UNIVERSIDADE FEDERAL DA PARAÍBA - UFPB
CENTRO DE INFORMÁTICA - CI
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Kelvin Brenand da Silva - 20180005270

Relatório Sobre Microprogramação da Disciplina Arquitetura de Computadores

João Pessoa
13 de agosto de 2020

Kelvin Brenand da Silva - 20180005270

Relatório Sobre Microprogramação da Disciplina Arquitetura de Computadores

Relatório referente ao desenvolvimento do projeto final da cadeira de Arquitetura de Computadores.

Prof. Dr. Ewerton Monteiro Salvador

Universidade Federal da Paraíba - UFPB
Centro de Informática - CI
Curso de Graduação em Ciência da Computação

João Pessoa
13 de agosto de 2020

Sumário

Lista de Figuras

1 Introdução

O MIPS (*Microprocessor without Interlocked Pipelined Stages*) é uma arquitetura de computadores RISC (*reduced instruction set computer*) muito utilizada para o aprendizado de microarquitetura de um modo geral. Utilizando a versão de 32bits do MIPS, neste trabalho, foi utilizada a técnica chamada microprogramação, que seria uma forma de fazer "instruções" em um nível elementar no processador, para desenvolver uma nova microinstrução. Denominada **nxtw** (*next word*), essa nova microinstrução é do tipo I, recebe como parâmetros dois registradores, efetua uma soma entre o segundo registrador e a constante 4 e armazena o resultado no primeiro registrador.

2 Fundamentação Teórica

A estrutura de uma instrução MIPS do tipo I possui, no seus 6 bits mais significativos, o campo Op (opcode), que representa a identificação da instrução. Esse opcode é único para cada instrução. Logo após têm os campos rs e rt que referenciam o primeiro e o segundo registrador, respectivamente. Nos 16 bits menos significativos fica um offset para um registrador base da memória.

Figura 1: Representação de uma instrução MIPS do tipo I.

2.1 Caminho de Dados

O caminho de dados é o componente do processador que realiza as operações contidas nas microinstruções. Para que uma microinstrução possa ser implementada, é essencial que ela seja seja possível de ser executada no caminho de dados. Isso ressalta a importância das decisões tomadas na criação da arquitetura, afinal, essas decisões irão impactar o processador como um todo.

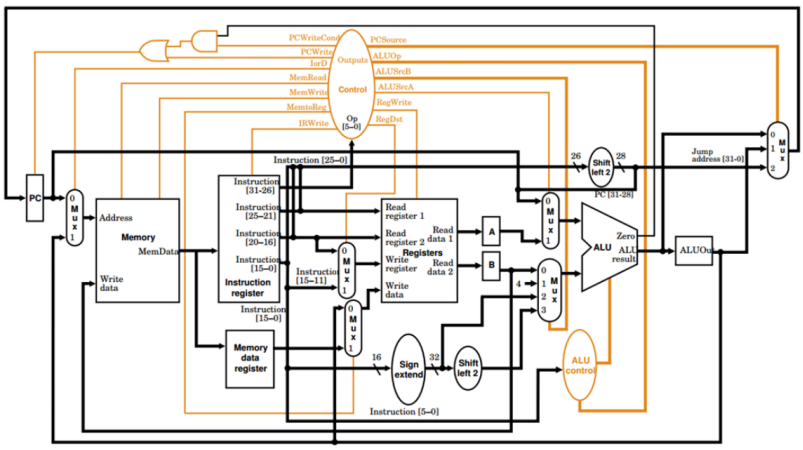

Figura 2: Representação do caminho de dados do MIPS. Na parte superior é possível visualizar a unidade de controle.

2.2 Tabelão

No “tabelão” é possível observar o conjunto de todos as operações que podem ser utilizados pelas microinstruções. É possível adicionar comandos a essa lista, entretanto, é necessário verificar se esse comando é permitido no caminho de dados. Vale ressaltar que essa lista é um recorte do conjunto total das operações possíveis no MIPS.

