PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR R4-D4

ALUNOS:
Giovanna Mendes Garbácio - 2021000712
Thiago Vieira Camara - 2021000697

Dezembro de 2022
Boa Vista/Roraima

PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR R4-D4

Dezembro de 2022
Boa Vista/Roraima

Resumo

Este trabalho aborda o projeto e implementação do processador R4-D4 de 8 bits, uni-ciclo, baseado na arquitetura MIPS. Neste relatório consta o processo de desenvolvimento desse processador na plataforma Logisim, detalhando seus componentes e testes realizados. Tal projeto foi realizado para apresentação da disciplina de Arquitetura e Organização de Computadores visando obtenção de nota parcial semestral e fixação do conteúdo programático da matéria.

O R4-D4 é capaz de realizar 14 operações com instruções de 8 bits auxiliadas por uma word de 8 bits se necessário, entre elas, operações aritméticas, lógicas, de manejo de memória e pulos condicionais e não-condicionais. Para isso, ela faz uso de 4 registradores, número esse que inspirou o nome do processador de R4-D4 junto a saga Star Wars.

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador R4-D4, bem como a descrição detalhada de cada etapa da construção do processador.

1.1 Plataforma de desenvolvimento

Para a implementação do processador R4-D4 foi utilizado o programa Logisim. Ele é um simulador de circuitos lógicos que permite testar os componentes a medida em que eles são construídos e torná-los subcircuitos para um projeto maior. A versão do Logisim utilizada neste trabalho foi a 2.7.1, a versão mais atual lançada em março de 2011.

Uma das grandes vantagens do uso dessa plataforma é a facilidade de arquitetar um circuito pela sua mecânica de agarrar, soltar itens e traçar conexões. O Logisim também disponibiliza alguns componentes usados com frequência já prontos para implementação, como é o caso de registradores, memórias, contadores, somadores, subtratores, comparadores e portas lógicas, que foram utilizados para compor o R4-D4.

Uma particularidade desse programa é a forma como as memórias integradas armazenam informações, elas o fazem utilizando o sistema numérico hexadecimal enquanto o resto dos componentes trabalham com o sistema binário de números. Ainda assim, é possível definir a largura em bits binários de endereço e dos dados da memória. Essa especificação exige que os códigos programados para os testes fossem traduzidos para hexadecimal para que eles pudessem ser carregados na memória de instruções.

A figura abaixo é uma tabela de estatísticas formada pela própria plataforma com base no processador. Os componentes referentes a biblioteca CPU são os subcircuitos construídos que foram encapsulados para favorecer a visualização do projeto e o restante são ferramentas fornecidas pelo próprio Logisim.

Componente	Biblioteca	Simples	Exclusiva	Recursiva
Banco de registradores	CPU	1	1	1
Unidade de controle	CPU	1	1	1
Memória de dados	CPU	1	1	1
ALU	CPU	1	1	1
Distribuidor	Conexão	5	20	20
Pino	Conexão	15	49	49
Túnel	Conexão	30	30	30
Clock	Conexão	1	1	1
Constante	Conexão	0	155	155
Porta NOT	Portas	3	7	7
Porta AND	Portas	2	3	3
Porta OR	Portas	0	2	2
Porta XOR	Portas	0	1	1
Buffer controlado	Portas	0	1	1
Multiplexador	Plexers	3	7	7
Demultiplexador	Plexers	0	1	1
Somador	Aritmética	0	1	1
Subtrator	Aritmética	0	1	1
Comparador	Aritmética	0	1	1
Registrador	Memória	0	4	4
Contador	Memória	1	1	1
RAM	Memória	1	2	2
Botão	Entrada/Saída	4	4	4
LED	Entrada/Saída	1	1	1
Rótulo	Base	30	60	60
TOTAL (sem subcircuitos do projeto)		96	352	352
TOTAL (com subcircuitos)		100	356	356

Figura 1 - Estatísticas do R4-D4 geradas pelo Logisim

1.2 Conjunto de instruções

O processador R4-D4 possui quatro registradores: R0, R1, R2 e R3, assim como três formatos de instruções de 8 bits: instruções do tipo R (lógicas, aritméticas e desvio condicional), do tipo I (operações com memória de dados e aritméticas imediatas) e do tipo J (desvio incondicional). Cada word de instrução de 8 bits é acompanhada por outros 8 bits auxiliares, eles são utilizado em operações imediatas como o número a ser adicionado ou subtraído diretamente do valor de um registrador e como o endereço de memória a ser manipulado pelos comandos load, store, jump e beq.

Reconhece-se que tal formato de instruções não seria eficiente em um processador real, contudo, visto a natureza desse projeto que visa exercitar e consolidar conceitos de funcionamento e componentes de uma CPU, o modelo aplicado é válido e não interfere na eficácia do R4-D4.

Sendo assim, as palavras principal e auxiliar devem ser traduzidas do sistema binário para o hexadecimal para que as instruções possam ser lidas pela memória de instrução do processador.

Seguem algumas considerações sobre a estrutura das instruções:

- **Opcode**: a operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
- **RA**: o registrador auxiliar, contém um operando fonte;
- **RP**: o registrador principal, contém um operando fonte, sendo também o registrador de destino para instruções que o necessitam;
- **WP:** Word Principal (palavra principal) é o conjunto de 8 bits que recebe o opcode e os códigos dos registradores;

- **WA:** Word Auxiliar (palavra auxiliar) são os 8 bits destinados a referenciar ou endereços de células de memória de dados e de instruções, ou números utilizados em operações imediatas, chamados de endereço e n.imediatos, respectivamente;
- **X:** Quando algum dos bits não é necessário para a instrução ele é representado como X.

