

Implementando o caminho de dados para um subconjunto da arquitetura do conjunto de instruções MIPS

Guilherme Zago Canesin¹

Universidade Tecnológica Federal do Paraná – UTFPR

*COCIC – Coordenação do Curso de Bacharelado em Ciência da Computação
Campo Mourão, Paraná, Brasil*

¹guilhermecanesin@alunos.utfpr.edu.br

Resumo

Este trabalho é o resultado do estudo dos caminhos de dados de um controlador com arquitetura MIPS, tendo como itens implementados as instruções, **add**, **sub**, **and**, **or**, **sll**, **lw**, **sw** e **beq**, utilizando lógica combinacional e implementação da ALU manualmente, e como acréscimo implementar também as instruções **addi** e **jump** utilizando a lógica combinacional.

1. Introdução

O trabalho é dividido em três partes, uma com as unidades de controle sequencial, outra com o controle combinacional, e por fim o controle combinacional com a implementação dos comandos JUMP e ADDI. Os circuitos foram todos feitos pelo software logisim. Começando pelas unidades de controle sequencial, um bloco com 9 entradas comanda os valores dos operadores do controlador (**opALU0**, **opALU1**, **branch**, **escreveMem**, **leMem**, **escreveReg**, **MemparaReg**, **ALUsrc** **RegDst**). Sendo sequencial, o controle é feito manualmente sobre os valores de cada operador específico, podendo ser 0 ou 1.

A parte combinacional utiliza o controle que recebe os sinais de operação do decodificador, este caso não são necessários múltiplos estados. Através das instruções tipo R, **Iw**, **sw** e **beq**, ele gera os sinais para cada saída de dado automaticamente. Por fim a terceira parte do trabalho foi a implementação dos comandos **JUMP** e **ADDI** na versão com unidades de controle combinacional. O **JUMP** é uma instrução que altera o fluxo da sequência de um código assembly, para uma determinada linha, e o **ADDI** adiciona um *immediate* à um registrador, ou seja, um valor numérico. Nos próximos tópicos serão especificadas cada uma das partes.

2. Datapath manual

Durante o desenvolvimento desta etapa foram estabelecidos os canais de caminho do controlador para outras estruturas de controle do MIPS, iniciamos com a construção do **ALUcontrol** como visto na Figura 1, responsável pelo recebimento dos sinais opAlu do controlador, realizar o processamento combinacional e repassar para ALU as operações.

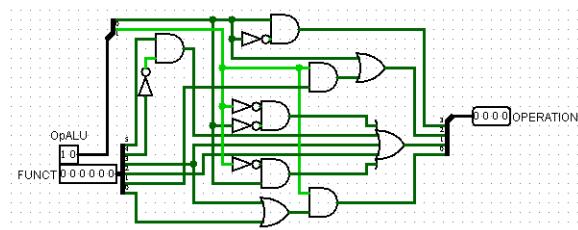


Figura 1: ALUcontrol

Depois de estabelecido os caminhos internos da ALUcontrol foram realizadas as rotas do controlador para as unidades a serem controladas.

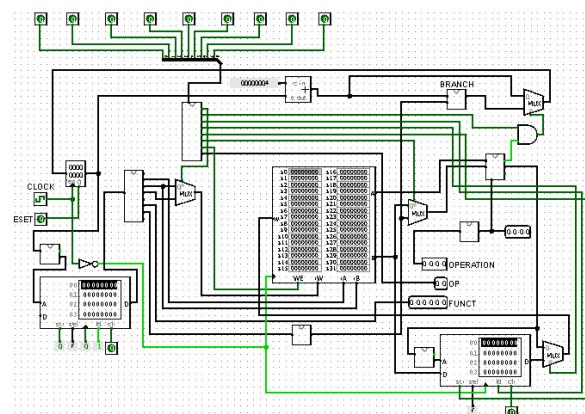


Figura 2: Datapath Manual

3. Estruturas de saída do controlador

4.

5. Saída regDst

6.

A saída regDst é responsável pela seleção do Mux dos registradores de Destino de acordo com a instrução.

2.1.1 Saída Branch

A saída Branch do controlador se liga ao MUX inicial do branch sobre uma porta AND para indicar se o programa deve seguir para o desvio ou não.

2.1.2 Saída LeMem

A saída responsável pela seleção do comando ld na memória de dados para Armazenamento nos registradores.

2.1.4 Saída MemparaReg

A saída responsável pela seleção do mux na saída da memória de dados para selecionar se a memória de dados ou a saída da ALU será salva em registrador.

2.1.5 Saída EscreveMem

A saída EscreveMem é responsável por selecionar a entrada do modulo de Memória de dados gravar a saída da ALU.

2.1.6 Saída OrigALU

A saída OrigAlu se conecta ao mux de seleção de entrada de dados, é responsável pela seleção de entrada de um valor imediato ou registrador.