Nome do campo	Valores para o campo	Função do campo com valor específico
Label	Qualquer string	Usado para especificar rótulos para controlar a seqüenciação de microcódigo. Os rótulos que terminam em 1 ou 2 são usados para despachar com uma tabela de desvios indexada com base no opcode. Outros rótulos são usados como destinos diretos na seqüenciação de microinstruções. Os rótulos não geram sinais de controle diretamente mas são usados para definir o conteúdo das tabelas de despacho e gerar controle para o campo Sequencing.
ALU Control	Add	Faz com que a ALU realize uma soma.
	Subt	Faz com que a ALU realize uma subtração; isso implementa a comparação para desvios.
	Func Code	Usa o campo funct da instrução para determinar o controle da ALU.
SRC1	PC	Usa o PC como a primeira entrada da ALU.
	A	O registrador A é a primeira entrada da ALU.
SRC2	B	O registrador B é a segunda entrada da ALU.
	4	Usa 4 para a segunda entrada da ALU.
	Extend	Usa a saída da unidade de extensão de sinal como a segunda entrada da ALU.
	Extshft	Usa a saída da unidade de deslocamento como a segunda entrada da ALU.
Register Control	Read	Lê dois registradores usando os campos rs e rt do IR como os números de registrador, colocando os dados nos registradores A e B.
	Write ALU	Escreve no banco de registradores usando o campo rd do IR como o número de registrador e o conteúdo de SaídaALU como os dados.
	Write MDR	Escreve no banco de registradores usando o campo rt do IR como o número de registrador e o conteúdo de MDR como os dados.
Memory	Read PC	Lê a memória usando o PC como o endereço; escreve o resultado em IR (e no MDR).
	Read ALU	Lê a memória usando SaídaALU como o endereço; escreve o resultado em MDR.
	Write ALU	Escreve na memória usando SaídaALU como o endereço e o conteúdo de B como os dados.
PCWrite Control	ALU	Escreve a saída da ALU no PC.
	ALUOut-cond	Se a saída Zero da ALU estiver ativa, escreve em PC o conteúdo do registrador SaídaALU.
	Jump Address	Escreve no PC o endereço de desvio da instrução.
Sequencing	Seq	Escolhe a próxima microinstrução seqüencialmente.
	Fetch	Vai para a primeira microinstrução para começar uma nova instrução.
	Dispatch i	Despacha usando a ROM especificada por i (1 ou 2).

Figura 3: O "tabelão" do MIPS.

3 Objetivos Gerais

O projeto consiste em criar uma microinstrução chamada **nxtw** (*next word*) que recebe dois registradores através dos campos **rs** e **rt**, **porém não recebe um operando imediato** (os 16 bits menos significativos da instrução são sempre preenchidos com 0). O objetivo dessa instrução é somar o conteúdo do registrador indicado no campo **rs** (que deverá conter um endereço de memória) com a constante 4 (ou seja, a instrução determina o endereço da próxima palavra de 4 bytes a partir do endereço do registrador indicado em **rs**), para em seguida armazenar esse resultado no registrador indicado no campo **rt**. Um exemplo de uso dessa instrução seria **nxtw $t2, $t1**, considerando que o registrador **$t1** (campo **rs**) contém um endereço de memória, e o registrador **$t2** (campo **rt**) é aquele que deverá armazenar o endereço do registrador **$t1** após ser somado com 4 ($\$t2 = \$t1 + 4$).

4 Implementação

Para funcionar de acordo com o objetivo, a microinstrução **nxtw** *t*2,**t1**, deve inicialmente ir para o *Intruction register*. Depois disso, usando o comando *Read* do *Register Control*, os valores dos bits de 25-21 (**rs**) da microinstrução chegam no registrador **A** e a constante 4 é colocada no registrador **B**. Então é feito uma soma (*add*) na ULA e o resultado é armazenado em **ALUOut**. Para que isso funcione, é necessario que, na unidade de controle, o **ALUScrA** tenha o valor **1**, o **ALUScrB** tenha **01**, que representa o 4 (Esse multiplexador é de 2bits) e que o **ALUOp** tenha **10**, que representa uma soma (*add*). E então se dá o fim do ciclo e ocorre um **seq** para o próximo estado.

No ciclo seguinte, como as opções presentes no *Register control* não permitem a implementação da microinstrução **nxtw**, foi necessário criar um novo comando: **Write rt ALU**. Esse comando escreve no banco de registradores usando o campo **rt** do **IR** como o número de registrador e o conteúdo de **ALUOut** como como os dados. E para que esse ciclo funcione corretamente, é essencial que, na unidade de controle, os valores de **RegDst** e **RemtoReg** sejam ambos igual a **0**. Além disso, que **RegWrite** seja utilizado. No fim, ocorre um **Fetch** para o estado **0**.

A explicação acima pode ser mais facilmente compreendia com o auxílio das figuras 4 e 5.

