8087应用和程序设计

上海市科协科技咨询服务中心情报中心

615研究所航空电子技术编辑部

编 者 的 话

《8087应用和程序设计》是一本专门介绍8087芯片的应用和程序设计的著作，是特为要了解、熟悉、学习和使用8087，特别是要在IBM PC上使用8087的人员编写的，对于其他使用8087的用户来说，也是一本很好的使用指南。

该书共十五章，分成三个部分：

第一部分（1—4章）在不十分涉及技术的水平下，讨论8087的工作能力。

第二部分（5—8章）对8087的指令进行深入的说明，同时介绍8087汇编语言程序设计的基本知识。并注意汇编语言程序与BASIC程序的连接。

第三部分（9—15章）则主要侧重于应用。在这一部分中开发了许多有用的8087汇编语言程序，你可以通过这些程序进一步学习8087的程序设计技巧，也可以直接使用这些程序。

该书在详细介绍8087的基础上以三分之一的篇幅介绍8087的应用，很多子程序不仅有8087的汇编语言程序，还有对应地用BASIC编写的程序，读者可以相互比较，加深对8087性能及程序设计的了解。更为可贵的是书中出现的几十个8087的子程序及其他应用程序，是作者在IBM PC上开发和运行的，并已装入软盘。

本书约30万字。根据1984年英文原版译成中文。由荀世斌、张 力译，夏华龙、刘道珍初校，并由顾诵淦、吕宗琪、顾良士、沈天伟、谢鸿飞复校。敬请批评指教。

目　　录

前　言

首先要说明一下我对8087发生浓厚兴趣的原因。多年来，我一直借助大型计算机开展科研工作。但当我购买了个人计算机以后，就发觉，许多工作在个人计算机上做比在大型机上要方便得多。不过，我很快又发现个人计算机用于处理大量数据计算时速度还不够快。

现在，有了8087，我就可用个人计算机解决许多大型研究课题。虽然有些问题非得使用大型计算机（而且总是需要的）才能解决，但是，个人计算的范围已扩大了十倍。

8087不仅速度快，而且使用非常方便。无论读者是“程序使用人员”或是“程序编写人员”，都会发现8087是一种超群的器件。我希望您会感到《8087应用与程序设计》一书，不仅是一本程序设计基础教材，而且是一本饶有趣味的读物。

本书阅读对象

- 想要了解8087用途的人（主要参阅第一部分1—4章的内容）。
- 希望学习8087编程方法的人（主要参阅第二部分5—8章和第三部分12章的内容）。
- 希望在其个人计算机上配备数值计算应用程序的人（主要参阅第三部分9—15章的内容）。

第一部分讨论8087的功能，但很少涉及技术性问题。如果你考虑买8087并想了解与8087兼容的硬件和软件，就可以阅读第一部分的内容。

第二和第三部分适合对技术问题比较感兴趣的读者。虽然本书从非常基础的讲起，但读者原来一些编程经验对学习这两部分是有帮助的。尽管你不必是一位计算机专家，但本书毕竟不是计算机入门课本。

第二部分（5—8章）详细讨论8087的指令，同时，还介绍8087汇编语言编程的一些基本知识。着重说明将汇编语言与 BASIC 程序的连接问题，其中包括把汇编语言程序同解释 BASIC 、编译 BASIC 语言程序连接起来的，极为详细的交互会话方式。

第三部分集中讨论8087的应用。我们在这一部分开发了许多有用的8087汇编语言程序。读者可以这些程序为例，进一步学习8087编程，也可以直接应用一览表中的那些程序。（第三部分第12章还说明了8087中一些极为先进的指令）

怎样阅读本书

你可以根据需要阅读有关章节，而不必从头至尾阅读。

多数读者很可能会发现，第一部分内容是资料性的，并且易于阅读。如果你想编写8087汇编语言程序，就需要钻研第二部分。（如果你是一位有8087编程经验的编程人员，你可以跳过第二部分，直接阅读第三部分。）如果你对应用感兴趣，而不关心内部的编程细节，则可阅读第三部分，假如你需要核对某些内容，则可以回过来阅读第二部分。

最后，你可以使用一览表中的程序。如果你仅想很快地求得答案，就不必了解为什么一个程序要以某种方式编写或其内部怎样操作，只管使用这个程序就行了。如果某一个程序对你很有用，你想修改它或自己编写一个类似的程序，你可再回过头来弄清楚某个程序的编程方式及其内部操作原理。

书中有几章，其开头是一个简介，然后是下面的记号：

一　　览　　表

可以看到，在这个记号下面有该章中出现的程序清单，同时，还有关于程序用途、所需输入、输出的简单说明。如果你想很快找出某个程序，就可利用这个一览表。问题不同，则解决问题的方法也不同。如果你只想运行程序，而不准备编写程序，那么，可以不读“方式与原理”这部分内容。只要浏览一下那些要求你向程序提供信息的内容就可以了。

