



南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



汇编语言与逆向技术

第5章 过程

王志

zwang@nankai.edu.cn

updated on 2022.10.13

南开大学 网络空间安全学院
2022-2023学年



允公允能 日新月异

本章知识点

1. 程序动态调试
2. 控制流转移指令
3. 过程的定义和使用
4. 调用链接库中的函数





南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



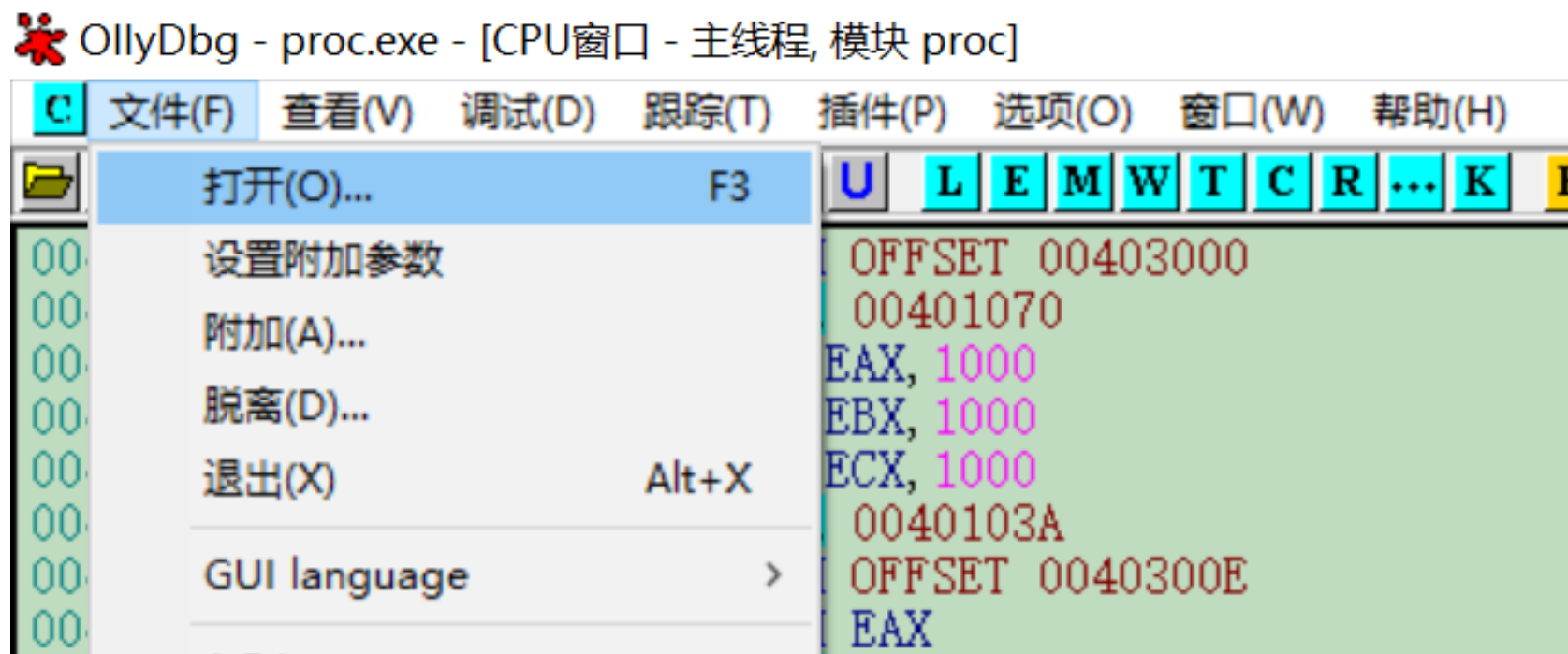
1. 程序动态调试



