Lista de Exercício 2 – OC II

Gabarito

Exercício 1

Considere as seguintes características para erro de previsão. Os desvio são 20% do processamento. O CPI sem parada é igual a 1,5.

	Previsão Não-Tomado	Previsão Tomado
Desvio Tomado	4	1
Desvio Não-Tomado	0	5

(a) Considerando que os desvios são tomados 75% do tempo. Determine o CPI desta máquina usando um sistema de previsão estático tomado. Faça o mesmo para o não-tomado.

Tomado: 1,5 + 0,2 x (0,75 x 1 + 0,25 x 5) = 1,5 + 0,4 = 1,9
Não-Tomado: 1,5 + 0,2 x (0,75 x 4 + 0,25 x 0) = 1,5 + 0,6 = 2,1

(b) Considere o uso de uma branch history table de 1 bit. Assuma que o previsor é inicializado no estado T. Dada a sequência de desvios TTTTNNTNTNTTTTNTNTNT, qual é a taxa de acerto e CPI correspondente?

D: TTTTNNTNTNTTTTNTNTNT
P: TTTTTNNTNTNTTTTNTNTN – acertos 8 / 20

Previsão T / Desvio T = 7
Previsão T / Desvio N = 6
Previsão N / Desvio T = 6
Previsão N / Desvio N = 1

CPI: 1,5 + 0,2 x (7/20 x 1 + 6/20 x 5 + 6/20 x 4 + 1/20 x 0) = 1,5 + 0,61 = 2,11

(c) Repita o item acima para o caso de uma branch history table de 2 bits. Assuma que o previsor é inicializado no estado PT. Os estados são:

PT – Pouco Tomado
MT – Muito Tomado
PN – Pouco Não-Tomado
MN – Muito Não-Tomado

D: TTTTNNTNTNTTTTNTNTNT
P: TTTTTTNNNNNNTTTTTTTT – acertos 11 / 20

Previsão T / Desvio T = 9
Previsão T / Desvio N = 5
Previsão N / Desvio T = 4
Previsão N / Desvio N = 2

CPI: 1,5 + 0,2 x (9/20 x 1 + 5/20 x 5 + 4/20 x 4 + 2/20 x 0) = 1,5 + 0,5 = 2,0

Exercício 2

Para este problema, assuma um processador VLIW com três unidades de inteiros (X, Y, Z), uma unidade de multiplicação (M) e duas unidades de *load* e *store* (LS0, LS1). Instruções de ALU tem uma latência de 1, multiplicação tem uma latência de 5 e *loads* tem uma latência de 2. Um desvio (*branch*) pode ser executado por ciclo e executa no pipeline Z. O seguinte código foi gerado assumindo uma técnica de escalonamento conhecida como EQ.

	X	Y	Z	M	LS0	LS1
1	ADDI R9, R0, 9	ADDI R10, R0, 10				
2	ADDI R6, R0, 6	ADDI R8, R0, 8	ADDI R5, R0, 5			
3					LW R6, 0(R7)	LW R8, 4(R7)
4	ADDI R12, R6, 1	ADDI R13 R8, 2				
5				MUL R7, R6, R9		
6				MUL R5, R8, R10		
7					LW R14, 8(R7)	
8	ADD R15, R16, R17					
9	ADD R14, R14, R5					
10	SUB R19, R18, R22					
11	ADD R5, R7, R5					

R0 tem valor zero!
Assuma que os endereços 0(R7), 4(R7) e 8(R7) contêm os valores 0, 10 e 1, respectivamente.

(a) Quais os valores de R12, R13 e R14 após a execução do código?

7, 10 e 6

(b) Sem alterar os nomes dos registradores, reescalone o código acima de forma a melhorar o desempenho sem impactar na lógica correta.

	X	Y	Z	M	LS0	LS1
1	ADDI R6, R0, 6	ADDI R8, R0, 8	ADDI R5, R0, 5		LW R14, 8(R7)	
2	ADDI R12, R6, 1	ADDI R13 R8, 2			LW R6, 0(R7)	LW R8, 4(R7)
3	ADD R14, R14, R5	ADDI R9, R0, 9	ADDI R10, R0, 10			
4				MUL R5, R8, R10		
5				MUL R7, R6, R9		
6	ADD R15, R16, R17					
7	SUB R19, R18, R22					
8						
9						
10	ADD R5, R7, R5					
11						

