Questionário

Arquitetura e Organização de Computadores

Os exercícios desta lista estão organizados em geral seguindo a apresentação do conteúdo em aula. No entanto, alguns exercícios podem requerem conhecimentos apresentados em aulas subsequentes. Naturalmente, não serão apresentados exercícios para as aulas reservadas para resolução de exercícios, provas, dúvidas, etc.

O Conteúdo a que se refere esta lista é encontrado nos capítulos 5, 6 e 7 da terceira edição do livro do Patterson e Hennessy (capa azul) e nos capítulos 4 e 5 da quarta edição (capa vermelha). Eu sugiro utilizar a terceira edição.

AULA 18: Organização de Computadores / Caminho de Dados MIPS-MONO / Subsistema de Busca de Instruções

(01) Com relação ao subsistema de busca de instruções, explique porque precisamos de um somador dedicado para somar o endereço do PC com a constante 4 para calcular o endereço da próxima instrução. Porque não podemos utilizar o somador localizado na ULA?

(02) Projete (ou isole a partir do diagrama do processador em anexo) o subsistema de busca de instruções simples (sem jump e beq).

(03) Explique dando exemplo como uma mesma ISA pode possuir duas ou mais organizações distintas. Em que contextos isso pode ocorrer?

(04) Na arquitetura MIPS1 toda instrução possui exatamente 32 bits. Isso torna o projeto do subsistema de busca de instruções simples, pois sabemos exatamente onde estará localizada na memória a próxima instrução (PC+4). Considere, no entanto a arquitetura x86 que possui tamanho de instrução variável, ou seja, o tamanho da instrução em bytes pode variar de 1 até várias dezenas de bytes. Como seria esquematicamente o subsistema de busca de instruções. Considere que o subsistema sempre lê um byte no início da busca de instruções e com base neste byte, após decodifica-lo, sabe quantos bytes subsequentes devem ser lidos.

(05) Desenvolva um programa capaz de iterar sobre um array de 10 posições e somar o número 42 ao valor originalmente contido em cada posição do array. Identifique o formato de cada instrução do seu programa. A seguir converta o programa resultante para código de máquina. Por fim, forneça o código de máquina em hexadecimal. Assuma que o programa começa na posição 0x0000 em hexa para fins de resolução de ETIQUETAS (LABELS).

AULA 19: Implementação da ULA e da Unidade de Controle

(06) Desenhe um diagrama da ULA contendo todas as entradas, saídas e sinais de controle. Defina também o número de bits de cada linha. A seguir explique a função de cada um.

(07) A ULA que vimos em sala de aula utiliza uma implementação simplista com o objetivo pedagógico e foi projetada sem consideração por desempenho ou consumo de energia. O que poderíamos mudar em seu projeto para melhorar o seu desempenho e minimizar o consumo de energia?

(08) Com relação a Unidade de Controle da implementação Monociclo, temos basicamente o problema de criar um circuito decodificador que converte 12 bits de entrada (opcode – 6 bits e funct - 6 bits) nos sinais de controle que habilitarão o caminho de dados correto para cada instrução. Utilize o circuito do processador dado nos anexos desta lista e preencha a tabela abaixo. (Não utilize a tabela de sinais de controle também dada nos anexos, apenas posteriormente para fins de verificação de sua resposta)

Instruction	op[5,4,3,2,1,0]	fc[5,4,3,2,1,0]	RegDst	Branch	MemRd	MemWr	M2R	AluSrc	RegWr	Jump
add	000000	100000								
sub	000000	100010								
and	000000	100100								
or	000000	100101								
nor	000000	100111								
xor	000000	100110								
slt	000000	101010								
addi	001000	----------								
andi	001100	----------								
ori	001101	----------								
xori	001110	----------								
sll	000000	000000								
srl	000000	000010								
beq	000100	----------								
j	000010	----------								
lw	100011	----------								
sw	101011	----------								

AULA 20: Implementação das Instruções do Tipo R

(09) Desenhe um diagrama do banco de registradores contendo todas as entradas, saídas e sinais de controle. Defina também o número de bits de cada linha. A seguir explique a função de cada um.

(10) Projete, tal como fizemos em sala de aula, um caminho de dados composto por: a)unidade de busca de instruções, a) banco de registradores; c) unidade lógica e aritmética e d) controle capaz de executar apenas instruções do tipo R (**add**, **sub**, **and**, **or**, **nor**, **xor**, **sll**, **srl**). Defina todos os sinais de controle e tamanhos em bits de todas as linhas de comunicação.

