

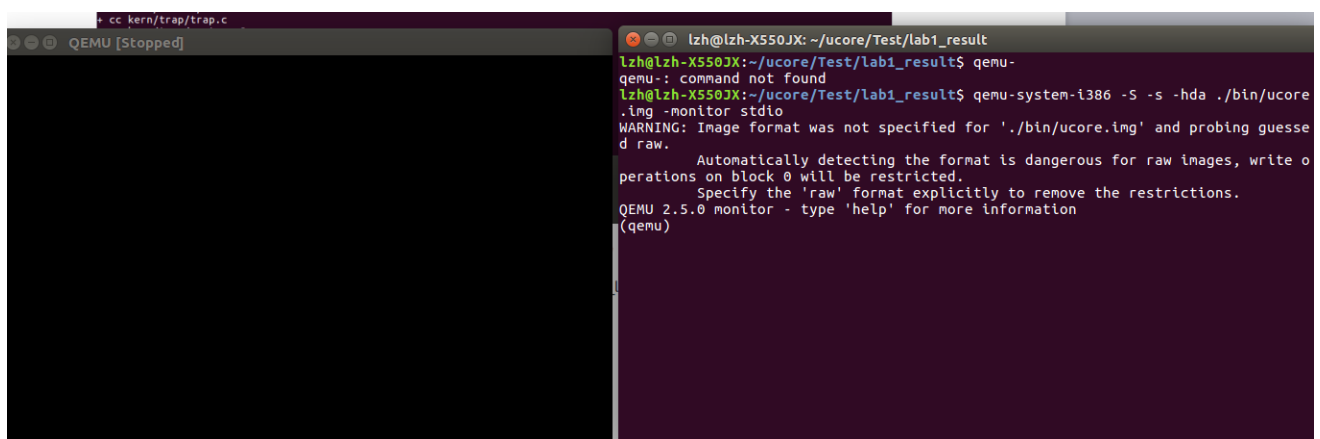
1. 在shell下调试lab1_result下ucore过程

在lab1_result下运行make

```
lzh@lzh-X550JX: ~/ucore/Test/lab1_result
lzh@lzh-X550JX:~/ucore/Test/lab1_result$ ls
boot kern libs Makefile report.md tools
lzh@lzh-X550JX:~/ucore/Test/lab1_result$ make
+ cc kern/init/init.c
+ cc kern/libs/stdio.c
+ cc kern/libs/readline.c
+ cc kern/debug/panic.c
kern/debug/panic.c: In function '__panic':
kern/debug/panic.c:27:5: warning: implicit declaration of function 'print_stackframe' [-Wimplicit-function-declaration]
    print_stackframe();
    ^
+ cc kern/debug/kdebug.c
+ cc kern/debug/kmonitor.c
+ cc kern/driver/clock.c
+ cc kern/driver/console.c
+ cc kern/driver/picirq.c
+ cc kern/driver/intr.c
+ cc kern/trap/trap.c
+ cc kern/trap/vectors.s
+ cc kern/trap/trapentry.S
+ cc kern/mm/pmm.c
+ cc libs/string.c
+ cc libs/printfmt.c
+ ld bin/kernel
+ cc boot/bootasm.S
+ cc boot/bootmain.c
+ cc tools/sign.c
+ ld bin/bootblock
'obj/bootblock.out' size: 500 bytes
build 512 bytes boot sector: 'bin/bootblock' success!
10000+0 records in
10000+0 records out
512000 bytes (5.1 MB, 4.9 MiB) copied, 0.0158818 s, 322 MB/s
1+0 records in
1+0 records out
512 bytes copied, 7.8399e-05 s, 6.5 MB/s
146+1 records in
146+1 records out
74880 bytes (75 kB, 73 KiB) copied, 0.000283252 s, 264 MB/s
lzh@lzh-X550JX:~/ucore/Test/lab1_result$
```

