Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2013/1
Prof. Marcus Vinicius Lamar

d0 d1 / d2 d3 d4 d5 d6 d7 d8

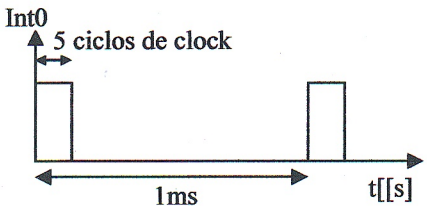
Nome: GABARITO Matrícula: □□/□□□□□□□

Prova 2

1)(6.0) Um dos dispositivos periféricos mais utilizados hoje em dia nos computadores é o controle usado em jogos. Na década de 70, a empresa Atari inovou ao introduzir um controle, chamado joystick, baseado em uma alavanca direcional e um único botão de tiro. Os comandos são dados por contatos elétricos digitais (liga/desliga). Neste controle existem 5 contatos elétricos, 4 direcionais (cima, baixo, esquerda, direita) e 1 para o botão de tiro. Devido à construção física do controle, os contatos direcionais podem ser acionados aos pares, possibilitando a identificação de 8 direções possíveis, com as 4 diagonais sendo identificadas pelo acionamento simultâneo de 2 contatos direcionais.

Desenvolva uma interface baseada em interrupção e MMIO que possibilite o uso simultâneo de 2 controles do modelo Atari (jogadores A e B). Sempre que a interrupção int0 estiver habilitada, o processador recebe a cada 1ms um pulso correspondente ao tempo de 5 ciclos de clock do processador no sinal de interrupção int0. Utilize o endereço 0x60000000 para endereçar um registrador que armazena em uma palavra de 32 bits os sinais vindos dos dois controles, e o bit 0 da palavra do registrador no endereço 0x60000004 para habilitar e desabilitar esta interrupção (Obs.: as instruções mtc0 e mfc0 não estão e nem devem ser implementadas).

Dica: Desabilite a interrupção enquanto a rotina de tratamento de exceções estiver sendo executada (*modo kernel*).



a) **(1.5)** Modifique adequadamente os Caminhos de Dados, fornecidos nas folhas em anexo, dos MIPS Uniciclo E Multiciclo, de forma a implementar a detecção e tratamento da interrupção int0 e a instrução eret. (Não precisa implementar o coprocessador 0, apenas as funcionalidades básicas requeridas nesta questão)

b) **(1.5)** Modifique adequadamente os Blocos Controladores, fornecidos nas folhas em anexo, dos MIPS Uniciclo, E Multiciclo, de forma a implementar a detecção e tratamento da interrupção int0 e a instrução eret.

c) **(1.0)** Escreva, a partir do endereço 0x80000100, uma rotina para o tratamento desta interrupção int0 que retorne ao programa do usuário, nos registradores $k0 e $k1, os comandos recebidos dos controles A e B respectivamente, com os seguintes valores:

Direção:		Tiro=direção+0x01
Cima= 0x02	cima/direita=0x0A	
Baixo=0x04	cima/esquerda= 0x12	
Direita = 0x08	baixo/direita= 0x0C	
Esquerda= 0x10	baixo/esquerda= 0x14	

d) **(2.0)** Escreva um programa exemplo que utilize o syscall PLOT, para plotar um pixel azul (jogador A) e um pixel vermelho (jogador B) nas respectivas posições (X,Y) na tela, movimentando-se sem deixar rastro de acordo com os comandos recebidos. Sempre que o botão de tiro estiver pressionado, o pixel deve se mover 2 vezes mais rápido.

Considere: Fundo branco homogêneo.
Tela com resolução de 320x240 pixels. Posição (0,0) corresponde ao canto inferior esquerdo.
Posições iniciais: jogador azul (80,60) jogador vermelho (240,180)
Não precisa verificar movimentação fora dos limites da tela.
O comando syscall automaticamente mascara a interrupção enquanto é executado.

Serviço	$v0	Argumentos	Resultados
...	...	...	...
Plot	45	$a0=X $a1=Y $a2=cor	Plota um pixel na posição (X,Y)=($a0,$a1) com a cor $a2
...	...	...	...

