高等学校计算机科学与技术专业教材

# 微型计算机原理及接口技术

李伯成　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
www.phei.com.cn

## 内 容 简 介

本书以 8086(88)为对象,主要介绍微型计算机的基本结构、汇编语言程序设计及基本的程序设计方法、内部存储器、接口技术及接口芯片的应用、微型计算机应用系统的可靠性分析及设计等内容。书中还对 Pentium 处理器做了简要介绍。

本书特别注意阐明基本概念、基本思路和基本方法,并着眼于工程应用。书中内容简明扼要、深入浅出,融入了作者多年的教学经验和工程应用的体会。本书可作为高校计算机、自动化等相关专业的教材使用,对一般工程技术人员来说也有较大的参考价值。



**图书在版编目(CIP)数据**

微型计算机原理及接口技术/李伯成编著.—北京:电子工业出版社,2002.11
高等学校计算机科学与技术专业教材
ISBN 7-5053-8215-2

Ⅰ.微… Ⅱ.李… Ⅲ.①微型计算机-理论-高等学校-教材②微型计算机-接口设备-高等学校-教材 Ⅳ.TP36

中国版本图书馆 CIP 数据核字(2002)第 088603 号

责任编辑:童占梅 凌毅
印 刷:北京牛山世兴印刷厂
出版发行:电子工业出版社 http://www.phei.com.cn
北京市海淀区万寿路 173 信箱 邮编 100036
经 销:各地新华书店
开 本:787×980 1/16 印张:25.75 字数:548 千字
版 次:2002 年 11 月第 1 版 2002 年 11 月第 1 次印刷
印 数:6000 册 定价:29.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。联系电话:(010)68279077

# 前　言

随着技术的发展和社会的进步，微型计算机在各行各业得到了广泛的应用。本书是为高校计算机、自动化等相关专业教学及一般科技人员学习微型计算机，掌握最基本的概念及方法而编写的。

如何学习微型计算机的相关知识，使自己很快入门是经常困扰初学者的问题。而微型计算机的发展异常迅速，使人们无所适从。对于硬件处理器来说，从通用 CPU、单片机、数字信号处理器(DSP)到专用处理器芯片，均由多个厂家研制生产，而每一种处理器又有许多系列和型号。我们认为，可以采取从特殊到一般的学习方法，即选择某一种国内比较流行的处理器(或单片机，或 DSP)，认真仔细地学好，牢牢掌握基本概念和基本方法，很快入门。只要入了门，熟悉了一种型号的微型机，再遇到其他类型的微型机将很容易掌握。这是因为它们的基本概念、基本思路是相同的，共性的东西是很多的。为此，本书以 80x86 为对象进行深入分析和描述。

学习本书开始的一段时间，会遇到大量的新的概念，而且许多是先期课程中没有涉及到的概念。因此，在开始时会感到头绪多、概念新、内容繁杂。在学习过程中采用不断循环、逐步深入的方式，即在学习后面的内容时，返回到前面章节，联系到一起，以加深理解。

本书的目的在于培养学生的工程思维能力，其内容偏重于工程应用。对于各种集成电路芯片(包括 CPU、存储器芯片、接口芯片等)，我们强调读者应抓住它们的外特性，能在工程上用好它们，便达到了目的。至于芯片内部的东西，只要最低限度地了解，满足工程上的使用即可。读者不可能也没有必要弄清楚芯片内部的细节。

书中涉及的大都是工程问题，要求读者运用所学的基本概念，提出解决问题的思路和方法。因此，许多问题可以有多种方法去解决，而不是只有一个标准答案。在学习本书时，请读者注意多练习、多实践，以便尽快掌握这种思维方法。

全书共分 7 章，前面 5 章是本书的基本内容，约需 50 学时左右的教学时间。若有更多的时间，可讲授第 6 章和第 7 章；否则可让学生自学。

在本书的编写过程中，力求突出三个主要特点：① 突出基本概念、基本思路和基本方法的描述。② 内容中融入了作者二十多年的教学及十余个科研项目的实践经验。③ 简明扼要、重点突出。

