Department of Electrical and Computer Engineering

Author: Zainab Hussein

Title: Multicycle Processor (Part 2)

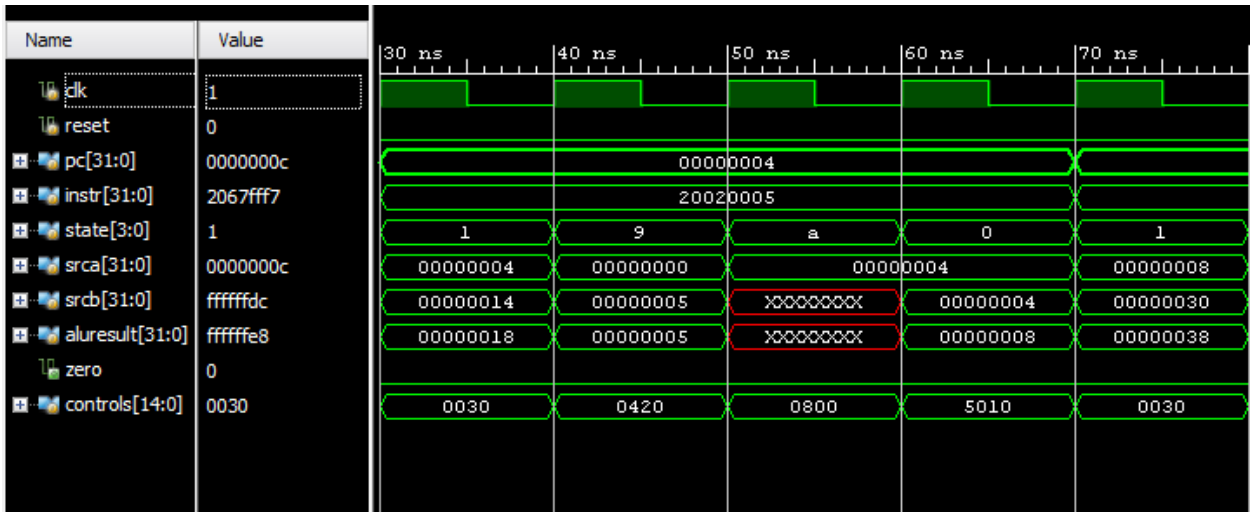
Date: 4-18-2017

1. Time spent: Lab period
2. Attached in last page
3. Datapath code

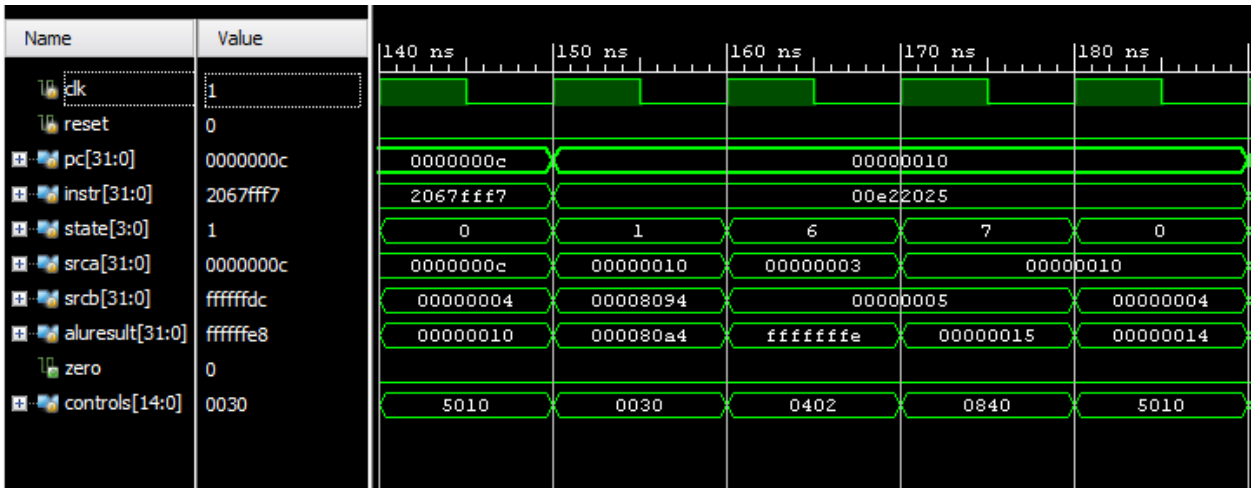
```
// The datapath unit is a structural verilog module.  That is,
// it is composed of instances of its sub-modules.  For example,
// the instruction register is instantiated as a 32-bit flopenr.
// The other submodules are likewise instantiated.
module datapath(input  logic        clk, reset,
                input  logic        pcen, irwrite, regwrite,
                input  logic        alusrca, iord, memtoreg, regdst,
                input  logic [1:0]  alusrcb, pcsrc,
                input  logic [2:0]  alucontrol,
                output logic [5:0]  op, funct,
                output logic        zero,
                output logic [31:0] adr, writedata,
                input  logic [31:0] readdata);
  // Below are the internal signals of the datapath module.
  logic [4:0]  writereg;
  logic [31:0] pcnext, pc;
  logic [31:0] instr, data, srca, srcb;
  logic [31:0] a;
  logic [31:0] aluresult, aluout;
  logic [31:0] signimm;   // the sign-extended immediate
  logic [31:0] signimmsh;  // the sign-extended immediate shifted left by 2
  logic [31:0] wd3, rd1, rd2;
  logic [31:0] pcjump;
  logic [27:0] shiftjump;
  // op and funct fields to controller
  assign op = instr[31:26];
  assign funct = instr[5:0];
