

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, x86-64 e MIPS (RISC)
6. Acesso e manipulação de dados estruturados

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

Análise de uma instrução de transferência de informação

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 o endereço...

<code>%eax</code>
<code>%edx</code>
<code>%ecx</code>
<code>%ebx</code>
<code>%esi</code>
<code>%edi</code>
<code>%esp</code>
<code>%ebp</code>

Análise da localização dos operandos na instrução `movl`

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

Modos de endereçamento à memória no IA-32 (1)

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

```
movl (%ecx), %eax
```
- **Deslocamento D(R)** $\text{Mem}[\text{Reg}[R] + D]$
 - conteúdo do registro **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
    movl 12(%ebp), %ecx
    movl 8(%ebp), %edx
    movl (%ecx), %eax
    movl (%edx), %ebx
    movl %eax, (%edx)
    movl %ebx, (%ecx)
    movl -4(%ebp), %ebx
    movl %ebp, %esp
    popl %ebp
    ret
```

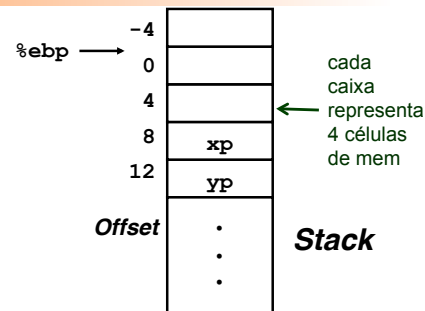
Arranque

Corpo

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;
}
```

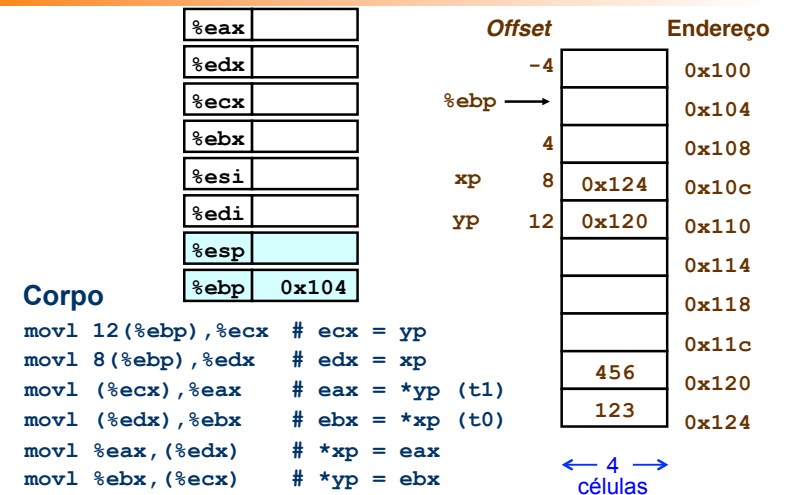


Registo	Variável
%ecx	yp
%edx	xp
%eax	t1
%ebx	t0

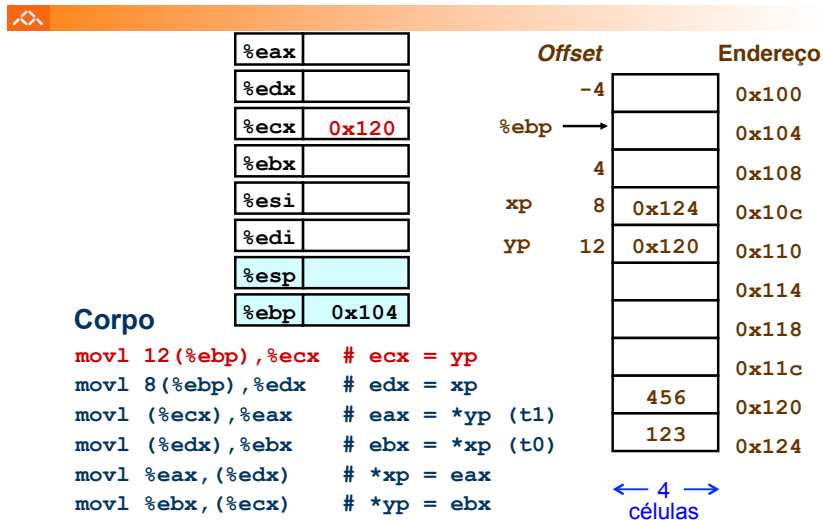
Corpo

```
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
```