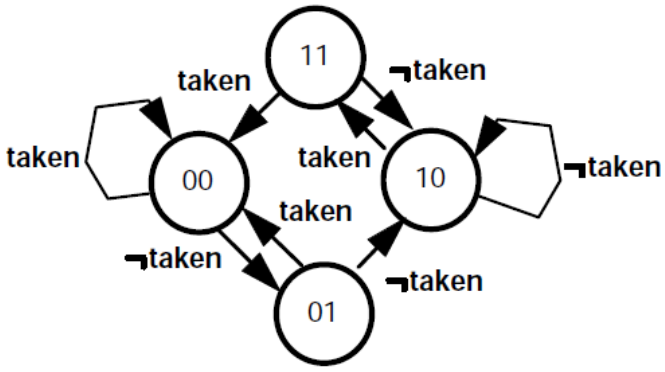
Exercício 3

Este problema investigará os efeito da **Previsão de Desvios (Branch Prediction)**. Ao longo do problema vamos considerar o seguinte código:

```
loop:
        LW     R4, 0(R3)
        ADDI   R3, R3, 4
        SUBI   R1, R1, 1
b1:
        BEQZ   R4, b2
        ADDI   R2, R2, 1
b2:
        BNEZ   R1, loop
```

Assuma que o valor inicial de R1 é n (n > 0)
Assuma que o valor inicial de R2 é 0 (R2 guarda o resultado do programa)
Assuma que o valor inicial de R3 é p (um ponteiro para o início de um array de inteiros de 32 bits)

Todas as previsões de desvios neste problema vai ser baseada em um previsor de 2 bits , como mostrado abaixo:



O estado **1X é considerado como não-tomado**. Já o estado *0X é considerado como tomado*.
Assumam que b1 e b2 não possuem conflito no BHT.

(a) O que o programa calcula? Ou seja, qual o valor contido em R2 quando o loop terminar?

O número de valores diferentes de zero

(b) Vamos investigar o funcionamento do previsor proposto. Assuma que as entradas do programa são n = 8 e p[0] = 1, p[1] = 0, p[2] = 1, p[3] = 0,... etc.; isto é, o array exibe elementos alternados de 1s e 0s. Preencha o restante da tabela abaixo. Qual é o número de predições incorretas?

A tabela contém uma entrada para cada desvio (b1 e b2) que é executado. Os bits do Previsor de Desvio (PD) na tabela são bits da BHT (Branch History Table). Para cada desvio, verifique os bits de PD correspondente para realizar a previsão. Em seguida, atualize os bits de PD da entrada.

do do Sistema		Previsor de Desvio (PD)		Comportamento do Desvio		
R3	R4	b1 bits	b2 bits	Previsto	Ocorrido	
4	1	10	10	N	N	
4	1	10	10	N	T	x
8	0			N	T	x
8	0			N	T	x
12	1			N	N	
12	1			T	T	
16	0			N	T	x
16	0			T	T	
20	1			N	N	
20	1			T	T	
24	0			N	T	x
24	0			T	T	
28	1			N	N	
28	1			T	T	
32	0			N	T	x
32	0			T	N	x

7

(c) Agora adicionamos um **bit de histórico global (desvios correlacionados)**. Preencha a tabela abaixo e novamente informe o número total de predições incorretas. Considere as mesmas condições do item b.

O bit histórico igual a 0 corresponde a não tomado e 1 a tomado.

Estado do Sistema				Previsor de Desvio (PD)				Comportamento do Desvio	
				b1 bits		b2 bits			
PC	R3	R4	Bit História	0	1	0	1	Previsto	Ocorrido
b1	4	1	1	10	10	10	10	N	N
b2	4	1	0	10	10	10	10	N	T
b1								N	T
b2								N	T
b1								N	N
b2								N	T
b1								N	T
b2								N	T
b1								N	N
b2								T	T
b1								N	T
b2								T	T
b1								N	N
b2								T	T
b1								N	T
b2								T	N

9

(d) Refaça o item c, mas agora **considere um segundo bit de histórico global.** Qual o número de predições incorretas?

Estado do Sistema				Previsor de Desvio (PD)								Comportamento do Desvio	
				b1 bits				b2 bits					
PC	R3	R4	Bit História	0 0	0 1	1 0	1 1	0 0	0 1	1 0	1 1	Previsto	Ocorrido
b1	4	1	1 1	10	10	10	10	10	10	10	10	N	N
b2	4	1	0 1	10	10	10	10	10	10	10	10	N	T
b1												N	T
b2												N	T
b1												N	N
b2												N	T
b1												N	T
b2												N	T
b1												N	N
b2												T	T
b1												T	T
b2												T	T
b1												N	N
b2												T	T
b1												T	T
b2												T	N

7

(e) Compare os resultados do itens b, c e d. Onde ocorre a maior parte das previsões incorretas em cada caso (no início, periodicamente, no fim, etc)? O que isto diz sobre a previsão com bits de histórico global? O que ocorreria para um valor de n maior? Explique resumidamente.

