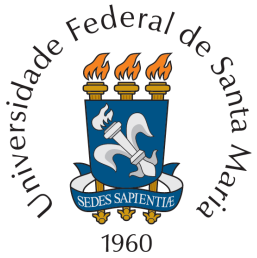


ORGANIZAÇÃO DE COMPUTADORES

TRABALHO 2:
RESOLUÇÃO DE QUESTÕES

JOÃO VITOR BELMONTE RATES

GIOVANI BARATTO

SANTA MARIA, RIO GRANDE DO SUL

2022

1. Escreva um procedimento, em assembly para o MIPS, para dividir dois números inteiros de 32 bits. Use o segundo algoritmo da divisão, apresentado em sala de aula. Escreva um programa, em assembly, para o MIPS, usando este procedimento para realizar a divisão x ÷ y, com x = 0x12341234 e y = 0x90357274. Mostre a saída da execução do seu programa no programa MARS. Verifique se o resultado da divisão apresentado pelo programa está correto.

O código que implementa a solução desta questão encontra-se no diretório q1.

2. Escreva um procedimento double cos(double x), em assembly para o MIPS, para calcular o cosseno de um ângulo x, dado em radianos. O procedimento calcula o cosseno usando uma série de Taylor expandida em x = 0 (veja a equação 1). No procedimento, trunque a a série em n = 7 (até o termo $x^{14}/14!$).

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2 \cdot n)!} x^{2 \cdot n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + \frac{x^{12}}{12!} - \frac{x^{14}}{14!} + \dots \qquad (1)$$

Crie um programa em assembly para o MIPS. O programa permite a entrada de um ângulo x em graus (◦), converte o ângulo para radianos, calcula o cosseno do ângulo usando o procedimento chama o procedimento cos() e imprime o resultado. Use o programa para calcular o seno de 57,23◦. Mostre a saída da execução do seu programa no programa MARS. Verifique se o resultado apresentado pelo seu programa está correto.

O código que implementa a solução desta questão encontra-se no diretório q2.

3. Represente o número X = 462,23438 em ponto flutuante, precisão simples. Mostre os passos na solução deste problema

Primeiramente é necessário obter a representação de X como número inteiro sem sinal, separando a parte inteira da fracionária

Parte inteira:

Divisão longa de X (parte inteira):		
Valor	Quociente	Resto
462	231	0
231	115	1
115	57	1
57	28	1
28	14	0
14	7	0
7	3	1
3	1	1
	Resultado:	111001110

Assim, a parte inteira de X, denotada I, será:

$I = 1\ 1100\ 1110_{\ 2}$

Parte Fracionária:

Multiplicação Longa de X (parte fracionária):			
Valor	Produto	Parte Decimal	Parte Inteira
0,23438	0,46876	0,46876	0
0,46876	0,93752	0,93752	0
0,93752	1,87504	0,87504	1

0,87504	1,75008	0,75008	1
0,75008	1,50016	0,50016	1
0,50016	1,00032	0,00032	1
0,00032	0,00064	0,00064	0
0,00064	0,00128	0,00128	0
0,00128	0,00256	0,00256	0
0,00256	0,00512	0,00512	0
0,00512	0,01024	0,01024	0
0,01024	0,02048	0,02048	0
0,02048	0,04096	0,04096	0
0,04096	0,08192	0,08192	0
0,08192	0,16384	0,16384	0
0,16384	0,32768	0,32768	0
0,32768	0,65536	0,65536	0
0,65536	1,31072	0,31072	1
0,31072	0,62144	0,62144	0
0,62144	1,24288	0,24288	1
0,24288	0,48576	0,48576	0
0,48576	0,97152	0,97152	0
0,97152	1,94304	0,94304	1
0,94304	1,88608	0,8860800001	1
0,8860800001	1,77216	0,7721600002	1
0,7721600002	1,54432	0,5443200003	1
0,5443200003	1,088640001	0,08864000067	1
Resultado:		0,00111100000000000101 0011111	

Observação: Devido à mudança decimal para binário, o valor obtido no resultado não é precisamente o buscado, mas sim uma aproximação (0,23437999933958053589).

Assim, a parte fracionária de X, denotada F, será:

$F = 0{,}001\ 1110\ 0000\ 0000\ 0010\ 1001\ 1111_2$

Tem-se que X = I+F:

$X = 1110\ 0111\ 0{,}001\ 1110\ 0000\ 0000\ 0010\ 1001\ 1111_2$

Agora de modo a obter a representação de X em ponto flutuante simples ($(1+F)*2^y$), inicia-se com $X=X*2^0$, e move-se a vírgula até o bit 1 mais a esquerda, para assim chegar a forma 1,F. Para cada deslocamento da vírgula para a direita, soma-se 1 ao expoente de 2.

Serão necessários 8 deslocamentos, assim tem-se:

$X = 1{,}110\ 0111\ 0001\ 1110\ 0000\ 0000\ 0010\ 1001\ 1111_2 * 2^8$

A parte fracionária de X agora é:

$F = 110\ 0111\ 0001\ 1110\ 0000\ 0000\ 0010\ 1001\ 1111$

Entretanto, como F deve conter, no máximo, 23 bits, faz-se o arredondamento:
$F = 110\ 0111\ 0001\ 1110\ 0000\ 0000_2$

Deve-se ainda encontrar o expoente polarizado (EP), para isto, soma-se, na base binária, o valor do expoente anteriormente encontrado ($8_{10} = 1000_2$) com o Peso (No caso da precisão simples, $127_{10} = 01111111_2$).

$$\begin{array}{rrr} \text{vai uns} & 1111 & 0 \\ 8 \rightarrow & 00001000_2 & \\ + \ 127 \rightarrow & 01111111_2 & \\ \hline = \ 135 & 10000111_2 & \end{array}$$

$EP = 1000111_2$

Resta então somente definir o bit S, que indica o sinal de X, 0 para positivo e 1 para negativo. Como X é positivo:

$S = 0$

Tendo F, EP e S definidos, pode-se então concatenar os bits na forma S||EP||F para chegar a representação de X em ponto flutuante de precisão simples

S EP F

0 | 100 0011 1 | 110 0111 0001 1110 0000 0000_2

Que em hexadecimal equivale a:

X = 0x43E71E00

4. Qual o valor decimal do número X = 0x78787800, representado em ponto flutuante,
precisão simples. Mostre os passos na solução deste problema.

Para obter o valor de X em decimal, primeiramente é necessário visualizá-lo como binário a fim de obter os seus valores determinantes, sinal (S), fração (F) e expoente polarizado (EP)

Partes	S	EP	F
n. bits	1	8	23
	0	111 1000 0	111 1000 0111 1000 0000 0000

Tendo definido os valores, usa-se a fórmula que define a notação de ponto flutuante:

$X_{10} = (-1)^{s}*(1,F)*2^{EP-peso}$

Para precisão simples, o peso tem valor 127.

$X_{10} = (-1)^{s}*(1,F)*2^{EP-127}$

Percebe-se que esta fórmula está escrita em notação decimal, portante antes de aplicar os valores obtidos da notação ponto flutuante de X, é necessário transcrevê-los de binário sem sinal com ponto fixo para decimal.