(11) Utilizando o projeto do caminho de dados apenas para instruções tipo R do exercício anterior, demonstre graficamente (pinte as linhas ativadas e escreva números onde for conveniente) como a seguinte instrução seria executada: **add $s0, $s1, $s2**

(12) Converta para código de máquinas as seguintes instruções:

a) add $s0, $s1, $s2
b) sub $s0, $s1, $s2
c) and $s0, $s1, $s2
d) or $s0, $s1, $s2
e) nor $s0, $s1, $s2
f) xor $s0, $s1, $s2

AULA 21: Implementação das Instruções do Tipo I

(13) Projete, tal como fizemos em sala de aula, um caminho de dados composto por: a)unidade de busca de instruções, a) banco de registradores; c) unidade lógica e aritmética e d) controle capaz de executar apenas instruções do tipo R (**add**, **sub**, **and**, **or**, **nor**, **xor**, **sll**, **srl**) e I (**addi**, **andi**, **ori**, **xori**). Defina todos os sinais de controle e tamanhos em bits de todas as linhas de comunicação.

(14) Utilizando o projeto do caminho de dados apenas para instruções tipo Re I do exercício anterior, demonstre graficamente (pinte as linhas ativadas e escreva números onde for conveniente) como a seguinte instrução seria executada: **addi $s0, $s1, 42**

(15) Estenda o projeto do exercício (13) para executar também a instrução **slt**. Que alterações são necessárias na ULA?

(16) Estenda o projeto do exercício (14) para executar também a instrução **beq.**

(17) O que mudaria na implementação do exercício 15 caso quiséssemos implementar o **bne** ao invés do **beq**?

(18) Explique o conceito de alinhamento de dados, e porque ele é importante para as instruções de acesso a memória.

(19) Estenda o projeto do exercício (15) para executar também a instrução **lw.** Nesta extensão faz-se necessário definir uma nova memória, a memória de dados (Data Memory).

(20) Porque é importante na implementação Monociclo que a memória de dados e a memória de texto sejam distintas?

(21) Estenda o projeto do exercício (18) para executar também a instrução **sw.**

AULA 22: Implementação de Instruções do Tipo J

(22) O que seria necessário para que o processador no anexo fosse capaz de executar a instrução **jr** (considere que a sintaxe de **jr** seja: 100001 sssss xxxxx xxxxx 00000 xxxxxx e que o registrador que contêm o endereço para o salto seja **jr** $rs)? Especifique as alterações de multiplexação e indique que subsistemas deverão ser alterados. Desenhe as alterações necessárias no diagrama do processador em anexo.

(23) O que seria necessário para que o processador no anexo fosse capaz de executar a instrução **jal?** (considere que a sintaxe de **jal** seja igual a da instrução **j**).

(24) Quais são as instruções que causam alteração no subsistema de busca de instrução? Descreva-as e desenhe o diagrama do subsistema de busca de instruções com os sinais de controle e multiplexadores relevantes.

(25) Imprima ou xeroque várias cópias do diagrama do processador no anexo. A seguir, selecione duas canetas de cores distintas e saliente as linhas de controle e de dados que serão ativadas na execução de cada uma das instruções a seguir. Escreva também associado as linhas de dados e controles os bits dos sinais.

a) add $s0, $s1, $s2

b) addi $s3, $7, 42

c) sub $s6, $t0, $t7

d) lw $a0, 8($sp)

e) j   0x000040

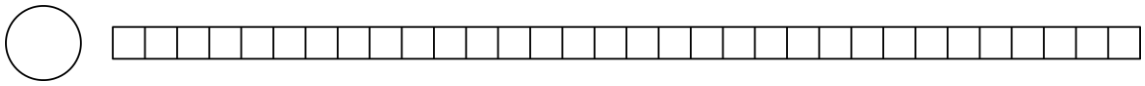
f) beq $s0, $s5, 0x00FC

g) sw $t6, 16($s7)

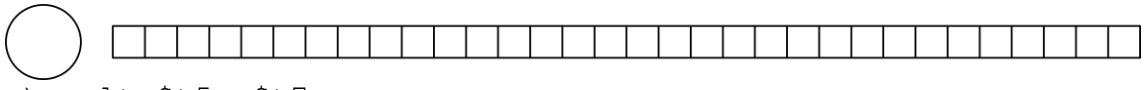
(25) Escreva um programa que, dado um array de 61 posições contendo as notas da P2 da disciplina GSI013 calcule a média da turma. A seguir monte o código de máquina das instruções que compoem o programa.

(26) Dada as instruções abaixo, utilize as tabelas 1 e 2 que se encontram ao final desta lista de exercícios para convertê-las para código de máquina.Indique no cículo a esquerda o tipo da instrução e divida os bits do código de máquina nos campos previstos nos três tipos de instruções descritos na tabela 3. Especifique também o modo de endereçamento de cada instrução.

