

Caminho de Dados de Ciclo Único e Controle MIPS

Propósito

Saber como implementar um subconjunto de instruções da arquitetura MIPS.

Método

Implementar o caminho de dados para um subconjunto da arquitetura do conjunto de instruções MIPS descrito no livro texto usando o simulador Logisim em quatro passos (10,0):

1. Entender o Logisim, bibliotecas internas e externas e subcircuitos.
2. (4,0) Implementar o caminho de dados utilizando o controle principal e da ALU manualmente para as instruções **add**, **sub**, **and**, **or**, **slt**, **lw**, **sw** e **beq**. Testar o controle manual utilizando o programa assembly MIPS teste_1.mem.
3. (3,0) Implementar o controle principal e da ALU utilizando lógica combinacional para as instruções **add**, **sub**, **and**, **or**, **slt**, **lw**, **sw** e **beq**. Testar o controle implementado utilizando o programa assembly MIPS teste_1.mem.
4. (3,0) Acrescentar ao controle principal utilizando lógica combinacional as instruções **addi** e **jump**. Testar o controle implementado utilizando o programa assembly MIPS teste_2.mem.

Agradecimentos: Este trabalho é uma versão de uma atividade desenvolvida por Thomas M. Parques e Chris Nevison na Colgate University e de uma atividade desenvolvida por Hakin Weatherspoon na Cornell University.

1. Entender o Logisim, bibliotecas internas e externas e subcircuitos

Baixe a versão estável do Logisim (2.7.2) desenvolvida pela Universidade de Cornell¹.

Crie uma nova pasta chamada mips_datapath. Baixe todos os arquivos necessários. Antes de continuar, certifique-se que esses arquivos estão na pasta mips_datapath: datapath.circ, control.circ, cpu32.circ, misc32.circ, cs1410.jar, teste_1.asm e teste_1.mem.

Abra o projeto datapath.circ no Logisim, Figura 1.

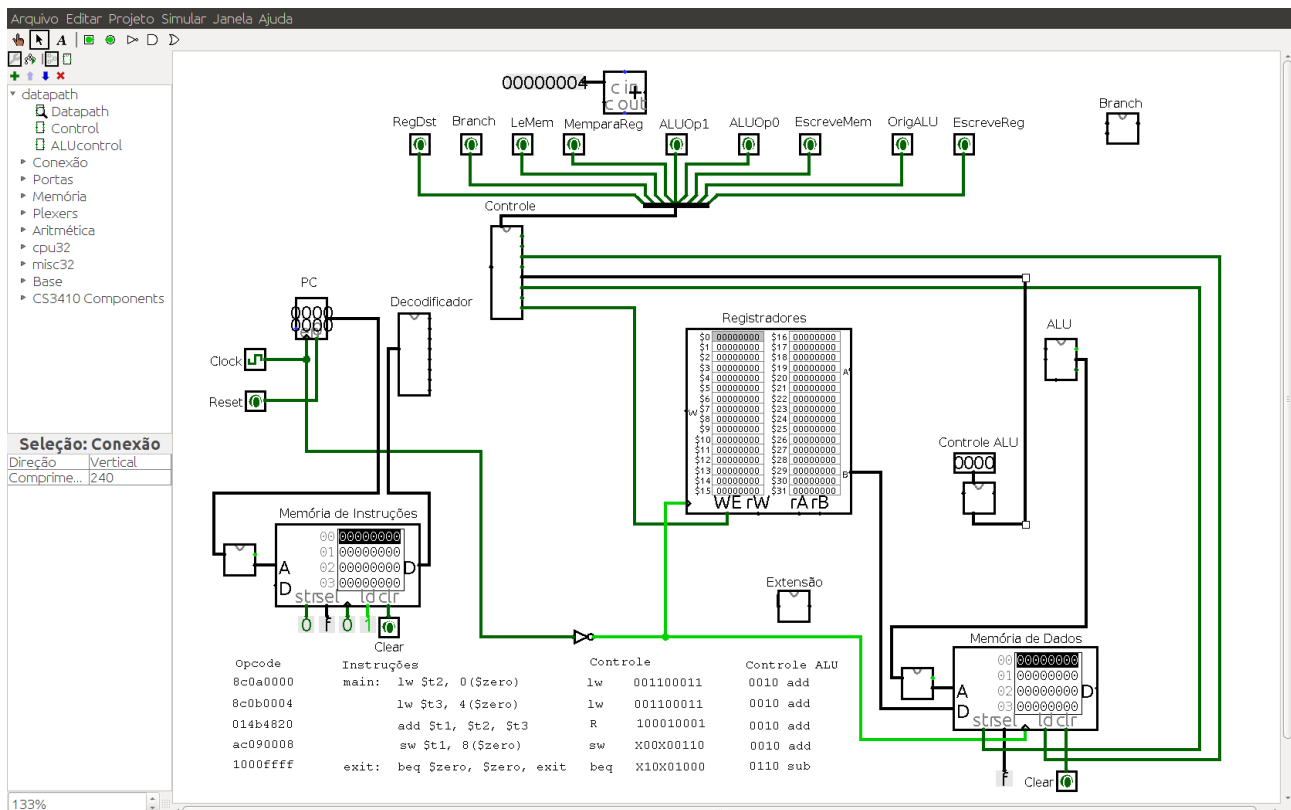


Figura 1 Projeto inicial datapath.circ.

