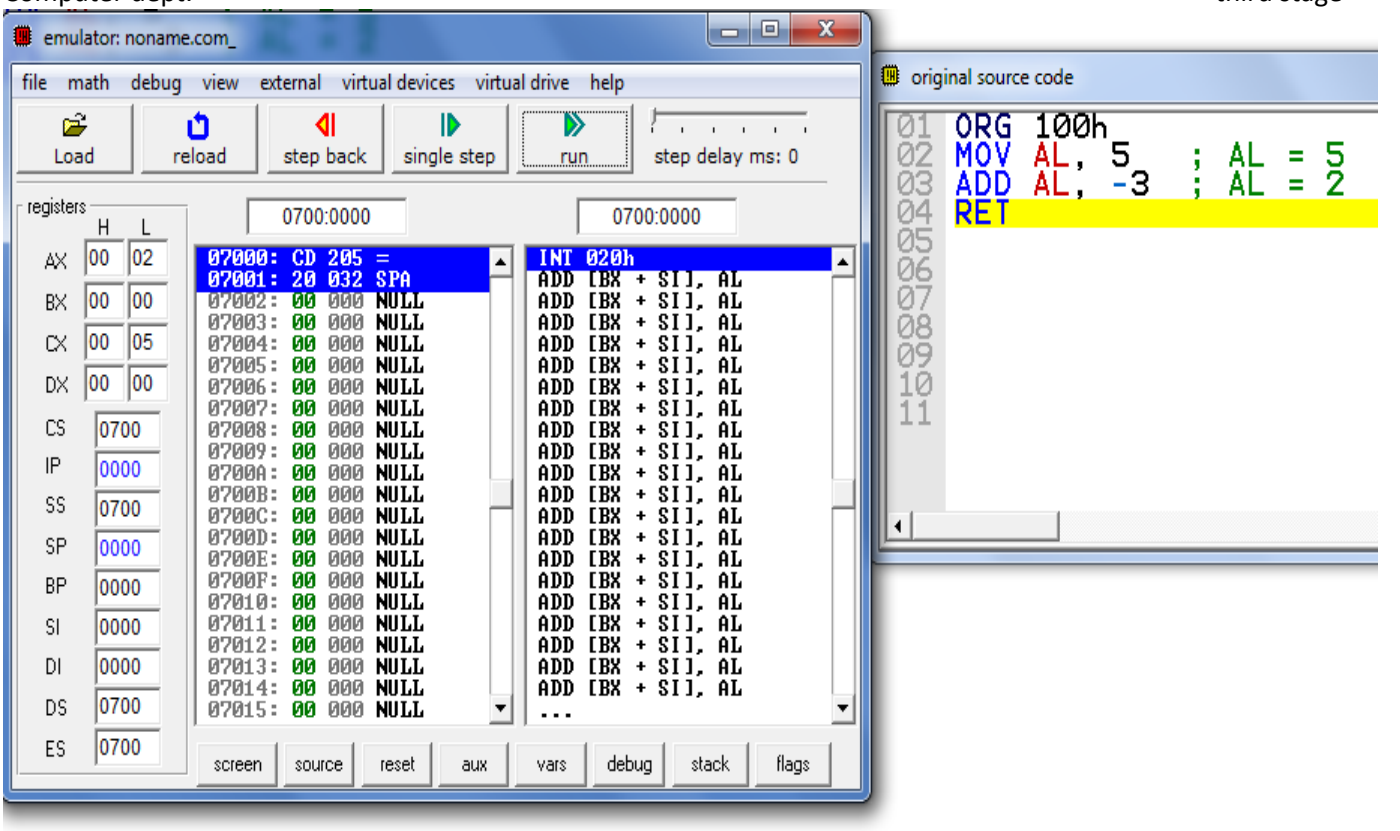
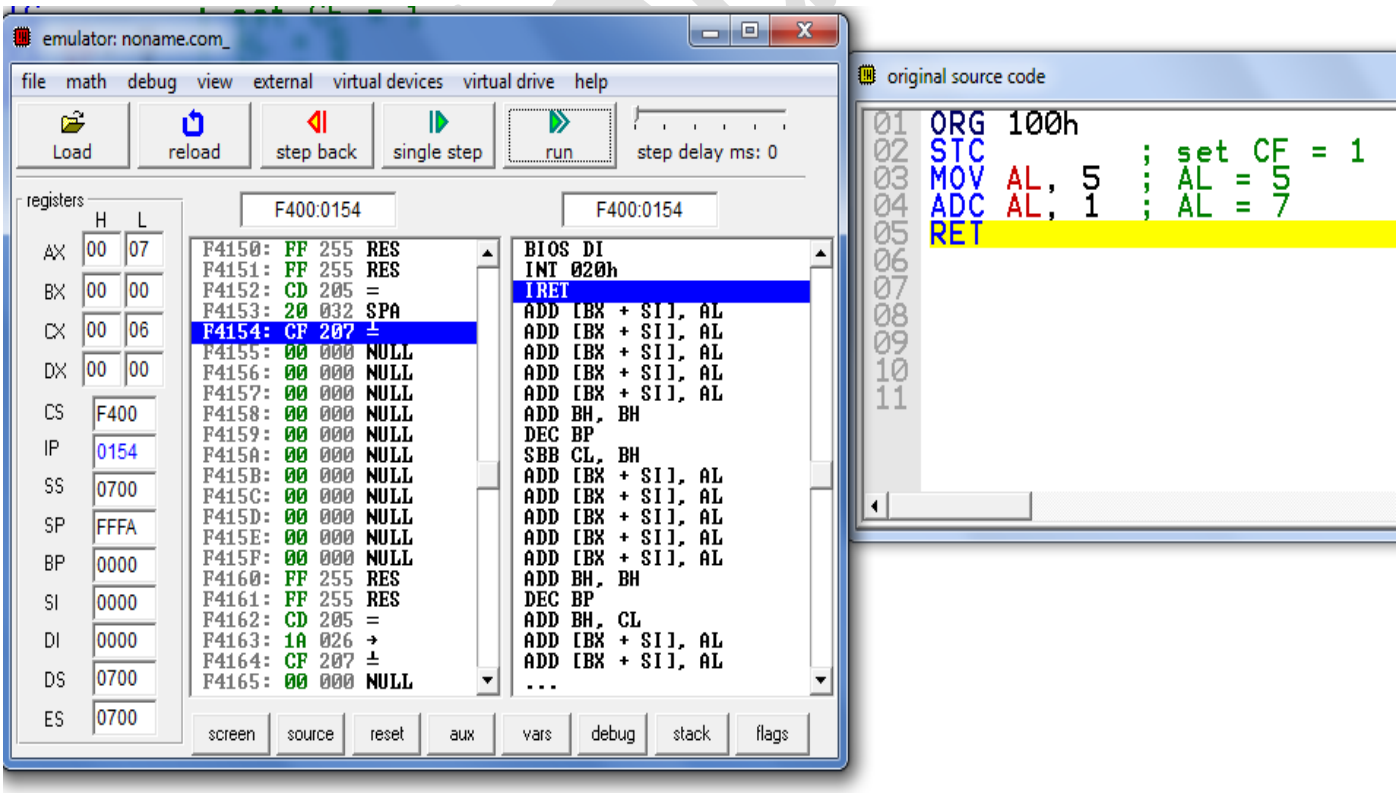
ADC	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Add with Carry. Algorithm: operand1 = operand1 + operand2 + CF Example: STC ; set CF = 1 MOV AL, 5 ; AL = 5 ADC AL, 1 ; AL = 7 RET C Z S O P A r r r r r r

ADD	REG, memory
	memory, REG
	REG, REG
	memory, immediate
	REG, immediate
		Add.
		
