



Nome: \_\_\_\_\_ Matrícula: \_\_\_\_\_

## Prova 2

1) (4.0) Uma das principais características da arquitetura MIPS é o acesso a memória de dados ocorrer apenas com o uso das instruções **lw** e **sw**, chamada portanto de *arquitetura load-store*. Existem arquiteturas que permitem outras instruções tenham acesso à memória.

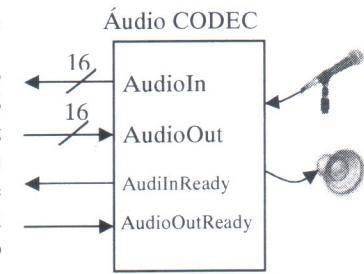
1.1) (3.0) Desenhe o caminho de dados completo para a organização MIPS multiciclo de modo a implementar, além das 9 instruções vistas em aula, as seguintes instruções:

- a)(0.5) jr \$rs # PC=R[\$rs]
- b)(0.5) bne \$rs,\$rt,LABEL # se R[\$rs]>R[\$rt] então PC=LABEL
- c)(1.0) addm1 \$rd,\$rs,\$rt # R[\$rd]=R[\$rs]+Memoria[R[\$rt]]
- d)(1.0) addm2 (\$rd),\$rs,\$rt # Memoria[R[\$rd]]=R[\$rs]+R[\$rt]

1.2)(1.0) Desenhe o diagrama de estados da Unidade de Controle para a CPU completa, isto é, a ISA com as instruções antigas vistas em aula e as novas instruções acima.

2) (5.0) O interfaceamento direto de dispositivos periféricos com microprocessadores, geralmente requer uso de portas de Entrada e Saída (I/O) e instruções *in* e *out*.

Deseja-se receber e enviar amostras de voz através de um simples CODEC de áudio (já configurado) mostrado na figura ao lado, onde as linhas de AudioIn e AudioOut são de 16 bits complemento de 2. O pino de AudioInReady é ativado quando uma nova amostra de voz está disponível em AudioIn. O pino de AudioOutReady deve ser ativado após a leitura de AudioIn e o envio da amostra de saída em AudioOut, indicando ao CODEC que a amostra de entrada foi lida e uma amostra de saída está disponível. Ao receber a ativação do sinal AudioOutReady o CODEC desativa o pino AudioInReady e espera que o pino SampleOutReady seja desativado para reiniciar o ciclo.



A frequência de clock do processador é de 10MHz e a frequência de amostragem do CODEC de 10kHz.

2.1) (2.0) Sabendo que o processador MIPS uniciclo desenvolvido em aula não possui portas de I/O.

- a) (1.0) Projete uma interface capaz de controlar este CODEC, que utilize apenas a ISA reduzida de 9 instruções.
- b) (1.0) Escreva o trecho do programa em Assembly MIPS que realize a tarefa de reproduzir no alto falante o sinal de áudio capturado pelo microfone amplificado de um fator de 2.

2.2) (3.0) Incremente o processador MIPS uniciclo desenvolvido em aula de modo a incorporar 4 portas de entrada e 4 portas de saída de 32 bits cada e as instruções:

inw \$rs,\$rt # R[\$rs]= dado disponível na porta de entrada R[\$rt]  
outw \$rs,\$rt # porta de saída R[\$rt] = R[\$rs]

- a) (1.0) Desenhe o Caminho de Dados completo indicando as alterações necessárias.
- b) (1.0) Especifique os requerimentos da Unidade de Controle completa com as alterações necessárias.
- c) (1.0) Escreva o trecho de programa em Assembly MIPS que realiza a tarefa de reproduzir no alto falante o sinal de áudio capturado pelo microfone amplificado de um fator de 2.

3) (3.0) Considerando apenas os seguintes tempos de atraso das unidades operativas do caminho de dados de uma CPU MIPS:

Operação com a ULA: 100ps  
Acesso ao Banco de Registradores: 50ps  
Leitura da memória: 200ps  
Escrita na memória: 300ps

Com relação ao trecho de programa em assembly MIPS ao lado:

3.1)(0.5) Para a implementação uniciclo, qual será a maior frequência de clock utilizável?  
Qual o tempo de execução deste trecho de código neste caso?

