PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE RORAIMA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES

RELATÓRIO DO PROJETO: PROCESSADOR N.I.N.A.

ALUNOS:
Jean Bertrand Paixão da Silva - 1201224412
Matheus Bedoni de Sousa - 2201424437

Março de 2017

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 N.I.N.A.

Março de 2017
Boa Vista/Roraima

Resumo

Esse é o nosso processador criado com a finalidade de aprender e entender como funciona a arquitetura e organização de computadores além de obter a nota necessária para passar na disciplina e assim poder passar para o próximo nível a fim de nos formarmos como bom profissionais de TI demos o nome do processador de N.I.N.A (Nanobytes Integrados Na Arquitetura).

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Especificação

Nesta seção é apresentado o conjunto de itens para o desenvolvimento do processador N.I.N.A. bem como a descrição detalhada de cada etapa da construção do processador.

1.1 Plataforma de desenvolvimento

Para a implementação do processador N.I.N.A. foi utilizado a IDE:Logsim 2.7.1

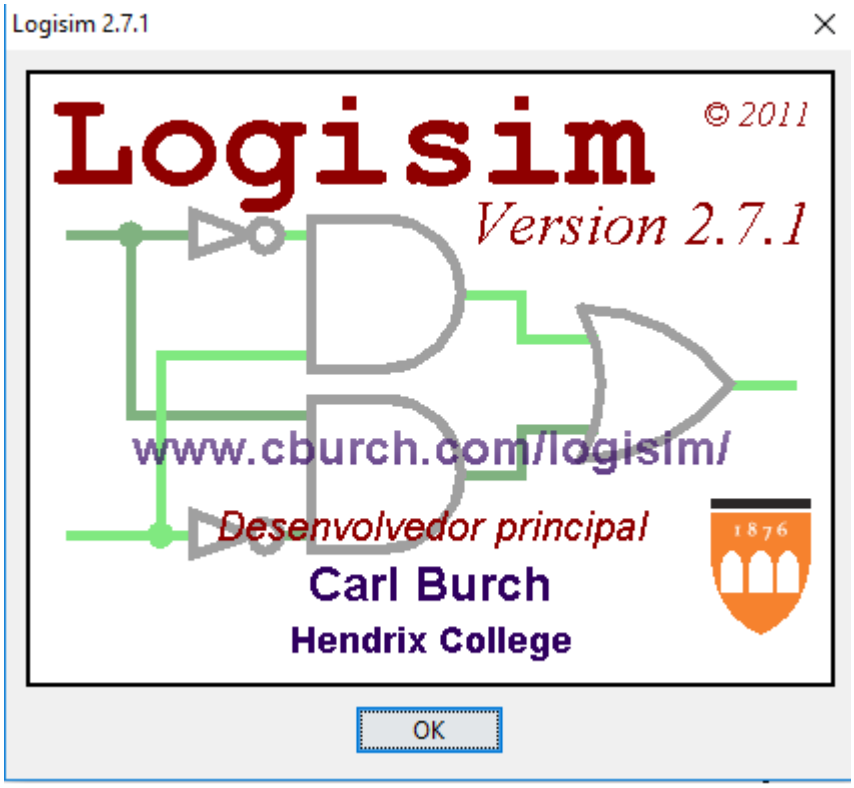

Figura 1 - Especificações no LogSim

1.2 Conjunto de instruções

O processador N.I.N.A. possui 16 registradores: S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14 S15. Assim como 3 formatos de instruções de 4 bits cada, Instruções do **tipo R, I, J**, 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;
- § **Reg2**: o registrador contendo o segundo operando fonte;
- § **Reg3:** adicionalmente para alguns tipos de instruções (ex. instruções do tipo R) é o registrador de destino;

Tipo de Instruções:

- Formato do tipo R: Este formato aborda instruções baseadas em operações aritméticas.

