Estrutura do tema ISC

1. Representação de informação num computador
2. Organização e estrutura interna dum computador
3. Execução de programas num computador
4. O processador e a memória num computador
5. Da comunicação de dados às redes
6. Evolução da tecnologia e da eficiência

Componentes (físicos) a analisar:

– a unidade de processamento / o processador:
  - ➢o nível ISA (*Instruction Set Architecture*): tipos e formatos de instruções, acesso a operandos, ...
  - CISC *versus* RISC
  - paralelismo no processador: *pipeline*, super-escalaridade, ...
  - paralelismo fora do processador: *on-chip* e *off-chip*

– a hierarquia de memória:
  cache, memória virtual, ...

– periféricos:
  - interfaces humano-computador (HCI)
  - arquivo de informação
  - comunicações (no tema 5...)

Ex. de código C

```
int sum(int x, int y)
{
  int t = x+y;
  return t;
}
```

Mesmo código em *assembly*

```
_sum:
    pushl   %ebp
    movl    %esp,%ebp
    movl    12(%ebp),%eax
    addl    8(%ebp),%eax
    movl    %ebp,%esp
    popl    %ebp
    ret
```

- operações num processador?
- como aceder a operandos?
- registos visíveis ao programador?
- tipos de instruções presentes num processador?
- formatos de instruções em linguagem máquina?
- instruções de *input/output* ?
- escalares multi-*byte* em memória?

- operações num processador?
- como aceder a operandos?
- registos visíveis ao programador?

Operações lógicas/aritméticas num processador

– operações mais comuns:
  - lógicas: `not`, `and`, `or`, `xor`, ...
  - aritméticas: `inc`/`dec`, `neg`, `add`, `sub`, `mul`, ...

– nº de operandos em cada operação
  - 3-operandos (RISC, ...)
  - 2-operandos (IA-32, ...)
  - 1-operando (microcontroladores, ...)
  - 0-operandos (*stack-machine*, ...)

– localização dos operandos
  - variáveis escalares (registos...)
  - variáveis estruturadas (memória...)

Modos de aceder a operandos

– em arquiteturas RISC
- em operações aritméticas/lógicas: operandos sempre em registo
- em *load/store*:1 ou 2 modos de especificar o endereço de memória

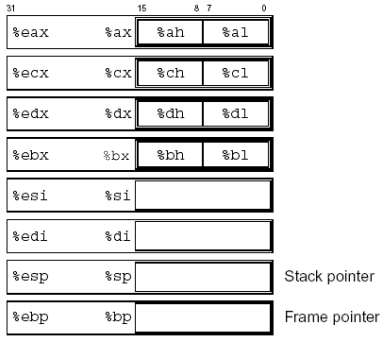
– em CISC, exemplo: IA-32 *(Intel Architecture 32-bits)*

Type	Form	Operand value	Name
Immediate	$*Imm*	*Imm*	Immediate
Register	E_a	R[E_a]	Register
Memory	*Imm*	M[*Imm*]	Absolute
Memory	(E_a)	M[R[E_a]]	Indirect
Memory	*Imm* (E_b)	M[*Imm* + R[E_b]]	Base + displacement
Memory	(E_b, E_i)	M[R[E_b] + R[E_i]]	Indexed
Memory	*Imm* (E_b, E_i)	M[*Imm* + R[E_b] + R[E_i]]	Indexed
Memory	(, E_i , s)	M[R[E_i] · s]	Scaled indexed
Memory	*Imm* (, E_i , s)	M[*Imm* + R[E_i] · s]	Scaled Indexed
Memory	(E_b, E_i , s)	M[R[E_b] + R[E_i] · s]	Scaled indexed
Memory	*Imm* (E_b, E_i , s)	M[*Imm* + R[E_b] + R[E_i] · s]	Scaled indexed

Registos visíveis ao programador (inteiros)

– em arquiteturas RISC: 32 registos genéricos...

– no IA-32:



Tipos de instruções presentes num processador

– transferência de informação
- de/para registos/memória, ...

