Universidade Tecnológica Federal do Paraná
Campus Apucarana
ARCO3A – Arquitetura e Organização de computadores

Projeto:

Mini simulador MIPS

Arnald Souza, RA: 2271923
Carlos Eduardo da Silva Ribeiro, RA: 2271931
João Pedro Neigri Heleno, RA: 2270323
Sandro Pinheiro Christe, RA: 2270404

Professor: André Luiz Tinassi D’ Amato

Apucarana – PR
2021

Resumo

Microprocessador sem estágios intertravados de pipeline ou *Microprocessor without Interlocked Pipeline Stages* (MIPS) é uma arquitetura criada em meados de 1990. Sua arquitetura é baseada em um registrador e por isso a CPU (Unidade Central de Processamento) utiliza apenas registradores para realizar suas operações aritméticas e lógicas e as suas instruções de acesso à memória só executam ou uma leitura da memória (load) ou uma escrita na memória (store).

Introdução

A linguagem C é uma das linguagens de alto nível criada que mais prosperou e que é usada com mais frequência já a algum tempo. Como afirma André Backes em seu livro. (BACKES, 2013, p.2).

> "A linguagem C é uma das mais bem-sucedidas linguagens de alto-nível já criadas e considerada uma das linguagens de programação mais utilizadas de todos os tempos. Define-se como linguagem de alto nível aquela que possui um alto nível de abstração relativamente elevado, que está mais próximo da linguagem humana do que do código de máquina."

Em C, a compilação parte do código-fonte que é a maneira em que os humanos possam entender o que está sendo feito, de modo que esse código não é entendido pelo computador. Para isso, o código-fonte é traduzido para o código de máquina. O ato de traduzir é chamado de compilação.

"**Compilação**: cada arquivo de código-fonte do seu programa é processado, sendo criado um arquivo "objeto" para cada um deles. Nessa etapa, não é gerado nenhum arquivo que o usuário possa executar. Em vez disso, o compilador produz as instruções de linguagem de máquina que correspondem ao arquivo de código-fonte compilado." (BACKES, 2013, p.14).

Para realizar a compilação do trabalho foi através do terminal do computador e ao acessá-lo foi necessário acionar o comando gcc que permite realizar a análise do programa para fazer a verificação de possíveis erros. A imagem a seguir demonstra como é o comando.

Figura 1 - Exemplificação do comando para compilação do programa.

Na exemplificação o programa.c é onde o algoritmo se encontra, -o programa.exe indicam o arquivo de saída do compilador -o arquivo executável que conterá o programa. Ao fazer isso o terminal retornará uma mensagem caso ocorra algum erro, se não houver nenhum erro é necessário executar o programa, como demonstra a figura abaixo.

Figura 2 - Exemplificação do comando para execução do programa.

Para realizar a execução de um algoritmo em C é preciso utilizar o arquivo executável gerado na etapa de compilação. Caso seja necessário pode-se passar um arquivo de parâmetro para auxiliar execução do programa, como demonstra a figura 2. E para execução do simulador será necessário passar como parâmetro um arquivo binário que conterá as instruções MIPS 32 bits a serem executadas

Objetivos

Exercitar e fixar os conhecimentos adquiridos sobre linguagem assembly e arquitetura de computadores. Reforçar o conhecimento sobre decisões de projeto e funcionamento de hardware. Exercitar conceitos sobre integração software e hardware por meio de simulação.

Descrição

Criar um simulador para uma versão do processador MIPS de 32 bits. O mesmo deverá ler um programa em código binário e executar algumas instruções.

Também deverá ter uma interface para facilitar o acompanhamento da evolução do programa e quais o estado de cada uma das instruções em execução (busca, decodificação, execução). A interface deverá mostrar qual

instrução está sendo executada em cada passo. O subconjunto de instruções que deverá ser implementado no simulador é apresentado pela figura 1.

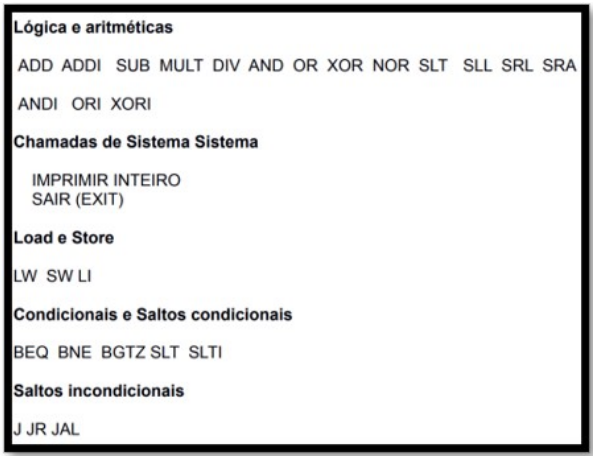

Figura 3 – Instruções a serem implantadas no MIPS 32 bits.

