1 系统调用

2 LDT 的实现

2.1 LDT 的定义

首先我们需要在 GDT 表中增加局部描述符表的描述符。

```
        [SECTION .gdt]
        LABEL_GDT: Descriptor 0,LDT
        ; 添加的描述符
        LABEL_DESC_LDT: Descriptor 0,LDTLen-1,DA_LDT
        ; 相应的选择符
        SelectorLDT equ LABEL_DESC_LDT - LABEL_GDT
```

然后我们需要定义一个 LDT 表。LDT 表和 GDT 表其实很类似，我在其中定义了一个指向 CODEA 代码段的段描述符。

```
        ; 定义LDT表
        [SECTION .ldt]
        ALIGN 32
        LABEL_LDT:
        LABEL_LDT_DESC_CODEA: Descriptor 0,CodeALen-1,DA_C+DA_32
        LDTLEN equ $-LABEL_LDT

        SelectorLDTCodeA equ LABEL_LDT_DESC_CODEA-LABEL_LDT+4
        LDTLen equ $-LABEL_LDT

        ; 定义在LDT表中段描述符指向的代码段
        [SECTION .la]
        ALIGN 32
        [BITS 32]
        LABEL_CODE_A:
            mov ax,SelectorVideo
            mov gs,ax
            mov edi,(80*12+0)*2
            mov ah,0Ch
            mov al,'L'
            mov [gs:edi],ax
        CodeALen equ $-LABEL_CODE_A
```

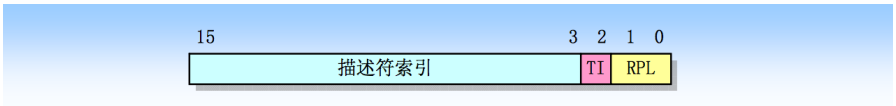
上面代码段中，我们应该注意的是这个语句。

```
        SelectorLDTCodeA equ LABEL_LDT_DESC_CODEA-LABEL_LDT+4
```

这个选择符的定义和 GDT 选择符的定义有不同。在解释为什么这么写之前，我想重新说一下自己对段选择符的认识。因为在第二次学习报告中，我对它的描述实在过于粗糙。

段选择符是段的一个 16 位标志符。段选择符并不直接指向段，而是指向段描述符表

中定义段的段描述符。段选择符的结构如下图。



从图中可以看出，段选择符有 3 个字段内容:

- 请求特权级 RPL。RPL 被用于特权级保护机制中。
- 表指示标志 TI。当 TI=0 时，表示描述符在 GDT 中。当 TI=1 时，表示描述符在 LDT 中。因为一个任务执行时，可以同时访问到 LDT 和 GDT，所以必须做这样的区别，以防在索引时放生混淆。
- 索引值。用于索引在 GDT 表或 LDT 表中的段描述符。

```
1        ; 这里特意加4，就是为了将段选择符中的第2位TI标志置一
2        SelectorLDTCodeA  equ  LABEL_LDT_DESC_CODEA-LABEL_LDT+4
```

2.2 LDT 的初始化

需要注意的是，既然在 GDT 表中添加了指向 LDT 表的段描述符，就应该在 16 位代码段中初始化它。

```
1        [SECTION  .16]
2        [BITS  16]
3
4            ; 初始化LDT在GDT中的描述符
5            xor  eax,eax
6            mov  ax,ds
7            shl  eax,4
8            add  eax,LABEL_LDT
9            mov  word  [LABEL_DESC_LDT + 2],  ax
10           shr  eax,16
11           mov  byte  [LABEL_DESC_LDT + 4],  al
12           mov  byte  [LABEL_DESC_LDT + 7],  ah
```

LDT 表和 GDT 表区别仅仅在于全局和局部的不同，所以初始化 LDT 表中描述符和之前的操作很类似。具体情况看下面的代码。

```
1        [SECTION  .16]
2        [BITS  16]
3            ; 初始化LDT表中的描述符
4            xor  eax,eax
5            mov  ax,ds
6            shl  eax,4
7            add  eax,LABEL_CODE_A
```

```
8               mov word [LABEL_LDT_DESC_CODEA + 2], ax
9               shr eax,16
10              mov byte [LABEL_LDT_DESC_CODEA + 4], al
11              mov byte [LABEL_LDT_DESC_CODEA + 7], ah
```

2.3 调用 LDT 中的代码段

```
1           [SECTION .s32]
2           [BITS 32]
3               ; 将LDT表的段选择符加载进LDTR寄存器中
4               ; 在LDTR寄存器中的LDT段描述符存放着LDT表的基址
5               ; 对LDT表中的代码段描述符进行寻址时，以LDTR中LDT表的描述符中的基址为准
6               ; 偏移量由段选择符制定，用于索引LDT表中存放着的代码段描述符
7               ; 当发生任务切换时，LDTR会更换新任务的LDT
8               mov ax,SelectorLDT
9               lldt ax
10              ; CPU根据代码段描述符中的TI标志判断是索引GDT表还是索引LDT表
11              jmp SelectorLDTCodeA:0
```

2.4 总结如何添加 LDT 表

- 添加一个 LDT 表，里面可以类似于 GDT 表，存放代码段、数据段或堆栈段。
- 在 GDT 表中添加新 LDT 表的段描述符。
- 在 16 位代码段中初始化新 LDT 表的段描述符。同时初始化 LDT 表中存放着的所有段描述符。