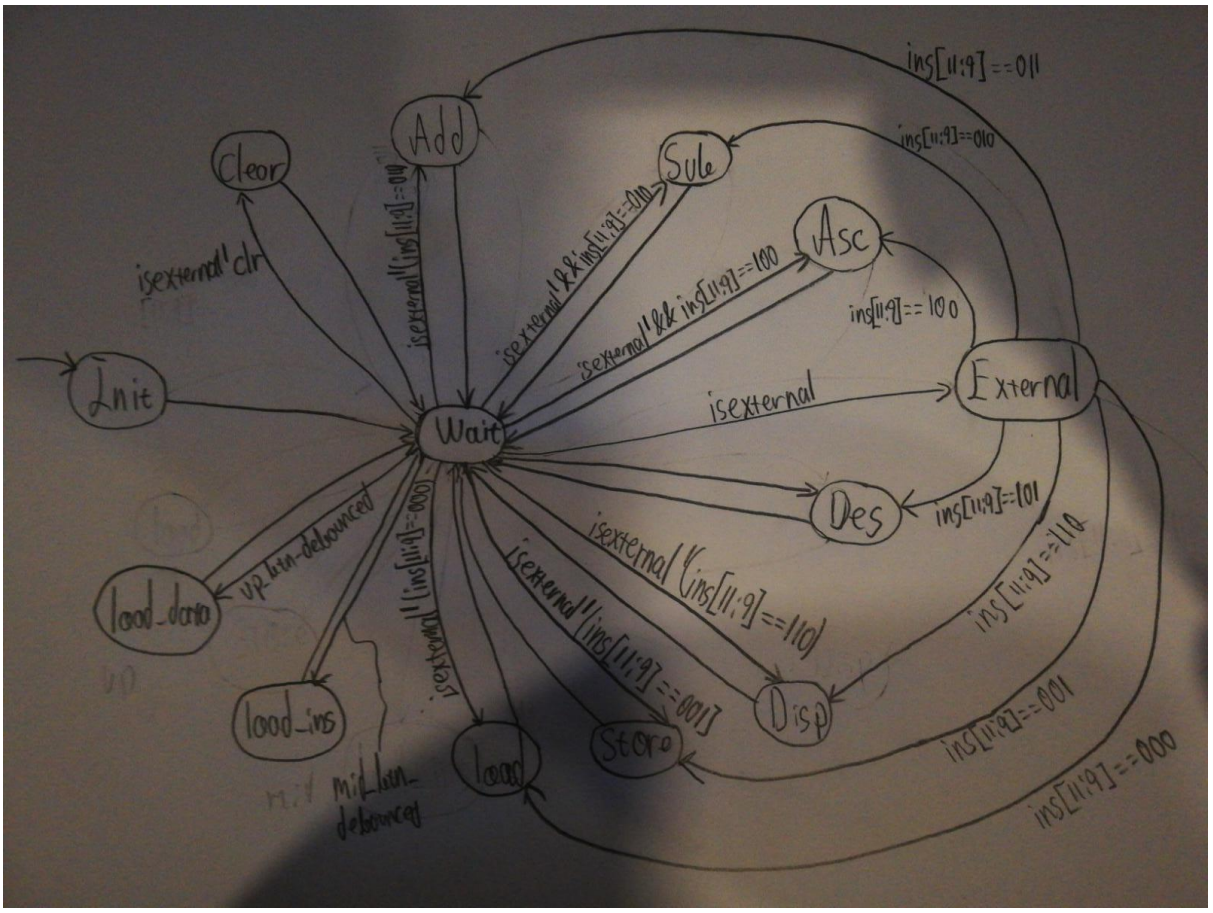
CS223
Digital Design
Sec. 1
Project

Erkin Aydın
ID: 22002956

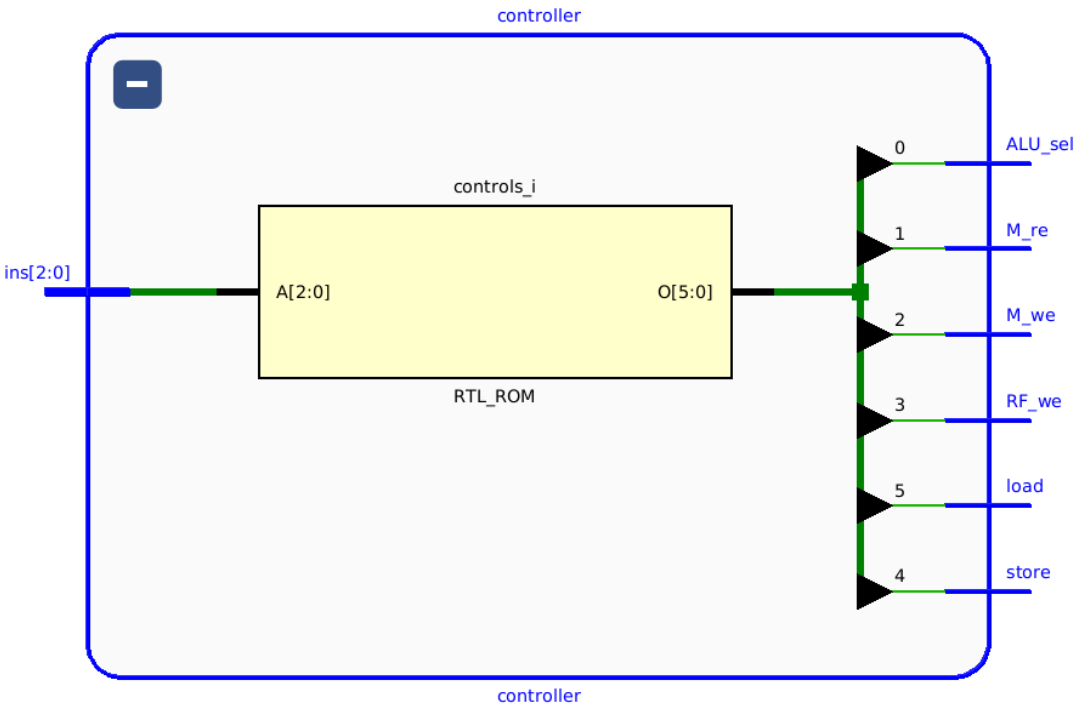
19/12/2022

