

浙江大学

本科实验报告

课程名称：操作系统

姓 名：徐文皓

学 院：计算机科学与技术学院

系：计算机科学与技术系

专 业：软件工程

学 号：3210102377

指导教师：夏莹杰

2023 年 09 月 23 日

浙江大学操作系统实验报告

实验名称: _____ RV64 环境熟悉 _____

电子邮件地址: _____ 手机: _____

实验地点: _____ 线上 _____ 实验日期: 2023 年 09 月 23 日

一、实验目的和要求

本实验的目的有：了解容器的使用；使用交叉编译工具，完成 Linux 内核代码编译；使用 QEMU 运行内核；熟悉 GDB 和 QEMU 联合调试。

本实验的要求是：独立完成作业；编译内核，使用 QEMU 启动后，远程连接 GDB 进行调试，并尝试使用 GDB 的各项命令；提交实验报告，记录实验过程，对每一步的命令以及结果进行必要的解释，记录遇到的问题和心得体会。

二、实验过程

通过 apt install 命令安装编译内核所需要的交叉编译工具链、用于构建程序的软件包、用于启动 RISCV64 平台上的内核的模拟器 QEMU，以及稍后用于对 QEMU 上运行的 Linux 内核进行调试的 GDB。

```
root@LAPTOP-K817AQPG:~# sudo apt install gcc-riscv64-linux-gnu
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-riscv64-linux-gnu binutils-x86-64-linux-gnu cpp-11-riscv64-linux-gnu cpp-riscv64-linux-gnu
  gcc-11-cross-base-ports gcc-11-riscv64-linux-gnu gcc-11-riscv64-linux-gnu-base gcc-12-cross-base-ports libasan6-riscv64-cross
  libatomic1-riscv64-cross libbinutils libc6-dev-riscv64-cross libc6-riscv64-cross libctf0 libgcc-11-dev-riscv64-cross libgcc-s1-riscv64-cross
  libgomp1-riscv64-cross linux-libc-dev-riscv64-cross
Suggested packages:
  binutils-doc gcc-11-locales cpp-doc gcc-11-doc autoconf automake libtool flex bison gdb-riscv64-linux-gnu gcc-doc
The following NEW packages will be installed:
  binutils binutils-common binutils-riscv64-linux-gnu binutils-x86-64-linux-gnu gcc-11-cross-base-ports gcc-11-riscv64-linux-gnu
  gcc-11-riscv64-linux-gnu-base gcc-12-cross-base-ports gcc-11-riscv64-linux-gnu libasan6-riscv64-cross libatomic1-riscv64-cross
  libc6-dev-riscv64-cross libc6-riscv64-cross libgcc-11-dev-riscv64-cross libgcc-s1-riscv64-cross libgomp1-riscv64-cross
  linux-libc-dev-riscv64-cross
The following packages will be upgraded:
  binutils binutils-common binutils-x86-64-linux-gnu libbinutils libctf0
5 upgraded, 16 newly installed, 0 to remove and 111 not upgraded.
Need to get 32.8 MB/38.2 MB of archives.
After this operation, 112 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