Formato geral das instruções:

Tabela 1 - Estrutura das instruções do R4-D4

ESTRUTURA GERAL DAS INSTRUÇÕES			
Divisão dos bits	**OPCODE**	**RA**	**RP**
Word Principal	**0-3**	**4-5**	**6-7**
Word Auxiliar	**0-7**		

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total $(Bit(0e1)^{NumeroTodaldeBitsdoOpcode} \therefore 2^X = X)$ de 16 opcodes **(0000 - 1111)**, no entanto, foram utilizados apenas 14 deles, que foram distribuídos entre as instruções, assim como é apresentado na Tabela 5.

Formato do tipo R: Este formato aborda instruções aritméticas e lógicas, e instruções de desvio condicional BEQ.

Tabela 2 - Estrutura da instrução do tipo R

ESTRUTURA DAS INSTRUÇÕES DO TIPO R			
Divisão dos bits	**OPCODE**	**RA**	**RP**
Word Principal	**0-3**	**4-5**	**6-7**
Word Auxiliar	**XXXXXXXX**		

Formato do tipo I: Este formato aborda instruções aritméticas imediatas e instruções Load e Store.

Tabela 3 - Estrutura da instrução do tipo I

ESTRUTURA DAS INSTRUÇÕES DO TIPO I			
Divisão dos bits	**OPCODE**	**RA**	**RP**
Word Principal	**0-3**	**XX**	**6-7**
Word Auxiliar	**0-7**		

Formato do tipo J: Este formato abrange a instrução Jump.

Tabela 4 - Estrutura da instrução do tipo J

ESTRUTURA DAS INSTRUÇÕES DO TIPO J			
Divisão dos bits	**OPCODE**	**RA**	**RP**
Word Principal	**0-3**	**XX**	**XX**
Word Auxiliar	**0-7**		

Instruções suportadas pelo R4-D4:

Tabela 5 - Visão geral das instruções do Processador R4-D4

OP CODE	INSTRUÇÃO	FORMATO	FORMATO DAS INSTRUÇÕES EM BINÁRIO	DESCRIÇÃO	REPRESENTAÇÃO EM FORMATO MIPS
0000	ADD	R	0000 RA RP XXXXXXXX	RP + RA	add $S0, $S1, $S1
0001	ADDi	I	0001 XX RP N.IMEDIATO	RP + número imediato	addi $S1, $S1, 1
0010	SUB	R	0010 RA RP XXXXXXXX	RP - RA	sub $S0, $S1, $S1
0011	SUBi	I	0011 XX RP N.IMEDIATO	RP - número imediato	subi $S0, $S0, 2
0100	BEQ	R	0100 RA RP ENDEREÇO	if(RA==RP), endereço	beq $S0, $S1, label
0101	JUMP	J	0101 XX XX ENDEREÇO	jump, endereço	j label
0110	SW	I	0110 XX RP ENDEREÇO	RP armazenado no endereço	sw $S0, 3($S2)
0111	LW	I	0111 XX RP ENDEREÇO	armazenar valor do endereço em RP	lw $S3, 3($S2)
1000	AND	R	1000 RA RP XXXXXXXX	RA and RP	and $S0, $S1, $S1
1001	NAND	R	1001 RA RP XXXXXXXX	RA nand RP	nand $S0, $S1, $S1
1010	OR	R	1010 RA RP XXXXXXXX	RA or RP	or $S0, $S1, $S1
1011	NOR	R	1011 RA RP XXXXXXXX	RA nor RP	nor $S0, $S1, $S1
1100	XOR	R	1100 RA RP XXXXXXXX	RA xor RP	xor $S0, $S1, $S1
1101	NOT	R	1101 XX RP XXXXXXXX	not RP	not $S0

1.3 Descrição do Hardware

Nesta seção são descritos os componentes do hardware que compõem o processador R4-D4, incluindo uma descrição de suas funcionalidades, valores de entrada e saída.

1.3.1 ALU ou ULA

O componente ALU (Unidade Lógica e Aritmética) tem como objetivo efetuar as operações aritméticas de soma e subtração e as operações lógicas, AND, NAND, OR, NOR, XOR e NOT. Para isso, ela recebe três entradas:

- **A** - dado de 8 bits que atua como primeiro operador;
- **B** - dado de 8 bits que atua como segundo operador;
- **OP ULA** - dado de 3 bits que, por meio de um multiplexador, seleciona qual operação será realizada de acordo com a tabela:

Tabela 6 - OpULA

OP ULA	OPERAÇÃO
000	Adição
001	Subtração
010	AND
011	NAND
100	OR
101	NOR
110	XOR
111	NOT

Em relação às saídas da ULA, ela também possui três:

- **S (saída)** - é o resultado da operação especificada pelo OpULA;
- **Z (zero)** - identifica de o resultado de uma subtração é 0;
- **OV (Overflow)** - identificador de overflow caso a operação exceda 8 bits.

