UFRR

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

Alunos: João Paulo Ramos, Karen Giovanna Furtado Melo.
Disciplina: Arquitetura e Organização de Computadores
Professor: Herbert Oliveira Rocha
Semestre: ERE 2020.2

18 de maio de 2021
Boa Vista - RR

UFRR

Processador Split

Detalhes do Processador Split

Arquitetura de 8 bits, feito com base no processador MIPS.

Todas suas operações são realizadas utilizando números inteiros.

Processador Split
Formato das Instruções

Operações do Tipo R		
Opcode	Reg1	Reg2
4 bits	2 bits	2 bits
7 - 4	3 - 2	1 - 0
Operações do Tipo I		
Opcode	Reg1	Reg2
4 bits	2 bits	2 bits
7 - 4	3 - 2	1 - 0
Operações do Tipo J		
Opcode	Endereço	
4 bits	4 bits	
7 - 4	3 - 0	

UFRR

Processador Split
Conjunto de instruções

Departamento de CIÊNCIA DA COMPUTAÇÃO

Opcode	Nome	Formato	Breve Descrição	Exemplo
0000	add	R	Soma	add $s0 $s1
0001	sub	R	Subtração	sub $s0 $s1
0010	mult	R	Multiplicação	mult $s0 $s1
0011	lw	I	Load Word	lw $s0 (memória00)
0100	sw	I	Store Word	sw $s0 (memória00)
0101	li	I	Load Immediate	li $s0 1
0110	beq	J	Desvio Condicional	beq endereço
0111	bne	J	Desvio Condicional	bne endereço
1000	J	J	Desvio Incondicional	J endereço(0000)
1001	If	R	Comparação para salto	if $s0 $s1 if $s0 == $s1