```

root@LAPTOP-K817AQPQ:~# sudo apt install autoconf automake autotools-dev curl libmpc-dev libmpfr-dev libgmp-dev \
    gawk build-essential bison flex texinfo gperf libtool patchutils bc \
    zlib1g-dev libexpat-dev git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'libexpat1-dev' instead of 'libexpat-dev'
The following additional packages will be installed:
  bzip2 dpkg-dev fakeroot libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libauthen-sasl-perl libclone-perl libcurl4
  libdata-dump-perl libdpkg-perl libencode-locale-perl libfakeroot libfile-fcntllock-perl libfile-listing-perl libfl-dev libfl2 libfont-afm-perl
  libgmpxx4ldbl libhtml-form-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl libhttp-cookies-perl
  libhttp-daemon-perl libhttp-date-perl libhttp-message-perl libhttp-negotiate-perl libio-html-perl libio-socket-ssl-perl libltdl-dev libltdl7
  liblwp-mediatypes-perl liblwp-protocol-https-perl libmailtools-perl libnet-http-perl libnet-smtp-ssl-perl libnet-ssleay-perl
  libtext-unidecode-perl libtimedate-perl libtiny-perl liburi-perl libwww-perl libwww-robotrules-perl libxml-libxml-perl
  libxml-namespacesupport-perl libxml-parser-perl libxml-sax-base-perl libxml-sax-expat-perl libxml-sax-perl lto-disabled-list m4
  perl openssl-defaults tex-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  acl adwaita-icon-theme alsa-topology-conf alsa-ucm-conf at-spi2-core dconf-gsettings-backend dconf-service fontconfig glib-networking
  glib-networking-common glib-networking-services gsettings-desktop-schemas gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-x
  gtk-update-icon-cache hicolor-icon-theme humanity-icon-theme ibverbs-providers ipxe-qemu libaa1 libaio1 libasound2 libasound2-data libasyncns0
  libatk-bridge2.0-0 libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libavc1394-0
  libboost-iostreams1.74.0 libboost-thread1.74.0 libbrlapi0.8 libcacao0 libcairo-gobject2 libcairo2 libcdparanoia0 libcolorl2 libcupsp2
  libdat1e1 libdaxctl1 libdconf1 libdecor-0-0 libdecor-0-plugin-1-cairo libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libdv4
  libepoxy0 libfdt1 libflac8 libgbm1 libgdk-pixbuf-2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libgfat10 libgfrpc0 libgfxdr0 libgl1
  libgl1-amd-glx libgl1-mesa-dri libglapi-mesa libglusterfs0 libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libgstreamer-plugins-base1.0-0
  libgstreamer-plugins-good1.0-0 libgtk-3-0 libgtk-3-bin libgtk-3-common libgudev-1.0-0 libharfbuzz0b libibverbs1 libiec61883-0 libiscsi7
  libjack-jackd2-0 liblcms2-2 liblvm15 libmp3lame0 libmpeg123-0 libndctl6 libnl-route-3-200 libnspr4 libnss3 libogg0 libopus0 liborc-0.4-0
  libpango-1.0-0 libpangocairo-1.0-0 libpangotf2-1.0-0 libpciaccess0 libpcsc-lite libpixmap-1-0 libpmmem1 libpmmemobj1 libproxy1v5 libpulse0
  librados2 libraw1394-11 librbd1 librdmacm1 librsync2-2 librsync2-common librsync2-common librsync2-common librsync2-common librsync2-common librsync2-common
  libslirp0 libsndfile1 libsoup2.4-1 libsoup2.4-common libspice1 libspice-server1 libtag1v5 libtag1v5-vanilla libthai-data libthai0 libtheora0
  libtvolame0 liburing2 libusbredirparser1 libv4l-0 libv4lconvert0 libvirglrenderer1 libvisual-0.4-0 libvorbis0a libvorbisenc2 libvpx7
  libvte-2.91-0 libvte-2.91-common libwaypack1 libwayland-client0 libwayland-cursor0 libwayland-egl1 libwayland-server0 libx11-6 libx11-xcb1
  libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-render0 libxcb-shm0 libxcb-sync1 libxcb-xf86vm0 libxcomposite1
  libxcursor1 libxdamage1 libxfixes3 libxi6 libxinerama1 libxkbcommon0 libxrandr2 libxrender1 libxshmfence1 libxss1 libxt6 libxv1 libxv86vm1
  qemu-block-extra qemu-system-common qemu-system-data qemu-system-gui qemu-utils seabios session-migration ubuntu-mono x11-common

```

```


root@LAPTOP-K817AQPQ:~# sudo apt install gdb-multiarch
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  gdb-multiarch

```

从 <https://www.kernel.org> 获得当前最新的 Linux 源码版本(此处为 6.6-rc3)的链接(<https://git.kernel.org/torvalds/t/linux-6.6-rc3.tar.gz>), 通过 `wget` 命令下载至根目录并通过 `tar` 命令进行解压。

The Linux Kernel Archives

[About](#)
[Contact us](#)
[FAQ](#)
[Releases](#)
[Signatures](#)
[Site news](#)



Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	https://rsync.kernel.org/pub/

Latest Release
6.5.5

mainline:	6.6-rc3	2023-09-24	[tarball]	[patch]	[inc. patch]	[view diff]	[browse]
stable:	6.5.5	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
stable:	6.4.16 [EOL]	2023-09-13	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	6.1.55	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	5.15.133	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	5.10.197	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	5.4.257	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.19.295	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
longterm:	4.14.326	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff] [browse] [changelog]
linux-next:	next-20230928	2023-09-28					[browse]

Other resources

[Git Trees](#)
[Patchwork](#)
[Mirrors](#)

[Documentation](#)
[Wikis](#)
[Linux.com](#)

[Kernel Mailing Lists](#)
[Bugzilla](#)
[Linux Foundation](#)

Social

[Site Atom feed](#)
[Releases Atom Feed](#)
[Kernel Planet](#)

