***2019***



**计算机组成原理 ·实验报告·**

j0242087[1]

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS1706 |
| 学 号： | U201714762 |
| 姓 名： | 梁一飞 |
| 电 话： | 15387221573 |
| 邮 件： | [349062884@qq.com](mailto:13456@qq.com) |
| 完成日期： | 2019-12-18 |



目 录

[1 CPU设计实验 2](#_Toc27667711)

[1.1 设计要求 2](#_Toc27667712)

[1.2 方案设计 2](#_Toc27667713)

[1.3 实验步骤 10](#_Toc27667714)

[1.4 故障与调试 16](#_Toc27667715)

[1.5 测试与分析 17](#_Toc27667716)

[2 总结与心得 19](#_Toc27667717)

[2.1 实验总结 19](#_Toc27667718)

[2.2 实验心得 19](#_Toc27667719)

[参考文献 20](#_Toc27667720)

# CPU设计实验

## 设计要求

构建一个32位MIPS CPU处理器，包括单周期硬布线CPU、多周期微程序CPU以及多周期硬布线CPU，该处理器应支持核心指令集中列出的所有指令，见表1.1，。具体指令功能参见附件中的MIPS标准文档。最终设计完成的CPU应能运行标准测试程序。

表 1.1 核心指令集

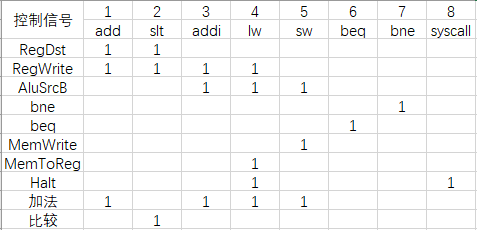
|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **指令** | **格式** | **备注** |
| 1 | Add | add $rd, $rs, $rt | 指令功能及指令格式  参考MIPS32指令集 |
| 2 | Add Immediate | addi $rt, $rs, immediate |
| 3 | Load Word | lw $rt, offset($rs) |
| 4 | Store Word | sw $rt, offset($rs) |
| 5 | Branch on Equal | beq $rs, $rt, label |
| 6 | Branch on Not Equal | bne $rs, $rt, label |
| 7 | Set Less Than | slt $rd, $rs, $rt |
| 8 | syscall（display or exit） | syscall | 系统调用，用于停机 |

## 方案设计

### 单周期硬布线CPU

（1）控制器信号表

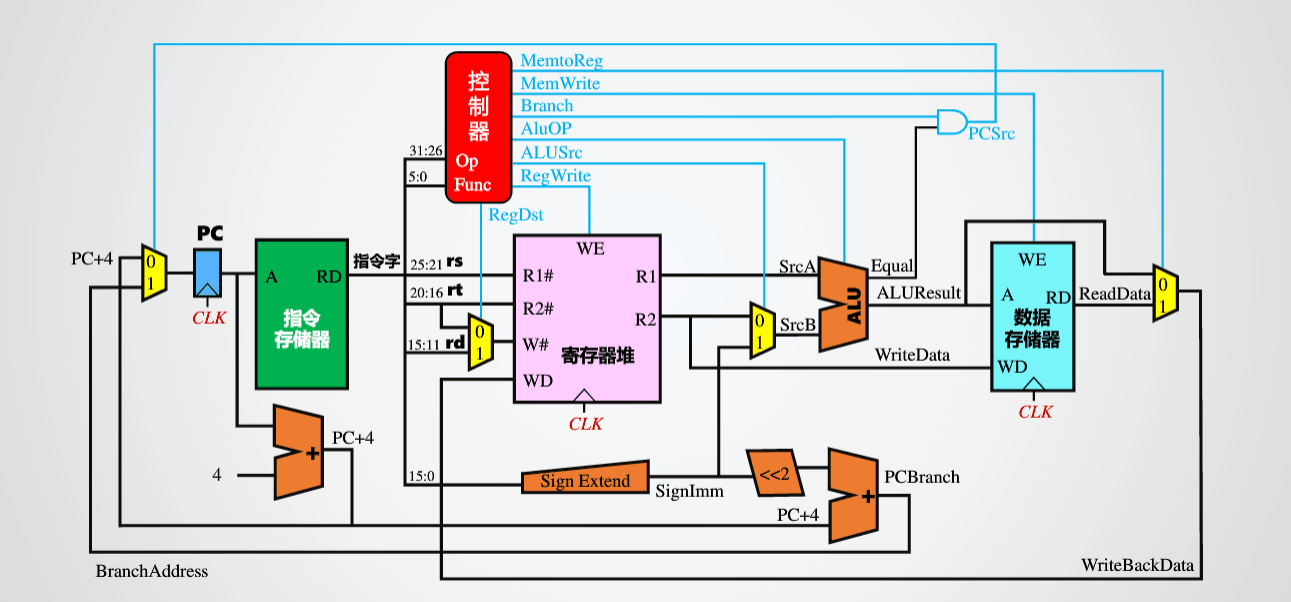
表 1.2 单周期硬布线控制器信号表



1. RegDst为1表示RegiFile写回地址由R型指令rd字段给出，否则由I型指令rt字段给出；
2. RegWrite为1表示数据写回RegiFile；
3. AluSrcB为1表示ALU的第二个操作数将由立即数扩展器给出，否则由RegiFile的第二个输出给出；
4. bne为1表示指令bne；
5. beq为1表示指令beq
6. MemWrite为1表示将ALU运算结果写入数据存储器DM；
7. MemToReg为1表示从数据存储器DM中选数据送入RegiFile；
8. Halt为1时系统停机；

（2）数据通路

图1.1.单周期硬布线CPU数据通路



### 多周期与单周期的差异

1. 不再区分指令存储器与数据存储器，指令和数据保存在同一个存储器中；
2. 部分功能单元可以在一条指令执行过程的不同周期中多次使用，如ALU；
3. 主要功能单元输出端都增加了寄存器，在后续时钟周期中要用到的所有数据需存储在相应的寄存器中，如数据寄存器DR用于存放从存储器读取的数据，指令存储器IR用于存放从存储器读出的指令；
4. ALU输出结果有三种情况：

1.分支目标地址，由beq、bne指令给出，该地址将被写到PC中；

2.指令的运算结果，将被写入到寄存器文件中；

3.为存储器访问指令lw和sw提供存储器地址；

1. PC作为指令计数器不再仅由时钟周期控制，增加了专门的写操作控制信号；

### 多周期控制信号

多周期CPU控制下，不同指令对应不同的时钟周期数，故无法像单周期CPU一次性给出一条指令的所有控制信号，而每一条指令的执行又可以拆解为“取指->译码->执行”三个阶段，其取指和译码两个阶段所有指令对应的数据通路相同，故只需分析每条指令执行阶段需要的时钟周期及每个时钟周期下所有的控制信号。每条指令的执行流程表如下所示：

表 1.4 取指阶段操作流程

|  |  |
| --- | --- |
| **指令阶段** | **操作流程** |
| 取指令 | IR<-(MEM[PC])  PC<-(PC)+4 |
| 译码及取操作数 | A<-(R[IR[25:21]])  B<-(R[IR[20:16]])  C<-(PC)+(S-EXT(IR[15:0])<<2) |

表 1.5 add指令执行操作流程

|  |  |
| --- | --- |
| **指令阶段** | **操作流程** |
| 加运算 | C<-(A)+(B) |
| 写回 | R[IR[15:11]]<-(C) |

表 1.6 slt指令执行操作流程

|  |  |
| --- | --- |
| **指令阶段** | **操作流程** |
| 比较运算 | C<-((A)<(B)) |
| 写回 | R[IR[15:11]]<-(C) |

表 1.7 lw指令执行操作流程

|  |  |
| --- | --- |
| **指令阶段** | **操作流程** |
| 计算地址 | C<-(A)+ S-EXT(IR[15:0]) |
| 访存 | DR<-(MEM[PC]) |
| 写回 | R[IR[20:16]]<-(DR) |

表 1.8 sw指令执行操作流程

|  |  |
| --- | --- |
| **指令阶段** | **操作流程** |
| 计算地址 | C<-(A)+ S-EXT(IR[15:0]) |
| 访存 | DR<-(MEM[PC]) |

