华东师范大学计算机科学与技术学院上机实践报告

课程名称：操作系统	**年级**：2022 级	**上机实践日期**：2024 年 3 月 26 日
指导教师：李东	**姓名**：岳锦鹏	**学号**：10213903403
实验名称：实验一　系统软件启动过程		

一、 完成相关实验内容后，回答以下问题：

1、 为何要开启 A20？uCore OS 是如何开启 A20 的？

如果不开启 A20 会导致地址线第 21 位永远为 0，无法完整进行内存寻址。uCore OS 通过向 8042 键盘控制器的命令端口和数据端口发送命令和数据来开启 A20。

2、 试分析段描述符表（GDT ）的结构，说明段描述符中每个字段的含义以及作用。uCore OS 是如何初始化 GDT 表的？

```
1 |   7    |     6      |     5     |   4    |   3    |   2    |   1    |   0    |
2 |76543210|7 6 5 4 3210|7 65 4 3210|76543210|76543210|76543210|76543210|76543210|
3 |--------|-|-|-|-|---- |-|--|-|----|--------|--------|--------|--------|--------|
4 |  BASE  |G|D|0|A|LIMIT|P|D |S|TYPE|<-------- BASE 23-0 ------>|<-- LIMIT 15-0 ->|
5 |  31-24 | |/| |V|19-16| |P |
6 |        |B| |L|     | |L |
```

- **BASE**: 段基址，由上图中的两部分(BASE 31-24 和 BASE 23-0)组成
- **G**：LIMIT的单位，该位 0 表示单位是字节，1表示单位是 4KB
- **D/B**: 该位为 0 表示这是一个 16 位的段，1 表示这是一个 32 位段
- **AVL**: 该位是用户位，可以被用户自由使用
- **LIMIT**: 段的界限，单位由 G 位决定。数值上（经过单位换算后的值）等于段的长度（字节）- 1。
- **P**: 段存在位，该位为 0 表示该段不存在，为 1 表示存在。
- **DPL**：段权限
- **S**: 该位为 1 表示这是一个数据段或者代码段。为 0 表示这是一个系统段（比如调用门，中断门等）
- **TYPE**: 根据 S 位的结果，再次对段类型进行细分。

uCore OS 初始化了一个空的表项、一个内核代码段、一个内核数据段。

3、 实模式和保护模式有何不同？uCore OS 是如何使能和进入保护模式的？

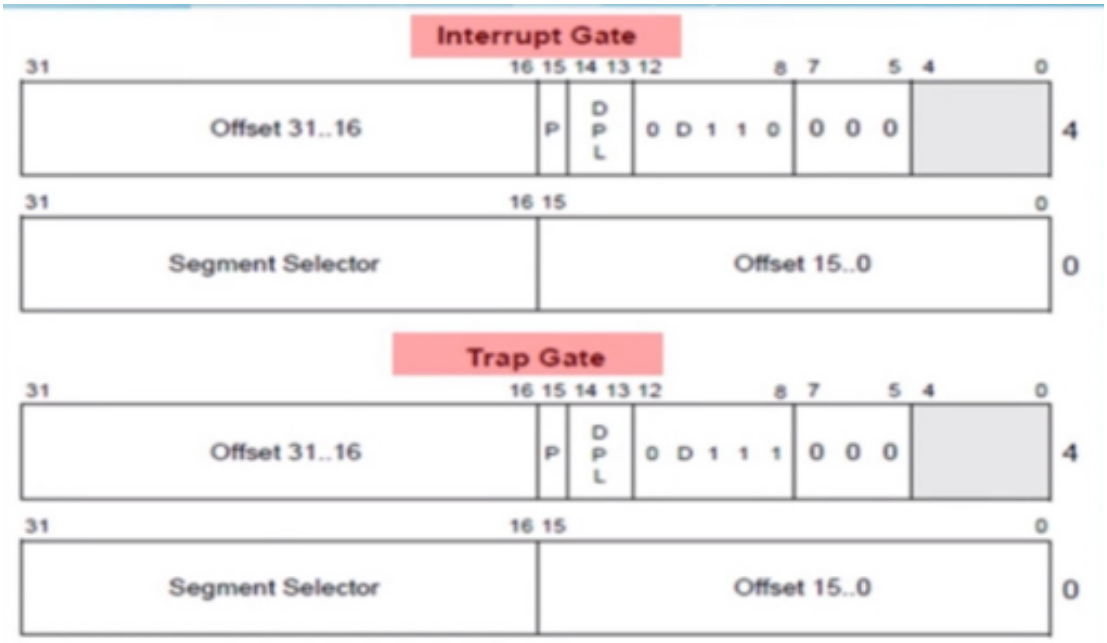
实模式的寻址范围不超过 1M，并且没有分段机制等，而保护模式的寻址范围则没有 1M 的限制，并且可以实现分段机制等。uCore OS 通过把控制寄存器的 0 位设置成 1 来使能和进入保护模式。

