基于X86的小型人机交互操作系统的研究与实践

学 院：信息科学与工程学院

专业班级：电子信息工程（东软大数据特色班）1901班

姓 名：吕恒熙

学 号：20191118015

指导教师：李伟工程师

2023年5月

基于X86的小型人机交互操作系统的研究与实践

**Research and Practice of X86-based Small Human-Computer Interaction Operating System**

## 摘 要

随着计算机技术的不断发展，人机交互操作系统在生活和工作中扮演着越来越重要的角色。然而，当前市场上大部分的人机交互操作系统都较为庞大和复杂，对于一些小型嵌入式系统来说，其资源占用和性能消耗都显得不太适合。因此，需要开发一种基于X86架构的小型人机交互操作系统，以满足这些小型系统的需求。

本研究设计并实现了一款基于X86架构的小型人机交互操作系统。首先，通过对市面上常见的人机交互操作系统进行研究和比较，确定了所需功能和性能指标。然后，采用了一些轻量级的技术和算法，对操作系统进行了设计和开发。最后，通过实际测试和性能评估，验证了该操作系统的稳定性和可靠性，并对其性能进行了优化。

本研究成功地开发了一款基于X86架构的小型人机交互操作系统，具有较小的资源占用和性能消耗，适合于嵌入式系统等小型应用场景。同时，该操作系统还具备较好的用户体验和可扩展性，可以满足用户不同的需求。本研究为小型人机交互操作系统的设计和开发提供了一种新思路，并为该领域的研究和应用做出了一定的贡献。

关键词：X86 操作系统 人机交互 嵌入式

**Abstract**

With the continuous development of computer technology, HCI operating systems are playing an increasingly important role in life and work. However, most of the current HCI operating systems on the market are large and complex, and their resource consumption and performance consumption are not suitable for some small embedded systems. Therefore, there is a need to develop a small HCI operating system based on the X86 architecture to meet the needs of these small systems.

This study designs and implements a small HMI operating system based on the X86 architecture. Firstly, by studying and comparing common HCI operating systems on the market, the required features and performance metrics were determined. Then, some lightweight techniques and algorithms were used to design and develop the operating system. Finally, the stability and reliability of the operating system were verified through practical testing and performance evaluation, and its performance was optimised.

This research has successfully developed a small HMI operating system based on the X86 architecture with a small resource footprint and performance consumption, which is suitable for small application scenarios such as embedded systems. At the same time, the operating system has good user experience and scalability to meet different needs of users. This study provides a new idea for the design and development of small HCI operating systems and contributes to the research and application in this field.

**Keywords:**X86 Operating System Human-Computer Interaction Embedded

目 录

[**摘 要** I](#_Toc135394671)

[**Abstract** II](#_Toc135394672)

[1 绪论 1](#_Toc135394673)

[2 系统分析 2](#_Toc135394674)

[2.1 需求分析 2](#_Toc135394675)

[2.2 可行性分析 2](#_Toc135394676)

[2.2.1 技术可行性 2](#_Toc135394677)

[2.2.2 操作可行性 3](#_Toc135394678)

[2.3 相关技术 3](#_Toc135394679)

[2.3.1 X86汇编 3](#_Toc135394680)

[2.3.2 C语言 3](#_Toc135394681)

[2.3.3 FAT12文件系统格式 4](#_Toc135394682)

[2.3.4 内存分段管理技术 4](#_Toc135394683)

[2.3.5 中断管理技术 5](#_Toc135394684)

[3 系统设计 6](#_Toc135394685)

[3.1 系统结构设计 6](#_Toc135394686)

[3.2 内存管理设计 6](#_Toc135394687)

[3.3 中断管理 9](#_Toc135394688)

[4 系统实现 11](#_Toc135394689)

[4.1 系统引导实现 11](#_Toc135394690)

[4.2 中断管理实现 12](#_Toc135394691)

[4.3 画面显示实现 13](#_Toc135394692)

[4.4 内存管理实现 13](#_Toc135394693)

[5 系统测试 15](#_Toc135394694)

[5.1 系统测试的目的 15](#_Toc135394695)

[5.2 系统测试用例 15](#_Toc135394696)

[5.2.1 内存分配测试 15](#_Toc135394697)

[5.2.2 画面描绘测试 16](#_Toc135394698)

[5.2.3 多任务测试 17](#_Toc135394699)

[5.2.4 中断测试 19](#_Toc135394700)

[6 成果与社会、经济的关系 24](#_Toc135394701)

[结 论 25](#_Toc135394702)

[致 谢 27](#_Toc135394703)

[参考文献 28](#_Toc135394704)

## 绪论

随着计算机技术的不断发展，操作系统作为计算机系统的核心部分，扮演着越来越重要的角色。不仅为用户提供了高效、稳定的计算环境，同时也促进了许多科学技术的发展和创新。为了满足不同用户的需求，现代操作系统的设计趋向于可定制性和可扩展性，允许用户进行自定义配置和扩展编程。在当前的多任务并行处理和虚拟化技术下，一款小型、高效的操作系统对于提高计算机性能和应用效率至关重要。

本文介绍了一款基于X86架构的小型操作系统，旨在提供稳定性、高效性、灵活性和可定制性等多方面的优点。该系统融合了汇编语言和C语言的编程思想，并采用多种算法和图形学原理，支持多种输入输出设备和自定义配置界面。该系统还具有多任务并行和虚拟机技术，可以在桌面和嵌入式系统等多种环境下运行，具有广泛的应用前景。

本文采用了实验方法对该操作系统的性能进行了测试和评估，并将结果进行了分析和比较。实验结果表明，该操作系统具有明显的优点，如高效、稳定、易于使用和可扩展性，可以在多串场合下得到应用。本文可以为操作系统的设计与应用提供有益的参考。

本文将对基于X86的小型人机交互操作系统做系统分析，然后详细描述操作系统的具体实现思路和细节，并且将实现过程中的一些安全模块进行测试，最后介绍一下研究成果和社会经济的关系，并对未来的研究提出一些建议和展望。

## 系统分析

* 1. 需求分析

随着计算机技术的发展，人们对于操作系统的需求越来越多样化和复杂化。在这些需求中，系统引导、图形描绘、内存管理和中断管理是非常重要的四个方面。

系统引导是操作系统启动的第一步，是系统正常运行的基础。在这个方面，我们需要一个快速、稳定、可靠的引导程序。该程序需要具备自检、初始化系统硬件和加载操作系统等功能。

图形描绘是操作系统界面显示的基础，对于用户的视觉体验至关重要。在这个方面，我们需要一个高效、美观、易用的图形描绘系统。该系统需要支持多种分辨率和颜色深度，并具备良好的图形渲染和动画效果。

内存管理是操作系统的核心功能之一，直接影响系统的性能和稳定性。在这个方面，我们需要一个高效、灵活、安全的内存管理系统。该系统需要能够对内存进行分配、释放和管理。

中断管理是操作系统响应外部事件和处理异步任务的基础。在这个方面，我们需要一个高效、可靠、安全的中断管理系统。该系统需要支持多种中断类型和优先级，并能够进行中断处理和嵌套中断的管理。同时，还需要能够进行中断向量的配置和调整，并提供一些常用的中断调试工具。

* 1. 可行性分析
     1. 技术可行性

基于X86的小型人机交互操作系统主要用到的开发语言为C语言和X86汇编。由于选用的平台是Linux，所以我所使用的编辑器是VIM，编译器采用Netwide Assembler和GNU Compiler Collection。在实模式中，操作系统是16位的，在这种情况下，C语言并不能得到很好的运用，采用X86汇编会更加灵活的操控各种寄存器，但是在进入保护模式之后，C语言的使用，可以大大缩短代码的编写难度和编写时间[1]。将编译器GCC的步骤进行分割，把链接器LD分开执行，将X86汇编和C语言代码编译后的目标文件通过LD链接在一起，实现操作寄存器使用汇编，需要处理逻辑使用C语言，有利于将更多的时间投入到实现操作系统功能上[2]。

* + 1. 操作可行性

纵观观所有的操作系统，在在早期IBM PC的时代，也就是20世纪80年代初，由IBM公司设计的BIOS（Basic Input/Output System）中，就规定了操作系统的一些标准[3]，计算机在开机之后，就进入BIOS程序，然后BIOS其中的一项功能就是引导操作系统启动，所以标准在IBM的文档中都可以查询到，在进入操作系统后，如何对硬件信息进行管理，只需要根据个人想法来即可，在现代操作系统中的一些规定，可能存在较重历史包袱，此次设计的操作系统，可以抛弃部分历史包袱，例如来使其更符合使用场景。

* 1. 相关技术
     1. X86汇编

X86汇编技术是一种低级编程语言，它是由英特尔公司发布的X86指令集的汇编版本。X86汇编代码将CPU指令以一种易于读写的形式表达出来，使得开发人员可以直接发出CPU指令，从而能够在更细粒度上控制CPU的操作和访问系统资源[4]。在很多情况下，X86汇编技术可以比高级语言更好地控制CPU和处理器的各种操作，因此在编写一些特定的应用程序时非常有用[5]。

在X86汇编技术中，程序员需要精细地控制CPU寄存器，这些CPU寄存器被用来存储变量和指令的结果。在进行一些高级操作（例如对硬件进行直接访问）时，X86汇编技术的寄存器级别访问可以比更高级别的程序设计语言更高效地访问系统资源，从而大大提高程序的性能。

对于使用X86汇编技术编写的程序，程序员通常需要直接了解CPU的工作原理，了解CPU寄存器的结构和应用程序的操作逻辑。由于X86汇编技术的代码具有高度的优化性和效率，但难以理解和维护，所以今天大多数开发人员使用更高级别的编程语言，并仅在需要直接访问硬件资源时使用X86汇编技术。

