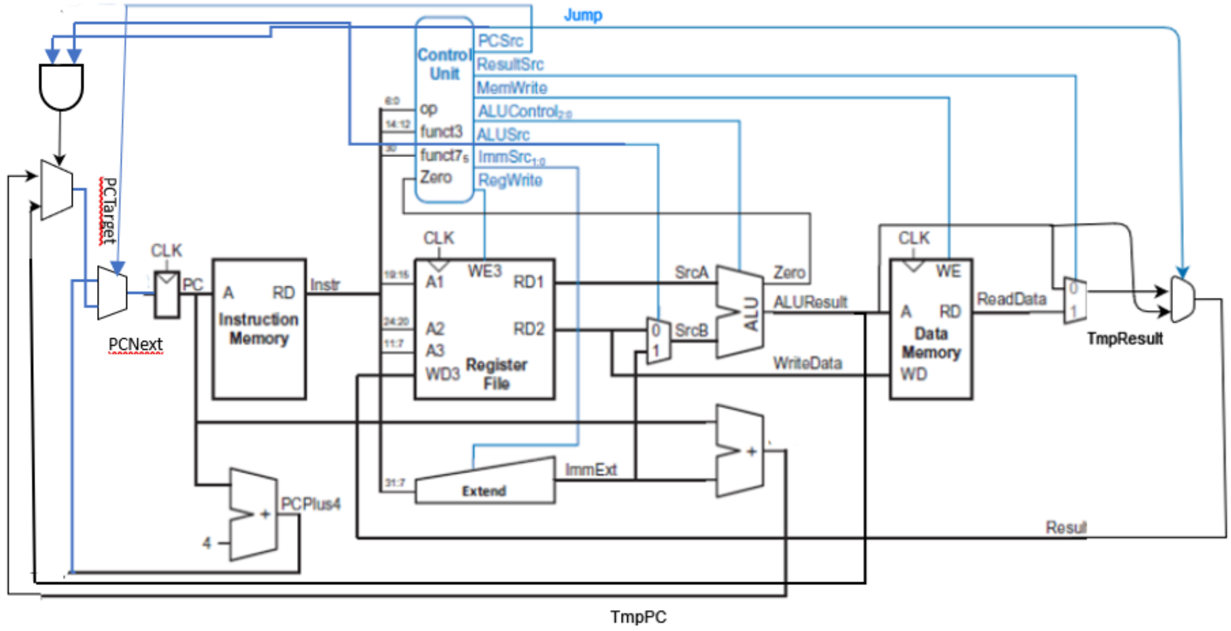
RISC-V REPORT

Abdelrahman Atef Youssef
Submitted to: Eng. Sara
Design Track

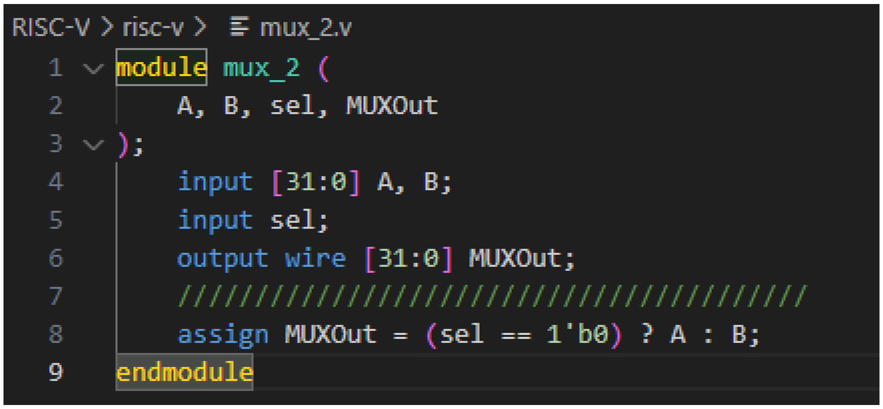
Table of Contents

Architecture:

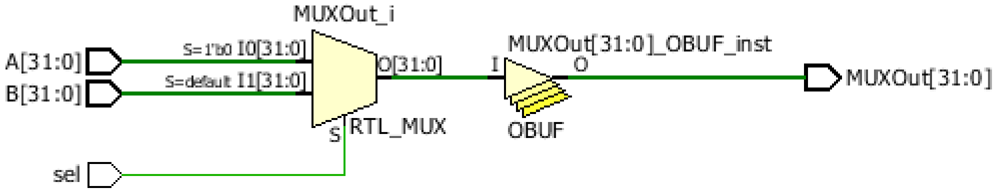


Codes & Block diagrams:

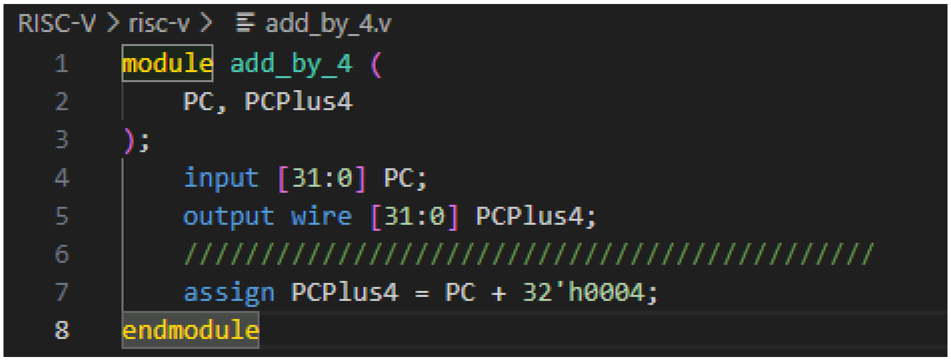
1) mux_2:



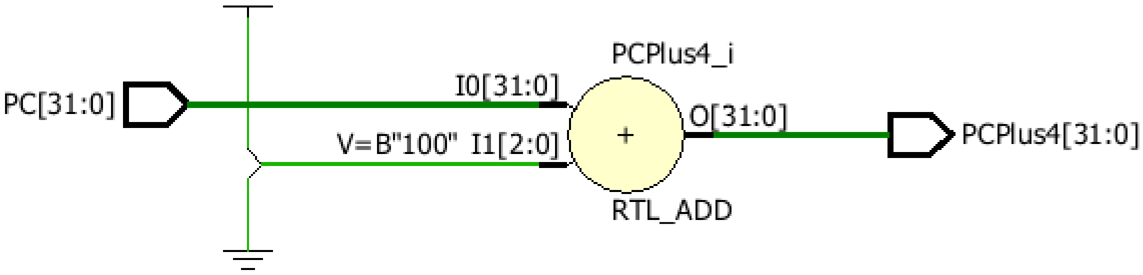
```verilog
module mux_2 (
    A, B, sel, MUXOut
);
    input [31:0] A, B;
    input sel;
    output wire [31:0] MUXOut;
    ////////////////////////////////////////
    assign MUXOut = (sel == 1'b0) ? A : B;
endmodule
```



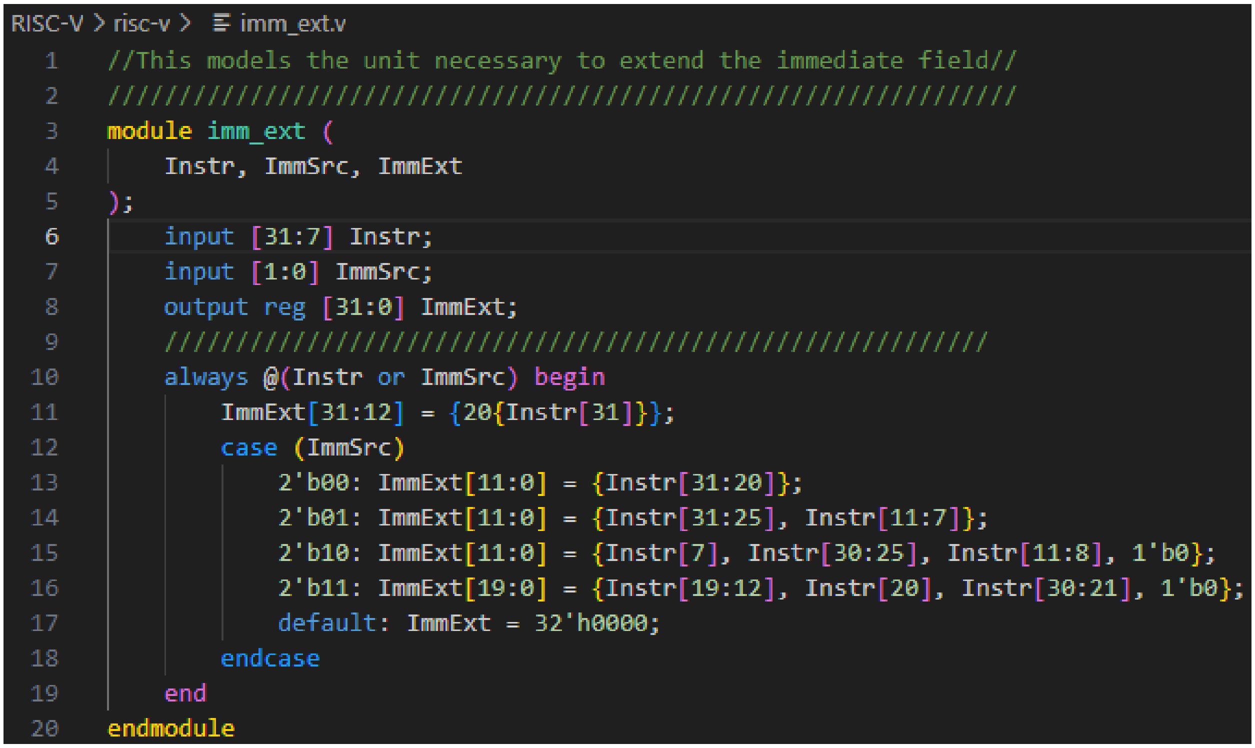
2) add_by_4:



```verilog
module add_by_4 (
    PC, PCPlus4
);
    input [31:0] PC;
    output wire [31:0] PCPlus4;
    ////////////////////////////////////////////
    assign PCPlus4 = PC + 32'h0004;
endmodule
```

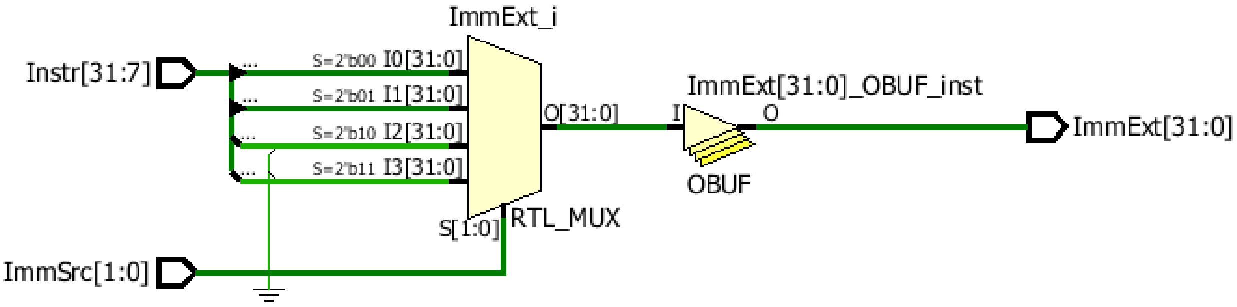


3) imm_ext:

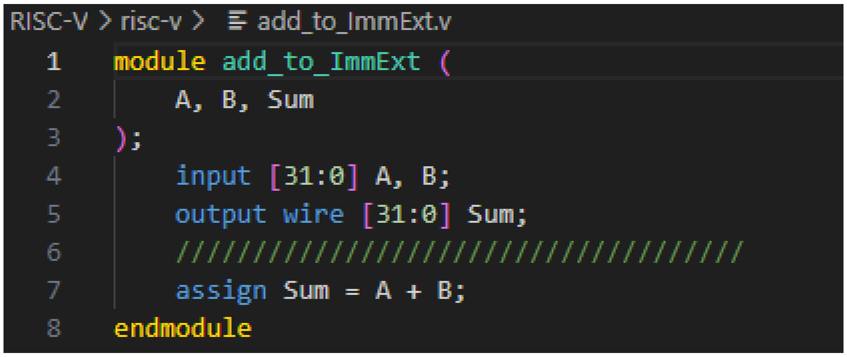


RISC-V > risc-v > imm_ext.v

```
//This models the unit necessary to extend the immediate field//
/////////////////////////////////////////////////////////////////
module imm_ext (
    Instr, ImmSrc, ImmExt
);
    input [31:7] Instr;
    input [1:0] ImmSrc;
    output reg [31:0] ImmExt;
    /////////////////////////////////////////////////////////////
    always @(Instr or ImmSrc) begin
        ImmExt[31:12] = {20{Instr[31]}};
        case (ImmSrc)
            2'b00: ImmExt[11:0] = {Instr[31:20]};
            2'b01: ImmExt[11:0] = {Instr[31:25], Instr[11:7]};
            2'b10: ImmExt[11:0] = {Instr[7], Instr[30:25], Instr[11:8], 1'b0};
            2'b11: ImmExt[19:0] = {Instr[19:12], Instr[20], Instr[30:21], 1'b0};
            default: ImmExt = 32'h0000;
        endcase
    end
endmodule
```

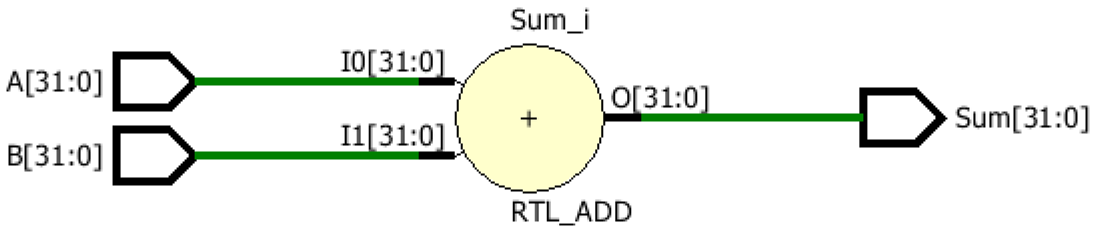


4) add_to_ImmExt:

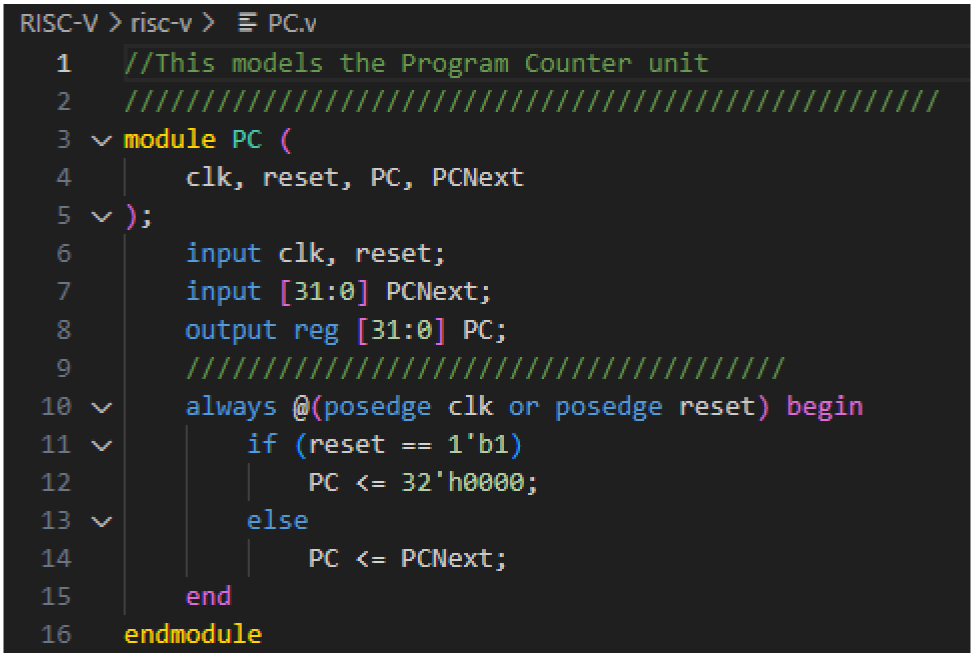


RISC-V > risc-v > add_to_ImmExt.v

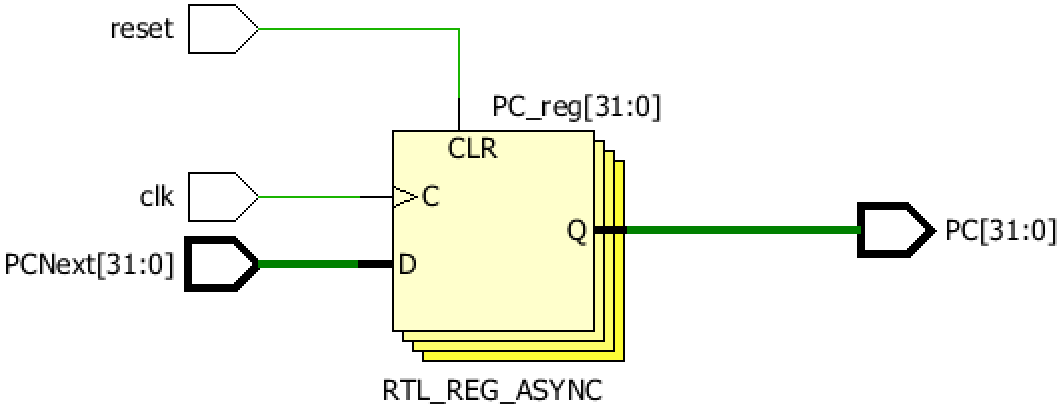
```
module add_to_ImmExt (
    A, B, Sum
);
    input [31:0] A, B;
    output wire [31:0] Sum;
    /////////////////////////////////////
    assign Sum = A + B;
endmodule
```



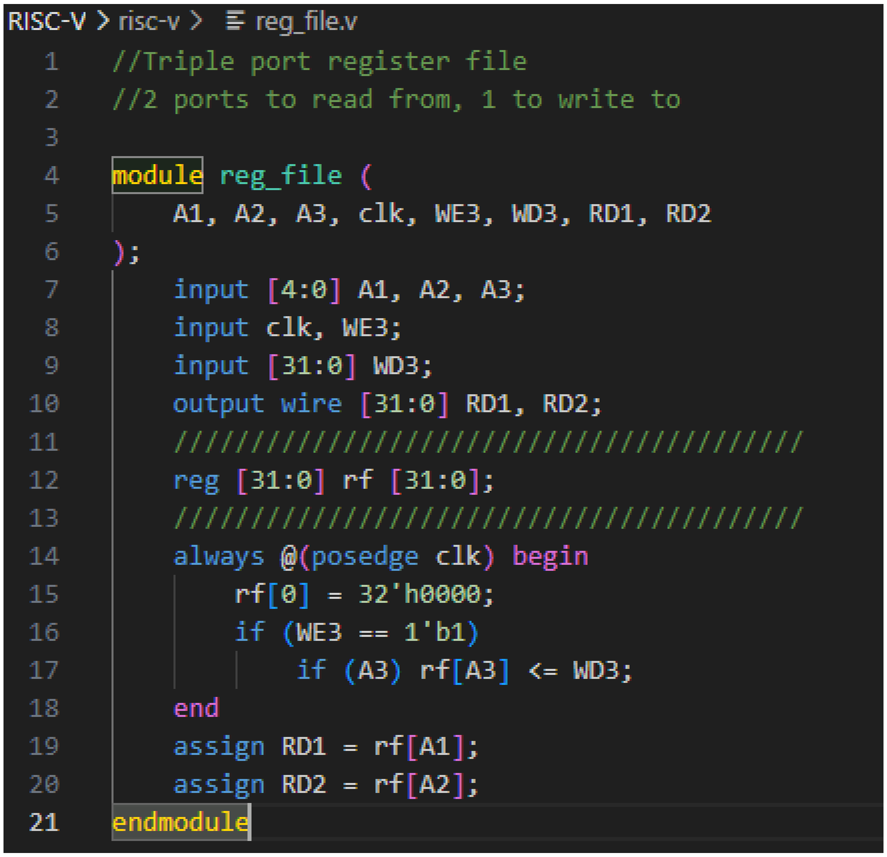
5) PC:



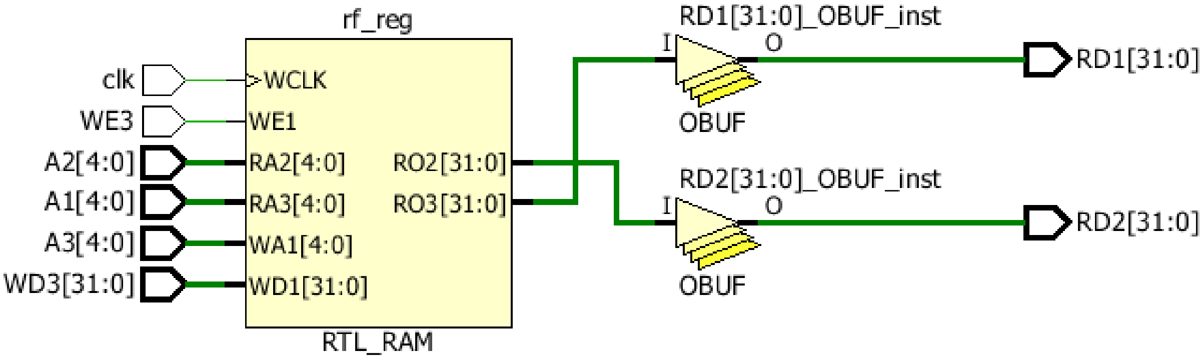
```
//This models the Program Counter unit
/////////////////////////////////////////////////////
module PC (
    clk, reset, PC, PCNext
);
    input clk, reset;
    input [31:0] PCNext;
    output reg [31:0] PC;
    ///////////////////////////////////////
    always @(posedge clk or posedge reset) begin
        if (reset == 1'b1)
            PC <= 32'h0000;
        else
            PC <= PCNext;
    end
endmodule
```



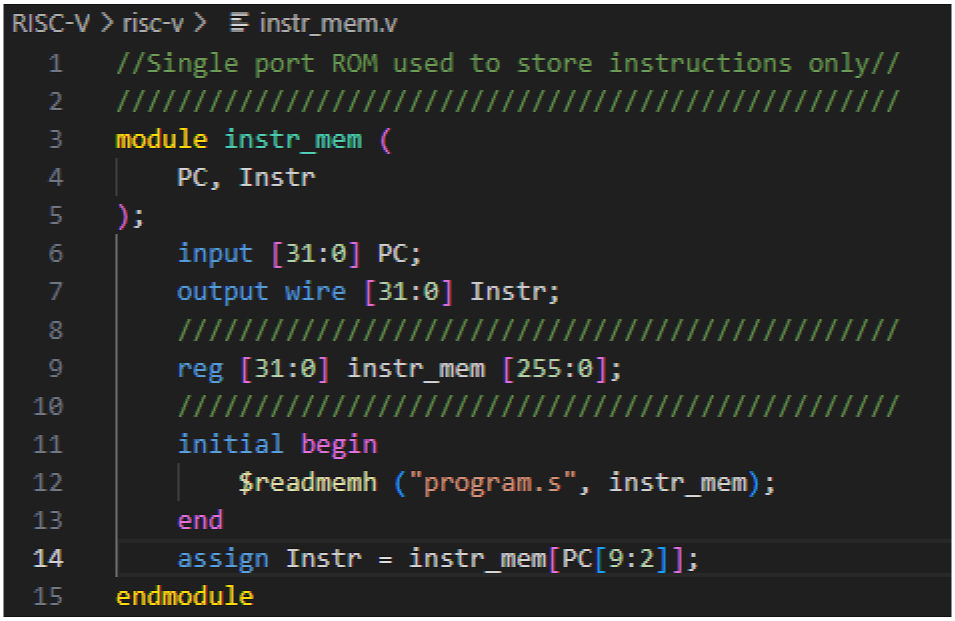
6) reg_file:



```verilog
//Triple port register file
//2 ports to read from, 1 to write to

module reg_file (
    A1, A2, A3, clk, WE3, WD3, RD1, RD2
);
    input [4:0] A1, A2, A3;
    input clk, WE3;
    input [31:0] WD3;
    output wire [31:0] RD1, RD2;
    //////////////////////////////////////////
    reg [31:0] rf [31:0];
    //////////////////////////////////////////
    always @(posedge clk) begin
        rf[0] = 32'h0000;
        if (WE3 == 1'b1)
            if (A3) rf[A3] <= WD3;
    end
    assign RD1 = rf[A1];
    assign RD2 = rf[A2];
endmodule
```

