

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR TORRE**

**ALUNOS:**

**Gabriel Arcanjo Campelo Fadoul – 2017023784**

**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 TORRE**

**Dezembro de 2018**

**Boa Vista/Roraima**

**Resumo**

Este trabalho apresenta o projeto final da disciplina de Arquitetura e Organização de Computadores, onde é implementado um processador Uniciclo de 16 bits baseado nos conhecimentos adquiridos em sala. O relatório se dedicará a especificar as etapas e processos do desenvolvimento do projeto, desde a organização das instruções, ao Port Map dos componentes. O Processador é a peças mais significativa dentro de um sistema computacional, levando em conta que ele gerencia todo o esquema de componentes, controlando as operações realizadas por cada unidade funcional.

**Conteúdo**

[1 Especificação 7](#_Toc531728545)

[1.1 Plataforma de desenvolvimento 7](#_Toc531728546)

[1.2 Conjunto de instruções 8](#_Toc531728547)

[1.3 Descrição do Hardware 10](#_Toc531728548)

[1.3.1 ULA 10](#_Toc531728549)

[1.3.2 Banco de Registradores 11](#_Toc531728550)

[1.3.3 Unidade de Controle 11](#_Toc531728551)

[1.3.4 Memória de dados 12](#_Toc531728552)

[1.3.5 Memória de Instruções 13](#_Toc531728553)

[1.3.6 Somador 13](#_Toc531728554)

[1.3.7 QAndBIT 14](#_Toc531728555)

[1.3.8 Muliplexado2x1\_16bits 14](#_Toc531728556)

[1.4 Datapath 14](#_Toc531728557)

[2 Simulações e Testes 16](#_Toc531728558)

[3 Considerações finais 17](#_Toc531728559)

**Lista de Figuras**

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

[Figura 2 - Bloco simbólico do componente ULA 10](#_Toc444681816)

[Figura 3 - Bloco simbólico do componente BancoRegistradores 10](#_Toc444681815)

[Figura 4 - Bloco simbólico do componente UnidadedeControle 12](#_Toc444681815)

[Figura 5 - Bloco simbólico componente memoria\_ROM2 12](#_Toc444681817)

[Figura 6 - Bloco simbólico do componente memram 12](#_Toc444681817)

[Figura 7 - Bloco simbólico do componente Somador. 13](#_Toc444681817)

[Figura 8 - Bloco simbólico do componente QAndBIT 13](#_Toc444681817)

[Figura 9 - Bloco simbólico do componente Multiplexado2x1\_16bits 13](#_Toc444681817)

[Figura 10 - Bloco simbólico do componente PC 14](#_Toc444681817)

[Figura 11 - Bloco simbólico do componente 13](#_Toc444681817)

**Lista de Tabelas**

[Tabela 1 – Tabela que mostra a lista de Op e Funct utilizadas na instrução tipo-r pelo processador . 8](#_Toc444681822)

[Tabela 2 – Tabela que mostra a lista de Op utilizadas na instrução tipo-i pelo procesador. 9](#_Toc444681823)

[T abela 3 – Tabela que mostra a lista de Op e endereço utilizadas na instrução tipo-j pelo processador 9](#_Toc444681824)

[T abela 4 – Detalhes das flags de controle do processador 11](#_Toc444681824)

# Especificação

É apresentado nesta seção o conjunto dos itens utilizados para o desenvolvimento do processador Torre, com as descrições de todas as etapas da construção do processador.

