**目录**

[2 系统整体设计 1](#_Toc444539074)

[2.1 汇编器 1](#_Toc444539075)

[2.2 模拟器 1](#_Toc444539076)

[2.3 公有模块 2](#_Toc444539077)

[3 数据结构设计 3](#_Toc444539078)

[3.1 符号信息结构 3](#_Toc444539079)

[3.2 标号信息结构 3](#_Toc444539080)

[4 系统各模块设计与实现 4](#_Toc444539081)

[4.1 预处理模块 4](#_Toc444539082)

[4.2 指令翻译模块 6](#_Toc444539083)

[4.3 初始化模块 7](#_Toc444539084)

[4.4 基础模块 7](#_Toc444539085)

[4.5 执行模块 8](#_Toc444539086)

[5 系统测试 10](#_Toc444539087)

[5.1题目1 10](#_Toc444539088)

[5.2 题目2 11](#_Toc444539089)

[5.3 题目3 12](#_Toc444539090)

[5.4 题目4 13](#_Toc444539091)

# 2 系统整体设计

## 2.1 汇编器

汇编器

预处理器

指令翻译

图2.1 汇编器模块结构图

汇编器由两大模块组成：预处理模块和指令翻译模块，如图2.1所示。

预处理器主要将源文件处理成1行1条指令的形式，解析伪指令和标号供后续使用。预处理过程总共进行3次：第一次去除空行、注释、前置空白字符；第二次解析去除伪指令；第三次解析去除标号。注：一行不能有多个标号，但标号后指令可有可无，但不能接伪指令。

指令翻译模块负责将指令翻译为二进制码，确定指令操作码并检查使用是否合法，然后按照对应的形式进行译码并将结果以二进制形式写入输出文件。

## 2.2 模拟器

模拟器

基础模块

执行模块

初始化模块

图2.2 模拟器模块结构图

模拟器由三大模块构成：初始化模块、执行模块、基础模块，如图2.2所示。

初始化模块主要进行内存分配，寄存器的初始化，注册终止函数，随后将数据段和代码段从文件中读入内存。

基础模块主要提供一些底层操作，如：设置、获取OF、CF位的值，跳转，读写寄存器（检查是否对Z寄存器进行写入），检查溢出，附加段的压栈和弹出等。并为执行模块提供相关变量的声明。

执行模块负责执行指令，从代码段中读一个字，拆分为指令和操作数，调用相关函数。并更改寄存器PC的值。

## 2.3 公有模块

主要被汇编器使用，主要功能为获取指令操作码，提供部分指令格式检查相关的函数，定义了保存指令助记符的数组、保存指令操作数个数的数组、保存指令编码函数索引的数组，并提供对外声明。

# 3 数据结构设计

汇编器中，使用两单链表存储标号和伪指令定义的符号信息。

## 3.1 符号信息结构

typedef struct symbol

{

char name[40]; //符号名称

int size; //包含元素个数

int type; //类型，1表示BYTE，2表示WORD

unsigned long location; //在数据段中的起始位置

union{

char \*byte\_ptr;

short \*word\_ptr;

}data; //保存的数据

struct symbol \*next; //后继结点

}Symbol;

## 3.2 标号信息结构

typedef struct label

{

char name[40]; //名称

unsigned long location; //对应行号

struct label \*next; //后继结点

}Label;

# 4 系统各模块设计与实现

模拟器

执行模块

取指

模拟内存

代码段

数据段

堆栈段

附加段

载入

目标文件

中间文件（只包含指令）

源文件

预处理

指令翻译

符号、标号信息

汇编器

建立

初始化

初始化模块

模拟寄存器

输出设备

输入设备

图4.1 系统流程图

## 4.1 预处理模块

包含文件：preprocess.c，preprocess.h

定义的函数：

1、FILE \*preprocess(FILE \*fin)

功能： 对文件fin进行预处理，解析伪指令和标号，处理完成后每行只有一条语句

返回值：用于汇编的文件指针

2、static void process1(FILE \*fin, FILE \*fout)

功能： 执行第一遍预处理，去除空行、注释和前置空白字符，fin为输入文件，fout为输出文件

返回值：无

3、static void process2(FILE \*fin, FILE \*fout)

功能： 执行第二遍预处理，解析伪指令并建立链表，fin为输入文件，fout为输出文件

返回值：无

4、static void process3(FILE \*fin, FILE \*fout)

功能： 指向第三遍预处理，解析标号并建立链表，fin为输入文件，fout为输出文件。经过此次预处理后，得到最后用于汇编的文件

返回值：无

5、static void removeSpace(FILE \*fin)