$S = 0_2 = 0_{10}$
$EP = 111\ 1000\ 0_2 = 240_{10}$
$F = 0,111\ 1000\ 0111\ 1000\ 0000\ 0000_2 = 0,941162109375_{10}$

Agora, então, pode-se aplicar os valores, em decimal, à fórmula:

$X_{10} = (-1)^{0}*(1,941162109375)*2^{240-127}$

Que terá como resultado:

$X_{10} = 2,0158179844829304e+34 = 2,0158179844829304 * 10^{34}$

5. Explique detalhadamente, usando as tabelas 1 e 2 e o diagrama de blocos do processador na figura 1, como a instrução sw $s0, 40($at) é executada pelo processador monociclo. Converta a instrução para linguagem de máquina, apresentando os campos. Apresente na figura os sinais de controle. Escreva um texto explicando como a instrução é executada.

Linguagem de Máquina:

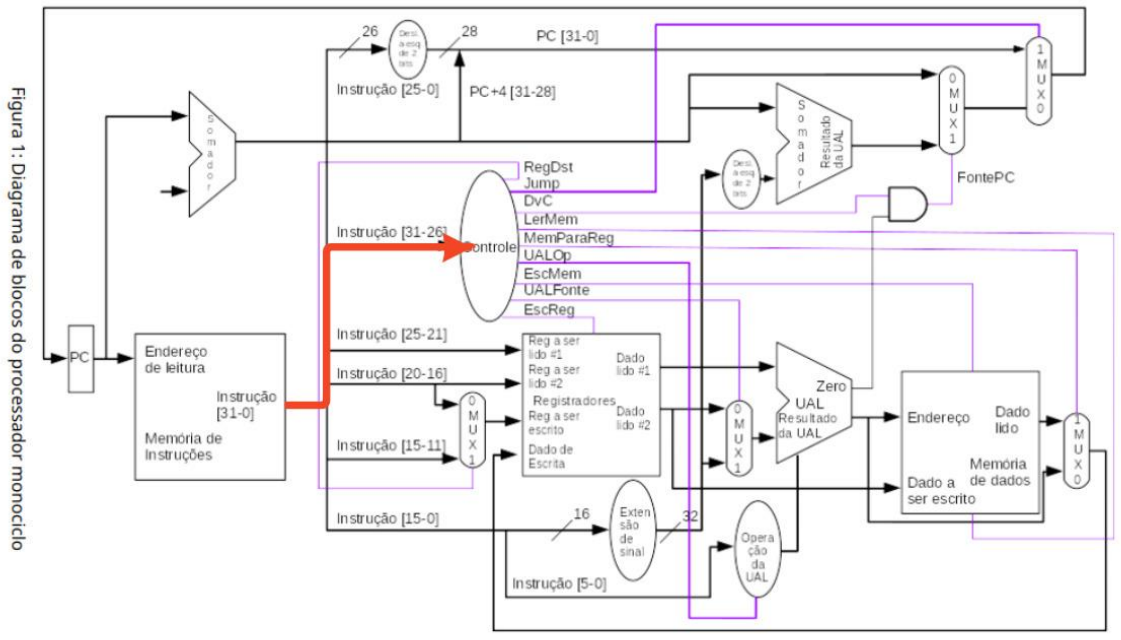
A instrução sw (store word) é uma instrução do tipo I, portanto é formada por

Opcode [31-26] | Rs [25-21] | Rt [20-16] | immediate [15-0]

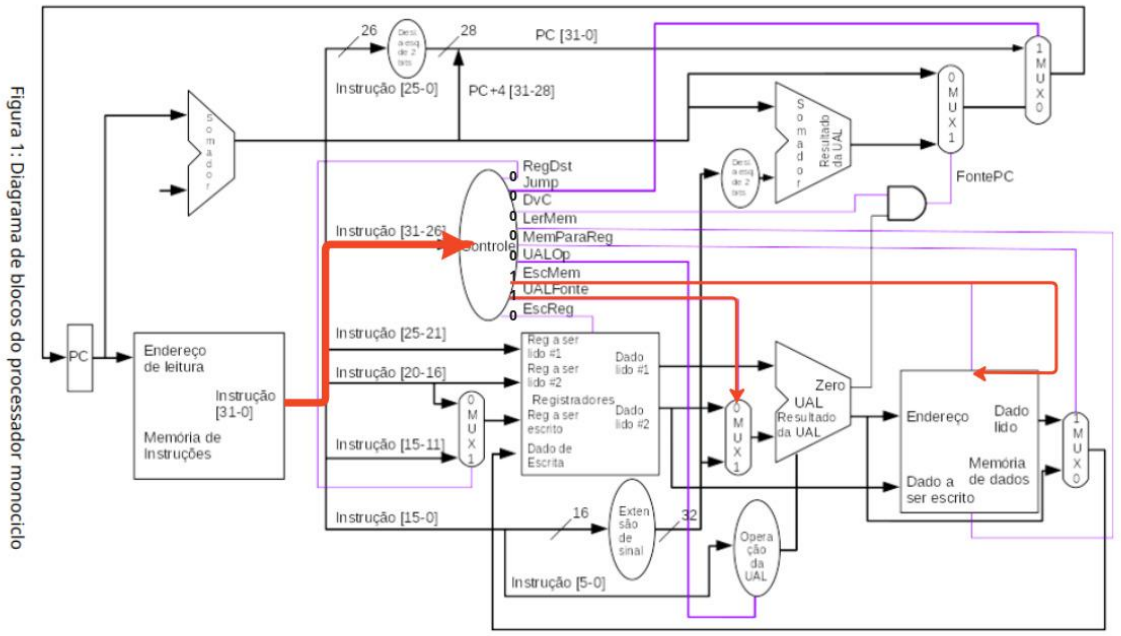
Com base na Tabela 1 e nos valores da instrução, esta tem o valor:

Opcode [31-26]	Rs [25-21]	Rt [20-16]	immediate [15-0]
101011	00001	10000	0000 0000 0010 1000

Execução da Instrução:
Com o valor de PC anteriormente setado para o endereço da instrução acima definida na memória de instruções, na próxima borda de subida do clock a instrução é amostrada e dividida. Sua porção opcode é direcionada à unidade de controle.



