UNIVERSIDADE FEDERAL DO CARIRI
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO GUILHERME CESÁRIO ALCÂNTARA
KARLA MIKAELLY PAZ DE ALMEIDA

ARQUITETURA DE COMPUTADORES

Processador desenvolvido no MIPS e Logisim

JUAZEIRO DO NORTE - CE
2024

FRANCISCO GUILHERME CESÁRIO ALCÂNTARA
KARLA MIKAELLY PAZ DE ALMEIDA

ARQUITETURA DE COMPUTADORES

Processador desenvolvido no MIPS e Logisim

Trabalho de Arquitetura de Computadores
no qual foi desenvolvido um processador
usando as ferramentas logisim e MARS-MIPS.

JUAZEIRO DO NORTE - CE
2024

SUMÁRIO

1. INTRODUÇÃO

Este relatório tem como objetivo explicar a implementação da parte 2 do trabalho avaliativo da disciplina de arquitetura de computadores do curso de Ciência da Computação da Universidade Federal do Cariri - UFCA. A avaliação consiste em desenvolver um processador que realiza uma multiplicação de duas matrizes, usando as ferramentas Logisim e MARS-MIPS. Toda a lógica de funcionamento do processador é construída no logisim, por meio de circuitos combinacionais e outras ferramentas (como memória de dados e de instruções), enquanto que as instruções que são executadas foram baseadas no funcionamento do MARS-MIPS.

O projeto foi dividido em duas partes, uma para cada versão do processador. A primeira versão é o processador monociclo, que executa apenas uma instrução por ciclo, enquanto que a segunda versão do trabalho implementa o pipeline. As mesmas ferramentas e instruções foram utilizadas para desenvolver as duas versões.

2. ESTRATÉGIA DE IMPLEMENTAÇÃO

A implementação das duas versões do projeto contempla o desenvolvimento de um circuito combinacional no logisim e de um conjunto de instruções a serem executadas sequencialmente a fim de se realizar a multiplicação de duas matrizes.

Para saber quais instruções o processador deve realizar, primeiro foram desenvolvidos dois códigos base de multiplicação de matrizes. O primeiro código realiza multiplicação de duas matrizes e foi escrito na linguagem de programação C. Segue abaixo o código em C:

```c
#include <stdio.h>

#define ROW1 2
#define COL1 2
#define ROW2 2
#define COL2 2

void matrix_multiplication(int mat1[][COL1], int mat2[][COL2], int result[][COL2]) {
    int i, j, k;

    for(i = 0; i < ROW1; i++) {
        for(j = 0; j < COL2; j++) {
            for(k = 0; k < COL1; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
}
```

Figura 1: imagem do código de multiplicação de matrizes escrito em C

Com base no código acima, foi escrito um outro programa em Assembly MIPS, com a mesma

finalidade de realizar a multiplicação de duas matrizes. Um trecho do programa escrito em Assembly MIPS segue abaixo:

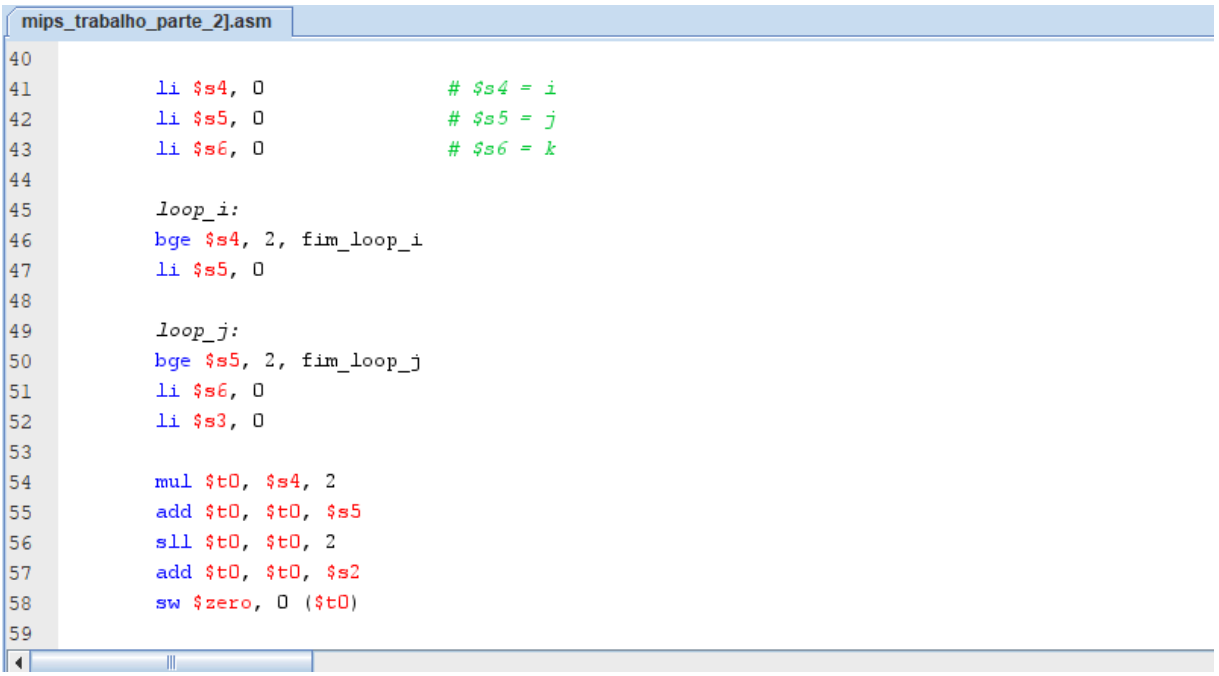

Figura 2: imagem do código de multiplicação de matrizes escrito em Assembly MIPS

Os códigos em C e Assembly podem ser acessados e visualizados sem mais detalhes em https://github.com/MikaellyL/MyProcessor.

Com base nas instruções do MIPS, foram definidas quais as instruções o processador do projeto seria capaz de reconhecer. Por exemplo, a instrução “add $t0, $t0, $s2”, seria escrita “09910000” em hexadecimal, que tem o formato “0000 1001 1001 0001 0000 0000 0000 0000” em binário. Essa instrução significa que deve ser realizada uma operação de soma (identificada como “0000”, os primeiros 4 bits da instrução), armazenando no registrador 9 (“1001”) o resultado da soma do que tem no registrador 9 (“1001”) mais o que tem no registrador 1 (“0001”). A tabela abaixo mostra a correspondência entre as instruções, seus formatos e seus números de operação:

operação	op_code	Formato da instrução				
add	0000	op_code	RW	R1	R0	-
mul	0010	op_code	RW	R1	R0	-
addi	0011	op_code	RW	-	R0	imediato
li	0110	op_code	RW	-	-	imediato
bge	0101	op_code	-	R1	R0	imediato

j	0100	op_code	-	-	-	imediato
sw	0111	op_code	-	R1	R0	-
lw	1000	op_code	RW	-	R0	-

No logisim, todas essas instruções são escritas em hexadecimal, e não em binário. Portanto, comandos de soma, multiplicação, armazenamento na memória podem ser exemplificados da seguinte forma:

- soma: 01190000
- multiplicação: 2ffe0000
- armazenamento na memória: 702e0000

3. PARTE 1: PROCESSADOR MONOCICLO

O modelo monociclo caracteriza-se pelo fato de que ele executa uma instrução por vez. Em outras palavras, a próxima instrução só começa a ser executada quando a atual termina. Sua implementação é consideravelmente mais simples de ser realizada, devido ao menor número de registradores e verificações envolvidos no processo de execução de uma instrução. A imagem abaixo ilustra como essa versão do processador foi implementada no logisim.

