Disciplina: CIC 116394 - Organização e Arquitetura de Computadores: Turma A
Professor: Marcus Vinicius Lamar

Laboratório 3
-CPU *femto*RISC-V UNICICLO-

André Dornelas Sanches - 17/0099369
Andressa Maria Monteiro Sena - 16/0112303
Eduardo de Souza Felix de Almeida - 17/0102254

Introdução

O software Deeds (Digital Electronics Education and Design Suite) é uma plataforma educativa que permite a simulação de circuitos lógicos. Por meio dos elementos fornecidos pelo software (portas lógicas, multiplexadores, circuitos aritméticos etc), será implementado um processador Uniciclo com suporte as seguintes operações da ISA RV32I: add, sub, and, or, slt, xor, lw, sw, addi, slli, lui, jal, beq e jalr. Existem outras opções de programas para este fim, como o Quartus Prime 18.1 da Intel ,por exemplo, que oferece suporte a linguagem de descrição de Hardware SystemVerilog, porém, optou-se pelo uso do Deeds por ser mais "leve" em relação ao concorrente.

O processador pode ser dividido em dois grupos: caminho de dados e unidade de controle. Sua implementação é uma associação de circuitos combinacionais (multiplexadores, ADD, ULA, portas lógicas) e circuitos sequenciais (PC, Registradores e Memória de Dados). Há mais dois componentes na estrutura do processador que irão variar de acordo com o tipo de arquitetura (Harvard) e de projeto(Uniciclo): Memória de Instruções e Bloco de Controle, combinacional e combinacional, respectivamente.

Objetivos

- Implementação de um processador uniciclo compatível com a ISA RISC-V 32I no programa de Simulação Deeds;
- Analisar o desempenho do processador montado;

Procedimentos e Análise de Dados

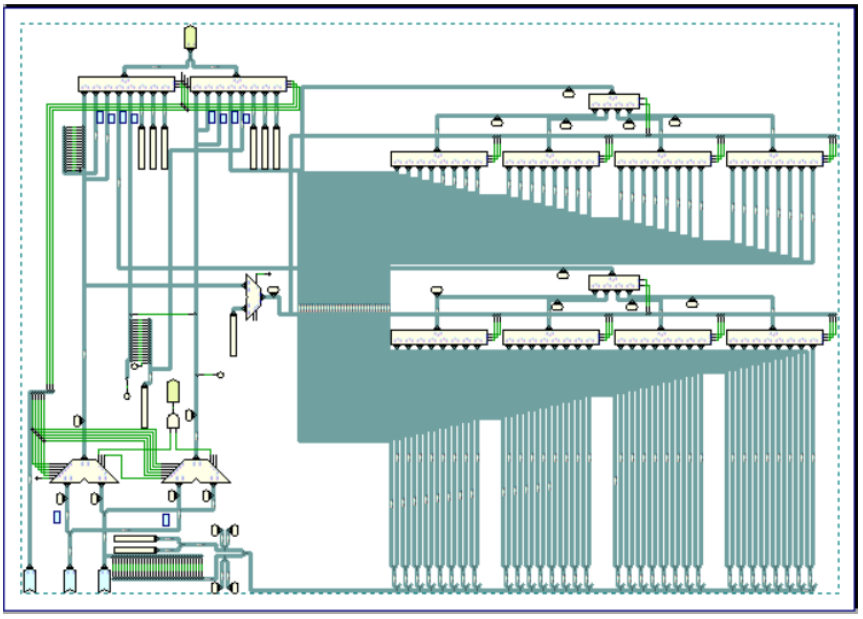
1.1)

Para a construção da unidade de controle da ULA (ALU Control) foram feitas as seguintes tabelas a partir da segunda foi construído o bloco ALUCtrl usando uma ROM.

Opcode	ALUOp	Operação	Funct7	Funct3	Operação da ULA	ALU Control [7:5]	ALU Control [4:0]
sw	000	sw	xxxxxxx	xxx	add	000	11001
lw	000	lw	xxxxxxx	xxx	add	000	11001
jalr	101	jalr	xxxxxxx	xxx	add	001	11001
beq	001	beq	xxxxxxx	xxx	sub	000	11100
lui	010	lui	xxxxxxx	xxx	F=B	010	00101
Tipo-I	011	addi	xxxxxxx	000	add	000	11001
Tipo-I	011	slli	xxxxxxx	001	F=B	011	00101
Tipo-R	100	add	0000000	000	add	000	11001
Tipo-R	100	sub	0100000	000	sub	000	11100
Tipo-R	100	slt	0000000	010	sub	100	11100
Tipo-R	100	xor	0000000	100	xor	000	10000
Tipo-R	100	or	0000000	110	or	000	01100
Tipo-R	100	and	0000000	111	and	000	01000

