## 操作系统

## 实验报告

# 班号：1403106

# 学号：1140310606

# 姓名：张茗帅

1. **请简述head.s 的工作原理。**

head.s的主要作用是用来在屏幕上打印A和B的。

从调试过程中可以看到，head.s的工作步骤如下：

1. **初始化系统栈**
2. **初始化IDT表（中断描述符表）**

在初始化IDT表的过程中，只是将同一个中段门描述符重复写入IDT表256次（IDT表的内存空间已经被申请好），然后加载IDTR寄存器（中断描述附表寄存器）

1. **初始化GDT表（全局描述符表）**

在初始化GDT表的过程中，GDT表已经事先写好放在内存里，这里只需要加载GDTR寄存器（全局描述符表寄存器）即可

1. **设置8253定是芯片**
2. **修改IDT表的第0x08项和第0x80项**

第0x08项为定时中断门描述符

第0x80项为系统调用陷阱门描述符

1. **为切换到任务0的准备工作（切换至用户模式）**

将EFLAGS中的嵌套任务标志复位

加载任务0的TSS段选择子到任务寄存器TR

加载任务0的LDT段选择到局部描述符表寄存器LDTR

保存当前任务号0到变量current中

开中断，将ss、esp、eflags、cs、eip的值都压入内核栈中，iret时自动将栈中值弹栈到对应的寄存器内，pc根据cs：eip找到用户程序执行。

1. **任务切换**

切换到用户模式task0任务下，task0通过调用系统中断，再由系统中断调用write子程序，在屏幕上不断打印字符A。当10ms时，时钟中断通过判断current的值来判断当前执行的是任务0还是任务1并进行交替切换，切换过程中，会将当前执行的任务现场保存到相应的tss中，当任务切换回该任务后，从tss中恢复全部数据，保证任务切换的连续性。从而能够在屏幕上不断地交替打印A和B。

