LABORATÓRIO DE ARQUITETURA DE COMPUTADORES

Experimento 3

Memória de Dados

SUMÁRIO

1 Introdução teórica

Este experimento é uma continuação do experimento anterior em direção à descrição do processador MIPS simplificado. Neste experimento vamos inserir a memória de dados e implementar as instruções LW (Load Word) e SW (Store Word). A instrução LW lê uma palavra de um endereço de memória e guarda em um registrador, enquanto a instruções SW grava em um endereço de memória uma palavra armazenada em um registrador.

1.1 Conjunto de Instruções

A Tabela 1 apresenta a estrutura dos tipos de instruções e a Tabela 2 apresenta um conjunto reduzido de instruções para o processador MIPS. As instruções LW e SW são do tipo I-format.

Tabela 1 – Tipos de Instruções

Field Size	6-bits	5-bits	5-bits	5-bits	5-bits	6-bits
R - Format	Opcode	Rs	Rt	Rd	Shift	Function
I - Format	Opcode	Rs	Rt	Address/immediate value		
J - Format	Opcode	Branch target address				

Tabela 2 – Tipos de Instruções

Mnemonic	Format	Opcode Field	Function Field	Instruction
Add	R	0	32	Add
Addi	I	8	-	Add Immediate
Addu	R	0	33	Add Unsigned
Sub	R	0	34	Subtract
Subu	R	0	35	Subtract Unsigned
And	R	0	36	Bitwise And
Or	R	0	37	Bitwise OR
Sll	R	0	0	Shift Left Logical
Srl	R	0	2	Shift Right Logical
Slt	R	0	42	Set if Less Than
Lui	I	15	-	Load Upper Immediate
Lw	I	35	-	Load Word
Sw	I	43	-	Store Word
Beq	I	4	-	Branch on Equal
Bne	I	5	-	Branch on Not Equal
J	J	2	-	Jump
Jal	J	3	-	Jump and Link (used for Call)
Jr	R	0	8	Jump Register (used for Return)

As instruções do tipo I-format somam o endereço definido na instrução ao conteúdo do registrador de origem Rs para formar o endereço de memória. O registrador Rt é o registrador de origem para a instrução SW e o registrador destino para a instrução LW.

1.2 Componentes de projeto

Este experimento adiciona mais um componente ao experimento 2: unidade de memória. Também faz alterações em outros componentes para que as instruções LW e SW possam ser implementadas.

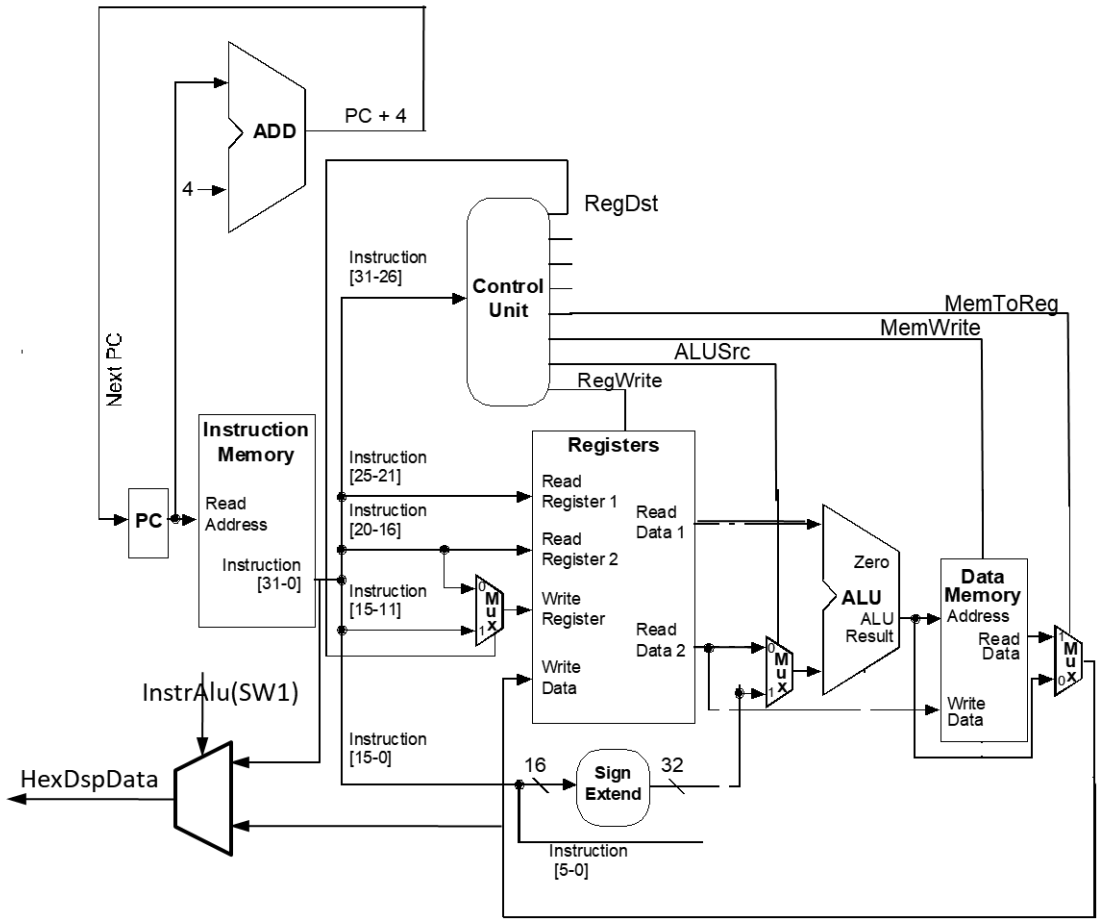

Figura 1 – Projeto MIPS parcial

2 Procedimento Preparatório

Agora vocês já sabem como declarar componentes e mapear os sinais de I/O dos componentes para interligá-los ao projeto. Então esta alteração na entidade TLE (Top Level Entity), quando é criado um novo componente, não estará mais explicita nos documentos dos experimentos, mas sempre serão necessárias.

