



Nome: GABARITO Matrícula:       /      

d0 d1 / d2 d3 d4 d5 d6 d7 d8

Matrícula:       /      

## Prova 2

1)(3.0) Em um sistema computacional o tratamento de interrupções e exceções é uma das tarefas mais complicadas de ser corretamente implementada. No MIPS, esta tarefa é realizada pelo coprocessador C0. O coprocessador C0 possui um banco de 32 registradores de 32 bits que são usados para configuração do processador principal, gerenciamento da Memória Cache, e reconhecimento e tratamento das interrupções e exceções.

Modifique adequadamente os Caminhos de Dados e os Blocos Controladores, fornecidos nas folhas em anexo, dos MIPS Uniciclo, Multiciclo E Pipeline, de forma a implementar a detecção e tratamento das seguintes exceções:

- |                                      |   |
|--------------------------------------|---|
| a) (0,25) exceção overflow           | # EPC=PC+4; CAUSE=0x30; PC=0x80000180                             |
| b) (0,25) exceção instrução inválida | # EPC=PC+4; CAUSE=0x28; PC=0x80000180                             |
| c) (0,5) instrução syscall           | # EPC=PC+4; CAUSE=0x20; PC=0x80000180      Opcode/Funct=0x00/0x0C |

Definição de projeto: O resultado errado de overflow é escrito no banco de registradores.

2) (4.0) Escreva uma rotina em Assembly que trate as exceções imprimindo na tela a mensagem de erro: (“CAUTION: exception %d detected at address %d\n”, CAUSE, EPC), e retorne a executar o programa do usuário a partir da próxima instrução. Considere que a rotina de tratamento da instrução syscall está a partir do endereço 0x80001000.

Dica1: No Modo Kernel vc dispõe das instruções reservadas:

mfc0 \$rt, \$rcop1	# \$rt=\$rcop1      Opcode/Funct=0x10/0x00
mtc0 \$rt, \$rcop1	# \$rcop1=\$rt      Opcode/Funct=0x10/0x10
eret	# PC=EPC      Opcode/Funct=0x10/0x18

Dica2: Cuidado com exceção dentro de exceção!

Dica 3: Registradores CAUSE = \$13 e EPC = \$14

3) (4.0) Considerando apenas os seguintes tempos de atraso das unidades operativas do caminho de dados de uma CPU MIPS:

Unidade	Tempo
Operações lógicas (and, or, xor) com a ULA	80ps
Operações aritméticas simples (+, -) com a ULA	150ps
Operações aritméticas complexas ( $\times, \div$ ) com a ULA	500ps
Leitura/Escrita no Banco de Registradores	50ps
Leitura da memória cache	300ps
Escrita na memória cache	150ps

O sistema possui uma memória RAM de 2GBytes com tempo de acesso de 100ns, uma memória cache de instruções de 1kWord e uma memória cache de dados também de 1kWord, considere que a taxa de acertos é sempre 100%.

Com relação ao trecho de programa em Assembly MIPS ao lado, onde o LABEL1 corresponde ao endereço 0x00400000, e está sendo executado em Modo Kernel:

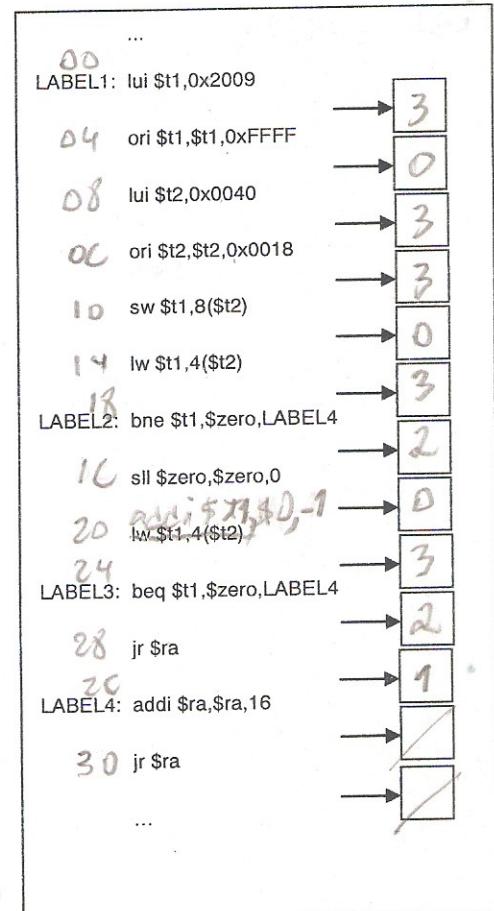
a) (1.0) Para a implementação uniciclo, qual será a maior frequência de clock utilizável? Qual o tempo de execução deste trecho de código neste caso?