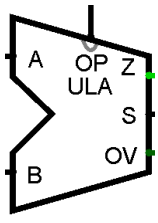

Figura 2 - Subcircuito ULA encapsulado

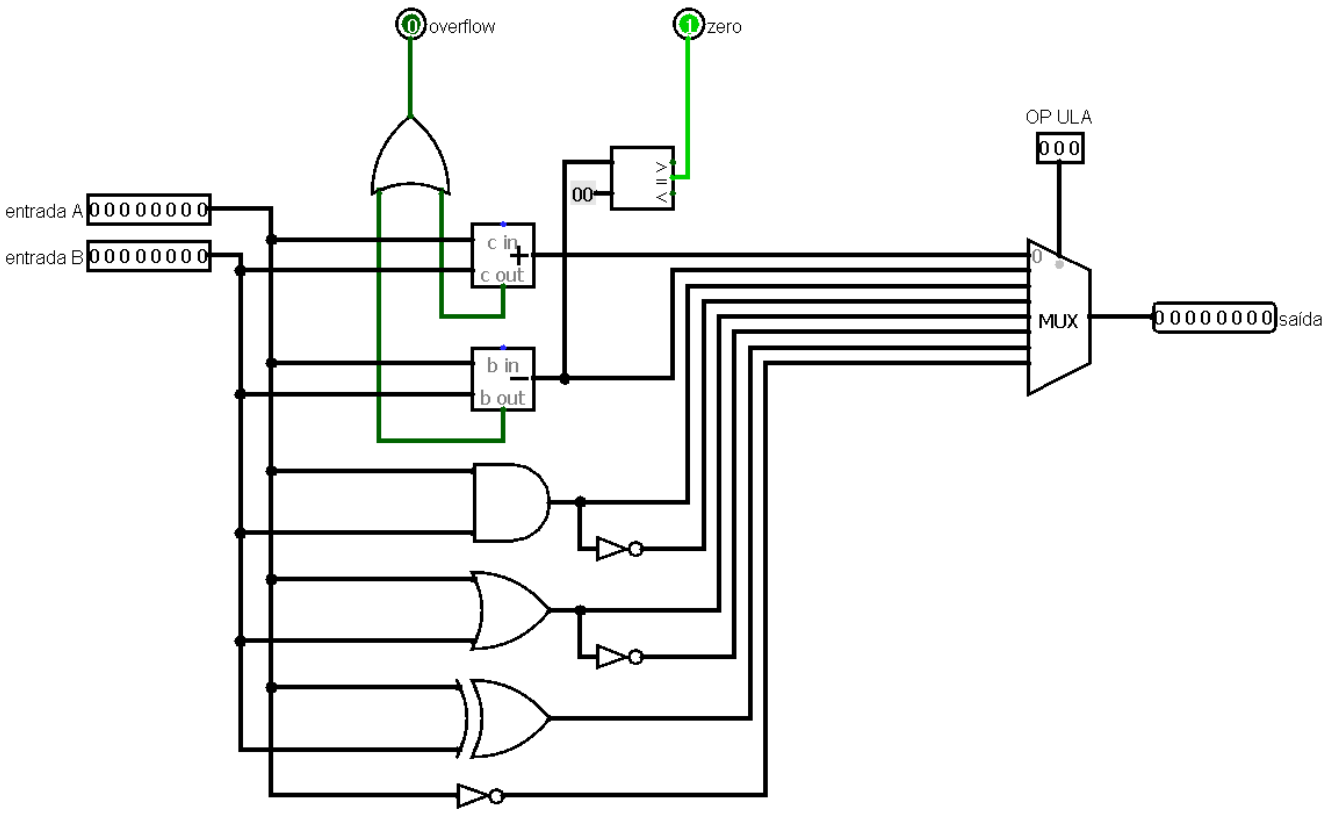

Figura 3 - Circuito ULA

1.3.1.1 ZERO

O output ZERO da ULA é utilizado na instrução BEQ, nela ocorre a comparação dos inputs das entradas A e B por meio de uma subtração, onde se o A e B forem iguais, o resultado será zero. A saída do subtrator é comparada com 0 em um comparador que acusa a igualdade energizando a trilha do ZERO, como ilustrado na figura 3. Posteriormente, tal trilha serve como uma das duas entradas para uma porta AND, a outra sendo a indicação da unidade de controle de uma operação de desvio condicional BEQ, que aciona o desvio para o endereço fornecido na instrução.

1.3.2 Banco de Registradores

O banco de registradores é um conjunto de registradores que armazenam temporariamente dados, de modo que eles possam ser facilmente acessados e deslocados. O processador R4-D4 foi nomeado como um trocadilho entre o personagem da saga Star Wars R2-D2 e o número de quatro registradores presentes na presente CPU.

Um registrador funciona de modo a armazenar uma informação que está sendo recebida quando houver uma subida do clock e o enable estiver como 1. Ao agrupar quatro registradores, eles compartilham do mesmo input, chamado de dado a ser escrito, do mesmo clock e do mesmo clear (que reseta o conteúdo

armazenado nos registradores para 0), esses três inputs foram colocados no canto superior direito do circuito integrado.

O enable é controlado por meio de um multiplexador, ele é ativado por uma flag da unidade de controle chamada RegWrite e o registrador que deve ser escrito é escolhido pelo input seletor chamado REG1 (reg a ser lido 1). O mesmo REG1 também seleciona a primeira saída (dado lido 1). Paralelamente, REG2 seleciona a segunda saída (dado lido 2), este não pode ser escrito. Esse sistema é o que define que o registrador de destino das operações com writeback seja REG1, portanto, pode-se dizer que RP é REG1 e RA é REG2.

O dado a ser escrito é um input provindo de um multiplexador exterior ao banco de dados, a informação pode ser provinda de um número imediato, da memória de dados ou do resultado de uma operação da ULA, selecionado pela flag da Unidade de Controle MemtoReg.

Os outputs nomeados REG 00, 01, 10 e 11 são apenas saídas de controle para melhor visualização do funcionamento do banco de registradores, eles representam R0, R1, R2 e R3 respectivamente. No circuito encapsulado, eles estão posicionados no canto superior esquerdo.

