# CprE 381 – Computer Organization and Assembly-Level Programming

# **MIPS Pipelined Processor w/wo Data Dependencies**

Author: Samuel Ferguson

			1
IF/ID Stage	ID/EX Stage	EX/MEM Stage	MEM/WB Stage
Flush	Flush	Flush	Flush
Stall	Stall	Stall	Stall
PC+4	Reg Write	Reg Write	Reg Write
Instruction	MemtoReg	MemtoReg	MemtoReg
	Branch	Branch	DMEM data out
	BranchNE	Mem Write	DMEM address in
	MemWrite	ALUout	Reg Write Address
	RegDst	ALU in B	JAL from Ctrl
	ALUop from Ctrl	Rs	PC + 4
	ALUSrc	Rt	
	Register rs Data	Reg Write Address	
	Register rt Data	JAL from Ctrl	
	Imed	Jump from Ctrl	
	Reg Write Address	PC + 4	
	Rs		
	Rt		
	Jump from Ctrl		
	JAL from Ctrl		
	PC+4		
	Branch Address		

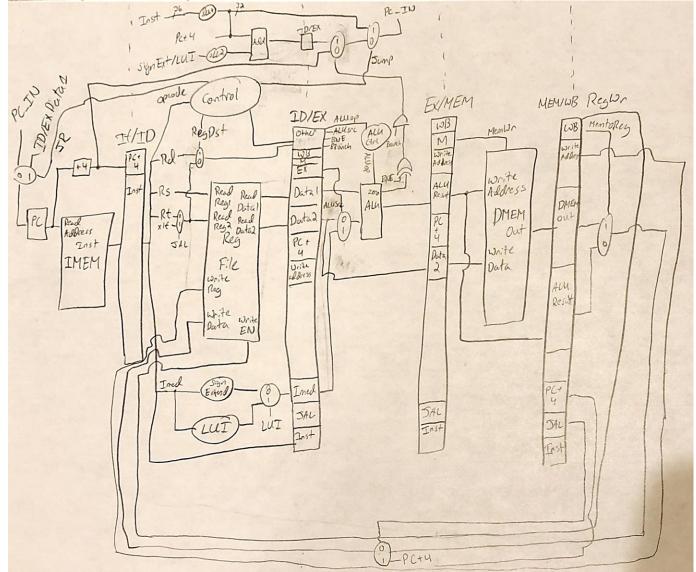
#### **Pipeline Register Signals**

## **Pipelined Register Signals Waveform**

🛿 🔷 if_id_pc_4	32'h0040014C	0000000	<u>100000004</u>	0000008	1000000C	00000010	00000014	00000018	0000001C
) 🔷 if_id_sInst	32'h00000000	00000000	200 1000 1	20020002	20030003	20040004	20050005	20060006	20070007
🛛 🔷 id_ex_read_data1	32'h00000000	00000000							
🛛 🔷 id_ex_read_data2	32'h00000000	00000000							
🗄 🔷 id_ex_inst	32'h00000000	00000000		20010001	20020002	20030003	20040004	20050005	20060006
d_ex_SEI_LUI_data2	32'h00000000	00000000		00000001	00000002	00000003	00000004	00000005	0000006
id_ex_RegWr	1								
id_ex_MemtoReg	0								
id_ex_Branch	0								
id_ex_DMemWr	0								
id_ex_RegDst	1								
id_ex_ALUSrc	0								
s_RegWrite	1								
s_DMemWrite	0								
♦ id_ex_jal	0								
) 🔷 id_ex_ALUop	10'h000	000		209					
👌 🔷 id_ex_RegWrAddr	5'h00	00		01	<b>0</b> 2	(03	(04	05	06
🔷 s_Pre_RegWrAddr	5'h00	00	01	02	(03	<sup>104</sup>	05	106	07
🔷 ex_mem_RegWrAddr	5'h00	(00			01	02	(03	04	05
<pre>ex_mem_RegWr</pre>	1								
<pre> ex_mem_MemtoReg </pre>	0								
<pre>ex_mem_Branch</pre>	0								
<pre> ex_mem_jal </pre>	0								
🛛 🔷 ex_mem_Inst	32'h00000000	00000000			20010001	20020002	20030003	20040004	20050005
) 🔷 id_ex_pc_4	32'h00400148	00000000		00000004	0000008	000000C	00000010	00000014	00000018
<pre> ex_mem_pc_4 </pre>	32'h00400144	00000000			00000004	00000008	1000000C	00000010	00000014
🛛 🔷 mem_wb_DmemOut	32'h00050000	00000000	00000007						00000009
🛛 🔷 mem_wb_ALUout	32'h00000000	0000000				00000001	100000002	00000003	00000004
> mem_wb_pc_4	32'h00400140	00000000				00000004	0000008	0000000C	00000010
mem_wb_MemtoReg	0								
mem_wb_jal	0								

PC + 4 is the only signal that propagates through all four stages of the pipeline because it is needed for the Jump and Link instruction to change the register write data to the PC +4 data when the instruction was executed. As you can see in the waveform, the PC+4 signal propagates from if\_id\_PC\_4 to mem\_wb\_pc\_4 four cycles later. The same is done with the next PC + 4 value of x8 and can be seen directly after each highlighted PC + 4 signal.

This is the schematic of the pipelined MIPS processor. The main challenges I faced with implementing the pipelined processor was pipelining address instructions such as Jump And Link and Branch. The Jump And Link instruction needed to be pipelined through all four registers to that it could store the address to register 31. The Branch instruction was pipelined to the EX stage where the ALU could calculate the Zero. For the assembly level instructions I needed to add three nops after each RAW instruction because that was the amount of registers between the WB stage and the ID stage. For jump and JAL instructions, I needed to use one nop so that the ID stage could jump without loading an instruction into the IF/ID register. I needed three nops after branch instructions because I needed to stop instructions from being loaded into the IF/ID register and ID/EX register and then nop for one more cycle to clear the current PC address. The jump register instruction needed two nops after the instruction to clear the PC value and then stop the IF/ID register from loading in a false instruction.



## Demonstration of Instructions without data dependency handling.

ADDI ADDI \$1, \$0, 1 (RED) ADDI \$2, \$0, 2 (YELLOW)

💶 🔶 s_RegWrData	32'h00000000	00000000			00000001	00000002
💶 🧇 s_RegWrAddr	5'h00	00			01	02
💶 🔶 s_rs	5'h00	00				
<b> </b>	5'h00	01	02	03	04	(05 (
💶 🔶 s_rd	5'h00	00				
💶 🤣 s_Imed	16'h0000	0001	0002	0003	0004	0005
💶 🧇 s_opcode	6'h00	08				
🖅 🎝 if_id_sInst	32'h00000000	20010001	20020002	20030003	20040004	20050005

## ADD & SUB & ADDU & SUBU

Add \$11, \$3, \$4 (RED) #Place 3+4=7 in \$11 Addu \$12, \$5, \$10 (YELLOW) #Place -3 + 5= 2 in \$12 Sub \$13, \$7, \$4 #Place 7-4=3 in \$13 (RED)

#### Subu \$14, \$10, \$5 #Place -3-5=-8 in \$14 (YELLOW)

s_RegWrData	32'h00000000	000	00000			00000007	00000002	00000003	FFFFFF8
s_RegWrAddr	5'h00		00			0B	( OC	OD.	Į0Ε į
s_rs	5'h00		03	05	07	0A	03		
s_rt	5'h00		04	(0A	04	05	07		
s_rd	5'h00		0B	(OC	0D	0E	, OF	10	11
s_Imed	16'h0000		5820	6021	6822	7023	7824	8025	8826
s_opcode	6'h00	00							
▶ if_id_sInst	32'h00000000		00645820	00AA6021	00E46822	01457023	00677824	00678025	00678826

## BNE

Looper:

add \$30, \$30, \$1 (RED) #Add 1 to \$30 until \$30 == 3

nop

nop

nop

bne \$30, \$3, Looper (YELLOW)

nop

nop

nop

## add \$11, \$3, \$4 (BLUE) # Place 7 in \$11

#### Iteration 1

s_RegWrData	32'h00000000	00000001	00000000		FFFFFFE	10000000
s_RegWrAddr	5'h00	1E	<u>100</u>		03	<u>‡00</u>
s_rs	5'h00	00	1E	<u>100</u>	1E	<u>‡00</u>
s_rt	5'h00	00	03	<u>100</u>	01	<u> 100</u>
s_rd	5'h00	00	1F	1 OO	1E	<u>‡00</u>
s_Imed	16'h0000	0000	FFFB	1 0000	F020	<u>‡0000</u>
s_opcode	6'h00	00	05	<u>100</u>		
if_id_sInst	32'h00000000	00000000	17C3FFFB	00000000	03C1F020	10000000

