

TRABALHO DE GRADUAÇÃO

RISC-V SiMPLE

Arthur de Matos Beggs

Brasília, Dezembro de 2018



**ENGENHARIA  
MECATRÔNICA**  
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA  
Faculdade de Tecnologia  
Curso de Graduação em Engenharia de Controle e Automação

## TRABALHO DE GRADUAÇÃO

### RISC-V SIMPLE

**Arthur de Matos Beggs**

*Relatório submetido como requisito parcial de obtenção  
de grau de Engenheiro de Controle e Automação*

Banca Examinadora

Prof. Marcus Vinicius Lamar, CIC/UnB  
*Orientador*

\_\_\_\_\_

Prof. Ricardo Pezzuol Jacobi, CIC/UnB  
*Co-Orientador*

\_\_\_\_\_

**Brasília, Dezembro de 2018**

## FICHA CATALOGRÁFICA

ARTHUR, DE MATOS BEGGS

RISC-V SiMPLE,

[Distrito Federal] 2018.

x, 101p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2018). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. RISC-V

2. ???

I. Mecatrônica/FT/UnB

II. Título (Série)

## REFERÊNCIA BIBLIOGRÁFICA

BEGGS, ARTHUR DE MATOS, (2018). RISC-V SiMPLE. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-*n*º022, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 101p.

## CESSÃO DE DIREITOS

AUTOR: Arthur de Matos Beggs

TÍTULO DO TRABALHO DE GRADUAÇÃO: RISC-V SiMPLE.

GRAU: Engenheiro

ANO: 2018

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito do autor.

---

Arthur de Matos Beggs

SHCGN 703 Bl G N° 120, Asa Norte

70730-707 Brasília – DF – Brasil.

## **Dedicatória**

*Dedico ao pato de borracha especialista em TI que sempre me ajuda a depurar meus códigos.*

*Arthur de Matos Beggs*

## Agradecimentos

*Agradecimentos!*

*Arthur de Matos Beggs*

---

## RESUMO

Resumo!

Palavras Chave: RISC-V

---

## ABSTRACT

Abstract!

Keywords: RISC-V

# SUMÁRIO

<b>1</b>	<b>Introdução.....</b>	<b>1</b>
1.1	MOTIVAÇÃO .....	1
1.2	POR QUE <i>RISC-V</i> ? .....	1
1.3	O PROJETO <i>RISC-V SiMPLE</i> .....	2
<b>2</b>	<b>A ISA <i>RISC-V</i> .....</b>	<b>3</b>
2.1	VISÃO GERAL DA ARQUITETURA .....	3
2.2	MÓDULO INTEIRO .....	5
2.3	EXTENSÕES.....	5
2.3.1	EXTENSÃO M .....	5
2.3.2	EXTENSÃO A .....	5
2.3.3	EXTENSÃO F .....	5
2.3.4	EXTENSÃO D .....	5
2.3.5	OUTRAS EXTENSÕES.....	5
2.4	ARQUITETURA PRIVILEGIADA .....	5
2.5	FORMATOS DE INSTRUÇÕES.....	5
2.6	FORMATOS DE IMEDIATOS .....	6
<b>3</b>	<b>Implementação .....</b>	<b>8</b>
3.1	CAMINHO DE DADOS .....	8
<b>4</b>	<b>Resultados.....</b>	<b>11</b>
<b>5</b>	<b>Conclusões.....</b>	<b>12</b>
5.1	PERSPECTIVAS FUTURAS.....	12
	<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>13</b>
	<b>Anexos.....</b>	<b>14</b>
<b>I</b>	<b>Descrição do conteúdo do CD.....</b>	<b>15</b>
<b>II</b>	<b>Programas utilizados.....</b>	<b>16</b>

