**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：郭宇航 学号：2016100104014 指导教师：薛瑞尼**

**实验地点：主楼A2-412 实验时间：2019.5.25**

1. **实验室名称：计算机实验室**
2. **实验项目名称：虚拟内存综合实验**
3. **实验学时：4**
4. **实验原理：**

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。

段描述符

索引号

GDT或LDT

图2 GDT或LDT示例

每一个段描述符由8个字节组成，如图3：

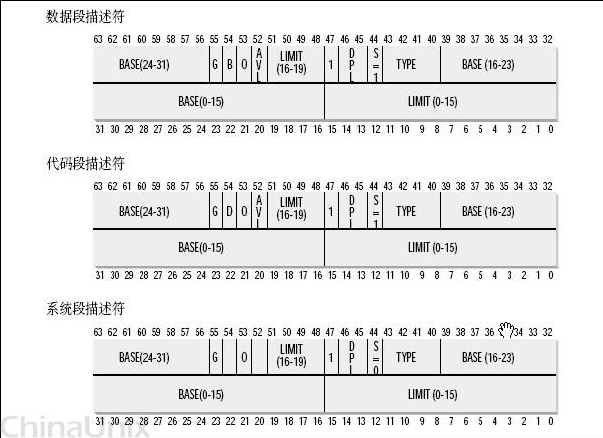


图3 段描述符

Base字段：它描述了一个段的开始位置：段基址。

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。段基地址＋段内偏移量，就得到线性地址。

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。

GDT

LDT

GDTR

LDTR

DS

段基址

段内偏移量

线性地址

(1)

(2)

(3)