在lab1_result下运行qemu-system-i386 -S -s -hda ./bin/ucore.img -monitor stdio

可以打开qemu



```
cc kern/trap/trap.c
QEMU [Stopped]
lzh@lzh-X550JX:~/ucore/Test/lab1_result
lzh@lzh-X550JX:~/ucore/Test/lab1_result$ qemu-
qemu-: command not found
lzh@lzh-X550JX:~/ucore/Test/lab1_result$ qemu-system-i386 -S -s -hda ./bin/ucore
.img -monitor stdio
WARNING: Image format was not specified for './bin/ucore.img' and probing guesse
d raw.
Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
QEMU 2.5.0 monitor - type 'help' for more information
(qemu)
```

ng)

qemu中的CPU并不会马上开始执行，这时我们启动gdb，然后在gdb命令行界面下，使用下面的命令连接到qemu：

```
lzh@lzh-X550JX: ~/ucore/Test/lab1_result/bin
lzh@lzh-X550JX:~/ucore/Test/lab1_result/bin$ ls
bootblock  kernel  sign  ucore.img
lzh@lzh-X550JX:~/ucore/Test/lab1_result/bin$ gdb kernel
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from kernel...done.
(gdb) target remote:1234
Remote debugging using :1234
0x0000ffff in ?? ()
(gdb)
```

Target remote:1234

然后测试一下memset,并在12打一个断点,此时qemu出现booting from hard disk...

```
lzh@lzh-X550JX: ~/ucore/Test/lab1_result/bin
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from kernel...done.
(gdb) target remote:1234
Remote debugging using :1234
0x0000ffff in ?? ()
(gdb) break memset
Breakpoint 1 at 0x102db9: file libs/string.c, line 271.
(gdb) break 12
Breakpoint 2 at 0x100000: file kern/init/init.c, line 12.
(gdb) c
Continuing.

Breakpoint 2, kern_init () at kern/init/init.c:17
17 kern/init/init.c: No such file or directory.
(gdb)

QEMU [Stopped]
SeaBIOS (version Ubuntu-1.8.2-1ubuntu1)

iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F92460+07ED2460 C980

Booting from Hard Disk...
```

继续运行c,直到qemu模拟器出现100triks

```
lzh@lzh-X550JX: ~/ucore/Test/lab1_result/bin
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from kernel...done.
(gdb) target remote:1234
Remote debugging using :1234
0x0000ffff in ?? ()
(gdb) break memset
Breakpoint 1 at 0x102db9: file libs/string.c, line 271.
(gdb) break 12
Breakpoint 2 at 0x100000: file kern/init/init.c, line 12.
(gdb) c
Continuing.

Breakpoint 2, kern_init () at kern/init/init.c:17
17 kern/init/init.c: No such file or directory.
(gdb) c
Continuing.

Breakpoint 1, memset (s=0x10ea16, c=0 '\000', n=4970) at libs/string.c:271
271 libs/string.c: No such file or directory.
(gdb) c
Continuing.

QEMU - Press Ctrl-Alt to exit mouse grab
0: es = 10
0: ss = 10
+++ switch to user mode +++
1: 0ring 3
1: cs = 1b
1: ds = 23
1: es = 23
1: ss = 23
+++ switch to kernel mode +++
2: 0ring 0
2: cs = 8
2: ds = 10
2: es = 10
2: ss = 10
100 ticks
100 ticks
kbd [000]
kbd [000]
kbd [055] 7
kbd [055] 7
kbd [055] 7
kbd [055] 7
100 ticks
kbd [055] 7
```

/Selection_024.png)

2. 练习3

BIOS将通过读取硬盘主引导扇区到内存，并转跳到对应内存中的位置执行bootloader。请分析bootloader是如何完成从实模式进入保护模式的。

提示：需要阅读小节“保护模式和分段机制”和lab1/boot/bootasm.S源码，了解如何从实模式切换到保护模式，需要了解：

- 为何开启A20，以及如何开启A20
- 如何初始化GDT表
- 如何使能和进入保护模式

分析bootloader 进入保护模式的过程。

从 `%cs=0 $pc=0x7c00`，进入后

首先清理环境：包括将flag置0和将段寄存器置0

```
.code16
cli
cld
xorw %ax, %ax
movw %ax, %ds
movw %ax, %es
movw %ax, %ss
```

开启A20：通过将键盘控制器上的A20线置于高电位，全部32条地址线可用，可以访问4G的内存空间。

```
seta20.1:                # 等待8042键盘控制器不忙
    inb $0x64, %al        #
    testb $0x2, %al       #
    jnz seta20.1          #

    movb $0xd1, %al       # 发送写8042输出端口的指令
    outb %al, $0x64       #

seta20.1:                # 等待8042键盘控制器不忙
    inb $0x64, %al        #
    testb $0x2, %al       #
    jnz seta20.1          #

    movb $0xdf, %al       # 打开A20
    outb %al, $0x60       #
```

