به نام خدا

**دانشگاه تهران**

**دانشکده مهندسی برق و کامپیوتر**

**معماری کامپیوتر**

# تمرین دستی2

| | |
|---|---|
| نام و نام خانوادگی | محمدپویا افشاری-ریحانه احمدپور |
| شماره دانشجویی | 810198494-810198577 |
| تاریخ ارسال گزارش | 1400/9/21 – یکشنبه 21 آذر |

## فهرست گزارش سؤالات

- سوال 1 ........................................................................................
- سوال 2 ........................................................................................
- سوال 3 ........................................................................................
- سوال 4 ........................................................................................

آپکودهایی که می‌توانیم از آن استفاده کنیم:

| Inst | Opcode | Function |
|---|---|---|
| add | 000000 | 000001 |
| sub | 000000 | 000010 |
| and | 000000 | 000100 |
| or | 000000 | 001000 |
| slt | 000000 | 010000 |
| addi | 000001 | - |
| slti | 000010 | - |
| lw | 000011 | - |
| sw | 000100 | - |
| beq | 000101 | - |
| j | 000110 | - |
| jr | 000111 | - |
| jal | 001000 | - |

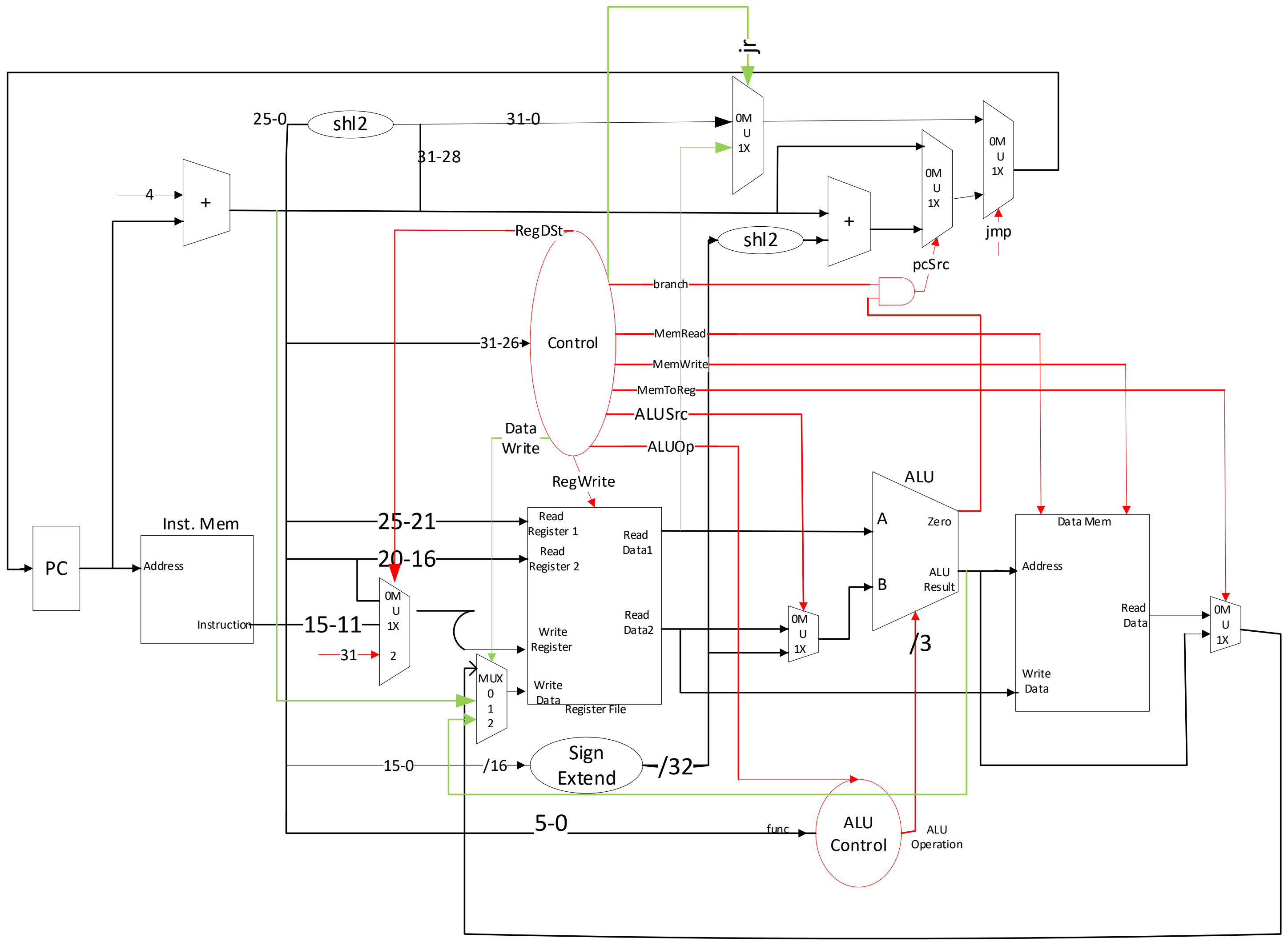
نوع دستوراتی که پردازنده پشتیبانی می‌کند:

**Arithmetic/Logical Instructions:** add, addi, sub, slt, slti, and, or

**Memory Reference Instruction:** lw, sw

**Control Flow Instructions:** j, jal, jr, beq

تصویر مسیر داده:



|  | RegDst | RegWrite | AluSrc | MemRead | MemWrite | MemToReg | Branch | AluOp | jmp | PCSrc | DataWrite | jr | operation |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Add | 01 | 1 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 001 |
| Sub | 01 | 1 | 1 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 010 |
| And | 01 | 1 | 1 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 011 |
| Or | 01 | 1 | 1 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 100 |
| Slt | 01 | 1 | 1 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 101 |
| Addi | 00 | 1 | 1 | 0 | 0 | 0 | 0 | 00 | 0 | 0 | 0 | 0 | 001 |
| Slti | 00 | 1 | 1 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 1 | 0 | 101 |
| Lw | 00 | 1 | 1 | 1 | 0 | 1 | 0 | 00 | 0 | 0 | 0 | 0 | 001 |
| Sw | 00 | 0 | 1 | 0 | 1 | 0 | 0 | 00 | 0 | 0 | 0 | 0 | 001 |
| Beq | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 01 | 0 | Zero | 0 | 0 | 010 |
| J | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 00 | 1 | 0 | 0 | 0 | ~(001) |
| Jr | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 00 | 1 | 0 | 0 | 1 | ~(001) |
| Jal | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 00 | 1 | 0 |  | 0 | ~(001) |