```

```
  assign pcjump = {pc[31:28],shiftjump};

  // ADD CODE HERE

  // datapath

  flopenr #(32)  pcreg( .clk(clk), .reset(reset), .en(pcen), .d(pcnext), .q(pc) );

  mux2 #(32)     pcmux( .d0(pc), .d1(aluout), .s(iord), .y(adr) );

  flopenr #(32)
instreg( .clk(clk), .reset(reset), .en(irwrite), .d(readdata), .q(instr) );

  flopr #(32)    datreg( .clk(clk), .reset(reset), .d(readdata), .q(data) );

  regdstmux2 #(32)
regDestmux( .d0(instr[20:16]), .d1(instr[15:11]), .s(regdst), .y(wa3) );  mux2 #(32)
mem2regmux( .d0(aluout), .d1(data), .s(memtoreg), .y(wd3) );

  signext        signEXT( .a(instr[15:0]), .y(signimm) );

  regfile
REGFILE( .clk(clk), .we3(regwrite), .ra1(instr[25:21]), .ra2(instr[20:16]), .wa3(wa3),
 .wd3(wd3), .rd1(rd1), .rd2(rd2) );

  flopr #(32)    srcareg( .clk(clk), .reset(reset), .d(rd1), .q(a) );

  flopr #(32)    srcbreg( .clk(clk), .reset(reset), .d(rd2), .q(writedata) );

  mux2 #(32)     alusrcamux( .d0(pc), .d1(a), .s(alusrca), .y(srca) );

  sl2            immShifter( .a(signimm), .y(signimmsh) );

  mux4 #(32)
alusrcbmux( .d0(writedata), .d1(32'h0004), .d2(signimm), .d3(signimmsh), .s(alusrcb),
 .y(srcb) );

  alu
ALU( .a(srca), .b(srcb), .f(alucontrol), .y(aluresult), .zero(zero) );

  sl2            jumpShifter( .a(instr[25:0]), .y(shiftjump) );

  flopr #(32)    aluoutreg( .clk(clk), .reset(reset), .d(aluresult), .q(aluout) );

  mux3 #(32)
alucontrolMux( .d0(aluresult), .d1(aluout), .d2(pcjump), .s(pcsrc), .y(pcnext) );

endmodule
```

