Arquitetura e Organização de Computadores

Exercícios - Parte III

António José Araújo

João Canas Ferreira

Bruno Lima

Mestrado Integrado em Engenharia Informática e Computação

Novembro de 2018

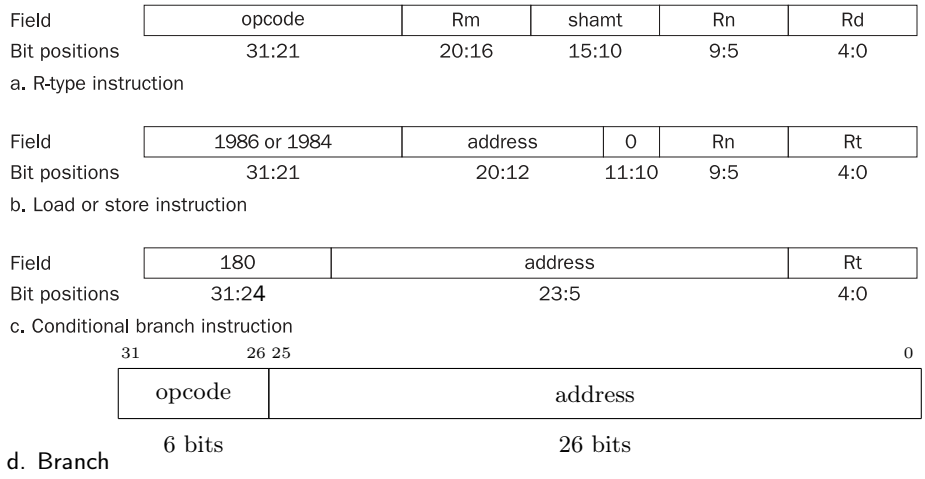
Conteúdo

Esta coletânea reúne exercícios resolvidos e propostos sobre a matéria lecionada em 2018/19 na unidade curricular de *Arquitetura e Organização de Computadores* do 1º ano do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

1 Organização de um CPU

As figuras foram extraídas do livro "Computer Organization and Design – ARM Edition", Patterson & Hennessey.

Resumo dos formatos de codificação e opcodes



Instrução	Opcode
ADD	100 0101 1000
SUB	110 0101 1000
AND	100 0101 0000
ORR	101 0101 0000
LDUR	111 1100 0010
STUR	111 1100 0000
CBZ	101 1010 0
B	000 101

ALU trabalha em 3 contextos diferentes.

1. instruções lógico-aritméticas: `ALUOp[1:0]=10`
2. cálculo de endereços: `ALUOp[1:0]=00`
3. comparação: `ALUOp[1:0]=01`

1.1 Exercícios resolvidos

Exercício 1

Considere o CPU da figura 1.1. Determine o valor de todos os sinais de controlo durante a execução da seguinte instrução:

```
ADD    X3, X8, X11
```

Indique também quais os componentes com funções úteis.

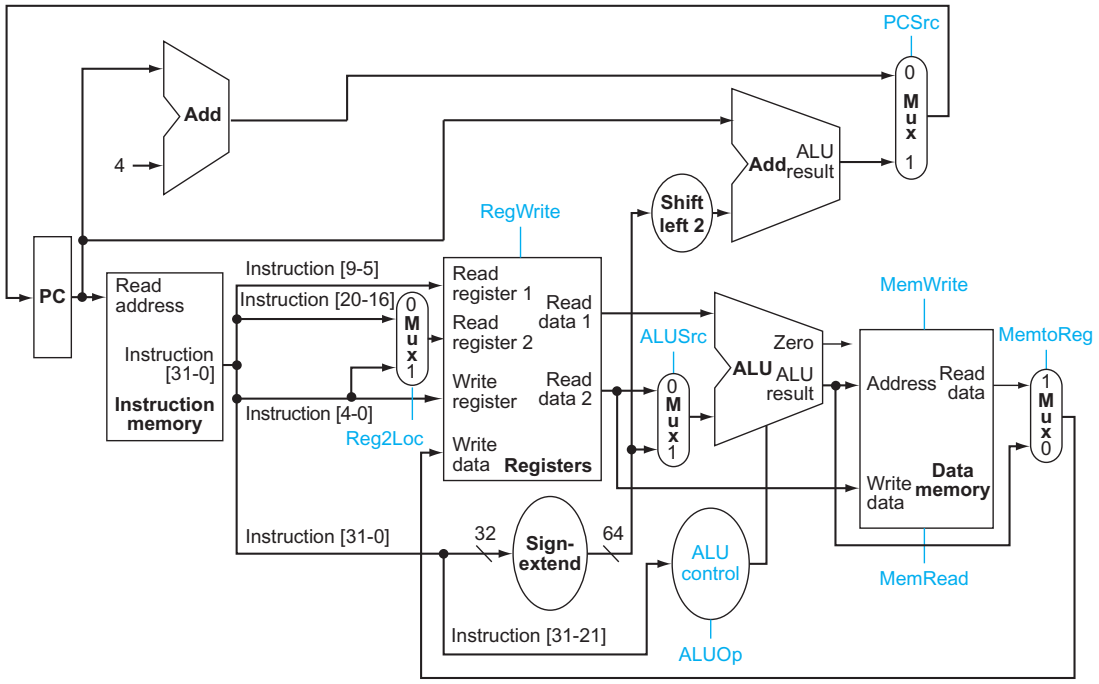


Figura 1.1: CPU com multiplexadores e sinais de controlo

Os valores dos sinais de controlo são determinados pela operação efetuada, que estabelece a forma como os recursos do CPU são usados.

A instrução `ADD` usa dois registos como fonte de operandos e um como destino do resultado. Como a instrução altera o banco de registos (para guardar o resultado da soma), deve ter-se `RegWrite=1`. Como o valor que é guardado no registo provém da ALU e não da memória, tem-se `MemtoReg=0`. Nas instruções tipo R, a especificação do registo do segundo operando é feita pelos bits 20:16 da instrução, pelo que `Reg2Loc=0`.

Esta instrução não acede a memória, seja para leitura, seja para escrita. Por isso, `MemRead=0` e `MemWrite=0`. Como também não é um salto condicional, `Branch=0`, o que implica `PCSrc=0` (porque `PcSrc = AND(Zero, Branch)`).

A instrução `ADD` usa a ALU para realizar a operação (adição) sobre dois operandos provenientes de registos. Um operando da ALU provém sempre do banco de registos; a origem do segundo é determinada pelo sinal `ALUSrc`. Como neste caso o segundo operando também provém do banco de registos, tem-se `ALUSrc=0`. A operação da ALU é controlada pelo sinal de 2 bits `ALUOp`. Como se trata de uma instrução do tipo R, tem-se `ALUOp=`10_2. Esta instrução não afeta o fluxo de instruções, pelo que `PCSrc=0`.

Todos os módulos têm funções úteis, exceto a memória da dados `(Data memory)`, o módulo de extensão de sinal `(Sign extend)`, o módulo de deslocamento `(Shift left 2)` e o somador usado para calcular o destino dos saltos relativos (`Add` ligado a `Shift left 2`).