尽管作者尽到了努力，由于水平所限，加上时间紧迫，错误不当之处在所难免，敬请批评指正。

# 目　录

# 第 1 章　微处理器及系统总线

在这一章里，首先描述微型计算机的基本组成，使读者了解微型计算机的基本框图以及构成微型计算机的各部分的功能。在此基础上，仔细介绍 8088(86)微处理器以及有关系统总线的许多问题。使读者明确，从硬件系统角度来说，构成系统的其他部分就是挂接在本章所描述的、由 CPU 形成的系统总线上。

## 1.1　微型计算机的基本结构

### 1.1.1　微型计算机的组成及各部分的功能

在这里我们将粗略地介绍微型计算机的组成以及各部分的功能。基本目的在于使读者在总体上对微型计算机有一个大概的认识。至于各部分的细节，则是本书后面章节的内容。

提到微型计算机的组成，读者应立即想到它是由硬件系统和软件系统两大部分构成的。

1. 硬件系统

微型计算机硬件系统如图 1.1 所示。

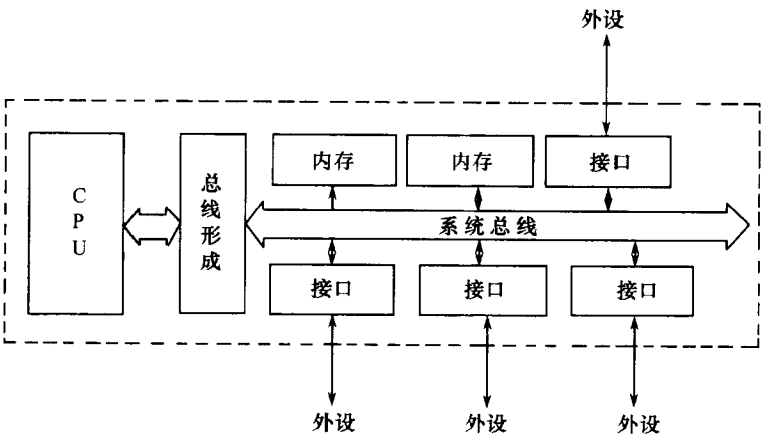


图 1.1　微型计算机的硬件结构

通常将图 1.1 中用虚线框起来的部分叫做微型计算机。若将该部分集成在一块集成电路芯片上,则就叫做单片微型计算机,简称单片机。若该部分再包括构成微型计算机所必需的外设,则就叫做微型计算机系统,实际上是指硬件系统。

微型计算机主要由如下几个部分组成:微处理器或称中央处理单元(CPU)、内部存储器(简称内存)、输入/输出接口(简称接口)及系统总线。

(1) CPU

CPU 是一个复杂的电子逻辑元件,它包含了早期计算机中的运算器、控制器及其他功能,能进行算术、逻辑及控制操作。现在经常见到的 CPU 均采用超大规模集成技术做成单片集成电路。它的结构很复杂、功能很强大。后面将仔细地对它加以说明。

(2) 内存

顾名思义,所谓内存就是指微型计算机内部的存储器。由图 1.1 可以看到,内存是直接连接在系统总线上的。因此,内存的存取速度比较快。由于内存价格较高,一般其容量较小。这与作为外设(外部设备)的外部存储器刚好相反,后者容量大而速度慢。

内存用来存放微型计算机要执行的程序及数据。在微型计算机工作过程中,CPU 从内存中取出程序执行或取出数据进行加工处理。这种由内存取出的过程称为读出,而将数据或程序存放于内存的过程就称为写入。

存储器由许多单元组成,每个单元存放一组二进制数。微型计算机中规定每个存储单元存放 8 位二进制数,8 位二进制数定义为一个字节。为了区分各个存储单元,就给每个存储单元编上不同的号码,人们把存储单元的号码叫做地址。内存的地址编号是由 0 开始的,地址顺序向下编排。例如,后面我们要介绍的 8088 CPU 的内存地址是从 00000H~FFFFFH,共 1 兆个存储单元,简称内存可达到 1 兆字节(1MB)。

如上所述,存储单元的地址一般用十六进制数表示,而每一个存储器地址中又存放着一组二进制(或用十六进制)表示的数,通常称为该地址的内容。值得注意的是,存储单元的地址和地址中的内容两者是不一样的。前者是存储单元的编号,表示存储器中的一个位置,然后者表示这个位置里存放的数据。正如一个是房间号码,另一个是房间里住的人一样。