## Plataforma de desenvolvimento

Para a implementação do processador Uniciclo de 16 bits foi utilizado a IDE Quartus Prime Lite com as seguintes especificações:

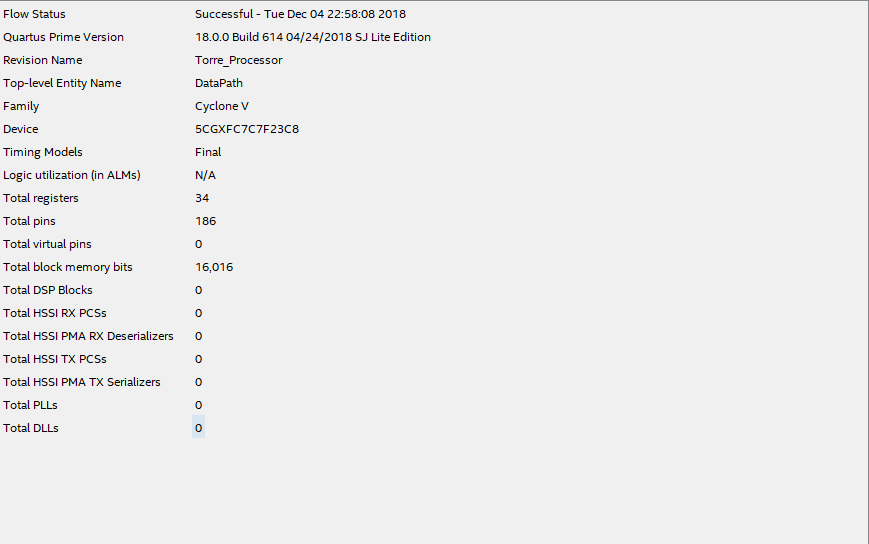


Figura 1 - Especificações no Quartus

## Conjunto de instruções

O processador Torre possui 8 registradores. Assim como 3 formatos de instruções de 16 bits cada, Instruções do **tipo R, I,** **J** seguem algumas considerações sobre as estruturas contidas nas instruções:

* **OpCode**: Campo principal de identificação de tipo de Instrução;
* **Rd**: O registrador do operando de destino. Ele recebe o segundo operando de destino.
* **Rs**: O registrador contendo o primeiro operando de origem;
* **Rt**: O registrador contendo o segundo operando de origem;
* **Funct**: Campo exclusivo para indicar para ULA qual operação aritmética será realizada, caso instrução tipo R.

Tipo de Instruções:

**- Formato do tipo R:** Chamado de Tipo-R, este formato simboliza o uso de registradores pela instrução, unicamente, instruções lógicas e aritméticas.

Formato para escrita de código na linguagem Brick:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OpCode | Rs | Rt | Rd | Funct |

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 | Rs | Rt | Rd | Funct |

**Visão geral das instruções do Processador Uniciclo de 16bits:**

O número de bits do campo **OpCode** das instruções é igual a quatro e do **funct** igual a três, sendo assim obtemos um total () de 16 **OpCodes** **(0-15)** e 8 **Funct (0-7)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

Tabela 1 – Tabela que mostra a lista de Opcode e funct utilizadas pelo processador Uniciclo de 16bits.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Op** | **Funct** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0000 | 000 | ADD | R | Soma | **Add** $S2, $S1, $S0 ou seja, $S0 = $S1 + $S2 |
| 0000 | 001 | SUB | R | Subtrai | **Sub** $S2, $S1, $S0 ou seja, $S0 = $S1 - $S2 |
| 0000 | 011 | MULT | R | Multiplica | **Mult** $S2, $S1, $S0 ou seja $S0 = $S1 \* $S2 |

**- Formato do tipo I:** Este formato é caracterizado por instruções de branch’s condicionais e transferência de dados.

Formato para escrita de código na linguagem Brick:

|  |  |  |  |
| --- | --- | --- | --- |
| OpCode | Rd | Rf | Imediato |

Formato para escrita em código binário:

|  |  |  |  |
| --- | --- | --- | --- |
| 4 bits | 3 bits | 3 bits | 4 bits |
| 15-12 | 11-9 | 8-6 | 5-0 |
| OpCode | Rd | Rf | Imediato |

**Visão geral das instruções do Processador Uniciclo de 16bits:**

O número de bits do campo **OpCode** das instruções é igual a quatro, sendo assim obtemos um total () de 16 **OpCodes (15-12)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 2.