O projeto datapath.circ é muito semelhante à Figura 2. Neste momento, o caminho de dados da Figura 1 não está completo.

1 <http://www.cs.cornell.edu/courses/cs3410/2015sp/logisim-2.7.2-cs3410-20140215.jar>

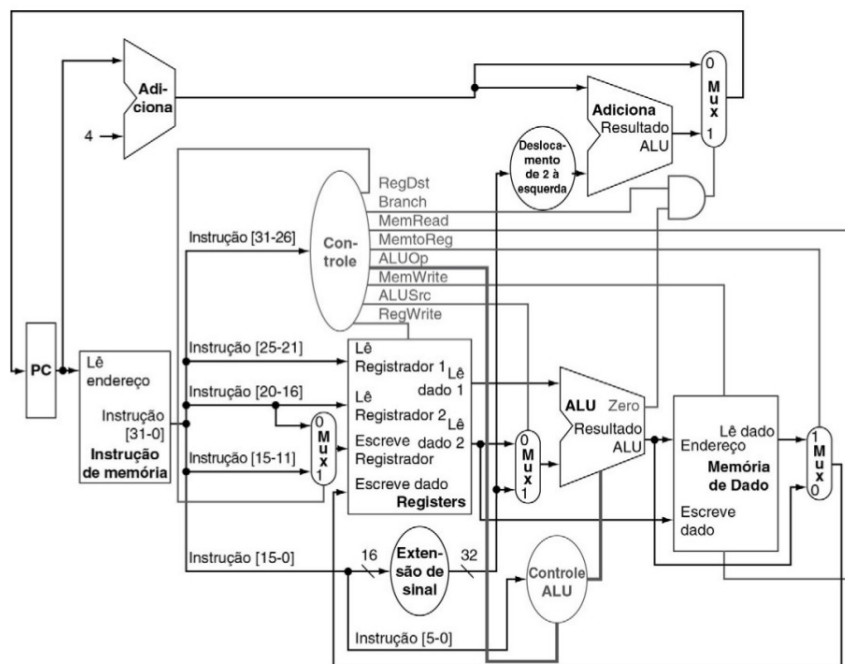


Figura 2 O Caminho de dados simples com a unidade de controle (Figura 4.17 do livro de texto).

O Logisim vem com algumas bibliotecas internas de componentes. Uma destas bibliotecas é a biblioteca memória. A partir desta biblioteca o componente registrador é utilizado para o contador de programa (PC).

Os componentes externos do Logisim utilizados no datapath.circ são mostrados na Tabela 1.

Tabela 1: Componentes utilizados das bibliotecas externas do Logisim.

Bibliotecas externas	cpu32	misc32	CS3410
Componentes utilizados	InstructionDecode SignExtend ALU BranchAddress JumpAddress ¹ Control ALUcontrol	WordAddress	Register File MIPS RAM

No circuito do caminho de dados também é possível ver uma série de caixas quadradas. Estas caixas representam subcircuitos². No lado esquerdo da memória de instruções existe uma caixa desta, clique o botão direito do mouse nesta caixa. A partir do menu pop-up, escolha "Ver WordAddress", Figura 3.

¹ Utilizado no item "4. Acrescentar ao controle principal utilizando lógica combinacional as instruções addi e jump"