Exercício 2

Suponha que a unidade de controlo usada com o CPU da figura 1.1 possui um defeito de fabrico que faz com que um determinado sinal de controlo tenha o valor fixo 0. Indique, justificando, quais as instruções que deixam de funcionar corretamente, considerando que o defeito ocorre em:

a) `RegWrite` b) `ALUOp[1]`

Para que uma instrução não funcione como esperado é necessário que o sinal, em operação correta, assuma um valor diferente daquele que é imposto pelo defeito de fabrico.

a) Como o defeito de fabrico impõe sempre `RegWrite=0`, todas as instruções que requerem `RegWrite=1` funcionarão incorretamente. As instruções com `RegWrite=1` são as que alteram o conteúdo de um registo: as instruções do tipo R e a instrução `LDUR`.

b) O sinal `ALUOp[1]=1` somente para as instruções lógico-aritméticas. Como o defeito impõe `ALUOp[1]=0`, a ALU comporta-se nestas instruções como se estivesse a efetuar um cálculo de endereços, i.e., a efetuar uma adição. Logo, o defeito de fabrico impede todas as instruções lógico-aritméticas, exceto a adição, de funcionarem corretamente: `SUB`, `AND` e `ORR`.

Exercício 3

Processadores podem apresentar defeitos de fabrico. Suponha que se pretende detetar se um CPU tem um certo defeito procedendo da seguinte forma:

1. preencher PC, registos e memória de instruções com valores selecionados apropriadamente;
2. executar *uma única* instrução;
3. ler os valores de PC, registos e memória de dados.

Os valores lidos são examinados para determinar se uma dada falta está presente ou não.

a) Apresente um teste que permita determinar se o bit 12 da saída da memória de instruções está sempre a 0 (*stuck-at-0*).

b) Repita a alínea anterior para uma falta *stuck-at-1* (saída sempre a 1). É possível usar um único teste para os dois casos? Se sim, como; se não, porquê?

Para detetar um defeito é preciso provocar uma situação em que o sinal afetado assuma normalmente o valor complementar daquele que é imposto pelo defeito.

a) Este defeito afeta o bit 12 das instruções obtidas de memória. Nas instruções de tipo R, este bit pertence ao campo `shamt`, que não é usado. Por isso, é necessário usar uma instrução de outro tipo. Uma solução possível:

```
CBZ X31, 128
```

Na ausência de defeito, PC ← PC + 128 × 4. Na presença do defeito, $128_{10} = 000\ldots10000000_2$ transforma-se em $000\ldots00000000_2$, já que o bit 12 (o décimo terceiro bit da instrução) é forçado a 0. Portanto, o valor de PC mantém-se.

b) Por um raciocínio análogo ao da alínea anterior, é necessário usar uma constante com o bit 12 a zero. Por exemplo: `CBZ X31, 256`.

Na ausência de defeito, temos PC ← PC + 256 × 4; no caso contrário, toma o valor PC ← PC + 384 × 4 (o campo de endereço muda de $000\ldots100000000_2$ para $000\ldots110000000_2$)

Não é possível testar simultaneamente a presença de um ou outro defeito, porque isso exigiria usar uma única instrução para impor condições opostas no local do defeito.

Exercício 4

A tabela seguinte indica a latência máxima dos blocos usados na implementação do CPU conforme indicado na figura 1.1.

I-Mem	Add	Mux	ALU	Regs	D-Mem	Controlo
400 ps	100 ps	30 ps	120 ps	200 ps	350 ps	100 ps

A latência dos restantes módulos é nula.
Determine o caminho crítico (caminho de propagação de sinais com maior latência) para a instrução `AND`.

É preciso encontrar o caminho ao longo do qual a propagação de sinais demora mais. O tempo de propagação deve ser contado a partir da saída do registo `PC` (que muda no início de cada ciclo). Para cada instrução, existem duas tarefas a realizar:

1. Calcular o endereço da próxima instrução.
2. Realizar a operação da instrução corrente.

Estas duas tarefas são independentes, exceto no caso das instruções que alteram o fluxo de instruções (saltos condicionais e incondicionais). Para a instrução `AND` as tarefas são independentes.

Para calcular o tempo de propagação associado a cada módulo, é preciso determinar o respetivo sinal de entrada que é atualizado em último lugar. Para um módulo de latência L com N entradas *relevantes* cujos valores estejam atualizados nos instantes $t_1, t_2, \ldots, t_n$, e latência máxima L, a saída estará atualizada no instante $t = \max(t_1, t_2, \ldots, t_n) + L$.

Por exemplo, a saída da memória de instruções `(I-MEM)` está atualizada 400 ps após o início da instrução. A unidade de controlo `(Control)` atualiza todos os sinais de controlo no instante

$$400\,\text{ps} + 100\,\text{ps} = 500\,\text{ps}$$

Para determinar o caminho crítico, é preciso determinar qual das duas tarefas demora mais.

1ª tarefa: A propagação de dados relativos ao cálculo do próximo endereço passa pelos seguintes módulos:

`Add` → `Mux`

A saída de `Add` está atualizada após 100 ps.

As entradas relevantes de `Mux` são a saída do 1º somador e o sinal de controlo. Este último está atualizado no instante $t = 500\,\text{ps}$, conforme calculado anteriormente. Portanto, a saída de `Mux` está atualizada em

$$t = \max(100\,\text{ps}, 500\,\text{ps}) + 30\,\text{ps} = 530\,\text{ps}$$

Portanto, o endereço da próxima instrução está calculado após 530 ps.

2ª tarefa: A realização da operação utiliza os seguintes módulos:

$$\texttt{I-Mem} \rightarrow \texttt{Control} \rightarrow \texttt{Mux} \rightarrow \texttt{Regs} \rightarrow \texttt{Mux} \rightarrow \texttt{ALU} \rightarrow \texttt{Mux}$$

O 1º `Mux` é o que está ligado à entrada `Read Register 2`, o 2º `Mux` é o da entrada da ALU e o 3º é o que está à saída da memória `D-Mem`. É de notar que a entrada de seleção do 1º `Mux` só está disponível ao fim de $400\,\text{ps} + 100\,\text{ps} = 500\,\text{ps}$. Por este motivo, nesta instrução (e em outras do tipo R) a unidade de controlo pertence ao caminho crítico.

A saída superior de `Regs` está pronta no instante

$$t = 400\,\text{ps} + 200\,\text{ps} = 600\,\text{ps}$$

A saída inferior de `Regs` está pronta no instante

$$t = 400\,\text{ps} + 100\,\text{ps} + 30\,\text{ps} + 200\,\text{ps} = 730\,\text{ps}$$

No instante $t = 730\,\text{ps}$ já todos os sinais de controlo estão atualizados, o que implica que, daqui para a frente, estes já *não* influenciam o cálculo do tempo de propagação.

Portanto, o resultado da operação está pronto (i.e., disponível na entrada `Write data` do banco de registos) no instante

$$t = 730\,\text{ps} + 30\,\text{ps} + 120\,\text{ps} + 30\,\text{ps} = 910\,\text{ps}$$

Falta verificar se as demais entradas relevantes do módulo `Regs` estão atualizadas neste instante. A entrada `RegWrite` está atualizada desde o instante 500 ps, porque é um sinal de controlo. A entrada `Write register` está pronta desde o instante 400 ps. Portanto, a entrada `Write data` é a última a ficar atualizada.

Concluindo, o caminho crítico é $\texttt{I-Mem} \rightarrow \texttt{Control} \rightarrow \texttt{Mux} \rightarrow \texttt{Regs} \rightarrow \texttt{Mux} \rightarrow \texttt{ALU} \rightarrow \texttt{Mux}$, com um tempo de propagação de 910 ps.

Exercício 5

Assuma que o processador da figura 1.1 recebe a seguinte instrução:

`1111 1000 0100 0001 0000 0000 0100 0011`

a) Indique os valores das saídas dos componentes `Sign-extend` e `Shift-left2`.

b) Indique os valores de entrada da unidade de controlo da ALU.

c) Qual é o valor de PC depois da execução da instrução?

