

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR MIPS 8-BITS**

**ALUNOS:**

**CRISTIANO LIMA SOUSA ROSA – 2018008595**

**REGINALDO QUEIROZ OLIVEIRA JÚNIOR- 2018028257**

**Maio de 20****21**

**Boa Vista/Roraima**



**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR MIPS 8-BITS**

**Maio de 20****2****1**

**Boa Vista/Roraima**

**Resumo**

Este trabalho aborda o projeto e implementação de um processador RISC, baseado na arquitetura de 8 bits, como projeto de conclusão da matéria DCC301 - ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES, para complemento da nota final, seguindo as instruções apresentadas em aula e dos documentos postados na plataforma Sigaa.

O processador RISC por sua vez executa instruções reduzidas, ou seja, ele quebra a instrução em várias menores e mais simples e todas assumem um tamanho padrão. Cada uma dessas instruções têm as características necessárias para que possa ser executada em apenas um ciclo de relógio.

O RISC também é amplamente utilizado em dispositivos móveis como celulares e tablets, os próprios notebooks e vídeo-games, pois são processadores menores, mais baratos e que tendem a consumir menos energia. Sem contar que atende à forte demanda por uma velocidade de processamento elevada nestes casos.

**Conteúdo**

[1 Especificação 6](#_Toc72192908)

[1.1 Plataforma de desenvolvimento 7](#_Toc72192909)

[1.2 Conjunto de instruções 7](#_Toc72192910)

[1.3 Descrição do Hardware 10](#_Toc72192911)

[1.3.1 ALU ou ULA 10](#_Toc72192912)

[1.3.2 BDRegister 10](#_Toc72192913)

[1.3.3 Clock 11](#_Toc72192914)

[1.3.4 Controle 11](#_Toc72192915)

[1.3.5 Memória de dados RAM 13](#_Toc72192916)

[1.3.6 Memória de Instruções ROM 13](#_Toc72192917)

[1.3.7 Somador 14](#_Toc72192918)

[1.3.8 And 14](#_Toc72192919)

[1.3.9 Mux\_2x1 15](#_Toc72192920)

[1.3.10 PC 15](#_Toc72192921)

[1.4 Datapath 16](#_Toc72192922)

[2 Considerações finais 17](#_Toc72192923)

**Lista de Figuras**

[Figura 1 - Especificações no Quartus 6](#_Toc444681815)

Figura 2 - Bloco simbólico do componente QALU ………………………………………………………………………………………...10

Figura 3 - Bloco simbólico do componente BDregister…………………………………………………………..10

FIGURA 4 - Detalhes das flags de controle do processador......................................................11

FIGURA 5- Bloco simbólico do componente Controle...............................................................11

FIGURA 6 - Bloco simbólico do componente MemRam............................................................12

FIGURA 7 - Bloco simbólico do componente MemRom...........................................................13

FIGURA 8 - Bloco simbólico do componente SOMADOR.........................................................14

FIGURA 9 - Bloco simbólico do componente AND..................................................................15

FIGURA 10 - Bloco simbólico do componente MUX2X1 ........................................................15

FIGURA 11 - Bloco simbólico do componente PC...................................................................16

FIGURA 12 - Datapath............................................................................................................16

**Lista de Tabelas**

[Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador MIPS 8BITS tipo R. 7](#_Toc444681822)

Tabela 2-Tabela que mostra a lista de opcodes utilizadas pelo processador mips 8 bits tipo I……………………………… 8

Tabela 3-Tabela que mostra a lista de opcodes utilizadas pelo processador mips 8 bits tipo I……………………………… 9

# Especificação

Para a elaboração do processador Mips 8 bits, foram utilizados os seguintes itens:

**Extensores:** como o nome já descreve responsáveis por aumentar o número de bits de um conjunto representando um número inteiro, aqui usamos dois extensores.

1. Extensor 2x8 para a memória
2. Extensor 4x8

**Multiplexadores:** componentes que vão implementar escolhas, recebendo duas ou mais entradas, neste caso trabalhamos apenas com duas entradas, em suma multiplexadores são conectores de n entradas para uma saída.

1. Multiplexador 2x1 Exclui Multiplexe
2. Multiplexador 2x1 Saída de 8bits
3. Multiplexador 2x1 Saída principal

**Memórias:**

1. MemRam: O componente memória RAM sou também chamado de memória de dados é o local onde os dados serão lidos e escritos.
2. MenRom: O componente memória ROM também chamado de memória de instruções é o local onde as instruções ficam gravadas.

**Somador:** responsável por incrementar o PC e realizar operações de Soma, cada entrada é um conjunto de 4 bits a saída é composta por 8 bits.