#### Iteration 2

s_RegWrData	32'h00000000	100000002	00000000		FFFFFFF	20000000 2
s_RegWrAddr	5'h00	1E	00		03	100
s_rs	5'h00	00	1E	<u>100</u>	1E	100
s_rt	5'h00	00	03	<u>100</u>	01	100
s_rd	5'h00	00	[1F	<u>100</u>	1E	00
s_Imed	16'h0000	0000	FFFB	X 0000	F020	0000
s_opcode	6'h00	00	05	X 00		
if_id_sInst	32'h00000000	00000000	17C3FFFB	X 0000000	03C1F020	10000000

## Iteration 3

s_RegWrData	32'h00000000	00000003	00000000			
s_RegWrAddr	5'h00	1E	00		03	<u>100</u>
s_rs	5'h00	00	1E	<u> 100</u>		03 )
s_rt	5'h00	00	03	<u>100</u>		04 )
s_rd	5'h00	00	1F	<u>100</u>		0B )
s_Imed	16'h0000	0000	FFFB	<u>10000</u>		5820 )
s_opcode	6'h00	00	05	<u> 100</u>		
if_id_sInst	32'h00000000	00000000	17C3FFFB	X 00000000		00645820

#### AND & OR & XOR & NOR

and \$15, \$3, \$7 (RED) #Place 3 in \$15

or \$16, \$3, \$7 (YELLOW) #Place 7 in \$16

xor \$17, \$3, \$7 (RED) #Place 4 in \$17

#### nor \$18, \$3, \$7 (YELLOW) #Place a -8 in \$18

s_RegWrData	32'h00000000	0000002	0000003	<b>İFFFFFF8</b>	0000003	00000007	00000004	<b>ĮFFFFFF</b> 8
s_RegWrAddr	5'h00	10C	( OD	,ΩE	(OF	10	11	<u>, 12</u>
s_rs	5'h00	03					Į0A	100
s_rt	5'h00	07					03	<u>107</u>
s_rd	5'h00	OF	10	11	12	13	14	15
s_Imed	16'h0000	7824	8025	8826	9027	982A	A02B	Å A880
s_opcode	6'h00	þo						
if_id_sInst	32'h00000000	00677824	00678025	00678826	00679027	0067982A	0143A02B	0007A880

#### SLT & SLTU & SLL & SLV & SRL & SRLV

slt \$19, \$3, \$7 (RED) #Place 1 in \$19

sltu \$20, \$10, \$3 (YELLOW) #Place 0 in \$20

sll \$21, \$7, 2 (RED) #place 28 in \$21

sllv \$22, \$7, \$2 (YELLOW) #place 28 in \$22

srl \$23, \$7, 2 (RED) #Place 1 in \$23

srlv \$24, \$7, \$2 (YELLOW) #place 1 in \$24

s_RegWrData	32'h00000000	00000007	00000004	FFFFFF8	00000001	00000000	0000001C		00000001	
s_RegWrAddr	5'h00	(10	11	12	I 13	14	15	16	17	18
s_rs	5'h00	03	) 0A	00	<u>102</u>	100	.02	<u>, oo</u>		
s_rt	5'h00	07	03	07				03	1E	<u>, oo</u>
s_rd	5'h00	13	14	15	<u>, 16</u>	17	18	19	00	
s_Imed	16'h0000	982A	A02B	A880	<u>, 8004</u>	<b>1</b> B882	C006	<u>C8C3</u>	0000	
s_opcode	6'h00	00							08	<u>, 00</u>
if_id_sInst	32'h00000000	0067982A	0143A02B	0007A880	00478004	10007B882	0047C006	0003C8C3	201E0000	00000000

#### SRA & SRV

sra \$25, \$3, 2 #Sift x3 sra 2 bits to right (RED)

srav \$26, \$3, \$2 #shift x3 sra 2 bits to the right (YELLOW)

s_RegWrData	32'h00000000	000	00001C	00000001		C0000000	
s_RegWrAddr	5'h00		16	17	18	19	1A
s_rs	5'h00		00	02	00		
s_rt	5'h00		03		( 1E	00	
s_rd	5'h00		19	1A	00		
s_Imed	16'h0000		C883	D007	0000		
s_opcode	6'h00	00			08	00	
if_id_sInst	32'h00000000		0003C883	0043D007	201E0000	00000000	

## BEQ & LUI

Looper\_2:

add \$30, \$30, \$1 (RED) #Add 1 to \$30 and branch if \$30 == 3 nop nop beq \$30, \$3, Looper\_2 (YELLOW) #\$30 = 1 so will continue nop nop

lui \$27, 5 (BLUE) #Place x00050000 in \$27

s_RegWrData	32'h00000000	00	00000			0	0000001	0000	0000		FFFFFFE
s_RegWrAddr	5'h00		00			1	E	00			03
s_rs	5'h00		1E	00				1E		00	
s_rt	5'h00		01	00				03		00	
s_rd	5'h00		1E	00				1F		00	
s_Imed	16'h0000		F020	0000				FFFB		0000	
s_opcode	6'h00	00						04		00	
if_id_sInst	32'h00000000		03C1F020	00000000				13C3	FFFB	00000000	
s_RegWrData	32'h00000000		FFFFFFF	E 10000000	00				0005	50000	
s_RegWrAddr	5'h00		103	( 00					1B		
s_rs	5'h00		00			05	03				
s_rt	5'h00		00	1B		0F	10		11		
s_rd	5'h00		00								
s_Imed	16'h0000		0000	0005		0007					
s_opcode	6'h00		00	OF		0C	( OD		(OE		
if_id_sInst	32'h00000000		00000000	3C 1B000	)5	30AF0007	7 3470	0007	3871	10007	

## ANDI & ORI & XORI & SLTI & SLTIU

andi \$15, \$5, (RED) 7 #Place 5 in \$15 ori \$16, \$3, 7 (YELLOW) #Place 7 in \$16 xori \$17, \$3, 7 (RED) #Place 4 in \$17 slti \$19, \$3, 7 (YELLOW) #Place 1 in \$19 sltiu \$20, \$10, 3 (RED) #Place 0 in \$20

s_RegWrData	32'h00000000	00000000		00050000	00000005	00000007	00000004	00000001	00000000
s_RegWrAddr	5'h00	00		1B	) OF	10	11	13	14
s_rs	5'h00	05	03			0A	00		
s_rt	5'h00	OF	10	11	13	14	11	00	
s_rd	5'h00	00					02	<u>t</u> oo	
s_Imed	16'h0000	0007				0003	1001	<u> 1 0000</u>	
s_opcode	6'h00	OC	0D	) OE	) OA	0B	() OF	<u>100</u>	
if_id_sInst	32'h00000000	30AF0007	34700007	38710007	28730007	2D540003	3C111001	00000000	

#### LW & SW

sw \$27, 0(\$17) (RED) #Place x00050000 in address 4 sw \$21, 4(\$17) (YELLOW) #Place x1c in address 8 lw \$21, 0(\$17) (RED) #Place x00050000 in \$21 lw \$22, 4(\$17) (YELLOW) #Place x1c in \$22

s_RegWrData	32'h00000000	00000000			100 10000	10010004	00050000	0000001C
s_RegWrAddr	5'h00	00			( 1B	15		16
s_rs	5'h00	11				<u> 100</u>		
s_rt	5'h00	18	15		16	<u>10</u>	00	
s_rd	5'h00	00						15
s_Imed	16'h0000	0000	0004	0000	0004	0040	0000	A820
s_opcode	6'h00	28		23		02	00	
if_id_sInst	32'h00000000	AE3B0000	AE350004	8E350000	8E360004	08100040	00000000	0000A820
s_DMemWr	0					1		
s_DMemAddr	32'h00000000	00000000		10010000	10010004	10010000	10010004	00000007
s_DMemData	32'h00000000	00000000		00050000	0000001C			00000007
s_DMemOut	32'h00050000	00000007			00000009	00050000	0000001C	

### Jump

j skip\_add (RED)

nop

add \$21, \$22, \$0 #Place x1c in \$21(THIS SHOULD BE SKIPPED)

skip\_add:

add \$21, \$0, \$0 (YELLOW) #Place x0 in \$21

s_RegWrData	32'h00000000		10010004	00050000	0000001C	00000007	0000000	
s_RegWrAddr	5'h00		15		16	10	00	(15)
s_rs	5'h00		00					
s_rt	5'h00		10	<u>(00</u>				(10)
s_rd	5'h00	00	)		15	00		
s_Imed	16'h0000		0040	0000	A820	0000		(0045)
s_opcode	6'h00		02	00				(03)
if_id_sInst	32'h00000000		08100040	00000000	0000A820	00000000		(OC100045)

