Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

RISC-V: Ambiente de montagem e simulação

Matheus Y. Matsumoto

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Ricardo Pezzuol Jacobi

Brasília
2017

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

RISC-V: Ambiente de montagem e simulação

Matheus Y. Matsumoto

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Ricardo Pezzuol Jacobi (Orientador)
CIC/UnB

Prof. Dr. Donald Knuth
Stanford University

Dr. Leslie Lamport
Microsoft Research

Prof. Dr. Ricardo Pezzuol Jacobi
Coordenador do Curso de Engenharia da Computação

Brasília, 26 de março de 2017

Dedicatória

Na *dedicatória* o autor presta homenagem a alguma pessoa (ou grupo de pessoas) que têm significado especial na vida pessoal ou profissional. Por exemplo (e citando o poeta): *Eu dedico essa música a primeira garota que tá sentada ali na fila. Brigado!*

Agradecimentos

Nos *agradecimentos*, o autor se dirige a pessoas ou instituições que contribuíram para elaboração do trabalho apresentado. Por exemplo: *Agradeço aos gigantes cujos ombros me permitiram enxergar mais longe. E a Google e Wikipédia.*

Resumo

O *resumo* é um texto inaugural para quem quer conhecer o trabalho, deve conter uma breve descrição de todo o trabalho (apenas um parágrafo). Portanto, só deve ser escrito após o texto estar pronto. Não é uma coletânea de frases recortadas do trabalho, mas uma apresentação concisa dos pontos relevantes, de modo que o leitor tenha uma ideia completa do que lhe espera. Uma sugestão é que seja composto por quatro pontos: 1) o que está sendo proposto, 2) qual o mérito da proposta, 3) como a proposta foi avaliada/validada, 4) quais as possibilidades para trabalhos futuros. É seguido de (geralmente) três palavras-chave que devem indicar claramente a que se refere o seu trabalho. Por exemplo: *Este trabalho apresenta informações úteis a produção de trabalhos científicos para descrever e exemplificar como utililzar a classe LaTeX do Departamento de Ciência da Computação da Universidade de Brasília para gerar documentos. A classe `UnB-CIC` define um padrão de formato para textos do CIC, facilitando a geração de textos e permitindo que os autores foquem apenas no conteúdo. O formato foi aprovado pelos professores do Departamento e utilizado para gerar este documento. Melhorias futuras incluem manutenção contínua da classe e aprimoramento do texto explicativo.*

Palavras-chave: risc,LaTeX, metodologia científica, trabalho de conclusão de curso

Abstract

O *abstract* é o resumo feito na língua Inglesa. Embora o conteúdo apresentado deva ser o mesmo, este texto não deve ser a tradução literal de cada palavra ou frase do resumo, muito menos feito em um tradutor automático. É uma língua diferente e o texto deveria ser escrito de acordo com suas nuances (aproveite para ler `http://dx.doi.org/10.6061%2Fclinics%2F2014(03)01`). Por exemplo: *This work presents useful information on how to create a scientific text to describe and provide examples of how to use the Computer Science Department's LaTeX class. The `UnB-CIC` class defines a standard format for texts, simplifying the process of generating CIC documents and enabling authors to focus only on content. The standard was approved by the Department's professors and used to create this document. Future work includes continued support for the class and improvements on the explanatory text.*

Keywords: LaTeX, scientific method, thesis

Sumário

Lista de Figuras

Capítulo 1

Introdução

Este projeto é um estudo da arquitetura RISC-V, e implementações de seus conceitos em ferramentas auxiliares. Estas ferramentas serão utilizadas para pesquisa, ensino, e aplicações de conceito estudados na área de arquitetura de computadoers utilizando a arquitetura mencionada.

RISC-V é uma arquitetura de conjunto de instruções aberta, criada na Universidade da Califórnia, em Berkeley. Originalmente foi pensada para ser utilizada na pesquisa e ensino da área de arquitetura de computadores, mas está se tornando um padrão de arquitetura aberta para a indústria. [1]

Seu nome é pronunciado na língua inglesa como *"risc five"*. O motivo de ser *"five"* é devido ao fato de que é o quinto maior projeto de uma ISA RISC desenvolvida na UC Berkeley. As primeiras foram RISC-I, RISC-II, SOAR, e SPUR. O numeral romano "V"de RISC-V também funciona com significado de *"variations"* e *"vectors"*.

