UFRR

Universidade Federal de Roraima
Centro de Ciência e Tecnologia
Departamento de Ciência da Computação

1

30 de novembro de 2023
Boa Vista - RR

Departamento de
CIÊNCIA DA COMPUTAÇÃO

Projeto final - AOC

Processador 16-bits

TRINITY

Alunos: Rafael da Silva, Vicente Sampaio, William Faray
Disciplina: Arquitetura e Organização de Computadores
Professor: Herbert Oliveira Rocha

UFRR

Resumo

- Instruções do processador
- Formato de instruções
- Datapath
- Testes de instruções
- Waveforms resultantes

Departamento de CIÊNCIA DA COMPUTAÇÃO

UFRR

Tabela de Instruções

Nome	Formato	Instrução	Opcode
SOMA	Tipo R	Soma de dois operandos de 16 bits	0000
SOMA IMEDIATA	Tipo R	Soma de um operando de 16 bits e um valor imediato	0001
SUBTRAÇÃO	Tipo R	Subtração de dois operandos de 16 bits	0010
SUBTRAÇÃO IMEDIATA	Tipo R	Subtração de um operando de 16 bits e um valor imediato	0011
LW	Tipo I	Carrega um valor de 16 bits da memória	0100
SW	Tipo I	Armazena um valor de 16 bits na memória	0101
LI	Tipo I	Carrega um valor imediato de 16 bits	0110
BEQ	Tipo J	Desvio condicional se os dois operandos forem iguais	0111
IF	Tipo J	Define a flag de condição se os dois operandos forem iguais	1000
J	Tipo J	Desvio incondicional para o endereço especificado	1001

