Lab3 Report

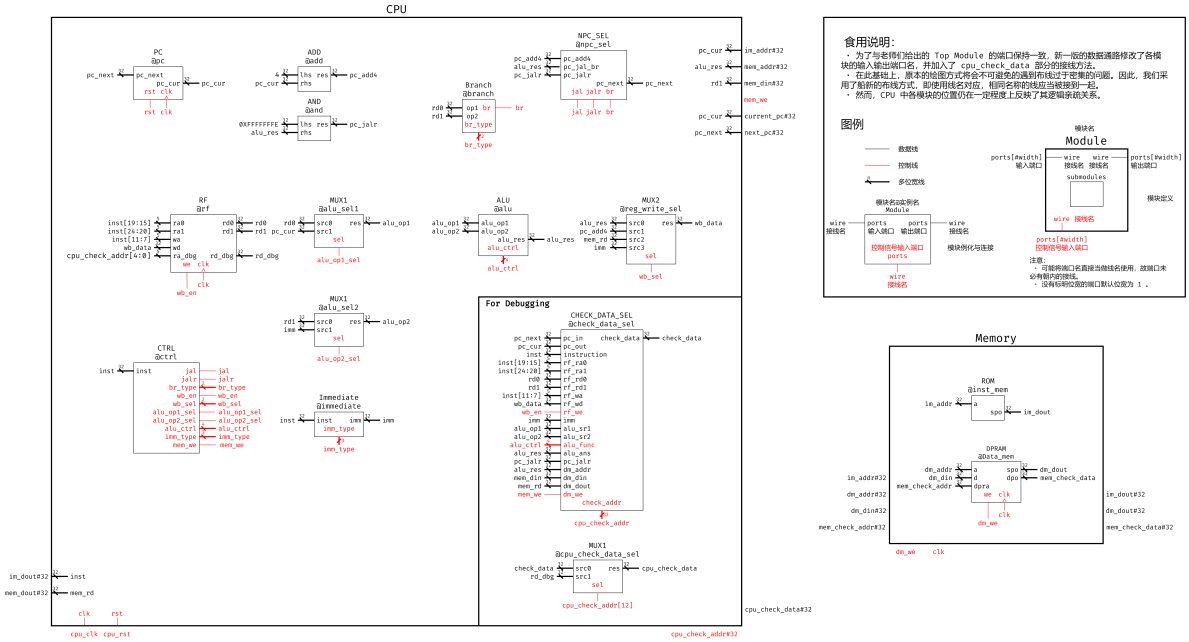
实验原理

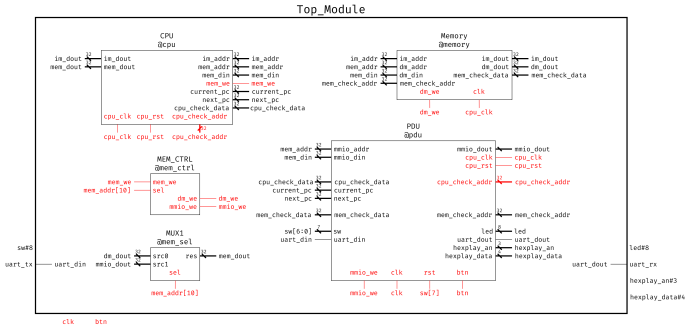
实验内容

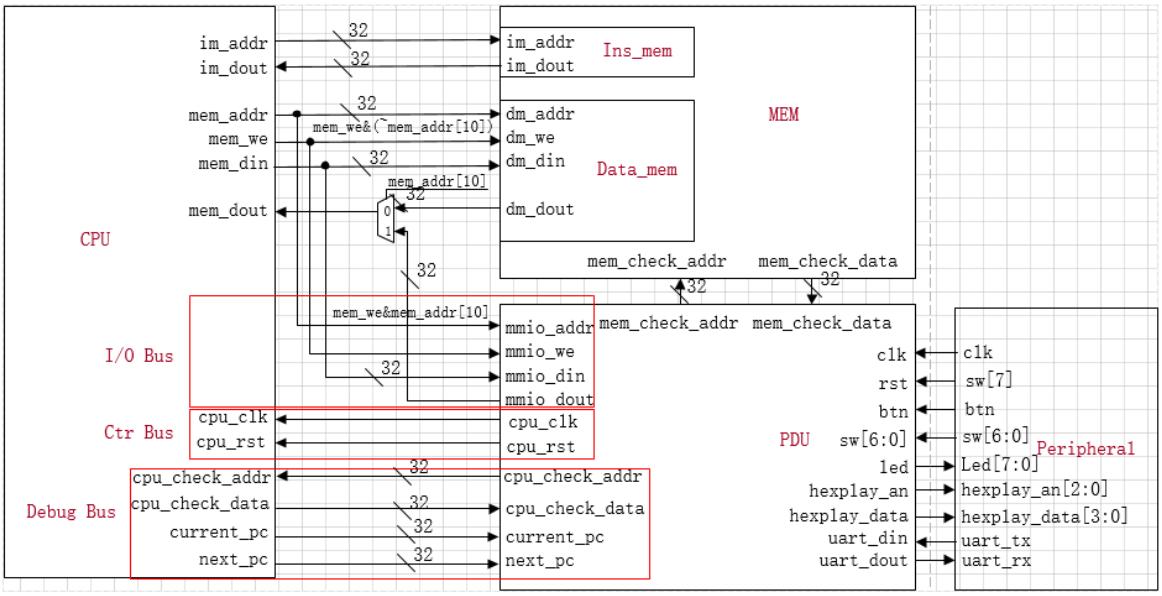
1. 深入理解单周期CPU数据通路、各种基本组件
2. 完成单周期CPU的硬件设计，并完成仿真
3. 了解外设与调试单元PDU的结构和使用
4. 将PDU接入CPU并完成上板测试
5. 在CPU上运行汇编程序

