Final Project

Darren Codipilly, Brandon Quant

COEN 122

Wednesday 2:15-5:00

Abstract:

For this project we created a datapath that takes in custom assembly instructions and converts those instructions into machine language. The components of the project include muxes, ALUs, the program counter, the instruction memory, the register file, the sign extender, the control unit, the data memory, and the various gates and buffers.

Description of the CPU:

For the instruction fetch stage, we start with the mux and based on the selects of the mux, the 32 bit value is then passed to the program counter. The program counter contains the address or location of the instruction being executed at the current time. Then the 8 bit output is then the input for the first ALU, the instruction memory, and the IF/ID buffer. The instruction memory holds the instructions that need to be decoded and executed. Then the 32 output of the instruction memory is then also passed as the input for the ID/IF buffer. The IF/ID connects the instruction fetch and the instruction decode stages, where the instruction memory and program counter are passed through.

For the instruction decode stage, the instruction memory is 32 bits, where 2 sets of six bits represent the registers that are inputs to the register file. The write address input comes from the write back stage for the rd register. The dataIn and the write inputs come from the memory access stage. The 4 most significant bits of the instruction memory would be the opcode and input to the control unit, where the control unit sets the flags and the aluOp. The 22 least significant bits of the instruction memory that are passed through the IF/ID buffer are extended to 32 bits in the sign extender. This output, along with the program counter output that is passed through the IF/ID buffer passes through the second ALU, where this output is then passed to the ID/EX buffer. All the outputs of this stage are passed through as inputs of the ID/EX buffer.

For the execution stage, the outputs from the register file that are passed through the ID/EX buffer are the inputs for the data memory. The rd register input is passed through as an input for the EX/WB buffer. The flags of MemRead and MemWrite are inputs into the data memory and the ALUop is the select for the third ALU. The rest of the flags are passed through as inputs to the EX/WB buffer. The inputs of the third ALU would be the outputs of the register file that is passed through the ID/EX buffer. The outputs of the third ALU as well as the N and Z flags are passed through the EX/WB buffer. Also, the output of the second ALU passes straight through to the EX/WB buffer.

For the memory/write back stage, we have 2 AND gates, where the first AND gates' inputs are the N and Branch Neg flags while the second AND gates' inputs are the Z and Branch Zero flags. The output of these 2 AND gates are inputs to an OR gate which also takes in the input of the jump flag. This output is then one of the selects for the first MUX in the instruction fetch

stage. The inputs of the second mux would be the third ALU"s output, the data memory output and the second ALU's output which are all passed through from the EX/WB buffer. The selects for the mux are MemtoReg and PCtoReg flags. The output of this second mux is the input is data input for the register file. The outputs of the third ALU and the data memory become the inputs for the first mux. The rd output becomes the write address for the register file in the instruction decode stage.

Truth Table for Control:

Op- code	regW rite	memt oReg	PCto Reg	branc hN	branc hZ	jump	jump Mem	mem Read	mem Write	aluO p
0000	0	0	0	0	0	0	0	0	0	0100
1111	1	0	1	0	0	0	0	0	0	0100
1110	1	1	0	0	0	0	0	1	0	0100
0011	0	0	0	0	0	0	0	0	1	0100
0100	1	0	0	0	0	0	0	0	0	0000
0101	1	0	0	0	0	0	0	0	0	0001
0110	1	0	0	0	0	0	0	0	0	0010
0111	1	0	0	0	0	0	0	0	0	0011
1000	0	0	0	0	0	1	0	0	0	0100
1001	0	0	0	0	1	0	0	0	0	0100
1010	0	0	0	0	0	0	1	1	0	0100
1011	0	0	0	1	0	0	0	0	0	0100

Summation Code:

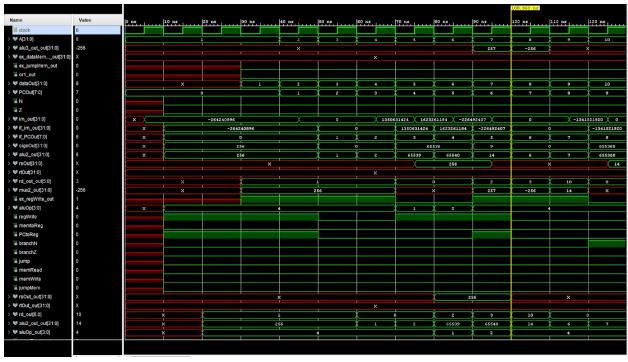
//Assume
$$x0 = 0$$
, $x1 = rs$, $x2 = rt$, $x7 = 3$, $x8 = 1$, $x9 = rd$

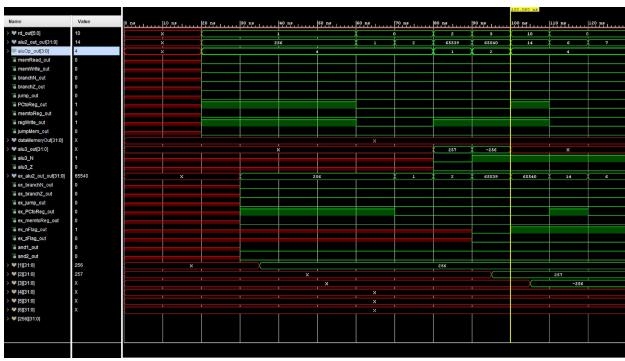
ADD x3, x1, x2 SUB x3, x3, x8 //x3 = rs+rt-1 LDPC x5, 7

SUB x4, x3,x1 //(rs-rt-1)-i BRZ x5 LD x6, [x1] //x6 = memory[i] ADD x0, x0, x6 INC x1, x1 J x7

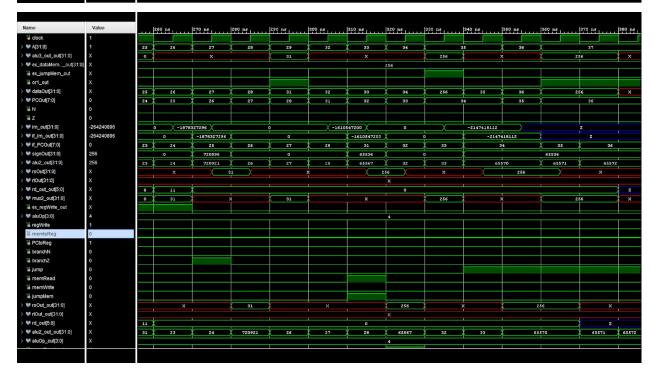
STUR x0, x9

Waveforms:





		Ι													
Name	Value		130 ns	140 ns	150 ns	160 ns	170 ns	180 ns	190 ns	200 ns	210 ns	220 ns	230 ns	240 ns	250 ns
¼ clock	1	1													
> W A[31:0]	1	10	11	12	15	16	17	18	19	20	21	22	23	24	25
> W alu3_out_out[31:0]	X		×	14		×		256		K	256		K	513	0
> W ex_dataMemout[31:0]	х	_				×							256		
a ex_jumpMem_out	X														
or1_out	х														
> W dataOut[31:0]	X	10	11	14	15	16	17	18	19	20	21	22	23	24	25
> W PCOut[7:0]	0	9	10	11	14	15	16	17	18	19	20	21	22	23	24
16 N	0														
le Z	0														
> W im_out[31:0]	-264240896			0	8053	72928		-5200	28160		10947	80928 19044	77184 / -2222	98103	0
> W if_im_out[31:0]	-264240896					805372928	X	0	-520028160	X	0	1094780928	1904477184	-222298103	0
> W if_PCOut[7:0]	0	8	9	10	11	14	15	16	17	18	19	20	21	22	23
> W signOut[31:0]	256			0		66560	X	0	65536	X	0	67584	263168	9	0
> W alu2_out[31:0]	256		9	10	11	66574	15	16	65553	18	19	67604	263189	31	23
> W rsOut[31:0]	X		14	,		2	56	×	2	56	×		256	\sim	x
> W rtOut[31:0]	X			×		2	56			×		21	57 2	56	x
> W rd_out_out(5:0)	X										4		0	5	6
> W mux2_out[31:0]	X		×	14		×		256		×	256		×.	513	0
a ex_regWrite_out	X														
> W aluOp[3:0]	4						4					0	3	4	
a regWrite	1														
¹å memtoReg	0														
1 PCtoReg	1														
branchN	0														
le branchZ	0														
le jump	0														
le memRead	0														
memWrite	0					1									
le jumpMem	0														
> W rsOut_out[31:0]	Х	x	14	1	×		256	*	×	256	:	¢ .	2	56	×
> W rtOut_out[31:0]	Х	×					256			×			257	256	х
> V rd_out[5:0]	Х					0				4			5	6	11
> W alu2_out_out[31:0]	Х	7	655368	9	10	11	66574	15	16	65553	18	19	67604	263189	31
> W aluOp_out[3:0]	X						4						0	3	4
		┨╌													
		Γ													



