

CS5250 Assignment Report

Part A: Linux Kernel Installation (20 marks)

1.1Install Virtualbox on your machine

1.2Install a Linux guest machine

1.3Build a specific kernel

1.4 build smallest kernel

Part B: x86 Assembly Programming (10 marks)

1.What is the IA32 (i.e., 32-bit Intel processor) instruction corresponding to these bytes, showing each step of your decoding process?

2. What is the 64-bit instruction (Intel64) encoding for the following instruction. You need to show each step of your workings.

3. Disassemble the following IA32 assembly program by hand and recover the C function that performs the same task.

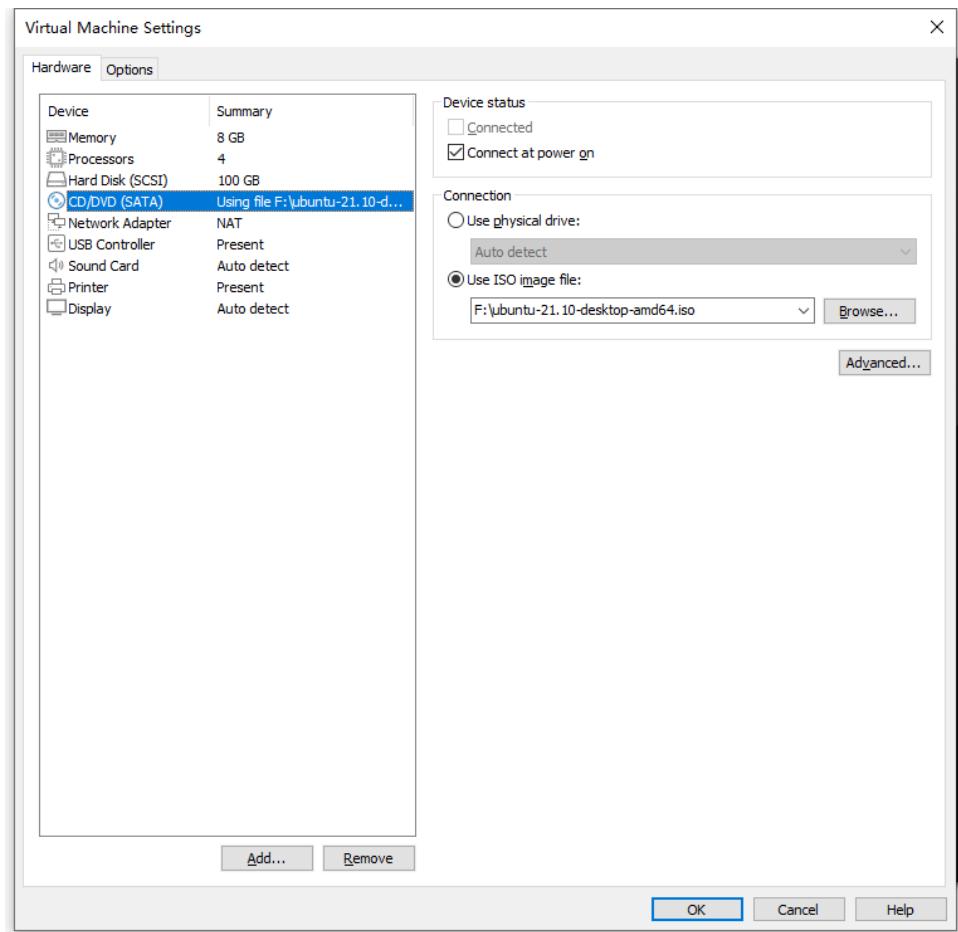
Part A: Linux Kernel Installation (20 marks)

1.1Install Virtualbox on your machine

- Instead of Virtualbox, I choose to use the VMware Workstation Pro 16.

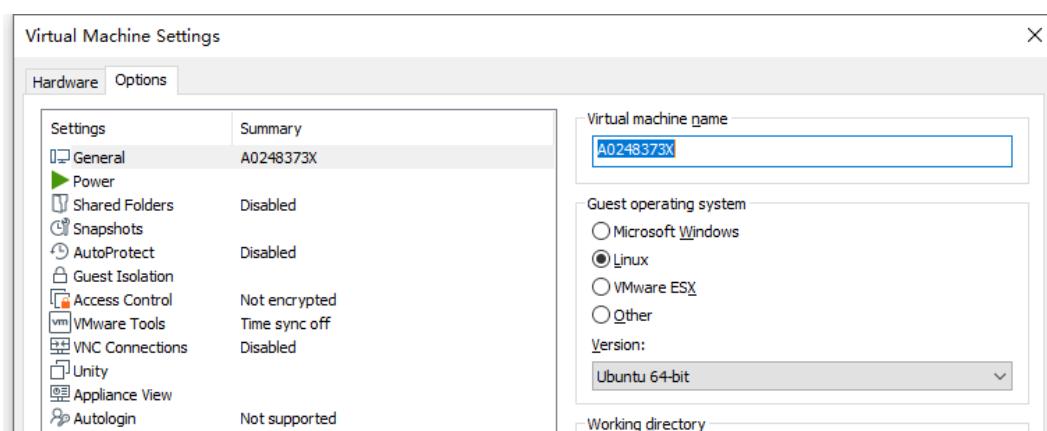
1.2Install a Linux guest machine

1. The version of Linux Ubuntu I use is 21.10 (64bit) version

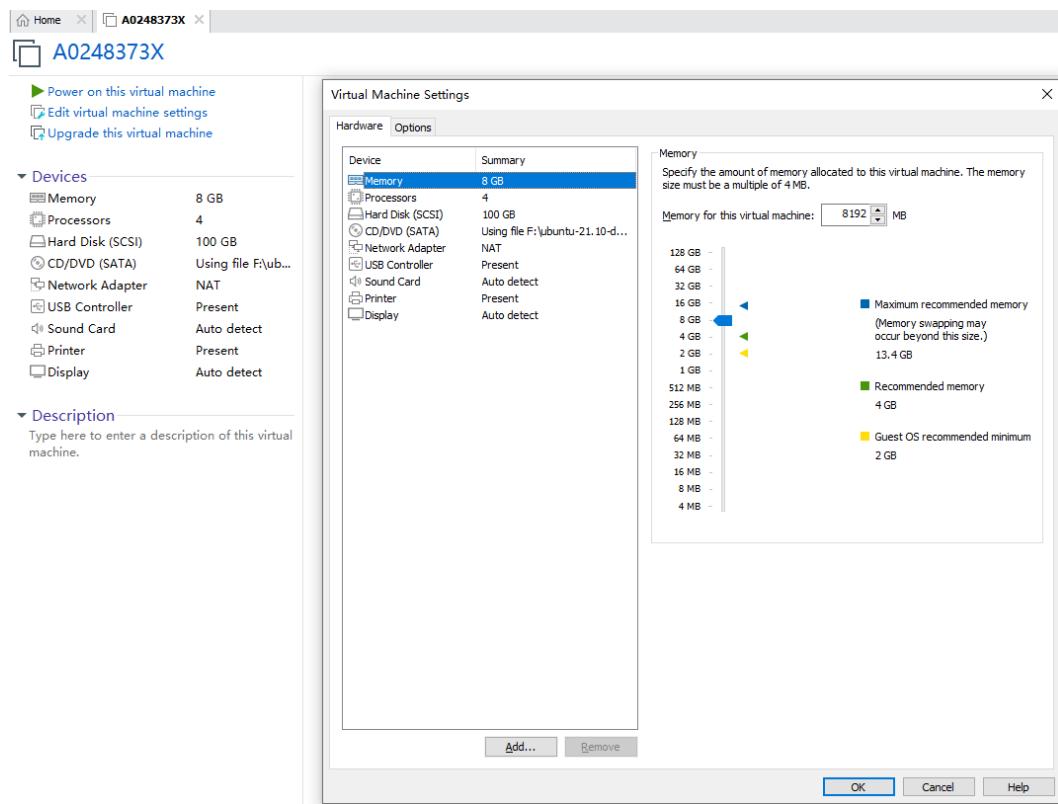


2. Set up a new virtual machine on VMware

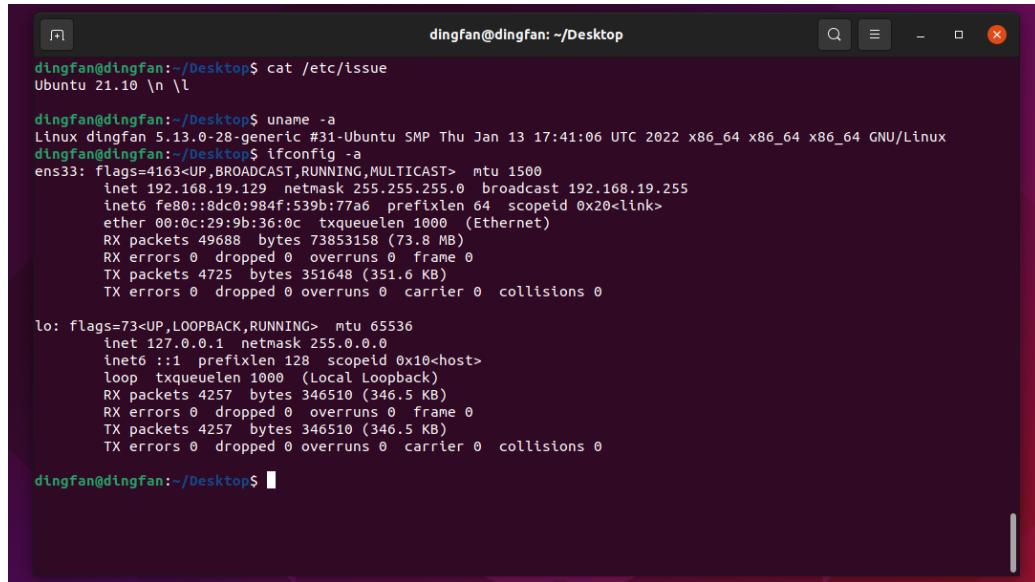
- VM Name and OS Type
 - Name: A0248373X
 - OS Type:
 - a. Operating System: Linux
 - b. Version: Ubuntu 64bit



- Memory : 8096MB
- Virtual Hard Disk: 100GB



3. Start virtual machine, and install the Ubuntu as guided.
 - a. Set username as my full name: dingfan
 - b. *Open the terminal in your guest machine, output the OS, the kernel version and also your MAC address and give a screenshot including all the information.*
 - output OS: `cat /etc/issue` Ubuntu 21.10
 - output kernel version: `uname -a` 5.13.0
 - output MAC address: `ifconfig -a` 00:0c:29:9b:36:0c



```

dingfan@dingfan:~/Desktop$ cat /etc/issue
Ubuntu 21.10 \n \l

dingfan@dingfan:~/Desktop$ uname -a
Linux dingfan 5.13.0-28-generic #31-Ubuntu SMP Thu Jan 13 17:41:06 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
dingfan@dingfan:~/Desktop$ ifconfig -a
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.19.129 netmask 255.255.255.0 broadcast 192.168.19.255
        inet6 fe80::8dc0:984f:539b:77a6 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:9b:36:0c txqueuelen 1000 (Ethernet)
                RX packets 49688 bytes 73853158 (73.8 MB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 4725 bytes 351648 (351.6 KB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

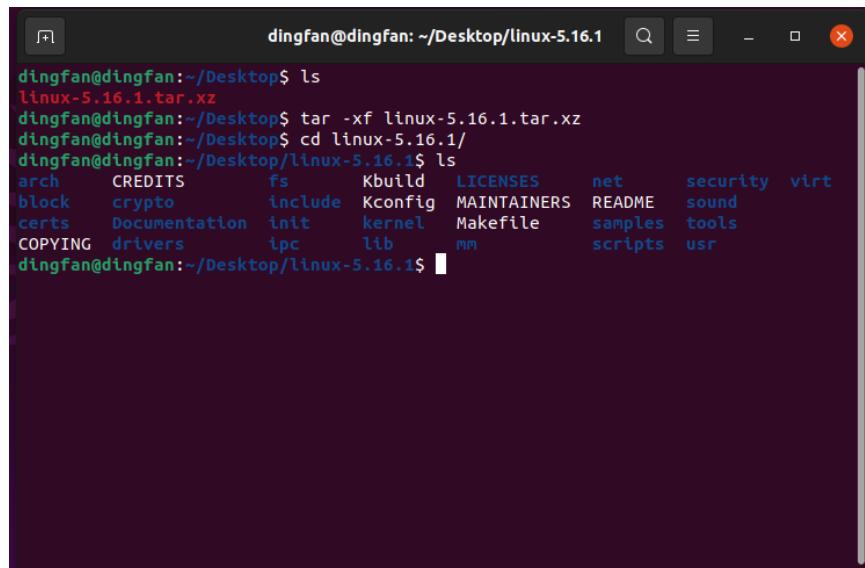
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
                RX packets 4257 bytes 346510 (346.5 KB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 4257 bytes 346510 (346.5 KB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

dingfan@dingfan:~/Desktop$ 

```

1.3 Build a specific kernel

1. Download `linux-5.16.1.tar.xz` from (<https://cdn.kernel.org/pub/linux/kernel/v5.x/>). Unzip the file using the `tar -xf` command, and enter the directory in terminal using `cd` command.



```

dingfan@dingfan:~/Desktop$ ls
linux-5.16.1.tar.xz
dingfan@dingfan:~/Desktop$ tar -xf linux-5.16.1.tar.xz
dingfan@dingfan:~/Desktop$ cd linux-5.16.1/
dingfan@dingfan:~/Desktop/linux-5.16.1$ ls
arch      CREDITS      fs          Kbuild      LICENSES      net          security      virt
block     crypto       include     Kconfig     MAINTAINERS  README      sound
certs     Documentation init      kernel     Makefile     samples     tools
COPYING   drivers      ipc        lib        mm          scripts    usr
dingfan@dingfan:~/Desktop/linux-5.16.1$ 

```

2. When try command `make menuconfig` I see some linux errors. After Google the problem, I figure it out that there are no specific header in my machine. The following is the problems and the related solutions.