Paleta de cores de 3 bits BGR: 0-preto, 1-vermelho, 2-verde, 3-amarelo, 4-azul, 5-magenta, 6-ciano, 7-branco

2) (6.0) Considerando os seguintes tempos de atraso das unidades operativas do caminho de dados de uma CPU MIPS:

Unidade	Tempo
Operações lógicas com a ULA	100ps
Operações aritméticas com a ULA	150ps
Leitura/Escrita no Banco de Registradores	50ps
Leitura da memória cachê	300ps
Escrita na memória cachê	150ps

O sistema possui uma memória DRAM de 4GiB com tempo de acesso de 6ns, uma memória cache de instruções de 32kiB e uma memória cache de dados também de 32kiB. Considere que a taxa de acertos da memória de instruções é de 100%, e a taxa de acertos na memória de dados de 80%.
Com relação ao procedimento em Assembly MIPS dado, onde o label PROC corresponde ao endereço 0x00400000, sendo executado em *Modo Kernel*, e que só é concluído após a execução do jr $ra, responda:

Obs.: beq e bne estão indicadas com argumentos $rs,$rt,deslocamento

a) **(1.0)** Para a implementação uniciclo, qual será a maior frequência de clock utilizável? Qual o tempo necessário para a execução deste procedimento?

b) **(1.0)** Para a implementação multiciclo vista em aula, qual será a maior frequência de clock utilizável? Qual o tempo necessário para a execução deste procedimento?

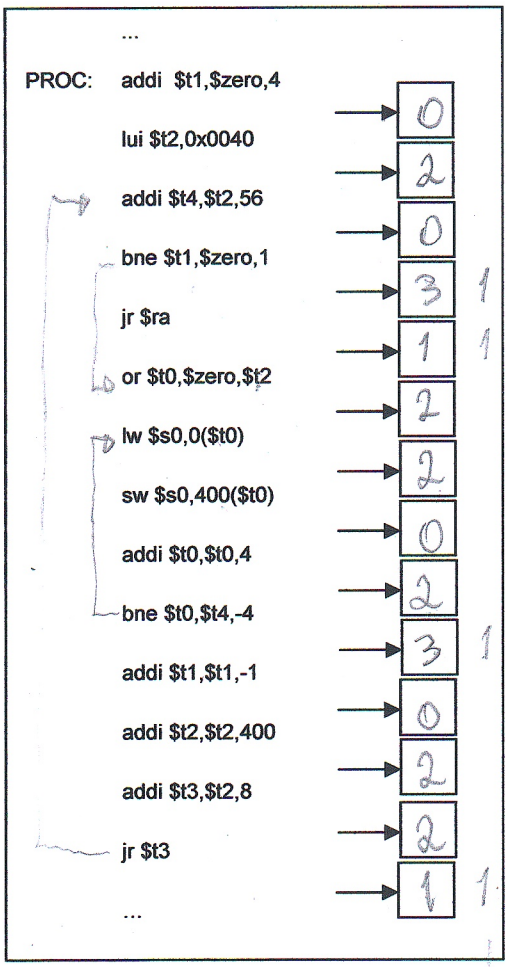
Obs.: Considere que o controle multiciclo possui um estado de espera para obter uma leitura/escrita da memória de dados em caso de falha da cache.

jr e bne necessitam 3 ciclos. lui e addi necessitam 4 ciclos.

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 ao lado, o número de bolhas necessário (obs.: se não houver necessidade de bolha coloque 'zero'). Considere que os registradores podem ser lidos e escritos no mesmo ciclo, que o branch não é previsto e é avaliado na etapa MEM, e as instruções j, jal e jr necessitam 2 ciclos.