Processador Split
RTL Viewer do Processador

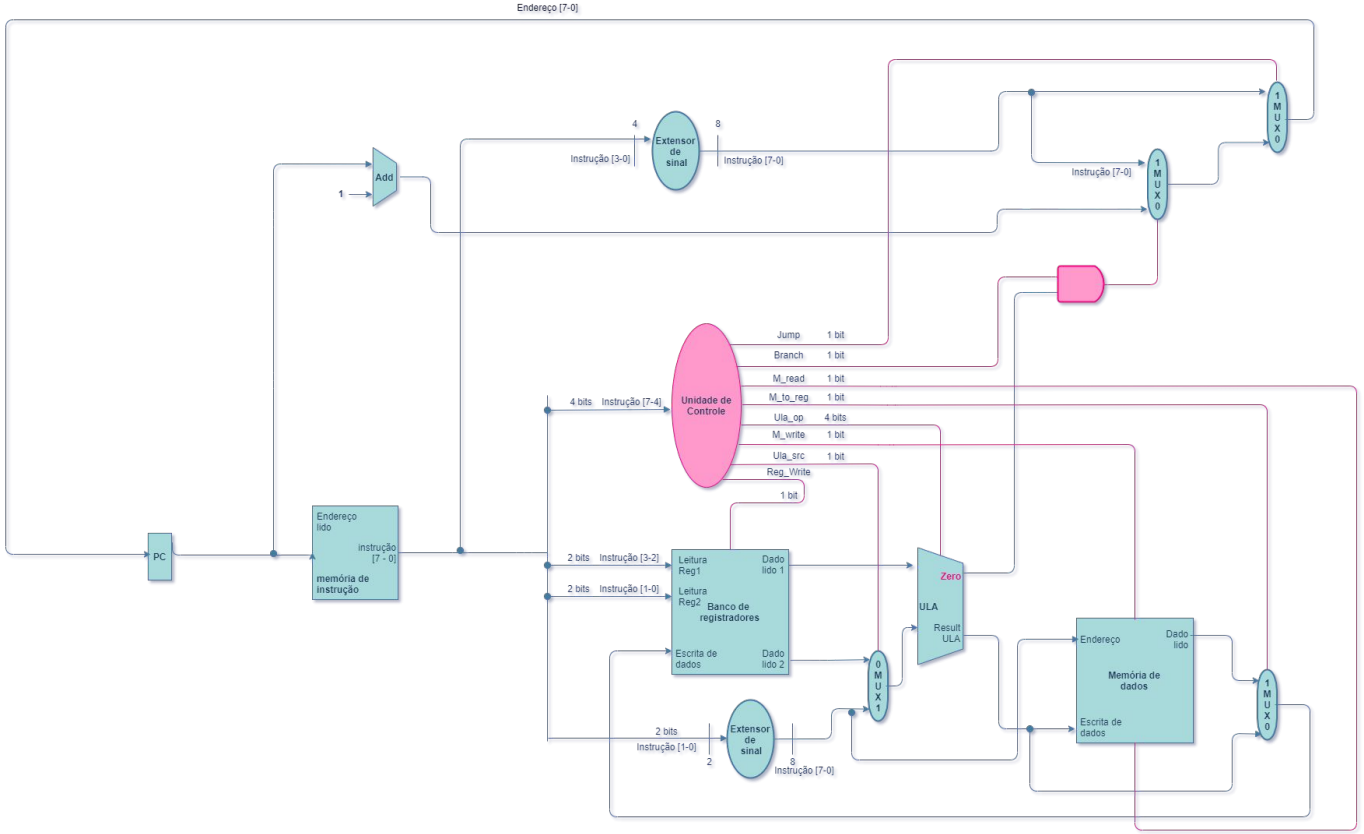
UFRR

Departamento de CIÊNCIA DA COMPUTAÇÃO

UFRR

Processador Split
Datapath do Processador

Departamento de CIÊNCIA DA COMPUTAÇÃO



UFRR

Processador Split Multiplicação

Departamento de CIÊNCIA DA COMPUTAÇÃO

```
21  architecture comportamento_ula of ula is
22
23      signal entrada_temp_if : std_logic; -- Variável temporária para o if
24      signal result_mult : std_logic_vector(15 downto 0);
25
26      begin
27          process(clock_ula)
28                  begin
29              case entrada_ula_op is
30                  when "0000" => -- add
31                      resultado_ula <= entrada_dado_lido1 + entrada_dado_lido2;
32
33                  when "0001" => -- sub
34                      resultado_ula <= entrada_dado_lido1 - entrada_dado_lido2;
35
36                  when "0010" => -- mult
37                      result_mult <= entrada_dado_lido1 * entrada_dado_lido2;
38                      if result_mult(8) = '1' or result_mult(9) = '1' or result_mult(10) = '1' or result_mult(11) = '1' or result_mult(12) =
39                      else
40                          resultado_ula <= result_mult(7 downto 0);
41                      end if;
42
43                  when "0011" => -- Load Word
44                      resultado_ula <= entrada_dado_lido1;
```

Vetor de 16 bits para armazenar o resultado da multiplicação de forma temporária

Processador Split Multiplicação


```
when "0010" => -- mult
    result_mult <= entrada_dado_lido1 * entrada_dado_lido2;
    if result_mult(8) = '1' or result_mult(9) = '1' or result_mult(10) = '1' or result_mult(11) = '1'
    or result_mult(12) = '1' or result_mult(13) = '1' or result_mult(14) = '1' or result_mult(15) = '1' then
    else
        resultado_ula <= result_mult(7 downto 0);
    end if;
```

Como não é possível representar um número maior que 256 com apenas 8 bits, então nós fizemos um tratamento para verificar se houve um overflow na multiplicação.

Se ocorrer, o resultado é ignorado, caso contrário é passado para a saída da ULA apenas os 8 bits mais significantes do vetor.