Assuma agora que o conteúdo dos registos é o seguinte:

X0	X1	X2	X3	X4	X5	X6	X7	X12
0	1	4	3	-4	5	6	8	1

d) Indique os valores de saída de cada multiplexador.

e) Indique os valores de entrada da ALU e dos dois somadores.

f) Indique os valores de todas as entradas do banco de registos.

A resolução usa a figura 1.1 como referência.

a) O componente `Sign-extend` tem à entrada a instrução, e produz um valor de 64 (com sinal) a partir de um campo da instrução. Neste caso, `Instr[26]=0`, pelo que o campo escolhido é `Instr[20:12]`. O valor do campo é 000010000_2, logo o valor à saída de `Sign-Extend` é:

`0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00001 0000`$_2$

ou `0x0000000000000010`

O componente `Shift-left2` produz à saída o valor da entrada deslocado de 2 bits para a esquerda: `0x000000000040`.

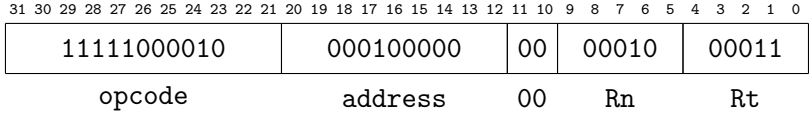
b) Para responder às alíneas seguintes é necessário saber de que instrução se trata (para se conhecerem os valores dos sinais de controlo).

Consultado a lista de *opcodes*, temos `opcode=11111000010`$_2$ `= 1986`$_{10}$, logo trata-se de uma instrução `LDUR`.

Para a instrução `LDUR`, a ALU é usada para calcular o endereço dos dados fazendo uma adição: `ALUop=00`.

c) Como a instrução `LDUR` não altera o fluxo de instruções, `PC` $\leftarrow$ `PC + 4`.

d) Descodificando a instrução, tem-se:



Portanto, `Rn=2` e `Rt=3`. A instrução completa é: `LDUR X3,[X2,#16]`.

Como a instrução usa a ALU para calcular a soma `X2+16`, o multiplexador à entrada da ALU seleciona a constante (após extensão de sinal): a saída desse multiplexador tem o valor 16.

Como na instrução `LDUR` o registo de destino é definido por `Instr[0:4]`, o multiplexador que alimenta a entrada `Read register 2` pode selecionar qualquer valor uma vez que é irrelevante para esta instrução. Logo, a sua saída pode ser 00010_2=2 ou 00011_2=3.

A instrução `LDUR` transfere um valor de memória para um registo. Portanto, a saída do multiplexador controlado por `MemtoReg` tem o valor proveniente da posição de memória lida (valor armazenado em memória a partir da posição 20).

Esta instrução não altera o fluxo de instruções. Portanto, a saída do multiplexador controlado por `PCSrc` é igual a `PC+4`.

e) A entrada superior da ALU tem o valor do endereço-base (registo `X2`):4.

A entrada inferior da ALU tem a constante 16 (em 64 bits).

As entradas do somador da esquerda são: `PC` e 4.

As entradas do somador de destino de saltos são: `PC+4` e o valor da constante multiplicado por 4 (efeito do deslocamento de 2 bits para a esquerda) $16 \times 4 = 64$ (em 64 bits).

f) Os valores das entradas do banco de registos são:

- `Read register 1`: valor de `Rn`: 2.
- `Read register 2`: o valor de `Reg2Loc` é irrelevante para esta instrução. Logo, o valor de `Read register 2` pode ser 00010_2=2 ou 00011_2=3.
- `Write register`: valor de `Rt`: 3
- `Write data`: valor lido de memória (do endereço 20).
- `RegWrite`: 1 (sinal de controlo para habilitar o armazenamento do novo valor).

1.2 Exercícios propostos

Exercício 6

Para as seguintes instruções em código-máquina ARMv8, determine as instruções *assembly* correspondentes. Indique o tipo de codificação, bem como os valores de todos os campos.

a) `0xF81F80A0`

b) `0xF8404023`

Exercício 7

Considere o CPU da figura 1.1. Determine o valor de todos os sinais de controlo durante a execução das instruções indicadas a seguir. Indique também quais os componentes com funções úteis.

a) `LDUR X3,[X7,#64]`

b) `STUR X8,[X2,#128]`

c) `CBZ X9,#-5`

Exercício 8

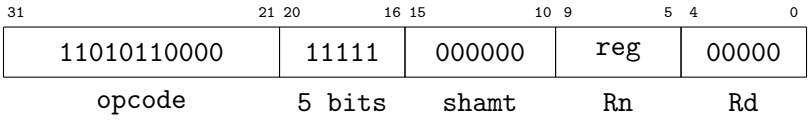
Suponha que a unidade de controlo usada com o CPU da figura 1.1 possui um defeito de fabrico que faz com que um determinado sinal de controlo tenha o valor fixo 0. Indique, justificando, quais as instruções que deixam de funcionar corretamente, considerando que a falha ocorre em:

a) `PCSrc` b) `MemWrite`

Exercício 9

Pretende-se implementar a instrução `BR reg` *(branch register)*. Esta instrução passa o fluxo de execução para a instrução cujo endereço está no registo `reg`: `PC` ← `reg`.

A codificação de `BR` tem o formato R com alguns campos fixos:



Indique quais os elementos e sinais de controlo que devem ser acrescentados ao CPU indicado na figura 1.1. Defina os valores dos sinais de controlo para a nova situação.

Exercício 10

Considere a instrução `LDUR X8, [X17,#40]`. Para responder às questões seguintes refira-se à figura 1.1.

a) Represente a instrução em linguagem-máquina.

b) Qual é o número de registo fornecido à entrada `Read register 1`? O valor deste registo é usado?

c) Responda à mesma questão para a entrada `Read register 2`.

d) Qual é o número de registo fornecido à entrada `Write register`? O valor deste registo é realmente alterado?

e) Qual é o valor dos sinais de controlo `MemRead` e `MemWrite`?

f) Responda às alíneas anteriores para a instrução:

```
Label CBZ X11, Label
```

Exercício 11

A tabela seguinte indica a latência máxima dos blocos usados na implementação do CPU indicado na figura 1.1.

I-Mem	Add	Mux	ALU	Regs	D-Mem	Controlo
400 ps	100 ps	30 ps	120 ps	200 ps	350 ps	100 ps

Determine o caminho crítico (caminho de propagação de sinais com maior latência) para as instruções seguintes, bem como o tempo de propagação correspondente.

a) `LDUR` b) `CBZ`

Exercício 12

Pretende-se projetar a unidade de controlo para uma implementação do CPU em que os componentes têm a seguinte latência máxima:

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-extend	Shift-left2	ALU Ctrl
400 ps	100 ps	30 ps	120 ps	200 ps	350 ps	20 ps	0 ps	50 ps

a) Determine o tempo disponível para a unidade de controlo gerar o sinal `MemWrite` sem aumentar a latência da implementação.

b) Determine qual é o sinal de controlo que pode demorar mais a ser gerado e qual é o tempo de que a unidade de controlo dispõe para esse efeito sem aumentar a latência da implementação.

c) Determine qual é o sinal de controlo que pode demorar *menos* a ser gerado (sinal crítico) e qual é o tempo de que a unidade de controlo dispõe para esse efeito sem aumentar a latência da implementação.