## JAL & JR

jal task (RED)

nop

task: #Loops until \$21 equals 3

add \$21, \$21, 1 (YELLOW) #increment \$21 by 1 three times

nop

nop nop

beq \$21, \$3, exit\_task (BLUE)

nop

nop

nop

jr \$ra (RED)

nop

nop

exit\_task:

## addi \$2, \$0, 10 (ORANGE) # Place "10" in \$v0 to signal an "exit" or "halt"

s_RegWrData	32'h000	,	0000000						00	400110	00	000000	0000	00001	10000	00000	
s_RegWrAddr	5'h00		15		00				1F		00		15		100		
s_rs	5'h00		00				15		100						15		100
s_rt	5'h00		10		00		15		100						03		00
s_rd	5'h00		00														
s_Imed	16'h000	0	0045		0000		000	1	100	00					10006	5	0000
s_opcode	6'h00		03		00		08		100						04		00
if_id_sInst	32'h000	00000	0C100	0045	00000	000	22B	50001	1000	000000					12A3	30006	0000000
s_RegWrData	32'h00000	0000	00000000		F	FFFFF	FE	00000	0000								
s_RegWrAddr	5'h00		00		10	)3		00									
s_rs	5'h00		00					1F		00					15		100
s_rt	5'h00		00												15		00
s_rd	5'h00		00														
s_Imed	16'h0000		0000					0008		0000					000	)1	0000
s_opcode	6'h00		00												08		100
if_id_sInst	32'h00000	0000	00000000					03E00	800	00000	000				228	350001	0000000
s_RegWrData	32'h0000	0000	00000000		0	00000	002	00000	0000				Ff	FFFFF	= (00	000000	
s_RegWrAddr	5'h00		00		1	.5		00					10	3	00		
s_rs	5'h00		00					15		<u>, 00</u>					1F		00
s_rt	5'h00		00					03		00							
s_rd	5'h00		00														
s_Imed	16'h0000		0000					0006		0000					100	08	0000
s_opcode	6'h00		00					04		00							
if_id_sInst	32'h0000	0000	00000000					12A3	0006	<u>1 00000</u>					103	E00008	1000000
s_RegWrData	32'h00000000										00003	00000000					
s_RegWrAddr	5'h00 5'h00	100			15		00			15		100 115	100			03	
s_rs s_rt	5'h00				15		00					03	100			(02	
s_rd	5'h00																
s_Imed	16'h0000	( 0000			0001		0000					0006	0000	)		(000/	4
s_opcode	6'h00				08		00					04	00			08	
if_id_sInst	32'h00000000	00000000			22850	0001	00000	000				12A30006	10000	00000		200	2000A

Overall the new forwarding unit and Hazard control unit have completely gotten rid of the nops in the code. However, branch changes include register flushing. So, there are intermediate nops being added to the instructions to accommodate for the delay of getting the new instruction addresses. The forwarding unit has allowed for a massive reduction in the amount of nops needed in the assembly level instruction and does not require any flushing or stalls except for instructions that are four apart. This allows for a much greater speedup in the overall processor.

#### Demonstration of an assembly bubble sort application to be run on processor.

0x00400000         0x3c011001         lui \$1,0x00001001         6:         lui \$1,0x00001001           0x00400000         0x34370000         ori \$23,\$1,0x0000000         7:         ori \$37,\$1,0x0000000           0x00400000         0x20100000         addi \$16,\$0,0x0000000         8:         addi \$30,\$0,0         \$100001           0x00400000         0x20100000         addi \$15,\$0,0x00000000         8:         addi \$30,\$0,0         \$10001           0x00400000         0x20100000         addi \$17,\$0,0x00000000         9:         addi \$3,\$0,9         \$2000000           0x00400010         0x2010000         addi \$12,\$0,0x00000000         10:         addi \$41,\$0,0         \$100001           0x00400010         0x200000a         addi \$12,\$0,0x00000000         11:         addi \$41,\$0,0         \$10001           0x00400010         0x30400010         0x30400010         12:         111 \$4,0x00001010         \$20000004           0x00400010         0x30440001         0x4,0x000000004         13:         ori \$4, \$4, \$4,0x00000004         \$3000000004           0x00400020         0x30440001         ox30440002         15,\$23,\$15         17:         add \$77, \$57, \$77           0x00400020         0x0040002         0x0440002         0x0400020         0x0040002         0x040002 <td< th=""><th></th></td<>	
0x00400008         0x2010000         addi \$16,\$0,0x00000000         8:         addi \$s0, \$0, 0 \$loop 1 counter           0x0040000c         0x20130009 addi \$16,\$0,0x00000009         9:         addi \$s3, \$0, 9           0x00400010         0x2010000 addi \$17,\$0,0x0000000         10:         addi \$s1, \$0, 0 \$loop 2 counter           0x0040010         0x3c041010         11:         addi \$s1, \$0, 10         addi \$s1, \$0, 0 \$loop 2 counter           0x0040010         0x3c041010         11:         addi \$s1, \$40, 0 \$loop 2 counter           0x0040010         0x3c041010         12:         111 \$4, 0x00001010           0x00400010         0x3c041010         13:         ori \$4, \$4, 0x0000004           0x00400010         0x3c041010         14:         s11 \$t7, \$s1, 2           0x00400020         0x00117880         s11 \$15,\$	
0x0040000c         0x20130009 addi ¢19,¢0,0x0000000         9:         addi ¢s3, ¢0, 9           0x00400010         0x20110000 addi ¢17,¢0,0x0000000         10:         addi ¢s1, ¢0, 0         #loop 2 counter           0x00400010         0x200c000a addi ¢12,¢0,0x0000000         11:         addi ¢s1, ¢0, 0         #loop 2 counter           0x00400010         0x200c000a addi ¢12,¢0,0x0000000         11:         addi ¢s1, ¢0, 10            0x00400010         0x30541010         112:         111 ¢4,0x00001010            0x0040001c         0x34840004 ori ¢4,¢4,0x00000004         13:         ori ¢4, ¢4, 0x00000004            0x00400020         0x00117880 s11 £15,¢17,0x00000002         16:         s11 ¢t7, ¢s1, 2            0x00400024         0x02ef7820 add £15,¢23,¢15         17:         add £t7, ¢s7, ¢t7            0x00400022         0x8de80000 [w ¢8,0x00000000(€15)         18:         [w \$t0,0(\$t7)]            0x00400022         0x8de90004 [w ¢9,0x00000000(€15)         19:         [w \$t1, 4(\$t7])	
0x00400010         0x20110000         addi \$17,\$0,0x00000000         10:         addi \$s1,\$0,0         \$loop 2 counter           0x00400014         0x200c000a addi \$12,\$0,0x0000000         11:         addi \$s4,\$0,0         \$loop 2 counter           0x00400016         0x30c41010         111         4ddi \$s4,\$0,000010         12:         101         \$4,0x00000004           0x00400012         0x3640004 ori \$s4,\$4000000004         13:         ori \$4,\$4,0x00000004         0x00400024           0x00400020         0x0017880         \$11 \$15,\$17,0x00000002         16:         \$11 \$t7,\$s1,\$2           0x00400024         0x02ef7820 add \$15,\$23,\$15         17:         add \$t7,\$s7,\$t7           0x00400022         0x8de\$0000 [w \$8,\$0x0000000\$(\$t5)         18:         !w \$t0,0(\$t7)           0x00400022         0x8de\$0000 [w \$5,\$0x000000\$(\$t5)         19:         !w \$t1,4(\$t7)	
0x00400014         0x200c000a addi \$12,\$0,0x000000a 11: addi \$t4, \$0, 10           0x00400018         0x3c041010         1ui \$4,0x00001010         12: lui \$4,0x00001010           0x0040001c         0x34840004 ori \$4,\$4,0x0000004         13: ori \$4, \$4, 0x0000004         0x00400024           0x0040002c         0x00417880 \$11 \$15,\$17,0x00000002         16: \$11 \$47, \$51, \$2           0x0040002c         0x0247820 add \$15,\$23,\$15         17: add \$t7, \$57, \$t7           0x0040002c         0x8de80000 [w \$8,0x0000004(\$15)         18: lw \$t0,0(\$t7)           0x0040002c         0x8de90004 lw \$9,0x00000004(\$15)         19: lw \$t1,4(\$t7)	
0x00400018         0x3c041010         lui \$4,0x00001010         l2:         lui \$4,0x00001010           0x00400010         0x34840004         ori \$4,54,0x00000004         l3:         ori \$4, \$4,0x0000004           0x00400020         0x00417880         sl1 \$15,\$17,0x00000002         l6:         sl1 \$77, \$s1, 2           0x00400024         0x02467820         add \$15,\$23,\$15         l7:         add \$77, \$s7, \$t7           0x00400022         0x8de80000 [w \$6,0x000000(\$15)         l8:         lw \$t0,0(\$t7)           0x00400022         0x8de90004         w \$1,4(\$t7)	
0x0040001c         0x34840004         ori \$4,\$4,0x00000004         13:         ori \$4,\$4,0x00000004           0x00400020         0x00417820         sl1 \$15,\$17,0x00000002         16:         sl1 \$17,\$sl,2           0x00400024         0x02247820         add \$15,\$23,\$15         17:         add \$t7,\$sl,\$t7           0x00400028         0x8de80000         h\$8,0x0000000(15)         18:         1w \$t0,0(\$t7)           0x0040002c         0x8de80000         h\$9,0x000000(4(515)         19:         1w \$t1, 4(\$t7)	
0x00400020         0x00117880         sl1 \$15,\$17,0x00000002         16:         sl1 \$t7, \$s1, 2           0x00400024         0x02ef7820         add \$15,\$23,\$15         17:         add \$t7, \$s7, \$t7           0x00400028         0x8de80000 [w \$8,0x00000000(\$15)         18:         lw \$t0, 0(\$t7)           0x0040002c         0x8de90004 [w \$5,0x00000000(\$15)         19:         lw \$t1, 4(\$t7)	
0x00400024         0x02ef7820         add \$15,\$23,\$15         17:         add \$t7, \$s7, \$t7           0x00400028         0x8de80000         1w \$8,0x00000000(\$15)         18:         1w \$t0, 0(\$t7)           0x0040002c         0x8de90004         1w \$9,0x00000004(\$15)         19:         1w \$t1, 4(\$t7)	
0x00400028         0x8de80000         1w \$\$0,00000000(\$15)         18:         1w \$\$0,0(\$t7)           0x0040002c         0x8de90004         1w \$\$0,00000004(\$15)         19:         1w \$\$1,4(\$t7)	
0x0040002c 0x8de90004 1w \$9,0x00000004 (\$15) 19: 1w \$t1, 4(\$t7)	
0x00400030 0x0109502a alt \$10.\$8.\$9 20: alt \$t2, \$t0, \$t1	
0x00400034 0x15400002 bne \$10,\$0,0x00000002 21: bne \$t2, \$zero, increment	
0x00400038 0xade90000 sw ¢9,0x00000000(¢15) 22: sw ¢tl, 0(¢t7)	
0x0040003c 0xade80004 sw \$8,0x00000004(\$15) 23: sw \$t0, 4(\$t7)	
0x00400040 0x22310001 addi \$17,\$17,0x0000 26: addi \$s1, \$s1, 1	
0x00400044 0x0270a822 sub \$21,\$19,\$16 27: sub \$55, \$s3, \$s0	
0x00400048 0x1635fff5 bne \$17,\$21,0xffffff5 28: bne \$\$1, \$\$5, loop	
0x0040004c 0x22100001 addi \$16,\$16,0x0000 29: addi \$s0, \$s0, 1	
	"exit" or
0x0040005c 0x0000000c syscall 43: syscall # Actually cause the halt	
0x00400050         0x20110000 addi \$17,\$0,0x00000000         30:         addi \$s1, \$0, 0           0x00400054         0x1613fff2bne \$16,\$19,0xfffffff2         31:         bne \$	"ex:

The order of values stored in memory are 1, 9, 6, 3, 5, 8, -3, 11, 2, 10. This is the starting order the values are sorted from.

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0xfffffffd	0x0000001	0x0000002	0x0000003	0x0000005	0x0000006	0x000000x0	0x00000009
0x10010020	0x000000a	0x000000b	0x00000000	0x00000000	0x00000000	0x00000000	0x000000x0	0x00000000
0x10010040	0x000000x0	0x000000x0	0x00000000	0x00000000	0x00000000	0x00000000	0x000000x0	0x00000000
0x10010060	0x000000x0	0x000000x0	0x00000000	0x00000000	0x00000000	0x00000000	0x000000x0	0x00000000

## Once sorted, the values are ordered to -3, 1, 2, 3, 5, 6, 8, 9, 10, 11.

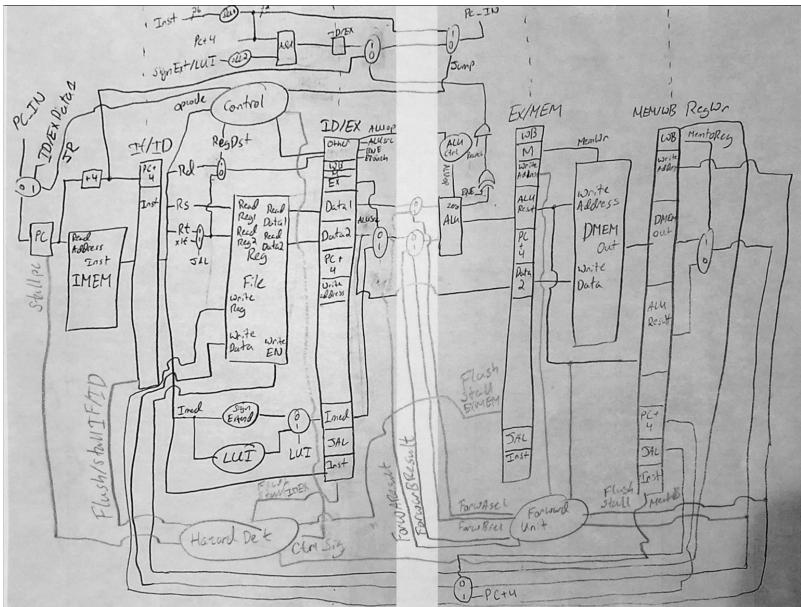
Mars simulation showed victory when running the code for validation in processor. \*\* Warining: your source directory contains a file without the .vhd extension \*\* \*\* control\_withselect.vhd.bak and other files without the .vhd extension (including .vhdl) will be ignored \*\* Please provide the assembly file to run. Use unix style paths like: MARsWork/Examples/addiSeq.asm MARsWork/Examples/Proj-C\_test2.as, Invalid path to assembly file to run. Use unix style paths like: MARsWork/Examples/addiSeq.asm MARsWork/Examples/Proj-C\_test2.as, Invalid path to assembly file to run. Use unix style paths like: MARsWork/Examples/addiSeq.asm MARsWork/Examples/Proj-C\_test2.asm Starting compilation... Successfully compiled vhdl Starting VHDL Simulation... Successfully simulated program! Victory!! Your processes matches MARS expected output with no mismatches!! Press any key to close . . . Reading pref.tcl Now that the MARs result confirms the validity of our program, I can run the testbench program to compare the MARs output and our processor's output. So, at the end of the end of bubble sort application, I added a lw instruction for each address to display the data values in sequential order.

s_DMemAddr	32'h00	10010000	10010004	10010008	1001000C	10010010	10010014	10010018	1001001C	10010020	10010024
s_DMemOut	32'hFF	FFFFFFD	00000001	00000002	00000003	00000005	00000006	80000008	00000009	A0000000	000000B

#### Timing of Processor without dependency handling.

The maximum frequency is 50.5 MHz. The critical path is 22.751 ns, it goes through id\_ex\_register to the ALU, to the instruction\_ctrl\_mod, to finally the instruction\_address\_module.

### <u>High-level schematic drawing of the interconnection between components for the MIPS Hardware-scheduled</u> pipelined processor with data dependency handling.



