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)?

☐ mp_T = 150 ns ☐ mp_T = 50 ns

☐ mp_T = 200 ns ☐ 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 `NOP`s"

☐ 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 exibe localidade espacial nos acessos a `i`."

☐ permite explorar a hierarquia de memória pois exibe localidade espacial nos acessos a `a[]`."

☐ permite explorar a hierarquia de memória pois exibe localidade temporal nos acessos a `a[]`."

☐ permite explorar a hierarquia de memória pois exibe localidade espacial nos acessos a `b[]`."

Nome: _____ Número:_____

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.

Nome: _____ Número:_____

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
```

acede à memória
acede à memória
não acede à memória
não acede à memória
não acede à memória

→ $\frac{2}{5}$ acedem

→ $\frac{3}{5}$ não acedem

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)?

- ☐ mp_T = 150 ns
- ☐ mp_T = 50 ns
- ☐ mp_T = 200 ns
- ☒ ☐ mp_T = 100 ns

$f = 2 \times 10^9$ Hz

$CPI_{CPU} = 2$

$mr_i = 0{,}03$

$mr_d = 0{,}05$

$T_{exec} = 150 \times 10^{-9}$ seg

$mp_T = ?$

$CPI = CPI_{CPU} + CPI_{Mem}$

$CPI_{Mem} = (mr_i + mr_d \times \%Mem) \times mp_T \times f$ (303, 0,05, $\frac{2}{5}$, $\times 2\times10^9$)

$mp_T \times f$ → se for em tempo
mp → se for em clock cycles

$T_{exec} = \frac{CPI \times \#I}{f}$ (nº de I executadas → $5 \times 5 = 25$; 5 instruções; 5 vezes que se repetem (até encontrar o 0))

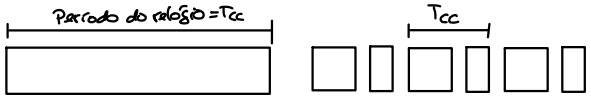
$\Leftrightarrow 150\times10^{-9} = \frac{(2+CPI_{Mem})\times 25}{2\times10^9}$

$CPI_{Mem} = (0{,}03+0{,}05\times\frac{2}{5})\times mp_T \times 2\times10^9$ → $mp_T = 100\times10^{-9} = 100$ 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 …

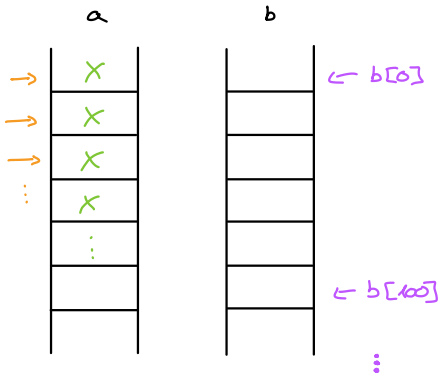
- ☒ resulta numa diminuição do CPI, uma vez que mais do que uma instrução se encontra em execução em cada ciclo." → O CPI ou mantém-se ou aumenta por causa das dependências de dados e por causa dos tempos dos registos entre os estágios
- ☒ resulta numa diminuição do número de instruções executadas, uma vez que algumas instruções são internamente transformadas em `NOP`s" → não é possível diminuir o nº de instruções de um programa
- ☑ 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; ...`

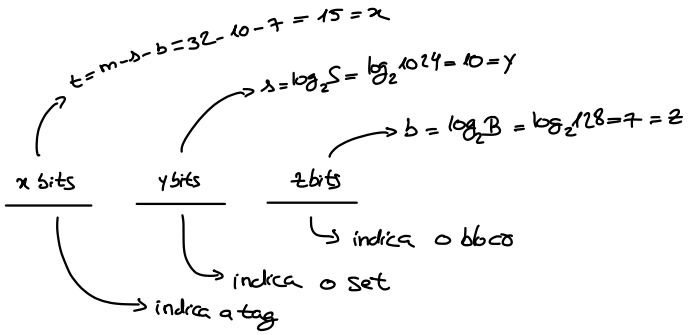
- ☒ permite explorar a hierarquia de memória pois exibe localidade espacial nos acessos a `i`." → i é um registo, por isso nem acede a memória
- ☑ permite explorar a hierarquia de memória pois exibe localidade espacial nos acessos a `a[]`."
- ☒ permite explorar a hierarquia de memória pois exibe localidade temporal nos acessos a `a[]`."
- ☒ permite explorar a hierarquia de memória pois exibe 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)?

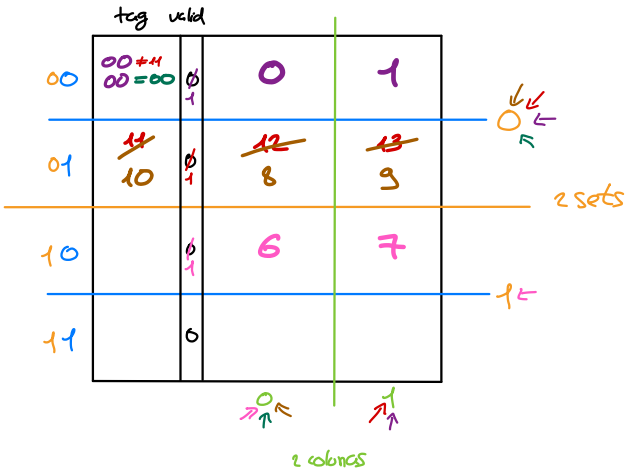
↳ quantidade de bits para o endereço

- ☑ *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	S=0 // L=0	00	cold miss
13	S=0 // L=1	11	cold miss
0	S=0 // L=0	00	hit
6	S=0 // L=1	01	cold miss
8	S=0 // L=1	10	colisão



TAG SET COLUNA

1 = 0001 → cold miss (não existe a tag) → ir buscar à memória

13 = 1101 → " " " " " " → " " " "

0 = 0000 → hit

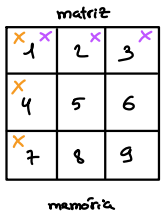
6 = 0110 → cold miss (não existe a tag) → ir buscar à memória

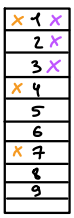
8 = 1000 → colisão (não está na cache e temos de tirar uma linha para pôr a nova, neste caso tiramos a que não é acedida a mais tempo, LRU)

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.





1º col=0 lin=0

2º col=0 lin=1 → não tem localidade espacial

3º col=0 lin=2

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

1º col=0 lin=0

2º col=1 lin=0 → tem localidade espacial

3º col=1 lin=1