《程序设计课程设计》实验报告

实验名称　　《模拟 CPU 程序设计》概要设计

班　　级　　**2019211318 班**

姓　　名　　叶淳瑜

学　　号　　**2019212686**

1 高层数据结构设计

全局常量/变量定义

为了追求高内聚低耦合的性质，该文件尽量少的使用全局变量。

```
typedef struct Instrution
{
    int op;                     //操作码
    int rs;                     //源操作数寄存器 1
    int rt;                     //源操作数寄存器 2
    int im;                     //立即数
    int address;                //指令地址
    int dec_instruction_high;   //十进制高 16 位指令
    char bin_instruction[34];   //二进制指令
    struct Instrution *next;
}
//代码链表结点

int pc = 0, ir = 0, fr = 0;         //系统寄存器
int gr[9] = {0};                    //通用寄存器
Instruction cache;                  //高速缓存器
int dataSegment[512] = {0};   //数据段(每个数代表 4Byte)
int codeSegment[1024] = {0}; //代码段(每个数代表 2Byte)
```

各模块常量与变量定义

```
LinkInstruction code;   //代码链表头结点
FILE *codeFile;         //文件
char tmp[34];           //2 进制指令字符串
LinkInstruction cur;    //代码链表指针
int offset;             //地址偏移量
```

2 系统模块划分

2.1 系统模块结构图

模块划分思路说明

模块分为 main.c, loadCode.c, analyse.c, runCode.c，分别对应了主函数，载入指令，分析指令，执行指令模块，模拟了冯·诺依曼体系结构 CPU 的真实执行过程。

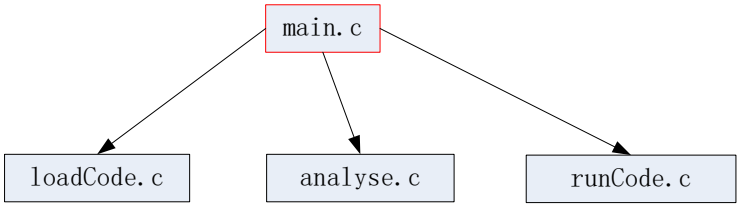


图 2-1 模块结构图

1. **模块名称 main.c**

模块功能简要描述：主函数，连接各个功能函数，完成主要功能

2. **模块名称 loadCode.c**

模块功能简要描述：载入代码，将代码段存入代码链表之中

3. **模块名称 analyse.c**

模块功能简要描述：分析指令类型，并跳转到相应函数

4. **模块名称 runCode.c**

模块功能简要描述：控制指令执行顺序并输出寄存器状态

2.2 各模块函数说明

序号	函数原型	功能	参数	返回值
1	int main(void);	主函数，协调整个 CPU 功能	void	0
2	void loadCode(LinkInstruction code, FILE *fp);	将指令载入到代码链表中	代码表，文件指针	void

3	void runCode(LinkInstruction code);	分析并执行指令	代码表	void
4	int B2D_trasition(char *s, int begin, int end, int mode);	2 进制转 10 进制(包括原码与反码)	2 进制串，起始位置，终止位置，模式	2 进制数对应的 10 进制整数
5	void printStatus();	输出各寄存器状态	无	void
6	void stop();	停机指令	无	void
7	void mov();	数据传送指令	无	void
8	void cmp();	比较指令	无	void
9	void jmp();	跳转指令	无	void
10	void arithmetic_operation();	算术运算指令	无	void
11	void logic_operation();	逻辑运算指令	无	void
12	void standard_io();	输入输出指令	无	void
13	LinkInstruction initInstruction();	初始化链表结点	无	指向代码结点的指针

