****

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR ProIA**

**ALUNOS:**

**Ibukun chife Didier Adjitche - 2016030686**

**Pedro Aleph Gomes de Souza Vasconcelos - 2016007150**

**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 ProIA**

**Janeiro de 2018**

**Boa Vista/Roraima**

**Resumo**

Esse projeto aborda a implementação de um processador baseado na arquitetura Mips.  
O processador ProIA apresenta as diversos componentes necessários para que o mesmo  
funcione e apresenta a característica de ter memórias separadas, ou seja, memória para  
dados e memória para instruções; sendo estes componentes a ULA, a unidade de  
controle, memória de dado, memória de instrução, banco de registradores e program counter.

Cada componente tem uma característica especifica, onde a unidade de controle vai  
enviar os sinais pelas flags para que de acordo com a instrução que deve operar, o PC  
apresenta a união a um somador, para que o mesmo possa fazer o cálculo da próxima  
instrução, As memórias servem para armazenar as informações, sendo uma para os  
dados que serão utilizados e outro para as instruções que devem ser executadas. A  
ULA apresenta as operações básicas, com exceção da divisão, e não trata os casos de overflow.

**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 10](#_Toc444681794)

[1.3.3 Clock 11](#_Toc444681795)

[1.3.4 Controle 11](#_Toc444681796)

[1.3.5 Memória de dados 13](#_Toc444681797)

[1.3.6 Memória de Instruções 13](#_Toc444681798)

[1.3.7 Somador **Erreur ! Signet non défini.**](#_Toc444681799)

[1.3.8 And **Erreur ! Signet non défini.**](#_Toc444681800)

[1.3.9 Mux\_2x1 **Erreur ! Signet non défini.**](#_Toc444681801)

[1.3.10 PC 14](#_Toc444681802)

[1.3.11 ZERO **Erreur ! Signet non défini.**](#_Toc444681803)

[1.4 Datapath 15](#_Toc444681804)

[2 Simulações e Testes 16](#_Toc444681805)

[3 Considerações finais 22](#_Toc444681806)

**Lista de Figuras**

[Figura 1 - Especificações no Quartus 7](#_Toc444681815)

[Figura 2 - Bloco simbólico do componente QALU gerado pelo Quartus 10](#_Toc444681816)

[Figura 19 - Resultado na waveform. **Erreur ! Signet non défini.**](#_Toc444681817)

**Lista de Tabelas**

[Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador XXXX. 9](#_Toc444681822)

[Tabela 2 - Detalhes das flags de controle do processador. 12](#_Toc444681823)

[Tabela 3 - Código Fibonacci para o processador Quantum/EXEMPLO. **Erreur ! Signet non défini.**](#_Toc444681824)

# Especificação

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

## Plataforma de desenvolvimento

Para a implementação do processador ProIA foi utilizado a IDE: Quartus Prime version 17.0.

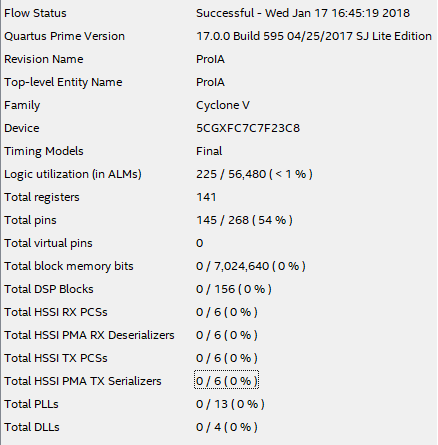


Figura 1 - Especificações no Quartus

## Conjunto de instruções

O processador ProIA possui 4 registradores: $s0, $s1, $s2 e $s3. Assim como 2 formatos de instruções de 8 bits cada, Instruções do **tipo R (operações aritméticas) Tipo I (load e store),** 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:** instruções baseadas em operações aritméticas.

Formato para escrita de código na linguagem Quantum:

|  |  |  |
| --- | --- | --- |
| OPCODE | Reg1 | Reg2 |

Formato para escrita em código binário:

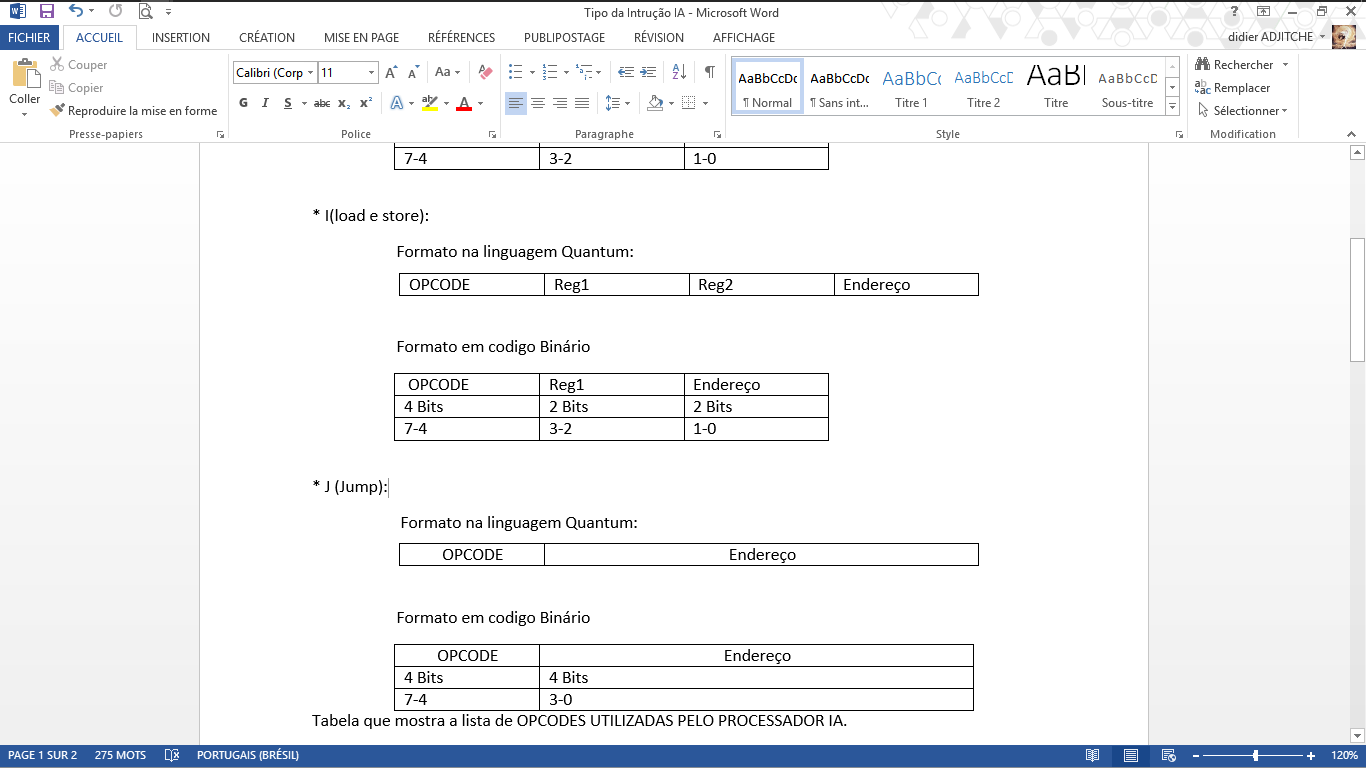


**- Formato do tipo I:** instruções de Load e Store.

Formato para escrita de código na linguagem Quantum:



Formato para escrita em código binário:



**- Instrução do tipo J (Jump)**

Formato para escrita de código na linguagem Quantum:

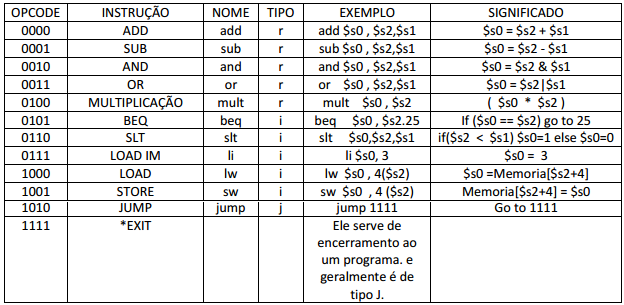
****

Formato para escrita em código binário:



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

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



## 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 ALU (Q Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: soma, subtração, divisão (considerando apenas resultados inteiros) e multiplicação. Adicionalmente o ALU efetua operações de comparação de valor como maior ou igual, menor ou igual, somente maior, menor ou igual. O componente ALU 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 duas saídas: **zero** – identificador de resultado (2bit) para comparações (1 se verdade e 0 caso contrário);e **result** – saída com o resultado das operações aritméticas; O processador não trata os casos de Overflow, os valores que ultrapassam sua capacidade de 8bits.

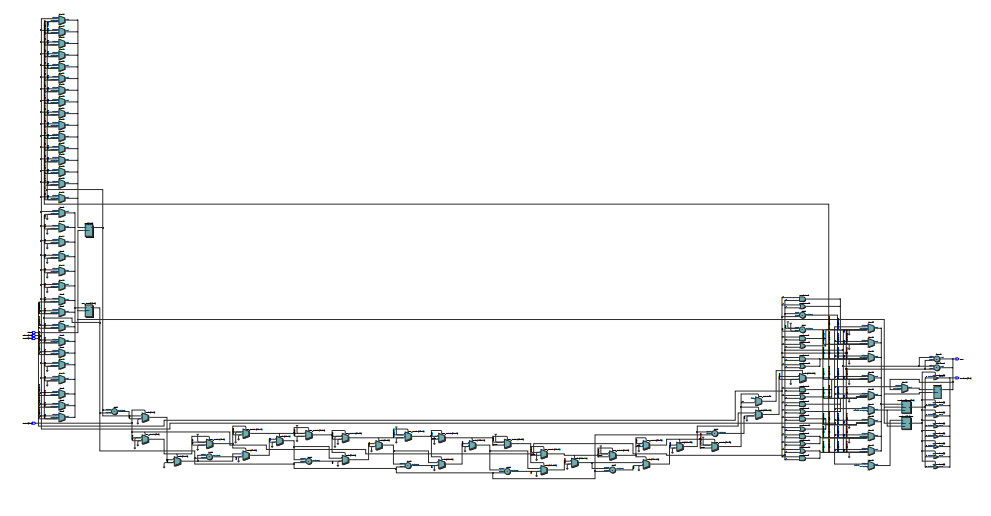
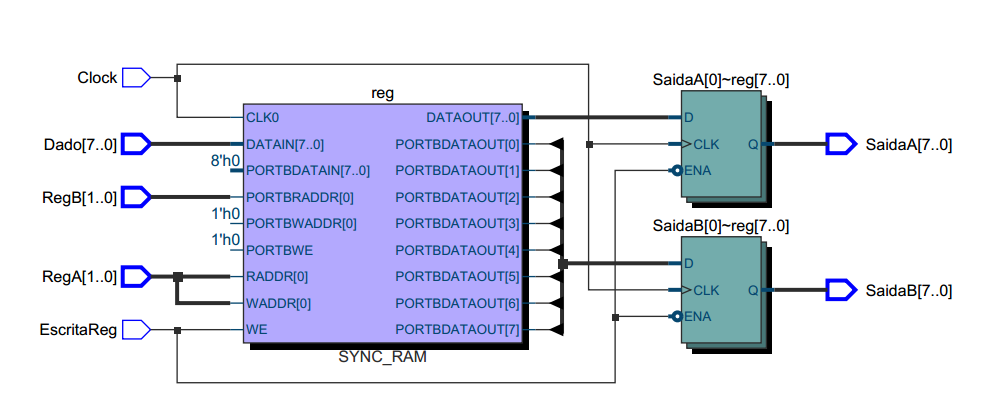