图4 逻辑地址到线性地址的转换

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。

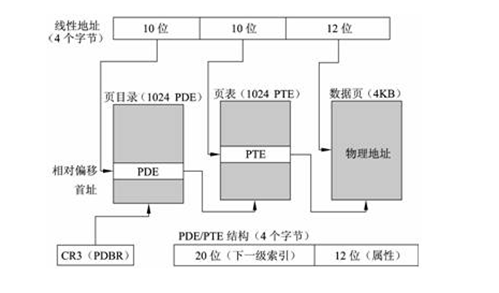


图5 线性地址结构

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。

1. **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。

1. **实验内容：**

**通过手工查看系统内存，并修改特定物理内存的值，实现控制程序运行的目的。**

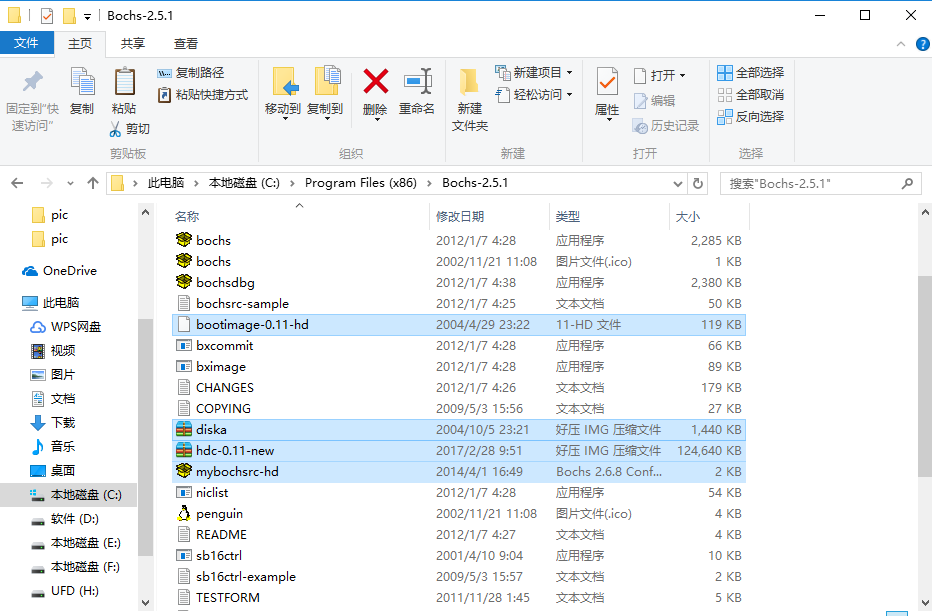
1. **实验器材（设备、元器件）：**

**Linux内核（0.11版）+ Boches虚拟机**

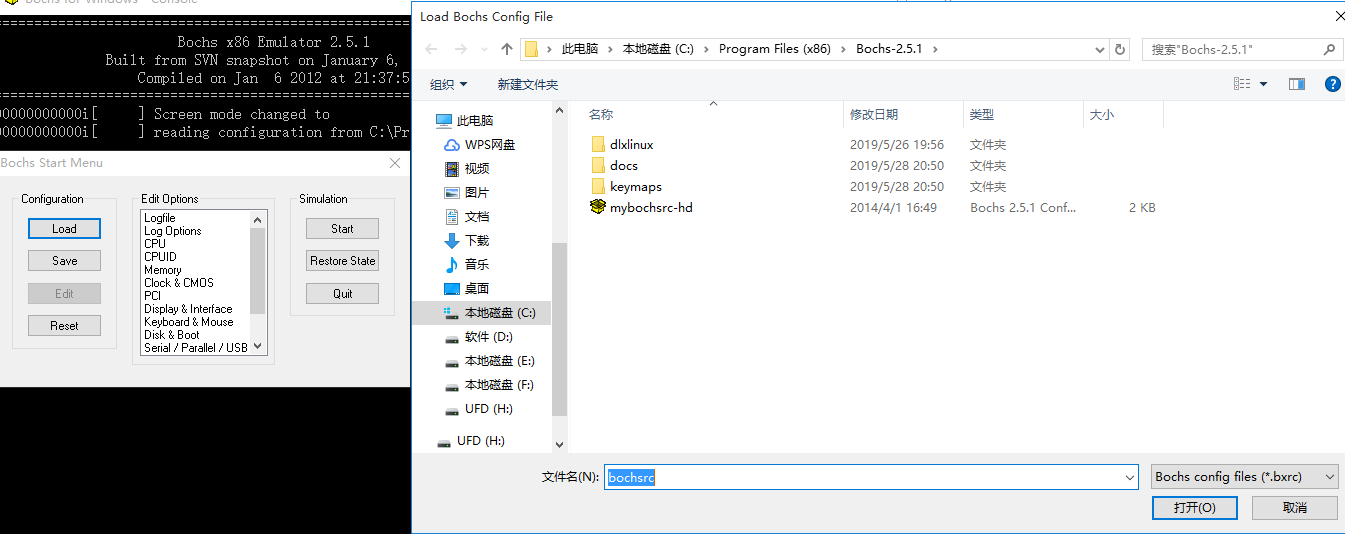
1. **实验步骤：**

**搭建实验环境：**

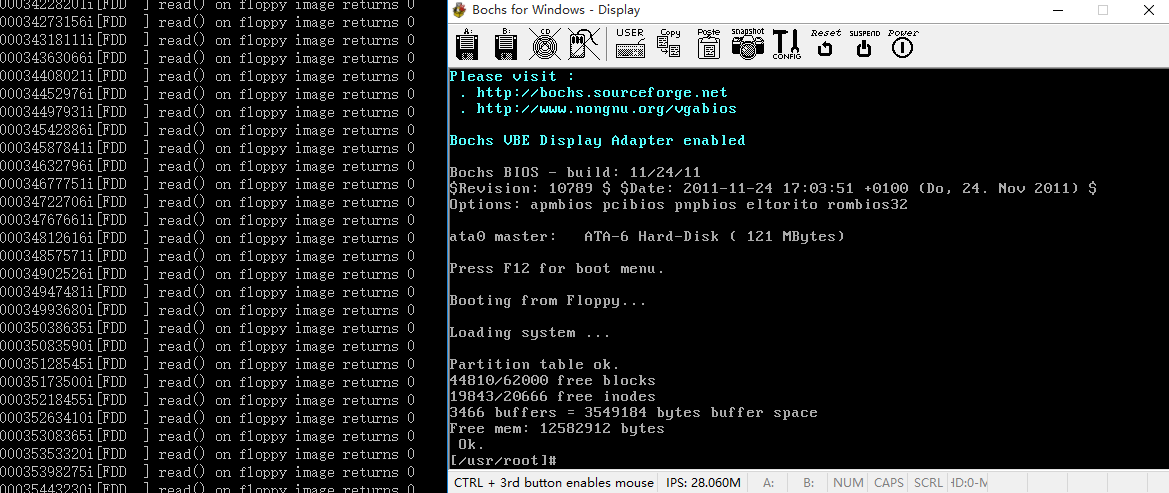
这里使用的是boches2.5.1的版本，安装过程如下：首先下载安装包成功安装boches2.5.1，然后将linux0.11的内核文件拷贝入安装boches2.5.1的文件夹下：