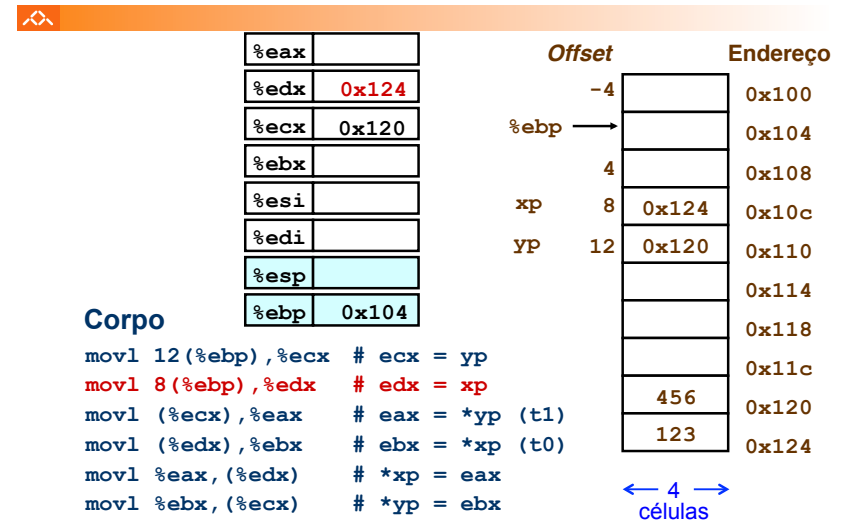
Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (3)



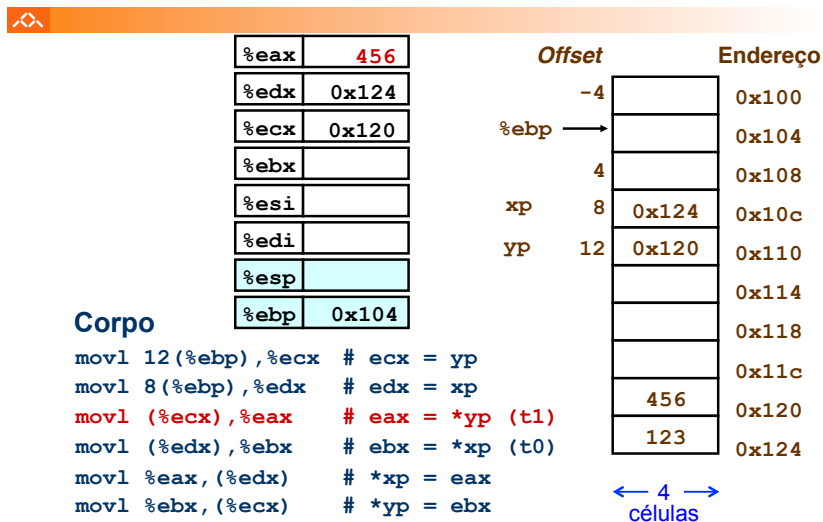
Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (4)



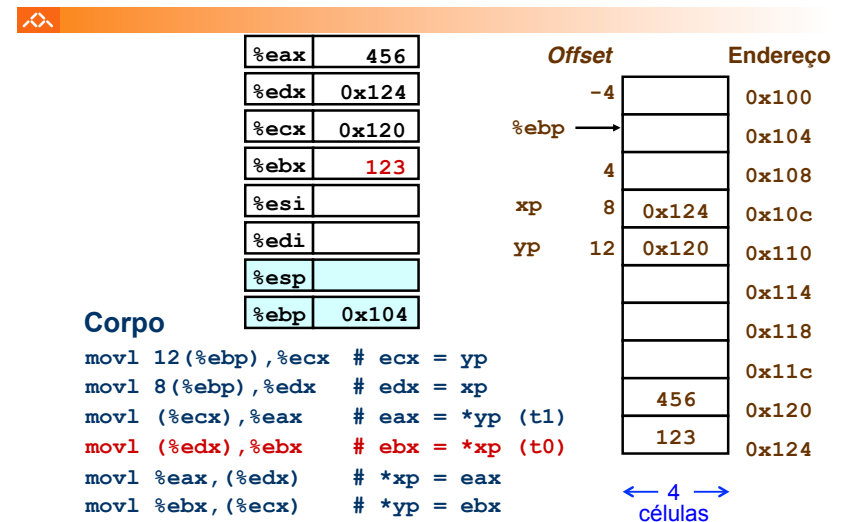
Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (5)