Figura 2 - Bloco simbólico do componente QALU gerado pelo Quartus

### Banco de Registradores 8bit

O banco de registradores é um componente digital composto por um conjunto de  
registradores que podem ser acessados de forma organizada. De uma maneira geral,  
podem ser executadas operações de leitura dos dados anteriormente gravados e de  
escrita de dados para modificar as informações internas. As informações que estão  
sendo processado em um determinado momento devem estar armazenadas no banco  
de registradores. Possuindo assim: dado - com 8 bits responsável pelo o  
armazenamento de dados; saida - possui 8 bits e mostra o resultado.

****

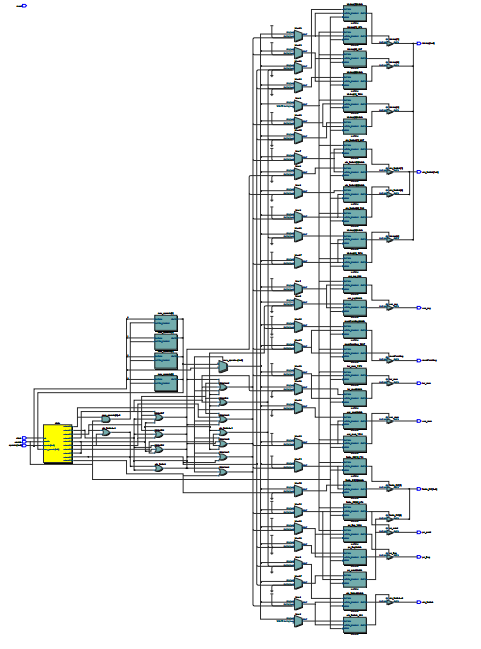
**Figura 3 - Bloco simbólico do componente banco de registradores gerado pelo Quartus**

### Clock

O clock é responsável por coordenar ações de dois ou mais componentes. Um sinal de clock oscila entre os estados alto e baixo (no estado de 1 e 0), normalmente usando um para definir algum tipo de operação.

### Unidade de Controle

A unidade de controle é responsável por enviar os comandos para as demais unidades. Estes comandos são necessários para a execução de uma determinada instrução. Para isso este componente tem como entrada o Opcode responsável pela instrução, ou seja, para a instrução de soma o opcode que será entrada da unidade de controle será o 0000. Com a entrada do opcode é ativado as flags de saída e com isso os demais componentes serão “configurados” para que a operação de soma seja executada.



**Figura 4 - Bloco simbólico da unidade de controle gerado pelo Quartus**

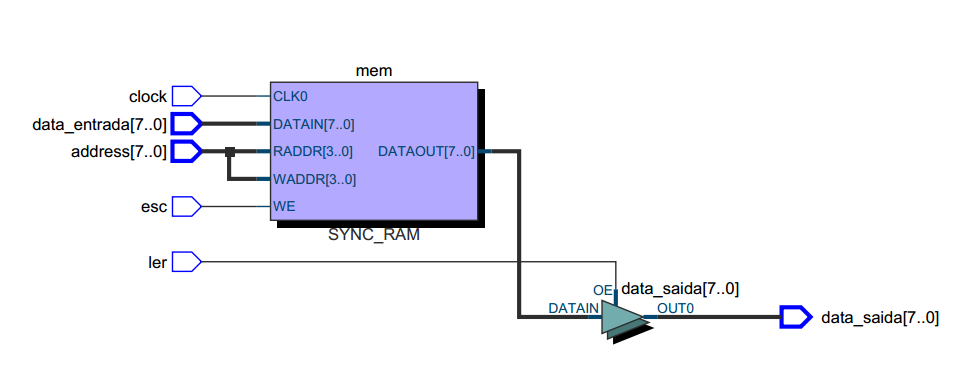
Abaixo segue a tabela, onde é feita a associação entre os opcodes e as flags de controle:

Tabela 2 - Detalhes das flags de controle do processador.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Comando | PcFlag | PcCond | EscReg | LerMem | EscMem | MemparaReg | FonteA | FonteB | PCFonte | ULAOP |
| Add | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 0000 |
| Sub | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 0001 |
| And | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 0010 |
| Or | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 0011 |
| Mult | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 0100 |
| BEQ | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 00 | 01 | 1010 |
| SLT | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 00 | 01 | 1011 |
| LOADIM | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 11 | 00 | 0111 |
| LOAD | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 10 | 00 | 1000 |
| STORE | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 10 | 00 | 1001 |
| JUMP | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 01 | 10 | 0000 |
| EXIT | Z | Z | Z | Z | Z | Z | Z | ZZ | ZZ | ZZZZ |