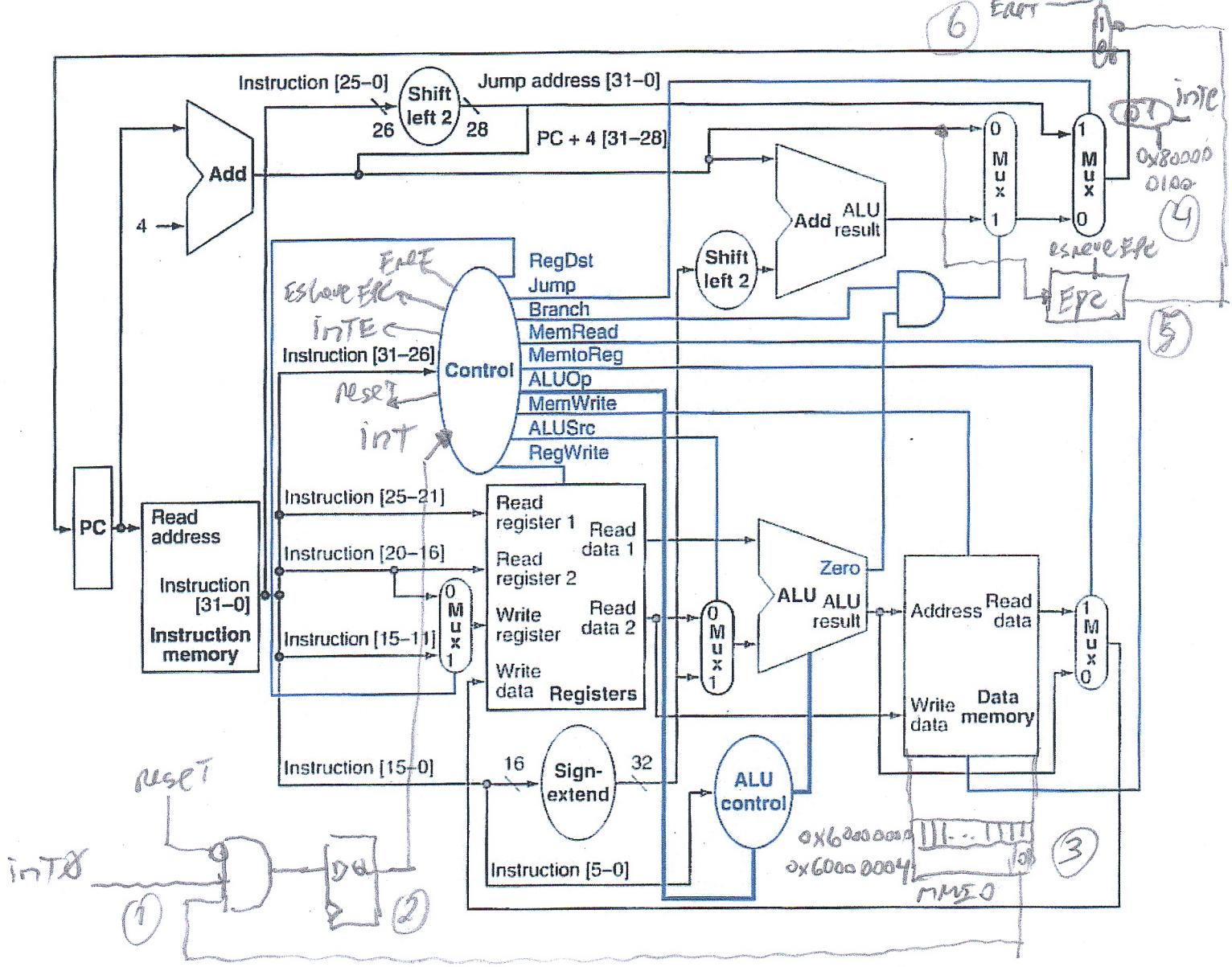
d) **(1.0)** Qual o tempo de execução para implementação em pipeline, se os hazards forem tratados eficientemente pelo processador com forwarding e/ou inserção de bolhas? Preencha o pipeline esquemático na folha em anexo indicando as instruções, bolhas e forwards necessários. Considere que os registradores 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.

No pipeline desconsidere o tempo de latência.



e) **(2.0)** Qual o endereço que a instrução jr $ra é executada?

