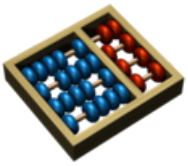
Universidade Estadual de Campinas
Instituto de Computação
MO601 – Arquitetura de Computadores II
Prof. Rodolfo Jardim de Azevedo



Projeto 2

Um simulador simples do processador RISC-V

Rubens de Castro Pereira
RA: 217146

Campinas - SP
3 de maio de 2023

Índice

1 Introdução **2**

2 Descrição geral **2**

3 Ambiente de Desenvolvimento **2**

4 Algoritmo de Simulação **3**

5 Testes **4**

6 Considerações sobre o aprendizado nesse projeto **4**

1 Introdução

Este relatório apresenta o projeto 2 da disciplina Arquitetura de Computadores II (MO601) ministrada pelo Prof. Rodoldo Jardim de Azevedo, cujo propósito é implementar um simulador básico do processador RISC-V RV32IM considerando o conjunto das instruçãos básicas de inteiros de 32 bits e, adicionalmente, as instruções de multiplicação e divisão inteiras de 32 bits. A seção 2 apresenta a descrição geral do projeto, a seção 3 relaciona as tecnologias utilizadas no ambiente de desenvolvimento, a seção 4 apresenta o algoritmo de simulação do processador, a seção 5 detalha os testes realizados e a seção 6 apresenta as considerações sobre o aprendizado neste projeto.

2 Descrição geral

O objetivo deste projeto é implementar um simulador do processador RISC-V RV32IM que executa o conjunto das instruçãos básicas de inteiros, as instruções de multiplicação e de divisão inteiras, em uma arquitetura de 32 bits.

O simulador implementa um modelo de processador contendo as estruturas de dados básicas com destaque à memória RAM com tamanho suficiente para acomodar as instruções e dados, 32 registradores e o contador de programa (*Program Counter - PC*).

Os programas codificados na linguagem C disponibilizados no repositório ACStone [1] representam a entrada para o simulador, sendo que estes possuem funcionalidades e utilidades reduzidas, contudo são excelentes artefatos para uso em simuladores de processadores. O repositório contém 77 programas fonte C de testes os quais contemplam todas as instruções implementadas neste simulador. O conjunto de instruções RISC-V é composto por 47 instruções básicas (*RV32I Base Instruction Set*) e 8 instruções de multplicaçao e divisão (*RV32M Standard Extension*) totalizando 55 instruções.

Os programas fonte foram compilados utilizando-se a arquitetura 32 bits, em seguida montados em formato assembler (*linked*) e utilizados como entrada no simulador do processador RISC-V.

Cada programa assembler foi processado separadamente produzindo um arquivo de saída com o log da execução conforme formato especificado na definição desse projeto (padrão ABI). Cada arquivo de log evidencia a simulação do respectivo programa fonte/assembler.

Ao final da simulação, um arquivo contendo as estatísticas é gerado juntamente com os arquivos de log dos programas contendo dados sobre a quantidade de instruções e o número de ciclos de execução de cada programa simulado.

3 Ambiente de Desenvolvimento

O ambiente de desenvolvimento do simulador está estruturado com as seguintes tecnologias:

- Sistema operacional: Windows 10
- Ambiente integrado de Desenvolvimento (IDE): Visual Studio Code (VS Code) version 1.76.2

[1] *https://github.com/rjazevedo/ACStone*

- Linguagem de programação: Python versão 3.9.5
- Pacotes Python: OS
- Ferramentas para compilação e montagem dos programas fonte C - *Embecosm - Tool Chain Downloads*
  - Web site: https://www.embecosm.com/resources/tool-chain-downloads/
  - Pacote selecionado para instalação do compilador RISC-V 32 bits:
    * Windows 10 - 32-bits GCC (.zip)
    * https://buildbot.embecosm.com/job/riscv32-gcc-win64/169/artifact/riscv32-embecosm-gcc-win64-20230402.zip

4 Algoritmo de Simulação

O algoritmo do simulador do processador RISC-V RV32IM é apresentado em *algorithm 1*.

Algorithm 1: Algoritmo do simulador do processador RISC-V RV32IM.

