Ministério da Educação
UNIVERSIDADE DE BRASÍLIA
Instituto de Ciências Exatas
Dep. Ciências da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2006/2
Prof.: Marcus Vinicius Lamar

Nome:_____ Matrícula: _____

Prova 2

1) (2.0) Na ISA do MIPS muitas das funções disponibilizadas são pseudo-instrução que o montador disponibiliza, traduzindo-as para um conjunto de instruções reais no momento da criação do código de máquina. Respeitando a convenção do uso dos registradores, implemente as seguintes pseudo-instruções.

a) (1.0) abs rt, rs # rt =abs(rs)
b) (1.0) addi.s ft, fs, imm # ft = fs+imm (imm número inteiro com sinal de 16 bits)

Dicas: instrução cvt.s.w fd, fs : converte um inteiro em fs em um ponto flutuante (precisão simples) em fd
Instrução mtc1 rs, ft : copia o conteúdo de rs para ft

2) (2.0) Defina *overflow*, apresente os requisitos do sistema de detecção (tabela verdade) e implemente o circuito lógico necessário a disponibilização do sinal *overflow* na ULA projetada em aula.

3) (2.0) Dado que a ULA do MIPS implementada em aula não possui a função XOR. Redesenhe os diagramas em blocos e defina os controles necessários para uma possível implementação, sem aumentar o número de bits de controle (4).
Dica: Não é necessário manter a compatibilidade de código da operação com a instrução real do MIPS;

4) (2.0) Apresente as modificações necessárias no bloco operativo e no bloco controlador de modo a implementar a detecção e tratamento da exceção *operação não reconhecida*. Modifique a implementação da Unidade de Controle Multiciclo por Microprogramação de modo a incorporar esse controle de exceção.

5) (3.0) Considerando apenas os seguintes tempos de atraso dos blocos operativos de uma CPU MIPS:

Operação Lógica da ULA: 100ps
Operação Aritmética da ULA: 150ps
Leitura do Banco de Registradores: 50ps
Escrita no Banco de Registradores: 70ps
Leitura da memória: 200ps
Escrita na memória: 250ps

Considere o seguinte trecho de programa:

```
        ...
        lw $t0, 100($t1)
        lw $t2, 104($t1)
        add $t3, $t0, $t2
        sub $t4, $t0, $t2
        beq $t3, $t4, LABEL1    #Considere a condição verdadeira
        sw $t3, 100($t1)
        j LABEL2
LABEL1: sw $t4, 100($t1)
        j LABEL2
        ....
```

a) (0.5) Qual o tempo de execução para uma implementação uniciclo do MIPS? Qual a maior frequência de clock utilizável?
b) (0.5) Qual o tempo de execução para uma implementação multiciclo do MIPS? Qual a maior frequência de clock utilizável?
c) (1.0) Para as duas implementações, em quais ciclos de clock a operação de adição é efetivamente calculada?
d) (1.0) Reescreva o código otimizando-o.

BOA SORTE!!!

2ª Prova - Gabarito 2006/2

TURMA A / B

1) a) abs rs, rt ⇒

```
SLT $at, rt, $ZERO      # $t = 1 rt < 0
beq $at, $ZERO, Positivo
SUB rt, $ZERO, rt
Positivo: OR rs, rt, $ZERO   # MOV rs, rt
```

b) addi.s Ft, Fs, imm ⇒

```
ORi $at, $ZERO, imm     # Li $at, imm
mtc1 $at, Ft
cvt.s.w Ft, Ft
add.s Ft, Fs, Ft
```

a) neg rt, rs ⇒

```
SUB rt, $ZERO, rs
```

ou

```
NOR $at, rs, rs
addi rt, $at, 1
```

b) multi Ft, Fs, imm ⇒

```
ORi $at, $ZERO, imm     # Li $at, imm
mtc1 $at, Ft
cvt.s.w Ft, Ft
mul.s Ft, Fs, Ft
```

