Lista 2

Arquitetura e Organização de Computadores

Junho de 2024

1. Traduza o programa abaixo, em C, para MIPS.

```
int add(int a, int b) {
    int sum;

    sum = a + b;
    return sum;
}

int negative(int a) {
    int neg_a;

    neg_a = 0 - a;
    return neg_a;
}

int diff(int a, int b) {
    int neg_b, difference;

    neg_b = negative(b);
    difference = add(a, neg_b);
    return difference;
}

void main() {
    int a, b, c, d;

    c = add(a, b);
    d = diff(a, b);
}
```

2. Traduza o programa abaixo, em C, para MIPS.

```
int factorial(int n) {
        if(n == 0) return 1;
        return n * factorial(n - 1);
}

void main()  {
        int num = 5;
        int fact = factorial(num);
}
```

3. Ao fim da execução do programa abaixo, quais os valores em `$sp` e `$s1`? Qual o menor valor armazenado, ao longo da execução deste programa, em `$sp`?

```
addi $sp, $0, 0x7FFFEFFC
j main

recursive:
lw $t0, 0($sp)
addi $sp, $sp, 4

bne $t0, $0, else
addi $sp, $sp, -4
sw $t0, 0($sp)
jr $ra

else:
addi $t0, $t0, -1
addi $sp, $sp, -8
sw $t0, 0($sp)
sw $ra, 4($sp)
jal recursive
lw $t0, 0($sp)
lw $ra, 4($sp)
addi $sp, $sp, 8
addi $t0, $t0, 1
addi $sp, $sp, -4
sw $t0, 0($sp)
jr $ra

main:
addi $s0, $0, 0xABCD
addi $sp, $sp, -4
sw $s0, 0($sp)
jal recursive
lw $s1, 0($sp)
addi $sp, $sp, 4
```

4. Ao fim da execução do programa abaixo, como está o banco de registradores?

```
addi $2, $0, 0xABCD
addi $3, $0, 0xFEDC
multu $2, $3
mflo $4
mfhi $5
mulu $6, $2, $3
mfhi $7

addi $2, $0, 0xABCDEF00
addi $3, $0, 0x00FECDBA
multu $2, $3
mflo $8
mfhi $9
mulu $10, $2, $3
mfhi $11
```

5. Ao fim da execução do programa abaixo, como está o banco de registradores?

```
addi $2, $0, 0xABCD
addi $3, $0, 0xFEDC
div $2, $3
mflo $4
mfhi $5
divu $2, $3
mflo $6
mfhi $7

addi $2, $0, 0xABCDEF00
addi $3, $0, 0x00FECDBA
div $2, $3
mflo $8
mfhi $9
divu $2, $3
mflo $10
mfhi $11
```

6. Quais valores hexadecimais podem representar o número `3.462408E-11` conforme a norma IEEE 754?

7. Qual número é armazenado como `0x49179EA0` conforme a norma IEEE 754?

8. Traduza o programa abaixo, em C, para MIPS.

```
#include <math.h>

float fop(float a, float b) {
    if(b == 0) return NAN;
    return a*b + a/b;
}

void main() {
    float a = 40.1;
    float b = 00.4;
    float c = fop(a, b);
}
```

Para as questões que pedem a descrição de instruções, considere os processadores apresentados no livro Digital Design and Computer Architecture. Se necessário expandi-lo, indique as alterações feitas.

9. Descreva, temporalmente, como as seguintes instruções são executadas em um processador de ciclo único. Para isso, indique quais sinais são transferidos entre quais componentes.

   Exemplo: ADD

$$\text{PC}_{\text{multiplexer}} \xrightarrow{\text{PC'}} \text{PC}_{\text{reg}} \begin{matrix} \xrightarrow{\text{PC}} \text{InstMem} \\ \xrightarrow{\text{PC}} \text{PC}_{\text{adder}} \xrightarrow{\text{PC+4}} \text{PC}_{\text{multiplexer}} \end{matrix}$$

$$\text{InstMem} \begin{matrix} \xrightarrow{\text{Instr}} \text{CtrlUnit (decodifica a instrução)} \xrightarrow{\text{RegDst}} \text{A3}_{\text{multiplexer}} \\ \xrightarrow{\text{Instr}} \text{RegFile (acessa os registradores)} \\ \xrightarrow{\text{rd}} \text{A3}_{\text{multiplexer}} \xrightarrow{\text{WriteReg}} \text{RegFile} \end{matrix}$$