Tabela 2 – Tabela que mostra a lista de OpCode utilizada pelo processador Uniciclo de 16bits.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Op** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0001 | LW | I | Load Word | **Lw** $S0, 100($S1), ou seja, $S0 = Memory[$S1 + 100] |
| 0010 | SW | I | Store Word | **Sw** $S0, 100($S1), ou seja, Memory[$S1 + 100] = $S0 |
| 0011 | BEQ | I | Branch on equal | **Beq** $S0, $S1, 25, ou seja, if($S0 == $S1) go to PC + 4 + 25 |
| 0100 | BNE | I | Branch on not equal | **Bne** $S0, $S1, 25, ou seja, if($S0 != $S1) go to PC + 4 + 25 |

**- Formato do tipo J:** Chamado tipo –J, este formato faz com o endereçamento seja o mais simples possível, ou seja, ele desvia para o endereço de destino.

Formato para escrita de código na linguagem Brick:

|  |  |
| --- | --- |
| Op | Endereço |

Formato para escrita em código binário:

|  |  |
| --- | --- |
| 4 bits | 12 bits |
| 15-12 | 11-0 |
| Op | Endereço |

**Visão geral das instruções do Processador Uniciclo de 16bits:**

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 3.

Tabela 3 – Tabela que mostra a lista de OpCode utilizadas pelo processador Uniciclo de 16bits.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Op** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0101 | Jump | J | Pula até o Local | **J** 2500, ou seja, go to 2500 |

## Descrição do Hardware

A partir daqui será destacada a funcionalidade de cada componente que compõe o projeto de forma um tanto simplificada e direta.

### ULA

A Unidade Lógica Aritmética (ULA) é responsável, como o próprio nome já retrata, a parte de operações lógicas e aritméticas. No geral, o componente foi feito para receber dois valores de 16 bits nas portas EntradaA e EntradaB, e o valor que entrar pelo Controle\_ULA, que é a concatenação do OpCode com o campo Funct, irá definir qual operação será realizada dentro do componente. As portas Saida\_to\_Dados e Saida\_to\_Mux recebem o mesmo valor de saída, somente foram separadas pois cada uma delas vai para um componente diferente. Já o ZeroULA se relaciona com o BEQ e BNE, se for um BEQ e a subtração das entradas der 0, a saída recebe 1, da mesma forma se for um BNE e a subtração der diferente de zero. Se não cair em nenhuma dessas situações a saída recebe 0.

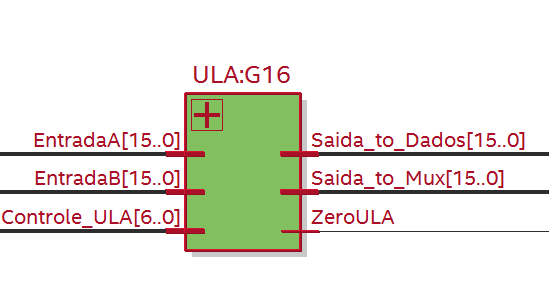


Figura 2 - Bloco simbólico do componente ULA

### Banco de Registradores

É um conjunto de registradores da CPU, ou seja, vários registradores onde serão armazenados na memória 8 palavras de 16 bits. São utilizados na execução de programas de computadores, disponibilizando um local para armazenar dados.