Label	ALU Control	SRC1	SRC2	Register Control	Memory	PCWrite Control	Sequencing
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC	Extshft	Read			Dispatch 1
Mem1	Add	A	Extend				Dispatch 2
LW2					Read ALU		Seq
				Write MDR			Fetch
SW2					Write ALU		Fetch
Rformat1	Func Code	A	B				Seq
				Write ALU			Fetch
BEQ1	Subt	A	B			ALUOut-cond	Fetch
JUMP1						Jump Address	Fetch
newInst	Add	A	4	Read			Seq
				Write rt ALU			Fetch

Figura 4: Microprograma completo com com a implementação da instrução **nxtw**. Observe que a penúltima microinstrução recebe a *label* “newInst”.

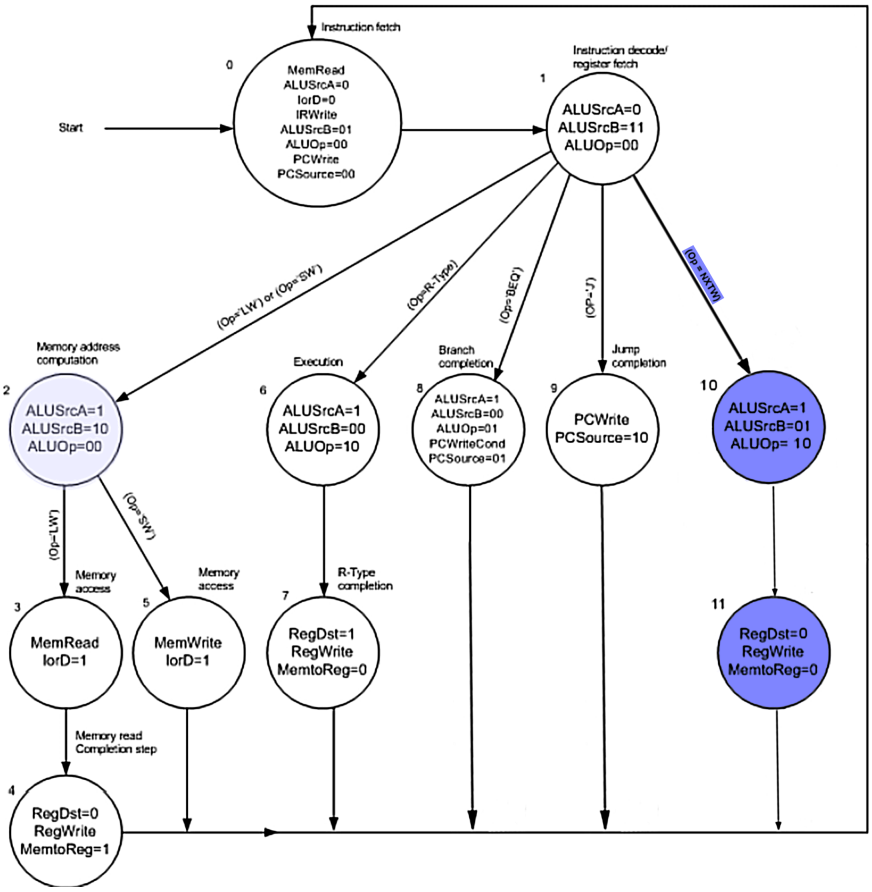

Figura 5: Maquina de estados com a microinstrução **nxtw** implementada.

5 Conclusões

Dessa forma, a partir do desenvolvimento desse projeto, foi possível realizar e entender todas as etapas necessárias para a criação de uma microinstrução, bem como entender de forma mais profunda o funcionamento da arquitetura MIPS e seu caminho de dados. Foi assim uma atividade de grande importância para a disciplina Arquitetura de Computadores, visto que os alunos puderam aplicar na prática o assunto estudado. Com isso, pode-se dizer que os objetivos desse projeto foram alcançados.

Referências

David A. Patterson, John L. Hennessy. "Organização e Projeto de Computadores - A Interface Hardware/Software". Elsevier, 3. ed., 2005

Aula 19 - Microarquitetura e Microprogramação (Parte $\frac{1}{4}$) - `https://youtu.be/6-YC5BjpnBc` – Acesso em: 12/08/2020.

Aula 20 - Microarquitetura e Microprogramação (Parte $\frac{2}{4}$) - `https://youtu.be/-ZMKSwvykPc` – Acesso em: 12/08/2020.

Aula 21 - Microarquitetura e Microprogramação (Parte $\frac{3}{4}$) - `https://youtu.be/Aza8QD98IVM` – Acesso em: 12/08/2020.

Aula 22 - Microarquitetura e Microprogramação (Parte $\frac{4}{4}$) - `https://youtu.be/gKlPlas3pT0` – Acesso em: 12/08/2020.