看雪学院-解密入门教学（二）笔记

原创

Cytosine 于 2017-02-03 15:46:15 发布 1362 收藏 1
分类专栏： 笔记_解密入门教学 汇编 寄存器 解密 逆向 文章标签： Note 寄存器 解密 汇编 逆向

本文链接：https://blog.csdn.net/Cytosine/article/details/54847400


笔记_解密入门教学 同时被 3 个专栏收录
2 篇文章 0 订阅
订阅专栏


汇编
1 篇文章 0 订阅
订阅专栏


寄存器
1 篇文章 0 订阅
订阅专栏

解密入门教学（二）— 看雪学院

原作地址：http://mp.weixin.qq.com/s?__biz=MjM5NTc2MDYxMw==&mid=2458281884&idx=2&sn=786b29f538d74709508f4e7d4cdfcd6e&chksm=b1

位

一个二进制位 ~ 一位

八位 ~ 一个字节

在内存中，以字节为单位存储信息。每一个字节单元给以一个唯一的存储器地址（即物理地址）

八个二进制位 ~ 八位 ~ 一个字节 即可表示所有ASCII码~ 一个英文字符或数字

十六个二进制位 ~ 十六位~ 两个字节~ 两个内存单元~ 一个汉字

三十二位 ~ 双字

六十四位 ~ 四字

需要掌握的十六个寄存器

通用寄存器

八个，分别是EAX EBX ECX EDX ESP EBP EDI ESI

EAX EBX ECX EDX数据寄存器

主要用于暂时存放计算过程中所用的操作数、结果或其他信息。

除直接访问外，还可分别对其高十六位和低十六位进行访问。低十六位就是把它们前面的E去掉，即EAX的低十六位为AX。

它们的低十六位又可以分别进行八位访问。即AX还可分为高八位AH和低八位AL。

所以，想操作的是一个八位数据：MOV AL或MOV AH

一个十六位数据：MOV AX

一个三十二位数据：MOV EAX

ESP EBP EDI ESI变址寄存器

主要用于在存储器寻址时，提供偏移地址。只能用字来访问。

没看懂的一段：从386以后，所有的寄存器都可以用来存储内存地址。（这里给你讲一个小知识，你在破解的时候是不是看到过[EBX]这样的形式呢？这就是说此时EBX中装的是一个内存地址，而真正要访问的，就是那那个内存单元中所存储的值）。

ESP堆栈指针寄存

堆栈是以“后进先出”方式工作的一个存储区，它必须存在于堆栈段中，因而其段地址存放于SS寄存器中。它只有一个出入口，所以只有一个堆栈指针寄存器。

ESP的内容在任何时候都指向当前的栈顶。

堆栈的基址开始于一个高地址，然后每当有数据入栈，它就向低地址的方向进行存储。相应的入栈指令是PUSH。

每当有一个数据入栈，ESP就跟着改变。总之，它永远指向最后一个压入栈的数据。

如果要用压入栈的数据，就用相应的出栈指令POP，POP指令执行后，ESP会加上相应的数据位数。

原文摘录：特别是现在到了Win32系统下面，堆栈的作用更是不可忽视，API所用的数据，均是靠堆栈来传送的，即先将要传送的数据压入堆栈，然后CALL至API函数，API函数会在函数体内用出栈指令将相应的数据出栈，然后进行操作。许多明码比较的软件，一般都是在关键CALL前，将真假两个注册码压入栈。然后在CALL内出栈后进行比较。所以，只要找到这个关键CALL，就能在压栈指令处，下d命令来查看真正的注册码。

EBP基址指针寄存器

ESP和EBP都可以与堆栈段寄存器SS联用来确定堆栈中的某一存储单元的地址。

ESP用来指示段顶的偏移地址，而EBP可作为堆栈区中的一个基地址以便访问堆栈中的信息。

ESI（源变址寄存器）和**EDI**（目的变址寄存器）一般与数据段寄存器DS联用，用来确定数据段中某一存储单元的地址。这两个变址寄存器有自动增量和自动减量的功能，可以很方便地用于变址。

