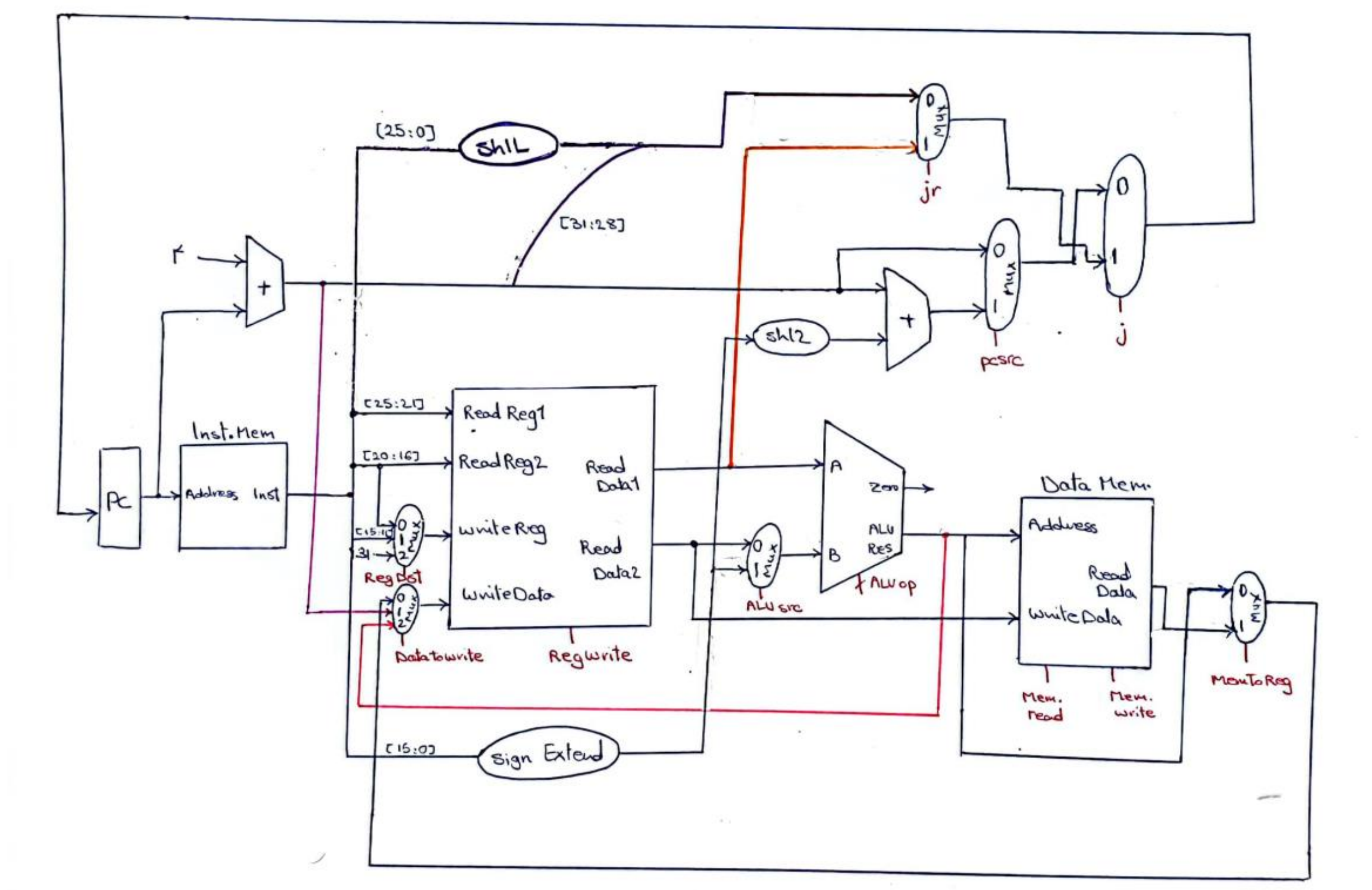
Computer Assignment2

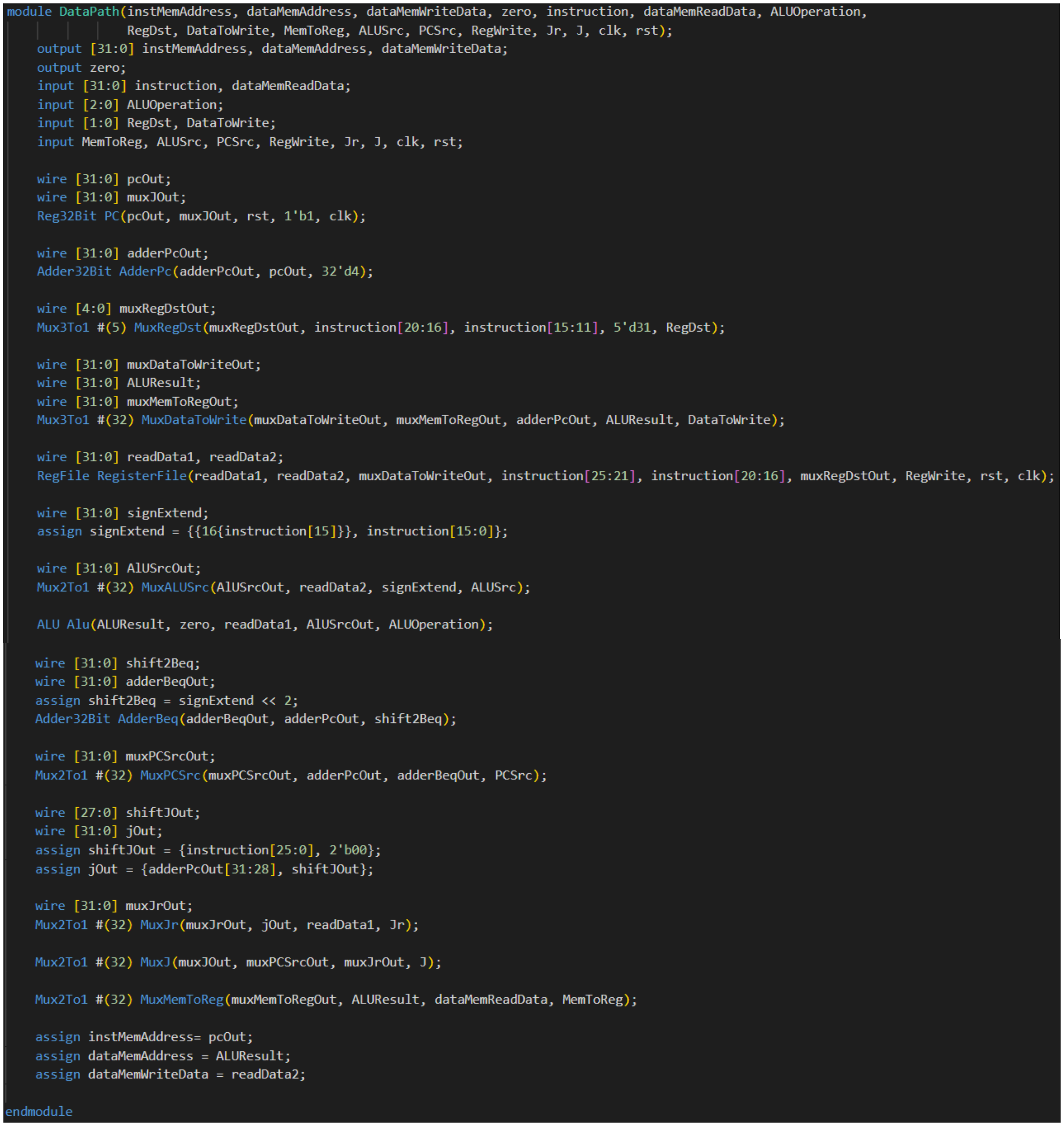
“Single Cycle MIPS Processor”

