

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR MAGIC**

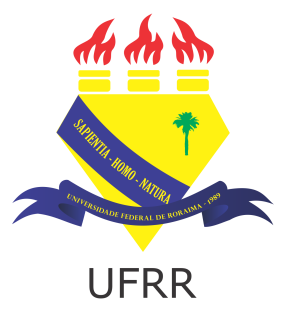
**ALUNOS:**

**Miller Raycell Monteiro Correia – 2017009560**

**Rodrigo de Andrade Rolim Bem - 2017009480**

**Dezembro 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 MAGIC**

**Dezembro de 2018**

**Boa Vista/Roraima**

**Resumo**

Com os conhecimentos que foram desenvolvidos no decorrer da disciplina foi solicitado que os alunos, como representação da nota de projeto final da disciplina desenvolvessem um processador MIPS uniciclo de 16 bits. No trabalho serão demonstrados tanto as técnicas usadas para criar o mesmo tanto quanto o processador em funcionamento.

**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 processador MAGIC foi desenvolvido por dois alunos do curso de ciência da computação, a linguagem utilizada foi a linguagem de síntese de circuitos digitais VHDL. O processador possui o conjunto de instruções de 16 bits e realiza operações do tipo R, I e J.

## Plataforma de desenvolvimento

Para a implementação do processador MAGIC foi utilizado a IDE: Quartus Prime 18.0 Lite Edition

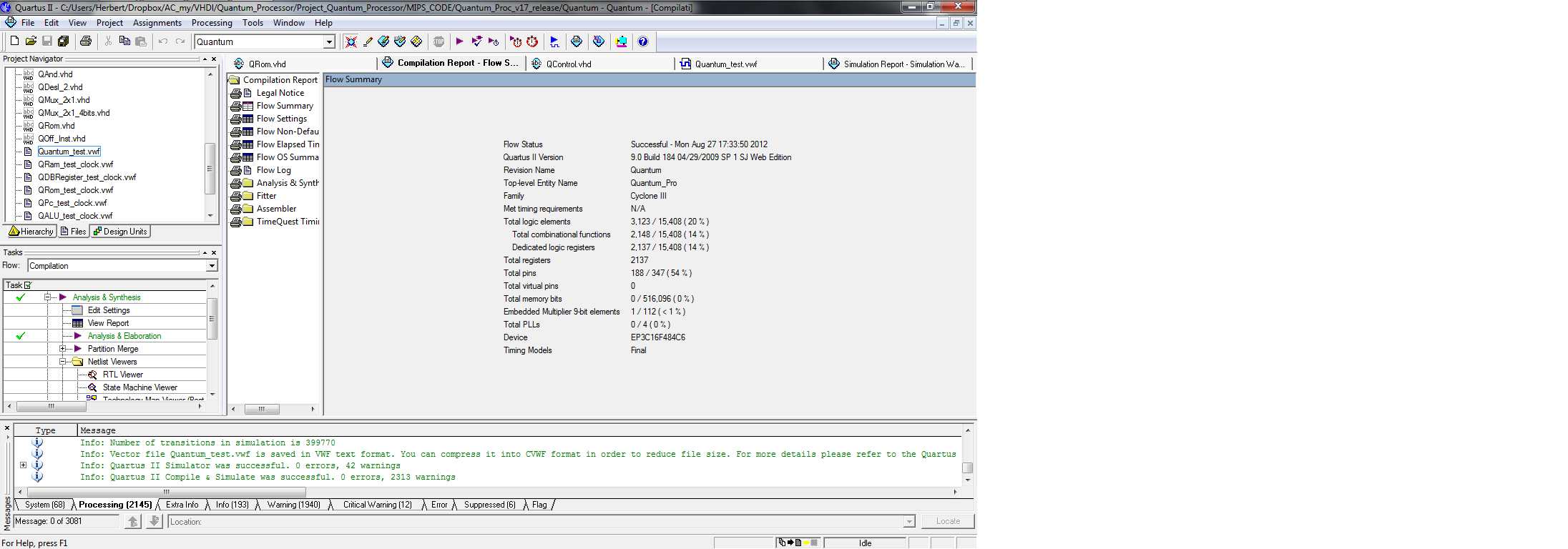


Figura 1 - Especificações no Quartus

## Conjunto de instruções

O processador MAGIC possui 8 registradores: S0, S1, S2, S3, S4, S5, S6, S7. Assim como 13 formatos de instruções de 16 bits cada, Instruções do tipo R (operações aritméticas), tipo I (load, store, BEQ, BNE), tipo J (desvios), seguem algumas considerações sobre as estruturas contidas nas instruções:

* **OPCODE**: A operação básica a ser executada pelo processador, tradicionalmente chamado de código de operação;
* **RS**: O registrador contendo o primeiro operando fonte e adicionalmente para alguns tipos de instruções (ex. instruções do tipo R) é o registrador de destino;
* **RT**: O registrador contendo o segundo operando fonte;
* **RD**: Registrador onde será armazenado o resultado da operação

Tipo de Instruções:

**- INSTRUÇÃO DO TIPO R:**

Esse formato de instrução é caracterizado pelas operações aritméticas, as operações que são suportadas pelo MAGIC são soma, subtração, multiplicação, and e or.