(3) 系统总线

目前,微型计算机都采用总线结构。所谓总线就是用来传送信息的一组通信线。由图 1.1 可以看到,系统总线将构成微型机的各个部件连接到一起,实现了微型机内部各部件间的信息交换。由于这种总线在微型机内部,故也将系统总线称为内总线。

如图 1.1 所示,一般情况下,CPU 提供的信号经过总线形成电路形成系统总线。概括地说,系统总线包括地址总线、数据总线和控制总线。这些总线提供了微处理器(CPU)与存储器、输入/输出接口部件的连接线。可以认为,一台微型计算机就是以 CPU 为核心,其他部件全都"挂接"在与 CPU 相连接的系统总线上,这样的结构为组成一个微型计算机带来了方便。人们可以根据自己的需要,将规模不一的内存和接口接到系统总线上。

需要内存大，接口多时，可多接一些；需要少时，少接一些，很容易构成各种规模的微型机。

另外，微型计算机与外设（也包括其他计算机）的连接线称为外总线，也称做通信总线。它的功能就是实现计算机与计算机或计算机与其他外设的信息传送。

微型计算机工作时，通过系统总线将指令读到CPU；CPU的数据通过系统总线写入内存单元；CPU将要输出的数据经系统总线写到接口，再由接口通过外总线传送到外设；当外设有数据时，经由外总线传送到接口，再由CPU通过内总线读接口读到CPU中。

（4）接口

微型计算机广泛地应用于各个部门和领域，所连接的外部设备是各式各样的。它们不仅要求不同的电平、电流，而且要求不同速率，有时还要考虑是模拟信号，还是数字信号。同时，计算机与外部设备之间还需要询问和应答信号，用来通知外设做什么或告诉计算机外设的情况或状态。为了使计算机与外设能够联系在一起，相互匹配有条不紊地工作，就需要在计算机和外部设备之间接上一个中间部件，以便使计算机正常工作，该部件就叫做输入/输出接口。为了便于CPU对接口读写，就为接口编号，称为接口地址。8088（86）接口地址从0000H到FFFFH编址，共64KB。

在图1.1中，虚线方框内的部分构成了微型计算机，方框以外的部分称为外部世界。微型计算机与外部世界相连接的各种设备，统称外部设备。例如，键盘、打印机、显示器、磁带机、磁盘等。另外，在微型计算机的工程应用中所使用的各种开关、继电器、步进电机、A/D及D/A变换器等均可看作微型计算机的外部设备（简称外设）。通过接口部件，微型机与外设协调地工作。接口部件使用很普遍，目前已经系列化和标准化，而且有许多具有可编程序功能，使用方便、灵活，功能也非常强。根据所使用的外部设备，人们可以选择适合要求的接口部件与外设相接。

2. 软件系统

在上面的叙述中简要地说明了构成微型计算机的硬件组成部分。但任何微型计算机要正常工作，只有硬件是不够的，必须配上软件。只有软、硬件相互配合，相辅组成，微型计算机才能完成人们所期望的功能。可以这么说，硬件是系统的躯体，而软件（即各种程序的集合）是整个系统的灵魂。不配备任何软件的微型机，我们称它为物理机或裸机。它和刚诞生的婴儿一样，只是具有有限的基本功能。一个小孩将来可以成为一个伟大的科学家，也可以成为一个无所事事的人。这主要取决于他本人和社会如何对他进行灌输。也就是说，在他的脑子中给他灌输怎样的知识。与此比喻相同，一台微型机，如给它配备简单的软件，它只能做简单的工作；如给它配上功能强的软件，它就可以完成复杂的工作。

微型计算机软件系统包括系统软件和应用软件两大类。

（1）系统软件

系统软件用来对构成微型计算机的各部分硬件，如CPU、内存、各种外设进行管理和协调，使它们有条不紊高效率地工作。同时，系统软件还为其他程序的开发、调试、运行提

供一个良好的环境。

提到系统软件，首先就是操作系统。它是由厂家研制并配置在微型计算机上的。一旦微型计算机接通电源，就进入操作系统。在操作系统支持下，实现人机交互；在操作系统控制下，实现对 CPU、内存和外部设备的管理以及各种任务的调度与管理。

