**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

EIC0083
2018-01-19
Duração 1:45H
Sem consulta

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

**Atenção:** Este teste tem 13 questões em 8 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-máquina da instrução `addi $t1, $s3, -4` é:

A. `0x22690004` **B.** `0x2269fffc` C. `0x21330004` D. `0x2133fffc`

[10] 2. Assumindo que os valores iniciais dos registos `$t1` e `$t2` são respetivamente `0xABABABAB` e `0xCDCDCDCD`, qual é o valor do registo `$t1` após a execução do seguinte fragmento de código:

```
sll $t2, $t1, 14
srl $t1, $t2, 8
or  $t1, $t1, $t2
```

**A.** `0xEAEAEAC0` B. `0x00EAC000` C. `0xEAEACAC0` D. `0xEAEFCDCD`

[10] 3. Qual das seguintes afirmações sobre uma memória *cache* do tipo *write-through* é falsa?

A. O conteúdo da memória principal está sempre atualizado.

B. No caso de uma falta num acesso de leitura (*read miss*) o valor é lido da memória principal e colocado na *cache* atualizando a etiqueta e alterando o valor de **v** para 1.

**C. No caso de uma falta num acesso de escrita (*write miss*) o valor é escrito na memória principal e na memória *cache* atualizando a etiqueta e alterando o valor de v para 1.**

D. No caso de um acerto num acesso de leitura (*read hit*) o valor é lido da memória cache.

[10] 4. Um CPU está equipado com uma memória *cache* unificada cuja taxa de faltas, $t_f$, é de 10%. Assumindo que em média 20% das instruções de um programa acedem a dados (i.e., são *load* ou *store*), qual das seguintes alternativas de *split cache* apresenta o mesmo desempenho?

**A. I-*cache* com $t_f = 8\%$ e D-*cache* com $t_f = 20\%$.**

B. I-*cache* com $t_f = 20\%$ e D-*cache* com $t_f = 8\%$.

C. I-*cache* com $t_f = 5\%$ e D-*cache* com $t_f = 5\%$.

D. I-*cache* com $t_f = 10\%$ e D-*cache* com $t_f = 20\%$.

[10] 5. Devido a uma avaria, os sinais `MemRead` e `MemToReg` estão permanentemente a 1. Após executar `add $t1,$t2,$t3` resulta `$t1=4`. Este valor corresponde ao:

A. conteúdo da memória na posição 4

B. resultado de `$t2 + $t3`

C. endereço de acesso à memória

**D. conteúdo da memória na posição dada por `$t2 + $t3`**

[10] 6. Qual é o valor de `$t1` após a execução da instrução assinalada com um asterisco (*)?

```
          li  $s0, 5
          li  $a0, 10
          jal rotina
          and  $v0, $v0 $zero
          add  $t1, $v0, $s0  # (*)
          ...
rotina:   add  $v0, $a0, $a0
          addi $ra, $ra, 4
          jr   $ra
```

A. 0 **B. 25** C. 15 D. 5

[10] 7. Para um dado programa, o processador P1 com $F_1 = 1\,\text{GHz}$ apresenta o mesmo tempo de execução que o processador P2 com $F_2 = 1{,}25\,\text{GHz}$. O tempo de execução de P1 fica maior que o de P2 se:

A. se passar a usar $F_2 = 1\,\text{GHz}$

B. se aumentar o valor do CPI médio de P2

C. se reduzir o valor do CPI médio de P1

**D. se aumentar 1,3 vezes o período do relógio de P1**

[10] 8. O tempo de execução de um programa está repartido entre a execução de instruções da classe A (60 % do tempo) e da classe B (40 % do tempo).

Qual das seguintes alterações leva ao melhor desempenho?

