

Arquitetura de Computadores

ISA – Instruções de transferência de dados

Bib: A – secções 6.2 (memory) e 6.3.6

João Pedro Patriarca (jpatri@cc.isel.ipl.pt), Gabinete F.0.23 do edifício F

ISEL, ADEETC, LEIC

Transferência de valores imediatos

- Instrução MOV Rd, #imm8
 - O valor da constante imm8 é escrito nos bits 0..7 do registro Rd
 - Os bits 8..15 do registro Rd são escritos com zeros
- Instrução MOVT Rd, #imm8
 - O valor da constante imm8 é escrito nos bits 8..15 do registro Rd
 - Os bits 0..7 do registro Rd permanecem inalterados

```
; R0=0x1234
mov    r0, #0x55
movt   r0, #0xAA
```

	15	8	7	0
R0	12		34	
R1				
R2				
R3				
R4				
R5				

Transferência de valores imediatos

- Instrução MOV Rd, #imm8
 - O valor da constante imm8 é escrito nos bits 0..7 do registo Rd
 - Os bits 8..15 do registo Rd são escritos com zeros
- Instrução MOVT Rd, #imm8
 - O valor da constante imm8 é escrito nos bits 8..15 do registo Rd
 - Os bits 0..7 do registo Rd permanecem inalterados

```
; R0=0x1234
mov    r0, #0x55
movt   r0, #0xAA
```

	15	8	7	0
R0	00		55	
R1				
R2				
R3				
R4				
R5				

Transferência de valores imediatos

- Instrução MOV Rd, #imm8
 - O valor da constante imm8 é escrito nos bits 0..7 do registro Rd
 - Os bits 8..15 do registro Rd são escritos com zeros
- Instrução MOVT Rd, #imm8
 - O valor da constante imm8 é escrito nos bits 8..15 do registro Rd
 - Os bits 0..7 do registro Rd permanecem inalterados

```
; R0=0x1234
mov    r0, #0x55
movt   r0, #0xAA
```

	15	8	7	0
R0	AA 55			
R1				
R2				
R3				
R4				
R5				

Instruções de transferência de dados com a memória

- Instruções que envolvem na sua execução acesso a memória
 - *Load*: a informação é lida da memória e escrita num registo
 - *Store*: a informação de um registo é escrita em memória
- Arquitetura *Load/Store*
 - Característica de arquiteturas RISC
 - As instruções de processamento de dados e de controlo de fluxo não envolvem acessos a memória
- No P16 não existe o modo de endereçamento direto; apenas existe o modo de endereçamento indireto e baseado/indexado
- O P16 permite acessos a 8 bits (*byte*) e 16 bits (*word*)
 - Leitura de *bytes*: LDRB
 - Escrita de *bytes*: STRB
 - Leitura de *words*: LDR
 - Escrita de *words*: STR
- O acesso a *words* tem de ser alinhado num endereço par
- Organização *little-endian*: LSB (*Least Significant Byte*) no endereço menor e MSB (*Most Significant Byte*) no endereço maior

Leitura e escrita de *bytes* (1 de 9)

Excerto de programa

```
mov    r0, #0x12
ldrb   r1, [r0]
ldrb   r2, [r0, #1]
mov    r5, #2
ldrb   r1, [r0, r5]
strb   r2, [r0]
strb   r3, [r0, r5]
strb   r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	AA		55	
R1	11		44	
R2	BB		66	
R3	CC		33	
R4	DD		EE	
R5	FF		FA	

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..7\}$ – distância em bytes relativamente à base

Leitura e escrita de *bytes* (2 de 9)

Excerto de programa

```
mov    r0, #0x12
ldrb   r1, [r0]
ldrb   r2, [r0, #1]
mov    r5, #2
ldrb   r1, [r0, r5]
strb   r2, [r0]
strb   r3, [r0, r5]
strb   r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00		12	
R1	11		44	
R2	BB		66	
R3	CC		33	
R4	DD		EE	
R5	FF		FA	

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..7\}$ – distância em bytes relativamente à base

Leitura e escrita de *bytes* (3 de 9)

Excerto de programa

```
mov    r0, #0x12
ldrb   r1, [r0]
ldrb   r2, [r0, #1]
mov    r5, #2
ldrb   r1, [r0, r5]
strb   r2, [r0]
strb   r3, [r0, r5]
strb   r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00		12	
R1	00		12	
R2	BB		66	
R3	CC		33	
R4	DD		EE	
R5	FF		FA	

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..7\}$ – distância em bytes relativamente à base

Leitura e escrita de *bytes* (4 de 9)

Excerto de programa