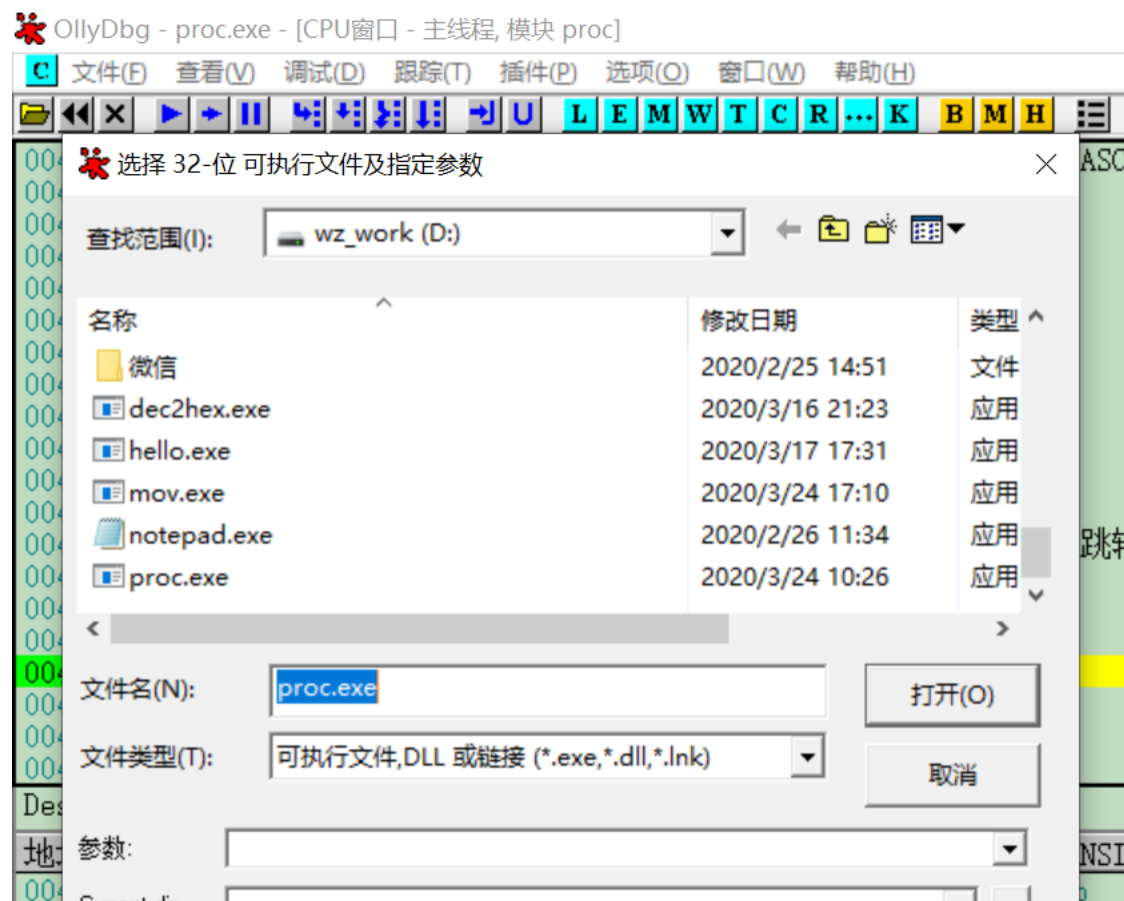
- 32位Windows调试器（Debugger）
 - 反汇编，将CPU指令翻译为汇编语句
 - 动态追踪汇编语句的执行
 - Ring3级
 - 官网地址， <http://www.ollydbg.de/>



OllyDbg加载可执行程序



OllyDbg加载可执行程序





日新月异 允允公允

OllYDbg - Lab09-01.exe - [CPU - main thread, module Lab09-01]

File View Debug Trace Options Windows Help

Disassembler Highlight: next instruction to be executed

Registers (FPU)

