# DOS TOOL USER GUIDE

目录

[DOS TOOL USER GUIDE 1](#_Toc439184503)

[编译环境 2](#_Toc439184504)

[运行环境 3](#_Toc439184505)

[“Hello World” 5](#_Toc439184506)

[IO Space 5](#_Toc439184507)

[Memory 7](#_Toc439184508)

[CPU MSR 9](#_Toc439184509)

[CPUID 10](#_Toc439184510)

[SMBUS 11](#_Toc439184511)

[PCI Configuration Space 13](#_Toc439184512)

[Memory Map INT15 E820 14](#_Toc439184513)

# 编译环境

本项目使用Borland C++ 3.1编译器，“BC.rar”已上传至项目附件。

下载“BC.rar”后，解压并进入根目录，运行“Borland C++ 3.1.vbs”即可，编译器界面如图1所示，请用户选择“Option”菜单下选项自行配置相关设定。

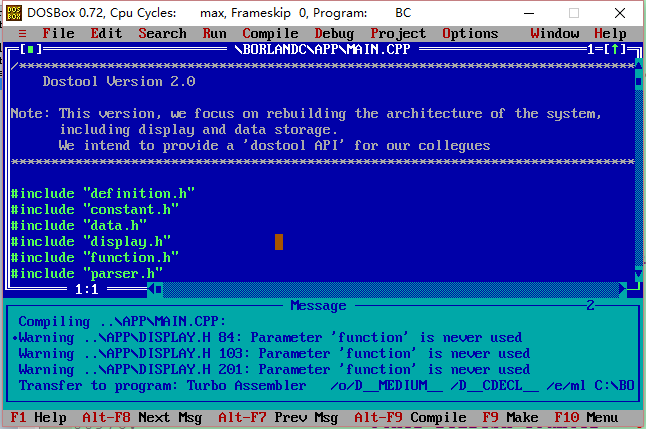


图 1

# 运行环境

软件于MS-DOS 7.1下运行和调试，“MSDOS7.10.ISO”镜像已上传至项目附件。下面将简要介绍运行环境搭建过程。

1. 启动UltraISO，图1为软件界面，在区域1中选择MS-DOS 7.1镜像存储路径；之后，双击区域2中的镜像文件，在区域3中将会看到镜像包含的文件。

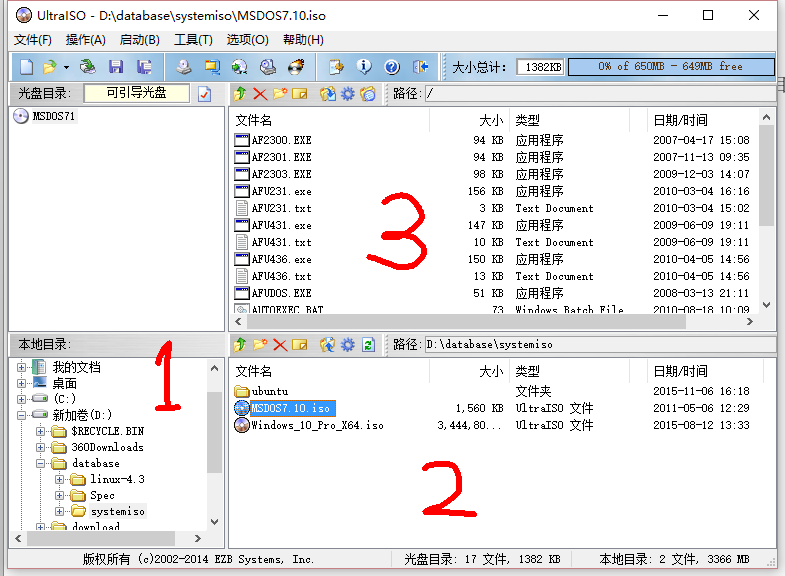


图 2

1. 点击“启动（B）”，选择“写入硬盘映像”，随后会弹出如图2所示窗口，在“硬盘驱动器”中选择需要写入镜像的硬盘，并勾选刻录校验，单击“写入”，等待映像写入完成即可。



