|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 武汉大学国家网络安全学院教学实验报告 | | | | | |
| 课程名称 | 操作系统设计与实践 | | 实验日期 | | 2024.9.16 |
| 实验名称 | 基本实验环境搭建 | | 实验周次 | | 第一周 |
| 姓名 | 学号 | | 专业 | | 班级 |
| 王亚鹏 | 2022302181161 | | 信息安全 | | 5班 |
| 杨依磊 | 2022302181159 | | 信息安全 | | 5班 |
| 杜泓波 | 20223021811 | | 信息安全 | | 5班 |
| 侯名扬 | 20223021811 | | 信息安全 | | 5班 |
| 1. 实验目的及实验内容   （要求掌握的知识；实验内容；原理分析） | | | | | |
| **总目标：搭建实验环境，熟悉开发与调试工具**  **实验内容**：  1. 在实验机上安装虚拟运行环境，并安装ubuntu  2. 下载bochs源码，编译并安装bochs环境  3. 使用bochs自带工具bximage创建虚拟软驱  4. 阅读、编译boot.asm，并反汇编阅读  5. 修改bochsrc，运行并调试你的第一个程序  6. 完成实验练习要求  **原理分析：**  **1. 引导扇区（Boot Sector）的基本概念**  引导扇区（Boot Sector）是磁盘的第一个扇区，通常大小为512字节。在PC机启动时，BIOS会加载指定引导设备的第一个扇区到内存地址0x7C00  **关键点**：   * 引导扇区的大小为512字节。 * 最后两字节必须是 0xAA55，这是引导扇区的标识。如果没有这个签名，BIOS会认为该扇区不是可引导的，无法启动。   **关键指令**：   * jmp $：无限循环，防止程序继续执行非法指令。 * 中断 int 10h：用于在屏幕上显示字符。   **2. 反汇编的过程**  反汇编是将二进制机器代码转换回汇编代码的过程。通过反汇编，可以查看生成的二进制文件（如 boot.bin）的底层汇编指令。使用反汇编指令（如 ndisasm），可以确认生成的指令是否与预期的汇编代码一致。  **3. 虚拟机调试过程**  **调试过程关键步骤**：   1. **设置断点**：在特定地址处设置断点，尤其是在引导扇区开始执行的位置 0x7C00。 2. **单步执行**：使用 s 命令逐步执行指令，观察每条汇编指令的效果。 3. **查看寄存器状态**：使用 r 命令查看寄存器（如 AX, BX, SI）的值，确保数据传输和逻辑正确执行。   **重要调试命令**：   * b 0x7C00：设置断点在 0x7C00 地址。 * c：继续执行代码。 * s：单步执行一条指令。 * r：查看寄存器状态。 | | | | | |
| 1. 实验环境及实验步骤   （本次实验所使用的器件、仪器设备等的情况；具体实验步骤） | | | | | |
| 实验环境：  **X86 32位Linux环境**  实验步骤：  **1. 删除0xAA55，观察程序效果**    **这是删除0xAA55之后程序的运行效果**  可以看到报出了一个错误。经查阅相关资料，程序最后两个字节必须是 0xAA55，这称为引导扇区签名。如果 BIOS 没有在 512 字节扇区的末尾找到这个签名，它将认为这个扇区不是有效的引导扇区，并尝试从其他设备启动。  **可能会观察到以下几种现象**：   1. **BIOS 跳过启动扇区**：BIOS 可能会直接跳过这个扇区，尝试从其他设备启动。如果没有其他有效的引导设备，系统可能会提示类似“**No bootable device found**”的错误。 2. **系统无法启动**：系统可能会进入死循环或停止，因为它没有找到有效的引导设备。 3. **没有输出**：如果没有有效的引导扇区，程序不会运行，也不会显示字符串 "Hello, OS world!"。   这些现象与我们的实际运行结果吻合。  **2. 