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

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR JUNIN

ALUNOS: João Eduardo Viana Leonel - 2021000679
Thiago Thomáz Santana de Nascimento - 2021021472
Vitor Jordão Carneiro Briglia - 2021013087

Dezembro de 2023
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 JUNIN

Dezembro de 2023
Boa Vista/Roraima

Resumo

Este trabalho aborda o processo de implementação de um processador de 16 bits. A arquitetura desse processador envolve a manipulação de dados e instruções em unidades de 16 bits, permitindo operações mais complexas do que processadores de 8 bits, embora com menor capacidade que arquiteturas de 32 ou 64 bits. Dentro desse processador, é possibilitada a utilização e execução de um total de 6 instruções.

O projeto começa com a definição da arquitetura, contemplando o conjunto de instruções, registradores, unidade de controle e lógica necessária para operações eficientes. A implementação desse design ocorre no ambiente de linguagens de descrição de hardware LogiSim, onde o design é traduzido em um circuito digital viável para fabricação.

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador **JUNIN**, um processador 16 bits que conta com 16 registradores e 6 instruções (tanto do tipo R, I e J), bem como a descrição detalhada de cada etapa da construção do processador.

1.1 Plataforma de desenvolvimento

Para a implementação do processador JUNIN foi utilizado a IDE: Logisim 2.7.1 por sua facilidade de visualizar o circuito em tempo real e praticidade de modificar problemas encontrados mais rapidamente.

Componente	Biblioteca	Simples	Exclusiva	Recursiva
Registrador 16bits	JUNIN ProcessadorFinal	0	16	16
UC	JUNIN ProcessadorFinal	1	1	1
Somador+1	JUNIN ProcessadorFinal	1	1	1
ULA	JUNIN ProcessadorFinal	1	1	1
BR	JUNIN ProcessadorFinal	1	1	1
Distribuidor	Conexão	12	19	49
Pino	Conexão	18	48	108
Túnel	Conexão	35	35	35
Clock	Conexão	1	1	1
Constante	Conexão	2	11	11
Porta AND	Portas	1	3	3
Porta OR	Portas	0	3	3
Multiplexador	Plexers	5	9	9
Demultiplexador	Plexers	0	2	2
Somador	Aritmética	1	3	3
Subtrator	Aritmética	0	1	1
Flip-Flop tipo D	Memória	0	16	256
Registrador	Memória	1	1	1
RAM	Memória	1	1	1
ROM	Memória	1	1	1
Rótulo	Base	4	4	4
TOTAL (sem subcircuitos do projeto)		82	158	488
TOTAL (com subcircuitos)		86	178	508

Figura 1 - Análise de circuitos gerado no logisim

1.2 Conjunto de instruções

O processador JUNIN possui 16 registradores. Assim como 6 formatos de instruções de 16 bits cada, Instruções do tipo R, I, J, seguem algumas considerações sobre as estruturas contidas nas instruções:

- **Opcode**: a operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
- **Reg1**: o registrador contendo o primeiro operando fonte e adicionalmente para alguns tipos de instruções (ex. instruções do tipo R) é o registrador de destino;
- **Imediato**: neste tipo de endereçamento, o operando é especificado diretamente no campo de endereço-base da instrução;

- **Reg2**: o registrador contendo o segundo operando fonte;
- **Reg3**: Registrador que armazena valores do tipo R.
- **Endereço**: um valor de endereço.

Tipo de Instruções:

- Formato do tipo R: Este formatado aborda instruções de Load (exceto *load Immediately*), Store e instruções baseadas em operações aritméticas.

Formato para escrita em código binário:

3 bits	4 bits	4 bits	4 bits	1 bits
15-13	12-9	8-5	4-1	0
Opcode	Reg1	Reg2	Reg3	Shamt

- Formato do tipo I: Este formato aborda instruções de Load, Store e BEQ (Instruções relacionadas à imediato).

Formato para escrita em código binário:

3 bits	4 bits	4 bits	4 bits	1 bits
15-13	12-9	8-5	4-1	0
Opcode	Reg1	Reg2	Reg3	Shamt

- Formato do tipo J: Este formato aborda a instrução Jump.