Figura 4 - Circuito integrado de Banco de registradores

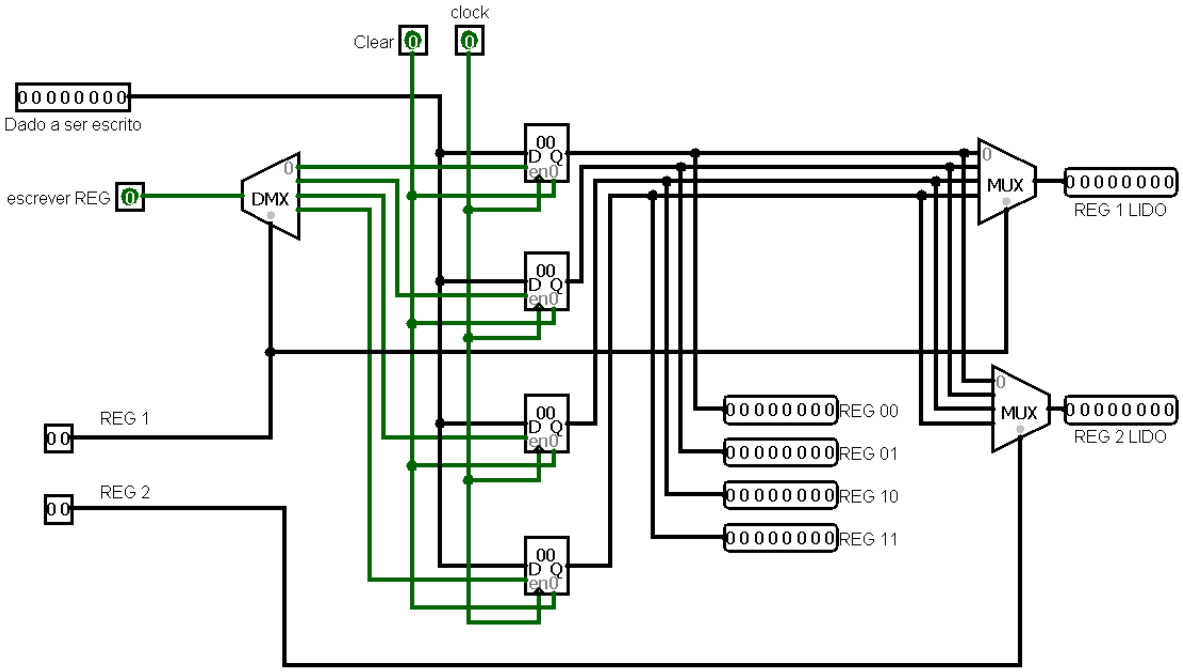

Figura 5 - Circuito do Banco de registradores

1.3.3 Clock

Todos os componentes que exigem entrada de clock foram conectados com um único clock por meio do uso de túneis, uma ferramenta disponibilizada pelo Logisim para diminuir a quantidade de fios visíveis e evitar curtos.

Figura 6 - Clock do R4-D4

1.3.4 Unidade de Controle

O componente Unidade de Controle tem como objetivo realizar o controle de todos os componentes do processador de acordo com o opcode de cada instrução, ou seja, os primeiros 4 bits da word principal. Esse controle é feito através das flags de saída Endereço, Jump, Branch, MemtoReg, ALUOP, MemWrite/MemRead, ALUSrc e RegWrite. Segue a descrição das flags e uma tabela onde é feita a associação entre os opcodes e as flags de controle:

- **Endereço:** ativado nas operações Load e Store, onde o dado recebido pela memória de dados é o endereço contido na word auxiliar, que serve de seletor da célula de memória a ser lida ou escrita;
- **Jump:** ativado apenas na operação Jump, ativa a porta AND que troca os sinais recebidos do Program Counter, fazendo com que o PC recebe o endereço contido na word auxiliar e vá para ele na próxima subida de clock;
- **Branch:** é uma flag ativada com a instrução Beq, que entra em uma porta AND com a saída zero da ULA, se ambas estiverem com nível alto de energia, o pulo condicional é ativado;
- **MemtoReg:** essa flag possui 2 bits que servem de seletor para um multiplexador cuja saída é o dado a ser escrito no banco de dados, 00 indica o recebimento da word auxiliar, 01 indica o resultado da ULA, 10 indica um dado lido da memória e 11 não é usado;
- **ULAOp:** define a operação a ser realizada na ULA como mostrado na Tabela 6. Algumas instruções utilizam o mesmo código da ULAOp, enquanto as que não usam ficam em 000;
- **MemWrite/MemRead:** ativada somente com a operação Store, onde ocorre escrita de informações na memória de dados, no restante das instruções essa flag permanece no estado 0, o estado de leitura;
- **ALUSrc:** essa flag define a saída do multiplexador que entra como segundo input na ULA, se 0, é o segundo dado lido do banco de registradores, se 1, é o conteúdo da word auxiliar;

- **RegWrite:** ativa a escrita do registrador principal com o conteúdo recebido de acordo com a flag ALUSrc.

Tabela 7 - Detalhes das flags de controle do processador

	Endereço	Jump	Branch	MemtoReg	ALUOp	MemWrite/ MemRead	ALUSRc	REG Write
ADD	0	0	0	01	000	0	0	1
ADDi	0	0	0	01	000	0	1	1
SUB	0	0	0	01	001	0	0	1
SUBi	0	0	0	01	001	0	1	1
BEQ	0	0	1	00	001	0	0	0
J	0	1	0	00	000	0	0	0
SW	1	0	0	00	000	1	0	0
LW	1	0	0	10	000	0	0	1
AND	0	0	0	01	010	0	0	1
NAND	0	0	0	01	011	0	0	1
OR	0	0	0	01	100	0	0	1
NOR	0	0	0	01	101	0	0	1
XOR	0	0	0	01	110	0	0	1
NOT	0	0	0	01	111	0	0	1