b) (1.0) Para a implementação multiciclo vista em aula, qual será a maior frequência de clock utilizável? Qual o tempo de execução deste trecho de código neste caso?

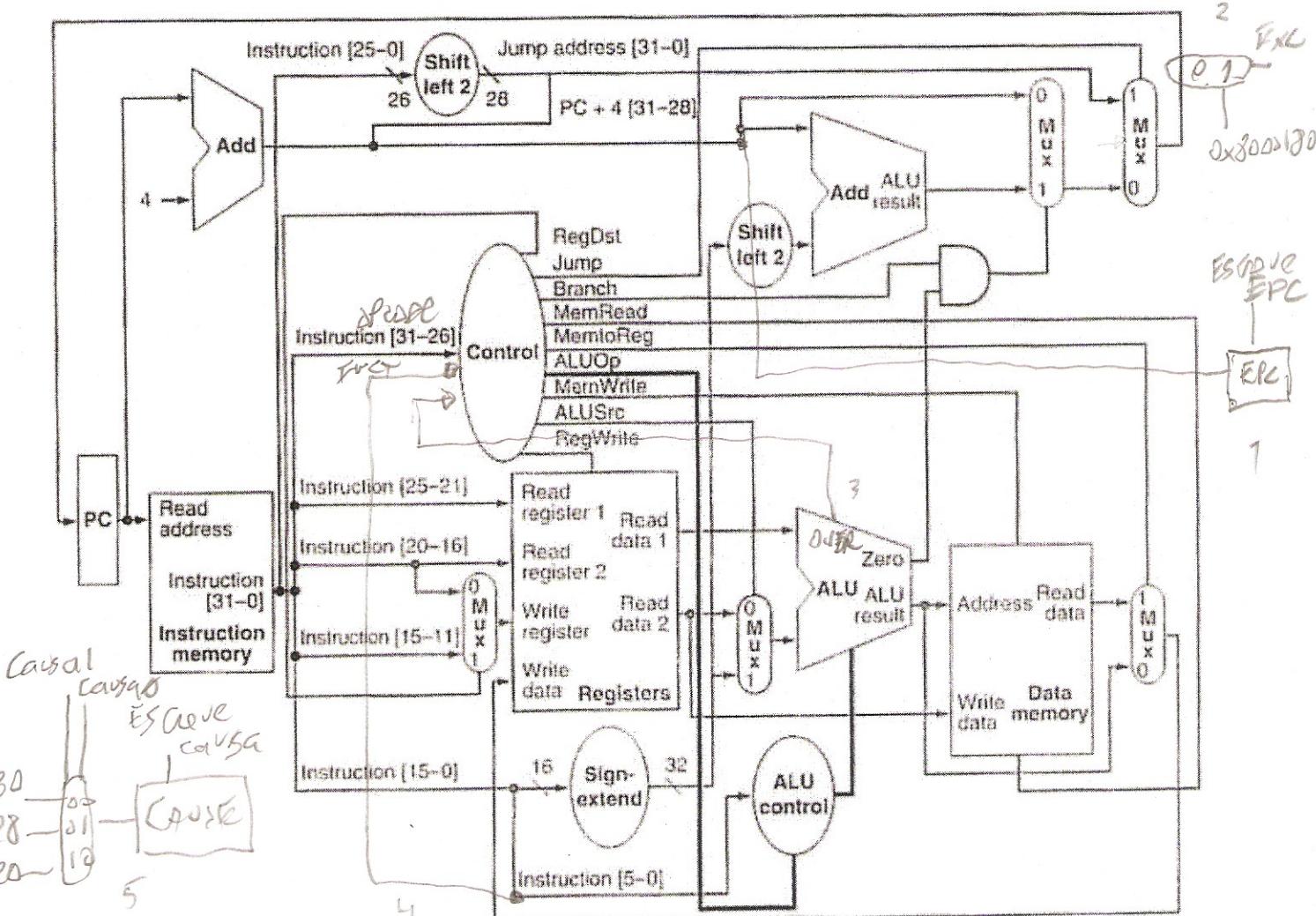
c) (1.0) Qual o tempo de execução da implementação pipeline, se todos os hazards forem tratados apenas com inserção de bolhas? Indique no espaço reservado o número de bolhas necessário (obs.: se não houver necessidade de bolha coloque ‘zero’). Considere que os registradores só podem ser lidos após os mesmos serem escritos no Banco de Registradores, que o branch não é previsto e é avaliado na etapa EX, e as instruções j, jal e jr necessitam 2 ciclos.