Formato para escrita de código na linguagem MAGIC:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Opcode | Reg1 | Reg2 | Reg3 | Operação |

Formato para escrita em código binário:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 bits | 3 bits | 3 bits | 3 bits | 3 bits |
| 15-12 | 11-9 | 8 - 6 | 5 - 3 | 2 - 0 |
| Opcode | Reg3 | Reg1 | Reg2 | Funct |

**- INSTRUÇÃO DO TIPO I**

São consideradas as instruções que compreendem loads e stores que são operações que trabalham diretamente com a memória e também engola as operações de desvios condicionais como BEQ e BNE.

Formato para escrita de código na linguagem MAGIC:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Reg1 | Reg2 | Valor |

Formato para escrita em código binário:

|  |  |  |  |
| --- | --- | --- | --- |
| 4 bits | 3 bits | 3 bits | 6 bits |
| 15-12 | 11-9 | 8 - 6 | 5 - 3 |
| Opcode | Reg2 | Reg1 | Valor |

**- INSTRUÇÃO DO TIPO J**

São consideradas as instruções que fazem os pulos, ou desvios, ou seja, são as operações de endereçamento de memória, muito utilizada para fazer laços de repetição, recursividade, entre outros.

Formato para escrita de código na linguagem MAGIC:

|  |  |
| --- | --- |
| Opcode | Valor do salto |

Formato para escrita em código binário:

|  |  |
| --- | --- |
| 4 bits | 12 bits |
| 15-12 | 11 - 0 |
| Opcode | Valor do salto |

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

O número de bits do campo Opcode das instruções é igual a quatro, sendo assim obtemos um total () de 16 Opcodes **(0000 - 1111)** 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 MAGIC.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0000 | Operação tipo R | R | Operações de soma que serão estabelecidas pelo campo funct | **add** $s2, $s0,$s1  **sub** $s2, $s0,$s1  **mult** $s2, $s0,$s1 |
| 0001 | Load | I | Fazer a escrita de valores em um registrador | **lw** $s1, $s2; |
| 0010 | Store | I | Armazenar valores na memória | sw |

## Descrição do Hardware

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

### PC

PC ou Program Counter é o componente responsável por armazenar o endereço da instrução que será realizada, no MAGIC ele foi construído como sendo um registrador flip flop do tipo d, a necessidade do mesmo ser flip flop foi para ter certeza que o endereço da instrução não será perdido e assim ter um pouco mais de certeza que o processador poderá ler um código por completo sem haver perda de endereços de informação. O MAGIC possui a integração do pc conter com um contador síncrono de 3 bits para evitar problemas com lixo de memória.

No PC do MAGIC existem duas entradas, sendo uma para o clock, que irá definir o tempo de execução e a entrada do endereço de memória da instrução, também existe uma saída que será que irá entrar na memória de instrução relativo ao endereço de memória da instrução que será executada.

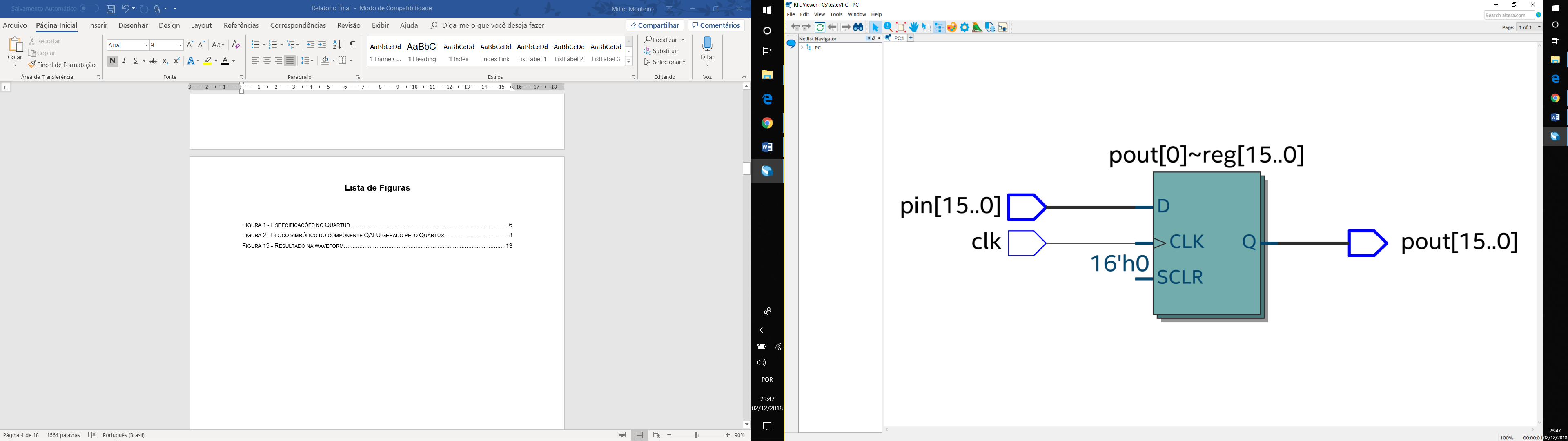


Figura 2 – PC RTL View

* + 1. **Memoria de instruções**

A memória de instruções é o segundo passo da execução de uma instrução, na memória de instruções é onde está armazenada as instruções do programa que será executado, no MAGIC ela foi construída como uma memória ROM de 16 bits podendo endereçar até 65536 endereços (, todos as instruções são armazenadas em uma matriz e o endereço que entra na mesma será o endereço da instrução a ser executada.