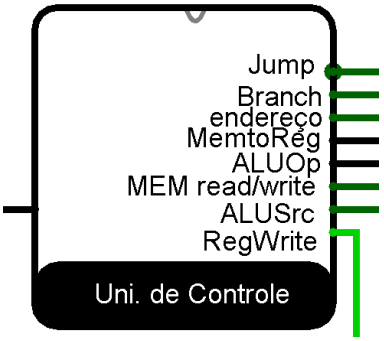

Figura 7 - Unidade de controle encapsulado

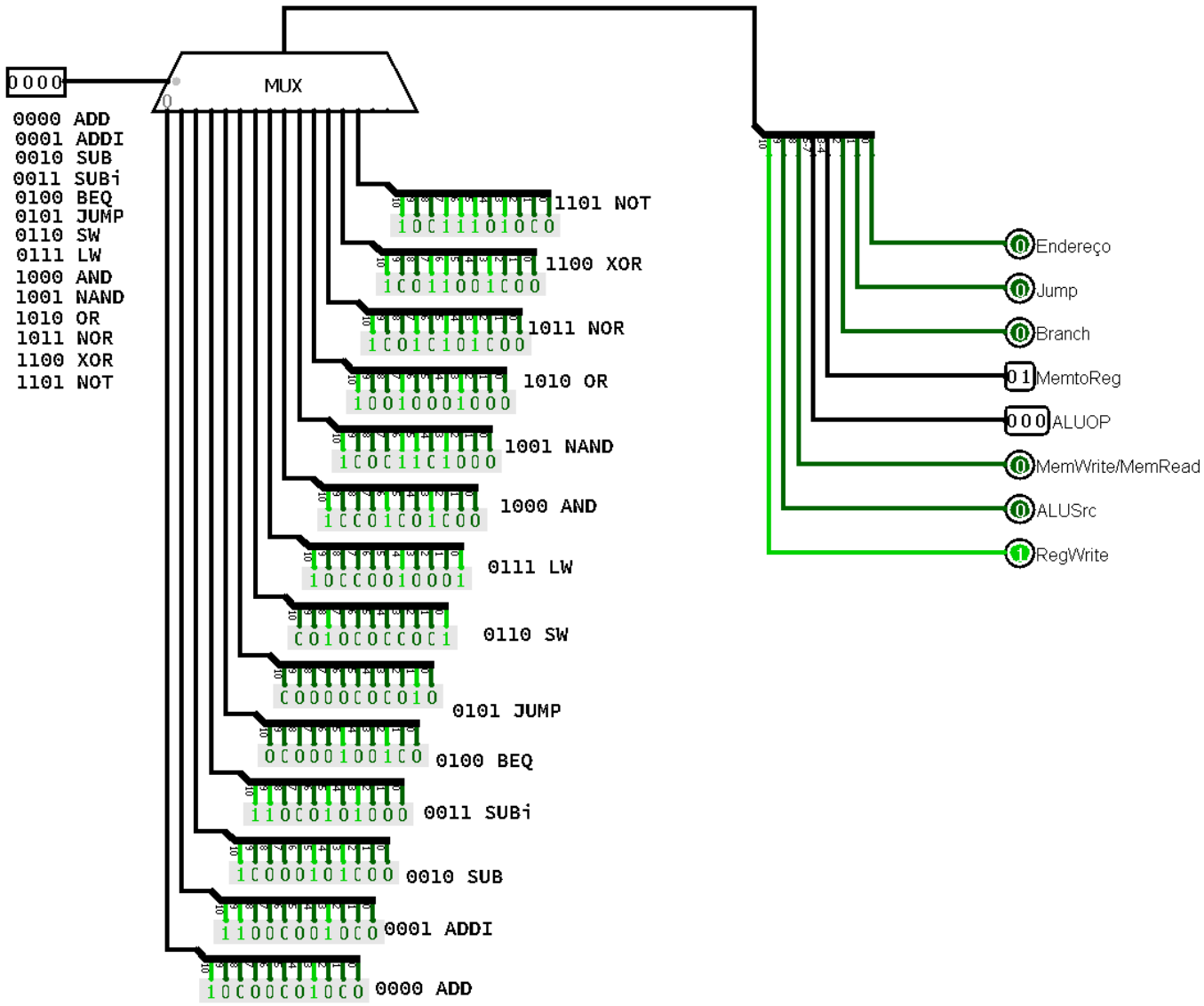

Figura 8 - Circuito Unidade de Controle

1.3.5 Memória de dados

A memória de dados serve como local de armazenagem para dados controlado pelas instruções Load e Store. Para sua construção foi utilizada a memória RAM provida pela plataforma Logisim, que foi definida para 8 bits de largura tanto para o endereço quanto para os dados.

Ela possui seis entradas, o seletor, que indica o endereço da célula de memória a ser lida ou escrita, o enable do seletor que está ligado a flag MemWrite/MemRead da unidade de controle, Load, Clear e o Clock, como saída existe apenas o dado lido, que serve para a função WriteBack, que retorna o valor da memória para o banco de registradores.