		Algorithm:
		
		operand1 = operand1 + operand2
		
		Example:
		MOV AL, 5 ; AL = 5
		ADD AL, -3 ; AL = 2
		RET
		C Z S O P A
		r r r r r r


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
H L
AX 00 02
BX 00 00
CX 00 05
DX 00 00
CS 0700
IP 0000
SS 0700
SP 0000
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
0700:0000
0700:0000
07000: CD 205 =
07001: 20 032 SPA
07002: 00 000 NULL
07003: 00 000 NULL
07004: 00 000 NULL
07005: 00 000 NULL
07006: 00 000 NULL
07007: 00 000 NULL
07008: 00 000 NULL
07009: 00 000 NULL
0700A: 00 000 NULL
0700B: 00 000 NULL
0700C: 00 000 NULL
0700D: 00 000 NULL
0700E: 00 000 NULL
0700F: 00 000 NULL
07010: 00 000 NULL
07011: 00 000 NULL
07012: 00 000 NULL
07013: 00 000 NULL
07014: 00 000 NULL
07015: 00 000 NULL
INT 020h
ADD [BX + SI], AL
...
screen source reset aux vars debug stack flags
original source code
01 ORG 100h
02 MOV AL, 5 ; AL = 5
03 ADD AL, -3 ; AL = 2
04 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
H L
AX 00 07
BX 00 00
CX 00 06
DX 00 00
CS F400
IP 0154
SS 0700
SP FFFA
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
F400:0154
F400:0154
F4150: FF 255 RES
F4151: FF 255 RES
F4152: CD 205 =
F4153: 20 032 SPA
F4154: CF 207 ±
F4155: 00 000 NULL
F4156: 00 000 NULL
F4157: 00 000 NULL
F4158: 00 000 NULL
F4159: 00 000 NULL
F415A: 00 000 NULL
F415B: 00 000 NULL
F415C: 00 000 NULL
F415D: 00 000 NULL
F415E: 00 000 NULL
F415F: 00 000 NULL
F4160: FF 255 RES
F4161: FF 255 RES
F4162: CD 205 =
F4163: 1A 026 →
F4164: CF 207 ±
F4165: 00 000 NULL
BIOS DI
INT 020h
IRET
ADD [BX + SI], AL
ADD BH, BH
DEC BP
SBB CL, BH
ADD BH, CL
...
screen source reset aux vars debug stack flags
original source code
01 ORG 100h
02 STC ; set CF = 1
03 MOV AL, 5 ; AL = 5
04 ADC AL, 1 ; AL = 7
05 RET

<table>
<tr>
<td>LEA</td>
<td>REG,
memory</td>
<td>
Load Effective Address.

Algorithm:

• REG = address of memory (offset)

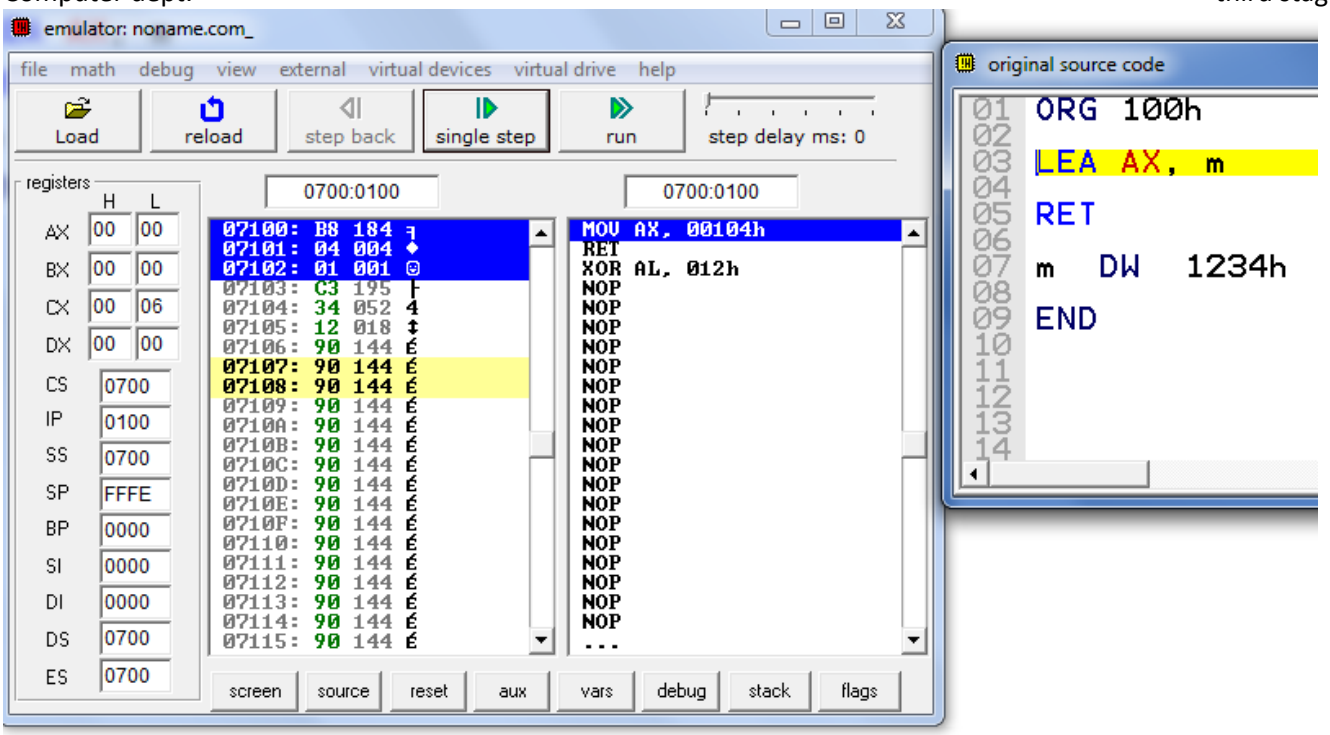
Generally this instruction is replaced by MOV when assembling when possible.