A memória de instrução do MAGIC conta com uma entrada que é o endereço de memória da instrução que será executada no momento, e 8 saidas: 1- clock: Clock do sistema que ordena os ciclos de execução; 2 – Opcode: Opcode que irá para a unidade de controle para que se possam ser ativadas as respectivas flags necessárias da instrução; 3 – rd: Primeiro registrador que será utilizado na instrução; 4 – rt: Segundo registrador que será usado na instrução; 5 – rs: Registrador de destino onde será armazenado os valores depois da fase de execução da instrução; 6 – funct: Campo que indica para a ULA qual será a operação que será executada no caso de uma instrução do tipo R; 7 – tipoi: Campo exclusivo das instruções do tipo I, que será o endereço do desvio, ou valor a ser armazenado; 8 – jump: Campo exclusivo da instrução do tipo J, que se refere a qual será o próximo endereço de salto

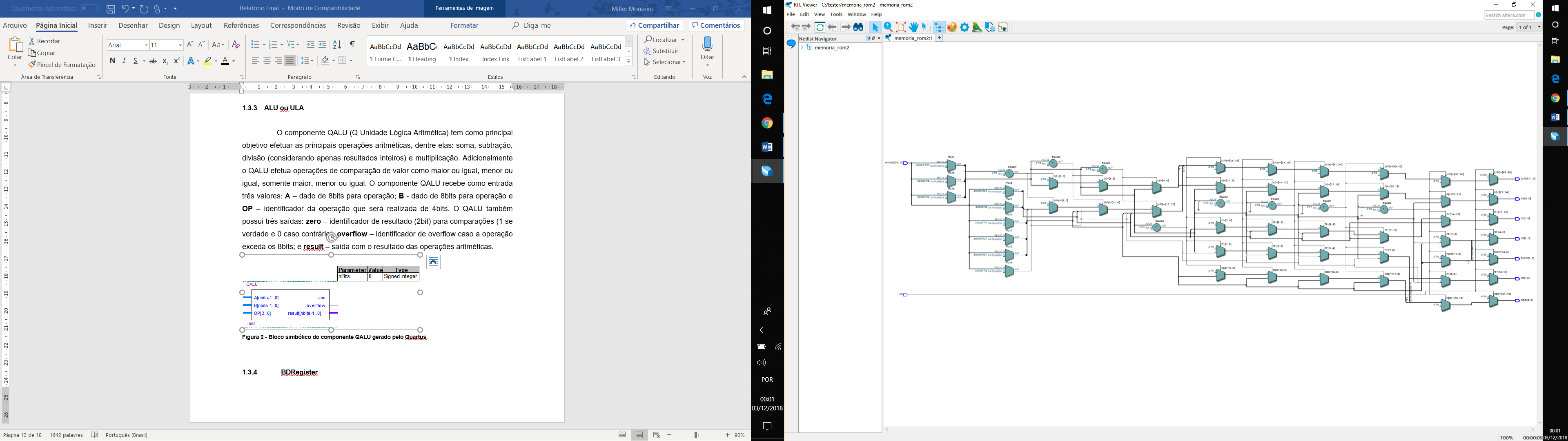


Figura 3 – Memoria de instrução RTL View

A figura da RTL View pode estar desorganizada, mas isso se deve ao fato da forma na qual o código foi descrito, o código usa a estrutura de decisão if para fazer a verificação da saída da memória, pois de acordo com o formato da instrução a saída será alterada, dessa forma durante o desenvolvimento foi necessário usar tal estrutura como o if, dessa forma se resulta com uma visualização não tão amigável, contudo tudo que se deseja saber pode ser extraído da mesma.

* + 1. **Banco de Registradores**

Registradores são as estruturas auxiliares do processador, seu funcionamento pode ser comparado a de uma variável, no mesmo será utilizado como intermediador dos valores que serão utilizados no código.

No MAGIC foram utilizados 8 registradores, que foi definido segundo o formato da instrução do mesmo, o componente que irá armazenar todos esses registradores é conhecido como banco de registradores, seu comportamento será definido de acordo com as entradas que o mesmo receber, o banco de registradores do MAGIC foi desenvolvido com 4 entradas, sendo 3 delas valores binários relativos aos registradores que serão utilizados na instrução que está sendo executada, e uma entrada sendo uma flag da unidade de controle. O Banco também possui duas saídas que são referentes aos dados para serem enviados a ULA para serem operados;

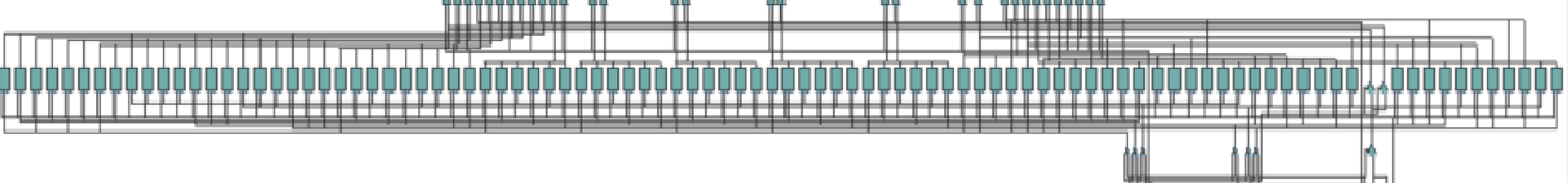


Figura 4 – Banco de Registradores RTL View

* + 1. **ULA**

A Unidade Lógica e Aritmética é o componente que realiza as operações dentro do processador, sua função incorpora realizar somas, subtrações, multiplicações, calcular endereços de desvios, entre outras.

