Universidade de São Paulo
Escola de Artes, Ciências e Humanidades
Docente: Prof. Dr. Clodoaldo A M Lima.
Discente: _____ No. USP:_____

1ª Questão) (0,5 ponto) Relaciona a coluna da esquerda com a coluna da direita

(I) Multicore	(III)	Múltiplos pipelines que operam em paralelo
(II) Superpipeline	(IV)	Execução de instruções fora de ordem em um pipeline.
(III) Superescalar	(II)	Pipelines com grande número de estágios.
(IV) Pipeline dinâmico	(V)	Múltiplos processadores compartilhando um espaço de endereços.
(V) Multiprocessadores	(I)	Múltiplos processadores em um único encapsulamento

2ª Questão) (1,5 Ponto) Usando o sistema de previsão local de desvio de 2 bits, mostrado na Figura1, um certo loop é executado duas vezes:

a) (0,5 Ponto) Considerando-se que o estado inicial seja 00, calcular a porcentagem de acertos e erros de previsão, considerando-se que o loop termina com 10 iterações;

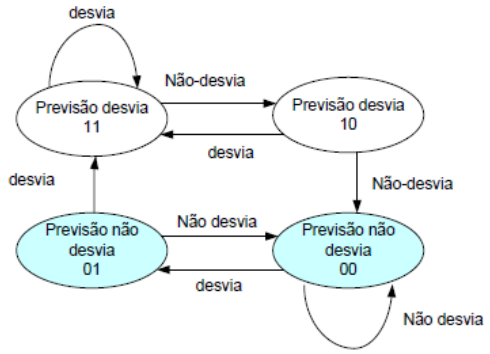
Em um preditor de dois bts, teremos na primeira vez que o loop for executado uma precisão de 70% e nas próximas execuções uma precisão de 90%. Considerando os dois loops, teremos 4 erros em 20 iterações, logo 80%. Observe que a partir do primeiro loop a precisão passa a ser 90%.

b) (0,5 Ponto) Comparar com o caso em que não use esse sistema de previsão, e apenas considere que a previsão seja sempre de desvio;

Esta predição é baseada no fato que os laços devem ser realizados, enquanto que os testes condicionais (else) devem sempre falhar. Logo se consideramos que um certo loop é executado duas vezes e que loop termina com 10 iterações. Neste caso, teremos nas duas vezes que forem executados um erro somente na última iteração. Logo um acerto de 90%.

c) (0,5 Ponto) comparar com o sistema de previsão de um bit

Se considerarmos um preditor de um bit, teremos um erro na primeira iteração do loop e um erro na última iteração, resultado em 80% de acerto. O preditor de 1 bit tem melhor precisão que o preditor de 2 bit no primeiro loop. No entanto, nos próximos loop, o preditor de 2 bit apresenta melhor precisão.



3ª Questão) (0,5 ponto) Identificar as situações de dependência (WAW, WAR, RAW) na seguinte sequência de código, do MIPS64:

DIVD F1, F3, F5
ADDD F4, F1, F9
SD F4, 0(R1)
SUBD F1, F10, F14
MULD F9, F10, F8

WAW – I1,I4
WAR – I4, I2; I5, I2
RAW- I1, I2; I3, I2

4ª Questão) (1,0 Ponto) Assinale verdadeiro (V) ou falso (F).
(Lembre-se: um item assinalado incorretamente anula um item assinalo corretamente)

(V) RISC apresenta poucos formatos de instrução e muitos registradores de uso genérico, enquanto CISC possui instruções de vários comprimentos (no mesmo conjunto)
(F) Arquitetura superpipeline baseia-se no aumento das unidades funcionais de forma que seja possível executar mais de uma instrução em cada ciclo de relógio
(V) Na RISC a complexidade esta no compilador, enquanto na CISC a complexidade esta no microprograma
(F) Uma arquitetura super-escalar consiste em aumentar o número de estágios da pipeline, conseguindo diminuir Tcc e aumentar a frequência de relógio
(V) Arquitetura vetorial especifica uma série de operações a realizar em vários dados, numa só instrução
(V) Uma arquitetura com grau de grau de super-escalaridade igual a 2 apresenta 2 ciclos de penalização (5 instruções) nos saltos previstos incorretamente
(F) Programas compilados para arquitetura CISC possuem garantia que serão menores que os compilados para RISC.
(F) No mecanimo de write back uma escrita modifica o dado na cache e memória juntos
(V) No caso em que não há escalonamento dinâmico, as instruções são emitidas pela ordem com que são geradas pelo compilador, executadas pela mesma ordem e terminadas ainda em ordem
(V) Tamanhos e posições das instruções são fixos e alinhados de acordo com o tamanho de uma palavra em RISC

