杜毅仁　李慕靖
王建中　陈启帆　编

16位微型计算机

SHI LIU WEI WEI
XING JI SUAN JI

——上册

上海交通大学出版社

计算机

十六位微型计算机

（上　册）

杜毅仁 李慕靖 王建中 陈启帆 编

上海交通大学出版社

内 容 简 介

本书从微型计算机的硬件、软件、系统结构等方面出发，通过 Intel 公司的 8086 系列以及 IBM 公司的个人计算机作为典型，进行系统的介绍和剖析，使读者获得有关微机的基础和应用知识。

全书共分上、中、下三册。上册通俗易懂地介绍了 8086 的硬件结构、指令系统及其程序设计语言，是全书的基础和核心部分；中册就整个 8086 系列，分别介绍了输入/输出处理器 8089、专用数值处理器 8087、操作系统固件 80130、8086 的改进型 80186 和 80286，同时，还对 MULTIBUS 这一标准的系统总线以及 8086 的其它配套系列芯片作了介绍；下册比较系统地分析了IBM公司的个人计算机，内容包括 PC 的硬件结构、系统固件及操作系统等。

本书具有硬件、软件相结合的特点，而且既能了解 8086、又能由此了解十六位微型计算机的一般原理及应用。因此，既可作为高等院校的教学参考书，也可作为研究院所、厂矿有关研究人员、工程技术人员乃至一般用户的工作手册。

十六位微型计算机

（上册）

杜毅仁 李慕靖 王建中 陈启帆 编

上海交通大学出版社出版

上海淮海中路 1984 弄 19 号

新华书店上海发行所发行

立信会计专科学校常熟市梅李印刷联营厂印装

开本：787×1092 毫米 1/16 印张：25.5 字数：583,000

1984年 7 月第一版 1986年 1 月第 2 次印

印数10,000—20,000

统一书号：15324·33 科技书目：114—234

定 价：4.70

目　　录

第一篇　8086的体系结构，系统设计和程序编制

第 一 篇

8086 的体系结构，系统设计和程序编制

本篇内容是对 8086 微处理机的一个全面介绍。它描述了 8086 的系统结构，以及怎样设计某一个 8086 的系统。论述尽可能详细，并且着重于举例和图解，希望它对于微型机的初学者和专业人员都有用。

本篇由三个主要部分组成——8086 的体系结构，8086 的系统设计和 8086 的程序编制。体系结构被细分成第二章 8086 的机器组成（寄存器、存贮器结构和寻址方式）和第三章 8086 的指令系统。8086 的系统设计在第四章中讨论，在这一章中介绍了如何把 8086 微处理器和其他部件组合起来以构成一个完整的微处理机系统。程序设计部分分成 8086 汇编语言程序设计（第五章和第六章）和 8086 高级语言设计（第七章）。

通过第一章的介绍，希望把水平不同的读者对于计算机和微型计算机的认识提高到一个共同的水平。如果读者已经具有那种知识并且迫切地想了解 8086 的话，则可以跳过第一章而直接进入第二章。

第一章 引 言

这一章叙述了微型计算机，特别是 8086 的技术特点和历史沿革。微型计算机除了在微型和价廉之外同任何计算机没有什么本质的不同，因此，本章将先从计算机的基本原理出发，然后叙述微型计算机的发展进程，最后阐述 8086 的特点。

