Adriana Kaori Kakazu	082220004
Beatriz dos Santos Buglio	082220028
Kauê dos Santos Andrade	082220027
Lucas Oliveira Silva	082220019
Robson Guilherme Ferrarezi	082220015
Vitoria Kaori Kuriyama	082220005

Documentação do Projeto — Simulação de Processador com LogiSim

Link do projeto: rioksVi/N1_Processador_MIPS

Objetivo

Este projeto tem como objetivo demonstrar, simular e explorar os principais componentes de um processador simplificado, utilizando o simulador LogiSim (ou Logisim Evolution). O foco é entender o funcionamento básico de um processador e sua implementação prática por meio de circuitos digitais.

Ferramentas Utilizadas

- **LogiSim (ou Logisim Evolution):** Simulador de circuitos digitais para modelagem e simulação de circuitos lógicos.
- **Ambiente Acadêmico:** Campus Virtual, que oferece vídeos e materiais complementares para aprofundamento teórico e prático.

Componentes do Projeto

1. Portas Lógicas

- **Tipos utilizados:** AND, OR, NOT.
- **Descrição:** As portas lógicas são os componentes básicos para a criação de circuitos combinacionais. No simulador, as entradas e saídas são manipuladas visualmente.
- **Aplicação:** Fundamentais para a construção de circuitos básicos e operações lógicas.

2. Multiplexador (MUX)

- **Função:** Seleciona uma entre várias entradas para enviá-la à saída com base em bits de seleção.
- **Descrição:** O número de entradas aumenta com a quantidade de bits de seleção, permitindo a construção de circuitos complexos.

3. Demultiplexador (DEMUX)

- **Função:** Direciona uma única entrada para uma das várias saídas possíveis, controlado por bits de seleção.

4. Circuitos Aritméticos

- **Função:** Somador, subtrator, multiplicador e divisor.
- **Descrição:** Implementados com blocos prontos no LogiSim, não sendo necessário desenhar as portas lógicas internamente.

5. Flip-Flop D

- **Função:** Elemento básico de memória sequencial. Armazena um bit com base no pulso de clock.
- **Utilização:** Fundamental para compreender o funcionamento de registradores e controle de execução.

6. Memórias

- **ROM (Read-Only Memory):** Memória de instruções (não volátil).
- **RAM (Random Access Memory):** Memória de dados (leitura e escrita).
- **Entradas e saídas:** Endereço, dados, clock, enable, load, clear.

7. Banco de Registradores

- **Função:** Representa os 32 registradores do processador MIPS. Ele é composto por 32 flip-flops. Como esse é um modelo simplificado, usaremos apenas 8 registradores (flip-flops).
- **Descrição:** Fornece dados para operações e armazena os resultados. Esses registradores são os blocos básicos que a CPU utiliza para lidar com instruções sequenciais.

8. Unidade Lógica e Aritmética (ULA)

- **Função:** Realiza operações como soma, subtração, AND, OR, etc.
- **Conexões:** A ULA está conectada ao banco de registradores e à memória para processar dados.

9. Program Counter (PC)

- **Função:** Controla qual instrução será executada.
- **Descrição:** O PC é incrementado a cada ciclo de instrução e aponta para o endereço da próxima instrução.

10. Clock

- **Função:** É um sinal que oscila entre 0 e 1 continuamente
- **Descrição:** A cada transição de 0 para 1, um novo ciclo de operação é iniciado, acionando os elementos do processador. Componentes que reagem ao clock, armazenam valores que são processados por circuitos combinacionais: Flip-flops, registradores, memória RAM.

Ciclo de Instrução (Visão Geral)

1. PC (Program Counter) aponta para o endereço da próxima instrução.
2. A Memória de Instruções fornece a instrução ao processador.
3. Os registradores envolvidos na instrução são lidos.
4. A ULA executa a operação (se necessário).
5. O resultado da operação é armazenado ou utilizado para ler/escrever na memória.
6. O PC é incrementado para apontar para a próxima instrução.

O processador funciona em um **ciclo infinito**, no qual os elementos de estado interagem com os circuitos lógicos, levando a um novo estado a cada ciclo. O processador MIPS, por exemplo, possui múltiplos elementos de memória (PC, registradores, memória de dados/instruções) e utiliza somadores e a ULA para processar dados. Os dados percorrem um caminho definido no processador, conhecido como caminho de dados.

Projeto no LogiSim

Para uso como memória de instruções foi usada uma memória ROM, por ser uma memória não volátil, e que não será necessário a reescrita dela, assim, a memória ROM é a melhor escolha. A memória de instruções terá uma largura de 18bits e um tamanho de 8bits. A largura é definida pelo formato de memória R (register), I (instruction) e J (J-lead), que tem uma largura de 18bits em sua estrutura.

A seguir são apresentadas três tabelas, demonstrando os formatos de memória e a divisão dos bits para cada tipo de dado.

Formato R (register)																	
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
opcode			rs			rt			rd			shamt			funct		

Formato I (instruction)																	
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
opcode			rs			rt			x	address/imm							

Formato J (J-lead)																	
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
opcode			xxxxxxxxx							address							

O banco de registradores é uma estrutura crucial para armazenar valores temporários utilizados nas operações do processador, como somas, deslocamentos e comparações. O modelo original do Neomips, que utiliza 32 registradores de 32 bits, mas foi escolhida a implementação de 8 registradores de 8 bits.

O processo começa com a adição dos registradores ao circuito no simulador digital. Cada registrador é configurado com 8 bits e recebe rótulos (Reg0, Reg1, ..., Reg7) para facilitar a identificação e o acesso. Como o projeto possui 8 registradores, são necessários 3 bits de endereçamento (já que $2^3 = 8$) para acessá-los individualmente. Dessa forma, foram criadas as entradas ReadReg1, ReadReg2 e WriteReg, todas com 3 bits, para especificar quais registradores devem ser lidos ou escritos.

Além disso, foi adicionada a entrada WriteData (com 8 bits), que fornecerá o valor a ser armazenado nos registradores, e uma entrada de controle chamada RegWrite, que, com 1 bit, define se a escrita nos registradores será permitida ou não. Para garantir a sincronização e controle do processo de leitura e escrita, foram conectados sinais de Clock e Clear a todos os registradores. O Clock garante que a escrita ocorra apenas em um pulso de clock, enquanto o Clear é utilizado para zerar todos os valores armazenados.

A leitura dos dados dos registradores é realizada por meio de dois multiplexadores (MUX), um para cada saída: ReadData1 e ReadData2. Cada MUX recebe como entrada todos os 8 registradores e, com base nos valores de ReadReg1 ou ReadReg2, seleciona o registrador correspondente para enviar o valor à saída. Para configurar os MUXs, foram utilizadas 8 entradas de 8 bits cada, com 3 bits de seleção. Para otimizar o layout do circuito e evitar fios cruzando o design, foram utilizados túneis nomeados, que permitem interconectar pontos do circuito de maneira mais organizada, usando apenas nomes em comum.

A escrita no banco de registradores é um processo que exige atenção especial, pois o valor de WriteData é compartilhado entre todos os registradores. Para garantir que apenas o registrador correto seja atualizado, foi utilizado um demultiplexador (DEMUX), que recebe o sinal de controle RegWrite. Com base no endereço WriteReg, o DEMUX habilita a escrita apenas no registrador selecionado, evitando a sobrescrição indevida dos outros registradores.

Após a implementação da lógica do banco de registradores, foi realizada uma série de testes para validar seu funcionamento. Inicialmente, o Clear foi ativado para garantir que todos os registradores estivessem zerados. Em seguida, foram atribuídos valores distintos a cada registrador, começando por 0 no Reg0, 1 no Reg1, e assim por diante até o Reg7. A sequência escolhida foi a de operações: o endereço do registrador é definido (WriteReg), o controle RegWrite é ativado, o valor de entrada é fornecido em WriteData, e é aplicado um pulso de Clock. Após a execução, é verificado se os dados foram corretamente armazenados ao alterar os valores de ReadReg1 e ReadReg2 e observar as saídas de ReadData1 e ReadData2.

Com todos os testes concluídos e a implementação validada, essa estrutura é encapsulada em um componente nomeado Banco de Registradores, que agora pode ser utilizado como um bloco modular no circuito principal do processador. O componente possui todas as

entradas e saídas necessárias para seu funcionamento correto: os três endereços de leitura e escrita (ReadReg1, ReadReg2, WriteReg), os dados de entrada e saída (WriteData, ReadData1, ReadData2), o sinal de controle RegWrite, além dos sinais de Clock e Clear.

Com a implementação do banco de registradores concluída, o sistema está funcional e pronto para ser integrado ao restante do processador. Esse componente modular é agora uma base sólida para o processamento de dados e pode ser expandido e testado conforme avançamos na construção do processador. Assim, é possível seguir para o próximo passo, que é a implementação da Unidade Lógica e Aritmética (ULA), responsável pelas operações matemáticas e lógicas.

O circuito da ULA (Unidade Lógica e Aritmética) possui um total de quatro entradas: dois números (A e B), uma entrada de 3 bits que indica qual das operações matemáticas ou lógicas será realizada com os dois números (por exemplo, 000 = soma) e uma entrada de deslocamento (Shift Amount); e duas saídas: o resultado da operação (R) e uma comparativa (Igual). Essa segunda saída retorna verdadeiro (1) se as entradas forem iguais.

Como o processador é de 8 bits, as entradas A e B e a saída também comportam essa mesma capacidade. A entrada de Função possui capacidade de 3 bits, refletindo as 8 possíveis operações (4 operações matemáticas e 4 operações lógicas), a saída de comparação utiliza apena 1 bit e a entrada de deslocamento tem capacidade de 3 bits, já que a quantidade de casas deslocadas será, no máximo, 7 (0 a 7 ocupam 8 posições).

No Logisim, já existem blocos prontos para cada operação. São selecionadas da pasta Aritmética os blocos Somador, Subtrator, Multiplicador, Divisor, Negador, Comparador, Deslocador para a esquerda e Deslocador para a direita. Em seguida, é atribuído um código para cada operação. Por questões de praticidade, foram atribuídos códigos entre 000 e 111 para as operações na ordem em que foram selecionados (definida acima).

Para a seleção da operação a ser utilizada, é utilizado um conjunto de Multiplexadores e Demultiplexadores. O circuito possui, no total, 2 Demultiplexadores e um Multiplexador, conectados, respectivamente, com as entradas A e B e com a saída das operações, além de ambos se conectarem à entrada da Função. Dependendo do código enviado pela função, Excluindo os deslocadores e o negador, todas as funções recebem A como valor primário e B como secundário. No Negador só é recebido o valor de A, que traz como resposta o valor negativo dele, e nos deslocadores recebe A como valor primário, e como valor secundário recebe Shamt, que representa o valor (quantidade) de deslocamento, ou deja o valor a ser deslocado.

Para o funcionamento da correto da ULA foi criado outro componente, o Controle ULA, ele serve para independentemente do tipo (formato) da memória, a ULA funcionar corretamente, isso ocorre porque, ao utilizar um formato tipo R, temos como seus números finais (2, 1, 0) “função”, que define a função a ser feita, porém em um formato tipo I, a função não existe, e em seu lugar temos o “Address”, mas a ULA se mantém necessária, por isso foi criado o Controle ULA.

Junto do Controle ULA foi criado uma entrada “ALUop” que define em um Multiplexador qual dado vai ser recebido na saída, a entrada “ALUop” é definida na unidade de controle. As opções no multiplexador vão para as funções: SOMADOR, MENOR QUE, FUNÇÃO e SOMADOR. Nessa ordem a partir de 0. Sendo que Função pode ser qualquer função que esteja no formato do dado.

A Unidade de Controle (UC) é o elemento central do processador, ele que controla o que vai ocorrer, o tipo de dado liberado e quando. Ela é “controlada” pelo UPCODE (OPCODE), que está no início de todos os formatos de dados (R, I e J). Sendo que para cada formato diferente tem um upcode diferente.

A UC tem uma entrada upcode, e 9 saídas: RegDst, RegWrite, ALUSrc, MemWrite, MemRead, MemToReg, Branch, Jump e ALUop.

RegDst é a entrada de seleção de um multiplexador que ajuda a selecionar o endereço do registrador que o dado vai ser salvo, dependendo do formato, vai ser uma entrada diferente.

RegWrite e MemWrite são entradas ENABLE elas autorizam o armazenamento de um registrador ou de uma memória.

MemRead é um ENABLE que autoriza a leitura da memória.

ALUSrc é uma entrada de seleção de um multiplexador que seleciona se vai entrar um registrador, ou um endereço (Address) na ULA, dependendo do formato (estrutura R, I ou J) da memória.

MemToReg é uma entrada de seleção de um multiplexador que seleciona sea saída da ULA ou uma memória da memória de dados, vai para o banco de registradores.

Jump é uma função que pula no contador de programas o valor do endereço de uma memória.

Branch compara o valor de dois registradores, se forem iguais, então ele dá um jump, no valor de seu endereço (Address).

A memória de dados é uma memória que registramos dados temporários, como resultados de uma soma, que podemos utilizar, mas não queremos salvar. Para isso foi utilizado uma memória RAM.

Na memória de dados o endereço da memória é dado pela saída da ULA, e o dado a ser registrado é a segunda saída do Banco de Registradores. Para a leitura não ser ao mesmo tempo que a escrita nesse tipo de memória precisamos diferenciar o clock, para o clock da saída da ULA não condizer com clock da memória, assim barramos (colocamos uma porta lógica NOT no seu clock) para fazer a diferenciação.

A saída da memória pode ser alimentada no Banco de Registradores, dependendo de um seletor e de sua estrutura.

Desta forma o processador está pronto para uso, é só colocar as operações na Memória de Instruções, e ativar o clock, que começa a funcionar.

Esse processador MIPS, é um processador simplificado, com menos bits e menos funções que muitos outros, mas é um processo completo e dá a ideia de como os processadores mais avançados funcionam.