## **Demonstration of Instructions with data dependency handling.** ADDI

addi \$1, \$0, 1 (RED) #Place 1 in \$1 addi \$2, \$0, 2 (YELLOW) #Place 2 in \$2

audi $\phi_2, \phi_0, z_1$		luce 2 ll	$1 \psi Z$			
s_RegWrData	32'h00000000	00000000			00000001	00000002
s_RegWrAddr	5'h00	00			01	02
s_rs	5'h00	00				
s_rt	5'h00	01	02	03	04	05
s_rd	5'h00	00				
s_Imed	16'h0000	0001	0002	0003	0004	0005
s_opcode	6'h00	08				
if_id_sInst	32'h00000000	20010001	20020002	20030003	20040004	20050005
s_mux_ALU_A_sel	1					
s_mux_ALU_B_sel	1					
s_forw_exmem_aluResultA	32'h00000000	00000000				
s_forw_exmem_aluResultB	32'h00000000	XXXXXXXXX				
if_id_we	1					
id_ex_we	1					
ex_mem_we	1					
mem_wb_we	1					
if_id_rst	0					
id_ex_rst	0					
ex_mem_rst	0					
mem_wb_rst	0					
pc_we	1					

## ADD & SUB & ADDU & SUBU

Add \$11, \$3, \$4 (RED) #Place 3+4=7 in \$11 Addu \$12, \$5, \$10 (YELLOW) #Place -3 + 5= 2 in \$12 Sub \$13, \$7, \$4 (RED) #Place 7-4=3 in \$13 Subu \$14, \$10, \$5 (YELLOW) #Place -3-5=-8 in \$14

s_RegWrData	32'h00000000	00000000	10000003	00000000	0000007	0000002	0000003	FFFFFF8
s_RegWrAddr	5'h00	00	11E	103	.08			OE
	5'h00	(03	105	07	OA OA	03		
s_rs			iii					
s_rt	5'h00	(04	10A	04	05	07		
s_rd	5'h00	OB	<u>10C</u>	OD	OE	OF	10	11
s_Imed	16'h0000	5820	6021	6822	7023	7824	8025	8826
s_opcode	6'h00	(00)						
if_id_sInst	32'h00000000	00645820	00AA6021	00E46822	01457023	00677824	00678025	00678826
s_mux_ALU_A_sel	1							
s_mux_ALU_B_sel	1							
s_forw_exmem_aluResultA	32'h00000000	0000003						
s_forw_exmem_aluResultB	32'h00000000	00000004						
if_id_we	1							
id_ex_we	1							
ex_mem_we	1							
mem_wb_we	1							
if_id_rst	0							
id_ex_rst	0							
ex_mem_rst	0							
mem_wb_rst	0							
pc_we	1							
DNE								

## BNE

Looper: add \$30, \$30, \$1 (RED) #add 1 to \$30 unitl \$30 == 1 bne \$30, \$3, Looper (YELLOW)

add \$11, \$3, \$4 (BLUE) #Place 7 in \$11

s_RegWrAddr s_rs s_rt s_rd s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel	5'h00 5'h00 5'h00 5'h00 16'h0000 6'h00 32'h00000000	0E 1E 01 1E 5020 0) 03C 1F020	(0D) (03) (1F)	(ОВ (03	1E	03	100			
s_rt s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel	5'h00 5'h00 16'h0000 6'h00	01 1E F020 0)	(1F	103		103	Ĭ 00			
s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel	5'h00 16'h0000 6'h00	01 1E F020 0)	(1F		00		1E			
s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel	16'h0000 6'h00	) F020 0)		04	00		01	03		
s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel	6'h00	0)		( OB	00		1E	1F		
if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel		0)	<b>XFFFE</b>	5820	0000		F020	, FFFE		
s_mux_ALU_A_sel s_mux_ALU_B_sel	32'h00000000 1		05	00				05		
s_mux_ALU_B_sel	1	0301020		E 00645820	0000000	0	03C1F	020 17C3	FFFE	
s_mux_ALU_B_sel										
	1									
s_forw_exmem_aluResultA	32'h00000000	10000007		1 0000000 1			1 00000	000		
s_forw_exmem_aluResultB	32'h00000000	00000004								
if_id_we	1									
id_ex_we	1									
ex_mem_we	1									
mem_wb_we	1									
if_id_rst	0					_				
id_ex_rst	0									
ex_mem_rst	0									
mem_wb_rst	0									
s_RegWrData	32'h00000000		0000000	FFFFFFF	0000000					
		0000000						0000003	· · · · · · · · · · · · · · · · · · ·	00000007
s_RegWrAddr	5'h00	00	1E	03	00	,		1E	03	ХOB
s_rs	5'h00 5'h00	00 03	1E 00	03	00 1E		03	1E 105	0 <u>3</u> 07	<u>(ов</u> (оа
s_rs s_rt	5'h00 5'h00 5'h00	00 03 04	1E 00 00	03	00 1E 01	03	04	1E 105 10A	03 07 04	0B 0A 05
s_rs s_rt s_rd	5'h00 5'h00 5'h00 5'h00	00 03 04 0B	1E 00 00 00	03	00 1E 01 1E	03 1F	04 (08	1E 05 10A 10C	03 07 04 0D	08 0A 05 0E
s_rs s_rt s_rd s_īmed	5'h00 5'h00 5'h00 5'h00 16'h0000	00 03 04 08 5820	1E 00 00	03	00 1E 01 1E	03 1F FFFE	04 0B 5820	1E 105 10A	03 07 04	0B 0A 05
s_rs s_rt s_rd s_Imed s_opcode	5'h00 5'h00 5'h00 5'h00 16'h0000 6'h00	00 03 04 08 5820 00	1E 00 00 00 000 0000		00 1E 01 1E 0 F020 0	03 1F FFFE 05	04 0B 5820 00	1E (05 (0A (0C (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_Imed s_opcode if_id_sInst	5'h00 5'h00 5'h00 5'h00 16'h0000	00 03 04 08 5820 00	1E 00 00 00		00 1E 01 1E 0 F020 0	03 1F FFFE 05	04 0B 5820	1E (05 (0A (0C (6021	03 07 04 00 6822	08 0A 05 0E
s_rs s_rt s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel	5'h00 5'h00 5'h00 5'h00 16'h0000 6'h00	00 03 04 08 5820 00	1E 00 00 00 000 0000		00 1E 01 1E 0 F020 0	03 1F FFFE 05	04 0B 5820 00	1E (05 (0A (0C (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1	00 03 04 08 5820 00 00645820	1E 00 00 00 000 0000		00 1E 01 1E 5020 03C 1F020	03 1F FFFE 05	04 08 5820 00 00645820	1E (05 (0A (0C (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel s_forw_exmem_aluResultA	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1 32'h00000000	00 03 04 08 5820 00 00645820 - 00000002	1E 00 00 00 000 0000		00 1E 01 1E F020	03 1F FFFE 05	04 0B 5820 00	1E (05 (0A (0C (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel s_forw_exmem_aluResultA s_forw_exmem_aluResultB	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1	00 03 04 08 5820 00 00645820	1E 00 00 00 000 0000		00 1E 01 1E 5020 03C 1F020	03 1F FFFE 05	04 08 5820 00 00645820	1E (05 (0A (0C (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel s_forw_exmem_aluResultA s_forw_exmem_aluResultB if_id_we	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1 32'h00000000	00 03 04 08 5820 00 00645820 - 00000002	1E 00 00 00 000 0000		00 1E 01 1E 5020 03C 1F020	03 1F FFFE 05	04 08 5820 00 00645820	1 <u>E</u> (05 (0A (0C) (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel s_forw_exmem_aluResultA s_forw_exmem_aluResultB if_id_we id_ex_we	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1 32'h00000000	00 03 04 08 5820 00 00645820 - 00000002	1E 00 00 00 000 0000		00 1E 01 1E 5020 03C 1F020	03 1F FFFE 05	04 08 5820 00 00645820	1 <u>E</u> (05 (0A (0C) (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_med s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel s_forw_exmem_aluResultA s_forw_exmem_aluResultA if_id_we id_ex_we ex_mem_we	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1 32'h00000000	00 03 04 08 5820 00 00645820 - 00000002	1E 00 00 00 000 0000		00 1E 01 1E 5020 03C 1F020	03 1F FFFE 05	04 08 5820 00 00645820	1 <u>E</u> (05 (0A (0C) (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel s_forw_exmem_aluResultA s_forw_exmem_aluResultB if_id_we id_ex_we ex_mem_we mem_wb_we	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1 32'h00000000	00 03 04 08 5820 00 00645820 - 00000002	1E 00 00 00 000 0000		00 1E 01 1E 5020 03C 1F020	03 1F FFFE 05	04 08 5820 00 00645820	1 <u>E</u> (05 (0A (0C) (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_med s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel s_forw_exmem_aluResultA s_forw_exmem_aluResultB if_id_we id_ex_we ex_mem_we mem_wb_we if_id_rst	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1 32'h00000000	00 03 04 08 5820 00 00645820 - 00000002	1E 00 00 00 000 0000		00 1E 01 1E 5020 03C 1F020	03 1F FFFE 05	04 08 5820 00 00645820	1 <u>E</u> (05 (0A (0C) (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_Imed s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel s_forw_exmem_aluResultA s_forw_exmem_aluResultA if_id_we id_ex_we ex_mem_we mem_wb_we if_id_rst id_ex_rst	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1 32'h00000000	00 03 04 08 5820 00 00645820 - 00000002	1E 00 00 00 000 0000		00 1E 01 1E 5020 03C 1F020	03 1F FFFE 05	04 08 5820 00 00645820	1 <u>E</u> (05 (0A (0C) (6021	03 07 04 00 6822	0B 0A 05 0E 7023
s_rs s_rt s_rd s_med s_opcode if_id_sInst s_mux_ALU_A_sel s_mux_ALU_B_sel s_forw_exmem_aluResultA s_forw_exmem_aluResultB if_id_we id_ex_we ex_mem_we mem_wb_we if_id_rst	5'h00 5'h00 5'h00 16'h000 6'h00 32'h00000000 1 1 32'h00000000	00 03 04 08 5820 00 00645820 - 00000002	1E 00 00 00 000 0000		00 1E 01 1E 5020 03C 1F020	03 1F FFFE 05	04 08 5820 00 00645820	1 <u>E</u> (05 (0A (0C) (6021	03 07 04 00 6822	0B 0A 05 0E 7023
	5'h00 5'h00	00 03	1E 00	03	00 1E			1E 105	0 <u>3</u> 07	, OB

