1	
2	
3	
4	
5	
6	
T	

1. Considere o seguinte programa escrito em *assembly* do Y86:

```
        movl $10,%ecx
ciclo:  movl 0(%eax), %edx
        addl %edx, %edx
        subl 1, %ecx
        jnz ciclo
```

Classe	CPI
Aritméticas/movimento de dados	1
Acessos à memória	2
Controlo de fluxo	3

Sabendo que o CPI de cada uma das classes de instruções é o indicado na tabela, calcule o CPI global na execução deste programa.

2. Explique qual a diferença entre as métricas ciclos por instrução (CPI) e ciclos por elemento (CPE) e explique a utilidade de cada métrica para avaliar o desempenho de um programa numa dada arquitetura.

Nome: _____ Número: _____

3. Um factor importante no desempenho da hierarquia de memória é o impacto de uma *miss* na *cache*, designado por *miss penalty*. Indique as várias formas que conhece para diminuir a grandeza da *miss penalty* no desenho de uma arquitetura.

4. Numa nova geração de uma dada arquitetura foi possível duplicar a frequência de relógio. Calcule o ganho obtido nessa nova arquitetura, relativamente à arquitetura base, para um programa, em que CPI*cpu* é 1, o *miss rate* de instruções é 3%, o *miss rate* de dados é 5% e com uma percentagem de acessos à memória de 40%. Considere também que a *miss penalty* na arquitetura original era de 100 ciclos.

5. Considere o seguinte programa em C.

```
void func(int h[], int I[], int W, int H) {
  for (y=0 ; y<H ; y++) {
    for (x=1 ; x<(W-1) ; x++) {
          kernel(h, I, y*W+x);
    }
  }
}
```

Considere duas opções para a função *kernel*:

a)
```
void kernel(int res[], int inp[], int ndx) {
  res[ndx] = (inp[ndx-1] + inp[ndx])/2;
}
```

b)
```
void kernel(int res[], int inp[], int ndx) {
  res[ndx] = (inp[ndx-1] + inp[ndx] + inp[ndx+1])/3;
}
```

Indique justificando se este programa apresenta localidade espacial e/ou localidade temporal, e qual das duas opções para a função *kernel* pensa ser mais "amigável" da hierarquia de memória.

6. Considere o seguinte programa em *assembly* do Y86. Apresente o mesmo programa em código máquina, indicando a sua disposição na memória. Inclua o endereço da posição de memória onde se inicia o armazenamento de cada instrução.

```
            .pos 0x00
            jmp main
t:          .long 0x10          # 4 bytes de dados
main:       irmovl t, %ecx
            mrmovl 0(%ecx), %eax
            addl %eax, %ecx
            rmmovl %ecx, 0(%ebx)
            halt
```