16-Bit RISC Processor

1. Verilog code for Instruction Memory:

```
`include "Parameter.v"
module Instruction_Memory(
 input[15:0] pc,
 output[15:0] instruction
);

 reg [`col - 1:0] memory [`row_i - 1:0];
 wire [3 : 0] rom_addr = pc[4 : 1];
 initial
 begin
  $readmemb("./test/test.prog", memory,0,14);
 end
 assign instruction =  memory[rom_addr];

endmodule
```

2.Verilog code for Register File:

```
`timescale 1ns / 1ps
module GPRs(
 input    clk,
 input    reg_write_en,
 input  [2:0] reg_write_dest,
 input  [15:0] reg_write_data,
 input  [2:0] reg_read_addr_1,
 output  [15:0] reg_read_data_1,
 input  [2:0] reg_read_addr_2,
 output  [15:0] reg_read_data_2
);
 reg [15:0] reg_array [7:0];
 integer i;
 //reg [2:0] i;
 initial begin
  for(i=0;i<8;i=i+1)
   reg_array[i] <= 16'd0;
 end
 always @ (posedge clk ) begin
   if(reg_write_en) begin
    reg_array[reg_write_dest] <= reg_write_data;
   end
 end

 assign reg_read_data_1 = reg_array[reg_read_addr_1];
 assign reg_read_data_2 = reg_array[reg_read_addr_2];
endmodule
```

3.Verilog code for Data Memory:

```
`include "Parameter.v"

module Data_Memory(
 input clk,
 input [15:0]   mem_access_addr,


 input [15:0]   mem_write_data,
 input     mem_write_en,
 input mem_read,
 output [15:0]   mem_read_data
);

reg [`col - 1:0] memory [`row_d - 1:0];
integer f;
wire [2:0] ram_addr=mem_access_addr[2:0];
initial
 begin
  $readmemb("./test/test.data", memory);

  f = $fopen(`filename);
  $fmonitor(f, "time = %d\n", $time,
  "\tmemory[0] = %b\n", memory[0],
  "\tmemory[1] = %b\n", memory[1],
  "\tmemory[2] = %b\n", memory[2],
  "\tmemory[3] = %b\n", memory[3],
  "\tmemory[4] = %b\n", memory[4],
  "\tmemory[5] = %b\n", memory[5],
  "\tmemory[6] = %b\n", memory[6],
  "\tmemory[7] = %b\n", memory[7]);
  `simulation_time;
  $fclose(f);
 end

 always @(posedge clk) begin
  if (mem_write_en)
   memory[ram_addr] <= mem_write_data;
 end
 assign mem_read_data = (mem_read==1'b1) ? memory[ram_addr]: 16'd0;


endmodule
```

4.Verilog code for ALU unit:

```
module ALU(

 input  [15:0] a,  //src1
 input  [15:0] b,  //src2
 input  [2:0] alu_control, //function sel

 output reg [15:0] result,  //result
 output zero
    );

always @(*)
begin
 case(alu_control)
 3'b000: result = a + b; // add
 3'b001: result = a - b; // sub
 3'b010: result = ~a;
 3'b011: result = a<<b;
 3'b100: result = a>>b;
 3'b101: result = a & b; // and
 3'b110: result = a | b; // or
 3'b111: begin if (a<b) result = 16'd1;
    else result = 16'd0;
    end
 default:result = a + b; // add
 endcase
end
assign zero = (result==16'd0) ? 1'b1: 1'b0;

endmodule
```

5. Verilog code for ALU Control Unit of the RISC processor:

```
`timescale 1ns / 1ps
module alu_control( ALU_Cnt, ALUOp, Opcode);
 output reg[2:0] ALU_Cnt;
 input [1:0] ALUOp;
 input [3:0] Opcode;
 wire [5:0] ALUControlIn;
 assign ALUControlIn = {ALUOp,Opcode};
 always @(ALUControlIn)
 casex (ALUControlIn)
   6'b10xxxx: ALU_Cnt=3'b000;
   6'b01xxxx: ALU_Cnt=3'b001;
   6'b000010: ALU_Cnt=3'b000;
   6'b000011: ALU_Cnt=3'b001;
   6'b000100: ALU_Cnt=3'b010;
   6'b000101: ALU_Cnt=3'b011;
   6'b000110: ALU_Cnt=3'b100;
   6'b000111: ALU_Cnt=3'b101;
   6'b001000: ALU_Cnt=3'b110;
   6'b001001: ALU_Cnt=3'b111;
  default: ALU_Cnt=3'b000;
  endcase
endmodule
```

