电子科技大学 计算机科学与工程学院

实验指导书

（实验）课程名称： 计算机操作系统

电子科技大学教务处制表

内存地址转换实验

- 实验所属系列：操作系统课程实验
- 实验对象：本科
- 相关课程及专业：计算机操作系统，C 语言，数据结构；计算机专业
- 实验类型：配套上机
- 实验时数： 2 学时

1. 实验目的

（1）掌握计算机的寻址过程

（2）掌握页式地址地址转换过程

（3）掌握计算机各种寄存器的用法

2. 实验内容

本实验运行一个设置了全局变量的循环程序，通过查看段寄存器，LDT 表，GDT 表等信息，经过一系列段、页地址转换，找到程序中该全局变量的物理地址。

3. 实验环境

Linux 内核（0.11）+Bochs 虚拟机

4. 实验相关知识

4.1 逻辑地址到线性地址的转换

逻辑地址：Intel 段式管理中：，“一个逻辑地址，是由一个段标识符加上一个指定段内相对地址的偏移量，表示为 [段标识符：段内偏移量]。”

段标识符：也称为段选择符，段标识符是由一个 16 位长的字段组成，其中前 13 位是一个索引号。后面 3 位包含一些硬件细节：

图 1 段选择符

索引号： 可以看作是段的编号，也可以看做是相关段描述符在段表中的索引位置。系统中的段表有两类：GDT 和 LDT。

GDT：全局段描述符表，整个系统一个，GDT 表中存放了共享段的描述符，以及 LDT 的描述符（每个 LDT 本身被看作一个段）

LDT：局部段描述符表，每个进程一个，进程内部的各个段的描述符，就放在 LDT 中。

T1 字段：Intel 设计思想是：一些全局的段描述符，就放在“全局段描述符表(GDT)”中，一些局部的，例如每个进程自己的，就放在所谓的“局部段描述符表(LDT)”中。那究竟什么时候该用 GDT，什么时候该用 LDT 呢？这是由段选择符中的 T1 字段表示的，T1=0，表示相应的段描述符在 GDT 中，T1=1 表示表示相应的段描述符在 LDT 中。

段描述符：具体描述了一个段。在段表中，存放了很多段描述符。我们可以通过段标识符的前 13 位，直接在段描述符表中找到一个具体的段描述符，也就是说，段标识符的前 13 位是相关段描述符在段表中的索引位置。