A. diminuir para metade o tempo de execução das instruções de classe A;

B. diminuir o tempo de execução das instruções de classe B para um quarto do tempo original;

C. reduzir o tempo de execução das instruções de classe A para um terço e aumentar o tempo de execução das instruções de classe B para o dobro;

**D. reduzir 1,5 vezes o tempo de execução das instruções de classe A e reduzir o tempo de execução das instruções de classe B para metade.**

[10] 9. Um programa gasta 75 % do tempo em transferências de dados para outro computador via rede sem fios. Quantas vezes é preciso aumentar a velocidade de transferência para obter uma redução do tempo de execução do programa *(speedup)* de duas vezes?

A. 4 B. 1,5 **C. 3** D. 2

[10] 10. Considerar o seguinte fragmento de código *assembly*.

```
slt  $t2, $t1, $t0
bne  $t2, $zero, L1
```

Para que valores de `$t0` e `$t1` é que o salto condicional é tomado?

A. `$t0` $= -10$ e `$t1` $= -10$

B. `$t0` $= -10$ e `$t1` $= 0$

**C. `$t0` $= -10$ e `$t1` $= -50$**

D. `$t0` $= 10$ e `$t1` $= 50$

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

## Parte II — Questões de Resposta Aberta

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

11. O programa abaixo aplica a sub-rotina `calc` aos elementos de uma sequência `nums`, acumula os resultados das invocações e apresenta o total acumulado antes de terminar a execução.

[15] (a) Completar o programa.

```
        .data
nums:   .word  0x00000005, 0xFFF8C000, 0xA00800C
nelem:  .word  3
        .text
init:   la     $s0, nums
        __lw__ $s1, nelem              # nº de elementos a processar
        addi   $s2, $zero, 0
ciclo:  beq    $s1, __$zero__, L1      # terminar ciclo
        lw     $a0, 0($s0)
        __jal__ calc                   # invocar sub-rotina
        add    $s2, $s2, __$v0__       # usar o resultado
        addi   $s0, $s0, 4
        addi   $s1, $s1, __-1__
        __j__  ciclo
L1:     move   $a0, __$s2__
        li     $v0, 1
        syscall                        # escrever valor inteiro no monitor
        li     $v0, 10
        syscall                        # terminar execução

# Sub-rotina calc tem 1 argumento
calc:   xor    $v0, $v0, __$v0__       # inicializar $v0 com zero
LC1:    beq    $a0, $zero, LC2         # terminar?
        andi   $t0, $a0, 1
        add    $v0, $v0, $t0
        srl    $a0, $a0, 1
        j      LC1
LC2:    jr     __$ra__                 # fim da subrotina
```

[10] (b) Determinar o número de instruções executadas pela sub-rotina `calc` quando é chamada pela primeira vez.

Assumindo que as instruções de salto (condicional ou incondicional) têm CPI=2 e todas as outras têm CPI=1, determinar também o valor de CPI médio. Indicar todos os cálculos.

A sub-rotina determina quantos bits 1 compõem a representação binária do argumento que recebe em `$a0`. Para isso possui um ciclo no qual é determinado o valor do bit menos significativo. Em cada iteração o conteúdo de `$a0` é deslocado um bit para a direita. O ciclo termina quando `$a0` é zero.
Da primeira vez que a sub-rotina é chamada o argumento é 5 (0 ... 0101), pelo que, o ciclo é repetido três vezes e o número de instruções executadas é:

$1(\texttt{xor}) + 3 \times 5(\texttt{beq a j}) + 1(\texttt{beq}) + 1(\texttt{jr}) = 18$

O número de ciclos correspondente é $1 + 3 \times (2 + 1 + 1 + 1 + 2) + 2 + 2 = 26$.

CPI = nº ciclos / nº instruções = 13/9 = 1,44

[10] (c) Considerar apenas argumentos de valor $2^k$ ($k$ inteiro, $0 \leq k \leq 31$). Explicar o resultado da sub-rotina e determinar o número de instruções executadas em função de $k$.

A representação binária de um argumento com o valor $2^k$ tem um único bit 1. Assim sendo, o resultado da sub-rotina é 1.