RTL schematics for Ascending and Descending Sorting operations:

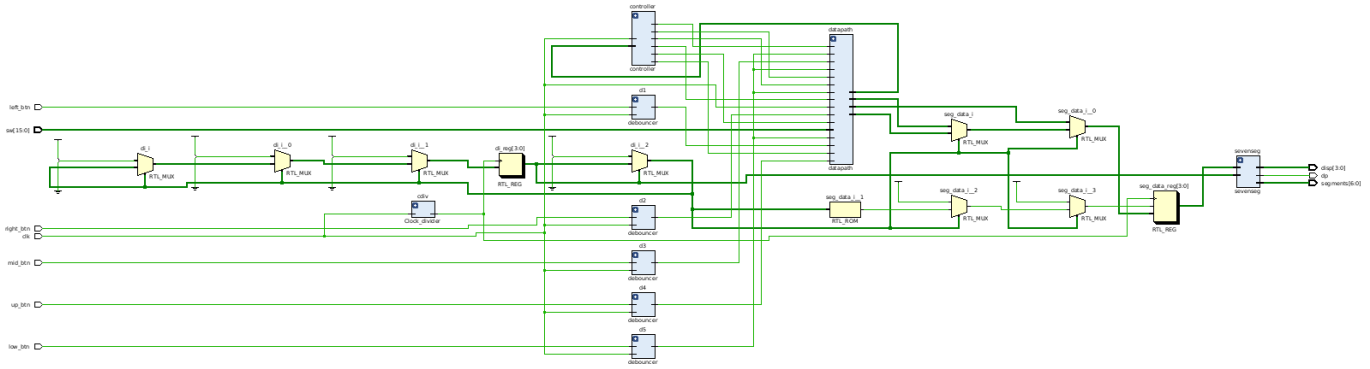
Controller High-Level State Machine Diagram