功能： 从fin中读取空白字符并抛弃，直到遇到非空白字符为止

返回值：无

6、static int readLine(FILE \*fin, char \*storage)

功能： 从fin中读取一行存储在storage中，遇到换行符或#则停止

返回值：此行之后是否有注释

7、static void eatComment(FILE \*fin)

功能： 去除行末注释，从fin中读取字符并抛弃直到遇到换行符

返回值：无

8、static int checkIdentifier(char \*name, int mode)

功能： 检查name指定的标号、符号是否合法，mode为1表示检查符号，2表示检查标号

返回值：若标号、符号已被声明过或与指令重名则返回0，否则返回1

9、void disposeList()

功能： 释放符号、标号链表，被注册为终止函数

返回值：无

10、void outputData(FILE \*fout)

功能： 将数据段的大小和内容写入文件fout

返回值：无

11、unsigned long transSymbol(char \*symName)

功能： 将符号名symName翻译为其在数据段中的起始地址

返回值：符号symName在数据段中的起始地址

12、unsigned long transLabel(char \*labName)

功能： 将标号名labName翻译为其表示的行号

返回值：标号labName表示的行号

## 4.2 指令翻译模块

包含文件：as.c，as.h

定义的函数：

1、void assembly(FILE \*fin, FILE \*fout)

功能： 先将数据段写入文件fout，再不断地从fin中读取指令，进行汇编，将二进制码写入文件fout

返回值：无

2、static int transREG(char \*reg)

功能： 检查寄存器名称是否合法，非法则产生错误、终止进程，合法则返回其编号

返回值：寄存器的编号

3、static unsigned long (\*encodeFunc[8])(char \*operands)

功能： 对应8种格式的指令的汇编函数，将操作数译为二进制码

返回值：操作数转换得到的二进制码

## 4.3 初始化模块

包含的文件：ssim.c

定义的函数：

1、static void init()

功能： 为模拟内存分配空间，初始化相关模拟寄存器

返回值：无

2、static void load(FILE \*fin)

功能： 从fin文件中读入数据段和代码段

返回值：无

## 4.4 基础模块

包含的文件：simBas.c，simBas.h

定义的函数：

1、void set\_OF(int flag)

功能： 将OF位设为flag的末位

返回值：无

2、int get\_OF()

功能： 获取OF位的值

返回值：OF位的值

3、void set\_CF(int flag)

功能： 将CF位设为flag的末位

返回值：无

4、int get\_CF()

功能： 获取CF位的值

返回值：CF位的值

5、void push\_ES(short data)

功能： 将data压入附加段，并移动EDI

返回值：无

6、short pop\_ES()

功能： 从附加段中弹出1字的数据并移动EDI

返回值：弹出的数据

7、void jump(long address)

功能： 跳转到address表示的位置

返回值：无

8、void writeREG(int index, short data)

功能： 将编号为index的寄存器值改为data，意图改动寄存器Z的值将引发错误并结束程序

返回值：无

9、short readREG(int index)

功能： 获取编号为index的寄存器的值

返回值：寄存器的值

10、void checkOF(int num)

功能： 检查num是否导致溢出，并设置OF位

返回值：无

## 4.5 执行模块

包含文件：ssim.c，inst.c，inst.h

定义的函数：

1、static void run()

功能： 取指执行。从代码段中取指令，调用相关函数执行指令

返回值：无

2、void (\*exeInst[32])(long operands)

功能： 指令执行函数，数组下标与指令操作码对应。各函数都以func\_助记符 命名

返回值：无

# 5 系统测试

## 5.1题目1

求 1+2+3+…+100，并输出运算结果。

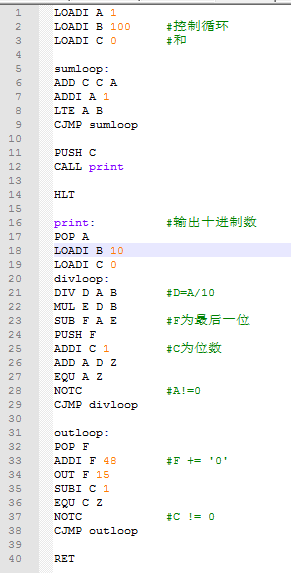


图5.1 题目1代码

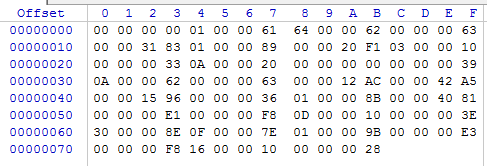


图5.2 题目1生成的目标文件

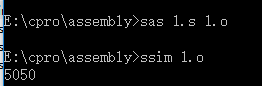


图5.3 题目1运行结果

## 5.2 题目2

求 1！+2！+3！+4！+5！ ，并输出运算结果

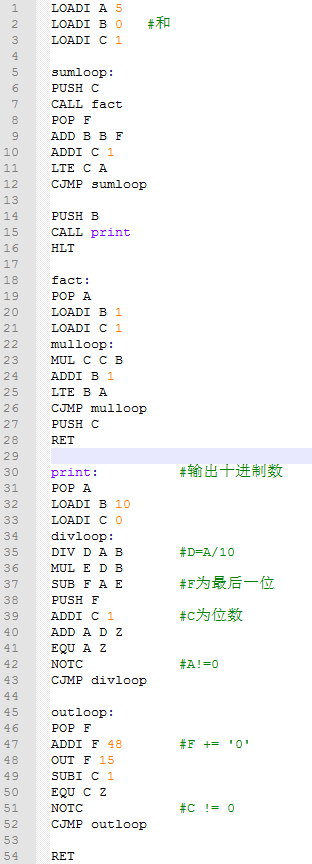


图5.4 题目2代码

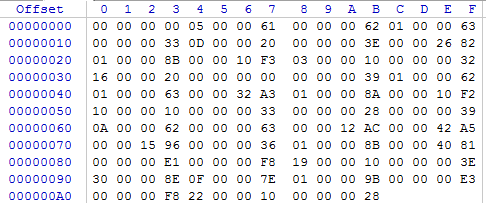


图5.5 题目2生成的目标文件

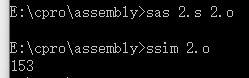


图5.6 题目2运行结果

## 5.3 题目3

将“ Simulator and Assembler”拷贝复制到新串并输出运算结果

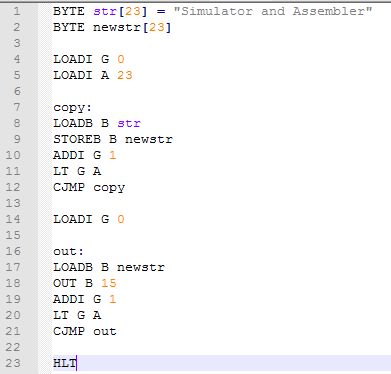


图5.7 题目3代码

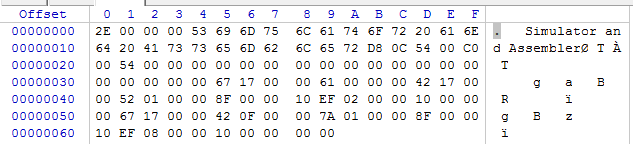


图5.8 题目3目标文件

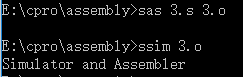


图5.9 题目3运行结果

## 5.4 题目4

将“I design a program. It is ”和” Simulator and Assembler”连接成为新串并输出运算结果。

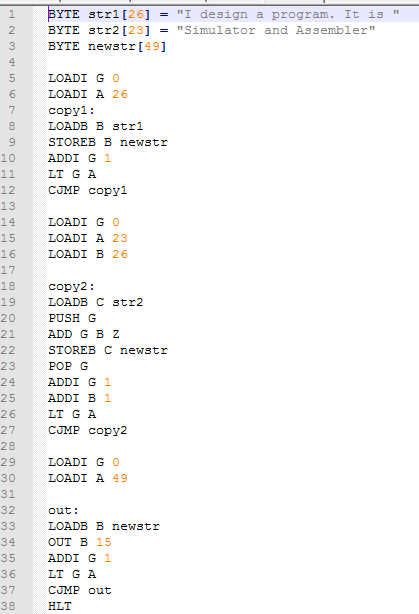


图5.10 题目4代码

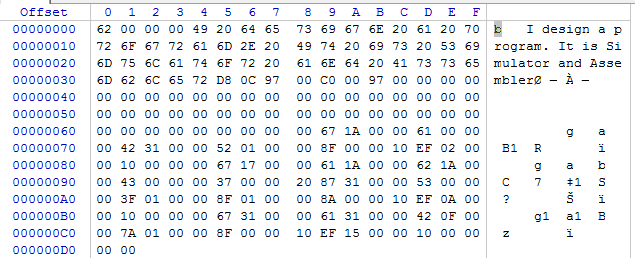


图5.11 题目4目标文件

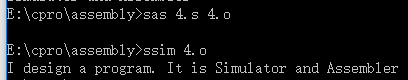


图5.12 题目4运行结果