

## 调试汇编代码

用Bochs在汇编级调试操作系统很简单，只需要运行“dbg-asm”，然后就得到了如下图所示的调试界面。

```

C:\Program Files\Bochs-2.3.6>\"C:\Program Files\Bochs-2.3.6\bochsdbg\" -q -f bochs.bxrc
000000000000i[APIC?] local apic in  initializing
=====
                        Bochs x86 Emulator 2.3.6
                        Build from CVS snapshot, on December 24, 2007
=====
000000000000i[      ] reading configuration from bochs.bxrc
000000000000i[      ] installing win32 module as the Bochs GUI
000000000000i[      ] using log file bochsout.txt
Next at t=0
<0> [0xffffffff] f000:fff0 (unk. ctxt): jmp far f000:e05b          ; ea5be000f0
<bochs:1> _

```

图1 Bochs调试操作系统的界面

此时是暂停在BIOS中。而我们的代码是从0x7C00位置开始的，所以先要在那里设一个断点，然后继续运行到断点：

```

break 0x7c00
continue
(0) Breakpoint 1, 0x00007c00 in ?? ()
Next at t=4967728
(0) [0x00007c00] 0000:7c00 (unk. ctxt): mov ax, 0x07c0          ; b8c007

```

接下来可以用命令help来查看调试系统的各种基本命令，这里给出了一些常用的命令

## 执行控制指令

c/cont/continue	连续执行
s/step/stepi [count]	执行count条指令，默认为1条，会跟进到函数和中断调用的内部
p/n/next [count]	执行count条指令，默认为1条，但跳过函数和中断调用
Ctrl+C	停止执行，并回到命令行提示符下
q/quit/exit	退出调试和执行

## 断点设置命令

vb/vbreak seg:offset	在虚拟地址上设置指令断点，其中seg和offset可以是以0x开始的十六进制数，或十进制，或者是以0开头的八进制数
lb/lbreak addr	在线性地址上设置断点，addr同上面的seg和offset
b/break/pb/pbreak addr	在物理地址上设置断点

info break	显示当前所有断点的信息
d/del/delete n	删除一个断点

## 内存操作指令

x /nuf addr 检查位于线性地址addr处的内存内容

xp /nuf addr 检查位于物理地址addr处的内存内容

其中参数n、u、f分别表示：

n为要显示内存单元的计数值，默认为1

u表示单元大小，默认值为w

b (bytes)	1字节
h (halfwords)	2字节
w (words)	4字节
g (giantwords)	8字节

f为显示格式，默认为x

x (hex)	显示为十六进制数
d (decimal)	显示为十进制数
u (unsigned)	显示为无符号十进制数
o (octal)	显示为八进制数
t (binary)	显示为二进制数
c (char)	显示为对应的字符

## 信息显示和CPU寄存器操作命令

r/reg/regs/registers 列表显示CPU寄存器及其内容

set \$reg=val 修改某寄存器的内容。除段寄存器和标志寄存器以外的寄存器都可以修改，如set \$eax=0x01234567

creg 列出所有的CR0-CR4寄存器

sreg 列出CPU全部状态信息，包括各个段选择子（cs，ds等）以及ldtr和gdtr等。

print-stack 打印堆栈情况。

info tab 显示页表

## 反汇编命令

u/disasm/disassemble start end，反汇编给定线性地址范围的指令。也可以是u /10 反汇编从当前地址开始的10条指令。如下图

```
Bochs for Windows - Console
Bochs x86 Emulator 2.3.6
Build from CVS snapshot, on December 24, 2007
=====
00000000000i[      ] reading configuration from bochs.bxrc
00000000000i[      ] installing win32 module as the Bochs GUI
00000000000i[      ] using log file bochsout.txt
Next at t=0
<0> [0xfffffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b          ; ea5be000f0
<bochs:1> vb 0x0000:0x7c00
<bochs:2> c
<28688> Breakpoint 4294967276, in 0000:7c00 (0x00007c00)
Next at t=6131376
<0> [0x00007c00] 0000:7c00 (unk. ctxt): jmp far 07c0:0005          ; ea0500c007
<bochs:3> info sreg
cs:s=0x0000, dl=0x0000ffff, dh=0x00009b00, valid=1
ds:s=0x0000, dl=0x0000ffff, dh=0x00009300, valid=1
ss:s=0x0000, dl=0x0000ffff, dh=0x00009300, valid=7
es:s=0x0000, dl=0x0000ffff, dh=0x00009300, valid=1
fs:s=0x0000, dl=0x0000ffff, dh=0x00009300, valid=1
gs:s=0x0000, dl=0x0000ffff, dh=0x00009300, valid=1
ldtr:s=0x0000, dl=0x00000000, dh=0x00000000, valid=0
tr:s=0x0000, dl=0x00000000, dh=0x00000000, valid=0
gdtr:base=0x000faeb2, limit=0x30
idtr:base=0x00000000, limit=0x3ff
<bochs:4> u /5
00007c00: <          >: jmp far 07c0:0005          ; ea0500c007
00007c05: <          >: mov ax, cs          ; 8cc8
00007c07: <          >: mov ds, ax          ; 8ed8
00007c09: <          >: mov es, ax          ; 8ec0
00007c0b: <          >: mov ss, ax         ; 8ed0
<bochs:5> _
```

图2 用Bochs调试操作系统