d) (1.0) Qual o tempo de execução para implementação em pipeline, se os hazards forem tratados eficientemente pelo processador com forwarding e/ou inserção de bolhas? Preencha o pipeline esquemático na folha em anexo indicando as instruções, bolhas e forwards necessários. Considere que os registradores podem ser escritos e lidos no mesmo ciclo, que o branch previsto como não tomado é avaliado na etapa ID, e as instruções j, jal e jr necessitam 2 ciclos.

**Boa Sorte!!!**



GABARITO



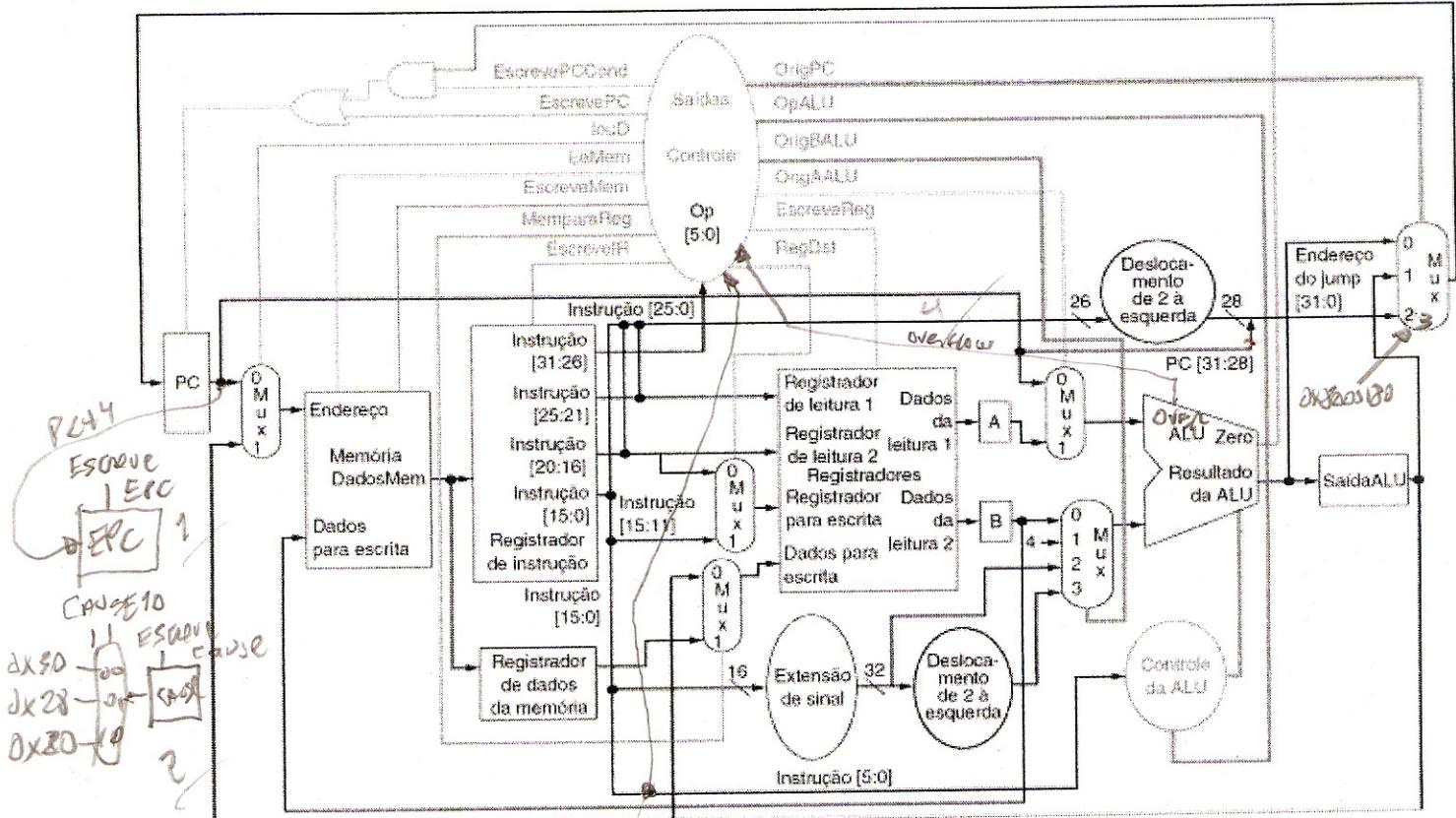
Obs.: Jump = 0;