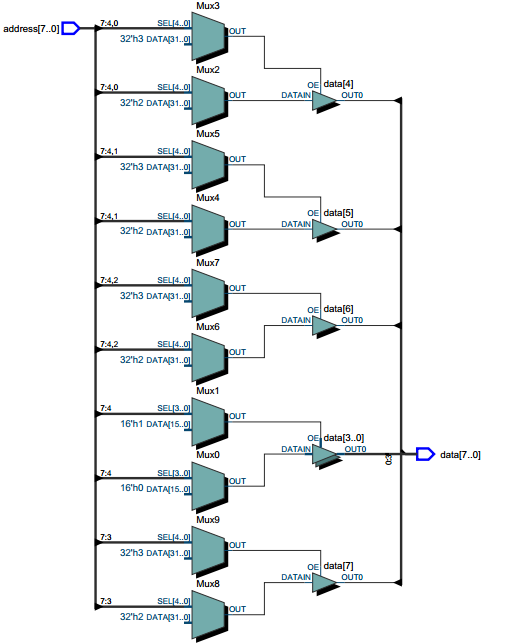
### Memória de dados

A memória de dados é usada para pelo processador para armazenar os arquivos e  
programas que estão sendo processada. A principal característica da memória de dados  
é que ela é volátil, ou seja, os dados se perdem ao reiniciar o computador. Ao ligar é  
necessário refazer todo o processo de carregamento, em que o sistema operacional e  
aplicativos usados são transferidos do HD para a memória, onde podem ser executados  
pelo processador. A memória de dados possui os seguintes componentes: clock - possui 1 bits e atua como um sinal para sincronismo; reset - 1 bits e usado para reiniciar; leitura - 1 bits responsável pela leitura do dado; escrita - 1 bits recebe os dados escritos; endereço – 8 bits ele pode armazenar dados ou buscar um determinado valor, depende da instrução;  
dado - 8 bits responsável pelo o armazenamento de dados ; saída - 8 bits, onde mostra  
o valor sinal da operação.

**Figura 4 - Bloco simbólico do componente da memória de dados gerado pelo Quartus**

### Memória de Instruções

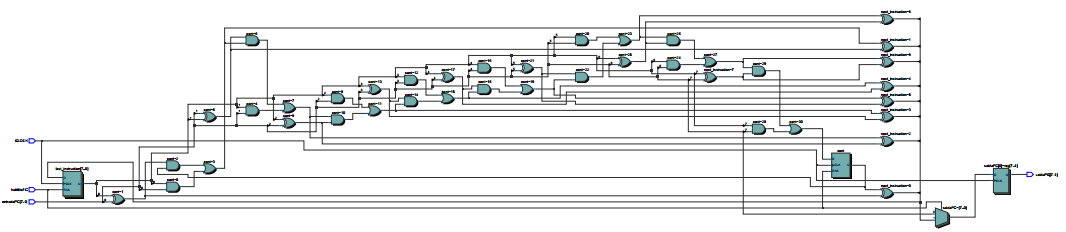
A memória de instruções é um termo genérico usado para designar as partes do  
computador ou dos dispositivos periféricos onde os dados e programas são  
armazenados. Sem uma memória de onde os processadores podem ler e escrever  
informações, não haveria nenhum computador digital de programa armazenado. Sendo  
formada por: 2 registradores - com 2 bits responsável pelo conjunto de dados armazenados; opcode - possui 4 bits para conseguir realizar determinadas tarefas; endereço - com 8 bits para armazenar dados.



**Figura 5 - Bloco simbólico do componente da memória de instruções gerado pelo Quartus.**

### PC

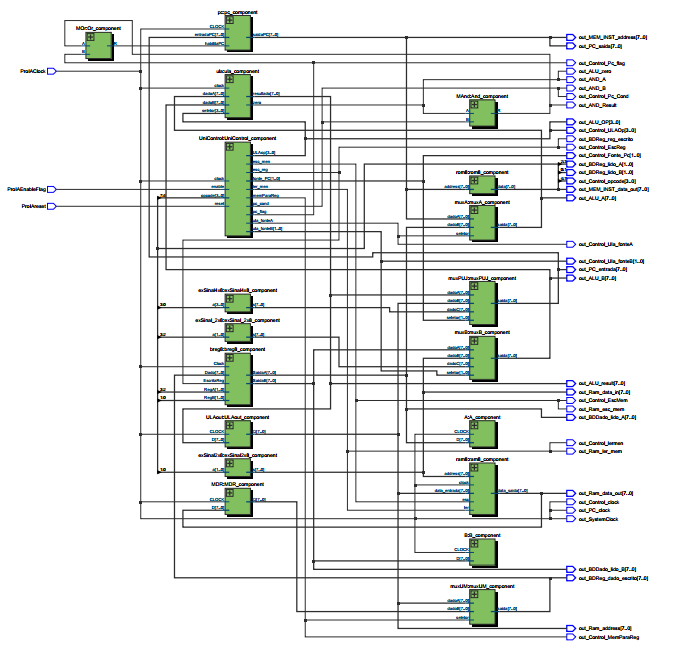
O pc (Program Counter) é a unidade responsável por armazenar instruções a ser  
executada, além disso pode ser feita algumas operações para descobrir a  
próxima instrução, pois tem um somador integrado. Sendo assim: clock - possui  
1 bits e atua como um sinal para sincronismo; reset - 1 bits e usado para reiniciar;  
flag - 1 bits e usado como interruptor (isto é, valores 1/0, ligado/desligado,  
ativo/inativo) permite otimizar as estruturas de dados; entrada - 8 bits e uma  
informação que é recebida e processada; saída - 8 bits mostra a saída do dado  
processado ; saida2 - 8 bits mostra a saída do dado processado.