Example:

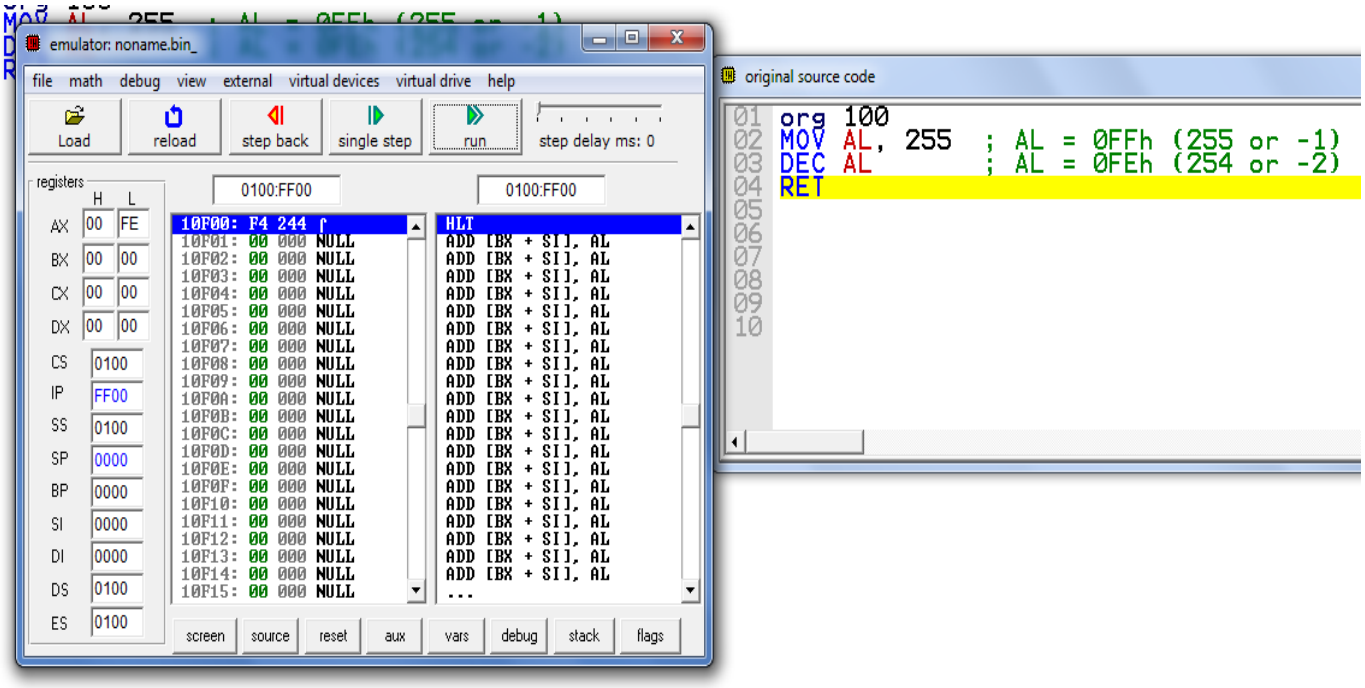
ORG 100h
LEA AX, m
RET
m DW 1234h
END

AX is set to: 0104h.
LEA instruction takes 3 bytes, RET takes 1 byte, we start at 100h, so the address of 'm' is 104h.

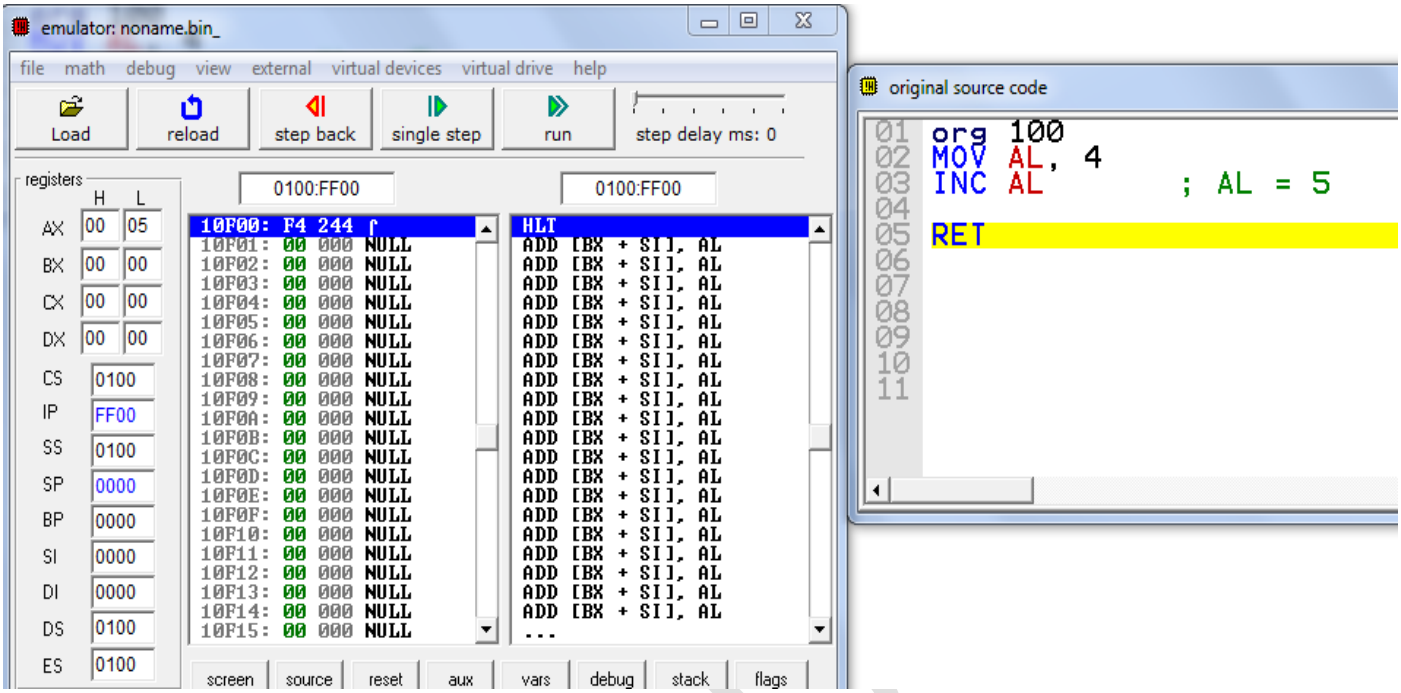
<table>
<tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr>
<tr><td colspan="6">unchanged</td></tr>
</table>
</td>
</tr>
</table>