Name	Value	١,	30 ne	1140 ne	1150 ne	1160 ne	170 ns	1180 ne	1190 ne	1200 ne	210 mg	1220 ne	1230 ne	1240 ne	1250 ne
> V rd_out[5:0]	х	±	30 113	240 115	1.00 1.00		177 23	100 115	130 113						
	X	7		V		•	—	J		4		0 J	5	6	11
> * alu2_out_out[3:0]	X	<u>-</u> 1	655368	9	10	11	66574	15	16	65553	18	19	67604	263189	31
la memRead_out	X	\vdash					4				-			↑ 3	1
a memWrite_out	X														_
le branchN_out	X														
le branchZ_out	X														_
le branciz_out	×														_
la PCtoReg_out	x														
a rectoreg_out	x									-					
la regWrite_out	x						+								
	X														
	X	\pm				×				-			255		
	X	×	14	V	×		256	V	x	256	V	x	256 513 0 X		
la alu3 N	x	_		1			200	1	î –	200	1	î –	513	↑ • •	1-
le alu3_Z	x														
_	x	6	7	655368	9	10	11	66574	15	16	65553	18	19	67604	263189
le ex_branchN_out	x	1		-		1	_ 	00371	1		1	10		07804	203103
	x														
	X						 							 	
	x				i			 			 			 	
le ex_memtoReg_out	x														
le ex_nFlag_out	x														
le ex zFlag out	X														
le and1_out	x														
le and2_out	x			1											
> 🕨 [1][31:0]	x	=							256						
> 🕨 [2][31:0]	x								257						
> • [3][31:0]	x	_							-256						
> W [4][31:0]	x	Т					×				<u> </u>		256		
> 🕨 [5][31:0]	x							×						X	513
> 🕨 [6][31:0]	x	×												χο.	
> 🕨 [256][31:0]	***														\top
		┱													

₩ rd_out[5:0]	х	11											ns 370 ns	z
₩ alu2_out_out[31:0]	X	31	23	24	720921	26	27	28	65567	32	33	65570	6557	
₩ aluOp_out[3:0]	x			1	720522	1 20	1	1 20	4	A 32		1 00070	1 0007	1 000
le memRead_out	x													
le memWrite_out	x													
le branchN_out	x													
branchZ out	x					ì								
le jump out	x													
le PCtoReg_out	x													
le memtoReg_out	x													
le regWrite_out	x													
le jumpMem_out	x													
₩ dataMemoryOut[31:0]	X								256					
₩ alu3_out[31:0]	X		х		31	<u> </u>	×		256	<u> </u>		256		x
le alu3_N	x					1				1		1		
le alu3_Z	X													
♥ ex_alu2_out_out[31:0]	x	2630	31	23	24	720921	26	27	28	65567	32	33	65570	655
& ex_branchN_out	x					1		1	1	1		1		
ex branchZ out	x													
le ex_jump_out	x													
& ex_PCtoReg_out	х													
& ex_memtoReg_out	х													
& ex_nFlag_out	х													
ex_zFlag_out	х													
le and1_out	х													
and2_out	х													
₩ [1][31:0]	х								256					
₩ [2][31:0]	х								257					
W [3][31:0]	х								-256					
W [4][31:0]	х								256					
₩ [5](31:0]	х								513					
₩ [6][31:0]	х								0					
₩ [256][31:0]														

