

Caminho de Dados de Ciclo Único e Controle MIPS

Ricardo Fedrigo, RA: 1711725
Caio Eduardo Theodoro, RA.: 2044560

26 de abril de 2021

1 Introdução

Neste relatório, será documentado e detalhado os processos da implementação do subconjunto de instruções da arquitetura MIPS estudados na disciplina de Arquitetura e Desenvolvimento de computadores, e também tem como propósito, entender o Logisim e suas bibliotecas internas e externas.

2 Desenvolvimento:

2.1 Funções básicas :

2.1.1 Sinais de controle :

Sinais de controle são um bloco com 9 entradas, que comandam os valores dos operadores do controlador.

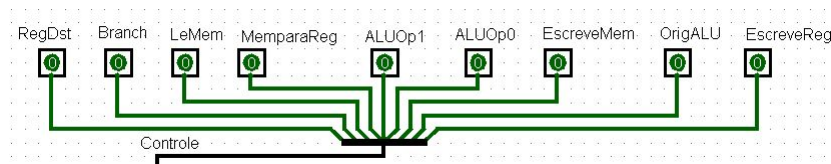


Figura 1 – Sinais de controle

- RegDst - Determina qual o registrador de destino (rt se 0, rd se 1).
- EscreveReg - Dá o sinal para escrever no registrador (0 para não, 1 para sim).

- OrigALU - Controla o multiplexador de seleção de entrada de dados. Ele faz a seleção de entrada de valor imediato ou de registrador.
- Branch(ou OrigPC) - Controla o multiplexador que atualiza o PC, caso 0, ele realiza $PC+=4$, caso 1, o PC segue para o desvio.
- LeMem - Controla a leitura da memória, (1 para ler, 0 para não ler. note que somente a instrução lw tem permissão para ler).
- EscreveMem - Controla a escrita da memória, (1 para escrever, 0 para não escrever. note que somente a instrução sw tem permissão para escrever).
- MemparaReg - Responsavel pela seleção do multiplexador na saída da memória de dados. Ele Determina se a memória de dados será salva em registrador.

2.1.2 Decodificador :

O Decodificador é o bloco que recebe a memória da instrução, e a subdivide dependendo do tipo.

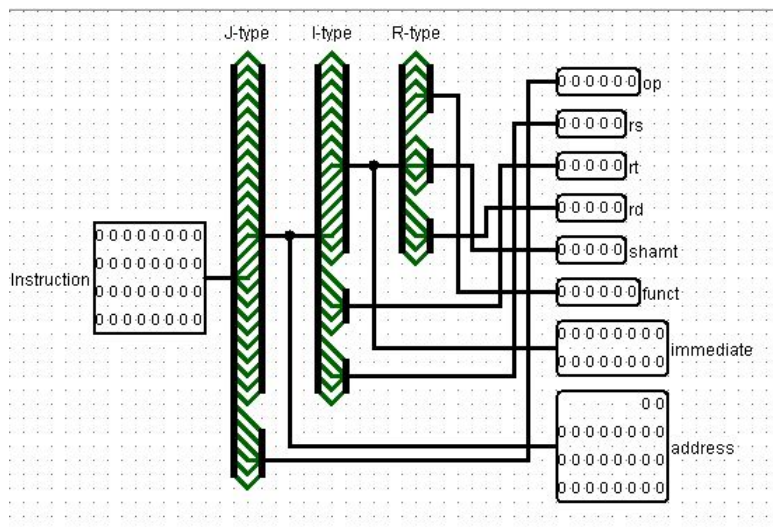


Figura 2 – Circuito do Decodificador

São três os tipos de formatos de instrução do decodificador: I, R e J.

I-Format – referenciam operandos da memória.

R-Format – realizam operações em dados nos registradores.

J-Format – incluem as instruções de desvio.

O tipo I é usado pelo lw, sw e beq na implementação, tipo R é usado para as operações aritmeticas add, sub, slt e or, já o tipo J, pelo jump. A Figura 3 mostra a divisão de bits de cada operação.

R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	16 bit address		
J	op	26 bit address				

Figura 3 – Formatos de instrução

Os canais do controlador partem para outras estruturas de controle do MIPS, como os registradores, o controle principal e o controle de ULA.

