Single-Cycle MIPS Processor
Digital Logic Design Lab
Adding Branch Logic to the Datapath

EECE 2323 SEC 02
Professor Thomas R. Consi

Eli MacColl
maccoll.e@northeastern.edu

Abstract:
This lab introduced the implementation of a Program Counter (PC) that supports branching as well as the incorporation of a seven segment display (SSD) to the datapath to show the output of the ALU operations. The PC is designed to iterate through the COE file containing machine code instructions generated from a set of assembly instructions that demonstrate the circuit's capability to perform signed multiplication. These additions conclude the development of a functional computer. The hardware was implemented to generate all the control logic with the VIO interface and SSD being used to observe and verify the instructions as they are carried out.

Introduction:

The purpose of this lab was to incorporate branching functionality into the PC design code and introduce the use of a seven segment display (SSD) to exhibit the ALU output from instructions. Building upon the circuit from the previous labs, by implementing these components and changes, we have a completed computer. We altered the PC design code that simply incremented through the instructions to be able to branch when instructed. To implement the SSD, the datapath was rewired in the top file such that output from the ALU is displayed and updated for each instruction. The hardware was used to generate all of the control logic, push button 0 (BTN0) resets the register file and PC and push button 1 (BTN1) increments the PC, branching when instructed. The VIO and SSD are used to verify the instructions as they are carried out, confirming it was done properly.

Results:

Entering the Design

For the rewiring of this circuit to implement branching to the PC design module, we were provided the outline of a top file that required us to complete the instantiations of all of the modules according to the provided wiring diagram. This involved altering the beginning of the datapath, the PC logic, by adding two inputs, an 8-bit immediate signal from J-type instructions and a 1-bit take_branch signal from the ALU. These were added alongside the debounced clock and reset signals.

PC Code:

```
module program_counter(input wire pb_clk_debounced,
        input wire rst_general,
        input wire [7:0] immediate,
        input wire [7:0] take_branch,
        output reg [7:0] pc);

 always @(posedge pb_clk_debounced or posedge rst_general) begin
   if(rst_general)
      pc <= 8'b00000000;
   else
      if(pb_clk_debounced)
         if(take_branch)
            pc <= pc+immediate;
         else
            pc <= pc+1;
 end
endmodule
```

Top file instantiations:

```
// Debounce
debounce debounce_clk(
.clk_in(clk),
.rst_in(rst_general),
.sig_in(top_pb_clk),
.sig_debounced_out(pb_clk_debounced)
);
// 7-Segment display module
Adaptor_display display(.clk(clk),              // system clock
  .input_value(alu_output),                     // 8-bit input [7:0] value to display
  .disp_en(disp_en),                            // output [3:0] 7 segment display enable
  .seg7_output(seg7_output));                   // output [6:0] 7 segment signals
/* Instantiate all other modules */
   program_counter p_counter(
      .pb_clk_debounced(pb_clk_debounced),
      .rst_general(rst_general),
      .immediate(immediate),
      .take_branch(take_branch),
      .pc(pc));

   instr_mem instruction_memory (
      .a(pc),      // input wire [7 : 0] a
      .spo(instruction)  // output wire [15 : 0] spo
      );
  //**********Instantiate Your instruction decoder here**********//
   inst_decoder decoder(
      .instruction(instruction),
      .RegWrite(RegWrite),
      .RegDst(RegDst),
      .ALUSrc1(ALUSrc1),
      .ALUSrc2(ALUSrc2),
      .ALUOp(ALUOp),
      .MemWrite(MemWrite),
      .MemtoReg(MemToReg),
      .opcode(opcode),
      .rs_addr(rs_addr),
      .rd_addr(rd_addr),
      .rt_addr(rt_addr),
      .immediate(immediate));
//**********Instantiate Your alu-regfile here**********//
   eightbit_alu alu(
      .sel(ALUOp),
      .a(alu_input1),
      .b(alu_input2),
      .f(alu_output),
      .ovf(alu_ovf),
      .take_branch(take_branch)
      );
   reg_file register_file(
      .rst(rst_general),
      .clk(pb_clk_debounced),
      .wr_en(RegWrite),
      .rd0_addr(rs_addr),
      .rd1_addr(rt_addr),
      .wr_addr(regfile_write_address),
      .wr_data(regfile_write_data),
      .rd0_data(read_data1),
      .rd1_data(read_data2)
```

```
        );
 //**********Instantiate Your data memory here**********//
data_memory data_memory(
          .a(alu_output),      // input wire [7 : 0] a
          .d(read_data2),      // input wire [8 : 0] d
          .clk(pb_clk_debounced),  // input wire clk
          .we(MemWrite),    // input wire we
          .spo(data_mem_out) // output wire [8 : 0] spo
          );
//**********Mux for regfile_writedata**********//
 MUX1 MUX_1(
        .ALUSrc1(ALUSrc1),
        .ReadData1(read_data1),
        .input1(alu_input1),
        .zero_register(zero_register)
        );
     MUX2 MUX_2(
        .ALUSrc2(ALUSrc2),
        .ReadData2(read_data2),
        .Instr_i(immediate),
        .input2(alu_input2)
        );
     MUX3 MUX_3(
        .DataMemOut(data_mem_out),
        .result(alu_result),
        .MemtoReg(MemToReg),
        .WriteData(regfile_write_data)
        );
     //**********Mux for RegDST**********//
     MUX4 MUX_4(
        .RegDst(RegDst),
        .rd_addr(rd_addr),
        .rt_addr(rt_addr),
        .WriteAddr(regfile_write_address)
        );
      //********** Instantiate the VIO here **********//
     vio_0 vio(
                      .clk(clk),
          .probe_in0(alu_output),
          .probe_in1(alu_ovf),
          .probe_in2(take_branch),
          .probe_in3(read_data1),
          .probe_in4(read_data2),
          .probe_in5(alu_input1),
          .probe_in6(alu_input2),
          .probe_in7(regfile_write_data),
          .probe_in8(data_mem_out),
          .probe_in9(opcode),
          .probe_in10(rs_addr),
          .probe_in11(rt_addr),
          .probe_in12(rd_addr),
          .probe_in13(immediate),
          .probe_in14(RegDst),
          .probe_in15(RegWrite),
          .probe_in16(ALUSrc1),
          .probe_in17(ALUSrc2),
          .probe_in18(ALUOp),
          .probe_in19(MemWrite),
          .probe_in20(MemToReg),
          .probe_in21(pc),
          .probe_in22(instruction)
```

);

Using the provided assembler code, the generated COE file contains instructions to carry out signed multiplication with two positive numbers, a positive and negative number, and two negative numbers. The instructions were first written in assembly and then inputted into the assembler to produce the machine code. Below is a screenshot of the outputted COE file containing the machine code and a table that shows the corresponding assembly instruction.

COE file contents:

```
memory_initialization_radix=16;
memory_initialization_vector=3105,3204,2dc0,3001,c2fe,d000,d3c0,31fa,3203
,2dc0,3001,c2fe,d000,d3c0,31f7,3203,2dc0,3001,c2fe,43c0,3f01;
```