Ao iniciar o projeto haviam poucas ferramentas relacionada a arquitetura RISC-V, e as ferramentas que haviam eram de difícil instalação, e configuração. Para este projeto foi implementado um montador, e um simulador para a ISA base RV32I. Este módulo base da arquitetura cobre os pontos principais de uma arquitetura funcional.

O sistema foi desenvolvido com intuito de ser multiplataforma, extensível, fácil utilização, e também para que outros possam facilmente continuar a evolução do projeto. Utilizar a plataforma Web para a disponibilização facilita a utilização imediata em qualquer dispositivo com um Browser e acesso a internet, sem a necessidade de instalações e configurações iniciais que podem trazer dificuldades e desviar a atenção principal que é o estudo da arquitetura RISC-V.

Estudar arquitetura de computadores é uma tarefa difícil, quanto mais fáceis e didáticas forem as ferramentas utilizadas, melhor para o aprendizado. Um grande ecossistema de uma solução pode fazer uma tecnologia evoluir muito mais rápido, por isso é muito

importante desenvolver ferramentas que auxiliam a aprendizagem e utilização dessa nova arquitetura.

1.1 Os Processadores: MIPS, ARM e outros antes do RISC-V

As ISAs MIPS e ARM tiveram grande influência na arquitetura RISC-V, porém existem alguns detalhes técnicos que desfavorecem o uso destas arquiteturas ao invés de criar a nova RISC-V, no ponto de vista dos autores do RISC-V [2],

O MIPS é uma ISA criada no começo dos anos 80, em Stanford, seu design utiliza a filosofia RISC, e facilita implementação de pipelines. MIPS foi implementado comercialmente pela primeira vez no processador R2000, em 1986.

Algumas desvantagens que desencorajam o uso do MIPS principalmente para implementações de alta performace:

- A ISA é exageradamente otimizada para o padrão de pipeline de cinco estágios em ordem, como jumps e branches são atrasados, isso complica implementações superescalares e super-pipelines. Sendo que o recurso de branch delay slot, não pode ser retirada por questões de compatibilidade.
- A ISA provê um pobre suporte para código de posições independentes. A revisão de 2014 melhorou endereçamento relativo ao contador de programa, porém as chamadas ainda necessitam geralmente de mais de uma instrução.
- Imediatos de 16 bits consomem muito espaço de codificação de instruções, deixando pouco espaço para futuras extensões da ISA, ou trabalhar com instruções comprimidas.
- Multiplicações e divisões utilizam recursos especiais de arquitetura.

Além dessas e algumas outras questões técnicas, MIPS não pode ser utilizado em várias situações pelo fato de ser proprietária. Históricamente, a patente da MIPS Technologies sobre instruções de *load* e *store* desalinhados, preveniu terceiros de implementar totalmente sua ISA.

Outra arquitetura popular que teve influência no MIPS é a arquitetura ARM, mais especificamente as arquiteturas ARMv7 e ARMv8. Estas arquitetura são baseadas na filosofia RISC, e são de longe as mais utilizadas no mundo. A arquitetura é desenvolvida hoje pela empresa britânica ARM Holdings. Inicialmente criada pela Acorn Computers Limited de Cambridge, Inglaterra, entre 1983 e 1985, baseado no processador RISC-I da Berkeley.

A utilização do ARM foi desconsiderada principalmente pelos seguintes motivos,

- Quando o projeto foi iniciado, a arquitetura estava na versão quatro. Nesta versão ainda não havia suporte para 64 bits.
- A ISA possui embutida uma ISA para instruções comprimidas e uma ISA de tamanho variável, porém são codificadas de forma diferente da ISA comum 32 bits, portanto os decodificadores são ineficientes, energeticamente, temporalmente e em relação a custo.
- A ISA tem muitos recursos que complicam implementações. Por exemplo o contador de programa é um dos registradores endereçáveis, e o bit menos significativo do contador seleciona qual ISA está executando (ARM tradicional ou de instruções comprimidas) e assim a instrução ADD, por exemplo, consegue modificar o valor do contador de programa.
- Mesmo que fosse possível implementar a arquitetura ARM legalmente, a quantidade de instruções é gigantesca e seria técnicamente bem desafiador.

