Universidade de Brasília
Campus Universitário Darcy Ribeiro

Disciplina: Organização e Arquitetura de Computadores
Turma: A
Prof.: Marcus Vinicius Lamar

Laboratório 4
CPU RISC-V MULTICICLO

Abdullah Zaiter - 15/0089392
Daniel Bauchspiess - 15/0078901
Danielle Almeida Lima - 14/0135740
José Reinaldo da Cunha - 14/0169148
Lucas dos Santos Schiavini - 14/0150749

Brasília, 13 de junho de 2018.

- **Questão 2**

2.1) Primeiramente, projetamos o nosso processador multiciclo compatível com a ISA RV32I tomando como base o caminho de dados explicitado em aula, o resultado encontra-se na Figura 1. A máquina de estados do bloco de controle está representada na Figura 2. A implementação do nosso processador encontra-se na pasta Grupo4/LAB4/.

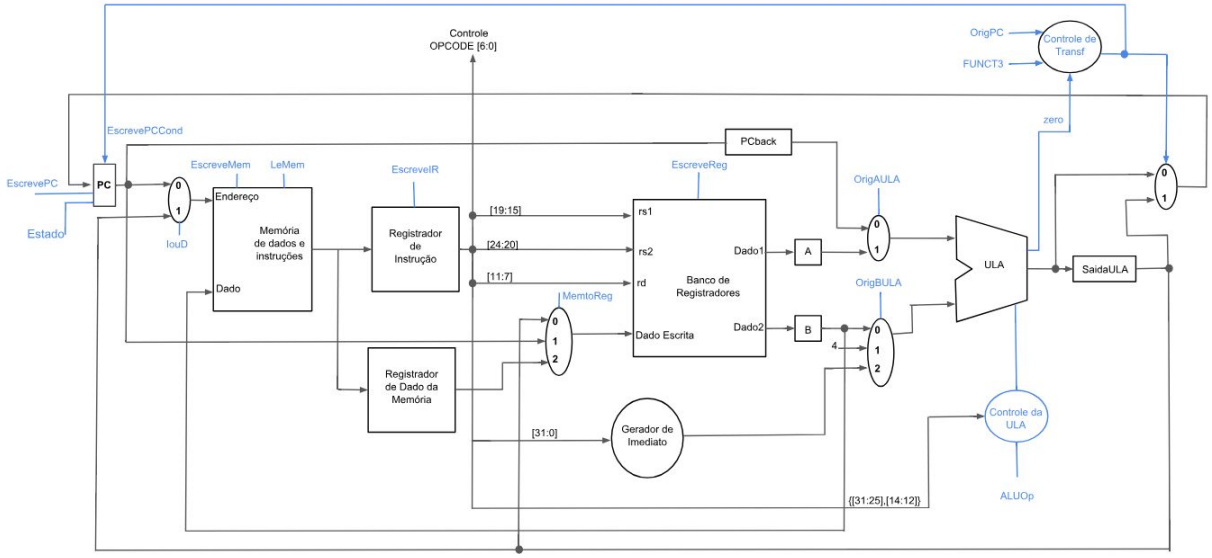


Figura 1 - Caminho de dados do processador multiciclo.

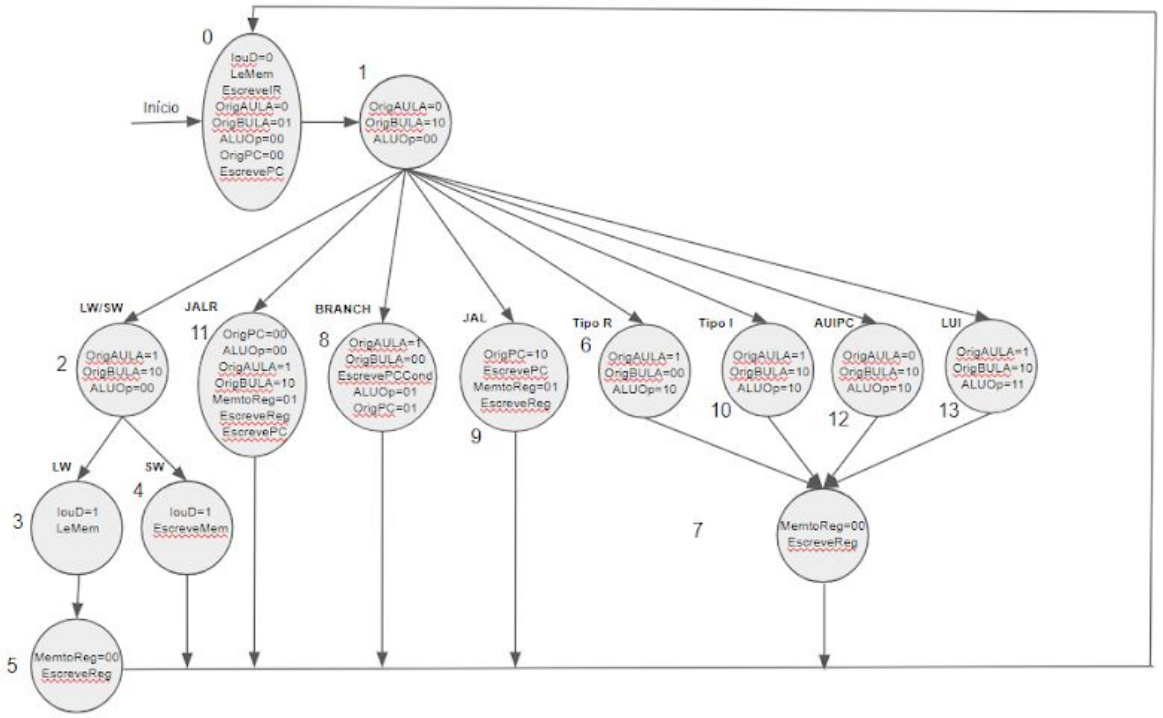


Figura 2 - Diagrama da máquina de estados do bloco de controle.

