电 子 科 技 大 学

实 验 报 告

学生姓名： 陈凯悦	学 号：2016060601020

一、实验课程名称：计算机操作系统

二、 实验项目名称：虚拟内存综合实验

三、实验目的：

通过实验，掌握段页式内存管理机制，理解地址转换的过程

四、实验内容：

实验首先运行了一个设置了全局变量的循环程序，通过手工查看系统内存，并修改特定物理内存的值，实现控制程序运行使之输出指定内容的目的。

五、实验环境

具有 Linux 内核的 Bochs 虚拟机+Windows10 系统

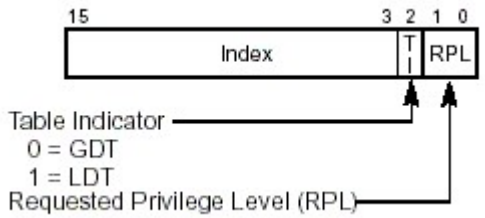
六、实验原理：

计算机中有三种地址，从计算机底层到用户层依次为物理地址，线性地址（虚拟地址），和逻辑地址；

物理地址是内存中实实在在的地址；逻辑地址是由编译器给出的在程序中放的位置，即在机器指令中，用来指定一个操作数或者一条指令的地址，例如在 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构成的；线性地址（或虚拟地址）是程序访问存储器所使用的逻辑地址，是逻辑地址到物理地址变换之间的中间层。

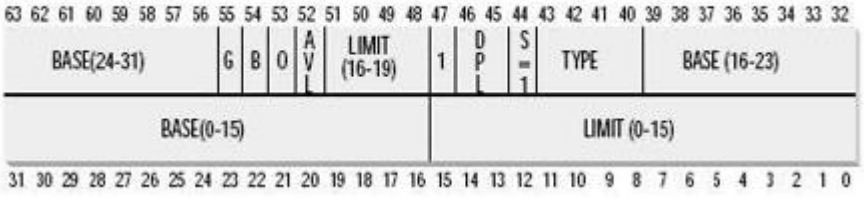
（一）逻辑地址到线性地址的转换

一个逻辑地址由段标识符和段内偏移量组成。段标识符是由一个 16 位长的字段组成，也叫段选择符，其结构如下：



其中，当 T1=0 时，直接通过索引号在全局段描述符表（GDT）中寻找一个具体的段描述符。 这个段描述符就描述了一个段，段描述符的结构如下：

数据段描述符



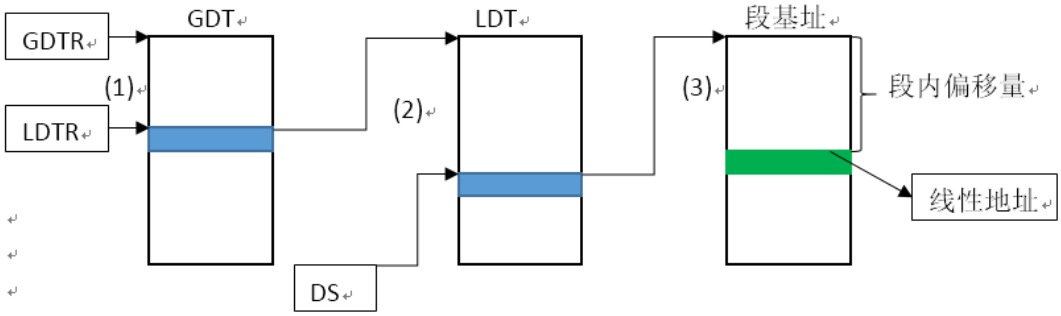
由此，就得到了基址 base，再与逻辑地址中的段内偏移量相加，就得到了要转换的线性地址了。（具体的转换过程见实验指导书）