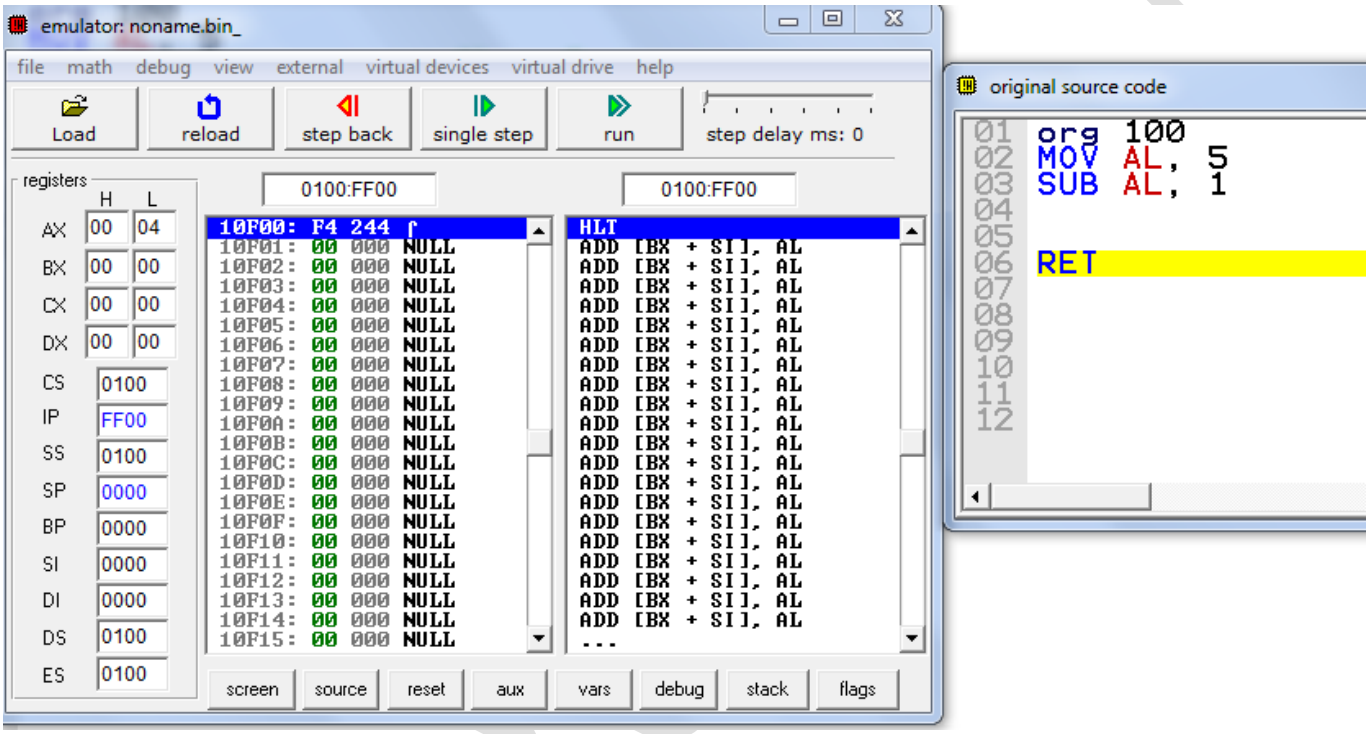
DEC	REG memory	Decrement. Algorithm: operand = operand - 1 Example: MOV AL, 255 ; AL = 0FFh (255 or -1) DEC AL ; AL = 0FEh (254 or -2) RET Z S O P A r r r r r CF - unchanged!



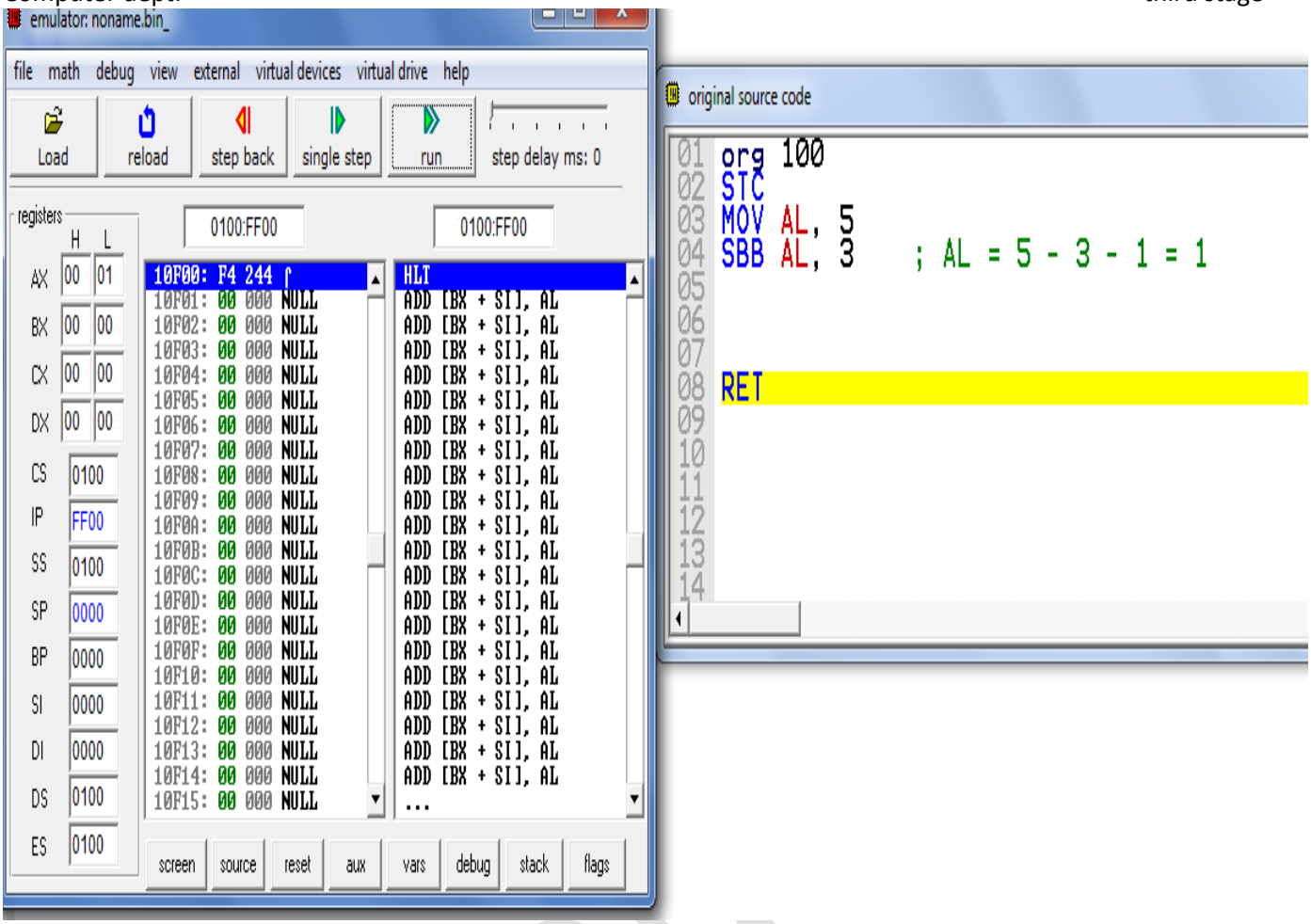
INC	REG memory	Increment. Algorithm: operand = operand + 1 Example: MOV AL, 4 INC AL ; AL = 5 RET Z S O P A r r r r r CF - unchanged!