2.2 Controle da ULA:

Para montar o controle da ULA, precisamos de duas entradas: ALUOp e Funct.

ALUOp – recebemos dos sinais de controle ALUOp0 e ALUOp1, sendo 1 0 os ALUops das operações aritmeticas.

Funct – recebemos do decodificador, ela é responsavel por sabermos qual operação(add, sub, or, slt e and). A Figura 4 mostra a relação da funct com as operações

Operação	Format	Opcode Field	Function Field
Add	R	0 (00000)	8 (1000)
Sub	R	0 (00000)	10 (1010)
And	R	0 (00000)	12 (1100)
Or	R	0 (00000)	13 (1101)
Slc	R	0 (00000)	14 (1110)

Figura 4 – Operações

Controle da ULA Circuito: Montamos com isso por fim o ALUControl, tendo como saída a operação para a ULA. A Figura 5 mostra o circuito.

2.3 Circuito

Tendo feito a ALU, posicionar os multiplexadores e direcionado os caminhos do decodificador e dos sinais de controle, temos, por fim, o circuito manual do MIPS. A Figura 6 Mostra o circuito.

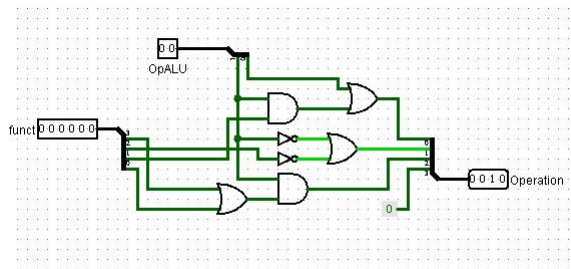


Figura 5 – Circuito ALUControl

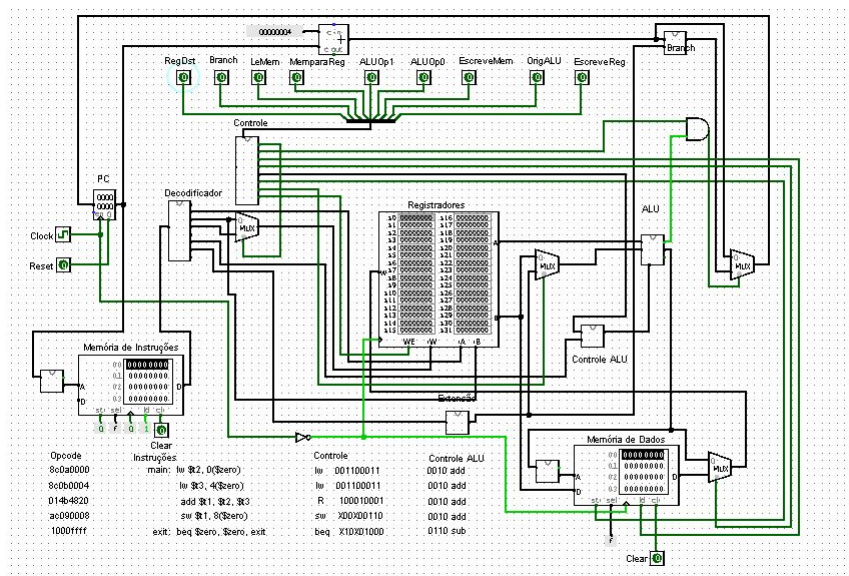


Figura 6 – Circuito Completo

2.4 Operações de teste

Vamos por fim, realizar a rotina do teste1.asm.

```

1      .data
2      .text
3      .globl main
4      main: lw $t2, 0($zero)
5             lw $t3, 4($zero)
6             add $t1, $t2, $t3
7             sw $t1, 8($zero)
8      exit: beq $zero, $zero, exit

```

Figura 7 – Arquivo de teste

2.4.1 lw t2,0(zero)

Primeiro temos lw t2,0(zero), que tem a tabela de controle de sinais ilustrado na Tabela 1:

RegDst	EscreveReg	OrigALU	Branch	EscreveMem	LeMem	MemparaReg	OPcode
0	1	1	0	0	1	1	00