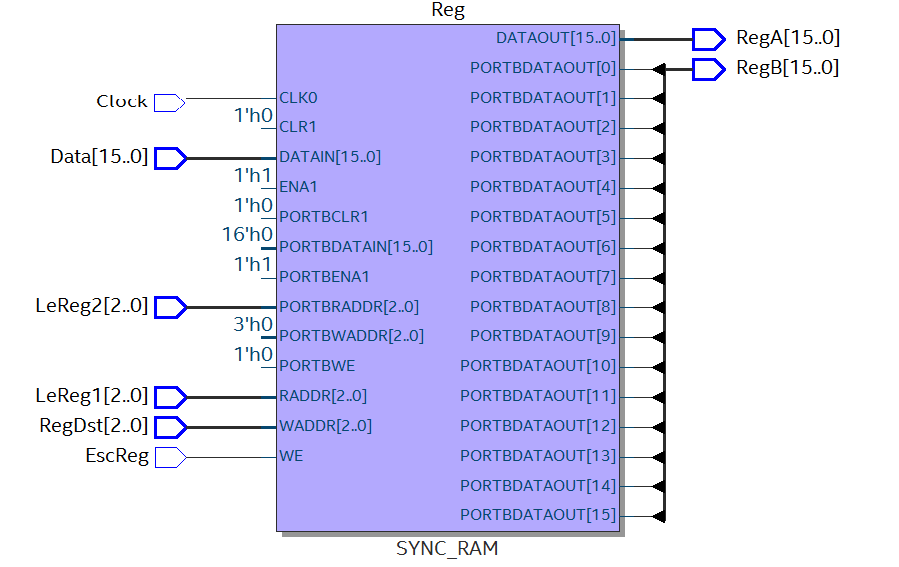


Figura 3 - Bloco simbólico do componente BancoRegistradores

### Unidade de Controle

A Unidade de Controle tem como função o gerenciamento, ou seja, a ativação e desativação de componentes dependendo da Instrução que será executada. Esse controle é feito através das flags de controle que indicam como os componentes irão se comportar.

As flags existentes são as seguintes:

* **regdest**: Direcionado para um multiplexador que configura o registrador de destino .
* **origalu**: Campo Op é passado para o CampoULA que concatena com o Funct para definir a operação da ULA.
* **memparareg**: Direcionado para um multiplexador que configura o valor a ser escrito no registrador de destino.
* **escrevereg**: Enviada para o Banco de Registradores e avisará que a entrada será escrita no registrador de destino .
* **lemem**: Enviado para a memória RAM e diz que ela irá ler o valor contido no endereço de entrada.
* **escrevemem**: Enviado para a memória RAM e diz que ela irá escrever no endereço de entrada.
* **branch:** Vai para a porta AND onde operações lógicas com o ZerodaULA decidirá qual valor um multiplexador irá tomar (Especificamente o Multiplexador de Branch).
* **aluSRC**: Decide qual valor irá para ULA para operar com o Registrador A.
* **jump**: Decidirá se o jump será ou não tomado, através de um multiplexador.

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

Tabela 4 - Detalhes das flags de controle do processador.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Comando | entrada | regdest | origalu | memparareg | escrevereg | lemem | escrevemem | branch | aluSRC | jump |
| Instruções tipo-R | 0000 | 1 | 0000 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Instrução Load | 0001 | 0 | 0001 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| Instrução Store | 0010 | Z | 0010 | Z | 0 | 0 | 1 | 0 | 0 | 0 |
| Instrução Jump | 0101 | Z | 0101 | Z | 0 | 0 | 0 | 1 | 0 | 1 |

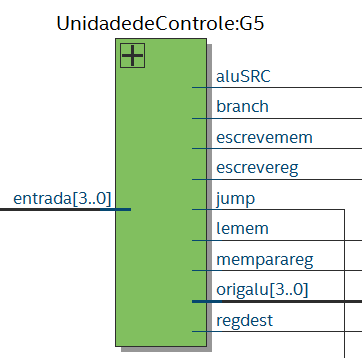


Figura 4 - Bloco simbólico do componente Unidade de Controle

### Memória de dados

A memória RAM é a nossa memória de dados onde serão armazenados valores de registradores e ela poderá ser tanto lida quanto escrita.

