# IA32 Reference Sheet (GNU assembler format)

## IA32 instructions

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

## Addressing modes

- **Immediate**
  $val Val
  val: constant integer value
  `movl $17, %eax`

- **Normal**
  (R) Mem[Reg[R]]
  R: register R specifies memory address
  `movl (%ecx), %eax`

- **Displacement**
  D(R) Mem[Reg[R]+D]
  R: register specifies start of memory region
  D: constant displacement D specifies offset
  `movl 8(%ebp), %edx`

- **Indexed**
  D(Rb,Ri,S) Mem[Reg[Rb]+S*Reg[Ri]+D]
  D: constant displacement 1, 2, or 4 bytes
  Rb: base register: any of 8 integer registers
  Ri: index register: any, except `%esp`
  S: scale: 1, 2, 4, or 8
  `movl 0x100(%ecx,%eax,4), %edx`

## Instruction suffixes

`b` byte
`w` word (2 bytes)
`l` long (4 bytes)

## Condition codes

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

## Registers