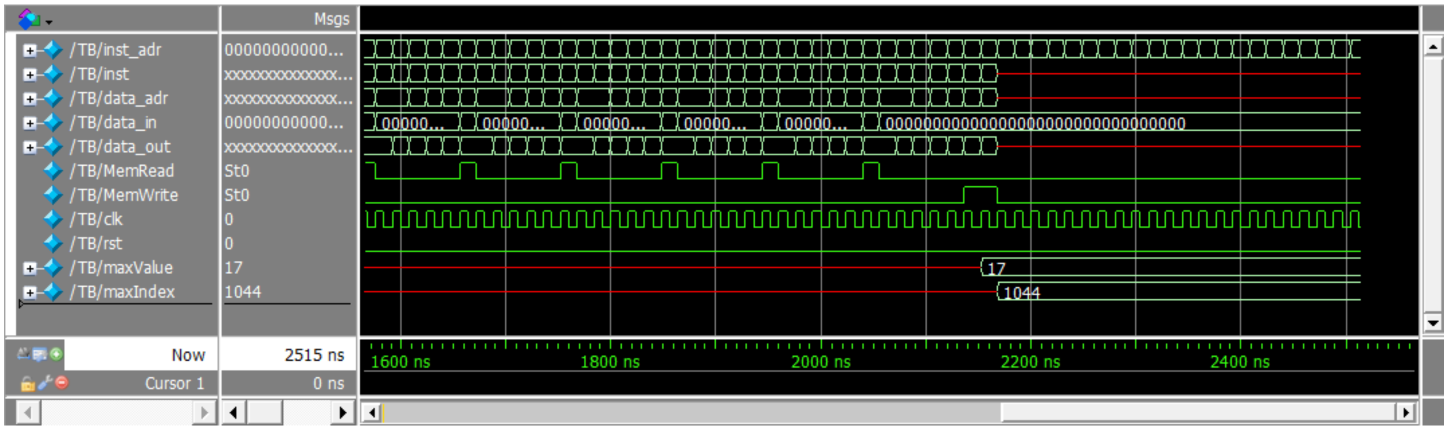
مسیر داده ساختاری:

```
module Datapath (
    clk,
    rst,
    inst_adr,
    inst,
    data_adr,
    data_out,
    data_in,
    RegDst,
    MemtoReg,
    ALUSrc,
    PCSrc,
    alu_operation,
    RegWrite,
    zero,
    dataToWrite,
    jr,
    jmp);

        input  clk, rst;
        output [31:0] inst_adr;
        input  [31:0] inst;
        output [31:0] data_adr;
        output [31:0] data_out;
        input  [31:0] data_in;
        input MemtoReg, ALUSrc, PCSrc, RegWrite;
        input  [2:0] alu_operation;
        output zero;

        //new signals coming from controller;
        input jr, jmp;
        //slti;
        input [1:0] dataToWrite,RegDst;
        wire [31:0] pc_out;
        wire [31:0] adder1_out;
        wire [31:0] ReadData1, ReadData2;
        wire [31:0] SignExt_out;
        wire [31:0] mux2_out;
        wire [31:0] aluOut;
        wire [31:0] adder2_out;
        wire [31:0] shl2_out;
        wire [31:0] mux3_out;
        wire [31:0] mux4_out;

        wire [4:0]  mux1_out;

        //new nets
        wire [31:0] mux5_out;
        wire [27:0] shl26_out;
        wire [31:0] in0MUX6;
        wire [31:0] mux6_out;
        wire [31:0] mux7_out;

        Adder ADDER_1(pc_out , 32'd4, 1'b0, adder1_out);

        mux3to1 #(32) MUX_1(inst[20:16], inst[15:11], 5'b11111, RegDst, mux1_out);

        //adding mux5 for writeData in RF
        mux3to1 #(32) MUX_5(mux4_out,  adder1_out, aluOut, dataToWrite, mux5_out);

        //set writeData input to mux5_out;
        RegisterFile  RF(mux5_out, inst[25:21], inst[20:16], mux1_out, RegWrite, rst, clk,
ReadData1, ReadData2);

        SignExt sign_ext(inst[15:0], SignExt_out);

        mux2to1 #(32) MUX_2(ReadData2, SignExt_out , ALUSrc, mux2_out);

        ALU alu(ReadData1, mux2_out, alu_operation, aluOut, zero);

        SHL2 #(32) shl2(SignExt_out, shl2_out);

        Adder ADDER_2(adder1_out, shl2_out, 1'b0, adder2_out);

        mux2to1 #(32) MUX_3(adder1_out, adder2_out, PCSrc, mux3_out);

        mux2to1 #(32) MUX_4(aluOut, data_in, MemtoReg, mux4_out);

        //adding MUX6
        SHL2 #(28) shl26(inst[25:0], shl26_out);
        assign in0MUX6 = {adder1_out[31:28], shl26_out};

        mux2to1 #(32) MUX_6(in0MUX6, ReadData1, jr, mux6_out);

        //adding MUX7
        mux2to1 #(32) MUX_7(mux3_out, mux6_out, jmp, mux7_out);

        register #(32) PC(mux7_out, rst, 1'b1, clk, pc_out);

        assign inst_adr = pc_out; //instruction generated for
        assign data_adr = aluOut;  //Address in data_memory where data should be written
        assign data_out = ReadData2; //Value that should be written in data_memory

endmodule
```

