

# 代码阅读

---

为什么`mov bp, ax`后，`int 10h`就能够取到`BootMessage`的地址了？

答案

因为这个指令之后，`es:bp`的值就是"Hello OS"字符串的首地址。

过程

16号中断是用来触发一个关于视频服务的功能的。当功能号是0x13时，打印字符串，`es:bp`是字符串地址。

`mov bp, ax`上一句是`mov ax, BootMessage`，即`ax=BootMessage`，即字符串首地址。`mov bp, ax`只是把字符串首地址赋值给`bp`而已。

而`es`在函数调用之前被赋值成了`CS`就是段首地址。

这样，`es:bp`就是`BootMessage`所在的实际地址。16号中断里就可以这样找到实际的`BootMessage`地址了。

[https://zh.wikipedia.org/wiki/INT\\_10H](https://zh.wikipedia.org/wiki/INT_10H)

运行到这行代码的时候`ax`里面的值是多少？

答案

0x7c1e

过程

由`ndisasm -o 0x7c00 boot.bin >> disboot.asm`得到反汇编代码`disboot.asm`。

在`./disboot.asm`地址00007C0B处，指令为`mov ax, 0x7c1e`，即将立即数0x7c1e mov给了`ax`。过于明显不需要实机验证。

这个值是不是`BootMessage`所在内存中的位置（即相对地址还是绝对地址）？

答案

在此例中是其所在内存中的位置。

过程

讲道理，此时`bp`只是相对地址，相对地址需要加上段地址才是实际地址，但是在此例中，计算机刚启动，所有段寄存器都是0，所以`bp`的值就是其字符串的绝对地址。

使用`bochsdbg`工具进行单步调试，在执行`int 10h`前（0x7C1B）设置断点，查看寄存器值和物理内存0x7c1e位置的值。

设置断点并执行到int 10h

```
<bochs:1> b 0x7c1b
<bochs:2> c
000000046621[BIOS] $Revision: 13073 $ $Date: 2017-02-16 22:43:52 +0100 (Do, 16. Feb 2017) $
000003180501[EBD] reset-disable command received
000003208191[BIOS] Starting rombios32
000003212571[BIOS] Shutdown flag 0
000003218401[BIOS] ram_size=0x2000000
000003222611[BIOS] ram_end=32M
000003226291[BIOS] Found 1 cpu(s)
000003764131[BIOS] bios_table_addr: 0x000f9cd8 end=0x000fcc00
000006084801[WINGUP] dimension update x=720 y=400 fontheight=16 fontwidth=9 bpp=8
000007042001[PCI] i440FX PMW write to PMW register 59 (TLB Flush)
000010321371[P2ISA] PCI IRQ routing: PIRQ# set to 0x0b
000010321561[P2ISA] PCI IRQ routing: PIRQ# set to 0x09
000010321751[P2ISA] PCI IRQ routing: PIRQ# set to 0x0b
000010321941[P2ISA] PCI IRQ routing: PIRQ# set to 0x09
000010322041[P2ISA] write: ELCR2 = 0x0a
000010329741[BIOS] PIIX3/PIIX4 init: elcr=00 0a
000010400971[BIOS] PCI: bus=0 devfn=0x00: vendor_id=0x8086 device_id=0x1237 class=0x0000
000010429741[BIOS] PCI: bus=0 devfn=0x01: vendor_id=0x8086 device_id=0x7000 class=0x0001
000010450941[BIOS] PCI: bus=0 devfn=0x09: vendor_id=0x8086 device_id=0x7010 class=0x0101
000010453231[PIDE] new BM-DMA address: 0xc000
000010459391[BIOS] region 4: 0x000c000
000010470531[BIOS] PCI: bus=0 devfn=0x0a: vendor_id=0x8086 device_id=0x7020 class=0x0c03
000010481571[UHCI] new base address: 0xc020
000010487731[BIOS] region 4: 0x000c020
000010489011[UHCI] new irq line = 9
000010507961[BIOS] PCI: bus=0 devfn=0x0b: vendor_id=0x8086 device_id=0x7113 class=0x0600
000010510281[ACPI] new irq line = 11
000010510401[ACPI] new irq line = 9
000010510621[ACPI] new SM base address: 0xb000
000010510791[ACPI] new SM base address: 0xb100
000010511071[PCI] setting SMRAM control register to 0x4a
000012152001[CPUE] Enter to System Management Mode
000012152001[CPUE] enter system management mode: temporary disable VMX while in SMM mode
000012152101[CPUE] RSM: Resuming from System Management Mode
000013792311[PCI] setting SMRAM control register to 0x0a
000013941301[BIOS] HP table addr=0x000f9d00 HPC table addr=0x000f9ce0 size=0xc8
000013959601[BIOS] SMBIOS table addr=0x000f9d00
000013981411[BIOS] ACPI tables: RSDP addr=0x000f9ee0 ACPI DATA addr=0x01ff0000 size=0xf72
000014013531[BIOS] Firmware waking vector 0x1fff00cc
000014021401[PCI] i440FX PMW write to PMW register 59 (TLB Flush)
000014030711[BIOS] bios_table_cur_addr: 0x000f9f04
000015314801[VBIOS] VGBios $Id: vgbios.c,v 1.76 2013/02/10 08:07:03 vruppert Exp $
000015315591[EXVGA] VBE known Display Interface 0xc0
000015315911[EXVGA] VBE known Display Interface 0xc5
000015345101[VBIOS] VBE Bios $Id: vbe.c,v 1.65 2014/07/08 18:02:25 vruppert Exp $
000140401891[BIOS] Booting from 0000:7c00
(0) Breakpoint 1: 0x0000000000007c1b in ?? ()
Next at t=14040254
(0) [0x000000007c1b] 0000:7c1b (unk. ctx): int 0x10 ; cd10
```

