

# CCF 252 - Trabalho Prático 2

## Caminho de Dados do RISC-V (versão simplificada)

Iury M. Pereira<sup>4671</sup>, João Vitor C. Lobo<sup>4693</sup>,  
Matheus N. Peixoto<sup>4662</sup>, Vinicius A. Gontijo<sup>4708</sup>

<sup>1</sup>Ciência da Computação – Universidade Federal de Viçosa (UFV) - Campus Florestal

iury.m.pereira@ufv.br, joao.lobo@ufv.br,

matheus.n.peixoto@ufv.br, vinicius.a.gontijo@ufv.br

**Abstract.** *This documentation describes implementing a simplified version of a RISC-V datapath in the Verilog hardware description language, showing its key points and how to run it on a Windows 10 machine.*

**Resumo.** *Esta documentação descreve a implementação de uma versão simplificada de um caminho de dados do RISC-V na linguagem de descrição de equipamentos físicos Verilog, exibindo seus pontos-chaves e como executá-lo em uma máquina com Windows 10.*

### 1. Introdução

A realização deste projeto consiste no desenvolvimento de um dos principais componentes de um computador, o Caminho de Dados [Figure 2], também conhecido por seu nome em inglês, *Datapath*

O caminho de dados pode ser definido como uma coleção de unidades lógico aritméticas responsáveis pela leitura, interpretação e execução de diversas instruções. Para isso, conta com funções que ficam encarregadas de fazerem operações como adições, subtrações, desvios, leitura e escrita de dados, dentre outras que serão abordadas ao longo desta documentação.

Para a execução do *Datapath* foi utilizada a linguagem Verilog, a qual é universalmente utilizada na descrição de equipamentos físicos (*Hardware Description Language — HDL*) para projetar sistemas eletrônicos. Verilog é frequentemente utilizada através de simulações para análises de performance e testes.

Na implementação deste trabalho foi utilizado o Icarus Verilog [Figure 1], uma ferramenta para a compilação do código em tal linguagem.



Figure 1. Icarus Verilog

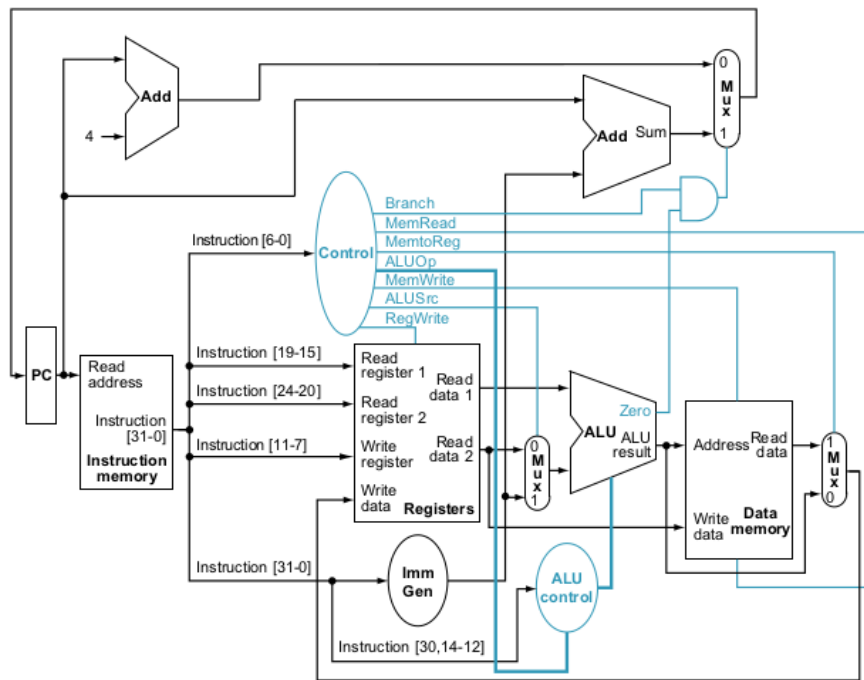


Figure 2. Caminho de Dados

## 2. Datapath (Caminho de Dados)

Como citado na introdução deste documento, o Caminho de Dados possui algumas funções que são de extrema importância para o seu funcionamento e, consequentemente, para o bom funcionamento do Computador.