§1.1 计算机概况

在讨论微型计算机之前，先让我们简短地介绍一下计算机。除了作为一个回顾以外，这一节还要引进一些在本书中使用的术语和概念。

组成一个计算机系统的基本部件表示在图 1.1 中。图 1.2 用非人物化的方法表示了同一个系统。下面我们集中在每一个方框的功能上来说明这样一个系统的特性。

计算机从一个输入设备取得数据，在计算和处理数据后，把最终结果送到输出设备。所要做的计算和处理可由一张被称为程序的指令表来说明。这个程序通常放在存贮器的程序区中。

计算机的操作由一个称为控制部件的设备控制。它检查、判断和控制机内各部件重复做下列三步：

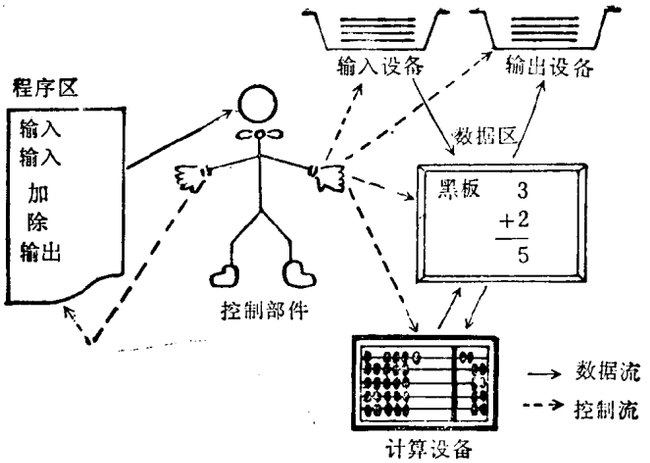

图 1.1 原始的计算系统

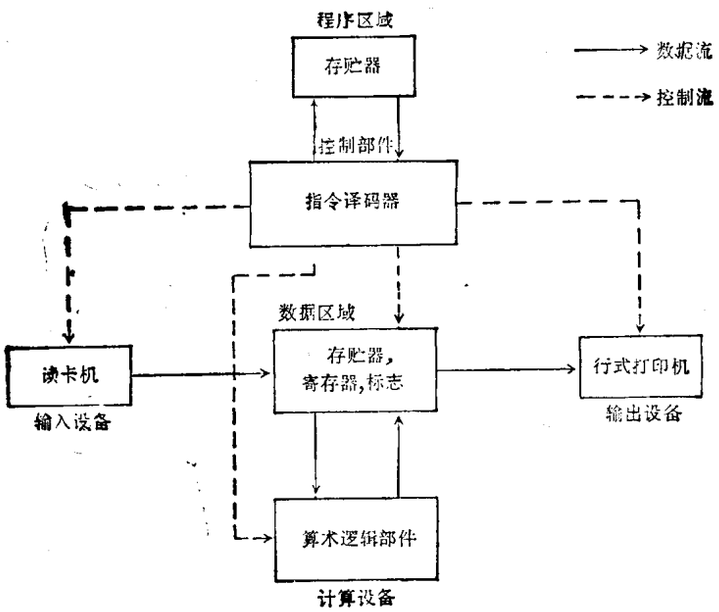

图 1.2 近代通用计算系统

1. 从程序区取指令。

2. 将指令译码以决定要执行何种操作。

3. 把控制信号发送到执行操作的设备去执行指令。

在指令执行期间实现的操作，大体上是由在设备之间传送数据和在设备内部对数据实行计算等部分组成。计算是由运算设备实现的。数据区域是指用来为计算提供数据和存放中间结果的那一部分存贮器区域，以及通用寄存器和特征位寄存器。

让我们通过分析一条“加法”指令的执行过程，来了解这个系统是怎样相互连接在一起的。控制部件发出一个控制信号，并把它送到程序区，请求下一条指令。程序区把一条指令发送到控制部件作为响应。控制部件对该指令译码后，发现是一条“加法”指令，于是控制部件就把

控制信号发送到：

(1) 数据区，要求它传送两个数值到运算设备。

(2) 要求运算设备把取得的两个值相加。

(3) 通知指令中指定的数据区单元接收相加的结果。

程序区和数据区虽然通常都是存放信息的存贮器的组成部分，但在每个区中所放的信息种类则有很大的不同。数据区放中间结果，它在程序执行期间是经常改变的。程序区放程序，它在执行某个既定程序时通常是不改变的（近年来，在执行过程中修改自身程序的工作方式，已经不怎么流行了）。在一些系统中，程序事实上是“固化”在存贮器中的，所以它们只能被读出，而不能被改变。具有这种性质的存贮器被称为只读存贮器（简写成 ROM）。很明显，ROM 不适合用于数据区。数据区由读-写存贮器组成，一般简称为 RAM（RAM 是随机访问存贮器的缩写）。

存贮器是一个连续存贮单元的集合，每个单元具有一个唯一的地址。每个单元含有一串连续的 2 进制数的位，这些位或是 0 或是 1，从而构成一个 2 进制数。关于 2 进制数在这一章的后面还要加以介绍。

前面已经说过数据区由寄存器和标志以及存贮器组成。象存贮器一样，寄存器也是用来存贮中间结果的。通常在寄存器中存取数据比在存贮器中更容易，速度也快得多。计算机使用标志作为指示器来决定下一步要做什么。标志可以分成二类，即记录前面执行指令所产生的结果的特征的状态标志和控制计算机操作的控制标志。例如，溢出标志就是一个状态标志，它指出操作的结果是否超过了计算机所能保持的精度。中断允许标志则是一个控制标志，它表示处理器是否能接受外部的中断。

