

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: Prolag**

**ALUNOS: Leonam de Sousa Silva – 2022007093**

**Ângelo Garcia Fernandez 2022010034**

**Guilherme Almeida da Luz- 2022010043**

**Novembro de 2023**

**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: Prolag**

**Novembro de 202****3**

**Boa Vista/Roraima**

**Resumo**

**Este projeto aborda a elaboração e implementação do processador uniciclo de 16 bits denominado “Prolag” baseado na arquitetura do processador MIPS. Programado na linguagem VHDL, que é utilizada de forma extensiva para o design de circuitos digitais. A implementação do projeto foi realizada na IDE Quartus Prime Lite Edition v18.1. O processador conta com Program Counter, Memória de Instruções (ROM), Banco de Registradores, Unidade Lógica Aritmética (ULA), Memória de Dados (RAM), Extensores de 6 para 16 bits e Extensor de 12 para 16 bits. Os testes**

**Dos componentes do processador “Prolag” foram realizados com o simulador Modelsim, tentando demonstrar o funcionamento das entradas e dos componentes.**

**Palavras-Chave: MIPS, VHDL, Processador, 16 Bits, Quartus Prime Lite.**

**Conteúdo**

[1 Especificação 7](#_Toc444681789)

[1.1 Plataforma de desenvolvimento 7](#_Toc444681790)

[1.2 Conjunto de instruções 8](#_Toc444681791)

[1.2.1 ALU ou ULA 9](#_Toc444681793)

[1.2.2 BDRegister 9](#_Toc444681794)

[1.2.3 Controle 9](#_Toc444681796)

[1.2.4 Memória de dados 10](#_Toc444681797)

[1.2.5 Memória de Instruções 10](#_Toc444681798)

[1.2.6 Somador 10](#_Toc444681799)

[1.2.7 And 10](#_Toc444681800)

[1.2.8 Mux\_2x1 10](#_Toc444681801)

[1.2.9 PC 10](#_Toc444681802)

[1.2.10 ZERO 11](#_Toc444681803)

[1.3 Datapath 11](#_Toc444681804)

[2 Simulações e Testes 13](#_Toc444681805)

[3 Considerações finais 14](#_Toc444681806)

4. Referências bibliograficas..........................................................................................................24

**Lista de Figuras**

Figura 1 – Especificações no Quartus…………………………………………………………………………….. 10

Figura 2 – ULA ………………………………………………………………………….........…............................ 13

Figura 3 – Unidade de controle ………………………………………………………………………….........….. 13

Figura 4 – RAM…………………………………………………………………………………………………………..….. 14

Figura 5 – ROM…………………………………………………………………………………………………………….... 15

Figura 6 – PC………………………………………………………………………………………………………………….. 16

Figura 7 – Datapht…………………………………………………………………………………………………………. 19

Figura 8 - Prolag (RLT View) ............................................................................................. 21

Figura 9 - Instrução do tipo R............................................................................................. 21

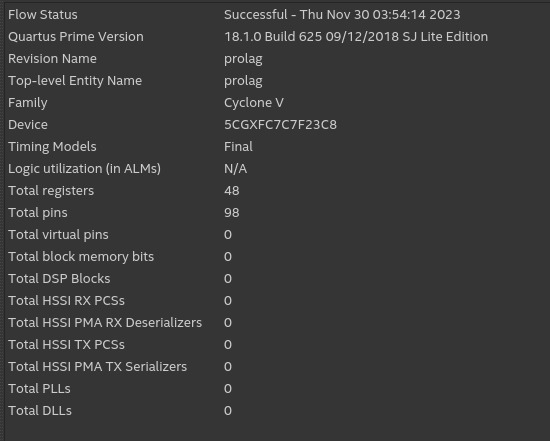
Figura 10 - Instrução do tipo I.............................................................................................. 22

Figura 11 - Instrução do tipo J............................................................................................. 23

# Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador Prolag e a descrição detalhada de cada etapa da construção do processador.

