

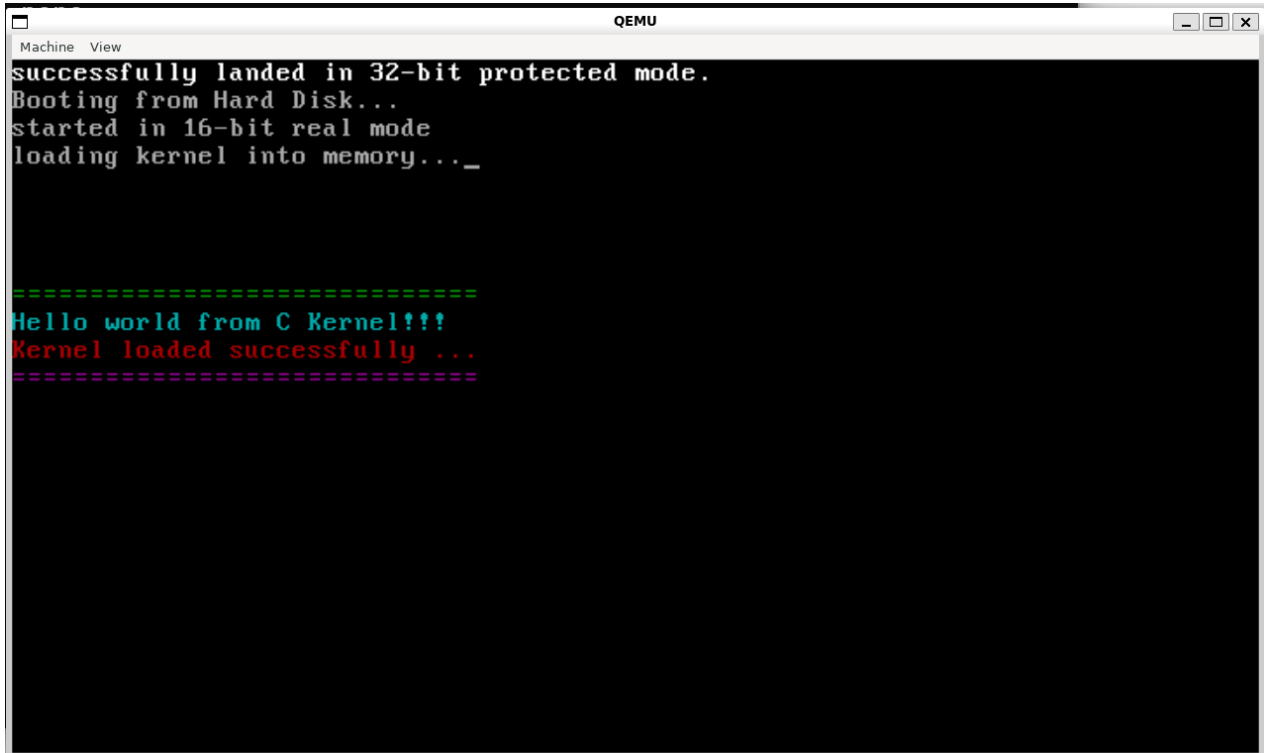
实验一 系统引导 Bootloader

1、实验目的

- 了解操作系统的启动过程（引导程序加载、硬件初始化、操作系统内核启动）
- 学习 BIOS 中断处理机制，掌握中断的设置和处理
- 掌握从磁盘中加载用户程序，理解内存模型
- 掌握全局描述符表（GDT），学会如何设置 GDT
- 了解实模式和保护模式，实模式和保护模式下的寻址方式

2、实验要求

在保护模式下加载磁盘中的 Hello World 程序，并成功运行输出如下界面：



```
Machine View
successfully landed in 32-bit protected mode.
Booting from Hard Disk...
started in 16-bit real mode
loading kernel into memory..._

=====
Hello world from C Kernel!!!
Kernel loaded successfully ...
=====
```

本次实验需要实现以下功能（提供的代码中也有注释）

- 1) 实现 `load_kernel_from_disk` 函数，通过调用 `INT 0x13` 中断，将操作系统内核从硬盘加载到内存中；
- 2) 设置 GDT 代码段、数据段、描述符；
- 3) 实现 `switch_to_pm` 函数（16-bit 实模式），加载 GDT，并通过 GDT 寻址方式调用 `init_pm`；实现 `init_pm` 函数（32-bit 保护模式），设置通用寄存器、段寄存器、指针寄存器等；
- 4) 完成 `enter_kernel.asm` 文件功能，跳转至操作系统 C 内核。

3、实验环境配置指南

第一步：在 Windows 系统中安装 WSL 虚拟机并运行 Ubuntu

1) 打开 windows 系统中的 power shell

```
> wsl --install ubuntu
```

2) 显示安装成功后重启系统，运行 wsl

第二步：在 Ubuntu 中实验环境

1) 安装 dependencies:

```
$ sudo apt-get update
```

```
$ sudo apt-get install make dosfstools mtools binutils nasm git
```

