# 微处理机原理及应用

胡大可　主编

浙江大学出版社

# 微处理机原理及应用

胡大可　主编

浙江大学出版社

(浙)新登字 10 号

主　　编　胡大可
编写人员　(以姓氏笔划为序)
　　王芷沁　宁钢民　严筱刚　胡大可　钱鸣奇

微处理机原理及应用
责任编辑　　杜希武

浙江大学出版社出版
杭州富阳何云印刷厂印刷
浙江省新华书店经销

787×1092　16 开　13 印张　320 千字
1993 年 9 月第 1 版　1994 年 6 月第 2 次印刷
印数：2001—7000
ISBN 7-308-01293-X/TP・094　定价：7.50 元

# 前　言

微处理机技术的飞速发展和越来越广泛的应用，使得它成为科学研究和生产活动中不可缺少的重要工具，成为学习掌握其它科学技术的重要的技术基础和辅助手段。学习微处理机技术显得比以往更为重要了。本书是作为工科院校的计算机教学统一教材而编写的，同时也可以作为工程技术人员的参考书。

阅读本书的读者应具备数字电路的基础知识。

本书在编写过程中参考了目前在高等院校中比较流行的多种微处理机课程教材。根据目前微处理机技术发展的现状，根据近年来单片机应用的迅速推广，本书选用单片机 8098 作为课程讲授时的基础，同时仍然注意了在本教材中所组织的内容的普遍性。全书较完整地包容了 MCS-96 单片机系列的技术内容，也选入了比较典型的通用可编程输出输入接口，作完整的介绍。

本书强调微处理机技术中的一般概念，如：CPU 结构、指令、寻址空间、总线、读写时序、中断、可编程接口技术、监控程序等。因此对本书内容的掌握，可以成为读者进一步学习其它类型的微处理机系统的良好基础。

本书内容深入浅出，注重实用。全书充实了较多的程序和应用实例，可以帮助读者理解和掌握书中的内容。这些例子中，有不少来源于近年来的教学科研成果之中。学习本书的内容，对读者所从事的微处理机应用工作会有一定的指导和帮助意义。

本书的大纲经由浙江大学计算机课程教学指导小组的讨论和确定。书中的第一、二、九章由胡大可编写，第三、四章由王芷沁编写，第五、六章由钱鸣奇编写，第七章由严筱刚编写，第八章由宁钢民编写，全书由胡大可进行通编。全书内容的组织可分为三个部分：第一章至第六章全面阐述微处理机技术中的基本概念；第七章分四大类型介绍各种可编程输出输入接口器件；第八章和第九章通过许多实例，讨论了微处理机技术的各种应用。本书的内容在讲授时，可以根据教学的实际情况，对第二、第三部分的内容，作全部讲授或部分讲授的处理。

与本书的课堂讲授相配套的还有一组 8098 单板机实验。有关实验的指导书《微处理机原理和应用实验教程》由徐向东编写，将同时出版。

由于编者的学识水平有限，书中难免有错误和不妥之处，恳请读者批评指正。在本书的编写过程中，得到浙江大学许多有经验的微机课程教师的帮助，朱仲才同志为本书的排版输出付出了辛勤劳动，在此一并表示感谢。

编者

1993 年 7 月

# 目　　录

# 第一章　概　论

## 第一节　引　言

### 一、计算机的发展

电子计算机诞生于本世纪的 40 年代。它的出现是本世纪有重大影响的科学技术成就之一。它有力地推动了各门科学技术的发展，其应用已深入到科学技术，工农业生产，国防建设，乃至家用电器等众多领域。成为在科学研究，经济建设和社会生活中所不可缺少的重要手段。计算机的应用程度成了衡量一个国家现代化的重要尺度。