Instrução	RegDst	OrigALU	Mempara Reg	Escreve Reg	EXC	ESCREVE EPC	ESCREVE CAUSA
formato R	1	0	0	1	0	0	0
lw	0	1	1	1	0	0	0
sw	X	1	X	0	0	0	0
beq	X	0	X	0	0	0	0
OPCODE	1	0	0	1	1	1	1
instr inv.	X	X	X	0	1	1	1
TIPO-R	X	X	X	0	1	1	1

Instrução	Le Mem	Escreve Mem	Branch	ALUOp1	ALUOp0	MulOp	DivOp
formato R	0	0	0	1	0	X	X
lw	1	0	0	0	0	X	X
sw	0	1	0	0	0	X	X
beq	0	0	1	0	1	X	X
mul	0	0	0	0	0	0	0
div	0	0	0	0	0	0	1
inst. Inv.	0	0	0	0	0	0	1
trap	0	0	0	0	0	1	0

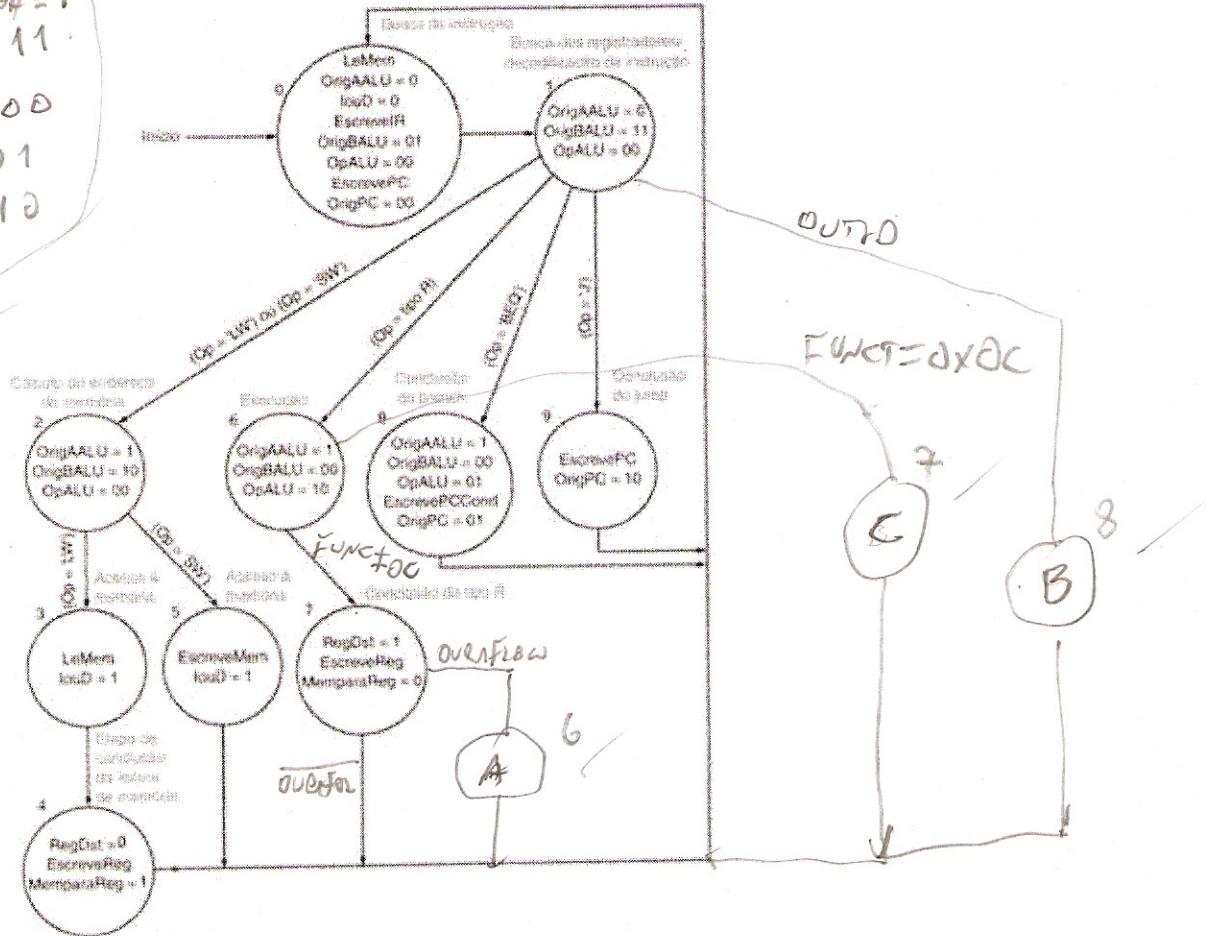
Lyscah

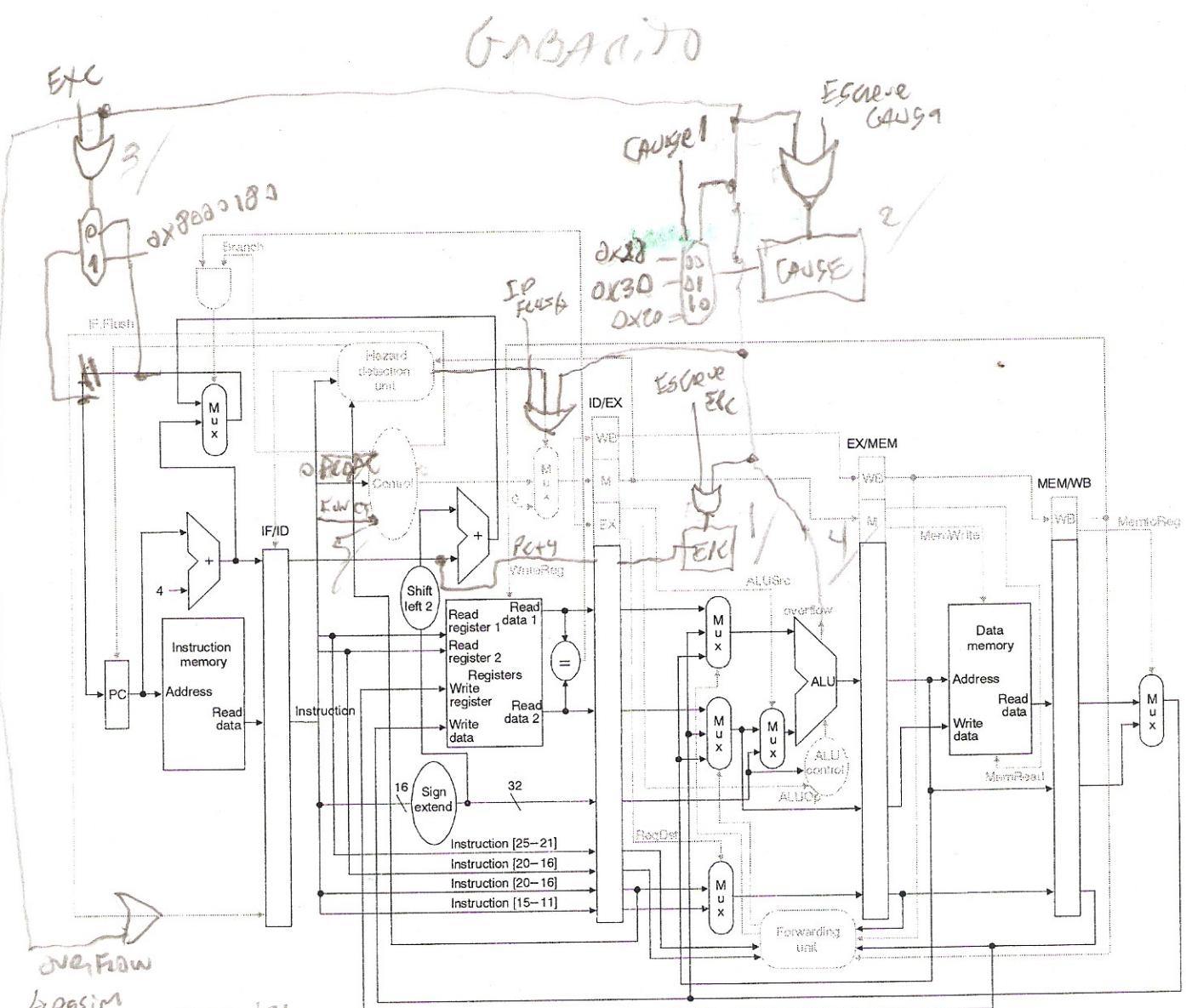
GABANITO



