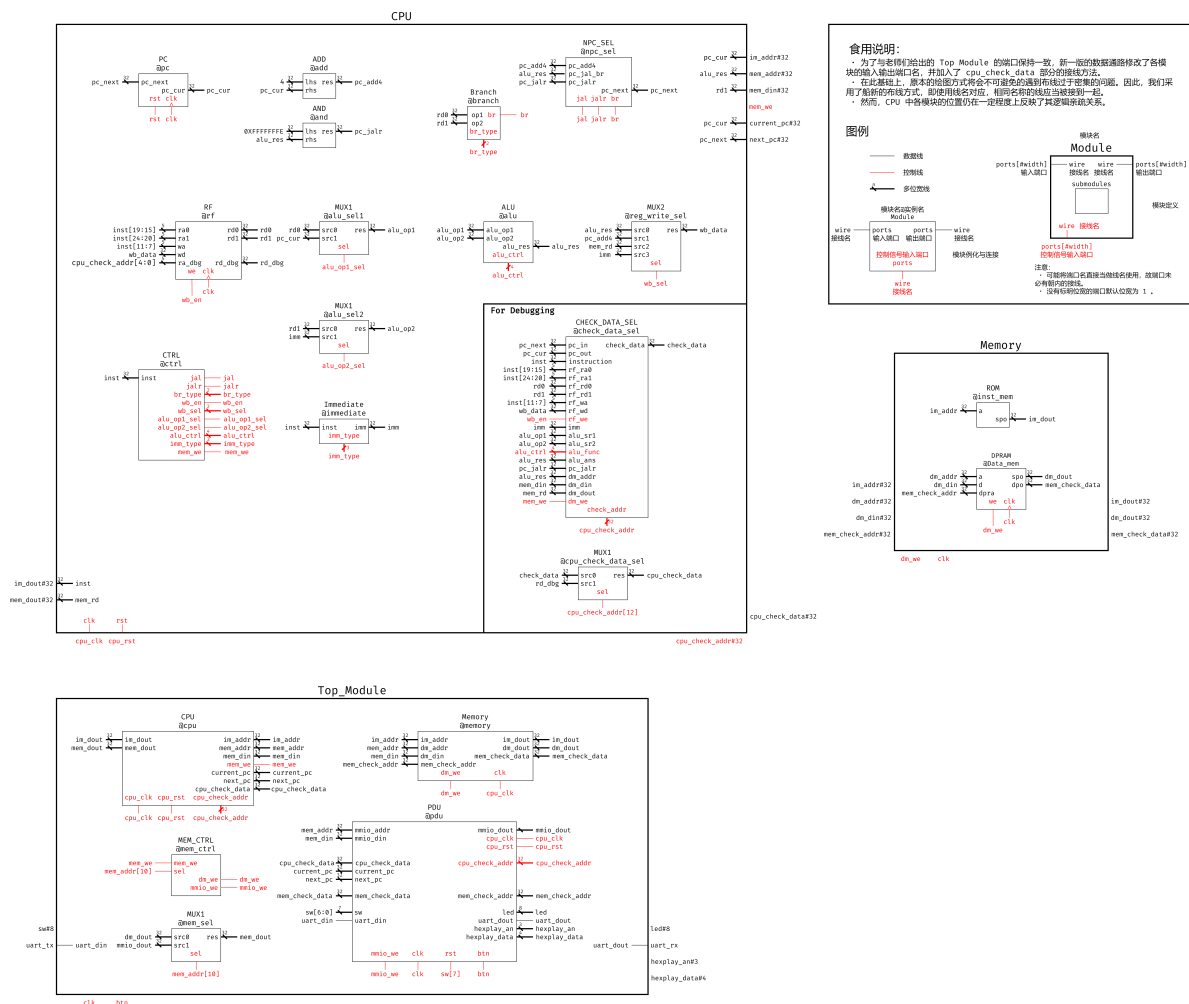
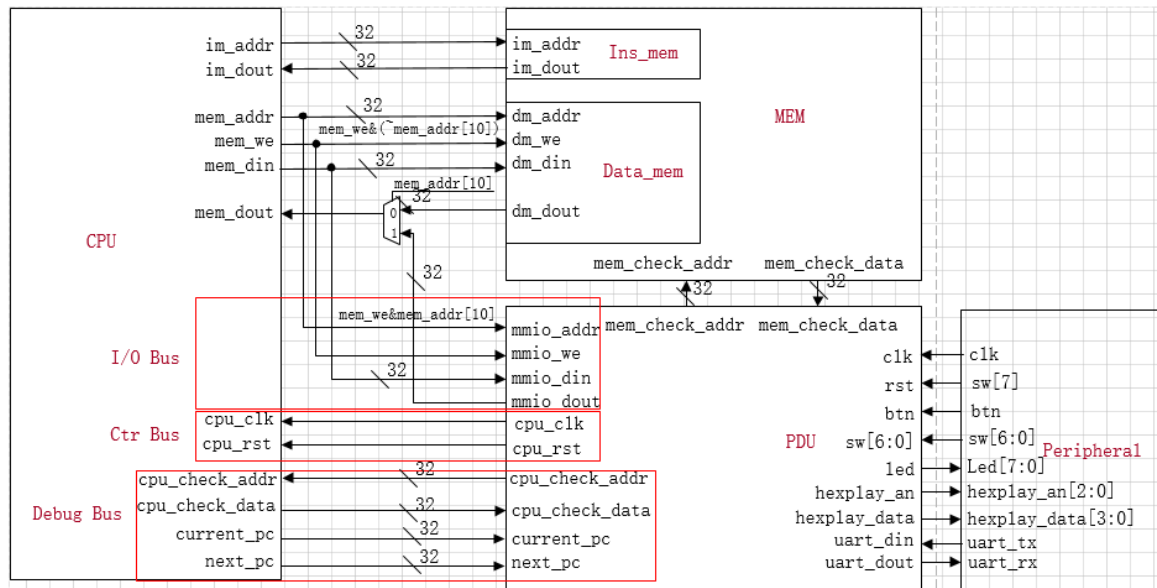