Como o controle geral não recebe a saída Zero da ULA nem o valor do funct3 da instrução, ele não é capaz de distinguir os tipos diferentes de branch, assim como determinar se o processador deve ou não pular ao próximo endereço. Por este motivo, foi criado o bloco controle de transferência, que recebe tais valores e determina corretamente o próximo endereço de PC para todas as instruções.

No nosso projeto do uniciclo decidimos colocar o ALUOp do LUI em 11 para que não fosse necessário o controle da ula receber o opcode, mantemos essa decisão de projeto para o processador multiciclo, por isso foi necessário criar um estágio separado na máquina de estados para o LUI, assim como é mostrado na Figura 2.

Para a implementação da máquina de estados, utilizamos a estratégia de separar as instruções em grupos que podem ser executados pelas mesmas sequências de operações. Esta etapa é facilitada uma vez que na própria implementação da ISA RV32 as instruções já são separadas de modo similar pelos campos de OPCODE. Os grupos definidos nesta primeira etapa foram: operações de load e store, JALR, JAL, operações de BRANCH, operações tipo R, operações tipo I e o LUI, AUIPC. A segunda etapa do projeto foi a definição dos estados necessários para o correto funcionamento de cada grupo. A terceira etapa foi a realização de que algumas das etapas eram análogas e seria ineficiente ter mais de um estado com os mesmos sinais de controle, deste modo tais estados foram agrupados em um só estado. A partir destas etapas foi criado o espaço de estados apresentado na figura 2.

Uma das maiores dificuldades foi resolver os branchs, uma vez que os saltos não eram realizados quando a condição era verdadeira, para isso modificamos a condição de escrita do PC. Antes havia as condições EscreveCond(saída do Controle de Transferência) que simbolizava um Branch bem sucedido, além da condição PCWrite para instruções comuns. Com isso os Branches, JAL e JALR funcionaram, no entanto ainda assim pulavam para um endereço errado. Portanto, além do supracitado foram adicionadas condições baseadas no Estado atual da instrução. Isso foi realizado, pois percebemos um acréscimo de 4 no endereço correto esperado para Jumps e Branches.

Para a implementação do processador multiciclo utilizamos por base partes do processador uniciclo já implementado para o laboratório 3. No entanto algumas partes deste novo processador diferem grandemente daquele produzido anteriormente. As maiores mudanças se dão no caminho de dados e no controle do processador. O caminho de dados foi reduzido uma vez que a mesma unidade pode, agora, ser utilizada mais de uma vez, como a memória e a ULA. O controle também sofreu grande alteração uma vez que agora este módulo é implementado através de uma máquina de estados, diferentemente do controle do uniciclo que era totalmente combinacional.

2.2) A Figura 3 apresenta os resultados encontrados para os requisitos físicos da implementação do nosso processador multiciclo compatível com a ISA RV32I.

Flow Status	Successful - Mon Jun 18 23:22:37 2018
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	TopDE
Top-level Entity Name	TopDE
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	3,137 / 32,070 (10 %)
Total registers	4288
Total pins	241 / 457 (53 %)
Total virtual pins	0
Total block memory bits	832,512 / 4,065,280 (20 %)
Total DSP Blocks	12 / 87 (14 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	2 / 6 (33 %)
Total DLLs	0 / 4 (0 %)

Figura 3: Requisitos Físicos

- Número de Elementos Lógicos (ALMs): 3,137 / 32,070 (10 %)
- Número de Registradores: 4288
- Quantidade de bits de memória: 832,512 / 4,065,280 (20 %)
- Número de blocos DSP usados: 12 / 87 (14 %)

2.3) Com o auxílio do TimeQuest e um clock de 50 MHz, encontramos:

- Caminho de maior atraso, tpd: FF=18.978 ns de SW[9] para VGA_G[5]
- Se algum requerimento não foi atendido: todos os requerimentos foram atendidos
- th: 4.629
- tsu: 2.277
- tco: 30.894
- slack setup: 3.982
- slack hold: -1.932
- slack minimum pulse width: 7.313
- Frequência máxima utilizável: 62.43 MHz

2.4) A implementação do processador uniciclo da ISA RV32IM foi feita utilizando por base a ISA RV32I apresentada anteriormente, deve-se somente definir qual a ISA desejada no arquivo topDE. Como reutilizamos a ULA, controle da ULA e parâmetros do nosso processador uniciclo, desse modo não precisamos fazer alterações nos arquivos. O caminho de dados é o mesmo da Figura 1, assim como o diagrama da máquina de estados do controle

da Figura 2 é o mesmo para a ISA RV32IM, uma vez que as instruções de multiplicação e divisão são do tipo R.

2.5) A Figura 4 apresenta os resultados encontrados para os requisitos físicos da implementação do nosso processador multiciclo compatível com a ISA RV32IM.