* + 1. C语言

C语言是一种高级编程语言，由美国计算机科学家Dennis Ritchie在20世纪70年代初开发。它是一种结构化编程语言，常被用于编写操作系统、编译器、网络协议、设备驱动程序、游戏等方面的软件[6]。C语言的编译器可以在多种操作系统上运行，并且编译出的代码可以在不同的计算机体系结构上运行。因此，C语言在跨平台开发方面具有很大的优势。本次的设计也正是采用了这一点，在linux上进行编译，在x86机器上运行。 编写操作系统时选择C语言的主要原因是因为C语言具有高级语言和低级语言的特点，C语言能够直接操作硬件，以及在低级别的系统编程中操纵内存、寄存器和处理器等底层细节，这使得它成为开发操作系统或设备驱动程序等系统级软件的首选语言[7]。同时，C语言作为一种高级语言，有着相对较高的抽象性 ：C语言具有比较高的抽象性，可以在操作系统设计中使用表达式、语句、条件、循环等高级特性，使操作系统设计更为规范和易于操作。面对嵌入式嵌入式开发环境来说，C语言的语法和标准库相对较小，所以编写的程序执行效率高，对于需要高性能的应用程序而言，C语言是一个理想的选择。

* + 1. FAT12文件系统格式

FAT12是一种文件系统格式，常用于早期的DOS操作系统中。其名称来源于它使用的文件分配表（File Allocation Table）结构和每个条目占用的12位数据。由于FAT12文件系统采用的是簇的概念，因此可以将一个或多个扇区（一般为512字节）组合成一个簇，以便于对磁盘空间的管理和文件的存储。在FAT12文件系统中，每个目录项通常为32个字节，用于存储文件名、扩展名、属性、时间戳等信息。与现代的文件系统相比，FAT12文件系统的优点在于简单易用，占用较少的存储空间。但是，由于其最大支持2.88MB的磁盘容量和较为受限的文件名格式，因此已经逐渐被现代文件系统所替代。但是本次设计是给嵌入式设备使用，往往没有太大的的储存。可以做到更好的节约成本和减小开发难度[8]。

* + 1. 内存分段管理技术

内存分段是一种将内存划分成多个不同大小或用途的区域的技术，常用于操作系统中进行内存管理。每个内存段由一组连续的物理页框组成，并具有唯一的段地址和长度。通过内存分段，内存管理可以更加灵活地控制内存的使用和保护，可以将内存空间划分成不同的区域，并将不同的权限授予不同的进程。这种技术广泛应用于现代操作系统中，可以有效地提高内存管理的效率和安全性[9]。

当应用程序或操作系统请求一段内存时，内存管理器可以从可用的内存中找到一个足够大小的内存段，并将其分配给应用程序或操作系统。内存管理器可以为每个内存段分配不同的访问权限，以防止应用程序或操作系统意外地访问或修改非法内存。

* + 1. 中断管理技术

计算机系统中的中断是一种重要的处理机制，因为它能够迅速地打断正在运行的程序，处理一些及时的任务，然后再返回到先前运行的程序。中断不能仅仅是寄存器值的改变，它涉及到操作系统和硬件之间的协议。中断管理技术是操作系统中很重要的一部分，它通常由中断向量、中断处理程序、中断控制器、中断请求以及中断屏蔽等几个部分共同组成。其中，中断向量表存储了中断处理程序的入口地址，中断处理程序是由操作系统负责编写的，中断控制器是硬件设备，它接收中断请求信号，然后向CPU发送中断信号，CPU控制中断请求的屏蔽和使能。操作系统会定期检查中断请求，如果有新的请求，就会根据中断号获取相应的中断处理程序地址，然后转到该程序执行，完成中断处理后，CPU会返回先前程序的执行。中断请求可以由控制器产生，也可以由用户软件产生。可以在程序中使用软件中断指令（int）来产生中断，这个指令使得CPU转到预定义的中断处理函数中，并向中断处理程序传递参数[10]。

## 系统设计

* 1. 系统结构设计

本次的结构设计是一个线性模型，从计算机的开机一步步进行，主要分为BIOS阶段，系统引导阶段，实模式，保护模式，管理硬件资源等模块，功能结构图如图3-1所示，BIOS由主板厂家提供，并向操作系统提供一些帮助，比如BIOS的其中一项功能是系统引导，读取硬盘的第一个块，这个块包含系统引导内容，让操作系统在这个阶段读取到内存当中，来实现操作系统的启动[11]。

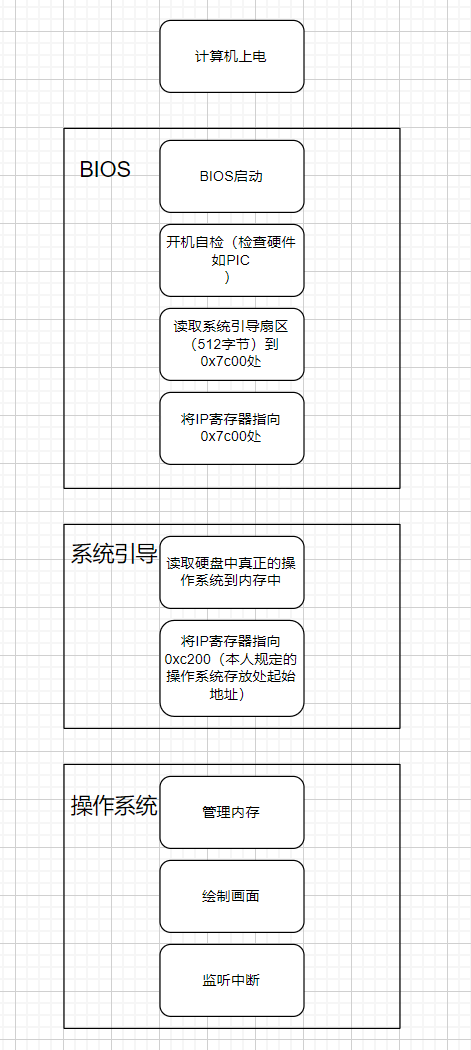


图3-1 系统功能结构图

* 1. 内存管理设计

内存管理模块可以说是整个操作系统最核心的模块之一，我在这里直接采取了分段式内存管理，比起分页式，分段更适合任务少的嵌入式设备使用，因为在任务比较少的嵌入式设备，使用分页式内存会浪费更多的内存，嵌入式设备本身内存资源较为紧缺，使用分段式无疑是更省内存，更加灵活的选择[12]。

分段式内存管理图可以很好的反应内存在使用中的状态，从视觉上高效合理的对分段式内存管理进行解读，如图3-2所示，刚开机，除了系统占用外，内存中一定存在大块的相连内存。

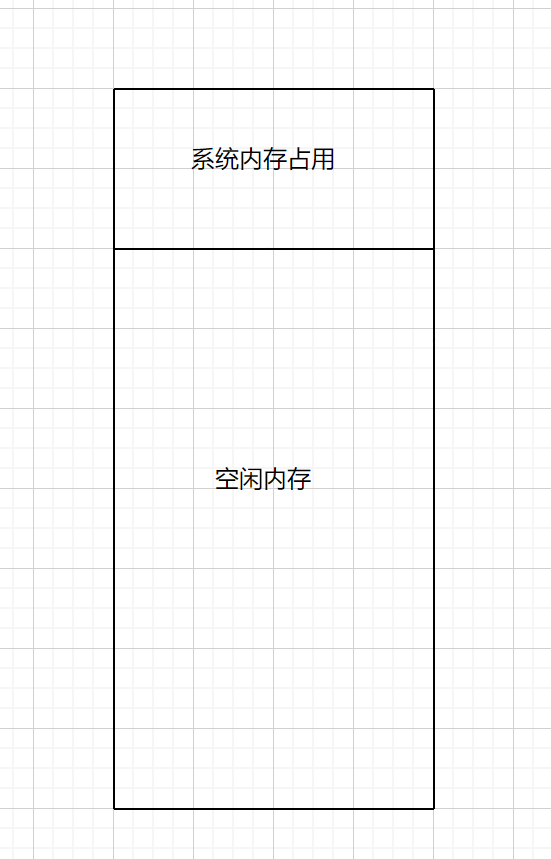


图3-2 刚开机内存占用图

在后续的使用过程中，内存的占用和释放如图3-3所示。假设计算机在刚开机的状态下，连续开三个任务A，B，C，那么其空间占用一定式连续的，当B任务结束之后，任务B原本占用的空间会被释放，标记为空白状态，当继续运行任务D时，会根据任务D的内存占用大小来进行选择，如若任务D内存占用小于等于任务B，则任务D会安排到原任务B的内存位置进行执行。否则，将会在大于等于任务D所需空白空间进行开辟内存空间使得任务D正常运行。

