Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso
Campus Cuiabá Octayde
Graduação em Engenharia da Computação

TRABALHO DE MICROPROCESSADOR
CRIANDO MIPS USANDO LOGISIM

R E L A T Ó R I O

Aluno:
Claudio Noberto – claudio.noberto@estudante.ifmt.edu.br

Professor:
Ruy de Oliveira

Cuiabá, MT
Fevereiro – 2025

Relatório do Trabalho Sobre MIPS Usando Logisim
Engenharia da Computação

Aluno:
Claudio Noberto – claudio.noberto@estudante.ifmt.edu.br

Disciplina: Microprocessadores – Normal.3131
Professor: *Ruy de Oliveira*

Resumo: Este relatório apresenta a implementação completa do microprocessador MIPS de 16 bits, incluindo o controlador principal e o ALUController, que gerenciam o fluxo de dados e a execução das instruções. O trabalho teve como objetivo compreender e analisar a interação entre o banco de registradores, a memória RAM e a Unidade Lógica e Aritmética (ULA), explorando conceitos fundamentais da arquitetura MIPS, como operações de leitura e escrita em memória, decodificação de instruções, controle de barramentos e geração de sinais de controle. São descritos o funcionamento das instruções implementadas, a estrutura do banco de registradores, da memória e da ULA, bem como os testes realizados para validar a comunicação entre os módulos e o correto funcionamento do controlador e do ALUController no ambiente virtual. A implementação final demonstra a capacidade do microprocessador em executar instruções de forma eficiente, garantindo a integridade dos dados e o controle preciso das operações.
Palavras-chave: MIPS, Arquitetura de computadores, Microprocessador, Logisim.

Cuiabá, MT
Fevereiro – 2025

SUMÁRIO

Criando o MIPS

1. INTRODUÇÃO

A arquitetura MIPS (Microprocessor without Interlocked Pipeline Stages) destaca-se como uma das bases mais influentes no estudo e desenvolvimento de processadores modernos. Originada na década de 1980, como parte do movimento RISC (Reduced Instruction Set Computer), a MIPS foi projetada para priorizar simplicidade e eficiência, reduzindo a complexidade do hardware e otimizando o desempenho por meio de um conjunto enxuto de instruções. Apesar das inovações que adicionaram recursos como pipeline avançado e execução fora de ordem, a essência minimalista do MIPS permanece amplamente utilizada como referência educacional. Sua clareza de design torna-se especialmente útil para entender os fundamentos da arquitetura de computadores, incluindo a interação entre registradores, memória e unidades de controle. Este relatório concentra-se na implementação completa da arquitetura MIPS de 16 bits, com ênfase na integração do controlador principal, do ALUController e dos demais módulos que compõem o microprocessador. A análise detalha o funcionamento desses componentes, desde a decodificação de instruções até a execução de operações aritméticas e lógicas, passando pelo controle de barramentos e pela comunicação eficiente entre o banco de registradores e a memória RAM. O trabalho demonstra como a modularidade e a simplicidade do design MIPS permitem a execução precisa e eficiente de instruções, reforçando os princípios que sustentam sua relevância no ensino e no desenvolvimento de sistemas computacionais.

2. FERRAMENTAS E METODOLOGIA

Para a implementação parcial do MIPS, foi utilizado o Logisim, um software de código aberto amplamente reconhecido por sua eficácia na simulação de circuitos digitais. A escolha pelo Logisim baseou-se em sua interface gráfica intuitiva, que facilita a criação e a visualização de sistemas digitais, tornando o processo acessível e didático.

3. FUNDAMENTAÇÃO TEÓRICA

O desenvolvimento do microprocessador MIPS exige a integração de diversos componentes lógicos e eletrônicos, que operam de forma sincronizada para executar instruções, manipular dados e gerenciar o fluxo de informações no sistema. A seguir, são

apresentados os conceitos principais e os componentes essenciais que foram aplicados na construção dos circuitos que conectam o banco de registradores à memória RAM.

3.1 MIPS

O MIPS é uma arquitetura de processador amplamente utilizada para introduzir os conceitos fundamentais da arquitetura de computadores. Ele foi projetado para executar instruções básicas, como leitura e escrita de dados, além de operações aritméticas e lógicas simples. Apesar de sua simplicidade em configurações educacionais, o MIPS incorpora elementos essenciais de um processador, incluindo banco de registradores, memória, barramentos e unidade de controle, oferecendo uma visão prática e clara sobre o funcionamento interno de sistemas digitais.

3.2 Somador de 6 Bits

O somador de 6 bits é um circuito usado para realizar a adição de dois números binários de até 6 bits. Ele trabalha propagando os valores de "carregamento" (carry) entre os bits, garantindo que o resultado da soma esteja correto. Quando conectado ao contador de programa (PC) em sistemas baseados na arquitetura MIPS, o somador é responsável por incrementar o valor do PC, geralmente adicionando 1 para avançar para a próxima instrução na memória. Em instruções de desvio, o somador pode ser usado para calcular o novo endereço baseado em deslocamentos. Esse componente é essencial para o funcionamento do processador, permitindo que o fluxo de instruções seja controlado de forma eficiente e precisa.

3.3 Contador de Programa

O contador de programa (PC - Program Counter) é um registrador essencial na arquitetura MIPS, responsável por armazenar o endereço da próxima instrução a ser executada. Ele atua como um controlador do fluxo de execução, garantindo que as instruções sejam lidas sequencialmente na maioria dos casos, ou desviadas para um endereço específico em situações de saltos ou chamadas de sub-rotinas. Em cada ciclo de clock, o PC é atualizado para apontar para a próxima instrução. Isso pode ser feito incrementando seu valor por uma constante (geralmente 4, no caso de endereçamento por bytes, ou 1, no caso de endereçamento por palavras) ou, em instruções de desvio e salto, carregando um novo valor calculado com base em deslocamentos ou endereços imediatos. No MIPS, o PC interage diretamente com outros componentes, como a memória de instruções, que fornece a instrução localizada no endereço indicado pelo PC, e o somador, que calcula o próximo valor do contador. Essa interação é crucial para o funcionamento de instruções como branch (desvio condicional) e jump (salto incondicional), que alteram o fluxo de execução programado.

3.4 Memória de Instruções (ROM)

A memória de instruções no MIPS de 16 bits é responsável por armazenar o conjunto de instruções que o processador deve executar. Cada instrução é codificada em formato binário e ocupam 16 bits (2 bytes), alinhados na memória. Essa organização simplifica o acesso e a leitura das instruções pelo processador. Durante a execução, o contador de programa (PC) fornece o endereço da próxima instrução à memória de

instruções. A memória então retorna o código binário correspondente, que será decodificado e processado pela Unidade de Controle (UC). Esse ciclo, conhecido como busca de instrução, garante o funcionamento contínuo do sistema. No MIPS de 16 bits, a memória de instruções é tipicamente separada da memória de dados em arquiteturas Harvard, permitindo acessos paralelos, ou compartilhada em arquiteturas Von Neumann, dependendo do design adotado. A memória de instruções é um componente essencial para o fluxo de execução, pois armazena e fornece as instruções necessárias de forma sequencial ou conforme indicado por desvios e saltos controlados pelo contador de programa.

