|  |
| --- |
| **Perigee OS设计文档** |
|  |
|  |
| [本文档详细描述了Perigee OS的设计思想和程序设计结构，旨在帮助预期读者了解整个系统的工作原理，设计思路和程序结构] |
|  |
|  |
| **Perigee Group** |
|  |

**Perigee OS 设计文档**



微软亚洲研究院

张旻昊

[1引言 2](#_toc34)

[1.1编写目的…………………………………………………………2](#1.1.1.1编写目的|outline)

[1.2背景………………………………………………………………2](#1.2.1.2背景|outline)

[1.3定义………………………………………………………………2](#1.3.1.3定义|outline)

[2总体设计 ……………3](#_toc48)

[2.1需求规定…………………………………………………………3](#3.2.1需求规定|outline)

2.2具体设计及处理流程....................................................................3

[3系统结构设计 14](#_toc233)

[3.1汇编编译………………………………………………………..14](#4.1.3.1用户接口|outline)

[3.2中断处理………………………………………………………..14](#4.2.3.2外部接口|outline)

[3.3指令解析………………………………………………………..14](#4.3.3.3内部接口|outline)

[4运行设计 14](#_toc240)

[4.1运行模块组合 14](#5.1.4.1运行模块组合|outline)

[4.2运行控制 14](#5.2.4.2运行控制|outline)

[4.3运行时间 15](#5.3.4.3运行时间|outline)

[5系统数据结构设计 15](#_toc247)

[5.1逻辑结构设计要点 15](#6.1.5.1逻辑结构设计要点|outline)

[5.2物理结构设计要点 15](#6.2.5.2物理结构设计要点|outline)

[6系统出错处理设计 15](#_toc254)

[6.1出错信息 15](#7.1.6.1出错信息|outline)

[6.2补救措施 16](#7.2.6.2补救措施|outline)

[6.3系统维护设计 16](#7.3.6.3系统维护设计|outline)

# 1引言

## 1.1编写目的

本文档详细描述了Perigee OS的设计思想和程序设计结构，旨在帮助项目组成员了解整个系统的工作原理，设计思路和程序结构，为后续的编码工作提供有力的支持。本文档预期读者为本项目组成员及操作系统课老师和助教。

## 1.2背景

**名称**：Perigee OS

**概述**：本系统是操作系统课程提出的作业项目，旨在设计一套模拟简易操作系统，尤其是CPU运行及调度的系统。

**开发环境**： Visual Studio2015 preview+Adobe Photoshop CS6

## 1.3定义

**PCB：Process Control Block**

**进程控制块**

**PSW：Program Status Word**

**程序状态字**

# 2总体设计

# 2.1需求规定

* **CPU模拟器：**

1）源程序使用Intel汇编语句，必须能处理以下类型汇编语句：

Arithmetic: INC, DEC, NEG, ADD, NEG；MUL,IMUL,DIV（整形运算即可）

Boolean: AND, OR, XOR, CMP

Data Transfer: MOV（MOVZX,MOVSX）

Control: JMP,LOOP（LOOPZ,LOOPE,LOOPNZ,LOOPNE）

Stack Operations：PUSH, POP

Using Procedures: CALL, RET

2）模拟器能单步执行程序，并实时输出执行当前语句时的各寄存器的值。

3）console,printer,disk全部是模拟实现，在程序源文件中通过语句或函数实现。console主要是指屏幕，能在定位到模拟的屏幕上的任意位置并输出内容。Printer要求能模拟输出送入到打印机缓冲区的数据。Disk要求能把模拟内存中的数据保存到模拟硬盘上并能从模拟硬盘读取数据到模拟内存中。