UFRR

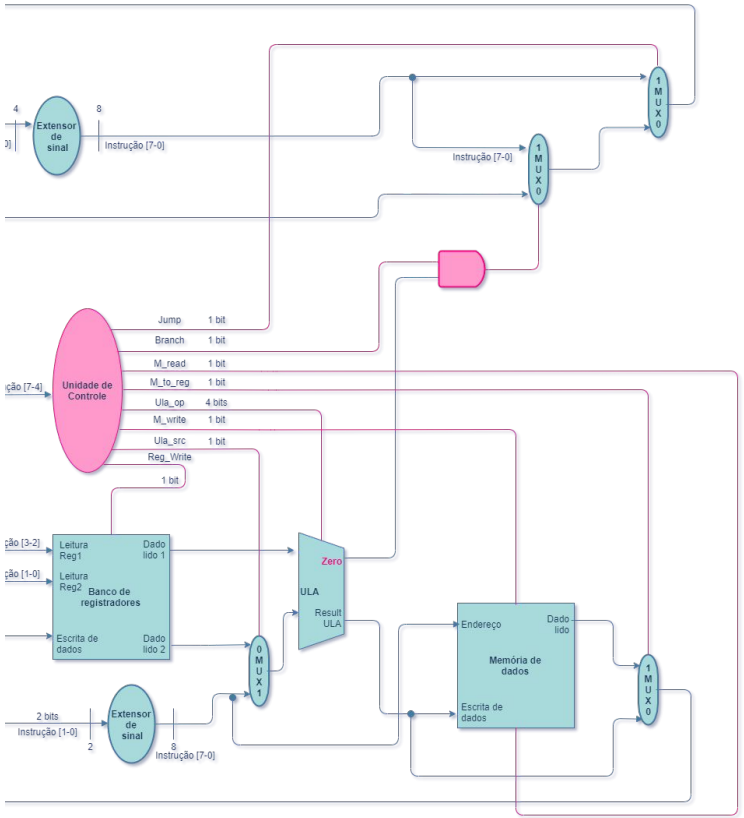
Processador Split
if beq e bne

Departamento de CIÊNCIA DA COMPUTAÇÃO

```
  when "1001" => -- if beq e bne
    if entrada_dado_lido1 = entrada_dado_lido2 then
        entrada_temp_if <= '1';
    else
        entrada_temp_if <= '0';
    end if;

when "0110" => -- breach if equal
    if entrada_temp_if = '1' then
        zero <= '1';
    else
        zero <= '0';
    end if;

when "0111" => -- breach if not equal
    if entrada_temp_if = '0' then
        zero <= '1';
    else
        zero <= '0';
    end if;
```

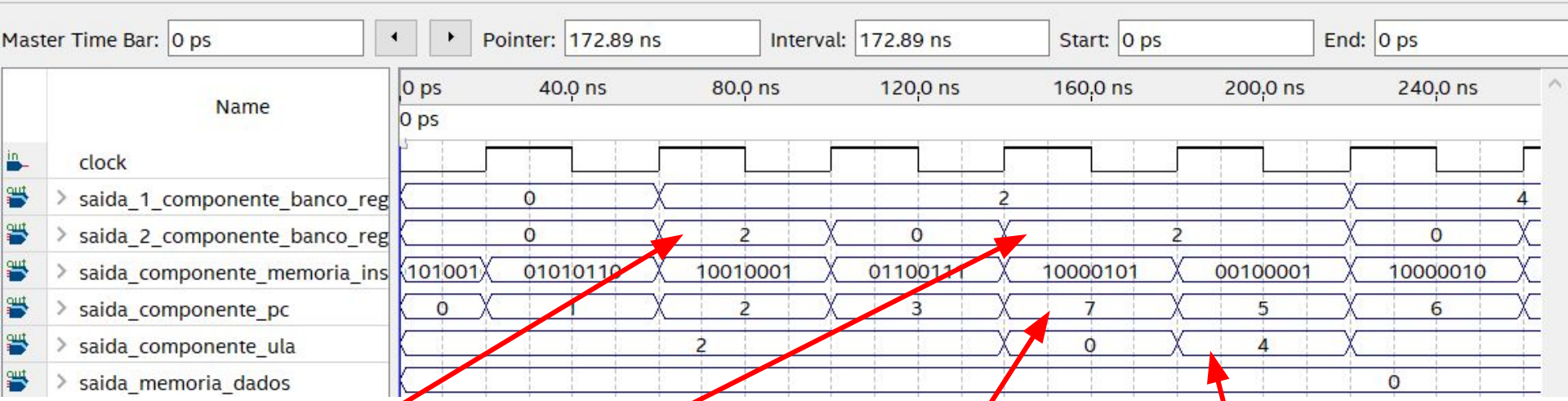


UFRR

Processador Split
Código 1 para testar processador

Departamento de CIÊNCIA DA COMPUTAÇÃO

Endereço	Linguagem de Alto Nível	Binário		
		Opcode	Reg2	Reg2
			Endereço	
			Dado	
0	li $s0 2	0101	00	10
1	li $s1 2	0101	01	10
2	if $s0 == $s1	1001	00	01
3	beq $s0 == $s1 --jump linha 7	0110	01	11
4	bne $s0 != $s1 --jump linha 8	0111	10	00
5	mult $s0 $s1 --s0 vira 4	0010	00	01
6	jump --linha 2	1000	00	10
7	jump linha 5	1000	01	01
8	sw $s0 00	0100	00	00
9	lw $s0 00	0011	00	00
10	li $s0 0	0101	00	00
11	li $s1 0	0101	01	00
12	li $s2 0	0101	10	00
13	li $s3 0	0101	11	00



O valor 2 é carregado no registrador $s0

li $s0 2

O valor 2 é carregado no registrador $s1

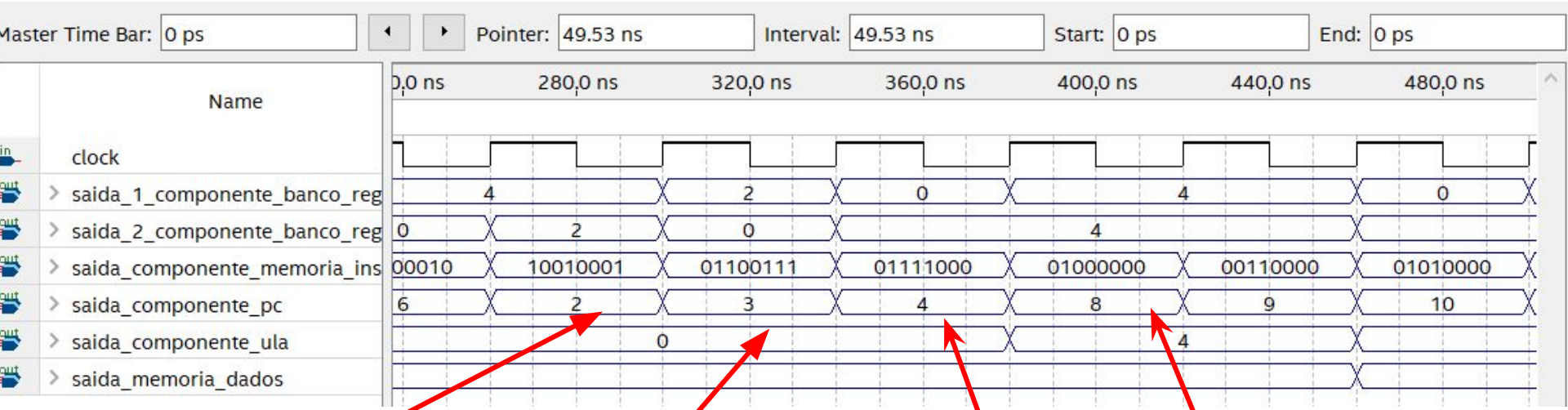
li $s1 2

O valor do registrador $s0 é igual ao registrador $s1, então realiza um salto para índice 7

beq $s0 == $s1

Multiplicação do registrador $s0 (2) e $s1 (2)

mult $s0 $s1



Pula para o índice 2

jump --linha 2

O valor do registrador $s0 não é igual ao registrador $s1, então prossegue o índice

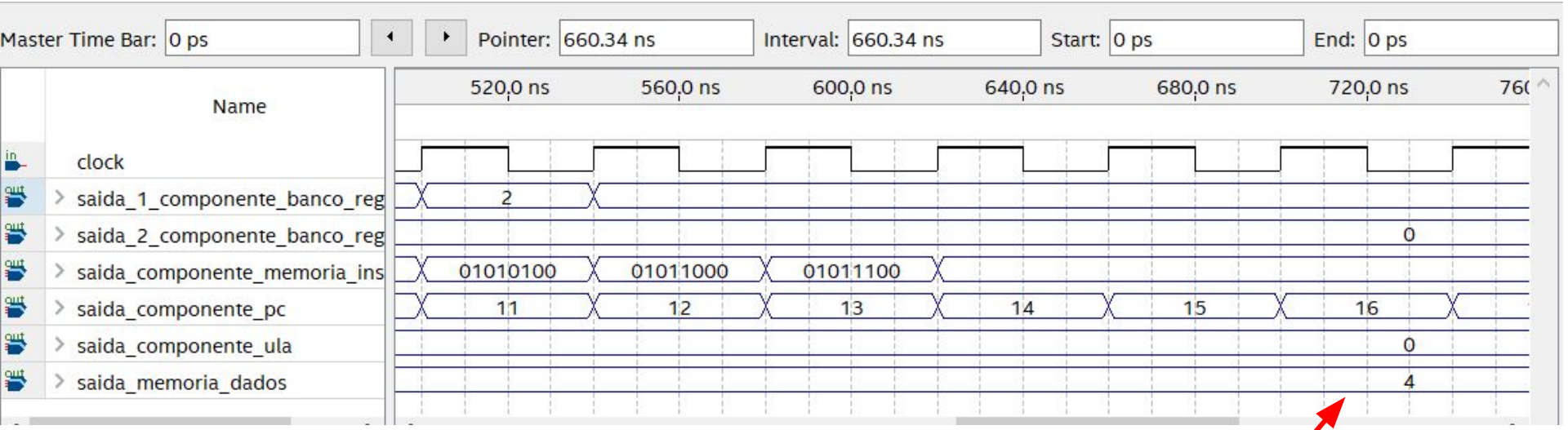
beq $s0 == $s1

O valor do registrador $s0 e $s1 são diferentes, portanto salta para índice 8

bne $s0 != $s1 --jump linha 8

Instrução de índice 8: Armazena o valor do registrador no endereço fornecido

sw $s0 00


Master Time Bar: 0 ps
Pointer: 660.34 ns
Interval: 660.34 ns
Start: 0 ps
End: 0 ps
Name
clock
saida_1_componente_banco_reg
saida_2_componente_banco_reg
saida_componente_memoria_ins
saida_componente_pc
saida_componente_ula
saida_memoria_dados
520,0 ns
560,0 ns
600,0 ns
640,0 ns
680,0 ns
720,0 ns
2
0
01010100
01011000
01011100
11
12
13
14
15
16
0
4

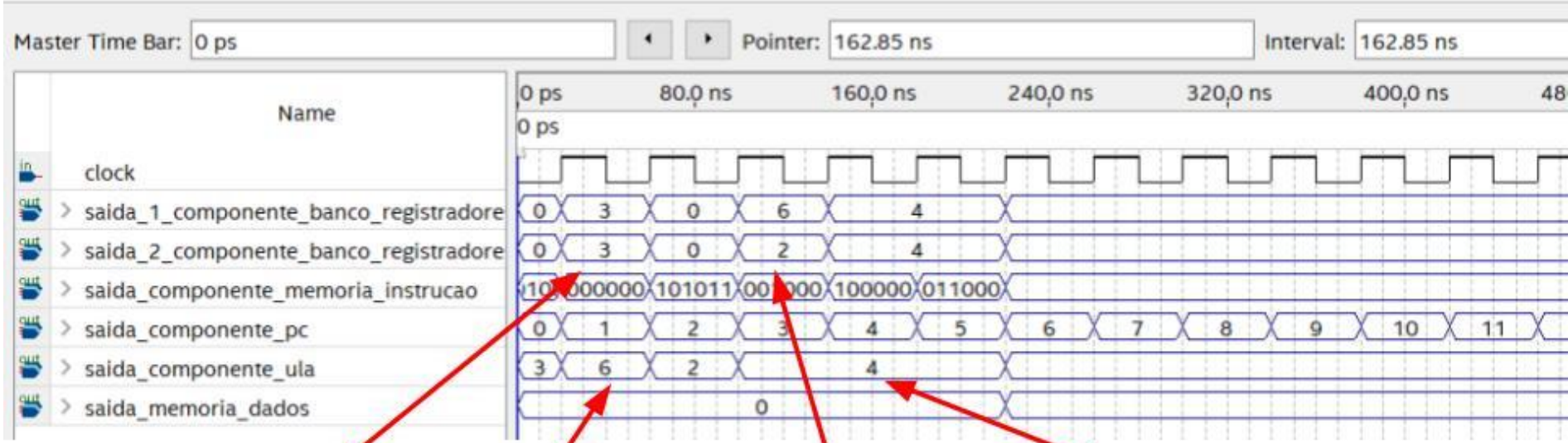
lw $s0 00

UFRR

Processador Split
Código 2 para testar processador

Departamento de CIÊNCIA DA COMPUTAÇÃO

Endereço	Linguagem de Alto Nível	Binário		
		Opcode	Reg2	Reg2
			Endereço	
			Dado	
0	li $s0 3	0000	00	00
1	add $s0 $s0 -- 6	0101	01	10
2	li $s1 2	0101	01	10
3	sub $s0 $s1 -- 6 - 2	0001	00	01
4	sw $s0 endereço(00)	0100	00	00
5	lw $s0 endereço(00)	0011	00	00



O valor 3 é carregado no registrador $s0

li $s0 3

Resultado da adição do valor de reg $s0

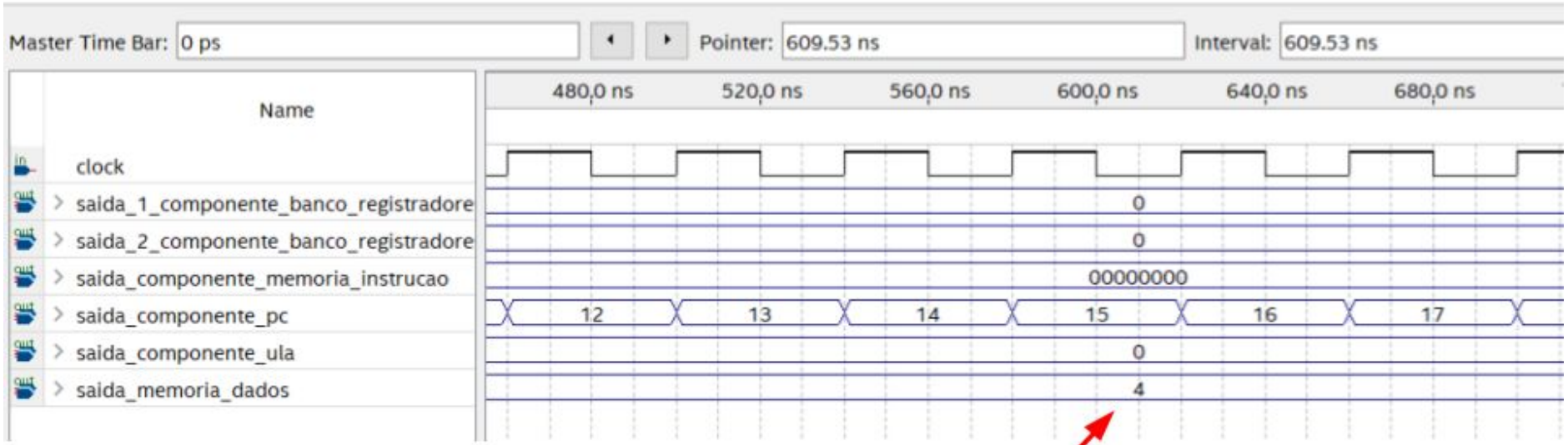
add $s0 $s0 6

Valor 2 é carregado no registrador $s1

li $s1 2

Corresponde ao resultado da subtração do registrador $s1 e $s0

sub $s0 $s1



O valor 4 do registrador $s0 foi carregado na memória de dados

lw $s0 endereço(00)

Processador Split
Limitações

Pelo fato de ser um processador de 8 bits só é possível realizar desvios condicionais e incondicionais entre a posição 0000 e 1111 da memória de instrução.

A instrução li só carrega valores entre 00 e 11 no registrador de destino.

As instruções load e store são feitas utilizando somente as quatro primeiras posições da memória de dados.

PATTERSON, D.; HENESSY, J. L. Organização e projeto de computadores: a interface hardware/software. 3ª Edição. São Paulo: Elsevier, 2005, 484 p.