Departamento de CIÊNCIA DA COMPUTAÇÃO

UFRR

Tabela de Formatos de Instruções

INSTRUÇÕES DO TIPO R

Opcode	Reg 1	Reg 2	Reg 3
4 bits	4 bits	4 bits	4 bits
15 - 12	11 - 8	7 - 4	3 - 0

INSTRUÇÕES DO TIPO I

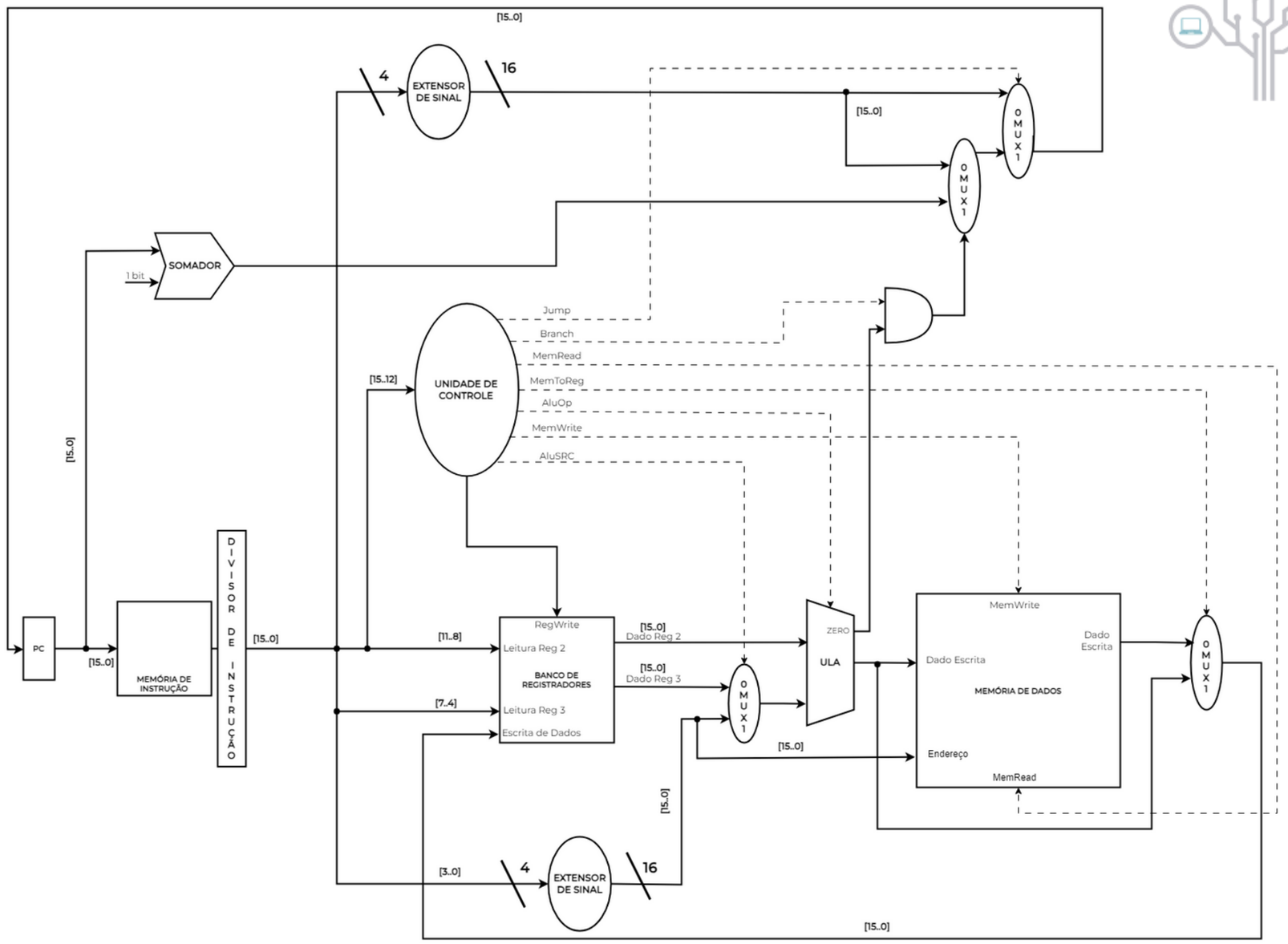
Opcode	reg 1	Reg 2	Reg 3
4 bits	4 bits	4 bits	4 bits
15 - 12	11 - 8	8 - 4	3 - 0

INSTRUÇÕES DO TIPO J

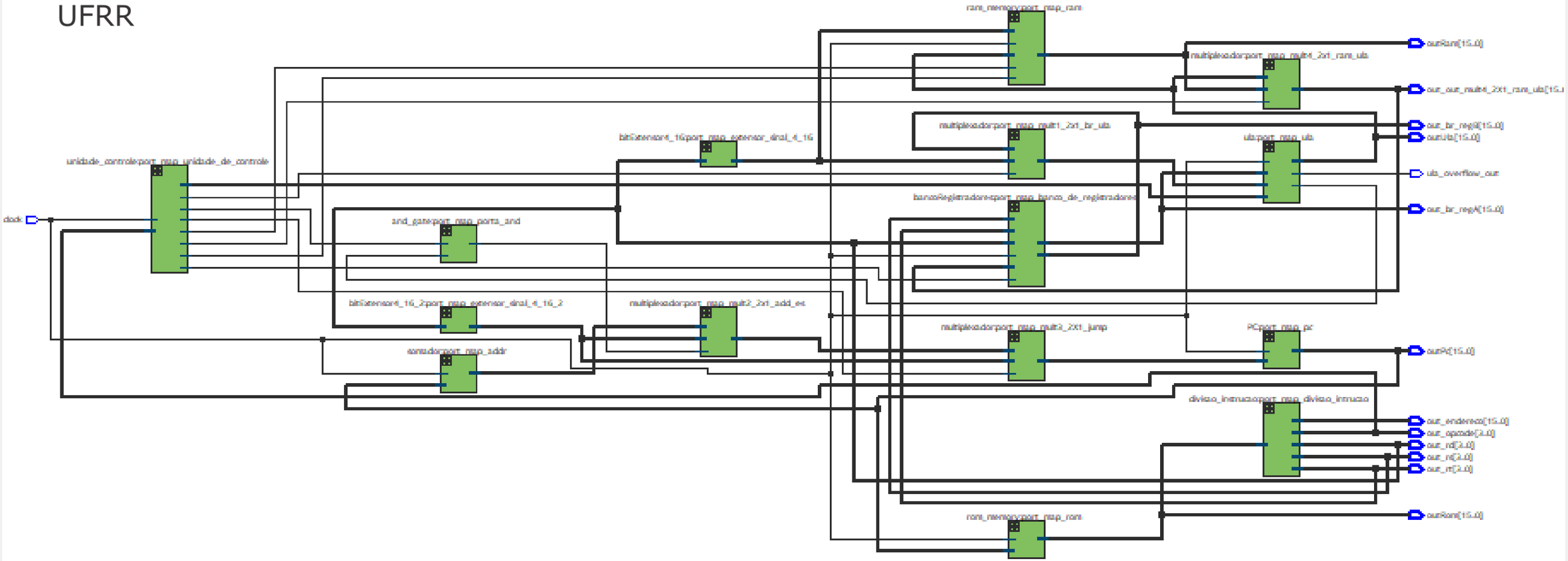
Opcode	Endereço
4 bits	12 bits
15 – 12	11 - 0