SUB	REG, memory
	memory, REG
	REG, REG
	memory, immediate
	REG, immediate
		Subtract.
		
		Algorithm:
		
		operand1 = operand1 - operand2
		
		Example:
		MOV AL, 5
		SUB AL, 1 ; AL = 4
		
		RET
		C Z S O P A
		r r r r r r

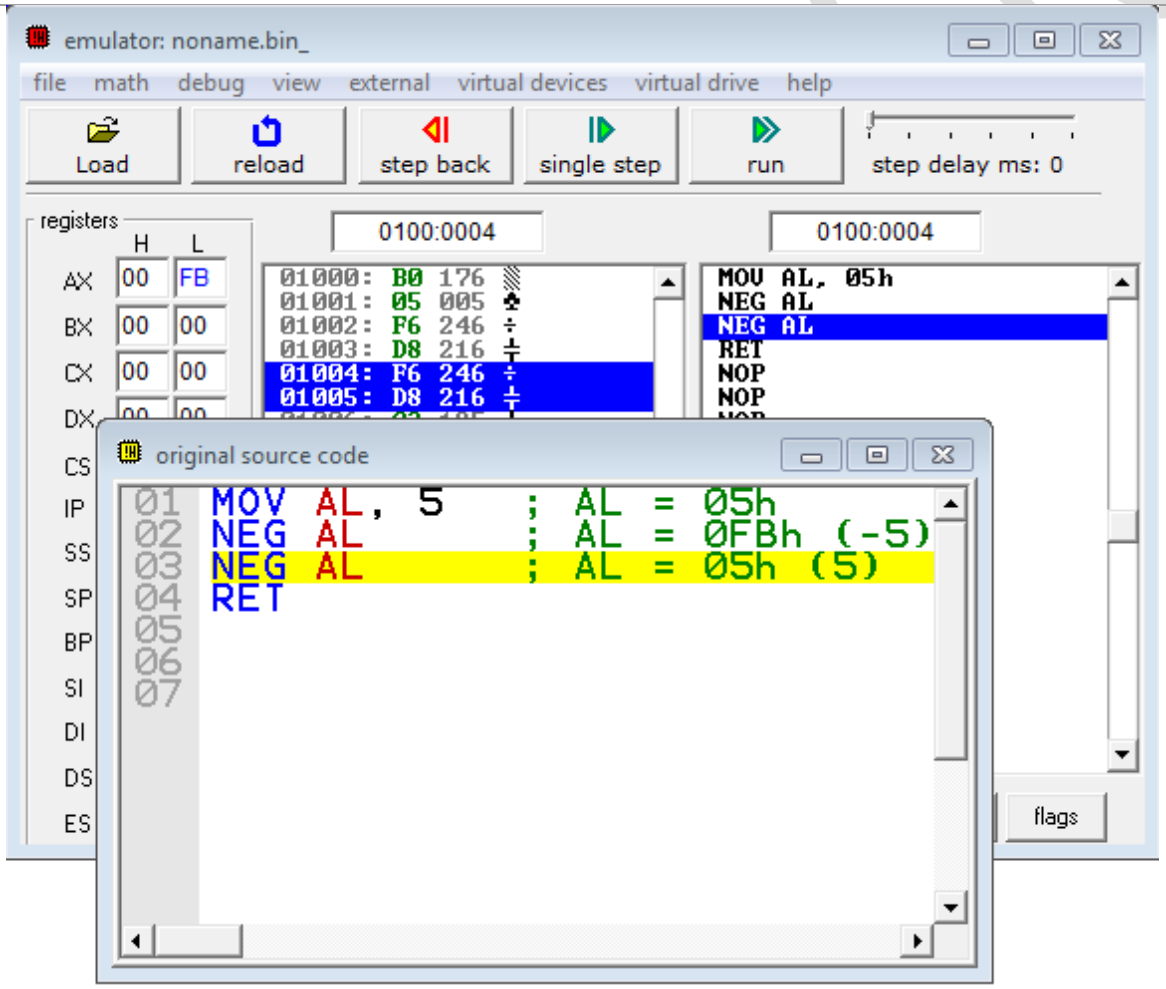


SBB	REG, memory memory, REG REG, REG memory, immediate REG, immediate	Subtract with Borrow. Algorithm: operand1 = operand1 - operand2 - CF Example: STC MOV AL, 5 SBB AL, 3 ; AL = 5 - 3 - 1 = 1 RET C Z S O P A r r r r r r


emulator: noname.bin
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 00 01
BX 00 00
CX 00 00
DX 00 00
CS 0100
IP FF00
SS 0100
SP 0000
BP 0000
SI 0000
DI 0000
DS 0100
ES 0100
0100:FF00
0100:FF00
10F00: F4 244 HLT
10F01: 00 000 NULL ADD [BX + SI], AL
10F02: 00 000 NULL ADD [BX + SI], AL
10F03: 00 000 NULL ADD [BX + SI], AL
10F04: 00 000 NULL ADD [BX + SI], AL
10F05: 00 000 NULL ADD [BX + SI], AL
10F06: 00 000 NULL ADD [BX + SI], AL
10F07: 00 000 NULL ADD [BX + SI], AL
10F08: 00 000 NULL ADD [BX + SI], AL
10F09: 00 000 NULL ADD [BX + SI], AL
10F0A: 00 000 NULL ADD [BX + SI], AL
10F0B: 00 000 NULL ADD [BX + SI], AL
10F0C: 00 000 NULL ADD [BX + SI], AL
10F0D: 00 000 NULL ADD [BX + SI], AL
10F0E: 00 000 NULL ADD [BX + SI], AL
10F0F: 00 000 NULL ADD [BX + SI], AL
10F10: 00 000 NULL ADD [BX + SI], AL
10F11: 00 000 NULL ADD [BX + SI], AL
10F12: 00 000 NULL ADD [BX + SI], AL
10F13: 00 000 NULL ADD [BX + SI], AL
10F14: 00 000 NULL ADD [BX + SI], AL
10F15: 00 000 NULL ...
screen source reset aux vars debug stack flags
original source code
01 org 100
02 STC
03 MOV AL, 5
04 SBB AL, 3 ; AL = 5 - 3 - 1 = 1
05
06
07
08 RET
09
10
11
12
13
14

^

NEG	REG memory	Negate. Makes operand negative (two's complement). Algorithm: • Invert all bits of the operand • Add 1 to inverted operand Example: MOV AL, 5 ; AL = 05h NEG AL ; AL = 0FBh (-5) NEG AL ; AL = 05h (5) RET C Z S O P A r r r r r r



