Projeto Cpu

Carlos O. Cunha Filho

César H. Kallas

O Centro de Ciências Exatas, Ambientais e de Tecnologias

Pontifícia Universidade Católica de Campinas – Campinas – Brasil

Faculdade de Engenharia de Computação

carcunha @ yahoo.com.br RA: 00066936

cesarkallas @ cesarkallas.net RA: 02099224

Resumo:

O projeto consiste no desenvolvimento de uma aplicação em VHDL que simule uma cpu.

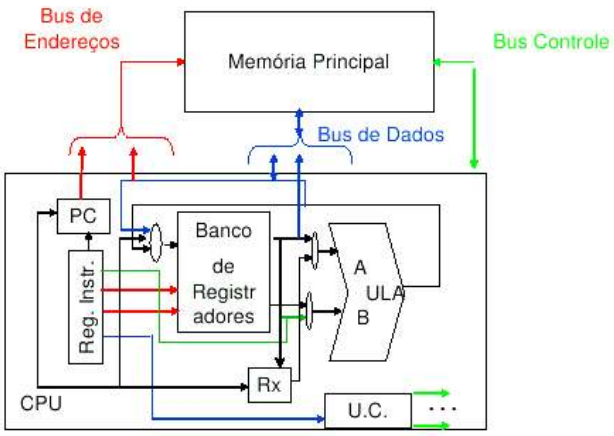
A CPU é composta por uma memória principal, um barramento de endereços e um de dados, um banco de registradores e outros componentes necessários para a executação das instruções determinadas.

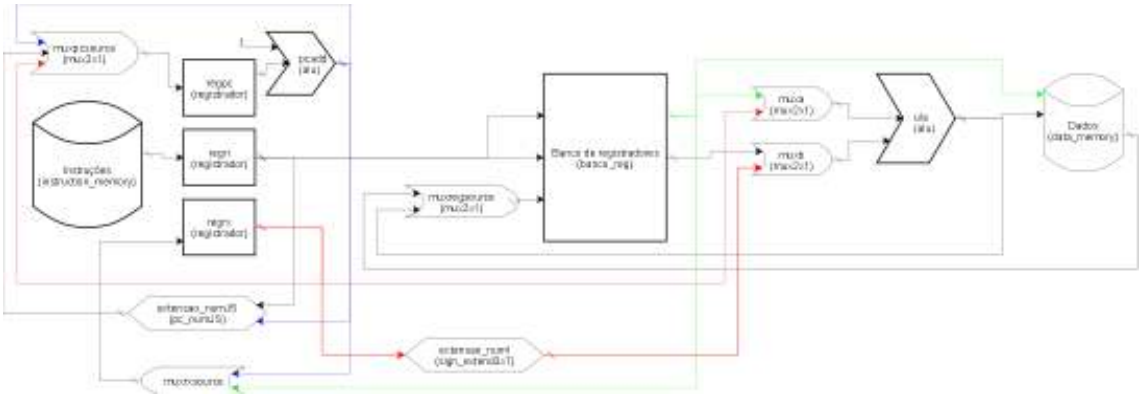
1. Propósito do projeto

O projeto consiste no desenvolvimento e implementação de uma CPU simulada, que execute várias instruções de memória, aritiméticas e de execução de código de programa.

A implementação foi feita na linguagem VHDL, usando como auxílio o software Quartus.

O projeto da CPU segue a seguinte topologia:





1.2 Descrição do projeto

O projeto possui três arquivos de programação:

- cpu1.vhdl

Possui as ligações entre todos os componentes, e as declarações de barramentos.

- data_memory.vhdl

Possui a descrição da implementação de uma estrutura de memória em forma de vetor. Tem como entrada um endereço, um sinal indicando se é leitura ou escrita, e 8 bits de entradas de dados. Tem como saída também 8 bits de dados. A saída tem sempre os dados do vetor indicado pelo endereço de entrada. Se o sinal de escrita é gerado, no endereço indicado como entrada, ele salva os 8 bits de entrada de dados.

- Instruction_memory.vhdl

Possui a descrição da implementação de uma estrutura de memória em forma de vetor. Tem como entrada um endereço de 8 bits, e como saída uma posição desse vetor, de 8 bits.

- mux2x1.vhdl e mux3x1.vhd

Possui a declaração de um multiplexador de acordo com o nome, 2x1 ou 3x1.

Para o mux2x1 ele tem um sinal de um bit para seleção da entrada que deve estar na saída, para o mux3x1 ele tem um sinal de dois bits para seleção da entrada que deve estar na saída.

- registrador.vhdl

Possui a descrição da implementação do registrador, inclusive com a declaração do buffer do conteúdo do registrador.

Nele também é descrito como funciona a interação do registrador com o barramento através do clock e de Rin.

