CS223 – Digital Design

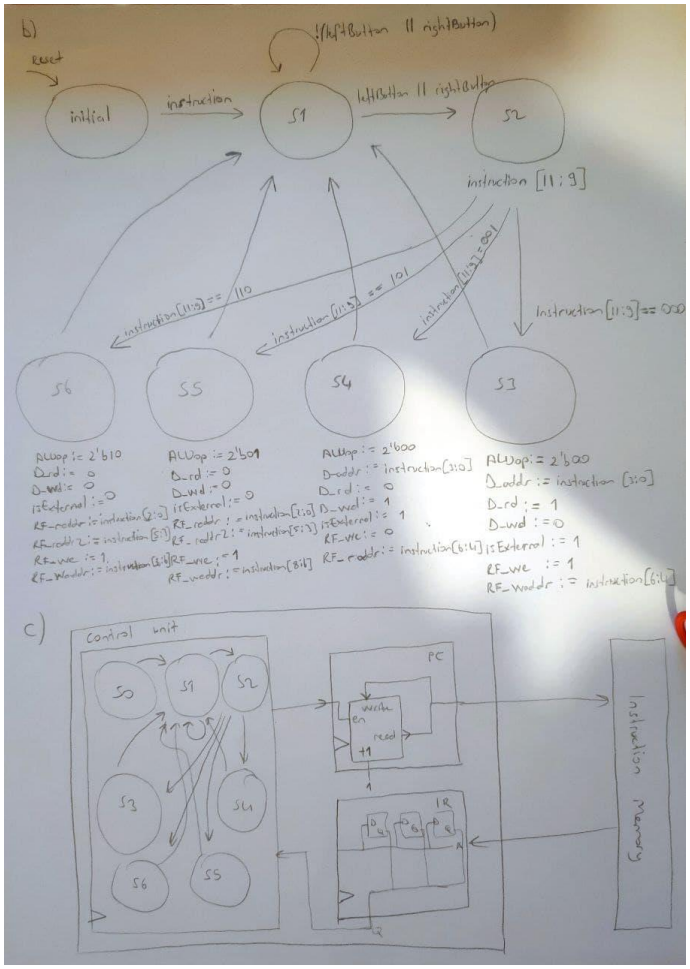
Ahmet Faruk Ulutaş

21803717

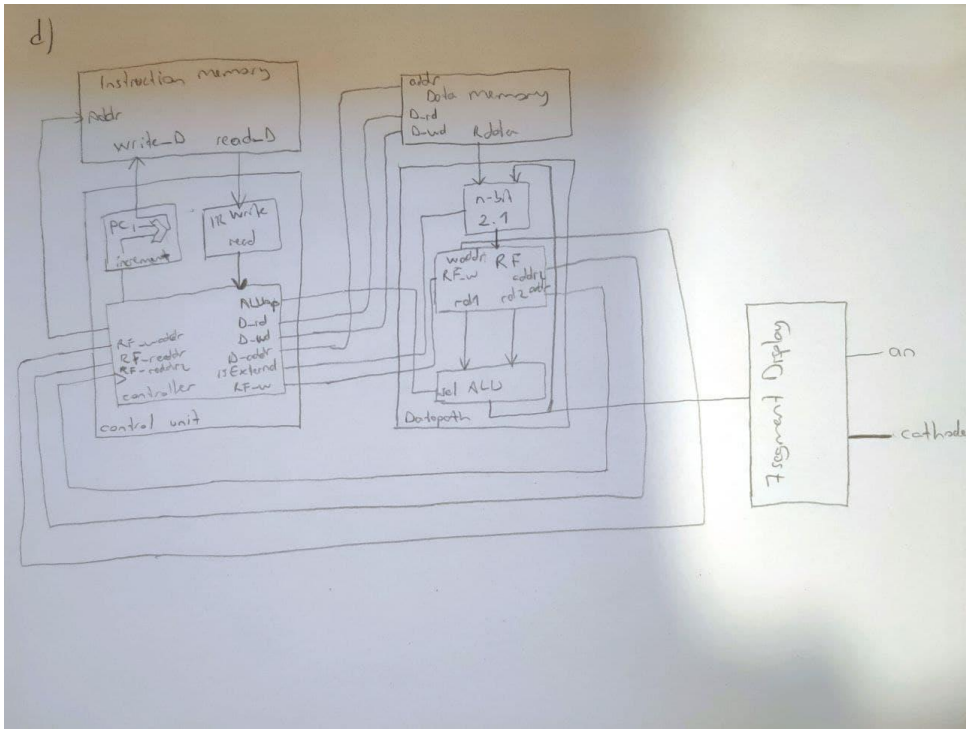
Section 1

Project Report

B) C)


b)
reset
initial
instruction
S1
!(leftButton || rightButton)
leftButton || rightButton
S2
instruction [11:9]
instruction[11:9] == 110
instruction[11:9] == 101
instruction[11:9] == 001
instruction[11:9] == 000
S6
S5
S4
S3
ALUop := 2'b10
D_rd := 0
D_wd := 0
isExternal := 0
RF_waddr := instruction[2:0]
RF_raddr1 := instruction[5:3]
RF_we := 1
RF_waddr := instruction[8:6]
ALUop := 2'b01
D_rd := 0
D_wd := 0
isExternal := 0
RF_raddr := instruction[2:0]
RF_raddr2 := instruction[5:3]
RF_we := 1
RF_waddr := instruction[8:6]
ALUop := 2'b00
D_addr := instruction[3:0]
D_rd := 0
D_wd := 1
isExternal := 1
RF_we := 0
RF_raddr := instruction[6:4]
ALUop := 2'b00
D_addr := instruction [3:0]
D_rd := 1
D_wd := 0
isExternal := 1
RF_we := 1
RF_waddr := instruction[6:4]
c)
control unit
S0
S1
S2
S3
S4
S5
S6
PC
write
en
read
+1
IR
D
A
Instruction Memory

D)