Formato para escrita em código binário:

3 bits	13 bits
15-13	12-0
Opcode	Endereço

Visão geral das instruções do Processador Junin:

O número de bits do campo **Opcode** das instruções é igual a 3, sendo assim obtemos um total $(Bit(0e1)^3 \therefore 2^3 = 8)$ de 8 **Opcodes (0-7)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

Opcode	Nome	Formato	Breve Descrição	Exemplo
000	LW	I	Load	**lw** $S0, Endereço memória
001	SW	I	Store	**Sw** $S0, Endereço memória
010	ADD	R	Soma	**add** $S0, $S1, ou seja, $S0 := $S0+$S1
011	SUB	R	Subtração	**sub** $S0, $S1, ou seja, $S0 := $S0-$S1
100	BEQ	I	Branch if Equal	**beq** $S0, $S1, Imediato
101	JUMP	J	Salto Incondicional	**jump** 0000
110	OPCode vazio (não implementado)			
111	OPCode vazio (não implementado)			

Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador Junin.

1.3 Descrição do Hardware

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

1.3.1 ALU ou ULA

Junin contém uma ULA que pode efetuar as operações de: Conjunção (00), Disjunção (01), Soma (10) e Subtração (11); Em operações aritiméticas, a ULA contém o componente ZERO integrado à ela para determinar se são valores iguais ou não. **Junin** possui 2 entradas de valores (valores 1 e 2 em binário de 16bits), um ENABLE (fio energizado que habilita a ULA), OPCode (Binário de 2bits que determina qual operação será realizada), e duas saídas: ZERO, OVERFLOW e SAÍDA (valor em binário de 16bits).

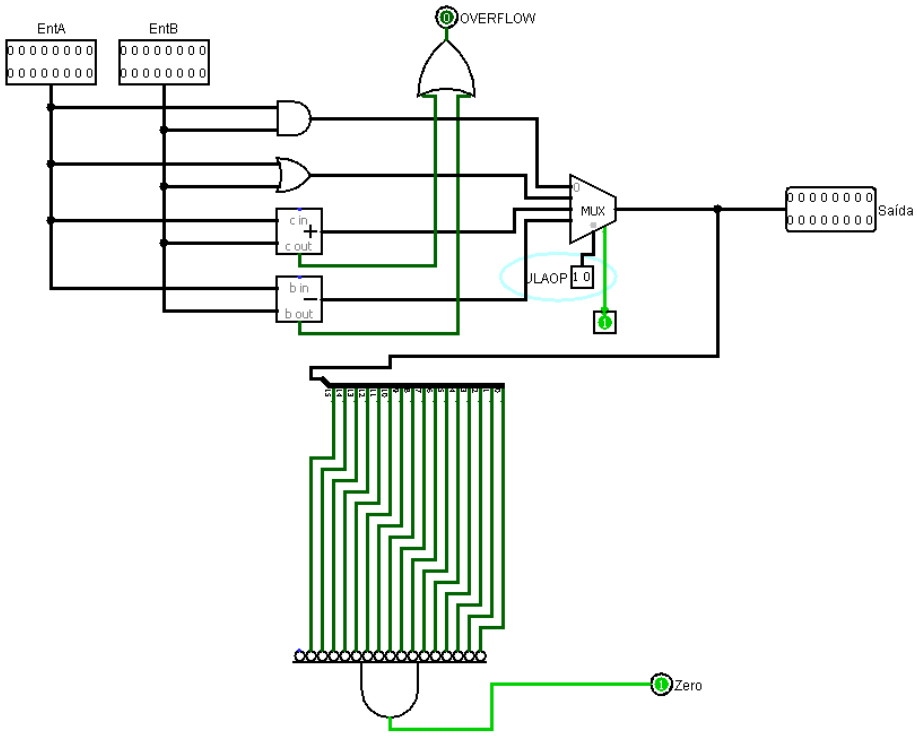

Figura 2 - Circuito interno da ULA.

OP ULA	OPERAÇÃO
00	AND
01	OR
10	ADD
11	SUB

Tabela 2 - OPCodes da ULA

1.3.2 Banco de Registradores (BR)

O banco de registradores é um componente digital composto por um conjunto de registradores que podem ser acessados de forma organizada. De uma maneira geral, podem ser executadas operações de leitura dos dados anteriormente gravados e de escrita de dados para modificar as informações internas. Este componente é um dos componentes mais importantes do fluxo de dados em um processador.

