

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR UNICICLO DE 16 BITS**

**ALUNOS:**

**Luigi Muller Sousa Linhares – 2017009506**

**Matheus Fellype de Moura Silva – 2017027110**

**Tarlison Sander Lima Brito – 2017013008**

**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 UNICICLO DE 16 BITS**

**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 usando os conhecimentos obtidos em sala de aula. Este trabalho mostrará como foi realizado essa implementação, desde a IDE que foi utilizada para criar o processador, da criação das instruções e do datapath até o desenvolvimento de cada componente onde cada um será descrito no decorrer do trabalho, chegando assim por fim na montagem do processador e na realização de testes para examinar seu funcionamento. O processador é umas das peças mais importantes, tendo em vista que ele gerencia todo o sistema computacional controlando as operações realizadas por cada unidade funcional; além disso temos que sua função é executar instruções que estão presentes na memória principal.

Somado a isto, podemos perceber que o processador é composto por unidade de controle onde ocorre a busca, a decodificação, a execução das instruções que são as palavras da linguagem de máquina, a unidade lógica, como portas lógicas e aritmética que executa as operações lógicas e aritméticas, ou seja, contas matemáticas tendo nesse processador o seu diferencial por conter multiplicação com ponto flutuante, e registradores onde são armazenados dados e endereços. Componentes estes são ligados por barramentos que faz com que haja um meio de comunicação compartilhado entre eles. Os testes do funcionamento de cada componente do processador foram realizados no IDE do Quartus (Quartus Prime 18.0) Lite Edition, onde foram observadas a RLT Viewer de cada componente assim como os testes de suas waveforms para que o processador funcione com eficiência.

**Conteúdo**

[1 Especificação 7](#_Toc444681789)

[1.1 Plataforma de desenvolvimento 7](#_Toc444681790)

[1.2 Conjunto de instruções 8](#_Toc444681791)

[1.3 Descrição do Hardware 10](#_Toc444681792)

[1.3.1 ALU ou ULA 10](#_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 12](#_Toc444681797)

[1.3.6 Memória de Instruções 12](#_Toc444681798)

[1.3.7 Somador 13](#_Toc444681799)

[1.3.8 And 13](#_Toc444681800)

[1.3.9 Mux\_2x1 13](#_Toc444681801)

[1.3.10 PC 14](#_Toc444681802)

[1.3.11 ZERO 14](#_Toc444681803)

[1.4 Datapath 14](#_Toc444681804)

[2 Simulações e Testes 16](#_Toc444681805)

[3 Considerações finais 18](#_Toc444681806)

**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 11](#_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 13](#_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 14](#_Toc444681817)

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

[Figura 8 – Figura RTL do processador uniciclo 16 bits 15](#_Toc444681817)

**Lista de Tabelas**

[Tabela 1 – Tabela que mostra a lista de Op e Funct utilizadas na instrulção 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 istrução tipo-j pelo processador 9](#_Toc444681824)

[T abela 4 – Detralhes das flags de controle do processador 12](#_Toc444681824)

[T abela 5 – Simulação e teste do processador 16](#_Toc444681824)

# Especificação

É apresentado nesta seção o conjunto dos itens utilizados para o desenvolvimento do processador uniciclo de 16 bits, 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: ......

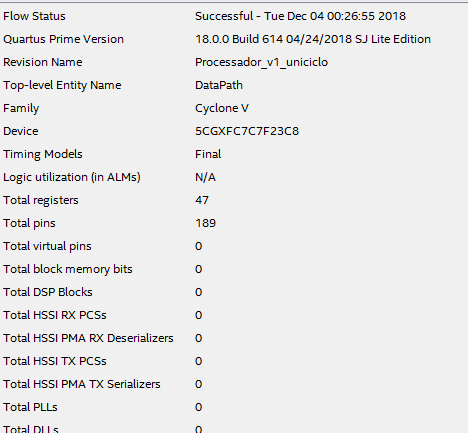


Figura 1 - Especificações no Quartus

## Conjunto de instruções

O processador uniciclo de 16 bits possui 8 registradores: S0, S1, S2, S3, S4, S5, S6. 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:

* **Op**: a operação básica a ser executada pelo processador, também chamado de código de operação;
* **Rd**: o registrador do operando de destino. Ele recebe o segundo operando de destino.
* **Rf**: o registrador contendo o primeiro operando de origem;
* **Rs**: o registrador contendo o segundo operando de origem;
* **Funct**: Função. Esse campo seleciona a variante específica da operação no campo op e, às vezes, é chamado de código de função.