初始化GDT表：一个简单的GDT表和其描述符已经静态储存在引导区中，载入即可

```
lgdt gdt desc
```

进入保护模式：通过将cr0寄存器PE位置1便开启了保护模式

```
movl %cr0, %eax
orl $CR0_PE_ON, %eax
movl %eax, %cr0
```

通过长跳转更新cs的基地址

```
ljmp $PROT_MODE_CSEG, $protcseg
.code32
protcseg:
```

设置段寄存器，并建立堆栈

```
movw $PROT_MODE_DSEG, %ax
movw %ax, %ds
movw %ax, %es
movw %ax, %fs
movw %ax, %gs
movw %ax, %ss
movl $0x0, %ebp
movl $start, %esp
```

转到保护模式完成，进入boot主方法

```
call bootmain
```

3. 练习6：完善中断初始化和处理（需要编程）

请完成编码工作和回答如下问题：

1. 中断描述符表（也可简称为保护模式下的中断向量表）中一个表项占多少字节？其中哪几位代表中断处理程序的入口？
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。
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”。

【注意】除了系统调用中断(T_SYSCALL)使用陷阱门描述符且权限为用户态权限以外，其它中断均使用特权级(DPL)为0的中断门描述符，权限为内核态权限；而ucore的应用程序处于特权级3，需要采用`int 0x80`指令操作（这种方式称为软中断，软件中断，Tra中断，在lab5会碰到）来发出系统调用请求，并要能实现从特权级3到特权级0的转换，所以系统调用中断(T_SYSCALL)所对应的中断门描述符中的特权级（DPL）需要设置为3。

要求完成问题2和问题3提出的相关函数实现，提交改进后的源代码包（可以编译执行），并在实验报告中简要说明实现过程，并写出对问题1的回答。完成这问题2和3要求的部分代码后，运行整个系统，可以看到大约每1秒会输出一行“100 ticks”，而按下的键也会在屏幕上显示。

提示：可阅读小节“中断与异常”。

A1:中断向量表一个表项占用8字节，其中2-3字节是段选择子，0-1字节和6-7字节拼成位移，两者联合便是中断处理程序的入口地址。

A2（1）每个中断服务套路（ISR）的入口地址在哪里？所有ISR的都存储在vectors中。

__vectors []在kern / trap / vector.S中，由tools / vector.c生成（在lab1中尝试“make”命令，在kern / trap DIR中找到vector.S）使用“extern uintptr_t __vectors [];”定义将在以后使用的外部变量。（2）在中断描述表（IDT）中设置ISR的条目。idt [256]是IDT,使用SETGATE宏来设置IDT的每个项目（3）设置IDT的内容后，通过“lidt”指令让CPU知道IDT在哪里。你不知道这条指令的意思吗？只是谷歌吧！并检查libs / x86.h以了解更多信息。注意：lidt的参数是idt_pd。试着找到它！

```
22 /*
23 * Interrupt descriptor table:
24 *
25 * Must be built at run time because shifted function addresses can't
26 * be represented in relocation records.
27 */
28 static struct gatedesc idt[256] = {{0}};
29
30 static struct pseudodesc idt_pd = {
31     sizeof(idt) - 1, (uintptr_t)idt
32 };
33
34 /* idt_init - initialize IDT to each of the entry points in kern/trap/vectors.S */
35 void
36 idt_init(void)
37 {
38     /* LAB1 YOUR CODE : STEP 2 */
39     /* (1) Where are the entry addrs of each Interrupt Service Routine (ISR)?
40      * All ISR's entry addrs are stored in __vectors, where is uintptr_t __vectors[] ?
41      * __vectors[] is in kern/trap/vector.S which is produced by tools/vector.c
42      * (Try "make" command in lab1, then you will find vector.S in kern/trap DIR)
43      * You can use "extern uintptr_t __vectors[];" to define this extern variable which will be used later.
44      * (2) Now you should setup the entries of ISR in Interrupt Description Table (IDT).
45      * Can you see idt[256] in this file? Yes, it's IDT! you can use SETGATE macro to setup each item of IDT
46      * (3) After setup the contents of IDT, you will let CPU know where is the IDT by using 'lidt' instruction.
47      * You don't know the meaning of this instruction? just google it! and check the libs/x86.h to know more.
48      * Notice: the argument of lidt is idt_pd. try to find it!
49     */
50     extern uintptr_t __vectors[]; // __vectors[] You can use "extern uintptr_t __vectors[];" to define this extern variable which will be used later.
51     int i;
52     for(i=0; i<256; i++) {
53         SETGATE(idt[i], 0, GD_KTEXT, __vectors[i], DPL_KERNEL); // use SETGATE macro to setup each item of IDT
54     }
55     SETGATE(idt[T_SWITCH_TOK], 0, GD_KTEXT, __vectors[T_SWITCH_TOK], DPL_USER);
56     lidt(&idt_pd); // just google it! and check the libs/x86.h to know more. And the argument of lidt is idt_pd
57 }
58
59 }
```