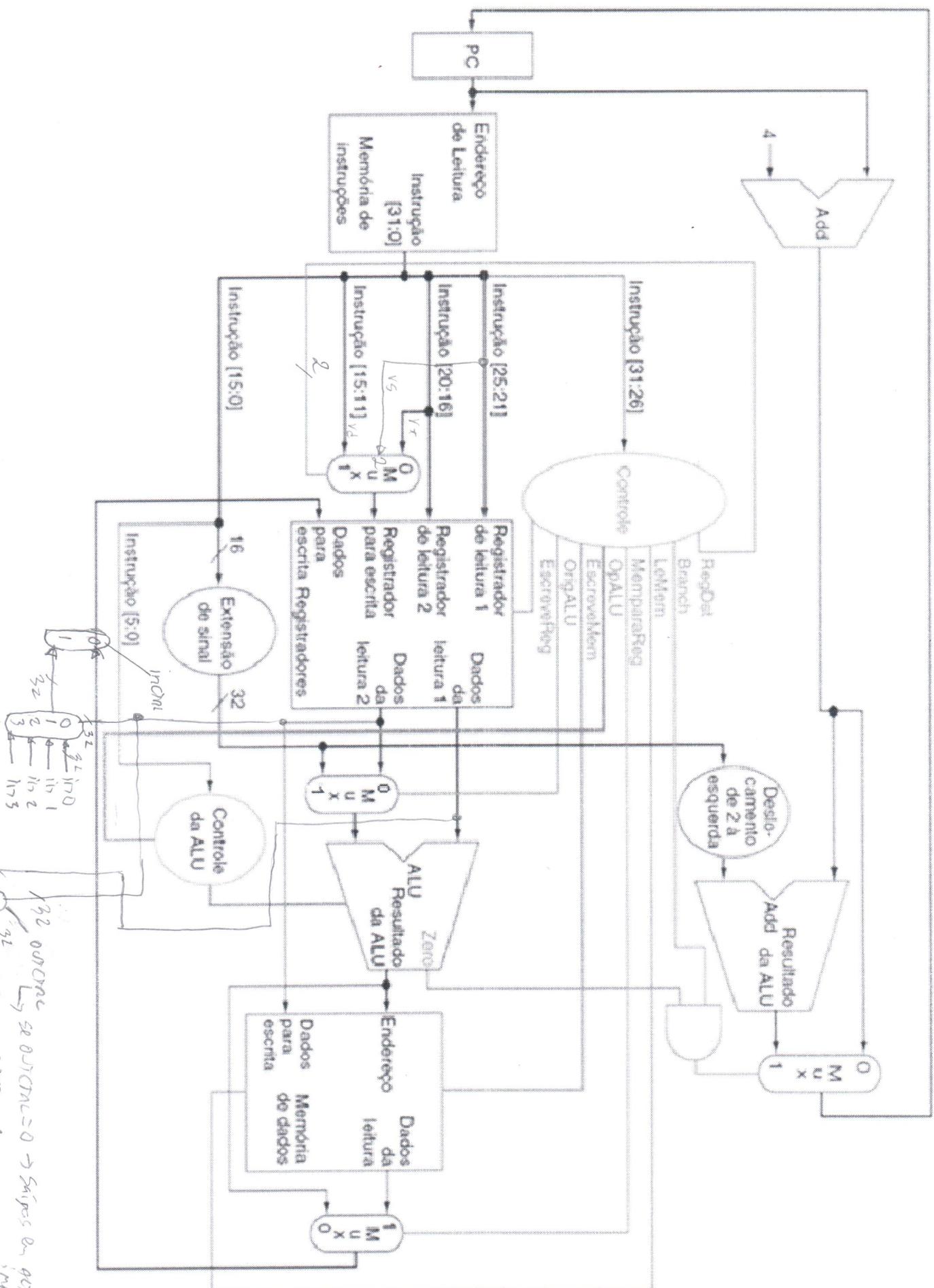
3.2)(0.5) Para a implementação multiciclo vista em aula, qual será a maior frequência de clock utilizável?  
Qual o tempo de execução deste trecho de código neste caso?