Controller Block Diagram



Controller/Datapath Top Module Block Diagram



Testbenchs

```
module processor_tb();

        logic clk, left_btn, mid_btn, right_btn, up_btn, low_btn;
        logic[15:0] sw;
        logic[6:0] segments;
        logic dp;
        logic[3:0] disp;

        always begin
        clk <= 0; #5;
        clk <= 1; #5;
        end

        processor pc(clk, left_btn, mid_btn, right_btn, up_btn, low_btn, sw, segments, dp,
disp);

        initial begin
        sw = 16'b0000_0110_0000_0000;
        left_btn = 0;
        mid_btn = 0;
        right_btn = 0;
        up_btn = 0;
        low_btn = 1;#10;
        low_btn = 0; #5;
        sw = 16'b0011_0110_0000_0000;
        up_btn = 1; #10;
        up_btn = 0; #10;
        sw = 16'b0001_1000_0000_0000;
        up_btn = 1; #10;
        up_btn = 0; #10;
        sw = 16'b1111_001_000_100_011;
        right_btn = 1; #10;
        right_btn = 0; #10;
        end
endmodule
```

```
module datapath_tb();

	logic clk, pc_we, pc_clr;
	logic IM_we, IM_clr;
	logic isexternal, load, store;
	logic[15:0] out_instruction;
	logic up_btn_debounced;
	logic RF_clr, RF_we;
	logic ALU_sel;
	logic M_clr, M_we, M_re;
	logic[11:0] ins;
	logic[6:0] seg1, seg2, seg3;

	datapath dp(clk, pc_we, pc_clr,
		IM_we, IM_clr,
		isexternal, load, store,
		out_instruction,
		up_btn_debounced,
		RF_clr, RF_we,
		ALU_sel,
		M_clr, M_we, M_re,
		ins,
		seg1, seg2, seg3);

	always begin
	clk <= 0; #5;
	clk <= 1; #5;
	end

	initial begin

	pc_we = 0;
	pc_clr = 0;
	IM_we = 0;
	IM_clr = 0;
	isexternal = 0;
	load = 0;
	store = 0;
	out_instruction = 16'b0000_0000_0000_0000;
	up_btn_debounced = 0;
	RF_clr = 0;
	RF_we = 0;
	ALU_sel = 0;
	M_clr = 0;
	M_we = 0;
	M_re = 0; #10;
	up_btn_debounced = 1;
	RF_we = 1;
```

```
        out_instruction = 16'b1111_1111_1111_1111; #10;
        end
endmodule

module dmem_tb();

        logic clk, M_clr, M_we, M_re;
        logic[3:0] M_add, M_wd, M_rd;

        always begin
        clk <= 0; #5;
        clk <= 1; #5;
        end
        dmem dmem(clk, M_clr, M_add, M_wd, M_we, M_re, M_rd);
        initial begin

        M_clr = 0;
        M_we = 0;
        M_re = 0;
        M_wd = 4'b0000; #10;
        M_add = 4'b0001;
        M_wd = 4'b1010; #5;
        M_we = 1; #5;
        M_we = 0; #5;
        M_re = 1; #5;
        M_re = 0;
        M_clr = 1; #20;
        M_clr = 0; #10;
        M_re = 1;

        end
endmodule

module imem_tb();

        logic clk, we, clr;
        logic[2:0] addr;
        logic[11:0] wd;
        logic[3:0] out;

        imem mem(clk, we, clr, addr, wd, out);
        always begin
        clk <= 0; #5;
        clk <= 1; #5;
        end

        initial begin
```

```
        addr = 3'b000;
        wd = 11'b010_1010_1010;
        we = 0;
        clr = 0; #5

        we = 1; #10;
        we = 0; #10;

        addr = 3'b111;
        we = 1; #5;
        we = 0;

        end
endmodule

module reg_file_tb();

        logic clk, RF_clr, RF_we;
        logic[2:0] RF_ad1, RF_ad2, RF_wa;
        logic[3:0] RF_wd, RF_d1, RF_d2;
        always begin
        clk <= 0; #5;
        clk <= 1; #5;
        end
        reg_file rf(clk, RF_clr, RF_ad1, RF_ad2, RF_wa, RF_wd, RF_we, RF_d1, RF_d2);

        initial begin

        RF_clr = 0;
        RF_we = 0;
        RF_ad1 = 3'b000;
        RF_ad2 = 3'b001;
        RF_wa = 3'b001;
        RF_wd = 4'b1010;#5;
        RF_we = 1; #10;
        RF_we = 0; #10;
        RF_clr = 1; #10;
        RF_clr = 0;


        end
endmodule

module buttoncheck(input logic btn, clk,
        output logic dp,
        output logic [6:0] segments,
        output logic[3:0] disp);
```

```
        logic[3:0] num;
        reg pb_d;
        logic p_up;
        logic p_down;
        sevenseg s(num, 4'b0101, dp, segments, disp);
        debouncer debounce(clk, btn, p_up);
        always begin@(posedge clk)
        if(p_up) begin
        num <= num + 4'b0001;
        end
        end
endmodule
```