lab report 6

Andrew Nady

900184042

Question 2:
Code

Note: all code is included in another folder

Required code:

Risc-v Code

```
`timescale 1ns / 1ps
module Full_dataPath( input clk,input fpga, input rst ,input [1:0] LedSel ,
input [3:0] ssdSel , output reg [7:0] leds, output  [6:0]SSDout, output[3:0]
anodes);

wire [31:0] adder1,adder2,PCmux,PCout;
wire cout1_ripple1,cout1_ripple2;
wire [31:0]ints_out;
wire [31:0]readdata1, readdata2 ;
wire  [31:0] gen_out; //should be reg || wire
wire branch;
 wire  memread;
  wire  memtoreg;
 wire [1:0]aluop;
  wire memwrite;
   wire  alusrc;
    wire regwrite;
    wire [3:0]aluS;
    wire [31:0]outmux_input_alu;
    wire [31:0]ALU_result;
    wire zero;
    wire [31:0] dataMem_out;
    wire [31:0] writingData;
    wire [31:0] shiftout;
    wire S;
    reg [12:0]num;

//instantiate 32bitreg PCin PCout
OneReg  pc( clk,1, PCmux, rst,  PCout );
ripple_carry  ripplecar0(PCout,
32'd4,
0,
adder1,
cout1_ripple1
);
//adder tany
ripple_carry  ripplecar1(PCout,
shiftout,
0,
adder2,
```

```
cout1_ripple2
);

//mux
ThirtytwoMUX  mux3(adder1,adder2,branch&zero ,PCmux);
//register (PCmux, PCin)  //reload the PC

//instmem
InstMem instmem(PCout[7:2],ints_out);
//control unit
Control_unit  controlunit(ints_out[6:2], branch,
   memread, memtoreg,
aluop, memwrite,  alusrc, regwrite );
//alu_control
ALU_Control   aluControl(aluop,  ints_out[14:12],ints_out[30] ,aluS);
//registerfile
RegisterFile regfile( clk, rst,
ints_out[19:15], ints_out[24:20], ints_out[11:7],
 writingData,
 regwrite,
 readdata1, readdata2  );
//immGenerator
immediate_generator   immgen( gen_out, ints_out );
//shifting left
 shiftLeft sh(gen_out ,shiftout);
//mux
 ThirtytwoMUX mux (readdata2,gen_out,alusrc ,outmux_input_alu);
 //ALU
 ALU alu(
 readdata1, outmux_input_alu,
 aluS,
 ALU_result,  zero );
 //dataMem
 DataMem  datamem( clk, memread, memwrite,
  ALU_result[7:2] , readdata2,  dataMem_out);
//mux after the datamem
ThirtytwoMUX mux_writing (ALU_result,dataMem_out,memtoreg ,writingData);



//switch or if on input ledsel to generate outputs leds
always @(*)
begin
        case(LedSel)
        2'b00 :
        begin
         leds[0]= regwrite;
         leds[1]= alusrc;
         leds[3:2]=  aluop;
         leds[4]=  memread;
         leds[5]= memwrite;
         leds[6]=  memtoreg;
         leds[7]=  branch;
```

```
        end
        2'b01:
        begin
        leds[3:0]=aluS;
        leds [4]=zero;
        leds[5]=branch&zero;
        leds[7:6] =0;

        end
        2'b11:
             leds[7:0]= {0,ints_out[6:0]}; //  monotring the opcode
        ////switch or if on input ssdsel to generate ssdnumber
        default : leds[7:0]= 0;
endcase

end
always@(*) begin
case (ssdSel)
4'b0000:num = PCout;
4'b0001: num =PCout+4;
4'b0010:  num = adder2;
4'b0011: num = PCmux;
4'b0100: num = readdata1;
4'b0101: num= readdata2;
4'b0110: num=  writingData;
4'b0111: num= gen_out;
4'b1000: num= shiftout;
4'b1001:num=outmux_input_alu;
4'b1010: num= ALU_result;
4'b1011:num=dataMem_out;
default : num = 0;
endcase
  end
//instantiate 7 seg

seven ssd ( fpga, num,anodes,SSDout);

endmodule
```