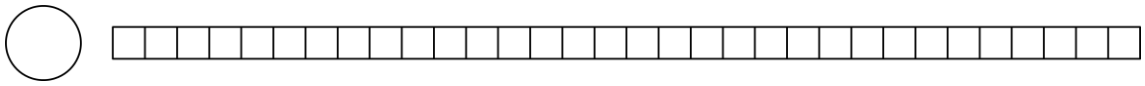
a) add $s0, $t0, $t1



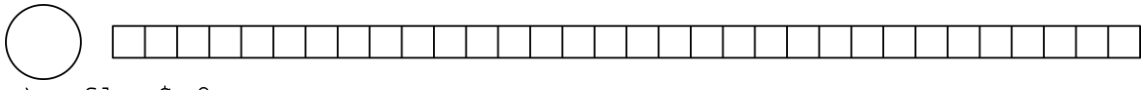
b) sub $s0, $s1, $s2



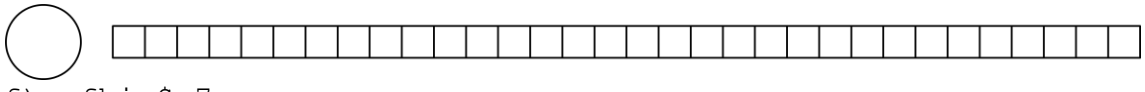
c) mult $t5, $t7



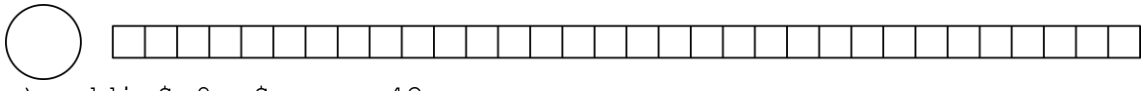
d) div $t5, $t7



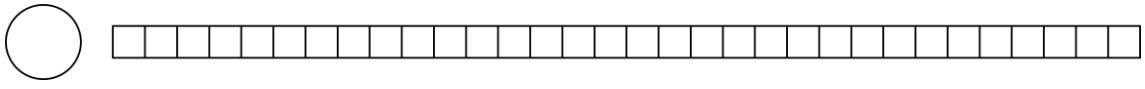
e) mflo $s0



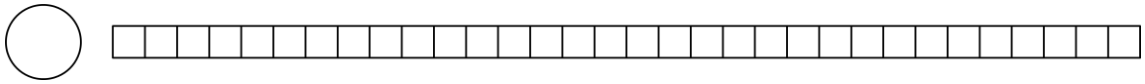
f) mfhi $s7



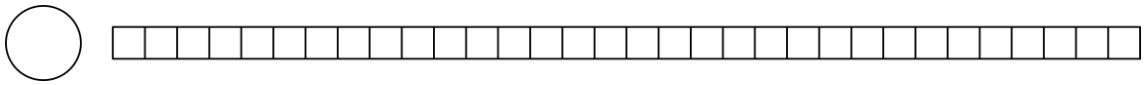
g) addi $s0, $zero, 42



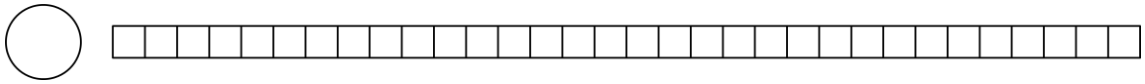
h) and $s0, $s1, $s2



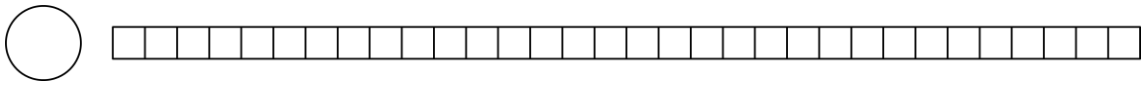
i) or $s0, $s1, $s2



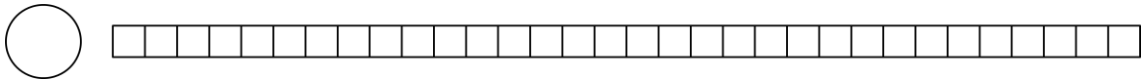
j) nor $s0, $s1, $s2



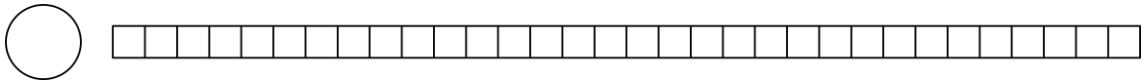
k) xor $s0, $s1, $s2



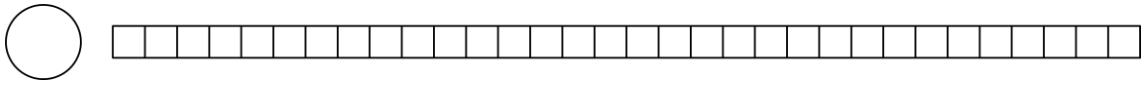
l) slt $t0, $s0, $s1