在操作系统平台下运行的各种高级语言、数据库系统、各种功能强大的工具软件以及本书将要涉及到的汇编语言均是系统软件的组成部分。

在操作系统及其他有关系统软件支持下，微型计算机的用户可以开发他们的应用软件。

(2) 应用软件

应用软件是针对不同应用、实现用户要求的功能软件。例如，Internet 网点上的 Web 页、各部门的 MIS 程序、CIMS 中的应用软件以及生产过程中的监测控制程序，等等。

各种应用软件根据其功能需求，在不同的软硬件平台上进行开发，可以选用不同的系统软件支持，例如不同的操作系统、不同的高级语言、不同的数据库，等等。应用软件的开发，采用软件工程的技术途径进行。

应用程序，一般都由用户自己开发完成。用户可以根据微型机应用系统的资源配备情况，确定使用何种语言来编写用户程序，既可以用高级语言也可以用汇编语言。高级语言功能强，且比较近似于人们日常生活用语习惯，因此比较容易编写；而用汇编语言编写的程序则具有执行速度快、对端口操作灵活的特点。在当前，人们通常用高级语言和汇编语言混合编程的方法来编写用户程序。

### 1.1.2 微型计算机的工作过程

如前所述，微型机在硬件和软件相互配合之下才能工作。如果我们仔细注意微型计算机的工作过程就会发现，微型机为完成某种任务，总是将任务分解成一系列的基本动作，然后再一个一个地去完成每一个基本动作。当这一任务所有的基本动作都完成时，整个任务也就完成了。这是计算机工作的基本思路。

CPU 进行简单的算术运算或逻辑运算，或从存储器取数，将数据存放于存储器，或由接口取数或向接口送数，这些都是一些基本动作，也称为 CPU 的操作。

尽管 CPU 的每一种基本操作都很简单，但几百、几千、几万、几十万甚至更多的基本操作组合在一起，就可以完成某种非常复杂的任务。可以说，现代的计算机可以完成我们所想到的任何工作，而最终就是通过一系列的简单操作来实现的。

通知微处理器进行某种操作的代码叫作指令。前面已经提到，微处理器只认识由 0 和 1 电平组成的二进制编码，其他什么都不认识，因此，指令就是一组由 0 和 1 构成的数字编码。微处理器在任何一个时刻只能进行一种操作。为了完成某种任务，就需把任务分解成若干基本操作，明确完成任务的基本操作的先后顺序，然后用计算机可以认识的指令来编排完成任务的操作顺序。计算机按照事先编好的操作步骤，每一步操作都由特定

的指令来指定，一步接一步地进行工作，从而达到预期的目的。这种完成某种任务的一组指令就称为程序，计算机的工作就是执行程序。

下面通过一个简单程序的执行过程，对微型计算机的工作过程做简要介绍。随着本书的讲述，对计算机的工作原理将逐步得到深入理解。

用微型计算机求解“7 + 10 = ?”这样一个极为简单的问题，必须利用指令告诉计算机该做的每一个步骤，先做什么，后做什么。具体步骤就是：

7→AL

AL + 10→AL

其含义就是把 7 这个数送到 AL 里面，然后将 AL 中的 7 和 10 相加，把要获得的结果存放在 AL 里。把它们变成计算机能够直接识别并执行的程序如下：

10110000
00000111　第一条指令

00000100
00001010　第二条指令

11110100　第三条指令

也就是说，上面的问题用 3 条指令即可解决。这些指令均用二进制编码来表示，微型机可以直接识别和执行。因此，人们常将这种用二进制编码表示的、CPU 能直接识别并执行的指令称为机器代码或机器语言。但直接用这种二进制代码编程序会给程序设计人员带来很大的不便。因为它们不好记忆，不直观，容易出错，而且出了错也不易修改。

为了克服机器代码带来的不便，人们用缩写的英文字母来表示指令，它们既易理解又好记忆。我们把这种缩写的英文字母叫做助记符。利用助记符加上操作数来表示指令就方便得多了。上面的程序可写成：

```
MOV AL,7
ADD AL,10
HLT
```

程序中第一条指令将 7 放在 AL 中；第二条指令将 AL 中 7 加上 10 并将相加之和放在 AL 中；第三条指令是停机指令。当顺序执行完上述指令时，AL 中就存放着要求的结果。

