高等学校计算机专业教材精选·计算机原理

微型计算机原理与接口技术

李伯成 编著

清华大学出版社

高等学校计算机专业教材精选·计算机原理

微型计算机原理与接口技术

李伯成 编著

清华大学出版社
北京

内 容 简 介

本书系统讲述微型计算机组成与接口技术。全书共6章，详细介绍了构成微型计算机的主要组成部分的工作原理和在工程上的实现方法。本书在内容上强调基本概念、工程上分析和解决问题的方法，在说明一些常用的典型接口芯片的基础上，重点放在利用这些概念和方法设计常见外设的接口。

本书是为本科及高职高专院校学生编写的教学用书，也可供一般工程技术人员所参考。

图书在版编目（CIP）数据

微型计算机原理与接口技术/李伯成编著．—北京：清华大学出版社，2012.7
（高等学校计算机专业教材精选·计算机原理）
ISBN 978-7-302-27603-6

Ⅰ．①微…　Ⅱ．①李…　Ⅲ．①微型计算机—理论　②微型计算机—接口技术　Ⅳ．①TP36

中国版本图书馆CIP数据核字(2011)第272496号

责任编辑：张　民　战晓雷
封面设计：傅瑞学
责任校对：焦丽丽
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：14.5　　**字　　数**：352千字
版　　次：2012年7月第1版　　**印　　次**：2012年7月第1次印刷
印　　数：1～4000
定　　价：23.00元

产品编号：044765-01

出版说明

我国高等学校计算机教育近年来迅猛发展，应用所学计算机知识解决实际问题，已经成为当代大学生的必备能力。

时代的进步与社会的发展对高等学校计算机教育的质量提出了更高、更新的要求。现在，很多高等学校都在积极探索符合自身特点的教学模式，涌现出一大批非常优秀的精品课程。

为了适应社会的需求，满足计算机教育的发展需要，清华大学出版社在进行大量调查研究的基础上，组织编写了《高等学校计算机专业教材精选》。本套教材从全国各高校的优秀计算机教材中精挑细选了一批很有代表性且特色鲜明的计算机精品教材，把作者们对各自所授计算机课程的独特理解和先进经验推荐给全国师生。

本系列教材特点如下。

(1) 编写目的明确。本套教材主要面向广大高校的计算机专业学生，使学生通过本套教材，学习计算机科学与技术方面的基本理论和基本知识，接受应用计算机解决实际问题的基本训练。

(2) 注重编写理念。本套教材作者群为各高校相应课程的主讲，有一定经验积累，且编写思路清晰，有独特的教学思路和指导思想，其教学经验具有推广价值。本套教材中不乏各类精品课配套教材，并努力把不同学校的教学特点反映到每本教材中。

(3) 理论知识与实践相结合。本套教材贯彻从实践中来到实践中去的原则，书中的许多必须掌握的理论都将结合实例来讲，同时注重培养学生分析问题、解决问题的能力，满足社会用人要求。

(4) 易教易用，合理适当。本套教材编写时注意结合教学实际的课时数，把握教材的篇幅。同时，对一些知识点按教育部教学指导委员会的最新精神进行合理取舍与难易控制。

(5) 注重教材的立体化配套。大多数教材都将配套教师用课件、习题及其解答，学生上机实验指导、教学网站等辅助教学资源，方便教学。

随着本套教材陆续出版，我们相信它能够得到广大读者的认可和支持，为我国计算机教材建设及计算机教学水平的提高，为计算机教育事业的发展做出应有的贡献。

清华大学出版社

前　言

微型计算机已广泛应用于各行各业，促进了社会的发展和进步。作为当代的工程技术人员，必须很好地掌握微型计算机的概念与技术。本书是为本科和高职高专院校理工科专业教学及一般工程技术人员学习微型计算机而编写的专业教材。

如何学习并掌握微型计算机的有关知识，并运用所学的知识解决具体的工程问题，是本书在编写过程中所特别关注的。微型计算机技术发展异常迅速，就硬件处理器而言，有各种类型的通用 CPU、单片微型计算机、数字信号处理器(DSP)、片上系统(SOC)以及专用处理器芯片，有多个厂商的相应产品可供选择，而且，新的处理器芯片还在不断地涌现。我们认为可以采取从特殊到一般的学习方法，即选择某一种典型的处理器(CPU、单片机或 DSP)，认真学习并掌握其中的基本概念和基本方法。掌握了一种典型的处理器，再迁移到其他型号的处理器就很容易掌握。这是因为它们的基本概念、基本思路和基本方法都是相同的，共性非常强。同时，为了能在有限的时间里把基本问题描述清楚，应以比较简单的处理器作为实例来解释复杂的概念(太复杂的处理器不适宜时间较短的课堂教学)。为此，本书以 8086 为对象进行分析与描述。

在学习本书的内容时，一开始会出现许多过去没有遇到过的名词和概念，初学者会感到头绪多、概念新、内容繁杂。在学习过程中可采取不断循环、逐步深入的方法。也就是在学习后面的内容时，不断地返回到前面的章节，将前后内容联系到一起，从而加强对内容的理解。

本书的目的侧重于培养学生的工程思维能力，在描述清楚基本概念的基础上，着重解决具体的工程应用问题。要求读者能利用所学的基本概念，提出解决工程问题的思路和方法，提高分析具体工程问题和解决问题的能力。全书课堂教学可安排 50～60 学时，实验实训可安排 20～24 学时。

本书力求以简明扼要的语言，重点突出地描述清楚基本概念和基本方法。作者尽了很大努力使本书通俗易懂，以适合大中专院校、高职高专院校的学生阅读使用。同时，在内容中融入作者以往的教学和科研工作的实例与经验。尽管如此，由于作者水平及时间上的限制，错误和不当之处在所难免，敬请批评指正。

作者

2011 年 7 月

目　录

第 1 章　微处理器及 PC 系统

本章将详细介绍 8086 处理器，并以此为基础说明微型计算机的构成。同时，将描述 PC 的结构及各组成部分的功能，为本书后续内容的学习打下必要的基础。

1.1　8086(88)处理器

本节将详细介绍 8086(88) CPU 的外部引脚、该处理器的一些必须了解的内部寄存器以及 8086(88) CPU 的时序，并在此基础上说明系统总线的形成。

1.1.1　微型计算机的组成及各部分的功能

下面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从总体上对微型计算机有所了解。而关于各部分的细节则是本书后面各章的内容。

微型计算机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。

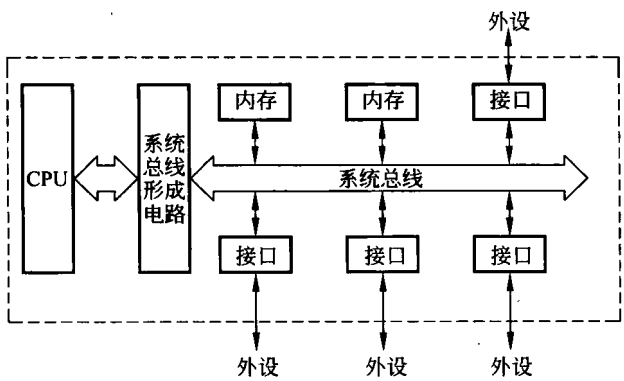

图 1.1　微型计算机结构框图

通常，将图 1.1 中用虚线框起来的部分称为微型计算机。若将该部分集成在一块集成电路芯片上，则称为单片微型计算机，简称单片机。若在该部分的基础上，再包括构成微型计算机所必需的外设，则构成了微型计算机系统，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、逻辑运算及控制操作。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。后面将详细地对它加以说明。

2）内存

内存是指微型计算机内部的存储器。从图 1.1 中可以看到，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。内存价格较高，因此其容量一般较小，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。这种由内存取出程序或数据的过程称为读出内存，而将程序或数据存放于内存的过程就称为写入内存。

存储器由许多单元组成，每个单元存放一组二进制数。8086(88)微处理器规定每个存储单元存放 8 位二进制数，将 8 位二进制数定义为一个字节。为了区分各个存储单元，为每个存储单元编上不同的号码，把存储单元的号码称为地址。内存的地址编号是从 0 开始的，按顺序向下编排。例如，后面将要介绍的 8086 CPU 的内存地址范围是 00000H～FFFFFH，共 1M 个存储单元，简称内存为 1 兆字节(1MB)。

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以二进制（或以十六进制）表示的数，通常称为该地址的内容。值得注意的是，内存单元的地址和地址中的内容两者不同。前者是存储单元的编号，表示存储器中的一个位置；而后者表示在这个位置存放的数据。可以将这两个概念比喻为一个是房间号码，另一个是房间里住的人。

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）与存储器和输入/输出接口部件的连接线。可以认为，一台微型计算机以 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；需要内存小、接口少时，则少接一些，就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。

微型计算机在工作时，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；当外设有数据输入时，经由外总线传送到接口，再由 CPU 通过内总线读接口，再读到 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种各样的。它们不仅要求不同的电平和电流，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将外设的情况或状态告诉计算机。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，称为输入/输出接口。

为了便于 CPU 对接口进行读写，就必须为接口编号，称为接口地址。8086(88)接口地址可按 0000H～FFFFH 编址，共 64K。

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，如键盘、打印机、显示器和磁盘等。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)。微型机与外设通过接口部件协调地工作。

2. 软件系统

上面简要地说明了构成微型计算机的硬件组成部分。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，相辅相成，微型计算机才能实现人们所期望的功能。可以说，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称为物理机或裸机。一台微型机，如果只给它配备简单的软件，它就只能做简单的工作；如果给它配上功能强大的软件，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、内存和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、调试和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。它是由软件厂商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。在操作系统的支持下，可以实现人机交互；在操作系统的控制下，能够实现对 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的支持下，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的应用、实现用户要求的功能软件。例如，Internet 网站上的 Web 页面、各部门的 MIS(管理信息系统)程序、CIMS(计算机集成制造系统)中的应用软件以及在由微型计算机构成的应用系统中的生产过程监测控制程序等。

1.1.2 微型计算机的工作过程

如前所述，微型计算机在硬件和软件相互配合之下才能工作。仔细观察微型计算机的工作过程就会发现，微型计算机为完成某一任务，总是将任务分解成一系列基本动作，然后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，从存储器取数或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、几十万以至更多的基本操作组合在一起就可以完成非常复杂的任务。

命令微处理器进行某种操作的代码称为指令。微处理器只能识别由 0 和 1 电平组成的二进制编码，因此，指令就是一组由 0 和 1 构成的数字编码。在微处理器的总线上，在任何一个时刻只能进行一种操作。为了完成某种任务，就需要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，然后用计算机可以识别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。

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，即先做什么，后做什么。具体步骤如下：

```
7→AL
AL+10→AL
```

其含义就是把 7 这个数送到 AL 中，然后将 AL 中的 7 和 10 相加，把要获得的结果存放在 AL 中。把它们变成计算机可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
```

即上面的问题用 3 条指令即可解决。这些指令均用二进制编码来表示，微型计算机可以直接识别和执行。因此，人们常将这种二进制编码表示的、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，而且在出错时也不易修改。

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。这种缩写的英文字母称为助记符。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 中的 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。微型计算机在执行程序时，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好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，而不认识助记符。因此，用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。有关这方面的知识将在后面的章节中

说明。

1.1.3 8086(88) CPU 的特点

8086(88) CPU 比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)。

8086(88) 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)。前者只负责不断地将指令从内存读到 CPU 中，而后者只负责执行读入 CPU 的指令。两者可以同时进行，并行工作。

为此，在 8086 CPU 中设置了一个 6B 的指令预取队列(8088 CPU 中的指令预取队列为 4B)。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，它就会从内存中读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。

8086(88) CPU 内部的地址线只有 16 位，因此，能够由 CPU 提供的最大地址空间只能为 64KB。为了扩大地址宽度，将存储器的空间分成若干段，每段为 64KB。为此，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)。8086(88)微处理器实际物理地址是由段地址和 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 8086(88)微处理器的地址空间扩大到 1MB。

(3) 在结构上和指令设置方面支持多微处理器系统。

利用 8086(88)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，当时的 CPU 设计者开发了专门用于浮点运算的协处理器 8087。将 8086(88)和 8087 结合起来，就可以组成运算速度很高的处理单元。为此，8086(88)在结构上和指令方面都已考虑了能与 8087 相连接的措施。

同时，为了能用 8086(88)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，在微处理器的结构和指令系统方面也做了统一考虑。

总之，8086(88)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。

1.1.4 8086 CPU 引脚及其功能

8086 CPU 是具有 40 条引脚(或称引出线)的集成电路芯片，其各引脚的定义如图 1.2 所示。为了减少芯片的引脚，有许多引脚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，这些引脚上的信号是不相同的。同时，8086 CPU 上有 MN/$\overline{MX}$输入引脚，用以决定 8086 CPU 工作在哪种工作模式之下。当 MN/$\overline{MX}$=1 时，8086 CPU 工作在最小模式之下。此时，构成的微型计算机中只包括一个 8086 CPU，且系统总线由 CPU 的引脚形成，微型机所用的芯片少。当 MN/ $\overline{MX}$=0 时，8086 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 8086 CPU 之外，还可以接另外的 CPU(如 8087、8089 等)，构成多微处理器系统。同时，这时的系统总线要由 8086 CPU 的引脚和总线控制器(8288)

共同形成，可以构成更大规模的系统。

1. 最小模式下的引脚

在最小模式下，8086 CPU 的引脚如图 1.2 所示（不包括括号内的信号）。下面介绍各引脚的功能。

A_{16}～A_{19}/S_3～S_6：是 4 条时间复用、三态输出的引脚。在 8086 CPU 执行指令的过程中，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。在这些状态信息中，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 和 S_3 的状态编码

S_4	S_3	编码所代表的段寄存器
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

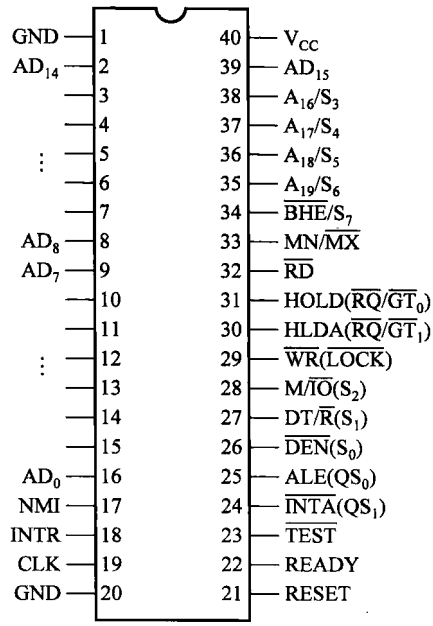

图 1.2　8086 CPU 的引脚

在 CPU 进行对接口的输入/输出操作时不使用这 4 位地址，故在送出接口地址的时间内，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（浮空或三态）状态。

AD_0～AD_{15}：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 8086 微处理器只有 40 条引脚，而它的数据线为 16 位，地址线为 20 位，因此引脚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 16 位地址信号和 16 位数据信号综合后，通过这 16 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，总是先给出存储器单元的地址或外设端口地址，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 外部配置一个地址锁存器，当这 16 条引脚出现地址信号时把地址信号锁存在锁存器中，利用锁存器的输出来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 16 条引脚就可以作为数据线进行数据的输出或输入操作了。

$M/\overline{IO}$：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，则访问的是 I/O 端口；若该引脚输出为高电平，则访问的是存储器。

$\overline{WR}$：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。

$DT/\overline{R}$：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，即 CPU 写数据到内存或接口；低电平为接收方向，即 CPU 由内存或接口读数据。该信号

通常用于数据总线驱动器 8286/8287(74245)的方向控制。

$\overline{DEN}$：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。它常用作数据总线驱动器的片选信号。

ALE：是三态输出控制信号，高电平有效。当它有效时，表明 CPU 经其引脚送出有效的地址信号。因此，它常作为锁存控制信号将 $A_0 \sim A_{19}$ 锁存于地址锁存器的输出端。

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

READY：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。若为高电平，表示存储器或 I/O 设备已准备好；若其为低电平，表明被访问的存储器或 I/O 设备还未准备好数据，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} 周期中继续采样 READY 信号，直至 READY 变为有效(高电平)，插入 T_{WAIT} 周期的过程才可以结束，进入 T_4 周期，完成数据传送。

INTR：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状态寄存器中的中断允许位(IF)而加以屏蔽。

$\overline{TEST}$：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)。该信号在每个时钟周期的上升沿由内部电路进行同步。

NMI：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复位内部状态寄存器中的中断允许位(IF)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：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 $\overline{DEN}$(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　复位后内部寄存器的状态

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
$A_{16} \sim A_{19}/S_3 \sim S_6$	浮动	ALE	低电平
$\overline{BHE}/S_7$	高电平	HLDA	低电平

续表

引脚名	状　态	引脚名	状　态
$\overline{DEN}(S_0)$	输出高电平后浮动	$\overline{RQ}/\overline{GT}_0$	高电平
$DT/\overline{R}(S_1)$	输出高电平后浮动	$\overline{RQ}/\overline{GT}_1$	高电平
$M/\overline{IO}(S_2)$	输出高电平后浮动	QS_0	低电平
$\overline{WR}$(LOCK)	输出高电平后浮动	QS_1	低电平

$\overline{INTA}$：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。在响应中断的过程中，由$\overline{INTA}$引出端送出两个负脉冲，可用作外部中断源的中断向量码的读选通信号。

HOLD：是高电平有效的输入信号，用于向CPU提出保持请求。当某一部件要占用系统总线时，可通过这条输入线向CPU提出请求。

HLDA：是CPU对HOLD请求的响应信号，是高电平有效的输出信号。当CPU收到有效的HOLD信号后，就会对其做出响应：一方面使CPU的所有三态输出的地址信号、数据信号和相应的控制信号变为高阻状态(浮动状态)；同时还输出一个有效的HLDA，表示处理器现在已放弃对总线的控制。当CPU检测到HOLD信号变低后，就立即使HLDA变低，同时恢复对总线的控制。

$\overline{BHE}/S_7$：是时间复用的三态输出信号。该信号低电平有效，用于读/写数据的高字节(D_8～D_{15})。用以保证8086可以一次读/写一个字节(高字节或低字节)或读/写一个字(16位)。

CLK：是时钟信号输入端。由它提供CPU和总线控制器的定时信号。8086 CPU的标准时钟频率为5MHz。

V_{CC}：是+5V电源输入引脚。

GND：是接地端。

2. 最大模式下的引脚

当MN/$\overline{MX}$加上低电平时，8086 CPU工作在最大模式之下。此时，除引脚24～34这几条引脚之外，其他引脚与最小模式完全相同，图1.2中括号内的信号就是最大模式下重新定义的信号。

S_2、S_1、S_0：是最大模式下由8086 CPU经三态门输出的状态信号。这些状态信号加到Intel公司提供的总线控制器8288上，可以产生系统总线所需要的各种控制信号。S_2、S_1和S_0的状态编码表示某时刻8086 CPU的状态，其编码如表1.4所示。

表1.4　S_2、S_1和S_0的状态编码

S_2	S_1	S_0	性　能	S_2	S_1	S_0	性　能
0	0	0	中断响应	1	0	0	取指
0	0	1	读I/O端口	1	0	1	读存储器
0	1	0	写I/O端口	1	1	0	写存储器
0	1	1	暂停	1	1	1	无作用

从表1.4可以看到，当8086 CPU进行不同操作时，其输出的S_2、S_1和S_0的状态是不一样的。因此，可以简单地理解为8288对这些状态进行译码，产生相应的控制信号。在本章的后面可以看到，8288总线控制器利用S_2、S_1和S_0为构成系统总线提供了足够的控制