2.3 函数调用图示及说明

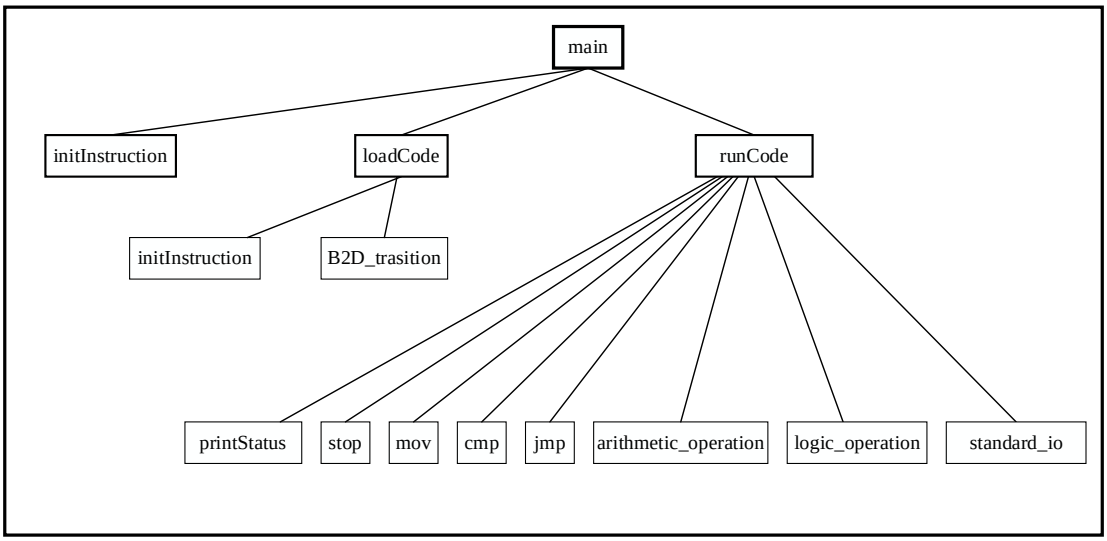


图 2-2 函数调用关系图

解释说明

1. main 函数通过调用函数 initInstruction，完成代码链表头结点的初始化。
2. main 函数通过调用函数 loadCode，runCode，完成载入指令、分析指令与执行指令。
3. loadCode 函数通过对函数 initInstruction 与 B2D_trasition 的调用，完成代码链表的初始化与完成 2 进制、10 进制之间的转换并写入到代码链表中。
4. runCode 函数调用 printStatus，完成寄存器状态的输出。
5. runCode 函数通过调用函数：stop, mov, cmp, jmp, arithmetic_operation, logic_operation, standard_io 等函数，完成对各条指令的分析与执行。

3 高层算法设计

主要步骤

程序的主要步骤为，对指令集文件中的每一行由 32 位二进制数字组成的指令，进行逐个读取，并存入内存的代码链表之中，链表结点包括以下内容：

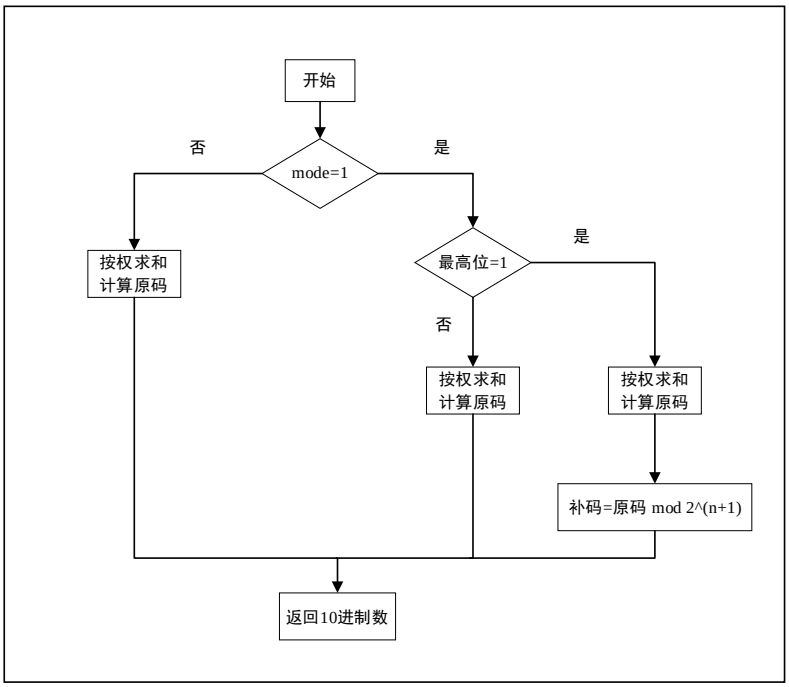
```
typedef struct Instrution
{
    int op;                     //操作码
    int rs;                     //源操作数寄存器1
    int rt;                     //源操作数寄存器2
    int im;                     //立即数
    int address;                //指令地址
    int dec_instruction_high; //十进制高16位指令
    char bin_instruction[34]; //二进制指令
    struct Instrution *next;
} * LinkInstruction, Instruction;
```