A seguir, serão explicadas algumas destas principais funções.

### 2.1. Program Counter (PC)

Contendo o endereço da instrução a ser executada, o Program Counter, a medida que o código vai sendo executado, é incrementado ou para seguir para a instrução seguinte ou, para no caso de um desvio, para a instrução condizente com a condicional.

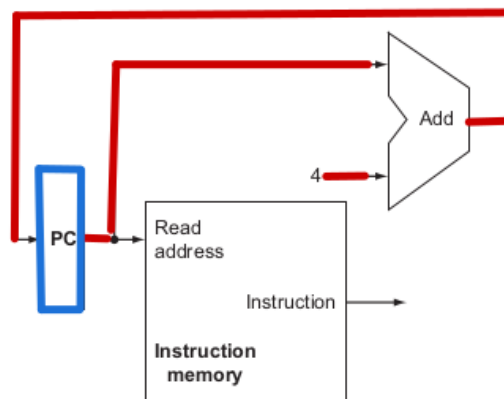


Figure 3. Program Counter (PC)

## 2.2. Instruction Memory

Este é o local no caminho de dados onde as instruções condizentes com o Program Counter são armazenadas e fornecidas aos demais componentes do datapath de acordo com o seu tipo.

Essa distribuição de dados se dá da seguinte forma:

- Do bit 0 até o sexto são encaminhados para a Unidade de Controle [seção 2.3]
- Do sétimo até o décimo primeiro são destinados para o Registrador de Escrita (Write Register)
- Os bits contidos entre o décimo quinto até o vigésimo-quarto são destinados aos Registradores de Leitura (Read Register)
- Também podem ser distribuídos para o Immediate Generator, o qual terá sua função explicada no decorrer da documentação.

Tais instruções estão presentes no arquivo "entradaBin.bin", as quais estão escritas em binário seguindo as especificações do RISC-V.

