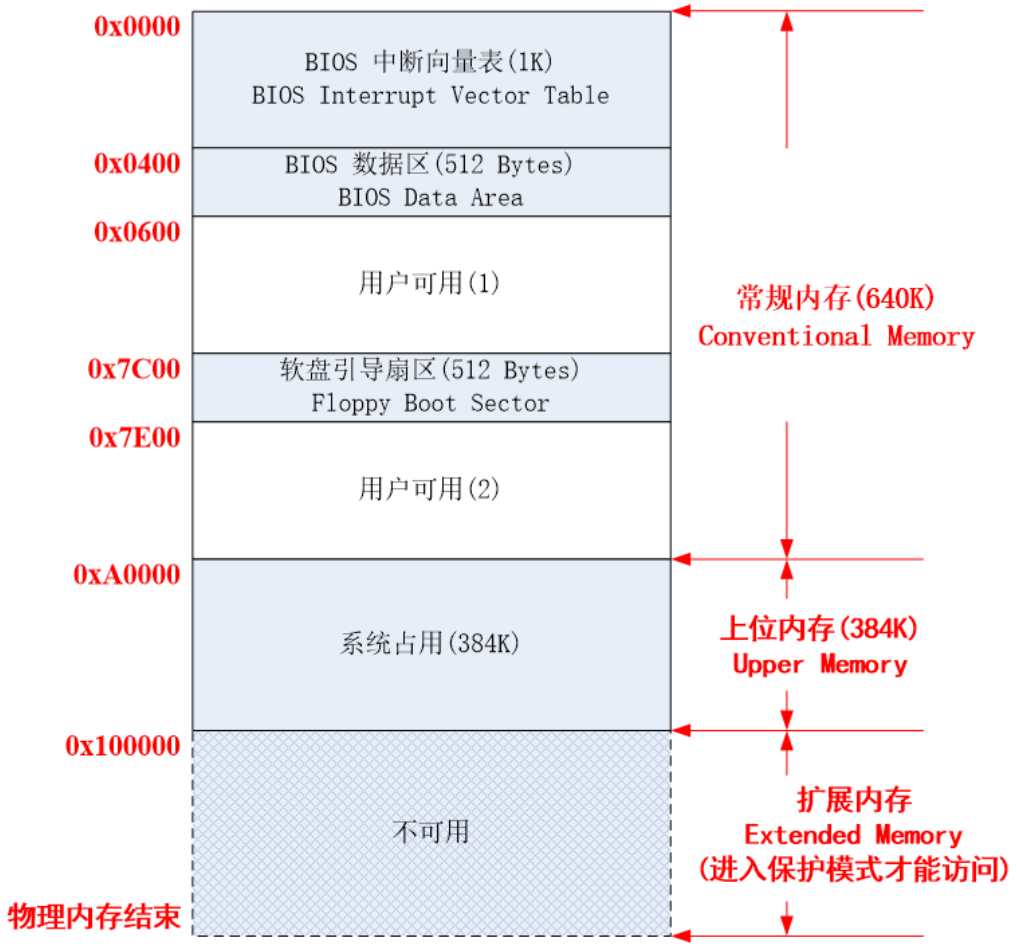
EOS操作系统命令（基于Bochs实现）

在console窗口上输入调试命令，在display窗口上显示操作系统的输出

console窗口显示的**指令相关信息**：

```
(0) context not implemented because BX_HAVE_HASH_MAP=0
[0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b          ; ea5be000f0
```

- `[0xfffffff0]` 表示指令的 `物理地址`
- `f000:fff0` 表示该条指令所在的 `逻辑地址`,(段地址：偏移地址)————>转换为物理地址后见上
- `jmp far f000:e05b` 是该条指令的 `反汇编代码`
- `ea5be000f0` 是该指令的内容（十六进制），单位为？？？



查看主要寄存器和内存中的数据

sreg：显示当前CPU中各个段寄存器的值（`s = XXX`）

```
(0) context not implemented because BX_HAVE_HASH_MAP=0
[0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b         ; ea5be000f0
<bochs:1> sreg
cs:s=0xf000, dl=0x0000ffff, dh=0xff0093ff, valid=1
ds:s=0x0000, dl=0x0000ffff, dh=0x00009300, valid=1
ss:s=0x0000, dl=0x0000ffff, dh=0x00009300, valid=1
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=0x0000ffff, dh=0x00008200, valid=1
tr:s=0x0000, dl=0x0000ffff, dh=0x00008b00, valid=1
gdtr:base=0x00000000, limit=0xffff
```

r：显示当前通用寄存器的值

```
<bochs:2> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:00000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000fff0
eflags 0x00000002
id vip vif ac vm rf nt IOPL=0 of df if tf sf zf af pf cf
```

xp /<字节数> 地址：查看从xx地址开始 的xx个字节 的`物理内存`

```
<bochs:3> xp /1024b 0x0000
[bochs]:
0x0000000000000000 <bogus+       0>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000008 <bogus+       8>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000010 <bogus+      16>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000018 <bogus+      24>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000020 <bogus+      32>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000028 <bogus+      40>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000030 <bogus+      48>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000038 <bogus+      56>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000040 <bogus+      64>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000048 <bogus+      72>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000050 <bogus+      80>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000058 <bogus+      88>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000060 <bogus+      96>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000068 <bogus+     104>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000070 <bogus+     112>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000078 <bogus+     120>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000080 <bogus+     128>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000088 <bogus+     136>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000090 <bogus+     144>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x0000000000000098 <bogus+     152>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x00000000000000a0 <bogus+     160>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x00000000000000a8 <bogus+     168>:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
```

调试命令

- vb <逻辑地址>：在相应的逻辑地址处添加了一个断点
- c：`调试命令`（continue）,开始执行模拟并进入调试模式。
- n: 单步调试（next）
- xp /<字节数> 地址：查看从xx地址开始 的xx个字节 的`物理内存`

列出断点：info break

删除断点：delete 断点编号（结合使用）

delete all——删除所有断点

```
break：用于设置断点。可以指定断点的地址，例如 break *0xaddress。
delete：用于删除断点。可以指定要删除的断点编号，例如 delete 1，或使用 delete all 删除所有断点。
info break：用于列出当前设置的断点信息，包括断点编号、地址和状态。

c（continue）：用于继续执行程序直到下一个断点或程序结束。
n（next）：        类似“逐过程调试”    用于单步执行程序，一次执行一行代码。
si（step into）：类似“逐语句调试”    用于进入函数或子例程的内部执行，如果当前指令是一个函数调用。
so（step over）：类似“跳出”          用于跳过函数调用，直接执行下一条指令。

info reg：用于查看寄存器的当前值和状态。
info mem：用于查看内存的内容。
x：用于显示内存中的数据。例如，x/10x 0xaddress 将显示指定地址处的前10个16进制值。
disasm：用于反汇编指令，可以查看指令的汇编代码。
info cpu：用于查看CPU的状态信息，如标志寄存器、指令计数器等。
quit：用于退出Bochs调试器。
```

调试步骤：

1. 使用vb设置相应的断点
2. 输入c开始执行调试（中间使用n、si、so等调试）
3. 记录此时console窗口输出的指令 （**该指令的字节码，以及下一跳指令的地址**）
4. 输入相关调试命令（sreg,r,xp）,进行验证