Tabela 1 – Tabela controle para lw t2,0(zero)

Essa operação gera os estados no circuito ilustrados pelo Figura 8 :

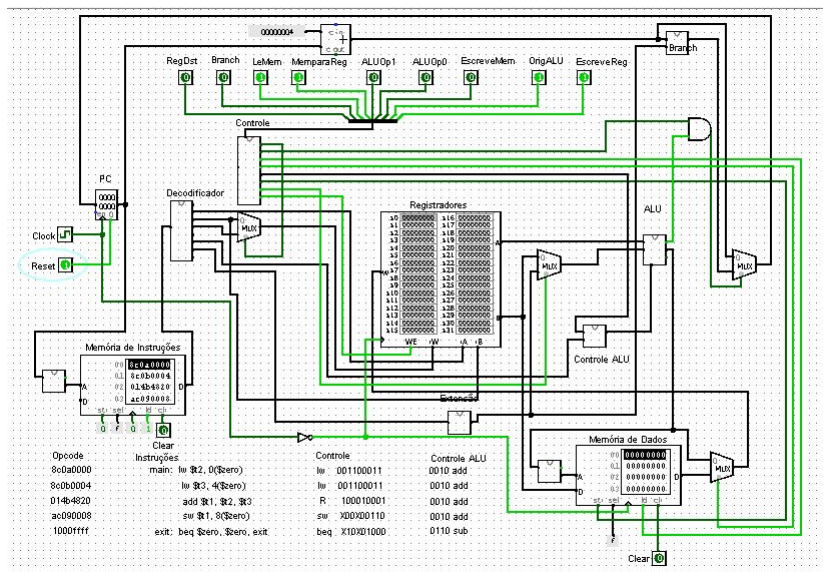


Figura 8 – Controle MIPS para lw t2,0(zero)

2.4.2 lw t3,4(zero)

Depois, temos lw t3,4(zero), que tem a tabela de controle de sinais ilustrado na Tabela 2:

RegDst	EscreveReg	OrigALU	Branch	EscreveMem	LeMem	MemparaReg	OPcode
0	1	1	0	0	1	1	00

Tabela 2 – Tabela controle para lw t3,4(zero)

Essa operação gera os estados no circuito ilustrados pelo Figura 9 :

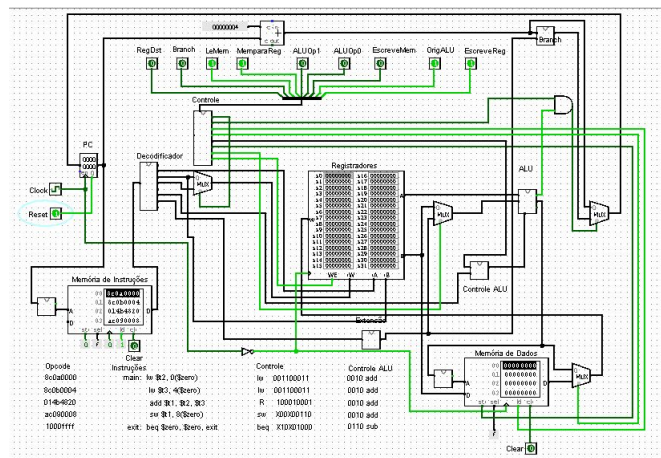


Figura 9 – Controle MIPS para lw $t3, 4(\text{zero})$

2.4.3 add t1,t2,t3

Temos add t1, t2, t3, que tem a tabela de controle de sinais ilustrado na Tabela 3:

RegDst	EscreveReg	OrigALU	Branch	EscreveMem	LeMem	MemparaReg	OPcode
1	1	0	0	0	0	0	10

Tabela 3 – Tabela controle para add $t1,t2,t3$

Essa operação gera os estados no circuito ilustrados pelo Figura 10 :