* **命令解释程序：**

1. 在CPU模拟器的基础上实现命令解释程序，将命令解析成汇编语言然后执行。
2. 命令自己定义，但必须实现：

Execute：后面接源文件名称，用来执行源文件

Help: 给出所有你实现的命令的用法

实现命令数目不得少于5条。

* **中断程序处理框架：**

1. 框架必须处理时钟中断和I/O中断。
2. 必须基于CPU模拟器和命令解释程序实现。
3. 必须符合中断处理程序的步骤。
4. 如有能力可实现嵌套中断处理。

* **实现进程管理模块：**

1） 正确描述进程，如进程的PCB等  
 2） 实现最基本的进程3种状态的切换，必须符合进程切换的步骤。

3） 模拟PSW

* **提供图形界面。**

# 2.2具体设计及处理流程

* **Perigee OS汇编指令格式：**

Add 指令（加法）：

add operand1,operand2

运算过程为operand1=operand1+operand2;

支持间接寻址 ，即操作数可以为如[bx] 的这种形式：add ax,[bx]

也可以是立即数 :add ax,5

也可是标签指定的内存数据区域: add dx,m1

Sub 指令（减法）：

sub operand1,operand2

运算过程为operand1=operand1-operand2;

Mul 指令（无符号乘法）：

mul operand

运算过程为 ax = ax\*operand;

Dix 指令（无符号除法）：

dix operand

运算过程为 ax=ax/operand;

Inc 指令（递增）：

inc operand

运算过程为 operand++;

Dec 指令（递减）：

dec operand

运算过程为 operand--;

以上为算术运算操作符

注意操作数的寻址方式和第一个的add相同

并且每次运算都会判断和改变PSW

And 指令（按位与）：

And operand1,operand2

运算过程为 operand1=operand1&operand2;

Or 指令（按位或）：

or operand1,operand2

运算过程为 operand1=operand1|operand2;

Xor 指令（按位异或）：

xor operand1,operand2

运算过程为 operand1=operand1^operand2;

以上为位运算操作符

Lea 指令（取数据地址）：

lea opreand,label

运算过程为：operand = [label]--（label 的首地址）

Mov 指令（数据转移）：

mov operand1,operand2

运算过程为：operand1=operand2

以上为数据操作指令

Cmp 指令（数据比较）：

cmp operand1,operand2

运算过程为： 如果operand1>operand2 , PSW 中的SF（符号标记位）为0

如果operand1<operand2, PSW 中的SF为1

如果operand1=operand2, PSW 中的ZF（零标记位） 为1

以上为数据比较指令

Jmp 指令（无条件跳转）：

jmp label

运算过程为pc->[label]

Js 指令（为负跳转）：

js label

运算过程为 ：如果PSW的SF为1 ，pc->[label]

如果不为1，跳过这行，什么都不做

Jns 指令（为正跳转）：

jns label

运算过程为 ：如果PSW的SF为0 ，pc->[label]

如果为1，跳过这行，什么都不做

Jz 指令（为0跳转）：

jz label

运算过程为 ：如果PSW的zF为1 ，pc->[label]