No MAGIC a ULA realiza as seguintes operações, soma, subtração, multiplicação, and e or, para realizar as operações soma e subtração foi utilizado a biblioteca arith do VHDL que permite que as operações sejam realizadas já tratando overflow, a multiplicação foi aplicada através do algoritmo de Booth, no qual realiza a operação com duas entradas de 8 bits resultando em uma saída de 16 bits, já que o MAGIC não é provido dos registradores High e Low para tratamento de multiplicação.

A ULA do MAGIC possui 3 entradas e 3 saídas, sendo que as entradas são: 1 – Entrada1: Dado 1 saído do banco de registradores; 2 – Entrada2: Dado 2 saído do banco de registradores; 3 - Flag que indica a operação que ocorrerá na ULA, as saídas são: 1- Saída para dados: que é a saída para a memória de dados; 2- Saída de dados para mux: Saida para o multiplexador que de acordo com a flag do mesmo irá definir o que será enviado para o write back; 3 – ZERO: Esse valor será usado para caso seja uma instrução de jump para fazer a escrita do PC.

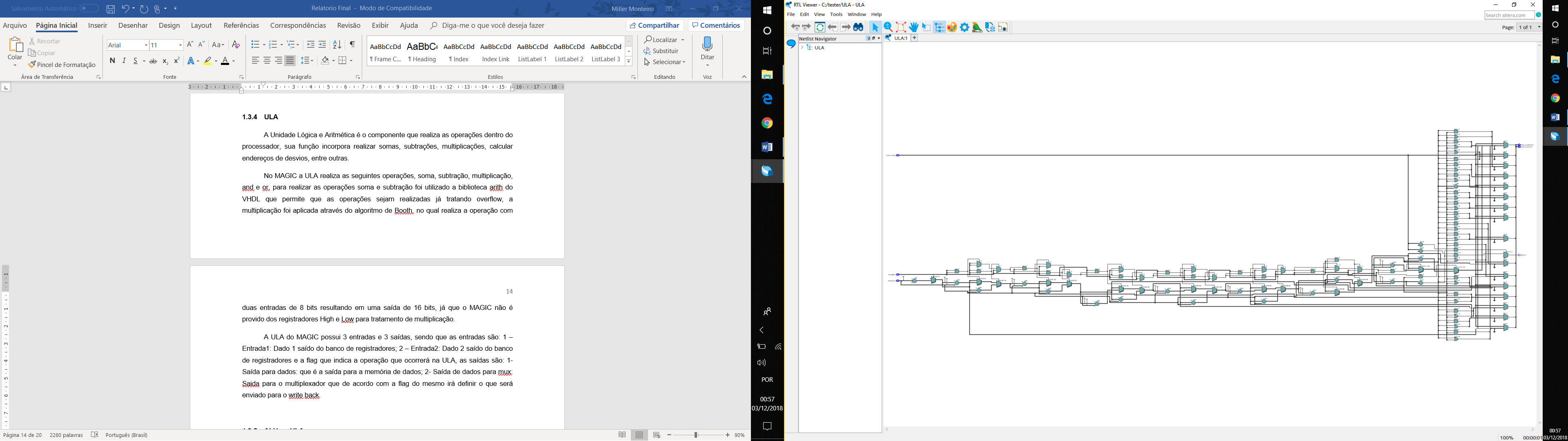


Figura 5 – ULA RTL View

* + 1. **Memória de Dados**

A memória de dados é o componente que guarda alguns valores durante a execução do programa, seu comportamento pode ser definido como leitura e escrita, ela pode não só somente ler um valor que já foi armazenado e manda-lo para o resto do circuito como fazer com que dados sejam armazenados dentro da mesma.

Seguindo seu comportamento a memória de dados no MAGIC foi descrita como uma memória RAM, na qual só foi endereçado 16 espaços de memória, e não 65536 como esperado e isso se deve a limitações do ambiente de desenvolvimento Quartus, pois com todo MAGIC unido já haviam sido endereçados 65536 endereços de memória e com mais 65536 endereços seria muita memória e para manter a saúde do computador no qual está sendo desenvolvido o Quartus veta esse tipo de declaração, logo definimos que apenas 16 endereços seriam mais do que o suficiente para o MAGIC, por isso foram definidos esses valores.

No MAGIC a Memória de Dados conta com 5 entradas e 1 saída, sendo as entradas: 1 – Endereço: É o valor que sai da ULA relativo ao endereço a ser acessado da memória; 2 – Clock: Clock do sistema que rege o funcionamento dos componentes; 3 – Entrada: É o dado 2 que sai do Banco de Registradores em caso de instruções do tipo load e store; 4 – rd: Flag que indica se será feito uma escrita na memória; 5 – wd: Flag que indica se será feito uma leitura na memória, e a saída é dado que foi solicitado.

