Lab3 实验报告

问题一

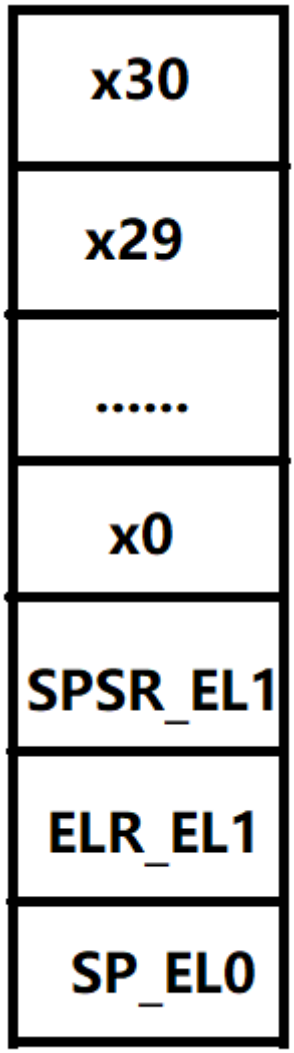
发生中断时，ARM 处理器在硬件层面保存中断前的 *PSTATE* 至 *SPSR_EL1*，保存中断前的 *PC* 到 *ELR_EL1*，保存中断原因信息到 ESR_EL1，把栈指针 SP 从 *SP_EL0* 换为 *SP_EL1*。
vbar_elx 寄存器存储了异常处理程序入口的基址地址。处理器会根据异常类型，切换 exception level（内核态），进入 *vbar_elx* 加上对应偏移量所对应的异常处理程序。在本 lab 中，只对设置了异常处理程序。
在进入对应的异常处理程序之前，需要先保存出错进程的状态。会先进入 `alltraps` 函数。在该函数中，先在内核空间的栈里保存一系列寄存器，调用 `trap` 函数执行异常的处理。完成 `trap` 函数在切换到用户态之前，再把保存的寄存器的值取出。
最后跳转回中断发生的位置，同时把当前 el 设置为0。

问题二

在 trapframe 中，需要把发生异常时所有通用寄存器值全部保存下来，所以在 trapframe 中设置 30 个 4 Byte 长度的变量（uint64_t） x0 ~ x30 来存这些寄存器。另外还需要保存 el0 下的状态寄存器 *ELR_EL1*， *SPSR_EL1*，和 *SP_EL0*。于是在 trapframe 中还需要增加 3 个 4 Byte 长度的变量（2 个 uint64_t 和 1 个 uint64_t *）。
关于是否需要把 X19 ~ X29 这些由调用者保存的寄存器保存到 trapframe 中的问题，我认为需要保存。对于多核处理器，调回的处理器可能不是原先的，需要保存原先处理器上所有的寄存器。对于单核处理器，可能异常恰好是因为调用者保存这些寄存器，向栈中压如寄存器的值，但此时 sp 已经到达新的页，但该页还未被分配的情况下发生。此时，这些寄存器还未被保存，所以 trapframe 中需要留出位置。

问题三

在处理异常之前，需要把 *PSTATE* 寄存器的 *DAIF* 寄存器置 0，取消异常屏蔽。
使用`stp`语句实现栈的push操作，令寄存器如图形式在 trapframe 中存储。在读写 *ELR_EL1*，*SPSR_EL1*，和 *SP_EL0* 这类状态寄存器时需要使用`mrs`和`msr`指令。



之后调用`trap`函数。通用寄存器中 x0 ~ x7 寄存器用于传参，观察 kernel8.asm 的`trap`函数，可以发现在汇编代码中`trap`函数关于传参的代码`ffff000000080e68: f9000fe0 str x0, [sp, #24]`，可知参数 *trapframe * tf* 是由 x0 寄存器传递。而在`trapall`函数中存储 结构体 trapframe 的位置在 sp，因此把 x0 的值赋为 sp 即可。
完成异常处理后，需要把 trapframe 中的寄存器还原回发生异常时的状态，即通过`ldp`指令从栈中按存入的反序弹出。之后，调用`eret`命令返回 *SPSR_EL0* 中的地址。同时，`eret`指令会更新*PSTATE* 到 *SPSR_EL0*，选择 *SP_EL0* 为栈指针，把模式从内核态切换为用户态。