Name	Value	350 ns	360 ns	370 ns	380 ns	390 ns	400 ns	410 ns	420 ns	430 ns	440 ns	450 ns	460 ns	470 ns
[™] clock	1													
> W A[31:0]	1	36	X					3	7					
> W alu3_out_out[31:0]	X	×	2	56	X			_		×				
> W ex_dataMemout[31:0]	X							256						
& ex_jumpMem_out	X													
% or1_out	X													
> W dataOut[31:0]	X	36	2	56	*					x				
> W PCOut[7:0]	0	35	X					3	16					
18 N	0													
16 Z	0													
> W im_out[31:0]	-264240896	-2140						Z						
> W if_im_out[31:0]	-264240896	-2147418112						٠ .	z					
> W if_PCOut[7:0]	0	34	35	<u> </u>					36					
> W signOut[31:0]	256							65536		-				
> W alu2_out[31:0]	256	65570	65571	*		-			65572					
> W rsOut[31:0]	X	256	X						x					
> W rtOut[31:0]	x							×						
> W rd_out_out[5:0]	x		0		1					z				
> W mux2_out[31:0]	x	×	2	56						×				
a ex_regWrite_out	X													
> W aluOp[3:0]	4		'					4						
le regWrite	1													
1₫ memtoReg	0													
¼ PCtoReg	1													
le branchN	0													
le branchZ	0													
1å jump	0	1												
le memRead	0													
le memWrite	0													
le jumpMem	0													
> W rsOut_out[31:0]	X	2	56						х					
> W rtOut_out[31:0]	X	1						×						
> W rd_out[5:0]	x		0	<u> </u>					z					
> W alu2_out_out[31:0]	x	65	570	65571	<u> </u>				65	572				
> W aluOp_out[3:0]	x							4						
						<u>'</u>	<u>'</u>	<u>'</u>				<u>'</u>		

Name		350 ns	360 ns	370 ns	380 ns	390 ns	400 ns	410 ns	420 ns	430 ns	440 ns	450 ns	460 ns	470 ns
> W rd_out(5:0)	x		o						Z					
> W alu2_out_out[31:0]	х	(5570	65571	X				65	572				
> W aluOp_out[3:0]	Х							4						
le memRead_out	х				ļ									
le memWrite_out	х				ļ									
le branchN_out	х				ļ		1		ļ			↓		
branchZ_out	х													
le jump_out	х						s							
PCtoReg_out	X													
le memtoReg_out	Х													
le regWrite_out	х													
le jumpMem_out	х								ļ					
> W dataMemoryOut[31:0]	х							256						
> W alu3_out[31:0]	х		256						x					
le alu3_N	Х													
le alu3_Z	Х													
> W ex_alu2_out_out(31:0)	Х	33	•	5570	65571	*				65572				
le ex_branchN_out	х													
le ex_branchZ_out	Х													
le ex_jump_out	Х													
& ex_PCtoReg_out	х													
le ex_memtoReg_out	Х													
le ex_nFlag_out	Х													
le ex_zFlag_out	Х													
and1_out	х													
and2_out	х													
> 🕨 [1][31:0]	х							256						
> 🕨 [2][31:0]	х							257						
> 🕨 [3][31:0]	х							-256						
> 🕨 [4][31:0]	Х							256						
> 🕨 [5][31:0]	х							513						
> 🕨 [6][31:0]	х							۰						
> 🕨 [256][31:0]														

Runtime Evaluation of CPU

Theoretical

Runtime of CPU = 1(delay of instruction memory) + 1(delay of data memory) + 1(delay of register file) + 3(delay of ALU)

$$= 1(2ns) + 1(2ns) + 1(1.5ns) + 3(2ns) = 10.5 \text{ ns}$$

10.5ns * 36 instructions = **378 ns**

Actual Runtime:

Total runtime = **360 ns**