$$\begin{aligned} \text{ES Creve PC} &= 1 \\ \text{ES Creve ELC} &= 1 \\ \text{ES Creve Capif} &= 1 \\ \text{Opie PC} &= 11 \end{aligned}$$

- A: CAUSE10 = 0
  - B: CAUSE10 = 01
  - C: CAUSE10 = 10





→ ASSIM  
OVERFLOW NÃO PRECISA  
SER FEITO NO CONTROLLE  
NÃO PRECISA DESSE CIRCUITO

ou coloca overflow entrando no controlle

Instrução	Linhas de controle do estágio de cálculo de endereço/execução					ESCRV EXC	ESCRV CAUSE	IP flush	EXC
	RegDst	OpALU1	OpALU2	DrigALU					
Formato R	1	1	0	0		0	0	X	
lw	0	0	0	1		0	0	X	
sw	X	0	0	1		0	0	X	
beq	X	0	1	0		0	0	X	
over	1	1	0	0	1	1	0	1	1
instinv	X	0	0	0	1	1	0	1	1
syscall	X	0	0	0	1	1	1	1	1

9

Instrução	Linhas de controle do estágio de acesso à memória		
	Branch	LeMem	Escreve Mem
Formato R	0	0	0
lw	0	1	0
sw	0	0	1
beq	1	0	0
over	0	0	0
instinv	0	0	0
syscall	0	0	0

10

Instrução	Linhas de controle do estágio de escrita do resultado	
	Escrvo Reg	Mem para Reg
Formato R	1	0
lw	1	1
sw	0	X
beq	0	X
over	1	0
instinv	0	0
syscall	0	0



# Universidade de Brasília

Departamento de Ciéncia da Computação  
Disciplina: CIC 116394 – Organização e Arquitetura de Computadores  
Prof. Marcus Vinicius Lamar