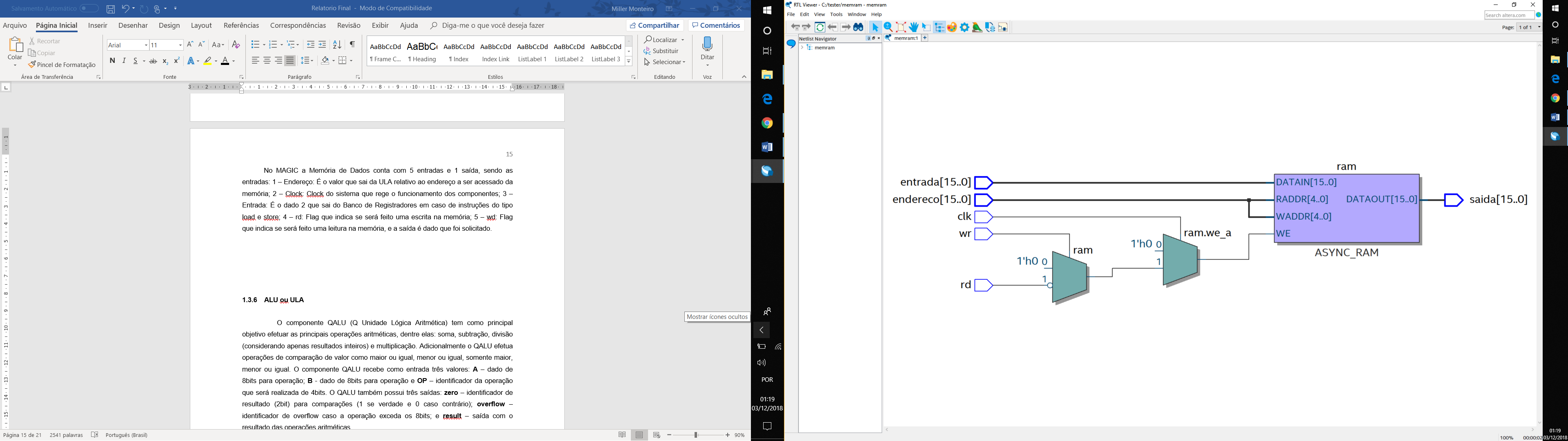


Figura 6 – Memória de dados RTL View

* + 1. **Unidade de Controle**

A unidade de Controle é um dos componentes mais importantes do processador, pois a mesma orquestra como a instrução vai ser realizada, quais componentes devem ser ativados, quais não devem ser ativados.

No MAGIC a Unidade de Controle conta com 9 flags:

Tabela 2 – Flags Unidade de Controle

|  |  |
| --- | --- |
| FLAG | FUNÇÃO |
| REGDEST | Essa flag indica que para um multiplexador que se encarrega de indicar se irá ter um registrador para fazer a escrita de dados (1 bit). |
| ORIGALU | Essa flag vai para a ULA indicando qual a operação que será realizada, ou seja, indicando o campo funct no caso de uma instrução do tipo R (4 bits). |
| MEMPARAREG | Essa flag vai para um multiplexador que irá definir qual o valor será devolvido para o registrador de destino, o resultado da ULA ou da Memória de Dados |
| ESCREVEREG | Essa flag indica que o registrador para escrita será utilizado. |
| LEMEM | Flag que indica para a Memória de Dados que será feito uma escrita na mesma. |
| BRANCH | Flag que indica para a porta AND e que é operado com a saída ZERO da ULA se a instrução será um branch; |
| ALUSRC | Flag que indica para um multiplexador qual a entrada que irá para a ULA, se será a saída do Banco de Registradores ou os bits 5:0 da instrução |
| JUMP | Flag que indica que um desvio será realizado. |

Além dessas saídas a Unidade de Controle possui duas entradas sendo uma o clock do sistema e outra o opcode da instrução.