#### AND & OR & XOR & NOR

and \$15, \$3, \$7 (RED) #Place 3 in \$15 or \$16, \$3, \$7 (YELLOW) #Place 7 in \$16

xor \$17, \$3, \$7 (RED) #Place 4 in \$17

nor \$18, \$3, \$7 (YELLOW) #Place a -8 in \$18

s_RegWrData	32'h0000000	00000002	0000003	FFFFFF8	00000003	00000007	00000004	FFFFFF8
s_RegWrAddr	5'h00	(OC	(OD	(OE	0F	10	11	12
s_rs	5'h00	03					, OA	<u>, 00</u>
s_rt	5'h00	07					03	07
s_rd	5'h00	OF	10	11	12	13	14	15
s_Imed	16'h0000	7824	8025	8826	9027	982A	A02B	A880
s_opcode	6'h00	00						
if_id_sInst	32'h0000000	00677824	00678025	00678826	00679027	0067982A	0143A02B	0007A880
s_mux_ALU_A_sel	1							
s_mux_ALU_B_sel	1							
s_forw_exmem_aluResultA	32'h0000000	0000003						
s_forw_exmem_aluResultB	32'h0000000	00000004						
if_id_we	1							
id_ex_we	1							
ex_mem_we	1							
mem_wb_we	1							
if_id_rst	0							
id_ex_rst	0							
ex_mem_rst	0							
mem_wb_rst	0							
pc_we	1							

#### **SLT & SLTU & SLL & SLV & SRL & SRLV** slt \$19, \$3, \$7 (RED) #Place 1 in \$19

slt \$19, \$5, \$7 (RED) #Place 1 in \$19 sltu \$20, \$10, \$3 (YELLOW) #Place 0 in \$20 sll \$21, \$7, 2 (RED) #place 28 in \$21 sllv \$22, \$7, \$3 (YELLOW) #place 56 in \$22 srl \$23, \$7, 2 (RED) #Place 1 in \$23 srlv \$24, \$7, \$2 (YELLOW) #place 1 in \$24

sriv \$24, \$7, \$2 (YE			ý	¥						
s_RegWrData	32'h00000000	00000007	00000004	<u>[FFFFFF8</u>	00000001	00000000	0000001C		00000001	
s_RegWrAddr	5'h00	110	111	. 12	13	14	15	16	17	18
s_rs	5'h00	03	Î OA	00	03	00	02	00		<u> </u>
s_rt	5'h00	07	<b>1</b> 03	07				00		16
s_rd	5'h00	13	14	15	16	17	18	00		19
s_Imed	16'h0000	982A	A02B	A880	B004	B882	C006	0000		C8C3
s_opcode	6'h00	00								
if_id_sInst	32'h00000000	0067982A	0143A02B	0007A880	00678004	00078882	0047C006	00000000		0016C8C3
s_mux_ALU_A_sel	1									1
s_mux_ALU_B_sel	1									
s_forw_exmem_aluResultA	32'h00000000	0000003								00000000
s_forw_exmem_aluResultB	32'h00000000	00000004								
if_id_we	1									
id_ex_we	1									
ex_mem_we	1									
mem_wb_we	1									
if_id_rst	0								1	
id_ex_rst	0									
ex_mem_rst	0									
mem_wb_rst	0									
pc_we	1									

## SRA & SRAV

sra \$25, \$3, 2 #Sift x3 sra 2 bits to right (RED) srav \$26, \$3, \$2 #shift x3 sra 2 bits to the right (YELLOW)

s_RegWrData	32'h00000000	00000038	00000001		C0000000	X
s_RegWrAddr	5'h00	16	17	18	19	[1A ]
s_rs	5'h00	00	02	<b>(</b> 00	1E	X
s_rt	5'h00	03		<mark>,</mark> 1Е	01	103 I
s_rd	5'h00	19	1A	<b>0</b> 0	1E	(1F)
s_Imed	16'h0000	C883	D007	(0000	F020	(FFFE )
s_opcode	6'h00	00		( <mark>08</mark>	00	(04 (
if_id_sInst	32'h00000000	0003C883	0043D007	201E0000	03C1F020	13C3FFFE
s_mux_ALU_A_sel	1					
s_mux_ALU_B_sel	1					
s_forw_exmem_aluResultA	32'h00000000	00000003				20000000 2
s_forw_exmem_aluResultB	32'h00000000	00000004				
if_id_we	1					
id_ex_we	1					
ex_mem_we	1					
mem_wb_we	1					
if_id_rst	0					
id_ex_rst	0					
ex_mem_rst	0					
mem_wb_rst	0					
pc_we	1					

# BEQ & LUI

## Looper\_2:

add \$30, \$30, \$1 (RED) #Add 1 to \$20 and branch if \$30 == 3 beq \$30, \$3, Looper\_2 (YELLOW) #\$30 = 1so will continue

lui \$27, 5 (BLUE) #Place x00050000 in \$27

s_RegWrData	32'h00000000	<u>C0000007</u>	C0000000	<u> 1 00000000 x 00000 x 00000 x 00000 x 000000</u>	00000001	<b>FFFFFFE</b>	00050000
s_RegWrAddr	5'h00	19	1A	1E		03	1B
s_rs	5'h00	1E		<u>100</u>	<u> 105</u>	100	
s_rt	5'h00	01	03	1B	<u>, OF</u>	100	
s_rd	5'h00	1E	1F	00			
s_Imed	16'h0000	F020	FFFE	0005	0007	0000	
s_opcode	6'h00	00	04	OF	()OC	<u>100</u>	
if_id_sInst	32'h00000000	03C1F020	13C3FFFE	3C1B0005	30AF0007	00000000	
s_mux_ALU_A_sel	1				·		
s_mux_ALU_B_sel	1						
s_forw_exmem_aluResultA	32'h00000000	00000000		00000001			
s_forw_exmem_aluResultB	32'h00000000	00000004					
if_id_we	1						
id_ex_we	1						
ex_mem_we	1						
mem_wb_we	1						
if_id_rst	0						
id_ex_rst	0						
ex_mem_rst	0						
mem_wb_rst	0						
pc_we	1						

## **ANDI \$ ORI & XORI & SLTI & SLTIU** andi \$15, \$5, (RED) 7 #Place 5 in \$15 ori \$16, \$3, 7 (YELLOW) #Place 7 in \$16

xori \$17, \$3, 7 (RED) #Place 4 in \$17 slti \$19, \$3, 7 (YELLOW) #Place 1 in \$19

s_RegWrData	32'h00000000	00000005	00000000		00000007	00000004	00000001	00000000
s_RegWrAddr	5'h00	<u>, OF</u>	00		10	11	13	14 (1
s_rs	5'h00	03			0A	00		t t
s_rt	5'h00	10	11	13	14	00		11
s_rd	5'h00	<u>¢</u> 0				00		02 1
s_Imed	16'h0000	0007			0003	0000		1001
s_opcode	6'h00	] OD	0E	0A	OB	00		(OF ):
if_id_sInst	32'h00000000	34700007	38710007	28730007	2D540003	00000000		3C111001
s_mux_ALU_A_sel	1							
s_mux_ALU_B_sel	1							
s_forw_exmem_aluResultA	32'h00000000	00000000						
s_forw_exmem_aluResultB	32'h00000000	00000004						
if_id_we	1							
id_ex_we	1							
ex_mem_we	1							
mem_wb_we	1							
if_id_rst	0						1	
id_ex_rst	0							
ex_mem_rst	0							
mem_wb_rst	0							
pc_we	1							

## LW & SW

sw \$27, 0(\$17) (RED) #Place x00050000 in address 4 sw \$21, 4(\$17) (YELLOW) #Place x1c in address 8 lw \$21, 0(\$17) (RED) #Place x00050000 in \$21 lw \$22, 4(\$17) (YELLOW) #Place x1c in \$22

