UNIVERSIDADE FEDERAL DE SANTA MARIA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Daniel Brenner Seitenfus

Nikolas Corrêa Machado

# DESENVOLVIMENTO DE UM SIMULADOR PARA O PROCESSADOR MIPS

Santa Maria, RS

2019

# SUMÁRIO

[1 INTRODUÇÃO 3](#_Toc12823243)

[2 OBJETIVOS 3](#_Toc12823244)

[2.1 Objetivo Geral 3](#_Toc12823246)

[2.2 Objetivos Específicos 3](#_Toc12823248)

[3 REVISÃO BIBLIOGRÁFICA 3](#_Toc12823252)

[4 ESPECIFICAÇÕES E METODOLOGIA 5](#_Toc12823266)

[5 EXPERIMENTOS E RESULTADOS 7](#_Toc12823280)

[6 DISCUSSÃO 8](#_Toc12823281)

[7 CONCLUSÕES E PERSPECTIVA 8](#_Toc12823282)

[8 BIBLIOGRAFIA 8](#_Toc12823283)

[9 ANEXOS 8](#_Toc12823290)

## INTRODUÇÃO

Os computadores, atualmente, podem ser analisados por diversos aspectos. Com a possibilidade de estudo em diversos níveis de abstração, torna-se possível a compreensão do funcionamento de um computador de formas distintas. É imprescindível que estudantes de computação compreendam e consigam analisar, por meio desses níveis, o funcionamento de componentes importantes para a organização de computadores.

A compreensão da união entre *software* e *hardware*, do fornecimento de instruções e como o processamento dessas é realizado em computadores, possui grande importância na disciplina de Organização de Computadores, assim, é nesse contexto que o artigo aborda o desenvolvimento de um algoritmo para simulação de um processador.

Este trabalho foi desenvolvido para a disciplina de Organização de Computadores, ministrada pelo Prof. Dr. Giovani Baratto, da Universidade Federal de Santa Maria.

## OBJETIVOS

## Aqui deve conter texto.

## 2.1 Objetivo Geral

## O objetivo principal deste relatório é documentar o desenvolvimento de um algoritmo para simular subconjunto de instruções do processor MIPS.

## 2.2 Objetivos Específicos

## - Realizar uma breve revisão bibliográfica acerca do tema abordado;

## - Documentar o processo de desenvolvimento do simulador

## - Relatar

## 3 REVISÃO BIBLIOGRÁFICA

## A simulação é uma técnica muito utilizada em diversas áreas, na Computação, em especial, é aplicada para reproduzir o comportamento de um sistema real em um computador. A técnica pode ser utilizada para testes, validação e, como no caso abordado no presente artigo, como meio para o processo de ensino-aprendizagem. Como uma ferramenta educacional, o simulador de um processador facilita a compreensão do funcionamento em baixo nível. Pegden (1991) define que a “simulação é processo de projetar um modelo computacional de um sistema real e conduzir experimentos com esse modelo com o propósito de entender seu comportamento e/ou avaliar estratégia para a sua operação”.

## Há diversos processadores em utilização no mercado atualmente. O trabalho trará o enfoque ao processador MIPS (Microprocessor without Interlocked Pipe Stages), que foi projetado na Universidade de Standford. A fabricação do processador começou em 1984 pela Silicon Graphics, Inc. e teve grande aceitação no mercado. Em meados da década de 90, estimava-se que um em cada três microprocessadores com a arquitetura RISC, em comercialização, eram MIPS. Atualmente ele é desenvolvido pela MIPS Technologies (LAMBOIA, 2008, p. 17).

## A linha de arquitetura do processador MIPS é chamada de RISC, abreviação para Reduced Instruction Set Computer ou Computador com um Conjunto Reduzido de Instruções. Um computador que implementa a arquitetura RISC possui uma Unidade de Controle muito mais simples, barata e mais eficiente, comparado a arquitetura CISC (Complex Instruction Set Computer ou Computador com um Conjunto Complexo de Instruções), por exemplo. Já os computadores CISC possuem arquiteturas complexas que facilitam o desenvolvimento de compiladores.