6. Verilog code for Datapath of the RISC processor:

```
`timescale 1ns / 1ps
module Datapath_Unit(
 input clk,
 input jump,beq,mem_read,mem_write,alu_src,reg_dst,mem_to_reg,reg_write,bne,
 input[1:0] alu_op,
 output[3:0] opcode
);
 reg  [15:0] pc_current;
 wire [15:0] pc_next,pc2;
```

```verilog
 wire [15:0] instr;
 wire [2:0] reg_write_dest;
 wire [15:0] reg_write_data;
 wire [2:0] reg_read_addr_1;
 wire [15:0] reg_read_data_1;
 wire [2:0] reg_read_addr_2;
 wire [15:0] reg_read_data_2;
 wire [15:0] ext_im,read_data2;
 wire [2:0] ALU_Control;
 wire [15:0] ALU_out;
 wire zero_flag;
 wire [15:0] PC_j, PC_beq, PC_2beq,PC_2bne,PC_bne;
 wire beq_control;
 wire [12:0] jump_shift;
 wire [15:0] mem_read_data;
 initial begin
  pc_current <= 16'd0;
 end
 always @(posedge clk)
 begin
   pc_current <= pc_next;
 end
 assign pc2 = pc_current + 16'd2;
 Instruction_Memory im(.pc(pc_current),.instruction(instr));
 assign jump_shift = {instr[11:0],1'b0};
 assign reg_write_dest = (reg_dst==1'b1) ? instr[5:3] :instr[8:6];
 assign reg_read_addr_1 = instr[11:9];
 assign reg_read_addr_2 = instr[8:6];


 GPRs reg_file
 (
  .clk(clk),
  .reg_write_en(reg_write),
  .reg_write_dest(reg_write_dest),
  .reg_write_data(reg_write_data),
  .reg_read_addr_1(reg_read_addr_1),
  .reg_read_data_1(reg_read_data_1),
  .reg_read_addr_2(reg_read_addr_2),
  .reg_read_data_2(reg_read_data_2)
 );

 assign ext_im = {{10{instr[5]}},instr[5:0]};

  ALU_Control_unit(.ALUOp(alu_op),.Opcode(instr[15:12]),.ALU_Cnt(ALU_Control));
 assign read_data2 = (alu_src==1'b1) ? ext_im : reg_read_data_2;
ALU.alu_unit(.a(reg_read_data_1),.b(read_data2),.alu_control(ALU_Control),.resul
t(ALU_out),.zero(zero_flag));
 assign PC_beq = pc2 + {ext_im[14:0],1'b0};
 assign PC_bne = pc2 + {ext_im[14:0],1'b0};
 assign beq_control = beq & zero_flag;
 assign bne_control = bne & (~zero_flag);
 assign PC_2beq = (beq_control==1'b1) ? PC_beq : pc2;
 assign PC_2bne = (bne_control==1'b1) ? PC_bne : PC_2beq;
 assign PC_j = {pc2[15:13],jump_shift};
 assign pc_next = (jump == 1'b1) ? PC_j :  PC_2bne;

  Data_Memory dm
   (
    .clk(clk),
    .mem_access_addr(ALU_out),
    .mem_write_data(reg_read_data_2),
    .mem_write_en(mem_write),
    .mem_read(mem_read),
    .mem_read_data(mem_read_data)
   );


 assign reg_write_data = (mem_to_reg == 1'b1)?  mem_read_data: ALU_out;
 assign opcode = instr[15:12];
endmodule
```