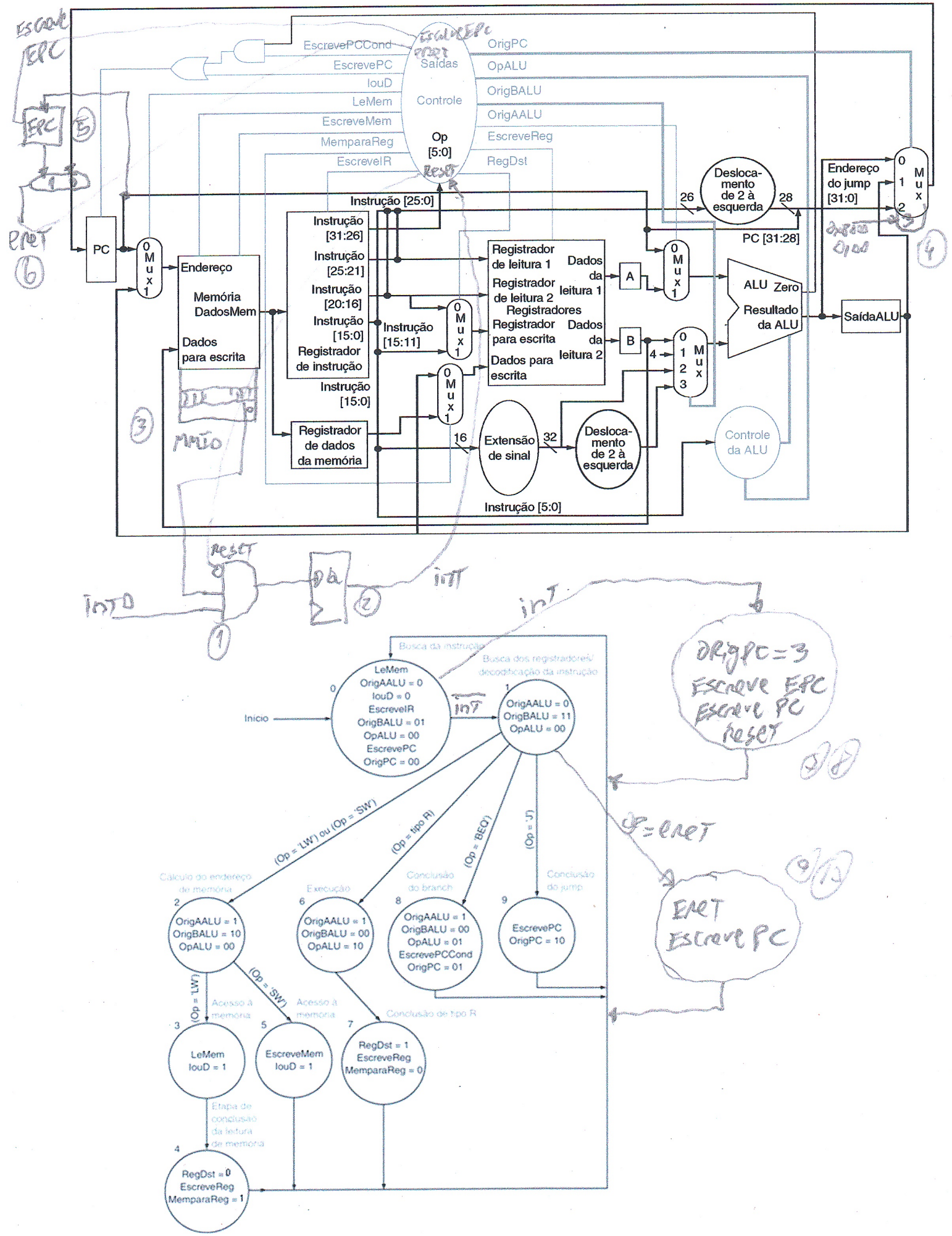
Boa Sorte!!!

