

**PODER EXECUTIVO**

**MINISTÉRIO DA EDUCAÇÃO**

**UNIVERSIDADE FEDERAL DE RORAIMA**

**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**

**ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES**

**RELATÓRIO DO PROJETO: PROCESSADOR Fluminense**

**ALUNOS:**

**Luiz Gustavo Dall’Agnol Cavalcante – 2021000632**

**Felipe de Sousa Borges – 20200020120**

**Yan Alberto Siqueira - 2020005518**

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

**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 Fluminense**

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

**Boa Vista/Roraima**

**Resumo**

Este trabalho aborda o projeto e implementação do processador Fluminense, que possui 4 registradores de 8 bits, construído no Logisim, sendo composto de 9 circuitos principais: PC, Banco de Registradores, Unidade de Controle, Seletor de Entrada, ULA – Controle, ULA, Processador principal, Pré-Seletor, Memória RAM e Valor de Memória 16 bits. Fluminense possui 5 formatos de instrução de 16 bits cada, R, RN, I, J e M. Nosso processador possui capacidade de suportar 16 instruções, porém só utilizamos de 14, sendo elas: AND, OR, NAND, NOR, SOMA, SUB, SOMAN, SUBN, BEQ, BNQ, J, LOAD, LOADN e STORE.

O processador foi testado a partir de experimentos simples de registro na memória, banco de dados. Por exemplo o teste Fibonacci, detalhado no restante do relatório.

**Conteúdo**

[1 Especificação 6](#_Toc444681789)

[1.1 Plataforma de desenvolvimento 6](#_Toc444681790)

[1.2 Conjunto de instruções 6](#_Toc444681791)

[1.3 Descrição do Hardware 8](#_Toc444681792)

[1.3.1 ULA 8](#_Toc444681793)

[1.3.2 Banco de registradores 9](#_Toc444681794)

[1.3.3 Clock 10](#_Toc444681795)

[1.3.4 Controle 10](#_Toc444681796)

[1.3.5 Memória de dados 11](#_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 13](#_Toc444681802)

[1.3.11 Select 14](#_Toc444681803)

[1.4 Datapath 15](#_Toc444681804)

[2 Simulações e Testes 16](#_Toc444681805)

[3 Considerações finais 17](#_Toc444681806)

**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. 10](#_Toc444681823)

[Tabela 3 - Código Fibonacci para o processador Quantum/EXEMPLO. 14](#_Toc444681824)

# Especificação

## Plataforma de desenvolvimento

Para a implementação do processador Fluminense foi utilizado o aplicativo: Logisim

## Conjunto de instruções

O processador Fluminense possui 4 registradores: R0, R1, R2 e R3. Assim como 5 formatos de instruções de 16 bits cada, Instruções do **tipo R, RN, I, J e M**, 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;
* **Número natural**: Algum membro do conjunto dos 256 números naturais disponíveis;
* **Endereço**: Um enderenço na memória de instrução ou na memória RAM;

Tipo de Instruções:

**- Formato do tipo R:** Este tipo de instruções envolve operações aritméticas e lógicas entre registradores, exceto BEQ e BNQ

Formato para escrita em código binário:

|  |  |  |  |
| --- | --- | --- | --- |
| 8 bits | 4 bits | 2 bits | 2 bits |
| 15-8 | 7-4 | 3-2 | 1-0 |
| NULL | Opcode | Reg1 | Reg2 |

**- Formato do tipo RN:** Este tipo de instruções envolve operações aritméticas entre registradores e números naturais e atribuição a números naturais a registradores

Formato para escrita em código binário

|  |  |  |  |
| --- | --- | --- | --- |
| 8 bits | 4 bits | 2 bits | 2 bits |
| 15-8 | 7-4 | 3-2 | 1-0 |
| Número  Natural | Opcode | Reg1 | NULL |

**- Formato do tipo I:** Este tipo de instrução compara o valor de dois registradores para determinar se ocorre ou não um salto na memória de instrução

Formato para escrita em código binário

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 bits | 4 bits | 4 bits | 2 bits | 2 bits |
| 15-12 | 11-8 | 7-4 | 3-2 | 1-0 |
| Endereço | NULL | Opcode | Reg1 | Reg2 |

**- Formato do tipo J:** Este tipo de instrução efetua o salto na memória de instrução

Formato para escrita em código binário

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 bits | 4 bits | 4 bits | 2 bits | 2 bits |
| 15-12 | 11-8 | 7-4 | 3-2 | 1-0 |
| Endereço | NULL | Opcode | NULL | NULL |