² <http://www.cburch.com/logisim/docs/2.7/pt/html/guide/subcirc/index.html>

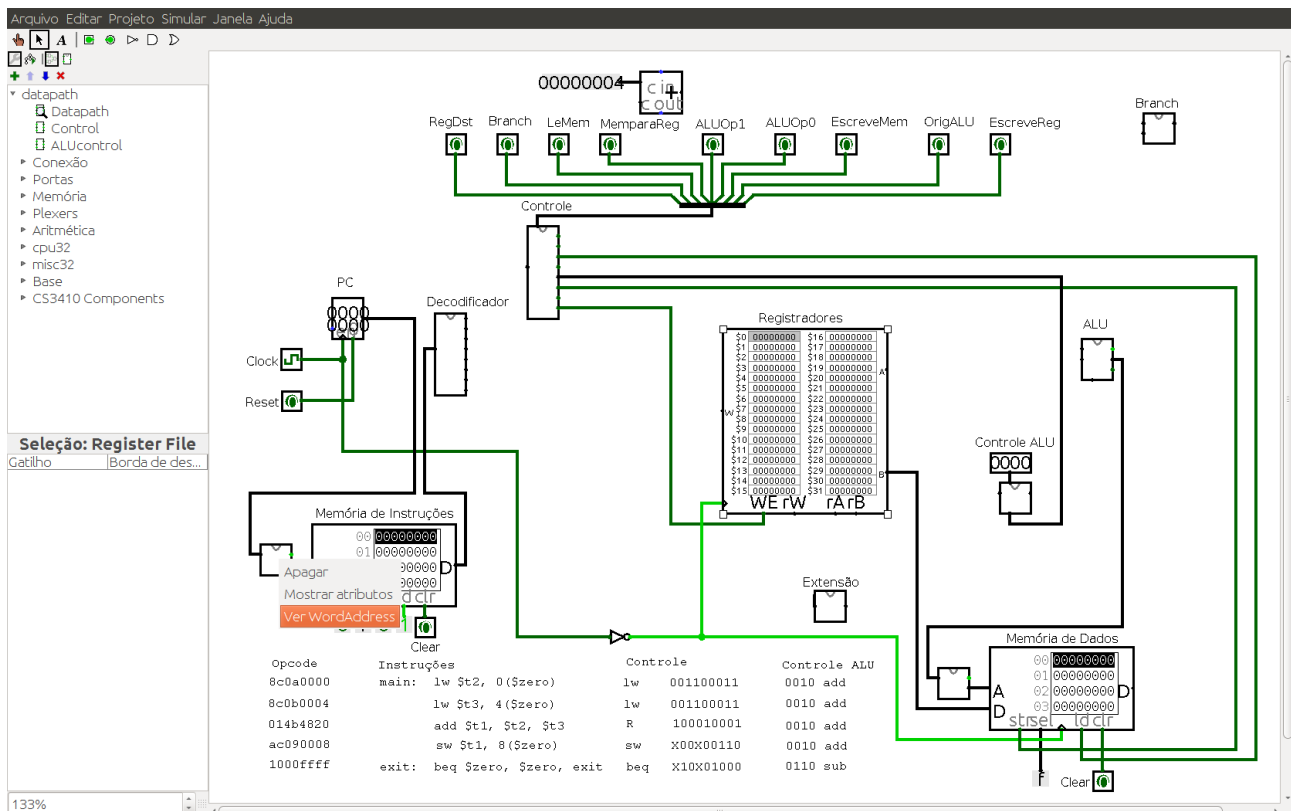


Figura 3 Acesso ao subcircuito WordAddress.

A Figura 4 mostra o subcircuito WordAddress usado para traduzir um endereço de 32 bits para um endereço de word de 8 bits.