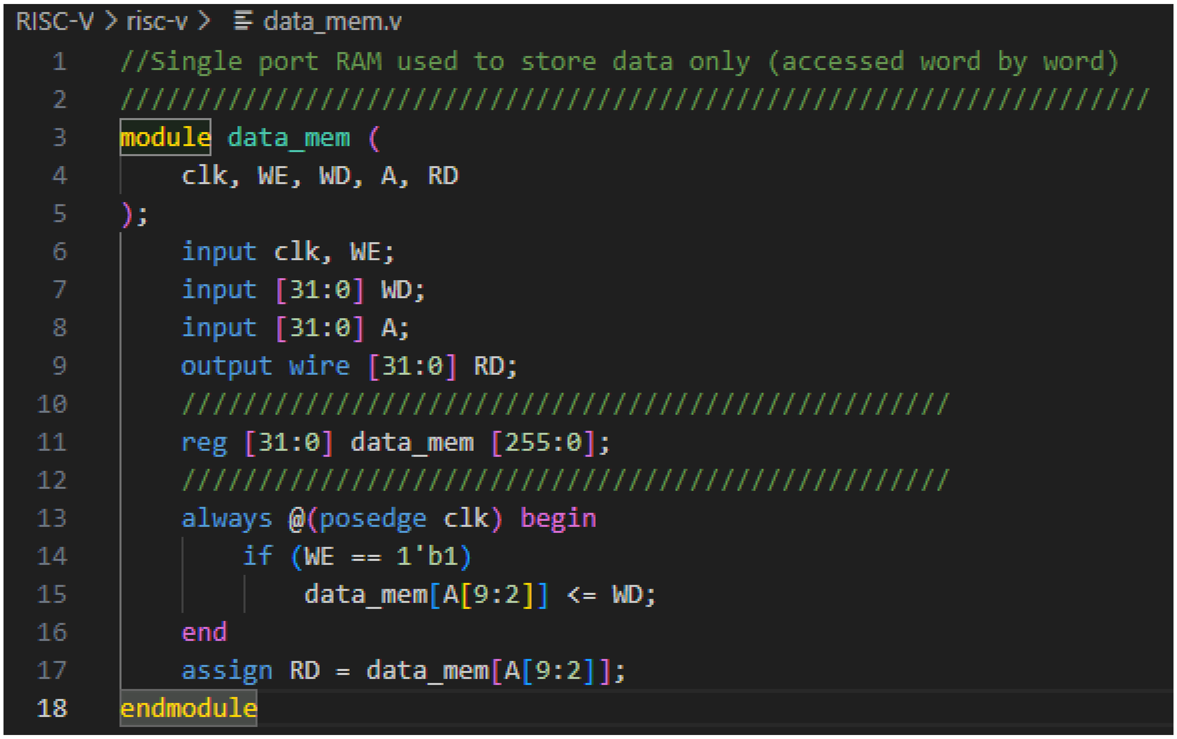


7) instr_mem:



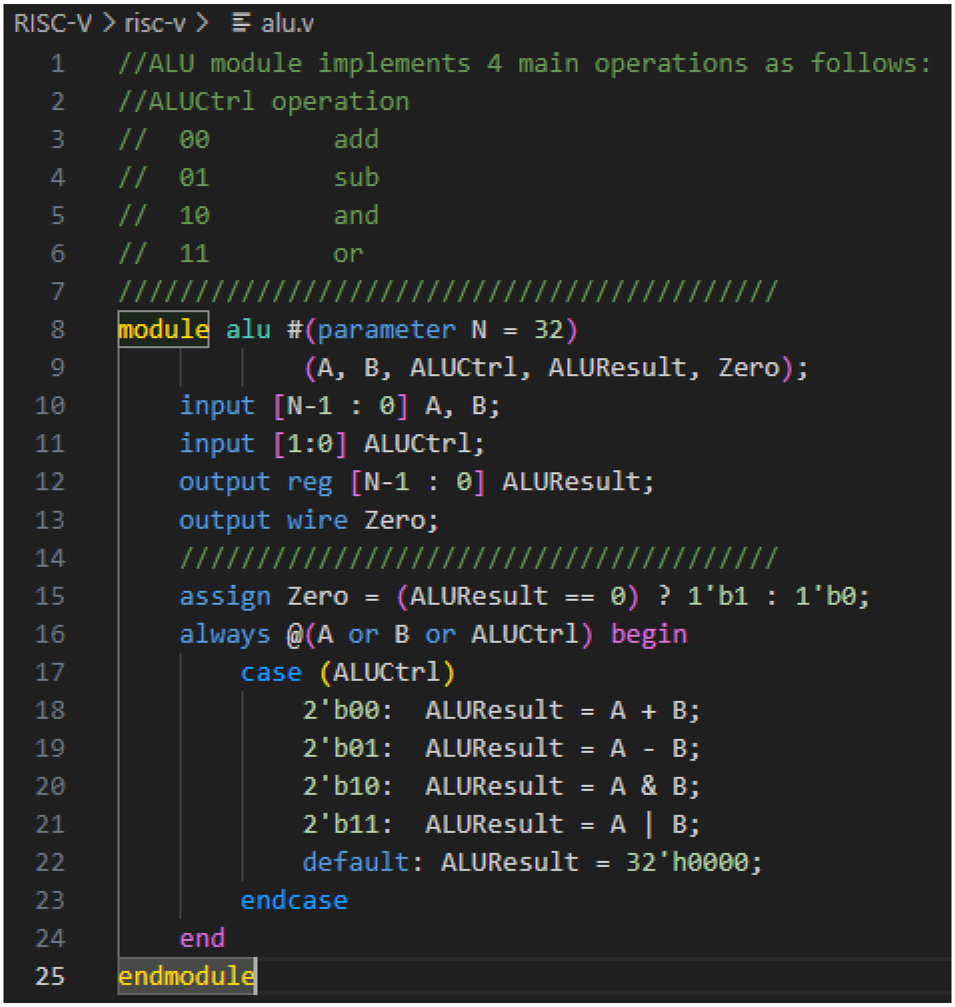
```
//Single port ROM used to store instructions only//
///////////////////////////////////////////////////
module instr_mem (
    PC, Instr
);
    input [31:0] PC;
    output wire [31:0] Instr;
    ///////////////////////////////////////////////
    reg [31:0] instr_mem [255:0];
    ///////////////////////////////////////////////
    initial begin
        $readmemh ("program.s", instr_mem);
    end
    assign Instr = instr_mem[PC[9:2]];
endmodule
```

8) data_mem:



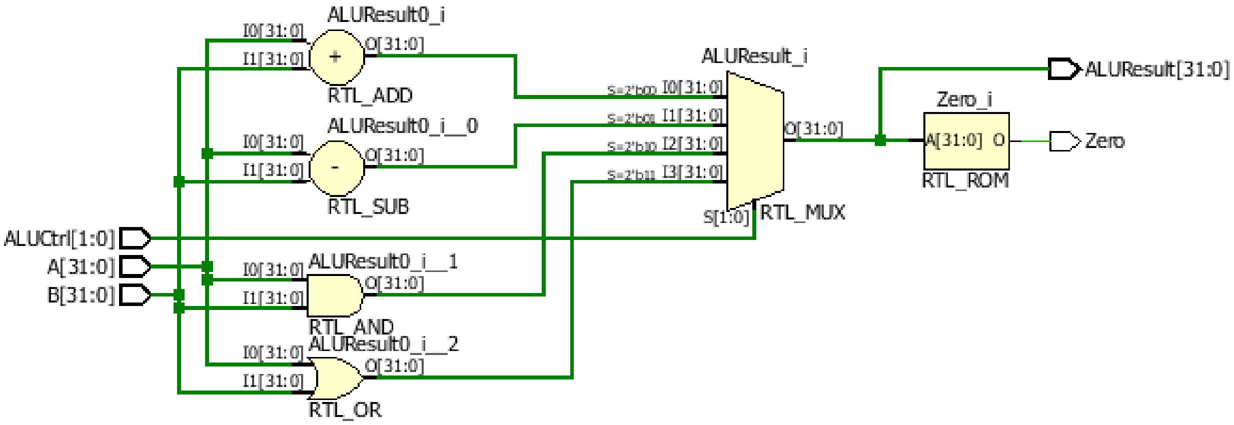
```
//Single port RAM used to store data only (accessed word by word)
///////////////////////////////////////////////////////////////////
module data_mem (
    clk, WE, WD, A, RD
);
    input clk, WE;
    input [31:0] WD;
    input [31:0] A;
    output wire [31:0] RD;
    //////////////////////////////////////////////////
    reg [31:0] data_mem [255:0];
    //////////////////////////////////////////////////
    always @(posedge clk) begin
        if (WE == 1'b1)
            data_mem[A[9:2]] <= WD;
    end
    assign RD = data_mem[A[9:2]];
endmodule
```

9) alu:

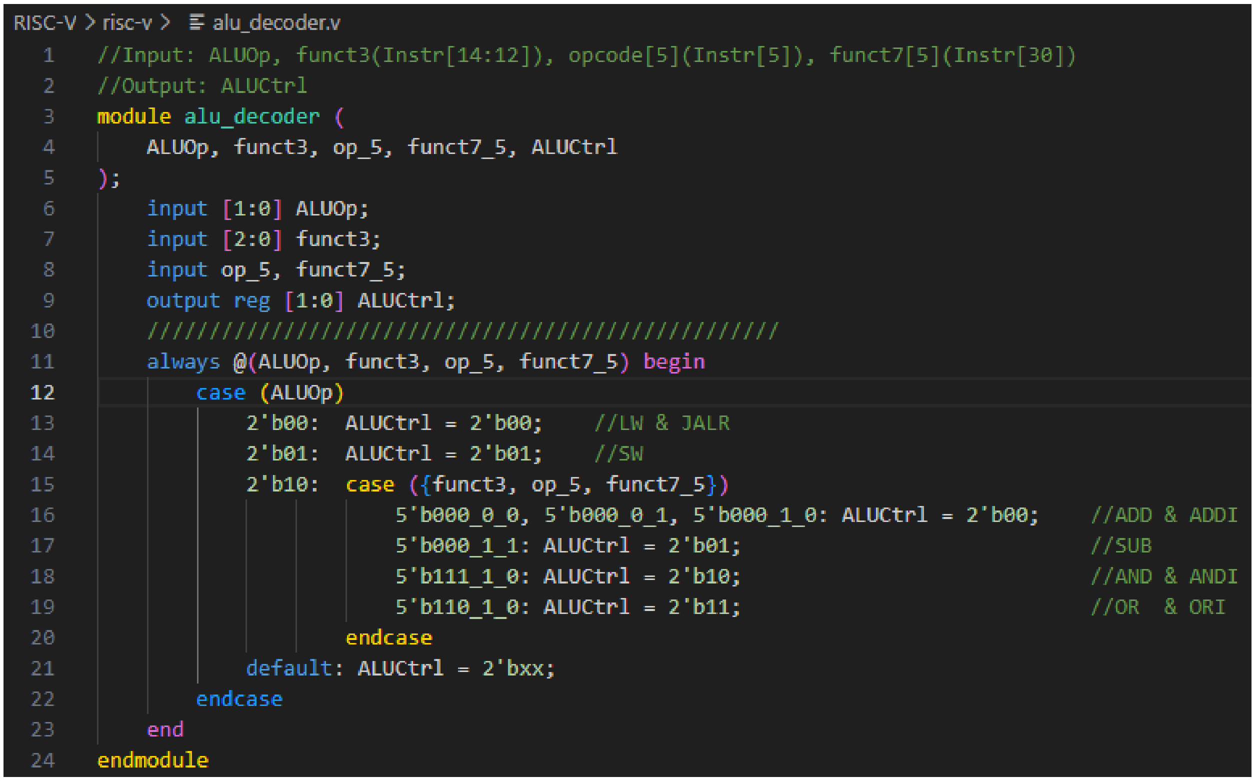


RISC-V > risc-v > alu.v

```verilog
//ALU module implements 4 main operations as follows:
//ALUCtrl operation
//  00        add
//  01        sub
//  10        and
//  11        or
///////////////////////////////////////////
module alu #(parameter N = 32)
            (A, B, ALUCtrl, ALUResult, Zero);
    input [N-1 : 0] A, B;
    input [1:0] ALUCtrl;
    output reg [N-1 : 0] ALUResult;
    output wire Zero;
    ///////////////////////////////////////
    assign Zero = (ALUResult == 0) ? 1'b1 : 1'b0;
    always @(A or B or ALUCtrl) begin
        case (ALUCtrl)
            2'b00:  ALUResult = A + B;
            2'b01:  ALUResult = A - B;
            2'b10:  ALUResult = A & B;
            2'b11:  ALUResult = A | B;
            default: ALUResult = 32'h0000;
        endcase
    end
endmodule
```