3.5 Banco de Registradores (8x16)

No processador MIPS, o Banco de Registradores é uma unidade fundamental que armazena dados temporários utilizados durante a execução de instruções. Ele consiste em um conjunto de registradores de acesso rápido, diretamente integrados ao núcleo do processador, permitindo maior eficiência em comparação com a memória principal. Quando nos referimos a um "Banco de Registradores (8x16)", estamos tratando de um modelo composto por 8 registradores, cada um com capacidade para armazenar 16 bits de dados. Essa configuração é menor que a encontrada em implementações completas do MIPS, mas pode ser utilizada em exemplos simplificados ou modelos educacionais. Os registradores desempenham funções essenciais, como armazenar operandos para operações aritméticas e lógicas, além de endereços ou resultados intermediários. O processador pode realizar operações simultâneas de leitura e escrita em dois registradores, dependendo da instrução. Por exemplo, em uma operação de soma, os valores podem ser carregados de dois registradores do banco, processados pela Unidade Lógica e Aritmética (ULA), e o resultado armazenado em outro registrador do mesmo banco. Essa estrutura é crucial para garantir o desempenho e a eficiência do processador durante a execução de programas.

3.6 Unidade Lógica e Aritmética (ULA)

A Unidade Lógica e Aritmética (ULA) é um componente essencial do processador MIPS, responsável por executar operações matemáticas e lógicas necessárias para o processamento de instruções. Ela interage diretamente com o Banco de Registradores e o caminho de dados, desempenhando um papel central na execução eficiente de programas. Entre suas principais funções estão as operações aritméticas, como soma, subtração, multiplicação e divisão, além das operações lógicas, como AND, OR, XOR e deslocamentos de bits (shift), essenciais para manipulação de dados binários. Além disso, a ULA é fundamental para a tomada de decisões, permitindo comparar valores e avaliar condições, como igualdade ou desigualdade, o que é crucial para a execução de instruções condicionais, como saltos em programas. Seu funcionamento é controlado pelo controlador do processador, que determina qual operação a ULA deve realizar com base na instrução em execução. Por exemplo, em uma soma, os operandos são enviados pelo Banco de Registradores, a ULA realiza o cálculo, e o resultado é armazenado novamente no registrador ou usado em outra etapa. Projetada para oferecer alta velocidade e precisão, a ULA é otimizada para garantir o desempenho eficiente do processador, sendo indispensável para a funcionalidade básica de qualquer arquitetura de CPU, incluindo o MIPS.

3.7 Controlador da ULA

O Controlador da ULA (ALUController) é um componente fundamental no microprocessador MIPS, responsável por gerenciar e coordenar as operações realizadas

pela Unidade Lógica e Aritmética (ULA). Sua principal função é interpretar os sinais de controle enviados pela unidade de controle principal e convertê-los em comandos específicos para a ULA, assegurando a execução correta de operações aritméticas (como adição, subtração, multiplicação e divisão) e lógicas (como AND, OR, XOR e deslocamentos). O ALUController recebe como entrada os sinais de controle derivados do campo funct das instruções do tipo R (register) ou do opcode das instruções do tipo I (immediate). Com base nesses sinais, ele gera comandos precisos, como o sinal ALUOp, que define a operação a ser realizada pela ULA. Além disso, o ALUController é responsável por mapear operações de alto nível (por exemplo, "realizar uma soma") em sinais de baixo nível que a ULA possa interpretar, utilizando lógica combinacional para traduzir os sinais de controle em ações específicas. Ele também suporta diferentes tipos de instruções, incluindo operações aritméticas e lógicas (tipo R), operações com valores imediatos (tipo I) e instruções de deslocamento. Integrado à ULA, o ALUController garante que os operandos fornecidos pelo banco de registradores ou pela memória sejam processados de maneira eficiente e precisa, desempenhando um papel crucial na execução das instruções e no fluxo de dados do microprocessador

3.8 Memória de Dados (RAM)

A Memória de Dados (RAM) no processador MIPS é uma unidade essencial para o armazenamento temporário de informações necessárias durante a execução de programas. Essa memória é do tipo volátil, ou seja, seus dados são perdidos quando o sistema é desligado, e é usada para guardar variáveis, dados intermediários e resultados que precisam ser acessados rapidamente pelo processador. No contexto do MIPS, a RAM é organizada em endereços, permitindo que o processador leia ou escreva dados específicos conforme necessário. Por exemplo, instruções de carga (load) e armazenamento (store) utilizam a memória para transferir informações entre a RAM e o Banco de Registradores. A RAM desempenha um papel crucial na execução de programas, permitindo o armazenamento temporário de instruções ou dados que não cabem nos registradores internos. Sua capacidade e velocidade têm impacto direto no desempenho do sistema, pois uma memória rápida reduz os atrasos no acesso aos dados, enquanto uma maior capacidade suporta programas mais complexos e que requerem mais espaço de armazenamento temporário. Assim, a Memória de Dados é indispensável para garantir o funcionamento eficiente do processador MIPS e de sistemas computacionais em geral.

3.9 Extensor de Bits

O Extensor de Bits é um componente do processador MIPS que ajusta o tamanho de dados binários para compatibilidade com o tamanho padrão dos registradores, garantindo a execução correta de operações. Ele é usado principalmente para expandir valores menores, como números de 6 bits, para 16 bits, quando são utilizados em instruções imediatas ou ao carregar dados da memória. O processo pode ser realizado de duas formas: na extensão com sinal (sign-extension), o bit mais significativo do número original é replicado nos bits adicionais para preservar o sinal (positivo ou negativo); já na extensão sem sinal (zero-extension), os bits extras são preenchidos com zeros. Essa funcionalidade é essencial para evitar erros ao operar com diferentes tamanhos de dados e assegurar que cálculos e transferências sejam feitos corretamente dentro do processador.

3.10 Controlador/Sinais de Controle

Os Sinais de Controle são essenciais no processador MIPS, pois coordenam o funcionamento de seus diversos componentes, garantindo que as instruções sejam executadas corretamente. No projeto inicial, esses sinais eram gerados manualmente, mas, na implementação final, eles passaram a ser gerenciados pelo Controlador Geral, uma unidade centralizada que automatiza e sincroniza o fluxo de execução das instruções. O Controlador Geral é responsável por interpretar o opcode e os campos adicionais da instrução, como o funct e os registradores envolvidos, e gerar os sinais de controle necessários para operar a ULA, o Banco de Registradores, a Memória, o Extensor de Bits e outros módulos do processador. Esses sinais determinam ações específicas, como a seleção de registradores para leitura ou escrita, o tipo de operação que a ULA deve executar, e se os dados devem ser carregados da memória ou armazenados nela. Por exemplo, durante uma instrução de soma, o Controlador configura a ULA para realizar a operação aritmética, define quais registradores fornecerão os operandos e onde o resultado será armazenado. Além disso, o Controlador Geral gerencia operações condicionais, como saltos (branch), e configura multiplexadores para selecionar os caminhos de dados corretos. Assim, ele funciona como o "sistema nervoso" do processador, orquestrando todas as operações de maneira sincronizada e garantindo que as instruções sejam executadas com precisão.