当 T1=1 时，先在局部段描述符表（LDT）中寻址，情况比较复杂，具体的转换过程为：

（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，从中就可以获得 LDT 的起始地址。

（2） 从逻辑地址中的高 13 为获取逻辑地址段在 LDT 中索引位置，查找 LDT，获取逻辑地址段的段描述符，从而获得逻辑地址段的基址

（3） 根据 DS 段的基地址，再加上逻辑地址中的段内偏移量，就得到了所需单元的线性地址。如下如图所示：



（二）线性地址到物理地址的转换

CPU 的页式内存管理单元，负责把一个线性地址，最终翻译为一个物理地址，线性地址是一个 32 为的地址，其结构如下所示：

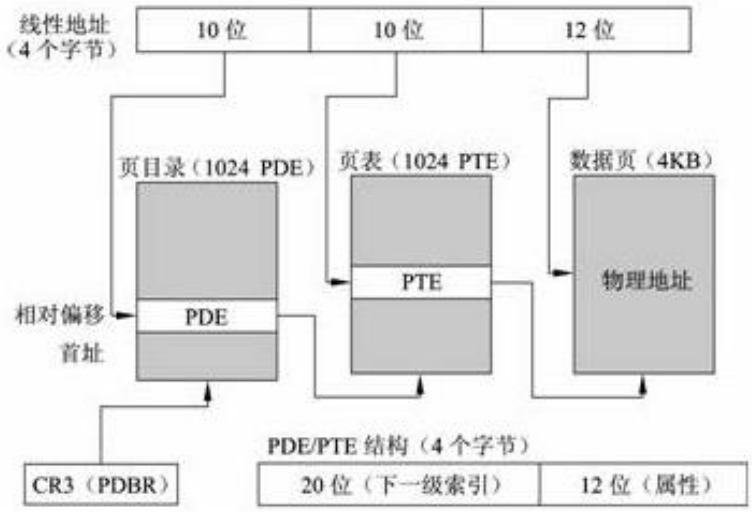
显然，有线性地址到物理地址的转换为：

（1） 从 cr3 中取出进程的页目录地址（操作系统负责在调度进程的时候，把这个地址装入对应寄存器）；

（2） 根据线性地址前十位，在数组中，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，不再是页的地址，而是一个页表的地址。（又引入了一个数组），页的地址被放到页表中去了。

（3） 根据线性地址的中间十位，在页表（也是数组）中找到页的起始地址；

（4） 将页的起始地址与线性地址中最后 12 位相加，得到最终我们想要的物理地址；



七、实验步骤

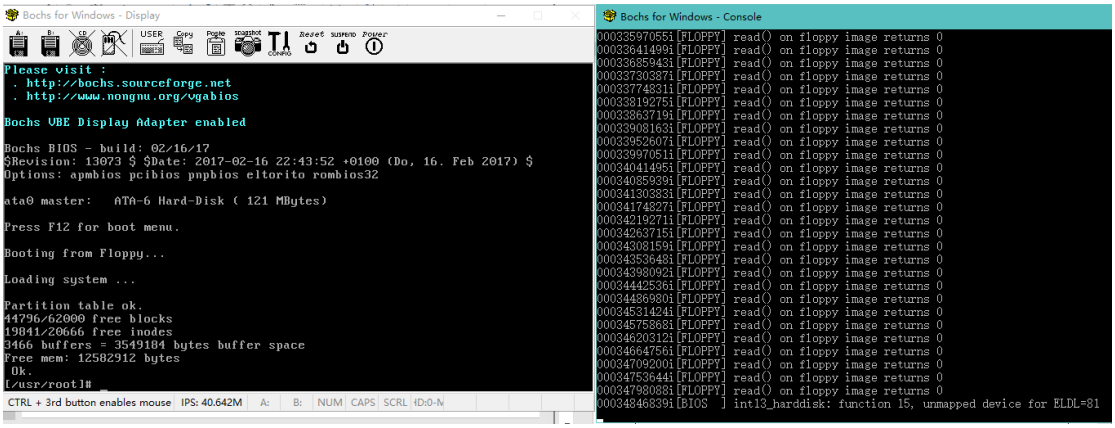
1. 环境的安装和配置

（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）双击 bochsdbg.exe 程序并运行，即可使用具有 debug 功能的 bochs 工具，在弹出的 Bochs Start Menu 中，点击 load 加载文件 mybochsrc-hd.bxrc，然后即可点击 start 启动 bochs 虚拟机。

