1. Seria possível conviver com um processador utilizando apenas instruções do tipo JAL, sem a instrução do tipo J? Existe alguma consequência indesejada? E o inverso, é possível conviver apenas com instruções do tipo J e não JAL?

Sim, é possível, pois JAL faz toda a funcionalidade da instrução J. Contudo, JAL consome mais energia que a instrução J, pois salva o $pc+4 no $ra. Adicionalmente, usando apenas JAL, o código deve ser construído com mais cuidado para não perder o endereço de retorno em um lugar onde se deseja apenas fazer um desvio.

O inverso não é possível, pois seria necessário ter uma instrução adicional que leia o $pc e salve seu valor em algum registrador. A instrução J apenas altera o $pc, não fazendo acesso deste registrador para leitura.

2. Porque uma instrução do tipo JAL deve ser utilizada em uma chamada à função, ao invés de uma instrução do tipo J? O que uma instrução realiza a mais que a outra?

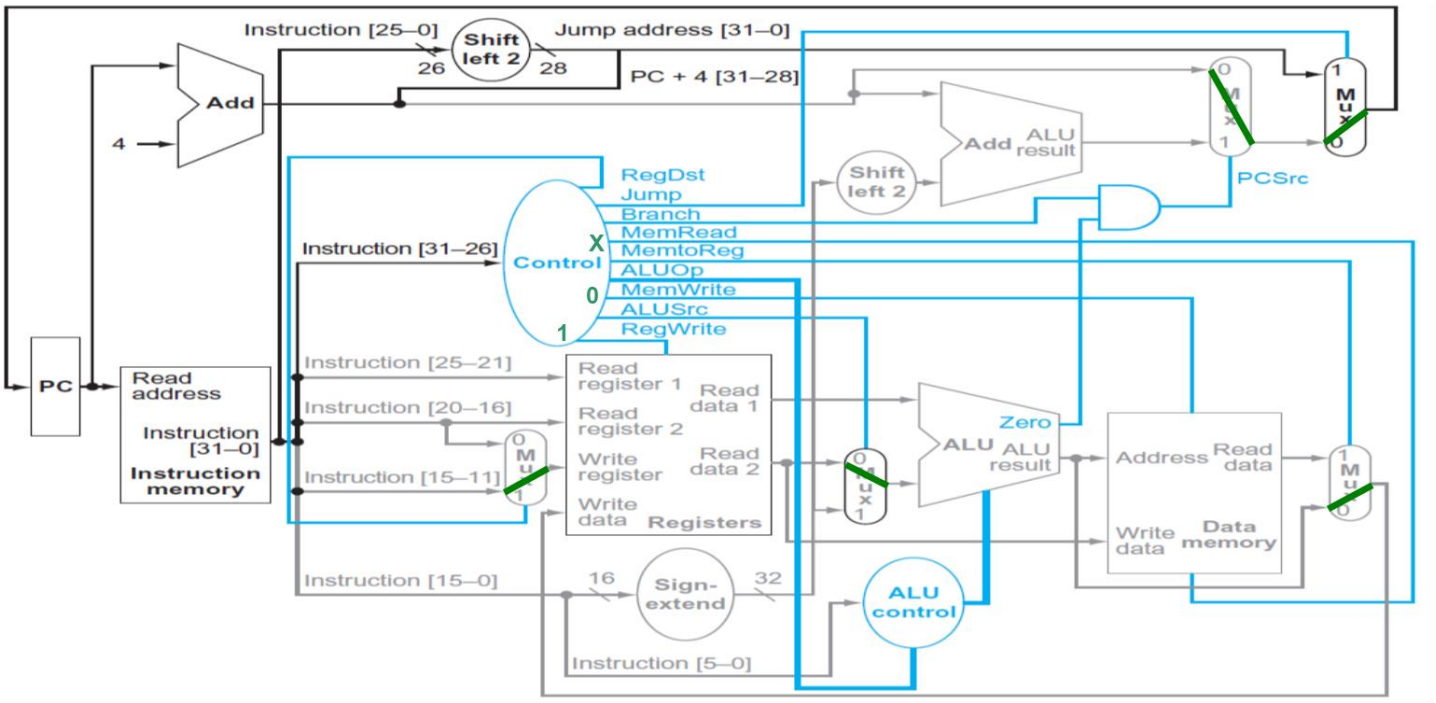
Porque a instrução JAL salva o endereço de retorno ($pc + 4) no registrador $ra, enquanto a instrução J apenas desvia o $pc para o endereço destino; desta forma, é possível retornar para o local que chama a função.