## 实验内容

- ## 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

```

1  always @(*) begin
2      case (imm_type)
3          3'h1: imm={{20{inst[31]}}, inst[31:20]};
4          //I-Type
5          3'h2: imm={{20{inst[31]}}, inst[31:25], inst[11:7]};
6          //S-Type
7          3'h3: imm={{20{inst[31]}}, inst[7], inst[30:25], inst[11:8],
8          {1'b0}}; //B-Type
9          3'h4: imm={inst[31:12], 12'b0};
10         //U-Type
11         3'h5: imm={{12{inst[31]}}, inst[19:12], inst[20], inst[30:21],
12         {1'b0}}; //J-Type
13         default: imm=32'b0;
14     endcase
15 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

```

1  always @(*) begin
2      br=0;
3      case (br_type)
4          3'h1:begin
5              br=(op1==op2)?1:0; //beq
6          end
7          3'h2:begin
8              br=(op1!=op2)?1:0; //bne
9          end
10         3'h3:begin
11             br=($signed(op1)<$signed(op2))?1:0; //blt
12         end

```

```

13     3'h4:begin
14         br=($signed(op1)>=$signed(op2))?1:0;    //bge
15     end
16     3'h5:begin
17         br=(op1<op2)?1:0;    //bltu
18     end
19     3'h6:begin
20         br=(op1>=op2)?1:0;    //bgeu
21     end
22 endcase
23 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

```

1  always @(*) begin
2      pc_next=pc_add4;
3      if(jal||br)
4          pc_next=pc_jal_br;
5      if(jalr)
6          pc_next=pc_jalr;
7  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

```

1  always @(*) begin
2      jal=0; jalr=0; br_type=0; wb_en=0;
3      wb_sel=0; alu_op1_sel=0; alu_op2_sel=0;
4      alu_ctrl=0; mem_we=0; imm_type=0;
5      case (inst[6:0])
6          7'b0010011:begin
7              alu_op1_sel=0;
8              alu_op2_sel=1;
9              imm_type=1;
10             wb_sel=0;
11             wb_en=1;
12             case (inst[14:12])
13                 3'b111: alu_ctrl=4'b0000;    //addi
14                 3'b001: alu_ctrl=4'b1001;    //slli
15                 3'b101:begin
16                     case (inst[31:25])
17                         7'b0000000: alu_ctrl=4'b1000;    //srli
18                         7'b0100000: alu_ctrl=4'b1010;    //srai
19                     endcase
20                 end
21             endcase
22         end
23         7'b0110011:begin
24             alu_op1_sel=0;
25             alu_op2_sel=0;
26             wb_sel=0;

```

