Nº de ordem: _____

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação
Arquitetura e Organização de Computadores
Teste 2

1º ano
2016-01-27
Duração 2:00
Sem consulta

Nome: _____ **Nº de estudante:** _____

Atenção: Este teste tem 13 questões em 5 páginas, num total de 200 pontos.

Parte I — Questões de Escolha Múltipla

Cada questão tem uma resposta certa. Respostas erradas não descontam.
As respostas às questões de escolha múltipla devem ser assinaladas com × na grelha seguinte.
Apenas as respostas indicadas na grelha são consideradas para efeitos de avaliação.

	Questão									
Opção	1	2	3	4	5	6	7	8	9	10
A				×		×				
B					×				×	
C			×				×	×		×
D	×	×								

Pontos: _____ / 100

[10] 1. O código de uma instrução é `0x01204020`. Esta instrução não utiliza o registo:

A. `$zero` B. `$t0` C. `$t1` **D.** `$t2`

[10] 2. Pretende-se executar a instrução `add $t7, $zero, $t1`, mas devido a uma falha na unidade de controlo, o sinal MemtoReg é sempre 1. Qual das seguintes opções faz o mesmo que o `add` pretendido?

A. `addi $t7, $t1, 0`

B. `sub $t7, $t1, $zero`

C. `lw $t7, 0($t1)`

D.
```
sw $t1, 0($t5)
lw $t7, 0($t5)
```

[10] 3. Um componente útil para a execução da instrução `addi` é:

A. Data memory B. Shift left 2 **C. Sign extend** D. Add que sucede Shift left 2

[10] 4. Numa implementação do MIPS a execução das instruções `lw` e `sw` demora, respetivamente, 880 ps e 860 ps. A latência do componente Mux é:

A. 20 B. 10 C. 0 D. 20/3

[10] 5. Apresente (em hexadecimal) o código-máquina da instrução:

```
addi $t0, $t1, -1
```

A. `0x2128fffe` **B.** `0x2128ffff` C. `0x21280001` D. `0x01281111`

[10] 6. Um programa científico despende 80 % do seu tempo a executar operações de vírgula flutuante. Para tornar o programa 4 vezes mais rápido pretende-se usar uma nova uma unidade de vírgula flutuante. Quanto mais rápida que a anterior deve ser essa unidade?

A. 16 B. 8 C. 12 D. É impossível.

[10] 7. Assuma as seguintes condições inicais:

`$t0 = 0x8abc7520 e $t2 = 0x11110000`

Determine o valor do registo `$t0` após ser realizada a instrução `or $t0, $t0, $t2`

A. `0x8abc0000` B. `0x9bbd5702` **C.** `0x9bbd7520` D. `0x8bbc5720`

[10] 8. Uma memória *cache* tem 32 blocos com 8 palavras por bloco e etiquetas de 16 bits. Quantos bits tem cada endereço?

A. 24 B. 21 **C. 26** D. 32

[10] 9. Um processador funciona a 3 GHz. Nesse processador, o programa P1 apresenta um CPI médio de 2,4 e o programa P2 apresenta um CPI médio de 2. Sabendo que o tempo de execução de cada um dos programas é 2 s, indique a afirmação verdadeira.

A. O programa P2 executa menos instruções que o programa P1.

B. O programa P2 executa 3×10^9 instruções.

C. Se a frequência baixar para 2 GHz, o programa P2 passa a ser mais rápido que P1.

D. Se o CPI de P1 aumentar, P2 fica mais lento que P1.

[10] 10. O desempenho de uma memória *cache* unificada usada com um CPU de 1 GHz foi considerado insuficiente. Sabendo que o CPI de base é 1, indique a alteração que leva à maior redução do CPI efetivo.

A. Baixar a taxa de instruções *load/store* de 50 % para 20 %.

B. Baixar o tempo de acesso à memória externa de 100 ns para 80 ns.

C. Baixar a taxa de faltas de 15 % para 10 %.

D. Nenhuma das outras alterações indicadas reduz o CPI efetivo.

(Continua)

Nome: _____ **Nº de estudante:** _____

Parte II — Questões de Resposta Aberta

Atenção: Responder diretamente no enunciado. **Justificar** todas as respostas.

11. A sub-rotina `substitui` procura a primeira ocorrência de um número N numa sequência, substituindo esse numero por 0 (zero).

    Os parâmetros da sub-rotina são, por ordem, os seguintes: 1) endereço-base da sequência; 2) número de elementos da sequência; 3) valor de N.

[15] (a) Completar o código da sub-rotina tendo em atenção as convenções relacionadas com o uso de registos.

```
substitui:  beq   $a1, $zero, final        # terminar?
            __lw__   $t0, 0( $a0 )         # obter um valor da sequência
            beq   __$t0__, $a2, L1         # é o valor procurado?
            addi  $a0, $a0, __4__          # preparar próxima iteração
            addi  $a1, $a1, __-1__
            j     substitui
L1:         __sw__    $zero, __0__($a0)    # substituir valor na sequência
final:      jr    __$ra__                  # retornar
```

[10] (b) Supondo que a sequência é {12, 56, 17, 21, 72, 7} e que N=21, determinar quantas instruções são executadas pela sub-rotina `substitui`.

Quando o valor do elemento da sequência é diferente de N, uma iteração do ciclo executa as seis primeiras instruções. Quando esse valor é igual a N são executadas mais cinco instruções: três para terminar a iteração e as duas instruções a seguir à etiqueta `L1`.

