System Software Lab

Department of Computer Science

National Tsing Hua University

User Guide of ARMvisor 1.0

2012

Contents

1 Introduction to ARMvisor

1.1 What is ARMvisor

ARMvisor is a virtual machine on ARM architecture. ARMvisor is based on Kernel-based Virtual Machine. ARMvisor have been developed from 2009. Because ARM architecture doesnt support virtualization in hardware instruction set until 2011, ARMvisor is para-virtualized virtual machine.

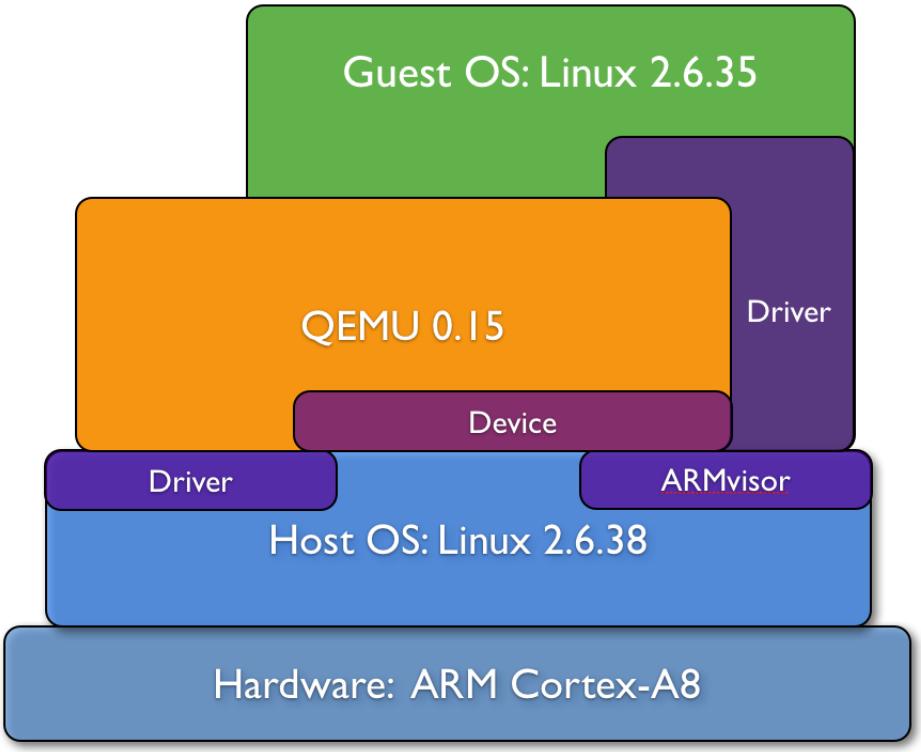

Figure 1: Overview of ARMvisor

In Figure 1, we can observe the architecture of ARMvisor. On the bottom, it is ARM-based hardware. Above the hardware, it's Host OS's kernel which include ARMvisor. Because ARMvisor is a part of Linux kernel, if you want to use ARMvisor, you have to use the kernel with ARMvisor. You can use our pre-compiled kernel image which will be introduced in the following section. If you want to compile the kernel by yourself which will also introduced in another following section.

In the user space of host OS, we need to use QEMU to provide the user to create and manage the VM. Besides, QEMU will also emulate I/O device for

Guest OS. If you want to use ARMvisor, you have to use the QEMU modified by us. You can use our pre-compiled QEMU image which will be introduced in the following section. If you want to compile modified QEMU by yourself which will also introduced in another following section. Otherwise, if you want to boot the Guest OS, you can need to give QEMU some parameters to run QEMU in KVM mode. You can use your own parameters to boot. You can also choose the script file provided by us. We will introduce how to use our script to boot Guest OS in the following section.