**Figura 7 - Bloco simbólico do componente da pc gerado pelo Quartus**

## Datapath

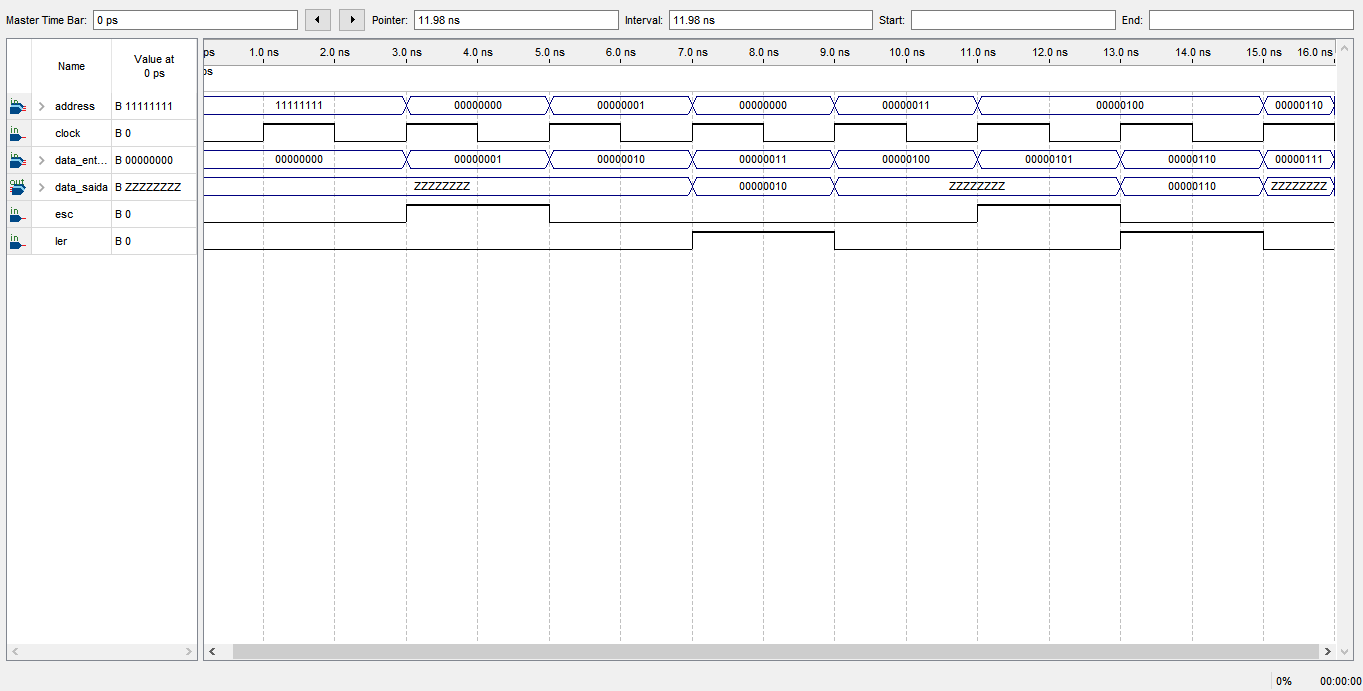
É a conexão entre as unidades funcionais formando um único caminho de dados e acrescentando uma unidade de controle responsável pelo gerenciamento das ações que serão realizadas para diferentes classes de instruções.

**Figura 8 - Bloco simbólico do componente da Datapath gerado pelo Quartus**

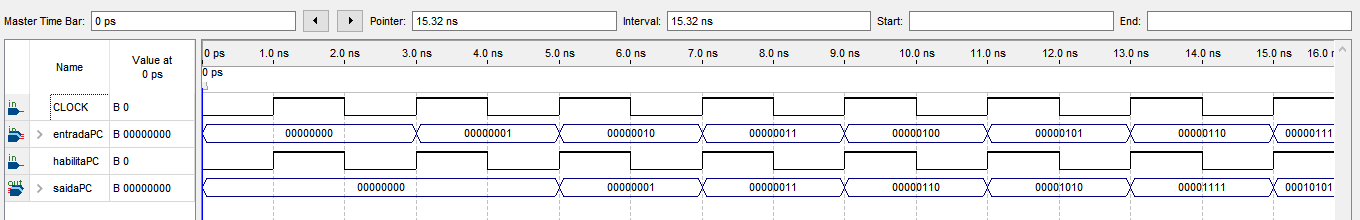
# 

# Simulações e Testes

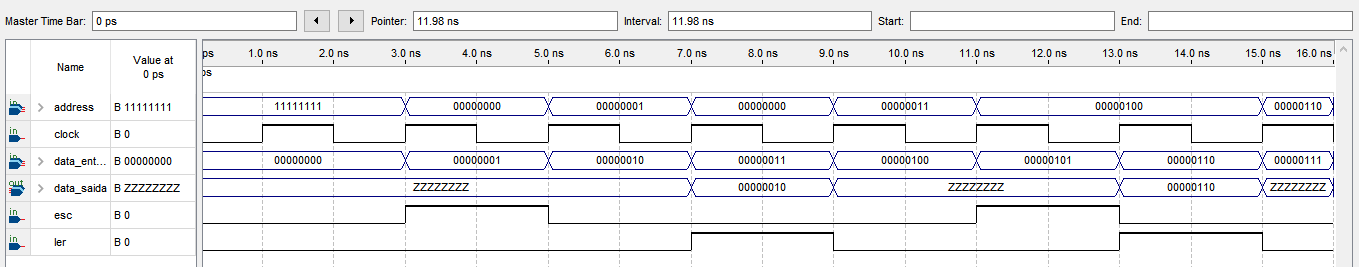
Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em especifico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador ProIA, os testes dos seguintes componentes forma realizados.