A3: 处理定时器中断（1）定时器中断后，使用全局变量（增加它）记录这个事件，比如kern / driver / clock.c中的ticks（2）每个TICK_NUM循环，您可以使用函数打印一些信息，例如print_ticks()

```
~/Terminal$ file /viewcode$ cd /terminal Help
152 static void
153 trap_dispatch(struct trapframe *tf) {
154     char c;
155
156     switch (tf->tf_trapno) {
157     case IRQ_OFFSET + IRQ_TIMER:
158         /* LAB1 YOUR CODE : STEP 3 */
159         /* handle the timer interrupt */
160         /* (1) After a timer interrupt, you should record this event using a global variable (increase it), such as ticks in kern/driver/clock.c
161          * (2) Every TICK_NUM cycle, you can print some info using a function, such as print_ticks().
162          * (3) Too Simple? Yes, I think so!
163          */
164         ticks++;
165         if (ticks % TICK_NUM == 0) {
166             print_ticks();
167         }
168         break;
169     case IRQ_OFFSET + IRQ_COM1:
170         c = cons_getc();
171         cprintf("serial [%03d] %c\n", c, c);
172         break;
173     case IRQ_OFFSET + IRQ_KBD:
174         c = cons_getc();
175         cprintf("kbd [%03d] %c\n", c, c);
176         break;
177     //LAB1 CHALLENGE 1 : YOUR CODE you should modify below codes.
178     case T_SWITCH_TOK:
179         if (tf->tf_cs != USER_CS) {
180             switchk2u = *tf;
181             switchk2u.tf_cs = USER_CS;
182             switchk2u.tf_ds = switchk2u.tf_es = switchk2u.tf_ss = USER_DS;
183             switchk2u.tf_esp = (uint32_t)tf + sizeof(struct trapframe) - 8;
184
185             // set eflags, make sure ucore can use io under user mode.
186             // if CPL > IOPL, then cpu will generate a general protection.
187             switchk2u.tf_eflags |= FL_IOPL_MASK;
188
189             // set temporary stack
190             // then iret will jump to the right stack
191             *((uint32_t *)tf - 1) = (uint32_t)&switchk2u;
192         }
193         break;
194     case T_SWITCH_TOK:
195         if (tf->tf_cs != KERNEL_CS) {
196             tf->tf_cs = KERNEL_CS;
197             tf->tf_ds = tf->tf_es = KERNEL_DS;
198             tf->tf_eflags &= ~FL_IOPL_MASK;
199             switchu2k = (struct trapframe *) (tf->tf_esp - (sizeof(struct trapframe) - 8));
200             memmove(switchu2k, tf, sizeof(struct trapframe) - 8);
201             *((uint32_t *)tf - 1) = (uint32_t)switchu2k;
202         }
203         break;
204     case IRQ_OFFSET + IRQ_IDE1:
205     case IRQ_OFFSET + IRQ_IDE2:
206         /* do nothing */
207         break;
208     default:
209         // in kernel, it must be a mistake
210         if ((tf->tf_cs & 3) == 0) {
211             print_trapframe(tf);
212         }
213     }
214 }
```

CodeStore/Mytest/Lab1/code/trap.c

4.实验小结

了解了几种调试命令。