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: 000000000000001;

OpCode: 000

Registradores utilizados:

Rx = 000;

Rv = 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.

/pratica2/Clock	St1												
/pratica2/DIN	00000000000000001	00000000000000001											
/pratica2/Reset	St0												
/pratica2/BusWires	00000000000001000			00000000	0000011	000000000	0000101	000000000	0001000				
/pratica2/Done	St0												
/pratica2/Counter	001	000	001	010		011		100		000		001	
/pratica2/R0_output	0000000000001000	0000000000000011								000000000	0001000		
/pratica2/R1_output	0000000000000101	0000000000000101											
/pratica2/R2_output	00000000000000001	00000000000000001											
/pratica2/R3_output	00000000000000001	00000000000000001											
/pratica2/R4_output	00000000000000001	00000000000000001											
/pratica2/R5_output	00000000000000001	00000000000000001											
/pratica2/R6_output	00000000000000001	0000000000000001											
/pratica2/R7_output	ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ												
/pratica2/A_output	0000000000000011	00000000000000001				0000000000	0000011						
/pratica2/W_output	222222222222												
/pratica2/RNin	zzzzzzzz00000000	zzzzzzz000000000						zzzzzzzz00	000001	zzzzzzzz00	000000		
/pratica2/IR_output	0000000001			00000000	1								
/pratica2/IRin	00000000001	0000000000	00000000001	(00000000)	00							00000	o
/pratica2/incr_pc	St0								,				
/pratica2/RNout	00000000	00000000		00000001		00000010		00000000					
/pratica2/Ain	St0												
/pratica2/Gin	St0												
/pratica2/Gout	St0												
/pratica2/DINout	St0												

• SUB:

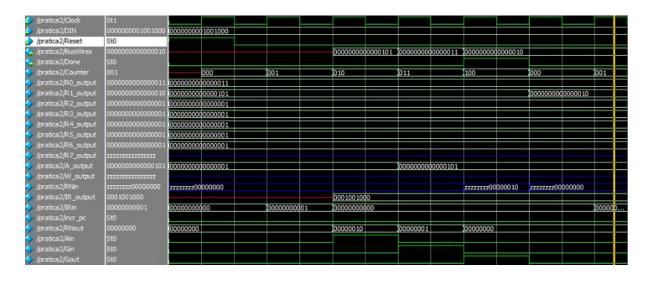
Exemplo de SUB executado: 000000001001000;

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.

/pratica2/Clock	-No Data-					
/pratica2/DIN	-No Data-	0000000110000001				
/pratica2/Reset	-No Data-					
/pratica2/BusWires	-No Data-			00000000000000101		
/pratica2/Done	-No Data-					4
/pratica2/Counter	-No Data-	000	001	(010	011	000
/pratica2/R0_output	-No Data-	000000000000011			00000000000000101	
/pratica2/R1_output	-No Data-	0000000000000101				
/pratica2/R2_output	-No Data-	(00000000000000001				
/pratica2/R3_output	-No Data-	(00000000000000001				
/pratica2/R4_output	-No Data-	(0000000000000001				
/pratica2/R5_output	-No Data-	(00000000000000001				
/pratica2/R6_output	-No Data-	(0000000000000001				
/pratica2/R7_output	-No Data-					
/pratica2/A_output	-No Data-	(00000000000000001				
/pratica2/W_output	-No Data-					
/pratica2/RNin	-No Data-	zzzzzzzz000000000		zzzzzzzz00000001	zzzzzzzz000000000	
/pratica2/IR_output	-No Data-			0110000001		
/pratica2/IRin	-No Data-	0000000000	00000000001	,00000000000		
/pratica2/incr_pc	-No Data-					
/pratica2/RNout	-No Data-	00000000		00000010	00000000	
/pratica2/Ain	-No Data-					
/pratica2/Gin	-No Data-					
/pratica2/Gout	-No Data-					

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:

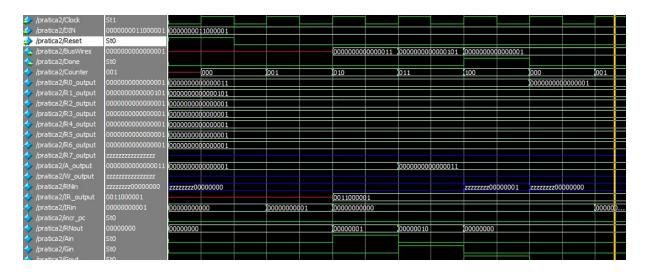
Exemplo de SLT executado: 0000000011000001;

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:

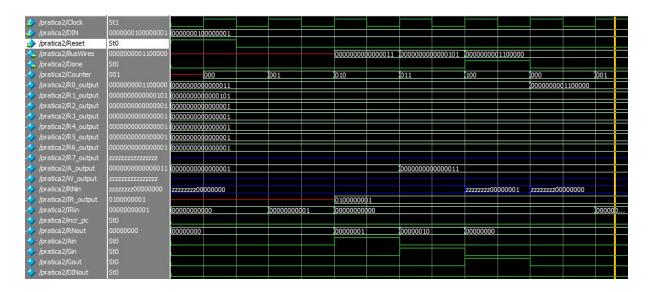
Exemplo de SLL executado: 0000000100000001;

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.

/pratica2/Clock	St0					-					
/pratica2/DIN	0000000101001000	0000000101001000									
/pratica2/Reset	St0										
/pratica2/BusWires	0000000000000101			000000000010	1000 000000000	0000011	000000000	0000101			
/pratica2/Done	St0										
/pratica2/Counter	001	000	001	010	011		100		000		001
/pratica2/R0_output	0000000000000011	0000000000000011									
/pratica2/R1_output	0000000000000101	0000000000101000							000000000	0000101	
/pratica2/R2_output	00000000000000001	00000000000000001									
/pratica2/R3_output	0000000000000001	00000000000000001									
/pratica2/R4_output	00000000000000001	00000000000000001									
/pratica2/R5_output	00000000000000001	0000000000000001									
/pratica2/R6_output	00000000000000001	00000000000000001									
/pratica2/R7_output	ZZZZZZZZZZZZZZZZZZ										
/pratica2/A_output	0000000000101000	00000000000000001			00000000	0101000					
/pratica2/W_output	ZZZZZZZZZZZZZZZZZ										
/pratica2/RNin	zzzzzzzz00000000	zzzzzzzz000000000					zzzzzzz00	000010	zzzzzzz00	000000	
/pratica2/IR_output	0101001000			0101001000							
/pratica2/IRin	00000000001	00000000000	00000000001	,00000000000							000000
/pratica2/incr_pc	St0										
/pratica2/RNout	00000000	00000000		00000010	00000001		00000000				
/pratica2/Ain	St0										
/pratica2/Gin	St0										
/pratica2/Gout	St0										
/pratica2/DINout	St0										

OR:

Exemplo de OR executado: 000000010000001;

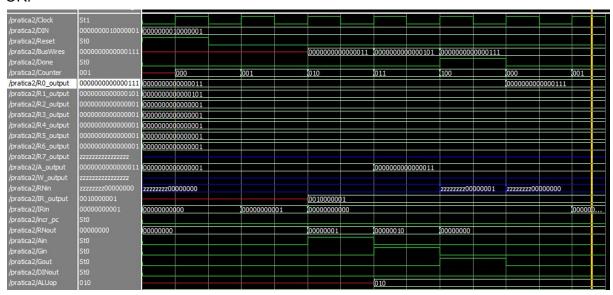
OpCode: 010

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 $Rx \leftarrow [Rx] \mid [Ry]$. 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	А	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

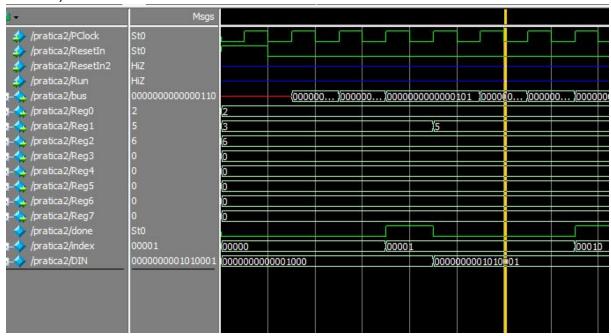
A solução para implementar o nosso processador ligado a uma memória de instruções foi a seguinte: Primeiramente ligamos um clock ao processador. Ligamos a saída done do processador como o clock da memória. Ligamos o dado de saída da memória ao dado de entrada do processador.

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

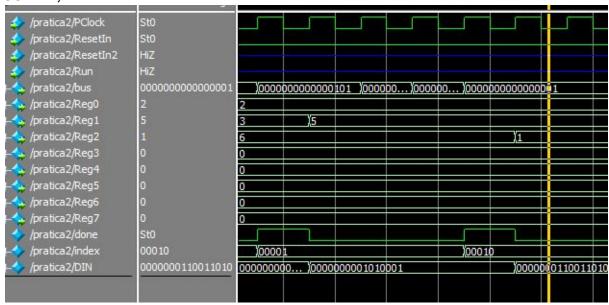
Estado Inicial:

ù÷	Msgs		
/pratica2/PClock	St1		
/pratica2/ResetIn	St0		
/pratica2/ResetIn2	HiZ		
🥠 /pratica2/Run	HiZ		
📆 🚣 /pratica2/bus	0000000000000011	(000	000 (000000 (0000
📭 👍 /pratica2/Reg0	2	2	
🚣 👍 /pratica2/Reg1	3	3	
🛨 🔷 /pratica2/Reg2	6	6	
💶 👍 /pratica2/Reg3	0	0	
∓- ∲ /pratica2/Reg4	0	0	
∓- 4∞ /pratica2/Reg5	0	0	
g -∳ /pratica2/Reg6	0	0	
- /pratica2/Reg7	0	0	
<pre>/pratica2/done</pre>	St0		
- ∕pratica2/index	00000	00000	(0000
	0000000000001000	000000000000001000	

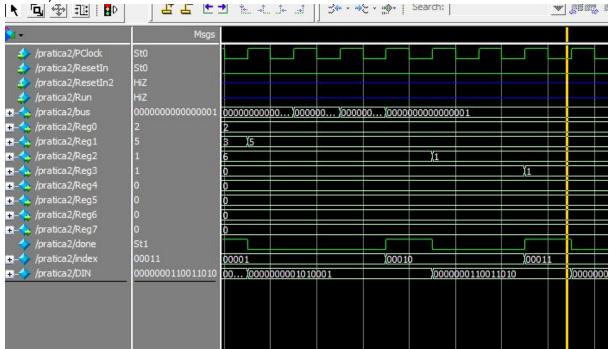
ADD R1, R0:



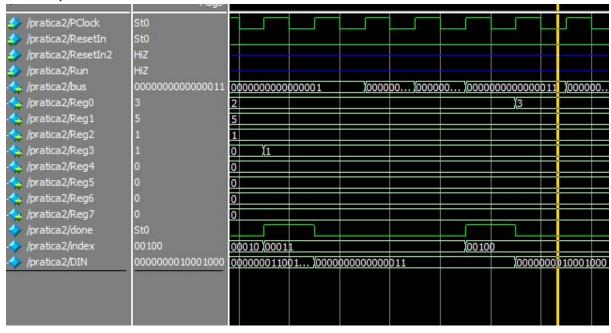
SUB R2, R1:



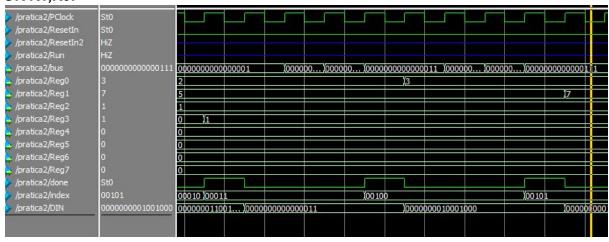
MV R3, R2:



ADD R0, R3:



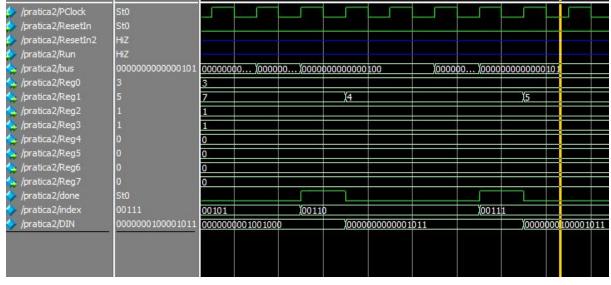
OR R1,R0:



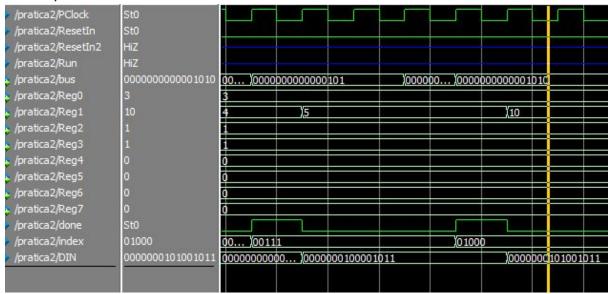
SUB R1,R0:

/pratica2/PClock	St1									
/pratica2/ResetIn	St0									
/pratica2/ResetIn2	HiZ									
/pratica2/Run	HiZ									
/pratica2/bus	00000000000000100	0000 00	0000)0000	000000000	111	(00000	00	000000000	100	
/pratica2/Reg0	3	3								
/pratica2/Reg1	4	5),7				(4		
/pratica2/Reg2	1	1								
/pratica2/Reg3	1	1			į.					
/pratica2/Reg4	0	0								
/pratica2/Reg5	0	0								
/pratica2/Reg6	0	0								
/pratica2/Reg7	0	0								
/pratica2/done	St0									
/pratica2/index	00110	00100	0010	1			0011	0		
/pratica2/DIN	0000000000001011	0000000010	0001000	(0000	00000100	1000		(0000	000000	001

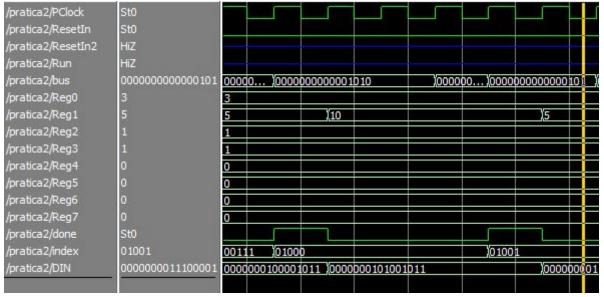
ADD R1, R3:



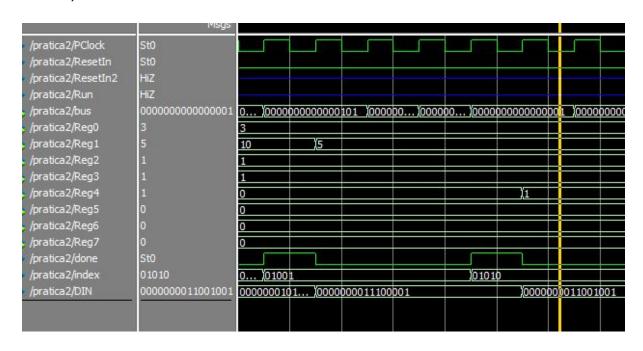
SLL R1, R3:



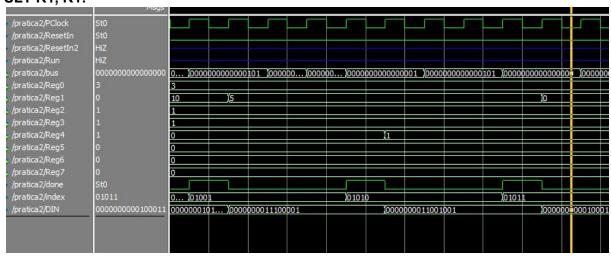
SRL R1, R3:



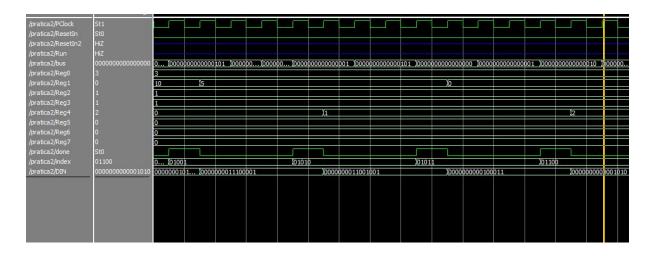
SLT R4, R1:



SLT R1, R1:



ADD R4, R3:



ADD R1, R2:

