**计算机组成原理**

**课内大作业报告**

**学 号\_\_\_\_\_\_\_\_20074221\_\_\_\_\_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_游佳慧\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_ 魏坚华 \_\_\_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_\_\_2022.5.8\_\_\_\_\_\_\_\_\_\_\_\_**

**成绩评价表**

|  |  |  |
| --- | --- | --- |
| **报告内容** | **报告结构** | **报告最终成绩** |
| **□丰富正确**  **□基本正确**  **□有一些问题**  **□问题很大** | **□完全符合要求**  **□基本符合要求**  **□有比较多的缺陷**  **□完全不符合要求** |  |
| **报告与Project功能一致性** | **报告图表** | **总体评价** |
| **□完全一致**  **□基本一致**  **□基本不一致** | **□符合规范**  **□基本符合规范**  **□有一些错误**  **□完全不正确** |  |

**教师签字:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

目录

[一、总体数据通路结构设计 3](#_Toc102934832)

[1.1 总体数据通路结构图 3](#_Toc102934833)

[1.2 模块设计图 4](#_Toc102934834)

[二、模块定义 7](#_Toc102934835)

[2.1 IFU模块定义 7](#_Toc102934836)

[2.1.1 基本描述 7](#_Toc102934837)

[2.1.2 模块接口 7](#_Toc102934838)

[2.1.3 功能定义 7](#_Toc102934839)

[2.2 GPR模块定义 8](#_Toc102934840)

[2.2.1 基本描述 8](#_Toc102934841)

[2.2.2 模块接口 8](#_Toc102934842)

[2.2.3 功能定义 8](#_Toc102934843)

[2.3 ALU模块定义 9](#_Toc102934844)

[2.3.1 基本描述 9](#_Toc102934845)

[2.3.2 模块接口 9](#_Toc102934846)

[2.3.3 功能定义 9](#_Toc102934847)

[2.4 EXT模块定义 10](#_Toc102934848)

[2.4.1 基本描述 10](#_Toc102934849)

[2.4.2 模块接口 10](#_Toc102934850)

[2.4.3 功能定义 10](#_Toc102934851)

[2.5 DM模块定义 11](#_Toc102934852)

[2.5.1 基本描述 11](#_Toc102934853)

[2.5.2 模块接口 11](#_Toc102934854)

[2.5.3 功能定义 11](#_Toc102934855)

[2.6 Control模块定义 12](#_Toc102934856)

[2.6.1 基本描述 12](#_Toc102934857)

[2.6.2 模块接口 12](#_Toc102934858)

[2.6.3 功能定义 12](#_Toc102934859)

[三、设计的机器指令描述 13](#_Toc102934860)

[四、测试程序 14](#_Toc102934861)

[五、测试结果 15](#_Toc102934862)

[5.1 GPR运行结果 15](#_Toc102934863)

[5.2 DM运行结果 16](#_Toc102934864)

[六、总结与收获 17](#_Toc102934865)

