哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：2020 春季

课程名称：计算机组成原理（实验）

实验名称：微程序控制器设计

实验性质：综合设计型

实验学时：4 地点：线上

学生班级：1801101

学生学号：180110115

学生姓名：方澳阳

评阅教师：

报告成绩：

实验与创新实践教育中心制

2020 年 5 月

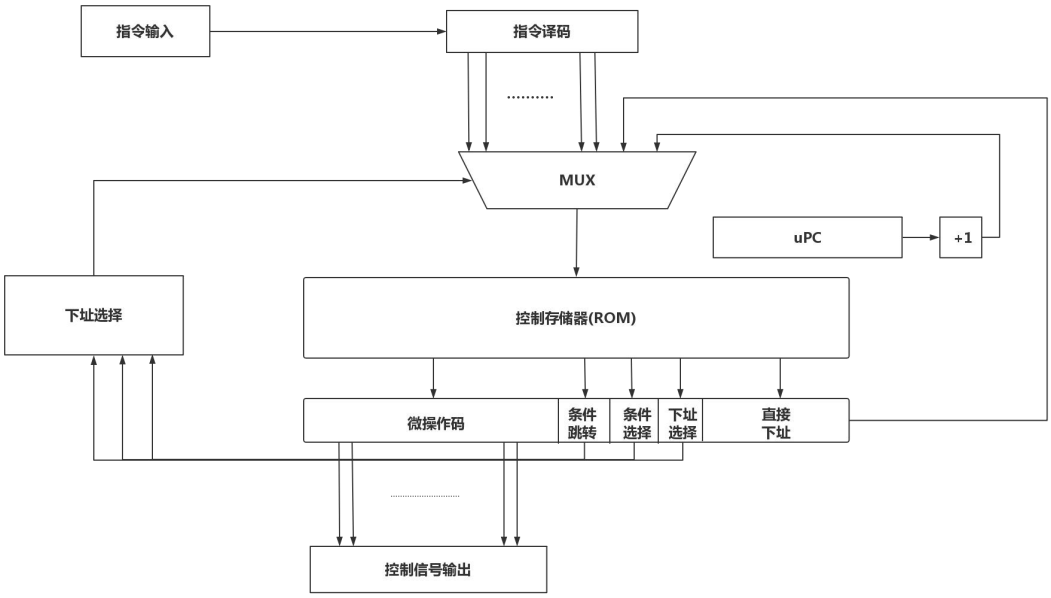
一、 实验项目

微程序控制器设计

二、 系统功能详细设计及实现

1. *对系统进行详细设计描述。用硬件框图描述系统主要功能及各模块之间的相互关系；（此部分内容为本报告重点内容，需详细描述）*

CU 框图:



CU 的工作是接收来自 IR 的指令，进行分析，然后给出相对应的微操作的信号。

先取微指令，再通过分析取到的微指令以及接收到的 IR 指令来决定下一阶段，即执行周期需要进行的操作。

接下来每次都根据取到的指令进行下址选择，如：若指令仍在执行周期中，则下一条微指令的地址为当前微指令的地址+1， 若执行周期结束，则下一条微指令则返回取指阶段，然后重复以上的过程，即实现了不断取指分析执行的过程。

其中，指令输入即为 8 位的机器码，因为指令集只有 10 条，因此只使用了高四位的数据位作为指令的标志。指令译码通过 case 语句分别处理 10 中不同情况下的指令操

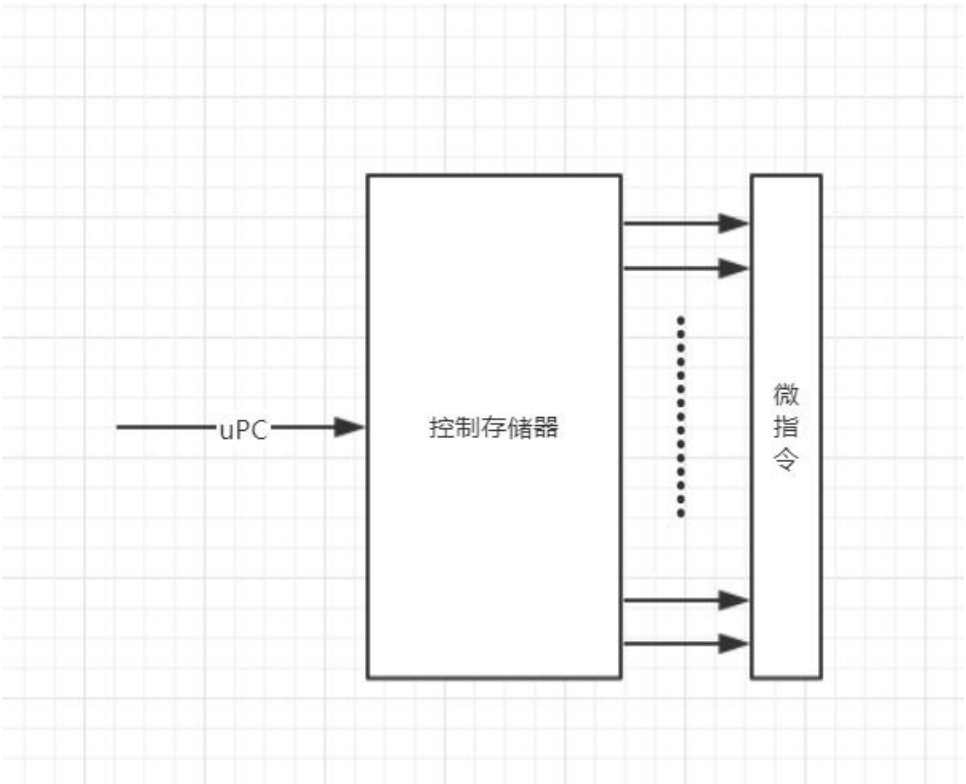
作。在每个 case 中再通过下址选择组合电路给出下一条微指令的地址，是回到取指指令的地址还是+1 还是跳转到某个指令的入口。