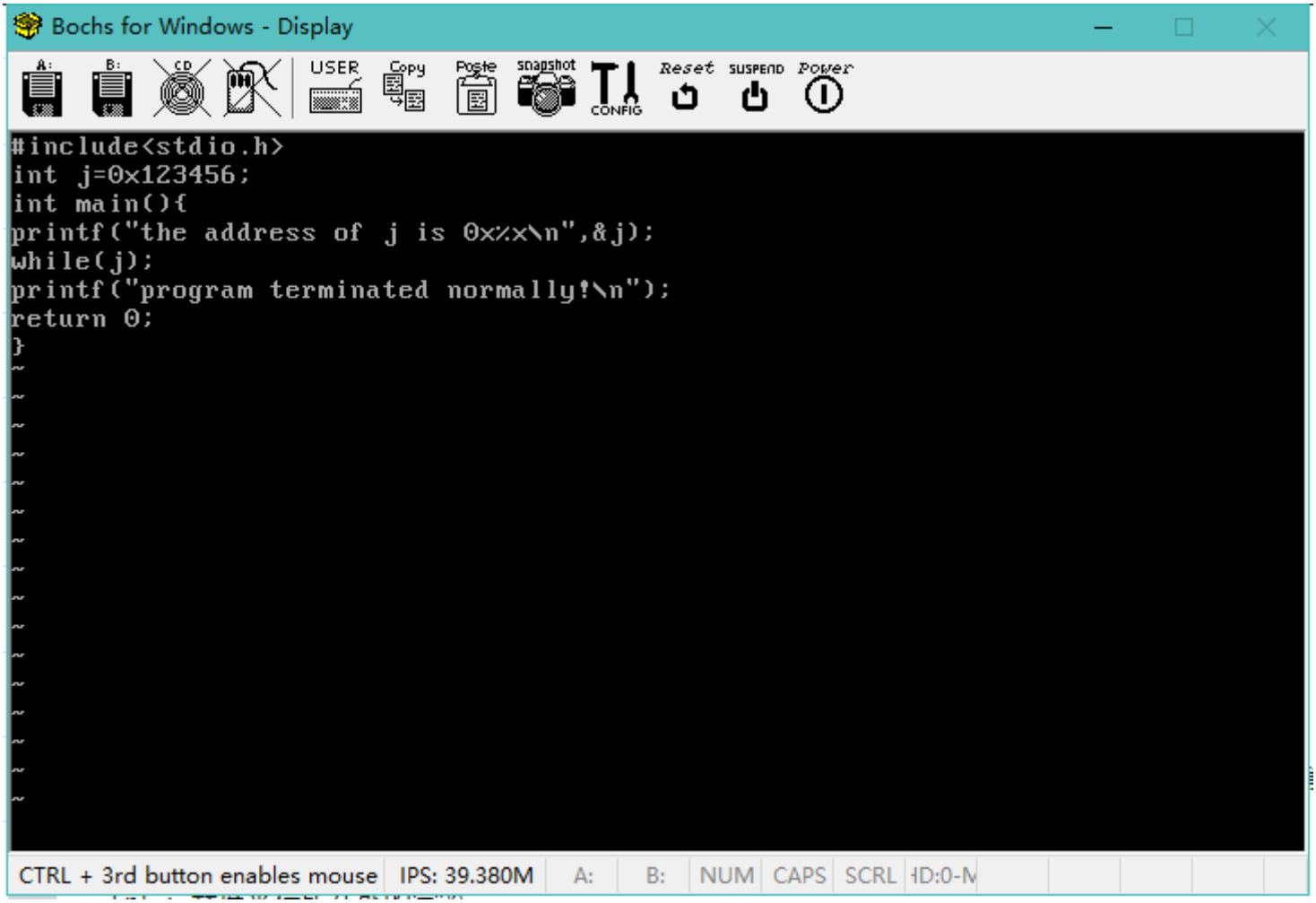
（4）弹出的两个界面分别为：bochs for Windows-Display 和 Bochs for Windows-Console，在 console 界面中输入 c 并回车，Linux 操作系统就正式加载好了。



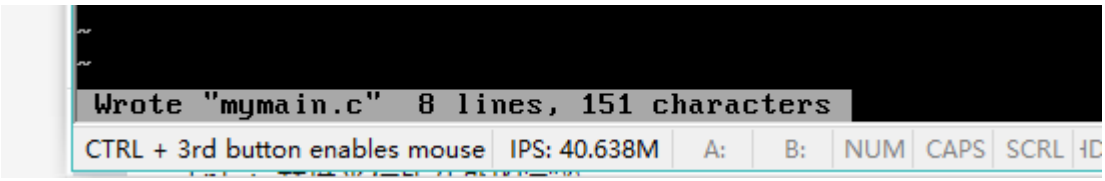
2.文件的编辑和编译

（1）在 Display 界面中输入 vi mymain.c，然后进入编辑界面，输入 i 进入编辑模式，输入要编写的程

序，如图：

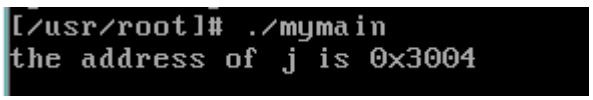


（2）按 esc 键退出编辑模式，进入命令行模式，在命令行模式下，按一下「:」冒号键进入「Last line mode」底行模式，输入 wq ，将 mymain.c 程序存盘并退出 vi。