5ª Questão) (1,0 Ponto) Considere o conjunto de instruções abaixo

					Latência
I1	Div	F6	F6	F4	4
I2	Lw	F2	45(R3)		1
I3	mult	F0	F2	F4	3
I4	Div	F8	F6	F2	4
I5	Sub	F10	F0	F6	1
I6	Add	F6	F8	F2	1

a) Identifique as situações de dependência (WAW, WAR, RAW) na seguinte sequência de código acima, do MIPS64:
WAW – I1 e I6;
WAR - I4 e I6; I5 e I6, I6 e I1
RAW – I1 e I4; I1 e I5; I2 e I3; I2 e I4; I2 e I6; I3 e I5; I4 e I6
Cada 0,045

b) Apresente uma sequência de termino em ordem e outra em fora de ordem (que execute no menor tempo)
1 2 -- --1 2 3 4 -- 3 5 4 6 5 6
1 2 2 3 1 4 3 5 5 4 6 6

Descontar 0.1 – se não indicou que o número significa fim da instrução

6ª Questão) (1,5 Ponto) Considere o trecho de programa no quadro abaixo e os conteúdos iniciais de registradores e posições de memória relevantes. Convenções: X – bolha, F - flush do pipeline, -- para estágio não usado, -_

adiantamento ou leitura após escrita no mesmo ciclo. Estágios do pipeline: BI(Busca), DI (Decodificação), EX (Execução) MEM (Memória) WB (Writeback)

addi $t4, $zero, 2 **root : add $t1, $t2, $t3** **lw $t3, 0x100($t1)** **sw $t3, 0x200($t1)** **subi $t4, $t4, 2** **beq $t4, $t3, root** **addi $t3, $t3, 0x100**	**Conteúdos iniciais da memória e dos registradores relevantes:** **$t1=0x100, $t2=0x100, $t3=0x100, $t4=0x100** **Mem [0x100-0x103]= 0x002345AB** **Mem [0x200-0x203]= 0x0000000A** **Mem [0x300-0x303]= 0x00000000** **Mem [0x400-0x403]= 0x00CD5F00**

a) (1,0 Ponto) Simule a execução completa do programa (considere unidade de adiantamento).

Ciclos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
addi $t4, $zero, 2	BI	DI	EX	M	WB																		
add $t1, $t2, $t3		BI	DI	EX	M	WB																	
lw $t3, 0x100($t1)			BI	DI	EX	M	WB																
sw $t3, 0x200($t1)				BI	DI	X	EX	M	WB														
subi $t4, $t4, 2					BI	X	DI	EX	M	WB													
beq $t4, $t3, root							BI	DI	EX	M	WB				Assumindo decisão sobre o desvio no estagio M, poderia ser no estágio EX								
addi $t3, $t3, 0x100								BI	DI	EX	F	F			Instrução carregada incorretamente, será descartada. Veja que era a última instrução.								
add $t1, $t2, $t3											BI	DI	EX	M	WB								
lw $t3, 0x100($t1)												BI	DI	X	M	WB							
sw $t3, 0x200($t1)													BI	DI	X	EX	M	WB					
subi $t4, $t4, 2														BI	X	DI	EX	M	WB				
beq $t4, $t3, root																BI	DI	EX	M	WB			
addi $t3, $t3, 0x100																	BI	DI	EX	M	WB		

Execução -0,25
Bolhas – 0,25
Adiantamento – 0,25
Instrução descartada – 0,25

b) (0,5 Ponto) O que a unidade de adiantamento (forward) está fazendo durante o quinto ciclo de execução? Se algumas comparações estiverem sendo feitas, mencione-as.