如果为0，跳过这行，什么都不做

以上为条件控制指令

Loop 指令（为0循环）：

loop label

运算过程为： 先取得cx的值，如果cx>0, pc->[label],cx--

如果cx<=0,跳过

以上为循环控制指令

Push 指令（压栈）：

push register

运算过程为： 将寄存器的值压入栈中，且栈指针下移

Pop 指令（出栈）：

pop register

运算过程为： 将栈顶的值传给寄存器，且栈指针上移

Call 指令（函数调用）：

call function\_name

运算过程为： 保存当前pc ,然后将pc 指向函数代码段地址

Ret 指令（调用返回）：

ret

运算过程： pc->调用处的下一行

以上为用户函数调用返回指令

Int 指令（系统函数处理）：

Int 21

运算过程： 根据ah的值来确定做什么，dx用于存储将要被使用到的数据：

ah=9: 输出以dx内容为首地址的内存的值，直到该标签快结束

ah=2:输出dx中的值

ah=12:程序结束

以上为系统处理指令

* **Perigee 汇编文件组织：**

首先要声明数据段，以 datas segment 开始 ，以datas ends 结束

如：

DATAS SEGMENT

prompt 3 {f,u,c}

space 2 {s,r}

MATRIX1 13

MATRIX2 2

result 4 {1,2,567,4}

M1 2 {1,2}

M2 3 {n,m,d}

M3 4

DATAS ENDS

数据的声明必须的两部分： 标记 +所要分配的大小

内容的初始化是可选的

初始化必须遵循以下规定： 以 '{ '开始，以 '}' 结束；

初始化可以是数字，可以是字符（不要单引号），数据间以逗号隔开 ，注意不能有空格；

初始化的大小不能大于所声明空间大小。

代码段要在数据段后写，以 codes segment 开始，以codes ends结束

如：

CODES SEGMENT

lea dx,prompt ;显示提示信息

mov ah,09 ;09功能调用，输出字符串

int 21

LEA DX,space ;换行

INT 21

LEA DX,M1 ;提示输入矩阵1

MOV AH,09

INT 21

mov cx,09 ;设置输入字符个数--为9

mov ah,01

mov bx, offset\_MATRIX1

input1 : int 21

mov [bx],al

inc bx

loop input1

lea dx,M2 ;提示输入矩阵2

mov ah,09

int 21

mov cx, 09

mov bx, offset\_MATRIX2

mov ah, 01

codes ends

有偏移offset 关键字的指令 请加下划线\_连标签；

对于函数的声明，请以#结束函数名，在调用中不需要加#；

函数名和函数代码首行要在同一行：

如：

function1# mov cx,03h

xor ax,ax

xor dx,dx

mov al, [bx]

sub al,30h

push dx

mov dl, [bp]

sub dl,30h

ret

* **自定义命令格式：**

help //显示帮助文档

execute file\_name //执行某一文件进程

open file\_name //打开文件

delete file\_name //删除文件

copy source\_file\_name destination\_path //复制文件

cut source\_file\_name destination\_path //剪切文件

dir directory\_file //打开目录文件

* **中断处理流程：**



# 3系统结构设计

* **汇编编译**



* **中断处理**



# 4运行设计

## 4.1运行模块组合

说明对系统施加不同的外界运行控制时所引起的各种不同的运行模块组合，说明每种运行所历经的内部模块和支持软件。

## 4.2运行控制

说明每一种外界的运行控制的方式方法和操作步骤。

## 4.3运行时间

说明每种运行模块组合将占用各种资源的时间。

# 5系统数据结构设计

## 5.1逻辑结构设计要点

给出本系统内所使用的每个数据结构的名称、标识符以及它们之中每个数据项、记录、文卷和系的标识、定义、长度及它们之间的层次的或表格的相互关系。

## 5.2物理结构设计要点

给出本系统内所使用的每个数据结构中的每个数据项的存储要求，访问方法、存取单位、存取的物理关系（索引、设备、存储区域）、设计考虑和保密条件。

## 5.3数据结构与程序的关系

说明各个数据结构与访问这些数据结构的形式:

# 6系统出错处理设计

## 6.1出错信息

用一览表的方式说朗每种可能的出错或故障情况出现时，系统输出信息的形式、含意及处理方法。

## 6.2补救措施

说明故障出现后可能采取的变通措施，包括：

1. 后备技术说明准备采用的后备技术，当原始系统数据万一丢失时启用的副本的建立和启动的技术，例如周期性地把磁盘信息记录到磁带上去就是对于磁盘媒体的一种后备技术；
2. 降效技术说明准备采用的后备技术，使用另一个效率稍低的系统或方法来求得所需结果的某些部分，例如一个自动系统的降效技术可以是手工操作和数据的人工记录；
3. 恢复及再启动技术说明将使用的恢复再启动技术，使软件从故障点恢复执行或使软件从头开始重新运行的方法。

## 6.3系统维护设计

说明为了系统维护的方便而在程序内部设计中作出的安排，包括在程序中专门安排用于系统的检查与维护的检测点和专用模块。 各个程序之间的对应关系，可采用如下的矩阵图的形式；