3.3)(1.0) Qual o tempo de execução da implementação pipeline, se todos os hazards forem tratados apenas com inserção de bolhas? Indique em cada linha do programa o número de bolhas necessário.

3.4)(1.0) Qual o tempo de execução da implementação pipeline, se os hazards forem tratados eficientemente pelo processador com forwarding e/ou bolhas e/ou execução fora de ordem? Desenhe o pipeline esquemático indicando as instruções, bolhas e forwards sugeridos.

...	
lw \$4, 100(\$5)	0
and \$6, \$3, \$5	1
sw \$4,60(\$4)	0
lw \$6,120(\$6)	2
sw \$6,140(\$6)	0
sub \$10,\$14,\$3	2
sw \$10,160(\$14)	0
lw \$21,140(\$10)	0
sw \$10,130(\$10)	0
add \$20,\$21,\$10	1
...	

Boa Sorte!!!



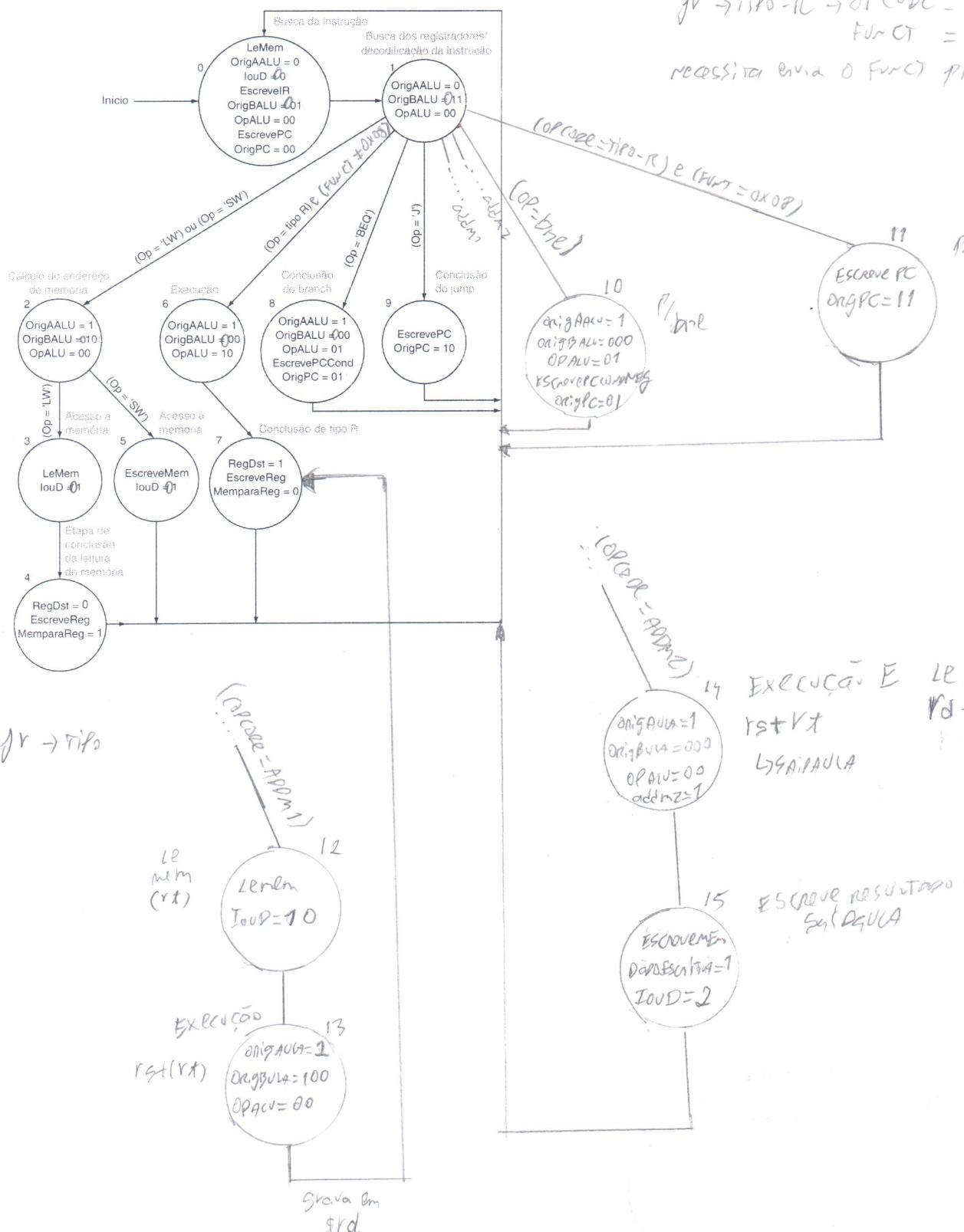
OpALU = 0 → Skips branch  
 OpALU = 1 → Jump if zero → Só pula se zero=1 → Só pula se zero=1  
 Só pula se zero=1  
 Recebe a entrada  
 Dados em que impõe