			+ = =							
s_RegWrData	32'h00000000	00000000		10010000		10010004	10000000		00050000	0000001C
s_RegWrAddr	5'h00	100		11	1B	15	100		15	16
s_rs	5'h00	11		<u>‡ oo</u>		11		00	16	100
s_rt	5'h00	1B	15	<u> 1 00</u>		15	16	10	<u> 100</u>	
s_rd	5'h00	00							15	
s_Imed	16'h0000	0000	0004	0000			0004	0033	A820	
s_opcode	6'h00	2B		<u>(</u> 00		23		02	<b>(</b> 00	
if_id_sInst	32'h00000000	AE3B0000	AE350004	00000000		8E350000	8E360004	08100033	02C0A820	0000A820
if_id_we	1									1
id_ex_we	1									
ex_mem_we	1									
mem_wb_we	1									
if_id_rst	0									
id_ex_rst	0									
ex_mem_rst	0									
mem_wb_rst	0									
pc_we	1									
s_DMemWr	0					1				
s_DMemAddr	32'h00000000	00000000	10010000		10010004	00000000		10010000	10010004	00000007
s_DMemData	32'h00000000	00000000	0000004	00050000	0000001C	00000000		0000001C	00000038	00000007
s_DMemOut	32'h00050000	00000007			0000009	00050000			0000001C	
-										

#### Jump

j skip\_add (RED)

add \$21, \$22, \$0 #Place x1c in \$21 (THIS SHOULD BE SKIPPED) skip\_add:

add \$21, \$0, \$0 (YELLOW) #Place x0 in \$21 jal task (BLUE)

s_RegWrData	32'h00000000	0000000	00050000	0000001C	00000007	00000000		
s_RegWrAddr	5'h00	<b> 0</b>	15	16	10	200		15
s_rs	5'h00	100	16	<u>, 00</u>			15	
s_rt	5'h00	10	00			10	15	
s_rd	5'h00	ф0	15			200		
s_Imed	16'h0000	0033	A820			0035	0001	
s_opcode	6'h00	02	00			03	08	
if_id_sInst	32'h00000000	08100033	02C0A820	0000A820		0C100035	22B50001	
if_id_we	1			1				
id_ex_we	1			1				
ex_mem_we	1							
mem_wb_we	1							
if_id_rst	0							
id_ex_rst	0							
ex_mem_rst	0							
mem_wb_rst	0							
pc_we	1							
s_DMemWr	0							
s_DMemAddr	32'h00000000	10010000	10010004	00000007	0000000			
s_DMemData	32'h00000000	0000001C	00000038	00000007	00000000			
s_DMemOut	32'h00050000	<u>ф0050000</u>	0000001C		00050000			

## JAL & JR

jal task (RED)

task: #Loops until \$21 equals 3

add \$21, \$21, 1 (YELLOW) #increment \$21 by 1 three times

beq \$21, \$3, exit\_task (BLUE)

jr \$ra (RED)

exit\_task:

addi \$2, \$0, 10 (ORANGE) # Place "10" in \$v0 to signal an "exit" or "halt"

			1n \$v0	to signal a						*	
							0		FFFD		
5'h00	00			15	1F	00		[03		100	
5'h00	00	<b></b> (	15			1F	100			15	
5'h00	10	<b></b> 6	15		03	00	100			Į 15	03
5'h00	00										
16'h0000	003	35 (0	0001			0008	( 0000			0001	
6'h00	03		08		04	00	100			<b>(</b> 08	(04
32'h0000000	0 OC	100035	22B50001		12A30001	03E0000	8 200000	00		22B50001	12A30001
1											
1											
1											
1											
0											
0											
0											
0											
1											
32'h00000000	00000000		00000001	FFFFFFE	00000000			00000002	FFFFF	FF 100000000	
5'h00	00		(15	03	00			15	03	100	
5'h00	and the local division of the local division	1, 1F	00		( 15		1F	00		15	
and the second		100	00		( 15	X03	00	00		15	03
(1900) (2000)		V			V						
						04					04
				)							
1					12200001	12400001	00200000				
1									+		
1									1		
1											
0											
0											
0	-										
·											
	32'h0000000 5'h00 5'h00 5'h00 16'h0000 6'h00 32'h0000000 1 1 1 1 1 1 0 0 0 0 0 1 32'h0000000 1 32'h00000000 5'h00	32'h00000000       000         5'h00       00         5'h00       00         5'h00       00         5'h00       00         5'h00       00         16'h0000       00         6'h00       03         32'h00000000       0C         1       1         1       1         0       0         0       0         0       0         1       0         0       0         0       0         1       0         0       0         0       0         0       0         5'h00       15         5'h00       00         5'h00       00         5'h00       00         16'h0000       0001         6'h00       001	32'h00000000       00000000         5'h00       00         5'h00       10         5'h00       00         5'h00       00         5'h00       00         5'h00       00         16'h0000       03         6'h00       03         32'h00000000       0C100035         1       1         1       1         1       1         0       0         0       0         0       0         1       1         1       1         1       1         1       1         0       0         0       0         0       0         0       0         0       0         0       0         1       1         1       1         1       1         1       1         1       1         1       1         1       1         1       1         1       1         1       1         1       1	32h00000000         00000000           5'h00         00           10         15           5'h00         00           16'h000         0035           0001         03           6'h00         03           32'h00000000         0C100035           22B50001         1           1	32h00000000         00000000         00           5h00         00         15           5h00         10         15           5h00         00         15           5h00         00         15           5h00         00         15           5h00         00         15           10         15         10           16'h000         0035         0001           6'h00         03         08           32'h00000000         0C 100035         22B 50001           1         1         1           1         1         1           1         1         1           0         0         1           0         0         1           0         0         1           1         1         1           1         1         1           0         0         1           0         0         1           1         1         1           1         1         1           1         1         1           1         1         1           1         0	32h00000000         00000000         00400004           5'h00         00         15         1F           5'h00         00         15         03           5'h00         00         15         03           5'h00         00         00         03           5'h00         00         00         03           16'h000         0035         0001         04           6'h00         03         08         04           32'h0000000         0C100035         22850001         12A30001           1         1         12A30001         12A30001           1         1         1         1         1           0         0         0         00         00           0         0         0         0         0           1         1         1         1         1           1         1         1         1         1           1         1         1         1         1           1         1         1         1         1           1         1         1         1         1           1         1         1         <	S'h00         00         15         1F         00           S'h00         00         15         15         1F         00           S'h00         10         15         03         00           S'h00         00         15         03         00           S'h00         00         15         03         00           S'h00         00         00         00         00         00           S'h00         00         00         00         00         00         00           S'h00         00         03         08         04         00 <td>32h00000000         00000000         00000000         00000000         00000000           5h00         00         15         1F         00         10         15         1F         00           5h00         00         15         03         00         100         10         100</td> <td>32h00000000         0000000         00400004         0000000         FFFF           Sh00         00         15         1F         00         03           Sh00         00         15         03         00         03           Sh00         00         15         03         00         00           Sh00         00         15         03         00         00           Sh00         00         15         03         00         00         00           Sh00         00</td> <td>32'h00000000         00000000         IFFFFFD         00000000         FFFFFFD           Sh00         00         15         1F         00         03           Sh00         10         15         03         00         100         03           Sh00         00         15         03         00         00         03           Sh00         10         15         03         00         00         00           Sh00         00         10         15         03         00         00         00           Sh00         00         00         10         15         03         00<!--</td--><td>32h00000000         00000000         15         00         00000000         15         00         00000000         15         00         00000000         15         00         000         15         00         000         15         00         000         15         00         000         15         00         000         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         16         00         16         00         16         00         16         00         16         17         00         16         17         00         15         16         16</td></td>	32h00000000         00000000         00000000         00000000         00000000           5h00         00         15         1F         00         10         15         1F         00           5h00         00         15         03         00         100         10         100	32h00000000         0000000         00400004         0000000         FFFF           Sh00         00         15         1F         00         03           Sh00         00         15         03         00         03           Sh00         00         15         03         00         00           Sh00         00         15         03         00         00           Sh00         00         15         03         00         00         00           Sh00         00	32'h00000000         00000000         IFFFFFD         00000000         FFFFFFD           Sh00         00         15         1F         00         03           Sh00         10         15         03         00         100         03           Sh00         00         15         03         00         00         03           Sh00         10         15         03         00         00         00           Sh00         00         10         15         03         00         00         00           Sh00         00         00         10         15         03         00 </td <td>32h00000000         00000000         15         00         00000000         15         00         00000000         15         00         00000000         15         00         000         15         00         000         15         00         000         15         00         000         15         00         000         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         16         00         16         00         16         00         16         00         16         17         00         16         17         00         15         16         16</td>	32h00000000         00000000         15         00         00000000         15         00         00000000         15         00         00000000         15         00         000         15         00         000         15         00         000         15         00         000         15         00         000         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         15         00         00         16         00         16         00         16         00         16         00         16         17         00         16         17         00         15         16         16

s_RegWrData	32'h00000000		00000003	0000000	
s_RegWrAddr	5'h00		15	03	00
s_rs	5'h00	1F	00		
s_rt	5'h00	00	00		02
s_rd	5'h00				
s_Imed	16'h0000	0008	0000		000A
s_opcode	6'b00		00		08
if_id_sInst	22,00000000		00000000		2002000A
if_id_we	1				
id_ex_we	1				
ex_mem_we	1				
mem_wb_we	1				
if_id_rst	0				
id_ex_rst	0				
ex_mem_rst	0				
mem_wb_rst	0				
pc_we	1				

## **BUBBLE SORT without nops**

Bubble sort was reintroduced without the majority of the nops except for two nops for the data dependencies pertaining to loading from data memory (lw).