设计流程

数据通路







关键模块设计

立即数模块（Immediate）

使用位拼接实现立即数的提取和位扩展

```
always @(*) begin
    case (imm_type)
        3'h1: imm={{20{inst[31]}}, inst[31:20]};
   //I-Type
        3'h2: imm={{20{inst[31]}}, inst[31:25], inst[11:7]};
   //S-Type
        3'h3: imm={{20{inst[31]}}, inst[7], inst[30:25], inst[11:8], {1'b0}};   //B-Type
        3'h4: imm={inst[31:12], 12'b0};
    //U-Type
        3'h5: imm={{12{inst[31]}}, inst[19:12], inst[20], inst[30:21], {1'b0}}; //J-Type
        default: imm=32'b0;
    endcase
end
```

跳转模块（Branch）

当`br_type==0`时，当前指令不是跳转指令；当`br_type`为1~6时，若满足相应条件，则`br=1`，则跳转

```
always @(*) begin
    br=0;
    case (br_type)
        3'h1:begin
            br=(op1==op2)?1:0;  //beq
        end
        3'h2:begin
            br=(op1!=op2)?1:0;  //bne
        end
        3'h3:begin
            br=($signed(op1)<$signed(op2))?1:0; //blt
        end
```

```
        3'h4:begin
            br=($signed(op1)>=$signed(op2))?1:0;    //bge
        end
        3'h5:begin
            br=(op1<op2)?1:0;   //bltu
        end
        3'h6:begin
            br=(op1>=op2)?1:0;  //bgeu
        end
    endcase
end
```

PC选择模块（NPC_SEL）

若`jal||br==1`，则`PC`跳转到`pc_jal_br`；若`jalr==1`，则`PC`跳转到`pc_jalr`。其中`jal` `jalr`由`CTRL`模块产生，`br`由`Branch`模块产生

```
always @(*) begin
    pc_next=pc_add4;
    if(jal||br)
        pc_next=pc_jal_br;
    if(jalr)
        pc_next=pc_jalr;
end
```

