NOME: _____ **Nº:**_____

TEÓRICA

As questões devem ser respondidas na própria folha do enunciado. As questões 1 a 4 são de escolha múltipla, e apenas uma das respostas está correcta, valendo 1 valor. Uma resposta errada desconta 1/3 de valor. As questões 5 e 6 valem 3 valores cada.

1. Considere os barramentos que permitem a comunicação entre os vários componentes de uma máquina. Qual das seguintes afirmações é verdadeira:
   - ☐ As vantagens de um barramento síncrono são a simplicidade de implementação, a facilidade de interligarem componentes com diferentes velocidades e o facto de poderem ser bastantes longos.
   - ☐ As vantagens de um barramento assíncrono são a facilidade de interligarem componentes com diferentes velocidades e o facto de poderem ser mais longos do que os barramentos síncronos.
   - ☐ Os barramentos assíncronos são mais fáceis de implementar que os síncronos, pois não exigem que todas as operações sejam efectuadas à mesma velocidade.
   - ☐ A principal vantagem dos barramentos reside no facto de permitirem ligar um número ilimitado de componentes entre si.
2. É comum a utilização de determinados conjuntos de programas, genericamente designados por *benchmarks*, para avaliar o desempenho dos computadores. Qual das seguintes afirmações é verdadeira:
   - ☐ Os *benchmarks* sintéticos são ideais para avaliar o desempenho dos computadores, pois trata-se de programas pequenos, fáceis de executar e cujos resultados são facilmente interpretados.
   - ☐ Para que a informação dada por uma *benchmark* seja útil para um grande número de utilizadores, é necessário que este teste reproduza com alguma fidelidade a carga a que a máquina é sujeita em condições de utilização normal.
   - ☐ Os resultados obtidos com *benchmarks* específicos para determinadas operações, como o Dhrystone (int) e Whetstone (fp), podem ser utilizados com confiança para desenhar outros programas que realizem o mesmo tipo de operações.
   - ☐ A documentação rigorosa das condições em que os testes de desempenho foram realizados não é uma tarefa importante, pois os resultados são bastante insensíveis às condições de teste.
3. Considere um programa P com 10^9 instruções, em que 10% das instruções implicam um acesso à memória. Executado na máquina M, P exibe uma *miss rate* de instruções de 5%, de dados de 10% e um CPI_{CPU} de 1,2 ciclos. M tem uma *cache* com linhas de 4 palavras e um acesso à memória central tem uma latência de 60 ns mais 10 ns por palavra. Sabendo que a frequência do relógio de M é de 1 GHz, qual dos seguintes valores corresponde ao tempo de execução de P em M:
   - ☐ 6,0 seg.
   - ☐ 16,2 seg.
   - ☐ 6,2 seg.
   - ☐ 7,2 seg.
4. Considere uma máquina com endereços de 64 *bits*, uma cache de 1024 Kbytes, linhas de 32 palavras, palavras de 8 bytes e mapeamento *fully associative*. Qual a distribuição dos *bits* do endereço para seleccionar o byte correcto na cache:
   - ☐ *Tag* = 44; Índice = 12; *Block Offset* = 5; *Byte Offset* = 3
   - ☐ *Tag* = 56; Índice = 0; *Block Offset* = 4; *Byte Offset* = 4
   - ☐ *Tag* = 12; Índice = 44; *Block Offset* = 5; *Byte Offset* = 3
   - ☐ *Tag* = 56; Índice = 0; *Block Offset* = 5; *Byte Offset* = 3

NOME: _____ **Nº:**_____

5. Explique, tendo em conta a figura anexa, quais os elementos do *datapath* do MIPS que são utilizados, e por que ordem, para executar a instrução addi $s0, $t1, 1000. Indique os vários estágios de execução da instrução e indique o valor dos sinais de controlo RegDst, RegWrite, ALUSrc, PCSrc, MemWrite, MemRead e MemToReg.

NOME: _____ **Nº:**_____

6. Relativamente à hierarquia de memória:
   a. Discuta os diferentes esquemas de colocação de blocos na *cache*, indicando as vantagens e desvantagens de cada um deles.
   b. Diga o que entende por localidade e discuta de que forma esta é responsável pelo sucesso da utilização de *caches* na redução do tempo de acesso à memória.

NOME: _____ **Nº :**_____

PRÁTICA

As questões devem ser respondidas em folha separada. As questões 1 e 3 valem 4 valores cada. A questão 2 vale 2 valores.

1. Considere o seguinte código em assembly do MIPS. Os números no início de cada linha servem apenas para efeitos de referência.

```
 1:   .data 0x50000000
 2:     x: .word 7,124,6,33,55,88,23,99,100,0
 3:   .text
 4:     la $t2,x
 5:     lw $t3,0($t2)
 6:     lw $t4,8($t2)
 7:     add $t5,$t3,$t4
 8:     addi $t6,$t5,-50
 9:     bgtz $t6,r1
10:     li $s1,0
11:     b fim
12:   r1:
13:     li $s1,1
14:   fim:
15:     jr $ra
```

   a) Transforme o código de forma a incluir somente instruções nativas do MIPS.
   b) Apresente, nos formatos binário e hexadecimal, as instruções das linhas 6 e 9, sabendo que $t2=9, $t4=11e $t6=13.
   c) Se este código for executado, indique os valores dos registos $t2, $t4 e $s1 imediatamente antes da execução da instrução da linha 15.

2. Complete a seguinte rotina de atendimento a excepções, escrita para o simulador SPIM, por forma a que na posição de memória identificada pela etiqueta **nt** seja mantido o número teclas premidas desde o início do programa.

```
.kdata
nt:  .word 0

.ktext 0x80000080
     mfc0 $k1, $13
     << completar >>
     bne $k0, $0, ret

     andi $k0, $k1, 0x100
     << completar >>

     la $k0, nt
     lw $k1, 0($k0)
     << completar >>

ret: mfc0 $k1, $14
     rfe
     jr $k1
```

NOME: _____ **Nº :**_____

3. Pretende-se codificar em *assembly* do MIPS o seguinte programa:

```
struct contaB {
     char dataAct;
     int  ordem;
     int  prazo;
     int  saldo;
     } conta[10];
int  i, Nact;
char num;
main() {
     for (i=10, Nact=0; i > 10; --i)
        if (conta[i].dataAct < num)
          conta[i].saldo = soma(conta[i].ordem, conta[i].prazo);
        else
          ++Nact;
     }
```

Complete a solução dada, assumindo que a função **soma** já está codificada e utilizando a seguinte atribuição de variáveis a registos:

i ≡ $s0, Nact ≡ $s1, num ≡ $s2 , conta ≡ $s3.

```
.data
conta:
.text
     main:
     b teste
ciclo:
```

```
teste:
     lw    $ra, 16 ($sp)
     lw    $s3, 12 ($sp)
     lw    $s2,  8 ($sp)
     lw    $s1,  4 ($sp)
     lw    $s0,  0 ($sp)
     addiu $sp, $sp, 20
     jr    $ra
```