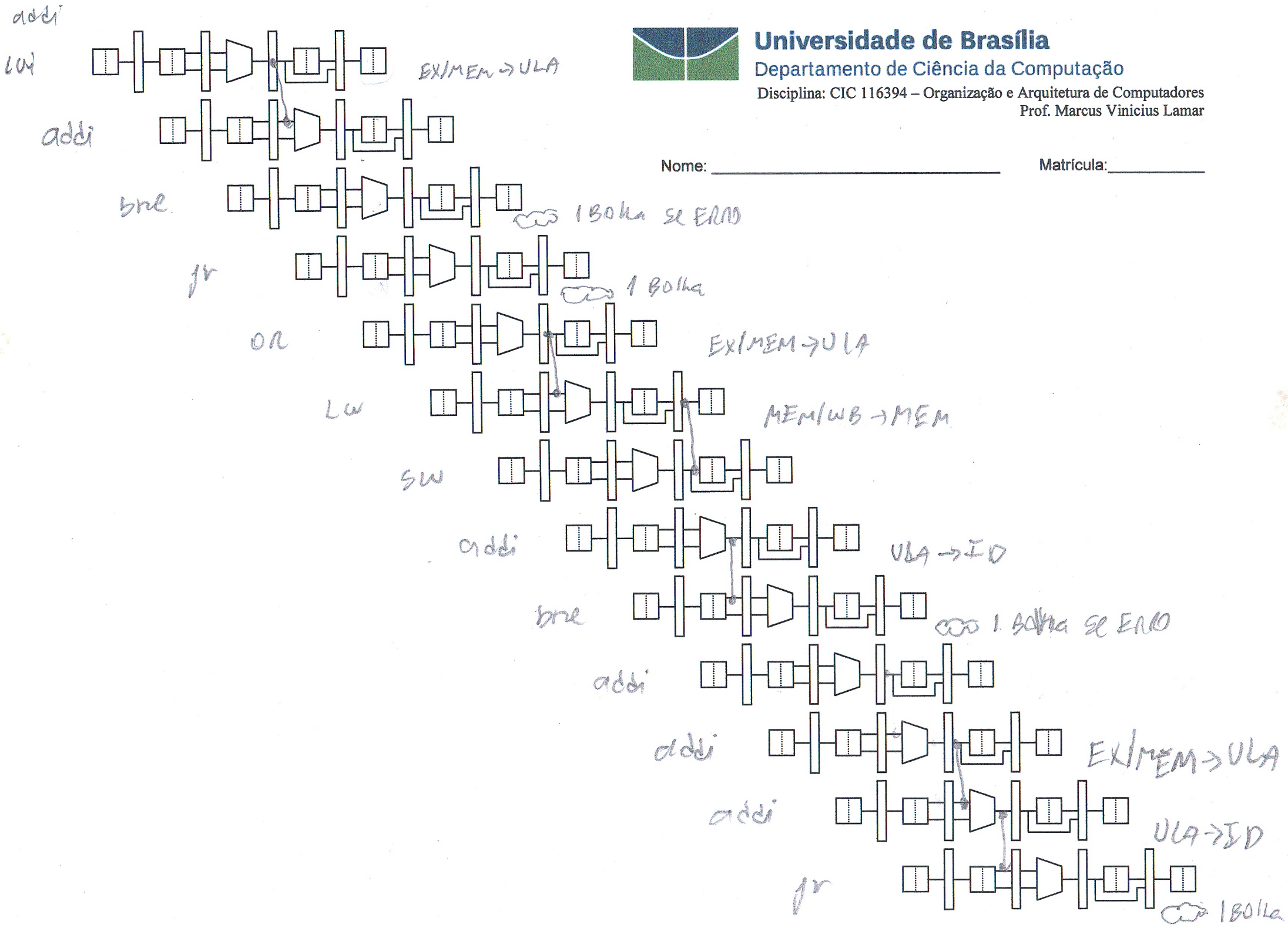


Não precisa do reg CAUSE

Instrução	RegDst	OrigALU	Mempara Reg	Escreve Reg	Le Mem	Escreve Mem	Branch
formato R	1	0	0	1	0	0	0
lw	0	1	1	1	1	0	0
sw	X	1	X	0	0	1	0
beq	X	0	X	0	0	0	1
ERET	X	X	X	0	0	0	0
INT	X	X	X	0	0	0	0

Instrução	ALUOp1	ALUOp0	ERET	Escreve EPC	INTE	RESET		
formato R	1	0	0	0	0	0		
lw	0	0	0	0	0	0		
sw	0	0	0	0	0	0		
beq	0	1	0	0	0	0		
ERET	0	0	1	0	0	0		
INT	0	0	0	1	1	1		


EscreveEPC
EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op [5:0]
Reset
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
EPC
PC
0 Mux 1
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:0]
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de dados da memória
Registradores
Registrador de leitura 1
Registrador de leitura 2
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
0 1 2 3 Mux
ALU
Zero
Resultado da ALU
SaídaALU
Deslocamento de 2 à esquerda
26
28
PC [31:28]
Endereço do jump [31:0]
0 1 2 Mux
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
Busca da instrução
Busca dos registradores/ decodificação da instrução
Início
LeMem
OrigAALU = 0
IouD = 0
EscreveIR
OrigBALU = 01
OpALU = 00
EscrevePC
OrigPC = 00
OrigAALU = 0
OrigBALU = 11
OpALU = 00
(Op = 'LW') ou (Op = 'SW')
(Op = tipo R)
(Op = 'BEQ')
(Op = 'J')
Cálculo do endereço de memória
Execução
Conclusão do branch
Conclusão do jump
OrigAALU = 1
OrigBALU = 10
OpALU = 00
OrigAALU = 1
OrigBALU = 00
OpALU = 10
OrigAALU = 1
OrigBALU = 00
OpALU = 01
EscrevePCCond
OrigPC = 01
EscrevePC
OrigPC = 10
(Op = 'LW')
(Op = 'SW')
Acesso à memória
Acesso à memória
Conclusão do tipo R
LeMem
IouD = 1
EscreveMem
IouD = 1
RegDst = 1
EscreveReg
MemparaReg = 0
Etapa de conclusão da leitura de memória
RegDst = 0
EscreveReg
MemparaReg = 1
int
OrigPC = 3
Escreve EPC
Escreve PC
Reset
PC = EPC
EPC
Escreve PC