C:\windows\system32\cmd.exe

** Warining: your source directory contains a file without the .vhd extension ** ** control_withselect.vhd.bak and other files without the .vhd extension (including .vhdl) will be ignored **
Please provide the assembly file to run. Use unix style paths like: MARsWork/Examples/addiSeq.asm >MARsWork/Examples/Proj-C_test2_Hazard_Det_Forwarding.asm starting compilation Successfully compiled vhdl
Starting VHDL Simulation Successfully simulated program!
Victory!! Your processes matches MARS expected output with no mismatches!! Press any key to close Reading pref.tcl
s_DMemAddr 32'h00000000 10010000 10010004 00000000 10010008 1001000C 10010010 10010014
s_DMemOut 32'hFFFFFFD 100000001 FFFFFFFD 100000002 00000003 00000005 00000006
s_DMemAddr 32'h10010014
s_DMemOut 32'h00000006 10000008 0000009 00000000 0000000000000
From the waveform, the stored values are still in the correct order of addresses.
Address Value

Address	value
X10010000	xFFFFFFD
X10010004	x00000001
X10010008	x0000002
X1001000c	x00000003
X10010010	x00000005
X10010014	x00000006
X10010018	x0000008
X1001001c	x00000009
X10010020	x0000000A
X10010024	x000000B

## FORWARD DEPENDENCIES

add \$11, \$1, \$2 #Place 3 in \$11 (RED) add \$12, \$11, \$1 #Place 4 in \$12 RAW \$11 (YELLOW) add \$13, \$11, \$1 #Place 4 in \$13 RAW \$11 (RED)

#### add \$14, \$<mark>11</mark>, \$1 #Place 4 in \$14 RAW \$11 (YELLOW) add \$13, \$11, \$**12** #Place 7 in \$13 RAW \$11 & \$12 (RED)

s_RegWrData	32'h00000000	10000007	[FFFFFFFD	00000000	10000003	100000004		100000000		00000004	100000007
s_RegWrAddr	5'h00	09	Į0Α	1E	, OB	1 oc	, 0D	00		OE	( OD
s_rs	5'h00	01	OB		200		OB		0D	1E	
s_rt	5'h00	02	01		200		01	(0C	0E	01	03
s_rd	5'h00	OB	0C	) OD	200		0E	(OD	0B	1E	↓1F
s_Imed	16'h0000	5820	6020	6820	20000		7020	6820	5820	F020	(FFFE
s_opcode	6'h00	00	<u> </u>								05
if_id_sInst	32'h00000000	00225820	01616020	01616820	20000000		01617020	016C6820	01AE5820	03C1F020	17C3FFFE
s_forw_exmem_aluResultA	32'h00000000	<b>0000000</b>		00000003	0000003		00000000			00000007	
s_forw_exmem_aluResultB	32'h00000000	XXXXXXXX								00000004	
s_mux_ALU_A_sel	1										
s_mux_ALU_B_sel	1										

From the wave form, you can see that the register values are being forwarded through the mux signals connected to the ALU when there is a read after write.

## HAZARD DETECTION

Jump & JAL

j skip\_add (RED)

add \$21, \$22, \$0 #Place x1c in \$21 (THIS SHOULD BE SKIPPED) skip add:

add \$21, \$0, \$0 (YELLOW) #Place x0 in \$21

jal task (BLUE)

s_RegWrData	32'h00000000	<b>0000000</b>	100050000	20000001C	20000007	100000000		
s_RegWrAddr	5'h00	<b>0</b> 0	15	16	10	200		, 15
s_rs	5'h00	00	16	<u>, 00</u>			15	
s_rt	5'h00	10	<b>1</b> 00			10	15	
s_rd	5'h00	00	15			<b>1</b> 00		
s_Imed	16'h0000	0033	A820			0035	0001	
s_opcode	6'h00	02	<u>100</u>			03	08	
f_id_sInst	32'h00000000	08100033	02C0A820	0000A820		0C100035	22B50001	
f_id_we	1							
d_ex_we	1							
x_mem_we	1							
nem_wb_we	1							
f_id_rst	0							
d_ex_rst	0							
ex_mem_rst	0							
nem_wb_rst	0							
oc_we	1							

As you can see, based off the waveform, when a Jump instruction is called, the ID/EX register is flushed to erase the bogus instruction.

## Branch

Looper\_2:

add \$30, \$30, \$1 (RED) #Add 1 to \$20 and branch if \$30 == 3 beq \$30, \$3, Looper\_2 (YELLOW) #\$30 = 1 so will continue

lui \$27, 5 (BLUE) #Place x00050000 in \$27

s_RegWrData	32'h00000000	C0000007	(C0000000	(00000000	00000001	FFFFFFE	00050000
s_RegWrAddr	5'h00	19	1A	1E		03	18
s_rs	5'h00	1E		00	05	00	
s_rt	5'h00	01	03	1B	<u>XOF</u>	00	
s_rd	5'h00	1E	1F	00			
s_Imed	16'h0000	F020	FFFE	0005	0007	0000	
s_opcode	6'h00	00	04	OF	0C	00	
if_id_sInst	32'h0000000	03C1F020	13C3FFFE	3C1B0005	30AF0007	00000000	
s_mux_ALU_A_sel	1				1		
s_mux_ALU_B_sel	1						
s_forw_exmem_aluResultA	32'h0000000	00000000		00000001			
s_forw_exmem_aluResultB	32'h0000000	00000004					
if_id_we	1						
id_ex_we	1						
ex_mem_we	1						
mem_wb_we	1						
if_id_rst	0						1
id_ex_rst	0						
ex_mem_rst	0						
mem_wb_rst	0						
pc_we	1					1	

As you can see the IF/ID register is flushed and the PC is stalled when a branch instruction is sent from the control unit to the hazard control unit.

# JR

jal task (RED)

task: #Loops until \$21 equals 3

add \$21, \$21, 1 (YELLOW) #increment \$21 by 1 three times

beq \$21, \$3, exit\_task (BLUE)

jr \$ra (RED)

exit\_task:

addi \$2, \$0, 10 (ORANGE) # Place "10" in \$v0 to signal an "exit" or "halt"

s_RegWrData	32'h00000000	00000000			004000D4	00000000		FFFFFFD	Ĵ 00000000	
s_RegWrAddr	5'h00	00		15	1F	00		03	100	
s_rs	5'h00	00	15			1F	100		15	
s_rt	5'h00	10	15		03	00	100		15	03
s_rd	5'h00	00								
s_Imed	16'h0000	0035	0001			0008	10000		0001	
s_opcode	6'h00	03	08		04	00	100		<b>(</b> 08	04
if_id_sInst	32'h00000000	0C100035	22B50001		12A30001	03E00008	00000000		22B50001	12A30001
if_id_we	1									
id_ex_we	1									
ex_mem_we	1									
mem_wb_we	1									
if_id_rst	0	_								
id_ex_rst	0									
ex_mem_rst	0									
mem_wb_rst	0									
pc_we	1									

From the waveform, after the jr is called the IF/ID register is flushed on the following clock cycle to erase the bogus instruction.

<u>Timing Report of Synthesis with data dependency handling.</u> The maximum frequency of the updated processor is 35.46MHz. The critical path is 31.141ns and follows: ID/EX register to the forwarding unit, forwarding unit to the Mux\_B\_forwarding unit, from Mux B to ALU, from the ALU to the hazard detection unit, and finally from hazard detection to the IF/ID register.

To improve the frequency further, I would need to redesign the hazard detection unit to use less resets because this only adds to the overall CPI and would also reduce the critical path length in the processor.