CS315 Lab 7

Name: 王奕童

SID: 11910104

我这个lab是和term project的几个组员张睿豪，谢岳臻一起完成的。

2 Before the Task

我们使用了张睿豪烧录好镜像的树莓派板子，期间使用了我的显示器做可视化桌面显示，通过VNC Viewer建立远程连接。

3 Lab Tasks

3.1 Task 1: Create and Install a Kernel Module

运行课件上的指令，并将 `dmesg` 的内容输出到文件中：

```
pi@raspberrypi:~/Desktop/lab7_2 $ dmesg > dmesg_old.txt
```

```
pi@raspberrypi:~/Desktop/lab7_2 $ dmesg > dmesg_new.txt
```

运行diff命令比较其中的不同，即为系统日志中的不同：

```
pi@raspberrypi:~/Desktop/lab7_2 $ diff dmesg_new.txt dmesg_old.txt
303,306d302
< [ 1798.687947] simple_module: loading out-of-tree module taints kernel.
< [ 1798.687965] simple_module: module license 'unspecified' taints kernel.
< [ 1798.687971] Disabling lock debugging due to kernel taint
< [ 1798.689505] Hello world.
```

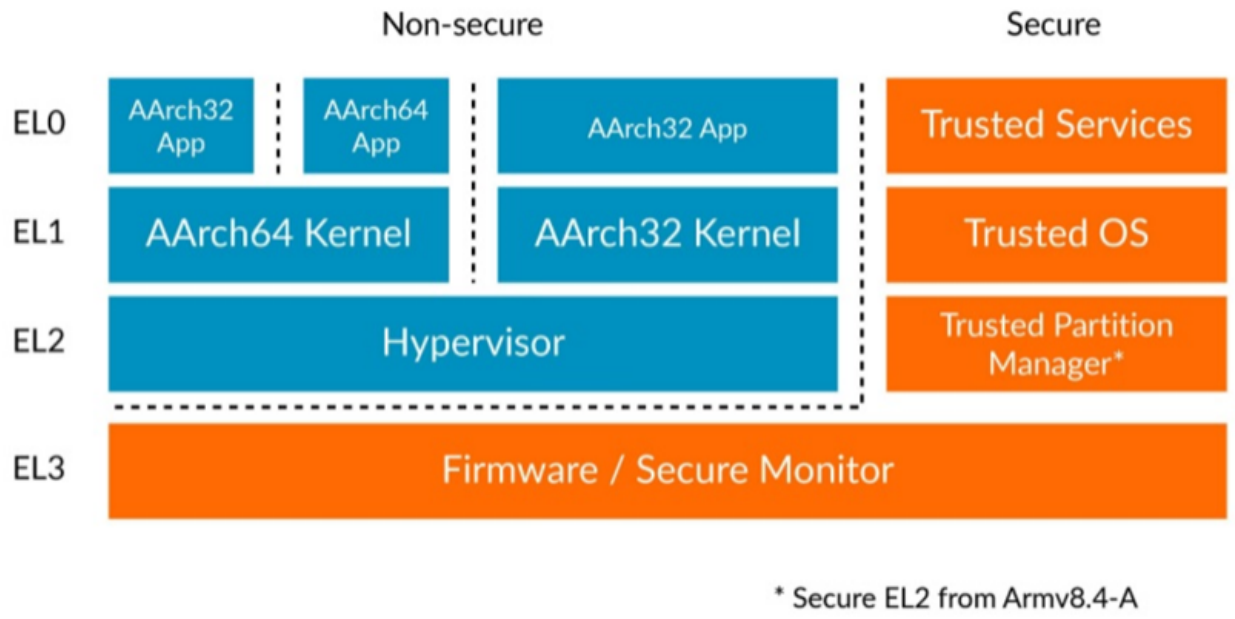
Task 1：What is the exception level and the security state of the core with loaded LKM?

LKM: loadable kernel modules

Exception level: EL1

Security State: Non-secure state

因为loaded LKM是kernel的一部分。



3.2 Task 2: Directly Access a High Privilege Register: SCR

Task 2.A: What is the complete instruction? Look up the manual and fill the instruction bellow. Then compile and execute.