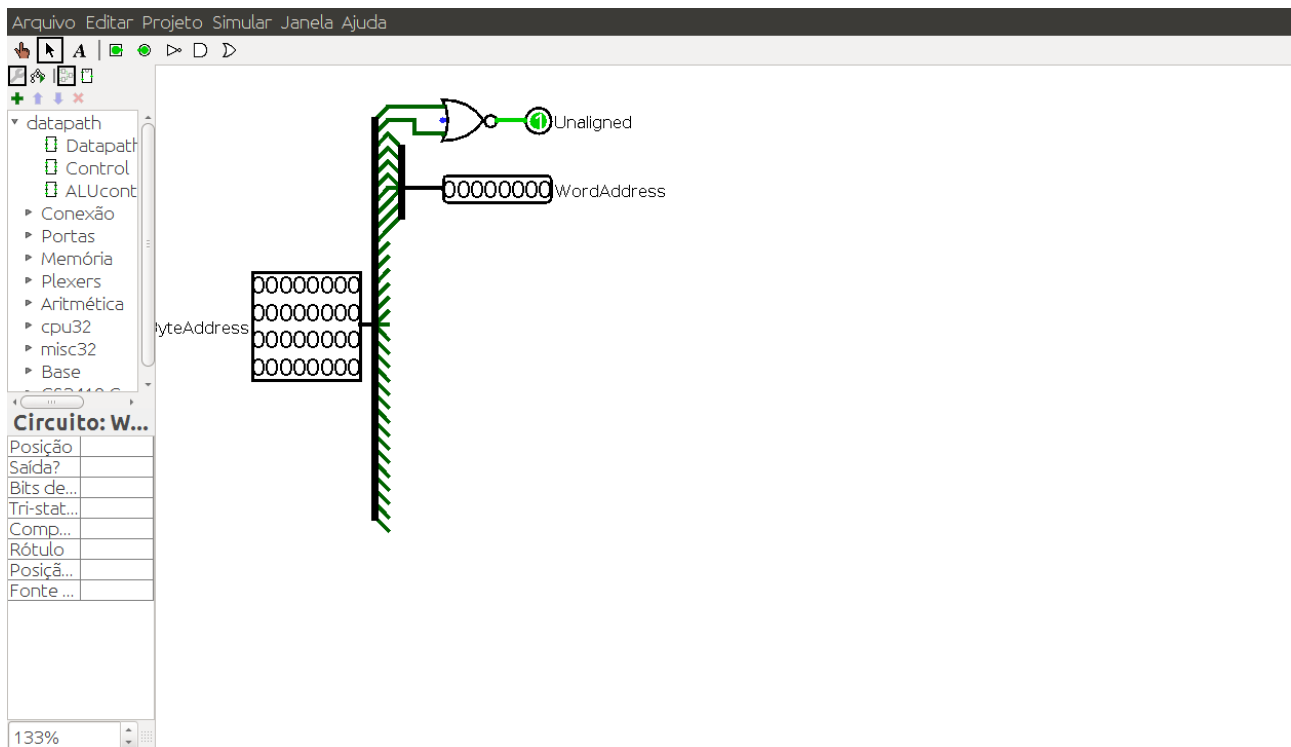


Figura 4 Subcircuito WordAddress.

Esses nove sinais de controle, sete da Tabela 2 e dois para OpALU, podem agora ser definidos baseados nos seis sinais de entrada da unidade de controle, que são os bits de opcode 31 a 26.

Tabela 2: Efeito de cada um dos sete sinais de controle.

Nome do sinal	Efeito quando inativo (0)	Efeito quando ativo (1)
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para entrada Registrador para escrita vem do campo rd (bits 15:11).
Branch	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio, caso a saída Zero da ALU, usada para comparação de igualdade, for verdadeira (1).
LeMem (MemRead)	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
MemparaReg (MemtoReg)	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.
EscreveMem (MemWrite)	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
OrigALU (ALUSrc)	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados de leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
EscreveReg (RegWrite)	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.

Na Tabela 3 é possível ver as configurações necessárias para instruções do formato **R**, **lw**, **sw** e **beq**. Olhe para o circuito no Logisim e consulte esta tabela. Tente entender os valores mostrados na tabela e que escolhas no caminho de dados estes valores representam.

Tabela 3: A definição das linhas de controle é determinada pelos campos opcode da instrução formato R, lw, sw e beq.

Instrução	RegDst	Branch	LeMem (MemRead)	MemparaReg (MemtoReg)	ALUOp1	ALUOp0	EscreveMem (MemWrite)	OrigALU (ALUSrc)	EscreveReg (RegWrite)
formato R	1	0	0	0	1	0	0	0	1
lw	0	0	1	1	0	0	0	1	1
sw	X	0	0	X	0	0	1	1	0
beq	X	1	0	X	0	1	0	0	0

Você não precisa construir uma unidade de controle funcional neste momento. No entanto, você deve compreender quais sinais de saída da unidade de controle são necessários para o trabalho no caminho de dados para as instruções **formato R**, **lw**, **sw** e **beq**.

Para controlar o caminho de dados, vamos começar emulando manualmente a unidade de controle, isto é, controlando manualmente a saída da unidade de controle. Neste passo o controle é manual e não terá o fio de ligação entre o componente Decodificador e o componente Controle.

Na parte superior do Controle ligue os pinos de entrada de 1 bit: RegDst, ALUOp1 e EscreveReg, Figura 9. Agora você pode usar esta entrada para controlar manualmente a unidade de controle para uma instrução do formato R.

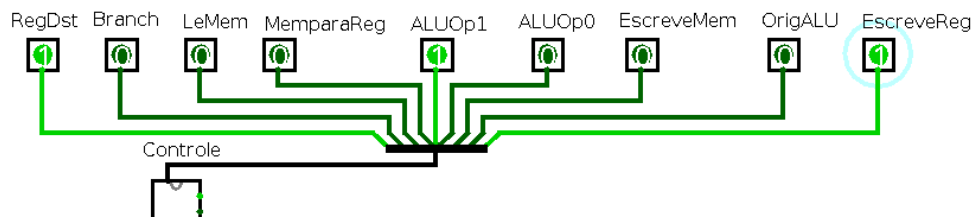


Figura 9 Pinos de entrada para controle manual.

Abra o circuito da unidade de controle, clique o botão direito do mouse no Controle. A partir do menu pop-up, escolha "Ver Control" e será mostrado o subcircuito da unidade de controle, Figura 10.

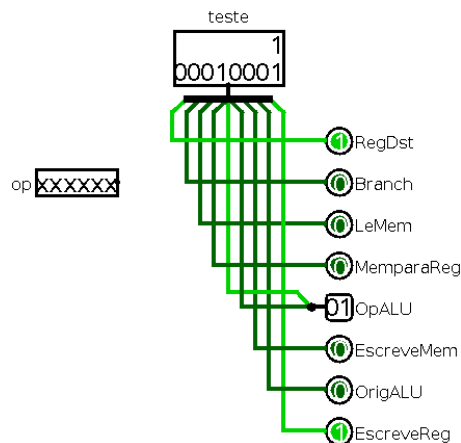


Figura 10 Subcircuito Control.

