Lab8

Replace the kernel.

```
lab6@ubuntu:~/Desktop/linux-rpi-4.14.y$ make -j8 ARCH=arm CROSS_COMPILE=tools/arm-bcm2708/gcc-l
inaro-arm-linux-gnueabihf-raspbian-x64/arm-linux-gnueabihf- bcm2709_defconfig
#
# configuration written to .config
#
```

```
lab6@ubuntu:~/Desktop/linux-rpi-4.14.y$ make -j8 ARCH=arm CROSS_COMPILE=tools/
inaro-arm-linux-gnueabihf-raspbian-x64/arm-linux-gnueabihf- bcm2709_defconfig
#
# configuration written to .config
#
lab6@ubuntu:~/Desktop/linux-rpi-4.14.y$ make -j8 ARCH=arm CROSS_COMPILE=tools/
inaro-arm-linux-gnueabihf-raspbian-x64/arm-linux-gnueabihf- menuconfig
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```

```
xinying@ubuntu:~/Desktop/linux$ make -j8 ARCH=arm CROSS_COMPILE=../tools/arm-bcm
2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf- zImage
 dtbs modules
  HOSTCC  scripts/kconfig/conf.o
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf  --silentoldconfig Kconfig
  CHK     include/config/kernel.release
  SYSHDR  arch/arm/include/generated/uapi/asm/unistd-eabi.h
  SYSHDR  arch/arm/include/generated/uapi/asm/unistd-common.h
```

```
xinying@ubuntu:~/Desktop/linux/arch/arm/boot$ ls
bootp  compressed  dts  Image  install.sh  Makefile  zImage
xinying@ubuntu:~/Desktop/linux$ make -j8 ARCH=arm CROSS_COMPILE=../tools/arm-bcm2708/gcc-linaro-arm-lin
ux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf- modules_install INSTALL_MOD_PATH=../modulespath
  INSTALL arch/arm/crypto/aes-arm-bs.ko
  INSTALL arch/arm/crypto/sha1-arm-neon.ko
  INSTALL arch/arm/crypto/aes-arm.ko
```

```
xinying@ubuntu:~/Desktop/linux$ ./scripts/mkknlimg ./arch/arm/boot/zImage /media/xinying/boot/ker
nel7.img
Version: Linux version 4.14.114-v7 (xinying@ubuntu) (gcc version 4.8.3 20140303 (prerelease) (cro
sstool-NG linaro-1.13.1+bzr2650 - Linaro GCC 2014.03)) #1 SMP Sat Nov 13 23:56:00 PST 2021
DT: y
DDT: y
270x: y
283x: y
xinying@ubuntu:~/Desktop/linux$
```

```
xinying@ubuntu:~/Desktop/linux$ cp /media/xinying/boot/kernel7.img /media/xinying/boot/kernel.img

xinying@ubuntu:~/Desktop/linux$ cp ./arch/arm/boot/dts/bcm
Display all 122 possibilities? (y or n)
xinying@ubuntu:~/Desktop/linux$ cp ./arch/arm/boot/dts/bcm2710-rpi-3-b-plus.dtb /media/xinying/bo
ot/
xinying@ubuntu:~/Desktop/linux$ cp ./arch/arm/boot/dts/overlays/*.dtb* /media/xinying/boot/overla
ys/
xinying@ubuntu:~/Desktop/linux$
```

Question 1:(20%) Can you prove that (1) you have replaced the kernel (with ”uname -r” or other approaches), and (2) you have built the nailgun module with new headers? Please provide a figure.

```
pi@raspberrypi:~$ uname -r
4.14.114-v7
pi@raspberrypi:~$
```

```
xinying@ubuntu:~/Desktop/Read_SCR$ make all
make ARCH=arm -C ../modulespath/lib/modules/4.14.114-v7/build M=/home/xinying/Desktop/Read_SCR CROSS_COMPILE=../tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf- modules
make[1]: Entering directory '/home/xinying/Desktop/linux'
  CC [M]  /home/xinying/Desktop/Read_SCR/nailgun.o
/home/xinying/Desktop/Read_SCR/nailgun.c: In function ‘nailgun_init’:
/home/xinying/Desktop/Read_SCR/nailgun.c:222:5: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
     struct nailgun_param *param = kmalloc(sizeof(t_param), GFP_KERNEL);
     ^
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/xinying/Desktop/Read_SCR/nailgun.mod.o
  LD [M]  /home/xinying/Desktop/Read_SCR/nailgun.ko
make[1]: Leaving directory '/home/xinying/Desktop/linux'
xinying@ubuntu:~/Desktop/Read_SCR$ ls
Makefile       Module.symvers  nailgun.ko     nailgun.mod.o
modules.order  nailgun.c       nailgun.mod.c  nailgun.o
```

Question 2:(20%) Can you run the Nailgun Attack on your new kernel? Please provide a figure. You can use ”dmesg” to show the execution result of Nailgun Attack.