表 1.9 beq指令执行操作流程

|  |  |
| --- | --- |
| **指令阶段** | **操作流程** |
| 送目标地址 | If(A==B)PC<-(C) |

表 1.10 bne指令执行操作流程

|  |  |
| --- | --- |
| **指令阶段** | **操作流程** |
| 送目标地址 | If(A!=B)PC<-(C) |

表 1.11 addi指令执行操作流程

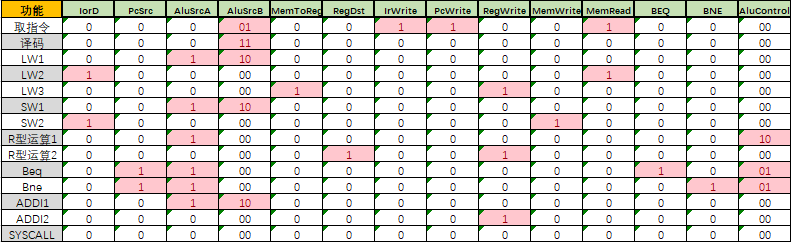
|  |  |
| --- | --- |
| **指令阶段** | **操作流程** |
| 加运算 | C<-(A)+ S-EXT(IR[15:0]) |
| 写回 | R[IR[20:16]]<-(C) |

表 1.12 syscall指令执行操作流程

|  |  |
| --- | --- |
| **指令阶段** | **操作流程** |
| 空操作，停机 | 锁住PC |

综上所述，给出所有指令不同阶段的控制信号表，如表1.13所示：

表 1.13 指令控制信号表



### 多周期微程序CPU

（1）状态编码

8条指令共计13个执行阶段，采用4位二进制编码表示微指令地址，如下表所示：

表 1.14 状态编码表

|  |  |  |
| --- | --- | --- |
| 微指令 | 状态 | 地址 |
| 取指令 | S0 | 0 |
| 译码 | S1 | 1 |
| LW1 | S2 | 2 |
| LW2 | S3 | 3 |
| LW3 | S4 | 4 |
| SW1 | S5 | 5 |
| SW2 | S6 | 6 |
| R型运算 | S7 | 7 |
| R型运算 | S8 | 8 |
| Beq | S9 | 9 |
| Bne | S10 | 10 |
| ADDI1 | S11 | 11 |
| ADDI2 | S12 | 12 |
| SYSCALL | S13 | 13 |

（2）微指令地址转移逻辑

采用下址字段法，配合判断状态P，取指令时P为1，表示控制存储器的地址由指令的第一个阶段的微程序地址给出，执行指令时P为0，表明下一条微指令地址由下址字段给出，地址转移逻辑如下表所示：

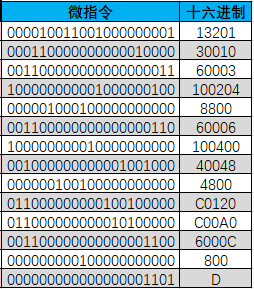
表 1.15 微指令地址转移逻辑



（3）控制存储器

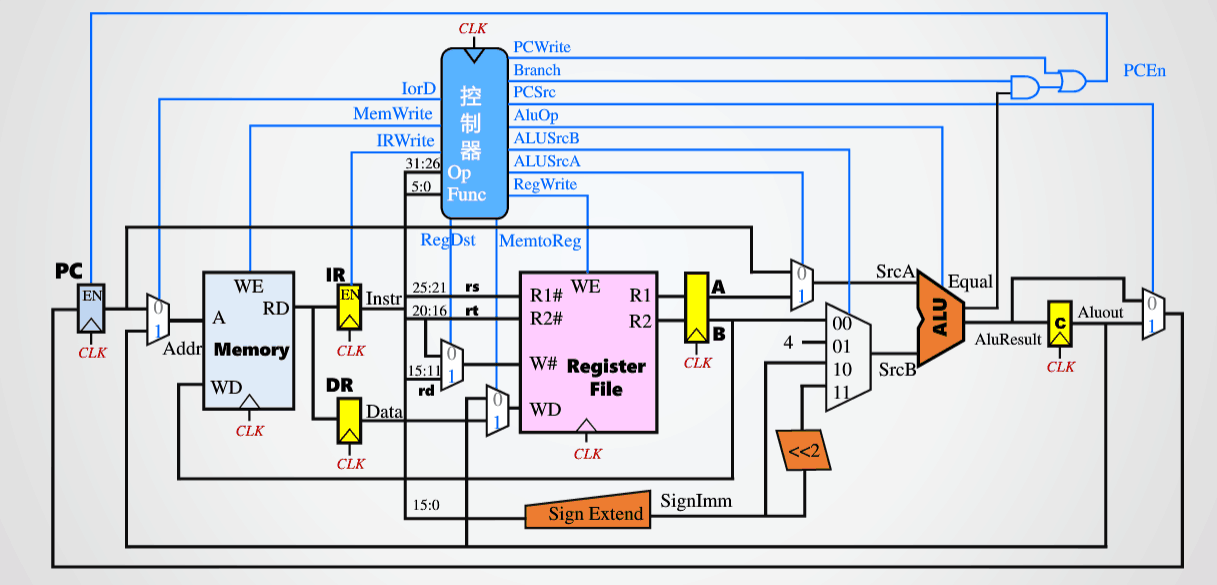
执行8条指令的控制信号需要16位编码，采用下址字段法需编码位数和微指令地址位数相同，即4位。再加上一位判断标志位P，故微指令编码21位，地址位编码4位。编码产生的微指令如下表所示：