（3）执行 gcc –o mymain mymian.c 命令，可以得到名字为 mymain 的二进制可执行文件，显示文件列表可以得到 Linux 系统中有两个文件，分别为 mymain.c 和可执行文件 mymain

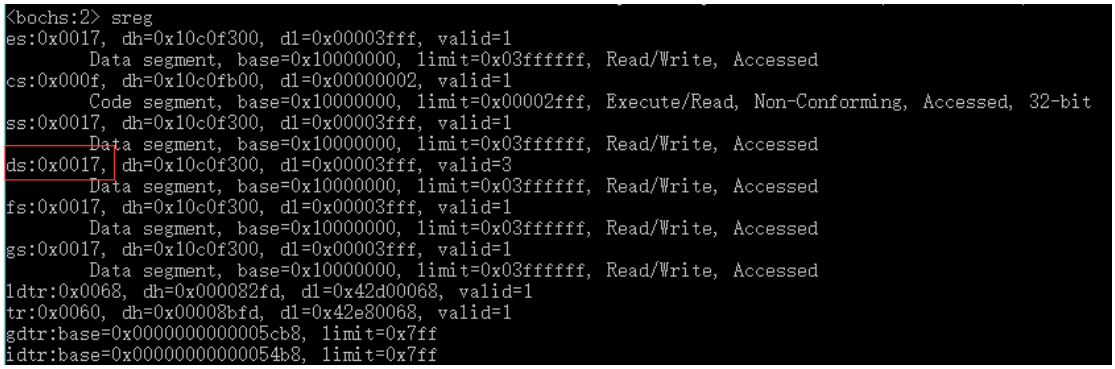
（4）在 Linux 操作系统中，运行 mymain 可执行文件，可以得到如下的结果：



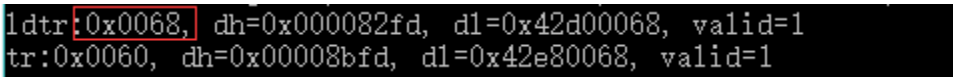
可以看出，程序进入了 while 死循环，接下来实验的目的就是使程序跳出死循环，输出“program terminated normally!”。

3.寻址并对目标值（j）进行修改（全程都在 Display 中执行）

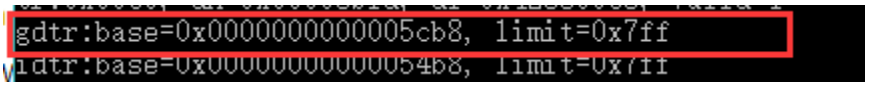
（1）输入 sreg 命令，查看段的具体信息，根据 ds 段的信息，可以确定 ds 段的前 13 位为索引号，倒数第三位为 T1。DS 段为 0x0017=0000 0000 0001 0111B，即索引号为 02H，T1=1，且段描述符在 LDT 表中的第三项。



（2）由于 T1=1，可知段描述符放在了 LDT 中，因此查看 LDTR 寄存器，其中存放了 LDT 在 GDT 中的位置，索引号为 LDTR 的前 13 位。LDTR 为 0x0068=0000 0000 0110 1000B，可以知道索引号为 0DH，即 LDT 起始位置存放在 GDT 的第 14 项。



（3）查看 GDTR 寄存器，其中存放了 GDT 在内存中的起始位置，由图可知为：5CB8H

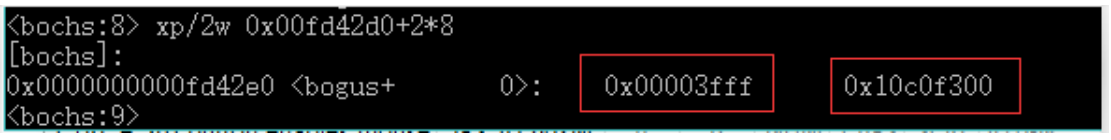


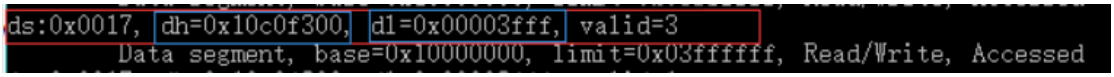
（4）由于每个段描述符由 8 个字节组成，因此，LDT 的首地址为：

GDT 在内存中的起始地址+8*LDT 在 GDT 中的偏移=5CB8H+0DH*8=5D20H

（5）执行 xp/2w0x5d20，可以得到 GDT 中对应的表项，从而得到 LDT 的段描述符，根据 LDT 的结构，可以知道 LDT 的基址为 0x00fd42d0。