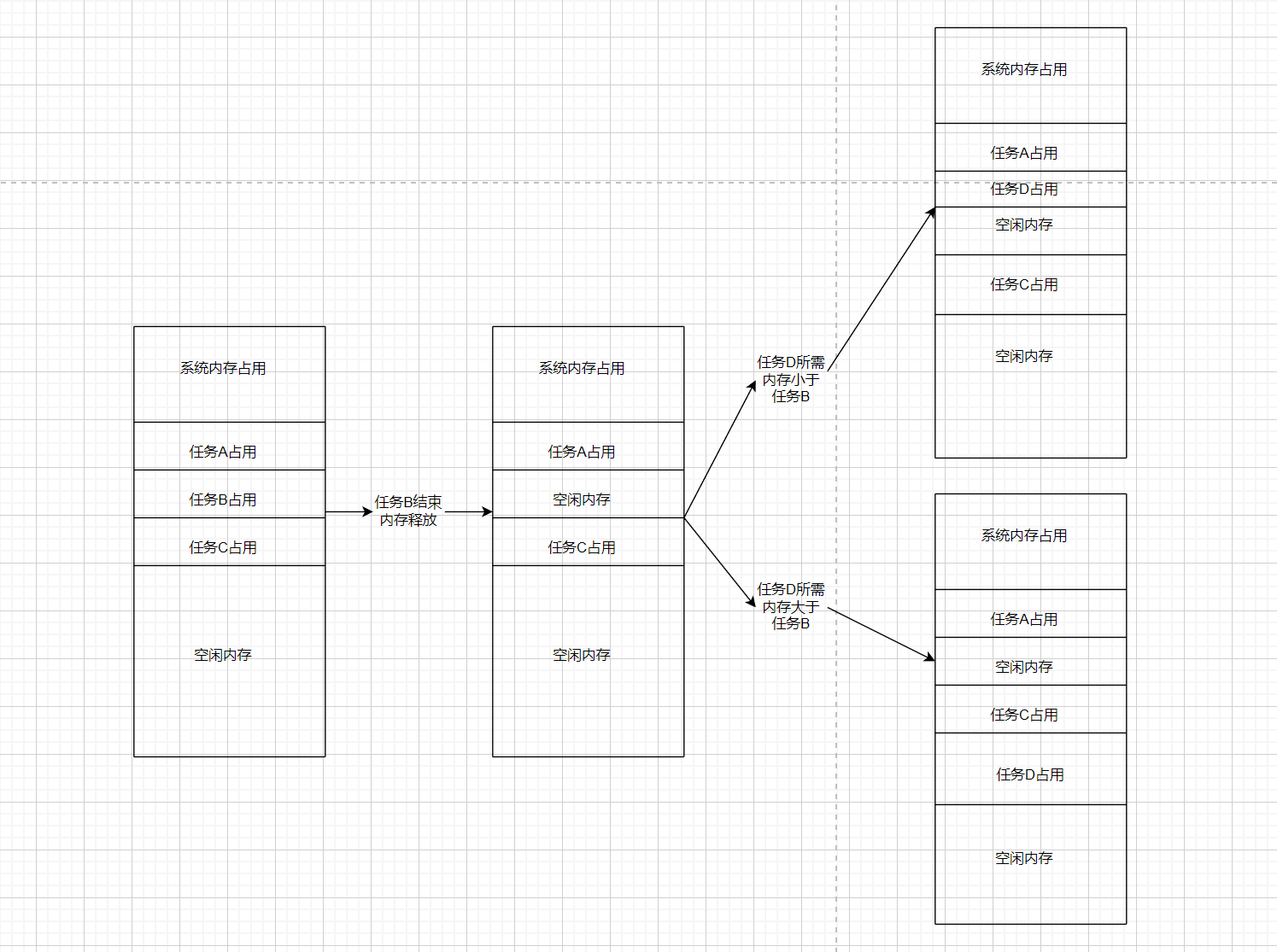


图3-3 内存占用释放图

分段式内存虽然在管理非常简单，但是弊端也很明显，假设系统运行一段时间之后就会产生非常多的碎片信息[13]。如图3-4所示。内存极其碎片，可能内存中所有的空闲内存超过所需内存，但由于过于碎片，导致任务执行失败，这时，操作系统就会执行内存压缩程序。内存压缩是一种处理内存碎片的方法，它通过重新排列已分配内存块来创建更大的连续内存块。找到串联的未分配内存块。这些块可以被合并为一个大块，以释放出更大的连续内存空间。找到所有被占用的内存块，并将它们移动到一个更紧凑的位置。这将创建一个大的连续空间，减少碎片的数量。在图3-4中将展示总空闲内存足够，但连续空闲内存不够的情况将会如何操作。

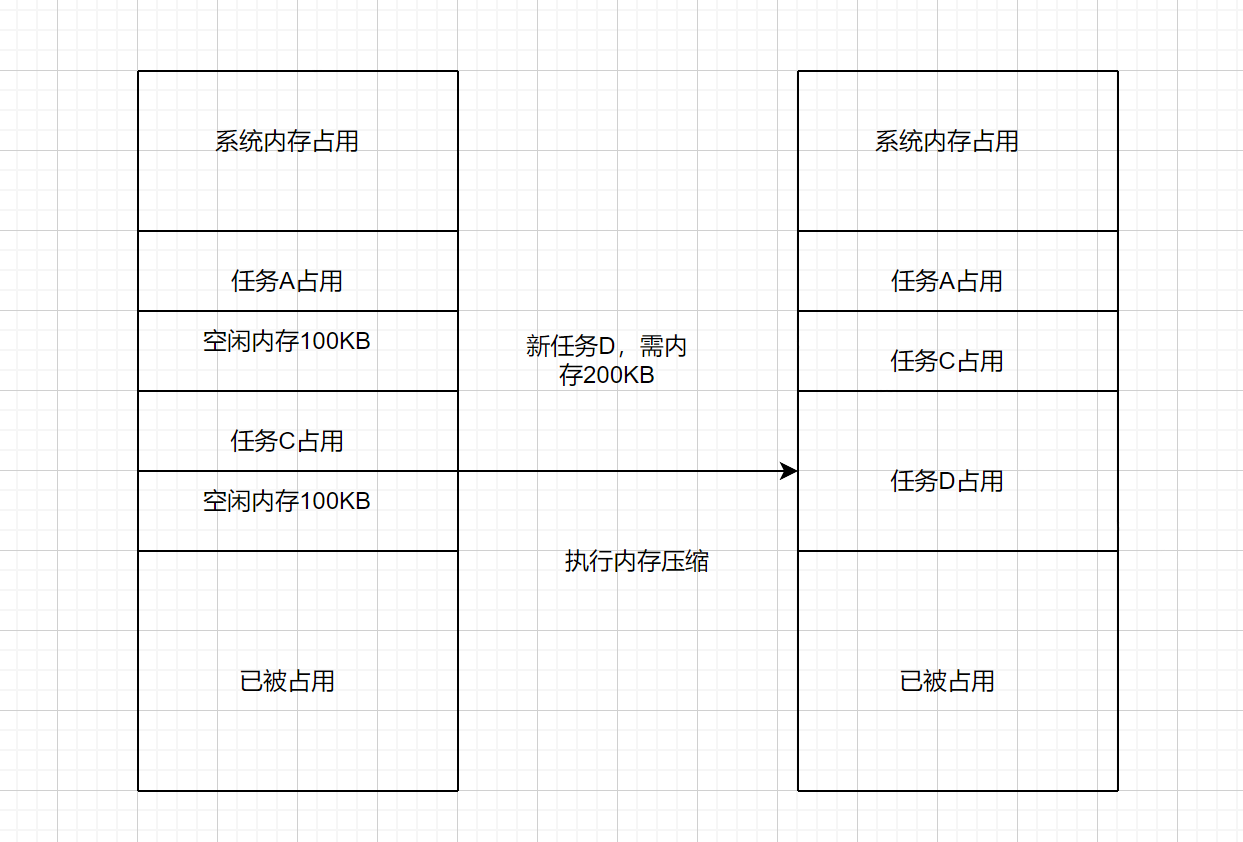


图3-4 内存压缩示意图

* 1. 中断管理

毫不夸张的说，整个操作系统就是建立在中断的基础之上的，如图3-5所示，这是一块计算机主板上的PIC简化图，这张图分为两个中断器，主中断器和从中断器，通过主中断器的端口2相连[14]。

中断管理是指在计算机运行时，发生了不可预测的事件或用户主动触发某些操作时，计算机会发出一个中断信号，暂停当前程序的执行，转而处理中断事件或响应用户的操作请求。这个过程需要中断管理来完成。中断管理由中断处理程序和中断向量表组成。

中断处理程序是一段特殊的程序代码，用于响应中断信号并进行相应的处理。中断向量表是一个由多个中断处理程序组成的表格，每个中断处理程序对应一个中断类型。当一个中断信号触发时，CPU会跳转到对应的中断处理程序，进行相应的处理。

中断管理的作用非常重要，它能够及时响应计算机系统发生的各种事件，包括硬件故障、输入输出请求、操作系统调用等。通过中断管理能够更好地保证计算机系统的可靠性和稳定性。同时，中断管理也能够提高计算机系统的并发性，允许多个程序同时运行，提高系统的工作效率。

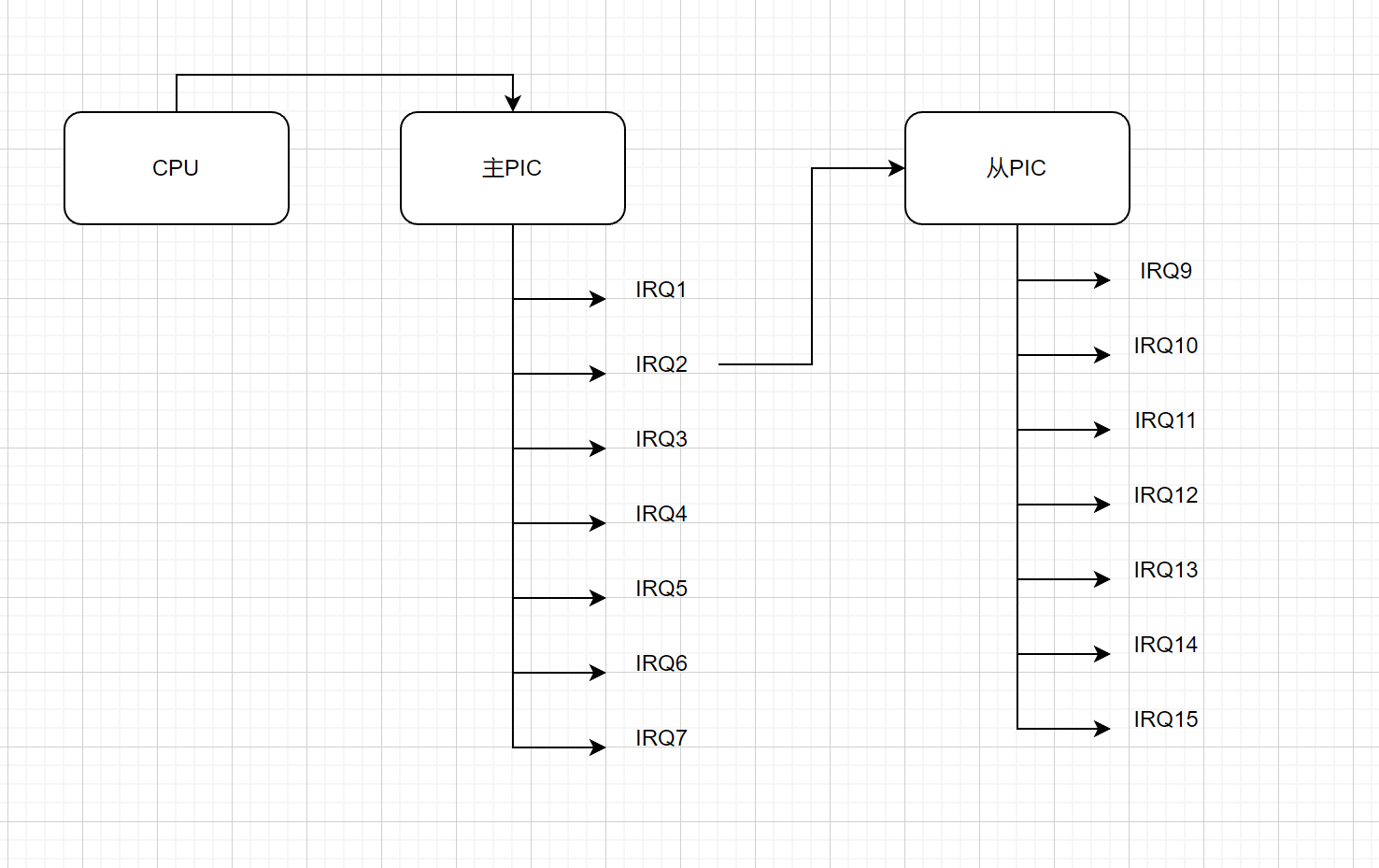


图3-5 中断PIC示意图

## 系统实现

* 1. 系统引导实现

计算机启动时，BIOS（Basic Input/Output System）会先执行硬件检测和初始化程序，然后会找到并加载引导扇区的第一个扇区，执行其中的指令。操作系统的引导就是通过这个扇区来启动的[15]。