Todas as linhas de saída da unidade de controle, exceto o OpALU, são de 1 bit.

No circuito do Logisim, clique com o botão direito sobre o ALU e escolha "Ver ALU", Figura 11.

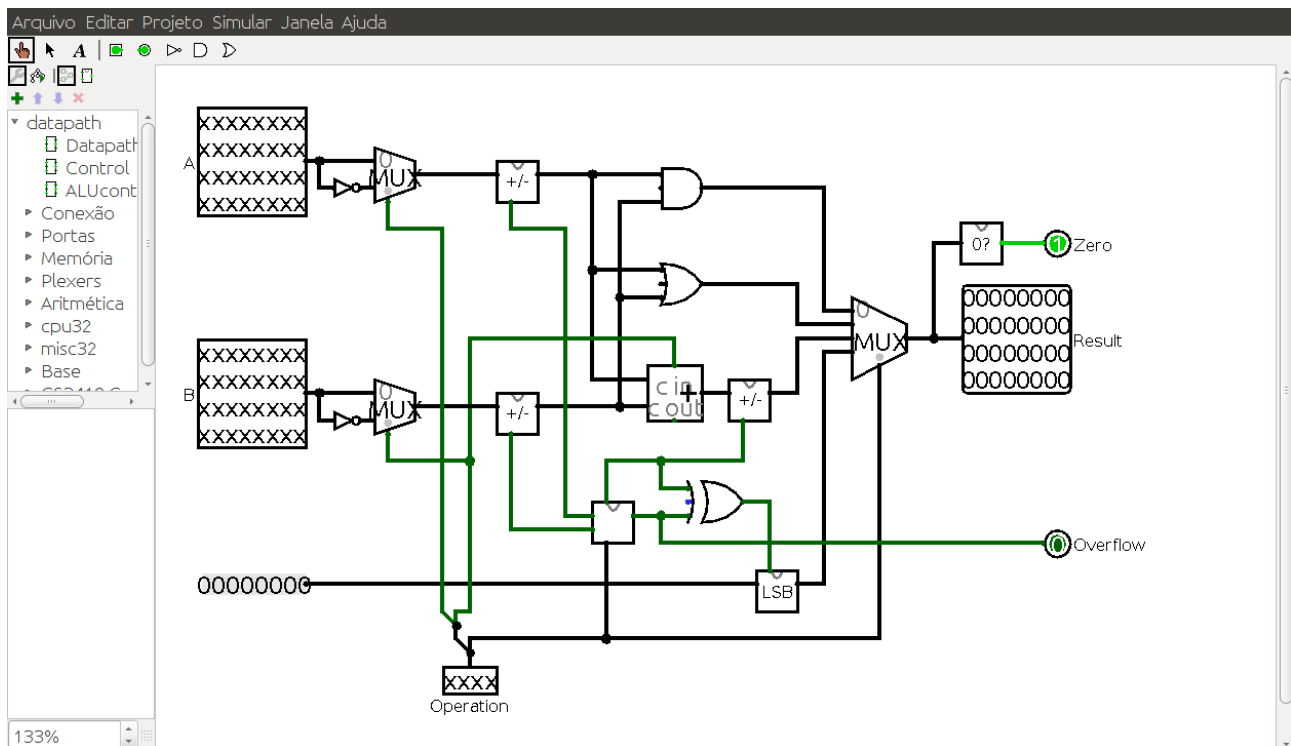


Figura 11 Subcircuito ALU.

Certifique-se de entender como a entrada Operation de 4 bits é usada para controlar a função da ALU de acordo com a Tabela 4.

Tabela 4: A forma como os bits de controle da ALU são definidos.

Opcode da instrução	ALUOp1	ALUOp0	Operação da instrução	Campo funct	Ação da ALU desejada	Entrada do controle da ALU
lw	0	0	load word	XXXXXX	Add	0010
sw	0	0	Store word	XXXXXX	Add	0010
Beq	0	1	Branch equal	XXXXXX	Sub	0110
Tipo R	1	0	Add	100000	Add	0010
Tipo R	1	0	Sub	100010	Sub	0110
Tipo R	1	0	AND	100100	AND	0000
Tipo R	1	0	OR	100101	OR	0001
Tipo R	1	0	Slt	101010	Set on less than	0111

De forma semelhante como para a unidade de controle, na parte superior do Controle ALU, no pino de entrada de 4 bit escolha a entrada de binária 0010. Agora você pode usar esta entrada para controlar manualmente a ALU para uma soma, Figura 12.

