



Estrutura do tema ISC

1. Representação de informação num computador
2. Organização e estrutura interna dum computador
3. Execução de programas num computador
- 4. Análise das instruções de um processador**
5. 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

O processador: análise do nível ISA

(Instruction Set Architecture) (1)



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?

O processador: análise do nível ISA

(Instruction Set Architecture) (2)



- 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 na instrução p/ cada operação

- 3-operandos (RISC, ...)
ex: `r1 = r2 + r3`
- 2-operandos (IA-32, ...)
ex: `r1 = r1 + r2`
- 1-operando (microcontroladores, ...)
ex: `Acc = Acc + r1`
- 0-operandos (*stack-machine*, ...)
ex: `ToS = ToS + 2ndToS`
ToS: Top-of-Stack

– localização dos operandos

- variáveis escalares, um só valor (em registos...)
- variáveis estruturadas (em memória...)

O processador: análise do nível ISA

(Instruction Set Architecture) (3)

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

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	$(, E_i, s)$	$M[R[E_i] \cdot s]$	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

O processador: análise do nível ISA

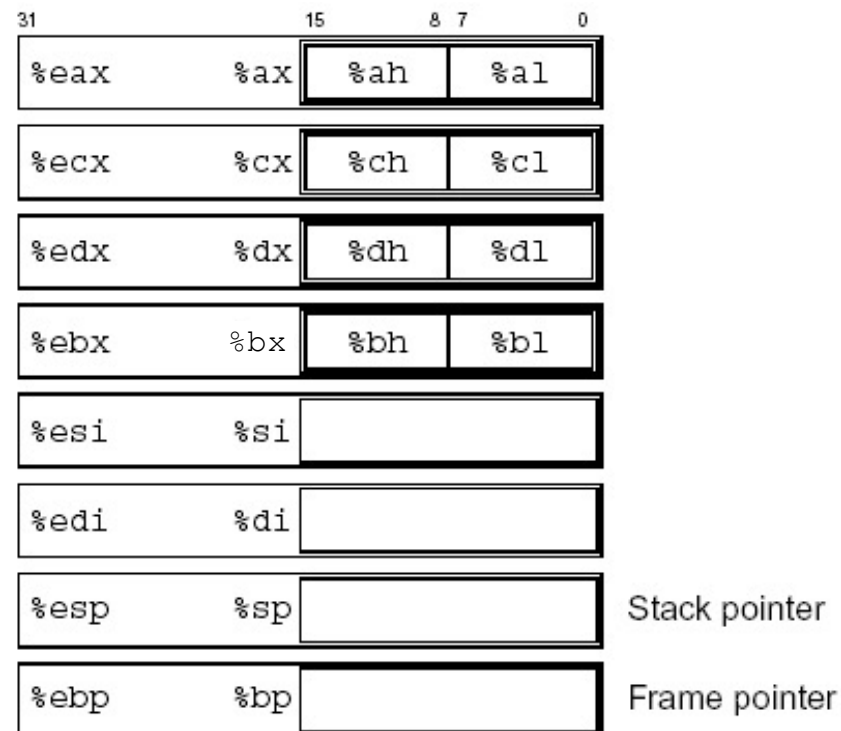
(Instruction Set Architecture) (4)



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

Registos visíveis ao programador (inteiros)

- em arquiteturas RISC: 32 registos genéricos...
- no IA-32:



O processador: análise do nível ISA

(Instruction Set Architecture) (5)

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



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...

O processador: análise do nível ISA

(Instruction Set Architecture) (6)



Ex: instruções de transferência de info no IA-32

mov	S, D	$D \leftarrow S$	Move (byte,word,long_word)
movzbl	S, D	$D \leftarrow \text{ZeroExtend}(S)$	Move Byte-Long Zero-Extended
movsbl	S, D	$D \leftarrow \text{SignExtend}(S)$	Move Byte-Long Sign-Extended
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 <i>Effective Address</i> / <i>Pointer</i>

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

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

O processador: análise do nível ISA

(Instruction Set Architecture) (7)



Ex: instruções aritméticas/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

O processador: análise do nível ISA

(Instruction Set Architecture) (8)



Ex: instruções de controlo de fluxo no IA-32

jmp **Label** **%eip \leftarrow 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 \geq)**

ja **Label** **Jump if Above (unsigned >)**

call **Label** **pushl %eip; %eip \leftarrow Label** **Procedure call**

ret **popl %eip** **Procedure return**

O processador: análise do nível ISA

(Instruction Set Architecture) (9)

- 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 numa instrução



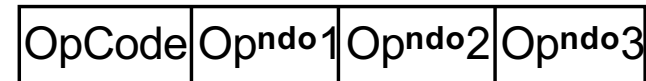
(a)



(b)



(c)



(d)

– comprimento das instruções

- variável (prós e contras; IA-32...)
- fixo (prós e contras; RISC...)