40 多年来，计算机已更新四代。第一代时间约在 1946 年至 1957 年，其主要特点是使用电子管作为逻辑元件，用延迟线及磁鼓作为存储器，软件主要是机器语言程序，符号语言刚刚开始使用。1946 年，第一台计算机 ENIAC 问世，它有 18000 个电子管，占地 150 平方米，重 30 吨，耗电 150 千瓦，加法运算速度为 5000 次/秒。这些技术指标，即使与今天的微处理机相比也是非常低的。但是，它却奠定了计算机的技术基础。例如，采用 2 进制数进行运算和控制，建立了程序设计的概念，等等。第二代出现的时间大约在 1958 年至 1964 年，其主要特点是用晶体管取代了电子管作为逻辑元件，使用了磁芯存储器，软件和硬件开始分化，软件方面出现了高级程序设计语言，如 FORTRAN、ALGOL 等，开始提出操作系统的概念。这一代计算机除了进行科学计算之外，在数据处理方面得到了广泛的应用，而且开始应用于生产过程控制。第三代出现的时间约在 1965 年至 1972 年，这一代计算机的主要特点是用中小规模集成电路取代了晶体管。由于采用了集成电路，使得计算机的体积更小，耗电更省，可靠性更高。在软件上，操作系统得到了进一步完善与发展，使计算机的使用变得更方便了。这一时期还出现小型机与大型、中型机的分化。第四代是从 1972 年到目前。主要特点是以大规模集成电路取代了中小型规模集成电路作为逻辑部件，主存储器也由磁芯存储器发展成大规模集成电路，实现了将计算机的中央处理单元集成在一块硅片上。于是出现了微处理机。这一时期在软件方面出现了非常大的发展，理论趋于完善，应用迅速推广，每天开发的软件以飞快的速度在增长。

当前计算机正在向四个方向发展：

（一）　微型化

由于大规模集成电路技术的发展，微型机的出现以及性能的不断提高已经模糊了传统的大、中、小、微型机的概念，现有的微型机性能有的超过了过去的小型机或中型机，但是体积却小得多，而且不要求严格的环境条件，价格非常低廉，大有取代中、小型机之势。微型机的出现开拓了计算机普及的新纪元。目前，集成电路集成度的提高和微处理机性能的发展势头仍非常强劲。

（二）　巨型机

当前计算机发展的另一个趋向是制造一些功能极强，运算速度极快的巨型计算机，以满

足许多尖端科学技术发展的需要。正在研究的巨型机，速度已达每秒百亿次，我国目前已能生产每秒十亿次的机型。

（三） 计算机网络

计算机网络技术是计算机发展的又一个重要方面，它能使多台计算机、数字终端、各种数据设备通过通信线路连成为一个更大的统一的系统，它们能共享其中的各种硬件及软件资源，组成一个功能非常强大的统一的整体。随着通讯技术、激光技术和光导纤维等技术的迅速发展，计算机网络将会有更大的发展。

（四） 人工智能

人工智能是计算机研究的热点之一，它以模拟人的智能和思维，如识别图形、 语言和物体，辅助设计，专家决策等作为研究目标。它的突破，将开辟计算机的发展新时代，有人称之为第五代计算机的时代。这样的计算机将能更好地延伸人的思维能力。

## 二、计算机的组成

到今天为止，计算机虽然发展到了第四代，但是就其组成而言基本上是属于冯·诺伊曼(Von Neumann) 型计算机，它们的结构都与 1946 年出现的第一台计算机大同小异，由五大部分组成。如图 1 - 1 所示。