Tipo de Instruções:

**- Formato do tipo R:** Chamado de tipo-R (de Registrador), este formato simboliza o uso de registradores pela instrução, normalmente, instruções lógicas e aritmética.

Formato para escrita de código na linguagem Quantum:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Op | Rd | Rf | Rs | 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 |
| Op | Rd | Rf | Rs | Funct |

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

O número de bits do campo **Op** das instruções é igual a quatro e do **funct** igual a três, sendo assim obtemos um total () de 3 **op** (15-12) e 3 **Funct (2-0)** 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** $S0, $S1 , $S2 ou seja, $S0 = $S1 + $S2 |
| 0000 | 001 | SUB | R | Subtração | **sub** $S0, $S1 , $S2 ou seja, $S0 = $S1 - $S2 |
| 0000 | 011 | MULT | R | Divisão | **mult** $S0, $S1 , $S2 ou seja, $S0 = $S1 \* $S2 |

**- Formato do tipo I:** Chamado de tipo-I (de Imediato), este formato simboliza instruções de transferência de dados, normalmente, instruções de carga e armazenamento.

Formato para escrita de código na linguagem Quantum:

|  |  |  |  |
| --- | --- | --- | --- |
| Op | 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 |
| Op | Rd | Rf | Imediato |

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

O número de bits do campo **Op** das instruções é igual a quatro, sendo assim obtemos um total () de 4 **Op (15-12)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 2.

Tabela 2 – Tabela que mostra a lista de Op utilizada pelo processador uniciclo de 16bits.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Op** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0001 | ADDI | I | Soma imediata | **addi** $S0, $S1, 100, ou seja, $S0 = $S1 + 100 |
| 0010 | LW | I | Load Word | **lw** $S0, 100($S1), ou seja, $S0 = Memory[$S1 + 100] |
| 0011 | SW | I | Store Word | **sw** $S0, 100($S1), ou seja, Memory[$S1 + 100] = $S0 |
| 0111 | BEQ | I | Branch on equal | **beq** $S0, $S1, 25, ou seja, if($S0 == $S1) go to PC + 4 + 100 |
| 0101 | BNQ | I | Branch on not equal | **bne** $S0, $S1, 25, ou seja, if($S0 != $S1) go to PC + 4 + 100 |

**- 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 Quantum:

|  |  |
| --- | --- |
| 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 **Op** das instruções é igual a quatro, sendo assim obtemos um total () de 1 **Op (15-12)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

Tabela 3 – Tabela que mostra a lista de Op utilizadas pelo processador uniciclo de 16bits.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Op** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 1111 | J | J | Jump | **J** 2500, ou seja, go to 10000 |

## Descrição do Hardware

É apresentado nesta seção a exposição dos componentes do hardware que integram o processador, incluindo a descrição de suas funcionalidades, valores de entrada e saída.

### ALU ou ULA

O componente ULA (Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas, considerando este processador: soma, subtração, multiplicação utilizando ponto flutuante. Adicionalmente o QALU efetua operações de comparação de valor como maior ou igual, menor ou igual, somente maior, menor ou igual. O componente ULA recebe como entrada três valores: **EntradaA** – dado de 16bits para operação; **EntradaB** - dado de 16bits para operação e **Controle\_ULA** – identificador da operação que será realizada de 7bits. A ULA também possui três saídas: **ZeroULA** – identificador de resultado (2bit) para comparações (1 se verdade e 0 caso contrário); **Saida\_to\_Mux** – identificador de overflow caso a operação exceda os 8bits; e **Saida\_to\_Dados** – saída com o resultado das operações aritméticas.

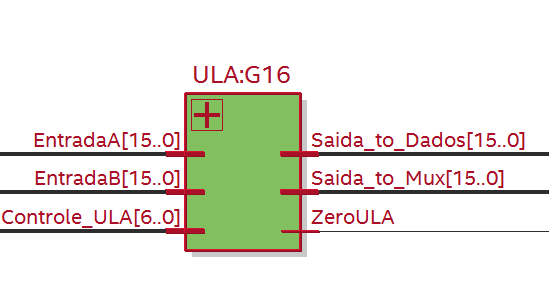


Figura 2 - Bloco simbólico do componente ULA

### BancoRegistradores

Um banco de registradores é uma coleção de registradores em que qualquer registrador pode ser lido ou escrito especificando o número do registrador no banco. O banco de registradores contém o estado dos registradores do computador.