表 1.16 微指令编码



（4）数据通路

图1.2.多周期微程序CPU数据通路



### 多周期硬布线CPU

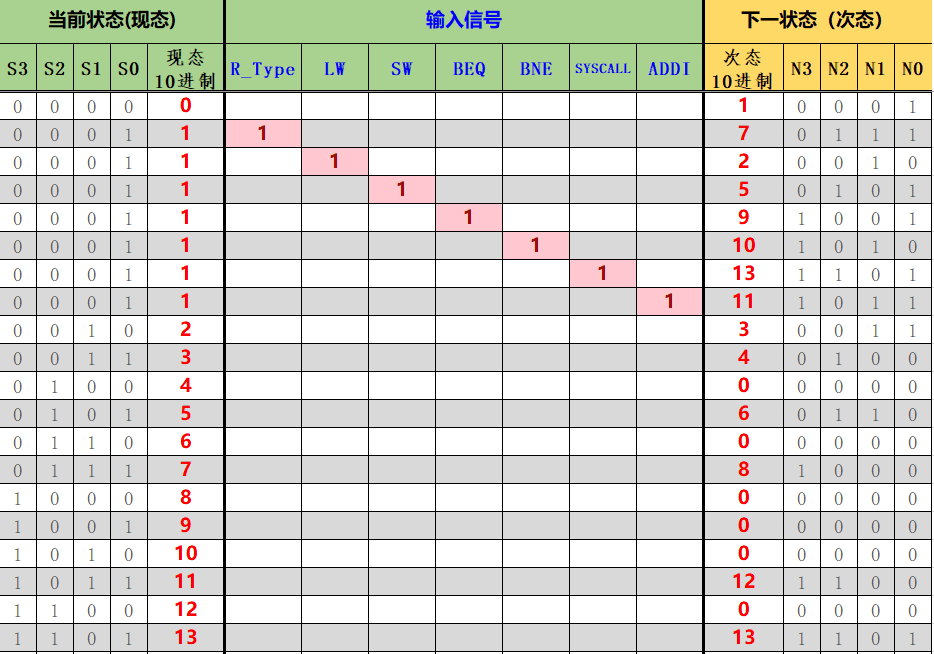
（1）状态编码

同多周期微程序CPU

（2）状态转移

用有限状态机FSM实现现态到次态的转换。FSM采用纯组合逻辑电路实现，FSM状态转换表如下：

表 1.17 FSM状态转换

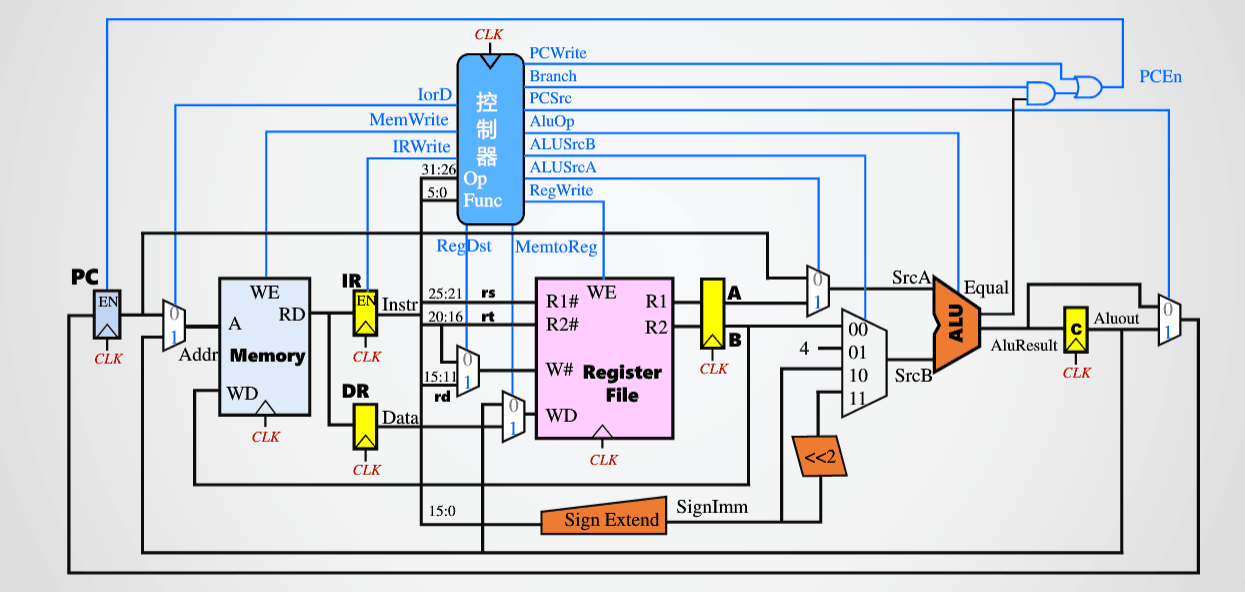


（3）控制存储器

地址编码4位，由于状态转移由FSM给出，故无需下址字段和判断字段P，减少为16位二进制编码。不过同样可采用21位二进制编码，不用下址字段和P，从而硬布线控制存储器与微程序控制存储器相同（表1.16）。

（4）数据通路

图1.3.多周期硬布线CPU数据通路



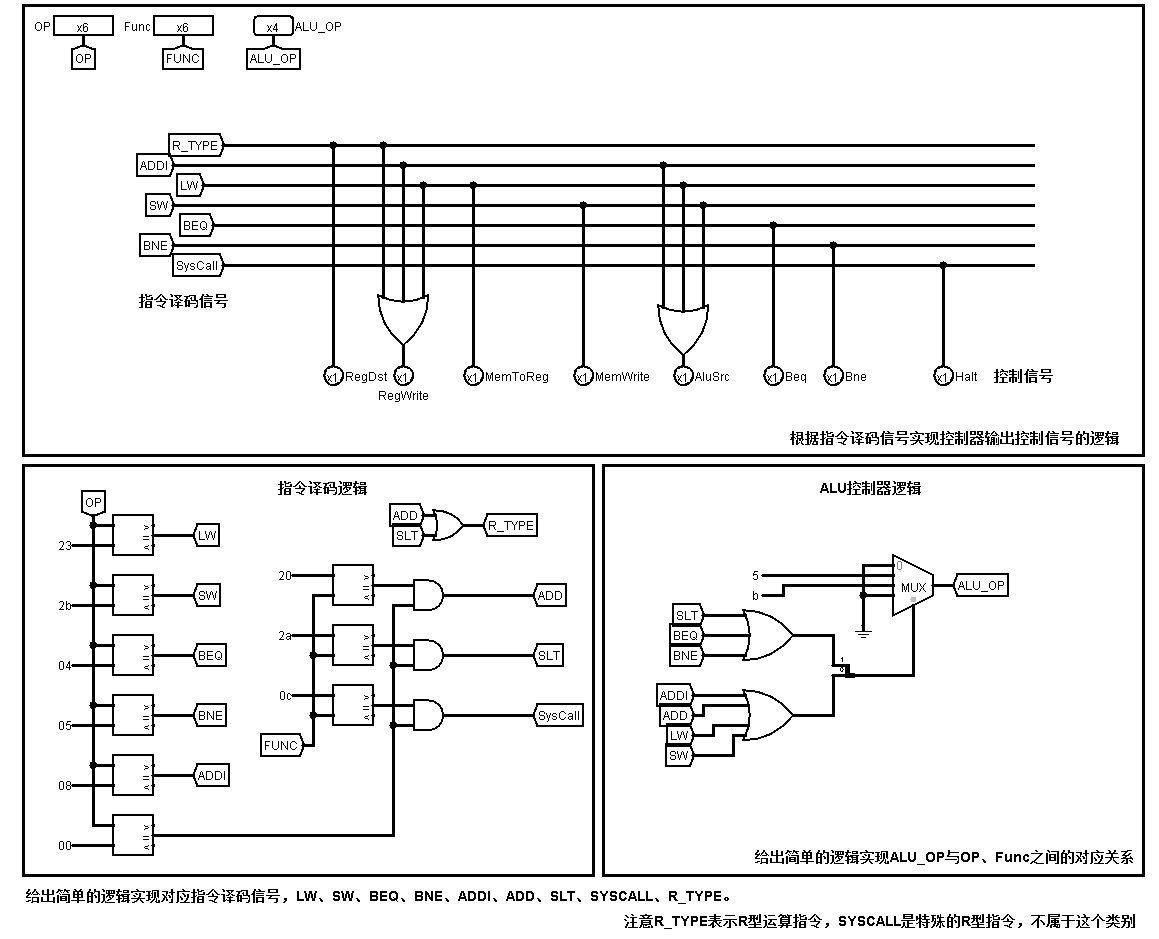
## 实验步骤

### 单周期硬布线CPU

1. 数据通路综合

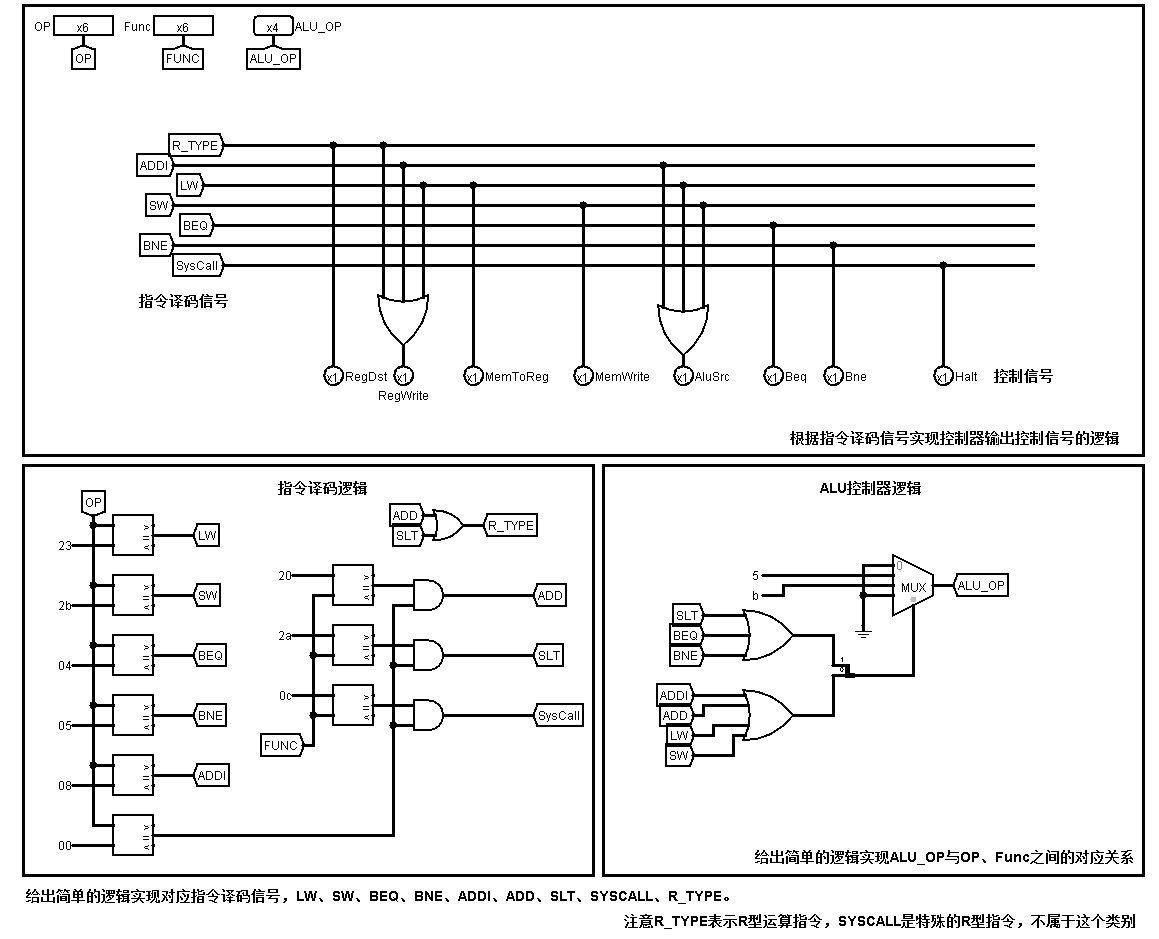
根据方案设计里所列出的功能部件、多路选择器控制信号、运算操作选择的产生条件绘制单周期硬布线控制器电路如下：