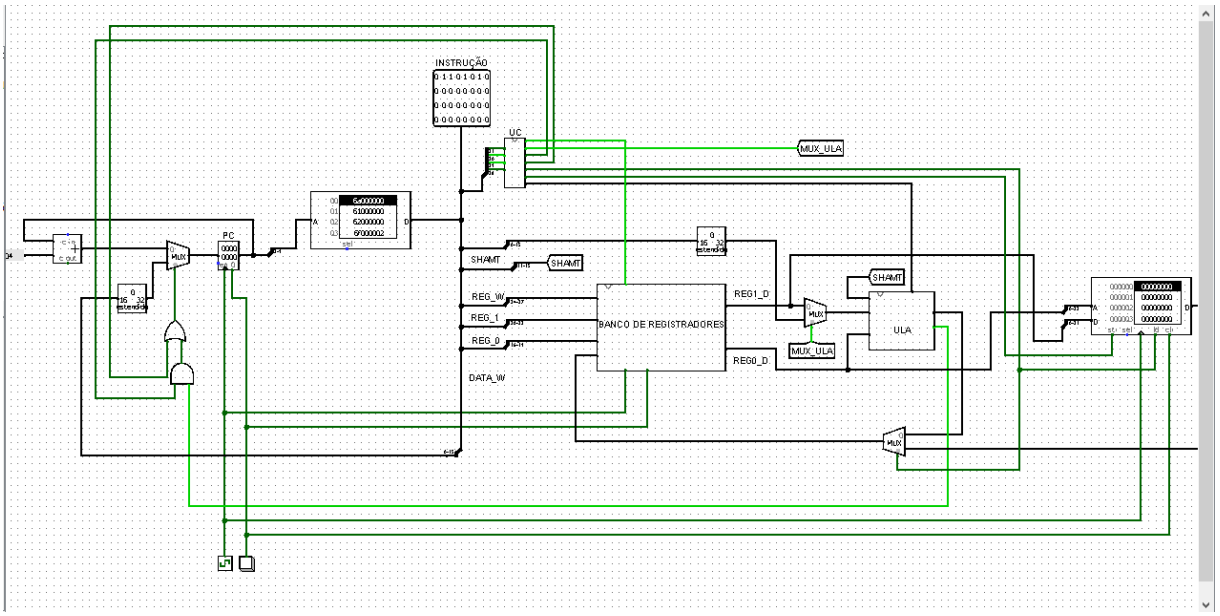

Figura 3: imagem do circuito geral do processador monociclo desenvolvido no logisim

Na imagem, pode-se perceber os seguintes componentes principais: uma memória de instruções, uma memória de dados, uma unidade de controle (UC), uma unidade lógico-aritmética (ULA), um banco de registradores e um registrador para armazenar o endereço da próxima instrução.

As matrizes que precisam ser efetivamente multiplicadas e a matriz resultante ficam salvas na memória de dados, separadas das instruções, que ficam na memória de instruções. Essa separação

entre as duas memórias é conveniente, por organizar melhor o projeto, o que facilita a manipulação dos endereços de memória.

Para que as duas matrizes fossem multiplicadas, foi necessário colocar todo o código de multiplicação de matrizes dentro da memória de instrução. Tal programa está descrito abaixo:

```
v2.0 raw
6a000000
61000000
62000000
6f000002
6b000000
6c000004
6d000008
50af008c
61000000
501f0084
62000000
63000000
29af0000
09910000
0ed90000
700e0000
502f0074
28af0000
08820000
0eb80000
840e0000
272f0000
07710000
0ec70000
850e0000
26450000
03360000
32020001
40000040
0ed90000
703e0000
31010001
```

40000024

3a0a0001

4000001c

A unidade de controle é responsável por verificar os 4 primeiros bits (que equivalem ao primeiro caractere) e informar à ULA qual operação realizar. Mais especificamente, a ULA realiza todas as operações que consegue, mas com base na informação recebida da UC. Ou seja, com base no op_code, a ULA seleciona qual resultado de qual operação será a saída. As imagens abaixo ilustram como é a UC e ULA por dentro:

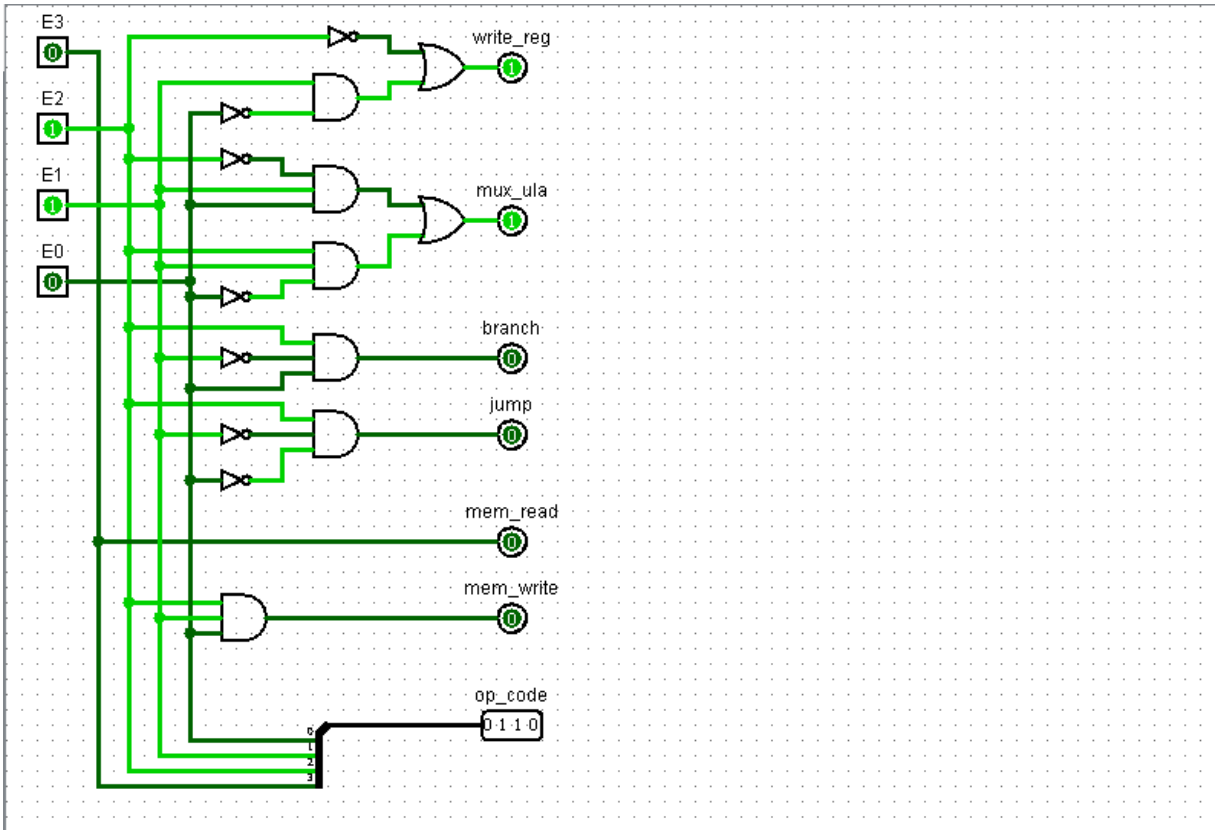

Figura 4: imagem do circuito da unidade de controle do processador monociclo no logisim