## Plataforma de desenvolvimento



Para a implementação do processador Prolag foi utilizado a IDE: Quartis Prime 18.1 Lite Edition

Figura 1 - Especificações no Quartus

## Conjunto de instruções

O processador Prolag possui registradores: S0 e S1. Que combinam com formatos de instruções do **tipo R, I e J** de 16 bits:

**Opcode**: a operação básica a ser executada pelo processador

**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: ( Add, Sub)** Instruções de operações aritméticas.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 bits | 3 bits | 3 bits | 3 bits | 0 bits | 3 bits |
| **Opcode** | **RS** | **RD** | **RT** | **SHAMT** | **FUNCT** |

**- Formato do tipo I :** **( Lw, Sw, Beq)** Instruções de Load, Store e Branch.

|  |  |  |  |
| --- | --- | --- | --- |
| 4 bits | 3bits | 3bits | 6bits |
| **OP** | **RS** | **RD** | **ENDEREÇO** |

**- Formato do tipo J : ( jump)** Instrução do tipo jump.

|  |  |
| --- | --- |
| 4 bits | 12bits |
| **OP** | **SALTO** |

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

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total () de 3 **Opcodes (0-1)** que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

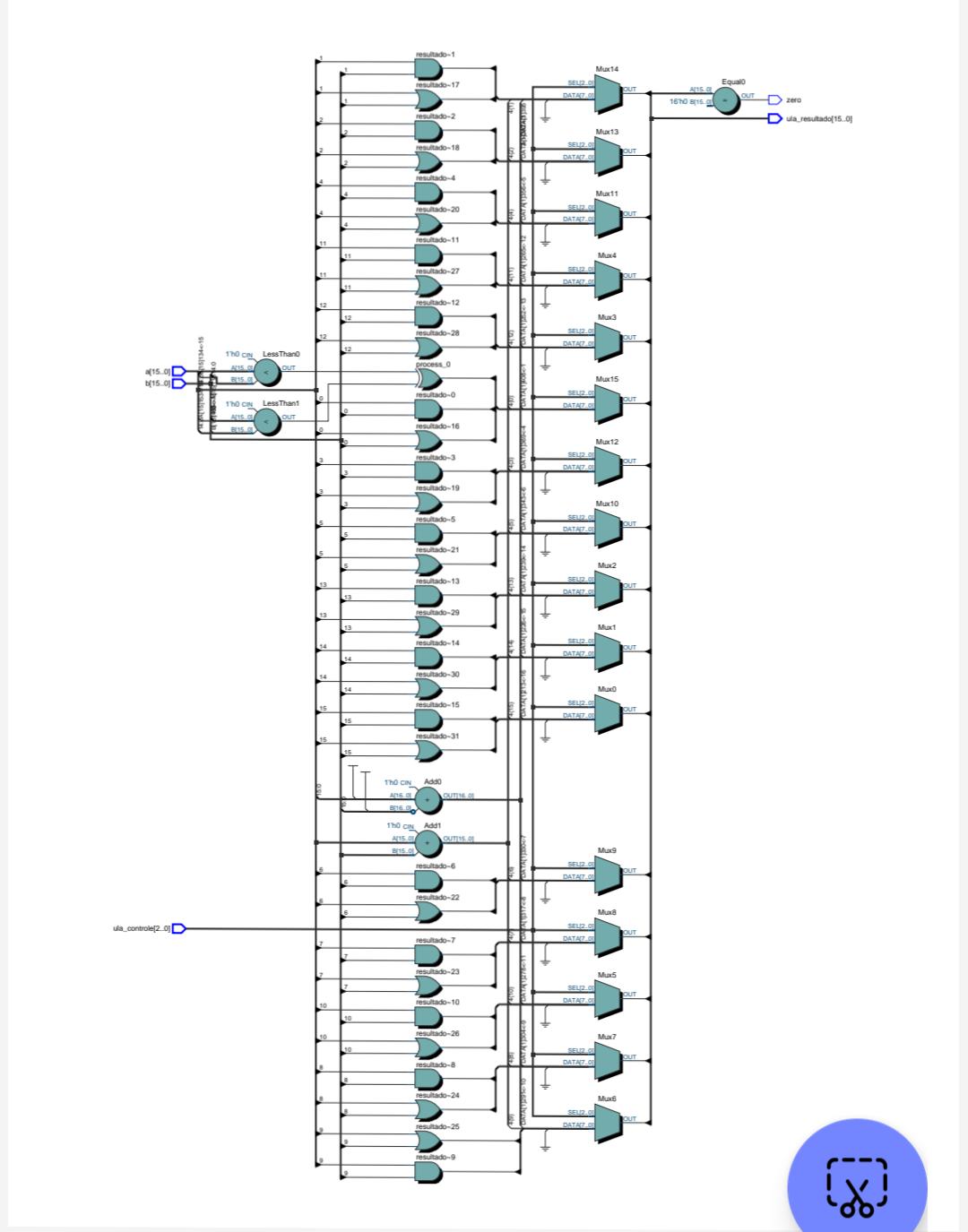
* + 1.  **ULA**

Figura 2 – ULA

**Branco de Registradores**

BDRegisterO BDRegister guarda valores e/ou resultados das operações realizadas pela ULA

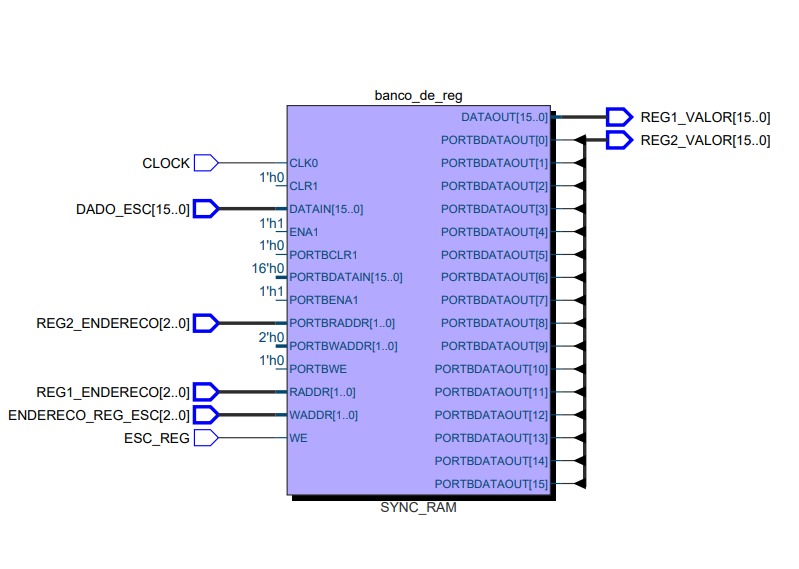


Figura 3- Banco De Registradores

**Clock**

Serve para manter os componentes funcionando apenas quando estiver ligado, portanto, Se for True/1.

### Controle

O componente Control tem como objetivo realizar o controle de todos os componentes do processador de acordo com o opcode.

### Memória de dados

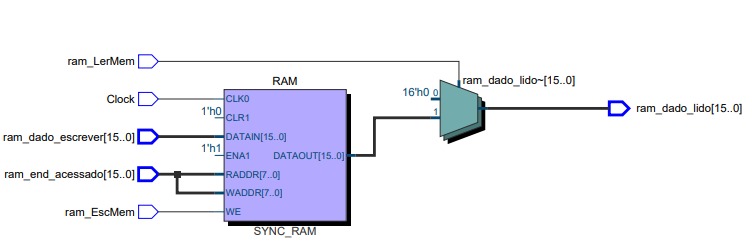


Figura 4 - RAM

Armazena os valores na memória e os disponibiliza para serem salvos nos registradores.