7. Verilog code for the Control Unit of the RISC processor:

```
`timescale 1ns / 1ps
module Control_Unit(
      input[3:0] opcode,
      output reg[1:0] alu_op,
      output reg jump,beq,bne,mem_read,mem_write,alu_src,reg_dst,mem_to_reg,reg_write
    );
always @(*)
begin
 case(opcode)
 4'b0000:
   begin
    reg_dst = 1'b0;
    alu_src = 1'b1;
    mem_to_reg = 1'b1;
    reg_write = 1'b1;
    mem_read = 1'b1;
    mem_write = 1'b0;
    beq = 1'b0;
    bne = 1'b0;
    alu_op = 2'b10;
    jump = 1'b0;
   end
 4'b0001:
   begin
    reg_dst = 1'b0;
    alu_src = 1'b1;
    mem_to_reg = 1'b0;
    reg_write = 1'b0;
    mem_read = 1'b0;
    mem_write = 1'b1;
    beq = 1'b0;
    bne = 1'b0;
    alu_op = 2'b10;
    jump = 1'b0;
   end
 4'b0010:
   begin
    reg_dst = 1'b1;
    alu_src = 1'b0;
    mem_to_reg = 1'b0;
    reg_write = 1'b1;
    mem_read = 1'b0;
    mem_write = 1'b0;
    beq = 1'b0;
    bne = 1'b0;
    alu_op = 2'b00;
    jump = 1'b0;
   end
 4'b0011:
   begin
    reg_dst = 1'b1;
    alu_src = 1'b0;
    mem_to_reg = 1'b0;
    reg_write = 1'b1;
    mem_read = 1'b0;
    mem_write = 1'b0;
    beq = 1'b0;
    bne = 1'b0;
    alu_op = 2'b00;
    jump = 1'b0;
   end
 4'b0100:
   begin
    reg_dst = 1'b1;
    alu_src = 1'b0;
    mem_to_reg = 1'b0;
    reg_write = 1'b1;
    mem_read = 1'b0;
    mem_write = 1'b0;
    beq = 1'b0;
    bne = 1'b0;
    alu_op = 2'b00;
    jump = 1'b0;
   end
 4'b0101:
   begin
    reg_dst = 1'b1;
    alu_src = 1'b0;
    mem_to_reg = 1'b0;
    reg_write = 1'b1;
    mem_read = 1'b0;
    mem_write = 1'b0;
    beq = 1'b0;
    bne = 1'b0;
    alu_op = 2'b00;
    jump = 1'b0;
   end
 4'b0110:
   begin
    reg_dst = 1'b1;
    alu_src = 1'b0;
    mem_to_reg = 1'b0;
    reg_write = 1'b1;
    mem_read = 1'b0;
    mem_write = 1'b0;
    beq = 1'b0;
    bne = 1'b0;
    alu_op = 2'b00;
    jump = 1'b0;
   end
 4'b0111:
   begin
    reg_dst = 1'b1;
    alu_src = 1'b0;
    mem_to_reg = 1'b0;
    reg_write = 1'b1;
    mem_read = 1'b0;
    mem_write = 1'b0;
    beq = 1'b0;
    bne = 1'b0;
    alu_op = 2'b00;
    jump = 1'b0;
   end
```

```
4'b1000:
  begin
   reg_dst = 1'b1;
   alu_src = 1'b0;
   mem_to_reg = 1'b0;
   reg_write = 1'b1;
   mem_read = 1'b0;
   mem_write = 1'b0;
   beq = 1'b0;
   bne = 1'b0;
   alu_op = 2'b00;
   jump = 1'b0;
  end
4'b1001:
  begin
   reg_dst = 1'b1;
   alu_src = 1'b0;
   mem_to_reg = 1'b0;
   reg_write = 1'b1;
   mem_read = 1'b0;
   mem_write = 1'b0;
   beq = 1'b0;
   bne = 1'b0;
   alu_op = 2'b00;
   jump = 1'b0;
  end
4'b1011:
  begin
   reg_dst = 1'b0;
   alu_src = 1'b0;
   mem_to_reg = 1'b0;
   reg_write = 1'b0;
   mem_read = 1'b0;
   mem_write = 1'b0;
   beq = 1'b1;
   bne = 1'b0;
   alu_op = 2'b01;
   jump = 1'b0;
  end
4'b1100:
  begin
   reg_dst = 1'b0;
   alu_src = 1'b0;
   mem_to_reg = 1'b0;
   reg_write = 1'b0;
   mem_read = 1'b0;
   mem_write = 1'b0;
   beq = 1'b0;
   bne = 1'b1;
   alu_op = 2'b01;
   jump = 1'b0;
  end
4'b1101:
  begin
   reg_dst = 1'b0;
   alu_src = 1'b0;
   mem_to_reg = 1'b0;
   reg_write = 1'b0;
   mem_read = 1'b0;
   mem_write = 1'b0;
   beq = 1'b0;
   bne = 1'b0;
   alu_op = 2'b00;
   jump = 1'b1;
   end
 default: begin
    reg_dst = 1'b1;
    alu_src = 1'b0;
    mem_to_reg = 1'b0;
    reg_write = 1'b1;
    mem_read = 1'b0;
    mem_write = 1'b0;
    beq = 1'b0;
    bne = 1'b0;
    alu_op = 2'b00;
    jump = 1'b0;
   end
 endcase
 end

endmodule
```

