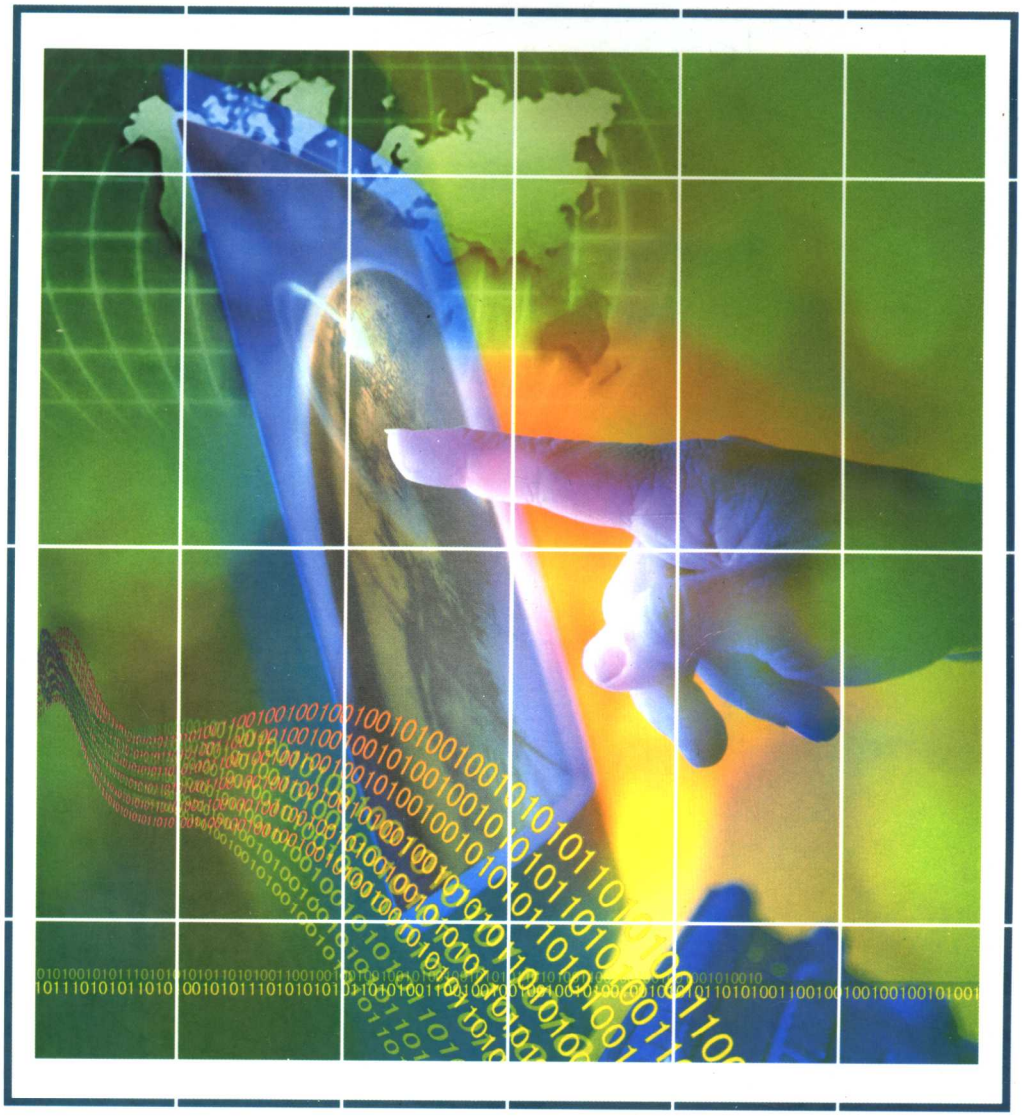
新世纪计算机类本科系列教材



微型计算机原理及接口技术

裘雪红　顾　新　侯伯亨　李伯成　编著

西安电子科技大学出版社
http://www.xduph.com

□ 新世纪计算机类本科系列教材

微型计算机原理及接口技术

裘雪红　顾　新　侯伯亨　李伯成　编著

西安电子科技大学出版社

内 容 简 介

本书以 80X86 为对象，主要介绍微型计算机的基本结构、基本工作原理、指令系统及汇编语言程序设计、内部存储器、基本 I/O 技术及典型的接口芯片，并重点介绍微机中的接口技术及其应用实例，阐明基本概念及工程实现的方法。

本书强调基本原理和概念，在此基础上着眼于微机的工程应用。通过本书的学习可为微机的应用建立坚实的知识基础。书中内容简明扼要、深入浅出，融入作者多年工程实践应用的经验及体会，便于阅读及学习。本书可作为本科生教材，同时对一般工程技术人员也具有较好的参考价值。

图书在版编目(CIP)数据

微型计算机原理及接口技术/裘雪红等著. —西安:西安电子科技大学出版社,2001.2

新世纪计算机类本科系列教材

ISBN 7-5606-0988-0

Ⅰ.微… Ⅱ.裘… Ⅲ.① 微型计算机—高等学校—教材 ②微型计算机—接口—高等学校—教材 Ⅳ.TP36

中国版本图书馆 CIP 数据核字(2001)第 00021 号

责任编辑 陈宇光 钟宏萍

出版发行 西安电子科技大学出版社(西安市太白南路 2 号)

电 话 (029)8227828 邮 编 710071

http://www.xduph.com E-mail:xdupfxb@pub.xaonline.com

经 销 新华书店

印 刷 西安文化彩印厂

版 次 2001 年 2 月第 1 版 2003 年 5 月第 4 次印刷

开 本 787 毫米×1092 毫米 1/16 印张 22.25

字 数 532 千字

印 数 16 001～24 000 册

定 价 25.00 元

ISBN 7-5606-0988-0/TP·0888

XDUP 1259001-4

* * * 如有印装问题可调换 * * *

前　言

本书是为高校师生及一般科技人员学习微型计算机的需要而编写的。

如何学习微型计算机并使自己很快入门是经常困扰初学者的问题。由于微机的发展日新月异，加之微处理器品种很多，每种微处理器本身又有许多系列，因而给学习者带来了不少困惑。对于这种现实，我们认为可以从特殊到一般进行学习，即选择比较流行的某种型号的微型机(或单片机)，认真仔细地学好，建立正确的概念。只要进了门，就容易掌握其他类型的微型机。因为，尽管微处理器型号不同，但它们之间共性的东西是很多的。为此，本书以 8086(8088)为对象，为读者做深入分析和描述，并在此基础上对更高性能的 CPU(80X86 以及 Pentium)，也做了简要介绍。对于课时较少的教学安排来说，有关 Pentium CPU 的内容可留给学生在学好本书基本内容之后进行阅读和自学，以逐步建立有关保护模式的有关概念。

在学习本书时，请读者注意这门课的一些特点。由于本书偏重于工程应用，因此，对于各种芯片(包括 CPU)，我们强调读者抓住其外部特性，以将它们用好为目的。至于芯片内部的介绍，则以工程够用为度。实际上，读者也没有必要搞清楚那些大(或超大)规模集成电路芯片的内部细节。

本书共分八章，从最基本的概念入手，引导读者逐步掌握微型机从硬件组成到软件编程的基本知识，使读者能初步掌握微型计算机组成原理和简单的应用。因此，在编写过程中力求重点突出，通俗易懂。在内容上做到简明扼要，深入浅出，便于各类人员阅读和学习。

本书第 1、2 章由李伯成编写，第 3、5 章由顾新编写，第 4、6、7 章由裘雪红编写，第 8 章由侯伯亨编写。全书由李伯成统稿。在本书编写过程中得到西安电子科技大学计算机学院的领导和出版社的关心和支持，在此表示感谢。由于水平所限，加之时间仓促，错误及不当之处在所难免，敬请读者指正。

编者

2000 年 8 月于西安电子科技大学

目　录

此为试读,需要完整PDF请访问: www.ertongbook.com

第1章　微型计算机概述

在本章中首先介绍微型计算机的概念框图，说明构成一个微型计算机的主要组成部分及各部分的功能与作用，并说明微型机的简单工作过程，以期从概念上对微型机建立初步的认识。此后，再从CPU的外部引线及内部寄存器开始，说明CPU的外特性及其工作时序。最后，叙述在此CPU的基础上，如何构成微型机的系统总线。

在本章及后面的章节中要用到一些预备知识，例如数制(二进制、十进制、十六进制等)及其运算、符号数的表示方法、常见的编码(例如BCD码、ASCII码等)以及数字电路和模拟电路的基本知识等，我们认为读者均已学习并掌握了这些知识，在本书中我们拿来就用，不再做任何解释。

1.1　微型计算机的发展史

电子数字计算机是高速度自动地进行算术和逻辑运算的电子设备，它的发明和应用标志着人类文明进入了一个新的历史阶段。可以说在人类发展史上，电子数字计算机的发明引起了一场极为深刻的工业革命。

从1946年世界上第一台计算机问世，到今天已有近60年的历史了，在这不长的时间里，计算机的发展已经历了四代。目前，各国正加紧研制、开发第五代计算机。

计算机的发展，从一开始就和电子技术，特别是半导体微电子技术和通信技术密切相关。按照构成计算机所采用的电子器件及其电路的变革，把计算机划分为若干“代”来标志计算机的发展，这已成为一种常识。

通常把以电子管及其电路为技术基础而构成的计算机称为第一代计算机(1946～1959年)，这种计算机是原始的，功能很弱。第二代(1959～1965年)是晶体管计算机时代，这一代计算机以半导体晶体管为主要元件，其性能比第一代计算机大为提高。从1965年到20世纪70年代初，数字集成电路的出现使计算机再次有了重大的进步，出现了以中、小规模集成电路为基础、并配置更完善软件的第三代计算机。70年代前期，随着大规模(LSI)、超大规模(VLSI)集成电路的诞生，电子计算机更是突飞猛进地向前发展，使计算机进入了第四代。现在人们正致力于第五代计算机的研究。

微型计算机属于第四代计算机。从1971年Intel公司生产出第一个微处理器4004开始，在短短的二十几年时间里，微处理器如雨后春笋般大量地涌现出来，出现了四代产品。

同时，其他公司也纷纷推出自己的微处理器产品及相应的配套产品。根据摩尔定律，半导体集成电路的集成度每18个月翻一番。由于集成度成倍成倍地增加，极大地促进了微处理器及其他集成电路器件(例如单片机、数字信号处理器DSP、可编程逻辑器件PLD、各

种超大规模专用集成电路等)的发展。支持这种集成电路发展的微细加工技术更是迅速发展，很快地由 90 年代中期的 0.35 μm 达到现在的 0.18 μm、0.13 μm，甚至有些实验室中已经做到线条宽度为 0.09 μm。

无疑，有了最微细的加工技术，就能在一定面积的芯片上集成更多的元件。例如，Pentium－Ⅲ的集成度已达到 3000 万个元件/片。而 DRAM 的集成度可以做到更高。同时，其功能也会更强劲。

在微处理器集成度不断提高的同时，性能也在飞速增强。从其时钟频率来说，由开始的 8086(88)的 5 MHz，增加到 Pentium－Ⅲ的 600 MHz。这还不是最快的微处理器，今天就有达到 800 MHz 和 1000 MHz 的处理器。

今天，巨型计算机的速度达到 10 万亿次(IPS)。简单的理解就是平均每秒可执行 10 万亿条指令。就 Pentium－Ⅲ来说也可轻而易举地达到 400 MIPS，平均每秒执行 400 个百万条指令(4 亿次)。

集成度愈高，芯片内部集成的功能部件就愈多，芯片内部结构也愈加复杂。因此，在学习芯片时要把其外部特性掌握好，能够熟练地连接使用它们；对于它们内部的东西，只要求最低限度的了解——够用即可。本书并不要求读者弄清芯片内部的细节。

下面将以最简单的早期的 8086(88)为例来阐述微型计算机中最基本的原理、结构及概念。有了这些最基本的东西，就可帮助读者去认识和理解更加复杂的微型计算机。

1.2 微型计算机的组成

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，其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。

1.2.1 硬件系统

微型计算机的硬件系统概念框图如图 1.1 所示。

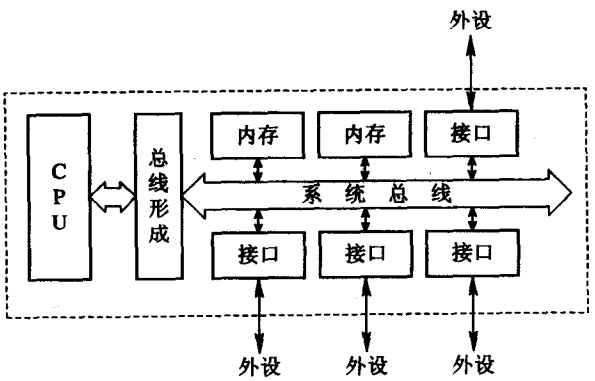


图 1.1 微型计算机硬件概念框图

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)、内部存贮器(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贮器。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)的外部存贮器刚好相反，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。

存贮器由许多单元组成，每个单元存放一组二进制数。微型计算机中规定每个存贮单元存放 8 位二进制数。8 位二进制数定义为一个字节。为了区分各个存贮单元，就给每个存贮单元编上不同的号码。人们把存贮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 兆个存贮单元，简称内存可达到 1 兆字节(1 MB)①。

如上所述，存贮单元的地址一般用十六进制数表示，而每一个存贮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，存贮单元的地址和地址中的内容两者是不一样的。前者是存贮单元的编号，表示存贮器中的一个位置，而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。顺便提及，在微型计算机外部，与外设或与其他计算机进行通信的连线称为外总线(亦称通信总线)。

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)与存贮器、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。

---

① 注：本书中用 B 表示字节(Byte)；用 b 表示二进制位(bit)

由图 1.1 可以看到，微型机在工作时，通过系统总线对存贮器的内容进行读写。同样通过系统总线，实现将数据由接口传送给外设或者将外设的数据由接口读到 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。

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。为了区分不同的接口，同样需要为接口赋以不同的地址。

1.2.2 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，只能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、运行提供了一个良好的环境。

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。在那里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。因此，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。

1.2.3 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，而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，或从存贮器取数、将数据存放于存贮器，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。

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，而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。

例如用微型计算机求解“7+10=?”这样一个极为简单的问题时，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

此为试读，需要完整PDF请访问：www.ertongbook.com

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，而后取第二条指令，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。

在本章里，读者首先要在自己的头脑中建立微型计算机的组成结构框图，大致了解各组成部分的作用。同时，还必须牢牢地记住微型计算机必须有软件(通常在物理上是看不见的)的支持才能工作。软件与硬件相互结合、相互支持才能使微型机充分发挥作用，实现更多、更强的功能。

同时，读者还必须记住微型计算机的核心部件是 CPU。它只能识别二进制编码，其他什么都不认识；它只会执行指令，其他事不会做；它执行程序，一条指令接一条指令不知疲倦地执行；执行一条指令是很快的，目前常见 CPU 可达到每秒几百万条、几千万条到几亿条指令。先行了解微型机的工作过程，对理解本书后面的章节是十分有利的。

习　题

1.1　计算机是哪年发明的？第一片微处理器是哪个公司在哪一年研制出来的？

1.2　微型计算机由哪两大部分组成，画出微型机硬件框图。

1.3　微型计算机中内存的作用是什么？有内存地址线 A_0～A_{19}共 20 条，其编成的地址有多少？

1.4　微型机中接口的作用是什么？若用 16 条地址线为接口编地址，最大可编多少个接口地址？

1.5　叙述微型机执行指令的过程。

第 2 章　8088 及其后续的 CPU

本章将详细介绍 8088 CPU，包括它的外部引线、内部寄存器、工作时序及系统总线的形成。认真理解这些内容，可为学习后面的章节打下良好的基础。同时，本章还将阐述 80X86 系列的发展情况并介绍 Pentium 100 的特点。有关 Pentium 的内容可在学完全书后再回来学习。

2.1　8088 CPU

在本节里将从 8088 CPU 的外部到它的内部进行详细地介绍。

2.1.1　概述

8088 是 8080 和 8085 的改进型，像 8080 和 8085 一样，它的指令是以字节为基础构成的。它的性能的提高，主要依赖于采取了以下一些特殊措施。

1. 建立 4 字节的指令预取队列

在以前的 8 位微处理器中，CPU 的工作过程是这样的：通过总线从存贮器中取出一条指令，然后执行该指令，如图 2.1 (*a*) 所示。在这种工作方式中，总线的利用率是很低的。例如 CPU 执行一条 INC A 指令。CPU 从存贮器中取出 INC A 指令操作码之后，在该指令执行过程中，只对内部寄存器进行操作，外部的总线是空闲的。

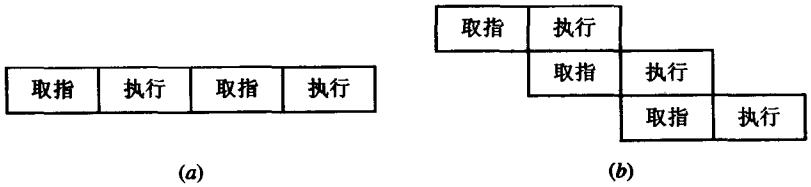


图 2.1　一般处理器与 8088 处理器指令执行过程对比

(*a*) 一般处理器指令执行过程；(*b*) 8088 处理器指令执行过程

为此在 8088 微处理器中，设置了一个 4 字节的指令预取队列，CPU 要执行的指令是从队列中取得的，而取指令的操作是由总线接口单元承担的。以此将取指令和执行指令这两个操作分别由两个独立的功能单元来完成。一旦总线接口单元发现队列中有两个字节以上的空位置时，就会自动地到存贮器中去取两个指令代码填充到指令预取队列中。这样，

8088 微处理器取指令和执行指令就可以并行进行(如图 2.1 (*b*) 所示)，从而提高了微处理器的指令执行速度，并使得总线利用率有明显的提高。

2. 设立地址段寄存器

8088 微处理器内部的地址线只有 16 位，因此能够由 ALU 提供的最大地址空间只能为 64 KB。为了扩大 8088 的地址宽度，人们将存贮器的空间分成若干段，每段为 64 KB。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 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 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 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 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 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 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 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。

2.1.2 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 2.2 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 8088CPU 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。

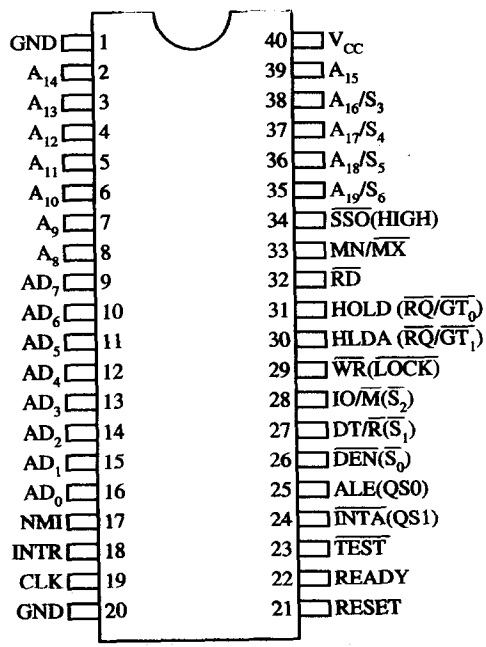


图 2.2 8088 处理器芯片引线图

1. 最小模式下的引线

在最小模式下，8088 CPU 的引线如图 2.2 所示(不包括括号内的信号)。它们是：

A_{16}～A_{19}/S_3～S_6：4 条时分复用、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 2.1 所示。