Ava Mirmohamadmahdi 810199501

Nesa Abbasimoghadam 810199457

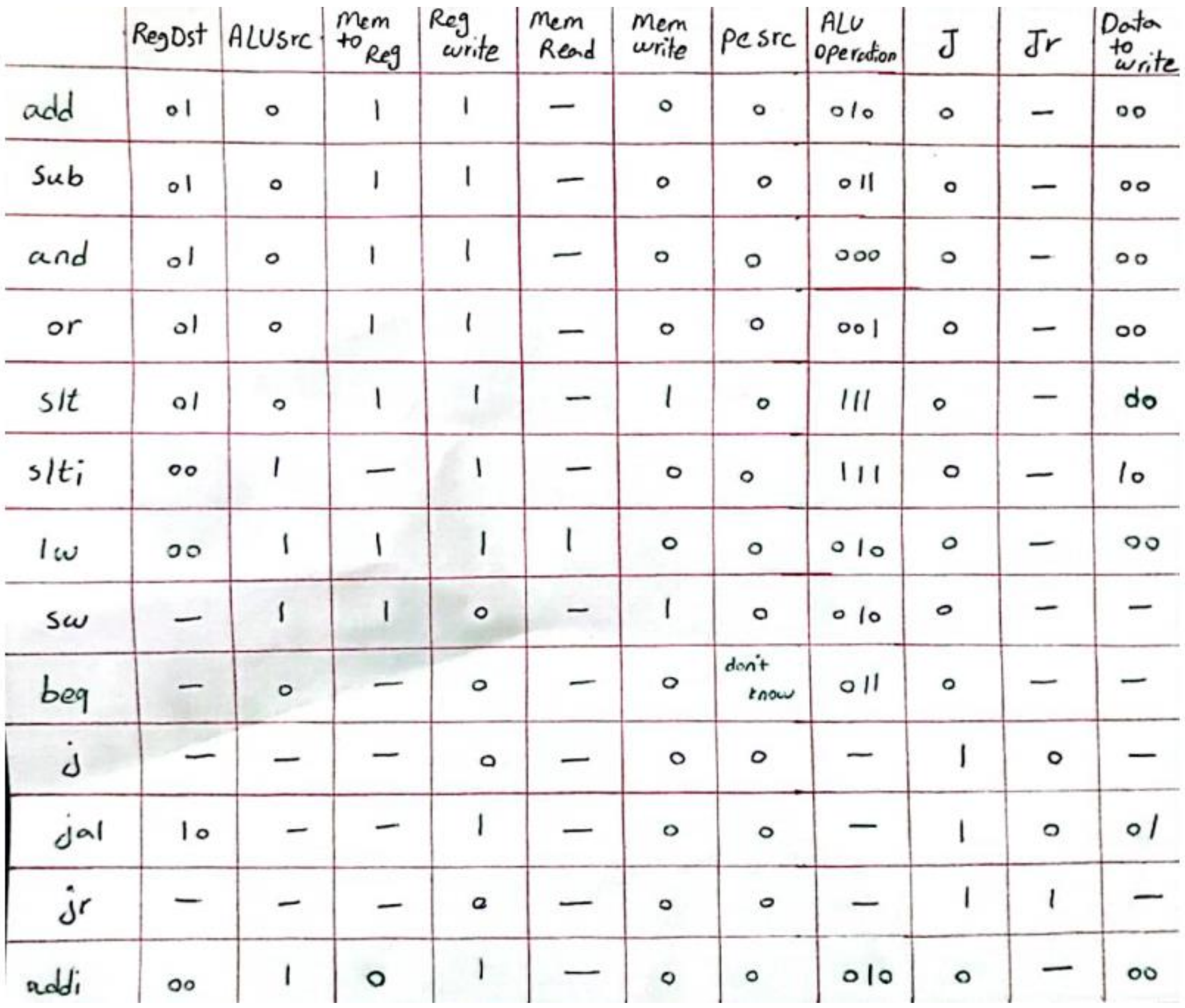
DataPath:



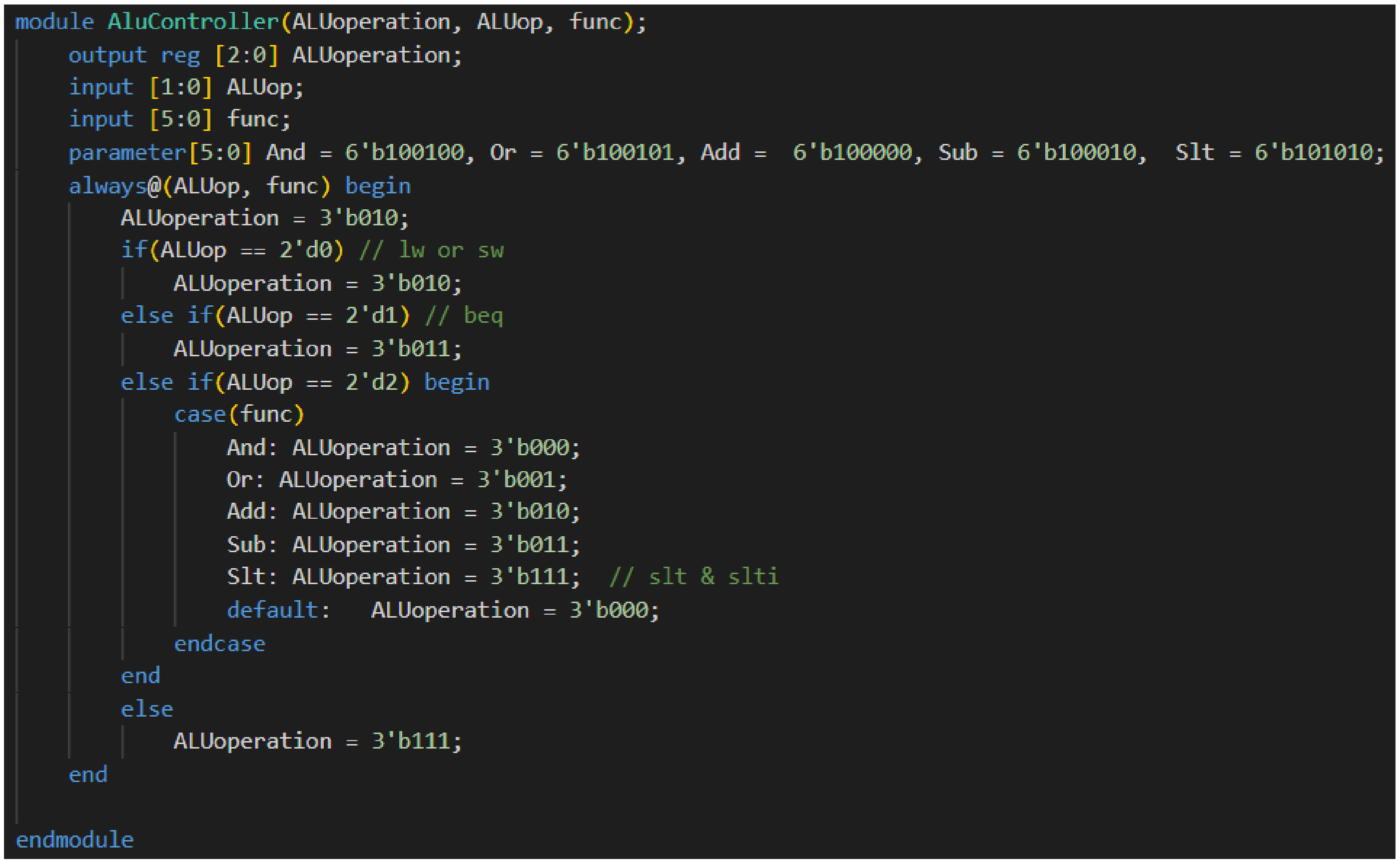


```verilog
module DataPath(instMemAddress, dataMemAddress, dataMemWriteData, zero, instruction, dataMemReadData, ALUOperation,
                RegDst, DataToWrite, MemToReg, ALUSrc, PCSrc, RegWrite, Jr, J, clk, rst);
    output [31:0] instMemAddress, dataMemAddress, dataMemWriteData;
    output zero;
    input [31:0] instruction, dataMemReadData;
    input [2:0] ALUOperation;
    input [1:0] RegDst, DataToWrite;
    input MemToReg, ALUSrc, PCSrc, RegWrite, Jr, J, clk, rst;

    wire [31:0] pcOut;
    wire [31:0] muxJOut;
    Reg32Bit PC(pcOut, muxJOut, rst, 1'b1, clk);

    wire [31:0] adderPcOut;
    Adder32Bit AdderPc(adderPcOut, pcOut, 32'd4);

    wire [4:0] muxRegDstOut;
    Mux3To1 #(5) MuxRegDst(muxRegDstOut, instruction[20:16], instruction[15:11], 5'd31, RegDst);

    wire [31:0] muxDataToWriteOut;
    wire [31:0] ALUResult;
    wire [31:0] muxMemToRegOut;
    Mux3To1 #(32) MuxDataToWrite(muxDataToWriteOut, muxMemToRegOut, adderPcOut, ALUResult, DataToWrite);

    wire [31:0] readData1, readData2;
    RegFile RegisterFile(readData1, readData2, muxDataToWriteOut, instruction[25:21], instruction[20:16], muxRegDstOut, RegWrite, rst, clk);

    wire [31:0] signExtend;
    assign signExtend = {{16{instruction[15]}}, instruction[15:0]};

    wire [31:0] AlUSrcOut;
    Mux2To1 #(32) MuxALUSrc(AlUSrcOut, readData2, signExtend, ALUSrc);

    ALU Alu(ALUResult, zero, readData1, AlUSrcOut, ALUOperation);

    wire [31:0] shift2Beq;
    wire [31:0] adderBeqOut;
    assign shift2Beq = signExtend << 2;
    Adder32Bit AdderBeq(adderBeqOut, adderPcOut, shift2Beq);

    wire [31:0] muxPCSrcOut;
    Mux2To1 #(32) MuxPCSrc(muxPCSrcOut, adderPcOut, adderBeqOut, PCSrc);

    wire [27:0] shiftJOut;
    wire [31:0] jOut;
    assign shiftJOut = {instruction[25:0], 2'b00};
    assign jOut = {adderPcOut[31:28], shiftJOut};

    wire [31:0] muxJrOut;
    Mux2To1 #(32) MuxJr(muxJrOut, jOut, readData1, Jr);

    Mux2To1 #(32) MuxJ(muxJOut, muxPCSrcOut, muxJrOut, J);

    Mux2To1 #(32) MuxMemToReg(muxMemToRegOut, ALUResult, dataMemReadData, MemToReg);

    assign instMemAddress= pcOut;
    assign dataMemAddress = ALUResult;
    assign dataMemWriteData = readData2;

endmodule
```