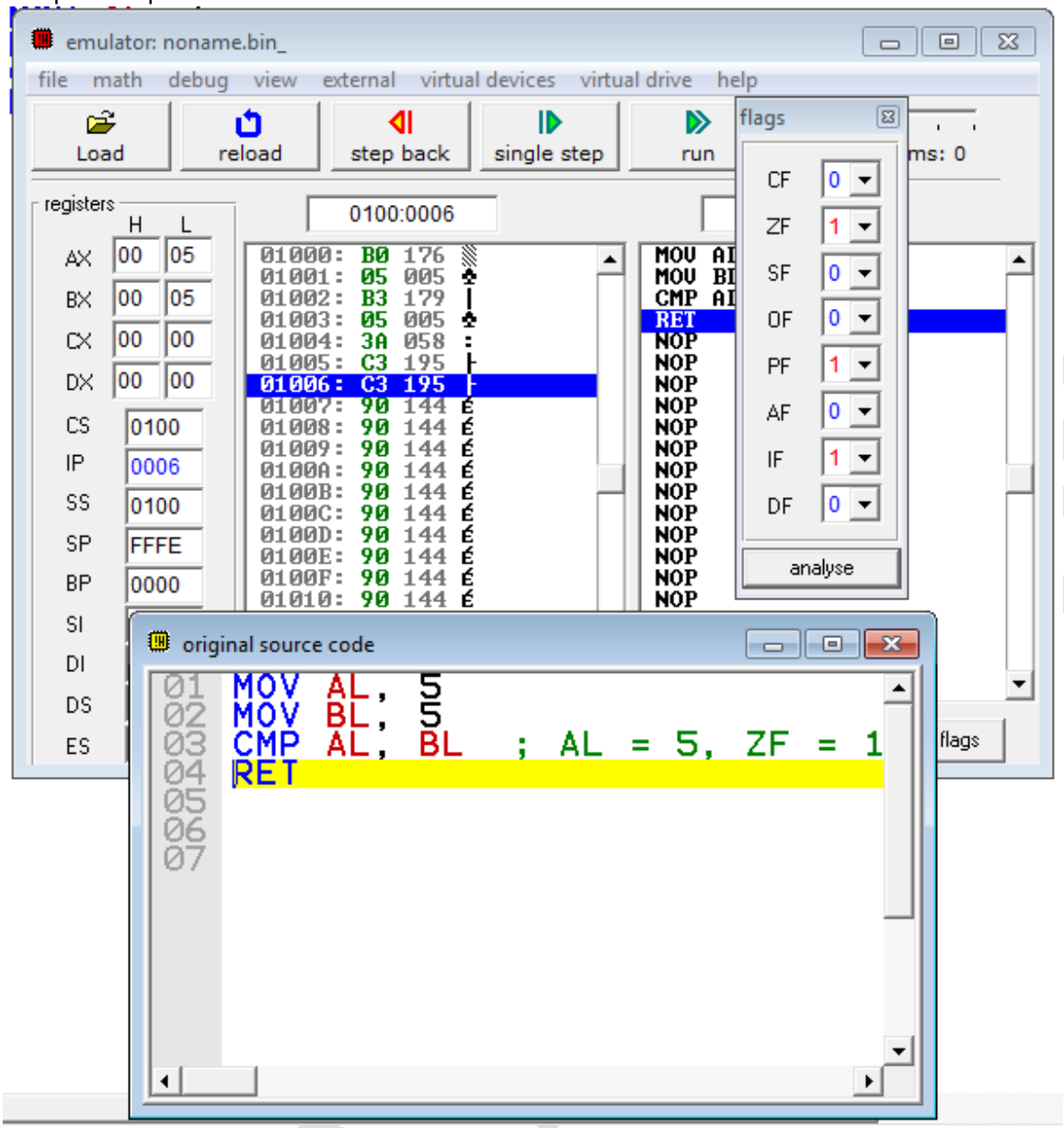
<table>
<tr>
<td>CMP</td>
<td>REG, memory
memory, REG
REG, REG
memory, immediate
REG, immediate</td>
<td>Compare.

Algorithm:

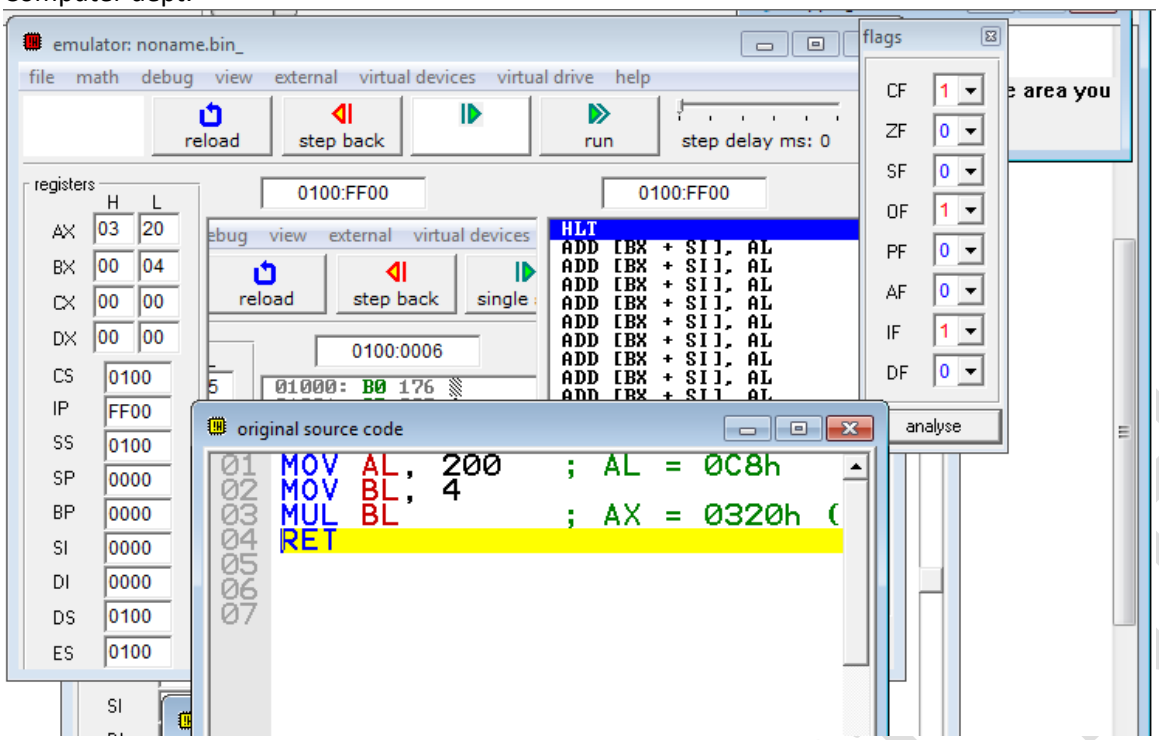
operand1 - operand2

result is not stored anywhere, flags are set (OF, SF, ZF, AF, PF, CF) according to result.

