南京大学
NANJING UNIVERSITY

Lab1 引导程序

实验目的

- 操作系统的启动过程
- 中断处理机制
- 定时器操作
- 加载用户程序

实验内容

- 在实模式下按一定时间间隔打印Hello World
  - 设置8253/4定时器芯片，每隔20ms左右（自定<1秒）产生一次时钟中断。
  - 通过设置时钟中断处理程序，实模式下在终端中，间隔一定时间打印一行Hello, World!
- 在保护模式下实现一个Hello World程序
  - 从实模式切换至保护模式，并在保护模式下在终端中打印Hello, World!（单次）
- 在保护模式下加载磁盘中的 Hello World 程序并运行
  - 在保护模式下，加载磁盘 1 号扇区中的 "Hello World" 程序到内存指定位置，执行该程序，并在终端输出 "Hello, World!"

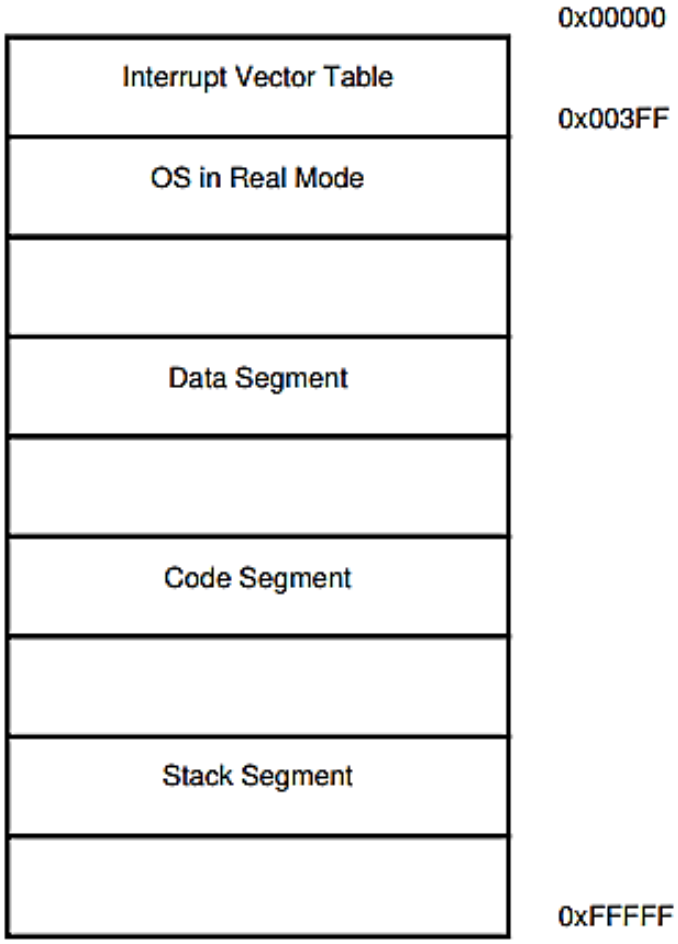
8086简介

- 8086 的寄存器集合
  - 通用寄存器(16 位): AX，BX，CX，DX，SP，BP，DI，SI
  - 段寄存器(16 位): CS，DS，SS，ES
  - 状态和控制寄存器(16 位): FLAGS，IP
- 寻址空间与寻址方式
  - 采用实地址空间进行访存，寻址空间为 2^20
  - 物理地址 = 段寄存器 « 4 + 偏移地址
- CS=0x0000:IP=0x7C00 和 CS=0x0700:IP=0x0C00 以及 CS=0x7C0:IP=0x0000 所寻地址是完全一致的

8086简介

- **8086 的中断**
  - 中断向量表存放在物理内存的开始位置(0x0000至0x03FF)
  - 最多可以有 256 个中断向量
  - 0x00 至 0x07 号中断为系统专用
  - 0x08 至 0x0F，0x70 至 0x77 号硬件中断为 8259A 使用

实模式下的用户程序



内存段的连续性

程序的代码段、数据段和堆栈段在物理内存中必须是连续的

段寄存器的设置

装载程序在加载用户程序时，需要根据程序的具体装载位置设置段寄存器

- CS：指向代码段的起始地址。
- DS：指向数据段的起始地址。
- SS：指向堆栈段的起始地址。

8086简介

- **安全性问题**
  - 程序采用物理地址来实现访存，无法实现对程序的代码和数据的保护
  - 一个程序可以通过改变段寄存器和偏移寄存器访问并修改不属于自己的代码和数据
- **分段机制本身的问题**
  - 段必须是连续的，从而无法利用零碎的空间
  - 段的大小有限制（最大为 64KB），从而限制了代码的规模