在计算机系统中的另一个设备是输入输出转接口（也称为 I/O 转接口）。一个输入输出转接口就象一扇通往外部世界的门，通过它才能与输入设备或输出设备传送信息。为了使原理图简化，I/O 转接口没有在图 1.1 和 1.2 中画出。

§1.2 数 据 格 式

一个存贮器单元的内容可以是程序中的一条指令，也可以是一个数据。指令在存贮单元里存放的方法，称为指令格式，它对各种计算机来说都是不相同的。8086 的指令格式将在第三章中详加描述。这里只介绍在 8086 中使用的数据格式。

由计算机处理的数据可以是数值信息也可以是非数值信息。一个工资程序可能大量地使用数值数据，而一个文本编辑程序却要进行大量的非数值操作。用来表示非数值数据的格式被称为 ASCII 码。

一、数字系统

我们已经习惯于把数表达为 10 进制数字的序列，例如 365。这个数可以理解成为 3个100，6 个 10 和 5 个 1。数的这种表达法常常称为基数为 10 的表达法。人有 10 个手指，用 10 为基数的表达法来表示数，我们感到是天经地义的。但计算机是用电平高低来计数的。为了可靠，只用 2 个电平值，要么是高电平，要么是低电平（或电平等于 0）。于是，对于计算机来说，用以 2 为基数的表达法来表示它们的数，也是天经地义的。所以，在计算机中数是以 2 进

制数字 (bits) 的序列来表示的.例如，11010，这个数字可以理解为 1 个 16，1 个 8，0 个 4，1 个 2 和 0 个 1 所组成的数的以 2 为基数的表达法．由于 2 进制数字只有 0 和 1，它逢 2 就进 1，所以我们只要记住 1+1 等于 10 而不是 2，不需要先把它们转换成 10 进制数，2 进制数也能直接进行＋、－、×、÷运算．

例如：

```
  1001      9 的 2 进制表示法
+ 0101      5 的 2 进制表示法
------
  1110     14 的 2 进制表示法
```

尽管计算机擅长于处理 2 进制表示的数，但这种很长的 2 进制数字序列人们却常常容易搞混淆．例如，181 的 2 进制表示是 10110101．为了压缩数字的长度，使它更便于记忆，人们根据 2 进制数的特点引出了数的 16 进制表示法，不言而喻，这种数的表示法的基数是 16．由于 2^4 等于 16，16 进制数必然和 2 进制数有着十分密切的联系．事实正是这样，16 进制数的每一个数字，恰好能用一个 2 进制的 4 位组来表示，它们的对应关系在表 1.1 中表示．这样 10110101 可以缩写成 B5，这样的数称为 16 进制数．如果人们生来就有 16 个手指的话，无疑这就是我们所要使用的数字系统．现实世界中也有应用 16 进制的，中国的老称，不是 1 斤等于 16 两吗！

表 1.1　16 进制数字表达法

2 进制数	16 进制数	10 进制数	2 进制数	16 进制数	10 进制数
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	A	10
0011	3	3	1011	B	11
0100	4	4	1100	C	12
0101	5	5	1101	D	13
0110	6	6	1110	E	14
0111	7	7	1111	F	15

二、带符号数

2 进制数的概念对于描述正数和 0 是很完美的，但当我们要把它用于负数时，就需要有一个附加的机构来指明该数的符号．这样做最简单的办法是把该数的最高位（最左位）作为符号位，例如：

```
0000 0100      是 +4
1000 0100      是 −4
0111 1111      是 +127
1111 1111      是 −127
```

这样的表达法称为带符号的表达法，但它有一个严重的缺点：即它需要一个新的运算规则．这一点在要从 0 中减去 +1 而希望得到 −1 时是很明显的，例如：

```
  0000 0000      带符号表示的 0
− 0000 0001      带符号表示的 +1
-----------
  1111 1111      带符号表示的 −127
```

若我们想要把在无符号数上用的2进制运算规则运用到带符号数上去，我们就需要另一种符号数表示法，用这种方法1111 1111代表 −1 而不是 −127。同样，从 −1 减去 +1 应该得到 −2。让我们实行这个减法来看 −2 应该是怎么样的。

```
  1111 1111        −1
− 0000 0001        减去 +1
-----------
  1111 1110        把这个称为 −2
```

这样，正数和负数似乎应该表示成这样：

```
     ⋮
0000 0011        +3
0000 0010        +2
0000 0001        +1
0000 0000         0
1111 1111        −1
1111 1110        −2
1111 1101        −3
     ⋮
```

事实上也正是这样，这种表达法称为2的补码表达法，它具有可以运用2进制加法和减法规则，並给出正确的2进制补码结果的性质，例如：

```
  0000 0011        2的补码表示的 +3
+ 1111 1110        2的补码表示的 −2
-----------
  0000 0001        2的补码表示的 +1
```

