Universidade do Vale do Itajaí - UNIVALI
Escola do Mar, Ciência e Tecnologia
Ciência da Computação

Carlos Eduardo Prezzi, Thomas Jefferson Sozio Hammes

Programação em Linguagem de Montagem

Arquitetura e Organização de Processadores – Prof. Thiago Felski

Itajaí - SC
01/09/2022

1.1: Código Fonte em Linguagem de Montagem MIPS

Afim de poupar espaço de relatório, apenas o código do Programa 1 será incluso no mesmo, porém os arquivos hexadecimais dos demais Programas está incluso na calculadora de CPI.

```
# Disciplina: Arquitetura e Organização de Computadores
# Atividade: Avalaliação 01 Programação em Linguagem de Montagem
# Programa 01
# Grupo: -Carlos Eduardo Prezzi
#        -Thomas Sozio Hammes

.data
      pedido: .asciiz "Digite o tamanho dos vetores (2-8): "
      tamanho_incorreto: .asciiz "Tamanho de vetor incorreto inserido, ele
deve varia entre dois e oito. "
      insercao_vetor_a_pre: .asciiz "Vetor_A["
      insercao_vetor_b_pre: .asciiz "Vetor_B["
      insercao_vetor_pos: .asciiz "]= "
      new_line: .asciiz "\n"
      Vetor_A: .word 0, 0, 0, 0, 0, 0, 0, 0
      Vetor_B: .word 0, 0, 0, 0, 0, 0, 0, 0

.text
      la $s7, Vetor_A #Passa os valores do Vetor_A para $s7
      la $s6, Vetor_B
      j pedir_tamanho_vetor

reportar_tamanho_incorreto:
      addi $v0, $zero, 4 #Passando o valor 4 para printar uma string na
syscall
      la $a0, tamanho_incorreto #Passando a string do pedido para o
argumento da syscall
      syscall #Chama as coisas passadas acima

pedir_tamanho_vetor:
      addi $v0, $zero, 4 #Passando o valor 4 para printar uma string na
syscall
      la $a0, pedido #Passando a string do pedido para o argumento da
syscall
      syscall #Chama as coisas passadas acima

      addi $v0, $zero, 5 #Passando o valor 5 para a syscall de leitura de
inteiro
      syscall #Inteiro vai ser armazenado em $v0
      addi $s0, $v0, 0 #Valor digitado armazenado em $s0

      addi $t0, $zero, 9 #Setando o valor de t0 para checar se o valor
digitado ï¿½ menor ou igual a 8
      slt $t1, $s0, $t0 #Se o valor digitado ï¿½ menor que 9, retorna
verdadeiro em $t1
      bne $t1, 1, reportar_tamanho_incorreto

      addi $t0, $zero, 1 #Setando o valor de t0 para checar se o valor
```

```
digitado ï¿½ maior ou igual a 2
      slt $t1, $t0, $s0 #Se o valor digitado ï¿½ menor que 2, retorna falso
em $t1
      bne $t1, 1, reportar_tamanho_incorreto

seta_vetor_a:
      addi $t4, $zero, 0 #Declara que vai mexer no A
      la $t3, insercao_vetor_a_pre #Argumento pra imprimir o prefixo do
vetor 1
      jal prepara_for_input #Pula pro for pra nï¿½o setar o valor de b

seta_vetor_b:
      addi $t4, $zero, 1 #Declara que vai mexer no B
      la $t3, insercao_vetor_b_pre #Argumento pra imprimir o prefixo do
vetor 2
      jal prepara_for_input  #Pula pro for pra poder voltar aqui depois

trocar_elementos: jal prepara_for_troca #Poderia mudar o endereço dos dois
vetores, mas creio que tenha que trocar literalmente os valores

      addi $v0, $zero, 4 #passa o valor para printar strings para v0
      la $a0, new_line #passa o valor de nova linha
      syscall

mostrar_elementos_a:
      addi $t4, $zero, 0 #Declara que vai mexer no A
      la $t3, insercao_vetor_a_pre #Argumento pra imprimir o prefixo do
vetor 1
      jal prepara_for_output #Pula pro for pra nï¿½o printar o valor de b

mostrar_vetor_b:
      addi $t4, $zero, 1 #Declara que vai mexer no B
      la $t3, insercao_vetor_b_pre #Argumento pra imprimir o prefixo do
vetor 2
      jal prepara_for_output  #Pula pro for pra poder voltar aqui depois

j parar


prepara_for_input:
      add $t0, $zero, $zero   #int i = 0;

for_input:
      addi $v0, $zero, 4 #passa o valor para printar strings para v0
      la $a0, ($t3) #Passa o prefixo do vetor
      syscall

      addi $v0, $zero, 1 #passa o valor para printar inteiros para v0
      add $a0, $zero, $t0 #passa o valor de i
      syscall

      addi $v0, $zero, 4 #passa o valor para printar strings para v0
      la $a0, insercao_vetor_pos #Passa o sufixo do vetor
      syscall
```

```
      addi $v0, $zero, 5 #Passando o valor 5 para a syscall de leitura de
inteiro
      syscall #Inteiro vai ser armazenado em $v0

      sll $t5, $t0, 2 #Multiplica o valor de i por 4 e armazena em t5

      beq $t4, 1, push_vetor_b #Se o valor de t5 é 1, deve mexer no B

push_vetor_a:
      add $t6, $t5, $s7 #Somando index à posição inicial do vetor A
      sw $v0, ($t6) #Escrevendo o valor passado anteriormente na posição do
$t6
      j soma_for_input

push_vetor_b:
      add $t6, $t5, $s6 #Somando index à posição inicial do vetor B
      sw $v0, ($t6) #Escrevendo o valor passado anteriormente na posição do
$t6

soma_for_input:
      add $t0, $t0, 1   #i++;

      bne $t0, $s0, for_input #volta pro for se i != x
      jr $ra #volta pro endereï¿½o anterior

prepara_for_troca:
      add $t0, $zero, $zero   #int i = 0;

for_troca:
      sll $t1, $t0, 2 #Multiplica o valor de i por 4 e armazena em t1
      add $t2, $t1, $s7 #Soma o valor acima como index do Vetor_A
      add $t3, $t1, $s6 #Soma o valor acima como index do Vetor_B
      lw $t4, ($t2) #Armazena o valor do Vetor_A em t4
      lw $t5, ($t3) #Armazena o valor do Vetor_B em t5
      sw $t5, ($t2) #Passa o valor do vetor b pro a
      sw $t4, ($t3) #Passa o valor temporarario do vetor a pro b

soma_for_troca:
      add $t0, $t0, 1   #i++;

      bne $t0, $s0, for_troca #volta pro for se i != x
      jr $ra #volta pro endereï¿½o anterior

prepara_for_output:
      add $t0, $zero, $zero   #int i = 0;

for_output:
      addi $v0, $zero, 4 #passa o valor para printar strings para v0
      la $a0, ($t3) #Passa o prefixo do vetor
      syscall

      addi $v0, $zero, 1 #passa o valor para printar inteiros para v0
      add $a0, $zero, $t0 #passa o valor de i
      syscall
```

```
	addi $v0, $zero, 4 #passa o valor para printar strings para v0
	la $a0, insercao_vetor_pos #Passa o sufixo do vetor
	syscall

	sll $t5, $t0, 2 #Multiplica o valor de i por 4 e armazena em t5

	beq $t4, 1, print_vetor_b #Se o valor de t5 é 1, deve mexer no B

print_vetor_a:
	add $t6, $t5, $s7 #Somando index à posição inicial do vetor B
	lw $t7, ($t6) #Escrevendo o valor do endereço na posição t7
	j soma_for_output

print_vetor_b:
	add $t6, $t5, $s6 #Somando index à posição inicial do vetor B
	lw $t7, ($t6) #Escrevendo o valor do endereço na posição t7

soma_for_output:
	addi $v0, $zero, 1 #passa o valor para printar inteiros para v0
	add $a0, $zero, $t7 #passa o valor de t7
	syscall

	addi $v0, $zero, 4 #passa o valor para printar strings para v0
	la $a0, new_line #passa o valor de nova linha
	syscall

	add $t0, $t0, 1   #i++;

	bne $t0, $s0, for_output      #volta pro for se i != x
	jr $ra #volta pro endereï¿½o anterior

parar:
	addi $v0, $zero, 10 #Setando o valor 10 para a syscall chamar a
parada do programa
	syscall
```

