帮助读者快速入门、快速熟悉
逆向软件的实战书籍

Reverse Analysis And
Tool HandBook

逆向分析实战

冀云◎编著

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

帮助读者快速入门、快速熟悉
逆向软件的实战书籍

Reverse Analysis And
Tool HandBook

逆向分析实战

冀云◎编著

人民邮电出版社
北京

图书在版编目（CIP）数据

逆向分析实战 / 冀云编著. — 北京 : 人民邮电出版社, 2017.9
ISBN 978-7-115-46579-5

Ⅰ. ①逆… Ⅱ. ①冀… Ⅲ. ①软件工程 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2017)第268817号

内 容 提 要

本书的主要内容为：数据的存储及表示形式、汇编语言入门、熟悉调试工具 OllyDbg、PE 工具详解、PE 文件格式实例（包括加壳与脱壳工具的使用）、十六进制编辑器与反编译工具、IDA 与逆向、逆向工具原理实现等。

本书可以作为程序员、安全技术的研究人员、安全技术爱好者阅读。

◆ 编　　著　冀　云
责任编辑　张　涛
责任印制　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：18
字数：426 千字　　2017 年 9 月第 1 版
印数：1 – 2 500 册　　2017 年 9 月北京第 1 次印刷

定价：59.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

前言

也许你想知道什么是软件逆向，也许你已经听说过软件逆向，从而想要学习软件逆向。不管你抱着什么目的翻开本书，笔者还是在你阅读本书之前，先来说一些软件逆向的知识！

什么是“软件逆向工程”

术语“逆向工程”源自硬件领域，在软件领域目前还没有明确的定义。就笔者个人的理解简单来说，软件逆向是通过观察分析软件或程序的行为、数据和代码等，来还原其设计实现，或者推导出更高抽象层次的表示。

软件工程与软件逆向工程的区别

对于软件工程而言，软件的设计讲究封装，将各个模块进行封装，将具体的实现进行隐藏，只暴露一个接口给使用者。对于模块的使用者而言，封装好的模块相当于一个“黑盒子”，使用者使用“盒子”时，无需关心“盒子”的内部实现，只需要按照模块预留的接口进行使用即可。

软件逆向工程对于软件工程而言，却是正好相反的。对软件进行逆向工程时要查看软件的行为，即软件的输入与输出的情况；要查看软件的文件列表，即软件使用了哪些动态链接库（哪些动态连接库是作者编写的，哪些动态连接库是系统提供的），有哪些配置文件，甚至还要通过一系列的工具查看软件的文件结构、反汇编代码等。

对比软件工程与软件逆向工程可以发现，软件工程是在封装、实现一个具备某种功能的“黑盒子”，而软件逆向工程则是在分析“黑盒子”并尝试还原封装的实现与设计。后者对于前者而言是一个相反的过程，因此称为“软件逆向工程”。

学习软件逆向工程与软件工程的区别

对于软件逆向工程而言，学习逆向知识，除了要学习逆向知识本身外，还需要掌握各种不同的逆向工具，或者说逆向知识中重要的一个环节就是逆向工具的使用。对于软件开发而言，软件开发工具在软件开发中所占据的位置远远达不到逆向工具在逆向领域中的位置。因此，读者在学习编程时可能更注重的是编程语言本身而不是工具，但是在学习逆向时，逆向知识是不可能抛开逆向工具而独立进行学习的。

本书的主要内容

本书全面讲解了软件逆向工程的知识，即包括主流的技术，如加壳与脱壳、汇编、数据的存储等；也有实用的工具，如主流的调试工具 OllyDbg、PE 工具、加壳与脱壳工具、十六进制编辑与反编译工具、IDA 与逆向、逆向工具实现等。

我们的目的是快速入门

对于软件逆向工程的初学者而言，我们的目的只有一个，那就是快速地入门，本书的目的也是为了帮助读者能够快速地熟悉软件逆向的知识和软件逆向的工具。本书以软件逆向工具为主线，以逆向工具为重点，对软件逆向相关知识进行了介绍。