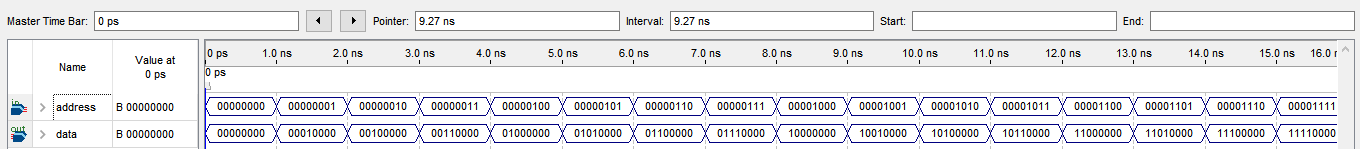
**Componente Banco de registradores 8bit**

****

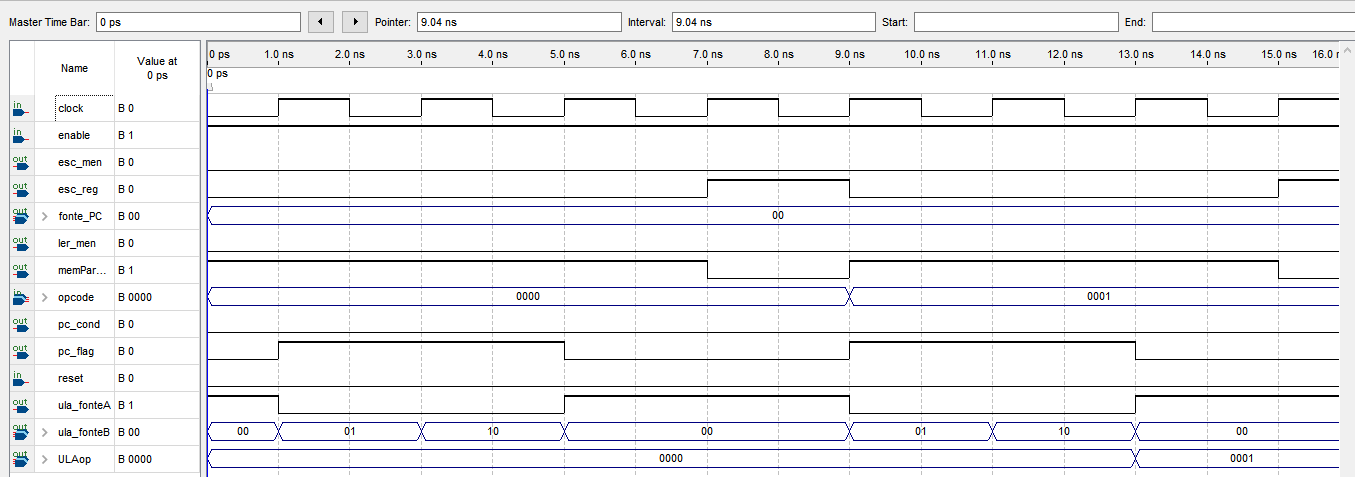
**Componente PC**

****

**Componente memória Ram 8bit**

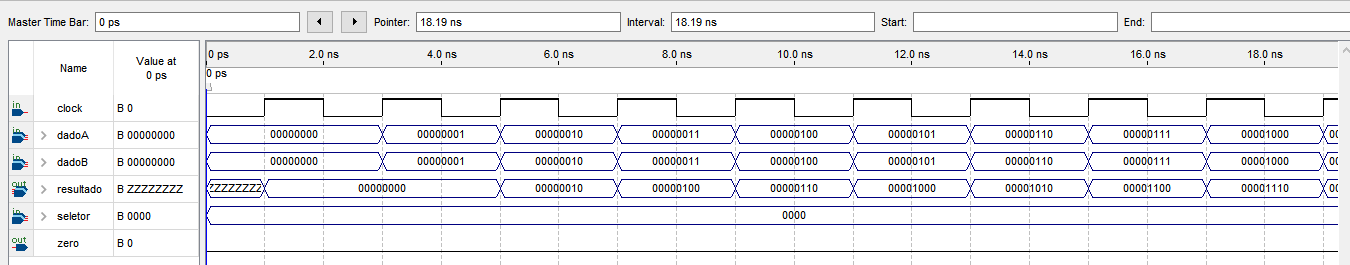
****

**Componente memória Rom 8bit**

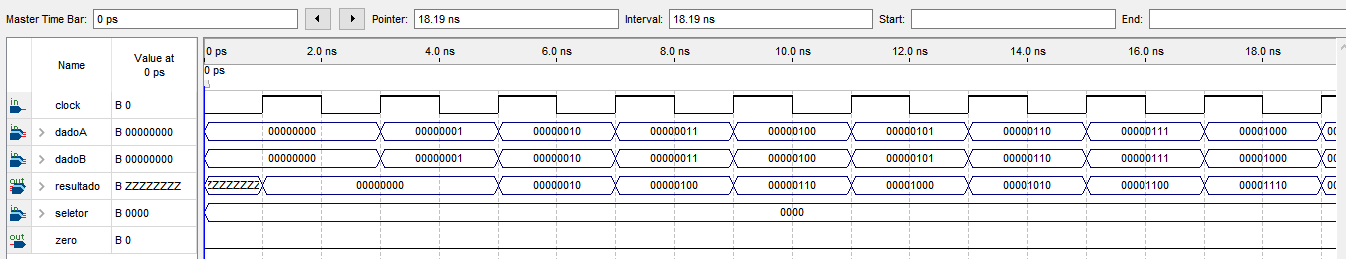
****

**Componente Unidade de Controle**

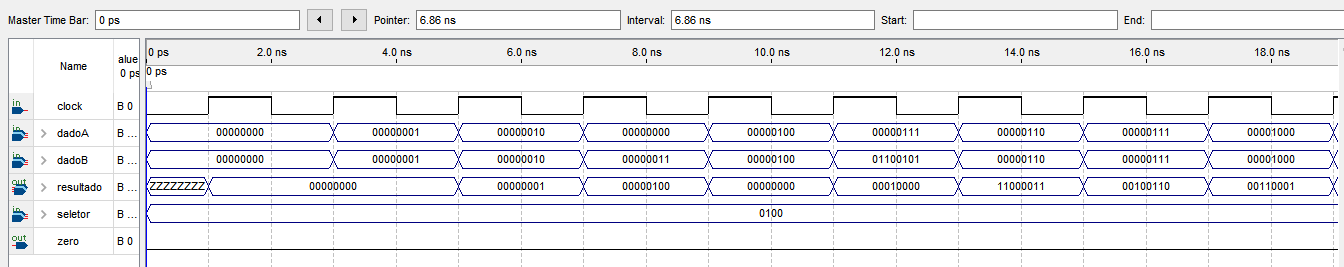
A seguir todos os testes na ULA...