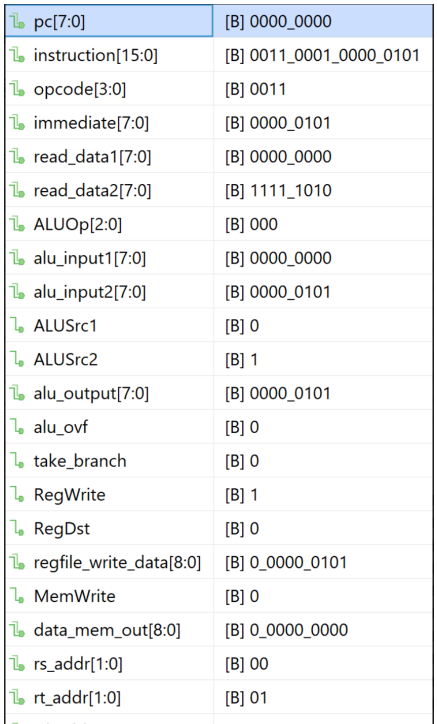
Assembly instruction	Machine code (Hexadecimal)	16-bit Binary instruction	opcode	Source register(s)	Destination or memory base address	Immediate or padding
addi $1, $0, 0x05	3105	0011000100000101	0011 - addi	00 - $0	01 - $1	00000101 = 0x05
addi $2, $0, 0x04	3204	0011001000000100	0011 - addi	00 - $0	10 - $2	00000100 = 0x04
mult1:						
add $3, $3, $1	2DC0	0010110111000000	0010 - add	11 - $3 01 - $1	11 - $3	000000 - padding
addi $0, $0, 0x01	3001	0011000000000001	0011 - addi	00 - $0	00 - $0	00000001 = 0x01
bne $0, $2, mult1	C2FE	1100001011111110	1100 - bne	00 - $0 10 - $2	X	11111110 = 0xFE
clr $0	D000	1101000000000000	1101 - clr	00 - $0	00 - $0	padding
clr $3	D3C0	1101001111000000	1101 - clr	11 - $3	11 - $3	padding
addi $1, $0, 0xFA	31FA	0011000111111010	0011 - addi	00 - $0	01 - $1	11111010 = 0xFA
addi $2, $0, 0x03	3203	0011001000000011	0011 - addi	00 - $0	10 - $2	00000011 = 0x03
mult2:						
add $3, $3, $1	2DC0	0010110111000000	0010 - add	11 - $3 01 - $1	11 - $3	000000 - padding
addi $0, $0, 0x01	3001	0011000000000001	0011 - addi	00 - $0	00 - $0	00000001 = 0x01
bne $0, $2, mult2	C2FE	1100001011111110	1100 - bne	00 - $0 10 - $2	X	11111110 = 0xFE

clr $0	D000	1101000000000000	1101 - clr	00 - $0	00 - $0	padding
clr $3	D3C0	1101001111000000	1101 - clr	11 - $3	11 - $3	padding
addi $1, $0, 0xF7	31F7	0011000111110111	0011 - addi	00 - $0	01 - $1	11110111 = 0xF7
addi $2, $0, 0x03	3203	0011001000000011	0011 - addi	00 - $0	10 - $2	00000011 = 0x03
mult3:						
add $3, $3, $1	2DC0	0010110111000000	0010 - add	11 - $3 01 - $1	11 - $3	000000 - padding
addi $0, $0, 0x01	3001	0011000000000001	0011 - addi	00 - $0	00 - $0	00000001 = 0x1
bne $0, $2, mult3	C2FE	1100001011111110	1100 - bne	00 - $0 10 - $2	X	11111110 = 0xFE
inv $3, $3	43C0	0100001111000000	0100 - inv	11 - $3	11 - $3	padding
addi $3, $3, 0x01	3F01	0011111100000001	0011 - addi	11 - $3	11 - $3	00000001 = 0x01

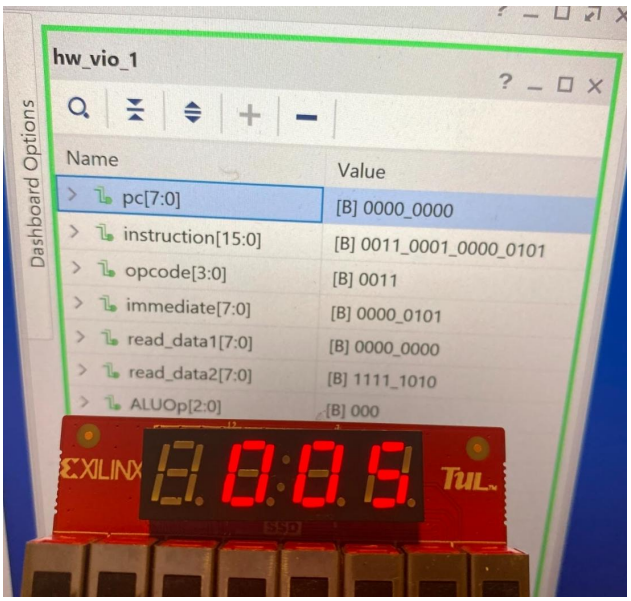
Testing in Hardware

For the hardware implementation of this lab, we were given a constraint file, *pdatapath.xdc*, that we had to add prior to using the circuit. With that added, we generated the bitstream and programmed the board as usual. We used the VIO interface and seven segment display to see values and verify the instructions. To test the design, we pressed BTN1 to increment the PC, which is now capable of branching, and confirmed that all of the inputs and outputs were correct.

VIO and SSD Sample Output: (The full series of screenshots are after the conclusion)

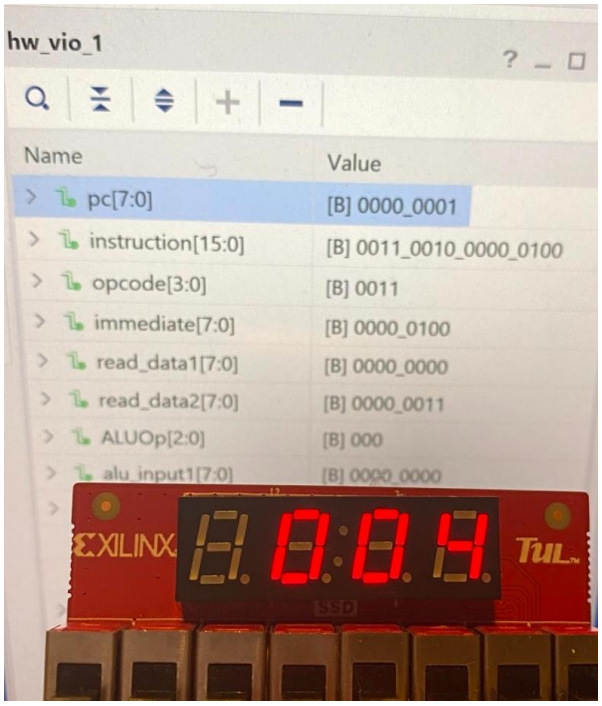


addi $1, $0, 0x05



pc[7:0]	[B] 0000_0001
instruction[15:0]	[B] 0011_0010_0000_0100
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0100
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0100
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0100
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0100
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 00
MemToReg	[B] 0

addi $2, $0, 0x04



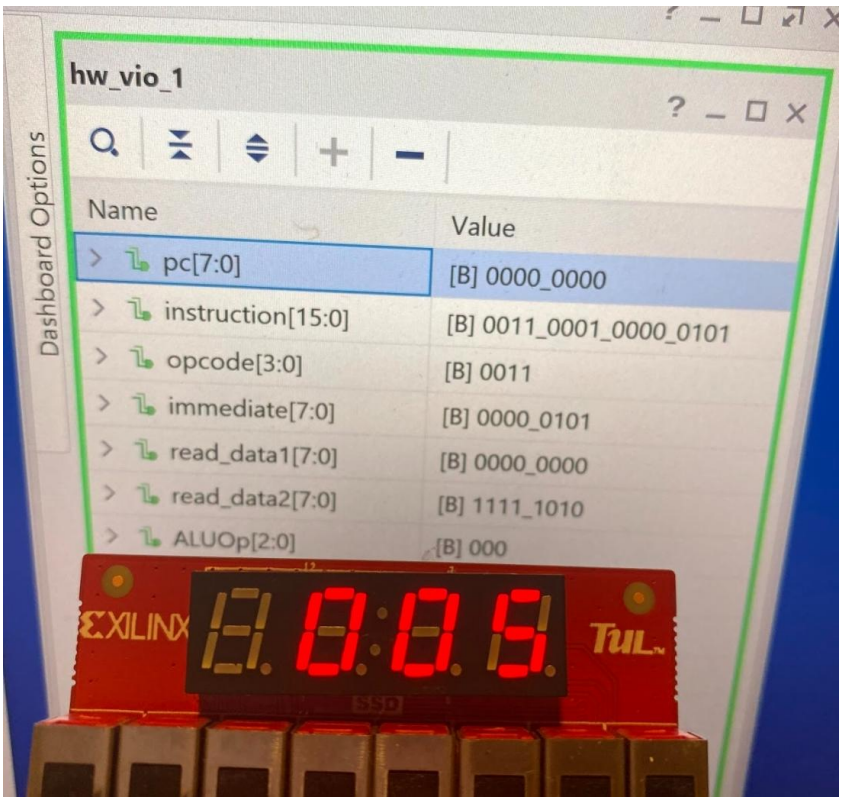
Conclusion:

The major takeaways from this lab was experience incorporating branching into the Program Counter module alongside designing a series of assembly instructions that utilizes branching in order to perform signed multiplication. We also got more practice recognizing the different fields in machine code instructions since we had to identify the different instruction types and verify that they were carried out correctly. Finally, with our circuit being capable of branching, we now have a working single-cycle MIPS processor.

