**《程序设计课程设计》实验报告**

**实验名称 《冯诺依曼式计算机CPU模拟器》概要设计<单核版>**

**班 级 2019211305**

**姓 名 于泳波**

# 1 高层数据结构设计

(包括：重要的数据常量定义、数据变量定义，即各模块要共享的数据类型和参数设计，相当于头文件内容，加文字描述)

**1.1全局常量/变量定义**

BYTE memory[32768]; // 内存全局变量

WORD ax[9]; // 8个16位寄存器，从1开始编号ax[1]~ax[8]，前四个为数据寄存器，后四个为地址寄存器

WORD ir, ip;

short flag;

**1.2全局类型定义**

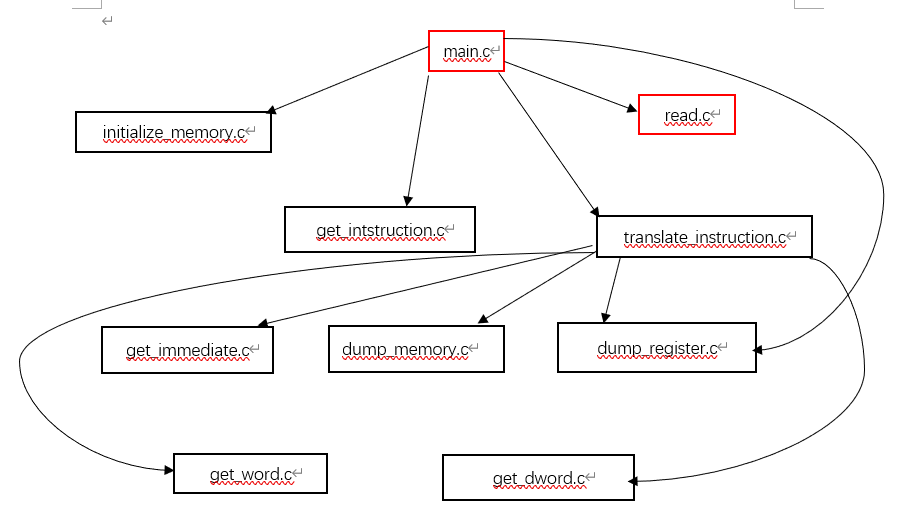
typedef unsigned char BYTE; // 无符号8位存储

typedef unsigned short WORD; // 无符号16位存储

typedef unsigned int DWORD; // 无符号32位存储

# 2 系统模块划分

## 系统模块结构图



1. read.c

功能简介：读取输入到寄存器

1. initialize\_memory.c

功能简介：把文件中的内容存入内存

1. get\_instruction.c

功能简介：把 ip 对应地址处连续2个字节存入ir中，然后ip+4

1. dump\_memory.c

功能简介：程序结束时输出内存的内容。代码段32位一输出，每行8个；数据段16位一输出，每行16个

1. translate\_instruction.c

功能简介: 翻译指令，根据函数指针表判断ir的前8位，调用对应的函数。

1. get\_immediate.c

功能简介：获取当前ir对应的立即数，通过get\_word(ip-2)进行获取。由于立即数为补码，所以通过强制转换将无符号存储形式转换为有符号存储形式

1. dump\_register.c

功能简介：每次指令结束输出寄存器内容

1. get\_word.c

功能简介：从内存中取连续两字节内容

1. get\_dword.c

功能简介：从内存中连续取四字节内容

## 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() | | 程序开始时执行，主函数控制流程 |  | 主函数返回值 | |
| 2 | void initialize\_memory() | | 把文件中的内容存入内存 |  |  | |
| 3 | void dump\_memory() | | 程序结束时输出内存的内容。代码段32位一输出，每行8个；数据段16位一输出，每行16个。 |  |  | |
| 4 | void dump\_register() | | 每次指令结束输出寄存器内容 |  |  | |
| 5 | void get\_instruction() | | 把 ip 对应地址处连续2个字节存入ir中，然后ip+4。 |  |  | |
| 6 | void translate\_instruction() | | 翻译指令，根据函数指针表判断ir的前8位，调用对应的函数。 |  |  | |
| 7 | WORD get\_word(WORD addr) | | 从内存中取连续两字节内容 | 指定内存地址 | 内存地址向后两个字节内容 | |
| 8 | DWORD get\_dword(WORD addr) | | 从内存中连续取四字节内容 | 指定内存地址 | 内存地址向后四个字节内容 | |
| 9 | void Stop() | | 停机指令，输出寄存器和内存内容并终止程程序 |  |  | |
| 10 | void Add() | | 进行加法运算 |  |  | |
| 11 | void Sub() | | 进行减法运算 |  |  | |
| 12 | void Mul() | | 进行乘运算 |  |  | |
| 13 | void Div() | | 进行除法运算 |  |  | |
| 14 | void Jmp() | | 判断后进行无条件跳转或条件跳转，跳转时需要回退ip+4的结果。 |  |  | |
| 15 | void Read() | | 读取用户输入到寄存器 |  |  | |
| 16 | void Write() | | 将某个寄存器内容输出 |  |  | |
| 17 | void Cmp() | | 寄存器之间进行比较，或寄存器和立即数之间进行比较，结果存入flag寄存器 |  |  | |
| 18 | void Load() | | 进行数据传送。 |  |  | |
| 19 | void And() | | 进行逻辑与运算 |  |  | |
| 20 | void Or() | | 进行逻辑或运算 |  |  | |
| 21 | void Not() | | 进行非操作 |  |  | |
| 22 | short get\_signed\_immediate() | | 获取当前ir对应的立即数，通过get\_word(ip-2)进行获取。由于立即数为补码，所以通过强制转换将无符号存储形式转换为有符号存储形式。 |  | 获取的当前指令的立即数 | |
| 23 | | Short get\_unsigned\_immediate() | 获取当前ir对应的立即数，通过get\_word(ip-2)进行获取。由于立即数为补码,无符号 |  | |  | |