Stack [0012FF88]=0
EBP=0012FF94

Lab09-01.<ModuleEntryPoint>

Memory dump

Stack

Paused

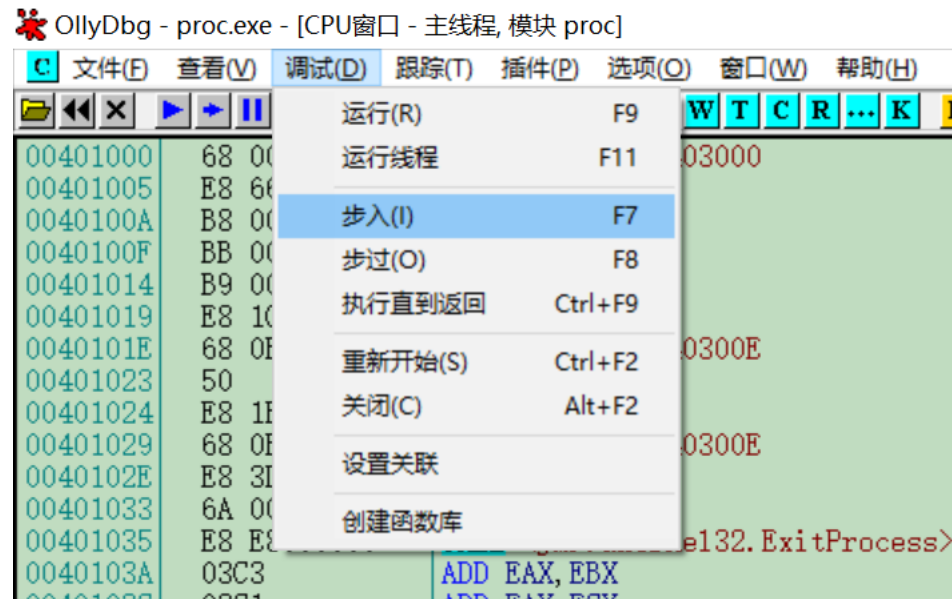
Detailed description of the OllYDbg interface: The screenshot shows the OllYDbg debugger window. The title bar reads 'OllYDbg - Lab09-01.exe - [CPU - main thread, module Lab09-01]'. The menu bar includes 'File', 'View', 'Debug', 'Trace', 'Options', 'Windows', and 'Help'. Below the menu is a toolbar with various icons for navigation and execution. The main window is divided into several panes. The left pane shows the disassembled code, with the instruction 'JMP SHORT 00403892' highlighted. A green box labeled 'Disassembler Highlight: next instruction to be executed' points to this instruction. The right pane shows the 'Registers (FPU)' window, displaying the values of various registers. A green box labeled 'Registers' points to this window. The bottom pane shows the 'Stack' window, displaying the memory dump. A green box labeled 'Stack' points to this window. The status bar at the bottom indicates 'Paused'.





允公允能 日新月异

步入、步过调试





允公允能 日新月异

调试“Hello World”

OllyDbg - hello.exe - [CPU窗口 - 主线程, 模块 hello]

文件(F) 查看(V) 调试(D) 跟踪(T) 插件(P) 选项(O) 窗口(W) 帮助(H)

U L E M W T C R ... K B M H

地址	十六进制数据	多字节 (ANSI/OEM - 简体中文)
00401000	68 00304000	PUSH OFFSET 00403000
00401005	E8 0A000000	CALL 00401014
0040100A	6A 00	PUSH 0
0040100C	E8 B1000000	CALL <JMP.&kernel32.ExitProcess>
00401011	CC	INT3
00401012	CC	INT3
00401013	CC	INT3
00401014	55	PUSH EBP
00401015	8BEC	MOV EBP, ESP
00401017	83C4 F4	ADD ESP, -0C
0040101A	6A F5	PUSH -0B
0040101C	E8 A7000000	CALL <JMP.&kernel32.GetStdHandle>
00401021	8945 FC	MOV DWORD PTR SS:[EBP-4], EAX
00401024	FF75 08	PUSH DWORD PTR SS:[EBP+8]
00401027	E8 24000000	CALL 00401050
0040102C	8945 F4	MOV DWORD PTR SS:[EBP-0C], EAX
0040102F	6A 00	PUSH 0

Stack [0019FF70]=0
Imm=hello.00403000, ASCII "Hello World!", LF, CR

寄存器 (FPU)