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**

包含：head.s主程序代码段，各个子程序，中断的代码段

其中主程序的代码段的内存地址为：0x00000000—0x000000ac

（我认为主程序的代码段为：程序开始到压栈之后，iret准备调用任务0时结束）

子程序setup\_gdt的内存地址为：0x000000ad—0x000000b4

子程序setup\_idt的内存地址为：0x000000b5—0x000000e4

子程序write\_char的内存地址为：0x000000e5—0x00000113

默认中断处理程序ignore\_int的内存内存地址为：

0x00000114—0x00000129

定时中断处理程序timer\_interrupt的内存地址为：

0x0000012a—0x00000165

系统调用中断int 0x80处理程序system\_interrupt的内存地址

为：0x00000166—0x0000017c

任务0的代码段的内存地址：0x000010e0—0x000010f3

任务1的代码段的内存地址：0x000010f4—0x00001107

**数据段**

包含：IDT表，GDT表，系统栈，任务0和任务1的LDT（局

部描述符表），任务状态段（TSS），内核栈，用户栈等，其中：

IDT表的内存地址为：0x00000198—0x00000998

GDT表的内存地址为：0x00000998—0x000009d8

任务0的LDT表的内存地址为：0x00000be0—0x00000bf8

任务0的TSS的内存地址为：0x00000bf8—0x00000c60

任务1的LDT表的内存地址为：0x00000e60—0x00000e78

任务1的TSS的内存地址为：0x00000e78—0x000000ee0

**栈段**

包含：系统内核栈，任务0的内核栈和用户栈，任务1的内

核栈和用户栈

系统栈（内核栈）的内存地址为：0x000009d8—0x00000bd8

任务0的内核栈的内存地址为：0x00000c60—0x00000e60

任务0的用户栈的内存地址为：0x000009d8—0x00000bd8

（任务0的用户栈与系统程序内核栈是同一块区域）

任务1的内核栈的内存地址为：0x00000ee0—0x000010e0

任务1的用户栈的内存地址为：0x00001108—0x000001308

**下面表格中是head.s的内存分布情况**

|  |  |  |
| --- | --- | --- |
| 名称 | 起始地址 | 终止地址 |
| head.s程序代码段  （包含子程序和中断处理程序） | 0x00000000 | 0x0000017c |
| Main(到书中第62行的iret指令) | 0x00000000 | 0x000000ac |
| Setup\_gdt | 0x000000ad | 0x000000b4 |
| Setup\_idt | 0x000000b5 | 0x000000e4 |
| Write\_char | 0x000000e5 | 0x00000113 |
| Ignore\_int | 0x00000114 | 0x00000129 |
| Timer\_interrupt | 0x0000012a | 0x00000165 |
| System\_interrupt | 0x00000166 | 0x0000017c |
| current | 0x0000017d | 0x00000180 |
| scr\_loc | 0x00000181 | 0x00000184 |
| lidt\_opcode | 0x00000186 | 0x0000018b |
| lgdt\_opcode | 0x0000018c | 0x00000191 |
| IDT表 | 0x00000198 | 0x00000998 |
| GDT表 | 0x00000998 | 0x000009d8 |
| 系统栈（内核栈） | 0x000009d8 | 0x00000bd8 |
| init\_stack信息 | 0x00000bd8 | 0x00000bdd |
| 任务0的LDT表 | 0x00000be0 | 0x00000bf8 |
| 任务0的TSS | 0x00000bf8 | 0x00000c60 |
| 任务0的内核栈 | 0x00000c60 | 0x00000e60 |
| 任务1的LDT表 | 0x00000e60 | 0x00000e78 |
| 任务1的TSS | 0x00000e78 | 0x000000ee0 |
| 任务1的内核栈 | 0x00000ee0 | 0x000010e0 |
| 任务0的代码段 | 0x000010e0 | 0x000010f3 |
| 任务1的代码段 | 0x000010f4 | 0x00001107 |
| 任务0的用户栈（同系统栈（内核栈）） | 0x000009d8 | 0x00000bd8 |
| 任务1的用户栈 | 0x00001108 | 0x000001308 |

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。**

①57~62行在进行内核程序和用户程序切换的准备工作：在内核模式下将ss、esp、eflags、cs、eip的值都压入内核栈中，iret时自动将栈中值弹栈到对应的寄存器内：

Ss=0x17，esp=0xbd8，eflags=0x246，cs=0x0f，eip=0x10e0

②Cs = 0000000000001111，根据表述符格式，从右向左查找，前两位11表示用户模式，之后1表示查LDT表，之后1表示LDT表中的第一项。根据57行之前对ldtr装载的内存地址，此时查找的ldt表为ldt0，ldt0的第一项为0x00c0fb00000003ff ，从而得到代码的基地址为0x00000000。Pc根据地址0x00000000：0x10e0找到要执行的下一条指令，此时已经来到用户模式下的task0。(ldt0与书中给出的略有差别，p.s.书中是第一项0x00c0fa00000003ff)：

**ldt0第一项**

注：在保护模式下，段选择子解释如下：

15 3 2 1 0

|  |  |  |
| --- | --- | --- |
| Index | TI | RPL |

cs经过对应解释之后可得到信息如下：

要执行的任务0的代码段的RPL = 3（11），TI = 1，index = 0x01即需要在任务0的LDT中查找第0x01项描述符，经过查找可获得描述符表如下：

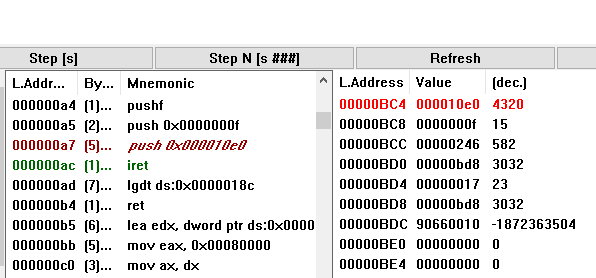
|  |  |  |  |
| --- | --- | --- | --- |
| 00 | C0 | F8 | 00 |
| 00 | 00 | 03 | FF |