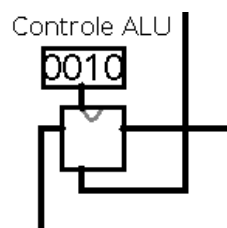


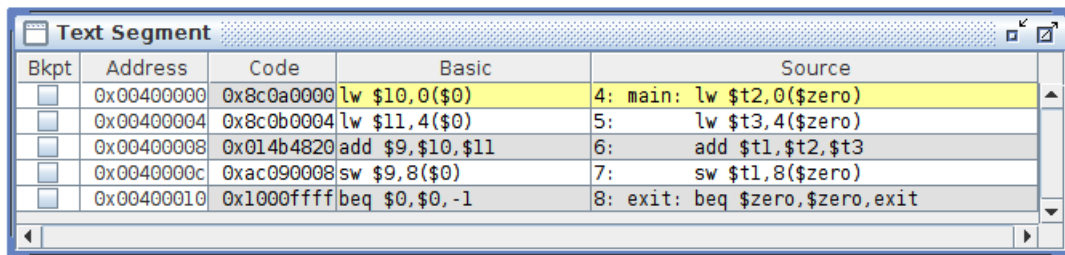
Figura 12 Controle manual da ALU.

2.1 Testar o controle manual

Após adicionar fios e possivelmente algum hardware (portas, multiplexadores, etc.) para fazer o caminho de dados executar as seguintes instruções MIPS: **add**, **sub**, **and**, **or**, **slt**, **lw**, **sw** e **beq** testar o caminho de dados utilizando o controle manual. Você deve estudar este programa assembly MIPS teste_1.asm:

```
main: lw $t2, 0($zero)
      lw $t3, 4($zero)
      add $t1, $t2, $t3
      sw $t1, 8($zero)
exit: beq $zero, $zero, exit
```

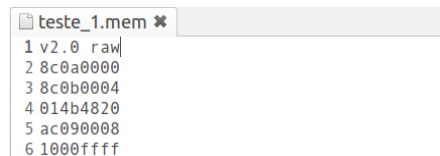
Utilize o simulador MARS para traduzir o programa assembly para o código de máquina, Figura 13.



Bkpt	Address	Code	Basic	Source
	0x00400000	0x8c0a0000	lw \$t0,0(\$0)	4: main: lw \$t2,0(\$zero)
	0x00400004	0x8c0b0004	lw \$t1,4(\$0)	5: lw \$t3,4(\$zero)
	0x00400008	0x014b4820	add \$9,\$t0,\$t1	6: add \$t1,\$t2,\$t3
	0x0040000c	0xac090008	sw \$9,8(\$0)	7: sw \$t1,8(\$zero)
	0x00400010	0x1000ffff	beq \$0,\$0,-1	8: exit: beq \$zero,\$zero,exit

Figura 13 Tradução do programa teste_1.asm.

Salve as instruções de máquinas em hexadecimal para o arquivo teste_1.mem. Uma instrução em cada linha. Sem prefixo 0x. A primeira linha deve conter “v2.0 raw”, Figura 14.



```
teste_1.mem *
1 v2.0 raw
2 8c0a0000
3 8c0b0004
4 014b4820
5 ac090008
6 1000ffff
```

Figura 14 Arquivo teste_1.mem.

Clique o botão direito do mouse sobre a memória RAM de instruções. Carregue a instrução de máquina do arquivo teste_1.mem para a memória de instruções, Figura 15.

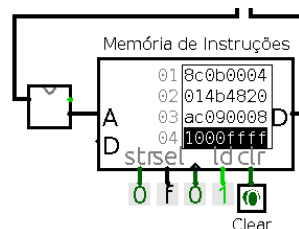


Figura 15 Memória de instruções.

Clique com o botão esquerdo no mouse sobre o endereço 00 da Memória de Dados e digite o valor 1. Clique com o botão esquerdo no mouse sobre o endereço 01 da Memória de Dados e digite o valor 2, Figura 16.

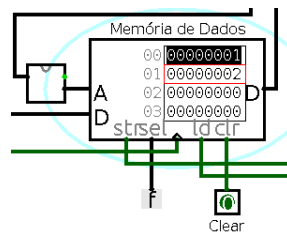


Figura 16 Memória de dados.

Antes de começar a testar o seu caminho de dados usando a pequena sequência de instruções e dados carregados na memória de instrução e dados, você deve ser muito cuidadoso:

- Verifique se você colocou as instruções de máquina de teste_1.mem na memória de instruções.
- Verifique se você colocou o dado 1 no endereço 00 e o dado 2 no endereço 01 da memória de dados.
- Verifique se o clock está alto (verde).
- Reset o PC e os registradores.

A primeira instrução do programa é a instrução:

lw \$t2, 0(\$zero)

Opcode: 08ca000

Controle: 001100011

Controle ALU: 0010

Manualmente, defina os valores da saída da unidade de controle e do controle ALU(use a Tabela 3 e a Tabela 4).

