



Estrutura do tema ISA do IA-32

1. Desenvolvimento de programas no IA-32 em Linux
2. Acesso a operandos e operações
3. Suporte a estruturas de controlo
4. Suporte à invocação/regresso de funções
5. Análise comparativa: IA-32 vs. x86-64 e RISC (MIPS e ARM)
6. Acesso e manipulação de dados estruturados

Acesso a operandos no IA-32: sua localização e modos de acesso



Localização de operandos no IA-32

- valores de constantes (ou valores imediatos)
 - incluídos na instrução, i.e., no Reg. Instrução (IR)
- variáveis escalares
 - sempre que possível, em registos (inteiros/apont) / *fp* ; se não...
 - na memória (inclui *stack*)
- variáveis estruturadas
 - sempre na memória, em células contíguas

Modos de acesso a operandos no IA-32

- em instruções de transferência de informação
 - instrução mais comum: `movx`, sendo *x* o tamanho (*b*, *w*, *l*)
 - algumas instruções atualizam apontadores (por ex.: `push`, `pop`)
- em operações aritméticas/lógicas



• Transferência simples

movl Source, Dest

- move um valor de 4 bytes (“*long*”)
- instrução mais comum em código IA-32

• Tipos de operandos

- imediato: valor constante do tipo inteiro
 - como a constante em C, mas com prefixo ‘\$’
 - ex.: \$0x400, \$-533
 - codificado com 4 bytes (em **movl**)
- em registo: um de 8 registos inteiros
 - mas... %esp e %ebp estão reservados...
 - e outros poderão ser usados implicitamente...
- em memória: 4 bytes consecutivos de memória (em **movl**)
 - vários modos de especificar a sua localização (o endereço)...

%eax
%edx
%ecx
%ebx
%esi
%edi
%esp
%ebp

Análise da localização dos operandos na instrução **movl**



	Fonte	Destino	Equivalente em C
movl	Imm	Reg <code>movl \$0x4,%eax</code>	<code>temp = 0x4;</code>
		Mem <code>movl \$-147, (%eax)</code>	<code>*p = -147;</code>
	Reg	Reg <code>movl %eax,%edx</code>	<code>temp2 = temp1;</code>
		Mem <code>movl %eax, (%edx)</code>	<code>*p = temp;</code>
	Mem	Reg <code>movl (%eax), %edx</code>	<code>temp = *p;</code>
		Mem não é possível no IA-32 efetuar transferências memória-memória com uma só instrução	



- **Indireto (*normal*)** **(R)** $\text{Mem}[\text{Reg}[\text{R}]]$
 - conteúdo do registo **R** especifica o endereço de memória

```
movl (%ecx), %eax
```
- **Deslocamento** **D(R)** $\text{Mem}[\text{Reg}[\text{R}] + \text{D}]$
 - conteúdo do registo **R** especifica início da região de memória
 - deslocamento c^{te} **D** especifica distância do início (em *bytes*)

```
movl -8(%ebp), %edx
```

**Exemplo de utilização de modos simples
de endereçamento à memória no IA-32 (1)**



```
void swap(int *xp, int *yp)
{
    int t0 = *xp;
    int t1 = *yp;
    *xp = t1;
    *yp = t0;
}
```

```
swap:
    pushl %ebp
    movl  %esp,%ebp
    pushl %ebx
}
Arranque

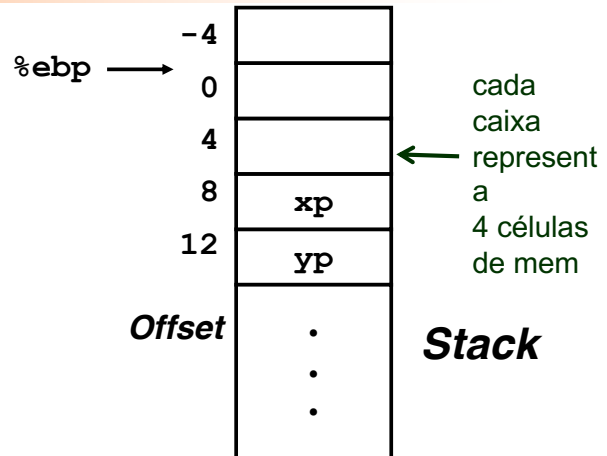
    movl 12(%ebp), %ecx
    movl 8(%ebp), %edx
    movl (%ecx), %eax
    movl (%edx), %ebx
    movl %eax, (%edx)
    movl %ebx, (%ecx)
}
Corpo

    movl -4(%ebp), %ebx
    movl %ebp, %esp
    popl %ebp
    ret
}
Término
```

Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (2)



```
void swap(int *xp, int *yp)
{
    int t0 = *xp;
    int t1 = *yp;
    *xp = t1;
    *yp = t0;
}
```



Registo	Variável
%ecx	yp
%edx	xp
%eax	t1
%ebx	t0

Corpo

```
movl 12(%ebp), %ecx    # ecx = yp
movl 8(%ebp), %edx     # edx = xp
movl (%ecx), %eax      # eax = *yp (t1)
movl (%edx), %ebx      # ebx = *xp (t0)
movl %eax, (%edx)      # *xp = eax
movl %ebx, (%ecx)      # *yp = ebx
```