- Banco_reg

Possui a implementação de um banco de registradores. Esse bloco é composto por 4 registradores, e dois multiplexadores para selecionar o que deve estar na saída. O sinal para seleção dos multiplexadores vem dos bits 1 e 2 da instrução (para o segundo registrador) e dos bits 3 e 4 (para o primeiro registrador). Tem como entrada um sinal reg_write, e 8 bits de dados, quando reg_write está setado, salva esses bits no registrador indicado por registrador 1.

- Pc_numJS.vhd

Pega os 4 bits mais significativos de PC+1, e concatena com os bits de 1 à 4 das instruções do tipo J (jump) ou tipo S (subrotina).

- Sign_extend3x7

Pega os 3 bits menos significativos de uma instrução do tipo I (imediato) e extende o sinal para geral um número de 8 bits que servirá como entrada de PC.

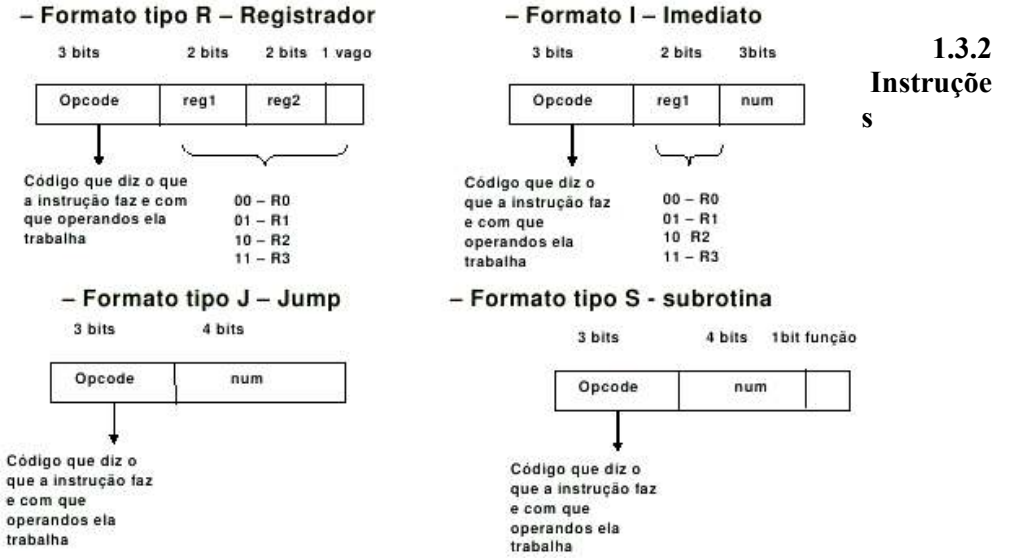
- Controle.vhd

Unidade de controle. Tem como entrada a instrução contida no registrador de instrução (regri) e através de sua máquina de estados, gera todos os sinais nos clocks certos para efetuar as operações especificadas.

1.3 Especificação

1.3.1 Instruções

A CPU possui quatro formato de instruções, de 8 bits cada, de acordo com a figura abaixo.



1.3.2 Instruções

Instrução		Opcode	Formato
MV	Rx,reg	000	Rx < Reg
LW	Reg,num	001	Reg < [Rx + num]
SW	Reg,num	010	[Rx + num] < Reg
ADD	Reg1,Reg2	011	Reg1 < Reg1 + Reg2
SUB	Reg1,Reg2	100	Reg1 < Reg1 - Reg2
AND	Reg1,Reg2	101	Reg1 < Reg1 e Reg2
JMP	num	110	PC < num
JAL	num	111 0	Rx < PC, PC < num
RET	--	111 1	PC < Rx

1.3.3 Unidade de controle e sinais

Estado	Descrição
Inicio	Inicializa o processador
S	Nesse estado, regri é escrito
S1	Seleciona o próximo estado a partir da instrução contida em regri
MV1	Faz Rx <- Reg e PC <- PC+1, volta para S
LW1	Reg <- [Rx+num] e PC <- PC+1, volta para S
SW1	[Rx+num] <- Reg e PC <- PC+1, volta para S
ADD1	Reg1 <- Reg1 + Reg2, volta para S
SUB1	Reg1 <- Reg1 – Reg2, volta para S
AND1	Reg1 <- Reg1 AND Reg2, volta para S
JMP1	PC <- num, volta para S
JAL_RET	Se o bit menos significativo for zero, estado <- JAL1, se for 1, estado <- RET1
JAL1	Rx <- PC+1, estado <- JAL2
JAL2	PC <- num, volta para S
RET1	PC <- Rx, volta para S

1.3.5 Bibliografia

Fundamentals of Digital Logic With VHDL Design – Stephen Brown and Zvonko Vranesic, Mc Graw Hill, Capítulo 7