

# Conjunto de instruções e arquitectura IA32

Tipo	Instrução	Efeito	Descrição
Transferência de Informação	mov? S, D	$D \leftarrow S$	Move (? = b,w,l)
	movsbl S, D	$D \leftarrow \text{SignExtend}(S)$	Move Sign-Extended Byte
	movzbl S, D	$D \leftarrow \text{ZeroExtend}(S)$	Move Zero-Extended Byte
	pushl S	$\%esp \leftarrow \%esp - 4; \text{Mem}[\%esp] \leftarrow S$	Push
	popl D	$D \leftarrow \text{Mem}[\%esp]; \%esp \leftarrow \%esp + 4$	Pop
	leal S, D	$D \leftarrow \&S$	Load Effective Address
Operações Aritméticas e Lógicas	incl D	$D \leftarrow D + 1$	Increment
	decl D	$D \leftarrow D - 1$	Decrement
	negl D	$D \leftarrow -D$	Negate
	notl D	$D \leftarrow \sim D$	Complement
	addl S, D	$D \leftarrow D + S$	Add
	subl S, D	$D \leftarrow D - S$	Subtract
	imull S, D	$D \leftarrow D * S$	32 bit Multiply
	xorl S, D	$D \leftarrow D \wedge S$	Exclusive-Or
	orl S, D	$D \leftarrow D \vee S$	Or
	andl S, D	$D \leftarrow D \& S$	And
	sall k, D	$D \leftarrow D \ll k$	Left Shift
	shll k, D	$D \leftarrow D \ll k$	Left Shift
	sarl k, D	$D \leftarrow D \gg k$	Arithmetic Right Shift
	shrl k, D	$D \leftarrow D \gg k$	Logical Right Shift
Teste	cmpl S <sub>2</sub> , S <sub>1</sub>	$(CF, ZF, SF, OF) \leftarrow S_1 - S_2$	Compare (? = b,w,l)
	testl S <sub>2</sub> , S <sub>1</sub>	$(CF, ZF, SF, OF) \leftarrow S_1 \& S_2$	Test (? = b,w,l)
Instruções de set	sete R <sub>8</sub>	$R_8 \leftarrow ZF$ (Sinónimo: setz R <sub>8</sub> )	Equal/Zero
	setne R <sub>8</sub>	$R_8 \leftarrow \sim ZF$ (Sinónimo: setnz R <sub>8</sub> )	Not Equal/Not Zero
	sets R <sub>8</sub>	$R_8 \leftarrow SF$	Negative
	setns R <sub>8</sub>	$R_8 \leftarrow \sim SF$	Non Negative
	setg R <sub>8</sub>	$R_8 \leftarrow \sim(SF \wedge OF) \& \sim ZF$ (Sinónimo: setnle R <sub>8</sub> )	Greater (signed >)
	setge R <sub>8</sub>	$R_8 \leftarrow \sim(SF \wedge OF)$ (Sinónimo: setnl R <sub>8</sub> )	Greater or equal (signed >=)
	setl R <sub>8</sub>	$R_8 \leftarrow SF \wedge OF$ (Sinónimo: setnge R <sub>8</sub> )	Less (signed <)
	setle R <sub>8</sub>	$R_8 \leftarrow (SF \wedge OF) \vee ZF$ (Sinónimo: setng R <sub>8</sub> )	Less or equal (signed <=)
	seta R <sub>8</sub>	$R_8 \leftarrow \sim CF \& \sim ZF$ (Sinónimo: setnbe R <sub>8</sub> )	Above (unsigned >)
	setae R <sub>8</sub>	$R_8 \leftarrow \sim CF$ (Sinónimo: setnb R <sub>8</sub> )	Above or equal (unsigned >=)
Instruções de salto	setb R <sub>8</sub>	$R_8 \leftarrow CF$ (Sinónimo: setnae R <sub>8</sub> )	Below (unsigned <)
	setbe R <sub>8</sub>	$R_8 \leftarrow CF \& \sim ZF$ (Sinónimo: setna R <sub>8</sub> )	Below or equal (unsigned <=)
	jmp Label	$\%eip \leftarrow \text{Label}$	Unconditional jump
	jmp *D	$\%eip \leftarrow *D$	Indirect unconditional jump
	je Label	Jump if ZF (Sinónimo: jz)	Zero/Equal
	jne Label	Jump if $\sim ZF$ (Sinónimo: jnz)	Not Zero/Not Equal
	js Label	Jump if SF	Negative
	jns Label	Jump if $\sim SF$	Not Negative
	jg Label	Jump if $\sim(SF \wedge OF) \& \sim ZF$ (Sinónimo: jnle)	Greater (signed >)
	jge Label	Jump if $\sim(SF \wedge OF)$ (Sinónimo: jnl)	Greater or equal (signed >=)
Invocação de Procedimentos	jl Label	Jump if $SF \wedge OF$ (Sinónimo: jnge)	Less (signed <)
	jle Label	Jump if $(SF \wedge OF) \vee ZF$ (Sinónimo: jng)	Less or equal (signed <=)
	ja Label	Jump if $\sim CF \& \sim ZF$ (Sinónimo: jnbe)	Above (unsigned >)
	jae Label	Jump if $\sim CF$ (Sinónimo: jnb)	Above or equal (unsigned >=)
	jb Label	Jump if CF (Sinónimo: jnae)	Below (unsigned <)
	jbe Label	Jump if $CF \& \sim ZF$ (Sinónimo: jna)	Below or equal (unsigned <=)
Invocação de Procedimentos	call Label	pushl %eip; %eip = Label	Procedure call
	call *Op	pushl %eip; %eip = *Op	Procedure call
	ret	popl %eip	Procedure return
	leave	movl %ebp, %esp; pop %ebp	Prepare stack for return