8. Verilog code for the 16-bit RISC processor:

```
`timescale 1ns / 1ps
module Risc_16_bit(
 input clk
);
 wire jump,bne,beq,mem_read,mem_write,alu_src,reg_dst,mem_to_reg,reg_write;
 wire[1:0] alu_op;
 wire [3:0] opcode;
 Datapath_Unit DU
 (
  .clk(clk),
  .jump(jump),
  .beq(beq),
  .mem_read(mem_read),
  .mem_write(mem_write),
  .alu_src(alu_src),
  .reg_dst(reg_dst),
  .mem_to_reg(mem_to_reg),
  .reg_write(reg_write),
  .bne(bne),
  .alu_op(alu_op),
  .opcode(opcode)
 );
 Control_Unit control
 (
  .opcode(opcode),
  .reg_dst(reg_dst),
  .mem_to_reg(mem_to_reg),
  .alu_op(alu_op),
  .jump(jump),
  .bne(bne),
  .beq(beq),
  .mem_read(mem_read),
  .mem_write(mem_write),
  .alu_src(alu_src),
  .reg_write(reg_write)
 );

endmodule
```

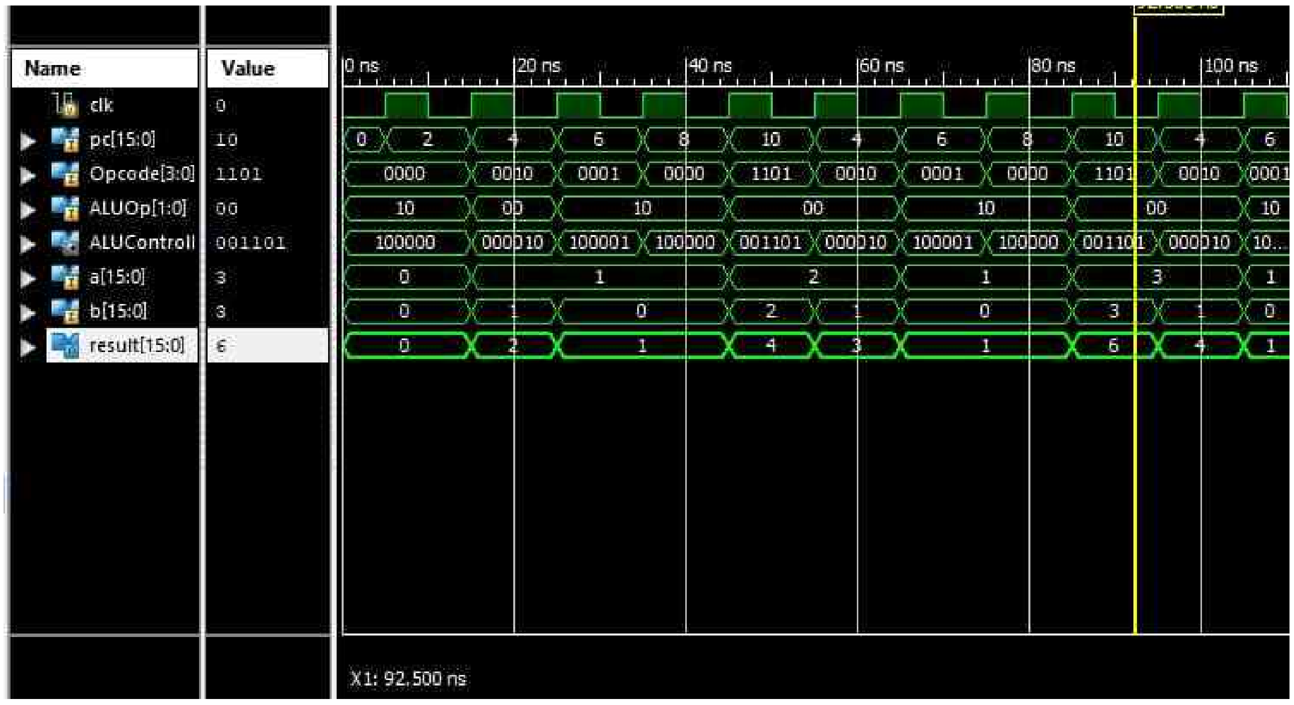
9. Verilog Testbench code for the 16-bit RISC Processor:

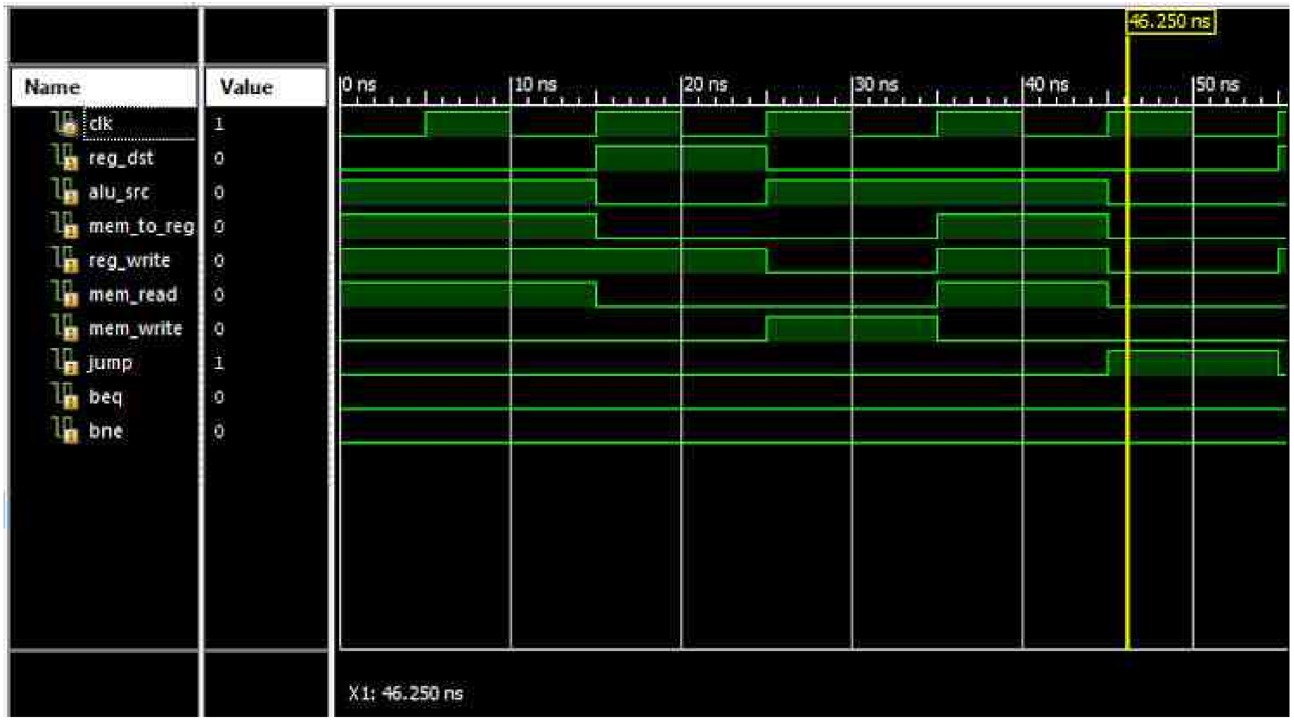
```
`timescale 1ns / 1ps
`include "Parameter.v"
module test_Risc_16_bit;

 reg clk;

 Risc_16_bit uut (
  .clk(clk)
 );

 initial
  begin
   clk <=0;
   `simulation_time;
   $finish;
  end

 always
  begin
   #5 clk = ~clk;
  end

endmodule
```

