Trabalho Prático 1: Programação Assembly para MIPS

Organização de Computadores I

Gabriel de Person Pereira de Melo - 2015042061

Lucas Fonseca Mundim - 2015042134

Marco Túlio Motta de Jesus - 2015097990

Pedro Nascimento Costa - 2015083388

1 Introdução

O Trabalho Prático 2 tem como objetivo o trabalho em linguagem *Verilog* para aprender o funcionamento de unidades funcionais de processadores. Para tal, a tarefa selecionada foi simples: implementar uma ALU, um Shifter, um Adder, um Mem e um RegBase.

Para o desenvolvimento do trabalho, foi utilizado o programa *ModelSIM* para a simulação de todo o processo do programa baseado no código.

2 Solução do Problema

O desenvolvimento do trabalho foi simples, a implementação mais complexa foi a do ALU/ULA, que conta com condições de opcode.

2.1 ULA

OPcodes 1, 2, 3 ,4, que indicam o que deve ser feito, um AND, OR, adição ou subtração e por ter quatro modalidades requer mais operações em

seu módulo, por isso elas se encontram separadas e organizadas, cada uma realizando sua devida função.

2.2 Memória

a parte da memória emprega a ideia simples de armazenamento em memória, isso é feito através de endereçamentos.

2.3 Registradores

o banco de registradores, que é uma representação dos registradores disponíveis, segue a ideia de endereçamento de cada registrador, respeitando seu tipo.

2.4 Adder

O adder, é composto por uma simples operação de adição entre dois valores, inputs somados que nos dão um output.

2.5 Shifter

shifter emprega complemento de dois, em caso de valor negativo, complementa os bits e faz o shift em si.

3 Avaliação Experimental

Os testes foram realizados no ModelSim, por programas que individualmente testam cada parte, ALU, somador, shifter, memória e banco de reg-

istradores. São eles os arquivos no formato *tb_"nome"*. Por exemplo, o teste do ALU chama-se *tp_alu.v*

3.1 Adder

No teste do Adder, temos a soma de dois números através de registradores, o resultado é monitorado em forma de inteiro.

3.2 Shifter

No teste do shifter, temos um valor armazenado sendo shiftado, com complemento de dois em caso de valor negativo, o valor é monitorado em forma binária.

3.3 ALU

No teste da ALU/ULA temos quatro testes, um para cada opcode, basta mudar o opcode da operação, foi verificado também que ocorre overflow quando deveria.

3.4 Memória

No teste da memória, foram testados armazenagem de valores e então sua impressão, verificando se os valores ficavam salvos.

3.5 Banco de Registradores

No teste do banco de registradores, foram testados armazenagem de valores e então sua impressão, verificando se os valores ficavam salvos.

4 Conclusão

O trabalho permitiu concluir de forma experimental que *HDL's* são bem mais complexas de se utilizar que quando comparadas com linguagens em mais alto nível.