Formato para escrita de código na linguagem NINA:

Tipo da Instrução	Reg1	Reg2	Reg3

Formato para escrita em código binário:

4 bits	4 bits	4 bits	4 bits
15-12	11-8	7-4	3-0
Opcode	RD	RT	RS

- Formato do tipo I: Este formatado aborda instruções baseadas em Load e Store.

I type

OP	**RD**	**RT**	**Adress**
15-12	**11-8**	**7-4**	**3-0**

- Formato do tipo J: Este formatado aborda instruções para pular (jump) para um endereço da memória.

J type

OP	RD	Endereço
15-12	11-9	8-0

Visão geral das instruções do Processador N.I.N.A:

O número de bits do campo **Opcode** das instruções é igual a quatro, sendo assim obtemos um total $(Bit(0\ e\ 1)^{número\ Total\ de\ Bits\ do\ Opcode} \therefore 2^2 = 4)$ de 15 **Opcodes (12-15)**que são distribuídos entre as instruções, assim como é apresentado na Tabela 1.

Opcode	Nome	Formato	Breve Descrição	exemplo
0000	ADD	R	Soma	ADD $S1, $S2, $S3, ou seja, $S1 = $S2 + $S3
0001	SUB	R	Subtração	SUB $S1, $S2, $S3, ou seja, $S1 = $S2 - $S3
0010	MULT	R	Multiplicação	MULT $S1, $S2, $S3, ou seja, $S1 = $S2 * $S3
0011	DIV	R	Divisão	DIV $S1, $S2, $S3, ou seja, $S1 = $S2 / $S3
0100	AND	R	e	AND $S1, $S2, $S3 ou seja, S1 = S2 & S3
0101	OR	R	ou	OR $S1, $S2, $S3 ou seja, S1 = S2 | S3
0110	XOR	R	ou exclusivo	XOR $S1, $S2, $S3, ou seja, S1 = ($\neg$S2*S3) + (S2*$\neg$S3)
0111	NAND	R	negação do e	NAND $S1, $S2, $S3, ou seja, S1 = $\neg$(S2 & S3)
1000	NOR	R	negação do ou	NOR $S1, $S2, $S3, S1 = $\neg$(S2 | S3)
1001	SW	I	Store	SW $S1, DESTINO, ou seja, dados do registrador S1 para a memória DESTINO
1010	LW	I	Load	LW $S1,DESTINO, ou seja, dados da memória DESTINO para o registrador S1
1011	BEQ	R	comparação de igualdade	BEQ $S1, $S3,DESTINO, ou seja, compara registradores e se forem igual vai para o endereço
1100	BNE	R	comparação de diferença	BNE $S1, $S3,DESTINO, ou seja, se os registradores forem diferentes vai para o endereço
1101	JAL	J	Jump	J 100, ou seja, vai para o endereço 100

Tabela 1 – Tabela que mostra a lista de Opcodes utilizadas pelo processador N.I.N.A.

1.3 Descrição do Hardware

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

1.3.1 ALU ou ULA

O componente NALU (N 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 NALU efetua operações de comparação de valor igual ou diferente O componente NALU recebe como entrada três valores: **A** – dado de 16 bits para operação; **B** - dado de 16 bits para operação e **OP** – identificador da operação que será realizada de 4 bits. O NALU também possui duas saídas: **result** – saída com o resultado das operações lógicas e aritméticas e uma saída de um bit para saber quando vai ser usado o beq e bne.