图 3

# “Hello World”

## IO Space

I/O Space 的读写依靠汇编语言指令 IN & OUT 实现，而参数的数据宽度可以通过寄存器AL，AX，EAX来控制。IN 指令与 OUT 指令都需要通过AX寄存器来实现读写。

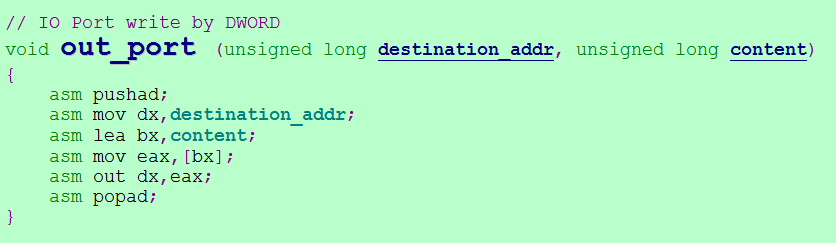


Figure 1．

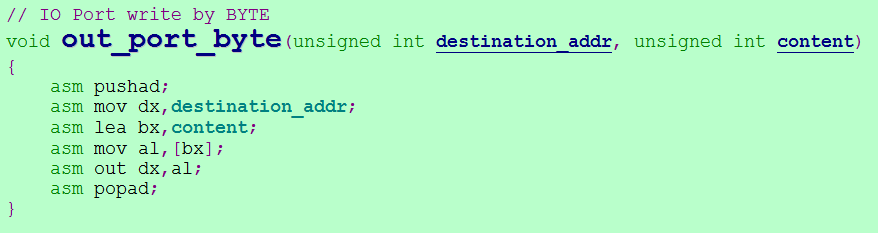


Figure 2



Figure 3

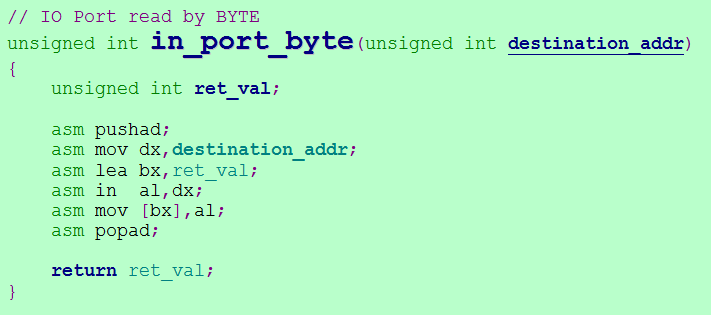


Figure 4

## Memory

Memory 的读写通过段地址：偏移地址的方式读取，在DOS环境下段地址为0，而内存地址为32位，因此偏移地址寄存器应使用EDI。同样，数据宽度同样可以通过寄存器AL，AX，EAX来控制。