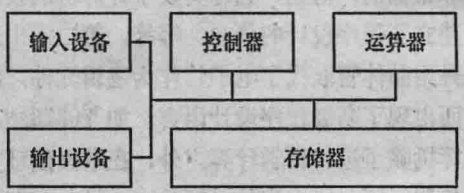


图 1 - 1 计算机组成框图

这五部分的作用简述如下：

（一） 输入设备

外界与计算机进行交往的入口，常用的输入设备有键盘、光或磁输入装置，数据输入通讯接口，等等。目前，人们正在努力开发能接受自然语言及图形文字的输入设备。

（二） 输出设备

计算机与外界交往的输出窗口，它把结果或各种信息以数字、字符、图形等形式表示出来。常用的输出设备有显示器，数码管，打印机，绘图仪，输出通讯接口，等等。

（三） 运算器

计算机中对各种信息进行运算的主要部件，由复杂的逻辑电路组成，一般包含各种寄存器，加法器，移位器，控制电路，等等。

（四） 控制器

它控制整个计算机，使得它能够自动地协调工作。控制器由指令译码电路、时序电路和逻辑电路组成，它对计算机的控制是通过解释指令代码，输出对内、对外的控制信号来实现的。

（五） 存储器

用来存放代码或数据。计算机的存储器分内、外存储器两大类别，所存放的内容，可以是参加运算的各种数据，可以是运算的结果，也可以是一些被称为指令的编码。计算机在工作时要不断从内存储器取得指令和数据，从而实现各种运算，当要存放的数据非常多时，内存储器的空间就不够大了。这时，外存储器就成为内存储器存放空间的补充和后备。常见的外存储器有软磁盘，硬磁盘，磁带和光盘等。

在上述五大组成部分中，习惯上人们把运算器和控制器看成一个整体，称为中央处理机或中央处理单元 (CPU, Central Processing Unit)，内存储器简称为内存 (MEMORY)，而把输入设备和输出设备合称为输入输出设备 (I/O)。

## 三、计算机的字长

计算机内所有的信息都以 2 进制的代码形式来表示，2 进制代码的位 (BIT) 数称为字长。计算机所处理的数，字长愈长，它能表示的数值范围就愈大，能表示的数值的有效位数愈多，计算的精度和运算速度也就愈高。但是，字长长了，用来表示 2 进制代码的逻辑电路就要增加，使得计算机的结构变得复杂而昂贵。字长短，虽然能表示的数值范围小了，数的有效位数也少了，但是这样的计算机，结构简单，造价也低廉。图 1 - 2 所示为一个字长是 8 位的代码。其中，D0 表示最低位 (LSB, Least Significant Bit)，D7 表示最高位 (MSB, Most Significant Bit)，每位数码只能取 0、1 这两个值。一个 8 位 2 进制数又称一个字节 (BYTE)。存储时地址连续的两个字节称一个字 (WORD)。存储时地址连续的两个字又称为双字 (DOUBLE WORD)。

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

图 1 - 2 代码形式

### (一) 代码内容

在计算机内一个代码所表示的内容是多种多样的，可以用它来表示一个数，也可以用它来表示一种操作，而所表示的数又可以有广泛的含义。一般说来，一个 2 进制数可以是数据代码、字符代码、指令代码或地址代码。至于一个代码表示的意义，从形式上是不能分辨的。但是在计算机内部它们是确定的。能够根据代码出现的不同场合来区分。

### (二) 数据代码

2 进制代码用来表示一个数据，要解决可表达的数值范围和数的正负这两个问题。前一个问题由采用定点数或浮点数等数据表达结构来解决。后一个问题用原码和补码等编码方法来解决。

1、无符号整数：假如计算机的字长是 1 字节，用它来表示无符号整数时，所能表示的数值范围为 0 ~ 255 。这是一个定点数，小数点定在 D0 位之后，它与 10 进制数的对应关系如表 1 - 1。数尾的字母 B 表示 2 进制。

2、有符号整数：若字长同样为 1 字节，用它表示有符号整数时，可采用定点补码制，所能表示的数值范围为 -128 ~ +127 。它与 10 进制的关系如表 1 - 2。D7 为 1 表示负数，D7 为 0 表示正数，而表示数值的位数剩下 7 位。

表 1 - 1　2 / 10 进制数对应关系
（无符号整数）

| 10 进 | 2 进 |
|---|---|
| 0 | 00000000B |
| 1 | 00000001B |
| 2 | 00000010B |
| ... | ..... |
| 253 | 11111101B |
| 254 | 11111110B |
| 255 | 11111111B |

表 1 - 2　2 / 10 进制数对应关系
（有符号整数）

| 10 进 | 2 进 |
|---|---|
| 127 | 01111111B |
| 126 | 01111110B |
| ... | ..... |
| 1 | 00000001B |
| 0 | 00000000B |
| -1 | 11111111B |
| ... | ..... |
| -127 | 10000001B |
| -128 | 10000000B |