表 2.1　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}：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：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 在读写存贮器和外设时，总是要先给出存贮单元的地址或外设端口的地址，然后才读写数据，因此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贮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/$\overline{M}$：CPU 的输出(三态)控制信号，用来区分当前操作是访问存贮器还是访问 I/O 端口。若该引脚输出为低电平，则访问存贮器；若该引脚输出为高电平，则访问 I/O 端口。

$\overline{WR}$：CPU 的输出控制信号(三态)。该引脚输出为低电平时，表示 CPU 正处于写存贮器或写 I/O 端口的状态。

DT/$\overline{R}$：CPU 的输出控制信号(三态)，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。

$\overline{DEN}$：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：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。

$\overline{RD}$：读选通输出信号(三态)，低电平有效。当其有效时，表示 CPU 正在进行存贮器读或 I/O 读操作。

READY：准备就绪输入信号，高电平有效。当 CPU 对存贮器或 I/O 进行操作时，在 T_3 周期开始采样 READY 信号。若其为低，表明被访问的存贮器或 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：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。

$\overline{TEST}$：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：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：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 2.2 所示。各引脚的状态如表 2.3 所示。表中从$\overline{S_0}$/($\overline{DEN}$)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 2.2　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 2.3　复位后各引脚的状态

引　脚　名	状　态	引　脚　名	状　态
AD_0～AD_7	浮动	$\overline{RD}$	输出高电平后浮动
A_8～A_{15}	浮动	$\overline{INTA}$	输出高电平后浮动
A_{16}/S_3～A_{19}/S_6	浮动	ALE	低电平
HIGH/($\overline{SSO}$)	高电平	HLDA	低电平
$\overline{S_0}$/($\overline{DEN}$)	输出高电平后浮动	$\overline{RQ}/\overline{GT}0$	高电平
$\overline{S_1}$/(DT/$\overline{R}$)	输出高电平后浮动	$\overline{RQ}/\overline{GT}1$	高电平
$\overline{S_2}$/(IO/$\overline{M}$)	输出高电平后浮动	QS0	低电平
$\overline{LOCK}$/($\overline{WR}$)	输出高电平后浮动	QS1	高电平

$\overline{INTA}$：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，可用作外部中断向量码的读选通信号。

HOLD：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：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，就会对其作出响应：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。

$\overline{SSO}$：状态输出线。它与 IO/$\overline{M}$ 和 DT/$\overline{R}$ 信号一起决定最小模式下现行总线周期的状态。它们的不同电平所表示的处理器操作情况如表 2.4 所示。

表 2.4　IO/$\overline{M}$、DT/$\overline{R}$、$\overline{SSO}$状态编码

IO/$\overline{M}$	DT/$\overline{R}$	$\overline{SSO}$	性　能
1	0	0	中断响应
1	0	1	读 I/O 端口
1	1	0	写 I/O 端口
1	1	1	暂　停
0	0	0	取　指
0	0	1	读存贮器
0	1	0	写存贮器
0	1	1	无作用

CLK：时钟信号输入端。由它提供 CPU 和总线控制器的定时信号。8088 的标准时钟频率为 5 MHz。

V_{CC}：5 V 电源输入引脚。

GND：接地端。

2. 最大模式下的引线

当 MN/$\overline{MX}$加上低电平时，8088 CPU 工作在最大模式之下。此时，除引线 24～34 之外，其他引线与最小模式完全相同

$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$：最大模式下由 8088 CPU 经三态门输出的状态信号。这些状态信号加到 Intel 公司同时提供的总线控制器(8288)上，可以产生系统总线所需要的各种控制信号。$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$的状态编码表示某时刻 8088 CPU 的状态。其编码如表 2.5 所示。

表 2.5　$\overline{S_0}$～$\overline{S_2}$的状态编码

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	性　能
0	0	0	中断响应
0	0	1	读 I/O 端口
0	1	0	写 I/O 端口
0	1	1	暂停
1	0	0	取指
1	0	1	读存贮器
1	1	0	写存贮器
1	1	1	无作用

此为试读，需要完整PDF请访问：www.ertongbook.com