寄存器	值	注释
EAX	0019FFCC	
ECX	00401000	hello.<ModuleEn
EDX	00401000	hello.<ModuleEn
EBX	00292000	
ESP	0019FF74	
EBP	0019FF80	
ESI	00401000	hello.<ModuleEn
EDI	00401000	hello.<ModuleEn
EIP	00401000	hello.<ModuleEn

返回到 KERNEL32.75BD0419

返回到 ntdll.77DB662D



南开大学
Nankai University



南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



2. 控制流转移指令



允公允能 日新月异

控制转移

- 控制转移（transfer of control）是一种改变汇编语句执行顺序的方法。
 - 无条件转移
 - 条件转移





允公允能 日新月异

无条件转移 Unconditional Jump

- 将CPU控制权直接转移到指定的汇编语句
 - 修改EIP为指定的内存地址
 - CPU从EIP指定的内存地址读取下一条机器指令



南开大学
Nankai University



允公允能 日新月异

JMP指令

- **JMP** 目的地址
- JMP指令实现CPU控制权的无条件跳转
- 目的地址是代码标号
 - 代码标号被**汇编器**翻译成内存地址
 - **CPU**看到的是内存地址，不是代码标号



南开大学
Nankai University



允公允能 日新月异

循环

top:

... ..

... ..

JMP top



无限循环





允公允能 日新月异

条件跳转指令 Conditional Jump

- JA
- JB
- JE
- JNE
-





允公允能 日新月异

CMP指令

- CMP指令，比较目的操作数和源操作数
 - CMP reg, reg
 - CMP reg, imm
 - CMP mem, reg
 - CMP mem, imm
 - CMP reg, mem





允公允能 日新月异

CMP指令

- 执行从源操作数中减掉目的操作数的减法操作
- 设置相应的标志位，不改变操作数
- 标志位：OF、SF、ZF、AF、PF、CF



南开大学
Nankai University



允公允能 日新月异

MOV EAX, 100h

MOV EBX, 200h

CMP EAX, EBX

JA L1

INVOKE StdOut, ADDR str1

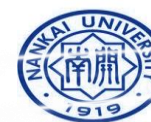
JMP L2

L1:

INVOKE StdOut, ADDR str2

L2:

INVOKE ExitProcess, 0



南开大学
Nankai University



条件跳转指令Conditional Jump

有符号数的条件跳转指令
Conditional jump instructions used on signed data

Instruction	Description	Flags tested
JE/JZ	Jump Equal or Jump Zero	ZF
JNE/JNZ	Jump not Equal or Jump Not Zero	ZF
JG/JNLE	Jump Greater or Jump Not Less/Equal	OF, SF, ZF
JGE/JNL	Jump Greater/Equal or Jump Not Less	OF, SF
JL/JNGE	Jump Less or Jump Not Greater/Equal	OF, SF
JLE/JNG	Jump Less/Equal or Jump Not Greater	OF, SF, ZF





条件跳转指令 Conditional Jump

无符号数的条件跳转指令

Conditional jump instructions used on **unsigned data**

Instruction	Description	Flags tested
JE/JZ	Jump Equal or Jump Zero	ZF
JNE/JNZ	Jump not Equal or Jump Not Zero	ZF
JA/JNBE	Jump Above or Jump Not Below/Equal	CF, ZF
JAЕ/JNB	Jump Above/Equal or Jump Not Below	CF
JB/JNAE	Jump Below or Jump Not Above/Equal	CF
JBE/JNA	Jump Below/Equal or Jump Not Above	AF, CF





允公允能 日新月异

LOOP指令

- **LOOP** 目的地址
- LOOP指令可以指定循环执行的次数 (loop count)
 - ECX寄存器作为循环计数器
 - LOOP指令执行时, ECX减1
 - 如果**ECX**不等于**0**, 跳转到目的地址
 - 如果**ECX**等于**0**, 不跳转, 顺序执行





允公允能 日新月异

LOOP指令

MOV EAX 10h

MOV ECX 10h

L1:

INC EAX

LOOP L1



南开大学
Nankai University



允公允能 日新月异

LOOP指令

- LOOP指令先ecx减1，然后判断ecx是否为0.
- LOOP is exactly like `dec ecx / jnz`





允公允能 日新月异

循环的嵌套

.data

count DWORD 0

.code

MOV ECX, 100; L1 循环100次

L1:

MOV count, ECX

MOV ECX, 10 ; L2 循环10次

L2:

... ..