2) overflow: - situações

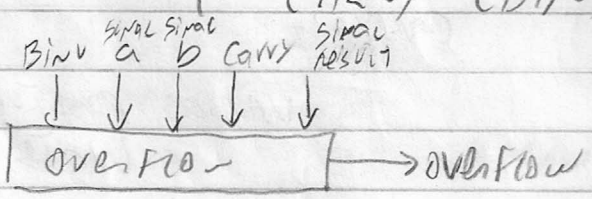
(A ≥ 0) + (B ≥ 0) = R < 0 ①

(A < 0) + (B < 0) = R ≥ 0 ②

(A ≥ 0) - (B < 0) = R < 0 ③

(A < 0) - (B ≥ 0) = R ≥ 0 ④

⑤ soma ou sub → 32 bit → carry out = 1

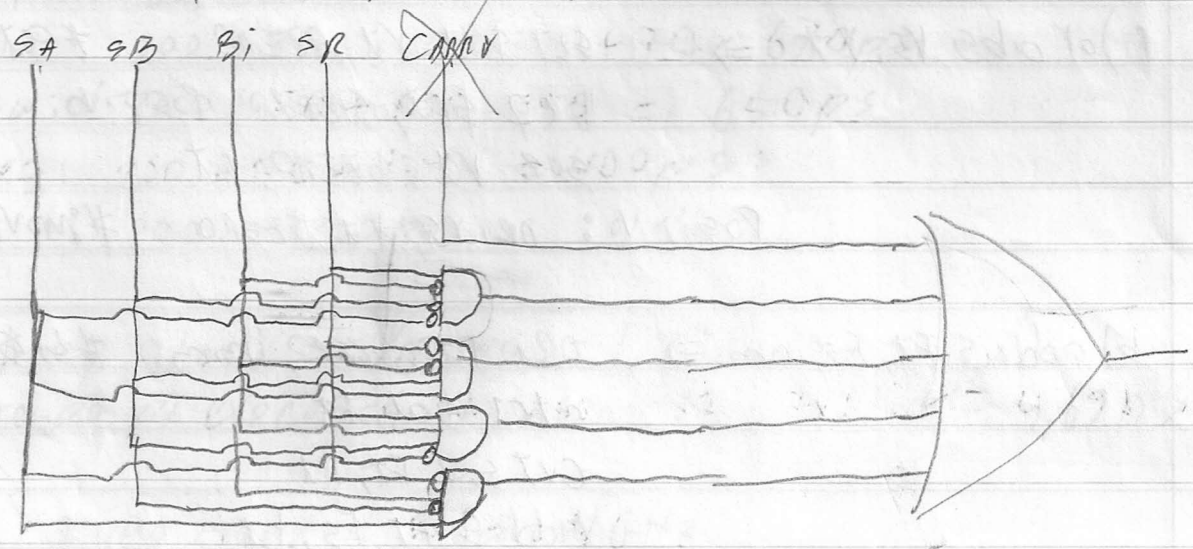


5 casos:

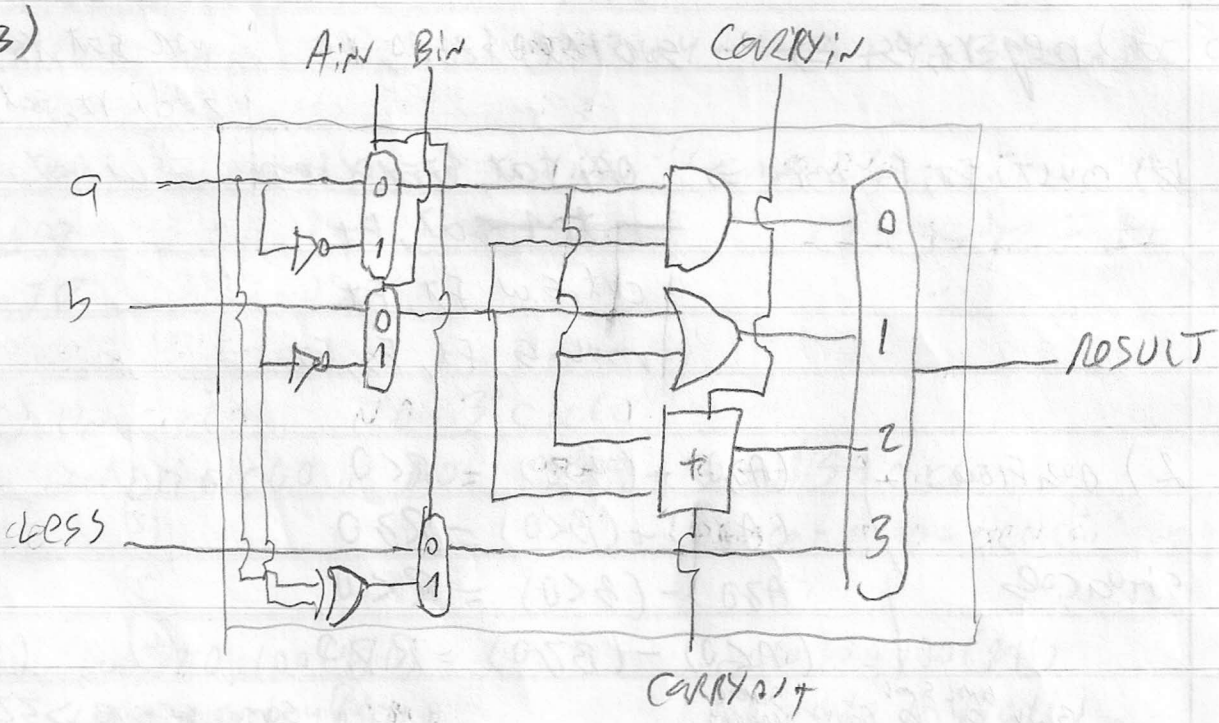
SA Sinal a	SB Sinal b	Bi BInv	CARRY	SR Sinal resultado	OVERFLOW
0	0	0	X	1	1
1	1	0	X	0	1
0	1	1	X	1	1
1	0	1	X	0	1
[X	X	X	1	X	1]