In the Guest OS, you need to use the special kernel which patched by us. Because our model is para-virtualization. You need to use patched kernel of the Guest OS. As a result, if you want to use ARMvisor, you have to use the patched kernel of the Guest OS. You can use our pre-compiled patched kernel image of the Guest OS which will be introduced in the following section. If you want to compile the patched kernel of the Guest OS by yourself which will also introduced in another following section.

1.2 How to use ARMvisor

You can run ARMvisor on the most of standard ARMv6/ARMv7 hardware. In following section, we choose QEMU to show that how to use ARMvisor.

2 Use ARMvisor on QEMU

For those who don't have real ARM-based device but still want to use ARMvisor, this section is what you should read!

In this section, we will introduce that how to use QEMU to emulate ARM device on your x86-based PC. Then, we will also show that how to run the pre-compiled host OS on your QEMU on your x86-based PC. Finally, we will introduce that how to put the pre-compiled Guest OS into the root file system of the Host OS and how to run the Guest OS. Just like the Figure 2.

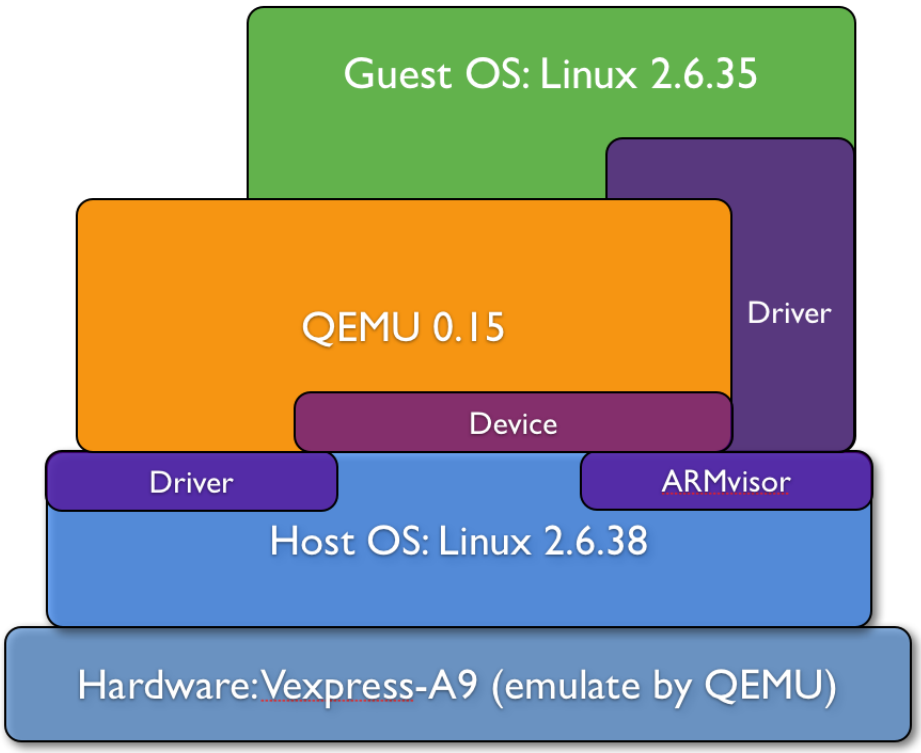

Figure 2: Overview of ARMvisor in the hardware emulated by QEMU

2.1 How to use QEMU to run the Host OS?

QEMU is a tool to emulate ARM device on x86 PC. For those who don't have real ARM-based device but still want to use ARMvisor, QEMU provides you a emulated ARM device running on your PC. In following sections, we will introduce that how to get QEMU and how to use QEMU to run the Host OS.

2.1.1 How to get QEMU to run the Host OS?

If you use Ubuntu/Debian on your PC, you could get QEMU from `apt-get`. If you use Fedora on your PC, you could get QEMU from `yum`. If your package management system doesn't include QEMU, you can also download the source code from QEMU website (http://wiki.qemu.org/Main_Page) and compile it by yourself.