## Legenda:

**D** – destino [Reg | Mem]      **S** – fonte [Imm | Reg | Mem]  
**D** e **S** não podem ser ambos operandos em memória.

**R<sub>8</sub>** – destino Reg 8 bits

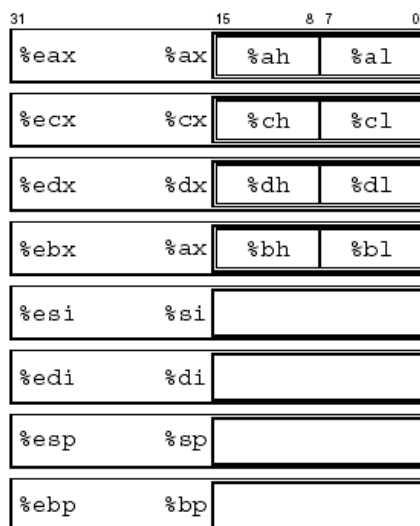


Ilustração 1- Conjunto de registos inteiros do IA32

Type	Form	Operand value	Name
Immediate	$\$ Imm$	$Imm$	Immediate
Register	$E_a$	$R[E_a]$	Register
Memory	$Imm$	$M[Imm]$	Absolute
Memory	$(E_a)$	$M[R[E_a]]$	Indirect
Memory	$Imm(E_b)$	$M[Imm + R[E_b]]$	Base + displacement
Memory	$(E_b, E_i)$	$M[R[E_b] + R[E_i]]$	Indexed
Memory	$Imm(E_b, E_i)$	$M[Imm + R[E_b] + R[E_i]]$	Indexed
Memory	$(, E_i, s)$	$M[R[E_i] \cdot s]$	Scaled indexed
Memory	$Imm(, E_i, s)$	$M[Imm + R[E_i] \cdot s]$	Scaled Indexed
Memory	$(E_b, E_i, s)$	$M[R[E_b] + R[E_i] \cdot s]$	Scaled indexed
Memory	$Imm(E_b, E_i, s)$	$M[Imm + R[E_b] + R[E_i] \cdot s]$	Scaled indexed

**Ilustração 2 - Modos de endereçamento IA32:** imediato, registo e valores em memória. O factor de escala  $s$  pode tomar o valor 1,2,4 ou 8;  $Imm$  pode ser uma constante de 0, 8, 16 ou 32 *bits*. O modo de endereçamento em memória reduz-se à forma  $Imm(E_b, E_i, s)$ , em que alguns dos campos podem não estar presentes.

**Tabela 1- Códigos de condições (flags)** . Descrevem atributos da última operação lógica ou aritmética realizada. Usadas para realizar saltos condicionais.

Símbolo	Nome	Descrição
CF	Carry Flag	A última operação gerou transporte.
ZF	Zero Flag	A última operação teve resultado zero
SF	Sign Flag	A última operação teve resultado negativo
OF	Overflow Flag	A última operação causou overflow em complemento para dois.