UNIVERSIDADE FEDERAL DE OURO PRETO - UFOP
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS - ICEA
ENGENHARIA DE COMPUTAÇÃO

RISC-V: Simulação e Depuração de um Processador Monociclo
Grupo 07

Laila Ferraz Souza Lima

João Monlevade – MG
16 de agosto de 2025

Resumo

Este documento detalha o processo de implementação, depuração e validação de um processador de ciclo único compatível com um subconjunto da arquitetura RISC-V, como projeto para a disciplina de Organização e Arquitetura de Computadores II. O trabalho abrange a descrição da arquitetura, a implementação dos módulos em Verilog e, com especial ênfase, o processo iterativo de verificação funcional. O conjunto de instruções do Grupo 07, incluindo operações aritméticas, lógicas, de acesso à memória de bytes e de desvio, foi implementado e testado, com a análise dos erros de simulação servindo como guia para a correção e validação final do projeto.

Abstract

This document details the implementation, debugging, and validation process of a single-cycle processor compatible with a subset of the RISC-V architecture, as a project for the Computer Organization and Architecture II course. The work covers the architecture's description, the implementation of modules in Verilog, and, with special emphasis, the iterative process of functional verification. The instruction set for Group 07, including arithmetic, logical, byte-level memory access, and branch operations, was implemented and tested, with the analysis of simulation errors serving as a guide for the final correction and validation of the design.

Sumário

1 Introdução

A arquitetura de conjuntos de instruções (ISA) RISC-V tem se destacado como uma plataforma aberta e extensível, ideal para o ensino e a pesquisa em arquitetura de computadores. Sua simplicidade e modularidade permitem o desenvolvimento incremental de processadores. Este projeto foca na implementação de um processador RISC-V de ciclo único para solidificar os conceitos de caminho de dados e controle, traduzindo o design teórico em uma implementação funcional em Verilog e validando seu comportamento.

2 Desenvolvimento do Processador

O desenvolvimento seguiu uma abordagem modular, onde cada componente principal do caminho de dados foi implementado como um módulo Verilog distinto e interconectado no módulo de topo.

2.1 Arquitetura e Conjunto de Instruções

A arquitetura implementada é a de ciclo único, onde cada instrução é completada em um único ciclo de clock. O design foi adaptado para o conjunto de instruções do Grupo 07:

- **Tipo-R (Aritmética/Lógica):** `add`, `or`, `and`, `srl`.
- **Tipo-I (Carga):** `lb` (Load Byte).
- **Tipo-S (Armazenamento):** `sb` (Store Byte).
- **Tipo-B (Desvio):** `beq` (Branch if Equal).

2.2 Módulos Implementados

O sistema foi dividido em módulos Verilog, incluindo as unidades de controle (`control_g7.v`, `alu_control_g7.v`), a Unidade Lógica e Aritmética (`alu_g7.v`), o banco de registradores (`reg_file.v`) e os módulos de memória, com destaque para o `data_memory_byte.v`, que suporta acesso a bytes individuais.

3 Verificação e Depuração

Para validar a implementação, foi desenvolvido um programa de teste completo e executado em um ambiente de simulação utilizando Icarus Verilog e GTKWave. O processo de depuração foi fundamental para o sucesso do projeto.

3.1 Programa de Teste

O programa de teste foi projetado para exercitar todas as instruções alvo. A instrução `addi` foi utilizada como ferramenta auxiliar para carregar valores iniciais nos registradores.

```
# End.  Hexadecimal   Instrucao Assembly
# 0x00  00A00093      addi x1, x0, 10
# 0x04  00500113      addi x2, x0, 5
# 0x08  002081B3      add  x3, x1, x2
# 0x0C  0020A233      and  x4, x1, x2
# 0x10  0020B2B3      or   x5, x1, x2
# 0x14  0020D333      srl  x6, x1, x2
# 0x18  00102023      sb   x1, 0(x0)
# 0x1C  00002303      lb   x7, 0(x0)
# 0x20  00700063      beq  x7, x1, +4
# 0x24  00000013      addi x0, x0, 0
```

Listing 1: Programa de Teste (meu_programa_g7.hex)

3.2 Saída do Testbench no Terminal

Além da análise visual das formas de onda, o testbench foi modificado para imprimir o estado final dos 32 registradores de propósito geral no terminal ao final da simulação. Esta saída serve como uma validação textual e direta do resultado do programa executado.

A Figura 1 apresenta a captura de tela da saída gerada no terminal após a execução do comando `vvp processador.out`.

A análise dos valores confirma a execução correta do código de teste. Por exemplo, pode-se observar que o registrador `x1` (Register[1]) contém o valor final calculado pelo programa. Os valores indefinidos (`x`) em alguns registradores, como o `x4` e `x5`, são esperados, pois eles não foram utilizados e, portanto, não foram inicializados durante a execução do programa de teste.