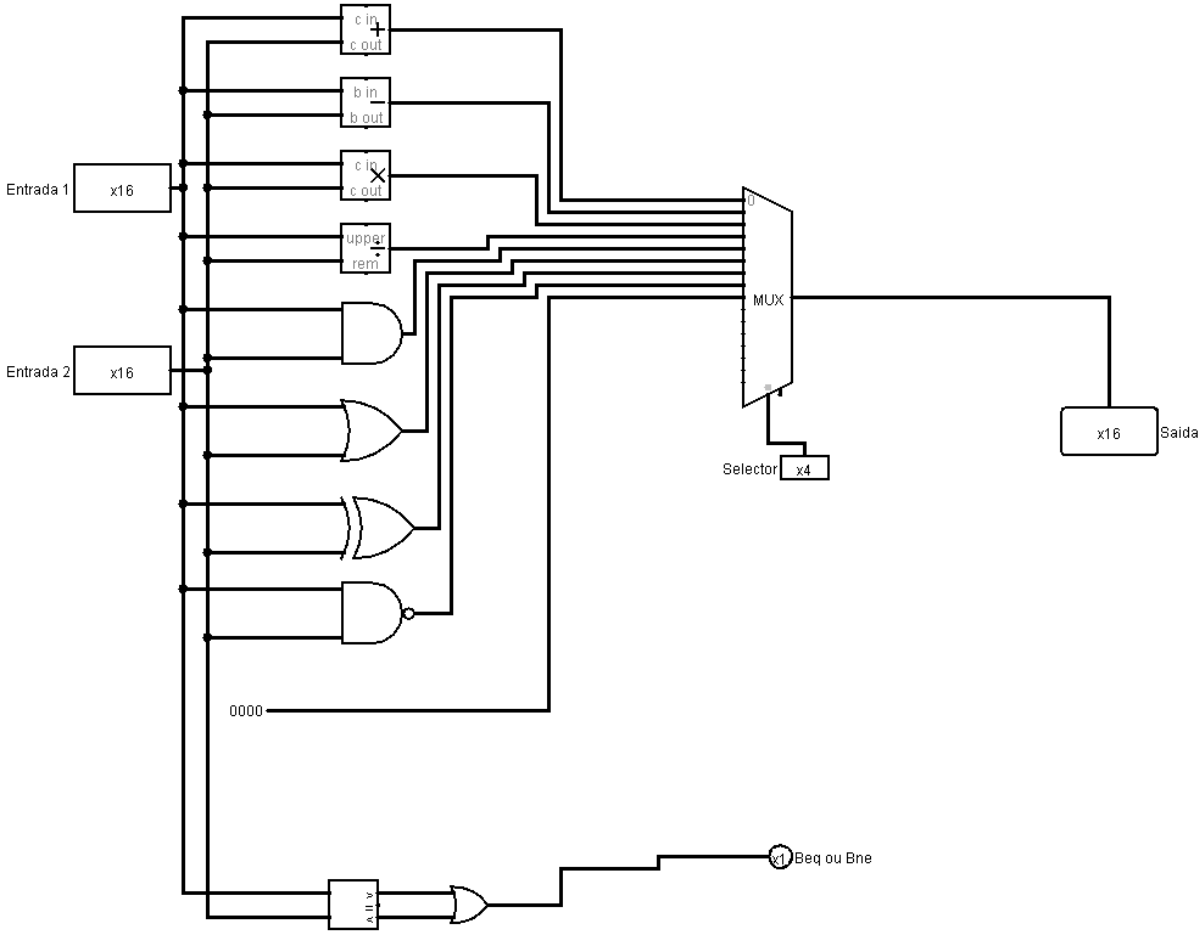

Figura 2 - Bloco simbólico do componente NALU gerado pelo LogSim

1.3.2 Banco de Registradores

Tem como finalidade guardar dados, da seguinte forma, possuindo quatro entradas sendo elas, endereço do registrador de leitura de 4 bits, segundo endereço do registrador de leitura também de 4 bits, o endereço do registrador de escrita de 4 bits e o dado para ser escrito de 16 bits. Também possui três valores de saída sendo eles, o dado guardado no registrador de leitura 1, o dado guardado no registrador de leitura 2 e por último o valor guardado no registrador de destino ambos de 16 bits. também possui dois controladores de dados de 1 bit sendo um para limpar os dados dos registradores e o outro para informar, quando um dado, está disponível, para ser armazenado nos registradores.

