****

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR Falc-On MM**

**ALUNOS:**

**João Pedro Oliveira Silva - 1201524403**

**Talles Bezerra de Assunção - 1201524410**

**Janeiro de 2018**

**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 Falc-On MM**

**Janeiro de 2018**

**Boa Vista/Roraima**

**Resumo**

Este trabalho aborda o projeto e implementação de um processador ....

**Conteúdo**

[1 Especificação 7](#_Toc444681789)

[1.1 Plataforma de desenvolvimento 7](#_Toc444681790)

[1.2 Conjunto de instruções 8](#_Toc444681791)

[1.3 Descrição do Hardware 9](#_Toc444681792)

[1.3.1 ALU ou ULA 9](#_Toc444681793)

[1.3.2 BDRegister 9](#_Toc444681794)

[1.3.3 Clock 9](#_Toc444681795)

[1.3.4 Controle 9](#_Toc444681796)

[1.3.5 Memória de dados 10](#_Toc444681797)

[1.3.6 Memória de Instruções 10](#_Toc444681798)

[1.3.7 Somador 10](#_Toc444681799)

[1.3.8 And 10](#_Toc444681800)

[1.3.9 Mux\_2x1 10](#_Toc444681801)

[1.3.10 PC 10](#_Toc444681802)

[1.3.11 ZERO 11](#_Toc444681803)

[1.4 Datapath 11](#_Toc444681804)

[2 Simulações e Testes 13](#_Toc444681805)

[3 Considerações finais 14](#_Toc444681806)

**Lista de Figuras**

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

[Figura 2 - Bloco simbólico do componente QALU gerado pelo Quartus 8](#_Toc444681816)

[Figura 19 - Resultado na waveform. 13](#_Toc444681817)

**Lista de Tabelas**

[Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador XXXX. 7](#_Toc444681822)

[Tabela 2 - Detalhes das flags de controle do processador. 9](#_Toc444681823)

[Tabela 3 - Código Fibonacci para o processador Quantum/EXEMPLO. 12](#_Toc444681824)

# Especificação

O Falc-On MM foi desenvolvido para a disciplina de Arquitetura e Organização de Computadores da Universidade Federal de Roraima no semestre 2017.2. O Processador é multiciclo, ou seja, executa cada tipo de instrução com diferentes ciclos de clock.

## Plataforma de desenvolvimento

Para a implementação do processador Falc-On MM foi utilizado a IDE Quartus Prime Lite Edition versão 17.0 fazendo uma simulação do dispositivo xxxx da família Cyclone xx.

Toda a descrição do hardware foi feita utilizando a linguagem VHDL que é usada para facilitar o design de circuitos digitais. A figura 1 mostra as especificações do projeto.



Figura 1 - Especificações no Quartus

## Conjunto de instruções

O processador Falc-On MM possui 4 registradores: S0, ... e SN. Assim como 13 formatos de instruções de 8 bits cada, Instruções do **tipo R, tipo I e tipo 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;
* **Reg1**: o registrador contendo o primeiro operando fonte e adicionalmente para alguns tipos de instruções (ex. instruções do tipo R) é o registrador de destino;
* **Reg2**: o registrador contendo o segundo operando fonte;

### Instruções do tipo R:

Abrange instruções de carregamento e gravação de dados na memória primária e instruções baseadas em operações aritméticas. A divisão de bits está descrita na tabela 1 abaixo.

Tabela 1 – Formato de instruções do tipo R

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Registrador 1** | **Registrador 2** |
| 7-4 | 3-2 | 1-0 |
| 4 bits | 2 bits | 2 bits |

### Instruções do tipo J:

Abrange instruções onde é necessário fazer um salto condicional ou não entre os endereços, como um *if* ou um *goto*, por exemplo. A divisão de bits está descrita na tabela 2 abaixo.

Tabela 2 - Formato de instruções do tipo J

|  |  |
| --- | --- |
| **Opcode** | **Endereço** |
| 7-4 | 3-0 |
| 4 bis | 4 bits |

### Instruções do tipo I:

Abrange instruções de gravação de dados na memória primária e operações imediatas. A divisão de bits está descrita na tabela 3 abaixo.

Tabela 3 - Formato de instruções do tipo I

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Registrador 1** | **Registrador 2** |
| 7-4 | 3-2 | 1-0 |
| 4 bits | 2 bits | 2 bits |

### Visão geral das instruções do Processador Falc-On MM:

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total () de 16 **Opcodes (0-15)** que são distribuídos entre as instruções, assim como é apresentado na tabela 4.

