操作系统实验报告--中断向量表

智能科学与技术　　2024 年 3 月 29 日

实验任务

使用 nasm 汇编编程实现一个 MBR 程序, 每隔 1 秒钟左右输出一个 (组) 字符 (自定), 具体内容包括: 1) 实现一个时钟中断处理例程, 按规定时间间隔输出字符 (可使用 BIOS 提供的 int 10h 实现字符输出); 2) 修改中断向量表, 将时钟中断处理例程起始地址填入中断向量表对应表项; 3) 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。

1 前置知识整合

1.1 中断向量表

中断向量表 (interrupt vector table) 包含中断服务程序地址的特定内存区域，这些服务程序是处理外部硬件中断请求的代码。这些中断服务程序 (函数) 在中断向量表中的位置是由半导体厂商定好的，当某个中断被触发以后就会自动跳转到中断向量表中对应的中断服务程序 (函数) 入口地址处。

计算机存储器从 0 开始的 1K（0000H～003FFH）的字节被用作中断向量表区，它一共存放了 256 个中断向量，每个中断向量占四个字节。这样正好就是 1K（256*4=1K）个字节。除了这两块专用的区域，其他区域用来存储一般的程序指令和数据。在这块区域（0000H～003FFH），还有那些用于进行中断处理的程序，这些程序就被称为中断服务程序。而这些程序代码起始地址则被称为中断服务程序的入口地址。

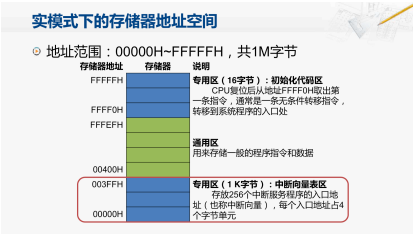

Figure 1: 内存地址

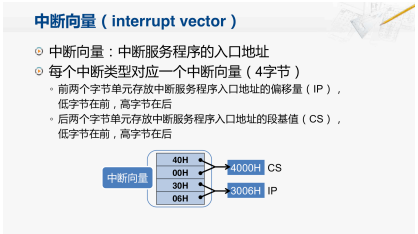

Figure 2: 中断向量

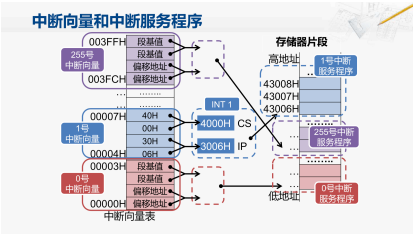

Figure 3: 中断向量存储

根据查询资料可以得到已有的中断向量表的中断向量及其功能，由此可知与时钟中断有关的中断号为 8h 与 1Ch。

```
中断向量表
内存地址(十六进制) 对应向量号(十六进制) 中断用途

                    *******BASIC中断向量******
0x3C4 - 0x3FF F1-FF 未使用
0x218 - 0x3C3 86-F0 BASIC程序运行时提供给BASIC解释程序作用
0x200 - 0x217 80-85 为BASIC保留
0x1E0 - 0x1FF 78-7F 未使用
```

0x1DC - 0x1DF 77 硬件中断15
0x1D8 - 0x1DB 76 硬件中断14
0x1D4 - 0x1D7 75 硬件中断13
0x1D0 - 0x1D3 74 硬件中断12
0x1CC - 0x1CF 73 硬件中断11
0x1C8 - 0x1CB 72 硬件中断10
0x1C4 - 0x1C7 71 硬件中断9
0x1C0 - 0x1C3 70 硬件中断
0x1A0 - 0x1BF 68-6F 未使用
0x180 - 0x19F 60-67 为用户程序保留的单元
0x128 - 0x17F 4A-5F 保留
0x124 - 0x127 49 指向键盘增强服务变换表
0x120 - 0x123 48 PC机使用，用于把PC机的键盘代码变换为标准的键盘代码
0x11C - 0x11F 47 保留

