Laboratório de Arquitetura e Organização de Computadores II

Prática II

Nome do Aluno: Gabriel Alves Barbosa e Gabriel Luís Silva Pereira

Obs: Editores de texto reduzem a qualidade e nitidez das imagens. Assim, caso esteja difícil a leitura e interpretação de alguma imagem, há uma cópia de todas elas na pasta “imagens das simulações”, neste mesmo arquivo zip onde se encontra este relatório.

Testes das Instruções:

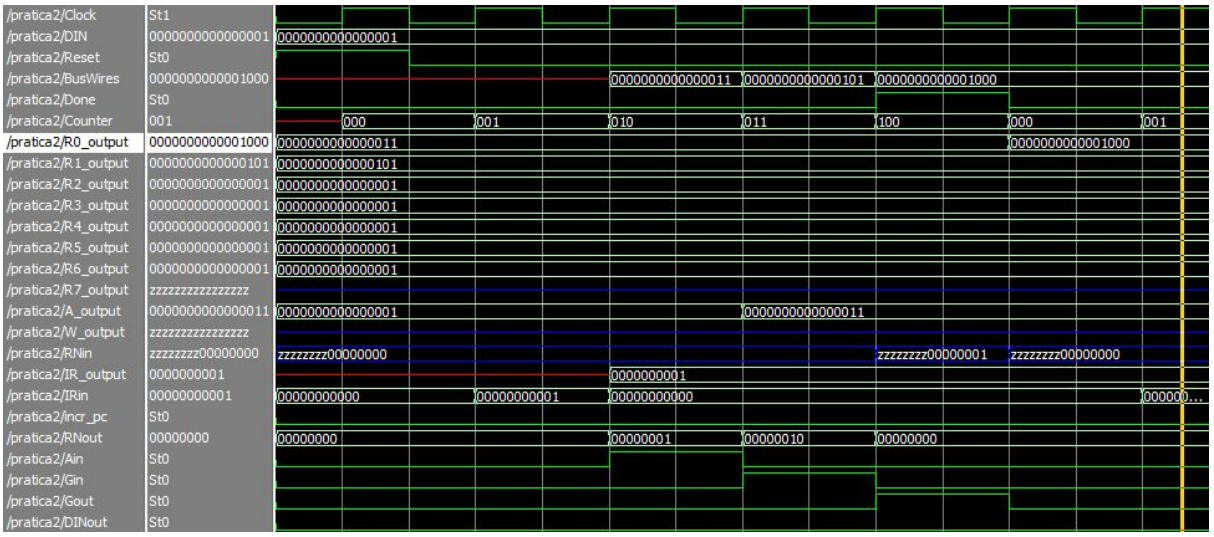
Demonstraremos aqui o funcionamento aqui das Instruções Básicas e das Instruções extras.

1) Instruções básicas:

- **ADD:**

Exemplo de ADD executado: 0000000000000001;
OpCode: 000
Registradores utilizados:
Rx = 000;
Ry = 001;

A imagem a seguir demonstra o funcionamento da função ADD. Repare que os registradores de R0 a R7 haviam sido previamente carregados com alguns valores. Especificamente, R0 está com o valor 3 e R1 está com o valor 5. Nossa função Add implementa uma soma do tipo Rx = Rx + Ry. Neste caso, Rx = R0 e Ry = R1. Assim, quando counter atingiu o valor 4 o sinal “done” ficou ativo, e o valor 8 (5+3) foi escrito no registrador R0.

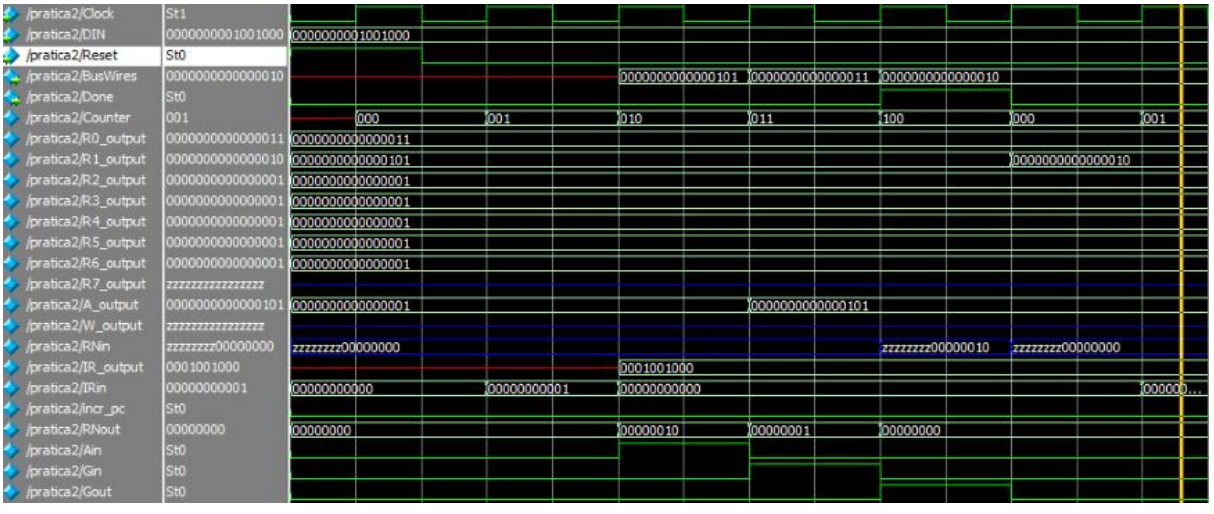


- **SUB:**

Exemplo de SUB executado: 0000000001001000;
OpCode: 001
Registradores utilizados:

Rx = 001;
Ry = 000;