（6）因为段描述符在 LDT 表中的偏移量为 2，每个段描述符占 8 个字节，所以执行 xp/2w 0x00fd42d0+2*8，可以查看到 LDT 中第三项段描述符，即 DS 段的描述符信息，可以看出它与 ds 寄存器中的数值相同。





根据 LDT 的数据结构，可以计算出 DS 段的基址为 0x10000000

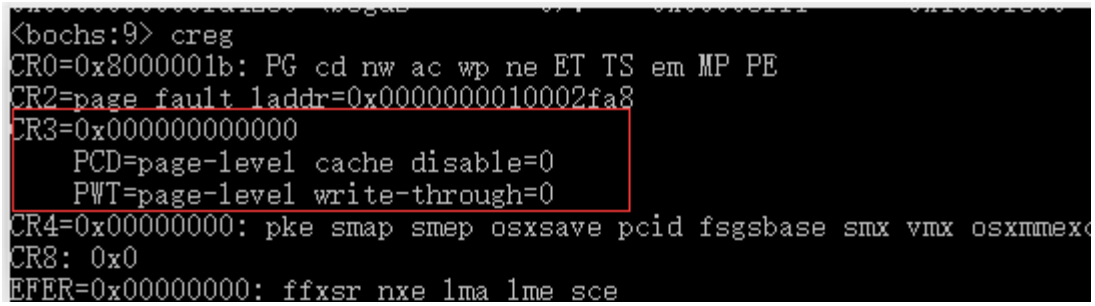
（7）程序中 j 的偏移地址为 0x3004，则可根据 DS 段的基址计算得到 j 的线性地址：

0x1000 0000+0x3004=0x1000 30040=0001 0000 0000 0000 0011 0000 0000 0100B

按照原理中线性地址的 10-10-12 结构进行划分，可以知道页目录索引为 1000 000B=64，页表项索引

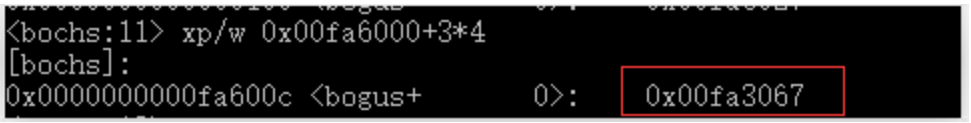
为 11B=3，页内偏移为 100B=4。

（8）使用 creg 查看寄存器 CR3 的值，为 0，则页目录表的起始地址为 0.

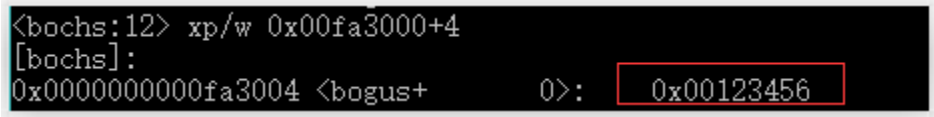


（9）由于页目录索引为 64，页目录表起始地址为 0，所以使用 xp/w 64*4 即可查看 PDE，为：0x00fa6027=0000 0000 1111 1010 0110 0000 0010 0111B，选其前 20 位并左移 12 位，作为下一级索引为：0x00fa6000

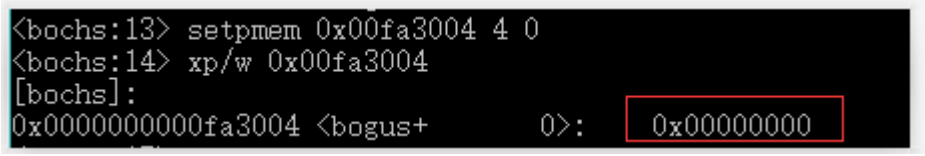
（10）页目录项索引为 3，执行 xp/w 0x00fa6000+3*4，即可查看 PTE 的值为 0x00fa3067，同样地，选其前 20 位并左移 12 位即可得到下一级索引，为 0xfa3000，由于页内偏移为 4，所以物理地址即为：0x00fa3000+4。