上表格中红框内即使通过cs获得的基地址0x00000000

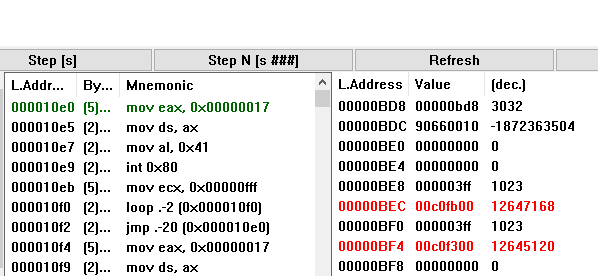
eip = 基地址+偏移地址 = 0x00000000+$task0 = 0x10e0

③iret前栈中情况：

iret指令执行前，栈内的情况如下，可以看出，此时往系统栈中push了五个值，栈顶为00000BC4



执行完iret指令后，栈内情况如下，此时栈顶为0x00000bd8，但此时的栈已经不是前面程序执行的系统栈了，而是任务0的用户栈，因为在iret之后，ss的值等于0x00000017，运用类似于上面的cs解释过程后可得到index = 0x10，TI = 1，RPL = 3（11）。



此时该栈的特权级为3，因为index = 0x10，TI = 1，所以需要在任务0的LDT中找到第0x10项描述符，如下

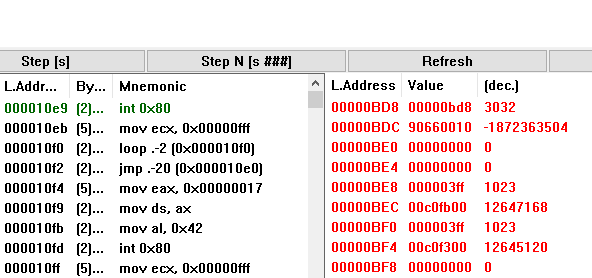
|  |  |  |  |
| --- | --- | --- | --- |
| 00 | C0 | F3 | 00 |
| 00 | 00 | 03 | FF |

则可找到任务0的用户栈段

1. **当任务进行系统调用时，即int 0x80时，记录栈的变化情况。当**

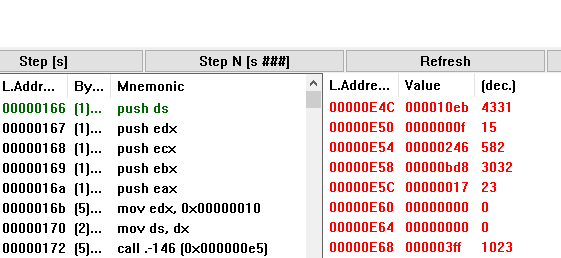
**执行完system\_interrupt函数，执行153行iret时，记录栈的变化情况。此外，当进入和退出system\_interrupt时，都发生了模式切换，请总结模式切换时，特权级是如何改变的？栈切换吗？如何进行切换的？**