```

27         wb_en=1;
28         case (inst[14:12])
29             3'b000:begin
30                 case (inst[31:25])
31                     7'b0000000: alu_ctrl=4'b0000;    //add
32                     7'b0100000: alu_ctrl=4'b0001;    //sub
33                 endcase
34             end
35             3'b111: alu_ctrl=4'b0101;    //and
36             3'b110: alu_ctrl=4'b0110;    //or
37         endcase
38     end
39     7'b0110111:begin
40         alu_op2_sel=1;
41         imm_type=4;
42         wb_sel=3;
43         wb_en=1;
44         alu_ctrl=4'b0000;    //lui
45     end
46     7'b0010111:begin
47         alu_op1_sel=1;
48         alu_op2_sel=1;
49         imm_type=4;
50         wb_sel=0;
51         wb_en=1;
52         alu_ctrl=4'b0000;    //auipc
53     end
54     7'b1101111:begin
55         jal=1;
56         alu_op1_sel=1;
57         alu_op2_sel=1;
58         imm_type=5;
59         wb_sel=1;
60         wb_en=1;
61         alu_ctrl=4'b0000;    //jal
62     end
63     7'b1100111:begin
64         jalr=1;
65         alu_op1_sel=0;
66         alu_op2_sel=1;
67         imm_type=1;
68         wb_sel=1;
69         wb_en=1;
70         alu_ctrl=4'b0000;    //jalr
71     end
72     7'b1100011:begin
73         alu_op1_sel=1;
74         alu_op2_sel=1;
75         imm_type=3;
76         alu_ctrl=4'b0000;
77         case (inst[14:12])
78             3'b000: br_type=1;    //beq
79             3'b001: br_type=2;    //bne
80             3'b100: br_type=3;    //blt
81             3'b101: br_type=4;    //bge

```

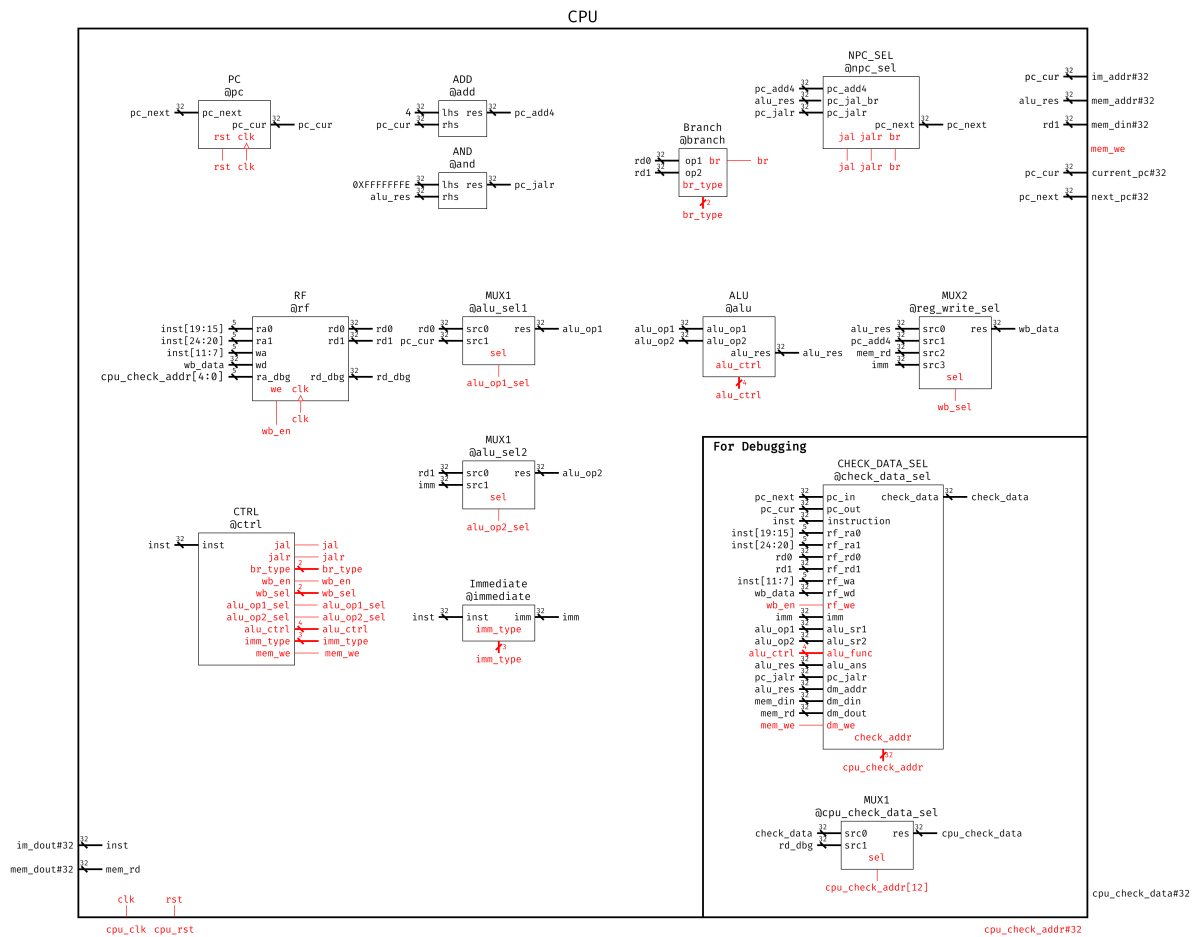
```