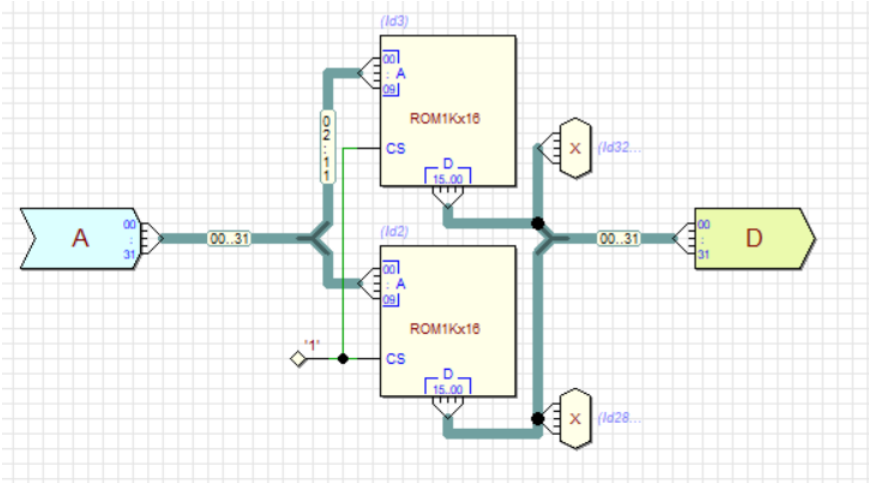
ALUOP	Funct[10]	Hexa	ALUCtrl	Hexa
000	xxxx	00 até F	00011001	19
000	xxxx	00	00011001	19
101	xxxx	50 até 5F	00111001	39
001	xxxx	10 até 1F	00011100	1C
010	xxxx	20 até 2F	01000101	45
011	x000	30 e 38	00011001	19
011	x001	31 e 39	01100101	65
100	0000	40	00011001	19
100	1000	48	00011100	1C
100	0010	42	10011100	9C
100	0100	44	00010000	10
100	0110	46	00001100	0C
100	0111	47	00001000	08

ULA:

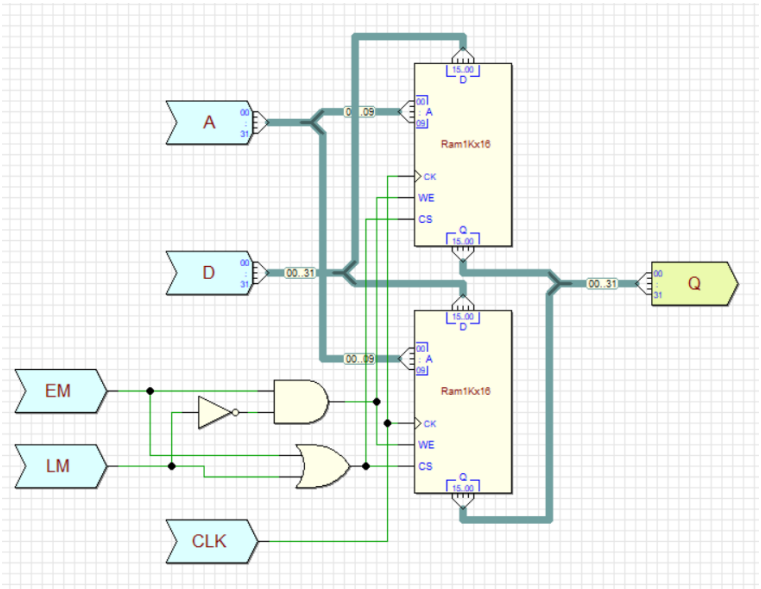


1.2)

Memória de Instruções:

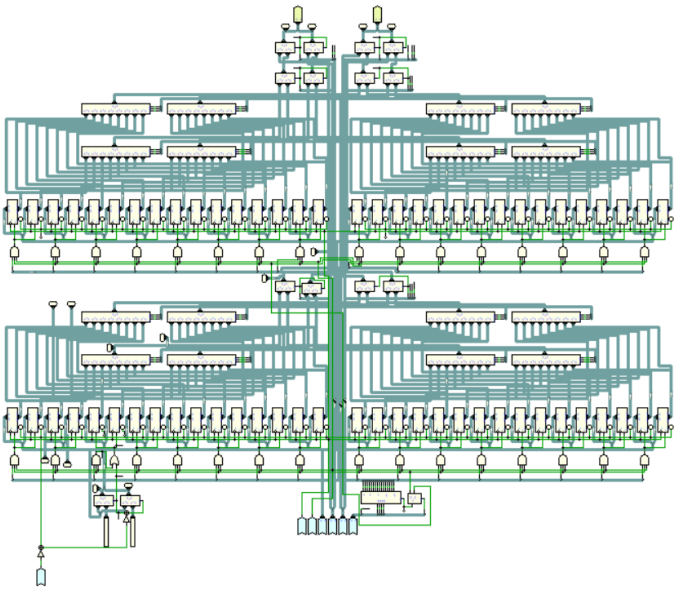


Memória de Dados:

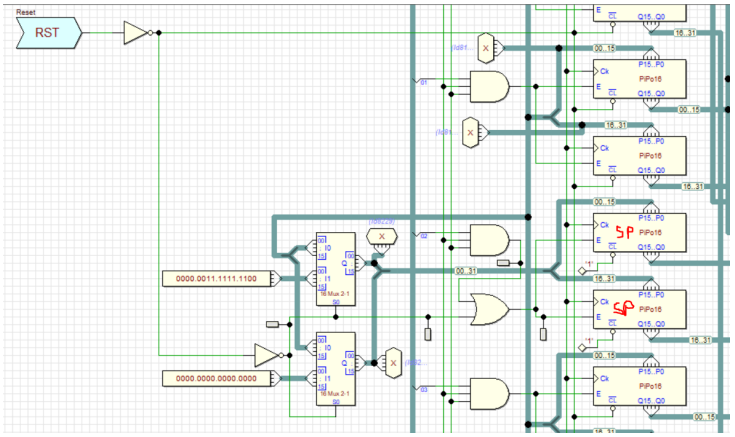


1.3)

Banco de Registradores:



Utilizou-se de dois MUX para setar o valor de sp em 0x000003FC:



1.4)

A partir do caminho de dados construído, apenas com a adição das funções jal e jalr, foram feitas as tabelas a seguir e a partir da segunda, foi implementada uma ROM no bloco Control.

Opcode	Instrução	ALUSrc	Mem2Reg	RegWrite	MemRead	MemWrite	Branch	CJ	CJR	ALUOp
sw	sw	1	X	0	0	1	0	0	0	000
lw	lw	1	1	1	1	0	0	0	0	000
jalr	jalr	1	X	1	0	0	0	1	1	000
beq	beq	0	X	0	0	0	1	0	0	001
jal	jal	X	X	1	0	0	0	1	0	XXX
lui	lui	1	0	1	0	0	0	0	0	010
Tipo-I	addi	1	0	1	0	0	0	0	0	011
Tipo-I	slli	1	0	1	0	0	0	0	0	011
Tipo-R	add	0	0	1	0	0	0	0	0	100
Tipo-R	sub	0	0	1	0	0	0	0	0	100
Tipo-R	and	0	0	1	0	0	0	0	0	100
Tipo-R	or	0	0	1	0	0	0	0	0	100
Tipo-R	slt	0	0	1	0	0	0	0	0	100
Tipo-R	xor	0	0	1	0	0	0	0	0	100

Opcode	Hexa	ALUSrc	Mem2Reg	RegWrite	MemRead	Mem Write	Branch	CJ	CJR	ALUOp	Hexa
sw	23	1	X	0	0	1	0	0	0	000	440
lw	03	1	1	1	1	0	0	0	0	000	780
jalr	67	1	X	1	0	0	0	1	1	000	518
beq	63	0	X	0	0	0	1	0	0	001	21
jal	6F	X	X	1	0	0	0	1	0	XXX	110
lui	37	1	0	1	0	0	0	0	0	010	502
Tipo-I	13	1	0	1	0	0	0	0	0	011	503
Tipo-I	13	1	0	1	0	0	0	0	0	011	503
Tipo-R	33	0	0	1	0	0	0	0	0	100	104
Tipo-R	33	0	0	1	0	0	0	0	0	100	104
Tipo-R	33	0	0	1	0	0	0	0	0	100	104
Tipo-R	33	0	0	1	0	0	0	0	0	100	104
Tipo-R	33	0	0	1	0	0	0	0	0	100	104
Tipo-R	33	0	0	1	0	0	0	0	0	100	104