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

Nome: _____ Matrícula: _____

OAC - A 20131

2º PROVA

GABARITO

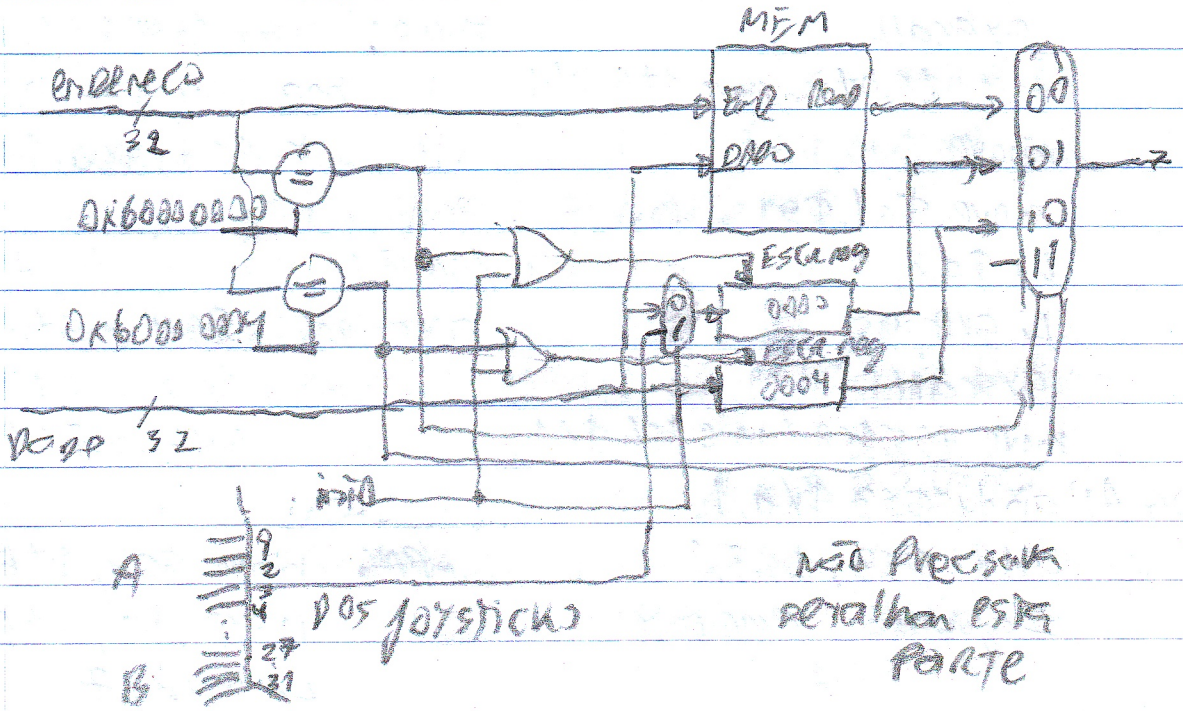
1) Requerimentos:

- Detecção INTR
  - → Receber sinal INTR na forma do gráfico
  - → Mascarar
  - → Definir formato dos dados no Registrador
  - → enviar sinal INTR mascarado p/ CONTROLE
  - → enviar endereço 0x8000 0100 p/ PC
  - → Armazenar PC+4 no EPC
- Instrução ERET
  - → Bloco controle detecta o opcode/funct
  - → Armazena em PC o valor de EPC

→ VER Folhas em Anexo.

OBS.: Nos endereços 0x6000 0000 e 0x6000 0004 o processador acessa 2 Registradores de 32 bits externos e não a posição na memória.