Instituição	RegDst	OrigALU	Mempara Req	Escreve Reg	Le Mem	Escreve Mem	Branch	ALUOp1	ALUOp0	Int/Ext	Out/In
formato R	01	0	0	1	0	0	0	1	0	0	0
lw	00	1	1	1	1	0	0	0	0	0	0
sw	XX	1	X	0	0	1	0	0	0	0	0
beq	XX	0	X	0	0	0	1	0	1	0	0
intw	10	X	X	1	0	0	0	X	X	1	0
outw	XX	X	X	0	0	0	X	X	0	0	1

2.2)  
b)

1.2)

SEMPRE ADDR2=0  
DEFALT DADO ESCRITA=0

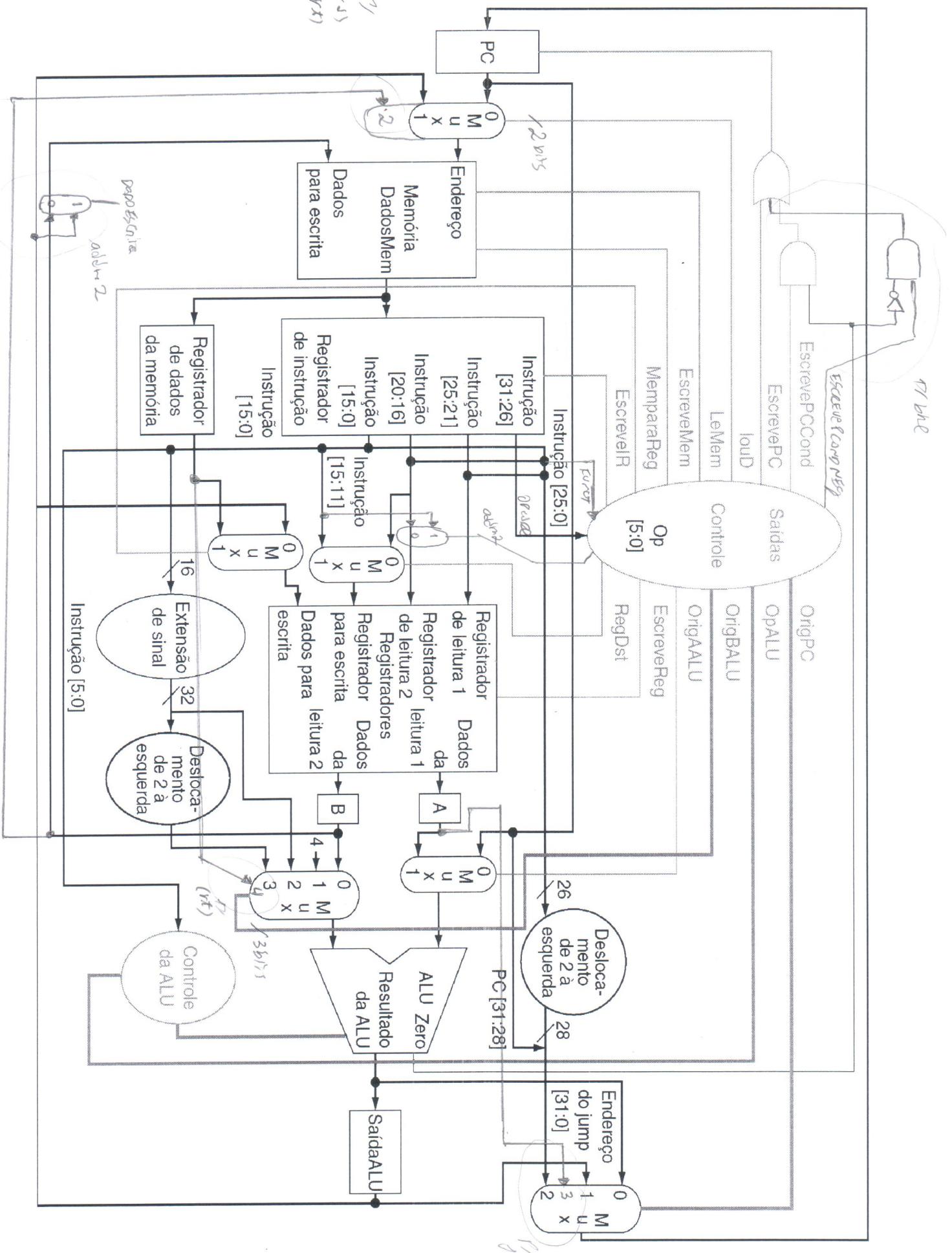


JV → TIPO-R → OPCODE = 0x00  
FuncT = 0x08  
necessita enviar o FuncT para NLL

TIPO = ADDM2

EXECUÇÃO E  
rst+vt  
LEGITIMIA

ESCREVER RESULTADO EM Vd  
Escrever resultado em Vd B



## OAC - TURN A

200912

