Department of Electrical and Computer Engineering

Author: Zainab Hussein

Title: Multicycle Processor (Part 2)

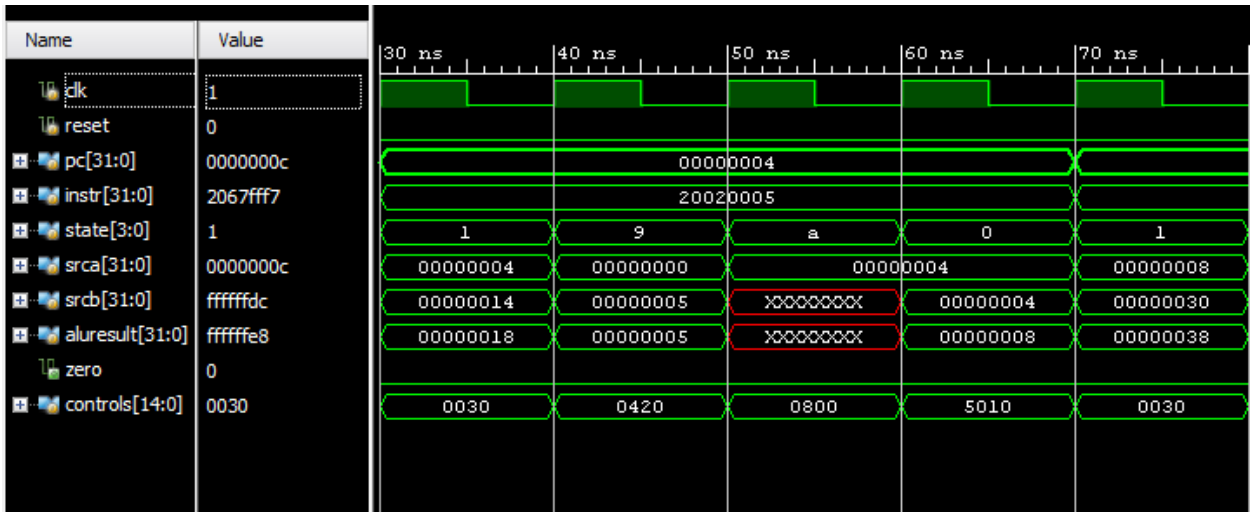
Date: 4-18-2017

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

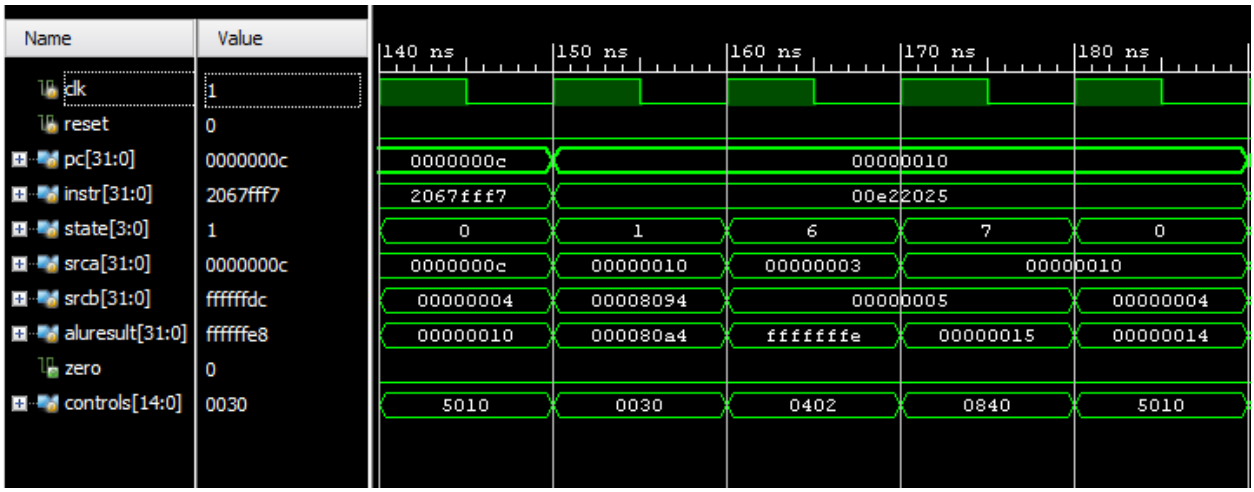
```
// The datapath unit is a structural verilog module.  That is,
// it is composed of instances of its sub-modules.  For example,
// the instruction register is instantiated as a 32-bit flopenr.
// The other submodules are likewise instantiated.
module datapath(input  logic        clk, reset,
                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
```