– operações aritméticas e lógicas
- soma, subtração, multiplicação, divisão, ...
- AND, OR, NOT, XOR, comparação, ...
- deslocamento de bits, ...

– controlo do fluxo de execução
- para apoio a estruturas de controlo
- para apoio à invocação de procedimentos/funções

– outras...

Ex: instruções de transferência de info no IA-32

mov	S, D	D←S	Move (byte,word,long_word)
movzbl	S, D	D←ZeroExtend(S)	Move Byte-Long Zero-Extended
movsbl	S, D	D←SignExtend(S)	Move Byte-Long Sign-Extended
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* / *Pointer*

D – destino: [Reg | Mem] **S** – *source*, fonte: [Imm | Reg | Mem]

D e **S** não podem ser ambos operandos em memória no IA-32

Ex: instruções aritméticas/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

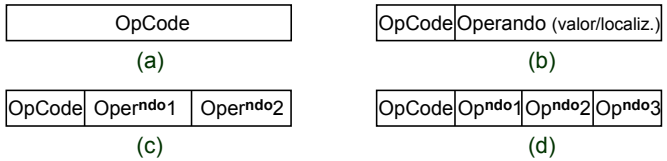
Ex: instruções de controlo de fluxo no IA-32

jmp	Label	%eip ← Label	Unconditional jump
je	Label		Jump if Zero/Equal
js	Label		Jump if Negative
jg	Label		Jump if Greater (signed >)
jge	Label		Jump if Greater or equal (signed >=)
ja	Label		Jump if Above (unsigned >)
call	Label	pushl %eip; %eip ← Label	Procedure call
ret		popl %eip	Procedure return

- registos visíveis ao programador?
- tipos de instruções presentes num processador?
- formatos de instruções em linguagem máquina?
- instruções de *input/output*?

Formatos de instruções em linguagem máquina

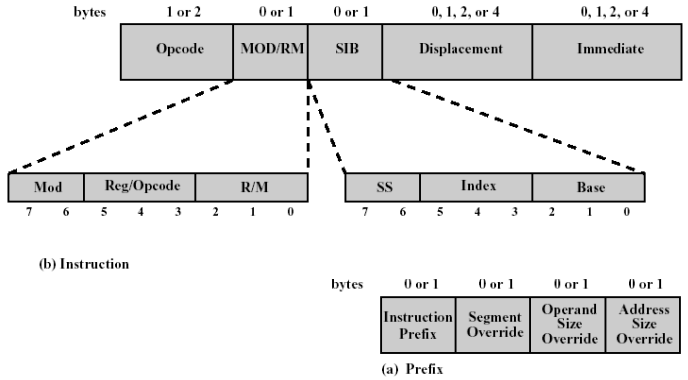
– campos duma instrução



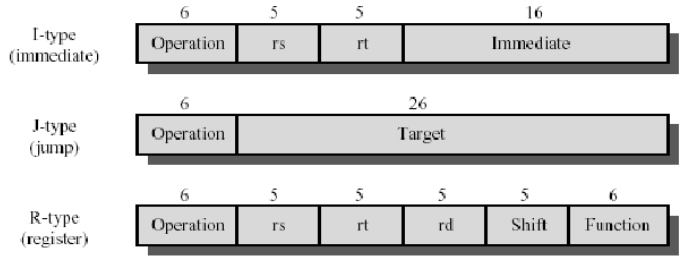
– comprimento das instruções
- variável (prós e contras; IA-32...)
- fixo (prós e contras; RISC...)

– exemplos de formatos de instruções

Formatos de instruções no IA-32



Formatos de instruções no MIPS (RISC)



Instruções de *input/output*

- finalidade
  - escrita de comandos
  - leitura de estado
  - escrita/leitura de dados
- específicas (requer sinais de controlo no *bus*...) ; ou
- idênticas ao acesso à memória
  - » *memory mapped I/O*

Escalares multi-*byte* em memória (como ordená-los)

- *little-endian*
- *big-endian*