## 2.3 函数调用图示及说明

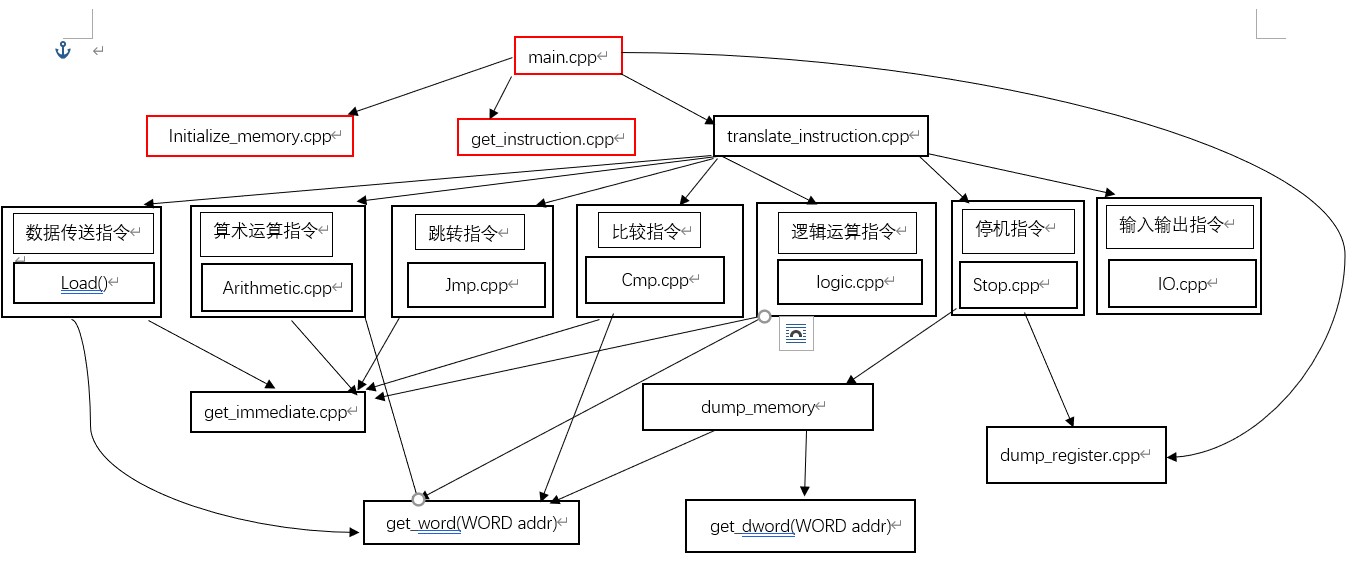
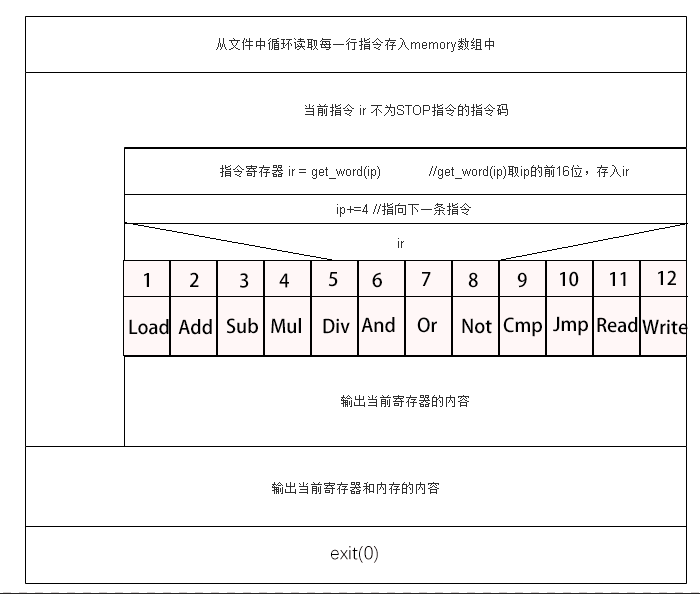


图2-3 函数调用关系图示例

解释说明：如图中所示，主函数对initialize\_memory的调用，完成从文件读内存以及内存的初始化；对get\_instruction的调用完成取指和ip+4；对translate\_instruction的调用，完成译指和指令对应函数的调用。每个指令模块可能会调用get\_immediate函数来获取当前指令的立即数，以及get\_word和get\_dword，用于获取对应内存地址处后两个或者四个字节的内存数据。当每次指令执行完成后，主函数调用dump\_register输出当前寄存器状态。当遇到停机指令时，调用dump\_memory按格式输出当前内存的状态。

# 3 高层算法设计

首先通过fopen()函数读文本的每一行，逐字节存入memory的代码段，然后循环通过get\_instruction()函数取指令，translate\_instruction()函数进行译指和执行指令的操作，执行指令完成后用dump\_register()函数输出寄存器的信息。程序在执行到Stop()函数时输出当前寄存器和内存信息后终止。



**教师评语：**