在执行int 0x80指令前，任务0的用户栈的情况如下：



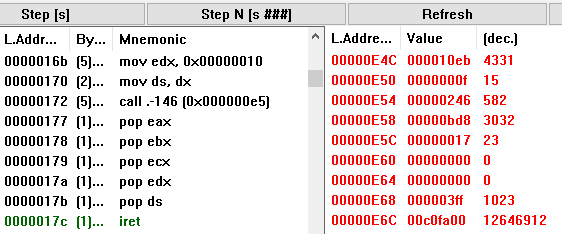
可以看到此时的栈顶为0x00000bd8，栈中没有值

在执行int 0x80指令后，此时任务0切换到内核栈，

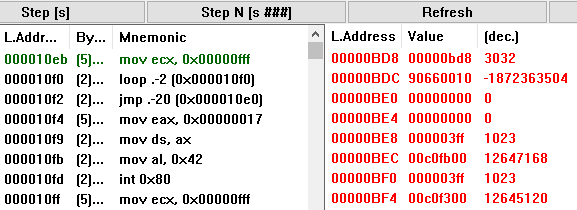


栈顶为0x00000E4C，此时栈中压的五个值是执行int 0x80，保护现场时push进栈里的。

执行153行iret指令前，任务0内核栈的情况如下



执行iret指令后，此时任务0切换到用户栈：



此时栈顶为0x00000bd8，栈内无值

**当发生模式切换时**，cs和ss都会重新被赋值

**当进入system\_interrupt时**（即执行int 0x80指令），在IDT表中第0x80项描述符为

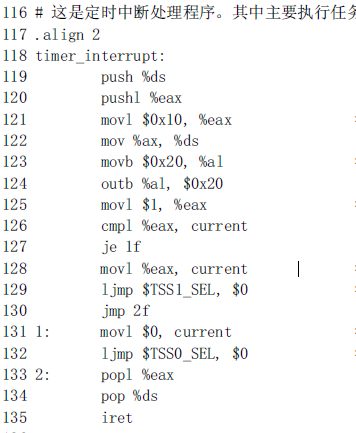
|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 00 | EF | 00 |
| 00 | 08 | 01 | 66 |

上图中的红框内为段选择子，被赋值于cs，cs由0x0f变为0x08，将该选择子解释后，特权级为0（00），故特权级由3（11）变为了0（00）。ss由0x17变为0x10。从上述可以看出：任务0由用户模式切换到了内核模式，同时栈也进行了切换，从当前任务的用户栈切换到了当前任务的内核栈。

**当退出system\_interrupt时**，重新执行到任务的代码段，恢复现场后，cs = 0x0f，ss = 0x17，eip指向指令int 0x80的下一句，esp = 0xbd8，eflags = 0x246。相应的特权级由0（00）变为了3（11），即从内核模式切换到了用户模式。同时栈也进行了切换，从当前任务的内核栈切换到了当前任务的用户栈。

**总之，模式切换必然会伴随着栈的切换**

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。**

****

执行完119—124行后，开始进入判断

125行:给eax赋值为1

126行:比较eax中的值与当前执行任务current的值

127行:如果eax中的值与current相等，及当前执行任务1，就

跳转131行，否则执行下一句

128行:如果eax中的值与current不等，则执行该句，将eax寄

存器的值赋给current

` 129行:跳转任务0

130行:跳转到133行

131行:将current赋值为0

132行:跳转到任务0

133—135行后iret

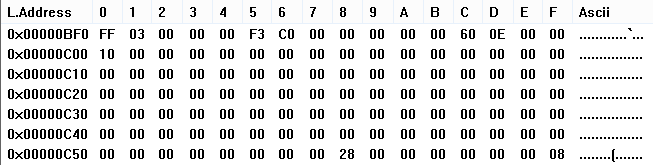
**①切换流程：**

任务0运行10ms，发生时钟中断执行timer\_interrupt，根据current的值（当前为0）进行切换：当前值为0则改为1，同时进行从任务0到任务1的切换，此时先将各种寄存器的值保存到tss0里，再将tss1中的值弹到对应寄存器中，pc根据cs：eip找到任务1程序起始地址开始执行任务1。

任务1执行10ms，发生时钟中断执行timer\_interrupt，将current的值由1改成0，同时进行从任务1到任务0的切换，此时先将各种寄存器的值保存到tss1里，再将tss0中的值弹到对应寄存器中，由于pc指向任务0切换离开时的下一条指令，则会继续执行完timer\_interrupt程序之后回到任务0程序离开时的下一条，继续打印A的工作。

