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

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR 8 BITS

ALUNOS:

Wanderson Morais de Sousa
Gilberto Alexsandro Almeida Pessoa

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 8 BITS

Dezembro de 2022
Boa Vista/Roraima

Resumo

Este trabalho aborda o projeto e implementação dos componentes de um processador CPU de 8 bits, utilizando do software logisim.

O relatório então servirá de instrumento de avaliação dos alunos para a disciplina de AOC (Arquitetura e Organização de Computadores), ministrada pelo professor Herbert Oliveira Rocha.

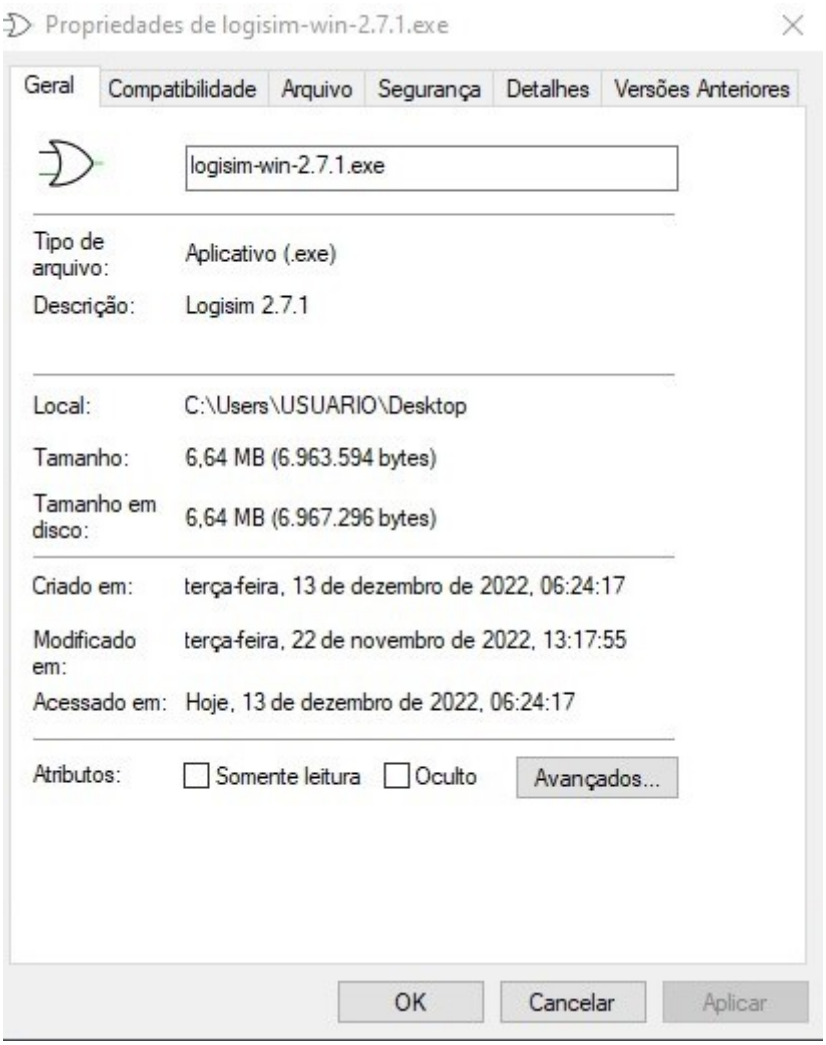
- ## Especificação

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

- ## Plataforma de desenvolvimento

Para a implementação do processador foi utilizado o Software: Logisim

Figura 1 - Especificações no Logisim



- ## Conjunto de instruções

O processador de 8 bits possui 4 registradores: $s0 $s1 $s2, $s3. Assim como 8 formatos de instruções de 8 bits cada, Instruções do tipo R (de registradores),I(de imediato),J(Jump), 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;
- **Reg2**: o registrador contendo o segundo operando fonte;

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 os teste:

Tipo da Instrução	Reg1	Reg2

Formato para escrita em código binário:

3 bits	2 bits	2 bits	1 bit
Opcode	Reg1	Reg2	vago

- Tipo I: Este tipo de instrução aborda carregamentos diretos na memória.

Formato-I:

3 bits	2 bits	2 bits	1 bit
Opcode	Reg1	num (imediato ou endereço)	vago

- Tipo J: Este tipo de instrução é responsável por desvios condicionais eincon-dicionais.