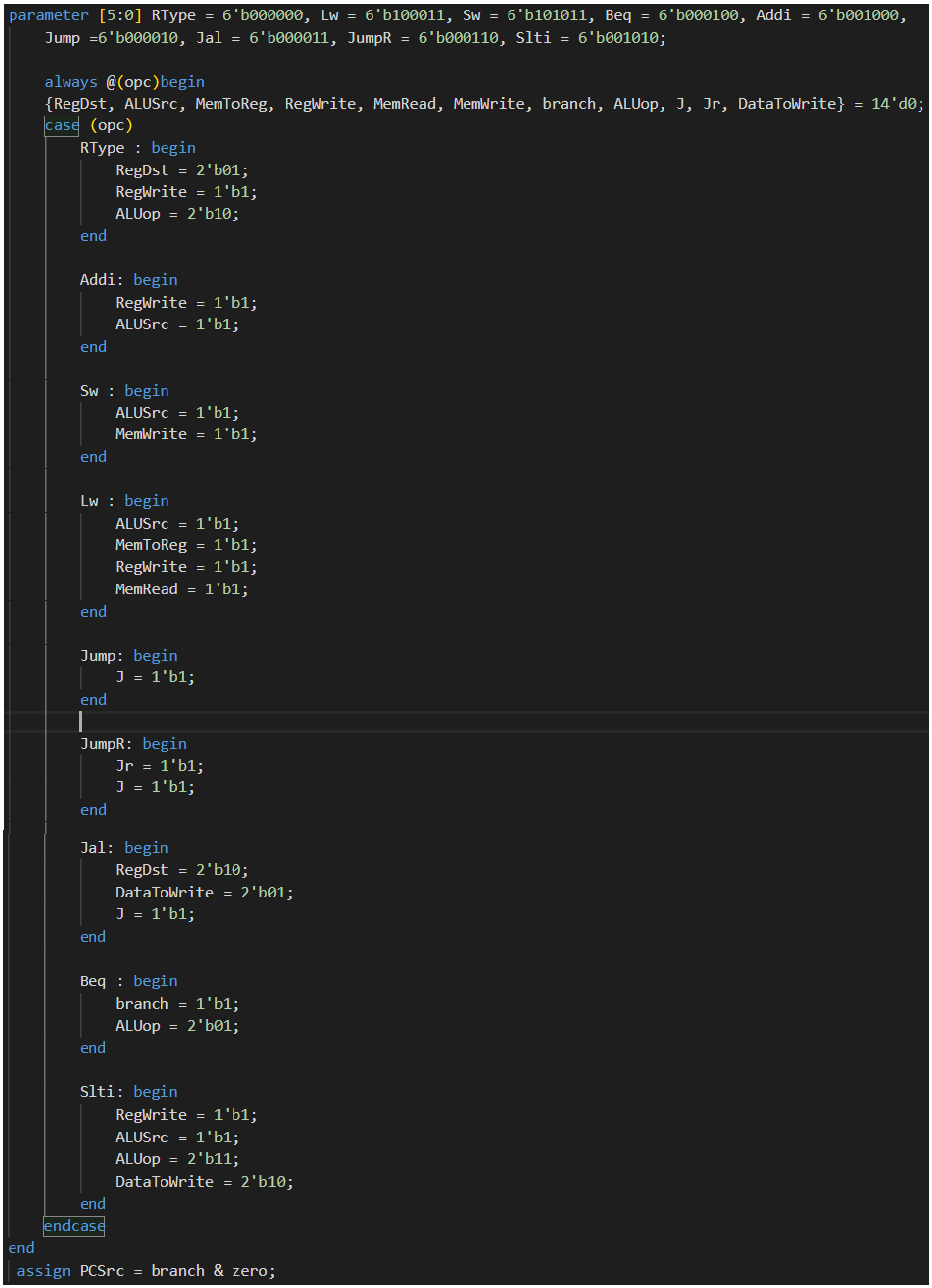
Controller:



	RegDst	ALUsrc	Mem to Reg	Reg write	Mem Read	Mem write	Pc src	ALU Operation	J	Jr	Data to write
add	01	0	1	1	—	0	0	010	0	—	00
sub	01	0	1	1	—	0	0	011	0	—	00
and	01	0	1	1	—	0	0	000	0	—	00
or	01	0	1	1	—	0	0	001	0	—	00
slt	01	0	1	1	—	1	0	111	0	—	do
slti	00	1	—	1	—	0	0	111	0	—	10
lw	00	1	1	1	1	0	0	010	0	—	00
sw	—	1	1	0	—	1	0	010	0	—	—
beq	—	0	—	0	—	0	don't know	011	0	—	—
j	—	—	—	0	—	0	0	—	1	0	—
jal	10	—	—	1	—	0	0	—	1	0	01
jr	—	—	—	0	—	0	0	—	1	1	—
addi	00	1	0	1	—	0	0	010	0	—	00

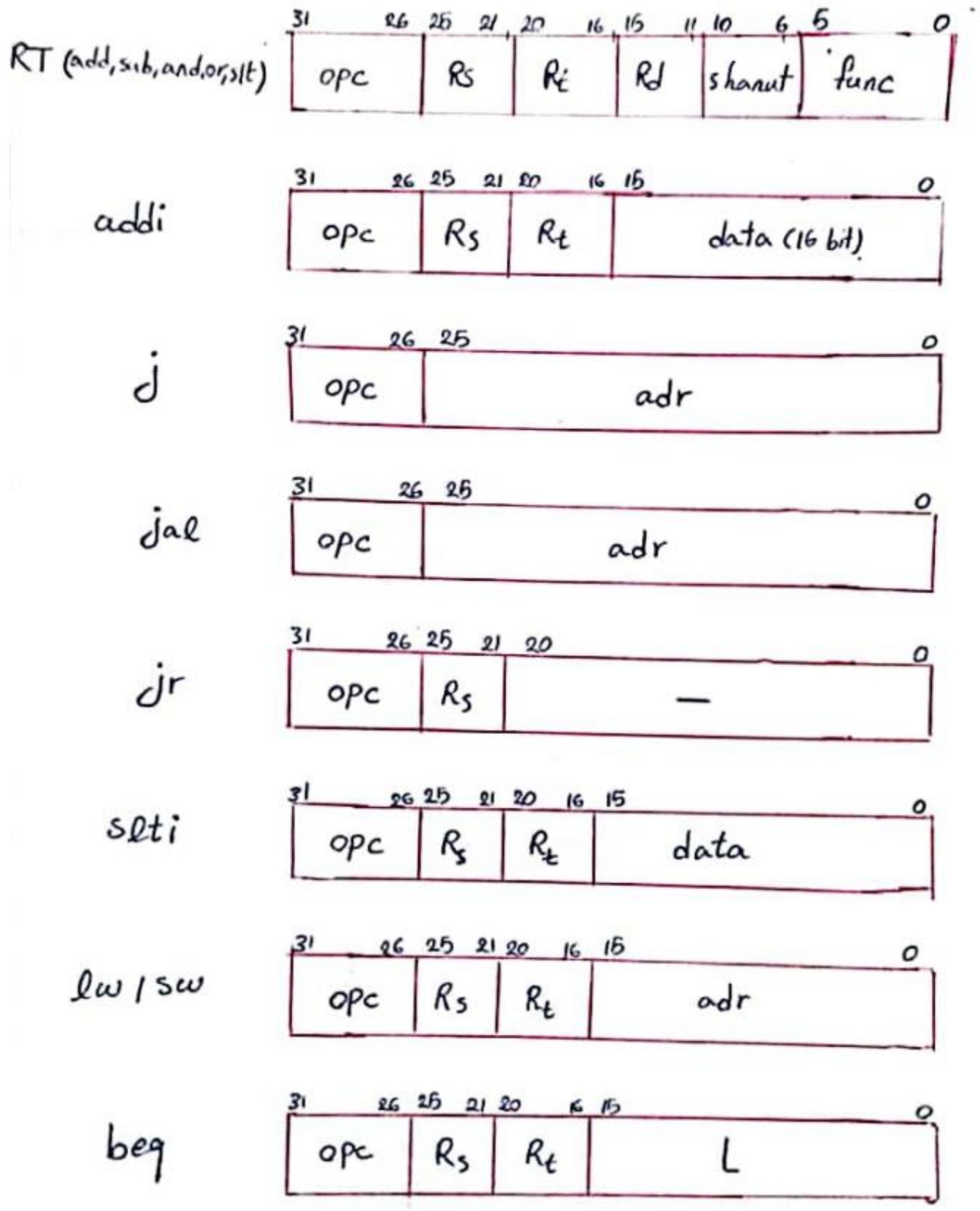


```verilog
module AluController(ALUoperation, ALUop, func);
    output reg [2:0] ALUoperation;
    input [1:0] ALUop;
    input [5:0] func;
    parameter[5:0] And = 6'b100100, Or = 6'b100101, Add =  6'b100000, Sub = 6'b100010,  Slt = 6'b101010;
    always@(ALUop, func) begin
        ALUoperation = 3'b010;
        if(ALUop == 2'd0) // lw or sw
            ALUoperation = 3'b010;
        else if(ALUop == 2'd1) // beq
            ALUoperation = 3'b011;
        else if(ALUop == 2'd2) begin
            case(func)
                And: ALUoperation = 3'b000;
                Or: ALUoperation = 3'b001;
                Add: ALUoperation = 3'b010;
                Sub: ALUoperation = 3'b011;
                Slt: ALUoperation = 3'b111;  // slt & slti
                default:   ALUoperation = 3'b000;
            endcase
        end
        else
            ALUoperation = 3'b111;