For Ubuntu/Debian user:

```
$ sudo apt-get install qemu-system
```

2.1.2 How to get the kernel image of Host OS?

You can download our pre-compilied kernel image of the Host OS (https://github.com/SSLab-NTHU/linux-host-armvisor/raw/master/arch/arm/boot/zImage).

```
$ mkdir ~/armvisor
```

```
$ mkdir ~/armvisor/kernel-host
```

```
$ cd ~/armvisor/kernel-host
```

```
$ wget http://tinyurl.com/93yrxuj
```

2.1.3 How to get the root file system of Host OS?

We don't provide the root file system of Host OS for you. You may reference Linaro website (https://wiki.linaro.org/Platform/DevPlatform/Rootfs) to try to build your root file system. If you want to build your root file system in the way you want, such as reference the Debian website or Fedora website, it is still available.

2.1.4 How to run the Host OS?

You can boot the Host OS by setting some parameters of QEMU. You can also the booting script provided by us (https://github.com/SSLab-NTHU/boot-host-script/raw/master/run-host.sh) to boot the Host OS.

```
$ cd ~/armvisor/
```

```
$ wget https://github.com/SSLab-NTHU/boot-host-script/raw/master/run-host.sh
```

You could modify the script for fitting your demands.

2.2 Put the Guest OS into the Root File System of the Host OS

Undoubtedly, the Guest OS, which include its kernel and root file system, needs to be put into the Root File System of the Host OS. Besides, because you need to use modified QEMU to create and manage VM, you also need to put the modified QEMU into the Root File System of the Host OS.

There are several way to put the Guest OS and modified QEMU into the Root

File System of the Host OS. You could mount the image file of the Root File System on your PC and put the Guest OS and modified QEMU into the Root File System of the Host OS. You could also use Network File System (a.k.a. NFS) to communicate between the folder on PC and the folder in the Root File System of the Host OS. In the following sections, we will introduce how to get the Guest OS and how to put the Guest OS into the Root File System of the Host OS via NFS.

2.2.1 How to get the Guest OS?

You could download the pre-compiled kernel image of the Guest OS from the website (https://github.com/SSLab-NTHU/linux-guest-armvisor/raw/master/arch/arm/boot/zImage).

```
$ mkdir ~/armvisor/kernel-guest
```

```
$ cd ~/armvisor/kernel-guest
```

```
$ wget http://tinyurl.com/9p52286
```

2.2.2 How to set NFS?

1. Before we set up for NFS, we need to install some packages.

   ```
   $ sudo apt-get install nfs-kernel-server uml-utilities
   ```

2. Download the Script provided by us

   QEMU will create the script file called as “qemu-ifup” and “qemu-ifdown” in `/etc`. When QEMU is executed and want to access network, QEMU will execute this script file. We can use this file to modify the network setting of QEMU.

   You could download script codes provided by us from the website (https://github.com/SSLab-NTHU/qemu-net-script/raw/master/qemu-ifup and https://github.com/SSLab-NTHU/qemu-net-script/raw/master/qemu-ifdown).

   ```
   $ cd ~/armvisor
   ```

   ```
   $ wget http://tinyurl.com/99qf44m
   ```

   ```
   $ wget http://tinyurl.com/8t2sz3y
   ```

You need to change these scripts' mode into executable and use them to replace ordinary one.

```
$ sudo chmod u+x qemu-ifup
```

```
$ sudo chmod u+x qemu-ifdown
```

```
$ sudo mv qemu-ifup /etc/
```

```
$ sudo mv qemu-ifdown /etc/
```

3. Set up the NFS So far, we have everything that we need to use NFS. We just only have to modify some configurations. We will create a folder which can be accessed by NFS server. Meanwhile, we will also set up the network to let that the network subsystem of the Ubuntu on PC can accept the connection from QEMU.

