MOV

x mov eip,0xaa

LEA EAX,DWORD PTR DS:[0x12345678 / ESP+8]

logic:

ADD

SUB

AND

OR

XOR

NOT

ADC 带进位加法

SBB 带借位减法

XCHG R/M,R/M

push 0x1234567 / r / m

esp-(内容->esp,当前地址指针--)

ebp不变

pop eax / m

esp+

ebp不变

pushad

1.eax,ecs,edx,ebx,esp,ebp,esi,edi依次存入->esp所指地址

2.esp减去相应地址代表栈顶

popad

pushad的逆向操作

MOVS(DF)

ESI -> EDI

STOS(DF)

EAX -> EDI

REP MOVSD | STOSD

repeat ECX

Eflags

CPAZSTIDO

Carry,Parity,Auxiliary,Zero,Signal,T?(单步标志),Interrupt,Direction,Overflow

JMP 0x4183fd / eax

(if dest < 128 byte,) -> JMP SHORT 0x4183fd

modify eip()

CALL 0x4183fa

modify eip&esp(next ins +curlength)

ret

modify eip&esp

equal pop eip， ESP->EIP

CMP R/M,R/M/IMM

like sub,两值相等->zf=1,且只改变zf位,不改R/M

s位代表符号，表示比较大小

TEST R,R

like CMP，但是是与操作

JE=JZ

jump equal(jump zero) 当zf为1时跳转

JNE=JNZ

jump not equal(jump not zero) 当zf不为1时跳转

JS 结果为负跳转

jump

JNS 结果非负跳转

JCC=JUMP

但是JCC是不同情况的jump，指的是不同情况下的jump

JP,JPE

PF为1时跳转，即有偶数个1位时

JNP,JPO

PF不为1时跳转，即有奇数个1位时

JNL,JGE

>=jump(有符号数)

JLE,JNG

<=jump(有符号数)

JNLE,JG

>jump(有符号数)

JL,JNGE

<jump(有符号数)

JO

OF为1jump

JNO

OF不为1跳转

JB，JNAE

<jump(无符号数)

JNB，JAE

>=jump(无符号数)

JBE，JNA

<=jump(无符号数)

JNBE，JA

>jump(无符号数)

CCCCCCCC int3填满内存 防止缓冲区溢出 R版没有

堆栈平衡

调用函数前后栈顶指针-栈地和调用前一样

外平栈：



CALL调用后平衡，谁调用谁平衡

OD:

break point:F2

bp MessageBoxA: 在MessageBoxA处打断点汇编4