2.1.7 Saída EscreveReg

Saída responsável pela seleção do método de entrada no banco de registradores.

creveReg, Mempara- Reg, ALUsrc e RegDst são geradas por um circuito. Este circuito recebe de entrada de dados 6 bits do decodificador, que a partir das instruções irá gerar a saída para cada operador.

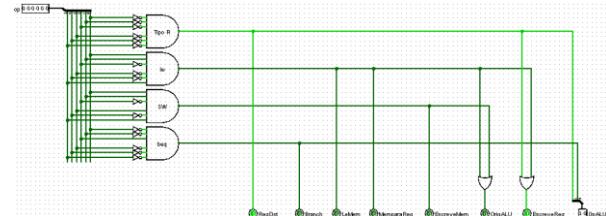


Figura 3: CONTROL

O Datapath do modelo com controle combinacional fica da seguinte forma.

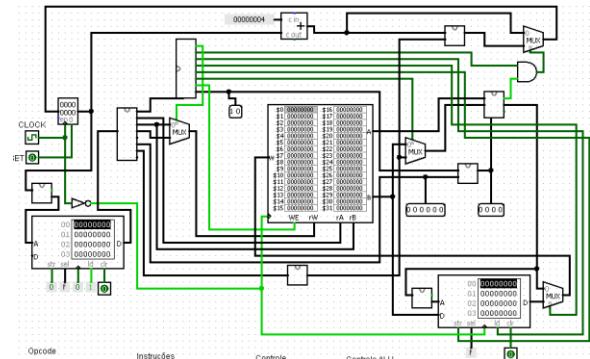


Figura 4: Datapath Combinacional

Agora o **Control** recebe dados do decodificador e as entradas para os operadores que antes ficava na parte superior do circuito, não são mais necessárias.

8. Datapath Combinacional com JUMP e ADDI.

Durante o desenvolvimento dessa etapa foram estabelecidas as rotas do controlador para Jump e Addi, adicionadas as estruturas de controle adicionais para utilização destas operações como visto na Figura 5.

7. Datapath Combinacional

O modelo combinacional é similar ao manual, porém sua particularidade é que os valores das saídas opALU0, opALU1, branch, escreveMem, leMem, es-

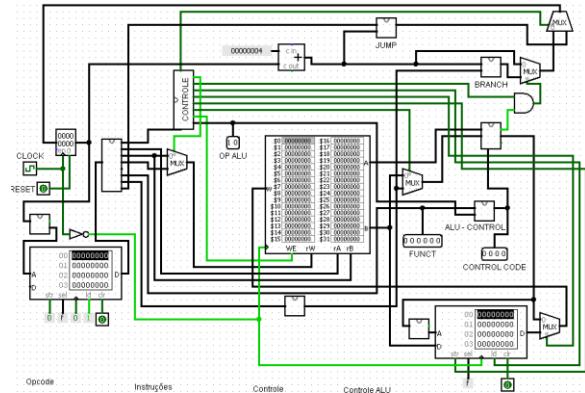


Figura 5: Datapath Combinacional com JUMP e ADDI.

9. 4.1. ADDI

Para implementação do AddI foi incluído no controlador uma porta para a operação como visto na Figura 6, a decodificação da instrução se dá pela tradução do opcode ADDI do mips que seria 001000 e ativação das estruturas de controle OrigALU para seleção do imediato e EscreveReg para escrita do registrador.

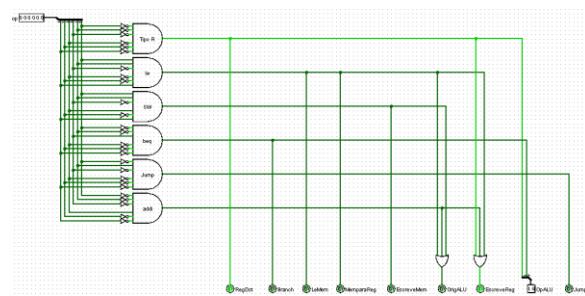


Figura 6: Control com ADDI e JUMP

4.2. JUMP

Para implementação da instrução Jump foi adicionado o JumpAddress para fazer o deslocamento de 2 bits como descrito no documento do trabalho. O mesmo é ligado a porta 1 de um MUX sobre a saída do MUX do Branch como descrito na Figura 7.