# LISTA DE FIGURAS

2.1	Codificação de instruções de tamanho variável da arquitetura <i>RISC-V</i> . .....	3
2.2	Formatos de Instruções da <i>ISA RISC-V</i> . .....	5
2.3	Formação do Imediato de tipo I. ....	6
2.4	Formação do Imediato de tipo S. ....	6
2.5	Formação do Imediato de tipo B. ....	6
2.6	Formação do Imediato de tipo U. ....	7
2.7	Formação do Imediato de tipo J. ....	7
3.1	Caminho de Dados implementado para o módulo I. ....	8



# LISTA DE TABELAS

# LISTA DE SÍMBOLOS

## Siglas

BSD	Distribuição de Software de Berkeley — <i>Berkeley Software Distribution</i>
CSR	Registradores de Controle e Estado — <i>Control and Status Registers</i>
FPGA	Arranjo de Portas Programáveis em Campo — <i>Field Programmable Gate Array</i>
hart	<i>hardware thread</i>
ISA	Arquitetura do Conjunto de Instruções — <i>Instruction Set Architecture</i>
MIPS	Microprocessador sem Estágios Intertravados de <i>Pipeline</i> — <i>Microprocessor without Interlocked Pipeline Stages</i>
OAC	Organização e Arquitetura de Computadores
RISC	Computador com Conjunto de Instruções Reduzido — <i>Reduced Instruction Set Computer</i>
SiMPLE	Ambiente de Aprendizado Uniciclo, Multiciclo e <i>Pipeline</i> — <i>Single-cycle Multicycle Pipeline Learning Environment</i>
RAS	Pilha de Endereços de Retorno — <i>Return Address Stack</i>

# Capítulo 1

## Introdução

### 1.1 Motivação

O mercado de trabalho está a cada dia mais exigente, sempre buscando profissionais que conheçam as melhores e mais recentes ferramentas disponíveis. Além disso, muitos universitários se sentem desestimulados ao estudarem assuntos desatualizados e com baixa possibilidade de aproveitamento do conteúdo no mercado de trabalho. Isso alimenta o desinteresse pelos temas abordados e, em muitos casos, leva à evasão escolar. Assim, é importante renovar as matérias com novas tecnologias e tendências de mercado sempre que possível, a fim de instigar o interesse dos discentes e formar profissionais mais capacitados e preparados para as demandas da atualidade.

Hoje, a disciplina de Organização e Arquitetura de Computadores da Universidade de Brasília é ministrada utilizando a arquitetura *MIPS32*. Apesar da arquitetura *MIPS32* ainda ter grande força no meio acadêmico (em boa parte devido a sua simplicidade e extensa bibliografia), sua aplicação na indústria tem diminuído consideravelmente na última década.

Embora a curva de aprendizagem de linguagens *Assembly* de alguns processadores *RISC* seja relativamente baixa para quem já conhece o *Assembly MIPS32*, aprender uma arquitetura atual traz o benefício de conhecer o *estado da arte* da organização e arquitetura de computadores.

Para a proposta de modernização da disciplina, foi escolhida a *ISA RISC-V* desenvolvida na Divisão de Ciência da Computação da Universidade da Califórnia, Berkeley como substituta à *ISA MIPS32*.

### 1.2 Por que *RISC-V*?

A *ISA RISC-V* (lê-se “*risk-five*”) é uma arquitetura *open source* com licença *BSD*, o que permite o seu livre uso para quaisquer fins, sem distinção de se o trabalho possui código-fonte aberto ou proprietário. Tal característica possibilita que grandes fabricantes utilizem a arquitetura para criar seus produtos, mantendo a proteção de propriedade intelectual sobre seus métodos de implementação e quaisquer subconjuntos de instruções não-*standard* que as empresas venham a

desenvolver, o que estimula investimentos em pesquisa e desenvolvimento.

Empresas como Google, IBM, AMD, Nvidia, Hewlett Packard, Microsoft, Oracle, Qualcomm e Western Digital são algumas das fundadoras e investidoras da *RISC-V Foundation*, órgão responsável pela governança da arquitetura. Isso demonstra o interesse das gigantes do mercado no sucesso e disseminação da arquitetura.

