**《程序设计课程设计》实验报告**

**实验名称 CPU模拟器概要设计报告**

**班 级 2020211305**

**学 号 2020211346**

**姓 名 倪玮昊**

# 高层数据信息设计

*包括：公共的数据常量定义、公共的数据类型(结构)定义、公共的数据变量定义，相当于头文件内容，加文字描述*

## 全局常量定义

## 全局变量定义

## int flag=0;//标志寄存器

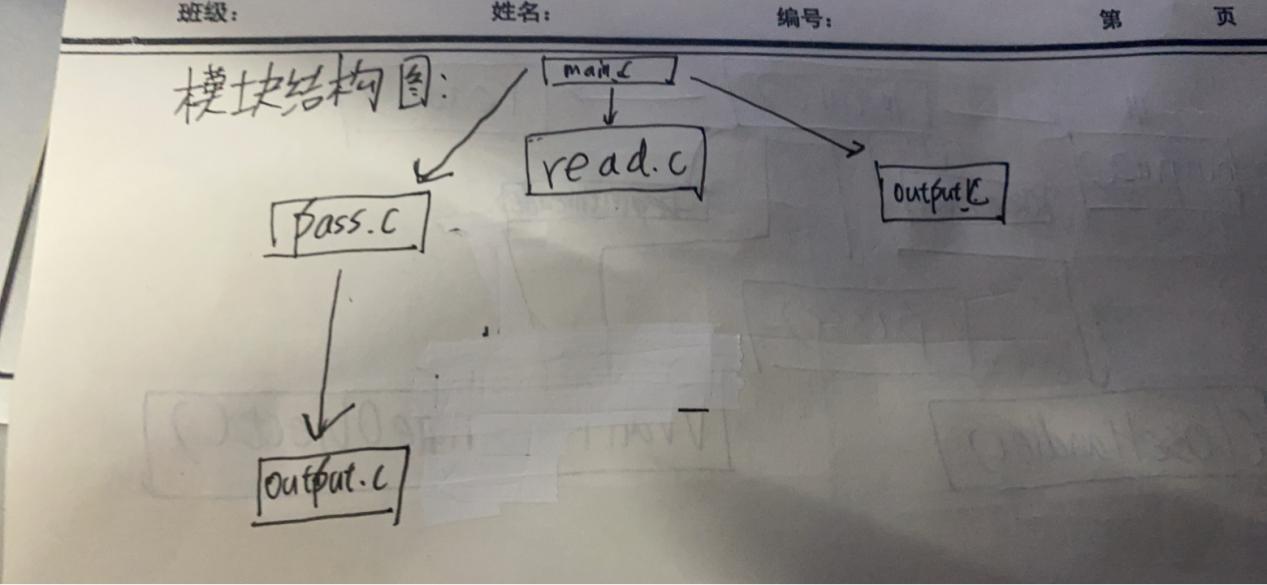
## int ip=0;//程序寄存器

## int ir=0;//指令寄存器

## int stop=0;

# 系统模块划分

## 系统模块结构图

**

## 模块说明

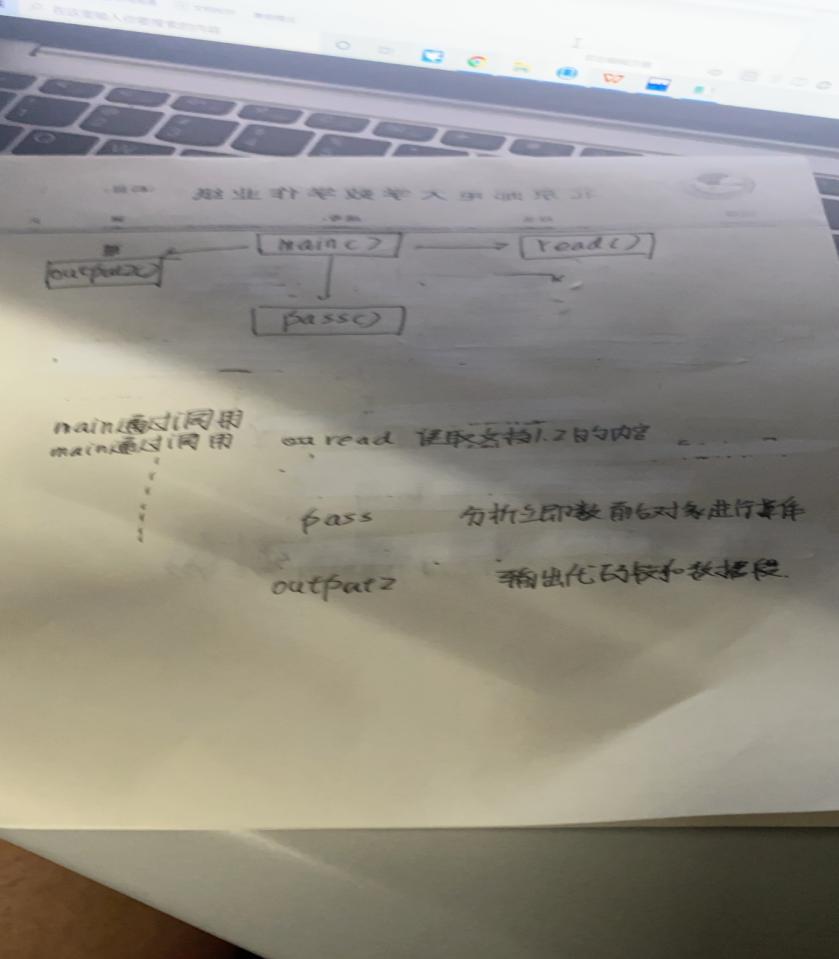
|  |  |  |  |
| --- | --- | --- | --- |
| 模块文件 | 模块功能 | 模块包含的函数名 | 函数功能 |
| read.c | 读取文件内容，并将内容储存至数组code[] | fopen(“dict.txt”,”r”) | 提取文本开头地址作为指针 |
|  |  |
| feof(fPtr) | 读取下一条指令的地址 |
| getc(fPtrt) | 通过地址来读取字符 |
|  |  |
| pass.c | 对指令寄存器中的指令进行解码，分析出指令的操作码，所需操作数，并进行操作 | tranImm(code,line) | 计算立即数 |
| tran(total1,front,back,data,ax,Imm) | 执行数据传输指令 |
| cal(total1,front,back,data,ax,Imm) | 执行算数运算指令 |
| logic(total1,front,back,data,ax,Imm) | 逻辑运算指令 |
| campare(total1,front,back,data,ax,Imm) | 比较指令 |
| jump(total1,front,back,data,ax,Imm) | 跳转指令 |
| inout(total1,front,back,data,ax,Imm); | 输出输入指令 |
| output1(code,line,ax) | 每次执行后输出状态 |
|  | | | |
| output.c | 输出代码段（8X16）  输出数据段（16X16） | output2(code,data); | 输出codeSegment和dataSegment |
| output1(code,line,ax); | 输出每次运行的状态 |