对于 2 进制数，求取补码可以用先取反码再加 1 的方法获得，以 -30 为例。

30 的原码：　00011110B

30 的反码：　11100001B

30 的补码：　11100010B

采用补码，能够把减法运算转化为加法运算，这里通过以下例子说明。

计算（45 - 30），用 8 位无符号整数表示为：

00101101B - 00011110B = 00001111B

如果改为 45 + (-30)，用 8 位补码表示为：

00101101B + 11100010B = 100001111B

这里出现了由于进位产生的第 9 位。但是，对于一个只能存放 8 位 2 进制数的系统，第 9 位将自然丢失。这样以上两种计算方法的结果是相同的。

3、浮点数：当要表示的数很大或者要求有效位数很多时，采用定点方法就有一定的困难。常采用浮点数表示，即用编码的一部分表示阶码，一部分表示尾数。尾数表示的通常是一个小数，它反映了数的有效位或精确度。阶码是 2 的幂次，它表示的是尾数应该乘以的倍率，这个倍率总是 2 的幂。

浮点数通常是多字节的。图 1 - 3 是一种 4 字节的浮点数结构。最左边一个字节的最高位 D31 代表尾数的符号，0 为正，1 为负，D30 位代表阶码的符号，0 为正，1 为负。D29 ~ D24 位表示阶数，它是有符号整数类型，从 D23 到 D0 的 3 个字节为尾数，它的数据类型为定点小数，小数点定在 D23 之前。表 1 - 3 是浮点数表达的例子。

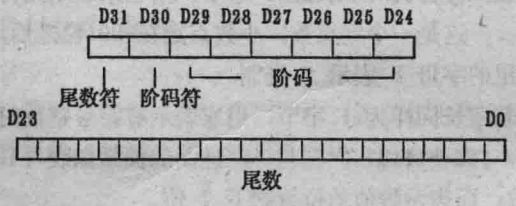


图 1 - 3　浮点数结构

表 1 - 3 浮点数表达举例

| 4 字节浮点数 | 10 进数 |
|---|---|
| 00000010 10000000 00000000 00000000B | 2.0 |
| 10000011 11000000 00000000 00000000B | -6.0 |
| 01111110 10000000 00000000 00000000B | 0.03125 |
| 00000000 00000000 00000000 00000000B | 0.0 |

浮点数的结构没有统一的规定，在不同的计算机上，甚至在不同的算法语言中，都可能采用不同的约定方式，图 1 - 3 的结构只是浮点数结构的一个例子。浮点数表示的数值范围比定点数大。在这个例子中，表达数的范围大约为 $+10^{38}$ ~ $-10^{38}$。尾数表示的小数可达到 $10^{-7}$ 的精确度。

（三）字符代码

2 进制代码也可用来表示字符。此时，它仅仅是一种编码的形式。例如计算机内常用的 ASCII 码 (American Standurd Code for Information Interchange)，它是美国国家标准信息交换代码，它是 7 位 2 进制数，可以表示 128 个字符。这些字符包括 0 ~ 9 这十个 10 进制数，26 个英文字母的大小写以及一些控制符号（见附录）。

（四）2 进制与 16 进制

假定计算机字长为 8 位，如用 2 进制来表示一个代码，就要写 8 个 0 或 1。这样不

表 1 - 4 2 / 16 进制关系

| 10 进 | 16 进 | 2 进 | 10 进 | 16 进 | 2 进 |
|---|---|---|---|---|---|
| 0 | 0H | 0000B | 8 | 8H | 1000B |
| 1 | 1H | 0001B | 9 | 9H | 1001B |
| 2 | 2H | 0010B | 10 | 0AH | 1010B |
| 3 | 3H | 0011B | 11 | 0BH | 1011B |
| 4 | 4H | 0100B | 12 | 0CH | 1100B |
| 5 | 5H | 0101B | 13 | 0DH | 1101B |
| 6 | 6H | 0110B | 14 | 0EH | 1110B |
| 7 | 7H | 0111B | 15 | 0FH | 1111B |

仅书写困难，不便阅读，而且容易出错。更方便的表达形式是 16 进制。这时只需要写两个数码。16 进制与 2 进制数的互换关系见表 1 - 4。16 进制数的每一位有 16 种状态，即 0 ~ 15，其中的 0 ~ 9 仍沿用 10 进制的数码，10 以上的数用字母 A、B、C、D、E、F 来代表。每一位 16 进制数等效于 4 位 2 进制数。例如一个 8 位 2 进制数 01000111B，可以表示成 16 进制数形式 47H，显然用 16 进制书写方便得多。

