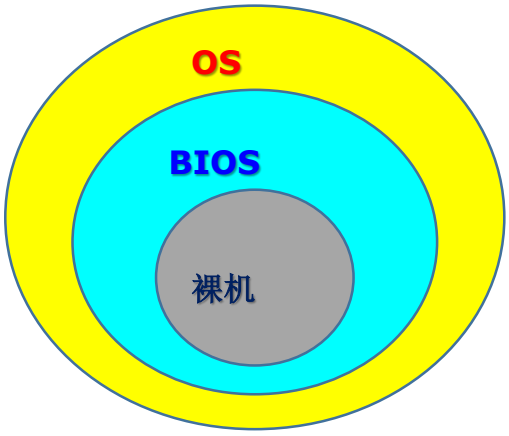
第 7 章　BIOS 和虚拟机

7.1　BIOS 及其调用



7.1.1　BIOS 简介

关于 BIOS

BIOS(Basic Input/Output System)就是基本输入输出系统，它被固化在 ROM 中。

BIOS 包含了主要 I/O 设备的处理程序和许多常用例行程序，它们一般以中断处理程序的形式存在。

BIOS 是覆盖在硬件系统上的第一层软件，它直接操纵 I/O 设备或者硬件设备，实现计算机系统的基本输入或输出。

简单使用机器的途径

BIOS 为用户使用“裸机”，提供简单的基本途径。在没有安装操作系统的计算机上，可以通过 BIOS 使用计算机。

BIOS 支持基本的键盘输入，能够根据用户的按键操作，得到对应键符的 ASCII 码等。它支持基本的显示输出，能够根据字符的 ASCII 码和显示属性（颜色），在屏幕上的指定位置显示对应的字符。它还支持基本的鼠标操作和打印操作等。还支持读写外部存储设备。

通过 BIOS，可以读取磁盘上的特定程序。利用 BIOS，这样的程序可以进行基本的输入和输出。

主要作用

在操作系统启动自举的过程中，BIOS 发挥重要作用。依靠 BIOS，操作系统完成启动自举。

简易的操作系统可以直接建立在 BIOS 的基础之上。曾经十分流行的磁盘操作系统 DOS(Disk Operating System)就是这样，通过 BIOS 操纵控制硬件。

Windows 和 Linux 等操作系统在启动成功后，会直接控制操纵硬件。这样的操作系统完全掌控硬件，实现完备的输入输出功能。

7.1.2　键盘输入和显示输出

键位和扫描码

键盘上的键可以分为五类：字符键(字母、数字和符号等)，功能键(如 F1 和 PgUp 等)，控制键(Ctrl、Alt 和左右 Shift)，双态键(如 Num Lock 和 Caps Lock 等)，特殊请求键(如 Print screen 等)。

字符键有对应的 ASCII 码，其他的键并没有 ASCII 码。

每个键有一个代表键位置的扫描码。

在用户实施按键动作后，键盘作为外部设备会发送扫描码到主机。

在用户按键后，键盘中断处理程序根据所按键的扫描码进行处理。它把字符键的扫描码和对应的 ASCII 码存到键盘缓冲区（某个确定的内存区域）；把功能键的扫描码存到键盘缓冲区；记录下控制键和双态键的状态；直接处理特殊请求键。

键盘 I/O 程序

BIOS 中提供键盘输入功能的程序被称为键盘 I/O 程序。每一个功能有一个编号。

功　　能	出　口　参　数	说　　明
AH=0 **从键盘读一个字符**	**AL=字符的 ASCII 码** **AH=字符的扫描码**	**如果无字符可读(键盘缓冲区空)，则等待；字符也包括功能键，对应 ASCII 码为 0**
AH=1 **判键盘是否有键可读**	**ZF=1 表示无键可读** **ZF=0 表示有键可读**	**不等待，立即返回** **AL=字符的 ASCII 码** **AH=字符的扫描码**
AH=2 **获取变换键当前状态**	**AL=变换键状态字节**	