Com base na Tabela 1, as saídas da Unidade de Controle, RegDst e MemParaReg têm valor X, ou seja, irrelevante para esta instrução, UALFonte e EscMem tem seus valores definidos para 1, e as demais saídas tem seus valores definidos para 0.



sw $s0, 40($at)

Opcode [31-26]	Rs [25-21]	Rt [20-16]	Immediate [15-0]
101011	00001	10000	0000000000101000

Os bits que representam o endereço do registrador $at, 21 à 25, são direcionados a entrada de leitura #1 do banco de registradores. O valor contido no registrador $at (00001), é exposto na saída de leitura #1, e então, direcionado a uma das entradas de dados da Unidade de Lógica Aritmética

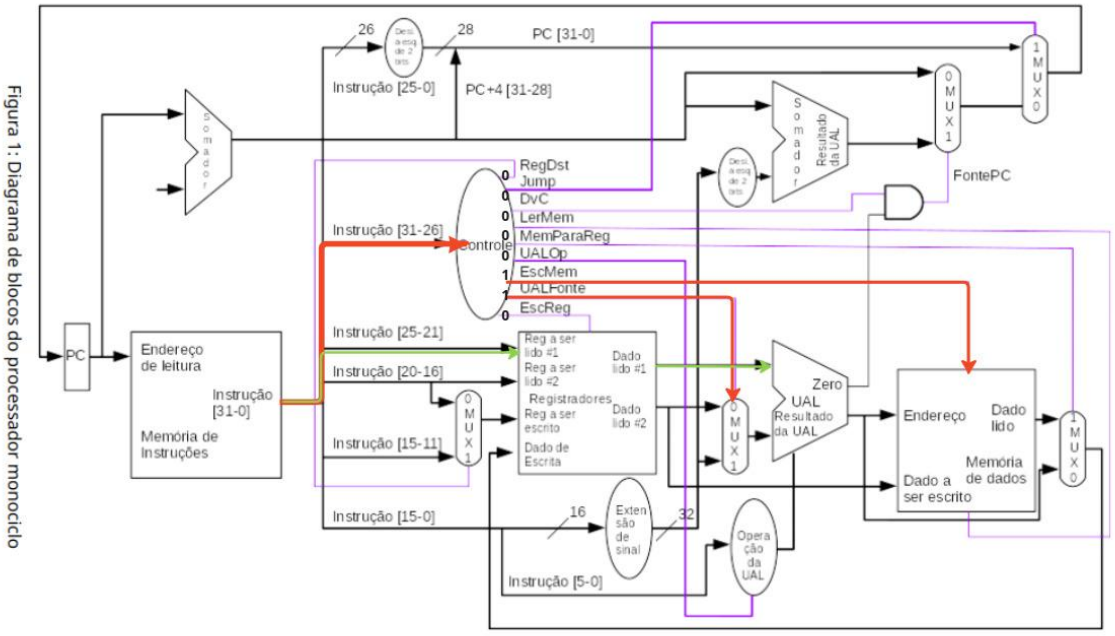


Figura 1: Diagrama de blocos do processador monociclo

sw $s0, 40($at)

Opcode [31-26]	Rs [25-21]	Rt [20-16]	Immediate [15-0]
101011	00001	10000	0000000000101000

Concomitantemente, os bits 20 à 16, que representam o endereço de $s0, são direcionados para o banco de registradores na entrada de leitura #2, e posteriormente o valor de $s0 é mostrado na saída #2. Este valor será exposto à entrada de escrita da memória de dados e ao multiplexador que antecede a Unidade de Lógica Aritmética, porém, como a unidade de controle definiu UALFonte para 1, o valor de $s0 não seguirá para a Unidade de Lógica Aritmética.

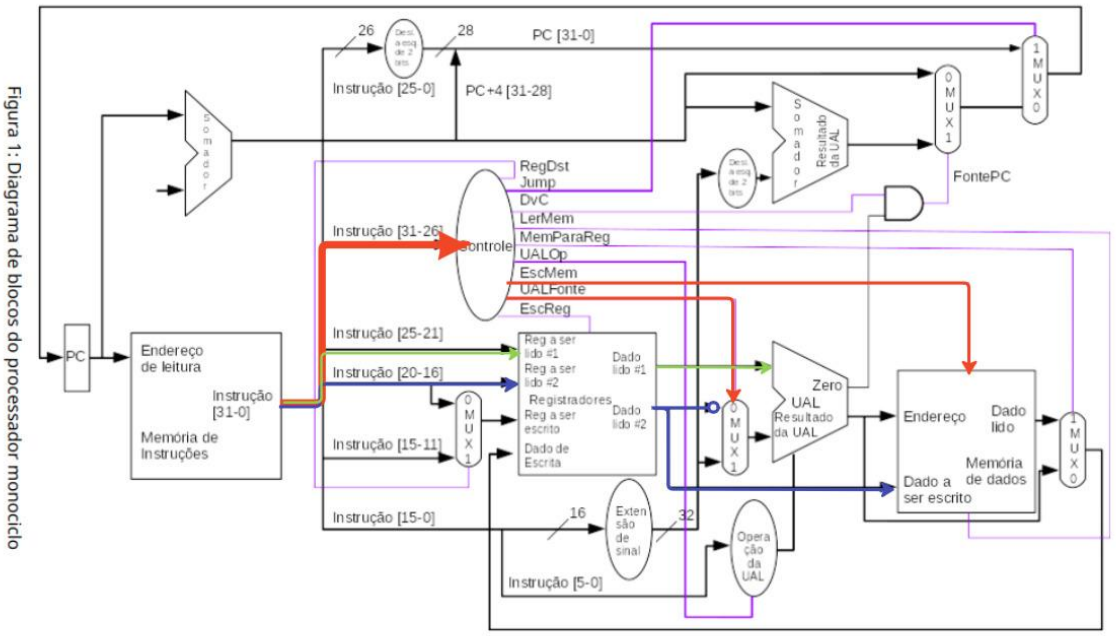
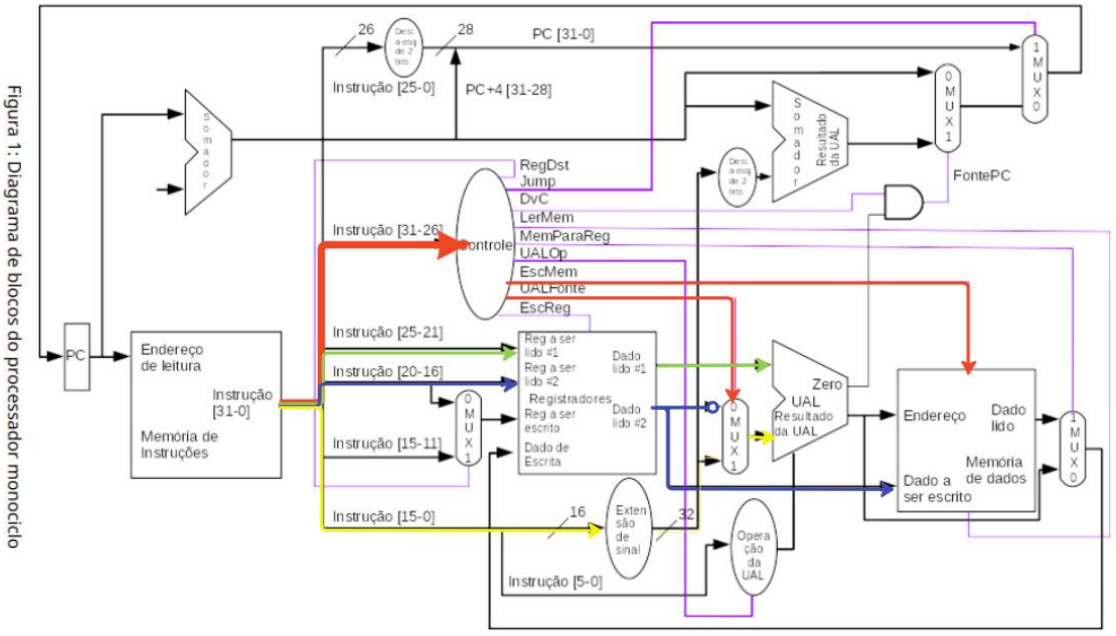