2 Memória Cache

2.1 Exercícios resolvidos

Exercício 1
Considere uma memória *cache* do tipo *direct-mapped*, com 8 blocos e 1 palavra por bloco. Complete a seguinte tabela, que representa a evolução do estado da memória *cache* para os dez acessos consecutivos indicados. Inicialmente, a memória *cache* está vazia. Os endereços são endereços de *palavras*.
Apresente também o conteúdo da memória *cache* após o último acesso.

Acesso	Endereço	Hit/Miss	Conteúdo do bloco	
			inicial	final
1	22			
2	26			
3	22			
4	26			
5	16			
6	3			
7	16			
8	8			
9	27			
10	10			

Como a memória tem 8 blocos, o índice do bloco é definido pelo resto da divisão do endereço por 8.

O efeito da sequência de acessos é descrito a seguir. A operação de obtenção do resto é representada por %:

$$a \ \% \ b = \text{resto da divisão de } a \text{ por } b.$$

1. Endereço 22 →posição 22 % 8 = 6; falha *(miss)*; cache[6] ←mem[22].
2. Endereço 26 →posição 26 % 8 = 2; falha *(miss)*; cache[2] ←mem[26].
3. Endereço 22 →posição 22 % 8 = 6; acerto *(hit)*; cache[6] tem conteúdo de mem[22].
4. Endereço 26 →posição 26 % 8 = 2; acerto *(hit)*; cache[2] tem conteúdo de mem[26].
5. Endereço 16 →posição 16 % 8 = 0; falha *(miss)*; cache[0] ←mem[16].

6. Endereço 3 →posição 3 % 8 = 3; falha *(miss)*; cache[3] ←mem[3].
7. Endereço 16 →posição 16 % 8 = 0; acerto *(hit)*; cache[0] tem conteúdo de mem[16].
8. Endereço 8 →posição 8 % 8 = 0; falha *(miss)*; cache[0] tinha conteúdo de mem[16]; cache[0] ←mem[8].
9. Endereço 27 →posição 27 % 8 = 3; falha *(miss)*; cache[3] tinha conteúdo de mem[3]; cache[3] ←mem[27].
10. Endereço 10 →posição 10 % 8 = 2; falha *(miss)*; cache[2] tinha conteúdo de mem[26]; cache[2] ←mem[10].

Portanto, a tabela fica:

Acesso	Endereço	Hit/Miss	Conteúdo do bloco	
			inicial	final
1	22	M	-	mem[22]
2	26	M	-	mem[26]
3	22	H	mem[22]	mem[22]
4	26	H	mem[26]	mem[26]
5	16	M	-	mem[16]
6	3	M	-	mem[3]
7	16	H	mem[16]	mem[16]
8	8	M	mem[16]	mem[8]
9	27	M	mem[3]	mem[27]
10	10	M	mem[26]	mem[10]

O conteúdo final da memória *cache* é:

Bloco	Conteúdo
0	mem[8]
1	-
2	mem[10]
3	mem[27]
4	-
5	-
6	mem[22]
7	-

Exercício 2

Um processador funciona a 2 GHz e possui uma memória *cache* unificada. A taxa de falhas no acesso à cache é 9% e o tempo de acesso à memória principal é 50 ns.

a) Calcule a penalidade de falha em ciclos.

b) Medições efetuadas para um conjunto de *benchmarks* revelaram que o número de ciclos de protelamento em acessos a memória (por instrução) é $\text{CPI}_{\text{prot}} = 12$.

Determine a percentagem de instruções que acede a dados.

a) Para obter a penalidade de falha em ciclos, determina-se o número de ciclos que o processador executa durante o tempo de acesso a memória principal.

$$T = \frac{1}{F} = 0{,}5\,\text{ns} \qquad \text{logo} \qquad p_f = \frac{50}{0{,}5} = 100$$

b) Seja n_a o número de acessos a memória por instrução e t_f a taxa de falhas.

O número médio de ciclos gastos em protelamento devidos a falhas no acesso a *cache* é:

$$\text{CPI}_{\text{prot}} = t_f \times p_f \times n_a$$

Portanto:

$$n_a = \frac{\text{CPI}_{\text{prot}}}{t_f \times p_f},$$

o que implica

$$n_a = \frac{12}{0{,}09 \times 100} = \frac{4}{3}$$

Cada instrução requer obrigatoriamente um acesso para a sua obtenção. Acessos adicionais são necessariamente acessos a dados. Então, o número médio de acessos a dados é: $n_a - 1 = \frac{1}{3} = 33{,}3\,\%$.

2.2 Exercícios propostos

Alguns exercícios foram extraídos ou adaptados do livro "Computer Organization and Design – The Hardware/Software Interface", Hennessy & Patterson, 4ª edição.

Exercício 3

Um CPU tem endereços de 18 bits e uma memória *cache* de mapeamento direto para palavras (32 bits). A memória *cache* tem 16 posições. A política de escrita é *write-through*. O conteúdo inicial da memória *cache* está indicado na tabela (em hexadecimal).

	conteúdo	etiqueta	v
0	12345678	abc	1
1	6548feab	123	1
2	3c1f56fd	678	1
3	afd12498	567	0
4	6198fa34	b7c	1
5	1929aaaa	8d1	0
6	bbabeedd	cd3	1
7	1123aa56	456	1
8	7611432a	001	1
9	ffffeffe	877	1
10	ddedd556	777	0
11	4444cccc	198	1
12	7627abed	fdf	1
13	8768888a	479	1
14	71672912	655	0
15	22256733	111	1

a) Determine o número de bits da etiqueta e do índice. Qual é o espaço de endereçamento? Qual a quantidade máxima de memória usável num sistema baseado neste CPU?

b) Indique as alterações da memória *cache* para a seguinte sequência de acessos (L=leitura, E=escrita):

tipo	endereço	valor
L	2df10	-
L	23454	-
L	3f7b0	-
E	048c4	1212abab
E	3f7d0	00001111
E	1dde8	aaaabbbb

Exercício 4

Assuma que o CPU do problema anterior é usado com uma memória *cache* também semelhante à anterior, mas que usa a política de escrita *write-back*. O conteúdo inicial da memória *cache* está indicado na tabela (em hexadecimal).

	conteúdo	etiqueta	v	d
0	12345678	abc	1	1
1	6548feab	123	0	0
2	3c1f56fd	678	1	0
3	afd12498	567	0	0
4	6198fa34	b7c	1	0
5	1929aaaa	8d1	0	1
6	bbabeedd	cd3	1	1
7	1123aa56	456	1	0
8	7611432a	001	1	1
9	ffffeffe	877	1	1
10	ddedd556	777	0	0
11	4444cccc	198	1	1
12	7627abed	fdf	0	1
13	8768888a	479	1	0
14	71672912	655	0	0
15	22256733	111	1	0

a) Explique a finalidade do campo `d`.

b) Indique as alterações da memória *cache* para a seguinte sequência de acessos (L=leitura, E=escrita):

tipo	endereço	valor
L	2df10	-
E	2df10	33334444
E	10c64	9999aaaa
L	23454	-
L	3f7b0	-
E	21de4	bbbb7777
E	2afdc	1212abab
E	3f7b0	00001111

Exercício 5

Uma memória *cache* do tipo *write-through* com 4 blocos de 8 bytes é usada como *D-cache* num processador ARMv7.