# 一、总体数据通路结构设计

## 1.1 总体数据通路结构图

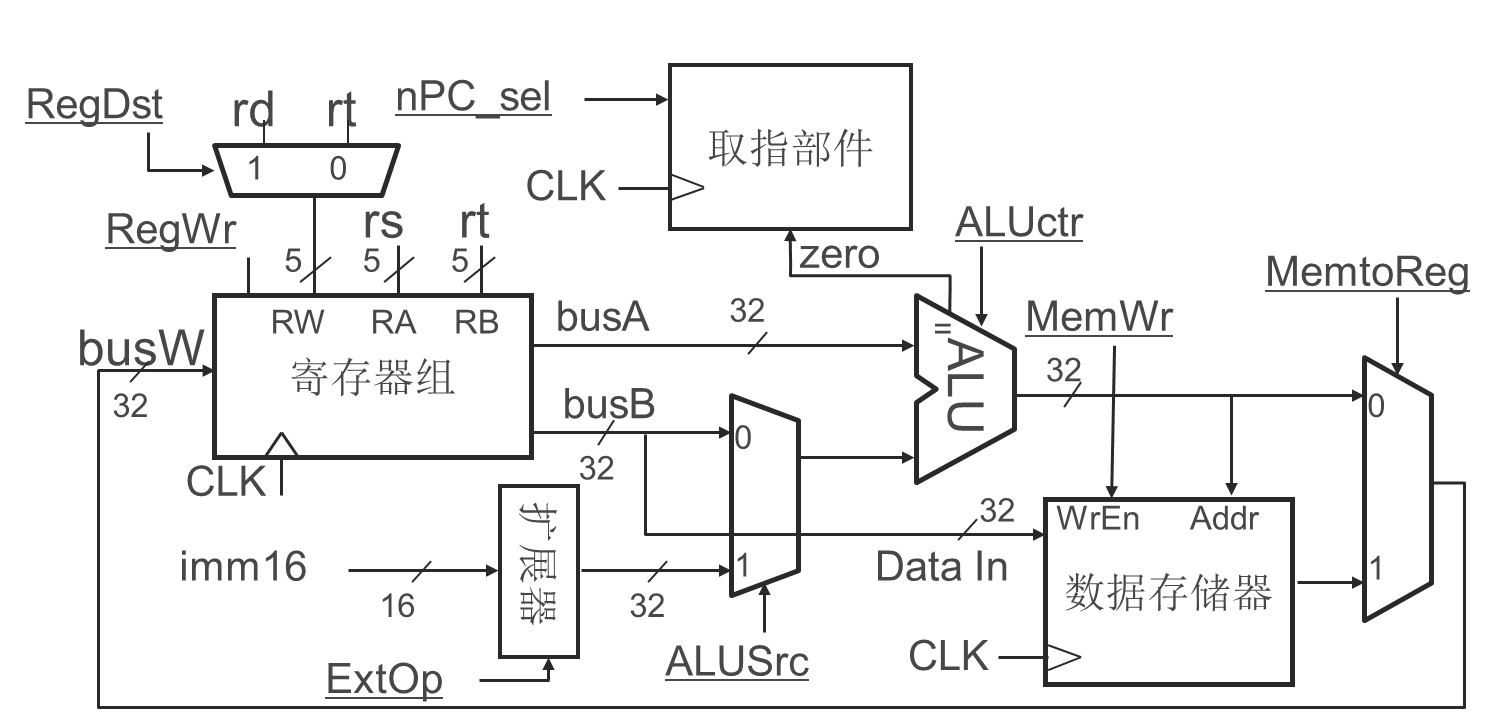


图 1：总体数据通路结构图

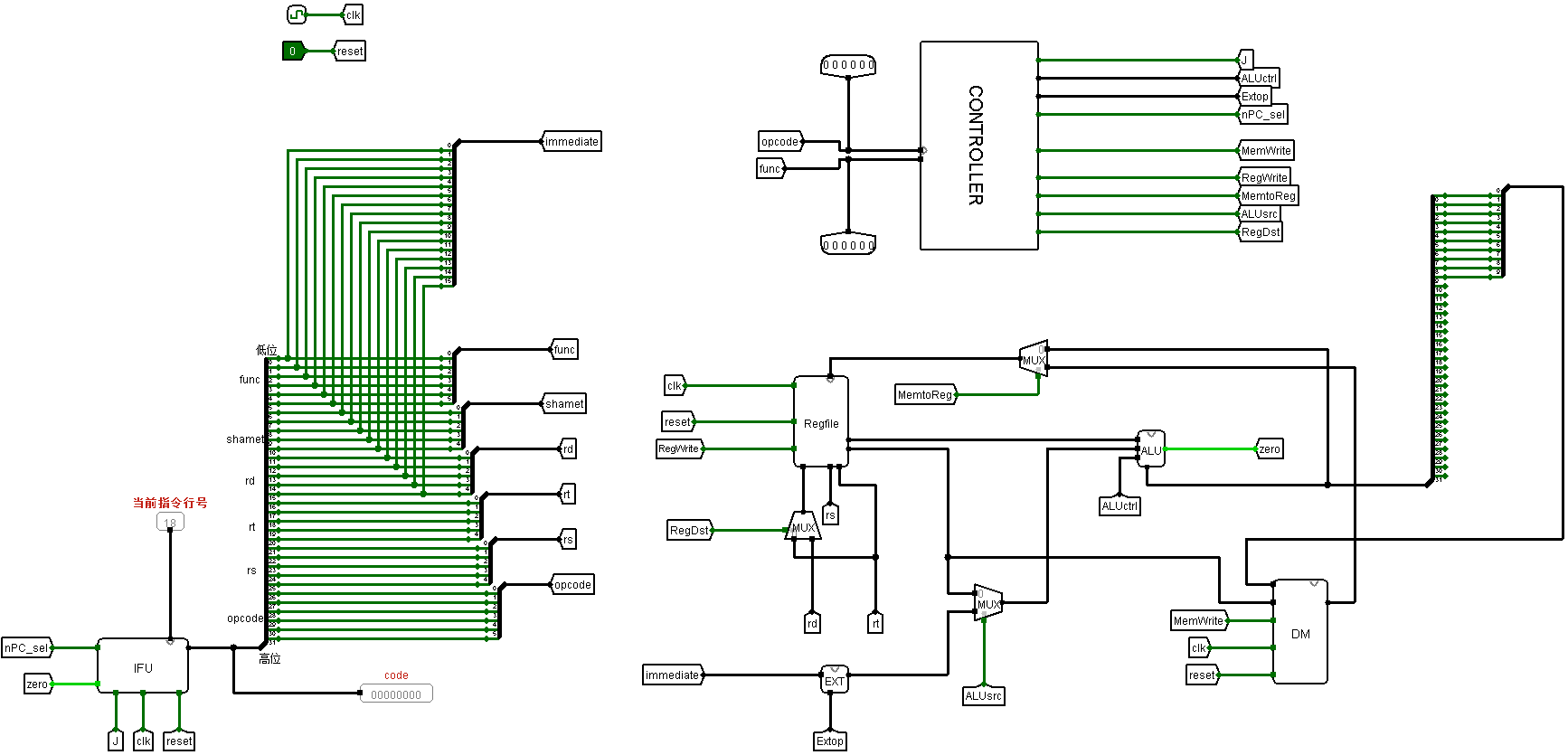


图 2：顶层设计

## 1.2 模块设计图

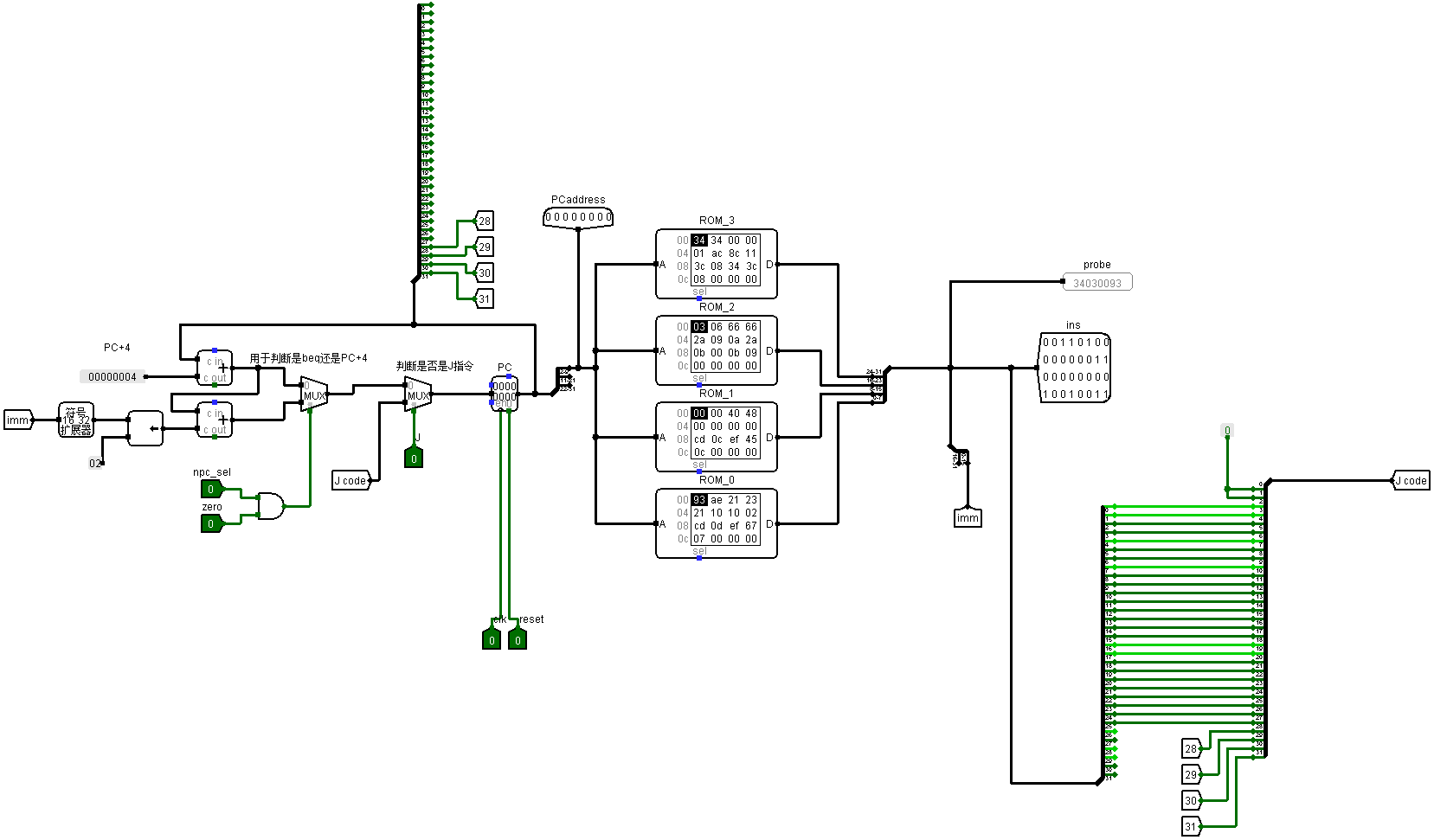


图 3：IFU模块设计图

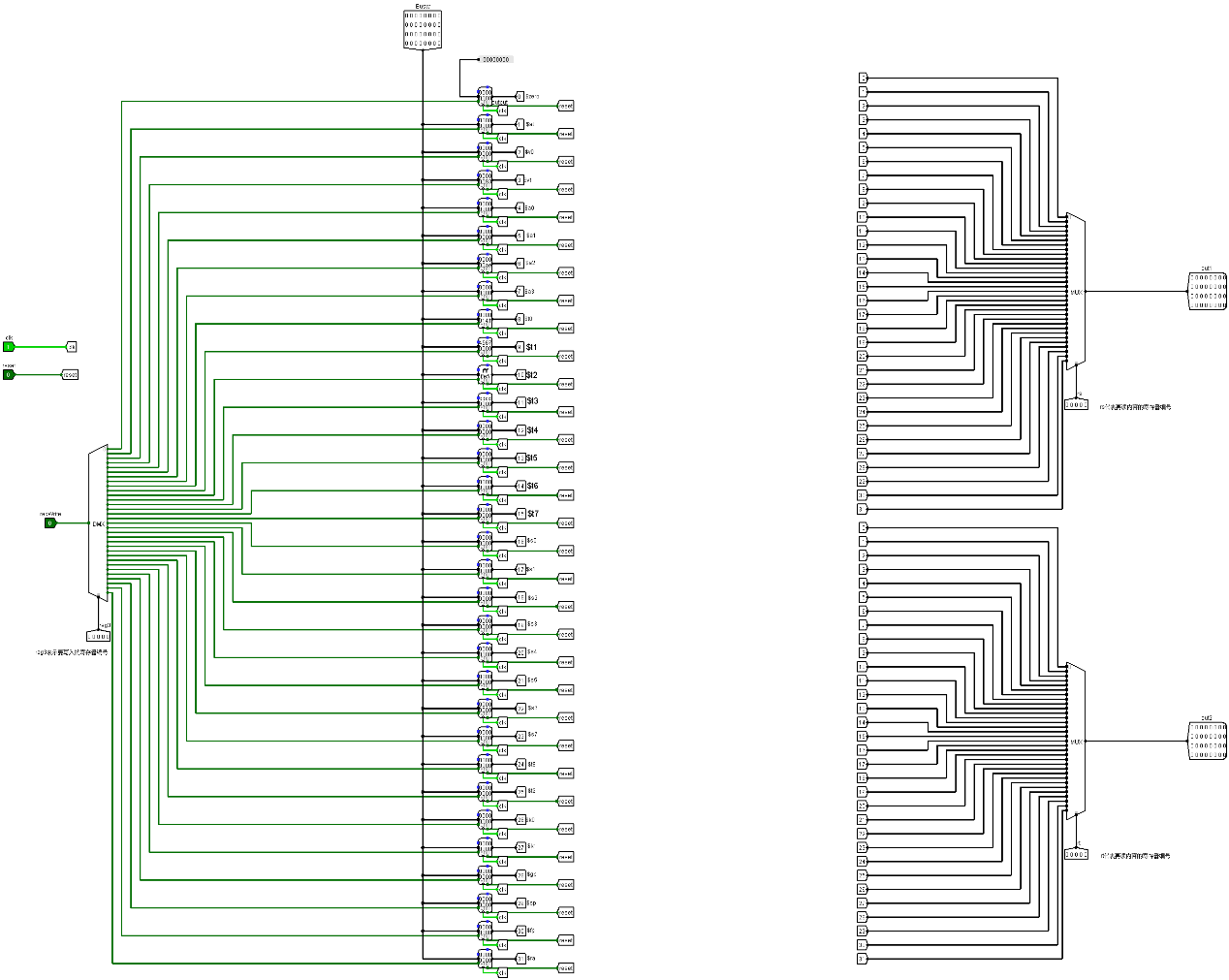


图 4：GPR模块设计图

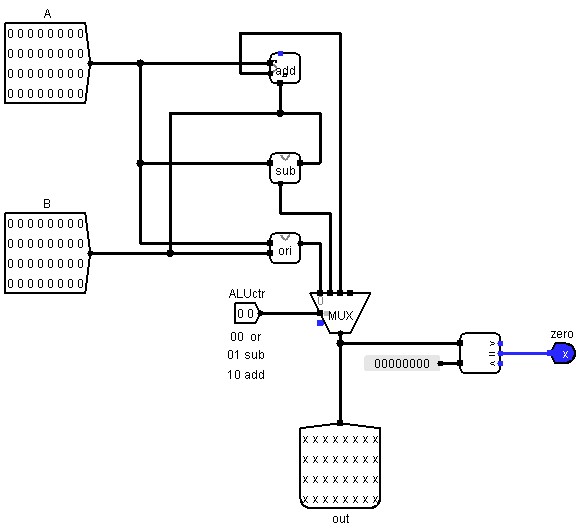


图 5：ALU模块设计图

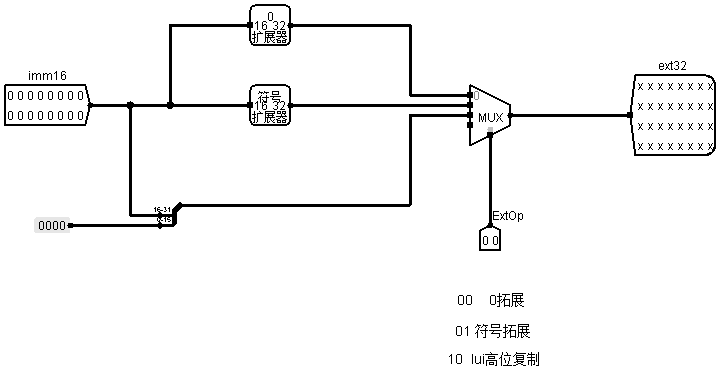


图 6：EXT模块设计图