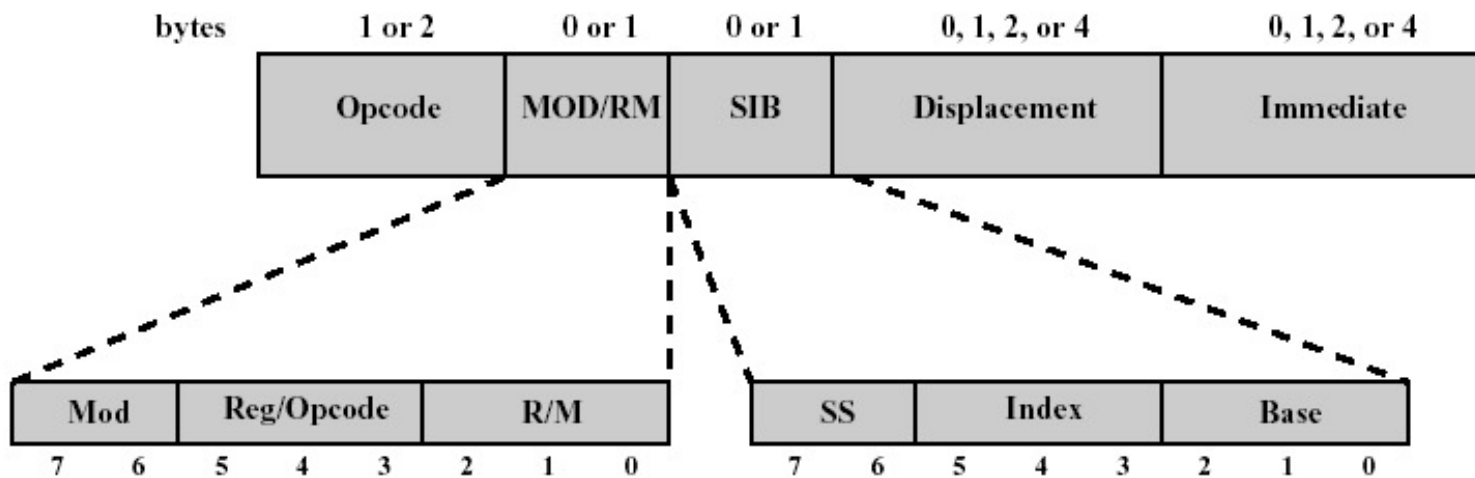
– exemplos de formatos de instruções

O processador: análise do nível ISA

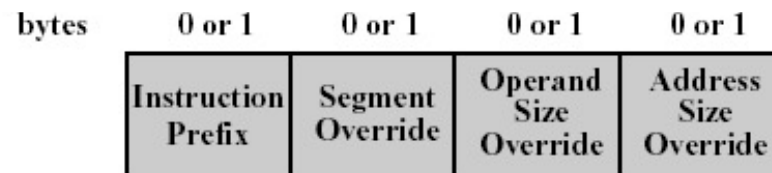
(Instruction Set Architecture) (10)



Formatos de instruções no IA-32



(b) Instruction



(a) Prefix

O processador: análise do nível ISA

(Instruction Set Architecture) (11)



Formatos de instruções no MIPS (RISC)

R-type	op	rs	rt	rd	shamt	funct
--------	----	----	----	----	-------	-------

Arithmetic instruction format

I-type	op	rs	rt	address/immediate
--------	----	----	----	-------------------

Transfer, branch, immediate.

J-type	op	target address
--------	----	----------------

Jump instruction

Field size	6 bits	5bits	5bits	5bits	5bits	6 bits
------------	--------	-------	-------	-------	-------	--------

O processador: análise do nível ISA

(Instruction Set Architecture) (12)



ARM Instruction Formats

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data processing immediate shift	cond			0	0	0		opcode		S		Rn																				Rm
data processing register shift	cond			0	0	0		opcode		S		Rn														Rs	0		shift	1		Rm
data processing immediate	cond			0	0	1		opcode		S		Rn																				immediate
load/store immediate offset	cond			0	1	0		P	U	B	W	L																				immediate
load/store register offset	cond			0	1	1		P	U	B	W	L																				Rm
load/store multiple	cond			1	0	0		P	U	S	W	L																				register list
branch/branch with link	cond			1	0	1		L																								24-bit offset

