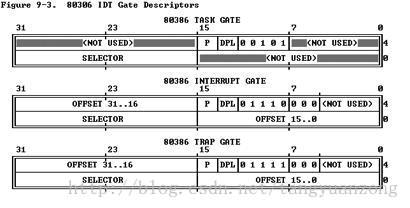
练习6--完善中断初始化和处理

1.  中断向量表中一个表项占多少个字节？其中哪几位代表中断处理代码的入口？

答：系统将所有的中断事件统一进行编号（0～255），这个编号称为中断向量。中断向量表的一个表项占8个字节，其结构如下：



　　0～15位：偏移地址的0～15位;

　　16～31位：段选择子;

　　32～47位：属性信息（包括DPL等）;

　　48～63位：偏移地址的16～31位。

其中第16~32位是段选择子，用于索引全局描述符表GDT来获取中断处理代码对应的 段地址，再加上第0~15、48~63位构成的偏移地址，即可得到中断处理代码的入口。

2. 请编程完善kern/trap/trap.c中对中断向量表进行初始化的函数idt\_init。在idt\_init函数中，依次对所有中断入口进行初始化。使用mmu.h中的SETGATE宏，填充idt数组内容。每个中断的入口由tools/vectors.c生成，使用trap.c中声明的vectors数组即可。

答：分析如下，

　　1) Ucore启动后，通过idt\_init函数初始化IDT表，IDT表的每个元素均为門描述符，记录一个中断向量对应的中断处理函数的段选择子、偏移量和属性（门类型、DPL等），所以初始化IDT表就是初始化每个中断向量的这些属性。

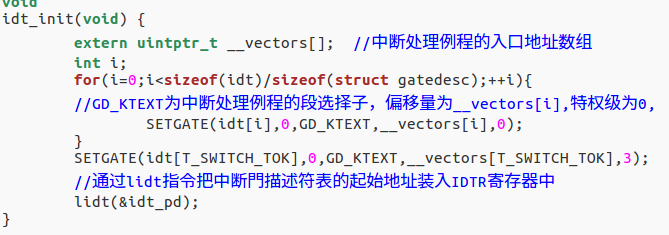
　　2) 除了系统调用(T\_SYSCALL)的门类型为陷阱門、DPL=3（用户级权限）以外，其它终端的门类型均为中断門、DPL=0（内核级权限，即仅能够使用int 0x30指令）。

　　3) vectors中存储了中断处理程序的入口程序和入口地址，即该数组中第i个元素对应第i个中断向量的中断处理函数地址。vectors定义在vector.S文件中，通过一个工具程序vector.c生成。而且由vector.S文件开头可知，中断处理函数属于.text的内容。因此，中断处理函数的段选择子即.text的段选择子GD\_KTEXT。从kern/mm/pmm.c可知.text的段基址为0，因此中断处理函数地址的偏移量等于其地址本身。

　　4) 使用mmu.h中的SETGATE宏来填充idt数组的内容，传递的参数有向量的首地址、門的类型、是否为系统调用、段选择子、偏移地址和DPL。

　　5) 完成IDT表的初始化之后，还需执行’LIDT’命令将IDT表的起始地址加载到IDTR寄存器中。LIDT指令的作用：使用一个包含线性地址基址和界限的内存操作数来加载IDT。用来在OS创建IDT时设定IDT的起始地址。该指令只能在特权级0执行。

　　根据上面分析，写出idt\_init函数的源代码：



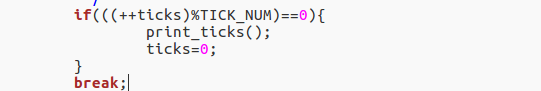
3. 请编程完善trap.c中的中断处理函数trap，在对时钟中断进行处理的部分填写trap函数中处理时钟中断的部分，使操作系统每遇到100次时钟中断后，调用print\_ticks子程序，向屏幕上打印一行文字”100 ticks”。

答：分析如下，

　　1) Trap函数只是调用了trap\_dispatch函数，而trap\_dispatch函数实现了对各种中断的处理，这题只要我们完成对时钟中断的处理，也就是trap\_dispatch函数中第一个case语句。

　　2) 可以使用kern/driver/clock.c中的全局变量ticks记录当前始终中断次数，每次发生时钟中断则将ticks加一，如果加一之后ticks==100，则调用print\_ticks子函数打印相关信息，并将ticks置0.

　　经过以上分析，写出如下源代码：



　　实现效果：

在lab1目录下执行’make qemu’，观察到如下结果，发现每过1s屏幕打印一次’100 ticks’，并且按下的键也会在屏幕上显示：

