实验三文档说明

1.本次实验需要完成的任务说明：

1. myOS/start32.S 中的 time_interrupt 和 ignore_int1 的填写
2. myOS/dev/i8253.c 和 myOS/dev/i8259A.c 的填写
3. myOS/i386/irq.s 的填写
4. myOS/kernel/tick.c 和 myOS/kernel/wallClock.c 的填写
5. userApp/startShell.c 的填写

2.小建议

为了更好地完成本次实验和后续实验，进一步理解整个程序是如何运行的，助教强烈大家按如下顺序阅读所有相应的程序

multibootheader/multibootHeader.S →
myOS/start32.S →
myOS/osStart.c →
userApp/startShell.c

3.各个模块的讲解

a) myOS/start32.S 中的 time_interrupt 和 ignore_int1 的填写

在该模块中你需要用到的汇编语言：

cld：将标志寄存器的方向标志位 DF 清零
pushf：将标志寄存器(Flag)中的值存入栈中
popf：将栈中的内容存入标志寄存器
pusha：将通用寄存器(eax, ebx 等)中的值存入栈中
popa：将栈中的内容存入通用寄存器中
call：函数调用(**可以调用 C 语言程序中的函数**)
iret：返回调用该函数的函数

关于 call 和 iret：本质上 call 和 iret 就是 CS 和 IP 寄存器的入栈和出栈，对于 CS 和 IP 寄存器的理解可以对应你们计组课上的 PC(Program Counter，程序计数器)

如何编写这两个函数呢？

1. 利用 push 来进行现场保护
2. call 相应的函数(**可以调用 C 语言程序中的函数**)
3. 利用 pop 来进行恢复现场
4. 利用 iret 来返回调用该函数的函数

b) myOS/dev/i8253.c 和 myOS/dev/i8259A.c 的填写

i8253 和 i8259A 都是可编程芯片，什么叫**可编程逻辑芯片**？就是意味着它可以在配置后进行使用，至于如何配置他们呢？我们只需要用 outb 函数往相应的地址输出相应的数值即可(**老师的 ppt 中有详细的配置地址与相应配置数值说明**)

关于 i8253 和 i8259A 的工作方式说明：

在配置好 i8253 和 i8259A 后，i8253 就相当于一个特定频率的时钟源，而且输出的时钟信号就当作中断信号挂载在 i8259A 上，i8259A 作为一个中断控制器，会使 CPU 去执行相应中断号的中断子程序(**也即 time_interrupt 和 ignore_int1，**大家可以思考一下这两个中断子程序的中断号到底是怎么确定的)

c) myOS/i386/irq.s 的填写

在该模块中你需要用到的汇编语言：

ret：返回调用该函数的函数

sti：开中断

cli：关中断

~~**这一块一共就四行……**~~

d) myOS/kernel/tick.c 的填写

```
#include "wallClock.h"
//在这里助教定义了如下全局变量供大家使用
int system_ticks;//记录 tick 的调用次数
int HH,MM,SS;//分布代表当前时间的“时：分：秒”


//如果你已经完成了 a)和 b)中的任务，那你一定也明白了我们会在什么时候调用 tick 函数吧？
//没错，我们会在由 i8253 引起的时钟中断而引起的中断子程序 time_interrupt 处理中调用
//tick 函数，由于 tick 函数的调用是有固定频率的，所以我们可以用它来进行时钟的输出
void tick(void){//你需要填写完整
    system_ticks ++;
    //你需要完整对 HH,MM,SS 的处理程序
    //……//
    oneTickUpdateWallClock (HH,MM,SS);// 调用的是 wallClock_hook 函数

    return;
```

```
    }
```

d) myOS/kernel/wallClock.c 的填写

```
//在阅读以下程序之前，你需要知道什么是函数指针，并且阅读一下 userApp/main.c 中的程序
//在 userApp/main.c 程序中，我们调用 setWallClockHook(setWallClock)以实现全局变量
//wallClock_hook 被赋值为 setWallClock 函数，从而我们在 tick.c 中调用的
//oneTickUpdateWallClock 的函数调用的 wallClock_hook 函数指针也即为 setWallClock 函数

//我们实现了机制与策略相分离，提供了用户重新编写 mysetWallClock，并调用
//setWallClockHook(mysetWallClock)的权利


void (*wallClock_hook)(int, int, int) = 0;
void oneTickUpdateWallClock(int HH, int MM, int SS){//调用 wallClock_hook 函数
    if(wallClock_hook) wallClock_hook(HH,MM,SS);
}
//
void setWallClockHook(void (*func)(int, int, int)) {//设置 wallClock_hook 的值
    wallClock_hook = func;
}

void setWallClock(int HH,int MM,int SS){//通过 vga 往合适的位置输出 HH:MM:SS，即显示时钟
}

void getWallClock(int *HH,int *MM,int *SS){//根据 vga 显存中的数值，返回时钟，并存到相应
//的指针指向位置中
}
```

e) userApp/startShell.c 的填写

**********要记得使用老师 PPT 中的串口重定向来进行串口的输入哦********

```
//这是助教提供的命令类型定义，可以修改，也可以直接使用
typedef struct myCommand {
    char name[80]; //命令名(可以作为唯一标识符使用)
    char help_content[200]; //该命令的使用说明
    int (*func)(int argc, char (*argv)[8]);//函数指针的概念，不懂的同学需要自行百度
}myCommand;


int func_cmd(int argc, char (*argv)[8]){
    //输出所有命令的命令名
    //你可以设计一个 myCommand 类型的数组，然后遍历它，输出所有命令的命令名
    //由于本实验只需要实现 cmd 和 help 命令，所以也可以直接输出 cmd 的命令名“cmd”和 help
    //的命令名“help”
}

//在这里我们初始化定义了一个类型为 myCommand 的 cmd 命令
myCommand cmd={"cmd\0","List all command\n\0",func_cmd};



int func_help(int argc, char (*argv)[8]){
  //根据 argv 中的内容来输出相应命令的 help_content 属性
  //比如 argv[1]中的字符串为’cmd\0’，我们就输出 cmd 命令的 help_content 属性
}

//在这里我们初始化定义了一个类型为 myCommand 的 help 命令
myCommand help={"help\0","Usage: help [command]\n\0Display info about
[command]\n\0",func_help};


void startShell(void){
    //我们通过串口来实现数据的输入
   char BUF[256]; //输入缓存区
```

```
    int BUF_len=0;    //输入缓存区的长度

    int argc;
    char argv[8][8];

    do{
        BUF_len=0;
        myPrintk(0x07,"Student>>\0");
        while((BUF[BUF_len]=uart_get_char())!='\r'){
            uart_put_char(BUF[BUF_len]);//将串口输入的数存入 BUF 数组中
            BUF_len++;  //BUF 数组的长度加
        }
        uart_put_chars(" -pseudo_terminal\0");
        uart_put_char('\n');

        //OK,助教已经帮助你们实现了“从串口中读取数据存储到 BUF 数组中”的任务，接下来你们要做
        //的就是对 BUF 数组中存取的数据进行处理(也即，从 BUF 数组中提取相应的 argc 和 argv 参
        //数)，再根据 argc 和 argv，寻找相应的 myCommand ***实例，进行***.func(argc,argv)函数
        //调用。

        //比如 BUF 中的内容为 “help cmd”
        //那么此时的 argc 为 2 argv[0]为 help argv[1]为 cmd
        //接下来就是 help.func(argc, argv)进行函数调用即可

    }while(1);
}
```

~~希望看到这句话的时候不是 ddl 的最后几天~~