    end

endmodule
```

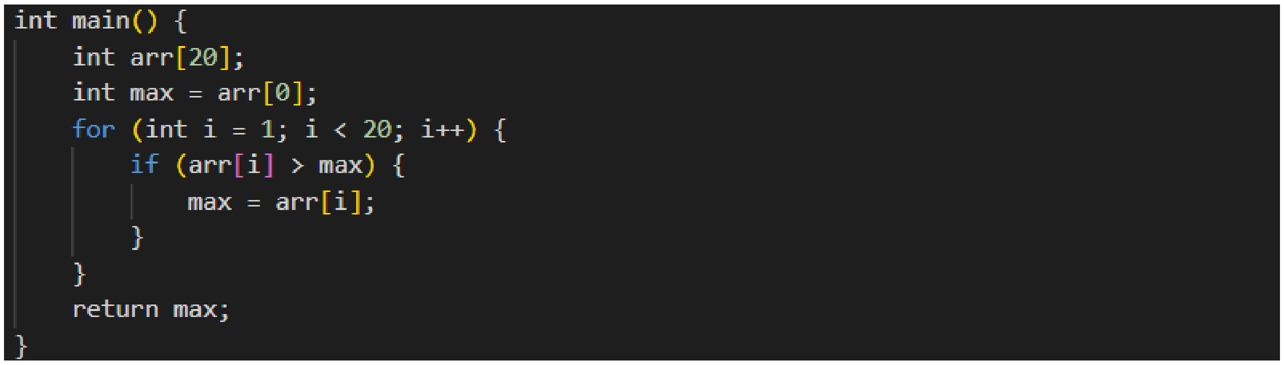


```
parameter [5:0] RType = 6'b000000, Lw = 6'b100011, Sw = 6'b101011, Beq = 6'b000100, Addi = 6'b001000,
    Jump =6'b000010, Jal = 6'b000011, JumpR = 6'b000110, Slti = 6'b001010;

    always @(opc)begin
    {RegDst, ALUSrc, MemToReg, RegWrite, MemRead, MemWrite, branch, ALUop, J, Jr, DataToWrite} = 14'd0;
    case (opc)
        RType : begin
            RegDst = 2'b01;
            RegWrite = 1'b1;
            ALUop = 2'b10;
        end

        Addi: begin
            RegWrite = 1'b1;
            ALUSrc = 1'b1;
        end

        Sw : begin
            ALUSrc = 1'b1;
            MemWrite = 1'b1;
        end

        Lw : begin
            ALUSrc = 1'b1;
            MemToReg = 1'b1;
            RegWrite = 1'b1;
            MemRead = 1'b1;
        end

        Jump: begin
            J = 1'b1;
        end

        JumpR: begin
            Jr = 1'b1;
            J = 1'b1;
        end

        Jal: begin
            RegDst = 2'b10;
            DataToWrite = 2'b01;
            J = 1'b1;
        end

        Beq : begin
            branch = 1'b1;
            ALUop = 2'b01;
        end

        Slti: begin
            RegWrite = 1'b1;
            ALUSrc = 1'b1;
            ALUop = 2'b11;
            DataToWrite = 2'b10;
        end
    endcase
end
 assign PCSrc = branch & zero;
```