Departamento de CIÊNCIA DA COMPUTAÇÃO

DATAPATH


[15..0]
4
EXTENSOR DE SINAL
16
[15..0]
0 MUX 1
0 MUX 1
1 bit
SOMADOR
Jump
Branch
MemRead
MemToReg
AluOp
MemWrite
AluSRC
[15..12]
UNIDADE DE CONTROLE
[15..0]
PC
[15..0]
MEMÓRIA DE INSTRUÇÃO
DIVISOR DE INSTRUÇÃO
[15..0]
[11..8]
[7..4]
RegWrite
Leitura Reg 2
BANCO DE REGISTRADORES
Leitura Reg 3
Escrita de Dados
[15..0]
Dado Reg 2
[15..0]
Dado Reg 3
0 MUX 1
ZERO
ULA
MemWrite
Dado Escrita
MEMÓRIA DE DADOS
Endereço
MemRead
Dado Escrita
0 MUX 1
[15..0]
[15..0]
[3..0]
4
EXTENSOR DE SINAL
16
[15..0]

UFRR



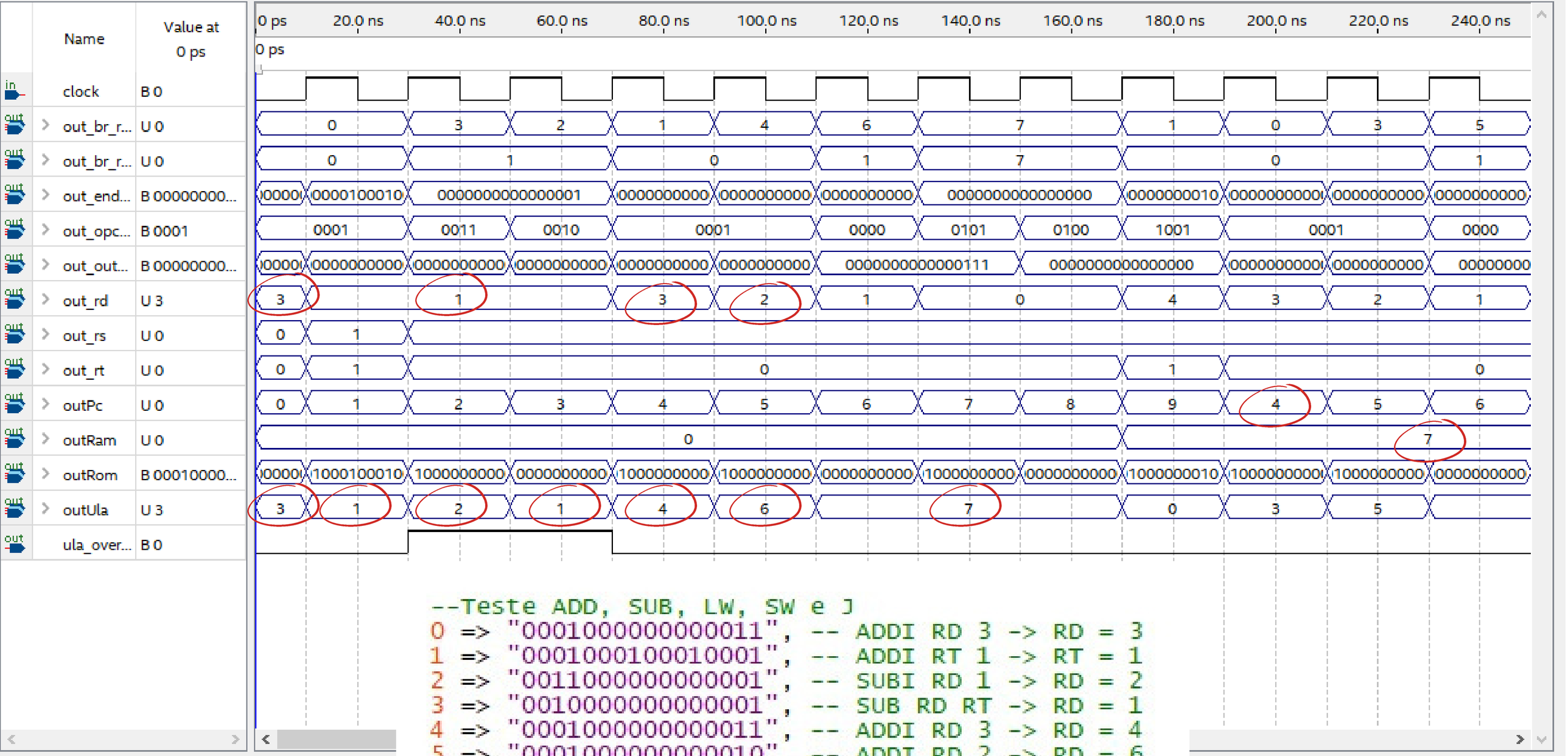
Departamento de
CIÊNCIA DA COMPUTAÇÃO

UFRR

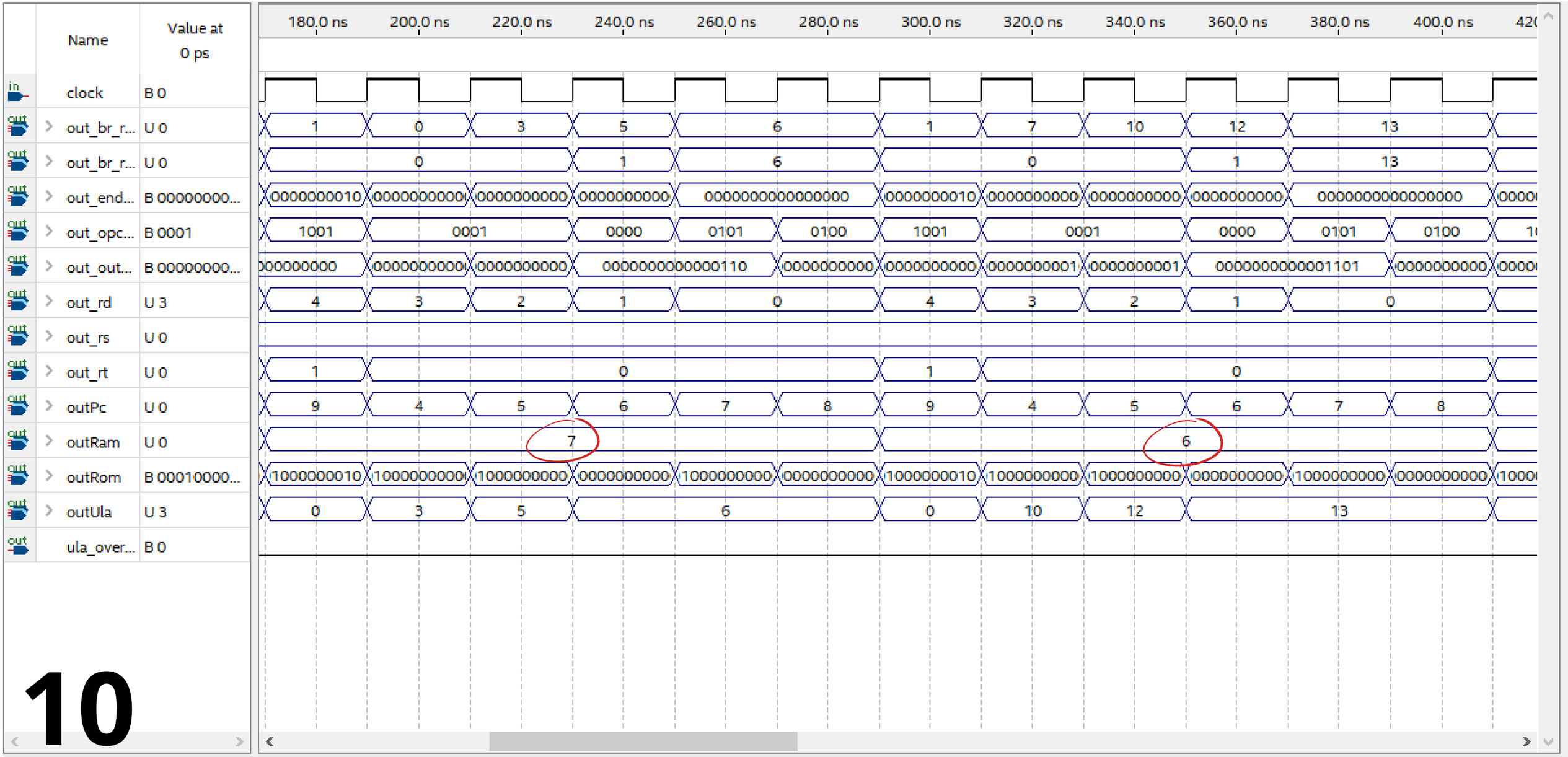
Teste primeiro algoritmo

Teste ADD, SUB, LW, e J					
Endereço Linguagem de alto nível		Binário			
		Opcode	Reg 3	Reg 2	Reg1
			Endereço		
			Dados		
0	ADDI S0 3 -> S0 = 3	0001	0000	0000	0011