Logo tem um circuito:



c) .KTEXT

```
sw $zero,4($fp)      # fp já deve estar com
addi $at,$zero,1       o valor 0x60000000
lw $k0,0($fp)          p ter certeza que será
srl $k1,$k0,26         mascarada logo na 1ª
andi $k0,$k0,0x1F      instrução
sw $at,4($fp)  # habilita
eret
```

(desabilita)

d)

```
.text
la $k0,0x60000000
li $s0,80
li $s1,60
li $t0,240
li $t1,180

LOOP: move $a0,$t0
      move $a1,$t1
      li $a2,4
      li $v0,45
      syscall
      move $t5,$t0 / move $t6,$t1
      move $a0,$s0        ↓
      move $a1,$s1      posições
      li $a2,1          anteriores
      li $v0,45           ↑
      syscall
      move $s5,$s0 / move $s6,$s1
JOGA: andi $s3,$k0,1
      addi $s3,$s3,1
      #ajusta a velocidade
      # 1 ou 2
```

```
CIMA: andi $s4,$k0,2
      beq $s4,$zero,BAIXOA
      add $s1,$s1,$s3
      j DIRA
BAIXOA: andi $s4,$k0,4
      beq $s4,$zero,DIRA
      sub $s1,$s1,$s3
DIRA: andi $s4,$k0,8
      beq $s4,$zero,ESQA
      add $s0,$s0,$s3
      j JOGB
ESQA: andi $s4,$k0,16
      beq $s4,$zero,JOGB
      sub $s0,$s0,$s3

JOGB: igual ao jogA
trocando: $s0 → $t0
          $s1 → $t1
          $k0 → $k1

LIMPA: move $a0,$t5
       move $a1,$t6
       li $a2,7
       li $v0,45
       syscall
```

move $a0, $s5
move $a1, $s6
li $a2, 7
li $v0, 45
syscall
j loop

2) o programa são 2 loops
1º $t1, 4 a 0 : repete 4x
2: $t0, de 0x00400000 a 0x0040 0000 + 56
repete 14x
↳ copia as 14 instruções do programa
p/ o endereço + 400 e executa de novo
a partir de lá

a) Uniciclo: instrução mais longa
↳ erro na cache c/ LOAD

$T = 300p + 50p + 150p + 6000p + 50p$
↓
100% acerto
mem. instr

$T = 6550p \rightarrow f = 152,67\ MHz$

1ª vez

$N^o ciclos = 2 + 4 \times (3 + 14 \times 4 + 4) + 2 + 1 = 257$
$t = 257 \times 6550p = 1,683\ \mu s$

b) Multiciclo: etapa + longa
como temos 1 estado de espera p/ falhas
na cache →
temos $T = 300ps \rightarrow f = 3,33\ GHz$
↳ cujo penalidade falha $= \frac{6n}{300p} = 20$ ciclos

LW + SW → 80% Acerta

nº ciclos = 4+4 + 4x(4+3+4 + 14(5+4+4+3) +
+ 4+4 + 4+3) + 4+3+3

Logo:
c/ Erro na cache

Acerto — Erro

Nº ciclos = 4+4 + 4x(4+3+4 + 14x(9x0,8 + 20x0,2 +
+ 4+3) + 4+4+4+3) + 4+3+3 = 1141,2

t = 1141,2 x 300p = 342,36 ns //

c) T = 300ps — só c/ bolhas
nº ciclos = 2+2 + 4x(3+5 + 14x(4+7) + 4+5) +
+ 2+3+1+1
= 896,6

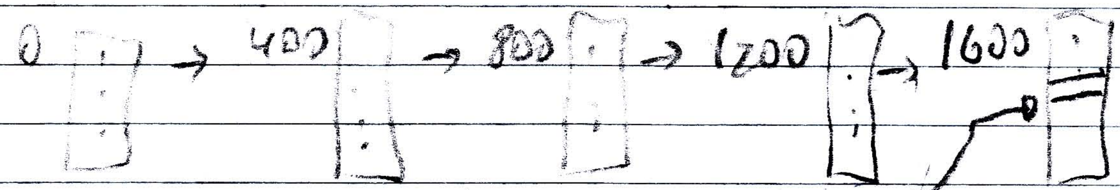
2x0,8 + 20x0,2 + 2

t = 896,6 x 300p = 268,98 ns //

d) T = 300ns — vetimo + beq — acerta
nº ciclos = 2 + 4x(3+1 + 14x(4+1) - 1 + 4 + 1) +
+ 2+1+1.
= 463,6

2x0,8 + 20x0,2 + 2

t = 463,6 x 300p = 139,08 ns //

e) 

Logo: endereço do jr $ra executado
END = 0x0040 0000 + 1600 + 16
END = 0x0040 0650

1616 → 0x650