



Nome: _____ Matrícula: _____

Prova 2

- 1) (6.0) No MIPS simplificado desenvolvido durante o curso várias instruções bastante úteis estão ausentes. Mantendo a compatibilidade do código em linguagem de máquina ao MIPS original, redesenhe o caminho de dados multiciclo, redefina os sinais de controle (se necessário), e implemente a unidade de controle através de um microprograma, de modo a implementar (além das instruções já existentes) todas as instruções abaixo:

- a) (2.0) jr \$r\$g \$
- b) (2.0) addi, subi, ori, andi, slti \$rt,\$rs,Imm # Imediato de 16 bits
- c) (2.0) nor \$rd,\$rs,\$rt

- 2) (4.0) Considerando apenas os seguintes tempos de atraso dos blocos operativos de uma CPU MIPS:

Operação Lógica da ULA (and,or): 100ps
Operação Aritmética da ULA (add,sub): 120ps
Leitura do Banco de Registradores: 50ps
Escrita no Banco de Registradores: 70ps
Leitura da memória: 150ps
Escrita na memória: 200ps

Dado o seguinte trecho de programa:

```
...  
    or $t0,$zero,$zero  
    lw $t1, 100($fp)      # $t1=1  
    lw $t2, 104($fp)      # $t2=100  
    lw $t4, 108($fp)      # $t4=4  
LABEL:  
    lw $t3,0($s0)          } Hazards dados: SÓ FORWARD MEM>MEM É SUFICIENTE  
    sw $t3,0($s1)  
    add $t0,$t0,$t1  
    add $s0,$s0,$t4  
    add $s1,$s1,$t4  
    slt $t5,$t0,$t2          } Hazards dados: FORWARD  
    and $t5,$t1,$t5          } Hazards dados: FORWARD  
    beq $t5,$zero, OUT        } Hazards dados: FORWARD  
    j LABEL  
OUT:  
    add $t0,$zero,$zero          } Hazards control: 3 Bolhas Bög  
    ...  
    ...
```

Hazards dados: SÓ FORWARD MEM>MEM É SUFICIENTE

Hazards dados: FORWARD

Hazards dados: FORWARD

Hazards control: 3 Bolhas Bög

Hazards control: 1 bolha

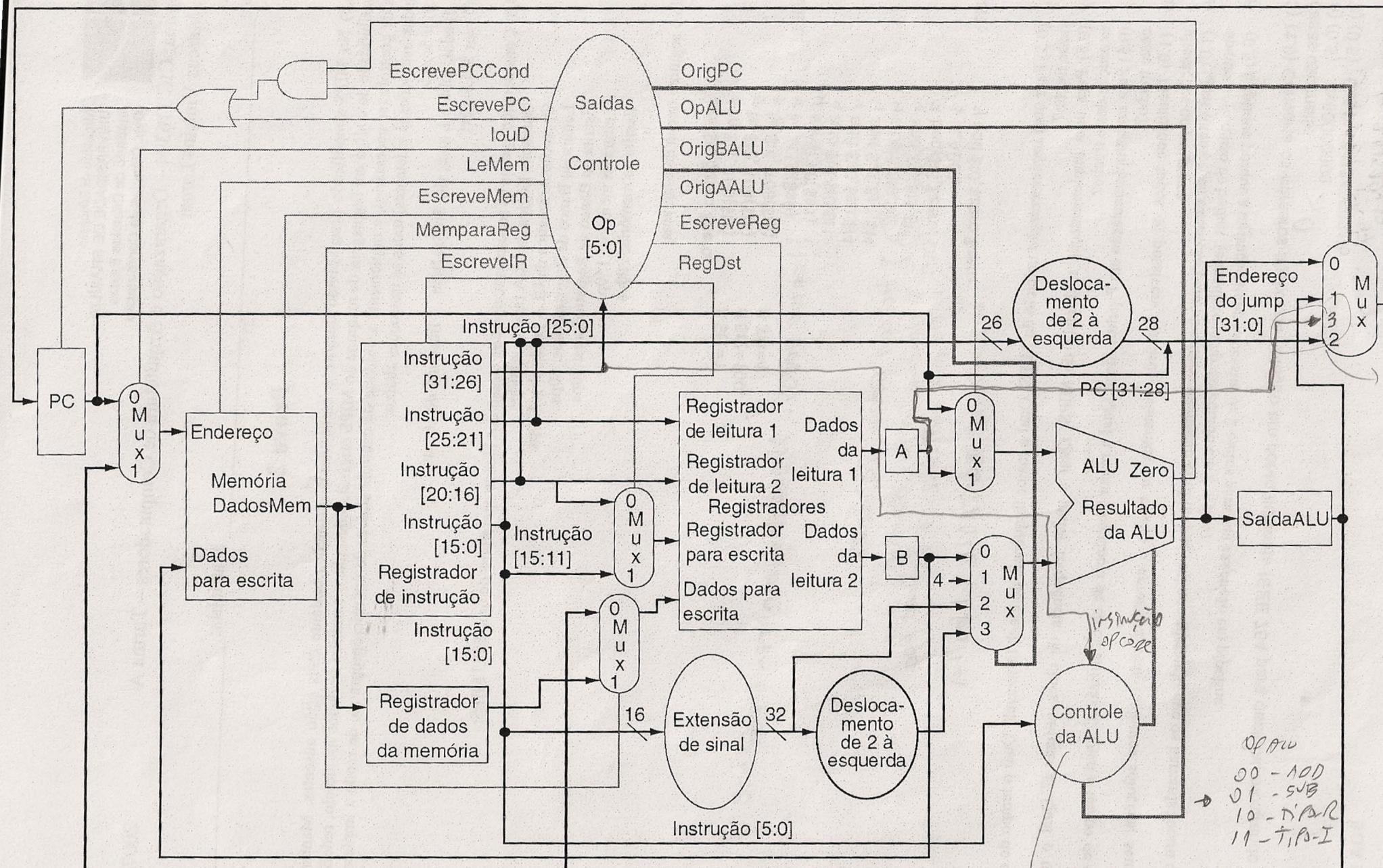
- a) (0.5) Para uma implementação uniciclo do MIPS: Qual a maior frequência de clock utilizável? Qual o tempo de execução deste trecho?
- b) (0.5) Para uma implementação multiciclo do MIPS: Qual a maior frequência de clock utilizável? Qual o tempo de execução deste trecho?
- c) (0.5) Para uma implementação em Pipeline ideal: Qual a maior frequência de clock utilizável? Qual o tempo de execução deste trecho?
- d) (1.0) Identifique todos os possíveis hazards existentes no trecho e apresente suas melhores soluções, sem realizar alterações no programa.
- e) (1.0) Qual o tempo de execução para uma implementação em Pipeline real considerando que os hazards foram corrigidos apenas com o uso de bolhas. (considere jump executado em 2 ciclos)
- f) (0.5) Explique porque a instrução jump necessita de 2 ciclos para sua execução em pipeline.