Formato-J:

3bits	5 bits
Opcode	num (Imediato ou Endereço)

Visão geral das instruções do Processador :

O número de bits do campo **Opcode** das instruções é igual a três, sendo assim obtemos um total () de 8 instruções **Opcodes (0-7)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

MNEMÔNICO	OPERANDOS	OPCODE	SIGNIFICADO
MOVE	Reg1, Reg2	000	Reg1<- Reg2
ADD	Reg1, Reg2	001	Reg1<-Reg1+Reg2

SUB	Reg1, Reg2	010	Reg1<-Reg1-Reg2
MULT	Reg1, Reg2	011	Reg1<-Reg1*Reg2
LW	Reg1,num	100	Reg1<-[Rx+num]
SW	Reg1,num	101	[Rx+num]<-Reg1
BEQ	Reg1, Reg2	110	Reg1 == Reg2, Label
J	num	111	PC<-num

Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador de 8 bits.

- ## Descrição do Hardware

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

- ### ALU ou ULA

O componente QALU (Q Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: soma, subtração, divisão (considerando apenas resultados inteiros) e multiplicação. Adicionalmente o QALU efetua operações de comparação de valor como maior ou igual, menor ou igual, somente maior, menor ou igual. O componente QALU recebe como entrada três valores: **A** – dado de 8bits para operação; **B** - dado de 8bits para operação e **OP** – identificador da operação que será realizada de 4bits. O QALU também possui três saídas: **zero** – identificador de resultado (2bit) para comparações (1 se verdade e 0 caso contrário); **overflow** – identificador de overflow caso a operação exceda os 8bits; e **result** – saída com o resultado das operações aritméticas.

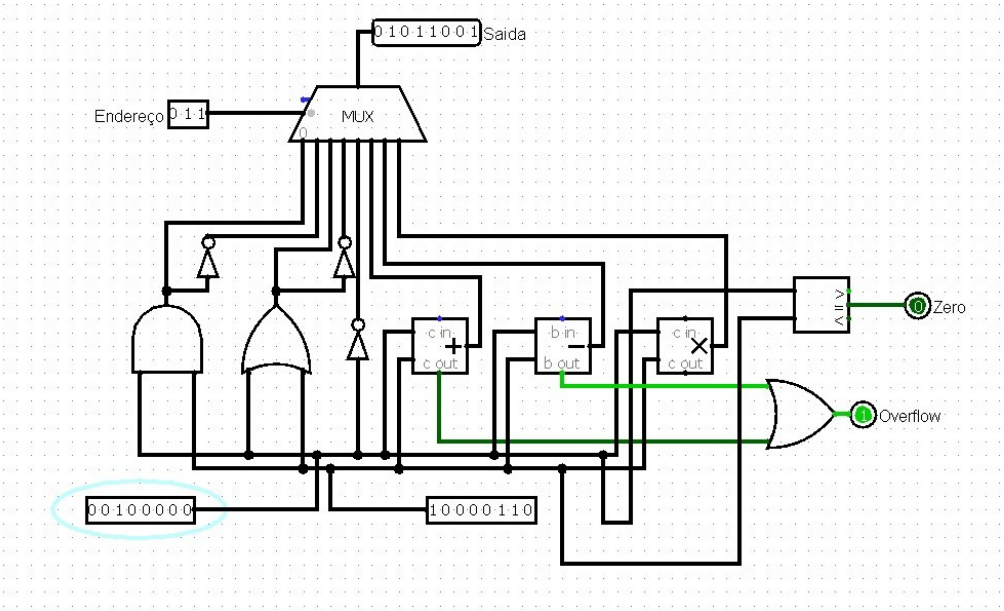


Figura 2 -ULA criada para utilizar no processador

- **Banco de Registradores**

[Todo] Descrição

- **Clock**

O clock que utilizamos no processador foi o proprio do Logisim e para fazer os os testes acionamos ele de forma manual.