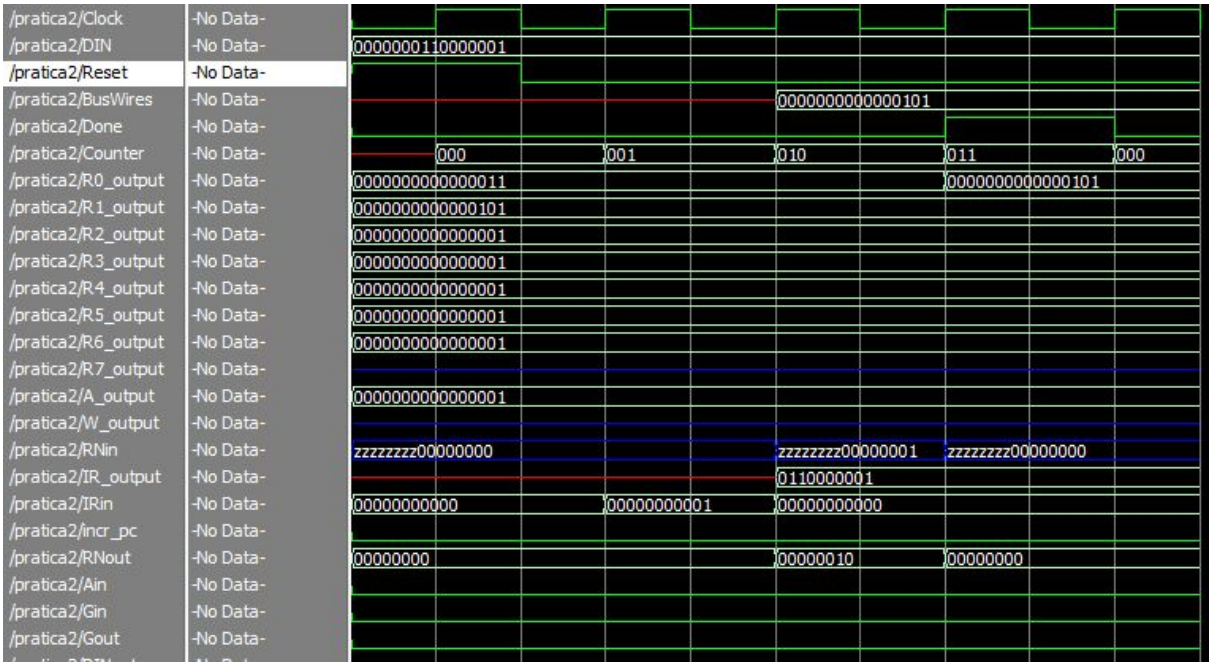
A imagem a seguir demonstra o funcionamento da função SUB. Repare que os registradores de R0 a R7 haviam sido previamente carregados com alguns valores. Especificamente, R0 está com o valor 3 e R1 está com o valor 5. Nossa função SUB implementa uma subtração do tipo Rx = Rx - Ry. Neste caso, Rx = R1 e Ry = R0. Assim, quando counter atingiu o valor 4 o sinal “done” ficou ativo, e o valor 2 (5-3) foi escrito no registrador R1.



- **MV:**

Exemplo de MV executado: 0000000110000001;
OpCode: 110
Registradores utilizados:
Rx = 000;
Ry = 001;

A imagem a seguir demonstra o funcionamento da função MV. Repare que os registradores de R0 a R7 haviam sido previamente carregados com alguns valores. Especificamente, R0 está com o valor 3 e R1 está com o valor 5. Nossa função MV implementa uma operação do tipo Rx = Ry. Neste caso, Rx = R0 e Ry = R1. Assim, quando counter atingiu o valor 3 o sinal “done” ficou ativo, e o valor 5 (que estava em R1) foi escrito no registrador R0.



- **MVI:**

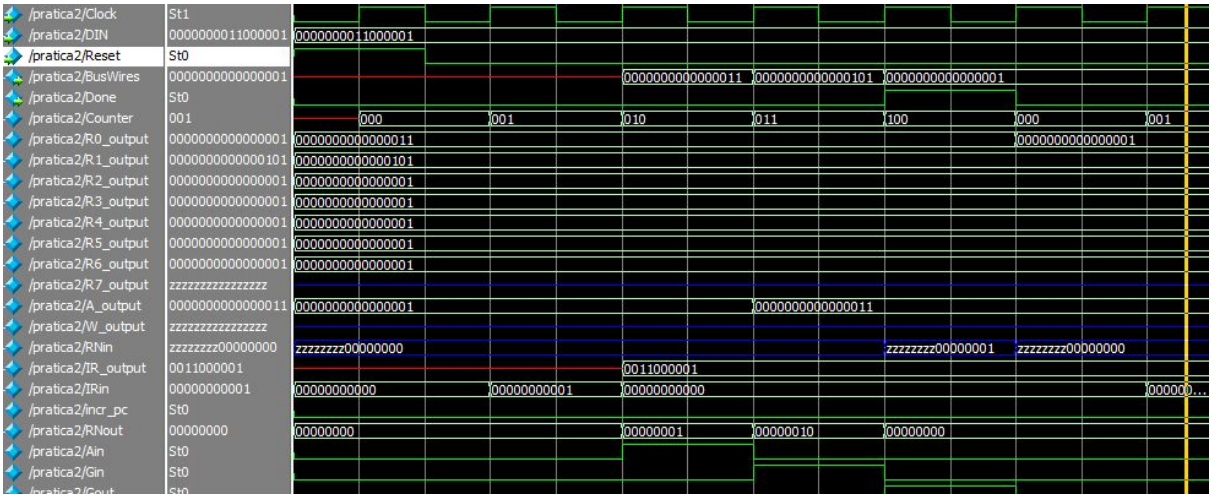
Infelizmente não conseguimos implementar a instrução MVI. Temos algumas ideias do que pode ter acontecido, mas não conseguimos implementá-las em tempo hábil. A fim de ser possível concluir a prática, ignoramos a instrução de MVI nos casos testes do projeto. De positivo, temos que esta foi a única falha do projeto.

2) Instruções Adicionais:

- **SLT:**

Exemplo de SLT executado: 0000000011000001;
OpCode: 011
Registradores utilizados:
Rx = 000;
Ry = 001;

A imagem a seguir demonstra o funcionamento da função SLT. Repare que os registradores de R0 a R7 haviam sido previamente carregados com alguns valores. Especificamente, R0 está com o valor 3 e R1 está com o valor 5. Nossa função SLT implementa uma operação do tipo Rx = (Rx < Ry) ? 1 : 0. Neste caso, Rx = R0 e Ry = R1. Assim, quando counter atingiu o valor 4 o sinal “done” ficou ativo, e o valor 1 (pois 3 é menor do que cinco) foi escrito no registrador R0.

