LAB3简介

OS LAB

2008-3-23

主要内容

- Lab3实习目的
- Lab3时间安排
- Lab3实习要求
- Lab3背景知识
- Lab3内容简介

Lab3实习目的

- 深化对于进程（process）概念的理解
  - 进程是一个有态活动实体，也是OS进行资源分配和调度的基本单位
  - 进程于程序的区别
- 通过实习熟悉OS对进程的管理
  - 进程的生命周期
- 理解并熟悉中断处理和系统调用（syscall）

Lab3时间安排

- Lab3时间：3月26日至4月15日（3周）
- 第1周
  - PartA 进程环境和中断处理
- 第2周
  - PartB 缺页、断点异常及系统调用
- 第3周
  - Challenge、代码调试和文档

Lab3实习要求

- Exercise1—10
  - 必做，写入文档（Exercise3阅读即可）
- Questions
  - 必做，写入文档
- Challenge1
  - 通过修改宏实现类同代码压缩
  - 必做
- Other Challenges
  - 选作

Lab3背景知识

- 进程
- 中断和异常
- 系统调用
- GCC内联汇编

进程

- 定义：Process

  进程是具有独立功能的程序关于某个数据集合上的一次运行活动，是系统进行资源分配和调度的独立单位，又称任务（Task）

- lab3中对应于：Environment

```
struct Env {
        struct Trapframe env_tf;        // Saved registers
        LIST_ENTRY(Env) env_link;       // Free list link pointers
        envid_t env_id;                 // Unique environment identifier
        envid_t env_parent_id;          // env_id of this env's parent
        unsigned env_status;            // Status of the environment
        uint32_t env_runs;              // Number of times environment has run

        //  Address space
        pde_t *env_pgdir;       // Kernel virtual address of page dir
        physaddr_t env_cr3;             // Physical address of page dir

};
```

进程的管理

- 系统为了管理进程设置一个专门的数据结构：进程控制块（PCB：Process Control Block），用它来记录进程的外部特征，描述进程的运动变化过程（又称进程描述符、进程属性）
- Lab3中使用数据结构Env作为PCB

```
struct Env *envs = NULL;          // All environments
struct Env *curenv = NULL;        // The current env
static struct Env_list env_free_list;    // Free list
```

进程的管理(续)

- PCB的内容
  - （1）进程描述信息（PID等）
  - （2）进程控制信息（当前状态、优先级等）
  - （3）所拥有的资源和使用情况（打开文件等）
  - （4）CPU现场保护信息（寄存器信息、段页指针等）
- Lab3的Env结构中定义的内容

（1）

```
envid_t env_id;
envid_t env_parent_id;
```

（2）

```
unsigned env_status;        // Status of the environment
uint32_t env_runs;          // Number of times environment has ru
```

（4）

```
struct Trapframe env_tf;    // Saved registers
// Address space
pde_t *env_pgdir;           // Kernel virtual address of page dir
physaddr_t env_cr3;         // Physical address of page dir
```

进程的管理(续)

- PCB表
  - 系统把所有PCB组织在一起，并把它们放在内存的固定区域，就构成了PCB表
  - PCB表的大小决定了系统中最多可同时存在的进程个数，称为系统的并发度
- Lab3中的Env结构表
  - 等价于PCB表
  - 由envs指针指向，共有1024（NENV）个表项，即JOS系统并发度为1024

进程状态

- 进程（process）的三种基本状态
  - 运行态、就绪态、等待态
  - 进程在消亡前处于且仅处于三种基本状态之一
- Lab3中的进程（Env）状态
  - unsigned env_status; //在Env结构中定义
  - 三个状态：FREE, RUNNABLE, NOT_RUNNABLE