重要的数字捣弄技术

本书除了详细讨论8087和数值编程之外，还介绍了解决数字捣弄的重要方法，讨论的依据是程序设计的两个准则：

- 10%的程序代码占用90%的执行时间。
- 不论使用什么编程语言，编写一个工作程序，其成本总是与程序长度的平方成正比。

编程人员有时会错误地为了编写一个“高效率的程序”而大费气力，一个绝妙的对策则是找出90%计算工作量的那部分10%程序的代码，然后，重写这段程序代码，以使达到最高效率，而编写其余90%的代码时，则要求尽量清晰。

为了提高程序的效率，常常要用汇编语言编写程序，由于对同一程序来说，其汇编语言代码的长度可能是其 BASIC 语言代码长度的10倍。所以汇编语言程序的调试要比 BASIC 难一百倍。整个程序都用汇编语言编写是没有意义的。但将关键性的程序用汇编语言编写确实很重要，这样，只需编写很少一部分汇编语言程序，就可以充分利用汇编语言运行速度快的优点。

考虑到许多数值编程问题都要使用同样的基本子程序，用汇编语言就更好了。实现数组相加的8087汇编语言程序比 BASIC 中的 FOR／NEXT 循环要复杂一些。但是这个8087程序只需要编写和调试一次，此后，每当求数组和时，调用该程序就行了，重复使用这个子程序比每次都写一个 FOR／NEXT 循环方便一些。计算机专业人员称这种重复使用子程序的设计方法为“模块化程序设计”。作为一个方便且功能很强的8087子程序模块的例子，可参看第14章的统计软件包。

实际上你可以作得更好。本书中（以及可选用的磁盘上）有许多用于数值计算的8087程序。虽然我们希望你下决心学习有关8087的所有性能，并编写自己的专用子程序，但也欢迎你从书中找出全部子程序并将它们用在你的装有8087的个人计算机上。

所需的硬件和软件

书中的程序可以在以Intel 8087数字数据处理器，以及Intel 8088和 8086微处理器系列为基础的计算机上运行，该系列除了8088和8086以外，还包括188，186和286微处理器及相应的8087处理器。所有的程序都在IBM PC上开发并经过检验。所有的计时都假定处理器时钟为5MHz的8088，且计时是近似值（例如：IBM PC运行时的计时值比实际规定的时间慢5%左右，以8086为基础的机器速度则快一些）。除非另有说明，所给的BASIC程序的计时都针对没有8087的解释的 BASIC。

8087汇编语言程序都可以作为子程序由 IBM PC上的Microsoft解释 BASIC或编译 BASIC程序调用。这些程序都假定数据的存贮格式与8087兼容（有关与8087兼容的软件的进一步讨论见第13章）。这些程序也可以与8087出现之前的 BASIC 一起运行。但你需

增加附录中的Microsoft到Intel的数据格式转换程序。

为了汇编书中的程序，你需要有一个能识别Intel全部助记符指令的汇编程序。（注意，IBM PC MACRO 汇编程序1.0版本虽然能生成8087指令，但却不能识别8087助记符指令。如果你乐意将8087助记符重新编码为8088的ESC指令，则仍可使用这个版本的汇编程序。在可选的软磁盘里，这些助记符重新编码，所以你可以使用 IBM 的 Macro Assembler。）由于 BASIC 是主要的个人语言，所以我们已经编写了要由 BASIC 调用，而不是由FORTRAN或其他在大型机上更通用的语言调用的全部程序。如果你希望将这些程序与其内部约定和 BASIC 不同的另一种语言相组合，你必须重新编写一些指令。在IBM PC上，所有这些程序可以在PC—DOS1.1版本操作系统的Microsoft BASIC下工作。如果要使用其它的计算机或不同的软件，某些细节可能有所不同。

第一章　将运行速度从分级提高到秒级

装有8087的个人计算机(PC)有三个优点：使用方便，计算精确和运行速度快。读者只要熟悉了本书的内容，就可用8087解决实际的计算问题。本章阐述8087及其广泛的应用。

本书自始至终试图以科学分析的方法理解8087。只要有可能，我们就讨论一般原理，诸如“为什么这样编程”，但尽量少涉及计算机工作的技术细节。为了使讨论具体起见，每种原理举一个应用实例加以说明，8087的功能虽然很强，但使用仍很方便。我们希望本书不仅能开阔你的思路，而且读起来饶有趣味。

怎样提供方便？

8087的设计着重突出使用方便，使其尽量接近笔算方法。你作为8087的一个用户，初次使用它也很容易，只要将8087插到个人计算机上，象通常那样运行程序就可以了（你必须有配合8087使用的BASIC或其它软件版本）。你不需再花气力，就可以期望程序的执行速度提高20%到10倍。