## O desempenho de um processador é sempre um fator de extrema importância quando se projeta um dispositivo. Um fator importante para o funcionamento eficiente de processadores é a previsibilidade. As instruções para arquiteturas RISC foram projetadas para serem o mais simples possível, possuindo um conjunto de instruções reduzidas e poucos endereços. A pouca diversidade de instruções e forma de endereçamento, além de demandar uma Unidade de Controle mais simples, faz com que a previsibilidade possa ser realizada. Como as intrusões variam pouco de uma para outra, é mais fácil para a Unidade de Controle prever quantos ciclos serão necessários para executa-las. Ao saber quantos ciclos serão necessários para executar um estágio de uma instrução, a Unidade de Controle saberá exatamente quando será possível iniciar o estágio de uma próxima instrução. (BRITO, s/d).

## Um ciclo de instrução caracteriza-se, basicamente, pela sequência de ações que o computador realiza para executar cada instrução em código de máquina. A expressão "ciclo de busca e execução" por vezes é empregada, pois descreve em essência o modo como um computador funciona: a instrução deve ser buscada na memória principal e depois executada pela CPU. (MANO, 2007).

## Outro fator importante para o desempenho é o acesso a memória. Sabendo-se que as instruções ficam armazenadas em memória, é inevitável que operações entre processador e memória sejam realizadas, e é com objetivo de obter os códigos que o processador executa a etapa de busca. Dados são trazidos da memória para servirem de entrada para a ULA, e/ou o resultado seja levado para armazenamento na memória.

## Caso a instrução utilizar dois ou mais parâmetros de memória, serão necessárias mais buscas. Como o acesso a memória é sempre mais lento do que o acesso aos registradores, instruções que necessitam de busca de dados em memória são muito mais lentas do que outras instruções.

## Como já referido, o padrão definido pelos projetistas do MIPS foi manter todas as instruções com o mesmo tamanho, exigindo, assim, diferentes tipos de formatos para diferentes tipos de instruções (PATTERSON, et. Al, 2017). Existem algumas categorias para as instruções, onde em cada, a palavra de 32 bits pode ser dividida em diversos campos, resultando em principalmente três formatos: R (com operandos tipo registradores), I (com operandos imediatos) e J (instruções de saltos incondicionais).

## Além da classificação dos formatos, ainda é possível classificar as instruções em seis subconjuntos: transferência de informação, operações aritméticas, operações lógicas e de comparação, operações de deslocamento de bits, instruções de salto, e exceções (UMINHO, s/d).

## Nesse contexto, torna-se imprescindível que o processador realize a tarefa de decodificação. Para que ela aconteça, sempre é identificado o operador (ou *opcode*), que sempre ocupa 6 bits. A partir dele, é reconhecido o formato da instrução e o reconhecimento dos campos necessários para a execução.

## Falar sobre execução.

## PROCEDIMENTOS

## ORGANIZAÇÃO DA MEMÓRIA

## 4 ESPECIFICAÇÕES E METODOLOGIA

## Para o desenvolvimento do algoritmo, utilizou-se a linguagem de montagem Assembly. Para a execução do código e testes de funcionamento, o programa que simula o funcionamento do processador MARS foi utilizado.

## Buscando manter padrões de codificação para o desenvolvimento do algoritmo simulador, definiu-se que a utilização de registradores seguiria a convenção padrão. A Figura 1 mostra o nome, o número, a utilização e se o conteúdo dele é preservado ou não em uma chamada.

## Figura 1 – Convenção para uso de registradores

## 

## Fonte: UMINHO, s/d.

## O código desenvolvido possui três entradas de dados: dois arquivos binários que são utilizados para popular os vetores de manipulação de instruções e memória; e um valor inteiro, informado por meio do teclado, que determina quantas instruções serão executadas.