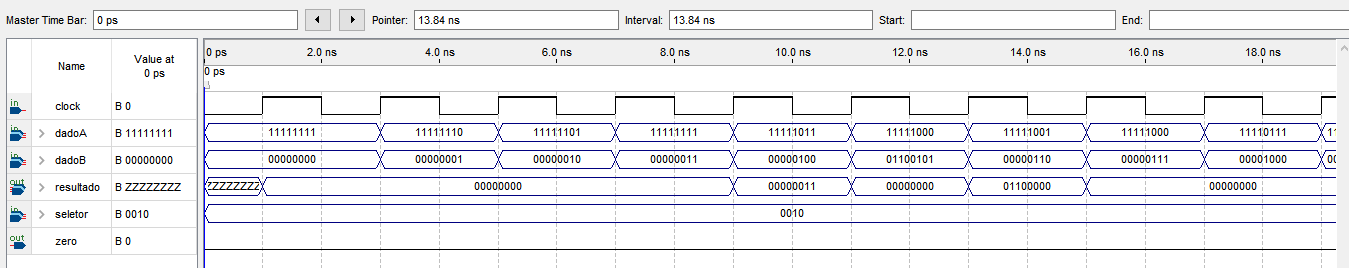
**ULA-SOMA**

****

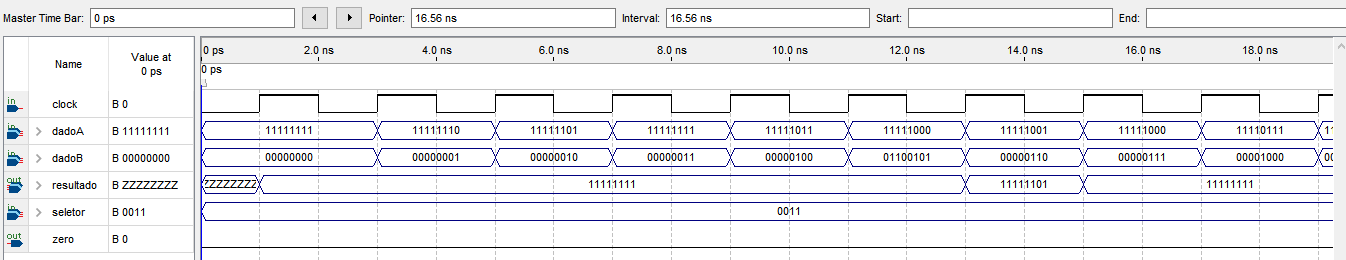
**ULA-SUBTRAÇÃO**

****

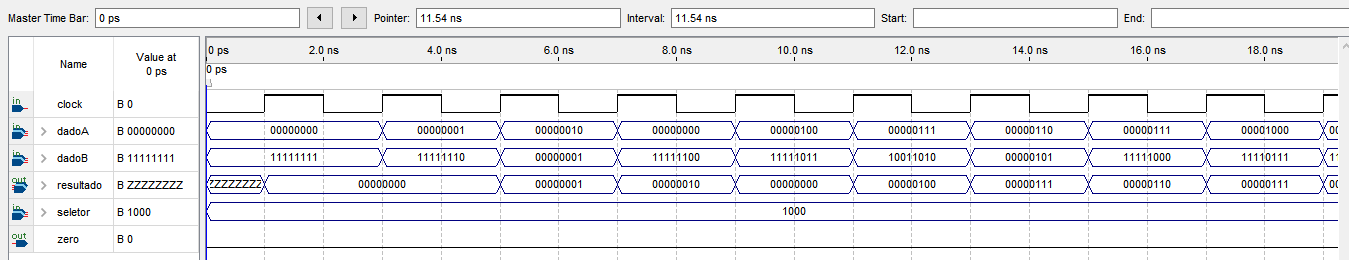
**ULA-MULTIPLICAÇÃO**

****

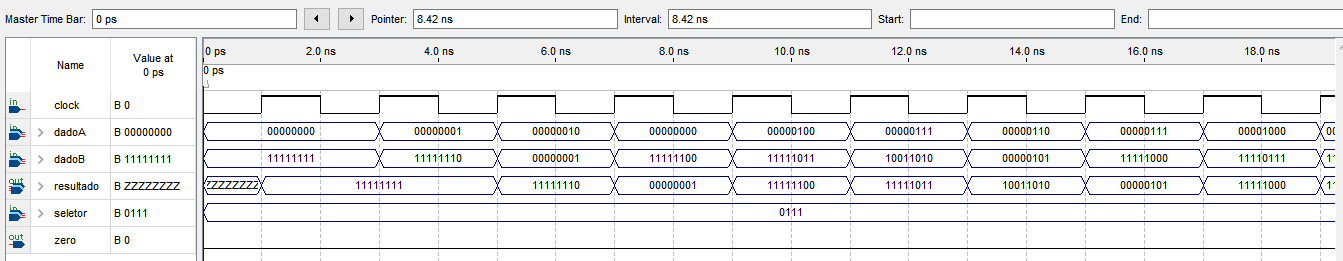
**ULA-AND**

****

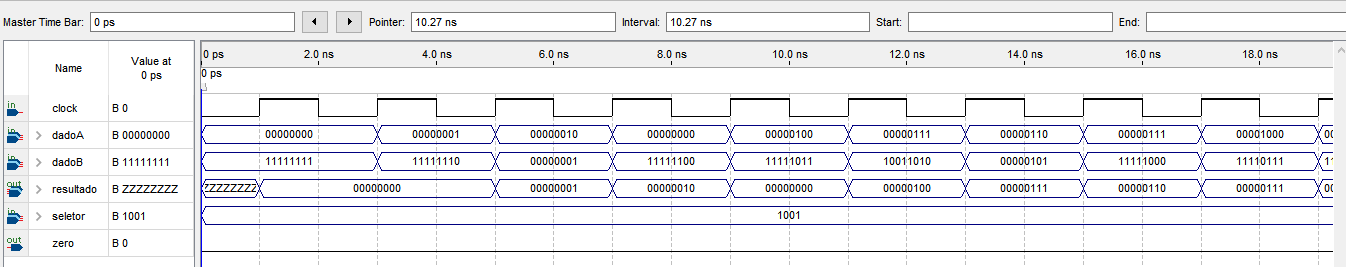
**ULA-OR**

****

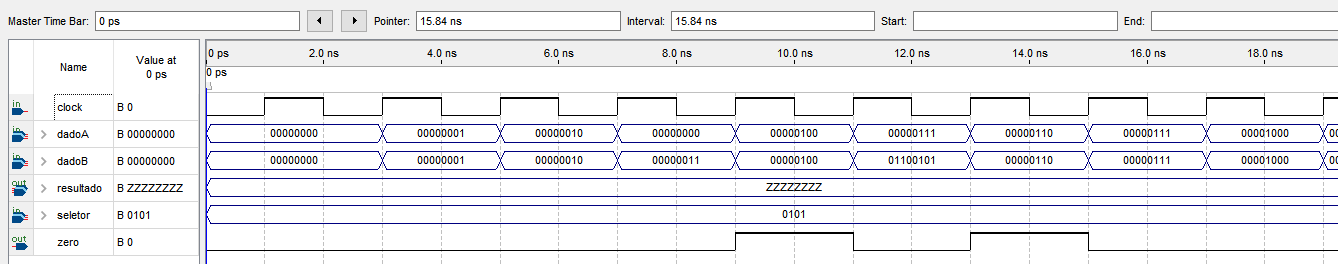
**ULA-LOAD**

****

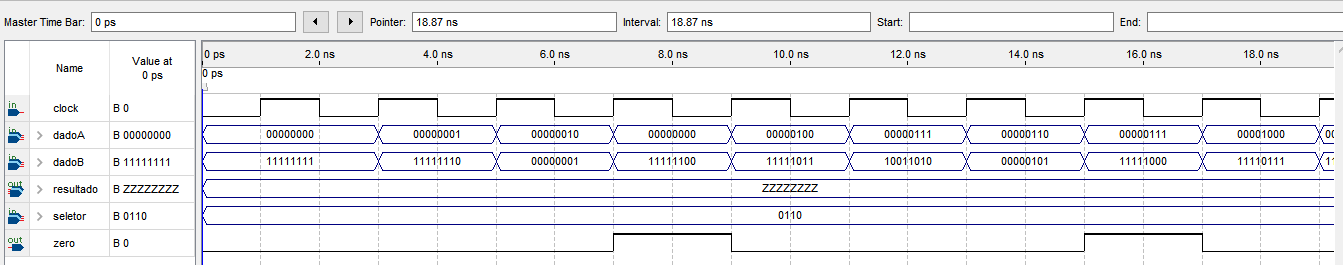