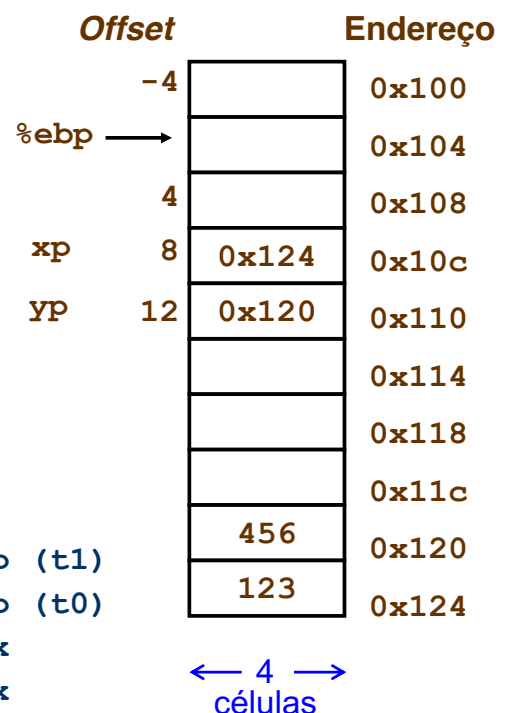
Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (3)



%eax	
%edx	
%ecx	
%ebx	
%esi	
%edi	
%esp	
%ebp	0x104

Corpo

```
movl 12(%ebp), %ecx    # ecx = yp
movl 8(%ebp), %edx     # edx = xp
movl (%ecx), %eax      # eax = *yp (t1)
movl (%edx), %ebx      # ebx = *xp (t0)
movl %eax, (%edx)      # *xp = eax
movl %ebx, (%ecx)      # *yp = ebx
```



Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (4)



Corpo

```

movl 12(%ebp), %ecx # ecx = yp
movl 8(%ebp), %edx  # edx = xp
movl (%ecx), %eax   # eax = *yp (t1)
movl (%edx), %ebx   # ebx = *xp (t0)
movl %eax, (%edx)   # *xp = eax
movl %ebx, (%ecx)   # *yp = ebx
    
```

%eax	
%edx	
%ecx	0x120
%ebx	
%esi	
%edi	
%esp	
%ebp	0x104

Offset	Endereço
-4	0x100
%ebp →	0x104
4	0x108
xp 8	0x10c
yp 12	0x110
	0x114
	0x118
	0x11c
	0x120
	0x124

← 4 →
células

Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (5)



Corpo

```

movl 12(%ebp), %ecx # ecx = yp
movl 8(%ebp), %edx  # edx = xp
movl (%ecx), %eax   # eax = *yp (t1)
movl (%edx), %ebx   # ebx = *xp (t0)
movl %eax, (%edx)   # *xp = eax
movl %ebx, (%ecx)   # *yp = ebx
    
```

%eax	
%edx	0x124
%ecx	0x120
%ebx	
%esi	
%edi	
%esp	
%ebp	0x104

Offset	Endereço
-4	0x100
%ebp →	0x104
4	0x108
xp 8	0x10c
yp 12	0x110
	0x114
	0x118
	0x11c
	0x120
	0x124

← 4 →
células

Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (6)



Corpo

```
movl 12(%ebp), %ecx # ecx = yp
movl 8(%ebp), %edx  # edx = xp
movl (%ecx), %eax   # eax = *yp (t1)
movl (%edx), %ebx   # ebx = *xp (t0)
movl %eax, (%edx)   # *xp = eax
movl %ebx, (%ecx)   # *yp = ebx
```

%eax	456
%edx	0x124
%ecx	0x120
%ebx	
%esi	
%edi	
%esp	
%ebp	0x104

Offset	Endereço
-4	0x100
%ebp →	0x104
4	0x108
xp 8	0x10c
yp 12	0x110
	0x114
	0x118
	0x11c
	0x120
	0x124

← 4 →
células

Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (7)



Corpo

```
movl 12(%ebp), %ecx # ecx = yp
movl 8(%ebp), %edx  # edx = xp
movl (%ecx), %eax   # eax = *yp (t1)
movl (%edx), %ebx   # ebx = *xp (t0)
movl %eax, (%edx)   # *xp = eax
movl %ebx, (%ecx)   # *yp = ebx
```

%eax	456
%edx	0x124
%ecx	0x120
%ebx	123
%esi	
%edi	
%esp	
%ebp	0x104

Offset	Endereço
-4	0x100
%ebp →	0x104
4	0x108
xp 8	0x10c
yp 12	0x110
	0x114
	0x118
	0x11c
	0x120
	0x124

← 4 →
células

Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (8)



Corpo

```

movl 12(%ebp), %ecx # ecx = yp
movl 8(%ebp), %edx  # edx = xp
movl (%ecx), %eax   # eax = *yp (t1)
movl (%edx), %ebx   # ebx = *xp (t0)
movl %eax, (%edx)  # *xp = eax
movl %ebx, (%ecx)   # *yp = ebx
  