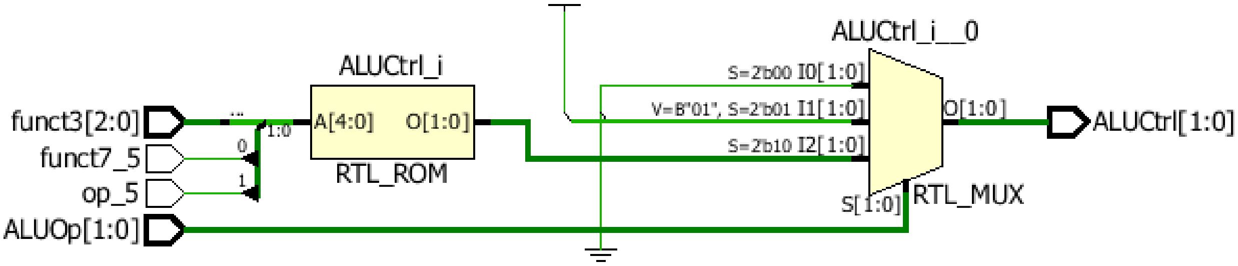


10)alu_decoder:

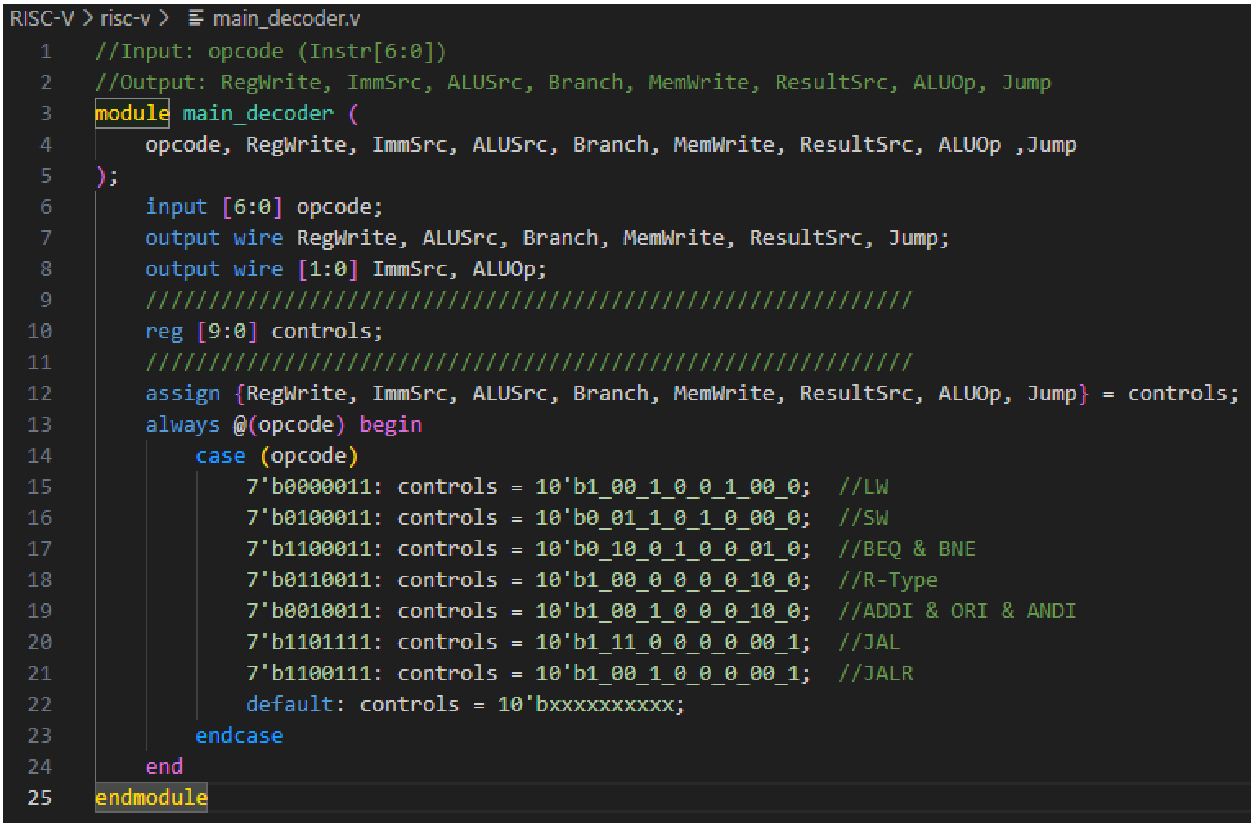


RISC-V > risc-v > alu_decoder.v

```verilog
//Input: ALUOp, funct3(Instr[14:12]), opcode[5](Instr[5]), funct7[5](Instr[30])
//Output: ALUCtrl
module alu_decoder (
    ALUOp, funct3, op_5, funct7_5, ALUCtrl
);
    input [1:0] ALUOp;
    input [2:0] funct3;
    input op_5, funct7_5;
    output reg [1:0] ALUCtrl;
    ///////////////////////////////////////////////////
    always @(ALUOp, funct3, op_5, funct7_5) begin
        case (ALUOp)
            2'b00:  ALUCtrl = 2'b00;    //LW & JALR
            2'b01:  ALUCtrl = 2'b01;    //SW
            2'b10:  case ({funct3, op_5, funct7_5})
                        5'b000_0_0, 5'b000_0_1, 5'b000_1_0: ALUCtrl = 2'b00;    //ADD & ADDI
                        5'b000_1_1: ALUCtrl = 2'b01;                            //SUB
                        5'b111_1_0: ALUCtrl = 2'b10;                            //AND & ANDI
                        5'b110_1_0: ALUCtrl = 2'b11;                            //OR  & ORI
                    endcase
            default: ALUCtrl = 2'bxx;
        endcase
    end
endmodule
```

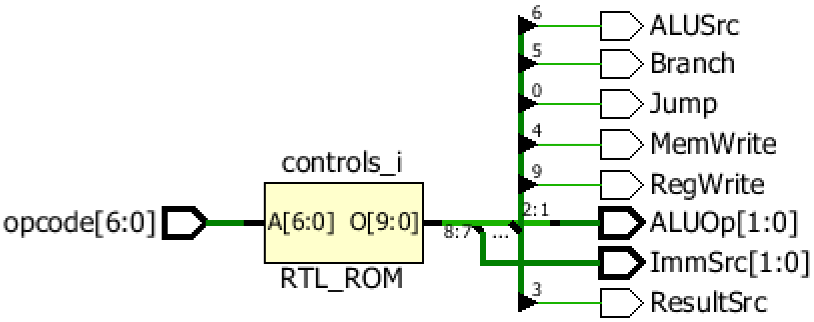


11) main_decoder:

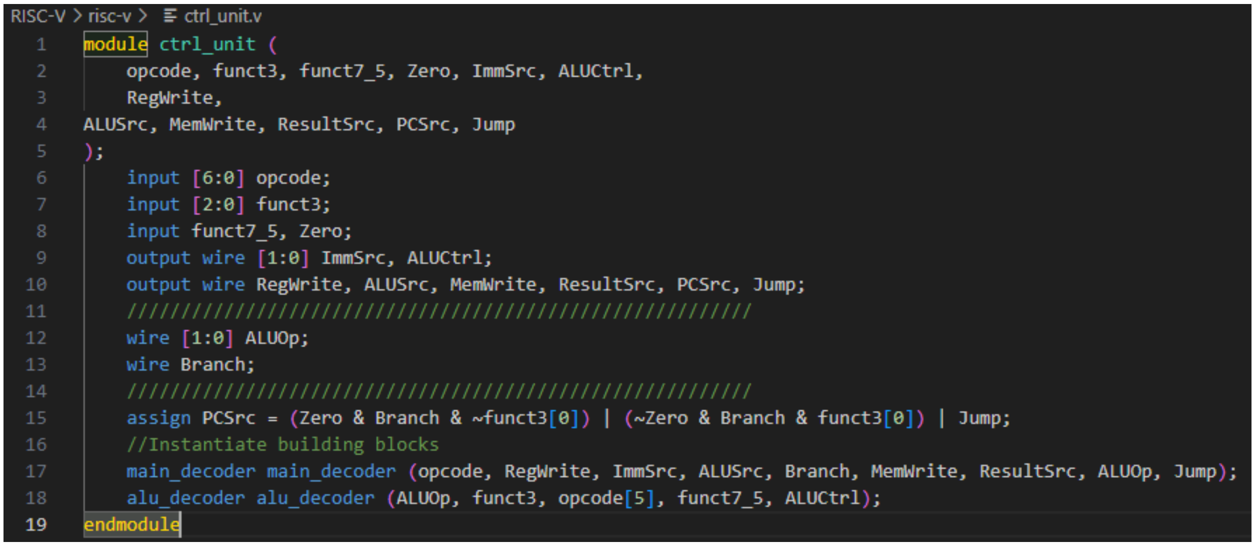


RISC-V > risc-v > ≡ main_decoder.v

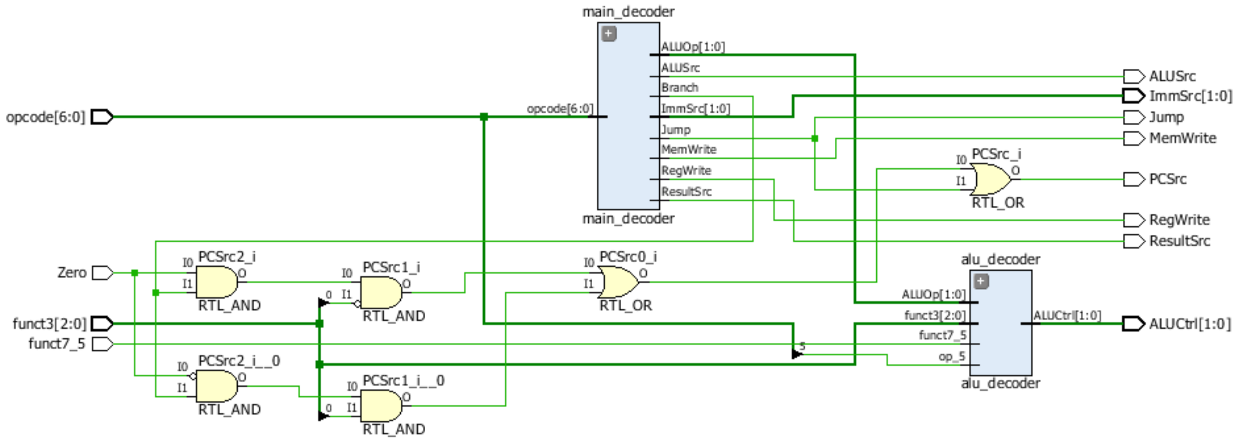
```
//Input: opcode (Instr[6:0])
//Output: RegWrite, ImmSrc, ALUSrc, Branch, MemWrite, ResultSrc, ALUOp, Jump
module main_decoder (
    opcode, RegWrite, ImmSrc, ALUSrc, Branch, MemWrite, ResultSrc, ALUOp ,Jump
);
    input [6:0] opcode;
    output wire RegWrite, ALUSrc, Branch, MemWrite, ResultSrc, Jump;
    output wire [1:0] ImmSrc, ALUOp;
    ////////////////////////////////////////////////////////////
    reg [9:0] controls;
    ////////////////////////////////////////////////////////////
    assign {RegWrite, ImmSrc, ALUSrc, Branch, MemWrite, ResultSrc, ALUOp, Jump} = controls;
    always @(opcode) begin
        case (opcode)
            7'b0000011: controls = 10'b1_00_1_0_0_1_00_0;  //LW
            7'b0100011: controls = 10'b0_01_1_0_1_0_00_0;  //SW
            7'b1100011: controls = 10'b0_10_0_1_0_0_01_0;  //BEQ & BNE
            7'b0110011: controls = 10'b1_00_0_0_0_0_10_0;  //R-Type
            7'b0010011: controls = 10'b1_00_1_0_0_0_10_0;  //ADDI & ORI & ANDI
            7'b1101111: controls = 10'b1_11_0_0_0_0_00_1;  //JAL
            7'b1100111: controls = 10'b1_00_1_0_0_0_00_1;  //JALR
            default: controls = 10'bxxxxxxxxxx;
        endcase
    end
endmodule
```