在串处理指令中，ESI和EDI作为隐含的源变址和目的变址寄存器时，ESI和DS联用，EDI和附加段ES联用，分别达到在数据段和附加段中寻址的目的。

专用寄存器

两个，EIP FLAGS

EIP指令指针寄存器 最重要哦！

用于存放代码段中的偏移地址。在程序运行的过程中，它始终指向下一条指令的首地址。它与段寄存器CS联用确定下一条指令的物理地址。

当这一地址送到存储器后，控制器可以取得下一条要执行的指令，而控制器一旦取得这条指令就马上修改EIP的内容，使它始终指向那些跳转指令。

FLAGS标志寄存器

又称PSW（program status word）程序状态寄存器

用于存放条件标识码、控制标识和系统标识的寄存器。（标识：zhi四声）

指令举例：

Cmp EAX,EBX ; 用EAX EBX相减

JNZ 00470395 ; 不相等的话，就跳到这里

这两条指令就是用EAX装的数减去EBX装的数，来比较这两个数是否相等。当Cmp指令执行后，就会在FLAGS的ZF（zero flag）零标志位上置相应值，如果结果为0，也就是他们两个相等的话，ZF置1，否则置0。

其它还有OF（溢出标志）SF（符号标志）CF（进位标志）AF（辅助进位标志）PF（奇偶标志）

段寄存器

六个，CS代码段 DS数据段 ES附加段 SS堆栈段 FS附加段 GS附加段

常用汇编指令

CMP A,B 比较A与B。其中A B可以是寄存器或内存地址，也可同时是两个寄存器，但不能同时都是内存地址。（许多明码比较的软件就用这个指令）

MOV A,B 把B的值送给A。其中A B可以是寄存器或内存地址，也可以同时是两个寄存器，但不能同时都是内存地址。

Xor a,a 异或操作，主要是用来将a清空

LEA 装入地址。例：LEA DX,string 将字符的地址装入DX寄存器。

PUSH 压栈

POP 出栈

ADD 加法指令。格式ADD DST,SRC

执行的操作：(DST)<-(SRC)+(DST)

SUB 减法指令。格式SUB DST,SRC

执行的操作：(DST)<-(DST)-(SRC)

MUL 无符号乘法指令。格式MUL SRC

执行的操作：字节操作(AX)<-(AL)*(SRC)

字操作 (DX,AX)<-(AX)*(SRC)

双字操作(EDX,EAX)<-(EAX)*(SRC)

DIV 无符号除法指令。格式DIV SRC

执行的操作：字节操作(AL)<-(AX)/(SRC)的商

(AH)<-(AX)/(SRC)的余数[1]

字操作 (AX)<-(DX,AX)/(SRC)的商

(DX)<-(DX,AX)/(SRC)的余数[2]

双字操作(EAX)<-(EDX,EAX)/(SRC)的商

(EDX)<-(EDX,EAX)/(SRC)的余数[3]

[1]16们被除数在AX中，8位除数为源操作数，结果的8位商在AL中，8位余数在AH中。

[2]32位被除数在DX,AX中。其中DX为高位字，16位除数为源操作数，结果的16位商在AX中，16位余数在DX中。

[3]64位的被除数在EDX,EAX中。其中EDX为高位双字；32位除数为源操作数，结果的32位商在EAX中，32位余数在EDX中。

NOP 无作用，可以用来抹去相应语句

CALL 调用子程序

控制转移指令：

JE JZ 若相等则跳

JNE JNZ 若不相等则跳

JMP 无条件跳

JB JL 若小于则跳

JA JG 若大于则跳

JGE 若大于等于则跳

JLE 若小于等于则跳

进制转换（计算机基础课程中学过，在此不再重复）

Q&A

Q： 寄存器可以随便用么，有没有什么限制？写个程序的时候那些变量什么的可以放在任意的寄存器么？

A： 寄存器有它的使用机制，及各个寄存器都有着明确的分工。

数据寄存器（EAX-EDX）它们都是通用寄存器，即在软件中，任何数据都可存放于此。但除此之外，它们又可以用于各自的专用目的。

EAX可作为累加器来使用，所以EAX是算数运算的主要寄存器。在乘除法等指令中指定用来存放操作数。比如在乘法中用AL或AX或EAX来装被乘数，而AX或DX:AX或EAX或EDX:EAX则用来装最后的积。

EBX一般在计算存储器地址时用作基址寄存器。

ECX常用来保存计数值，如在移位指令它用来装位移量、循环和串处理指令中作隐含的计数器。

EDX一般在作双字长运算时把DX和AX组在一起存放一个双字长数。

例：二进制数01101000110101000100100111010001，要把它寄存起来，就可以把0110100011010100(即高十六位)放在DX中，把0100100111010001(即低十六位)放在AX中，这个数表示为DX:AX。当然完全可以用一个EDX就把这个数给装下。此外，还可以用EDX:EAX来装一个64位数据。

推荐的汇编参考书：《80x86汇编语言程序设计》沈美明主编