拷贝之后打开bochsdbg可执行文件，加载mybochsrc-hd文件：

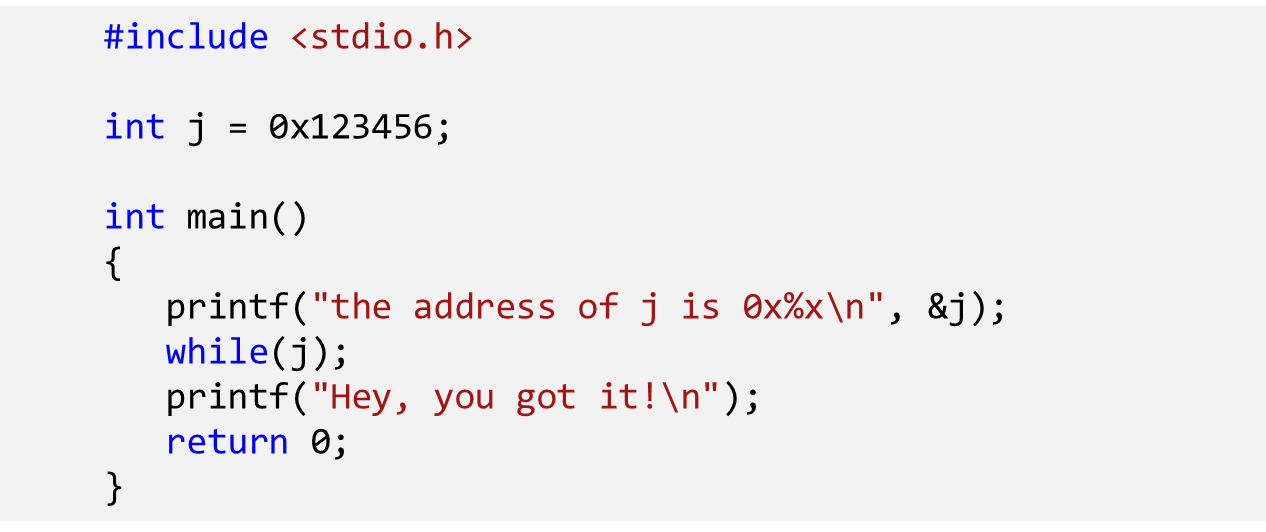


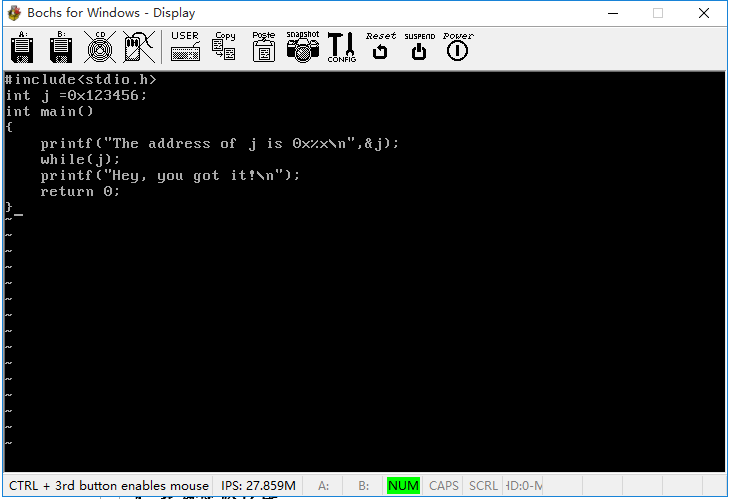
加载成功后再控制窗口输入c启动boches下虚拟的Linux运行窗口：



由此boches2.5.1+Linux0.11的内核实验环境已经配置完成。

2.实验主体部分：

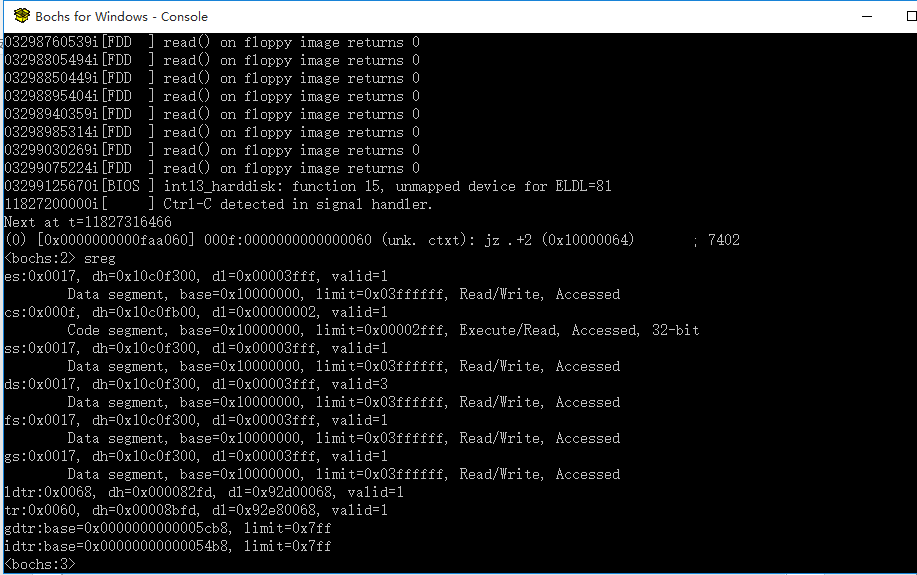
首先启动vi编辑，新建一个名为experiment2.c的文件，输入以下C语言代码：  