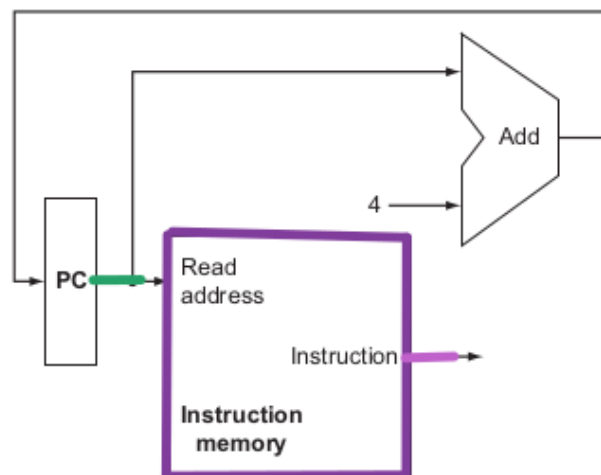


Figure 4. Instruction Memory

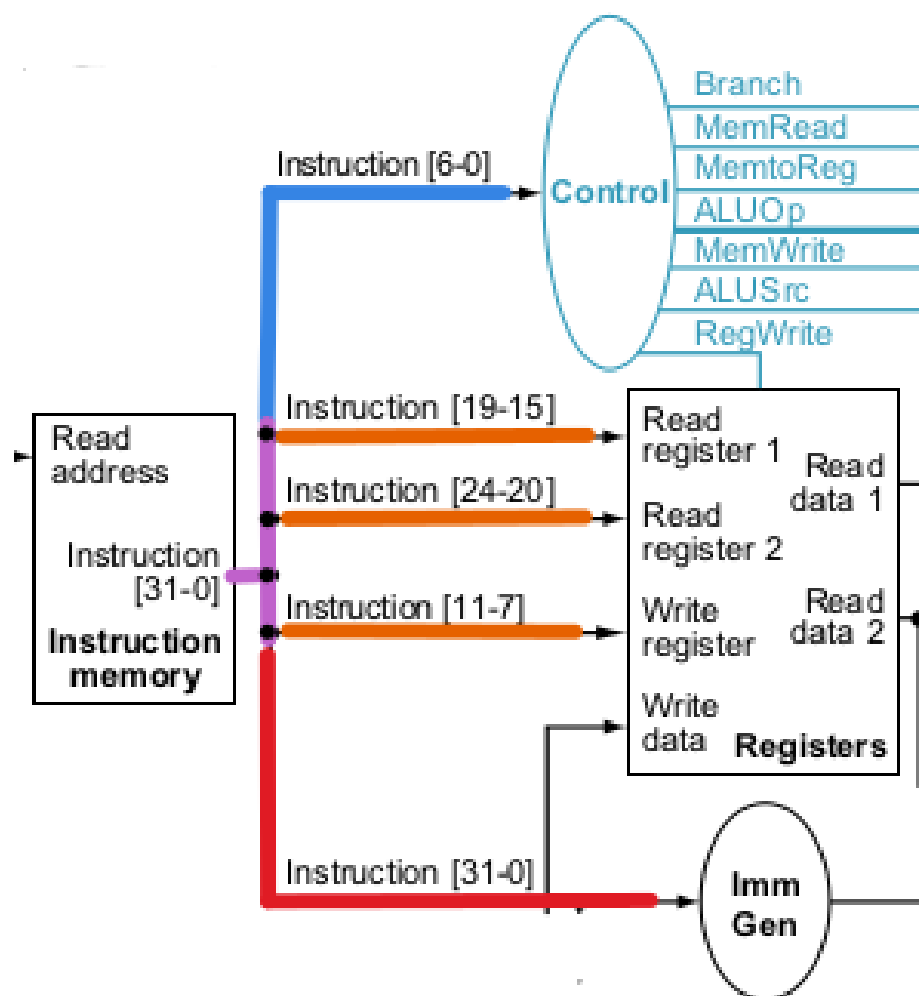


Figure 5. Divisão dos bits a partir do *Instruction Memory*

### 2.3. Control Unit (Unidade de Controle)

Parte do datapath que recebe os bits correspondentes ao opcode da instrução e em seguida determina os caminhos a serem percorridos pelos dados.

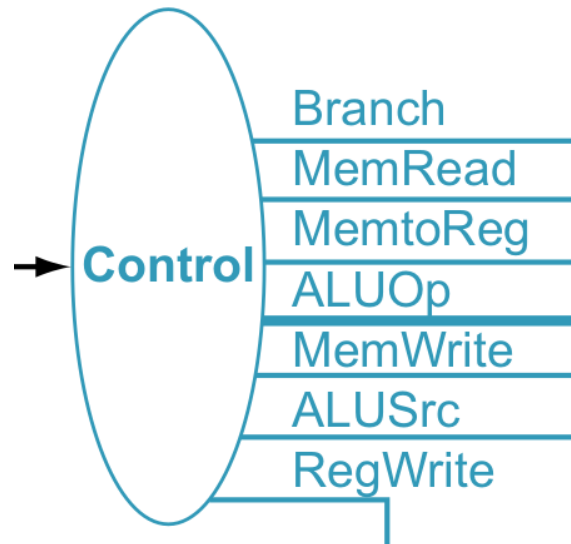


Figure 6. Unidade de Controle

### 2.4. Register File (Registradores)

Nessa parte ficam contidos todos os registradores, possuindo duas porta de leitura e uma de escrita.

Como adiantado na seção 2.2, parte dos bits da instrução são divididos entre as portas de entrada de acordo com o tipo de registrador correspondente.

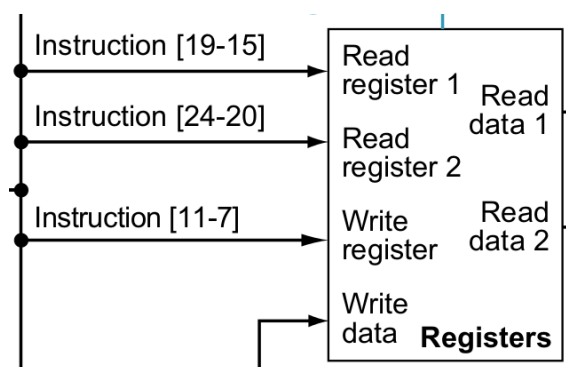


Figure 7. Register File (Registradores)

### 2.5. Sign-extend (Extensor de Sinal)

O extensor de sinais é um dos componentes do datapath que é utilizado para aumentar o tamanho dos dados que passam por ele. Funciona replicando o bit mais significativo, mantendo, dessa maneira, o sinal original.

