

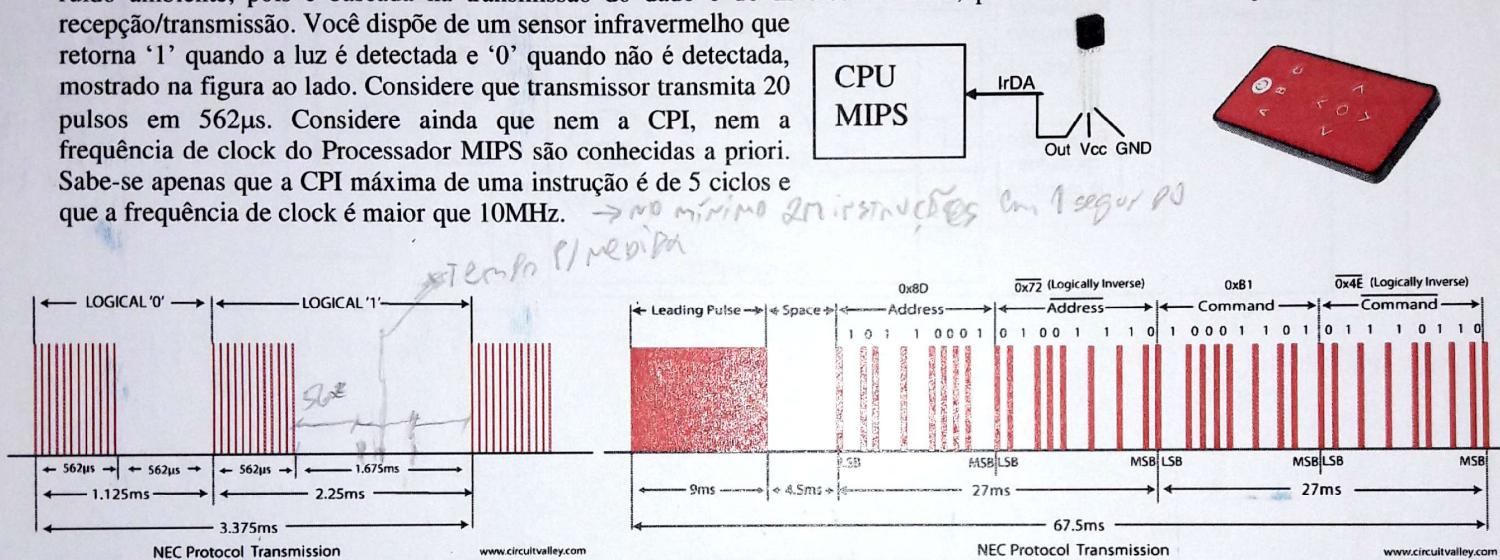


Nome: GABARITO

Matrícula:       /      /      /      /      /      /      /      /

## Prova 2

**1)(5.0)** O controle remoto sem fio veio revolucionar o modo como interagimos com os aparelhos eletroeletrônicos no século XX, principalmente com a TV. A tecnologia de transmissão e recepção de dados por meio de luz na faixa de frequência do infravermelho possibilitou uma grande redução de custo dos circuitos eletrônicos necessários. A empresa NEC desenvolveu um protocolo muito utilizado até hoje para realizar esta comunicação. Este protocolo, apresentado nas figuras abaixo, possibilita o envio de um Endereço e um Comando, ambos de 8 bits, de forma bastante confiável e com alta imunidade ao ruído ambiente, pois é baseada na transmissão do dado e do inverso do dado, permitindo a fácil detecção de erros de recepção/transmissão. Você dispõe de um sensor infravermelho que retorna '1' quando a luz é detectada e '0' quando não é detectada, mostrado na figura ao lado. Considere que transmissor transmita 20 pulsos em 562μs. Considere ainda que nem a CPI, nem a frequência de clock do Processador MIPS são conhecidas a priori. Sabe-se apenas que a CPI máxima de uma instrução é de 5 ciclos e que a frequência de clock é maior que 10MHz.



**a)(1.0)** Considere uma modificação na ISA MIPS na qual o registrador \$t9 seja dedicado aos dispositivos de IO. Nesta nova forma de interfaceamento, o bit 0 do registrador \$t9 corresponde à saída do sensor IrDA. Desenhe as modificações estruturais que devem ser realizadas internamente ao Banco de Registradores da CPU principal. Obs.: Os outros 31 bits deste registrador continuam operando normalmente.