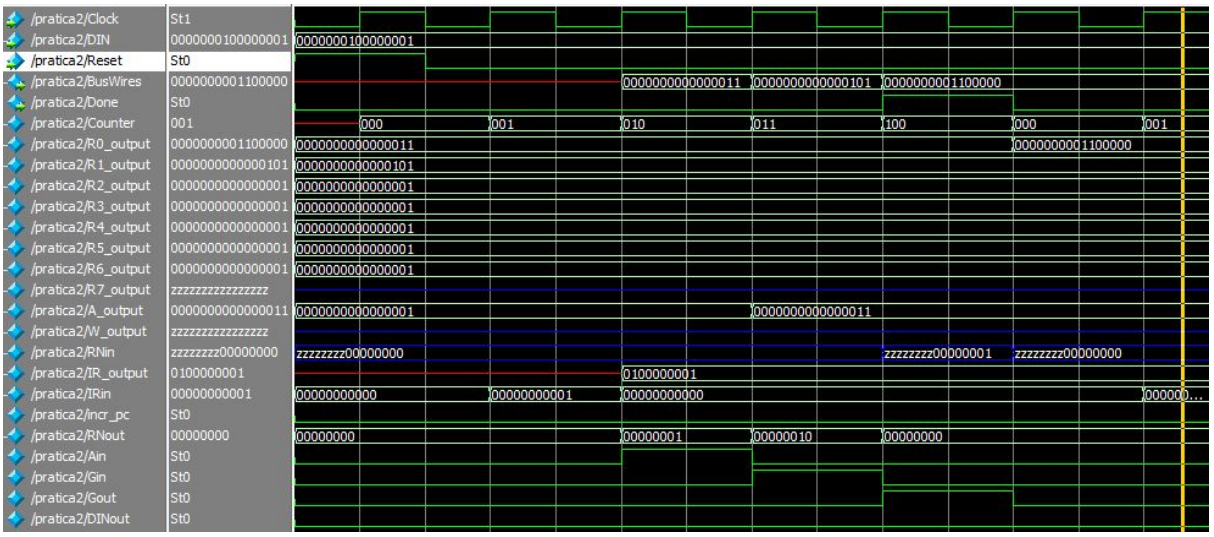


- **SLL:**

Exemplo de SLL executado: 0000000100000001;
OpCode: 100
Registradores utilizados:

Rx = 000;
Ry = 001;

A imagem a seguir demonstra o funcionamento da função SLL. Repare que os registradores de R0 a R7 haviam sido previamente carregados com alguns valores. Especificamente, R0 está com o valor 3 e R1 está com o valor 5. Nossa função SLL implementa uma operação do tipo Rx = [Rx] << [Ry] . Neste caso, Rx = R0 e Ry = R1. Assim, quando counter atingiu o valor 4 o sinal “done” ficou ativo, e o valor 96 (pois o valor 3 foi multiplicado por 2 cinco vezes) foi escrito no registrador R0.

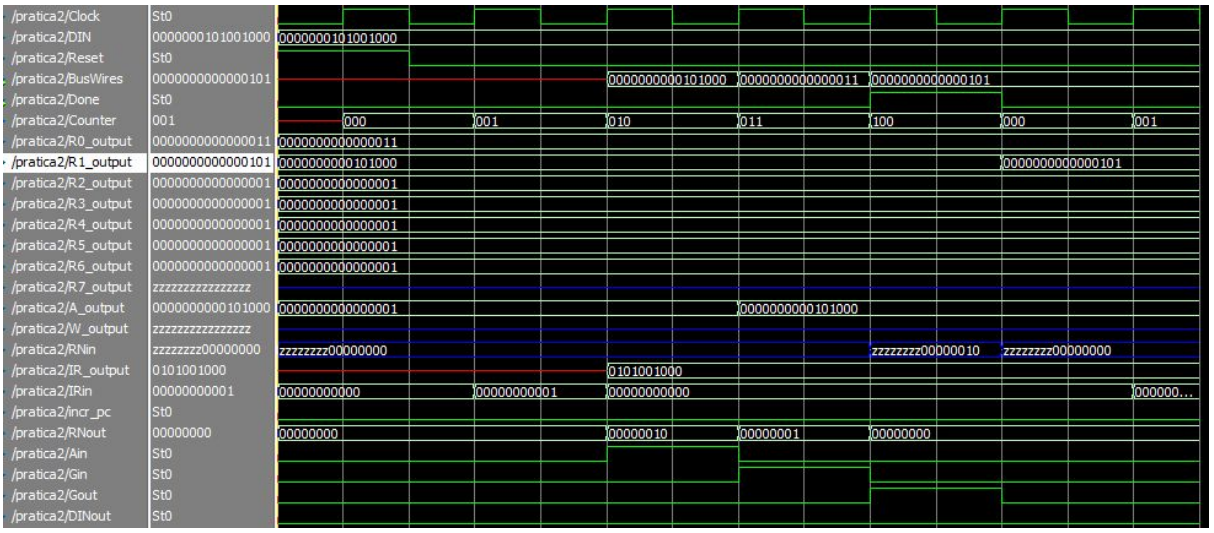


- **SRL:**

Exemplo de SRL executado:0000000101001000 ;
OpCode: 101
Registradores utilizados:

Rx = 001;
Ry = 000;

A imagem a seguir demonstra o funcionamento da função SRL. Repare que os registradores de R0 a R7 haviam sido previamente carregados com alguns valores. Especificamente, R0 está com o valor 3 e R1 está com o valor 40. Nossa função SLL implementa uma operação do tipo Rx = [Rx] >> [Ry] . Neste caso, Rx = R0 e Ry = R1. Assim, quando counter atingiu o valor 4 o sinal “done” ficou ativo, e o valor 5(pois o valor 40 foi dividido por 2 três vezes) foi escrito no registrador R0.



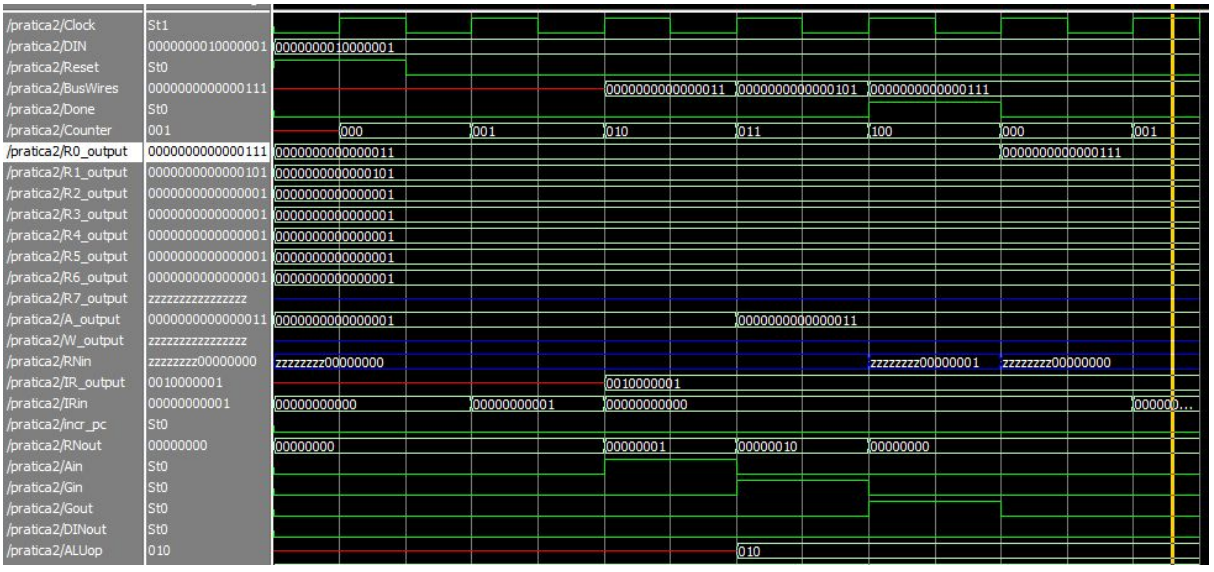
- **OR:**

Exemplo de OR executado: 0000000010000001;
OpCode: 010
Registradores utilizados:

Rx = 000;
Ry = 001;

A imagem a seguir demonstra o funcionamento da função OR. Repare que os registradores de R0 a R7 haviam sido previamente carregados com alguns valores. Especificamente, R0 está com o valor 3 e R1 está com o valor 5. Nossa função OR implementa uma operação do tipo Rx ← [Rx] | [Ry]. Neste caso, Rx = R0 e Ry = R1. Assim, quando counter atingiu o valor 4 o sinal “done” ficou ativo, e o valor 7 (que resulta do or entre 3 e t em binário) foi escrito no registrador R0.

OR:



Lendo as instruções da memória Rom:

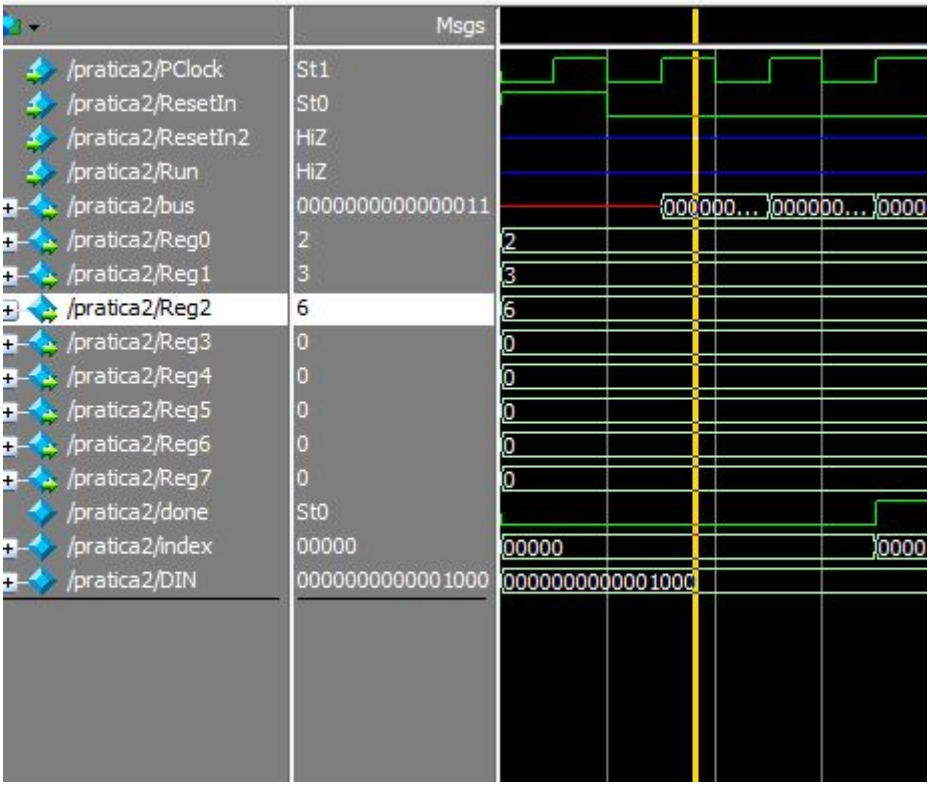
Como não conseguimos implementar a instrução mvi (única instrução que ficou faltando), tivemos que removê-la do exemplo de testes. Assim, o nosso teste, com o resultado esperado para cada um dos registradores encontra-se abaixo:

Instrução	R0	R1	R2	R3	R4
Estado inicial	2	3	6	0	0
ADD R1, R0	2	5	6	0	0
SUB R2, R1	2	5	1	0	0
MV R3, R2	2	5	1	1	0
ADD R0, R3	3	5	1	1	0
OR R1,R0	3	7	1	1	0
SUB R1,R0	3	4	1	1	0
ADD R1, R3	3	5	1	1	0
SLL R1, R3	3	A	1	1	0
SRL R1, R3	3	5	1	1	0
SLT R4, R1	3	5	1	1	1
SLT R1, R1	3	0	1	1	1
ADD R4, R3	3	0	1	1	2
ADD R1, R2	3	1	1	1	2

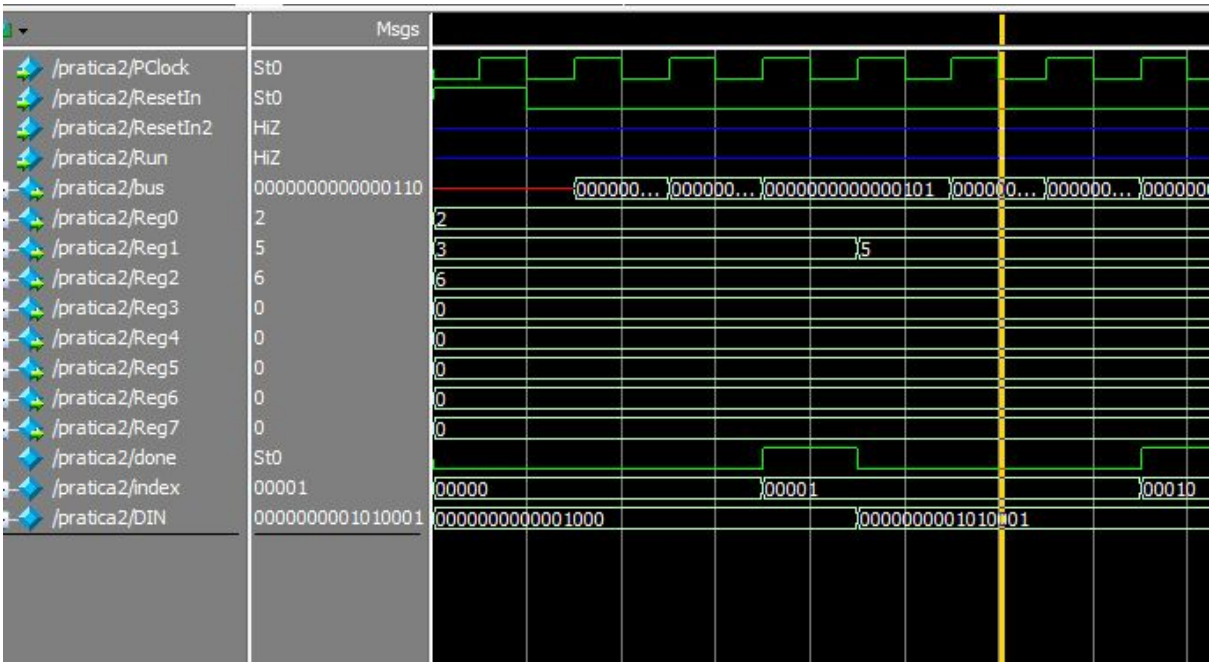
A solução para implementar o nosso processador ligado a uma memória de instruções foi a seguinte: Primeiramente ligamos um clock ao processador. Ligamos a saída done do processador como o clock da memória. Ligamos o dado de saída da memória ao dado de entrada do processador.