O conteúdo da memória *cache* é o seguinte (em hexadecimal):

bloco	conteúdo								etiqueta	v
	7	6	5	4	3	2	1	0		
0	2a	5d	04	aa	78	00	9c	23	2900002	1
1	1b	b2	34	bb	7a	10	9f	a3	0000893	1
2	99	52	36	cc	88	b0	3c	2b	7bcd001	1
3	42	15	90	cd	71	ab	3f	6d	65abfff	0

Assuma que `R0` = `0x52000044`, `R1`=`0xf79a0034` e `R3`=0.

a) Qual é o comprimento da etiqueta?

b) Qual é o valor de `R2` após a execução da instrução `LDR R2,[R0,#0]` ?

c) Que modificações da memória *cache* são provocadas pela execução da instrução `STR R3,[R0,#-4]` ?

d) Qual é o valor de `R3` após a execução da instrução `LDRB R3,[R1,#0]` ?

Exercício 6

Para cada um dos sistemas indicados a seguir considere que o tempo de acesso a memória principal é de 70 ns e que 36% das instruções acedem a dados em memória. O acesso a memória principal tem início após falta de acesso à memória *cache*.

Processador	Tamanho	Taxa de faltas	Tempo de acerto
P1	1 KiB	11,4%	0,62 ns
P2	2 KiB	8,0%	0,66 ns

a) Assumindo que é o tempo de acerto que determina o período de relógio, determine as frequências de operação dos dois sistemas.

b) Determine o tempo médio de acesso à memória para os dois casos.

c) Assumindo um CPI básico de 1, qual é o CPI de cada um dos processadores? Qual é o processador mais rápido?

Exercício 7

Um CPU (com F=1 GHz) está equipado com memórias *cache* para instruções e para dados, cujas taxas de faltas são, respetivamente, 5 % e 10 %. O tempo de acesso a memória principal é 80 ns (a acrescentar ao tempo de acesso a memória *cache*).

Em média, 40 % das instruções de um programa acedem a dados (i.e., são *load* ou *store*).

a) Determine a taxa de faltas global da memória *cache* em número de faltas por 1000 instruções.

b) Suponha que se pretendia equipar o CPU com uma memória *cache* unificada. Determine a máxima taxa de faltas desta alternativa para que ela apresente o mesmo desempenho que a versão *split cache*.

c) Assuma que, na ausência de faltas de *cache*, o CPU tem $\text{CPI}_{\text{ideal}}$=1,2. Determine o CPI efetivo para os seguintes casos:

   i) sistema sem memória *cache*;

   ii) sistema com memória *cache*.

3 Desempenho

Alguns destes exercícios são extraídos ou adaptados do livro "Computer Organization and Design – The Hardware/Software Interface", D. Hennessy e J. Patterson, 4ª edição.

3.1 Exercícios resolvidos

Exercício 1
Um computador possui três classes de instruções, A, B e C. O CPI de cada uma delas é, respetivamente, 1, 2 e 3. Um projetista está a implementar um compilador e precisa de escolher uma de duas sequências de instruções a usar nesse computador. Dessas sequências é conhecido o número de instruções de cada classe, conforme mostra a tabela seguinte.

Sequência de instruções	Nº de instruções por classe		
	A	B	C
1	2	1	2
2	4	1	1

a) Determine em qual das duas sequências de instruções é executado o maior número de instruções.

b) Mostre qual das sequências é executada de forma mais rápida.

c) Calcule o valor do CPI para cada sequência.

a) Na sequência 1 são executadas $2 + 1 + 2 = 5$ instruções e na sequência 2 são executadas $4 + 1 + 1 = 6$ instruções. É pois na sequência 2 que são executadas mais instruções.

b) O número de ciclos é dado por

$$N_{\text{ciclos}} = \sum_i N_i \times \text{CPI}_i$$

onde N_i se refere ao número de instruções da classe i e CPI_i é o número de ciclos por instrução dessa classe.

Sequência 1: $N_{\text{ciclos}} = 2 \times 1 + 1 \times 2 + 2 \times 3 = 10$

Sequência 2: $N_{\text{ciclos}} = 4 \times 1 + 1 \times 2 + 1 \times 3 = 9$

A sequência 2 é a mais rápida, porque é executada em menos ciclos de relógio.

c)

$$\text{CPI} = \frac{N_{\text{ciclos}}}{N_{\text{inst}}}$$

Sequência 1: CPI = 10/5 = 2

Sequência 2: CPI = 9/6 = 1,5

Exercício 2

Considere um computador com três classes de instruções, A, B e C, para as quais o valor de CPI é 1, 2 e 3, respetivamente. Para ser executado nesse computador, um programa é obtido através de dois compiladores distintos, originando o número de instruções de cada classe indicado na tabela.

Sequência de instruções	Nº de instruções por classe		
	A	B	C
Compilador 1	5×10^9	1×10^9	1×10^9
Compilador 2	10×10^9	1×10^9	1×10^9

Assuma que o computador funciona a 500 MHz.

a) Indique a sequência que é executada em menos tempo.

b) Determine a qual das sequências de instruções corresponde o maior valor de MIPS (milhões de instruções por segundo).

a) O tempo de execução é dado por

$$t_{\text{exec}} = \frac{N_{\text{ciclos}}}{f_{\text{CLK}}}$$

sendo o número de ciclos de relógio determinado por

$$N_{\text{ciclos}} = \sum_i N_i \times \text{CPI}_i$$

Então, o tempo de execução da sequência de instruções obtida por cada compilador é

$$t_{\text{exec\_C1}} = \frac{(5 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9}{500 \times 10^6} = \frac{10 \times 10^9}{500 \times 10^6} = 20\,\text{s}$$

e

$$t_{\text{exec\_C2}} = \frac{(10 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9}{500 \times 10^6} = \frac{15 \times 10^9}{500 \times 10^6} = 30\,\text{s}$$

Atendendo aos tempos de execução calculados, conclui-se que o compilador 1 gera o programa mais rápido.

b)

$$\text{MIPS} = \frac{N_{instr}}{t_{exec} \times 10^6}$$

Portanto,

$$\text{MIPS}_1 = \frac{(5+1+1)\times 10^9}{20\times 10^6} = 350$$

e

$$\text{MIPS}_2 = \frac{(10+1+1)\times 10^9}{30\times 10^6} = 400$$

Daqui se conclui que o compilador 2 proporciona uma taxa de execução de instruções superior.

Exercício 3

A execução de um programa num computador A, o qual funciona com um sinal de relógio de 2 GHz, demora 10 s. Um computador B que está a ser desenvolvido é capaz de executar o mesmo programa em 6 s. Este computador B pode atingir uma frequência de relógio superior à do computador A, embora consuma 1,2 vezes mais ciclos de relógio do que o computador A a executar o referido programa. Determine a frequência do sinal de relógio a que B pode funcionar.

O tempo de execução de um programa com N_{instr} instruções é dado por

$$t_{\text{exec}} = N_{\text{instr}} \times \text{CPI} \times T_{\text{CLK}}$$

A razão dos tempos de execução no computador A e no computador B é

$$\frac{t_{\text{exec\_A}}}{t_{\text{exec\_B}}} = \frac{N_{\text{instr}} \times \text{CPI}_A \times f_{\text{CLK\_B}}}{N_{\text{instr}} \times \text{CPI}_B \times f_{\text{CLK\_A}}} = \frac{10}{6}$$

Como $\text{CPI}_B = 1{,}2 \times \text{CPI}_A$, então resulta $f_{\text{CLK\_B}} = f_{\text{CLK\_A}} \times 1{,}2 \times \frac{10}{6} = 2 \times f_{\text{CLK\_A}}$, conclui-se que $f_{\text{CLK\_B}} = 4\,\text{GHz}$.