O número de iterações realizadas é $k+1$ e o número de instruções executadas é $1+(k+1)\times 5+2 = 5 \times k + 8$.

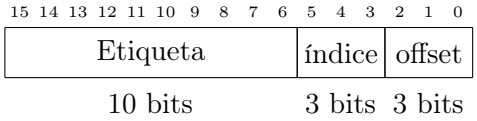
Prova continua na próxima folha

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

12. A tabela seguinte apresenta o conteúdo (em hexadecimal) de uma memória *cache* do tipo *write-back* com 8 blocos de 8 bytes usada como *D-cache* num CPU com endereços de 16 bits.

| bloco | conteúdo | | | | | | | | etiqueta | v | d |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | |
| 0 | aa | cc | de | hf | 34 | 33 | 11 | 01 | 235 | 1 | 0 |
| 1 | bb | ad | 45 | 4f | af | de | 21 | 99 | 391 | 1 | 1 |
| 2 | cc | 34 | ab | 1f | 56 | cd | ff | ff | 023 | 1 | 1 |
| 3 | dd | 67 | 22 | 2b | 32 | 56 | 32 | 21 | 198 | 0 | 1 |
| 4 | ee | 32 | 11 | 9f | aa | ba | ab | bb | 311 | 1 | 0 |
| 5 | ff | 10 | 00 | 04 | 01 | 02 | 03 | 04 | 278 | 0 | 0 |
| 6 | 11 | 03 | 41 | 32 | cc | dd | ee | ff | 212 | 1 | 1 |
| 7 | 22 | 01 | 65 | 01 | 05 | 06 | 07 | 08 | 387 | 0 | 1 |

[5] (a) Como é decomposto o endereço para acesso à memória *cache*? Justifique.




Offset: 3 bits. Como o conteúdo de cada bloco tem 8 bytes, serão necessários 3 bits para designar um byte do bloco.
Índice: 3 bits. Como a cache possui 8 blocos, são necessários 3 bits para representar os índices de 0 a 7.
Etiqueta: 10 bits. Como temos 16 bits e já gastámos 6, os restantes $(16 - 6 = 10)$ serão para a etiqueta.

[15] (b) Indique (se possível) o valor (byte) em memória principal no endereço `0xc467`. Justifique.

Decompondo o endereço temos: `0xC467` → `1100 0100 0110 0111` → `1100010001 100 111`.
Portanto, etiqueta: $311_{16}$, índice: 4, *offset*: 7.

As etiquetas são iguais e o bloco é válido porque `v=1` (*read hit*).

O valor em memória *cache* é `0xEE` (elemento 7 do bloco 4) e corresponde ao valor em memória principal porque `d=0`.

[15] (c) Explique quais as alterações que ocorrem na *cache* e na memória principal durante a leitura do valor (byte) residente no endereço `0xe48d`.

Decompondo o endereço temos: `0xE48D` → `1110 0100 1000 1101` → `1110010010 001 101`. Portanto, etiqueta: $392_{16}$, índice: 1, *offset*: 5.

Como as etiquetas são diferentes, ocorre uma falta de leitura *read miss*. Os dados do bloco são válidos (`v=1`), mas não correspondem à posição de memória pretendida.

Como `v=1` e `d=1` é necessário fazer *write-back*: o valor `0x45` é escrito na memória principal no endereço `0xE44D` (endereço calculado com a etiqueta residente em *cache*).

É necessário ler de memória principal o bloco que contém o valor no endereço `0xE48D`, escrevendo-o na posição 1 da *cache* (posição 5) e atualizando a etiqueta desse bloco para `392`. Os indicadores devem ficar com os valores `v=1` e `d=0`.

Prova continua na próxima folha

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

13. Considerar a organização interna do CPU MIPS simplificado estudada nas aulas.

[10] (a) Para que instrução (ou instruções) os sinais de controlo assumem simultaneamente os seguintes valores: `RegWrite=1` e `ALUSrc=1` ? Justifique.