在X86架构的计算机中，引导扇区大小为512字节，在这512字节中，有200个字节是引导程序的主体（Boot Loader），其余的是分区表（Partition Table）和引导扇区标识（Boot Signature）。其中，引导程序的主要功能是将操作系统的核心代码从硬盘或其他存储介质中读取到内存中，并且跳转到操作系统的入口地址，让操作系统开始执行。IPL实现如以下代码所示。

ORG 0x7c00 ; 指明程序的装载地址

entry:

MOV AX, 0 ; 初始化寄存器

MOV SS, AX

MOV SP, 0x7c00

MOV DS, AX

; 读取硬盘

MOV AX, 0x0820

MOV ES, AX

MOV CH, 0 ; 柱面0

MOV DH, 0 ; 磁头0

MOV CL, 2 ; 扇区2

readloop:

MOV SI, 0 ; 记录失败次数的寄存器

retry:

MOV AH, 0x02 ; AH=0x02：读盘

MOV AL, 1 ; 1个扇区

MOV BX, 0

* 1. 中断管理实现

中断管理是操作系统的核心部分之一，对于一个小型的人机交互操作系统来说，中断管理的实现至关重要。中断是相对于程序的主动执行而言，由硬件或软件发出的一种请求，请求处理器暂停目前正在执行的程序，并转而去执行其他的程序或服务。中断分为硬件中断和软件中断两种。硬件中断是由外部设备产生的，比如I/O设备请求操作系统服务，键盘输入等等。软件中断是由程序员通过特定的指令发出的，用于请求操作系统提供服务。操作系统为了能够及时、正确地响应中断请求，必须设计好中断处理程序（Interrupt Handler）。中断处理程序通常由四个部分组成：中断向量表、中断处理程序、中断服务例程和中断退出程序。

中断向量表是一个具有固定长度的表格，每个表项对应着一个特定的中断号，每个中断号对应着一种中断处理程序。中断向量表的位置由IDTR（Interrupt Descriptor Table Register）指定，其起始地址保存在IDT中[16]。

中断服务例程是一个通用的程序模块，通常位于一个公共的系统软件中，用于响应所有可能的中断和异常事件。中断服务例程通常非常简单，它会通过调用特定的中断处理程序，来处理不同类型的中断事件。键盘中断处理如下所示。

void init\_keyboard(struct FIFO32 \*fifo, int data0) {

keyfifo = fifo;

keydata0 = data0;

wait\_KBC\_sendready();

io\_out8(PORT\_KEYCMD, KEYCMD\_WRITE\_MODE);

wait\_KBC\_sendready();

io\_out8(PORT\_KEYDAT, KBC\_MODE);

}

void int\_handler21(int \*esp) {

io\_out8(PIC0\_OCW2, 0x61); // 通知PIC IRQ-1的受理已经完成

int data = io\_in8(PORT\_KEYDAT);

fifo32\_put(keyfifo, data + keydata0);

}

* 1. 画面显示实现

一个系统是否简洁易用，主要看其图形界面是否设计得当，因为设计主要面向嵌入式设备，所以没有设计较为复杂的界面和交互，使用BIOS提供的0x10号中断来进行描绘，在X86架构的计算机中，BIOS会通过这个中断来进行文本和图形输出。您可以通过设置寄存器的值来实现不同的显示效果。例如，AH寄存器控制功能码，AL寄存器用于存储字符或颜色，BH和BL寄存器用于存储屏幕行列数位置等参数。画面描绘代码如下[17]。

MOV AX, 0x9000

MOV ES, AX

MOV DI, 0

MOV AX, 0x4f00

INT 0x10

CMP AX, 0x004f

JNE scrn320

; 检查VBE的版本

MOV AX, [ES:DI+4]

CMP AX, 0x0200

JB scrn320

; 取得画面模式信息

MOV CX, VBEMODE

MOV AX, 0x4f01

INT 0x10

CMP AX, 0x004f

JNE scrn320

* 1. 内存管理实现

分段式内存管理是一种操作系统中常用的内存管理方式，该方式将内存空间划分为一段一段的区域，在每个段中分配不同的程序或数据，从而实现内存的分配和管理。首先，我们需要在操作系统启动时进行内存检测操作。通过这个操作，我们可以获取可用内存区域的信息，以及内存区域的大小、起始地址等。然后，我们需要将可用内存区域按照一定的规则进行划分，生成多个“段”。我们可以将一个或多个进程或数据分配到不同的段中，从而实现内存的分配和管理。最后，我们需要设计一个内存回收的机制[18]。当一个程序或数据不再使用时，我们需要将其所占用的内存空间回收释放，以便其他程序或数据可以使用。同时，我们还需要注意内存的碎片问题，在新的程序或数据需要分配内存时，应该尽量避免出现过多的碎片。内存分配函数如下。

unsigned int memman\_alloc(struct MemMan \*man, unsigned int size) {

for (unsigned int i = 0; i < man->frees; i++) {

if (man->free[i].size >= size) {

// 找到了足够大的内存

unsigned int a = man->free[i].addr;

man->free[i].addr += size;

man->free[i].size -= size;

if (man->free[i].size == 0) {

// 如果free[i]变成了0，就减掉一条可用信息

man->frees--;

for (; i < man->frees; i++) {

man->free[i] = man->free[i + 1]; // 代入结构体

}

}

return a;

}

}

return 0; // 没有可用空间

}

## 系统测试

* 1. 系统测试的目的

系统测试是软件测试中的关键环节，通过对整个软件系统的集成测试、验收测试、冒烟测试和性能测试等，以保证软件质量和功能的稳定。系统测试是软件测试中的关键环节，通过对整个软件系统的集成测试、验收测试、冒烟测试和性能测试等，以保证软件质量和功能的稳定。系统测试可以发现软件中的缺陷和故障，并且及时进行修改和验证，以确保软件的稳定性和可靠性。对软件整个系统进行测试，提高软件的质量水平。

* 1. 系统测试用例
     1. 内存分配测试

对于嵌入式设备来说，内存的大小并不是统一的，为了适配不同的机型，此次基于x86的小型人机交互操作系统可以支持最小40MB内存，由于32位操作系统限制，只能使用32位寄存器来寻址，理论最大可以支持到4G内存。

