USP – UNIVERSIDADE DE SÃO PAULO
EESC – ESCOLA DE ENGENHARIA DE SÃO CARLOS
SEL – DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Anderson Hiroshi de Siqueira - 9313197
Paulo Augusto Alves Luz Viana, Paulo - 9313624
Marcos Paulo Souto Monteiro - 9313691

Projeto: Implementação em VHDL de um processador MIPS16e

SEL0632 – Linguagens de Descrição de Hardware
Prof. Maximiliam Luppe

São Carlos – SP
Dezembro de 2016

1. Introdução

O objetivo deste trabalho é apresentar a implementação do MIPS16e multicycle em VHDL, utilizando os conhecimentos adquiridos durante o semestre na disciplina de linguagens de descrição de hardware.

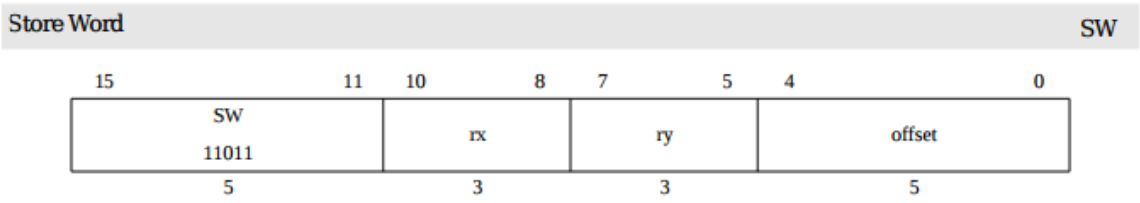
O MIPS (originalmente um acrônimo para Microprocessor without Interlocked Pipeline Stages) é uma arquitetura de conjunto de instruções (ISA) desenvolvida pela MIPS Technologies. (Wikipédia).

O projeto trata-se então de uma extensão do MIPS32, o MIPS16e é uma arquitetura de conjunto de instruções de 16 bits, sendo que neste caso foi implementado a versão multicycle.

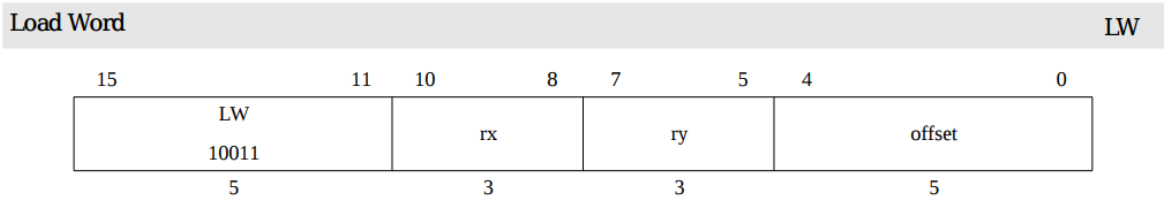
A versão multicycle significa que cada instrução recebida pelo microprocessador levará mais de uma ciclo para ser concluída, isto é, o microprocessador funcionará como uma máquina de estados, que processa e executa a instrução de forma síncrona ao longo de vários pulsos de clock, sendo uma vantagem desse tipo de implementação o barateamento de custo do projeto uma vez que esta forma de processamento permite a reutilização de componentes.

2. Instruções

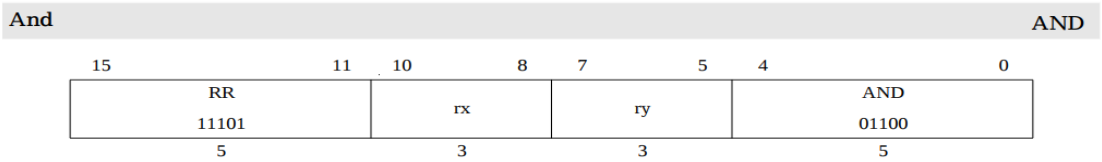
Neste projeto foram implementadas as seguintes instruções: Store Word, Load Word, Or, And, Not e Branch. Seus códigos, como definidos pela MIPS Tecnologies estão apresentados abaixo.



Tem como função armazenar o conteúdo do registrador ry no endereço de memória rx+(offset).