Flow Status	Successful - Mon Jun 18 23:41:20 2018
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	TopDE
Top-level Entity Name	TopDE
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	3,138 / 32,070 (10 %)
Total registers	4341
Total pins	241 / 457 (53 %)
Total virtual pins	0
Total block memory bits	832,512 / 4,065,280 (20 %)
Total DSP Blocks	12 / 87 (14 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0

Figura 4: Requisitos Físicos

- Número de Elementos Lógicos (ALMs): 3,138 / 32,070 (10 %)
- Número de Registradores: 4341
- Quantidade de bits de memória: 832,512 / 4,065,280 (20%)
- Número de blocos DSP usados: 12 / 87 (14%)

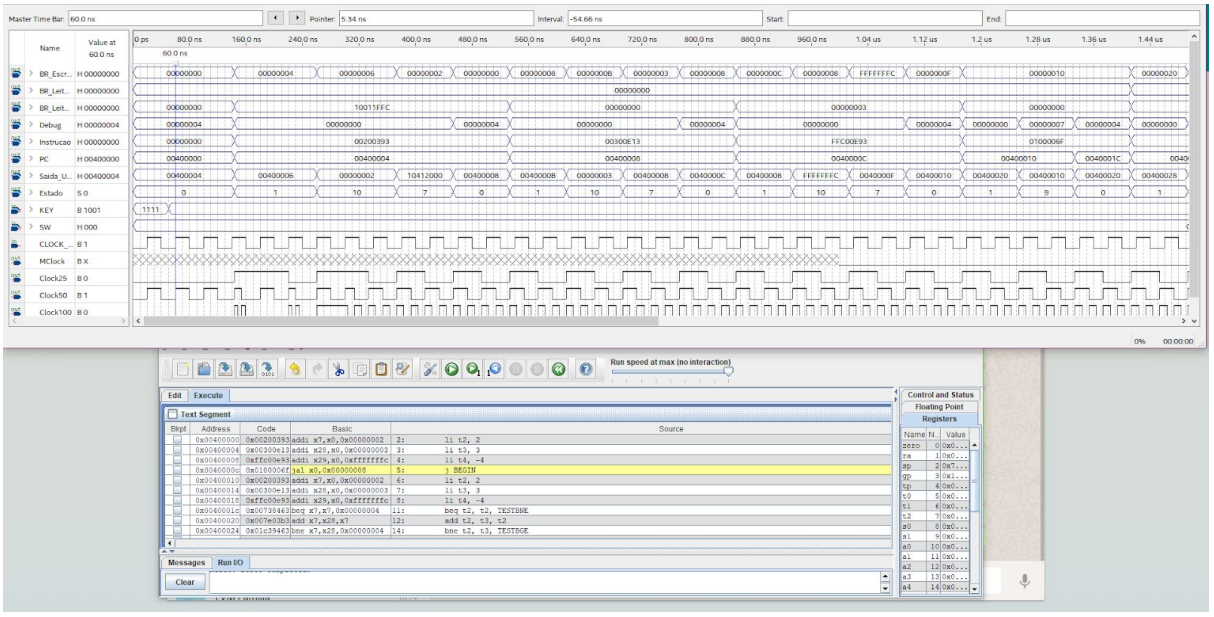
2.6) Com o auxílio do TimeQuest e um clock de 50 MHz, encontramos:

- Caminho de maior atraso, tpd: 14.646
- Se algum requerimento não foi atendido: não, todos foram atendidos.
- th: 0.614
- tsu: 3.496
- tco: 5.883
- slack setup: 11.163
- slack hold: 0.149
- slack minimum pulse width: 15.286
- Frequência máxima utilizável: 90.85 MHz

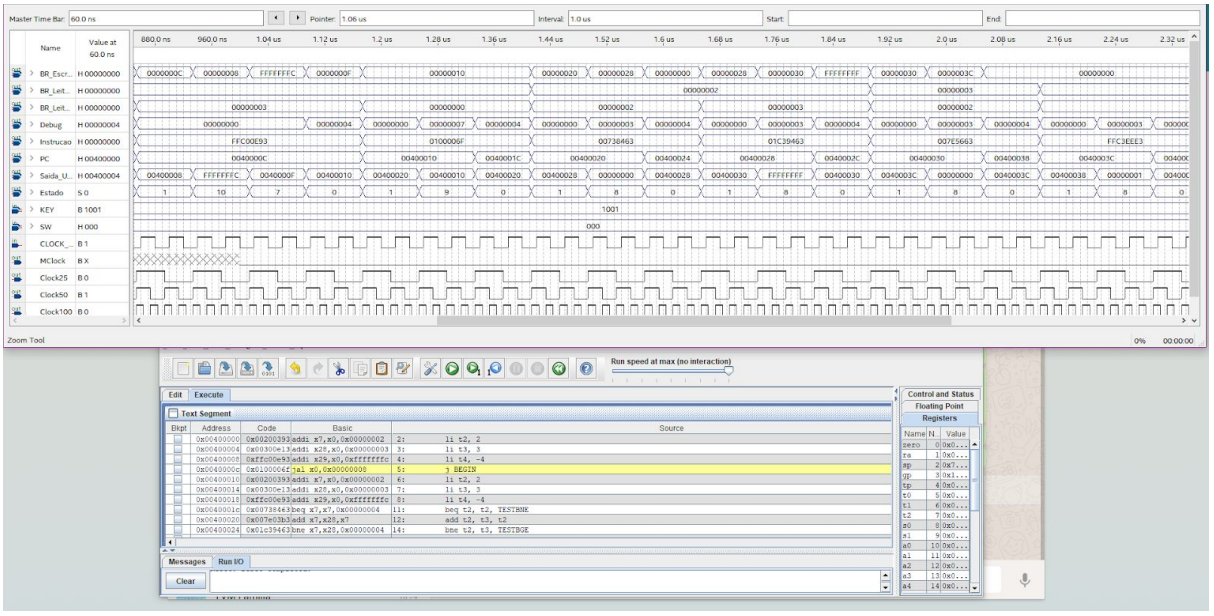
- **Questão 3**

A simulação por forma de onda foi feita no quartus, mas devido à baixa resolução das imagens no relatório, as mesmas se encontram numa pasta chamada Imagens Relatórios, e os testbenches utilizados foram colocados numa pasta TestBench para referência. Assim, temos as seguintes operações realizadas no processador:

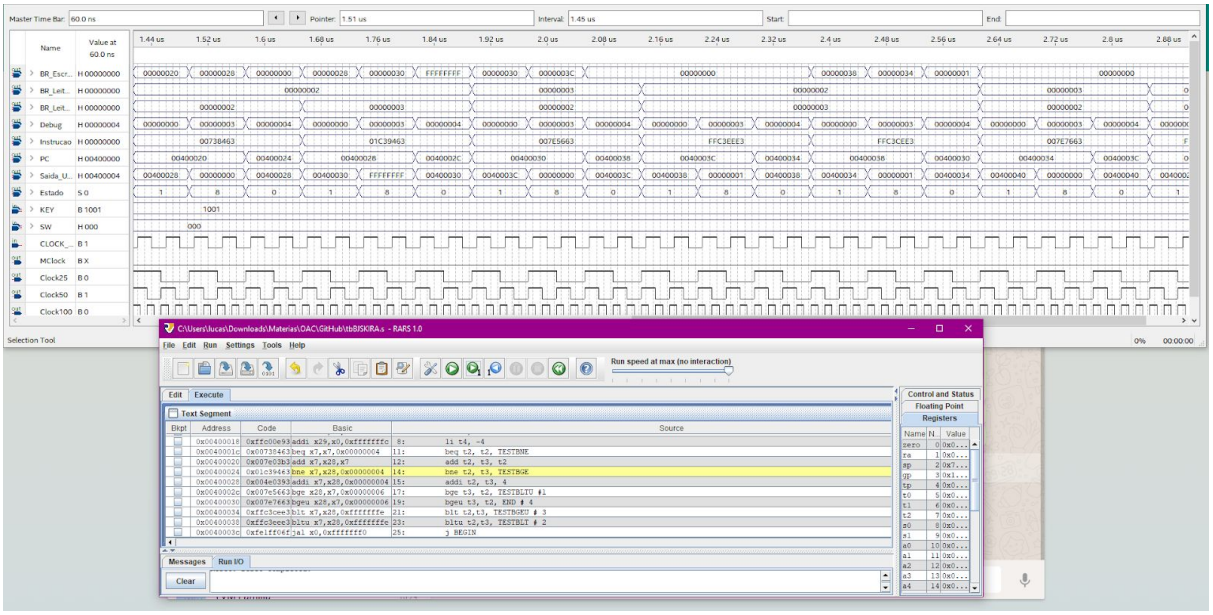
Branches e Jumps:



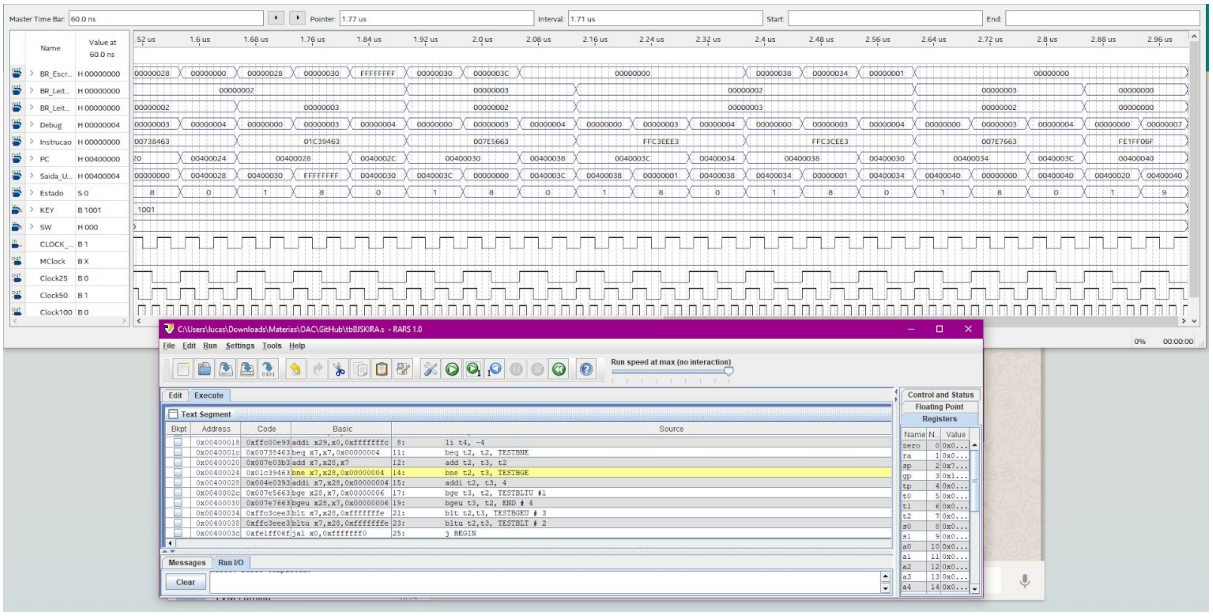
Teste Branches e JUMPS Parte 1: O estado 8 é o branch e o 9 é o jump.



Teste Branches e JUMPS Parte 2: O estado 8 é o branch e o 9 é o jump.



Teste Branches e JUMPS Parte 3: O estado 8 é o branch e o 9 é o jump.