输入成功后esc保存并退出:wq

在控制窗口按住ctrl+c启动断点调试模式.

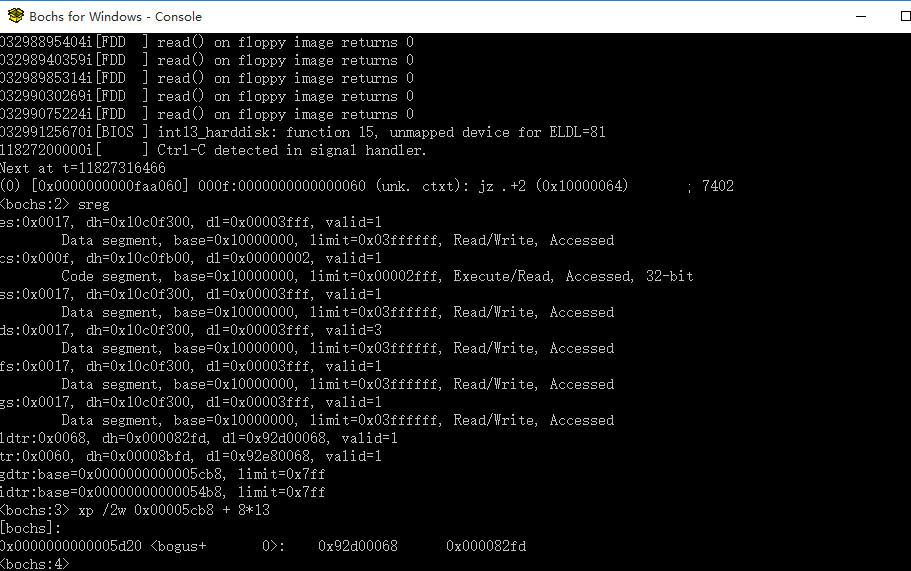
由于逻辑地址是由一个段标识符+offset组成，而段标识符的存储位置在由段标识符中的T1字段决定，首先我们需要寻找段标识符的信息，因此在控制窗口中输入sreg查看寄存器的信息：



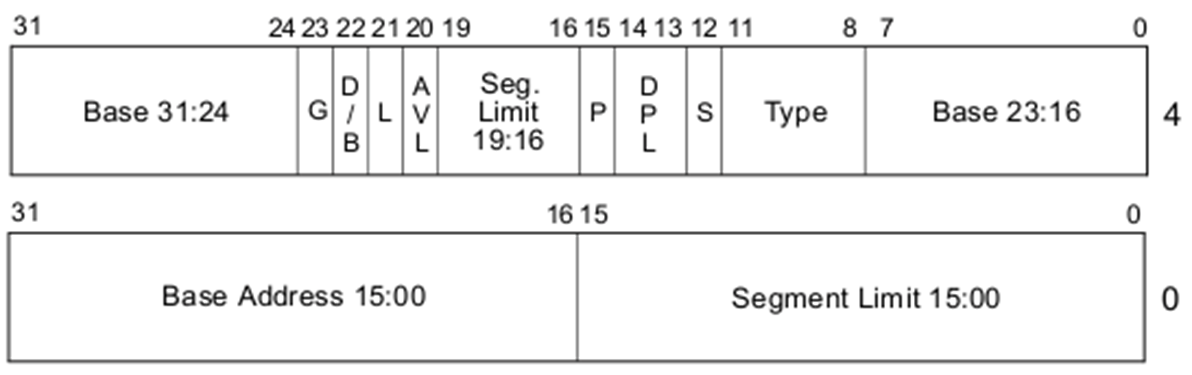
我们实验数据存储在ds数据寄存器中，而我们看到ds寄存器的段信息是：0x0017，换算为二进制后我们发现对应的T1字段为1，索引号为2，因此我们知道此段标识符存储在LDT段表中的第三项。