******DOS中断向量*******
0x118 - 0x11B 46 第二硬盘参数块
0x114 - 0x117 45 保留
0x110 - 0x113 44 PC机使用，用于指向低分辨率图形字符参数表
0x108 - 0x10F 42-43 未使用
0x104 - 0x107 41 硬盘参数块
0x0C0 - 0x0CB 34-40 未使用
0x0CC - 0x0CF 33 鼠标中断
0x0C0 - 0x0CB 30-32 未使用
0x0BC - 0x0BF 2F 多路服务中断
0x0B8 - 0x0BB 2E 基本SHELL程序装入
0x0AC - 0x0B7 2B-2D 未使用
0x0A8 - 0x0AB 2A Microsoft 网络接口
0x0A4 - 0x0A7 29 快速写字符
0x0A0 - 0x0A3 28 DOS安全使用
0x09C - 0x09F 27 终止并驻留程序
0x098 - 0x09B 26 绝对磁盘写功能
0x094 - 0x097 25 绝对磁盘读功能
0x090 - 0x093 24 严重错误处理(用户不能直接调用)
0x08C - 0x08F 23 Ctrl+Break 处理地址(用户不能直接调用)
0x088 - 0x08B 22 程序中止时DOS返回地址(用户不能直接调用)
0x084 - 0x087 21 DOS系统功能调用
0x080 - 0x083 20 DOS中断返回

*****数据表指针********
0x07C - 0x07F 1F 图形字符扩展码
0x078 - 0x07B 1E 软盘参数块
0x074 - 0x077 1D 视频参数块

*****提供给用户的中断*****
0x070 - 0x073 1C 定时器控制的软中断
0x06C - 0x06F 1B Ctrl + Break控制的软中断

```
                    *********BIOS中断*********
0x068 - 0x06B 1A 时钟管理
0x064 - 0x067 19 引导装入程序--系统自举
0x060 - 0x063 18 BASIC入口代码--ROM BASIC入口代码
0x05C - 0x05F 17 打印机输出
0x058 - 0x05B 16 键盘I/O
0x054 - 0x057 15 盒式磁带I/O
0x050 - 0x053 14 RS-232串行通讯口I/O
0x04C - 0x04F 13 磁盘I/O
0x048 - 0x04B 12 测定存储器容量
0x044 - 0x047 11 设备检验
0x040 - 0x043 10 屏幕显示I/O

                  ********8259中断向量*********
0x03C - 0x03F F LPT2控制器中断--并行打印机(IRQ7)
0x038 - 0x03B E 磁盘控制器中断--软磁盘(IRQ6)
0x034 - 0x037 D LPT2控制器中断--硬磁盘(并行口)(IRQ5)
0x030 - 0x033 C 异步通信(primary)--串行通信接口1(IRQ4)
0x02C - 0x02F B 异步通信(secondary)--串行通信接口2(IRQ3)
0x028 - 0x02B A 彩色/图形(IRQ2)
0x024 - 0x027 9 键盘(IRQ1)
0x020 - 0x023 8 定时器(IRQ0)

                  ********8088中断向量*********
0x01C - 0x01F 7 保留
0x018 - 0x01B 6 保留
0x014 - 0x017 5 打印屏幕
0x010 - 0x013 4 溢出
0x00C - 0x00F 3 断点指令
0x008 - 0x00B 2 非屏蔽中断
0x004 - 0x007 1 单步(用于DEBUG)
0x000 - 0x003 0 除以零
```

1.2 8254/3 定时器

8254/3 定时器是一种计时器/计数器芯片，通常用于计算机系统中的定时器功能。它是 Intel 8254 芯片系列的一部分，被广泛用于早期个人计算机和嵌入式系统中。8254 芯片有三个独立的计数器，每个都可以被编程为不同的计时或计数模式。它们通常用于生成精确的时间延迟、产生脉冲信号、执行时间测量等任务。当 8253/8254 定时器产生时间中断时，它会通过 8259 中断控制器发送中断请求给处理器，处理器则会暂停当前的执行，转而执行与该中断相关的中断服务例程，以处理时间中断。这两个设备的协作使得处理器能够在特定的时间间隔内进行中断处理，实现定时和时间相关的功能。