如果你要充分利用8087的硬件功能，就需要有专为8087设计的软件。关于为8087购买软件的注意事项将在第三章中详细讨论。这里仅给予综述。

8087兼容软件中最重要的内容实际上与硬件有关。8087能扩充主处理器的能力同时又不影响主处理器正常操作，因此，在“8087出现之前”设计的任一软件，在加入8087后应当仍能正常运行。

这种百分之百的“向上兼容”好处很大，但总有一弊。在系统中加入8087后，使用8087出现以前编制的那些程序根本不能提高速度。例如，你一、二分钟就可以在IBMPC中插上一片8087（为了编写这本书中的程序，作者在IBMPC上加装8087用了一分钟），你所有的BASIC解释程序或编译程序都会正确运行，但不会提高运行速度。所以当我们讲到加上8087可以提高运算速度这一优点时，也暗示要使用8087兼容软件。

（你只要稍微进行一点重新编程工作，就可使用8087运行“8087出现以前”的那些BASIC版本和其它软件。这个问题将在第三章讨论。为使你放心，可以告诉你，这本书中引用的所有程序都是用“8087出现以前”的软件编写的）。

你购买使用8087的软件时，应注意到一个潜在的危险，就是在同时运行使用8087的软件和8087出现以前的软件时，可能（尽管可能性不大）会带来一些麻烦。详细讨论见第三章。

假定你拥有使用8087的BASIC或其他程序设计语言的版本，就可以运行你平时所有的程序。与没有使用8087的性能相比，进行大量数值计算的程序速度将大大提高。如果你确实要用计算机进行大量的数值处理。总得使用一个专门编制的高速8087子程序库。

本书第三部分（9—15章）有数值计算用的最重要的子程序。你只要了解每个子程序的用法说明，将子程序输入计算机，并将它们与BASIC程序相连接就行了。（与这本书配套供应的磁盘上已存入这些子程序并汇编好）与用8087以前的BASIC程序相比，使用这些子程序能提高执行速度10到200倍（在极少情况下，速度可提高500倍）。

本书第二部份（5—8章）介绍8087最高级的应用：用户自己怎样编写子程序。由本书实例可以看出，由于8087设计得精巧，用汇编语言对8087编程就比较简单。在阅读了这些例子及所用的指令后，你在编写专用程序时就不会有什么困难了。

怎样保证精确？

如果你不在乎要得到精确的答案，那末设计一个编写容易、执行迅速的程序就不需要什么技巧。8087最重要的特性就是它的计算精度非常高，8087有三个改善精度的特点：

- 内部计算产生的精度位数比BASIC双精度数还多11位—这相当于3位十进制数字。
- 内部计算的数值范围极宽，8087能表示的数大到10^{4932}，小到10^{-4932}。因此在计算过程中极少发生上溢或下溢。事实上，数字精度和数值范围比以往大部分主计算机上的要大得多。
- 8087的设计使其能处理许多错误情况，并自动进行适当的恢复，因此，用简单的“笔算”算法就很可能正常工作。出现故障时，8087则按照恰当的工作原则进行处理，而不是输出错误的结果。

怎样提高速度？

装上了8087的PC究竟有多快？最好将它与标准的中型计算机或没有装上8087的微型机比较一下。对8087最引人注目的评价也许在将它的速度与价值几十万美元的中型计算机相比之后就十分清楚了。虽然8087比一台价值50万美元的机器慢几倍，但它的价格却不止便宜几倍。

进行精确的比较总是很难的。但一些数字可以使你对8087的速度有个印象。中速的主计算机执行一次乘法运算约需1到5微秒，而一台超级小型机需要1微秒，一台5美元的台式小型机约需3微秒。高效率的8088软件执行一次乘法需要400微秒（双精度运算则需900微秒）。但在PC机上增加一块廉价的8087后，完成同样的任务只需要20到30微秒。

首先，微型机是用来执行原来要由大型机完成的大量数值计算的一种经济有效的设备。PC机用了8087之后，其速度为大型机的1／4到1／20，但成本只有大型机的1／10到1／160。虽然对于某些任务，大型机总比微型机更为经济有效，但配有8087的个人计算机是第一个与相应的大型机在经济上开展竞争的微机。

大多数拥有PC的人，所关心的与其说是个人计算机与大型中型计算机进行比较，毋宁说是怎样使用8087来提高个人计算机的速度。PC加上8087之后能否提高速度，这取决于具体用途和使用8087的方式。（看完本书后，你就会知道充分利用配有8087的PC的方法）其关键是要懂得8087是一种数值数据处理器。8087只能提高数值计算程序的速度。如果只使用PC进行字处理，则99%的工作与8087无关。如果只是处理偶然出现的数值，为此增加8087就小题大做了。

使用8087能否加速运算，这在很大程度上取决于使用8087的方式。但总的来说，8087已将运算速度的量级从分级提高到秒级。

特定的速度比较

你究竟能从8087获得多大好处，这不仅取决于8087的硬件速度，也取决于所用的软件。速度问题将在第四章进行深入讨论，这里只初步讨论一下。

8087能提高速度到秒级，这要视软件花在“内务”操作上的时间与花在数值计算上的

时间之比而定。8087能加速数值计算，但对内务操作上所花的时间就无能为力了。表1—1表示当你借助8087进行一个“内务操作”少，速度高的程序时可能得到的各种结果。