Figura 1: Diagrama de blocos do processador monociclo

sw $s0, 40($at)

Opcode [31-26]	Rs [25-21]	Rt [20-16]	Immediate [15-0]
101011	00001	10000	0000000000101000

O campo immediate, bits 0 à 15, são direcionados ao extensor de sinal que mantém o valor aritmético, mas estende a quantidade de bits de 16 para 32 bits. Então o valor de 32 bits é direcionado ao multiplexador que, como definido pela Unidade de Controle em UALFonte, passa o valor imediato para a segunda entrada da Unidade de Lógica Aritmética.



sw $s0, 40($at)

Opcode [31-26]	Rs [25-21]	Rt [20-16]	Immediate [15-0]
101011	00001	10000	0000000000101000

miro

Com base na Tabela 2, como já visto, a Unidade de Controle enviou o sinal 00 para a operação da Unidade de Lógica Aritmética. Os bits 0 à 5 da instrução também são direcionados a operação da Unidade de Lógica Aritmética, isto se refere ao campo funct de instruções do tipo R, como está instrução é do tipo I, está entrada é irrelevante. Para a Unidade de Lógica Aritmética é enviado o sinal 0010, que indica soma.

A Unidade de Lógica Aritmética soma os valores de sua entrada, 40 e o valor contido em $at, e direciona a saída para a entrada de endereços da memória de dados. A memória de dados então contém no seu endereço o valor de $at somado 40, em sua entrada dados o valor de $s0, como a Unidade de Controle enviou o sinal 1 para EscMem(Escrever na memória). Logo, será gravado o valor de $s0 em 40($at).

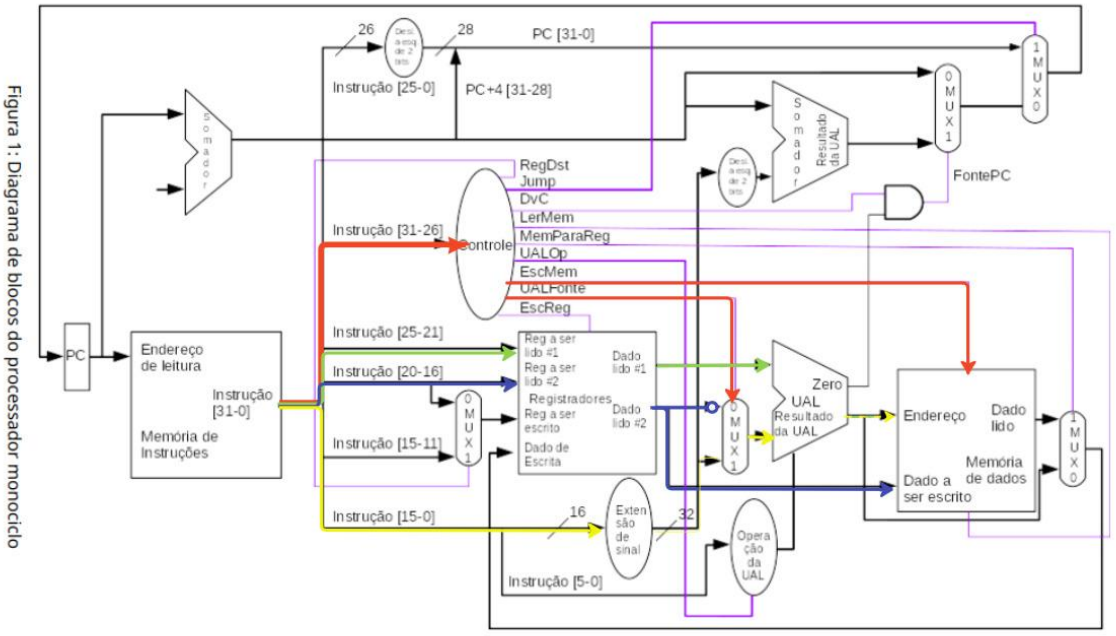


Figura 1: Diagrama de blocos do processador monociclo

sw $s0, 40($at)

Opcode [31-26] | Rs [25-21] | Rt [20-16] | Immediate [15-0]

101011 | 00001 | 10000 | 0000000000101000

miro

A partir deste ponto a instrução já foi executada, resta ainda atualizar o valor de PC, segue que PC além de ser direcionado a memória de instruções, é direcionado a um somando que soma o valor atual com 4, após isso o resultado enviado para um multiplexador na entrada 0, que permite a passada do valor, pois seletor estará definido para 0, pela Unidade de Controle, o mesmo processo ocorre novamente em outro multiplexador e o sinal passa por ele, novamente definido pela unidade de controle, com isso o valor de PC+4 é redirecionado ao registrador de PC. Terminando assim a execução da instrução, e definindo PC para o endereço da próxima instrução.

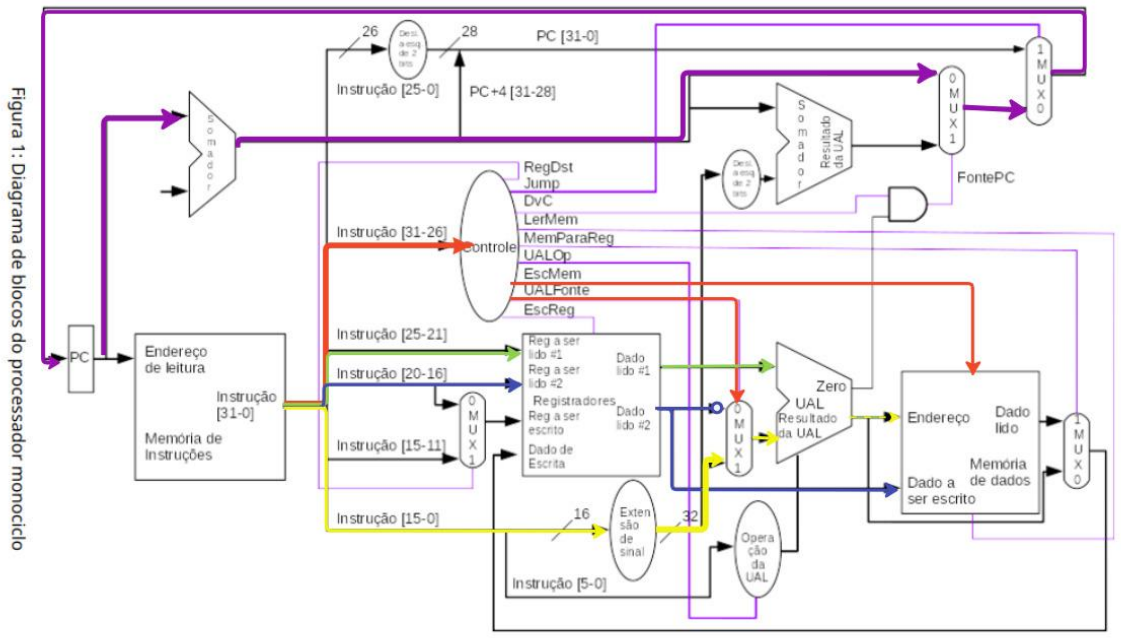


Figura 1: Diagrama de blocos do processador monociclo

sw $s0, 40($at)

Opcode [31-26]	Rs [25-21]	Rt [20-16]	Immediate [15-0]
101011	00001	10000	0000000000101000

Os campos da instrução também são direcionados a outras partes do circuito, entretanto, a Unidade de Controle garante que estas partes do circuito não tenham efeito algum ao setar os valores necessários para 0.