**b)(4.0)** Escreva um procedimento em Assembly MIPS que ao ser chamado i) aguarde por uma transmissão recebida pela interface IrDA, retorne ii) no byte menos significativo do registrador \$v0 o dado de Address, iii) no byte menos significativo do registrador \$v1 o dado de Command. iv) Caso algum erro seja detectado, ambos registradores devem retornar o valor 0xFFFFFFFF. Explique claramente suas ideias e comente adequadamente seu código!

**2)(6.0)** Implemente as instruções abaixo nos processadores UNICICLO, MULTICICLO e PIPELINE nas folhas em anexo, desenhando as modificações internas necessárias na i) ULA e no ii) Banco de Registradores, nos iii) Caminhos de Dados e nos iv) sinais dos Blocos de Controle:

**a)(3.0) Multiply and Add**

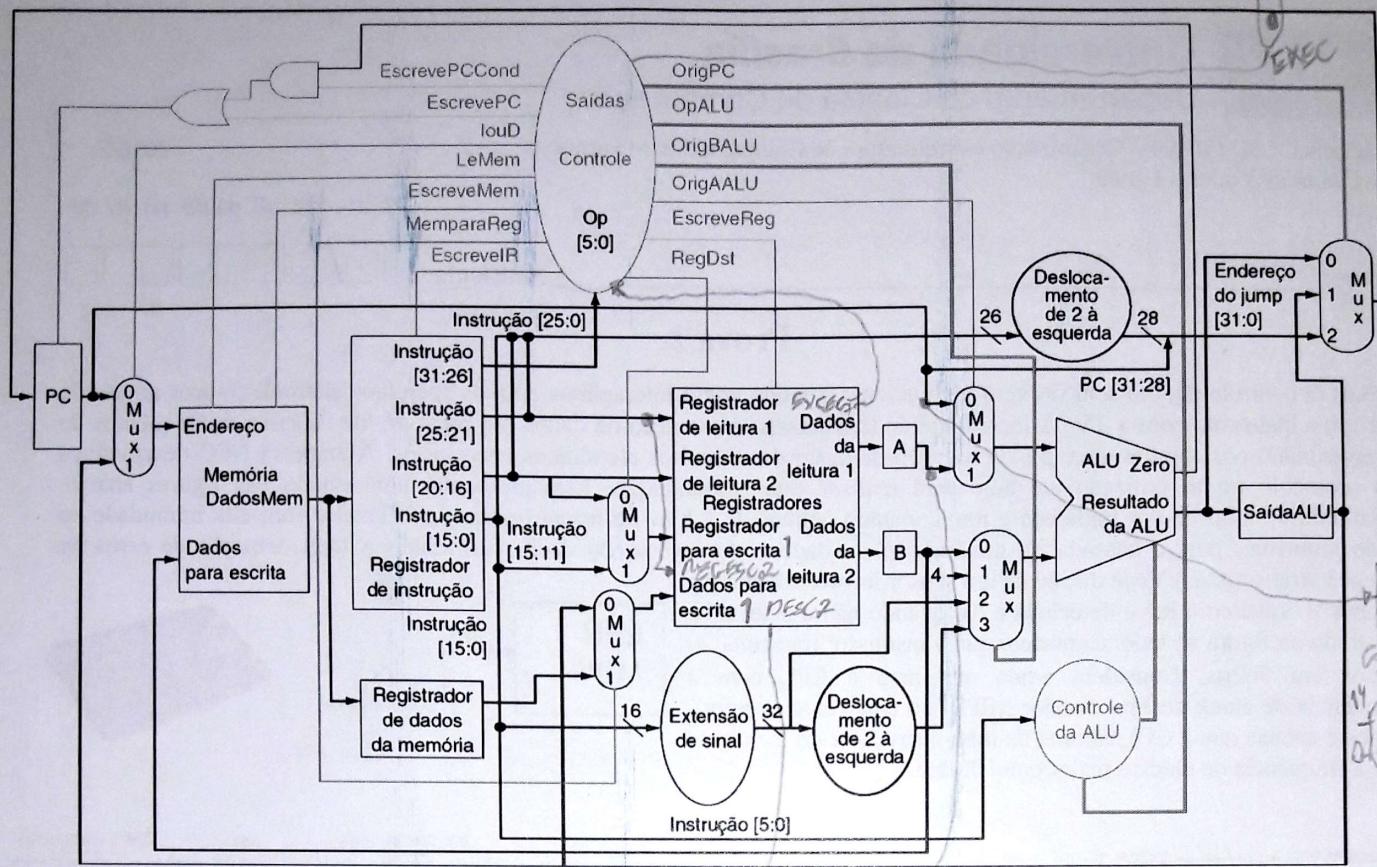
madd rs, rt # {Hi,Lo}={Hi,Lo}+R[rs]×R[rt] rd=0 Tipo-R Opcode:0x00 Funct=0x1C

