PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA
Bacharelado em Engenharia de Software

Ian Asenjo Dominguez Cunha, Laura Lourdes Coutinho Rodrigues,Ricardo Christovão da Silva, Warley Leandro dos Anjos

Trabalho Prático III: Simulador de Instruções do MIPS

Belo Horizonte
2020

Ian Asenjo Dominguez Cunha, Laura Lourdes Coutinho Rodrigues, Warley Leandro dos Anjos

Trabalho Prático III: Simulador de Instruções do MIPS

Trabalho sobre Simulador de Instruções do MIPS, apresentado como requisito parcial à aprovação na disciplina Arquitetura de Computadores.

Professor: Pedro Henrique

Belo Horizonte

2020

SUMÁRIO

1.Categoria

1.1 Unidade Lógica Aritmética

A Unidade Lógica Aritmética (ULA) é um componente do caminho de dados que realiza operações lógicas e aritméticas. Normalmente, a ULA recebe dois operandos como entrada, e uma entrada de controle que permite especificar qual operação deverá ser realizada.
Por esse motivo, a construção de uma ULA se baseia em dois fundamentos principais: o controle de fluxo de dados e a construção de circuitos que implementam operações.
Exemplos de operações realizadas pelo componente são lógica (AND,OR,NOT,XOR,NAND) e aritméticas (ADD, SUB, MULT,DIV).

1.2 Banco de Registradores

O componente Banco de Registradores é composto por um conjunto de registradores que são acessados de forma organizada. De uma maneira geral, podem ser executadas operações de leitura dos dados anteriormente gravados e de escrita de dados para modificar as informações internas.As informações que estão sendo processados em um determinado momento devem estar armazenadas no banco de registradores.

1.3 Memória Principal

Sua função é armazenar informações que são ou serão manipuladas pelo sistema para que elas possam ser recuperadas quando necessário.Existem duas únicas operações possíveis que podem ser realizadas em uma memória Load (LW) e Store (SW).

2. Metodologia

Para desenvolvimento do trabalho os grupo utilizou para comunicação as ferramentas Discord e Whatsapp, o desenvolvimento foi feito pelo Visual Studio Code, dispondo do recurso Live Share. As tarefas foram organizadas pelo Trello, em que um membro ficou responsável por auxiliar e o código fonte da aplicação está disponível no GitHub.
A linguagem utilizada para o desenvolvimento do trabalho foi Java, pois todos os integrantes já estão familiarizados com suas ferramentas.

3. Considerações Finais

Este trabalho faz parte de uma série que tem como objetivo final a construção de um simulador de uma máquina MIPS.
Foi implementado um programa que simula o funcionamento de uma máquina MIPS, dotada de Unidade Lógica e Aritmética, Banco de Registradores e Memória Principal.

Referências

https://www2.pcs.usp.br/~labdig/pdffiles_2014/banco-registradores.pdf

http://www.dca.fee.unicamp.br/~tavares/courses/2015s2/ea773-3.pdf

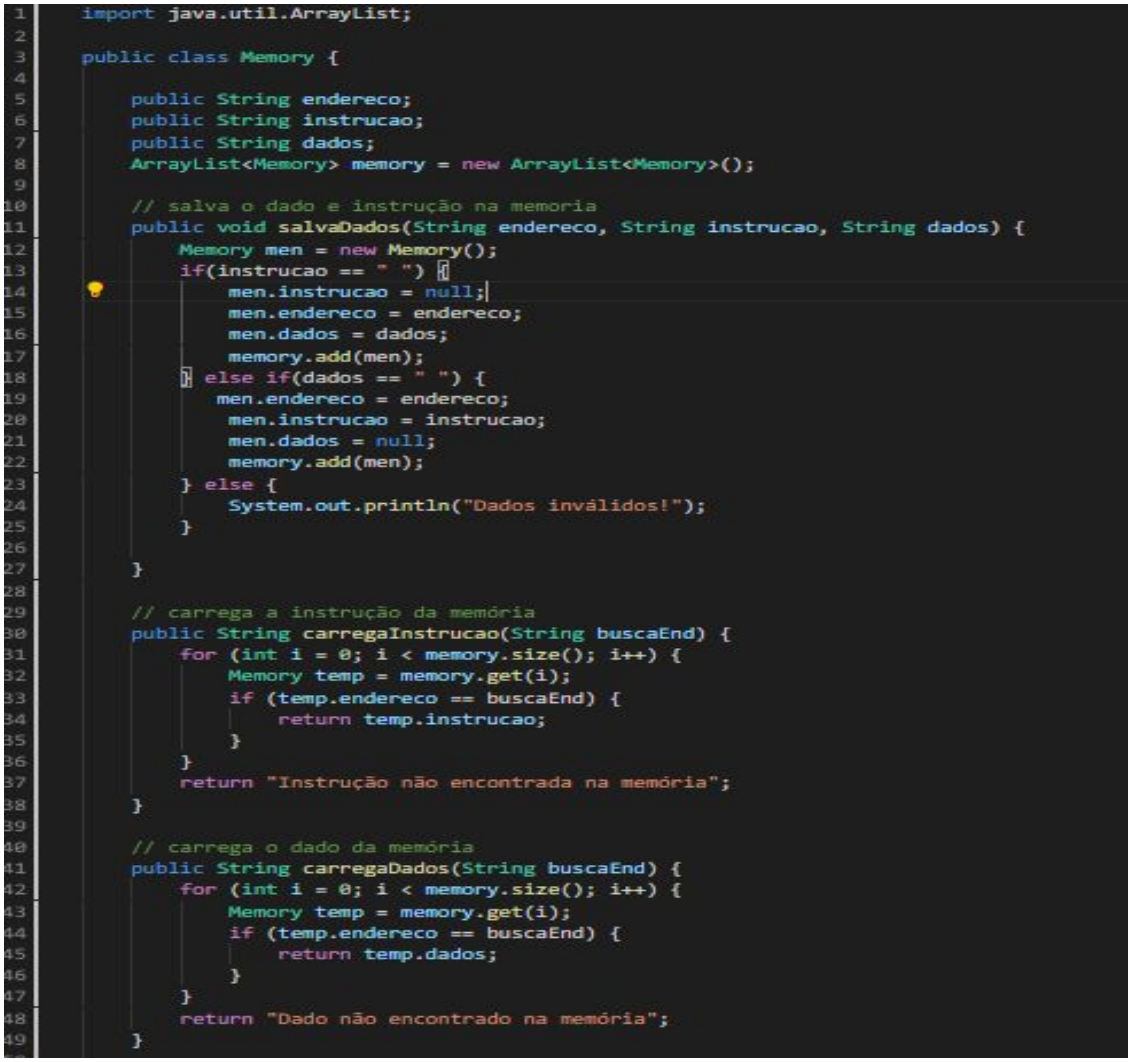
https://www.ic.unicamp.br/~pannain/mc722/aulas/arq_hp5.pdf