Projeto

OVERFLOW = ~~CARRY~~ + $\overline{SA} \cdot \overline{SB} \cdot \overline{Bi} \cdot SR + SA \cdot SB \cdot \overline{Bi} \cdot \overline{SR} +$
$+ \overline{SA} \cdot SB \cdot Bi \cdot SR + SA \cdot \overline{SB} \cdot Bi \cdot \overline{SR}$



3)



Operações implementadas

AND	0000
OR	0001
SOMA	0010
SUB	0110
SLT	0111
NOR	1100

Várias possíveis soluções

→ Bloquear o CARRY in

→ escolher 1111 → XOR

Logo BIT AIN → seleciona saída 3

4) OVERFLOW e instrução não definida
ver livro: Fig. 5.39 e 5.40

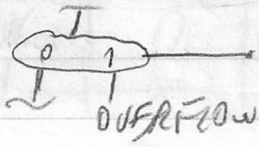
Mudança na μinstrução
→ incluir sinais OVERFLOW na ULA
→ incluir sinais causaInt, escreveCausa, escreveEPC

ALUCtrl	SRC1	SRC2	RegCtrl	MemD	PCWrite	Exceção	Seq

Exceção	OVERFLOW	causaInt = 1 EscreveCausa EscreveEPC
	Instrução não reconhecida	causaInt = 0 EscreveCausa EscreveEPC
PCWriteCnd	Exceção	OrIPC = 11 EscrevePC

LABEL	ALUCtrl	SRC1	SRC2	RegCtrl	MemD	PCWrite	Exceção	Seq
FETCH	Add	PC	4		Read PC	ALU		Seq
	Add	PC	EXTSHFT	Read				Dispatch 1
MEM1	Add	A	EXTEND					Dispatch 2
LW2					Read ALU			Seq
				WriteMDR				Fetch
SW2					Write ALU			Fetch
RFORM1	FUNCODE	A	B					Seq
				WriteALU				FETCH
OVERFLOW	SUB	PC	4			EXCEÇÃO	OVERFLOW	FETCH
NOT RECOG1	SUB	PC	4			EXCEÇÃO	INSTRUÇÃO NÃO RECOG.	FETCH
BEQ1	SUB	A	B			ALUCOND.		FETCH
JUMP1						JUMP ADD		FETCH

DISPATCH 1 → qualquer outro endereço deve apontar para "NOT RECOG"
Atenção ao controle de próximo estado



OVERFLOW se houver Overflow
Vai p/ OVERFLOW 3

Caso contrário segue o seq.

5)

a) Unicicl0:

Tipo-R: 200 + 50 + 150 + 70 = 470ps

LW: 200 + 50 + 150 + 200 + 70 = 670ps

SW: 200 + 50 + 150 + 250 = 650ps

Beq: 200 + 50 + 150 = 400ps

J: 200 = 200ps

Freq. clock = $\frac{1}{670ps}$ = 1,492 GHz

Tempo p/ execução 7 instruções: 7 x 670p = 4,69ns

b) Freq. clock = $\frac{1}{250p}$ = 4 GHz

Tempo execução:

Tipo-R: 4 ciclos

LW: 5 ciclos

SW: 4 ciclos

Beq: 3 ciclos

JP: 3 ciclos

Total:

5 + 5 + 4 + 4 + 3 + 4 + 3 = 28 ciclos

Logo: Tempo execução:

28 x 250p = 7ns

c) Unicicl0: NO 3º ciclo

MULTIciclO: NO 5 + 5 + 3 = 13º ciclo

(Busca + Decod + execução)

d)

```
lw $t0, 100($t1)
lw $t2, 104($t1)
ADD $t3, $t0, $t2
sw $t3, 100($t1)
mov $t4, $t3   # OPCIONAL
j LABEL2
```

```
LW $t0, 100($t1)
Lw $t2, 104($t1)
beq $t2, $zero, LABEL2
add $t3, $t0, $t2
sw $t3, 100($t1)
j LABEL2
```