2 实验设计

根据查询到的资料，在系统加电初始化期间, 把系统定时器初始化为每隔约 55ms 发出一次中断请求,CPU 在响应定时器中断请求后转入 8H 号中断处理程序,BIOS 提供的 8H 号中断处理程序中有一条中断指令 INT 1CH, 所以每秒要调用到约 18.2 次 1CH 号中断处理程序, 而 BIOS 的 1CH 号中断处理程序实际上并没有做任何工作, 只有一条中断返回指令。根据前置知识，考虑编写时钟中断处理例程并直接替换原 8h 号中断处理程序，然后修改定时器频率。综上，将实验分为以下三个部分进行实现与测试：

（1）保留原定时器频率不进行修改，实现一个时钟中断处理程序，使它能每隔 1s 输出字符 A，并且每次输出不覆盖上一次输出；

（2）将时钟中断处理程序的地址写进中断向量表中 8h 号对应的中断向量中；

（3）修改 8254/3 定时器的频率，使之变为每隔 20ms 发出时钟中断。

3 实验过程

3.1 实验环境与准备

1. 使用 QEMU 创建一个简单的虚拟机用于实验。

```
qemu-img create -f raw disk.raw 1G
```

2. GDB 调试工具

3.2 Part1：时钟中断处理程序

3.2.1 实验过程

实验基于文本模式来进行字符输出。具体来说，通过操作视频内存来实现字符的显示。在 x86 架构中，文本模式下的视频内存地址通常是 0xB8000。每个字符都由两个字节组成，一个字节用于存储 ASCII 码，另一个字节用于控制字符的颜色和其他属性。在以下这段代码中，将字符'A' 的 ASCII 码值写入到视频内存中指定位置，从而在屏幕上显示出'A' 字符。通过 counter 进行计数，由于未设置定时器之前时钟中断每隔 55ms 发出，那么当时钟中断触发了 18 次时达到 1s 左右，此时在下一个字符位置输出字符 A，否则不输出。

```
clock_interrupt_handler:
    inc byte [counter] // 将计数器的值增加 1
    mov ax, [counter]
    cmp ax, 50 // 将counter中的值与 50 比较
    jl skip_display // 如果 AX 中的值小于 50, 则跳转到 'skip_display'

    mov dl, 'A' // 将字符 'A' 的 ASCII 值移动到 DL 寄存器中
    mov ax, 0xb800 // 将十六进制值 0xB800 移动到 AX 寄存器中（视频内存段）
    mov es, ax // 将 AX 寄存器中的值移动到 ES 寄存器中（视频内存段）
    mov byte [es:di], dl // 将 DL 寄存器中的值移动到内存地址 ES:DI 处的字节中（视频内
        存）
    add di, 2 // es: di指向屏幕下一个字符对应的内存地址
```

```
12
13      mov word [counter], 0 // 将计数器的值重置为 0
14
15      mov al, 20h
16      out 20h, al // 将 AL 寄存器中的值输出到端口 20h（中断控制器）
17      iret // 从中断返回
18
19  //未达到1s的时间间隔
20  skip_display:
21      mov al, 20h
22      out 20h, al
23      iret
```

3.2.2 实验过程遇到的实验困难与解决

初版并未考虑到每次中断后需要告知 CPU 中断结束的问题，或者说，x86 在执行中断处理例程之前默认会关闭中断响应，因此每次中断处理例程结束后需要发送一个 EOI 给 8259A，以便继续接受中断，即增加“mov al, 20h ， out 20h, al ”，否则无法连续处理时钟中断程序，会停在第一次中断后。

3.3 Part2：改写中断向量表

根据前置知识中对中断向量表的分析，将自实现的时钟中断处理程序的地址写到中断向量表 8h 号中断向量处。

```
1  xor ax, ax
2  mov ds, ax // 将数据段寄存器 DS 设置为零，这样数据段的偏移地址将为零
3  mov bx, 32 // 将寄存器 BX 设置为32=8*4，表示中断向量表中的8H号的位置，将时钟中断处理程
       序的偏移地址写进此处
4  mov word [bx], clock_interrupt_handler-$$ // 将时钟中断处理程序的偏移地址减去当前位置
       的偏移地址，然后将结果写入到中断向量表中
5  mov word [bx+2], 07c0h // 将时钟中断处理程序的段地址写进此处
```