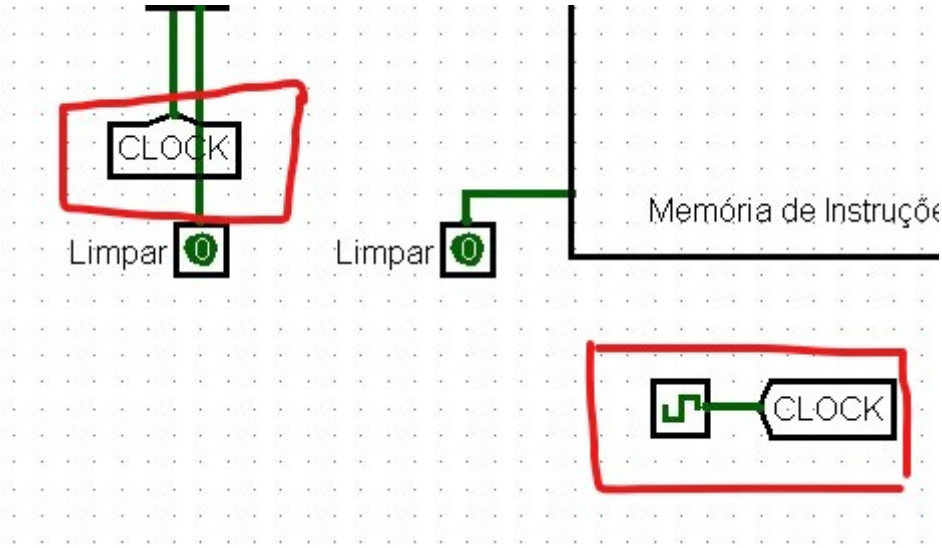


Figura 3 - Imagem do Clock utilizado

- **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:

- **Jump;**
- **Branch;**
- **MemtoReg;**
- **AluOP;**
- **MemWrite/ MemRead;**
- **AluSrc;**
- **RegWrite;**

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

Tabela 2 - Detalhes das flags de controle do processador.

Instruções	Jump	Branch	MemtoReg	AluOp	MemWrite/ MemRead	AluSrc	ReadWrite
MOVE	0	0	01	000	0	0	1
ADD	0	0	01	101	0	0	1
SUB	0	0	01	110	0	0	1
MULT	0	0	01	111	0	0	1
LW	0	0	10	000	0	0	1
SW	0	0	00	000	1	0	0
BEQ	0	1	00	110	0	0	0
JUMP	1	0	00	000	0	0	0

- **Memória de dados**

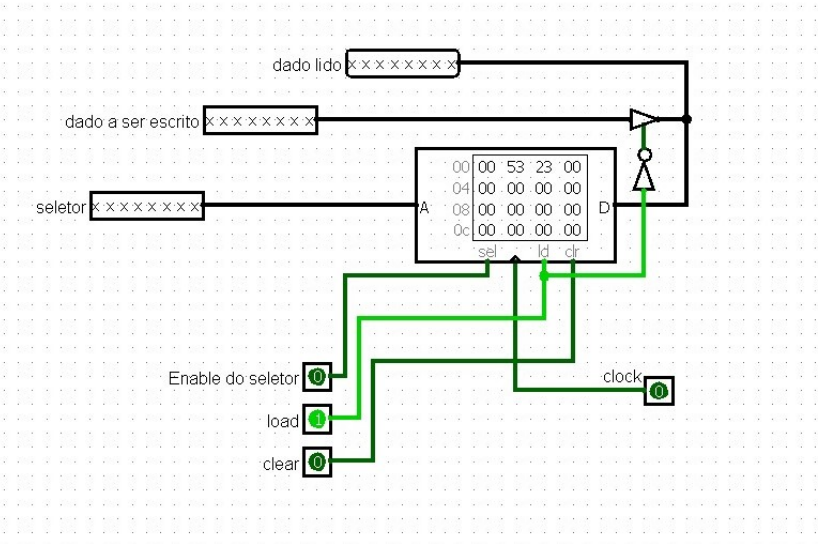


Figura 4 - Memória de dados (RAM)