```
pi@raspberrypi:~/workspace/Read_SCR$ ls
Makefile       Module.symvers  nailgun.ko     nailgun.mod.o
modules.order  nailgun.c       nailgun.mod.c  nailgun.o
```

```
[  374.516386] Step 6: Switch to EL3
[  374.516391] Step 7: Read SCR
[  374.516395] Step 8: Restore context
[  374.516401] Step 9: Send restart request to the target processor
[  374.516406] Step 10: Wait the target processor to restart
[  374.516414] All done! The value of SCR is 0x00000131
```

The table constructed is as follows:

level 1 table(0x3200000)	mapping address	level 2 table(0x32010000)	mapping address	level 3 table(0x32020000)	mapping address
0x32030003	0x0-0x3fffffff(1 GB)	0x32020003	0x40000000-0x401fffff(2MB) (table)	0x400004cb	
0x0	table	0x0		0x40000000	
0x32010003	0X40000000-0X7fffffff(1 GB)	0x402004c9	(block)	0x40014cb	
0x0	table	0x00400000		0x40000000	
0x800004c9	0x80000000-0xbfffffff(1 GB)	0x404004c9		.....	
0x00400000	blcok	0x00400000		0x0	**0x32020180**(0x40030000-0x40030ffff)
0xc00004c9	0xc0000000-0xffffffff(1 GB)	.....(all 512*2MB)	(.....-0x7fffffff)	0x0	invalid
0x00400000	block	0x32030000	**(0x0-0x3fffffff)**		
		0x0000007ED	(block)		
		0X0			
		0x00200007ED	(block)		
		0X0			
		.....			
		0x0(0x32030c80)	(invalid)		
		0x0			

Question 3:(30%) With the provided source codes, can you explain the process of traslating an IPA, 0x40030000+"last 3 numbers of your student ID", to the same value of PA? (e.g., if your ID is 12150073, then you should translate 0x40030073). In this question, you should mention the (1) address of each descriptor, and (2) value of each descriptor.

It's not the same address.

Input address:**0x40030039**

level 1 address([31:30]of IPA +[31:5] of base): 0x32000000+0x01<<3=**0x32000008**

level 1 entry: 0x0000000032010003(last two bit is 11, indicates it's a table reference)

leve 2 address:([31:12] of level 1 entry+[29:21] input address+000) :**0x32010000**

level 2 entry: **0x0000000032020003**(last two bit is 11, indicates it's a table reference)

level 3 address([31:12] of level 2 entry+[20:12] input address+000) :**0x32020180**

level 3 entry :**0x0** ==> indicate it's invalid, so it should return a fault.

Question 4:(30%) With the provided source codes, can you explain the process of traslating an IPA, 0x40030000+"last 7 numbers of your student ID", to the same value of PA? (e.g., if your ID is 12150073, then you should translate 0x42150073). In this question, you should mention the (1) address of each descriptor, and (2) value of each descriptor.

It's the same address.

Input address:**0x41912039**

level 1 address([31:30]of IPA +[31:5] of base): 0x32000000+0x01<<3=**0x32000008**

level 1 entry: 0x0000000032010003(last two bit is 11, indicates it's a table reference)

leve 2 address:([31:12] of level 1 entry+[29:21] input address+000) :**0x32010060**

level 2 entry: **0x400000418004c9**(last two bit is 01, indicates it's a block)

return address([31:21of the level 2 blcok + [20:0] of the input entry]):0x41912039

Question 5:(20% Bonus) If we have reserved a 4KB memory space (0xa000 0000 0xa000 0fff), and we want to forbid EL1&0 to access it, what can we do? You need to (1) submit the "head.S" file, and (2)explain how the Stage-2 translation work on your translation table. For example, you can use an IPA 0xa0000120, and explain how the access failed.

Example:

Input address: 0xa0000120
level1 address([31:30]of IPA +[31:5] of base):0x32000000+0x10<<3 = **0x320000010**

level 1 entry : 0x0000000032050003 ==>table

leve 2 address:([31:12] of level 1 entry+[29:21] input address+000):0x32050800

level 2 entry: 0x0000000032060003 ==>table

level 3 address([31:12] of level 2 entry+[20:12] input address+000) :**0x32060000**

level 3 entry :**0x0** ==> indicate it's invalid, so it should return a fault.