   (a) Create a folder as the NFS folder

   ```
   $ mkdir ~/armvisor/nfs
   ```

   (b) Add the following words into `/etc/exports`

   ```
   /home/username/armvisor/nfs 192.168.16.0/24(rw,async,no_root_squash,no_subtree_check)
   ```

   ```
   /home/username/armvisor/nfs localhost(rw,async,no_root_squash,no_subtree_check)
   ```

   (c) Add the following words into `/etc/hosts.allow` to open the connection for NFS

   ```
   nfsd:ALL portmap:ALL mountd:ALL
   ```

   (d) Let NFS server reload the configuration

   ```
   $ sudo /etc/init.d/portmap restart
   ```

   ```
   $ sudo /etc/init.d/nfs-kernel-server restart
   ```

2.2.3 Put the Guest OS into NFS folder

1. Change directory into the NFS folder

   ```
   $ cd ~/armvisor/nfs
   ```

2. Download the pre-compiled modified QEMU for running the Guest OS

   You can get the pre-compiled modified QEMU from the website (https://github.com/SSLab-NTHU/qemu-guest-armvisor/raw/master/arm-softmmu/qemu-system-arm).

```
$ cd ~/armvisor/nfs
```

```
$ wget http://tinyurl.com/8bhuwtq
```

```
$ mv 8bhuwtq qemu-system-arm
```

3. Download the pre-compiled kernel image of the Guest OS

```
$ cd ~/armvisor/nfs
```

```
$ wget http://tinyurl.com/9p52286
```

```
$ mv 9p52286 zImage-guest
```

4. Write a booting script for running the Guest OS

   Use `vim` to create "run-guest.sh" which include the words in below:

```
sudo qemu-system-arm -M realview-eb -cpu arm1136 -kernel zImage-guest -initrd guestfs.gz
-nographic -enable-kvm -append "console=ttyAMA0"
```

   Then change its mode into executable

```
$ chmod u+x run-guest.sh
```

2.3 Run the Host OS and Guest OS

1. Run the Host OS

```
$ cd ~/armvisor
```

```
$ ./run-host.sh
```

2. Mount NFS folder

   After the host has already been in the shell, you can mount the NFS folder to get the files of the Guest OS

```
$ mount -t nfs 192.168.16.1:/home/username/armvisor/nfs /mnt
```

3. Run the Guest OS

```
$ cd /mnt
```

```
$ ./run-guest.sh
```

3 Build source codes by yourself

3.1 Prerequisites

Before you want to build ARMvisor by yourself, you should make sure that you have all necessary environment.

3.1.1 Software environment

- Toolchain binary files (You can get it from `apt-get`)
- Patched QEMU source code (You can download it from our GitHub site)
- Host OS kernel source code: Patched Linux kernel source code with ARMvisor (You can download it from our GitHub site)
- Guest OS kernel source code: Patched Linux kernel source code (You can download it from our GitHub site)

3.2 How to build ARMvisor by yourself

3.2.1 How to install and use the toolchain

If you use Ubuntu, it's quiet easy to get the toolchain. You just use the `apt-get` to get it.

```
$ apt-get install gcc-arm-linux-gnueabi
```

3.2.2 How to build your Host OS kernel image with ARMvisor from source code

Before you want to build kernel images by yourself, you should make sure that you have already downloaded the patched host kernel source code.

1. Please clone `linux-host-armvisor` by "git". In this document, we will clone the repository in "`~/armvisor/src`" folder.
   (a) Change into the directory
   ```
   $ mkdir ~/armvisor/src
   ```

```
$ cd ~/armvisor/src
```

(b) Clone the source code from GitHub

```
$ git clone https://github.com/SSLab-NTHU/linux-host-armvisor.git
```

2. Change default cross-compiler to the cross-compiler provided by us in Makefile

(a) Open the Makefile

```
$ cd ~/armvisor/src/linux-host-armvisor
```

```
$ vim Makefile
```

(b) Search the keyword: "CROSS_COMPILE"

(c) Change default cross-compiler to "arm-linux-gnueabi"

```
CROSS_COMPILE ?= arm-linux-gnueabi-
```