**- Formato do tipo M:** Este tipo de instrução ou armazena os valores da memória em registradores ou armazena os valores de registradores na memória

Formato para escrita em código binário

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 bits | 4 bits | 4 bits | 2 bits | 2 bits |
| 15-12 | 11-8 | 7-4 | 3-2 | 1-0 |
| Endereço | NULL | Opcode | Reg1 | NULL |

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

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total () de 16, porém só utilizamos 14 **Opcodes (0010-1111)** 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 Fluminense.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Opcode** | **Nome** | **Formato** | **Breve Descrição** | **Exemplo** |
| 0010 | AND | R | Conjunção bit a bit | **And** r0, r1 |
| 0011 | OR | R | Disjunção bit a bit | **Or** r0, r1 |
| 0100 | NAND | R | Negação da conjunção bit a bit | **Nand** r0, r1 |
| 0101 | NOR | R | Negação da disjunção bit a bit | **Nor** r0, r1 |
| 0110 | SOMA | R | Soma entre reg | **Soma** r0, r1 |
| 0111 | SUB | R | Subtração entre reg | **Sub** r0, r1 |
| 1000 | SOMAN | RN | Soma entre reg e número natural | **Soman** r0, 15 |
| 1001 | SUBN | RN | Subtração entre reg e número natural | **Subn** r0, 11 |
| 1010 | BEQ | I | Efetua jump se igualdade entre regs | **Beq** r0, r1 : 3 |
| 1011 | BNQ | I | Efetua jump se desigualdade entre regs | **Bnq** r0, r1 : 11 |
| 1100 | J | J | Efetua jump | **J** : 9 |
| 1101 | LOAD | M | Armazena valor da memória em reg | **Load** r0 : 15 |
| 1110 | LOADN | RN | Armazena número natural em reg | **Loadn** r0, 10 |
| 1111 | STORE | M | Armazena valor de reg na memória | **Store** r0 : 6 |
|  | | | | |

## Descrição do Hardware

### ULA

O componente ULA (Unidade Lógica Aritmética) tem como principal objetivo efetuar as principais operações aritméticas, dentre elas: soma e subtração. Adicionalmente a ULA efetua operações de comparação de valor como igualdade e desigualdade. O componente ULA 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. A ULA também possui duas saídas: **Resultado** – que é o valor que será enviado para ser armazenado no banco de registradores; **Boolean** – que é o valor que determinará se ocorrerá ou não um jump, nas instruções BNQ e BEQ. No processador Fluminense a ULA principal funciona com a integração de outro componente, a ULA-Controle.

(Abaixo, ULA)

Diagrama, Esquemático

Descrição gerada automaticamente

(Abaixo, ULA – Controle)

Diagrama

Descrição gerada automaticamente

### Banco de registradores

O componente Banco de Registradores tem como principal objetivo armazenar valores para uso imediato, ele possui quatro entradas: **R1** – enderenço do primeiro registrador e de certas instruções do registrador destino; **R2** – enderenço do segundo registrador; **ATIV** – Booleano que determina se ocorrerá a escrita ou leitura no BR; **Clock** – Possibilita a mudança de estado. BR também possui duas saídas: **RS** – que é o valor do primeiro registrador; **RT** – que é o valor do segundo.

Diagrama

Descrição gerada automaticamente

### Clock

Responsável por controlar as mudanças de estado dentro do processador. No processador Fluminense ele existe primariamente no componente da Unidade de Controle

### Controle

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

Diagrama

Descrição gerada automaticamente

Esse controle é feito através das flags de saída abaixo:

* **ULAcontrol**: 0-1.
* **Jump**: 0-1.
* **BRcontrol**: 0-1.
* **ClockBR**: 0-1.
* **ClockRAM**: 0-1.
* **RAMcontrol**: 0-1.