- **Memória de Instruções**

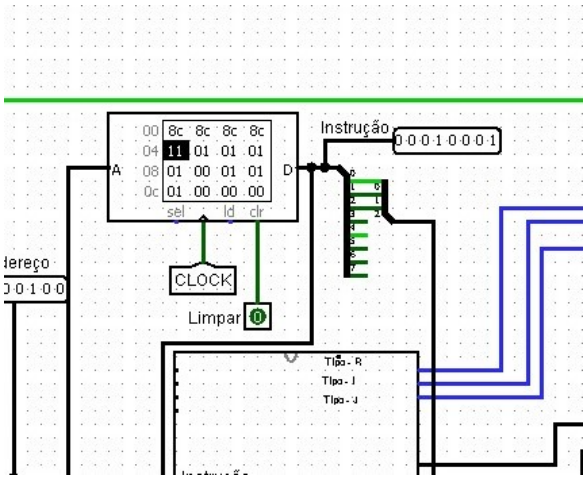


Figura 5 - Memória de Instruções

- **Somador**

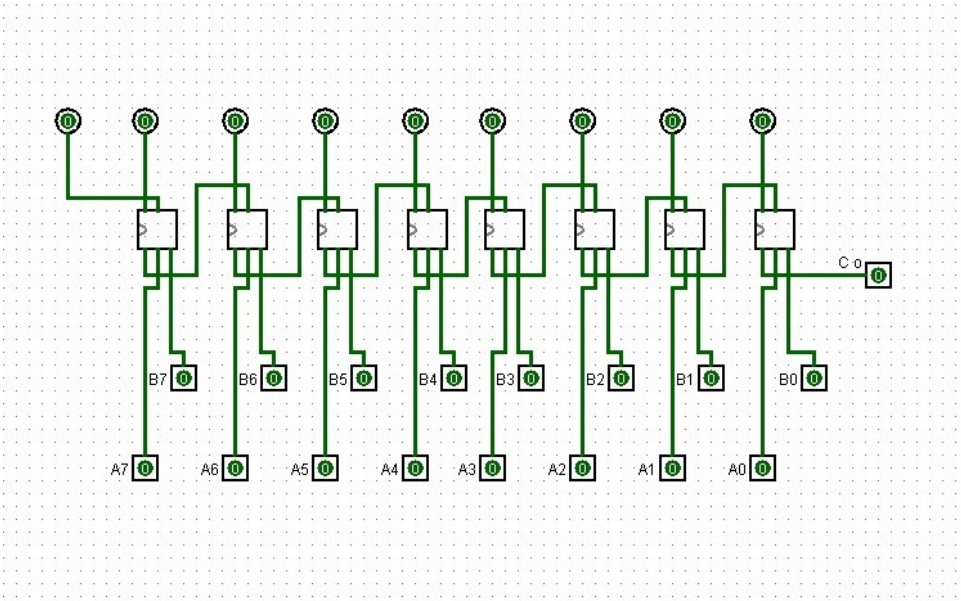


Figura 6 - Somador

- **And**

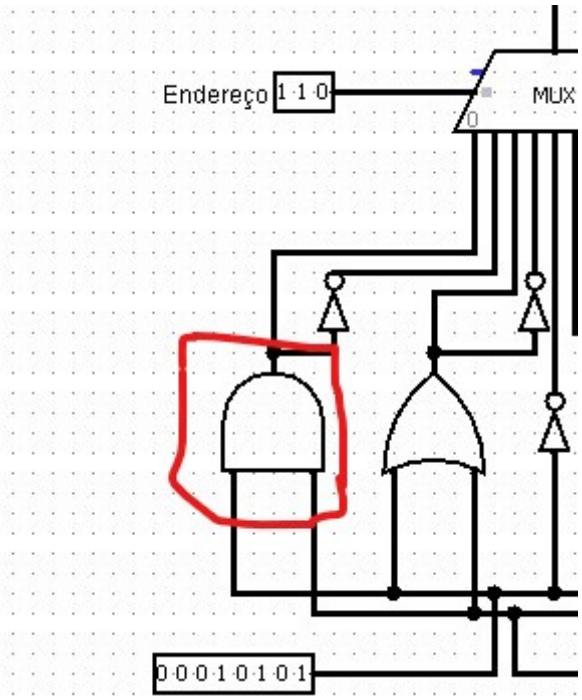


Figura 7 - And da ULA.

