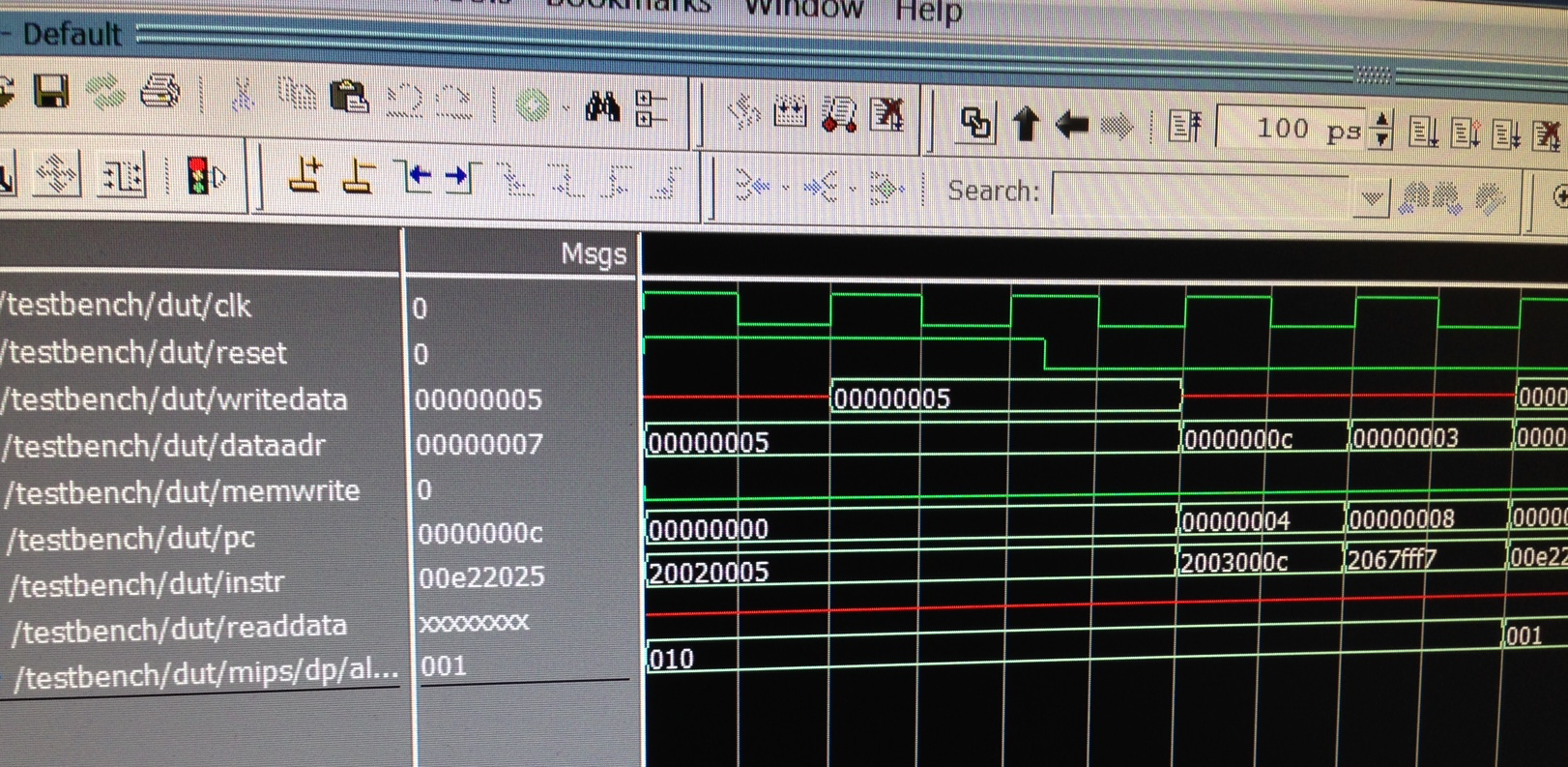
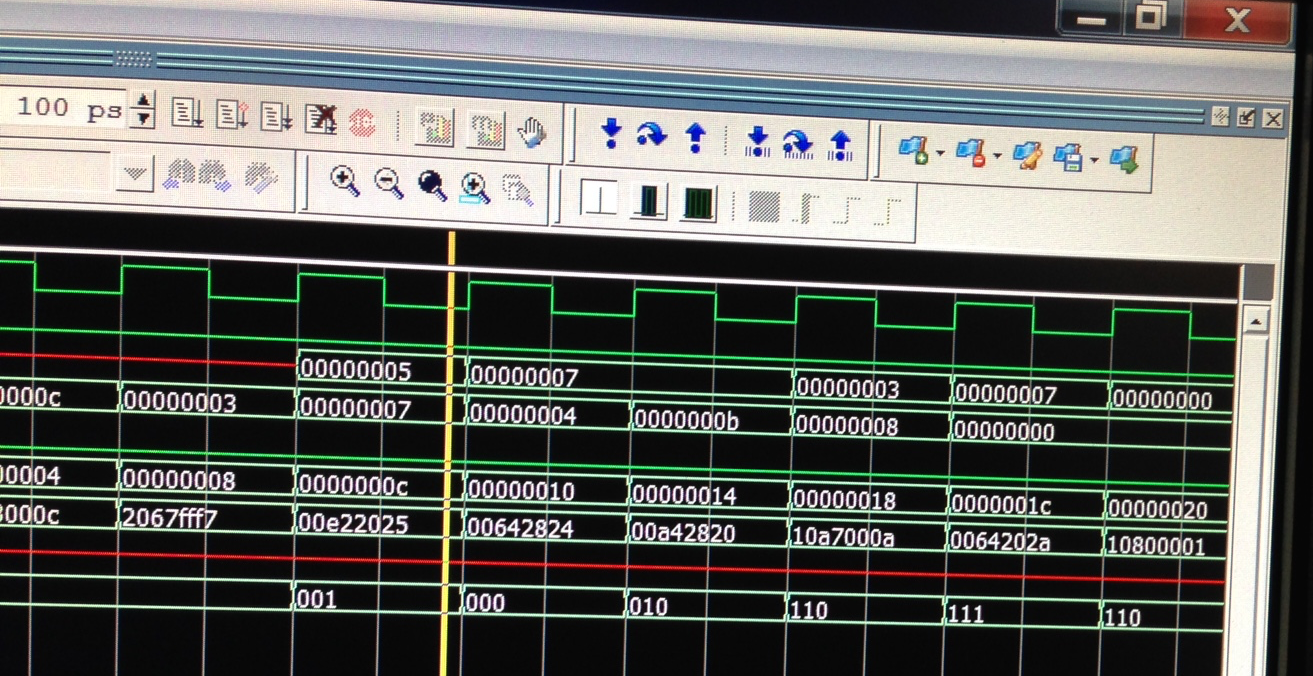