```

root@LAPTOP-K817AQP:~# wget https://git.kernel.org/torvalds/t/linux-6.6-rc3.tar.gz
--2023-09-28 23:52:10-- https://git.kernel.org/torvalds/t/linux-6.6-rc3.tar.gz
Resolving git.kernel.org (git.kernel.org)... 145.40.73.55, 43.247.171.1, 96.45.80.1, ...
Connecting to git.kernel.org (git.kernel.org)|145.40.73.55|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/snapshot/linux-6.6-rc3.tar.gz [following]
--2023-09-28 23:52:11-- https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/snapshot/linux-6.6-rc3.tar.gz
Reusing existing connection to git.kernel.org:443.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: 'linux-6.6-rc3.tar.gz'

linux-6.6-rc3.tar.gz      [
<=> ] 218.68M  1.29MB/s   in 2m 48s

2023-09-28 23:55:00 (1.30 MB/s) - 'linux-6.6-rc3.tar.gz' saved [229304334]

```

```

root@LAPTOP-K817AQP:~# tar -xf linux-6.6-rc3.tar

```

使用 Git 工具 clone 实验指导书提供的仓库(https://gitee.com/zju_xiayingjie/os23fall-stu.git), 其中已经准备好了根文件系统的镜像。

```

root@LAPTOP-K817AQP:~# git clone https://gitee.com/zju_xiayingjie/os23fall-stu.git
Cloning into 'os23fall-stu'...
remote: Enumerating objects: 128, done.
remote: Counting objects: 100% (128/128), done.
remote: Compressing objects: 100% (103/103), done.
remote: Total 128 (delta 24), reused 98 (delta 6), pack-reused 0
Receiving objects: 100% (128/128), 1.93 MiB | 984.00 KiB/s, done.
Resolving deltas: 100% (24/24), done.

```

为了方便起见, 将 Linux 源码解压后的文件夹更名为 linux。

进入该文件夹, 通过 make 对 Linux 内核进行编译。指定 RISC-V 架构和 riscv64-linux-gnu-交叉编译工具链, 使用默认配置。

```

root@LAPTOP-K817AQP:~# mv linux-6.6-rc3 linux
root@LAPTOP-K817AQP:~# cd linux
root@LAPTOP-K817AQP:~/linux# make ARCH=riscv CROSS_COMPILE=riscv64-linux-gnu- defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX scripts/kconfig/lexer.lex.c
YACC scripts/kconfig/parser.tab.[ch]
HOSTCC scripts/kconfig/lexer.lex.o
HOSTCC scripts/kconfig/menu.o
HOSTCC scripts/kconfig/parser.tab.o
HOSTCC scripts/kconfig/preprocess.o
HOSTCC scripts/kconfig/symbol.o
HOSTCC scripts/kconfig/util.o
HOSTLD scripts/kconfig/conf
*** Default configuration is based on 'defconfig'
#
# configuration written to .config
#

```

```

Kernel: arch/riscv/boot/Image.gz is ready

```

使用 QEMU 运行内核。

```
root@LAPTOP-K817AQP:~/linux# qemu-system-riscv64 -nographic -machine virt -kernel arch/riscv/boot/Image \
-device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" \
-bios default -drive file=/root/os23fall-stu/src/lab0/rootfs.img,format=raw,id=hd0

OpenSBI v0.9

Platform Name      : riscv-virtio,qemu
Platform Features  : timer,mfdeleg
Platform HART Count : 1
Firmware Base      : 0x80000000
Firmware Size      : 100 KB
Runtime SBI Version : 0.2
```

退出 QEMU 后，在当前 Terminal 输入以下命令重新启动它，并暂停 CPU 的执行，提供 tcp::1234 端口用于远程通信。