由于我们知道LDT是存储在GDT中的，需要通过LDTR进行索引，因此再查看LDTR寄存器的段信息：0x0068，换算后可以发现LDT存储在GDT的索引为13，表示为GDT中的第14项。

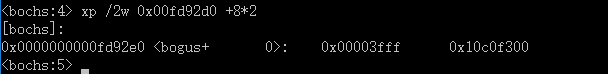
由于GDTR的基地址为：0x5cb8，而GDT中存储的段的属性是一个8字节的数据结构，我们使用xp /2w指令查看LDT在GDT中的索引信息：xp /2w 0x5cb8+13\*8：



得到LDT的段描述符的信息：0x92d00068和0x000082fd，将其第16-31,0-7,24-31位进行拼接得到LDT的基地址为：0x00fd92d0



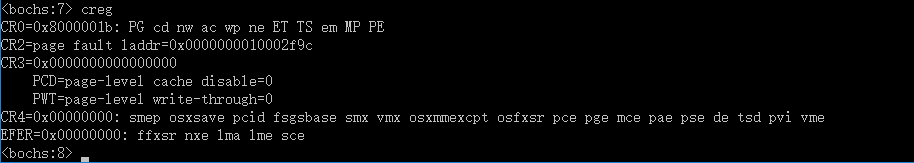
根据ds段标识符信息，ds段存储在LDT中的第3项，使用xp /2w 0x00fd92d0+2\*8查看ds寄存器的段描述符：



基于ds的寄存器的段描述符可以得到其基地址：0x10000000

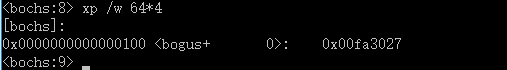
进而很容易计算出j的线性地址为：0x10000000+0x3004，换算为二进制可以得到其页目录号为64，页表号为3，页内偏移为4

使用creg查看寄存器CR3的信息：



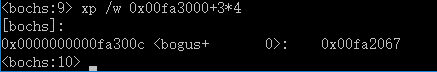
不难发现寄存器CR3的值为0，表明页目录表的起始地址为0

计算一级页表：xp /w 64\*4:



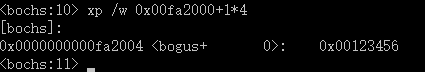
得到地址为：0x00fa3027，则可以得到下一级页表索引为：0x00fa3000

计算二级页表：xp /w 0x00fa3000+3\*4:



得到地址0x00fa2067，则可以得到下一级页表的索引为：0x00fa2000

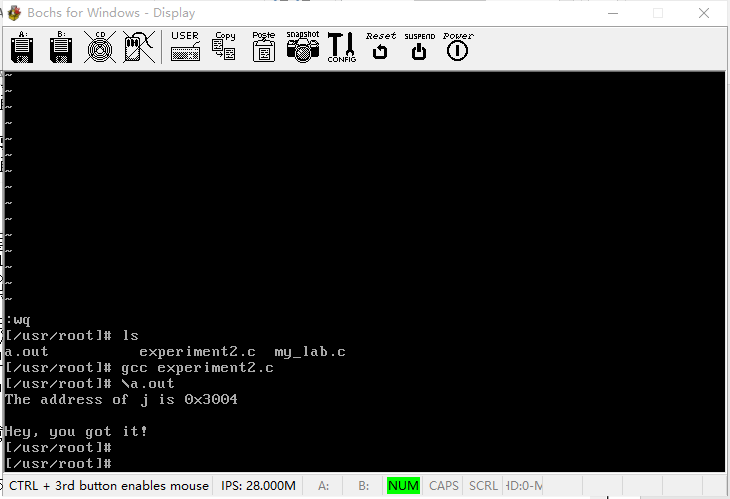
计算三级页表：xp /w 0x00fa2000+4



得到物理地址为：0x00123456，表示我们寻址成功，下面直接将此地址的值修改为0，修改指令为：setpmem 0x00fa2004 4 0



修改之后在控制窗口键入c重新运行experiment2.c代码，得到如下结果：



系统提示“Hey, you got it!”成功跳出死循环，表明j的值修改成功，即物理寻址成功。

1. **总结及心得体会：**

通过本次实验，我对操作系统内存中的地址转换原理和机制有了更加深刻的认识，学会了计算机的寻址过程、地址转换过程以及各类寄存器的使用方法，同时拓展了linux虚拟机下的相关操作知识。

1. **对本实验过程及方法、手段的改进建议：**

**暂无。**

**报告评分：**

**指导教师签字：**