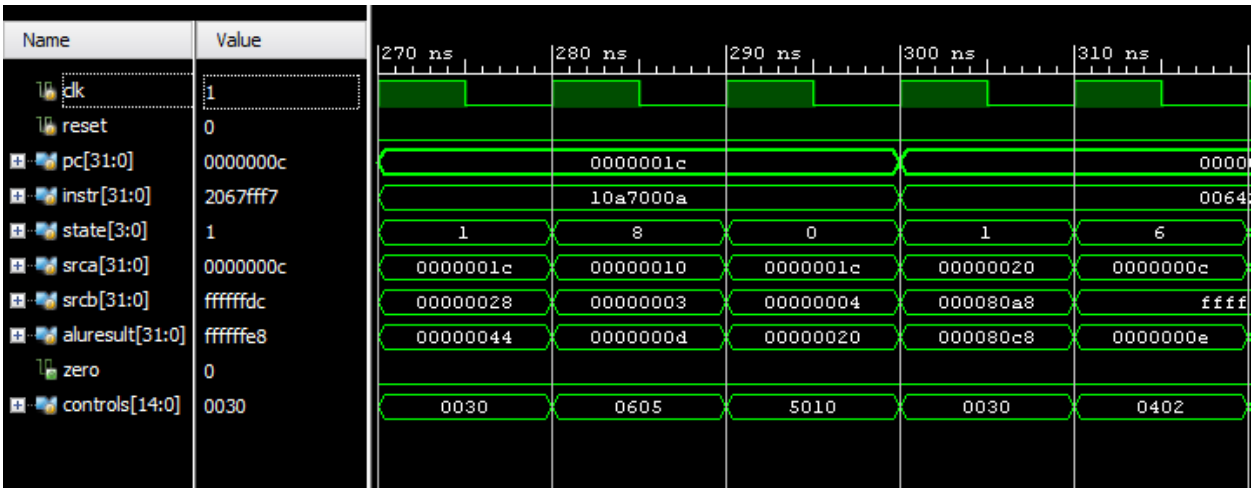
4. Simulation waveforms



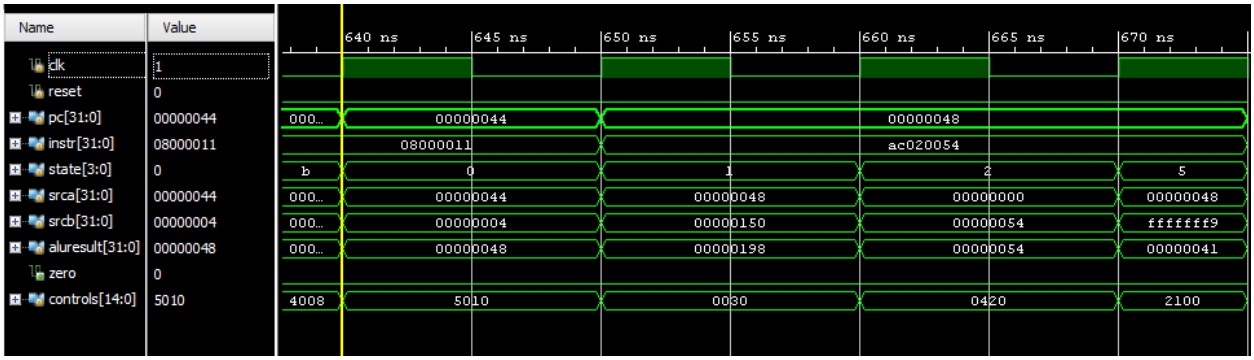
This addi instruction is as expected in the table.



This or instruction is as expected in the table



This beq instruction is as expected in the table, and shows the correct transition to the slt instruction.



This shows correct transition from the jump to sw instruction, as in the table

5. The results of the simulations match the expected results outlined in the prelab table of instruction trace. Yes the simulation indicates succeeded after I changed the final value at register 84 from 7 to -7.