- 3) (1.0) Converta os números abaixo da notação em ponto flutuante IEEE 754 para decimal, mostrando todas as casas decimais:

- a) (0.5) 0x80000000
- b) (0.5) 0x54442d18400921fb

↳ 8,61980910⁹⁷

BOA SORTE!!!



2º Prova Gabarito

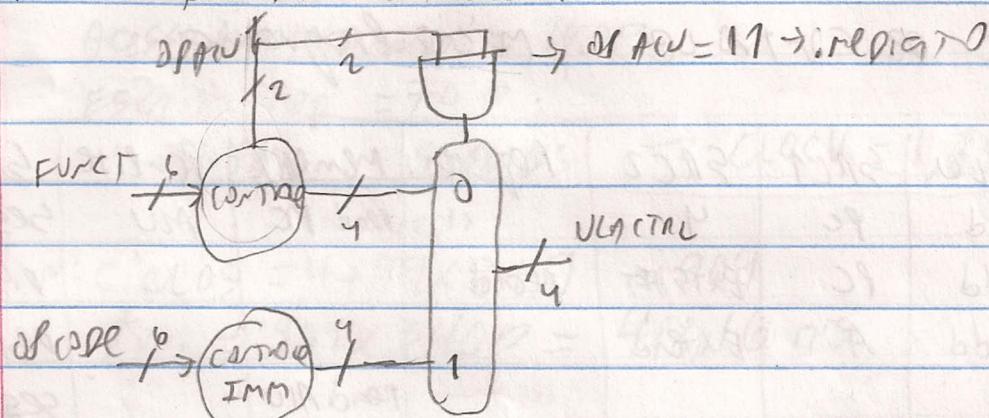
1) caminho de dados: no verso da folha de questões necessárias

a) jrs → armazena valor de \$rs no PC

b) addi, subi, andi, ori, seti → JLR operan c/ imediatos 16 bits

c) MUL → ULA já está programada, precisa apenas o controle da ULA

codificação no controle da ULA



⇒ não precisa programar os circuitos combinacionais de controle
além de definir os

OPACU	FUNCT	MULCTRL	
OPA1	OPA2	F5 F4 F3 F2 F1 F0	OP3 OP2 OP1 OP2
0	0	X X X X X X	0 0 1 0 add
0	1	X X X X X X	0 1 1 0 sub
1	0	1 0 0 0 0 0	0 0 1 0 add
1	0	1 0 0 0 1 0	0 1 1 0 sub
1	0	1 0 0 1 0 0	0 0 0 0 and
1	0	1 0 0 1 0 1	0 0 0 1 or
1	0	1 0 1 0 1 0	0 1 1 1 set
→ 1	0	1 0 0 1 1 1	1 1 0 0 nor

OPACU=11

	OPCODE	ALUCYCLE	opcodes dfa
addi	001000	0010	add
andi	001100	0000	and
ori	001101	0001	or
SLTi	001010	0111	slt
subi	001110	0110	sub

↳ → # usada un opcode live!