Para esta instrução, a ALU é utilizada para calcular o endereço da memória de dados.

Para que os dados sejam armazenados em um registrador ou memória, o sinal de clock para estas unidades devem mudar a partir de baixo para alto (sensível na borda de subida). Lembre-se que você começou com o clock de alto:

Clique no clock para mudar de alto (verde), Figura 17, e para baixo (preto), Figura 18.

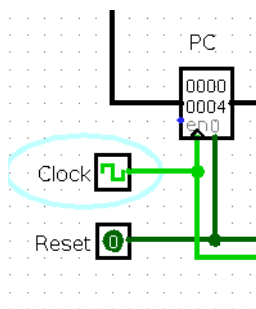


Figura 17 Clock alto.

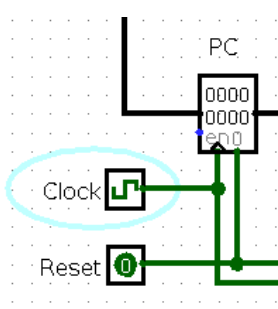


Figura 18 Clock baixo.

No circuito você pode ver que o sinal de clock enviado aos registradores (e para a memória de dados) é invertido, Figura 1.

A segunda instrução do programa é a instrução:

lw \$t3, 4(\$zero)

Opcode: 08cb004

Controle: 001100011

Controle ALU: 0010

Manualmente, defina os valores da saída da unidade de controle e do controle ALU(use a Tabela 3 e a Tabela 4).

Para esta instrução, a ALU é utilizada para calcular o endereço da memória de dados.

A Figura 19 mostra os registradores após a execução das duas primeiras instruções:
\$t2 (\$10) = 1

\$t3 (\$11) = 2

Registradores

\$0	00000000	\$16	00000000
\$1	00000000	\$17	00000000
\$2	00000000	\$18	00000000
\$3	00000000	\$19	00000000
\$4	00000000	\$20	00000000
\$5	00000000	\$21	00000000
\$6	00000000	\$22	00000000
\$7	00000000	\$23	00000000
\$8	00000000	\$24	00000000
\$9	00000000	\$25	00000000
\$10	00000001	\$26	00000000
\$11	00000002	\$27	00000000
\$12	00000000	\$28	00000000
\$13	00000000	\$29	00000000
\$14	00000000	\$30	00000000
\$15	00000000	\$31	00000000

WE rW rArB

Figura 19 Registradores.

A terceira instrução do programa é a instrução:

add \$t1, \$t2, \$t3

Opcode: 014b4820

Controle: 100010001

Controle ALU: 0010

Manualmente, defina os valores da saída da unidade de controle e do controle ALU(use a Tabela 3 e a Tabela 4).

Para esta instrução, a ALU é utilizada para calcular a soma do conteúdo dos registradores \$t2 e \$t3.

A Figura 20 mostra os registradores após a execução da terceira instrução:

\$t1 = \$t2 + \$t3

\$t1 (\$9) = 3

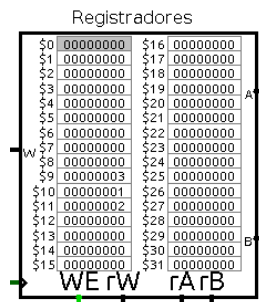


Figura 20 Registradores.

A quarta instrução do programa é a instrução:

sw \$t1, 8(\$zero)

Opcode: 0c090008

Controle: 000000110

Controle ALU: 0010

Manualmente, defina os valores da saída da unidade de controle e do controle ALU(use a Tabela 3 e a Tabela 4).

Para esta instrução, a ALU é utilizada para calcular o endereço da memória de dados.

A Figura 21 mostra os registradores após a execução da quarta instrução:



Figura 21 Registradores.

A última instrução do programa é a instrução:

beq \$ zero, \$ zero, exit

Opcode: 1000ffff

Controle: 010001000

Controle ALU: 0110

Manualmente, defina os valores da saída da unidade de controle 0,25e do controle ALU(use a Tabela 3 e a Tabela 4).

Para comparar os dois registradores a ALU pode ser utilizada. A ALU realiza a subtração dos valores de dados dos dois registradores.

A saída Zero da ALU será definida para 1 quando o resultado da ALU é 0 ($rs == rt$). Este sinal é utilizado para decidir se vamos desviar para ($PC = (PC + 4) + Offset$) ou continuar a executar a próxima instrução ($PC = PC + 4$).

O montador produz um desvio que é codificado nas instruções de desvio como uma constante imediata de 16 bits. Esta constante é definida como o número de instruções para saltar se a condição de desvio é verdadeira.

Verifique se o seu caminho de dados pode fazer um beq (branch equal) corretamente. Neste caso o desvio é para a própria instrução beq.

