

1. [1,5 valores] - Considere o seguinte excerto de um programa escrito em *assembly* e a executar numa máquina com cache:

```
ciclo: movl 0(%ebx), %edx
      movl $10, 0(%ebx)
      addl $4, %ebx
      cmpl $0, %edx
      jnz ciclo
```

Considere que o registo `%ebx` aponta para o início de um array de inteiros (4 bytes) com os seguintes valores: `{-10, 30, 1024, -33, 0}`. Note que o ciclo termina quando o valor lido do array for 0. A frequência do relógio é de 2 GHz, o  $CPI_{CPU}$  é 2, a *miss rate* de instruções é de 3% e a de dados de 5%. Sabendo que o tempo de execução deste programa é de 150 ns, qual é a *miss penalty* (expressa em tempo)?

- |  |  |
|--|--|
| <input type="checkbox"/> $mp_T = 150$ ns | <input type="checkbox"/> $mp_T = 50$ ns  |
| <input type="checkbox"/> $mp_T = 200$ ns | <input type="checkbox"/> $mp_T = 100$ ns |

2. [1,5 valores] - Complete a afirmação abaixo :

“A técnica de *pipelining*, relativamente a uma arquitectura sequencial de ciclo único, acelera o desempenho de um processador pois ...

- ☐ resulta numa diminuição do CPI, uma vez que mais do que uma instrução se encontra em execução em cada ciclo.”
- ☐ resulta numa diminuição do número de instruções executadas, uma vez que algumas instruções são internamente transformadas em `NOPS`”
- ☐ resulta numa diminuição do período do relógio, uma vez que este deve ser apenas tão longo quanto o estágio mais demorado do *pipeline*.”
- ☐ resulta num aumento da frequência devido a ciclos de *stalling* causados por dependências de dados e/ou controlo.”

3. [1,5 valores] - Complete a afirmação abaixo:

“O programa `for (i=0 ; i<N ; i++) a[i] = b[100*i] * 2; ...`

- ☐ permite explorar a hierarquia de memória pois exhibe localidade espacial nos acessos a `i`.”
- ☐ permite explorar a hierarquia de memória pois exhibe localidade espacial nos acessos a `a[]`.”
- ☐ permite explorar a hierarquia de memória pois exhibe localidade temporal nos acessos a `a[]`.”
- ☐ permite explorar a hierarquia de memória pois exhibe localidade espacial nos acessos a `b[]`.”

4. [1,5 valores] - Quantos *bits* tem a *tag* de uma hierarquia de memória (S=1024, E=8, B=128, m=32)?

- ☐  $t = 15$  ☐  $t = 17$   
☐  $t = 10$  ☐  $t = 12$

5. [2,0 valores] A tabela abaixo apresenta na coluna da esquerda uma sequência de endereços (m=4) de acesso à memória gerados por um determinado programa. As 3 colunas seguintes referem-se a um modo de mapeamento numa cache que usa o algoritmo de substituição LRU. Preencha-as indicando em que *set*/linha (dentro do *set*) mapeia cada endereço, qual a *tag* associada a essa linha depois deste acesso e indicando se se trata de um *cold miss*, colisão ou de um *hit*. Considere a cache inicialmente fria.

Addr	(S=2,E=2,B=2,m=4)	tag	cold miss/hit/colisão
1			
13			
0			
6			
8			

6. [2,0 valores] O excerto de código abaixo calcula a soma de todos os elementos de uma matriz de inteiros. A matriz tem ALTURA \* LARGURA elementos.

```
for (col=0 ; col<LARGURA ; col++) {
    for (lin=0 ; lin < ALTURA ; lin++) {
        soma += matriz[lin*LARGURA+col];
    }
}
```

Reescreva o programa para que seja possível explorar de forma mais eficaz a hierarquia da memória, **justificando** a sua resposta.