- a. run `sudo apt-get install build-essential` to install `gcc` package

```
dingfan@dingfan:~/Desktop/linux-5.16.1$ make menuconfig
Command 'make' not found, but can be installed with:
sudo apt install make      # version 4.3-4ubuntu1, or
sudo apt install make-guile # version 4.3-4ubuntu1
dingfan@dingfan:~/Desktop/linux-5.16.1$ sudo apt-get install build-essential
```

- b. run `sudo apt-get install libncurses-dev` to install `ncurses` package

```
dingfan@dingfan:~/Desktop/linux-5.16.1$ make menuconfig
HOSTCC scripts/basic/fixdep
*
* Unable to find the ncurses package.
* Install ncurses (ncurses-devel or libncurses-dev
* depending on your distribution).
*
* You may also need to install pkg-config to find the
* ncurses installed in a non-default location.
*
make[1]: *** [scripts/kconfig/Makefile:211: scripts/kconfig/mconf-cfg] Error 1
make: *** [Makefile:619: menuconfig] Error 2
dingfan@dingfan:~/Desktop/linux-5.16.1$ sudo apt-get install libncurses-dev
```

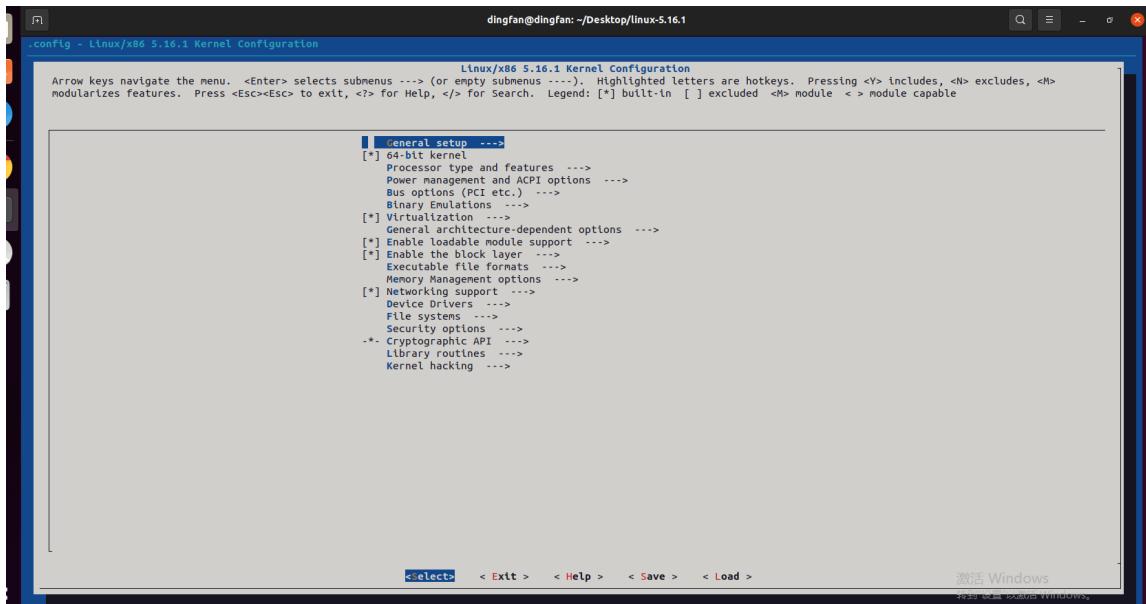
- c. run `sudo apt-get install flex` to install `flex` package

```
dingfan@dingfan:~/Desktop/linux-5.16.1$ make menuconfig
UPD    scripts/kconfig/mconf-cfg
HOSTCC scripts/kconfig/mconf.o
HOSTCC scripts/kconfig/lxdialog/checklist.o
HOSTCC scripts/kconfig/lxdialog/inputbox.o
HOSTCC scripts/kconfig/lxdialog/menubox.o
HOSTCC scripts/kconfig/lxdialog/textbox.o
HOSTCC scripts/kconfig/lxdialog/util.o
HOSTCC scripts/kconfig/lxdialog/yesno.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX    scripts/kconfig/lexer.lex.c
/bin/sh: 1: flex: not found
make[1]: *** [scripts/Makefile.host:9: scripts/kconfig/lexer.lex.c] Error 127
make: *** [Makefile:619: menuconfig] Error 2
dingfan@dingfan:~/Desktop/linux-5.16.1$ sudo apt-get install flex
```

- d. run `sudo apt-get install bison` to install `bison` package

```
dingfan@dingfan:~/Desktop/linux-5.16.1$ make menuconfig
LEX    scripts/kconfig/lexer.lex.c
YACC   scripts/kconfig/parser.tab.[ch]
/bin/sh: 1: bison: not found
make[1]: *** [scripts/Makefile.host:17: scripts/kconfig/parser.tab.h] Error 127
make: *** [Makefile:619: menuconfig] Error 2
dingfan@dingfan:~/Desktop/linux-5.16.1$ sudo apt-get install bison
Reading package lists... Done
```

3. When successfully run command `make menuconfig`, I can see the kernel configuration menu as follow:



During the “make menuconfig” stage, there are three different choices, built-in, excluded

and module. (i) What do they mean? (ii) Which are the ones that will appear in the kernel image?

i. (i) What do they mean?

- By google, I know that this `menuconfig` is used to generate a “.config ” file. This “.config ” file is file is a configuration file for compiling the linux kernel.
- the choice `[*] built in` means that, the feature will be compiled into the new linux kernel “vmlinuz”
- the choice `[] excluded` means that the feature will not be compiled
- the choice `<M> module` means that the feature will be compiled as a module, it can be dynamically loaded into the memory when needed

ii. Which are the ones that will appear in the kernel image?

- the features that with `[*] built in` choice will appear in the kernel image
- the features that with `<M> module` choice are not in kernel image, they are in the source directory of each module.

4. run `make` or `sudo make -j8` to compile the kernel

- run `sudo make -j8` meet some errors, and I resolve these problems by google. The problems and solutions are:

- run `apt-get install libdw-dev` to install the `libelf-dev`

```
DESCEND bpf/resolve_btfids
YACC    scripts/genksyms/parse.tab.[ch]
MKDIR   /home/dingfan/Desktop/linux-5.16.1/tools/bpf/resolve_btfids/libbpf/
<stdin>:1:10: fatal error: libelf.h: No such file or directory
compilation terminated.
HOSTCC  /home/dingfan/Desktop/linux-5.16.1/tools/objtool/fixdep.o
HOSTCC  /home/dingfan/Desktop/linux-5.16.1/tools/bpf/resolve_btfids/fixdep.o
```

```
Trash : included from /home/dingfan/Desktop/linux-5.16.1/tools/objtool/include/objtool/objtool.h:15,
from weak.c:10:
/home/dingfan/Desktop/linux-5.16.1/tools/objtool/include/objtool/elf.h:10:10: fatal error: gelf.h: No such file or directory
10 | #include <gelf.h>
 |           ^
compilation terminated.
make[3]: *** [/home/dingfan/Desktop/linux-5.16.1/tools/build/Makefile.build:97: /home/dingfan/Desktop/linux-5.16.1/tools/objtool/weak.o] Error 1
```

```
dingfan@dingfan:~/Desktop/linux-5.16.1$ sudo apt-get install libdw-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required.
```

- run `sudo apt-get install libssl-dev` to install the `openssl`

```
HOSTCC  scripts/sign-file
scripts/sign-file.c:25:10: fatal error: openssl/opensslv.h: No such file or directory
25 | #include <openssl/opensslv.h>
 |           ^
compilation terminated.
make[1]: *** [scripts/Makefile.host:95: scripts/sign-file] Error 1
```

```
dingfan@dingfan:~/Desktop/linux-5.16.1$ sudo apt-get install libssl-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

- After google, I know the following bug can be resolved by modified the “.config” file:

```
AS      arch/x86/entry/entry_64.o
AR      init/built-in.a
make[1]: *** No rule to make target 'debian/canonical-certs.pem', needed by 'certs/x509_certificate_list'. Stop.
make[1]: *** Waiting for unfinished jobs....
CC      certs/system_keyring.o
```

- set `CONFIG_SYSTEM_TRUSTED_KEYS = ""`
- set `CONFIG_SYSTEM_REVOCATION_KEYS = ""`
- before modify

```

Activities Text Editor Feb 4 00:05
Open .config ~/Desktop/linux-5.16.1 Save
10815 CONFIG_SIGNED_PE_FILE_VERIFICATION=y
10817
10819 # Certificates for signature checking
10820 #
10821 CONFIG_MODULE_SIG_KEY='certs/signing_key.pem'
10822 CONFIG_MODULE_SIG_KEY_TYPE_RSA=y
10823 CONFIG_MODULE_SIG_KEY_TYPE_ECDSA is not set
10824 CONFIG_SYSTEM_TRUSTED_KEYRING=y
10825 CONFIG_SYSTEM_TRUSTED_KEYS='debian/canonical-certs.pem'
10826 CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
10827 CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
10828 CONFIG_SECONDARY_TRUSTED_KEYRING=y
10829 CONFIG_SYSTEM_BLACKLIST_KEYRING=y
10830 CONFIG_SYSTEM_BLACKLIST_HASH_LIST=''
10831 CONFIG_SYSTEM_REVOCATION_LIST=y
10832 CONFIG_SYSTEM_REVOCATION_KEYS='debian/canonical-revoked-certs.pem'
10833 # end of Certificates for signature checking
10834
10835 CONFIG_BINARY_PRINT=y
10836
10837 #
10838 # Library routines
10839
10840 CONFIG_RAID6_PQ=m
10841 CONFIG_RAID6_PQ_BENCHMARK=y
10842 CONFIG_LINEAR_RANGES=y
10843 CONFIG_PACKING=y
10844 CONFIG_BTREVERSE=y
10845 CONFIG_GENERIC_STRNCPY_FROM_USER=y
10846 CONFIG_GENERIC_STRLEN_USER=y
10847 CONFIG_GENERIC_NET_UTILS=y
10848 CONFIG_CRC32_FTABLE_FIRST_BIT=y
10849 CONFIG_CRC32=y
10850 # CONFIG_PRIME_NUMBERS is not set
10851 CONFIG_RATIONAL=y
10852 CONFIG_GENERIC_PCI_IOMAP=y

```

激活 Windows
转到“设置”以激活 Windows
.config selected (261.8 kB)

- after modify