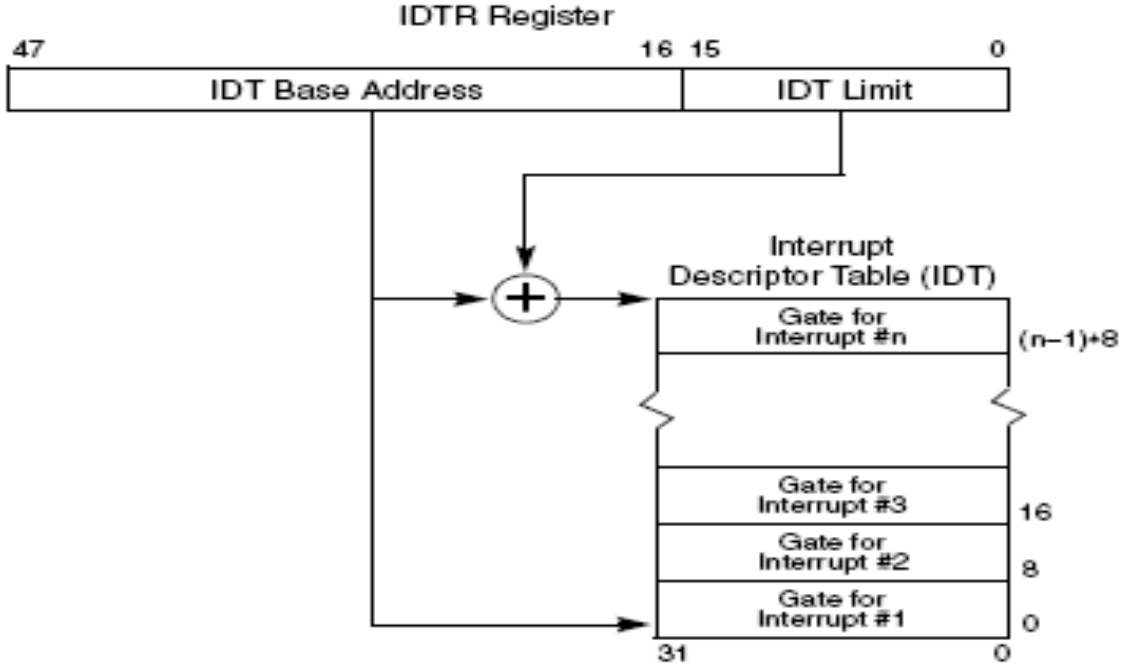
```
// Values of env_status in struct Env
#define ENV_FREE        0
#define ENV_RUNNABLE    1
#define ENV_NOT_RUNNABLE 2
```

进程映像（要素）

- 进程（process）要素
  - 代码段（用户程序）
  - 数据段（用户数据）
  - 用户栈（堆栈）
  - 进程控制块PCB （进程属性）
- ELF可执行文件格式
  - Lab3中加载一个用户进程对应的代码和数据时读取的对象是ELF格式文件
  - 在Lab3中由于没有文件系统，因此ELF可执行映像是内嵌在内核中的