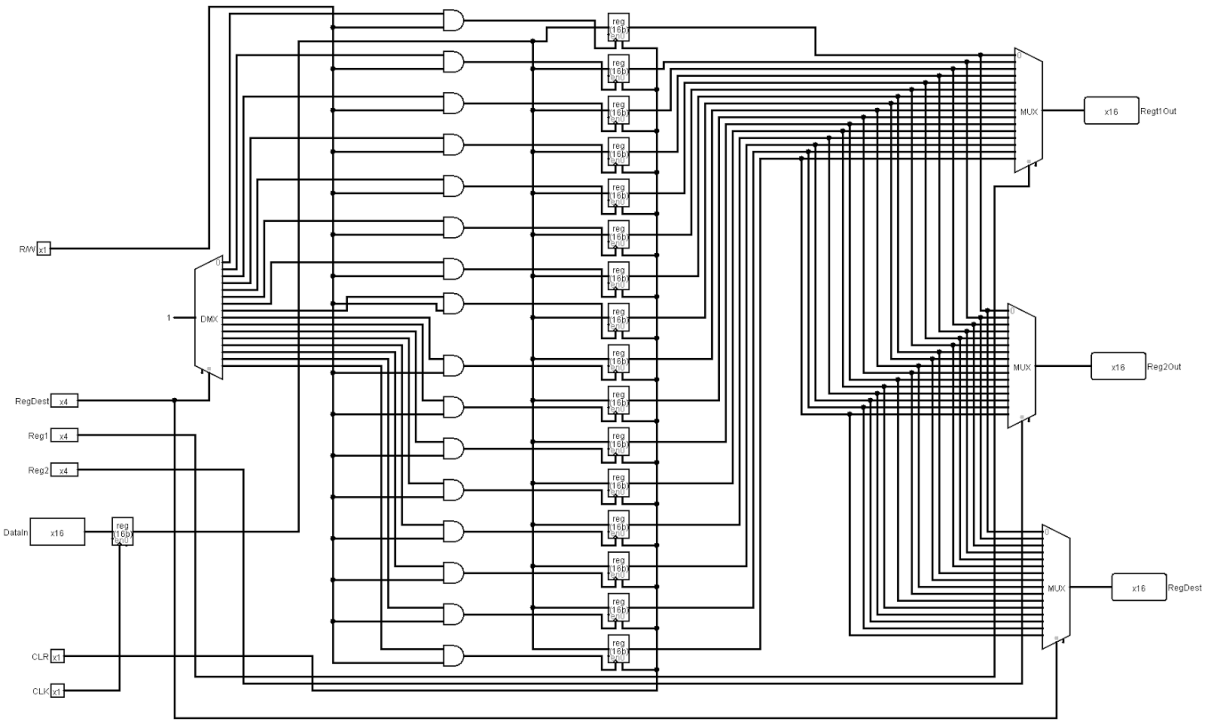

Figura 3 - Banco de Registradores gerada pelo LogSim

1.3.3 Controle

O componente Controle 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:

§ **Jump**: 1 bit.
§ **BEQ ou BNE**: 1 bit.
§ **ULA fonte ou MemParaReg**: 1 bit
§ **RegParaMem ou MemParaReg**: 1 bit.
§ **RegParaMem**:1 bit.
§ **ULAop:** 4 bit.