## Quando o algoritmo é executado, o arquivo “text.bin” e “data.bin” são abertos, no código, conforme demonstra o fluxograma presente na Figura 2.

## Figura 2 – Fluxograma para abertura de arquivo

## C:\Users\danie\Downloads\Desenho sem título.jpg

## Fonte: autores

## A leitura do arquivo “text.bin” é realizada buscando uma instrução por vez. Uma vez lidas, são adicionas no vetor que representa a memória de texto. O endereço de memória efetivo (para atribuição da instrução ao vetor) é calculado a partir da Fórmula 1, onde o *endereço\_base* contém o endereço de memória do primeiro elemento do vetor; *deslocamento* é um contador que itera a cada instrução lida; e *tamanho\_bytes* é o tamanho de qualquer instrução do processador MIPS.

## 

## 

## 1. Buscar instrução da memória. 2. Ler registradores enquanto a instrução é decodificada. O formato das instruções MIPS permite que a leitura e a decodificação ocorram simultaneamente. 3. Executar a operação ou calcular um endereço. 4. Acessar um operando na memória de dados. 5. Escrever o resultado em um registrador.

## 5 EXPERIMENTOS E RESULTADOS

## Acho que falar como a gente testou se estava funcionando...

## 6 DISCUSSÃO

## Relatar aqui dificuldades e se a gente conseguiu fazer tudo... acho que é isso

## Embora o objetivo do trabalho seja complexo, conseguimos desenvolver algumas funções do processador e testá-las

## 7 CONCLUSÕES E PERSPECTIVA

## Após o desenvolvimento e testes, pode-se perceber que é possível um programa escrito em Assembly MIPS simular um conjunto de instruções do processador MIPS.

## 8 BIBLIOGRAFIA

## BRITO, A. V. Introdução a Arquitetura de Computadores. Disponível em: <http://producao.virtual.ufpb.br/books/edusantana/introducao-a-arquitetura-de-computadores-livro/livro/livro.chunked/index.html>. Acesso em: 25 jun. 2019.

## LAMBOIA, F. 2008. Analise comparativa de uso dos conjuntos de Instruções dos microprocessadores de 32 bits mips, powerpc e sparc. Disponível em: <https://www.acervodigital.ufpr.br/bitstream/handle/1884/17292/dissertacao\_Fabiany.pdf?sequence=1>. Acesso em: 25 jun. 2019.

## MANO, R. Representação de Instruções. Disponível em: <https://web.archive.org/web/20070503073902/http://wwwusers.rdc.puc-rio.br/rmano/ri2cinst.html>. Acesso em: 28 jun. 2019.

## PATTERSON, D. A.; HENNESSY, J. L. (2014). Organização e projeto de computadores: a interface hardware/software. 4a ed. Elsevier.

## PEGDEN, C. D,; SHANNON, R. E.; SADOWSKI, R. P. Introduction to simulation using SIMAN. McGraw-Hill, NY. 2 ed., 1990.

## UMINHO. Arquitectura de Computadores. Disponível em: <http://gec.di.uminho.pt/lesi/ac20102/AssemblyMIPS.pdf>. Acesso em 28 jun. 2019.

## 9 ANEXOS

**Capa**

**Sumário**

**1 Introdução**

**2 Objetivos**

**2.1 Objetivo geral**

**2.2 Objetivo específicos**

**3 Revisão bibliográfica**

(Conceitos, o que é busca, decodificação, execução?

Definição de algumas instruções mais importantes

Como construir um simulador a nível de software?

Não colocar história)

**4 Especificações e metodologia**

O que eu quero que meu processador faça? Decodificar algumas instruções

Como eu vou fazer esse programa? Usar linguagem assembly, usei o simulador MARS, colocar o fluxograma de execução

Colocar os procedimentos (leitura de arquivo, decodificação)

**5 Experimento e resultados**

**6 Discussão**

Funcionou? Mais ou menos?

**7 Conclusões e perspectivas**

**8 Bibliografia**

**9 Anexos**

Códigos