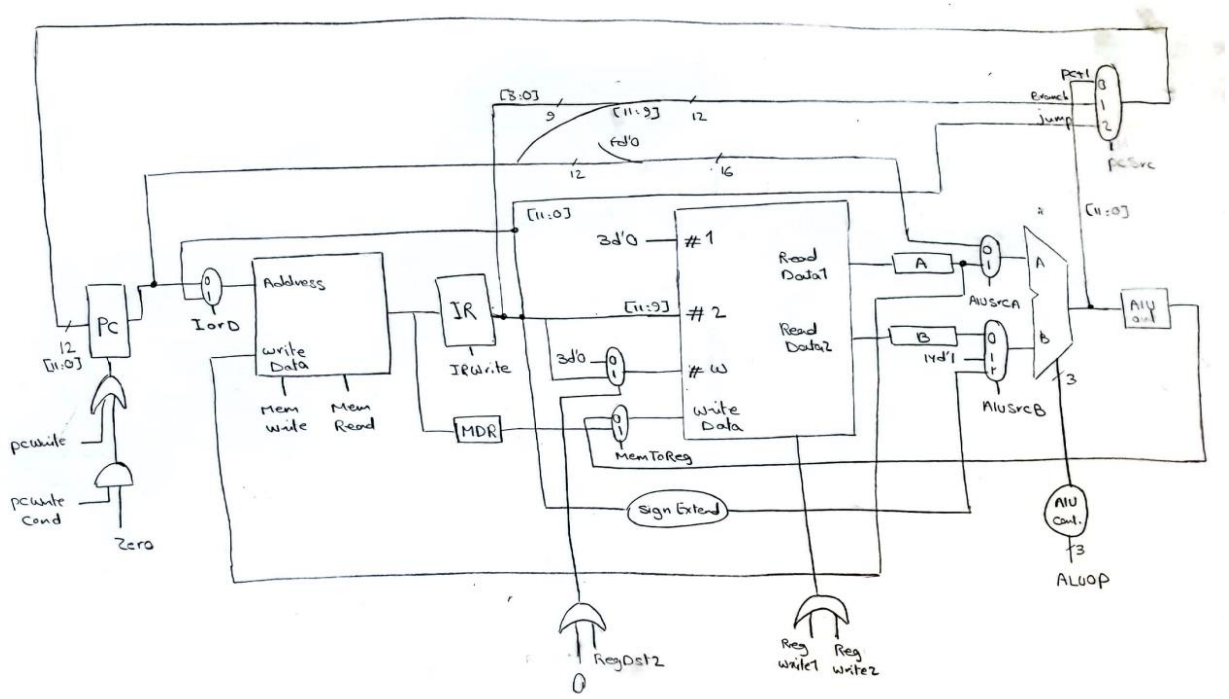
Computer Assignment4

“Multi Cycle Processor”