Entre outros motivos, estes são alguns dos principais que levaram os autores do RISC-V desenvolverem sua própria ISA e não utilizar algumas já consolidadas para seus projetos.

Além dessas, existem outras que foram consideradas, como *SPARC*, arquitetura open-source desenvolvida pela Sun Microsystems, *Alpha*, desenvolvida pela Digital Equipment Corporation, *OpenRISC*, evolução da arquitetura educacional open-source *DLX* desenvolvida pelo Patterson e Henessy.

1.2 Importância acadêmica e industrial do RISC-V

RISC-V foi desenvolvido por Krste Asanoviü, Andrew Waterman, e Yunsup Lee na Universidade da Califórnia, Berkeley, com colaboração de David Patterson. Eles precisavam de uma ISA que pudessem estudar, implementar com liberdade para fins acadêmicos, porém como visto na seção anterior, nenhuma se encaixava perfeitamente no que eles precisavam. Na maioria das vezes, muito complexas, ou muito específicas, ou simplesmente fechadas.

Com essa necessidade surgiu o RISC-V, uma arquitetura de conjunto de instruções de propósito geral, open-source, modular, com a ambição de ser um conjunto universal, livre e grátis para utilização em um amplo espectro de problemas, desde soluções embarcadas, até soluções de alta performance, e aprendizagem de máquina por exemplo.

Com grandes empresas colaboradoras como Google, Nvidia, Western Digital, Oracle, e também as companhias de chip IBM, AMD, e Qualcomm, a ISA tem ganhado espaço nas áreas comerciais também.

A Western Digital, por exemplo, anunciou em um Workshop que irá liderar a industria na troca por mais ISAs RISC-V utilizando um bilhão de núcleos RISC-V em seus dispositivos [3]. Em outro workshop, NVIDIA apresentou por que e como irá implementar novos núcleos para seus micro-controladores utilizando o RISC-V. [4]

Os criadores do RISC-V fundaram uma empresa chamada SiFive, na qual eles vendem kits de desenvolvimento estilo arduino com processadores RISC-V, ou então kits mais avançados capazes de rodar linux, como rasperry pi, entre outras coisas.

Os ataques Spectre e Meltdown que exploram vulnerabilidades na arquitetura de processadores modernos, mostram uma grande importância tanto acadêmica e industrial. Pois a mitigação desses ataques não podem ser estudados e resolvidos com facilidade em arquiteturas fechadas como as da Intel ou ARM. Porém utilizando ISAs open-source podemos estudar melhores alternativas de como resolver esse tipo de problemas arquiteturais com muito mais facilidade e rapidez.

1.3 Ambiente de desenvolvimento

Neste projeto foi desenvolvido um ambiente de desenvolvimento para a arquitetura RISC-V. O ambiente inclui um editor de texto para a linguagem assembly, um montador e um simulador. A implementação foi realizada na plataforma web, maiores detalhes serão apresentados no capítulo 3.

Este conjunto de ferramentas auxiliam o processo de aprendizagem da arquitetura. O fato da plataforma ser acessível por qualquer navegador torna a imersão inicial mais eficaz, pois a pessoa interessada não precisará gastar tempo e esforço com problemas que podem ocorrer na instalação ou configuração da ferramenta.

1.4 Explicação dos capítulos

Este primeiro capítulo apresenta o contexto, motivação, objetivo do projeto que foi realizado.

No capítulo 2, apresentaremos a fundamentação teórica para o desenvolvimento do projeto, incluindo maiores detalhes técnicos da arquitetura e conhecimentos necessários para entender o desenvolvimento do ambiente proposto.

No capítulo 3 mostraremos a implementação, arquitetura de software, decisões de projeto, e as aplicações da fundamentação teórica no projeto.

No quarto capítulo apresentaremos o resultados que obtivemos, exemplos de utilização com uma sequência lógica.

