Homework 4

P1 (a) add x_1, x_2, x_3

0000000 00011 00010 000 00001 0110011

(b) addi x_1, x_2, 100

000001100100 00010 000 00001 0010011

(c) lb x1, 4(x2)

000000000100 00010 000 00001 0000011

(d) beq x6, x8, 1024

0 000000 01000 00110 000 00001 1100011

P2 a) x1=100, x2=100, x3=200

b) x0 = 0, x1 = 100

c) 0xdead = -8531; 0xbeef = -16657; 0x1024 = 4132

x1 = 0xdead, x2 = 0xbeef, x4 = 0xdead.

d) x1 = -1, x2 = 1, x3 = 100

e) x_1=0 → x2=8, PC=20 → x3=24, PC = 8 → x1 = 100 → x1=300 → nop

x1=300, x2 = 8, x3 = 24.

P3 a) nop: addi x0, x0, 0

b) mv rd, rs: addi rd, rs, 0

c) li rd, imm: lui rd, imm[31:12]

addi rd, rd, imm[]

slli rd, rd, 5'd12

addi rd, rd, imm[11:0]

d) beqz rs, imm: beq rs, x0, imm

e) j imm: jal x0, imm

f) bgt rs1, rs2, imm: blt rs2, rs1, imm

P4 a)

WBSel JAL ↑ JALR ↑

pc+4 10: Jal(1101111) Jalr(1100111)
alu 01: add(0110011) addi(0010011) auipc(0010111)
mem 00: lw(0000011)

OP-FP OP-IMM AUIPC LOAD

```
sig[1:0]
assign sig[1] = (opcode == JAL) || (opcode == JALR)
assign sig[0] = (opcode == OP-FP) || (opcode == OP-IMM) || (opcode == AUIPC)
```

b) MemRW, 0: read, 1: write

0: add(0110011) addi(0010011) lw(0000011)
1: sw(0100011)

```
assign sig = (opcode == STORE)
```

c) PCSel

PC+4 0: add(0110011); addi(0010011); lw(0000011); sw(0100011)
alu 1: beq(1100011); jal(1101111); jalr(1100111)

```
assign sig = (opcode == BRANCH) || (opcode == JALR) || (opcode == JAL)
```

d) BSel

rs2 0: add(0110011)
imm 1: addi(0010011); lw(0000011); sw(0100011); beq(1100011); ...

```
assign sig = (opcode != OP-FP)
```