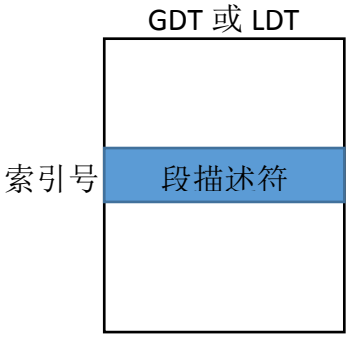

图 2 GDT 或 LDT 示例

每一个段描述符由 8 个字节组成，如图 3：

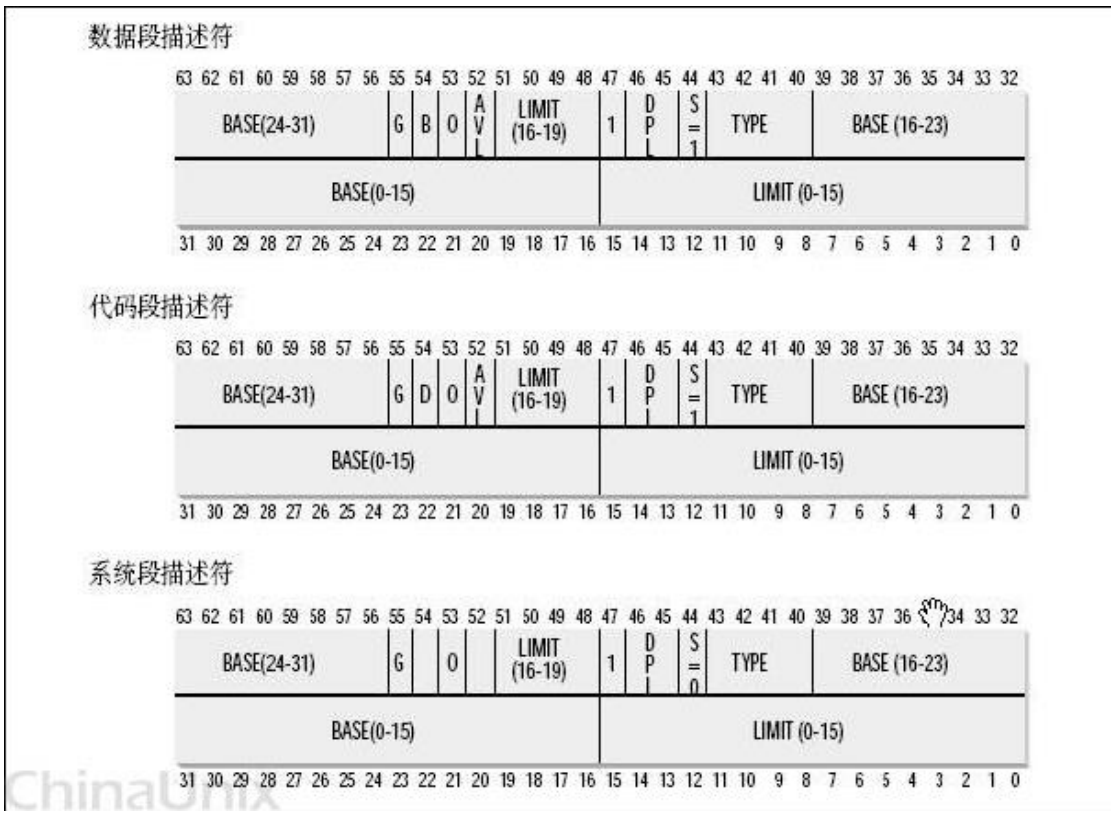

图 3 段描述符

Base 字段：它描述了一个段的开始位置：段基址。Base(24-31)：基地址的高 8 位，

Base(16-23)：基地址的中间 8 位，Base(0-15)：基地址的低 16 位。（这里的段基址，不是相应的段在内存中的起始地址，而是程序编译链接以后，这个段在程序逻辑(虚拟)地址空间里的起始位置。）

相关寄存器：

GDTR：存放 GDT 在内存中的起始地址和大小

LDTR：分两种情况：

(1) 当段选择符中的 T1=1 时，表示段描述符存放在 LDT 中，如何找到 LDT 呢，LDT 本身也被看作一个段，LDT 的起始地址存放在 GDT 中，此时 LDTR 存放的就是 LDT 在 GDT 中的索引。这也是本实验关注的情况。

(2) 当段选择符中的 T1=0 时，表示段描述符存放在 GDT 中，通过 GDTR 找到 GDT，此时 LDTR 存放的是 LDT 的起始地址，当 T1=0 时，不涉及对 LDT 和 LDTR 的使用。

段选择符：如在 DS，SS 等寄存器内存储，取高 13 位作为在相应段表（如上例中

的 DS 的高 13 位为对应段在 LDT）中的索引。

线性地址： 段标识符用来标明一个段的编号，具体的，我们需要通过段的编号，查找段表，来获得这个段的起始地址，即段基址。如前所述，这里的段基址，不是相应的段在内存中的起始地址，而是程序编译链接以后，这个段在逻辑地址空间里的起始位置。进一步的，段基地址＋段内偏移量，就得到线性地址（即要访问的数据在整个程序逻辑(虚拟)地址空间中的位置）。