Resultados obtidos

No desenvolver do simulador foram adicionadas as bibliotecas e variáveis que são utilizadas ao decorrer do programa, também foi feita a leitura do arquivo binário que contém as instruções que são usadas durante a execução do algoritmo. (Figura 4).

```c
#include<stdio.h>
#include<stdint.h>
#include<stdlib.h>

#define SIZE 256

uint32_t r[32];
uint32_t ram[SIZE];
uint32_t codigo_operacao, registrador_fonte, registrador_temporario, registrador_destino, quantidade_deslocada, funcao,imediato, instrucao, endereco;

uint32_t ic, pc;

int main(int argc, const char * argv[]){

  FILE *arq;
  int numeroBits, aux;

  r[0] = 0;//$zero -> r[0]

  if(argc != 2){
    printf("Erro: Número inválido de parâmetros!\n\n");
    exit(1);
  }

  arq = fopen(argv[1], "r+b");

  if (arq == NULL){
    printf("Problemas ao abrir arquivo\n");
    exit(1);
  }
```

Figura 4 – Inicialização do algoritmo.

Em seguida foi feito a decodificação das instruções que foram armazenadas dentro da memória RAM, essa decodificação consiste em separar de acordo

com o tipo de instrução *R*, *I* ou *J,* o *OPCODE*, registrador fonte, registrador temporário, registrador destino, *SHAMT*, função, imediato e endereço de memória. Para isso foi utilizado um sistema de máscaras que realiza a operação lógica *and* entre os bits da instrução. Após aplicar a máscara ideal para o tipo de resultado necessário é feito um *shift* lógico a direita para alinhar o *LSB (Least significant bit).* (Figura 5).

```
numeroBits = fread(ram, 4, 100, arq);// Leitura do arquivo com as instruções

for(int pc = 0; pc < numeroBits; pc++){

  ic = (uint32_t)ram[pc];
  instrucao = ((ic & 0xFF000000) >> 24 | (ic & 0x00FF0000) >> 8 | (ic & 0x0000FF00) << 8 |  (ic & 0x000000FF) << 24);

  ic = instrucao;

  // Recebe o OPCODE da instrução MIPS
  codigo_operacao = (ic & 0xfc000000) >> 26;
  // Recebe os registradores das instruções
  registrador_fonte = (ic & 0x3E00000) >> 21;
  registrador_temporario = (ic & 0x1F0000) >> 16;
  registrador_destino = (ic & 0xF800) >> 11;
  // Recebe o SHAMT e o Funct, são necessária em instruções do tipo R
  quantidade_deslocada = (ic & 0x7C0) >> 6;
  funcao = (ic & 0x3F);
  // Recebe o Imediato, necessário em instruções do tipo I
  imediato = (ic & 0xffff);
  // Recebe o Endereço, para fazer o salto incondicional
  endereco = (ic & 0x3FFFFFF);
  //printf("\nopcode %01x | rs %01x | rt %01x | rd %01x | shamt %01x | funct %01x\n\n", codigo_operacao, registrador_fonte,registrador_temporario,registrador_destino,quantidade_deslocada,funcao);
```

Figura 5 –Decodificação de instruções.

Após a decodificação da instrução é utilizada uma estrutura *switch/case* para direcionar o fluxo de programa para a operação indicada pela instrução.

```
switch(codigo_operacao){

  case 0x0:
    switch(funcao){

      case 0x20: // ADD
        r[registrador_destino] = r[registrador_temporario] + r[registrador_fonte];
      break;

      case 0x22:// SUB
        r[registrador_destino] = r[registrador_temporario] - r[registrador_fonte];
      break;

      case 0x18:// MULT
        r[registrador_destino] = r[registrador_temporario] * r[registrador_fonte];
      break;

      case 0x1a:// DIV
        r[registrador_destino] = r[registrador_temporario] / r[registrador_fonte];
      break;

      case 0x24:// AND
        r[registrador_destino] = (r[registrador_temporario] & r[registrador_fonte]);
      break;

      case 0x25:// OR
        r[registrador_destino] = (r[registrador_temporario] | r[registrador_fonte]);
      break;

      case 0x26:// XOR
        r[registrador_destino] = (r[registrador_temporario] ^ r[registrador_fonte]);
      break;

      case 0x27:// NOR
        r[registrador_destino] = ~(r[registrador_temporario] | r[registrador_fonte]);
      break;
```

Figura 6 - Direcionamento de Fluxo por meio de switch/case.

Referências

Backes, André. **Linguagem C [recurso eletrônico]: completa e descomplicada / André Backes**. – Rio de Janeiro: Elsevier, 2013.

PATTERSON, David A.; HENNESSY, John L. **Organização e projeto de computadores: a interface hardware/software.** 5. ed. Rio de Janeiro: Elsevier, 2017.

STALLINGS, William**.; Arquitetura e Organização de Computadores**. 10. ed. São Paulo: Pearson Education do Brasil, 2017. ISBN 9788543020532

TANENBAUM, A. S.; AUSTIN, T. **Organização Estruturada de Computadores**. 6. ed. São Paulo: Pearson Education do Brasil, 2013. ISBN 9788581435398.

GATTO, Elaine Cecília. **Arquitetura de Conjunto e Instruções MIPS**. Disponível em: https://www.embarcados.com.br/arquitetura-de-conjunto-de-instrucoes-mips/. Acesso em: 25 de novembro de 2021.