这里填写的内容是 `c1, c1, 0`

```
/*
 * The init function of the module.
 */
static int __init simple_module_init(void)
{
    uint32_t reg;
    asm volatile("mrc p15, 0, %0, c1, c1, 0":"=r"(reg));
    printk(KERN_INFO "SCR %x.\n", reg);
    return 0;
}
```

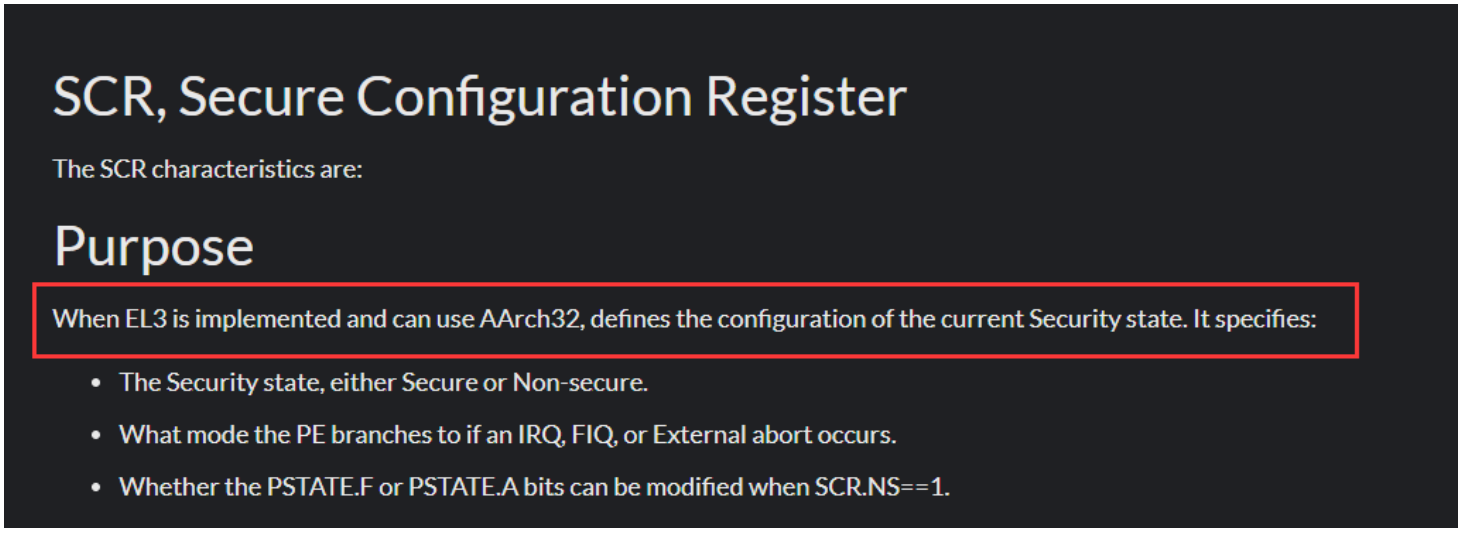
运行后得到了段错误：

```
pi@raspberrypi:~/Desktop/lab7_2 $ make
make -C /lib/modules/5.4.51-v7+/build M=/home/pi/Desktop/lab7_2 modules
make[1]: 进入目录“/usr/src/linux-headers-5.4.51-v7+”
  CC [M]  /home/pi/Desktop/lab7_2/hard_module.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/pi/Desktop/lab7_2/hard_modul
e.o
see include/linux/module.h for more information
  CC [M]  /home/pi/Desktop/lab7_2/hard_module.mod.o
  LD [M]  /home/pi/Desktop/lab7_2/hard_module.ko
make[1]: 离开目录“/usr/src/linux-headers-5.4.51-v7+”
pi@raspberrypi:~/Desktop/lab7_2 $ sudo insmod hard_module.ko
段错误
```

（这里重命名了一下，因为不太清楚如何卸载相关的module）

Task 2.B: Why the segmentation fault occurs?

这是因为SCR需要EL3权限，而LKM在EL1权限，权限不足就会触发段错误。



3.3 Task 3

Task 3.A: What is the instruction to read DBGAUTHSTATUS? Suppose we want to store it in Rt.