图1.4.单周期硬布线控制电路



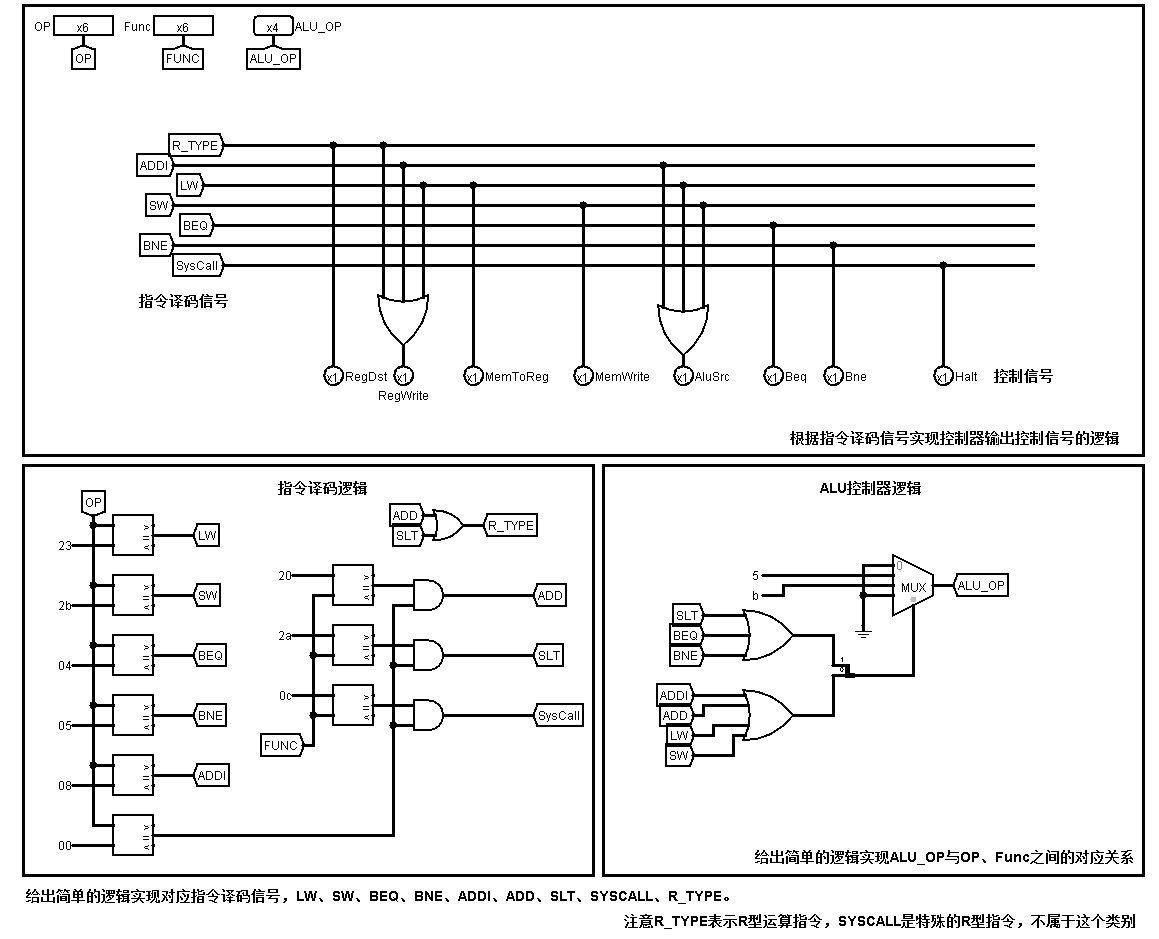
指令译码逻辑如下：

图1.5.单周期硬布线指令译码逻辑



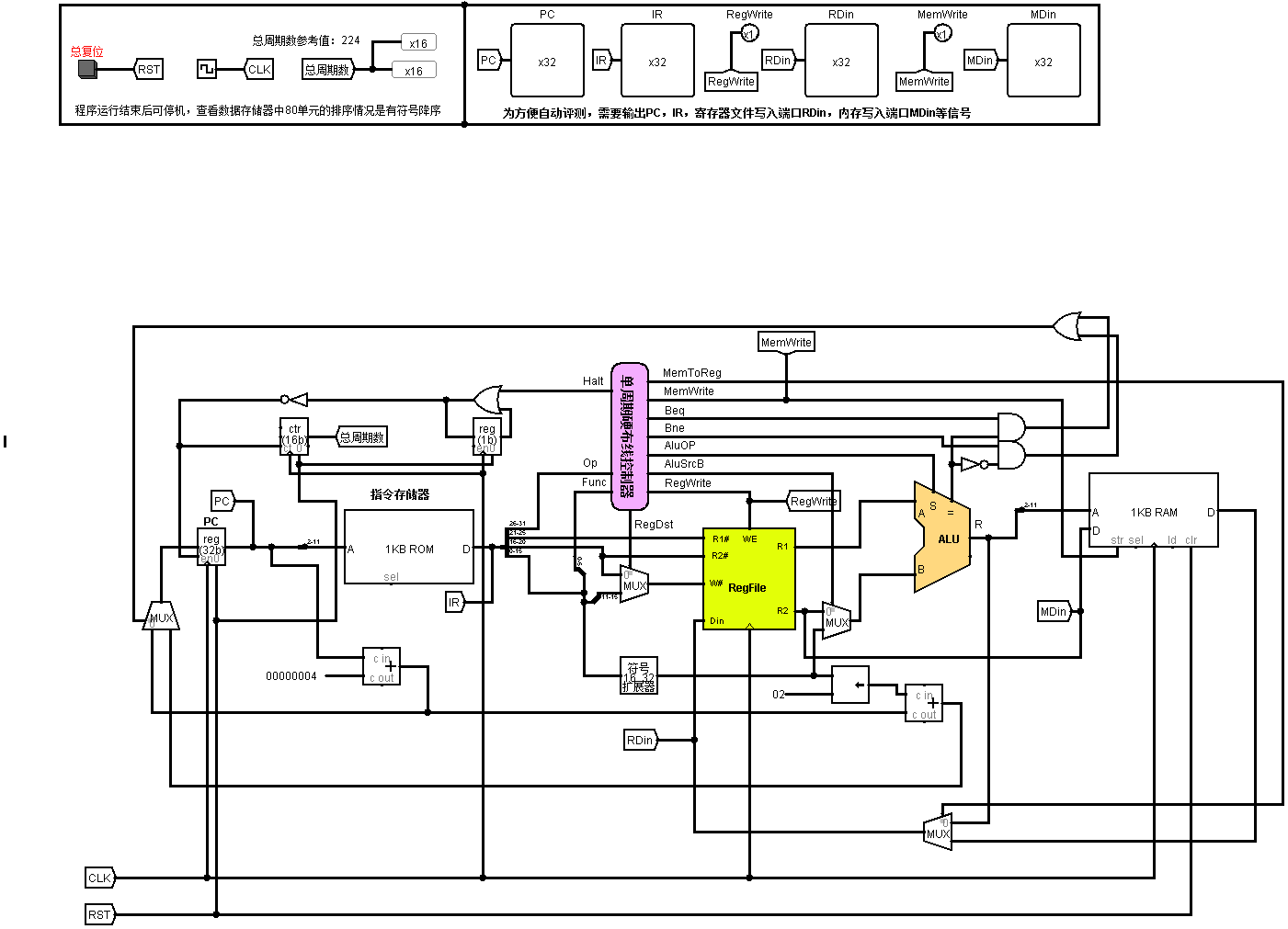
ALU控制器逻辑如下：

图1.6.单周期硬布线ALU控制器逻辑



1. 综合以上，根据数据通路绘制最终电路如下

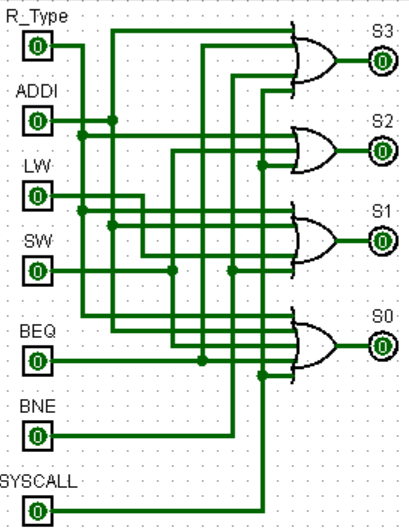
图1.7.单周期硬布线CPU



### 多周期微程序CPU

(1)根据微程序地址转移逻辑表，生成微程序地址转移逻辑电路如下：

图1.8.微程序地址转移逻辑



生成的地址转移逻辑表达式为：

S3 = ADDI + BEQ + BNE + SYSCALL；