它也具有每个非负数（正数或0）的最高位是0和每个负数的最高位是1的性质。这样，就和带符号的表达法一样，最高一位起符号作用。

一个2的补码数的相反数可以通过改变每一位的值和末位加 +1 来获得。例如，可以从 +3 的2的补码表达法中取得 −3 的2的补码表达法如下：

```
  0000 0011        2的补码表示的 +3
  1111 1100        每一位都改变了的 +3
+ 0000 0001        末位加 +1
-----------
  1111 1101        2的补码表示的 −3
```

关于2的补码数有一点要引起警惕，即当要把一个8位的2的补码数扩展成16位（例如，使它能和一个16位的2的补码数相加）时，怎样正确地处置前面附加的8位。请先看下面的例子。

假如要给0000 0000 0000 0011（2的补码表示的 +3），加上0000 0001（2的补码表示的 +1），在这种情况下毫无疑问，我们会在 +1 的左边简单地加上8个0，然后再把两个数相加。

```
  0000 0000 0000 0011        (用2的补码表示的 +3)
+ 0000 0000 0000 0001        (用2的补码表示的 +1)
---------------------
  0000 0000 0000 0100        (用2的补码表示的 +4)
```

但是，如果我们要把1111 1111（2的补码表示的 −1），加到0000 0000 0000 0011（2的补码表示的 +3），却必须在 −1 的左边加上8个1（加0会使它变成正数）。这个加法就是：

```
  0000 0000 0000 0011      (用2的补码表示的 +3)
+ 1111 1111 1111 1111      (用2的补码表示的 -1)
-------------------
  0000 0000 0000 0010      (用2的补码表示的 +2)
```

这样，把一个8位数扩展成16位时要这样进行：

数值	8位表示	16位表示
+1	0000 0001	0000 0000 0000 0001
-1	1111 1111	1111 1111 1111 1111

扩展一个2的补码数的规则，是把原来符号位的值扩展到附加的8位上去。这种操作称为符号扩展，只有这样，才能正确执行补码运算。8086中就应用这种方法。

三、字　符

字符可以用一个位的序列来表示。我们要表示的最低限度的字符集是26个字母和10个数字，共36个字符。若要能区别大写和小写（另外26个字符）字母，并且能表示一些特殊的字符（例如+和*），则超过了64个字符，这样至少需要7位才能表示全部所需要的字符（一个6位的数所能表达的最大值是64）。一种通用的7位编码称为ASCII码（American Standard Code for Information Interchange），它被表示在附录C中。一个8位的存贮器单元称为一个字节，因为它能很方便地用来存放一个ASCII编码的字符（第8有时用来作为正确性检查位）。

§1.3 堆　　栈

堆栈是在微处理机中经常出现的概念。堆栈的另一个名称是“下推表”或“后进先出队列”。这些名称可以从堆积自助食堂碟子的设备中想象出来。当一个洗干净的碟子被放在碟子堆的顶上时，它把下面的碟子往下压一层，当这个顶上的碟子被人取去时，所有的碟子就往上弹一层。很明显，最后放在堆栈顶上的碟子将是最先被人取走的那一个碟子，而最先放在堆栈中的碟子将是最后一个被取走的碟子，这就是后进先出名称的来源。

值得注意的是，堆栈在计算机中是很有用的机构，为了说明这一点，我们有必要先看看子程序。子程序（有时也称为过程）是一个程序的一部分，它是被主程序调用来执行特殊任务的，它提供了把要解的全部问题分为较小和较简单的很多模块的方便。一个子程序本身还可以调用其它子程序，去进行更细分的工作。当一个子程序完成任务以后，它就返回到调用它的主程序，这样就形成了一个子程序调用序列，后一个子程序压在前一个子程序的上面，直到最后一个被调用的子程序为止。这最后一个被调用的子程序正是第一个要返回的子程序。换一句话说，子程序的重迭调用具有后进先出的特点。因此，堆栈就在此时发挥了作用。实际上，当一个子程序被调用时，某些信息必须保护起来。这些信息可能包括一些寄存器中的内容和当前设置的标志，当然也包括子程序最终要返回的地址。当子程序完成任务以后，它要恢复这些被保护的信息，把它们送回到原来的寄存器中去，其中包括把标志位恢复成原来的值，并且使用“返回地址”使之回到主程序去。由于最后一个被调用的子程序是第一个要返回的子程序，所以最后被保护的信息将是第一个要被恢复的。这样信息就必须要被堆放得如同自助食堂的碟子一样。至此，我们已经描述了堆栈是怎样动作的和为什么在计算机中是一种有用的机构。现在我们来看计算机的堆栈是如何实现的。由于堆栈必须保存信息，所以它肯定是某种存贮

