Universidade Estadual do Oeste do Paraná - UNIOESTE
Centro de Engenharias e Ciências Exatas - CECE
Disciplina de Organização e Arquitetura de Computadores
Docente: Camile Frazão Bordini

Isabela Pimentel Loebel
Kevin Willian de Oliveira
Gustavo Juliano Borges

Trabalho 2 e 3
Relatório

Foz do Iguaçu
2024

1. Introdução

Na disciplina de Organização e Arquitetura de Computadores, foi estudada a arquitetura RISC-V monociclo, sendo um conjunto de instruções (ISA) baseado em princípios RISC (acrônimo de *Reduced Instruction Set Computing*, em português, “Computação de conjunto de instruções reduzidas”). RISC-V é livre para ser usado para qualquer finalidade, permitindo a qualquer pessoa ou empresa projetar e vender chips e software RISC-V sem precisar pagar royalties.

Iniciado em 2010 na Universidade da Califórnia, sendo um projeto de código aberto sem fins lucrativos, muitos colaboradores são voluntários ou fazem parte de outras empresas e trabalham no projeto de fora da universidade.

O RISC-V foi projetado para implementações de alto desempenho e baixo consumo de energia. Sendo um conjunto limpo e modular, trabalhando com bases de 32, 64 e 128 bits, com várias opções de extensão em ponto flutuante.

A fim de validar em prática os conhecimentos obtidos durante a disciplina, foi disposto um problema a ser implementado, ou seja, uma simulação da implementação monociclo desta arquitetura, considerando uma base de 32 bits. Para dar início a esse projeto e sequência neste relatório é necessário entender alguns termos importantes como:

- **Arquitetura**: Conjunto de recursos percebidos pelo programador em linguagem de máquina, tais como: registradores, tipos de dados, organização de memória, modos de endereçamento, conjunto de instruções;

- **RISC-V:** Estendendo a sigla temos *Reduced Instruction Set Computing - V,* indicando sua natureza de conjunto de instruções de computação com um conjunto reduzido de operações. Uma arquitetura que tem como princípios: “simplicidade favorece a regularidade” e “menor é mais rápido”;

- **Sinais de controle:** Sinais elétricos ou lógicos utilizados para controlar diversas operações no processador. Esses sinais são fundamentais para coordenar o funcionamento interno do processador e garantir que as instruções sejam executadas corretamente. Em uma implementação RISC-V esses sinais são responsáveis em controlar unidades funcionais, selecionar registradores, controlar multiplexadores, sequenciar instruções, controlar fluxo de dados;

- **Registradores:** Os registradores são pequenas unidades de armazenamento localizadas no processador de um computador. Eles são usados para armazenar

temporariamente dados e instruções que estão sendo processados ativamente pela CPU. A RISC-V especifica um conjunto fixo de registradores de propósito geral *x0* a *x31* e alguns registradores especiais, como o PC e os registradores de *status*;

- **PC:** Registrador usado para controle de instrução a ser analisada, considerando que toda instrução terá 32bits, sendo 4 bytes, o PC deve incrementar a cada 4 espaços para caminhar entre as instruções;

- **Funct7:** Um campo de bits usado para codificar parte da operação a ser realizada pela instrução. Em instruções do tipo R, *funct7* é usado para especificar operações de ALU. Por exemplo, em operações como add e sub, *funct7* pode ser usado para diferenciar entre as duas operações;

- **Funct3:** Outro campo de bits usado para codificar parte da operação a ser realizada pela instrução. Assim como *funct7, funct3* é usado em instruções do tipo R para especificar operações de ALU, porém mais detalhadamente. Por exemplo, em uma instrução add, *funct7* pode indicar que é uma operação de adição, enquanto *funct3* pode especificar se é uma adição com sinal ou sem sinal;

- **RS1:** O campo que indica o primeiro registrador de origem em uma instrução, é usado para especificar o registrador de onde os dados serão lidos como entrada para a operação;

- **RS2:** Assim como *rs1,* em instruções de operação de registrador a registrador, *rs2* é usado para especificar o segundo registrador de onde os dados serão lidos como entrada para a operação.

2. Problema Proposto

Implementar um simulador RISC-V Monociclo em uma linguagem de programação de sua escolha. O simulador deve conseguir executar o seguinte conjunto de instruções: *add, sub, and, or, addi, lw, sw, beq* e *bne*.

O programa deve interpretar corretamente as instruções fornecidas nos 3 arquivos de testes. Observam-se as operações:

- *addi* (formato I: *opcode*=0010011; *funct3*=000)
- *bne* (formato S: *opcode*=1100011; *funct3*=001)

3. Desenvolvimento

Para obtenção de uma solução satisfatória, buscou-se entender a lógica de funcionamento do processamento da entrada binária, sua tradução e por fim, o que de fato ela deveria executar. Como todo sistema, a solução proposta primeiramente verifica a validade do valor inserido como entrada, ou seja, o bloco de binários a ser analisado.