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

Comando	Jump	BEQ ou BNE	ULA fonte ou MemParaREg	RegParaMem ou MemParaReg	RegParaMem	UlaOp
add	0	0	0	0	0	0000
sub	0	0	0	0	0	0001
mult	0	0	0	0	0	0010
DIV	0	0	0	0	0	0011
AND	0	0	0	0	0	0100
OR	0	0	0	0	0	0101
XOR	0	0	0	0	0	0110
NAND	0	0	0	0	0	0111
NOR	0	0	0	0	0	1000
SW	0	0	0	1	1	1001
LW	0	0	1	1	0	1010
BEQ	0	1	0	0	0	1011
BNE	0	1	0	0	0	1100
JAL	1	0	0	0	0	1101

Tabela 2 - flags da unidade de controle

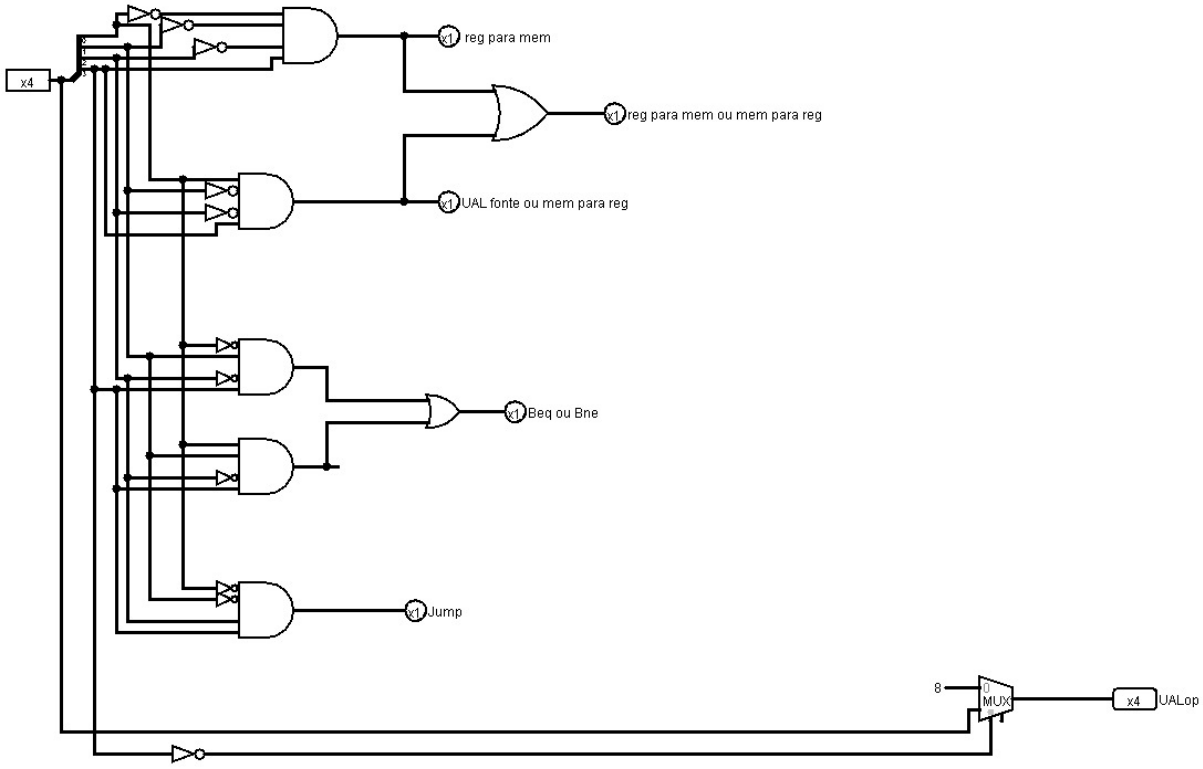

Figura 4 - Unidade de controle gerada pelo LogSim

1.3.4 Memória de dados

Trata-se da memória RAM de 16 bits ela armazena os dados temporários para que o processador possa acessá las de maneira mais rápida.

1.3.5 Memória de Instruções

Guarda as instruções que serão processadas possui como entrada um endereço de 16 bits, e a saída de 16 bits que é a instrução encontrada pelo endereço.

1.3.6 PC

é um registrador que controla a memória de instruções buscando a instrução a ser executada pelo processador

1.4 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.