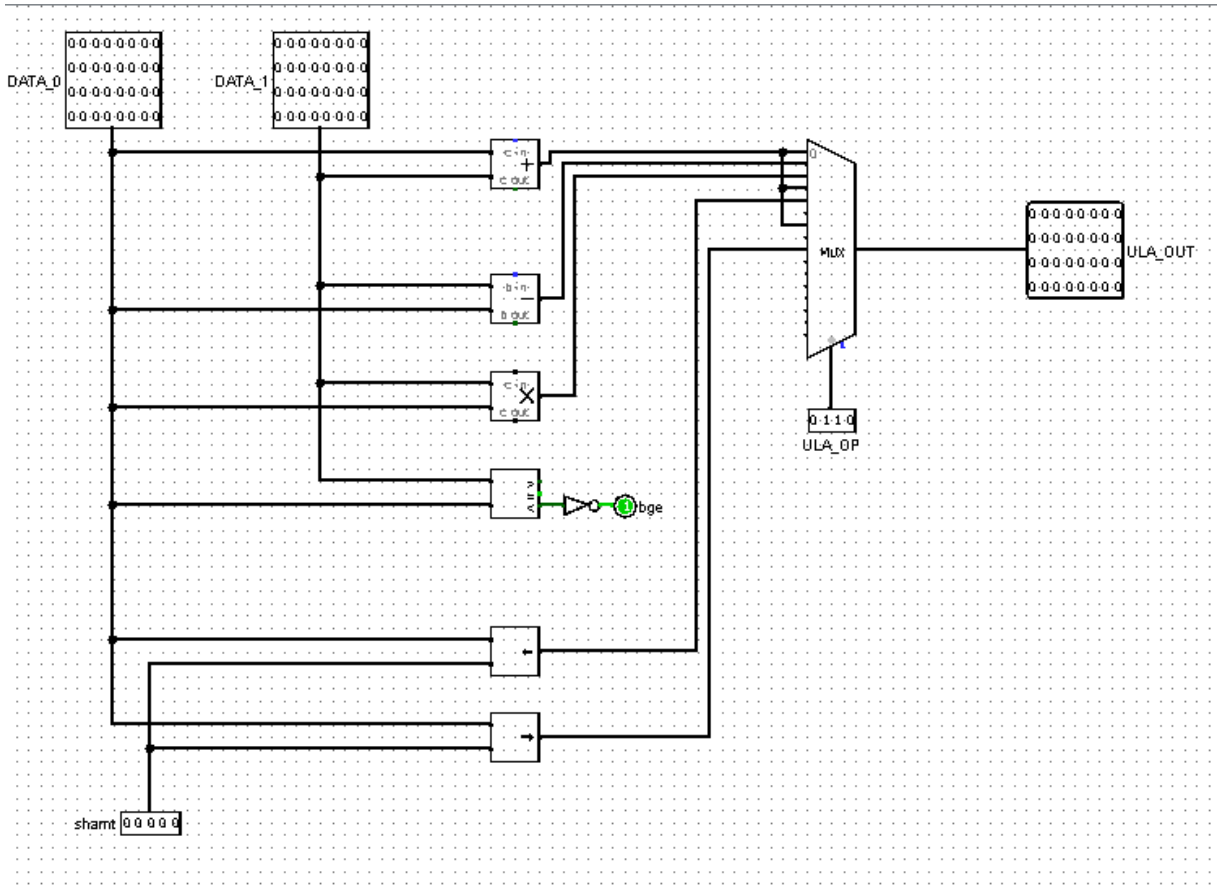

Figura 5: imagem do circuito da unidade lógico-aritmética do processador monociclo no logisim

O resultado que sai da ULA é armazenado no registrador especificado na instrução, o qual se encontra dentro do banco de registradores. Em suma, no banco de registradores, que possui 16 registradores, ficam armazenados os dados dos cálculos de cada instrução, além de endereços de memória que precisam ser acessados para atualizar as matrizes. As imagens dos demais componentes estão abaixam a fim de ilustrar como cada parte do processador foi implementada:

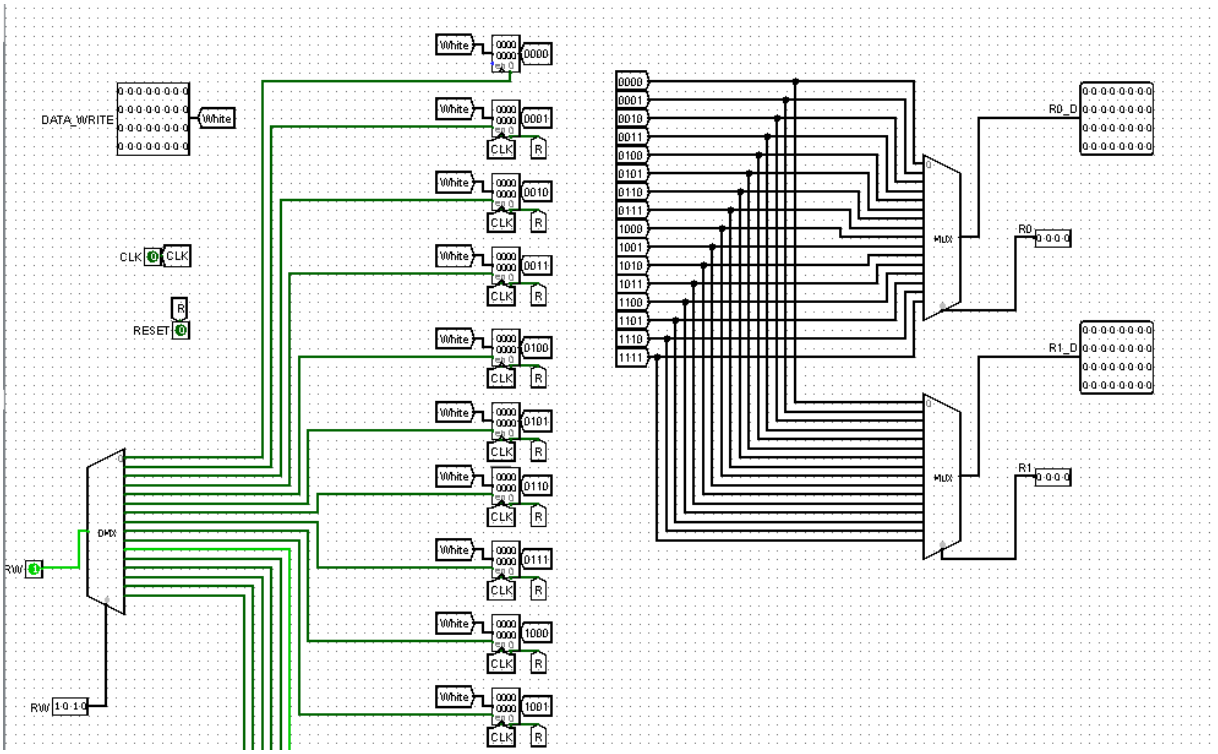

Figura 6: imagem do circuito do banco de registradores do processador monociclo no logisim