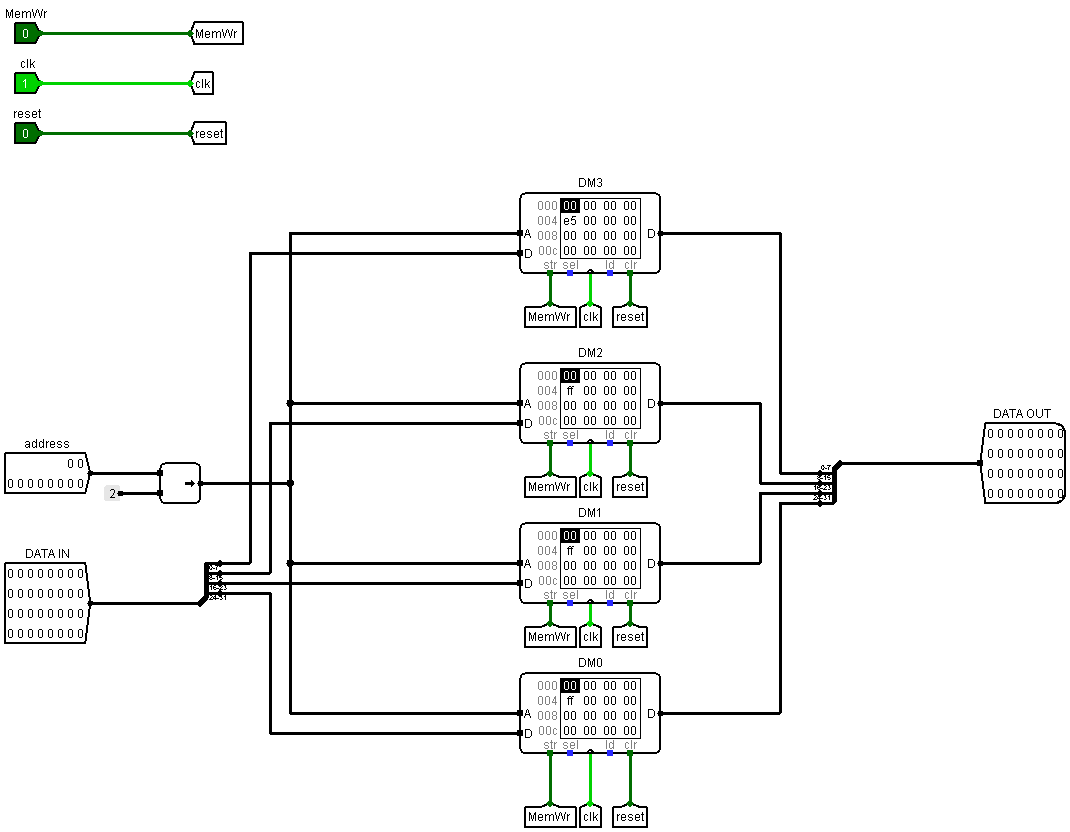


图 7：DM模块设计图

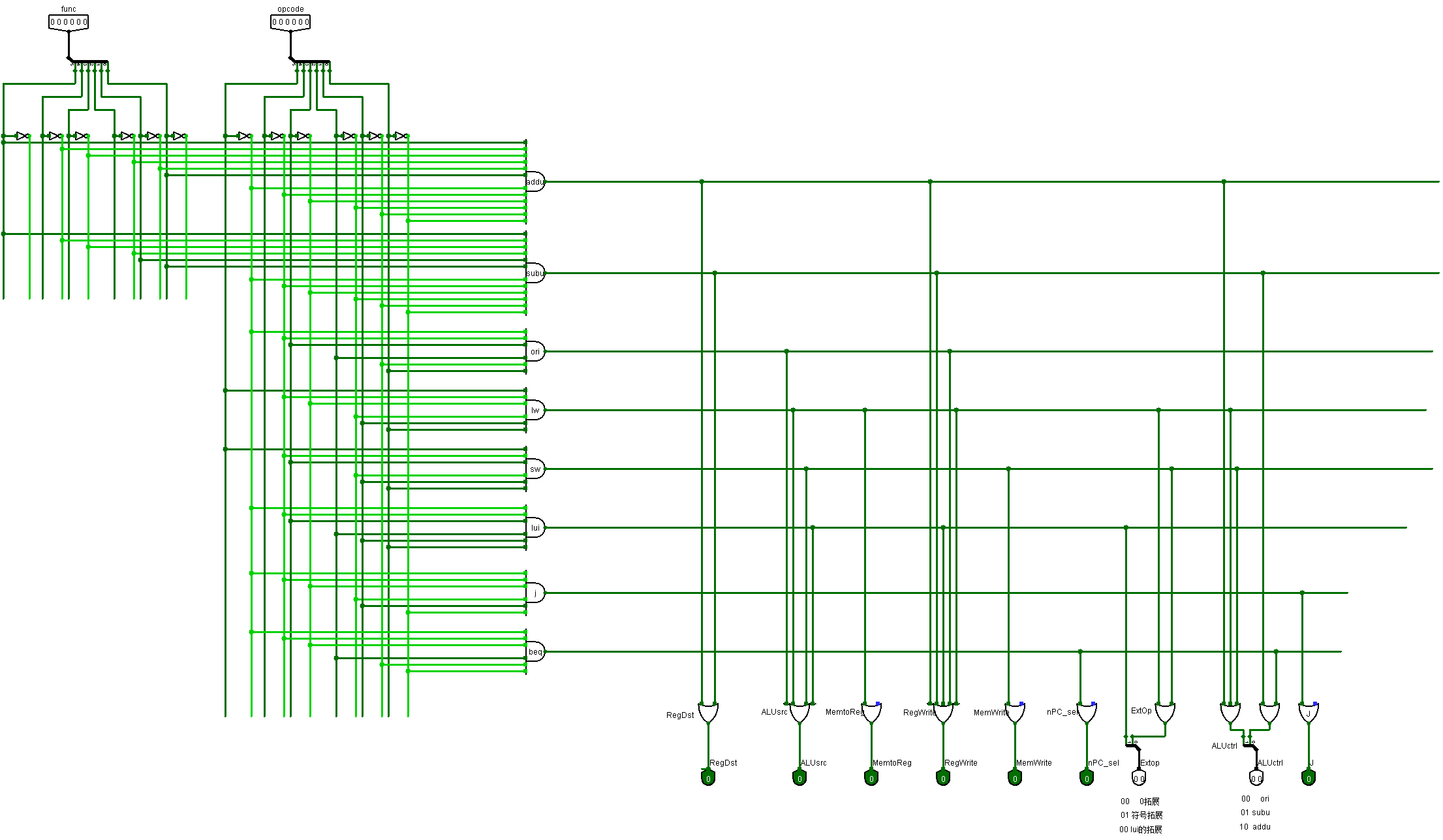


图 8：Control模块设计图

# 二、模块定义

## 2.1 IFU模块定义

### 2.1.1 基本描述

IFU主要功能是完成取指令功能。IFU内部包括PC、IM(指令存储器)以及其他相关逻辑。IFU除了能执行顺序取值令外，还能根据BEQ指令的执行情况决定顺序取值令还是转移取值令。

### 2.1.2 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| npc\_sel | I | 当前指令是否为beq指令标志。  1：当前指令为beq  0：当前指令非beq |
| zero | I | ALU计算结果为0标志。  1：计算结果为0  0：计算结果非0 |
