Lista de Exercícios

1ª Questão) Traduza o seguinte loop para C. Assuma que o inteiro i é armazenado no registrador $t1, $s2 armazena o inteiro chamado result, e $s0 armazena o endereço base do inteiro MemArray

```
addi $t1, $0, $0
LOOP: lw $s1, 0($s0)
add $s2, $s2, $s1
addi $s0, $s0, 4
addi $t1, $t1, 1
slti $t2, $t1, 100
bne $t2, $s0, LOOP
```

2ª Questão) Considere o seguinte loop em MIPS

```
LOOP: slt $t2, $0, $t1
beq $t2, $0, DONE
subi $t1, $t1, 1
addi $s2, $s2, 2
j LOOP
DONE:
```

a)Assuma que o registrador $t1 é inicializado com o valor 0. Qual é o valor no registrador $S2 assumindo que $S2 inicialmente possui valor zero.

b)Para o código escrito em assembly MIPS, escreva a rotina equivalente em C. Assuma que os registradores $s1, $s2, $t1 e $t2 são inteiros A, B, i e temp, respectivamente.

3ª Questão) Traduza o seguinte código em C para assembly MIPS. Use o número mínimo de instruções. Assuma que os valores de a, b, i e j estão armazenados nos registradores $s0, $s1, $t0 e $t1 respectivamente. Também, assuma que registrador $s2 armazene o endereço base da vetor D

```
for(i=0; i<a; i++)
for(j=0; j<b; j++)
D[4*j] = i + j;
```

4ª Questão) Descreva detalhadamente o que cada código faz

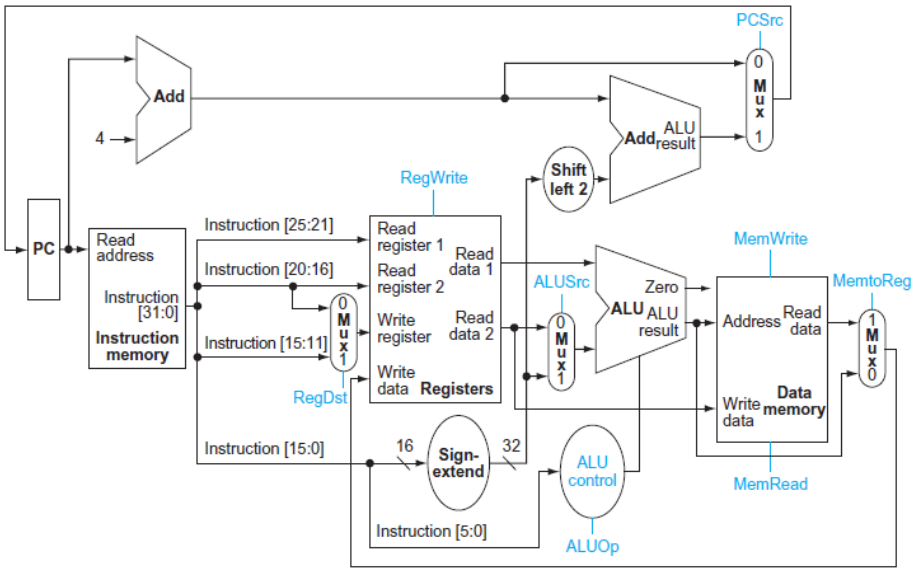
```
.text
main:
li $v0, 5
syscall
move $t0, $v0
li $v0, 5
syscall
move $t1, $v0
bgt $t0, $t1, t0_bigger
move $t2, $t1
b endif
t0_bigger:
move $t2, $t0
endif:
move $a0, $t2
li $v0, 1
syscall
li $v0, 10
syscall
```

```
.text
.globl main
main:
li $a0, 5
jal Funcao
move $s0, $v0
li $v0, 10
syscall
Funcao:
sub $sp,$sp,4
sw $ra, 0($sp)
li $t1, 1
slti $t0, $a0, 2
beq $t0, $zero, Calcula
add $v0, $zero, $zero
beq $a0, $zero, Sai
add $v0, $t1, $zero
Sai:
lw $ra, 0($sp)
add $sp, $sp, 4
```

	jr $ra ***Calcula:*** ***add $a1, $a0, $zero*** ***Loop:*** ***sub $a1, $a1, $t1*** ***jal Multiplica*** ***add $a0, $v0, $zero*** ***bne $a1, $t1, Loop*** ***j Sai*** ***Multiplica:*** ***mult $a0, $a1*** ***mflo $v0*** jr $ra

5ª Questão) Considere a seguinte idéia: modificar a arquitetura do conjunto de instruções e retirar a capacidade de especificar um deslocamento para instruções de acesso à memória. Especificamente, todas as instruções *load-store* com deslocamento diferente de zero devem tornar-se pseudo-instruções e devem se implementadas por meio de duas instruções. Por exemplo:
addi $at, $t1, 104 # some o deslocamento a um registrador temporário
lw $t0, $at # nova forma de fazer lw $t0, 104($t1)
Que mudanças devem ser feitas no caminho de dados multiciclo e no controle para que essa arquitetura simplificada possa funcionar?

6ª Questão) Apresente os valores dos sinais de controle *ALUSrc, ALUOp, MemRead, MemWrite, RegWrite, MemtoReg* e *RegDst,* na implementação da instrução *addi* no MIPS com multiplos ciclos de clock, indicando em que estágio cada um desses sinais são usados.



7ª Questão) Por que a instrução addi não pode ser considerada uma instrução de formato R, com opcode igual 000000?

8ª Questão) Considere o trecho de programa no quadro abaixo e os conteúdos iniciais de registradores e posições de memória relevantes.
addi $t4, $zero, 2
root : add $t1, $t2, $t3
lw $t3, 0x100($t1)

sw $t3, 0x200($t1)
subi $t4, $t4, 2
beq $t4, $t3, root
addi $t3, $t3, 0x100
Conteúdos iniciais da memória e dos registradores relevantes:
$t1=0x100, $t2=0x100, $t3=0x100, $t4=0x100
Mem [0x100-0x103]= 0x002345AB
Mem [0x200-0x203]= 0x0000000A
Mem [0x300-0x303]= 0x00000000
Mem [0x400-0x403]= 0x00CD5F00
Simule a execução completa do programa