

IA32 Instructions

<code>movl Src, Dest</code>	<code>Dest = Src</code>
<code>addl Src, Dest</code>	<code>Dest = Dest + Src</code>
<code>subl Src, Dest</code>	<code>Dest = Dest - Src</code>
<code>imull Src, Dest</code>	<code>Dest = Dest * Src</code>
<code>sall Src, Dest</code>	<code>Dest = Dest << Src</code>
<code>sarl Src, Dest</code>	<code>Dest = Dest >> Src</code>
<code>shrl Src, Dest</code>	<code>Dest = Dest >> Src</code>
<code>xorl Src, Dest</code>	<code>Dest = Dest ^ Src</code>
<code>andl Src, Dest</code>	<code>Dest = Dest & Src</code>
<code>orl Src, Dest</code>	<code>Dest = Dest Src</code>
<code>incl Dest</code>	<code>Dest = Dest + 1</code>
<code>decl Dest</code>	<code>Dest = Dest - 1</code>
<code>negl Dest</code>	<code>Dest = - Dest</code>
<code>notl Dest</code>	<code>Dest = ~ Dest</code>
<code>leal Src, Dest</code>	<code>Dest = address of Src</code>
<code>cmpl Src2, Src1</code>	Sets CCs Src1 – Src2
<code>testl Src2, Src1</code>	Sets CCs Src1 & Src2
<code>jmp label</code>	jump
<code>je label</code>	jump equal
<code>jne label</code>	jump not equal
<code>js label</code>	jump negative
<code>jns label</code>	jump non-negative
<code>jg label</code>	jump greater (signed)
<code>jge label</code>	jump greater or equal (signed)
<code>jl label</code>	jump less (signed)
<code>jle label</code>	jump less or equal (signed)
<code>ja label</code>	jump above (unsigned)
<code>jb label</code>	jump below (unsigned)
<code>push Src</code>	<code>Mem[%esp-4] = Src,</code> <code>%esp = %esp-4</code>
<code>pop Dest</code>	<code>Dest = Mem[%esp],</code> <code>%esp = %esp+4</code>
<code>call label</code>	<i>push address of next instruction,</i> <code>jmp label</code>
<code>ret</code>	<code>%eip = Mem[%esp],</code> <code>%esp = %esp+4</code>

Addressing Modes

Immediate	<code>\$val</code>	Val
	<code>movl \$17, %eax</code>	
Normal	<code>(R)</code>	<code>Mem[Reg[R]]</code>
	Register R specifies memory address	
	<code>movl (%ecx), %eax</code>	
Displacement	<code>D(R)</code>	<code>Mem[Reg[R]+D]</code>
	Register R specifies start of memory region	
	Constant displacement D specifies offset	
	<code>movl 8(%ebp), %edx</code>	
Indexed	<code>D(Rb, Ri, S)</code>	<code>Mem[Reg[Rb]+S*Reg[Ri]+ D]</code>
	D: Constant "displacement" 1, 2, or 4 bytes	
	Rb: Base register: Any of 8 integer registers	
	Ri: Index register:	
	S: Scale: 1, 2, 4, or 8	
	<code>movl 0x100(%ecx, %eax, 4), %edx</code>	

Condition Codes

CF Carry Flag
ZF Zero Flag
SF Sign Flag
OF Overflow Flag

Registers

`%eax`

`%ebx`

`%ecx`

`%edx`

`%edi`

`%esi`

`%ebp`

`%esp`