Com N=21 há a considerar 3 elementos diferentes de N no início da sequência. Logo, o número de instruções executadas é $3 \times 6 + 5 = 23$

[15] (c) Suponha que o programa a que pertence a sub-rotina anterior é executado em dois computadores A e B. O período do sinal do relógio dos computadores A e B é 300 ps e 400 ps respectivamente, e o numero de ciclos de relógio consumidos por instrução (CPI) é 4 no computador A e 2,5 no computador B. Mostre qual dos computadores é o mais rápido a executar o programa.

$$t_{exec_A} = N_{instr} \times 4 \times 300 = 1200 \times N_{instr} \text{ ps}$$

$$t_{exec_B} = N_{instr} \times 2{,}5 \times 400 = 1000 \times N_{instr} \text{ ps}$$

Então,

$$\frac{t_{exec_A}}{t_{exec_B}} = \frac{12}{10} = 1{,}2$$

O computador B é o mais rápido.

12. A seguinte sequência de instruções está em memória a partir do endereço `0x00000014`.

```
        bne $t0, $zero, segue
        lw  $t1, 0($a0)
segue:  add $t2, $t2, $t1
```

[10] (a) Complete a tabela seguinte com o valor dos sinais de controlo para a execução da instrução `lw`.

RegWrite	RegDst	PCSrc	ALUSrc	MemWrite	MemtoReg
1	0	0	1	0	1

[10] (b) Relativamente à instrução `bne`, determine o valor decimal dos sinais Read data 2 e entrada 1 do *Mux* controlado por PCSrc. Estes valores são úteis para a execução da instrução?

O sinal Read data 2 tem o valor do registo identificado pelos bits [20:16] do código da instrução `bne`. Como se trata do registo `$zero`, o seu valor é 0.
A entrada 1 do *Mux* controlado por PCSrc corresponde ao endereço da instrução `add` (a executar se `$t0` $\neq$ `$zero`). O valor deste sinal resulta da adição do endereço da instrução seguinte (`lw`) com 4 vezes o número de instruções a avançar para chegar à instrução com etiqueta `segue`. Sabendo que o endereço de `bne` é 20, o valor que assim resulta é $(20 + 4) + 4 \times 1 = 28$.

Read data 2 é utilizado na comparação do conteúdo dos registos declarados na instrução. A entrada 1 do *Mux* controlado por PCSrc também é utilizada, definindo o endereço da próxima instrução quando o salto é tomado.

[10] (c) Assuma que o caminho crítico das instruções `lw` e `add` inclui o banco de registos e que o tempo de execução é, respetivamente, 850 ps e 600 ps. Determine a latência de *Data memory* sabendo que é 11 vezes a latência de *Mux*.

Nestas condições, o caminho crítico das instruções `lw` e `add` é, respetivamente,

$$\textit{Instruction memory} \rightarrow \textit{Registers} \rightarrow \textit{ALU} \rightarrow \textit{Data memory} \rightarrow \textit{Mux}$$

e

$$\textit{Instruction memory} \rightarrow \textit{Registers} \rightarrow \textit{Mux} \rightarrow \textit{ALU} \rightarrow \textit{Mux}.$$

Tendo em consideração os componentes comuns do caminho crítico das instruções, conclui-se que

$$\mathrm{t}_{\texttt{lw}} = \mathrm{t}_{\texttt{add}} - \mathrm{t}_{\textit{Mux}} + \mathrm{t}_{\textit{Data memory}} = \mathrm{t}_{\texttt{add}} + 10 \times \mathrm{t}_{\textit{Mux}}$$

Pelo que

$$850 = 600 + 10 \times \mathrm{t}_{\textit{Mux}} \Leftrightarrow \mathrm{t}_{\textit{Mux}} = 25\,\mathrm{ps}$$

$$\mathrm{t}_{\textit{Data memory}} = 11 \times 25 = 275\,\mathrm{ps}$$

13. Um CPU tem endereços de 20 bits e usa uma memória *cache* com 1 palavra/bloco. A memória *cache* do tipo *write-back* usa 6 bits para cada índice e 12 bits para cada etiqueta. Uma parte do conteúdo da memória *cache* está indicada (em hexadecimal) na tabela seguinte:

	conteúdo	etiqueta	v	d
0	12345678	ABC	1	1
1	6548FEAB	123	0	0
2	3C1F56FD	678	1	0
3	AFD12498	567	1	1
4	6198FA34	B7C	1	0
5	1929AAAA	8D1	0	1
...	...	...	...	...

[10] (a) Determinar o número de blocos e o número total de bits usados na memória *cache*.

Como o número de bits do índice é 6, o número de blocos é $2^6 = 64$.
Cada bloco tem 32 bits de dados, 12 bits de etiqueta, 1 bit de validade e 1 bit de atualização (d), o que dá um total de 46 bits por bloco.
Logo, o número total de bits é $64 \times 46 = 2944$.

[10] (b) Determinar, se possível, a posição em memória do valor `0x6198FA34`.

O valor está no bloco 4 da memória *cache* com etiqueta *0xb7c*. Este valor é válido ($v = 1$) e igual ao valor em memória principal ($d = 0$).
Portanto, o valor `0x6198fa34` está na posição `1011 0111 1100 | 0001 00 | 00`, i.e., `0xB7C10`.

[10] (c) Determinar, se possível, qual o valor atualmente armazenado na posição de memória `0x5670C`.

O endereço em binário é `0101 0110 0111 | 0000 11 |00`. Logo, se a memória *cache* contiver o conteúdo desejado será no bloco 3.
O bloco 3 é válido porque $v = 1$ e a sua etiqueta `0x567` é idêntica à do endereço indicado.
Contudo, como $d = 1$, o seu conteúdo é (potencialmente) diferente do valor atual ainda guardado em memória.
Logo, não é possível determinar o valor ainda armazenado na memória principal (o valor em memória *cache* é mais recente)..