S2 = R\_Type + SW + SYSCALL；

S1 = R\_Type + ADDI + LW + BNE；

S0 = R\_Type + ADDI + SW + BEQ + SYSCALL。

（2）绘制控制电路、指令译码逻辑电路、ALU控制器逻辑电路如下：

图1.9.微程序控制电路

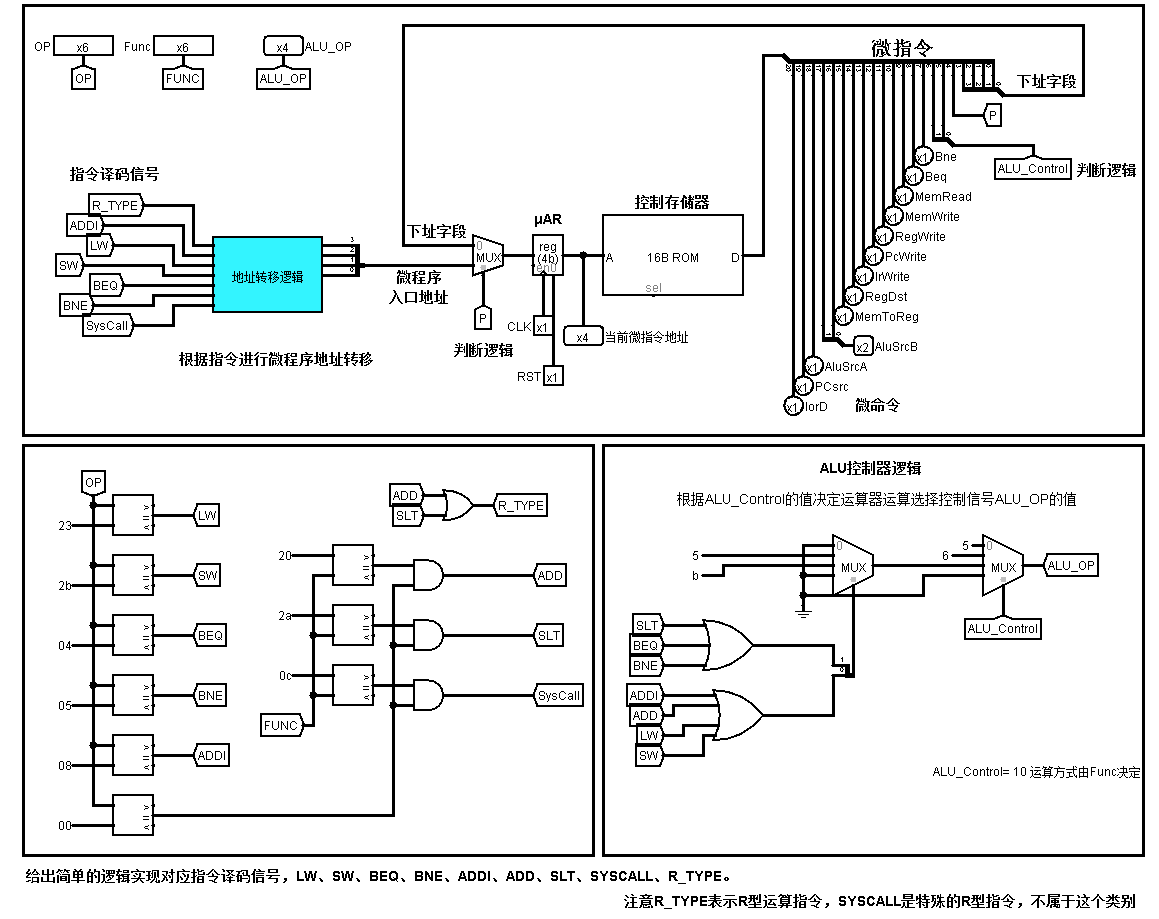


图1.10.微程序指令译码逻辑电路

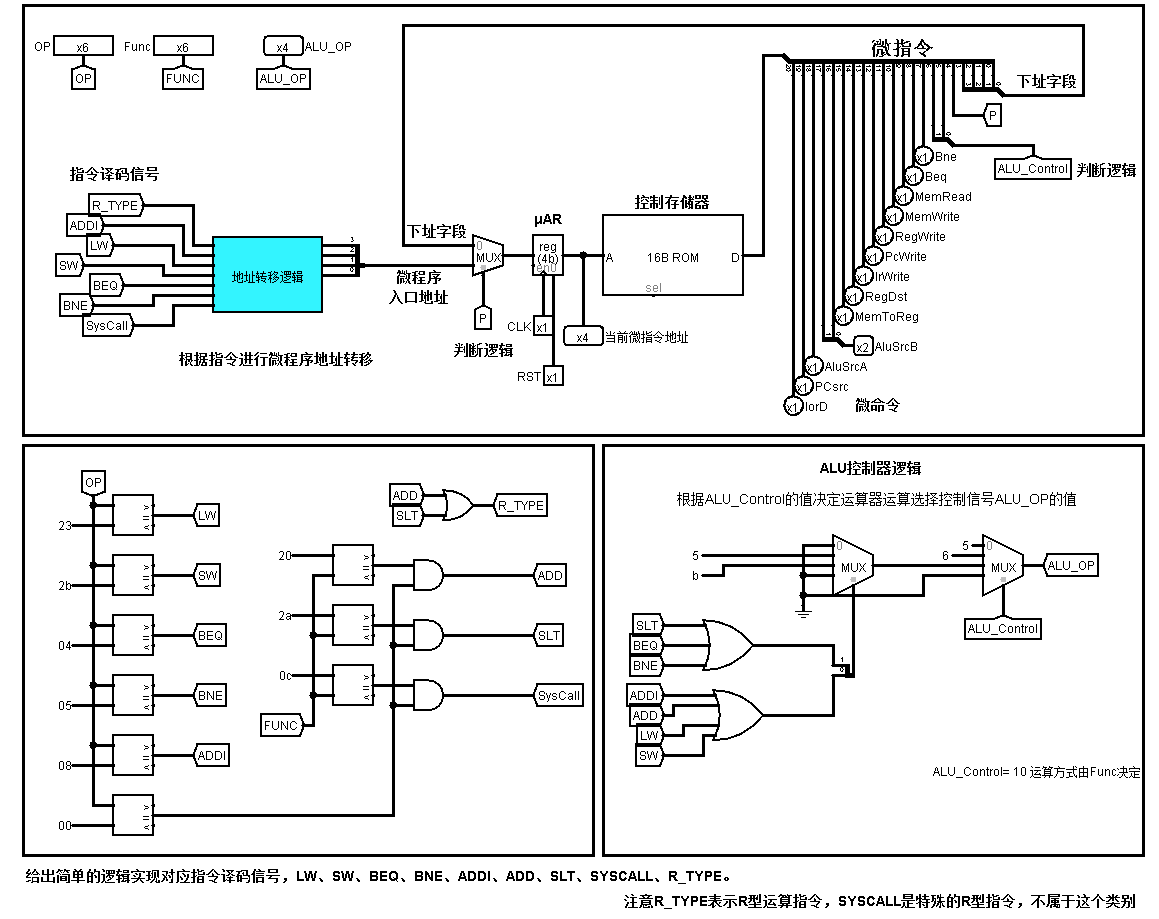
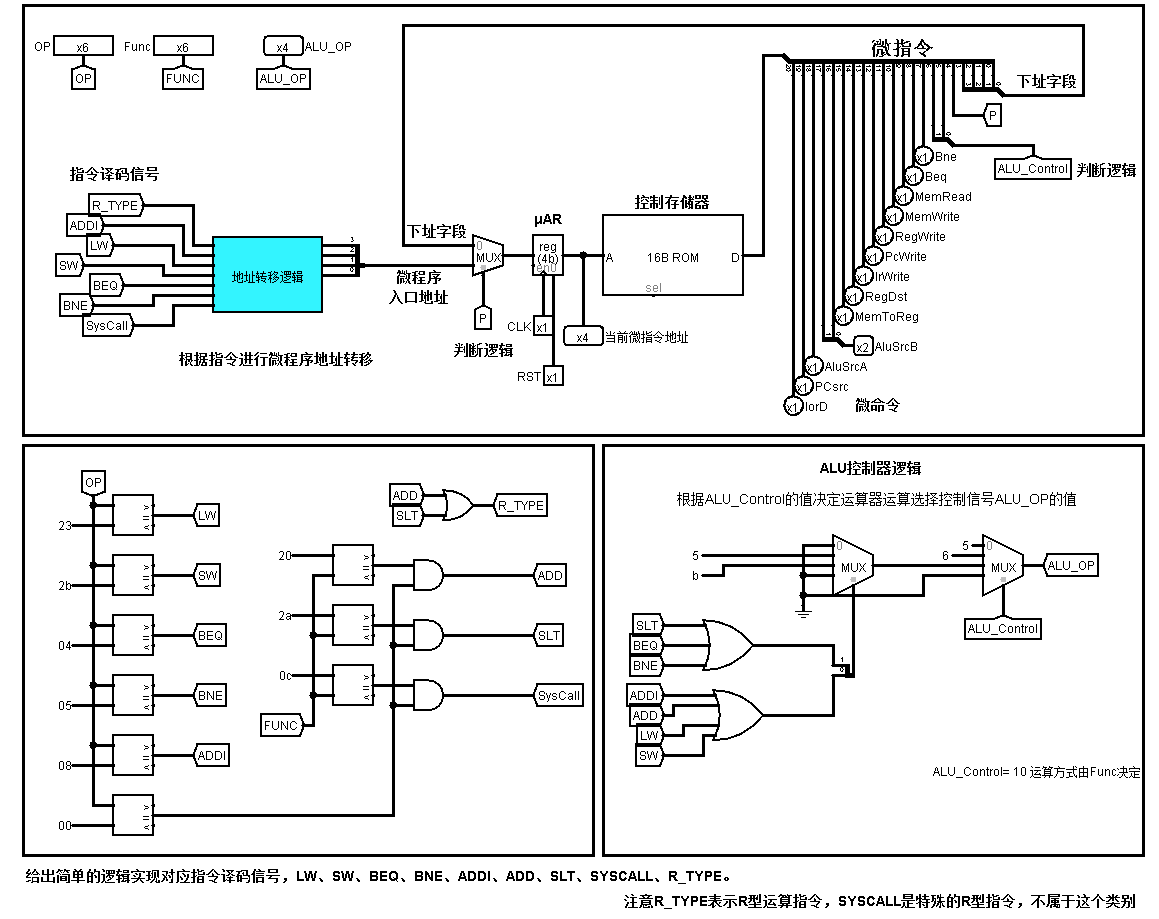
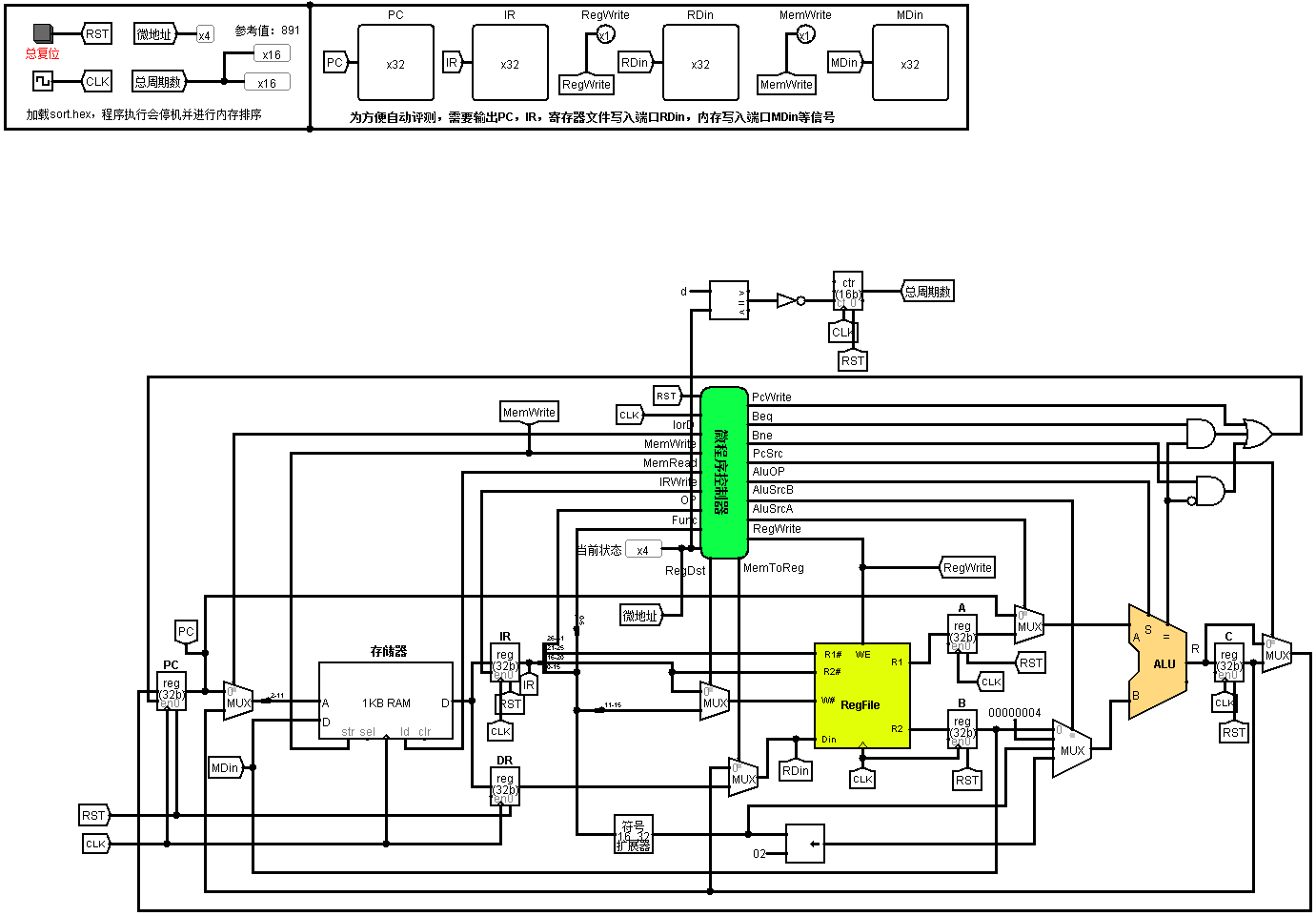