Nota: a questão refere-se apenas ao **CPU simplificado**, cuja organização foi objeto de estudo.

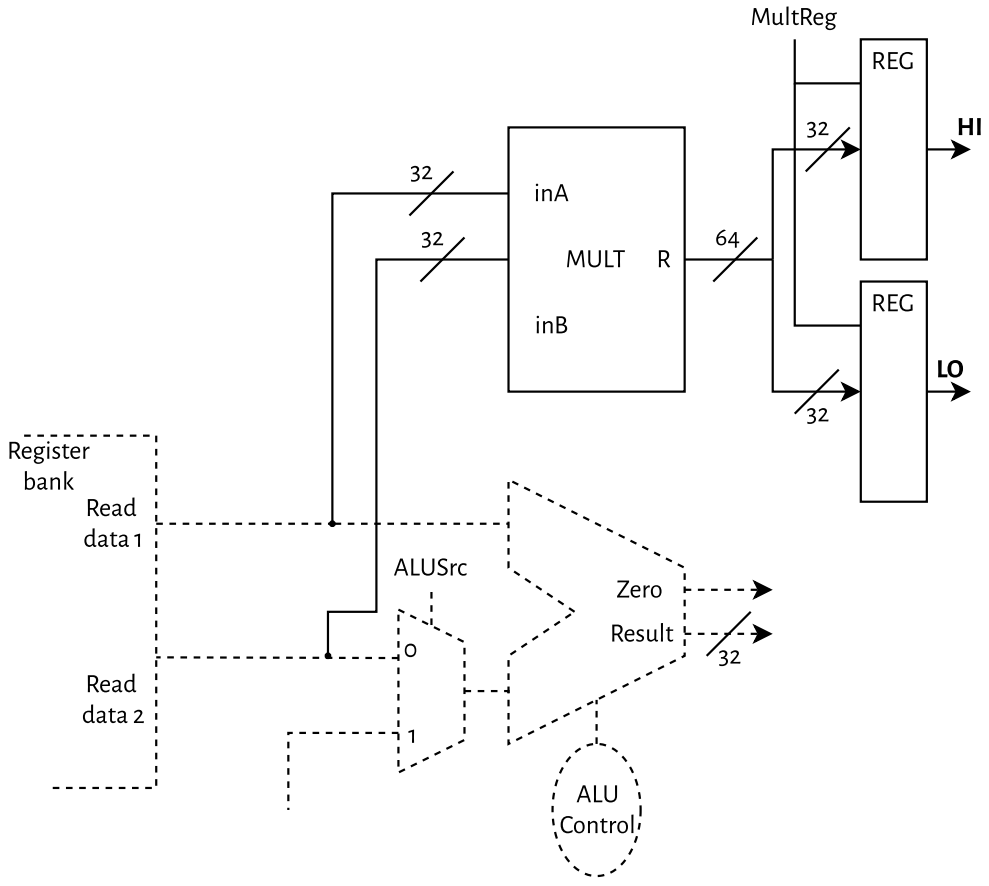
`RegWrite=1` implica que um valor é escrito numa posição do banco de registos.

`ALUSrc=1` implica que os 16 bits menos significativos do código da instrução são usados no cálculo realizado na ALU.

Instruções do tipo R não têm `ALUSrc=1` e instruções do tipo J não têm `RegWrite=1`. Das instruções tipo I, nem `beq` nem `sw` alteram o conteúdo do banco de registos.

Das instruções suportadas pelo CPU simplificado, apenas `lw` e `addi` estão na situação indicada no enunciado.

(b) Para implementar a instrução `mult`, **acrescentaram-se** ao CPU os componentes indicados a traço contínuo na figura abaixo. O módulo `MULT` é um multiplicador com operandos de 32 bits e resultado de 64 bits. O sinal `MultReg` habilita a escrita nos registos `HI` e `LO` e é gerado pelo controlador `ALU Control`. O sinal de relógio e os restantes módulos do CPU não estão representados.