(d) Save and quit "Makefile"

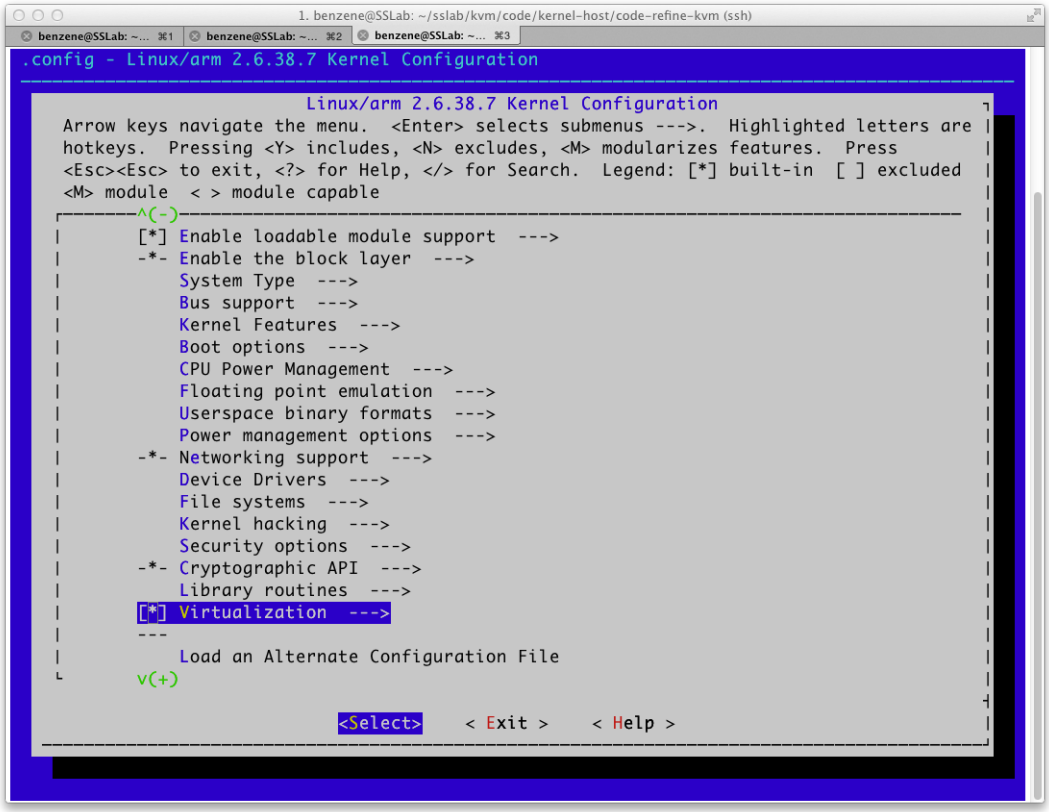

Figure 3: The menuconfig of the Host OS

3. Change directory into linux-host-armvisor, then load default configuration "config-vexpressa9-armvisor". You will get the basic environment setting provided by us for Versitile Express A9. The basic environment setting can let you run Linux kernel with ARMvisor.

   (a) Use default configuration provided by us

   ```
   $ cd ~/armvisor/src/linux-host-armvisor
   ```

   ```
   $ cp config-vexpressa9-armvisor .config
   ```

   (b) If you want to change some configs by yourself, you can `make menuconfig` to modify configs by yourself.

      i. Menuconfig

      In Figure 5, you can see that we have added a new configuration label, called "Virtualization" in menuconfig. We have opened it in the default configuration provided by us. If you want to use ARMvisor, PLEASE DO NOT TURN IT OFF!

ii. Optimization

In Figure 4, you can see that we provide several optimization flag for you. These optimization can provide tremendous speed-up. However, if you open the optmization flag in the kernel of the Host OS, YOU HAVE TO OPEN THE SAME FLAG WHEN COMPILING THE KERNEL OF THE GUEST!