控制模块（CTRL）

根据不同指令的行为，以及数据通路，输出不同的控制信号

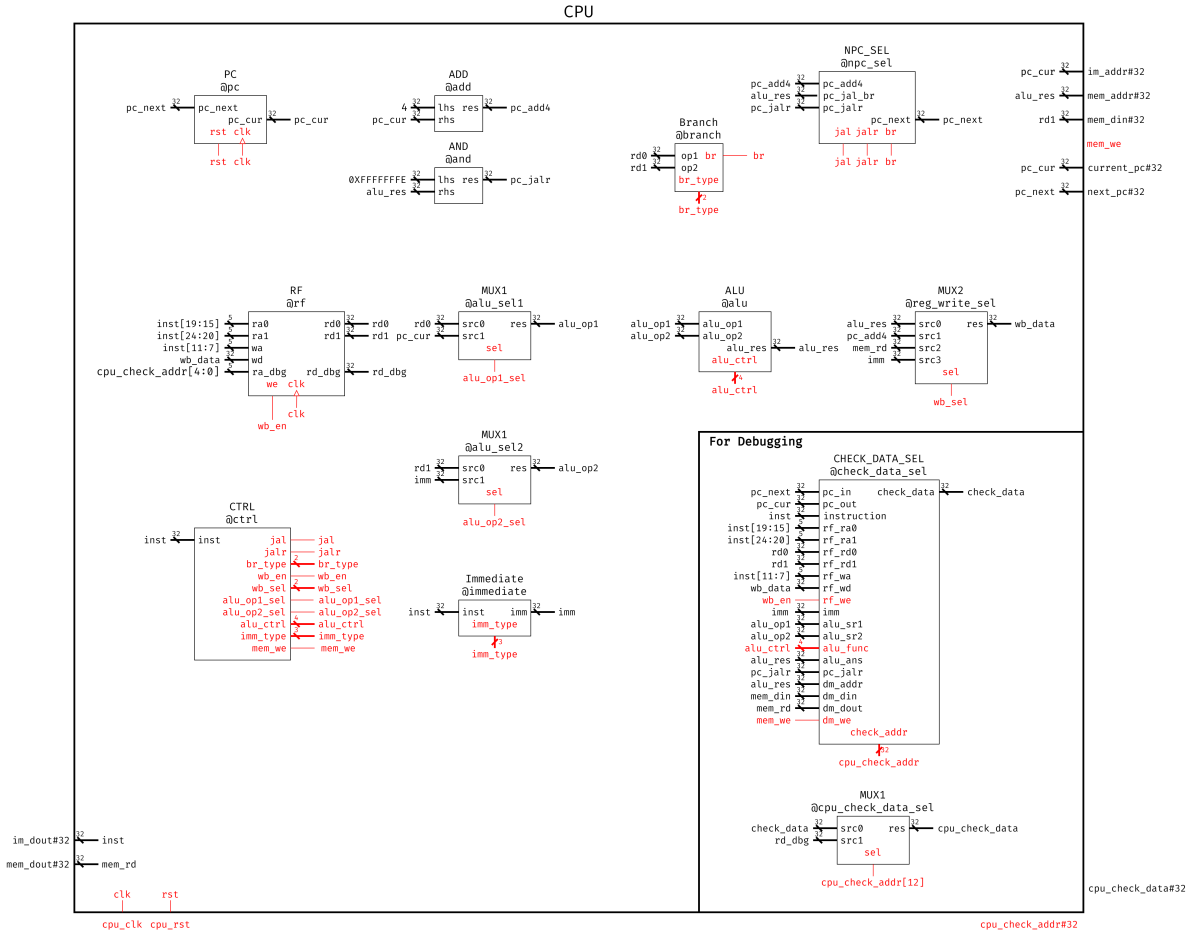
```
always @(*) begin
    jal=0; jalr=0; br_type=0; wb_en=0;
    wb_sel=0; alu_op1_sel=0; alu_op2_sel=0;
    alu_ctrl=0; mem_we=0; imm_type=0;
    case (inst[6:0])
        7'b0010011:begin
            alu_op1_sel=0;
            alu_op2_sel=1;
            imm_type=1;
            wb_sel=0;
            wb_en=1;
            case (inst[14:12])
                3'b111: alu_ctrl=4'b0000;   //addi
                3'b001: alu_ctrl=4'b1001;   //slli
                3'b101:begin
                    case (inst[31:25])
                        7'b0000000: alu_ctrl=4'b1000;   //srli
                        7'b0100000: alu_ctrl=4'b1010;   //srai
                    endcase
                end
            endcase
        end
        7'b0110011:begin
            alu_op1_sel=0;
            alu_op2_sel=0;
            wb_sel=0;
```

```
            wb_en=1;
            case (inst[14:12])
                3'b000:begin
                    case (inst[31:25])
                        7'b0000000: alu_ctrl=4'b0000;   //add
                        7'b0100000: alu_ctrl=4'b0001;   //sub
                    endcase
                end
                3'b111: alu_ctrl=4'b0101;   //and
                3'b110: alu_ctrl=4'b0110;   //or
            endcase
        end
        7'b0110111:begin
            alu_op2_sel=1;
            imm_type=4;
            wb_sel=3;
            wb_en=1;
            alu_ctrl=4'b0000;   //lui
        end
        7'b0010111:begin
            alu_op1_sel=1;
            alu_op2_sel=1;
            imm_type=4;
            wb_sel=0;
            wb_en=1;
            alu_ctrl=4'b0000;   //auipc
        end
        7'b1101111:begin
            jal=1;
            alu_op1_sel=1;
            alu_op2_sel=1;
            imm_type=5;
            wb_sel=1;
            wb_en=1;
            alu_ctrl=4'b0000;   //jal
        end
        7'b1100111:begin
            jalr=1;
            alu_op1_sel=0;
            alu_op2_sel=1;
            imm_type=1;
            wb_sel=1;
            wb_en=1;
            alu_ctrl=4'b0000;   //jalr
        end
        7'b1100011:begin
            alu_op1_sel=1;
            alu_op2_sel=1;
            imm_type=3;
            alu_ctrl=4'b0000;
            case (inst[14:12])
                3'b000: br_type=1;  //beq
                3'b001: br_type=2;  //bne
                3'b100: br_type=3;  //blt
                3'b101: br_type=4;  //bge
```

