

汇编实验二

1. 认识OllyDbg/immunityDebugger

00401004 CC INT3

00401005 \$ E9 06000000 JMP 425.00401010

0040100A CC INT3

0040100B CC INT3

0040100C CC INT3

0040100D CC INT3

0040100E CC INT3

0040100F CC INT3

00401010 > 33C0 XOR EAX,EAX

00401012 . 33D2 XOR EDX,EDX

00401014 . 33DB XOR EBX,EBX

00401016 . 66:B8 6400 MOV AX,64

0040101A . B3 07 MOV BL,7

0040101C . F6F3 DIV BL

0040101E . 6A 00 PUSH 0

00401020 . E8 05000000 CALL <JMP.&KERNEL32.ExitProcess>

00401025 CC INT3

代码区

00401010=425.00401010

Address	Hex dump
00404000	28 40 00 00 00 00 00 00 00 00 00 00 96 40 00 00
00404010	58 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00404020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00404030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00404040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00404050	00 00 00 00 00 00 00 00 00 00 CA 3A 23 77 00 00 00 00

数据段

Registers (FPU)

EAX 0000020E

ECX 00000000

EDX 00000000

EBX 00000007

ESP 0018FF88

EBP 0018FF90

ESI 00000000

EDI 00000000

EIP 0040101E 425.0040101E

C 0 ES 002B 32bit 0 (FFFFFFFF)

P 1 CS 0023 32bit 0 (FFFFFFFF)

A 0 SS 002B 32bit 0 (FFFFFFFF)

Z 1 DS 002B 32bit 0 (FFFFFFFF)

S 0 FS 0053 32bit 7FFDD000 (FFFF)

T 0 GS 002B 32bit 0 (FFFFFFFF)

D 0

0018FF88 772386E3 銑#w RETURN t

0018FF8C 7FFDE000 . 幟

0018FF90 0018FFD4 ?□.

0018FF94 7767BD99 銑cw RETURN t

0018FF98 7FFD

0018FF9C 32BB

0018FFA0 00000000

CPU 寄存器

堆栈段

代码区

线性地址

指令机器码

指令汇编代码

00401004	CC	INT3
00401005	\$ E9 06000000	JMP 425.00401010
0040100A	CC	INT3
0040100B	CC	INT3
0040100C	CC	INT3
0040100D	CC	INT3
0040100E	CC	INT3
0040100F	CC	INT3
00401010	> 33C0	XOR EAX,EAX
00401012	. 33D2	XOR EDX,EDX
00401014	. 33DB	XOR EBX,EBX
00401016	. 66:B8 6400	MOV AX,64
0040101A	. B3 07	MOV BL,7
0040101C	. F6F3	DIV BL
0040101E	. 6A 00	PUSH 0
00401020	. E8 05000000	CALL <JMP.&KERNEL32.ExitProcess>
00401025	CC	INT3

数据段

线性地址

数据段中数据

Address	Hex dump
00404000	28 40 00 00 00 00 00 00 00 00 00 00 00 96 40 00 00
00404010	58 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00404020	00 00 00 00 00 00 00 00 88 40 00 00 00 00 00 00 00
00404030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00404040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00404050	00 00 00 00 00 00 00 00 CA 3A 23 77 00 00 00 00 00

CPU寄存器

通用寄存器

Registers (FPU)

EAX 0000020E
ECX 00000000
EDX 00000000
EBX 00000007
ESP 0018FF88
EBP 0018FF90
ESI 00000000
EDI 00000000

指令指针寄存器

EIP 0040101E 425.0040101E

标志寄存器

C 0 ES 002B 32bit 0 (FFFFFFFF)
P 1 CS 0023 32bit 0 (FFFFFFFF)
A 0 SS 002B 32bit 0 (FFFFFFFF)
Z 1 DS 002B 32bit 0 (FFFFFFFF)
S 0 FS 0053 32bit 7FFDD000 (FFF)
T 0 GS 002B 32bit 0 (FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g

堆栈段

堆栈寄存器

栈顶

堆栈段

Registers (FPU)

EAX 0000020E

ECX 00000000

EDX 00000000

EBX 00000007

ESP 0018FF88

EBP 0018FF90

0018FF88 772386E3

0018FF8C 7FFDE000

0018FF90 0018FFD4

0018FF94 7767BD99

0018FF98 7FFDE000

0018FF9C 32BBCE4C

0018FFA0 00000000

0018FFA4 00000000

0018FFA8 7FFDE000

0018FFAC C9D49CFF

0018FFB0 FFFFFFF800

0018FFB4 00000000

0018FFB8 00000000

0018FFBC 0018FF9C

0018FFC0 FFFFFFFA80

0018FFC4 0018FFE4

0018FFC8 77625191

0018FFCC 45C5D410

实验二

【课堂练习】

- 试编程实现找出k个完美数，正整数n成为完美数是指n等于其所有真因子的和。

如 $6=1+2+3$, $28=1+2+4+7+14$

- 试编程实现正整数的素数分解。

$$72=2^3*3^2$$

作业

- 1.有3个整数数组中，对每个数组统计被3除余数分别为0、1与2的整数个数,并在屏幕显示统计结果。
- 2.给定内存中整数数组，及给定一个参考整数Key，试调整数组使得小于Key的数存放在数组左边，大于或等于Key数放在数组右边，试编写程序完成上述任务并输出分界位置及数组。
- 3.从键盘读取一串字符S,该字符串包含一个简单算术表达式（含两个正整数（如：3434*45）的四则运算），试编程解析该字符串，并实现其语义。

4.正整数的2的幂次方表示：任何一个正整数都可以表示成2的幂次方和。如 $15=2^3+2^2+2+2^0$

1.算法 printPerfNumbers(k):

```
count=0;
```

```
n=6;
```

```
while (count < k) {
```

```
    if isPerfNumber(n) {
```

```
        print(n);
```

```
        count++;
```

```
    }
```

```
    n++;
```

```
}
```

算法 isPerfNumber(n):

```
sum=1;
```

```
factor=2;
```

```
while (factor <= n/2) {
```

```
    if n% factor ==0 {
```

```
        sum = sum + factor;
```

```
    }
```

```
    factor=factor+1;
```

```
}
```

```
if sum==n { return 1;}
```

```
return 0;
```

2. 算法 factorNumber(n):

p=2;

while (p <= n/2) {

 If isPrime(p) {

 求最大整数k 满足: $n \%(p^k) == 0$;

 if k>0 then 保存(p,k);

 }

 p=p+1;

}

2. 算法 factorNumber(n):

 p=2;

again:

 If $p \leq n/2$ then

 If isPrime(p) then {

 ;find greatest k such that $n \%(p^k) == 0$;

 ;if $k > 0$ then save(p,k)

 if maxExp (n, p) > 0 then save(p,k)

 }

 p=p+1;

 goto again

final:

算法 isPrime(n):

if $n == 1$ then return 0;

for ($i=2$; $i \leq n/2$; $i++$)

 if $n \% i == 0$ then return 0;

}

return 1;

算法 maxExp (n, p):

k=0;

while (n%p==0) {

 k=k+1

 n=n/p;

}