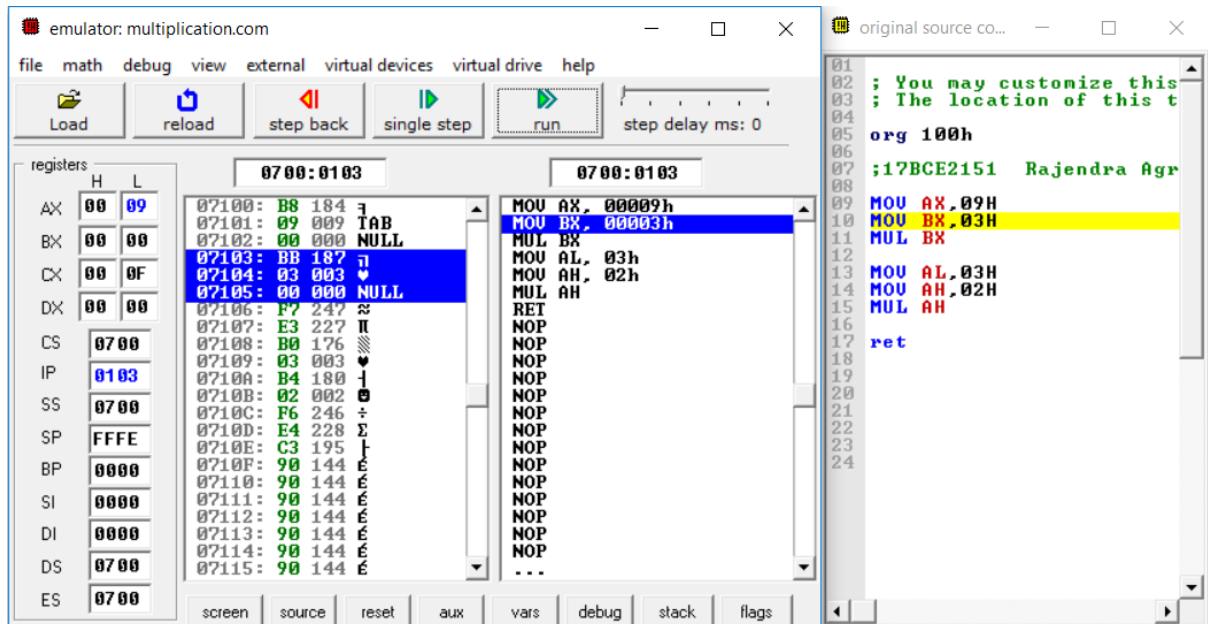
The screenshot shows the emu8086 emulator interface. The title bar reads "emulator: multiplication.com". The menu bar includes file, math, debug, view, external, virtual devices, virtual drive, and help. The toolbar has icons for load, reload, step back, single step, run, and step delay ms: 0. The registers window shows the following values:

	H	L
AX	00	00
BX	00	00
CX	00	0F
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

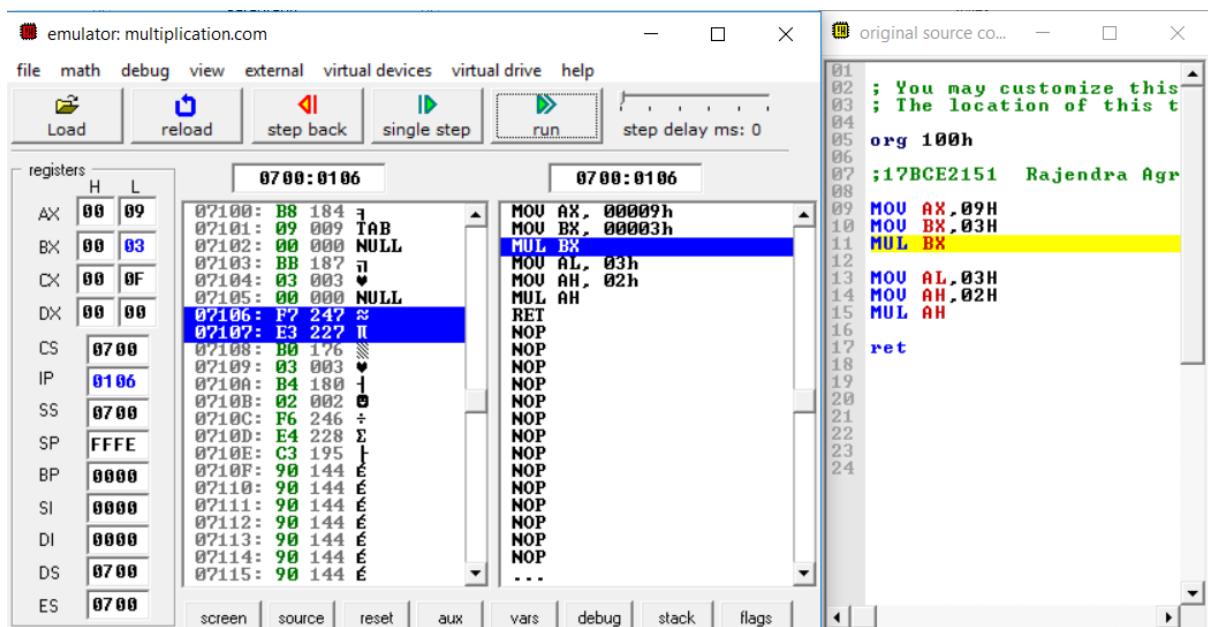
The memory dump window shows the first 15 bytes of memory starting at address 0700:0100. The code window on the right shows the assembly code:

```
01 ; You may customize this
02 ; The location of this t
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agr
07
08 MOU AX,09H
09 MOU BX,03H
10 MUL BX
11
12 MOU AL,03H
13 MOU AH,02H
14 MUL AH
15
16 ret
17
18
19
20
21
22
23
24
```