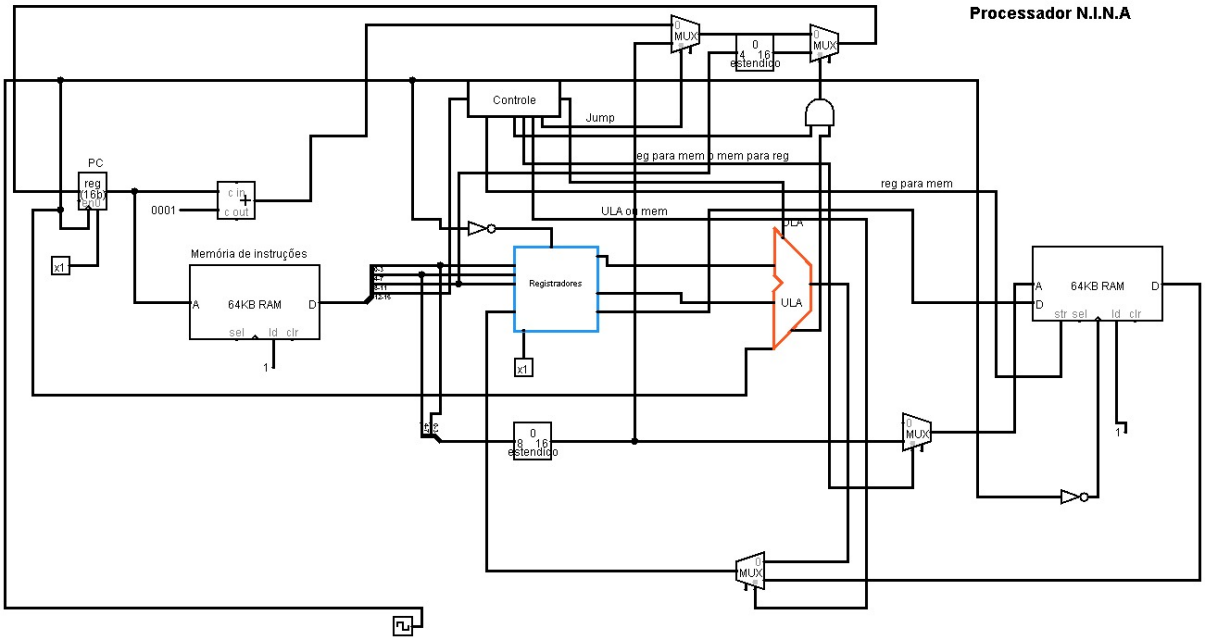

Figura 5 - Datapath gerado pelo LogSim

2 Simulação e Teste

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 N.I.N.A utilizaremos como exemplo o código para calcular o número da sequência de Fibonacci.

	OP	RD	R1	R2	ENDEREÇO	HEXADECIMAL
lw $s0,m(0)	1001	0000			00000000	9000
lw $s1,m(1)	1001	0001			00000001	9101
add $s2,$s3,$s4	0000	0011	0001	0111		0317
add $s4,$s3,$s10	0000	0111	0001	0100		0714
add $s3,s2,$s10	0000	0001	0011	0100		0134
sw $s2,m(2)	1000	0011			00000010	8302
lw $s7,m(1)	1001	1000			00000001	9801
add $s11,$s11,$s7	0000	1001	1001	1000		0998
bne $s11,$s0,Início	1011	0001	1001	0000		b199

Tabela 3 - Código Fibonacci para o processador N.I.N.A/EXEMPLO.

Descrição dos testes

As imagens seguintes mostra o resultado do programa no logisim

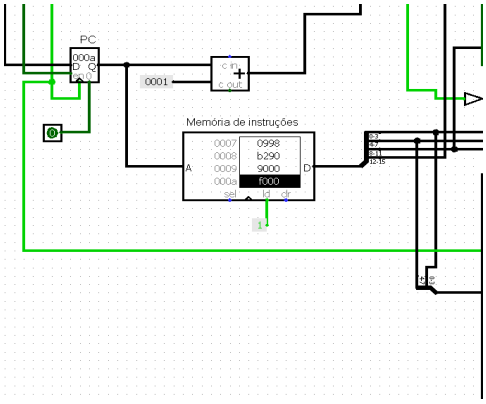

figura 6 - teste parte 1.