修改程序中输出为，一个包含自己名字的字符串**    如图所示，红色字体打印出了我的名字Wangyapeng。只需要我们在源代码的字符串部分修改为自己的名字，再把字符个数对应修改即可（下图的红色部分）    **3. 把生成的可执行文件反汇编，查看输出的内容，并在虚拟机启动过程，设置断点进行调试**  首先我们要用nasm boot.asm -o boot.bin生成boot.bin这个文件，生成的 boot.bin 文件是一个原始的二进制文件，仅包含机器码，它会被 BIOS 加载到内存并从地址 0x7C00 开始执行。  经查阅资料可以得知ndisasm指令可以对文件进行反汇编，ndisasm -b 16 boot.bin，输出结果如下：    可以看到，反汇编的结果与我们的汇编源码大致相同，拥有同样的逻辑结构，只是有一些指令并不完全吻合。接下来我们使用bochs调试boot.asm，设置不同的断点，查看执行效果。  **Step\_1：**    设置第一个断点位0x7c00，这正是我们程序开始的地方    接着再输入‘c’回车，可视化界面展现出来。    使用s进行单步调试也可以显示此时每一步的汇编指令，十分清楚方便    也可以使用b命令设置一个稍远的、打印完字符串之后的断点，可以看到红色的字符串已经打印完成。 | | | | | |
| 1. 实验过程分析   （详细记录实验过程中发生的故障和问题，进行故障分析，说明故障排除的过程及方法。根据具体实验，记录、整理相应的数据表格等） | | | | | |
| 故障及其排除方法：  1.在所有步骤无误，进行bochs -f ./bochsrc时，打开调试界面后无缘无故地报出错误：NO DEVICES。之后自动卡退。  解决方案：在本次实验环境中，进行这一步要求具有管理员权限，因此应该在指令前加上sudo，此时再次启动，可以正常运行。(经查询资料大概率是因为没有管理员权限时bochsrc文件中BIOS 和 VGA BIOS 文件的路径在usr文件夹下，需要高级权限查找链接)  2.控制循环100次的实验中错误地使用CX寄存器，导致结构混乱，赋值错误。  解决方案：想当然地使用最常用的循环计数寄存器CX，而没有考虑其是否已经被其他代码段使用。将CX寄存器换为未被使用过的寄存器例如SI，可以解决这类问题。  3.在循环调试过程中只能一直按n命令单步调试，非常麻烦。  解决方案：其实可以通过计算每一条指令的地址，使用b命令设置断点,再使用c命令执行到此处。例如在我的boot\_100.asm中：  **代码起始地址**：   * BIOS 引导程序通常从 0x7C00 地址开始执行，这是引导扇区加载的位置。    **指令长度**：   * mov si, 100：这条指令的长度是 3 字节。 * call DispStr：这是一个调用指令，长度为 3 字节。 * dec si：这条指令长度是 2 字节。 * jnz PrintLoop：这条跳转指令长度是 2 字节。 * jmp $：这条无限循环指令的长度是 2 字节。   这样就可以算出想要到达的位置的地址，方便跳转 | | | | | |
| 1. 实验结果总结   （对实验结果进行分析，完成思考题目，并提出实验的改进意见） | | | | | |
| **结果分析**：  **1. 程序成功启动与引导**   * **实验现象**：你编写的引导扇区程序能够成功在虚拟机中启动。这意味着程序满足了BIOS对于引导扇区的要求：文件大小为512字节，并且末尾包含 0xAA55 作为引导扇区标志。 * **原因分析**：当计算机启动时，BIOS从启动设备的第一个扇区（引导扇区）加载512字节到内存地址 0x7C00，并将控制权交给该地址的代码。程序中的 jmp $ 保证了程序在显示完字符串后不会继续执行无效代码，进入一个安全的无限循环，避免系统崩溃。   **2. 删除 0xAA55 后程序无法正常运行**   * **实验现象**：删除引导扇区末尾的 0xAA55 标志后，BIOS无法识别该扇区为有效引导扇区，导致程序无法正常运行。 * **原因分析**：BIOS依赖引导扇区末尾的 0xAA55 作为合法引导扇区的标志。