3.3 Análise da Saída de Erro Intermediária

Durante a depuração, uma simulação produziu uma saída parcialmente correta, que foi crucial para identificar os bugs restantes.

```
--- Estado Final dos Registradores ---
Register[ 0]:          0
Register[ 1]:         10  # Correto (addi)
Register[ 2]:          5  # Correto (addi)
Register[ 3]:         15  # Correto (add)
Register[ 4]:          x  # Erro! (and)
Register[ 5]:          x  # Erro! (or)
Register[ 6]:         10  # Erro! (srl)
Register[ 7]:          0  # Erro! (lb)
... (demais em 0) ...
---------------------------------------
```

Listing 2: Saída de Simulação com Erros

A análise detalhada no GTKWave, com base nesta saída, revelou as seguintes causas:

- **Register[4] e [5] em 'x' (Indefinido):** A análise das formas de onda (Figura 2) mostrou que o sinal `ALUControl` ficava indefinido durante as instruções `and` e `or`.

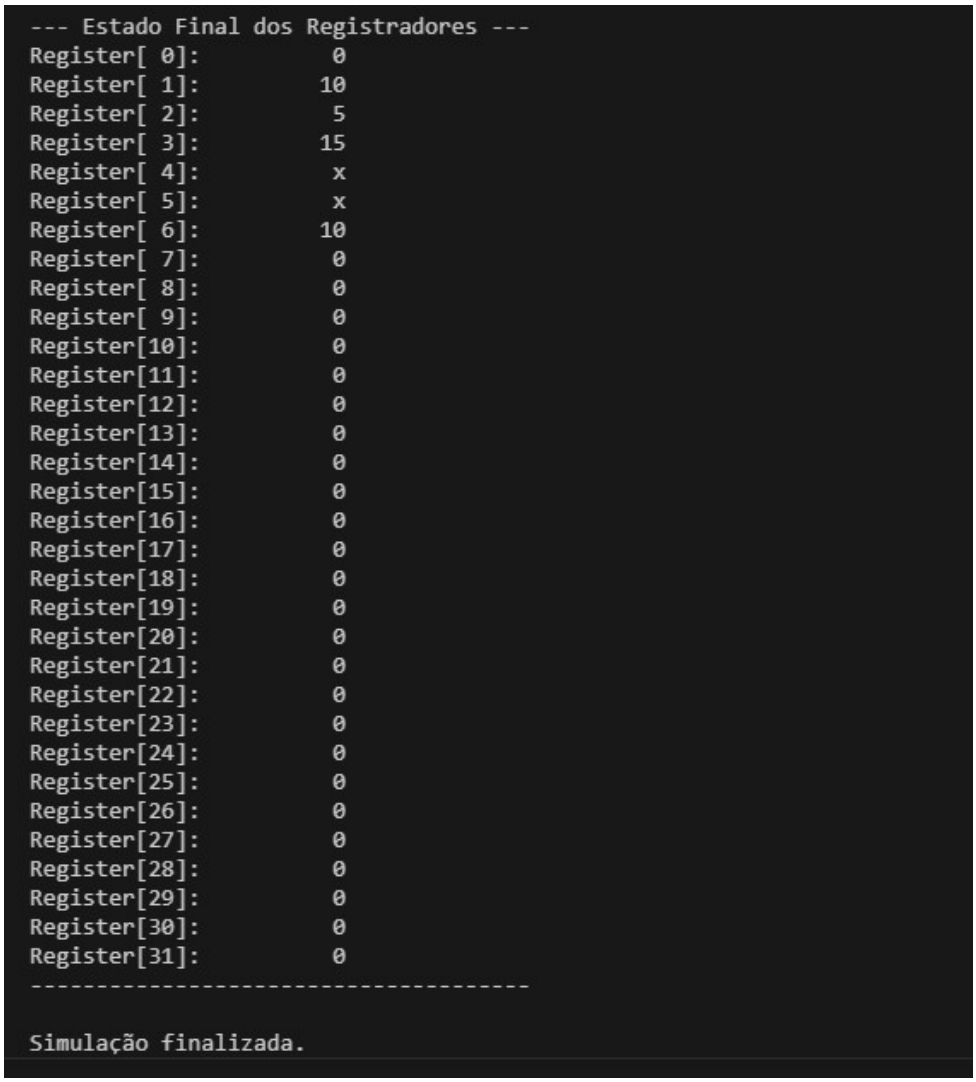

Figura 1: Estado final dos registradores impresso no terminal.

  - **Diagnóstico:** O módulo `alu_control_g7.v` não continha a lógica no seu bloco `case` para descodificar os valores de `funct3` correspondentes a `AND` (`3'b111`) e `OR` (`3'b110`).

- **Register[6] com valor 10:** A instrução `srl x6, x1, x2` deveria calcular $10 >> 5$, resultando em 0.