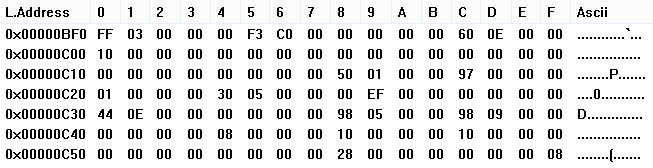
**②任务0切换到任务1时：**

**Tss0（0xbf8~0xc5f）切换前：**



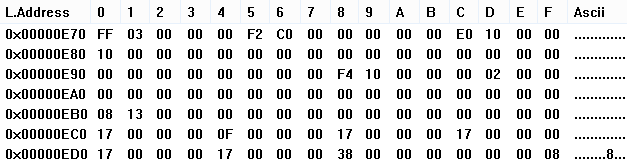
|  |  |  |
| --- | --- | --- |
| (I/0 Map Base Address)08 00 | | Reserved00 00 |
| (Reserved)00 00 | | (DT 段选择子)0 28 |
| (Reserved)00 00 | | (GS)00 00 |
| (Reserved)00 00 | | (FS)00 00 |
| (Reserved)00 00 | | (DS)00 00 |
| (Reserved)00 00 | | (SS)00 00 |
| (Reserved)00 00 | | (CS)00 00 |
| (Reserved)00 00 | | (ES)00 00 |
| (EPI)00 00 00 00 | | |
| (ESI)00 00 00 00 | | |
| (EBP)00 00 00 00 | | |
| (ESP)00 00 00 00 | | |
| (EB)X00 00 00 00 | | |
| (EDX)00 00 00 00 | | |
| (ECX)00 00 00 00 | | |
| (EAX)00 00 00 00 | | |
| (EFALGS)00 00 00 00 | | |
| (EIP)00 00 00 00 | | |
| (CR3（PDBR）)00 00 00 00 | | |
| （Reserved）00 00 | (SS2)00 00 | |
| （ESP2）00 00 00 00 | | |
| (Reserved)00 00 | (SS1)00 00 | |
| (ESP1)00 00 00 00 | | |
| (Reserved)00 00 | (SS0)00 10 | |
| (ESP0)00 00 0E 60 | | |
| (Reserved)00 00 | (Previous Task Link)00 00 | |

**Tss0切换后：**



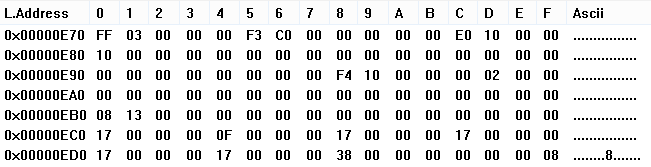
|  |  |  |
| --- | --- | --- |
| (I/0 Map Base Address)08 00 | | Reserved00 00 |
| (Reserved)00 00 | | (DT 段选择子)00 28 |
| (Reserved)00 00 | | (GS)00 00 |
| (Reserved)00 00 | | (FS)00 00 |
| (Reserved)00 00 | | (DS)00 10 |
| (Reserved)00 00 | | (SS)00 10 |
| (Reserved)00 00 | | (CS)00 08 |
| (Reserved)00 00 | | (ES)00 00 |
| (EPI)00 00 09 98 | | |
| (ESI)00 00 05 98 | | |
| (EBP)00 00 00 00 | | |
| (ESP)00 00 0E 44 | | |
| (EBX)00 00 00 00 | | |
| (EDX)00 00 EF 00 | | |
| (ECX)00 00 05 30 | | |
| (EAX)00 00 00 01 | | |
| (EFALGS)00 00 00 97 | | |
| (EIP)00 00 01 50 | | |
| (CR3（PDBR）)00 00 00 00 | | |
| （Reserved）00 00 | (SS2)00 00 | |
| （ESP2）00 00 00 00 | | |
| (Reserved)00 00 | (SS1)00 00 | |
| (ESP1)00 00 00 00 | | |
| (Reserved)00 00 | (SS0)00 10 | |
| (ESP0)00 00 0E 60 | | |
| (Reserved)00 00 | (Previous Task Link)00 00 | |