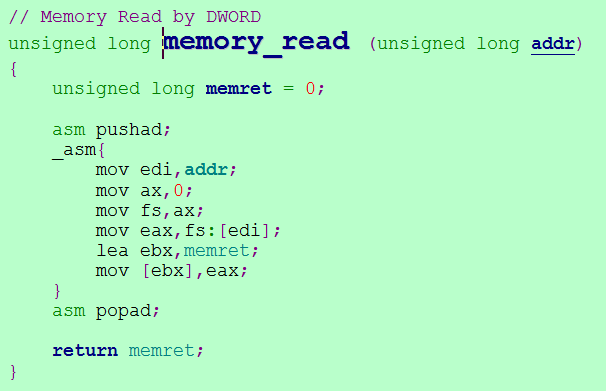


Figure 5．

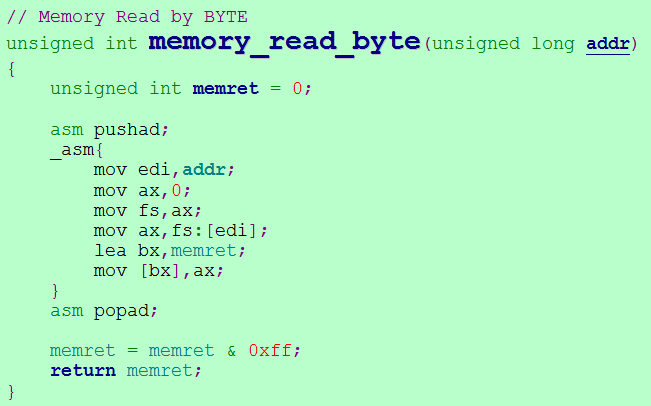


Figure 6．

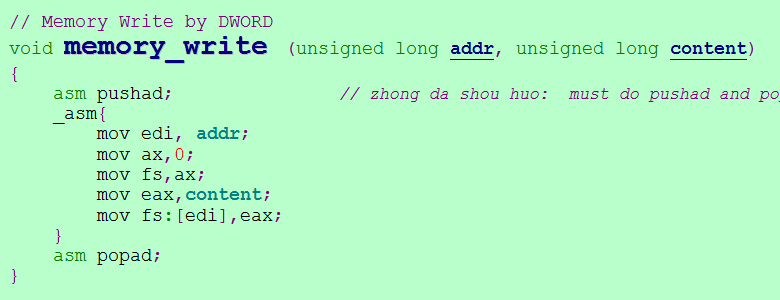


Figure 7．

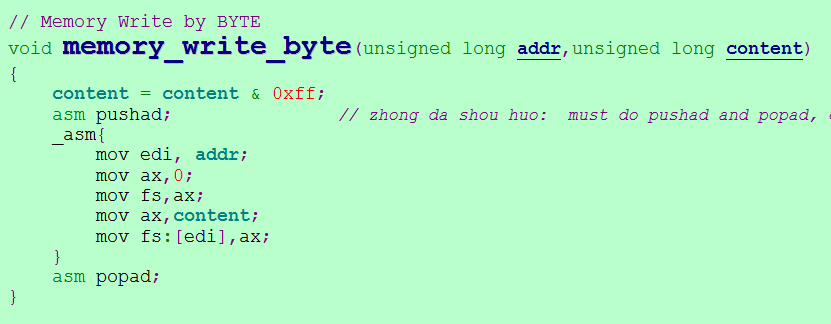


Figure 8．

## CPU MSR

CPU MSR读写通过RDMSR和WRMSR指令实现，由于这两条指令全部为32位指令，因此，在DOS环境下无法以指令形式执行，编译器无法识别。只有通过扩展指令集，并以直接写入机器码的形式实现。RDMSR和WRMSR指令通过ECX传入MSR寄存器地址。

RDMSR，读取ECX中存储的地址，并返回EDX:EAX的64位数据。

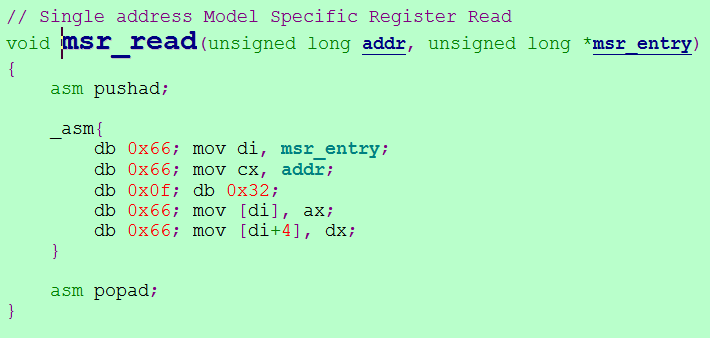


Figure 9.

WRMSR，将EDX:EAX中存储的数据写入ECX指向的地址。

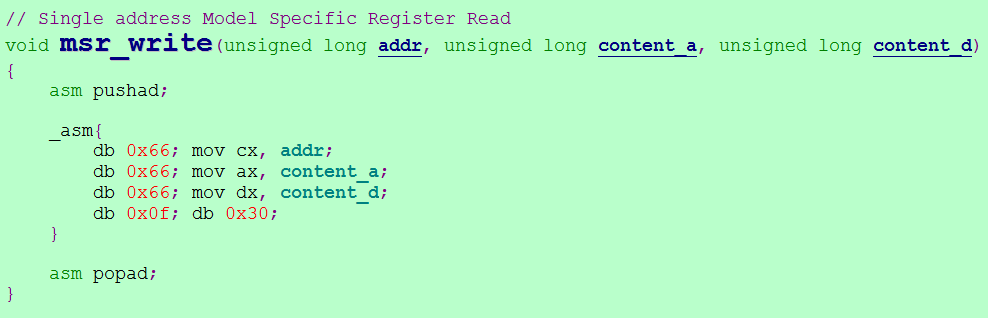


Figure 10.

