Addressing modes

There are three general types of addressing modes:

- Immediate addressing modes.
- Register addressing modes.
- Memory addressing modes.

Immediate addressing modes.

Suppose that in a program we need to put the number 526AH in the CX register. The MOV CX, 526AH instruction can be used to do this. This instruction will put the *immediate* hexadecimal number 526AH in the 16- bit CX register. This is referred to as immediate addressing mode because the number to be loaded into the CX register will be put in two memory locations immediately following the code for the MOV instruction.

MOV CL, 48H

MOV CX, 526AH

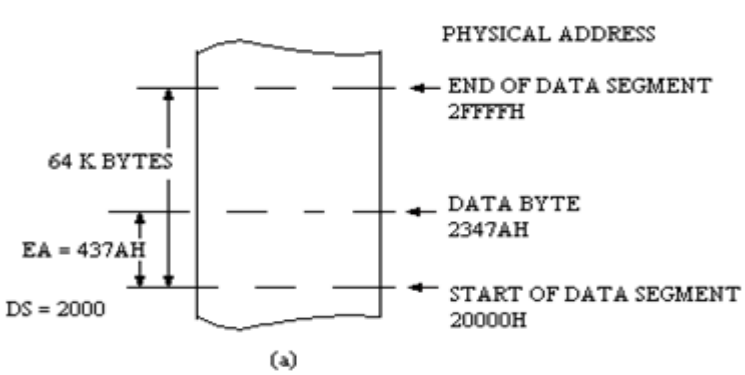
- Register Addressing mode

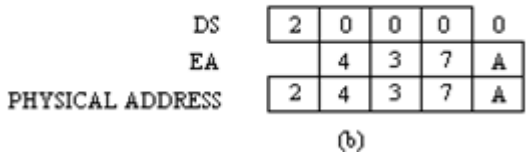
MOV CX, AX

MOV CH, AL

Memory Addressing Modes

To access data in memory the 8086 must produce a 20-bit physical address. It is done by adding a 16-bit value called *the effective address* to one of the four segment bases. This effective address (EA) represents the *displacement* or *offset* of the desired operand from the segment base. Any of the segment bases can be specified, but the data segment is the one most often used





(b)

Direct Addressing Mode

For the simplest memory addressing mode the effective address is just an 8-bit or 16-bit number written directly in the instruction. The instruction MOV CL ,[437AH] is an example. The brackets around the 437AH are shorthand for "the contents of the memory location at a displacement from the segment base of". When executed, this instruction will copy the contents of the memory location, at a displacement of 437AH from the data segment base into the CL register. This addressing mode is called *direct* because the displacement of the operand from the segment base is specified directly in the instruction.

Another example of this addressing mode is the instruction MOV BX, [437AH]. When executed, this instruction copies a word from memory into BX register. Since each memory address of the 8086 represents a byte of storage, the word must come from two memory locations. The byte at a displacement of 437AH from the data segment base will be copied into BL. The contents of the next higher address, displacement 437BH will be copied into BH register. The 8086 will automatically access the required number of bytes in memory for a given instruction.

MOV CL ,[437AH]

MOV BX, [437AH].