```

%eax	456
%edx	0x124
%ecx	0x120
%ebx	123
%esi	
%edi	
%esp	
%ebp	0x104

Offset		Endereço
-4		0x100
%ebp →		0x104
4		0x108
xp 8	0x124	0x10c
yp 12	0x120	0x110
		0x114
		0x118
		0x11c
	456	0x120
	456	0x124

← 4 →
células

Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (9)



Corpo

```

movl 12(%ebp), %ecx # ecx = yp
movl 8(%ebp), %edx  # edx = xp
movl (%ecx), %eax   # eax = *yp (t1)
movl (%edx), %ebx   # ebx = *xp (t0)
movl %eax, (%edx)   # *xp = eax
movl %ebx, (%ecx)  # *yp = ebx
  
```

%eax	456
%edx	0x124
%ecx	0x120
%ebx	123
%esi	
%edi	
%esp	
%ebp	0x104

Offset		Endereço
-4		0x100
%ebp →		0x104
4		0x108
xp 8	0x124	0x10c
yp 12	0x120	0x110
		0x114
		0x118
		0x11c
	123	0x120
	456	0x124

← 4 →
células



- **Indirecto** (R) $\text{Mem}[\text{Reg}[\text{R}]] \dots$
- **Deslocamento** D(R) $\text{Mem}[\text{Reg}[\text{R}] + \text{D}] \dots$
- **Indexado** D(Rb,Ri,S) $\text{Mem}[\text{Reg}[\text{Rb}] + \text{S} * \text{Reg}[\text{Ri}] + \text{D}]$

D: Deslocamento constante de 1, 2, ou 4 bytes

Rb: Registo base: quaisquer dos 8 Reg Int

Ri: Registo indexação: qualquer, exceto %esp

S: Scale: 1, 2, 4, ou 8

Casos particulares:

(Rb,Ri) $\text{Mem}[\text{Reg}[\text{Rb}] + \text{Reg}[\text{Ri}]]$

D(Rb,Ri) $\text{Mem}[\text{Reg}[\text{Rb}] + \text{Reg}[\text{Ri}] +$

D]

(Rb,Ri,S) $\text{Mem}[\text{Reg}[\text{Rb}] + \text{S} * \text{Reg}[\text{Ri}]]$

AJProença, Sistemas de Computação, UMinho, 2019/20

15

Exemplo de instrução do IA-32 apenas para cálculo do apontador para um operando (1)



leal Src, Dest

- **Src** contém a expressão para cálculo do endereço
- **Dest** vai receber o resultado do cálculo da expressão
- nota: lea => load effective address

• **Tipos de utilização desta instrução:**

- cálculo de um endereço de memória (sem aceder à memória)
 - Ex.: tradução de `p = &x[i];`
- cálculo de expressões aritméticas do tipo

$$a = x + k * y \quad \text{para } k = 1, 2, 4, \text{ ou } 8$$

• **Exemplos ...**

Exemplo de instrução do IA-32 apenas para cálculo do apontador para um operando (2)



leal Source, %eax

%edx	0xf000
%ecx	0x100

Source	Expressão	-> %eax
0x8 (%edx)	0xf000 + 0x8	0xf008
(%edx, %ecx)	0xf000 + 0x100	0xf100
(%edx, %ecx, 4)	0xf000 + 4*0x100	0xf400
0x80(, %edx, 2)	2*0xf000 + 0x80	0x1e080

Instruções de transferência de informação no IA-32



movx	S, D	$D \leftarrow S$	Move (<u>b</u> yte, <u>w</u> ord, <u>l</u> ong-word)
movsbl	S, D	$D \leftarrow \text{SignExtend}(S)$	Move Sign-Extended Byte
movzbl	S, D	$D \leftarrow \text{ZeroExtend}(S)$	Move Zero-Extended Byte
push	S	$\%esp \leftarrow \%esp - 4; \text{Mem}[\%esp] \leftarrow S$	Push
pop	D	$D \leftarrow \text{Mem}[\%esp]; \%esp \leftarrow \%esp + 4$	Pop
lea	S, D	$D \leftarrow \&S$	Load Effective Address

D – destino [Reg | Mem] **S** – fonte [Imm | Reg | Mem]
D e **S** não podem ser ambos operandos em memória no IA-32



inc	D	$D \leftarrow D + 1$	Increment
dec	D	$D \leftarrow D - 1$	Decrement
neg	D	$D \leftarrow -D$	Negate
not	D	$D \leftarrow \sim D$	Complement
add	S, D	$D \leftarrow D + S$	Add
sub	S, D	$D \leftarrow D - S$	Subtract
imul	S, D	$D \leftarrow D * S$	32 bit Multiply
and	S, D	$D \leftarrow D \& S$	And
or	S, D	$D \leftarrow D S$	Or
xor	S, D	$D \leftarrow D \wedge S$	Exclusive-Or
shl	k, D	$D \leftarrow D \ll k$	Left Shift
sar	k, D	$D \leftarrow D \gg k$	Arithmetic Right Shift
shr	k, D	$D \leftarrow D \gg k$	Logical Right Shift