Complete VIO and SSD Outputs:

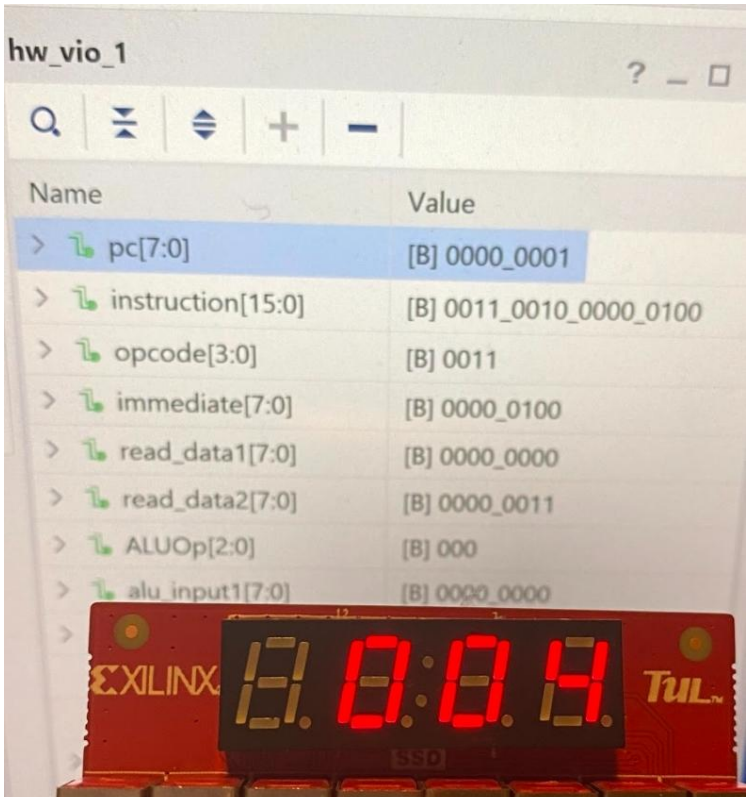
pc[7:0]	[B] 0000_0000
instruction[15:0]	[B] 0011_0001_0000_0101
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0101
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 1111_1010
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0101
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0101
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0101
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 00
MemToReg	[B] 0

addi $1, $0, 0x5



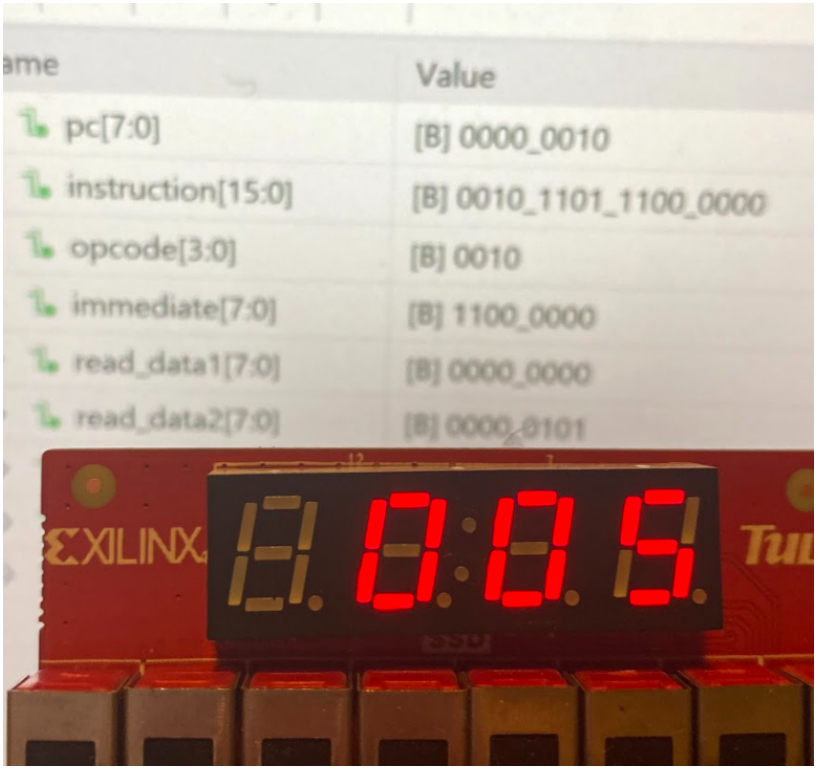
pc[7:0]	[B] 0000_0001
instruction[15:0]	[B] 0011_0010_0000_0100
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0100
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0100
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0100
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0100
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 00
MemToReg	[B] 0

addi $2, $0, 0x4



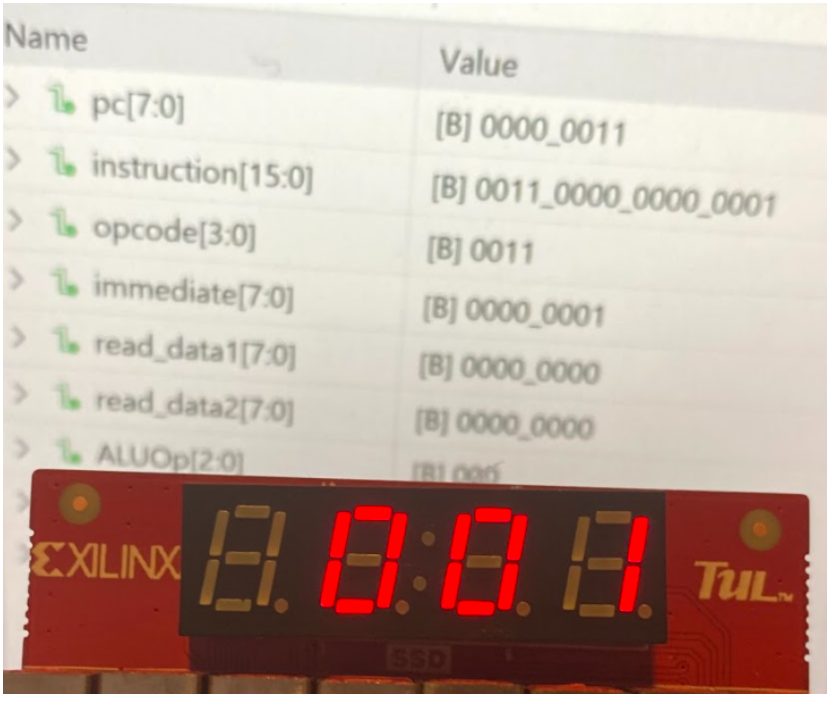
pc[7:0]	[B] 0000_0010
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0000_0101
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0101
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0101
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_0000_0101
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0

mult1:
add $3, $3, $1



addi $0, $0, 0x1

pc[7:0]	[B] 0000_0011
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0000_0000
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0001
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0001
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

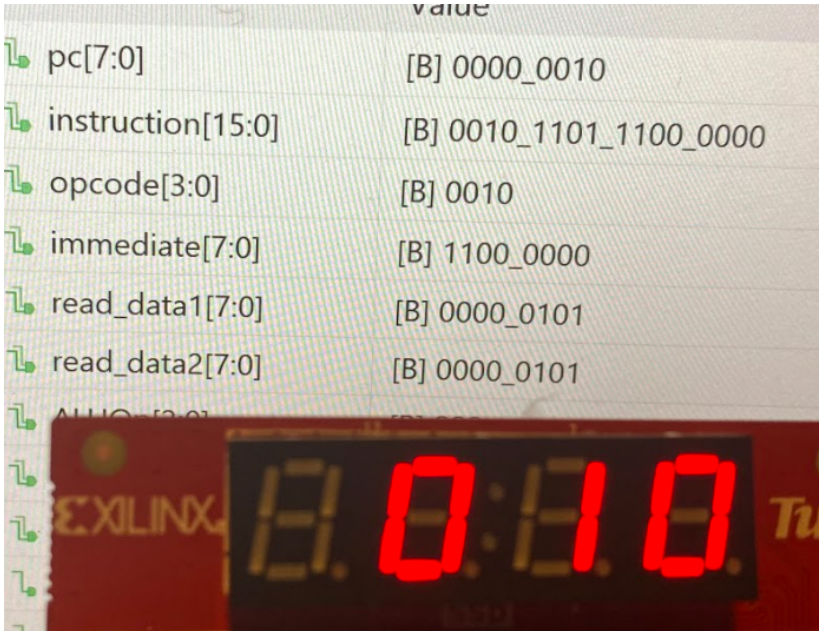


bne $0, $2, mult1

pc[7:0]	[B] 0000_0100
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0001
read_data2[7:0]	[B] 0000_0100
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0001
alu_input2[7:0]	[B] 0000_0100
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 1
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

