Instruction set do IA-32

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 ˆ 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? S2, S1	(CF, ZF, SF, OF) ← S1 – S2	Compare (? = b,w,l)
	test? S2, S1	(CF, ZF, SF, OF) ← S1 & S2	Test (? = b,w,l)
Instruções de set	sete R8	R_8 ← ZF (Sinónimo: setz R8)	Equal/Zero
	setne R8	R_8 ← ~ZF (Sinónimo: setnz R8)	Not Equal/Not Zero
	sets R8	R_8 ← SF	Negative
	setns R8	R_8 ← ~SF	Non Negative
	setg R8	R_8 ← ~(SF^OF) & ~ZF (Sinónimo: setnle R8)	Greater (signed >)
	setge R8	R_8 ← ~(SF^OF) (Sinónimo: setnl R8)	Greater or equal (signed >=)
	setl R8	R_8 ← SF^OF (Sinónimo: setnge R8)	Less (signed <)
	setle R8	R_8 ← (SF^OF) | ZF (Sinónimo: setng R8)	Less or equal (signed <=)
	seta R8	R_8 ← ~CF & ~ZF (Sinónimo: setnbe R8)	Above (unsigned >)
	setae R8	R_8 ← ~CF (Sinónimo: setnb R8)	Above or equal (unsigned >=)
	setb R8	R_8 ← CF (Sinónimo: setnae R8)	Below (unsigned <)
	setbe R8	R_8 ← CF & ~ZF (Sinónimo: setna R8)	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

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

Alberto Proença & Luís Paulo Santos, Mar-03