```

Activities Text Editor Feb 4 00:07
Open .config ~/Desktop/linux-5.16.1 Save
10811 CONFIG_XC99_CERTIFICATE_PARSER=y
10812 CONFIG_PKCS5_PRIVATE_KEY_PARSER=n
10813 CONFIG_TPM_KEY_PARSER=m
10814 CONFIG_PKCS7_MESSAGE_PARSER=y
10815 CONFIG_PKCS7_TEST_KEY=n
10816 CONFIG_SIGNED_PE_FILE_VERIFICATION=y
10817
10818 #
10819 # Certificates for signature checking
10820 #
10821 CONFIG_MODULE_SIG_KEY='certs/signing_key.pen'
10822 CONFIG_MODULE_SIG_KEY_TYPE_RSA=y
10823 # CONFIG_MODULE_SIG_KEY_TYPE_ECDSA is not set
10824 CONFIG_SYSTEM_TRUSTED_KEYRING=y
10825 CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
10826 CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
10827 CONFIG_SECONDARY_TRUSTED_KEYRING=y
10828 CONFIG_SYSTEM_BLACKLIST_KEYRING=y
10829 CONFIG_SYSTEM_BLACKLIST_HASH_LIST=''
10830 CONFIG_SYSTEM_REVOCATION_LIST=y
10831 CONFIG_SYSTEM_REVOCATION_KEYS=''
10832 # end of Certificates for signature checking
10833
10834
10835 CONFIG_BINARY_PRINT=y
10836
10837 #
10838 # Library routines
10839
10840 CONFIG_RAID6_PQ=m
10841 CONFIG_RAID6_PQ_BENCHMARK=y
10842 CONFIG_LINEAR_RANGES=y
10843 CONFIG_PACKING=y
10844 CONFIG_BTREVERSE=y
10845 CONFIG_GENERIC_STRNCPY_FROM_USER=y
10846 CONFIG_GENERIC_STRLEN_USER=y
10847 CONFIG_GENERIC_NET_UTILS=y

```

激活 Windows
转到“设置”以激活 Windows
.config selected (261.7 kB)

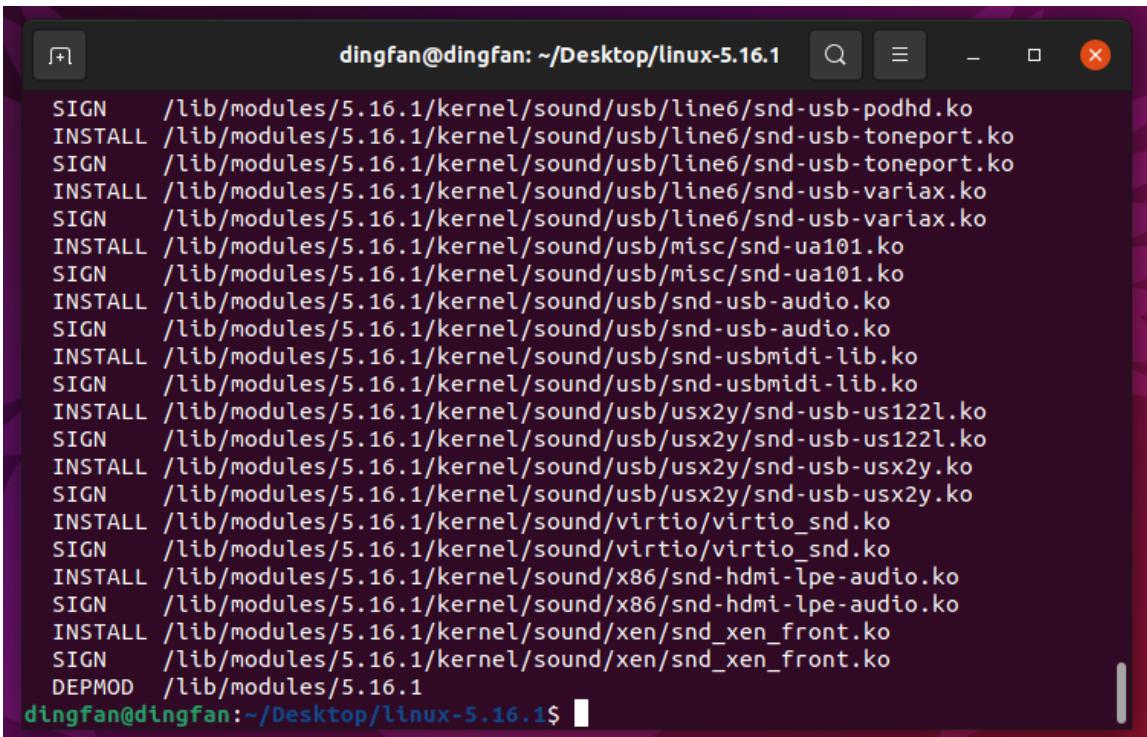
d. run `sudo apt-get install dwarves` to install `dwarves`

```
CC [M] drivers/iio/industrialio-sw-device.o
CC [M] drivers/iio/industrialio-sw-trigger.o
CC [M] drivers/iio/light/tsl4531.o
CC [M] drivers/iio/light/us5182d.o
CC [M] drivers/iio/industrialio-triggered-event.o
CC [M] drivers/iio/light/vcnl4000.o
CC [M] drivers/iio/light/vcnl4035.o
LD [M] drivers/iio/industrialio.o
CC [M] drivers/iio/light/veml6030.o
CC [M] drivers/iio/light/veml6070.o
CC [M] drivers/iio/light/vl6180.o
CC [M] drivers/iio/light/zopt2201.o
GEN .version
CHK include/generated/compile.h
LD vmlinux.o
MODPOST vmlinux.symvers
MODINFO modules.builtin.modinfo
GEN modules.builtin
BTF: .tmp_vmlinux.btf: pahole (pahole) is not available
Failed to generate BTF for vmlinux
Try to disable CONFIG_DEBUG_INFO_BTF
make: *** [Makefile:1161: vmlinux] Error 1
dingfan@dingfan:~/Desktop/linux-5.16.1$
```

e. run `sudo make -j8` successfully

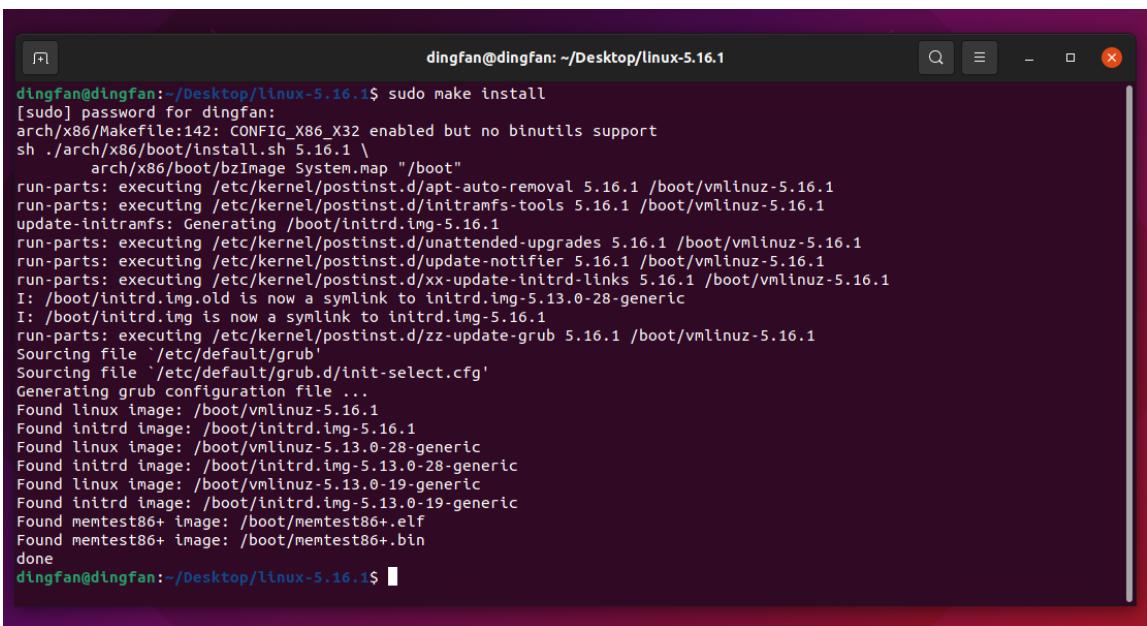
```
LD [M] sound/soc/xilinx/snd-soc-xlnx-spdif.ko
LD [M] sound/soc/xtensa/snd-soc-xtfpga-i2s.ko
LD [M] sound/soundcore.ko
LD [M] sound/synth/emux/snd-emux-synth.ko
LD [M] sound/synth/snd-util-mem.ko
LD [M] sound/usb/6fire/snd-usb-6fire.ko
LD [M] sound/usb/bcd2000/snd-bcd2000.ko
LD [M] sound/usb/caiaq/snd-usb-caiaq.ko
LD [M] sound/usb/hiface/snd-usb-hiface.ko
LD [M] sound/usb/line6/snd-usb-line6.ko
LD [M] sound/usb/line6/snd-usb-pod.ko
LD [M] sound/usb/line6/snd-usb-podhd.ko
LD [M] sound/usb/line6/snd-usb-toneport.ko
LD [M] sound/usb/line6/snd-usb-variax.ko
LD [M] sound/usb/misc/snd-ua101.ko
LD [M] sound/usb/snd-usb-audio.ko
LD [M] sound/usb/snd-usbmidi-lib.ko
LD [M] sound/usb/usx2y/snd-usb-us122l.ko
LD [M] sound/usb/usx2y/snd-usb-usx2y.ko
LD [M] sound/virtio/virtio_snd.ko
LD [M] sound/x86/snd-hdmi-lpe-audio.ko
LD [M] sound/xen/snd_xen_front.ko
dingfan@dingfan:~/Desktop/linux-5.16.1$
```

5. run `make modules_install`



```
dingfan@dingfan: ~/Desktop/linux-5.16.1$ make modules
SIGN    /lib/modules/5.16.1/kernel/sound/usb/line6/snd-usb-podhd.ko
INSTALL /lib/modules/5.16.1/kernel/sound/usb/line6/snd-usb-toneport.ko
SIGN    /lib/modules/5.16.1/kernel/sound/usb/line6/snd-usb-toneport.ko
INSTALL /lib/modules/5.16.1/kernel/sound/usb/line6/snd-usb-variax.ko
SIGN    /lib/modules/5.16.1/kernel/sound/usb/line6/snd-usb-variax.ko
INSTALL /lib/modules/5.16.1/kernel/sound/usb/misc/snd-ua101.ko
SIGN    /lib/modules/5.16.1/kernel/sound/usb/misc/snd-ua101.ko
INSTALL /lib/modules/5.16.1/kernel/sound/usb/snd-usb-audio.ko
SIGN    /lib/modules/5.16.1/kernel/sound/usb/snd-usb-audio.ko
INSTALL /lib/modules/5.16.1/kernel/sound/usb/snd-usbmidi-lib.ko
SIGN    /lib/modules/5.16.1/kernel/sound/usb/snd-usbmidi-lib.ko
INSTALL /lib/modules/5.16.1/kernel/sound/usb/usx2y/snd-usb-usx2l.ko
SIGN    /lib/modules/5.16.1/kernel/sound/usb/usx2y/snd-usb-usx2l.ko
INSTALL /lib/modules/5.16.1/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
SIGN    /lib/modules/5.16.1/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL /lib/modules/5.16.1/kernel/sound/virtio/virtio_snd.ko
SIGN    /lib/modules/5.16.1/kernel/sound/virtio/virtio_snd.ko
INSTALL /lib/modules/5.16.1/kernel/sound/x86/snd-hdmi-lpe-audio.ko
SIGN    /lib/modules/5.16.1/kernel/sound/x86/snd-hdmi-lpe-audio.ko
INSTALL /lib/modules/5.16.1/kernel/sound/xen/snd_xen_front.ko
SIGN    /lib/modules/5.16.1/kernel/sound/xen/snd_xen_front.ko
DEPMOD  /lib/modules/5.16.1
dingfan@dingfan:~/Desktop/linux-5.16.1$
```

6. run `sudo make install`



```
dingfan@dingfan: ~/Desktop/linux-5.16.1$ sudo make install
[sudo] password for dingfan:
arch/x86/Makefile:142: CONFIG_X86_X32 enabled but no binutils support
sh ./arch/x86/boot/install.sh 5.16.1 \
    arch/x86/boot/bzImage System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 5.16.1 /boot/vmlinuz-5.16.1
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.16.1 /boot/vmlinuz-5.16.1
update-initramfs: Generating /boot/initrd.img-5.16.1
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.16.1 /boot/vmlinuz-5.16.1
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.16.1 /boot/vmlinuz-5.16.1
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.16.1 /boot/vmlinuz-5.16.1
I: /boot/initrd.img.old is now a symlink to initrd.img-5.13.0-28-generic
I: /boot/initrd.img is now a symlink to initrd.img-5.16.1
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.16.1 /boot/vmlinuz-5.16.1
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.16.1
Found initrd image: /boot/initrd.img-5.16.1
Found linux image: /boot/vmlinuz-5.13.0-28-generic
Found initrd image: /boot/initrd.img-5.13.0-28-generic
Found linux image: /boot/vmlinuz-5.13.0-19-generic
Found initrd image: /boot/initrd.img-5.13.0-19-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
dingfan@dingfan:~/Desktop/linux-5.16.1$
```

- we can `cd` the `/boot/` directory to check whether the new kernel is in the right place
- before run command `sudo make install`, I cannot see the new kernel

```
dingfan@dingfan: /boot/grub
SIGN    /lib/modules/5.16.1/kernel/sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL /lib/modules/5.16.1/kernel/sound/virtio/virtio_snd.ko
SIGN    /lib/modules/5.16.1/kernel/sound/virtio/virtio_snd.ko
INSTALL /lib/modules/5.16.1/kernel/sound/x86/snd-hdmi-lpe-audio.ko
SIGN    /lib/modules/5.16.1/kernel/sound/x86/snd-hdmi-lpe-audio.ko
INSTALL /lib/modules/5.16.1/kernel/sound/xen/snd_xen_front.ko
SIGN    /lib/modules/5.16.1/kernel/sound/xen/snd_xen_front.ko
DEPMOD  /lib/modules/5.16.1
dingfan@dingfan:~/Desktop/linux-5.16.1$ cd /boot/
dingfan@dingfan:/boot$ ls
config-5.13.0-19-generic      memtest86+.elf
config-5.13.0-28-generic      memtest86+_multiboot.bin
efi                           System.map-5.13.0-19-generic
grub                          System.map-5.13.0-28-generic
initrd.img                     vmlinuz
initrd.img-5.13.0-19-generic  vmlinuz-5.13.0-19-generic
initrd.img-5.13.0-28-generic  vmlinuz-5.13.0-28-generic
initrd.img.old                 vmlinuz.old
memtest86+.bin
```

- after run command `sudo make install`, I can see the new kernel with version “5.16.1”