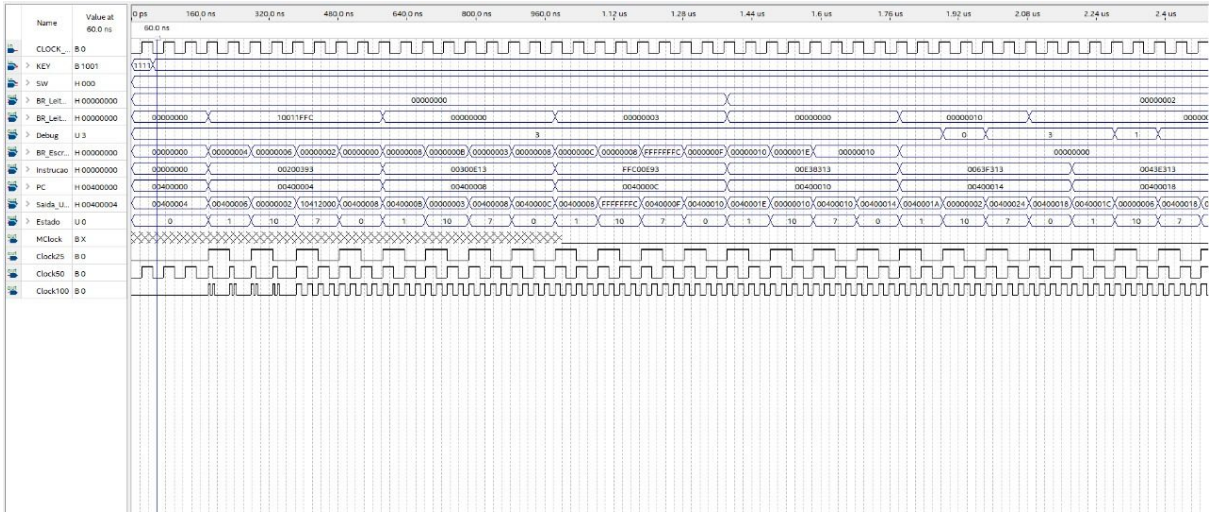


Teste Branches e JUMPS Parte 4: O estado 8 é o branch e o 9 é o jump.

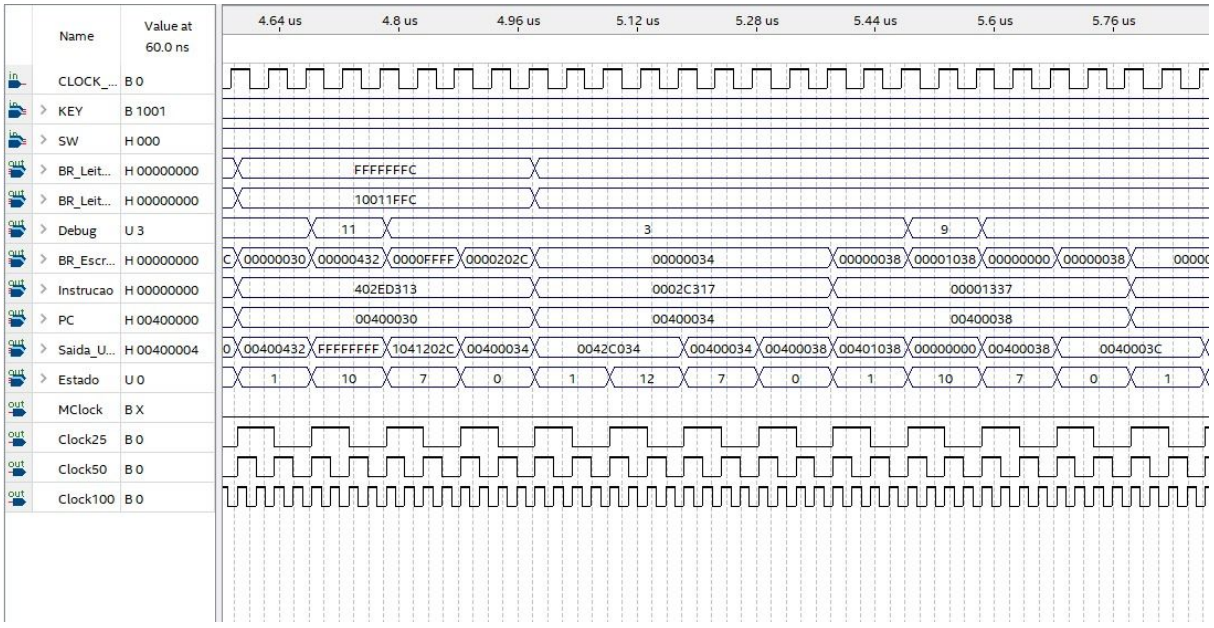
Operações tipo I:

Bkpt	Address	Code	Basic		Source
	0x00400000	0x00200393	addi x7,x0,0x00000002	2:	li t2, 2
	0x00400004	0x00300e13	addi x28,x0,0x00000003	3:	li t3, 3
	0x00400008	0xffc00e93	addi x29,x0,0xfffffffc	4:	li t4, -4
	0x0040000c	0x00e38313	addi x6,x7,0x0000000e	6:	addi t1, t2, 14 # t1 = 16
	0x00400010	0x0063f313	andi x6,x7,0x00000006	7:	andi t1, t2, 6 # t1 = 2
	0x00400014	0x0043e313	ori x6,x7,0x00000004	8:	ori t1, t2, 4 # t1 = 6
	0x00400018	0x0063c313	xori x6,x7,0x00000006	9:	xori t1, t2, 6 # t1 = 4
	0x0040001c	0x0033a313	slti x6,x7,0x00000003	10:	slti t1, t2, 3 # t1 = 1
	0x00400020	0x003eb313	sltiu x6,x29,0x0000...	11:	sltiu t1, t4, 3 # t1 = 0
	0x00400024	0x00139313	slli x6,x7,0x00000001	12:	slli t1, t2, 1 # t1 = 4
	0x00400028	0x0013d313	srli x6,x7,0x00000001	13:	srli t1, t2, 1 # t1 = 1
	0x0040002c	0x402ed313	srai x6,x29,0x00000002	14:	srai t1, t4, 2 # t1 = -1
	0x00400030	0x0002c317	auipc x6,0x0000002c	16:	auipc t1, 44 # t1 = pc da proxima instr
	0x00400034	0x00001337	lui x6,0x00000001	17:	lui t1, 1 # t1 = 4096 (h1000)