6. Explique detalhadamente, usando as tabelas 1 e 2 e o diagrama de blocos do processador na figura 1, como a instrução bne $a0, $a1, loop é executada pelo processador monociclo. O endereço desta instrução é 0x004000034 e loop é um rótulo para o endereço 0x00400018. Converta a instrução para linguagem de máquina, apresentando os campos. Apresente na figura os sinais de controle. Escreva um texto explicando como a instrução é executada.

Linguagem de Máquina e Configuração:

A instrução bne(branch on not equal) é uma instrução do tipo I, portanto é formada por:

Opcode [31-26] | Rs [25-21] | Rt [20-16] | immediate [15-0]

Entretanto, a instrução bne não possuí um valor de OpCode e este desing de processador não apresenta uma implementação da instrução. Para que seja possível será necessário fazer uma pequena alteração de hardware.

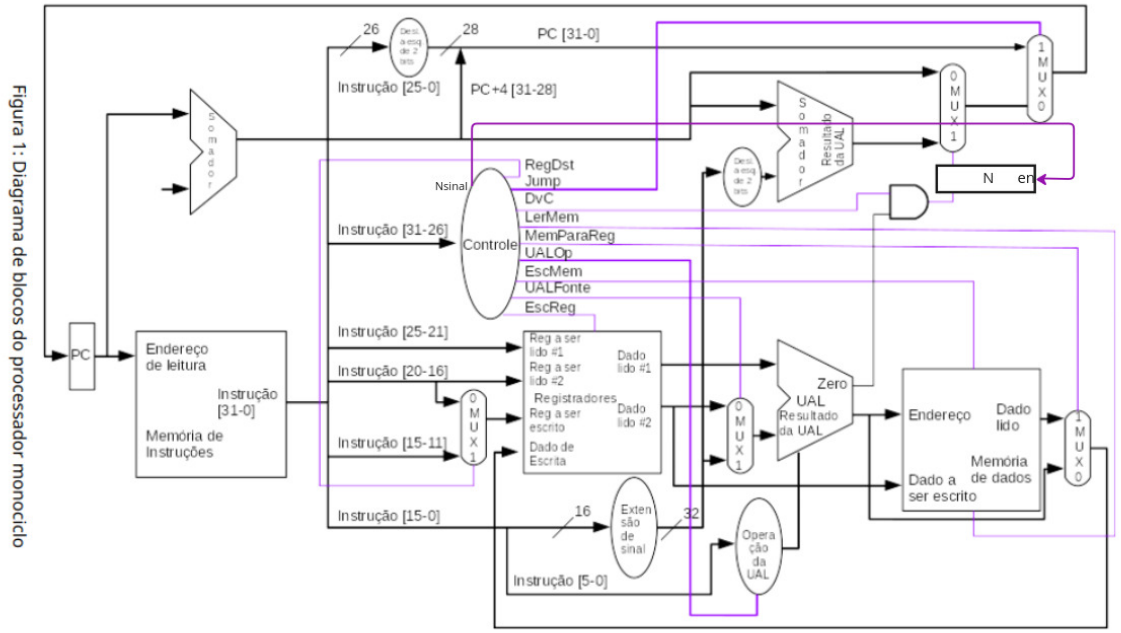
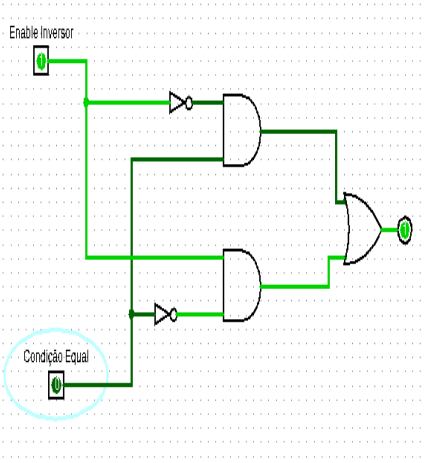


Figura 1: Diagrama de blocos do processador monociclo

A instrução bne se assemelha muito a instrução beq, a qual já está implementada no design do processador, difere-se pela negação da condição de ação. Portanto, pode-se implementar a instrução bne ao adicionar um inversor no bit

(N) controlado por um sinal de saída adicional da Unidade de controle, denominado Nsinal (negar sinal). O circuíto deste inversor está definido abaixo;

Enable Inversor	Condição Equal (beq)	Saída
0	X	X
1	X	!X



Agora com o circuito corrigido, podemos definir os campos da instrução, define-se arbitrariamente o campo Opcode da instrução bne para 000101, para o cálculo do valor imediato, deve-se considerar que mais a frente, neste texto, será visto que o valor imediato é multiplicado por 4 (shift left 2) e somado ao valor de PC mais 4. Visto isto para o cálculo do valor imediato, realiza-se a operação inversa. Obtém a diferença do valor de PC mais 4 e o endereço do loop e divide-por 4.

Os deslocamentos usados nesta e na próxima questão, aproveitam-se do fato de que, devido o alinhamento a memória, os endereços de PC sempre conterão os dois bits menos significativos iguais a 0. Assim:

0x04000034
- 0x04000018
= 0xFFFFFFD4 >>2 → 0xFFFFFFF9

Deste valor selecionamos somente os 16 bits menos significativos. Ficando:

0xFFF9

Com base na Tabela 1, nos valores da instrução e nos valores calculados, esta tem o valor:

Opcode [31-26] | Rs [25-21] | Rt [20-16] | immediate [15-0]
000101 | 00100 | 00101 | 1111 1111 1111 1001

Execução da Instrução:

Com o valor de PC previamente definido e amostrado na entrada do banco de memórias, ná próxima borda de subida do clock, a execução se iniciará. Então o valor correspondente a instrução em linguagem de máquina sairá do banco de registradores, os 6 bits mais significativos serão direcionados, o campo opcode, serão direcionados a Unidade de Controle. Está ira definir os mesmos valores de beq, 1 para DvC e UalOp0 e 0 para UALFonte, EscReg, LerMem, EscMem, Jump e UalOp1. Além de 1 para o adicional Nsinal. As demais saídas não interferem na execução do código independente de sua saída.