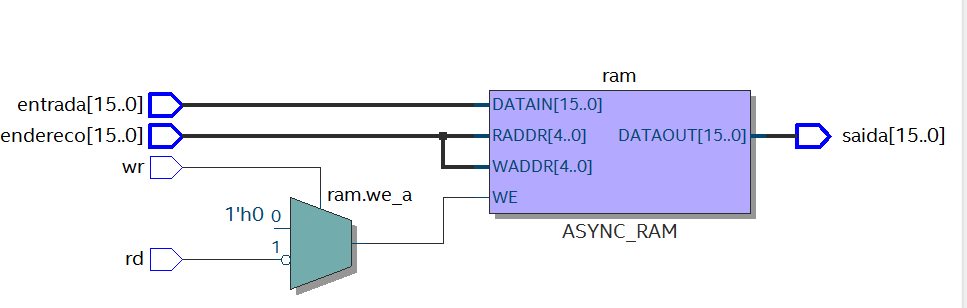


Figura 5 - Bloco simbólico do componente Memória RAM

### Memória de Instruções

A memória ROM é a representante da memória de Instruções dentro do projeto. Nela estão as nossas Instruções pré-setadas ao início da execução do projeto.

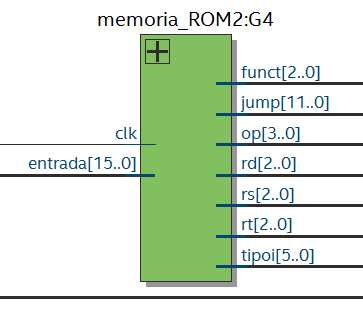


Figura 6 - Bloco simbólico gerado através do RTL da Memoria ROM

### Somador

Componente Simples com duas entradas de 16 bits onde serão somados os dois valores e serão jogados na saída. Esse padrão de somador é utilizado tanto no Somador do PC quanto o Somador de Jump.

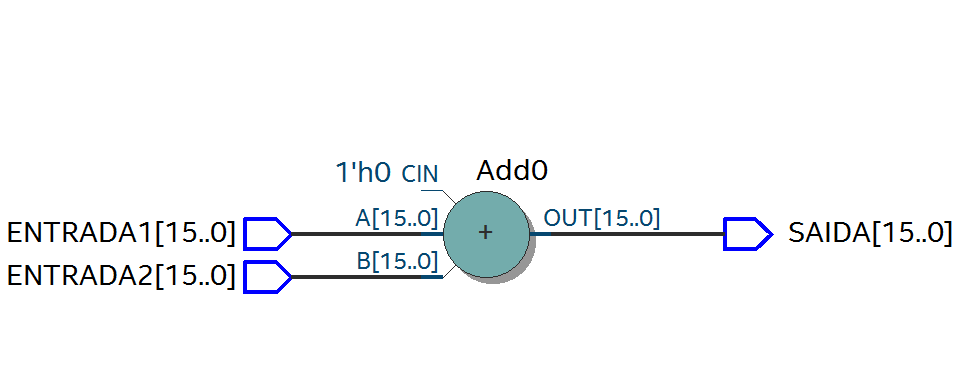


Figura 7 - Bloco simbólico do componente Somador

### QAndBIT

QAndBIT é uma porta lógica AND, ou seja, é um circuito lógico que possui ao menos duas entradas, e que fornece na saída o valor lógico 1 apenas quando todas as entradas também tiverem o nível lógico igual a 1 (alto).