1.5)

A máxima frequência de clock utilizável foi calculada a partir do tempo mínimo para executar a instrução mais longa, que no caso é a load word. Esse tempo de atraso é o menor período que o processador aguenta, portanto a sua máxima frequência é dada por:

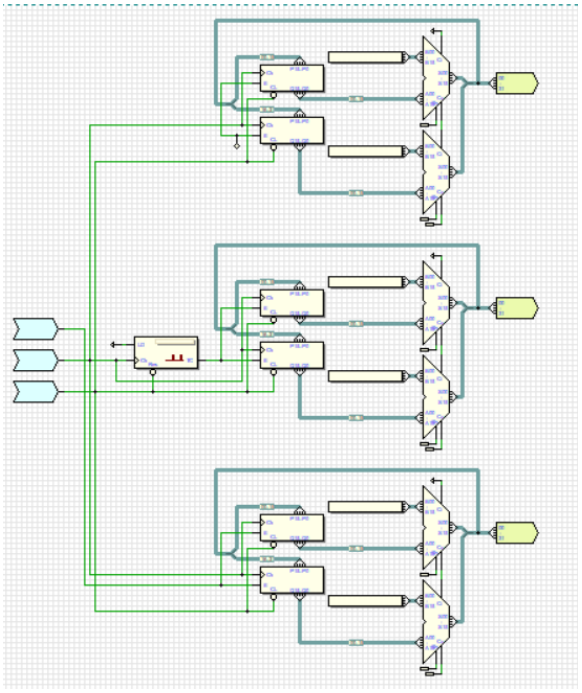
$$f = \frac{1}{menor\ T}\ [Hz]$$

Porém não foi possível calcular os valores de atraso pois o Deeds trava quando se tenta abrir o Timing Diagram Simulation.

Também tentou-se calcular empiricamente a partir de qual frequência o processador começava a dar erros, mas ao chegar na frequência 1200 Hz o programa também travou e precisou ser pausado.

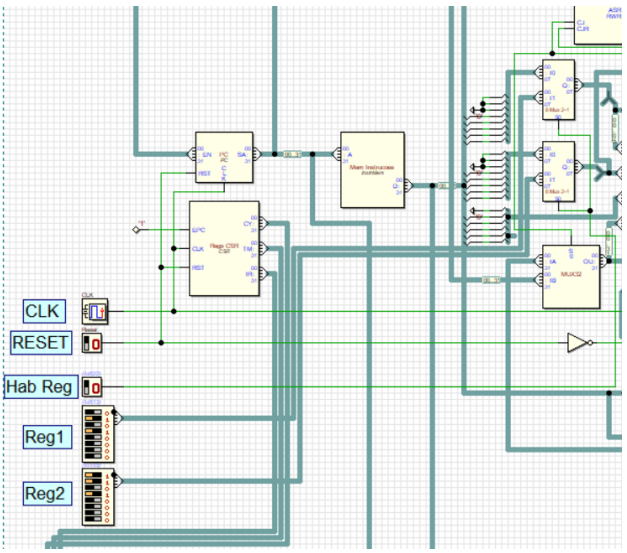
1.6)

Registradores CSR:



1.7)

Escolha dos registradores foi feita usando 2 MUX:



1.8)

Link do vídeo:
https://drive.google.com/file/d/1spX2J36WMoEzgntsy53ZP_qjneOb2DOM/view?usp=sharing

Os resultados esperados para cada operação podem ser vistos como comentários em cada linha de instrução no arquivo TestBench.s

1.9)

O tempo necessário para a execução do programa é dado pela expressão $t_{exec} = I \times CPI \times T$, onde I é o número de instruções total, CPI é a média da quantidade de clocks necessários à execução do programa, e T é o período do clock utilizado.

A quantidade de instruções pode ser encontrada ao final do programa no registrador instret implementado. A CPI do processador uniciclo é 1 e o menor período de clock possível é o inverso da maior frequência utilizável no processador.

Como não foi possível implementar uma forma de parar o processador ao final do programa e a memória de instruções por padrão reinicia o programa sempre que o mesmo chega ao fim, não pudemos verificar o número de instruções executadas ao fim do programa.

O funcionamento do arquivo *test.s* no processador pode ser visto neste link: https://drive.google.com/file/d/1KWiGbthOOzefqUXq7R7KRCeqrXsoaV8S/view?usp=sharing