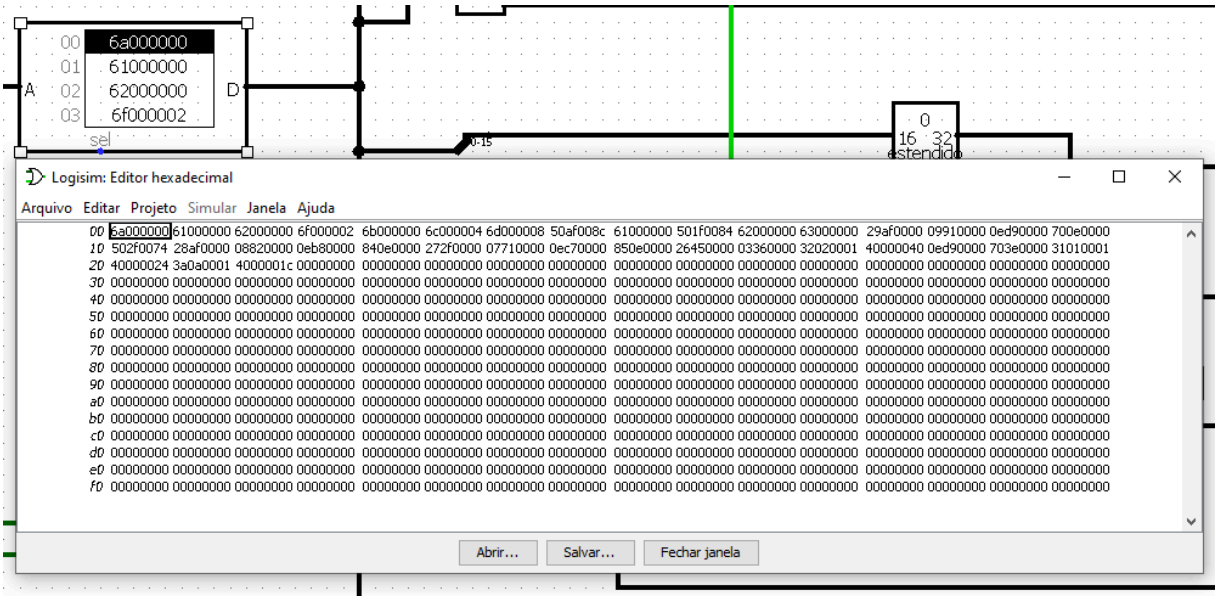

Figura 7: imagem da memória de instruções do processador monociclo no logisim

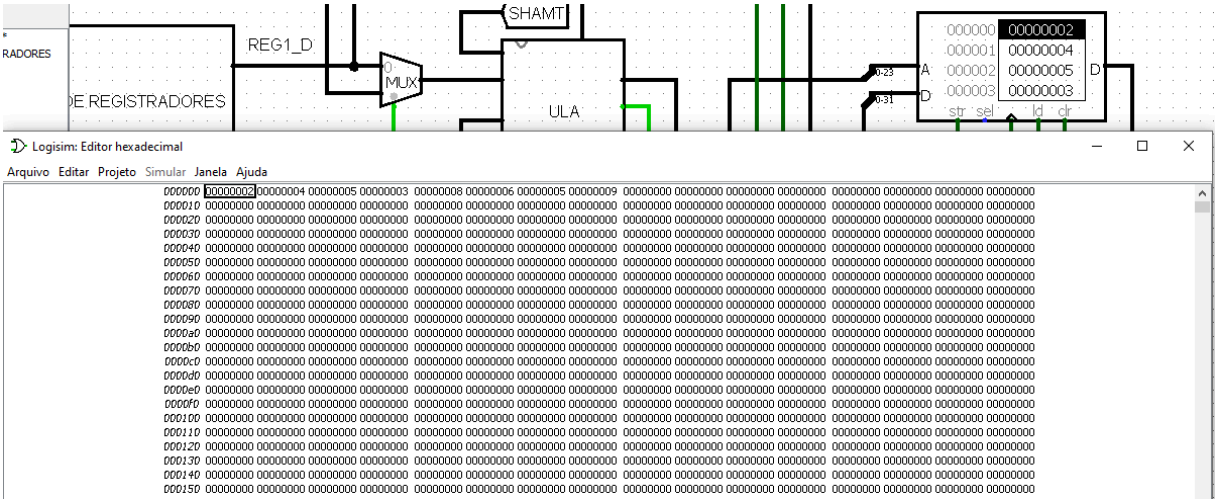

Figura 8: imagem da memória de dados do processador monociclo no logisim

4. PARTE 2: PROCESSADOR PIPELINE

O processador que opera com pipeline possui a vantagem de conseguir executar mais de uma instrução por ciclo. Isso se faz possível iniciando a execução de uma instrução antes do final da outra, o que ocorre por conta do paralelismo inerente ao funcionamento das instruções. Para tanto, são utilizados registradores para fragmentar o processo de execução de uma única instrução e, dessa forma, permitir que instruções em diferentes estágios estejam sendo executadas pelo processador ao mesmo tempo. Isso otimiza o tempo de execução de um conjunto de instruções que seja grande o