Step 2: Move 09H into AX for word mode operation.



Step 3: Move 03H into BX.



Step 4: Multiply BX with AX and store result into AX.

The screenshot shows a debugger interface with two panes. The left pane displays assembly code and registers. The right pane shows the original source code. The assembly code pane has two columns: addresses 0700:0108 and 0700:0109. The source code pane shows the assembly instructions corresponding to the code in the assembly pane.

Registers	H	L
AX	00	1B
BX	00	03
CX	00	0F
DX	00	00
CS	0700	
IP	0108	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

Assembly code (0700:0108):

```
07100: B8 184 TAB
07101: 09 009 NULL
07102: 00 000 NULL
07103: BB 187 I
07104: 03 003 V
07105: 00 000 NULL
07106: F7 247 Z
07107: E3 227 R
07108: B0 176 H
07109: 03 003 V
```

Assembly code (0700:0109):

```
MOU AX, 00009h
MOU BX, 00003h
MUL BX
MOU AL, 03h
MOU AH, 02h
MUL AH
RET
```

Source code:

```
; You may customize this
; The location of this t
org 100h
;17BCE2151 Rajendra Agr
MOU AX,09H
MOU BX,03H
MUL BX
MOU AL,03H
MOU AH,02H
MUL AH
ret
```

Step 5: Move 03H into AL for byte mode operation.

The screenshot shows a debugger interface with two panes. The left pane displays assembly code and registers. The right pane shows the original source code. The assembly code pane has two columns: addresses 0700:010A and 0700:010B. The source code pane shows the assembly instructions corresponding to the code in the assembly pane.

Registers	H	L
AX	00	03
BX	00	03
CX	00	0F
DX	00	00
CS	0700	
IP	010A	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

Assembly code (0700:010A):

```
07100: B8 184 TAB
07101: 09 009 NULL
07102: 00 000 NULL
07103: BB 187 I
07104: 03 003 V
07105: 00 000 NULL
07106: F7 247 Z
07107: E3 227 R
07108: B0 176 H
07109: 03 003 V
0710A: B4 180 I
```

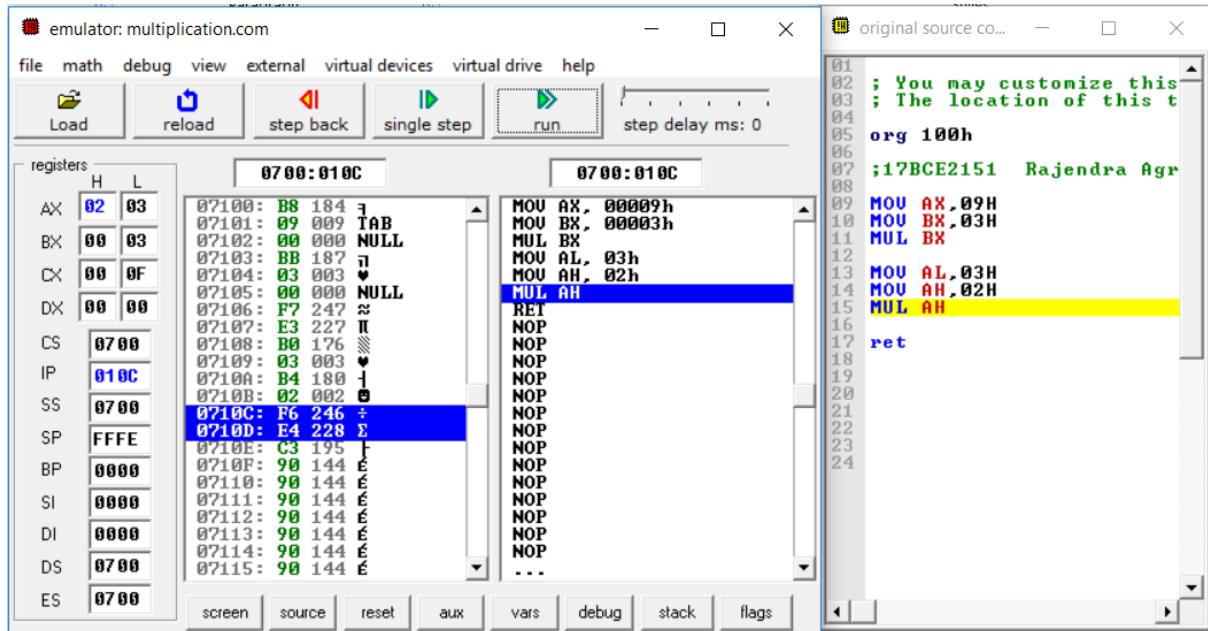
Assembly code (0700:010B):

```
MOU AX, 00009h
MOU BX, 00003h
MUL BX
MOU AL, 03h
MUL AH
RET
```

