



## Estrutura do módulo AC

1. Noção de computador
2. Execução de programas num computador
3. Arquitectura ao nível do ISA
  - i. Análise do nível ISA do processador
  - ii. Optimização do desempenho
4. As tecnologias da informação e das comunicações



## Componentes (físicos) a analisar:

- o processador (info adicional):
  - o nível ISA (*Instruction Set Architecture*): tipos/formatos de instruções, acesso a operandos, ...
  - paralelismo no CPU: *pipeline*, superescalaridade, ...
  - CISC *versus* RISC
- a hierarquia de memória:  
*cache*, memória virtual, ...
- periféricos:
  - interfaces humano-computador (HCI)
  - arquivo de informação
  - comunicações (*a referir com as tecnologias...*)

## O processador: análise do nível ISA (*Instruction Set Architecture*)



### Tópicos a analisar

- operações num processador
- registos visíveis aos programador
- modos de acesso aos operandos
- tipos de instruções presentes num CPU
- formatos de instruções em linguagem máquina
- instruções de *input/output*
- ordenação de *bytes*

## Análise do nível ISA (*Instruction Set Architecture*) (1)

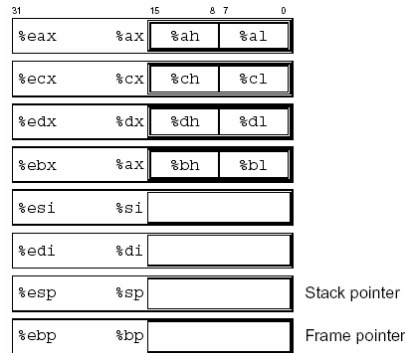


### Operações num processador

- nº de operandos em cada instrução
  - 3-operandos (RISC, ...)
  - 2-operandos (IA32, ...)
  - 1-operando (microcontroladores, ...)
  - 0-operandos (*stack-machine*, ...)
- localização dos operandos
  - variáveis escalares (registos...)
  - variáveis estruturadas (memória...)

## Registos visíveis aos programador (inteiros)

- em arquitecturas RISC (32 registos genéricos...)
- no IA32



## Modos de acesso aos operandos

- em arquitecturas RISC
  - em operações aritméticas/lógicas, sempre em registo
  - em *load/store* usando 1 ou 2 modos de endereço à memória
- no IA32

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	$(, E_i, s)$	$M[R[E_i] \cdot s]$	Scaled indexed
Memory	$Imm(, E_i, s)$	$M[Imm + R[E_i] \cdot s]$	Scaled Indexed
Memory	$(E_b, E_i, s)$	$M[R[E_b] + R[E_i] \cdot s]$	Scaled indexed
Memory	$Imm(E_b, E_i, s)$	$M[Imm + R[E_b] + R[E_i] \cdot s]$	Scaled indexed

## Tipos de instruções presentes num CPU

- operações aritméticas e lógicas
  - soma, subtração, multipl, div, ...
  - AND, OR, NOT, XOR, comparação, ...
  - deslocamento de bits, ...
- transferência de informação
  - de/para registos/memória, ...
- controlo do fluxo de execução
  - para apoio a estruturas de controlo
  - para apoio à invocação de procedimentos
- outras...

## Ex: instruções aritméticas/lógicas no IA32

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

### Ex: instruções de transferência de info no IA32

mov 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

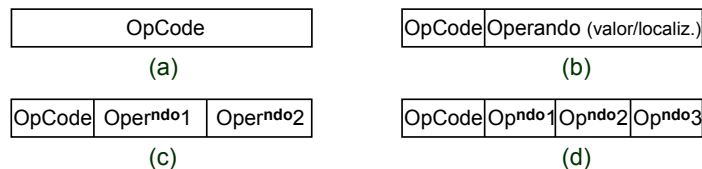
**D** – destino [Reg | Mem]      **S** – fonte [Imm | Reg | Mem]  
**D** e **S** não podem ser ambos operandos em memória

### Ex: instruções de controlo de fluxo no IA32

jmp Label	$\%eip \leftarrow \text{Label}$	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

### Formatos de instruções em linguagem máquina

#### – campos duma instrução

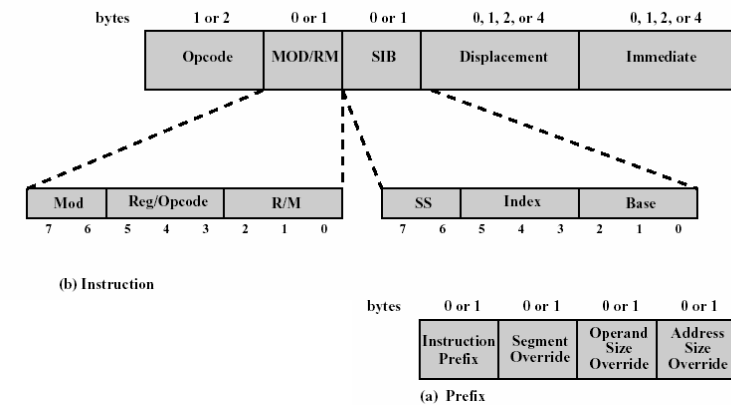


#### – comprimento das instruções

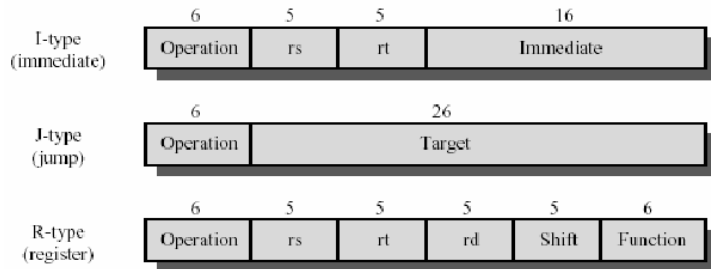
- variável (prós e contras; IA32...)
- fixo (prós e contras; RISC...)