3. Quais são os valores dos sinais de controle necessários para executar a instrução **`move $t3, $t5`** na organização do MIPS monociclo apresentada em aula, sabendo que esta é uma pseudo-instrução implementada pela instrução **`addu $t3, $zero, $t5`**?

`addu rd, rs, rt`

0	rs	rt	rd	0	0x21
6	5	5	5	5	6

RegDst	= 1	Branch	= 0	MemtoReg	= 0	MemWrite	= 0
Jump	= 0	MemRead	= X	RegWrite	= 1	ALUSrc	= 0



4. Assinale qual das instruções a seguir produz o conjunto de sinais de controle abaixo na organização do MIPS monociclo apresentada em aula.

RegDst	= X	Branch	= 1	MemtoReg	= X	MemWrite	= 0
Jump	= 0	MemRead	= X	RegWrite	= 0	ALUSrc	= X

( ) `jr rs`

0	rs	0	8
6	5	15	6

(X) `bgez rs, label`

1	rs	1	label
6	5	5	16

( ) `jal target`

3	target
6	26

( ) `add rd, rs, rt`

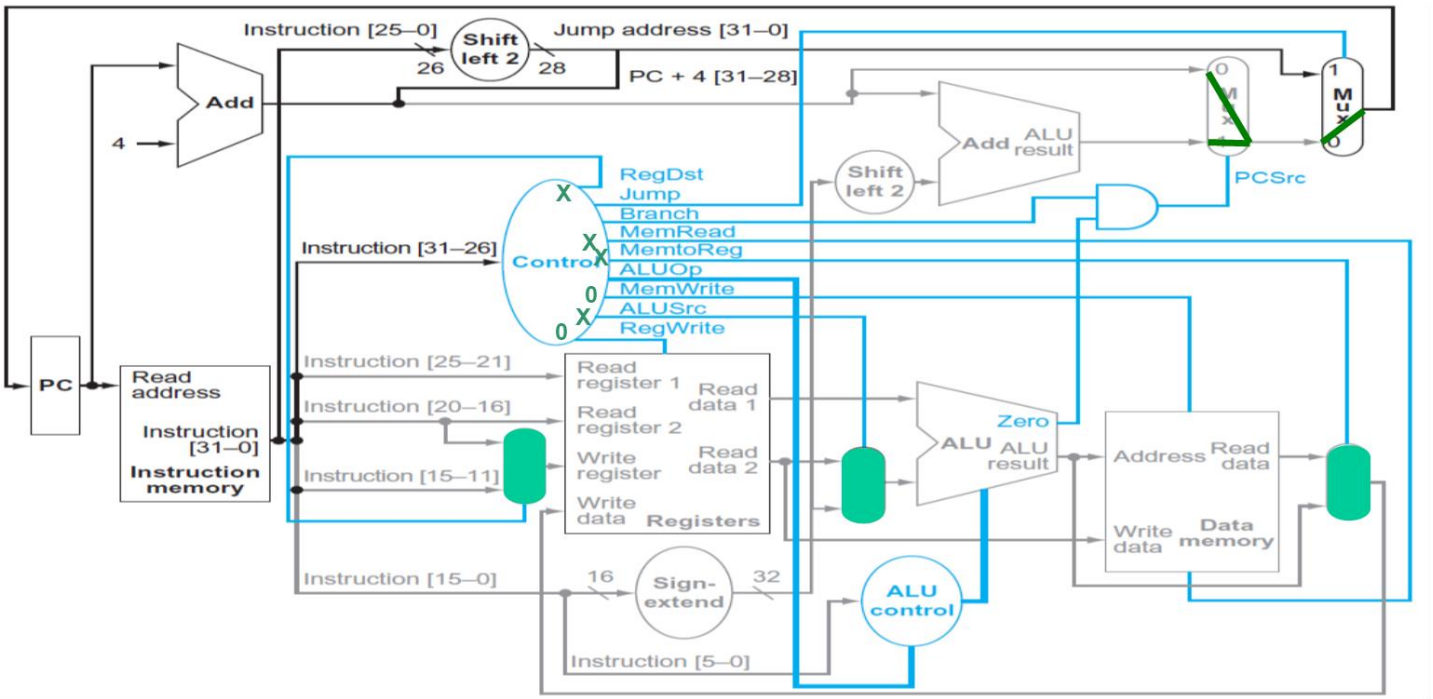
0	rs	rt	rd	0	0x20
6	5	5	5	5	6

( ) `bne rs, rt, label`

5	rs	rt	label
6	5	5	16

( ) `sll rd, rt, shamt`

0	0	rt	rd	shamt	0
6	5	5	5	5	6

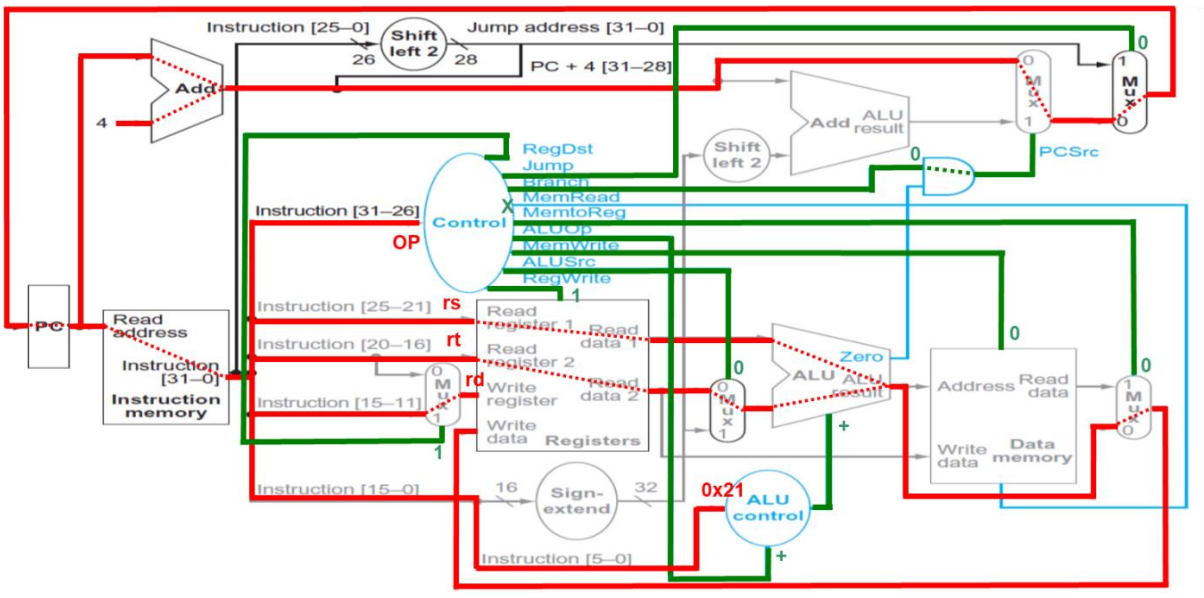


5. Ilustre sobre a figura completa do MIPS todos os caminhos de dados e sinais de controle necessários para o funcionamento da instrução add (soma à registrador)