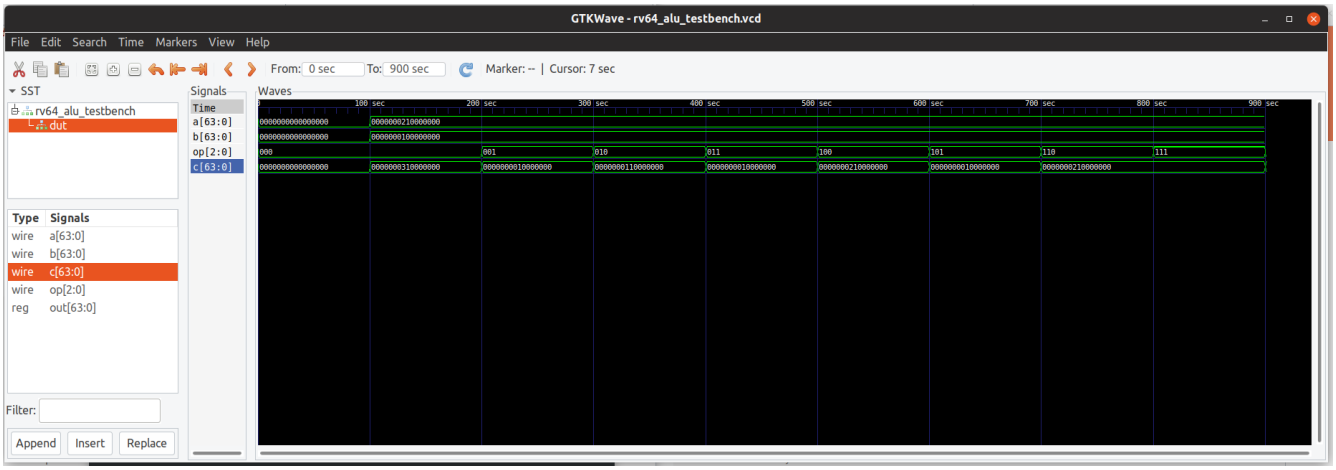
```
module load_decoder(
    input [31:0] addr,
    input [31:0] raw_data,
    input lb, lbu, lh, lhu, lw,
    output [31:0] wb_data
);
    reg [31:0] data;
    assign wb_data = data;
    always @(*) begin
        case ({lb, lbu, lh, lhu, lw})
            5'b10000: data = $signed(raw_data[7:0]);
            5'b01000: data = $unsigned(raw_data[7:0]);
            5'b00100: data = $signed(raw_data[15:0]);
            5'b00010: data = $unsigned(raw_data[15:0]);
            5'b00001: data = raw_data;
            default: data = 32'b0;
        endcase
    end

endmodule
```

```
module load_decoder_testbench();
    reg [31:0] addr;
    reg [31:0] raw_data;
    reg lb, lbu, lh, lhu, lw;
    wire [31:0] wb_data;
    initial raw_data = 32'b10000000000000001000000010000000;
    initial lb = 1'b0;
    initial lbu = 1'b0;
    initial lh = 1'b0;
    initial lhu = 1'b0;
    initial lw = 1'b0;

    load_decoder dut(
        .addr(addr),
        .raw_data(raw_data),
        .lb(lb),
        .lbu(lbu),
        .lh(lh),
        .lhu(lhu),
        .lw(lw)
    );

    initial begin
        $dumpfile("load_decoder_testbench.vcd");
        $dumpvars(0, load_decoder_testbench);

        #100;
        lb = 1'b1;
        #100;
        lb = 1'b0;
        lbu = 1'b1;
        #100;
        lbu = 1'b0;
        lh = 1'b1;
        #100;
        lh = 1'b0;
        lhu = 1'b1;
        #100;
        lhu = 1'b0;
        lw = 1'b1;
        #100;
        lw = 1'b0;
        $finish();
    end
endmodule
```

```
`define ALU_ADD 0
`define ALU_ADDW 1
`define ALU_SUB 2
`define ALU_SUBW 3
`define ALU_SLL 4
`define ALU_SLLW 5
`define ALU_SRA 6
`define ALU_SRAW 7
module rv64_alu(
    input [63:0] a,
    input [63:0] b,
    input [2:0] op, // op can be any of values`define'd above
    output [63:0] c
);

    reg [63:0] out;
    assign c = out;
    always @(*) begin
        case (op)
            3'd0: out = $signed(a) + $signed(b);
            3'd1: out = $signed(a[31:0]) + $signed(b[31:0]);
            3'd2: out = $signed(a) - $signed(b);
            3'd3: out = $signed(a[31:0]) - $signed(b[31:0]);
            3'd4: out = $unsigned(a) << b;
            3'd5: out = $signed($unsigned(a[31:0]) << b);
            3'd6: out = $signed(a) >> b;
            3'd7: out = $signed(a) >> b;
            default: out = 64'b0;
        endcase
    end

endmodule
```

```
module rv64_alu_testbench();
    reg [63:0] a;
    reg [63:0] b;
    reg [2:0] op;
    wire [63:0] c;
    initial a = 64'b0;
    initial b = 64'b0;
    initial op = 3'b0;

    rv64_alu dut(
        .a(a),
        .b(b),
        .op(op),
        .c(c)
    );

    initial begin
        $dumpfile("rv64_alu_testbench.vcd");
        $dumpvars(0, rv64_alu_testbench);

        #100;
        a = 64'h210000000;
        b = 64'h100000000;
        op = 3'd0;
        #100;
        op = 3'd1;
        #100;
        op = 3'd2;
        #100;
        op = 3'd3;
        #100;
        op = 3'd4;
        #100;
        op = 3'd5;
        #100;
        op = 3'd6;
        #100;
        op = 3'd7;
        #100;
        op = 3'd0;
        $finish();
    end

endmodule
```

Above: problem 5 waveform.



Above: problem 6 waveform.