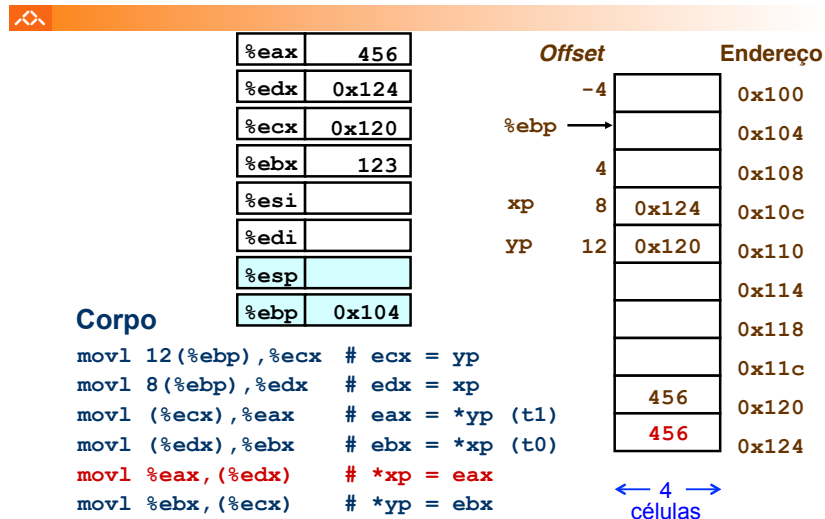
Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (6)



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



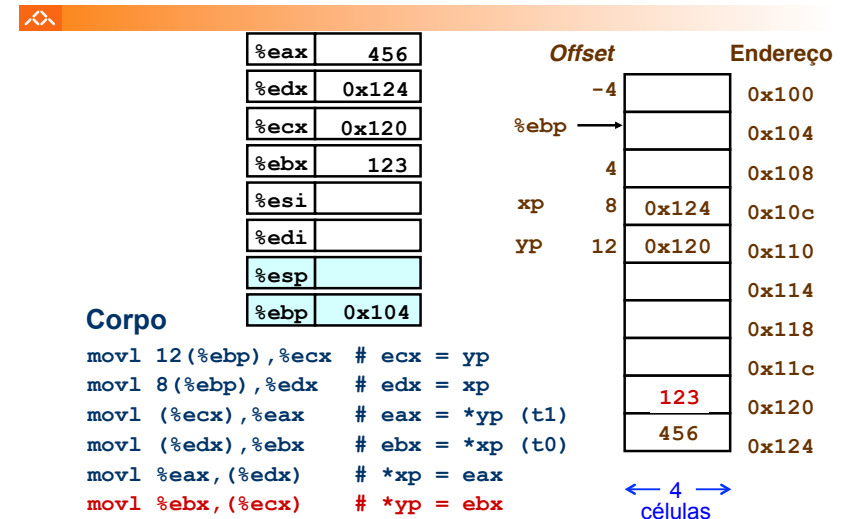
Exemplo de utilização de modos simples de endereçamento à memória no IA-32 (8)



AJProença, Sistemas de Computação, UMinho, 2018/19

13

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



AJProença, Sistemas de Computação, UMinho, 2018/19

14

Modos de endereçamento à memória no IA-32 (2)

- **Indirecto** (R) Mem[Reg[R]] ...
- **Deslocamento** D(R) Mem[Reg[R] + D] ...
- **Indexado** D(Rb,Ri,S) Mem[Reg[Rb] + S*Reg[Ri] + 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)	Mem[Reg[Rb] + Reg[Ri]]
D(Rb,Ri)	Mem[Reg[Rb] + Reg[Ri] + D]
(Rb,Ri,S)	Mem[Reg[Rb] + S*Reg[Ri]]

AJProença, Sistemas de Computação, UMinho, 2018/19

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 para k = 1, 2, 4, ou 8
- **Exemplos ...**

AJProença, Sistemas de Computação, UMinho, 2018/19

16

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 ← S Move (byte, word, long-word)
movsbl S,D D ← SignExtend(S) Move Sign-Extended Byte
movzbl S,D D ← ZeroExtend(S) Move Zero-Extended Byte
push S %esp ← %esp - 4; Mem[%esp] ← S Push
pop D D ← Mem[%esp]; %esp ← %esp + 4 Pop
lea S,D D ← &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

Operações aritméticas e lógicas no IA-32

inc D D ← D + 1 Increment
dec D D ← D - 1 Decrement
neg D D ← -D Negate
not D D ← ~D Complement
add S,D D ← D + S Add
sub S,D D ← D - S Subtract
imul S,D D ← D * S 32 bit Multiply
and S,D D ← D & S And
or S,D D ← D | S Or
xor S,D D ← D ^ S Exclusive-Or
shl k,D D ← D << k Left Shift
sar k,D D ← D >> k Arithmetic Right Shift
shr k,D D ← D >> k Logical Right Shift