PA4

姓名：张浩南
学号：181860134

实验进程

PA4.1

实现 `_kcontext`，`schedule` 函数，修改_EVENT_YIELD事件的操作（不再只输出一句话），修改trap.S中的 `__am_asm_trap` ，使esp指向新的上下文结构
实现 `_ucontext` 并修改调度的代码，实现多道程序系统的功能

PA4.2

添加cr0,cr3寄存器实现 `page_translate` ，并修改 `isa_vaddr_read/isa_vaddr_write` 完成分页机制；添加 `_map` 和 `mm_brk` 函数，修改 `loader,_ucontext` 等函数的实现，使其符合分页机制，运行dummy得到good trap并成功运行仙剑奇侠传；在fs.c中添加yield的调用，实现hello与pal的双进程运行。

PA4.3

添加 `cpu.INTR` ，修改raise_intr代码让处理器进入关中断状态，在 `dev_raise_intr`，`exec_once` 中添加有关INTR的代码，实现抢占式的分时多任务；添加切换功能，成功加载3份仙剑奇侠传；修改之前PA4.2中存在的bug

必答题

理解分页机制

核心思想：将连续的存储空间分割成小片段, 以这些小片段为单位进行组织, 分配和管理，这些小片段统称为**页面**（分为虚拟页和物理页）
分页机制是一种按需分配的虚拟存储管理机制，其需要将虚拟页映射到对应的物理页上去，为了便于描述，分页机制引入**页表**结构，页表中的表项用于描述这种映射关系，页表项内容包括虚拟页的存放位置、装入位、修改位、访问权限位等。而每次加载程序时给程序分配物理页并为其准备一个新页表用于构建映射关系。
i386的分页机制采用二级页表的结构，其中一级页表**页目录**中的页目录项会存储对应页表的首地址，而对应页表的页表项中会存储对应的基地址，再通过页内偏移量合成物理地址。
其中页目录项和页表项中会有一些特殊字段用于分页保护和标志修改等功能。
CR3为**页目录基址寄存器**，用于保存页目录表在内存中的起始位置。

分时多任务的具体过程

当 `init_proc` 如下加载程序时

```
switch_boot_pcb();
  fg_pcb=1;
  context_uload(&pcb[0], "/bin/hello");
  context_uload(&pcb[1], "/bin/pal");
  context_uload(&pcb[2], "/bin/pal");
  context_uload(&pcb[3], "/bin/pal");
```

操作系统nanos-lite会在调用am中的 `_protect()` 以后进行 `loader()` ，在loader函数中与之前不同是，操作系统使用分页机制对其进行存储和加载，通过 `new_page` 获得新的物理页，并用到am中 `_map` 以建立虚拟地址与物理地址的映射关系，退出后调用am中的 `_ucontext` ，在栈上创建必要的上下文信息和参数信息，而分页机制则保证了进程在对应虚拟地址存取信息时，在nemu中能够通过 `page_translate()` 函数等完成到物理地址的转换，从而获得正确的信息
之后在nemu中运行时，每10ms在 `timer_intr()` 触发一次时钟中断，此时nemu检测到 `isa_query_intr()` 为true，同时也会在该函数中调用raise_intr，为中断事件做准备并产生异常号。而操作系统接受到 `_EVENT_IRQ_TIMER` 后调用 `_yield()` 强行暂停该进程，最后在schedule中更换当前进程，完成进程转换，如此反复形成分时运行。

遇到的问题及相应措施

PA4是正式阶段的最后一步，但在实验过程中也感觉到了PA4.2部分的困难度，以至于出现的错误直到PA4.3中才得以发现；
在PA4.2中实现完分页机制，成功在内核虚拟空间下运行仙剑奇侠传后，我开始实现 `_map` 并修改 `loader` ，起初修改loader时对应当清空的位置分配物理页出现错误导致无法运行，后来更改完成后运行dummy成功，而在下面实现完mm_brk后却出现了无法运行仙剑，运行hello程序只能输出一行然后中止的情况，具体错误主要是 `page_translate` 时出现了present=0的情况。
接下来尝试修了几个小bug无果后，询问同学得知将 `_sbrk` 略加修改：

```
//void *_sbrk(intptr_t increment),init_brk初始为NULL
intptr_t pre_brk=init_brk,now_brk=pre_brk+increment;
if (init_brk==NULL) init_brk=(uintptr_t)&_end;
if (_syscall_(SYS_brk,pre_brk,increment,0)==0)
{
   init_brk=now_brk;
   return (void*)pre_brk;
}
else return (void*)-1;
```

这个操作显然是错误的，但奇怪的是竟然能够成功运行hello和进入仙剑奇侠传，由于我当时也没有什么很好的办法，就暂时将其搁置了。
而到了PA4.3，实现完按键切换后，我发现只能 `F1-F2-F3` 的顺序，而再次按 `F2/F1` 时就会 `page_translate` 出错，我查了半天后来发现，nemu根本就无法进入仙剑的游戏页面，只能看过场动画，即进入新的游戏/游戏存档都会崩溃，这就是PA4.2遗留下来的问题。
解决过程使用”大腿“的代码进行替换，发现是loader的问题，我的loader中未对当前pcb的max_brk作修改，添加上相应操作后成功运行。
这次问题又一次深刻提醒，当你你确信写的代码有一定问题，连你自己都不能被说服时，那么除非马上修改，否则相应的错误一定会在后面找上门来。