```
mov    r0, #0x12
ldrb   r1, [r0]
ldrb   r2, [r0, #1]
mov    r5, #2
ldrb   r1, [r0, r5]
strb   r2, [r0]
strb   r3, [r0, r5]
strb   r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	00			12
R2	00			34
R3	CC			33
R4	DD			EE
R5	FF			FA

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..7\}$ – distância em bytes relativamente à base

Leitura e escrita de *bytes* (5 de 9)

Excerto de programa

```
mov    r0, #0x12
ldrb   r1, [r0]
ldrb   r2, [r0, #1]
mov    r5, #2
ldrb   r1, [r0, r5]
strb   r2, [r0]
strb   r3, [r0, r5]
strb   r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	00			12
R2	00			34
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..7\}$ – distância em bytes relativamente à base

Leitura e escrita de *bytes* (6 de 9)

Excerto de programa

```
mov    r0, #0x12
ldrb   r1, [r0]
ldrb   r2, [r0, #1]
mov    r5, #2
ldrb   r1, [r0, r5]
strb   r2, [r0]
strb   r3, [r0, r5]
strb   r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	00			56
R2	00			34
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..7\}$ – distância em bytes relativamente à base

Leitura e escrita de *bytes* (7 de 9)

Excerto de programa

```
mov    r0, #0x12
ldrb   r1, [r0]
ldrb   r2, [r0, #1]
mov    r5, #2
ldrb   r1, [r0, r5]
strb   r2, [r0]
strb   r3, [r0, r5]
strb   r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	00			56
R2	00			34
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
78		0015
56		0014
34		0013
34		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..7\}$ – distância em bytes relativamente à base

Leitura e escrita de *bytes* (8 de 9)

Excerto de programa

```
mov    r0, #0x12
ldrb   r1, [r0]
ldrb   r2, [r0, #1]
mov    r5, #2
ldrb   r1, [r0, r5]
strb   r2, [r0]
strb   r3, [r0, r5]
strb   r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	00			56
R2	00			34
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
78		0015
33		0014
34		0013
34		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..7\}$ – distância em bytes relativamente à base

Leitura e escrita de *bytes* (9 de 9)

Excerto de programa

```
mov    r0, #0x12
ldrb   r1, [r0]
ldrb   r2, [r0, #1]
mov    r5, #2
ldrb   r1, [r0, r5]
strb   r2, [r0]
strb   r3, [r0, r5]
strb   r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	00			56
R2	00			34
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
EE		0015
33		0014
34		0013
34		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..7\}$ – distância em bytes relativamente à base

Leitura e escrita de *words* (1 de 9)

Excerto de programa

```
mov    r0, #0x12
ldr    r1, [r0]
ldr    r2, [r0, #1]
mov    r5, #2
ldr    r1, [r0, r5]
str    r2, [r0]
str    r3, [r0, r5]
str    r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	AA		55	
R1	11		44	
R2	BB		66	
R3	CC		33	
R4	DD		EE	
R5	FF		FA	

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..15\}$ – distância em *bytes* – índices ímpares geram *Warning* na compilação
- Leitura e escrita de *words* alinhadas em endereços pares
- Organização *little-endian* (por oposição à organização *big-endian*)

Leitura e escrita de *words* (2 de 9)

Excerto de programa

```
mov    r0, #0x12
ldr    r1, [r0]
ldr    r2, [r0, #1]
mov    r5, #2
ldr    r1, [r0, r5]
str    r2, [r0]
str    r3, [r0, r5]
str    r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00		12	
R1	11		44	
R2	BB		66	
R3	CC		33	
R4	DD		EE	
R5	FF		FA	

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..15\}$ – distância em *bytes* – índices ímpares geram *Warning* na compilação
- Leitura e escrita de *words* alinhadas em endereços pares
- Organização *little-endian* (por oposição à organização *big-endian*)

Leitura e escrita de *words* (3 de 9)

Excerto de programa

```
mov    r0, #0x12
ldr    r1, [r0]
ldr    r2, [r0, #1]
mov    r5, #2
ldr    r1, [r0, r5]
str    r2, [r0]
str    r3, [r0, r5]
str    r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00		12	
R1	34		12	
R2	BB		66	
R3	CC		33	
R4	DD		EE	
R5	FF		FA	

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..15\}$ – distância em *bytes* – índices ímpares geram *Warning* na compilação
- Leitura e escrita de *words* alinhadas em endereços pares
- Organização *little-endian* (por oposição à organização *big-endian*)

Leitura e escrita de *words* (4 de 9)