相关资料8254定时器芯片

- **8254 定时器芯片（PIT）**
  - 功能：用于生成时钟信号和定时中断
  - 广泛应用于 x86 架构的计算机系统
  - 生成周期性时钟信号、延迟事件，常用于操作系统时间管理、定时器中断等任务
- **工作原理**
  - 8254 定时器通过设置预定的计时值来生成定时中断
  - 可以配置为多种工作模式，如周期性模式、一脉冲模式等
  - 应用领域：操作系统中的时间管理。定时中断、延迟任务等

相关资料8254定时器芯片

- **中断号与中断向量表**
  - 中断号：x86 架构中，中断号是 8 位值，对应中断向量表中的入口地址
  - 定时器中断：中断号 0x1C 对应定时器中断服务程序
- **中断向量表**
  - 存储中断处理程序的入口地址
  - 每个中断号对应一个特定的中断向量
  - 在 8086 处理器中，中断向量表从物理内存地址 0x0000 开始，每个中断号占用 4 字节（2 字节偏移地址，2 字节段基址）

相关资料8254定时器芯片

- **定时器中断与 IRQ0**
  - IRQ0（硬件定时器中断）由 8254 计时器触发。
  - 默认 I/O 端口范围：0x20-0x23，对应中断号 0x08。
  - 软件定时器中断号 0x1C 对应 IVT 地址范围 0x70-0x73。
- **重定向中断**
  - 修改中断向量表的偏移地址和段基址，指向自定义的中断处理程序。

80386 保护模式

- **保护模式带来的变化**
  - 通用寄存器（从 16 位扩展为 32 位）： EAX， EBX， ECX， EDX， ESI， EDI， EBP， ESP
  - 段寄存器（维持 16 位）： CS， DS， SS， ES， FS， GS
  - 状态和控制寄存器（32/64 位）： EFLAGS， EIP， CR0， CR1， CR2， CR3
  - 系统地址寄存器： GDTR， IDTR， TR， LDTR
  - 调试与测试用寄存器： DR0，… ， DR7， TR0， … ，TR7

相关资料 80386保护模式

寄存器类型	8086 寄存器	80386 寄存器
通用寄存器	AX, BX, CX, DX SP, BP, DI, SI	EAX, EBX, ECX, EDX ESI, EDI, EBP, ESP
段寄存器	CS, DS, SS, ES	CS, DS, SS, ES, FS, GS
段描述符寄存器	无	对程序员不可见
状态和控制寄存器	FLAGS, IP	EFLAGS, EIP CR0, CR1, CR2, CR3
系统地址寄存器	无	GDTR, IDTR, TR, LDTR
调试寄存器	无	DR0, ..., DR7
测试寄存器	无	TR0, ..., TR7

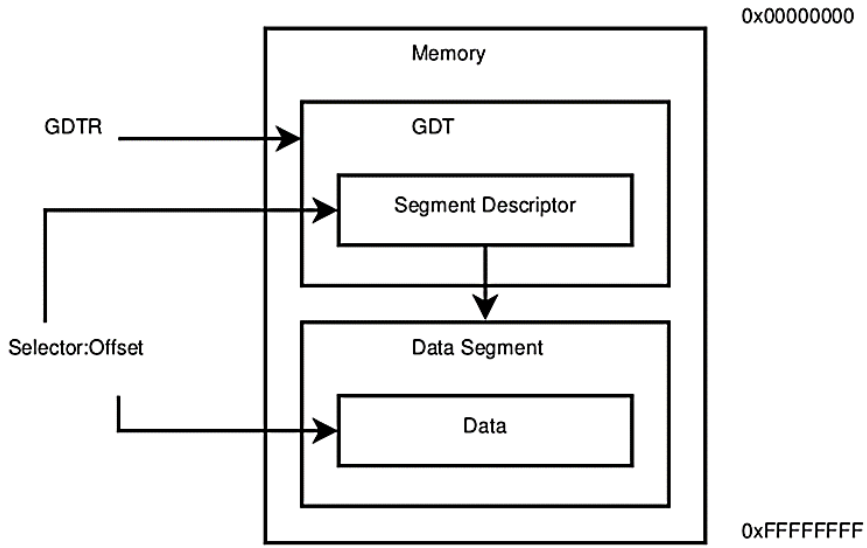
保护模式下的寻址

在保护模式下，分段机制通过**段选择子**和**全局描述符表（GDT）**实现内存寻址

- 段选择子与段描述符
  - **段选择子：** 16 位的值，其中高 13 位表示段描述符在 GDT 中的索引，低 3 位表示权限等信息
  - 通过段选择子，CPU 可以在 GDT 中找到对应的段描述符，其中存储了段的物理首地址。
- 物理地址计算
  - 物理地址 = 段描述符中的物理首地址 + 偏移量（32 位）。
  - 段寄存器
    - 段选择子存储在段寄存器中，如 CS、DS、SS、ES、FS 和 GS。