考虑手册第392页的说明：

Accessing DBGAUTHSTATUS_EL1

Accesses to this register use the following encodings in the System register encoding space:

```
MRS <Xt>, DBGAUTHSTATUS_EL1
```

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1110	0b110

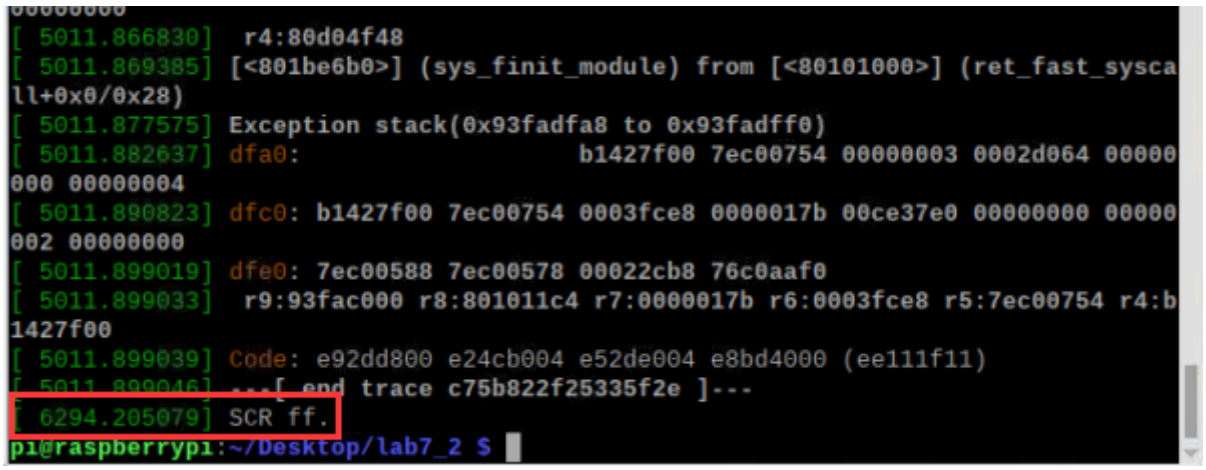
指令因此就是：

```
mrc p14, 0, %0, c7, c14, 6
```

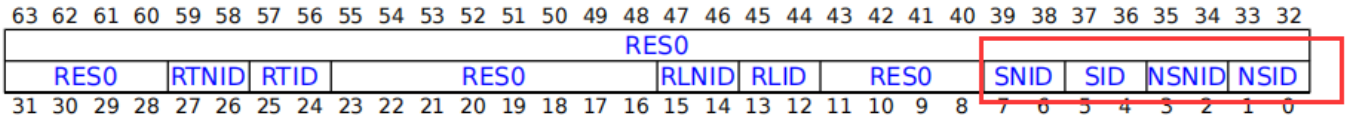
对应的代码是

```
asm volatile("mrc p14, 0, %0, c7, c14, 6":"=r"(reg))
```

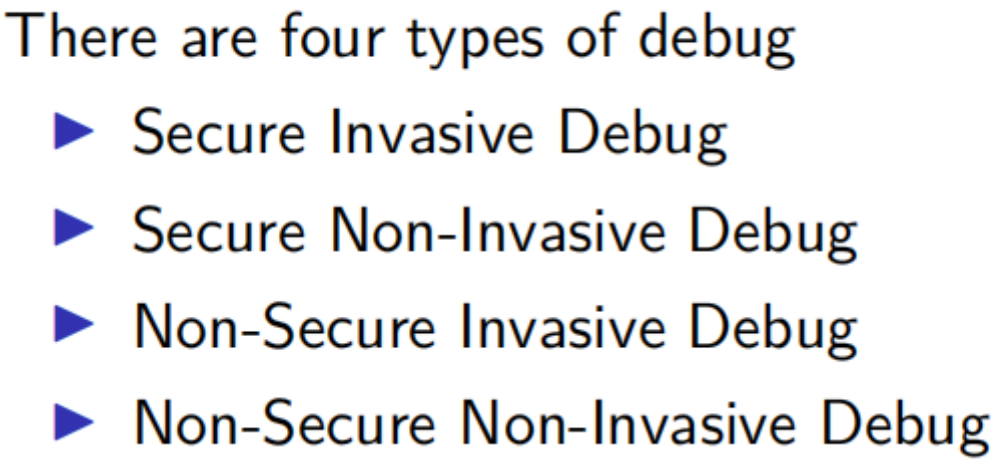
Task 3.B: Take single_module.c as an example, write your kernel module to read the signal. A screenshot of the result is needed, and tell us what is the meaning of the result. What kind of debug events are enabled?