无论是学习编程，还是学习逆向，我们学习的都是技术，技术的掌握重点在于练习和实践，因此希望读者在学习本书的过程中不断地练习和实践，从而能够真正达到快速入门的效果。

学习本书要求读者有C语言的编程知识，如果连一点编程的基础都没有，那么可能很多章节的知识是无法掌握的，学习起来会相当吃力。

免责声明

本书是一本面向零基础读者的书籍，书中介绍的是逆向工程的入门知识，仅供读者自学之用。对本书介绍的知识用于非法用途的，导致的后果请自行承担，和作者及出版社无关，请读者自觉遵守国家的法律。

由于作者水平有限，必定会有差错，敬请谅解。

祝大家学习愉快！

本书编辑联系邮箱：zhangtao@ptpress.com.cn。

作者

目录

第1章 数据的存储及表示形式

学习过计算机的读者都知道，计算机中的各种数据都是以二进制形式进行存储的，无论是文本文件、图片文件，还是音频文件、视频文件、可执行文件等，统统都是由二进制文件存储的。学习过计算机的读者在学习计算机基础的时候一定学习过进制转换，也一定学习过数据的表示方式等，大部分人在学习这部分知识时会觉得枯燥、无用，但是对于学习逆向知识和使用逆向工具，数据的存储及表示形式是必须要掌握的。

本章借助 OllyDbg 这款调试工具来一起讨论数据的存储及表示形式，让读者对于学习计算机的数据存储及表示可以更加的感性，从而脱离纯粹理论性的学习。

本章内容较为枯燥，但是着实是学习逆向的基础知识，对于从来没有接触过逆向或者是刚开始接触逆向的读者，本章内容还是有一定帮助的。

本章关键字： 进制 数据表示 数据转换 数据存储

1.1 进制及进制的转换

了解进制的概念及进制的转换是学习逆向的基础，因为计算机使用的进制是二进制，它又不同于我们现实生活中使用的十进制，因此我们必须学习不同的进制及进制之间的转换。

1.1.1 现实生活中的进制与计算机的二进制

我们在现实生活中会接触到多种多样的进制，通常见到的有十进制、十二进制和二十四进制等。下面分别对这几种进制进行举例说明。

十进制是每个人从上学就开始接触和学习的进制表示方法。所谓的十进制，就是逢十进一，最简单的例子就是 9+1=10。这个无需过多解释。

十二进制也是我们日常生活中常见的表示方法。所谓的十二进制，就是逢十二进一，例如 12 个月为 1 年，13 个月就是 1 年 1 个月。

二十四进制也是我们日常生活中常见的表示方法。所谓的二十四进制，就是逢二十四进一，例如 24 小时为 1 天，25 小时就是 1 天 1 小时。

介绍了以上现实生活中的例子后，我们再来说说计算机中的二进制。根据前面各种进制的解释，我们可以想到，二进制就是逢二进一。这里举个不太恰当的例子，例如 2 斤就是 1 公斤。

在计算机中为什么使用二进制呢？简单说就是计算机用高电平和低电平来表示 1 和 0 最为方便和稳定，高电平被认为是 1，低电平被认为是 0，这就是所谓的二进制的来源。

由于二进制在阅读上不方便，计算机又引入了十六进制来直观地表示二进制。所谓的十六进制，就是逢十六进一。

因此在计算机中，我们常见的数据表示方法有二进制、十进制和十六进制。

1.1.2　进制的定义

在学习小学数学的时候我们就学习了十进制，十进制一共有十个数字，从 0 一直到 9，9 再往后数一个的时候要产生进位，也就是逢十进一。总结十进制的定义则是，由 0 到 9 十个数字组成，并且逢十进一。

举一反一地来说，二进制的定义是，由 0 到 1 两个数字组成，逢二进一。十六进制的定义是由 0 到 9 十个数字和 A 到 F 六个字母组成，逢十六进一。

由此，我们衍生出 N 进制的定义是，由 N 个符号组成，逢 N 进一。

表 1-1 所列为这三种进制的数字表。

表 1-1　　二进制、十进制和十六进制数字表

数　制	基　数	数　字
二进制	2	0 1
十进制	10	0 1 2 3 4 5 6 7 8 9
十六进制	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