LOOP L2

MOV ECX, count

LOOP L1



南开大学
Nankai University



允公允能 日新月异

数组求和

.data

array DWORD 100h, 200h, 300h, 400h

.code

MOV **ECX**, LENGTHOF array ; 循环次数

MOV EDI, OFFSET array

MOV EAX, 0

L1:

ADD EAX, [EDI]

ADD EDI, TYPE array

LOOP L1





允公允能 日新月异

字符串赋值

.data

src BYTE "Hello World", 0Dh, 0Ah, 0

dst BYTE SIZEOF src DUP(0), 0

.code

MOV ECX, SIZEOF src

MOV ESI, 0

L1:

MOV AL, BYTE PTR src[ESI]

MOV BYTE PTR dst[ESI], AL

INC ESI

LOOP L1





允公允能 日新月异

.data

num BYTE 1, 2, 3, 4, 5, 6, 7, 8, 9, 0

.code

MOV ECX, 10

MOV ESI, 0

L1:

MOV AL, BYTE PTR num[ESI]

ADD AL, 30h

MOV BYTE PTR num[ESI], AL

INC ESI

LOOP L1





南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



3. 过程的定义和使用



允公允能 日新月异

过程

- C++中的函数定义

```
return_type function_name( parameter list ) {  
  
    body of the function  
  
}
```





允公允能 日新月异

过程

- **返回类型**：一个函数可以返回一个值。**return_type** 是函数返回值的数据类型。
- **函数名称**：这是函数的实际名称。函数名和参数列表一起构成了函数签名。



南开大学
Nankai University



允公允能 日新月异

过程

- **参数**：参数就像是占位符。当函数被调用时，向参数传递一个值，这个值被称为**实际参数**。参数列表包括函数参数的类型、顺序、数量。
- **函数主体**：函数主体包含一组定义函数执行任务的语句。



南开大学
Nankai University



允公允能 日新月异

过程

```
int max(int num1, int num2) {  
    // 局部变量声明  
    int result;  
    if (num1 > num2)  
        result = num1;  
    else  
        result = num2;  
    return result;  
}
```





允公允能 日新月异

过程

- 汇编语言中，一般使用术语“**过程**”（procedure）
表示高级语言中的函数、方法
 - 函数（function），C/C++中称为函数
 - 方法（method），JAVA中称为方法





允公允能 日新月异

过程

- 汇编语言早于“面向对象”、“面向函数”的编程语言
 - 编译器是如何将这些高级语言中的函数、过程翻译成汇编语言的？





允公允能 日新月异

过程

- 汇编语言把过程定义为以返回语句结束的命名语句块。
 - 使用PROC和ENDP伪指令来声明过程
 - 必须定义一个过程名字（标识符）





允公允能 日新月异

过程

my_proc **PROC**

... ..

ret

my_proc **ENDP**

- 除启动过程之外，其它过程以ret指令结束
 - 将CPU控制权转移到过程被调用的地方



南开大学
Nankai University



允公允能 日新月异

启动过程（main）

main PROC

... ..

main ENDP

- 启动过程（main）的返回语句是
 - INVOKE ExitProcess, 0
 - 将CPU的控制权转移给Windows操作系统





允公允能 日新月异

过程的定义

- MyProc过程，计算寄存器EAX、EBX、ECX之和

MyProc **PROC**

ADD EAX, EBX

ADD EAX, ECX

RET

MyProc **ENDP**





允公允能 日新月异

过程的调用与返回

- **CALL**指令将CPU的控制权转移到新的内存地址执行指令，实现过程的调用
- **RET**指令将CPU的控制权返回到程序中过程被调用的地方继续执行



南开大学
Nankai University



允公允能 日新月异

过程的调用与返回

main PROC

MOV EAX, 1000h

MOV EBX, 1000h

MOV ECX, 1000h

CALL MyProc

INVOKE ExitProcess, 0

main ENDP

MyProc PROC

ADD EAX, EBX

ADD EAX, ECX

RET

MyProc ENDP



南开大学
Nankai University



允公允能 日新月异

过程的调用与返回

- 过程返回地址的保存
 - CALL指令调用之后，将过程的返回地址压入堆栈，将过程入口地址赋值给EIP，实现CPU控制权的转移
 - RET指令调用之后，将过程的返回地址赋值给EIP寄存器，实现CPU控制权的转移



南开大学
Nankai University



过程的调用与返回

00401000	68 00304000	PUSH OFFSET 00403000	ASCII "Hello World", CR, LF	寄存器 (FPU)
00401005	E8 66000000	CALL 00401070		EAX 00003000
0040100A	B8 00100000	MOV EAX, 1000		ECX 00001000
0040100F	BB 00100000	MOV EBX, 1000		EDX 0019FF1C
00401014	B9 00100000	MOV ECX, 1000		EBX 00001000
00401019	E8 1C000000	CALL 0040103A		ESP 0019FF70
0040101E	68 0E304000	PUSH OFFSET 0040300E		EBP 0019FF80
00401023	50	PUSH EAX		ESI 0040103F proc.<ModuleEntry
00401024	E8 1B000000	CALL 00401044		EDI 0040103F proc.<ModuleEntry
00401029	68 0E304000	PUSH OFFSET 0040300E		EIP 0040101E proc.0040101E
0040102E	E8 3D000000	CALL 00401070		C 0 ES 002B 32Bit 0(FFFFFFFF)
00401033	6A 00	PUSH 0		P 1 CS 0023 32Bit 0(FFFFFFFF)
00401035	E8 E8000000	CALL <JMP.&kernel32.ExitProcess>	跳转至 KERNEL32.ExitProcess	A 0 SS 002B 32Bit 0(FFFFFFFF)
0040103A	03C3	ADD EAX, EBX		Z 0 DS 002B 32Bit 0(FFFFFFFF)
0040103C	03C1	ADD EAX, ECX		S 0 FS 0053 32Bit 308000(FFF)
0040103E	C3	RETN		T 0 GS 002B 32Bit 0(FFFFFFFF)
0040103F	E8 BCFFFFFF	CALL 00401000		D 0
00401044	55	PUSH EBP		O 0 LastErr 00000000 ERROR_SU
00401045	BB 00100000	MOV EBP, ECD		
Stack [0019FF6C]=proc.0040101E				
地址	十六进制数据	多字节 (ANSI/OEM - 简体中文)	0019FF70 00401044 返回到 proc.00401000 来自 proc.00401	
00403000	48 65 6C 6C 6F 20 57 6F 72 6C 64 0D 0A 00 00 00	He l l o W o r l d	0019FF74 74CB0419 返回到 KERNEL32.74CB0419	





南開大學

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月異



4. 调用链接库中的函数



允公允能 日新月异

链接库

- 链接库（Link Library）是一个文件，包含已经编译成机器码的过程。
 - `includelib \masm32\lib\masm32.lib`
 - `includelib \masm32\lib\kernel32.lib`





允公允能 日新月异

PROTO伪指令

- PROTO伪指令用于声明链接库中的过程
- `include \masm32\include\masm32.inc`
- `include \masm32\include\kernel32.inc`





允公允能 日新月异

PROTO伪指令

- StdOut
 - StdOut PROTO :DWORD
- StdIn
 - StdIn PROTO :DWORD,:DWORD
- ExitProcess
 - ExitProcess PROTO STDCALL :DWORD





南开大学

NANKAI UNIVERSITY, P.R. CHINA 1919

允公允能 日新月异



汇编语言与逆向技术

第5章 过程

王志

zwang@nankai.edu.cn

updated on 2022.10.13

南开大学 网络空间安全学院
2022-2023学年



允公允能 日新月异

本章知识点

1. 程序动态调试
2. 控制流转移指令
3. 过程的定义和使用
4. 调用链接库中的函数