add rd, rs, rt: rd = rt + rs

`addu rd, rs, rt`

0	rs	rt	rd	0	0x21
6	5	5	5	5	6

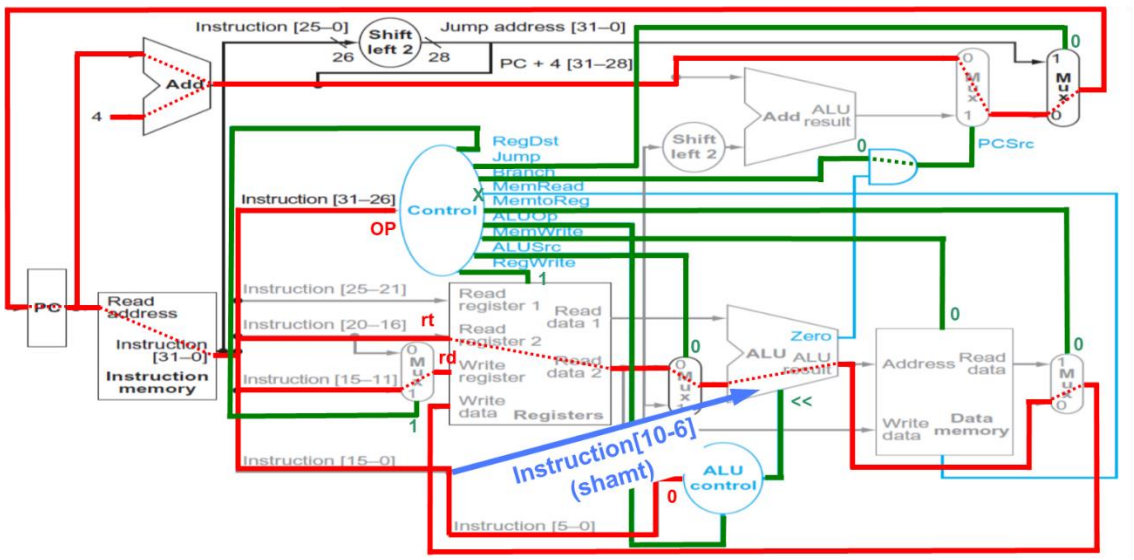


6. Analise se organização do MIPS monociclo apresentada em aula dá suporte à execução da instrução **`sll`** (e.g., **`sll $t3, $t5, 5`**). Caso não tenha suporte, diga porque, e sugira suscintamente uma solução.

sll rd, rt, shamt

0	0	rt	rd	shamt	0
6	5	5	5	5	6

Não tem suporte, pois a organização do MIPS não trata o campo shamt que indica o deslocamento. Duas soluções possíveis são (i) a inclusão de uma unidade de deslocamento que receba o registrador $rt e o campo shamt e efetue o deslocamento de bits. Adicionalmente, é necessário multiplexar a saída da unidade de deslocamento com a ULA; (ii) aumentar a funcionalidade da ULA, recebendo o campo shamt, bem como alterar o sinal ALUOp



7. Descreva o que caracteriza uma organização multiciclo, comparando com as organizações monociclo e pipeline.

A organização multiciclo divide o fluxo de dados do processador em estágios separados por barreiras temporais, sendo a execução de cada instrução controlada por uma máquina de estados finita.

Semelhante à organização monociclo, a organização multiciclo não suporta a execução de instruções simultâneas. Contudo, as instruções de uma organização multiciclo podem ser executadas com diferentes latências, enquanto que a monociclo tem todas as instruções executadas com a mesma latência.

A organização pipeline se assemelha com a multiciclo por ter barreiras temporais. Contudo, a organização pipeline suporta a execução simultânea de mais de uma instrução, desta forma, todas as instruções têm a mesma latência.

8. Sabendo das características de cada classe de instrução e dos estágios que compõem o MIPS básico (B: Busca, D: Decodificação; E: Execução, M: Memória de dados e W: Write back), diga quais estágios que são executados por todas as instruções, e quais são executados por instruções das classes: (i) à registrador; (ii) desvio condicional; (iii) store e (iv) load.

Todas as instruções ➔ B, D

À registrador ➔ B, D, E, W

Desvio condicional ➔ B, D, E

Store ➔ B, D, E, M

Load ➔ B, D, E, M, W

9. Apresenta a fórmula da métrica Ciclo por Instrução (CPI), considerando número de instruções e CPIs de classes de instruções de uma organização multiciclo.

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times \#I_i)}{\sum_{i=1}^{n} \#I_i}$$

CPI – CPI do programa

CPI_i – CPI da classe de instruções i

$\#I_i$ – número de instruções da classe i

n – número de classes de instruções

10. Aplique a fórmula obtida acima no cálculo da CPI nos programas P1 e P2 a serem executados na organização multiciclo com as CPIs de classes de instrução descritas abaixo.

	CPI_i	$\#I_i$	
		P1	P2
A	5	100	200
B	4	200	200
C	3	300	300
D	2	400	300

$$CPI_1 = \frac{\sum_{i=1}^{4}(CPI_i \times \#I_i)}{\sum_{i=1}^{4}\#I_i} = \frac{5 \times 100 + 4 \times 200 + 3 \times 300 + 2 \times 400}{100 + 200 + 300 + 400} = \frac{500 + 800 + 900 + 800}{1000} = \frac{3000}{1000} = 3$$

$$CPI_2 = \frac{\sum_{i=1}^{4}(CPI_i \times \#I_i)}{\sum_{i=1}^{4}\#I_i} = \frac{5 \times 200 + 4 \times 200 + 3 \times 300 + 2 \times 300}{200 + 200 + 300 + 300} = \frac{1000 + 800 + 900 + 600}{1000} = \frac{3300}{1000} = 3.3$$

11. Discuta aspectos de dissipação de potência e consumo de energia em máquinas monociclo e pipeline.

A dissipação de potência é uma medida instantânea que leva em consideração a atividade de um processador. O consumo de energia é o somatório da potência ao longo do tempo.

