Projeto RISC-V Multiciclo - OAC

Turma 03

Giovanni Minari Zanetti - 202014280
Mateus Gomes de Araújo - 202014440

Sumário

1 Introdução

O presente trabalho tem como objetivo desenvolver e simular uma arquitetura RISC-V multiciclo em VHDL. Este projeto visa a implementação de um processador que seja capaz de executar um conjunto básico de instruções RISC-V. A simulação será realizada no ModelSim, com memória e registradores organizados conforme os requisitos do processador RISC-V.

2 Descrição do Trabalho

2.1 Instruções Implementadas

As instruções obrigatórias implementadas incluem:

- **Instruções de carregamento e armazenamento de dados:** LW (Load Word) e SW (Store Word);
- **Instruções aritméticas e lógicas:** ADD, ADDI, SUB, AND, OR, XOR, SLT;
- **Instruções de controle de fluxo:** JAL, JALR, BEQ, BNE;
- **Instruções de manipulação de endereços:** AUIPC, LUI.

Além disso, foram implementadas instruções adicionais específicas dos seguintes grupos:

- **Grupo Imm:** ORI, ANDI, XORI;
- **Grupo Comp:** SLTI, SLTU, SLTIU;
- **Grupo Shift:** SLL, SRL, SRA;
- **Grupo Shift Imediato:** SLLI, SRLI, SRAI;
- **Grupo Branch:** BGE, BLT.

2.2 Descrição da Implementação

A arquitetura foi organizada conforme a figura de organização básica do RISC-V multiciclo. O processador é composto por uma unidade de controle e uma unidade operativa. A implementação inclui um banco de registradores, unidade aritmética e lógica (ULA), e uma memória de dados com 4096 palavras de 32 bits. A carga do programa e dos dados foi feita a partir de arquivos de texto, com as instruções sendo carregadas a partir do endereço 0 e os dados a partir do endereço 0x2000.

A fim de simular seu funcionamento, uniu-se os componentes dos trabalhos anteriores por meio do controle do multiciclo, e definiu-se um comportamento baseado em processos para executar a máquina de estados do controle e a sincronia dos componentes. Foram usados 6 estados no total, que funcionam da seguinte forma:

2.2.1 Início

Essa etapa é considerada um pré-estado, pois é executada apena para inicializar a Memória de Dados e Instruções. Um sinal memReady é ativado quando a memória é inicializada corretamente, e assim o multiciclo começa de fato a funcionar, ao passar para o estado Fetch.

2.2.2 Fetch

Na primeira etapa, a instrução é buscada na Memória de Dados e Instruções. O valor atual do contador de programa (PC) é usado como endereço para a leitura da instrução. Os passos são:

- O valor do PC é enviado para a memória;
- A instrução armazenada nesse endereço é carregada no registrador de instrução (IR);
- O PC é incrementado (PC = PC + 4) para apontar para a próxima instrução a ser executada.

2.2.3 Decode

Na segunda etapa, o processador decodifica a instrução para entender o que deve ser executado e quais registradores serão utilizados. Os valores dos registradores necessários são lidos do banco de registradores:

- A instrução no IR é decodificada para determinar o opcode e os campos de registradores (rs1, rs2, rd);
- São lidos os valores dos registradores rs1 e rs2 do banco de registradores e armazenados em registradores internos;
- Se a instrução for do tipo imediata, o valor do imediato é extraído.

2.2.4 Execute

Dependendo do tipo de instrução, diferentes operações ocorrem nessa etapa:

- Instruções aritméticas/lógicas: A ULA realiza a operação (como ADD, SUB, AND, etc.) usando os operandos lidos dos registradores;
- Instruções de carregamento/armazenamento (Load/Store): O endereço de memória é calculado somando o valor do registrador base ao deslocamento imediato;
- Instruções de desvio (Branch): O destino do desvio é calculado com base no deslocamento imediato e no valor do PC.

2.2.5 Memory

Para instruções de leitura ou escrita na memória (LW e SW), essa etapa acessa a memória de dados:

- Leitura (Load): O processador usa o endereço calculado na etapa anterior para ler o valor da memória e armazená-lo temporariamente;
- Escrita (Store): O valor do registrador de origem é escrito no endereço de memória calculado.

2.2.6 WriteBack

Na última etapa, o resultado da operação é escrito de volta no banco de registradores:

- Instruções aritméticas/lógicas: O resultado da operação da ULA é escrito no registrador de destino (rd);
- Instruções de Load: O valor lido da memória é escrito no registrador de destino (rd).