A licença também permite que qualquer indivíduo produza, distribua e até mesmo comercialize sua própria implementação da arquitetura sem ter que arcar com *royalties*, sendo ideal para pesquisas acadêmicas, *startups* e até mesmo *hobbyistas*.

O conjunto de instruções foi desenvolvido tendo em mente seu uso em diversas escalas: sistemas embarcados, *smartphones*, computadores pessoais, servidores e supercomputadores, o que permitirá maior reuso de *software* e maior integração de *hardware*.

Outro fator que estimula o uso do *RISC-V* é a modernização dos livros didáticos. A nova versão do livro utilizado em OAC, Organização e Projeto de Computadores, de David Patterson e John Hennessy, utiliza a *ISA RISC-V*.

Além disso, com a promessa de se tornar uma das arquiteturas mais utilizadas nos próximos anos, utilizar o *RISC-V* como arquitetura da disciplina de OAC se mostra a escolha ideal no momento.

### 1.3 O Projeto *RISC-V SiMPLE*

O projeto *RISC-V SiMPLE* (*Single-cycle Multicycle Pipeline Learning Environment*) consiste no desenvolvimento de um processador com conjunto de instruções *RISC-V*, sintetizável em *FPGA* e com *hardware* descrito em *Verilog*. A microarquitetura implementada nesse trabalho é uniciclo, escalar, em ordem, com um único *hart* e com caminho de dados de 64 bits. Trabalhos futuros poderão utilizar a estrutura altamente configurável e modularizada do projeto para desenvolver as versões em microarquiteturas multiciclo e *pipeline*.

O processador contém o conjunto de instruções I (para operações com inteiros, sendo o único módulo com implementação mandatória pela arquitetura) e as extensões *standard* M (para multiplicação e divisão de inteiros) e F (para ponto flutuante com precisão simples conforme o padrão IEEE 754 com revisão de 2008). O projeto não implementa as extensões D (ponto flutuante de precisão dupla) e A (operações atômicas de sincronização), e com isso o *soft core* desenvolvido não pode ser definido como de propósito geral, G (que deve conter os módulos I, M, A, F e D). Assim, pela nomenclatura da arquitetura, o processador desenvolvido é um *RV64IMF*.

O projeto contempla *traps*, interrupções, exceções, *CSRs*, chamadas de sistema e outras funcionalidades de nível privilegiado da arquitetura.

O *soft core* possui barramento Avalon para se comunicar com os periféricos das plataformas de desenvolvimento. O projeto foi desenvolvido utilizando a placa DE2-115 com *FPGA Altera Cyclone* e permite a fácil adaptação para outras placas da Altera.

## Capítulo 2

# A ISA RISC-V

### 2.1 Visão Geral da Arquitetura

A ISA RISC-V é uma arquitetura modular, sendo o módulo base de operações com inteiros mandatório em qualquer implementação. Os demais módulos são extensões de uso opcional. A arquitetura não suporta *branch delay slots* e aceita instruções de tamanho variável. A codificação das instruções de tamanho variável é mostrada na Figura 2.1. As instruções presentes no módulo base correspondem ao mínimo necessário para emular por *software* as demais extensões (com exceção das operações atômicas).