Source code:

```
; You may customize this
; The location of this t
org 100h
;17BCE2151 Rajendra Agr
MOU AX,09H
MOU BX,03H
MUL BX
MOU AL,03H
MOU AH,02H
MUL AH
ret
```

Step 6: Move 02H into AH.



The screenshot shows a debugger interface with two main panes. The left pane displays the assembly code and registers. The right pane shows the original source code. The assembly code pane has two columns: addresses 0700:010C and 0700:010E. The registers pane shows the CPU state. The source code pane shows the assembly instructions with some lines highlighted in yellow.

Registers	H	L
AX	02	03
BX	00	03
CX	00	0F
DX	00	00
CS	0700	
IP	010C	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```

0700:010C           0700:010C
B8 184 TAB         MOU AX, 00009h
09 009 TAB         MOU BX, 00003h
00 000 NULL        MUL BX
BB 187             MOU AL, 03h
03 003 HEART       MOU AH, 02h
00 000 NULL        MUL AH
F7 247 ≈           RET
E3 227 R           NOP
B0 176 ≡           NOP
03 003 HEART       NOP
B4 180 I           NOP
02 002 C           NOP
F6 246 ÷           NOP
E4 228 Σ           NOP
C3 195 I           NOP
90 144 E           ...

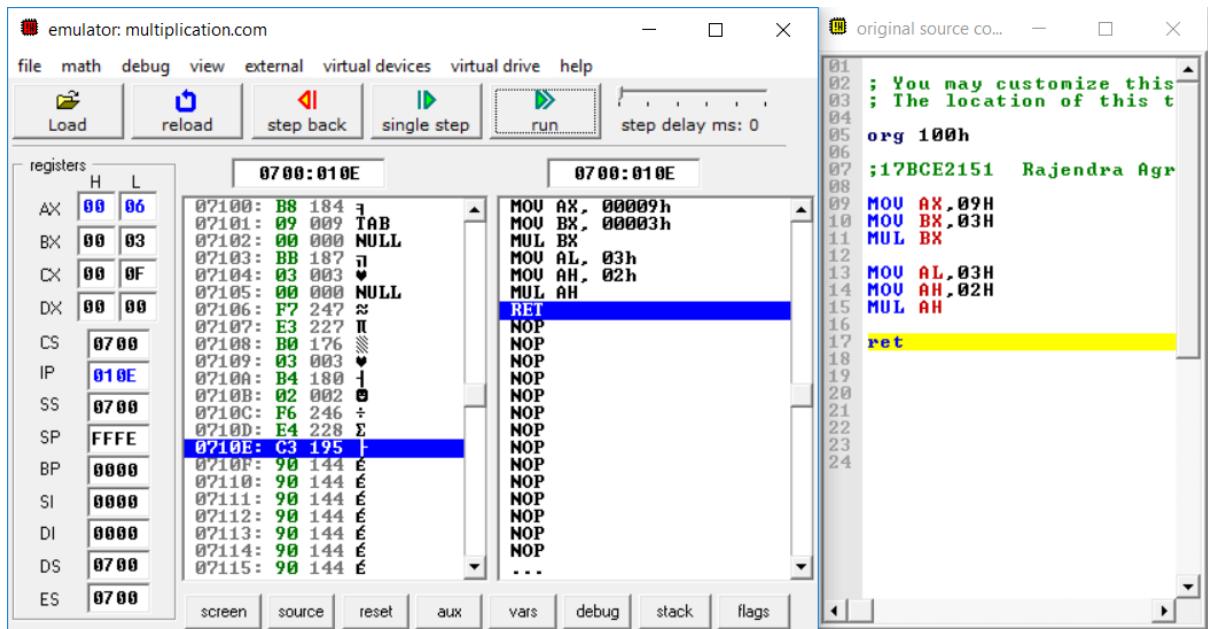
```

```

; You may customize this
; The location of this t
org 100h
;17BCE2151 Rajendra Agr
MOU AX, 09H
MOU BX, 03H
MUL BX
MOU AL, 03H
MOU AH, 02H
MUL AH
ret

```

Step 7: Multiply AH with AL and store result into AX.



The screenshot shows a debugger interface with two main panes. The left pane displays the assembly code and registers. The right pane shows the original source code. The assembly code pane has two columns: addresses 0700:010E and 0700:010F. The registers pane shows the CPU state. The source code pane shows the assembly instructions with some lines highlighted in yellow.

Registers	H	L
AX	00	06
BX	00	03
CX	00	0F
DX	00	00
CS	0700	
IP	010E	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