Quinto e último capítulo descreve objetivos atingidos, pontos positivos e negativos, dificuldades e possíveis melhorias junto com idéias de implementações futuras para melhorar o projeto.

Capítulo 2

Fundamentação teórica

Este capítulo aborda a base teórica necessária para o desenvolvimento do projeto.

2.1 Arquitetura RISC-V: ISA

Sua arquitetura obedece aos padrões RISC (Reduced Instruction Set Computing), tendo instruções simples e completas. Foi projetada para ser rápida, ocupar pouco espaço físico, ter baixo consumo de energia, ser extensível, e compatível com entre suas versões. Por ser reduzida, se encaixa perfeitamente para fins acadêmicos, e pesquisas.

Outra característica importante é a sua extensibilidade. Por padrão sua base é inteira, para arquiteturas de 32, 64 e 128 bits, porém existem módulos de extensão. Para descrever quais implementações são utilizadas utilizam-se as nomenclaturas RV32I, RV64I e RV128I, para as implementações padrões RISC-V 32 bits, 64, e 128.

Existem módulos padrões e não-padrões de extensões [1]:

- Os módulos padrões são aqueles que não possuem conflitos entre si e é utizado para propósitos gerais.
- Os módulos não-padrões são módulos especializados, podendo conflitar com outros módulos. A previsão é de que no futuro hajam muitos módulos desse tipo.

As padrões desenvolvidas atualmente adicionam as letras "MAFDQLCBJTPVN", sendo que cada letra representa uma extensão

- M: Multiply/Divide
- A: Atomic
- F: Single-Precision Floating-Point
- D: Double-Precision Floating-Point

- Q: Quad-Precision Floating-Point
- L: Decimal Floating-Point
- C: 16-bit Compressed Instructions
- B: Bit Manipulation
- J: Dynamic Languages
- T: Transactional Memory
- P: Packed-SIMD Extensions
- V: Vector Extensions
- N: User-Level Interrupts

As extensões IMAFD são chamadas de extensões de propósito geral e são abreviadas por G, por exemplo, uma arquitetura de 32 bits que utilizam todas as extensões de propósito geral é chamada de RV32G.

Ainda existe uma outra variação que é a extensão "E", que se difere das outras pois é projetada para sistemas embarcados. Este módulo diminui a quantidade de registradores para 16 e o tamanho também é reduzido para 16 bits.

2.1.1 Objetivos

Seus projetistas sempre são perguntados o motivo ao qual eles quiseram desenvolver uma nova ISA. Alguns dos motivos para o qual usar uma ISA comercial são a existência de suporte de um ecossistema de software, tanto ferramentas de desenvolvimento, portabilidade e ferramentas educacionais, outros benefícios seriam a grande quantidade de documentação, tutoriais e exemplos para o desenvolvimento.

Porém estas vantagens são pequenas na prática, e listam várias disvantagens ao utilizar ISAs comerciais,

- ISAs comerciais são proprietárias
- ISAs comerciais são populares somente em alguns nichos do mercado
- ISAs comerciais vem e vão
- ISAs populares são complexas
- ISAs comerciais dependem de outros fatores para trazer aplicações
- ISAs comerciais populares não são projetadas para extensibilidade

- Uma ISA comercial modificada é uma nova ISA

A posição dos projetistas é que, em um sistema computacional, a ISA talvez seja a interface mais importante, e não existe razão pra que esta seja proprietária. [2]

- eficiente energetica
- compatibilidade
- simples
- escalavel
- modular

2.1.2 História

A ISA RISC-V foi originalmente desenvolvida na Universidade da Califórnia, Berkeley, na Divisão de Ciência da computação, no departamento de Engenharia Elétrica e Ciência da Computação. Baseada na experiência com projetos passados de seus projetistas, a definição da ISA foi iniciada no verão de 2010.