1.2: Código do MIPS em Hexadecimal

O próprio MARS possui uma função para exportar o arquivo .asm para um arquivo de texto hexadecimal. O mesmo foi executado para para obter os valores de cada programa.

```
3c011001        200c0001        0000000c        0000000c
34370088        3c011001        20020005        20020001
3c011001        342b0076        0000000c        00082020
343600a8        0c10002e        00086880        0000000c
08100009        0c100046        20010001        20020004
20020004        20020004        102c0003        3c011001
3c011001        3c011001        01b77020        3424007f
34240025        34240083        adc20000        0000000c
0000000c        0000000c        08100043        00086880
20020004        200c0000        01b67020        20010001
3c011001        3c011001        adc20000        102c0003
34240000        342b006d        21080001        01b77020
0000000c        0c100051        1510ffea        8dcf0000
20020005        200c0001        03e00008        08100064
0000000c        3c011001        00004020        01b67020
20500000        342b0076        00084880        8dcf0000
20080009        0c100051        01375020        20020001
0208482a        0810006e        01365820        000f2020
20010001        00004020        8d4c0000        0000000c
1429fff1        20020004        8d6d0000        20020004
20080001        21640000        ad4d0000        3c011001
0110482a        0000000c        ad6c0000        34240083
20010001        20020001        21080001        0000000c
1429ffed        00082020        1510fff7        21080001
200c0000        0000000c        03e00008        1510ffe5
3c011001        20020004        00004020        03e00008
342b006d        3c011001        20020004        2002000a
0c10002e        3424007f        21640000        0000000c
```