- **Mux_2x1**

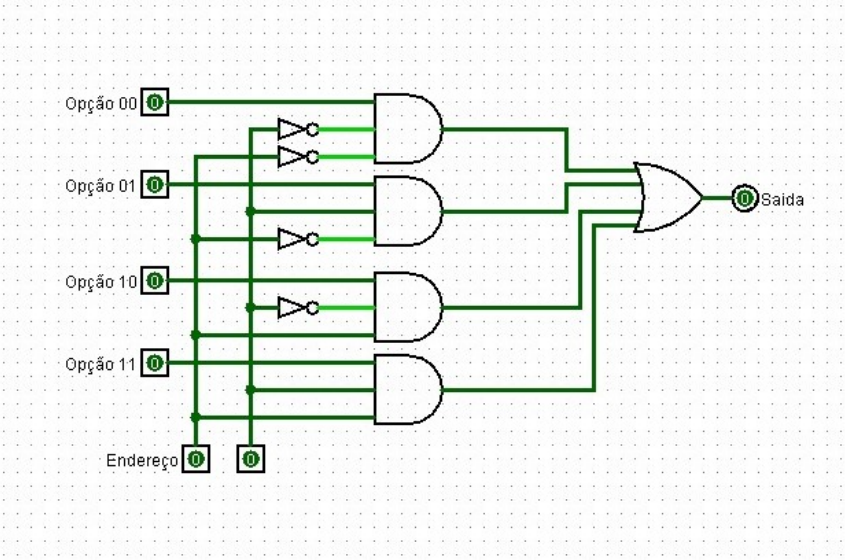


Figura 8 - Multiplexador 2x1

- **PC**

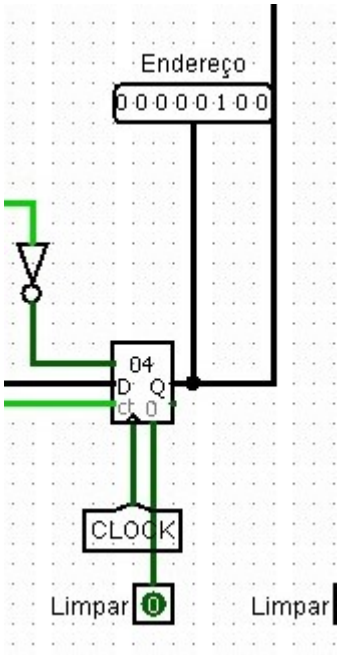


Figura 9 - Pc

- **ZERO**

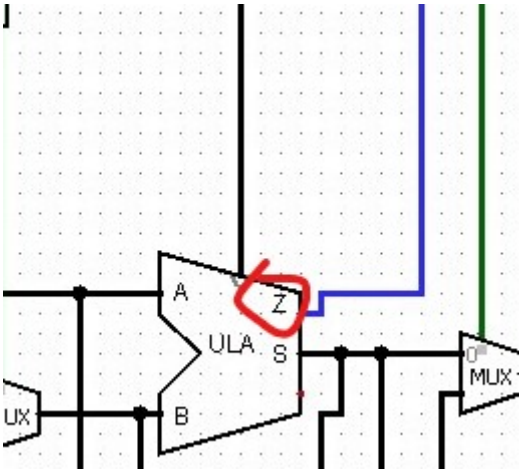


Figura 10 - Zero

- ## Datapath

É a conexão entre as unidades funcionais formando um único caminho de dados e acrescentando uma unidade de controle responsável pelo gerenciamento das ações que serão realizadas para diferentes classes de instruções.

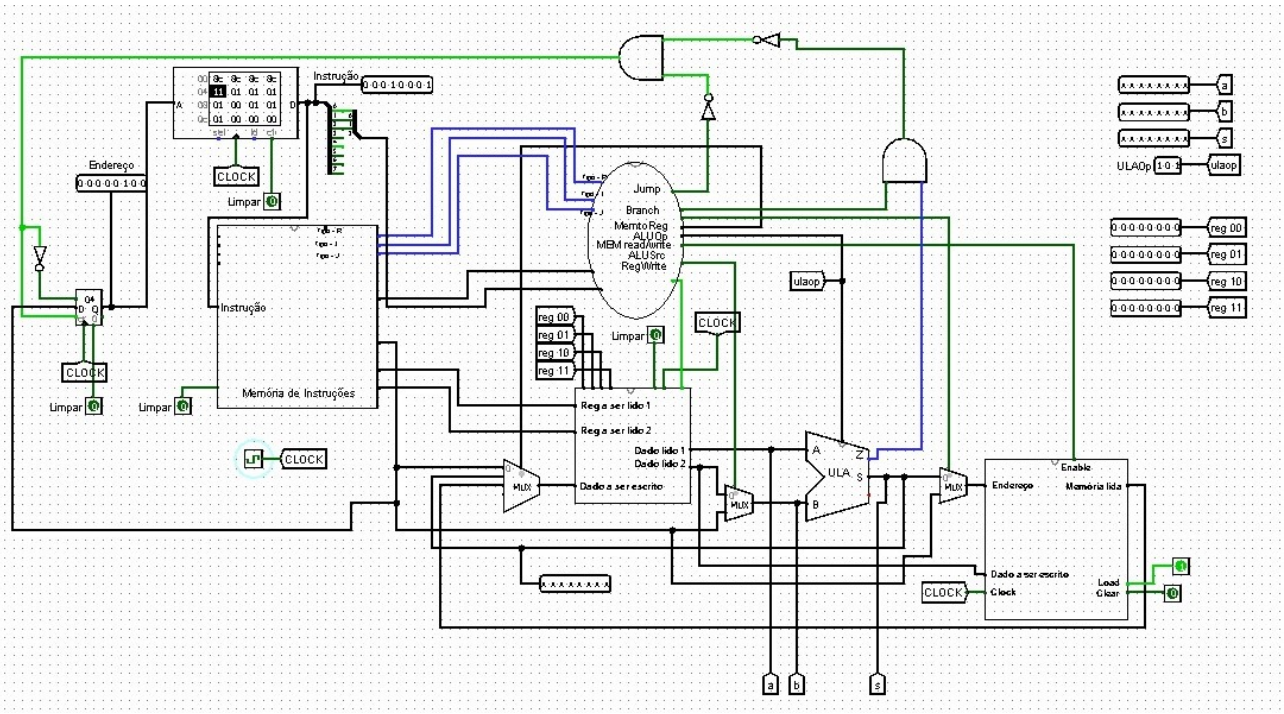


Figura 10 - Datapath construido.