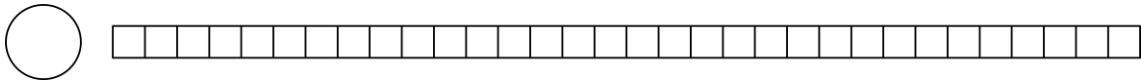
m) andi $s0, $s1, 15



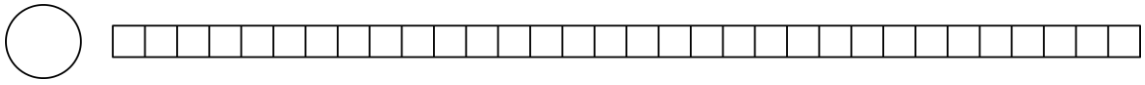
n) ori $s0, $s1, 0xAAAA



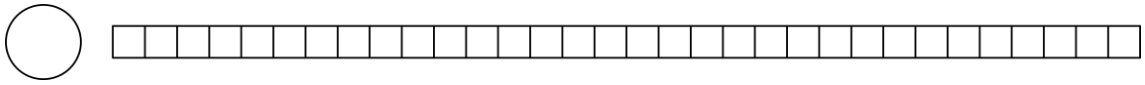
o) xori $s0, $s1, 0x00FF



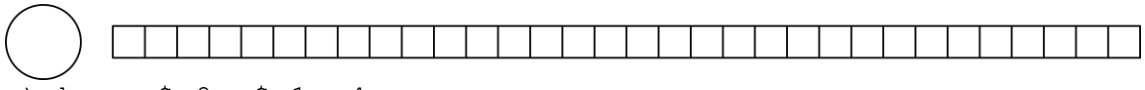
p) slti $t0, $s0, 42



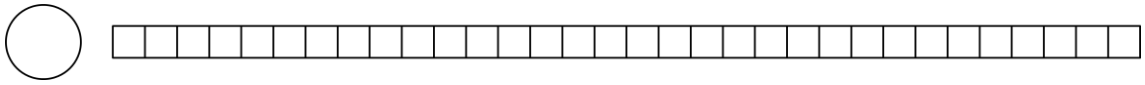
q) sll $s0, $s1, 14



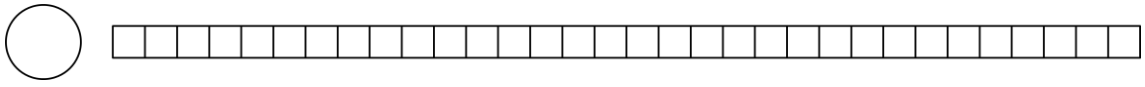
r) srl $s0, $s1, 17



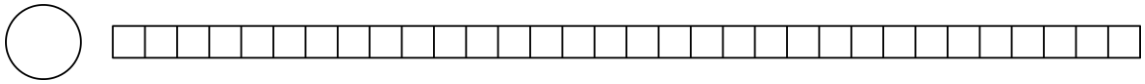
s) beq $s0, $s1, 4



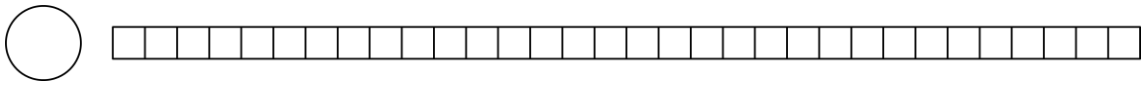
t) bne $s0, $s2, 8



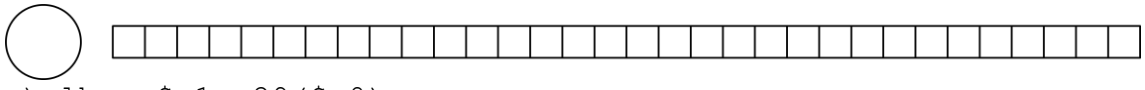
u) j LABEL # (LABEL = 0x00100040)



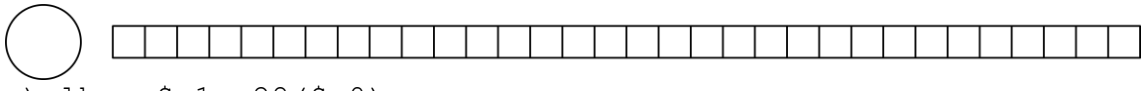
v) jr $s5



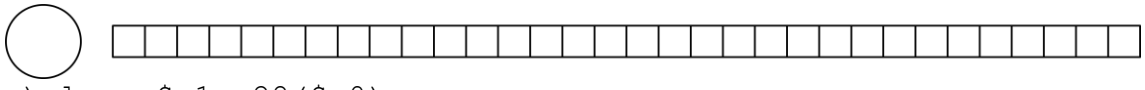
w) jal LABEL # (LABEL = 0x00102080)