Sua ISA e conjunto de instruções são contruções que usam como tijolos vários projetos

Os primeiros processadores RISC-V fabricados foram escritos em Verilog e manufaturados em tecnologia de pré-produção de 28 nm FD-SOI (*Fully Depleted Silicon On Insulator*) da companhia STMicroeletronics com o nome *Raven-1*

2.1.3 RISC-V Foundation

A *RISC-V Foundation* é uma organização sem fins lucrativos, criada para direcionar futuro desenvolvimento e incentivar a utilização da ISA RISC-V. [5]

O presidente do conselho atualmente é Krste Asanovic, professor do departamento de Engenharia elétrica e ciência da computação na Univerisdade da Califórnia em Berkeley. Também co-fundado da empresa SiFive Inc., a qual incentiva do uso comercial de processadores RISC-V.

E o vice-presidente é o professor David Patterson, muito conhecido pelo livro *Computer Architecture: A Quantitative Approach*, que escreveu juntamente com John Hennessy, e suas pesquisas relacionadas a RISC, RAID, e Redes de estações de trabalho.

Outros membros incluem:

- Zvonimir Bandic, pesquisador e diretor da Western Digital Corporation.
- Charlie Hauck, CEO da Bluespec Inc.

- Frans Sijstermans, vice presidente de engenharia da NVIDIA.
- Ted Speers, chefe de arquitetura de produtos e planejamento do grupo SoC da Microsemi.
- Rob Oshana, gerente de desenvolvimento de software e negócios em segurança na NXP Semiconductors.

e também, Sue Leininger, gerente de comunidade e Rick O'Connor, director executivo. [6]

2.1.4 Open Source

O modelo de licenciamento que RISC-V utiliza é a *BSD Open Source License*. Ou seja, em caso de utilização, apenas dar créditos aos autores, no caso a UC Berkeley. [7]

O fato da ISA ser Open Source traz grandes vantagens, principalmente na parte de distribuição e compartilhamento.

Na parte comercial por exemplo, qualquer pessoa pode criar suas implementações para seus objetivos específicos e comercializar, com seu código fonte sendo aberto ou fechado, apenas é requisitado pela licença que os autores sejam reconhecidos. Diminuindo custos de uso de patentes ou implementações do zero.

Outra vantagem, defendida pelos autores, também defendida no mundo do software livre é a questão de vulnerabilidades na solução. Apesar de parecer contra intuitivo, o fato do código ser aberto, as vulnerabilidades podem ser encontradas e consertadas com maior rapidez, sendo dispensável um auditor para lidar com vulnerabilidades implantadas por desenvolvedores maliciosos de dentro da própria empresa.

Ou mesmo que as vulnerabilidades não tenham sido feitas de forma maliciosa, bugs acontecem e o fato do código ser fechado pode deixar que esta fique escondida por algum tempo antes de poder ser descoberta e explorada, como foi o caso das vulnerabilidades *Spectre* e *Meltdown* [8] que ganharam atenção na mídia recentemente e estão diretamente ligados ao mundo dos processadores por explorarem vulnerabilidades arquiteturais [9].

A RISC-V Foundation escreveu em seu site oficial que não foram encontrados impactos em nenhuma implementação até janeiro de 2018. Apesar dos ataques não serem específicos de uma ISA ou outra, mostra uma vantagem de se ter uma ISA aberta, a velocidade de se corrigir os erros pela comunidade é muito maior. E a possibilidade de poder experimentar e testar novas formas eficientes para resolver estes problemas é histórica. [10].

2.1.5 Características

2.1.6 Como funciona

2.1.7 Instruções

Resumo de instruções

Tipos de instruções

Formatos de instruções

2.2 Montador

2.2.1 Conceito

2.2.2 Algoritmo de duas passagens

2.3 Aplicações web

As aplicações web, são aquelas projetadas para que sua utilização seja feita através de um navegador com acesso à Internet.

2.3.1 Arquitetura

Como a maioria das aplicações web, utilizamos o modelo Cliente - Servidor.
    - FrontEnd
- BackEnd
- API

2.3.2 Vantagens e desvantagens

A grande vantagem e motivo principal pela escolha dessa plataforma é poder ser utilizado de qualquer dispositivo com um browser moderno e acesso à internet. Qualquer pessoa pode acessar a solução acessando um link e começar a desenvolver e estudar códigos escritos para a arquitetura RISC-V. Outra vantagem é a correção de bugs e atualizações para novas versões. Para que os usuários tenham seus softwares atualizados basta atualizar o software em um ponto apenas.