OPCODE FUNCT
fr: 000000 001000

* Unidad de control : Mi第一个 programma

LABEL	ALU	SRC1	SRC2	REGISTER	MEMORY	PCWRITE	SEQ
Fetch	add	PC	4		read PC	ALU	seq
	add	PC	EXTSHFT	read			DISPATCH 1
Mem1	Add	A	Extend				DISPATCH 2
LW2					write MDR		seq
					write ALU		Fetch
SW2					write ALU		Fetch
RFOM 1	FUNCT	A	B				seq
					write ALU		Fetch
BEG1	SUBT	A	B			ALU OUT	Fetch
JMP1						JUMP ADDRESS	Fetch
JR1						Reg address	Fetch
Imm1	(Imm)	A	Extend		write IMM		seq
							Fetch

adicionar no PCWRITE

reg ADDRESS = escrivendo registrador A no PC

orig PC = 11

adicionar na ALU control

Imm = ofera regA com IMMEXTENDE SHOR

orig A ALU = 1 orig B ALU = 10 OPALU = 11

A) ciclos no campo memory

WRITE IMM = escreve no banco de reg em RT a 59,09 ns de ds
Reg DST = 0 mem para Reg = 0

Adiciona 1 de desfase 1 para carregar os
cabeçalhos JRI e Imm 1

2) MIPS UNICICLO 1lw

Leitura instrução = 150 p

Leitura reg = 50 p

ALU máximo = 120 p

Acesso memória leitura = 150 p

Escrita reg = 70 p

TOM₂ : 540 ps → CLOCK = 1,8518 GHz

(2.1)

nº ciclos = 4 + 99 × 9 + 9 = 904

t_{multip} = 904 × 540 p = 488,16 ns

b) MIPS multiciclo

Tclock = 200 ps → escrita memória → CLOCK = 5 GHz

nº ciclos = 19 + 99 × 35 + 36 = 3.520

t_{multip} = 3.520 × 200 p = 704 ns

c) MIPS pipeline ideal

Tclock = 200 ps → escrita memória → CLOCK = 5 GHz

nº ciclos = 4 + 99 × 9 + 9 = 904 s/carrega

= 904 + 4 = 908 c/carrega

t_{pipeline} = 908 × 200 ps = 180,8 ns ou 181,6 ns

d) na forma de questões

e) OR \$10, \$20, \$20

LW \$t1, 100(\$fp)

LW \$t2, 104(\$fp)

LW \$t4, 108(\$fp)

LW \$t3, 0(\$sp)

B01+G

B02+G

SW \$t3, 8(\$s1)

add \$t0, \$t0, \$t1

add \$s0, \$s0, \$t4

add \$s1, \$s1, \$t4

Set 215, \$10, 215
BOLTA
BOLTA
and \$15, \$11, \$15
BOLTA
BOLTA
BEG \$15, \$ZERO, \$0
BOLTA
BOLTA
BOLTA
LABEL
BOLTA
OUT: add \$18, \$zero, \$zero

$$\text{Molar mass of } \text{C}_1\text{Cl}_1\text{O}_5 = 1.903 \text{ g/L at } 25^\circ\text{C}$$

$$t_{\text{aperen}} = 1903 \times 200 \mu = 380,6 \text{ ns} \text{ ou } 381,4 \text{ ns}$$

$\hookrightarrow \text{Exphit} = 0$ $\hookrightarrow \text{fracat} = 0$

Lego racer Ø zero

b) 0101010001000100000110100011000101000000
0000100100100011111011

$$= (-1)^0 (1 + 0.261029454861) \cdot 2$$

$$= 8,619,199,51282 \times 10^{97}$$

$$(1348 - 1023) = 325$$

2) f) Porque identifica-se que trata-se de uma instrução jump (Decompilação da instrução) no 2º ciclo logo sozinho nesta etapa o valor de PC pode ser definido corretamente, necessitando-se especificar a instrução seguinte que começará a ser executada na 2ª etapa.
 $(PC+4)$