Para os passos seguintes, utilizamos como material de apoio, o quadro de constantes das instruções RISC-V disponibilizado nos materiais de aula, podendo ser visualizado na figura 1 abaixo.

Name (Bit position)	31:25	24:20	19:15	14:12	11:7	6:0	Comments
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format

Figura 1 - Quadro de constantes.

Também mostrou-se necessário o entendimento da maneira que o gerador de imediato constrói as constantes e estende o último bit, para isso, visualize a figura 2.

		Immediate Output Bit by Bit																													
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Instruction	Format	Immediate Input Bit by Bit																															
Load, Arith. Imm.	I	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I31	I30	I29	I28	I27	I26	I25	I24	I23	I22	I21	I20
Store	S	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	I11	I10	I9	I8	I7							
Cond. Branch	SB	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"	I7												0

Figura 2 - Gerador de imediato.

Com isso esclarecido, inicia-se o processo de tradução, visto que o sistema deverá entender a entrada, e após isso, realizar sua operação. Usaremos de exemplo um bloco de binários, podendo ser verificado na tabela 1.

Binário bruto	00000000110001011000011100110011

Tabela 1 – Bloco binário de exemplo.

A estratégia de tradução é feita de forma que o primeiro artefato a ser analisado é o *opcode*, sendo os 7 últimos números do binário como mostrado na figura 1, o qual indica, qual o tipo da instrução, podendo ser visualizado na tabela 2 os possíveis valores e seu tipo significante.

opcode	tipo da instrução
0110011	Tipo R
0100011	Tipo S
1100011	Tipo B
0010011	Tipo I
0000011	Tipo I

Tabela 2 – *Opcode* e sua instrução correspondente.

Após obtermos o tipo da instrução, é encaminhado para o seu processamento, onde será feito o restante das traduções, visto que, assim como mostrado na figura 1, cada tipo de instrução obtém seus valores de formas diferentes.

Seguindo o exemplo da tabela 1, ao iniciar o tratamento do restante das informações contidas no binário da instrução, é feita a separação do binário, onde cada variável (*rs1, rs2, rd, funct3* e *funct7*) receberão seus valores separando as partes relevantes aos mesmos da instrução de entrada.

Binário bruto	00000000110001011000011110 (sem opcode)				
Binário particionado	0000000	01100	01011	000	01110
Constante tipo R	*funct7*	*rs2*	*rs1*	*funct3*	*rd*

Tabela 3 – Bloco binário de exemplo separado e variáveis correspondentes.

Separadas as variáveis, verifica-se o *funct7*, no caso da instrução do tipo R, podendo ser os valores indicados na tabela 4. Nota-se que a segunda opção de *funct7* oferece mais que um tipo de operação resultante, isso deve-se ao fato que a percepção final de qual será a operação é obtida na verificação do *funct3*, como mostrado na tabela 5.

funct7	operação
0100000	sub
0000000	add **OU** or **OU** and

Tabela 4 – *Funct7* e operação correspondente.

funct3	operação
000	add
110	or
111	and

Tabela 5 – *Funct3* e operação correspondente.

Com isso traduzido, é feito o processamento de fato da instrução, e por fim, serão exibidos os sinais de controle de cada operação, seguindo a tabela 6.

	Sinais de Controle						
	ALUOp	**Branch**	**ALUScr**	**MemToReg**	**ReqWrite**	**MemRead**	**MemWrite**
addi	00	0	1	0	1	0	0
add, sub, and e or	10	0	0	0	1	0	0
sw	00	0	1	-	0	0	1
lw	00	0	1	1	1	1	0
beq	01	1	0	-	0	0	0
bne	11	1	0	0	0	0	0

Tabela 6 – Sinais de controle referente à instrução analisada.

Por fim, é listado os registradores e a pilha da memória também para serem exibidas em tela para visualização passo a passo das instruções. Todo o processo é repetido, com fluxo variável consoante o tipo da instrução a ser interpretado, para melhor visualização, foi disponibilizado um fluxograma, visualize a Figura 3.