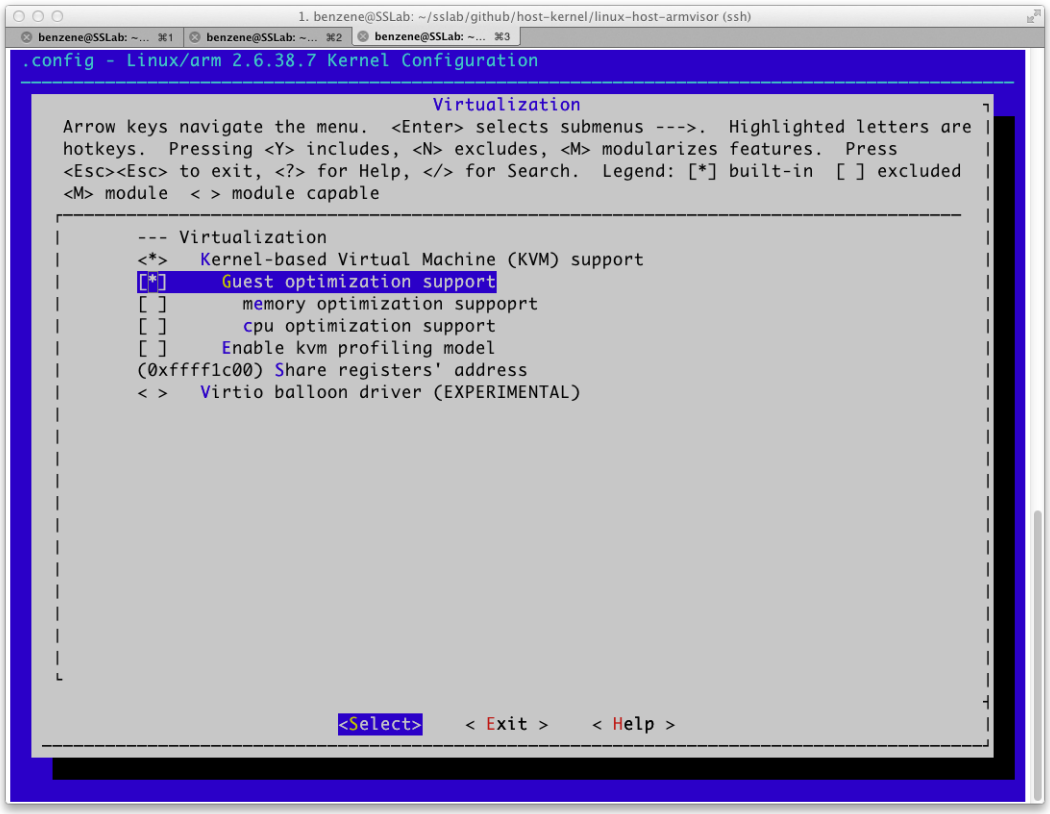

Figure 4: The Optimization configs in menuconfig of the Host OS

(c) Make After "`make`" has been finished, you could find the zImage file in "`arch/arm/boot/zImage`"

```
$ make -j8
```

4. When it is finished, you will find your kernel image in "~/armvisor/src/linux-host-armvisor/arch/arm/boot/zImage"

3.2.3 How to build the QEMU which is for booting the Guest OS?

blah

1. Please clone `linux-guest-armvisor` by "git". In this document, we will clone the repository in "~/armvisor/src" folder.

   (a) Change into the directory

   ```
   $ cd ~/armvisor/src
   ```

   (b) Clone the source code from GitHub

   ```
   $ git clone https://github.com/SSLab-NTHU/qemu-guest-armvisor.git
   ```

2. Compile it by the script provided by us

   (a) Change into the directory

   ```
   $ cd ~/armvisor/src/qemu-guest-armvisor
   ```

   (b) Change the mode of the script file

   ```
   $ chmod u+x configure-kvm.sh
   ```

   (c) Use the script file to set up the configuration

   ```
   $ ./configure-kvm.sh
   ```

   (d) Compile

   ```
   $ make
   ```

   You can find the binary file in the "`arm-softmmu/qemu-system-arm`".