Instruction Formats:

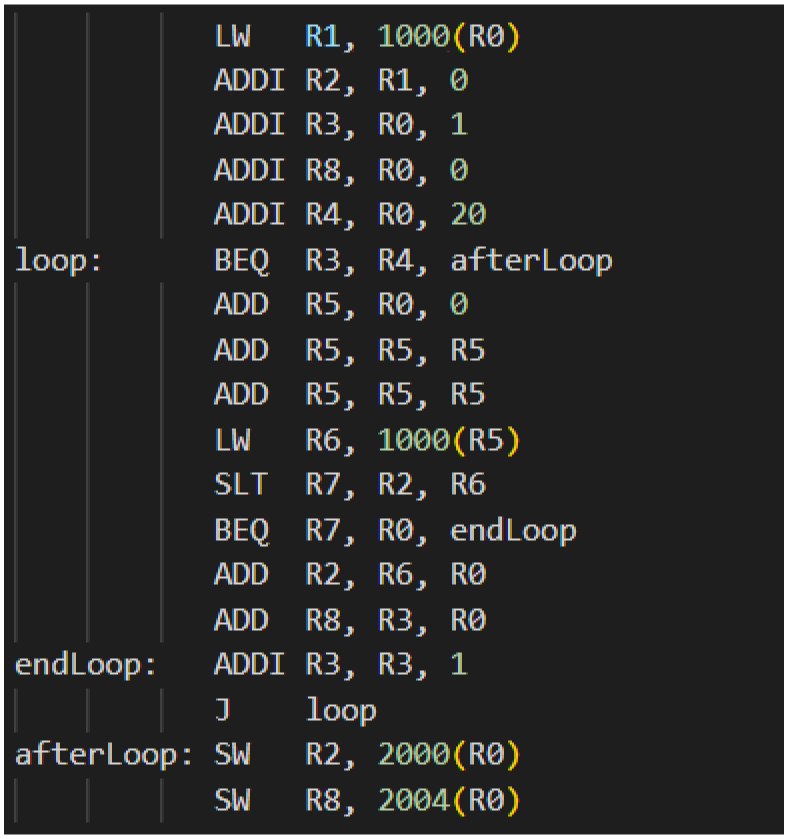


Cpp code – Find Max Numebr:



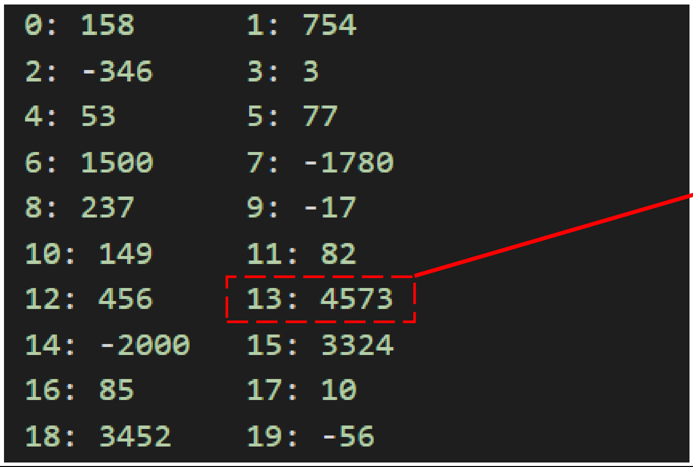
```cpp
int main() {
    int arr[20];
    int max = arr[0];
    for (int i = 1; i < 20; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}
```

Assembly Instructions:

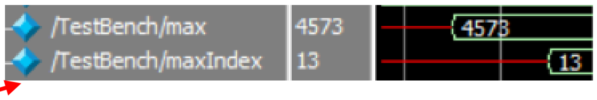


```
           LW   R1, 1000(R0)
           ADDI R2, R1, 0
           ADDI R3, R0, 1
           ADDI R8, R0, 0
           ADDI R4, R0, 20
loop:      BEQ  R3, R4, afterLoop
           ADD  R5, R0, 0
           ADD  R5, R5, R5
           ADD  R5, R5, R5
           LW   R6, 1000(R5)
           SLT  R7, R2, R6
           BEQ  R7, R0, endLoop
           ADD  R2, R6, R0
           ADD  R8, R3, R0
endLoop:   ADDI R3, R3, 1
           J    loop
afterLoop: SW   R2, 2000(R0)
           SW   R8, 2004(R0)
```

