## 操作系统

## 实验报告

# 班号：\_\_\_\_\_1403202\_\_\_\_\_

# 学号：\_\_\_1140320206\_\_\_\_

# 姓名：\_\_\_\_\_霍峻杰\_\_\_\_\_\_

1. 请简述head.s 的工作原理。
2. 请记录head.s 的内存分布状况，写明每个数据段，代码段，栈段的起始与终止的内存地址。
3. 简述Head.s 57~62行在做什么？简述iret执行后，pc如何找到下一条指令？并记录iret执行前后，栈是如何变化的。
4. 当任务进行系统调用时，即int 0x80时，记录栈的变化情况。当执行完system\_interrupt函数，执行153行iret时，记录栈的变化情况。此外，当进入和退出system\_interrupt时，都发生了模式切换，请总结模式切换时，特权级是如何改变的？栈切换吗？如何进行切换的？
5. 当时钟中断发生，进入到timer\_interrupt程序，请**详细记录**从任务0切换到任务1，以及又过了10ms，从任务1切换回到任务0，整个流程是怎样的？TSS是如何变化的？各个寄存器的值是如何变化的？请详细总结任务切换的过程。
6. 请简述head.s 的工作原理。

在内核程序执行指令jmp 0,8后，通过找到GDT表的描述符1，得到线性基地址和偏移量都是0，进入保护模式，开始执行0x00处的head.s 代码。Head.s代码是运行在32位保护模式下的，总体来说，包括初始化设置代码，时钟中断代码，系统调用中断代码，和两个任务的代码。

首先，加载ds,ss,esp。然后在新的位置重新设置IDT表和GDT表。将IDT的256个中断门都填成默认的处理过程的描述符（处理过程为打印c）。重新初始化后的GDT表包含了8个描述符(第一个描述符不用，第二个是内核代码段，第三个是内核数据段，第四个是显示内存段，剩下四个是TSS0,LDT0,TSS1,LDT1)，然后重新加载所有段寄存器。

然后设置8253定时芯片，把计数器通道0设置成每隔10ms向中断控制器发送一个中断请求信号。设置频率为100Hz，即发送中断周期为0.01s==10ms。在写入初始计数值时，要分两次吧它写入通道0。

现在，IDT表都是默认的中断处理的描述符。要在IDT表第8和第128项处分别设置定时中断门描述符和系统调用陷阱门描述符。和初始化时类似，dx中为中断门类型，eax中高地址为内核代码段选择符0x08，低地址为中断处理程序的地址（定时时钟和陷阱门）。

然后，移动到任务0中执行来操作堆栈内容，在对战中建立中断返回的场景。任务0的TSS段选择符加载到TR，LDT段选择符加载到LDTR，当前任务号保存到current变量。开启中断，ss,esp,eflags,cs,eip压栈。

时钟中断处理，先让ds指向内核数据段，立刻允许其他硬件中断。判断，若是当前任务为0，则把1存入current，去执行任务1，反之，存0，去执行任务0.任务0为打印A，任务1为打印B。跳转执行时，找到GDT中的对应TSS，在TSS中记录了对应的LDT，在LDT中找到对应的eip位置（要执行指令的地址），跳转到此处执行任务，此时TR中的值也随之改变，为当前执行的任务TSS。

1. 请记录head.s 的内存分布状况，写明每个数据段，代码段，栈段的起始与终止的内存地址。

代码段

|  |  |  |
| --- | --- | --- |
| 名称 | 起始地址 | 终止地址 |
| Startup\_32 | 0x00 | 0xac |
| Setup\_idt | 0xb5 | 0xe4 |
| Setup\_gdt | 0xad | 0xb4 |
| Write\_char | 0xe5 | 0x113 |
| Ignore\_int | 0x114 | 0x129 |
| Timer\_interrupt | 0x12b | 0x165 |
| System\_interrupt | 0x166 | 0x17c |
| Task0 | 0x10e0 | 0x10 |
| Task1 | 0x10f4 | 0x1107 |

数据段

|  |  |  |
| --- | --- | --- |
| 名称 | 起始地址 | 终止地址 |
| current | 0x17d | 0x180 |
| Scr\_loc | 0x181 | 0x184 |
| Lidt\_opcode | 0x186 | 0x18b |
| Lgdt\_opcode | 0x18c | 0x191 |
| idt | 0x198 | 0x997 |
| gdt | 0x998 | 0x9d7 |
| Ldt0 | 0xbe0 | 0xbf7 |
| Tss0 | 0xbf8 | 0xc5f |
| Ldt1 | 0xc60 | 0xe77 |
| Tss1 | 0xe78 | 0xedf |

堆栈段

|  |  |  |
| --- | --- | --- |
| 名称 | 起始地址 | 终止地址 |
| Init\_stack | 0x9d8 | 0xbd8 |
| Krn\_stk0 | 0xc60 | 0xe60 |
| Krn\_stk1 | 0xe00 | 0x10e0 |
| User\_stk1 | 0x1108 | 0x1308 |

1. 简述Head.s 57~62行在做什么？简述iret执行后，pc如何找到下一条指令？并记录iret执行前后，栈是如何变化的。

57：把任务0当前局部空间数据段（堆栈段）选择符入栈（ss入栈）

58：堆栈指针入栈（esp入栈）

59：标志寄存器入栈（eflags入栈）

60：当前局部空间代码段选择符入栈（cs入栈）

61：代码指针入栈（eip入栈）

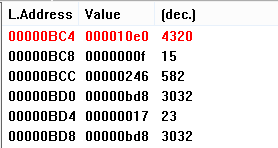
62：终端返回指令，切换到特权级3的任务0中执行。

**如何找到下一条指令：**

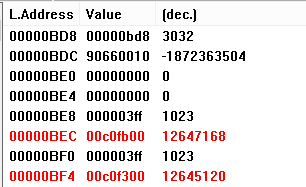
Iret后，堆栈中压入的ss，esp,eflags,cs,ip都会被弹出，并且被恢复，处理器根据cs查找对应的ldt，得到对应的线性基地址，在加上偏移地址ip，得到下一条线性地址0x10e0。