之后将指针指向头结点，当指针不为 NULL，则将指令从内存存入高速缓存器 cache 中，然后将指令的前 16 位存入指令寄存器 ir 中，程序计数器 pc 指向下一指令，然后分析指令并执行指令。执行完成后，通过 pc 的值判断程序的下一条指令，继续上述操作。并且程序每执行完一条，打印寄存器状态，直到读取到停机指令。

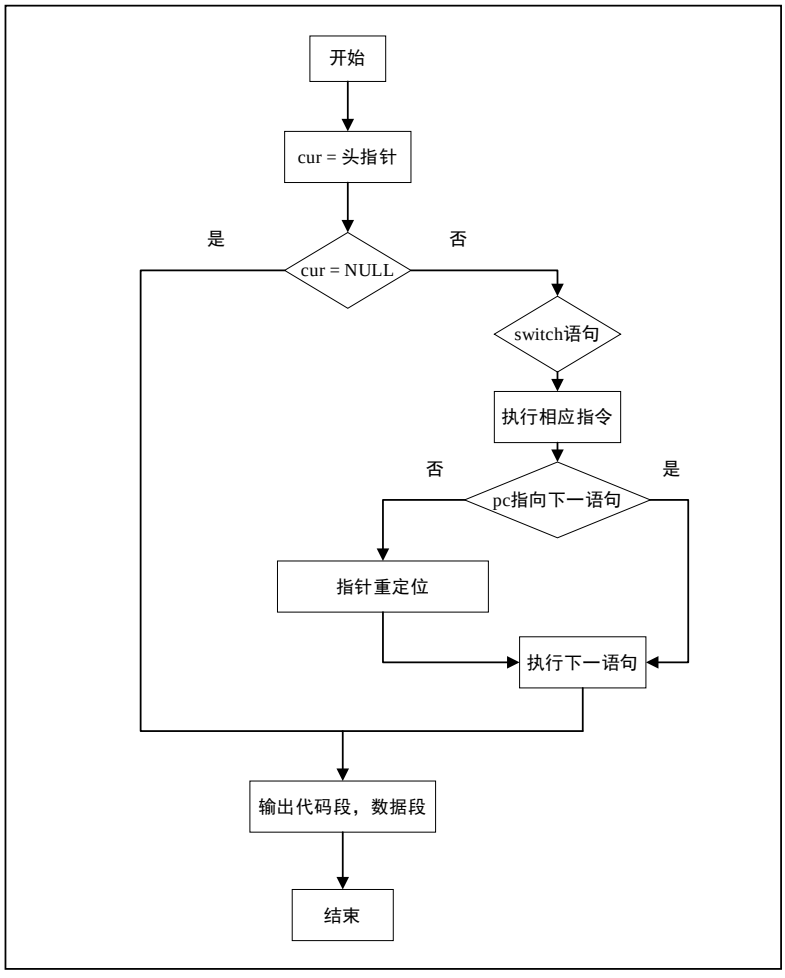
最后打印此时内存中的代码段和数据段，程序到此完全结束。

核心代码

1. 2 进制转 10 进制算法


开始
否
mode=1
是
按权求和
计算原码
最高位=1
是
否
按权求和
计算原码
按权求和
计算原码
补码=原码 mod 2^(n+1)
返回10进制数

2. 程序执行算法


开始
cur = 头指针
是
cur = NULL
否
switch语句
执行相应指令
否
pc指向下一语句
是
指针重定位
执行下一语句
输出代码段，数据段
结束

教师评语：