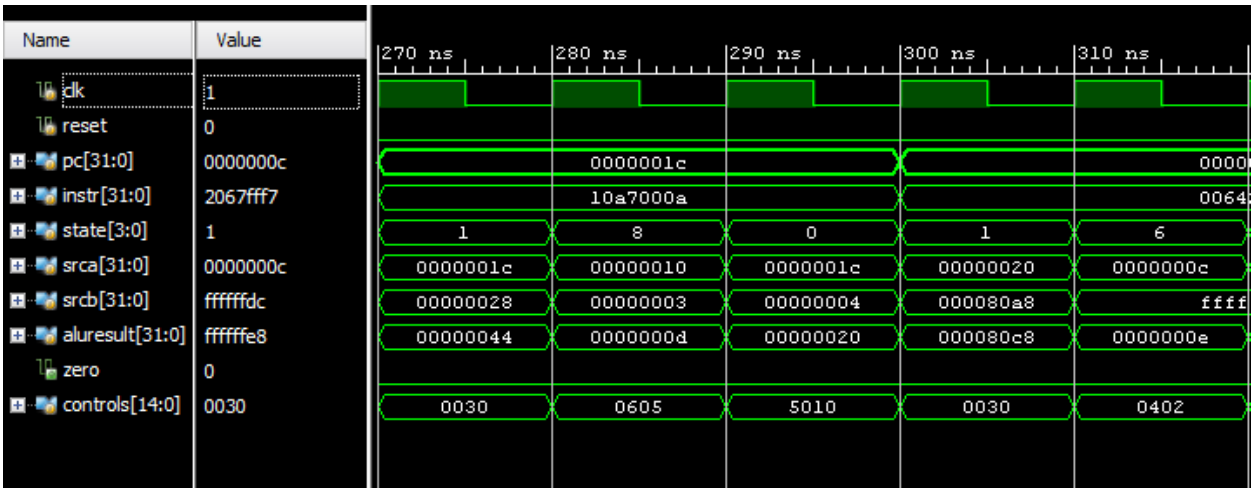
4. Simulation waveforms



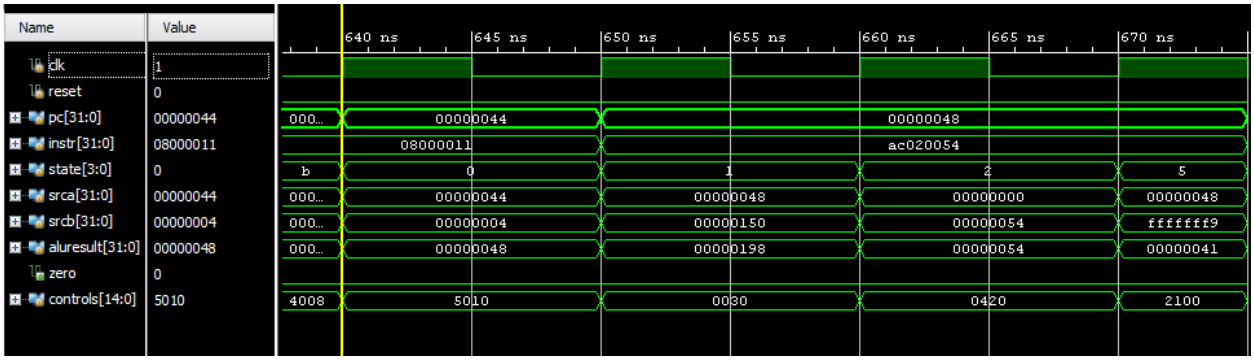
This addi instruction is as expected in the table.



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.