Top_module -v Code

```
`timescale 1ns / 1ps
module Top_module(input clk,input uart_in,
 output [6:0]SSDout,
 output [7:0]leds,
 output wire [7:0]leds2,
  output [3:0]Anode);

// wire [7:0]leds2;
//assign sAnode=4'b1111;
// assign Anode=4'b0000;
```

```
 UART_receiver_multiple_Keys keyys(
    clk,
    uart_in, // input receiving data line  ,
     leds // output
  );

   Full_dataPath riscv(  leds[0],clk ,  leds[1] ,leds [3:2] ,
   leds [7:4], leds2,   SSDout,  Anode);
endmodule
```

Constrain file:

```
set_property PACKAGE_PIN E3 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property PACKAGE_PIN C4 [get_ports uart_in]
set_property IOSTANDARD LVCMOS33 [get_ports uart_in]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets leds[0]]




#set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports {
LED[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
#set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports {
LED[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
#set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports {
LED[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
#set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports {
LED[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
#set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports {
LED[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
#set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {
LED[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
#set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports {
LED[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
#set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {
LED[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
#set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {
LED[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
#set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports {
LED[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
#set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {
LED[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
#set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports {
LED[11] }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
#set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {
LED[12] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
#set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports {
LED[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
```

```
#set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports {
LED[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
#set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 } [get_ports {
LED[15] }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]
set_property PACKAGE_PIN U16 [get_ports leds[7]]
set_property IOSTANDARD LVCMOS33 [get_ports leds[7]]
set_property PACKAGE_PIN U17 [get_ports leds[6]]
set_property IOSTANDARD LVCMOS33 [get_ports leds[6]]
set_property PACKAGE_PIN V17 [get_ports leds[5]]
set_property IOSTANDARD LVCMOS33 [get_ports leds[5]]
set_property PACKAGE_PIN R18 [get_ports leds[4]]
set_property IOSTANDARD LVCMOS33 [get_ports leds[4]]
set_property PACKAGE_PIN N14 [get_ports leds[3]]
set_property IOSTANDARD LVCMOS33 [get_ports leds[3]]
set_property PACKAGE_PIN J13 [get_ports leds[2]]
set_property IOSTANDARD LVCMOS33 [get_ports leds[2]]
set_property PACKAGE_PIN K15 [get_ports leds[1]]
set_property IOSTANDARD LVCMOS33 [get_ports leds[1]]
set_property PACKAGE_PIN H17 [get_ports leds[0]]
set_property IOSTANDARD LVCMOS33 [get_ports leds[0]]


set_property PACKAGE_PIN V16 [get_ports leds2[7]]
set_property IOSTANDARD LVCMOS33 [get_ports leds2[7]]
set_property PACKAGE_PIN T15 [get_ports leds2[6]]
set_property IOSTANDARD LVCMOS33 [get_ports leds2[6]]
set_property PACKAGE_PIN U14 [get_ports leds2[5]]
set_property IOSTANDARD LVCMOS33 [get_ports leds2[5]]
set_property PACKAGE_PIN T16 [get_ports leds2[4]]
set_property IOSTANDARD LVCMOS33 [get_ports leds2[4]]
set_property PACKAGE_PIN V15 [get_ports leds2[3]]
set_property IOSTANDARD LVCMOS33 [get_ports leds2[3]]
set_property PACKAGE_PIN V14 [get_ports leds2[2]]
set_property IOSTANDARD LVCMOS33 [get_ports leds2[2]]
set_property PACKAGE_PIN V12 [get_ports leds2[1]]
set_property IOSTANDARD LVCMOS33 [get_ports leds2[1]]
set_property PACKAGE_PIN V11 [get_ports leds2[0]]
set_property IOSTANDARD LVCMOS33 [get_ports leds2[0]]



#seven
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports {
SSDout[6] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports {
SSDout[5] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports {
SSDout[4] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports {
SSDout[3] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports {
SSDout[2] }]; #IO_L13P_T2_MRCC_14 Sch=ce
```

```
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports {
SSDout[1] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports {
SSDout[0] }]; #IO_L4P_T0_D04_14 Sch=cg



set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports {
Anode[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports {
Anode[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports {
Anode[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports {
Anode[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
#set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports {
sAnode[0] }]; #IO_L8N_T1_D12_14 Sch=an[4]
#set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports {
sAnode[1] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
#set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports {
sAnode[2] }]; #IO_L23P_T3_35 Sch=an[6]
#set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports {
sAnode[3] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]
```

