

电子科技大学计算机科学与工程学  
院

# 实 验 报 告

(实验) 课程名称 计算机操作系统

# 电子科技大学 实 验 报 告

学生姓名：卢晓雅

学 号：2020080904026

指导教师：刘杰彦

实验地点：主楼 A2-412

实验时间：2022 年 12 月 3 日星期六

一、实验室名称：操作系统实验室

二、实验项目名称：内存地址转换实验

三、实验学时：4 学时

四、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

五、实验环境

Linux 内核 (0.11) +Bochs 虚拟机

六、实验内容和原理

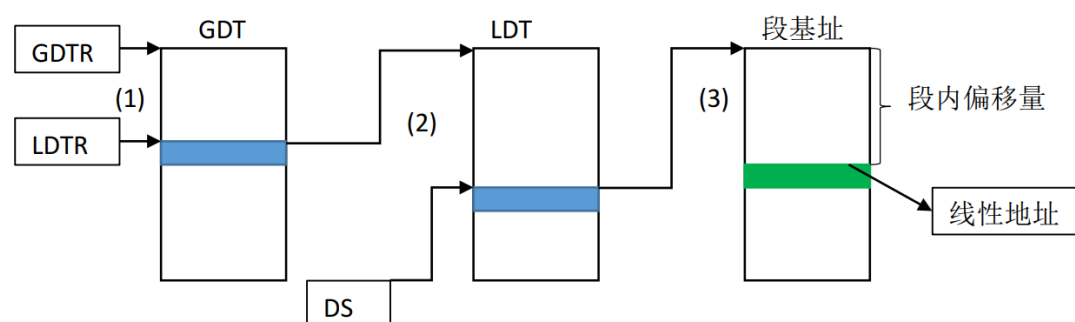
实验内容：根据 X86 计算机的寻址机制查找程序中的某变量的物理地址，并访问相应的内存单元，验证其中存储的内容与我们设置的初始化值是否一致，再将该内存单元中的内容清零，触发结束循环的条

件，使程序成功运行至结束。

实验原理：

## 1. X86 计算机的寻址机制

先将逻辑地址转换为线性地址，转换方式如图一，注意在程序中使用取地址符“&”输出的地址是段内偏移量，图一中的段基址是逻辑地址。



图一

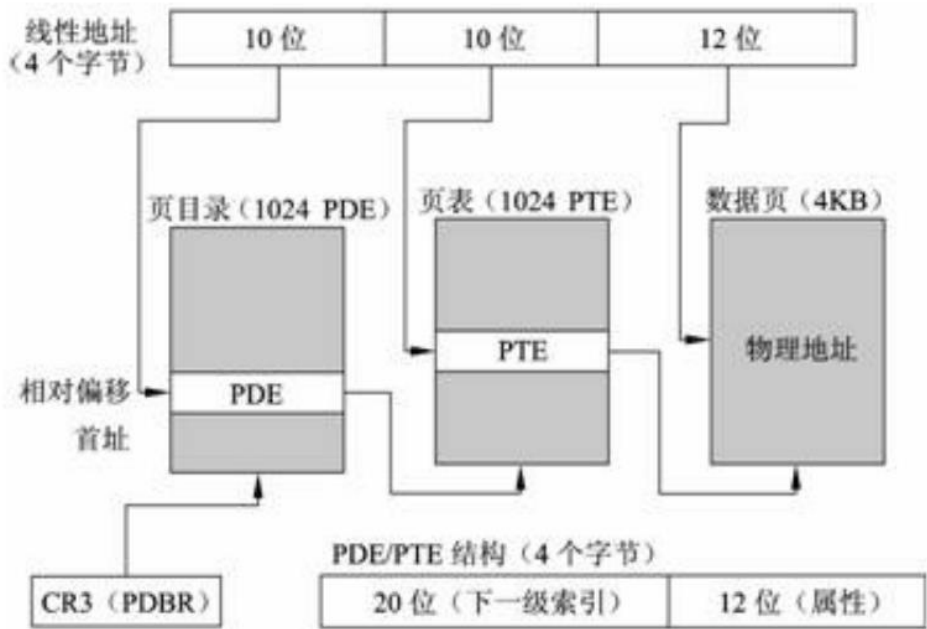
具体步骤如下：

(1) 从 **GDTR** 中获得 **GDT** 的起始地址，从 **LDTR** 中获得 **LDT** 在 **GDT** 中的偏移量，根据偏移量定位到相应的表项，从中获取 **LDT** 的起始地址；