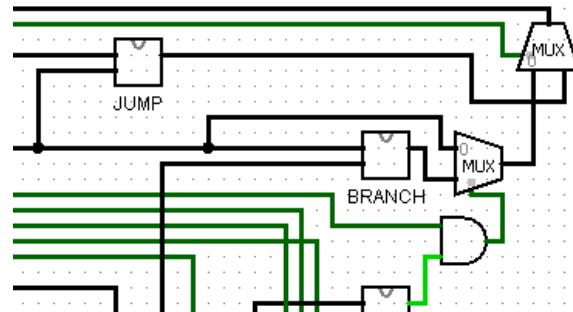


Figura 7: Estrutura do JUMP

As entradas desse JumpRegister são dadas pelo pc+4 e a saída Address do decoder, sua saída é ligada ao mux que determina se segue a saída do mux do Branch ou Jump. Como mostrado na Figura 6, foi adicionado uma estrutura dentro do módulo de controle para controlar a saída JUMP, dada pelo OPCODE 000010.

10. Testes

5.1 Teste_01.asm

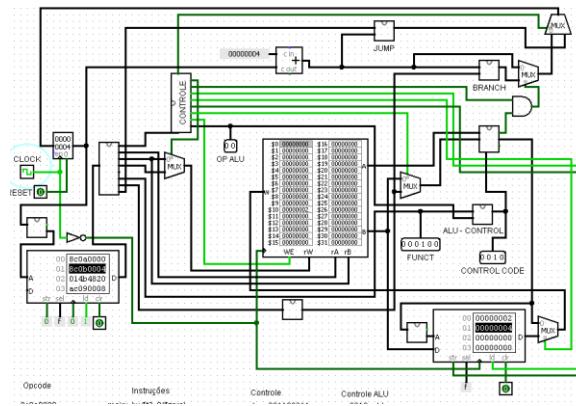


Figura 8: lw \$t2, 0(\$zero)

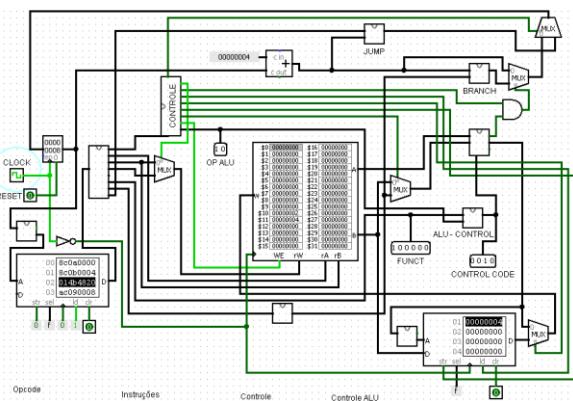


Figura 9: lw \$t3, 4(\$zero)