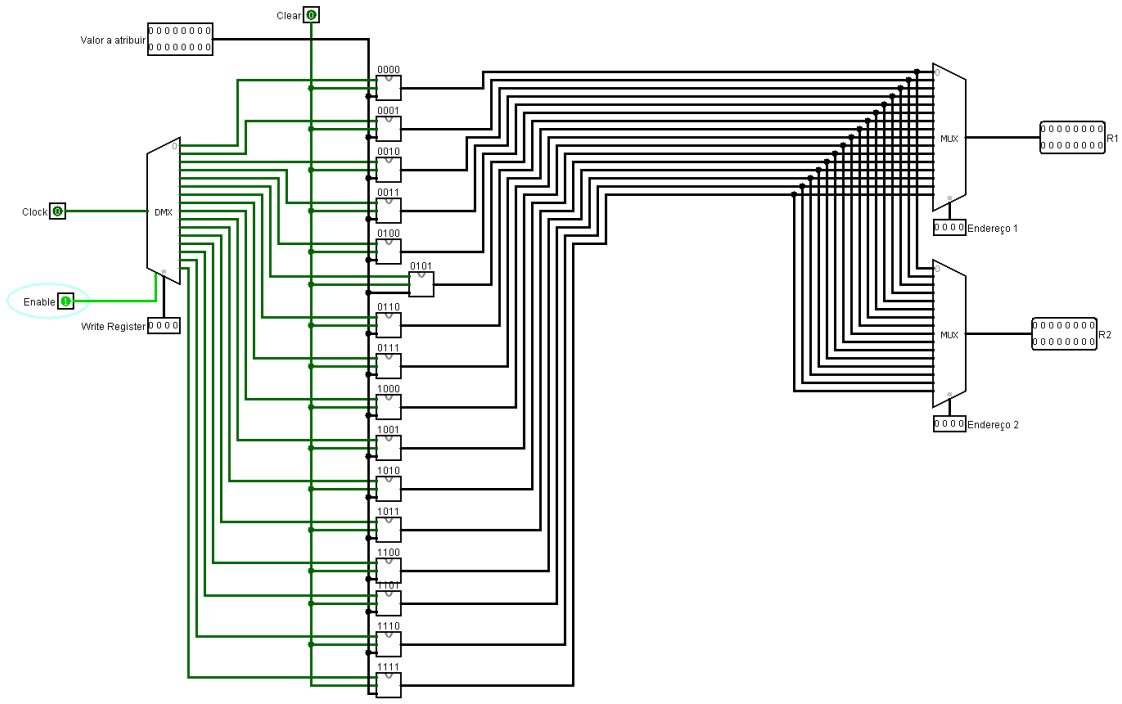

Figura 3 - Circuito interno do Banco de Registradores.

1.3.3 Clock

O clock tem apenas um pino, uma saída com largura de 1 bit, cujo valor representa o estado corrente do clock, os componentes que foram necessários para a entrada de clock estão conectados com um único clock.

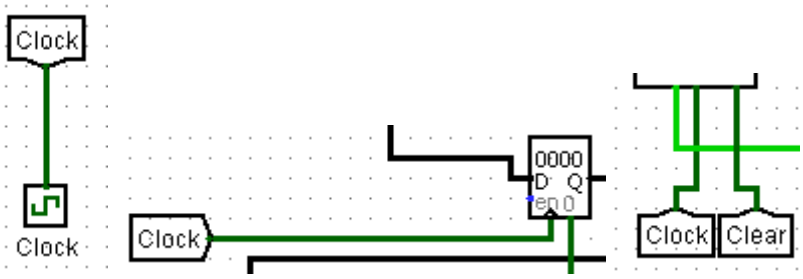

Figura 4 - Circuito do Clock.