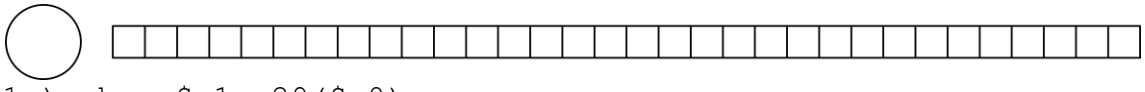
x) lb $s1, 29($s0)



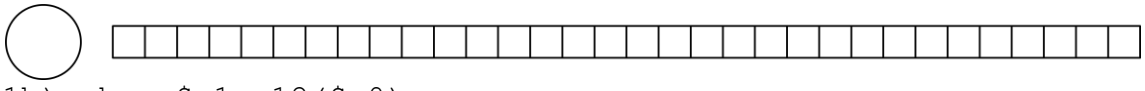
y) lh $s1, 28($s0)



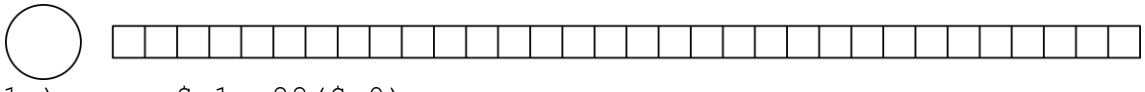
z) lw $s1, 28($s0)



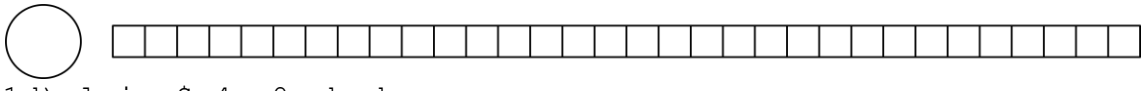
1a) sb $s1, 29($s0)



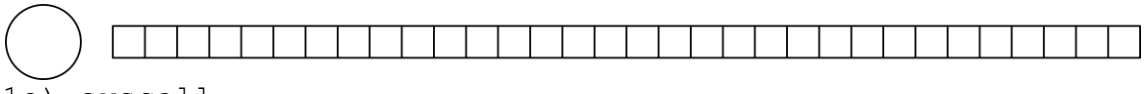
1b) sh $s1, 12($s0)



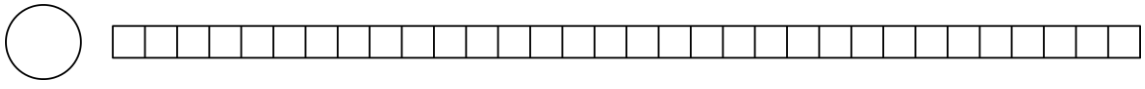
1c) sw $s1, 88($s0)



1d) lui $s4, 0xabcd



1e) syscall



(27) Escreva um programa que, dado um array de 100 posições de inteiros longos inverta seus elementos (v[99]↔v[0], v[98]↔v[1],...). A seguir converta todas as instruções do programa para código de máquina.

Aula 23: Desempenho MONO / Considerações sobre a implementação Multiciclo

(28) Processadores podem ser implementados utilizando a técnica MONOCICLO assim como a técnica MULTICICLO. Explique porque os processadores modernos não são monociclo.

(29) É possível implementar um processador monociclo que utilize a técnica de pipeline? Justifique sua resposta.

(30) Considere que a instrução mais lenta em um processador monociclo requeira 500ηs para ser executada. Um dado programa requer que 1542 instruções sejam executadas. Qual será o tempo total de execução do programa?

(31) Considere que um processador baseado na arquitetura MIPS1 é implementado utilizando duas organizações distintas, uma monociclo e outra multiciclo sem pipelining. Considere ainda que a instrução mais lenta em um processador monociclo seja o **LW** que requer 800ηs para ser executado. Na organização multiciclo sem pipeline o subsistema mais lento requer 200ηs para alcançar um estado estável. Data a tabela abaixo, estime quanto tempo as organizações mono e multiciclo respectivamente, requererão para executar o programa.