器，事实上，存贮器的任何可用部分（只读存贮器除外）均可被用来作为一个堆栈。所需要的只是一个指向该堆栈最末一个单元的指示器。这个指示器常常称为栈顶指示器，而被该指示器指着的存贮单元通常被称为堆栈顶。当一块新的信息被推入堆栈中时（或称为压入堆栈），堆栈指示器被递减以使它指向下一个存贮器单元，而被推入的信息就存放在该存贮器单元内。当一个信息从堆栈中推出时（或称为弹出），信息从堆栈指示器指着的单元中弹出来，而堆栈指示器将被递增以指示当前堆栈顶的位置。

§1.4 8086 存贮器的分段

前面的章节举例说明了存贮器可以用来存放程序（代码）、存放数据（数和字符）和作为堆栈。这样，8086 事实上把它的存贮器分成代码段、数据段和堆栈段就不足为奇了。关于存贮器将在第二章中讨论。

§1.5 微型计算机的发展简史

在概述了计算机的基本概念之后，了解一下计算机的发展历史，并且看看它的演化到微型计算机的过程是有好处的。

一、从大型计算机到微型计算机

在 50 年代，所有的电子设备（收音机、电视机以及计算机）都是笨重的电子管设备。那个时代的计算机被称为第一代计算机。例如：IBM 的 650 和 704。这些计算机包含有若干个电子设备的机架，安装在很大的房间里。50 年代末期，晶体管和其它固态电路开始登上历史舞台并开始取代电子管。利用这种技术的计算机被称为第二代计算机。例如IBM 7090和Burroughs B5500。

在 60 年代，许多分立的电子器件（电阻、电容、晶体管等）被组合在一个单一的集成电路内（简称 IC）。IC 的面积比一张邮票还小，一般封装在象蜈蚣一样的管壳里。它能方便地插入一个系统。这种可插可拔的集成电路常称为中小规模集成电路。用 IC 制造的计算机是第三代计算机。例如 IBM360，GE635 和 Burroughs B6700。由于集成电路技术的飞速发展，到了 70 年代初期，在图 1.2 中的许多部件可以集成到一块很小的硅晶片上，例如，Intel 4004 和 8008，从而开始了晶片上的计算机时代。

到了这个时候，不仅是计算机的体积已经急剧地减小，而且价格也大大下降。电子管计算机的价格是以百万美元来计算的，而晶片上的计算机的最初价格是 300 美元左右，在几年之内竞争使得价格下降到 10 美元以下。

晶片上的计算机被称为微处理机或微处理器。虽然这些术语有时可以互换使用，但是却有不同的含义。一个微处理器是指单一的片子，它通常由控制部件、算术逻辑部件、寄存器、标志、输入/输出转接口等部件组成，也就是通常称为 CPU（中央处理部件）的部分，故微处理器就是微型计算机中的 CPU。程序和数据存贮器以及输入/输出设备通常不在微处理器片子上。微处理机相当于一台计算机的主机部分。微处理机通常由一片微处理器，许多存贮器片子和输入/输出接口片及外围电路组成。当然，也有整个微处理机包含在一个片子上的，例

如，Intel 的 8048，这被称为整单片微型计算机，也可称为整单片微处理机。由于微型计算机发展极快，其结构也日新月异，就是主要部件的名称也很不统一，有时令人捉摸不定。就目前情况来看，按微处理器、微处理机和微型计算机的层次命名还比较合理。它们的关系如图 1.3 所示。

微处理器：是指把运算器和控制器看成一个整体，将其做在一片大规模集成电路 (LSI) 上的器件。对该器件的称呼很混乱，有的称为中央处理器 (CPU)，有的称为微处理器 (MPU)，有的称为微处理机 (μP)，也有称为通用微处理机的。看来称之为微处理器比较合理，称之为中央处理器 (CPU) 也可以，因为 CPU 在计算机范围内用得比较久了，不会引起人们的误解。