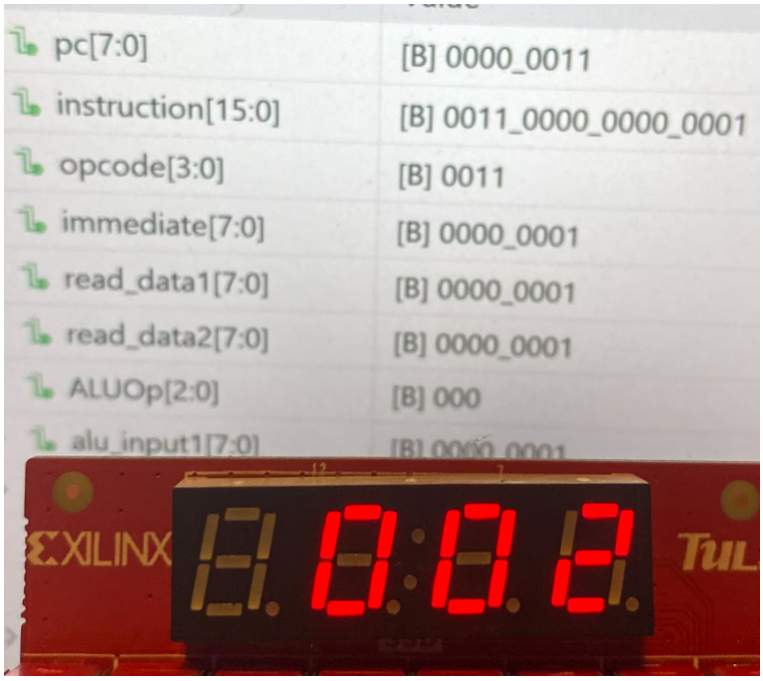
add $3, $3, $1

pc[7:0]	[B] 0000_0010
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 0000_0101
read_data2[7:0]	[B] 0000_0101
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0101
alu_input2[7:0]	[B] 0000_0101
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_1010
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_0000_1010
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0



addi $0, $0, 0x1

pc[7:0]	[B] 0000_0011
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0001
read_data2[7:0]	[B] 0000_0001
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0001
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0010
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0010
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

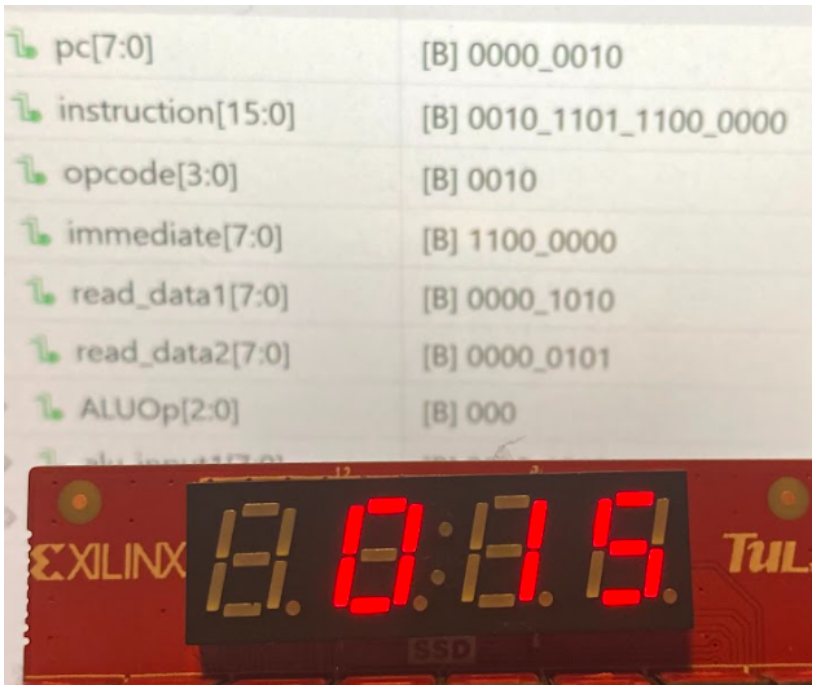


bne $0, $2, mult1

pc[7:0]	[B] 0000_0100
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0010
read_data2[7:0]	[B] 0000_0100
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0010
alu_input2[7:0]	[B] 0000_0100
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 1
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

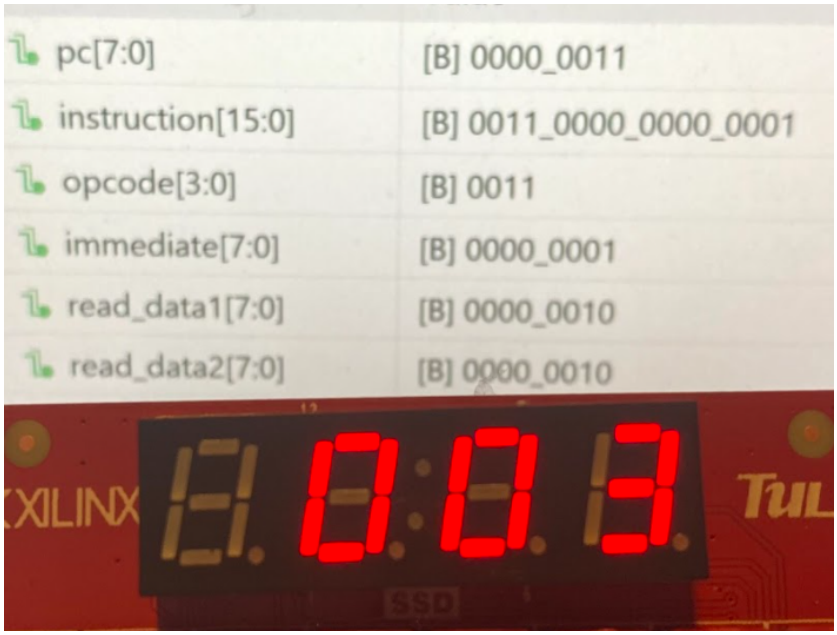
add $3, $3, $1

pc[7:0]	[B] 0000_0010
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 0000_1010
read_data2[7:0]	[B] 0000_0101
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_1010
alu_input2[7:0]	[B] 0000_0101
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_1111
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_0000_1111
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0



addi $0, $0, 0x1

pc[7:0]	[B] 0000_0011
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0010
read_data2[7:0]	[B] 0000_0010
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0010
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0011
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0011
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

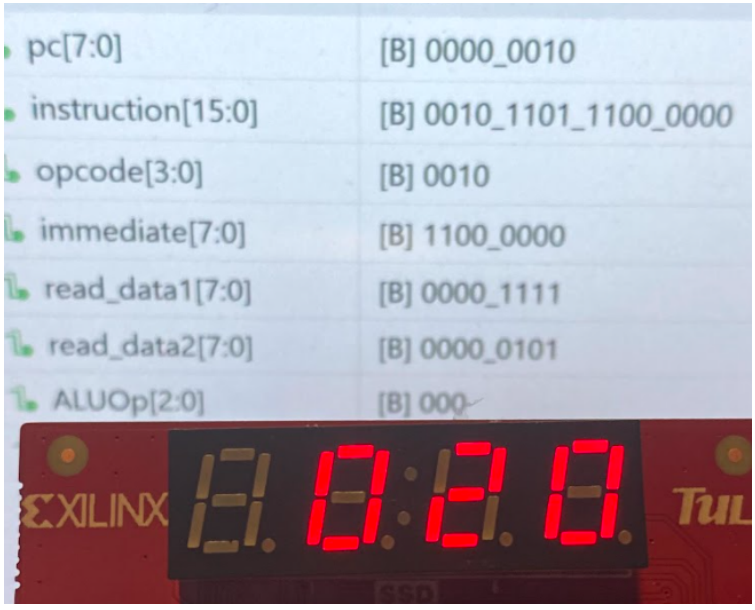


bne $0, $2, mult1

pc[7:0]	[B] 0000_0100
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0011
read_data2[7:0]	[B] 0000_0100
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0011
alu_input2[7:0]	[B] 0000_0100
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 1
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