$$\text{CtrlUnit} \begin{matrix} \xrightarrow{\text{ALUSrc}} \text{SrcB}_{\text{multiplexer}} \\ \xrightarrow{\text{ALUCtrl}} \text{ALU} \end{matrix}, \text{RegFile} \begin{matrix} \xrightarrow{\text{RegData1/SrcA}} \text{ALU} \\ \xrightarrow{\text{RegData2}} \text{SrcB}_{\text{multiplexer}} \xrightarrow{\text{SrcB}} \text{ALU} \end{matrix}$$

$$\text{CtrlUnit} \begin{matrix} \xrightarrow{\text{MemWrite}} \text{DataMem} \\ \xrightarrow{\text{MemtoReg}} \text{Result}_{\text{multiplexer}}, \text{ALU} \xrightarrow{\text{ALUResult}} \text{Result}_{\text{multiplexer}} \xrightarrow{\text{Result}} \text{RegFile} \\ \xrightarrow{\text{RegWrite}} \text{RegFile} \end{matrix}$$

   (a) sub
   (b) jr
   (c) jal
   (d) beq
   (e) sw
   (f) lw
   (g) addi

10. Descreva, temporalmente, como as seguintes instruções são executadas em um processador multi-ciclo. Para isso, indique quais sinais são transferidos entre quais componentes.

    Exemplo: ADD

$$1\ \mathrm{PC_{reg}} \xrightarrow{\mathrm{PC}} \mathrm{Adr_{multiplexer}} \xrightarrow{\mathrm{Adr}} \mathrm{Mem} \xrightarrow{\mathrm{Instr/Data}} \mathrm{InstReg,\ CtrlUnit} \begin{array}{l} \xrightarrow{\mathrm{PC_{write}}} \mathrm{PCEn_{OR}} \xrightarrow{\mathrm{PCEn}} \mathrm{PC_{reg}} \\ \xrightarrow{\mathrm{PCSrc}} \mathrm{PCSrc_{multiplexer}} \\ \xrightarrow{\mathrm{IR_{write}}} \mathrm{InstReg} \\ \xrightarrow{\mathrm{IorD}} \mathrm{Adr_{multiplexer}} \\ \xrightarrow{\mathrm{ALUSrcA}} \mathrm{SrcA_{multiplexer}} \\ \xrightarrow{\mathrm{ALUSrcB}} \mathrm{SrcB_{multiplexer}} \\ \xrightarrow{\mathrm{PCSrc}} \mathrm{PCSrc_{multiplexer}} \end{array},$$

$$\begin{array}{r} \mathrm{PC_{reg}} \xrightarrow{\mathrm{PC}} \mathrm{SrcA_{multiplexer}} \\ 4 \xrightarrow{4} \mathrm{SrcB_{multiplexer}} \end{array} \begin{array}{l} \xrightarrow{\mathrm{SrcA}} \\ \xrightarrow{\mathrm{SrcB}} \end{array} \mathrm{ALU} \xrightarrow{\mathrm{PC+4}} \mathrm{PCSrc_{multiplexer}} \xrightarrow{\mathrm{PC+4}} \mathrm{PC_{reg}}$$

$$2\ \mathrm{InstReg} \begin{array}{l} \xrightarrow{\mathrm{Funct\ \&\ Op}} \mathrm{CtrlUnit} \\ \xrightarrow{\mathrm{rt\ \&\ rs}} \mathrm{RegFile} \xrightarrow{\mathrm{rd1\ \&\ rd2}} \mathrm{RegFileData_{reg}} \end{array}$$

$$3\ \mathrm{RegFileData_{reg}} \begin{array}{l} \xrightarrow{\mathrm{A}} \mathrm{SrcA_{multiplexer}} \\ \xrightarrow{\mathrm{B}} \mathrm{SrcB_{multiplexer}} \end{array} \begin{array}{l} \xrightarrow{\mathrm{SrcA}} \\ \xrightarrow{\mathrm{SrcB}} \end{array} \mathrm{ALU} \xrightarrow{\mathrm{ALURes}} \mathrm{ALUOut_{reg}, CtrlUnit} \begin{array}{l} \xrightarrow{\mathrm{ALUCtrl}} \mathrm{ALU} \\ \xrightarrow{\mathrm{ALUSrcA}} \mathrm{SrcA_{multiplexer}} \\ \xrightarrow{\mathrm{ALUSrcB}} \mathrm{SrcB_{multiplexer}} \end{array}$$