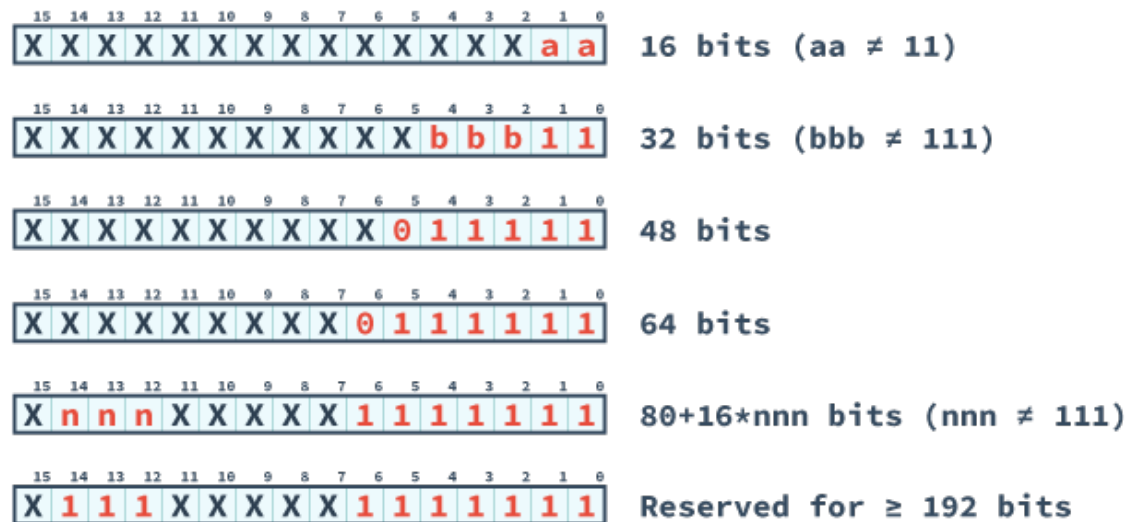


Figura 2.1: Codificação de instruções de tamanho variável da arquitetura RISC-V.

A nomenclatura do conjunto de instruções implementado segue a seguinte estrutura:

- As letras “RV”;
- A largura dos registradores do módulo Inteiro;
- A letra “I” representando a base Inteira. Caso o subconjunto Embarcado (*Embedded*) seja implementado, substitui-se pela letra “E”;
- Demais letras identificadoras de módulos opcionais.

Assim, uma implementação com registradores de 64 bits somente com o módulo base de Inteiros é denominado “RV64I”.

## 2.2 Módulo Inteiro

## 2.3 Extensões

### 2.3.1 Extensão M

### 2.3.2 Extensão A

### 2.3.3 Extensão F

### 2.3.4 Extensão D

### 2.3.5 Outras Extensões

## 2.4 Arquitetura Privilegiada

## 2.5 Formatos de Instruções

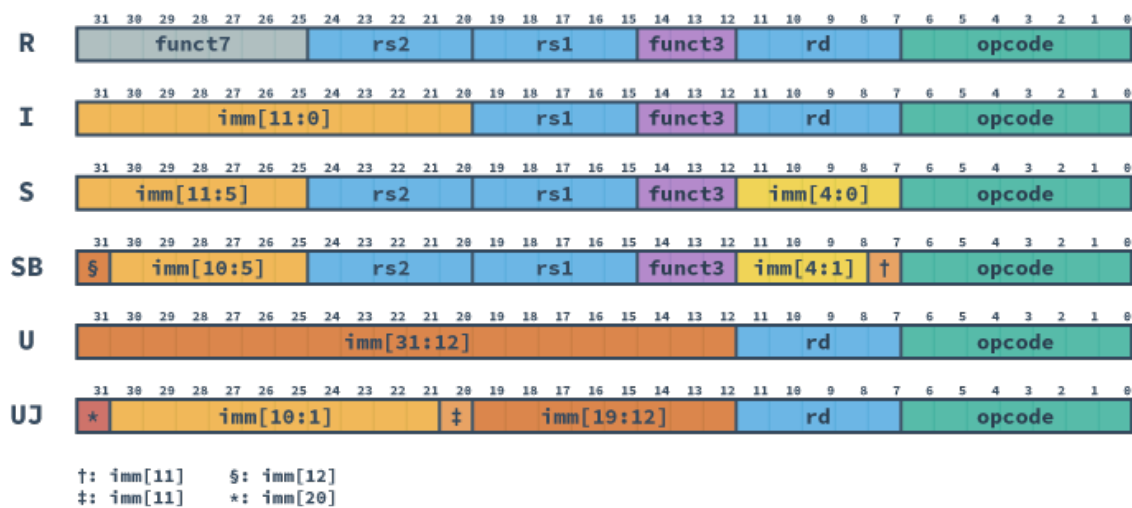


Figura 2.2: Formatos de Instruções da *ISA RISC-V*.