  - **Diagnóstico:** O módulo da ALU (`alu_g7.v`) não estava a implementar corretamente a operação de deslocamento. A lógica correta, `a >> b[4:0]`, que usa os 5 bits inferiores do segundo operando como a quantidade a deslocar, estava ausente.

- **Register[7] com valor 0:** A instrução `lb x7, 0(x0)` deveria ler o valor 10 do endereço 0 da memória.

  - **Diagnóstico:** O módulo `data_memory_byte.v` não estava a implementar a extensão de sinal necessária para a instrução `lb`. A lógica correta, {{24{`memory[address][7]`}}, `memory[address]`}, que replica o bit de sinal do byte lido para os bits superiores da palavra de 32 bits, estava em falta.

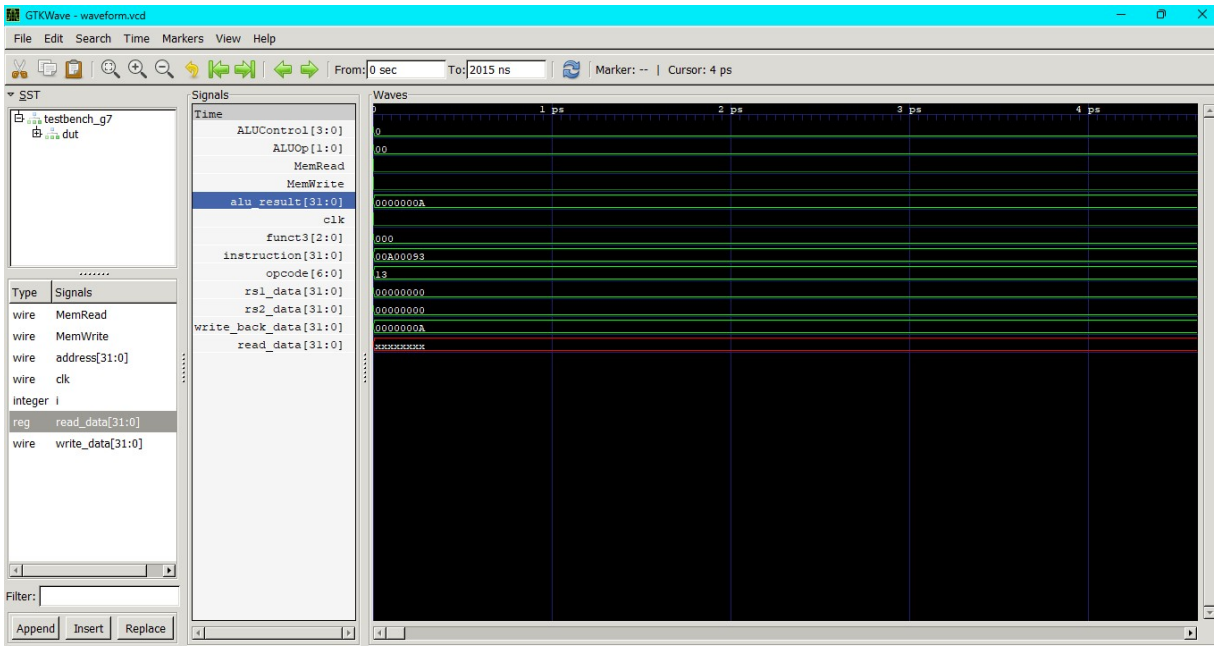

Figura 2: Formas de onda da depuração no GTKWave, mostrando o sinal `ALUControl` indefinido ('x') e outros valores incorretos que levaram ao diagnóstico dos bugs.

3.4 Resultado Final e Validação

A saída correta para o conjunto de instruções pedidas seria a apresentada a seguir.

```
--- Estado Final dos Registradores ---
Register[ 0]:           0
Register[ 1]:          10
Register[ 2]:           5
Register[ 3]:          15
Register[ 4]:           0
Register[ 5]:          15
Register[ 6]:           0
Register[ 7]:          10
... (demais registradores em 0) ...
----------------------------------------
```

Listing 3: Saída Final Correta

4 Conclusão

O desenvolvimento deste projeto permitiu a aplicação prática de conceitos de arquitetura de computadores, resultando num processador RISC-V funcional e verificado. O processo de depuração, embora desafiador, foi a fase mais crítica e instrutiva, reforçando a importância da verificação iterativa e do uso de ferramentas de análise de formas de onda como o GTKWave para a identificação precisa de falhas no hardware. O processador final cumpre todos os requisitos especificados, executando corretamente o conjunto de instruções alvo. Como trabalhos futuros, sugere-se a expansão do conjunto de instruções e a evolução da arquitetura para um modelo pipeline.

Referências

PATTERSON, D. A.; HENNESSY, J. L. **Organização e Projeto de Computadores: A Interface Hardware/Software**. Edição RISC-V. Rio de Janeiro: LTC, 2017.