1.3.4 Unidade de Controle

O componente Control tem como objetivo realizar o controle de todos os componentes do processador de acordo com o opcode. Esse controle é feito através das flags de saída abaixo:

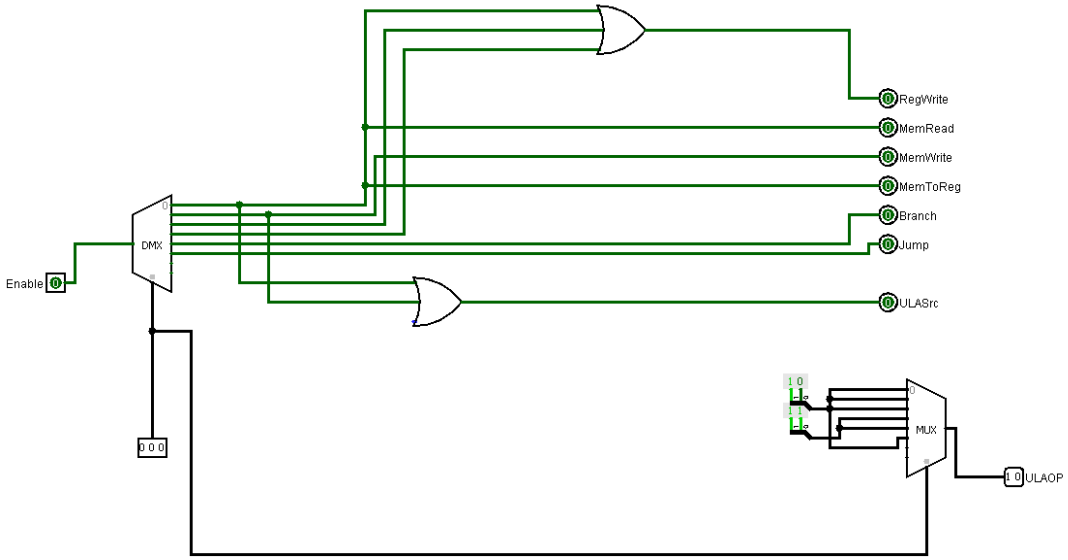

Figura 5 - Circuito interno da UC.

- **RegWrite**: 0,1.
- **MemRead**: 0,1.
- **MemWrite**: 0,1.
- **MemToReg**: 0,1.
- **Branch**: 0,1.
- **Jump**: 0,1.
- **ULASource:** 0,1.
- **ULAOP**: 10,11.

Abaixo segue a tabela, onde é feita a associação entre os opcodes e as flags de controle:

Tabela 3 - Detalhes das flags de controle do processador.

Comando	RegWrite	MemRead	MemWrite	MemToReg	Branch	Jump	ULASrc	ULAOP
LW	1	1	0	1	0	0	1	10
SW	0	0	1	0	0	0	1	10
ADD	1	0	0	0	0	0	0	10
SUB	1	0	0	0	0	0	0	11
BEQ	0	0	0	0	1	0	0	11
JUMP	0	0	0	0	0	1	0	10

1.3.5 Memória de dados (RAM)

O componente RAM, é o componente mais complexo nas bibliotecas predefinidas do Logisim. A memória RAM é uma memória de acesso rápido aleatório, que permite a leitura como a memória ROM e a escrita de arquivos. Diferentemente da memória de leitura, a RAM é volátil e tem suas informações perdidas com o desligamento da energia.

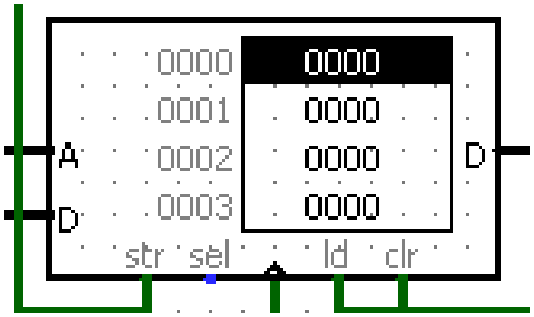

Figura 6 - Memória RAM.

1.3.6 Memória de Instruções (ROM)

A memória de instruções, é o componente responsável por armazenar os passos e instruções relativas aos mesmos. Será a ROM que enviará para o divisor uma instrução associada ao passo recebido pelo PC. Este tipo de memória é não-volátil, ou seja, ela não perde os dados com o desligamento da energia.