2.1: Código Fonte do Programa em Java

Classe Binary Converter

Responsável em converter um arquivo hexadecimal para um array de strings binárias para facilitar e garantir a comparação dos valores dos ciclos correspondentes à cada função. Utiliza a criação de um BigInter, que converte determinada string em seu valor binário de acordo com a raiz desejada, no caso 16 para hexadecimal.

```
package br.com.univali.cpicalc;

import java.io.File;
import java.io.FileNotFoundException;
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class BinaryConverter {

    public List<String> hexFileToBinaryArray(File hexFile) {
        final List<String> binaryArray = new ArrayList<>();
        try {
            Scanner scanner = new Scanner(hexFile);
            while (scanner.hasNext()) {
                String bitString = new BigInteger(scanner.nextLine(),
16).toString(2);
                while (bitString.length() < 32) {
                    bitString = "0".concat(bitString);
                }
                binaryArray.add(bitString);
            }
        } catch (FileNotFoundException fileNotFoundException) {
            System.out.println("Arquivo não encontrado");
        }
        return binaryArray;
    }
}
```

Classe CpiCalculator

É a parte mais importante do programa, é responsável em ler cada linha do hexadecimal e comparar o opcode com o valor registrado no HashMap responsável por registrar cada ciclo gasto para determinada operação. Caso não haja valor registrado no HashMap, o programa adverte qual opcode está faltando, assim facilitando a manutenção.

Por fim o mesmo mostra a quantidade de ciclos, instruções e CPI, também retornando o CPI.

```
package br.com.univali.cpicalc;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class CpiCalculator {
    private final Map<String, Integer> instructionCycles;
    private final List<String> missingInstructions;
    private final int load = 5;
    private final int store = 4;
    private final int rType = 4;
    private final int branch = 3;
    private final int jump = 3;

    public CpiCalculator() {
        missingInstructions = new ArrayList<>();
        instructionCycles = new HashMap<String, Integer>() {{
            put("000000", rType);
            put("001111", load);
            put("001101", branch);
            put("000010", jump);
            put("000011", jump);
            put("000100", branch);
            put("001000", store);
            put("000101", branch);
            put("101011", store);
            put("100011", load);
            put("001001", branch);
        }};
    }

    public float calculateCpi(List<String> bitInstructionArray) {
        float cycles = 0;
        float instructions = 0;

        for (String line : bitInstructionArray) {
            String opcode = line.substring(0, 6);
            Integer opcodeCycles = instructionCycles.get(opcode);
            if (opcodeCycles == null) {
                if (!missingInstructions.contains(opcode)) {
                    System.out.println("No cycle value for opcode " +
opcode);
                    missingInstructions.add(opcode);
                }
```

```
            } else {
                instructions++;
                cycles += opcodeCycles;
            }
        }

        System.out.println("Ciclos: " + cycles + " | Instruções: " +
instructions + " | CPI: " + cycles / instructions);
        return instructions / cycles;
    }
}
```

Classe Main

Invoca os métodos necessários para a execução do programa para cada arquivo presente na pasta assets.

```
import br.com.univali.cpicalc.BinaryConverter;
import br.com.univali.cpicalc.CpiCalculator;

import java.io.File;

public class Main {
    public static void main(String[] args) {
        BinaryConverter binaryConverter = new BinaryConverter();
        CpiCalculator cpiCalculator = new CpiCalculator();
        File dir = new File("./assets");
        for (File file : dir.listFiles()) {
            System.out.println(file.getName() + ':');

cpiCalculator.calculateCpi(binaryConverter.hexFileToBinaryArray(file));
            System.out.println();
        }
    }
}
```

2.2: Execução do Programa Java

```
programa1:
Ciclos: 436.0 | Instruções: 112.0 | CPI: 3.892857

programa2:
Ciclos: 288.0 | Instruções: 75.0 | CPI: 3.84

programa3:
Ciclos: 229.0 | Instruções: 59.0 | CPI: 3.881356
```