Lab 4 Report

Single Cycle Design

BAOTUNG C. TRAN

EEL4713C

Added Hardware :

Andi and Ori : For this instruction, I had to add a zero extender into my design. Which therefore required me to add a mux that controlled whether the immediate value from the instruction was zero extended or sign extended.

BNE and BEQ : These instructions required me quite a bit of extra logic. The problem was that before, we were using the zero signal from the alu to determine whether to branch or not. But the problem was, there was no way to differentiate between BNE and BEQ. Therefore I added two and gates and an or gate, and also added two more control signals called BEQ and BNE to the controller to differentiate when they would branch.

Jump and Jump and Link : For these instructions I added a mux to the address of the PC counter. I also added a control signal named JJAL, which controlled the mux. When it was set to 1, it would take the value of the jump address instead of just PC+4. For jump and link, I added an additional mux at the "Write Register" port of the register file, which was also controlled by JJAL. When JJAL was true, the mux would output 11111, which would select register (31) and then write the PC+4 to that address.

Jump Register : For this instruction I added a mux after the Jump and Link mux. The mux select was controlled by a signal JR. The JR signal came from the alucontrol. This is because JR is an r-type instruction meaning, the control signal could not come from the main controller. When JR is true, the mux would switch to inputting R[rs] into the PC.

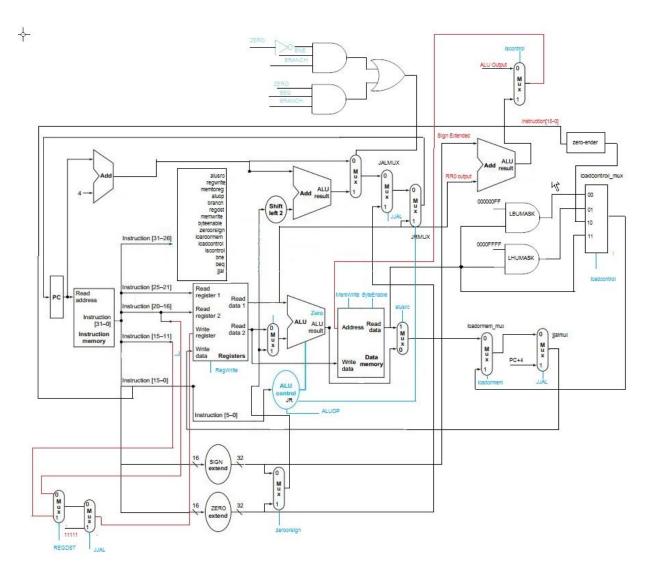
LBU, LHU, LUI, LW : These instructions were quite tricky. I firstly had to use and gates to mask M[R[rs]+SignExtImm for LBU and LHU. I then added a four-to-two mux to choose between LBU, LHU, LUI, and LW. This would go into a mux called loadormem_mux. This mux would then choose whether to output what came from the alusrc_mux(chooses between data memory or alu output) or one of the load instruction values to write to the register file.

SLL and SRL : Since these were R-Type instructions, I had to add a shdir signal to the alucontrol to tell them which way to shift.

SB, SW, SH : For these instructions, I added an adder that summed the values of the SignExtImm + R[rs]. This value would then go into a mux that took the inputs of SignExtImm+R[rs] and the ALU output. This mux was controlled by "Iscontrol" which I added to the controller. By use the ByteEnable signal on the data memory, I could control to either write a byte, half word, or word.

**** My current bdf and vhd file increment the pc by just 1. During my own test file named "lab4baotung_test.mif", I incremented the pc counter by +4.

Below is the diagram of the previous hardware and the hardware that I had to include to create the entire single cycle processor. To start my Instruction Memory at x00400000, I manually my register file to reset to x00400000.



Control Signals Added:

Zeroorsign : Controls whether immediate value is zero-ext or sign-ext. I had to use this for andi, sltiu, addiu and ori, as these instructions were zero extended. If it is 1, it is zero extended, if its 0, it is sign extended

Loadormem : Chooses whether to output whats coming from the alusrc mux or, either half-word, byte, word, or immediate value that is zero extended(lui).

Loadcontrol : Select line of mux that chooses between half-word, byte, word, or upper immediate.

Lscontrol : Select line of mux that chooses either the output of the ALU or R[rs] + SignExtImm to go into address of data memory.

BNE and BEQ : Signals used to differentiate when its BEQ and BNE.

JJAL : Signal used to control two mux select lines when using the Jump or Jump and Link instruction.

JR: Placed in alucontrol since its an R-Type instruction. Used to control mux select line that goes into the "Write Register" input of register file. When it's set to 1, it will automatically set "Write Register" to 11111 or register(31).

Control Signals For New Instructions :

The complete values of the signals can be found in my controller and alu32control, which is located at the end of the report.

The following are ONLY the signals that were configured for the new instructions which required the new control signals.

These signals are directly from my controller:

```
when "001001" => -- addiu
zeroorsign <= '1';</pre>
aluop <= "001";
regwrite <= '1';
regdst <= '0';</pre>
ALUSRC <= '1';
when "001100" => -- andi
zeroorsign <= '1';</pre>
aluop <= "101";
regwrite <= '1';
regdst \leq 0';
ALUSRC <= '1';
when "000100" => --BEQ
branch <= '1';</pre>
BEQ <= '1';
aluop <= "110"; -- subtraction
when "000101" => --BNE
branch <= '1';
BNE <= '1';
aluop <= "110";
when "001101" => -- ORI
zeroorsign <= '1';</pre>
aluop <= "010";
regwrite <= '1';
regdst <= '0'
```

```
when "001011" => -- sltiu
zeroorsign <= '1';
aluop <= "100"; -- tells alu control its an i instruction
regwrite <='1'; -- enables register file to write
regdst <='0'; -- writes to rt
alusrc <='1'; -- takes in immediate value taht is sign extended.
memtoreg <= '0'; --takes value, either 0 or 1 from aluout, and then will put it into rt.
when "100100" => -- lbu
aluop <= "001"; -- immediate instruction
memtoreg <= '1';
alusrc <= '1';
regwrite <='1';
loadcontrol <= "00"; --chooses mask lbu
loadormem <= '1';</pre>
|| scontrol <= '1';
when "100101" => --lhu
aluop <= "001"; -- immediate instruction
memtoreg <= '1';
alusrc <= '1';
regwrite <='1';
loadcontrol <= "01"; -- chooses mask lhu
loadormem <= '1';</pre>
lscontrol <= '1';</pre>
when "100011" => --lw
aluop <= "001"; --immediate instruction
memtoreg \leq 1';
alusrc <= '1';
regwrite <='1';
loadcontrol <= "11"; --choses entire rs+signext
loadormem <= '1';</pre>
lscontrol <= '1';</pre>
when "101000" => --sb
 BYTEENABLE <= "0001";
 aluop <= "001"; --immediate instruction
 memtoreg <= '1';
 aluop <= "001";
 lscontrol <='1';</pre>
```

```
memwrite <= '1';
```

```
when "101001" => --sh
 BYTEENABLE <= "0011";
 aluop <= "001"; --immediate instruction
 memtoreg \leq 1';
 aluop <= "001";
 lscontrol <='1';</pre>
 memwrite <= '1';
 when "101011" => --sW
 BYTEENABLE <= "1111";
 aluop <= "001"; --immediate instruction
 memtoreg <= '1';
 aluop <= "001";
 lscontrol <='1';</pre>
 memwrite <= '1';
when "001111" => --lui
 aluop <="001";
 regwrite <= '1';</pre>
 loadcontrol <="10";
 loadormem <= '1';</pre>
 lscontrol <='1';</pre>
when "001111" => --lui
 aluop <="001";
 regwrite <= '1';</pre>
 loadcontrol <="10";
 loadormem <= '1';
 lscontrol <='1';</pre>
when "000010" => --jump
 jjal <= '1';
when "000011" => --jal
 jjal <= '1';
 regwrite <= '1';
```

There were also some R-Type instructions which I had to change. Since the instruction is thrown to the alu32control, I had to change some instructions there as well. They will be listed below. SRL, SLL, and JR are all R-Type instructions which required extra signals such as shdir for the shift direction and JR for the JR instruction.

elsif (func = "000010") then - **SRL** shdir <= '0'; control <= "0011"; -- srl control

elsif (func = "000000") then -- sll control <= "0011"; --sll control shdir <= '1';

elsif (func ="001000") then - jr control <= "1110"; jr <= '1';

Simulation Testing:

Disclaimer : I will only annotate each instruction ATLEAST once. If I have annotated it already in a previous slide, I might not annotate it again to save space and from you reading redundant information.

Lab4_demo.mif

000:04000134;ori1,0,\$4001:03000234;ori2,0,\$3002:05000334;ori3,0,\$5003:FF0F1E3C;lui30,\$0F004:FCFFDE37;ori30,30,005:7856043C;lui4,\$567006:CD4B8434;ori4,4,\$4	FF \$FFFC 8
--	-------------------

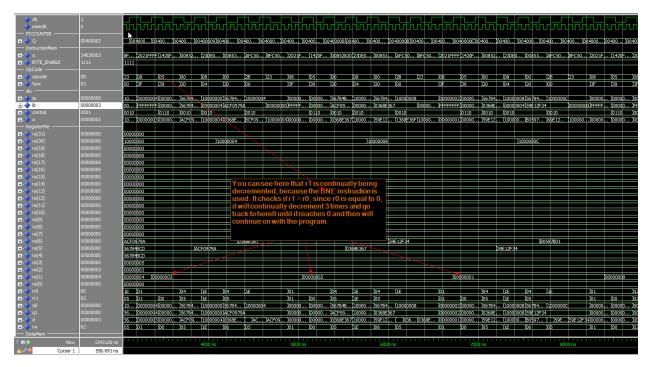
📣 dk	0									
Vnemclk	1						╧┥┍┶┍┱		╪╧┍┑┍┕┶╪	
	<u> </u>								\square \square \square	
	00400006	00400000	00400001	004000	02 1004	00003	00400004	00400005	00400006	00400007
- InstructionMem		-	200100001	2004000	<u> 400 -</u>	00003	200100001	200-100000	100400000	,00100007
	34844BCD	0 3401000)4)3402	0003 /3	1030005	3C1E0FFF	37DEFFFC)3C0456	78),34844	CD (30050000
	1111	1111	<u>, , , , , , , , , , , , , , , , , , , </u>	<u>, , , , , , , , , , , , , , , , , , , </u>		00120111				
DataMem										
+	CD	00 04	03	10	5)FF)FC)78)CD) <u>00</u>
+	F	F								
+	56780000	00000000	0000	0000 (0	000000	00000000	0FFF0000	000000	00),56780	00000000
	0000000	0)3)37	DEFFFC	3C1E0FFF	3C04567	8)0000	00000			(3401)
OpCode										
😐 🔶 opcode	0D	00)0D				0F)OD)0F)(D	<u>)0C</u>
🖅 🔶 func	0D	00)04/	(03	<u>)0</u>	5	3F)30	(38)(DD)00
RegisterFile										
💶 🔶 ra(31)	0000000	00000000								
😐	0FFFFFC	00000000					OFFF0000	OFFFFFFC		
😐	00000000	00000000								
😐 🔶 ra(18)	00000000	0000000					4			
😐 🔶 ra(17)	00000000 🗡	0000000					,			
💶 🔶 ra(16)	OD states that	the					I instruction			
	process will pe	erform					F into the f	list		
	ORI, thre cons	ecutive			anorreg	ISLEF 30	. The next			
	times from 400				Structio	n UD, WI	II ORI what	ts in r		
	400002, Writin						nd then plac			
+	5 in registers 1			IN	30 agai	n.			\vdash	
	3.	, <u> </u>	_							
+	00000000	000000	- Andrew -							
+	00000000	00000000	~							
+	00000000	00000000								
	00000000	00000000								
+ √ ra(4)	56780000	00000000		and the second second					56780000	(56784BCD
	0000005	00000000		- 20-	1000	00005				
+	0000003	00000000	7	1000000						
+	0000004	00000000	00000004			0F	again then	nlaced 5	678 in	
+	0000000	00000000				the	unner of r	praced 5	678in /	
0m	04	00				the	upper of ren of	n with AB	CD and) <u>oo</u>
rr1	04	00 01	02)0:	3		ced in R4 a	n miur4D		<u>)</u> 05
q0	56780000	00000000						iyani.	30	0000000 (000
😐 🔶 q1	56780000	00000000	0000	0000)0	000000	000			30	0000000 (0000000
😐 🔶 d	56784BCD	0)0000000	4 (0000	0003 (0	000005	0FFF-0000		100/000	ψ υ <u>კ</u>ეσ784	CD (00000000
· → rw	04	00 01	02	<u>(</u> 0)	3	(1E		04		<u>)</u> 05

:	00000530;	andi	5,0,\$0	
:	20288500;	add	5,4,5	here0:
:	0400DE23;	addi	30,30,\$4	
:	20308500;	add	6,4,5	
:	0000C5AF;	SW	5,30,\$0	
:	0000C58F;	lw	5,30,\$0	
:	FFFF2120;	addi	1,1,\$FFFF	
	:	: 20288500; : 0400DE23;	: 20288500; add : 0400DE23; addi : 20308500; add : 0000C5AF; sw : 0000C58F; lw	: 20288500; add 5,4,5 : 0400DE23; addi 30,30,\$4 : 20308500; add 6,4,5 : 0000C5AF; sw 5,30,\$0 : 0000C58F; lw 5,30,\$0

\$ 1•	Msgs								
🔷 dk	1								
I memclk	1								
. ⊥- ♦ Q	00400007	00400007 004	<u>), 80000</u>	400009	0040000A	0040000B	004000¢C	<u>)00400000)</u>	0040000E)
InstructionMem									
≖- ◆ q	34844BCD	34)30050000	00852820	23DE0004	0085302	20 AFC5	0000 (8FC5)	0000 2021FFFF	(1420FFF9
	1111	1111							
OpCode									
I → opcode	0D	DD (OC	00	08	00) <u>2</u> B	(23	08)05
· ∎	0D	00 <u>(</u> 00	20	04	20)00		3F)39
alu									
🛨 🔶 ia	56780000	56)00000000	<u>156784BCD</u>	OFFFFFFC				00000004	00000003
∎-∻ ib	00004BCD	0000000000		00000004)56784B0		00000	0000 FFFFFFFF	100000000
<u> </u>	0001	0001 0000	0010	0010		0010			0110
	56784BCD	56)00000000	56784BCD	10000000	ACF097	9A <u>(</u> 6678	4BCD (10000	00000003	00000003
RegisterFile									
	0000000	0000000			4000000				
	0FFFFFC 00000000	DFFFFFFC D000000			10000000				
+	00000000	0000000			<u> </u>				
+	00000000	Andi function							
+	00000000	control to go t							
+	00000000	with r0 to crea							
	00000000	back in 5		putsit	a	dd - opco	ode is 0, fur s r4+r5 and	nc is	
	00000000	DACKIII J							
+	00000000				b	ack into	5. Control o	ofalu	
+	00000000	0000000			is	5 now 00	10 to add		
+	0000000	0000000							
+	00000000	0000000				*			
	00000000	0000000							
+	0000000	0000000			/				
ra(6)	0000000	0000000				ACF0979A			
😐 🔶 ra(5)	0000000	0000000	(56	784BCD					
. <u>+</u> → ra(4)	56784BCD	56784BCD							
. <u>+</u> ra(3)	0000005	00000005							
. → ra(2)	0000003	0000003							
. → ra(1)	00000004	0000004)	0000003
. ⊥	0000000	0000000							
∎	04	04 (00	04	<u>(1E</u>	04	<u>(1E</u>		01	
. ⊞- ↔ m1	04	04)05		<u>(1E</u>	05			01)00
. ≖- 今 q0	56780000	56)00000000	,56784BCD	OFFFFFFC			0000	00000004	00000003
. ⊞- ∲ q1	56780000	56)00000000		OFFFFFFC				00000004)00000000
_ _ ♦ d	56784BCD	56)00000000	,56784BCD	(10000000	ACF097		56784BCD 56784		00000003
	04	04 (05		<u>(1E</u>	06	05		01	<u>)oo</u>
DataMem									

4 .	Msg	sĺ							
+ -◆ q	34010004	34)30050000	00852820	23DE0004	00853020	AFC50000	18FC50000	2021FFFF	1420FFF9
	1111	1111	100032020	72302000 T	,00033020	7,71 00000		2021111	<u></u>
- OpCode									
+	0D	OD XOC	200	08) <u>00</u>)2B	123	08)05
+	04	0D)00	120	104)20	<u>,20</u>		J3F	139
alu				<u></u>				121	100
	0000000	56)00000000	156784BCD	0FFFFFFC	(56784BCD	10000000		00000004	00000003
u → ib	0000004	00)00000000	,30701202	00000004	(56784BCD		200000000	FFFFFFFF	200000000
+	0001	0001)0000	10010	0010	,00701000	X0010			0110
− • • • • • • • • • • • • • • • • • • •	00000004	56)00000000		10000000	ACF0979A)66784BCD	10000000	00000003	00000003
RegisterFile			,3070 1000	<u></u>					<u></u>
+	0000000	0000000							
+	0000000	OFFFFFFC		Ť1	0000000				
+	0000000	0000000							
+	0000000								
+	00000000		nc = 20. r6	= 14+15		* †			
+	0000000	000			1000		ctoro what	ovorio	
+	0000000	00000000		/	, P	OP 2B will			
+	0000000	0000000		/			e address		
+	0000000	0000000					000000.Y		
+	0000000	0000000				see the da	ta coming f	rom	
+	00000000	0000000		1		memory ar	id the addr	ess	
+	00000000	0000000		1 and 1		being writt	ento. /		
+	0000000	0000000							
+	0000000	0000000				<u> </u>			
+	0000000	0000000				_ /			
+	0000000	00000000 Fu	inc 23, will			CF0979A			
+	0000000	00000000 at	r30, into r5	. Since th	e values		/		
∓ → ra(4)	0000000	56784BCD ar	e the same	, r5 will no	ot change,	. / .			
∓ → ra(3)	00000000	00000005 bu	it you can s	see the da	ita on the				
+	00000000	00000003 0U	it you can s itput of the	data men	nory and				
+	00000000		e address.		1			Ĭo	000003
+	0000000	0000000							
011 ↔_+	00	04 00				/ X1E		01	
	01	04)05		X1E	X05			01	100
+ → q0	00000000	56)00000000)56784BCD	(OFFFFFFC	156784BCD	10000000		00000004	100000003
	0000000	56)00000000		JOFFFFFFC	56784BCD	\wedge		00000004	00000000
	0000004	56)00000000)56784BCD	(10000000	ACF0979A	10 1567848	CD),56784BCD	00000003	00000003
+	01	04 05		χ1E	106	X05		101	100
DataMem									
+	04	CD)00)CD	200	19A	700		03	103
+	F	F							
+> data	00000000	56)00000000		(OFFFFFFC	(56784BCD		× 1	00000004	00000000
	37DEFFFC	0000000 3401000	04 0000000			000 1567848	CD 4	,3C1E0	

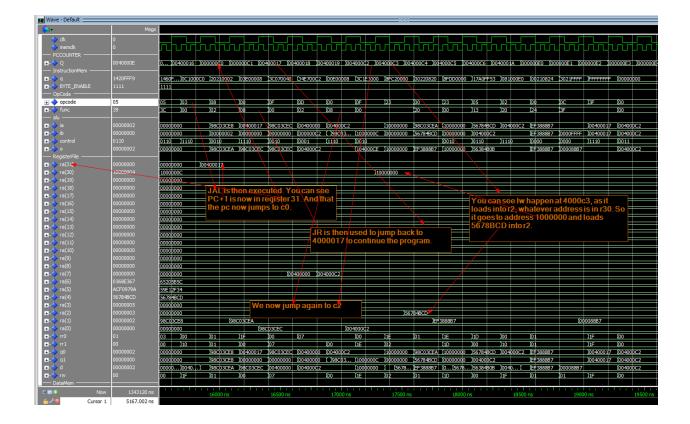
00e	:	F9FF2014;	bne	1,0,here0	
00f	:	FFFF4220;	addi	2,2,\$FFFF	here1:
010	:	1111C620;	addi	6,6,\$1111	
011	:	FDFF4014;	bne	2,0,here1	
012	:	FFFF6320;	addi	3,3,\$FFFF	here2



🔶 dk	0								-			
A memdk	ő											
- PCCOUNTER												
	0040000E	X0040000F X00	400010 1004	400011 100	140000E 10	1400010	00400011	0040000 100-	10010 10	400011 00	400012 100	1400013 1004
- InstructionMem			100010 ,00	100011 /00		5100010			100010 ,00		/100012 /00	
∓- ⊲	1420FFF9	14)2042FFFF	20C61111	1440FFFD	2042FFFF	20C6111	1)1440FFF	D J2042EEEE	20C61111	J1440FFFD	2063EEEE	100060820
	1111	1111										
OpCode												
+ 🔶 opcode	05	05)08		05)08		05	108		105	108	100
	39	39)3F	111	30 💊)3F) <u>11</u>	30)3 F	111	30 /	Ĵ3F	120
— alu ———												
🚛 🔶 ia	0000002	00 00000003	80597801	00000002		B0598C1	2 0000000	1	80599D23	00000000	00000005	B059AE34
	00000000	00 FFFFFFFF	00001111	00000000	FFFFFFFFF	0000111	1 0000000	O FFFFFFFF	000011111	00000000	FFFFFFFFF	B059AE34
•	0110	0110 0010		0110	0010		0110)0010		0110	0010	
	0000002	00 00000002	B0598C12	00000002	00000001	8059902	3 0000000	1 00000000	B059AE34	00000000	00000004	60835C68
RegisterFile												
😐 🔶 ra(31)	00000000	00000000										
😐	10000004	1000000C					No.	Λ.				
😐	00000000	0000000							/			
😐	00000000	0000000							/			
💻	00000000	0000000			honrog	rom the	n movor	to 10000	E who	ti ni		
	00000000	00000000			ne prog		moves	to 400000 in r2 down	JE, WIIE	en		
🖭 🕁 ra(15)	00000000	0000000										
	00000000	0000000		C	necks IT	it is eq	ual to ru,	which is r nes back	lotand			
	00000000	0000000		C C	onunue	S to 100	p for 2 u	nes back	o nere i			
	00000000	0000000			etorem	ovingto	400010	. Through	this who			
	00000000	0000000		t t	meitis	adding	\$1111 to	D 16.				
	00000000	0000000										
	00000000	0000000										
	00000000	0000000					_					
	00000000	00000000										
🛨 🕁 ra(6)					/			<u>₹</u>				
	0368E367	80597801) <u>B0</u>	598C12		,	80599D23)B0	59AE34		
■ → ra(5)	ACF0979A	B0597B01 59E12E34)B0:	598C12		Ż	80599D23)BC	59AE34		
😐 🔶 ra(4)	ACF0979A 56784BCD	B0597B01 59E12F34 56784BCD)BOI	598C12			80599D23)BO	59AE34		
■ ra(4) ■ ra(3)	ACF0979A 56784BCD 00000005	B0597B01 59E12F34 56784BCD 00000005		598C12			80599D23			59AE34)(00	2000004
■ → ra(4) ■ → ra(3) ■ → ra(2)	ACF0979A 56784BCD 00000005 00000003	80597801 59E12F34 567848CD 00000005 00000003 (00)B05	598C12	.01	0000001	80599023)B0	159AE34)joc	
• ra(4) • ra(3) • ra(2) •	ACF0979A 567848CD 00000005 00000003 00000002	E0597801 59E12F34 56784ECD 00000005 00000003 X00 00000000		598C12	jo (0)		80599D23			59AE34)joc	
	ACF0979A 567848CD 00000005 00000003 00000002 00000000	E0597801 59E12F34 56784ECD 00000005 00000003 X00 00000000 00000000	000002***			0000001		X001	00000			(60B
■ → ra(4) ■ → ra(3) ■ → ra(2) ■ → ra(1) ■ → ra(0) ■ → rr0	ACF0979A 56784BCD 00000005 00000003 00000002 00000000 01	B0597B01 59E12F34 56784BCD 00000005 00000003 000 00000000 00000000 01 002	000002****	02		0000001	<u>)02</u>		00000	<u>)</u> 02)03)60B
■ → ra(4) ■ → ra(3) ■ → ra(2) ■ → ra(1) ■ → ra(0) ■ → rn0 ■ → rn1	ACF0979A 567848CD 0000005 00000003 00000002 00000000 01 00	B0597801 59E12F34 56784BCD 00000005 00000000 00000000 00000000 01 02 00 002)06)06	02)02)06)06) <u>()2</u>)(02)02	000000 000000 006 006)(02)(00	<u>)03</u>)608)06)06
$\begin{array}{c} \hline \hline$	ACF0979A 567848CD 00000005 00000002 00000000 01 00 00000000 00000002	B0597801 59E12F34 56784BCD 00000003 00000000 00000000 00000000 01 002 00 002 00 002 00 002	000002)05)05)05)80597801	02 00 0000002	<u>)</u> 02)06)06)06)06)06)06)06)06)06)06)02)00 2)0000000) <u>02</u> 1	000000 005 005 005 00599023)(02)(00)(00)(00)(00)(0)	<u>)03</u>)03)00000005)608)06)06)8059AE34
$C \rightarrow ra(4)$ $C \rightarrow ra(3)$ $C \rightarrow ra(2)$ $C \rightarrow ra(1)$ $C \rightarrow ra(0)$ $C \rightarrow r1$ $C \rightarrow r1$ $C \rightarrow r1$ $C \rightarrow q0$ $C \rightarrow q1$	ACF0979A 567848CD 00000005 00000003 00000002 00000000 01 00 000000002 00000002 00000002	80597801 59E12F34 567348CC 00000005 00000000 00000000 01 022 00 022 0000000003 0000000003	000002)06)80597801)80597801)02)00)00000002)00000002	<u>)02</u>)00000002)06)06)06)06)80598C1)80598C1)02)00 .2)0000000 .2)0000000)02 1 0)00000001	000000 06 06 80599D23 80599D23)(02)(00)(00000000)(00000000))03)03)00000005)00000005)608)06)05)8059AE34)8059AE34
C → ra(4) C → ra(3) C → ra(2) C → ra(2) C → ra(1) C → ra(0) C → ra(1) C → ra(2) C → ra(3) C → ra(4) C → ra(4) C → ra(5) C → ra(1) C → ra(2)	ACF-0979A 567848CD 00000003 00000002 00000000 01 00 00000000 00 00000002 00000002	80597801 59E12F34 567848CD 00000005 00000000 00000000 01 022 00 022 0000000003 0000000003 0000000000	000002 006)06)80597801)80597801)80597801	02 00 0000002 0000000 0000000)02)00000002)00000001)06)06)06)80598C1)80598C1)80598C1	X02 X00 .2 X0000000 .2 X0000000 .3 X0000000	X02 1 0 X00000001 1 X00000000	000000 06 06 00599D23 00599D23 00599AE34)02)00)00000000)00000000)00000000	X03 X03 X000000005 X00000005 X00000004)608)06)8059AE34)8059AE34)8059AE34)60835C68
C - ↔ ra(4) C - ↔ ra(3) C - ↔ ra(2) C - ↔ ra(2) C - ↔ ra(1) C - ↔ ra(0) C -	ACF0979A 567848CD 00000005 00000003 00000002 00000000 01 00 000000002 00000002 00000002	80597801 59E12F34 567348CC 00000005 00000000 00000000 01 022 00 022 0000000003 0000000003	000002)06)80597801)80597801	02 00 0000002 0000000 0000000	<u>)02</u>)00000002)06)06)06)06)80598C1)80598C1)02)00 .2)0000000 .2)0000000)02 1 0)00000001	000000 06 06 06599023 00599023 005994E34)(02)(00)(00000000)(00000000))03)03)00000005)00000005)608)06)05)8059AE34)8059AE34
C → ra(4) C → ra(3) C → ra(3) C → ra(2) C → ra(2) C → ra(0) C → ra(0)	ACF0979A 567848CD 00000005 00000003 00000000 01 00 00000000 00000002 00000002 00000000	20597801 5912784 56178400 00000005 00000005 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000)06)06)06)80597801)80597801)80598012)05	02 00 00000002 00000002 00000002) <u>02</u>) <u>00000002</u>) <u>00000001</u>) <u>02</u>)06)06)06)06)06)00598C1)00599C2)00599D2)006)02)00 2)0000000 2)0000000 3)0000000)00)02 1 0)00000001 1)00000000)02	000000 006 005 00599D23 00599D23 0059AE34 005)(02)(00)(00000000))(00000000))(00000000)03)03)00000005)00000005)00000004)03)06)06)06)0059AE34)0059AE34)60835C68)01
0 - 4 ra(4) 0 - 4 ra(3) 0 -	ACF0979A 557346CD 00000005 00000003 00000000 01 00 00000000 00000000	B0597801 S9E1754 S9E1754 S92401 D000005 D000005 D000005 D000005 D000000 D19700 D19700 D19700 D00000 D192 D0	0000002 006)06)80597801)80597801)80598C12)06	02 00 00000002 00000002 00000002	<u>у</u> 02 <u>уоооооооооооооооооооооооооооооооооооо</u>)06)06)06)06)06)00598C1)00599C2)00599D2)006)02)00 2)00 2)0000000 3)0000000 3)0000000)00)02 1 0)00000001 1)00000000)02	000000 06 06 060 060 00599D23 00599D23 00599D23 00599D23 00599D23)(02)(00)(00000000))(00000000))(00000000)03)03)00000005)00000005)00000005)00000004)03)06)06)06)0059AE34)0059AE34)60835C68)01

🔷 dk 🕁 memdk	0	- truttint	┍┥┥┝╘╼┯┤		╴┍┟┲┥	╧╴┍┾	╤┶┍╆	┍┥┥┝						+				╎╴┍┝╞	╦┙┞╼
- PCCOUNTER		التلقا ولقات ا																	
⊳- ∲ Q	0040000E	00400013 004000	14 00400015 00	0400012	00400013	00400014	1 004000	15 0040	0012 00	400013 00	400014	0040001	5 004	00012 00	400013 00	400014	00400015	00400012	00400013
— InstructionMem ——		ر صند صدن کے																	
н> q	1420FFF9	206 000 60820 000	13020 1460FFFC	2063FFFF	000060	320 00C1	3020 146	OFFFC	063FFFF	00C60820	00C130	20 (146	OFFFC	2063FFFF	00C60820	00C130	0 1460FFF	-C (2063F	FFF 00C60
	1111	1111																	
	_																		
🔷 opcode	05	0 <u>8 (</u> 00	05	08	<u>(00</u>		05	, 10	8)00		<u>)05</u>	/	08	00		05	03	(00
-🔶 func	39	3F (20	J3C)3F	(20		<u> 3¢</u>		F	20		3C		(3F	20	~)3C)3F	20
⊢� ia	0000002	000)B059AE34	00000004		(110D0)			00003		33271FD4			00002		99755F7C		000000		CC60
-🔶 ib	00000000		0835 00000000		111000	<u>19C (22</u>				33271504)664E				99755F7C	32EAE			FFF CC60
-🔶 control	0110	0010	0110	0010			.011		010			011		0010			0110	0010	
- > •	0000002	000)60B35C68 [1	10D0 00000004	0000000	3 <u>(221A1</u>	538 [33]	271FD4,000	00003	0000002	664E3FA8	19975	5F7C(000	00002	00000001	32EABEF8	<u>(CC60</u>		01 00000	000 (9800)
- RegisterFile																			
	00000000 10000004	0000000					_												
ra(30)	00000004	1000000C																	
-◆ ra(19) -◆ ra(18)	00000000	0000000			TI	ie prog	ram the	n start	s to de	crement	what	is in r4	land						
	00000000	0000000			<u> </u>	mpares	s it with	r0, and	d loops	back to	here2	24 time	es						
ra(17) ra(16)	00000000	0000000								it is con	tinual	ly doir	ıg r6						
	00000000	0000000			==	r ,1+r6 , a	and r _i 1 =	r6+r1											
	00000000	0000000			 /														
- ra(13)	00000000	00000000			//				-		-								
🔶 ra(12)	00000000	00000000		/	/		$/ \vdash$)	~							
						- 1						~							
🔶 ra(11)		0000000																	
	00000000	0000000		/ /		/			Ì			X	<u> </u>						
🔶 ra(10)				\sim		_/						- Ar	A.						
	00000000	0000000				\neg							A. A.						
- ra(10) - ra(9) - ra(8)	00000000 00000000	0000000	and the second sec										and the second						
 ↓ ra(10) ↓ ra(9) ↓ ra(8) ↓ ra(7) 	00000000 00000000 00000000	00000000 00000000 00000000	Jijopoaec 🌌	ver of		/)33271Fi)4				(99755F)	c	Jan			CC601E74		
 √ ra(10) √ ra(9) √ ra(8) √ ra(7) √ ra(6) 	00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000	ATIODOA9C #	ver de la compañía de		/)33271FI)4)99755F7	c				CC601E74		
 ✓ ra(10) ✓ ra(9) ✓ ra(8) ✓ ra(7) ✓ ra(6) ✓ ra(5) 	00000000 00000000 00000000 00000000 0368E367	00000000 0000000 0000000 0000000 0059AE34 59E12F34 59E12F34 567848CD	JIIODOA9C 8	and the second sec) <mark>(3</mark> 3271FI	04				<u>)99755₽7</u>	c				CC501E74		
 → ra(10) → ra(9) → ra(8) → ra(7) → ra(6) → ra(5) → ra(4) 	00000000 0000000 0000000 0368:367 ACF0979A 567848CD 0000005	00000000 0000000 00000000 00000000 00000)IIODOASC #		0000003)33271FI	04)00	000002		<u>}99755</u> ∓7	c)))00	000001		CC601E74		(00000000
- √ ra(11) - √ ra(10) - √ ra(8) - √ ra(8) - √ ra(6) - √ ra(6) - √ ra(5) - √ ra(1) - √ ra(1)	00000000 0000000 0000000 0368E367 ACF0979A 567848CD	00000000 0000000 0000000 0000000 0059AE34 59E12F34 59E12F34 567848CD		, , , , , , , , , , , , , , , , , , ,	/) 221A1538		04)00		4E3FA8)99755F7	C)00		TABEF 8	CC601E74		<u>(00000000</u>

016 017 018 019	: : :	C000100C; 4000073C; 0803E734; 0800E000;	jal lui ori jr	\$00400300 7,\$0040 7,7,\$308 7	
01a [01b	: 0b	E0001008; f] : 00000000;	j	\$00400380	back:
0c0	:	02002120;	addi	1,1,\$2	
0c1	:	0800E003;	jr	31	
0c2	:	00101E3C;	lui	30,\$1000	
0c3	:	0000C28F;	lw	2,30,\$0	
0c4	:	20082200;	add	1,1,2	



0c5 : 0000DD8F;	lw	29,30,\$0
0c6 : 53FFA017;	bne	29,0,back
[0c70df] : 00000000;		
0e0 : 24082100;	and	1,1,1
0e1 : FFFF2130;	andi	1,1,\$FFFF
0e2 : FFFFFFF;	GFO	
[0e30ff] : 00000000;		

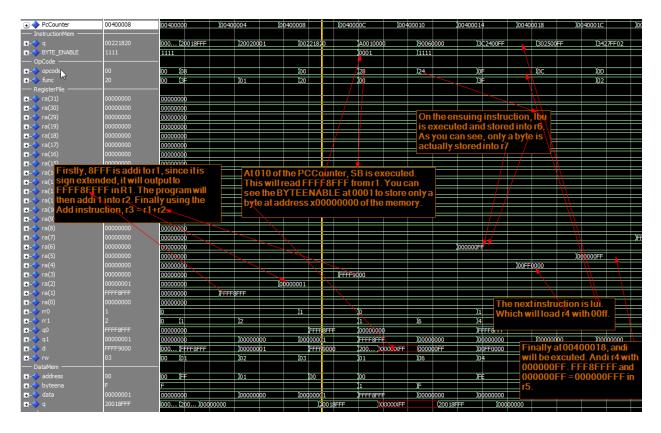
€ 1•	Msgs		
< dk	0		
📥 memclk	0		
∓- ∕ Q	004000C5	14000C5 004000C6 0040001A 000000E0 00000	0001 0000000000000000000000000000000000
InstructionMem ——			
	8FDD0000	.)8F0 D0000 (17A0FF53)081000E0)00210824 (30	
	1111		<u>, , , , , , , , , , , , , , , , , , , </u>
— OpCode ———			
+	23)23)05)02)00 <u>)</u> 00	c);3F);00
+	00	100 113 120 124 13F	
— alu —			
uiu ∓–♦ ia	1000000	. 10000000 156784BCD 1004000C2 1)EF3888B7	00400017 004000C2
<u>+</u> ↓ ib	00000000		000FFFF 00400017 004000C2
+	0010		000 /1110 /0011
	1000000		
➡	1000000	. <u>(10000000) (56384808) (00</u>	00088B7 004000C2
	00400017	10001	
+	1000000	400017	
⊥			
	0000000	000000 (56784BCD	
+	0000000	00000b	
+ 🔷 ra(18)	0000000		adin 20 Then
+ → ra(17)	0000000	At c5, D[1000000] is loade 00000 BNE is to check r0 and r29.	Since they are
∓ → ra(16)	0000000	BNE IS to check ru and r29.	which then
+	0000000	not equal it jumps to "back" jumps to e0 and finishes ou	
+	0000000		it the program.
+	0000000		
+ → ra(12)	0000000		
+	0000000		
+⊢	0000000	00000	
+	0000000	00000	
+	0000000	00000	
+	004000C2	1000C <mark>2</mark>	
-	65205B5C	20585 <mark>C</mark>	
-	59E12F34	E12F3 <mark>4</mark>	
⊥	56784BCD	7848CD	
+	0000000	0000 <mark>0</mark>	
∓ –� ra(2)	56784BCD	784BCD	
+	EF3888B7	3888 7)000088B7
-	004000C2	100002	
•	1E	(1E)(1D)00)01	<u>χīe χoo</u>
-	1D	(1D)00 (10)01	<u>, 15)00</u>
-	1000000	. (10000000)56784BCD)004000C2)EF 3888B7	00400017 00400DC2
-	0000000	. (00000000)004000C2)EF 3888B7	00400017 00400DC2
	0000000	.) (5678)56384B0B)0040))EF3888B7 (00	
u → +	1D	(1D)00 (1F)01 (01	

Lab4baotung_test.mif

*** The annotated mif file will be included in the report

I created my own program that ran through every single besides for addiu. Addiu will be annotated when looking at the program flow of lab4_test.mif. Also, the pc is incremented by +4, and not +1.

000 : 20018fff; 004 : 20020001; 008 : 00221820; 00C : A0010000; 010 : 90060000; 014 : 3C2400FF; 018 : 302500FF;

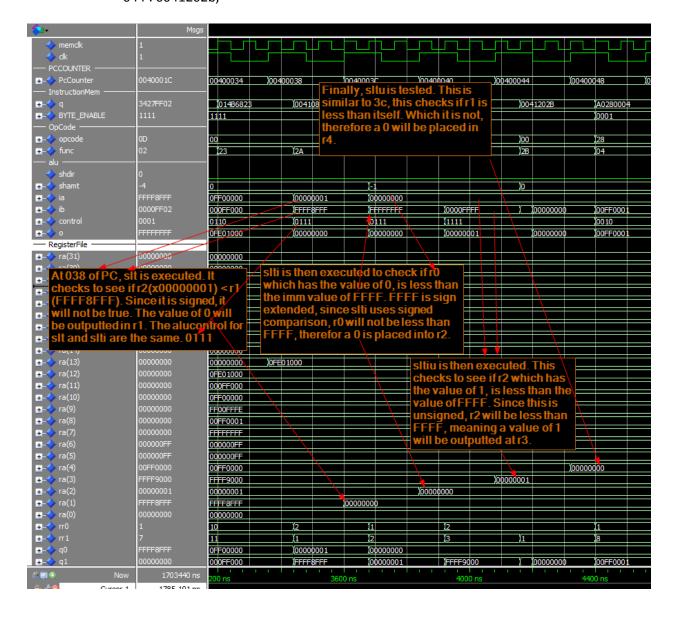


01C: 3427FF02; 020:00824025; 024:00884827; 028:00045100; 02C:00045902; 030:014B6022; 034:014B6823;

∲ ⊒•	M	sgs	7						
🔷 dk	0								
- PCCOUNTER									
	00400010	0040001C	00400020	00400024	00400028	0040002C	00400030	00400034)00400
InstructionMem									
	90060000	30250)3427FF02	0082402	5 (0088482)	00045100	00045902	01486022	0148682	3
BYTE_ENABLE	1111	1111							_
OpCode									
	24	OC (OD	00						ي كما ت
	00	3F)02	25)27)00	02	22	23	
RegisterFile									
😐 – 🔷 ra(31)	0000000	0000000							
. → ra(30)	00000000	0000000							ي و و و ا
. 	0000000	0000000							ر <u>م</u>
∓ → ra(19)	00000000	00000000							
	er 40001C, Or	0000 Noriotho	executed be	woord					ي وي ال
	etween r4 and	0000 Nor Is the	executed be						ي وي ا
	0 or 00000001	and r8, res							
= 00FF0001	into r8.	0000 at r9. NOR	c = 1100 aluc						
		0000							
•		0000000							OFEO:
. → ra(12)	00000000	00000000						0FE01000	
	00000000	00000000					000FF000		
	00000000	0000000				0FF00000			
	00000000	0000000			FF00FFFE				
	0000000	0000000		00FF0001					
	0000000	0000000)FFFFFNFF						إعصده
	00000000	000000FF							
	00000000	2000000FF							
	00000000 FFFF9000	00FF0000							
	00000001	FFFF9000 00000001							
+	FFF8FFF	FFFF8FFF							
+	00000000	00000000	, ,						
	00000000	1	<u>1</u> 4	<u>}</u>	Хо		<u> </u>		
	e	5 17	12	18	<u>)0</u>)(4		<u>110</u>		
±-√ q0	00000000	FFFF8FFF	122 X00FF		00000000)0FF00000		
+	00000000	00000 00000000	1 20000				(000FF000		
	0000000FF	00000)FFFFFFFF	0000 X00FF			X000FF000			
■_→ rw	06	00000	108	109	000000 00	JOB	10C	20D	
			100		, jun	700	, vc	100	
uu ∓ ∲ia	00000000	FFFF8FFF	(00FF	0000	00000000		(0FF00000		
∎ → ib	00000000	00000 0000FF02	1 20000				X000FF000		
	0010	0000 10001	1,0000	11100)0011		10110		
±-→ 0	00000000	00000)FFFFFFFF	(OOFF			(000FF000			
		00000	,0011	2004 gr 00[11]	2011-00000	2000FT 000	Jon CO 1000		

& -	Msgs															
	1															
🔶 dk	1		R.													
PCCOUNTER	_															
	0040001C	0040)004	00024		00400028	0040	002C		00400030		00400	034)004	00038		0040003
InstructionMem																
🖅 🔶 q	3427FF02	00824025	0088	4827	0004510	0	000459	02	01	B6022		014868	23	0041	082A),
BYTE_ENABLE	1111	1111														
OpCode																
	0D	00														l (
	02	25	27		<u>)oo</u>		02		22			23)2A		L X
alu																
I shdir	0															
∎	-4	0			4)0							<u>1</u>
	FFFF8FFF	00FF0000	-	_	0000000					00000				0000		⊥`
	0000FF02	00000001			00FF000	0		<u> </u>		FF000)FFFF		
	0001	0001	(1100		0011	_			01					0111		(
∃ - ◇ ○	FFFFFFF	00FF0001)FEØÓ	FFFE	0FF0000	0)000FFC	00	<u>,0FE</u>	01000)0000	0000)(
RegisterFile	0000000		_	/				4								
	00000000	0000000														
		0000000						<u> </u>								
,≞ 🛧 At 028, sll is	executed with	a shami	i eest	On th	e ensuing i	nstruct	ion sr	lis =								
of 4 to the le	ft. shdir goes	io 1,		used	to shift r4 r	iaht hv	4 hits	and								
Indicating le	eft. This instruc	tion		nlace	itin r11 Y	oucan	see sh	dir								<u> </u>
🕂 shifts r4 left 🛉	eft. This instruc by 4 bits and p	laces it		nnes	itin r11. Y back to 0, i	ndicati	ngash	iff =								
	in the alu cont	rol		to the	right.	laicau	ig a si	···· =								
indicate shift	ft w/direction f	orboth			ingine			⊨								
sll and srl.													Yoer	01000		
+ ra(12)	0000000	00000000									0FE01	000	<u>JUFI</u>	01000		
+	00000000	00000000							000FF000		JUFEUI	000		<u> </u>		
= ra(10)	00000000	00000000)OFFC	0000	- '								
	00000000	00000000			FFOOFFFE	joiric	0000				/					
	00000000	00000 (00F								i			<u> </u>			
	00000000	FFFFFFF	10001						The nex							
	000000FF	000000FF						8	ind sub	ou. In I	both, r	10-r1	l is ex	cuted		
= ra(5)	000000FF	000000FF						a	ind pla	ced ir	1r12 [′] a	nd r13	3.			
+	00FF0000	00FF0000							especi							
+	FFFF9000	FFFF9000														
=_→ ra(2)	00000001	00000001														
\rightarrow ra(1)	FFFF8FFF	FFFF8FFF														0000000
	00000000	00000000														
	1	4			Хо				<u>х</u> іо					12		Y
∎ m1	7	2)8		<u>14</u>) <u>11</u>					<u>)/2</u> 11		Ý
	FFFF8FFF	2 00FF0000			0000000	0				00000				10000	0001	- A
= − → q1	00000000	00000001		0001	00FF000					FF000)FFFF		Ť
A R O Now		1		1.1	1 1 1 1 1							1			1.1	
				240	10 ns		280	0 ns			320	00 ns			36	00 ns
Cursor 1	1/85.191 ns															

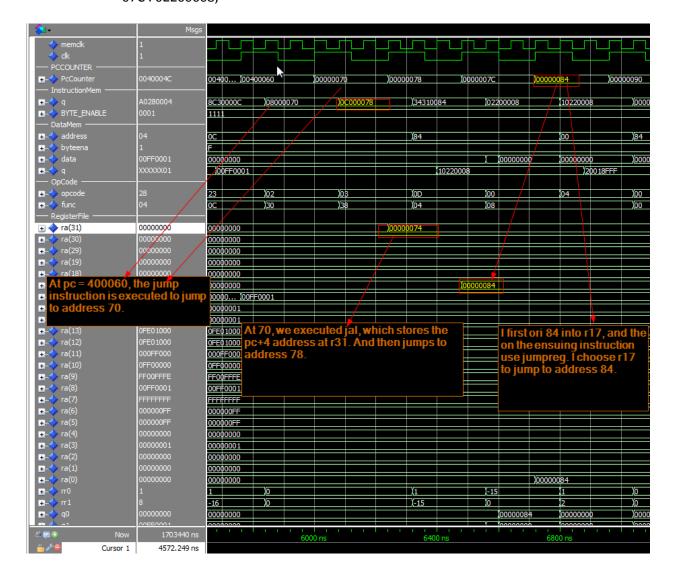
038 : 0041082A; 03C : 2822ffff; 040 : 2C43ffff; 044 : 0041202b;



- 048 : A0280004;
- 04C:A4280008;
- 050 : AC28000C;
- 054:902E0004;
- 058:942F0008;
- 05C:8C30000C;

- {2	Msgs															
I memcik	1															
🔶 dk	1															
- PCCOUNTER																
	0040004C	0)004000	48	0040004C	0040005))00	100054)00	400058		004000	5C	004	00060		00000
InstructionMem																
 → <u>q</u>	A0280004	0041202B	A0280004	A428	0008)A	C28000C)902E00	04	942	F0008		8C30000	с	08000	070	
<u>■</u> <u>→</u>	0001	1111	0001	0011	<u>)</u> 1	111										
DataMem																
	04	00	04)08)0	C C)04		08			0C				
🖅 🥧 byteena	1	F	1)3)F											
💶 - 🔷 data	00FF0001	00000000					000000)	0000000		0000000				
. ⊞- 今 d	XXXXXXX01	20018FFF	, x000000	01	XXXX0001	00FF0001		20020001		00220	0001)0	DFF0001			
OpCode																
IIII → opcode	28	00	28	29)2		24		25			23		02		
	04	2B	04)08)0	c /	04		08			0C)30		
RegisterFile					•	<u>+</u>										
+ 🔷 ra(31)	00000000	00000000	During E	00 48 40	c, and 50, w	,	.h									
	00000000															
	0000000	0000000	SII, dilus	onoblo	executed. Y going from 0	01 to 0011										
	0000000	00000000	the 11114	enable	joing noin u											
ra(18) (17)	0000000	00000000		o accou	nt for each c g the byte, f	ase. The										
	00000000	00000000	program	IS STOLL	ig ine byte, r	w, and wi	or									
	00000000 00000000				ne address o	0100,04,a	na				Vaaaaaaa		,00F	F0001		
			08 in me	mory							000000	01				
	0000000	0000000	T						00001							
	00000000	00000000						<u> </u>	000001		=					
	0FE01000	0FE01000	<u> </u>					 /	000001							
	0FE0 1000 0FE0 1000	0FE01000 0FE01000						100	000001							
	0FE01000 0FE01000 000FF000	0FE01000 0FE01000 000FF000						100	000001							
	0FE0 1000 0FE0 1000 000FF000 0FF00000	0FE01000 0FE01000 000FF000 0FF00000														
	0FE01000 0FE01000 000FF000	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE				54	to 5c wil		ute Ib) Ih and	diw, a	nd loa	,			
□→ ra(13) □→ ra(12) □→ ra(12) □→ ra(10) □→ ra(9) □→ ra(8)	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE 00FF7001	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE 00FF0001				54	to 5c wil values		ute Ib	lh an d Oc in	d Iw, a to r14	and loa -r16				
	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE				54 the	to 5c wi		ute Ib) Ih an d Oc in	d Iw, a to r14	and loa -r16	ad			
□→ ra(13) □→ ra(12) □→ ra(12) □→ ra(10) □→ ra(9) □→ ra(8)	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE 00FF0001 FFFFFFFF	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE 00FF0001 FF0FFFFF				54 the	to 5c wi values		ute Ib) Ih an 10 c in	d Iw, a to r14	nd loa -r16	ad			
$\begin{array}{c} \mathbf{p}_{-\infty} & ra(13) \\ \mathbf{p}_{-\infty} & ra(12) \\ \mathbf{p}_{-\infty} & ra(11) \\ \mathbf{p}_{-\infty} & ra(0) \\ \mathbf{p}_{-\infty} & ra(9) \\ \mathbf{p}_{-\infty} & ra(6) \\ \mathbf{p}_{-\infty} & ra(6) \\ \mathbf{p}_{-\infty} & ra(6) \end{array}$	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE 00FF0001 FFFFFFFF 000000FF	0FE01000 0FE01000 00FF000 0FF0000 FF00FFFE 00FF0001 FFFFFFF 000000FF				54 the	to 5c will		ute Ib) Ih an d Oc in	d Iw, a to r14	nd loa -r16	ad			
$\begin{array}{c} \mathbf{p}_{-\infty} ra(13) \\ \mathbf{p}_{-\infty} ra(12) \\ \mathbf{p}_{-\infty} ra(10) \\ \mathbf{p}_{-\infty} ra(9) \\ \mathbf{p}_{-\infty} ra(8) \\ \mathbf{p}_{-\infty} ra(7) \\ \mathbf{p}_{-\infty} ra(6) \\ \mathbf{p}_{-\infty} ra(6) \\ \mathbf{p}_{-\infty} ra(5) \end{array}$	0FE01000 0FE01000 000FF000 0FF00000 0FF00FFFE 00FF0001 FFFFFFFF 000000FF 000000FF	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE 00FF0001 FFFFFFF 000000FF 000000FF				54 the	to 5c wil values		ute Ib	Jih an 1 Oc in	d Iw, a to r14	ind loa -r16	ad			
P→→ ra(13) P→→ ra(12) P→→ ra(11) P→→ ra(10) P→→ ra(8) P→→ ra(7) P→→ ra(6) P→→ ra(5) P→→ ra(4)	0FE01000 0FE01000 000FF0000 0FF00000 FF000FFE 00FF0001 FFFFFFF 000000FF 000000FF 000000FF	0FE01000 0FE01000 0FF0000 0FF00000 FF00FFFE 00FF0001 FFFFFFF 000000FF 000000FF 0000000				54 the	to 5c wil values		ute Ib) Ih an 1 Oc in	d Iw, a to r14	and loa -r16	ad			
$\begin{array}{c} \mathbf{p}_{-\infty} & ra(13) \\ \mathbf{p}_{-\infty} & ra(12) \\ \mathbf{p}_{-\infty} & ra(10) \\ \mathbf{p}_{-\infty} & ra(9) \\ \mathbf{p}_{-\infty} & ra(9) \\ \mathbf{p}_{-\infty} & ra(8) \\ \mathbf{p}_{-\infty} & ra(6) \\ \mathbf{p}_{-\infty} & ra(6) \\ \mathbf{p}_{-\infty} & ra(6) \\ \mathbf{p}_{-\infty} & ra(4) \\ \mathbf{p}_{-\infty} & ra(3) \end{array}$	0FE01000 0FE01000 000FF0000 0FF00000 FF000FFE 00FF0001 FFFFFFF 0000000FF 0000000FF 0000000FF 000000	0FE01000 0FE01000 0FE0000 0FF0000 FF00FFE 00FF001 FFFFFFF 000000FF 000000FF 0000000 0000001				54 the	to 5c will		ute Ib) Ih an 1 Oc in	d Iw, a to r14	nd loa -r16	ad			
$\begin{array}{cccc} \mathbf{r}_{0}(13) \\ \mathbf{r}_{-} & \mathbf{r}_{0}(12) \\ \mathbf{r}_{-} & \mathbf{r}_{0}(11) \\ \mathbf{r}_{-} & \mathbf{r}_{0}(10) \\ \mathbf{r}_{-} & \mathbf{r}_{0}(9) \\ \mathbf{r}_{-} & \mathbf{r}_{0}(8) \\ \mathbf{r}_{-} & \mathbf{r}_{0}(8) \\ \mathbf{r}_{-} & \mathbf{r}_{0}(6) \\ \mathbf{r}_{-} & \mathbf{r}_{0}(2) \\ \mathbf{r}_{-} & \mathbf{r}_{0}(2) \end{array}$	0FE01000 0FE01000 000FF000 00FF0000 FF00FFFE 00FF0001 FFFFFFF 000000FF 000000FF 00000000	0FE01000 0FE01000 0FF0000 0FF00000 FF00FFFE 00FF0001 FFFFFFF 000000FF 000000FF 0000000 00000001 00000000				54 the	to 5c will		ute Ib) Ih an 1 Oc in	d Iw, a to r14	nd loa -r16				
$\begin{array}{c} \mathbf{p}_{-\infty} ra(13) \\ \mathbf{p}_{-\infty} ra(12) \\ \mathbf{p}_{-\infty} ra(10) \\ \mathbf{p}_{-\infty} ra(9) \\ \mathbf{p}_{-\infty} ra(9) \\ \mathbf{p}_{-\infty} ra(6) \\ \mathbf{p}_{-\infty} ra(6) \\ \mathbf{p}_{-\infty} ra(6) \\ \mathbf{p}_{-\infty} ra(6) \\ \mathbf{p}_{-\infty} ra(2) \\ \mathbf{p}_{-\infty} ra(2) \\ \mathbf{p}_{-\infty} ra(2) \\ \mathbf{p}_{-\infty} ra(1) \end{array}$	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE 00FF0001 FFFFFFF 000000FF 000000FF 0000000F 000000	0FE01000 0FE01000 0FF0000 FF00FFE 00FF0001 FFFFFFF 000000FF 0)000000 00000001 00000001 0000000				54 the	to 5c wii values		ute Ib	P Ih and 1 Oc in	d Iw, a to r14	and loa -r16)0		
$\begin{array}{c} \mathbf{R} - \mathbf{v} & ra(13) \\ \mathbf{R} - \mathbf{v} & ra(12) \\ \mathbf{R} - \mathbf{v} & ra(12) \\ \mathbf{R} - \mathbf{v} & ra(10) \\ \mathbf{R} - \mathbf{v} & ra(9) \\ \mathbf{R} - \mathbf{v} & ra(9) \\ \mathbf{R} - \mathbf{v} & ra(9) \\ \mathbf{R} - \mathbf{v} & ra(7) \\ \mathbf{R} - \mathbf{v} & ra(1) \\ \mathbf{R} - \mathbf{v}$	0FE01000 0FE01000 000FF000 0FF00000 FF00FFFE 00FF0001 FFFFFFF 000000FF 000000FF 0000000F 000000	0FE01000 0FE01000 0FF0000 FF00FFE 00FF0001 FFFFFFF 000000FF 0)000000 00000001 00000001 0000000				54 the	to 5c wii values		ute Ib	J Ih and d Oc in	to r14	-16	ad)0)0		
$\begin{array}{cccc} \mathbf{r}_{0}(13) \\ \mathbf{r}_{} & \mathbf{r}_{0}(12) \\ \mathbf{r}_{} & \mathbf{r}_{0}(11) \\ \mathbf{r}_{} & \mathbf{r}_{0}(0) \\ \mathbf{r}_{$	0FE01000 0FE01000 000FF0000 0FF00000 FF00FFFE 00FF0001 FFFFFFF 000000FF 000000FF 00000000	0FE01000 0FE01000 00FF0000 0FF0000 FF00FFFE 000000FF 000000FF 0000000FF 0000000F 0000000 00000000	2000 11 18			54 the	Xalues	Il exec at 00 (ute Ib)4 and χ ₁₅	1 Oc in	to r14	- r16				
$\begin{array}{cccc} r_{0}(13) \\ r_{-} & r_{0}(12) \\ r_{-} & r_{0}(11) \\ r_{-} & r_{0}(0) \\ r_{-} & r_{0}(1) \\ r_{-} & r_{0}(1) \\ r_{-} & r_{0}(1) \\ r_{-} & r_{0}(0) \\ r_{-} & r_{0}(0) \\ r_{-} & r_{0}(0) \\ r_{-} & r_{0} \\ r_{-} & r_{0}(1) \\ r_{-} & r_{0}(0) \\ r_{0}$	0FE01000 0FE01000 000FF000 0FF0000 FF00FFFE 000F0001 FFFFFFF 000000FF 00000000 00000000	07E01000 07E01000 007F0000 07F00000 FF00FF0 007F00001 FF0FFFF 0.00000FF 0.00000F 0.00000F 0.000000 0000000 0000000 2 1 00000000 2 1 00000000	000 11 38 200FF000 1				Values	ll exec at 00 (ute Ib)4 and χ ₁₅) Ih and 1 Oc in	to r14	- r16 -16		<u>)o</u>		
$\begin{array}{cccc} r_{0}(13) \\ r_{-} & r_{0}(12) \\ r_{-} & r_{0}(11) \\ r_{-} & r_{0}(0) \\ r_{-} & r_{0}(1) \\ r_{-} & $	0FE01000 0FE01000 000FF0000 0FF00000 FF00FFFE 00FF0001 FFFFFFF 000000FF 000000FF 00000000	0FE01000 0FE01000 0FF0000 0FF00000 0FF00000 0FF00000 0FF00000 000000	000 11 18 000FF000 1 1200)x00000)XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	54 the	Values	ll exec at 00 ((15) (15)	1 Oc in	to r14	- r16 -16 00000000 002)00)0)00400	064	
$\begin{array}{c} \mathbf{C} & \rightarrow & ra(13) \\ \mathbf{C} & \rightarrow & ra(12) \\ \mathbf{C} & \rightarrow & ra(11) \\ \mathbf{C} & \rightarrow & ra(10) \\ \mathbf{C} & \rightarrow & ra(9) \\ \mathbf{C} & \rightarrow & ra(9) \\ \mathbf{C} & \rightarrow & ra(8) \\ \mathbf{C} & \rightarrow & ra(6) \\ \mathbf{C} & \rightarrow & ra(2) \\ \mathbf{C} & \rightarrow & ra(2) \\ \mathbf{C} & \rightarrow & ra(1) \\ $	0FE01000 0FE01000 000FF000 0FF0000 FF00FFFE 000F0001 FFFFFFF 000000FF 00000000 00000000	07E01000 07E01000 007F0000 07F00000 FF00FF0 007F00001 FFFFFFF 000000FF 0. X00000 000000F 0. X00000 0000000 0000000 2 1 00000000 2 1 00000000	000 11 38 200FF000 1		XXXX0001		Values	ll exec at 00 (ute Ib)4 and λ15	1 Oc in	to r14	- r16 -16		<u>)o</u>	064	
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	0FE01000 0FE01000 000FF000 0FF0000 FF00FFFE 000F0001 FFFFFFF 000000FF 00000000 00000000	07E01000 07E01000 00FF0000 FF00FF0001 FF0FFFF 000F0001 FF0FFFF 000000F 000000F 0000000 0000000 0000000 2 1 0000000 0000000 0000000 04 0000000 04 0000000 04 0000000 04	000 11 18 000FF000 1 1200)x00000		Xxxxx0001		Values		(15) (15)	1 Oc in	to r14	- r16 -16 00000000 002)00)0)00400	064	
C → ra(13) C → ra(12) C → ra(10) C → ra(9) C → ra(9) C → ra(9) C → ra(6) C → ra(5) C → ra(5) C → ra(7) C → ra(2) C → ra(2) C → ra(2) C → ra(1) C → ra(1) C → ra(2) C → ra(0FE01000 0FE01000 000FF000 00FF0000 FF00FFE 00FF0001 FFFFFFF 000000FF 000000FF 0000000 000000	07E01000 07E01000 00FF0000 FF00FF0001 FF0FFFF 000F0001 FF0FFFF 000000F 000000F 0000000 0000000 0000000 2 1 0000000 0000000 0000000 04 0000000 04 0000000 04 0000000 04	11 38 2000				Values X14 X000000 X000000 X00		(15) (15)	1 Oc in	to r14	- r16 -16 00000000 002)01)0)00400		

060 : 08000070; 070 : 0C000078; 078 : 34310084; 07C : 02200008;



084 : 10220008; 088 : 10210008; 094 : 14210008; 098 : 14220008;

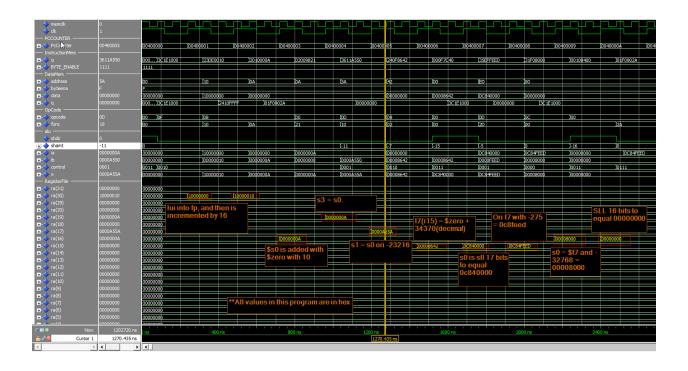
												_				
memclk	0							L				ĽП				
🔶 dk	0				L										μ.	
- PCCOUNTER																
	00000098	0000007C 00	000084	1000	00088		0000009	4	00000	0098		0000	000A4		00000	0A8
 InstructionMem 																
	14230008	02200008	10230008	}	1021	0008	11	4210008		<u>)142</u>	30008		/1000	00000		-
⊢� BYTE_ENABLE − DataMem	1111	1111														+
- Datamem	FF	84)FF		10						FF	/	84			
		0 1			0	0						_/-	104			+
⊢ data	, 00000001	00000000	00000001		ľo	0000000				=	000000	101	Yooo	00084		⊨
	20018FFF	10230008		0000000)20018F						000000	1000	11023	0009	+
– OpCode –––––	20010111			000000		120010					- 17	00000			0008	╞
⊢<>> opcode	05	00	04)(0	5					00			
⊢◆ func	08	08					10				/)00			
- RegisterFile		_											00			
A (24)	00000074	4														
At 084, we ch	eck BEQ betwe	en 🚰								7						
registers r1 a	nd r3. Since the	ey are 🔚										_				
not equal, it w	nd r3. Since the ill not branch to	o / 🔚			Fina	lly, at	94, BI	NEise	xecut	ed to	o che	eck	F			⊨
address 94. a	t 088, it checks	BEQ			betw	een r	1 and i	r1. Sin	ce the	y ar	e eqi	ualit	E			⊨
of r1 to itself.	since it is equa 4. The displace	lit 🛓			will r	notbra	anch to	1. Sin Da0 si	nce th	e dis	splao	cemei	nt 🗏			+
branches to 9	4 The displace	ement			is 8.	At 098	8,wee	execut	e BNE	aga	ain b	etwee	en 🗏			⊨
is 8 from the c	current pc coun	ter			r1 an	id r3. 9	Since	they a	re not	equ	al, w	e wil	I			⊨
is 8 from the o	current pc coun	iter.			r1 an bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ al to	al, w) A4,	ve wil , and v	I we∣			
- is 8 from the c - 88+8 = 94.	current pc cour				r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ al to	al, w o A4,	ve wil , and v	l we			
- is 8 from the c - 88+8 = 94. -	current pc cour	iter. <u>1</u> 10			r1 ar bran	id r3. 9 ch to 9	Since 98+8, ogran	they a which	re not is equ	equ ial to	al, w o A4,	re wil , and v	l we			
- is 8 from the c - 88+8 = 94. - -> ra(12)		1 1 0 0FE0 1000			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ ial to	al, w) A4,	re will , and v	l we			
- is 8 from the c 88+8 = 94. - ✓ ra(12) - ✓ ra(11)	OFE0 1000 000FF000	ter. 1 1 0 0FE0 1000 000FF000			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ ial to	al, w) A4,	re will , and v	l we			
- is 8 from the c - 88+8 = 94. 	0FE01000 00FF000 0FF0000	0FE01000 00FF000 0FF0000			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ ial to	al, w > A4,	ie will , and v	l we			
is 8 from the c 88+8 = 94. ✓ ra(12) ✓ ra(11) ✓ ra(10) ✓ ra(9)	0FE01000 00FF0000 0FF00000 FF00FFFE	0FE01000 000FF000 0FF0000 FF00FFFE			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ al to	al, w A4,	ie will , and v	l we			
- is 8 from the c - 88+8 = 94. - → ra(12) - → ra(10) - → ra(9) - → ra(8)	0FE01000 00FF000 0FF0000	0FE01000 000FF000 0FF0000 0FF00000 FF00FFFE 00FF0001			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ al to	al, w) A4,	re will , and v	l we			
is 8 from the c 88+8 = 94. → ra(12) → ra(11) → ra(0) → ra(8) → ra(7)	0FE01000 000FF000 0FF0000 0FF00000 FF00FFFE 00FF0001	Iter. I 0 0 00FE01000 000FF0000 0FF00000 0FF00000 FF00FFFE 00FF00001 FFFFFFFF 00FF00001			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ al to	al, w) A4,	re will	l we			
is 8 from the c 88+8 = 94. → ra(12) → ra(11) → ra(10) → ra(9) → ra(8) → ra(7) → ra(6)	0FE01000 000FF000 0FF00000 FF00FFFE 00FF0001 FFFFFFFF	Image: 1 Image: 1			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ al to	al, w A4,	re will	l we			
is 8 from the c 88+8 = 94. → ra(12) → ra(10) → ra(9) → ra(6) → ra(6) → ra(5)	0FE01000 000FF000 0FF0000 0FF00000 FF00FFFE 00FF0001 FFFFFFFF 000000FF	OFE01000 00FF000 0FF00000 FF00FFFE 00FF0001 FFFFFFF 000000FF 00000FF			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ ial to	al, w) A4,	re will				
- is 8 from the c - 88+8 = 94. - √ ra(12) - √ ra(10) - √ ra(9) - √ ra(8) - √ ra(6) - √ ra(6) - √ ra(6) - √ ra(4)	0FE01000 000FF000 0FF00000 FF00FFE 00FF0001 FFFFFFF 000000FF 000000FF	OFE01000 000FF0000 0FF00FFE 00FF0001 FF0FFFF 00FF000FF 0000F000FF 0000000F 00000000F 0000000			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ al to	al, w >A4,	re will, and v				
is 8 from the c 88+8 = 94. ✓ ra(12) ✓ ra(10) ✓ ra(9) ✓ ra(7) ✓ ra(6) ✓ ra(5) ✓ ra(3)	0FE01000 000FF000 0FF00000 FF00FFFE 00FF0001 FFFFFFFF 000000FF 000000FF 000000FF	OFE01000 000FF000 000FF000 0FF00000 FF00FFFE 00FF0001 FFFFFFF 000000FF 000000F 00000000 00000000			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equ al to	al, w) A4,	re will, and v				
- is 8 from the c - 88+8 = 94. - 7 ra(12) - 7 ra(10) - 7 ra(9) - 7 ra(8) - 7 ra(6) - 7 ra(5) - 7 ra(4) - 7 ra(3) - 7 ra(2)	0FE01000 000FF000 0FF00000 FF00FFE 00FF0001 FFFFFFF 000000FF 000000FF 0000000F 000000	OFE01000 0000FF000 0000FF000 0FF00000 FF00FFFE 0000000 0000000 0000000 0000000 0000000 0000000 0000000 0000000			r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ		al, w) A4,	ve will , and v				
- is 8 from the c - 88+8 = 94. - 7 ra(12) - 7 ra(10) - 7 ra(9) - 7 ra(8) - 7 ra(6) - 7 ra(6) - 7 ra(7) - 7 ra(1) - 7 ra(2) - 7 ra(1)	0FE01000 000FF000 0FF0000 FF00FFE 00FF0001 FFFFFFF 000000FF 0000000FF 00000000	OFE01000 00FE01000 00FF000 0FF00000 0FF00000 FF00FFFE 000000FF 000000F 00000000 00000000 00000000 00000000 00000000 00000000 00000000	000684		r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ		al, w) A4,	ve will , and v				
is 8 from the c 88+8 = 94. - ra(12) - - ra(10) - - ra(9) - <	0FE01000 00FF000 0FF0000 FF00FFE 00FF001 FFFFFFF 000000FF 000000FF 0000000FF 000000	OFE01000 00FF0000 0FF0000 FF0FFFE 000000F 0000000 0000000 0000000 0000000 0000000 0000000 0000000 00000000 00000000 00000000 00000000 00000000	000084		r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ		al, w) A4,	re will , and y				
- is 8 from the c - 88+8 = 94. - 74 ra(12) - 74 ra(10) - 74 ra(10) - 74 ra(9) - 74 ra(9) - 74 ra(7) - 74	0FE01000 00FF000 0FF0000 FF00FFE 00FF001 FFFFFFF 000000FF 000000FF 0000000FF 000000	OFE01000 00FE01000 00FF000 0FF00000 0FF00000 FF00FFFE 000000FF 000000F 00000000 00000000 00000000 00000000 00000000 00000000 00000000			r1 an bran end	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ		al, w A4,	ve will , and y				
- is 8 from the c - 88+8 = 94. - 74 ra(12) - 74 ra(10) - 74 ra(10) - 74 ra(9) - 74 ra(9) - 74 ra(9) - 74 ra(7) - 74 ra(6) - 74 ra(6) - 74 ra(3) - 74 ra(1) - 74 ra(1) - 74 ra(0) - 74 ra(0) - 74 ra(1) - 74 ra(0) - 74 ra(1) - 74 ra(0) - 74 ra(1) - 74 ra(0) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(1) - 74 ra(1) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 75	OFE01000 00FF000 0FF00FF000 0FF00FFE 00FF0001 FFFFFFFF 000000FF 0000000F 00000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	OFE01000 00FE01000 00FF0000 0FF00FFE 00FF000F 0000000F 0000000 0000000 0000000 0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	1		r1 ar bran	id r3. 9 ch to 9	Since 98+8,	they a which	re not is equ	equi	al, w	re will , and y		00084		
- is 8 from the c - 88+8 = 94. - 74 ra(12) - 74 ra(10) - 74 ra(10) - 74 ra(9) - 74 ra(9) - 74 ra(9) - 74 ra(7) - 74 ra(6) - 74 ra(6) - 74 ra(3) - 74 ra(1) - 74 ra(1) - 74 ra(0) - 74 ra(0) - 74 ra(1) - 74 ra(0) - 74 ra(1) - 74 ra(0) - 74 ra(1) - 74 ra(0) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(1) - 74 ra(1) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 74 ra(1) - 74 ra(2) - 74 ra(1) - 75	0FE01000 00FF000 0FF0000 FF00FFE 00FF001 FFFFFFF 000000FF 000000FF 0000000FF 000000	OFE01000 00FF0000 0FF0000 FF0FFFE 000000F 0000000 0000000 0000000 0000000 0000000 0000000 0000000 00000000 00000000 00000000 00000000 00000000)	rl an bran end t	id r3. 9 ch to 9	Since 98+8, ogram	they a which	re not is equ		al, w) A4,	, and v		00084		
- is 8 from the c + 88+8 = 94. - √ ra(12) - √ ra(10) - √ ra(10) - √ ra(8) - √ ra(6) - √ ra(6) - √ ra(6) - √ ra(7) - √ ra(6) - √ ra(1) - √ ra(1) - √ ra(0) - √ ra(1) - √ ra(2) - √ ra(1) - √ ra(1) - √ ra(1) - √ ra(1) - √ ra(2) - √ ra(1) - √ ra(OFE01000 00FE01000 00FF0000 0FF00000 FF00FFFE 00FF0001 FFFFFFFF 0000000FF 000000001 000000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000	OFE01000 0000FF000 0000FF000 0FF00000 FF00FFFE 000F0000 0000000 0000000 0000000 0000000 0000000 00000000 00000000 00000000 00000000 00000000 00000000 000000000 000000000 000000000 0000000000	1 3 00000000)	rl an bran end t	ad r3. S ch to S the pr	Since 98+8, ogram	they a which	re not is equ		al, w A4,					

LAB4_TEST.MIF :

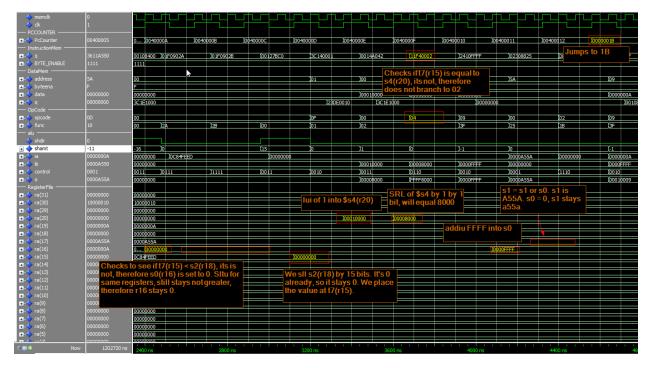
For this program, I will only explain the control flow of the program. I will also annotate the "addiu" instruction as I did not annotate that instruction in the previous simulations of lab4demo and lab4baotung.

∻	Msgs																
I memclk	0																
🧄 dk										Fi	Γ -						
PCCOUNTER							Γ							_			
	00400005	00400000	00400001		00400002		004000	03	004	100004		00400	005	0040000	6	00400007	
InstructionMem																	
 > q	3611A550	000)3C1E1000	230	E0010	201	0000A		02009821		3611	A550		240F8642)0	00F7C40)35F	FFEED
		1111										_					
DataMem																	
address		00	(10		(0A			0A)5A			42)0	0)ED	
💶 🔶 byteena		F															
💶 🔶 data		0000000	(100	00000	000	00000							00000000)0	0008642	000	840000
. 	0000000	000)3C1E1000		2410FF	F	01F0	902A				000000	00			3C1E1	000	00000
— OpCode ———	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~																
🖅 🔶 opcode		(00)(0F) 08					00) <mark>0D</mark>			09)(0	<u>)</u> OD	
🖅 🔶 func		00	<u>)</u> 10		(OA			21		(10/			02)0	0	(2D	
— alu ———																	
🔶 shdir																	
主 🔷 shamt	-11	0)-11			<u>)</u> -7	χ-	15)-5	
主 🔶 ia	000000A	0000000	(100	00000	000	00000		0000000					00000000			300	840000
	0000A550	0000000	000	00010	000	0000A		00000000	/	0000	A550		00008642)(0008642)000	OFEED
		0011 0010								0001			0010)0	011	000	1
⊥ ♦ 0		0000000	(100	00010	000	0000A		00000004		0000	A55A		00008642)0	C840000	300	84FEED
RegisterFile																	
		0000000					,										
		0000000	10000000		10000010												
. ≖ –� ra(29)		0000000															
. ≖ –� ra(20)		0000000					d.										
		0000000)000	A0000							
💶 🔶 ra(18)	0000000	0000000															
💶 🔷 ra(17)	0000A55A	0000000				/						0000A					
		0000000			The	addiui	instru	ction is the ze r t7, or	s exec	cuted a	atx40	0005				كصع	
		0000000			8642	is ad	ded to	the ze	ro ve	ctor ar	nd the	n		0000864	2	0C840000	
		0000000			place	ed in r	egiste	rt7, or	regis	ter 15	. Add I	nas t	he			كمعيه	
		0000000			contr	ol sia	nalof	001 [′] 0 iı	n the a	alu.						لتسريه	کھی
	0000000	0000000														كتعري	کھ
		0000000														لتسوع	<u>کست</u>
	0000000	0000000														لتسوع	
		0000000														لتسرع	
	0000000	0000000														لتسوع	کسی
		0000000														لتسوير	کسے:
	0000000	0000000														لتصعد	کھے
	0000000	0000000														كسع	کھ
	0000000	0000000							1.1								
A 📰 💿 📃 Now	1202720 ns) ns		400 ns			800) ns			120	0 ns			1600 ns		

[0x000000] [0x000001] [0x000002] [0x000003] [0x000004] [0x000005] [0x000006] [0x000007] [0x000008]	0x3C1E1000 0x23DE0010 0x2010000A 0x02009821 0x3611A550 0x240F8642 0x000F7C40 0x35EFFEED 0x31F08000	<pre># lui \$fp, 4096 (\$fp = 4096 << 16) # addi \$fp, \$fp, 16 (\$fp = \$fp + 16) # addi \$s0, \$zero, 10 (\$s0 = 10) # addu \$s3, \$s0, \$zero (\$s3 = \$s0) # ori \$s1, \$s0, -23216 (\$s1 = \$s0 -23216) # addiu \$t7, \$zero, 34370 (\$t7 = 34370) # s11 \$t7, \$t7, 17 (\$t7 = \$t7 << 17) # ori \$t7, \$t7, -275 (\$t7 = \$t7 -275) # andi \$s0, \$t7, -32768 (\$s0 = \$t7 & -32768)</pre>
[0x000007] [0x000008] [0x000009]		# ofi \$t/, \$t/, -275 (\$t/ = \$t/ -275) # andi \$s0, \$t7, -32768 (\$s0 = \$t7 & -32768) # sll \$s0, \$s0, 16 (\$s0 = \$s0 << 16)



s0) $\$s2 = 1$
\$s0) \$s2 = 1
15)
L)
4) goto 2)
65535)
\$s0)
1



[0x000013] [0x000014] [0x000015] [0x000016] [0x000017] [0x000018] [0x000019] [0x00001a] [0x00001b]	0x02602820 0x02202020 0x0C00001E 0xAFC20000 0xAFC30004 0x23DE0004 0x8FD10000 0x23DE0004 0x2673FFFF	<pre># add \$a1, \$s3, \$zero (\$a1 = \$s3) # add \$a0, \$s1, \$zero (\$a0 = \$s1) # jal 0x001E (jump & link to addr 0x0078) # sw \$v0, 0(\$fp) (mem[\$fp + 0] = \$v0) # sw \$v1, 4(\$fp) (mem[\$fp + 4] = \$v1) # addi \$fp, \$fp, 4 (\$fp = \$fp + 4) # lw \$s1, 0(\$fp) (\$s1 = mem[\$fp + 0]) # addi \$fp, \$fp, 4 (\$fp = \$fp + 4) # addi \$fp, \$fp, 4 (\$fp = \$fp + 4) # addi \$fp, \$fp, 4 (\$fp = \$fp + 4) # addiu \$s3, \$s3, 65535 (\$s3 = \$s3 + 65535)</pre>
[0x00001c]	0x1660FFF6	# bne \$zero, \$s3, -10 (if (\$zero != \$s3) goto -
10)		
[0x00001d]	0x0800001D	# j 0x001D (jump to addr 0x0074)
[0x00001e]	0x00851022	# sub \$v0, \$a0, \$a1 (\$v0 = \$a0 - \$a1)
[0x00001f]	0x00851827	# nor \$v1, \$a0, \$a1 (\$v1 = ?(\$a0 \$a1))
[0x000020]	0x03E00008	# jr \$ra (jump \$ra)

This program will infinitely loop forever. What happens is that it will continually jump back to x13 because it will check is s3(r19) is equal to 0. S3 will never equal to 0, because we are constantly incrementing it by 65535. It will go back to 13, execute until it JAL to x1E. It will execute to 20, where it will jump back to the address at the return address register which is x16. At x16, it will execute to 1c, where it executes the BNE instruction to check if s3 is equal to 0 again, and then jump back to x13 again.

The annotations of the rest of the program flow are on the following page.

> memclk	0	lsgs								F r																
dk	1					1	+																		╤╼╧	
CCOUNTER -																										
PcCounter	00400005	0040001	2)000	000 1B		000000	1¢)0000	0013	()	00000014		00000	15	<u>)</u> (000	000 1E		0000001	-	0000	0020)(0000016		00000	Q17,
> q	3611A550	0810001	8	2673	FFFF		1660FFF6		02602	820	022	02020		0C1000	al to 1	le, 16 i	s)0	0851827		03E000	08)AF(20000		AF
BYTE_ENABLE	1111	11111													placed	d in r31										
ataMem						c	iecks t	o see	ifs3 i	s _																
address	5A	5A		09		e	qual to	\$zero	its n	ot 📥		5A				51)A	4				(10			14
byteena	F	-	_	V	_	= s	brand	hesb	ack-1	0																
🕨 data	00000000	0000000		0000		-		ick to								00010	009				1000	00000)FFF	FA551		
≥ q	00000000	0000000	0		00108	400		×				(0000	0000)FFFF	A551	-
OpCode	~			Vac		<u> </u>			Vac							Van										
> opcode > func	0D 10	02 18)09)3F		⊨=K	05) <u>00</u>)20					03 1E) <u>00</u>)22		32	_		600		(28			104
iu	10	16		<u></u>		<u>+ ť</u>	30		120					<u>,1E</u>		122		12			08		,00			-104
shdir	0																									
shamt	-11	0)-1)o																	
ia	A0000000	0000000	0)(0000	0004	і 	00010009		19			0000A55		0000000)0000/	554				loor	00016	Ý100	00010		
ib	0000A550	0000000		10000			00000000					(COLONIA DE		0000000		000010						00000		FA551		
control	0001	1110	×	0010			0110		0010					11110		0110		Yı	100		11110	00000	(00)			
0	0000A55A	0000A55	A	0001			00010009		10000			0000A55	A			FFFFA	551		FE5AA4					FA561		
RegisterFile		_																								
ra(31)	00000000	0000000	0)(OOC	00016										
ra(30)	10000010	1000001	0																							í E
ra(29)	00000000	0000000	0																							
> ra(20)	00000000	0000800	0																							
> ra(19)	A0000000	0000000	A			000100	9																			
> ra(18)	00000000	0000000			-		FFOF																			
> ra(17)	0000A55A	0000A55	A		s3 =	S3 +t	65535																			i.
> ra(16)	A0000000	0000FFF			ī																					
ra(15)	00000000	0000000																								
ra(14)	00000000	0000000																								4
ra(13)	00000000	0000000																								
ra(12)	00000000	0000000																								
ra(11)	00000000	0000000																								
ra(10)	00000000	0000000																								
ra(9)	00000000	0000000							_	-							_									
ra(8)	00000000	0000000								a1(r5)) = s3(r	-19)				he valu										
ra(7)	00000000	0000000														ottom,	but i									
ra(6)	00000000	0000000									00010009		is c	learly	A55A											
ra(5)	00000000	0000000									00010009		Yooooa													
•	Now 1202720	ns	1.1.1	1.1	1	0 ns	1.1	1.1.1	1.1	00 ns	1.1		1	0 ns	1.1	1.1.1	1	0 ns	1.1	1.1	1.1	lıı Ons	1.1		1.1	 00 n
	Cursor 1 1270.435				480	iu ns			520	JU NS			560	U ns			601	JU NS			640	U NS			680	101

nemclk clk	0													
- PCCOUNTER														
PcCounter	00400005	00000)0000001E	(0000001F	00000020	00000016	00000017	00000018	00000019)(0000001A	(0000001B)00000	01C	00000013
- InstructionMem					10									
. ⊞- ∲ d	3611A550	0C10001E 008510	22 <u>)0</u> 6 JR 3	Gra, jumps back to	16 2000	0 (AFC300	04)2306	E0004 (8F	D10000	23DE0004)267	SFFFF	1660FFF6	02602820
BYTE_ENABLE		1111												
DataMem												BNE, c	necks if s	3 == 0,
	5A	5A 151	JA4		<u>) (10</u>	14)14	Ű14		118)08	since it	s not goe	s back to 📃
byteena	F	F.										13		
💽 🔶 data	00000000	00000000 000100	09	(0000000	FFFFASS		5AA4 ()1000	0010 (00	00A55A)10000014	(000			
. ⊞- 今 d	00000000	0000000			F	FFFA551	FFE5AA4			230	0004	31F08000	00000	0000
- OpCode				SW	\$v0(r2) at	10 sw	\$v1(r3) at 1	4		Incro	ases fp by	Apapin		
	0D	03 00	/								ases ip by		05	00
· ⊞	10	1E 22	27	<u>(08</u>		4		Ĭ00					36	20
RegisterFile														
💽 🔶 ra(31)	0000000	00000 00000016												
💽	10000010	10000010						[10000014			(10000018			
	00000000	0000000						fo in or o o	bod					<u>ک کا ک</u>
	00000000	00008000						ip increas	seu					
	A0000000	00010009						Dy 4				00020	008	
	0000000	0000000												
•	0000A55A	0000A55A							X	FFESAA4		Adds	65535	
• va(16)	A000000A	0000FFFF							Iw	rinto s1(r17) ad	deci	mal) to st	3(r19)
•	00000000	0000000							da	atamemory				
ra(14) ra(14)	00000000	0000000							w	hich contair	sthe =			
ra(13) (13)	00000000	0000000								lue fffe5aa4				
	0000000	0000000								nuo moouu	· _			
ra(11) ra(11)	0000000	0000000												
• ra(10)	0000000	0000000												
•	00000000	0000000					TT	his program vill never bec	will infin	itely loop ba	ick to 13 b	because s3	(r19)	
•	00000000	0000000					w w	vill never bec	ome 0. It	will continu	e to go fre	om x13, ial	to 1e.	
	00000000	ooooooo subtrac	ts ao(r4) and	nor a0(r4) and	l a1(r5) to		i	ump back to t	he ra whi	ich contains	x16 Atx	16 it will n	in i	<u> </u>
	0000000	00000000 a1(r5).	a55a - 10009 =	and places it i				hrough 1c, an						
	0000000	00010009 ffffa551	intov0(r2)	FFFE5AA4				mough ro, u	u uion ju	imp buok to	rouguin			
	0000000	00004554												
•	0000000	00000000		FFFE5AA4										
•	00000000	0000000	JFFFFA551											
	00000000	0000000			foren e a a a									
	00000000	00000000			FFFESAA4						- V			
0m 🔶 🔲	10	0 4		131)30		Vere			100	<u>(19</u>			
⊞ -⇒ m1	1/	16 15		10	2	13)30	X17)30	(19		0	
Alate Now	1202720.5 ns		6000 ns	6400 ns	100000	6800 ns		7200 ns		7600 ns	1000	80	10 ns	8

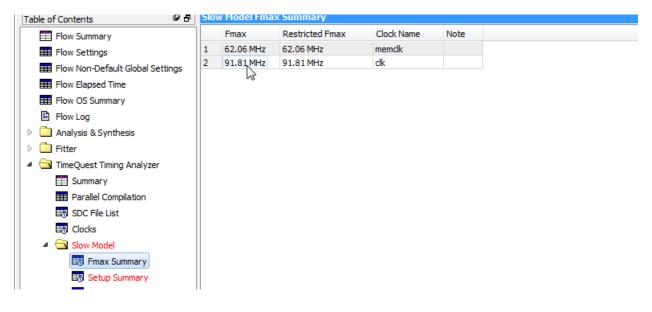
I had a problem with the timing report on Quartus for the slow model, showing that my regular clock was up to around 92 MHz, and my memclk only being around 48 Mhz. Aside from that, logically, each component that I would add would obviously decrease the max frequency of the process. I believe that the lw, lbu, and lhu would decrease the fmax the most because of the access to memory. There are many things you could do speed up your design though. The other instructions only needed extra control signals, which should not have decreased the speed as much since it is mostly sequential logic. I believe only the instructions that require us to access and load from memory will increase the critical path.

As you can see below, the longest delay path is 11.030 ns, which is ironically from the instruction memory and not the data memory.

Flow Summary	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
Flow Settings	1 -23.813	alu32:inst13[result[13]	PC:inst30[tempQ[15]	InstructionMem:inst laltsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0	dk	0.500	-13.323	11.030
Flow Non-Default Global Settings	2 -22.948	alu32:inst13[result[13]	PC:inst30[tempQ[15]	InstructionMem:instialtsyncram:altsyncram compauto_generated/ram_block1a0~porta_address_reg0		1.000	-12.958	11.030
	3 -23.763	alu32:inst13[result[0]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0	dk	0.500	-13.297	11.006
Flow Elapsed Time	4 -22.898	alu32:inst13[result[0]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0	ck	1.000	-12.932	11.006
Flow OS Summary	5 -23.936	alu32:inst13[result[14]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0	ck	0.500	-13.475	11.001
Flow Log	6 -23.071	alu32:inst13 result[14]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0	dk	1.000	-13.110	11.001
Analysis & Synthesis	7 -23.797	alu32:inst13 result[4]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0	ck	0.500	-13.349	10.988
Fitter	8 -22.932	alu32:inst13 result[4]	PC:inst30 tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0	dk	1.000	-12.984	10.988
Assembler	9 -23.799	alu32:inst13[result[10]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0	dk	0.500	-13.393	10.946
	10 -22.934	alu32:inst13[result[10]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated[ram_block1a0~porta_address_reg0	dk	1.000	-13.028	10.946
TimeQuest Timing Analyzer	11 -23.767	alu32:inst13[result[8]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0	ck	0.500	-13.392	10.915
E Summary	12 -22.902	alu32:inst13[result[8]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0	ck	1.000	-13.027	10.915
Parallel Compilation	13 -23.628	alu32:inst13[result[15]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0	clk	0.500	-13.322	10.846
Clocks	14 -22.763	alu32:inst13 result[15]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0	ck	1.000	-12.957	10.846
4 Slow Model	15 -23.607	alu32:inst13 result[11]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.390	10.757
Emax Summary	16 -22.742	alu32:inst13 result[11]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		1.000	-13.025	10.757
	17 -23,508	alu32:inst13 result[20]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.372	10.676
Setup Summary	18 -22.643	alu32:inst13 result[20]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		1.000	-13.007	10.676
Hold Summary	19 -23,448	alu32:inst13[result[12]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0		0.500	-13.324	10.664
Recovery Summary	20 -22.583	alu32:inst13[result[12]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0		1.000	-12.959	10.664
Removal Summary	21 -23,434	alu32:inst13[result[6]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0		0.500	-13.358	10.616
Minimum Pulse Width Summa	22 -22.569	alu32:inst13 result[6]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0	ck	1.000	-12.993	10.616
Worst-Case Timing Paths	23 -23,420	alu32:inst13 result[5]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.359	10.601
	24 -22.555	alu32:inst13 result[5]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		1.000	-12.994	10.601
Datasheet Report	25 -23,435	alu32:inst13 result[9]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.391	10.584
Fast Model	26 -22.570	alu32:inst13[result[9]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		1.000	-13.026	10.584
Multicorner Timing Analysis Sumr	27 -23.321	alu32:inst13[result[1]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0		0.500	-13.326	10.535
Multicorner Datasheet Report Su	28 -22.456	alu32:inst13[result[1]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0		1.000	-12.961	10.535
Clock Transfers	29 -23.274	alu32:inst13[result[21]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0		0.500	-13.355	10.459
Report TCCS	30 -22,409	alu32:inst13 result[21]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		1.000	-12.990	10.459
	31 -23,261	alu32:inst13 result[7]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.357	10.444
Report RSKM	32 -23, 187	alu32:inst13 result[22]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.283	10.444
Unconstrained Paths	33 -22.396	alu32:inst13[result[7]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		1.000	-12.992	10.444
Messages	34 -22.322	alu32:inst13 result[22]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		1.000	-12.918	10.444
EDA Netlist Writer	35 -23.041	alu32:inst13[result[19]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.290	10.291
	36 -22,176	alu32:inst13[result[19]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0		1.000	-12.925	10.291
	37 -22.905	alu32:inst13[result[3]	PC:inst30[tempQ[15]	InstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a0~porta_address_reg0		0.500	-13.359	10.086
	38 -22,798	alu32:inst13[result[13]	PC:inst30[tempQ[26]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.273	10.065
	39 -22.797	alu32:inst13[result[13]	PC:inst30[tempQ[29]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.273	10.064
	40 -22.748	alu32:inst13 result[0]	PC:inst30 tempQ[26]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.247	10.041
	41 -22.747	alu32:inst13 result[0]	PC:inst30 tempQ[29]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.247	10.040
	42 -22.921 43 -22.920	alu32:inst13 result[14]	PC:inst30 tempQ[26]	InstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a0~porta_address_reg0		0.500	-13.425	10.036
		alu32:inst13[result[14]	PC:inst30[tempQ[29]	InstructionMem:inst altsyncram:altsyncram_compauto_generated arm_block1a0~porta_address_reg0		0.500	-13.425	10.035
	44 -22.782 45 -22.781	alu32:inst13[result[4]	PC:inst30[tempQ[26]	InstructionMem:inst jalts yncram:alts yncram_compauto_generated iram_block 1a0~porta_address_reg0		0.500	-13.299	10.023
		alu32:inst13[result[4]	PC:inst30[tempQ[29]	InstructionMem:inst jaltsyncram:altsyncram_compauto_generated iram_block1a0~porta_address_reg0		0.500	-13.299	10.022
		alu32:inst13[result[18]	PC:inst30[tempQ[15]	InstructionMem:inst altsyncram:altsyncram_compauto_generated an_block1a0~porta_address_reg0		0.500	-13.499	9.998
III	47 -22.784		PC:inst30[tempQ[26] PC:inst30[tempQ[29]	InstructionMem:inst altsyncram:altsyncram_compauto_generated iram_block1a0~porta_address_reg0 InstructionMem:inst altsyncram:altsyncram_compauto_generated iram_block1a0~porta_address_reg0		0.500	-13.343	9.981

After adding all the new instructions. I recompiled and got a new delay report, which is found below. It increased the delay by about 5.0ns.

	10 NODE	Launch Clock	Latch Clock	Relationship	CIOCK SKEW	Data Del
	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg7	memclk	memck	1.000	0.022	16.090
InstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg1	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg7	memclk	memclk	1.000	0.022	16.090
instructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg2 = I	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg7	memclk	memclk	1.000	0.022	16.090
instructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg3	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg7	memclk	memclk	1.000	0.022	16.090
instructionMem:inst[altsyncram:altsyncram_computo_generated]ram_block1a22~porta_address_reg4	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg7	memclk	memclk	1.000	0.022	16.090
InstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg5	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg7	memclk	memclk	1.000	0.022	16.090
InstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg6	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg7	memclk	memclk	1.000	0.022	16.090
InstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg7	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg7	memclk	memclk	1.000	0.022	16.090
InstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg0	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memclk	1.000	0.022	16.004
InstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg1	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memclk	1.000	0.022	16.004
InstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg2	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memclk	1.000	0.022	16.004
InstructionMem:inst[altsyncram:altsyncram_computo_generated]ram_block1a22~porta_address_reg3	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memck	1.000	0.022	16.004
InstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg4_1	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memck	1.000	0.022	16.004
InstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg5_1	DataMemory:inst4laltsyncram:altsyncram_componto_generated/ram_block1a16~porta_address_reg6	memclk	memck	1.000	0.022	16.004
	DataMemory:inst4laltsvngram:altsvngram componto generated/ram block1a16~porta address reg6	memclk	memck	1.000	0.022	16.004
	DataMemory:inst4laltsvncram:altsvncram componto generatedIram block1a16~porta address reo6	memck	memck	1.000	0.022	16.004
InstructionMem:Inst laitsyncram:altsyncram compauto generated ram block 1a6~porta address reg0 I	DataMemory:inst4 altsyncram:altsyncram componto generated ram block1a16~porta address reg7	memck	memck	1.000	0.013	15,983
	DataMemory:inst4laltsvngram:altsvngram componto generated/ram block1a16~porta address reg7	memck	memck	1,000	0.013	15,983
	DataMemory:inst4laltsvncram:altsvncram componto_generated/ram_block1a16~porta_address_reg7	memck	memok	1.000	0.013	15,983
	DataMemory:inst4laltsyncram:altsyncram componto generated/ram block1a16~porta address reg7	memck	memck	1.000	0.013	15,983
	DataMemory:inst4laltsyncram:altsyncram componto generated/ram block1a16~porta address reg7	memck	memcik	1.000	0.013	15,983
	DataMemory:inst4laltsvncram:altsvncram componto generated/ram block1a16~porta address reg7	memck	memck	1.000	0.013	15,983
	DataMemory:inst4jaltsyncram:altsyncram componto_generated/ram_block1a16~porta_address_reg7	memck	memck	1.000	0.013	15,983
	DataMemory:inst4laltsvncram:altsvncram_componto_generated/ram_block1a16~porta_address_reg7	memck	memck	1.000	0.013	15,983
	DataMemory:inst4laltsvncram:altsvncram_componto_generated/ram_block1a16~porta_address_reg/	memck	memck	1.000	0.022	15.940
	DataMemory:inst4jaitsyncram:altsyncram componto_genetatediram_block1a16~porta_address_regs	memck	memck	1.000	0.022	15.940
	DataMemory:instellaltsyncram:altsyncram.componto_generated/ram_block1a16~porta_address_reg3	memck	memck	1.000	0.022	15,940
	DataMemory:instellaltsyncram:altsyncram.componto_generated/ram_block1a16~porta_address_rego	memck	memck	1.000	0.022	15.940
				1.000	0.022	15.940
	DataMemory:inst4[altsyncram:altsyncram_componto_generated]ram_block1a16~porta_address_reg3	memck	memck			15.940
	DataMemory:inst4[altsyncram:altsyncram_componto_generated]ram_block1a16~porta_address_reg3	memclk	memck	1.000	0.022	
	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg3	memclk	memck	1.000	0.022	15.940
	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg3	memclk	memck	1.000	0.022	15.940
	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg5	memclk	memclk	1.000	0.022	15.936
	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg5	memck	memck	1.000	0.022	15.936
	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg5	memclk	memclk	1.000	0.022	15.936
	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg5	memclk	memclk	1.000	0.022	15.936
nstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg4	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg5	memclk	memclk	1.000	0.022	15.936
	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg5	memclk	memclk	1.000	0.022	15.936
	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg5	memclk	memclk	1.000	0.022	15.936
nstructionMem:inst altsyncram:altsyncram_computo_generated ram_block1a22~porta_address_reg7	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg5	memclk	memclk	1.000	0.022	15.936
nstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a6~porta_address_reg0	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memclk	1.000	0.013	15.897
nstructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a6~porta_address_reg1	DataMemory:inst4[altsyncram:altsyncram_componto_generated]ram_block1a16~porta_address_reg6	memclk	memclk	1.000	0.013	15.897
nstructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a6~porta_address_reg2	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memck	1.000	0.013	15.897
instructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a6~porta_address_reg3	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memclk	1.000	0.013	15.897
instructionMem:inst[altsyncram:altsyncram_compauto_generated]ram_block1a6~porta_address_reg4	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memclk	1.000	0.013	15.897
instructionMem:inst altsyncram:altsyncram_compauto_generated ram_block1a6~porta_address_reg5	DataMemory:inst4 altsyncram:altsyncram_componto_generated ram_block1a16~porta_address_reg6	memclk	memck	1.000	0.013	15.897
nstructionMem-instilaltsvorram-altsvorram comp. auto. generated Iram block1a6xporta address. reg6	DataManaguingt/Halteringgamialteringgam company to congrated/cam block1a16-points address coof-	and a second sec	memok	1.000	0.013	15 897



I added a lot of multiplexors to my design. What I could have done was look directly at the instruction memory and look at the function code. Some of the instructions change only by one bit, and I could have used that one bit as a control signal to when to active a select line for a multiplexor instead of adding a control signal to my controller or alu32control. This would have greatly reduced the logic from the controller as well sped up the processor as well, as the controller would have no need as much sequential logic.

Controller signals :

```
case opcode is -- rtypes first
   when "000000" => -- this will include ALL R TYPE INSTRCUTIONS besides for JR
   aluop <= "000";
   regwrite <= '1';
   regdst <= '1'; -- changes regdst to write to d, and reads from s and t.
   alusrc <='0'; -- takes in value from rs to alu</pre>
```

when "001000" => -- addi

aluop <= "001"; regwrite <= '1'; regdst <= '0'; -- changes write destination to t. ALUSRC <= '1'; -- takes in immediate value to alu</pre>

```
when "001001" => -- addiu
```

```
zeroorsign <= '1';
aluop <= "001";
regwrite <= '1'; -- enables register file to write
regdst <= '0'; -- changes write destination to t.
ALUSRC <= '1';</pre>
```

```
when "001100" => -- andi
zeroorsign <= '1';
aluop <= "101"; -- tells alucontrol its an i instruction
regwrite <= '1'; -- enables register file to write
regdst <= '0'; -- changes write destination to rt
ALUSRC <= '1'; -- takes in immediate value
```

```
when "000100" => --BEQ
branch <= '1';
BEQ <= '1';
aluop <= "110"; -- subtraction
when "000101" => --BNE
```

```
branch <= '1';
BNE <= '1';
aluop <= "110";
```

```
when "001101" => -- ORI
zeroorsign <= '1';
```

```
aluop <= "010"; -- tells alucontrol its an i instruction
regwrite <= '1'; -- enables register file to write
regdst <= '0'; -- changes write destination to rt
ALUSRC <= '1'; -- takes in immediate value
when "001010" => -- slti
aluop <= "011"; -- tells alu control its an i instruction
regwrite <='1'; -- enables register file to write
regdst <='0';
alusrc <='1'; -- takes in immediate value taht is sign extended.
memtoreg <= '0'; --takes value, either 0 or 1 from aluout, and then will put it into rt.
when "001011" => -- sltiu b hex
zeroorsign <= '1';</pre>
aluop <= "100"; -- tells alu control its an i instruction
regwrite <='1'; -- enables register file to write
regdst <='0'; -- writes to rt
alusrc <='1'; -- takes in immediate value taht is sign extended.
memtoreg <= '0'; --takes value, either 0 or 1 from aluout, and then will put it into rt.
when "100100" => -- lbu
aluop <= "001"; -- immediate instruction
memtoreg \leq 1';
alusrc <= '1';
regwrite <='1';
loadcontrol <= "00"; --chooses mask Ibu
loadormem <= '1';
when "100101" => -- lhu
aluop <= "001"; -- immediate instruction
memtoreg \leq 1';
alusrc <= '1';
regwrite <='1';
loadcontrol <= "01"; -- chooses mask lhu
loadormem <= '1';</pre>
lscontrol <= '1';</pre>
when "100011" => --lw
aluop <= "001"; --immediate instruction
memtoreg <= '1';
alusrc <= '1';
regwrite <='1';
loadcontrol <= "11"; --choses entire rs+signext</pre>
loadormem <= '1';</pre>
lscontrol <= '1';</pre>
```

```
when "101000" => --sb
 --regdst <= '0';
 BYTEENABLE <= "0001";
 aluop <= "001"; --immediate instruction
 memtoreg <= '1';</pre>
 aluop <= "001";
 lscontrol <='1';</pre>
 memwrite <= '1';
when "101001" => --sh
 --regdst <= '0';
 BYTEENABLE <= "0011";
 aluop <= "001"; --immediate instruction
 memtoreg <= '1';
 aluop <= "001";
 lscontrol <='1';</pre>
 memwrite <= '1';
when "101011" => --sW
 --regdst <= '0';
 BYTEENABLE <= "1111";
 aluop <= "001"; --immediate instruction
 memtoreg <= '1';
 aluop <= "001";
 lscontrol <='1';</pre>
 memwrite <= '1';
when "001111" => --lui
 aluop <="001";
 regwrite <= '1';
 loadcontrol <="10";</pre>
 loadormem <= '1';
 lscontrol <='1';</pre>
when "000010" => --jump
 jjal <= '1';
when "000011" => --jal
 jjal <= '1';
 regwrite <= '1';
when others =>
aluop <= "111";
end case;
```

--end if;

Alu32control :

begin

```
process(ALUop, func)
begin
jr <= '0';
 shdir <= '0';
        if (ALUop = "000") then -- R TYPE
                if (func = "100000") then -- add
                        control <= "0010"; -- add
                elsif (func = "100001") then -- addu
                 control <= "0010";
                 shdir <= '0';
                elsif (func = "100100") then -- and func number // func code same as Ibu
                         control <= "0000"; -- and
                elsif (func = "100101") then -- or
                         control <= "0001"; -- or control
                elsif (func = "100111") then -- nor
                         control <= "1100"; -- nor control
                elsif (func = "101010") then -- slt
                         control <= "0111"; -- slt control
                elsif (func = "101011") then -- sltu
                         control <= "1111"; -- sltu control
                elsif (func = "000000") then -- sll
                         control <= "0011"; --sll control
                         shdir <= '1';
                elsif (func = "000010") then-- srl
                 shdir \leq 0';
                         control <= "0011"; -- srl control
                elsif (func = "100010") then --sub
                         control <= "0110"; -- sub control
                elsif (func ="100011") then --subu
```

```
control <= "0110"; -- subucontrol
        elsif (func ="001000") then -- jr
         control <= "1110";
         jr <= '1';
else
control <= "1110"; --VALUE THATS NOT USED
end if;
elsif (ALUop = "001")then -- I TYPE - -- this includes addi, addiu, beq, bne, lbu, lhu, ll, lw,
                                         sb, sc, sh, sw
control <= "0010";
elsif (ALUop = "010") then -- I TYPE --- this includes ori
control <= "0001";
elsif (ALUop = "011") then -- I TYPE -- this includes slti
control <= "0111";
elsif (ALUop = "100") then -- I TYPE -- this includes sltiu
control <= "1111";
elsif (AlUop = "101") then -- I Type - -- this includes ANDI
control <= "0000"; -- aND
elsif (ALUop = "110") then -- subtraction immediate-- this includes JAL
control <= "0110"; -- substraction
elsif (AlUop = "111") then -- operations which outputs 1110,
                          -- which does not doing anything -- this inlcudes JR, LUI, J
control <= "1110"; --VALUE THATS NOT USED
```

end if;