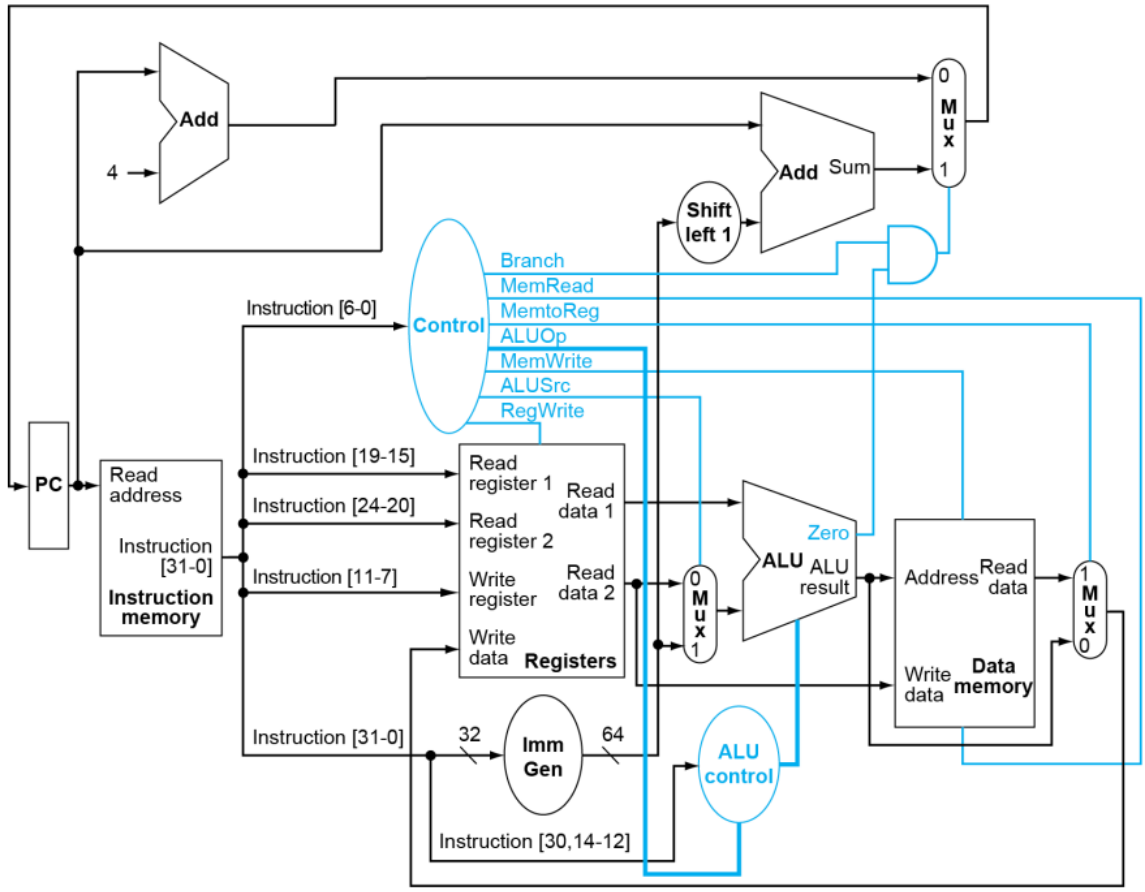
```
                3'b110: br_type=5;  //bltu
                3'b111: br_type=6;  //bgeu
            endcase
        end
        7'b0000011:begin
            case (inst[14:12])
                3'b010:begin
                    alu_op1_sel=0;
                    alu_op2_sel=1;
                    imm_type=1;
                    alu_ctrl=4'b0000;
                    wb_sel=2;
                    wb_en=1;        //lw
                end
            endcase
        end
        7'b0100011:begin
            case (inst[14:12])
                3'b010:begin
                    alu_op1_sel=0;
                    alu_op2_sel=1;
                    imm_type=2;
                    alu_ctrl=4'b0000;
                    mem_we=1;       //sw
                end
            endcase
        end
    endcase
end
```

数据通路的差异

本实验数据通路



课件中的数据通路



1. `CTRL` 模块仅由 `Inst[6:0]` 控制，使得ALU控制信号必须由额外的 `ALU_Control` 模块控制
2. 跳转指令中仅支持 `beq` 指令
3. ALU模块的操作数种类更少，支持的指令更少

4. 回写的数据类型更少

体会

希望其他课实验手册都能像COD助教写的这么好