考虑手册390页：

> **Field descriptions**
>
> 
>

课件中有对这4种的说明：



因为它是ff，所以就是4种debug都处于enabled状态。

3.4 Task 4: Enable the Halting Debug

对于里面问号内容的填写：

```
#define EDLAR_OFFSET 0xFB0
#define OSLAR_OFFSET 0x300

iowrite32(0xC5ACCE55, param->debug_register + EDLAR_OFFSET);
iowrite32(0xC5ACCE55, param->cti_register + EDLAR_OFFSET);
iowrite32(0x0, param->debug_register + OSLAR_OFFSET);
iowrite32(0x0, param->cti_register + OSLAR_OFFSET);
```

参考手册第4088页，得到EDLAR的offset内容：

Accessing EDLAR

EDLAR can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
Debug	0xFB0	EDLAR

This interface is accessible as follows:

- When FEAT_DoPD is not implemented or IsCorePowered(), accesses to this register are **WO**.
- Otherwise, accesses to this register generate an error response.

参考手册第4759页，得到OSLAR的offset内容为 `0xC5ACCE55` ：

Accessing OSLAR_EL1

Note

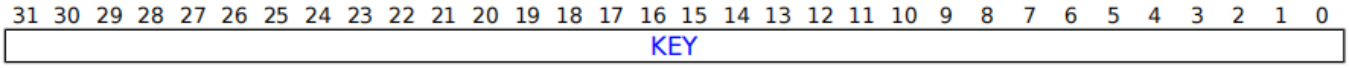
SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

OSLAR_EL1 can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x300	OSLAR_EL1

参考手册第4087页，得到EDLAR的解锁内容：

When Software Lock is implemented:



KEY, bits [31:0]

Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface.

Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

参考手册第1429页，得到OSLAR的解锁内容w为 `0x0`：

OSLK, bit [1]

OS Lock Status.

OSLK	Meaning
0b0	OS Lock unlocked.
0b1	OS Lock locked.

The OS Lock is locked and unlocked by writing to the OS Lock Access Register.

3.5 Halt the Processor

3.6 Task 5: Switch to the EL3 and read the SCR

Task 5.A: We mention how to access SCR directly in Task 2. You need to prepare an instruction, who reads SCR and store it to R1. Then convert it to machine code (do it on yourself) and execute it on the target.

首先读取SCR的指令是 `mrc p15,0,r1,c1,c1,0`

翻译为机器码，得到其大端存储为 `0xee111f11`：

mrc p15, 0, r1, c1, c1, 0

ARM ARM (thumb) AArch64 Mips (32) Mips (64) PowerPC (32) PowerPC (64)
Sparc x86 (16) x86 (32) x86 (64)

Inline Python

Assemble

Assembly - Little Endian

```
"\x11\x1f\x11\xee"
```

Assembly - Big Endian

```
"\xee\x11\x1f\x11"
```

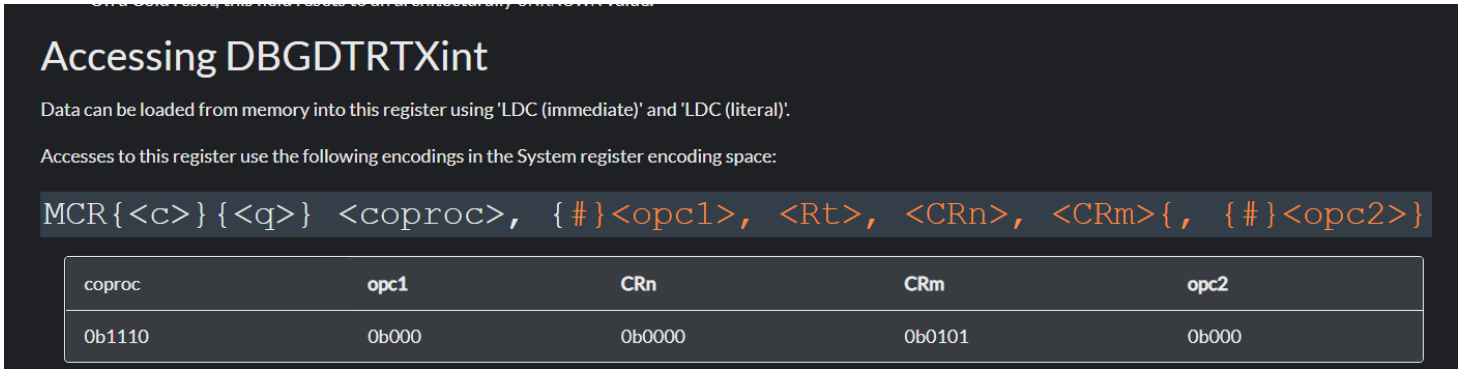
因此相应的代码是

```
execute_ins_via_it(param->debug_register, 0x1f11ee11);
```

Task 5.B: After you finish Task 5.a, you need to transfer the value in R1 on core 0, to the local variable scr. It will be printed later. DBGDTRTXint and DBGDTRTX would be helpful in your implementation.