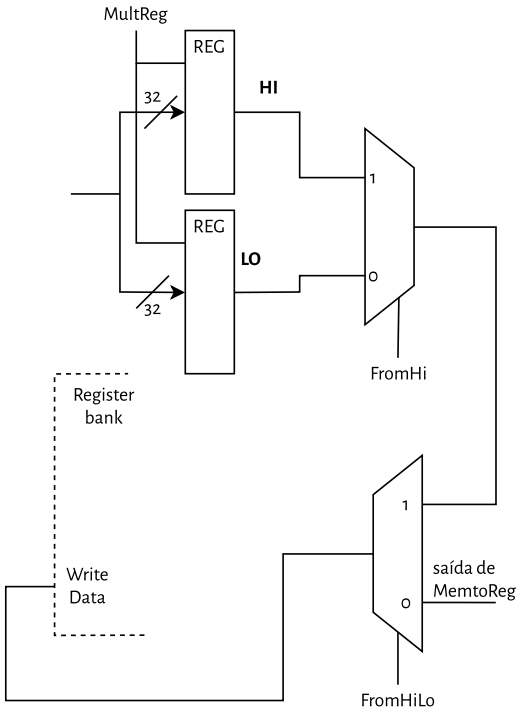


[10] i. Indique, justificando, o valor de todos os sinais de controlo do CPU durante a execução de `mult`. Indique também o valor de `MultReg` durante a execução das restantes instruções. Ter em consideração a forma como a instrução é codificada.

A instrução `mult` não altera o conteúdo do banco de registos: `RegWrite=0`, `RegDst=X` e `MemtoReg=X`.
O resultado da operação realizada na ALU não é usado (`ALUSrc=X`). Contudo, o controlador deve respeitar o valor do campo `funct` da instrução (já que esta é do tipo R), logo `ALUOp=10`$_2$.
A memória de dados não deve ser acedida, logo `MemRead=MemWrite=0`.
Como não se altera o fluxo de execução, `PCSrc=0` (equivalente: `Branch=0`).
Na execução da instrução `mult`, deve ser `MultReg=1` para que o resultado da multiplicação seja armazenado nos registos `LO` e `HI`; deve ser `MultReg=0` para todas as outras instruções suportadas pelo CPU simplificado.

[10] ii. Pretende-se acrescentar agora suporte para as instruções `mfhi` e `mflo`. Explique, com o auxílio de um diagrama, como é que isso pode ser feito e quais os valores de todos os sinais de controlo do CPU. Quaisquer sinais de controlo adicionais devem ser produzidos pelo controlador `ALU Control`.

É necessário criar percursos que levem os dados dos registos `HI` ou `LO` até à entrada `Write Data` do banco de registos.




Os valores do sinais de controlo são: `RegDst=1` (registo de destino determinado pelos bits [15:11] da instrução); `RegWrite=1` (escrita no banco de registos); `ALUSrc=X` (resultado da ALU não é usado); `ALUOp=10`$_2$ para o controlador auxiliar usar a informação do campo `funct`; `MemRead=MemWrite=0` (a memória de dados não é usada); `MemtoReg=X` (valor à saída deste multiplexador não é usado); `MultReg=0` (os registos `HI` e `LO` não são alterados). Os novos sinais têm os valores `FromHiLo=1` (valor provém dos registos especiais) e `FromHi=0` para `mflo` e `FromHi=1` para `mfhi`. Durante a execução das outras instruções, `FromHiLo=0`, `FromHi=X`.
(Nota: outra alternativa seria alterar o multiplexador associado a `MemtoReg` para ser do tipo 4:1 e necessitaria de 2 bits de seleção gerados por combinação de sinais do controlador principal e do controlador auxiliar.)

Fim do enunciado.