não usar	
1	
2	
3	
4	
5	
T	

1. Uma dada máquina tem uma frequência do relógio de 2GHz e a respectiva *cache* apresenta uma *miss rate* de instruções de 2% e de dados de 4%. A *miss penalty* é de 20 nanosegundos. O CPI do CPU é dado pela tabela abaixo para diferentes tipos de instruções:

Tipo instrução	CPI_{CPU}	Tipo instrução	CPI_{CPU}
Acesso memória	4	Restantes	2

Para um valor de %ecx = 1000 um dos excertos do programa abaixo executa em 7.4 microsegundos. Indique, justificando, qual.

```
excerto1:
  addl %eax, (%ebx)
  addl $4, %ebx
  decl %ecx
  jnz excerto1
```

```
excerto2:
      addl %eax, %ebx
      addl $4, %ebx
      subl $2, %ecx
      jnz excerto2
```

2. A lógica combinatória de um dado processador tem uma latência de 400 ps e pode ser dividida em sub-blocos com latências arbitrárias (a soma das latências de todos os sub-blocos combinatórios tem que ser 400 ps). Um bloco de registos tem uma latência de 20 ps. Calcule, para este processador, a máxima frequência do relógio possível para a organização sequencial e para organizações em pipeline com 2 e 10 estágios.

Nome: _____ Número: _____

3. A tabela abaixo apresenta o estado de uma cache com um total de 4 linhas e para 2 organizações: *direct mapping* (S=4, E=1, B=4, m=5) e *2-way set associative* (S=2, E=2, B=4, m=5). As colunas **S** e **L** identificam o *set* e a linha para cada uma das organizações. A coluna ***tag*** apresenta o valor deste campo para cada uma das organizações. O ***valid*** bit indica se essa linha da *cache* é válida. As 4 colunas mais à direita da seguinte tabela apresentam, em hexadecimal, o valor de cada um dos *bytes* carregados na respectiva linha da *cache*.

direct map		2-way set associative				Bytes			
L	tag	S	L	tag	valid	00	01	10	11
00	1	0	0	10	1	0x23	0x7B	0xFF	0x00
01	--	0	1	--	0	--	--	--	--
10	--	1	0	--	0	--	--	--	--
11	1	1	1	11	1	0x12	0x05	0x8F	0xD0

Para a sequência de endereços indicados abaixo (em base 10), indique, para cada uma das organizações, se se trata de um *hit*, *cold miss* ou colisão. **Indique o *set* e a linha** (dentro do *set*) em que o endereço mapeia. **No caso de um *cold miss* ou colisão indique o valor da *tag* após o acesso.** Considere um algoritmo de substituição LRU para a organização *2-way set associative*.

addr=07 (direct map)

addr=07 (*2-way*)

addr=14 (direct map)

addr=14 (*2-way*)

4. Para cada um dos ciclos abaixo indique, justificando, se é vectorizável. Se identificar dependências de dados entre iterações calcule a respectiva distância e indique o seu tipo: *Write After Read* (WAR) ou *Read After Write* (RAW).

```
float a[SIZE], b[SIZE];

for (i=1 ; i <SIZE-1 ; i++)
  a[i] = a[i+1] * b[i-1];
```

```
float a[SIZE], b[SIZE];

for (i=1 ; i < SIZE-2 ; i++)
  a[i] = a[i-1] / b[i+2];
```

Nome: _____ Número: _____

5. Para o código abaixo proponha uma implementação que explore *Thread Level Parallelism* usando o OpenMP. Justifique as suas opções; em particular justifique cada uma das directivas e/ou cláusulas OpenMP utilizadas.

```
#define S 1000000
float a[S];
int i ,j;
for (i=0 ; i < S ; i++) {
  a[i] = 0.;
  for (j=0 ; j < i ; j++) {
    a[i] += (float)j;
} }
```