2.3 Dificuldades Encontradas

As principais dificuldades encontradas durante a implementação foram:

- **Integração dos módulos:** A interligação dos diversos componentes apresentou desafios devido ao número elevado de sinais e à necessidade de sincronização;
- **Incremento do PC:** Foi necessário usar um registrador que funciona como buffer do PC para poder incrementá-lo corretamente e evitar alterar o RI de forma incorreta;
- **Implementação de algumas instruções adicionais:** Instruções como BGEU e BLTU, que envolvem branches com rs2 sem sinal, foram desafiadoras.

3 Testes Realizados

A fim de testar o funcionamento do RISC-V Multiciclo, o professor disponibilizou um arquivo .asm com diversas instruções. Por meio do RARS, exportou-se a parte de instruções e dados para arquivos .txt em formato "hexadecimal text".

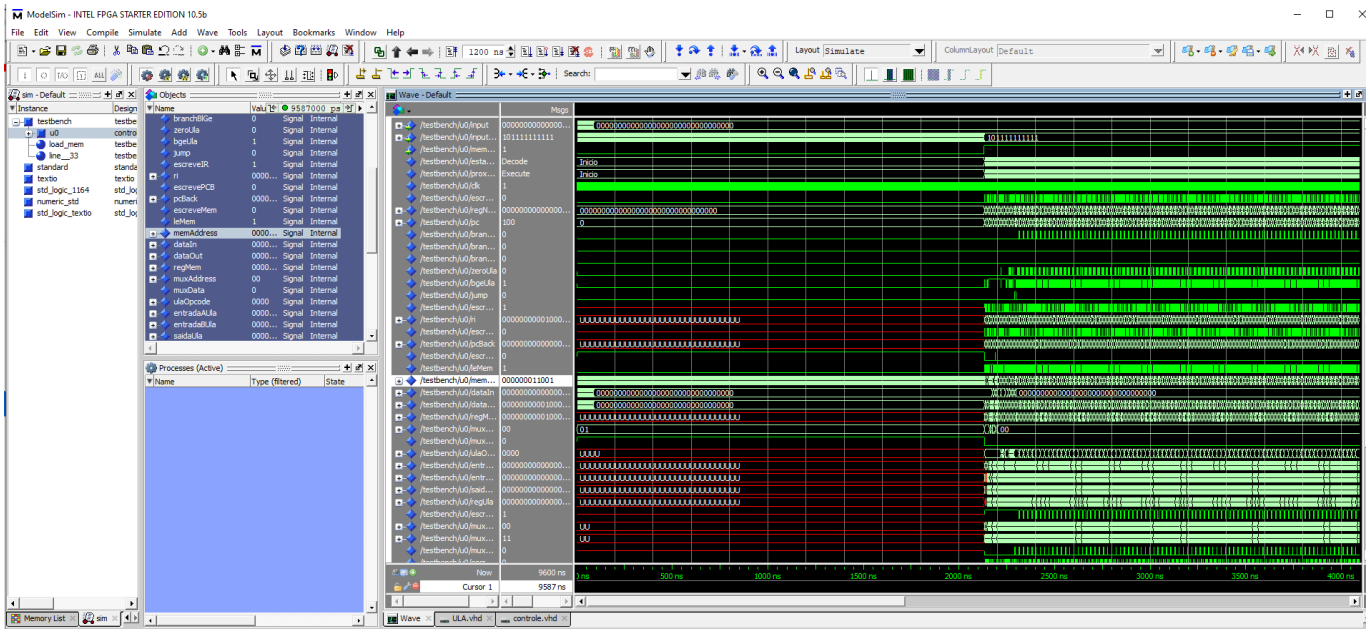

Figura 1: Execução do programa.

A figura 1 mostra a execução do programa fornecido pelo professor. As formas de onda representam o comportamento dos sinais ao longo do tempo. É possível observar nitidamente a fase de inicialização da Memória de Dados e Instruções e a execução do programa.

4 Conclusão

O projeto do processador RISC-V multiciclo em VHDL foi concluído com sucesso, implementando um conjunto básico de instruções, bem como instruções adicionais. O processador foi testado e simulado corretamente, executando os programas gerados a partir do RARS. O principal desafio foi a integração dos módulos e o incremento do PC, mas esses obstáculos foram superados ao longo do desenvolvimento.