Nessa imagem mostra as instruções do programa, e a distribuição dos bits até o banco de registradores e a unidade de controle.

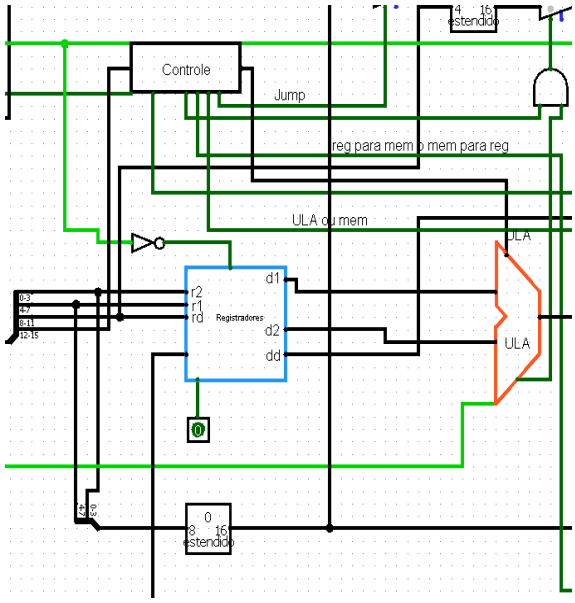

Figura 7- teste parte 2

Nessa imagem mostra os bits sendo distribuídos até o banco de registradores e a unidade de controle.

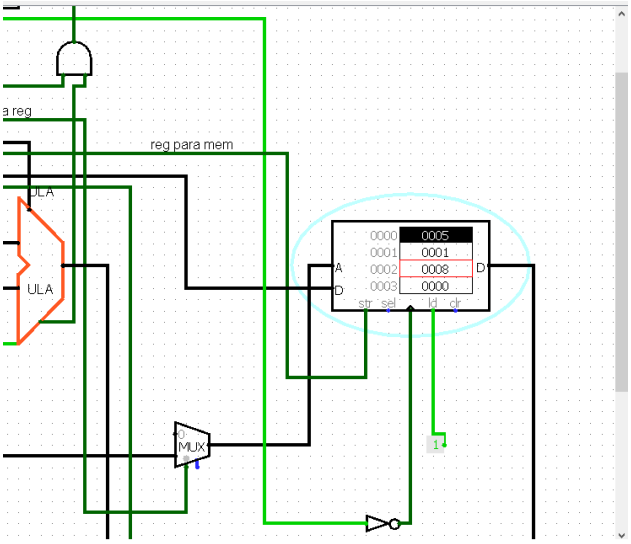

Figura 8 - teste parte 3

Nesse exemplo o programa faz um loop cinco vezes, resultando em um valor 8 que é guardado no endereço de memória 0000000000000010.

3 Considerações finais

Este trabalho apresentou o projeto e implementação do processador de 16 bits denominado de N.I.N.A com ele foi possível aprender de melhor forma como funciona a arquitetura de um processador, assim como entender o funcionamento de cada um dos seus componentes, a fim de que possamos escrever um programa onde utiliza o hardware de forma que possamos aproveitar cada um de seus componentes assim otimizando o nosso código. Pois quando se entende como o Hardware funciona pode-se escrever um software mais específico gerando assim um programa mais eficient

4 Referências

http://www.cburch.com/logisim/pt/
https://sites.google.com/site/aocufrr2016/
https://www.youtube.com/watch?v=QfH0QN9yPt8
Organização e Projeto de Computadores - David A. Patterson