e)

```
module dataMem_tb();


logic clk, D_wr, D_rd;
logic [3:0] D_addr;
logic [3:0] W_data;
logic [3:0] R_data;


// instantiate device to be tested
dataMem dut (clk, D_wr, D_rd, D_addr, W_data, R_data );


// generate clock to sequence tests
always
   begin
      clk <= 1; # 5; clk <= 0; # 5;
   end


// check results
```

```
initial begin
D_wr = 1; D_rd = 0;
W_data = 4'b0000; D_addr = 4'b0000;
#10;
W_data = 4'b0010; D_addr = 4'b0001;
#10;
W_data = 4'b0100; D_addr = 4'b0010;
#10;
W_data = 4'b0110; D_addr = 4'b0011;
#10;
W_data = 4'b1000; D_addr = 4'b0100;
#10;
W_data = 4'b1010; D_addr = 4'b0101;
#10;
W_data = 4'b1100; D_addr = 4'b0110;
#10;
W_data = 4'b1110; D_addr = 4'b0111;
#10;
D_wr = 0; D_rd = 1;
W_data = 4'b0000; D_addr = 4'b0000;
#10;
W_data = 4'b0010; D_addr = 4'b0001;
#10;
W_data = 4'b0100; D_addr = 4'b0010;
#10;
W_data = 4'b0110; D_addr = 4'b0011;
#10;
W_data = 4'b1000; D_addr = 4'b0100;
#10;
W_data = 4'b1010; D_addr = 4'b0101;
#10;
```

```
W_data = 4'b1100; D_addr = 4'b0110;
#10;
W_data = 4'b1110; D_addr = 4'b0111;
#10;
end
Endmodule

module controller_tb();

   logic [11:0] instruction;
   logic [1:0] ALUop;
   logic D_rd;
   logic D_wd;
   logic [3:0] D_addr;
   logic isExternal;
   logic RF_we;
   logic [2:0] RF_waddr;
   logic [2:0] RF_raddr;
   logic [2:0] RF_raddr2;

   controller dut( instruction, ALUop, D_rd, D_wd, D_addr, isExternal, RF_we, RF_waddr, RF_raddr,
RF_raddr2);

   initial begin
      instruction = 12'b000_00_001_0010; //same
      #10;
      instruction = 12'b000_11_001_0010; //same
      #10;
      instruction = 12'b001_00_001_0100;
      #10;
      instruction = 12'b101_010_000_001;
```

```
    #10;
    instruction = 12'b110_010_000_001;
    #10;
  end

Endmodule

`timescale 1ns / 1ps

module instrMem_tb();
logic [2:0] address;
logic [11:0] instruction;

InstructionMemory IM2(address, instruction);

initial

  begin
  address = 3'b000; #10;
  address = 3'b001; #10;
  address = 3'b010; #10;
  address = 3'b011; #10;
  end

Endmodule

module instReg_tb();
logic clk;
logic reset;
logic [11:0] instr;
logic [11:0] instrOut;
```

```
instReg ireg(clk, reset, instr, instrOut);

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

initial

   begin
   reset = 1;
      instr = 12'b000000010010; # 10;
      instr = 12'b000110010010; # 10;
      instr = 12'b001000010100; # 10;
      instr = 12'b101010000001; # 10;
      instr = 12'b110010000001; # 10;
      instr = 12'b000000000000; # 10;
      instr = 12'b000000000000; # 10;
      instr = 12'b000000000000; # 10;
      instr = 12'b000000000000; # 10;
      instr = 12'b000000000000; # 10;
      instr = 12'b000000000000; # 10;
      instr = 12'b000000000000; # 10;

   end

Endmodule

module RegisterFile_tb();
logic clk;
```

```
logic writeEnable;
logic [2:0] writeAddress;
logic [2:0] readAddress1;
logic [2:0] readAddress2;
logic [3:0] writeData;
logic [3:0] readData1;
logic [3:0] readData2;

RegisterFile rf(clk, writeEnable, writeAddress, readAddress1, readAddress2, writeData, readData1, readData2);

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

  initial begin

  writeEnable = 1;
  writeAddress = 3'b000; readAddress1 = 3'b001; readAddress2 = 3'b010;
  writeData = 4'b1000; #10;
  writeAddress = 3'b001; readAddress1 = 3'b010; readAddress2 = 3'b011;
  writeData = 4'b1100; #10;
  writeAddress = 3'b010; readAddress1 = 3'b011; readAddress2 = 3'b100;
  writeData = 4'b1110; #10;

  writeEnable = 0;
  writeAddress = 3'b000; readAddress1 = 3'b000; readAddress2 = 3'b000;
  writeData = 4'b1000; #10;
  writeAddress = 3'b001; readAddress1 = 3'b001; readAddress2 = 3'b001;
  writeData = 4'b1100; #10;
```

```
    writeAddress = 3'b010; readAddress1 = 3'b010; readAddress2 = 3'b010;
    writeData = 4'b1110; #10;

    end
Endmodule

module PCTestBench();
logic clk;
logic reset;
logic increment;
logic set;
logic [2:0] address;
logic [11:0] instr;
logic [11:0] instrOut;
logic [2:0] addressOut;

PC PC(clk, reset, increment, set, address, instr, addressOut, instrOut);
    initial
    begin
        reset <= 1; # 22; reset <= 0;
    end

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



initial
```

```
    begin

    increment = 1; set = 0; address = 3'b000; # 10;
    increment = 1; set = 0;# 10;
    increment = 1; set = 0;# 10;
    increment = 0; set = 1;# 10;
    instr = 12'b111111111111; # 10;
    instr = 12'b111111111011; # 10;
    instr = 12'b111111111000; # 10;


end


endmodule
```