表1—1　用标准测试程序比较BASIC和8087的速度（时间单位：秒）

程　　序	50 × 50 矩阵乘法	5000 平方根
BASIC	1200	52
8087程序	8	0.35

表1—1中8087以前的BASIC与8087专用程序进行的时间比较，这个程序可在本书后面找到。这种改善是8087和优良软件配合使用的典型结果。根据不同的应用，8087硬件可提高速度10到50倍左右，其余的改进则归功于软件中内务操作较少。如果使用8087时运行一个内务操作较多的软件，你基本上看不出这种改善。（计算机中嵌入的BASIC解释程序必定是一个内务操作较多的软件）。由于8087只能加速数值计算，这类软件中花费在数值计算上的时间较少，用于内务操作和数值计算的时间之和几乎不会低于上表给出的数值。尽管如此，改进还是很大的。

使用8087需要做些什么准备？

当然，你需要8087。你的个人计算机既可能已配备了8087，也可能需要在现有的机器上插上8087。你可以在任何一台以Intel 8088或8086系列为基础的PC上增加8087。增加8087的困难程度取决于制造厂家在设计计算机时是否为8087提供空插座。即使没有提供空插座，仍旧可以增加8087。然而，要做到这一点需要一些技术知识。

庆幸的是，许多厂家确实为8087提供了空插座，特别是IBM（以及制造兼容PC的公司），在推出第一台PC时，就专门在主电路板上为8087留出一个空插座。为了增加8087，只需将8087插入这个空插座即可。插上8087十分方便（安装8087，根本不需要别人帮助，）实际上，它比在计算机的“扩充槽”上安装一块印刷电路板还方便。（如果你对计算机内部情况确实一无所知，就请别人帮帮忙）。

你一旦打开了机器的外壳后，插上8087不消一分钟。然后，可能你还要对硬件进行一次修改。你的计算机可能有8087出现以前的软件，如已经写入只读存储器的BASIC解释程序。假如你从制造厂家那里得到了与8087兼容的软件，你同时要更换ROM片子。

那些拥有不是以Intel 8088，8086系列为基础的个人计算机的人们，怎样办呢？他们能利用8087的高速度特性吗？不幸得很，回答是“不能”。8087只能与Intel系列一起工作。然而，由于Intel系列产品用得非常广泛，有些有远见的厂家现在开始销售一些带有Intel处理器的插件板，这些插件板适用于APPIE和其他计算机。这些插件板有一些已含有8087片子，有一些则留有增加8087的位置。这些插件板並不能使处理器的原有程序加速，但是它们允许你使用这本书里的程序以及其他与8087兼容的软件。

第二章　Intel 8087 芯片

处理器和协处理器

任何计算机的“大脑”是CPU，即中央处理器。IBM PC 和其它“第二代”个人计算机的“大脑”是Intel 8088。8088是完全通用的单片中央处理器，它有一套完整的指令系统，可用于8位和16位整数算术运算、逻辑运算和输入/输出操作。象大多数微处理器一样，8088也缺少大型和中型计算机中的先进的数学运算指令。

Intel 8087数值数据处理器（NDP）增加了新而完善的数学运算指令，从而扩充了Intel 8088的指令系统，8087高速硬件进行的一些数学运算，若用软件实现可能需要上千条指令。8087的硬件操作速度却比等效的软件快10到200倍。

从程序员的观点来看，8087在8088的指令系统中又增加了若干指令，并采用了很有用的，附加的处理器寄存器。为什么不将所有的功能包含在一个芯片上，而要制作一个附属芯片呢？

原因有三个：

- 8087是一个在单个芯片上集成75000个晶体管的非常完善的运算器件。即使8087仅“限于”数值处理，它也比通用的8088复杂得多（也贵得多）。制成两个独立的芯片降低了开发成本，并让用户和系统制造商根据不同的用途装配不同的系统。
- 8088（及其同系列的产品）在8087问世前好几年已投放市场。在设计16位个人计算机时，几个制造厂家都留出一个空插座，在IBM PC上标明为“协处理器插座”，因此有了8087之后，这些机器很容易升级。
- 由于8087和8088是两个器件，所以，它们可能同时执行指令。在实际编程时，这意味着8087进行一个数值运算时，8088则在为下一个数值运算作准备。

本章后面概述一下8087的功能，第五章给出更详细的技术性说明。

8087概述

8087可用作8088的“协处理器”。8087“监视着”8088取出的指令。8087执行自己的指令，同时让8088的指令通过，8088也能“注视”所有的指令，执行它自己的指令，同时让8087的指令通过。8088还为8087提供重要的服务。当8088遇到一条8087指令时，就计算这条指令所需的存储器地址，供8087使用，然后8088立即开始取下一条指令。这样的协处理器结构允许8087和8088同时执行指令，从而大大提高了整个系统的性能。