```
root@LAPTOP-K817AQP:~/linux# qemu-system-riscv64 -nographic -machine virt -kernel arch/riscv/boot/Image \
-device virtio-blk-device,drive=hd0 -append "root=/dev/vda ro console=ttyS0" \
-bios default -drive file=/root/os23fall-stu/src/lab0/rootfs.img,format=raw,id=hd0 -S -s
```

新建一个 Terminal，打开 GDB。

```
root@LAPTOP-K817AQP:~# gdb-multiarch linux/vmlinux
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from linux/vmlinux...
(No debugging symbols found in linux/vmlinux)
(gdb) 
```

使用 GDB 对内核进行远程调试。

```
(gdb) target remote :1234
Remote debugging using :1234
0x0000000000000100 in ?? ()
(gdb) b start_kernel
Breakpoint 1 at 0xffffffff80a006ac
(gdb) continue
Continuing.

Breakpoint 1, 0xffffffff80a006ac in start_kernel ()
(gdb) quit
A debugging session is active.

    Inferior 1 [process 1] will be detached.

Quit anyway? (y or n) y
Detaching from program: /root/linux/vmlinux, process 1
Ending remote debugging.
[Inferior 1 (process 1) detached]
```

在这里，我们尝试使用并解释 GDB 的一些指令。

输入 layout src，可以查看源代码，便于调试理解。

```
(gdb) target remote :1234
Remote debugging using :1234
0x00000000000001000 in ?? ()
(gdb) layout src
```

我们输入 b(break) start_kernel，在函数 void start_kernel(void)处设置断点。输入 continue，程序运行至断点处，可以看到 start_kernel 函数即将执行。

```
init/main.c
866     memblock_free(unknown_options, len);
867 }
868
869 asmlinkage __visible __init __no_sanitize_address __noreturn __no_stack_protector
870 void start_kernel(void)
871 {
872     char *command_line;
873     char *after_dashes;
874
B+> 875     set_task_stack_end_magic(&init_task);
876     smp_setup_processor_id();
877     debug_objects_early_init();
878     init_vmlinux_build_id();
879
880     cgroup_init_early();
881
882     local_irq_disable();
883     early_boot_irqs_disabled = true;
884
885     /*

remote Thread 1.1 In: start_kernel L875 PC: 0xffffffff80a006ac
(gdb) b start_kernel
Breakpoint 1 at 0xffffffff80a006ac: file init/main.c, line 875.
(gdb) continue
Continuing.

Breakpoint 1, start_kernel () at init/main.c:875
(gdb) bt
#0  start_kernel () at init/main.c:875
#1  0xffffffff80001158 in _start_kernel ()
Backtrace stopped: frame did not save the PC
```

此时通过 info breakpoints，我们发现目前设置了一个断点，并且触发了一次。

```
(gdb) info breakpoints
Num      Type             Disp Enb Address            What
1        breakpoint       keep y  0xffffffff80a006ac in start_kernel at init/main.c:875
breakpoint already hit 1 time
```

通过 f(frame)，我们发现当前的函数栈帧位于 start_kernel 处。

```
(gdb) f
#0  start_kernel () at init/main.c:876
```

为了进一步地了解以上指令，我们在源代码中观察到 start_kernel 函数接下来要调用一个名为 set_task_stack_end_magic 的函数，我们在该函数设置一个断

点，并通过 `continue` 运行到此断点。

```
(gdb) b set_task_stack_end_magic
Breakpoint 2 at 0xffffffff8000d6c2: set_task_stack_end_magic. (2 locations)
(gdb) continue
Continuing.