**Tss1（0xe78~0xedf）切换前：**



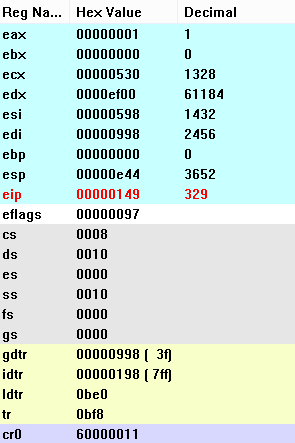
|  |  |
| --- | --- |
| (I/0 Map Base Address)08 00 | Reserved00 00 |
| (Reserved)00 00 | (DT 段选择子)00 38 |
| (Reserved)00 00 | (GS)00 17 |
| (Reserved)00 00 | (FS)00 17 |
| (Reserved)00 00 | (DS)00 17 |
| (Reserved)00 00 | (SS)00 17 |
| (Reserved)00 00 | (CS)00 0F |
| (Reserved)00 00 | (ES)00 17 |
| (EPI)00 00 00 00 | |
| (ESI)00 00 00 00 | |
| (EBP)00 00 00 00 | |
| (ESP)00 00 13 08 | |
| (EBX)00 00 00 00 | |
| (EDX)00 00 00 00 | |
| (ECX)00 00 00 00 | |
| (EAX)00 00 00 00 | |
| (EFALGS)00 00 02 00 | |
| (EIP)00 00 10 F4 | |
| (CR3（PDBR）)00 00 00 00 | |

**Tss1切换后：**

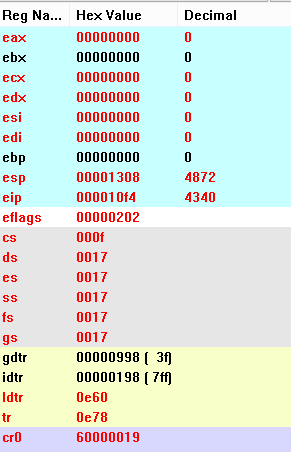


|  |  |  |
| --- | --- | --- |
| (I/0 Map Base Address)08 00 | | Reserved00 00 |
| (Reserved)00 00 | | (DT 段选择子)00 38 |
| (Reserved)00 00 | | (GS)00 17 |
| (Reserved)00 00 | | (FS)00 17 |
| (Reserved)00 00 | | (DS)00 17 |
| (Reserved)00 00 | | (SS)00 17 |
| (Reserved)00 00 | | (CS)00 0F |
| (Reserved)00 00 | | (ES)00 17 |
| (EPI)00 00 00 00 | | |
| (ESI)00 00 00 00 | | |
| (EBP)00 00 00 00 | | |
| (ESP)00 00 13 08 | | |
| (EBX)00 00 00 00 | | |
| (EDX)00 00 00 00 | | |
| (ECX)00 00 00 00 | | |
| (EAX)00 00 00 00 | | |
| (EFALGS)00 00 02 00 | | |
| (EIP)00 00 10 F4 | | |
| (CR3（PDBR）)00 00 00 00 | | |
| （Reserved）00 00 | (SS2)00 00 | |
| （ESP2）00 00 00 00 | | |
| (Reserved)00 00 | (SS1)00 00 | |
| (ESP1)00 00 00 00 | | |
| (Reserved)00 00 | (SS0)00 10 | |
| (ESP0)00 00 10 E0 | | |
| (Reserved)00 00 | (Previous Task Link)00 00 | |