Entrada: *programa assembler (exemplo: 000.main.asm)*

Saída: *arquivo de saída (log) do programa simulado (exemplo: 000.main.log)*

```
1  ExecutarPrograma(programa assembler, PC) begin
2      while não alcançar a instrução ebreak do
3          ler instrução indicada pelo PC
4          decodificar instrução conforme o tipo específico (R/I/S/B/U/J)
5          executar a instrução de acordo com sua implementação
6          gerar linha de log de saída da instrução executada
7      end
8  end
9  /* Programa principal que gerencia a simulação do processador RISC-V
      RV32IM na execução de programas.  */
10 foreach programa assembler do
11     inicializar processador RISC-V criando memória RAM, registradores e PC
12     ler programa assembler
13     armazenar programa lido na memória RAM
14     inicializar contador do programa (PC) com o endereço de memória da primeira
         instrução que será executada
15     ExecutarPrograma(programa assembler, PC)
16     gravar arquivo com o log das instruções executadas
17     gravar estatística da simulação
18 end
```

5 Testes

A execução dos testes compreende duas etapas: 1) geração dos programas em formato assembler (*linked*); e 2) simulação da execução dos programas assembler de acordo com o processador RISC-V RV32IM.

A etapa 1 é realizada a partir dos programas previamente compilados e fornecidos pelo professor (extensão "riscv"), localizados em *https://drive.google.com/drive/u/2/folders/1ei8E-qk2dwQvCWJv8ovGJiIdLjw7yVfP*. Com isso, a etapa de compilação foi previamente realizada bastando criar os programas assembler por meio da execução do arquivo em lote *dump-programas-compilados.bat*. Este *script* executa o utilitário *riscv32-unknown-elf-objdump.exe* gerando os programas assembler (extensão "asm") a partir dos programas compilados.

A etapa 2 realiza a simulação do processador RISC-V executando cada programa assembler e gerando um arquivo de log (extensão "log") da simulação a fim de evidenciar a execução.

Os arquivos foram organizados em uma pasta de testes específica denominada *test* localizada dentro da pasta do projeto *RCP-MO601-Project-02*. A pasta *test* contém todos os programas fonte em C (extensão "c"), os programas compilados previamente e os programas assembler gerados na etapa 1. Para facilitar a visualização do resultado do simulador, foi criada a pasta *log* dentro da pasta *test* para acomodar os arquivos de saída da simulação.

Ao final da simulação, as estatísticas da simulação estarão disponíveis no arquivo *_riscv_simulator_statistic.txt* encontrado na pasta *log*.

6 Considerações sobre o aprendizado nesse projeto

Neste projeto foi possível compreender de maneira detalhada como os processadores executam os programas desenvolvidos em linguagem de alto nível a partir das instruções geradas pelo compilador. Quando se cria software e sistemas de informação, essa camada do software localizada muito próxima do hardware se torna completamente oculta aos desenvolvedores e é exatamente nesse contexto que o simulador do processador atua, decodificando e executando cada instrução oriunda de processos anteriores de compilação e montagem (*linked*).

Conhecer em detalhes uma Arquitetura do Conjunto de Instruções (*Instruction Set Architecture - ISA*) RISC-V foi bem interessante, demonstrando o quão simples pode ser uma instrução executada por um processador e ao mesmo tempo poderosa, dando suporte às linguagens de programação de alto nível.

As convenções utilizadas na especificação das instruções RISC-V foi provavelmente o maior desafio encontrado no desenvolvimento deste projeto. A documentação da implementação de determinadas instruções presume conhecimentos prévios e, por isso, a dificuldade no entendimento.

Por fim, destaco o aprendizado relativo à arquitetura do conjunto de instruções do processador RISC-V associado aos conteúdos da disciplina apresentados durante as aulas. Além disso, o objetivo proposto pelo projeto foi alcançado tendo em vista as evidências da execução dos programas assembler pelo simulador.

Referências

Andrew Waterman1, and Krste Asanovi, The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2, 2017. https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf.

Msyksphinz. RISC-V Instruction Set Specifications, 2019. Acessado em abril de 2023. Disponível em https://msyksphinz-self.github.io/riscv-isadoc/html/rvi.html.

RISC-V Instruction Encoder/Decoder. Acessado em abril de 2023. Disponível em https://luplab.gitlab.io/rvcodecjs