UNIVERSIDADE FEDERAL DE VIÇOSA
CAMPUS FLORESTAL
Dupla: Adriano Marques Martins 02640 e Ruan Evangelista Formigoni 02661
Disciplina: Organização de Computadores I - CCF252
Prof. José Augusto Miranda Nacif

Trabalho Prático II
MIPS Simplificado

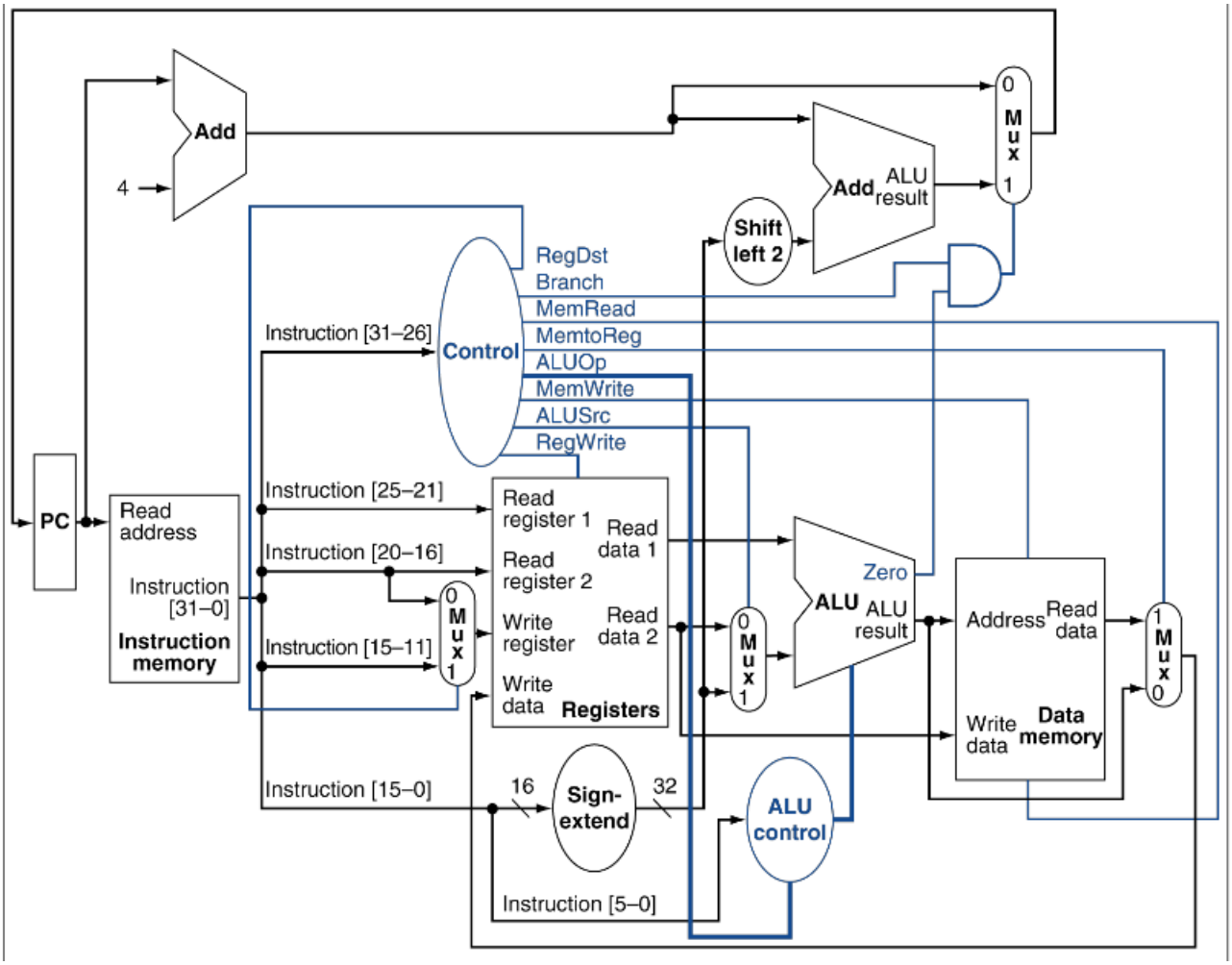
Betim, 3 de Maio de 2017

INTRODUÇÃO:

O trabalho consiste na implementação do MIPS simplificado sendo capaz de executar as instruções “lw”, “sw”, “add”, “sub”, “or”, “slt” e “beq”.

A implementação faz o uso do PC para endereçar instruções, em seguida, faz a busca da instrução na memória de instruções, faz a leitura do endereço dos registradores para determinar os dados de saída, o restante do caminho de dados varia de acordo com a instrução sendo executada.

MIPS SIMPLIFICADO:



programcounter.v : (PC)

Recebe inicialmente o endereço 0 como valor inicial e a cada subida de clock, a saída recebe a entrada.

instructionmemory.v: (Instruction Memory)

Recebe como entrada a saída do PC, e possui dentro dela um vetor de instruções, onde a saída recebe a instrução localizada no endereço indicado pelo parâmetro de entrada.

maincontrolunit.v: (control)

Foi construído um “case”, que recebe como parâmetro os 6 bits do campo “OP” da instrução gerando os sinais de controle, os valores atribuídos aos sinais de controles são pré-definidos.

multiplexor.v: (mux)

Foi definido um único arquivo contendo todos os multiplexadores do caminho de dados, eles diferem apenas no número de bits em suas entradas.

registerfile.v: (registers)

O banco de registradores possui uma matriz na qual uma linha representa um registrador de 32 bits (4 bytes). Ao receber o endereço de RS e RT, o banco de registradores tem como saída (reg1content e reg2content) seus respectivos valores. A escrita é feita somente durante um negedge.

signext.v: (sign-extend)

Replica o bit mais significativo, extendendo a entrada de 16 bits para 32 bits.

alu_control.v: (ALU control)

o módulo recebe como entrada a operação (2 bits) e o campo funct (6 bits) da instrução e processa o sinal de 4 bits em sua saída.

arithmeticlogiunit.v: (ALU)

Criado com base na ALU definia no apêndice C do material, suporta as operações lógicas “and” e “or”, aritméticas de soma e subtração, e operações de comparação.

datamemory.v: (Data Memory)

Assim como o módulo da memória de instrução, este possui um vetor contínuo de memória que armazena valores. Possui também, dois controles, um de leitura e outro de escrita que precisam ser ativados para a execução de tais operações.

sll.v: (Shift Left 2)

Realiza dois deslocamentos para a esquerda.

andGate: (Porta And)

Uma porta “and” que foi feita como um módulo para simplificar e melhor exemplificar a implementação do *top level.*

Processor.v

Módulo principal, que conecta todos os módulos com suas respectivas entradas e saídas, definindo também os tempos para as bordas de subida e descida do clock e geração do arquivo.vcd