Tipo	Instrução	Efeito	Descrição
Transferência de Informação	mov? S, D	D←S	Move (? = b,w,l)
	movsbl S, D	D←SignExtend(S)	Move Sign-Extended Byte
	movzbl S, D	D←ZeroExtend(S)	Move Zero-Extended Byte
	pushl S	%esp ← %esp - 4; Mem[%esp] ← S	Push
	popl D	D←Mem[%esp]; %esp ←%esp+ 4	Pop
	leal S, D	D← &S	Load Effective Address
Operações Aritméticas e Lógicas	incl D	D← D +1	Increment
	decl D	D← D –1	Decrement
	negl D	D← -D	Negate
	notl D	D← ˜D	Complement
	addl S, D	D← D + S	Add
	subl S, D	D← D – S	Subtract
	imull S, D	D← D * S	32 bit Multiply
	xorl S, D	D← D ^ S	Exclusive-Or
	orl S, D	D← D | S	Or
	andl S, D	D← D & S	And
	sall k, D	D← D << k	Left Shift
	shll k, D	D← D << k	Left Shift
	sarl k, D	D← D >> k	Arithmetic Right Shift
	shrl k, D	D← D >> k	Logical Right Shift
	imull S	%edx : %eax ← S × %eax	Signed 64 bit Multiply
	mull S	%edx : %eax ← S × %eax	Unsigned 64 bit Multiply
	cltd	%edx : %eax ← SignExtend(%eax)	Convert to Quad Word
	idivl S	%edx ← %edx : %eax mod S; %eax ← %edx:%eax ÷ S	Signed Divide
	divl S	%edx ← %edx : %eax mod S; %eax ← %edx : %eax ÷ S	Unsigned Divide
Teste	cmp? S_2, S_1	(CF, ZF, SF, OF) ← S_1 – S_2	Compare (? = b,w,l)
	test? S_2, S_1	(CF, ZF, SF, OF) ← S_1 & S_2	Test (? = b,w,l)
Instruções de *set*	sete R_8	R_8 ← ZF (Sinónimo: setz R_8)	Equal/Zero
	setne R_8	R_8 ← ~ZF (Sinónimo: setnz R_8)	Not Equal/Not Zero
	sets R_8	R_8 ← SF	Negative
	setns R_8	R_8 ← ~SF	Non Negative
	setg R_8	R_8 ← ~(SF^OF) & ~ZF (Sinónimo: setnle R_8)	Greater (signed >)
	setge R_8	R_8 ← ~(SF^OF) (Sinónimo: setnl R_8)	Greater or equal (signed >=)
	setl R_8	R_8 ← SF^OF (Sinónimo: setnge R_8)	Less (signed <)
	setle R_8	R_8 ← (SF^OF) | ZF (Sinónimo: setng R_8)	Less or equal (signed <=)
	seta R_8	R_8 ← ~CF & ~ZF (Sinónimo: setnbe R_8)	Above (unsigned >)
	setae R_8	R_8 ← ~CF (Sinónimo: setnb R_8)	Above or equal (unsigned >=)
	setb R_8	R_8 ← CF (Sinónimo: setnae R_8)	Below (unsigned <)
	setbe R_8	R_8 ← CF & ~ZF (Sinónimo: setna R_8)	Below or equal (unsigned <=)
Instruções de salto	jmp *Label*	%eip ← *Label*	Unconditional jump
	jmp *D	%eip ← *D	Indirect unconditional jump
	je *Label*	Jump if ZF (Sinónimo: jz)	Zero/Equal
	jne *Label*	Jump if ~ZF (Sinónimo: jnz)	Not Zero/Not Equal
	js *Label*	Jump if SF	Negative
	jns *Label*	Jump if ~SF	Not Negative
	jg *Label*	Jump if ~(SF ^OF) & ~ZF (Sinónimo: jnle)	Greater (signed >)
	jge *Label*	Jump if ~(SF^OF) (Sinónimo: jnl)	Greater or equal (signed >=)
	jl *Label*	Jump if SF^OF (Sinónimo: jnge)	Less (signed <)
	jle *Label*	Jump if (SF^OF) | ZF (Sinónimo: jng)	Less or equal (signed <=)
	ja *Label*	Jump if ~CF & ~ZF (Sinónimo: jnbe)	Above (unsigned >)
	jae *Label*	Jump if ~CF (Sinónimo: jnb)	Above or equal (unsigned >=)
	jb *Label*	Jump if CF (Sinónimo: jnae)	Below (unsigned <)
	jbe *Label*	Jump if CF & ~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] **R_8** – destino Reg 8 *bits*
D e **S** não podem ser ambos operandos em memória.

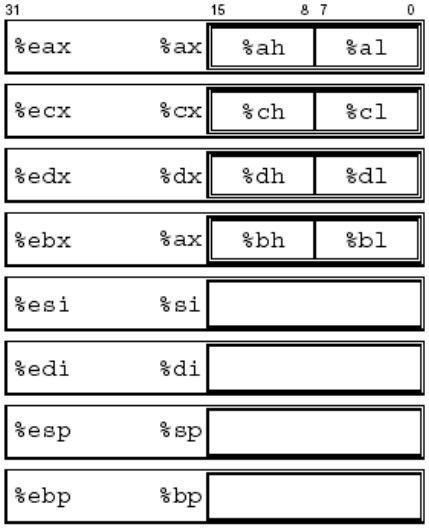

Ilustração 1- Conjunto de registos inteiros do IA32

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*(Eb, Ei, 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.