Tabela 4 – Tabela que mostra a lista de Opcodes utilizadas pelo processador Falc-On MM.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0000 | ADD | R | Soma | Add $r1, $r2 -> $r1 = $r1 + $r2; |
| 0001 | SUB | R | Subtração | Sub $r1, $r2 -> $r1 = $r1 - $r2; |
| 0010 | ADDI | I | Soma imediata | Addi $r1, XX -> $r1 = $r1 + 000000XX; |
| 0011 | MULT | R | Multiplicação | Mult $r1, $r2 -> $r1 = $r1 \* $r2; |
| 0100 | AND | R | AND lógico | And $r1, $r2 -> $r1 = $r1 and $r2; |
| 0101 | OR | R | OR lógico | Or $r1, $r2 -> $r1 = $r1 or $r2; |
| 0110 | NOT | R | NOT lógico | Not $r1, $r2 -> $r1 = not $r1, $r2 é ignorado; |
| 0111 | XOR | R | XOR lógico | Xor $r1, $r2 -> $r1 = $r1 xor $r2; |
| 1000 | LW | I | Load word | Lw $r1, $r2 -> $r1 = 0($r2); |
| 1001 | SW | I | Store word | Sw $r1, $r2 -> 0($r2) = $r1; |
| 1010 | LI | I | Load imediato | Li $r1, XX -> $r1 = 000000XX; |
| 1011 | JMP | J | Jump | Jmp XXXX -> salto para XXXX; |
| 1100 | JR | J | Jump registrador | Jr $r1, $r2 -> salto para o valor em $r1, $r2 é ignorado; |

## Descrição do Hardware

Nesta seção são descritos os componentes do hardware que compõem o processador Falc-On MM, incluindo uma descrição de suas funcionalidades, valores de entrada e saída. Segue a descrição de cada um.

### ALU ou ULA

## Datapath

# Simulações e Testes

Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em especifico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador XXXX utilizaremos como exemplo o código para calcular o número da sequência de Fibonacci..

Tabela 6 - Código Fibonacci para o processador Quantum/EXEMPLO.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endereço** | **Linguagem de Alto Nível** | **Binário** | | |
| Opcode | Reg2 | Reg1 |
| Endereço | |
| Dado | | |
| 0 | **LI** $S0, 0 | 1111 | 00 | 00 |
| 1 | 00000000 | | |
| 2 | **LI** $S3, 6 | 1111 | 00 | 11 |
| 3 | 00000110 | | |
| 4 | **SW** $S3, $S0 | 0111 | 00 | 11 |
| 5 | **LI** $S1, 1 | 1111 | 00 | 01 |
| 6 | 00000001 | | |
| 7 | **LRT** $S2, $S1 | 0110 | 01 | 10 |
| 8 | **LI** $S3, 3 | 1111 | 00 | 11 |
| 9 | 00000011 | | |
| 10 | **LW** $S0, $S0 | 0101 | 00 | 00 |
| 11 | **CMPG** $S3,$S0 | 1010 | 00 | 11 |
| 12 | **JMP fim** | 1101 | 0000 | |
| 13 | 00011010 | | |
| 14 | **loop\_fib:** **LI** $S0, 1 | 1111 | 00 | 00 |
| 15 | 00000001 | | |
| 16 | **ADD** $S3, $S0 | 0010 | 00 | 11 |
| 17 | **LRT** $S0, $S2 | 0110 | 10 | 00 |
| 18 | **ADD** $S2, $S1 | 0010 | 01 | 10 |
| 19 | **LRT** $S1, $S0 | 0110 | 00 | 01 |
| 20 | **LI** $S0, 0 | 1111 | 00 | 00 |
| 21 | 00000000 | | |
| 22 | **LW** $S0, $S0 | 0101 | 00 | 00 |
| 23 | **CMPLE** $S3,$S0 | 1001 | 00 | 11 |
| 24 | **JMP loop\_fib** | 1101 | 0000 | |
| 25 | 00001110 | | |
| 26 | **Fim: DEBUG** $S2, $S2 | 0001 | 10 | 10 |

**[Todo] Descrição dos testes**

**Verificação dos resultados no relatório da simulação:** Após a compilação e execução da simulação, o seguinte relatório é exibido.



Neste ponto o processador inicia a execução das instruções, são

esperados dois ciclos de clock para que o sistema estabilize.

Estes são os pinos de saída para observação dos resultados, entre eles nós podemos citar: PC,

Memória de Instruções, ULA, Controladora e assim por diante.

Figura 19 - Resultado na waveform.

# Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 8 bits denominado de XXXX....