在调用键盘 I/O 程序时，把功能编号置入 AH 寄存器，然后发出特定的调用指令“INT　16H”。调用返回后，从有关寄存器中取得出口参数。

键盘输入示例

```
CLEAR:
    MOV   AH, 1
    INT   16H                    ;键盘缓冲区空吗?
    JZ    .OK                    ;已清空，跳转
    MOV   AH, 0
    INT   16H                    ;从键盘缓冲区取走一个字符
    JMP   CLEAR                  ;直到键盘缓存区空为止
.OK:
    MOV   AH, 0
INT   16H                        ;等待键盘输入
```

显示方式

有两类显示方式：图形显示方式和文本显示方式。每一类显示方式还含有多种显示模式。

文本显示方式指以字符为单位显示的方式。字符通常指字母、数字、普通符号(如运算符号)和一些特殊符号（如菱形块和矩形块）。

现在几乎不采用文本显示方式，但这是最基本显示方式。

经典文本显示方式

最经典的文本显示模式是 25 行 80 列。在该文本显示模式下，显示器的屏幕被划分成 80 列 25 行，所以每一屏最多可显示 2000（80×25）个字符。

用行号和位号组成的坐标来定位屏幕上的每个可显示位置。左上角的坐标规定为（0,0），向右增加列号，向下增加行号，于是右下角的坐标便是（79,24）。

显示 I/O 程序

BIOS 中提供显示输出功能的程序被称为显示 I/O 程序。每一个功能有一个编号。

在调用显示 I/O 程序的某个功能时，应根据要求设置好入口参数，把功能编号置入 AH 寄存器，然后发出特定的调用指令“INT　10H”。调用返回后，从有关寄存器中取得出口参数。

在屏幕上显示的字符代码及其属性被依次保存在显示缓冲区（某个确定的内存区域）中。可以认为显示页号是显示缓冲区的编号。调用显示 I/O 程序的 5 号功能，可选择当前显示页。通常，总是使用第 0 页。

功　　能	入口参数	出口参数	说　　明
AH=2 **置光标位置**	**BH=显示页号** **DH=行号** **DL=列号**		**左上角坐标是(0,0)**
AH=8 **读取光标位置处的字符和属性**	**BH=显示页号**	**AH=属性** **AL=字符代码**	
AH=9 **将字符和属性写到光标位置处**	**BH=显示页号** **AL=字符代码** **BL=属性** **CX=字符重复次数**		**光标不移动**
AH=10 **将字符写到光标位置处**	**BH=显示页号** **AL=字符代码** **CX=字符重复次数**		**①光标不移动** **②不带属性**
AH=14 **TTY 方式显示**	**BH=显示页号** **AL=字符代码**		**光标处显示字符并后移光标；解释回车、换行、退格和响铃等控制符**

显示输出示例

利用 BIOS 在当前光标位置处显示字母 E，
然后光标移动到下一个显示位置处：

```
    MOV    BH, 0                    ;第 0 页
    MOV    AL, 'E'                  ;字符为 E
    MOV    AH, 14                   ;14 号功能
INT    10H                      ;TTY 方式显示
```

利用 BIOS 在当前光标位置处显示指定字符 2 次，
但光标并不移动：

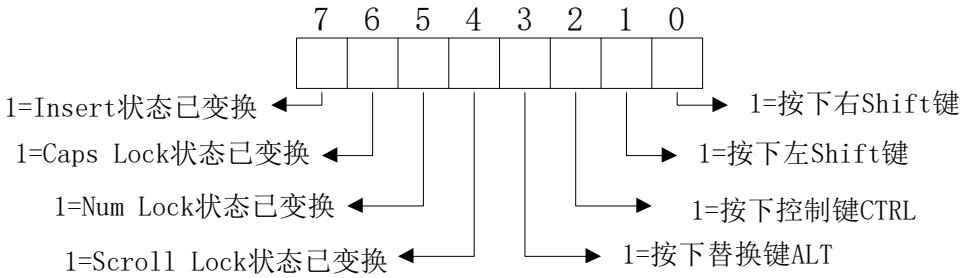
```
    MOV    BH, 0                    ;第 0 页
    MOV    CX, 2                    ;2 个
    MOV    AL, 'A'                  ;字符为 A
    MOV    AH, 10                   ;10 号功能
    INT    10H                      ;当前光标处按指定属性显示字符
```