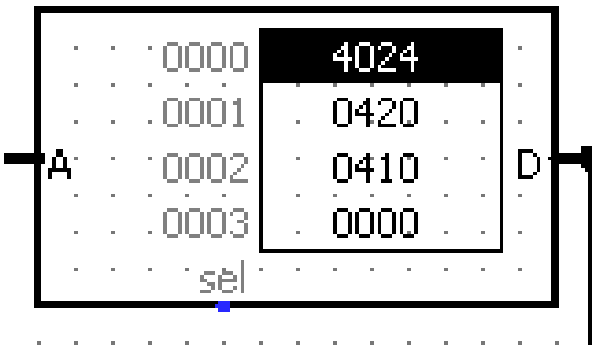

Figura 7 - Memória ROM.

1.3.7 Somador

O circuito somador completo pode ser representado por três entradas, A, B e Carry de entrada, ou Carry In, que são somados e obtemos o resultado da soma, ou sinal S de saída, e Carry de Saída, ou Carry Out. A tabela verdade pode ser representada conforme a tabela a seguir para diferentes valores de entrada.

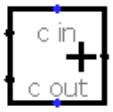

Figura 8 - Somador.

1.3.8 And

É uma porta lógica em dois operandos que resulta em um valor lógico verdadeiro somente se todos os operado tem um valor verdadeiro.

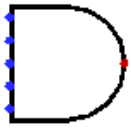

Figura 9 - AND.

1.3.9 Mux_2x1

Os multiplexadores é um seletor utilizado para definir qual dos valores de entrada será o valor de saída, utilizando como endereço de seleção, flags da unidade de controle.

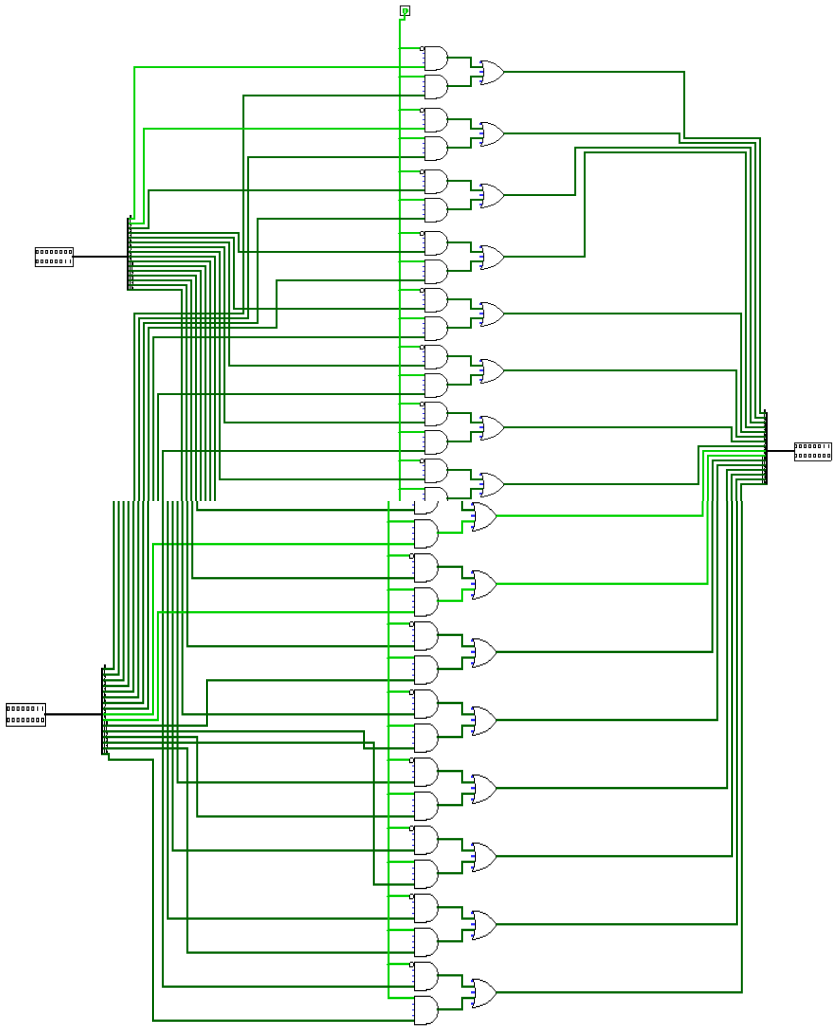

Figura 10 - Multiplexador.