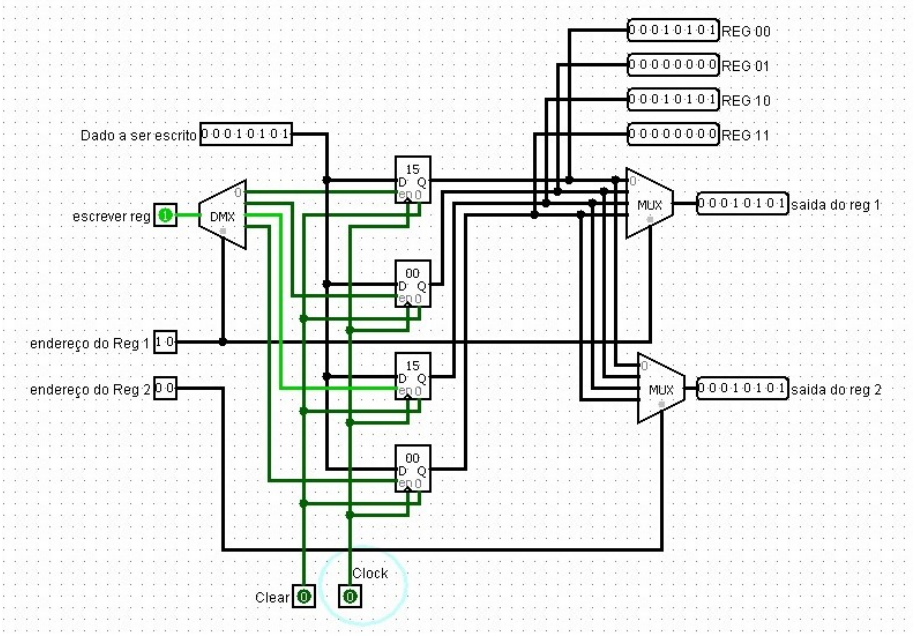
- ## Simulações e Testes

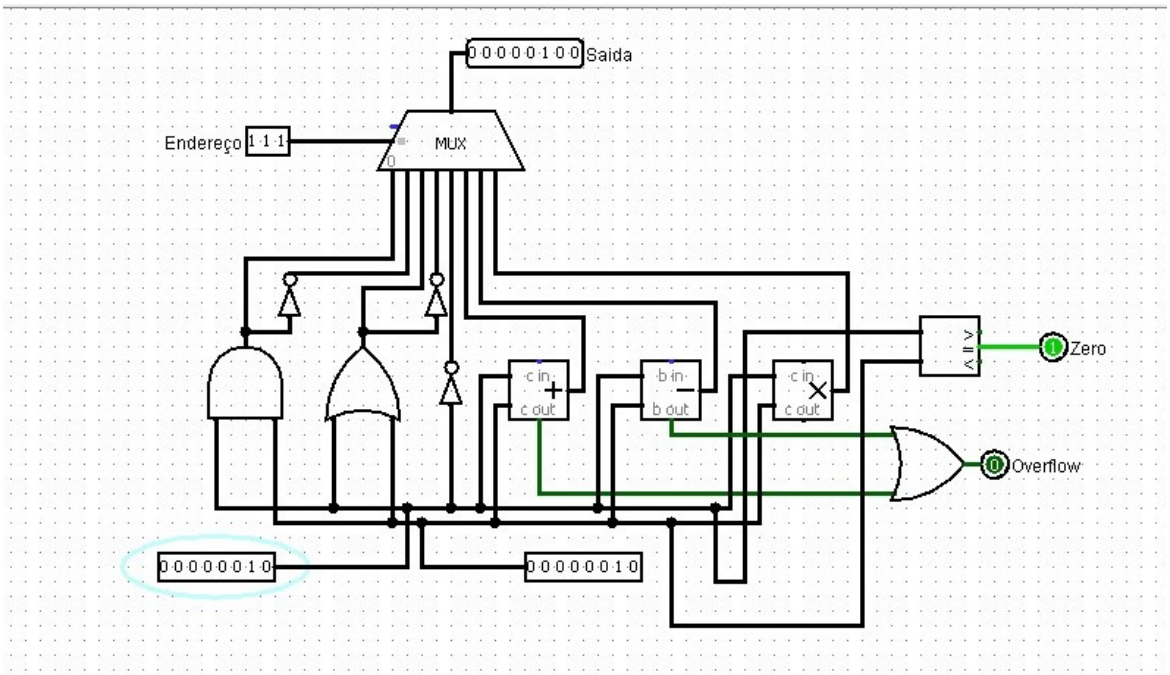
Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em especifico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do