Uma desvantagem é ter o custo de um servidor rodando a aplicação para que seja acessível por vários usuários. Outro fator crítico é ter um único ponto de falha, diferente de uma rede peer-to-peer distribuída. Em relação as desvantagens elas não representam uma significância alta pois se trata de uma ferramenta livre, ou seja, caso haja algum tipo de indisponibilidade do servidor, o usuário poderá baixar o software e rodar em sua própria máquina.

2.4 Extensiblidade

Este projeto tem como objetivo poder ser extendido por outros interessados no estudo da arquitetura.

para casos onde se deseja performance, podem conectados modulos através de extensões para python como cython [11]

Capítulo 3

Ambiente Proposto

Este capítulo descreve com detalhes o ambiente de desenvolvimento que foi realizado neste projeto.

3.1 Ambiente proposto

O desenvolvimento do ambiente foi realizado para a plataforma web. Consiste em três partes principais:

- Editor de texto
- Montador
- Simulador

Toda a parte de interação com o usuário utiliza as tecnologias web: *javascript*, para ter uma interface dinâmica, utilizando a biblioteca *jQuery*. Para a estruturação utilizou-se a linguagem de marcação *HTML*, e a folha de estilos *CSS* para formatação do layout.

Apesar de ter sido desenvolvido para rodar na web, os componentes "Montador"e "Simulador", podem ser utilizados separados, por exemplo em linha de comando. Basta ter instalado um interpretador *Python 3.x*

3.2 Arquitetura de software

Como na maioria das aplicações web, utilizamos o modelo Cliente-Servidor. Neste modelo, o usuário é um cliente, que faz requisições ao servidor.

Neste projeto o usuário faz requisição da página inicial, onde ele pode escrever seu código, então a partir disso ele irá enviar este código para o servidor, requisitando a

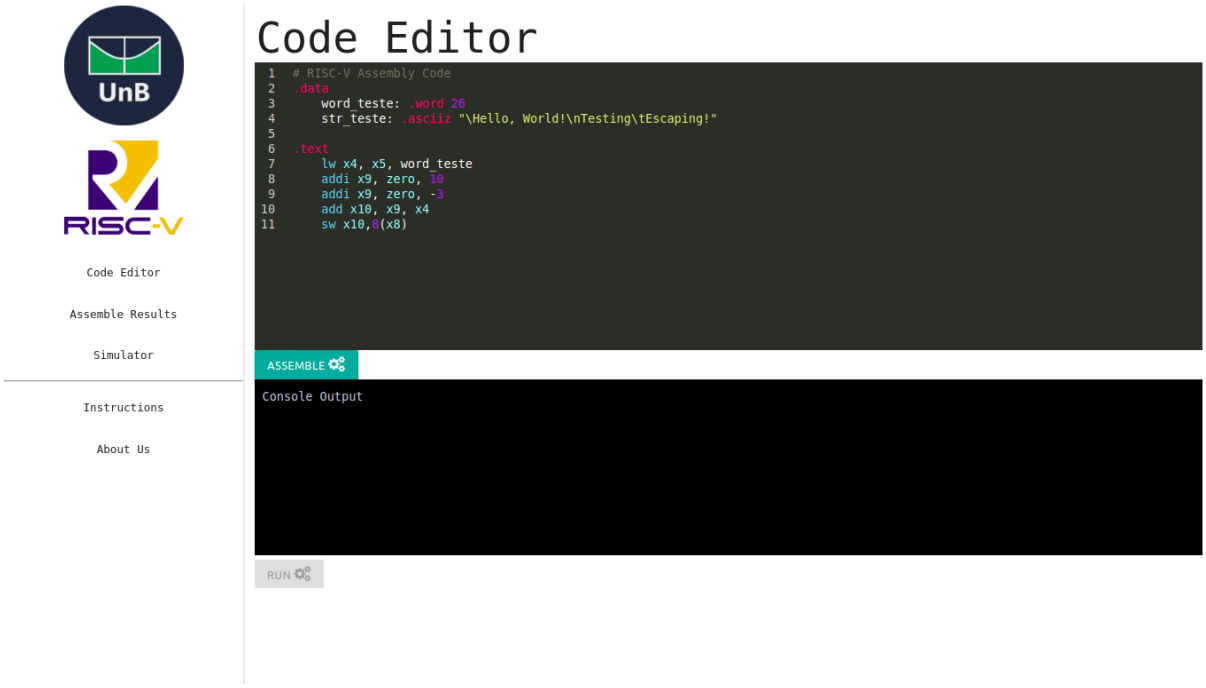

Figura 3.1: Página inicial, mostra o editor de texto com um código exemplo.