Exercício 4

Considere dois computadores A e B que possuem o mesmo conjunto de instruções. O período do sinal de relógio dos computadores A e B é 250 ps e 500 ps, respetivamente, e o número de ciclos de relógio consumidos por instrução (CPI) é 2 no computador A e 1,2 no computador B, respetivamente. Mostre qual dos computadores é o mais rápido a executar um programa.

$$t_{exec_A} = N_{instr} \times 2 \times 250 = 500 \times N_{instr}\ \text{ps}$$

e

$$t_{exec_B} = N_{instr} \times 1{,}2 \times 500 = 600 \times N_{instr}\ \text{ps}$$

Então,

$$\frac{t_{exec_B}}{t_{exec_A}} = \frac{6}{5} = 1{,}2$$

O computador A é o mais rápido.

Exercício 5

Um dado programa é executado em dois computadores diferentes, A e B, tendo-se obtido os seguintes indicadores:

Indicador	A	B
Nº de instruções	10^7	8×10^6
Frequência	4 GHz	4 GHz
CPI	1	1,1

a) Indique qual o computador com maior valor de MIPS.

b) Indique qual o computador mais rápido.

a)

$$\text{MIPS} = \frac{N_{instr}}{t_{exec} \times 10^6} = \frac{N_{instr}}{N_{instr} \times \text{CPI} \times T_{CLK} \times 10^6} = \frac{f_{CLK}}{\text{CPI} \times 10^6}$$

Logo,

$$\text{MIPS}_A = \frac{4 \times 10^9}{1 \times 10^6} = 4 \times 10^3$$

e

$$\text{MIPS}_B = \frac{4 \times 10^9}{1{,}1 \times 10^6} \approx 3636$$

A é o computador com maior valor de MIPS.

b)

$$t_{exec} = N_{instr} \times \text{CPI} \times T_{CLK} = \frac{N_{instr}}{\text{MIPS} \times 10^6}$$

$$t_{exec_A} = \frac{10^7}{4 \times 10^3 \times 10^6} = 2{,}5\,\text{ms}$$

$$t_{exec_B} = \frac{8 \times 10^6}{3636 \times 10^6} \approx 2{,}2\,\text{ms}$$

B é o computador mais rápido.

Exercício 6

Para um dado programa foram obtidas as seguintes medidas:

- Ocorrência de operações de vírgula flutuante (VFL) (exceto raiz quadrada): 25 %
- Ocorrência da operação de raiz quadrada: 2 %
- CPI médio de operações em VFL: 4,0
- CPI de raiz quadrada: 20
- CPI médio para outras instruções: 1,33

Considere duas alternativas para melhorar o desempenho: reduzir o CPI da operação de raiz quadrada para 2 ou baixar o CPI de todas as instruções de VFL para 2,5. Compare as duas alternativas usando a equação de desempenho do CPU.

O número de instruções e a frequência do relógio do computador continuam a ser os mesmos nas duas alternativas. Basta por isso comparar os valores de CPI.

$$\text{CPI}_1 = 0{,}25 \times 4 + 0{,}02 \times 2 + (1 - 0{,}25 - 0{,}02) \times 1{,}33 \approx 2{,}01$$

$$\text{CPI}_2 = 0{,}25 \times 2{,}5 + 0{,}02 \times 20 + (1 - 0{,}25 - 0{,}02) \times 1{,}33 \approx 2{,}00$$

Embora os valores sejam muito próximos, na segunda alternativa ("... baixar o CPI de todas as instruções de VFL para 2,5") o tempo de execução é menor, ou seja, o desempenho é superior.

Exercício 7
Um processador executa um programa em que 30 % do tempo de execução é gasto em adições de vírgula flutuante, 25 % em multiplicações de vírgula flutuante e 10 % em divisões de vírgula flutuante. Para a nova geração desse processador, a equipa de projeto propõe três aperfeiçoamentos possíveis na unidade de vírgula flutuante, consistindo em tornar o:

1. somador duas vezes mais rápido;
2. multiplicador três vezes mais rápido;
3. divisor dez vezes mais rápido.

Indique qual destas alternativas poderá proporcionar o maior aumento de desempenho.

A lei de Amdahl pode ser expressa por

$$s_{total}(f) = \frac{1}{(1-f) + \frac{f}{s}}$$

onde f representa a fração de tempo gasto na execução da parte melhorada e s traduz o aumento de rapidez dessa parte.

A aplicação da lei de Amdahl às três situações descritas permite pois quantificar o aumento de desempenho:

1. $f = 0{,}3$ e $s = 2 \quad \rightarrow \quad s_{total}(f) = \frac{1}{0{,}7+0{,}3/2} = 1{,}18$
2. $f = 0{,}25$ e $s = 3 \quad \rightarrow \quad s_{total}(f) = \frac{1}{0{,}75+0{,}25/3} = 1{,}20$
3. $f = 0{,}1$ e $s = 10 \quad \rightarrow \quad s_{total}(f) = \frac{1}{0{,}9+0{,}1/10} = 1{,}10$

O maior ganho de desempenho é obtido com a melhoria do multiplicador de vírgula flutuante (alternativa 2). Pode ainda concluir-se que tornar o divisor muito mais rápido leva ao menor aumento de desempenho devido à baixa ocorrência de divisões em vírgula flutuante (10 %). Na verdade, mesmo tornando o divisor infinitamente mais rápido, o *speed-up* obtido seria 1,11.

Exercício 8

A execução de um programa num computador demora 1000 s, sendo 75 % do tempo gasto em multiplicações e divisões. Pretende-se remodelar o computador com *hardware* mais rápido para a realização das operações de multiplicação e divisão. Calcule quanto mais rápidas devem ser as operações de multiplicação e divisão para que o programa seja:

a) três vezes mais rápido;

b) quatro vezes mais rápido.

a) Aplicando a lei de Amdahl, o aumento de rapidez do programa é dado por

$$s_{total}(f) = \frac{1}{(1-f) + \dfrac{f}{s}}$$

onde f representa a fração de tempo gasto na execução da parte melhorada e s traduz o aumento de rapidez dessa parte. Assim,

$$3 = \frac{1}{(1-0{,}75) + \dfrac{0{,}75}{s}} = \frac{1}{\dfrac{1}{4} + \dfrac{3}{4 \times s}} = \frac{4 \times s}{s+3}$$

resultando

$$3 \times s + 9 = 4 \times s \Leftrightarrow s = 9$$

isto é, as operações devem ser 9 vezes mais rápidas.

b) Da mesma forma, resulta

$$4 = \frac{1}{(1-0{,}75) + \dfrac{0{,}75}{s}} = \frac{1}{\dfrac{1}{4} + \dfrac{3}{4 \times s}} = \frac{4 \times s}{s+3}$$

e

$$4 \times s + 12 = 4 \times s$$

que não tem solução, concluindo-se ser impossível o programa ficar 4 vezes mais rápido.

3.2 Exercícios propostos

Exercício 9

A tabela apresentada mostra a frequência de funcionamento de 3 processadores, o número de instruções executadas em cada um deles e o respetivo tempo de execução.

Processador	F_{CLK} (GHz)	Nº de instruções	Tempo (s)
P_1	2	20×10^9	7
P_2	1,5	30×10^9	10
P_3	3	90×10^9	9

a) Calcule o número de instruções executadas por ciclo de relógio, por cada processador.

b) Calcule a frequência a que o processador P_2 deve funcionar para que o tempo de execução das instruções seja igual ao apresentado para P_1.

c) Calcule o número de instruções a executar em P_2 de modo a que o seu tempo de execução seja igual ao de P_3.