Nome: Gaynor.50

Matrícula: \_\_\_\_\_

Matrícula: \_\_\_\_\_

$w_i$

$an_i$

$Lu_i$

$OR_i$

$SW_i$

$LW_i$

$lw$

branch previous cons nos tempos  
não  
o bloco,

$addi$

$neg$

$jr$

o 1 volta

o 1 volta

" "

o 1 volta

o 1 volta

o 1 volta

OAC-A

2012/2

2º PROVA  
GABARITO

1) Folhas em ANEXO

DBS: EPC ← PCL+4  $\Delta$  EFLR → Escalve Mem Reg

2)

eKDATA

MSG1: .ASCII "CAUTION: EXCEPTION"

MSG2: .ASCII "detected at address"

MSG3: .ASCII "\n"

.KTEXT # endereço 0x80000180

05 mfco \$k0, \$13 # SALVA EPC, CAUSE  
mfco \$k1, \$14 e ao e V8  
addi \$sp, \$sp, -16 PI Pode chamar  
sw \$k0, 0(\$sp) syscall PRINT de  
05 sw \$k1, 4(\$sp) dentro do KTEXT?  
sw \$a0, 8(\$sp)  
sw \$v0, 12(\$sp)

05 li \$v1, 0x20 # é syscall?  
breq \$v1, \$k0, CALLSYS # 0x80001000

la \$a0, MSG1  
li \$v0, 4  
SYSCALL } "CAUTION..."

move \$a0, \$k0  
li \$v0, 1  
SYSCALL } CAUSE 05

la \$a0, MSG2  
li \$v0, 4  
SYSCALL } "detect..."

addi \$a0,\$k1,-4	} nostra PC end PC+4	EPC 0,5
li \$v0,1		
syscall		
la \$a2,MSG3	} "n"	
li \$v0,4		
syscall		

TIMEEXEC: lw \$k0,0(\$sp)

lw \$k1,4(\$sp) 0,5

lw \$a0,8(\$sp)

lw \$v0,12(\$sp)

addi \$sp,\$sp,16

mtcd \$x8913 0,5

mtcd \$k1,\$14

ERET

CALLSYS:

0,5

timeexec 0,4

✓ Pode-se considerar ou não  
 3) O tempo de sons → necessita Hi e Lo

a) UMLCIO

$$LW: 300 + 50 + 150 + 300 + 50 = 850$$

$$MUL: 300 + 50 + 500 + 50 = 900$$

superior: → escrever Hi e Lo

$$T_{UMI} = 900 \text{ ns} \rightarrow f_{UMI} = 1,1 \text{ MHz} // \\ = 1,176 \text{ MHz}$$

$$0x3009FFFF = addi $T1, $ZERO, -1$$

$$\text{endereço } 0x00400018 \rightarrow \text{Label 2}$$

$$\text{Logo: } SW \$T1, 8(B12)$$

$$\hookrightarrow \text{MUL} \quad LW \$T1, 4(B12) \rightarrow \text{addi } \$T1, \$ZERO, -1$$

$$LW \$T1, 4(B12)$$

$$\hookrightarrow \$T1 = \text{SU } \$0, \$0, \$0 = 0x00000000$$

Assim: 00 + 28 = 11 instruções

$$T_{UMI} = 11 \times 900 \text{ ns} = 9900 \text{ ns} // \\ 11 \times 850 \text{ ns} = 9350 \text{ ns}$$

b) MULT, CICLO

etapa + Lenta → VLA fazendo MULT

E escrito no Hi e Lo?

$$T_M = 500n + 50n = 550n \text{ s} \\ = 1,818 \text{ MHz} // \\ f_{MUL} = 2 \text{ MHz}$$

$$T_{MULT} = (4+4+4+4+4+5+3+4+4+3+3) \times T$$

$$T_{MULT} = 42 \times 550 \text{ ns} = 23.100 \text{ ns} // \\ = 42 \times 500 \text{ ns} = 21000 \text{ ns}$$

$$c) t_{\text{pipe}} = (11 + 20) \cdot 5500 \text{ m} = 170500 \text{ ns} //$$
$$\cdot 500 \approx 155000 \text{ ns}$$

$$d) t_{\text{pipe}} = (11 + 4) \cdot 5500 \text{ m} = 82500 \text{ ns} //$$
$$\cdot 500 = 41250 \text{ ns}$$