Instr.	MONO	MULTI
	#ciclos	#ciclos
lw	1	5
sw	1	4
add	1	4
addi	1	4
sub	1	4
beq	1	3
slt	1	4
j	1	2

Instr.	**Prog. A**	**Prog. B**
	# Instr.	**# Instr.**
lw	200	125
sw	200	70
add	550	2048
addi	50	4242
sub	30	27
beq	65	113
slt	30	117
j	20	514
TOTAL	**1145**	**7256**

AULA 24: Instruções do Tipo R, I e J na implementação Multiciclo

(32) O que muda nas instruções do tipo R, I e J na organização multiciclo em relação a organização monociclo vista em sala de aula? Discuta em termos da mecânica das instruções. Ignore registradores extra e o subsistema de controle.

(33) Porque a organização multiciclo requer um subsistema de controle mais complexo? Seria possível implementar o controle de uma organização multicilo utilizando um sistema digital combinacional? Justifique sua resposta.

(34) Considere a tabela abaixo. Ela lista um subconjunto de instruções da ISA MIPS1, o número de ciclos de clock que cada instrução requer em uma organização multiciclo, o tempo em ηs e os subsistemas que são necessários para a execução de cada instrução. Responda as seguintes perguntas:

Instrução	**# ciclos**	**Duração**	**Subsistemas**				
add	4	120 ηs	BI	DI	ULA		ER
sub	4	120 ηs	BI	DI	ULA		ER
and	4	120 ηs	BI	DI	ULA		ER
or	4	120 ηs	BI	DI	ULA		ER
nor	4	120 ηs	BI	DI	ULA		ER
xor	4	120 ηs	BI	DI	ULA		ER
slt	4	120 ηs	BI	DI	ULA		ER
addi	4	120 ηs	BI	DI	ULA		ER
andi	4	120 ηs	BI	DI	ULA		ER
ori	4	120 ηs	BI	DI	ULA		ER
xori	4	120 ηs	BI	DI	ULA		ER
beq	3	120 ηs	BI	DI	ULA		
lw	5	160 ηs	BI	DI	ULA	MEM	ER
sw	4	160 ηs	BI	DI	ULA	MEM	
j	2	80 ηs	BI	DI			

a) O que significam **BI, DI, ULA, MEM** e **ER**?

b) No subsistema **DI** duas etapas importantes na execução de uma instrução ocorrem. Quais são elas? Ambas estas etapas ocorrem em todas as instruções?

c) Monde o diagrama de alocação de subsistemas ao longo do tempo para os seguintes segmentos de código:

```
addi $s0, $zero, 42
addi $s1, $zero, 1
add  $s2, $zero, $zero
```

```
addi $s0, $zero, 42
addi $s7, $s0, -22
addi $s1, $s0, 1
add  $s2, $s0, $s1
sub  $s3, $s1, $s2
```


```
addi $s0, $zero, 42
lw   $s7, 0($s3)
addi $s1, $s0, 1
add  $s2, $s0, $s1
sw   $s1, 0($s3)
```

AULA 27: Pipelining

(35) Considere que a arquitetura MIPS1 possui duas organizações, uma Monociclo e outra utilizando a técnica de Pipelining. Na organização monociclo cada instrução requer 160ηs e na organização utilizando pipelining o clock possui período de 40 ηs. Para fins deste exercício considere a tabela apresentada no exercício (34). Qual o tempo necessário para executar um programa de 10^2, 10^3, 10^4, 10^5, 10^6, 10^9 e 10^{12} instruções em ambas as organizações. Considere para o caso da organização com pipelining que todas as instruções requerem exatamente 4 ciclos de clock.

(36) Considere os seguintes dados abaixo acerca do tempo de execução de cada um dos subsistemas que compõem um processador que utiliza a técnica de pipelining:

Passo		**Tempo**
BI –	**Busca Instrução**	**200 ηs**
DI –	**Ler Regs / Decod. Instrução**	**100 ηs**
EX –	**Executar Operação**	**200 ηs**

MEM – Acessar Memória 200 ηs
ER – Escrita Registrador 100 ηs

a) Monte o gráfico de alocação de subsistemas para a parcela de programa dado abaixo:

```
lw    $s1,0($s7)
add   $s0,$s1,$s2
sub   $s2,$s0,$s1
beq   $s0,$s2, SALTO
```

b) Discuta as situações no programa acima implicam em atrasos e complicações na execução em pipelining do programa acima.

(37) Explique os conceitos de **Hazard de Dados** e **Hazard de Controle.**

(38) Reordene a seguinte parcela de código de modo a evitar stalls.

```
addi   $s0, $zero, 42
addi   $s1, $zero, 1
add    $s3, $s1,   $s2
lw     $s7, 0($t0)
sub    $s6, $s7, $s1
add    $t0, $zero, $zero
addi   $t0, $zero, 41
...
```

Aula 28: Hierarquia de Memórias e Caches

(39) Desenhe um diagrama exemplificando os tipos tamanhos e velocidades dos diferentes tipos de memória utilizados comumente em sistemas computacionais.