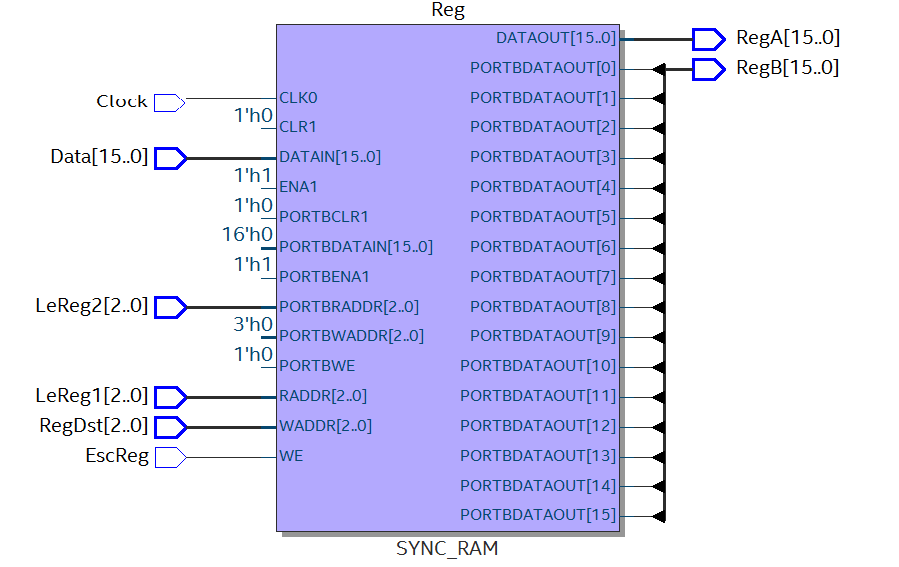


Figura 3 - Bloco simbólico do componente BancoRegistradores

### Clock

O clock é a referência de tempo necessária a CPU. Circuito eletrônico oscilador que gera uma forma de onda quadrada, essencial para o sequenciamento das operações eletrônicas realizadas pela CPU. Está relacionado com a frequência (taxa) de operação do processador.

### UnidadedeControle

A Unidade de Controle tem a função de dirigir e coordenar as atividades do processador em relação às demais unidades do sistema. Todas as atividades internas de uma determinada máquina são controladas pela UC. AS funções básicas da Unidade de Controle são, por exemplo, controle de entrada de dados, interpretação de cada instrução de um programa, coordenação de armazenamento de informações, análise das instruções dos programas, controle de saída de dados, decodificação dos dados, etc. Esse controle é feito através das flags de saída abaixo:

* **regdest**: Decide qual registrador será escolhido para leitura.
* **origalu**: Envia a operação para a operação para a ULA.
* **memparareg**: Decide se o dado a ser guardado no registrador de destino será da ULA ou da memória de dados.
* **escrevereg**: Controla se haverá escrita no banco de registradores.
* **lemem**: controla se o dado da memória de dados sairá na saída dela.
* **entrada:** Informação com o opcode.
* **jump:** controla se haverá desvio condicional;
* **aluSRC:** controla o multiplexador que decide se envia o segundo registrador de leitura ou o valor imediato.
* **branch:** decide se o PC recebe o endereço do jump ou a outra opção.

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 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| Instruçãos Load | 0001 | 0 | 0001 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Instruçãos Store | 0010 | Z | 0010 | Z | 0 | 0 | 1 | 0 | 0 | 0 |
| Instrução Jump | 0101 | Z | 0101 | Z | 0 | 0 | 0 | 1 | 0 | 1 |