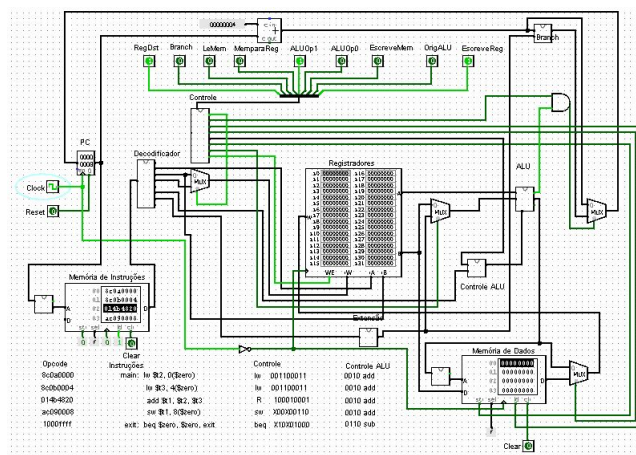


Figura 10 – Controle MIPS para add $t1,t2,t3$

2.4.4 sw t1,8(zero)

Temos sw t1,8(zero), que tem a tabela de controle de sinais ilustrado na Tabela 4:

RegDst	EscreveReg	OrigALU	Branch	EscreveMem	LeMem	MemparaReg	OPcode
1	1	0	0	0	0	0	00

Tabela 4 – Tabela controle para sw t1,8(zero)

Essa operação gera os estados no circuito ilustrados pelo Figura 11 :

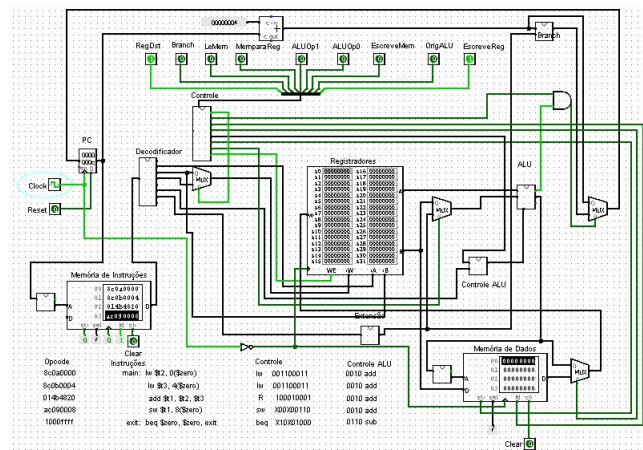


Figura 11 – Controle MIPS para sw t1,8(zero)

2.4.5 beq zero,zero,exit

Temos beq zero,zero,exit, que tem a tabela de controle de sinais ilustrado na Tabela 5:

RegDst	EscreveReg	OrigALU	Branch	EscreveMem	LeMem	MemparaReg	OPcode
0	0	0	1	0	0	0	01

Tabela 5 – Tabela controle para beq zero,zero,exit

Essa operação gera os estados no circuito ilustrados pelo Figura 12 :

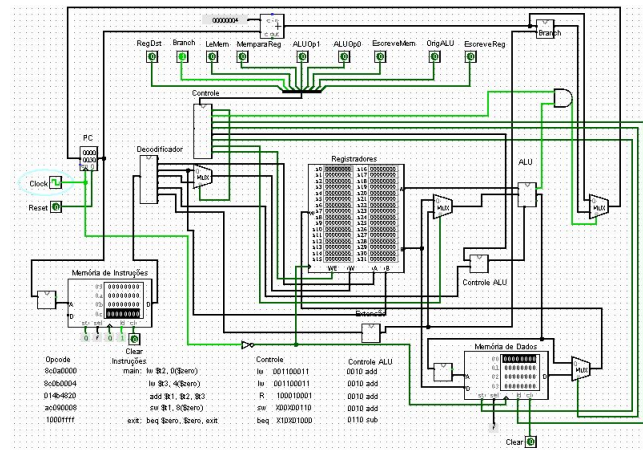


Figura 12 – Controle MIPS para beq *zero,zero,exit*

Após realizado saída, o teste é finalizado.

2.5 Conclusão

Na disciplina de Arquitetura e Redes de Computadores, torna-se fundamental o entendimento desses conceitos básicos de aplicação. A partir deles podemos, por fim, entender os componentes básicos da implementação de controle MIPS e Caminho de Dados de Ciclo Único.