如果没有这个标志，BIOS会认为该扇区不可引导，并跳过它或尝试从其他设备启动。这说明 0xAA55 对引导扇区的合法性起到了关键作用。   **3. 修改输出字符串为自己的名字**   * **实验现象**：将显示字符串修改为包含自己名字的字符串，程序成功显示出新的内容。 * **原因分析**：通过修改 BootMessage 数据段中的字符串，程序能够通过 int 10h 系统中断成功在屏幕上显示内容。这表明程序正确使用了BIOS提供的视频输出中断，并且数据段和显示逻辑正常工作。   **4. 反汇编生成的二进制文件**   * **实验现象**：使用 ndisasm 反汇编生成的 boot.bin 文件，可以看到对应的汇编指令与原始代码一致，能够验证编译器生成的机器码是正确的。 * **原因分析**：通过反汇编工具，可以将二进制文件中的机器码还原为汇编指令。通过检查反汇编结果，可以确认编译后的代码与原始代码的逻辑一致，确保程序执行符合预期。   **5. 通过修改程序控制输出100次**   * **实验现象**：通过修改程序，控制循环100次 * **原因分析**：通过使用 SI作为循环计数器，结合 loop 或 jnz 指令控制程序的循环次数。计数器每次递减，直到100次后停止。通过运行过程中对SI寄存器值的查看证明了这点   **思考题目：**  **1. 为什么要jmp $，如何改造程序，让这个输出过程执行100次**  jmp $ 是一条无限循环指令。它的作用是阻止程序继续执行其他未定义的代码，确保程序在显示字符串后停止运行，不会执行无效或随机的内存区域。   **防止非法操作**：在汇编语言中，指令一旦执行结束，如果没有明确的跳转或停止指令，CPU 会继续执行内存中的其他代码。如果这些代码是无效的，可能会导致程序崩溃、系统异常甚至重启。因此，jmp $ 会让 CPU 停留在当前地址，确保程序不会进入未知区域。   **保持程序运行**：通常在操作系统或引导程序中，需要让程序在某个稳定状态下停留，而不是直接返回。通过 jmp $，程序就会进入一个死循环，不会退出或继续执行无效指令。  要让字符串的显示过程执行 100 次，**可以在 DispStr 子程序中加入一个循环，控制显示次数（参见boot\_100.asm）**。在 BIOS 中，使用寄存器 SI 可以非常方便地实现循环。我们可以在程序运行过程中使用r命令查看各个寄存器的值，如果SI被赋值为100之后每次运行会减1，那么最终就会循环100次。    如图所示，一开始SI寄存器的值为0（ESI的后16位）    可以看到执行mov si, 100之后，后四位变为0064，在十进制下就是100。    使用n命令执行一次循环，再使用r命令查看寄存器的值，可以看到SI的值为0063，减少了1，与我们的程序符合。这样就可以运行100次循环。  **2. 回答：为什么要对段寄存器进行赋值**  在16位实模式下（如BIOS引导程序所使用的模式），内存地址是通过**段地址:偏移地址**的形式进行访问的。为了能够正确访问内存中的数据和代码，需要使用**段寄存器**来指定内存的段基地址，偏移寄存器来指定段内偏移。因此，在汇编语言编写的引导程序中，必须对**段寄存器（如 DS, ES, SS 等）**进行赋值  **3. 回答：如何在该程序中调用系统中断**  调用系统中断是通过 int 指令来完成的。Boot.asm中使用int 10h调用系统中断显示字符串  **调用系统中断的基本步骤：**   1. **确定要调用的中断号**：中断号决定了调用哪个中断服务例程。例如，BIOS 中断 int 10h 用于视频输出，int 13h 用于磁盘操作。 2. **设置相关寄存器的值**：不同的中断服务例程通过寄存器传递参数。具体的寄存器内容取决于中断的功能。例如，int 10h 中断使用 AH 来指定功能，AL、BH、CX 等寄存器传递额外参数。 3. **执行 int 指令**：通过 int 指令调用中断服务。   **改进意见：**  **1. 增强错误处理机制**   * **现状**：目前的引导扇区代码执行的是简单的输出和无限循环，没有涉及错误处理。 * **改进建议**：可以添加错误处理机制。例如，当调用BIOS中断失败时（如磁盘读取失败），程序应显示错误信息，并提供基本的重试或退出功能。这将帮助你理解如何处理程序异常情况，并增强程序的健壮性。     **2. 增加用户输入功能**   * **现状**：实验中只涉及程序自动执行，没有处理用户输入。 * **改进建议**：可以在程序中增加用户输入功能，通过键盘读取用户输入的字符或命令，并根据输入做出响应。这可以帮助你理解如何使用 BIOS 键盘中断（int 16h）处理用户交互。 | | | | | |
