实验报告（一）

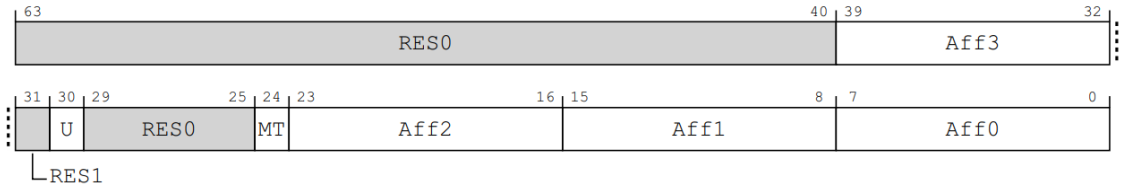
思考题1：

在函数 `_start` 的开头，首先将系统寄存器 `mpidr_el1` 中的内容加载到普通寄存器 `x8` 中，根据手册（如下图示），`mpidr_el1` 的低八位包含可分辨CPU核心的信息，因此将 `x8` 和 `0xFF` 按位相与做掩码得到低八位信息，若等于0则表示让零号核进入初始化流程，其余核心则将不断在两个 `b` 分支语句间无限循环跳转，达到暂停执行目的。

Attributes

MPIDR_EL1 is a 64-bit register.

Field descriptions



Aff0, bits [7:0]

Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior. The assigned value of the MPIDR.{Aff2, Aff1, Aff0} or MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole.

练习题2：

在 `arm64_elX_to_el1` 函数的 `LAB 1 TODO 1` 处填写的代码为：

```
1    mrs x9, CurrentEL
```

通过系统寄存器 `CurrentEL` 获取当前异常级别。

练习题3：

在 `arm64_elX_to_el1` 函数的 `LAB 1 TODO 2` 处填写的代码为：

```
1    adr x9, .Ltarget
2    msr elr_el3, x9
3    mov x9, SPSR_ELX_DAIF | SPSR_ELX_EL1H
4    msr spsr_el3, x9
```

首先，将目标跳转地址的 label （`.Ltarget`）放入 `x9` 寄存器；再将该地址写入异常链接寄存器 `elr_el3` 供 `eret` 指令控制异常返回后执行的指令地址。接着，通过 `x9` 设置 `spsr_el3` 寄存器，达到暂时屏蔽所有中断并使用内核栈的目的。

思考题4：

接下来要运行C代码，函数调用时需要栈来保存临时变量、返回地址等信息。如果不预先设置栈地址，C语言中定义的局部变量将无从存放，程序不能正常运行。

思考题5：

在C语言中，规定未初始化的全局变量和静态变量初始值为零。如果不清理 `.bss` 段，之后使用未初始化的全局变量时将出现错误值。

练习题6：

在 `uart.c` 中 `LAB 1 TODO 3` 处填写的代码为：

```
1   early_uart_init();
2   for(int i = 0; str[i] != '\0'; i++)
3       early_uart_send(str[i]);
```

首先初始化 UART 串口，再输出字符串。

练习题7：

在 `tools.S` 中 `LAB 1 TODO 4` 处填写的代码为：

```
1   orr x8, x8, #SCTLR_EL1_M
```

通过设置 `M` 字段启用MMU。