Exercício 10

Considere que dois processadores distintos, P_1 e P_2, implementam o mesmo conjunto de instruções e que estas se podem enquadrar em quatro classes diferentes (A, B, C e D). A frequência e o CPI por classe de instruções de cada implementação encontram-se na tabela.

Processador	F_{CLK} (GHz)	CPI A	CPI B	CPI C	CPI D
P_1	1,5	1	2	3	4
P_2	2	2	2	2	2

a) Seja um programa com 1 milhão de instruções divididas pelas quatro classes da seguinte forma: 10 % são da classe A, 20 % são da classe B, 50 % são da classe C e 20 % são da classe D. Verifique qual dos processadores é mais rápido a executar o programa.

b) Calcule o CPI médio de cada processador.

c) Calcule o número de ciclos de relógio necessários à execução do programa em cada processador.

Exercício 11

Dois processadores, P_1 e P_2, implementam de forma diferente o mesmo conjunto de instruções, composto por cinco classes de instruções. P_1 funciona com um relógio de 4 GHz e P_2 funciona a 6 GHz. O número médio de ciclos de relógio para cada classe de instruções é dado pela tabela que se segue.

Classe	CPI de P_1	CPI de P_2
A	1	2
B	2	2
C	3	2
D	4	4
E	3	4

O desempenho de pico *(peak performance)* é definido como a cadência máxima a que um computador pode executar a sequência de instruções mais favorável (para uma dada tarefa). Determinar o desempenho de pico de P_1 e P_2 expresso em instruções por segundo. Comentar o resultado.

Exercício 12

A tabela seguinte apresenta o número de operações em vírgula flutuante (VFL) executadas em três programas diferentes e o respetivo tempo de execução em três computadores A, B e C, distintos.

Programa	Nº operações VFL	Tempo de execução (s)		
		Comp. A	Comp. B	Comp. C
1	5×10^9	2	5	10
2	20×10^9	20	20	20
3	40×10^9	200	50	15

a) Mostre qual o computador mais rápido em termos do tempo total de execução.

b) Determine quantas vezes esse computador é mais rápido do que os outros dois.

Exercício 13

A tabela mostra o número total de instruções executadas por um programa, bem como a sua distribuição por tipo.

Total	Aritméticas	*Store*	*Load*	Salto
700	500	50	100	50

a) Assuma que o tempo de execução das instruções aritméticas corresponde a 1 ciclo de relógio, *loads* e *stores* correspondem a 5 ciclos de relógio e as instruções de salto correspondem a 2 ciclos de relógio. Calcule o tempo de execução do programa num processador a 2 GHz.

b) Calcule o valor do CPI para o programa.

c) Se o número de instruções de *load* for reduzido para metade, quanto mais rápida é a execução do programa e qual o valor do CPI?

Exercício 14

Um processador executa um programa constituído por vários tipos de instruções, entre as quais existem instruções de transferência de dados. Supondo que é possível melhorar o tempo de execução das instruções de transferência de dados em 11 vezes, calcule a percentagem do tempo de execução gasto por essas instruções de forma a conseguir-se melhorar o desempenho do processador em 5 vezes.

Exercício 15

Medindo o tempo de acesso a disco verificou-se que 80 % do tempo de execução de um determinado programa num computador era gasto em acessos ao disco. Substituindo o disco de tal computador constatou-se que este era duas vezes mais rápido.

a) Calcule quanto melhorou o desempenho do computador.

b) Verifique qual a melhoria de desempenho que seria alcançada se o disco tivesse um tempo de acesso desprezável.

Exercício 16

Suponha que se pretende melhorar um computador acrescentando-lhe uma unidade vetorial. (Estas unidades implementam instruções específicas para o tratamento de vetores.) Um cálculo executado em modo vetorial é 10 vezes mais rápido que o normal.

a) Desenhe um gráfico que mostre o ganho de rapidez (*speedup*) em função da percentagem de tempo de execução em modo vetorial (percentagem de vetorização).

b) Que percentagem de vetorização é necessária para se obter um ganho de rapidez de 2?

c) Considerar o caso da alínea anterior. Com a utilização da unidade vetorial, o computador demora metade de tempo a realizar os cálculos. Em que percentagem desse novo tempo (mais curto) é que a unidade vetorial está a ser usada?

d) Que percentagem de vetorização é necessária para obter metade do máximo ganho de rapidez possível?

e) Suponha que se determinou empiricamente que a percentagem de vetorização é de 70 %. Os projetistas de *hardware* acham que é possível duplicar o desempenho da unidade vetorial. Em alternativa, o grupo de compilação poderia tentar aumentar a utilização do modo vetorial, como forma de aumentar o desempenho. Que aumento da percentagem de vetorização (em relação à situação atual) seria necessário para obter o mesmo resultado que a duplicação do desempenho de *hardware*? Que alternativa é mais aconselhável?

Exercício 17

Um processador suporta 4 classes de instruções. Durante a execução de um dado programa, a percentagem de tempo gasta com instruções de cada classe foi o seguinte:

Classe	A	B	C	D
Tempo (%)	30	20	40	10

Uma nova versão do processador executa instruções de classe A três vezes mais rapidamente e instruções de classe C quatro vezes mais rapidamente. Determinar o ganho de rapidez *(speedup)* obtido pela nova versão.

Soluções dos exercícios propostos

1 Organização de um CPU

Exercício 6

a) `STUR X0,[X5,#-8]`

b) `LDUR X3,[X1,#4]`

Exercício 7

Ter em atenção que `PcSrc = AND(Zero, Branch)`.

a) `RegWrite=1, MemRead=1, ALUSrc=1, MemWrite=0, MemtoReg=1, PCSrc=0, ALUOp=00, Reg2Loc=X.`

Todos os componentes realizam trabalho útil, exceto `Shift left 2` e o somador para endereço de saltos (`Add`). O valor `Read data 2` também não é utilizado.

b) `RegWrite=0, MemRead=0, ALUSrc=1, MemWrite=1, MemtoReg=X, PCSrc=0, ALUOp=00, Reg2Loc=1.`

Todos os componentes realizam trabalho útil, exceto `Shift left 2`, somador para endereços de saltos (`Add`) e multiplexador `MemtoReg`. utilizado.

c) `RegWrite=0, MemRead=0, ALUSrc=0, MemWrite=0, MemtoReg=X, ALUOp=01, Reg2Loc=1.`

$$\texttt{PCSrc} = \begin{cases} 0 & \text{se } \texttt{X9} \neq 0 \\ 1 & \text{se } \texttt{X9} = 0 \end{cases}$$

Todos os componentes realizam trabalho útil, exceto memória de dados (`D-Mem`) e multiplexador `MemtoReg`. O valor `Read data 1` também não é utilizado.

Exercício 8

a) Instrução `CBZ`.

b) Instrução `STUR`.

Exercício 9

Acrescentar um multiplexador que receberá na entrada 0 a saída proveniente do multiplexador controlado por `PCSrc` e na entrada 1 a saída `Read data 1` do banco de registos. Este novo multiplexador deverá ser controlado por um sinal adicional de controlo, `BranchReg`, tal que `BranchReg=1` se opcode=`11010110000` e `BranchReg=0` no caso contrário.

Exercício 10

a) `0xF8428228`.

b) 17. Sim.

c) Indefinido (`Reg2Loc=X`). Não.

d) 8. Sim.

e) `MemRead=1, MemWrite=0`.

f) `0xB4FFFFEB`; 31, não; 11, sim; 11, não; `MemRead=0, MemWrite=0`.