(40) Com o advento da tecnologia SSD que apresenta um desempenho muito melhor que os discos rígidos mecânicos é possível afirmar que sistemas computacionais não precisam mais de memória RAM? Justifique sua resposta.

(41) Qual é a ideia que motiva a utilização de memórias cache?

(42) Explique o que vem a ser o **princípio da localidade**. Dê exemplos. Diferencie **localidade temporal** de **localidade espacial**.

(43) No contexto de memórias cache, o que vem a ser a arquitetura Harvard. Explique com exemplos.

(44) Explique o conceito de **HIT** e **MISS** no contexto de memórias cache.

(45) Considere um modelo de mapeamento de cache simples em que uma memória de 32 palavras é mapeada para uma cache de 8 palavras. Quantas posições de memória são mapeadas para a mesma posição na cache?

(46) Considere um modelo de mapeamento de cache simples em que uma memória de 512K (2^{20}) palavras são mapeadas para uma cache de 1k (2^{11}) palavras. Quantas posições de memória são mapeadas para a mesma posição na cache?

(47) Explique para que servem a **TAG** e o **bit de validade** em uma cache.

(48) Considerando os exercícios 45 e 46, calcule quantos bits serão necessários para compor a memória principal e para a memória cache. Considere que a cache precisa de bits de tag e bits de validade.

TABELA 1 - TIPOS DE INSTRUÇÕES

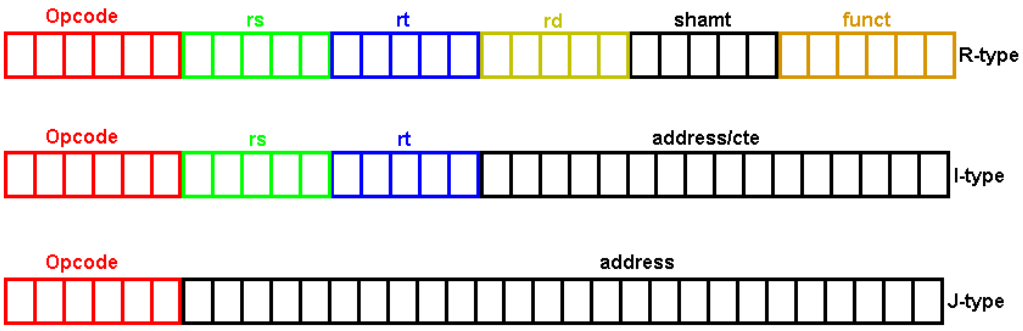
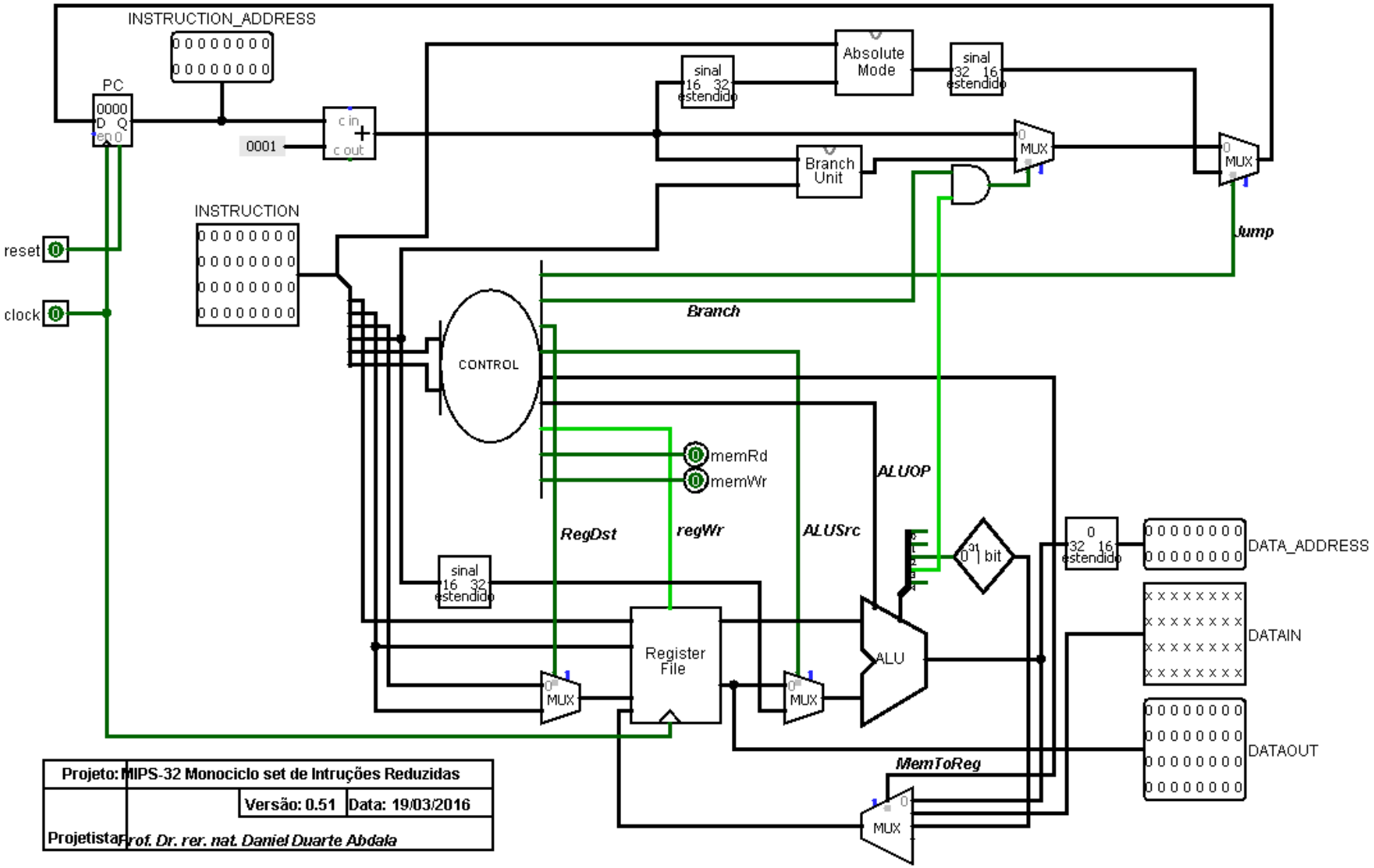