在各种进制数的后面加上一个字母，以示区别：在 2 进制数的后加一字母 B(Binary)，16 进制数后加一字母 H(Hexadecimal)。当用 16 进制表示时，如果首位是字母，则更要添加一个 0，如 0ABH、0C3H 等，以免与计算机程序中的其它字符串混淆。

## 四、指令

指令是控制计算机进行各种操作和运算的命令，它以代码的形式出现。对于不同的计算机，指令代码的编码规律是不同的，它们都有一套独有的代码指令，这套代码指令就叫做指令系统。指令从形式上看与数字代码完全一样，一个指令可以由一个字节或多个字节组成。

### （一）指令格式

指令的代码是由操作码和地址码两部分组成，操作码部分在指令里的作用是规定计算机进行的操作，而地址码部分则指明计算机进行操作的数从何处来，操作的结果放到哪里去。

### （二）指令的地址码

地址码部分可能包括三个部分，即：

| 操作码 (OP) | 第一操作数地址 (S1) | 第二操作数地址 (S2) | 终地址 (D) |
|---|---|---|---|

其中第一操作数地址是第一个数的取数地址，第二操作数地址是第二个数的取数地址，它们又称源地址，将两数运算后的结果，存到终地址内，但是许多指令不一定包含所有这些地址码，因此根据地址码的不同组成又可分成几类。

**1、三地址指令**

| OP | S1 | S2 | D |
|---|---|---|---|

这条指令的动作是对源地址 S1 和 S2 的内容进行操作，并存入终地址 D。

**2、双地址指令**

| OP | S | D |
|---|---|---|

双地址指令的操作是对源地址 S 或源地址 S 与终地址 D 的内容进行操作，结果存到终地址 D 内。

**3、单地址指令**

| OP | S 或 D |
|---|---|

单地址指令仅指出了源地址或终地址，而用约定的办法规定缺省的地址。

**4、隐地址指令**

| OP |
|---|

隐地址指令是将地址隐含入操作码中，因此这类指令非常简洁紧凑。

### （三）指令的书写形式

一条实际的指令是一个代码，可以直接用代码数来表示，也可以用更方便的助记符来表示。

1、机器码：指令代码是一组 2 进制数，可以用 2 进制代码书写，也可以用 16 进制代码书写，例如，某条指令用 2 进制形式为 01111000B、16 进制形式为 78H。2 进制或者 16 进制形式书写的指令代码，称之为机器码。机器码是指令在计算机内的表现形式，它能被计算机识别和执行。但是用机器码形式书写的指令不形象，不易阅读，有了错误也难以发

现。为了克服这些缺点，人们又改进为用符号来书写指令。

2、助记符：用英文缩写字母来表示指令将更易被人们所接受。如“加法”的英文为ADDITION，加法指令记为ADD，“减法”的英文是SUBTRACTION，减法指令记为SUB，又如“装入”的英文是LOAD，数据装入指令记为LD，等等。这种能帮助指令记忆的符号称为助记符。

这里有这样一条8098的指令：把20H、21H单元中的一个16位数和22H、23H单元中的16位数相加，和存入24H、25H单元中。如果用机器码来表示这个运算过程，为以下四个字节的代码。它们的2进制形式为：

```
01000100B
00100010B
00100000B
00100100B
```

如果用16进制来表示将更简洁一些，为：

```
44H, 22H, 20H, 24H
```

但是这两种用代码直接表示的方法都很难看明白，如果改成助记形式就成了：

```
ADD  24H, 20H, 22H
```

这样就直观一些了，但仍不够方便，于是有进一步的形式：

```
DATA1    EQU  20H
DATA2    EQU  22H
SUM      EQU  24H
ADD      SUM, DATA1, DATA2
```

这就一目了然了。首先，20H地址定义为DATA1，22H地址定义为DATA2，24H地址定义为SUM，然后将地址为DATA1与DATA2的存储单元中的数取出，相加后，和存入地址为SUM的存储单元中，前三行语句与计算机的执行过程无关，但是却使第四行计算机直接执行的语句更易读懂。从这个简单的例子中我们已经能初步看出助记符在编写程序过程中的作用。

助记符与机器码是一一对应的。显然用助记符书写指令较之机器码要形象得多，便于阅读，有错也便于发现和修改。但是，这种形式的指令计算机是不能接受的，必须将助记符转换成机器码才能让计算机识别和执行，这个变换过程称为汇编（Assemmbly）。

## 第二节　微处理机与单片微机

1972年，美国INTEL公司推出了第一个8位微处理机8008，它的基本处理单位是字长8位的2进制数。一般认为，它是微处理机的鼻祖。不久INTEL又推出了8080A，这个微处理机较为完善，在技术上影响极大。以后，由主要是从INTEL公司中分离出来的设计人员，在Motolora公司和Zilog公司先后推出了个自的8位微处理机M6800和Z80，加上INTEL的对8080A改进后的8085，成为应用非常广泛的几种8位微处理机品种。

1978年，随着INTEL推出8086，又开始了16位微处理机的时代。此后基本形成了INTEL和iAPX86系列与Motolra的M68000系列的竞争局面。并且逐步向32位、64位发展。这段时期微处理机的性能有很大提高，应用向原先由大型、小型计算机系统占据的传

统应用领域渗透。在科学技术进步的进程中，微处理机起的作用已不可取代，同时它的技术发展前景尚不可限量。

1976 年，INTEL 推出了第一个单片微计算机 8048，开始了一个单片微机的时代，单片微机在单一硅片上集成了中央处理机，内存储器及输入输出部件，仅用一片集成电路即能构成了一台完整计算机，前景十分诱人。经过一段时间的发展及完善，1983 年 INTEL 推出了第一个 16 位的单片微机 8096，它的集成度达到了 100000 个晶体管，功能非常强大。

就单片微机目前的技术水平而言，在大多数应用场合，仍需要用若干片集成电路来组成计算机系统，因此单片微机的定义是较为含糊的。INTEL 对微处理机及单片微机作了分类，一类，是主要是用于构成通用微型计算机的 CPU，称为微处理机，如 8086、80286、80386、80486 等等。它的技术发展集中在提高处理能力和速度，结构向小型、大型机的 CPU 靠拢。另一类，主要用于构成各种特定用途的系统，用于控制、检测、通讯等众多领域，这后一类一般称为单片微机。它又可分为两类，一类，如 80186、80376，它们的特点是集成度较高，使得构成系统时电路得以简化，主要用于各种专用系统；同时 80186、80376 分别与 8086、80386 在机器指令级别上兼容，从而得到可利用普通微机作开发调试之用的便利。另一类，如 8051、8096 等，以集成度非常高，片上 I/O 功能丰富，性能价格比高为特点。用它们来组成系统，所包含的芯片相当的少，有可能仅用一片即能构成系统，以满足某些应用的需要。这也就是单片机名称的来由。

INTEL 把目前所谓的单片微机称为嵌入式微计算机 (Embedded Controller)，也许是更为贴切的。因此从本质上讲，单片微机除了片上集成了较多的存储器及输入输出部件，并且一般不用来构成通用微机系统等与微处理机不同的特点外，它与微处理机在工作原理上是一致的，结构上是非常接近的。为了适合目前的习惯，本书仍沿用单片微机或单片机的叫法。

## 第 三 节　　微处理机的工作过程

计算机是怎样工作的？虽然计算机有各种各样的应用方式，可能是科学计算，可能是对一批数据进行检索，也可能被用于对一个生产过程的控制，但是从计算机系统的建立与运行的角度去看，它们的基本过程是一致的。大致可分为几个阶段：建立数学模型，设计算法、编制程序，翻译代码，装入代码，最终是计算机的运行。

### （一）　建立数学模型

凡是要用计算机处理的某个实际问题，必须先用数学方法加以表述，这可能是一组数学公式，也可能是对一批数据的组织方法。

### （二）　设计算法

如果数学模型较为简单，这个模型的计算方法是比较容易设计出来的。当模型较为复杂时，计算方法设计的合理性将对解决问题的效率有很大的影响。这个算法的设计不仅要考虑模型的数学原理，还要考虑计算机实现这些算法时的自身特点。

### （三）　编制程序

程序是计算机指令的有序集合，是把算法表达为对计算机操作的控制过程。程序是由语句组成的，而语句就是计算机指令的助记形式。

（四） 翻译代码

计算机只能接受计算机指令，又称为机器代码。以助记形式表述的程序，必须翻译成机器代码才能被计算机所接受。这个过程可以用人工方法完成，但更多是用计算机自动翻译的方法来实现，这一过程也称为编译或汇编，前者是对高级语言程序而言，后者对汇编语言程序而言。

（五） 装入代码

由翻译而得来的指令代码要装入计算机存储器，由控制器逐条取得，逐条执行，从而实现由指令表述的算法。这一装入过程有多种形式。可能是经过所谓固化的手段，把代码直接写入内存储器中；也可能是由计算机本身的程序把代码取入内存储器。这可以由人工从输入装置，例如键盘，逐一把代码输入；也可以是通过读外存储器和通讯的方法取得代码。

（六） 计算机运行

计算机程序的执行，是通过 CPU 周而复始地从存储器中逐条取得指令，解释指令和执行指令来实现的。在程序中，指令是逐条顺序安放的，因此在一般情况下指令也是顺序执行的。但是如果仅仅如此，这样的程序不可能解决复杂的问题。在程序中还有一些指令是用于判断程序执行的状态变化，并且决定程序的执行顺序是否应该改变。而一旦执行顺序发生变化，程序就会从一个新的指令地址开始取指令执行程序，正是这种程序执行顺序的变化，可以实现非常复杂、非常丰富多彩的程序功能。

# 第 二 章　　单片微机 8098 的结构

图 2 - 1 是 8098 的结构方框图。

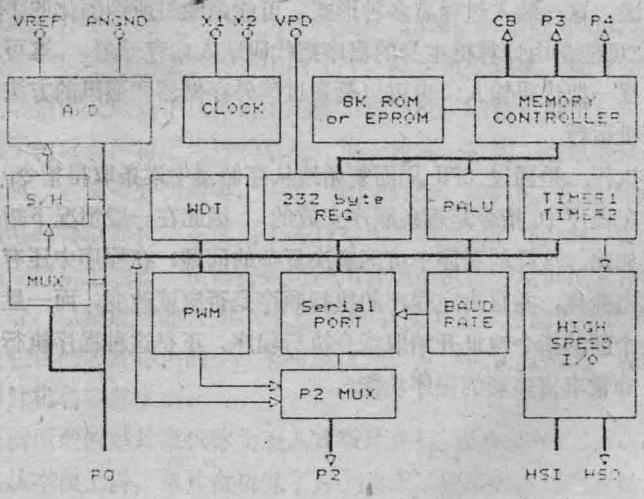


图 2 - 1　8098 结构方框图

8098 片内有时钟电路 (CLOCK)，可以接受或发生时序信号，成为所有信号的时间基准。完成运算功能的部件是寄存器算术逻辑单元 (RALU)。 8098 通过数据总线、地址总线 (AD0 ~ 7、A8 ~ 15，即 P3、P4) 和控制总线 (CB，包括 $\overline{RD}$、$\overline{WR}$、READY、ALE、$\overline{EA}$ 等) 与外围器件联接。存储器控制器 (MEMORY CONTROLLER) 是所有控制信号的产生地，控制信号中的一部分用于 8098 内部的各个部件的控制，对外形成 $\overline{RD}$、$\overline{WR}$、ALE 等信号，这些信号控制着指令和数据的流动。片内有 256 字节的读写存储器 (RAM)，其中包括 232 字节的寄存器 (REG) 和功能寄存器 (SFR)，对于 8398 和 8798 还包含有 8K 字节的只读存储器 (ROM 或 EPROM)。

8098 片内还有很多功能部件，它们是：

定时器 (TIMER1 和 TIMER2)

高速输入输出 (HIGH SPEED I/O)

串行通讯接口和波特率发生器 (Serial PORT, BAUD)

监视定时器 (WDT)

模拟输入，包括采样保持 (S/H)、模数转换 (A/D) 等

脉冲宽度调制输出 (PWM)

下文将对 8098 内部的各个功能模块分别作介绍。

8098 有 48 个引脚，图 2 - 2 是 8098 的管脚图。

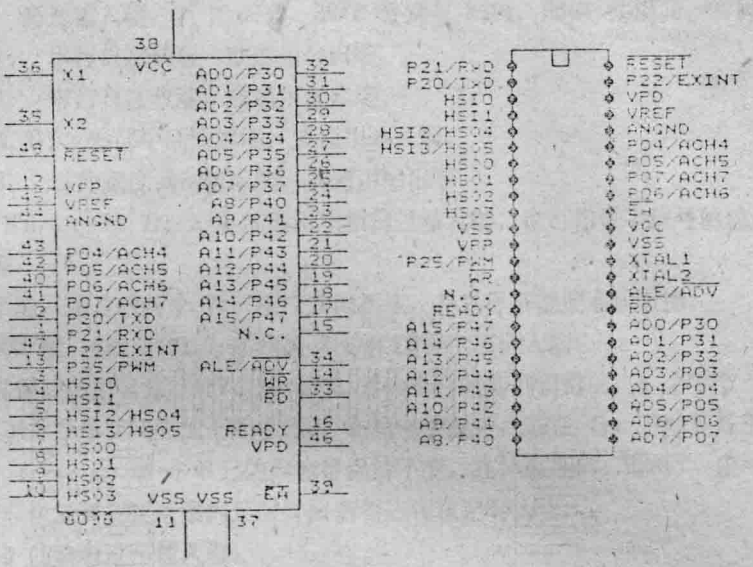


图 2 - 2　8098 管脚图和封装形式

8098 的 48 个引脚的功能分别说明如下。

AD0 ~ AD7 [P3.0 ~ P3.7]:　具有漏极开路输出的 8 位双向口。也可以用作多路复用地址/数据总线，这时必须在外部用分离电路分离地址与数据信号。它们的引脚内部具有较强的电平上拉。

A8 ~ A15 [P4.0 ~ P4.7]:　具有漏极开路输出的 8 位双向口。也用作地址总线，内部具有较强的电平上拉。

ALE / $\overline{\text{ADV}}$:　地址锁存允许 (ALE) 或地址有效输出 ($\overline{\text{ADV}}$)，由 CCR 寄存器选择。两者都提供一个锁存信号，以便把地址从 AD0 ~ AD7 中分离出来。当选定 $\overline{\text{ADV}}$ 功能时，在总线周期结束时，此引脚变高，$\overline{\text{ADV}}$ 可作为片选信号之用。

$\overline{\text{RD}}$:　片外存储器的读信号，输出低电平有效。

$\overline{\text{WR}}$:　片外存储器的写信号，输出低电平有效。

READY:　就绪信号，输入高电平有效。它用于延长对片外存储器的访问周期，以便使 8098 能够与低速存储器或动态存储器匹配；它也可用于多处理机总线共享时的控制，当进行直接存储器存取 (DMA) 时，使 8098 进入保持状态。当输入为低电平时，总线周期最多可延长至 1000 μs，当不使用片外存储器时，此信号无效。通过 CCR 寄存器可控制插入到总线周期中的等待状态数。READY 引脚内部有微弱的电平上拉，当无外部驱动时，此引脚为高电平。

$\overline{\text{EA}}$:　片外存储器存取选择，输入。当 $\overline{\text{EA}}$ 为 0，CPU 执行程序时总是从片外存储器取得指令；当 $\overline{\text{EA}}$ 为 1，CPU 访问地址为 2000H ~ 3FFFH 时转向片内。8398，8798 等型号在此地址片内有存储器，$\overline{\text{EA}}$ 决定是否要利用这些片内存储器。8098 在此地址空间片内无存储器，$\overline{\text{EA}}$ 应当接为 0。此引脚内部有下拉电路，若引脚无驱动，它总保持低电平。

$\overline{\text{RESET}}$:　复位输入端，低电平有效。复位时应至少保持 2 个状态周期的低电平，此后由低变高的跳变使时钟同步，并且产生 10 个状态周期的系统初始化序列，然后便使 8098 转向 2080H 单元执行程序。