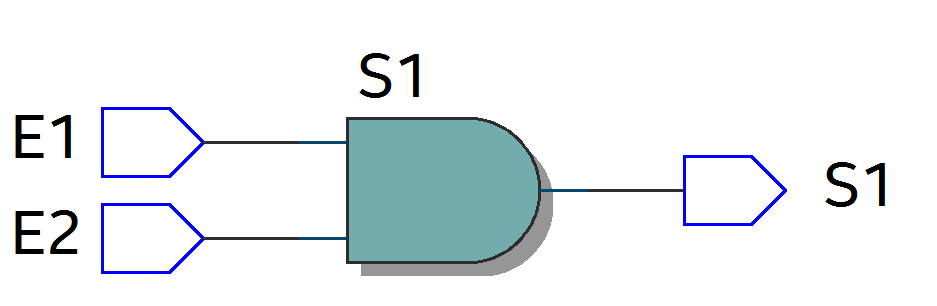


Figura 8 - Bloco simbólico do componente QAndBIT

### Muliplexado2x1\_16bits

Multiplexador padrão, com duas entradas de 16 bits e um seletor que pode assumir valores 0 e 1, e de acordo com o valor do seletor, a saída recebe um dos dois valores.

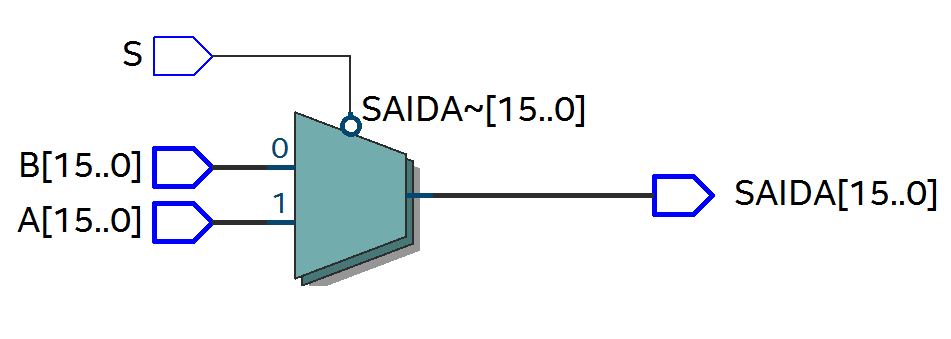
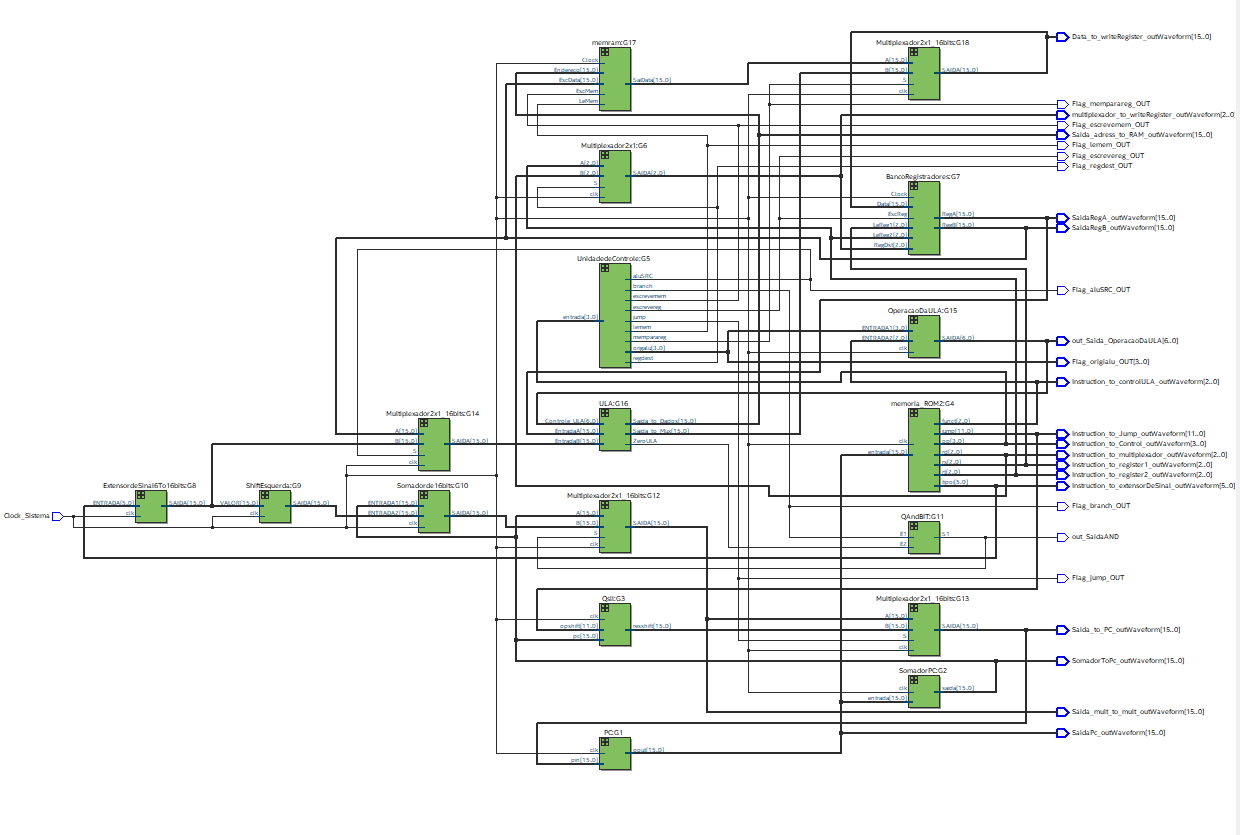