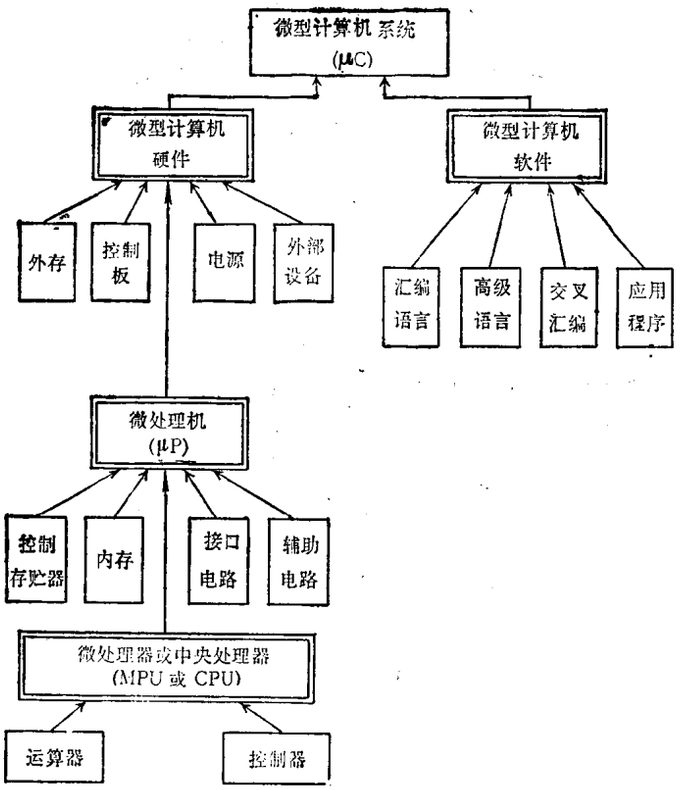

图 1.3　微型计算机的组成

微处理机：由图 1.3 可知，微处理器和其他部件，即存贮器片、输入/输出接口片以及辅助电路用某种方法结合在一起以后，就构成微处理机，它相当于一台计算机的主机部分，而且在某些专用场合，已经可以作为一台专用微型机使用了。把上述各种功能的片子组合在一起的方法主要有两种：一种是常规的装配方法；另一种是大规模集成法，即把上述功能片都集成在一片 LSI 或 VLSI (超大规模集成电路) 上，从而构成整单片微处理机，也有人称它为整单片计算机的。事实上，它们只能是一些专用机。

微型计算机：狭义地说，只要将上述微处理机加上电源部件、控制板、配上外部设备就构成一台微型计算机了，但这只是微型计算机的硬件部分，完整的微型计算机还应包括软件。有时把上述硬件系统和软件系统的总和称为微型计算机系统。这才是人们通常所指的微型计算机。

二、从8008 到 8086

微处理器时代是从 1971 年 Intel 推出 4004 和 8008 微处理器开始的。这是第一代微处理器。这两种片子都是为专门的应用而设计的，4004 主要用于计算器，8008 则主要用于计算机终端设备。这些微处理器虽然比较新奇，但并没有得到足够的重视。直到 1974 年，当 8008 成长为 8080 (第二代微处理器) 时，才引起了计算机工业的震动。8080 是第一个精心设计，可以广泛地在各方面应用的微处理器。它很快就风靡世界，变成了“标准”的微处理器工业产品。

微处理机现在已经能够实现老式的计算机能实现的各种任务，并且价格已经便宜到能够

到达业余爱好者手中的程度。许多 Intel 以外的公司也开始制造 8080 片子，而且有一些公司（著名的 Zilog 公司）造出了 8080 的增强型产品(即 Z-80)。Intel 公司本身也在 1976 年推出了增强型产品 8085。但是直到 1978 年 Intel 公司生产出 8086 为止，微处理器的基本特点并没有重大的改变。8086 与 8080/8085 是向上兼容的（以前为 8080/8085 开发的软件能够方便地转化为 8086 的软件），而 8086 的先进性足以称为第三代微处理器。