4、 Bootloader 是如何利用 ELF 文件头的相关属性加载和运行 uCore OS Kernel 的？

首先判断文件头的 e_magic 是否等于 ELF_MAGIC，之后根据 e_phoff 找到程序头表的位置，在程序头表中找到分段数和各个分段的偏移位置、大小，接着把各个分段加载到内存中，最后跳转到 e_entry 入口开始执行。

5、 分析中断描述符表（IDT）的结构，说明中断描述符中每个字段的含义以及作用。uCore OS 是如何实现中断机制的？

中断描述符表中的每个表项称为中断描述符，它可以确定相应的中断处理程序的位置，中断描述符的结构和每个字段的含义以及作用如下：



	7	6	5	4	3	2	1	0	字节
2	76543210	76543210	7 65 4 3210	76543210	76543210	76543210	76543210	76543210	比特
3			1 -- 0 1110						占位
4	offset in segment		P D S TYPE		segment selector		offset in segment		含义
5	31-16		P				15-0		
6			L						

它确定了段选择子、段内偏移、该段是否已调入内存、该中断的特权级、中断类型（中断门或者陷阱门）等。

uCore OS 将每个中断门都初始化为带中断号参数的调用（最后调用 trap.c 中的内容），之后把系统调用的中断门的特权级设置为用户态，最后加载中断描述符表寄存器。

二、 程序设计与实现的基本思路

1、 大部分实现的功能在 trap.c 文件中，首先从实验视频中可以得知，在触发中断后会进入 trap.c 的执行流程，之后根据不同的中断号执行不同的流程，由于题目要求通过键盘中断实现计时器，因此主要关注键盘中断和时钟中断；

2、 根据题意，在键盘按下 S 时开始，P 暂停，E 停止，C 继续，A 正计时，B 倒计时。那么首先考虑键盘中断，这里很明显很适合用 switch 语句，根据键盘中断后得到的字符不同，执行不同的流程；

3、 这里还需要注意整个功能是存在不同的状态的（其实是因为单线程才会有这么多状态，如果多线程都不需要考虑状态，事件驱动就行），首先很明显可以知道有正在计时状态和停止状态。然后我们考虑这两个状态会在什么时候互相转化，从停止状态转到开始状态可以是按下 S 或者 C，从开始状态转到停止状态可以是按下 P 或 E，或者是倒计时到 0 了。而且题目需要实现正计时和倒计时，这是两种不同的模式，因此还需要一个状态变量记录当前的模式，通过按 A 和 B 切换；

4、 还要注意，当按了 B 后进入倒计时，此时要输入时间，但是这时候还没有 gets 等函数的实现，需要自己通过键盘中断进行输入，那么在键盘中断的时候就不仅需要捕获字母，也需要捕获数字，但我们不希望在输入数字的时候不小心输入了字母就打断了数字输入，这时候就需要增加状态了，需要一个状态表示正在输入数字，在按下 B 后切换到这个状态，当按下 Enter 后切换到计时状态或停止状态；

5、 当倒计时到 0 的时候，应该切换到停止状态，并且还需要输出一条信息表示当前计时到 0 了，但是这里就要注意，从计时状态切换到停止状态时不应该输出这条信息，因此这里需要多加一个中间状态，用来表示倒计时结束的准备输出，这个状态在下一次时钟中断（可以类比数字逻辑电路中，同步时序电路，在时钟有效边沿到来时）转换到停止状态，并且输出这条信息；

6、 关于检测频率，原先的计时器在每次时钟中断时执行 ticks++，而时钟中断频率在 clock.c 文件中设置的是时钟频率除以 100，也就是每 0.01 秒触发一次，尝试过改成除以 1000，但会导致走时变慢，应该是因为触发中断太频繁了时间开销太大，也尝试过改成除以 10，又会导致走时过快，看来还是原始的除以 100 最准时；

7、 既然每 0.01 秒触发一次中断，那么如果是每 TICK_NUM 次中断打印一次时间，直观感受就是精度为 TICK_NUM * 0.01 秒，按照题目要求那么 TICK_NUM 应该取 10，当然为了提升精度也可以取更小，比如 1；

8、 当键入退格时字符为 \b，用于在输入数字时进行退格；当键入回车时字符为 \n，用于确定倒计时的时间；循环打印时间时需要使用 \r，表示将光标回到当前行的开头，这样再打印时间就会把之前的时间覆盖掉；

9、 当然还存在一些 bug，比如从 10.000 倒计时到 9.990 时就会无法覆盖最后一个 0，导致看起来是 9.9900；还有时间精度还是不准的问题；欢迎给出修复建议（可以通过 Issue 的方式，下面会给出网址）。

三、 代码

1、 `https://gitea.shuishan.net.cn/10213903403/os_kernel_lab`

2、 也可以看上传的附件。

注：① 要求实验报告以及代码以附件形式提交。② 实验报告提交的截止期为 4 月 1 日。