思考题1：

_start函数首先将寄存器mpidr_el1的值加载到x8中，并将x8与0xFF做按位与操作获取x8的最后8位，查阅相关手册可知mpidr_el1的低8位与定义PE的行为相关

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.

而在_start函数中，后续会进行x8是否为零的检验，如果x8=0，则进入到primary中开始初始化，如果不等于0，则根据clear_bss_flag的值在跳转指令中作进行无限循环达到暂停执行的目的。

思考题2：

```
mrs x9, CurrentEL
```

思考题3：

```
adr x9, .Ltarget
msr elr_el3, x9
mov x9,  SPSR_ELX_DAIF | SPSR_ELX_EL1H
msr spsr_el3, x9
```

思考题4：

在运行C代码前需要设置栈来存放一些局部变量以及返回地址，如果不设置栈，程序无法正常运行

思考题5：

BSS 是程序中存储未初始化全局变量和静态变量的一种内存段，其内容在程序加载时会被初始化为零或默认值。如果不清除，在之后使用未初始化全局变量时可能会出错

思考题6：

```
early_uart_init();
int i = 0;
for(int i=0;str[i]!='\0';i++)
{
early_uart_send(str[i]);
}
```

思考题7：

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

思考题8：

优势：

1. 一般情况下减少了页表需要占据的空间，允许页表中出现空洞，有效压缩了页表的大小
2. 页表以更小的单位存在，与进程的关联性更紧密，一定程度提高了内存的命中率

劣势：

1. 在极端情况下，多级页表可能占用更多的内存
2. 结构更复杂，访问速度可能更慢
3. 可能会引入更多的PageFault，每次加载页的时候都会引起延迟

以4KB粒度映射的页表页数量：

物理页：4G/4K=1M

3级页表：1M/512 = 2048

2级页表：2048/512=4

一级页表：1

零级页表：1

共：2054

以2MB粒度映射的页表页数量：

物理页：4G/2M = 2048

二级页表：2048/512=4

一级页表：1

零级页表：1

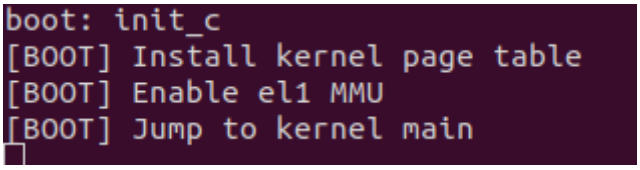
共：6

思考题9：

```
boot_ttbr1_l0[GET_L0_INDEX(kva)] = ((u64) boot_ttbr1_l1) | IS_TABLE | IS_VALID;
boot_ttbr1_l1[GET_L1_INDEX(kva)] = ((u64) boot_ttbr1_l2) | IS_TABLE | IS_VALID;
```

思考题10：

为低地址配置页表：在设置好MMU后，Chcore的指令地址位于低地址空间，因此需要配置低地址页表



如果不设置，不能正常运行。