由于找不到40MB内存大小的内存条，在此处，测试使用虚拟机进行测试，测试环境使用qemu，使用-m指令，控制虚拟机最大的内存限制。

在操作系统中，做了指令mem-view，查看总内存和剩余内存以及占用百分比。首先设置内存大小为40MB,如图5-1所示。

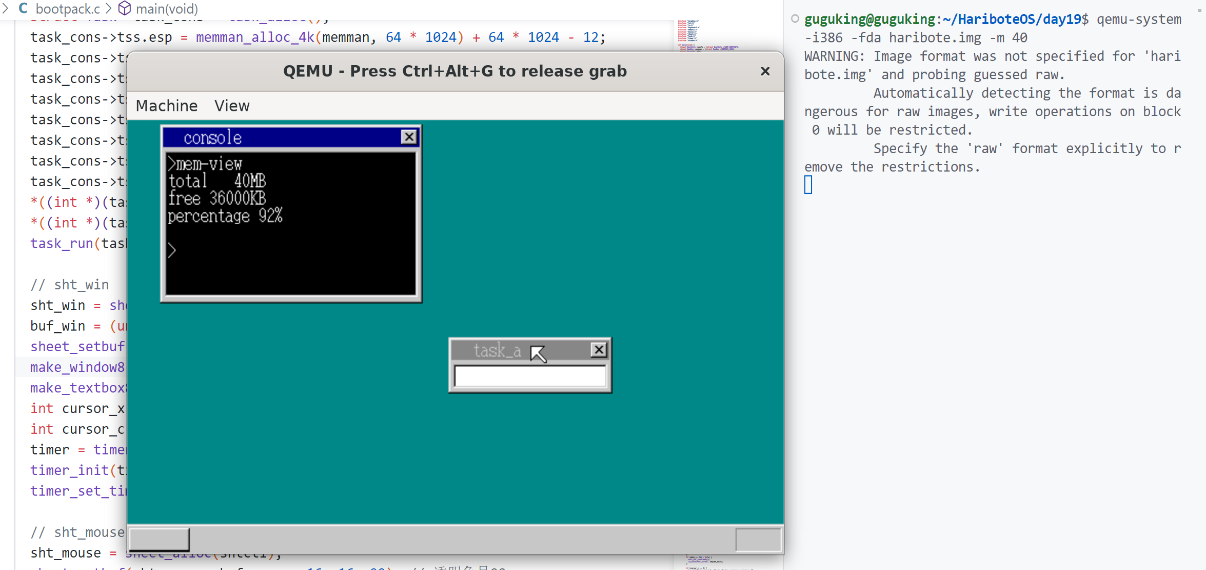


图5-1 系统设置内存40MB时显示

再来测试一下当内存超过4G时情况，设置内存大小为5000MB（5000MB>4096MB）,情况如图5-2所示。

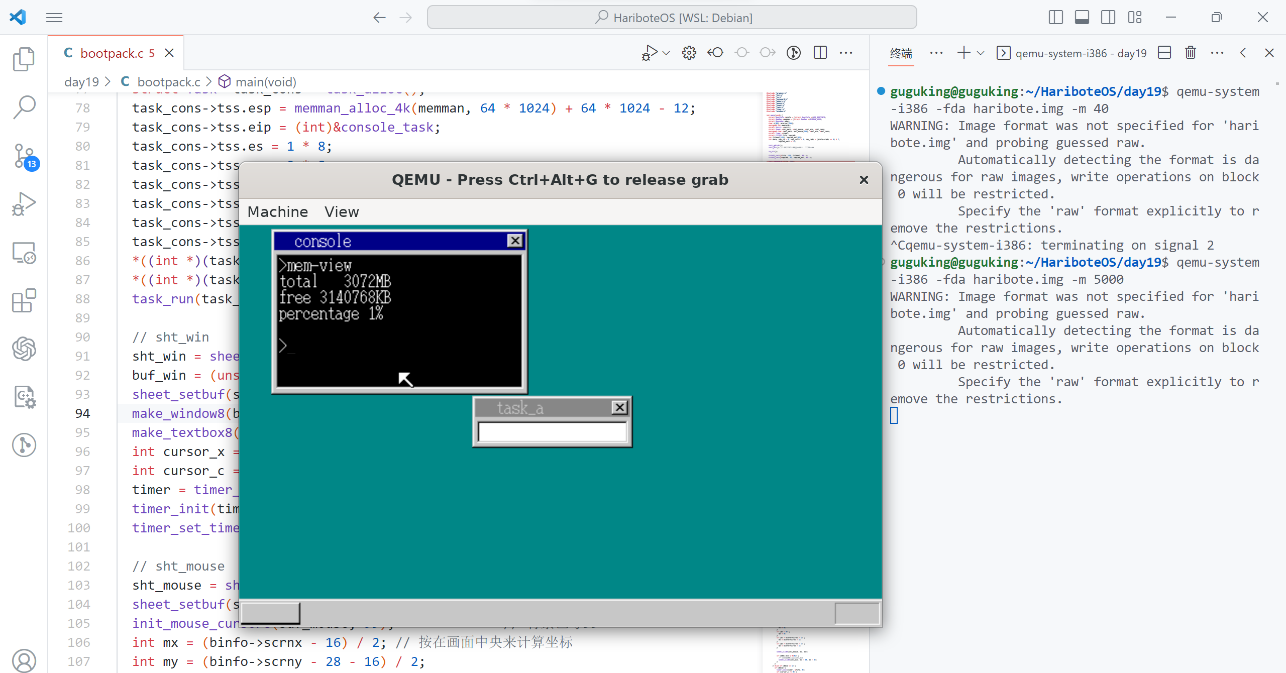


图5-2 系统设置内存5000MB时显示

此时可以看到操作系统中显示，内存总计为3072MB，显示不为4096MB，这不符合32位操作系统的理论内存上限大小。理论来说，32位寄存器，可以存储的最大值是2的32次方减1，即4294967295. 但是，在实际应用中，32位系统无法真正地访问全部4GB的物理内存，因为部分内存地址空间需要用于操作系统本身和其他设备的映射。这些区域包括系统BIOS、显存、I/O端口等等。因此，32位操作系统所能访问的实际物理内存是小于4GB的。

* + 1. 画面描绘测试

本次设计中，由于画面描绘功能是使用BIOS提供的中断0X10来实现的，只能画出相对较为简单的画面，但是现代嵌入式设备，可能会提供大小分辨率不同，大小不同的屏幕来输出信息，比如在机械工业上，CNC的机床可能只有一块640\*400分辨率，8bit的小屏幕来提供信息。但是在奶茶店的点单机上，可能会提供一块1280\*1024的较为大且清晰的显示器帮助客人点单。要提高分辨率，需要使用BIOS的VESA BIOS extention功能，这是由众多显卡厂商共同提出的标准，只有遵守这个标准才能使用BIOS描绘出更高分辨率的画面。本次测试依旧使用qemu进行模拟，因为真机环境无法截图。当我们处于不同的分辨率下，在qemu模拟出来的画面中，会有不同的大小显现，图5-3所示是当分辨率处于640\*400时产生的效果。

可以对比整个画面比例进行比较，在图5-4中，所示时分辨率处于800\*600的效果。

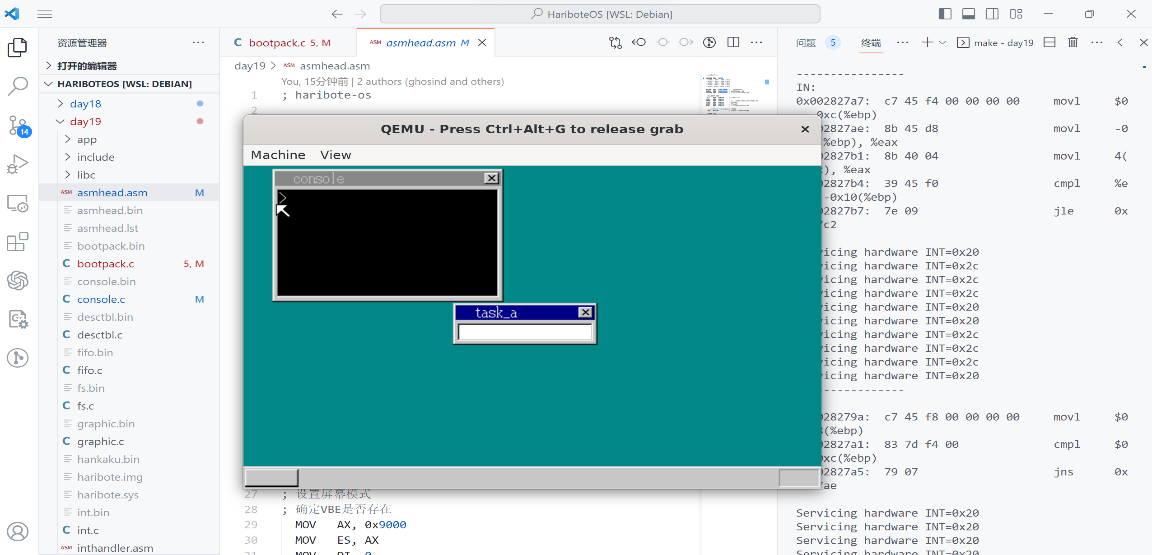


图5-3 分辨率为640\*400时画面大小

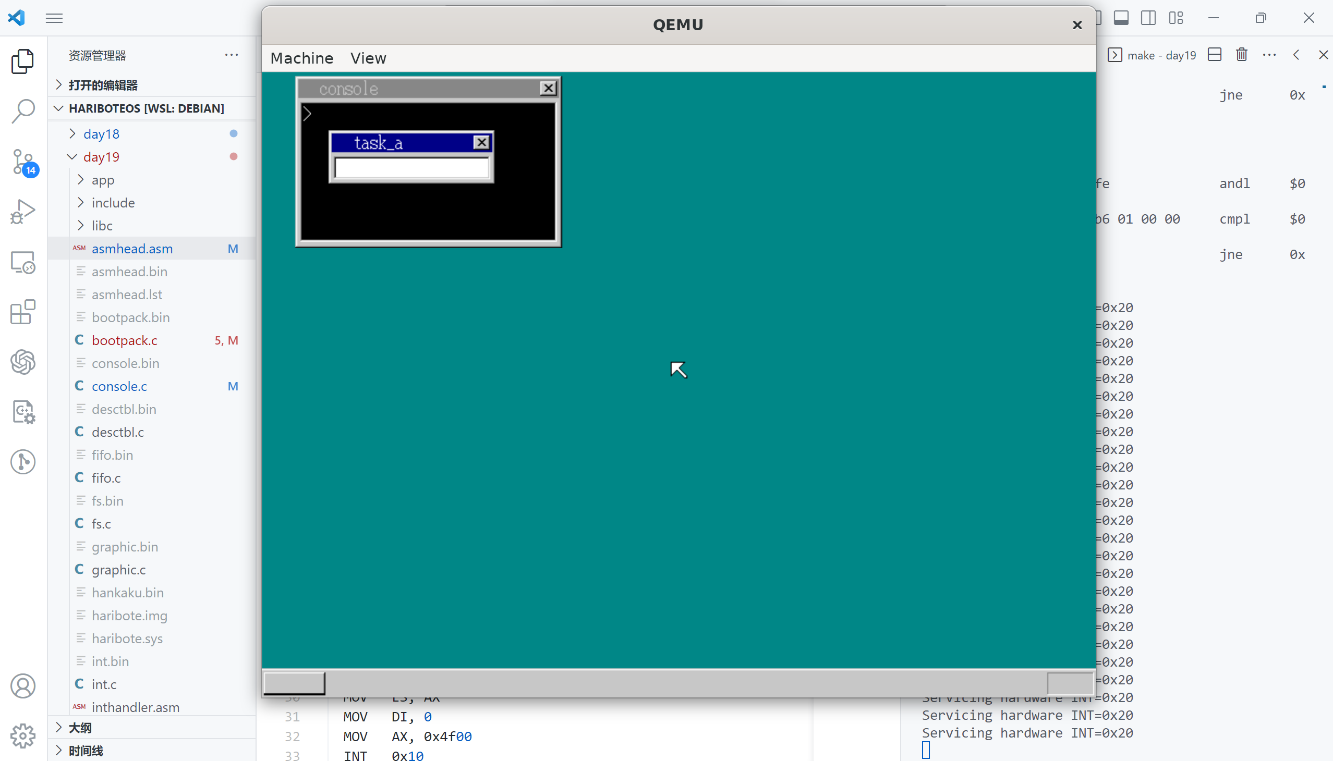


图5-4 分辨率为800\*600时画面大小

* + 1. 多任务测试

本次测试中，是为了测试在多任务情况下，本设计的实际执行情况。多任务是现代计算机操作系统中的一个重要特性，允许系统同时执行多个任务，从而提高了系统的效率和可用性。在传统的、单任务的计算机操作系统中，每次只能运行一个程序，无法同时完成多个任务，为用户带来更多的不便。多任务操作系统的引入，打破了这种限制，允许用户同时进行多项任务，从而更加高效地完成工作。本次设计中，采用的硬件是基于X86架构的，X86架构的CPU是一种现代化的通用微处理器，它是基于英特尔所开发的8086/8088处理器而来。后来，它被广泛用于台式计算机、服务器和移动设备等领域。X86架构CPU的特点是易于生产和开发，能够支持多任务处理，提供了高性能的计算能力和低功耗。所以即使是嵌入式系统，支持多任务依旧很重要。