A unidade de adiantamento está realizando duas ações em paralelo:
1 - Comparando os registradores fonte da instrução LW $t3,0x100($t1) com os registradores destino das instruções ADD $t1, $t2, $t3 e ADDI $t4, $zero, 2.
2 - Adiantando o valor correto de $t1 da instrução ADD $t1, $t2, $t3 para a instrução LW $t3,0x100($t1).

7ª Questão) (1,0 Ponto) Considere a seguinte sequencia de instruções, e assuma que estas sejam executadas em um pipeline com 5 estágios (BI(Busca), DI (Decodificação), EX (Execução) MEM (Memória) WB (Write-back))

Sequencia Instruções
lw $1, 40 ($6) **add $2, $3, $1** **sw $2, 20($4)** **add $1, $6, $4** **and $1, $1, $4**

a) (0,4 Ponto) Quais dependências são conflitos (hazards) que podem ser resolvidos com adiantamento? Quais dependências que são conflitos e irão provocar a parada (bolhas) na execução?

Sem adiantamento

Ciclos	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16
lw $1, 40 ($6)	BI	DI	EX	MEM	WB										
add $2, $3, $1		BI	DI	X	X	EX	MEM	WB	Esperando $a1						
sw $2, 20($4)			BI	X	X	DI	X	X	EX	MEM	WB	Esperando $S2			
add $1, $6, $4						BI	X	X	DI	EX	MEM	WB			
and $1, $1, $4									BI	DI	X	X	EX	MEM	WB
													Esperando $s2		

Com adiantamento

Ciclos	1	2	3	4	5	6	7	8	10	11
lw $1, 40 ($6)	BI	DI	EX	MEM	WB					
add $2, $3, $1		BI	DI	X	EX	MEM	WB			
sw $2, 20($4)			BI	X	DI	EX	MEM	WB		
add $1, $6, $4					BI	DI	EX	MEM	WB	
and $1, $1, $4						BI	DI	EX	MEM	WB

Conflitos

lw $1, 40 ($6) e add $2, $3, $1 - RAW (pode produzir duas bolhas)
lw $1, 40 ($6) e add $1, $6, $4, - WAW (não pode ser resolvido com adiantamento)
lw $1, 40 ($6) e add $1, $1, $4, - WAW, RAW (não gera bolha neste exemplo)
add $2, $3, $1 e sw $2, 20 ($4) – RAW
add $1, $6, $4, and $1, $1, $4 – RAW, WAW

Conflitos de dados resolvidos 0.25
add $s2, $s3, $1, sw $2, 20($4)
add $1, $6, $4, and $1, $1, $4
Conflitos que irão provocar uma parada 0.25
lw $1, 40 ($6), add $2, $3, $1

b) (0,4 Ponto) Se não há adiantamento ou detecção de conflito, insira nops para assegura a execução correta e desenhe o diagrama de execução do pipeline para este código
Instruções 0,25
Diagrama 0,15

Sequencia Instruções
lw $1, 40 ($6)
nops
nops
add $2, $3, $1
nops
nops
sw $2, 20($4)
add $1, $6, $4
nops
nops
and $1, $1, $4

Ciclos	1	2	3	4	5	6	7	8	10	11						
lw $1, 40 ($6)	BI	DI	EX	MEM	WB											
Nops		X	X	X	X	X										
Nops			X	X	X	X	X									
add $2, $3, $1				BI	DI	EX	MEM	WB								
Nops					X	X	X	X	X							
Nops						X	X	X	X	X						
sw $2, 20($4)							BI	DI	EX	MEM	WB					
add $1, $6, $4								BI	DI	EX	MEM	WB				
Nops									X	X	X	X	X			
Nops										X	X	X	X	X		
and $1, $1, $4											BI	DI	EX	MEM	WB	

c) (0,4 Ponto) Repita o item anterior, mas adicione nops somente quando um conflito não pode ser evitado por mudando ou rearranjando estas instruções. Você pode assumir o registrador R7 para guardar valores temporários em seu código modificado.