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

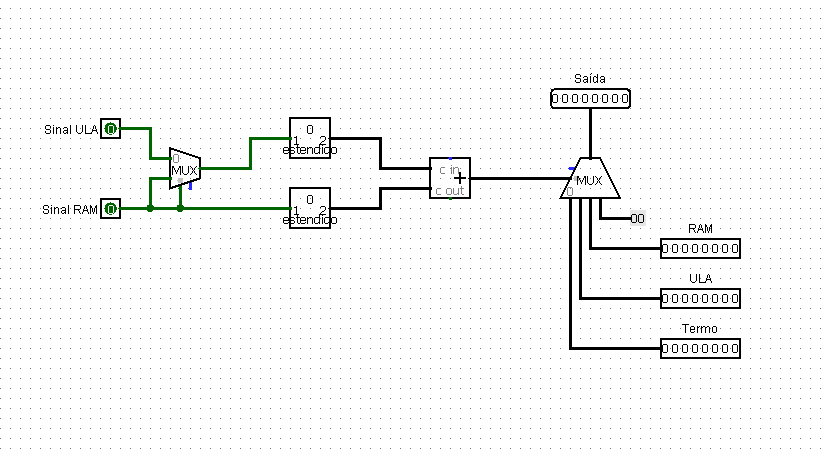
Tabela 2 - Detalhes das flags de controle do processador.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Comando | ULA control | Jump | BRcontrol | ClockBR | ClockRAM | RAMcontrol |
| And | 1 | 0 | 0 | 1 | 1 | 0 |
| Or | 1 | 0 | 0 | 1 | 1 | 0 |
| Nand | 1 | 0 | 0 | 1 | 1 | 0 |
| Nor | 1 | 0 | 0 | 1 | 1 | 0 |
| Soma | 1 | 0 | 0 | 1 | 1 | 0 |
| Sub | 1 | 0 | 0 | 1 | 1 | 0 |
| Soman | 1 | 0 | 0 | 1 | 1 | 0 |
| Subn | 1 | 0 | 0 | 1 | 1 | 0 |
| Beq | 1 | 0 | 0 | 0 | 0 | 0 |
| Bnq | 1 | 0 | 0 | 0 | 0 | 0 |
| J | 0 | 1 | 0 | 0 | 0 | 0 |
| Load | 0 | 0 | 1 | 1 | 0 | 0 |
| Loadn | 0 | 0 | 1 | 1 | 0 | 0 |
| Store | 0 | 0 | 0 | 0 | 1 | 1 |

### Memória RAM

A memória RAM tem como principal função armazenar dados de maneira mais permanente do que o banco de registradores. Ela recebe quatro entradas: **Endereço** – que é a localização da célula de memória que será lida ou escrita; **Input** – que é o valor que será registrado; **Ativ** – que irá determinar se a escrita ou a leitura será efetuada; **Clock** – Controla a mudança dos estados. Apenas uma saída, sendo o **Output** – que é o valor lido pela RAM. Exemplificado pelo componente seletor de entrada e pré-seletor abaixo.

(Abaixo, Seletor de entrada)