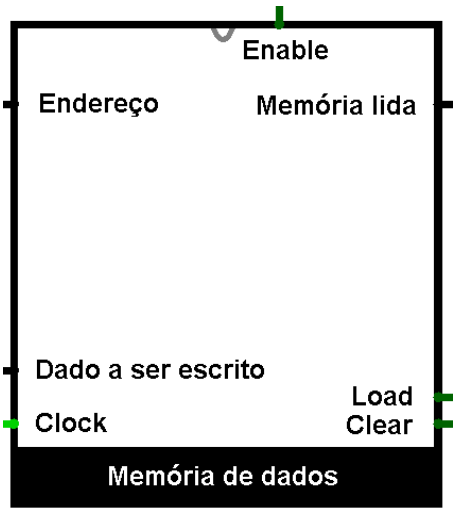

Figura 9 - Memória de dados encapsulada

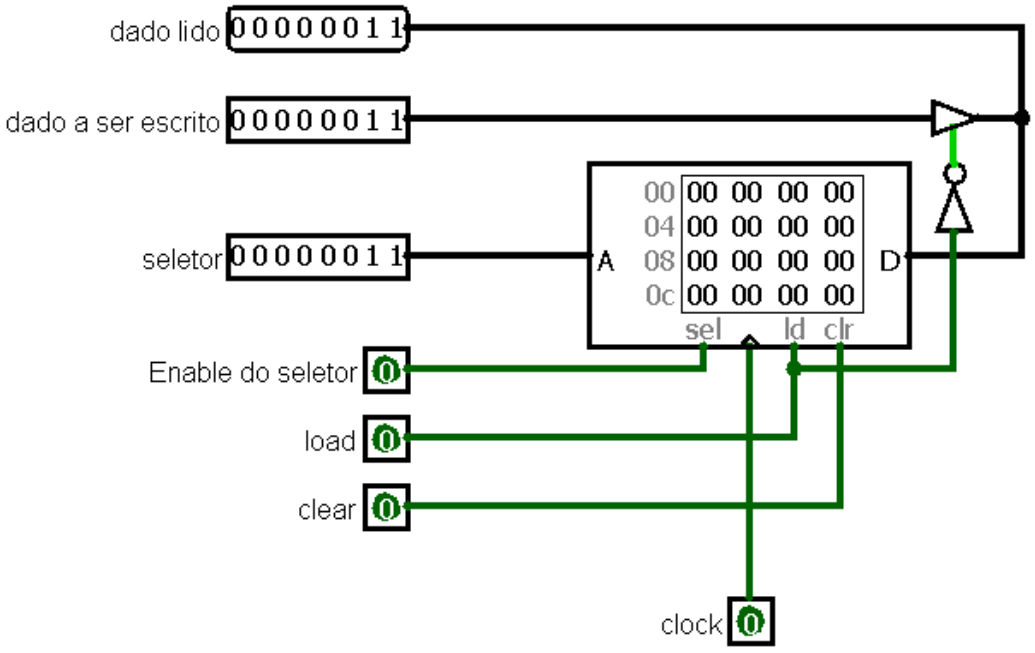

Figura 10 - Circuito da memória de dados

1.3.6 Memória de Instruções

Da mesma forma que a memória de dados apresentada no item 1.3.5, a memória de instruções foi feita com uma memória RAM, recebendo o endereço da instrução a ser lida do Program Counter a cada subida de clock. Diferentemente da memória de dados, a largura em bits dos dados é de 16 para acomodar ambas words principal e auxiliar na célula, a largura do endereço é de 8 bits. Uma

vantagem do uso desse componente pronto é que ele aceita o carregamento da memória a partir de uma imagem, facilitando a passagem de código para testes da CPU. O botão Clear retorna a memória para seu estado inicial com todos os dados iguais a 0.

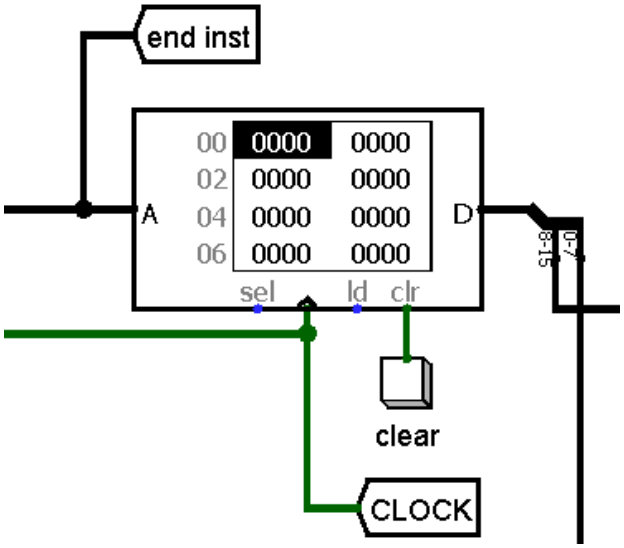

Figura 11 - Circuito da memória de instruções

1.3.7 PC

O Program Counter, chamado de PC, é formado por um contador que funciona de modo a somar 1 a seu valor a cada subida de clock. Ele serve para passar o endereço da instrução a ser executada pelo processador para a memória de instruções, seja ele consecutivo ou um endereço específico nos casos das operações Jump e Beq, essas ativam o recebimento do input D provindo da word auxiliar. O botão clear retorna o contador ao estágio 00.

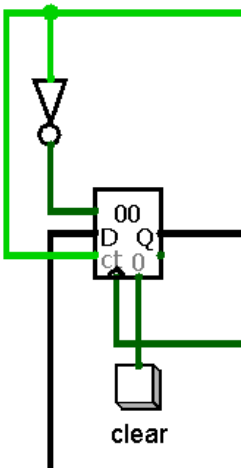

Figura 12 - Circuito do PC

1.4 Datapath

É a conexão entre as unidades funcionais formando um único caminho de dados e adicionando uma unidade de controle responsável pelo gerenciamento das ações que serão realizadas para diferentes classes de instruções. É o circuito do processador completo, ligados os subcircuitos da unidade de controle, banco de registradores, memória de dados e unidade lógica e aritmética com multiplexadores, memória de instruções e program counter. Inclui-se também portas AND e NOT para o procedimento das instruções Jump e Beq, além de distribuidores para divisão dos bits das instruções.

