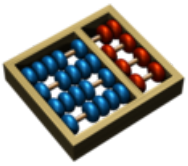
Universidade Estadual de Campinas
Instituto de Computação
MO601 – Arquitetura de Computadores II
Prof. Rodolfo Jardim de Azevedo



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

Este relatório apresenta o resultado do 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 a forma como os testes foram realizados e a seção 6 apresenta as considerações sobre o aprendizado nesse projeto.

2 Descrição geral

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

O projeto utiliza como entrada os programas codificados na linguagem C disponibilizados no repositório ACStone (*https://github.com/rjazevedo/ACStone*), que possuem funcionalidades e utilidades reduzidas, contudo são excelentes ferramentas para o uso em simuladores de instruções do processador RISCV. Nesse repositório foram disponibilizados 77 programas C os quais contemplam o total de instruções utilizadas neste simulador. O conjunto de instruções RISC-V possui 47 instruções básicas (*RV31I Base Instruction Set*) e 8 instruções de multplicaçao e divisão (*RV32M Standard Extension*), totalizando 55 instruções.

Os programas fonte C foram compilados utilizando-se a arquitetura 32 bits (RV32IM), em seguida montados em formato assembler (*linked*) e utilizados como entrada no simulador do processador RISCV.

O simulador implementa um modelo de processador contendo as estruturas de dados básicascom destaque à memória RAM com tamanho suficiente para acomodar as instruções geradas dos programas compilados e as estuturas de dados maipuladas por esses programas, 32 registradores e contador de programa (PC - *Programa Counter*). Cada programa foi processado separadamente e um arquivo de log foi gerado conforme o formato especicado na definição desse projeto e o padrão ABI, sendo que cada arquivo de saída representa a evidencia da simulação de um programa fonte.

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
- Linguagem de programação: Python versão 3.9.5
- Pacotes Python: OS

- Ferramenta para compilação dos programas C: *Embecosm - Tool Chain Downloads*
  - https://www.embecosm.com/resources/tool-chain-downloads/
  - Pacote selecionado para instalação do compilador RISC-V 32 bits: Windows 10 - 32-bit 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 em formato assembler (linked - exemplo: 000.main.asm)*

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

```
1  ExecutarPrograma(programa, 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 do
11     inicializar processador RISC-V criando memória RAM, registradores e PC
12     ler programa
13     armazenar programa na memória RAM
14     inicializar contador do programa (PC) com o endereço de memória da primeira
         instrução que será executada
15     ExecutarPrograma(programa, PC)
16     gravar arquivo com o log das instruções executadas
17 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 conforme o processador RISCV.

A etapa 1 é realizada com a execução do arquivo em lote *dump-programas-compilados.bat* que utiliza o programa *riscv32-unknown-elf-objdump.exe* para a criação dos programas em assembler (extensão ”asm”) a partir dos programas compilados (extensão ”riscv”).

A etapa 2 realiza a simulação do processador RISCV executando os programas em assembler e gerando como saída um arquivo com extensão ”log”para cada programa simulado, evidenciando sua execução.

Os arquivos foram organizados em uma pasta de testes específicos denominada *test* e 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 fornecidos pelo professor e os programas em assembler gerados na etapa1. Para facilitar a visualização do resultado do simulador, foi criada a pasta *log* localizada dentro da pasta *test* a fim de acomodar os arquivos de saída da simulação.

A listagem abaixo apresenta os programas simulados com sucesso:

1. 000.main.c	14. 025.cast.c	27. 053.mul.c	40. 072.bool.c	53. 111.if.c
2. 011.const.c	15. 026.cast.c	28. 054.mul.c	41. 073.bool.c	54. 112.if.c
3. 012.const.c	16. 027.cast.c	29. 055.mul.c	42. 074.bool.c	
4. 013.const.c	17. 031.add.c	30. 056.mul.c	43. 075.bool.c	55. 113.if.c
5. 014.const.c	18. 032.add.c	31. 057.mul.c	44. 076.bool.c	56. 114.if.c
6. 015.const.c	19. 033.add.c	32. 058.mul.c	45. 077.bool.c	
7. 016.const.c	20. 034.add.c	33. 061.div.c	46. 078.bool.c	57. 115.if.c
8. 017.const.c	21. 041.sub.c	34. 062.div.c	47. 081.shift.c	58. 116.if.c
9. 018.const.c	22. 042.sub.c	35. 063.div.c	48. 082.shift.c	
10. 021.cast.c	23. 043.sub.c	36. 064.div.c	49. 083.shift.c	59. 117.if.c
11. 022.cast.c	24. 044.sub.c	37. 065.div.c	50. 084.shift.c	60. 118.if.c
12. 023.cast.c	25. 051.mul.c	38. 066.div.c	51. 085.shift.c	
13. 024.cast.c	26. 052.mul.c	39. 071.bool.c	52. 086.shift.c	61. 119.if.c

Os programas relacionados a seguir tiveram problemas durante a simulação e, portanto, não foi possível gerar os arquivos de saída (log):

1. 121.loop.c	5. 125.loop.c	9. 133.call.c	13. 143.array.c
2. 122.loop.c	6. 126.loop.c	10. 134.call.c	14. 144.array.c
3. 123.loop.c	7. 131.call.c	11. 141.array.c	15. 145.array.c
4. 124.loop.c	8. 132.call.c	12. 142.array.c	16. 146.array.c

6 Considerações sobre o aprendizado nesse projeto

Neste projeto foi possível comprender em detalhes 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 muito próxima do hardware se torna completamente oculta aos desenvolvedores de sistemas de informação e software e é exatamente nesse contexto que o simulador do processador atua, avaliando (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.

Os resultados alcançados por esse trabalho foram

———————————— ——————-

O assunto tratado nesse projeto é muito diferente dos temas que venho trabalhando nas últimas três décadas de desenvolvimento de sistemas de informação. Contudo, desafios são excelentes oportunidades de aprendizado, fixação de novos conceitos e exploração de tecnologias e ferramentas.

Neste projeto foi possível apreender conhecimentos principalmente relacionados à área de arquitetura de computadores com destaque ao conceito de atraso em circuitos lógicos e funcionamento das portas lógicas, elementos muitos simples, mas combinadas entre si podem produzir resultados interessantes e importantes. A compreensão do conceito de atraso no circuito lógico aplicado às portas lógicas ao mesmo tempo que os sinais de entrada das variáveis são inseridos, representou o ponto mais relevante deste trabalho. Embora houveram dificuldades no início, mas após estudos e diálogos com o professor, foi possível compreender a ideia do atraso e, sabe-se que quando entendemos a definição de um problema, boa parte de sua resolução está encaminhada.

Outro ponto de destaque é o uso de novas ferramentas no desenovlvimento e entrega do trabalho combinando os produtos de GitHub e Docker, atualmente ferramentas de ampla utilização não somente no meio acadêmico mas no âmbito organizacional e empresarial.

Ressalto dois pontos importantes para o sucesso do trabalho: 1) a especificação do projeto está muito bem detalhada permitindo o pleno entendimento do trabalho a ser desenvolvido e das entregas esperadas; e 2) elevada disponibilidade do professor no atendimento aos alunos.

Por fim, destaco que os resultados alcançados nesse projeto são fruto da combinação de uma completa especificação do projeto, boa disponibilidade do professor em esclarecer e colaborar com o estudante, elevada dedicação no desenvolvimento do projeto conforme especificado e foco na entrega do produto dentro do prazo esperado.

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. https://msyksphinz-self.github.io/riscv-isadoc/html/rvi.html.