假设一个场景，后台进行程序处理，前台进行文档编辑，为了更直观的显示，我在此处运行两个窗口，在我编写的简易终端（terminal）上输出从0-9999，在这个时间中，在前台运行一个简易打字台，进行输入，观察卡顿情况，接下来的图5-5和5-6会看到终端中不断地输出数字同时前台还可以进行输入。

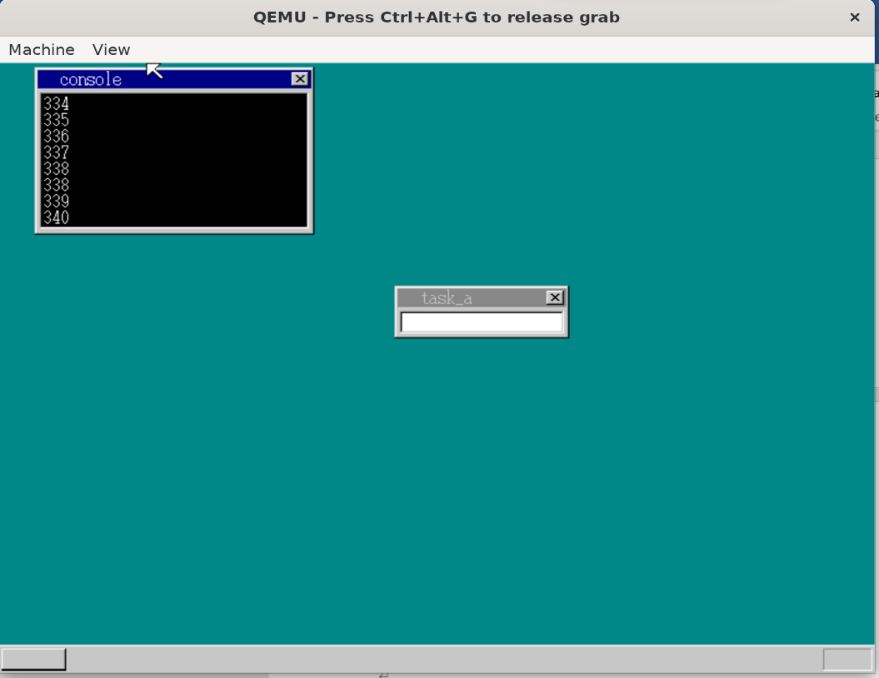


图5-5 多任务测试初始状态

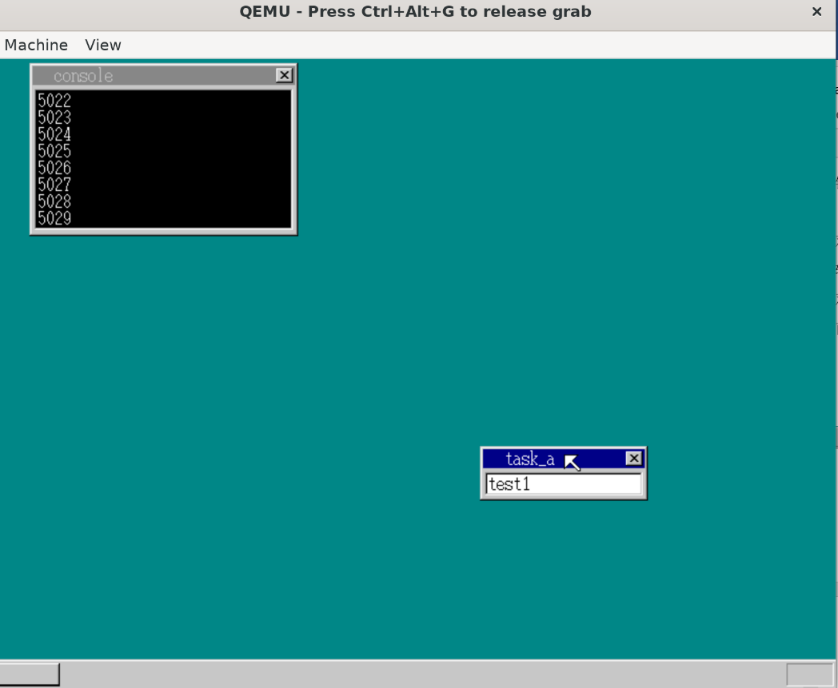


图5-6 多任务测试在task\_a输入

* + 1. 中断测试

在计算机系统中，中断是一种机制，允许优先级较高的任务打断当前正在运行的任务，并立即对其进行处理。当发生中断时，CPU会暂停当前运行的程序，跳转到中断处理程序中去执行相应的操作，一旦中断操作完成，CPU会返回中断之前的程序继续执行。中断是操作系统和硬件之间的桥梁，它允许硬件向操作系统发出信号，告诉系统有重要的事件需要处理，例如键盘输入、网络连接、存储设备、计时器等。

键盘和鼠标都是在中断的基础上进行工作的，在早期的计算机系统中，键盘和鼠标并没有像现在这么普遍，因此KBC（键盘控制器）是单独存在的，它主要用于控制和管理键盘的输入和输出。但随着鼠标的普及，计算机制造商为了给鼠标提供输入和输出，就将鼠标的信号也接入到了KBC中。

KBC（键盘控制器）和PIC（可编程中断控制器）是通过一个叫做中断控制器8259A的芯片相连的。该芯片有一个IRQ（中断请求）线路，用于接收从KBC和其他设备发出的中断请求。这些请求会被PIC进行管理和处理，从而实现对计算机系统中的各种设备的中断管理。当用户按下键盘上的键或移动鼠标时，KBC会通过中断请求信号通知给PIC，然后PIC会禁止其他IRQ请求，以在优先级较高的中断处理程序运行时处理该请求。这样可以有效避免竞争和冲突，保证计算机系统的顺畅运行。如图5-7所示，代表了KBC和PIC相连。

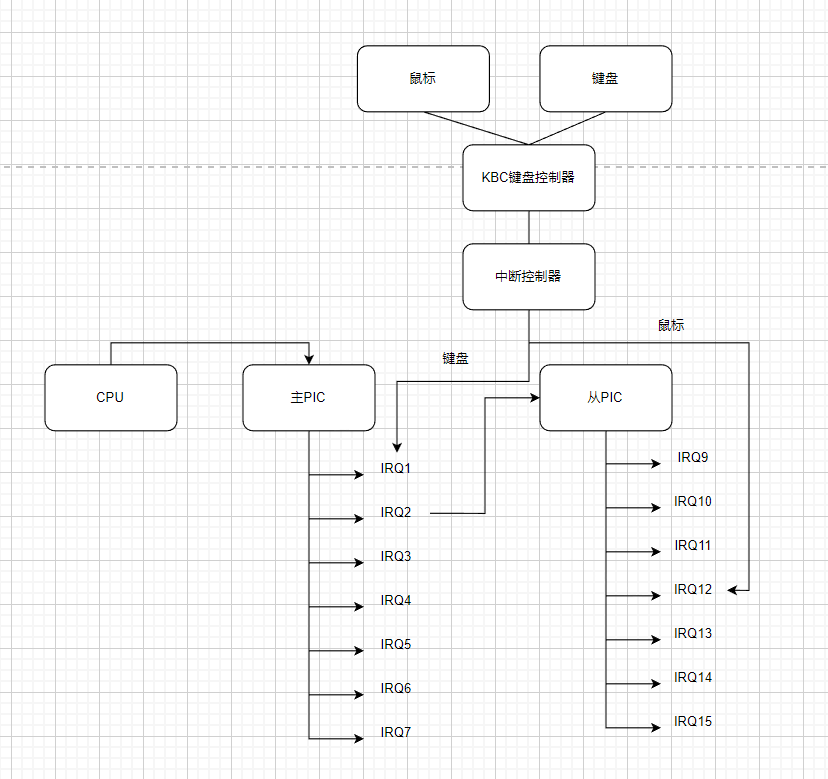


图5-7 键盘鼠标与中断的关系

键盘的中断使用IRQ1进行处理，而鼠标的中断使用IRQ12进行处理，但是由于鼠标和键盘同时都使用KBC键盘控制器，那么不可避免的会产生一些混淆，在KBC中对键盘和鼠标的中断进行了处理。当键盘产生中断时，KBC会将一个字节的数据（scan code）发送给中断控制器，以表示按下或松开的键的状态。这个字节的第7位通常被设为1，以表示这是一个键盘中断。而当鼠标产生中断时，KBC会将一个3个字节的数据包(BBB)发送给中断控制器，对应着鼠标的运动状态以及左右键的点击状态。这个数据也是通过IRQ12线发送的，但是它的第7位被设置为0，以表示这是一个鼠标中断。因此，当CPU接收到中断请求时，它可以通过检查接收到的数据来确定是键盘中断还是鼠标中断。CPU将会从不同的中断处理程序中进行处理，以响应来自键盘或鼠标的不同中断信号。

为了更好的展示鼠标键盘工作的场景，我将makefile进行改写，在qemu的参数中加上-d in\_asm，则当我在终端中使用qemu运行操作系统时，会指示QEMU打印指令级别汇编代码（即CPU执行的汇编代码）以及每条指令对应的内存地址和机器码等信息。这样就可以看到当中断发生时，操作系统做出的回应。

