<https://www.cnblogs.com/Pual623548198/p/6633263.html>

**一、起源**

　　吃鸡蛋的典故！

**二、为什么有大小端之分**

　　在计算机系统中，是以字节为单位的，每个地址单元都对应着一个字节，一个字节为8bit。但是在C语言中除了8bit的char之外，还有16bit的short型，32bit的long型（要看具体的编译器），另外，对于位数大于8位的处理器，例如16位或者32位的处理器，由于寄存器宽度大于一个字节，那么必然存在着一个如果将多个字节安排的问题。因此就导致了大端存储模式和小端存储模式。例如一个16bit的short型x，在内存中的地址为0x0010，x的值为0x1122，那么0x11为高字节，0x22为低字节。对于大端模式，就将0x11放在低地址中，即0x0010中，0x22放在高地址中，即0x0011中。小端模式，刚好相反。我们常用的X86结构是小端模式，而KEIL C51则为大端模式。很多的ARM，DSP都为小端模式。有些ARM处理器还可以由硬件来选择是大端模式还是小端模式。

**三、大小端的定义**

　　简单来说，大端模式就是存储器的高地址存放低字节；小端模式就是存储器的低地址存放低字节。

　　举例来说，假设有一个int a=0x12345678，大小端区别见下表：

|  |  |  |
| --- | --- | --- |
| 存储器地址 | 大端模式 | 小端模式 |
| 0x00 | 12 | 78 |
| 0x01 | 34 | 56 |
| 0x02 | 56 | 34 |
| 0x03 | 78 | 12 |

**四、代码实现大下端判断**

[复制代码](javascript:void(0);)

1 void BigOrLittleEndianJudge(void)

2 {

3 unsigned short a = 0x1234;

4 #if 0

5 //（1）取a的地址

6 //（2）取a的地址的值，将其转化成unsigned char类型的数

7 //（3）将该数取出来

8 printf("%X\n",(\*(unsigned char \*)&a));

9 #endif

10 if((\*(unsigned char \*)&a) == 0x12) //如果

11 {

12 　　printf("大端模式");

13 }

14 else

15 {

16 　　printf("小端模式");

17 }

18 }

[复制代码](javascript:void(0);)

**参考文献**

http://blog.csdn.net/zxh2075/article/details/53927425