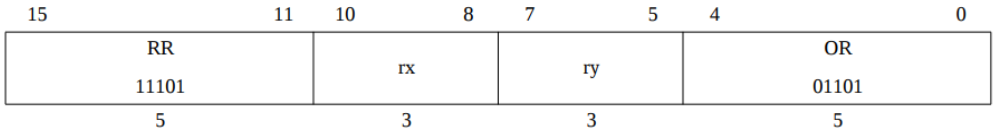


Tem como função armazenar o um dado da memória que se encontra no endereço rx+(offset) no registrador ry.



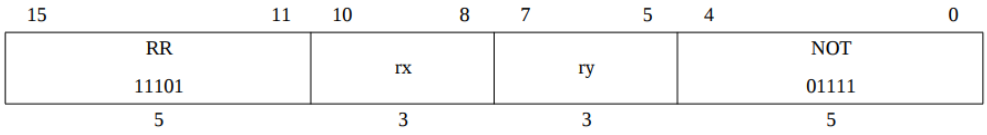
A instrução And faz a operação lógica entre os bit armazenados nos registradores rx e ry, e então armazena o resultado em rx.

Or OR



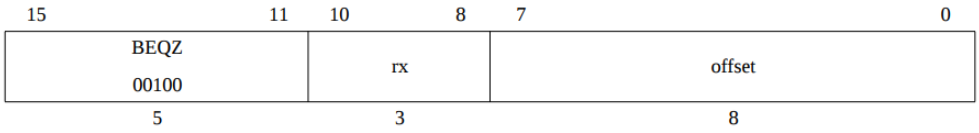
A instrução Or faz a operação lógica entre os bit armazenados nos registradores rx e ry, e então armazena o resultado em rx.

Not NOT



A instrução Not faz a operação lógica Not nos bit armazenados no registrador ry e então armazena o resultado em rx.

Branch on Equal to Zero BEQZ



O valor do offset é multiplicado por dois e tem seu sinal estendido, então este valor é somado ao endereço da última instrução.

3. Diagrama de Blocos

Esta seção é dedicada a exposição dos diagramas utilizados para implementação do MIPS16e. Os diagramas foram retirados e adaptados do livro Digital Design and Computer Architecture, Second Edition-Morgan Kaufmann (2012) [David Harris, Sarah Harris].

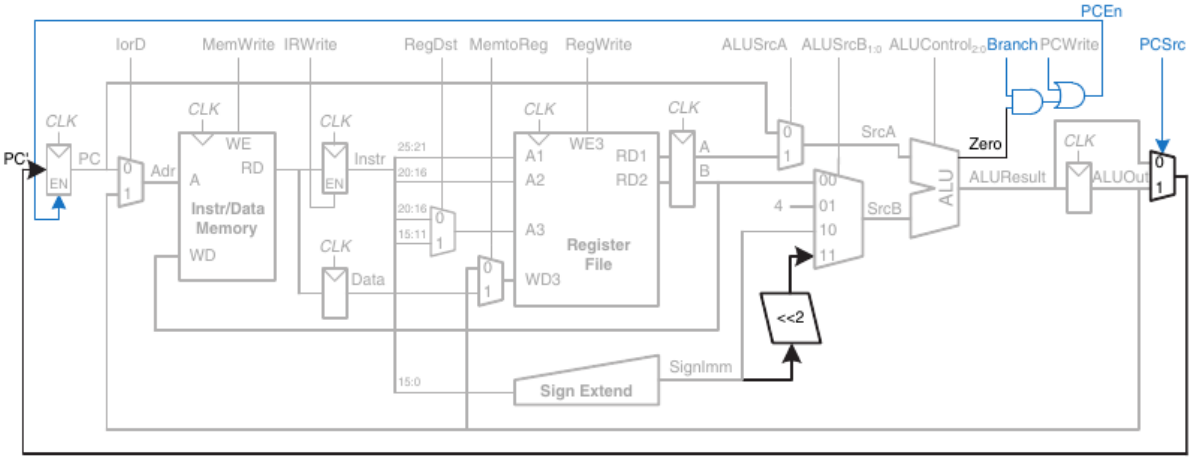
3.1. Datapath

O datapath, assim como o nome sugere, é a parte do microprocessador em que os dados são armazenados e/ou processados, nele se encontram os registradores tanto os reservados para sincronização do datapath no multicycle quanto os utilizados para armazenar dados pedidos pelo usuário.