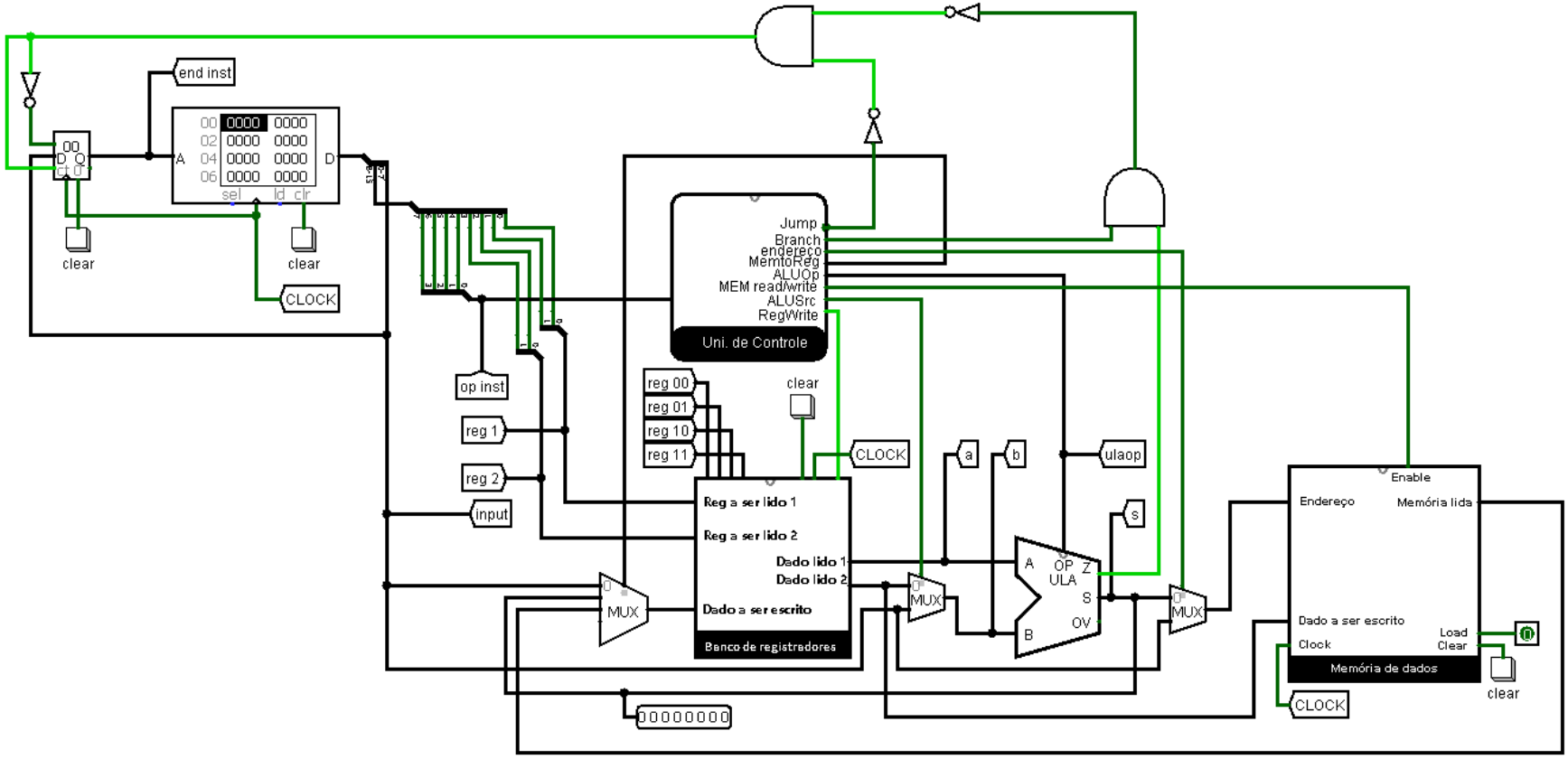

Figura 13 - Datapath do R4-D4

2 Simulações e Testes

Objetivando analisar e verificar a eficácia do processador, realizou-se alguns testes para observar o R4-D4 em funcionamento. Para tal utilizou-se como exemplo os códigos para calcular o número da sequência de Fibonacci e efetuar todas as operações suportadas pelo processador. Após a execução da simulação do código fibonacci e o teste das operações, os seguintes relatórios são exibidos.

Fibonacci: esse código teve como objetivo demonstrar o uso do processador com um objetivo real, no caso, realizar a contagem da sequência de fibonacci até o limite de representação de 8 bits ser atingido. Nesse programa são ativados o PC, a memória de instrução, o banco de registadores e a ULA. O output é verificado na saída do resultado da ULA das operações de ADD, quando ele retorna ao banco de registradores para ser escrito no RP, na operação jump o retorno é sempre 3 (0000 0011) e deve ser desconsiderado. A sequência segue: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233. Ele para em 233 pois o próximo número é 377, que ultrapassa os 8 bits suportados pelo processador.

Tabela 8 - Detalhes do código para contagem da sequência de Fibonacci

Descrição da instrução	Endereço de memória	Instrução em Binário	Instrução em Hexadecimal
Addi R0 + 1	0000 0001	00010000 00000001	1001
Addi R1 + 1	0000 0010	00010001 00000001	1101
Add R0 + R1	0000 0011	00000001 00000000	0100
Add R1 + R0	0000 0100	00000100 00000000	0400
Jump 0000 0011	0000 0101	01010000 00000011	5003

