Copying Data

Courtesy: UMBC and JBLearning

`mov` Instructions

- `mov` ("move") instructions are really copy instructions, like simple assignment statements in a high-level language.
- Format: `mov` *`destination, source`*

Effect on Flags

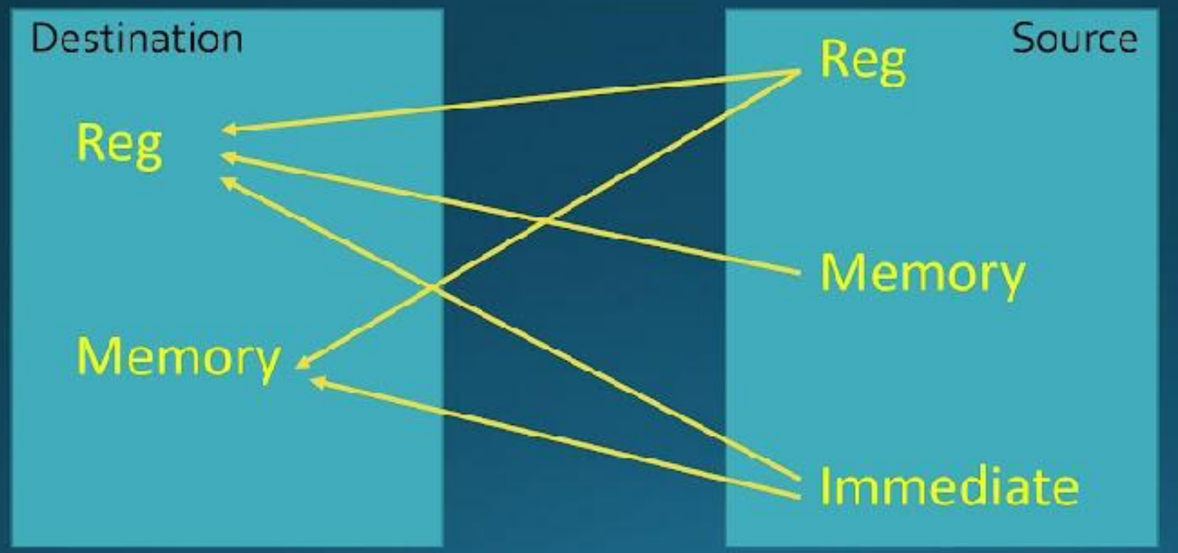
- In general, an instruction may have one of three effects:
  - No flags are altered
  - Specific flags are given values depending on the results of the instruction
  - Some flags may be altered, but their settings cannot be predicted
- No `mov` instruction changes any flag.

Operand Restrictions

- Operands must be same size
- Can't move from memory to memory
  - `mov nbr1, nbr2`
    illegal if *nbr1* and *nbr2* reference doublewords in memory
  - Instead use a register
    ```
    mov eax, nbr2
    mov nbr1, eax
    ```
- Can only move one byte, word, or doubleword at a time

Operand Restrictions

- mov <destination>, <source>
- several different opcodes
- Depends on type(s) of destination and source



Mov EDX, EBX
Mov sum, EAX

Mov EAX, sum

Mov EAX, -45
Mov sum, 36

Machine Code

- Depends on operand type(s), with several different opcodes used for `mov` instructions
- Word-size and doubleword-size instructions use same opcodes, but word-size instructions have 66 prefix byte
- Object and source code from listing file

```
B0 9B              mov   al, 155
66| B8 009B        mov   ax, 155
B8 0000009B        mov   eax, 155
```

Machine Code – Move a byte

Dest	Source	Opcode	#Bytes in Obj code
AL	Immediate byte	B0	2
CL	Immediate byte	B1	2
DL	Immediate byte	B2	2
BL	Immediate byte	B3	2
AH	Immediate byte	B4	2
CH	Immediate byte	B5	2
DH	Immediate byte	B6	2
BH	Immediate byte	B7	2
Register 8 (AH…DL)	Register 8 (AH…DL)	8A	2
AL	Memory byte direct address	A0	5
Register 8	Memory byte	8A	2+
Memory byte	Immediate byte	C6	3+
Memory byte, direct	AL	A2	5
Memory byte	Register 8	88	2+

Machine Code – Move a byte

```
41    000001CE E9              bytenum BYTE -23
42
43    00000000              .CODE
44    00000000              MoveByte PROC
45    00000000  B0 0A               mov AL, 10
46    00000002  B1 15               mov CL, 21
47    00000004  B2 1F               mov DL, 31
48    00000006  B3 29               mov BL, 41
49    00000008  B4 0A               mov AH, 10
50    0000000A  B5 15               mov CH, 21
51    0000000C  B6 1F               mov DH, 31
52    0000000E  B7 29               mov BH, 41
53
54    00000010  8A E7               mov AH, BH
55    00000012  8A D5               mov DL, CH
56
57    00000014  A0 000001CE R          mov AL, bytenum
58    00000019  8A 15 000001CE R       mov DL, bytenum
59
60    0000001F  C6 05 000001CE R       mov bytenum, 100
61          64
62    00000026  A2 000001CE R          mov bytenum, AL
63    0000002B  88 35 000001CE R       mov bytenum, DH
64
65    00000031  8D 1D 000001CE R       lea EBX, bytenum
66    00000037  88 33               mov BYTE PTR [EBX], DH
```