1	ADDI S1 1 - > S1 = 1	0001	0001	0001	0001
2	SUBI S0 1 -> S0 = 2	0011	0000	0000	0001
3	SUB S0 S1 - > S0 = 1	0010	0000	0000	0011
4	ADDI S0 3 -> S0 = 4	0001	0000	0000	0010
5	ADDI S0 2 -> S0 = 6	0001	0000	0000	0010
6	ADD S0 S1 - > S0 = 7	0000	0000	0000	0001
7	SW S0	0101	0000	0000	0000
8	LW S0	0100	0000	0000	0000
9	J 0100	1001	0000	0001	0100

Departamento de CIÊNCIA DA COMPUTAÇÃO




```
--Teste ADD, SUB, LW, SW e J
0 => "0001000000000011", -- ADDI RD 3 -> RD = 3
1 => "0001000100010001", -- ADDI RT 1 -> RT = 1
2 => "0011000000000001", -- SUBI RD 1 -> RD = 2
3 => "0010000000000001", -- SUB RD RT -> RD = 1
4 => "0001000000000011", -- ADDI RD 3 -> RD = 4
5 => "0001000000000010", -- ADDI RD 2 -> RD = 6
6 => "0000000000000001", -- ADD RD RT -> RD = 7
7 => "0101000000000000", -- SW RD
8 => "0100000000000000", -- LW RD
9 => "1000000000010100", -- J 0100
```