Renomeando o **add $2, $3, $1, para add $7, $3, $1**
add $1, $6, $4, and $1, $1, $4, → add $8, $6, $4, and $8, $8, $4,

Sequencia Instruções
lw $1, 40 ($6)
add $2, $3, $1
sw $2, 20($4)
add $1, $6, $4
and $1, $1, $4

Sequencia Instruções
lw $1, 40 ($6)
add $7, $6, $4
nops
add $2, $3, $1
nops
and $7, $7, $4
sw $2, 20($4)

d) (0,3 Ponto) Um conflito estrutural (duas instruções tentando acessar a memória) pode ser resolvido pelo compilador inserindo uma instrução nops?

Não, pois como o conflito estrutural mencionado é devido a dois acessos a memória. Logo adicionar um nops, não resolve, pois teremos que continuar acessando a memória para buscar a instrução.
Não -0.1
Justificativa – 0.2

e) (0,25 Ponto) Suponha as instruções abaixo. Qual o procedimento a ser adotado pela unidade de detecção de conflito

load $1,(10) $2
add $2, $1, $3

Deve se inserir uma bolha (0.15 ponto). Neste caso, todos os sinais de controle deverão ser fixados em 0 (zero) para os estágios EX, MEM e ER. Estes valores de sinais de controle são gerados no estagio DI e são passados adiante a cada ciclo de relógio, produzindo o efeito desejado (nenhum registrador ou memória é escrito)

f) (0,25 Ponto) Apresente o teste de conflito realizado no estagio EX e MEM pela unidade de adiantamento.

Conflitos no Estágio EX
se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (EX/MEM.RegistradorRd = DI/EX.RegistradorRs)) Adianta.A = 10
se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (EX/MEM.RegistradorRd = DI/EX.RegistradorRt)) Adianta.B = 10

Conflitos no Estágio MEM
se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (MEM/ER.RegistradorRd = DI/EX.RegistradorRs)) Adianta.A = 01
se (EX/MEM.EscReg = 1
e (EX/MEM.RegistradorRd ≠ 0)
e (MEM/ER.RegistradorRd = DI/EX.RegistradorRt)) Adianta.B = 01

erro no sinal igual descontar 0,1

8ª. Questão) (1.0 Ponto) Mostrar o resultado (décimo ciclo) do uso do placar(scoreboard) para a sequência de instruções, considerando-se que a instrução LD leva 1 ciclo para execução; MULD, 6 ciclos. ADDD e SUBD levam 3 ciclos; e DIVD, 20 ciclos.

LD F2, 34(R2)
LD F6, 45(R3)
MULD FO, F2, F4
SUBD ,F8, F6, F2
DIVD F10, F0, F6
ADDD F6, F8, F2

Instruction status

Instruction		j	k	Issue	Read operan	Executi comple	Write Result
LD	F2	34+	R2	1	2	3	4
LD	F6	45+	R3	5	6	7	8
MULT	F0	F2	F4	6	7	13	14
SUBD	F8	F6	F2	7	9	12	13
DIVD	F10	F0	F6	8	15	35	36
ADDD	F6	F8	F2	14	15	18	19

Functional unit status

Time	Name	Busy	Op	dest Fi	S1 Fj	S2 Fk	FU for j Qj	FU for k Qk	Fj? Rj	Fk? Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	No								

Register result status

Clock	F0	F2	F4	F6	F8	F10	F12	...	F30
39	FU								

9ª Questão) (1,5 Ponto) Mostrar o resultado do uso do algoritmo de Tomasulo, com os mesmos números de ciclos para a mesma sequência de instruções da questão anterior.

Estado da instrução

Instrução	j	k	Emite	EX final	Escreve resultado
lw F2, 34(R2)			1	3	4
lw F6, 45(R3)			2	4	5
mult F0, F2, F4			3	10	11
sub ,F8, F6, F2			4	8	9
div F10, F0, F6			5	31	32
add F6, F8, F2			6	12	13

	Busy	Ender (A)
LD1	N	
LD2	N	
LD3	N	

Estações de reserva

Tempo restante	RS	Busy	Op	S1 Vj	S2 Vk	RS Qj	RS Qk
	Add1	N					
	Add2	N					
	Add3	N					
	Mult1	N					
	Mult2	N					

Registro de estado dos resultados

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU									