8086 初阶

体系结构、系统设计和程序设计入门

〔美〕S. P. 莫尔斯 著

高志伟 译

科学出版社

1984

内 容 简 介

本书是掌握8086有关知识的入门书. 全书共分六章: 第一章简单介绍了有关计算机和微型计算机的普及知识; 第二、三章详细阐述了8086的寄存器和存储器结构以及8086的指令系统; 第四章介绍了如何用8086及其配套器件进行系统设计; 第五、六章以充分的例证分别介绍了如何运用ASM-86汇编语言和PL/M 86高级语言进行程序设计. 书中的附录部分列出了8086的指令系统、操作码空间和ASCII码. 译者补充了iAPX86和88, 8089, iAPX86/10和88/10, iAPX86/20和88/20, iAPX86/30和88/30, iAPX286/10等产品的简介, 有助于读者进一步了解Intel公司16位微处理器的系列发展概貌.

本书的论述详尽、例证充分、插图丰富, 可供计算机专业人员、微型计算机使用部门、研究部门以及高等院校参考, 也适合作8086的训练教材.

Stephen P. Morse

THE 8086 PRIMER

An Introduction to Its Architecture, System Design, and Programming

Hayden Book Company, 1980

8086 初阶

体系结构、系统设计和程序设计入门

〔美〕S. P. 莫尔斯 著

高志伟 译

责任编辑 李淑兰 [illegible]明湘

科学出版社出版

北京朝阳门内大街137号

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1984年6月第一版 开本: 850×1168 1/32

1984年6月第一次印刷 印张: 10 3/8

印数: 0001—25,300 字数: 263,000

统一书号: 15031·571

本社书号: 3561·15-8

定价: 1.95元

译 者 序

微处理器的发展史，已经越过了十个年头．十年来，微处理器无论在品种和数量上都有了飞速的发展．各种报刊杂志上介绍各类微处理器的文章屡见不鲜，有关微处理器应用的专题论文比比皆是．

国内外应用微处理器之所以会得到蓬勃发展，固然是因为微处理器在价格和性能上所具备的优越性．从价格上来讲，利用微处理器代替电子电路中大量其他类型的元器件，会直接使产品的成本下降．另一方面，因印制电路、机壳、电源的缩小，以及测试和维修手段的简化，又会间接使产品的成本下降．而且，利用微处理器进行硬件设计也比较简单，因而设计的投资比较少，加上硬件设计和软件设计的灵活性，又会使设计费用大为降低．从性能上来讲，微处理器的应用为产品性能的改进开辟了新的途径，这是因为元器件数量的减少，就会自然而然地使产品可靠性提高，而且微处理器所具备的特殊智能，则是各种传统元器件所无法比拟的．

美国 Intel 公司是世界上最先推出微处理器的厂家．他们生产的 8080A，很快就与 Motorola 公司的 MC6800 和 Zilog 公司的 Z80 一起成为最通用的 8 位微处理器．Intel 公司又于 1978 年最先推出 16 位通用微处理器 8086，与后来 Motorola 公司的 MC68000 和 Zilog 公司的 Z8000，成为举世瞩目的三种 16 位机．关于这三种片子的优劣，许多人曾发表过各种高见，众说纷纭，不一而足．比较一致的意见，似乎是从计算机的角度出发，认定 MC68000 为佼佼者．然而，随着近几年的发展，8086 取得了很大的进展，使目前的竞争集中于 8086 和 MC68000 之间．下面我们简单对比一下 8086 和 MC68000 的优缺点以及它们的发展趋势，使读者对两者都有一个大致的概念．

目　录

第一章 绪 论

本章的目的是从技术上和发展史的角度，对微型计算机进行一般的介绍，而重点涉及8086. 微型计算机除了在其规模方面特殊之外，其余与任何计算机别无二致. 基于这点，我们将从概述计算机基础知识入手，进而阐述微型计算机的整个发展过程. 最后我们还要说明：什么是8086引人注目之处.

1.1 计算机概论

我们在讲述微型计算机之前，首先要简明扼要地归纳一下计算机的概貌. 本节除了作一般概述之外，还要介绍全书都将使用的各种术语和概念.

图1.1示出构成计算机系统的各个基本单元，图1.2示出相同的系统，只是所有能辨认的部分均用非人格化的方框代替. 下

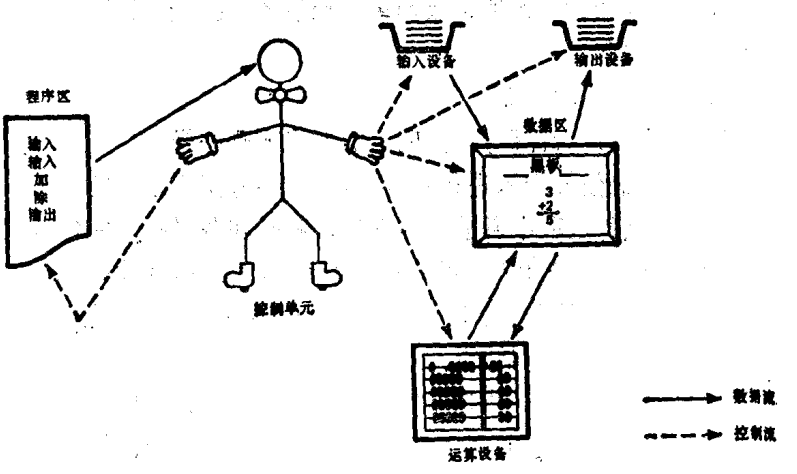

图1.1 原始的计算机系统

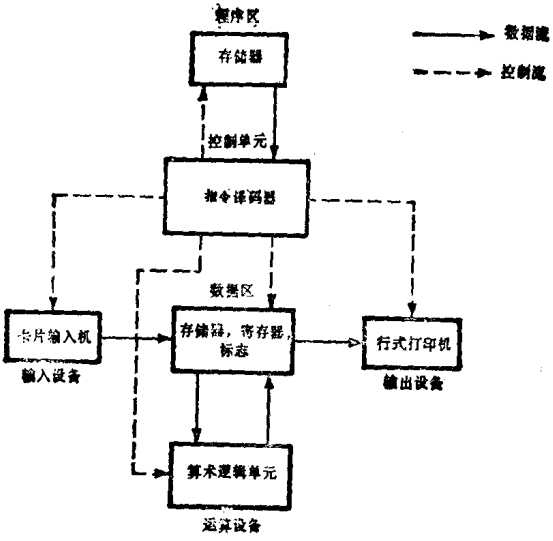

图1.2 现代的通用计算机系统

面我们来重点阐述每个方框的功能，从而考察这种系统的性能.

计算机的作用，乃是从**输入设备**获取数据，然后处理这些数据，再将最后的结果提供给**输出设备**. 由一系列指令规定的某项处理内容就叫做**程序**. 程序存储在程序区内.

计算机的操作是由所谓**控制单元**控制的，控制单元则反复执行如下三个步骤：

1. 从程序区取指令；

2. 对指令进行译码，以便确定需要完成哪些操作；

3. 将控制信号送入进行这些操作的设备，实现指令的执行.

在指令执行过程中所进行的操作，就是在各设备之间传送数据，并且在某一设备中对这些数据进行计算. 这些计算是由**运算设备**完成的. **数据区**用来提供这些计算所需要的输入，并且保存这些计算所产生的中间结果.

为了弄清所有这一切是如何联系在一起的，让我们分析一下某种指令执行的情形，这里我们采用的是“加法”指令. 控制单元将控制信号送入程序区，再请求下一条指令. 程序区作出响应，将

指令送入控制单元，由控制单元将指令译码，判断出这是“加法”指令，然后把控制信号送到如下地方：(1) 数据区，令其将两个数值传送到运算设备；(2) 运算设备，令其将收到的两个数值相加；(3) 数据区，令其接受相加后的结果。

程序区和数据区的相似之处，就在于两者都由存储器组成，它们都能够保存信息。然而，在每个区内保存的信息，种类上有很大差别。数据区保存的是中间结果，在程序执行过程中，这些中间结果经常改变。程序区保存的是程序，而程序在执行过程中通常是不会改变的。（近年来，那些能够自行修改的程序已经不吃香了。）在某些系统中，程序实际上是“铭刻”在存储器内的，这样一来就不再改变了，只能进行读出。具有这种特点的存储器叫做只读存储器（缩写为 ROM）。显然，ROM 不适于用作数据区。数据区是由既可以读又可以写的存储器组成的，这种存储器因偶然的关系被叫做 RAM，实际上应当叫做 RWM。（RAM 乃是随机存储器的缩写，可惜这个名称并不能将其概念解释清楚。）

存储器乃是一些按顺序排列的存储单元集合体，每个存储单元都有一个独一无二的地址。每个存储单元均由一系列码元组成（这里“码元”就是“比特”，是“二进制数”的缩写），这些码元就是存储单元的内容，每个码元不是 0 就是 1。本章后面将要更多地谈到二进制数。

除了存储器之外，数据区还由一些寄存器和标志组成。寄存器与存储器相似，也用来保存中间结果。在寄存器内存取数值，通常要比在存储器内更容易，也更快。计算机把标志当作跟踪执行情况的指示器。标志有两种类型：(1) 状态标志，用来记录有关前面执行过的指令所产生的结果信息；(2) 控制标志，用来控制计算机的操作。状态标志的例子之一，就是指示出计算机处理的结果太大。控制标志的例子之一，就是通知计算机以较慢的速率（例如每小时一个指令）去执行指令。

计算系统中的另一个设备就是端口。所谓端口，乃是让信息在输入输出设备之间来回通过的一个门户。为了简单起见，图

1.1 和图 1.2 未曾示出端口。

1.2 数 据 格 式

存储单元的内容既能代表程序中的指令，又能代表一段数据。以存储单元内码元序列形式所存储的各种指令方式就叫做指令格式，并且可以因计算机而异。8086 的指令格式将在第三章加以介绍，这里论述 8086 使用的数据格式。

计算机所处理的数据，既可以是数字式的(数值)，也可以是非数字式的(字符)。工资单程序可以广泛使用数字式数据，而文本编辑程序则多半采用非数字式数据。存储非数字式数据的格式，就是大家熟悉的 ASCII 码。

数　系

我们习惯于用十进制数列来表示数值，比如 365，代表有 3 个百，6 个十和 5 个一。这种方式有时叫做以 10 为基数的表示法，这样做绝非偶然，因为我们都有十个手指头，所以我们就用以 10 为基数的表示法来数我们的数。计算机并没有手指头，它们靠电压来计数。为了可靠起见，它们只采用两个电压电平，亦即要么有电压，要么没有，从而计算机就很难(虽然并非完全不可能)将这两种情形搞错。这就是说，计算机希望用一系列二进制数(比特)来表示数，这就是以 2 为基数的表示法，比如 11010，表示有 1 个十六，1 个八，0 个四，1 个二，0 个一。二进制数可以直接进行加减乘除，而不需要首先将其变换为十进制数，只需要记住 1 加 1 等于 10 (1 个二和 0 个一)而不是 2。举例来说

```
  1 0 0 1    九的二进制表示法
+ 0 1 0 1    五的二进制表示法
-----------
  1 1 1 0    十四的二进制表示法
```

对于一些长的二进制数列来说，尽管计算机并不会将最小位数搞混，然而我们却容易弄错。举例来说，10110101 是一百八十

表1.1 十六进制表示法

4位分组	十六进制数字	数 值
0 0 0 0	0	零
0 0 0 1	1	一
0 0 1 0	2	二
0 0 1 1	3	三
0 1 0 0	4	四
0 1 0 1	5	五
0 1 1 0	6	六
0 1 1 1	7	七
1 0 0 0	8	八
1 0 0 1	9	九
1 0 1 0	A	十
1 0 1 1	B	十一
1 1 0 0	C	十二
1 1 0 1	D	十三
1 1 1 0	E	十四
1 1 1 1	F	十五

一的二进制表示。为了简单起见，我们设想一种方案，按每次4位一组，将长的二进制数列缩短。每4位一组，用一个字符来表示，如表1.1所示。于是，10110101就缩减成为B5，这就是所谓十六进制数，要是我们生下来就有16个手指头的话，那么我们早就会使用这种数系了。

带符号位的数

二进制记数法，最适于描述正数和零。然而，当我们想要估计负数的时候，我们就需要一种辅助的手法，来指出数的正负号。完成这项任务的最简单途径，就是采用该数的最高有效位(即最左边一位)，来指示正负号。举例来说

```
0 0 0 0  0 1 0 0    应为 +4
1 0 0 0  0 1 0 0    应为 -4
0 1 1 1  1 1 1 1    应为 +127
1 1 1 1  1 1 1 1    应为 -127
```

这样一种表示法，就叫做**符号与数值**表示法，这种方法有一个严重的缺点：它要求一套新的算术规则．当我们打算采用二进制算术来做 0 减去 +1 并且希望得到 −1 时，这个缺点就变得非常明显了．

```
 0 0 0 0  0 0 0 0    用带符号位的数值表示的 0
-0 0 0 0  0 0 0 1    用带符号位的数值表示的 +1
-----------------
 1 1 1 1  1 1 1 1    用带符号位的数值表示的 −127
```

如果我们想要在带符号位的数上，采用我们已在不带符号位的数上采用过的相同二进制算术，那么我们就需要采用一种带符号位的数表示法，使得 1111 1111 表示成为 −1 而不是 −127．而且，−1 减去 +1 应当得出 −2．我们来进行这种减法，以便看看 −2 究竟应当是怎么一回事．

```
 1 1 1 1  1 1 1 1    这里为 −1
-0 0 0 0  0 0 0 1    减去 +1
-----------------
 1 1 1 1  1 1 1 0    也可以叫做 −2
```

由此可以看出，我们应当将正负数表示如下：

```
        ⋮
0 0 0 0  0 0 1 1    +3
0 0 0 0  0 0 1 0    +2
0 0 0 0  0 0 0 1    +1
0 0 0 0  0 0 0 0     0
1 1 1 1  1 1 1 1    −1
1 1 1 1  1 1 1 0    −2
1 1 1 1  1 1 0 1    −3
        ⋮
```

这就是所谓**二进制补码**表示法，其特点是二进制的加法和减法都会得出正确的二进制补码结果．举例来说

```
 0 0 0 0  0 0 1 1    用二进制补码表示的 +3
+1 1 1 1  1 1 1 0    用二进制补码表示的 −2
-----------------
 0 0 0 0  0 0 0 1    用二进制补码表示的 +1
```

它的另一个特点是：每个非负数(正数或零)的最高有效位都是 0，而每个负数的最高有效位则为 1．这样一来，正好与带符号的数

值表示法一样，最高有效位就作为符号位．关于带符号位的数的详细情况，将在第三章加以讨论．

改变二进制补码数的每位数值，然后再加 +1，就可以改变其符号．举例来说，我们可以按下列方法，将二进制补码表示的+3，变成二进制补码表示的 −3

0 0 0 0　0 0 1 1	用二进制补码表示的 +3
1 1 1 1　1 1 0 0	每一位都改变了的 +3
+0 0 0 0　0 0 0 1	用二进制补码表示的 +1
1 1 1 1　1 1 0 1	用二进制补码表示的 −3

关于二进制补码数，有一件事值得我们谨慎从事．如果想要将一个 8 位的二进制补码数扩充到 16 位(举例来说由此可以相加成为 16 位的二进制补码)，那么我们首先就得考虑附加的 8 位该如何办．

假令我们现在欲将 0000 0001 (用二进制补码表示的 +1) 与 0000 0000 0000 0011 (用二进制补码表示的 +3) 相加，在这种情况下，毫无疑问我们只需在 +1 的左边加添 8 个 0，然后相加

0 0 0 0　0 0 0 0　0 0 0 0　0 0 1 1	用二进制补码表示的 +3
+0 0 0 0　0 0 0 0　0 0 0 0　0 0 0 1	用二进制补码表示的 +1
0 0 0 0　0 0 0 0　0 0 0 0　0 1 0 0	用二进制补码表示的 +4

然而，如果我们欲将 1111 1111(用二进制补码表示的 −1)与 0000 0000 0000 0011(用二进制补码表示的 +3)相加，那么我们就得在 −1 的左边加添 8 个 1 (加添 0 就会使之变成正数)，然后再相加

0 0 0 0　0 0 0 0　0 0 0 0　0 0 1 1	用二进制补码表示的 +3
+1 1 1 1　1 1 1 1　1 1 1 1　1 1 1 1	用二进制补码表示的 −1
0 0 0 0　0 0 0 0　0 0 0 0　0 0 1 0	用二进制补码表示的 +2

由此看来，将 8 位数扩充到 16 位数，就类似于下面的情形：

数值	8位表示法	16位表示法
+1	0 0 0 0　0 0 0 1	0 0 0 0　0 0 0 0　0 0 0 0　0 0 0 1
−1	1 1 1 1　1 1 1 1	1 1 1 1　1 1 1 1　1 1 1 1　1 1 1 1

扩充二进制补码数的规则，就是要在该数的左边加添附加的若干

位，而加添的每一位，在数值上均与原符号位相同．这样的过程就叫做符号的扩充．

字　符

各种字符均可以用二进制比特数列来表示．我们至少必须能用 26 个字母和 10 个数字来表示总共 36 个字符．但是，还应当注意，要能够区分大写和小写字母（亦即另外还有 26 个字母），同时还要能够表示一些专用的字符（如＋和＊等等）．这样一来，如今我们的字符就超过了 64 个，从而至少需要 7 位来表示一个字符（亦即一个 6 位数可以具有的最大值只是 64）．有一种通用的 7 位编码制，就是所谓 ASCII（美国信息交换标准码），请参见本书的附录 3．每一个 8 位的存储单元叫做存储器的一个字节，用来存放 ASCII 编码字符是很方便的（第 8 位有时用来检查另外 7 位的有效性）．

1.3 堆　栈

所谓堆栈，乃是微处理器和较大一些的计算机常常使用的一个概念．堆栈还有“下推表”或者“后进先出排队”等别名．之所以使用这些名称，乃是想要传达出一种好比自助餐厅盘子堆放设备的形象．每当把一个新的盘子放到一堆盘子上头的时候，就使得它下面的所有盘子都下降了一级．把堆栈顶上的一个盘子取走，所有的盘子便又升了一级．最后一个放到堆栈上的盘子，便是首先将要取走的一个．

为了便于理解计算机是如何进行上述工作的，首先我们就得介绍例行子程序．所谓例行子程序（有时又叫做过程），乃是一套程序之中可以调用来完成指定任务的各个部分，这样便提供了一种手段，将总的需要解决的问题，分解成为一些较小的且较简单的部分．例行子程序本身又可以调用其他一些例行子程序，去进一步分解所要完成的任务．一条例行子程序在完成其任务之后，便

把控制信号送回到原先调用它的例行程序中去．结果就产生了一系列例行子程序，其中每一条都可以调用其他例行子程序，直至所调用的最后一条例行子程序都返回．换句话说，调用的最后一条例行子程序，将是首先返回的例行子程序．

在调用例行子程序的时候，必须保留一定数量的信息．这可能包括某些寄存器的当前内容和标志的当前调整位．它必然包括调用例行程序之中的地址，而该例行程序又是例行子程序最后要返回控制的．每当例行子程序完成其任务之后，就会恢复这些保留的信息，从而也就可以恢复受影响的寄存器之中的内容，将标志设置在原来的位置，并且使用"返回地址"，把控制信号送回给适当的指令．然而，由于最后调用的一个例行子程序，就是第一个要恢复的例行子程序，因此所保留的最后一段信息，也就必须首先加以恢复．这样一来，也就必须如同自助餐厅堆放盘子那样，把信息也堆放起来．

到此为止，我们已经讲述了堆栈是如何工作的，以及堆栈为何在计算机中是有用的．现在我们再来看看，计算机的堆栈是如何实现的．由于堆栈必须保存信息，因此就必须是某种类型的存储器．实际上，现在所能买到的任何一种存储器（只读存储器除外），都可以用作堆栈，只是需要有一个指示器来指示存储器堆栈部分内存放的最后一段信息．这种指示器通常就叫做堆栈指针，它所指示的信息通常叫做栈顶．当一段新的信息放到堆栈上的时候（这个过程就叫做推入），堆栈指针就会更新一次，以便指示下一个存储单元，然后信息就存入该单元．当一段存储器从堆栈内重新恢复的时候（这个过程就叫做弹出），该信息就从堆栈指针所指示的那一个存储单元内得到恢复，同时堆栈指针再次更新，只是这次所指示的方向相反．

1.4 8086 存储器的利用（预备知识）

上几节业已说明：存储器可以用来保存程序（代码），存储数

据(数值或字符)，也可以用作堆栈．这样一来，对于 8086 实际上是把存储器分成代码分段、数据分段和堆栈分段，也就不足为奇了．该存储器的这些分段，将在第二章加以讨论．