微型计算机在工作之前，必须将用机器代码表示的程序存放在内存的某一区域里。微型机执行程序时，通过总线首先将第一条指令取进微处理器并执行它，然后取第二条指令，执行第二条指令，依次类推。计算机就是这样按照事先编排的顺序，依次执行指令。这里要再次强调，计算机只能识别机器代码，它不认识助记符。因此，助记符编写的程序必须转换为机器代码才能为计算机所直接识别。有关这方面的知识，我们将在下面的章节中说明。

## 1.2 8088(86) CPU

在本节中将仔细介绍 8088(86) CPU 的外部引线和它们的某些必须知道的内部寄存器。

### 1.2.1 概述

8088(86) CPU 较同时代的其他微处理器具有更高的性能,在制造过程中采取了一些特殊的技术措施。

1. 设置指令预取队列(指令队列缓冲器)

我们可以形象地想像 8088(86) CPU 集成了两种功能单元:总线接口单元(BIU)和指令执行单元(EU)。前者只管不断地从内存将指令读到 CPU 中,而后者只管执行读来的指令。两者可以同时进行,并行工作。

为此,在 8088 CPU 中设置了一个 4 个字节的指令预取队列(8086 CPU 中的指令预取队列为 6 个字节)。EU 要执行的指令是由 BIU 从内存取出先放在队列中,而 EU 从队列中取出指令执行。一旦 BIU 发现队列中空出两个字节以上的位置,它就会从内存中取指令代码放到预取队列中,从而提高了 CPU 执行指令的速度。

2. 设立地址段寄存器

8088(86) CPU 内部的地址线只有 16 位,因此能够由 ALU 提供的最大地址空间只能为 64KB。为了扩大它们的地址宽度,人们将存储器的空间分成若干段,每段为 64KB。另外,在微处理器中还设立一些段寄存器,用来存放段的起始地址(16 位)。8088(86)微处理器实际地址是由段地址和 CPU 提供的 16 位偏移地址,按一定规律相加而形成的 20 位地址($A_0$~$A_{19}$),从而使 8088(86)微处理器的地址空间扩大到 1 MB。

3. 在结构上和指令设置方面支持多微处理器系统

众所周知,利用 8088(86)的指令系统进行复杂的运算,如多字节的浮点运算,超越函数的运算等,往往是很费时间的。为了弥补这一缺陷,人们开发了专门用于浮点运算的协处理器 8087。将 8088(86)和 8087 结合起来,就可以组成运算速度很高的处理单元。为此,8088(86)在结构上和指令方面都已考虑了能与 8087 相连接的措施。

另一方面,为了能用 8088(86)微处理器构成一个共享总线的多微处理器系统结构,以提高微型计算机的性能,同样在微处理器的结构上和指令系统方面也做了统一考虑。

总之,8088(86)微处理器不仅将微处理器的内部寄存器扩充至 16 位,从而使寻址能

力和算术逻辑运算能力有了进一步提高，而且由于采取了上述一些措施，使微处理器的综合性能与 8 位微处理器相比，有了明显的提高。

### 1.2.2　8088 CPU 引线及其功能

8088 CPU 是一块具有 40 条引出线的集成电路芯片，其各引出线的定义如图 1.2 所示。为了减少芯片的引线，有许多引线具有双重定义和功能，采用分时复用方式工作，即在不同时刻，这些引线上的信号是不相同的。同时，8088 CPU 上有 MN/$\overline{MX}$输入引线，用以决定 8088 CPU 工作在哪种模式之下。当 MN/$\overline{MX}$ = 1 时，8088 CPU 工作在最小模式之下。此时，构成的微型机中只包括一个 8088 CPU，且系统总线由 CPU 的引线形成，微型机所用的芯片最少。当 MN/$\overline{MX}$ = 0 时，8088 CPU 工作在最大模式之下。在此模式下，构成的微型计算机中除了有 8088 CPU 之外，还可以接另外的 CPU（如 8087），构成多微处理器系统。同时，这时的系统总线要由 8088 CPU 的引线和总线控制器（8288）共同形成，可以构成更大规模的系统。

