Operating System Design & Implementation

Lab3: X86 I/O System and Interrupt

TA:陳勇旗

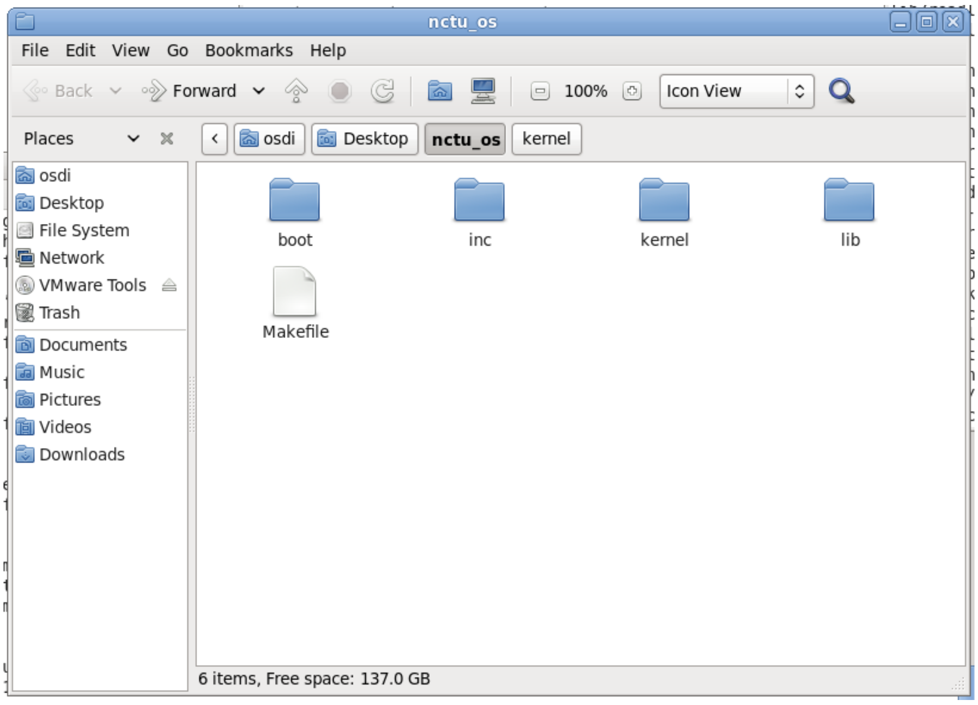
Objective:

In this lab you can learn

- Understand how the x86 system real/ protected mode and basic memory segmentation mechanism.
- Understand the basic I/O mechanism.
- Understand how to write an x86 interrupt service routine (ISR).
- Implement keyboard and timer interrupt initial functions.

1. Trace the simple kernel code

In this lab we already prepared a simple kernel in lab3 folder. You can check out from your SVN folder.



After checkout the lab, you can just use below commands to start this kernel, moreover you can use the qemu monitor to dump more register information that will help you debugging the OS.

Reference: http://en.wikibooks.org/wiki/QEMU/Monitor

```
$make all

$qemu –hda kernel.img –monitor stdio
```

1.1. Files

File Name	Description
boot/boot.S	A simple boot loader. It only changes CPU into protected mode and setup basic GDT.
boot/main.c	A simple ELF loader that will load the kernel image to expect memory address.
kernel/entry.S	Kernel entry, there we just setup 8*4096 bytes as kernel stack space and jump into C environment.
kernel/main.c	Kernel initial function
kernel/picirq.c	Programmable interrupt controller driver, it is used for setup external hardware interrupt.
kernel/kbd.c	Keyboard driver, used for read character from keyboard.
kernel/screen.c	Simple video driver, it allows you output string to screen.
kernel/timer.c	Simple system clock driver
kernel/trap.c	Trap handler
kernel/trap_entry.S	The trap/interrupt entry, you need to define the interrupt entry point here.
kernel/shell.c	A simple command shell, You can use it to debug your kernel.
kernel/kern.ld	Kernel linker script, it tells linker how kernel memory placement is.
Lib/*.c	In this folder we prepared some useful library for you (Such as printf, memcpy, strlen…)

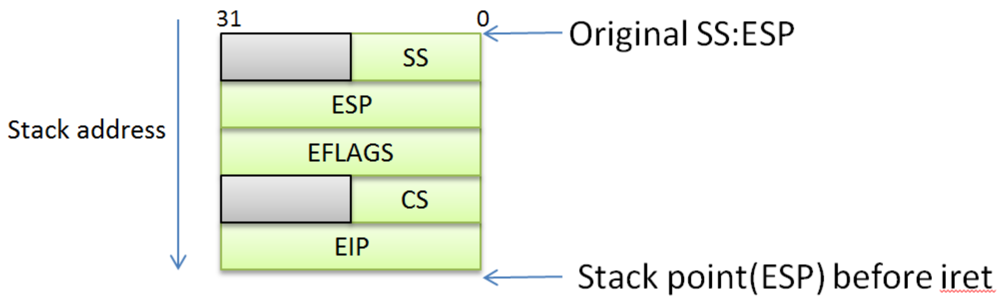
2. Lab Background

See the lab3_reference.pdf file and trace the lab3 source code.

