Arquitetura de Computadores

Exercício sobre codificação de instruções

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

ISEL, ADEETC, LEIC

Enunciado

As instruções apresentadas na Tabela 1 constituem o conjunto de instruções de um processador a 8 bits do tipo *Reduced Instruction Set Computer* (RISC) com quatro registos de uso geral e um registo de estado com a designação *Current Program Status Register* (CPSR) que, entre outros dados, inclui um bit que indica que a última operação realizada deu resultado zero (flag Z). A capacidade total de endereçamento deste processador é de 256 bytes, sendo os acessos à memória realizados apenas ao byte.

Instrução	Descrição
1. `add rd, rn`	`rd ← rd + rn`
2. `b rn`	`PC ← rn`
3. `bzs label`	`(CPSR.Z == 1) ? PC ← PC + label : PC ← PC + 1`
4. `cmp rn, rm`	`(rn – rm == 0) ? CPSR.Z ← 1 : CPSR.Z ← 0`
5. `ldr rd, [rn]`	`rd ← mem[rn]`
6. `mov rd, #imm3`	`rd ← imm3`
7. `str rs, [#imm3]`	`mem[rn] ← rs`

`rd`, `rm` e `rn` representam um dos registos de uso geral do processador, `imm3` simboliza um número natural codificado com 3 bits, `label` identifica um endereço na vizinhança de ±16 bytes da instrução de salto em causa e PC referencia o registo *Program Counter* do processador.

Questões #1

a) Classifique as sete instruções nas seguintes classes, de acordo com a função que realizam: manipulação de dados, transferência de dados e controlo.

Legenda: MD – Manipulação de Dados; TD – Transferência de dados; C – Controlo

1. MD; 2. C; 3. C; 4. MD; 5. TD; 6. MD; 7. TD

b) Indique, justificando, o número de bits a utilizar para codificar os campos de todas as instruções.

opcode = 3 bits rd = rn = rm = 2 bits imm3 = 3 bits label = 5 bits

Questões #2

c) Proponha um mapa de codificação para o conjunto de instruções apresentado, considerando a utilização de um código de comprimento fixo com oito bits e um esquema de codificação uniforme.

	7	6	5	4	3	2	1	0
`1. add rd, rn`	opcode			-	rn		rd	
`2. b rn`	opcode			-	rn		-	-
`3. bzs label`	opcode			label				
`4. cmp rn, rm`	opcode			-	rn		rm	
`5. ldr rd, [rn]`	opcode			-	rn		rd	
`6. mov rd, #imm3`	opcode			imm3			rd	
`7. str rs, [#imm3]`	opcode			imm3			rs	

opcode = bits(7..5)
rd = rm = rs = bits(1..0)

imm3 = bits(4..2)
rm = bits(3..2)

label = bits(4..0)

Questões #2

d) Indique, justificando, um valor para o código de operação (*opcode*) de cada instrução.

	7	6	5	4	3	2	1	0
1. `add rd, rn`	**0**	**0**	**0**	-	rn		rd	
2. `b rn`	**0**	**1**	**0**	-	rn		-	-
3. `bzs label`	**0**	**1**	**1**	label				
4. `cmp rn, rm`	**0**	**0**	**1**	-	rn		rm	
5. `ldr rd, [rn]`	**1**	**0**	**0**	-	rn		rd	
6. `mov rd, #imm3`	**1**	**1**	**1**	imm3			rd	
7. `str rs, [#imm3]`	**1**	**0**	**1**	imm3			rs	

O código 110 não foi utilizado

Questões #3

e) Considerando agora que o segundo operando da instrução `mov` é uma constante codificada com quatro bits (`mov rd, #imm4`), repita as alíneas b) a d).

	7	6	5	4	3	2	1	0
`1. add rd, rn`	0	0	0	-	rn		rd	
`2. b rn`	0	1	0	-	rn		-	-
`3. bzs label`	0	1	1	label				
`4. cmp rn, rm`	0	0	1	-	rn		rm	
`5. ldr rd, [rn]`	1	0	0	-	rn		rd	
`6. mov rd, #imm3`	1	1	imm4				rd	
`7. str rs, [#imm3]`	1	0	1	imm3			rs	

A alteração da dimensão da constante esgotou os códigos possíveis a 3 bits

Questões #3

f) Repita o procedimento anterior mas considerando agora que a constante é codificada com 4 bits, quer para a instrução `mov`, quer para a instrução `str`.

	7	6	5	4	3	2	1	0
`1. add rd, rn`	0	0	0	**0**	rn		rd	
`2. b rn`	0	1	0	-	rn		-	-
`3. bzs label`	0	1	1	label				
`4. cmp rn, rm`	0	0	1	-	rn		rm	
`5. ldr rd, [rn]`	**0**	0	0	**1**	rn		rd	
`6. mov rd, #imm3`	1	1	imm4				rd	
`7. str rs, [#imm3]`	1	0	imm4				rs	

A alteração da dimensão da constante forçou a alteração da dimensão dos códigos de operação de forma a garantir códigos únicos; opcode = bits(7..4)