```
dingfan@dingfan:/boot$ ls
config-5.13.0-19-generic      memtest86+.elf
config-5.13.0-28-generic      memtest86+_multiboot.bin
config-5.16.1                 System.map-5.13.0-19-generic
efi                           System.map-5.13.0-28-generic
grub                          System.map-5.16.1
initrd.img                     vmlinuz
initrd.img-5.13.0-19-generic  vmlinuz-5.13.0-19-generic
initrd.img-5.13.0-28-generic  vmlinuz-5.13.0-28-generic
initrd.img-5.16.1              vmlinuz-5.16.1
initrd.img.old                 vmlinuz.old
memtest86+.bin
dingfan@dingfan:/boot$
```

7. Set up bootloader

- The order of OS booting is in `/boot/grub/grub.cfg`
- open this file we can see, it seems that the `sudo make install` has helped me set the “5.16.1” kernel as the default option

```
Activities E File Edit View Win Help || + - X grubcfg  
Open ... grubcfg  
Save ...  
load video  
gfmode $linux_gfx_mode  
157 insmod gzio  
158 if [ $xen ]; then insmod xzio; insmod lzopio; fi  
159 insmod part_gpt  
160 insmod ext2  
161 insmod ext4  
162 if [ $xen ]; then  
163     search --no-floppy --fs-uuid --setroot --hint-bios=hda0,gpt3 --hint-efi=hda0,gpt3 --hint-baremetal=ahci0,gpt3 598bedc0-06ac-4f23-8ce1-6c0395447025  
164 else  
165     search --no-floppy --fs-uuid --setroot 598bedc0-06ac-4f23-8ce1-6c0395447025  
166 fi  
167 Linux /boot/vmlinuz-5.16.1 root=UUID=598bedc0-06ac-4f23-8ce1-6c0395447025 ro quiet splash $vt_handoff  
168 initrd /boot/initrd.img-5.16.1  
169 }  
170 submenut 'Advanced options for Ubuntu' $menuentry_id_option 'gnulinux-advanced-598bedc0-06ac-4f23-8ce1-6c0395447025' {  
171     menuprintf 'Ubuntu' with Linux 5.16.1 --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux_5.16.1-advanced-598bedc0-06ac-4f23-8ce1-6c0395447025' {  
172         submenu 'Ubuntu' $menuentry_id_option 'gnulinux_5.16.1-advanced-598bedc0-06ac-4f23-8ce1-6c0395447025' {  
173             load video  
174             gfmode $linux_gfx_mode  
175             insmod gzio  
176             if [ $xen ]; then insmod xzio; insmod lzopio; fi  
177             insmod part_gpt  
178             insmod ext2  
179             set root=hda0,gpt3  
180             if [ $xen ]; then  
181                 search --no-floppy --fs-uuid --setroot --hint-bios=hda0,gpt3 --hint-efi=hda0,gpt3 --hint-baremetal=ahci0,gpt3 598bedc0-06ac-4f23-8ce1-6c0395447025  
182             else  
183                 search --no-floppy --fs-uuid --setroot 598bedc0-06ac-4f23-8ce1-6c0395447025  
184             fi  
185             echo 'Loading Linux 5.16.1...'  
186             linux /boot/vmlinuz-5.16.1 root=UUID=598bedc0-06ac-4f23-8ce1-6c0395447025 ro quiet splash $vt_handoff  
187             echo 'Loading initial ramdisk ...'  
188             initrd /boot/initrd.img-5.16.1  
189 }  
190 menuprintf 'Ubuntu, with Linux 5.16.1 (recovery mode)' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux_5.16.1-recovery-598bedc0-06ac-4f23-8ce1-6c0395447025' {  
191     recordfail  
192     load_video  
193     insmod gzio  
194     if [ $xgrub ]; then insmod xzio; insmod lzopio; fi  
195     insmod part_gpt  
196     insmod ext2  
197     set root=hda0,gpt3  
198     if [ $xen ]; then  
199         search --no-floppy --fs-uuid --setroot --hint-bios=hda0,gpt3 --hint-efi=hda0,gpt3 --hint-baremetal=ahci0,gpt3 598bedc0-06ac-4f23-8ce1-6c0395447025  
200     else  
201         search --no-floppy --fs-uuid --setroot 598bedc0-06ac-4f23-8ce1-6c0395447025  
202     fi  
203     echo 'Loading Linux 5.16.1...'  
204     linux /boot/vmlinuz-5.16.1 root=UUID=598bedc0-06ac-4f23-8ce1-6c0395447025 ro recovery nomodeset dis_unicode_ldr  
205     echo 'Loading initial ramdisk ...'  
206     initrd /boot/initrd.img-5.16.1  
207 }
```

- I also modify the file `/etc/default/grub`
 - set the `GRUB_TIMEOUT_STYLE= hidden` to `#GRUB_TIMEOUT_STYLE= hidden`. Therefore, when the OS reboot, it will show the grub menu and I can choose the specific kernel manually to start up.
 - set `GRUB_TIMEOUT=30`, the machine will stay in the grub menu for 30 seconds.
 - the following picture is the modified version of `/etc/default/grub`

```
dingfan@dingfan:~/Desktop$ sudo cat /etc/default/grub
[sudo] password for dingfan:
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
#GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=30
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command 'vbeinfo'
#GRUB_GFXMODE=640x480

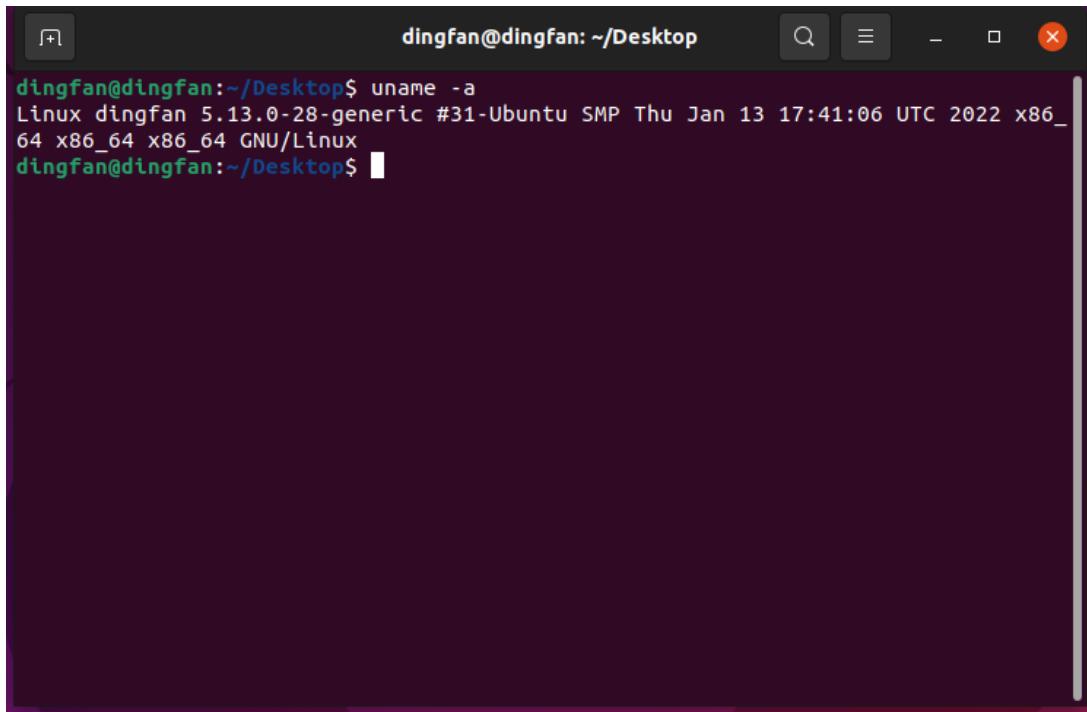
# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"
dingfan@dingfan:~/Desktop$
```

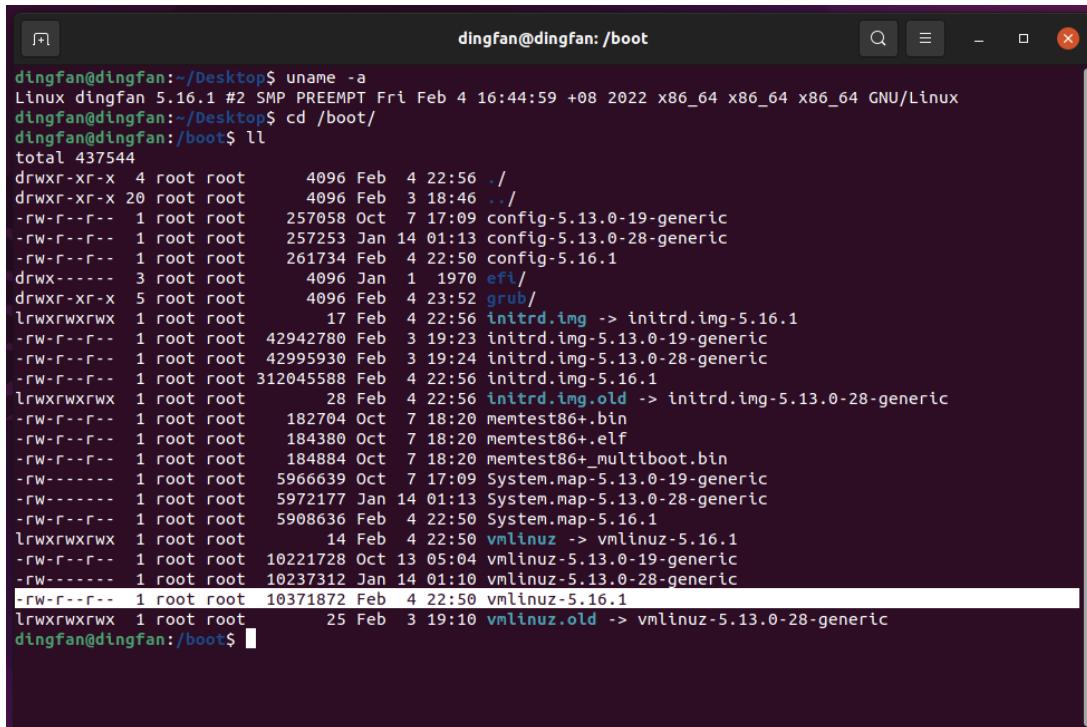
8. Reboot the virtual machine

a. before reboot my kernel version is 5.13.0



```
dingfan@dingfan:~/Desktop$ uname -a
Linux dingfan 5.13.0-28-generic #31-Ubuntu SMP Thu Jan 13 17:41:06 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
dingfan@dingfan:~/Desktop$
```

b. after reboot my kernel version is 5.16.1



```
dingfan@dingfan:~/Desktop$ uname -a
Linux dingfan 5.16.1 #2 SMP PREEMPT Fri Feb 4 16:44:59 +08 2022 x86_64 x86_64 x86_64 GNU/Linux
dingfan@dingfan:~/Desktop$ cd /boot/
dingfan@dingfan:/boot$ ll
total 437544
drwxr-xr-x  4 root root      4096 Feb  4 22:56 .
drwxr-xr-x 20 root root      4096 Feb  3 18:46 ..
-rw-r--r--  1 root root   257058 Oct  7 17:09 config-5.13.0-19-generic
-rw-r--r--  1 root root  257253 Jan 14 01:13 config-5.13.0-28-generic
-rw-r--r--  1 root root  261734 Feb  4 22:50 config-5.16.1
drwxr-xr-x  3 root root     4096 Jan  1 1970 efi/
drwxr-xr-x  5 root root     4096 Feb  4 23:52 grub/
lrwxrwxrwx  1 root root      17 Feb  4 22:56 initrd.img -> initrd.img-5.16.1
-rw-r--r--  1 root root 42942780 Feb  3 19:23 initrd.img-5.13.0-19-generic
-rw-r--r--  1 root root 42995930 Feb  3 19:24 initrd.img-5.13.0-28-generic
-rw-r--r--  1 root root 312045588 Feb  4 22:56 initrd.img-5.16.1
lrwxrwxrwx  1 root root      28 Feb  4 22:56 initrd.img.old -> initrd.img-5.13.0-28-generic
-rw-r--r--  1 root root 182704 Oct  7 18:20 memtest86+.bin
-rw-r--r--  1 root root 184380 Oct  7 18:20 memtest86+.elf
-rw-r--r--  1 root root 184884 Oct  7 18:20 memtest86+_multiboot.bin
-rw-----  1 root root 5966639 Oct  7 17:09 System.map-5.13.0-19-generic
-rw-----  1 root root 5972177 Jan 14 01:13 System.map-5.13.0-28-generic
-rw-r--r--  1 root root 5908636 Feb  4 22:50 System.map-5.16.1
lrwxrwxrwx  1 root root      14 Feb  4 22:50 vmlinuz -> vmlinuz-5.16.1
-rw-r--r--  1 root root 10221728 Oct 13 05:04 vmlinuz-5.13.0-19-generic
-rw-r--r--  1 root root 10237312 Jan 14 01:10 vmlinuz-5.13.0-28-generic
-rw-r--r--  1 root root 10371872 Feb  4 22:50 vmlinuz-5.16.1
lrwxrwxrwx  1 root root      25 Feb  3 19:10 vmlinuz.old -> vmlinuz-5.13.0-28-generic
dingfan@dingfan:/boot$
```

1.4 build smallest kernel

1. In this part, I try to excluded some features base on the default configuration of version “5.16.1”.
2. In this process, I have encountered failures several times. Because I excluded the important features sometimes, and the kernel cannot boot up successfully.
3. One of the failure is that I excluded many features in the “networking support”, that the system cannot reboot successfully as the following screenshot:

```

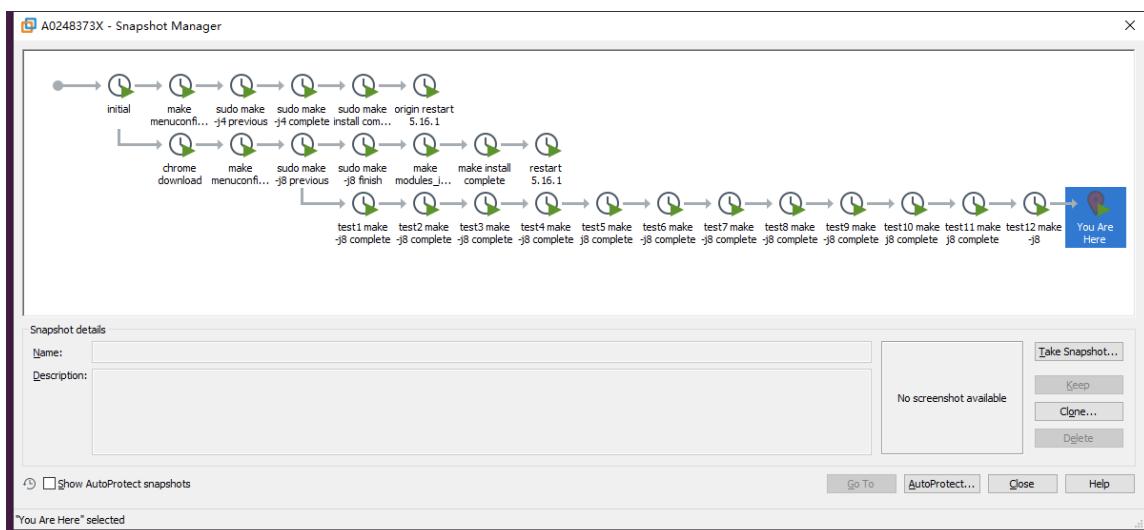
Failed to create socket: Function not implemented
Failed to initialize udev control socket: Function not implemented
Failed to create manager: Function not implemented
Gave up waiting for root file system device.  Common problems:
  - Boot args (cat /proc/cmdline)
  - Check rootdelay= (did the system wait long enough?)
  - Missing modules (cat /proc/modules; ls /dev)
ALERT!  UUID=bef59208-5d99-47b6-ae7e-a711b39033f1 does not exist.  Dropping to a
shell!

BusyBox v1.30.1 (Ubuntu 1:1.30.1-6ubuntu3.1) built-in shell (ash)
Enter 'help' for a list of built-in commands.

(initramfs)

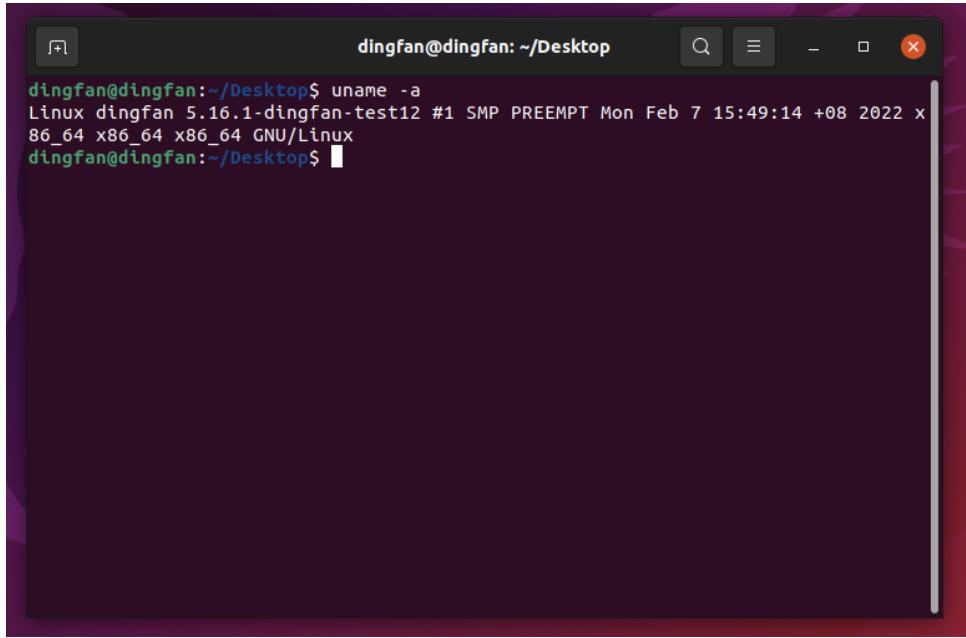
```

4. When I encounter failures, I need to goto the previous snapshot of my virtual machine and retry excluded features.



5. The smallest kernel:
 - The name of my smallest kernel version is **5.16.1-dingfan-test12**

the screenshot when reboot successfully



A screenshot of a terminal window titled "dingfan@dingfan: ~/Desktop". The window contains the following text:

```
dingfan@dingfan:~/Desktop$ uname -a
Linux dingfan 5.16.1-dingfan-test12 #1 SMP PREEMPT Mon Feb 7 15:49:14 +08 2022 x
86_64 x86_64 x86_64 GNU/Linux
dingfan@dingfan:~/Desktop$ █
```

- As a result, the size of the smallest kernel I built is [6.1MB](#), which is highlight in the following picture