Uma máquina monociclo suporta apenas uma única instrução sendo executada, enquanto que o pipeline suporta uma instrução por estágio. Desta forma, é provável que a organização monociclo dissipe menos potência, já que terá menos partes do circuito em atividade. Por outro lado, a organização pipeline pode terminar as tarefas mais rapidamente, desta forma, a energia pode ser otimizada.

12. Com relação à organização pipeline, descreva o que é (i) estágio, (ii) barreira temporal, (iii) ciclo, (iv) latência, (v) balanceamento e (vi) vazão.

Estágio é uma divisão lógica/física do pipeline reservada para a execução de uma instrução.

Barreira temporal é um elemento de armazenamento do tipo registrador que separa cada estágio do pipeline.

Ciclo é o tempo de duração do estágio mais lento.

Latência é o tempo que uma instrução leva para percorrer todos os estágios do pipeline; seu valor pode ser estimado multiplicando o número de estágios pelo tempo de um ciclo.

Balanceamento é uma técnica para que os estágios tenham caminhos críticos próximos (idealmente iguais).

Vazão é o número de instruções que são executadas por unidade de tempo.

13. Fale porque o pipeline pode ser considerado uma técnica transparente para o programador de alto nível. O que isto tem a ver com a ideia de dar suporte à código legado?

Porque o programador de alto nível não precisa se preocupar como a organização do processador é construída. Basta saber qual o ISA que o programa será executado independente da organização. Um código implementado para uma organização monociclo pode ser utilizado para uma geração futura pipeline.

14. Porque um pipeline executando um número muito grande de instruções sem dependência tende a ter uma CPI próxima a 1?

Instruções sem dependência em um pipeline são executadas a cada ciclo de relógio, ou seja, uma instrução por ciclo de relógio. Adicionalmente, a primeira instrução do programa necessita percorrer todo o pipeline. Assim, apenas a primeira instrução levaria mais de um ciclo (levaria o número de estágios do pipeline). Se o programa for muito grande, o número de estágios do pipeline se torna insignificante.

$$\#C_{pipeline} = (n-1) + P = n + (P-1)$$

$\#C_{pipeline}$ – número de ciclos de um pipeline

n – número de instruções do programa

P – profundidade do pipeline (número de estágios)

$$CPI_{pipeline} = \frac{\#C_{pipeline}}{n} = \frac{n + (P-1)}{n} = 1 + \frac{P-1}{n}$$

$CPI_{pipeline}$ – CPI de um pipeline com instruções sem dependência

$$CPI_{MAX\_pipeline} = \lim_{n\to\infty}(CPI_{pipeline}) = \lim_{n\to\infty}\left(1 + \frac{P-1}{n}\right) = 1$$

$CPI_{MAX\_pipeline}$ – CPI máxima de um pipeline com instruções sem dependência

15. O que é caminho crítico para um circuito sequencial? O que ele tem a ver com a definição do relógio de operação do circuito?

Caminho crítico é o caminho de maior latência de todos os sinais que percorrem o circuito, considerando todo o atraso combinacional, mais o tempo para escrita na barreira temporal. A frequência do relógio de um circuito sequencial deve ser igual ou inferior ao inverso do tempo do caminho crítico.

16. O que pode acontecer se a frequência de relógio não respeitar o tempo definido pelo caminho crítico?

A barreira temporal pode capturar uma informação errada, que ocorreu antes do sinal estabilizar.

17. Diga quais são os três tipos de conflito que um pipeline básico tem. Descreva suscintamente cada um deles.

Conflito estrutural – é um conflito de hardware que ocorre quando uma combinação de instruções não pode ocorrer em um dado instante de execução. Este conflito é resolvido em tempo de projeto.