| clk | I | 时钟信号 |
| reset | I | 复位信号。  1：复位  0：无效 |
| ins[31:0] | O | 32位MIPS指令 |
| PCaddress | O | 查看IFU输出第几条指令 |

### 2.1.3 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 复位 | 当复位信号有效时，PC被设置为0x00000000。 |
| 2 | 取指令 | 根据PC从IM中取出指令。 |
| 3 | 计算下一条指令地址 | 如果当前指令不是beq指令，则PC🡨PC+4  如果当前指令是beq指令，且zero为0，则PC🡨PC+4  如果当前指令是beq指令，并且zero为1，则PC🡨PC+4+(sign\_ext(ins[15:0])<<2) |
| 4 | 计算当前输出指令行号 | PCaddress统计IFU中被取指令的条数 |

## 2.2 GPR模块定义

### 2.2.1 基本描述

GPR的主要功能是完成对32个32位寄存器的读写功能，有两个读端口和一个写端口，根据RA和RB的值分别选择要读取的寄存器，根据RW的值选择要写入的寄存器。

### 2.2.2 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| regWrite | I | 当前指令是否写入寄存器。  1：当前指令写入寄存器  0：当前指令不写入寄存器 |
| reset | I | 复位信号。  1：复位  0：无效 |
| clk | I | 时钟信号。 |
| rs[4:0] | I | 当前指令需要被读出数据的寄存器地址1。 |
| rt[4:0] | I | 当前指令需要被读出数据的寄存器地址2。 |
| rw[4:0] | I | 当前指令需要写入的寄存器地址。 |
| Busw[31:0] | I | 当前指令需要写入的数据。 |
| out1[31:0] | O | 当前指令读出的数据1。 |
| out2[31:0] | O | 当前指令读出的数据2。 |

### 2.2.3 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 复位 | 当复位信号有效时，所有寄存器数据清零。 |
| 2 | 取数据 | 根据rs和rt的地址从寄存器中取出数据。 |
| 3 | 写数据 | 如果regWrite有效且clk时钟信号触发边沿，则根据rw的地址将数据写入该地址所对应寄存器中。 |