add $3, $3, $1

pc[7:0]	[B] 0000_0010
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 0000_1111
read_data2[7:0]	[B] 0000_0101
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_1111
alu_input2[7:0]	[B] 0000_0101
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0001_0100
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_0001_0100
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0



pc[7:0]	[B] 0000_0011
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0011
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0011
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0100
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0100
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

addi $0, $0, 0x1

I forgot to take the picture for this instruction, but the LED displayed a value of 4 as verified by the alu_output probe having the value of 8'b00000100, which has a 2's complement value of 4

pc[7:0]	[B] 0000_0100
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0100
read_data2[7:0]	[B] 0000_0100
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0100
alu_input2[7:0]	[B] 0000_0100
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

bne $0, $2, mult1

The take_branch probe is 0, which means that the values of register 0 and 2 are equal and the computation is complete

clr $0

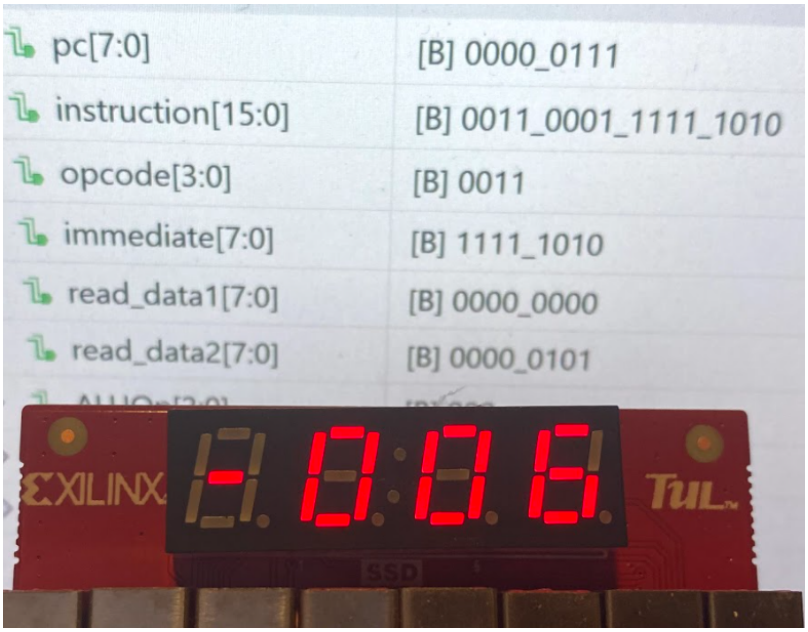
pc[7:0]	[B] 0000_0101
instruction[15:0]	[B] 1101_0000_0000_0000
opcode[3:0]	[B] 1101
immediate[7:0]	[B] 0000_0000
read_data1[7:0]	[B] 0000_0100
read_data2[7:0]	[B] 0000_0100
ALUOp[2:0]	[B] 010
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0100
ALUSrc1	[B] 1
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 00
MemToReg	[B] 0

clr $3

pc[7:0]	[B] 0000_0110
instruction[15:0]	[B] 1101_0011_1100_0000
opcode[3:0]	[B] 1101
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0001_0100
ALUOp[2:0]	[B] 010
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0001_0100
ALUSrc1	[B] 1
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 11
rd_addr[1:0]	[B] 11
MemToReg	[B] 0

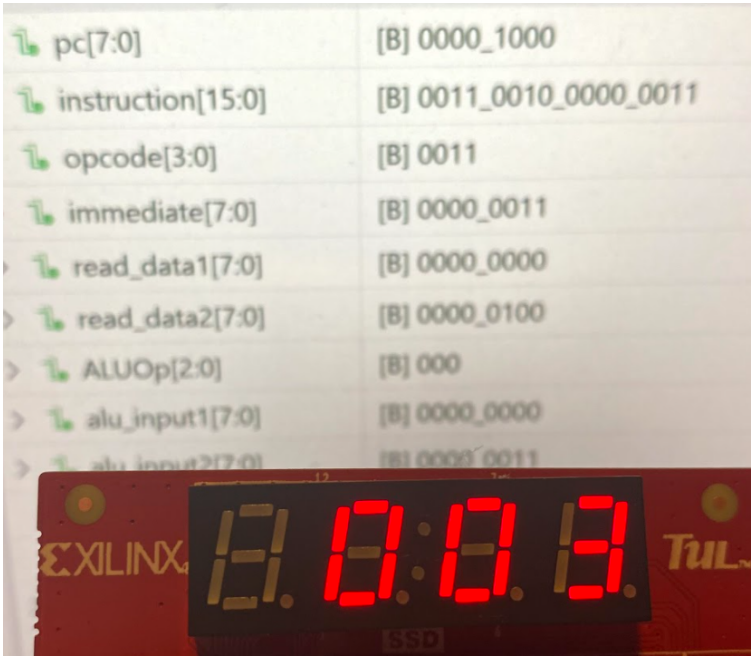
addi $1, $0, 0xFA

pc[7:0]	[B] 0000_0111
instruction[15:0]	[B] 0011_0001_1111_1010
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 1111_1010
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0000_0101
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 1111_1010
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 1111_1010
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_1111_1010
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0



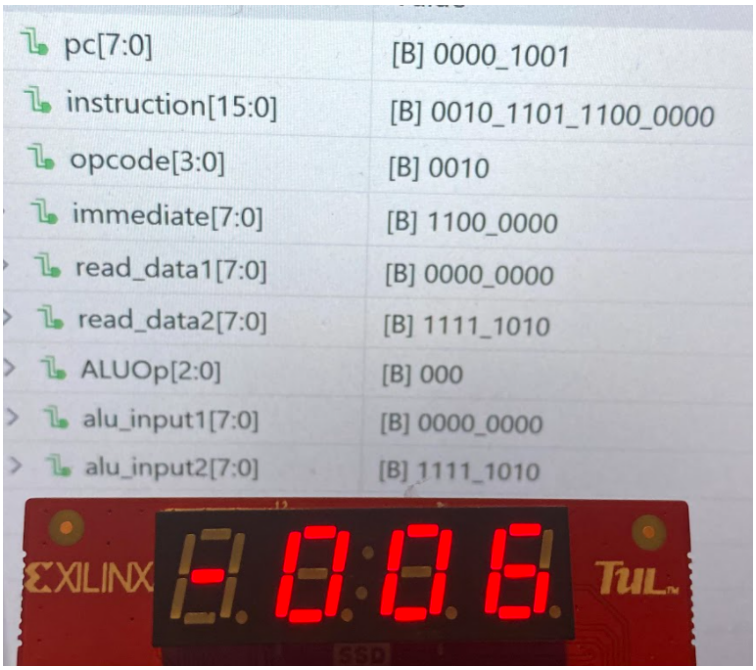
addi $2, $0, 0x3

pc[7:0]	[B] 0000_1000
instruction[15:0]	[B] 0011_0010_0000_0011
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0011
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0000_0100
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0011
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0011
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0011
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 11
MemToReg	[B] 0