1. **RegDst**: Define qual registro destino será utilizado (por exemplo, o registrador rt ou o registrador rd).
2. **CLK**: O sinal de relógio controla quando as operações ocorrem no processador, sincronizando as ações.
3. **RegWrite**: Permite ou impede a escrita de novos valores nos registradores.
4. **CLR**: Serve para limpar determinados componentes do processador, como registradores e a memória.
5. **PC_Enable**: Habilita ou desabilita o carregamento do contador de programa, controlando a sequência de execução das instruções.
6. **ALUsrc**: Define se a entrada da unidade aritmético-lógica (ALU) será um operando imediato ou um registrador.
7. **MemWrite**: Controla se as operações de escrita na memória estão ativas.
8. **MemRead**: Controla se as operações de leitura na memória estão ativas.
9. **MemToReg**: Define de onde o resultado da memória deve ser encaminhado (memória ou registrador).
10. **Jump**: Controla a execução de instruções de salto incondicional (jump), direcionando o contador de programa (PC) para um endereço específico.
11. **Branch**: Gerencia instruções de desvio condicional (branch), ativando o sinal de Branch quando uma condição é atendida (por exemplo, se dois registradores são iguais) e redirecionando o fluxo de execução.
12. **Halt**: Interrompe a execução do processador, finalizando o ciclo de instruções e parando o funcionamento do sistema.

Com a implementação do Controlador Geral, o processador MIPS ganhou maior autonomia e eficiência, eliminando a necessidade de controle manual dos sinais e permitindo a execução automatizada e precisa das instruções. O Controlador Geral, em conjunto com o ALUController, forma o núcleo de gerenciamento do processador, garantindo que todas as operações sejam realizadas de acordo com a arquitetura MIPS e seus princípios de simplicidade e modularidade.

3.11 Conjunto de Instruções

O conjunto de instruções do processador MIPS de 16 bits é baseado em uma arquitetura RISC (Reduced Instruction Set Computer), que prioriza simplicidade e

eficiência. As instruções possuem tamanho fixo de 16 bits, o que facilita a decodificação e execução no hardware. Esse conjunto é composto por três formatos principais: R, I e J, cada um projetado para atender a diferentes tipos de operações.

- **Instruções do Tipo R**: São usadas para operações aritméticas e lógicas entre registradores, como soma, subtração e operações lógicas (AND, OR, etc.). O formato inclui campos para especificar até dois registradores de entrada, um registrador de destino e um campo para o código da função (funct), além do opcode que identifica a instrução. Exemplos: add, sub, and, or.
- **Instruções do Tipo I**: São destinadas a operações que envolvem um registrador e um valor imediato, ou acesso à memória. O formato possui campos para dois registradores (um de base e um de destino), um valor imediato de 5 bits e o opcode. Exemplos incluem: lw (load word), sw (store word), addi (add immediate) e instruções de comparação.
- **Instruções do Tipo J**: São utilizadas para saltos de longo alcance, permitindo alterar o fluxo do programa para uma posição específica na memória. O formato contém um opcode e um endereço de 11 bits, possibilitando saltos dentro do espaço de endereçamento de 16 bits. Exemplos: j (jump) e jal (jump and link).

No MIPS de 16 bits, cada instrução é representada em linguagem assembly, que utiliza uma notação específica para definir as operações e seus operandos. Essa linguagem se baseia em mnemônicos (como add, sub, lw, sw) seguidos dos registradores ou valores constantes utilizados. Na arquitetura MIPS de 16 bits, a notação para registradores utiliza o símbolo $ e um número que identifica o registrador. Neste caso, há 8 registradores disponíveis, que vão de $0 a $7. Os registradores são usados para armazenar temporariamente dados e endereços durante a execução das instruções. Quando se trata de valores imediatos ou constantes, estes são escritos diretamente, sem nenhum símbolo adicional antes ou depois. Já para instruções que acessam a memória, como lw (load word) e sw (store word), a sintaxe é diferente. Nessas instruções, o endereço de memória a ser acessado é especificado como a soma de uma constante e o valor armazenado em um registrador, utilizando parênteses para delimitar o registrador. Por exemplo, na instrução add $1, $3, $2, que é do tipo R, a operação realizada será $2 = $1 + $3, ou seja, o registrador $2 receberá a soma dos conteúdos de $1 e $3. Já na instrução lw $7, 1($3), que é do tipo I, o registrador $7 carregará o valor armazenado na posição de memória correspondente à soma de 1 com o conteúdo de $3. Assim, se $3 contiver o valor 7, $7 receberá o dado da posição de memória 8 (7 + 1).

Ademais, para que o computador possa executar as instruções escritas em assembly, elas precisam ser convertidas para o formato binário, conhecido como código de máquina. Esse processo traduz as instruções para um padrão que o hardware do processador MIPS pode entender e executar. No MIPS de 16 bits, cada instrução ocupa exatamente 16 bits e é organizada em palavras, cuja estrutura varia dependendo do formato da instrução: R, I ou J. No formato R, usado para operações aritméticas e lógicas entre registradores, a palavra de 16 bits é dividida em 4 bits para o opcode, que identifica a operação, 3 bits para o registrador fonte (rs), 3 bits para o registrador alvo (rt), 3 bits para o registrador de destino (rd) e 3 bits para o campo funct, que detalha a função específica a ser executada. Por exemplo, a instrução add $0, $1, $2 seria codificada atribuindo os valores correspondentes aos campos acima. No formato I, usado para operações com valores imediatos ou acesso à memória, a divisão da palavra é feita em 4 bits para o opcode, 3 bits para o registrador fonte (rs), 3 bits para o registrador destino (rt) e 6 bits para o valor imediato. Por exemplo, a instrução addi $1, $2, 15 será codificada com os valores do opcode, registradores e o imediato no formato correspondente. No formato J, usado para saltos no fluxo de execução, a organização é composta por 4 bits para o

opcode e 12 bits para o endereço de destino. Por exemplo, a instrução j 50 será representada com o opcode e o endereço de destino apropriados.

A tabela abaixo apresenta todas as instruções junto com os seus formatos correspondentes.

Instrução	Tipo	Opcode	rs	rt	rd	funct	HEX	Assembly
add	R	1111	001	010	011	000	F29C	add $3,$1,$2
sub	R	1111	001	010	011	001	F29D	sub $3,$1,$2
and	R	1111	010	011	100	010	F4E2	and $4,$2,$3
or	R	1111	010	011	100	011	F4E3	or $4,$2,$3
xor	R	1111	010	011	100	100	F4E4	xor $4,$2,$3
nand	R	1111	011	100	110	101	F765	nand $6,$3,$4
nor	R	1111	011	100	110	110	F766	nor $7,$3,$4
slt	R	1111	010	011	001	111	F4C7	slt $1,$2,$3
ori	I	1010	101	100	000	111	A580	ori $,$,$
lw	I	1011	010	011	000	101	B4C5	lw $3,5($2)
sw	I	1100	000	011	000	101	C0C5	sw $3,5($0)
beq	I	1101	001	010	000	110	D4A6	beq $2,$1, 6
addi	I	1000	010	001	001	001	8411	addi $2,$1,9
slti	I	1001	010	001	001	001	9411	slti $2,$1,9
andi	I	0110	101	100	000	111	6B07	andi $5,$4,7
j	J	0111	000	000	000	111	7007	j 7
Halt		0001	x	x	x	x	1xxx	halt

Figura 1. Tabela de Instruções do MIPS de 16 Bits.