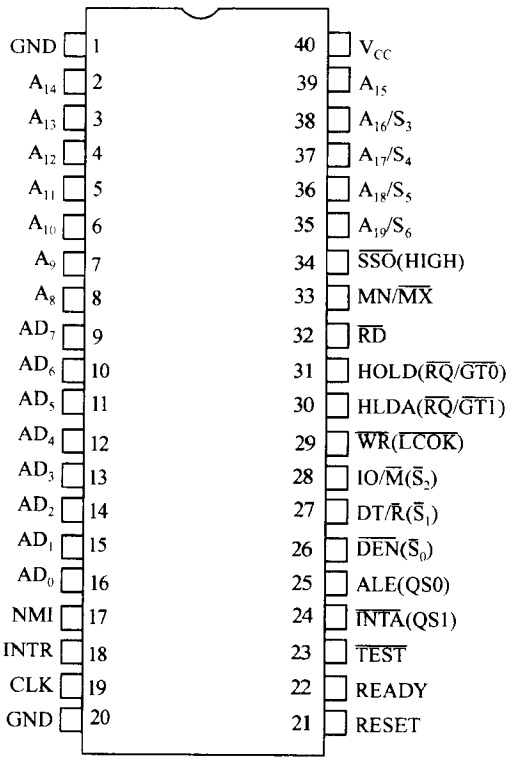


图 1.2　8088 微处理器引线图

1. 最小模式下的引线

在最小模式下，8088 CPU 的引线如图 1.2 所示（不包括括号内的信号）。它们是：

$A_{16}$～$A_{19}$/$S_3$～$S_6$：这是 4 条时间复用、三态输出的引线。在 8088 CPU 执行指令过程中，某一时刻从这 4 条线上送出地址的最高 4 位——$A_{16}$～$A_{19}$。而在另外时刻，这 4 条线送出状态 $S_3$～$S_6$。这些状态信息里，$S_6$ 始终为低，$S_5$ 指示状态寄存器中的中断允许标志的状态，它在每个时钟周期开始时被更新，$S_4$ 和 $S_3$ 用来指示 CPU 现在正在使用的段寄存器，其信息编码如表 1.1 所示。

**表 1.1　$S_4$、$S_3$ 的状态编码**

| $S_4$ | $S_3$ | 所代表段寄存器 |
|---|---|---|
| 0 | 0 | 数据段寄存器 |
| 0 | 1 | 堆栈段寄存器 |
| 1 | 0 | 代码段寄存器或不使用 |
| 1 | 1 | 附加段寄存器 |

在 CPU 进行输入输出操作时，不使用这 4 位地址，故在送出地址的时间里，这 4 条线

的输出均为低电平。

在一些特殊情况下(如复位或 DMA 操作时),这 4 条线还可以处于高阻(或浮空、或三态)状态。

$A_8$～$A_{15}$: 它们是三态输出引线。在 CPU 寻址内存或接口时,由这些引线送出地址 $A_8$～$A_{15}$。在某种特殊情况下,这些引线也可以处于高阻状态。

$AD_0$～$AD_7$: 它们是地址、数据时分复用的输入输出信号线,其信号是经三态门输出的。由于 8088 微处理器只有 40 条引脚,而它的数据线为 8 位,地址线为 20 位,因此引线数不能满足信号输入/输出的要求。于是在 CPU 内部就采用时分多路开关,将低 8 位地址信号和 8 位数据信号综合后,通过这 8 条引脚输出(或输入)。利用定时信号来区分是数据信号还是地址信号。通常 CPU 在读写存储器和外设时,总是要先给出存储器单元的地址或外设端口的地址,然后才读写数据,因而地址和数据在时序上是有先后的。如果在 CPU 外部我们配置一个地址锁存器,把在这 8 条引脚上先出现的地址信号锁存起来,用锁存器的输出去选通存储器的单元或外设端口,那么在下一个时序间隔中,这 8 条引脚就可以作为数据线进行输入或输出操作了。

IO/$\overline{M}$: 它是 CPU 的三态输出控制信号,用来区分当前操作是访问存储器还是访问 I/O 端口。若该引脚输出为低电平,则访问存储器;若该引脚输出为高电平,则访问 I/O 端口。

$\overline{WR}$: 它是 CPU 的三态输出控制信号。该引脚输出为低电平时,表示 CPU 正处于写存储器或写 I/O 端口的状态。

DT/$\overline{R}$: 该引脚是 CPU 的三态输出控制信号,用于确定数据传送的方向。高电平为发送方向;低电平为接收方向。该信号通常用于数据总线驱动器 8286/8287 的方向控制。