Figura 9 - Bloco simbólico do componente Multiplexado2x1\_16bits

## Datapath

A peça central no desenvolvimento no projeto, onde foram implementados os port maps de todos os componentes assim como os barramentos, permitindo a comunicação entre cada componente. Junto com a Unidade de Controle, foi possível de organizar todos eles de acordo com as instruções passadas.



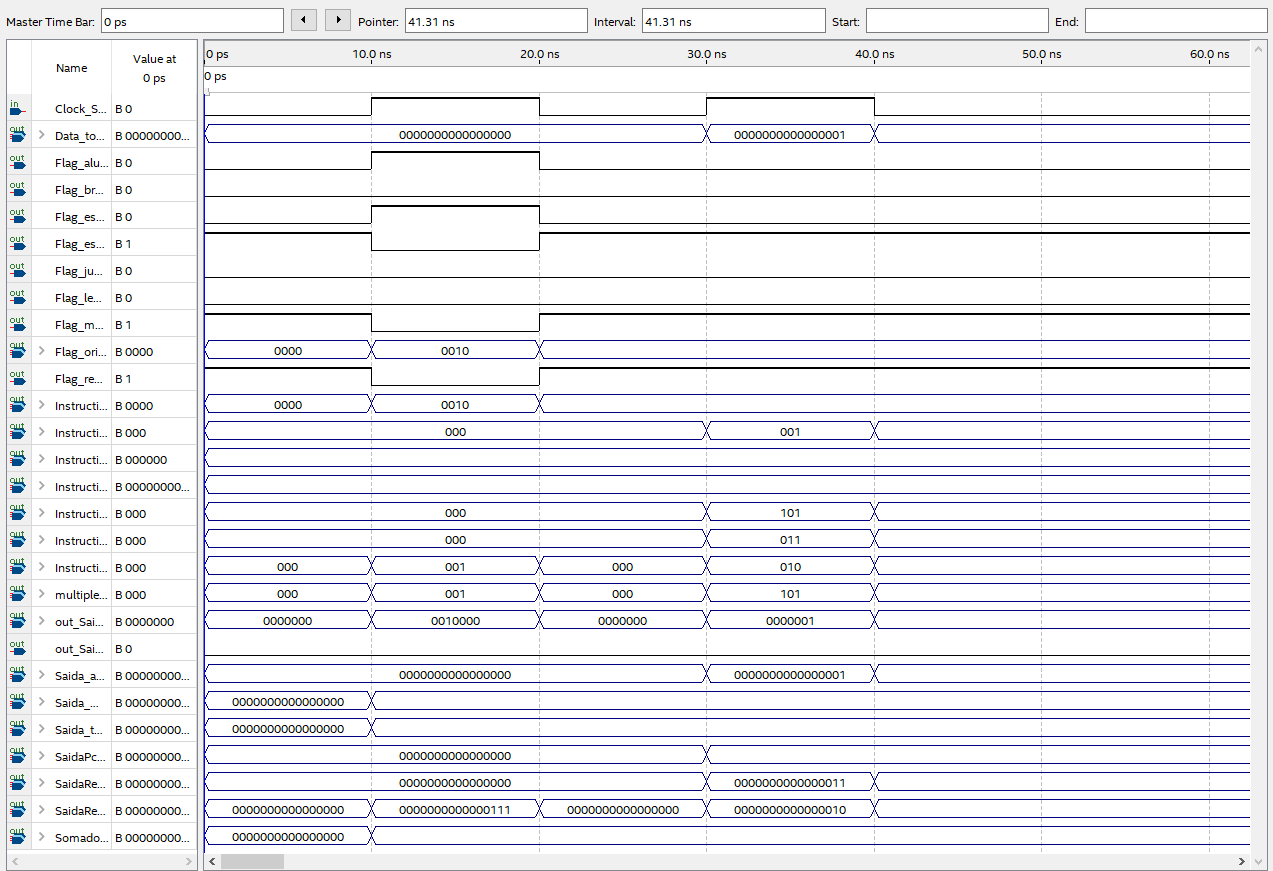
# Simulações e Testes

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 Torre utilizaremos como exemplo o código

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Endereço** | **Linguagem de Alto Nível** | **Binário** | | | | | | | |
| Opcode | Reg1 | | Reg2 | RegD | | | Funct |
| Endereço | | | | | | |
| 0 | SW R0, R1(0) | 0010 | 000 | | 001 | | XXX | | XXX |
| 000000 | | | | | | |
| 1 | SUB R3, R2, R5 | 0000 | 011 | 010 | | | | 101 | 001 |

Como foi dito durante a apresentação do processador, ele não havia capacidade de fazer duas operações de tipos diferentes em um mesmo WaveForm, porém, com um pouco de persistência foi possível contornar este problema, mostrando sua funcionalidade através deste código simples. Em contrapartida, as instruções do tipo J não estão em pleno funcionamento, por falha nos desvios e concatenação dos dados.

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



Estes são os valores que saem dos registradores

Neste caso especifico, os valores são 3 e 2