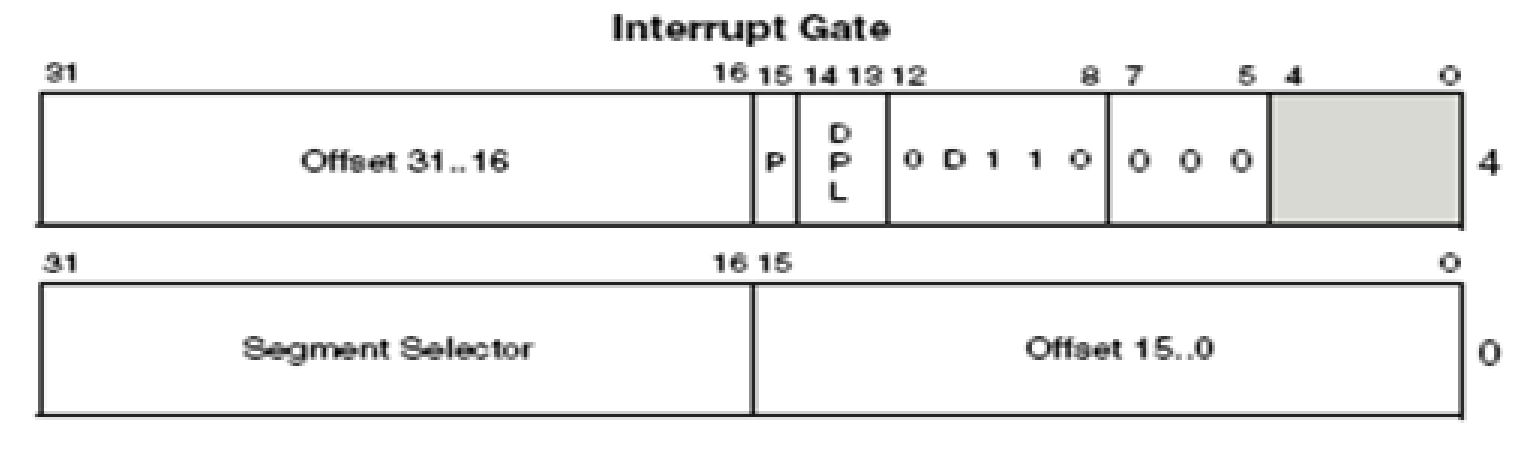
中断和异常

- 中断描述符表(IDT)
  - 中断描述符和每一个中断和异常向量的门描述符有关
  - 门描述符就是每一个中断或异常处理程序的入口地址



中断和异常

- 中断门(Interrupt Gate)
  - 指向目标代码：SegSelector+Offset
  - 权限：DPL

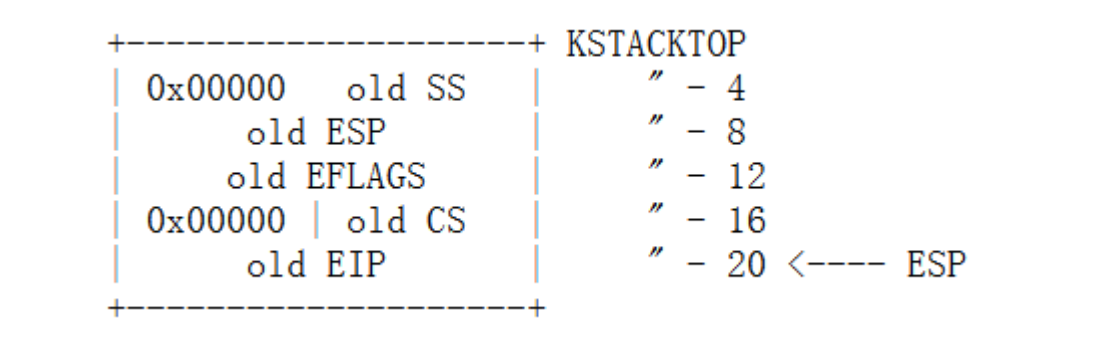


中断和异常

- JOS中断映射布局
  - 内部处理器异常：IDT[0--31]
  - 系统调用：　　int $0x30
- 堆栈切换
  - 内核模式下的异常处理：将异常参数压到当前堆栈
  - 用户模式下的异常/中断处理：切换到TSS中SS0，ESP0所指的堆栈

中断和异常

- 发生在用户模式下的除零中断(Int0)实例分析
  - 切换到TSS中SS0(GD_KD)，ESP0 (KSTACKTOP)所指的堆栈
  - 在内核堆栈压入必要信息



  - 设置%CS:%EIP使其指向IDT中0号中断对应中断门中保存的中断处理函数地址
  - 由除零中断处理函数处理该中断并返回

系统调用

- 系统调用开启
  - #define T_SYSCALL 0x30 /* system call */
  - 设置idt[0x30]的dpl为3允许用户调用
- 系统调用流程(x86)
  - 通过寄存器传递
  - 传递系统调用号:AX ； 5个参数:DX, CX, BX, DI, SI
  - 返回值:AX
- 理解user/syscall.c中syscall()

