汇编相关的网站：

http://bbs.pediy.com 看雪学院(国内讨论破解)

http://www.52pojie.cn 讨论破解

http://forum.exetools.com 讨论汇编、破解的网站

http://www.tuts4you.com 国外讨论破解

http://www.woodmann.com/crackz 国外老的破解教程

http://www.masm32.com 国外 32 位汇编网站

https://zjusec.com 浙大信息安全小组 AAA

linux 环境下的汇编语言例子：

http://10.71.45.100/bhh/hello_linux.zip

db, dw, dd, dq, dt:

unsigned char == 汇编的 byte(字节) 8 位

unsigned short int== 汇编的 word(字) 16 位

unsigned long int==汇编的 double word(双字)32 位

这三种类型定义的关键词分别为:db dw dd，例如:

```
a db 12h; unsigned char a = 0x12;
b dw 1234h; unsigned short int b=0x1234;
c dd 12345678h; unsigned long int c=0x12345678;
```

汇编语言在定义时并不区分有符号还是非符号数，例如:

a db 0FFh; 到底代表 255 还是-1,在定义时并不确定

在引用变量 a 时可以用指令来区分它是非符号还是有符号，例如 imul a 指令表示乘以-1，而 mul a 指令则表示乘以 255。

dd 也可以用来定义一个 32 位的小数即 float 类型的小数,例如:

```
pi dd 3.14; float pi=3.14;
```

dq 定义 64 位整数 quadruple word 或 double 类型小数

例如:

```
x dq 123456787654321h; __int64 x=...;
y dq 3.14; double y = 3.14;
```

还有一个 dt 可以用来定义一个 80 位的小数即 long double 类型的小数:

```
z dt 3.14; 10 字节的小数，相当于 C 语言的 long double
         ; printf("%Lf", z);
```

小端规则：

先存放低 8 位，后存放高 8 位的规则称为：

```
Little-Endian (小端规则)
short int a = 0x1234;
```

或用汇编语法写成：`a dw 1234h`

设 a 的地址为 1000，则 a 的值在内存中的布局如下所示：

地址	值
1000	0x34；低 8 位在前
1001	0x12；高 8 位在后

`0x1234 = 0001 0010 0011 0100`

以下 2 个代码可以验证小端规则：

```
main()
{
   unsigned short int a = 0x1234;
   unsigned char *p;
   p = (unsigned char *)&a;
   printf("%X %X", p[0], p[1]);
}

main()
{
   unsigned char a[2]={0x12, 0x34};
   unsigned short int *p;
   p = (unsigned short int *)a;
   printf("%X", *p);
}

    long int a = 0x12345678;
```

或用汇编语法写成：a dd 12345678h

设 a 的地址为 1000，则 a 的值在内存中的布局如下所示：

地址　　值

1000　　0x78；低 8 位在前

1001　　0x56

1002　　0x34

1003　　0x12；高 8 位在后

8 位、16 位、32 位整数的取值范围：

C 语言中用来表示 8 位非符号数的类型是：unsigned char，范围[00h,0FFh]即[0,255]

16 位非符号数类型是:unsigned short int，范围是[0000h, 0FFFFh]即[0,65535]

32 位非符号数类型是:unsigned long int，范围是[00000000h, 0FFFFFFFFh]即[0,2^{32}-1]

8 位符号数的范围是[-128, +127]即[80h,7Fh]

16 位符号数的范围是[-32768, +32767]即[8000h,7FFFh]

32 位符号数的范围是[-2147483648, +2147483647]即[80000000h,7FFFFFFFh]

零扩充和符号扩充：

当把一个宽度较小的值赋给宽度较大的变量时，会发生扩充。

扩充包括零扩充及符号扩充两种。

char a = -1;　二进制=1111 1111B

short int b;

b = a;　b 的二进制=1111 1111 1111 1111

char a = 127; 二进制=0111 1111B

short int b;

b = a;　b 的二进制=0000 0000 0111 1111

```
unsigned char a = 0x80; 二进制=1000 0000B
short int b;
b = a;   b 的二进制=0000 0000 1000 0000

char a = -1; 二进制=1111 1111B
unsigned short int b;
b = a;   b 的二进制=1111 1111 1111 1111B
```

IEEE745 标准中单精度小数(即 float 类型)的表示：

42	FE	C0	00
01000010	11111110	11000000	00000000

0	1000010	1	1111110	11000000	00000000

偏置指数(8 位非符号数)　　尾数(23 位)

=133

-127 常数

=6 实际的指数　求得的6代表2^6

偏置指数(8 位)

符号位(1 位)

最高位为0表示正数

为1表示负数

```
42 fe c0 00
1.1111110 11000000 00000000
1111111.0 11000000 00000000
127.375
```

用以下程序可以输出 127.375 在内存中的 4 个字节：

```
main()
{
   float f=127.375;
   unsigned char *p;
   int i;
   p = (unsigned char *)&f;
   for(i=0; i<4; i++)
```

```
        printf("%02X",  p[i]);
    }
```

当32位为全0时，不按照前面的规则进行计算，而将此float类型处理为0