montagem, e então receberá a saída do montador, ou a mensagem apropriada no caso de erros.

Com o código montado, pode se salvar em arquivos os dados da montagem em diferentes formatos, binário, hexadecimal ou em formato *MIF ("Memory Initialization File")*, utilizado para inicializar memórias em FPGA's da Altera. Esses dados podem ser enviados novamente para o servidor em uma nova requisição de simulação, e então o servidor irá responder os resultados do programa, qual o estado de memória final, valores dos registradores, e se houver, mensagens de saída.

- modelo cliente servidor - api

3.3 Interface web

A interface web foi feita com as tecnologias padrões da web, HTML, CSS e Javascript. E faz parte do frontend da aplicação, onde o usuário irá interagir com a ferramenta.

Para a parte dinâmica do site foi utilizada a biblioteca jQuery, facilitando o desenvolvimento de funcionaliades no front-end principalmente na manipulação de elementos da tela.

Para o layout foi utilizado o framework Materialize CSS, com sua utilização perdemos menos tempo em detalhes de layout, e podemos focar mais nas funcionalidades.

- single page application exemplod tela - codemirror

3.4 Montador

Utilizou-se neste projeto o algoritmo de duas passagens para montagem. Implementamos apenas as funcionalidades básicas para traduzir códigos *assembly RISC-V* para código de máquina.

entradas e saidas
exemplo de tela

artefatos

3.5 Simulador

imagem exemplo de tela entradas e saidas
artefatos

3.6 Novos módulos

- systemC
- precisão de ciclo
- avaliação de energia

Capítulo 4

Resultados e Avaliação do Sistema

Neste capítulo mostraremos exemplos de resultados de utilização da ferramenta, para isso demonstraremos com alguns códigos simples a interação com o sistema e seus resultados,

- Soma simples
- Sequência de fibonacci

4.1 entradas do sistema

exemplos de codigos
imagens

4.2 saidas do sistema

dados de saida
tela de simulação

4.3 simulação

regitradores
memory map
console
montador

Capítulo 5

Conclusões

5.1 Objetivos atingidos

Neste projeto conseguimos realizar um primeiro estudo da arquitetura RISC-V, e implementar uma interface de fácil utilização, multiplataforma, onde um usuário pode escrever códigos na linguagem assembly RISC-V, realizar a montagem do código, e rodar uma simulação deste código.

O sistema permite exportar as saídas do montador para que o usuário possa simular seu código em outras ferramentas de simulação por exemplo *Spike*, ou uma implementação própria em uma *FPGA*.

Os módulos de montagem e simulação foram implementados como biblitoecas separadas, por isso podem ser utilizadas em outros contextos. Por exemplo, em linha de comando.

5.2 Pontos positivos e negativos

Alguns pontos positivos da ferramenta são aqueles já citados quando descrevemos as vantagens de se utilizar uma plataforma web, e alguns outros,

- Não necessidade de instalação e configurações para começar a utilzar a ferramenta
- Multiplataforma
- Extensibilidade

Desvantagens dessa solução hospedada são,

- Custo de hospedagem
- Possível indisponibilidade

Porém, apesar dessas desvantagens citadas acima, o fato da solução ser aberta diminui o peso dessas desvantagens, pois o código pode ser baixado e rodado localmente.

Assim, mesmo que haja algum problema de hospedagem, pode-se obter facilmente o código fonte, e tendo um interpretador instalado em seu computador, se pode fazer sua própria hospedagem local, pois o interpretador já fornece um servidor web imbutido nativamente para desenvolvimento.