Para n grande, o último previsor seria melhor.

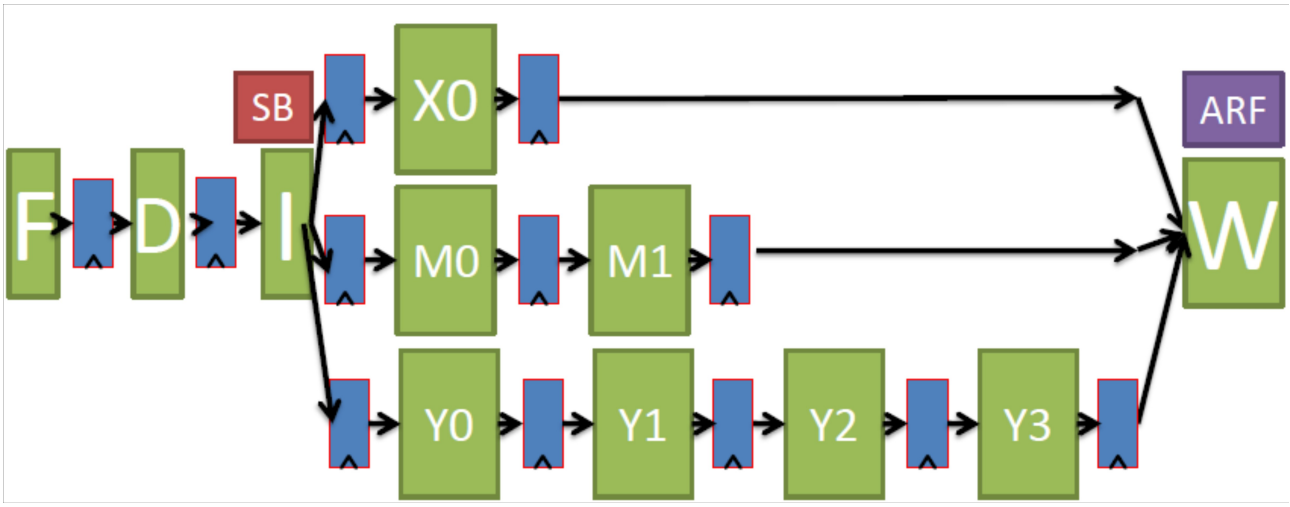
(f) As entradas utilizadas neste problema são muito regulares. O que você esperaria que ocorresse caso as entradas fossem aleatórias (igual probabilidade do elemento do array ser 0 ou 1)? Dos 3 previsores que vimos neste problema, qual deles seria o melhor para este tipo de entradas? O resultado é o mesmo para n pequeno e grande?

Sem padrão. Bit de histórico não ajuda. Não depende de n.

Exercício 4

Considere o processador I2O2 visto em sala e apresentado abaixo. Além disso, resolva todos os itens da questão considerando o seguinte código:

Processador I2O2



Código:

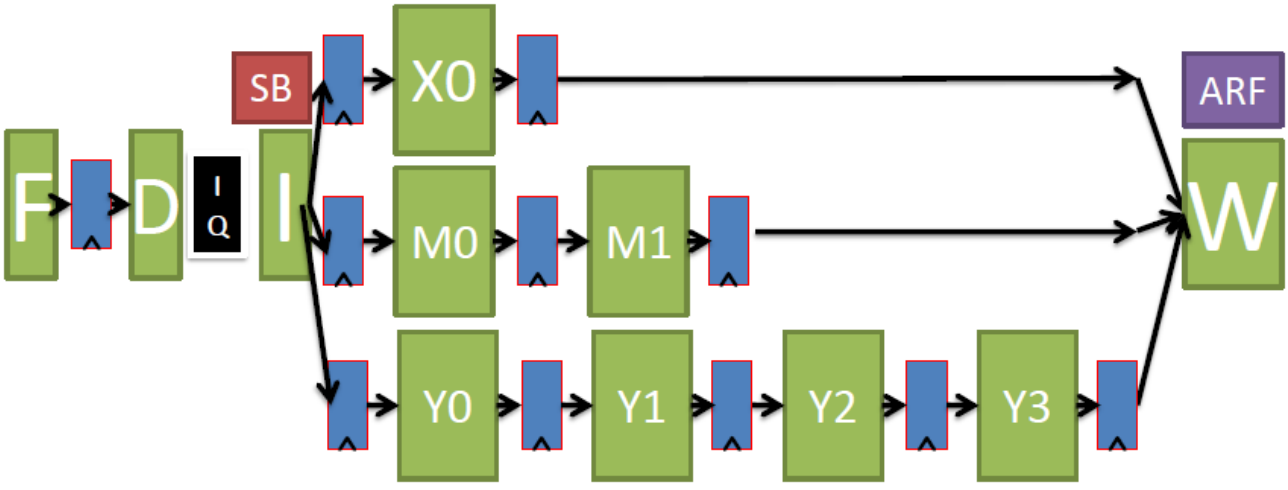
```
1:    MUL   R1, R2, R3
2:    ADD   R4, R2, R1
3:    MUL   R2, R7, R8
4:    LW    R10, 0(R12)
5:    MUL   R4, R10, R1
6:    SW    R4, 0(R2)
7:    ADD   R5, R6, R7
8:    ADD   R4, R3, R1
9:    BNE   R4, R0
10:   ADDI  R3, R1, 10
```

(a) Mostre o diagrama do pipeline deste trecho de código, considerando que o desvio da instrução 9 é não-tomado e a instrução 10 pode ser disparada em seguida. Encaminhamentos são permitidos.

(b) Mostre o estado do Scoreboard quando a instrução 3 está no estágio I do pipeline. Encaminhamentos são permitidos.

	P	F	4	3	2	1	0
R1	1	Y					1
R2							
R3							
R4	1	X				1	

(c) Considere agora o processador IO3 visto em aula e apresentado abaixo. Como fica o novo diagrama do pipeline? Apenas uma instrução pode ser disparada por vez. Quando a instrução está no IQ esperando o disparo, considere como estágio i.



(d) Se ocorrer um problema de desalinhamento de memória no estágio M1 da instrução 6, quais problemas para o tratamento da interrupção.

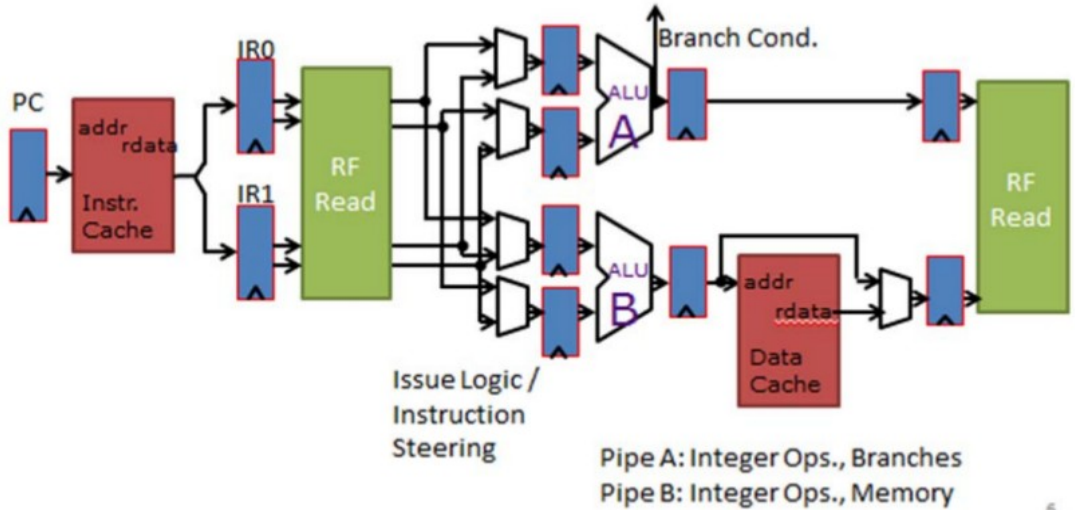
(f) Como resolver este problema sem reescalonar o código e sem perder desempenho? Mostre uma imagem que ilustre a sua alternativa e descreva os detalhes.

Exercício 5

(a) Mostre o diagrama de pipeline para o programa abaixo considerando o pipeline de 5 estágios do MIPS para instruções inteiras. A instrução MUL é executada por uma unidade funcional exclusiva por 4 estágios. Considere apenas uma ALU e uma unidade funcional de multiplicação. Considere todos os tipos de encaminhamento (fowarding).