Ava Mirmohamadmahdi 810199501

Nesa Abbasimoghadam 810199457

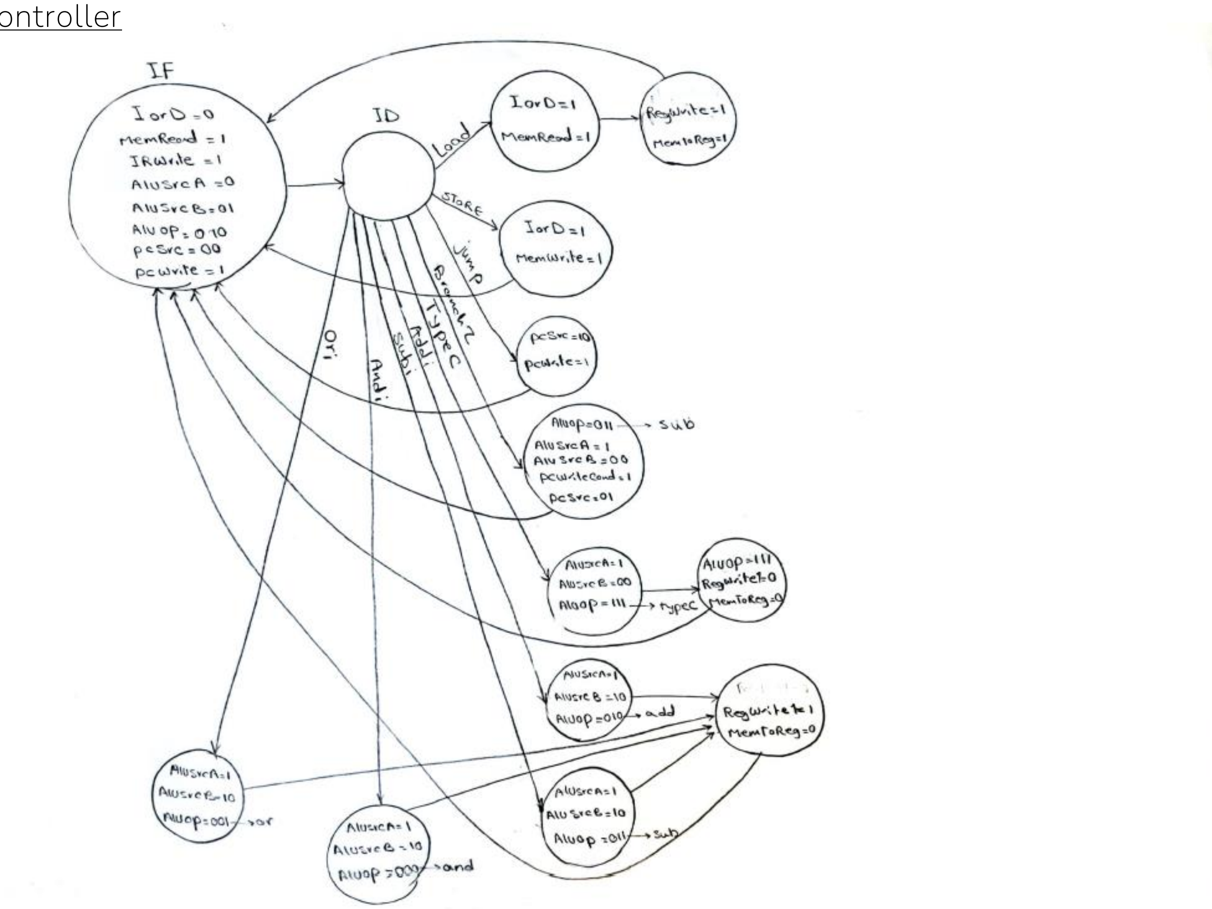
DataPath



DataPath Verilog

```
module DataPath(memAddress, memWriteData, IROut, zero, memReadData, ALUOperation,
                PCSrc, ALUSrcB, PCLoad, IOrD, RegDst, MemToReg, IRWrite, ALUSrcA, RegWrite, clk, rst);
    output [11:0] memAddress;
    output [15:0] memWriteData, IROut;
    output zero;
    input [15:0] memReadData;
    input [2:0] ALUOperation;
    input [1:0] PCSrc, ALUSrcB;
    input PCLoad, IOrD, RegDst, MemToReg, IRWrite, ALUSrcA, RegWrite, clk, rst;

    wire [11:0] pcOut;
    wire [11:0] muxPcSrcOut;
    Register #(12) PC(pcOut, muxPcSrcOut, rst, PCLoad, clk);

    wire [11:0] muxIOrDOut;
    Mux2To1 #(12) MuxIOrD(muxIOrDOut, pcOut, IROut[11:0], IOrD);

    Register #(16) IR(IROut, memReadData, rst, IRWrite, clk);

    wire [15:0] MDROut;
    Register #(16) MDR(MDROut, memReadData, rst, 1'b1, clk);

    wire [2:0] muxRegDstOut;
    Mux2To1 #(3) MuxRegDst(muxRegDstOut, 3'd0, IROut[11:9], RegDst);

    wire [15:0] ALURegOut;
    wire [15:0] muxMemToRegOut;
    Mux2To1 #(16) MuxMemToReg(muxMemToRegOut, ALURegOut, MDROut, MemToReg);

    wire [15:0] readData1, readData2;
    RegFile RegisterFile(readData1, readData2, muxMemToRegOut, 3'd0, IROut[11:9], muxRegDstOut, RegWrite, rst, clk);

    wire [15:0] signExtend;
    assign signExtend = {{4{IROut[11]}}, IROut[11:0]};

    wire [15:0] RegOutA, RegOutB;
    Register #(16) RegA(RegOutA, readData1, rst, 1'b1, clk);
    Register #(16) RegB(RegOutB, readData2, rst, 1'b1, clk);

    wire [15:0] AluSrcOutA, AluSrcOutB;
    wire [15:0] extendedPC;
    assign extendedPC = {4'd0, pcOut};
    Mux2To1 #(16) MuxALUSrcA(AluSrcOutA, extendedPC, RegOutA, ALUSrcA);
    Mux3To1 #(16) MuxALUSrcB(AluSrcOutB, RegOutB, 16'd1, signExtend, ALUSrcB);

    wire [15:0] ALUResult;
    ALU Alu(ALUResult, zero, AluSrcOutA, AluSrcOutB, ALUOperation);

    Register #(16) RegALU(ALURegOut, ALUResult, rst, 1'b1, clk);

    wire [11:0] extendedBranch;
    assign extendedBranch = {pcOut[11:9], IROut[8:0]};
    Mux3To1 #(12) MuxPCSrc(muxPcSrcOut, ALUResult[11:0], extendedBranch, IROut[11:0], PCSrc);

    assign memAddress= muxIOrDOut;
    assign memWriteData = RegOutA;

endmodule
```