1. Somador 4 bits

**PC:** registrador interno que armazena/conta o endereço da próxima instrução a ser executada.

**ULA:** executa operações logicas aritméticas sobre par de operandos, sua saída consiste no resultado da operação.

**Unidade De Controle:** caminho para realização das escolhas dos tipos de dados a ser executado, dados tipo R, I, J.

**Porta AND:** Porta lógica com duas entradas, equivale a uma multiplicação.

**Registrador**: armazenam dados e acumulam resultados, com isso os valores não são perdidos de um ciclo para o outro.

## Plataforma de desenvolvimento

Para a implementação do processador Mips 8 bits foi utilizado a IDE:

Tabela

Descrição gerada automaticamente

Figura 1 - Especificações no Quartus

## Conjunto de instruções

O processador Mips 8 bits possuem 2 registradores: rs e rt. Assim como 3 formatos de instruções de 8 bits cada, Instruções do tipo R, I, J, seguem algumas considerações sobre as estruturas contidas nas instruções:

* **Opcode:** a operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
* **rt**: o registrador contendo o primeiro operando fonte e adicionalmente para alguns tipos de instruções (ex. instruções do tipo R) é o registrador de destino;
* **rs**: o registrador contendo o segundo operando fonte;

Tipo de Instruções:

**- Formato do tipo R:** Este formatado aborda instruções de Load (exceto *load Immediately*), Store e instruções baseadas em operações aritméticas.

Formato para escrita de código na linguagem Quantum:

|  |  |  |
| --- | --- | --- |
| Tipo da Instrução | rt | rs |

Formato para escrita em código binário:

|  |  |  |
| --- | --- | --- |
| 4 bits | 2 bits | 2 bits |
| 7-4 | 3-2 | 1-0 |
| Opcode | rt | rs |

**Visão geral das instruções do Processador MIPS 8 BITS**

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total () que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador MIPS 8 BITS.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0000 | ADD | R | Soma | **add** $S0, $S1 ,ou seja, $S0 := $S0+$S1 |
| 0001 | ADDI | R | Soma incondicional | **addi** $S0, $S1 ,ou seja, $S0 := $S0+$S1 |
| 0010 | SUB | R | Subtração | **sub** $S0, $S1 ,ou seja, $S0 := $S0 - $S1 |

**- Formato do tipo I:** Este formatado aborda instruções de Load e Store, Beq,Bne.

Formato para escrita de código na linguagem Quantum:

|  |  |  |
| --- | --- | --- |
| Tipo da Instrução | rt | Address |

Formato para escrita em código binário:

|  |  |  |
| --- | --- | --- |
| 4 bits | 2 bits | 2 bits |
| 7-4 | 3-2 | 1-0 |
| Opcode | rt | Address |

**Visão geral das instruções do Processador MIPS 8 BITS**

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total () que são distribuídos entre as instruções, assim como é apresentado na Tabela 2.

Tabela 2 – Tabela que mostra a lista de Opcodes utilizadas pelo processador MIPS 8 BITS.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0011 | Lw | I | Load | lw $t0, 10($s1) ou seja  $s0 = Memória[$s1+10] |
| 0100 | Sw | I | Store | sw $s0, 10($s1) ou seja Memória[$s1+10] = $s1 |
| 0101 | Beq | I | Desviar se igual | beq $s0, $s1, L1 ou seja  if($s0 == $s1) go to L1 |
| 0110 | Bne | I | Desviar se diferente | bne $s0, $s1, L1 ou seja  if($s0 != $s1) go to L1 |

**- Formato do tipo J:** Este formatado aborda instruções de Jump de memória

Formato para escrita de código na linguagem Quantum:

|  |  |  |
| --- | --- | --- |
| Tipo da Instrução | op | Address |

Formato para escrita em código binário:

|  |  |
| --- | --- |
| 4 bits | 4 bits |
| 7-4 | 3-0 |
| Opcode | Address |

**Visão geral das instruções do Processador MIPS 8 BITS**

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total () que são distribuídos entre as instruções, assim como é apresentado na Tabela 3.

Tabela 3 – Tabela que mostra a lista de Opcodes utilizadas pelo processador MIPS 8 BITS.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0111 | Jump | J | Desvio incodicional | j 2500 ou seja  go to 2500 |

## Descrição do Hardware

Nesta seção são descritos os componentes do hardware que compõem o processador Quantum, incluindo uma descrição de suas funcionalidades, valores de entrada e saída.

### ALU ou ULA

O componente QALU (Q Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: soma, subtração, (considerando apenas resultados inteiros) . O componente QALU recebe como entrada três valores: **A** – dado de 8bits para operação; **B** - dado de 8bits para operação e **OP** – identificador da operação que será realizada de 4bits. O QALU também possui três saídas: **zero** – identificador de resultado (2bit) para comparações (1 se verdade e 0 caso contrário); **overflow** – identificador de overflow caso a operação exceda os 8bits; e **result** – saída com o resultado das operações aritméticas.

Diagrama, Esquemático

Descrição gerada automaticamente

Figura 2 - Bloco simbólico do componente QALU gerado pelo Quartus

### BDRegister

O componente BDRegister (Banco de Registradores) tem como principal objetivo armazenar dados e acumular resultados, com isso os valores não são perdidos de um ciclo para o outro. O componente BDRegister recebe como entrada seis valores: **A** – dado de 8bits para operação; **B** - dado de 8bits para operação , além de controleEscReg vindo da UC, e um escRegi de 8bits como saída da MemRam após passar melo Mux2x1.