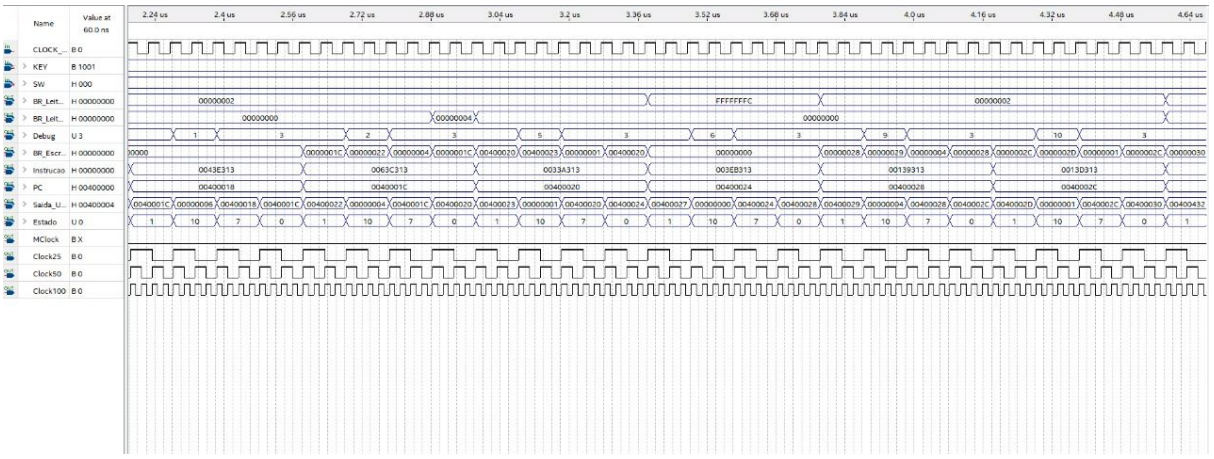
Testebench para as instruções do tipo I, auipc e lui no RARS