3.4 Part3：修改定时器频率

使用以下步骤来修改 8253/8254 的发出时钟中断的频率，使之每 20ms 发送一次：

1. 选择计数器通道 0：8253 有三个独立的计数器通道：通道 0、通道 1 和通道 2。使用通道 0 来实现 20 毫秒的定时中断。

2. 设置工作方式：通道 0 的工作方式有多种，我们选择方式 3（方波发生器）。方式 3 会产生周期性的方波输出，适合用于定时中断。

3. 计算计数初值：8253 的时钟频率通常为 1.19318 MHz。

计数初值 = 时钟频率 × 定时时间

计数初值 = 1.19318 MHz × 20 ms = 23863

4. 编写汇编代码：使用汇编语言编写程序，将 8253 的计数器通道 0 设置为方式 3，并初始化计数初值为 23863。

```
    ; 设置8253的控制字，选择通道0并设置为方式3
    mov dx, 0x43
    mov al, 0x36
    out dx, al

    ; 设置计数初值为23863
    mov dx, 0x40
    mov ax, 23863
    out dx, al
    mov al, ah
    out dx, al
```

3.5 整合

以上三部分分别进行测试成功后，对三部分进行整合，根据新设的频率，将 counter 的比较跳转上调到 50（1000/20=50）。

```
//设置代码段的起始位置
SECTION MBR vstart=0x7c00
_start:
    //清屏
    mov ax, 0600h
    mov bx, 0700h
    mov cx, 0
    mov dx, 184fh
    int 10h

    //写入中断向量表
    xor ax,ax
    mov ds,ax
    mov bx,32
    mov word [bx],clock_interrupt_handler-$$
    mov word [bx+2],07c0h

    //设置时钟频率
    ; 设置8253的控制字，选择通道0并设置为方式3
    mov dx, 0x43
    mov al, 0x36
    out dx, al

    ;设置计数初值为23863
    mov dx, 0x40
    mov ax, 23863
    out dx, al
    mov al, ah
    out dx, al
```

```
    //开启中断
    sti
    jmp $

//时钟中断处理例程
clock_interrupt_handler:


    inc byte [counter]

    mov ax, [counter]
    cmp ax, 50
    jl skip_display //未达到50次中断，那么不打印

    mov dl, 'A'
    mov ax,0xb800
    mov es,ax
    mov byte[es:di],dl
    add di,2 //es: di指向屏幕下一个字符对应的内存地址

    mov word [counter],0

    mov al,20h
    out 20h,al
    iret

skip_display:
    mov al,20h
    out 20h,al
    // 结束中断处理例程
    iret

counter dw 0 // 计数器变量，用于记录时钟中断触发次数
times 510-($-$$) db 0 // 填充剩余空间使程序大小为512字节
dw 0xAA55 // MBR标志
```

相关命令行

```
编译新bin: nasm -f bin -o time.bin time.asm
替换新bin: dd if=time.bin of=disk.raw bs=512 count=1
打开: qemu-system-x86_64 -drive file=disk.raw,format=raw
```

4 实验结果

启动虚拟机，在启动界面以每秒一个字符的速度打印字符 A，且每次输出不覆盖下一次输出，以便观察结果与计算速度。

Figure 4: 启动 3s 后

Figure 5: 启动 4s 后

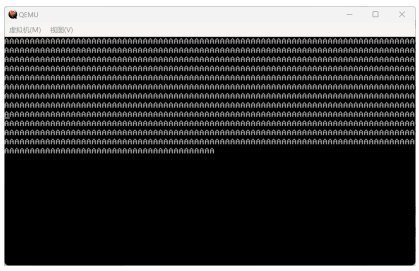

Figure 6: 启动 n s 后

5 参考文章

1. 终于把 int 8h 调通了
2. NASM 汇编语言与计算机系统 10-中断向量表 0 号中断 (clistiirethlt)
3. 9.2 中断向量表的结构
4. 9.4 中断的处理过程
5. 10.2 输入输出接口的编址方式
6. 中断向量表