## 2.3 ALU模块定义

### 2.3.1 基本描述

ALU的主要功能是完成算术运算和逻辑运算，本次设计的ALU可执行的算术运算包括加法和减法，逻辑运算包括或运算。多路选择器根据ALU控制信号判断ALU应进行的运算，选择出运算结果后对其做按位与运算判断是否为0。

### 2.3.2 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| A | I | 参与运算的第一个输入数据。 |
| B | I | 参与运算的第二个输入数据 |
| ALUctr | I | ALU控制信号。  00：或运算  01：减法运算  10：加法运算 |
| out[31:0] | O | ALU运算结果。 |
| zero | O | 运算结果是否为零的标志位。  1：运算结果为0  0：运算结果非0 |

### 2.3.3 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 加法运算 | ALUctr=10时，out=A+B |
| 2 | 减法运算 | ALUctr=01时，out=A-B |
| 3 | 或运算 | ALUctr=00时，out=A|B |

## 2.4 EXT模块定义

### 2.4.1 基本描述

EXT的主要功能是完成16位立即数扩展，根据EXTop信号的不同值分别进行0扩展、符号位扩展或lui指令高位复制扩展，扩展为32位立即数输出。

### 2.4.2 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| imm16[15:0] | I | 需要扩展的16位立即数。 |
| Extop | I | 符号扩展控制信号。  00：高位0扩展  01：符号位扩展  10：低16位补零扩展 |
| Imm32[31:0] | O | 完成扩展的32位立即数。 |

### 2.4.3 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 0扩展 | 高16位补零 |
| 2 | 符号位扩展 | 最高有效位（符号位）复制填满高16位 |
| 3 | 低16位补零扩展 | 低16位补零 |

## 2.5 DM模块定义

### 2.5.1 基本描述

DM是数据存储器，主要功能是完成存储器读写。当写入使能有效时，根据输入的地址将输入的数据写入存储器的相应位置，并输出从该地址读取的数据。

### 2.5.2 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| address | I | 需要读或写的存储器地址。 |
| DataIn[31:0] | I | 需要写入的数据。 |
| reset | I | 复位信号。  1：复位  0：无效 |
| clk | I | 时钟信号。 |
| MemWr | I | 写入使能信号。  1：允许写入  0：不允许写入 |
| DataOut[31:0] | O | 从输入地址读出的数据。 |

### 2.5.3 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 复位 | 当复位信号有效时，所有寄存器数据清零。 |
| 2 | 读数据 | 根据输入的寄存器地址读出数据。 |
| 3 | 写数据 | 根据输入的地址将输入数据写入存储器的相应位置。 |

## 2.6 Control模块定义

### 2.6.1 基本描述

Control主要功能是完成对指令功能的判断和确定每条指令对应的控制信号。根据输入指令的操作码和功能码判断指令，并输出各单元的写使能信号、各多选器的选择信号和ALU的控制信号。

### 2.6.2 模块接口

|  |  |  |
| --- | --- | --- |
| **信号名** | **方向** | **描述** |
| func[5:0] | I | 当前指令的功能码。 |
| opcode[5:0] | I | 当前指令的操作码。 |
| ALUctr | O | ALU控制信号。  00：或运算  01：减法运算  10：加法运算 |
| ALUSrc | O | 选择第二个ALU操作数。  0：操作数为寄存器取出的值  1：操作数为经EXT扩展后的32位立即数 |
| Extop | O | 控制EXT的扩展方式 |
| nPC\_sel | O | beq指令标志。  1：是beq指令  0：非beq指令 |
| MemWrite | O | DM写使能信号。 |
| RegWrite | O | GPR写使能信号。 |
| MemtoReg | O | 选择写入寄存器的数据。  0：写入的数据是ALU计算输出结果  1：写入的数据是DM输出结果 |
| RegDst | O | 写入寄存器的目标寄存器号来源。 |
| J | O | 控制PC是否转移到J指令的地址。 |

### 2.6.3 功能定义

|  |  |  |
| --- | --- | --- |
| **序号** | **功能名称** | **功能描述** |
| 1 | 产生控制信号 | 对输入指令的所有控制信号赋值。 |

# 三、设计的机器指令描述

|  |  |  |  |
| --- | --- | --- | --- |
| **指令操作码助记符** | **机器指令代码** | | **指令功能** |
| opcode | funct |
| addu | 000000 | 100001 | 分别从rs和rt寄存器中取出两个数做无符号数加法，结果放入rd寄存器中。 |
| subu | 000000 | 100011 | 分别从rs和rt寄存器中取出两个数做无符号数减法，结果放入rd寄存器中。 |
| ori | 001101 |  | 从rs寄存器中取出一个数与高位零扩展后的16位立即数做或运算，结果放入rt寄存器。 |
| lui | 110000 |  | 从rs寄存器中取出一个数与低位补零扩展后的16位立即数做或运算，结果放入rt寄存器。 |
| sw | 101011 |  | 从t寄存器中取出一个数，根据基地址+偏移 |
| lw | 100011 |  | 根据基地址+偏移量算出地址，从该地址对应存储器中取出一个数存入rt寄存器中。 |
| beq | 000100 |  | 分别从s和t寄存器中取出两个数比较是否相等，若相等则进行分支跳转，PC<-PC+4+符号扩展imm16,若不相等则不跳转，PC<-PC+4。 |
| j | 000010 |  | 无条件跳转，PC<-{PC+4[31:28]  imm26}。 |