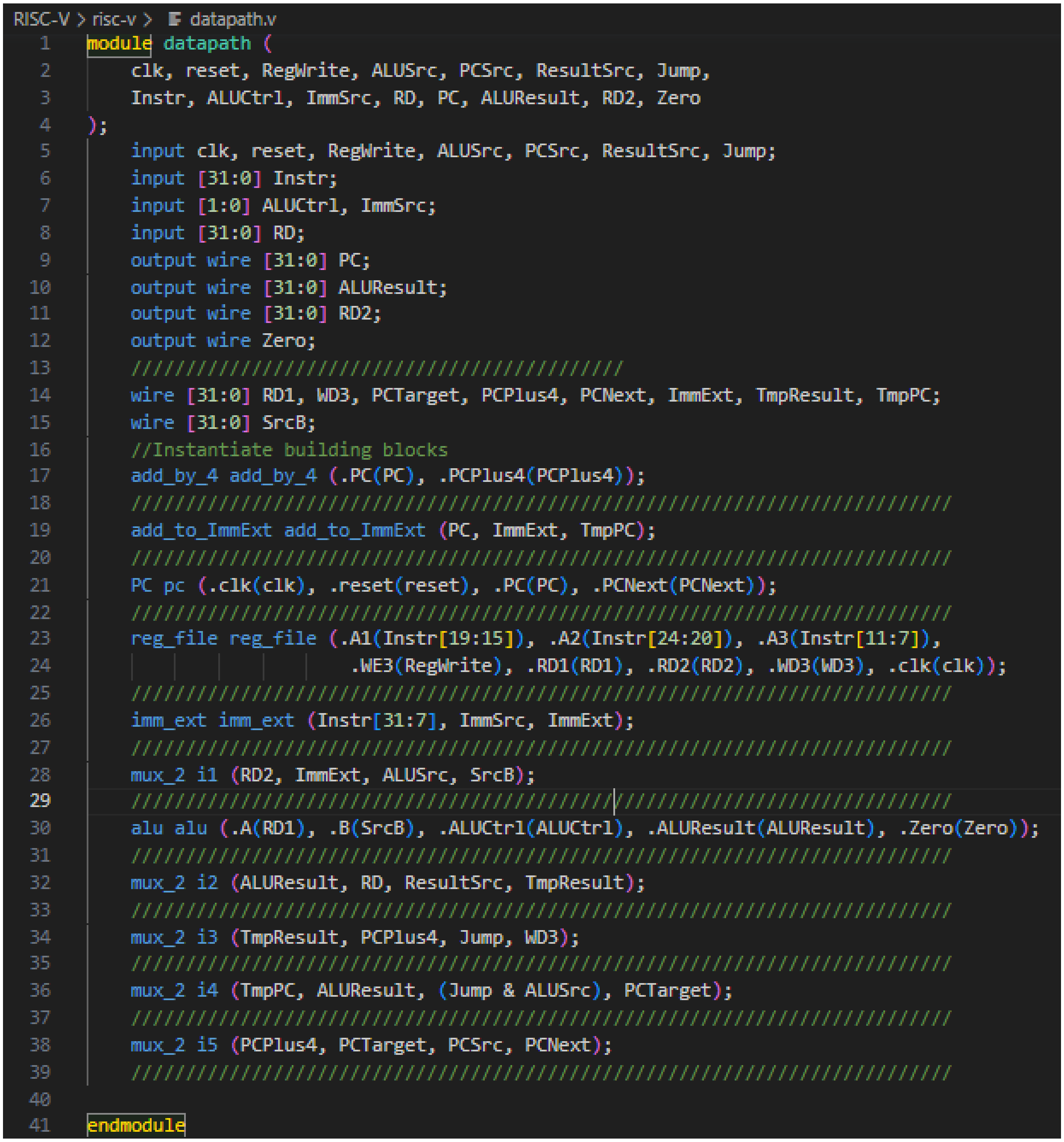
12) ctrl_unit:



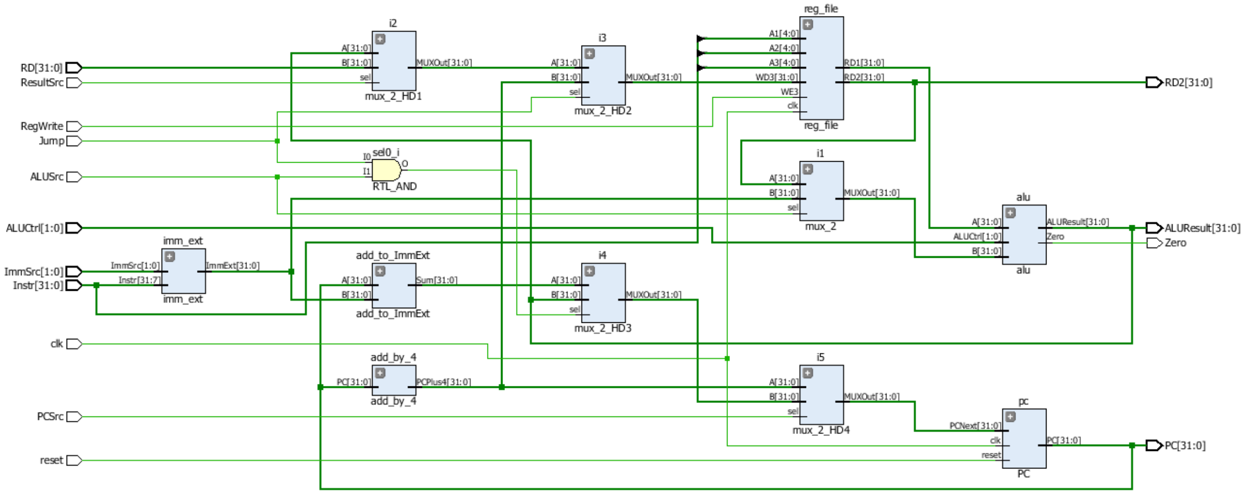
```
module ctrl_unit (
    opcode, funct3, funct7_5, Zero, ImmSrc, ALUCtrl,
    RegWrite,
ALUSrc, MemWrite, ResultSrc, PCSrc, Jump
);
    input [6:0] opcode;
    input [2:0] funct3;
    input funct7_5, Zero;
    output wire [1:0] ImmSrc, ALUCtrl;
    output wire RegWrite, ALUSrc, MemWrite, ResultSrc, PCSrc, Jump;
    /////////////////////////////////////////////////////////
    wire [1:0] ALUOp;
    wire Branch;
    /////////////////////////////////////////////////////////
    assign PCSrc = (Zero & Branch & ~funct3[0]) | (~Zero & Branch & funct3[0]) | Jump;
    //Instantiate building blocks
    main_decoder main_decoder (opcode, RegWrite, ImmSrc, ALUSrc, Branch, MemWrite, ResultSrc, ALUOp, Jump);
    alu_decoder alu_decoder (ALUOp, funct3, opcode[5], funct7_5, ALUCtrl);
endmodule
```



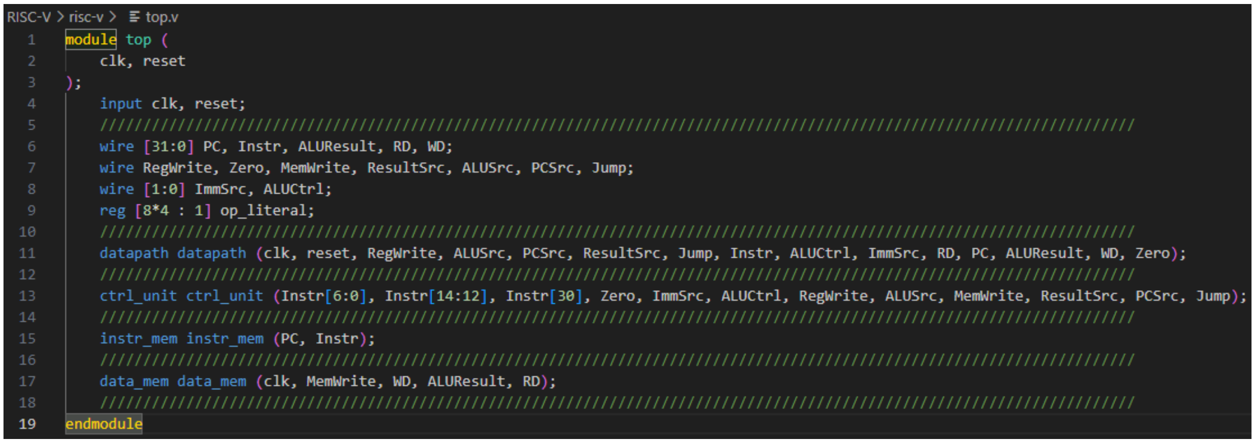
13)datapath:



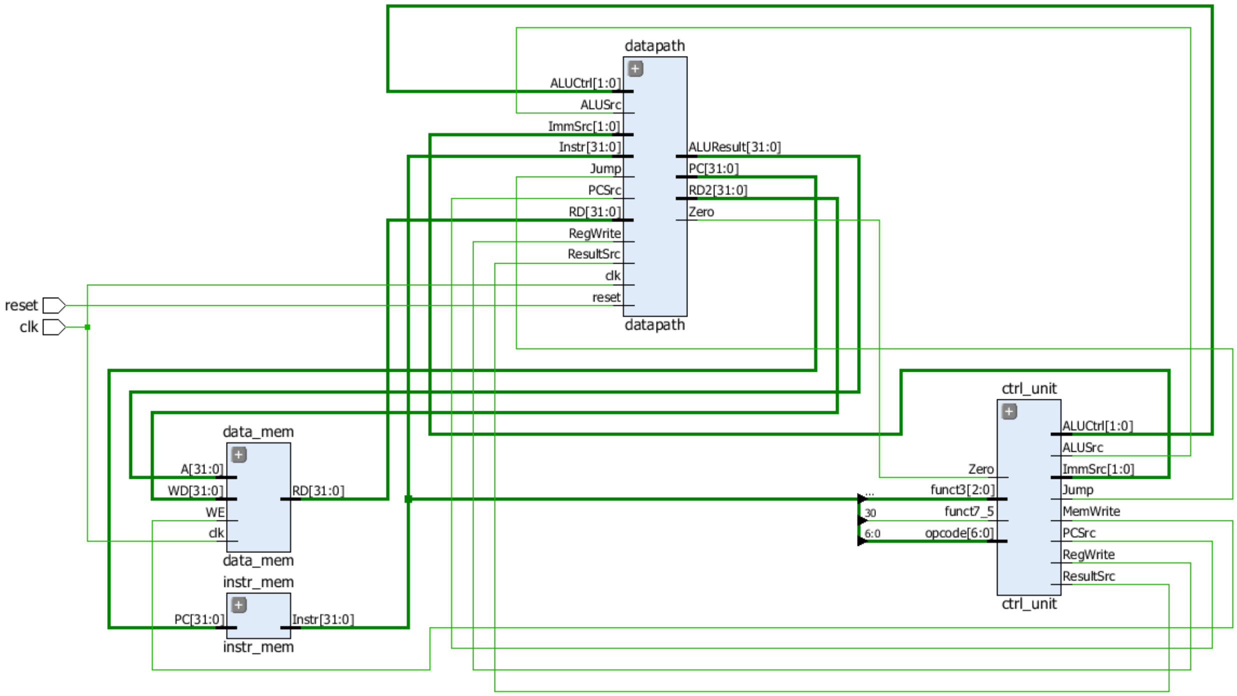
```verilog
module datapath (
    clk, reset, RegWrite, ALUSrc, PCSrc, ResultSrc, Jump,
    Instr, ALUCtrl, ImmSrc, RD, PC, ALUResult, RD2, Zero
);
    input clk, reset, RegWrite, ALUSrc, PCSrc, ResultSrc, Jump;
    input [31:0] Instr;
    input [1:0] ALUCtrl, ImmSrc;
    input [31:0] RD;
    output wire [31:0] PC;
    output wire [31:0] ALUResult;
    output wire [31:0] RD2;
    output wire Zero;
    /////////////////////////////////////////////
    wire [31:0] RD1, WD3, PCTarget, PCPlus4, PCNext, ImmExt, TmpResult, TmpPC;
    wire [31:0] SrcB;
    //Instantiate building blocks
    add_by_4 add_by_4 (.PC(PC), .PCPlus4(PCPlus4));
    ////////////////////////////////////////////////////////////////////////////
    add_to_ImmExt add_to_ImmExt (PC, ImmExt, TmpPC);
    ////////////////////////////////////////////////////////////////////////////
    PC pc (.clk(clk), .reset(reset), .PC(PC), .PCNext(PCNext));
    ////////////////////////////////////////////////////////////////////////////
    reg_file reg_file (.A1(Instr[19:15]), .A2(Instr[24:20]), .A3(Instr[11:7]),
                        .WE3(RegWrite), .RD1(RD1), .RD2(RD2), .WD3(WD3), .clk(clk));
    ////////////////////////////////////////////////////////////////////////////
    imm_ext imm_ext (Instr[31:7], ImmSrc, ImmExt);
    ////////////////////////////////////////////////////////////////////////////
    mux_2 i1 (RD2, ImmExt, ALUSrc, SrcB);
    ////////////////////////////////////////////////////////////////////////////
    alu alu (.A(RD1), .B(SrcB), .ALUCtrl(ALUCtrl), .ALUResult(ALUResult), .Zero(Zero));
    ////////////////////////////////////////////////////////////////////////////
    mux_2 i2 (ALUResult, RD, ResultSrc, TmpResult);
    ////////////////////////////////////////////////////////////////////////////
    mux_2 i3 (TmpResult, PCPlus4, Jump, WD3);
    ////////////////////////////////////////////////////////////////////////////
    mux_2 i4 (TmpPC, ALUResult, (Jump & ALUSrc), PCTarget);
    ////////////////////////////////////////////////////////////////////////////
    mux_2 i5 (PCPlus4, PCTarget, PCSrc, PCNext);
    ////////////////////////////////////////////////////////////////////////////

endmodule
```



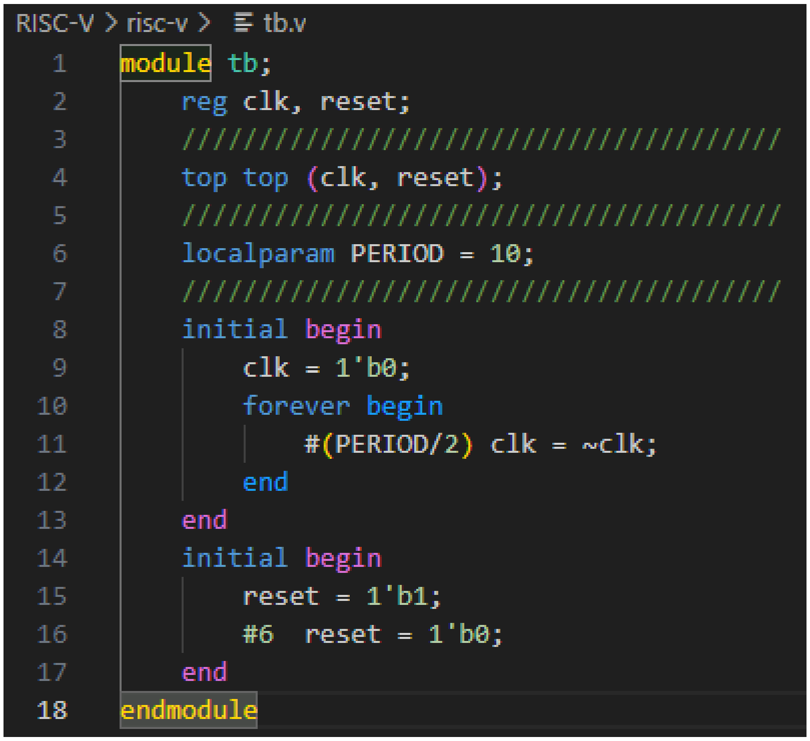
14)top:



```
module top (
    clk, reset
);
    input clk, reset;
    //////////////////////////////////////////////////////////////////////////////////////////////////////////
    wire [31:0] PC, Instr, ALUResult, RD, WD;
    wire RegWrite, Zero, MemWrite, ResultSrc, ALUSrc, PCSrc, Jump;
    wire [1:0] ImmSrc, ALUCtrl;
    reg [8*4 : 1] op_literal;
    //////////////////////////////////////////////////////////////////////////////////////////////////////////
    datapath datapath (clk, reset, RegWrite, ALUSrc, PCSrc, ResultSrc, Jump, Instr, ALUCtrl, ImmSrc, RD, PC, ALUResult, WD, Zero);
    //////////////////////////////////////////////////////////////////////////////////////////////////////////
    ctrl_unit ctrl_unit (Instr[6:0], Instr[14:12], Instr[30], Zero, ImmSrc, ALUCtrl, RegWrite, ALUSrc, MemWrite, ResultSrc, PCSrc, Jump);
    //////////////////////////////////////////////////////////////////////////////////////////////////////////
    instr_mem instr_mem (PC, Instr);
    //////////////////////////////////////////////////////////////////////////////////////////////////////////
    data_mem data_mem (clk, MemWrite, WD, ALUResult, RD);
    //////////////////////////////////////////////////////////////////////////////////////////////////////////
endmodule
```



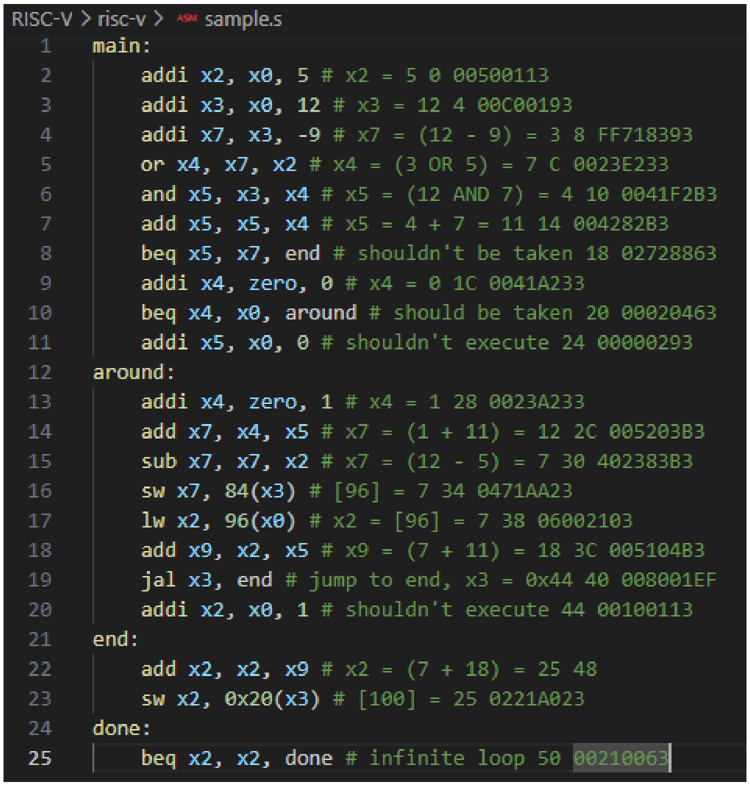
15)tb:



```
module tb;
    reg clk, reset;
    ////////////////////////////////////////
    top top (clk, reset);
    ////////////////////////////////////////
    localparam PERIOD = 10;
    ////////////////////////////////////////
    initial begin
        clk = 1'b0;
        forever begin
            #(PERIOD/2) clk = ~clk;
        end
    end
    initial begin
        reset = 1'b1;
        #6  reset = 1'b0;
    end
endmodule
```

Testbench:

Program (1):

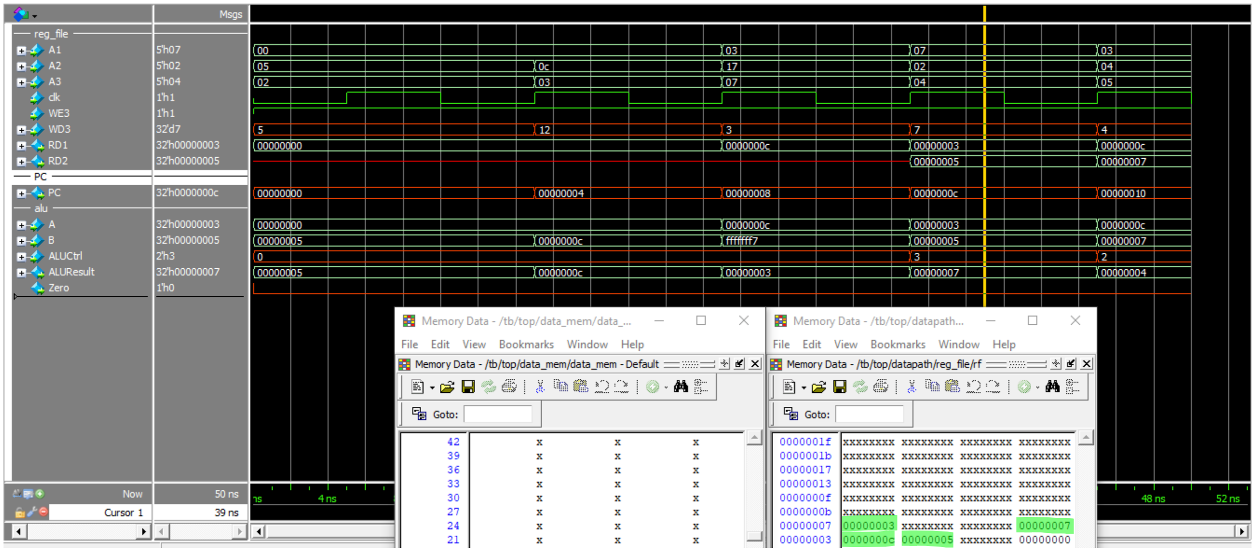


RISC-V > risc-v > sample.s

```
main:
    addi x2, x0, 5 # x2 = 5 0 00500113
    addi x3, x0, 12 # x3 = 12 4 00C00193
    addi x7, x3, -9 # x7 = (12 - 9) = 3 8 FF718393
    or x4, x7, x2 # x4 = (3 OR 5) = 7 C 0023E233
    and x5, x3, x4 # x5 = (12 AND 7) = 4 10 0041F2B3
    add x5, x5, x4 # x5 = 4 + 7 = 11 14 004282B3
    beq x5, x7, end # shouldn't be taken 18 02728863
    addi x4, zero, 0 # x4 = 0 1C 0041A233
    beq x4, x0, around # should be taken 20 00020463
    addi x5, x0, 0 # shouldn't execute 24 00000293
around:
    addi x4, zero, 1 # x4 = 1 28 0023A233
    add x7, x4, x5 # x7 = (1 + 11) = 12 2C 005203B3
    sub x7, x7, x2 # x7 = (12 - 5) = 7 30 402383B3
    sw x7, 84(x3) # [96] = 7 34 0471AA23
    lw x2, 96(x0) # x2 = [96] = 7 38 06002103
    add x9, x2, x5 # x9 = (7 + 11) = 18 3C 005104B3
    jal x3, end # jump to end, x3 = 0x44 40 008001EF
    addi x2, x0, 1 # shouldn't execute 44 00100113
end:
    add x2, x2, x9 # x2 = (7 + 18) = 25 48
    sw x2, 0x20(x3) # [100] = 25 0221A023
done:
    beq x2, x2, done # infinite loop 50 00210063
```

Wavefroms:

Note: In the figures below, the waveform shows some internal signals of interest accompanied with "reg_file" and "data_mem" memories to facilitate observing changes to them. I tried to show up the intermediate changes through several snapshots of the simulation. Moreover, changes in both memories are highlighted so that it becomes hopefully easy to trace.