## 2º Prova - FABRILITO

1) Caminho de dados em Folha anexa

- a) jr \$rs → armazena \$rs no PC
  - b) bne \$rs,\$rt, LABEL → similar ao beg
  - c) addi \$rd,\$rs, (\$rt)

↳  $\text{Sift} \rightarrow$  endereçar memória pt leitura

L>5km, can T; PO R

d) addm2 (\$rd),\$rs,\$rt

↳  $\text{Frd} \rightarrow$  se descomponen moléculas de ESCITA

↳ similar STORE

7.2) na Folha em anexo

2)

201) ISA 9 Instruções → acesso à dispositivo externo.ável

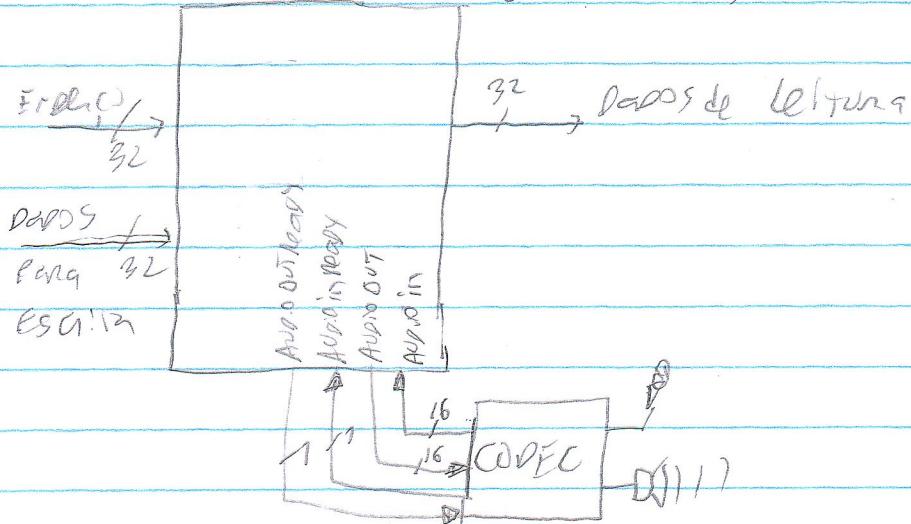
é feito pelas insinuações de SW, logo malefícios de dispositivo em ENHIFU de memória

4 virtudes de poens → 4 grandesas de min. 1

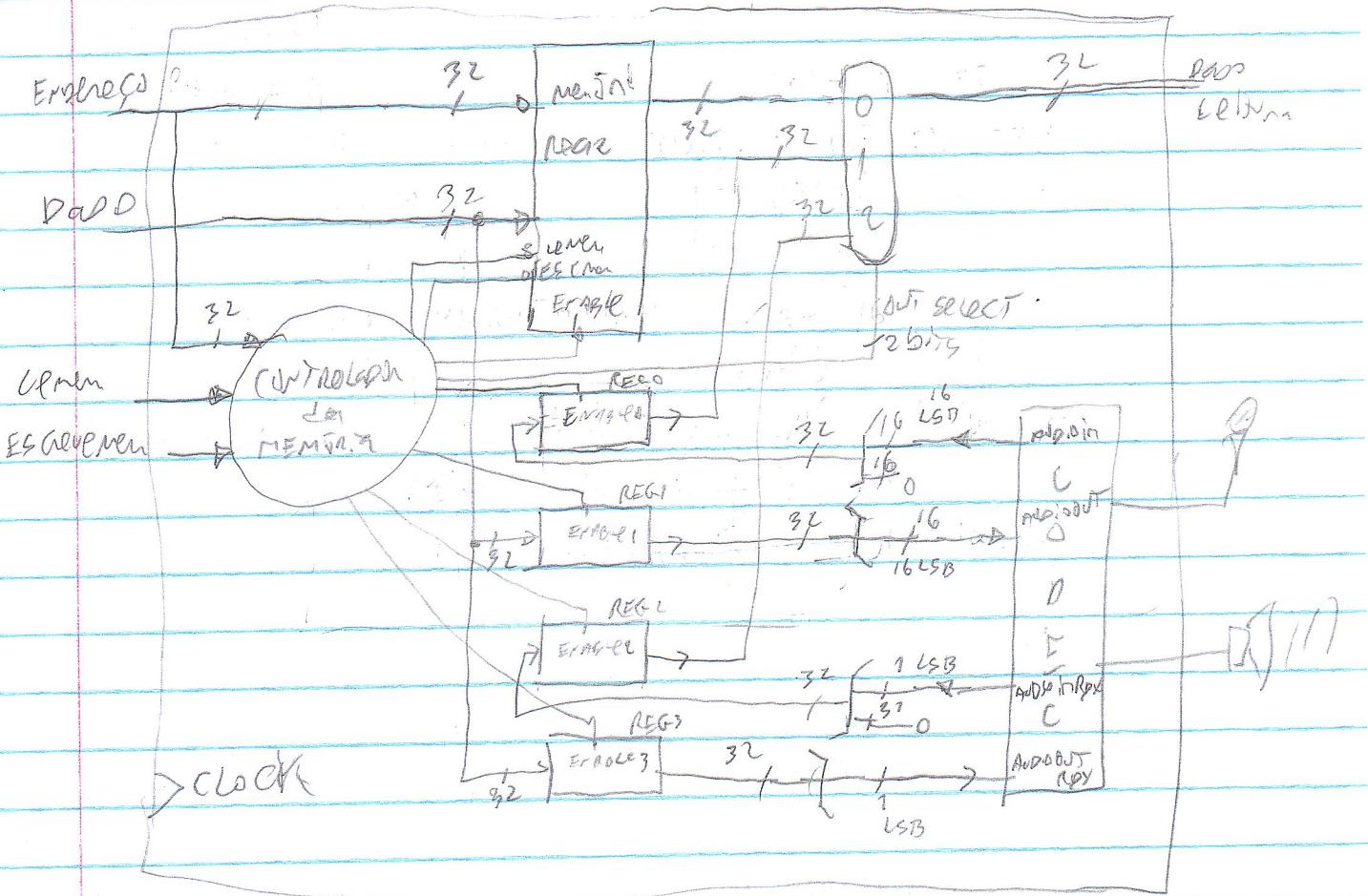
$(w, sw \rightarrow \text{label})$  escribe 32 bits (figs):

## REMINING DAWGS (MISSOURI)

9)



## Mapamento do dispositivo: MEMÓRIA



Carregadores → controla onde se encontra a escrita

Lêdores → controla onde se encontra a leitura

REG0 e REG2 → ESCRITORES pelo CODEC  
LÊDORES pelo PROCESSADOR

REG1 e REG3 → ESCRITORES pelo PROCESSADOR  
LÊDORES pelo CODEC

Por exemplo: REG0 → 0x1000 0000 Dados in

REG1 → 0x1000 0001 Dados out

REG2 → 0x1000 0002 In ready

REG3 → 0x1000 0004 Out ready

Li \$t0, 1  
 b) la \$t0, 0x1000 0000 # END. BASE  
 inicio: sw \$zero, 12(\$t0) # desativa OUT ready  
 loop: lw \$t1, 8(\$t0) # leitura FAZ  
 beg \$t1, \$zero, loop # aguarda inReady = 1  
 lw \$t2, 0(\$t0) → SLL \$t2, \$t2, 1 # LE  
 sw \$t2, 4(\$t0) # Escreve DATA OUT  
 sw \$t0, 12(\$t0) # ativa OUT ready  
 loop2: lw \$t1, 8(\$t0) # leitura FAZ  
 beg \$t0, \$t1, loop2 # aguarda inReady = 0  
 sw \$zero, 12(\$t0) # desativa OUT ready  
 f inicio

## 2.2) MIPS unificado

a) Caminho de dados na Folha em anexo

b) na Folha em anexo

c) Li \$s0, 0 # data in  
 Li \$s1, 1 # data out  
 Li \$s2, 2 # inReady  
 Li \$s3, 3 # outReady

0	→	data in
1	→	data out
2	→	inReady
3	→	outReady

inicio: OUTW \$s0, \$s3 # outReady=0  
 loop: lh \$t1, \$s2  
 beg \$t1, \$zero, loop # aguarda inReady = 1  
 inw \$t2, \$s0 → SLL \$t2, \$t2, 1 # LE DATA IN \$t2  
 OUTW \$t2, \$s1 # Escreve DATA OUT  
 OUTW \$s1, \$s3 # Setando OUT ready  
 loop2: lh \$t1, \$s2  
 beg \$s0, \$t1, loop2 # aguarda inReady = 0  
 OUTW \$s0, \$s3 # desativa OUT ready  
 f inicio

3)

3.1) tempo:

$$t_W = 200p + 50p + 100p + 200p + 50p = 600ps$$

$$t_W = 200p + 50p + 100p + 300p = 650ps \rightarrow \text{inversor} + 4,7q_1$$

$$\log_2 f_{\max} = \frac{1}{650p} \approx 1,5384 \text{ GHz}$$

$$t_{\text{OM}} = 10 \times 1 \times 650p = 6,5ns$$

$$3.2) f_{\max} = \frac{1}{300p} = 3,3333 \text{ GHz}$$

$$t_{\text{multi}} = (5+4+4+5+4+4+4+5+4+4) \times 300p$$

$$t_{\text{multi}} = 12,9 \text{ ns}$$

$$3.3) t_{\text{triple}} = (10 + 6 \text{ bolhas}) \times 300p = 4,8 \text{ ns}$$

3.4) somente c/ execuções fora de ordem

lw \$4,100(\$8)  sem bolhas

and \$6,\$3,\$5  sem forwards

sub \$10,\$14,\$3 

sw \$4,60(\$4) 

lw \$6,120(\$6) 

sw \$10,160(\$14) 

lw \$21,140(\$10) 

sw \$6,140(\$6) 

sw \$10,130(\$10) 

add \$20,\$21,\$10 

$$t_{\text{triple}} = 10 \times 300p = 3 \text{ ns}$$