所有段寄存器的值都是0，当然包括es。

```
sreg
es:0x0000, dh=0x00009300, dl=0x0000ffff, valid=1
Data segment, base=0x00000000, limit=0x0000ffff, Read/Write, Accessed
cs:0x0000, dh=0x00009300, dl=0x0000ffff, valid=1
Data segment, base=0x00000000, limit=0x0000ffff, Read/Write, Accessed
ss:0x0000, dh=0x00009300, dl=0x0000ffff, valid=7
Data segment, base=0x00000000, limit=0x0000ffff, Read/Write, Accessed
ds:0x0000, dh=0x00009300, dl=0x0000ffff, valid=1
Data segment, base=0x00000000, limit=0x0000ffff, Read/Write, Accessed
fs:0x0000, dh=0x00009300, dl=0x0000ffff, valid=1
Data segment, base=0x00000000, limit=0x0000ffff, Read/Write, Accessed
gs:0x0000, dh=0x00009300, dl=0x0000ffff, valid=1
Data segment, base=0x00000000, limit=0x0000ffff, Read/Write, Accessed
ldtr:0x0000, dh=0x00008200, dl=0x0000ffff, valid=1
tr:0x0000, dh=0x00008b00, dl=0x0000ffff, valid=1
gdtr:base=0x00000000000000f9a37, limit=0x30
idtr:base=0x0000000000000000, limit=0x3ff
```

bp的值是7c1e。bp即rbp的低8位。

```
<bochs:11> info cpu
rax: 00000000_00001301 rcx: 00000000_00090010
rdx: 00000000_00000000 rbx: 00000000_0000000c
rsp: 00000000_0000ffd4 rbp: 00000000_00007c1e
rsi: 00000000_000e0000 rdi: 00000000_0000ffac
```

所以es:bp的值就是0x7c1e，也就是10h中断所认为的字符串的位置。

物理地址0x7c1e的值如下，以char格式打出9个字节，可以看到就是Hello OS。

```
<bochs:10> xp /9bc 0x7c1e
[bochs]:
0x0000000000007c1e <bogus+ 0>: H e l l o 0 S
0x0000000000007c26 <bogus+ 8>: \0
```