7.1.3　应用举例

获得用户按键，显示所按键对应的字符，重复这一过程直到用户按下 SHIFT 键后结束程序运行。
演示程序 dp71.asm

```
    %define    L_SHIFT    00000010B
    %define    R_SHIFT    00000001B
    ;
    SECTION   TEXT
    BITS   16                       ;16 位代码
    ORG    100H                     ;COM 类型可执行程序
START:
    MOV    AH, 2                    ;取变换键状态字节
    INT    16H
    TEST   AL, L_SHIFT   +   R_SHIFT    ;判是否按下 SHIFT 键
```

```
        JNZ     OVER                ;按下，转
        MOV     AH, 1               ;判断是否有按键
        INT     16H
        JZ      START               ;无，继续下一轮检查
        MOV     AH, 0               ;取得所按键
        INT     16H
        MOV     BH, 0
        MOV     AH, 14              ;TTY 方式显示所按键
        INT     10H
        JMP     START               ;继续下一轮
    OVER:
        mov     ah, 4cH
        int     21H
```



演示程序 dp72.asm

在屏幕指定位置处显示彩色字符串。

```
        SECTION   TEXT
        BITS    16                  ;16 位代码
        ORG     100H                ;COM 类型可执行程序
    Begin:
        PUSH    CS
        POP     DS
        MOV     SI, Hello           ;SI=字符串首地址
        MOV     DL, [CurCol]        ;DL=光标列号
    MOV     AL, [SI]                ;取得待显示字符
    Lab1:
        MOV     DI, [Count]         ;行数（内循环的计数）
        MOV     DH, [CurLin]        ;DH=光标行号
        MOV     BL, [Color]         ;BL=显示属性初值
        MOV     BH, 0               ;在第 0 页显示
        MOV     CX, 1               ;显示 1 个字符
    Lab2:
        MOV     AH, 2
        INT     10H         ;设置光标位置
        ;
        MOV     AH, 9
        INT     10H         ;显示字符（AL）
        ;
        INC     DH          ;调整光标的行
        INC     BL          ;调整显示属性
        DEC     DI          ;行数减 1
        JNZ     Lab2        ;不为 0，继续下一行
        INC     DL                  ;调整光标的列
        INC     SI                  ;指向下一个待显示字符
        MOV     AL, [SI]            ;取得待显示字符
```

```
    OR      AL, AL                  ;字符串结束标志？
    JNZ     Lab1                    ;否，继续显示
    MOV     DH, 19
    MOV     DL, 0
    MOV     AH, 2
    INT     10H                     ;重新设置光标到位置（19,0）
    ;
    mov     ah, 4ch                 ;仍然调用 DOS 功能，结束程序
    int     21h
    ;
Hello       db      "Hello,world",0     ;显示信息
CurLin      db      5                   ;起始光标行号
CurCol      db      8                   ;起始光标列号
Color       db      0x07                ;每行起始显示属性
Count       dw      6                   ;行数
```

7.2 PC 机启动和磁盘 I/O

7.2.1 PC 机启动过程

启动主要步骤

加电自检（POST）
配置 BIOS
引导操作系统

加电自检（Power On Self Test）

加电（RESET）开始执行的地址 FFFF:0000

```
甲版本相关指令              地址            机器码
JMP     F000:FFD1           ;FFFF:0000      EAD1FF00F0
JMP     E878                ;F000:FFD1      E9A4E8
MOV     AL,30               ;F000:E878      B030

乙版本相关指令              地址            机器码
JMP     F000:E05B           ;FFFF:0000      EA5BE000F0
XOR     AX, AX              ;F000:E05B      31C0
```

检测关键部件，包括 CPU、低端内存、中断控制器等
检测显示控制卡，确定显卡 BIOS 程序，并初始化显卡

测试所有的 RAM
检测标准硬件设备，包括硬盘、CD-ROM、串口和并口等，确定对应的 BIOS 程序，并初始化
查找确定即插即用设备，并初始化

配置 BIOS

获取机器系统的基本配置信息
获取相关设备的 I/O 端口地址和参数等

设置相关设备的 I/O 程序入口点（中断向量）

引导操作系统

按照“启动顺序”，读取主引导记录（MBR）
读取主引导记录到内存 0000:7C00 开始处
转 0000:7C00 处执行

7.2.2 磁盘 I/O

磁盘

磁盘是计算机系统中重要的外部存储设备。与内部存储器相比，它容量大，但存取速度慢。
现在只使用硬磁盘（硬盘），软磁盘几乎没有了。
现在硬盘又可分为机械硬盘（HDD 传统硬盘）、固态硬盘（SSD 盘，新式硬盘）、混合硬盘（HHD）。
绝大多数硬盘都是固定硬盘，被永久性地密封固定在硬盘驱动器中。

硬盘结构

机械硬盘由多个铝制或者玻璃制的碟片组成。碟片外覆盖有铁磁性材料。碟片成为磁片。
磁片上分布若干同心圆的磁道；磁道又分为若干扇区。
不同磁头下（磁片上）的相同半径的磁道，构成柱面。

扇区及其地址

扇区作为物理介质，指磁道上的一段存储数据的弧形区域。
扇区作为度量单位，表示 512 字节。通常一个扇区可以存储 512 字节的数据。
传统上，通过柱面（Cylinder）、磁头（Head）、扇区号（Sector）这三维地址来指定磁盘上的某个扇区。一个硬盘的柱面数、磁头数和每个磁道的扇区数，就决定了硬盘的最大容量。
现在，可以采用绝对扇区号（一维地址）来指定磁盘上的某个扇区。

磁盘 I/O 程序

BIOS 中提供磁盘输入输出功能的程序被称为磁盘 I/O 程序。它提供磁盘(包括硬盘和软盘)的复位、读写、校验和格式化等功能。每一个功能有一个编号。
在调用磁盘 I/O 程序时，按调用的功能，准备好读写缓存区中的数据,设置好相应的参数，把功能编号置入 AH 寄存器，然后发出特定的调用指令“INT 13H”。调用返回后，从有关寄存器中取得出口参数，从读写缓冲区中取得数据。

功 能	入口参数	出口参数	说 明
AH=2 **读扇区**	**ES=缓冲区地址段值** **BX=缓冲区地址偏移** **AL=扇区数** **DH=磁头号** **DL=驱动器号** **CH=柱面号(低 8 位)** **CL(高 2 位)=柱面号(高 2 位)** **CL(低 6 位)=扇区号**	**进位标志 CF=0 表示读成功，缓冲区含有读入的数据；CF=1 表示读出错，AX 中存放出错状态。**	**驱动器号:** **C 盘=80H** **D 盘=81H** **柱面号、磁头号、扇区号，指定读写首个扇区的地址。**
AH=3 **写扇区**	同上	**进位标志 CF=0 表示写成功；CF=1 表示写出错，AX 中存放出错状态。**	

扇区采用 CHS 编址方式

扇区采用 CHS 编址模式（柱面、磁头和扇区的编址模式）时，最多只能访问 8GB 左右的硬盘。

1024 * 256 * 64（扇区）= 1024 * 1024 * 16（扇区）
= 16M （扇区）
= 8G （字节）

磁盘地址数据包（DAP）

磁盘地址数据包（Disk Address Packet），是供扩展磁盘 I/O 程序所使用的数据结构。

DAP 的结构如下：

```
struct   DiskAddressPacket
{
    BYTE    PacketSize;    //数据包尺寸(16 字节)
    BYTE    Reserved;      //保留==0
    WORD    BlockCount;    //传输数据块个数(以扇区为单位)
    DWORD   BufferAddr;    //传输缓冲地址(segment:offset)
    QWORD   BlockNum;      //起始扇区的逻辑块号（一维地址）
};
```

磁盘 I/O 程序（扩展）

扇区采用逻辑块编址模式（Logical Block Addressing）时，可以访问“无限大”的硬盘。

在 LBA 编址模式下，扇区地址不再表示硬盘中的实际物理地址（柱面、磁头和扇区）。LBA 编址方式将 CHS 这种三维编址方式转变为一维的线性编址，它把硬盘所有的物理扇区的 C/H/S 编号通过一定的规则转变为一线性的编号，系统效率得到大大提高，避免了烦琐的磁头/柱面/扇区的寻址方式。在访问硬盘时，由硬盘控制器再将这种逻辑地址转换为实际硬盘的物理地址。

扩展读

入口:

AH = 42h
DL = 驱动器号
DS:SI = 磁盘地址数据包(DAP)

返回:

CF = 0, AH = 0 成功
CF = 1, AH = 错误码

功能：这个调用将磁盘上的数据读入内存。如果出现错误，DAP 的 BlockCount 项中则记录了出错前实际读取的数据块个数。

扩展写

入口:

AH = 43h
AL = 控制写校验
DL = 驱动器号
DS:SI = 磁盘地址数据包(DAP)

返回:

CF = 0, AH = 0 成功
CF = 1, AH = 错误码

功能：这个调用将内存中的数据写入磁盘。如果出现错误, DAP 的 BlockCount 项中则记录了出错前实际写

入的数据块个数。

磁盘 I/O 示例

利用扩展的磁盘 I/O 程序，读取磁盘上的首个（第 0 个）扇区，到内存的 0000:7C00H 处

```
MOV     AX, CS
MOV     DS, AX
MOV     SI, DiskAP
MOV     DL, 80H
MOV     AH, 42H
INT     13H

DiskAP:
    DB      10H         ;DAP 尺寸
    DB      0           ;保留
    DW      1           ;扇区数
    DW      7C00H       ;缓冲区偏移
    DW      0000H       ;缓冲区段值
    DD      0           ;起始扇区号的低 4 字节
    DD      0           ;起始扇区号的高 4 字节
```

7.2.3 主引导记录

关于主引导记录

主引导记录（MBR，Main Boot Record）是位于启动磁盘首个扇区的一段引导（Loader）代码。所谓启动磁盘指准备启动操作系统的磁盘。所谓首个扇区指磁盘上的逻辑块号（LBA）为 0 的扇区，也就是 CHS 地址（0 道、0 面、1 扇区）的扇区。

主引导记录负责引导操作系统。从磁盘上当前活动分区读取操作系统的引导程序，然后转引导程序。

传统的主引导记录由三部分组成：

主引导程序（446 字节）

主引导程序的执行步骤

自身腾挪:自身代码块搬移，为读入操作系统引导程序做准备

识别当前活动分区

读取操作系统引导程序

转操作系统引导程序

自身腾挪

Windows7 之 32 位系统主引导程序开始代码

```
0000:7C00 33C0              XOR     AX,AX
0000:7C02 8ED0              MOV     SS,AX
0000:7C04 BC007C            MOV     SP,7C00
0000:7C07 8EC0              MOV     ES,AX
0000:7C09 8ED8              MOV     DS,AX
0000:7C0B BE007C            MOV     SI,7C00
0000:7C0E BF0006            MOV     DI,0600
0000:7C11 B90002            MOV     CX,0200
```

```
0000:7C14 FC            CLD             复制 MBR（512 字节）从 0000:7C00 处开始到 0000:0600 处开始
0000:7C15 F3            REPZ
0000:7C16 A4            MOVSB
0000:7C17 50            PUSH  AX
0000:7C18 681C06        PUSH  061C      跳转到 0000:061C 处搬移后，继续执行
0000:7C1B CB            RETF

0000:7C1C FB            ...
```

DOS6.22 系统主引导程序开始代码

```
0000:7C00 FA            CLI
0000:7C01 31C0          XOR   AX,AX
0000:7C03 8ED8          MOV   DS,AX
0000:7C05 8EC0          MOV   ES,AX
0000:7C07 8ED0          MOV   SS,AX
0000:7C09 BC007C        MOV   SP,7C00
0000:7C0C FB            STI
0000:7C0D FC            CLD
0000:7C0E 89E6          MOV   SI,SP     复制 MBR（512 字节）从 0000:7C00 处开始到 0000:0600 处开始
0000:7C10 BF0006        MOV   DI,0600
0000:7C13 B90001        MOV   CX,0100
0000:7C16 F3            REPZ
0000:7C17 A5            MOVSW           跳转到 0000:06DC 处搬移后，继续执行
0000:7C18 EADC060000    JMP   0000:06DC
```

主引导程序代码分析

识别当前活动分区

所谓分区指磁盘上的一段连续区域。引入分区概念后，一个物理硬盘可分为几个较小的逻辑分区。在不同的磁盘分区，可以安装不同的操作系统，这样可以在硬盘上安装多个操作系统。当前使用操作系统所在的分区，被称为活动分区。

主引导记录中含有一张磁盘分区表。分区表含有 4 项，可以表示 4 个分区。每项 16 个字节，由分区标记、分区起始扇区地址等信息构成。识别分区标记，可以识别出活动分区。

Windows7 之 32 位系统主引导程序代码

```
0000:0661 666800000000  PUSH  00000000
0000:0667 66FF7608      PUSH  dword ptr [BP+08]   在堆栈中，形成 16 字节的 DAP
0000:066B 680000        PUSH  0000
0000:066E 68007C        PUSH  7C00
0000:0671 680100        PUSH  0001
0000:0674 681000        PUSH  0010
0000:0677 B442          MOV   AH,42
0000:0679 8A5600        MOV   DL,[BP+00]      取得驱动器号（硬盘号）
0000:067C 8BF4          MOV   SI,SP
0000:067E CD13          INT   13
0000:0680 9F            LAHF
0000:0681 83C410        ADD   SP,+10          平衡堆栈，弹出 16 字节
```

```
0000:0684 9E              SAHF
0000:0685 EB14            JMP     069B
```

在堆栈中，形成 16 字节的 DAP

```
    0010        ;DAP 尺寸和保留
    0001        ;读扇区数
    7C00        ;缓冲区偏移
    0000        ;缓冲区段值
    xxxxxxxx    ;起始扇区号的低 4 字节
    00000000    ;起始扇区号的高 4 字节
```

```
0000:072A EA007C0000      JMP    0000:7C00        假设一切正常，不考虑读出错等情况
```

7.3 虚拟机

7.3.1 虚拟化技术

模拟和仿真

模拟（simulation）与仿真（emulation）都是对真实的事或物的模仿。英文单词的中文翻译应该都可以。

模拟更在乎结果，输入是真的，过程是假的。例如，模拟的 CPU、软件等。

仿真更在乎过程，过程是真的，输入是假的。例如，产生仿真报文的过程。

另一种观点：模拟是以模型为基础的拟合；仿真是以功能为基础的效仿。

虚拟化技术（王鹏《云技术与大数据技术》）

广义地看，虚拟化技术就是一种逻辑简化技术，实现物理层向逻辑层的变化。采用虚拟化技术后，一个系统对外表现出的运动方式是一种逻辑化的运动方式，而不是真实的物理运动方式。采用虚拟化技术能实现对物理层运动复杂性的屏蔽，使系统对外运行状态呈现出简单的逻辑运行状态。

虚拟化：

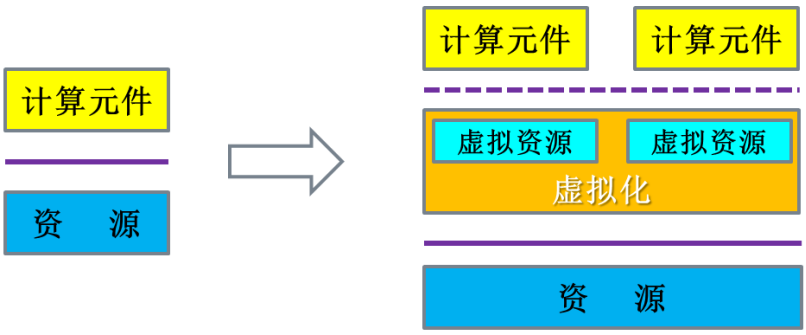
对象：计算机的各种资源（包括基础设施、系统和软件）

过程：将各种资源进行抽象、转换

结果：为这些资源提供标准的接口来接收输入和提供输出，使用户能以更好的方式应用这些资源。

虚拟化技术（任永杰《KVM 虚拟化技术实战与原理解析》）

虚拟化是指计算元件在虚拟的基础上面而不是真实的基础上运行，是一个为了简化管理、优化资源的解决方案。



将一般的计算模型抽象成为一定的物理资源和运行于之上的计算元件，它们之间通过定义的物理资源接口进行交互。

资源可以表现为各种各样的形式：操作系统及其系统调用作为资源 x86 平台包括处理器、内存和外设作为资源

实现虚拟化的关键点：虚拟化层必须能够截获计算元件对物理资源的直接访问，并将其重定向到虚拟资源。

根据虚拟化层通过纯软件的方法，还是利用物理资源提供的机制来实现这种“截获并重定向”，把虚拟化分为软件虚拟化和硬件虚拟化。

软件虚拟化，就是利用纯软件的方法在现有的物理平台上实现对物理平台访问的截获和模拟。

硬件虚拟化，就是物理平台本身提供了对特殊指令的截获和重定向的硬件支持机制。硬件帮助软件实现对关键硬件资源的虚拟化，从而提升性能。

虚拟机

虚拟机（Virtual Machine）指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统

流行的虚拟机软件有 VMware、VirtualBox 和 VirtualPC，它们都能在 Windows 系统上虚拟出多个计算机。

7.3.2 虚拟机 VirtualBox

VirtualBox 简介（改编自百度百科）

VirtualBox 是一款开源的虚拟机软件。

它不仅具有丰富的特色，而且性能也很优异。它提供用户在 32 位或 64 位的 Windows、Linux 及 Solaris 操作系统上虚拟其它 x86 的操作系统。

用户可以在 VirtualBox 上安装并且运行 Solaris、Windows、DOS、Linux 等系统作为客户机操作系统。

VirtualBox 特点

VirtualBox 作为虚拟机管理器，使用比较方便。

能够创建和管理多台虚拟机。在这些虚拟机上，能够分别安装不同的客户机操作系统。每个客户机系统都能够独立运行，就像不同的机器，可以独立地打开、暂停与停止。

宿主机操作系统与客户机操作系统之间能相互通信，而且还能够同时使用网络。

在 VirtualBox 创建的虚拟机上，可以不安装操作系统，直接引导和执行特定的程序。

利用这一特点，可以在虚拟机上直接运行纯目标代码。采用这种方式运行程序，优点是可以不受操作系统的约束，“为所欲为”；缺点是没有操作系统可以依靠，除了利用 BIOS 外，其他都必须“自力更生”。

7.3.3 主引导记录示例

生成一个新的主引导记录：仅仅在屏幕指定位置处显示字符串“Hello world!”。

编写源程序

显示字符串的代码

字符串信息

标记（55AA）

生成纯二进制代码

写入虚拟磁盘文件

启动虚拟机

编写源程序

```
    section    text                ;开始名为 text 的段
    bits    16                      ;16 位段模式
BEGIN:
    MOV    AX, CS
    MOV    DS, AX                   ;当前数据段与代码段一致
    ;
    MOV    BH, 0                    ;指定显示页 0
    MOV    DH, 5                    ;光标行号
    MOV    DL, 8                    ;光标列号
    MOV    AH, 2
    INT    10H
    ;
    CLD                             ;字符串操作方向
```

```
    MOV     SI, hello                ;指向字符串首（代码段的相对地址）
    ADD     SI, 7C00H                ;指向字符串首（内存中的固定地址）
LAB1:
    LODSB                            ;取一个字符
    OR      AL, AL                   ;判断结束标记
    JZ      LAB2                     ;是，跳转结束
    MOV     AH, 14
    INT     10H                      ;TTY 方式显示字符
    JMP     LAB1                     ;继续
LAB2:
OVER:
    JMP     OVER                     ;进入无限循环——有意进入无限循环！不做别的工作!!
    ;
hello   db     "Hello world!", 0
        ;
times   510 - ($ - $$) db     0      ;填充 0，直到 510 字节
        db     55h, 0aah              ;最后 2 字节，共计 512 字节
```

$ 表示当前位置的偏移
$$ 表示当前段开始位置的偏移
times 是汇编指示，表示重复
这里就是重复伪指令“db 0”，重复次数：510-($ - $$)

生成纯二进制代码
利用汇编器 NASM，生成纯二进制代码文件的方法：

```
nasm   dp74.asm   -f bin   -o dp74
```

写入虚拟磁盘文件
利用 VHDwriter，把纯二进制代码文件写入 VirtualBOX 的虚拟机 VM_ASM 的对应虚拟磁盘文件。

启动虚拟机
在 VirtualBox 中，启动虚拟机 VM_ASM，可得运行结果：

7.3.4 引导程序设计

新的引导程序
生成一个新的主引导记录，作为引导程序：引导另一个程序（假设存放在某个扇区中）

编写源程序
自身腾挪（设把被引导程序装到 7C00 开始区域）
从磁盘指定位置处，读入被引导程序
跳转到刚装载的被引导程序
生成纯二进制代码
写入虚拟磁盘文件（引导程序和被引导程序）
启动虚拟机

源程序

```
    section    text
    bits    16
BEGIN:
    MOV    AX, CS
    MOV    SS, AX
    MOV    SP, 7C00H
    MOV    DS, AX       ;源数据段与代码段一致
    MOV    SI, BEGIN    ;指向源字符串首（相对地址）
    ADD    SI, 7C00H    ;指向源字符串首（绝对地址）
    PUSH   0060H
    POP    ES           ;目标数据段的段值为 0060H
    MOV    DI, 0        ;目标段的偏移
    CLD                 ;字符串操作方向
    MOV    CX, 200H     ;自身 512 字节
    PUSH   ES
    PUSH   BEGIN2
    REP    MOVSB
    RETF
BEGIN2:
    PUSH   CS
    POP    DS
    MOV    DX, mess1
    CALL   PutStr
    CALL   GetChar
    ;
    MOV        SI, DiskAP          ;指向 DAP
    MOV        DL, 80H             ;C 盘
    MOV        AH, 42H             ;扩展的读——从 C 盘的指定扇区，读入被引导程序，到 0000:7C00H 处
    INT    13H                 ;
    ;
    MOV    AX, 0
    MOV    ES, AX
    CMP    WORD [ES:7DFEH], 0AA55H
    JNZ    OVER
    PUSH   WORD 0
    PUSH   WORD 7C00H          跳转到被引导程序 跳转到 0000:7C000H 处
RETF
OVER:
        MOV    DX, mess2
        CALL   PutStr
        JMP    $               如果被引导程序有误，进入无限循环
        ;
GetChar:
        MOV    AH, 0
        INT    16H
        RET
        ;
PutStr:
```

```
        MOV     SI, DX
LAB1: LODSB                     ;取一个字符
        OR      AL, AL          ;判断结束标记
        JZ      LAB2            ;是，跳转结束
        MOV     AH, 14
        INT     10H             ;TTY 方式显示字符
        JMP     LAB1            ;继续
LAB2: RET
DiskAP:
      DB      10H               ;DAP 尺寸
      DB      0                 ;保留
      DW      1                 ;扇区数 设被引导程序只有 1 个扇区
      DW      7C00H             ;缓冲区偏移
      DW      0000H             ;缓冲区段值 设被引导程序装载区域的地址
      DD      123               ;磁盘起始绝对扇区号的低 4 字节
      DD      0                 ;磁盘起始绝对扇区号的高 4 字节 设被引导程序在磁盘上的位置
      ;
mess1       db      "Press any key.....", 0
mess2       db      "Error.....", 0
      ;
times     510 - ($ - $$) db     0     ;填充 0，直到 510 字节
          db      55h, 0aah           ;最后 2 字节，共计 512 字节 特定标记
```

生成纯二进制代码

写入虚拟磁盘文件

引导程序

被引导程序

启动虚拟机