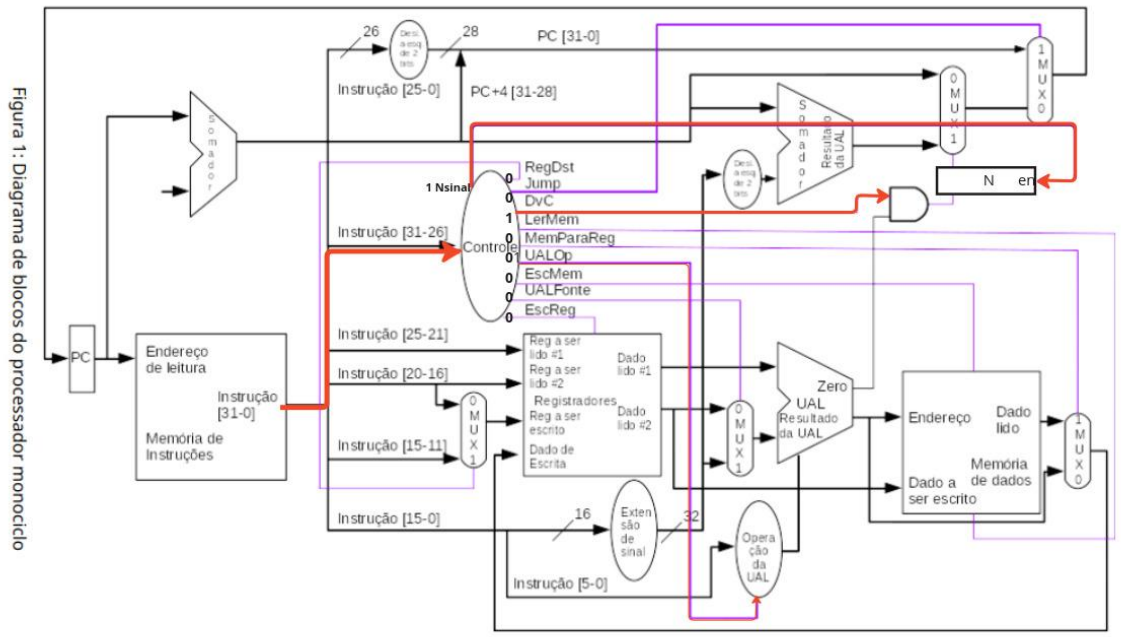


Figura 1: Diagrama de blocos do processador monociclo

Opcode [31-26] | Rs [25-21] | Rt [20-16] | immediate [15-0]
000101 | 00100 | 00101 | 1111 1111 1111 1001

miro

Os bits 21 à 25, representante do campo rs, são direcionados a entrada de leitura #1 no banco de registradores, então o valor correspondente ao endereço do registrador rs, $a0, é direcionado a uma das entradas da Unidade de Lógica Aritmética.

Concomitantemente, os bits 16 à 25, representante de rt, é direcionado a entrada de leitura #2 do banco de registradores, então o valor correspondente ao endereço do registrador rt, $a1, é direcionado ao multiplexador que precede a Unidade de Lógica aritmética na entrada 0, como o sinal que controla este multiplexador, UALFonte está setado para 0, este valor segue para a segunda entrada da unidade de lógica aritmética.

Ambos campos rs e rt são direcionados a outras entradas relacionados a escrita e leitura em memória e no banco de registradores, porém, as permissões de

alteração foram previamente negadas pela Unidade de Controle, impedindo a alteração dos dados.

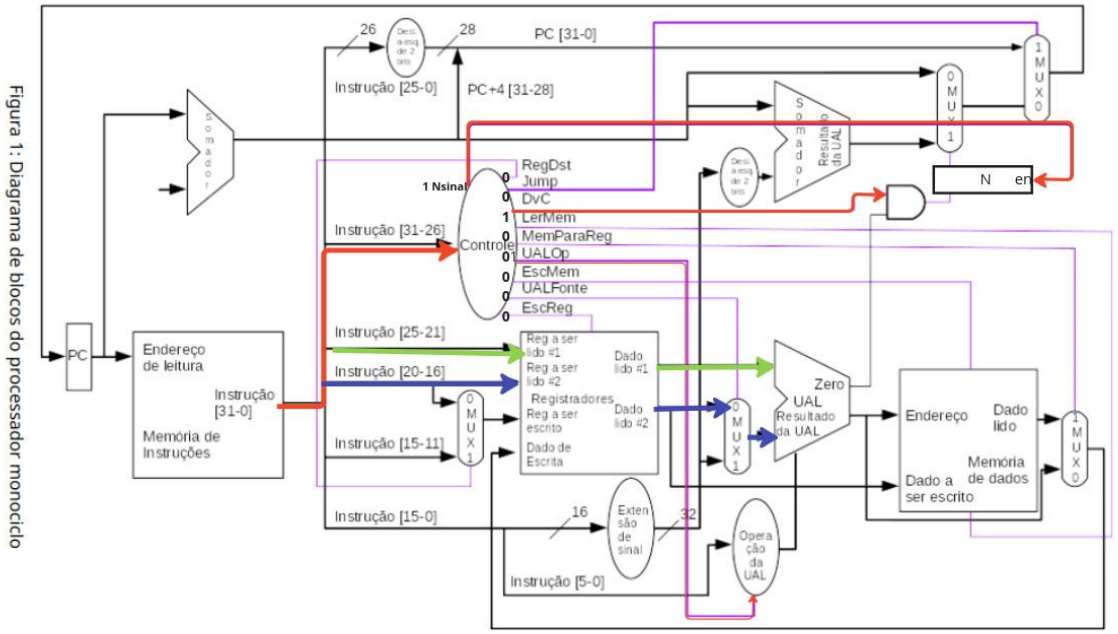


Figura 1: Diagrama de blocos do processador monociclo

Opcode [31-26] | Rs [25-21] | Rt [20-16] | immediate [15-0]
000101 | 00100 | 00101 | 1111 1111 1111 1001

miro

Concomitante a isto, os bits 0 a 15, representante do campo imediato são direcionados a um extensor de sinal para 32 (estende o número de bits, mas não altera o valor numérico) e então é direcionado a um deslocador para a esquerda de 2 bits (multiplicação por 4), para por fim ser direcionado ao somador, parte do campo imediato também é direcionado a entrada função do seletor de operação da Unidade de Lógica aritmética, entretanto, neste caso, não interferirá na saída do seletor.