GCC内联汇编

- GCC支持在C/C++代码中嵌入汇编代码，这些汇编代码被称作GCC Inline Assembly——GCC内联汇编
- 功用：
  - 将一些C/C++语法无法表达的指令直接潜入C/C++代码中
  - 允许我们直接在C/C++代码中使用汇编编写简洁高效的代码

GCC内联汇编——基本格式

- 基本内联汇编的格式是：

```
__asm__ __volatile__("Instruction List");
```

- 说明：

1. __asm__是GCC 关键字asm 的宏定义：
   #define __asm__ asm
   __asm__或asm 用来声明一个内联汇编表达式，所以任何一个内联汇编表达式都是以它开头的，是必不可少的。
2. Instruction List 是汇编指令序列：（1）每条指令都必须被双引号括起来 (2)两条指令必须用换行或分号分开。
3. __volatile__是GCC 关键字volatile 的宏定义。如果用了它，则是向GCC 声明不允许对该内联汇编优化。

GCC内联汇编——扩展格式

- 扩展的内联汇编格式为

```
__asm__ __volatile__("Instruction List"
: Output
: Input
: Clobber/Modify);
```

- 说明：

1. Output 用来指定当前内联汇编语句的输出
2. Input 域的内容用来指定当前内联汇编语句的输入，Output和Input中，格式为形如“constraint”(variable)的列表（逗号分隔）
3. Clobber/Modify 声明当前内联汇编在Instruction List中对某些寄存器或内存进行修改。不能有Input或Output中限制的寄存器
4. 寄存器前必须使用两个百分号(%%)，而不是像基本汇编格式一样在寄存器前只使用一个百分号(%)

Lab3内容简介——Part A

- 阅读inc/env.h, kern/env.c, kern/env.h等，并了解进程环境相关背景
- 分配一个Environment数组（PCB表）
  - 在i386_vm_init()中完成Environment数组的内存分配和内存映射
- 完成进程环境的创建和运行
  - 实现env_init(), map_segment(), env_setup_vm(), load_icode(), env_create()，完成进程环境的创建
  - 实现env_run()，完成进程环境的运行

Lab3内容简介——Part A

- 处理中断和异常
  - 阅读《Intel 80386 manual》（见lab2参考资料）中第九章中断异常部分相关内容，了解x86的中断处理机制
  - 阅读kern/trap.h, inc/trap.h，了解中断映射布局
  - 修改kern/trapentry.S和trap.c完成中断向量表（IDT）的初始化
  - 确保通过实习文档中测试
  - 回答实习文档中的相关问题

Lab3内容简介——Part B

- 处理缺页异常（Page Faults）
  - 修改kern/trap.c中的trap_dispatch()函数，将缺页中断(14号中断T_PGFLT)分发到page_fault_handler() 来处理
  - 确保通过文档中的测试
- 处理断点异常(Breakpoints Exception)
  - 修改kern/trap.c中的trap_dispatch()函数，调用monitor来处理断点异常
  - 确保通过文档中的测试
  - 回答Ex6后面的两个问题

Lab3内容简介——Part B

- 系统调用
  - 阅读文档，了解JOS系统调用实现机制
  - 修改kern/trapentry.S和kern/trap.c，添加系统调用中断（中断号为T_SYSCALL）
  - 在kern/trap.c的trap_dispatch()函数中使用合适的参数调用内核的syscall函数来处理系统调用
  - 在kern/syscall.c中实现syscall()
  - 确保通过文档中的测试

Lab3内容简介——Part B

- 用户态进程启动过程(Startup)
  - 理解文档中对该过程的描述
  - 在用户lib库中补充代码，完成这个过程
- 缺页异常和内存保护
  - 了解文档中使用的内存保护机制
  - 修改kern/trap.c中的缺页中断处理函数来实现这种保护机制
  - 确保通过测试

0011

END