Tabela

Descrição gerada automaticamente com confiança média

Figura 3- Bloco simbólico do componente BDregister gerado pelo Quartus

### Clock

Também conhecido como (ciclos de clock por instrução, clocks por instrução, ou CPI) é um aspecto da performance de um processador. Vamos assumir uma arquitetura RISC clássica, com as 5 etapas seguintes:

Instruction fetch cycle (IF)

Instruction decode/Register fetch cycle (ID)

Execution/Effective address cycle (EX)

Acesso à memória (MEM)

Write-back cycle (WB)

Cada etapa requer um ciclo de clock e uma instrução passa pelas fases dos estágios seqüêncialmente. Sem pipelining, uma nova instrução é buscada na etapa 1 apenas após a instrução anterior terminar na etapa 5. Portanto, sem pipelining o número de ciclos usados para executar uma instrução é 5. Esta é a definição de CPI.

### Controle

O componente Control tem como objetivo realizar o controle de todos os componentes do processador de acordo com o opcode ... Esse controle é feito através das flags de saída abaixo:

* **DvC**.
* **Jump**.
* **EscMem**.
* **MemEsc**.
* **OrigULA**.
* **MemParaReg**
* **RegEsc:**
* **Wrt\_LRT**

Abaixo segue a tabela, onde é feita a associação entre os opcodes e as flags de controle:

figura 4 - Detalhes das flags de controle do processador.

Tabela, Calendário

Descrição gerada automaticamente

Gráfico, Gráfico de dispersão, Gráfico de caixa estreita

Descrição gerada automaticamente

Figura 5 - Bloco simbólico do componente Controle gerado pelo Quartus

### Memória de dados RAM

O componente memória RAM sou também chamado de memoria de dados é o local onde os dados serão lidos e escritos, possui 5 entradas uma de clock da Ram, uma de Endereço-de 8bits que vem do resultado de cálculo de endereço da ULA, **Escrita de dados**- de 8bits ,uma **menEsc**

E uma **memLe**r que vem diretamente da unidade de controle,

Diagrama

Descrição gerada automaticamente

Figura 6 - Bloco simbólico do componente MemRam gerado pelo Quartus

### Memória de Instruções ROM

O componente memória ROM também chamado de memória de instruções é o local onde as instruções ficam gravadas, ela recebe duas entradas uma do Pc\_endereço de 8bits ,e uma de clock.

Diagrama

Descrição gerada automaticamente

Figura 7 - Bloco simbólico do componente MemRom gerado pelo Quartus

### Somador

Basicamente o somador funciona incrementando o PC para que ele aponte para a próxima instrução, recebe como entrada dois valores: **A** – dado de 8bits para operação; **B** - dado de 8bits para operação, e uma saída de 8bits também.

Diagrama

Descrição gerada automaticamente Figura 8 - Bloco simbólico do componente SOMADOR gerado pelo Quartus

### And

Porta lógica AND (E) (também é chamada de conjunção lógica) é uma operação lógica em dois operandos que resulta em um valor lógico verdadeiro somente se todos os operados tem um valor verdadeiro. Equivale a uma multiplicação. Supondo que essa porta lógica tem duas entradas e que em uma entrada A está um bit em nível lógico alto e na outra entrada B um bit em nível lógico baixo, assim: A = 1 e B = 0. A saída S será um bit em nível lógico baixo pois, 1 x 0 = 0, logo S = 0. A conjunção lógica utiliza o símbolo "∧", e " p ∧ q " lê-se " p e q ". É a segunda operação booleana básica.

Diagrama

Descrição gerada automaticamente com confiança média

Figura 9- Bloco simbólico do componente AND gerado pelo Quartus

### Mux\_2x1

O componente MUX ou Multiplexador (Multiplexer em inglês), é utilizado para selecionar um sinal de entrada, para que este esteja presente na saída.

Diagrama

Descrição gerada automaticamente

Figura 10 - Bloco simbólico do componente MUX2X1 gerado pelo Quartus

### PC

O componente Pc é o registrador interno que armazena e conta o endereço da próxima instrução a ser executada, ele recebe duas entradas uma do sinal de clock e uma entrada Q por exemplo de 8bits, e na sua saída também saem 8bits que se dirigem para a memoriaRom que vai armazenar o endereço da instrução do Pc.

Diagrama

Descrição gerada automaticamente

Figura 11 - Bloco simbólico do componente PC gerado pelo Quartus

## Datapath

É a conexão entre as unidades funcionais formando um único caminho de dados e acrescentando uma unidade de controle responsável pelo gerenciamento das ações que serão realizadas para diferentes classes de instruções.

Diagrama, Esquemático

Descrição gerada automaticamente

Figura 12 - Datapath

# Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 8 bits denominado de Mips 8 bits, relatamos ainda que o seguinte trabalhou obedeceu todos os quesitos pedidos , vale ressaltar que todas as imagens e códigos aqui postados foram totalmente autorais, sendo assim o relatório de conclusão deste presente projeto final para obtenção de nota.