**ULA-LOAD IM**

****

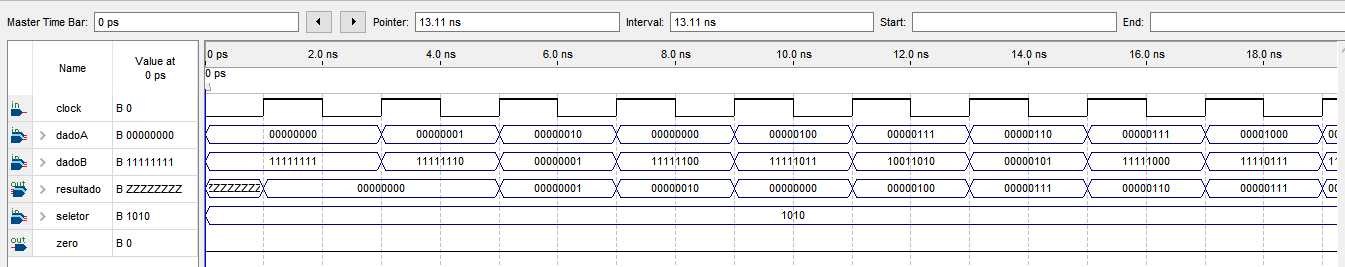
**ULA-STORE**

****

**ULA-BEQ**

****

**ULA-SLT**

****

**ULA-JUMP**

# Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 8 bits denominado de ProIA como trabalho final da disciplina Arquitetura e Sistema de Computadores, com o objetivo de demostrar o que aprendemos ao longo da disciplina na prática, o funcionamento como um todo com seus componentes, o que fazem e como funcionam.