8087结构的主要特点就是有8个8位的数据寄存器。这些寄存器按曲型的“后进先出”栈的形式编排，这种编排方式提高了向量操作速度，又改进了高级语言编译程序的生成代码效率。（第五章将详细讨论“先进后出”栈。）80位寄存器的宽度使8087的计算精度极高。虽然8087指令系统可以识别存储器中七种不同的数据类型，但当这些数据进入8087后，都自动地转换成80位的内部表示形式。这使程序员省去了进行数据类型转换的巨大麻烦。

指令分类

8087有68条指令可分为六类（这种分类方式是为了便于说明8087的功能，使用8087不必记住这种分类）。

这六类指令为：

数据传送类（第六章讨论）。这类指令用在8087的内部寄存器和存储器之间传送数据和8087内部寄存器之间传送数据。

算术运算类（第六章讨论）。8087指令系统的核心是进行加、减、乘、除运算；以及诸如平方根和绝对值等指令。

超越函数类（第十二章讨论。）8087硬件能计算对数和三角函数（这些指令即使在大型计算机上也罕见）。

常数指令类（第六章和第十二章讨论）。8087中有七个最常用的常数，如0，1，π等。

比较类（第六和第七章讨论），这类指令用来进行小于、等于或大于比较以及其它类似的测试操作。

处理器控制类（第十二章讨论）。这类指令使程序员能全面控制8087的运算。其中有些指令也与比较指令以及8088的转移指令一起使用，以控制程序的流向。

数据类型

第五章深入考察8087七种通用的数据类型。然而在讨论最平常的8087编程时，真正重要的只有几种类型。BASIC中可以直接使用的数据类型仅有整型、单精度和双精度。通常只有后两种用来处理数值数据。如果8087主要用于科学计算，则关于数据类型只要记住下列三点：

1．单精度数字（8087中称为短实数）有6或7个十进制数位的精度，在存储器中占4个字节。

2．双精度数字（8087中称为长实数）有15或16个十进制数位的精度，在存储器中占8个字节。

3．临时实数，它是8087进行所有计算时内部使用的格式。它的精度超过18位十进制数。一个临时实数在存储时占10个字节。

如果你是一位进行数值计算的新手，这三种数据类型大概可以满足你95%的用途。但是8087还能处理下面四种数据类型：

1．整数（8087中称为字整数），占两个字节，主要用于检索数组和其它数据结构。BASIC和8087使用相同形式表达整数。

2．短整数为4个字节，带符号的最大字整数为32768，而最大的短整数为2×10^{9}。

3．长整数为8个字节。长整数的精度可以比双精度实数高2或3位十进制数，表示的最大数值为10^{18}。

4．压缩十进制类型，用于事务和数据处理操作。一个压缩十进制数占10个存储器字节，可以表示18位十进制数，与前面的三种数据类型不同，压缩十进制格式使用十进制表示而不是二进制表示。每一个0—9的数由四位二进制数表示，然后这些十进制数位每两个压缩在一个字节里。

对比之下，8088硬件所能识别的数据类型仅限于一个或二个字节的二进制整数和短的压缩十进制数值。8087以前的BASIC和其它高级语言的所有数值处理都是由依靠整数操

作所建立的软件进行的。8087则不需要这样的软件。以8087为基础的系统不仅处理速度快，而且程序所使用的存储器空间也少得多，且数值结果更加可靠。

计算机怎样利用8087的功能？

下一章将讨论8087的软件。为了理解为什么有些软件与8087兼容，有些则不与8087兼容，熟悉一下8088／8087的协作处理机是有益的。

在设计8088指令系统时已考虑了日后的扩充。其中有一条指令称为ESC（换码）指令，8088知道换码指令实际上要求对8087执行一次操作，所以它根本不处理这条指令，而让8087来处理。8087使用的那些指令都是8088ESC指令的各种变型。

当装上8088和8087后，便把这种结合看成一个已扩充了功能的大计算机。使用ESC指令的软件要正确操作，就必须使用8087芯片。8087出现以前设计的软件不使用ESC指令，因此也就不能利用这种新的能力。

如果你使用机器语言来编写程序，就知道你是否使用了ESC指令。但在你使用计算机其它大部分时间里，这种内部细节你是管不着的。下一章将讨论与8087兼容以及与8087不兼容的某些软件。

第三章　购买与编制8087——兼容软件

购买或编制供8087使用的软件要特别注意些什么呢？第一个问题总是“软件干些什么？”，第二个问题是“性能怎么样？”。这一章分三节来研究软件兼容性。第一节讨论有关兼容性的一些重要的技术细节，第二节分析为什么有些软件能生成执行速度非常快的程序，而另一些则不能。最后一节则依据编程方便性和计算速度讨论各类软件的优点。