Além disso o datapath contém a alguns outros componentes como a unidade lógica aritmética (ALU - sigla em inglês), este importante componente é responsável por fazer cálculos como operações lógicas (Or, And, Not) e também cálculos aritméticos (soma, multiplicação).

Outro componente importante é o SignInn, responsável por estender palavras de bit que vem das instruções para que elas alcancem 16 bits e possam ser operadas normalmente na ALU.

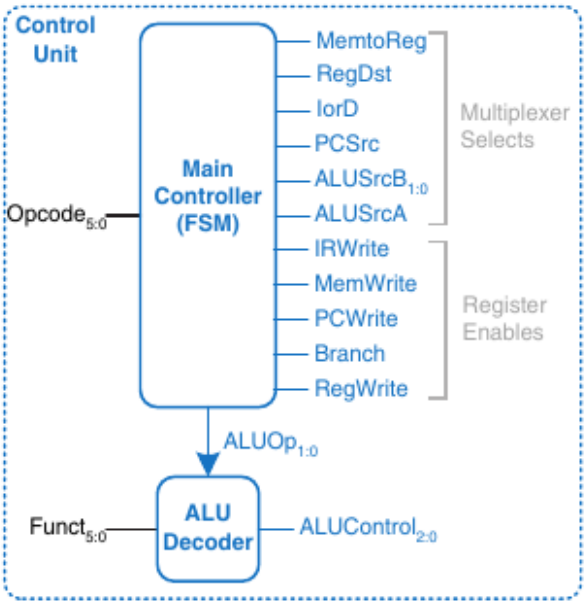
A figura abaixo foi retirado do livro mencionado no início desta seção, e foi utilizado como base para implementação do projeto. A maior adaptação se deveu ao fato deste diagrama ser de um datapath de 32 bits, enquanto o datapath implementado foi reduzido para palavras de 16 bits.



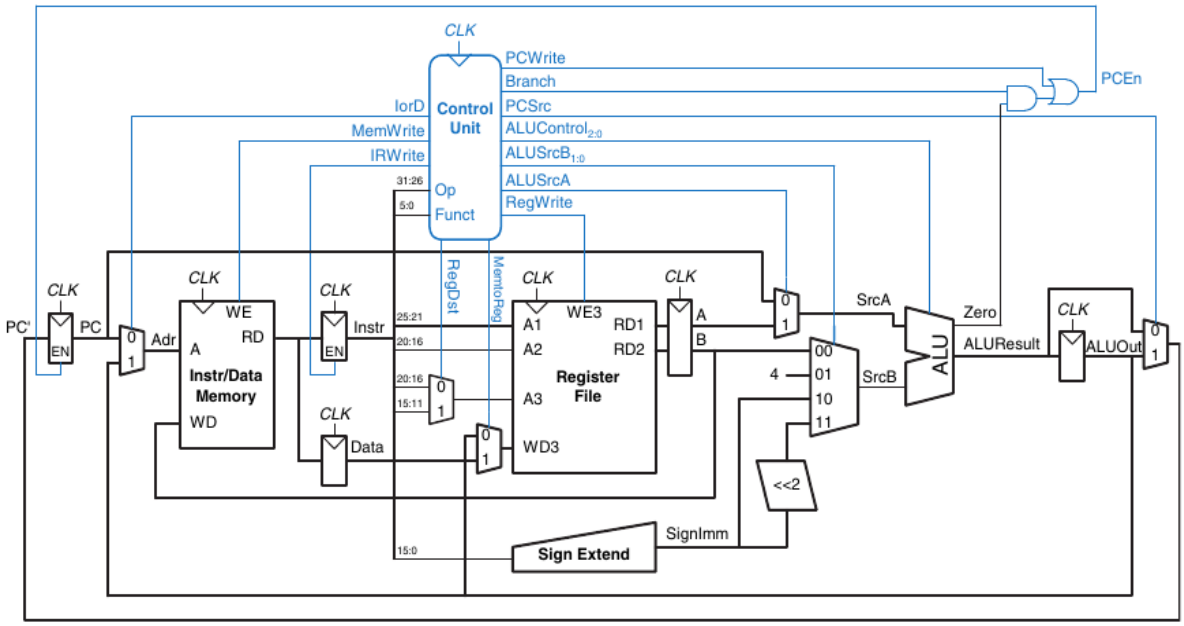
3.2. Unidade de controle

A unidade de controle é responsável por enviar sinais de controle para o datapath, controlando o fluxo de informação pelos canais certos em função da instrução recebida.