首先看图5-8，在操作系统背景上打印鼠标键盘的状态，首先是鼠标的坐标，然后是键盘产生的中断，分别代表按下左键，右键，和当前位置与上一个位置之间的坐标比较。

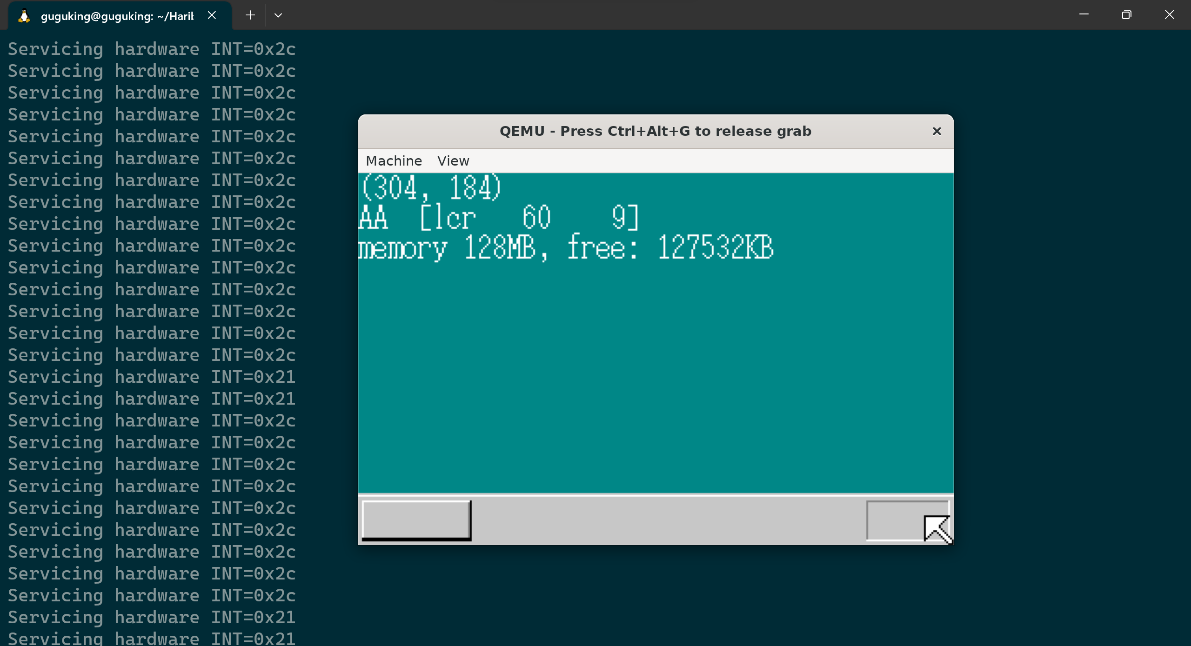


图5-8 打印键盘鼠标中断

当按下按键A时，键盘控制器(KBC)会检测到该事件，并发送一个中断请求给可编程中断控制器(PIC)。PIC会将该请求发送给CPU，CPU会在触发中断后停止正在执行的程序，并跳转到中断处理程序中去执行。这个中断处理程序会处理键盘中断请求，读取键盘缓冲区中的数据，并将其存储在内存中。最终，CPU将从中断处理程序返回到之前执行的程序中，程序可以从内存中读取被存储的键盘数据。在处理键盘中断时，CPU需要检查中断向量表以确定中断的类型，以便在正确的中断处理程序中进行处理。键盘中断的中断向量为0x21。在中断处理程序执行完毕后，CPU会从之前保存的状态中恢复，然后返回到之前执行的程序中，程序可以继续执行。这个过程通常在微秒级别内完成，用户几乎不会察觉到延迟。在键盘中断请求被处理之前，CPU会将所有其他中断请求屏蔽掉。这可以确保中断处理程序可以优先处理键盘中断请求，并且不会受到其他中断请求的干扰。如图5-9所示。