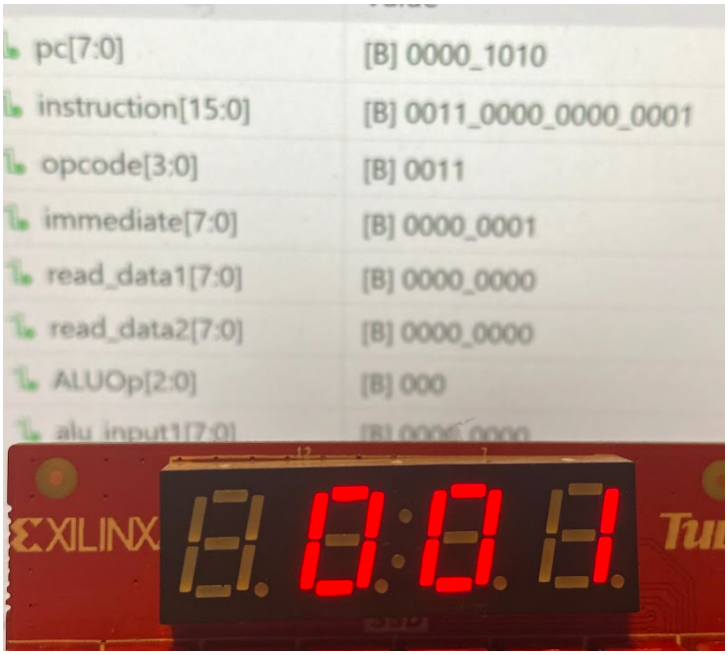
pc[7:0]	[B] 0000_1001
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 1111_1010
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 1111_1010
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 1111_1010
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_1111_1010
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0

mult2:
add $3, $3, $1



pc[7:0]	[B] 0000_1010
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0000_0000
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0001
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0001
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

addi $0, $0, 0x1

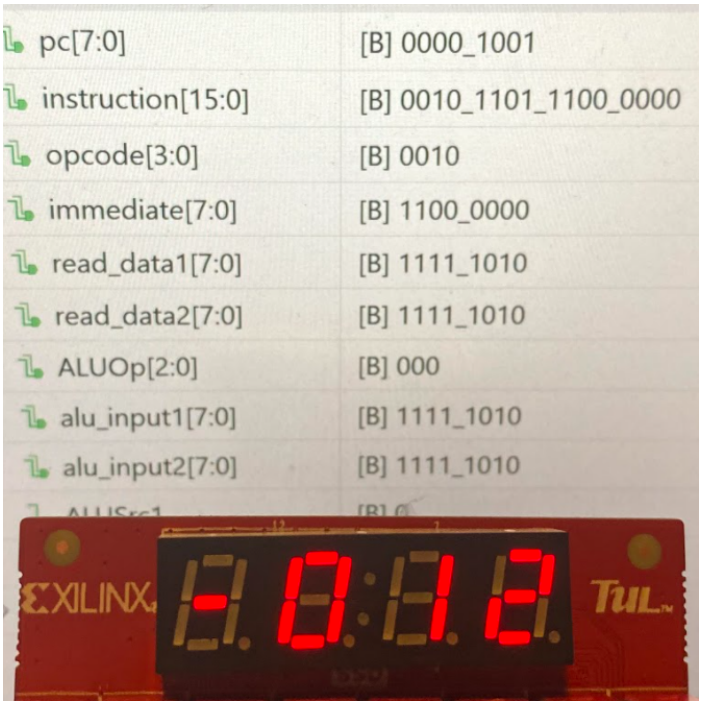


bne $0, $2, mult2

pc[7:0]	[B] 0000_1011
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0001
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0001
alu_input2[7:0]	[B] 0000_0011
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 1
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

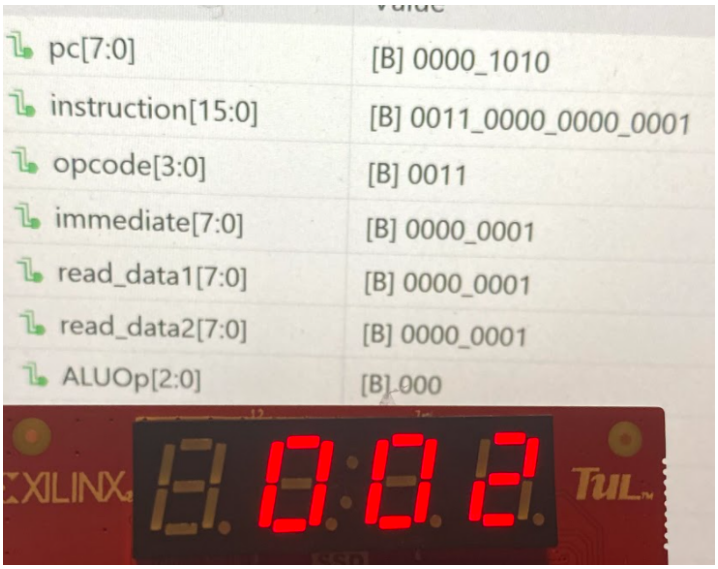
add $3, $3, $1

pc[7:0]	[B] 0000_1001
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 1111_1010
read_data2[7:0]	[B] 1111_1010
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 1111_1010
alu_input2[7:0]	[B] 1111_1010
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 1111_0100
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_1111_0100
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0



addi $0, $0, 0x1

pc[7:0]	[B] 0000_1010
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0001
read_data2[7:0]	[B] 0000_0001
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0001
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0010
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0010
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

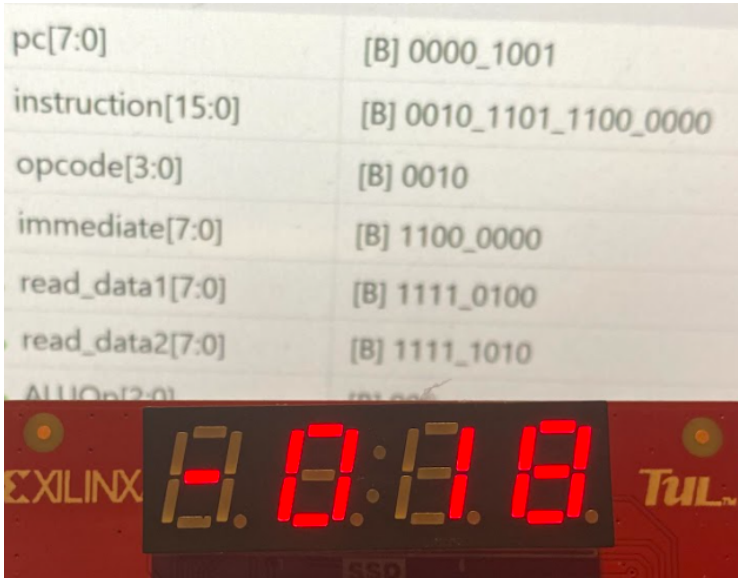


bne $0, $2, mult2

pc[7:0]	[B] 0000_1011
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0010
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0010
alu_input2[7:0]	[B] 0000_0011
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 1
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

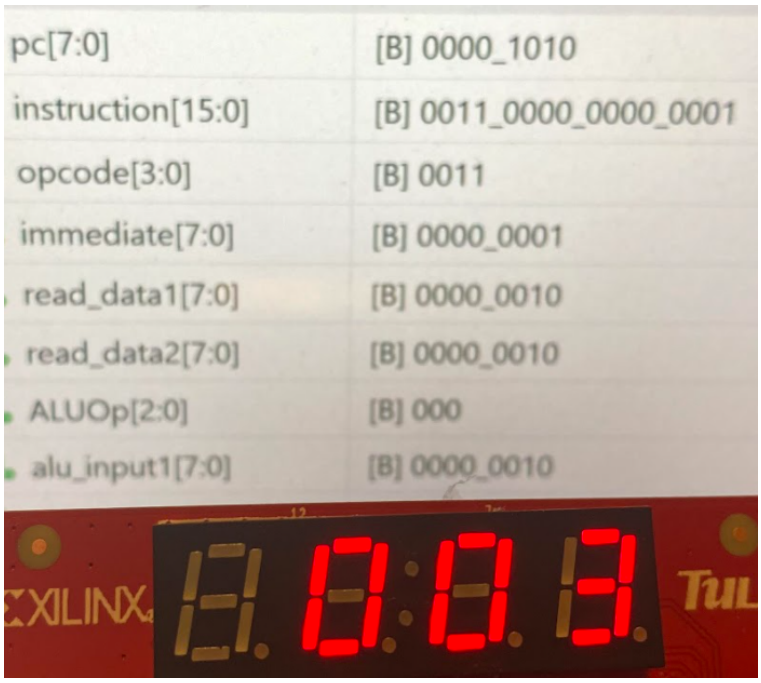
add $3, $3, $1

pc[7:0]	[B] 0000_1001
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 1111_0100
read_data2[7:0]	[B] 1111_1010
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 1111_0100
alu_input2[7:0]	[B] 1111_1010
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 1110_1110
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_1110_1110
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0



addi $0, $0, 0x1

pc[7:0]	[B] 0000_1010
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0010
read_data2[7:0]	[B] 0000_0010
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0010
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0011
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0011
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0



bne $0, $2, mult2

pc[7:0]	[B] 0000_1011
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0011
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0011
alu_input2[7:0]	[B] 0000_0011
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

clr $0

pc[7:0]	[B] 0000_1100
instruction[15:0]	[B] 1101_0000_0000_0000
opcode[3:0]	[B] 1101
immediate[7:0]	[B] 0000_0000
read_data1[7:0]	[B] 0000_0011
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 010
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0011
ALUSrc1	[B] 1
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 00
MemToReg	[B] 0

clr $3

pc[7:0]	[B] 0000_1101
instruction[15:0]	[B] 1101_0011_1100_0000
opcode[3:0]	[B] 1101
immediate[7:0]	[B] 0000_0000
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 1110_1110
ALUOp[2:0]	[B] 010
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 1110_1110
ALUSrc1	[B] 1
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 11
rd_addr[1:0]	[B] 11
MemToReg	[B] 0

