Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
Curso: Bacharelado em Ciência da Computação
Disciplina: Arquitetura e Organização de Computadores
Prof. Dr. João Fabrício Filho
Monitor: Rafael Augusto de Souza

Os exercícios foram adaptados do livro
Patterson, David A. Hennessy, John L. Organização e Projeto de Computadores. Disponível em: Minha Biblioteca, (5a. edição). Grupo GEN, 2017.

1. <§2.7> De acordo com o código em C abaixo e considerando os pares (registradores, variáveis) como correspondentes: ($t0,i) / ($t1,j) / ($s0,n) / ($s1,k), / ($s2,&D) / ($s3,&E), responda as questões.

```
C/C++
int k = n-1;
for(int i = 0; i < n; i++){
  D[i] = E[k];
  k--;
}
```

   a. Construa o código *assembly* correspondente ao algoritmo acima.

   b. Considere inicialmente $s0 = 7, Mem[&D]={1,7,2,5,1,2.9}, Mem[&E]={1,2,3,4,5,6,7}. Calcule quantas instruções serão executadas até a finalização do algoritmo.

   c. Considerando os valores iniciais da questão anterior, calcule quantos ciclos de clock são necessários para que o algoritmo execute, dado que o CPI de manipulação de memória é 2 e de outras operações é 1.

2. <§2.7> Considere o código de alto nível abaixo para responder à esta questão:

```
C/C++
int soma = 0;
for(i = 0; i < n; i++){
  for(j = 0; j < m; j++){
     soma += D[i] + D[j];
     D[j] = soma;
  }
}
```

O código *assembly* correspondente está apresentado abaixo, considerando os seguintes pares (registradores,variáveis): ($t0,i) / ($t1,j) / ($s0, n) / ($s1,m) / ($s2,soma) / ($s3,&D).

```
Unset
add $s2, $0, $0
add $t0, $0, $0
FOR1:
slt $t2, $t0, $s0
beq $t2, $0, EXIT1
add $t1, $0, $0
FOR2:
slt $t2, $t1, $s1
beq $t2, $0, EXIT2
sll $t3, $t1, 2
sll $t4, $t0, 2
add $t4, $t4, $s3
add $t3, $t3, $s3
lw $t5 , 0($t4)
lw $t6 , 0($t3)
add $s2, $s2, $t5
add $s2, $s2, $t6
sw $s2, 0($t3)
addi $t1, $t1, 1
j FOR2
EXIT2:
addi $t0, $t0, 1
j FOR1
EXIT1:
```

a. Considerando os valores iniciais de $s0 = 1 / $s1 = 7 / Mem[&D] = [7,2,4,1,8,3,9] calcule quantas instruções serão executadas e como ficará o vetor D após a finalização dessa execução.

b. Sabe-se que o CPI de manipulação de memória é 5, de controle é 3 e de outras operações é 1. Calcule quantos ciclos de *clock* serão necessários para executar o algoritmo, considerando os valores iniciais da questão anterior.

3. Considere o código de alto nível abaixo:

```
C/C++
int max = 0;
for(int i =0; i< n; i++){
  if(V[i] > max){
    max = V[i];
  }
}
for(int i=0; i<n; i++){
  C[V[i]]++;
}
```

```
int k = 0;
for(int i = 0; i < max+1; i++){
  for(int j = 0; j < C[i]; j++){
    V[k] = i;
    k++;
  }
}
```

a. Escreva o código *assembly* do algoritmo acima.

b. Considerando os pares ($s0,n) / ($s1, max) / ($s2,k) / ($s3,&V) / ($s4,&C) / ($t0,i) / ($t1,j) e também os valores iniciais de $s0 = 8, Mem[&V] = {2,3,6,7,1,0,4,2} e Mem[&C] = {0,0,0,0,0,0,0,0} calcule quantos ciclos de *clock* serão necessários para executar o algoritmo. Considere CPI de manipulação de memória = 2 e CPI de outras operações = 1.

4. <§2.7 > Considere o código de alto nível abaixo para responder às próximas questões.

```
C/C++
for(int i = 0; i < n-1;i++){
  for(int j = 0; j<n-i-1;j++){
    if(V[j]>V[j+1]){
      int x = V[j+1];
      V[j+1] = V[j];
      V[j] = x;
    }
  }
}
```

a. Construa o código *assembly* equivalente ao código.

b. Considere os pares ($s0,x) / ($s1,n) / ($s2, &V) / ($t0,i) / ($t1,j), os valores iniciais de $s1 = 10 e Mem[&V] = [9,8,7,6,5,4,3,2,1,0] e também o CPI de 5 para manipulação de memória, de 3 para controle e de 2 para outras operações. Calcule quantas instruções serão executadas e quantos ciclos de *clock* serão necessários.

5. Dado o código em C abaixo e os pares correspondentes de registradores e variáveis, escreva o código em MIPS Assembly. Considere que os valores iniciais de n e m estão nos registradores $s0 e $s1, e &A e &B nos registradores $s2 e $s3.

```C/C++
int soma = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
        if (A[i] != B[j]) {
            soma += A[i];
        }
    }
}
```

a. Construa o código *assembly* equivalente ao código.

b. Considere os valores de $s0 = 5, $s1 = 3, Mem[&A] = {1, 2, 3, 4, 5} e Mem[&B] = {3, 6, 2}. Calcule quantas instruções são executadas e o valor final da variável soma após a execução.

6. Dado o código em C abaixo, escreva o código correspondente em MIPS Assembly. Suponha que &V seja armazenado em $s2, max e min estejam em $s3 e $s4, e o tamanho do vetor V em $s0. Inicialmente, max = -32768 e min = 32767.

```C/C++
int max = -32768;
int min = 32767;
for (int i = 0; i < n; i++) {
    if (V[i] > max) max = V[i];
    if (V[i] < min) min = V[i];
}
```

a. Construa o código *assembly* equivalente ao código.

b. Com Mem[&V] = {5, -10, 13, 7, 0, 20, -5, 3}, calcule o número total de instruções executadas.

c. Suponha CPI de manipulação de memória = 3, e outras operações = 1. Quantos ciclos de clock são necessários para executar o trecho de código?

7. Implemente o código MIPS para o cálculo do produto escalar de dois vetores, considerando que os endereços base de A e B estão em $s2 e $s3, e o tamanho do vetor n está em $s0. Armazene o resultado no registrador $s4.

```C/C++
int resultado = 0;
for (int i = 0; i < n; i++) {
    resultado += A[i] * B[i];
}
```

a. Construa o código *assembly* equivalente ao código.

b. Calcule o número de instruções e o total de ciclos de clock, supondo CPI de 2 para manipulação de memória e 4 para multiplicação.