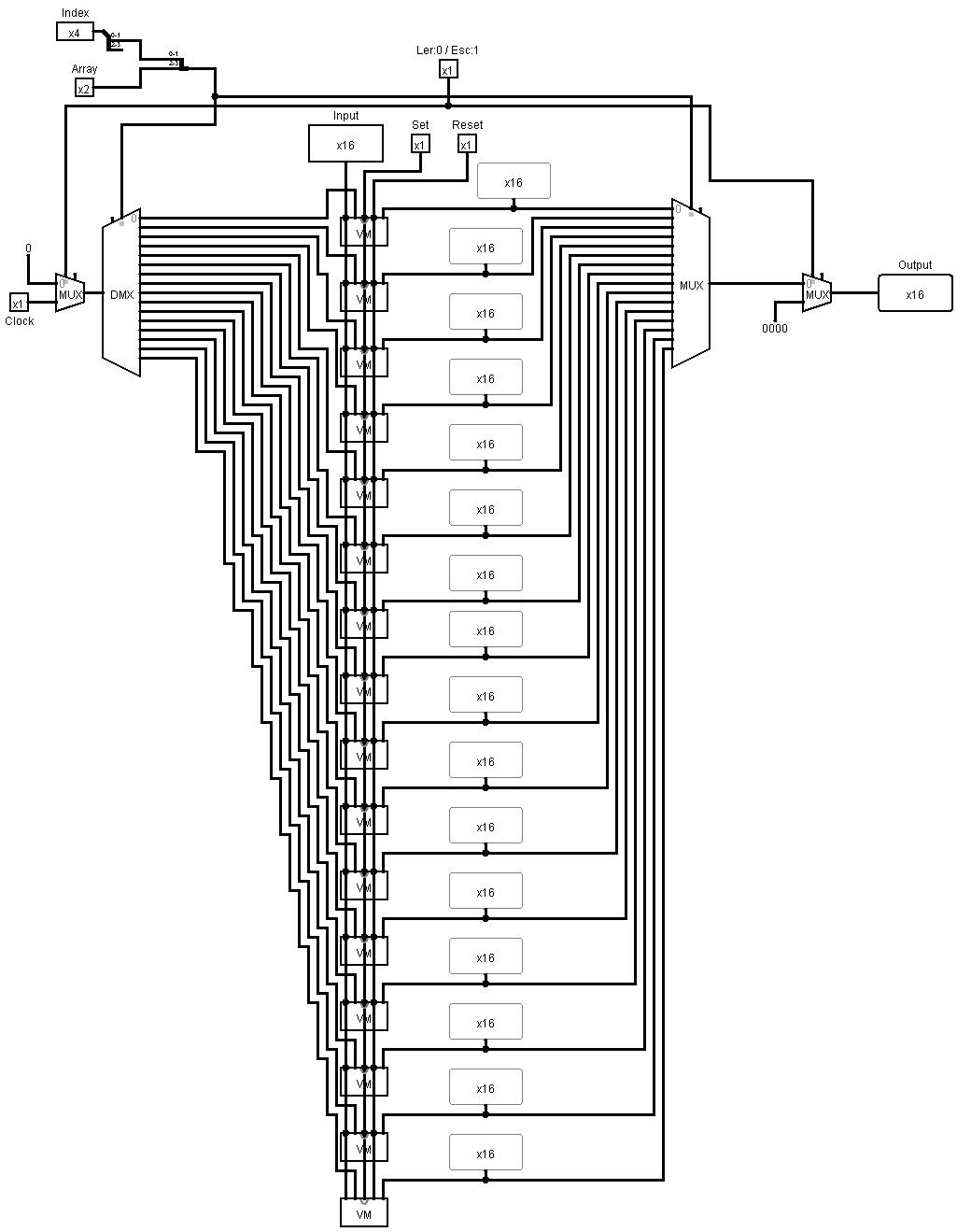


(Abaixo, Pré-seletor)

Diagrama, Esquemático

Descrição gerada automaticamente

(Abaixo, estrutura da memória RAM)



### Memória de Instruções

Tem como principal função armazenar as instruções que serão executadas. Subcomponente da memória RAM, ajudando a processar os 16 bits.

Diagrama, Esquemático

Descrição gerada automaticamente

### Somador

Sua principal função é a de somatória de bits.

### And

Tem como sua principal função a conjunção de bits.

### Mux\_2x1

Usado para determinar qual valor será utilizado.

### PC

Tem como principal função controlar a execução das instruções e possibilitar o salto