|  | | | | | |
| 1. 各人实验贡献与体会（每人各自撰写） | | | | | |
| 实际上每个人都各自独立地完成了实验  1.王亚鹏：全程参与实验，编写了100次循环的新汇编程序、调试查看循环过程中各个寄存器的值、了解了引导扇区以及BIOS的基本概念、体会了汇编程序在bochs中的调试过程  个人体会：  **1. 深入理解程序执行流程**  理解了 BIOS 的功能：初始化硬件设备、加载启动设备的第一个扇区到内存中的特定位置（0x7C00），并将执行控制交给引导程序。还有对于0xAA55的删除之后程序崩溃让我意识到程序执行的合法区域。删除之后BIOS 无法识别该引导扇区。0xAA55是 BIOS 用于判断一个扇区是否是合法启动扇区的重要依据。  **2. 掌握了bochs下汇编程序的基本调试操作**   * b 0x7C00：设置断点在 0x7C00 地址。 * c：继续执行代码。 * s：单步执行一条指令。 * r：查看寄存器状态。   特别是通过每条汇编指令的大小去计算每条指令的地址，再使用b命令设置断点，这让调试有了很大的进展。  **3. 反汇编的使用**  学会了如何使用 ndisasm 进行反汇编，查看生成的二进制文件的实际内容，确保其与原始汇编代码一致。反汇编的过程让我更加直观地理解了汇编代码与机器码之间的关系，有助于确认代码的正确性。  **4. 实验中的挑战**  实验过程中，设置寄存器和调用中断时出现过一些错误，导致程序无法按预期执行。通过调试工具，修复了错误，例如**CX寄存器多次使用**这一类低级错误困扰了我许久，实在是不应该。这个过程增强了我的调试能力和分析问题的能力，我认为很有帮助。  2.同学：承担了……任务  个人体会：…… | | | | | |
| 1. 教师评语   （实验报告的考评：依据实验内容完整度、实验步骤清晰度、实验结果与分析正确性、实验心得与思考的全面性、实验报告文档的规范性等五个维度综合考评）   |  |  | | --- | --- | | 85-100 | * 实验内容完整或者有超出课程实验大纲的内容； * 实验步骤详尽，能够体现完整的实验过程； * 实验结果正确且实验数据分析得当； * 实验心得与思考全面并且有自己的独立思考； * 实验报告文档规范、排版整齐。 | | 75-84 | * 实验内容较为完整； * 实验步骤较为详尽，能够体现实验过程； * 实验结果正确且实验数据分析较为得当； * 实验心得与思考全面； * 实验报告文档规范、排版较为整齐。 | | 60-74 | * 实验内容有缺失； * 实验步骤不够详尽，不能够体现完整的实验过程； * 实验结果部分正确； * 实验心得与思考无或者不够深入； * 实验报告文档规范性有待增强。 | | 60以下 | * 实验内容严重缺失、实验态度不够端正 * 实验步骤不够详尽，不能够体现完整的实验过程； * 实验结果部分正确； * 实验心得与思考无或者不够深入； * 实验报告文档规范性有待增强。 | | | | | | |
|  | | | | | |
| **教师评分（请填写好姓名、学号）** | | | | | |
| 姓名 | | 学号 | | 分数 | |
| 王亚鹏 | | 2022302181161 | |  | |
| 杨依磊 | | 2022302181159 | |  | |
| 杜泓波 | | 20223021811 | |  | |
| 侯名扬 | | 20223021811 | |  | |
| 教师签名：  年 月 日 | | | | | |