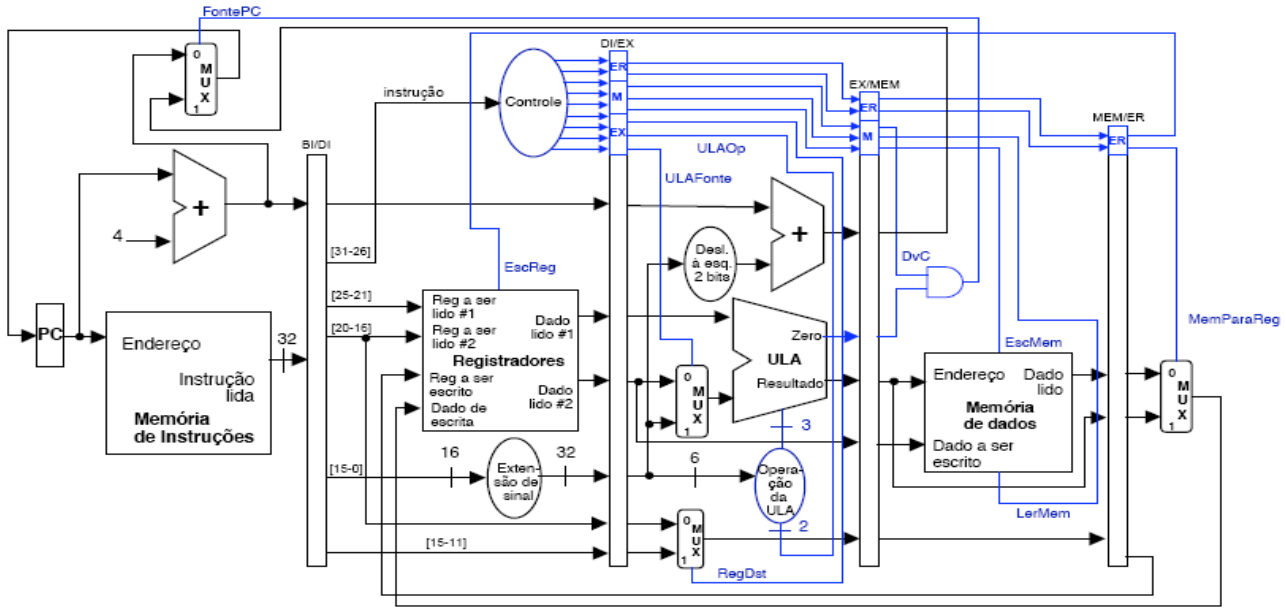
Conflito de controle – é devido aos saltos e desvios de um programa que não podem ser determinados antes da instrução de salto ou desvio ser decodificada/executada. Este conflito é tratado em tempo de execução.

Conflito de dados – é devido à dependência de dados entre instruções que estão executando dentro de um pipeline. Uma instrução subsequente depende do dado de uma instrução que ainda não terminou sua execução. Este conflito é tratado em tempo de execução.

18. Porque na implementação do MIPS vista em aula, o conflito de controle de uma instrução de desvio incondicional é resolvido no segundo estágio, enquanto que o conflito de uma instrução condicional requer passar pelo terceiro estágio?

A instrução de salto incondicional depende apenas da decodificação que é resolvida no segundo estágio. Por outro lado, a instrução de desvio condicional requer a comparação de operandos que é realizada no terceiro estágio (da ULA).

19. Note que a figura abaixo tem as barreiras temporais com diferentes sinais de controle, sendo que estes sinais de controle são reduzidos de uma barreira para outra subsequente. Justifique o porquê desta implementação.



Na organização pipeline os sinais de controle são propagados junto com a instrução, de forma que as ações de cada estágio do pipeline estarão sincronizadas. Quando um conjunto de sinais de controle terminou sua execução ele não precisa ser propagado para o próximo estágio. Este é o motivo que o número de sinais de controle é reduzido a cada nova barreira temporal.

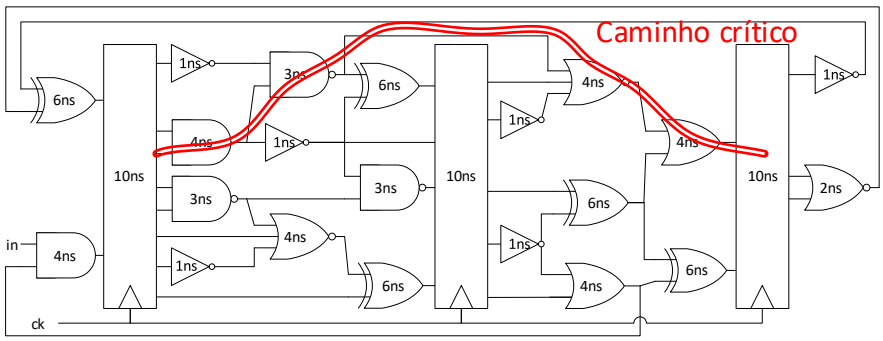
20. Explique porque a abordagem de banco de registradores de dois estágios permite que a leitura dos operandos da instrução I2 ocorra no mesmo ciclo de relógio da escrita dos operandos da instrução I1, sendo que os operandos de I2 são escritos na instrução I1.