1.5 微型计算机史话

前面我们已经简述了计算机的基本概念，现在让我们来谈谈计算机的历史，看看计算机是如何发展到微型计算机的．

从大型计算机到微型计算机

五十年代，一切电子设备(收音机、电视机、计算机等等)都是用体积庞大的电子管器件制作的．当时的计算机有时又称为第一代计算机，例如 IBM 650 和 IBM 704．这类计算机要安装在能容纳好几台电子设备的大房间里．五十年代末期，晶体管和其他固体器件开始取代电子管．采用这种工艺的计算机就叫做第二代计算机，例如 IBM 7090 和宝莱 B 5500．

到了六十年代，可以把许多分立元件(电阻器、电容器、晶体管等等)组合成一块复杂的电子元件，这就是所谓集成电路(缩写为 IC)．IC 乃是在一块比邮票还要小的硅片上制造的，采用类似于蜈蚣的外形结构，以便于插入某个系统之中．这种可插式的集成电路，就叫做芯片．用若干块芯片制成的计算机，就是第三代计算机，例如 IBM 360，GE 635，宝莱 B 6700．可是，集成电路工艺仍然在继续不断发展，到了七十年代初期，图 1.2 中的许多部件，已经能够一起放在单块芯片上了，这就是 Intel 公司的 4004 和 8008．这就导致出现了新的术语单片计算机．

就在这一时期，计算机不仅在体积上大幅度地缩小了，而且价格也在大大下跌．电子管计算机的售价为数百万美元，单片计算机的售价最初还要五百美元左右，经过短短数年的竞争，售价就下跌到十美元以下了．

单片计算机就是所谓微型计算机或微处理器．这两个术语尽

管有时可以互换，然而还是有所差别的．微处理器只是单块芯片，它通常包含有控制单元、算术与逻辑单元、寄存器、标志以及存储器与输入/输出设备之间的接口．至于程序存储器和数据存储器，还有输入/输出设备，通常都不在该芯片上．所谓微型计算机乃是一套完整的计算机系统，它包括一片微处理器、若干片存储器和输入/输出设备．有时整个计算机系统也都包含在一块芯片上(例如 Intel 公司的 8048)，这就是所谓单片微型计算机．

随着计算机体积的缩小，售价的降低，在一些专用系统中(例如现金收入记录机、计算器和电传打字机等等)应用它们，也就变得很经济了．计算机在交通管理红绿灯系统中的应用实例，示于图 1.3 中．在这类专用控制场合，经常可以看到微处理器的使用，这就不足为奇了．

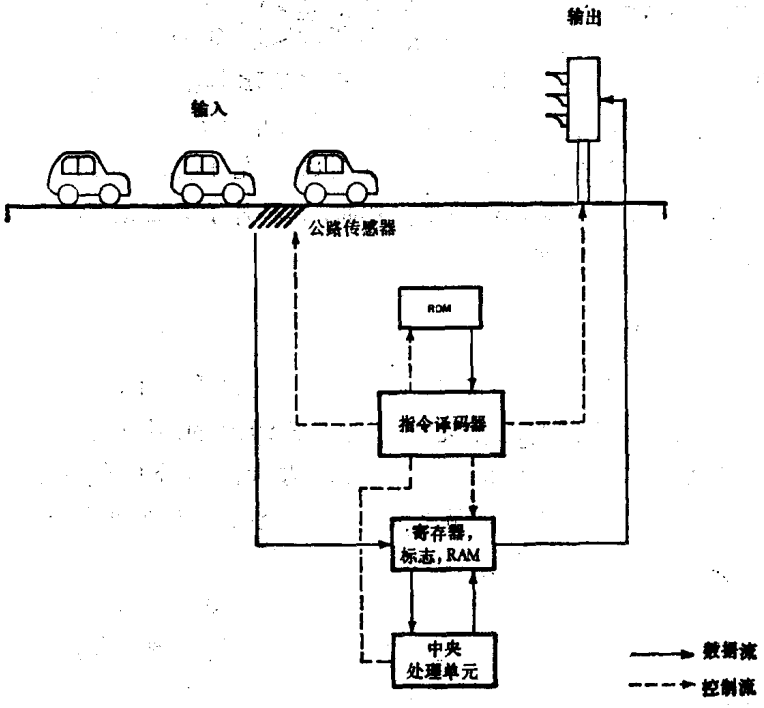

图 1.8　专用计算机系统