Name
Value at
0 ps
clock
B 0
out_br_r...
U 0
out_br_r...
U 0
out_end...
B 00000000...
out_opc...
B 0001
out_out...
B 00000000...
out_rd
U 3
out_rs
U 0
out_rt
U 0
outPc
U 0
outRam
U 0
outRom
B 00010000...
outUla
U 3
ula_over...
B 0
180.0 ns
200.0 ns
220.0 ns
240.0 ns
260.0 ns
280.0 ns
300.0 ns
320.0 ns
340.0 ns
360.0 ns
380.0 ns
400.0 ns

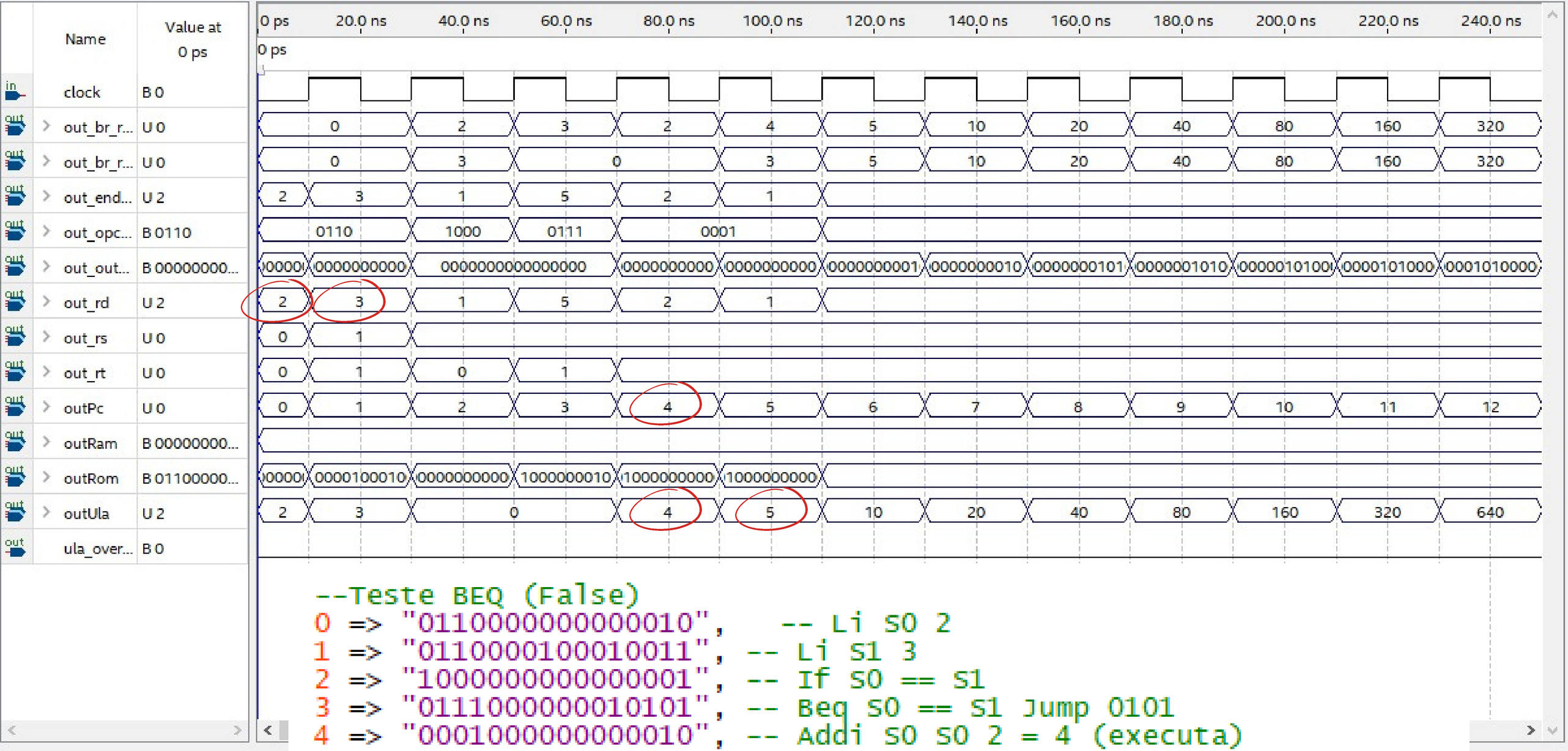
10

UFRR

Teste Beq (false)

Teste BEQ (False)					
Endereço Linguagem de alto nível		Binário			
		Opcode	Reg 3	Reg 2	Reg1
			Endereço		
		Dados			
0	Li S0 2	0110	0000	0000	0010
1	Li S1 3	0110	0001	0001	0011