2) 确认 64 位架构的内核:

```
$ dpkg --print-architecture
```

```
amd64
```

然后打开多架构支持:

```
$ dpkg --print-foreign-architectures
```

```
i386
```

如果没有显示 i386, 则需要手动打开:

```
$ sudo dpkg --add-architecture i386
```

```
$ sudo apt-get update
```

```
$ sudo apt-get dist-upgrade
```

3) 安装 gcc multilib:

```
$ sudo apt-get install gcc-multilib g++-multilib
```

4) 安装 qemu 模拟器:

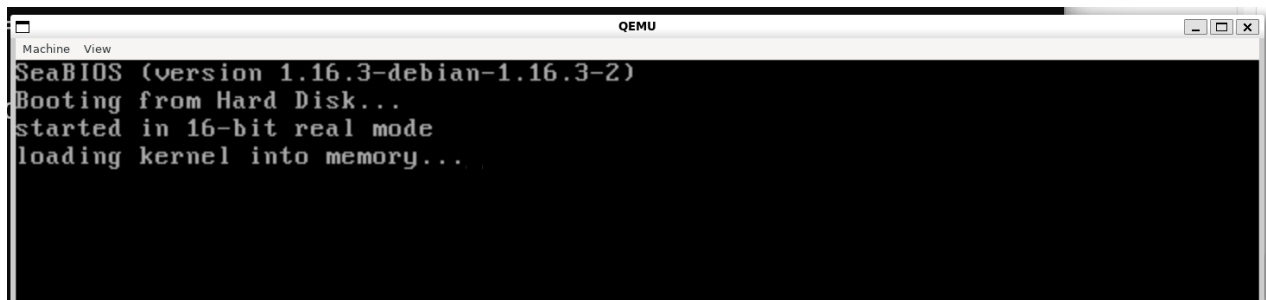
```
$ sudo apt-get install qemu-system-x86
```

第三步：编译实验一代码

1) 解压实验一压缩包，进入目录:

```
$ make
```

此时应该能够正确编译代码并看到以下结果（当然，原始代码只有一个空框架，具体功能需要你来完成）:



2) 在 wsl terminal 中（注意不是 qemu 界面下）按 Ctrl+C 退出 qemu 模拟器

4、代码注释、运行要求

- 代码请给出详细注释，解释每行汇编指令在做什么。如果是多行代码完成一个任务（如先比较寄存器的值再跳转）可适当合并注释。
- 最终提交的代码包含 1) boot.asm 和 2) enter_kernel.asm 两个文件，除此之外请勿提交其他代码。需要保证在所提供的环境中，这两个文件能够正确编译并正确实现实验所要求的功能。
- 打分的时候，助教会把你所提交的以上两个文件复制到初始文件夹下并执行 make 命令，出现内核 Hello World 界面即算通过。

5、实验报告撰写要求

实验报告内容与形式：

- 1) 实验报告中简单阐述如何实现 load_kernel_from_disk, 设置 GDT, switch_to_pm, 加载操作系统内核这四个功能，以及你在实验过程中遇到的问题和思考。
- 2) 请解释所提供的 Makefile 如何完成汇编代码、C 代码的编译和链接，最后生成的二进制镜像文件中有哪些段？各个段的起始偏移量分别是多少。
- 3) 请描述 BIOS 软件中断（例如 INT 0x13）的执行过程。
- 4) 在实验报告最后以附录的形式分别粘贴 boot.asm 和 enter_kernel.asm 代码。**助教会对所有实验报告进行查重，请各组独立完成编码与实验报告。我们对抄袭零容忍。**

6、实验报告与代码提交要求（会影响最后评分，请务必按格式要求提交）

提交内容，只提交三个文件：

- 1) boot.asm 文件
- 2) enter_kernel.asm 文件
- 3) 实验报告 word 文档，命名方式：OSLab1-成员 1 学号姓名-成员 2 学号姓名.docx

例如：OSLab1-B22035678 张三- B22035679 李四.docx

将以上三个文件打成一个压缩包，命名方式：OSLab1-成员 1 学号姓名-成员 2 学号姓名.zip。由班长或学委收集，统一发给任课老师。

7、实验评分标准

给分点	分数
提交的代码能够正确编译，Qemu 中正确出现 Hello World 界面	20%
Part 1 load_kernel_from_disk 功能正确	10%
Part 2 GDT 设置正确	10%
Part 3 switch_to_pm 功能正确	10%
Part 4 enter_kernel.asm 功能正确	10%
代码注释详细、正确	20%
实验报告文档内容详实	20%
总计	100%

参考资料:

- x86 寄存器

<https://www.eecg.utoronto.ca/~amza/www.mindsec.com/files/x86regs.html>

- x86 汇编

<https://arthurchiao.art/blog/x86-asm-guide-zh/>

<https://www.cs.virginia.edu/~evans/cs216/guides/x86.html>

- GDT

<https://blog.csdn.net/abc123lzf/article/details/109289567>

https://wiki.osdev.org/GDT_Tutorial

- INT 0x13 中断

<https://blog.csdn.net/wyyy2088511/article/details/118943195>

[https://wiki.osdev.org/Disk_access_using_the_BIOS_\(INT_13h\)](https://wiki.osdev.org/Disk_access_using_the_BIOS_(INT_13h))