

汇编实验一

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

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

代码区

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 (FFFFF)

T 0 GS 002B 32bit 0 (FFFFFFFF)

D 0

CPU
寄存器

0018FF88 772386E3 銑#w RETURN t

0018FF8C 7FFDE000 . 幟

0018FF90 0018FFD4 ?□.

0018FF94 7767BD99 銑cw RETURN t

0018FF98 7FFD

0018FF9C 32BB

0018FFA0 00000000

堆栈段

数据段

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

2、容易混淆指令

除法指令

➤ 除数是8位寄存器：

被除数是16位寄存器AX

指令执行完

商存放在AL

余数存放在AH

容易混淆指令

1. 除法指令

➤ 除数是16位寄存器：

被除数是16位寄存器DX与16位寄存器AX组成：

$$(DX) * 2^{16} + (AX)$$

指令执行完

商存放在AX

余数存放在DX

容易混淆指令

除法指令

➤ 除数是32位寄存器:

被除数是32位寄存器EDX与32位寄存器AX组成:

$$(EDX) * 2^{32} + (EAX)$$

指令执行完

商存放在EAX

余数存放在EDX

容易混淆指令

TEST指令

➤ Test op1, op2

执行操作 op1 and op2, 并根据执行的结果是否为0,
设置cpu中标志寄存器ZF位(即Z位)

3. 汇编程序控制结构

1. 分支结构

2. 循环结构

3. 子程序、函数调用与返回

1.试用汇编语言实现一函数，该函数输入是32位无符号整数与基数（2,8,10,16），输出是对应的2、8、10与16进制输出。

2.试用汇编语言编写程序，该程序具有如下菜单：

1) 2进制

2) 8进制

3) 10进制

4) 16进制

请输入选项：

待输入选项后显示如下提示

请输入无符号数：

待输入整数后，输出该整数对应的选项的进制表示

实验一

- 内存中有**1**个**32**位整数数组，编程求整数数组最大值的元素下标，并在屏幕显示该下标的值。
- 内存中有**3**个大小不同的**32**位整数数组，编程分别求这**3**个整数数组最大值的元素下标，并在屏幕显示这些下标的值。
- 内存中有**1**个**32**位整数数组，编程用选择排序方法对该整数数组排序，并在屏幕显示该数组排序前后的值。

- 试编写汇编程序求无符号整数逆序，比如输入254679，程序输出将是976452
- 试求两个整数最大公约数

辗转相除法：

求 $x=18$ 与 $y=14$ 最大公约数

$$r0 = x \% y = 18 \% 14 \quad (=4)$$

$$r1 = 14 \% r0 = 14 \% 4 \quad (=2)$$

$$r2 = 4 \% r1 = 4 \% 2 \quad (=0)$$

Algorithm E (*Euclid's algorithm*). Given two positive integers m and n , find their *greatest common divisor*, that is, the largest positive integer that evenly divides both m and n .

E1. [Find remainder.] Divide m by n and let r be the remainder. (We will have $0 \leq r < n$.)

E2. [Is it zero?] If $r = 0$, the algorithm terminates; n is the answer.

E3. [Reduce.] Set $m \leftarrow n$, $n \leftarrow r$, and go back to step E1.

again:

$r = x \% y;$

if $r == 0$ then{

y 是最大公约数

goto **final**

} else {

$x = y;$

$y = r;$

goto **again**

}

final:

$x=18, y=14$

$r0 = x \% y = 18 \% 14 \quad (=4)$

$r0 \neq 0$

$x=14, y=4$

$r1 = x \% y = 14 \% 4 \quad (=2)$

$r1 \neq 0$

$x=4, y=2$

$r2 = x \% y = 4 \% 2 \quad (=0)$

$r2=0, y$ 是最大公约数