```

0700:010E           0700:010E
B8 184 TAB         MOU AX, 00009h
09 009 TAB         MOU BX, 00003h
00 000 NULL        MUL BX
BB 187             MOU AL, 03h
03 003 HEART       MOU AH, 02h
00 000 NULL        MUL AH
RET
E3 227 R           NOP
B0 176 ≡           NOP
03 003 HEART       NOP
B4 180 I           NOP
02 002 C           NOP
F6 246 ÷           NOP
E4 228 Σ           NOP
C3 195 I           NOP
90 144 E           ...

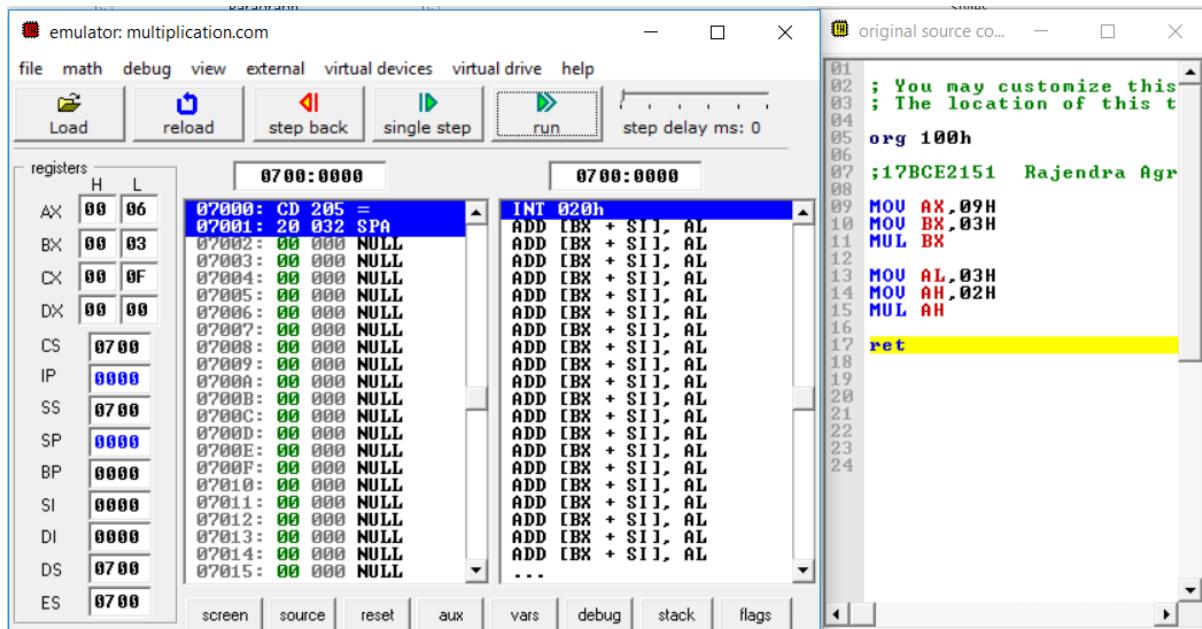
```

```

