

Nota: cada resposta errada nas questões 1 a 6 desconta 0.33 valores

1. [1,0 valores] - Considere o seguinte excerto de um programa escrito em *assembly* e a executar numa máquina com cache:

```
loop:  movl 0(%ebx), %edx
        mull $2, %edx
        addl %edx, %eax
        addl &4, %ebx
        subl $2, %ecx
        jnz ciclo
```

Considere que o registo `%ecx` tem inicialmente o valor 10. O programa é executado numa máquina com frequência do relógio igual a 3 GHz, $CPI_{CPU} = 1$, a *miss rate* de instruções é 1% e a de dados é 6%. Sabendo que *miss penalty* é de 150 ns, qual o tempo de execução deste programa?

Handwritten calculations and options:

$\#I = 30$
 $\%Miss = \frac{5}{30} = \frac{1}{6}$

$T_{exec} = \frac{CPI \times 30}{3 \times 10^9} = \frac{10 \times 30}{3 \times 10^9} = 100 \text{ ns}$

$CPI = CPI_{CPU} + CPI_{MEM} = 1 + 9 = 10$

$(0.01 + 0.06 \times \frac{1}{6}) \times 450 = 9$

Options:

- ☐ $T_{exec} = 150 \text{ ns}$
- ☐ $T_{exec} = 33 \text{ ns}$
- ☐ $T_{exec} = 50 \text{ ns}$
- ☒ $T_{exec} = 100 \text{ ns}$

2. [1,0 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. [1,0 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 `a[]` e a `b[]`.”

seguido, logo é espacial?

$a[0]$
 $a[1]$
 $a[2]$

$b[100]$
 $b[200]$
 $b[300]$

Nome: _____

Número: _____

4. [1,0 valores] – O código

```
for (i=1 ; i<S ; i+=1) a[i] = (a[i-1] / (i>10 ? 2.:3.));
```

não vectoriza devido:

- ☒ a uma dependência de dados RAW entre iterações.
- ☐ aos dados processados não se encontrarem em posições consecutivas de memória.
- ☐ à possibilidade de aliasing entre a[i] e a[i-1].
- ☐ à estrutura condicional incluída dentro do ciclo for.

5. [1,0 valores] – Considere um processador com um bloco de lógica combinatória que pode ser dividido em 4 blocos, cada com uma duração de 214, 283, 252 e 201 picosegundos. Com uma organização em pipeline de 4 estágios este processador permite um ciclo de relógio mínimo de 333 picosegundos. A frequência máxima da organização sequencial correspondente a esta lógica combinatória é de: $T_{cc_{seq}} = \max(214, 283, 252, 201) + T_{reg} = 333 \text{ ns}$ $333 - 283 = T_{reg} = 50 \text{ ps}$

$T_{cc_{seq}} = 214 + 283 + 252 + 201 + T_{reg} = 1000 \text{ ps} = 1 \text{ ns}$ $f = \frac{1}{T_{cc}} = \frac{1}{1 \text{ ns}} = 1 \text{ GHz}$

<input type="checkbox"/>	$f = 3.0 \text{ GHz}$	<input checked="" type="checkbox"/>	$f = 1.0 \text{ GHz}$	<input type="checkbox"/>	$f = 2 \text{ GHz}$	<input type="checkbox"/>	$f = 1.5 \text{ GHz}$
--------------------------	-----------------------	-------------------------------------	-----------------------	--------------------------	---------------------	--------------------------	-----------------------

6. [1,0 valores] – O *loop unrolling* tem potencial para disponibilizar mais instruções para execução em paralelo num contexto de superescalaridade. Para o código abaixo seleccione a opção de *unrolling* que disponibiliza potencialmente mais *instruction level parallelism*.

```
int a[SIZE], i, sum=0;
for (i=0; i < SIZE ; i++) sum +=a[i];
```

4º

☐

```
int a[SIZE], i, sum=0;
for (i=0; i < SIZE ; i+=2) {
    sum +=a[i];
    sum +=a[i+1];
}
```

1º

☒

```
int a[SIZE], i, sum=0, sum_a=0;
for (i=0; i < SIZE ; i+=2) {
    sum_a +=a[i];
    sum +=a[i+1];
}
sum += sum_a;
```

3º

☐

```
int a[SIZE], i, sum=0;
for (i=0; i < SIZE ; i+=4) {
    sum +=a[i];
    sum +=a[i+1];
    sum +=a[i+2];
    sum +=a[i+3];
}
```

2º

☐

```
int a[SIZE], i, sum=0;
for (i=0; i < SIZE ; i+=4) {
    sum +=a[i] + a[i+1];
    sum +=a[i+2] + a[i+3];
}
```

7. [2.0 valores] Considere uma máquina com as seguintes características:

- registos com 4 bytes e todas as instruções com 4 bytes de tamanho;
- cache com 4 linhas, cada linha com 8 bytes; transferências de memória para a cache (e vice-versa) começam sempre num endereço múltiplo de 8 e o bloco transferido tem 8 bytes de tamanho.

Considere o código apresentado no exercício 8, sabendo que a instrução I1 está em memória a partir do endereço 804. Considere a cache inicialmente fria. Indique, justificando qual a *miss rate* total (instruções e dados) para a execução desse código (considere todas as iterações do ciclo)

Memória

	???
804	I1
808	I2
812	I3
816	I4
820	I5

Contador

#I	HHH
4 Mem	1
Ac. I	Ac. D
0	0

$h_n = \frac{2}{6} = \frac{1}{3}$

$m_n = 1 - h_n = 1 - \frac{1}{3} = \frac{2}{3}$

Cache

Múltiplo de 8 →

800	???	I1
808	I2	I3
816	(1.5x)	???
824	I4	I5

8. [2.0 valores] Considere o programa abaixo e considere uma máquina com execução especulativa (saltos previstos como tomados) *out-of-order* e as unidades funcionais:

- . MEM – acessos à memória (2 ciclos);
- . INT + J – operações sobre inteiros e saltos (1 ciclo).

I1: mov \$2, %ecx
I2: mov (%ebx), %edx
I3: add %edx, %eax
I4: dec %ecx
I5: jz I2

Preencha a tabela abaixo, indicando em cada ciclo do relógio qual a instrução em execução em cada unidade. Considere todas as iterações do ciclo.

	1	2	3	4	5	6	7	8	9	10
MEM	I2	I2		I2	I2					
INT + J	I1	I4	I3	I5	I4	I3	I5			

1ª iteração

2ª iteração

Nome: _____

Número: _____