$\overline{DEN}$: 这是 CPU 经三态门输出的控制信号。该信号有效时,表示数据总线上有有效的数据。它在每次访问内存或接口以及在中断响应期间有效。它常用做数据总线驱动器的片选信号。

ALE: 三态输出控制信号,高电平有效。当它有效时,表明 CPU 经其引线送出有效的地址信号。因此,它常作为锁存控制信号将 $A_0$～$A_{19}$ 锁存于地址锁存器的输出端。

$\overline{RD}$: 它是读选通三态输出信号,低电平有效。当其有效时,表示 CPU 正在进行存储器读或 I/O 读操作。

READY: 它是准备就绪输入信号,高电平有效。当 CPU 对存储器或 I/O 进行操作时,在 $T_3$ 周期开始采样 READY 信号。若其为低,表明被访问的存储器或 I/O 还未准备好数据,则应在 $T_3$ 周期以后,插入 $T_{WAIT}$周期(等待周期),然后再在 $T_{WAIT}$周期中再采样 READY 信号,直至 READY 变为有效(高电平),$T_{WAIT}$周期才可以结束,进入 $T_4$ 周期,完成数据传送。

INTR：它是可屏蔽中断请求输入信号，高电平有效。CPU 在每条指令执行的最后一个 $T$ 状态采样该信号，以决定是否进入中断响应周期。这条引脚上的请求信号，可以用软件复位内部的中断允许位而加以屏蔽。

$\overline{TEST}$：它是可用 WAIT 指令对该引脚进行测试的输入信号，低电平有效。当该信号有效时，CPU 继续执行程序；否则 CPU 就进入等待状态（空转）。这个信号在每个时钟周期的上升沿由内部电路进行同步。

NMI：它是非屏蔽中断输入信号，边沿触发，正跳变有效。这条引脚上的信号不能用软件予以屏蔽，所以由低到高的变化将使 CPU 在现行指令执行结束后就引起中断。

RESET：它是 CPU 的复位输入信号，高电平有效。为使 CPU 完成内部复位过程，该信号至少要在 4 个时钟周期内保持有效。复位后 CPU 内部寄存器的状态如表 1.2 所示，各输出引脚的状态如表 1.3 所示。表中从$\overline{DEN}/(\overline{S_0})$到$\overline{INTA}$各引脚均处于浮动状态。当 RESET 返回低电平时，CPU 将重新启动。

**表 1.2　复位后的内部寄存器状态**

| 内部寄存器 | 内　容 | 内部寄存器 | 内　容 |
|---|---|---|---|
| 状态寄存器 | 清　除 | SS 寄存器 | 0000H |
| IP | 0000H | ES 寄存器 | 0000H |
| CS 寄存器 | FFFFH | 指令队列寄存器 | 清除 |
| DS 寄存器 | 0000H | | |

**表 1.3　复位后各引脚的状态**

| 引脚名 | 状　态 | 引脚名 | 状　态 |
|---|---|---|---|
| $AD_0 \sim AD_7$ | 浮动 | $\overline{RD}$ | 输出高电平后浮动 |
| $A_8 \sim A_{15}$ | 浮动 | $\overline{INTA}$ | 输出高电平后浮动 |
| $A_{16}/S_3 \sim A_{19}/S_6$ | 浮动 | ALE | 低电平 |
| $\overline{SSO}$/HIGH | 高电平 | HLDA | 低电平 |
| $\overline{DEN}/(\overline{S_0})$ | 输出高电平后浮动 | $\overline{RQ}/\overline{GT0}$ | 高电平 |
| $DT/\overline{R}/(\overline{S_1})$ | 输出高电平后浮动 | $\overline{RQ}/\overline{GT1}$ | 高电平 |
| $IO/\overline{M}/(\overline{S_2})$ | 输出高电平后浮动 | QS0 | 低电平 |
| $\overline{WR}/(\overline{LCOK})$ | 输出高电平后浮动 | QS1 | 低电平 |

$\overline{INTA}$：它是 CPU 输出的中断响应信号，是 CPU 对外部输入的 INTR 中断请求信号的响应。在响应中断过程中，由$\overline{INTA}$引出端送出两个负脉冲，可用做外部中断源的中断向量码的读选通信号。