1.3.10 PC

O PC é um registrador flip-flop tipo D que armazena o endereço da instrução que está sendo executada no momento.

Figura 11 - Program Counter (PC).

1.3.11 ZERO

O Zero fica dentro da ULA, e é utilizado apenas no caso das operações comparativas. Sua função é apenas inicializar a flag necessária para realizar a comparação. Ele recebe todos os valores de saída do cálculo e o overflow do subtrator,

passando tudo por um AND negando as entradas, caso todos os valores forem 0, a saída Zero é positiva.

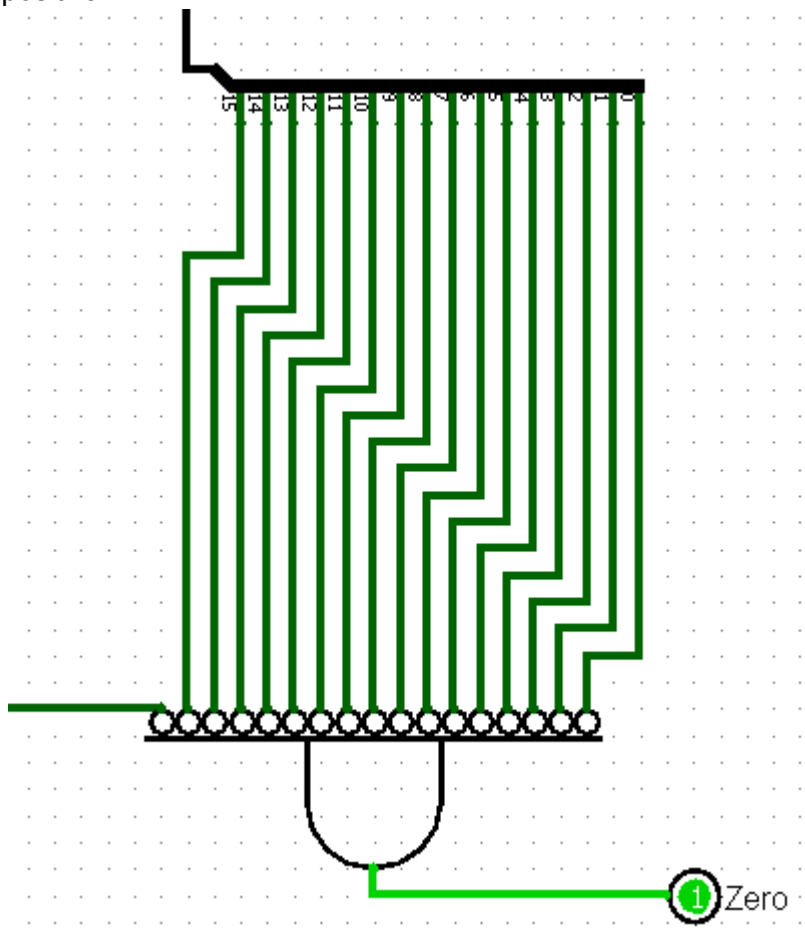

Figura 12 - Zero (criado na plataforma Logisim).

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. A unidade de controle desempenha um papel crucial ao gerenciar as operações do sistema. Ela interpreta as instruções recebidas, decodifica-as e determina as ações específicas que cada unidade funcional precisa executar para realizar tarefas como cálculos, manipulação de dados ou operações lógicas. É como se fosse o "gerente" do sistema, coordenando e direcionando o fluxo de trabalho para garantir que as operações sejam realizadas corretamente e no momento adequado, de acordo com as instruções processadas.