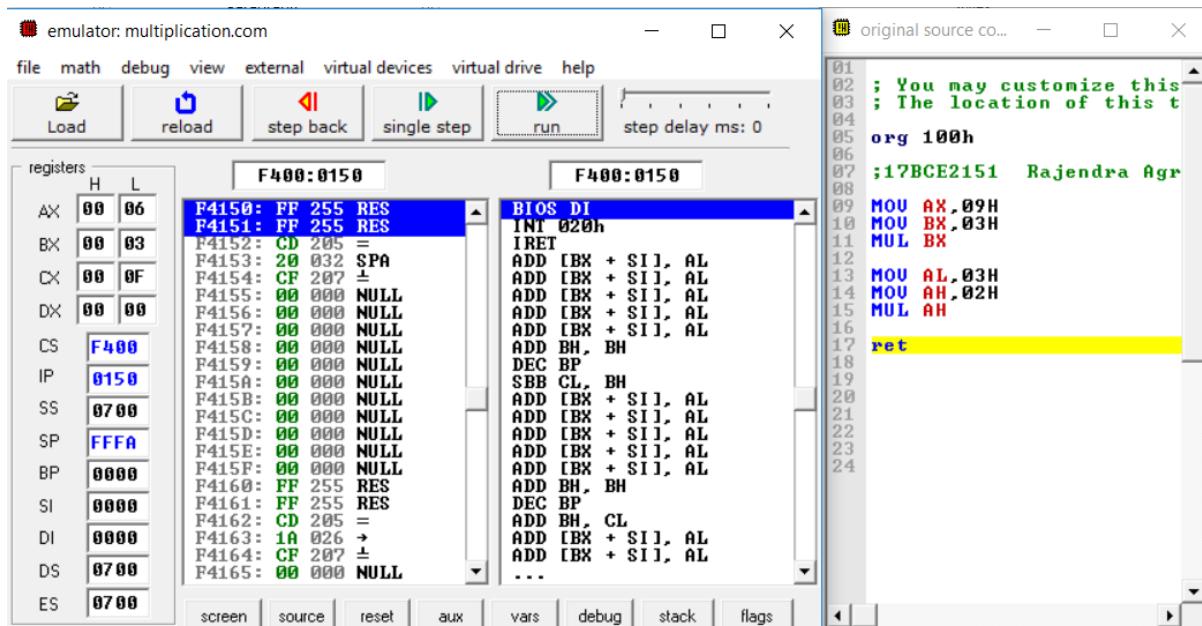
; You may customize this
; The location of this t
org 100h
;17BCE2151 Rajendra Agr
MOU AX, 09H
MOU BX, 03H
MUL BX
MOU AL, 03H
MOU AH, 02H
MUL AH
ret

```

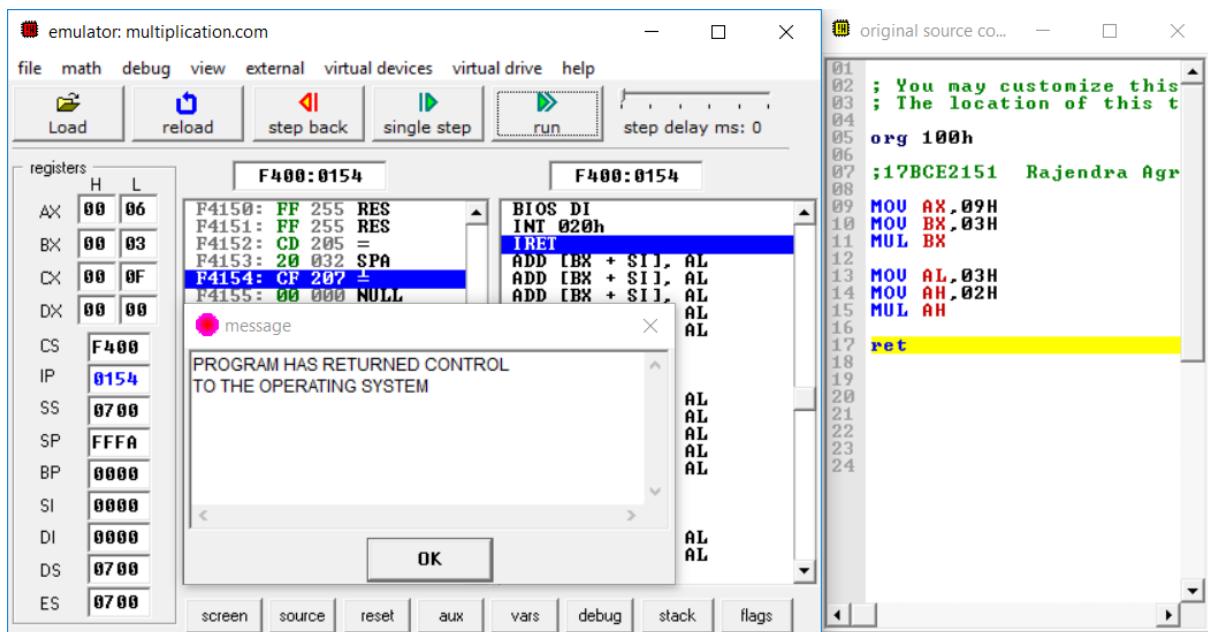
Step 8: Executing Return.



Step 9: Executing Return.



Step 10: Execution Complete.



4. Division:

Code Snippet:

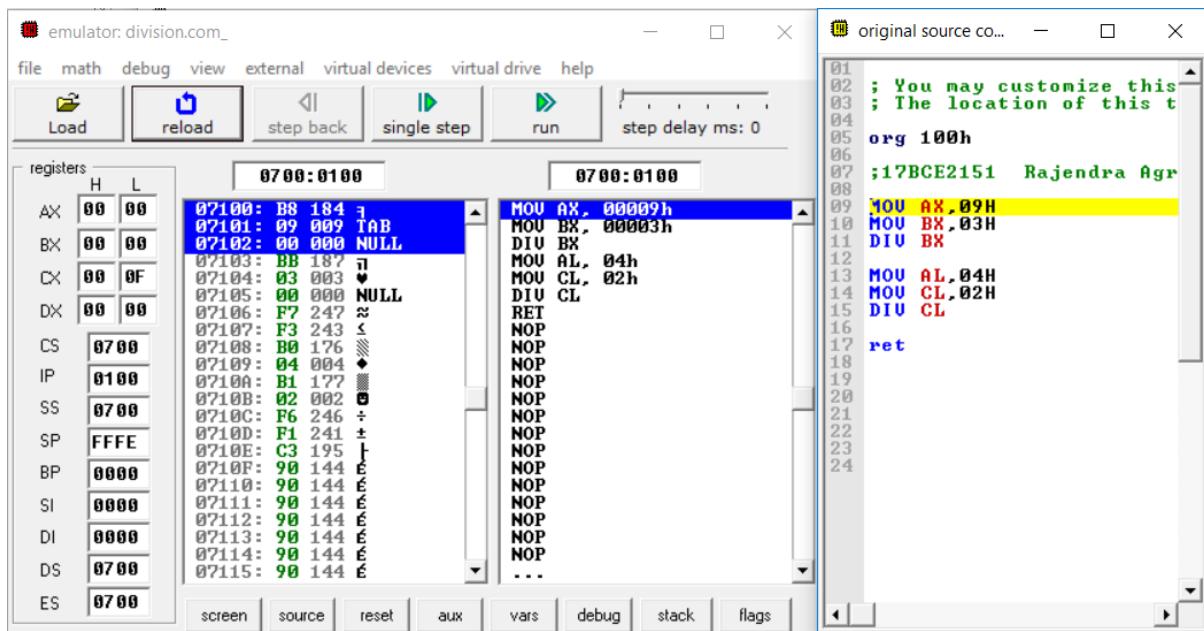
The screenshot shows the emu8086 assembler interface with the file 'division.asm' open. The code is as follows:

```
01 ; You may customize this and other start-up templates;
02 ; The location of this template is c:\emu8086\inc\0_com_template.txt
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agrawal
07
08 MOU AX,09H
09 MOU BX,03H
10 DIV BX
11
12 MOU AL,04H
13 MOU CL,02H
14 DIV CL
15
16 ret
17
18
19
20
21
22
```

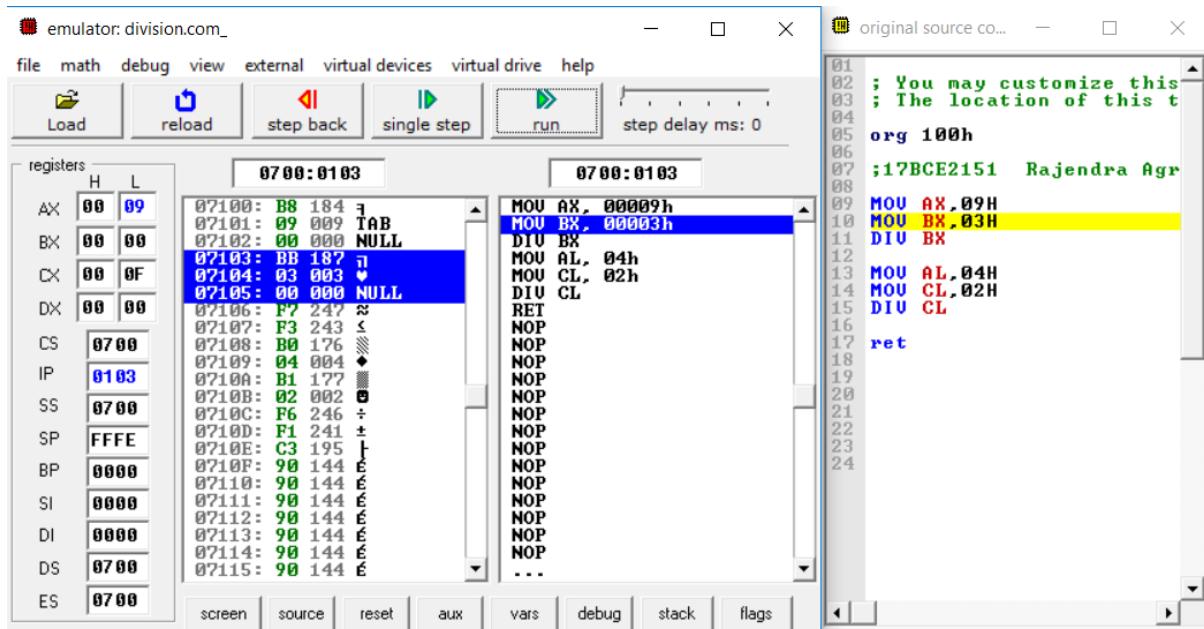
The code performs a division operation where AX is divided by BX, and the quotient is stored in AL and the remainder in CL.

Step by Step Execution:

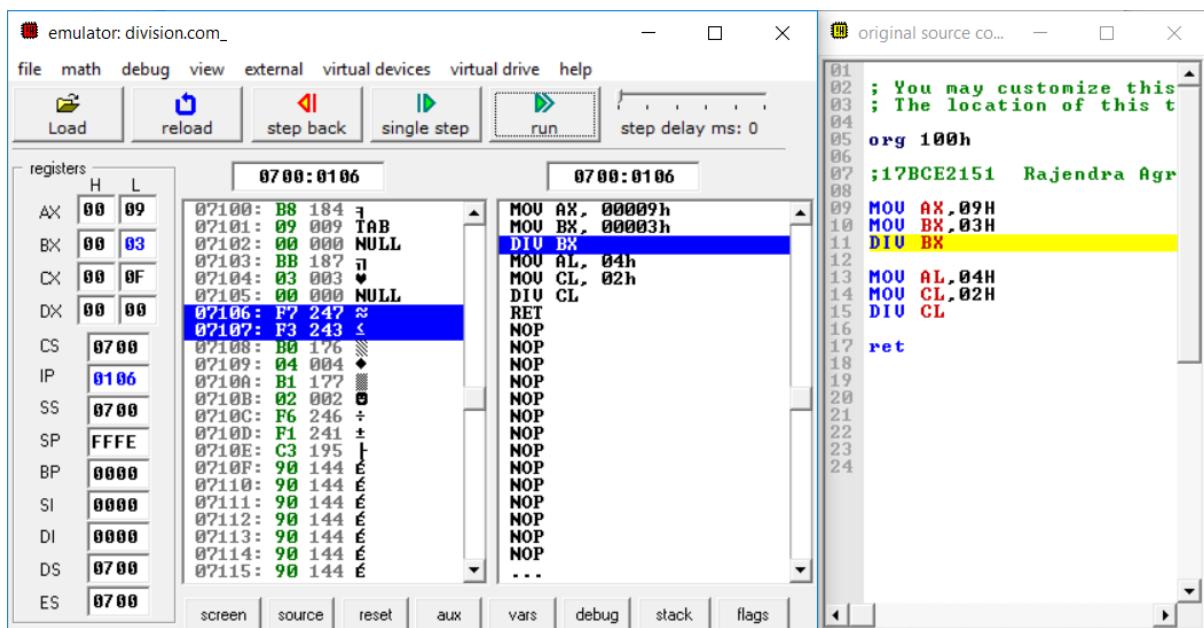
Step 1: Load the program.



Step 2: Move 09H into AX for word mode operation.



Step 3: Move 03H into BX.



Step 4: Divide AX by BX and store result into AX.

The screenshot shows a debugger interface with two windows. The left window displays the CPU Registers pane with the following values:

Registers	H	L
AX	00	03
BX	00	03
CX	00	0F
DX	00	00
CS	0700	
IP	0108	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The right window shows the assembly code:

```
01 ; You may customize this
02 ; The location of this t
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agr
07
08 MOU AX, 00009h
09 MOU BX, 00003h
10 DIV BX
11
12 MOU AL, 04H
13 MOU CL, 02H
14 DIV CL
15
16 ret
17
18
19
20
21
22
23
24
```

Step 5: Move 04H into AL for byte mode Operation.

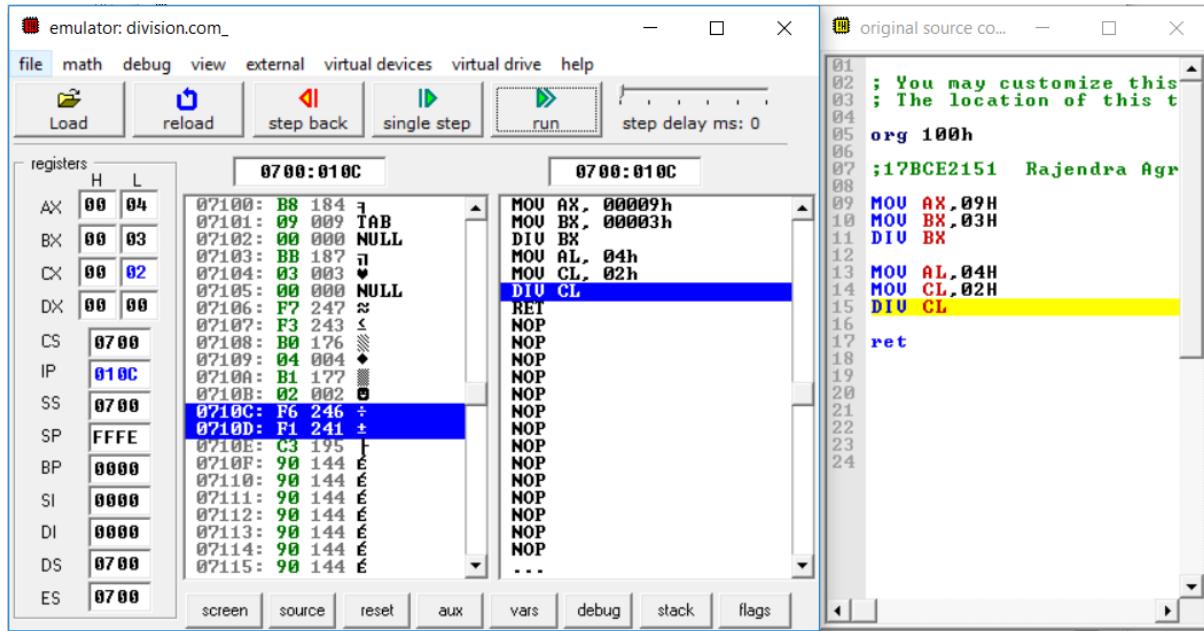
The screenshot shows a debugger interface with two windows. The left window displays the CPU Registers pane with the following values:

Registers	H	L
AX	00	04
BX	00	03
CX	00	0F
DX	00	00
CS	0700	
IP	010A	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

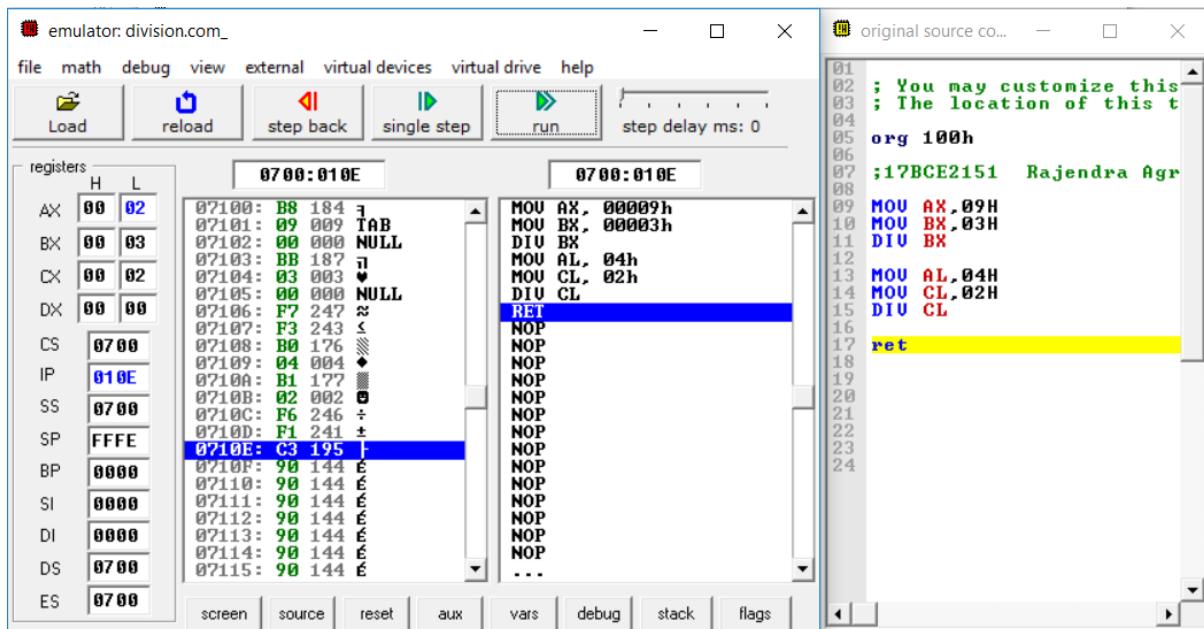
The right window shows the assembly code:

```
01 ; You may customize this
02 ; The location of this t
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agr
07
08 MOU AX, 00009h
09 MOU BX, 00003h
10 DIV BX
11
12 MOU AL, 04H
13 MOU CL, 02H
14 DIV CL
15
16 ret
17
18
19
20
21
22
23
24
```

Step 6: Move 02H into CL.



Step 7: Divide AL by CL and store result into AX.



Step 8: Executing return.

The screenshot shows a debugger interface with two panes. The left pane displays assembly code and registers. The assembly code pane shows the following code:

```
01 ; You may customize this
02 ; The location of this t
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agr
07
08 MOU AX,09H
09 MOU BX,03H
10 DIU BX
11
12 MOU AL,04H
13 MOU CL,02H
14 DIU CL
15
16 ret
17
18
19
20
21
22
23
24
```

The registers pane shows the following values:

	H	L
AX	00	02
BX	00	03
CX	00	02
DX	00	00
CS	0700	
IP	0000	
SS	0700	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The right pane shows the assembly code being executed, with the instruction at address 0700:0000 highlighted in yellow:

```
0700:0000 INT 020h
```

Step 9: Executing return.

The screenshot shows a debugger interface with two panes. The left pane displays assembly code and registers. The assembly code pane shows the following code:

```
01 ; You may customize this
02 ; The location of this t
03
04 org 100h
05
06 ;17BCE2151 Rajendra Agr
07
08 MOU AX,09H
09 MOU BX,03H
10 DIU BX
11
12 MOU AL,04H
13 MOU CL,02H
14 DIU CL
15
16 ret
17
18
19
20
21
22
23
24
```

The registers pane shows the following values:

	H	L
AX	00	02
BX	00	03
CX	00	02
DX	00	00
CS	F400	
IP	0150	
SS	0700	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The right pane shows the assembly code being executed, with the instruction at address F400:0150 highlighted in yellow:

```
F400:0150 BIOS DI
```

Step 10: Execution complete.