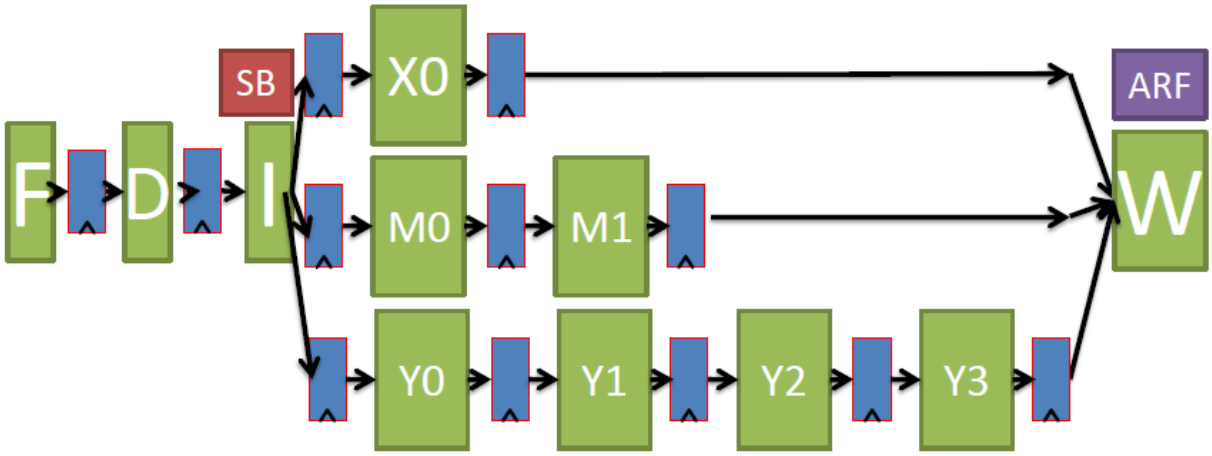
```
0: ADD      R15, R2, R3
1: SUB      R1, R12, R16
2: ADDIU    R11, R10, 1
3: MUL      R5, R1, R4
4: MUL      R7, R5, R6
5: ADDIU    R18, R11, 1
6: ADDIU    R14, R18, 1
7: ADDIU    R13, R18, 2
8: SW       R5, 0(R14)
9: SW       R7, 0(R14)
```

(b) Mostre o diagrama do pipeline do código abaixo sendo executado no **processador superescalar de 2-vias em ordem** apresentado abaixo. Assuma que desvios só são executados no pipeline A e load/store só são executados no pipeline B. Considere encaminhamento quando possível e nenhum problema de alinhamento.




```
ADD      R5, R6, R7
SUB      R6, R7, R8
LW       R10, 0(R6)
ADDIU    R12, R13, 1
LW       R15, 4(R6)
LW       R15, 4(R15)
ADD      R6, R9, R10
ADDIU    R8, R10, R11
```

(c) Considere o processador I2O2 visto em sala e apresentado abaixo. Mostre o estado do Scoreboard a cada ciclo da execução do código abaixo. Encaminhamentos são permitidos.



Processador I2O2

Código:

```
MUL   R1, R2, R3
ADD   R4, R2, R1
LW    R3, 20(R4)
ADD   R4, R3, R2
```

	P	F	4	3	2	1	0
R1	1	Y	1				
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1	1	Y		1			
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1	1	Y			1		
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1	1	Y				1	
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1	1	Y					1
R2							
R3							
R4	1	X				1	

	P	F	4	3	2	1	0
R1	0	Y					1
R2							
R3	1	M			1		
R4	1	X					1

	P	F	4	3	2	1	0
R1	0	Y					1
R2							
R3	1	M				1	
R4	0	X					1

	P	F	4	3	2	1	0
R1	0	Y					1
R2							
R3	1	M				1	
R4	0	X					1

	P	F	4	3	2	1	0
R1	0	Y					1
R2							
R3	1	M					1
R4	1	X				1	
	P	F	4	3	2	1	0
R1	0	Y					1
R2							
R3	0	M					1
R4	1	X					1

	P	F	4	3	2	1	0
R1	0	Y					1
R2							
R3	0	M					1
R4	0	X					1

	P	F	4	3	2	1	0
R1							
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1							
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1							
R2							
R3							
R4							
	P	F	4	3	2	1	0

R1							
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1							
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1							
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1							
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1							
R2							
R3							
R4							

	P	F	4	3	2	1	0
R1							
R2							
R3							
R4							