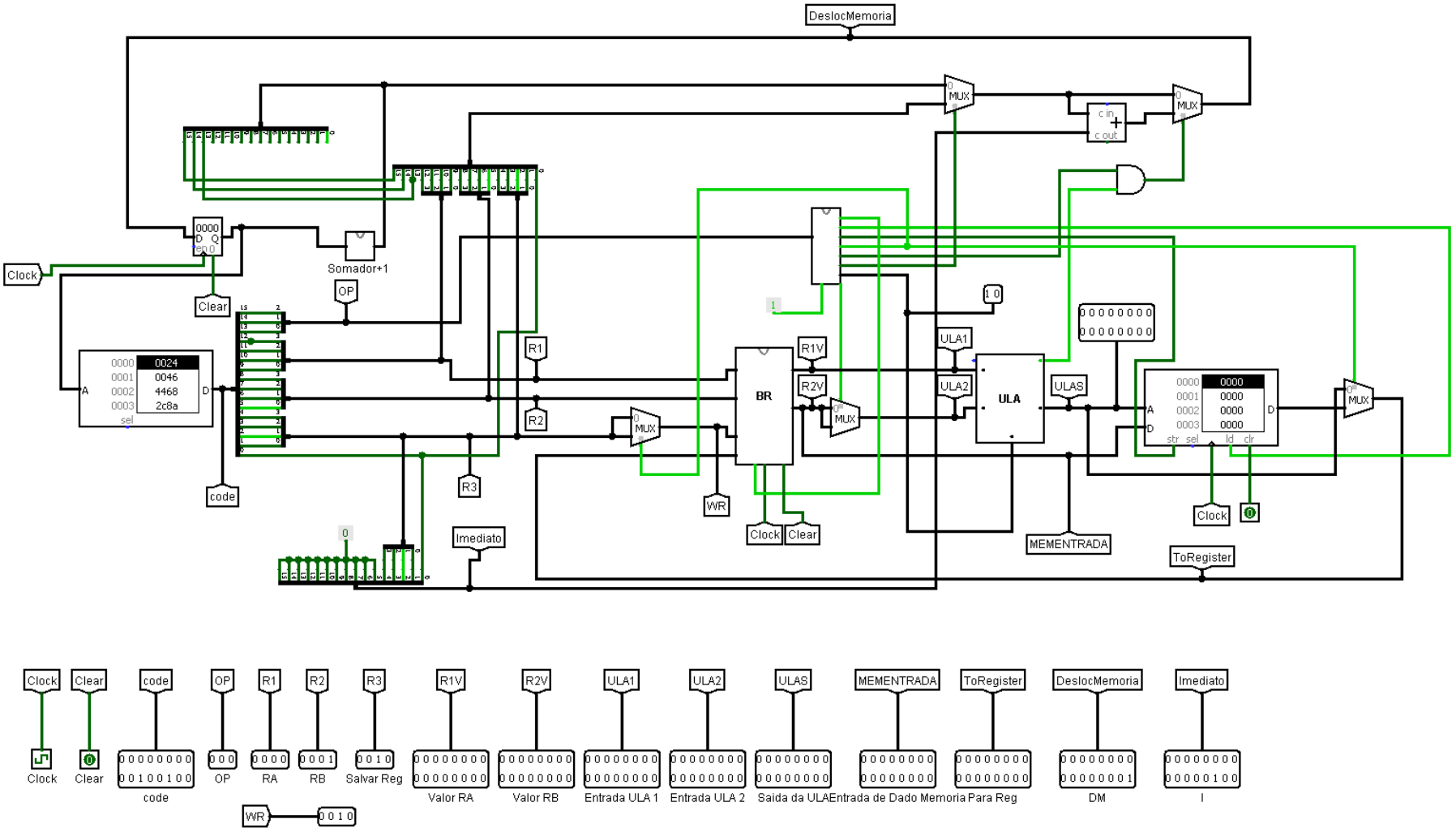

Figura 13 - Datapath (completo criado na plataforma Logisim).

2 Simulações e Testes

Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em especifico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador **Junin** utilizaremos como exemplo um código simples que realiza todas as operações que nosso processador suporta.

Tabela 4 - Teste realizado no Junin

Endereço	Linguagem de Alto Nível	Binário				
		Opcode	Reg1	Reg2	Reg3	Shamt
			Endereço			
			Dado			
0000	**lw $R3, $R0, $R1**	000	0000	0001	0010	0
0001	**lw $R4, $R0, $R3**	000	0000	0010	0011	0
0002	**add $R5, $R3, $R4**	010	0010	0011	0100	0
0003	**sw $R6, $R7, $R5**	001	0110	0100	0101	0
0004	**sub $R8, $R6, $R3**	011	0101	0010	0111	0
0005	**beq $R3, $R4, $R9**	100	0010	0011	1000	1
0017	**jump 0031**	101	0000000110001			
0031	**jump 0005**	101	0000000000101			
0005	**beq $R3, $R4, $R9**	100	0010	0011	1000	1
0017	**jump 0031**	101	0000000110001			
0031	**jump 0005**	101	0000000000101			