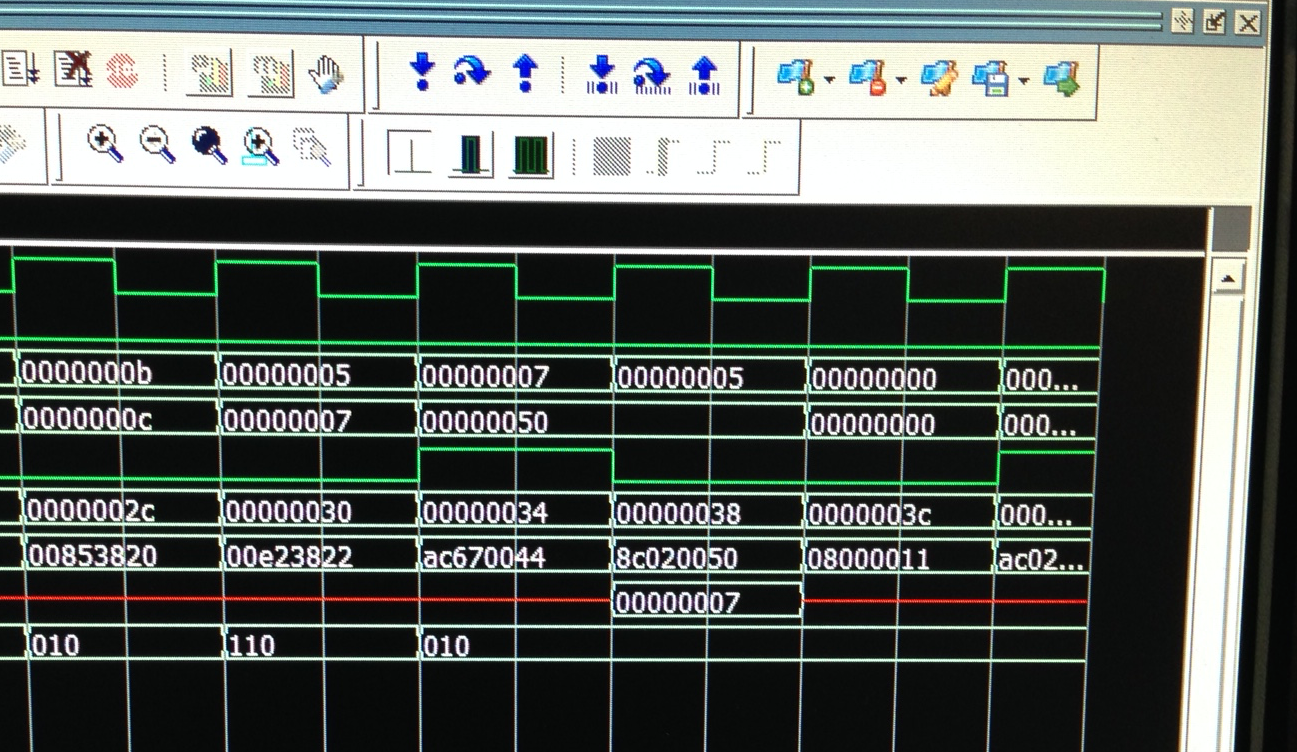
**Computer Engineering Lab C**

