ASSIGNMENT 4 LONG NGUYEN EEL4713 3/17/2012

INTRODUCTION

In this lab I worked on designing a single cycle MIPS processor which executed the 29 Core Instructions. Parts from the previous lab 2 were combined with additional hardware in order to create the mips_core file in this lab. A major goal of this lab is to make sure the all the instructions work for the next labs.

DESIGN AND TESTING

The following is the single cycle datapath. A more detailed datapath can be found in the bdf files,



LAB4_TEST

The first instruction test for the processor came from the lab4_test.mif. This test gave me a number of difficulties in when trying to simulate in model sim. As it would turn out, the register file was not functioning properly and became the source of a very long headache since beginning in the week.

With Data Noncode Noncod Noncode Noncode <		00400000	00400000 0040	0001	0040002	00400003	0040004	00400005	100400006
• • • • • • • • • • • • • • • • • • •									
AU or 00 00 00 00 00 00 000 0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000 000000000 00000000 0		0000000	0000 1000000	11000010	1000 1000000	Vananana A	00004554	Ϊοοοοοοο	
o On On </td <td></td> <td></td> <td></td> <td>10000010</td> <td><u>,1000000074</u></td> <td>ACCOCCA</td> <td>20000ASDA</td> <td>,00000000</td> <td></td>				10000010	<u>,1000000074</u>	ACCOCCA	20000ASDA	,00000000	
□ □000000 □0000000 □0000000 □0000000 □0000000 □0000000 □0000000 □0000000 □0000000 □0000000 □0000000 □00000000 □000000000000000000000000000000000000		000		1010		1111	011	V010	1111
Instruction Mem Imit S20, S1000 Imit S20, S10000 Imit S20, S100000 Imit S20, S100000 Imit S20, S1000000 Imit S20, S1000000 Imit S20, S10000000 Imit S20, S100000000000000000000000000000000000					1000 1000000				
c 00000000 000000		0000000			<u>,1000,0000000A</u>		JUUUUASSA	0000000	
nv 0 13 15 16 15 15 16	-					V			· · · · · · · · · · · · · · · · · · ·
Register File Concerned		0000000		23DE0010					000F7C
Image: Conservery Con		0	030]16	<u>]19</u>		<u>15</u>	
c 3(1) 0000000 00000000 00000000 10000000 10000000 0 </td <td>— Register File ———</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	— Register File ———								
1000000000000000000000000000000000000	⊒-🔶 ra	{00000000} {00000000} {00000000} {	{00000000} <u>{</u> 000000}	00000} {10000000}	{00){00000000} {10000010}	{00){00000000} {10000010}	{ <u>00 }{00000000} {10000010} {</u>	00 (00000000) {10000010}	{00000000} {0000000
c (29) 00000000 00000000 next instruction adds 10 to \$30 next instruction a	🛓 🔶 (31)								
T ≤ (28) 00000000 00000000 next instruction adds 10 to \$30 Image: Comparison of the comparison of th	🛓 🔶 (30)	00000000 \$1000 was written to ra(30) y	which is \$fp 1000	0000	10000010				
b (28) 00000000 0000000 0000000 0000000 c (27) 00000000 0000000 0000000 0000000 c (23) 00000000 0000000 0000000 0000000 c (24) 00000000 0000000 0000000 0000000 c (21) 00000000 0000000 0000000 0000000 c (13) 00000000 0000000 0000000 0000000 c (14) 00000000 0000000 0000000 0000000 c (13) 0000000 0000000 0000000 0000000 c (13) 0000000 0000000 0000000 0000000 c (14) 00000000 0000000		0000000	00000000		instruction adds 10 t	o \$20			
b < (26)		0000000	0000000	пех	i mstruction adds 10 t	0 \$30			
c ≤ (25) 00000000 00000000 00000000 00000000 00000000 c ≤ (25) 00000000 00000000 00000000 00000000 00000000 c ≤ (22) 00000000 00000000 00000000 00000000 00000000 c ≤ (21) 00000000 00000000 00000000 00000000 00000000 c ≤ (21) 00000000 00000000 00000000 00000000 00000000 c ≤ (20) 00000000 00000000 00000000 00000000 00000000 c ≤ (13) 00000000 00000000 00000000 00000000 00000000 00000000 c ≤ (13) 00000000 00000000 00000000 00000000 00000000 00000000 00000000 c ≤ (13) 00000000 00000000 00000000 00000000 00000000 00000		0000000	0000000						
b < (24)	🛉 🔶 (26)	0000000	00000000						
b < (24)	🛓 🔶 (25)	0000000	0000000						
• 2(2) 0000000 0000000 0000000 addu adds \$16 to \$19 and both now have the same contents • • • • • • • • • • • • • • • • • • •	🛓 🔶 (24)	0000000	0000000						
c > (21) 0000000 0000000 addu adds \$16 to \$19 and both now have the same contents a < a < a < a < a < a < a < a < a < a <									
c 200 00000000 00000000 addit addit site to site and both now have the same contents c 3 (19) 00000000 00000000 00000000 00000000 c 3 (17) 00000000 00000000 00000000 00000000 00000000 c 4 (17) 00000000 00000000 00000000 00000000 00000000 00000000 c 5 (16) 00000000 00000000 00000000 00000000 00000000 00000000 c 5 (16) 00000000 00000000 00000000 00000000 00000000 00000000 c 5 (13) 00000000 00000000 00000000 00000000 00000000 00000000 c 5 (13) 00000000 00000000 00000000 00000000 00000000 00000000 00000000 c 5 (13) 00000000									
• (20) 00000000 00000000 00000000 00000000 • (19) 00000000 00000000 00000000 00000000 • (17) 00000000 00000000 00000000 00000000 • (17) 00000000 addi \$16,0, \$000A adds \$000A to register 16 0000000A 0000000A • (16) 00000000 addi \$16,0, \$000A adds \$000A to register 16 0000000A 0000000A • (14) 00000000 00000000 00000000 00000000 00000000 • (14) 00000000 00000000 00000000 00000000 00000000 • (12) 00000000 00000000 00000000 00000000 00000000 00000000 • (12) 00000000 00000000 00000000 00000000 00000000 0000000 • (10) 00000000 00000000 00000000 00000000 00000000 00000000 • (10) 00000000 00000000 00000000 00000000 00000000 0000000 • (10) 00000000 00000000 00000000 00000000 00000000 00000000 • (10)		0000000				addu adda \$16 to \$10 o	ad both now have the g	ame contents	
n 0000000 0000000 0000000 0000000 0000000 0000000 0000000 0000000 0000000 0000000 00000000	🛉 🔶 (20)					addu adds \$1818 \$19 a		mecontents	
c < (17) 0000000 add is 16,0, \$0000 add is 0,0, \$0000 A adds \$0000 A to register 16 0000000 A adds \$0000 A adds \$0000 A to register 16 0000000 A adds \$0000 A adds \$0000 A adds \$0000 A to register 16 0000000 A adds \$0000 A adds \$0000 A to register 16 0000000 A adds \$0000 A ad							<u>)0000000A</u>		
•••••••••••••••••••••••••••••									
• • (15) 0000000 • • • • • • • • • • • • • • • • • • •								0000A55A	
c > (15) 00000000 00000000 00000000 c > (14) 00000000 00000000 0	T ()			i \$16,0, \$000A a	adds \$000A to register	16 0000000A	the next ori instru	uction inserts \$4550 ir	nto ra(17)
••• (13) 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000 00000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 00000000000000 0000000000000000	😐 - 🔶 (15)						the next off motif		uoru(11)
0 00000000 0000000 0000000 0000000 0000000 0000000 0000000 0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000 000000000 00000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000000000000 00000000000000000000000000000000 000000000000000000000000000000000000									
0 00000000 00000000 00000000 00000000 00000000 00000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000000000000000000000000000000									
D-> (10) 000000000 000000000 000000000 0000000000 000000000000000000000000000000000000	🕂 🕂 (12)								
■ Now 1327390000 ps 100000 ps 300000 ps 300000 ps	•••• (11)								
■ Now 1327390000 ps 100000 ps 300000 ps 300000 ps	1 0)								
s 10000 ps 20000 ps 30000 ps	📮 - 🔷 (9)	0000000	00000000						
	Now	1327390000 ps	DS		100000 ps	2000	Тталалалар)00 ps	300000 ps	
	Cursor 1	0 ps	0 ps						

Once the datapath was working correctly again, this simulation was successfully completed.

🖅 🔶 PCin		0040000C	00400	008	(004000	09)00400	00A	004	0000B		004	0000C		00400	00D	
■→ PCout		0040000B	00400	007	004000	008	00400	009	004	0000A		004	0000B		00400	00C	
		0000000	00000000	0000FEED		00008000		8000000	000	0000		İ	000000	1		008000	0
		111	111	011		2000		1111									
> aluout		0000000		0000FEED		00008000		80000000	Ĭooo	0000		- i	000000	1		0000800	0
	on Mem ——											Ī					
∎-⇒ q		01F0902A	00057040	35EFFEED		31F08000		00108400		01F0902			01F0902	R		00127BC	0
= v ч ∎-⇒ rw		18	15			16				18		- 1	10110302			115	•
- Register	File	10	13	<u></u>		<u>,10</u>										<u></u>	
=> ra	File	{00000000} {10000010} {00000000} {00000000} {0	(0000000	0} {10000010} {000		0003 (100001	0) (00) (0000		00 100	000003 (10	0000103 (000	00000	1 (0000000	1 (000000	100000	0000 (10)	000103 (0
H - (31		00000000 (100000103 (00000003 (00000003 (0	00000000		JU	00071000001	07 100 <u>X</u> 0000	10000107	-100 <u>/</u> 300	000007110	00001033000	100000	710000000	1710000000	100000	10007 1100	
		10000010	10000010									-					
		00000000	00000000									-					
		0000000	0000000									1					
. (27		0000000	0000000									i					
1 - (26		0000000	0000000									ĺ					
. (25		0000000	0000000									1					
😐 🔶 (24		0000000	0000000									1					
📃 😐 🔶 (23		0000000	0000000														
💷 🔶 (22		0000000	0000000									ļ					
😐 🔶 (21		0000000	0000000														
• - > (20		0000000	0000000														
1))	000000A	000000A		a (15) mit	h itself to	t	ne next instr	uction a	nds xFEI	ED with x	8000					
18		0000000	0000000		. ,	n itsen to									00000	001	
17		0000A55A	0000A55A		t xFEED		V		V			_			1 . 6 1	1 1 4 1	
1		8000000	0000000A		Vagaar		00008	000	,800	0000		-		and	shifted	right by	74
		0000FEED 00000000	00000000		<u>0000FE</u>	ED											
	+) >\	0000000	00000000							bits are	then shift	edle	eft by 4				
	2)	0000000	00000000							o first to	wo shifts	are d	onesim	od ubile	theles	t ie une	ioned
• • • • (12	-)	00000000	00000000							ie mrst t	so sintes	areu	one sign	eu winne	the las	t 15 UIIS	Igned.
	Now	1337035000 ps	A DOM: N		a bara				1.1.1.1			1.1	i i biri		1.1.1.1		1.1.1.1
0.40			40000	0 ps		5	600000 ps			60000	00 ps				70000) ps	
in / 😒	Cursor 1	636444 ns										63644	4 ncl				

<mark>€</mark> 1×	Msgs															
	0040000B	0040000E	00400	00F	0040	0012		00400	0018	0040	001C	00	410013		00410014	
	0000000	0008000	0000.	. 00000000		000000	00		00000014	20000		4			000	0A
— ALU —																
■-◆ ALUop	111	111		001		000			010		001		111			
	00000000	00008000		. 00000000		1000000	00		00000014			1				10.4
- Instruction Mem		00008000		.,00000000		0000000				,0000		·				0
	01F0902A	00144042		11F40002		081000	10		2673FFFF		1660FFF	e	0260282		022	102
		0014A042					18					•		0		.02
	18	20		20		<u>16</u>			19		<u>ľo</u>		5			
Register File															V-	
. ⊟- ∲ ra			<u>{]{0000</u>	<u>0000} {10000010} ·</u>	<u>{00000000}</u>	{0000000	<u>0} {000000</u>	003 {0000	<u> {00000} {0000000} </u>	<u> (00]{(000</u>	00000} {100	<u>)00010} {000000</u>	<u>00000003 {00</u>	<u>} {0000000</u>	<u>}{00000000}</u>	} {
😐 - 🔶 (31)	0000000	0000000														
⊡ -∲ (30)	10000010	10000010														
😐 🔶 (29)	0000000	0000000											_			
⊡ - ◇ (28)	0000000	0000000														
⊡ - → (27)	0000000	0000000														
(26)	0000000	0000000														
⊡ -→ (25)	0000000	0000000														
(24)	0000000	0000000														
	0000000	0000000		(20) = ra(15) t	hua hua	ahing	o x00.40	0010								
	00000000	00000000	ra	(20) = Fa(15) t	nus brai	iening	10 X0040	0012		a	lds SFFF	Ftora(16)				
u → (21) u → (20)	00000000	00000000	00008	000								1 0010(10)				
■-→ (20) ■-→ (19)	00000000 0000000A	00010000 0000000A	,00008	000						0000	0014					
□-→ (19) □-→ (18)	00000000	0000000A								,0000	0014					
(18) +	00000455A	00000001											+			
(17) +	80000000	80000000											+			
(10)	0000FEED	00008000														
± - (14)	00000000	00000000														
(14)	00000000	00000000														
(13)	00000000	00000000														
	0000000	00000000														
	0000000	00000000														
	0000000	00000000														
≜≣⊛ Now	- 1337035000 ps	in the co				1.1.1.1				1.1.1.1				a la cara		
				900000 ps				100000	00 ps			1100000 ps			12	20000
A 20 Cursor 1	636444 ns															_

Unfortunately it seems the branch functions did not properly branch as expected. For instruction x11F0002, the signals should've branched back to x00400010. In this figure the branch jumped too far ahead. Due to time constraints, a throughout troubleshooting was not performed. However, there could be multiple reasons for this failure. Firstly, I would check the controller to make sure the BEQ and BNE opcodes were correctly initialized. If not that, I would check connections in the bdf and begin checking the hardware.

What this instruction should've continued to do would be to add further numbers into ra(16) and ra(17). Then the next instruction would be a jump and if correctly initialized the jump would take the PC to 0040001B. There is a BNE after the jump to branch back to 00400013. The instructions would continue to test the jump register function and move on to load word. The load word function should be able to read from the data memory into the register. Continuing, the program ends with a BNE which depending on the value would loop through the instructions more.

LAB4_DEMO

The lab4_demo file tests many of the instructions further. As before, the branch functions were not able to be fixed just yet. The simulation up to that section is as follows:

1 (19)	10000000	0000000						
±	0000000	0000000						
± - (17)	0000000	0000000						
1 → (16)	0000000	0000000						
u - → (15)	0000000	0000000						
• - (14)	0000000	0000000						
- (13)	0000000	0000000						
• - > (12)	0000000	0000000						
∳ - ◇ (11)	0000000	0000000						
—	0000000	0000000						
■ - → (9)	0000000	0000000						
↓ -→ (8)	0000000	0000000						
■ - ◇ (7) ■ - ◇ (6)	0000000	0000000	first three OI	RI instructions writes 4,	2 and 5 to the			
	0000000				s, and s to the			
• (5)	0000000		ective registe	ers. (1,2, and 3)				56780000
■	0000000	00000000			0000005			,56780000
■-→ (3) ■-→ (2)	00000000	00000000		0000003				
· (2) • · ◆ (1)	00000000		00004	0000003				
⊥ -→ (0)	00000000	00000000	100001					
	0	0 11	12	<u> </u>	130		<u>)</u> 4	
Program Counter —								
-> PC INC	00400001	00400001 004	100002	100400003	00400004	100400005	100400006	00400007
-> PCin	00400001		100002	00400003	00400004	00400005	X00400006	(00400007
- PCout			00002	00400002	00400004	00400003	00400005	00400007
	00400000	00400000 004	100001	0040002	,0040003		00400005	00400006
Write Data			V		V			
-🔷 output	0000000	0000)00000004	,00000003	0000005	0FFF0000	0FFFFFC	56780000	56784B
- ALU								
🔶 ALUop	000	000) 011						
🔶 aluout	0000000	0000)00000004	0000003	0000005	0FFF0000	0FFFFFC	56780000	56784B
💠 ALUsrc	0							
 Instruction Mem — 								
-🔶 q	0000000	000034010004	34020003	(34030005	J3C1E0FFF	37DEFFFC	3C045678	34844B
Data Mem								
							· · · · · · · · · · · · · ·	
Now		15		100000 ps	200	0000 ps	300000 ps	
Cursor 1	0 p:	5 <mark>0 ps</mark>						
)								•
to 364544 ps								
y renered as	Į							

🔷 RST	1		¥-								¥-		
💎 ra	{00000000} {00000000} {00000000} {00000000		<u>00}{00000000} {</u>	<u>{00000000} {00}</u>	000000000000000000000000000000000000000)FFF0000} {00}	<u>{00000000} {(</u>	FFFFFC} {	<u>00}{00000</u>	0000}{0FFFFFFC}	<u>{00}{0000000</u>	0} {0FFFFFFC} {0	0000000} {00
⊡ -∲ (31)	0000000	00000000											
⊡ -∲ (30)	0000000	00000000			FFF0000)	0FFFFFFC						
± - > (29)	0000000	00000000	Lui 30	\$0FFF loads	OFFF to r	a(30) then or	i 30 30 SF	FFC chan	ores it to	XFFFC			
± - (28)	0000000	00000000	Lui oo	,φ 0111 IOaus φ	0111 1011		1 00,00,91	er o enun	iges it to	AITIC			
⊡ - ◇ (27)	0000000	00000000			_								
+ - (26)	0000000	00000000			_								
± - > (25)	0000000	00000000			_								
• - • (24)	0000000	00000000			_								
± - > (23)	0000000	00000000											
- (22)	0000000	00000000											
⊡ -∲ (21)	0000000	00000000											
⊡ -∲ (20)	0000000	0000000											
⊡ -∲ (19)	0000000	0000000											
+ - (18)	0000000	00000000											
• - > (17)	0000000	0000000											
• - ? (16)	0000000	0000000											
😐 🔶 (15)	0000000	0000000											
😐 🔶 (14)	0000000	0000000											
😐 🔷 (13)	0000000	0000000											
😐 - 🔷 (12)	0000000	00000000											
😐 🔶 (11)	0000000	0000000											
+ - (10)	0000000	0000000											
⊡ -◆ (9)	0000000	0000000											
!! - ? (8)	0000000	00000000											
😐 🔶 (7)	0000000	0000000											
😐 🔶 (6)	0000000	00000000					1	78 loads	no(4) mi	th \$5678			
🛨 🔷 (5)	0000000	0000000					iui 4,500	78 Ioaus					
⊡ -→ (4)	0000000	0000000							567800		56784BCD		
😐 🔶 (3)	0000000	0000000	00000005							tl	nen, ORI 4,4,	\$4BCD is OR	led to ra(4)
• - > (2)	0000000	0000003											
⊡ -→ (1)	0000000	00000004											
⊡ -� (0)	0000000	0000000											
🔶 rw	0	3	30				4)5		
🖲 Now	892440000 ps			200000 ps			300	l і і і і 000 ps			400000 ps		
6A 0	^												

🔷 RST	1	Ŭ																		
🖃 🔶 ra	{0000000} {000000	00} {00000000} {00000000} {0	{0000000} {	0F	00000} {0F	FFFFFC} {	00}{000	00000} {1	0000003 {	00 {000	00000} {1	000000} {	00000000}	{0000000)} {000000}	0 {{00000	0000} {10	000000} {	00 {{0000	0000}
1 -🔷 (31)	00000000		0000000																	
. (30)	0000000		0FFFFFFC				1000	0000												
. (29)	0000000		0000000																	
. (28)	0000000		0000000				🗖 addi a	dds \$4	tora(30) yieldi	ng x100	0								
😐 - 🔷 (27)	0000000		0000000																	
🛓 🔷 (26)	0000000		0000000																	
🛓 🔷 (25)	0000000		0000000																	
😐 🔷 (24)	0000000		0000000																	
😐 🔷 (23)	0000000		0000000																	
😐 🔶 (22)	0000000		0000000																	
🛓 🔶 (21)	0000000		0000000																	
🖣 🔶 (20)	0000000		0000000																	
😐 - 🔶 (19)	0000000		0000000																	
± - > (18)	0000000		0000000																	
😐 🔶 (17)	0000000		0000000																	
😐 🔶 (16)	0000000		0000000																	
😐 - 🔶 (15)	0000000		0000000																	
• (14)	0000000		0000000																	
1 - (13)	0000000		0000000																	
± - ? (12)	0000000		0000000																	
<u>∎</u> -♦ (11)	0000000		0000000																	
± - > (10)	0000000		0000000																	
• - > (9)	00000000 00000000		00000000					()			-) I		(-) 0							
(8)	0000000		00000000 r	a(4) is t	hen add	led to r	a(5)	ra(4)	is addec	i to ra(5) and s	aved in	ra(6) fo	r add 6,	4,5					_
∎-� (7) ∎-� (6)	00000000		0000000		the con					ÍACE.	979A									
 	00000000		00000000	5678		come o				JACF	1979M					00000	000			
± ↔ (3)	00000000		56784BCD	<u></u>	HOCD											100000	000			
• • • (•) • • • (3)	00000000		00000005																	
(3)	00000000		00000003																	
😐 - 🔶 (1)	0000000		00000004																100000	003
$\mathbf{\underline{I}} \rightarrow (1)$	00000000		00000000																	
	0		5		30			<u>)</u> 6			5						<u>)</u> 1			0
1. 1 1. 1	Now	892440000 ps	5000			1.1.1			00 ps		i i la		7000			1.1.1.1		80000		1.1.1
a 20	Cursor 1	0 ps		oo ps				0000	oo ps				7000	o ps				80000	io ps	
		0 ps																		

Branch here 0 begins after this instruction . The loop should decrement ra(1) until it is 0. Here 1 does the same for ra(2). Here 2 should also decrement ra(3), however, the malfunction begins to throw the instructions off.

*	Msgs													
🛓 🔶 (23)	0000000	00000000												
😐 - 🔶 (22)	0000000	00000000												
🛓 🔶 (21)	0000000	00000000												
🛓 🔶 (20)	0000000	00000000												
+ - > (19)	0000000	0000000												
- (18)	0000000	00000000												
😐 - 🔶 (17)	0000000	00000000												
🛓 🔶 (16)	0000000	0000000												
😐 🔶 (15)	0000000	0000000												
😐 🔶 (14)	0000000	0000000												
🛓 - 🔷 (13)		0000000												
🛓 - 🔶 (12)	0000000	0000000												
😐 - 🔶 (11)	0000000	00000000												
🛓 🔶 (10)	0000000	0000000												
• (9)	0000000	0000000									addi 6	6 \$1111	addsto	ra(6)
📮 - 🔷 (8)	0000000	00000000									auui o	,0,91111	auusto	1 a(0)
😟	0000000	0000000												
••••••••••••••••••••••••••••••••••••••	0000000	ACF0979A										ACFO	A8AB	
₫-� (5)	0000000	56784BCD	000000	0										
😟 - 🔶 (4)	0000000	56784BCD												
🛓 🔶 (3)	0000000	0000005			he	re0 begins to	decrem	ent ra(1)						
<u> </u> 	0000000	0000003				Ĭ			0000	0002 hei	rel will	branch	until ra	(2) is 0
₽ -◇ (1)		00000001			0000	0000								
⊡- → (0)	0000000	0000000												
	0	5	X	1		<u>jo</u>		<u>)</u> 2) <mark>6</mark>			Ĭ0	
— Instruction Mem ——														
	0000000	AFC 8FC50000		2021FFFF		1420FFF9		2042FFFF		20C61111			1440FFFD	
— Program Counter —														
	00400001	0043000D	0043000)F	0043	DOOF	10043	30010	0043	0011		00430	012	Y
	00400001	0043000D	0043000			10043000F		30010	0043			10043	.)0044000F	ř
■ → PCout	00400000	0043000C	0043000		0043			3000F	0043			00430		
	00400000	00430000	,0043000		,0043	JUUE		3000F	,0043	0010			011	\\\
Write Data								V					V	
	0000000	5678)00000000)00000000		0000002	,0000)ACF0A8A	B	,ACF	<u> 100000002</u>	
— ALU ———														
see Now	992445000 ps										1.1.1.1	1.1.1.1		
		20000	100 ps			2100000 ps			22000	00 ps				2300000 ps
Set Cursor 1	0 ps													

😐 🔷 (26)	0000000	0000000							i I						
• - (25)	0000000	0000000												کر کر کار	
1 - (24)	0000000	0000000													
😐 🔶 (23)	0000000	0000000													
• - > (22)	0000000	0000000													
e - (21)	0000000	0000000													
⊡ -� (20)	0000000	0000000													
🛨 - 🔷 (19)		0000000													
•••• (18)	0000000	0000000													
↓ -◆ (17)	0000000	0000000													
🕂 🔶 (16)	59E1569A	59E1569A													
🕂 🔶 (15)	0000000	0000000													
😐 - 🔶 (14)	0000000	0000000													
• (13)	0000000	0000000													
•	0000000	0000000													
🛓 🔶 (11)	0000000	0000000		41 1	1	1. 16				1.0					
🕂	0000000	0000000					nction, ra(2	,							
🛓 🔶 (9)	0000000	0000000		However, o	ther regist	ers are beir	ng altered th	at should	not	be.					
•	0000000	0000000													
😐 🔶 (7)		0000000													
■ (6) ■ (5)	ACF0CACD	ACF0CACD		ACF097)ac)acf(D					
🛓 - 🔷 (5)	0000000	0000000			. 567800	<u>)</u> 5678 <u>)</u> 00 <u>)</u> 5	678)00000000								
+ - (4)	56784BCD	56784BCD		56784BCD				,	Ļ						
u (1) u	0000004	0000000	(000000					ľ	b00.	<u>)0000)000</u>	<u>))0000</u>	<u>)000000</u>	00		
± 🔷 (2)	0000000	0000000	0000003					. <u>)</u> 00 <u>)</u> 00000							
•	0000000	59E1959C	00000004	<u>)oo</u>	000003 000	<u>)00002 (0000</u>	<u>0001 (0000000</u>	0	59E1	1959A			59E1959C		
😟 - 🔶 (0)	0000000	0000000													
≞- 今 rw	1	0	<u>(_(_(30)</u> 4	5 1 15 1 1		1115_1111	105-100000		μuμ						
— Instruction Mem ——															
⊥- 🔶 q	00C60820	0000000			0000000				<u>iot</u>				100000000	0000000	
- Program Counter															
	00450114				mmm	rhanna		TTTTT	m		1 mm	rrr			
Interpretent of the second	00450114			מבררררר			יוררחר								
	00450113	γγγγγγγγγγγ			┟╌┎╌┎╺┎	┎┎┎┎┎┎┎╓╓╓	┙┷┷┷┷┉┷┷┉┉						╵╵╵╵		
■ Vrite Data	00430113														
	000445000														
Now	992445000 ps	000000 ps	16000	000 ps	170	00000 ps	180	00000 ps			0000 ps		2000	0000 ps	
a∠ Q Cursor 1	18375375 ps							18375	5375 p	os					

LAB4_INSTTEST

This test is supposed to demonstrate the functionality of all 29 instructions. However, due to time constraints, I was only able to do some, and hope to troubleshoot the rest after this lab is turned in.

🖕 - 🔶 (22)	0000000	00000000																
u - ↔ (21)	0000000	00000000																
—	0000000	00000000																
🛓 🔶 (19)	0000000	00000000																
🛓 🔶 (18)	0000000	00000000																
🛓 - 🔶 (17)	0000000	00000000																
🛓 🔶 (16)	0000000	00000000																
u - ↔ (15)	0000000	00000000																
	0000000	00000000																
🕂 🔶 (13)	0000000	00000000																
• (12)	0000000	00000000																
(11)	0000000	00000000																
• - > (10)	0000000	00000000																
• - (9)	0000000	00000000																
(8)	0000000	00000000																
🛓 - 🔶 (7)	0000000	00000000																
± - ◇ (6)	0000000	00000000	The f	irst 2 a	ddi instruc	tions add	values to	the 📕	1	he outp	outs of t	he test (code ca	n 🗾				
⊡ -→ (5)	0000000	00000000			be used by				b	e seen i	n the W	rite Dat	ta divid	er 🗖				
+ - > (4)	0000000	00000000	regis		be used by	iui thei ms	uction	.5										
↓ -◆ (3)	0000000	00000000																
■-→ (2) ■-→ (1)	0000000	00000000				00000008												
⊡ -◇ (1)	0000000	00000000	0000	0002														
• - > (0)	0000000	00000000																
	0	0 1		2		3			4			2),o
₽- ♦ a	0000000	0000 20	010002	200200	0.8	2403	FFF		3064FF	F		102200	1F		142200	F		20000000
- Program Counter -		COCONING LO	010002	LOOLOO		n= 1001			1,500111						ALIEL00			
E-> PC_INC	00400001	00400001	0040	0002		00400003		0040	0004		0040	0005		00400	1006		0040	0026
		_										004000			004000			
	00400001	00400001				00400003		0040					05		-	-5		
■→ PCout	00400000	00400000	<u>)0040</u>	0001		<u>)00400002</u>		0040	0003		0040	0004		0040	005		0040	0025
— Write Data ———																		
	0000000	0000)00	000002	000000	08	00000	000					FFFFFF	FA					0000000
— ALU ———																		
▪→ ALUop	000	000 💭 01	0						000			001						111
≗≣⊛ Now	1465740000 ps	1 1 1				100000 ps	1.1.1.1		1.1.1	2000	00.05				3000	00.05		and a second
⊖ / 9 Cursor 1	0 ps	0.00				100000 ps				20000	00 p3				5000	00 ps		
Cur5011	0 0 05	no osi																

The current functionality of the datapath is limited due to the failure to implement several instructions. Work will continue on adding the necessary hardware to perform the last instructions needed before the next assignment.

TIMING SIMULATION SPEED

Timing analysis of the processor was performed in Quartus. For some reason, the aluctrl was clocked as well.

Slov	v Model Fmax	Summary		
	Fmax	Restricted Fmax	Clock Name	Note
1	31.07 MHz	31.07 MHz	clk	
2	31.66 MHz	31.66 MHz	mclk	
3	64.61 MHz	64.61 MHz	alu32control:aluctrl control[0]	
4	334.45 MHz	225.43 MHz	$Instruction {\tt Mem: instmem altsyncram: altsyt altsyncram_b971: auto_generated q_a[0]}$	limit due to hold check
5	417.71 MHz	120.34 MHz	InstructionMem:instmem altsyncram:altsy altsyncram_b971:auto_generated q_a[26]	limit due to hold check

CONCLUSION

Without a properly functioning datapath, I am unable to satisfactorily conclude this assignment. The datapath is however very close to being finished and troubleshooting should not take too long, though it is longer than I have to turn this paper in. All VHDL and BDF files are included in the submission.