3.2.4 How to build the kernel image of the Guest OS?

1. Please clone `linux-guest-armvisor` by "git". In this document, we will clone the repository in "~/armvisor/src" folder.

   (a) Change into the directory

   ```
   $ cd ~/armvisor/src
   ```

   (b) Clone the source code from GitHub

   ```
   $ git clone https://github.com/SSLab-NTHU/linux-guest-armvisor.git
   ```

2. Change default cross-compiler to the cross-compiler provided by us in Makefile

(a) Open the Makefile

```
$ cd ~/armvisor/src/linux-guest-armvisor
```

```
$ vim Makefile
```

(b) Search the keyword: "CROSS_COMPILE"

(c) Change default cross-compiler to "arm-linux-gnueabi"

```
CROSS_COMPILE ?= arm-linux-gnueabi-
```

(d) Save and quit "Makefile"

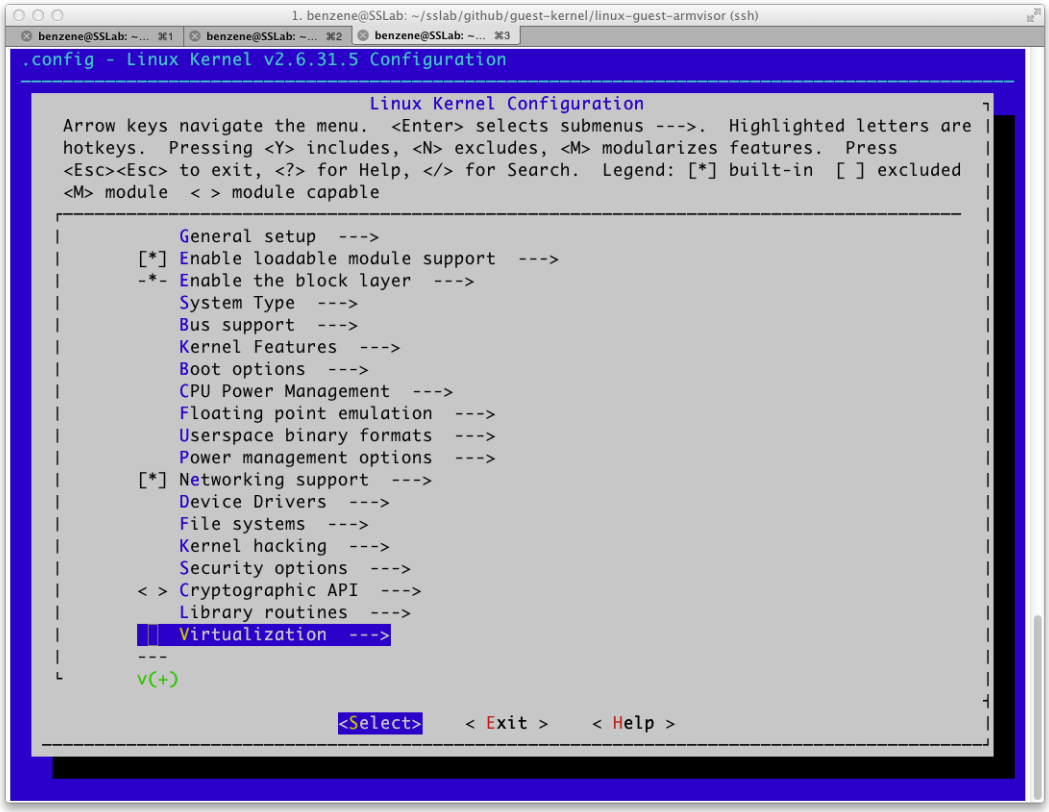

Figure 5: The menuconfig of the Guest OS

3. Change directory into linux-guest-armvisor, then load default configuration "config-guest-armvisor". You will get the basic environment setting provided by us. The basic environment setting can let you run Linux kernel with ARMvisor.