## 2.6 Formatos de Imediatos

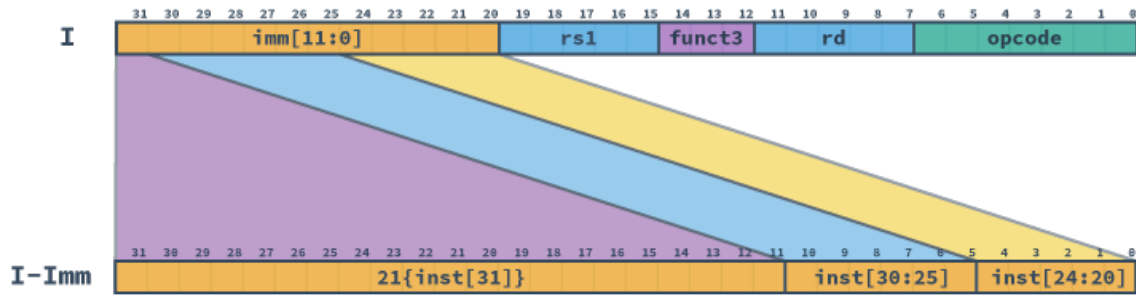


Figura 2.3: Formação do Imediato de tipo I.

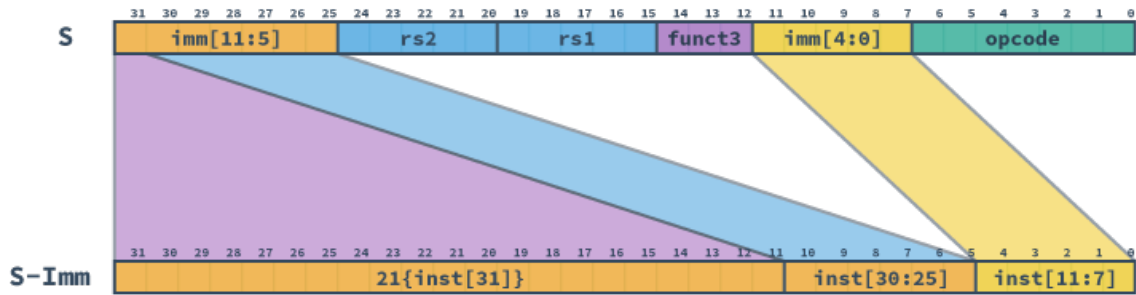


Figura 2.4: Formação do Imediato de tipo S.

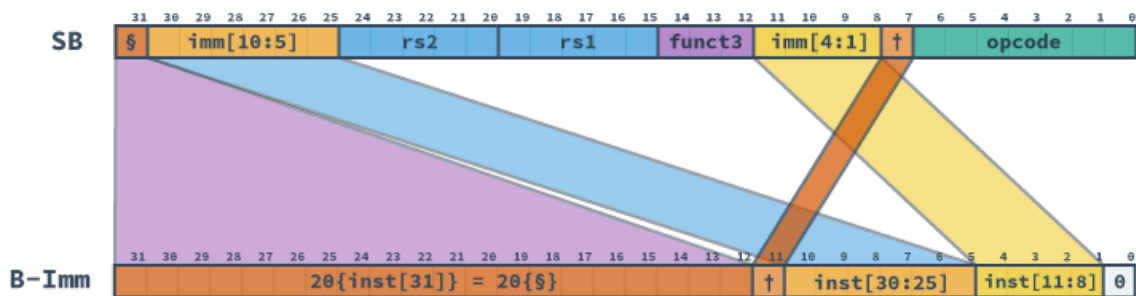


Figura 2.5: Formação do Imediato de tipo B.