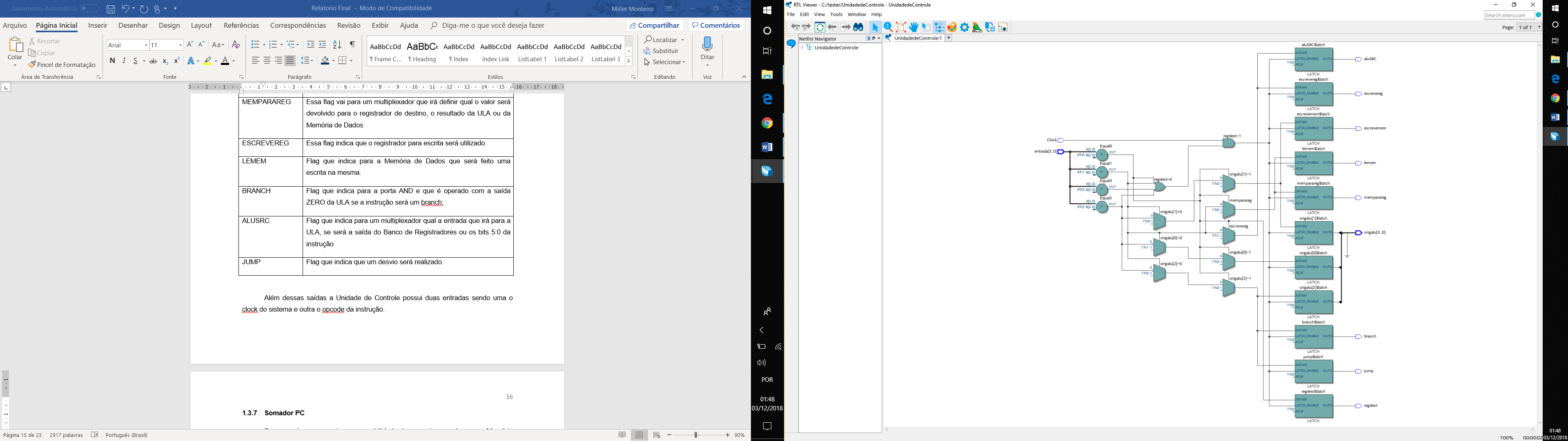


Figura 7 – Unidade de Controle RTL View

* + 1. **Somador PC**

Componente que possui a responsabilidade de computar o endereço na Memória de Instruções da próxima instrução a ser utilizada, para o MIPS seria utilizado que uma word possui 2 bytes (16 bits), porém o funcionamento do MAGIC é diferente, nele as instruções estão armazenadas em uma matriz, logo o PC deve guardar a posição na matriz da instrução, então o somador irá se comportar somando mais 1 para cada execução que será o endereço da próxima instrução.

O Somador PC do MAGIC conta com uma entrada, o PC atual e tem como saída o PC + 1.

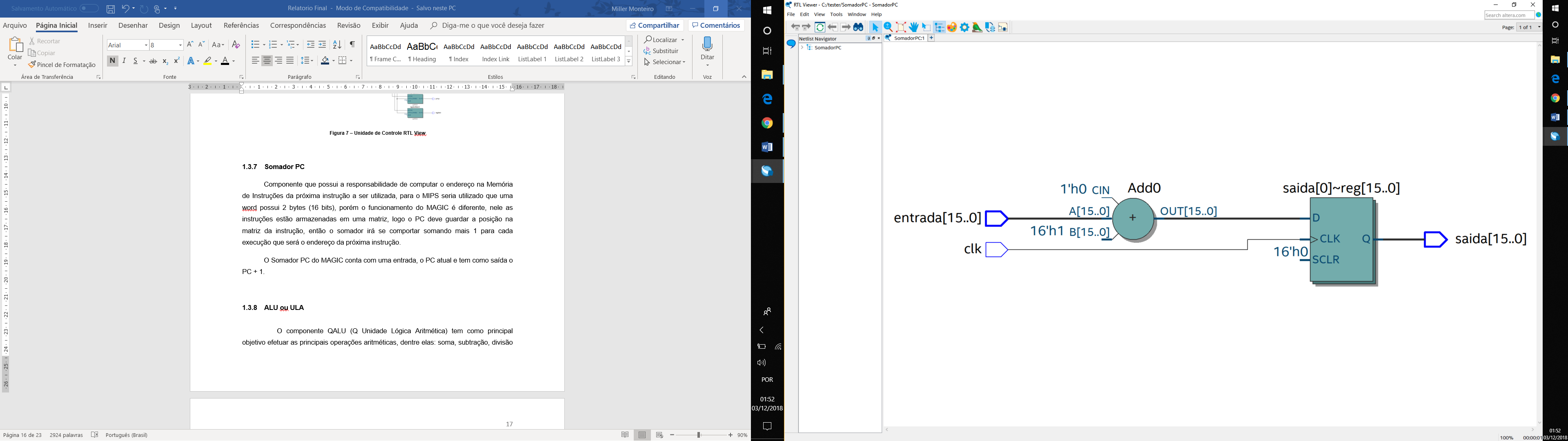


Figura 8 – Somador PC RTL View

* + 1. **Extensor de Sinal**

Extensor de sinal é o componente no qual é relativo ao desvio no caso de uma operação do estilo branch, no MAGIC ele compreende os bits 5:0 desse tipo de instrução, como todo o processador foi pensado para ser trabalhado com 16 bits, com 6 não teríamos resultados satisfatórios, por isso se pega os 6 bits e se concatena com 10 bits ‘0’ a frente, para que não haja alteração do valor proposto e assim pode se continuar a execução da instrução sem problemas.

No MAGIC o Extensor de sinal possui duas entrada que são os 6 bits e o clock do sistema, e uma saída que são os 16 bits para o salto.