Breakpoint 2, set_task_stack_end_magic (tsk=0xffffffff8140dac0 <init_task>) at ./include/linux/sched/task_stack.h:31
```

此时，输入 `bt(backtrace)`，发现 `start_kernel` 调用了 `set_task_stack_end_magic`，当前栈顶函数为 `set_task_stack_end_magic`。

```
(gdb) bt
#0 set_task_stack_end_magic (tsk=0xffffffff8140dac0 <init_task>) at ./include/linux/sched/task_stack.h:31
#1 0xffffffff80a006d2 in start_kernel () at init/main.c:875
#2 0xffffffff80001158 in _start_kernel ()
```

输入 `finish`，发现 `set_task_stack_end_magic` 被执行完成。

```
(gdb) finish
Run till exit from #0 set_task_stack_end_magic (tsk=0xffffffff8140dac0 <init_task>) at ./include/linux/sched/task_stack.h:31
start_kernel () at init/main.c:876
```

我们经过几次 `n(next)`，运行到 889 行。

此时我们发现 883 行的语句 `early_boot_irqs_disabled = true` 已被执行。这时，如果输入 `display early_boot_irqs_disabled`，查看这一变量的值，可以看到为 `true`。

```
init/main.c
883     early_boot_irqs_disabled = true;
884
885     /*
886      * Interrupts are still disabled. Do necessary setups, then
887      * enable them.
888      */
> 889     boot_cpu_init();
890     page_address_init();
891     pr_notice("%s", linux_banner);
```

```
(gdb) display early_boot_irqs_disabled
3: early_boot_irqs_disabled = true
```

三、讨论和心得

在本次及以后的实验中，我将采用 VS Code 远程连接 WSL: Ubuntu，VS Code 提供了资源管理器，还是比较方便的。

尽管这是我第一次正式接触 Terminal，但过程总体来说还是比较顺利的，假期学习的 Shell 相关知识得到了实践检验。在本次实验中，只是在解压 Linux 源码下载文件时遇到了一些障碍，`tar.gz` 文件是需要进行两次解压的，我查阅相关资料完成了这一操作。

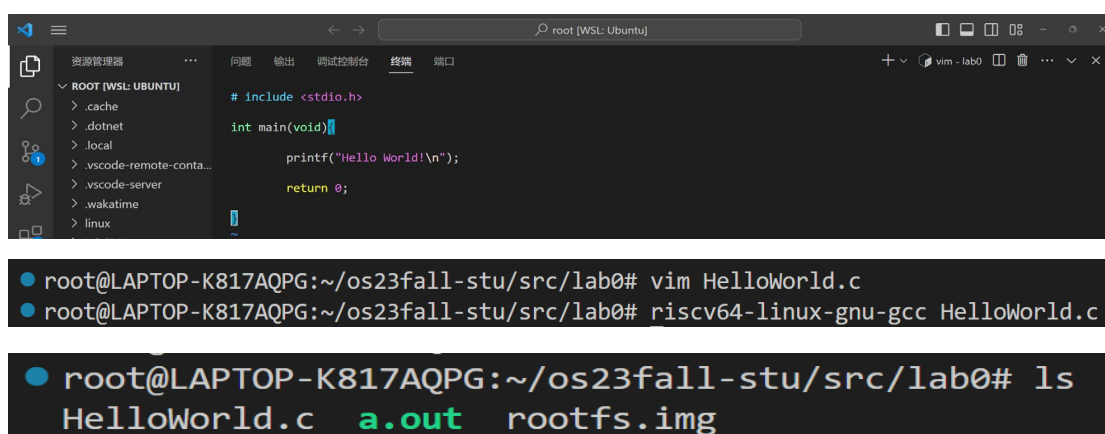
作为软件工程专业的学生，之前在计算机系统原理课程中仅学习了 MIPS，

目前尚未接触过 RISC-V 架构，之后可能要主动对其建立一些了解。

四、思考题

1. 使用 riscv64-linux-gnu-gcc 编译单个 .c 文件

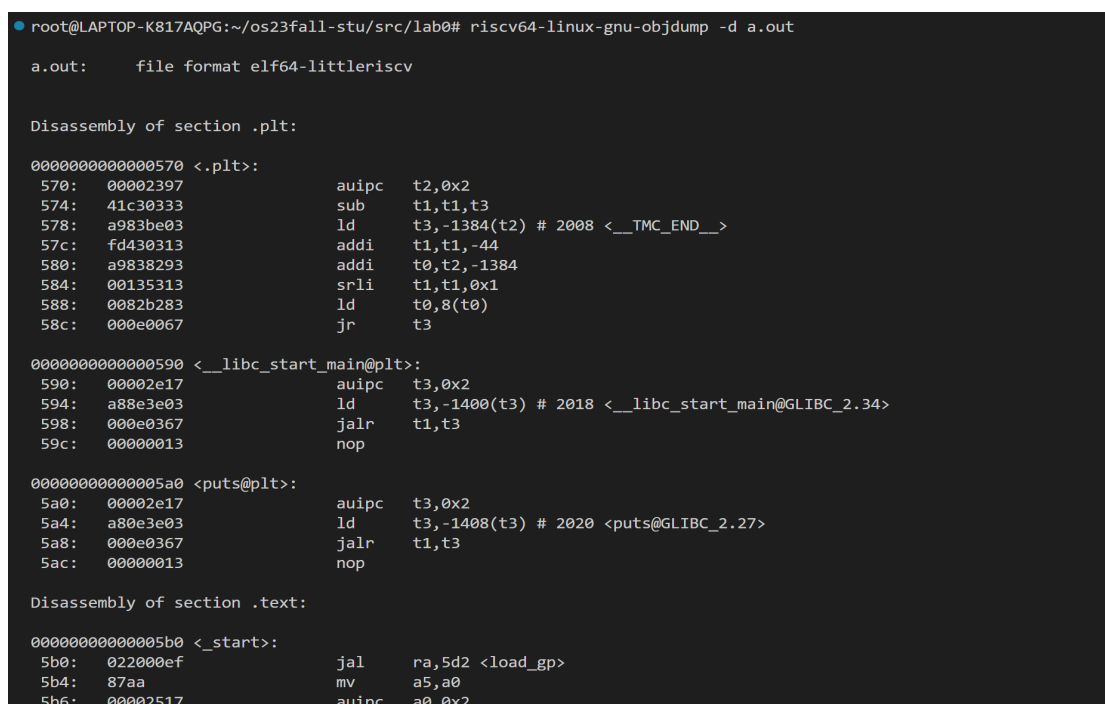
首先我们在 lab0 文件夹内通过 vim 获得一个实现打印 Hello World! 功能的 C 源文件，之后通过 riscv64-linux-gnu-gcc HelloWorld.c 即可编译。使用 ls 命令，我们发现当前文件夹下产生了一个 a.out 文件。