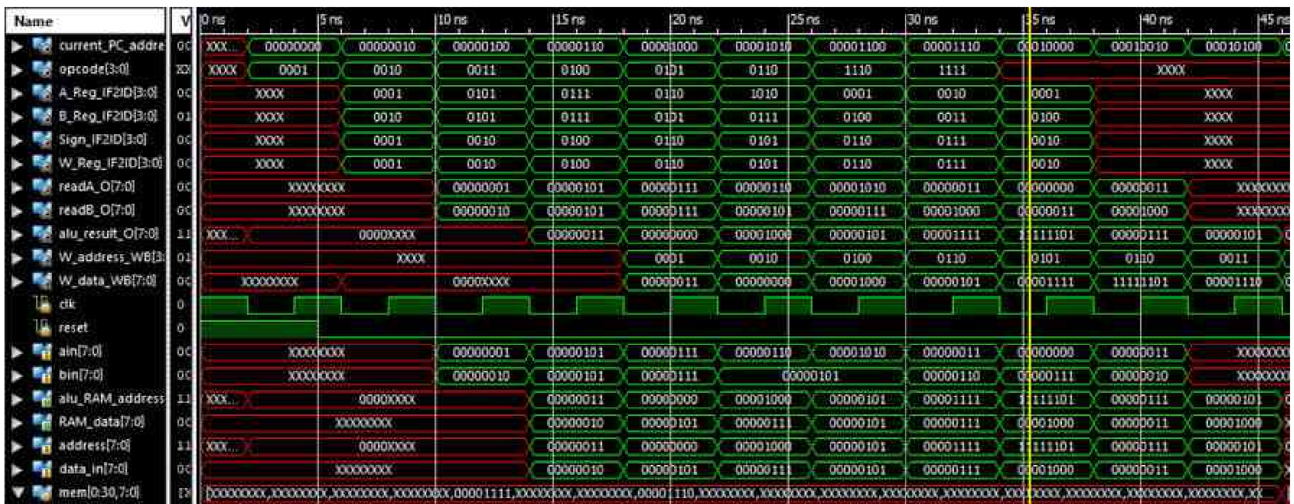
Parameter file:

```
`ifndef PARAMETER_H_
`define PARAMETER_H_
`define col 16
`define row_i 15
`define row_d 8
`define filename "./test/50001111_50001212.o"
`define simulation_time #160

`endif
```

OUTPUT WAVEFORMS:


Name
Value
clk 0
pc[15:0] 10
Opcode[3:0] 1101
ALUOp[1:0] 00
ALUControl 001101
a[15:0] 3
b[15:0] 3
result[15:0] 6
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
X1: 92.500 ns


Name
Value
clk 1
reg_dst 0
alu_src 0
mem_to_reg 0
reg_write 0
mem_read 0
mem_write 0
jump 1
beq 0
bne 0
46.250 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
X1: 46.250 ns


Name
current_PC_addre
opcode[3:0]
A_Reg_IF2ID[3:0]
B_Reg_IF2ID[3:0]
Sign_IF2ID[3:0]
W_Reg_IF2ID[3:0]
readA_O[7:0]
readB_O[7:0]
alu_result_O[7:0]
W_address_WB[3:
W_data_WB[7:0]
clk
reset
ain[7:0]
bin[7:0]
alu_RAM_address
RAM_data[7:0]
address[7:0]
data_in[7:0]
mem[0:30,7:0]
0 ns
5 ns
10 ns
15 ns
20 ns
25 ns
30 ns
35 ns
40 ns
45 ns