图1.11.微程序ALU控制逻辑电路



（3）综合以上，根据数据通路画出多周期微程序CPU电路如下：

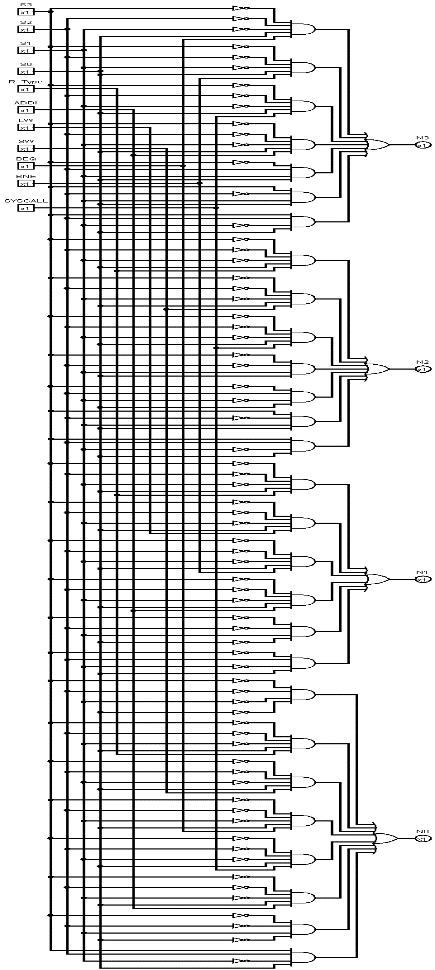
图1.12.多周期微程序CPU电路



### 多周期硬布线CPU

（1）根据FSM状态转移表，生成状态机FSM电路如下：

图1.13.状态机FSM



（2）绘制控制电路、指令译码逻辑电路、ALU控制器逻辑电路如下：