Controller



Aluop	func	Aluoperation	Signals
010		010 Add	Regwrite2 =0 RegDst2=0
011		011 Sub	Regwrite2 =0 RegDst2=0
000		000 and	Regwrite2 =0 RegDst2=0
001		001 or	Regwrite2=0 RegDst2=0
111 (typec)	00000001	101 A را به عنوان ورودی می‌دهد	Regwrite2=1 RegDst2=1
111 (typec)	00000010	100 B را به عنوان ورودی می‌دهد	Regwrite2=1 RegDst2=0
111 (typec)	00000100	010 Add	Regwrite2=1 RegDst2=0
111 (typec)	00001000	011 Sub	Regwrite2=1 RegDst2=0
111 (typec)	00010000	000 and	Regwrite2=1 RegDst2=0
111 (typec)	00100000	001 or	Regwrite2=1 RegDst2=0
111 (typec)	01000000	110 A را not می‌کند	Regwrite2=1 RegDst2=0
111 (typec)	10000000	111 Nop	Regwrite2=0

Controller Verilog

```
always @(ps, opc) begin
    case(ps)
        IF: ns = ID;
        ID: begin
            case(opc)
                LOAD:  ns = Load1;
                STORE: ns = Store;
                JUMP:  ns = Jump;
                BRANCHZ: ns = BranchZ;
                TYPEC: ns = TypeC1;
                ADDI:  ns = Addi;
                SUBI:  ns = Subi;
                ANDI:  ns = Andi;
                ORI:  ns = Ori;
            endcase
        end
        Load1: ns = Load2;
        Load2: ns = IF;
        Store: ns = IF;
        Jump: ns = IF;
        BranchZ: ns = IF;
        TypeC1: ns = TypeC2;
        TypeC2: ns = IF;
        Addi: ns = IState;
        IState: ns = IF;
        Subi: ns = IState;
        Andi: ns = IState;
        Ori: ns = IState;
    endcase
end
```

```
always @(ps) begin
    {PCWrite, PCWriteCond, IOrD, MemWrite, MemRead, IRWrite, MemToReg, RegWrite1, ALUSrcB, PCSrc, ALUop, ALUSrcA} = 16'd5;
    case(ps)
        IF: begin
            IOrD = 1'b0;
            MemRead = 1'b1;
            IRWrite = 1'b1;
            ALUSrcA = 1'b0;
            ALUSrcB = 2'b01;
            ALUop = 3'b010;
            PCSrc = 2'b00;
            PCWrite = 1'b1;
        end
        ID: begin
            {MemWrite, MemRead, IRWrite, RegWrite1, PCWrite, PCWriteCond} = 6'b000000;
        end
        Load1: begin
            IOrD = 1'b1;
            MemRead = 1'b1;
        end
        Load2: begin
            RegWrite1 = 1'b1;
            MemToReg = 1'b1;
        end
        Store: begin
            IOrD = 1'b1;
            MemWrite = 1'b1;
        end
        Jump: begin
            PCSrc = 2'b10;
            PCWrite = 1'b1;
        end
        BranchZ: begin
            ALUop = 3'b011;
            ALUSrcA = 1'b1;
            ALUSrcB = 2'b00;
            PCWriteCond = 1'b1;
            PCSrc = 2'b01;
        end
        TypeC1: begin
            ALUSrcA = 1'b1;
            ALUSrcB = 2'b00;
            ALUop = 3'b111;
        end
        TypeC2: begin
            ALUop = 3'b111;
            RegWrite1 = 1'b0;
            MemToReg = 1'b0;
        end
        Addi: begin
            ALUSrcA = 1'b1;
            ALUSrcB = 2'b10;
            ALUop = 3'b010;
        end
        IState: begin
            RegWrite1 = 1'b1;
            MemToReg = 1'b0;
        end
        Subi: begin
            ALUSrcA = 1'b1;
            ALUSrcB = 2'b10;
            ALUop = 3'b011;
        end
        Andi: begin
            ALUSrcA = 1'b1;
            ALUSrcB = 2'b10;
            ALUop = 3'b000;
        end
        Ori: begin
            ALUSrcA = 1'b1;
            ALUSrcB = 2'b10;
            ALUop = 3'b001;
        end
    endcase
end
```