兼容性——技术细节

假定我们讨论一个已经转换成计算机的“机器语言”的程序，这个程序可能使用了上一章提到过的驱动8087的ESC指令，也可能没有用到ESC指令。如果没有用ESC指令，这个程序就与8087不相干，你的计算机有没有8087芯片，该程序运行的速度都是一样的。如果这个程序使用了8087指令，那么当然必须有8087才能运行。

软件兼容性另一个同样重要的问题取决于软件设计的细节。所有计算机中的数字都以0和1的特定位组合格式表示，计算机不同，表示同一个数所用的位组合格式也各异。在大多数情况下，我们并不关心计算机使用哪种格式，因为我们并不能看到单个的0和1。重要的是计算机硬件要知道怎样解释它使用的格式（8087使用的表示方式已作为工业标准，要想知道其格式，请见第五章）。

在8087出现以前，以8088系列为基础的个人计算机硬件只能进行整数运算。由于硬件没有规定非整数数值的表示方式，所以每个软件设计人员可以自由选择自己的表示格式。实际上，这意味着不论谁编制编程语言的翻译程序（编译程序，解释程序和汇编程序），都要决定何种数的表示格式。由于Microsoft公司已成为16位计算机编程语言的主要提供者，故绝大多数软件都使用由Microsoft决定的程式。

不幸的是，Microsoft的格式与8087的格式不相同。

由于这种矛盾，8087以前的软件和与8087兼容的软件之间不能交换各自用内部格式表示的数据。在你正确插上8087以后，你可以十分可靠地运行8087以前的软件或与8087兼容的软件。如果你试图将8087以前的翻译程序生成的程序与8087兼容的翻译程序所生成的程序相连接，通常你会得到一堆无意义的信息。进一步讲，如果你试图在这样的程序之间交换数据，那么当这些数据是按计算机内部格式存贮时，得到的也是一堆无用的信息。如果数据不按内部格式存放，这些程序之间很可能可以交换数据。还没有一个通用的标准来判定两个软件之间是否会出现矛盾，因而你需要熟悉每个程序的细节。本章第三节将给出一些有这种冲突的例子。

什么因素决定程序的执行速度？

有三个基本因素决定了程序的执行速度：所用的解题方法（也就是计算机专业人员常说的“算法”）；所用硬件的速度和编程语言翻译程序的性能。第一个是最重要的因素。即使计算机再快，如果算法太差，也会使解题的速度慢如蜗牛。第三部分有关应用的各章里给出了关于数值运算编程中许多题目的高速解法。

有了8087硬件的速度问题就迎刃而解。假如硬件是速度的唯一决定因素，那么有了8087，程序执行时间可能减少到原来的1／10到1／50。

然而硬件不是唯一的决定因素。利用8087，程序的执行速度到底能提高多少，是百分之几，还是高达200倍，这取决于程序如何翻译成计算机能理解的指令。正是由于上述原因，同时，由于你可以对所用的翻译程序行使大量的控制，所以我们关心的是第三个因素。

源程序的转换

假定我们让计算机将变量A和变量B相加，并把结果存放在变量C中。典型的命令语句可能为：

C = A + B

从命令到结果，处理流程分三个阶段：

- 转换时间
- 调用时间
- 计算时间

转换时间是计算机用来解决干什么的时间。例如，每次BASIC解释程序遇见"C = A + B"，它就将这个等式的含义解释为"先找出存储器中的变量A，再找出变量B，接着将它们相加，最后将和放在变量C中"。BASIC编译程序所作的转换与解释程序一样，不过它只转换一次，不必每次执行都重新进行转换。解释过的程序在下次转换阶段仍要花费很多时间，而编译过的程序则根本不要再花时间。

调用时间是计算机计算变量的地址和调用相应的内部子程序所花的时间。例如，BASIC ROM包含有一个浮点加法子程序，解释程序调用这个子程序执行A加B操作。由BASIC编译程序生成的代码调用程序库中的一个类似的程序。

计算时间是计算机执行实际加法所花的时间。8087所有的直接好处都是来自计算阶段的改善。

由于8087只提高计算速度，所以大部分时间用于计算的程序就会因此而加速。而大部分时间用于转换和调用的程序其速度提高不多。要减少转换和调用时间就得选择合适的翻译程序。

你可能认为我们总是选择给出最快结果的翻译程序，然而，有时还要权衡。例如，编译程序生成的程序比解释程序要执行得快，但解释程序更便于使用。实际上，几乎每台个人计算机都有固化的BASIC解释程序，但并非每台个人计算机都有固化的BASIC编译程序。

那么，每个阶段的重要性如何呢？这个问题的答案取决于问题本身。第一章中给出了矩阵乘法和5000次平方根运算的一些有代表性的时间。我对8087以前的BASIC解释程序，与8087兼容的BASIC编译程序以及8087汇编语言程序在各个阶段所花的时间进行了估算。表3—1给出了单个加法和乘法（就矩阵程序而言）以及对向量的一个元素开平方所花的时间（单位为微秒），我不得不告诉你，表3—1中的时间比本书中其他地方给出的时间精度要差得多，但它还是可以作为权衡考虑的一个粗略的参考。

在下一节的讨论中，我们要多次用到表3—1。虽然表3—1表明汇编语言程序速度快，但它没有反映出通常编写汇编语言程序比编写BASIC程序要多做许多工作。凭经验，一个汇编语言程序比用BASIC写的程序代码量要多十倍。

大部分数值计算使用所谓的"线性操作"。可以将一小簇程序，如矩阵乘法程序等放在一起，以解决各种不同的线性问题。若有一个包含有这些子程序的库，例如将这本书中的

表3—1 执行时间分类（时间单位：微秒）

	矩阵问题（转换 调用 计算）			平方根（转换 调用 计算）		
解释程序	（8400）		1200	（3600）		6800
编译程序	0	135	56	0	66	70
汇编语言程序	0	10	56	0	0	70

程序组合成一个程序库，你不必亲自编写任何子程序就可以解决大部分问题。

平方根的例子有些不同，它是“非线性”操作，与线性操作完全不同，没有一小簇程序可以重新组织在一起，以满足解决非线性问题的需要。所以，解决非线性问题更加需要定制程序设计。需要的编程越多，就越要考虑有利于编程方便而不是计算速度。

后面几章将给出用汇编语言编写的矩阵乘法和平方根运算的程序。作为汇编语言程序，这两个程序编写都不很困难。（当然，你也没有必要编写这些特殊的程序，因为有现成的可用。）

计算精度

精度应与速度一样受到重视。8087的计算精度很高，但大部分翻译程序不允许你访问8087的80位寄存器。汇编语言能充分利用8087的精确性。因为有一些编译程序是专用于8087的（值得注意的是，它们都是Intel开发的）。这些编译程序虽然能提供80位数据操作，但目前尚未普遍使用。

对于某些问题，为了得到80位的精度，编程麻烦一些还是值得的。但对于“日常”应用，用双精度就够了。(对于要进行大量数值计算的应用，双精度的损失是个人计算机上使用的几个著名的编译程序所不能接受的。）这本书中的汇编程序在其“精确”的计算部分使用80位的数据，其他部分用的则是通常的单精度或双精度的数据类型。

与8087兼容的软件

本节讨论购买和编写与8087兼容的软件的一些方法。在每一种方法中，我们在编程方便性和执行速度之间进行权衡。所讨论的方法是：

- 使用成套的程序
- 8087硬件配上8087以前的软件
- BASIC 解释程序
- 具有8087浮点程序库的编译程序
- 生成8087“本机”码（native code）的编译程序
- 适用于 BASIC 的汇编语言模块
- 纯粹的汇编语言代码

成套程序的使用

8087使用“成套”程序究竟有多少好处，这取决于所写程序的质量。同所写的任何程序

相比，一个真正写得好的成套程序，更能充分利用8087的能力。这并不是因为编程人员知道你在这本书中没有发现的有关8087的任何秘密。而是由于编程人员对要销售几千份拷贝的每一个程序都尽力花时间压缩它的执行时间，那怕是压缩1个微秒。遗憾的是，还没有一个真正令人满意的方法如何了解一个成套程序未经“现场测试”其质量是良好的。除此以外，软件手册几乎从不给出有关执行速度的任何信息。

你可以找到三种广为宣扬的软件包（就与8087兼容而论）。

第一，有一些程序，只打算用于8087，并不要求与早期的软件或没有8087的微机兼容。许多应用问题在没有8087的微机上不能解决（要化相当长的运行时间）。对于解决这种问题的程序，兼容性不是主要的问题。实际上，8087的速度对于某些应用至关重要，所以，有些软件公司在个人计算机厂家销售8087之前，就已向市场投放用于8087的软件包了。

第二，有些程序要么在8087上运行，要么不在8087上运行，而有些程序只有一个版本，两种情况下都能运行，又有一些程序则有两个版本：一个用于8087，另一个则不使用8087。

第三，有些程序忽略8087，这些程序几乎都可以在8087上运行。只要计算机上有与8087兼容的BASIC解释程序，那些用BASIC语言写的程序则能自动地在8087上运行。

成套程序的第一个注意事项，许多高效率的程序以所谓二进制文件的方式在磁盘上存贮信息。二进制文件存贮数据使用计算机的内部数字表达方式，而不是使用磁盘存储器常用的“ASCII”表示方式（这种方式使程序避免在内部和外部格式之间进行转换，因此数据存贮和索取非常快）。象前面所讨论的那样，8087使用的内部数据表示方式与8087出现以前的大多数软件不同。由于这个原因，8087出现以前的二进制文件和与8087兼容的二进制文件是互不兼容的。

如果你使用8087出现以前的程序在磁盘上保存二进制文件，然后又使用与8087兼容的文件，这时你就不能将盘上的文件再读出来。而且，由于你通常不了解文件格式的说明，所以你不可能自己转换这些文件。为了防止这一点，当你要使用一个具有二进制文件的成套程序时，就要用该程序将二进制文件转换成ASCII文件。同时你仍可使用8087出现之前的软件，尔后再将ASCII文件转换成二进制文件。

成套程序的第二个注意事项，许多高效率的程序使用少量的汇编程序代码，以加速重要计算。一般情况下，你没法发现某个软件包是否使用了机器代码。如果计算机语言程序以为数字是用Microsoft的原始格式存贮的，而程序的BASIC部分则以Intel格式操作，……好了，你可以想象出结果了。

在8087硬件上配备8087出现之前的软件

如果我们可以不用关心软件就能从8087受益那就太好了。鉴于上述原因，这当然是不可能的。例如，假如在备有8087出现以前的BASIC解释程序的机器上加上8087，你的BASIC程序仍可以运行，但是它们并没有使用8087。一般情况下不是这样。

然而，要知道的是翻译程序而不是程序本身需要与8087兼容。如果你有与8087兼容的BASIC解释程序或其他与8087兼容的翻译程序。你原有的BASIC程序仍可运行，并可以利用8087（这说明了为什么应尽量使用BASIC或其他标准“高级语言”的一个重要理由。如果硬件改变了，例如增加了8087，你只要获得一个新的翻译程序，一般就不需要重

新编写你的应用程序了）。

在Intel 和 Microsoft 的数据格式之间直接地进行相互转换，就可能将与8087兼容的软件和8087出现以前的程序连接在一起（转换程序在附录里）。例如，你可以在IBM PC的原有8087汇编程序下使用本节中的8087程序，但另外要进行一些 BASIC 的编程。

BASIC解释程序

根据你购买PC时间的不同，PC上可能有与8087兼容的BASIC解释程序，或者以购买这样的程序来代替机器中原来的 BASIC ROM，对于大多数应用，BASIC 解释程序提供最方便的编程，但执行速度极慢。

事实上，8087不影响解释程序进行转换或调用阶段的操作速度，加上8087使计算阶段的操作加速，见表3—1。象矩阵乘法这样的问题，大部分工作用在转换和调用中。所以，你不用期望使用 8087 的速度能比8087出现以前的 BASIC 提高10%到15%。

在平方根问题中，在全部执行时间中计算时间占的比例很大，因此，8087对计算时间影响非常大。我们能够期望整个计算速度提高三倍。一些非线性函数，例如三角函数运算，计算阶段甚至要化费更多的时间。在某些情况下，计算速度可提高8倍。

现在我们得出一个结论：

如果你的大量数值计算大都是线性运算，那么，使用8087及改进的 BASIC 解释程序提高的速度是有限的。

如果你的大量数值计算大都要使用非线性函数，那么，使用8087和改进的 BASIC 解释程序，其速度是不使用8087的 PC 的几倍。

使用8087的 BASIC 版本时，有一个特别要注意的地方。不论别人对你怎么说，8087和非8087的 BASIC 版本终究不是完全兼容的（尽管它们很接近）。由于浮点数的表示方式不同，就没有办法使它们完全兼容。这里存在两个基本上不可解决的问题。

首先，这两个程序的浮点表示的精度和范围有些不同，尤其是双精度数，Intel 格式的精度大约要低一位十进制数，用以增加阶码的范围。极少数情况下，在原来的 BASIC 解释程序下工作正常的程序，当用于 BASIC 解释程序的8087版本时，由于舍入误差会给出错误的结果。在新版本 BASIC 的解释程序下运行的程序，如果使用旧的 BASIC 解释程序执行，通常会出现溢出。幸运的是，这类问题很少见，而且不太可能成为大多数用户所关心的重要问题。

其次，有些程序使用 MKS$，MKD$，CVS 和CVD一类 BASIC 函数，进行浮点数和字符串的相互转换。这种函数在两种 BASIC 版本中都有。但是，假如你用其中的一个版本向磁盘上存贮了许多数字，再用另一个版本将数字读出来，你会在得不到任何错误指示的情况下，取回一些作废的数字。如果你在不同版本程序运行中间，使用二进制表示的文件进行存贮，务必保证将这些文件从一个 BASIC 版本（适用的表示方法）转换成另一版本的表示方式。

具有8087浮点程序库的编译程序

编译程序与解释程序不同，它对源语言程序只转换一次，而不是在每次执行一行代码时都进行转换。编译程序也有一些缺点，程序转换的时间相对于（解释程序）要长些；所生成的代码比解释程序生成的代码所占空间要多一些；降低了程序调试速度以及成本较高。但它们优于解释程序，这里无可争辩的是：它们能省掉程序执行中的转换阶段，因此大