Excerto de programa

```
mov    r0, #0x12
ldr    r1, [r0]
ldr    r2, [r0, #1]
mov    r5, #2
ldr    r1, [r0, r5]
str    r2, [r0]
str    r3, [r0, r5]
str    r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00		12	
R1	34		12	
R2	34		12	
R3	CC		33	
R4	DD		EE	
R5	FF		FA	

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..15\}$ – distância em *bytes* – índices ímpares geram *Warning* na compilação
- Leitura e escrita de *words* alinhadas em endereços pares
- Organização *little-endian* (por oposição à organização *big-endian*)

Leitura e escrita de *words* (5 de 9)

Excerto de programa

```
mov    r0, #0x12
ldr    r1, [r0]
ldr    r2, [r0, #1]
mov    r5, #2
ldr    r1, [r0, r5]
str    r2, [r0]
str    r3, [r0, r5]
str    r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	34			12
R2	34			12
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..15\}$ – distância em *bytes* – índices ímpares geram *Warning* na compilação
- Leitura e escrita de *words* alinhadas em endereços pares
- Organização *little-endian* (por oposição à organização *big-endian*)

Leitura e escrita de *words* (6 de 9)

Excerto de programa

```
mov    r0, #0x12
ldr    r1, [r0]
ldr    r2, [r0, #1]
mov    r5, #2
ldr    r1, [r0, r5]
str    r2, [r0]
str    r3, [r0, r5]
str    r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	78			56
R2	34			12
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..15\}$ – distância em *bytes* – índices ímpares geram *Warning* na compilação
- Leitura e escrita de *words* alinhadas em endereços pares
- Organização *little-endian* (por oposição à organização *big-endian*)

Leitura e escrita de *words* (7 de 9)

Excerto de programa

```
mov    r0, #0x12
ldr     r1, [r0]
ldr     r2, [r0, #1]
mov     r5, #2
ldr     r1, [r0, r5]
str     r2, [r0]
str     r3, [r0, r5]
str     r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	78			56
R2	34			12
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
78		0015
56		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..15\}$ – distância em *bytes* – índices ímpares geram *Warning* na compilação
- Leitura e escrita de *words* alinhadas em endereços pares
- Organização *little-endian* (por oposição à organização *big-endian*)

Leitura e escrita de *words* (8 de 9)

Excerto de programa

```
mov    r0, #0x12
ldr    r1, [r0]
ldr    r2, [r0, #1]
mov    r5, #2
ldr    r1, [r0, r5]
str    r2, [r0]
str    r3, [r0, r5]
str    r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	78			56
R2	34			12
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
CC		0015
33		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..15\}$ – distância em *bytes* – índices ímpares geram *Warning* na compilação
- Leitura e escrita de *words* alinhadas em endereços pares
- Organização *little-endian* (por oposição à organização *big-endian*)

Leitura e escrita de *words* (9 de 9)

Excerto de programa

```
mov    r0, #0x12
ldr     r1, [r0]
ldr     r2, [r0, #1]
mov     r5, #2
ldr     r1, [r0, r5]
str     r2, [r0]
str     r3, [r0, r5]
str     r4, [r0, #3]
```

Banco de registos

	15	8	7	0
R0	00			12
R1	78			56
R2	34			12
R3	CC			33
R4	DD			EE
R5	00			02

Memória

7	0	
DD		0015
EE		0014
34		0013
12		0012

- Registo base $\in \{R0..R7\}$
- Registo destino/fonte e índice $\in \{R0..R15\}$
- Constante índice $\in \{0..15\}$ – distância em *bytes* – índices ímpares geram *Warning* na compilação
- Leitura e escrita de *words* alinhadas em endereços pares
- Organização *little-endian* (por oposição à organização *big-endian*)

Leitura de *words* em endereços baseado no PC

- Instrução LDR Rd, [pc, #imm6]
- Resultado da instrução: $Rd = Mem[pc + imm6 * 2]$
- A constante imm6 representa um valor no domínio dos naturais entre 0 e 127
- A distância da definição da *word* relativamente ao valor do registo PC na execução do LDR não pode exceder 64 *words*
- Exemplo:

```
ldr    r0, var          ; r0 = 0xa5a5
ldr    r1, var + 2      ; r1 = 0x1234
...
var: .word 0xa5a5, 0x1234
```
- A utilidade desta instrução será comprovada em sessões futuras

Desafios

1. Dada uma sequência em memória de 10 inteiros sem sinal a 16 bits, escreva no registo R0 o valor do maior inteiro
2. Dada uma sequência em memória de 10 inteiros com sinal a 16 bits, escreva no registo R0 o valor do maior inteiro
3. Dada uma *string* de caracteres em memória apenas com letras minúsculas, escreva no registo R0 o índice do carácter com código mais alto presente na *string*. Note que o índice do primeiro carácter na *string* corresponde ao índice 0, a *string* termina com o valor 0 ('\0') e os códigos dos caracteres são consecutivos na tabela ASCII ('a'=0x61 .. 'z'=0x7A)