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
2015-01-28
Duração 2:00
Sem consulta

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

Atenção: Este teste tem 15 questões em 4 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	11	12
A				×					×			
B	×										×	
C		×	×			×				×		
D					×		×	×				×

Pontos: _____ / 120

[10] 1. Pretende-se ler o 3º elemento de uma sequência (de palavras de 32 bits) residente em memória, e cujo endereço base está no registo `$s3`. A instrução que realiza esta operação é:

A. `lw $t0, 12($s3)` **B.** `lw $t0, 8($s3)`
C. `sw $t0, 3($s3)` D. `lw $t0, 3($s3)`

[10] 2. A instrução MIPS `add $s3, $s2, $t2` é representada em linguagem-máquina por:

A. `0x024a9822` B. `0x026a9020` **C.** `0x024a9820` D. `0x026a9022`

[10] 3. Inicialmente tem-se `$t0 = 0xABAB00CC` e `$t1 = 0xFFFF0000`. Qual é o valor de `$t0` após a execução da seguinte sequência de instruções:

```
and $t2, $t0, $t1
sra $t0, $t2, 16
```

A. `0xABAB0000` B. `0x0000ABAB` **C.** `0xFFFFABAB` D. `0xABABFFFF`

[10] 4. Considere a seguinte sequência de instruções:

```
ciclo:  slt  $t3, $t2, $zero
        bne  $t3, $zero, L1
        addi $t4, $t4, 1,
        addi $t2, $t2, -5
        j    ciclo
L1:
```

Qual o valor de `$t2` que garante que são executadas exatamente 22 instruções?

A. 18 B. 20 C. 14 D. 22

[10] 5. Durante o processamento da instrução `addi` (add immediate) tem-se:

A. `MemRead=0` e `ALUSrc=0`
B. `MemWrite=1` e `MemRead=0`
C. `RegDst=1` e `MemWrite=0`
D. `RegDst=0` **e** `MemtoReg=0`

[10] 6. Considere o seguinte fragmento de código *assembly*:

```
      beq $t2, $t3, L1    # salto condicional na posição 0x00002014
      ...                 # uma instrução
      ...                 # uma instrução
L1:   ...                 # uma instrução
```

A instrução `beq` está guardada em memória na posição `0x0002014`. Quando é executada a instrução `beq`, qual é o valor presente na entrada inferior do *multiplexer* controlado pelo sinal `PCSrc`?

A. `0x00002018` B. `0x0000000C` **C.** `0x00002020` D. `0x00002024`

[10] 7. Que instrução usa o componente Shift left 2?

A. `addi` B. `lw` C. `add` **D.** `beq`

[10] 8. Para que instrução é que se tem `ALUSrc=0`?

A. `lw` B. `sw` C. `addi` **D.** `sub`

[10] 9. Uma memória *cache* tem 32 blocos de 4 bytes. Cada etiqueta tem 16 bits Quantos bits tem cada endereço?

A. 23 B. 20 C. 21 D. 18

[10] 10. Considere uma memória *cache* de instruções com 8 blocos de 4 bytes e acesso por mapeamento direto. Supondo que a memória *cache* está inicialmente vazia, indique a sequência de acessos consecutivos que causa 4 faltas.

A. `0xABC4, 0x31E4, 0x31E4, 0x31E4`
B. `0xABC4, 0x31E4, 0x31E4, 0xABC4`
C. `0xABC4, 0x31E4, 0xABC4, 0x31E4`
D. `0xABC4, 0xABC4, 0x31E4, 0x31E4`

[10] 11. Considere um CPU com endereços de 20 bits ($A_{19} \ldots A_0$). O sistema de memória correspondente tem 3 circuitos do tipo RAM (barramento de dados com 8 bits). Dois circuitos RAM ocupam as gamas de endereços `0x00000-0x001FF` e `0xF0400-0xF07FF`, respetivamente.

Indique qual das seguintes funções de descodificação de endereços pode ser ligada à entrada `CS` (*chip select*) do terceiro circuito.

A. $A_{19} \cdot A_{18} \cdot A_{16}$
B. $A_{19} \cdot \overline{A_{18}} \cdot A_{16}$
C. $A_{19} \cdot A_{18} \cdot A_{17}$
D. $\overline{A_{19}} \cdot \overline{A_{18}}$

[10] 12. Um circuito ROM tem endereços de 16 bits e 8 bits de dados por posição. Qual é a capacidade total do circuito?