3. Implementar o controle principal e da ALU utilizando lógica combinacional para as instruções add, sub, and, or, slt, lw, sw e beq

Crie uma cópia pasta mips_datapath (que contém a implementação do controle manual) com o nome mips_dapath_control. Utilize esta nova pasta para implementar a unidade de controle da ALU e da unidade de controle principal.

Para implementar a unidade de controle da ALU e a unidade de controle principal utilize como apoio as informações contida no Apêndice C Mapeando o Controle no Hardware, item C.2 Implementando unidade de controle combinacionais.

3.1 Testar o controle implementado

Teste o caminho de dados com o controle principal e da ALU implementados com o programa assembly MIPS teste_1.asm. Verifique se obtém os mesmos resultados do controle manual.

4. Acrescentar ao controle principal utilizando lógica combinacional as instruções addi e jump

Na Tabela 5 é possível ver as configurações necessárias para instruções do formato **R**, **lw**, **sw** e **beq**. Olhe para o circuito no Logisim e consulte esta tabela. Tente entender os valores mostrados na tabela e que escolhas no caminho de dados estes valores representam e insira as configuração dos sinais para as instruções addi (adição imediata) e jump (desvio incondicional), conforme Tabela 5 e Figura 22.

Tabela 5: A definição das linhas de controle é determinada pelos campos opcode da instrução formato R, lw, sw, beq e addi.

Instrução	RegDst	Branch	LeMem	MemparaReg	ALUOp1	ALUOp0	EscreveMem	OrigALU	EscreveReg
formato R	1	0	0	0	1	0	0	0	1
lw	0	0	1	1	0	0	0	1	1
sw	X	0	0	X	0	0	1	1	0
beq	X	1	0	X	0	1	0	0	0
addi	?	?	?	?	?	?	?	?	?
jump	?	?	?	?	?	?	?	?	?

Crie uma cópia pasta mips_datapath_control (implementado no item 3) com o nome mips_dapath_control_extend. Utilize esta nova pasta para implementar a unidade de controle principal com as novas instruções.

Para implementar a unidade de controle principal utilize como apoio as informações contida no Apêndice C Mapeando o Controle no Hardware, item C.2 Implementando unidade de controle combinacionais.

Para implementar o deslocamento de 2 à esquerda necessário para a instrução jump, Figura 22, utilize o componente JumpAddress da biblioteca externa cpu32.

4.1 Testar o controle implementado

Teste o caminho de dados com o controle principal com o programa assembly MIPS teste_2.asm, Figura 23.

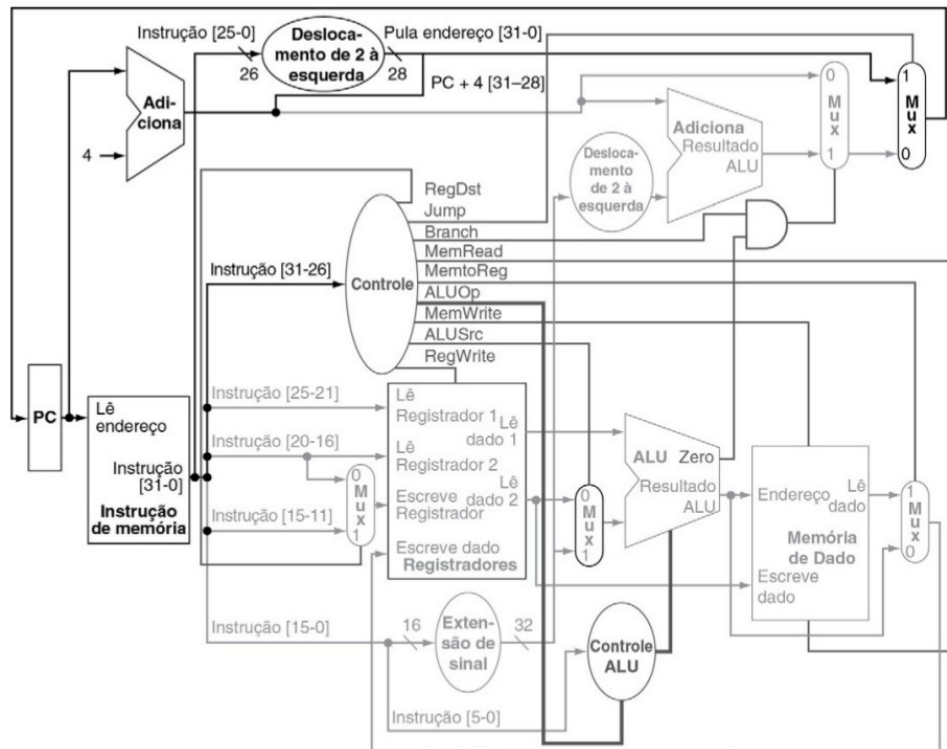


Figura 22 O controle e o caminho de dados simples - instrução jump (Figura 4.24 do livro texto).