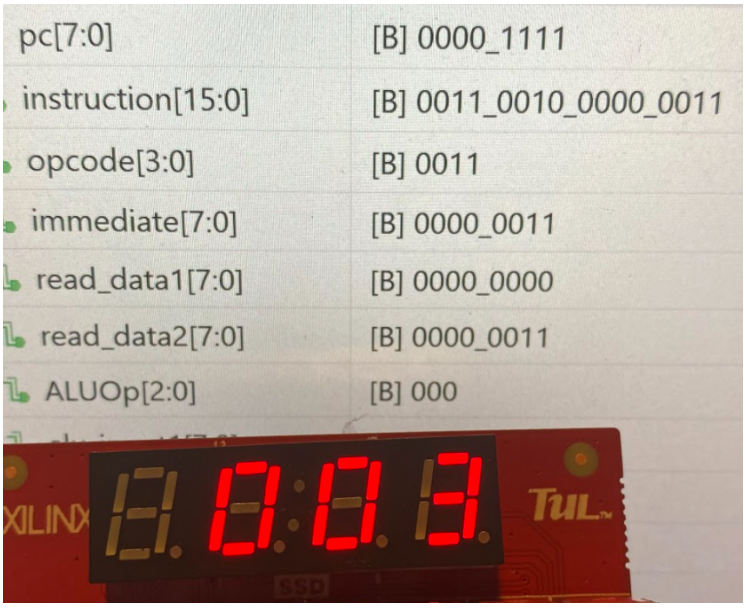
addi $1, $0, 0xF7

I forgot to take the picture for this instruction, but the LED displayed a value of -9 as verified by the alu_output probe having the value of 8'b11110111, which has a 2's complement value of -9

pc[7:0]	[B] 0000_1110
instruction[15:0]	[B] 0011_0001_1111_0111
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 1111_0111
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 1111_1010
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 1111_0111
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 1111_0111
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_1111_0111
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0

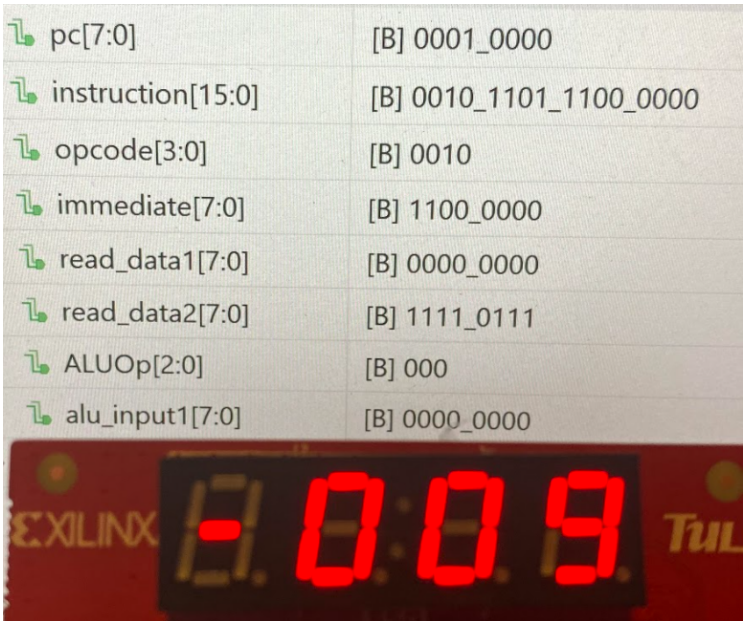
pc[7:0]	[B] 0000_1111
instruction[15:0]	[B] 0011_0010_0000_0011
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0011
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0011
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0011
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0011
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 11
MemToReg	[B] 0

addi $2, $0, 0x3



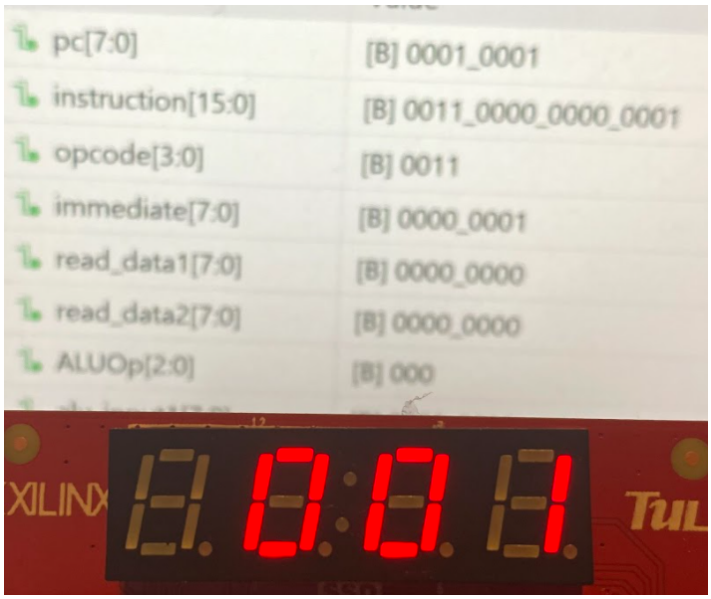
pc[7:0]	[B] 0001_0000
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 1111_0111
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 1111_0111
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 1111_0111
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_1111_0111
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0

mult3:
add $3, $3, $1



addi $0, $0, 0x1

pc[7:0]	[B] 0001_0001
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0000
read_data2[7:0]	[B] 0000_0000
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0000
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0001
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0001
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

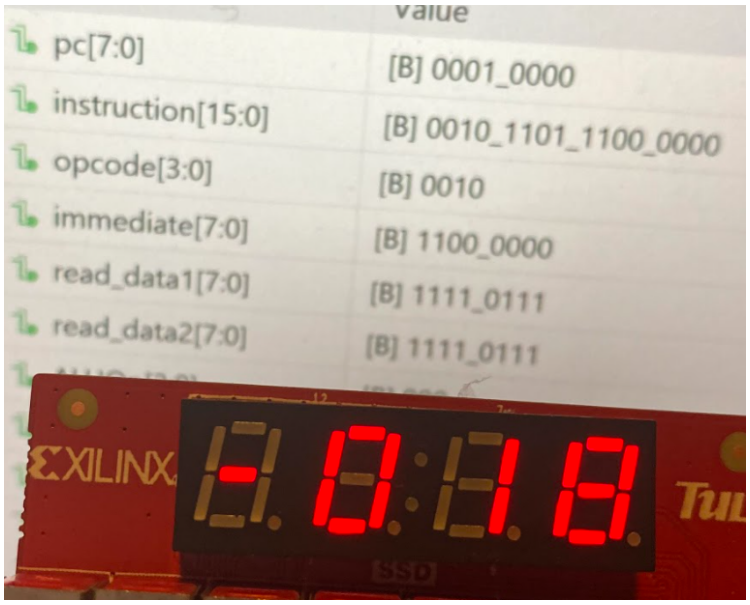


bne $0, $2, mult3

pc[7:0]	[B] 0001_0010
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0001
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0001
alu_input2[7:0]	[B] 0000_0011
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 1
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 00
MemToReg	[B] 0