Question 3:
assembly code && binary

Assembly

```
.text
main:


lw t4, 0(zero)      #pc=0
lw t5, 4(zero)      #pc=4
lw t6, 8(zero)      #pc=8
loop:
beq t4,t6,exit      #pc=12
add  t4,t4,t5        #pc=16
beq zero,zero,loop    #pc=20

exit:
sw t4, 12(zero).       #pc=24
```

Assembly

```
00000000000000000010111010000011
00000000100000000010111100000011
00000000100000000010111110000011
00000001111111101000011001100011
00000001111011101000111010110011
11111110000000000000110011100011
00000001110100000010011000100011
```

No unique task, we just wrote this code to test the jumbing and if it will loop or not

Question 4:

Experiment 5

LedSel

pc	00	01	11
0	11001010	01001000	11000000
4	11001010	01000000	11000000
8	11001010	01000000	11000000
12	00100011	01100000	11000110
16	10010000	01000000	11001100
20	00100011	01101100	11000110
24	01000110	01000000	11000100

SSDsel

PC	PC+4	Bramch adder	pcMux	Readdata1	Readdata2	Writing data	gen_out	shiftout	outmux_input_alu	ALU result	dataMem_out
0	1	10	11	100	101	110	111	1000	1001	1010	1011
0	4	0	4	0	0	0	0	0	0	0	0
4	8	12	8	0	0	1	4	8	4	4	1
8	12	24	12	0	0	10	8	16	8	8	10
12	16	24	16	0	10	0	6	12	10	-4 Unsigned= (8182)	0
16	20	74	20	0	1	1	29	58	1	1	0
20	24	12	12	0	0	0	-4 Unsigned= (8182)	-8 Unsigned= (8182)	0	0	0
loop											
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
24	28	48	28	0	10	0	12	24	12	12	0

Testing on FPGA:

we tested this program on the FPGA so with the same sequence of the SSDsel as the number is displayed on the FPGA and the leds will be the same as the above table of ledSel, so I will include some screenshots that I took from the data above.

instruction -> lw t4, 0(zero) #pc=0

As shown above that when the LEDsel =00, then 11001010.

Where

leds[0]= 1;
leds[1]= 1;
leds[3:2]= 00;
leds[4]= 1;
leds[5]= 0;
leds[6]= x;
leds[7]= 0;

```
begin
leds[0]= regwrite;
leds[1]= alusrc;
leds[3:2]= aluop;
leds[4]= memread;
leds[5]= memwrite;
leds[6]= memtoreg;
leds[7]= branch;
end
```

As shown above that when the LEDsel =01, then 01001000.

Where

leds[0]= 0;
leds[1]= 1;
leds[3:2]= 00; //where alus 0010
leds[4]= 1; //zero flag =1
leds[5]= 0; //branch =0
leds[6]= 0;
leds[7]= 0;