Parte 1: Teste para as instruções do tipo I, auipc e lui

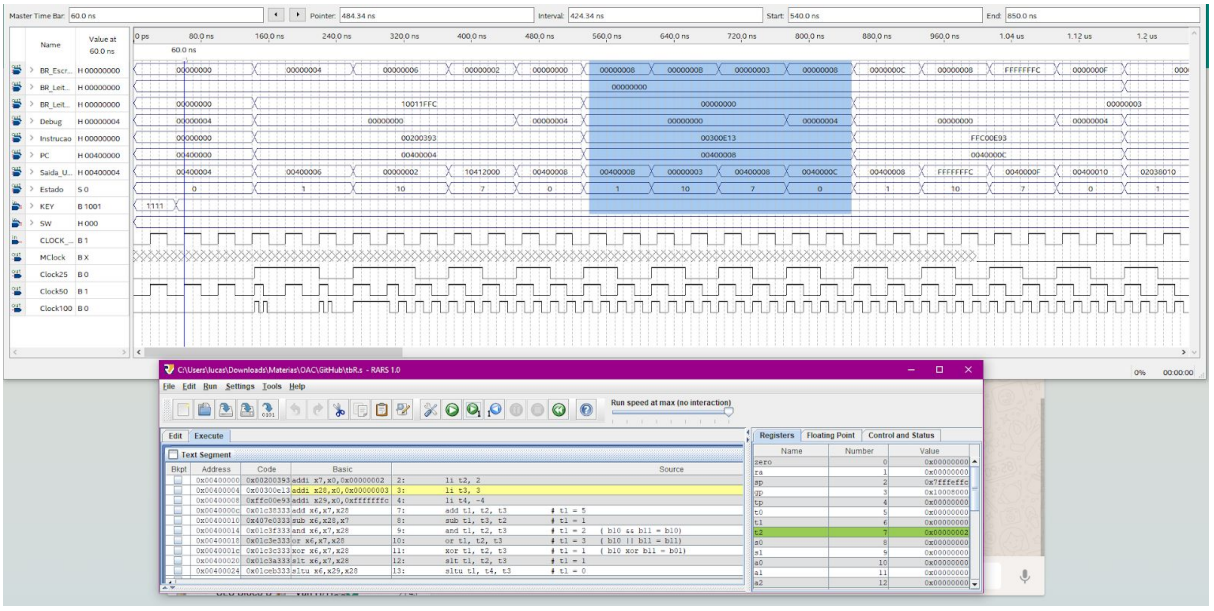


Parte2: Teste para as instruções do tipo I, auipc e lui

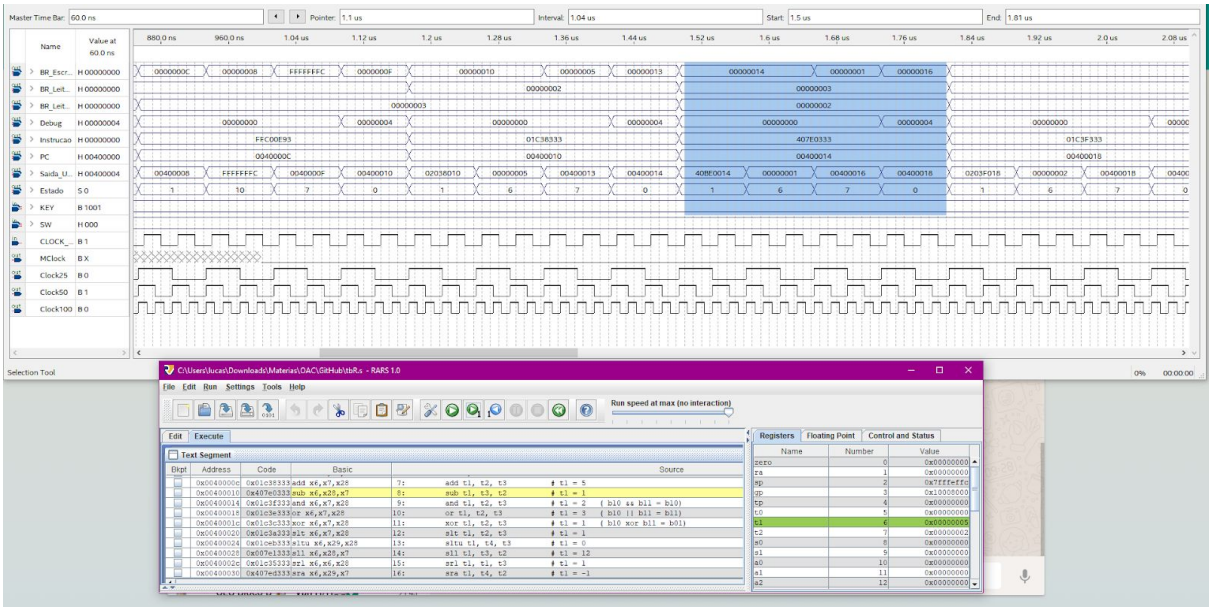


Parte3: Teste para as instruções do tipo I, auipc e lui

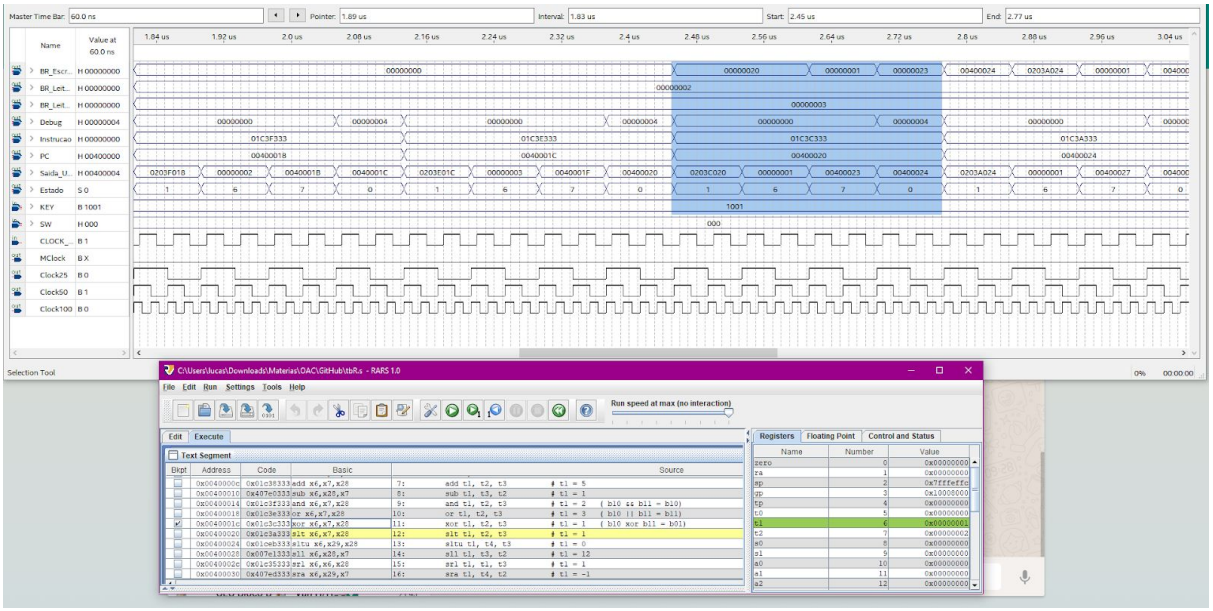
Operações tipo R:



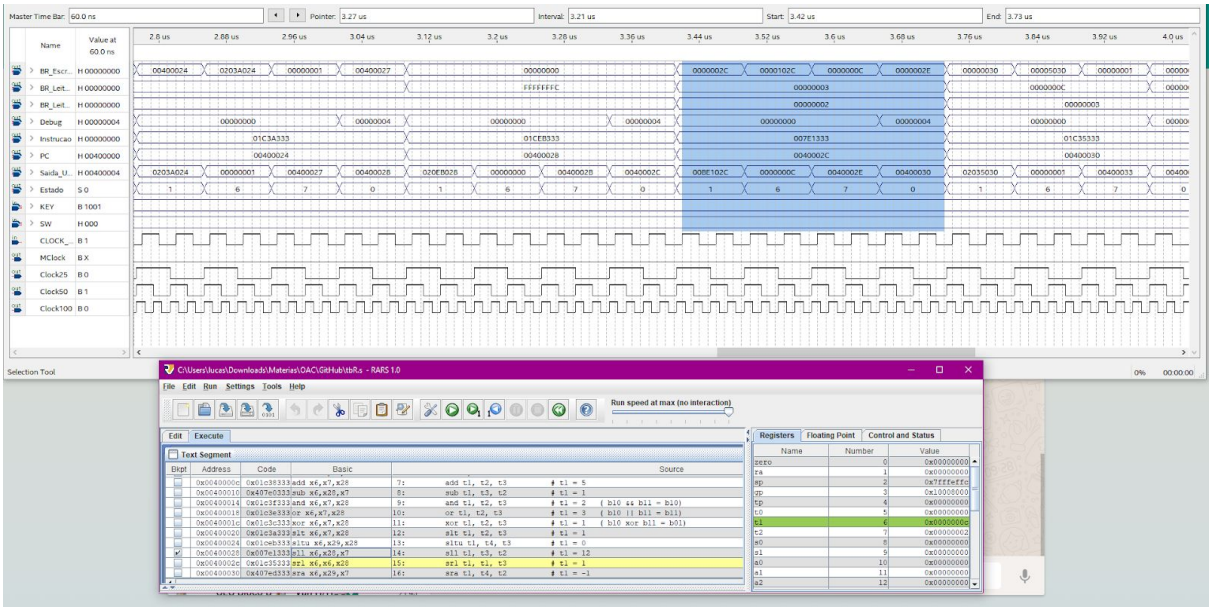
Parte 1: Teste para Tipo R



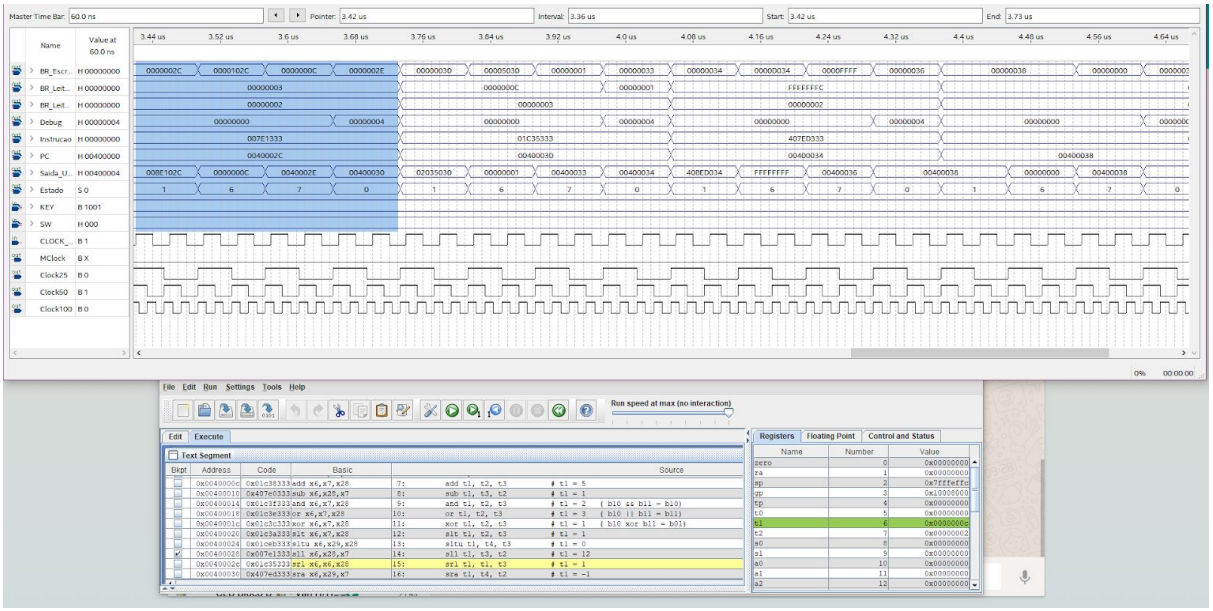
Parte 2: Teste para Tipo R



Parte 3: Teste para Tipo R

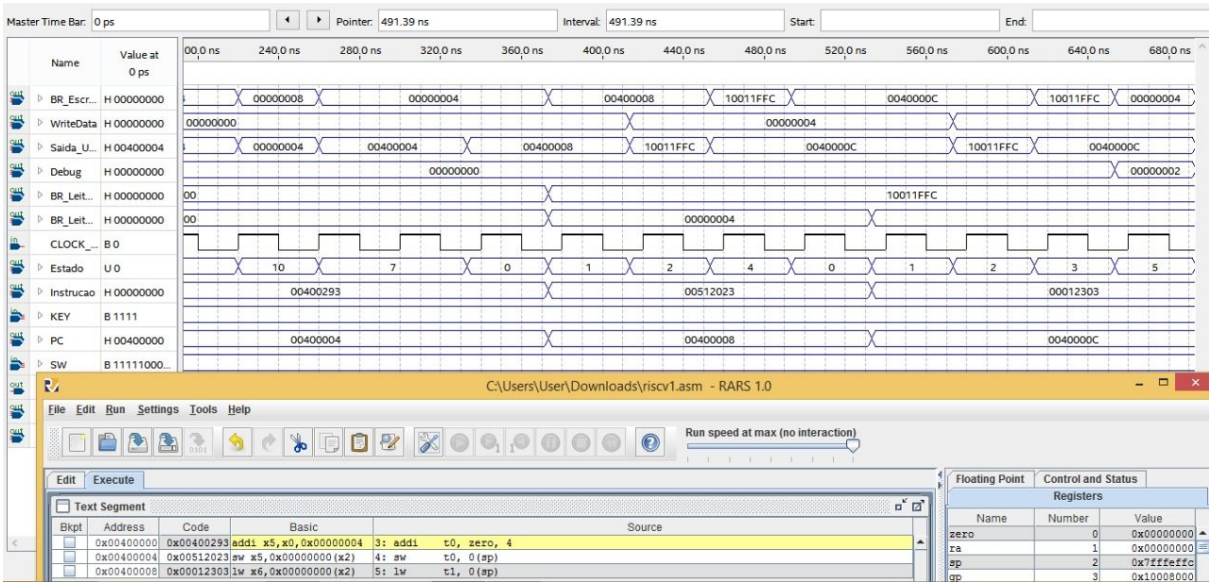


Parte 4: Teste para Tipo R



Parte 5: Teste para Tipo R

Operação LW e SW:



Teste para as instruções lw e sw.