(2) 从 **DS** 中的高 13 位获取 **DS** 段在 **LDT** 中偏移量并定位到相应的表项，获取 **DS** 段的段描述符，根据段描述符的划分，获取 **DS** 段的基地址；

(3) 根据 **DS** 段的基地址 + 段内偏移量，定位到 **DS** 段内的具体位置，获取所需单元的线性地址。

再将线性地址转换为物理地址，X86 系统使用二级页表，线性地址的划分方式如图二。



图二

转换步骤如下：

- (1)从 CR3 寄存器中取出顶级页表（页目录表）的起始地址；
- (2) 根据线性地址前十位，即顶级页表中的相对偏移，定位到对应的页表项，获取二级页表的起始地址。
- (3) 根据线性地址的中 10 位，即二级页表中的相对偏移，定位到对应的页表项，获取数据页的起始地址；
- (4) 将线性地址中最后 12 位，即页内偏移量，与页的起始地址相加，即能得到最终的物理地址；

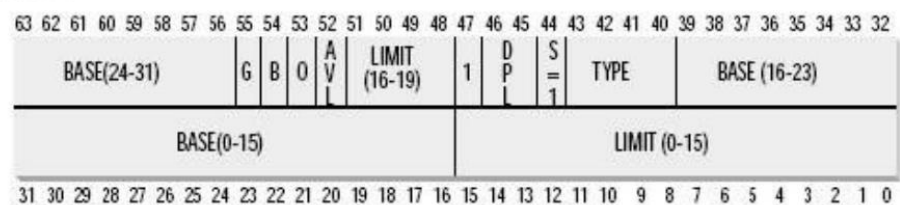
2. 寻址机制的硬件支持：

a) 段表和页表的结构

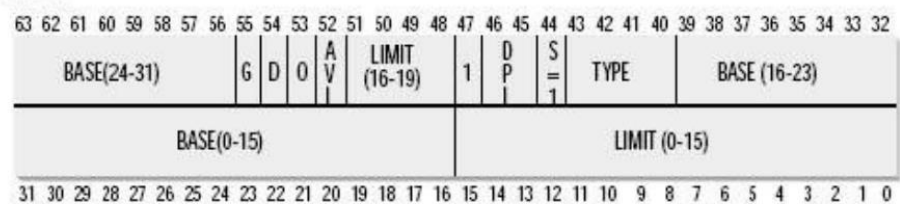
X86 系统有两种段表，全局段描述符表（GDT）：整个系统一个，GDT 表中存放了共享段以及 LDT 的描述符；局部段描述符表（LDT）：每个进程一个，存放进程内部的各个段的描述符。使用 TI 字段指示段描述符是存放在 LDT（TI=1）还是 GDT（TI=0）中。