Tendo como entrada dado\_escrever (que define o que vai ser salvo na memória);

Endereco – acaessado (onde o valor será salvo na memória); EscMem – flag que define

Se algo vai ser escrito na memória ou não; LeMem – flag que define se algo vai ser lido

Da memória e passada para o registrador; clk – clock. E ele retorna com o que foi lido na

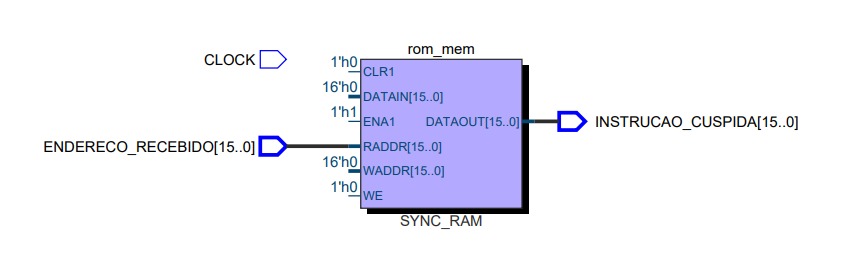
Memória – dado\_lido.

### Memória de Instruções

Onde todas as instruções serão executadas e é a partir dela que o opcode vai para a

Unidade de controle e que sabemos quais registradores usar. A ROM recebe um

Endereço que vem do PC e executa o que está nesta instrução.

. Figura 5 -ROM

### Somador

Soma cada PC para que ele execute as instruções

### And

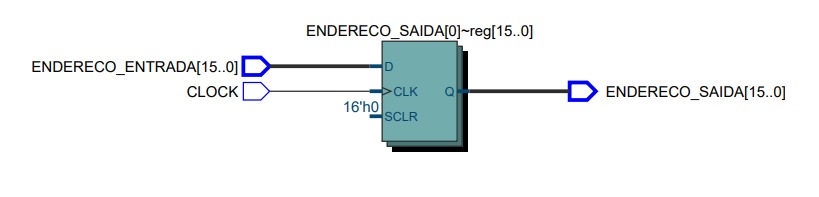
Verifica se vai ou não ocorrer um branc

### Mux\_2x1

O multiplexador auxilia a decidir que trilha será usada dependendo do valor da flag que está recebendo.

### PC

Responsável em mandar o endereço para a próxima instrução para a memória de instruções.

 Figura 6 – Pc

### ZERO

Identificador de resultado (1bit) para comparações (1 se verdade e 0 se falso)

