PA4 实验报告

4-1

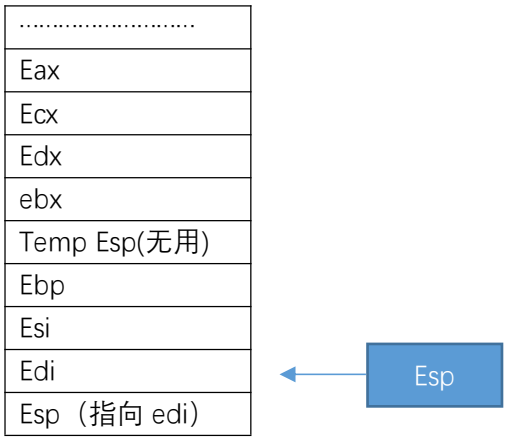
1. 关于 INT 0x80 指令的执行过程

首先 nemu 读取指令码，读取到 int 对应的指令码后，跳转到 int 指令函数中。在 int 指令中，我们调用了 raise_intr()函数，将调用号作为参数传入，调用号由 int 指令读取 eip+1 处的一个字节获得。随后在 raise_intr()函数中，我们将标志寄存器，段寄存器和 eip 压栈，之后读取 IDTR 中存放的 IDT 首地址，再根据传入的调用号跳转到相对应的中断处理程序。

这个跳转过程主要通过 kernel 中的 irq_handle()函数实现。首先该函数接收一个 TrapFrame*类型的参数，该参数由 esp 传入得到，因为在 do_irq 中我们可以看到在函数 irq_handle()被调用之前有一条 push %esp 的语句，显然是将 esp 作为参数传递给了 irq_handle()函数。

这里我们分析一下在 irq_handle 函数调用之前栈的情况。

首先我们进行了 pusha 指令，将所有寄存器压栈，随后我们又将%esp 压栈



所以通过最后将 esp 压栈，此时 esp 指向的是栈中 edi 的位置，我们即可以得到所有寄存器中的内容，TrapFrame 结构体对应的内容即是这段栈中的内容。

知道了参数传递的方法和参数内容后，在 irq_handle()函数中，我们调用了 sys_call()函数，将 TrapFrame 结构体作为参数传入。首先判断 eax 寄存器中的内容，决定了执行什么功能，调用对应的处理函数 sys_XXX()进行处理，最后返回。

2. 关于时钟中断与系统调用的异同

首先时钟中断利用的是 CPU 每次执行指令后检查一下中断引脚，当中断引脚被置为 1 时系统进入中断处理程序。调用时通过中段号进行中断处理程序跳转，后续中断处理程序的执行与系统调用相同。

4-2

1.键盘监听事件的注册是如何实现的？

通过调用 make_pio_handler 宏，我们定义了关于键盘的端口处理函数 handler_keyboard_data

随后我们在关于所有端口处理函数的数组 pio_handler_table[ ]中，将该处理函数添加进去，并且为其指定了一个端口号，从而完成了键盘监听事件的注册。

3. 从键盘按下一个键到控制台输出一个字符，系统的执行过程是什么？

首先我们在键盘按下一个键，键盘监听函数得到一个按键的扫描码，随后 echo 函数将该扫描码读取后，将其转换成对应的大写字母，以 ASCII 码形式存储。随后对有关串口的端口进行写入，再由串口的驱动程序将其显示在控制台上。