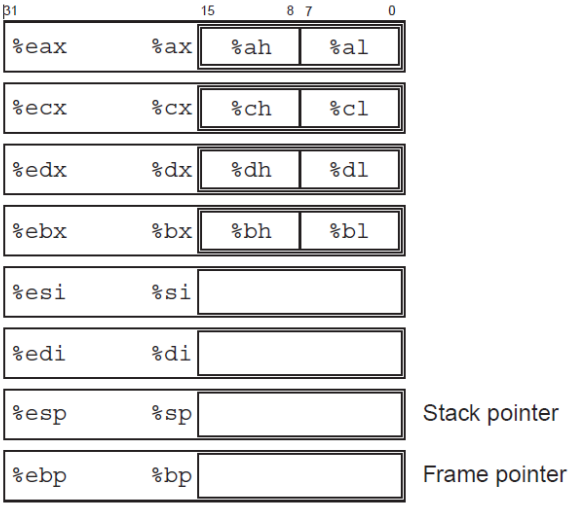
COMPUTADORES E PROGRAMAÇÃO

Instruções assembly IA32

Registradores do IA32:



Endereço de memória = I + R[Eb] + R[Ei] * s
I = deslocamento do tipo imediato (mas sem o $)
R[Eb] representa o valor armazenado no registrador base Eb
R[Ei] representa o valor armazenado no registrador de índice Ei
s é o fator de escala (1, 2, 4 ou 8), relacionado ao tamanho do tipo dos objetos da estrutura

S = source D = destination

Movimentação de dados:
movb S,D: S -> D (move um byte)
movw S,D: S -> D (move uma palavra de 16 bits)
movl S,D: S -> D (move uma palavra dupla de 32 bits)

Movimentação com extensão do sinal:
movsbw S,D: sinalEstendido(S) -> D (de byte para palavra)
movsbl S,D: sinalEstendido(S) -> D (de byte para palavra dupla)
movswl S,D: sinalEstendido(S) -> D (de palavra para palavra dupla)

Movimentação com extensão de 0s:
movzbw S,D: zeroEstendido(S) -> D (de byte para palavra)
movzbl S,D: zeroEstendido(S) -> D (de byte para palavra dupla)
movzwl S,D: zeroEstendido(S) -> D (de palavra para palavra dupla)

Movimentação de dados sobre a pilha:
pushl S: abre 4 posições na pilha fazendo %esp= %esp-4 e armazena o valor S (4 bytes) a partir do novo topo (M[%esp] S)
popl D: copia 4 bytes do topo para D (D M[%esp]) e atualiza ponteiro, liberando 4 bytes (%esp= %esp+4)

Operações unárias:
inc D: D+1 -> D (incremento de 1)
dec D: D-1 -> D (decremento de 1)
neg D: -D -> D (negativo do número)
not D: ~D -> D (complemento do número bit a bit)

Operações binárias:
add S, D: D + S -> D (adição)
sub S, D: D - S -> D (subtração)
imul S, D: D * S -> D (multiplicação, resultado em 32 bits)
xor S, D: D^S -> D ("ou-exclusivo" lógico bit a bit)
or S, D: D | S -> D ("ou" lógico bit a bit)
and S, D: D & S -> D ("e" lógico bit a bit)

Operações de deslocamento:
sal k, D: D << k -> D (deslocamento aritmético à esquerda)
shl k, D: D << k -> D (deslocamento lógico à esquerda = sal)
sar k, D: D >> k -> D (deslocamento aritmético à direita)
shr k, D: D >> k -> D (deslocamento lógico à direita)

Operação de endereço efetivo de carga:
leal S, D [&S -> D];

Operações Aritméticas Especiais:

Instrução	Efeito	Descrição
imull S	%edx:%eax <- S x %eax	Mult. completa (64 bits) com sinal
mull S	%edx:%eax <- S x %eax	Mult. completa (64 bits) sem sinal
cltd	%edx:%eax <- estende sinal(%eax)	Estende 32 bits para 64 bits
idivl S	%edx <- %edx:%eax mod S (resto) %eax <- %edx:%eax / S (quociente)	Divisão com sinal
divl S	%edx <- %edx:%eax mod S (resto) %eax <- %edx:%eax / S (quociente)	Divisão sem sinal

Códigos de condição:
1 CF (Carry Flag)
2 ZF (Zero Flag)
3 SF (Sign Flag)
4 OF (Overflow Flag)

Classe de instruções TEST e CMP:
test S1, S2 (testa S2 & S1)
cmp S1, S2 (testa S2 - S1)

Instruções SET:

Instrução	Sinônimo	Efeito	Condição causal
sete D	setz	D <- ZF	Equal/zero
setne D	setnz	D <- ~ZF	Not equal/not zero
sets D		D <- SF	Negative
setns D		D <- ~SF	Nonnegative
setg D	setnle	D <- ~(SF^OF) & ~ZF	Greater (signed >)
setge D	setnl	D <- ~(SF^OF)	Greater or equal (signed >=)
setl D	setnge	D <- (SF^OF)	Less (signed <)
setle D	setng	D <- (SF^OF) | ZF	Less or equal (signed <=)
seta D	setnbe	D <- ~CF & ~ZF	Above (unsigned >)
setae D	setnb	D <- ~CF	Above or equal (unsigned >=)
setb D	setnae	D <- CF	Below (unsigned <)
setbe D	setna	D <- CF | ZF	Below or equal (unsigned <=)

Principais instruções de desvio:

je (ou jz):	ZF	(igual/zero)
jne (ou jnz):	~ZF	(diferente/não-zero)
js:	SF	(negativo)
jns:	~SF	(não negativo)
jg (ou jnle):	~(SF^OF) & ~ZF	(maior com sinal)
jge (ou jnl):	~(SF^OF)	(maior ou igual com sinal)
jl (ou jnge):	SF^OF	(menor com sinal)
jle (ou jng):	(SF^OF) | ZF	(menor ou igual com sinal)