21st CENTURY
规划教材

面向21世纪高等院校计算机系列规划教材
COMPUTER COURSES FOR UNDERGRADUATE EDUCATION

新概念汇编语言教程

张增年　杨爱民
铁新城　刘玉英　编著

科学出版社
www.sciencep.com

新概念汇编语言教程

张增年　杨爱民　铁新城　刘玉英　编著

科学出版社

北京

内 容 简 介

本书针对 8086/8088 微处理器详细论述了微型计算机汇编语言程序设计的原理和方法，以简洁的语言风格剖析了汇编语言程序的基本结构，数据表示法，指令集，伪指令，操作符和寻址方式，屏幕处理和算术运算以及编程技巧，各种调试程序的手段等。

该书既是计算机专业汇编语言课程的教科书，也是电子类专业的教科书，同时可作为从事信息技术人员学习汇编语言和微机原理的参考书。

图书在版编目（CIP）数据

新概念汇编语言教程 / 张增年等编著.—北京：科学出版社，2004
ISBN 7-03-014239-X

Ⅰ.新… Ⅱ.张… Ⅲ.汇编语言-程序设计-高等学校-教材 Ⅳ.TP313

中国版本图书馆 CIP 数据核字（2004）第 086019 号

责任编辑：李振格 熊盛新 / 责任校对：柏连海
责任印制：吕春珉 / 封面设计：三函设计

科学出版社 出版
北京东黄城根北街16号
邮政编码:100717
http://www.sciencep.com
源海印刷有限责任公司印刷

科学出版社发行 各地新华书店经销
*
2004 年 8 月第 一 版 开本：787×1092 1/16
2004 年 8 月第一次印刷 印张：16 3/4
印数：1—4 000 字数：376 000

定价：22.00 元

（如有印装质量问题，我社负责调换〈路通〉）

前　言

汇编语言是计算机专业课程中十分重要的一门专业基础课程，是信息工程等专业学生学习微机原理课程前必须学习的基础课程。汇编语言课程有两方面的作用：一是为专业课程的学习准备良好的基础；二是要引导学生掌握学习专业课程的一般方法。本教程从构思到具体写作都和一般教材有很大区别。

我们首先要说明的是实践在汇编这门课中的重要地位。一般来说，在计算机课程中，实践都占据着相当重要的地位。这一结论的依据不全在于所谓“计算机是一个要求具体操作和实际技能的专业”，更在于如果缺乏实践，就绝不可能完整地理解计算机理论。关于这一点的详细阐述显然会使前言过分庞大，但是我们仍要以实际的情况辅以说明。学习汇编的学生大致可分为两类：一种学生自始至终坚持实践，按这种方法去做，尽管对汇编的理解有高低的区分，但至少都可以达到基本合格的程度（只要他化费了足够的努力）；另一种学生不重视乃至根本不实践，这种学生在学习了一段时间之后便再也无法实践，或者会认为“汇编是一门太难的课，无法学懂”，或者满足于“理论理解尚好，实践有待以后加强”之类的说法。需要再次强调的是，学好汇编对于后续专业课有显著的促进作用，而汇编学习的失败将不仅仅是一门课的失败，还会影响几乎全部专业课程的学习。

学习任何课程的正确方法都是循序渐进，学习汇编也不例外，而且与其他课程比起来显得更加重要，其原因主要在于汇编语言的涉及面很广，其中的知识单元不容易分解为相对简单的元素，汇编语言的环境简陋，要求学习者有更好的独力学习（工作）能力和主动精神。为此，本教程在这方面做了特殊的努力。从理论上来说，任何学习都是在原有基础上的一种延伸。如果脱离了原有的基础，延伸无所依托，学习就变为空中楼阁，这样学生所做的只能是死记硬背，不是学习，而是记忆。但是人的记忆力是有限量的，如果不通过正确的手段和方法将记忆的内容转化为可理解的内在能力，那么记忆的超载一方面导致遗忘，一方面导致“消化不良”。不少学生所体验的厌学情绪就和这种“消化不良”有直接的联系。

因此，任何学习都应该依靠原有的基础，而且每次的进展都不能跨度太大，理论上把这样的进展称为转移。在本教程中，转移是决定章、节划分的内在线索，学习者不再被要求死记硬背或者“囫囵吞枣”，每一步的进展都是有线索可循的，进展之后都会有实质能力的获得和提高。

我们希望指明两个结论：（1）学习是一种个体化的行为，尽管各人可以经历相似的学习阶段，使用相似的学习方法，但各人的历程有早有晚，时间长短不一，强度也因人而异，结局也不尽相同，因此期望通过模仿别人而达到学习目标基本上是不可能的；学习只能依靠自身的努力，达到自我能力的提高。（2）学习无所谓“苦”，无所谓“乐”。

如果有人能设计出一种方法，让人保持在“乐”的情绪中学习，我们并不反对，但就我们所知，真正的学习状态是“专注”。在此“专注”之下，任何“苦”或“乐”都没有地位，也起不到帮助或者阻碍的作用。

由此可见，我们自然就可以讨论学习目标的问题了。一般认为，学习目标有两个：知识的获得和能力的提高。但是深究起来，就可以发现上述说法很有疑问，似乎学习者可以单独地追求任一目标而舍弃（至少是不关心）另一目标。实际上，一个人不可能简单地获取知识而不提高能力（那样他只是死记硬背，更像一个计算机的磁盘。另外，死记硬背的东西也不能被称为知识，因为知识要求理解和应用），同时提高能力不可能凭空获得，其必要过程就是获取知识。两者间的联系不仅必然，而且显然，即学习者的根本目标是提高能力，其间知识的获取是一必要步骤；如果知识的获取不充分，能力的提高必然不完全。

更进一步，能力可以分为两类：在一般性能力之外还有一种，可称为重构性能力。尽管一般性能力更容易被观察和检测，但重构性能力是更基本的。正是重构性能力与学习一一对应，如果没有重构，学习就变得几乎不可能。在本教程中，更加注重的就是对于重构性能力的培养和训练。一般性能力可能由于与记忆力相混淆而导致误测，而重构性能力则不太可能被混淆。具体来说，我们遵循“模仿中有创新”的原则，要求学习者在学习过程中通过创新活动而达到目标。

有关本教程的写法有必要在此做一说明，因为与众不同之处并不在标新立异，学习者使用本教程的方法也应该有所不同，如：

（1）路口和检测点。路口和检测点是本教程的最大与众不同之处。在路口处，学习者可以根据自己的情况（爱好）选择阅读教程中的不同部分，当然也可以把不同分叉的内容都读一遍。路口的意图在于向学习者指出不同的思路，如果它能帮助学习者开阔眼界，养成容纳发散性思维的习惯，那么其任务就基本完成了。检测点是本教程构思的核心之一。为了防止“消化不良”，急于求成，并且督促学习者实践，学习者完成检测点所交给的任务是十分必要的。对于不甚仔细认真的学习者它可以起到帮助检查的作用，对于学习速度偏快的学习者可以起到稳定节奏的作用。

（2）注和 CAI。为了行文流畅以及让学习者把注意力集中在关键的问题上，凡属于细枝末节，或可能超出课程范围的内容都放在注之中（注的内容一般都属于课程之内，但也有超出课程的）。学习者在刚接触这一风格时可能不太适应，而且频繁地翻阅注解也确实不方便，为此我们设计了与本教程配套的 CAI 软件；在该软件中，查阅注解是相当方便的。

（3）To the teacher 和 To the student。这两个写法源于 L.G.Alexander 的《新概念英语》，正是该书在某些方面给予本书作者相当的启发。作者也曾考虑过不采用英文，但因为未能找到一个既清晰简明又能准确表达原意的译法而作罢。

（4）其他参考书。原则上，鼓励学习者去使用任何一本合理的参考书，但作者不希望学习者在较早阶段为此化费太多时间，原因很简单：如果学习者在其他参考书上发现只能用死记硬背来学习，那么他在汇编这门课程中改掉坏习惯就要比学会正确的好习惯更困难，更耗费精力和时间。

本书共 18 章。第 1 章介绍汇编语言中数的表示及 PC 机的体系结构；第 2 章介绍 Debug 工具的使用；第 3 章介绍了汇编程序的格式、上机调试方法；第 4、5 章介绍汇编语言中数据的定义与程序的组织结构；第 6、7 章介绍利用汇编语言编写屏幕显示处理的各种方法；第 8 章介绍字符串处理的五条指令；第 9、10 章介绍汇编语言中的算术运算；第 11 章介绍汇编语言中对多重循环的处理；第 12 章介绍用汇编语言实现在屏幕上作图方法；第 13 章介绍有关打印机的操作；第 14 章介绍对表的查询与排序；第 15 章介绍汇编语言中的伪操作指令；第 16 章介绍磁盘与 DOS 的文件系统以及相应的编程方法；第 17 章介绍汇编语言中的参数传递及汇编语言与高级语言之间相互调用时的参数传递；第 18 章介绍常用的 BIOS 中断及端口编程。书中提供了许多注释、附图、程序示例及思考题，每章的后面均有习题，以方便读者复习，巩固学习效果。

本书的第 1、2、3、4、8、13、14 章由张增年编写，第 5、6、7、9、10、12、15、16、17 章由铁新城编写，第 11、18 章由杨爱民编写，刘玉英编写、整理了附录部分，并参加了部分文稿的录入与校对工作。铁新城负责全书的组织与定稿。

作者首先要感谢浙江万里学院将本书列入浙江万里学院教材建设基金资助计划，感谢湛江师范学院领导给予的关心和鼓励。作者还要感谢安嘉翔、乔立恭教授。因为构成支撑本书的理论中有一部分来源于他们近年来的一系列研究成果。

限于时间仓促，差错遗漏在所难免，恳请读者指正。如果需要与本书配套的 CAI 软件，可与作者联系，发 E-mail 至 zzn@zwu.edu.cn 中即可。

目　录

第1章 绪 论

本章主要从汇编语言这一层上来介绍计算机软件（数的表示）和计算机硬件（目前仅限于CPU和RAM以及一点儿有关总线的内容）。在任何时候，从汇编的角度出发的看法总是十分重要的。与此相应，一方面学习越来越多的指令和编程技巧，一方面越来越深入地理解计算机体系结构，把握住这两点并且解决好它们之间的相互关系既是汇编教学中的困难之处，也是汇编教学成功的正确途径。

在开始学习本章前，希望你能够了解有关汇编的一些情况，它可以帮助你决定是否学习它。

汇编的英文缩写是MASM（见注释[1]），对于程序员（见注释[2]）来说，MASM是必需的工具之一（见注释[3]）。在早期，汇编几乎是程序员开发所用的唯一工具。在汇编语言之前，程序员使用机器码（机器语言）编程，可以理解为直接和计算机（硬件）打交道。汇编语言的产生允许程序员更多地从软件的角度出发来设计、分析和开发系统。之后，汇编语言作为工具一直在不停地发展，其功能也不断增多。

作为学习方法上的指导，有必要指出：在汇编学习过程中，实践有着极其重要的作用；简单的实践可以是一道习题的演算，对于示例的某种验证，更多的是探寻对于某个问题的解答，在已有的基础上做出有创新意义的新构思并将其付诸实践。

1.1 数在计算机中的表示

在计算机内部，所有的操作都是数字的操作，而这些数字的表示、存储和运算使用的都是二进制(见注释[4])，我们将从二进制开始介绍计算机中使用的数值。

【二进制】

二进制采用0、1表示数，逢2进位，因此

$2=2^1$ 表示为 $(10)_2$　　$4=2^2$ 表示为 $(100)_2$

以此类推。

计算二进制数的大小可以按上述规则进行。例如

$(11111111)_2=2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0=255$

$(1111111111111111)_2=2^{15}+2^{14}+\cdots\cdots+2^1+2^0=65535$

（关于二进制的数学上的描叙述，可参看注释(见注释[5])

【Bit和Byte】

在计算机中，最基本的数据单元是Bit（比特），它只有1位，可表示0或1。将8个Bit组合在一起称为Byte（字节）。在汇编中，字节是最常用到的数据单位。计算机芯片大

多是以 8 个 Bit 为基础设计的。在一个 Byte 中，各个 Bit 的编号如图 1.1 所示。

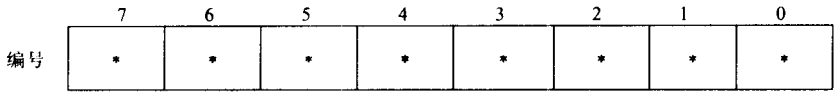

图 1.1 在 1 个 Byte 中各 Bit 的编号

一个 Byte 所表示的数值大小有限，为了使用的方便，也可将 2 个 Byte 合在一起。此时的数据单元长度有 16 个 Bit，称为 Word（字）（其中各 Bit 从 0～15 编号）。计算机中的内存是以 Byte 为单位计算的。显然，Byte 越多，越有“记忆”能力。习惯上，内存更多地以 K 为计算单位，1K=1024 Byte，比 K 更大的单位是 M（兆），1M =1K×K= 1024×1024（Byte）≈ 1 百万（字节）。比如内存为 640K 表示有 640×1024 Byte，而 1 个 40M 的磁盘可以存储 40×1024×1024 Byte 的信息。

如前所述，在微机上，数据单位是 Byte（即 8 个 Bit），一个 Byte 可以表示 0～255 共 256 个数。在实际应用中，也可以把这 256 个数看作 256 个符号（即每一个数字对应于一个符号）。由于 256 这个数字十分有限，因此有必要选取人们最为常用的一组符号，同时为了避免混乱，符号和数字的对应应该有统一的规定。目前通用的是 ASCII 码(见注释[6])，在 ASCII 码中字母‘A’～‘Z’对应于 65～90，数字‘0’～‘9’对应于 48～57(见注释[7])，等等。

【二进制运算】

二进制加法运算的规则如下：

0 + 0 = 0　　0 + 1 = 1　　1 + 0 = 1　　1 + 1 = 10

以 Byte 的加法为例

```
    00111100                    60
+） 00110101                   +53
  ————————      （翻译为十进制）————
    01110001                   113
```

再看下一个例子

```
    11111111                   255
+） 00000001                 +   1
  ————————          （或）     ———
（1）00000000                    0
```

【路口】

在以 Byte 为基础设计的计算机中，上述加法的进位丢失(见注释[8])，结果为 0。如何解决这一困难呢？下面通过两种方式来解答这一问题。

【解答 1】

再说明一下问题的来由：原本二进制（或者其他进制）不存在位数的限制，每次进位实质上是再加一位有效数字。但计算机的位数是有限的，于是就有了上述悖论。

让我们考虑把二进制表示看成一种符号体系，该体系所表示的数应当符合标准的加法规则。仍以以上两个算式为例：对于“00111100”+“00110101”=“01110001”其中各个数所表示的数符合加法规则。那么对于“11111111”+“00000001”=“00000000”，其中“00000000”表示0，“00000001”表示1，于是有“11111111”+1=0，或“11111111”=0-（+1）=-1。

十分有趣，原本我们以为1个Byte表示范围为0～255，但现在发现，为了使其满足加法运算的规则，这256个符号应划分为127个正数，1个0和128个负数。读者可以根据-1的得来自己找出-n的一般表示。

【解答2】

由于一个Byte只有256的容量，因此其可以表示的数只有256个，当两个数相加之和大于或等于256时，就会超出界限（溢出），从而使结果无效。保证其结果的有效性可以有两种方法：

（1）在加法时检查是否溢出。

（2）引入负数的概念。为此，规定Byte最左边的位为符号位，为0时表示正数，为1时表示负数。于是11111111+00000001=0，表示的是：-1+1=0（注意：-1不是10000001），负数表示是由和正数相运算的规则而来的。比如00001010为+10（十进制），而00001010+11110110=00000000，因此-10（十进制）表示为11110110。

【路口归一处】

至此，我们介绍了二进制的加法，二进制的减法可以由加法直接得来，因为

$$M-N=M+(-N)$$

二进制的乘法规则也相当简单，具体为

$$0\times 0=0 \qquad 0\times 1=0 \qquad 1\times 0=0 \qquad 1\times 1=1$$

有关二进制以及二进制在计算机中的运算更多的细节将在第10章中介绍。

【检测点】

回答以下问题：

（1）15，27，100的二进制表示是什么？

（2）-15，-27，-100的二进制表示是什么？你是如何得到结果的？

（3）（思考题）（见注释[9]）在以Byte为基础的运算中，何时会发生结果无效的情况？为什么？

（4）“A”、“0”的ASCII码是多少？请表示为二进制。

（5）（思考题）“0”+“0”=48+48=96 ≠“0”这个结果该如何解释？

【小结】

在以Byte为基础的二进制运算中，存在两种可能的约定：一种约定由1个Byte表示0～255共256个非负整数，另一种约定由1个Byte表示-128～+127之间的整数，此时，负数的二进制表示可以由正数的二进制而得到，方法是：-N=N的反码+1。所谓反码是指把原来的码取反（0→1，1→0）。比如-5=+5 的反码+1=00000101 的反码+1=11111010+1=

11111011。这种不采用正负号而采用符号位（即最高位为 1 表示负数，0 表示正数）的数字表示系统又被称为“补码”方式（采用“补码”的最大好处是正、负数格式和长度一致，且可以直接放在一起运算）。作为一个特例，-0 = + 0 的反码+1=11111111+1=0。

【十六进制】

由于计算机内部采用二进制，因此对于了解计算机内部的工作情况来说，熟悉二进制是必不可少的。但二进制由于位数太长而很不方便，十六进制将 4 位二进制数合成一位，减小了数据长度，成为汇编中最为常用的数据格式（见注释[10]）。

二进制和十六进制的对应关系如下：

0000=0　　0001=1　　0010=2　　……　　1001=9

但是 1010=?

在人类通用的阿拉伯数字中，只有 10 个符号用于表示十进制中的 10 个数，而十六进制需要 16 个符号，为此，我们采用字母来代替不足的部分：

1010 = A = （10）$_{10}$　　1011 = B = （11）$_{10}$　　1100 = C = （12）$_{10}$

1101 = D = （13）$_{10}$　　1110 = E = （14）$_{10}$　　1111 = F = （15）$_{10}$

十六进制和二进制之间的转换十分方便，比如：

B9 = 10111001　　（因为 B = 1011，9 =1001）

十六进制转换成十进制和二进制到十进制的转换相似，还是以上面的数字为例：

$B9 = B \times 16^1 + 9 \times 16^0 = 11 \times 16 + 9 \times 1 = 176 + 9 = 185$

【检测点】

（1）十六进制 AA，EE 相当于十进制数多少？

（2）将 255，150 转换为十六进制。

（3）你能总结出十进制数转换为十六进制数的规则吗（请独立思考、相互讨论之后再看（见注释[11]）？

各种进制可以同时在汇编中使用，为了区分，在各个进制后用不同的字母加以标识，B（Binary）表示二进制，D（Decimal）表示十进制，H（Hexadecimal）表示十六进制，Q 表示八进制，由于十进制是最常用的，D 也可以省略。比如：20H=00100000B=32D=32。

1.2 PC 机的体系结构

本节简要介绍 PC 机的组成，主要介绍在汇编中最为频繁地用到的 CPU 中的寄存器和内存的情况。

【PC 机（见注释[12]）的硬件组成】

PC 机系列（包括 PC，XT，AT（见注释[13]））的体系结构相比于其他系列是相对简单的。其硬件组成为系统部件、键盘和可选部件。其中系统部件包括系统板、电源和扩展槽。可选件是通过扩展槽和系统板发生关联的。有些可选件直接插在扩展槽上，比如通讯口；另外一些通过适配卡而起作用，比如显示器通过插在扩展槽上的监示器适配卡和主机通讯。

可选件的种类繁多，除上述通讯口和显示器口外，像磁盘驱动器、打印机和游戏杆等都属于可选件。

系统板是计算机最主要的组成部份，主要由CPU、ROM、RAM和总线组成。其中CPU（见注释[14]）是计算机的“心脏”，CPU发布指令并执行运算；ROM（见注释[15]）也称只读存储器，主要的任务是帮助系统启动；RAM（见注释[16]）又称内存，是程序员所使用的主要资源之一，在开机之后，RAM既可读，又可写。总线是联系CPU和内存扩展槽的部件，所有的数据流动都要通过总线进行（见注释[17]）。

【检测点】

（1）说明一个简单的PC机的大致组成。

（2）说明ROM和RAM的不同之处。

（3）硬盘属于可选件，而软盘驱动器则属于系统部件。这个观点正确吗？

【CPU的组成和功能】

以8088（见注释[18]）为例，CPU由两部分组成：执行部件（EU）和总线接口部件（BIU）如图1.2所示。

其中执行部件又分为运算（算术运算和逻辑运算）部件、控制部件和寄存器；总线接口部件又分为总线控制器、指令队列和段寄存器。

在8088的构成中，特别值得一提的是指令队列的工作方式。8088所执行的指令来自于内存，每次都由BIU通过总线将内存中的指令取到指令队列中，交由EU执行。实际上BIU总是提前取指令，在EU执行时，BIU并行地又去取下一条指令，从而提高了处理速度。

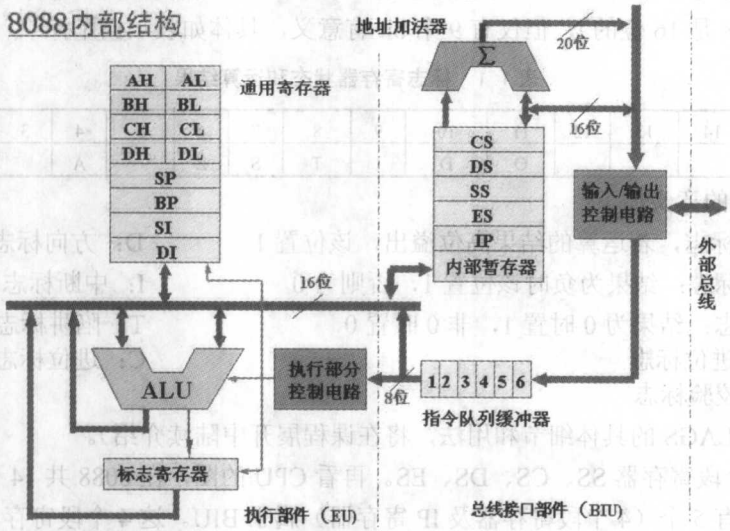

图1.2 8088内部结构

【寄存器】

CPU中的主要部件是寄存器，因为决定内存地址和进行一般运算的均由寄存器完成。

8088 有 14 个寄存器，其用途各不相同。

（1）4 个通用寄存器：AX、BX、CX、DX。一般的计算都在这 4 个寄存器中进行。8088 采用 16 位结构，其中所有的寄存器都为 16 位，换句话说，其可表示的数和运算的范围为 0～65535 或-32768～+32767（见注释[19]）。同时，这四个寄存器也可以分为两半来使用，每一半为 8 位，如图 1.3 所示。

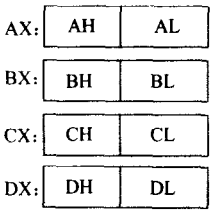

图 1.3 4 个通用寄存器

尽管这 4 个寄存器均可用于计算，但在用法上还是有所区别，在以后将循续渐进地予以介绍。

（2）2 个变址寄存器：SI、DI。其中 SI 称为“源变址”（source index），DI 称为“目的变址”（destination index）。它们主要用于连续数据的处理。

（3）2 个指针寄存器：SP、BP。通过这 2 个寄存器，程序员可以灵活地访问堆栈（见注释[20]）中的数据。

（4）指令指针寄存器 IP。用于指示程序执行到何处。在程序设计中一般不会使用它（它由系统自动管理），下一章将作具体介绍。

（5）标志寄存器 FLAGS。标志寄存器记录 CPU 的当前状态和运算结果，共有 16 个 bit（因为 8088 是 16 位的），但仅有 9 个 bit 有意义，具体如表 1.1 所示。

表 1.1 标志寄存器状态和运算结果

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
标志					O	D	I	T	S	Z		A		P		C

各个标志的意义如下：

O：溢出标志，若运算的结果高位溢出，该位置 1

S：符号标志：结果为负时该位置 1，否则为 0

Z：零标志：结果为 0 时置 1，非 0 时置 0

A：辅助进位标志

P：奇偶校验标志

D：方向标志

I：中断标志

T：陷阱标志

C：进位标志

（有关 FLAGS 的具体细节和用法，将在课程展开中陆续介绍）。

（6）4 个段寄存器 SS、CS、DS、ES。再看 CPU 的图，在 8088 共 14 个寄存器中有 9 个属于 EU，有 5 个（4 个段寄存器及 IP 寄存器）属于 BIU。这 4 个段寄存器不参与运算，只和数据及代码在内存中的地址分配有关，为了理解它们，我们先去看一看内存的情况。

【RAM 的寻址方式】

在一台典型的 8088 微机上，内存为 1M=1K×K=1024×1024 byte 分别编号为 0、1、2……直到（1M-1）（见注释[21]）。因此，只要知道数据所在的单元，就可以对其访问（读或写）。

让我们仔细地考查从内存到CPU的数据转移是如何做到的。

【解答1】

以1M内存为例，1M=1K×K=$2^{10}\times 2^{10}=2^{20}$。因此对其编号需要20位2进制数。由于这样大的一个数超出了8088的结构（8088为16位的，其全部寄存器都是16位的）。因此，仅通过单一寄存器无法为内存编号，也就不能正确地寻址。8088的设计者采用的方法是用两个寄存器联合起来以完成对于1M内存的寻址。不过这两个寄存器不是简单重叠（仍然只有16位），也不是简单拼接（这样将有32位，对于8086来说太大了），而是有一个寄存器向左偏移4位。这样，8088用于寻址的位数是16+4=20，正好对应于1M内存。偏移4位的寄存器就是段寄存器，SS、CS、DS、ES分别用于栈段、代码段和数据段附加数据段。

【解答2】

对于内存的典型访问方法是先将其地址（数据所在单元的编号）放入某寄存器，再通过该寄存器的值去作访问。然而，8088的寄存器只有16位，因而其可访问的范围只能在$2^{16}=2^{6}\times 2^{10}=64K$之内，远远小于1M。请你思考一下再看下一段。

使用2个寄存器。打个比方：居民住宅往往先分为单元，再在单元内分号。同样内存也可以先分为段，在段内再编号。段的地址即由段寄存器来决定，总共4个段寄存器SS、CS、DS、ES分别用于栈段、代码段和数据段附加段。具体来说，2个寄存器可以组成共32位的长度，可以对$2^{32}=2^{2}\times 2^{10}\times 2^{10}\times 2^{10}=4096M$的内存寻址，但实际上，8088不需要这么长。因此，将段寄存器重叠12位，只偏移出4位，组成16+4=20位的长度刚好实现1M内存的寻址。

【检测点】

了解内存的寻址以及内存和CPU的关系对于汇编程序员是入门的第一步，因此，务必认真地解答以下各题。

（1）判断下列说法是否正确：

① 寄存器的作用是存储数据。

② 所有寄存器都可以作运算。

③ 程序员只操作寄存器。

④ 内存可以作运算，只是速度比较慢。

⑤ 8088可以访问2M以内的内存。

⑥ 内存的最低地址是1，最高是1024×1024。

（2）考虑某个数据在内存64K和640K的地方。用2个寄存器DS和BX描述这一地址，其中DS为段寄存器。找出至少1个解答（见注释[22]）。

【内存的结构】

内存由RAM和ROM共同组成，一般ROM占据最高端，在RAM的最低端是DOS，中间留给用户，用户一段段地使用RAM，每段最大不超过64K。