## 2.6. MUX (Multiplexadores)

Também podendo ser chamado de seletor de dados, o multiplexador tem como função receber dados, escolher dentre multiplas fontes (de acordo com as configurações de sua linha de controle) e orientar uma dessas fontes ao seu destino.

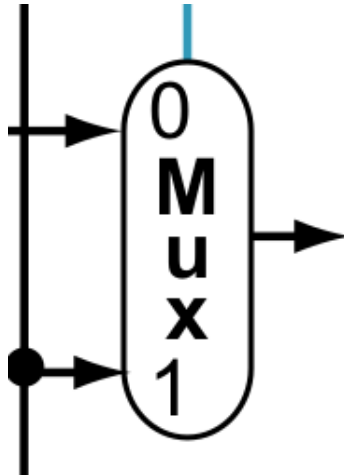


Figure 8. MUX (Multiplexadores)

## 2.7. Arithmetic-Logical Unit (ALU)

A ALU (em português, ULA - Unidade Lógica Aritmética) é a parte responsável pela execução da instrução lógicas (OR, XOR, AND, por exemplo), aritméticas (adições, subtrações, comparações, multiplicações e divisões), além das instruções de deslocamento

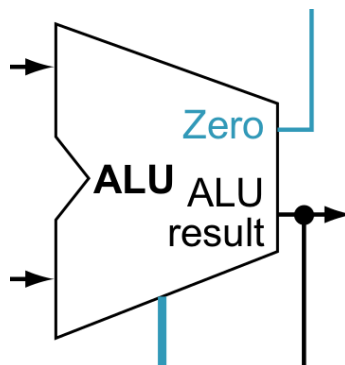


Figure 9. Arithmetic-Logical Unit (ALU)

## 2.8. ALU Control

A ALU Control determina o que a Unidade Lógica Aritmética executará. Isso é feito com base no tipo de operação e nos seus códigos de controle.

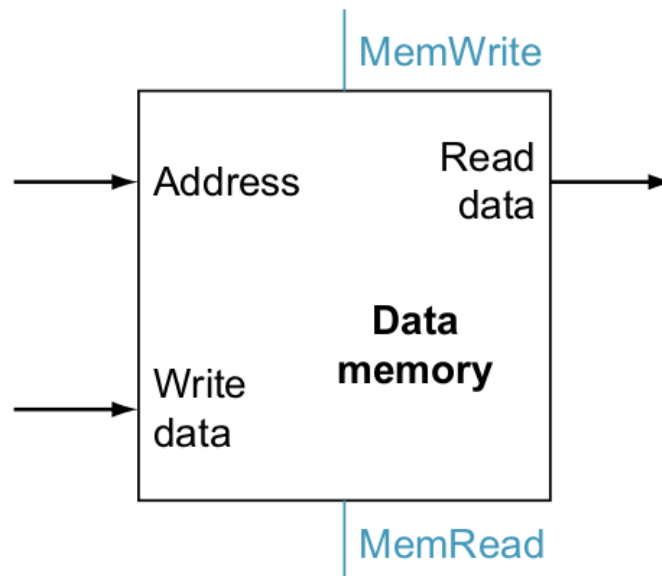
**Table 1. Códigos de controle da ALU, com as linhas de controle da ALUOp**

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract

Instruction opcode	ALUOp	Operation	Funct7 field	Funct3 field	Desired ALU action	ALU control input
lw	00	load word	XXXXXXX	XXX	add	0010
sw	00	store word	XXXXXXX	XXX	add	0010
beq	01	branch if equal	XXXXXXX	XXX	subtract	0110
R-type	10	add	0000000	000	add	0010
R-type	10	sub	0100000	000	subtract	0110
R-type	10	and	0000000	111	AND	0000
R-type	10	or	0000000	110	OR	0001

## 2.9. Data Memory

Em português Memória de Dados, é onde são escritas as instruções de armazenamento.



**Figure 10. Data Memory (Memória de Dados)**

### 3. Instruções Suportadas

Além das 7 instruções presentes na especificação deste trabalho (ADD, SUB, AND, OR, LD, SD, BEQ), o código ainda suporta a instrução do tipo I "ADDI"

### 4. Execução

Para a execução deste programa, como citado no capítulo "Introdução", é preciso que o software "Icarus Verilog" esteja instalado na máquina.