段描述符划分如图三：

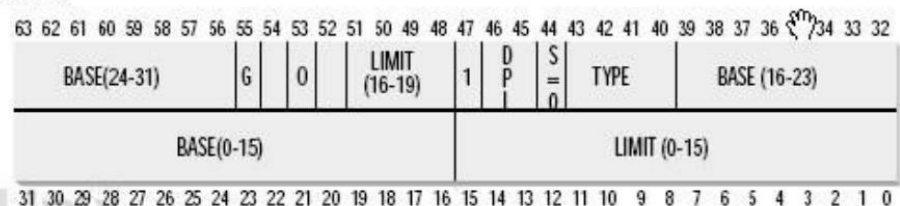
数据段描述符



代码段描述符

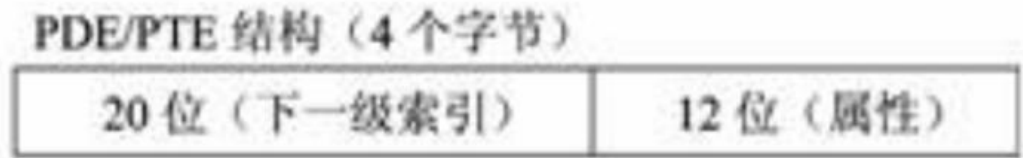


系统段描述符



图三

因为页的大小为 4KB，所以页内偏移量为 12 位，物理地址为 32 位时，页框号对应页表项中的高 20 位。页目录项和页表项的划分如图四：



图四

b) 寻址机制相关的寄存器

**DS（data segment）寄存器：**存储当前的程序数据段对应的段选择符。

**GDTR：**存放 GDT 在内存中的起始地址和大小。

**LDTR：**TI=1 时，LDTR 存放 LDT 在 GDT 中的索引。

**CR3：**存放根页表的起始物理地址。

### 3. Linux 系统操作命令

**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

**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

## 七、实验步骤及结果分析：

1. 启动虚拟机。
2. 编写mytest.c程序（如图五），编译生成可执行文件，运行该文件，输出变量j逻辑地址的段内偏移量部分，如图六。

```
#include <stdio.h>
int j=0xB0904026;

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

图五

```
[/usr/rootl# mytest
the address of j is 0x3004
```

图六

3. 在控制窗口中按Ctrl+c进入调试状态。
4. 首先查看段标识符,定位到段表项:在控制窗口中输入 sreg命令,查看数据段寄存器和段描述符寄存器中的具体信息,如图七,ds中存储的段标识符信息是 0x0017 (0000 0000 0001 0111),对应的TI (第14位)为1,由此可知相应的段描述符存储在局部段描述符表LDT中,这也与变量j是程序的变量相符,段索引号 (高13位)为2,即表示在局部描述符表LDT的偏移量为2。
4. 然后访问该段表项,得到ds段的逻辑基地址:

1) 查找LDT段表的基地址: LDT作为公共的段,其段表项在GDT中。查看LDTR寄存器,其中存放了LDT在GDT中的偏移量 (LDT也是作为一个全局段)。0x0068 (0000 0000 0110 1000) 对应TI=0,段索引号为13。GDTR存放了 GDT 的起始地址,用 `xp /2w 0x00005cb8+13*8` (`xp /w`命令用于显示物理地址对应的内存中存储的内容, LDT对应段表项的物理地址=GDT的起始地址+偏移量\*每个描述符所占的字节数)

查看GDT中对应表项，得到的LDT段描述符，如图八，根据图三对应的段描述符划分，我们可以得到LDT的基址为0x00fd92d0。

```
<bochs:2> sreg
es:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=1
    Data segment, base=0x10000000, limit=0x03ffffff, Read/Write, Accessed
cs:0x000f, dh=0x10c0fb00, dl=0x00000002, valid=1
    Code segment, base=0x10000000, limit=0x00002fff, Execute/Read, Accessed,
    32-bit
ss:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=1
    Data segment, base=0x10000000, limit=0x03ffffff, Read/Write, Accessed
ds:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=3
    Data segment, base=0x10000000, limit=0x03ffffff, Read/Write, Accessed
fs:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=1
    Data segment, base=0x10000000, limit=0x03ffffff, Read/Write, Accessed
gs:0x0017, dh=0x10c0f300, dl=0x00003fff, valid=1
    Data segment, base=0x10000000, limit=0x03ffffff, Read/Write, Accessed
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
<bochs:3>
```

图七

```
<bochs:3> xp /2w 0x5cb8+13*8
[bochs]:
0x0000000000005d20 <bogus+    0>:    0x92d00068    0x000082fd
<bochs:4>
```

图八

2) 定位到LDT中相应的段表项并查看，偏移量为2，执行xp /2w 0x00fd92d0+2\*8，如图九，与 ds 寄存器（dl、dh）中的数值完全相同，得到ds段的基地址为0x10000000，与图三中的base一致。

```
<bochs:4> xp /2w 0x00fd92d0+2*8
[bochs]:
0x0000000000fd92e0 <bogus+    0>:    0x00003fff    0x10c0f300
<bochs:5>
```

图九

5. 计算线性地址，加上输出的段内偏移量。

$0x10000000 + 0x3004 = 0x10003004$ 。将线性地址被划分为三部份，得到第一级页表内的索引(高10位)为0x40，第二级页表内的索引(中间10位)为0x03，页内偏移(低12位，因为页面大小为4K)为 0x004。



6. 接下来将线性地址转换成物理地址，使用creg查看寄存器CR3值为0，即页目录表（第一级页表）的起始地址为0。