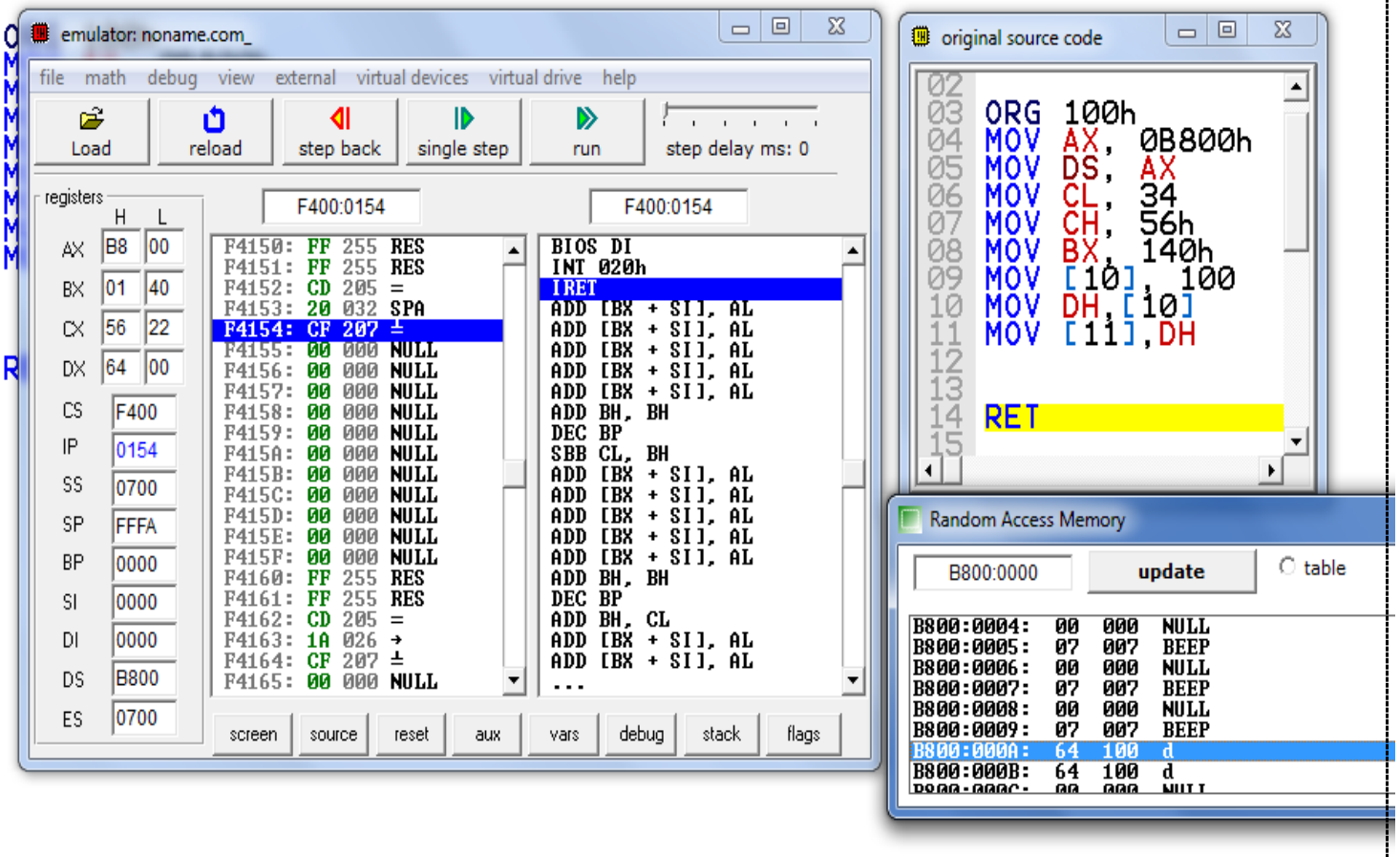
The instruction MOV[437AH], BX for example will copy the contents of the BX register to two memory locations in the data segment. . The contents of BL will be copied to the memory location as a displacement of 437AH and the contents of BH will be copied to the memory location at a displacement of 437BH.