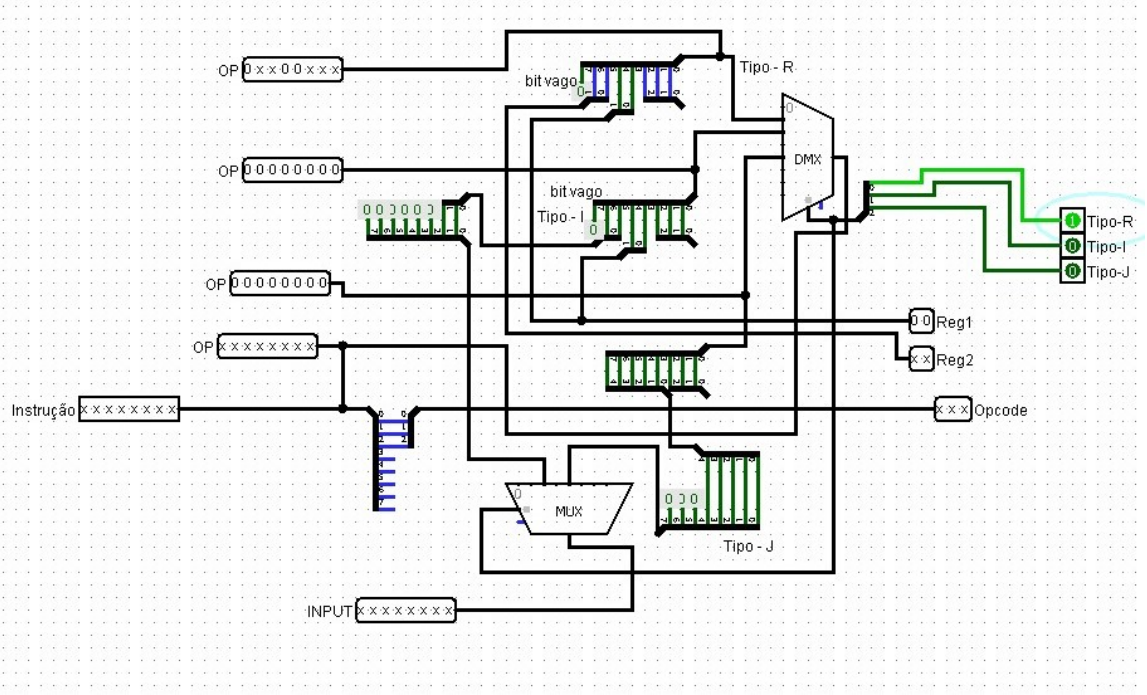
processador de 8 bits. Para fazer o teste é escrito dois dígitos na primeira memoria RAM esses digitos estão em hexadecimal, mas depois ele é transformado em binario e passado para a memoria de intruções. Abaixo veremos os testes feitos.