**b)(3.0) Load Word and Increment**

lwinc rt, imm(rs) # R[rt]=Mem[R[rs]+imm] e R[rs]=R[rs]+4 Tipo-I Opcode:0x27

Obs: Considere que se rt=rs o processador gera uma exceção com código de causa 14 e a rotina de tratamento de exceção no endereço 0x80000000 é chamada.

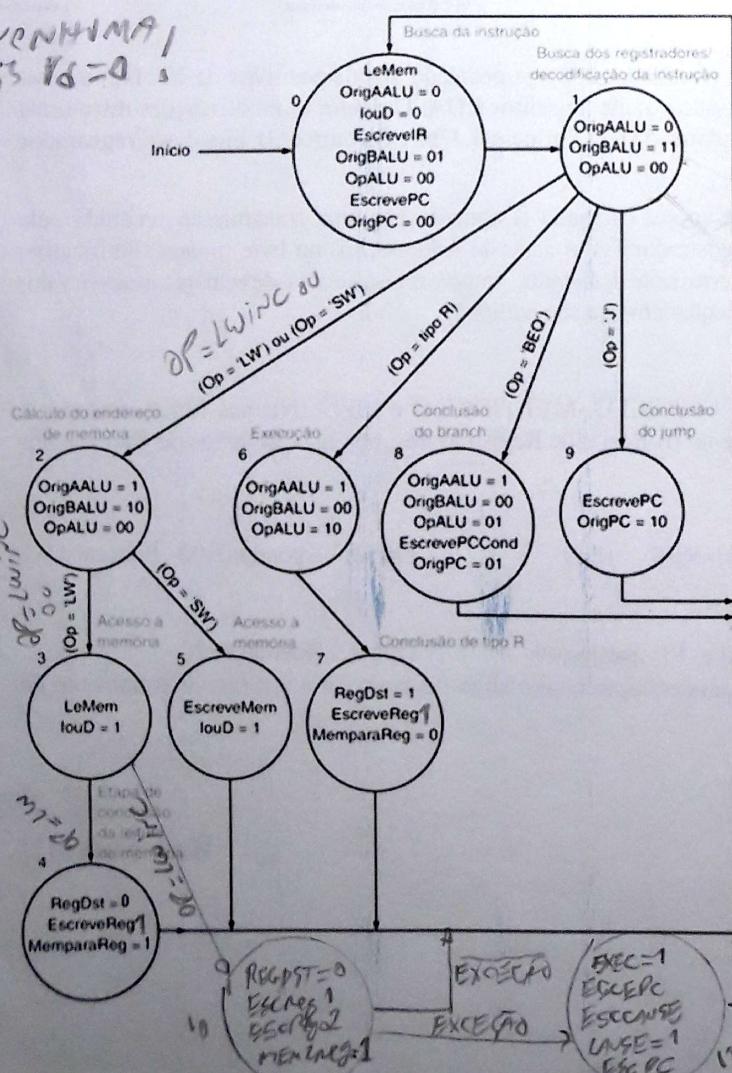
**Boa Sorte!!!**

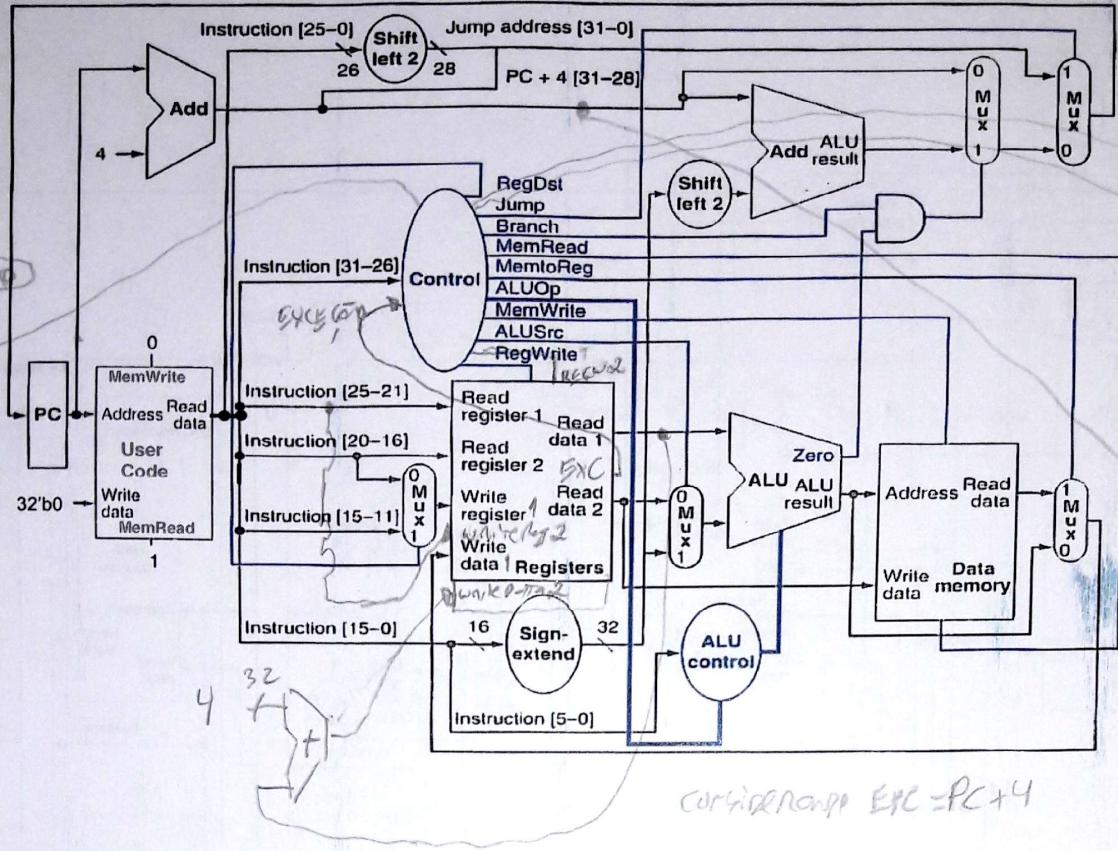


a) NENHUMA

o Esperando EPC = PC + 4

a) NENHUMA /  
Pois Rd = 0