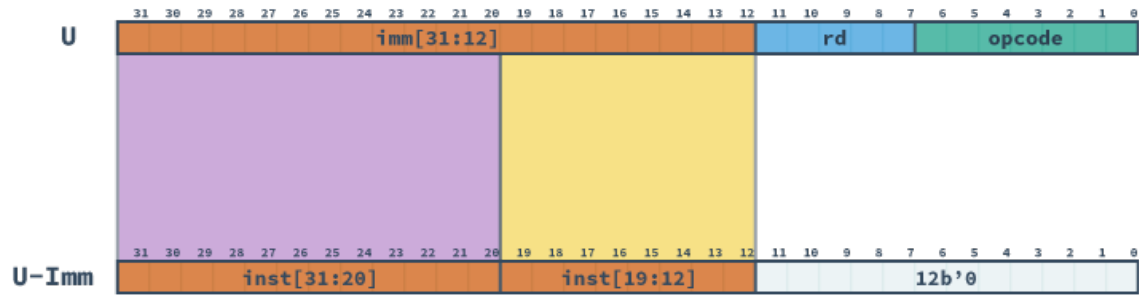


Figura 2.6: Formação do Imediato de tipo U.

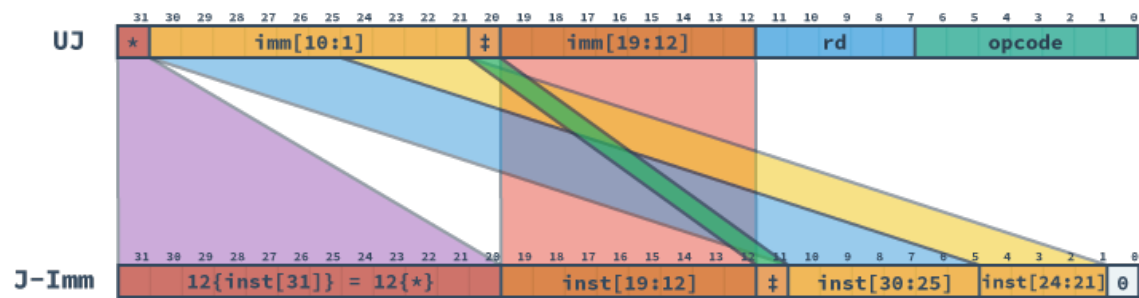


Figura 2.7: Formação do Imediato de tipo J.

## Capítulo 3

# Implementação

### 3.1 Caminho de Dados

O caminho de dados projetado para a implementação da microarquitetura uniciclo é apresentado na Figura 3.1.