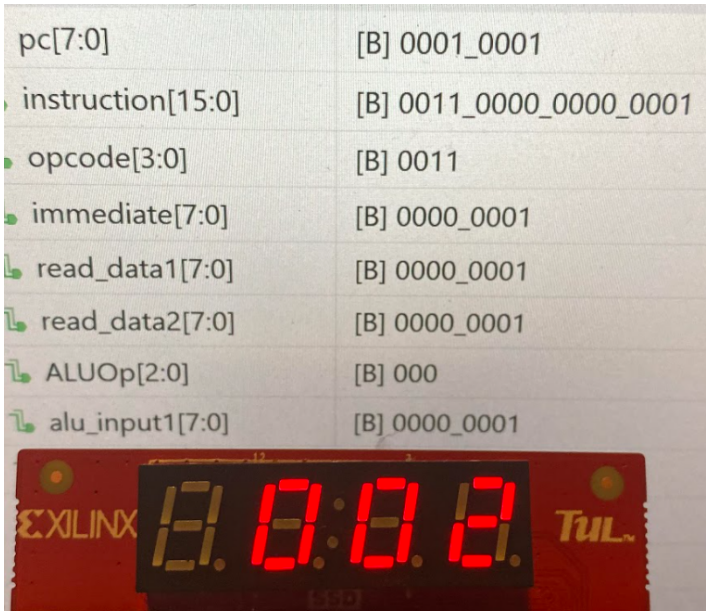
add $3, $3, $1

pc[7:0]	[B] 0001_0000
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 1111_0111
read_data2[7:0]	[B] 1111_0111
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 1111_0111
alu_input2[7:0]	[B] 1111_0111
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 1110_1110
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_1110_1110
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0



addi $0, $0, 0x1

pc[7:0]	[B] 0001_0001
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0001
read_data2[7:0]	[B] 0000_0001
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0001
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0010
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0010
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

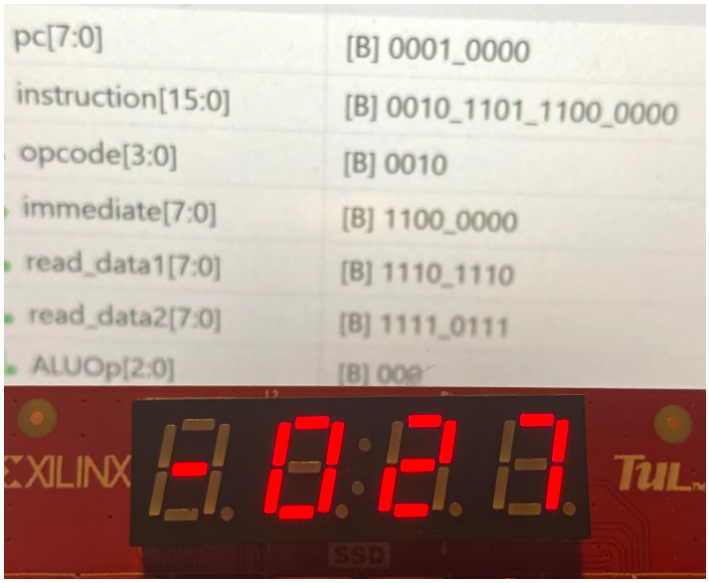


pc[7:0]	[B] 0001_0010
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0010
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0010
alu_input2[7:0]	[B] 0000_0011
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 1
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 00
MemToReg	[B] 0

bne $0, $2, mult3

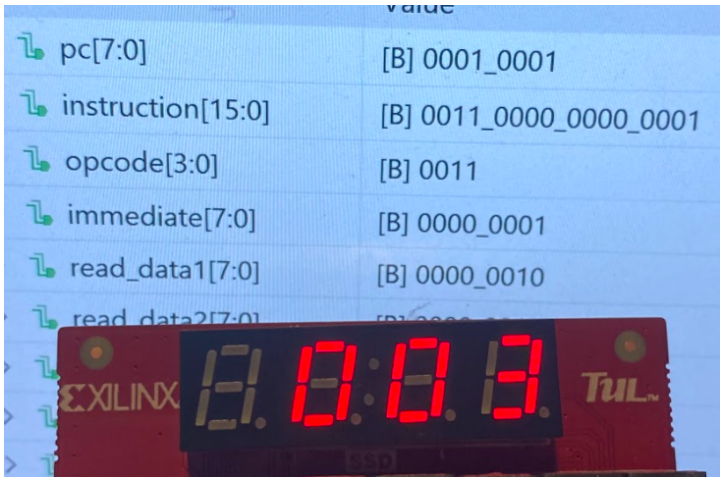
pc[7:0]	[B] 0001_0000
instruction[15:0]	[B] 0010_1101_1100_0000
opcode[3:0]	[B] 0010
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 1110_1110
read_data2[7:0]	[B] 1111_0111
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 1110_1110
alu_input2[7:0]	[B] 1111_0111
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 1110_0101
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_1110_0101
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 01
rd_addr[1:0]	[B] 11
MemToReg	[B] 0

add $3, $3, $1



addi $0, $0, 0x1

pc[7:0]	[B] 0001_0001
instruction[15:0]	[B] 0011_0000_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0000_0010
read_data2[7:0]	[B] 0000_0010
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0000_0010
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0000_0011
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0011
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 00
rd_addr[1:0]	[B] 10
MemToReg	[B] 0

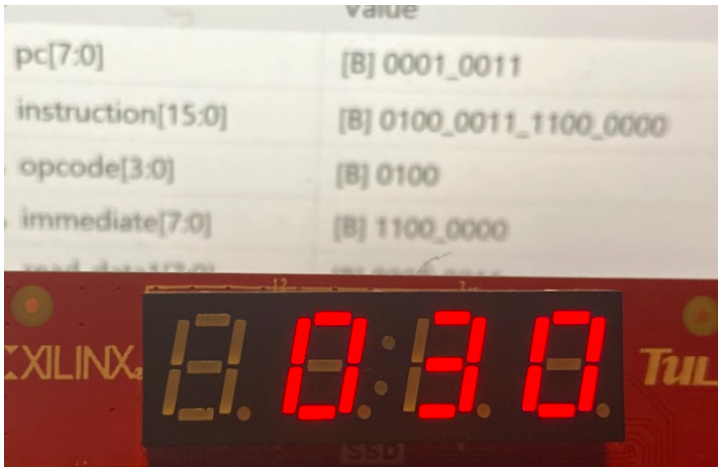


bne $0, $2, mult3

pc[7:0]	[B] 0001_0010
instruction[15:0]	[B] 1100_0010_1111_1110
opcode[3:0]	[B] 1100
immediate[7:0]	[B] 1111_1110
read_data1[7:0]	[B] 0000_0011
read_data2[7:0]	[B] 0000_0011
ALUOp[2:0]	[B] 111
alu_input1[7:0]	[B] 0000_0011
alu_input2[7:0]	[B] 0000_0011
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0000_0000
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 0
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0000_0000
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 10
rd_addr[1:0]	[B] 00
MemToReg	[B] 0

inv $3, $3

pc[7:0]	[B] 0001_0011
instruction[15:0]	[B] 0100_0011_1100_0000
opcode[3:0]	[B] 0100
immediate[7:0]	[B] 1100_0000
read_data1[7:0]	[B] 0000_0011
read_data2[7:0]	[B] 1110_0101
ALUOp[2:0]	[B] 001
alu_input1[7:0]	[B] 0000_0011
alu_input2[7:0]	[B] 1110_0101
ALUSrc1	[B] 0
ALUSrc2	[B] 0
alu_output[7:0]	[B] 0001_1010
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 1
regfile_write_data[8:0]	[B] 0_0001_1010
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 00
rt_addr[1:0]	[B] 11
rd_addr[1:0]	[B] 11
MemToReg	[B] 0



addi $3, $3, 0x1

pc[7:0]	[B] 0001_0100
instruction[15:0]	[B] 0011_1111_0000_0001
opcode[3:0]	[B] 0011
immediate[7:0]	[B] 0000_0001
read_data1[7:0]	[B] 0001_1010
read_data2[7:0]	[B] 0001_1010
ALUOp[2:0]	[B] 000
alu_input1[7:0]	[B] 0001_1010
alu_input2[7:0]	[B] 0000_0001
ALUSrc1	[B] 0
ALUSrc2	[B] 1
alu_output[7:0]	[B] 0001_1011
alu_ovf	[B] 0
take_branch	[B] 0
RegWrite	[B] 1
RegDst	[B] 0
regfile_write_data[8:0]	[B] 0_0001_1011
MemWrite	[B] 0
data_mem_out[8:0]	[B] 0_0000_0000
rs_addr[1:0]	[B] 11
rt_addr[1:0]	[B] 11
rd_addr[1:0]	[B] 10
MemToReg	[B] 0