Saulo Lordão Andrade Barros

Relatório Final

São Cristóvão, Sergipe
2013

Saulo Lordão Andrade Barros

Relatório Final

Relatório final referente ao desenvolvimento de um simulador de processador, como última nota da disciplina de Arquitetura de Computadores no curso de Graduação em Ciência da Computação.

Universidade Federal de Sergipe – UFS

Departamento de Computação

Ciência da Computação

São Cristóvão, Sergipe

2013

Resumo

O domínio dos conceitos de Arquitetura de Computadores é crucial para a formação de um Cientista da Computação. Com este trabalho objetiva-se melhorar o entendimento do aluno sobre o funcionamento de um processador além de introduzi-lo a um projeto de uma arquitetura de processador.

Palavras-chaves: arquitetura de computadores. simulador de processador.

Lista de ilustrações

Lista de tabelas

Sumário

Introdução

O presente trabalho é a conclusão da disciplina de Arquitetura de Computadores, ministrada pelo Prof. Marco Túlio Chella, na qual foi demonstrado o funcionamento e as unidades de processadores, inspirando-se principalmente na arquitetura Von Neumann.

Para consolidar tais conhecimentos, foi requisitado este trabalho como última nota da disciplina. O objetivo é desenvolver um sistema computacional com processador e memória, desde o projeto da arquitetura até a implementação de um simulador do sistema, a fim de simular fielmente o funcionamento de um processador, analisando as operações a cada ciclo de *clock*. O simulador de processador descrito se chama **Simula-CPU** (implementado em Java), baseado no processador hipotético **CPU(12237514)**, criado pelo autor para este trabalho.

1 Planejamento do simulador

1.1 Pré-Projeto

Foram poucas as mudanças conceituais entre o projeto final e o pré-projeto. As unidades continuam sendo as mesmas (e com as mesmas funções), havendo apenas a inclusão de um decodificador e a ampliação da memória, devido a uma reestruturação da palavra. Ainda houve algumas mudanças no conjunto de instruções e no funcionamento de algumas.

1.2 Unidades do sistema

1.2.1 Memória

A memória do sistema utiliza palavras de 16 bits, armazenando tanto números positivos quanto números negativos (a partir do complemento de dois). Há capacidade para 32768 palavras de 16 bits, totalizando 512KiB de memória.

1.2.2 Barramentos

Há três barramentos: Controle, Dados e Endereço. Toda a comunicação entre Processador e Memória é feita através destes três barramentos.

1.2.3 Processador

1.2.3.1 Registradores

O processador possui 8 registradores de 16 bits, implementados com o tipo *short*. Destes, quatro são acessíveis ao programador.

a) AX - Registrador acessível ao programador;

b) BX - Registrador acessível ao programador;

c) CX - Registrador acessível ao programador;

d) DX - Registrador acessível ao programador;

e) PC - Contador de Programa;

f) IR - Registrador de Instrução;

g) MAR - Memory Adress Register. Funciona como buffer dos endereços trocados entre o processador e o barramento;

h) MBR - Memory Buffer Register. Funciona como buffer dos dados trocados entre o processador e o barramento;

1.2.3.2 Unidade de Controle de Fluxo e de Movimento de Dados

Responsável por executar todas as operações que envolvam fluxo do programa ou movimento de dados. Encapsula ainda o decodificador de instrução;

1.2.3.3 Decodificador

Decodifica a instrução recebida, analisando se necessita de operando e qual o tipo de operando utilizado por ela;

1.2.3.4 ULA

Responsável por todas as operações lógico-aritméticas. Possui um registrador de estado, que armazena os estados "Operandos Iguais", "Resultado Negativo"e "Resultado Zero".

1.3 Instruções

A implementação da

Código	Mnemônico	Funcionamento
000000	ADD %AX, %BX	AX := AX + BX
000001	SUB %AX, %BX	AX := AX - BX
000010	MUL %AX, %BX	CX:DX := AX * BX
000011	DIV %AX, %BX	CX:DX := AX / BX
000100	NOT %AX	AX := AX
000101	AND %AX, %BX	AX := AX & BX
000110	OR %AX, %BX	AX := AX | BX
000111	XOR %AX, %BX	AX B̂X
00000	NOP	Consome um ciclo de clock.
00001	JMP $<end>	PC := <end>
00010	JZ	se ULA.estadoVazio() então PC := PC + 2
00011	JNZ	se não ULA.estadoVazio() então PC := PC + 2
00100	JE	se ULA.operandosIguais() então PC := PC + 2
00101	JNE	se não ULA.operandosIguais() então PC := PC + 2
00110	JNG	se ULA.resultadoNegativo() então PC := PC + 2
00111	HLT	Encerra o simulador
01000	MOV %AX, %BX	AX := BX
01001	MOV %AX, $<end>	AX := Memoria[<end>]
01010	MOV %AX, #<valor>	AX := <valor>
01011	MOV $<end>, %AX	Memoria[<end>] := AX
01100	MOV $<end>, #<valor>	Memoria[<end>] := <valor>

1.4 Implementação

O Simula-CPU foi desenvolvido na linguagem de programação Java. Para representar os dados internos foram utilizados números inteiros, enquanto a memória utiliza um vetor de inteiros. Cada unidade descrita acima foi implementada em uma classe própria, aproveitando o paradigma de Orientação a Objetos.

Os barramentos foram implementados como objetos Singleton, possuindo apenas um barramento de cada tipo, acessíveis através de métodos estáticos. Estratégias similares foram usadas nas classes de Memória e Processador. É importante notar que todas as unidades internas ao processador não são vistas fora do "pacote"onde estão, encapsulando o funcionamento destas.

Conclusão