```
<bochs:5> creg
CR0=0x8000001b: PG cd nw ac wp ne ET TS em MP PE
CR2=page fault laddr=0x0000000010002fa8
CR3=0x0000000000000000
PCD=page-level cache disable=0
PWT=page-level write-through=0
CR4=0x00000000: smep osxsave pcid fsgsbase smx vmx osxmmexcpt osfxsr pce pge mce
pae pse de tsd pvi vme
EFER=0x00000000: ffxsr nxe lma lme sce
```

图十

7. 使用 `xp /w 0+0x40*4`（页表项大小为4B）查看第一级页表项中记录的内容为0x00fa9027，如图十一，根据PDE的划分，得到下一级页表的起始地址（高20位）为0x00fa9000。

```
<bochs:6> xp /w 64*4
[bochs I:
0x00000000000000100 < bogus+ 0>: 0x00fa9027
```

图十一

8. 使用 `xp /w 0x00fa9000+0x03*4` 查看第二级页表项中记录的内容为0x00fa7067，根据PDE的划分，得到下一级页表的起始地址为0x00fa7000。与0x004拼接出物理地址为0x00fa7004。

```
<bochs:7> xp /w 0x00fa9000+3*4
[bochs I:
0x0000000000fa900c < bogus+ 0>: 0x00fa7067
```

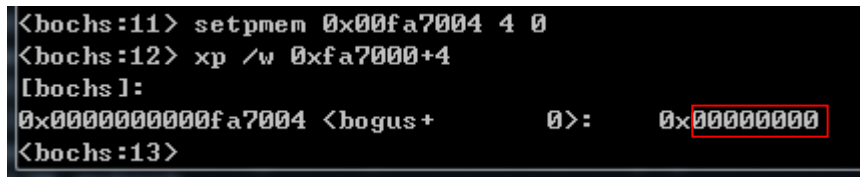
图十二

9. 使用 `xp /w 0x00fa7004`，查看转换出的物理地址对应的内存单元中存储内容为0x80904026，与我们所设的值相同。

```
<bochs:9> xp /w 0xfa7000+4
[bochs I:
0x0000000000fa7004 < bogus+ 0>: 0x80904026
```

图十三

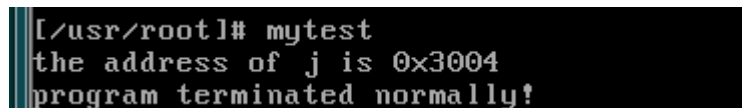
10. 使用 `setpmem 0x00fa7004 4 0`, 设置 `0x00fa7004` 开始的四个字节均为0, 再次查看内容, 如图十所示, 已重置成功。



A screenshot of the Bochs debugger interface. The command line shows `<bochs:11> setpmem 0x00fa7004 4 0`. The next line shows `<bochs:12> xp /w 0xfa7000+4`. The output line shows `[bochs]: 0x0000000000fa7004 <bogus+ 0>: 0x00000000`, where the value `0x00000000` is highlighted with a red box. The prompt `<bochs:13>` is visible at the bottom.

图十四

11. 输入c继续运行, 显示程序正常结束。



A screenshot of a terminal window. The prompt is `[/usr/root]#`. The command `mytest` has been executed. The output shows `the address of j is 0x3004` followed by `program terminated normally!`.

图十五

## 八、实验结论:

根据地址转换的原理能成功将逻辑地址转换成线性地址, 再转换成物理地址, 并查看对应的内存单元中存储的内容。

## 九、总结及心得体会:

X86 系统的设计与我们课堂所学的段页式有所不同, 段页式是分段后在段内分页, 逻辑地址可直接划分成段号、页号和页内偏移量, 而 X86 使用了一个线性地址作为过渡, 把分段和分页变成了相对独立的两个层次。细节方面, X86 采用了两种段表, GDT 和 LDT 分别存放共享段和某程序的私有段对应的段表项, 这样能方便实现模块的动态链接和共享, 更灵活, 另外某个程序的 LDT 也作为一段, 其段表项存放在 GDT 中。

在程序中使用取地址符 “&” 只能得到段内偏移量, 段索引号还

要通过查询段描述符才能得知。

#### **十、对本实验过程及方法、手段的改进建议：**

实验中要重点理解段基址和线性地址的意义，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。段基址+段内偏移量就得到了线性地址，线性地址表示要访问的数据在整个程序逻辑(虚拟)地址空间中的位置。

使用 `sreg` 命令之后其实就自动计算出段基址了，即图七中的 `base`，实际上可以不使用 `ds` 寄存器中的段标识符来推导段基址。

**报告评分：**

**指导教师签字：**