

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/retorno de funções
5. Acesso e manipulação de dados estruturados
6. Análise comparativa: IA-32 (CISC) e MIPS (RISC)

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 1, 2, ou 4 bytes
- 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
 - 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>	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 IA32 efetuar transferências memória-memória com uma só instrução	

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

- **Indirecto (normal) (R)** Mem [Reg [R]]
 - registo R especifica o endereço de memória

```
movl (%ecx), %eax
```
- **Deslocamento D(R)** Mem [Reg [R] +D]
 - registo R especifica início da região de memória
 - deslocamento constante D especifica distância do início

```
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
```

Stack

Offset

–4

0

4

8

12

•

•

•

← cada caixa representa 4 células de mem

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
```

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

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

%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
456	0x120
123	0x124

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 (5)

%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
456	0x120
123	0x124

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 (6)

%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
456	0x120
123	0x124

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 (7)

%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
456	0x120
123	0x124

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 (8)

%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
	456
	0x120
	456
	0x124

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 (9)

%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
	123
	0x120
	456
	0x124

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
    
```

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]]

Exemplo de instrução do IA-32 apenas para cálculo do endereço efectivo do operando (1)

- leal Src, Dest**
- **Src** contém a expressão para cálculo do endereço
 - **Dest** vai receber o resultado do cálculo da expressão
- Tipos de utilização desta instrução:
 - cálculo de um endereço sem acesso à memória
 - Ex.: tradução de p = &x[i];
 - cálculo de expressões aritméticas do tipo **x + k*y** para k = 1, 2, 4, or 8
 - Exemplo ...

Exemplo de instrução do IA-32 apenas para cálculo do endereço efectivo do operando (2)

Instruções de transferência de informação no IA-32

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

movx S, D $D \leftarrow S$ Move (byte, word, long-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

Operações aritméticas e lógicas 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