LABORATÓRIO DE ARQUITETURA DE COMPUTADORES

Experimento 2

Registradores, Decodificação e Controle

(Banco de registradores, decodificação de instruções e sinais de controle)

GRUPO: 1 TURMA: A

Antônio Medeiros RA: 620521

Daniel Souza Bertoldi RA: 620548

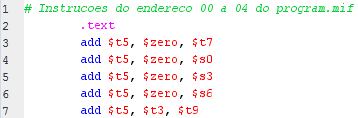
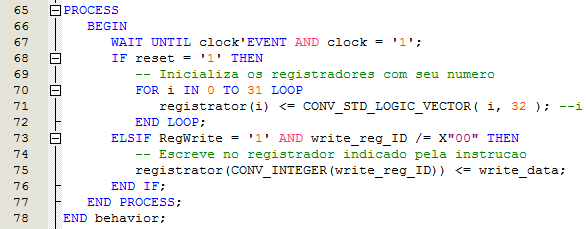
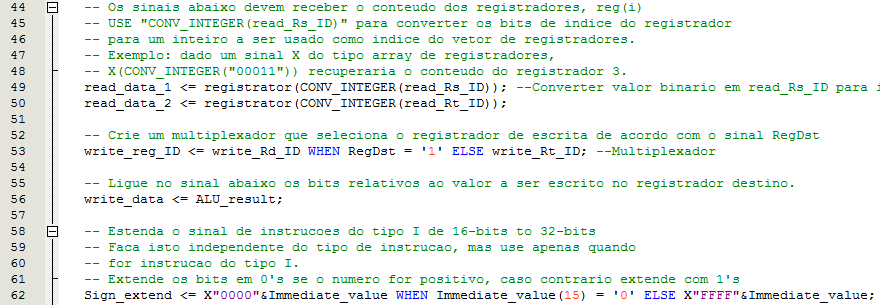
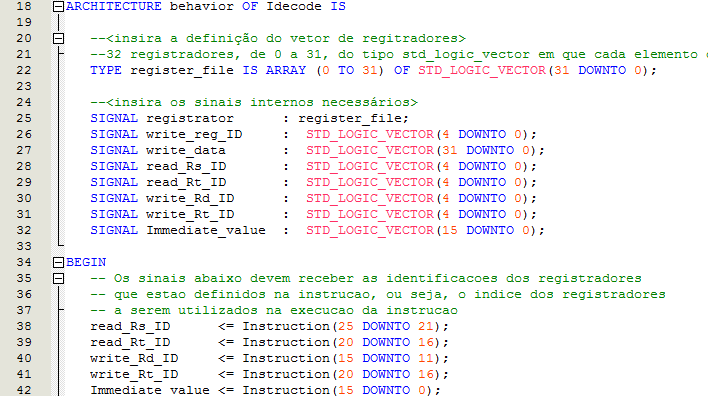
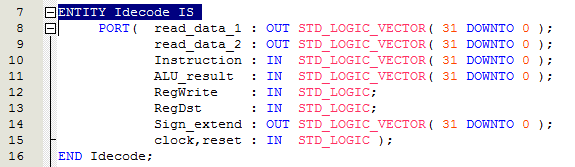
Guilherme Gonçalves RA: 620386

Lucas Buzzo RA: 620408

# Resumo (1/2 a 1 página)

Neste experimento, aprendemos a construir os componentes de Decodificador, ULA e Controle, como cada componente interage entre si e como interligá-los com a *top level*. O grupo não teve grandes dificuldades para a realização do experimento.

# Código

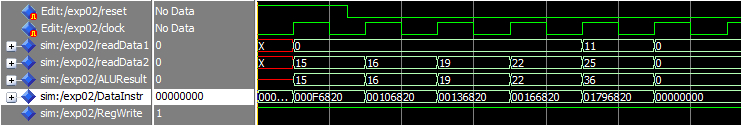


Utilizamos o próprio simulador de MIPS, o MARS para descrever a linguagem assembly das instruções contidas no program.mif. $t5 (Registrador 13) irá receber a soma de $zero (Registrador 0) com $t7 (Registrador 15) ou $s0 (Registrador 16) ou $s3 (Registrador 19) ou $s6 (Registrador 22) no caso das quatro primeiras instruções, depois $t5 irá receber a soma de $t3 (Registrador 11) com $t9 (Registrador 25).

O código de máquina da instrução criada, no caso a instrução “01796820” fica:

000000 01011 11001 01101 00000 100000

# Simulação e teste



## Discussão

Após inicializarmos a placa, deixando o reset nível lógico alto, os registradores são inicializados, como pode ser visto nas linhas 68 - 72 do código VHDL, com cada registrador recebendo seu respectivo valor. Após o primeiro pulso de clock nível alto, o Ifetch irá enviar ao DataInstr sua primeira instrução, como pode ser visto na simulação das ondas, que é “000F6820”. Convertendo para binário, teremos:

0000 0000 0000 1111 0110 1000 0010 0000

Reajustando os bits para representar o formato R da placa MIPS, temos:

Opcode Rs Rt Rd Shift Function

000000 00000 01111 01101 00000 100000

Interpretando estes números, vemos que o Opcode especifica que usaremos o formato R, e usaremos o registrador 0 (Rs), o registrador 15 (Rt), e o registrador destino 13 (Rd), não usaremos o Shift e a Função utilizada será a soma. Portanto:

A variável “read\_data\_1” lê os dados armazenados no registrador Rs após converter seu valor binário para um inteiro, para que possa acessar o índice do registrador desejado (linha 49) e quem recebe essa informação é o “readData1”, o mesmo ocorre para a variável “read\_data\_2”, que recebe o índice em binário de Rt, faz a conversão e “readData2” recebe a informação. Como pode ser visto nos próximos pulsos de clock, as primeiras quatro instruções utilizam do primeiro registrador que contém 0 e somam com o valor de outro registrador, no caso da primeira instrução, o registrador 15, que contém o valor 15. Logo, o valor do ALUResult será 15, como pode ser notado no formato de onda.

Ao sabermos isto, fica mais intuitivo interpretar os próximos pulsos de clock. Na segunda instrução, “00106820”, o Rs guarda o índice do registrador 0, e o Rt guarda o índice do registrador 16, a operação continua sendo a de soma, que será armazenada no registrador 13. A mesma explicação se repete para as próximas instruções, sendo alteradas apenas os registradores utilizados no Rt para a realização da soma.

Por fim, chegamos na quinta e última instrução do nosso program.mif, “01796820”. Como já foi explicado a leitura por trás da instrução, só nos resta perceber que o registrador usado no Rs foi o registrador 11, e no Rt usamos o registrador 25, de acordo com readData1 e readData2. Portanto, o resultado armazenado em ALUResult será 36, finalizando o experimento.

Por fim, como o Opcode sempre receberá 000000 até mesmo na inicialização, o RegWrite sempre ficará nível alto ao decorrer do experimento.