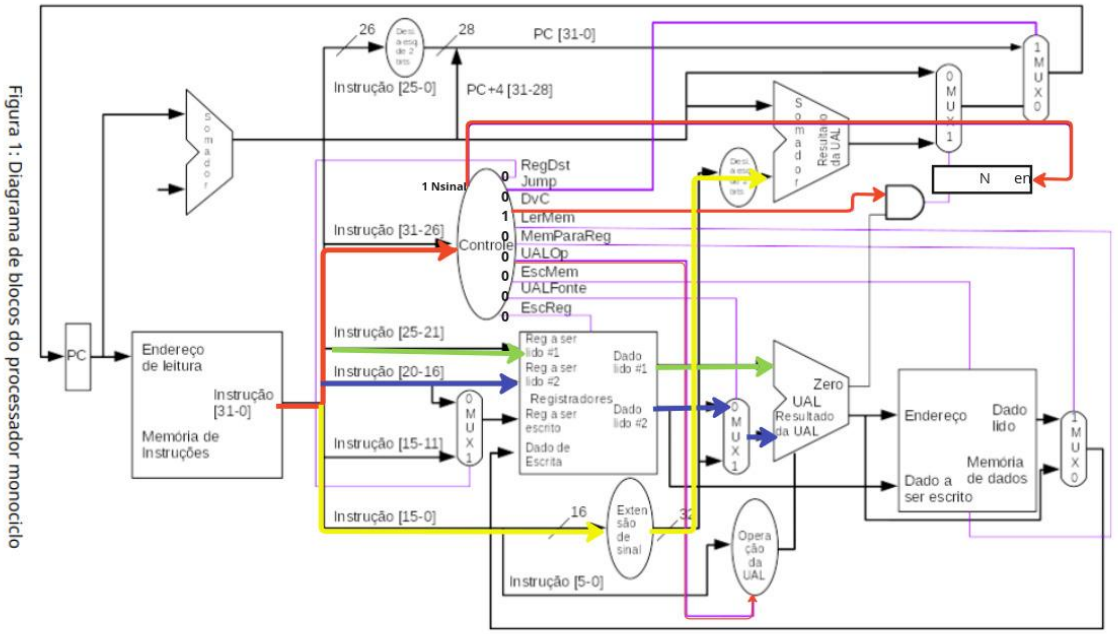


Figura 1: Diagrama de blocos do processador monociclo

Opcode [31-26] | Rs [25-21] | Rt [20-16] | immediate [15-0]
000101 | 00100 | 00101 | 1111 1111 1111 1001

O valor de PC é somado a 4 e então é direcionado a outra extremidade do somador onde está o valor imediato já alterado. Uma bifurcação de PC tem seus bits mais significativos concatenados com os bits 0 a 25 (passados por um deslocamento de 2 bits) entretanto este segmento não terá importância, ele será direcionado a um multiplexador controlado pela instrução da Unidade de Controle Jump, que setada para 0, interrompe sua propagação.

O valor que passará por este multiplexador é o resultado de outro multiplexador, em sua extremidade 0 o valor de PC+4 e na outra o valor PC+4 somado ao valor imediato com sinal estendido e multiplicado por 4.

O sinal enviado para o seletor de operação da Unidade de Lógica Aritmética(ULA) foi 01, equivalente à subtração de suas entradas, enviando à ULA o sinal 0110, a saída zero da ULA dependerá do valor de suas entradas serem iguais, o sinal zero passará por uma porta lógica and e será propagado, pois a mesma é definida pelo sinal DvC que está ativo, sucessivamente passará pelo inversor, que estará ativo devido ao sinal Nsinal.

A saída de Nsinal será a negação de do sinal zero da ULA, ou seja, caso os registradores rs e rt difiram, o sinal de zero será ativo(1), obviamente, caso sejam iguais, o sinal será inativo(0). Está saída por fim é direcionada ao multiplexador, que caso receba 0, propagará até o registrador de PC o valor de PC+4, caso receba 1, propagará ao registrador de PC o valor de PC+4+(Imediato<<2).

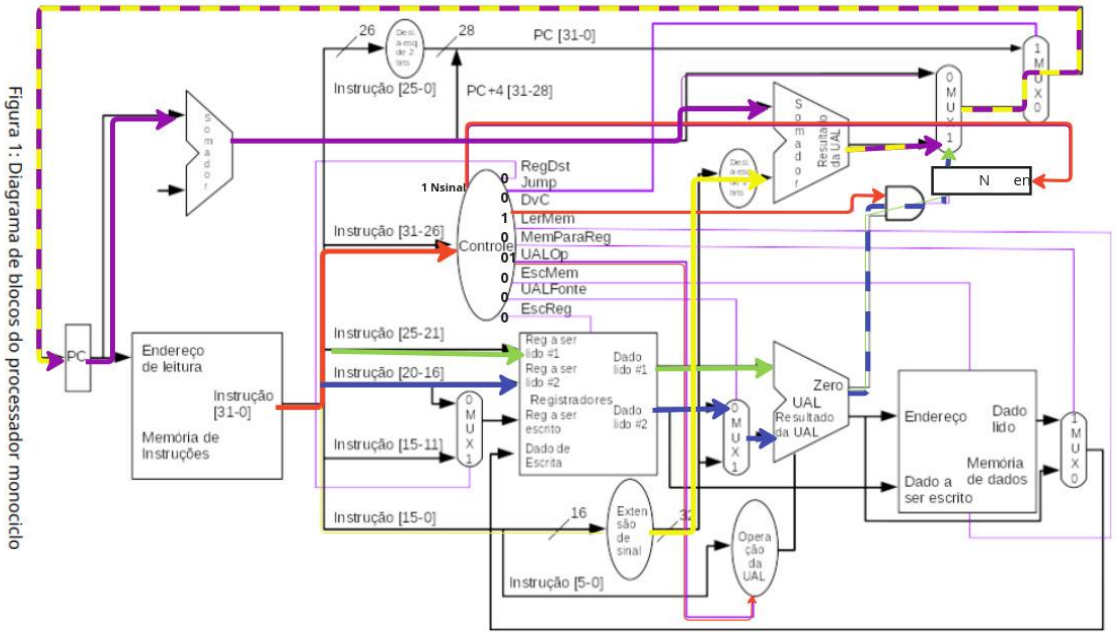


Figura 1: Diagrama de blocos do processador monociclo

Opcode [31-26] | Rs [25-21] | Rt [20-16] | immediate [15-0]
000101 | 00100 | 00101 | 1111 1111 1111 1001

7. Explique detalhadamente, usando as tabelas 1 e 2 e o diagrama de blocos do processador na figura 1, como a instrução j loop é executada pelo processador monociclo. O endereço desta instrução é 0x004000034 e loop é um rótulo para o endereço 0x00400014. Converta a instrução para linguagem de máquina, apresentando os campos. Apresente na figura os sinais de controle. Escreva um texto explicando como a instrução é executada.

Linguagem de Máquina:

A instrução j (store word) é uma instrução do tipo J, portanto é formada por

Opcode [31-26] | Immediate [25-0]

Com base na Tabela 1 e nos valores da instrução, esta tem o valor:

Opcode [31-26] | Immediate [25-0]
1010 11 | 00 0001 0000 0000 0000 0000 0101

Equivalente em hexadecimal a:

0xAC100005

Execução da Instrução:

Com o valor de PC em 0x004000034, a próxima borda de subida do clock, o valor correspondente ao endereço de memória de instrução em PC é direcionado ao resto do circuíto, seus 6 bits mais significativos são direcionados a Unidade de Controle. Esta então, a partir dos valores que identificam a instrução j (jump), a Unidade de Controle enviam o sinal 1 para a saída jump e 0 para LerMem(ler memória), EscReg (escrever no registrador) e EscMem (escrever na memória). As demais saídas da Unidade de Controle não interferiram na execução da instrução, independente do valor definido.