**Configuring MIPS Single Cycle Processor**

**Time spent on Lab:** 2 hours

**For Table from Waveforms:**



**Memfile:**

20020005

2003000c

2067fff7

00e22025

00642824

00a42820

10a7000a

0064202a

10800001

20050000

00e2202a

00853820

00e23822

ac670044

8c020050

08000011

20020001

ac020054

**Mips:**

module mips(input logic clk, reset,

output logic [31:0] pc,

input logic [31:0] instr,

output logic memwrite,

output logic [31:0] aluout, writedata,

input logic [31:0] readdata);

logic memtoreg, branch,

pcsrc, zero,

alusrc, regdst, regwrite, jump;

logic [2:0] alucontrol;

controller c(instr[31:26], instr[5:0], zero,

memtoreg, memwrite, pcsrc,

alusrc, regdst, regwrite, jump,

alucontrol);

datapath dp(clk, reset, memtoreg, pcsrc,

alusrc, regdst, regwrite, jump,

alucontrol,

zero, pc, instr,

aluout, writedata, readdata);

endmodule

module controller(input logic [5:0] op, funct,

input logic zero,

output logic memtoreg, memwrite,

output logic pcsrc, alusrc,

output logic regdst, regwrite,

output logic jump,

output logic [2:0] alucontrol);

logic [1:0] aluop;

logic branch;

maindec md(op, memtoreg, memwrite, branch,

alusrc, regdst, regwrite, jump,

aluop);

aludec ad(funct, aluop, alucontrol);

assign pcsrc = branch & zero;

endmodule

module maindec(input logic [5:0] op,

output logic memtoreg, memwrite,

output logic branch, alusrc,

output logic regdst, regwrite,

output logic jump,

output logic [1:0] aluop);

logic [8:0] controls;

assign {regwrite, regdst, alusrc,

branch, memwrite,

memtoreg, jump, aluop} = controls;

always\_comb

case(op)

6'b000000: controls <= 9'b110000010; //Rtype

6'b100011: controls <= 9'b101001000; //LW

6'b101011: controls <= 9'b001010000; //SW

6'b000100: controls <= 9'b000100001; //BEQ

6'b001000: controls <= 9'b101000000; //ADDI

6'b000010: controls <= 9'b000000100; //

default: controls <= 9'bxxxxxxxxx; //???

endcase

endmodule

module aludec(input logic [5:0] funct,

input logic [1:0] aluop,

output logic [2:0] alucontrol);

always\_comb

case(aluop)

2'b00: alucontrol <= 3'b010; // add

2'b01: alucontrol <= 3'b110; // sub

default: case(funct) // RTYPE

6'b100000: alucontrol <= 3'b010; // ADD

6'b100010: alucontrol <= 3'b110; // SUB

6'b100100: alucontrol <= 3'b000; // AND

6'b100101: alucontrol <= 3'b001; // OR

6'b101010: alucontrol <= 3'b111; // SLT

default: alucontrol <= 3'bxxx; // ???

endcase

endcase

endmodule

module datapath(input logic clk, reset,

input logic memtoreg, pcsrc,

input logic alusrc, regdst,

input logic regwrite, jump,

input logic [2:0] alucontrol,

output logic zero,

output logic [31:0] pc,

input logic [31:0] instr,

output logic [31:0] aluout, writedata,

input logic [31:0] readdata);

logic [4:0] writereg;

logic [31:0] pcnext, pcnextbr, pcplus4, pcbranch;

logic [31:0] signimm, signimmsh;

logic [31:0] srca, srcb;

logic [31:0] result; // next PC logic

flopr #(32) pcreg(clk, reset, pcnext, pc);

adder pcadd1(pc, 32'b100, pcplus4);

sl2 immsh(signimm, signimmsh);

adder pcadd2(pcplus4, signimmsh, pcbranch);

mux2 #(32) pcbrmux(pcplus4, pcbranch, pcsrc, pcnextbr);

mux2 #(32) pcmux(pcnextbr, {pcplus4[31:28], instr[25:0], 2'b00}, jump, pcnext); // register file logic

regfile rf(clk, regwrite, instr[25:21], instr[20:16], writereg, result, srca, writedata);

mux2 #(5) wrmux(instr[20:16], instr[15:11], regdst, writereg);

mux2 #(32) resmux(aluout, readdata, memtoreg, result);

signext se(instr[15:0], signimm);

// ALU logic

mux2 #(32) srcbmux(writedata, signimm, alusrc, srcb);

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

endmodule

**Alu code:**

module alu (input logic [31:0] a,

input logic [31:0] b,

input logic [2:0] f,

output logic [31:0] y,

output logic zero);

assign zero = (y==0);

always @(f, a, b) begin

case (f)

0: y<= a & b;

1: y<= a | b;

2: y<= a + b;

4: y<= a & (~b);

5: y<= a | (~b);

6: y<= a - b;

7: y<= a < b? 1: 0;

default: y<= 0;

endcase

end

endmodule

**Lab D report contains the other part of this report**