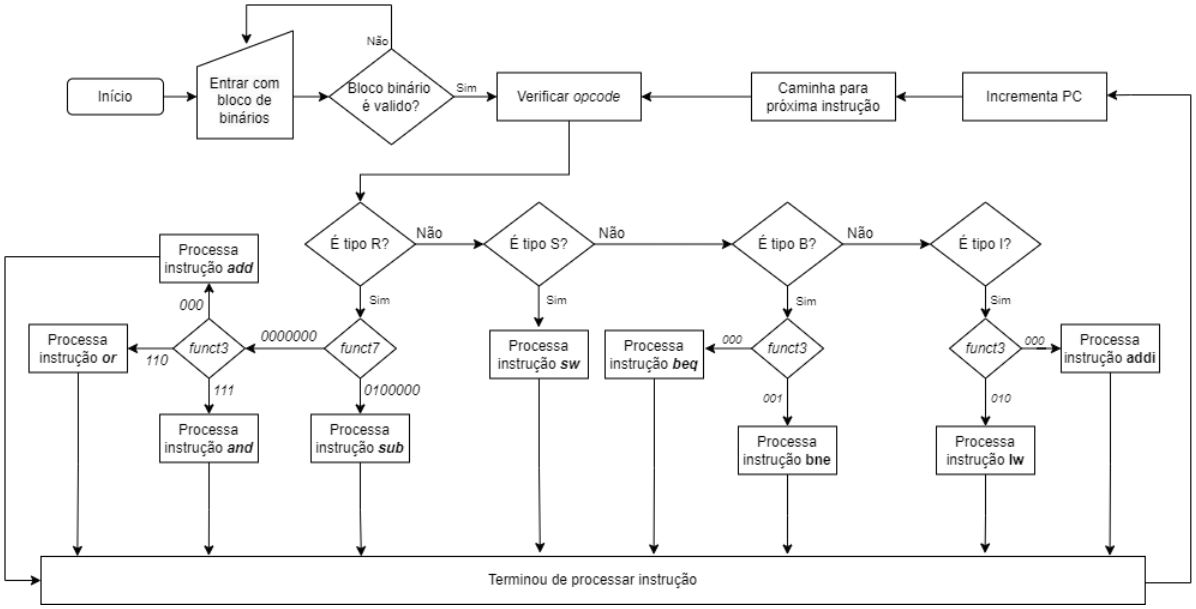

Figura 3 – Fluxograma do projeto.

3.1. Executando o programa

Para a execução do programa desenvolvido, será necessário seguir os passos:

- Extraia os arquivos para uma pasta separada;
- Abra o arquivo HTML no navegador (de preferência use o Google Chrome);
- Cole o bloco de binários no campo de entrada (sem espaços excedentes, somente linhas com 32 bits);
- Clique em enviar para preparar a máquina;
- Clique em avançar/voltar instrução para executar as instruções passo-a-passo.

4. Pontos positivos e negativos

- Pontos negativos

Controle das constantes na instrução bne, visto que caso a instrução seja do tipo bne e a constante for negativa, é necessário fazer o tratamento com o “complemento de 2”.

- Pontos positivos

A implementação do simulador proporcionou uma oportunidade única para explorar a lógica por trás da execução de instruções em um processador monociclo. Isso permitiu uma compreensão mais profunda dos mecanismos internos de um processador RISC-V e como as instruções são interpretadas e executadas no nível do hardware.

O trabalho envolveu colaboração com colegas de classe, promovendo habilidades de trabalho em equipe, comunicação e resolução de problemas em um ambiente colaborativo, habilidades essenciais em muitos contextos profissionais.

5. Conclusão

A implementação do simulador RISC-V monociclo representou um desafio substancial e pertinente, proporcionando uma aplicação prática dos conhecimentos adquiridos na disciplina de Organização e Arquitetura de Computadores. Este empreendimento contribuiu significativamente para uma compreensão detalhada dos conceitos fundamentais da arquitetura de computadores, além de permitir uma exploração minuciosa da lógica subjacente à execução de instruções em um processador monociclo.

Através de uma análise meticulosa das instruções contidas nos conjuntos de testes fornecidos, foi possível conceber e implementar um simulador apto a interpretar de maneira precisa as instruções RISC-V monociclo, aderindo estritamente às suas especificações. Esta conquista foi possível mediante a utilização criteriosa das informações contidas nos registros de constantes das instruções, combinada com uma compreensão profunda da lógica por trás da geração de imediatos. Como resultado, conseguimos traduzir eficazmente as instruções binárias em operações legíveis e aplicáveis, assim como realizar as operações a elas referentes.

6. Referencias Bibliográficas

PATTERSON, D. A.; HENNESSY, J. L. Computer Organization and Design: The Hardware Software Interface - Risc-V edition, second edition, 2021.

WIKIPEDIA. **RISC-V**. Disponível em: <https://pt.wikipedia.org/wiki/RISC-V>. Acesso em 20 de mar de 2024.

BORDINI, C. F. Formação das constantes. 2024. Material de aula.