三、8086 成功的奥秘

8086 究竟提供了什么，使它能立即获得成功？要了解答案，我们必须先看看 8080/8085 的不足之处。

8080/8085 早期的成功，鼓励着用户把它使用在越来越大的系统中。这些系统变得如此之大，以致 8080/8085 的 64KB（其中 K=1024，B 表示字节）的寻址范围远远不能满足他们的需要了。8086 的寻址范围超过一百万存贮器字节单元。除此之外，8080/8085 也被越来越多地用在需要比 8 位长的快速处理领域中。8080/8085 的 8 位字长使得长的数据必须要由几个 8 位组来构成，而每一组又必须分开操作，这样就增加了处理时间。8086 虽然是基于 16 位字长操作的，但它也保留了处理 8 位数据的能力，从而使得短的数据仍能被 8086 有效地处理。考虑到 8080/8085 用于通用计算机系统时缺少乘法和除法指令，还缺少带符号数的操作，因而在使用上很不方便，8086 就提供了以前缺少的这些运算指令。由于越来越多的 8080/8085 程序采用高级语言编写，它们要先被翻译成 8080/8085 的机器语言，然后才能被执行。鉴于 8080/8085 的寻址方式不能支持把用高级语言写的程序转换成有效的 8080/8085 代码，8086 的寻址方式则设计得能适应高级语言的处理。从大量的应用中发现 8080/8085 在变换数据串时能力很差，8086 则设计得能有效地处理数据串。最后，随着系统变得日益复杂，单靠一个处理器来执行系统的全部功能的方法，越来越行不通。但是，8080/8085 没有能力与其他处理器合作，而 8086 则设计成能在多处理器环境下运行。特别是 8086 与两个出色的协处理器，即数字数据处理器 8087 和 I/O 处理器 8089 天衣无缝的联合，加上 80186 和性能更强的 8086 的增强型 80286（详见本书第二篇）的出现，更把 8086 的性能推到了前所未有的高度。所有这些就是 8086 成功的奥秘所在。

习　　题

1. 简述输入/输出部件，控制部件和算术逻辑部件在计算机中的作用。
2. ROM、RAM 有什么区别，哪一种适合于作为数据区？
3. 计算机一般有几类标志，它们的作用各是什么？
4. 控制部件的主要作用是什么？试以一条“减法”指令来分析控制的过程。
5. 计算机的所谓数据区由哪些部分组成？
6. 试把 10 进制数 37，58，103，196，512，1024 化为 2 进制数和 16 进制数，计算机中还有一种经常使用的 8 进制数(数字是 0，1，2，3，4，5，6，7)，你能把这些数化成 8 进制数吗？
7. 有四个 8 位的带符号数 0100 1001，1000 0001，0001 0101，1100 1000 试把它们符号扩展为 16 位数，若要扩展为 32 位数，怎么扩展？
8. 堆栈的工作原理是什么？试用一迭盘子体会一下放入和取出的操作。
9. 什么是子程序，它的主要作用是什么？
10. 什么是微处理器，微处理机，微型计算机，微型计算机系统，它们的区别和联系怎样？
11. 8086 和 8080 相比有哪些主要优点？

第二章　8086 的组成

§2.1　概　　述

描述一台计算机的一种方法是描述组成该计算机的功能部件。这些部件和它们之间的相互作用称为该计算机的结构。例如，在计算机中有多少寄存器，这些寄存器起什么作用，可以连接多少存贮器，存贮器是怎么被寻址的，以及可以使用哪种输入/输出机构等等。

8086 是一块含有构成一台计算机的绝大多数部件的大规模集成电路片子。其中包括控制计算机所有功能的控制电路，所有寄存器和标志位以及算术逻辑操作部件。存贮器和输入/输出转接口虽然没有包含在片上，但可以很方便地与其他外围支持片子相连接从而形成一台计算机。这些片子的集合有时称为微处理机，其中央处理器 (CPU) 则称为微处理器。

如果想简要地描述 8086 CPU 的结构，它应当是这样的：8086 有 4 个寄存器集。其中第1个集含有 4 个用来保存中间结果的通用寄存器；第 2 个集含有 4 个用来在存贮器中寻找信息时，作为指示器和变址器的寄存器；第 3 个集含有 4 个用来支持存贮器分段的段寄存器；第 4 个集含有指令指示器。8086 中还有 9 个标志位。这些标志用来记录处理器的状态和控制它的操作。8086 可以访问多至一百万个存贮器字节和多至 65,000 个输入和输出转接口。这一章的前半部分将详细地推敲这些特点，至于8086片子的引脚功能和内部结构则放在本章末尾来讨论。

一般的计算机指令都涉及到寻找指定的操作数 (要处理的数据)，在这些操作数上执行一个操作，并把结果放回到指定的单元。操作数和结果单元由指令指定，可以在存贮器中也可以在寄存器中。用来指定这些单元的机构称为计算机的操作数寻址方式。8086 的操作数寻址方式将在这一章的下半部分详加描述。在指定的操作数上进行操作的真实指令在第三章中描述。

§2.2　存贮器结构

8086 系统中的存贮器是一个可以多至 2^{20} (大约 1,000,000) 个 8 位数量的字节序列。每一个字节单元分配一个唯一的地址 (无符号数，2 进制从 0000 0000 0000 0000 0000 到 1111 1111 1111 1111 1111；16 进制从 00000 到 FFFFF)，这一点在图 2.1 中说明