Diagrama, Esquemático

Descrição gerada automaticamente

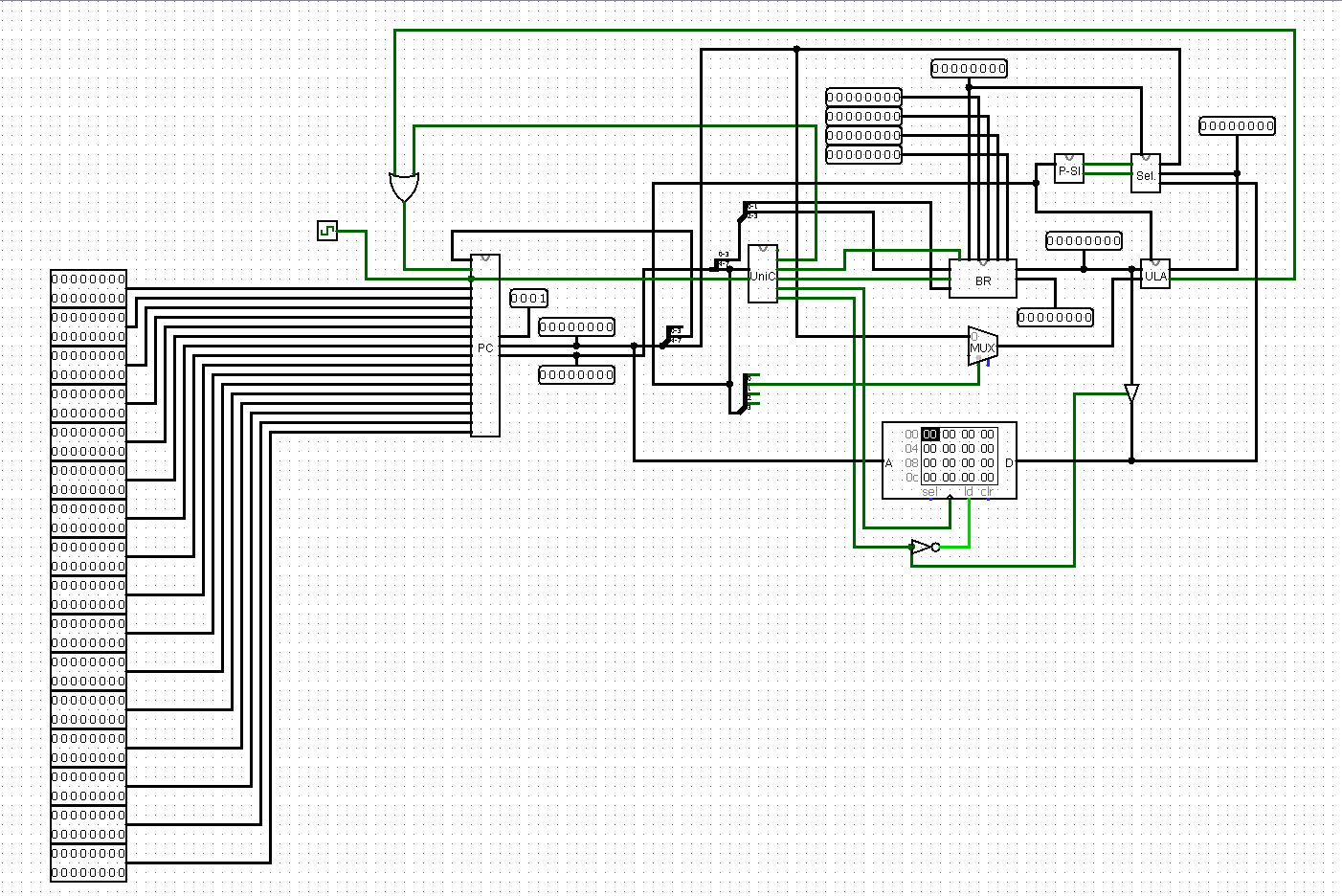
### Select

Seletor de entrada tem como principal função determinar qual valor será enviado ao banco de registradores. Possui 5 entradas: Termo – um valor obtido diretamente da instrução; ULA IN – valor obtido da ULA; RAM IN – valor obtido da RAM; Sinal ULA – valor que sinaliza preferência da ULA; Sinal RAM – valor que sinaliza preferência da RAM. Possui um valor de saída, sendo o Selecionado – que seria o valor selecionado

## 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...

(Processador principal)



# Simulações e Testes

Objetivando analisar e verificar o funcionamento do processador, efetuamos alguns testes analisando cada componente do processador em específico, em seguida efetuamos testes de cada instrução que o processador implementa. Para demonstrar o funcionamento do processador Fluminense utilizaremos como exemplo o código que guarda valores nos registradores, insere-os na memória RAM.

Tabela 3 - Código para o processador Fluminense.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Endereço** | **Linguagem de Alto Nível** | **Binário** | | | | | |
| Opcode | | Reg2 | | | Reg1 |
| Endereço | | | | | |
| Dado | | | | | |
| 0 | **Loadn** r0 : 13 | 1101 | | 00 | | | 00 |
| 00001101 | | | | | |
| 1 | **Loadn** r1 : 16 | 1101 | | 01 | | | 00 |
| 00010000 | | | | | |
| 2 | **Loadn** r2 : 11 | 1101 | | 11 | | | 00 |
| 00001011 | | | | | |
| 3 | **Loadn** r3 : 10 | 1101 | 10 | | 00 | | |
| 00001010 | | | | | |
| 4 | **Store** r0 : 6 | 1111 | | 00 | | | 00 |
| 00000100 | | | | | |
| 5 | **Store** r1 : 3 | 1111 | | 01 | | | 00 |
| 00000011 | | | | | |
| 6 | **Store** r2 : 1 | 1111 | | 10 | | 00 | |
| 00000001 | | | | | |
| 7 | **Store** r3 : 0 | 1111 | | 11 | | | 00 |
| 00000000 | | | | | |

**Esse teste alocou valores a cada um dos registradores e então os guardou em lugares determinados da memória.**

# Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 16 bits denominado de Fluminense. O grupo chegou à conclusão que o objetivo de aprendizado íntimo sobre a construção e funcionamento de um processador, fora alcançando plenamente. Graças a metodologia prática do projeto e avisos prévios da complexidade do trabalho, podemos realizar a atividade com paciência e aprender organicamente.

Apesar da dificuldade inicial de conectar os conteúdos aprendidos em sala de aula com a realidade prática do trabalho, concluímos que o tempo alocado fora suficiente para realizá-lo sem pressa ou aperto. Porém foram altas as dificuldades de reunião do grupo, por agendas conflituosas, resultando em um projeto com menos tempo de trabalho que o esperado.