```

dingfan@dingfan: /boot
drwxr-xr-x 1 root root 257253 Jan 14 01:13 config-5.13.0-28-generic
-rw-r--r-- 1 root root 194469 Feb 5 18:30 config-5.16.1
-rw-r--r-- 1 root root 139133 Feb 7 16:24 config-5.16.1-dingfan-test12
-rw-r--r-- 1 root root 139234 Feb 6 18:16 config-5.16.1-testv10
-rw-r--r-- 1 root root 139425 Feb 6 20:40 config-5.16.1-testv11
-rw-r--r-- 1 root root 169798 Feb 6 03:28 config-5.16.1-testv4
-rw-r--r-- 1 root root 171017 Feb 6 00:21 config-5.16.1-testv4.old
-rw-r--r-- 1 root root 146138 Feb 6 13:01 config-5.16.1-testv6
-rw-r--r-- 1 root root 140016 Feb 6 14:54 config-5.16.1-testv7
-rw-r--r-- 1 root root 139222 Feb 6 17:10 config-5.16.1-testv9
drwxr-xr-x 3 root root 4096 Jan 1 1970 efi/
drwxr-xr-x 5 root root 4096 Feb 7 16:24 grub/
lrwxrwxrwx 1 root root 32 Feb 7 16:24 initrd.img -> initrd.img-5.16.1-dingfan-test12
-rw-r--r-- 1 root root 42942780 Feb 3 19:23 initrd.img-5.13.0-19-generic
-rw-r--r-- 1 root root 42995930 Feb 3 19:24 initrd.img-5.13.0-28-generic
-rw-r--r-- 1 root root 296024094 Feb 5 18:35 initrd.img-5.16.1
-rw-r--r-- 1 root root 19731279 Feb 7 16:24 initrd.img-5.16.1-dingfan-test12
-rw-r--r-- 1 root root 23263087 Feb 6 18:17 initrd.img-5.16.1-testv10
-rw-r--r-- 1 root root 19728919 Feb 6 20:41 initrd.img-5.16.1-testv11
-rw-r--r-- 1 root root 200716565 Feb 6 03:32 initrd.img-5.16.1-testv4
-rw-r--r-- 1 root root 107493274 Feb 6 13:03 initrd.img-5.16.1-testv6
-rw-r--r-- 1 root root 19731657 Feb 6 14:55 initrd.img-5.16.1-testv7
-rw-r--r-- 1 root root 0 Feb 6 17:10 initrd.img-5.16.1-testv9.new
lrwxrwxrwx 1 root root 25 Feb 7 16:24 initrd.img.old -> initrd.img-5.16.1-testv11
-rw-r--r-- 1 root root 182704 Oct 7 18:20 memtest86+.bin
-rw-r--r-- 1 root root 184380 Oct 7 18:20 memtest86+.elf
-rw-r--r-- 1 root root 184884 Oct 7 18:20 memtest86+_multiboot.bin
-rw----- 1 root root 5966639 Oct 7 17:09 System.map-5.13.0-19-generic
-rw----- 1 root root 5972177 Jan 14 01:13 System.map-5.13.0-28-generic
-rw-r--r-- 1 root root 5446468 Feb 5 18:30 System.map-5.16.1
-rw-r--r-- 1 root root 3818233 Feb 7 16:24 System.map-5.16.1-dingfan-test12
-rw-r--r-- 1 root root 9967213 Feb 6 18:16 System.map-5.16.1-testv10
-rw-r--r-- 1 root root 3839460 Feb 6 20:40 System.map-5.16.1-testv11
-rw-r--r-- 1 root root 5100322 Feb 6 03:28 System.map-5.16.1-testv4
-rw-r--r-- 1 root root 5099329 Feb 6 00:21 System.map-5.16.1-testv4.old
-rw-r--r-- 1 root root 4953438 Feb 6 13:01 System.map-5.16.1-testv6
-rw-r--r-- 1 root root 3858248 Feb 6 14:54 System.map-5.16.1-testv7
-rw-r--r-- 1 root root 9973504 Feb 6 17:10 System.map-5.16.1-testv9
lrwxrwxrwx 1 root root 29 Feb 7 16:24 vmlinuz -> vmlinuz-5.16.1-dingfan-test12
-rw-r--r-- 1 root root 10221728 Oct 13 05:04 vmlinuz-5.13.0-19-generic
-rw----- 1 root root 10237312 Jan 14 01:10 vmlinuz-5.13.0-28-generic
-rw-r--r-- 1 root root 9462400 Feb 5 18:30 vmlinuz-5.16.1
-rw-r--r-- 1 root root 6127712 Feb 7 16:24 vmlinuz-5.16.1-dingfan-test12
23884800 Feb 6 18:16 vmlinuz-5.16.1-testv10
-rw-r--r-- 1 root root 6182240 Feb 6 20:40 vmlinuz-5.16.1-testv11
-rw-r--r-- 1 root root 8708192 Feb 6 03:28 vmlinuz-5.16.1-testv4
-rw-r--r-- 1 root root 8589152 Feb 6 00:21 vmlinuz-5.16.1-testv4.old
-rw-r--r-- 1 root root 8484320 Feb 6 13:01 vmlinuz-5.16.1-testv6
-rw-r--r-- 1 root root 6202688 Feb 6 14:54 vmlinuz-5.16.1-testv7
-rw-r--r-- 1 root root 23887328 Feb 6 17:10 vmlinuz-5.16.1-testv9
lrwxrwxrwx 1 root root 22 Feb 6 20:40 vmlinuz.old -> vmlinuz-5.16.1-testv11
dingfan@dingfan: /boot$ 

```

- compared to the original kernel(5.16.1) , the kernel size change from **10MB** to **6.1MB**

6. List the options that I disabled. (the feature I excluded):

<u>Aa</u> Features	≡ sublevel-1	≡ sublevel-2	≡ sublevel-3
<u>Processor type and features</u>	Enable MPS table		
<u>Processor type and features</u>	Support for extended (non-PC) x86 platforms		
<u>Processor type and features</u>	Enable Maximum number of SMP Processors and NUMA Nodes		
<u>Processor type and features</u>	Enable support for 16-bit segments		
<u>Power Management and ACPI options</u>	Garbage collector for user space wakeup sources		
<u>Power Management and ACPI options</u>	Power Management Debug Support		
<u>Binary Emulation</u>	x32 ABI for 64-bit mode		
<u>Virtualization</u>	all		
<u>Executable file formats</u>	Write ELF core dumps with partial segments		
<u>Executable file formats</u>	Enable core dump support		
<u>Networking Support</u>	networking options		
<u>Networking Support</u>	networking options	TCP/IP networking	
<u>Networking Support</u>	networking options	security marking	

<u>Aa</u> Features	☰ sublevel-1	☰ sublevel-2	☰ sublevel-3
<u>Networking Support</u>	networking options	timestamping in phy devices	
<u>Networking Support</u>	networking options	802.1d Ethernet Bridging	
<u>Networking Support</u>	networking options	802.10/802.1ad VLAN Support	
<u>Networking Support</u>	networking options	Phonet protocols family	
<u>Networking Support</u>	networking options	IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks support	
<u>Networking Support</u>	networking options	Qos and/or fair queueing	
<u>Networking Support</u>	networking options	Data Center Bridging support	
<u>Networking Support</u>	networking options	DNS Resolver support	
<u>Networking Support</u>	networking options	The IPV6 protocol	
<u>Networking Support</u>	networking options	MPTCP: Multipath TCP	
<u>Networking Support</u>	networking options	appletalk protocol support	
<u>Networking Support</u>	networking options	open vswitch	
<u>Networking Support</u>	networking options	network coding	
<u>Networking Support</u>	networking options	Multiprotocol Label Switching	
<u>Networking Support</u>	networking options	Use percpu variables to maintain network device refcount	
<u>Networking Support</u>	amateur radio support	all	
<u>Networking Support</u>	bluetooth subsystem support	all	

Aa Features	☰ sublevel-1	☰ sublevel-2	☰ sublevel-3
<u>Networking Support</u>	IPV6 support for rxrpc		
<u>Networking Support</u>	rxrpc kerberos security		
<u>Networking Support</u>	wireless	all	
<u>Networking Support</u>	RF switch subsystem support	all	
<u>Networking Support</u>	plan 9 resource sharing support(9p2000)	all	
<u>Networking Support</u>	CAIF support	all	
<u>Networking Support</u>	use in-kernel support for dns lookup		
<u>Networking Support</u>	NFC subsystem	all	
<u>Networking Support</u>	packet-sampling netlink channel		
<u>Networking Support</u>	Network light weight tunnels	all	
<u>Networking Support</u>	netlink interface for ethtool		
<u>Device Drivers</u>	PCI	PCI Express Hotplug driver	
<u>Device Drivers</u>	PCI	PCI Express ASPM control	
<u>Device Drivers</u>	PCI	Enable PCI resource re-allocation detection	
<u>Device Drivers</u>	PCI	Xen PCI Frontend	
<u>Device Drivers</u>	Generic Driver Options	Support for uevent helper	
<u>Device Drivers</u>	Generic Driver Options	Automount devtmpfs at /dev, after the kernel mounted the rootfs	
<u>Device Drivers</u>	Generic Driver Options	Allow device coredump	
<u>Device Drivers</u>	Macintosh device drivers		
<u>Device Drivers</u>	network device support	all	

Aa Features	☰ sublevel-1	☰ sublevel-2	☰ sublevel-3
<u>Device Drivers</u>	input device support		
<u>Device Drivers</u>	input device support	Joystick interface	
<u>Device Drivers</u>	input device support	Event debugging	
<u>Device Drivers</u>	input device support	touchscreens	
<u>Device Drivers</u>	input device support	joysticks/gamepads	
<u>Device Drivers</u>	input device support	Tablets	
<u>Device Drivers</u>	input device support	hardware I/O support	ct82c710 Aux port controller
<u>Device Drivers</u>	input device support	hardware I/O support	Parallel port keyboard adapter
<u>Device Drivers</u>	input device support	hardware I/O support	Raw access to serio ports
<u>Device Drivers</u>	input device support	hardware I/O support	Altera UP PS/2 controller
<u>Device Drivers</u>	input device support	hardware I/O support	Gameport support
<u>Device Drivers</u>	character devices	Legacy (BSD) PTY support	
<u>Device Drivers</u>	character devices	Non-standard serial port support	
<u>Device Drivers</u>	character devices	GSM MUX line discipline support	
<u>Device Drivers</u>	character devices	HSDPA Broadband Wireless Data Card - Globe Trotter	
<u>Device Drivers</u>	character devices	Parallel printer support	
<u>Device Drivers</u>	character devices	Support for user-space parallel port device drivers	
<u>Device Drivers</u>	character devices	IPMI top-level message handler	
<u>Device Drivers</u>	character devices	Applicom intelligent fieldbus card support	
<u>Device Drivers</u>	character devices	PCMCIA character devices	

<u>Aa</u> Features	≡ sublevel-1	≡ sublevel-2	≡ sublevel-3
<u>Device Drivers</u>	character devices	ACP Modem (Mwave) support	
<u>Device Drivers</u>	character devices	Hangcheck timer	
<u>Device Drivers</u>	character devices	Telecom clock driver for ATCA SBC	
<u>Device Drivers</u>	PPS support		
<u>Device Drivers</u>	Generic Thermal sysfs driver		
<u>Device Drivers</u>	remote controller support		
<u>Device Drivers</u>	multimedia support		
<u>Device Drivers</u>	sound card support		
<u>Device Drivers</u>	usb support		
<u>Device Drivers</u>	MMC/SD/SDIO card support		
<u>Device Drivers</u>	sony memorystick card support		
<u>Device Drivers</u>	accessibility support		
<u>Device Drivers</u>	Auxiliary Display support		
<u>Device Drivers</u>	VFIO Non-Privileged userspace driver framework		
<u>Device Drivers</u>	Microsoft Hyper-V guest support		
<u>Device Drivers</u>	Xen driver support		
<u>Device Drivers</u>	Generic Dynamic Voltage and Frequency Scaling (DVFS) support		
<u>Device Drivers</u>	External Connector Class (extcon) support		
<u>Device Drivers</u>	staging drivers		
<u>Device Drivers</u>	Platform support for Chrome hardware		
<u>Device Drivers</u>	Platform support for Mellanox hardware		
<u>Device Drivers</u>	Microsoft Surface platform-specific device drivers		

<u>Aa</u> Features	☰ sublevel-1	☰ sublevel-2	☰ sublevel-3
<u>Device Drivers</u>	Memory Controller drivers		
<u>Device Drivers</u>	Industrial I/O support		
<u>Device Drivers</u>	Non-Transparent Bridge support		
<u>Device Drivers</u>	VME bridge support		
<u>Device Drivers</u>	Pulse-Width Modulation (PWM) Support		
<u>Device Drivers</u>	IndustryPack bus support		
<u>Device Drivers</u>	Reset Controller Support		
<u>Device Drivers</u>	PHY Subsystem		
<u>Device Drivers</u>	Generic powercap sysfs driver		
<u>Device Drivers</u>	MCB support		
<u>Device Drivers</u>	Thunderbolt support for Apple devices		
<u>Device Drivers</u>	Reliability, Availability and Serviceability (RAS) features		
<u>Device Drivers</u>	android	android drivers	
<u>Security Options</u>	all		
<u>Cryptographic API</u>	all		
<u>Library Routines</u>	all		
<u>Kernel Hacking</u>	all		

Part B: x86 Assembly Programming (10 marks)

1.What is the IA32 (i.e., 32-bit Intel processor) instruction corresponding to these bytes, showing each step of your decoding process?

03 64 02 c5 (base 16)

solution:

1. (Byte1: **03H**), because this instruction is a IA32 instruction. So the first byte is a Opcode byte.

- The Opcode of IA32 can be 1, 2 or 4 byte. Because the first opcode byte is not **0FH** or **0F 38H**, we can know that the opcode is 1 byte.
- After looking the Table A-2 One-Byte Opcode Map, **03H** is **ADD (Gv, Ev)**

Table A-2. One-byte Opcode Map: (00H – F7H) *

	0	1	2	3	4	5	6	7
0	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz	PUSH ES ⁶⁴	POP ES ⁶⁴
1	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz	PUSH SS ⁶⁴	POP SS ⁶⁴
2	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz	SEG=ES (Prefix)	DAA ⁶⁴
3	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz	SEG=SS (Prefix)	AAA ⁶⁴
4	eAX REX	eCX REX.B	eDX REX.X	eBX REX.XB	eSP REX.R	eBP REX.RB	eSI REX.RX	eDI REX.RXB
5	rAX/r8	rCX/r9	rDX/r10	rBX/r11	rSP/r12	rBP/r13	rSI/r14	rDI/r15
6	PUSHA ⁶⁴ /PUSHAD ⁶⁴	POPA ⁶⁴ /POPAD ⁶⁴	BOUND ⁶⁴ Gv, Ma	ARPL ⁶⁴ Ew, Gw MOVSD ⁶⁴ Gv, Ev	SEG=FS (Prefix)	SEG=GS (Prefix)	Operand Size (Prefix)	Address Size (Prefix)
7	O	NO	B/NAE/C	NB/AE/NC	Z/E	NZ/NE	BE/NA	NBE/A
8	Eb, Ib	Ev, Iz	Eb, Ib ⁶⁴	Ev, Ib	Eb, Gb	Ev, Gv	Eb, Gb	Ev, Gv
9	NOP PAUSE(F3) XCHG r8, rAX	rCX/r9	rDX/r10	rBX/r11	rSP/r12	rBP/r13	rSI/r14	rDI/r15
A	AL, Ob	rAX, Ov	Ob, AL	Ov, rAX	MOVS/B Yb, Xb	MOVS/W/D/Q Yv, Xv	CMPS/B Xb, Yb	CMPW/W/D Xv, Yv
B	AL/R8B, Ib	CL/R9B, Ib	DL/R10B, Ib	BL/R11B, Ib	AH/R12B, Ib	CH/R13B, Ib	DH/R14B, Ib	BH/R15B, Ib
C	Shift Grp 2 ^{1A} Eb, Ib	near RET ⁶⁴ Ev, Ib	near RET ⁶⁴ Iw	LES ⁶⁴ Gz, Mp VEX+2byte	LDS ⁶⁴ Gz, Mp VEX+1byte	Grp 11 ^{1A} - MOV Eb, Ib		Ev, Iz
D	Shift Grp 2 ^{1A} Eb, 1	Ev, 1	Eb, CL	Ev, CL	AAM ⁶⁴ Ib	AAD ⁶⁴ Ib		XLAT/ XLATB
E	LOOPNE ⁶⁴ /LOOPNZ ⁶⁴ Jb	LOOP ⁶⁴ /LOOPZ ⁶⁴ Jb	LOOP ⁶⁴ Jb	JrCXZ ⁶⁴ /Jb	IN AL, Ib	OUT eAX, Ib	Ib, AL	Ib, eAX
F	LOCK (Prefix)	INT1	REPNE XACQUIRE (Prefix)	REP/REPE XRELEASE (Prefix)	HLT	CMC	Unary Grp 3 ^{1A} Eb	Ev

- G** means the operand is REG, **E** means the operand is REG or MEM
- so the instruction is a ADD instruction

ADD—Add

Opcode	Instruction	Op/ En	64-bit Mode	Compat/ Leg Mode	Description
04 ib	ADD AL, imm8	I	Valid	Valid	Add imm8 to AL.
05 iw	ADD AX, imm16	I	Valid	Valid	Add imm16 to AX.
05 id	ADD EAX, imm32	I	Valid	Valid	Add imm32 to EAX.
REX.W + 05 id	ADD RAX, imm32	I	Valid	N.E.	Add imm32 sign-extended to 64-bits to RAX.
80 /0 ib	ADD r/m8, imm8	M ₁	Valid	Valid	Add imm8 to r/m8.
REX + 80 /0 ib	ADD r/m8*, imm8	M ₁	Valid	N.E.	Add sign-extended imm8 to r/m8.
81 /0 iw	ADD r/m16, imm16	M ₁	Valid	Valid	Add imm16 to r/m16.
81 /0 id	ADD r/m32, imm32	M ₁	Valid	Valid	Add imm32 to r/m32.
REX.W + 81 /0 id	ADD r/m64, imm32	M ₁	Valid	N.E.	Add imm32 sign-extended to 64-bits to r/m64.