## Datapath

Figura 7 - Datapaht

**1.4.Prolag. RLT View**

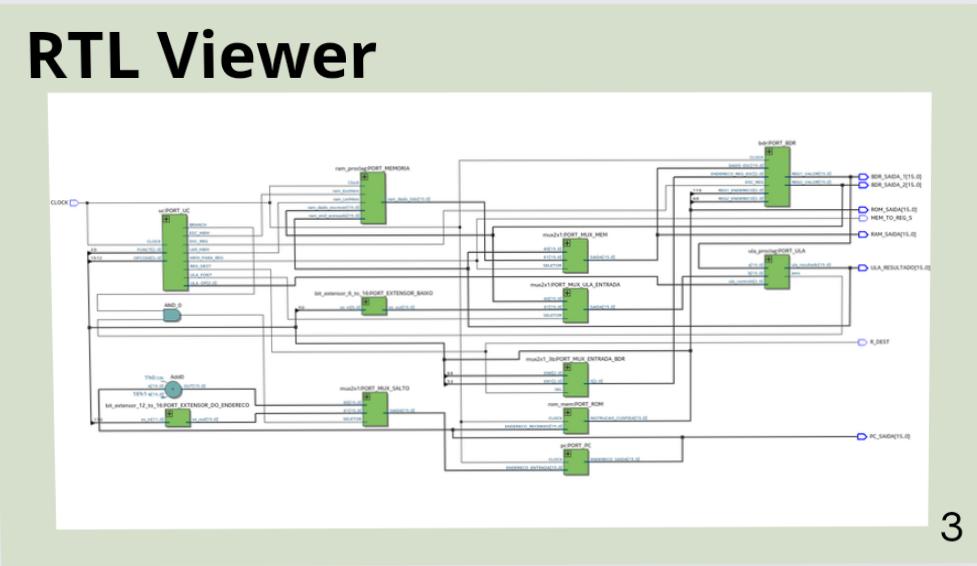
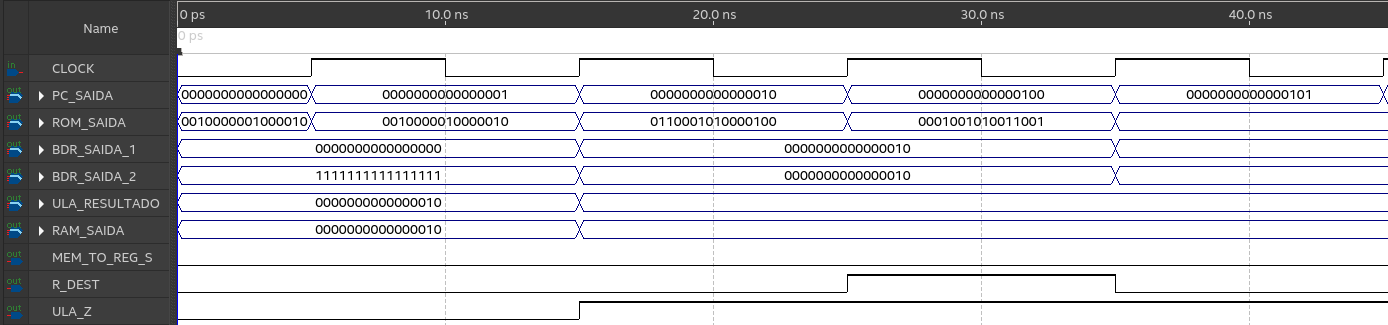
****

Figura 8 - Prolag (RLT View)

# 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.

Figura 9 – Instrução Tipo R

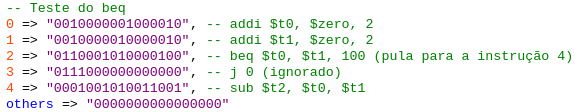
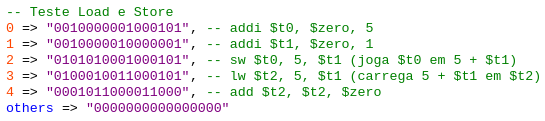
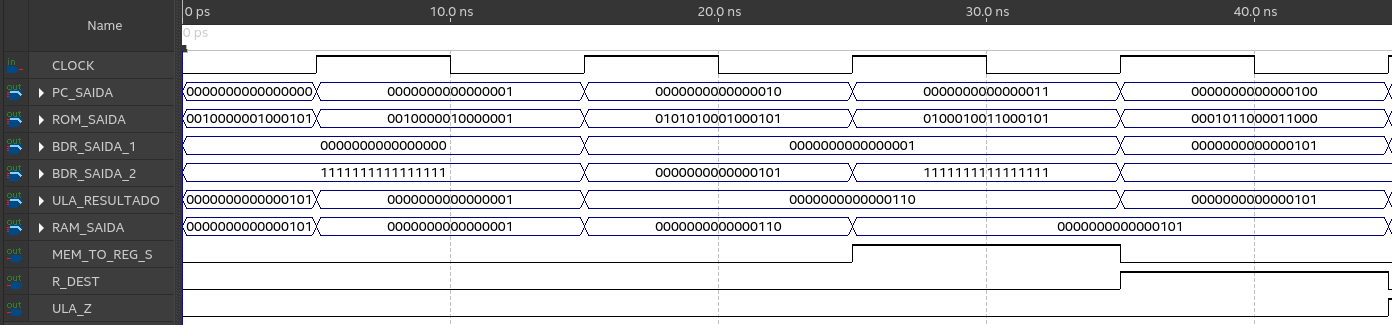
No teste acima, vemos a inicialização de ambos os registradores $t0 e $t1 com a constante 2. Depois, é realizado um beq (branch equal) comparando os valores dos registradores inicializados, para, se forem iguais, realizar o salto para a instrução no endereço 4 da memória de instruções, o que nesse caso é verdadeiro. No final, temos a subtração do resultado de $t0 e $t1 sendo armazenado em $t2 (nesse caso, o resultado é zero).