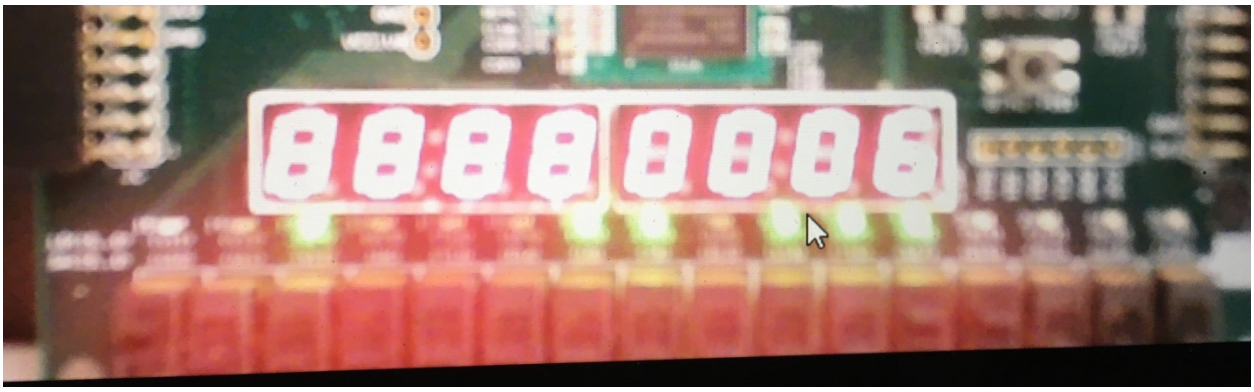
```
begin
        leds[3:0]=aluS;
        leds [4]=zero;
        leds[5]=branch&zero;
        leds[7:6] =0;
        End
```

leds[7:0]= {0,ints_out[6:0]}; // monitoring the opcode
The output should be 1100 0000 where ledSel=11

instruction -> beq t4,t6,exit #pc=12

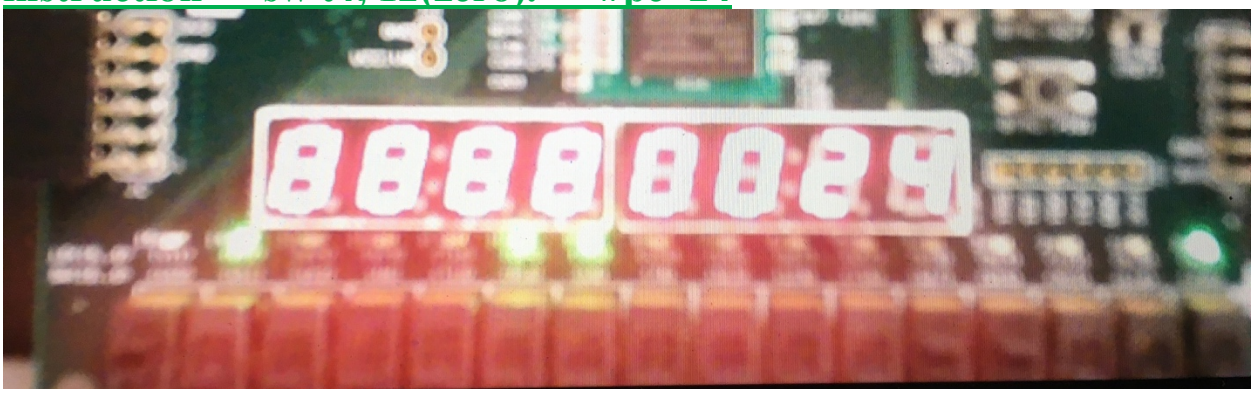
As shown, SSDsel=0000,PC=12,
Ledsel=00, then leds =00100011

```
leds[0]= 0;
leds[1]= 0;
leds[3:2]=  01;
leds[4]=  0;   //memread=0
leds[5]= 0;  //memwrite=0
leds[6]=  1  //memreg=x
leds[7]=  1;  //beanch =1
```



SSD=111, Imm gen =6

instruction -> sw t4, 12(zero). #pc=24



As shown, SSDsel=0000,PC=24,
Ledsel=11, then leds =0100 0110

```
Leds[0]= 0;
leds[1]= 1;
leds[3:2]=  00;
leds[4]=  0;   //memread=0
leds[5]= 1;  //memwrite=0
leds[6]=  1  //memreg=1
leds[7]=  0;  //beanch =1
```

As shown, SSDsel=0101,read data 2=10,

Ledsel=11,
leds[7:0]= {0,ints_out[6:0]};
Opcode =11000100