Lab 1

- 李逸岩 519021911103
- lyy0627@sjtu.edu.cn

1

```
    # MPDIR_EL1 是core identification Register, Aff1字段会存储CPUID(Cortex A75是
0x0-0x7)
    # 如果是四核可能是0x0-0x3, 移动到tmp register X8
    mrs x8, mpidr_el1
    # 将X8和0对比，只有相等的时候才进入primary branch继续初始化工作，这保证了只有CPUID为0的
才能完成初
    # 其他核将在这一步结束
    cbz x8, primary
```

2

```
    mrs x9, CurrentEL

    # Test
    # (gdb) p/x $x9
    # >>>>  0xc
```

根据文档，0b1100对应EL3级别，实现正确。

3

```
    # (0b1111 << 6) | (0b0101) 放入x11
    mov x11, SPSR_ELX_DAIF | SPSR_ELX_EL1H
    # 把x11 放入spsr_el3, 实际上将0, 2, 6, 7, 8 位置为1.
    # bits [3:0]表示的是EL等级和选中的栈指针，按照要求，我们需要跳转到EL1并且使用SP_EL1
    # 选择EL1h 的条目，所以M[3: 0] = 0b0101.
    # M[9: 6]为四个Mask位，分别代表FIQ interrupt mask, IRQ interrupt mask,
    # SError interrupt mask, Debug exception mask. 为了暂时屏蔽所有中断，应该把
    # M[8: 6]设置为1.
    msr spsr_el3, x11
    # When taking an exception to EL3, holds the address to return to.
    # 将Ltarget的地址存入，执行eret之后就会取出Ltarget从而执行流到Ltarget返回start函数
    adr x11, .Ltarget
    # 将x11 放入elr_re3
    msr elr_el3, x11
    # barrier
    isb
    eret
    # 执行流回到start函数，符合预计结果
```

4

因为C语言在函数调用，函数返回，寄存器溢出，函数传参等情况下需要栈空间。

如果不设置栈，当控制流运行到init_c的第一条指令(入栈指令) `883a8:  a9bf7bfd   stp    x29, x30, [sp, #-16]!` 的时候就无法继续执行了。

5

ICS中讲到，初始化为0的全局变量会被存入.bss section. 如果有这样的变量但是在加载时没有zero .bss, 那么程序可能出现意料之外的错误。此外，如果程序员默认全局变量不初始化会被初始化为0，程序也可能出现错误。

6

```
        while (*str != '\0') {
                early_uart_send(*(str++));
        }
        // >> boot: init_c
        // 测试通过
```

7

```
    orr     x8, x8, #SCTLR_EL1_M
    # SET Bit of MMU to 0b1
```

continue后，输出了:

```
boot: init_c
[BOOT] Install boot page table
[BOOT] Enable el1 MMU
[BOOT] Jump to kernel main
```

然后程序无输出。终止GDB执行，发现程序运行在0x200的位置上，测试通过。

Ref

https://developer.arm.com/documentation/

https://users.ece.utexas.edu/~adnan/gdb-refcard.pdf

https://modexp.wordpress.com/2018/10/30/arm64-assembly/