Lab 1

思考题

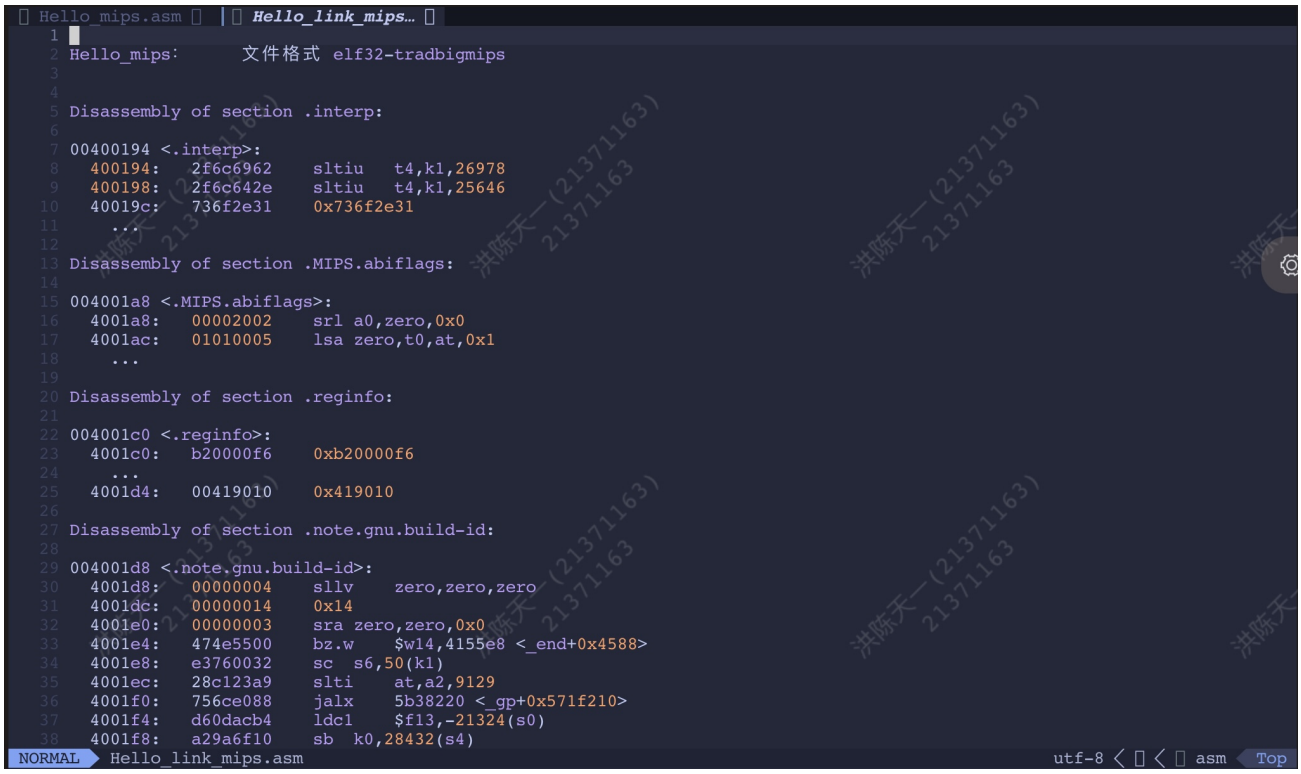
Thinking 1.1

- `-D`：`--disassemble-all`，表示以反汇编的形式显示目标文件的内容。
- `-S`：`--source`，表示在输出文件的反汇编代码之后，同时显示源代码和汇编代码之间的对应关系。

编译与解析过程：

```
$ gcc -E Hello.c > Hello.pre
$ gcc -c Hello.c
$ objdump -DS Hello.o > Hello.asm
$ gcc -o Hello Hello.c
$ objdump -DS Hello > Hello_link.asm
$ mips-linux-gnu-gcc -E Hello_mips.c > Hello_mips.pre
$ mips-linux-gnu-gcc -c Hello_mips.c
$ mips-linux-gnu-objdump -DS Hello_mips.o >Hello_mips.asm
$ mips-linux-gnu-gcc -o Hello_mips Hello_mips.c
$ mips-linux-gnu-objdump -DS Hello_mips >Hello_link_mips.asm
$
```

MIPS 交叉编译工具链结果：



Thinking 1.2

readelf 程序为 64 位，而其本身只能解析 32 位程序。

Thinking 1.3

在MIPS体系结构中，启动入口地址为 0xBFC00000，这是硬件逻辑规定的地址，但这并不意味着内核必须放置在这个地址处才能被正确地启动。实际上，内核可以放置在内存的任何位置，只要在启动过程中正确地设置好启动地址即可。

GXemul 已经提供了 bootloader 的引导（启动）功能。MOS 操作系统不需要再实现 bootloader 的功能。在 MOS 操作系统的运行第一行代码前，我们就已经拥有一个正常的程序运 行环境，内存和一些外围设备都可以正常使用。GXemul 支持加载 ELF 格式内核，所以启动流程被简化为加载内核到内存，之后跳转到内核的入口，启动就完成了。

难点分析

`readelf` 与 `printk` 只要掌握基本的 c 语言知识便较为简单，主要难点在于对操作系统内存位置分配的理解。

实验感想

这次实验中我学到了 `elf` 格式文件的相关知识，也了解了操作系统的内存分配。实现 `printf` 功能的要求，其实我在上学期任程设助教时为了出题就尝试过一些，于是便比较轻松地完成了。（但是当时出的题只有一个同学过了）