Cycle	Reset	PC	Instr	(FSM) state	Src A	SrcB	ALUResult	Zero	Control Word
1	1	00	0	0	00	04	04	0	5010
2	0	04	addi 20020005	1	04	x	x	0	0030
3	0	04	addi 20020005	9	00	05	05	0	0420
4	0	04	addi 20020005	10	x	x	x	0	0800
5	0	04	addi 20020005	0	04	04	08	0	5010
6	0	08	addi 2003000c	1	08	x	x	0	0030
7	0	08	addi 2003000c	9	00	0c	0c	0	0420
8	0	08	addi 2003000c	10	x	x	x	0	0800
9	0	08	addi 2003000c	0	08	0c	0c	0	5010
10	0	0c	addi 2067fff7	1	0c	x	x	0	0030
11	0	0c	addi 2067fff7	9	0c	-9	03	0	0420
12	0	0c	addi 2067fff7	10	x	x	x	0	0800
13	0	0c	addi 2067fff7	0	0c	10	10	0	5010
14	0	10	or 00e22025	1	10	x	x	0	0030
15	0	10	or 00e22025	6	03	05	07	0	0402
16	0	10	or 00e22025	7	x	x	x	0	0840
17	0	10	or 00e22025	0	10	04	14	0	5010
18	0	14	and 00642824	1	14	x	x	0	0030
19	0	14	and 00642824	6	0c	07	04	0	0402
20	0	14	and 00642824	7	x	x	x	0	0840
21	0	14	and 00642824	0	14	04	18	0	5010
22	0	18	add 00a42820	1	18	x	x	0	0030
23	0	18	add 00a42820	6	04	07	0b	0	0402
24	0	18	add 00a42820	7	x	x	x	0	0840
25	0	18	add 00a42820	0	18	04	1c	0	5010
26	0	44	Beq 10a7000a	1	1c	28	44	0	0030
27	0	1c	Beq 10a7000a	8	0b	03	08	0	0605
28	0	1c	Beq 10a7000a	0	1c	04	20	0	5010
29	0	20	Slt 0064202a	1	20	x	x	0	0030
30	0	20	Slt 0064202a	6	0c	07	00	0	0402
31	0	20	Slt 0064202a	7	x	x	x	0	0840
32	0	20	Slt 0064202a	0	20	24	24	0	5010
33	0	24	Beq 10800001	1	24	04	28	1	0030
34	0	28	Beq 10800001	8	00	00	07	1	0605
35	0	28	Beq 10800001	0	28	04	2c	1	5010
36	0	2c	Slt 00e2202a	1	2c	x	x	0	0030
37	0	2c	Slt 00e2202a	6	2c	05	31	0	0402
38	0	2c	Slt 00e2202a	7	x	x	x	0	0840
39	0	2c	Slt 00e2202a	0	2c	04	30	0	5010
40	0	30	add 00853820	1	30	x	x	0	0030
41	0	30	add 00853820	6	01	0b	0c	0	0402
42	0	30	add 00853820	7	x	x	x	0	0840
43	0	30	add 00853820	0	30	04	34	0	5010
44	0	34	sub 00e23822	1	34	x	x	0	0030
45	0	34	sub 00e23822	6	0c	05	07	0	0402
46	0	34	sub 00e23822	7	x	x	x	0	0840
47	0	34	sub 00e23822	0	34	04	38	0	5010
48	0	38	sw ac670044	1	38	x	x	0	0030
49	0	38	sw ac670044	2	0c	44	50	0	0420
50	0	38	sw ac670044	5	x	x	x	0	2100
51	0	38	sw ac670044	0	38	04	3c	0	5010
52	0	3c	lw 8c020050	1	3c	x	x	0	0030
53	0	3c	lw 8c020050	2	00	50	50	0	0420
54	0	3c	lw 8c020050	3	x	x	x	0	0100
55	0	3c	lw 8c020050	4	x	x	x	0	0880
56	0	3c	lw 8c020050	0	3c	04	40	0	5010
57	0	40	J 08000011	1	40	44	84	0	0030
58	0	40	J 08000011	b	x	x	x	0	4008
59	0	40	J 08000011	0	40	48	48	0	5010
60	0	44	Sw ac020054	1	48	150	198	0	0030
61	0	44	Sw ac020054	2	00	54	54	0	0420
62	0	44	Sw ac020054	5	x	x	x	0	2100

Table 1. Expected Instruction Trace