Exercício 11

a) `I-Mem` → `Regs` → `ALU` → `D-Mem` → `Mux` . Tempo: 1100 ps.

b) `I-Mem` → `Control` → `Mux` → `Regs` → `Mux` → `ALU` → `Mux (PCSrc)`. Tempo: 910 ps.

Exercício 12

a) 320 ps.

b) `RegWrite`. 700 ps para a instrução mais lenta `(LDUR)`.

c) `Reg2Loc`. 120 ps. Evita que `STUR` demore mais que 1100 ps, correspondente à instrução mais lenta (`LDUR`).

2 Memória Cache

Exercício 3

a) Etiqueta: 12 bit; índice: 4 bit; espaço de endereçamento: `0x00000 - 0x3ffff`; capacidade máxima: 256 KiB.

b)
1. Lê valor `0x6198fa34` de memória *cache*.
2. Lê valor de memória principal e coloca-o na posição 5, com v=1 e etiqueta `0x8d1`.
3. Lê valor de memória principal e coloca-o na posição 12, com v=1 e etiqueta `0xfde`.
4. Escreve valor `0x1212abab` na posição 1, v=1, e atualiza a memória principal no endereço `0x048c4`.
5. Não altera memória *cache* (não existe informação em *cache* para este endereço, porque a etiqueta é diferente); coloca valor `0x00001111` em memória principal no endereço `0x3f7d0`.
6. Não altera memória *cache* (não existe informação válida em *cache* para este endereço); coloca valor `0xaaaabbbb` em memória principal no endereço `0x1dde8`.

Exercício 4

a) O campo `d` indica se o valor de conteúdo em memória *cache* é diferente do valor em memória principal.

b) A determinação do índice e da etiqueta faz-se como no problema anterior.

1. Índice 4, etiqueta `0xb7c` e `v=1`: *read hit*. Ler valor `0x6198fa34` de memória *cache* (bloco 4).
2. Endereço igual ao da alínea anterior: *write hit*. Como `d=0`, não é necessário fazer *write back*. O valor `0x33334444` é escrito na memória *cache* (bloco 4), `v=1`, `d=1`, etiqueta `0xb7c`.
3. Índice 9, etiqueta `0x431`, `v=1`: *write miss*. Como `v=1` e `d=1`, é necessário fazer *write back*: escrever o valor `0xffffeffe` (posição 9) em memória (endereço `0x21de4`). O valor `0x9999aaaa` é colocado na posição 9 da memória *cache*, com `v=1`, `d=1`, etiqueta `0x431`.
4. Índice 5, etiqueta `0x8d1` e `v=0`: *read miss*. Como `v=0`, não faz *write back* (valor de `d` não interessa neste caso). Ler valor de memória principal e colocá-lo na posição 5, com `v=1`, `d=0`, etiqueta `0x8d1`.
5. Índice 12, etiqueta `0xfde`, `v=0`: *read miss*. Como `v=0`, não faz *write back*. Ler valor de memória principal e colocá-lo na posição 12, com `v=1`, `d=0`, etiqueta `0xfde`.
6. Índice 9, etiqueta `0x877`. O bloco 9 foi alterado pelo terceiro acesso, pelo que a etiqueta é diferente (etiqueta atual do bloco `0x431`). Como `v=1`, ocorre um *write miss*. Como `d=1`, é necessário efetuar *write back*: escrever o valor valor `0x9999aaaa` na posição de endereço `0x00010c64`. Escrever valor `0xbbbb7777` no bloco 9 da memória *cache*, `v=1`, `d=1`, etiqueta `0x877`.
7. Índice 7, etiqueta `0xabf`, `v=1`: como as etiquetas não coincidem, ocorre *write miss*. Como `d=0`, não existe necessidade de fazer *write-back*. Colocar o valor `0x1212abab` em memória *cache* (bloco 7) com `v=1`, `d=1`.
8. Índice 12, etiqueta `0xfde`, `v=1` (mesmo endereço que no quinto acesso, que alterou este bloco): *read hit*. Basta atualizar valor em memória *cache*: escreve valor `0x00001111` no bloco 12 com `v=1`, `d=1`, etiqueta `0xfde` (mantém etiqueta).

Exercício 5

a) 27 bits

b) `0x2a5d04aa`

c) O conteúdo do bloco 0 é alterado para `2a 5d 04 aa 00 00 00 00`.

d) `0x000000cc`

Exercício 6

a) P1: 1,61 GHz; P2: 1,52 GHz.

b) P1: 8,6 ns; P2: 6,26 ns.

c) Ter em atenção que existem $1 + 0{,}36$ acessos a memória por instrução.
P1: CPI$=1 + 0{,}114 \times 1{,}36 \times 70/0{,}62=18{,}5$; P2: CPI$=12{,}54$.

Exercício 7

a) 90 faltas/10^3 instruções.

b) 6,43 %.

c) Um acesso a memória custa 80 ciclos (período de relógio: 1 ns).

i) $\text{CPI}_{\text{efetivo}} = 1{,}2 + 1{,}4 \times 80 = 113{,}2$ ciclos/instrução.

ii) $\text{CPI}_{\text{efetivo}} = 1{,}2 + 0{,}0643 \times 1{,}4 \times 80 = 8{,}4$ ciclos/instrução.

3 Desempenho

Exercício 9

a) $\text{IPC}_{P_1} = 1{,}43$ (instruções/ciclo)
$\text{IPC}_{P_2} = 2{,}00$ (instruções/ciclo)
$\text{IPC}_{P_3} = 3{,}33$ (instruções/ciclo)

b) 2,1 GHz

c) 27×10^9

Exercício 10

a) P_1: $t_{exec} = 1{,}87$ ms $\quad$ P_2: $t_{exec} = 1{,}00$ ms
P_2 é o processador mais rápido.

b) $\text{CPI}_{P_1} = 2{,}8$ (ciclos/instrução)
$\text{CPI}_{P_2} = 2{,}0$ (ciclos/instrução)

c) P_1: $2{,}8 \times 10^6$ $\quad$ P_2: 2×10^6

Exercício 11

P_1: 4×10^9 (instruções/segundo) $\quad$ P_2: 3×10^9 (instruções/segundo)

Exercício 12

a) C é o computador mais rápido (45 s).

b) C é 1,67 vezes mais rápido que B e 4,93 vezes mais rápido que A.

Exercício 13

a) $t_{exec} = 675$ ns

b) CPI = 1,93 (instruções/segundo)

c) A execução é 1,23 mais rápida e CPI = 1,69

Exercício 14

88 %

Exercício 15

a) 1,67 b) 5

Exercício 16

a) Desenhar o gráfico da função

$$S(f) = \frac{1}{\frac{f}{10} + (1 - f)} = \frac{10}{10 - 9 \times f} \qquad 0 \leq f \leq 1$$

b) $f = \frac{5}{9} \approx 0{,}56 = 56\,\%$

c) Percentagem de tempo gasto em modo vetorial após introdução da alteração: $\frac{1}{9} \approx 11\,\%$

d) *Speedup* máximo: 10; para obter *speedup* de 5 é preciso que $f = \frac{8}{9} \approx 88{,}9\,\%$

e) Duplicação de desempenho da unidade vetorial: $S \approx 2{,}99$. Sem duplicação de desempenho, o mesmo *speedup* é obtido para $f \approx 0{,}74 = 74\,\%$. Provavelmente é mais fácil passar de 70 % para 74 % do que duplicar o desempenho da unidade vetorial.

Exercício 17

O ganho de rapidez da nova versão é 2.