图1.14.硬布线控制电路

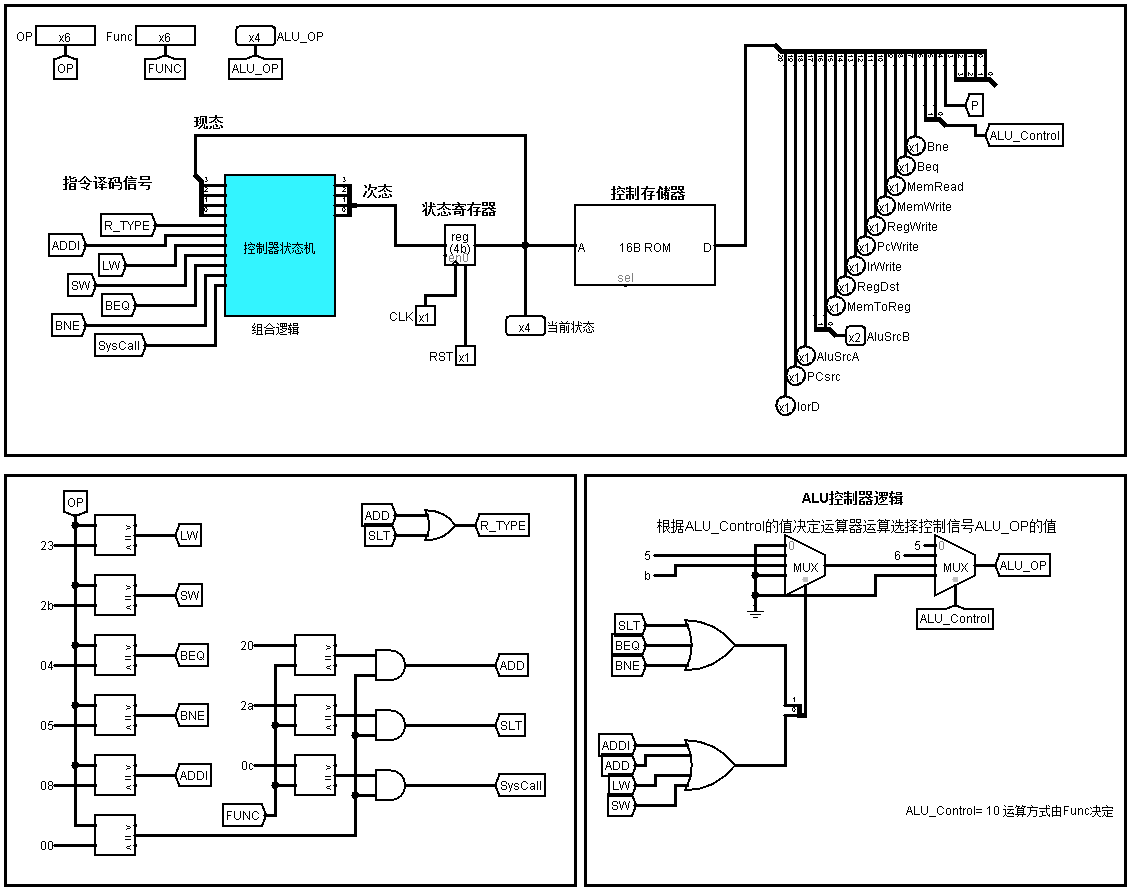


图1.15.硬布线指令译码逻辑电路

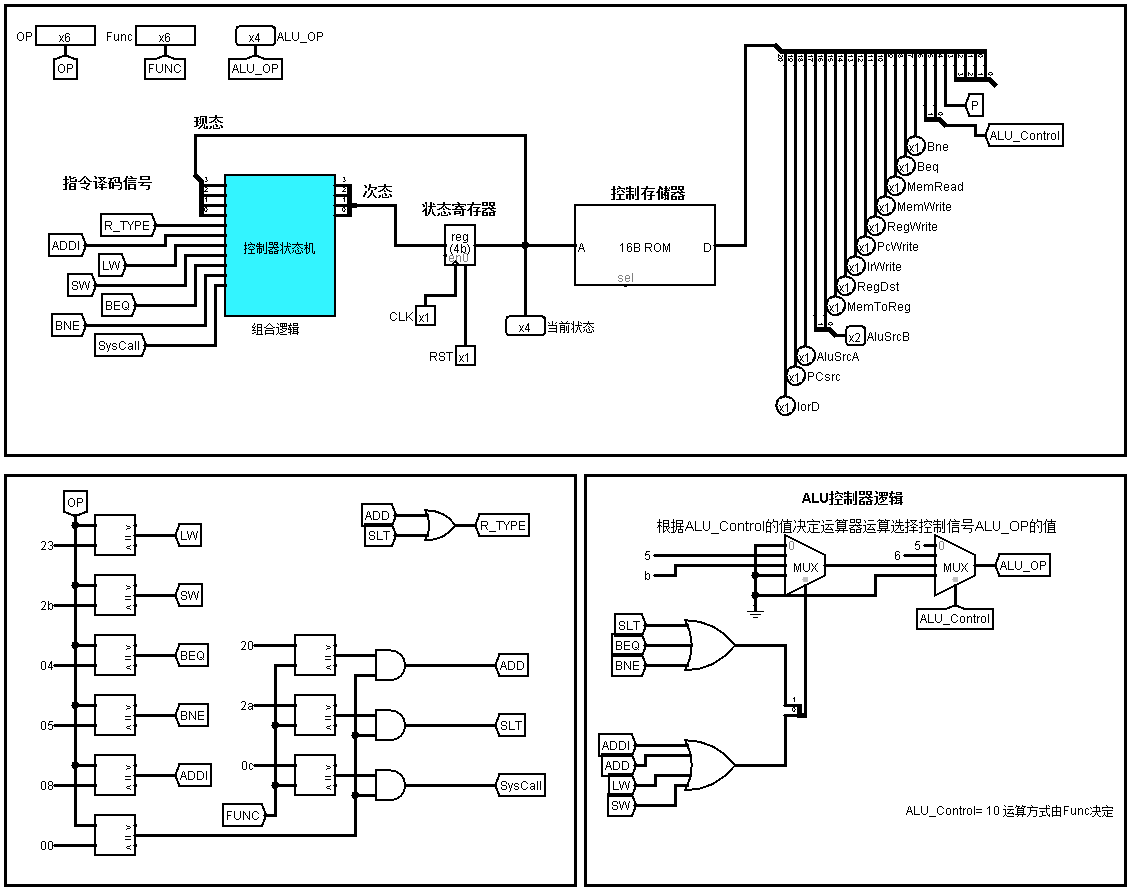
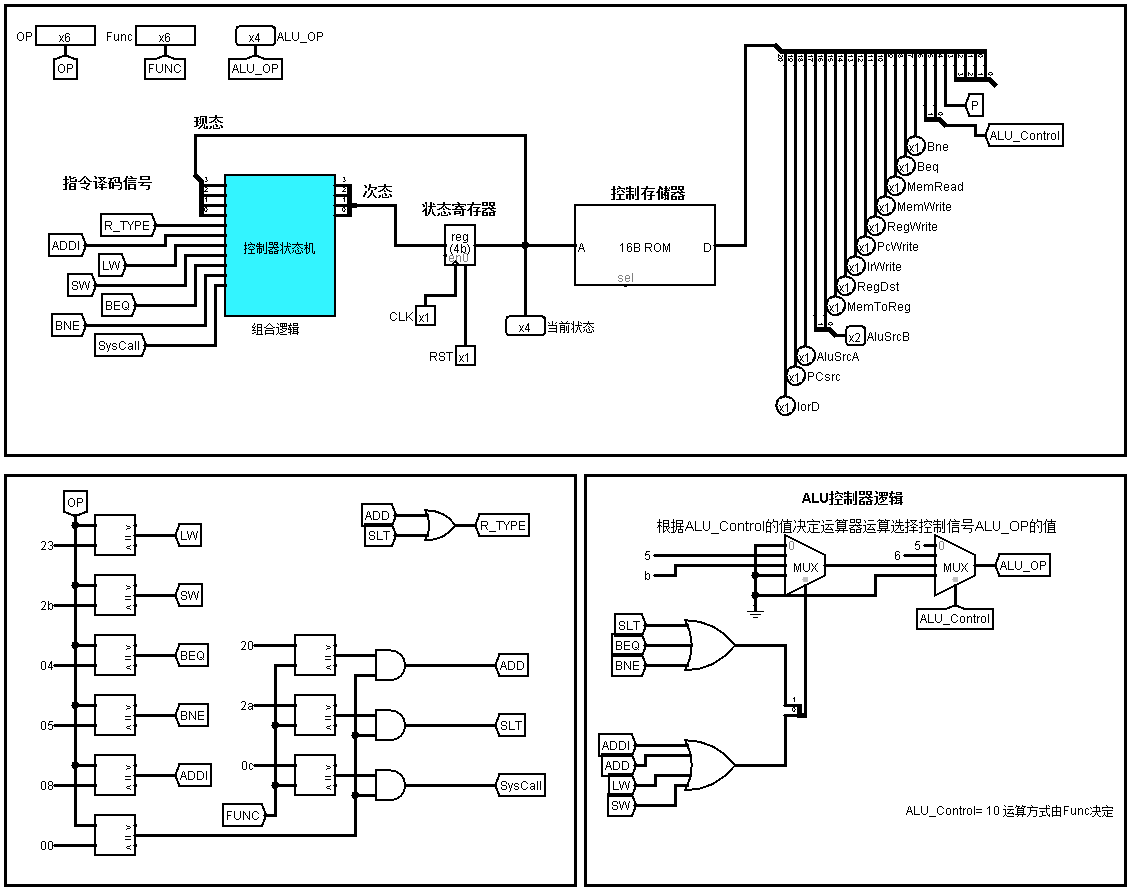
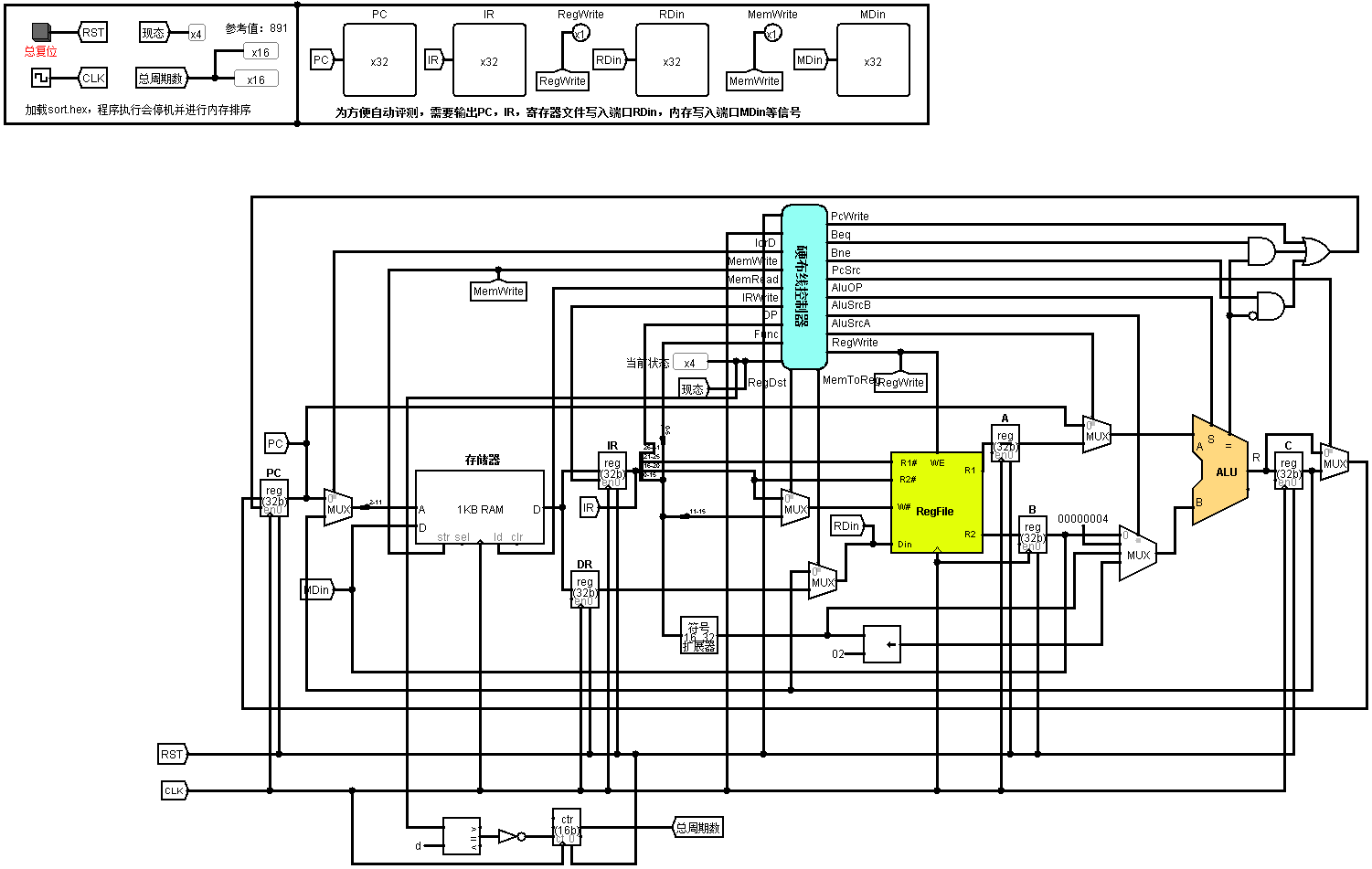