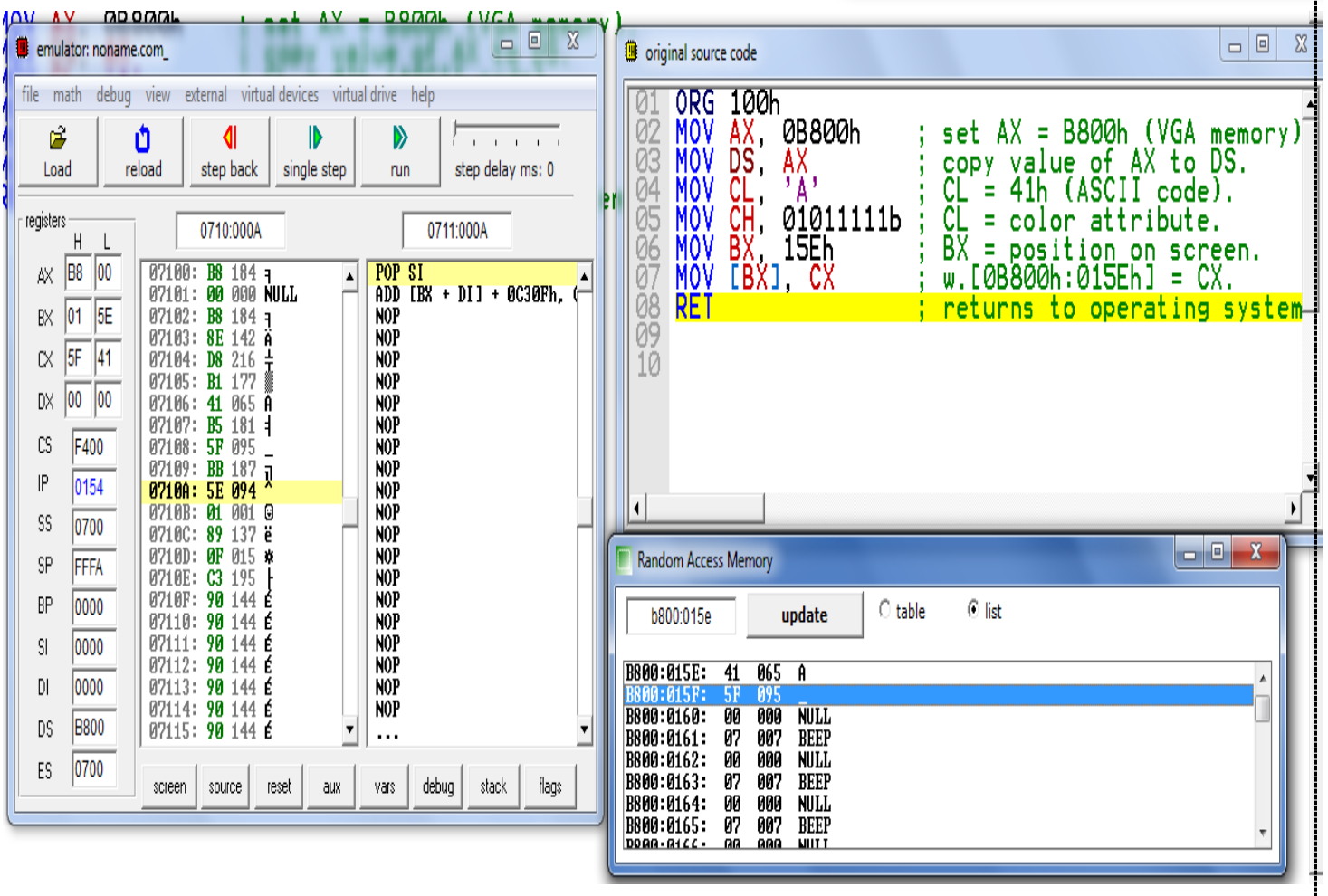
Indirect Addressing mode

In the indirect addressing mode, the memory address is not directly given. A register is used to indicate the address where the data can be found. Therefore , the register acts as an indirect address to locate the data. For example, in the instruction MOV (BX), CX the source of data is the CX register. The destination where the data are to be placed or copied to, is the address pointed to by the BX register. The brackets ( ) around BX indicate that the BX register contains an address and not a numeric value.

MOV	- REG, memory - memory, REG -REG, REG - memory, immediate - REG, immediate - SREG, memory - memory, SREG - REG, SREG - SREG, REG .	. Copy operand2 to operand1 . The MOV instruction cannot : • set the value of the CS and IP registers . • copy value of one segment register to another segment register (should copy to general register first). • copy immediate value to segment register (should copy to general register first). Algorithm : operand1 = operand2 C Z S O P A unchanged

Examples


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 B8 00
BX 01 40
CX 56 22
DX 64 00
CS F400
IP 0154
SS 0700
SP FFFA
BP 0000
SI 0000
DI 0000
DS B800
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 [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
SBB CL, BH
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD BH, BH
DEC BP
ADD BH, CL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
screen source reset aux vars debug stack flags
original source code
02
03 ORG 100h
04 MOV AX, 0B800h
05 MOV DS, AX
06 MOV CL, 34
07 MOV CH, 56h
08 MOV BX, 140h
09 MOV [10], 100
10 MOV DH,[10]
11 MOV [11],DH
12
13
14 RET
15
Random Access Memory
B800:0000
update
table
B800:0004: 00 000 NULL
B800:0005: 07 007 BEEP
B800:0006: 00 000 NULL
B800:0007: 07 007 BEEP
B800:0008: 00 000 NULL
B800:0009: 07 007 BEEP
B800:000A: 64 100 d
B800:000B: 64 100 d


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 B8 00
BX 01 5E
CX 5F 41
DX 00 00
CS F400
IP 0154
SS 0700
SP FFFA
BP 0000
SI 0000
DI 0000
DS B800
ES 0700
0710:000A
0711:000A
07100: B8 184 ╕
07101: 00 000 NULL
07102: B8 184 ╕
07103: 8E 142 Ä
07104: D8 216 ╪
07105: B1 177 ▒
07106: 41 065 A
07107: B5 181 ╡
07108: 5F 095 _
07109: BB 187 ╗
0710A: 5E 094 ^
0710B: 01 001 ☺
0710C: 89 137 ë
0710D: 0F 015 ☼
0710E: C3 195 ├
0710F: 90 144 É
07110: 90 144 É
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
07115: 90 144 É
POP SI
ADD [BX + DI] + 0C30Fh, (
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen source reset aux vars debug stack flags
original source code
01 ORG 100h
02 MOV AX, 0B800h ; set AX = B800h (VGA memory)
03 MOV DS, AX ; copy value of AX to DS.
04 MOV CL, 'A' ; CL = 41h (ASCII code).
05 MOV CH, 01011111b ; CL = color attribute.
06 MOV BX, 15Eh ; BX = position on screen.
07 MOV [BX], CX ; w.[0B800h:015Eh] = CX.
08 RET ; returns to operating system
09
10
Random Access Memory
b800:015e
update
table
list
B800:015E: 41 065 A
B800:015F: 5F 095 _
B800:0160: 00 000 NULL
B800:0161: 07 007 BEEP
B800:0162: 00 000 NULL
B800:0163: 07 007 BEEP
B800:0164: 00 000 NULL
B800:0165: 07 007 BEEP