$$4\ \mathrm{ALUOut_{reg}} \xrightarrow{\mathrm{ALUOut}} \mathrm{WD3_{multiplexer}} \xrightarrow{\mathrm{ALUOut}} \mathrm{WD3_{RegFile}, CtrlUnit} \begin{array}{l} \xrightarrow{\mathrm{RegDst}} \mathrm{RegDst_{multiplexer}} \\ \xrightarrow{\mathrm{MemtoReg}} \mathrm{WD3_{multiplexer}} \\ \xrightarrow{\mathrm{RegWrite}} \mathrm{RegFile} \end{array}$$

    (a) sub
    (b) jr
    (c) jal
    (d) beq
    (e) sw
    (f) lw
    (g) addi

    Para as 8 seguintes questões, considere um processador pipeline que implemente todas as tratativas de hazards.

11. Em qual estágio o registrador de `PC` é incrementado? Em qual estágio o novo valor é disponibilizado à memória de instruções? Em quais condições o valor anterior é mantido para a próxima execução?

12. Em qual estágio a tratativa de data hazards para `lw` é feita? Qual condição é verificada nesta tratativa? Quando este tipo de hazard é detectado, como é tratado?

13. Descreva como, no estágio **Execute**, é feita a avaliação e tratativa de hazards.

14. Quais sinais, no estágio **Memory**, contribuem para a detecção ou tratativa de hazards? Como estes são avaliados ou utilizados?

15. Quais sinais, no estágio **WriteBack**, contribuem para a detecção ou tratativa de hazards? Como estes são avaliados ou utilizados?

16. Em qual estágio de uma instrução `beq` os hazards pertinentes a ela são detectados? Como estes são avaliados e tratados?

17. Em um processador pipeline, o que significar, "limpar" ou "resetar" um registrador de pipeline? Como isso impede a continuação da instrução a partir do estágio em que ocorre?

18. No processador pipeline implementado no livro de referência, é possível realizar forwarding do estágio **WriteBack** para o estágio **Memory**? Em qual condição isto é necessário? Qual a justificativa para a decisão tomada na implementação apresentada pelo livro?

    Para as próximas duas questões, considere o programa abaixo.

```
i00: addi $s0, $0, 10
i01: sw   $s0, 0($sp)
i02: addi $s0, $0, 2
i03: addi $s1, $0, 3
i04: add  $s2, $s1, $s0
i05: sub  $s1, $s2, $s0
i06: lw   $s3, 0($sp)
i07: add  $s4, $s3, $s3
i08: beq  $s4, $s3, one
i09: add  $s0, $0, $0
i10: add  $s1, $0, $0
i11: add  $s2, $0, $0
i12: add  $s3, $0, $0
i13: beq  $0, $0, zero
one:
i14: add  $s0, $0, 1
i15: add  $s1, $0, 1
i16: add  $s2, $0, 1
i17: add  $s3, $0, 1
zero:
i18: add  $s6, $s2, $s3
i19: add  $s5, $s0, $s1
i20: add  $s7, $s6, $s5
```

19. Considerando uma arquitetura que detecte hazards, mas não empregue forwarding, indique, na tabela abaixo, em qual estágio cada instrução está em cada ciclo de clock. Utilize as células da coluna reservada para indicar a instrução executada. Reserve, conforme necessário, linhas para indicar os ciclos de clock.

inst\ciclo	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
i0:																	
i1:																	

20. Considerando uma arquitetura que detecte hazards e empregue forwarding, stall e previsão de desvio (assume que não é tomado) indique, na tabela abaixo, em qual estágio cada instrução está em cada ciclo de clock. Neste exercício, não se restrinja ao processador implementado no livro. Indique os recursos que achar pertinentes. Utilize as células da coluna reservada para indicar a instrução executada. Reserve, conforme necessário, linhas para indicar os ciclos de clock.

inst\ciclo	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
i0:																	
i1:																	

21. Reorganize o programa a seguir a fim de evitar hazards sem alterar seu estado final.

```
or   $t2, $t1, $t0
nor  $t9, $t8, $t7
addi $s2, $s1, 1
add  $s2, $s1, $s0
sub  $s4, $s3, $s2
xor  $t2, $t1, $t0
sw   $t2, 0($sp)
lw   $t9, 4($sp)
subi $t0, $t0, 1
and  $t7, $t8, $t9
```