## CPUID

CPUID 同样为32位指令，需要以机器码的形式使用。

CPUID 通过读取置于EAX中的功能号，将对应的4 \* 32位数据返回于EAX, EBX, ECX, EDX中。

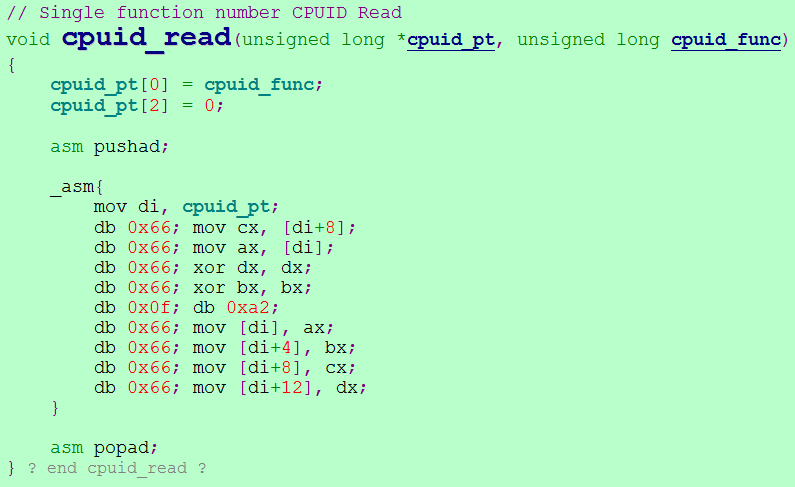


Figure 11.

## SMBUS

把大象放冰箱里需要三步，但是从SMBus读一个BYTE出来就需要很多步，反正我是信了。

1. 通过遍历PCI configuration space 并对比Class code，找到SMBus controller，从中读出其占用的IO地址空间，即SMBus base address；
2. 操作SMBus之前首先要判断SMBus是否可用，判断依据为SMBus base address是否为0x1E，不是的话，写成0x1E，然后死循环监测SMBus是否Ready；
3. 向SMBus base address + 4写入要读取数据的slave address，后死循环监测SMBus是否Ready；
4. 向SMBus base address + 3写入slave的地址偏移量，然后死循环监测SMBus是否Ready；
5. 向SMBus base address + 2写入0x48（以BYTE方式读取）；
6. 读取 SMBus base address 是否为0x2（是否完成操作），然后死循环监测SMBus是否Ready；
7. 读取SMBus base address + 5中数据。
8. 置于为什么是这样的，自己看SMBus SPEC, P-SPEC, IRS.

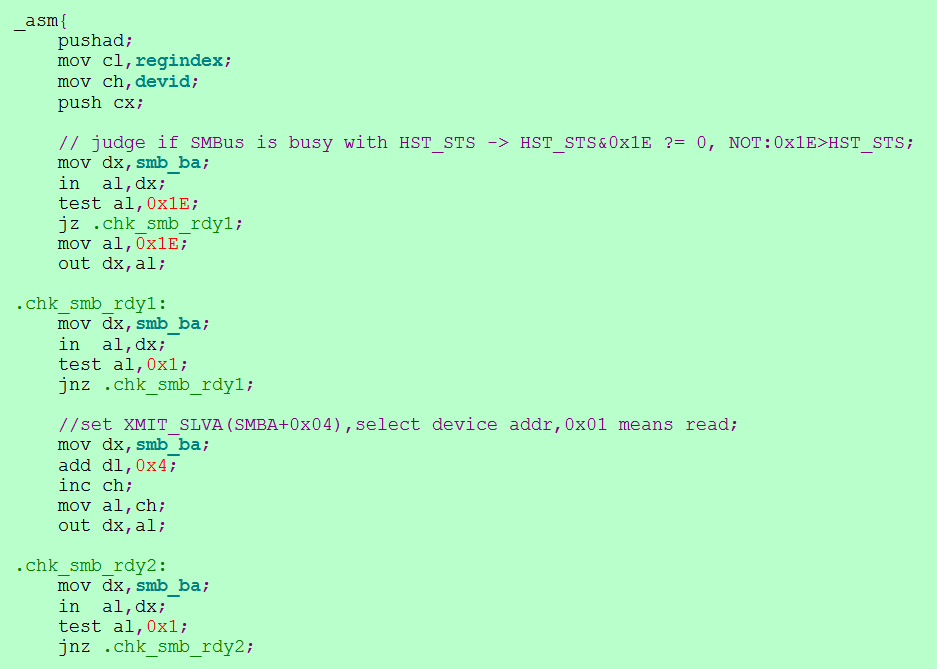


Figure 12.

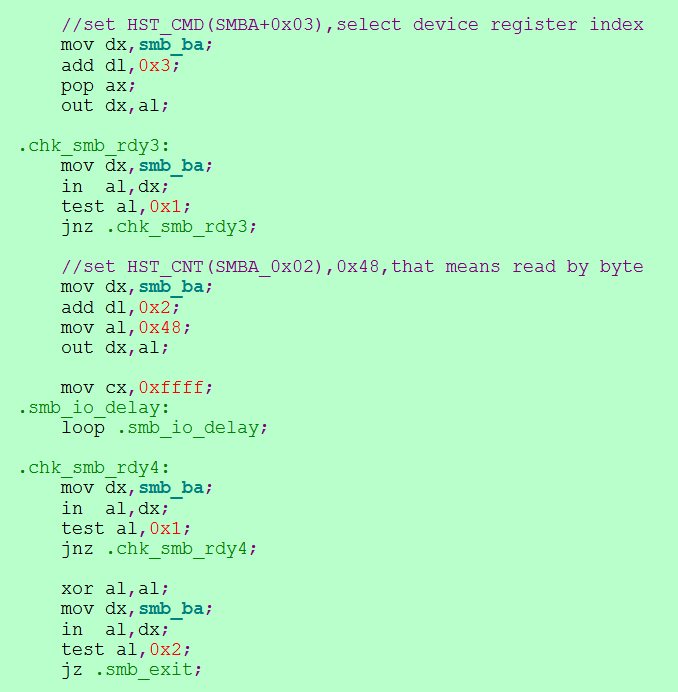


Figure 13.

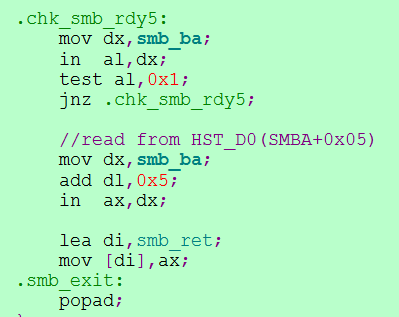


Figure 14.

## PCI Configuration Space

PCI Local Bus Specification 3.2.2.3.2节定义，PCI configuration space地址与数据必须以DWORD形式传输，CONFIG\_ADDRESS端口为0XCF8，CONFIG\_DATA端口为0XCFC。CONFIG\_ADDRESS结构如下图所示，

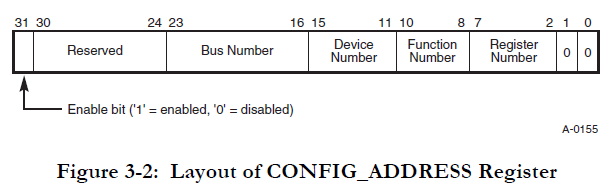


Figure ．

获取某OFFSET数据伪码可表示为：

MOV EAX, 0x80000000 + BUS<<16 + DEV<<11 + FUN<<8 + REG<<2;

MOV DX, 0XCF8;

OUT DX, EAX;

MOV DX, 0XCFC;

IN EAX, DX;

## PCI Express Extended Configuration Space

PCI configuration space以及PCIE Extended configuration space可以通过内存读取，详见PCI Express Base Specification 第7章以及PCI Firmware Specification 4.1节。

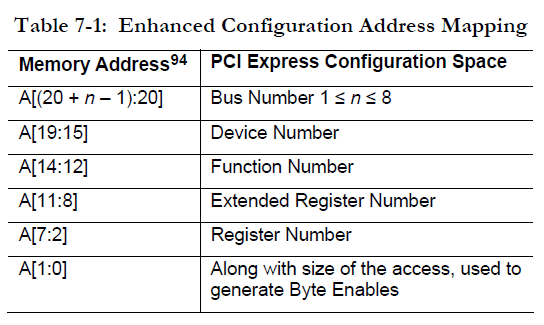


Figure ．

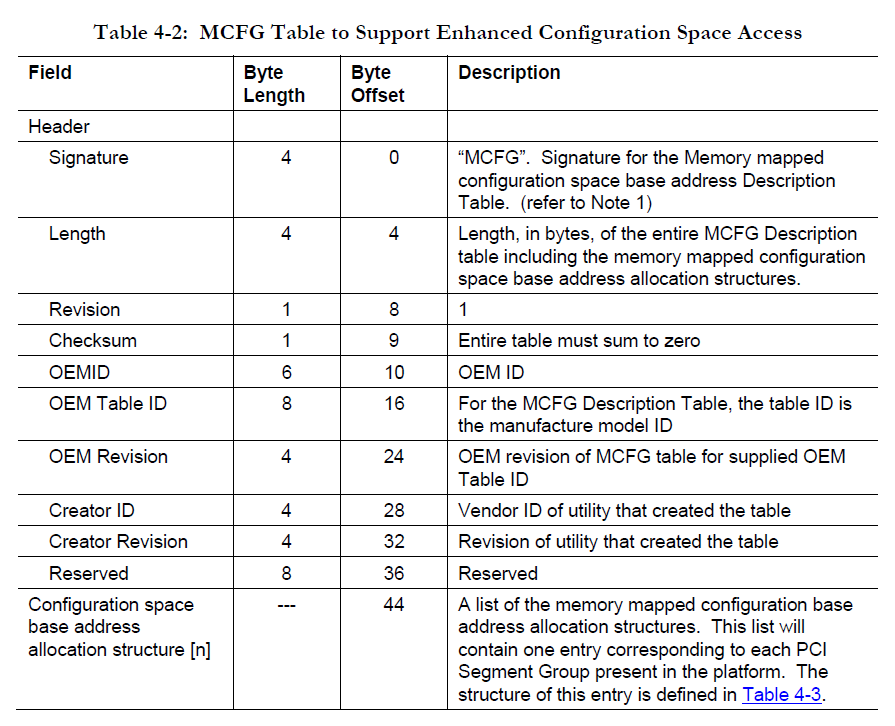


Figure ．

## Memory Map INT15 E820

请参考ACPI SPEC以及相关流程图，否则看不懂，真的。

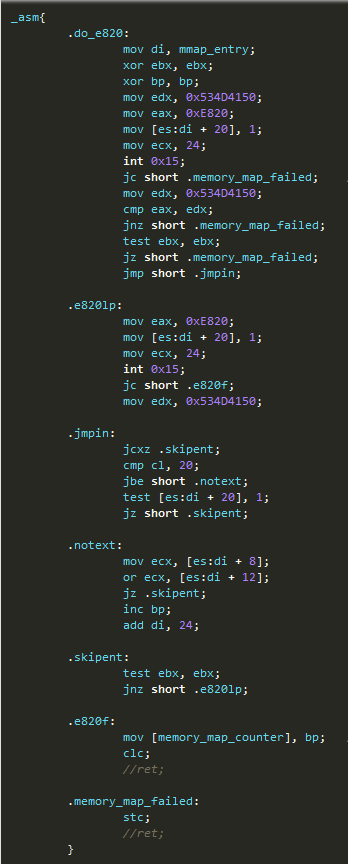


Figure 18.