从逻辑地址到线性地址的转换过程，如图 4 所示（以 T1=1 为例，此时从段选择符中分离出段描述符和 T1 字段，T1=1，表明段描述符存放在 LDT 中）；

（1）从 GDTR 中获得 GDT 的地址，从 LDTR 中获得 LDT 在 GDT 中的偏移量，查找 GDT，从中获取 LDT 的起始地址；

（2）从 DS 中的高 13 位获取 DS 段在 LDT 中索引位置，查找 LDT，获取 DS 段的段描述符，从而获取 DS 段的基地址；

（3）根据 DS 段的基地址＋段内偏移量，获取所需单元的线性地址。

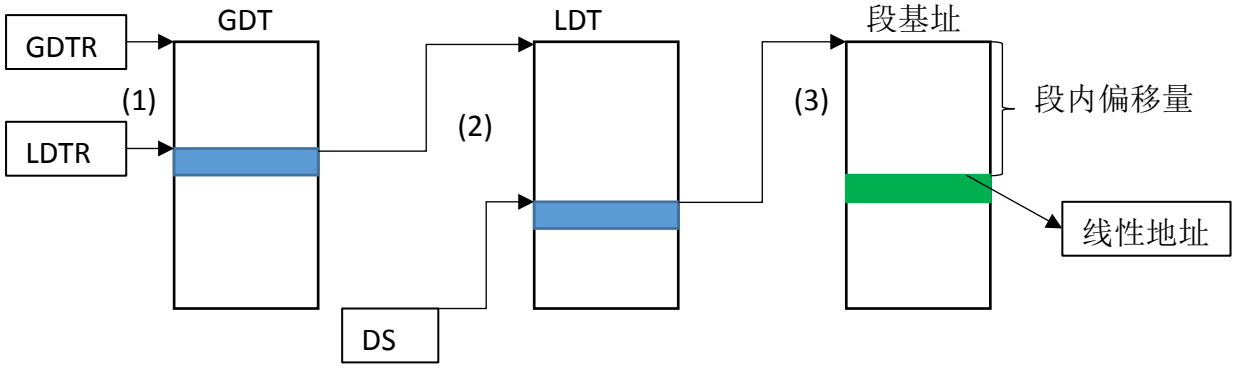

图 4 逻辑地址到线性地址的转换

4.2 线性地址到物理地址的转换

物理地址：分段是面向用户，而分页则是面向系统，以提高内存的利用率，简言之，内存空间是按照分页来管理的。一个 32 位的机器，支持的内存空间是 4G，在页面大小为 4KB 的情况下，如果采用二级分页管理方式，线性地址结构

如图 5 所示。

每一个 32 位的线性地址被划分为三部份， 页目录索引(10 位)：页表索引(10 位)：偏移(12 位，因为页面大小为 4K)。最终，我们需要根据线性地址，来获得物理地址。

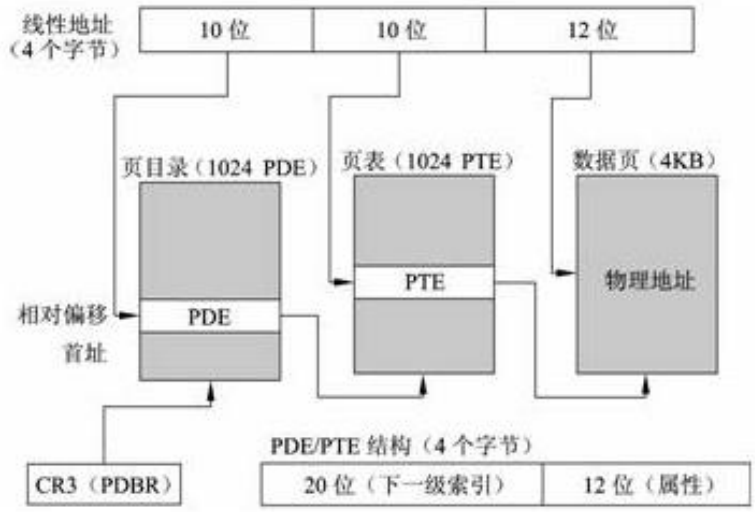

图 5 线性地址结构

将线性地址转换成物理地址的步骤：

(1)、因为页目录表的地址放在 CPU 的 cr3 寄存器中，因此首先从 cr3 中取出进程的页目录表地址（操作系统负责在调度进程的时候，已经把这个地址装入对应寄存器）；

(2)、根据线性地址前十位，在页目录表中，找到对应的索引项，因为引入了二级管理模式，页目录中的项，不是页的地址，而是一个页表的起始地址。

(3)、查找页表，根据线性地址的中间十位，在页表中找到数据页的起始地址；

(4)、将页的起始地址与页内偏移量（即线性地址中最后 12 位）相加，得到最终我们想要的物理地址；

5. 实验参考步骤

（1）编写 实验使用的示例程序:

```
#include <stdio.h>
```

```
int j = 0x123456;//必须改为自己学号的末6位，否则视为未独立完成！！！

int main()
{
    printf("the address of j is 0x%x\n", &j);
    while(j);
    printf("program terminated normally!\n");
    return 0;
}
```

（2）理解 X86 计算机的寻址机制，理解全局描述符表 GDT，局部描述符表 LDT 等数据结构的内容。

（3）使用 sreg 查看查看 GDTR，LDTR，DS 等寄存器内容，了解寄存器的数据格式。

（4）根据寄存器和相关的数据结构，计算变量 j 的线性地址。

（5）使用 creg 查看控制寄存器信息

（6）根据线性地址和页内偏移，基于页式地址转换，计算变量 j 在内存中的物理地址。

（7）根据 j 的物理地址，修改 j 的值，将 j 变为 0，从而使得上述程序能够结束循环。

另参后详细实验步骤

6. 实验报告

独立完成，实验报告要求给出具体的实验原理分析，实验步骤描述中要有相关步骤的截图说明。

详细实验步骤示例

1、点击 bochs.exe 安装 bochs。

2、拷贝 bootimage-0.11-hd、diska.img、hdc-0.11-new.img、mybochsrc-hd.bxrc 至安装目录。

3、在安装目录中找到 bochsdbg.exe 程序，并运行。

4、在弹出的界面中，点击“Load”加载配置文件“mybochsrc-hd.bxrc”。随后，点击“Start”启动 Bochs 虚拟机。

5、虚拟机启动后，出现两个窗口，一个为 Bochs 控制窗口，另一个为 Linux 操作系统运行窗口（主显示窗口）。

5、在控制窗口输入“c”后回车，加载 Linux 操作系统。

6、在 Linux 操作系统中，使用 vi 工具编写 mytest.c 源文件。随后执行“gcc -o mytest mytest.c”命令编译并生成“mytest”可执行文件。