# 四、测试程序

|  |  |  |
| --- | --- | --- |
| **机器码** | **指令** | **注释说明** |
| 34030093 | ori $3,$0,0x93 | 立即数0x93和$0的内容按位或存入$3 |
| 340600ae | ori $6,$0,0xae | 立即数0xae和$0的内容按位或存入$6 |
| 00664021 | addu $8,$3,$6 | $3和$6的内容无符号相加存入$8 |
| 00664823 | subu $9,$3,$6 | $3的内容减去$6的内容存入$9 |
| 012a0021 | addu $0,$9,$10 | $9和$10的内容无符号相加存入$0 |
| ac090010 | sw $9,16($0) | $9的内容写入以$0内容为基地址偏移16个字节地址指向的存储器单元 |
| 8c0a0010 | lw $10,16($0) | 取出以$0内容为基地址偏移16个字节地址指向的存储器单元内容存入$10 |
| 112a0002 | l3:beq $9,$10,l1 | 比较$9和$10内容，若相等则跳转至l1 |
| 3c0bcdcd | lui $11,0xcdcd | 将0xcdcd存入$11高16位，低16位补零 |
| 08000c0d | j end | 无条件跳转至end |
| 340befef | l1:ori $11,$0,0xefef | 立即数0xefef和$0的内容按位或存入$11 |
| 3c094567 | lui $9,0x4567 | 将0x4567存入$9高16位，低16位补零 |
| 08000c07 | j l3 | 无条件跳转至l3 |
|  | end: | end |

# 五、测试结果

## 5.1 GPR运行结果

|  |  |
| --- | --- |
| 图 9：Mars中的GPR | 图 10：logisim中的GPR |

说明：由图可知，在Mars中仿真后寄存器的结果与在logisim中的寄存器结果一致。

## 5.2 DM运行结果

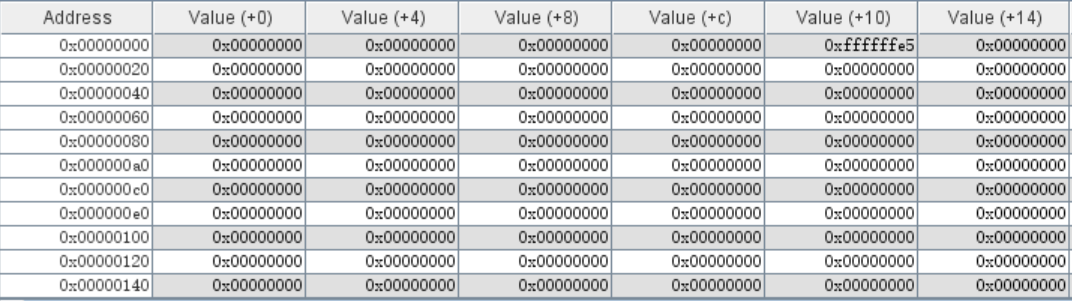


图 11：Mars中的DM

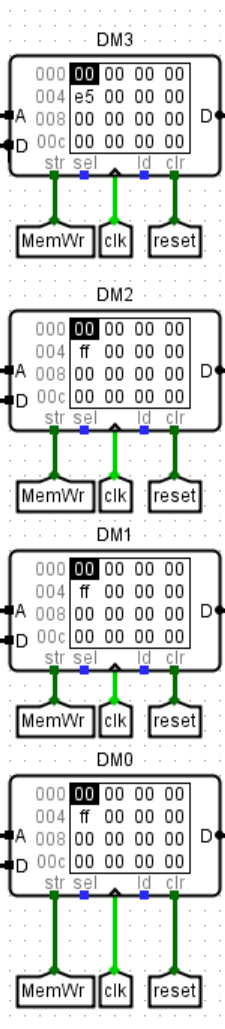




图 12：logisim中的DM

说明：在Mars中仿真后数据存储器的结果与在logisim中的数据存储器结果一致。

# 六、总结与收获

本次大作业的内容是利用Logisim平台和Mars仿真器构建32位单周期CPU处理器。从分析搭建总体架构，到分布设计各个模块，再到将各个模块进行连接，从而形成完整的数据通路，一步步自己搭建CPU的过程加深了我对每个指令的理解，对MIPS体系结构设计有了一定的认识，也加强了我对于模块化层次化设计的能力，构造数字系统时采用分层设计和模块设计可以将复杂的问题拆解成若干容易解决的小问题，降低了解决问题的难度，完成这次大作业的经历对于我之后的学习有很大帮助。