3. Lab3 Requirements

3.1. Setup the trap stack same as Trapframe structure

In x86 system when a trap/interrupt occurred CPU will push some information into stack such as program counter (CS:EIP), stack point(SS:ESP) and error flag (EFLAGS) before trap. The placement will like below picture.



In this lib you need push trap number and more information (see TrapTrame structure and trap_entry.S) into stack, it can help interrupt handler known the CPU status before trap.

Hint: The trap number define in inc/trap.h

3.2. Implement keyboard interrupt setup

Read and modify the trap.c and trap_entry.S to enable keyboard interrupt.

Hint: We already write a keyboard interrupt handler in kbd.c, you need setup the interrupt descriptor table (IDT) in trap_init(), and define the entry point at trap_entry.S

3.3. Implement timer interrupt setup produce

Same as keyboard interrupt setup. Enable the system timer interrupt and let CPU can start counting the tick.

3.4. Add kernelinfo command

In shell.c, complete the mon_kerninfo function to print the kernel code and data size when user input kerninfo command.

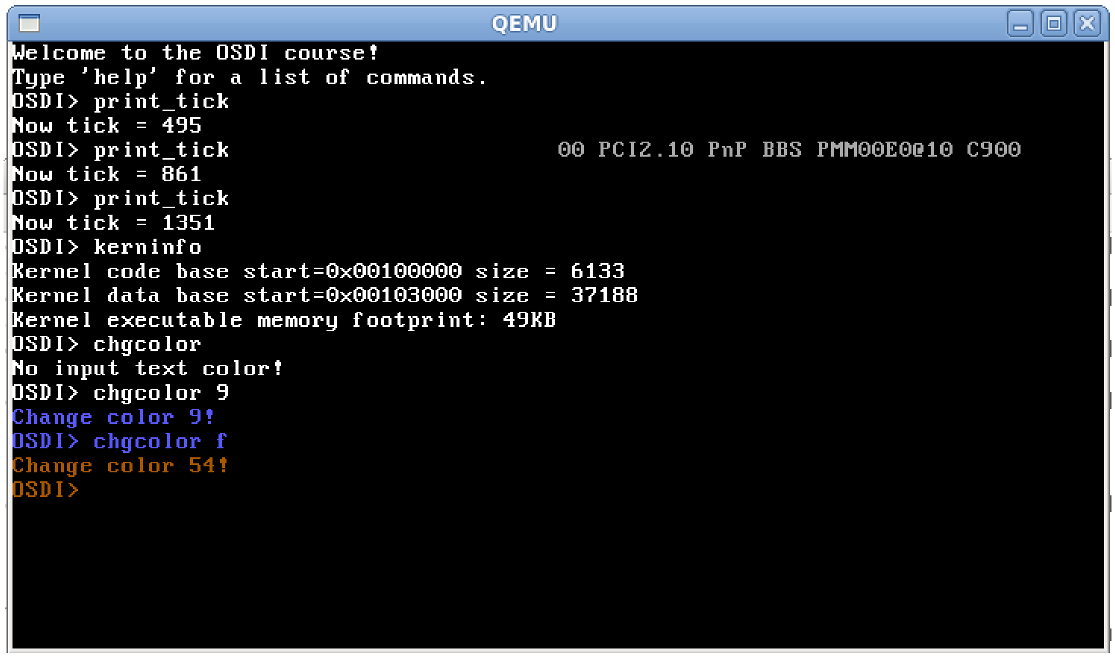
Hint: You can use the symbol form linker script (kern.ld) to calculate the section and memory footprint size.

3.5. Add chgcolor command

To better known how the VGA screen work, add a command "*chgcolor*" command to shell.c. When user input this command, the screen changes the text color. :)

4. Lab Result Demo

In this demo, you need to show result just like the following picture, and simply explain the interrupt flow.



```
Welcome to the OSDI course!
Type 'help' for a list of commands.
OSDI> print_tick
Now tick = 495
OSDI> print_tick                        00 PCI2.10 PnP BBS PMM00E0@10 C900
Now tick = 861
OSDI> print_tick
Now tick = 1351
OSDI> kerninfo
Kernel code base start=0x00100000 size = 6133
Kernel data base start=0x00103000 size = 37188
Kernel executable memory footprint: 49KB
OSDI> chgcolor
No input text color!
OSDI> chgcolor 9
Change color 9!
OSDI> chgcolor f
Change color 54!
OSDI>
```

After you complete the lab, don't forget commit your modification to SVN.