83 /0 ib	ADD r/m16, imm8	M ₁	Valid	Valid	Add sign-extended imm8 to r/m16.
83 /0 ib	ADD r/m32, imm8	M ₁	Valid	Valid	Add sign-extended imm8 to r/m32.
REX.W + 83 /0 ib	ADD r/m64, imm8	M ₁	Valid	N.E.	Add sign-extended imm8 to r/m64.
00 /r	ADD r/m8, r8	M _R	Valid	Valid	Add r8 to r/m8.
REX + 00 /r	ADD r/m8*, r8*	M _R	Valid	N.E.	Add r8 to r/m8.
01 /r	ADD r/m16, r16	M _R	Valid	Valid	Add r16 to r/m16.
01 /r	ADD r/m32, r32	M _R	Valid	Valid	Add r32 to r/m32.
REX.W + 01 /r	ADD r/m64, r64	M _R	Valid	N.E.	Add r64 to r/m64.
02 /r	ADD r8, r/m8	R _M	Valid	Valid	Add r/m8 to r8.
REX + 02 /r	ADD r8*, r/m8*	R _M	Valid	N.E.	Add r/m8 to r8.
03 /r	ADD r16, r/m16	R _M	Valid	Valid	Add r/m16 to r16.
03 /r	ADD r32, r/m32	R _M	Valid	Valid	Add r/m32 to r32.
REX.W + 03 /r	ADD r64, r/m64	R _M	Valid	N.E.	Add r/m64 to r64.

NOTES:

*In 64-bit mode, r/m8 can not be encoded to access the following byte registers if a REX prefix is used: AH, BH, CH, DH.

2. (Byte2: **64H**) the second byte (after opcode byte) of the instruction is a ModR/M byte.

- **64H** \Rightarrow **01100100** (base 2)
- the 3 field of ModR/M is
 - MOD: **01**
 - REG: **100**
 - R/M: **100**
- after looking the table

Table 2-2. 32-Bit Addressing Forms with the ModR/M Byte

r8(r) r16(r) r32(r) mm(r) xmm(r) (In decimal) /digit (Opcode) (In binary) REG =			AL AX EAX MM0 XMM0 0 000	CL CX ECX MM1 XMM1 1 001	DL DX EDX MM2 XMM2 2 010	BL BX EBX MM3 XMM3 3 011	AH SP ESP MM4 XMM4 4 100	CH BP EBP MM5 XMM5 5 101	DH SI ESI MM6 XMM6 6 110	BH DI EDI MM7 XMM7 7 111
Effective Address	Mod	R/M	Value of ModR/M Byte (in Hexadecimal)							
[EAX]	00	000	00	08	10	18	20	28	30	38
[ECX]		001	01	09	11	19	21	29	31	39
[EDX]		010	02	0A	12	1A	22	2A	32	3A
[EBX]		011	03	0B	13	1B	23	2B	33	3B
[--][--] ¹		100	04	0C	14	1C	24	2C	34	3C
disp32 ²		101	05	0D	15	1D	25	2D	35	3D
[ESI]		110	06	0E	16	1E	26	2E	36	3E
[EDI]		111	07	0F	17	1F	27	2F	37	3F
[EAX]+disp8 ³	01	000	40	48	50	58	60	68	70	78
[ECX]+disp8		001	41	49	51	59	61	69	71	79
[EDX]+disp8		010	42	4A	52	5A	62	6A	72	7A
[EBX]+disp8		011	43	4B	53	5B	63	6B	73	7B
[--][--]+disp8		100	44	4C	54	5C	64	6C	74	7C
[EBP]+disp8		101	45	4D	55	5D	65	6D	75	7D
[ESI]+disp8		110	46	4E	56	5E	66	6E	76	7E
[EDI]+disp8		111	47	4F	57	5F	67	6F	77	7F
[EAX]+disp32	10	000	80	88	90	98	A0	A8	B0	B8
[ECX]+disp32		001	81	89	91	99	A1	A9	B1	B9
[EDX]+disp32		010	82	8A	92	9A	A2	AA	B2	BA
[EBX]+disp32		011	83	8B	93	9B	A3	AB	B3	BB
[--][--]+disp32		100	84	8C	94	9C	A4	AC	B4	BC
[EBP]+disp32		101	85	8D	95	9D	A5	AD	B5	BD
[ESI]+disp32		110	86	8E	96	9E	A6	AE	B6	BE
[EDI]+disp32		111	87	8F	97	9F	A7	AF	B7	BF
EAX/AX/AL/MM0/XMM0 ECX/CX/CL/MM1/XMM1 EDX/DX/DL/MM2/XMM2 EBX/BX/BL/MM3/XMM3 ESP/SP/AH/MM4/XMM4 EBP/BP/CH/MM5/XMM5 ESI/SI/DH/MM6/XMM6 EDI/DI/BH/MM7/XMM7	11	000	C0	C8	D0	D8	E0	E8	F0	F8
		001	C1	C9	D1	D9	E1	E9	F1	F9
		010	C2	CA	D2	DA	E2	EA	F2	FA
		011	C3	CB	D3	DB	E3	EB	F3	FB
		100	C4	CC	D4	DC	E4	EC	F4	FC
		101	C5	CD	D5	DD	E5	ED	F5	FD
		110	C6	CE	D6	DE	E6	EE	F6	FE
		111	C7	CF	D7	DF	E7	EF	F7	FF

NOTES:

- The [-][-] nomenclature means a SIB follows the ModR/M byte.
- The disp32 nomenclature denotes a 32-bit displacement that follows the ModR/M byte (or the SIB byte if one is present) and that is added to the index.
- The disp8 nomenclature denotes an 8-bit displacement that follows the ModR/M byte (or the SIB byte if one is present) and that is sign-extended and added to the index.

- MOD: **01** means one-byte signed displacement follows addressing mode byte
- REG: **100** means the source or destination register is **ESP**
- R/M: **100** combined with MOD: **01** means that 8-bit displacement(1 byte) that follows the ModR/M byte and SIB Byte
 - so the third byte of the instruction is a SIB byte, the fourth byte of the instruction is a displacement byte

- (Byte3: **02H**), according to the decoding of ModR/M byte, we know the third byte is a SIB byte

- 02H** \Rightarrow **00000010** (base2)
 - scale : **00**
 - index: **000**
 - base: **010**

- after looking the following table

Table 2-3. 32-Bit Addressing Forms with the SIB Byte

r32 (In decimal) Base = (In binary) Base =			EAX	ECX	EDX	EBX	ESP	[*]	ESI	EDI
Scaled Index	SS	Index	Value of SIB Byte (in Hexadecimal)							
[EAX]	00	000	00	01	02	03	04	05	06	07
[ECX]		001	08	09	0A	0B	0C	0D	0E	0F
[EDX]		010	10	11	12	13	14	15	16	17
[EBX]		011	18	19	1A	1B	1C	1D	1E	1F
none		100	20	21	22	23	24	25	26	27
[EBP]		101	28	29	2A	2B	2C	2D	2E	2F
[ESI]		110	30	31	32	33	34	35	36	37
[EDI]		111	38	39	3A	3B	3C	3D	3E	3F
[EAX*2]	01	000	40	41	42	43	44	45	46	47
[ECX*2]		001	48	49	4A	4B	4C	4D	4E	4F
[EDX*2]		010	50	51	52	53	54	55	56	57
[EBX*2]		011	58	59	5A	5B	5C	5D	5E	5F
none		100	60	61	62	63	64	65	66	67
[EBP*2]		101	68	69	6A	6B	6C	6D	6E	6F
[ESI*2]		110	70	71	72	73	74	75	76	77
[EDI*2]		111	78	79	7A	7B	7C	7D	7E	7F
[EAX*4]	10	000	80	81	82	83	84	85	86	87
[ECX*4]		001	88	89	8A	8B	8C	8D	8E	8F
[EDX*4]		010	90	91	92	93	94	95	96	97
[EBX*4]		011	98	99	9A	9B	9C	9D	9E	9F
none		100	A0	A1	A2	A3	A4	A5	A6	A7
[EBP*4]		101	A8	A9	AA	AB	AC	AD	AE	AF
[ESI*4]		110	B0	B1	B2	B3	B4	B5	B6	B7
[EDI*4]		111	B8	B9	BA	BB	BC	BD	BE	BF
[EAX*8]	11	000	C0	C1	C2	C3	C4	C5	C6	C7
[ECX*8]		001	C8	C9	CA	CB	CC	CD	CE	CF
[EDX*8]		010	D0	D1	D2	D3	D4	D5	D6	D7
[EBX*8]		011	D8	D9	DA	DB	DC	DD	DE	DF
none		100	E0	E1	E2	E3	E4	E5	E6	E7
[EBP*8]		101	E8	E9	EA	EB	EC	ED	EE	EF
[ESI*8]		110	F0	F1	F2	F3	F4	F5	F6	F7
[EDI*8]		111	F8	F9	FA	FB	FC	FD	FE	FF

NOTES:

1. The [*] nomenclature means a disp32 with no base if the MOD is 00B. Otherwise, [*] means disp8 or disp32 + [EBP]. This provides the following address modes:

MOD bits Effective Address

00 [scaled index] + disp32

01 [scaled index] + disp8 + [EBP]

10 [scaled index] + disp32 + [EBP]

- scale : 00 means Index*1

- index: 000 means EAX

- base: 010 means EDX

4. (Byte4: C5H), according to the decoding of ModR/M byte, we know the fourth byte is signed one byte displacement.

- C5H ⇒ 11000101 (base 2) ⇒ -59 (decimal)

5. According to the previous 4 steps decoding process, we know the instruction is



ADDL -59(%edx, %eax), %esp

2. What is the 64-bit instruction (Intel64) encoding for the following instruction. You need to show each step of your

workings.

`addq -0xf00(%rax, %r12, 2), %rbp`

solution:

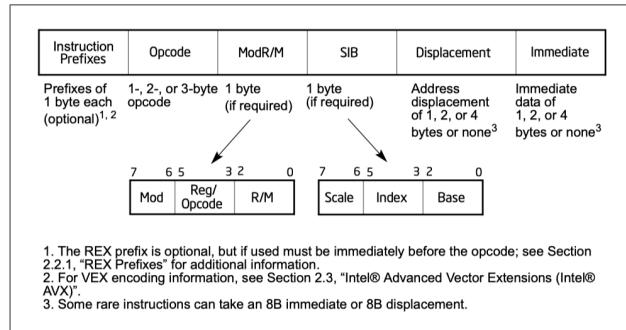


Figure 2-1. Intel 64 and IA-32 Architectures Instruction Format

1. Encoding the opcode byte and part of the prefix byte:

- from the instruction we know it is a add instruction that Add `m64` to `r64`
- after looking the following table

ADD—Add

Opcode	Instruction	Op/ En	64-bit Mode	Compat/ Leg Mode	Description
04 ib	ADD AL, imm8	I	Valid	Valid	Add imm8 to AL.
05 iw	ADD AX, imm16	I	Valid	Valid	Add imm16 to AX.
05 id	ADD EAX, imm32	I	Valid	Valid	Add imm32 to EAX.
REXW + 05 id	ADD RAX, imm32	I	Valid	N.E.	Add imm32 sign-extended to 64-bits to RAX.
80 /0 ib	ADD r/m8, imm8	Mi	Valid	Valid	Add imm8 to r/m8.
REX + 80 /0 ib	ADD r/m8 , imm8	Mi	Valid	N.E.	Add sign-extended imm8 to r/m8.
81 /0 iw	ADD r/m16, imm16	Mi	Valid	Valid	Add imm16 to r/m16.
81 /0 id	ADD r/m32, imm32	Mi	Valid	Valid	Add imm32 to r/m32.
REXW + 81 /0 id	ADD r/m64, imm32	Mi	Valid	N.E.	Add imm32 sign-extended to 64-bits to r/m64.
83 /0 ib	ADD r/m16, imm8	Mi	Valid	Valid	Add sign-extended imm8 to r/m16.
83 /0 ib	ADD r/m32, imm8	Mi	Valid	Valid	Add sign-extended imm8 to r/m32.
REXW + 83 /0 ib	ADD r/m64, imm8	Mi	Valid	N.E.	Add sign-extended imm8 to r/m64.
00 /r	ADD r/m8, r8	MR	Valid	Valid	Add r8 to r/m8.
REX + 00 /r	ADD r/m8 , r8	MR	Valid	N.E.	Add r8 to r/m8.
01 /r	ADD r/m16, r16	MR	Valid	Valid	Add r16 to r/m16.
01 /r	ADD r/m32, r32	MR	Valid	Valid	Add r32 to r/m32.
REXW + 01 /r	ADD r/m64, r64	MR	Valid	N.E.	Add r64 to r/m64.
02 /r	ADD r8, r/m8	RM	Valid	Valid	Add r/m8 to r8.
REX + 02 /r	ADD r8 , r/m8	RM	Valid	N.E.	Add r/m8 to r8.
03 /r	ADD r16, r/m16	RM	Valid	Valid	Add r/m16 to r16.
03 /r	ADD r32, r/m32	RM	Valid	Valid	Add r/m32 to r32.
REXW + 03 /r	ADD r64, r/m64	RM	Valid	N.E.	Add r/m64 to r64.

NOTES:

*In 64-bit mode, r/m8 can not be encoded to access the following byte registers if a REX prefix is used: AH, BH, CH, DH.

- Because we know it is a add instruction that Add `m64` to `r64`, the Opcode is `REX.W+03/r`
- So we can get the opcode byte is `03H`
- Because this is a 64-bit instruction, the prefix byte is required. The `REX.W` is a part of instruction prefix byte.

- According to the following table, the first 5 bits of prefix byte is **01001**; the last 3 bits of prefix byte is **RXB**, the value of **RXB** needs to be specified in the following encoding step.

Table 2-4. REX Prefix Fields [BITS: 0100WRXB]

Field Name	Bit Position	Definition
-	7:4	0100
W	3	0 = Operand size determined by CS.D
		1 = 64 Bit Operand Size
R	2	Extension of the ModR/M reg field
X	1	Extension of the SIB index field
B	0	Extension of the ModR/M r/m field, SIB base field, or Opcode reg field

2. Encoding the ModRM byte, SIB byte and displacement byte

- According to the part of the instruction **-0xf00(%rax, %r12, 2), %rbp**, we know the 64-bit encoding instruction, must have a displacement byte (bytes) follows the ModRM byte and SIB byte, as the following table:

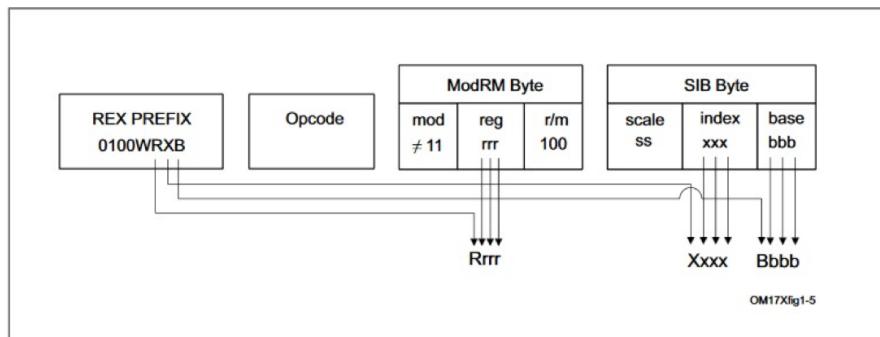


Figure 2-6. Memory Addressing With a SIB Byte

a. Encoding the ModRM byte

- 0xf00** is the displacement which is longer than 1 byte (8 bit). After looking the following table, we know the displacement needs to be represented as a 4byte displacement. The effective address is **[SIB +dis32]**. So we can get the value of MOD and B.R/M:

- MOD: **10**
- B.R/M: **1.100** \Rightarrow B: **0**, R/M: **100**

ModRM in 64 bit

32/64-bit		B.R/M																	
Mod		0.000 AX	0.001 CX	0.010 DX	0.011 BX	0.100 SP	0.101 BP	0.110 SI	0.111 DI	1.000 R8	1.001 R9	1.010 R10	1.011 R11	1.100 R12	1.101 R13	1.110 R14	1.111 R15		
00		[r/m]		[SIB]		[RIP/EIP1.2 + disp32]		[r/m]		[SIB]		[RIP/EIP1.2 + disp32]		[r/m]					
01		[r/m + disp8]		[SIB + disp8]		[r/m + disp8]		[SIB + disp8]		[r/m + disp8]									
10		[r/m + disp32]		[SIB + disp32]		[r/m + disp32]		[SIB + disp32]		[r/m + disp32]									
11		r/m																	

- According to the following table with `%rbp`, we can get the value of R.Reg = `0.101`, the X.Reg in this table refer to the R.Reg in Intel Manual, R is the specific bit in prefix byte

Registers

The registers are encoded using the 4-bit values in the X.Reg column of the following table. X.Reg is in binary.

X.Reg	8-bit GP	16-bit GP	32-bit GP	64-bit GP	80-bit x87	64-bit MMX	128-bit XMM	256-bit YMM	16-bit Segment	32-bit Control	32-bit Debug
0.000 (0)	AL	AX	EAX	RAX	ST0	MMX0	XMM0	YMM0	ES	CR0	DR0
0.001 (1)	CL	CX	ECX	RCX	ST1	MMX1	XMM1	YMM1	CS	CR1	DR1
0.010 (2)	DL	DX	EDX	RDX	ST2	MMX2	XMM2	YMM2	SS	CR2	DR2
0.011 (3)	BL	BX	EBX	RBX	ST3	MMX3	XMM3	YMM3	DS	CR3	DR3
0.100 (4)	AH, SPL ¹	SP	ESP	RSP	ST4	MMX4	XMM4	YMM4	FS	CR4	DR4
0.101 (5)	CH, BPL ¹	BP	EBP	RBP	ST5	MMX5	XMM5	YMM5	GS	CR5	DR5
0.110 (6)	DH, SIL ¹	SI	ESI	RSI	ST6	MMX6	XMM6	YMM6	-	CR6	DR6
0.111 (7)	BH, DIL ¹	DI	EDI	RDI	ST7	MMX7	XMM7	YMM7	-	CR7	DR7
1.000 (8)	R8L	R8W	R8D	R8	-	MMX0	XMM8	YMM8	ES	CR8	DR8
1.001 (9)	R9L	R9W	R9D	R9	-	MMX1	XMM9	YMM9	CS	CR9	DR9
1.010 (10)	R10L	R10W	R10D	R10	-	MMX2	XMM10	YMM10	SS	CR10	DR10
1.011 (11)	R11L	R11W	R11D	R11	-	MMX3	XMM11	YMM11	DS	CR11	DR11
1.100 (12)	R12L	R12W	R12D	R12	-	MMX4	XMM12	YMM12	FS	CR12	DR12
1.101 (13)	R13L	R13W	R13D	R13	-	MMX5	XMM13	YMM13	GS	CR13	DR13
1.110 (14)	R14L	R14W	R14D	R14	-	MMX6	XMM14	YMM14	-	CR14	DR14
1.111 (15)	R15L	R15W	R15D	R15	-	MMX7	XMM15	YMM15	-	CR15	DR15

1: When any REX prefix is used, SPL, BPL, SIL and DIL are used. Otherwise, without any REX prefix AH, CH, DH and BH are used.

o R : `0`

o Reg : `101`

- Because I already get : MOD: `10`, Reg : `101`, R/M: `100`,
- the ModRM byte is `10101100` (base2) \Rightarrow `ACH`

b. Encoding the SIB byte

- according to `(%rax, %r12, 2)`

- we can get
 - Index register: `%r12`
 - Base register: `%rax`
 - scale value: `2` ⇒ scale = `01`
- after looking the following table
- after looking the following table, we can get the value of X.index and B.base

		B.Base															
Mod	X.Index	0.000	0.001	0.010	0.011	0.100	0.101	0.110	0.111	1.000	1.001	1.010	1.011	1.100	1.101	1.110	1.111
		AX	CX	DX	BX	SP	BP	SI	DI	R8	R9	R10	R11	R12	R13	R14	R15
	0.000 AX																
	0.001 CX																[base + (index * s) + disp32]
	0.010 DX																
	0.011 BX																
	0.100 ² SP																[base + disp32]
	0.101 BP																
	0.110 SI																
	0.111 DI																
10	1.000 R8																
	1.001 R9																
	1.010 R10																[base + (index * s) + disp32]
	1.011 R11																
	1.100 R12																
	1.101 R13																
	1.110 R14																
	1.111 R15																

- X.index: `1100` ⇒ X: `1`, index: `100`
- B.base: `0.000` ⇒ B: `0`, base: `000`
- because scale = `01`, index: `100`, base: `000`
- the SIB byte is `01100000` (base2) ⇒ `60H`

c. Encoding the prefix byte

- Because when encoding opcode byte we get the first 5 bits of prefix byte is `01001`, and get R: `0`, X: `1`, B: `0` in latest steps
- the prefix byte is `01001010` (base2) ⇒ `4AH`

d. Encoding the displacement bytes

- $-0xf00 \Rightarrow 0xfffff100$
- so the displacement bytes is ff ff f1 00
- because the code must uses the GCC (i.e., AT&T) assembly syntax in this report, the displacement bytes need to be transfer into Little Endian: 00 f1 ff ff

3. According to the previous steps, the 64-bit instruction encoding is



4A 03 AC 60 00 F1 FF FF (base 16)

3. Disassemble the following IA32 assembly program by hand and recover the C function that performs the same task.

Disassemble the following IA32 assembly program by hand and recover the C function that performs the same task.