- Crie um novo projeto de nome Exp03 e copie os arquivos disponíveis no AVA para a pasta do projeto, assim como seus arquivos .vhd do experimento 2. Altere o nome do arquivo Exp02 para Exp03, assim como o nome da entidade no arquivo.
- Selecione o menu “Assigments - Settings”, escolha a aba “Files” e clique em “Add All”, feche a janela. Isto fará com que o Quartus reconheça os arquivos copiados como parte do projeto e evitará alguns *warnings*. Com isto a lista dos arquivos poderá ser vista do lado esquerdo da janela escolhendo a guia “Files”.

Complemente os arquivos VHDL: Control, Idecode e Execute seguindo as orientações abaixo. Não se esqueça de editar o Exp03 conforme mostra a Figura 1.

- TLE (Exp03): crie o multiplexador que seleciona para gravar no registrador o dado da memória ou da ULA. Mapeie para a entrada correta do IDECODE.
- IDECODE: Edite a entrada WriteData para receber a saída do MUX criado acima, ao invés do ALUResult do experimento anterior.
- CONTROL: crie os sinais MenToReg, MenWrite e ALUSrc. Será necessário editar outros já existentes. Pense quais sinais precisam ser ativados para as instruções novas.
- EXECUTE: crie o multiplexador que seleciona entre read_data_2 e SignExtend como entrada do somador.
- DMEMORY: esta entidade é fornecida. Olhe o código para entender o tipo de memória de dados definida. O arquivo dmemory.mif define o conteúdo inicial da memória. Será necessário defini-la em Exp03 e fazer o *port map*.
- TLE (Exp03): atualize os *Port Maps* das entidades modificadas e altere o multiplexador do HexDspData para receber a saída do multiplexador criado no IDECODE. Verifique se clock <= NOT clockPB.

```
--  Configuracao da memoria
DEPTH = 256;            % valor decimal com numero de enderecos %
WIDTH = 32;             % valor decimal com numero de bits da palavra %
-- Configuracao da apresentacao
ADDRESS_RADIX = HEX;        % Opcional: endereco em formato hexadecimal %
DATA_RADIX = HEX;           % Opcional: dado em formato hexadecimal %
-- Define o conteudo inicial da memoria
Content
Begin
-- atribui valores a enderecos especificos
 00 : 8C020000;        % atribui dado hexa 8C020000 para endereco 00 %
 01 : 8C030001;        % atribui dado hexa 8C030001 para endereco 01 %
 02 : 00430820;        % atribui dado hexa 00430820 para endereco 02 %
 03: AC010003;
 04: 8C040003;
 05: 1022FFFF;
 06: 1021FFF9;
-- valores defaut, preenche memoria com zeros
 [07..FF] : 00000000;
End;
```

Figura 2 – Estado Inicial de Memória de Instrução (Program.mif)

2.1 Pin Planner

Após finalizado o projeto importe a atribuição de pinos através do menu *Assignments – Import Assignments* e escolha o arquivo Exp03.csv. Abra o PinPlanner e verifique se está tudo certo. Então configure os pinos não utilizados como “As input tri-state”.

NÃO SE ESQUEÇA DE COMPILAR O PROJETO APÓS ESTA CONFIGURAÇÃO.

3 Experimento

Implemente a descrição no simulador Quartus II e compile. Simule e teste o modelo descrito para verificar se o comportamento é como o esperado.

SIMULAÇÃO (usar RTL Simulation)

- Crie um sinal de clock com período de 100pS e duty cycle de 50%, início alto.
- Crie um sinal de reset que inicia ativo (=1) e permanece até 75pS, indo para zero até o final da simulação. Este intervalo de tempo é necessário para a simulação de leitura de memória.
- Adicione os sinais de saída que deseja monitorar.
- Use a opção de visualizar "Memory list" para acompanhar as alterações de memória e registradores.

TESTE

Somente se a simulação for bem-sucedida, faça upload para a placa após o *check list*. Para teste na placa faça:

- Ative o reset, dê um pulso de clock com o reset ativo, após desative o reset.
- O conteúdo da primeira posição de memória deve aparecer no display de LCD.
- Mude a chave SW1 para mostrar o resultado da soma (ou dado lido da memória se for uma instrução load), retorne para ver a instrução. Confira se o resultado é o esperado para aquela instrução, considerando o valor dos registradores (ou memória) naquele momento.
- Pulse o clock para ler as demais posições.

4 Resultados

Relatório

Faça um relatório e entregue até a data definida através do AVA em arquivo pdf.

- Identificação do experimento e grupo conforme modelo.
- Não é necessário o código VHDL.
- Assembly das instruções contidas em program.mif e explicação sobre qual o resultado obtido.
- Imagem do simulador com as ondas de simulação e teste.
- Explicação pontual **RELACIONANDO** o código VHDL (apresente o código necessário) e instrução da memória com as mudanças nos sinais de saída para a imagem de simulação apresentada. Esta explicação deve deixar claro que a simulação representa o funcionamento esperado.

Apresentação

Apresentar o funcionamento na placa para a professora na próxima aula.

5 Bibliografia

D'Amore, R. VHDL: Descrição e Síntese de Circuitos Digitais. LTC. 2005.

Hamblen, J.O; Hall, T.S. & Furman, M.D – Rapid Prototyping of Digital Systems: SOPC Edition. Springer, 2008.