

Addressing and Basic Machine Language

Note Title

9/19/2007

On 32-bit systems (x86) logical address space is 64 GB

⇒ 36 bit addr

$$2^{36} = 2^6 \times 2^{30}$$

$$\approx 64 \times 10^9$$

segment - offset addressing

36 bits of addr on 32 bit arch

"segment" is 4 GB range of memory space
within segment only need 32 bits

individual segments used for:

code
data
stack

CS, DS, SS 32-bit reg. store segment addr

bits of seg. reg. are high order bits of seg. addr
low 4 bits are 0.

offset addresses identify specific mem. location
within segment

these are "addr" manipulated by program

Notation: pair is written segment : offset

e.g.

0A7C312E : 00007F38

logical addr. segment $\times 2^4$ + offset

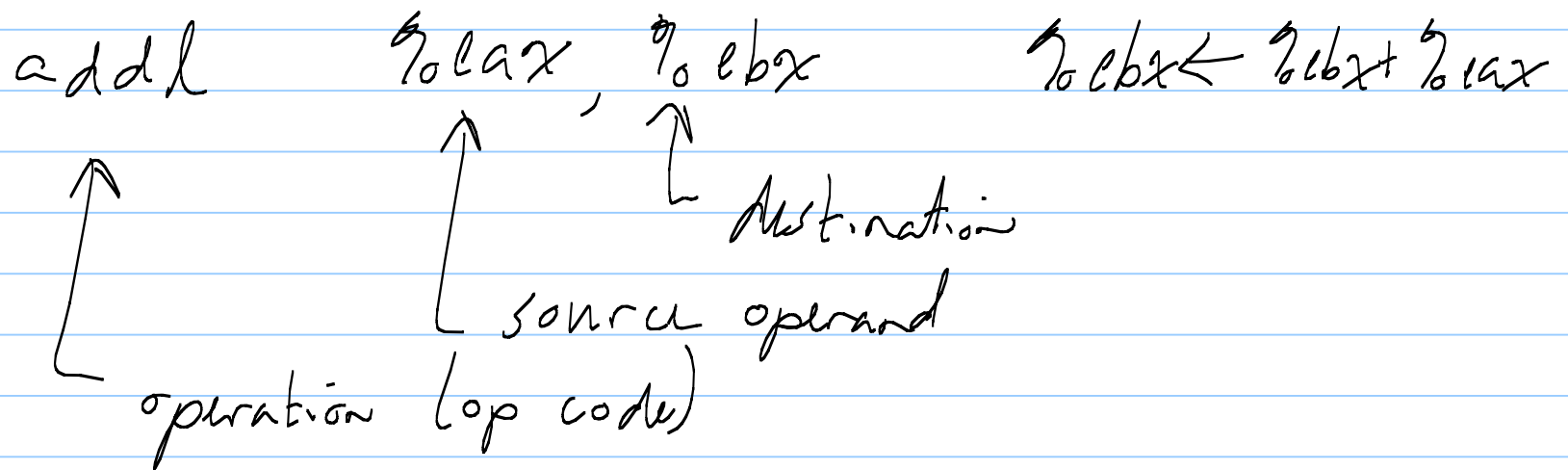
$$\begin{array}{r} 0A7C312E0 \\ 00007F3E \\ \hline 0A7C39218 \end{array} \leftarrow \text{logical addr.}$$

seg. reg. managed by OS

only worry about offset

Instructions must specify data for operations
addr modes

- register addressing - operand is in register



- immediate addr - operand is explicitly present

subl \$10, %edx $\%edx \leftarrow \%edx - 10$

↑ designates "immediate" operand

- direct/absolute addr - memory addr of operand is explicitly in inst.

addl 0x24, %ecx $\%ecx \leftarrow \%ecx + M[0x24]$

movl %edx, var1 $M[var1] \leftarrow \%edx$

direct offset addr - mem addr of operand
is sum of literal val (in inst) and
register (called offset)

movl var2(%esi), %ecx

$\%ecx \leftarrow M[\text{var2} + \%esi]$

Intel terms

base - displacement

"base" register

"displacement" is literal

Indirect addr memory addr stored in reg
register indirect

movl (%ebx), %eax $\%eax \leftarrow M[\%ebx]$

Indexed addr add two reg to form addr
addl (%ebx, %esi), %ecx

$\%ecx \leftarrow \%ecx + M[\%ebx + \%esi]$

base index

scaled indexed

scale reg by 1, 2, 4, 8

`movl $0x7f, (%ebx, %edi, 2)`

$M[\%ebx + \%edi * 2] \leftarrow 0x7f$

full form - literal (direct), base, index, scale

`movl 0x50(%ebx, %esi, 4), %eax`

$\%eax \leftarrow M[0x50 + \%ebx + \%esi * 4]$

Basic Prog. Structure

statements * directives

statements translate into machine inst.

directive is inst. for assembler

addl \$01, %eax

<op> l	32-bit	operands
<op> w	16-bit	"
<op> b	8-bit	"

standard form

label opcode operands comment

/# comments */

j comment

comment