图1.16.硬布线ALU控制器逻辑电路



（3）综合以上，结合数据通路绘制多周期硬布线CPU电路如下：

图1.17.多周期硬布线CPU电路



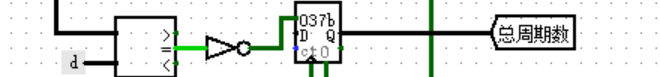
## 故障与调试

### 多周期CPU周期计数问题

**故障现象：**多周期CPU运行sort.hex程序时，能正确得到相应的内存布局，但是总周期计数无法停下。

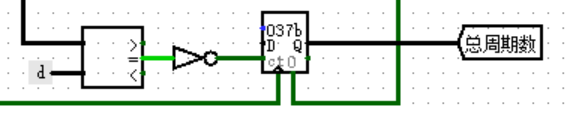
**原因分析：**如图所示，计数器连接错误：

图1.18.错误连接



**解决方案：**修改连线如图所示：

图1.19.正确连接



## 测试与分析

### 单周期CPU执行sort.hex

（1）内存布局

在80号单元处出现6,5,4,3,2,1,ffff的有符号降序数据如图所示：

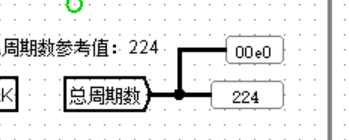
图1.20.单周期执行sort.hex后内存布局



（2）时钟周期

执行完毕后，系统停机，时钟周期数为224，如图所示：

图1.21.单周期执行sort.hex后时钟周期



### 多周期微程序CPU执行sort.hex

（1）内存布局

在80号单元处出现6,5,4,3,2,1,ffff的有符号降序数据如图所示：

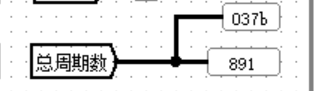
图1.22.多周期微程序CPU执行sort.hex后内存布局



2）时钟周期

执行完毕后，系统停机，时钟周期数为891，如图所示：

图1.22.多周期微程序CPU执行sort.hex后时钟周期



### 1.5.3 多周期硬布线CPU执行sort.hex

（1）内存布局

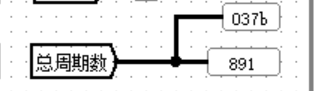
在80号单元处出现6,5,4,3,2,1,ffff的有符号降序数据如图所示：

图1.23.多周期硬布线CPU执行sort.hex后内存布局



2）时钟周期

执行完毕后，系统停机，时钟周期数为891，如图所示：

图1.24.多周期硬布线CPU执行sort.hex后时钟周期

# 总结与心得

## 实验总结

本次实验主要完成了以下工作：

1.实现了MIPS单周期CPU控制存储器

2.实现了MIPS多周期CPU控制存储器；

3.实现了MIPS多周期CPU微程序地址转移逻辑；

4.实现了MIPS多周期CPU硬布线控制器状态机；

5.实现了MIPS CPU数据通路综合，最终的CPU功能正确。

## 实验心得

通过本次实验，我学会了使用工程化方法生成电路，掌握了logism电路分析操作方法。熟悉了MIPS核心8条指令，并对MIPS的三类指令有了大致的实现思路。掌握了CPU的运行机理以及模块划分，并学会了准确生成每一个控制信号。同时通过实验，也对课程内容有了更深的理解，在实验过程中，也温习了课程知识。在本次实验中，给我帮助最大的是谭老师的MOOC课堂，通过观看学习，让我对实验内容有了宏观上的把握，大大提高了实验效率。对于实验的建议就是，老师对实验的通知及要求可以发到钉钉上，如果直接发到qq群里，很容易被同学的消息所覆盖，从而导致部分同学不清楚具体要求，造成不必要的麻烦。

# 参考文献

1. DAVID A.PATTERSON(美).计算机组成与设计硬件/软件接口(原书第5版).北京:机械工业出版社.
2. David Money Harris(美).数字设计和计算机体系结构（第二版）. 机械工业出版社
3. 谭志虎,秦磊华,胡迪青.计算机组成原理实践教程.北京:清华大学出版社，2018年.
4. 秦磊华，吴非，莫正坤.计算机组成原理. 北京:清华大学出版社，2011年.
5. 袁春风编著. 计算机组成与系统结构. 北京:清华大学出版社，2011年.
6. 张晨曦，王志英. 计算机系统结构. 高等教育出版社，2008年.

|  |
| --- |
| 一、原创性声明 |
| 本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。  特此声明！  **作者签字: 嵌入签名图片** |
| 二、对课程实验的学术评语（教师填写） |
|  |
| 三、对课程实验的评分（教师填写） |
| |  |  |  |  | | --- | --- | --- | --- | | 评分项目  （分值） | 报告撰写  （30分） | 课设过程  （70分） | 最终评定  （100分） | | 得分 |  |  |  | |
| **指导教师签字:** |