http://www.ic.uff.br/~boeres/slides_FAC/FAC-aula3.pdf

Anexos

https://github.com/WarleyLeandro/simulador-de-instrucoes-MIPS

Figura 1: Memória



```java
import java.util.ArrayList;

public class Memory {

    public String endereco;
    public String instrucao;
    public String dados;
    ArrayList<Memory> memory = new ArrayList<Memory>();

    // salva o dado e instrução na memoria
    public void salvaDados(String endereco, String instrucao, String dados) {
        Memory men = new Memory();
        if(instrucao == " ") {
            men.instrucao = null;
            men.endereco = endereco;
            men.dados = dados;
            memory.add(men);
        } else if(dados == " ") {
            men.endereco = endereco;
            men.instrucao = instrucao;
            men.dados = null;
            memory.add(men);
        } else {
            System.out.println("Dados inválidos!");
        }

    }

    // carrega a instrução da memória
    public String carregaInstrucao(String buscaEnd) {
        for (int i = 0; i < memory.size(); i++) {
            Memory temp = memory.get(i);
            if (temp.endereco == buscaEnd) {
                return temp.instrucao;
            }
        }
        return "Instrução não encontrada na memória";
    }

    // carrega o dado da memória
    public String carregaDados(String buscaEnd) {
        for (int i = 0; i < memory.size(); i++) {
            Memory temp = memory.get(i);
            if (temp.endereco == buscaEnd) {
                return temp.dados;
            }
        }
        return "Dado não encontrado na memória";
    }
```

Figura 2: Memória Continuação

```java
    // gera todos endereços de 32
    public String geraEndereco(int indice) {
        String[] enderecos = { "0x000", "0x001", "0x002", "0x003", "0x004", "0x005", "0x006", "0x007", "0x008", "0x009",
                "0x010", "0x011", "0x012", "0x013", "0x014", "0x015", "0x016", "0x017", "0x0018", "0x0019", "0x020",
                "0x021", "0x022", "0x023", "0x024", "0x025", "0x026", "0x027", "0x028", "0x029", "0x030", "0x031" };
        return enderecos[indice];
    }

    // imprimir arquivo
    public ArrayList<Memory> imprimeMemoria() {
      return memory;
    }

}
```

Figura 3: ALU

```java
public class ALU {
    public static final short ADD = 0;
    public static final short SUB = 1;
    public static final short MULT = 2;
    public static final short DIV = 3;
    public static final short AND = 4;
    public static final short OR = 5;
    public static final short XOR = 6;
    public static final short NOR = 7;
    public static final short SLT = 8;
    public static final short SLL = 9;
    public static final short SRL = 11;
    public static final short JR = 12;
    public static final short ADDI = 13;
    public static final short ANDI = 14;
    public static final short ORI = 15;
    public static final short SLTI = 16;
    public static final short BEQ = 17;
    public static final short LW = 18;
    public static final short SW = 19;
    public static final short J = 20;
    public static final short JAL = 21;

    public String getInstrucao(int indice) {
        String[] name = { "ADD", "SUB", "MULT", "DIV", "AND", "OR", "XOR", "NOR", "SLT", "SLL", "SRL", "JR", "ADDI", "ANDI", "ORI", "SLTI", "BEQ", "LW", "SW", "J", "JAL" };
        return name[indice];
    }

}
```

Figura 5: Trello da Equipe