#### – exemplos de formatos de instruções

### Formatos de instruções no Pentium



## 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*

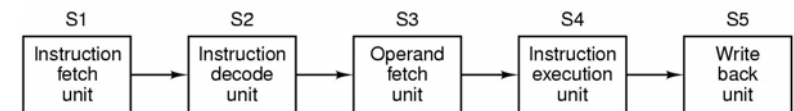
## Ordenação de *bytes*

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

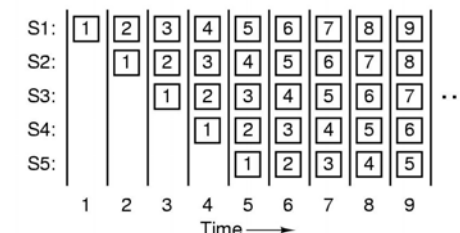
## Modos de otimizar o desempenho no h/w

- com paralelismo
  - ao nível do processo (sistemas paralelos/distribuidos)
  - ao nível da instrução (*Instruction Level Parallelism*)
    - só nos dados (processadores vectoriais)
    - paralelismo desfasado (*pipeline*)
    - paralelismo "real" (superescalar)
- com hierarquia de memória
  - *cache ...*

## Exemplo de *pipeline*

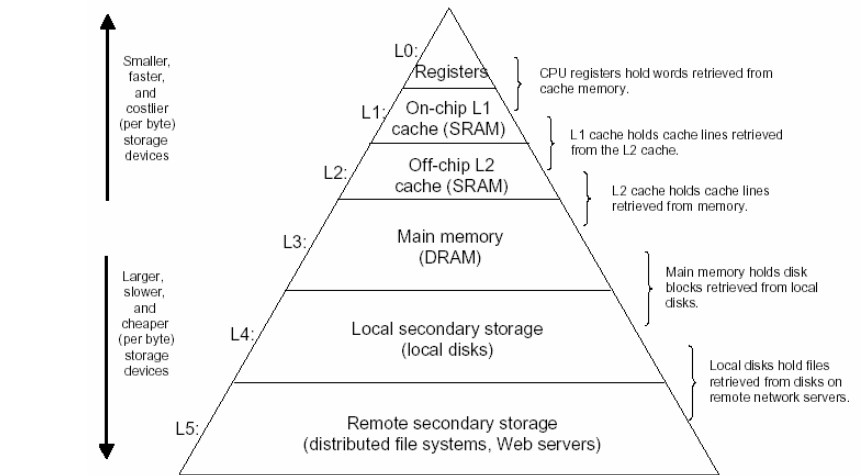
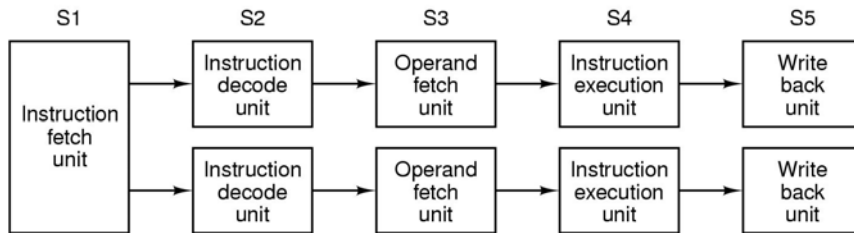


(a)

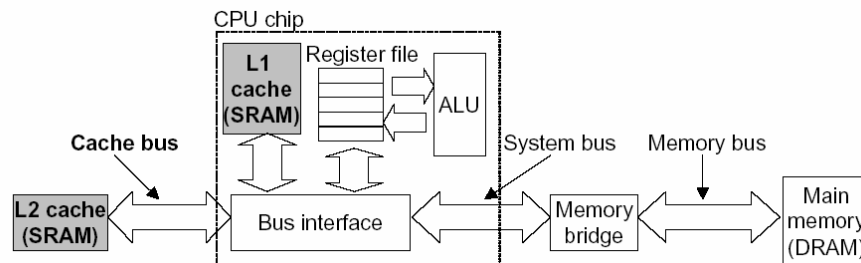


(b)

Exemplo de superescalaridade (nível 2)



A cache em arquitecturas IA32



CISC versus RISC

Caracterização das arquitecturas RISC

- conjunto reduzido e simples de instruções
- formatos simples de instruções
- operandos sempre em registos
- modos simples de endereçamento à memória
- uma operação elementar por ciclo máquina