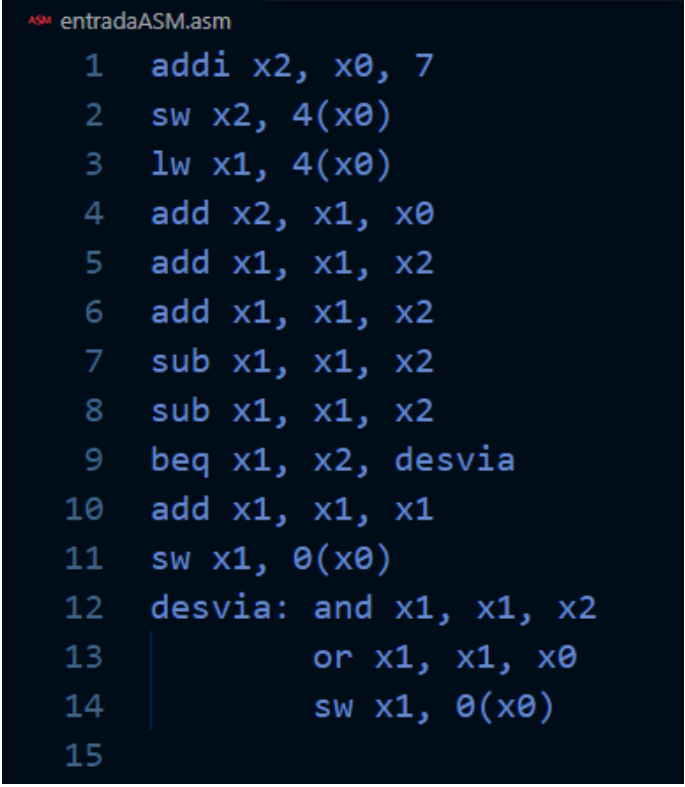
Caso ainda não esteja, basta instalá-lo pelo seguinte link: <http://bleyer.org/icarus/>

Feito isso, agora no terminal do computador, basta digitar os seguintes comandos:

```
iverilog -o dataPath_tb.vvp dataPath_tb.v
vvp dataPath_tb.vvp
```

### 5. Resultados

Utilizando os seguintes comandos em RISC-V:



```
ASMA entradaASM.asm
1  addi x2, x0, 7
2  sw x2, 4(x0)
3  lw x1, 4(x0)
4  add x2, x1, x0
5  add x1, x1, x2
6  add x1, x1, x2
7  sub x1, x1, x2
8  sub x1, x1, x2
9  beq x1, x2, desvia
10 add x1, x1, x1
11 sw x1, 0(x0)
12 desvia: and x1, x1, x2
13          or x1, x1, x0
14          sw x1, 0(x0)
15
```

Figure 11. Comandos em Assembly RISC-V

foi obtido como resultado final após a execução:



```
Register [      0]:      0
Register [      1]:      7
Register [      2]:      7
Register [      3]:      0
Register [      4]:      0
Register [      5]:      0
Register [      6]:      0
Register [      7]:      0
Register [      8]:      0
Register [      9]:      0
Register [     10]:      0
Register [     11]:      0
Register [     12]:      0
Register [     13]:      0
Register [     14]:      0
Register [     15]:      0
Register [     16]:      0
Register [     17]:      0
Register [     18]:      0
Register [     19]:      0
Register [     20]:      0
Register [     21]:      0
Register [     22]:      0
Register [     23]:      0
Register [     24]:      0
Register [     25]:      0
Register [     26]:      0
Register [     27]:      0
Register [     28]:      0
Register [     29]:      0
Register [     30]:      0
Register [     31]:      0
-----FIM DE PROGRAMA-----
dataPath_tb.v:18: $finish called at 56 (1s)
```

Figure 12. Resultado final do Dataphat

## 6. Considerações Finais

Com a obtenção dos resultados demonstrados no capítulo anterior, o grupo obteve êxito na execução do código na linguagem Verilog da versão simplificada de um *Datapath* (Caminho de Dados)

## **7. Referências**

[Patterson and Hennessy 2021]

### **References**

Patterson, D. A. and Hennessy, J. L. (2021). Computer organization and design: Risc-v edition. In , editor, *2a Edição*. Editora Morgnan Kaufman.