A imagem abaixo ilustra o diagrama utilizado como base para implementação da unidade de controle do projeto.



Conectando sua entradas e saídas, temos o seguinte diagrama.

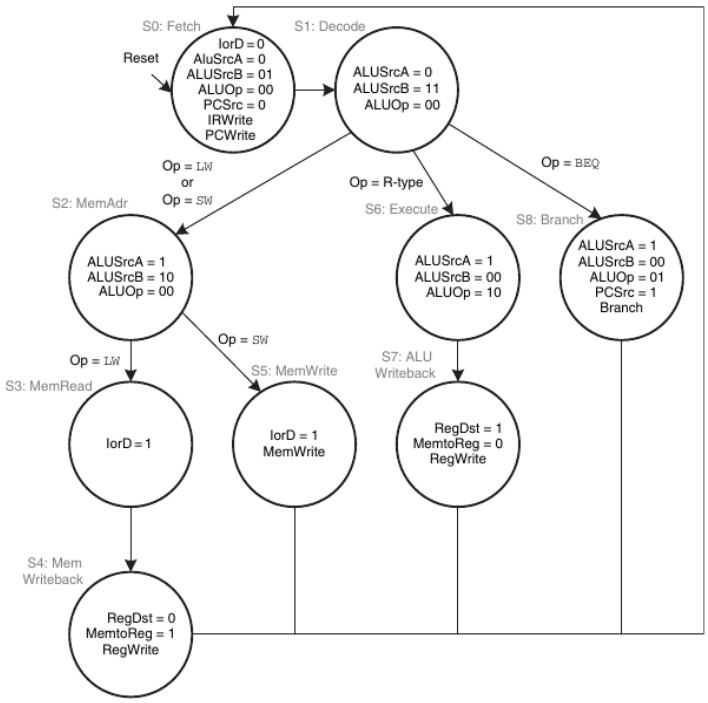


É importante lembrar que apesar da memória se encontrar no meio do diagrama, ela é um elemento externo e não faz parte do microprocessador.

3.3. Máquina de Estados

Para que o microprocessador funcione em multi-ciclo, é necessário que a unidade de controle trabalhe com uma máquina de estados, chaveando os sinais de controle e fazendo com que a informação siga o fluxo correto e execute a instrução de forma correta.

Dessa forma foi seguido o diagrama proposto no livro citado no início desta seção, sendo este reproduzido abaixo.



4. Conclusão

A linguagem VHDL se mostrou uma ferramenta muito poderosa na implementação de circuitos lógicos complexos como um microprocessador, permitindo dividir o problema em componentes menores e agrupá-los em entidades de maior nível apenas interligando os blocos e adicionando lógicas combinacionais e sequenciais simples em sua ligações.

Além disso as bibliotecas STD_LOGIC_1164 e NUMERIC_STD foram de extrema utilidade. A primeira, STD_LOGIC, resolveu problemas de como representar estados como o estado indiferente e o estado de alta impedância, dessa forma ampliando o escopo de possibilidades da linguagem. A segunda, NUMERIC_STD, diminuiu um grande problema que é o de realizar operações aritméticas com vetores de bit, além disso adiciona tipos com *signed* e *unsigned,* permitindo que o programador possa alcançar um melhor nível de abstração enquanto desenvolve e executa seu projeto de hardware.

Ainda existem ferramentas voltadas apenas para simulação, tais como o *wait*, que apesar de não ser sintetizável é essencial para gerar simulações e testar o hardware projetado, de forma tanto a modelar entradas quanto simular atrasos internos.

Apesar das poucas instruções implementadas neste microprocessador e do pequeno escopo dentro do total que o VHDL pode abranger que esta disciplina apresentou, o projeto foi capaz de demonstrar a poder da linguagem VHDL e sua praticidade, ilustrando como o conhecimento de uma linguagem de descrição de hardware é de suma importância na vida de um engenheiro eletricista, principalmente para aqueles que têm interesse em se especializar na área de sistemas digitais.