## 函数说明

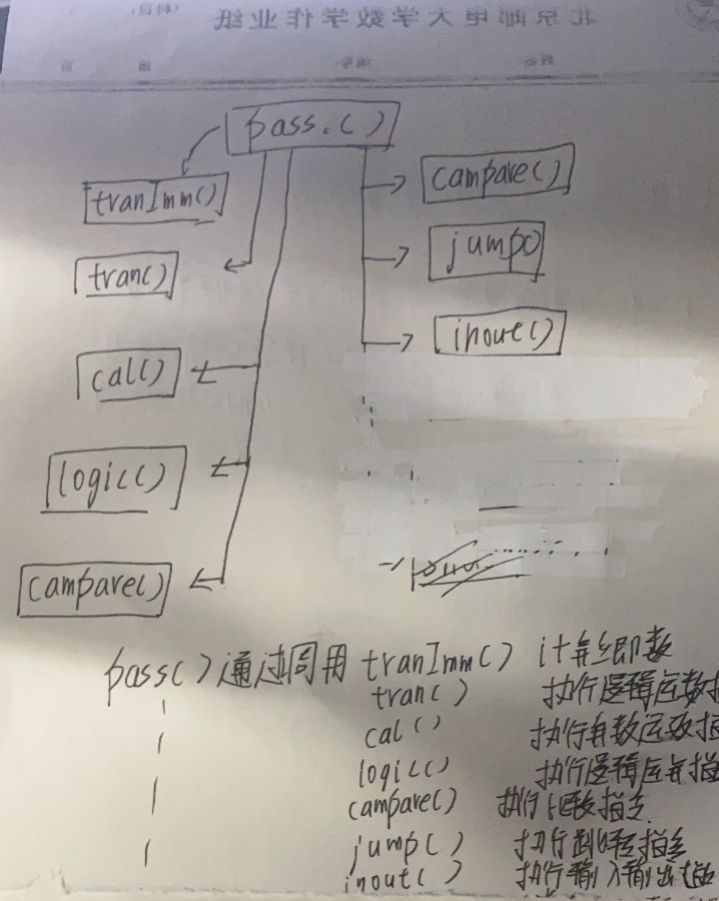
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数原型 | 功能 | 参数 | 返回值 |
| 1 | void read(char code[]) | 记录文本内容 | code是空数组 | void |
| 2 | void pass (char code[],int data[],int ax[]) | 对文本分析，提取出十进制的操作码和对象立即数，并根据操作码进行不同任务 | code是文本，data是空数组，  ax是空数组， | void |
| 3 | void output2 (char code[],int data[]); | 输出8\*16代码  16\*16数据 | ip为程序的寄存器  code代码  data为数据 | void |
| 4 | void output1 (char code[],int line,int ax[]) | 输出每次运行状态 | Code为文本  Line为行数  Ax为寄存器 | Void |
| 5 | void tran (int total1,int front,int back,int data[],int ax[],int Imm); | 将立即数传递给寄存器  或进行寄存器之间的传递 | Total1为操作码  front为前对象  back为后对象  Imm为立即数  flag为空指针  ax为寄存器  data是数据段 | void |
| 6 | void cal (int total1,int front,int back,int data[],int ax[],int Imm); | 将立即数传递给寄存器  或进行寄存器之间的传递 | Total1为操作码  front为前对象  back为后对象  Imm为立即数  flag为空指针  ax为寄存器  data是数据段 | void |
| 7 | void logic (int total1,int front,int back,int data[],int ax[],int Imm); | 将立即数传递给寄存器  或进行寄存器之间的传递 | Total1为操作码  front为前对象  back为后对象  Imm为立即数  flag为空指针  ax为寄存器  data是数据段 | void |
| 8 | void campare (int total1,int front,int back,int data[],int ax[],int Imm); | 将立即数传递给寄存器  或进行寄存器之间的传递 | Total1为操作码  front为前对象  back为后对象  Imm为立即数  flag为空指针  ax为寄存器  data是数据段 | void |
| 9 | void jump (int total1,int front,int back,int data[],int ax[],int Imm); | 将立即数传递给寄存器  或进行寄存器之间的传递 | Total1为操作码  front为前对象  back为后对象  Imm为立即数  flag为空指针  ax为寄存器  data是数据段 | void |
| 10 | void inout (int total1,int front,int back,int data[],int ax[],int Imm); | 将立即数传递给寄存器  或进行寄存器之间的传递 | Total1为操作码  front为前对象  back为后对象  Imm为立即数  flag为空指针  ax为寄存器  data是数据段 | void |