```

.text
.globl foo
.type foo, @function
foo:
    pushl %ebp
    movl %esp, %ebp
    pushl %ebx
    subl $4, %esp
    cmpl $-1, 8(%ebp)
    jge .L2
    movl $-1, %eax
    jmp .L3
.L2:
    cmpl $2, 8(%ebp)
    jg .L4
    movl $1, %eax
    jmp .L3
.L4:
    movl 8(%ebp), %eax
    subl $2, %eax
    subl $12, %esp
    pushl %eax
    call foo
    addl $16, %esp
    movl %eax, %ebx
    movl 8(%ebp), %eax
    subl $3, %eax
    subl $12, %esp
    pushl %eax
    call foo
    addl $16, %esp
    addl %ebx, %eax
.L3:
    movl -4(%ebp), %ebx
    leave
    ret

```

solution:

- suppose the name of the function is `func`, the arguments pass to callee `func` is `num1, num2, num3.....` (if need). When the function is called, the assembly program will jump to the label `foo`:

- the following code is the setup process:

```
pushl %ebp
movl %esp, %ebp
pushl %ebx
subl $4, %esp
```

- push the old `%ebp` value to the stack to save it;
- set the new `%ebp` value of callee `func`, it points to the bottom of the callee `func` on stack
- save the `%ebx` value on stack prior to using, because the `%ebx` register is Callee-Save Temporaries
- set the new `%esp` value of callee `func`, it points to the top of the callee `func` on stack

- the following code is related to a “if statement”

```
cmpl $-1, 8(%ebp)
jge .L2
movl $-1, %eax
jmp .L3
```

- `8(%ebp)` get the first argument `num1`, which is passed from caller
- `cmpl` is a compare instruction which compare `-1` and `num1`
- `jge` is a jump instruction, which is used combine with the above `cmpl`, if `num1` greater or equal than `-1` (`num ≥ -1`), the assembly program jump to label `.L2`
 - otherwise(`num < -1`), set `%eax = -1`, then the assembly program jump to label `.L3`
- the Pseudocode is following:

```
if (num1 >= -1){
    //jump to .L2;
}else if(num < -1){
    %eax = -1;//(return -1;)
```

```
//jump to .L3  
}
```

4. similarly, the following code is also related to a “if statement”

```
.L2:  
    cmpl $2, 8(%ebp)  
    jg    .L4  
    movl $1, %eax  
    jmp   .L3
```

- `8(%ebp)` get the first argument `num1`, which is passed from caller
- `cmpl` is a compare instruction which compare `2` and `num1`
- `jg` is a jump instruction, which is used combine with the above `cmpl`, if `num1` greater than `2` (`num1 > 2`), the assembly program jump to label `.L4`
 - otherwise(`num≤2` , also `num≥ -1` because the program jumped from step3), set `%eax = 1` , then the assembly program jump to label `.L3`
- the Pseudocode is following:

```
if(num1 > 2){  
    //jump to .L4  
}else if(num1 <=2 && num1 >=-1){  
    %eax=1; //return 1  
    //jump to .L3;  
}
```

5. the label `.L4` is a complex part which, as caller, call the `func` ,as callee, twice time.

- a. the first call `func(num1 - 2)`

```
movl 8(%ebp), %eax  
subl $2, %eax  
subl $12, %esp  
pushl %eax  
call foo  
addl $16, %esp  
movl %eax, %ebx
```

- the 2 instructions: `movl 8(%ebp), %eax` and `subl $2, %eax` ⇒ set `%eax = num1 - 2`

- the 2 instructions: `subl $12, %esp`, `pushl %eax` is the setup process before call the `func`
- instruction : `call foo`, call the `func(num1-2)`, the return value is in `%eax`
- instruction : `addl $16, %esp`, is the finish process after call the `func(num1-2)`
- instruction: `movl %eax, %ebx` => set `%ebx` equal to the return value of `func(num1 - 2)`

b. the second call `func(num1 - 3)`

```

movl 8(%ebp), %eax
subl $3, %eax
subl $12, %esp
pushl %eax
call foo
addl $16, %esp
addl %ebx, %eax

```

- the 2 instructions: `movl 8(%ebp), %eax` and `subl $3, %eax` ⇒ set `%eax = num1 - 3`
- the 2 instructions: `subl $12, %esp`, `pushl %eax` is the setup process before call the `func`
- instruction : `call foo`, call the `func(num1-3)`, the return value is in `%eax`
- instruction : `addl $16, %esp`, is the finish process after call the `func(num1-3)`
- instruction: `movl %ebx, %eax`, set `%eax = %eax + %ebx`, which is `func(num1-3) + func(num1 - 2)`

c. the Pseudocode is following:

```

%eax = func(num1-3) + func(num1 - 2); // return func(num1-3) + func(num1 - 2);

```

6. the following code is a finish process

```

.L3:
    movl -4(%ebp), %ebx
    leave
    ret

```

- because the return value of `func` is get from `%eax`

- the above code is equal to `return %eax`

7. According to the above analysis, the C function can be recovered as following:



```
int func(int num){  
    if(num < -1){  
        return -1;  
    }else if(num >= -1 && num <=2 ){  
        return 1;  
    }else{  
        return func(num - 2) + func(num-3);  
    }  
}
```