Estes são os pinos de dados para o registrador de escrita, ou seja, o resultado da operação na ULA

Nesse clock é executada a instrução SUB do Registrador 3 com o Registrador 2 e guardando no Registrador 5

Nesse clock é executada a instrução Store do Registrador 1 no endereço 0 da Memória Ram

Figura 11 - Resultado na Waveform.

# Considerações finais

Este trabalho apresentou o projeto e implementação do processador Uniciclo de 16 bits denominado como Torre. O projeto foi uma ótima abordagem prática de todos os conceitos vistos em sala de aula, fazendo com que, o aluno não só se contente em estudar o assunto teoricamente, como também desperte o interesse do mesmo à parte prática.

Apesar da falha na implementação total do projeto, a experiência adquirida será levada para frente, e até mesmo um possível desenvolvimento futuro da plataforma aqui desenvolvida.

De vários conceitos vistos em sala de aula, aplicamos toda a ideia básica de distribuição de bits para a construção de Instruções únicas para nosso processador. Além disso, o projeto nos fez aprofundar muito mais no funcionamento do processador, nos fez desenvolver questões que se nos mantivéssemos somente na parte teórica, não as teríamos desenvolvido. Em resumo, foi um projeto de implementação desafiadora que testa todo o conhecimento absorvido em aula, e é realmente um ótimo validador da capacitação do aluno frente a disciplina de Arquitetura e Organização de Computadores.