## 2.3 函数调用图示及说明

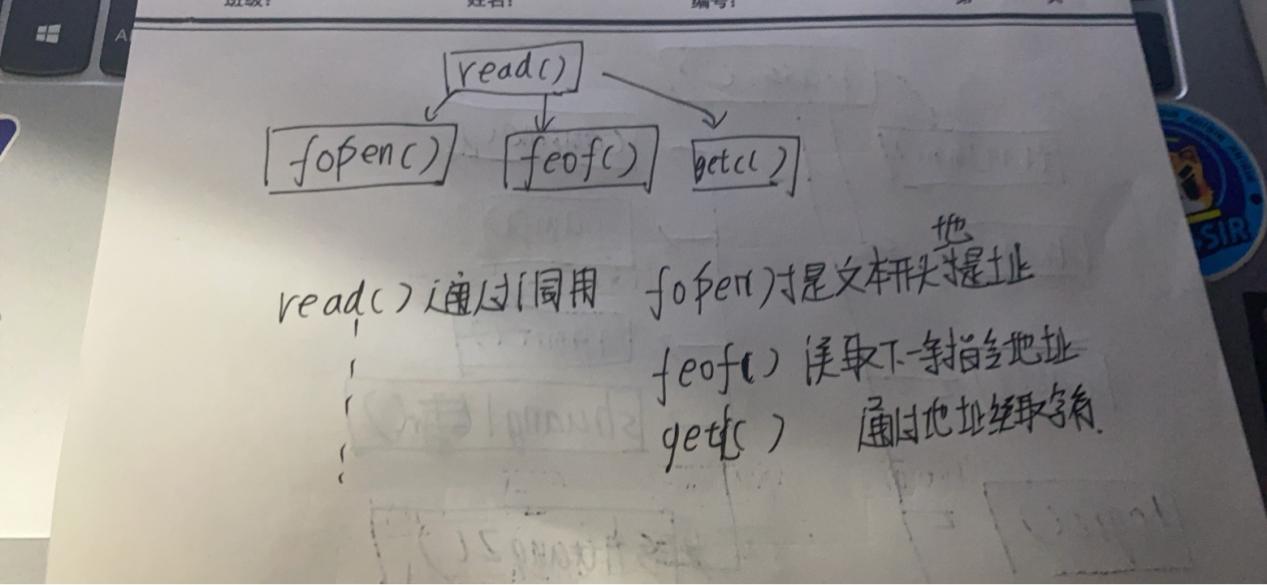
main.c



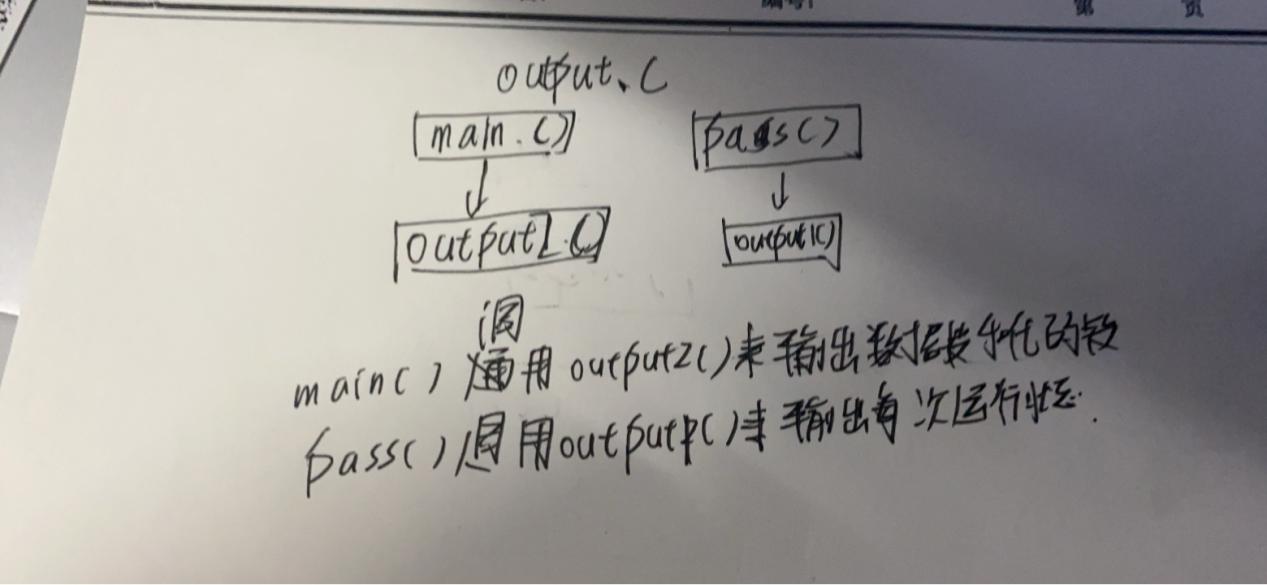
pass.c



read.c



output.c



# 3 核心算法设计

*首先建立主函数，内有三个分函数read pass output，read负责将文本内容记录到数组中，通过判断操作码的值，进入不同的函数中。*

*当操作码=1时，如果后对象=0，则前操作对象的寄存器=立即数，否则判断前操作对象是否小于等于4，如果是，则后操作对象的寄存器等于后操作对象寄存器中的数字。*

*当操作码=2,3,4,5时，如果后操作对象=0，当操作码等于2时，前操作码的寄存器等于立即数加自身；当操作码等于3时，前操作码的寄存器等于自身减立即数；等于4时，前操作码的寄存器等于自身乘立即数；等于5时，前操作码等于自身除以立即数。否则，当操作码等于2时，前操作码的寄存器等于后操作码寄存器中的数字加自身；当操作码等于3时，前操作码的寄存器等于自身减后操作码寄存器中的数字；等于4时，前操作码的寄存器等于自身乘后操作码寄存器中的数字；等于5时，前操作码等于自身除以后操作码寄存器中的数字。*