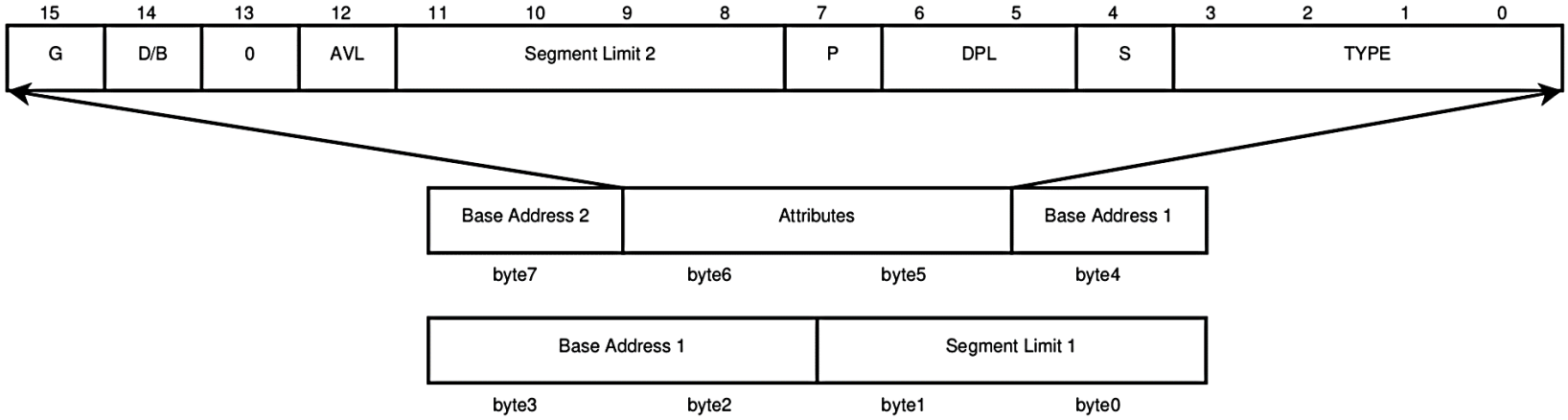
GDT

- **GDT（全局描述符表，Global Descriptor Table）**
  - x86 架构中用于保护模式内存管理的关键数据结构
  - 定义了内存段的属性，包括段的大小、基地址、访问权限等信息
- **GDTR（全局描述符表寄存器），**
  - 80386 及以后的处理器引入了GDTR，用于存储全局描述符表（GDT）在内存中的基地址和表长界限



段描述符

- **段描述符**
  - **大小**：8字节（64位）
  - **段基址**：第 2，3，4，7 字节，共 32 位
  - **段限长**：第 0，1 字节及第 6 字节的低 4 位，共 20 位，表示该段的最大长度
    - G = 0：段限长为实际长度，最大 1MB。
    - G = 1：段限长左移 12 位（×4KB），最大 4GB。



段描述符的属性

段描述符的属性：D/B 位

- D/B 位的含义取决于段类型：
  - **代码段（D 位）：**
    - D = 1：使用 32 位地址和 32/8 位操作数。
    - D = 0：使用 16 位地址和 16/8 位操作数。
  - **数据段（B 位）：**
    - B = 1：段的上界为 4GB。
    - B = 0：段的上界为 64KB。
- **堆栈段（B 位）：**
  - B = 1：使用 32 位操作数，堆栈指针为 ESP。
  - B = 0：使用 16 位操作数，堆栈指针为 SP。

80386保护模式

- **段描述符的属性**
  - **AVL**：保留位，通常设为 0。
  - **P**：存在位，P = 1 表示段在内存中。
  - **DPL**：描述符特权级，取值 0-3，0 为最高特权级，3 为最低特权级，表示访问该段所需的最低 CPU 特权级。
  - **S**：描述符类型标志，S = 1 表示代码段或数据段，S = 0 表示系统段（如 TSS、LDT）或门描述符。

80386保护模式

- **段描述符的属性**
  - TYPE:当 S 为 1，TYPE 表示的代码段，数据段的各种属性如下表所示

bit 3	Data/Code	0 (data)
bit 2	Expand-down	0 (normal) 1 (expand-down)
bit 1	Writable	0 (read-only) 1 (read-write)
bit 0	Accessed	0 (hasn't) 1 (accessed)

bit 3	Data/Code	1 (code)
bit 2	Conforming	0 (non-conforming) 1 (conforming)
bit 1	Readable	0 (no) 1 (readable)
bit 0	Accessed	0 (hasn't) 1 (accessed)

拓展

磁盘分区方式

1. MBR（Master Boot Record）

- **概述**：传统的磁盘分区格式，广泛应用于旧式计算机系统。
- **特点**：
  - **最大磁盘容量**：支持最大 2TB。
  - **分区数限制**：最多支持 4 个主分区（或 3 主分区 + 1 扩展分区）。
  - **引导方式**：通过 BIOS 启动。
- **缺点**：
  - 不支持大于 2TB 的磁盘。
  - 分区数受限。

2. GPT（GUID Partition Table）

- **概述**：现代的磁盘分区标准，主要用于支持 UEFI 启动的系统。
- **特点**：
  - **最大磁盘容量**：理论上支持 9.4ZB，实际可支持超过 2TB。
  - **分区数**：最多支持 128 个分区。
  - **保护机制**：提供分区表备份，提高数据恢复能力。
- **优点**：
  - 支持大于 2TB 的磁盘。
  - 更多的分区和更高的可靠性。
  - 与 UEFI 启动兼容。

拓展 启动方式（BIOS 与 UEFI）

1. BIOS 引导（传统启动）

- **概述**：BIOS 是传统的启动固件，支持 MBR 分区的磁盘。
- **启动过程**：
  - BIOS 进行自检（POST）。
  - 查找启动设备并读取 MBR。
  - 启动引导程序（如 GRUB），加载操作系统。
- **优缺点**：
  - **优点**：兼容性强，广泛支持旧计算机。
  - **缺点**：不支持超过 2TB 的磁盘，分区数目受限。

2. UEFI 引导（现代启动）

- **概述**：UEFI 是现代计算机的启动方式，支持 GPT 分区。
- **启动过程**：
  - UEFI 固件加载并读取 EFI 系统分区（ESP）。
  - 启动管理程序（如 GRUB）加载操作系统。
- **优点**：
  - 支持超过 2TB 的磁盘。
  - 支持更多分区（最多 128 个）。
  - 更快的启动时间与更强的安全性（如 Secure Boot）。

系统的启动

BIOS 与系统启动

1. BIOS 启动

PC 启动时，首先在实模式下运行 BIOS，进行系统自检和初始化。

2. 加载 MBR

BIOS 读取磁盘的 MBR（主引导记录，位于 0 柱面、0 磁头、0 扇区的 512 字节）到内存的 0x7C00 处。

3. 执行 Bootloader

BIOS 执行跳转指令，设置 CS = 0x0000，IP = 0x7C00，开始运行 Bootloader。

系统的启动

- **Bootloader**
  - 代码框架 lab/bootloader/中包含 start_x.s(AT&T 语法的 x86
  - 汇编)以及 boot.c(C 语言)两个源文件
    - start_3.s 作用是将 80386 处理器从实模式切换至 32 位的保护模式，并完成各个段寄存器以及栈顶指针ESP的初始化
    - boot.c 作用是将存储在 MBR 之后的磁盘扇区中的程序加载至内存的特定位置并跳转执行

系统的启动

实模式切换保护模式

1. **关闭中断**：确保在切换过程中不被打断。
2. **启用 A20 地址线**：使得内存可以访问 1MB 以上的地址。
3. **加载 GDTR 寄存器**：设置全局描述符表的基址和限制。
4. **设置 CR0 寄存器的 PE 位**：将 CR0 寄存器的第 0 位（PE 位）设置为 1，表示进入保护模式。
5. **长跳转进入保护模式代码**：由于无法直接修改 CS 寄存器，使用长跳转指令来切换到保护模式。
6. **初始化段寄存器和 ESP**：初始化 DS、SS、ES、FS、GS 等段寄存器，并设置栈顶指针 ESP。

作业提交

- 请仔细阅读lab1的Makefile，不同的指令将会编译出不同的镜像
- 需要你在实验报告中简单阐述这三个功能的实现，以及你在实验过程中的一些思考
- 作业截止时间:2025-3-11 23:59:59