

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR Fla721**

**ALUNO:**

**Ramsés Messias de Oliveira Carvalho - 2017009328**

**Dezembro de 201****9**

**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 Fla721**

**Dezembro de 201****9**

**Boa Vista/Roraima**

**Resumo**

Este trabalho aborda o projeto e implementação do processador Fla721, sendo um processador de arquitetura Cisc, baseado no Mips, orientado para realizar instruções de 8bits, podendo realizar instruções de formato R, I e J. Como instruções lógicas e aritméticas, como adição e subtração, instruções de desvio condicional e incondicional, como branch equal e jump.

Esse processador foi feito com a utilização da IDE Quartus PrimeTM, e com o auxílio da linguagem de descrição de hardware conhecida como VHDL.

**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 Controle 9](#_Toc444681796)

[1.3.4 Memória de dados 10](#_Toc444681797)

[1.3.5 Memória de Instruções 10](#_Toc444681798)

[1.3.6 Somador 10](#_Toc444681799)

[1.3.7 And 10](#_Toc444681800)

[1.3.8 Mux\_2x1 10](#_Toc444681801)

[1.3.9 PC 10](#_Toc444681802)

[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 - Parte do código referente ao componente ULA8. 8](#_Toc444681816)

[Figura 3 - Parte do código referente ao componente Bancoreg.. 9](#_Toc444681817)

[Figura 4 - Parte do código referente ao componente ucontrol.. 10](#_Toc444681817)

[Figura 5 - Parte do código referente ao componente ram8bit.. 10](#_Toc444681817)

[Figura 6 - Parte do código referente ao componente ROMMemory.. 11](#_Toc444681817)

[Figura 7 - Parte do código referente ao componente Somador.. 12](#_Toc444681817)

[Figura 8 - Parte do código referente ao componente Andd.. 12](#_Toc444681817)

[Figura 9 - Parte do código referente ao componente Multip2x1.. 12](#_Toc444681817)

[Figura 10 - Parte do código referente ao componente PC.. 13](#_Toc444681817)

[Figura 11 – Datapath idealizado.. 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

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador Fla721, bem como a descrição detalhada de cada etapa da construção do processador.

## Plataforma de desenvolvimento

Para a implementação do processador Fla721 foi utilizado a IDE: Quartus Prime lite, versão 18.1.0 e o simulador ModelSim Altera.

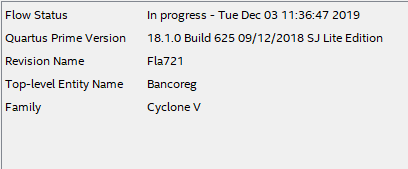


Figura 1 - Especificações no Quartus

## Conjunto de instruções

O processador Fla721 possui 4 registradores: $S0, $S1. Assim como 3 formatos de instruções de 8 bits cada, Instruções do **tipo R,I e 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;

Tipo de Instruções:

**- Formato do tipo R:** Este formatado aborda instruções lógicas e aritméticas.

Formato para escrita de código na linguagem Quantum:

|  |  |  |
| --- | --- | --- |
| Tipo da Instrução | Reg1 | Reg2 |

Formato para escrita em código binário:

|  |  |  |
| --- | --- | --- |
| 4 bits | 2 bits | 2 bits |
| 7-4 | 3-2 | 1-0 |
| Opcode | Reg2 | Reg1 |

**Visão geral das instruções do Processador Fla721:**

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 1.

Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador Fla721.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0010 | LW | I | Load Word | **lw** $S0, memória(00) |
| 0010 | ADD | R | Soma | **add** $S0, $S1 ,ou seja, $S0 := $S0+$S1 |
| 0011 | SUB | R | Subtração | **sub** $S0, $S1 ,ou seja, $S0 := $S0 - $S1 |
| 0011  0100 | SW  BEQ | I  I | Store Word  Branch Equal | **sw** $S0, memória(00)  **sw** $S0, memória(00) |
| 0101 | JUMP | J | JUMP | **Jump** endereço(1111) |

## 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 ULA8 (Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: soma e subtração. Adicionalmente o ALU8 efetua operações de comparação de valor como maior ou igual, menor ou igual, somente maior, menor ou igual. O componente ALU8 recebe como entrada três valores: **A** – dado de 8bits para operação; **B** - dado de 8bits para operação e **ULAop** – identificador da operação que será realizada de 4bits. O ALU8 também possui duas saídas: **zero** – identificador de resultado (2bit) para comparações (1 se verdade e 0 caso contrário); e **S** – saída com o resultado das operações aritméticas.

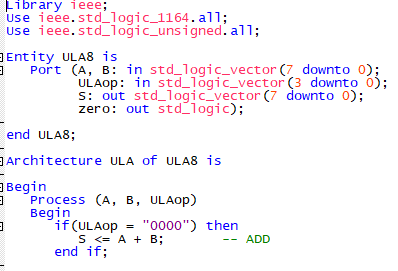


figura 2 - Parte do código referente ao componente ULA8.

### BDRegister

O componente Bancoreg (Banco de Registradores) tem como principal objetivo armazenar os dados referentes a serem operados. Possui 6 valores de entradas: **clock, regwrite** -dado de 1 bit para operação. **endreg1, endreg2** - dado referente aos valores a serem armazenados com 4 bits para operação. **entrada, writedata** – com 4 bit para armazenar o valor no endereço de memória referente.

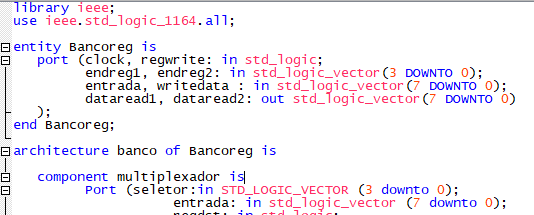


figura 3 - Parte do código referente ao componente Bancoreg.

### Controle

O componente uControl 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:

* **regdst**: XXXX.
* **branch**: XXXX.
* **memtoread**: XXXX.
* **memtoreg**: XXXX.
* **ulaop**: XXXX.
* **memwrite**: XXXX.
* **ulasrc:** XXXX.
* **regwrite**: XXXX.

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

Tabela 2 - Detalhes das flags de controle do processador.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Comando | regdst | branch | memtoread | ulaop | memwrite | ulasrc | regwrite |
| add | 1 | 0 | 1 | 0000 | 1 | 1 | 1 |
| sub | 1 | 0 | 1 | 0001 | 1 | 1 | 1 |
| lw | 0 | 0 | 1 | 0010 | 1 | 1 | 0 |

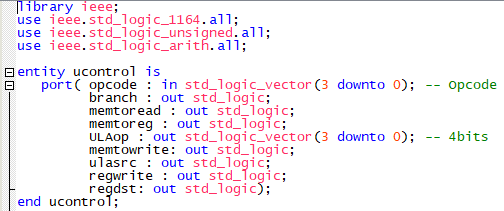


figura 4 - Parte do código referente ao componente ucontrol.

### Memória de dados

O componente ram8bit (Memória de Dados) tem como principal objetivo armazenar temporariamente os dados, após serem efetuadas as operações de soma, subtração, load ou store. O componente ram8 recebe como entrada 5 valores: **A** – dado de 8bits para operação; **B** - dado de 8bits para operação e **clock** – para situar a operação. **memwrite, memread**  - para determinar se será escrito um valor na memória ou lido um valor da memória, cada um com 1 bit. **ende** – para situar a posição dos valores armazenados, com 8 bits de sinal. **datain** – resultado (8bit) das operações realizadas na ula8; e **dataout** – saída com o resultado do valores posteriormente armazenados.