2	If S0 == S1	1000	0000	0000	0001
3	Beq S0 == S1 jump 0101	0111	0000	0001	0101
4	Addi S0 S0 2 = 4 (EXECUTA)	0001	0000	0000	0010
5	Addi S0 SO 1 = 5 (resultado final)	0001	0000	0000	0001

Departamento de CIÊNCIA DA COMPUTAÇÃO



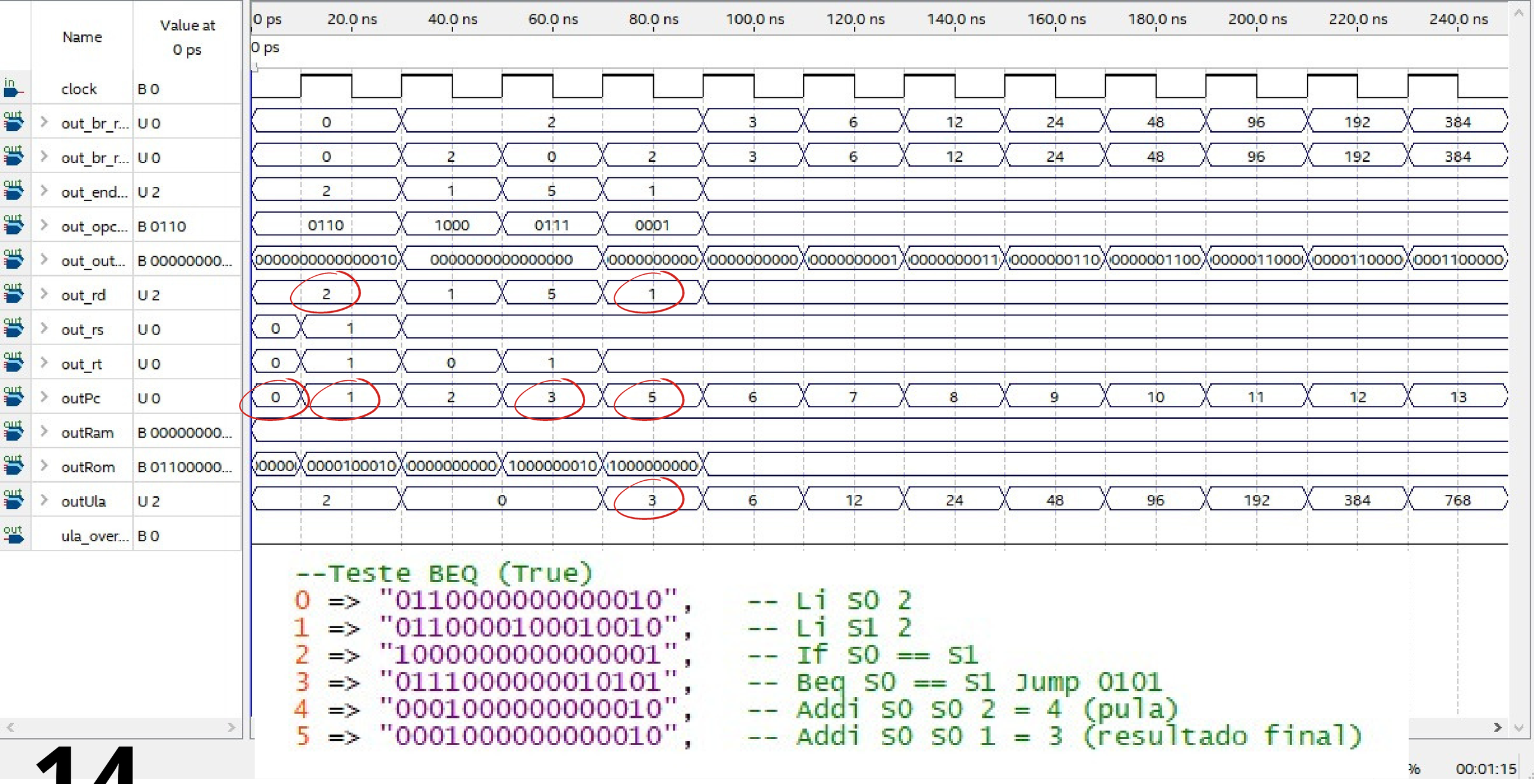

```
--Teste BEQ (False)
0 => "0110000000000010",    -- Li S0 2
1 => "0110000100010011", -- Li S1 3
2 => "1000000000000001", -- If S0 == S1
3 => "0111000000010101", -- Beq S0 == S1 Jump 0101
4 => "0001000000000010", -- Addi S0 S0 2 = 4 (executa)
5 => "0001000000000001", -- Addi S0 S0 1 = 5 (resultado final)
```