- S = For data processing instructions, updates condition codes
- S = For load/store multiple instructions, execution restricted to supervisor mode
- P, U, W = distinguish between different types of addressing mode
- B = Unsigned byte (B==1) or word (B==0) access
- L = For load/store instructions, Load (L==1) or Store (L==0)
- L = For branch instructions, is return address stored in link register

O processador: análise do nível ISA

(Instruction Set Architecture) (13)

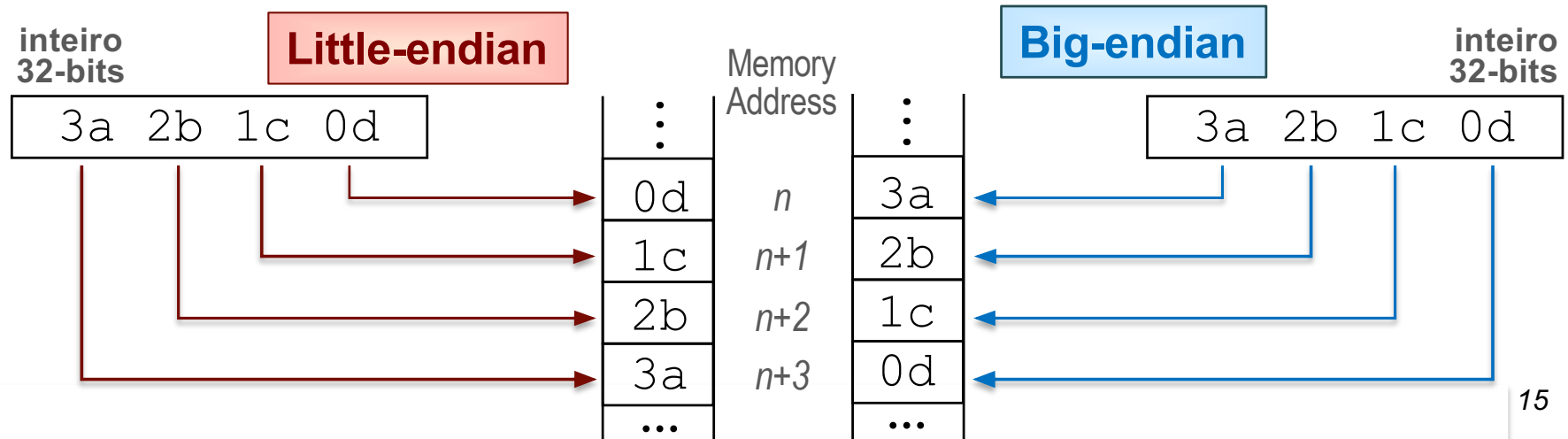
Formatos de instruções em linguagem

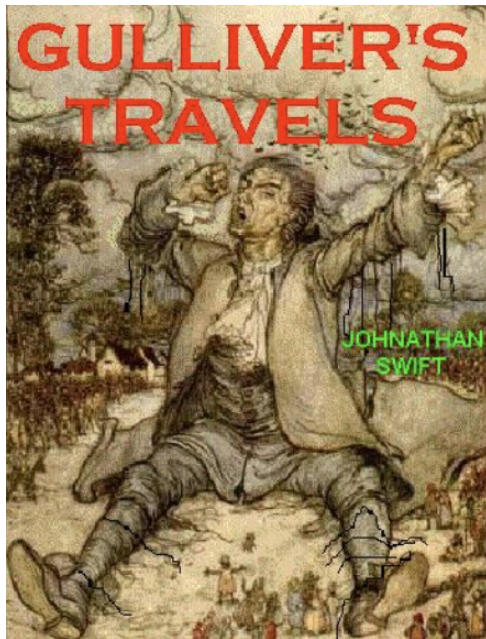
- instruções de *input/output* ?
- escalares multi-byte em memória?

Instruções de *input/output*

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

Escalares multi-byte em memória





Little-Endian vs. Big-Endian

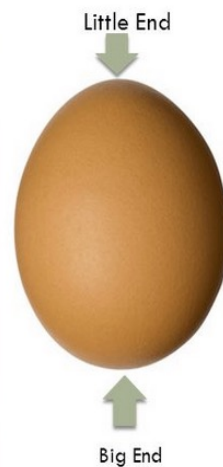
Little-Endian vs. Big-Endian

Origin of the terms

Jonathan Swift, *Gulliver's Travels*

- A law requiring all citizens of Lilliput to break their soft-eggs at the little ends only
- A civil war breaking between the Little Endians and the Big-Endians, resulting in the Big Endians taking refuge on a nearby island, the kingdom of Blefuscu
- Satire over holy wars between Protestant Church of England and the Catholic Church of France

Gulliver's Travels



Big-endians crack soft-boiled eggs at the big end, and little-endians crack them at the other end in the story.