16 进制数地址	2 进 制 数 地 址
00000	0000 0000 0000 0000 0000
00001	0000 0000 0000 0000 0001
00002	0000 0000 0000 0000 0010
00003	0000 0000 0000 0000 0011
⋮	⋮
FFFFE	1111 1111 1111 1111 1110
FFFFF	1111 1111 1111 1111 1111

存 贮 器

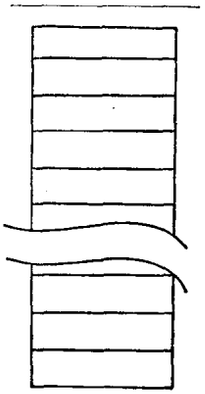

图 2.1　存贮器地址

在存贮器中任何两个相邻的字节被定义为一个字，在一个字中的每一个字节有一个字节地址，并且这 2 个地址中的较小的一个被用来作为该字的地址。字的例子在图 2.2 中表示。

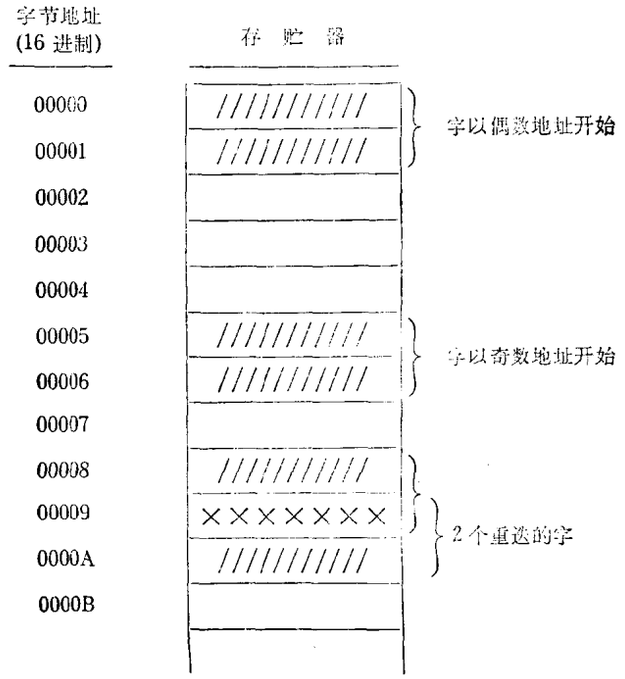

图 2.2 字在存贮器中的例子

一个字含有 16 位。具有较高存贮器地址的字节含有该字的高 8 位。具有较低存贮器地址的字节含有该字的低 8 位。初看起来，这一点似乎很自然。高字节当然应该具有较高的存贮器地址。但是，当考虑到存贮器是一个从最低地址开始到最高地址为止的字节序列时，很明显，8086 是反向地存放它的字的(或许它们应当称作反字)。这种情况可以用图 2.3 (a) 中的例子来说明。

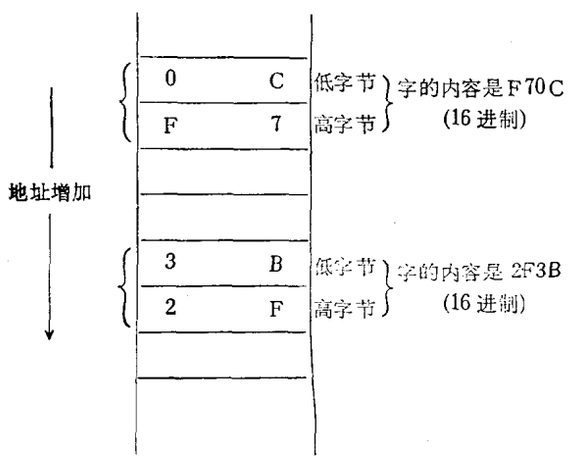

图 2.3(a) 在存贮器中"反字"存放的例子

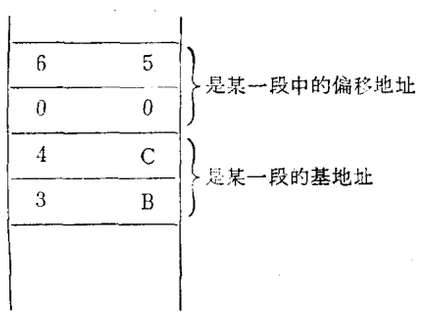

图 2.3(b) 一个指示器的例子

在 8086 中还有一类特殊的数据，它们是按双字 (即相邻的两个字) 进行存贮的，通常被称为指示器。它主要用于指示当前可寻址段以外的数据和代码的地址。指示器的较低地址字是