Endereço Instrução	OP cpu	R1	R2	ULA A	ULA B	ULA S	Dado a ser escrito	Dado a ser escrito
0000 0000	0000	00	00	0000 0000	0000 0000	0000 0000	0000 0000	0
0000 0001	0001	00	00	0000 0000	0000 0001	0000 0001	0000 0001	1
0000 0010	0001	01	00	0000 0000	0000 0001	0000 0001	0000 0001	1
0000 0011	0000	01	00	0000 0001	0000 0001	0000 0010	0000 0010	2
0000 0100	0000	00	01	0000 0001	0000 0010	0000 0011	0000 0011	3
0000 0101	0101	00	00	0000 0011	0000 0011	0000 0110	0000 0011	3
0000 0011	0000	01	00	0000 0010	0000 0011	0000 0101	0000 0101	5
0000 0100	0000	00	01	0000 0011	0000 0101	0000 1000	0000 1000	8
0000 0101	0101	00	00	0000 1000	0000 1000	0001 0000	0000 0011	3
0000 0011	0000	01	00	0000 0101	0000 1000	0000 1101	0000 1101	13
0000 0100	0000	00	01	0000 1000	0000 1101	0001 0101	0001 0101	21
0000 0101	0101	00	00	0001 0101	0001 0101	0010 1010	0000 0011	3
0000 0011	0000	01	00	0000 1101	0001 0101	0010 0010	0010 0010	34
0000 0100	0000	00	01	0001 0101	0010 0010	0011 0111	0011 0111	55
0000 0101	0101	00	00	0011 0111	0011 0111	0110 1110	0000 0011	3
0000 0011	0000	01	00	0010 0010	0011 0111	0101 1001	0101 1001	89
0000 0100	0000	00	01	0011 0111	0101 1001	1001 0000	1001 0000	144
0000 0101	0101	00	00	1001 0000	1001 0000	0010 0000	0000 0011	3
0000 0011	0000	01	00	0101 1001	1001 0000	1110 1001	1110 1001	233

Figura 14 - Teste Fibonacci

Teste operações: esse código visa testar o funcionamento de cada uma das 14 operações do processador para certificar se ocorreram os resultados esperados.

Tabela 9 - Detalhes do código de teste de operações

Descrição da instrução	Endereço de memória	Instrução em Binário	Instrução em Hexadecimal
Addi R0 + 5	0000 0001	00010000 00000101	1005
Addi R0 + 1	0000 0010	00010001 00000001	1101
Add R0 + R1	0000 0011	00000001 00000000	0100
Subi R0 - 2	0000 0100	00110000 00000010	3002
Sub R1 - R0	0000 0101	00100001 00000000	2100
Beq R0 R1, 00010000	0000 0110	01000001 00010000	4110
J, 00010010	0001 0000	01010000 00010010	5012
Sw R0,	0001 0010	01100000 00000100	6004
Lw R0,	0001 0011	01110010 00000100	7204
Not R0	0001 0100	11010000 00000000	d100
Xor R0 R1	0001 0101	11000001 00000000	c100
Nor R0 R1	0001 0110	10110001 00000000	b100
Nand R0 R1	0001 0111	10010001 00000000	9100
And R0 R1	0001 1000	10000001 00000000	8100

Endereço Instrução	OP cpu	R1	R2	ULA A	ULA B	ULA S	RAM(430,160)[4]
0000 0001	0001	00	00	0000 0000	0000 0101	0000 0101	0000 0000
0000 0010	0001	01	00	0000 0000	0000 0001	0000 0001	0000 0000
0000 0011	0000	01	00	0000 0001	0000 0101	0000 0110	0000 0000
0000 0100	0011	00	00	0000 0101	0000 0010	0000 0011	0000 0000
0000 0101	0010	01	00	0000 0110	0000 0011	0000 0011	0000 0000
0000 0110	0100	01	00	0000 0011	0000 0011	0000 0000	0000 0000
0001 0000	0101	00	00	0000 0011	0000 0011	0000 0110	0000 0000
0001 0010	0110	00	00	0000 0011	0000 0011	0000 0110	0000 0000
0001 0011	0111	10	00	0000 0000	0000 0011	0000 0011	0000 0011
0001 0100	1101	00	00	0000 0011	0000 0011	1111 1100	0000 0011
0001 0101	1100	01	00	0000 0011	1111 1100	1111 1111	0000 0011
0001 0110	1011	01	00	1111 1111	1111 1100	0000 0000	0000 0011
0001 0111	1001	01	00	0000 0000	1111 1100	1111 1111	0000 0011
0001 1000	1000	01	00	1111 1111	1111 1100	1111 1100	0000 0011

Figura 15 - Teste Operações

3 Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 8 bits R4-D4. Sua construção foi facilitada pela realização de um projeto anterior apresentado à mesma disciplina de Arquitetura e Organização de Computadores, que se constituiu da montagem de alguns circuitos no Logisim, que permitiu a familiarização com a plataforma utilizada e ainda possibilitou a reutilização de alguns dos subcircuitos já desenvolvidos neste trabalho.

No entanto, foram encontradas dificuldades no decorrer do projeto. Primeiramente, um obstáculo foi as especificações do limite da quantidade de bits das instruções, em teoria o processador R4-D4 se enquadra como uma CPU de 8 bits contudo, suas instruções acumulam um total de 16 bits, que são distribuídos no processador em dois conjuntos de 8 bits. Apesar desse sistema possuir uma maior largura de dados na memória de instrução (16 bits), ele permite também que essa memória e a memória de dados possuam 8 bits de largura de endereço, aumentando sua capacidade, possibilitando melhor aproveitamento e facilitado o processo de programação em baixo nível executada nesse processador. Outra dificuldade encontrada foi que apesar de já haver um conhecimento prévio a esse trabalho do funcionamento do Logisim, ele não possui uma comunidade de usuários muito ativa como a comunidade que utiliza a plataforma Quartus, que era uma outra opção para criação da CPU.

No geral, foi uma experiência interessante de aprendizado sobre o funcionamento de um processador e dos níveis mais baixos de programação, onde foi possível aplicar de forma prática a teoria e conceitos vistos em sala de aula.

4 Referências

PATTERSON, D.; HENESSY, J. L. **Organização e projeto de computadores: a interface hardware/software.** 3ª Edição. São Paulo: Elsevier, 2005, 484 p.

THE GUIDE TO BEING A LOGISIM USER. **Logisim: a graphical tool for designing and simulating logic circuits**. Disponível em: <http://www.cburch.com/logisim/docs/2.7/en/html/guide/index.html>. Acesso em: 1 de dez. de 2022;