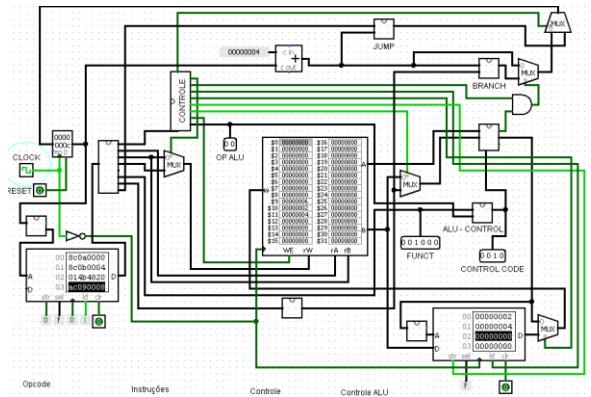


Figura 10: `add $t1,$t2,$t3`

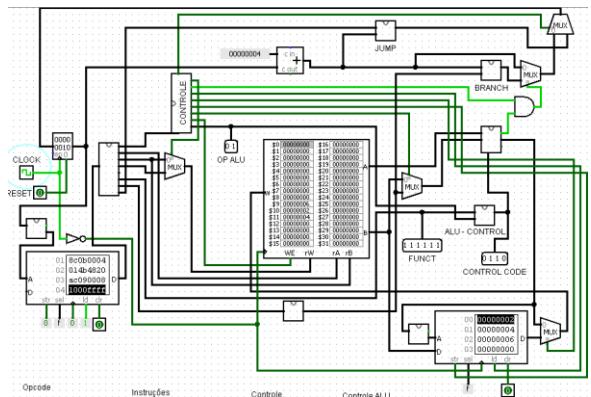


Figura 11: `sw $t1, 8($zero)`

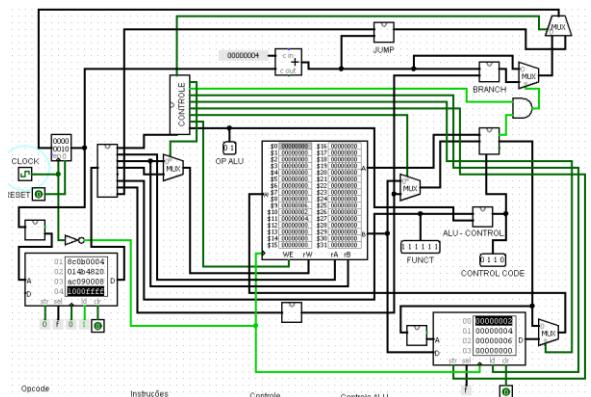


Figura 12: `beq $zero, $zero, exit`

5.2. Teste_02.asm

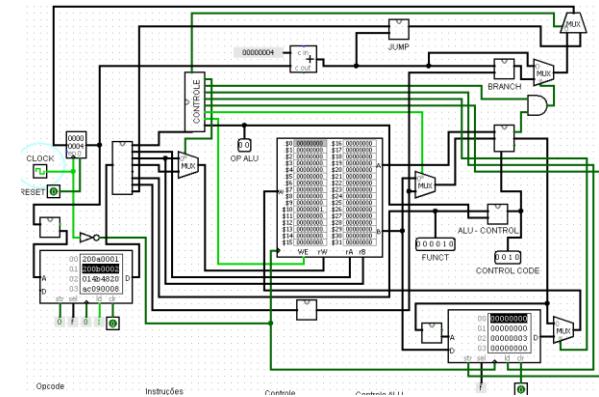


Figura 13: `addi $t2, $zero, 1`

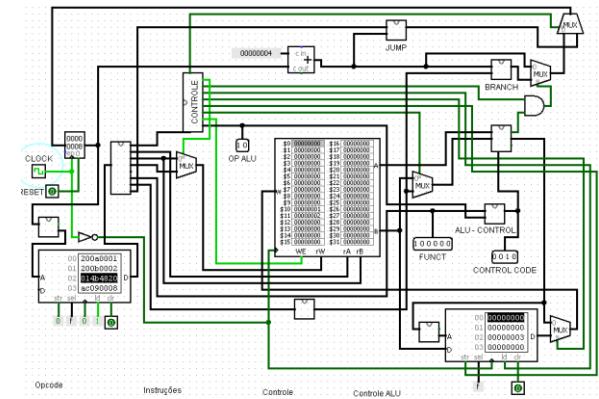


Figura 14: `addi $t3, $zero, 2`

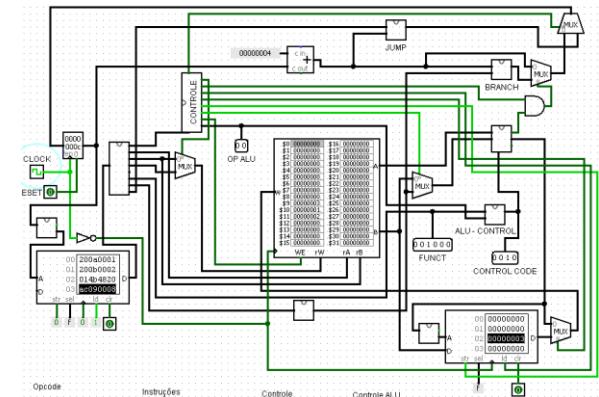


Figura 15: `add $t1,$t2,$t3`

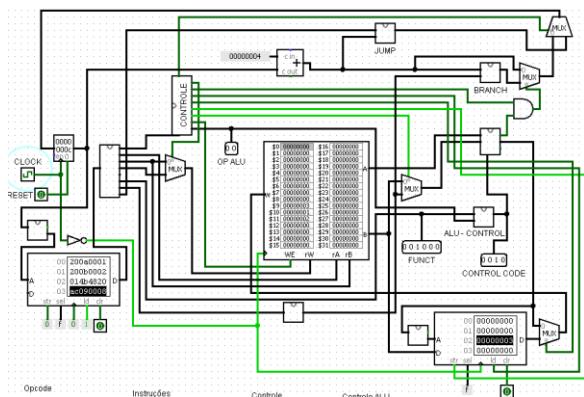


Figura 16: sw \$t1, 8(\$zero)

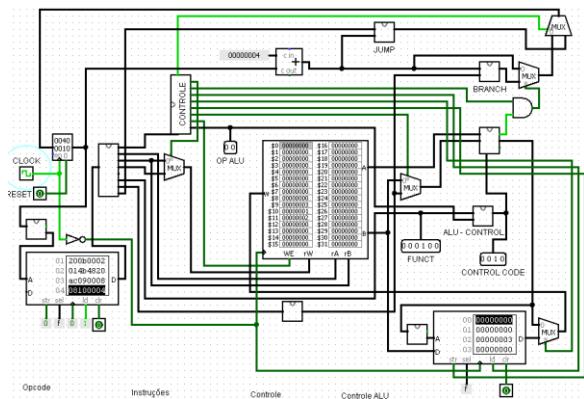


Figura 17: j exit

11. Conclusão

Concluindo, a partir dos resultados obtidos nos testes pode-se comprovar as funcionalidades do microprocessador. Em todas as etapas foi possível realizar as instruções com êxito no software logismim.

Referências

H. John, P. David, Arquitetura de Computadores. Elsevier Editora Ltda, 2013.