在5A中，SCR的值已经放入了R1寄存器。将R1寄存器写入DBGDTRTXint的指令是:

```
mcr p14,0,r1,c0,c5,0
```



翻译为机器码的大端存储为 `0xee001e15` ， 因此代码为：

```
execute_ins_via_it(param->debug_register, 0x1e15ee00);
```

最后通过ioread读取到scr中：

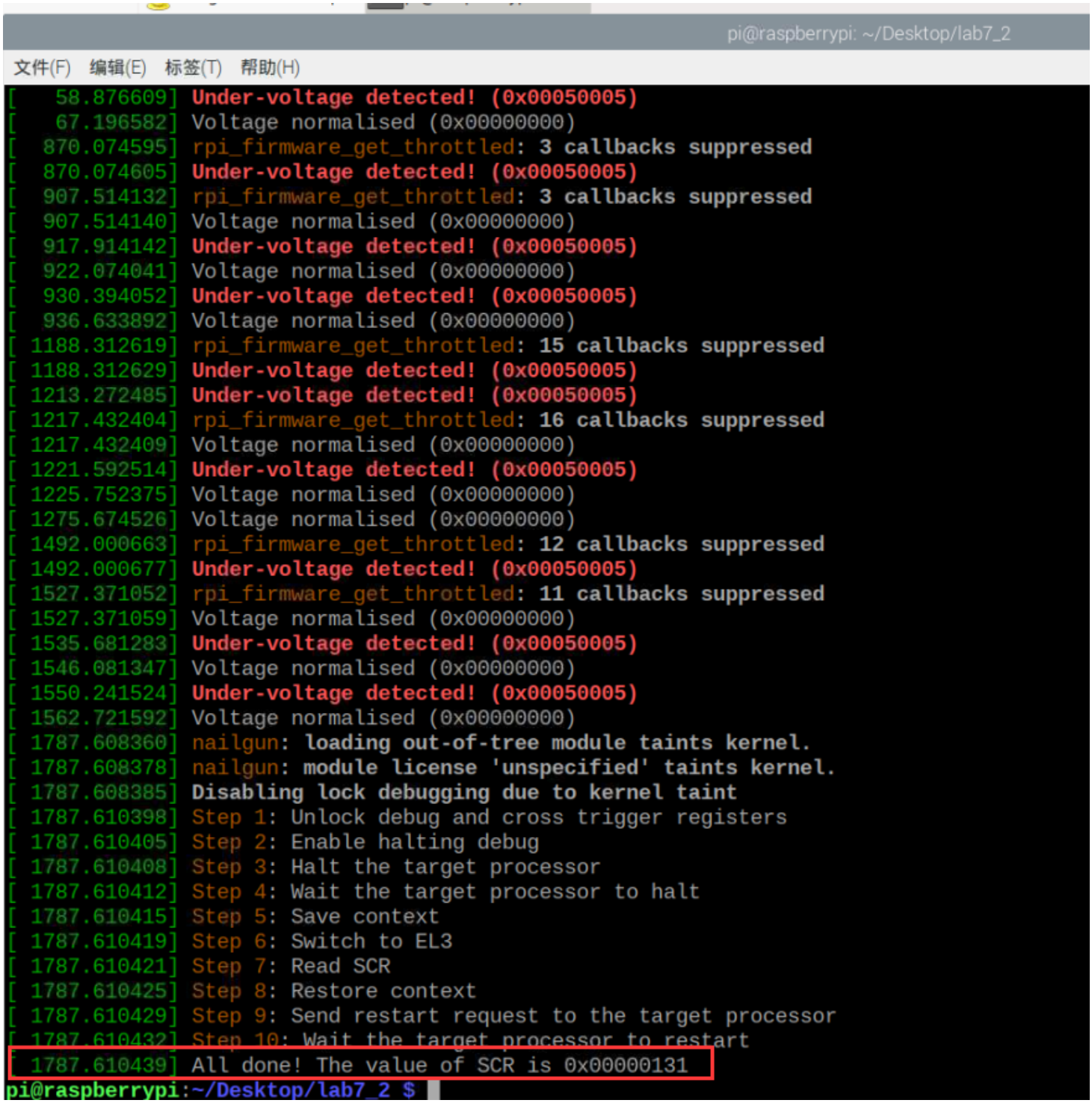
```
scr = ioread32(param->debug_register + DBGDTRX_OFFSET);
```

