Lab #3
Objective: Boot Linux on QEMU ARM emulator
Outcomes:
After this lab, you will be able to
•Build Linux kernel image for ARM versatile board
•Build root file system with Busybox
•Run a user program on QEMU under Linux

Build kernel image
Download latest version of the Linux kernel from www.kernel.org. I use 3.14.31. Try using the stable 4.2.4 version.
In linux-x.x folder, do the following
$ make ARCH=arm versatile_defconfig
$ make ARCH=arm menuconfig (Enable EABI support,)
$ make ARCH=arm CROSS_COMPILE=arm-none-eabi- all
This will start the building of the kernel using the ARM cross compiler; the build will create, among other binaries, a compressed kernel in a file called zImage located in “arch/arm/boot
Build rootfile system with Busybox
Download Busybox from http://www.busybox.net. I use busybox-1.23.1. Try using 1.23.2.
BusyBox combines tiny versions of many common UNIX utilities into a single small executable. BusyBox has been written with size-optimization and limited resources in mind and is easily customizable for embedded systems.
$ tar xjf busybox-1.23.1-tar.bz2
$ cd busybox-1.23.1
$ make ARCH=arm CROSS_COMPILE=arm -linux-gnueabi- defconfig
$ make ARCH=arm CROSS_COMPILE=arm -linux-gnueabi- menuconfig
Check the option to build Busybox as a static executable. The setting can be found in “Busybox Settings --> Build Options
$ make ARCH=arm CROSS_COMPILE=arm -linux-gnueabi- install
$ cd _install
$ mkdir proc sys dev etc etc/init.d
Create a file _install/etc/init.d/rcS with the following content
#!/bin/sh
mount -t proc none /proc
mount -t sysfs none /sys
/sbin/mdev -s
$ chmod +x _install/etc/init.d/rcS
From the _install directory
$ find . | cpio -o --format=newc > ../rootfs.img
$ cd ..
$ gzip -c rootfs.img > rootfs.img.gz
The cpio tool makes a list of files and outputs an archive; the newc format is the format of the initramfs file system that the Linux kernel recognizes.
Note: When the Linux kernel boots the system, it must find and run the first user program, generally called "init". User programs live in filesystems, so the Linux kernel must find and mount the first (or "root") filesystem in order to boot successfully. Ordinarily, available filesystems are listed in the file /etc/fstab so the mount program can find them. But /etc/fstab is itself a file, stored in a filesystem. Finding the very first filesystem is a chicken and egg problem, and to solve it the kernel developers bundle a small ram-based initial root filesystem into the kernel, and if this filesystem contains a program called "/init" the kernel runs that as its first program. After this step a new root filesystem can be mounted from a different device. The previous root (from initrd) is then either moved

to the directory/initrd or it is unmounted.
QEMU passes the file system binary image to the kernel using the initrd parameter.
From the qemu-x.x directory launch qemu
$ ./arm-softmmu/qemu-system-arm -M versatilepb -m 128M -kernel linux-3.14.31/arch/arm/boot/zImage -initrd rootfs.img.gz -append "root=/dev/ram rdinit=/sbin/init"
You should see “#” prompt. The shell can be used normally, for example you can run ls command to find the same directory structure as the Busybox _install directory

To do:
1. Add the following bootup message – Welcome!!! Bienvenidos!!! Bienvenue!!! Willkommen!!!
2.. Examine proc/cpuinfo to verify the processor and board type
3. Execute a dynamically linked C userland code on QEMU that prints “Hello World !”. The executable should reside in /home/”yourname” directory on the emulated machine.