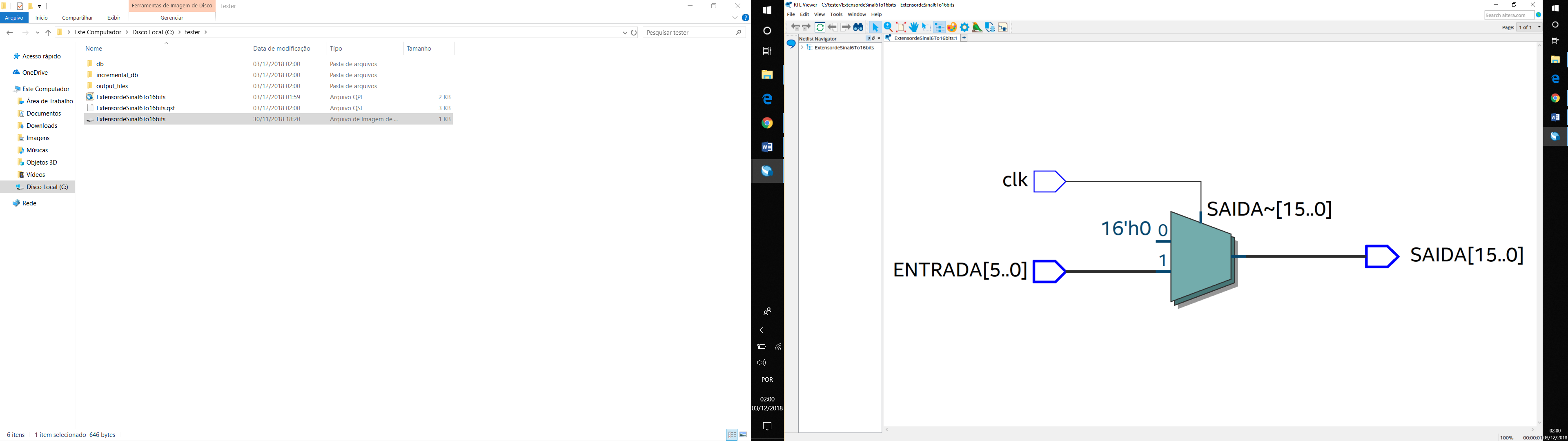


Figura 9 – Extensor de Sinal RTL View

* + 1. **Shift à esquerda**

Esse componente basicamente trata a sua entrada e tira seus dois bits mais significativos e os repõe com 0 para que o número de bits seja o mesmo.

O MAGIC possui dois tipos de Shifter, sendo um normal e um para jump que apenas concatena os bits a frente e depois concatena novamente com os 2 bits mais significativos do PC atual e isso seguirá para se calcular o endereço do jump.

Esses componentes contam com 2 entradas, sendo uma o clock do sistema e uma a entrada que será operada e contam com uma saída com o valor operado, o shift para jump possui mais uma entrada que é o PC para que seja concatenado.