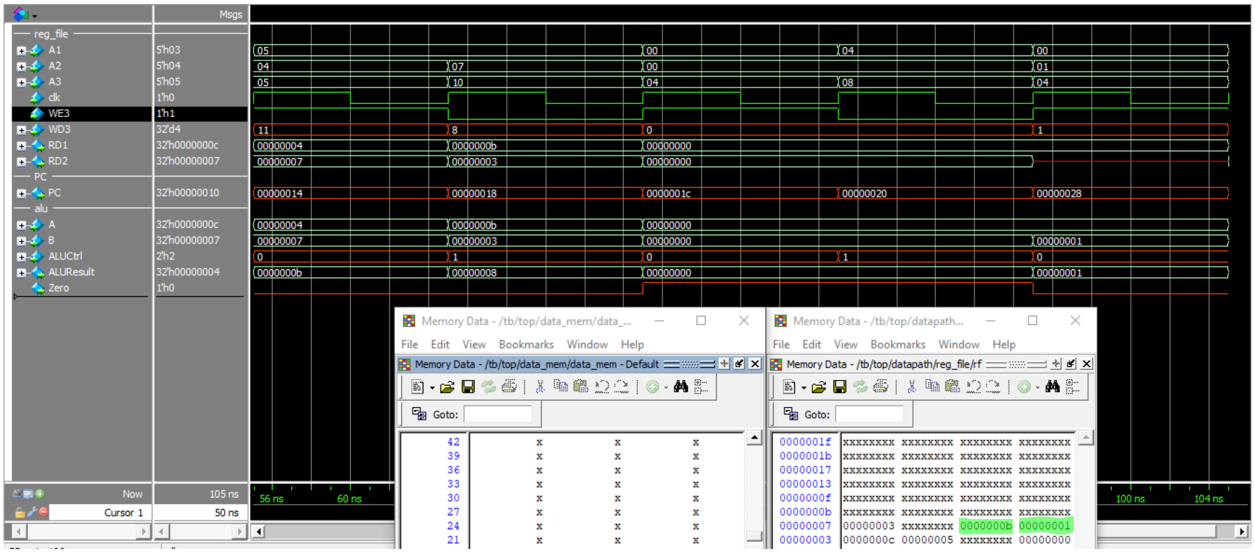
```
Msgs
reg_file
A1        5'h07
A2        5'h02
A3        5'h04
clk       1'h1
WE3       1'h1
WD3       32'd7
RD1       32'h00000003
RD2       32'h00000005
PC
PC        32'h0000000c
alu
A         32'h00000003
B         32'h00000005
ALUCtrl   2'h3
ALUResult 32'h00000007
Zero      1'h0
Now       50 ns
Cursor 1  39 ns
```

Memory Data - /tb/top/data_mem/data_mem - Default

```
42    x    x    x
39    x    x    x
36    x    x    x
33    x    x    x
30    x    x    x
27    x    x    x
24    x    x    x
21    x    x    x
```

Memory Data - /tb/top/datapath/reg_file/rf

```
0000001f xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000001b xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000017 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000013 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000000f xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000000b xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000007 00000003 xxxxxxxx xxxxxxxx 00000007
00000003 0000000c 00000005 xxxxxxxx 00000000
```



```
reg_file
A1          5'h03
A2          5'h04
A3          5'h05
clk         1'h0
WE3         1'h1
WD3         32'd4
RD1         32'h0000000c
RD2         32'h00000007
PC
PC          32'h00000010
alu
A           32'h0000000c
B           32'h00000007
ALUCtrl     2'h2
ALUResult   32'h00000004
Zero        1'h0
Now         105 ns
Cursor 1    50 ns
```

Msgs
05 | 00 | 04 | 00
04 | 07 | 00 | 01
05 | 10 | 04 | 08 | 04
11 | 8 | 0 | 1
00000004 | 0000000b | 00000000
00000007 | 00000003 | 00000000
00000014 | 00000018 | 0000001c | 00000020 | 00000028
00000004 | 0000000b | 00000000
00000007 | 00000003 | 00000000 | 00000001
0 | 1 | 0 | 1 | 0
0000000b | 00000008 | 00000000 | 00000001
56 ns
60 ns
100 ns
104 ns

Memory Data - /tb/top/data_mem/data_...

File Edit View Bookmarks Window Help

Memory Data - /tb/top/data_mem/data_mem - Default

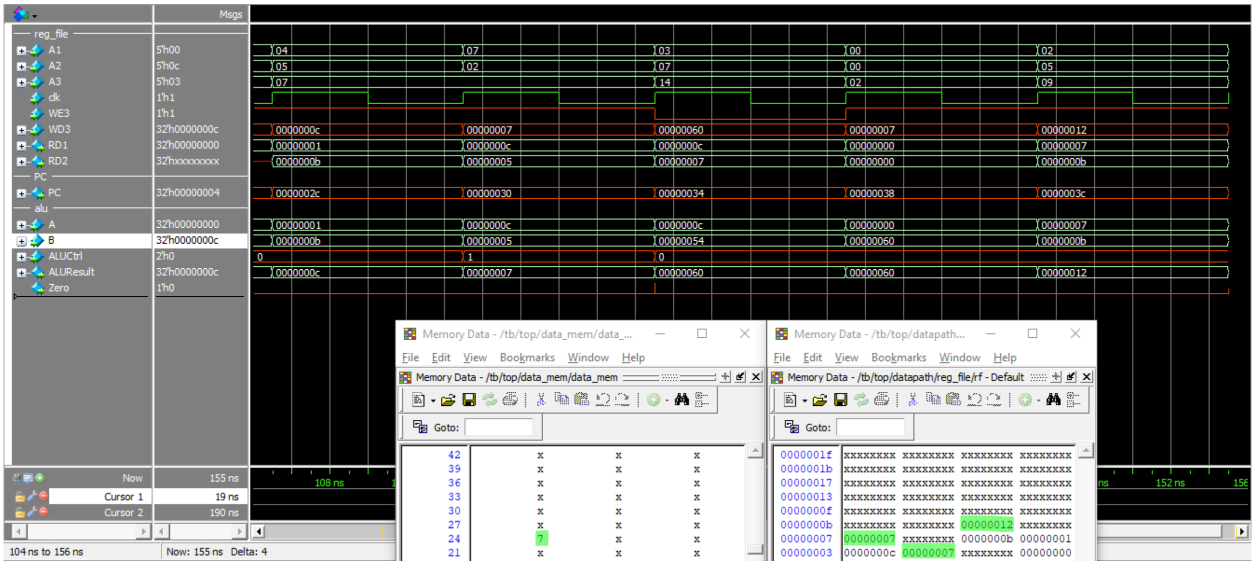
```
42   x   x   x
39   x   x   x
36   x   x   x
33   x   x   x
30   x   x   x
27   x   x   x
24   x   x   x
21   x   x   x
```

Memory Data - /tb/top/datapath...

File Edit View Bookmarks Window Help

Memory Data - /tb/top/datapath/reg_file/rf

```
0000001f  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000001b  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000017  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000013  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000000f  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000000b  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000007  00000003 xxxxxxxx 0000000b 00000001
00000003  0000000c 00000005 xxxxxxxx 00000000
```



```
reg_file
A1          5'h00
A2          5'h0c
A3          5'h03
clk         1'h1
WE3         1'h1
WD3         32'h0000000c
RD1         32'h00000000
RD2         32'hxxxxxxxx
PC
PC          32'h00000004
alu
A           32'h00000000
B           32'h0000000c
ALUCtrl     2'h0
ALUResult   32'h0000000c
Zero        1'h0
Now         155 ns
Cursor 1    19 ns
Cursor 2    190 ns
```

Msgs
04 | 07 | 03 | 00 | 02
05 | 02 | 07 | 00 | 05
07 | 14 | 02 | 09
0000000c | 00000007 | 00000060 | 00000007 | 00000012
00000001 | 0000000c | 0000000c | 00000000 | 00000007
0000000b | 00000005 | 00000007 | 00000000 | 0000000b
0000002c | 00000030 | 00000034 | 00000038 | 0000003c
00000001 | 0000000c | 0000000c | 00000000 | 00000007
0000000b | 00000005 | 00000054 | 00000060 | 0000000b
0 | 1 | 0
0000000c | 00000007 | 00000060 | 00000060 | 00000012
108 ns
152 ns

104 ns to 156 ns   Now: 155 ns Delta: 4

Memory Data - /tb/top/data_mem/data_...

File Edit View Bookmarks Window Help

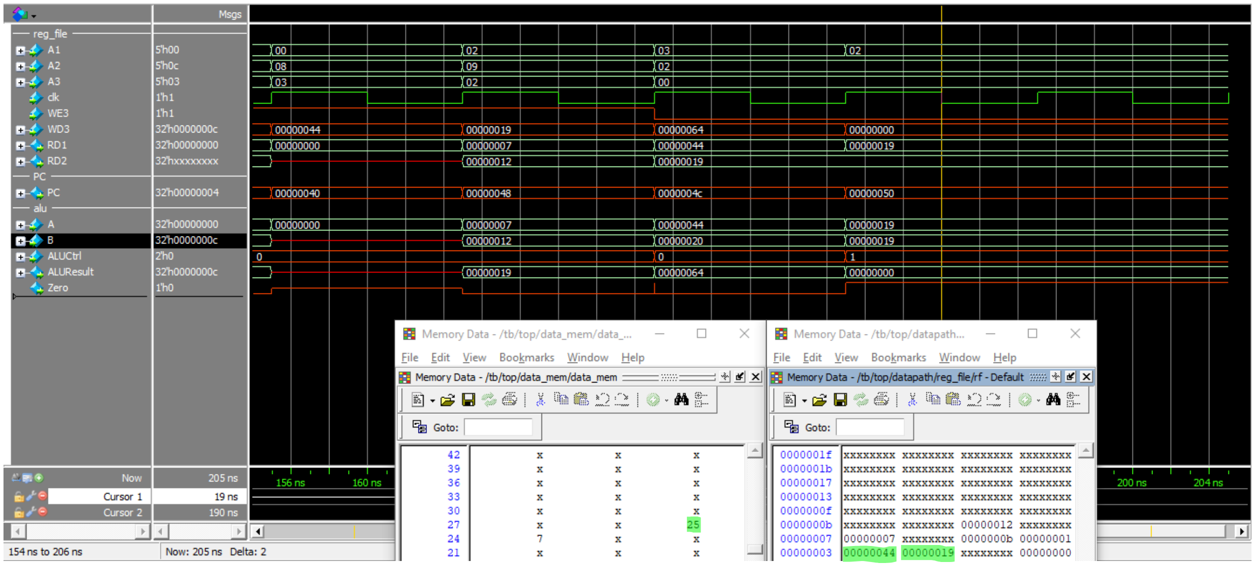
Memory Data - /tb/top/data_mem/data_mem

```
42   x   x   x
39   x   x   x
36   x   x   x
33   x   x   x
30   x   x   x
27   x   x   x
24   7   x   x
21   x   x   x
```

Memory Data - /tb/top/datapath...