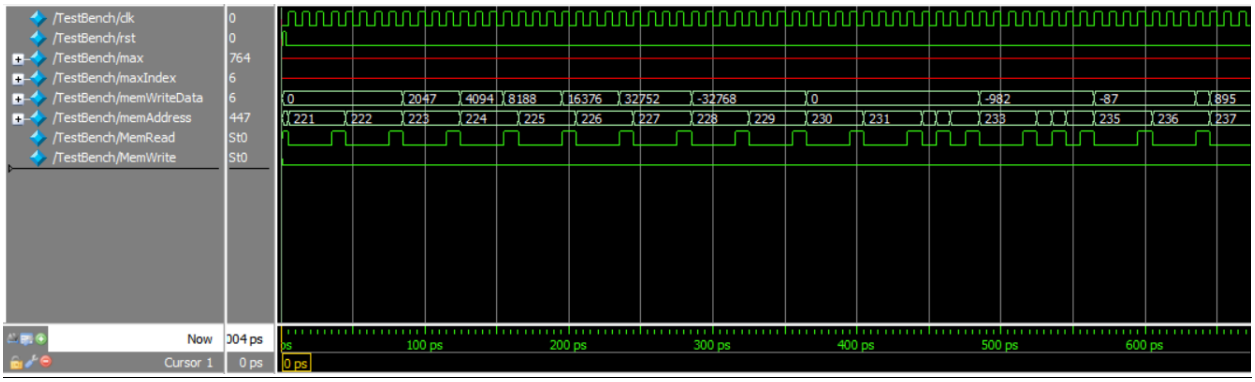
Instructions

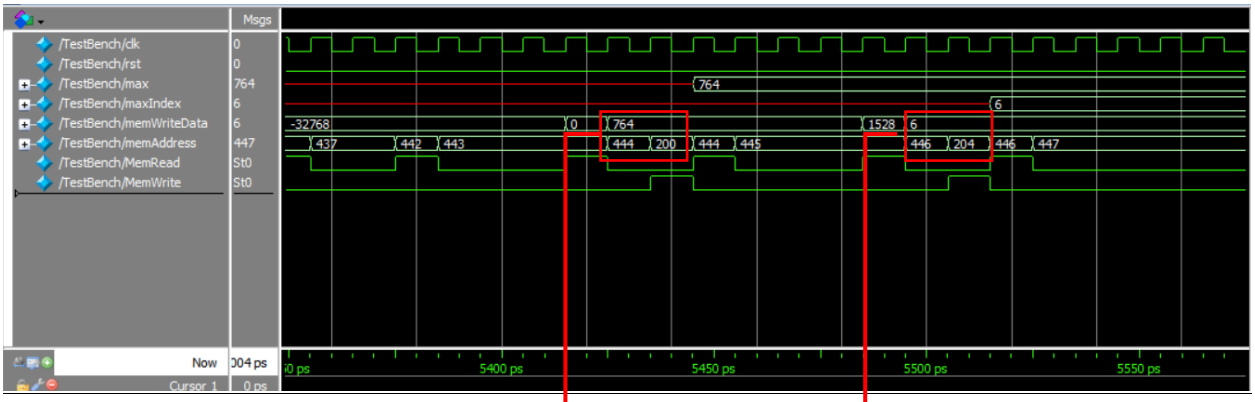
```
andi 0          # R0 = 0
ori 2047        # R0 = 2047
add 0           # R0 = 4094
add 0           # R0 = 8188
add 0           # R0 = 16376
add 0           # R0 = 32752
addi 16         # R0 = 32768
mvto 4          # R4 = 0x8000
andi 0          # R0 = 0
mvto 5          # R5 = 0

mvto 6          # R6 = 0 maxIndex
load 100        # R0 = mem[100]
mvto 7          # R7 = mem[100] -> max

load 101        # R0 = mem[101]
mvto 2          # R2 = mem[101] -> temp
sub 7           # R0 = temp - max
and 4           # R0 & 0x8000
branch 5, 239   # R0 == 0
jump 244

mvfrom 2        # R0 = temp
mvto 7          # max = temp
andi 0          # R0 = 0
ori 01          # R0 = 1
mvto 6          # R6 = 1 maxIndex
```

20 Random Numbers

```
0: 546      1: -87
2: 0        3: 178
4: 13       5: -23
6: 764      7: 15
8: 369      9: -783
10: -1      11: 3
12: 125     13: 93
14: -263    15: -7
16: 37      17: 88
18: 20      19: -1
```

Waveforms





مقدار بزرگترین عنصر در خانه ۲۰۰ ذخیره شده است.

اندیس بزرگترین عنصر در خانه ۲۰۴ ذخیره شده است.