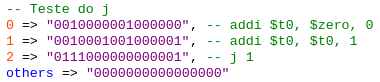
Figura 10 **–** Instruções do tipo I

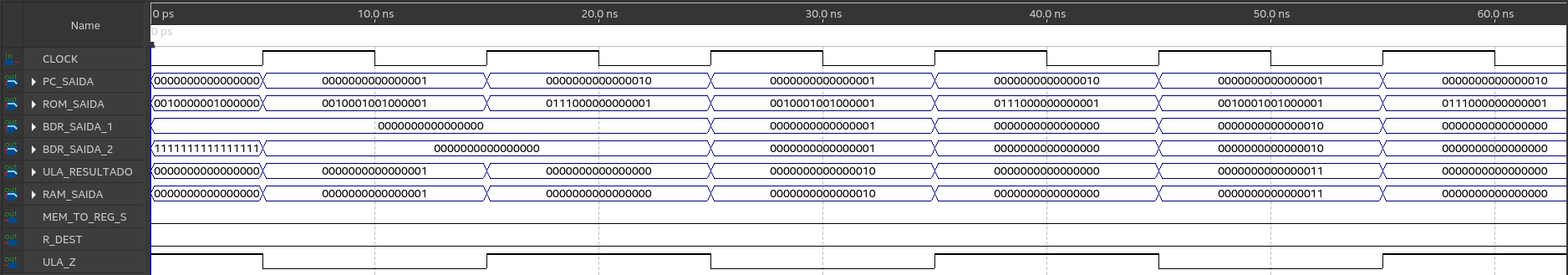
****



Agora, inicializamos $t0 $t1 com os valores 5 e 1, respectivamente. No endereço 2, armazenamos o valor de $t0 (5) em 5 + o valor de $t1 na memória de dados. Em seguida, carregamos o mesmo valor da memória(5 + $t1) para o registrador $t2. Afinal, realizamos uma soma de $t2 com $zero para visualizar que o valor do load foi carregado com sucesso.

Figura 11 –Instrução tipo J

****

Por fim, inicializamos o registrador $t0 com 0 (para limpar qualquer valor que pudesse ter ali), incrementamos ele em 1, através do addi $t0, $t0, 1, e pulamos de volta para a instrução 1 (o addi), repetindo indefinidamente, mostrando que o j está funcionando.

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

# Considerações finais

Este trabalho apresentou o projeto e implementação do processador uniciclo de 16 bits referente ao projeto final de disciplina de Arquitetura e Organização de computadores. O processador contém todos os requisitos para que tenha essa alcunha pois contém operações aritméticas, memória de dados, condicionais, controle de dados e afins.

1. **Referências bibliográficas**

**Organização e Arquitetura de Computadores – 4ª ed. David A. Patterson e John L. Hennessy**

**8 Bit Microprocessor Design Using VHDL. Disponível em:** [**https://www.youtube.com/watch?v=tTlhIDgAGYY**](https://www.youtube.com/watch?v=tTlhIDgAGYY)**. Acesso em: 30 nov. 2023.**

**‌FPGA4student.com. VHDL code for MIPS Processor. Disponível em:** [**https://www.fpga4student.com/2017/09/vhdl-code-for-mips-processor.html**](https://www.fpga4student.com/2017/09/vhdl-code-for-mips-processor.html)**. Acesso em: 24 nov. 2023.**