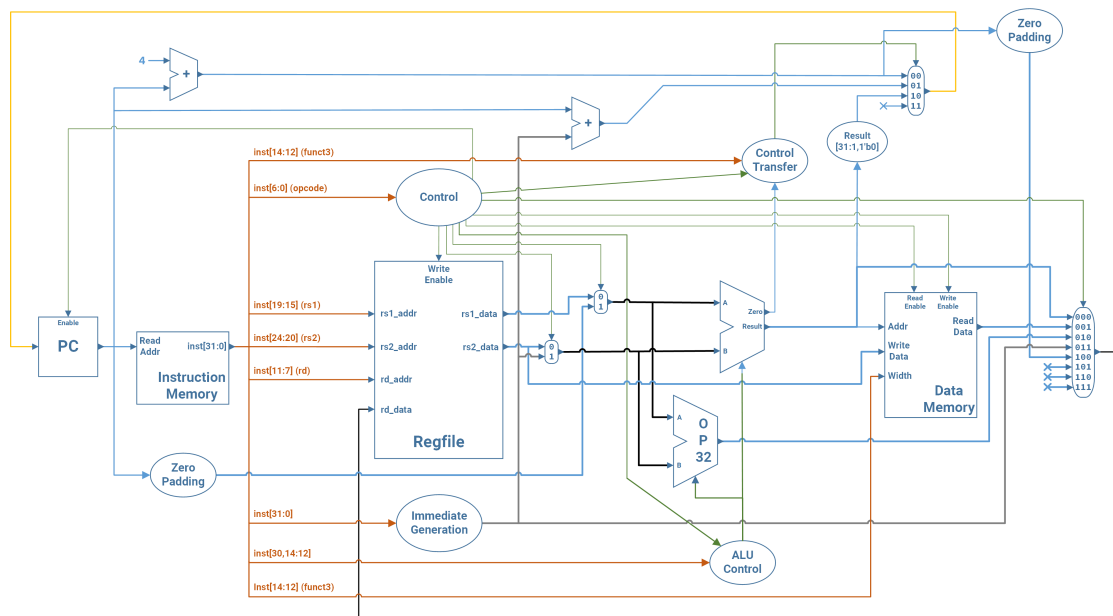


Figura 3.1: Caminho de Dados implementado para o módulo I.

O *datapath* possui um banco de 32 registradores de uso geral de 64 bits cada. A memória possui arquitetura Harvard, sendo a memória de instruções (*text*) *read-only* e a memória de dados (*data*) *read-write*. São implementadas 49 instruções, sendo elas:

- LUI: Load Upper Immediate;

- AUIPC: Add Upper Intermediate to Program Counter;
- JAL: Jump And Link;
- JALR: Jump And Link Register;
- BEQ: Branch if EQual;
- BNE: Branch if Not Equal;
- BLT: Branch if Less Than;
- BGE: Branch if Greater or Equal;
- BLTU: Branch if Less Than Unsigned;
- BGEU: Branch if Greater or Equal Unsigned;
- LB: Load Byte;
- LH: Load Halfword;
- LW: Load Word;
- LBU: Load Byte Unsigned;
- LHU: Load Halfword Unsigned;
- SB: Store Byte;
- SH: Store Halfword;
- SW: Store Word;
- ADDI: ADD Immediate;
- SLTI: Set on Less Than;
- SLTIU: Set on Less Than Unsigned;
- XORI: XOR Immediate;
- ORI: OR Immediate;
- ANDI: AND Immediate;
- SLLI: Shift Left Logical Immediate;
- SRLI: Shift Right Logical Immediate;
- SRAI: Shift Right Arithmetic Immediate;
- ADD: ADD;
- SUB: SUB;

- SLL: Shift Left Logical;
- SLT: Set on Less Than;
- SLTU: Set on Less Than Unsigned;
- XOR: XOR;
- SRL: Shift Right Logical;
- SRA: Shift Right Arithmetic;
- OR: OR;
- AND: AND;
- LWU: Load Word Unsigned;
- LD: Load Double;
- SD: Store Double;
- ADDIW: ADD Immediate Word-size;
- SLLIW: Shift Left Logical Immediate Word-size;
- SRLIW: Shift Right Logical Immediate Word-size;
- SRAIW: Shift Right Arithmetic Immediate Word-size;
- ADDW: ADD Word-size;
- SUBW: SUB Word-size;
- SLLW: Shift Left Logical Word-size;
- SRLW: Shift Right Logical Word-size;
- SRAW: Shift Right Arithmetic Word-size;

Para que o processador seja completamente compatível com a especificação da *ISA*, falta implementar tratamentos de exceções, interrupções e *traps*, Registradores *CSR*, instruções de chamada ao ambiente (ECALL/EBREAK), instruções de *fencing* de memória, suporte ao acesso desalinhado à memória de dados e pilha de endereço de retorno (RAS).

## Capítulo 4

# Resultados

## Capítulo 5

# Conclusões

Concluir

### 5.1 Perspectivas Futuras

Perspectivas futuras

## REFERÊNCIAS BIBLIOGRÁFICAS

# ANEXOS



# **I. DESCRIÇÃO DO CONTEÚDO DO CD**

Descrever CD.

## II. PROGRAMAS UTILIZADOS

Quais programas foram utilizados?