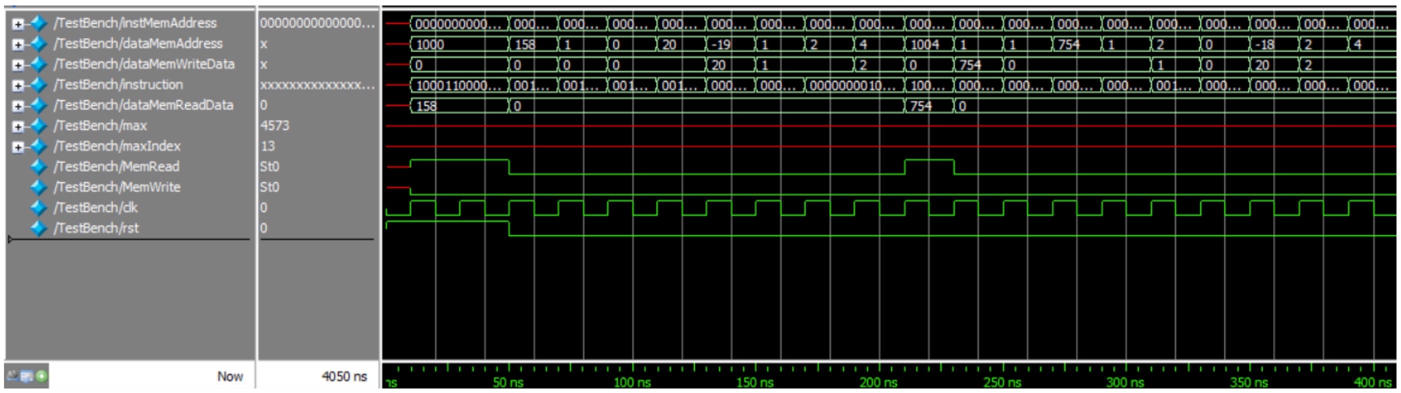
20 random numbers:

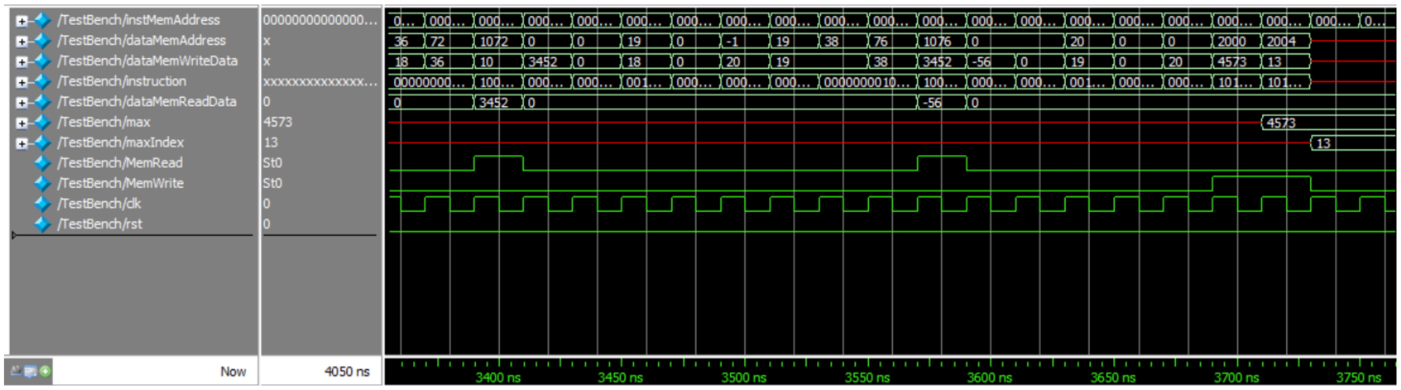


```
0: 158        1: 754
2: -346       3: 3
4: 53         5: 77
6: 1500       7: -1780
8: 237        9: -17
10: 149       11: 82
12: 456       13: 4573
14: -2000     15: 3324
16: 85        17: 10
18: 3452      19: -56
```



```
/TestBench/max       4573    4573
/TestBench/maxIndex  13      13
```


/TestBench/instMemAddress
/TestBench/dataMemAddress
/TestBench/dataMemWriteData
/TestBench/instruction
/TestBench/dataMemReadData
/TestBench/max
/TestBench/maxIndex
/TestBench/MemRead
/TestBench/MemWrite
/TestBench/clk
/TestBench/rst
Now
4050 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
400 ns


/TestBench/instMemAddress
/TestBench/dataMemAddress
/TestBench/dataMemWriteData
/TestBench/instruction
/TestBench/dataMemReadData
/TestBench/max
/TestBench/maxIndex
/TestBench/MemRead
/TestBench/MemWrite
/TestBench/clk
/TestBench/rst
Now
4050 ns
3400 ns
3450 ns
3500 ns
3550 ns
3600 ns
3650 ns
3700 ns
3750 ns