**栈变化：**

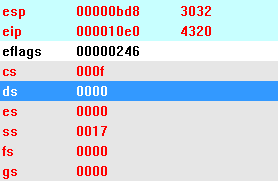
**Iret前**



**Iret后**

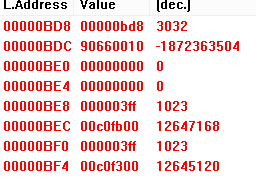


**刚才的寄存器值被还原，eip可以看到是0x10e0**

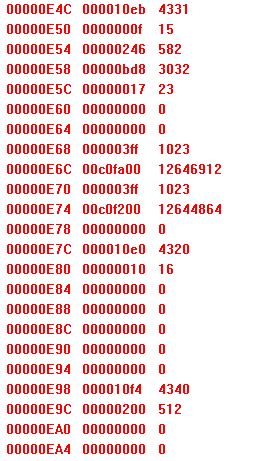


1. 当任务进行系统调用时，即int 0x80时，记录栈的变化情况。当执行完system\_interrupt函数，执行153行iret时，记录栈的变化情况。此外，当进入和退出system\_interrupt时，都发生了模式切换，请总结模式切换时，特权级是如何改变的？栈切换吗？如何进行切换的？

Int 0x80前

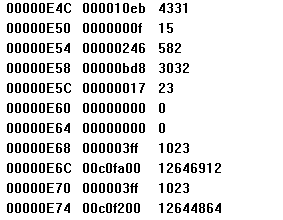


Int 0x80后

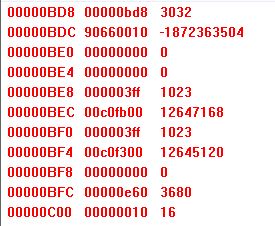


Int 0x80后，堆栈变为这个任务的系统栈，压入了发生中断前的ss,esp,eflags,cs,eip。

当执行完system\_interrupt函数，执行iret前：



执行iret后：



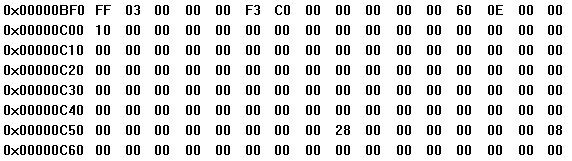
由上述堆栈段变化可以知道，在进入和退出system\_interrupt，都会发生模式切换。因为任务开始的为用户模式，特权级为3，而系统中断后，需要调用内核程序，需要切换到内核模式，使之特权级为0.在进行模式切换的时候，栈也要随之进行切换，用户模式有自己的栈，内核模式也有属于自己的系统栈。

在退出时，特权级3到0是不允许的，但是用中断返回操作iret可以实现(在初始堆栈手动设置返回环境)。

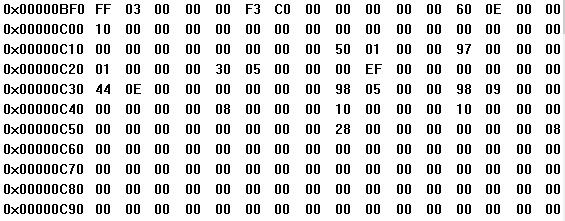
1. 当时钟中断发生，进入到timer\_interrupt程序，请**详细记录**从任务0切换到任务1，以及又过了10ms，从任务1切换回到任务0，整个流程是怎样的？TSS是如何变化的？各个寄存器的值是如何变化的？请详细总结任务切换的过程。

任务0切换到任务1：硬件将任务0的寄存器的值（eflags，cd，es等），状态等存到renwu0的tss中（TSS0）。通过TSS1选择符查找gdt，找到任务1的ldt1，TSS1,加载到对应的寄存器中。跳转，去执行任务1.

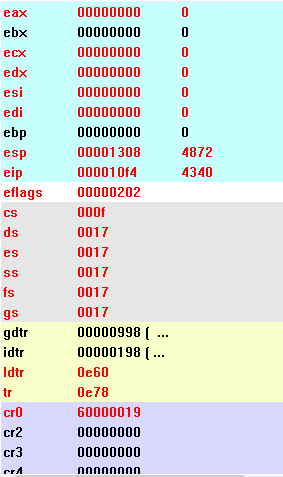
切换前tss0：



切换后tss0：

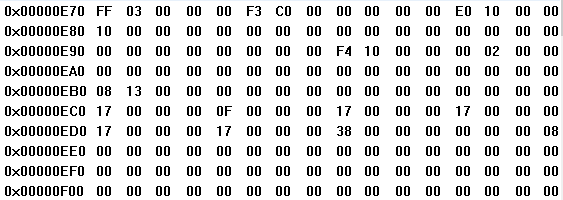


寄存器值：

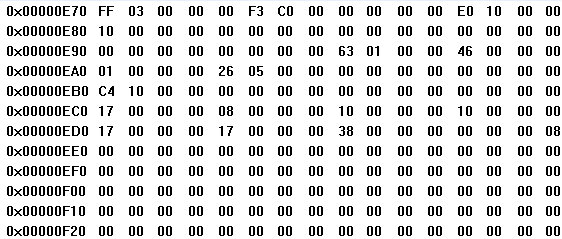


10ms后，任务1切换到任务0：与上面的描述过程相似，不再赘述。

切换前tss1:



切换后tss1：



寄存器值：