O banco de registrador de dois estágios opera com as duas bordas de relógio. No primeiro instante (e.g., borda de subida do relógio) é feita a escrita do resultado de I1. No segundo instante (e.g., borda de descida do relógio) é feita a leitura dos operandos de I2 que deve corresponder ao registrador escrito em I1.

21. Uma multiplicação pode ser realizada de forma combinacional ou sequencial, com diversas implementações. Discorra sobre este assunto falando sobre as características prováveis de cada escolha.

Uma implementação combinacional provavelmente será mais rápida que uma equivalente sequencial, pois deverá ter o menor caminho crítico. Por outro lado, deve consumir maior área e dissipar mais potência. Conforme as técnicas utilizadas, tanto a implementação combinacional, quanto sequencial, podem variar a quantidade de recursos empregados, consumo de energia, dissipação de potência e área utilizada.

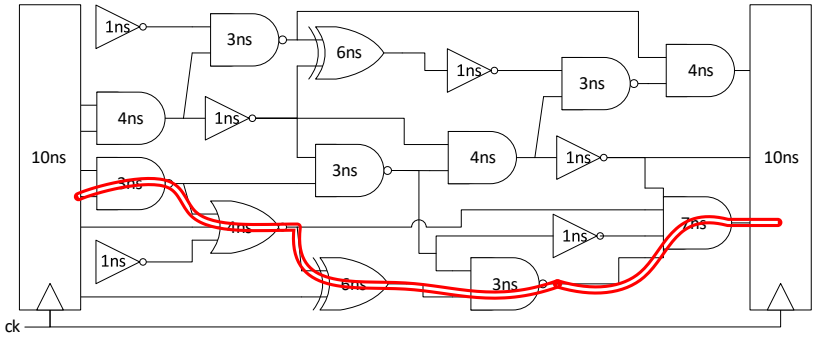
22. Qual a frequência máxima do circuito abaixo (ck) em MHz. Desconsidere atrasos de fios. Marque na figura qual é o caminho crítico.



T = 4ns + 3ns + 4ns + 4ns + 10ns
= 25ns

f = 1 / T
= 1 / 25ns
= 0.04 GHz
= 40 MHz

23. Ajude a um projetista calcular qual a máxima frequência de operação do circuito abaixo. Adicionalmente, é possível alcançar uma frequência de relógio superior a 50MHz com a inserção de mais uma barreira temporal. Caso não seja possível, faça uma proposta de organização com uma barreira temporal que alcance um resultado próximo ao proposto.



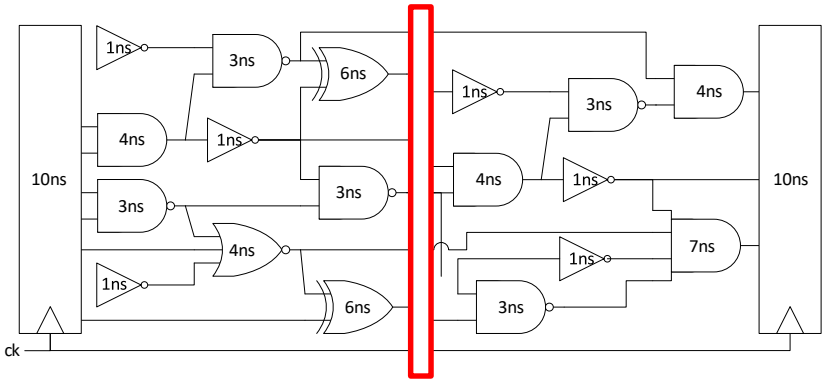
t=33ns

T = 3ns + 4ns + 6ns + 3ns + 7ns + 10ns
= 33ns

f_{MAX} = 1 / T
= 1 / 33ns
= 30.30MHz

t = 1 / f
= 1 / 50MHz
= 0.02us
= 20ns

PROPOSTA:



t=13ns t=12ns

Na configuração que o circuito está, não é possível acelerar o relógio para 50MZ, pois para alcançar esta frequência seria necessário um caminho crítico de no máximo 20ns; consequentemente, um caminho combinacional de 10ns. Todavia, o caminho combinacional que faz parte do caminho crítico tem latência de 23ns, sendo possível no máximo reduzir pela metade

24. Diga como é implementada uma bolha e como o uso de bolhas pode resolver os conflitos de dados e controle.

Bolhas são tipicamente instruções do tipo NOP. Elas apenas ocupam o estágio sem causar alteração. O objetivo é atrasar as instruções que estão no pipeline enquanto a dependência seja resolvida. No caso de um conflito de controle, a bolha normalmente é chamada de flush, pois ocorre a substituição de uma instrução buscada inadequadamente, por uma instrução NOP.