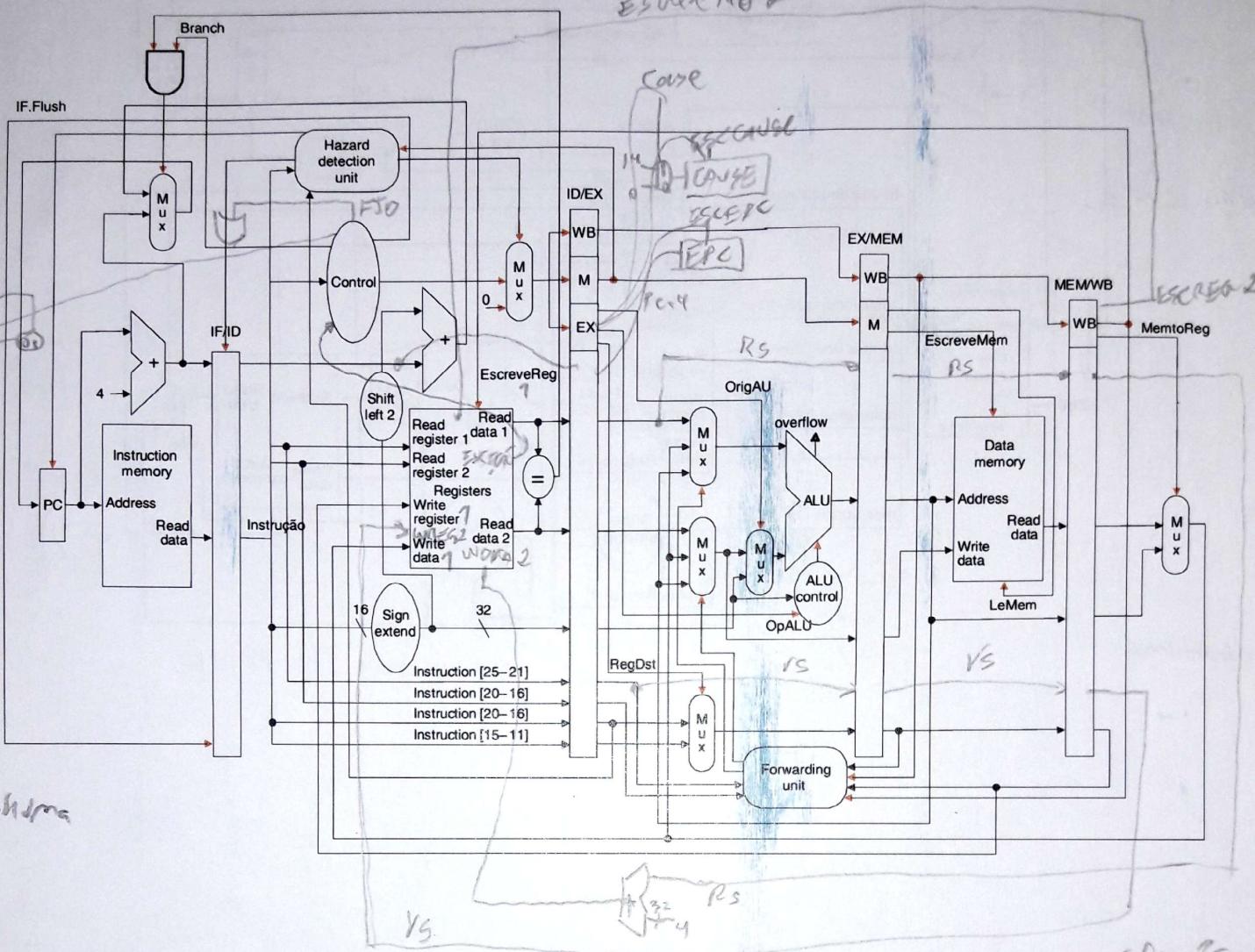
correspondence EPC = PC + 4

a) Nenhuma

a) Nenhuma Pois rd=0

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

Instrução	ALUOp1	ALUOp0	EXEC	ESCPIC	ESCAPE	Cause	
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	
lui	0	0	0	0	0	0	
EXECUÇÃO	X	X	1	1	1	1	



a) Maphema P0.3  $Vd = 0$

Instrução	Linhas de controle do estágio de cálculo de endereço/execução						
	RegDst	OpALU1	OpALU0	OrigALU	ESC EXC	F10 UNSE	Cause
Formato R	1	1	0	0	0	0	0
lw	0	0	0	1	0	0	0
sw	X	0	0	1	0	0	0
beq	X	0	1	0	0	0	0
lwinc	0	0	0	1	0	0	0
EXEC	X	X	X	X	1	1	1

EXC	F10
0	0
0	0
0	0
0	0
0	0
0	0
1	1

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	
lwinc	0	1	0	
EXEC	X	0	0	

Instrução	Linhas de controle do estágio de escrita do resultado			
	Escreve 1 Reg	Mem para Reg		
Formato R	1	0	0	
lw	1	0	1	
sw	0	0	X	
beq	0	0	X	
lwinc	1	1	1	
EXEC	0	0	X	

DAC-A

2016/2

2:8 PVA

GABAN, D

9) Nova forma de interlocu&ccedil;c&atilde;o

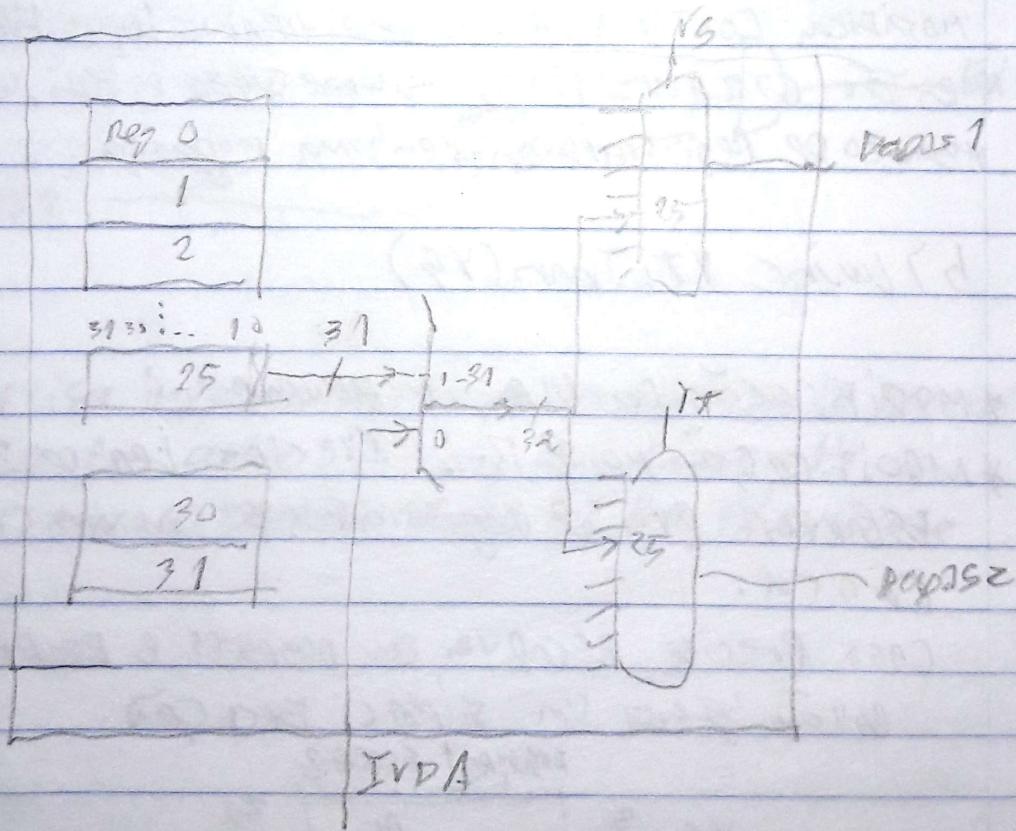
A PARTIR DO Banco de Registradores desenvolvidos em AULA,

Como o bit 0 do registrador 25 (S/F) só é afetado  
Para leitura: slave 23 porta 12

ESCRIBA:

film magnification

## LEITURGIE:



b) na folha em anexo \*VER NA FOLHA DE QUESTÃO,

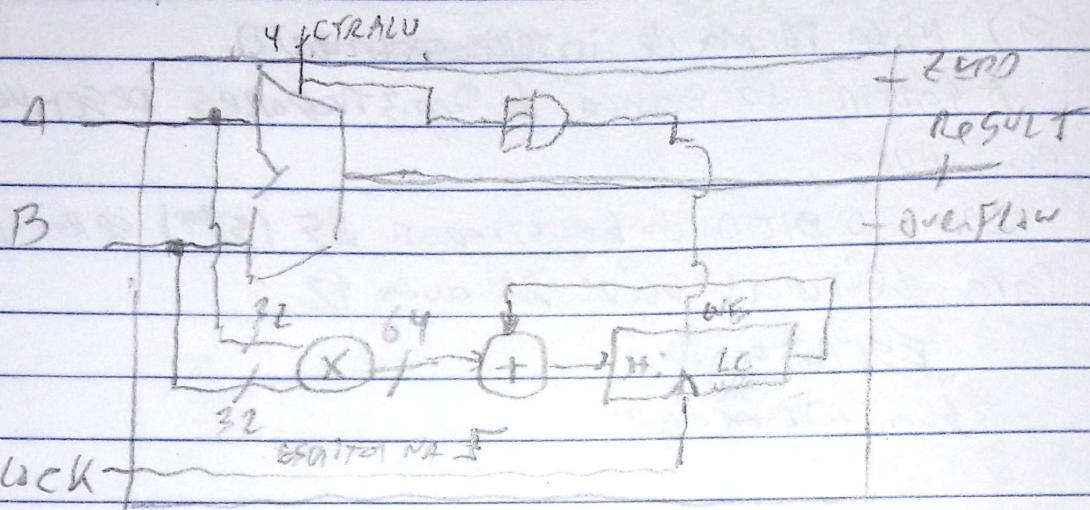
AVALIAÇÕES:

- sincronização (1,0) → ESTIMATIVO
- reading ruler (0,5) (CONTADOR)
- space (0,5)
- escrita em área euso (1,0)
- verificação de ERRO (1,0)

2)

a) madd RS, VT

\* MODIFICAÇÃO DA ULA



SÓ DEVE MUDAR O MODO DE FUNCIONAMENTO PARA  $FUNCT = 0x1C$

MODIFICAÇÃO CONTROLE DA ULA PARA  $OPCODE = 10_2$  E  $FUNCT = 0x1C$

ENTÃO  $CTR ALU = 1111_2 \rightarrow$  write enable da ULA.

\* Banco de Registradores: Nenhuma mudança

b) twinc VT, Imm(VS)

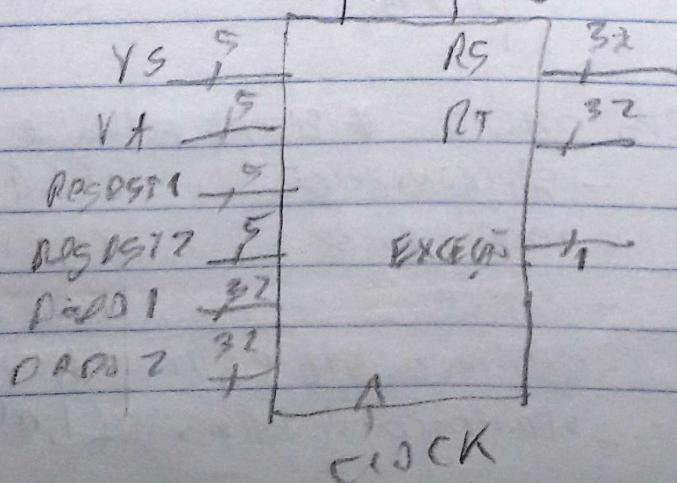
\* MODIFICAÇÃO DA ULA: NENHUMA

\* MODIFICAÇÃO NO B.R.: Precisa ter o 2 e

ESCREVER EM 2 REGISTRADES EM UM CICLO DE CLOCK.

CASO PRECISE ESCREVER EM REGIST1 E REGIST2 E SEUS VALORES FORAM IGUAIS  
ENTÃO GERA UM SINAL EXCEÇÃO.

REGIST1 E REGIST2



slide 23:

ESCRITA

Jm SECTOR

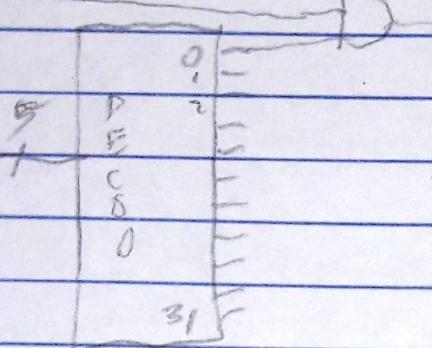
P/Copy Reg

CLOCK

DADO 1 → <sup>32</sup>

ESCRV REG1

regPSR1



0<sup>32</sup>

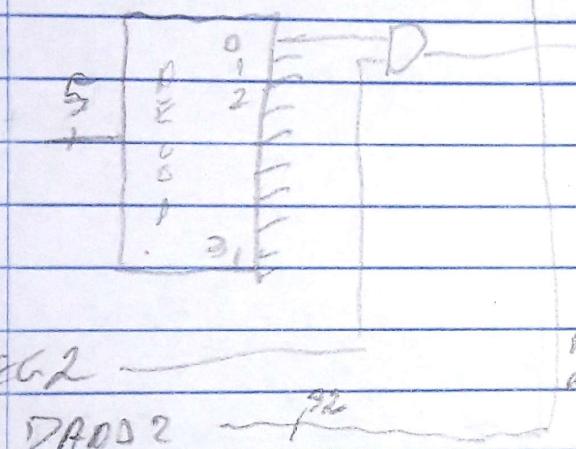
↓

0

ESCRV REG2

DADO 2 → <sup>32</sup>

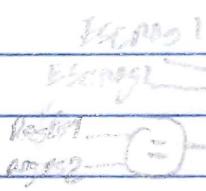
regPSR2



0<sup>32</sup>

0

31



EXCEÇÃO

\* Precisa implementar TODO o tratamento de exceções

visto em aula. Apesar de cause = 14.

C o Sinal EXCEÇÃO vindo do PR P/ controle

```
1: .data
2:
3: .text
4: MAIN:    jal IRDA
5: li $v0,10
6: syscall
7:
8: IRDA:    addi $sp,$sp,-20 # salva registradores na pilha
9: sw $ra,0($sp)
10: sw $s0,4($sp)
11: sw $s1,8($sp)
12: sw $s2,12($sp)
13: sw $s3,16($sp)
14:
15: jal SYNC    # mede o tempo de 1/2 ciclo (tempo =1) e posiciona no primeiro dado valido
16: move $a0,$v0    # salva rm $a0
17:
18: jal LEDADO  # le address
19: move $s0,$v0
20: jal LEDADO  # le ~address
21: move $s1,$v0
22: jal LEDADO  # le comando
23: move $s2,$v0
24: jal LEDADO  # le ~comando
25: move $s3,$v0
26:
27: nor $t4,$zero,$s0
28: bne $t4,$s1, ERRO
29: nor $t4,$zero,$s2
30: bne $t4,$s3, ERRO
31: move $v0,$s0
32: move $v1,$s2
33:
34: FIM:    sw $ra,0($sp)
35: sw $s0,4($sp)
36: sw $s1,8($sp)
37: sw $s2,12($sp)
38: sw $s3,16($sp)
39: addi $sp,$sp,20
40: jr $ra
41:
42: ERRO:   la $v0,0xFFFFFFFF
43: la $v1,0xFFFFFFFF
44: j FIM
45:
46: # Sincronizacao inicial
47: SYNC:   li $v0,0    #contador
48: LOOP1:  bne $t9,$zero,LOOP1 #aguarda inicio em zero
49:
50: # conta o tempo em 1, isto é, quanto devemos contar $v0 para o tempo de 562u/(2*20) s
```

```
gundos = 14.05us
51: LOOP2: beq $t9,$zero,SAI1
52:      addi $v0,$v0,1
53:      j LOOP2
54: SAI1:   # precisamos esperar (9m+4.5m)/14.05u = 960.8 ciclos de $v0 LOOPS para o inici
o dos dados
55:      li $t0,960
56:      mult $v0,$t0
57:      mflo $t0
58:      li $t1,0
59: LOOP3: beq $t1,$t0,FIM2
60:      addi $t1,$t1,1
61:      j LOOP3
62: FIM2:   jr $ra
63:
64:
65:
66: LEDADO: # Para saber se é 0 ou 1
67:      li $v0,0      # dado a ser lido
68:      li $t4,0      # contador do bit
69: LOOPBYTE: beq $t4,8,FIMBYTE
70:
71:      # 1) espera-se 562u+562u+56u/2 isto é, (1405us)/14.05u = 100 ciclos de $a0
72:      li $t0,100
73:      mult $a0,$t0
74:      mflo $t0
75:      li $t1,0
76: LOOP4: beq $t1,$t0,SAI2
77:      addi $t1,$t1,1
78:      j LOOP4
79:
80:      # 2) mede-se durante $a0*2 ciclos se houve bit 1
81: SAI2:   sll $t0,$a0,1  #x2
82:      li $t1,0
83:      li $t2,0
84: LOOP5: beq $t1,$t0,SAI3
85:      or $t2,$t2,$t9      # basta ter um pulso no período para detectar bit logico 0 (ZE
RO)
86:      addi $t1,$t1,1
87:      j LOOP5
88:
89: SAI3:   xori $t2,$t2,0x0001  # para inverter o bit recebido
90:      sllv $t2,$t2,$t4      # Desloca o bit recebido para a posicao adequada
91:      or $v0,$v0,$t2      # insere o bit no byte de dados
92:      addi $t4,$t4,1      # incrementa o contador do bit recebido
93:
94:      # 3) espera-se 562u+56u/2 isto é, (843us)/14.05u = 60 ciclos de $a0
95:      li $t0,60
96:      mult $a0,$t0
97:      mflo $t0
98:      li $t1,
```

```
99: LOOP6:    beq $t1,$t0,LOOPBYTE
100:      addi $t1,$t1,1
101:      j LOOP6
102:
103: FIMBYTE:   jr $r
```