Descrição dos testes

```
       code              RA      RB      Salvar reg      Valor a atribuir
0000 0000 0010 0100     0000    0001      0010        0000 0000 0000 0001
0000 0000 0100 0110     0000    0010      0011        0000 0000 0000 0001
0100 0100 0110 1000     0010    0011      0100        0000 0000 0000 0010
0010 1100 1000 1010     0110    0100      0101        0000 0000 0000 0010
0110 1010 0100 1110     0101    0010      0111        0000 0000 0000 0001
1000 0100 0111 0001     0010    0011      1000        0000 0000 0000 0000
1010 0000 0011 0001     0000    0001      1000        0000 0000 0000 0000
1010 0000 0000 0101     0000    0000      0010        0000 0000 0000 0000
1000 0100 0111 0001     0010    0011      1000        0000 0000 0000 0000
1010 0000 0011 0001     0000    0001      1000        0000 0000 0000 0000
1010 0000 0000 0101     0000    0000      0010        0000 0000 0000 0000
1000 0100 0111 0001     0010    0011      1000        0000 0000 0000 0000
```

Figura 14 - Registro de teste 1/3

Valor a atribuir	Valor RA	Valor RB
0000 0000 0000 0001	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0001	0000 0000 0000 0000	0000 0000 0000 0001
0000 0000 0000 0010	0000 0000 0000 0001	0000 0000 0000 0001
0000 0000 0000 0010	0000 0000 0000 0000	0000 0000 0000 0010
0000 0000 0000 0001	0000 0000 0000 0010	0000 0000 0000 0001
0000 0000 0000 0000	0000 0000 0000 0001	0000 0000 0000 0001
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0000	0000 0000 0000 0001	0000 0000 0000 0001
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0000	0000 0000 0000 0001	0000 0000 0000 0001

Figura 15 - Registro de teste 2/3

Entrada ULA1	Entrada ULA2	Saída da ULA
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0000	0000 0000 0000 0001	0000 0000 0000 0001
0000 0000 0000 0001	0000 0000 0000 0001	0000 0000 0000 0010
0000 0000 0000 0000	0000 0000 0000 0010	0000 0000 0000 0010
0000 0000 0000 0010	0000 0000 0000 0001	0000 0000 0000 0001
0000 0000 0000 0001	0000 0000 0000 0001	0000 0000 0000 0000
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0001	0000 0000 0000 0001	0000 0000 0000 0000
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0000	0000 0000 0000 0000	0000 0000 0000 0000
0000 0000 0000 0001	0000 0000 0000 0001	0000 0000 0000 0000

Figura 16 - Registro de teste 3/3

O código realizado apresenta as funções suportadas pelo processador, sendo eles: LW, SW, ADD, SUB, BEQ, JUMP. Nesse teste ele realiza o ‘load’ de 2 valores pré-definidas na memória RAM, somas os 2, realiza um store na memória RAM no ‘0002’ e então realiza uma subtração (sem salvar na memória RAM); A partir daqui é feito um loop de saltos, um sendo o salto condicional de igualdade ativado por serem iguais e 2 (dois) JUMP realizados seguidos, sendo que o primeiro aciona o PC para a instrução do próximo JUMP e o segundo aciona o PC para a instrução do BEQ.

3 Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 16 bits denominado de **Junin**. Inicialmente o processador foi algo difícil de ser pensado, mas com o tempo as ideias de como executar foram surgindo depois de um longo esforço de pesquisas e testes. A atividade de Laboratório de Circuitos elaborada pelo professor Herbert Oliveira, foi de grande ajuda para entender como funcionam os componentes de um processador.

O projeto possui algumas limitações, porém com espaço para atualizações, como transformar o bit alocado para a função 'SHAMT' como um novo bit para o UC e OpULA, conseguindo assim 8 opções de funções a mais, como LI e ADDI, e 8 novas operações para a ULA, como Multiplicação e Divisão.

O processador serve seu quesito de ser um componente didático eficaz e ainda possui margem de evolução para um componente de uso real.