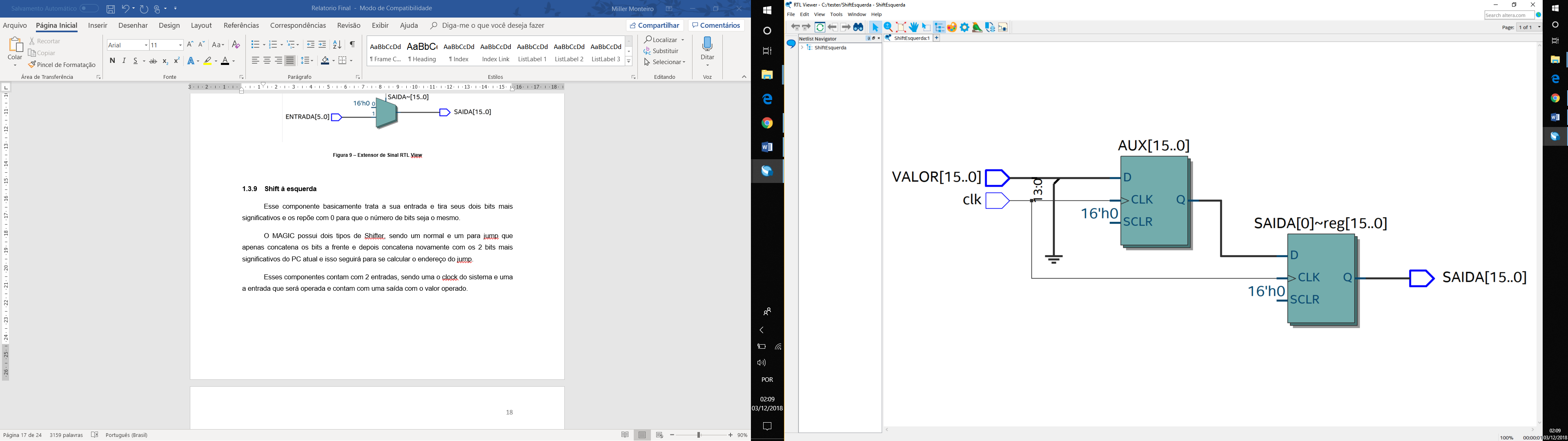


Figura 10 – Shifter de 2 bits a esquerda RTL View

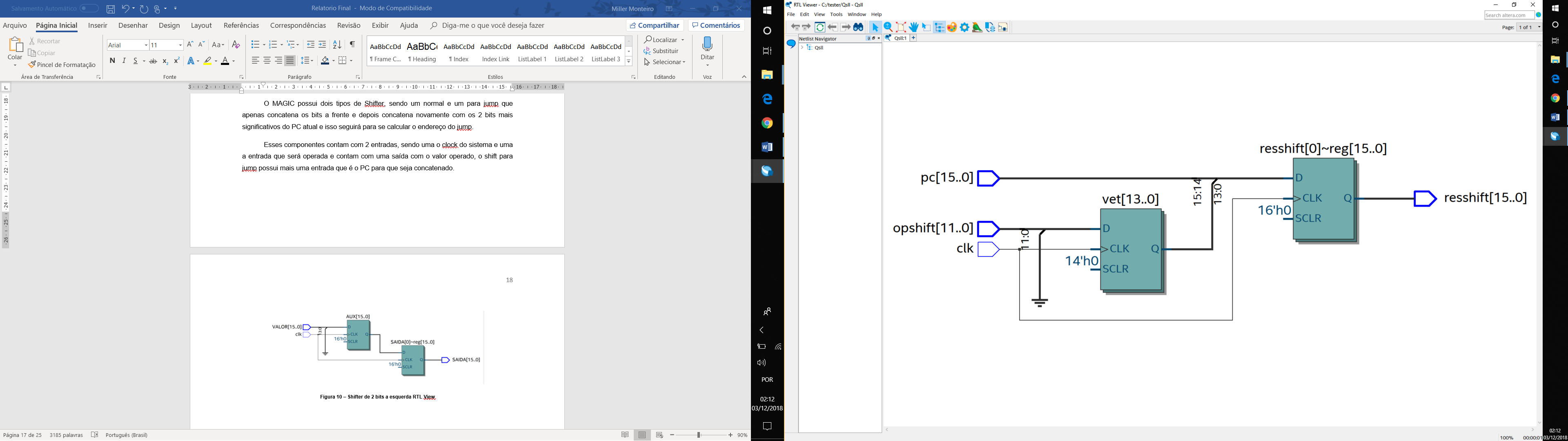


Figura 11 – Shifter de 2 bits a esquerda e concatena com PC RTL View

### ALU ou ULA

O componente QALU (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 QALU efetua operações de comparação de valor como maior ou igual, menor ou igual, somente maior, menor ou igual. 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.



Figura 2 - Bloco simbólico do componente QALU gerado pelo Quartus

### BDRegister

**[Todo] Descrição**

### Clock

**[Todo] Descrição**

### 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**: XXXX.
* **en\_data**: XXXX.
* **EscMem**: XXXX.
* **MemParaReg**: XXXX.
* **UlaOp**: XXXX.
* **LwSwOp**: XXXX.
* **EscReg:** XXXX.
* **Wrt\_LRT**: XXXX.
* **FlagPC**: 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 | DvC | off\_instruction | en\_data | Esc  Mem | Mem  ParaReg | UlaOp | LwSw  Op | EscReg | Wrt\_LRT | FlagPc | enJmp |
| add | 0 | 1 | 1 | 0 | 1 | 0000 | 0 | 1 | 0 | 1 | 0 |
| sub | 0 | 1 | 1 | 0 | 1 | 0001 | 0 | 1 | 0 | 1 | 0 |
| div | 0 | 1 | 1 | 0 | 1 | 0111 | 0 | 1 | 0 | 1 | 0 |
| Inicialização | 0 | 1 | 1 | Z | Z | ZZZZ | Z | Z | Z | 1 | 0 |

### Memória de dados

**[Todo] Descrição**

### Memória de Instruções

**[Todo] Descrição**

### Somador

**[Todo] Descrição**

### And

**[Todo] Descrição**

### Mux\_2x1

**[Todo] Descrição**

### PC

**[Todo] Descrição**

### ZERO

**[Todo] Descrição**

## 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...

**[Todo] Figura RTL**

# Simulações e Testes

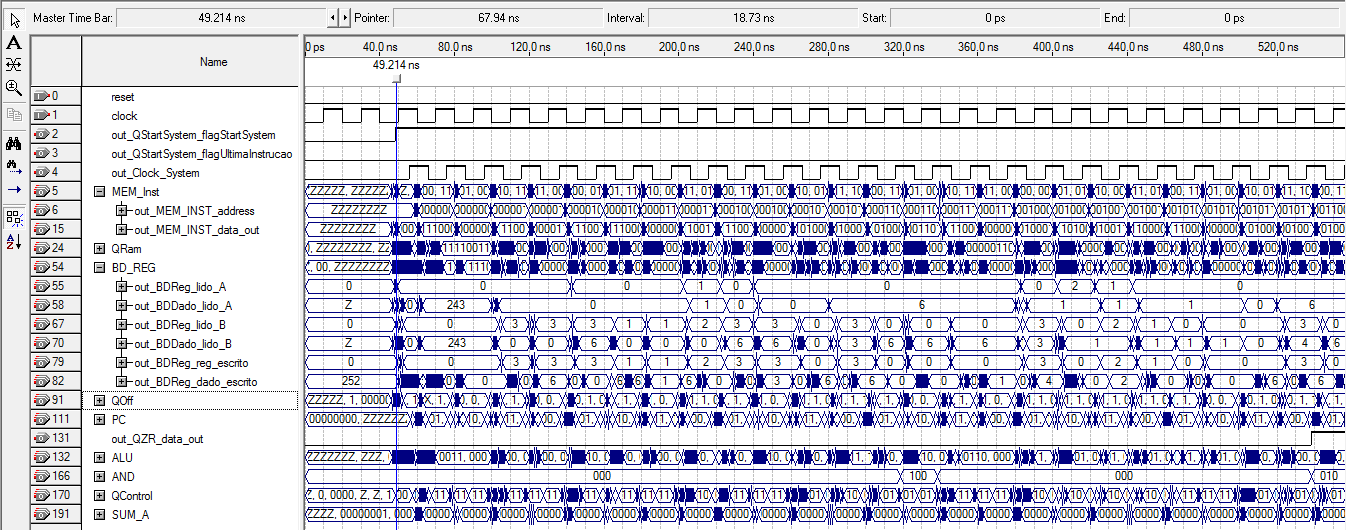
Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em especifico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador XXXX utilizaremos como exemplo o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

Tabela 3 - 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 3 - Resultado na waveform.

# Considerações finais

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