*当操作码=6,7,8时，如果后操作对象=0，当操作码等于6时，若前操作码的寄存器和立即数都不等于0，则前操作码的寄存器等于1，否则等于,0；操作码等于7时，若前操作码的寄存器和立即数不都等于0，则前操作码的寄存器等于1，否则等于0；若后操作对象不等于0，当操作码等于6时，若前操作码的寄存器和后操作码寄存器中的数字都不等于0，则前操作码的寄存器等于1，否则等于0；当操作码等于7时，若前操作码的寄存器和后操作码寄存器中的数字都不等于0，则前操作码的寄存器等于1，否则等于0；*

*当操作码等于9时，如果后操作码等于0，若前操作码的寄存器大于立即数，则标志寄存等于1；若前操作码的寄存器等于立即数，则标志寄存等于0；若前操作码的寄存器小于立即数，则标志寄存等于-1；如果后操作码不等于0；若前操作码的寄存器大于后操作码寄存器中的数字，则标志寄存等于1；若前操作码的寄存器等于后操作码寄存器中的数字，则标志寄存等于0；若前操作码的寄存器小于后操作码寄存器中的数字，则标志寄存等于-1；*

*当操作码等于10时，如果后操作码等于0，则程序寄存等于自身加立即数减4；如果后操作码等于1且标志寄存等于0，则程序寄存等于自身加立即数减4；如果后操作码等于2且标志寄存等于1，则程序寄存等于自身加立即数减4；如果后操作码等于3且标志寄存等于-1，则程序寄存等于自身加立即数减4；*

*当操作码等于11或2时，若为11，则打印换行符，并给前操作码的寄存器的地址赋值；若为12，则打印换out：，并给前操作码的寄存器赋值；*

*当操作码等于0时，回传stop=1；跳出main.c中的循环函数用output2函数以十进制的方式输出8\*16代码段和 16\*16数据段*