汇编语言的准备知识-给初次接触汇编者1 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/138/2021_2022__E6_B1_87_E7_BC_96_E8_AF_AD_E8_c98_138282.htm 汇编语言和CPU以及内存,端口等硬件知识是连在一起的. 这也是为什么汇编语言没有通用性的原因. 下面简单讲讲基本知识(针对INTEL x86及其兼容机) x86汇编语言的指令,其操作对象是CPU上的寄存器,系统内存,或者立即数. 有些指令表面上没有操作数, 或者看上去缺少操作数, 其实该指令有内定的操作对象, 比如push指令, 一定是对SS:ESP指定的内存操作, 而cdq的操作对象一定是eax / edx. 在汇编语言中,寄存器用名字来访问. CPU 寄存器有好几类, 分别有不同的用处: 1. 通用寄存器: EAX,EBX,ECX,EDX,ESI,EDI,EBP,ESP(这个虽然通用,但很少被用做除了堆栈指针外的用途) 这些32位可以被用作多种用途, 但每一个都有"专长". EAX 是"累加器"(accumulator), 它是很多加法乘法指令的缺省寄存器. EBX 是"基地址"(base)寄存器, 在内存寻址时存放基地址. ECX 是计数器(counter), 是重复(REP)前缀指令和LOOP指令的内定计数器. EDX是...(忘了..哈哈)但它总是被用来放整数除法产生的余数. 这4个寄存器的低16位可以被单独访问,分别用AX,BX,CX和DX. AX又可以单独访问低8位(AL)和高8位(AH), BX,CX,DX也类似. 函数的返回值经常被放在EAX中. ESI/EDI分别叫做"源/目标索引寄存器"(source/destination index),因为在很多字符串操作指令中, DS:ESI指向源串,而ES:EDI指向目标串. EBP是"基址指针"(BASE POINTER), 它最经常被用作高级语言函数调用的"框架指针"(frame pointer). 在破解的时候,经常可以看见一个标准的函

数起始代码: push ebp .保存当前ebp mov ebp,esp .EBP设为当前堆栈指针 sub esp, xxx .预留xxx字节给函数临时变量. ... 这样一来,EBP 构成了该函数的一个框架, 在EBP上方分别是原来的EBP, 返回地址和参数. EBP下方则是临时变量. 函数返回时作 mov esp,ebp/pop ebp/ret 即可. ESP 专门用作堆栈指针. 2. 段寄存器: CS(Code Segment，代码段) 指定当前执行的代码段. EIP (Instruction pointer, 指令指针)则指向该段中一个具体的指令. CS:EIP指向哪个指令, CPU 就执行它. 一般只能用jmp, ret, jnz, call 等指令来改变程序流程,而不能直接对它们赋值. DS(DATA SEGMENT, 数据段) 指定一个数据段. 注意:在当前的计算机系统中, 代码和数据没有本质差别, 都是一串二进制数, 区别只在于你如何用它. 例如, CS 制定的段总是被用作代码, 一般不能通过CS指定的地址去修改该段. 然而,你可以为同一个段 申请 一个数据段描述符"别名"而通过DS来访问/修改. 自修改代码的程序常如此做. ES,FS,GS 是辅助的段寄存器, 指定附加的数据段. SS(STACK SEGMENT)指定当前堆栈段. ESP 则指出该段中当前的堆栈顶. 所有push/pop 系列指令都只对SS:ESP指出的地址进行操作.