UFRR

Teste Beq (true)

Teste BEQ (True)					
Endereço Linguagem de alto nível		Binário			
		Opcode	Reg 3	Reg 2	Reg1
			Endereço		
		Dados			
0	Li S0 2	0110	0000	0000	0010
1	Li S1 2	0110	0001	0001	0010
2	If S0 == S1	1000	0000	0000	0001
3	Beq S0 == S1 jump 0101	0111	0000	0001	0101
4	Addi S0 S0 2 = 4 (pula)	0001	0000	0000	0010
5	Addi S0 SO 1 = 3 (resultado final)	0001	0000	0000	0001

Departamento de CIÊNCIA DA COMPUTAÇÃO


Name
Value at 0 ps
clock B 0
out_br_r... U 0
out_br_r... U 0
out_end... U 2
out_opc... B 0110
out_out... B 00000000...
out_rd U 2
out_rs U 0
out_rt U 0
outPc U 0
outRam B 00000000...
outRom B 01100000...
outUla U 2
ula_over... B 0
0 ps
20.0 ns
40.0 ns
60.0 ns
80.0 ns
100.0 ns
120.0 ns
140.0 ns
160.0 ns
180.0 ns
200.0 ns
220.0 ns
240.0 ns
--Teste BEQ (True)
0 => "0110000000000010", -- Li S0 2
1 => "0110000100010010", -- Li S1 2
2 => "1000000000000001", -- If S0 == S1
3 => "0111000000010101", -- Beq S0 == S1 Jump 0101
4 => "0001000000000010", -- Addi S0 S0 2 = 4 (pula)
5 => "0001000000000010", -- Addi S0 S0 1 = 3 (resultado final)
00:01:15

14

Referências

https://github.com/Lucas-Ladislau/DCC301_IanSantos_LucasAnderson_UFRR_2022/tree/main

https://www.fpga4student.com/2017/09/vhdl-code-for-mips-processor.html

https://www.fpga4student.com/2017/01/verilog-code-for-single-cycle-MIPS-processor.html