5.3 Dificuldades

No começo do projeto tentou-se implementar um editor de texto em javascript sem utilização de bibliotecas terceiras, porém a tarefa se mostrou muito difícil de ser realizada, então optou-se por utilizar a biblioteca CodeMirror, muito utilizada por vários sites populares de desenvolvimento.

Na parte do montador foi onde houve o maior esforço. A parte documentada que diz respeito a linguagem de programação ainda é escassa, porém para a solução limitada que foi feita neste projeto foi o suficiente.

O simulador foi bem simplificado, por isso não houveram grandes dificuldades. O desenvolvimento do simulador tinham algumas dificuldades de saber como as instruções por exemplo de "branch"e "jump"funcionavam à nível de hardware, pois algumas informações sobre estas não ficam explicitas nas documentações.

Grande parte do esforço também foi direcionado à integração do frontend com o backend, pois pode-se considerar dois sistemas diferentes. O que era um dos objetivos, para que a ferramenta seja flexível, e possa ser extendida e evoluida mais a frente.

5.4 Melhorias e trabalhos futuros

Existem muitas funcionalidades a serem feitas, citando alguns temos:

- Expandir arquitetura para 64-bits
- Sistema de login/logout, com um sistema de usuários, poderiamos montar um sistema para salvar e compartilhar códigos com outros usuários. Também inserir módulos personalizados.
- Contador de instruções.
- Bitmap mapping, para termos uma representação visual da memória do sistema.
- Adição de instruções e pseudo-instruções customizadas.

- Utilização de bibliotecas ou frameworks mais modernos para frontend como ReactJS, AngularJS, VueJS.
- Melhorar interface para dispositivos móveis.
- Step run.

A parte de extensões da solução não foi realizada, porém teria muitas aplicações e diferenciaria bastante de outras ferramentas. Para isso precisaríamos ter uma documentação bem especificada, para que haja compatibilidade da interface entre os módulos de extensão.

Uma das funcionalidades que seria muito interessante ter realizado no projeto seria a possível extensão de códigos em C, para podermos utilizar modelos descritos em mais baixo nível feitos por exemplo com a biblioteca *SystemC*. Com isso poderíamos testar com maior facilidade implementações baseados em performace.

Referências

[1] Waterman, Editors Andrew e RISC V Foundation Krste Asanovic: *The risc-v instruction set manual, volume i: User-level isa, document version 2.2.* https://riscv.org/specifications/, May 2017. 1, 6

[2] Waterman, Andrew: *Design of the RISC-V Instruction Set Architecture.* Tese de Doutoramento, EECS Department, University of California, Berkeley, Jan 2016. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.html. 2, 8

[3] Armasu, Lucian: *Big tech players start to adopt the risc-v chip architecture.* https://www.tomshardware.com/news/big-tech-players-risc-v-architecture,36011.html, November 2017. 4

[4] NVIDIA: *Nvidia risc-v story.* https://riscv.org/wp-content/uploads/2016/07/Tue1100_Nvidia_RISCV_Story_V2.pdf, July 2016. 4

[5] Foundation, RISC V: *Sobre a risc-v foundation.* https://riscv.org/risc-v-foundation/, 2018. Accessed: 2018-06-20. 8

[6] Foundation, RISC V: *Membros do conselho risc-v foundation.* https://riscv.org/leadership/, 2018. Accessed: 2018-06-20. 9

[7] Foundation, RISC V: *Frequentyl asked questions.* https://riscv.org/faq/, 2018. 9

[8] Technology, Graz University of: *Meltdown and spectre exploits.* https://meltdownattack.com/, 2018. Accessed: 2018-06-30. 9

[9] *Two security flaws in modern chips cause big headaches for the tech business.* https://www.economist.com/science-and-technology/2018/01/04/two-security-flaws-in-modern-chips-cause-big-headaches-for-the-tech-business. Accessed: 2018-06-20. 9

[10] Krste Asanović, Rick O'Connor: *Building a more secure world with the risc-v isa.* https://riscv.org/2018/01/more-secure-world-risc-v-isa/, 2018. Accessed: 2018-06-20. 9

[11] *Cython c-extensions for python.* http://cython.org/. Accessed: 2018-06-20. 11