کنترل داده ترکیبی:

```
module Controller (
    opcode,
    func,
    zero,
    RegDst,
    MemtoReg,
    RegWrite,
    ALUsrc,
    MemRead,
    MemWrite,
    PCSrc,
    operation,
    dataToWrite,
    jr,
    jmp);

    input [5:0] opcode;
    input [5:0] func;
    input zero;
    output  PCSrc;
    output logic MemtoReg, RegWrite, ALUsrc, MemRead, MemWrite;
    output [2:0] operation;
    output logic [1:0] RegDst, dataToWrite;//our signals
    output logic jr, jmp;//our signals

    logic [1:0] ALUOp;
    logic beq;

    parameter [5:0]
        RT_OPC = 6'b000000, //RType opcode
        ADDI_OPC = 6'b000001, //IType
        SLTI_OPC = 6'b000010, //IType
        LW_OPC = 6'b000011, //IType(mem ref)
        SW_OPC = 6'b000100, //IType(mem ref)
        BEQ_OPC = 6'b000101, //IType(control flow)
        J_OPC = 6'b000110, //JType
        JR_OPC = 6'b000111, //RType
        JAL_OPC = 6'b001000; //JType

    ALUController ALU_CTRL(ALUOp, func, operation);

    always @(opcode, operation)
    begin
        {RegDst, ALUsrc, MemtoReg, RegWrite, MemRead, MemWrite, beq, ALUOp, jmp, jr,
dataToWrite} = 15'd0;
        case (opcode)
            //RType instructions //add,sub,and,or,slt
            RT_OPC: {RegDst, RegWrite, ALUOp} = {2'b01, 1'b1, 2'b11};

            //IType Add immediate (addi) instruction
            ADDI_OPC: {RegWrite, ALUsrc} = 2'b11;

            //IType Set Less Than immediate (SLTi) instruction
            SLTI_OPC: {RegWrite, ALUsrc, ALUOp, dataToWrite} = {1'b1, 1'b1, 2'b10, 2'b10};

            //IType(mem ref) Load Word (lw) instruction //using initial state for ALUOp which
is ALUOp=2'b00
            LW_OPC: {ALUsrc, MemtoReg, RegWrite, MemRead} = 4'b1111;

            //IType(mem ref) Store Word (sw) instruction //using initial state for ALUOp
which is ALUOp=2'b00
            SW_OPC: {ALUsrc, MemWrite} = 2'b11;

            //IType(control flow) Branch on equal (beq) instruction
            BEQ_OPC: {beq, ALUOp} = {1'b1, 2'b01};

            //JType Jump (j) instruction
            J_OPC: {jmp} = 1'b1;

            //RType Jump (j) instruction
            JR_OPC: {jr, jmp} = {2'b11};

            ///JType Jump and link (jal) instruction
            JAL_OPC: {RegWrite, RegDst, dataToWrite, jmp} = {1'b1, 2'b10, 2'b01, 1'b1};

        endcase
    end
    assign PCSrc = beq ? (zero) : 1'b0;

endmodule
```

صحت طراحی:



نوع دستوراتی که برای الگوریتم ماکزیمم گیری از آن استفاده کردیم:

```
0:
{ADDI_OPC,R0,R2,MEM_START};//addi R2,R0,1000 : R2=R0+1000=1000; //init R2 to MEM_START=1000 to stop the loop
000001.00000.00010.0000001111101000
1:
{ADDI_OPC,R0,R3,MEM_END};//addi R3,R0,1076 : R3=R0+1080=1080; //init R2 to MEM_END=1080 to start the loop at defined location
000001.00000.00011.0000010000111000
2:
{BEQ_OPC,R2,R3,END};//beq R2,R3,END : R2==R3? PC<-PC+4+END<<2 : PC<-PC+4; //END:PC points to next instruction //loop condition //lable LOOP //jump to END=
000101.00010.00011.0000000000000111
3:
{LW_OPC,R2,R4,MEM_START};//lw R4,0(R2) : R4<-RAM[R2+0000] // load num i in R4 Register
000011.00010.00100.0000000000000000
4:
{RT,R5,R4,R7,SHAMNT,SLT_FUNC};//slt R7,R5,R4 : R5<R4 ? R7<-1 : R7<-0; //res of if condition store in R7 //we have one in R7 if max < num[i]
000000.00101.00100.00111.00000.101010
5:
{BEQ_OPC,R7,R0,NEXT};//beq R7,R0,NEXT : R7==R0? PC<-PC+4+NEXT<<2 : PC<-PC+4; //if they aren't equal(means max is less than num[i]) update max and max index otherwise jump to next num //lable LOOP //jump to NEXT
000101.00111.00000.0000000000000010
6:
{ADDI_OPC,R4,R5,ZERO_BITS};//addi R5,R4,0 : R5=R4+0 //update max_value(R5) to R4
000001.00100.00101.0000000000000000
7:
{ADDI_OPC,R2,R6,ZERO_BITS};//addi R6,R2,0 : R6=R2+0 //update max_index(R6) to R3
000001.00010.00110.0000000000000000
8:
{ADDI_OPC,R2,R2,loopStep};//addi R2,R2,4 : R2=R2+4 //update loop counter in R3
000001.00010.00010.0000000000000100
9:
{J_OPC,ZERO_BITS,ZERO_BITS,LOOP};//J LOOP//lable NEXT
000110.00000.00000.0000000000000010
10:
{SW_OPC,R0,R5,MAXVAL_END_ADDR};//sw R5,2000(R0) : RAM[R0+2000]<-R5 //lable END
000100.00000.00101.0000011111010000
11:
{SW_OPC,R0,R6,MAXIDX_END_ADDR};//sw R6,2004(R0) : RAM[R0+2004]<-R6
000100.00000.00110.0000011111010100
```