```
root@LAPTOP-K817AQP:~/os23fall-stu/src/lab0# vim HelloWorld.c
root@LAPTOP-K817AQP:~/os23fall-stu/src/lab0# riscv64-linux-gnu-gcc HelloWorld.c
root@LAPTOP-K817AQP:~/os23fall-stu/src/lab0# ls
HelloWorld.c  a.out  rootfs.img
```

2. 使用 riscv64-linux-gnu-objdump 反汇编 1 中得到的编译产物

通过 riscv64-linux-gnu-objdump -d a.out 对刚刚产生的 a.out 文件进行反汇编，我们可以看到其有关的汇编码。



```
root@LAPTOP-K817AQP:~/os23fall-stu/src/lab0# riscv64-linux-gnu-objdump -d a.out

a.out:      file format elf64-littleriscv

Disassembly of section .plt:

000000000000570 <.plt>:
570: 00002397      auipc  t2,0x2
574: 41c30333      sub    t1,t1,t3
578: a983be03      ld     t3,-1384(t2) # 2008 <__TMC_END__>
57c: fd430313      addi   t1,t1,-44
580: a9838293      addi   t0,t2,-1384
584: 00135313      srli   t1,t1,0x1
588: 0082b283      ld     t0,8(t0)
58c: 000e0067      jr     t3

000000000000590 <__libc_start_main@plt>:
590: 00002e17      auipc  t3,0x2
594: a88e3e03      ld     t3,-1400(t3) # 2018 <__libc_start_main@GLIBC_2.34>
598: 000e0367      jalr   t1,t3
59c: 00000013      nop

0000000000005a0 <puts@plt>:
5a0: 00002e17      auipc  t3,0x2
5a4: a80e3e03      ld     t3,-1408(t3) # 2020 <puts@GLIBC_2.27>
5a8: 000e0367      jalr   t1,t3
5ac: 00000013      nop

Disassembly of section .text:

0000000000005b0 <_start>:
5b0: 022000ef      jal    ra,5d2 <load_gp>
5b4: 87aa         mv     a5,a0
5b6: 00002517      auipc  a0,0x2
```


3.使用一些 GDB 常用命令对 Linux 进行调试

(1)查看汇编代码

输入 layout asm，结果如图示。

```
> 0x1000 auipc t0,0x0
0x1004 addi a2,t0,40
0x1008 csrr a0,mhartid
0x100c ld a1,32(t0)
0x1010 ld t0,24(t0)
0x1014 jr t0
0x1018 unimp
0x101a .2byte 0x8000
0x101c unimp
0x101e unimp
0x1020 unimp
0x1022 .2byte 0x8700
0x1024 unimp
0x1026 unimp
0x1028 fnmadd.s ft6,ft4,fs4,fs1,unknown
0x102c unimp
0x102e unimp
0x1030 c.slli64 zero
0x1032 unimp
0x1034 unimp

remote Thread 1.1 In: L?? PC: 0x1000
(gdb)
```

(2) 在 0x80000000 处下断点

(3) 查看所有已下的断点

(4) 在 0x80200000 处下断点

(5) 清除 0x80000000 处的断点

```
(gdb) b * 0x80000000
Breakpoint 1 at 0x80000000
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x0000000080000000
(gdb) b * 0x80200000
Breakpoint 2 at 0x80200000
(gdb) delete 1
(gdb)
```

(6) 继续运行直到触发 0x80200000 处的断点

```
B-> 0x80200000 li s4,-13
0x80200002 j 0x802010d0
0x80200005 nop
0x80200008 unimp
0x8020000a addi s0,sp,8
0x8020000c unimp
0x8020000e unimp
0x80200010 sw s0,0(s0)
0x80200012 .4byte 0x157
0x80200016 unimp
0x80200018 unimp
0x8020001a unimp
0x8020001c unimp
0x8020001e unimp
0x80200020 c.slli64 zero
0x80200022 unimp
0x80200024 unimp
0x80200026 unimp
0x80200028 unimp
0x8020002a unimp

remote Thread 1.1 In: L?? PC: 0x80200000
(gdb) continue
Continuing.

Breakpoint 2, 0x0000000080200000 in ?? ()
```

(7) 单步调试一次

```
(gdb) si
0x0000000080200002 in ?? ()
```

(8)退出 QEMU

```
Boot HART ID           : 0
Boot HART Domain       : root
Boot HART ISA           : rv64imafdcsu
Boot HART Features      : scounteren,mcounteren,time
Boot HART PMP Count     : 16
Boot HART PMP Granularity : 4
Boot HART PMP Address Bits: 54
Boot HART MHPM Count    : 0
Boot HART MHPM Count    : 0
Boot HART MIDELEG       : 0x0000000000000222
Boot HART MEDELEG       : 0x000000000000b109
QEMU: Terminated
root@LAPTOP-K817AQP:~/linux#
```

4. 使用 make 工具清除 Linux 的构建产物

使用 make clean 指令，可以清除以前的 make 命令编译后所产生的 object 文件及其生成的可执行文件。

```
root@LAPTOP-K817AQP:~/linux# make clean
CLEAN  drivers/firmware/efi/libstub
CLEAN  drivers/gpu/drm/radeon
CLEAN  drivers/scsi
CLEAN  drivers/tty/vt
CLEAN  init
CLEAN  kernel
CLEAN  lib/raid6
CLEAN  lib
CLEAN  security/apparmor
CLEAN  security/selinux
CLEAN  usr
CLEAN  .
CLEAN  modules.builtin modules.builtin.modinfo .vmlinux.export.c
root@LAPTOP-K817AQP:~/linux#
```

5. vmlinux 和 Image 的关系和区别是什么？

二者的关系是，Image 是 Linux 编译时对 vmlinux 经过 OBJCOPY 处理后生成的二进制内核映像。二者的区别是，vmlinux 是 Linux 内核编译出来的原始的 elf 格式内核文件，是一个包含 linux kernel 的静态连结的可执行文件，可以用于定位内核问题，但不能直接引导 Linux 系统启动；而 Image 是可以直接引导 Linux 启动的二进制内核映像。二者都未经过压缩。