Pa4 实验报告

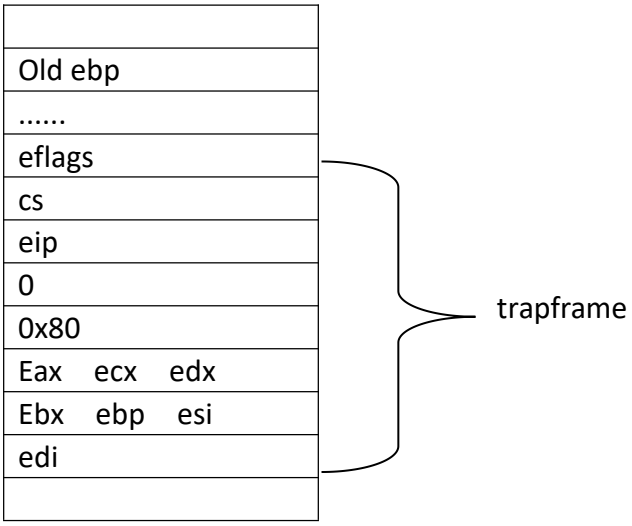
§4-1.3.1　通过自陷实现系统调用

1：

系统执行到 int $0x80 自陷指令后，即获取 intr_no = 0x80 并执行 raise_sw_intr()。在 raise_intr()函数中，将 eflags，cs 和 eip 依次压栈，并根据 IDT 的地址编号查询得到中断处理程序的入口地址，根据中断描述符的类型修改 IF 的值，修改 cpu.eip 的值。根据入口地址，跳转到 do_irq.s 中的入口函数，然后执行一系列 push，函数调用操作，输出 hello world；随后返回到 do_irq.S 中，执行 popa 和 iret 指令，恢复用户进程的通用寄存器；此时，cpu.eip 指向 int 指令的后一条指令 HIT_GOOD_TRAP。

2：将参数 trapframe 指针的内容传给 irq_handle;

栈图：



§4-1.3.2　响应时钟中断

1:

内部异常是在指令执行的过程中在 CPU 内部检测到的，一旦由 CPU 检测到，那么立即可以通过 raise_sw_intr()来启动异常处理流程。

而时钟中断是通过外部中断实现的。具体来说，就是通过专门设置的两个引脚，分别接可屏蔽中断请求线 INTR 和不可屏蔽中断请求线 NMI。当引脚被置为高电平，即逻辑值 1，的时候，意味着有一个中断事件到来了，需要 CPU 引起关注。如果处于开中断状态，CPU 会在每一条指令执行结束后，查看中断引脚的值，若发现有中断需要处理，则查询中断号并调用相应的处理程序。

§4-2.3.3　完成键盘的模拟

1：在 echo 函数中，调用 add_irq_handle 函数，将 IRQ_t 存入 handler 数组，由此完成注册。

2:echo 函数不断调用 hlt 指令。hlt 指令创建一个捕获键盘输入的线程，不断产生外部中断，进而实现字符的输入转换为控制台输出对应的字符。