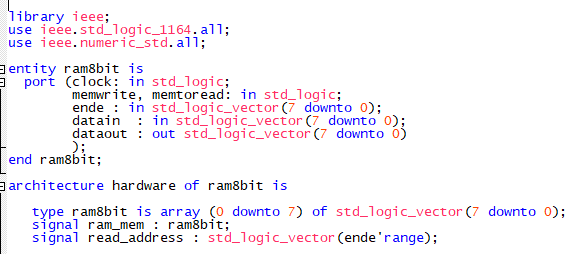


figura 5 - Parte do código referente ao componente ram8bit.

### Memória de Instruções

O componente ROMMemory (Memória de Instruções) tem como principal objetivo armazenar as instruções a serem executadas. O componente ROMMemory recebe como entrada 1 valores: **entrada** – dado de (8bits) referente a instrução a ser executada; e uma saida: **saida** – saída com o valores referentes as instruções a serem trabalhadas.

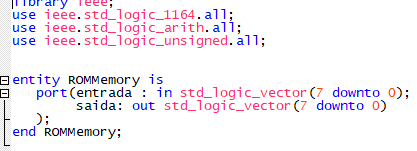


figura 6 - Parte do código referente ao componente ROMMemory.

### Somador

O componente Somador (Somador de 1 bit) tem como principal objetivo iterar as instruções a serem executadas, adicionando um 1bit em seu endereço para atingir a próxima instrução. O componente Somador recebe como entrada 2 valores: **entrada** – dado de (8bits) referente a instrução que está sendo executada; **clock** – sinal de 1 bit para gerar o ciclo necessário para a realização de cada instrução; e uma saida: **saida** – saída com o valores referentes as instruções para o próximo ciclo de clock, com 8 bits.

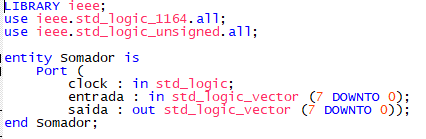


figura 7 - Parte do código referente ao componente Somador.

### And

O componente Andd (And) tem como principal objetivo fazer a operação lógica “e”, entre duas entradas. O componente Andd recebe como entrada 2 valores: **A** – dado de (1 bit) ; **B** – sinal de 1 bit; e uma saida: **x** – saída com os valores da operação and sobre as entradas.

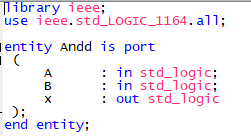


figura 8 - Parte do código referente ao componente Andd.

### Multip2x1

O componente Multip2x1 (Multiplexador) tem como principal objetivo selecionar o valor que passará para o resto do circuito. O componente Multip2x1 recebe como entrada 2 valores: **entrada1** – dado de 8 bit; **entrada2** – dado de 8 bit; **seletor** – que seleciona qual linha irá passar; e uma saida: **saida** – saída com os valores selecionados na entrada.

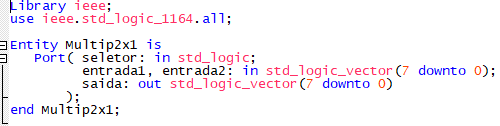


figura 9 - Parte do código referente ao componente Multip2x1.

### PC

O componente PC, tem como principal objetivo armazenar e passar para o restante do processador as instruções a serem executadas. O componente Multip2x1 recebe como entrada 2 valores: **clock** – dado de 1 bit; **input** – dado de 8 bit referente as instruções a serem executadas; e uma saida: **saida** – saída com os valores referentes as instruções a serem realizadas.

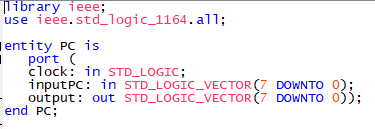


figura 10 - Parte do código referente ao componente PC.