25. Preencha o diagrama com a execução do trecho de código em linguagem de montagem do MIPS. Convenção: X^k [bolha no estágio k], F [Flush], - [estágio sem operação], B [Busca], D [Decodificação], E [Execução], M [Memória de dados], W [Write-back].

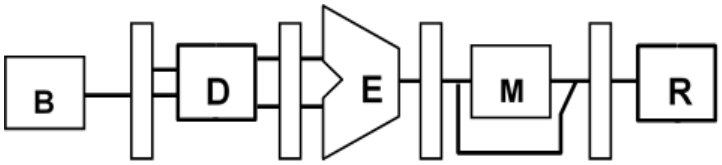
Instrução	Ciclo																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
add $1, $2, $2	B	D	E	-	W															
lw $2, 100($1)		B	D	X^D	X^D	E	M	W												
sub $3, $1, $1			B	X^B	X^B	D	E	-	W											
sw $2, 50($1)						B	D	X^D	E	M	-									
add $2, $3, $3							B	X^B	D	E	-	W								
sub $2, $2, $4									B	D	X^D	X^D	E	-	W					

26. Ilustre a execução do trecho de código no MIPS pipeline, colocando os estágios (B, D, E, M, W), bolha (X^k – bolha no estágio k) e flush (F) necessários. O código deve ser executado até que todas instruções que estiverem executando alcançarem o último ciclo disponível no quadro. Considere que A e B são vetores de 4 inteiros com endereços de memória 0x100 e 0x200, respectivamente, e que o código inicia com $t0=0, $t1 = &A[0], $t2 = &B[0], A[]={3,6,9,12} e B[]={2,4,8,16}. Ao final apresente os conteúdos de A, B e de todos os registradores utilizados.

Instrução	Ciclo																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
ini: bgt $t0, 4, fim	B	D	E	-	-					B	D	E	-	-					B	D	E
lw $t3, 0($t1)		B	D	E	M	W					B	D	E	M	W					B	D
addi $t1, $t1, 4			B	D	E	-	W					B	D	E	-	W					B
sw $t3, 0($t2)				B	D	X^D	E	M	-				B	D	X^D	E	M	-			
addi $t2, $t2, 4					B	X^B	D	E	-	W				B	X^B	D	E	-	W		
addi $t0, $t0, 1							B	D	E	-	W					B	D	E	-	W	
j ini								B	D	-	-	-					B	D	-	-	-
fim: li $v0, 10									B	F	F	F	F					B	F	F	F

A[]={3,6,9,12} B[]={3,6,8,16} $t0=2 $t1=0x108 $t2=0x208 $t3=6 $v0=

27. Complemente o exercício acima, apresentando a execução no diagrama temporal abaixo. Ao final, diga quantas bolhas ocorreram devido às dependências de dados e de controle.



	B	D	E	M	W
1	bgt				
2	lw	bgt			
3	addi	lw	bgt		
4	sw	addi	lw	bgt	
5	addi	sw	addi	lw	bgt
6	addi	sw	NOP	addi	lw
7	addi	addi	sw	NOP	addi
8	j	addi	addi	sw	NOP
9	li	j	addi	addi	sw
10	bgt	NOP	j	addi	addi
11	lw	bgt	NOP	j	addi
12	addi	lw	bgt	NOP	j
13	sw	addi	lw	bgt	NOP
14	addi	sw	addi	lw	bgt
15	addi	sw	NOP	addi	lw
16	addi	addi	sw	NOP	addi
17	j	addi	addi	sw	NOP
18	li	j	addi	addi	sw
19	bgt	NOP	j	addi	addi
20	lw	bgt	NOP	j	addi
21	addi	lw	bgt	NOP	j

Bolhas devido à dependência de dados: 2
Bolhas devido à dependência de controle: 2