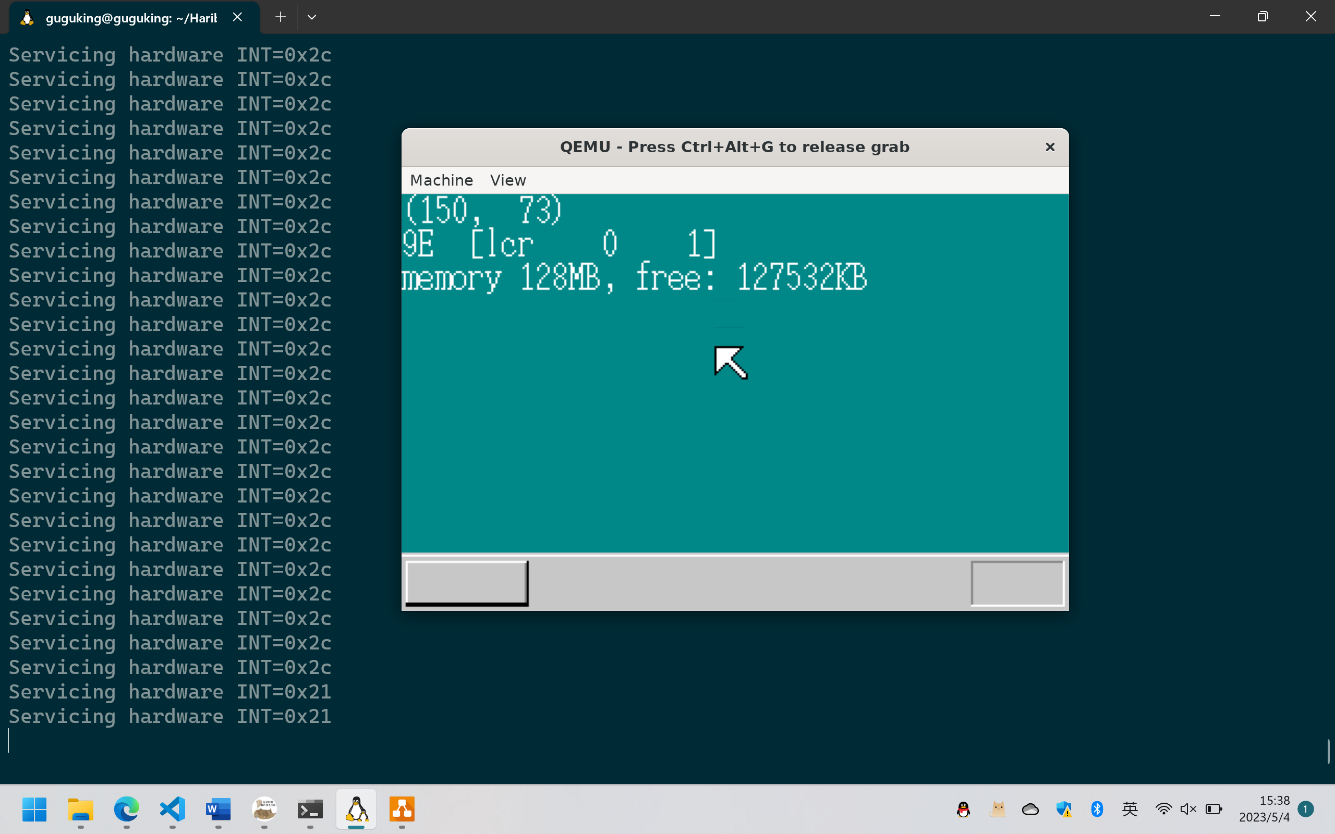


图5-9 按键A显示中断

鼠标的操作类似，鼠标中断产生时，CPU也会屏蔽其他中断，直到鼠标中断处理完成。图5-10显示当鼠标左键按下时情况。

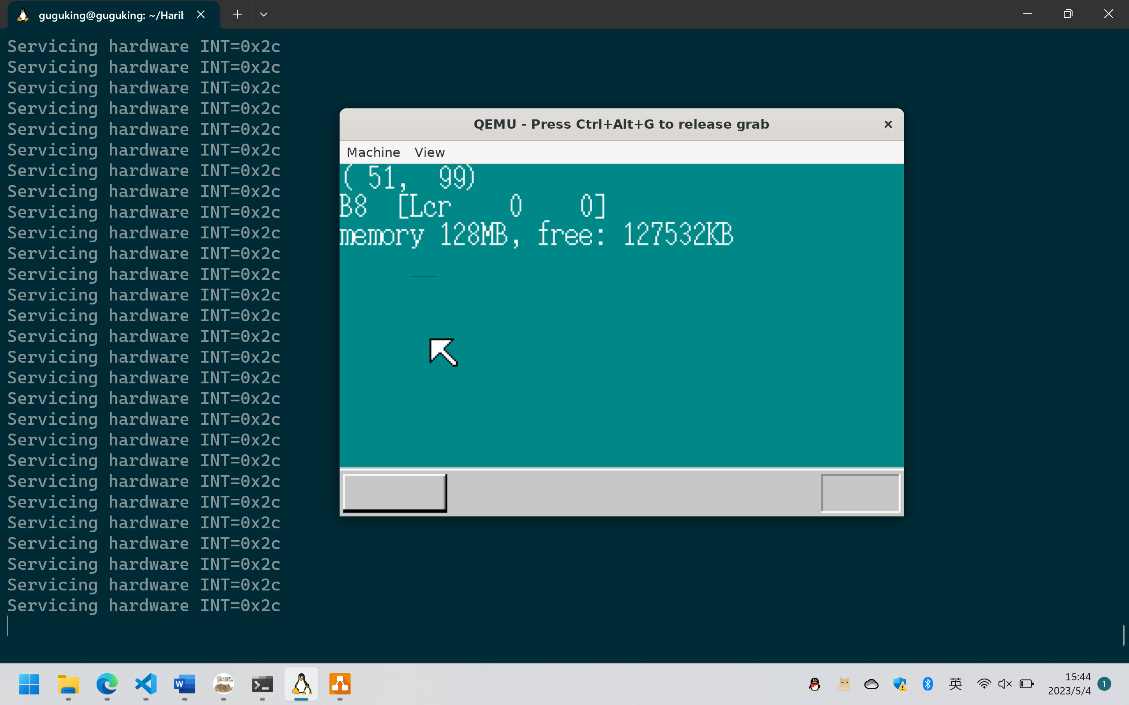


图5-10 鼠标左键显示中断

图5-11表示当鼠标右键按下时情况。

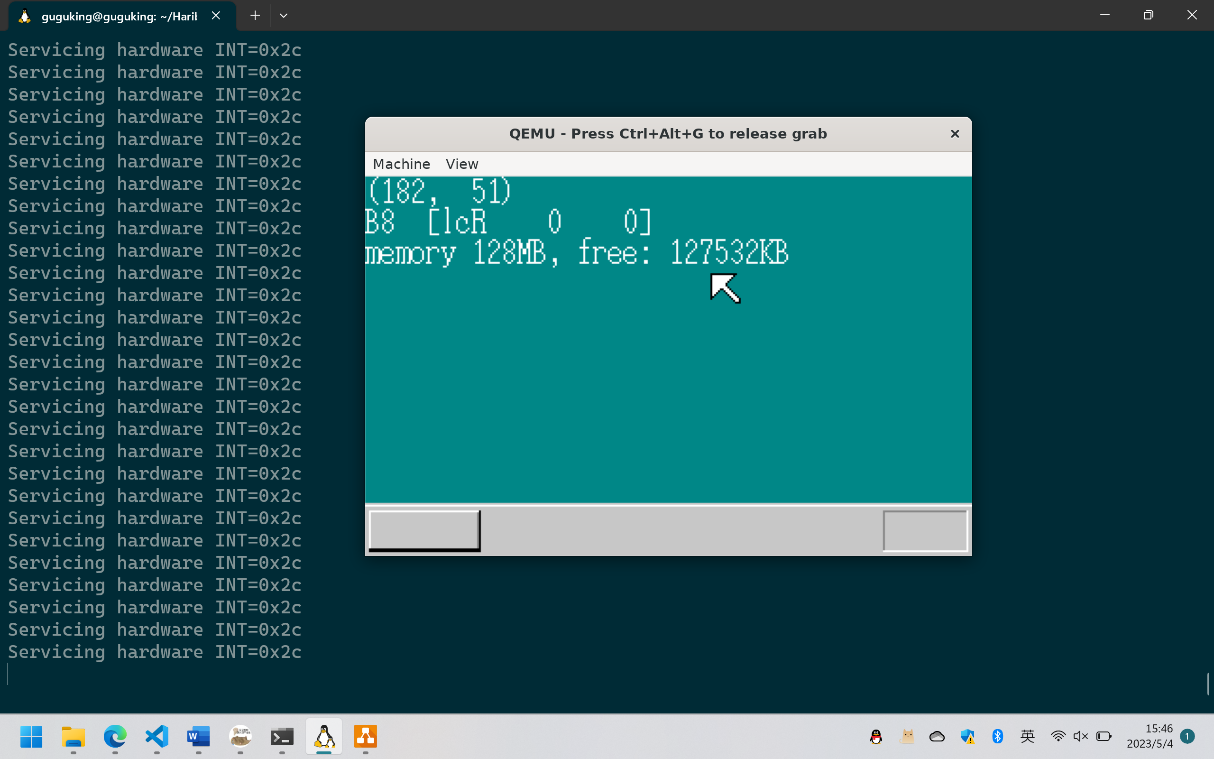


图5-11 鼠标右键显示中断

## 成果与社会、经济的关系

随着互联网技术和智能化水平的不断提升，人们对于嵌入式系统的需求也越来越高。嵌入式系统是一种特殊的计算机系统，它被嵌入到其它设备中，并且运行着专门的程序。嵌入式系统的应用范围非常广泛，从智能家居到医疗设备、从智能交通到工业自动化等等。而人机交互操作系统则是嵌入式系统中至关重要的一环，主要负责用户与设备之间的交互，在实现嵌入式系统智能化、便捷化、高效化方面具有非常重要的作用。嵌入式系统的应用非常广泛，其中最为常见的是在工业自动化中。在工业领域，随着生产方式的不断进步，生产线的效率要求越来越高，而嵌入式系统的使用，可以实现设备间的互联互通，使得生产管理更加便捷、科学。同时，通过嵌入式系统的精确测量、自动化控制，可以大大减少人工干预，提高生产效率，避免了人力浪费和不必要的错误。嵌入式系统作为信息科技领域中的重要一环，对于推动科技领域的发展起到了重要的推动作用。嵌入式系统的普及和应用，将引领科技更加开放、更加自由、更加先进的发展，促进了信息技术与各行各业的深度融合。随着嵌入式技术的发展和普及，人们对于科学技术发展的认知和理解也将越来越深刻。以嵌入式系统为基础的各种产品，如智能手机、智能家居智能家电等都不断受到市场的欢迎。在嵌入式技术不断发展的背景下，不断有更多的产品应运而生，市场需求不断扩大。嵌入式系统的应用、提升、创新也将直接带动市场的发展，进一步带动经济的发展。基于X86的小型人机嵌入式交互操作系统凭借着其强大的功能、广泛的应用和持续的技术革新，对于社会和经济都起到了积极的推动作用。随着嵌入式技术的不断创新和发展，无论是从技术创新、市场拓展还是从促进经济发展乃至改善人们生活质量等方面，都将产生更加深远的影响。因此，在今后对于嵌入式技术的研究中，不仅需要注重技术的创新发展，更要始终牢记嵌入式系统的应用宗旨，为推动社会和经济的发展而不懈努力。

## 结 论

我完成了基于X86的小型人机嵌入式交互操作系统的开发。从最开始的需求分析到最后的系统产出，这个过程让我充实且收获颇丰。

最开始，我考虑到嵌入式操作系统需要考虑资源限制和功耗等问题，因此选择了使用C语言和汇编的方式。我对嵌入式操作系统的整体结构和实现方式进行了研读和学习，最终决定采用裸机编程方式。使用裸机编程的好处是可以直接控制硬件，提高系统的响应速度，并取消了操作系统的开销，从而使嵌入式系统具有较高的稳定性。我还使用了版本管理工具Git，这样能够更好地跟踪自己的代码变化，提高代码的可维护性和可读性。

编译器的选择时比较为难，在老师知道我想将此项目进行开源之后，推荐我使用开源的编译器和较新的版本，最终确定了使用C99标准版的GNU GCC作为C语言编译器，NASM作为汇编编译器。

在进行开发的时候，我遇到了很多问题，比如系统启动流程、中断处理、时钟管理等方面的问题。对于这些问题，我深入学习了X86硬件的相关知识，并阅读了大量相关的编程手册和文献，最终成功地解决了这些问题。

在系统设计方面，我考虑到嵌入式系统需要具备实时性和易用性，因此选择了使用汇编语言进行编程。在系统设计的过程中，我将操作系统分为了用户界面和内核两个层次，使用汇编语言实现了内核层的各种操作和功能，同时使用C语言实现了用户层的各种操作和功能。

在开发内存管理模块的过程中，我切身体会到了自己的操作系统与Windows和Linux的差距，Windwos和Linux系统上，内存是分页和分段同时实现的，先分页在分段，让程序开发人员在面对内存时，有着统一的API，不需要根据内存来对自己的应用程序做适配，在内存压缩算法上，成熟的操作系统使用页面交换技术和内存压缩技术来最大程度地利用系统内存。并且会自动根据计算机运行状态来进行压缩，以保证性能的释放，而我采用了最简单的内存压缩技术，在内存即将满载的时候，会一定程度上影响正在执行的程序效率，在以后的版本更迭中，我想我会要进一步改善这个问题。

这个项目的开发过程让我受益匪浅。不仅学到了诸多硬件和软件方面的知识，也提高了自己的编程能力和代码实现能力。我相信，这个小型人机嵌入式交互操作系统在开源之后，经过开源工作者们共同的努力，这个项目可以变得真正的进入嵌入式环境。

## 致 谢

首先，我要感谢我的导师李伟，谢谢您为我提供了这次难得的机会，让我能够接触到更广阔的知识和经验。在完成整个毕业设计的过程中，您给予了我许多指导和支持，让我学习到了更多的实践经验和方法，我会将这些知识铭记在心，并将其付诸实践。

回首四年大学的求学历程，我收获了知识，也收获了朋友，在此论文完成之际，提笔拜谢。感谢大学期间所有任课老师们，正是他们的传道授业解惑，使我学习和掌握了比较扎实的专业基础知识，顺利地完成了大学课程；感谢导师田宁讲师，在整个毕业论文的写作过程中，不厌其烦，一审再审，给予我莫大的指导和帮助；当然，还要感谢给予我帮助的同窗们和我的家人，谢谢你们的支持与鼓励。

## 参考文献

1. 孙悦,周洲.操作系统原理课程中断技术教学的探讨[J].现代计算机,2020,No.699(27):77-80.
2. Randal E.Bryant.深入理解计算机系统[M].机械工业出版社,2020,10(11):1-20.
3. 吴斯,梁心雨.基于X86架构的操作系统微内核设计与实现[J].大众科技,2020,22(10):12-14+34.
4. 吴元斌.C语言程序的理解与编译优化[J].现代计算机,2020,No.690(18):93-96
5. 川合秀实著.30天自制操作系统[M].周自恒,李黎明等,译.北京:人民邮电出版社,2012,20-35.
6. 林卓,齐晓斌,田丹.基于运行时嵌入式系统的动态升级技术研究[J].航空计算技术,2022,52(06):93-97.
7. 王烨,张颖,唐璞.基于Linux系统的PCIE高速数据卡驱动设计[J].数字技术与应用,2022,40(10):217-221.DOI:10.19695/j.cnki.cn12-1369.2022.10.66.
8. 李雪源,陈朋瑞,叶上华.基于Linux平台的多任务调度器的设计与实现[J].航空计算技术,2022,52(05):123-125+129.
9. 陈培德,吴建平,刘宏杰等.MBR磁盘转换为GPT磁盘的研究与实现[J].计算机技术与发展,2022,32(07):99-104.
10. 贾巧雯,马昊玉,厉严等.一种嵌入式Linux系统上的新型完整性度量架构[J]. 计算机研究与发展,2022,59(10):2362-2375.
11. 陈宇,曾颜,张先勇.嵌入式系统中虚拟化设备的设计与实现[J].电子设计工程, 2022,30(24):125-129.DOI:10.14022/j.issn1674-6236.2022.24.026.
12. 龚爽.Linux系统下计算机C语言的编程技巧探讨[J].电脑编程技巧与维护,2 022,No.441(03):40-42.DOI:10.16184/j.cnki.comprg.2022.03.039.
13. 何恺.嵌入式Linux下内存泄漏的检查和解决[J].现代计算机,2020,No..683(1 1):78-82.
14. 侯光霞,吕大鹏,梅涛.嵌入式操作系统中引导软件研究[J].信息技术与信息化, 2022,No.266(05):99-102.
15. 林玉哲,蒋金虎,张为华.多内核操作系统综述[J].计算机系统应用2022,31(05): 21-29.DOI:10.15888/j.cnki.csa.008426.
16. Avtar S,Navjot K,Harpreet K. Extensive performance analysis of OpenDayLight (ODL) and Open Network Operating System (ONOS) SDN controllers[J]. Micro processors and Microsystems,2022, 95.
17. Science-Computer Science;Investigators from National University of Defense Te chnology Release New Data on Computer Science(Towards Functional Verifyin g a Family of Systemc Tlms)[J]. Computer Weekly News,2020.
18. ELJADIRI L,ASSAYAD I.Generic Framework Architecture for Verifying Embedded Components[J].International Journal of Advanced Computer Science and Applications (IJACSA),2020,11(6).