Memory Data - /tb/top/datapath/reg_file/rf - Default

```
0000001f  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000001b  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000017  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000013  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000000f  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000000b  xxxxxxxx xxxxxxxx 00000012 xxxxxxxx
00000007  00000007 xxxxxxxx 0000000b 00000001
00000003  0000000c 00000007 xxxxxxxx 00000000
```



	Msgs
reg_file	
A1	5'h00
A2	5'h0c
A3	5'h03
clk	1'h1
WE3	1'h1
WD3	32'h0000000c
RD1	32'h00000000
RD2	32'hxxxxxxxx
PC	
PC	32'h00000004
alu	
A	32'h00000000
B	32'h0000000c
ALUCtrl	2'h0
ALUResult	32'h0000000c
Zero	1'h0

Now	205 ns
Cursor 1	19 ns
Cursor 2	190 ns

154 ns to 206 ns

Now: 205 ns Delta: 2

Memory Data - /tb/top/data_mem/data_mem

```
42    x    x    x
39    x    x    x
36    x    x    x
33    x    x    x
30    x    x    x
27    x    x    25
24    7    x    x
21    x    x    x
```

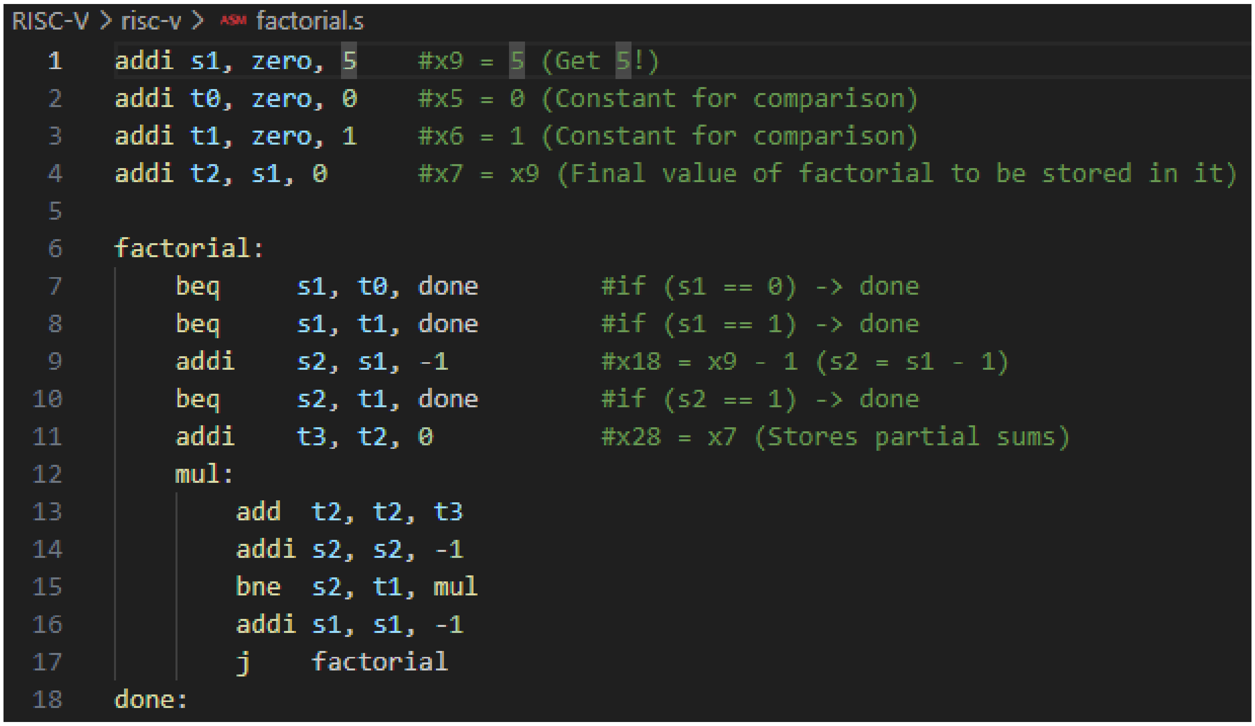
Memory Data - /tb/top/datapath...
File Edit View Bookmarks Window Help
Goto:

Memory Data - /tb/top/datapath/reg_file/rf - Default

```
0000001f xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000001b xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000017 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000013 xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000000f xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000000b xxxxxxxx xxxxxxxx 00000012 xxxxxxxx
00000007 00000007 xxxxxxxx 0000000b 00000001
00000003 00000044 00000019 xxxxxxxx 00000000
```

Program (2):

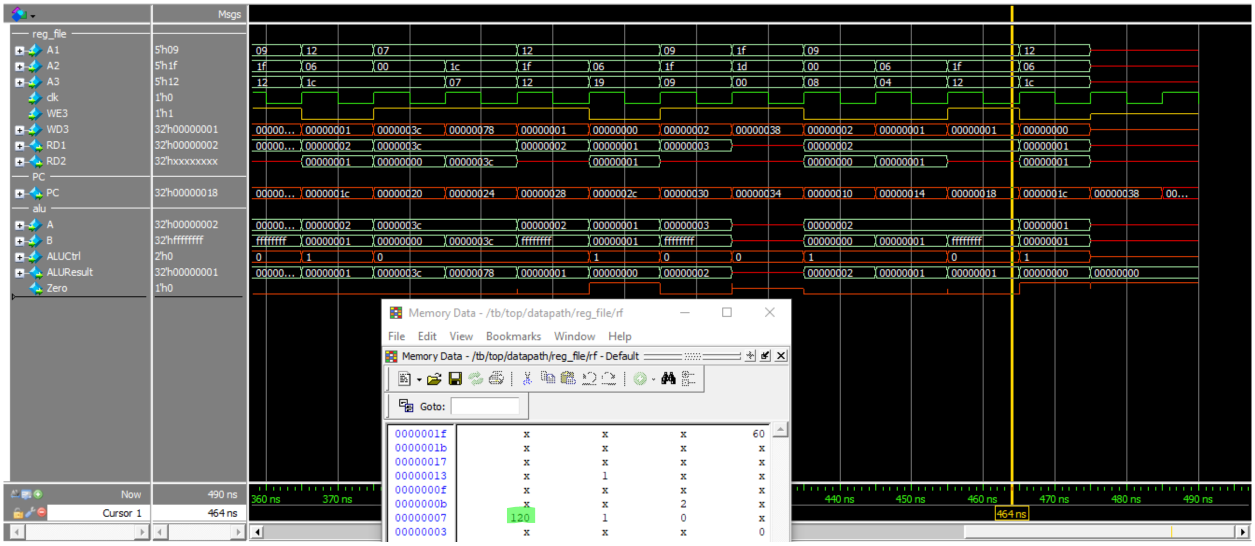
This is a simple program that I wrote to calculate the factorial of a given number. Following is the source assembly code followed by simulation snapshots.



RISC-V > risc-v > factorial.s

```
addi s1, zero, 5     #x9 = 5 (Get 5!)
addi t0, zero, 0     #x5 = 0 (Constant for comparison)
addi t1, zero, 1     #x6 = 1 (Constant for comparison)
addi t2, s1, 0       #x7 = x9 (Final value of factorial to be stored in it)

factorial:
    beq     s1, t0, done         #if (s1 == 0) -> done
    beq     s1, t1, done         #if (s1 == 1) -> done
    addi    s2, s1, -1           #x18 = x9 - 1 (s2 = s1 - 1)
    beq     s2, t1, done         #if (s2 == 1) -> done
    addi    t3, t2, 0            #x28 = x7 (Stores partial sums)
    mul:
        add  t2, t2, t3
        addi s2, s2, -1
        bne  s2, t1, mul
        addi s1, s1, -1
        j    factorial
done:
```

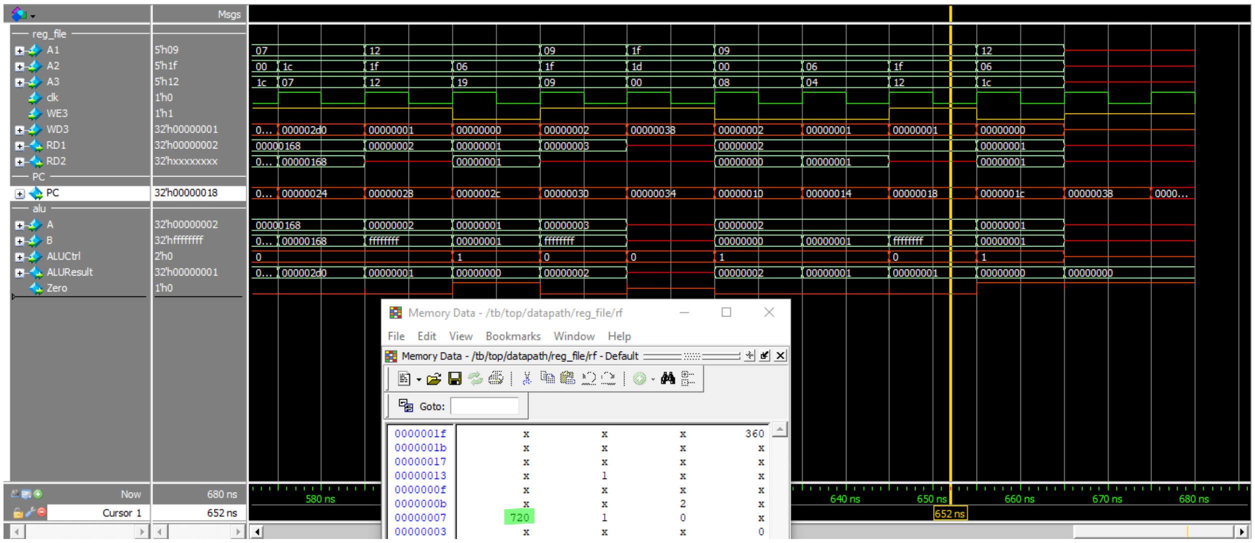
Example (1): Factorial of 5!


Msgs
Memory Data - /tb/top/datapath/reg_file/rf
File Edit View Bookmarks Window Help
Memory Data - /tb/top/datapath/reg_file/rf - Default
Goto:
360 ns
370 ns
440 ns
450 ns
460 ns
470 ns
480 ns
490 ns
464 ns

```
reg_file
A1        5'h09
A2        5'h1f
A3        5'h12
clk       1'h0
WE3       1'h1
WD3       32'h00000001
RD1       32'h00000002
RD2       32'hxxxxxxxx
PC
PC        32'h00000018
alu
A         32'h00000002
B         32'hffffffff
ALUCtrl   2'h0
ALUResult 32'h00000001
Zero      1'h0
Now       490 ns
Cursor 1  464 ns
```

```
0000001f      x      x      x     60
0000001b      x      x      x      x
00000017      x      x      x      x
00000013      x      1      x      x
0000000f      x      x      x      x
0000000b      x      x      2      x
00000007    120      1      0      x
00000003      x      x      x      0
```

Example (2): Factorial of 6!


Msgs
Memory Data - /tb/top/datapath/reg_file/rf
File Edit View Bookmarks Window Help
Memory Data - /tb/top/datapath/reg_file/rf - Default
Goto:
580 ns
640 ns
650 ns
660 ns
670 ns
680 ns
652 ns

```
reg_file
A1        5'h09
A2        5'h1f
A3        5'h12
clk       1'h0
WE3       1'h1
WD3       32'h00000001
RD1       32'h00000002
RD2       32'hxxxxxxxx
PC
PC        32'h00000018
alu
A         32'h00000002
B         32'hffffffff
ALUCtrl   2'h0
ALUResult 32'h00000001
Zero      1'h0
Now       680 ns
Cursor 1  652 ns
```

```
0000001f      x      x      x    360
0000001b      x      x      x      x
00000017      x      x      x      x
00000013      x      1      x      x
0000000f      x      x      x      x
0000000b      x      x      2      x
00000007    720      1      0      x
00000003      x      x      x      0
```