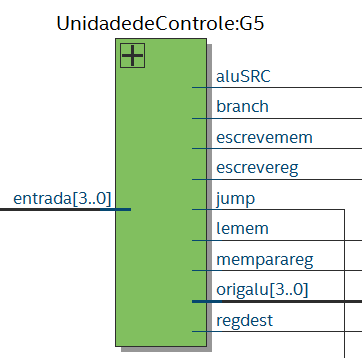


Figura 4 - Bloco simbólico do componente UnidadedeControle

### Memória de instruções

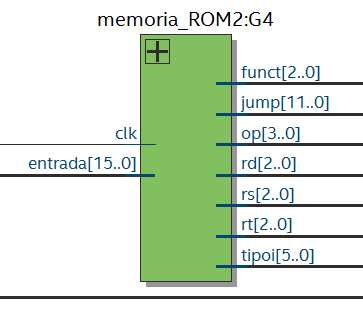


Figura 5 - Bloco simbólico do componente memoria\_ROM2

### Memória de dados

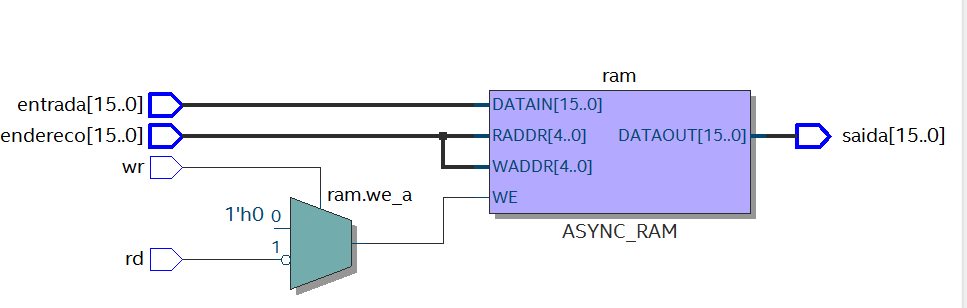


Figura 6 - Bloco simbólico do componente memram

### Somador

É o componente onde será feita a operação matemática de adição a fim de incrementar o PC. Ele possui duas entradas, ENTRADA1 e ENTRADA2, as duas sendo de 16bits e uma saída, SAIDA também de 16bits.

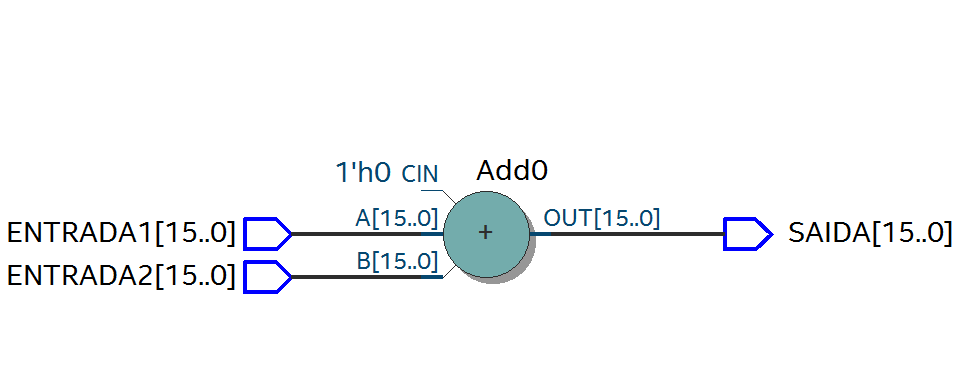


Figura 7 - Bloco simbólico do componente Somador

### 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).

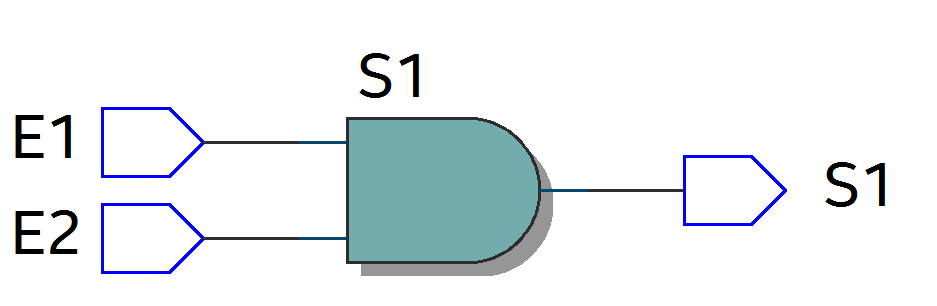


Figura 8 - Bloco simbólico do componente QAndBIT

### Muliplexado2x1\_16bits

Circuito combinacional que têm a finalidade de selecionar, a partir das variáveis de seleção, uma de suas entradas, conectando-a eletronicamente a sua única saída.