### 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, aqui teremos uma forma idealizada do datapath.

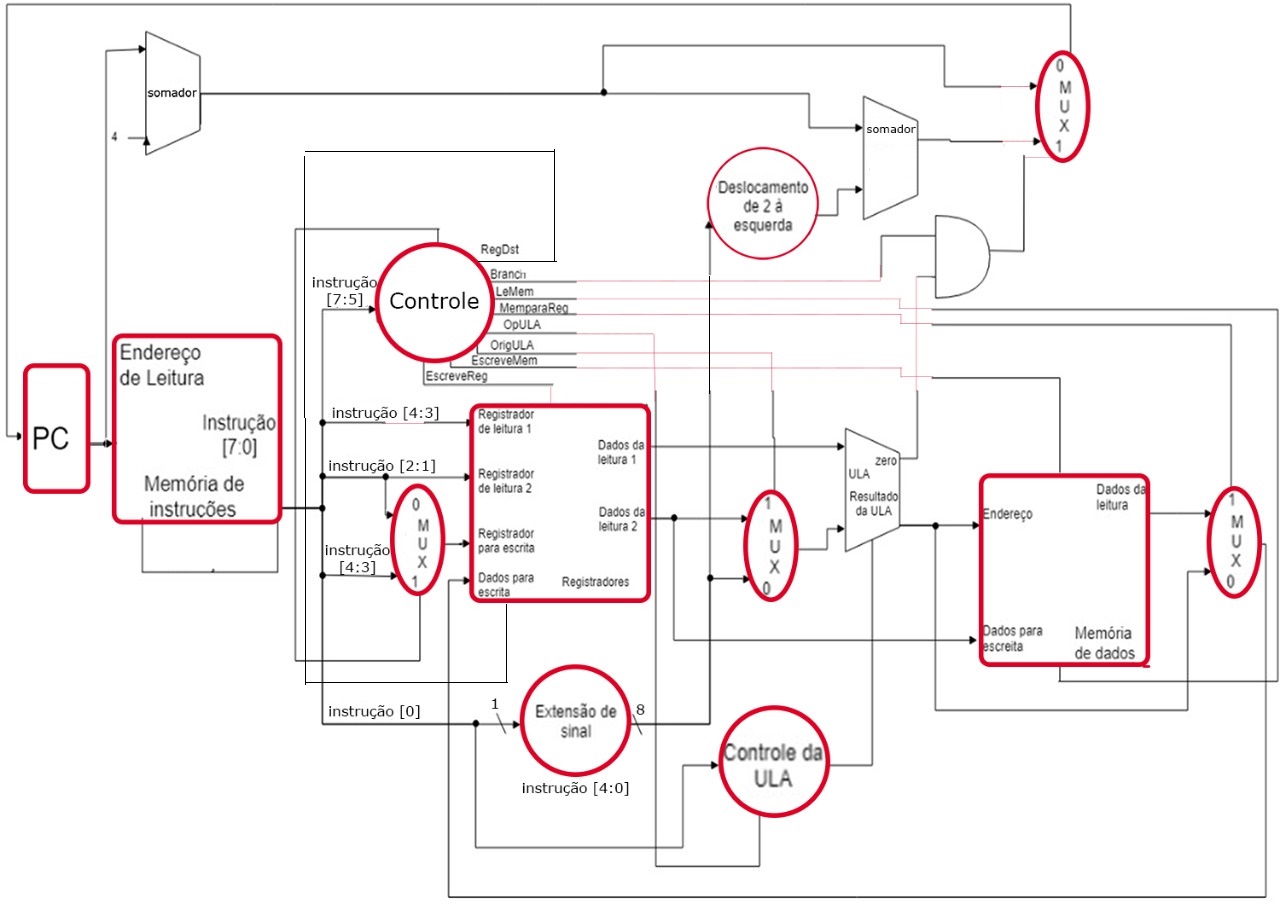


figura 11 – Datapath idealizada.

# Simulações e Testes

Como o processador não apresentou funcionamento, pularemos essa parte para sua implementação mais adiante.

# Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 8 bits denominado de Fla721. Com a orientação do professor Hebert Oliveira, foi possível realizar este feito, e fomos capaz de aprender sobre o funcionamento dos processadores assim como suas pecualiaridades