   (a) Use default configuration provided by us

   ```
   $ cd ~/armvisor/src/linux-guest-armvisor
   ```

   ```
   $ cp config-guest-armvisor .config
   ```

   (b) If you want to change some configs by yourself, you can `make menuconfig` to modify configs by yourself.

      i. Menuconfig

      In Figure 5, you can see that we have added a new configuration label, called "Virtualization" in menuconfig. If you want to open the optimization, go into the "Virtualization" label.

      ii. Optimization

      In Figure 6, you can see that we provide several optimization flag for you. These optimization can provide tremendous speed-up. However, if you open the optmization flag in the kernel of the Guest OS, YOU HAVE TO OPEN THE SAME FLAG WHEN COMPILING THE KERNEL OF THE HOST!

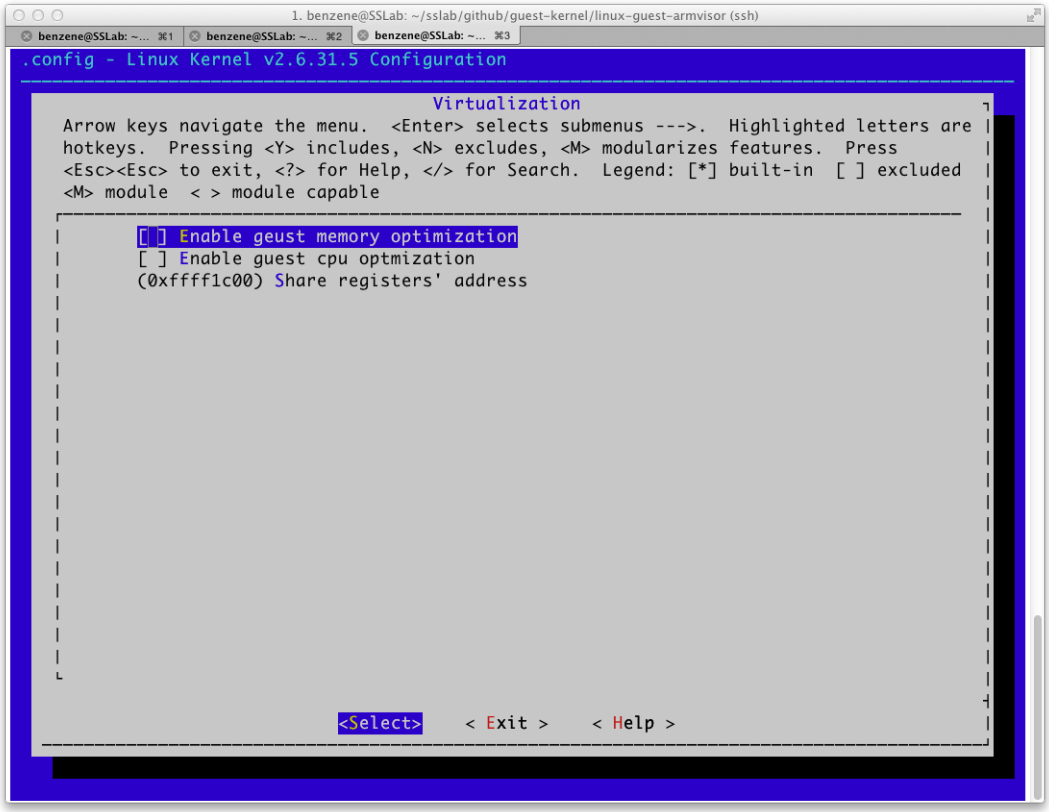

Figure 6: The Optimization configs in menuconfig of the Guest OS

(c) Make After "`make`" has been finished, you could find the zImage file in "`arch/arm/boot/zImage`"

```
$ make -j8
```