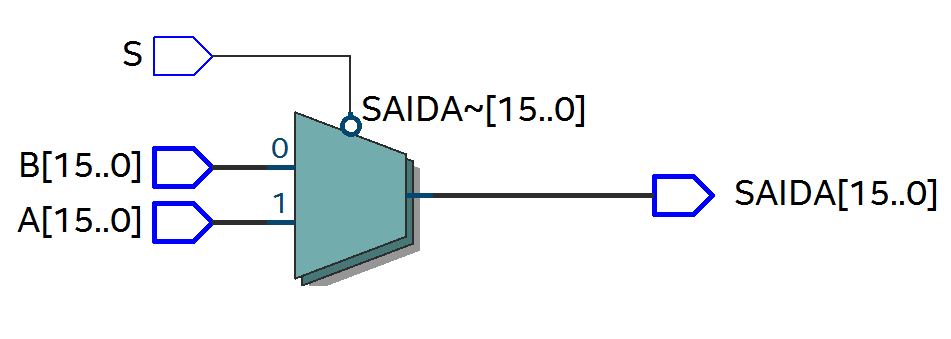


Figura 9 - Bloco simbólico do componente Multiplexado2x1\_16bits

### PC

O PC contém o endereço de memória que será utilizado para buscar a próxima instrução a ser executada pela CPU. Por se tratar de um processo de busca de instruções, o PC fica instalado na Unidade de Controle. O PC possui conexão direta com o barramento de endereços

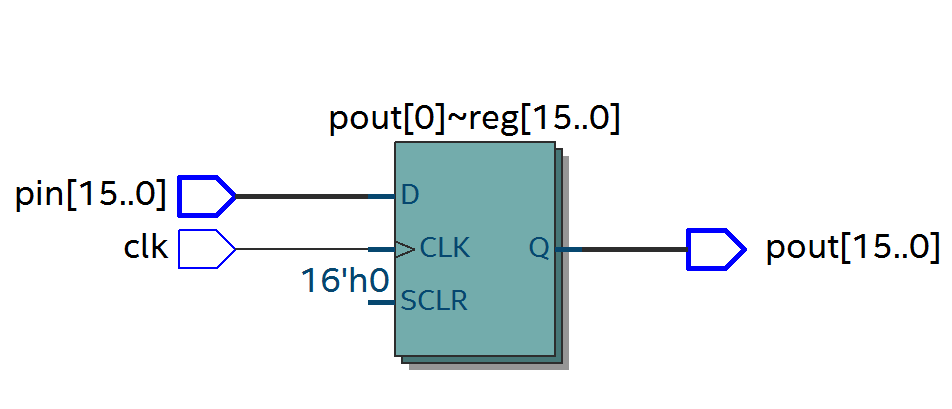


Figura 10 - Bloco simbólico do componente PC

### ZERO

Indica se o resultado da operação aritmética ou lógica. Se indicar 1 o resultado é igual a zero, e se indica 0 o resultado é diferente de zero. Ela é usada na função beq e bne, essa flag vai para o AND e significa que o pulo vai ser tomado, ou seja, a próxima instrução vai ser o endereço do pulo. Ele irá mandar ativado dependendo da instrução, se for uma instrução de beq era mandar ativado para fazer o pulo se forem iguais, se for beq e não forem iguais não manda ativado; no bne, se são iguais ele manda zero, se for diferente ele manda 1. E nos outros casos onde a operação não é realizada, ou seja, não é usada, ele já manda 0.

## 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. Ou seja, um caminho de dados com uma coleção de unidades funcionais, como lógicas aritméticas e multiplicadores, que executam operações de processamento de dados, registros e barramento.

**Figura RTL do processador uniciclo de 16 bits**

# 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 5 - 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 11 - Resultado na waveform.

# Considerações finais

Este trabalho apresentou o projeto e implementação do processador uniciclo de 16 bits como projeto final de disciplina de arquitetura de computadores, tendo o objetivo de colocar todo aprendizado obtido em prática. O projeto requereu amplos assuntos da disciplina, afim de que tenhamos um desenvolvimento mais aprimorado do assunto, foram seguindo vários passos começando desde criação das instruções, do desenho do datapath, até a programação dos componentes do processador partindo assim para a sua montagem. O processador contém todos os requisitos para ser assim denominado, por exemplo, contém não só operações aritméticas, como memória de dados e instruções, desvios condicionais, controle de dados, etc.

Somado a isto, foi abordado no trabalho como ocorreu a criação do processador, no começo onde se foi criada as instruções para cada tipo de operação que seria feito, por exemplo, tipo-R, tipo-I e tipo-J. Depois disso, o datapath foi desenvolvido para melhor visualização dos componentes com seus devidos barramentos, por fim na criação dos componentes e sua junção para a implementação final do processador. O projeto foi algo produtivo, onde cada conhecimento adquirido e implementado neste processador ajudará no desenvolvimento quanto ao conhecer mais aprofundado do assunto, saindo da parte mais superficial da descrição dos componentes até a sua funcionalidade em si por debaixo dos panos.