suficiente, como frequentemente ocorre no cotidiano. Em outras palavras, em boa parte dos casos, o modelo pipeline é mais rápido que o modelo monociclo.

Com a intenção de adaptar o processador monociclo apresentado na Parte 1, foram inseridos novos registradores no circuito combinacional desenvolvido no logisim.

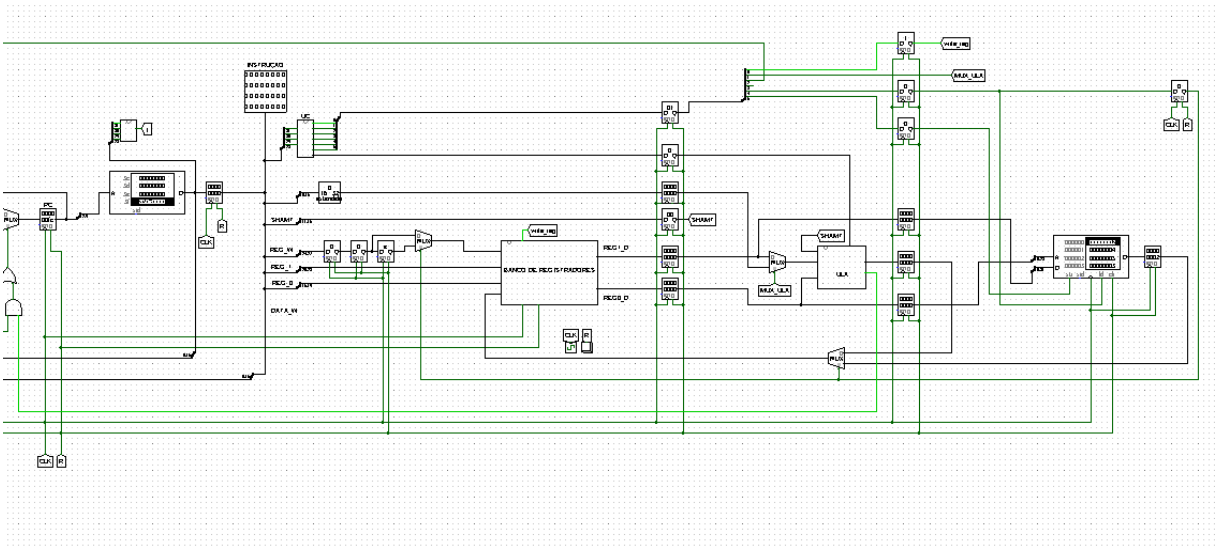

Figura 9: imagem do circuito geral do processador pipeline desenvolvido no logisim

Nem todos os componentes podem ser vistos com clareza na imagem, devido ao formato e tamanho do circuito, mas é conveniente deixar muito claro que a base do processador é a mesma do monociclo, com a grande diferença de que foram inseridos os registradores para implementar a fragmentação das instruções previamente mencionada. A imagem abaixo exibe alguns desses registradores que ficam entre a ULA e a memória de dados.