O MIPS de 16 bits mantém a filosofia de simplicidade e eficiência da arquitetura RISC, mas adaptado para um espaço de endereçamento e largura de barramento menores. As instruções são projetadas para realizar operações específicas de forma direta, favorecendo alta performance e simplicidade no hardware. Além disso, o conjunto de instruções é ortogonal, garantindo facilidade na programação e compilação, mesmo em um ambiente com recursos limitados.

4. CIRCUITOS DESENVOLVIDOS

A seguir, é apresentada uma descrição detalhada de cada componente implementado no Logisim-evolution para a construção parcial do processador MIPS.

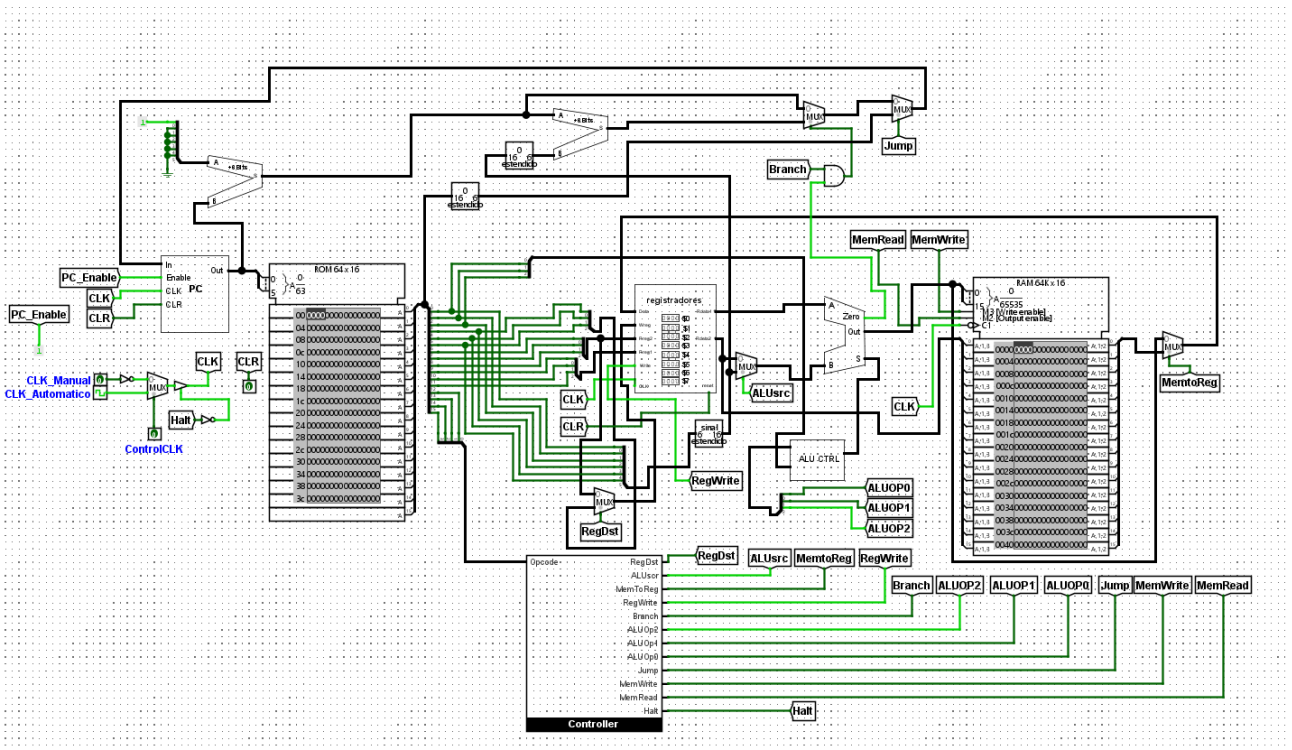

Figura 2. Esquema do MIPS.

4.1. Somador de 6 Bits

O Somador de 6 Bits desenvolvido no Logisim, conforme mostrado no circuito, é composto por seis módulos de somadores completos (full adders) conectados em cascata para realizar a adição de dois números binários de 6 bits, representados pelas entradas A e B. Cada somador completo processa um bit correspondente de A e B, considerando também o bit de transporte (carry-in) da operação do bit menos significativo para o mais significativo. No circuito, cada somador completo possui três entradas: A, B e Cin (carry-in), e gera duas saídas: S (soma) e Cout (carry-out). O primeiro somador, correspondente ao bit menos significativo, recebe o carry-in inicial como 0, enquanto os somadores subsequentes utilizam o carry-out do estágio anterior como carry-in. As saídas de soma de todos os somadores são conectadas em série para formar a saída final de 6 bits (Saída), que representa o resultado da adição. O funcionamento do circuito é simples e eficiente: cada estágio do somador calcula a soma parcial e propaga o carry para o próximo estágio. Esse método permite que o circuito adicione qualquer par de números binários de até 6 bits, gerando o resultado correto com suporte ao transporte entre os bits.

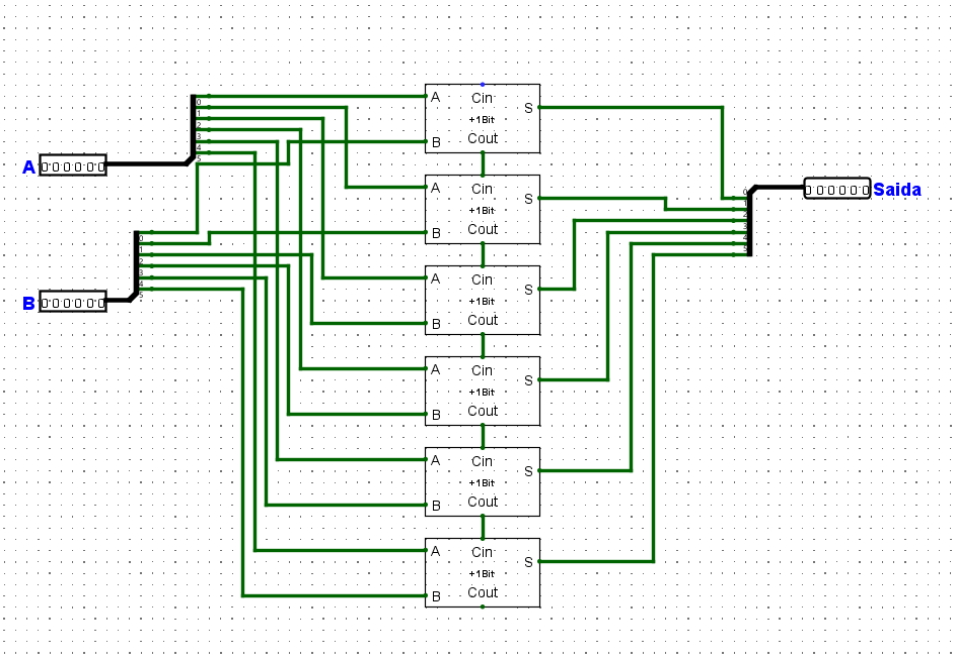

Figura 3. Circuito do Somador de 6 Bits.

4.2. Contador de Programa

O Contador de Programa (PC), representado no circuito, é responsável por controlar a sequência de execução das instruções no processador MIPS. Ele armazena o endereço da próxima instrução a ser buscada na memória e é projetado para incrementar automaticamente a cada ciclo de clock, permitindo o fluxo sequencial do programa. O circuito é composto por registradores tipo D flip-flop, que armazenam os bits do endereço atual. A entrada CLK (clock) controla a sincronização do incremento, enquanto a entrada CLR (clear) permite zerar o valor do contador, reiniciando sua operação. Adicionalmente, há uma entrada de controle chamada “PC_On”, que habilita ou desabilita o funcionamento do contador, e a entrada A, que possibilita carregar manualmente um endereço específico no contador, permitindo saltos ou alterações no fluxo do programa. O funcionamento do circuito se dá em dois modos principais: no modo sequencial, o contador incrementa automaticamente o valor armazenado a cada ciclo de clock, enquanto no modo de carregamento manual, o valor presente na entrada A é transferido para os flip-flops, alterando diretamente o endereço armazenado. A saída do contador é conectada a outros componentes do processador, como a unidade de controle e a memória, determinando qual instrução será executada em seguida. Esse circuito é fundamental para a execução ordenada de instruções no processador, garantindo o controle preciso do fluxo de execução, seja de forma sequencial ou com alterações controladas, como em instruções de salto.

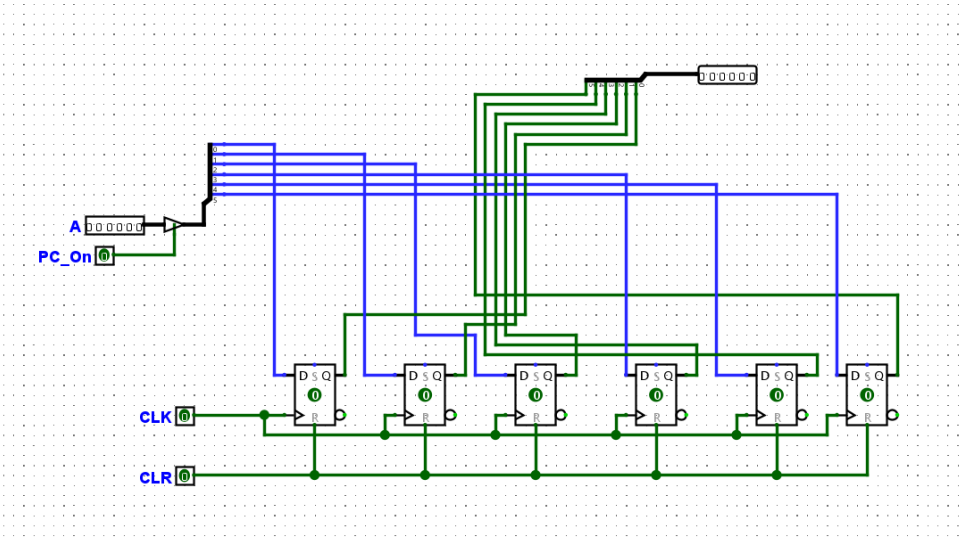

Figura 4. Circuito do Contador de Programa.

4.3. Memória de Instruções (ROM)

A Memória de Instruções, implementada no circuito utilizando uma memória ROM (Read-Only Memory), é responsável por armazenar o conjunto de instruções que serão executadas pelo processador MIPS. Essa memória é organizada em 64 palavras de 16 bits, conforme exibido no circuito, e cada endereço armazena uma instrução codificada em HEX. A entrada de 5 bits no barramento de endereços permite acessar qualquer uma das 64 posições de memória, enquanto a saída de 16 bits fornece a instrução correspondente ao endereço selecionado. A ROM, por ser uma memória de leitura, contém um conjunto de instruções previamente gravadas e não pode ser alterada durante a execução, o que a torna ideal para a simulação de programas fixos. No contexto do funcionamento do processador, o Contador de Programa (PC) fornece o endereço da próxima instrução, que é utilizado para acessar a ROM e carregar a instrução na saída. Essa instrução é então decodificada e executada pelos demais componentes do processador. A utilização da ROM no Logisim para simular a Memória de Instruções simplifica o desenvolvimento e a execução de programas no processador MIPS, além de permitir a visualização e manipulação das instruções diretamente. Esse componente é essencial para garantir o funcionamento correto do ciclo de busca e execução no processador.

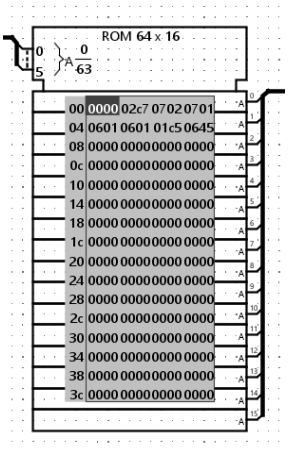

Figura 5. Imagem da ROM do Sistema.

4.4. Banco de Registradores (8x16)

O banco de registradores é um componente fundamental em arquiteturas como a do processador MIPS, sendo responsável por armazenar temporariamente dados utilizados nas operações de processamento. No circuito apresentado, temos um banco de registradores 8x16, que consiste em 8 registradores, cada um com 16 bits de largura. A entrada principal, indicada como "Entrada", é o barramento de dados que alimenta os registradores. A seleção de qual registrador será escrito é realizada por um demultiplexador (DMX), controlado pelo sinal "Wreg". Esse sinal, juntamente com o pulso de "Write", determina o momento e o registrador específico para realizar a escrita de dados. Adicionalmente, há um sinal de "Reset" que possibilita a reinicialização de todos os registradores para um estado inicial, tipicamente zerado. Cada registrador possui uma entrada de habilitação (WE - Write Enable), permitindo que apenas o registrador selecionado receba os dados da entrada. Durante o processo de leitura, dois multiplexadores (MUX) selecionam os registradores que terão seus valores encaminhados para as saídas "Rdata1" e "Rdata2", correspondendo às saídas "Sr1" e "Sr2", respectivamente. Os sinais de controle dos multiplexadores determinam quais registradores serão lidos e enviados às respectivas saídas, permitindo a utilização dos dados armazenados no banco em outras partes do processador. Este circuito exemplifica um banco de registradores típico utilizado no processador MIPS, oferecendo operações básicas de leitura e escrita com controle eficiente, enquanto garante a capacidade de armazenamento e acesso rápido aos dados necessários para a execução de instruções.

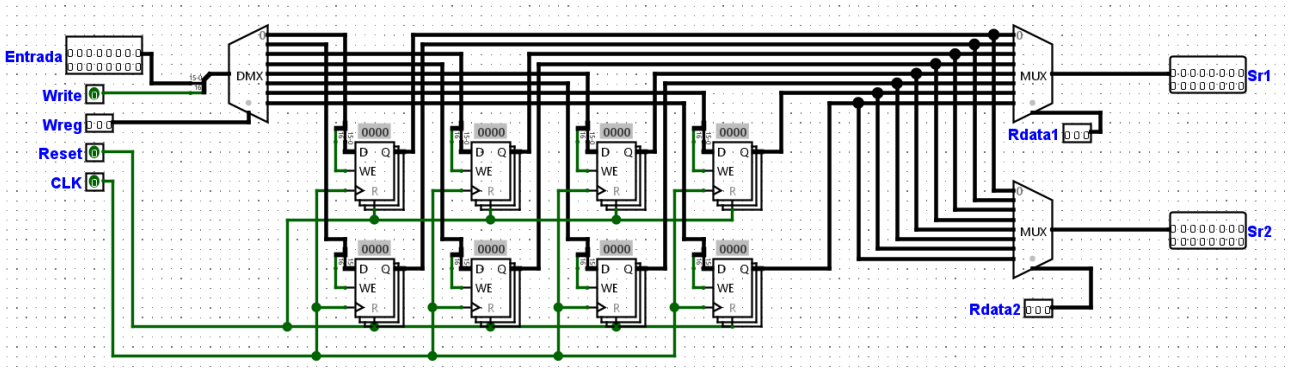

Figura 6. Circuito do Banco de Registradores.

4.5. Unidade Lógica e Aritmética (ULA)

A Unidade Lógica e Aritmética (ULA) é um componente essencial de processadores como o MIPS, sendo responsável pela execução de operações aritméticas e lógicas definidas pelas instruções do programa. O circuito apresentado representa uma ULA configurada para operar com dados de 16 bits, recebendo duas entradas principais, "A" e "B", e um sinal de controle denominado "Selector", que especifica a operação a ser realizada. O seletor, com 3 bits de largura, controla qual operação será executada, conforme descrito na tabela de mnemônicos apresentada no circuito. Entre as operações suportadas estão soma (SOMA), subtração (SUB), operações lógicas como AND, OR, NAND, NOR e XOR. A execução das operações é feita por uma combinação de portas lógicas e um somador/subtrator. O somador/subtrator é ativado para realizar cálculos aritméticos, dependendo do valor do seletor. Operações lógicas, por sua vez, são implementadas diretamente utilizando portas AND, OR, NAND, NOR e XOR, que processam os bits das entradas "A" e "B" em paralelo. Após o processamento, os

resultados das operações são encaminhados para um multiplexador (MUX). Este multiplexador seleciona qual resultado será enviado à saída "Saída", de acordo com o valor do seletor, garantindo que o resultado correto seja utilizado para os cálculos subsequentes ou armazenado no banco de registradores. Essa implementação da ULA é um exemplo claro de como operações aritméticas e lógicas são realizadas em um processador. O circuito destaca a interação entre os sinais de controle e o hardware subjacente para fornecer uma unidade funcional que atende às necessidades de execução de instruções no MIPS.

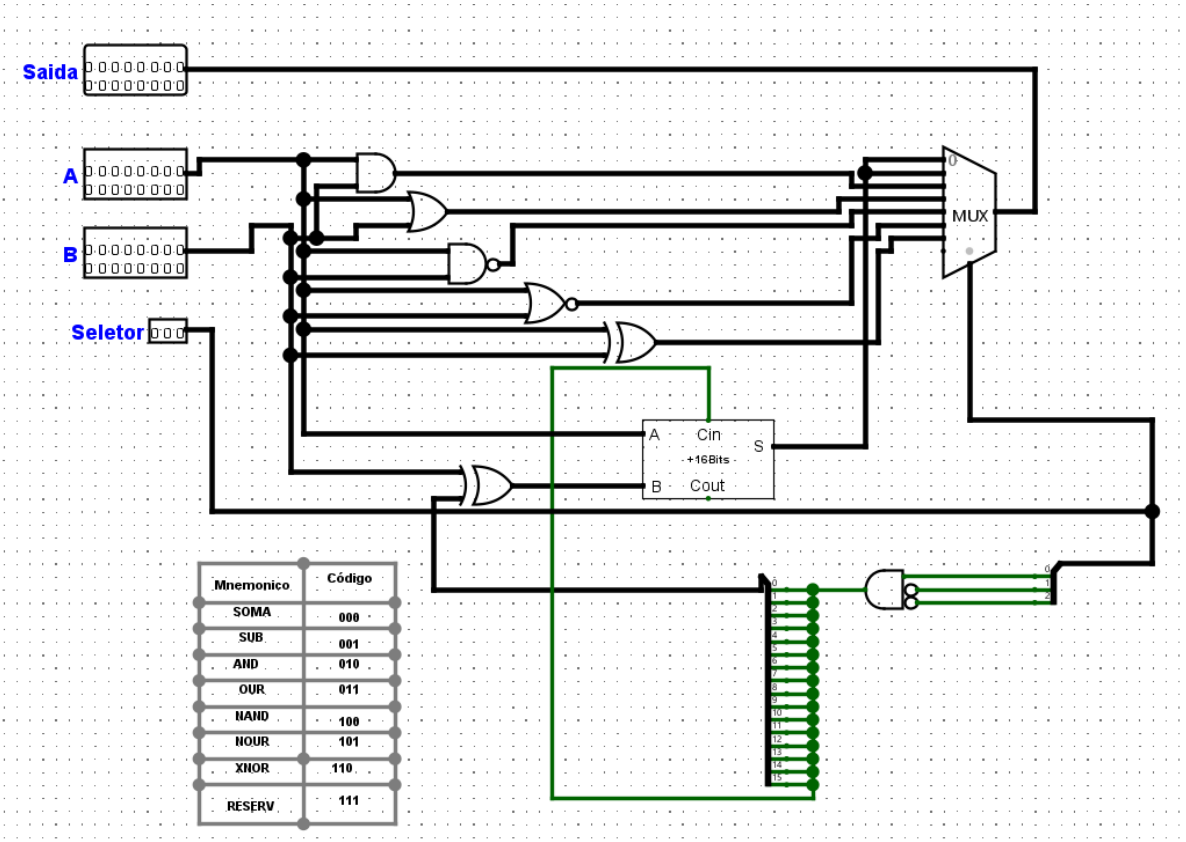

Figura 7. Circuito da ULA.

4.6. Memória de Dados (RAM)

A memória de dados (RAM) é utilizada para armazenar temporariamente informações necessárias durante a execução do processador, como dados intermediários ou valores de instruções. No Logisim, a RAM é configurável, permitindo definir o número de endereços e a largura de cada palavra. Ela possui como entradas o endereço, os dados a serem armazenados e sinais de controle, como o Write Enable (WE), que habilita a escrita, e o Clock (CLK), que sincroniza as operações. Para leitura, o endereço especifica a posição da memória, e o dado armazenado é retornado na saída. Para escrita, os dados de entrada são armazenados no endereço indicado quando o sinal de escrita está ativo. A RAM do Logisim facilita a simulação e integração com outros componentes, como a ULA e o banco de registradores, refletindo o funcionamento real de um processador.

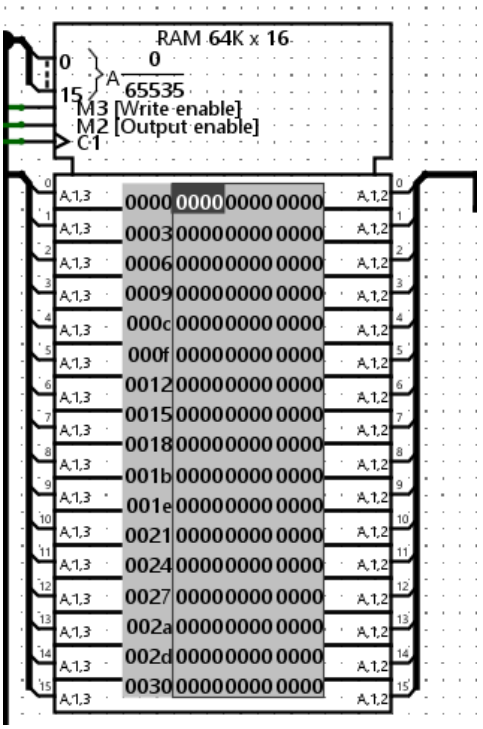

Figura 8. Imagem da RAM do Sistema.

4.7 Controlador da ULA

O Controlador da ULA (ALUController) é um componente essencial no microprocessador MIPS, responsável por interpretar os sinais de controle e definir as operações que a Unidade Lógica e Aritmética (ULA) deve executar. No circuito apresentado, o Controlador da ULA recebe entradas como o campo funct das instruções do tipo R e o opcode das instruções do tipo I, gerando os sinais de controle necessários para configurar a ULA. Esses sinais, representados pelo OpALU, determinam a operação específica a ser realizada, como operações aritméticas, lógicas ou manipulações de dados. Por exemplo, o valor 011 corresponde a uma operação do tipo R, enquanto 100 indica uma operação de adição imediata (ADDI). O circuito suporta uma variedade de operações, incluindo ADDI, SUBI, ORI, ANDI e instruções do tipo R, além de um estado "NADA" para operações que não requerem ação da ULA. O Controlador da ULA utiliza uma lógica combinacional para mapear as entradas (funct e opcode) nos sinais de controle apropriados. Essa lógica é implementada por meio de portas lógicas que traduzem os valores de entrada em sinais de saída (OpALU). O circuito também inclui um funct pool, que é utilizado para identificar operações específicas dentro das instruções do tipo R. O sinal OpALU gerado pelo Controlador é enviado diretamente para a ULA, que executa a operação correspondente com base nos operandos fornecidos, garantindo que as operações aritméticas e lógicas sejam realizadas de maneira precisa e eficiente. Em resumo, o Controlador da ULA desempenha um papel fundamental na execução das instruções do microprocessador MIPS. Ele traduz os sinais de alto nível em comandos de baixo nível que a ULA pode interpretar, garantindo que as operações sejam realizadas corretamente. Sua implementação combina lógica combinacional e mapeamento de operações, permitindo que o processador execute uma variedade de instruções de forma eficiente e precisa. Esse componente é essencial para a modularidade e o desempenho da arquitetura MIPS, reforçando sua importância no projeto geral do microprocessador.

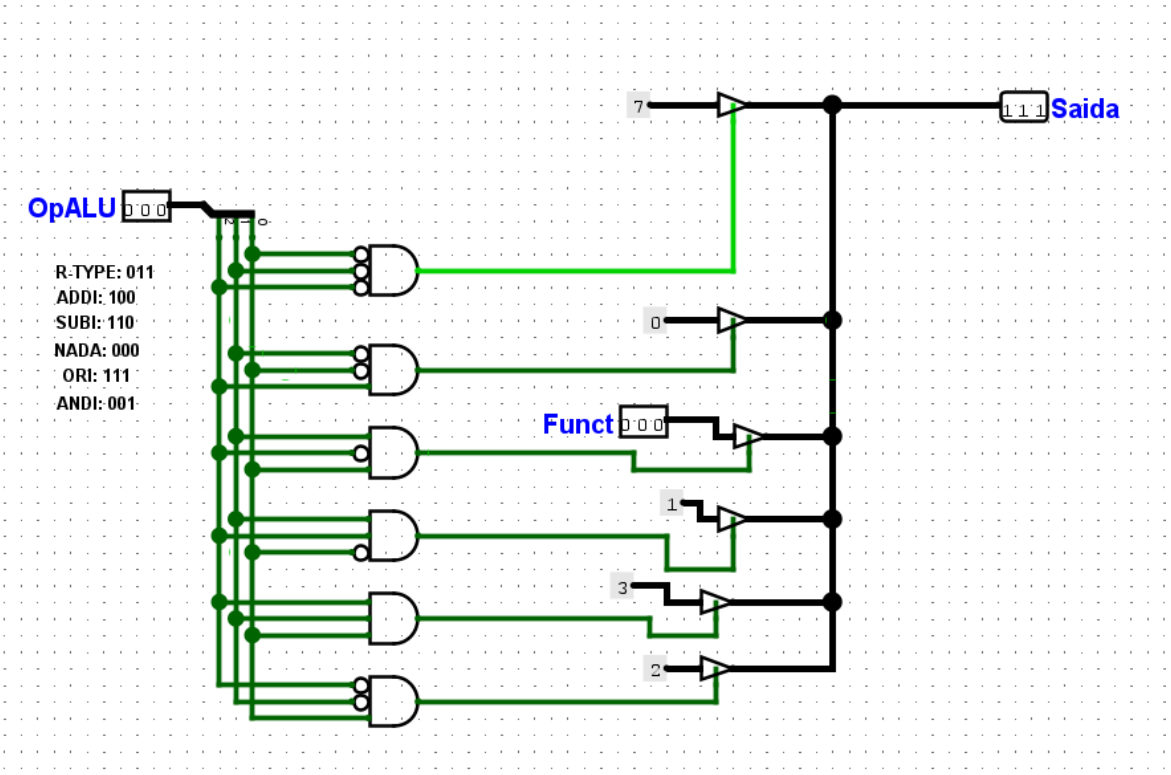

Figura 9. Imagem do Controlador da ULA.

4.8 Controlador de Sinais

O Controlador de Sinais é um dos principais componentes do processador MIPS de 16 bits, responsável por gerar os sinais de controle que coordenam a execução das instruções. Ele recebe como entrada o campo Opcode da instrução e, com base nesse valor, ativa ou desativa sinais que controlam o funcionamento dos principais blocos do processador, como a ULA, os registradores e a memória. No circuito apresentado, o Opcode é utilizado para definir a configuração correta dos sinais de controle por meio de uma lógica combinacional implementada com portas lógicas AND, OR e NOT. Cada combinação do Opcode ativa um conjunto específico de sinais, como RegDst, ALUSrc, MemToReg, RegWrite, Branch, ALUOp0, ALUOp1, Jump, MemWrite, MemRead e Halt. Esses sinais determinam como os dados são manipulados e armazenados ao longo do ciclo de execução da instrução. Por exemplo, se a instrução for do tipo R, o sinal RegDst será ativado para indicar que o destino do resultado está no campo rd da instrução. Instruções que acessam a memória, como lw e sw, ativam MemRead e MemWrite, respectivamente. O sinal Jump é ativado quando a instrução é um desvio incondicional, enquanto Branch é ativado para desvios condicionais como beq. A implementação do controlador é baseada em uma tabela verdade que mapeia os valores de Opcode para os sinais de controle correspondentes. Essa tabela serve como base para a construção do circuito lógico, garantindo que cada instrução seja corretamente interpretada e executada pelo processador. Em resumo, o Controlador de Sinais é responsável por coordenar a operação dos diferentes componentes do processador, garantindo que cada instrução seja processada corretamente. Ele atua como o "cérebro" da unidade de controle, traduzindo o Opcode em comandos que direcionam a execução do programa.

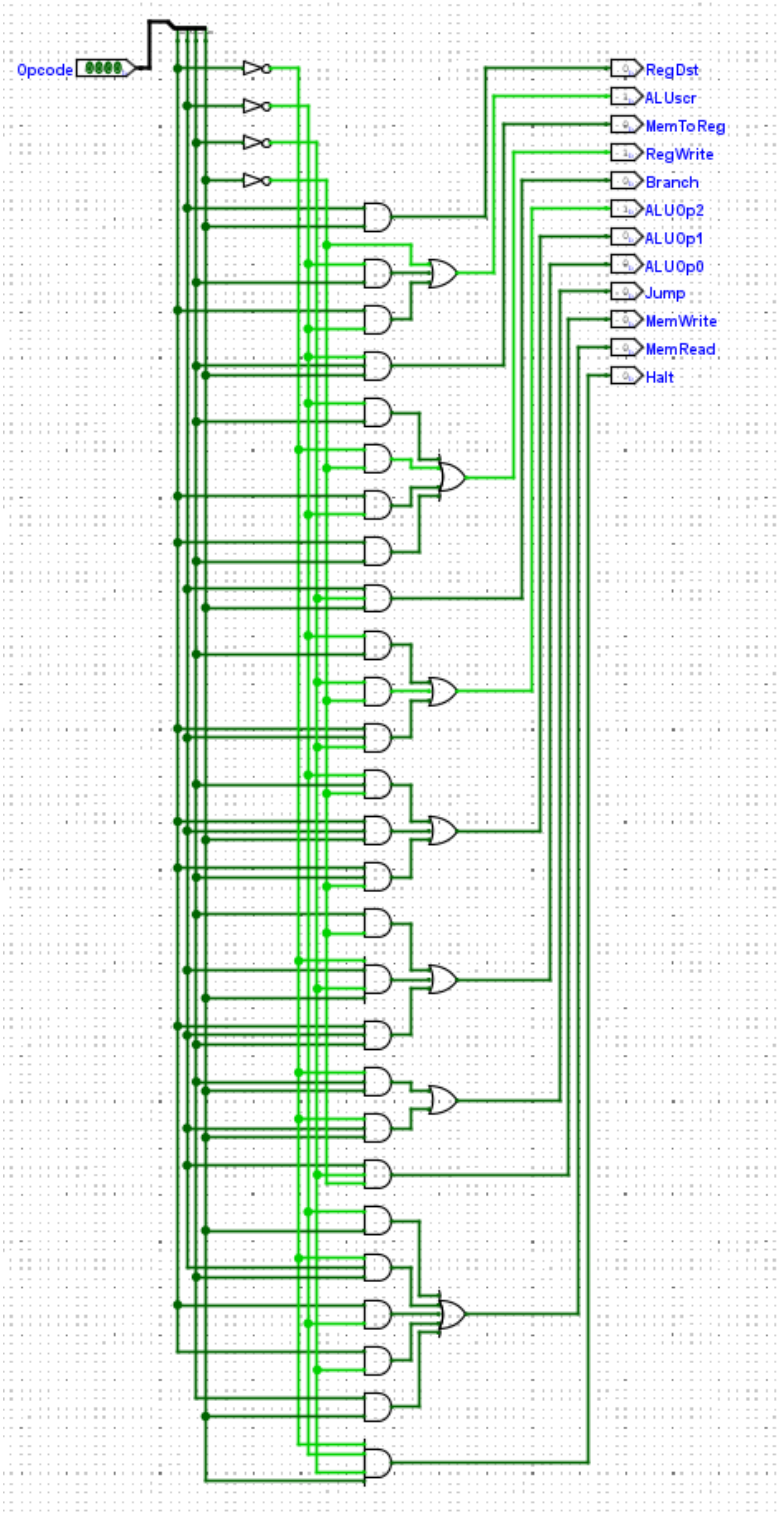

Figura 10. Imagem do Controlador do MIPS.

Opcode	RegDst	ALUscr	MemToReg	RegWrite	Branch	ALUOp2	ALUOp1	ALUOp0	Jump	MemWrite	MemRead	HALT
1111 (R TYPE)	1	0	0	1	0	0	1	1	0	0	1	0
1010 (ORI)	0	1	0	1	0	1	1	1	0	0	1	0
1011 LW	0	1	1	1	0	1	0	0	0	0	1	0
1100 SW	X	1	X	0	0	1	0	0	0	1	1	0
1101 BEQ	X	0	X	0	1	1	1	0	0	0	1	0
1000 ADDI	0	1	0	1	0	1	0	0	0	0	1	0
1001 STLI	0	1	0	1	0	0	0	0	0	0	1	0
0110 ANDI	0	1	0	1	0	0	0	1	0	0	1	0
0111 J	X	X	X	0	0	0	0	0	1	0	1	0
0001 (HALT)	0	0	0	0	0	0	0	0	0	0	1	1

Figura 11. Imagem da Tabela Verdade Utilizada para Criar o Controlador.

5. CONCLUSÃO

Este relatório apresentou o desenvolvimento completo de um processador MIPS de 16 bits, abrangendo desde a implementação dos principais componentes até a integração final do sistema. Foram desenvolvidos e testados módulos essenciais, incluindo o somador de 6 bits, o contador de programa, a memória de instruções (ROM), o banco de registradores, a Unidade Lógica e Aritmética (ULA), a memória de dados (RAM), o extensor de bits e o controlador de sinais. Além disso, o conjunto de instruções foi totalmente definido e implementado, garantindo a correta execução de operações aritméticas, lógicas, de memória e controle de fluxo. A validação dos circuitos no Logisim permitiu verificar a operação integrada do processador, confirmando que os módulos funcionam de maneira coordenada e eficiente. O controlador de sinais e a ULA foram fundamentais para garantir a correta interpretação das instruções, possibilitando a execução de programas completos no sistema desenvolvido. Com a finalização deste projeto, o processador MIPS de 16 bits agora possui todas as funcionalidades essenciais para processar instruções de maneira autônoma e eficiente. A abordagem modular adotada demonstrou-se eficaz, permitindo uma construção organizada e facilitando futuras expansões. O desenvolvimento desse processador proporcionou um entendimento aprofundado dos princípios da arquitetura MIPS, reforçando a importância do design lógico e da integração de hardware na construção de sistemas computacionais.

6. REFERÊNCIAS

OLIVEIRA, Ruy de. Microprocessadores MIPS. Material didático. Disponibilizado em aula pelo professor. Acesso em: 11 Fev. 2025.

TANENBAUM, Andrew S.; AUSTIN, Todd M. Organização Estruturada de Computadores. 6. ed. São Paulo: Pearson, 2014.

PATTERSON, David A.; HENNESSY, John L. Computer Organization and Design: The Hardware/Software Interface. 5th ed. Amsterdam: Morgan Kaufmann, 2014.

HENNESSY, John L.; PATTERSON, David A. Computer Organization and Design: The Hardware/Software Interface. 5. ed. Amsterdam: Elsevier, 2014.

LOGISIM. *Logisim Evolution: Digital Logic Simulator*. Disponível em: https://logisim-evolution.org. Acesso em: 11 fev. 2025.