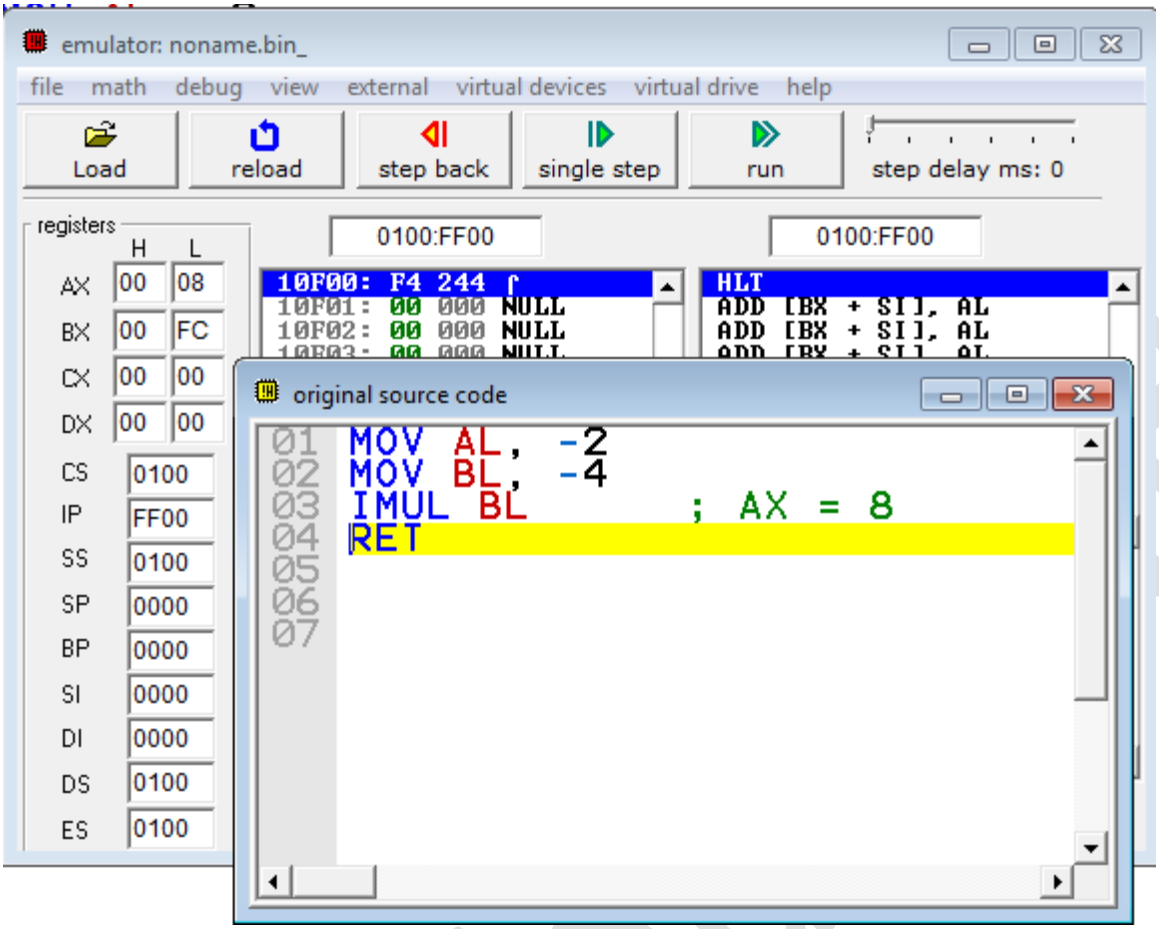
Example:
MOV AL, 5
MOV BL, 5
CMP AL, BL ; AL = 5, ZF = 1 (so equal!)
RET
<table>
<tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr>
<tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr>
</table>
</td>
</tr>
</table>



MUL	REG
	memory
		Unsigned multiply.
		
		Algorithm:
		
		when operand is a **byte**:
		AX = AL * operand.
		when operand is a **word**:
		(DX AX) = AX * operand.
		
		Example:
		MOV AL, 200 ; AL = 0C8h
		MOV BL, 4
		MUL BL ; AX = 0320h (800)
		RET
		
		C Z S O P A
		r ? ? r ? ?
		
		CF=OF=0 when high section of the result is zero.