**各种寄存器切换前：**

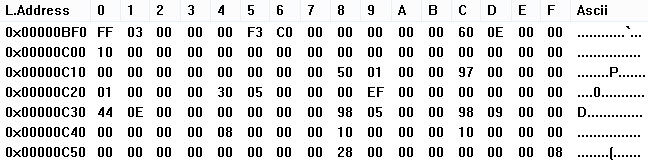


**各种寄存器切换后：**



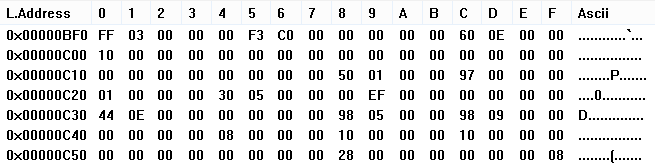
**③从任务1切换到任务0时：**

**Tss0切换前：**



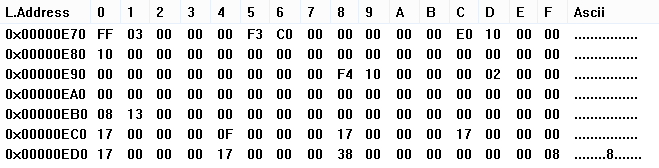
|  |  |  |
| --- | --- | --- |
| (I/0 Map Base Address)08 00 | | Reserved00 00 |
| (Reserved)00 00 | | (DT 段选择子)00 28 |
| (Reserved)00 00 | | (GS)00 00 |
| (Reserved)00 00 | | (FS)00 00 |
| (Reserved)00 00 | | (DS)00 10 |
| (Reserved)00 00 | | (SS)00 10 |
| (Reserved)00 00 | | (CS)00 08 |
| (Reserved)00 00 | | (ES)00 00 |
| (EPI)00 00 09 98 | | |
| (ESI)00 00 05 98 | | |
| (EBP)00 00 00 00 | | |
| (ESP)00 00 0E 44 | | |
| (EBX)00 00 00 00 | | |
| (EDX)00 00 EF 00 | | |
| (ECX)00 00 05 30 | | |
| (EAX)00 00 00 01 | | |
| (EFALGS)00 00 00 97 | | |
| (EIP)00 00 01 50 | | |
| (CR3（PDBR）)00 00 00 00 | | |
| （Reserved）00 00 | (SS2)00 00 | |
| （ESP2）00 00 00 00 | | |
| (Reserved)00 00 | (SS1)00 00 | |
| (ESP1)00 00 00 00 | | |
| (Reserved)00 00 | (SS0)00 10 | |
| (ESP0)00 00 0E 60 | | |
| (Reserved)00 00 | (Previous Task Link)00 00 | |

**Tss0切换后：**



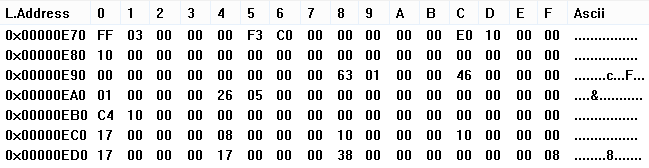
|  |  |  |
| --- | --- | --- |
| (I/0 Map Base Address)08 00 | | Reserved00 00 |
| (Reserved)00 00 | | (DT 段选择子)00 28 |
| (Reserved)00 00 | | (GS)00 00 |
| (Reserved)00 00 | | (FS)00 00 |
| (Reserved)00 00 | | (DS)00 10 |
| (Reserved)00 00 | | (SS)00 10 |
| (Reserved)00 00 | | (CS)00 08 |
| (Reserved)00 00 | | (ES)00 00 |
| (EPI)00 00 09 98 | | |
| (ESI)00 00 05 98 | | |
| (EBP)00 00 00 00 | | |
| (ESP)00 00 0E 44 | | |
| (EBX)00 00 00 00 | | |
| (EDX)00 00 EF 00 | | |
| (ECX)00 00 05 30 | | |
| (EAX)00 00 00 01 | | |
| (EFALGS)00 00 00 97 | | |
| (EIP)00 00 01 50 | | |
| (CR3（PDBR）)00 00 00 00 | | |
| （Reserved）00 00 | (SS2)00 00 | |
| （ESP2）00 00 00 00 | | |
| (Reserved)00 00 | (SS1)00 00 | |
| (ESP1)00 00 00 00 | | |
| (Reserved)00 00 | (SS0)00 10 | |
| (ESP0)00 00 0E 60 | | |
| (Reserved)00 00 | (Previous Task Link)00 00 | |