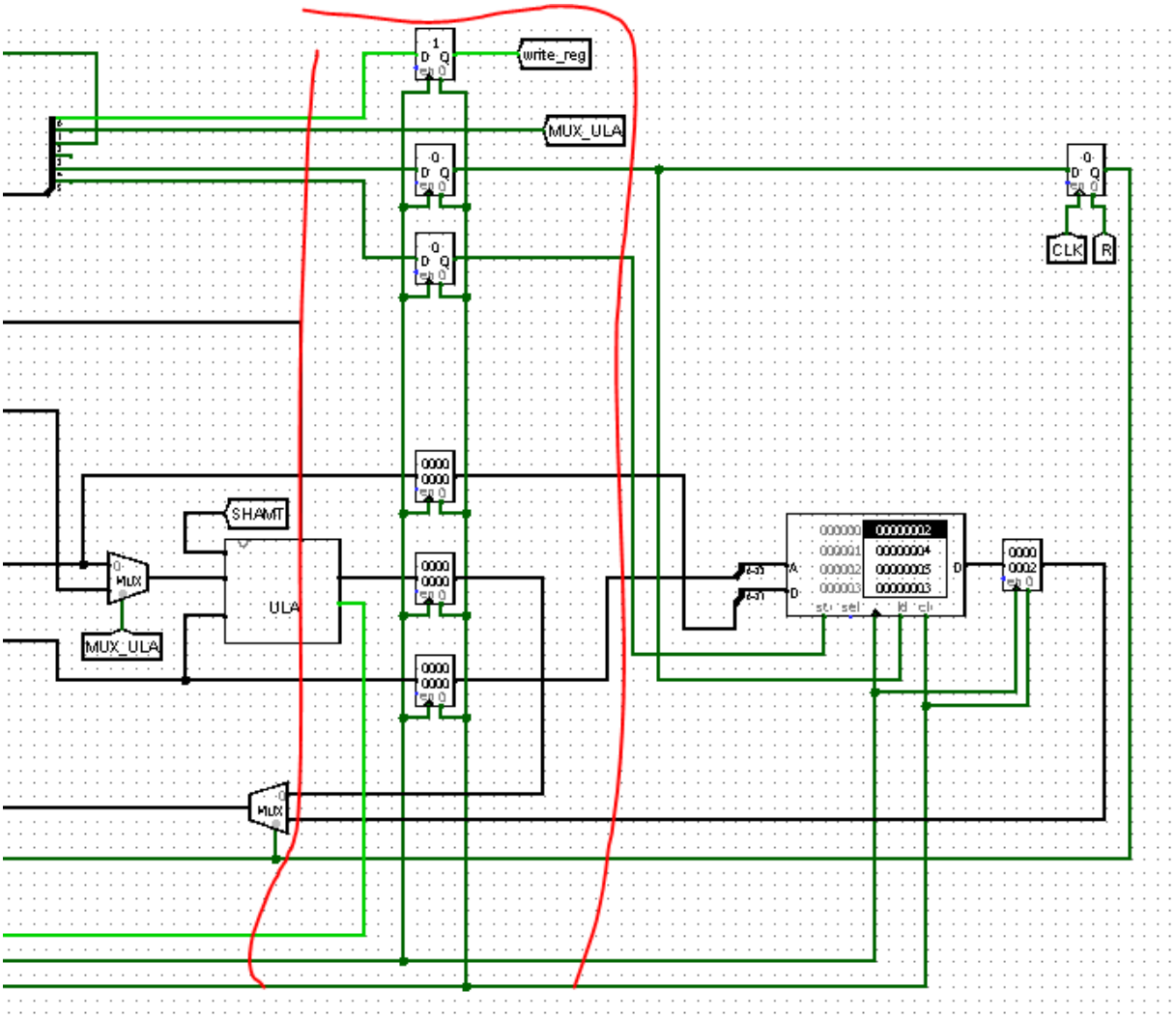

Figura 10: imagem de registradores usados para implementar o pipeline

A grande diferença entre as duas versões do processador é que a versão pipeline usa mais registradores para implementar a fragmentação das instruções em etapas. Cada componente do circuito, como a ULA e a UC, foi implementado da mesma maneira no logisim tanto para a versão monociclo quanto para a versão pipeline.

No que diz respeito a execução das instruções na versão pipeline, convém afirmar que foram inseridas

bolhas entre algumas instruções, ou seja, linhas na memória de instruções que estão preenchidas com “00000000”, para evitar conflito entre certas instruções.

Para saber mais detalhes do projeto, ler com mais detalhes os códigos e ver mais claramente o circuito implementado no logisim, acesse o seguinte link para ser direcionado para o repositório do projeto disponível no github: https://github.com/MikaellyL/MyProcessor.