TABELA 2 - CHAMADAS DO SISTEMA

$V0	DESCRIÇÃO	ARGUMENTOS / VALOR DE RETORNO
1	imprime inteiro	$a0-int para impressão
4	imprime string	$a0-end do 1° byte da string
5	lê inteiro	$v0-int lido
8	lê string	$a0-end da string / $a1-num de bytes
9	aloca mem. na heap	$a0-num de bytes / $v0 -end do 1° byte
10	termina execução	retorna o controle ao SO
11	imprime caractere	$a-char para impressão
12	lê caractere	$v0-char lido



Projeto: MIPS-32 Monociclo set de Intruções Reduzidas		
	Versão: 0.51	Data: 19/03/2016
Projetista	*Prof. Dr. rer. nat. Daniel Duarte Abdala*	

Registers

$zero	0
$at	1
$v0-$v1	2-3
$a0-$a3	4-7
$t0-$t7	8-15
$S0-$S7	16-23
$t8-$t9	24-25
$k0-$k1	26-27
$gp	28
$sp	29
$fp	30
$ra	31

Instruction Format

Control Signals

Instruction	op[5,4,3,2,1,0]	fc[5,4,3,2,1,0]	RegDst	Branch	MemRd	MemWr	M2R	AluSrc	RegWr	Jump
add	0 0 0 0 0 0	1 0 0 0 0 0	0	0	0	0	00	0	1	0
sub	0 0 0 0 0 0	1 0 0 0 1 0	0	0	0	0	00	0	1	0
and	0 0 0 0 0 0	1 0 0 1 0 0	0	0	0	0	00	0	1	0
or	0 0 0 0 0 0	1 0 0 1 0 1	0	0	0	0	00	0	1	0
nor	0 0 0 0 0 0	1 0 0 1 1 1	0	0	0	0	00	0	1	0
xor	0 0 0 0 0 0	1 0 0 1 1 0	0	0	0	0	00	0	1	0
slt	0 0 0 0 0 0	1 0 1 0 1 0	0	0	0	0	10	0	1	0
addi	0 0 1 0 0 0	----------	1	0	0	0	00	1	1	0
andi	0 0 1 1 0 0	----------	1	0	0	0	00	1	1	0
ori	0 0 1 1 0 1	----------	1	0	0	0	00	1	1	0
xori	0 0 1 1 1 0	----------	1	0	0	0	00	1	1	0
sll	0 0 0 0 0 0	0 0 0 0 0 0	0	0	0	0	00	0	1	0
srl	0 0 0 0 0 0	0 0 0 0 1 0	0	0	0	0	00	0	1	0
beq	0 0 0 1 0 0	----------	x	1	0	0	xx	0	0	0
j	0 0 0 0 1 0	----------	x	0	0	0	xx	-	0	1
lw	1 0 0 0 1 1	----------	1	0	1	0	01	1	1	0
sw	1 0 1 0 1 1	----------	x	0	0	1	xx	1	0	0