**Tss1切换前：**



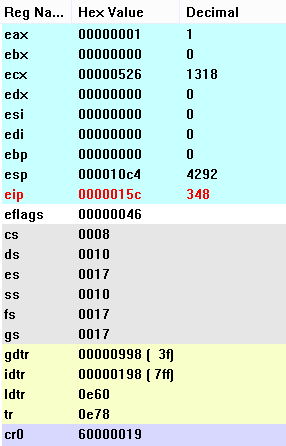
|  |  |  |
| --- | --- | --- |
| (I/0 Map Base Address)08 00 | | Reserved00 00 |
| (Reserved)00 00 | | (DT 段选择子)00 38 |
| (Reserved)00 00 | | (GS)00 17 |
| (Reserved)00 00 | | (FS)00 17 |
| (Reserved)00 00 | | (DS)00 17 |
| (Reserved)00 00 | | (SS)00 17 |
| (Reserved)00 00 | | (CS)00 0F |
| (Reserved)00 00 | | (ES)00 17 |
| (EPI)00 00 00 00 | | |
| (ESI)00 00 00 00 | | |
| (EBP)00 00 00 00 | | |
| (ESP)00 00 13 08 | | |
| (EBX)00 00 00 00 | | |
| (EDX)00 00 00 00 | | |
| (ECX)00 00 00 00 | | |
| (EAX)00 00 00 00 | | |
| (EFALGS)00 00 02 00 | | |
| (EIP)00 00 10 F4 | | |
| (CR3（PDBR）)00 00 00 00 | | |
| （Reserved）00 00 | (SS2)00 00 | |
| （ESP2）00 00 00 00 | | |
| (Reserved)00 00 | (SS1)00 00 | |
| (ESP1)00 00 00 00 | | |
| (Reserved)00 00 | (SS0)00 10 | |
| (ESP0)00 00 10 E0 | | |
| (Reserved)00 00 | (Previous Task Link)00 00 | |

**Tss1切换后：**

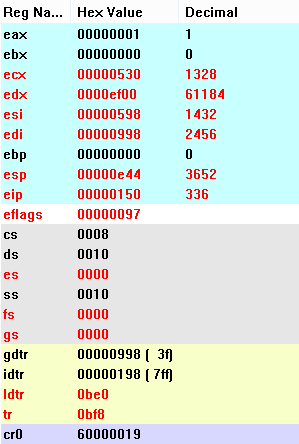


|  |  |  |
| --- | --- | --- |
| (I/0 Map Base Address)08 00 | | Reserved00 00 |
| (Reserved)00 00 | | (DT 段选择子)00 38 |
| (Reserved)00 00 | | (GS)00 17 |
| (Reserved)00 00 | | (FS)00 17 |
| (Reserved)00 00 | | (DS)00 10 |
| (Reserved)00 00 | | (SS)00 10 |
| (Reserved)00 00 | | (CS)00 08 |
| (Reserved)00 00 | | (ES)00 17 |
| (EPI)00 00 00 00 | | |
| (ESI)00 00 00 00 | | |
| (EBP)00 00 00 00 | | |
| (ESP)00 00 10 C4 | | |
| (EBX)00 00 00 00 | | |
| (EDX)00 00 00 00 | | |
| (ECX)00 00 05 26 | | |
| (EAX)00 00 00 01 | | |
| (EFALGS)00 00 00 46 | | |
| (EIP)00 00 01 63 | | |
| (CR3（PDBR）)00 00 00 00 | | |
| （Reserved）00 00 | (SS2)00 00 | |
| （ESP2）00 00 00 00 | | |
| (Reserved)00 00 | (SS1)00 00 | |
| (ESP1)00 00 00 00 | | |
| (Reserved)00 00 | (SS0)00 10 | |
| (ESP0)00 00 10 E0 | | |
| (Reserved)00 00 | (Previous Task Link)00 00 | |

**各种寄存器切换前：**



**各种寄存器切换后：**



**④切换总结**：

发生任务切换时，当前任务的各种寄存器的当前值将自动全部保存进当前任务的TSS中，将要执行的新任务的TSS中的值弹到对应的各种寄存器中。这个过程由硬件自动完成，实现了当前任务的现场保存和新任务的装载，这样每当切换到一个新任务时，就能够通过相应寄存器的值正常地执行程序。比如，通过ldtr找到ldt表，得到新任务代码基地址，通过eip找到新任务程序首地址，通过ss和esp找到新任务的栈等等。