O resultado do teste aplicado ao nosso processador pode ser observado na figura abaixo:

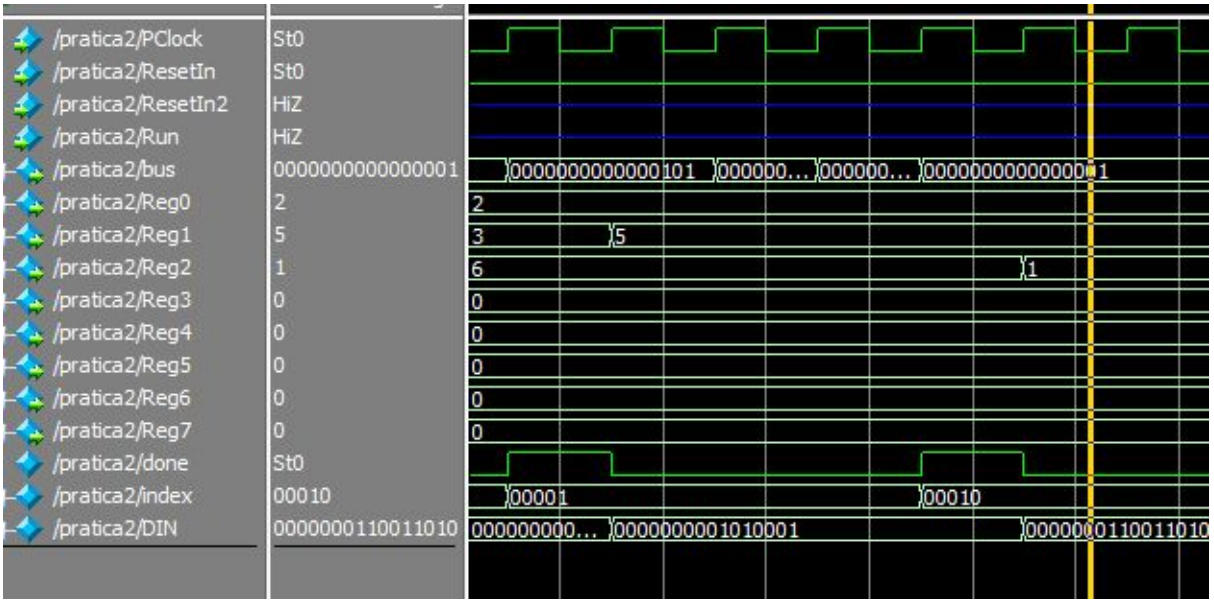
Estado Inicial:



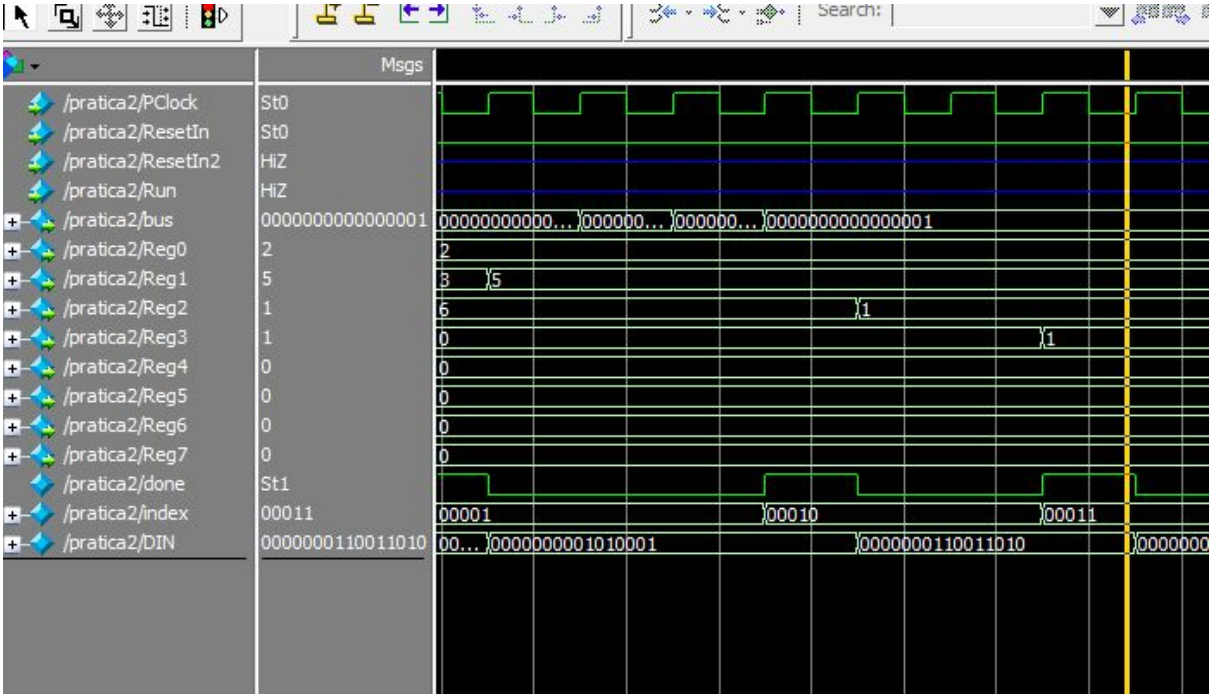
ADD R1, R0:



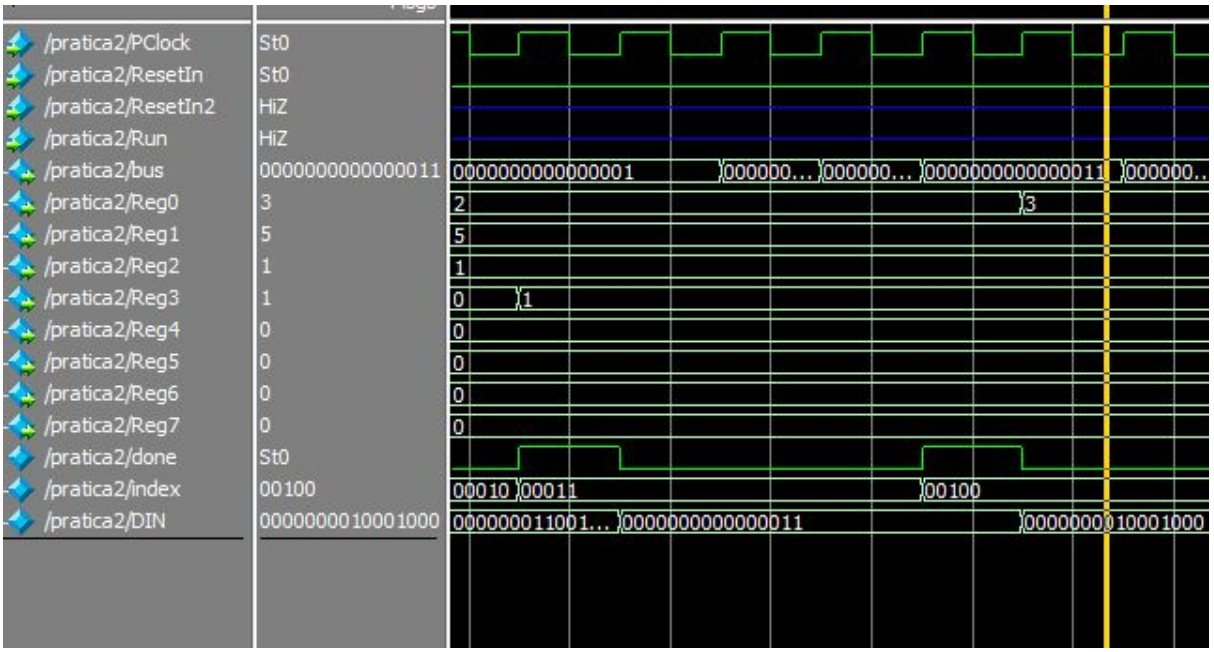
SUB R2, R1:



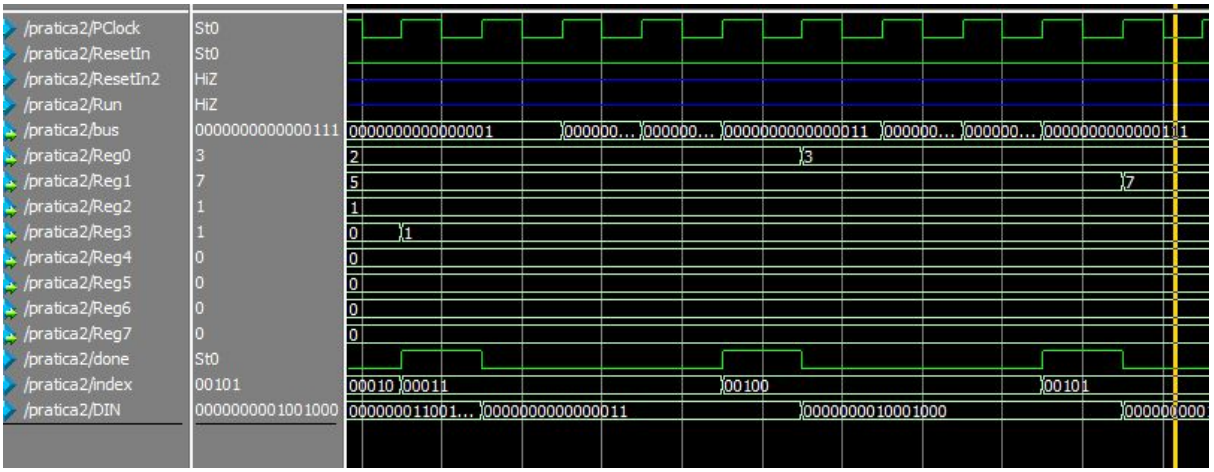
MV R3, R2:



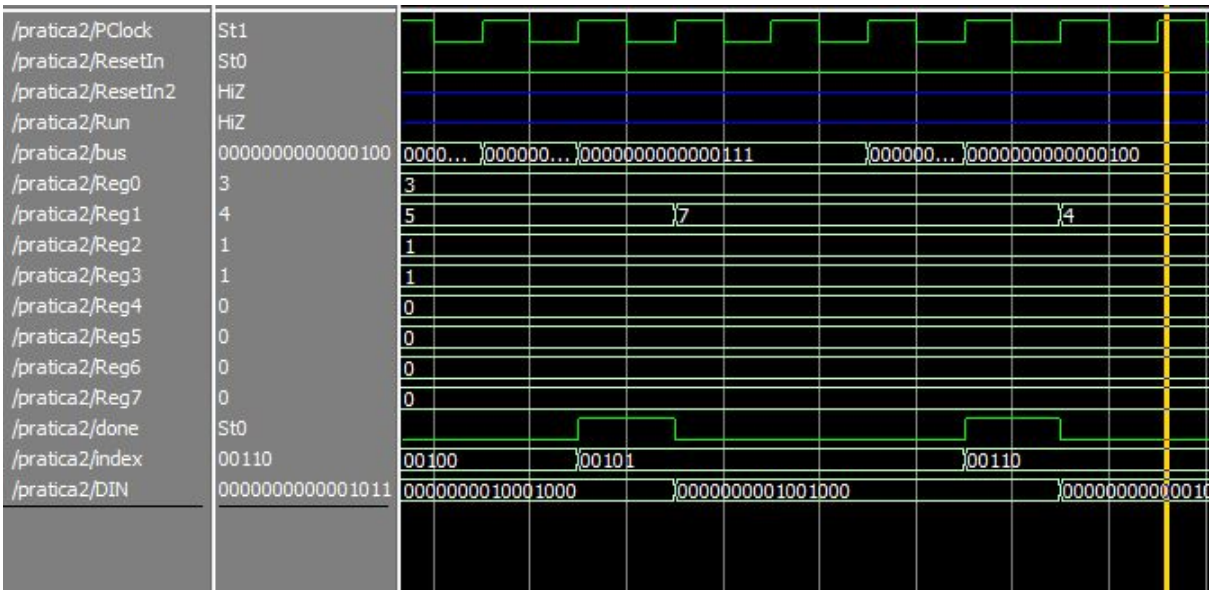
ADD R0, R3:



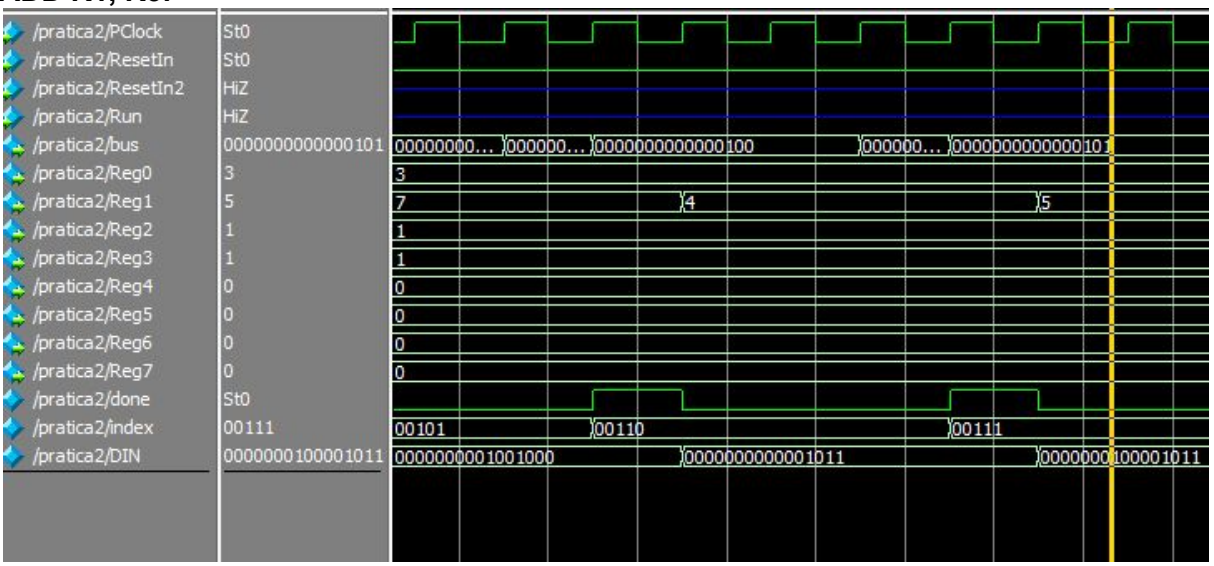
OR R1,R0:



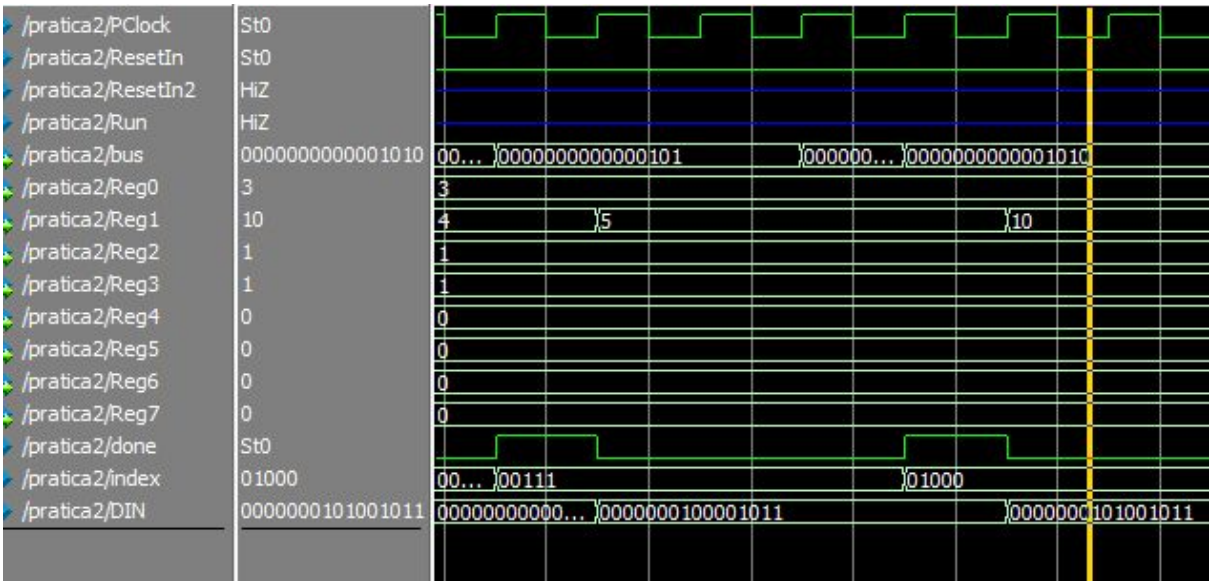
SUB R1,R0:



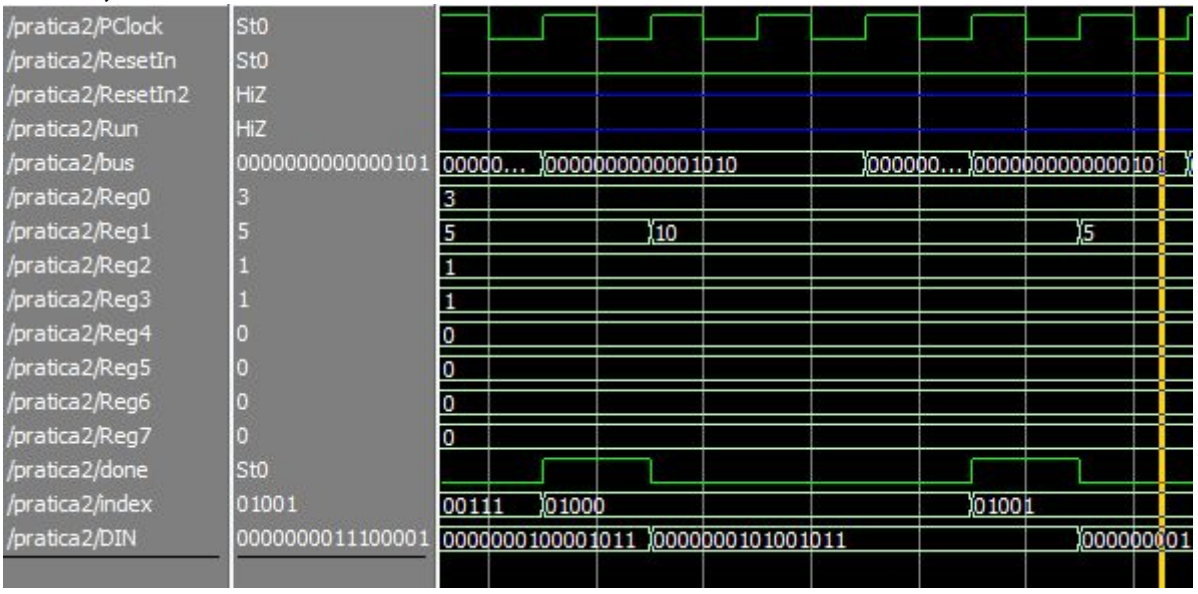
ADD R1, R3:



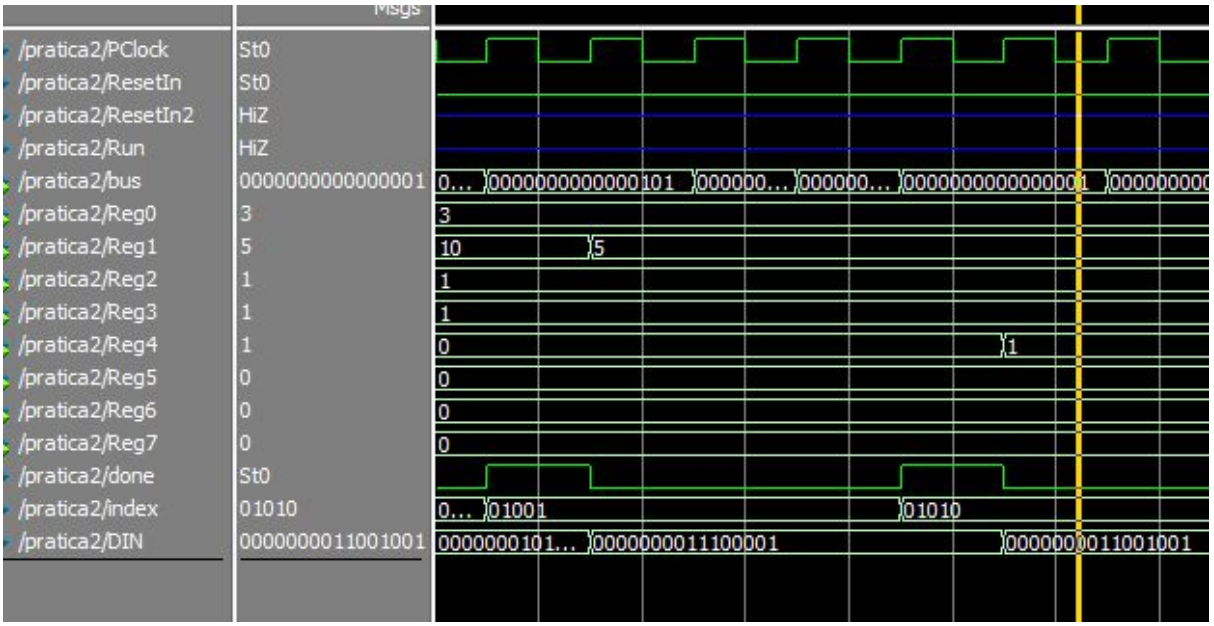
SLL R1, R3:



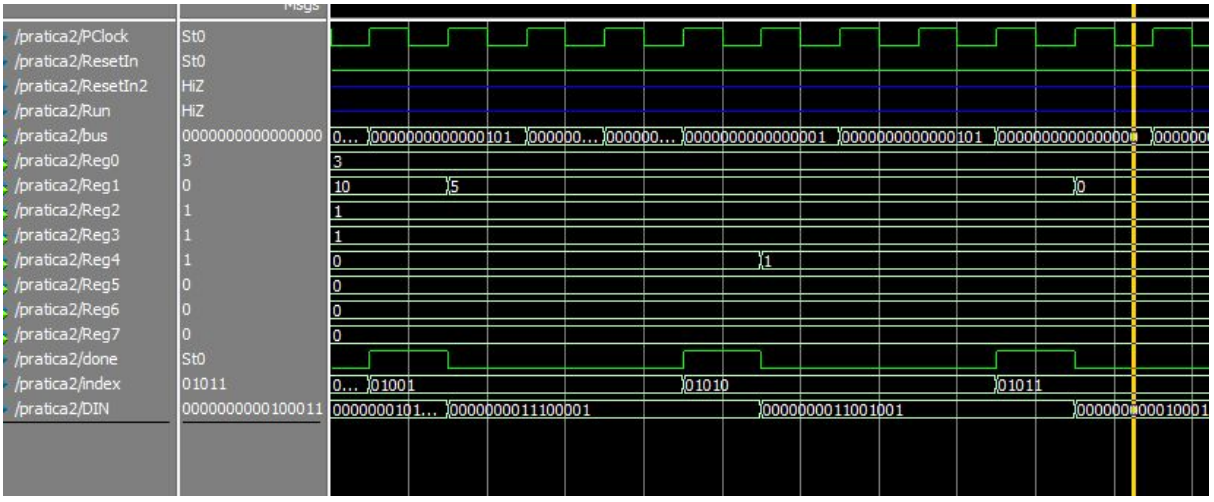
SRL R1, R3:



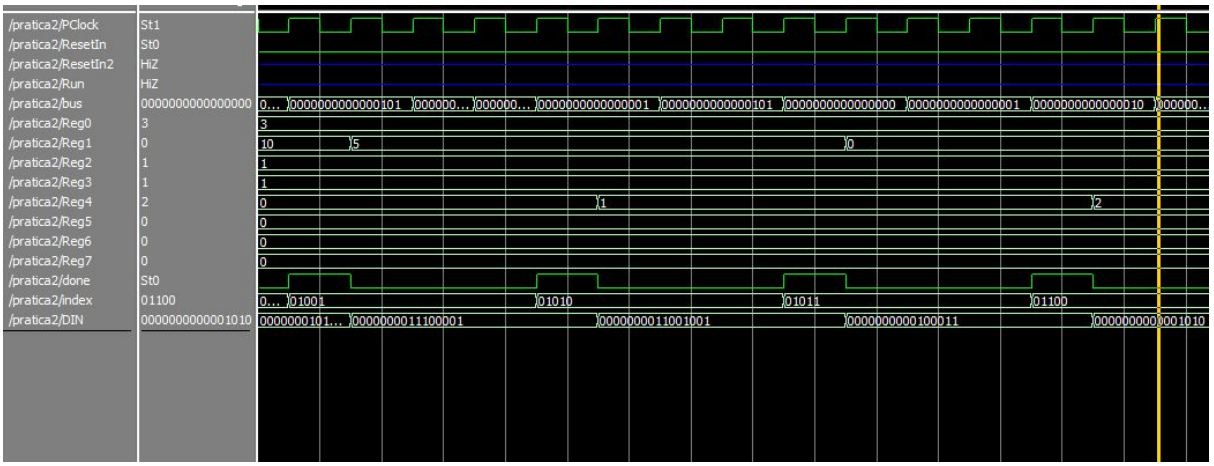
SLT R4, R1:



SLT R1, R1:



ADD R4, R3:



ADD R1, R2: