读取 E820 的程序

原始程序来自网上，是 ASM+C 的。我使用 Tasm5+TC3.0 编译成功，但是无法运行。经过改造成为纯 C 的程序，使用 TC3 编译通过，并且在 VMWare 和实体机上测试成功（必须在纯 DOS 下）。

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

typedef unsigned char      BYTE;
typedef unsigned short     WORD;
typedef unsigned long      DWORD;
typedef unsigned char      BOOL;

char type_str[][25]=
{     "Undefined                ",
      "AddressRangeMemory",
      "AddressRangeReserved",
      "AddressRangeACPI",
      "AddressRangeNVS",
      "AddressRangeUnusable"
};

DWORD Int_E820(DWORD b,WORD seg,WORD offset,DWORD len)
{
   int hi,low;
    _asm{
      //If you want to debug with TD
      //you can enable the below code
      //db 0xcd,0x03            //Int 3
      db 0x60                   //pusha

                                //ebx data
      db 0x66,0x8b,0x5e,0x04 //mov ebx,[bp+4]
                                //buffer segment to ES
      db 0x8b,0x46,0x08         //mov ax,[bp+8]
      mov es,ax
                                //buffer offset to DI
      db 0x8b,0x7e,0x0a         //mov di,[bp+10]
```

```
                                //buffer size in ecx
        db 0x66,0x8b,0x4e,0x0c //mov ecx,[bp+12]

        db 0x66,0xb8,0x20,0xe8,0x00,0x00 //mov eax,e820
        db 0x66,0xba,0x50,0x41,0x4d,0x53 //mov edx,'SMAP'

        db 0xcd,0x15                //int15

        jnc int_ck
        db 0x66,0xbb,0xff,0xff,0xff,0xff //mov ebx,0xffffffff
        jmp int_ok
        }

    int_ck:
         _asm{
        db 0x66,0x53            //push ebx
        db 0x66,0xbb,0x50,0x41,0x4d,0x53
        db 0x66,0x3b,0xc3    //cmp eax,ebx
        db 0x66,0x5b            //pop ebx
        je int_ok
        db 0x66,0xbb,0xff,0xff,0xff,0xff //mov ebx,0xffffffff
        }

    int_ok:
         _asm{
        mov low,bx
        db 0x66,0xc1,0xeb,0x16 //shr ebx,16
        mov hi,bx

        db 0x61
      }
    //TC will use DX:AX as the return value of DWORD
    //And I have chenged to below form for the reason of easy to unstandard
    return ((hi << 16)+low);
}

void main(void)
{
        BYTE buf[20];
        DWORD bx=0,type;
        WORD seg,offset;
        int i;
        BYTE far *pt;
```

```
int cnt=0;

pt = (BYTE far *)buf;
seg = FP_SEG(pt);
offset = FP_OFF(pt);

puts(" NO       Base Address                    Length                                  Type");
puts("---+-------------------+-------------------+---------------------------------");


do
{
memset(buf,0,sizeof(buf));
bx = Int_E820(bx,seg,offset,sizeof(buf));

if(bx != 0xffffffff)
{
      printf("              %02d:              %02X%02X%02X%02X_%02X%02X%02X%02X
",cnt,buf[7],buf[6],buf[5],buf[4],buf[3],buf[2],buf[1],buf[0]);
      printf("                                 %02X%02X%02X%02X_%02X%02X%02X%02X
",buf[15],buf[14],buf[13],buf[12],buf[11],buf[10],buf[9],buf[8]);

      type = 0;//type
      for(i = 3; i >= 0; i--)
      {
      type <= 8;
      type |= buf[i+16];
      }
      if(type > 5)
      {
      printf("    Undefined\n");
      }
      else
      {
      printf("    %s\n",type_str[type]);
      }
      cnt++;
}
else
{
      printf("E820 Fail\n");
      break;                          o
}
}while(bx);
```

}

程序运行结果（VMWare 7.12）：

```
C:\TC\BIN>tc
 NO    Base Address           Length                    Type
---+-----------------+-----------------+--------------------------------
 00: 00000000_00000000   00000000_0009F800   AddressRangeMemory
 01: 00000000_0009F800   00000000_00000800   AddressRangeReserved
 02: 00000000_000DC000   00000000_00008000   AddressRangeReserved
 03: 00000000_000E8000   00000000_00018000   AddressRangeReserved
 04: 00000000_00100000   00000000_00DF0000   AddressRangeMemory
 05: 00000000_00EF0000   00000000_0000F000   AddressRangeACPI
 06: 00000000_00EFF000   00000000_00001000   AddressRangeNVS
 07: 00000000_00F00000   00000000_00100000   AddressRangeMemory
 08: 00000000_E0000000   00000000_10000000   AddressRangeReserved
 09: 00000000_FEC00000   00000000_00010000   AddressRangeReserved
 10: 00000000_FEE00000   00000000_00001000   AddressRangeReserved
 11: 00000000_FFFE0000   00000000_00020000   AddressRangeReserved
_
```

使用 iru 工具取得的结果

```
File  Config  Tools   System Quit                          Insyde Software Corp.

CPU: GenuineIntel (ID:06A7) Intel(R) Core(TM) i3-2330M CPU @ 2.20GHz

APM BIOS version: 01.02
Traditional memory: 640K
[SMBIOS] SMBIOS Version        = 2.4,       Number of DMI Structs = 62h
         SMBIOS Table Address  = 000F68F0h,  Length = 001Fh
         DMI Table Address     = 000E0010h,  Length = 0E73h
[E820] Start Address               End Address                 Type
       0000000000000000(0000KB)    000000000009F7FF(0638KB) 1=Available
       000000000009F800(0638KB)    000000000009FFFF(0640KB) 2=Reserved
       00000000000DC000(0880KB)    00000000000E3FFF(0912KB) 2=Reserved
       00000000000E8000(0928KB)    00000000000FFFFF(0001MB) 2=Reserved
       0000000000100000(0001MB)    00000000000EEFFFF(0014MB) 1=Available
       0000000000EF0000(0014MB)    00000000000EFEFFF(0014MB) 3=ACPI Data
       0000000000EFF000(0014MB)    00000000000EFFFFF(0015MB) 4=ACPI NVS
       0000000000F00000(0015MB)    00000000000FFFFFF(0016MB) 1=Available
       00000000E0000000(3584MB)    00000000EFFFFFFF(3840MB) 2=Reserved
       00000000FEC00000(4076MB)    00000000FEC0FFFF(4076MB) 2=Reserved
       00000000FEE00000(4078MB)    00000000FEE00FFF(4078MB) 2=Reserved
       00000000FFFE0000(4095MB)    00000000FFFFFFFF(4096MB) 2=Reserved

Type:PCI   Bus 00   Device 00   Function 00  <0x80000000>
DOS mode Ver 1.3.26                                          11:49:18
```

仔细观察会发现上述结果似乎存在不同，这是由于上面的给出来的是 Length 而后者给出来的是 Range 造成的。

2012-10-17

www.lab-z.com

Zoologist