控制存储器存放了 29 条微指令，11 个微程序(其中有一个是 NOP)，微程序的入口地址分别为 0,4,7,12,16,17,188,19,22,25,28. 通过 uPC,控制存储器可以给出微指令.

再通过微指令中的微操作码, 可以输出当前操作需要的信号。执行占用总线、传输数据等操作。

同时，微指令中的下址选择、条件选择、条件跳转等字段就可以配合指令译码给出下一条指令的地址。

2. 模块描述。包括模块功能，输入、输出端口及变量含义，时序等。




```
curr_code = ucode_mem[uPC];//取出微指令
```

控制存储器的输入是 uPC,输出是一条微指令.

```
assign PC_i = curr_code[29];
assign IR_i = curr_code[28];
assign MAR_i = curr_code[27];
assign MDR_i = curr_code[26];
assign ACC_i = curr_code[25];
```

```
assign SP_i = curr_code[24];
assign R_i = curr_code[23];
assign PC_o = curr_code[22];
assign MDR_o = curr_code[21];
assign ACC_o = curr_code[20];
assign SP_o  = curr_code[19];
assign R_o = curr_code[18];
assign PC_Sel = curr_code[17];
assign MDR_Sel = curr_code[16];
assign ACC_Sel = curr_code[15];
assign ALU_Sel = curr_code[14];
assign MemRead = curr_code[13];
assign MemWrite = curr_code[12];
wire condJMP = curr_code[11];
wire condSel = curr_code[10];
wire [1:0]nextAddrSel = curr_code[9:8];
wire [7:0]addr = curr_code[7:0];
```

通过 wire 为微指令的各个端口输出.

MUX 模块:

通过指令译码 IR 的高 4 位的值,以及下址选择 `nextAddrSel` 位来判断下地址应该是什么地址

```
case (IR[7:4])
        4'b0000: begin
            if(nextAddrSel==2'b00)
            begin
                nextAddress <= uPC+1;
            end
            else if(nextAddrSel==2'b01)
            begin
                nextAddress <= addr;
            end
            else
            begin
                nextAddress <= 8'h1c;
            end
        end
        4'b0010:begin
            if(nextAddrSel==2'b00)
            begin
                nextAddress <= uPC+1;
            end
            else if(nextAddrSel==2'b01)
            begin
                nextAddress <= addr;
            end
            else
            begin
                nextAddress <= 8'h7;
```

```
        end
    end
    4'b0011:begin
        if(nextAddrSel==2'b00)
        begin
            nextAddress <= uPC+1;
        end
        else if(nextAddrSel==2'b01)
        begin
            nextAddress <= addr;
        end
        else
        begin
            nextAddress <= 8'hc;
        end

    end
    4'b0100:begin
        if(nextAddrSel==2'b00)
        begin
            nextAddress <= uPC+1;
        end
        else if(nextAddrSel==2'b01)
        begin
            nextAddress <= addr;
        end
        else
        begin
            nextAddress <= 8'h10;
        end

    end
    4'b0101:begin
        if(nextAddrSel==2'b00)
        begin
            nextAddress <= uPC+1;
        end
        else if(nextAddrSel==2'b01)
        begin
            nextAddress <= addr;
        end
        else
        begin
            nextAddress <= 8'h11;
        end

    end
    4'b0110:begin
        if(nextAddrSel==2'b00)
        begin
            nextAddress <= uPC+1;
        end
        else if(nextAddrSel==2'b01)
        begin
            nextAddress <= addr;
        end
        else
        begin
            nextAddress <= 8'h12;
        end
```

```
        end
        4'b0111:begin
            if(nextAddrSel==2'b00)
            begin
                nextAddress <= uPC+1;
            end
            else if(nextAddrSel==2'b01)
            begin
                nextAddress <= addr;
            end
            else
            begin
                nextAddress <= 8'h13;
            end

        end
        4'b1000:begin
            if(nextAddrSel==2'b00)
            begin
                nextAddress <= uPC+1;
            end
            else if(nextAddrSel==2'b01)
            begin
                nextAddress <= addr;
            end
            else
            begin
                nextAddress <= 8'h16;
            end
        end
        4'b1001:begin
            if(nextAddrSel==2'b00)
            begin
                nextAddress <= uPC+1;
            end
            else if(nextAddrSel==2'b01)
            begin
                nextAddress <= addr;
            end
            else
            begin
                nextAddress <= 8'h19;
            end
        end
        4'b1010:begin
            if(nextAddrSel==2'b00)
            begin
                nextAddress <= uPC+1;
            end
            else if(nextAddrSel==2'b01)
            begin
                nextAddress <= addr;
            end
            else
            begin
                nextAddress <= 8'h4;
            end
        end
        default: begin
            if(nextAddrSel==2'b00)
            begin
```

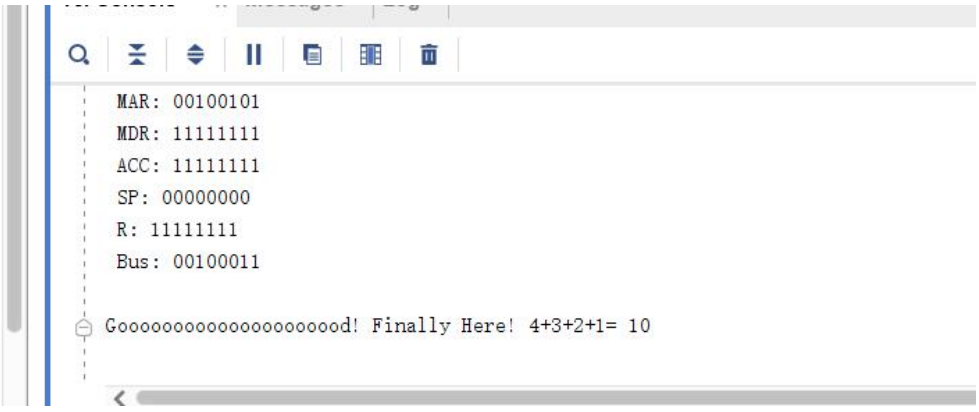
```
                nextAddress <= uPC+1;
            end
            else if(nextAddrSel==2'b01)
            begin
                nextAddress <= addr;
            end
            else
            begin
                nextAddress <= 8'h0;
            end
        end
    endcase
```

通过时序电路，改变 uPC 的值，维持一时钟周期的来自控制存储器的微指令输出，即输出各种信号

```
always@(posedge clk)
begin
    if(reset)
    begin
        uPC<=addr;
    end
    else if (condJMP==1&&condSel==0&&ZF==0) begin//jz,但是不为 0,就回到取指
        uPC<=addr;
    end
    else if(condJMP==1&&condSel==1&&ZF==1)begin//jnz,但是为 0,就回到取指
        uPC<=addr;
    end
    else
    begin
        uPC<=nextAddress;
    end
end
```

三、 调试报告

出现的问题:
本地显示通过,但是在 OJ 上说代码编译错误



分析原因及解决方案:
由于 OJ 上说是代码编译错误，所以首先排查是代码的语法规范问题

于是经过测试发现是语法问题.

```
    if()
    else if()
    else
    begin
        case ()
        endcase
    end
end
```

由于我为了节省代码量,在 ifelse 里面嵌套了 case 语句。在实验的指导手册里没有提到不能嵌套使用这个，所以我一开始没有发现。

后来我把 ifelse 语句放到 case 里面就通过了。但是会导致生成的电路较为复杂。

四、 总结及实验课程感想

在实验刚开始的时候总是对本次实验要做的一头雾水，不知道要干嘛。看了一遍指导书也不是很明白。然后就只能一遍一遍看，然后再结合课本上的理论知识，再看几遍，就能大概知道本次实验要做什么。光知道要做什么也还不够，像 BOOTH 算法那个实验，不能很好地结合时序和逻辑电路，就会导致某个信号慢了一个周期，两个接口的信号速度不匹配的情况。这个时候就需要继续琢磨如何利用好组合电路。

这几个实验基本来说只要有了思路就能实现一个基础版的电路，重点还是在于调

试。找到自己错误的地方在哪里是最为重要的。

计算机组成原理完结了，但是后面还有计算机设计与实践，冲冲冲！这一切还远没有结束！