即完整代码为：

```
execute_ins_via_it(param->debug_register, 0x1f11ee11);
execute_ins_via_it(param->debug_register, 0x1e15ee00);
scr = ioread32(param->debug_register + DBGDTRX_OFFSET);
```

3.7 Task 6: Restore the Context and Exit

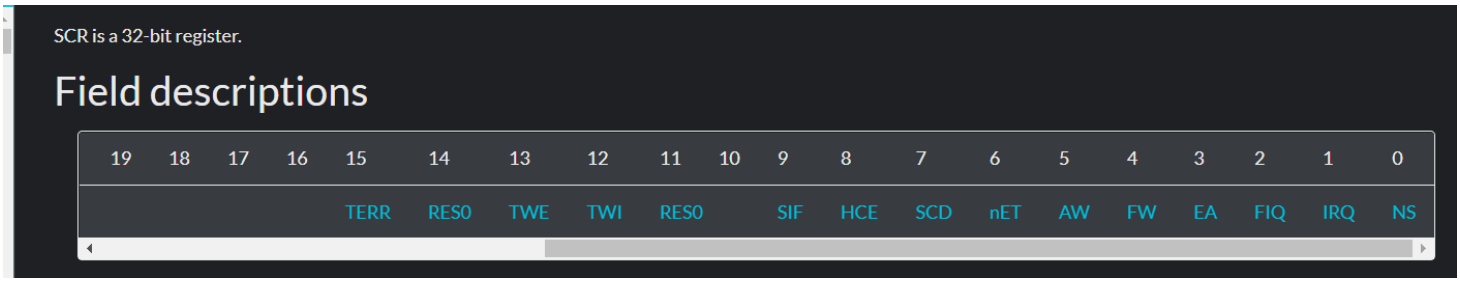
Build and run nailgun.ko, and see what happen. Explain the value of SCR.



执行后，SCR的数值为 `0x00000131` ，转换为二进制为（共32个bit，前缀0均省略）：

```
100110001
```

查看SCR的定义，



参考手册，有如下位被设为1：

- NS: PE处于Non-secure state。
- FW: 从任一安全状态获取的FIQ都被PSTATE.F屏蔽。当PSTATE.F为0时，FIQ进入EL3。
- AW: PSTATE.A会屏蔽从任一安全状态获取的外部中止。当PSTATE.A为0时，abort进入EL3。
- HCE: HVC指令在非安全态的EL1和EL2可用。

4 Submissions

In your report, you should also contain the answer to following questions:

1. During this lab, what is the base address of Cross Trigger Interface in Raspberry Pi 3? Can your find the global address of CTICONTROL register in Raspberry Pi 3 according to the Arm Reference Manual? Answer the address value and show your calculation. (hint: Find the offset)

Base address在代码中有定义，为 `0x40038000`

```
#define CTI_REGISTER_ADDR 0x40038000
```

手册3968页有规定offset为 `0x0`：

Accessing CTICONTROL

CTICONTROL can be accessed through the external debug interface:

Component	Offset	Instance
CTI	0x000	CTICONTROL

因此global address就是 `0x40038000 + 0x0 = 0x40038000`

2. Do we have another way to unlock the OS Lock in this lab except memory mapping? If yes, how to do that? Justify your answer.

这个题目的解法是来源于张睿豪和刘晟淇。也可以使用内联汇编的方法来写入指定的寄存器。具体做法是将3.4中Task4的填空的后两句语句换为下面的语句：

```
// iowrite32(0x0, param->debug_register + OSLAR_OFFSET);
// siowrite32(0x0, param->cti_register + OSLAR_OFFSET);
asm volatile("eor %0, %0, %0":"=r"(reg));
asm volatile("mcr p14, 0, %0, c1, c0, 4":"=r"(reg));
```

In addition, you should also submit your file named as UID_nailgun.c

请见 `11910104_nailgun.c` 。