8、在 Linux 操作系统中，运行“./mytest”可执行文件。

9、控制窗口按 Ctrl+c，进入调试状态。

10、在控制窗口中输入 sreg 命令，查看段的具体信息。可以看到 ds 段的段标识符信息是 0x0017（0000 0000 0001 0111）,对应 TI=1，表明段描述符在 LDT 中，右移 3 位之后为 0x02，即表示在局部描述符表 LDT 的偏移量为 2。

```
ss:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=1
        Data segment, base=0x10000000, limit=0x03ffffff,
ds:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=3
        Data segment, base=0x10000000, limit=0x03ffffff,
fs:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=1
        Data segment, base=0x10000000, limit=0x03ffffff,
gs:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=1
        Data segment, base=0x10000000, limit=0x03ffffff,
ldtr:0x0068, dh=0x000082fd, dl=0x92d00068, valid=1
tr:0x0060, dh=0x00008bfd, dl=0x92e80068, valid=1
gdtr:base=0x0000000000005cb8, limit=0x7ff
idtr:base=0x00000000000054b8, limit=0x7ff
```

11、查看 LDTR 寄存器，其中存放了 LDT 在 GDT 的位置。0x0068 对应 TX=0，右移 3 位之后为 0x0D，即在 GTD 中的索引为(0x0D) 13。

12、gdtr 存放了 GDT 的起始地址，用 xp /2w 0x00005cb8+13*8（每个描述符占 8 个字节）查看 GDT 中对应表项，得到的 LDT 段描述符，从而我们可以得到 LDT 的基址为 0x00fd92d0。

```
<bochs:3> xp /2w 0x5cb8+13*8
[bochs]:
0x0000000000005d20 <bogus+       0>:    0x92d00068      0x000082fd
```

13、用 xp /2w 0x00fd92d0+2*8，查看 LDT 中第 2 项段描述符（即 ds 段的描述符信息，应与 ds 寄存器（dl、dh）中的数值完全相同）。

```
<bochs:4> xp /2w 0xfd92d0+16
[bochs]:
0x0000000000fd92e0 <bogus+       0>:    0x00003fff      0x10c0f300
```

14、计算出 ds 段的基地址为 0x10000000（与用 sreg 得到信息一致）。

15、计算线性地址 0x10000000+0x3004= 0x10003004， 将其用 0 补满 32 位(0001 0000 0000 0000 0011 0000 0000 0100)，然后按照 10-10-12 比特的方式划分，为 0x40-0x03-0x04。即外页偏移为 0x40,内页偏移为 0x03，页内偏移为 0x04。

16、使用 creg 查看寄存器 CR3 值为 0，即页目录表的起始地址为 0。

17、使用 xp /w 64*4 查看 PDE 为 0x00fa9027,下一级索引为 0x00fa9000。（或执行 xp /2w 0x40*4）

```
<bochs:5> xp /2w 0x40*4
[bochs]:
0x0000000000000100 <bogus+       0>:    0x00fa9027      0x00000000
```

18、使用 xp /w 0x00fa9000+3*4 查看 0x00fa6067,下一级索引为 0x00fa6000，得到物理地址为 0x00fa6000+4。

```
<bochs:6> xp /2w 0xfa9000+3*4
[bochs]:
0x000000000fa900c <bogus+       0>:    0x00fa6067      0x00000000
```

19、使用 xp /w 0x00fa6000+4,内容为 **0x00123456** 与我们所设的值相同。（必须显示为自己学号的末 6 位，否则视为未独立完成！！！）

```
<bochs:9> xp /w 0xfa6000+4
[bochs]:
0x000000000fa6004 <bogus+       0>:    0x00123456
```

20、使用 setpmem 0x00fa6004 4 0,设置 0x00fa6004 开始的四个字节均为 0，并检查是否成功。

21、成功之后输入 c 继续运行，显示程序正常结束。

命令解释：

ctrl+c　　进入调试状态

c　　继续执行，直到遇到断点

sreg　　查看段寄存器和段描述符寄存器

creg　　查看控制寄存器

xp [/nuf] addr　　显示物理地址的内容

其中 n 指定显示的单元数，默认是 1；

u 指定每个显示单元的大小（b 表示字节、h 表示字（2 字节）、w 表示双字（4 字节）），默认是 w；

f 指定显示格式（x 十六进制、d 有符号十进制、u 无符号十进制、o 八进制、t 二进制、c 字符），默认是 x

setpmem [addr] [size] [val] 设置物理内存某地址的内容。addr 地址、size 字节数、val 值