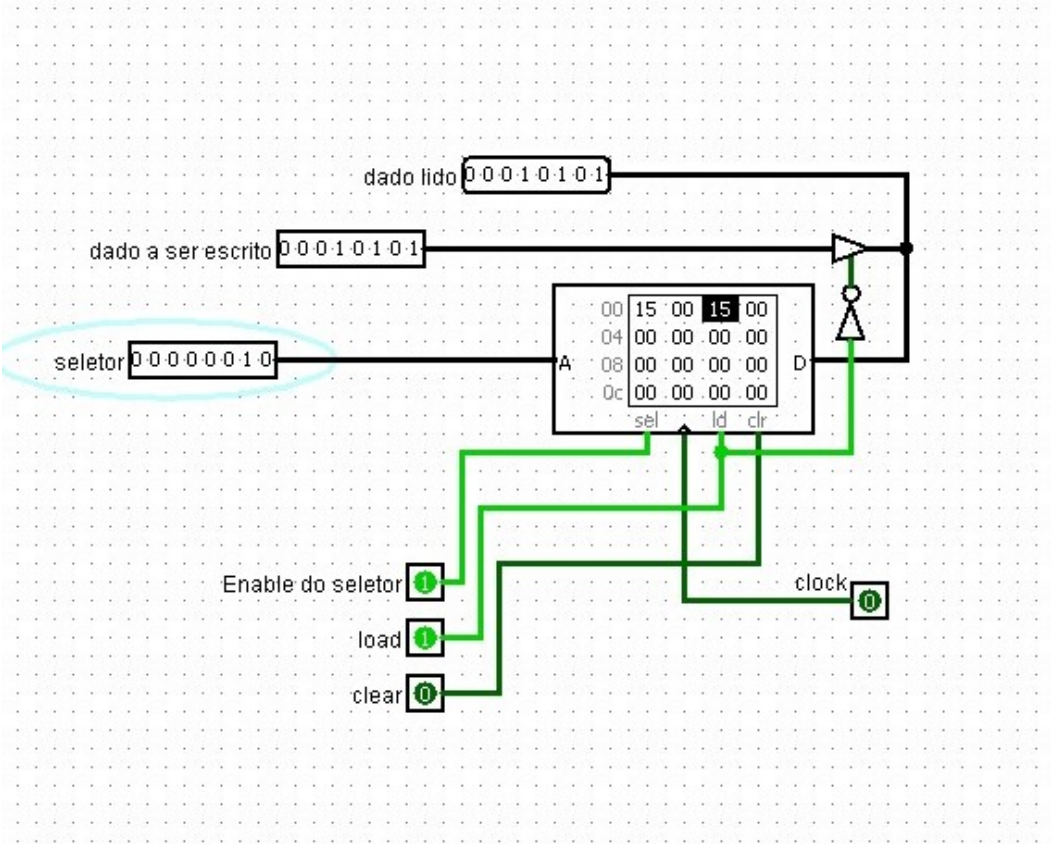
Verificação dos resultados no relatório da simulação: Não conseguimos fazer os testes, mas montamos um prototipo de um processador. Então decidimos que aqui colocaremos os testes dos componentes internos.

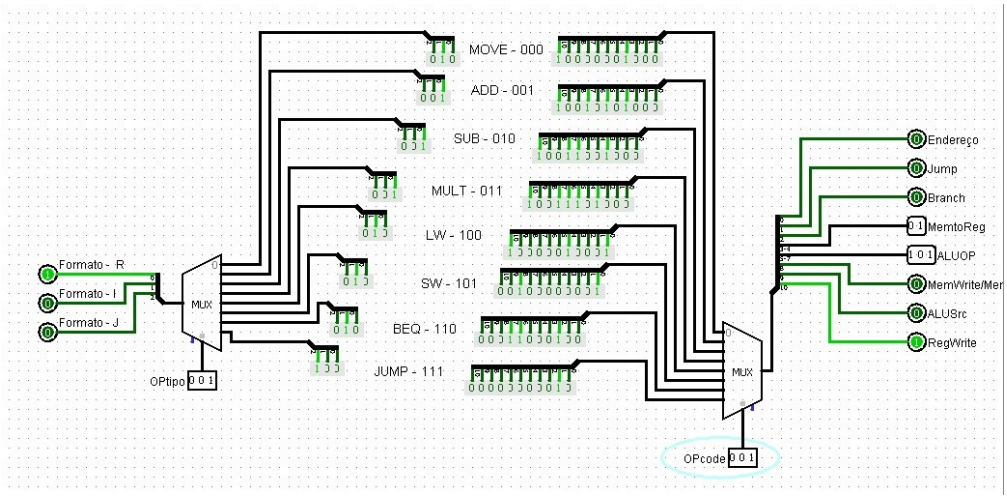
Testes dos componentes internos do processador:






OP
bit vago
Tipo - R
OP
DMX
bit vago
Tipo - I
Tipo-R
Tipo-I
Tipo-J
OP
Reg1
OP
Reg2
Instrução
Opcode
MUX
Tipo - J
INPUT


dado lido
dado a ser escrito
seletor
A
D
sel
ld
clr
Enable do seletor
load
clear
clock



- **Considerações finais**

Este projeto mostrou os desafios de trabalhar com hardware, também nos fez a aprender muito e buscar por conhecimento afim de concluir o projeto. Por conta de da dificuldade enfrentada por trabalhar pela primeira vez com esse tipo de hardware, acabamos não conseguindo fazer rodar da forma devida, mas os componentes internos estão funcionando como mostrado nos teste, isso com certeza nos mostra que ainda temos um vasto caminho que podemos percorrer, nos aprofundar. Muitos conceitos foram absorvidos e serão levados conosco para enfrentar futuros desafios.

- **Repositório**

https://github.com/GilbertoAlexsandro/DCC301_Projeto__Final_AOC_GilbertoAlexsandro_WandersonMorais_UFRR_2022