82         3'b110: br_type=5;    //bltu
83         3'b111: br_type=6;    //bgeu
84     endcase
85 end
86 7'b0000011:begin
87     case (inst[14:12])
88         3'b010:begin
89             alu_op1_sel=0;
90             alu_op2_sel=1;
91             imm_type=1;
92             alu_ctrl=4'b0000;
93             wb_sel=2;
94             wb_en=1;            //lw
95         end
96     endcase
97 end
98 7'b0100011:begin
99     case (inst[14:12])
100        3'b010:begin
101            alu_op1_sel=0;
102            alu_op2_sel=1;
103            imm_type=2;
104            alu_ctrl=4'b0000;
105            mem_we=1;           //sw
106        end
107    endcase
108 end
109 endcase
110 end

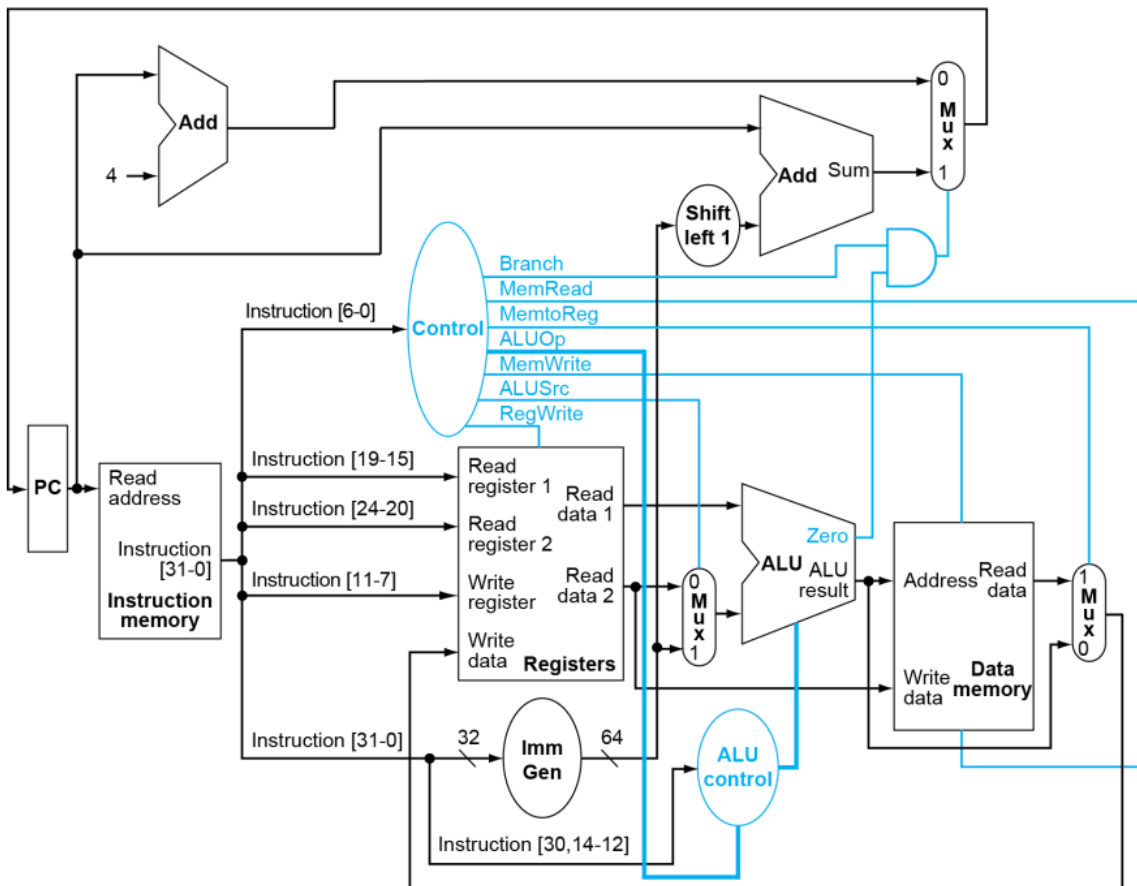
```

## 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\_Control1 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