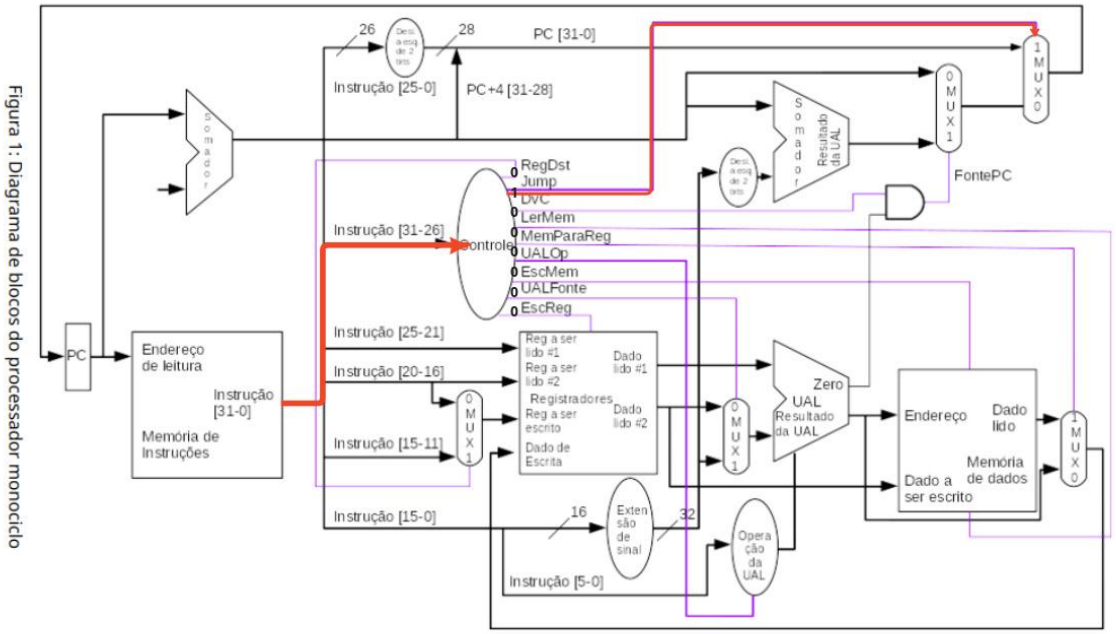
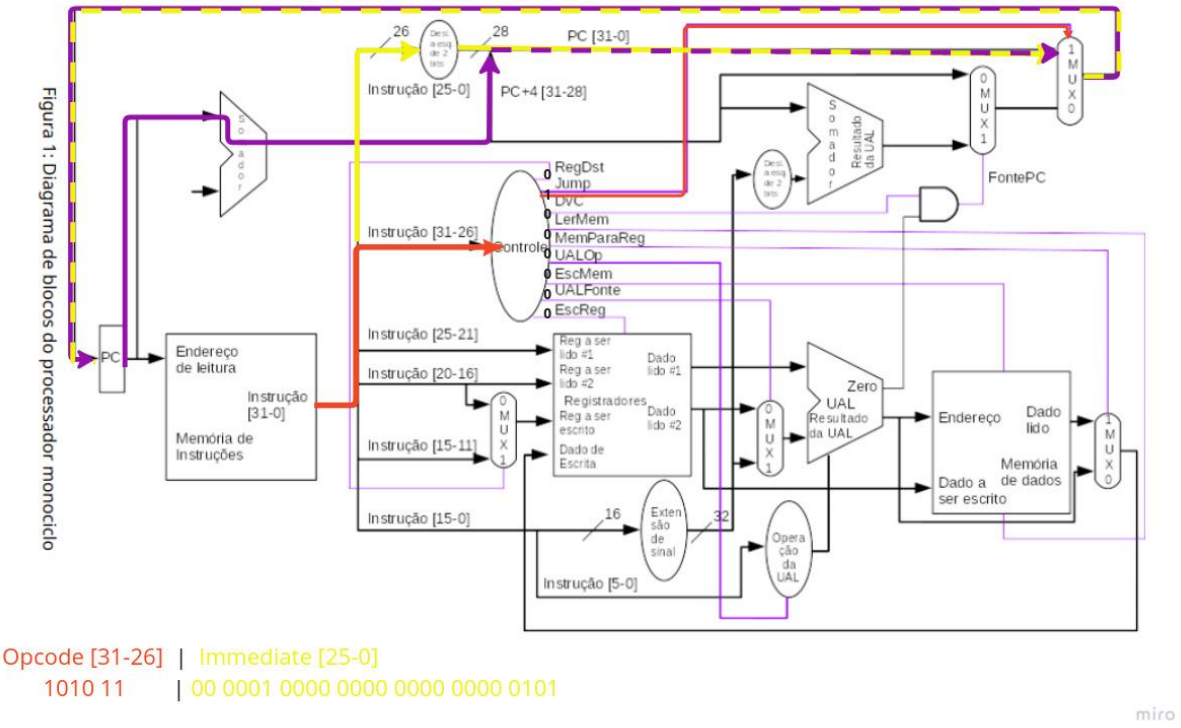


Figura 1: Diagrama de blocos do processador monociclo

Opcode [31-26] | Immediate [25-0]
1010 11 | 00 0001 0000 0000 0000 0000 0101

Os bits de 0 a 25, que representam a porção imediata da instrução, são enviados a um *shifter* que o desloca 2 bits para a esquerda, tornando assim um valor de 28 bits.

Concomitante a isto, o valor de PC é somado com 4, posteriormente os 4 bits mais significativos de PC são concatenados à esquerda do valor imediato de 28 bits, formando assim um valor de 32 bits que representa o novo valor de PC. Uma cópia de PC+4, antes da concatenação, é enviada para uma sequência de dois multiplexadores. Este último multiplexador recebe na entrada 1 o novo valor de PC, e como o seletor deste multiplexador é definido pela saída Jump da Unidade de Controle, o novo valor de PC segue para o registrador de PC.



Referências:

Moodle Presencial - UFSM: Acesso ao site. Disponível em: <https://ead06.proj.ufsm.br/pluginfile.php/4239330/mod_folder/content/0/textos/numeros_ponto_flutuante.pdf?forcedownload=1>. Acesso em: 26 jan. 2023.

Moodle Presencial - UFSM: Acesso ao site. Disponível em: <https://ead06.proj.ufsm.br/pluginfile.php/4239330/mod_folder/content/0/textos/algoritmoMultiplicacaoDivisao.pdf?forcedownload=1>. Acesso em: 26 jan. 2023.

Moodle Presencial - UFSM: Acesso ao site. Disponível em: <https://ead06.proj.ufsm.br/pluginfile.php/4239330/mod_folder/content/0/slides/4.3%20processador%20monociclo2.pdf?forcedownload=1>. Acesso em: 26 jan. 2023.

PATTERSON, D. A. **Computer organization and design.** San Francisco: Elsevier Science & Technology, 2013.