（11）使用 xp/w 0xfa3000+4，得到的内容为 0x00123456，与源程序中 j 的初值一样。

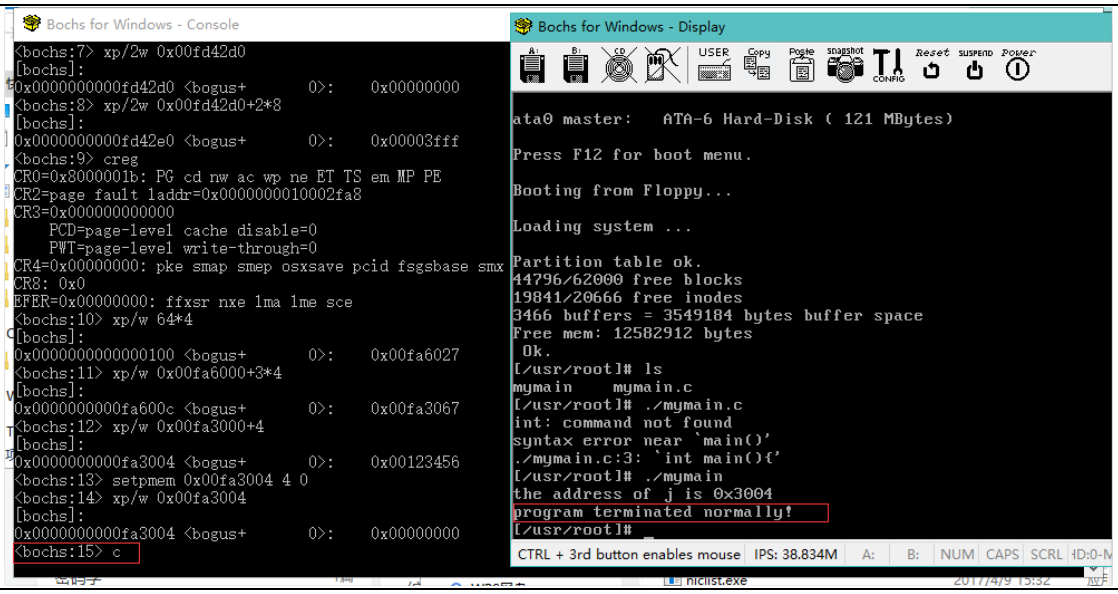


（12）修改目标位置的值，通过命令 setpmem 0x00fa3004 4 0，即可将从 0x00fa3004 开始的四个字节都设置为 0，然后通过 xp/w 0xfa3000+4 命令检查是否修改成功，可以确定，j 的值已经修改成功。



4.修改内存后 mymain 的运行结果

回到 console 中，输入 c 继续运行原程序，可以看到在 Display 中显示程序显示“program terminated normally”，说明程序已跳出 while 循环，j 的值已经被正常修改，程序正常结束，实验成功！



八、实验结果：

根据上图，可以看出通过逻辑地址到线性地址，线性地址到物理地址的寻址，页式地址的转换过程，结合计算机内不同寄存器的位置和作用，成功地根据 j 的逻辑地址找到其物理地址并修改成功，实验中 j 的值被正确修改为 0，程序也正常退出了，本实验成功。

九、总结及心得体会：

（1）万事开头难，在本实验刚开始的时候，由于对 bochs 这一虚拟机不熟悉，导致在配置环境中走了很多弯路，最后通过查阅各种相关资料，详细了解了 bochs 的机制，终于配置好了 bochs 的环境，可以成功启动 bochs 了，虽然过程比较坎坷，但是由此更加深了我对 bochs 虚拟机和其内部的 Linux 内核的认识和理解，今后再使用类似的虚拟机会更加得心应手。

（2）根据已知的逻辑地址，解读其地址结构，再以此寻找段描述符，段内偏移，根据操作系统内存管理的层次结构，逐层寻址，找到真实的物理地址并修改值，在此过程中进一步理解了 CPU 中不同地址的结构及其相互关系和页式内存的原理，通过实验来很好地巩固了理论知识。

十、对本实验过程及方法、手段的改进建议：

可以使用一个更加综合的原程序，例如可以和资源管理结合，利用修改内存达到程序的目的。

报告评分：

指导教师签字：