1.1.3　进制的转换

在逆向当中，我们直接面对的通常是十六进制，而由于很多原因，我们需要将其当作十进制或二进制来查看，当然也有可能需要根据二进制转换成十六进制或十进制。所以，我们就需要掌握进制之间的转换。

1．二进制转十进制

二进制整数的每个位都是 2 的幂次方，最低位是 2 的 0 次方，最高为是 2 的（N-1）次方，我们通过一个例子进行说明。我们把二进制数 10010011 转换成十进制数，计算方式如下：

$10010011 = 1\times2^7+0\times2^6+0\times2^5+1\times2^4+0\times2^3+0\times2^2+1\times2^1+1\times2^0=128+0+0+16+0+0+2+1=147$

我们得出的结果是，把二进制 10010011 转换成十进制后是 147。我们用计算机进行验算，如图 1-1 和图 1-2 所示。

从图 1-1 和图 1-2 中可以看出，我们的计算结果是正确的，由此读者在计算二进制时按照上面转换的例子进行转换即可。

注意：（1）当读者打开计算器时，可能出现的样子与书中的样子不相同，可能看着会更简单。为了能够计算不同进制，我们选择菜单栏的“查看”→“程序员”，即可使用进制之间的转换。

（2）在刚开始学习的时候，建议读者自行进行进制的转换，熟练以后再使用计算器进行转换。

图 1-1 验算二进制（一）

图 1-2 验算二进制（二）

2．十六进制与二进制的转换

由于一个简单的数值用二进制表示需要很长的位数，这样对于阅读很不方便，因此汇编和调试器常用十六进制表示二进制。十六进制的每个位可以代表 4 个二进制位，因为 2 的 4 次方刚好是 16。这样，在二进制与十六进制之间就产生了一个很好的对应关系，如表 1-2 所列。

表 1-2 二进制对应的十六进制与十进制数

二进制	十进制	十六进制	二进制	十进制	十六进制
0000	0	0	0110	6	6
0001	1	1	0111	7	7
0010	2	2	1000	8	8
0011	3	3	1001	9	9
0100	4	4	1010	10	A
0101	5	5	1011	11	B
1100	12	C	1110	14	E
1101	13	D	1111	15	F

根据此表，我们可以很快地把二进制和十六进制进行转换，把上例的二进制 10010011 转换成十六进制，转换过程如下：

第一步，把 10010011 从最低开始按每四位分为一组，不足四位前面补 0，划分结果为 1001 0011；

第二步，把划分好的组进行查表，1001 对应十六进制是 9，0011 对应的十六进制是 3。

那么，二进制 10010011 转换成十六进制后的值是 93。读者可以通过计算器自行进行验算。

在逆向中常用的就是二进制与十进制的转换，或者是二进制与十六进制的转换，其他的转换方式读者可以自行查找资料进行学习。关于十六进制和二进制需要记住的重要一点就是，一位十六进制数可以表示四位二进制数。

1.2 数据宽度、字节序和 ASCII 码

前面介绍了计算机中常用的进制表示方法和转换，现在读者知道了计算机存储的都是二进制的数据，那么接下来要讨论的是在计算机中数据存储的单位以及数据是如何存储在存储空间的。

1.2.1 数据的宽度

数据的宽度是指数据在存储器中存储的尺寸。在计算机中，所有数据的基本存储单位都是字节（byte），每个字节占 8 个位（位是计算机存储的最小单位，而不是基本单位，因为在存储数据时几乎没有按位进行存储的）。其他的存储单位还有字（word）、双字（dword）和八字节（qword）。

图 1-3 给出各个存储单位所包含的位数。

在计算机编程中，常用的几个重要数据存储单位分别就是 byte、word 和 dword，这几个存储单位稍后我们会使用到。

常用存储单位所占字节数与位数

单位	所占位数	所占字节数
字节(byte)	8位	1字节
字（word）	16位	2字节
双字（dword）	32位	4字节
八字节（qword）	64位	8字节

图 1-3　常用存储单位所在字节数与位数

1.2.2 数值的表示范围

在计算机中存储数值时，也是要依据前面介绍过的数据宽度进行存储的，那么在存储数据时由于存储数据的宽度限制，数值的表示也是有范围限制的。那么 byte、word 和 dword 能存储多少数据呢？我们先来计算一下，如果按位存储的话，能存储多少个数据，再分别来计算以上三种单位能够存储的数值的范围。

计算机使用二进制进行数据存储时，一位二进制最多能表示几个数呢？因为是二进制数，只存在 0 和 1 两个数，所以一位二进制数最多能表示两个数，分别是 0 和 1。那么，两位二进制最多能表示几个数呢？因为一位二进制数能表示两个数，所以两位二进制数则能表示 2 的 2 次方个数，即 4 个数，分别是 0、1、10、11。进一步地，三位二进制数能表示的就是 2 的 3 次方个数，即 8 个数，分别是 0、1、10、11、100、101、110、111。

上面的过程可以整理成表 1-3。

表 1-3　　*N* 位二进制位能够表示的数

二进制位数	表示数的个数	表示的数	2 的 *N* 次方
1	2	0、1	2 的 1 次方
2	4	0、1、10、11	2 的 2 次方
3	8	0、1、10、11、100、101、110、111	2 的 3 次方

根据表 1-3 计算的 byte、word 和 dword 三种数据存储宽度能表示的数据的范围如表 1-4 所列。

表 1-4 无符号整数的表示范围

存储单位	十进制范围	十六进制范围	2 的 N 次方
byte	0～255	0～FF	2 的 8 次方
word	0～65535	0～FFFF	2 的 16 次方
dword	0～4294967295	0～FFFFFFFF	2 的 32 次方

2 的 8 次方是 256，为什么数值只有 0～255 个呢？因为计算机计数是从 0 开始，从 0 到 255 同样是 256 个数，这里的 2 的 8 次方表示能够表示数值的个数，而不是能够表示数值的最大的数。

这里只给出了无符号整数的表示范围，那么什么是无符号呢？数值分为有符号数和无符号数，有符号数是分整数和负数的，而无符号数值有整数没有负数。负数在计算机中的表示有符号数时借助了最高位来进行，如果最高位是 0，那么就是整数，如果最高位是 1 则是负数。关于有符号数和无符号数不必过多地纠结，因为计算机表示数据是不区分有符号还是无符号的，有符号还是无符号是人在进行区分。这里就不做过多地解释了。

1.2.3 字节序

字节序也称为字节顺序，在计算机中对数值的存储有一定的标准，而该标准随着系统架构的不同而不同。了解字节存储顺序对于逆向工程是一项基础知识，在动态分析程序的时候，往往需要观察内存数据的变化情况，这就需要我们在掌握数据的存储宽度、范围之后，进一步了解字节顺序。

通常情况下，数值在内存中存储的方式有两种，一种是大尾方式，另一种是小尾方式。关于字节序的知识，通过一个简单的例子就可以掌握。

比如有 0x01020304（C 语言中对十六进制数的表示方式）这样一个数值，如果用大尾方式存储，其存储方式为 01 02 03 04，而用小尾方式进行存储则是 04 03 02 01，用更直观的方式展示其区别，如表 1-5 所列。

表 1-5 字节顺序对比表

大尾方式		小尾方式	
数据	地址值	数据	地址值
01	00000000H	04	00000000H
02	00000001H	03	00000001H
03	00000002H	02	00000002H
04	00000003H	01	00000003H

从两个地址列可以看出，地址的值都是一定的，没有变化，而数据的存储顺序却是不相同的。从表中可以得到如下结论。

大尾存储方式：内存高位地址存放数据低位字节数据，内存低位地址存放数据高位字节数据；

小尾存储方式：内存高位地址存放数据高位字节数据，内存低位地址存放数据低位字节数据。

通常情况下，Windows 操作系统兼容的 CPU 为小尾存储方式，而 Unix 操作系统兼容的 CPU 多为大尾存储方式。在网络中传输的数据的字节顺序使用的是大尾存储方式。

1.2.4　ASCII 码

计算机智能存储二进制数据，那么计算机是如何存储字符的呢？为了存储字符，计算机必须支持特定的字符集，字符集的作用是将字符映射为整数。早期字符集仅仅使用 8 个二进制数据位进行存储，即 ASCII 码。后来，由于全世界语言的种类繁多，又产生了新的字符集 Unicode 字符编码。

ASCII 码是美国标准信息交换码的字母缩写，在 ASCII 字符集中，每个字符由唯一的 7 位整数表示。ASCII 码仅使用了每个字节的低 7 位，最高位被不同计算机用来创建私有字符集。由于标准 ASCII 码仅使用 7 位，因此十进制表示范围是 0～127 共 128 个字符。

在编程与逆向中都会用到 ASCII 码，因此有必要记住常用的 ASCII 字符对应的十六进制和十进制数。常用的 ASCII 字符如表 1-6 所列。

表 1-6　　常用 ASCII 码表

字　符	十进制	十六进制	说　明
LF	10	0AH	换行
CR	13	0DH	回车
SP	32	20H	空格
0～9	48～57	30H～39H	数字
A~Z	65~90	41H~5AH	大写字母
a~z	97~122	61H~7AH	小写字母

表 1-6 是经常使用到的 ASCII 字符，这些字符是经常会见到和用到的，希望读者能将其保存，以便使用之时可以快速查阅。

Unicode 编码是为了使字符编码更进一步符合国际化而进行的扩展，Unicode 使用一个字（也就是两个字节，即 16 位）来表示一个字符。这里不做过多的介绍。

1.3　在 OD 中查看数据

在逆向分析中，调试工具可以说是非常重要的。调试器能够跟踪一个进程的运行时状态，在逆向分析中称为动态分析工具。动态调试会用在很多方面，比如漏洞的挖掘、游戏外挂的分析、软件加密解密等方面。本节介绍应用层下最流行的调试工具 OllyDbg。

OllyDbg 简称 OD，是一款具有可视化界面的运行在应用层的 32 位的反汇编逆向调试分析工具。OD 是所有进行逆向分析人员都离不开的工具。它的流行，主要原因是操作简单、参考文档丰富、支持插件功能等。

熟悉 OD

OD 的操作非常简单，但是由于逆向是一门实战性和综合性非常强的技术，因此要真正熟练掌握 OD 的使用却并不是容易的事，单凭操作而言看似没有太多的技术含量，但是其真正的精髓在于配合逆向的思路来达到逆向者的目的。

1. OD 的选型

为什么先介绍 OD 的选型，而不直接开始介绍 OD 的使用呢？OD 的主流版本是 1.10 和待崛起的 2.0。虽然它的主流版本是 1.10，但是它仍然存在很多修改版。所谓修改版，就是由用户自己对 OD 进行修改而产生的，类似于病毒的免杀。OD 虽然是动态调试工具，但是由于其强大的功能经常被很多人用在软件破解等方面，导致很多作者的心血付诸东流。软件的作者为了防止软件被 OD 调试，加入了很多专门针对 OD 进行调试的反调试功能来保护自己的软件不被调试，从而不被破解；而破解者为了能够继续使用 OD 来破解软件，则不得不对 OD 进行修改，从而达到反反调试的效果。

调试、反调试、反反调试，对于新接触调试的爱好者来说容易混淆。简单来说，反调试是阻止使用 OD 进行调试，而反反调试是突破反调试继续进行调试。OD 的修改版本之所以很多，目的就是为了能够更好地突破软件的反调试功能。

因此，如果从学习的角度来讲，建议选择原版的 OD 进行使用。在使用的过程中，除了会掌握很多调试技巧外，还会学到很多反调试的技巧，从而掌握反反调试的技巧。如果在实际的应用中，则可以直接使用修改版的 OD，避免 OD 被软件反调试，从而提高逆向调试分析的速度。

注意：修改版本的 OD 可以叫着千奇百怪的名字，比如 OllyICE、OllySeX 等。

2. 熟悉 OD 主界面

OD 的发行是一个压缩包，解压即可运行使用，运行 OD 解压目录总的 ollydbg.exe 程序，就会出现一个分布恰当、有菜单有面板和能输入命令的看着很强大的软件窗口，如图 1-4 所示。

在图 1-4 的 OD 调试主窗口中的工作区大致可以分为 6 个部分，按照从左往右、从上往下，这 6 部分分别是反汇编窗口、信息提示窗口、数据窗口、寄存器窗口、栈窗口和命令窗口。下面分别介绍各个窗口的用法。

反汇编窗口：该窗口用于显示反汇编代码，调试分析程序主要在这个窗口中进行，这也是进行调试分析的主要工作窗口。

信息提示窗口：该窗口用于显示与反汇编窗口中上下文环境相关的内存、寄存器或跳转来源、调用来源等信息。

数据窗口：该窗口用于以多种格式显示内存中的内容，可使用的格式有 Hex、文本、短型、长型、浮点、地址和反汇编等。

寄存器窗口：该窗口用于显示各个寄存器的内容，包括前面介绍的通用寄存器、段寄存器、标志寄存器、浮点寄存器。另外，还可以在寄存器窗口中的右键菜单选择显示 MMX 寄存器、3DNow!寄存器和调试寄存器等。

栈窗口：该窗口用于显示栈内容、栈帧，即 ESP 或 EBP 寄存器指向的地址部分。

命令窗口：该窗口用于输入命令来简化调试分析的工作，该窗口并非基本窗口，而是由 OD 的插件提供的功能，由于几乎所有的 OD 使用者都会使用该插件，因此有必要把它也列入主窗口中。

图 1-4　OD 调试主窗口

3. 在数据窗口中查看数据

前面已经介绍，OD 是一款应用层下的调试工具，它除了可以进行软件的调试以外，还可以帮助我们学习前面介绍的数据宽度、进制转换等知识，而且能够帮助我们学习汇编语言。本节主要介绍通过 OD 的数据窗口来观察数据宽度。

为了能够直观地观察内存中的数据，我们通过 RadAsm 创建一个没有资源的汇编工程，然后编写一段自己的汇编代码，代码如下：

```
    .386
    .model flat, stdcall
    option casemap:none

include windows.inc
include kernel32.inc
includelib kernel32.lib

    .data
var1    dd  00000012h   ; 16进制
var2    dd  12          ; 10进制
var3    dd  11b         ;  2进制
; 字节
b1      db  11h         ; 16进制
b2      db  22h
b3      db  33h
b4      db  44h
```

```
; 字
w1      dw  5566h       ; 16 进制
w2      dw  7788h
; 双字
d       dd  12345678h   ; 16 进制

    .code
start:
    invoke ExitProcess, 0

    end start
```

在上面的代码中，定义了 10 个全局变量。首先，var1、var2 和 var3 分别定义了 dword 类型的 3 个变量，其中 var1 的值是十六进制的 12h，var2 的值是十进制的 12，var3 的值是 2 进制的 11b。b1 到 b4 四个变量是字节类型的，w1 和 w2 两个变量是字类型的，d 变量是 dword 类型的。

注意：在汇编代码中定义变量，db 表示字节类型，dw 表示字类型，dd 表示双字类型。而在表示数值的时候，以 h 结尾的表示十六进制数，以 b 结尾的表示 2 进制数，结尾处没有修饰符的默认为十进制数。

这 10 个全局变量就是我们要考察的关键。在 RadAsm 中进行编译连接后，直接按下 Ctrl + D 这个快捷键，即可在 RadAsm 安装时自带的 OD 中打开。在 OD 调试器中打开该程序后，观察它的数据窗口（如图 1-5 所示）。

地址	HEX 数据																ASCII
00403000	12	00	00	00	0C	00	00	00	03	00	00	00	11	22	33	44	▮....... ...▮"3D
00403010	66	55	88	77	78	56	34	12	00	00	00	00	00	00	00	00	fU近xV4▮........
00403020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00403030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00403040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00403050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00403060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

图 1-5 数据窗口中查看变量

在图 1-5 中，数据窗口一共有 3 列，分别是地址列、HEX 数据列和 ASCII 列。这 3 个列，可以通过单击鼠标右键来改变现实方式和显示的列数。在地址 00403000 处开始的 4 个字节 12 00 00 00 是十六进制的 12，也就是在汇编代码中定义的 var1；在地址 00403004 处的 4 个字节 0C 00 00 00 是十六进制 0C，也就是在汇编代码中定义的 var2，var2 变量定义的值是十进制的 12，也就是十六进制的 0C；在地址 00403008 处的 4 个字节 03 00 00 00 是十六进制的 03，也就是在汇编代码中定义的 var3，var3 变量定义的值是 2 进制的 11，也就是十六进制的 03。

这 3 个变量在我们定义的时候都是以 dd 进行的，都是 dword 类型的变量，分别各占用 4 字节，因此在内存中，前 3 个变量分别是 12 00 00 00、0C 00 00 00 和 03 00 00 00。

在地址 0040300C 处的值是 11 22 33 44，这 4 个值分别是我们定义 b1、b2、b3 和 b4 4 个字节型的变量，这 4 变量按照内存由低到高的顺序显示分别是 11、22、33、44。

在地址 00403010 处显示的值是 66 55 88 77，这 4 个值分别对应我们定义的 w1 和 w2 两个字型变量，但是我们定义的变量 w1 的值是 5566h，w2 的值是 7788h，在内存中为何显示的是 6655 和 8877 呢？这就是我们提到过的字节顺序的问题。我们的主机采用的是小尾方式存储的数据，也就是数据的低位存放在内存的低地址中，数据的高位存放在内存的高地址中，因此在地址 00403020 中存放的是 5566H 的低位数据 66，在地址 00403021 中存放的是 5566H 的高位数据 55，在内存看时，顺序是相反的。

在地址 00403014 处存放的是 78 56 34 12，这是我们定义的最后一个变量 d，它也是按照小尾方式存储在内存中的。因此，在查看内存时顺序也是反的。

OD 提供了多种查看内存数据的方式，通过在数据窗口中单击鼠标右键，会弹出如图 1-6 所示菜单。

当在数据窗口中选择数据时，右键的菜单提供编辑、赋值、查找、断点功能，如图 1-7 所示。

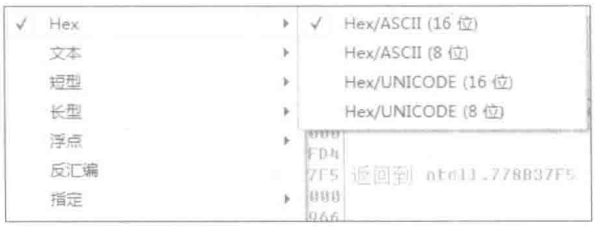

图 1-6　查看数据方式的菜单选项

图 1-7　OD 中对数据操作的菜单

4．通过命令窗口改变数据窗口显示方式

在图 1-4 中的最下方可以看到有一个输入命令的编辑框，在此处可以输入 OD 的相关命令以提高调试的速度。本小节就介绍如果通过命令窗口来改变数据窗口的显示方式。

在上面代码中定义变量时，使用了 db、dw 和 dd 三种类型，在 OD 的命令窗口中也同样可以使用者 3 个命令，其格式分别如表 1-7 所列。

表 1-7　　命令窗口改变数据显示命令格式

命　令	格　式	说　明	举　例
db	db address	按字节的方式查看	db 403000
dw	dw address	按字的方式查看	dw 403000
dd	dd address	按双字的方式查看	dd 403000

将表 1-7 中的命令在命令窗口中进行输入，数据窗口的变化和数值显示的变化分别如图 1-8、图 1-9 和图 1-10 所示。

图 1-8　dd 命令显示的数据窗口

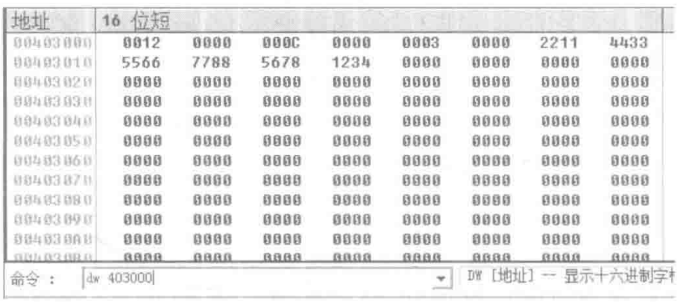

图 1-9　dw 命令显示的数据窗口