A. 32 KiB B. 16 KiB C. 128 KiB **D. 64 KiB**

Parte II — Questões de Resposta Aberta

Atenção: Responder diretamente no enunciado.

[30] 13. A sub-rotina `contar` determina o número de vezes que um dado valor ocorre numa sequência de palavras de 32 bits.

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

Completar o código da sub-rotina tendo em atenção as convenções relacionadas com o uso de registos.

```
contar:   move   __$v0__, $zero            # inicializa contador
loop:     beq    $a1, $zero, final         # terminar?
          lw     $t0, 0(__$a0__)           # obter um valor da sequência
          bne    __$t0__, $a2, L1          # contabilizar?
          addi   $v0, $v0, 1
L1:       addi   $a0, $a0, __4__           # preparar próxima iteração
          addi   $a1, $a1, __-1__
          j      loop
final:    jr     __$ra__                   # retornar
```

14. A tabela seguinte indica a latência máxima dos blocos usados na implementação do CPU MIPS.

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

Assumir que todos os outros componentes têm latência zero.

[10] (a) Suponha que a latência do *multiplexer* controlado pelo sinal `RegDst` passa de 30 ps para o dobro. Explicar se o tempo mínimo de execução das instruções `add` e `lw` é alterado e, em caso afirmativo, de que forma?

O tempo mínimo de execução não é alterado, embora o valor da entrada `RegWrite` do banco de registos fique pronto mais tarde (560 ps em vez de 530 ps). A entrada `WriteData` fica pronta significativamente mais tarde, já que, por exemplo, a leitura de registos (necessária para ambas as instruções) apenas está completa após 600 ps.

[10] (b) Repetir a alínea anterior, mas supondo agora que é antes a latência do *multiplexer* controlado pelo sinal `ALUSrc` que passa de 30 ps para o dobro.

O tempo mínimo de execução de `add` é alterado, porque a segunda entrada da ALU é atrasada (de 630 ps para 660 ps) e esta é a última entrada da ALU a ficar pronta (está no caminho crítico, já que o controlo da ALU está pronto após 500 ps e a primeira entrada está pronta após 600 ps). O tempo mínimo de execução de `lw` não é alterado: neste caso a segunda entrada da ALU fica pronta após 460 ps (em vez de 430 ps), mas a última entrada da ALU a ficar pronta é a mesma em ambos os casos: a entrada superior após 600 ps.

15. Um processador com $\text{CPI}_{\text{base}} = 1$ usa uma memória *cache* unificada com uma taxa de faltas de 10 %. O processador funciona a 1 GHz, enquanto um acesso a memória externa (principal) demora 60 ns. Considere que, em média, 50 % das instruções executadas são do tipo *load* ou *store*.

[10] (a) Determinar o CPI efetivo.

$$\text{CPI}_{\text{prot}} = 1{,}5 \times 0{,}1 \times \frac{60\,\text{ns}}{1\,\text{ns}} = 9$$

$$\text{CPI} = \text{CPI}_{\text{base}} + \text{CPI}_{\text{prot}} = 1 + 9 = 10$$

[10] (b) Assumindo que um dado programa executa 10^{10} instruções, calcule o respetivo tempo de execução sem e com memória *cache*.

Sem memória *cache*, $\text{CPI}_{\text{prot}} = 1{,}5 \times \frac{60\,\text{ns}}{1\,\text{ns}} = 90$. Consequentemente:

$$T_{\text{exec}} = 10^{10} \times (1 + 90) \times \frac{1}{10^9} = 910\,\text{s}$$

Com memória *cache* (usando CPI_{prot} da alínea anterior):

$$T_{\text{exec}} = 10^{10} \times (1 + 9) \times \frac{1}{10^9} = 100\,\text{s}$$

[10] (c) Determinar o número de faltas de *cache* por mil instruções.

1000 instruções $\Rightarrow (1 + 0{,}5) \times 1000 = 1500$ acessos a memória.
Em 1500 acessos, ocorrem $1500 \times 0{,}1 = 150$ faltas.
Logo, ocorrem 150 faltas / mil instruções.