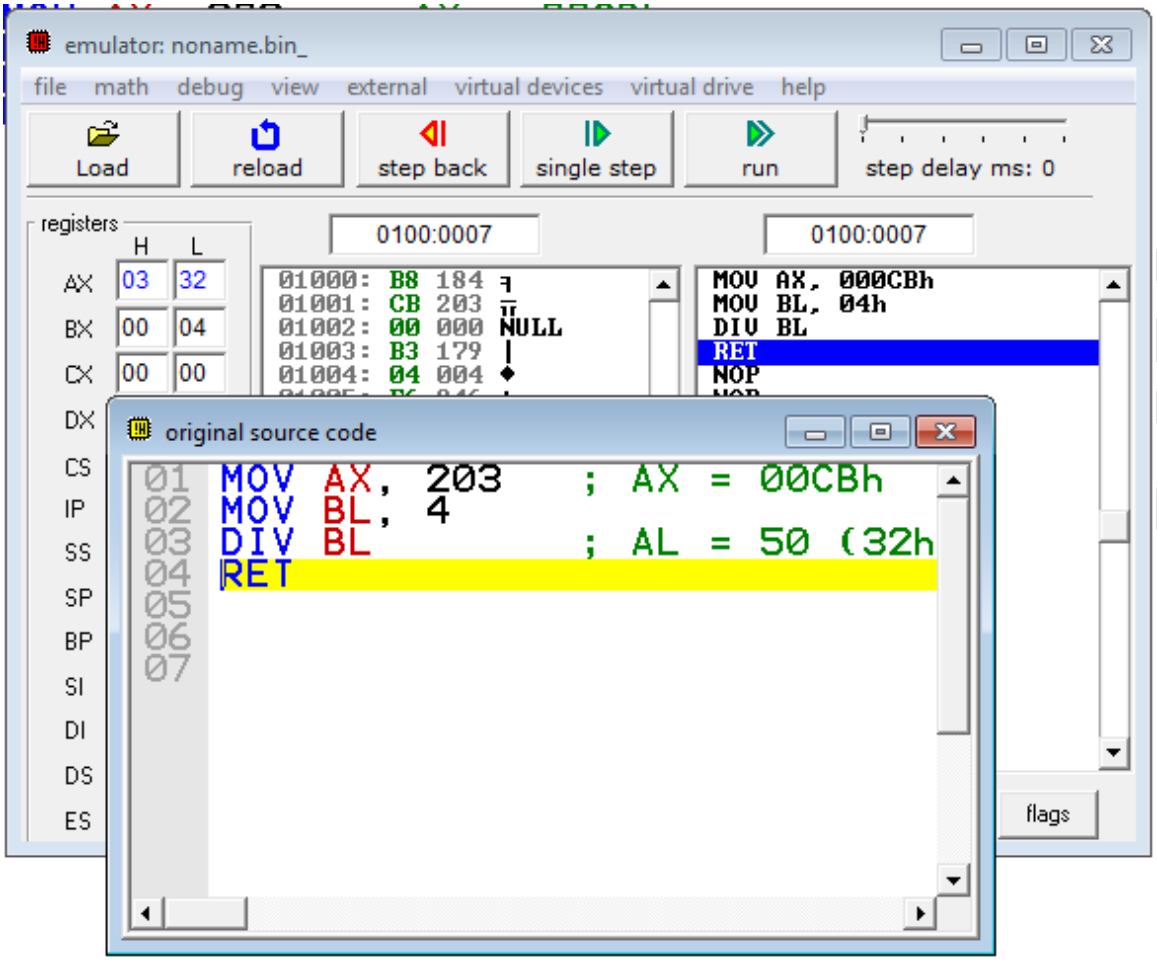


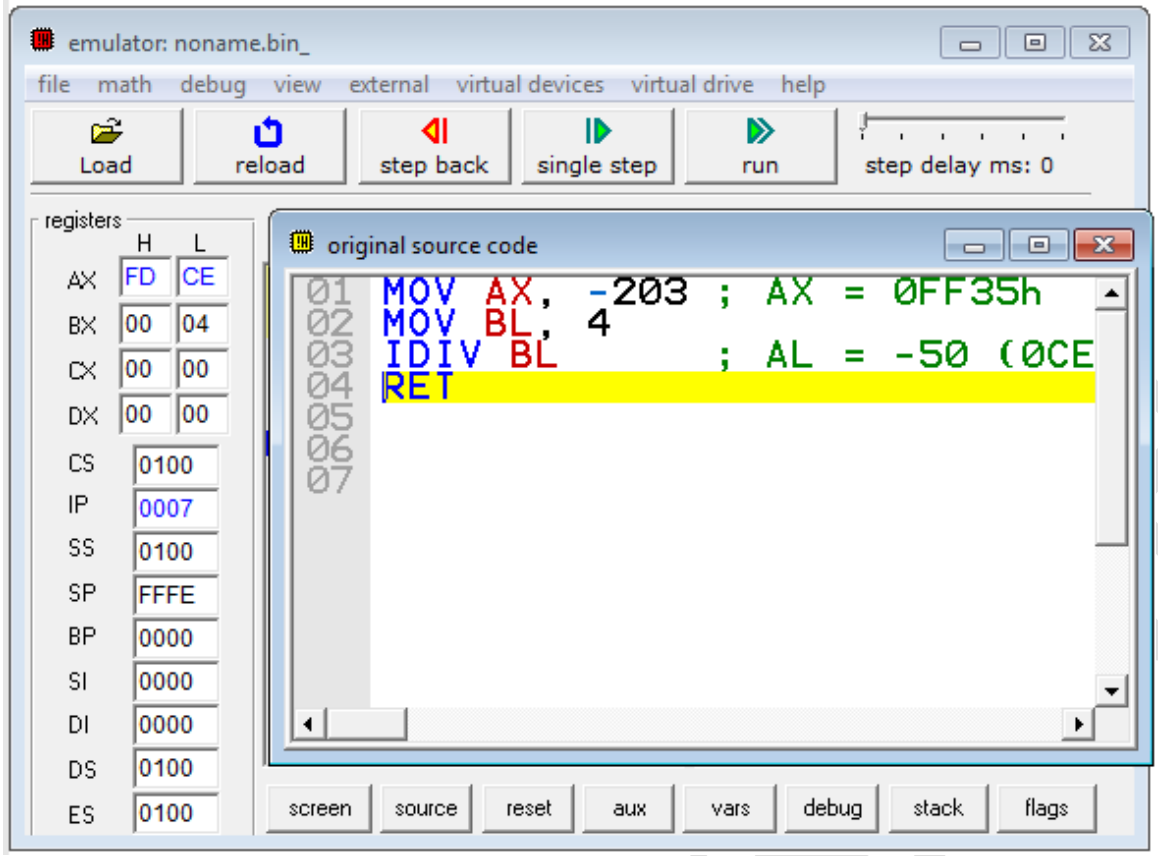
IMUL	REG
	memory
		Signed multiply.
		
		Algorithm:
		
		when operand is a **byte**:
		AX = AL * operand.
		when operand is a **word**:
		(DX AX) = AX * operand.
		
		Example:
		MOV AL, -2
		MOV BL, -4
		IMUL BL ; AX = 8
		RET
		
		C Z S O P A
		r ? ? r ? ?
		
		CF=OF=0 when result fits into operand of IMUL.


emulator: noname.bin_
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 00 08
BX 00 FC
CX 00 00
DX 00 00
CS 0100
IP FF00
SS 0100
SP 0000
BP 0000
SI 0000
DI 0000
DS 0100
ES 0100
0100:FF00
0100:FF00
10F00: F4 244 HLT
10F01: 00 000 NULL ADD [BX + SI], AL
10F02: 00 000 NULL ADD [BX + SI], AL
original source code
01 MOV AL, -2
02 MOV BL, -4
03 IMUL BL ; AX = 8
04 RET
05
06
07

DIV	REG
	memory
		Unsigned divide.
		
		Algorithm:
		
		when operand is a byte:
		AL = AX / operand
		AH = remainder (modulus)
		
		when operand is a word:
		AX = (DX AX) / operand
		DX = remainder (modulus)
		
		Example:
		
		MOV AX, 203 ; AX = 00CBh
		
		MOV BL, 4
		
		DIV BL ; AL = 50 (32h), AH = 3
		
		RET
		
		C Z S O P A
		? ? ? ? ? ?

IDIV	REG
	memory
		Signed divide.
		
		Algorithm:
		
		when operand is a **byte**:
		AL = AX / operand
		AH = remainder (modulus)
		when operand is a **word**:
		AX = (DX AX) / operand
		DX = remainder (modulus)
		
		Example:
		MOV AX, -203 ; AX = 0FF35h
		MOV BL, 4
		IDIV BL ; AL = -50 (0CEh), AH = -3 (0FDh)
		RET
		C Z S O P A
		? ? ? ? ? ?


emulator: noname.bin_
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 03 32
BX 00 04
CX 00 00
DX
CS
IP
SS
SP
BP
SI
DI
DS
ES
0100:0007
01000: B8 184
01001: CB 203
01002: 00 000 NULL
01003: B3 179
01004: 04 004
0100:0007
MOV AX, 000CBh
MOV BL, 04h
DIV BL
RET
NOP
flags
original source code
01 MOV AX, 203 ; AX = 00CBh
02 MOV BL, 4
03 DIV BL ; AL = 50 (32h
04 RET
05
06
07


emulator: noname.bin_
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 FD CE
BX 00 04
CX 00 00
DX 00 00
CS 0100
IP 0007
SS 0100
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0100
ES 0100
original source code
01 MOV AX, -203 ; AX = 0FF35h
02 MOV BL, 4
03 IDIV BL ; AL = -50 (0CE
04 RET
05
06
07
screen source reset aux vars debug stack flags