新编 16/32 位

微型计算机原理及应用

（第 2 版）

李 继 灿 主 编

李继灿 李华贵
沈疆海 郭麦成 编 著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 简 介

本书以国内外广泛使用的 16/32 位微处理器及其系统为背景，以模型机为入门，Intel 8086/8088 16 位机为基础，追踪 Intel 主流系列高性能微机的技术发展方向，比较全面、系统、深入地介绍了微机系统基础、汇编语言程序设计、存储器、输入/输出与中断、常用的可编程接口芯片和应用以及从 80286 到 Pentium 系列的技术发展，并配有多媒体 CAI 教学光盘。

本书不仅适用于从事微型计算机硬件教学与科研工作的需要，而且，对于深化计算机硬件教学的现代化改革也进行了系统而深入的探索。

本书内容先进，结构新颖，资料翔实，深入浅出，文笔流畅，便于教学与自学。它既可以作为高等院校各专业微型计算机硬件的通用教材和成人高等教育的培训教材、自学读本，也可供广大科技工作者参考。

书　　名：新编 16/32 位　微型计算机原理及应用

作　　者：李继灿　主编

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)

http://www.tup.tsinghua.edu.cn

印 刷 者：北京昌平环球印刷厂

发 行 者：新华书店总店北京发行所

开　　本：787×1092 1/16　**印张**：23.25　**字数**：533千字

版　　次：2001 年 8 月第 2 版　2001 年 8 月第 1 次印刷

书　　号：ISBN 7-900631-98-4

印　　数：00001～10000

定　　价：36.00(含盘)

前　言

自从 1997 年 7 月《新编 16/32 位微型计算机原理及应用》第 1 版由清华大学出版社出版以来，已连续出版发行 5 次，受到全国许多高等院校与广大读者的欢迎和使用。

根据作者对几年来使用该书实践经验的总结，并汲取了使用该书的高校师生与广大读者的意见和建议，特别是由于计算机硬件技术的飞速发展，我们在清华大学出版社的大力支持与帮助下，集中了一批有丰富教学经验与多媒体 CAI 开发实践经验的老师，以弹性优化组合与动态跟踪模式，用 8 个月的时间，日以继夜地连续奋战，对教材进行了迅速地、全面地修订与更新。本书是《新编 16/32 位微型计算机原理及应用》教材的全新修订版。其主要特点如下：

1. 删除了原书中一些重复、陈旧或繁杂的内容。对过渡性的 80386/80486 32 位微机系统及其汇编语言编程等内容也作了精简，使教材篇幅大为精练，便于教师教学做到“少而精”。

2. 进一步优化了教材组织结构。在保持“以模型机为入门，Intel 8086/8088 16 位机为基础，追踪 Intel 主流系列高性能微机的技术发展方向”这一基本结构特色的基础上，对中断子程序设计、中断技术与接口应用技术等内容都作了必要的调整与归纳，使结构更加紧凑与合理，便于教师组织教学和学生自学。

3. 及时增加与扩充了 Pentium 系列微处理器及微机系统的最新技术和实用知识，对目前流行的 32 位微机的总线结构，微机系统的配置与性能指标，硬盘技术、光盘驱动器技术、新型的存储器，PentiumⅡ / PentiumⅢ / PentiumⅣ以及 Itanium(安腾)微处理器及其系统等，都作了十分精练的介绍与解析，这对于提高微机硬件教学质量和增强大学生对新技术的适应性非常必要和有用。

4. 本书的一大特色，是配套了《微型计算机原理 CAI》教学光盘。该多媒体教学软件是作者多年教学经验的结晶，它以超文本、超链接的形式，集文字、声音、图形、图像于一体，并以大量生动、直观的动画演示和精心设计的友好界面，来辅助复杂抽象的计算机教学，对教师的教学会起到事半功倍的效果。

5. 经过多次反复认真地交叉校对、互审与统稿，最大限度地消除了原书中的一些疏漏与错误；同时，还为每章增配了习题，提高了教材的可读性，便于教师组织教学与考核。

《新编 16/32 位微型计算机原理及应用》第 1 版由李继灿(主编)与李华贵合作编著。本书是该书的全新修订版，由李继灿、李华贵、沈疆海、郭麦成合作编著与修订。全书共分 8 章。第 1、2 章介绍了微型计算机的公共基础知识与运算基础。第 3 章详细介绍了 8086/8088 微处理器及其系统。第 4 章介绍了 8086/8088 汇编语言程序设计。第 5 章除详细介绍了半导体存储器以外，还简要介绍了磁表面存储器、光盘存储器以及一些新型的存储器。第 6 章为输入/输出与中断，重点介绍了接口的基本概念，8086/8088 的中断系

统及其处理方法。第 7 章为可编程接口芯片及应用，主要介绍 Intel 系列的典型接口芯片及其应用技术。最后的第 8 章为从 80286 到 Pentium 系列的技术发展。其中，Pentium 微处理器系列及相关技术的发展，动态跟踪了目前微处理器及其系统的最新技术发展方向。

本书由李继灿教授策划并任主编，负责全书的大纲拟定、编著与统稿。李华贵教授任副主编，负责部分章节及习题的编著与统稿。郭麦成副教授与沈疆海同志分别负责第 4 章与第 5 章的编著与修改，并参与了程序与文图的校核。

本书编写过程中，始终受到清华大学出版社的大力支持与宝贵帮助，特别是蔡鸿程编审与李幼哲编审多年来一直关注并指导本书质量的不断提升，使本书得以及时更新与改版；此外，柯玉杞同志为本书的排版付出了辛勤的劳动，李爱珺同志为本书的终校、分类与总汇作了非常认真、细致的工作。没有他们的帮助，本书在短期内连续几次出版与发行是难以想象的。在此，本人谨表示最诚挚的感谢。

由于编者水平有限，加之时间仓促，书中难免存在一些不足与疏漏之处，恳请读者提出宝贵意见、建议和指正。

李继灿

2001 年 3 月 17 日

第 1 章　微机系统导论

电子计算机是 20 世纪最新科技成就之一。自从 1946 年第一台电子计算机问世以来，随着计算机逻辑元件的不断更新，它已经历了电子管、晶体管、集成电路以及大规模、超大规模集成电路四代发展时期。微型计算机（简称微机）是第四代电子计算机向微型化方向发展的一个非常重要的分支。

本章首先从总体上说明微机系统组成的基本概念，并对硬件系统和软件系统两大部分的具体组成予以简要介绍；然后，重点讨论典型的单总线微机硬件系统结构，微处理器组织及各部分的作用，存储器组织及其读写操作过程；在此基础上，将微处理器和存储器结合起来组成一个最简单的微机模型，通过具体例子说明微机的运行机理与工作过程；最后，给出评价微机系统性能的主要性能指标。

1.1　微机系统组成

一、几个基本定义

微处理器、微型计算机和微型计算机系统，这是三个含义不同但又有着密切依存关系的基本概念。

（一）微处理器

微处理器简称 μP 或 MP（Microprocessor），是指由一片或几片大规模集成电路组成的具有运算器和控制器功能的中央处理器部件，又称为微处理机。它本身并不等于微型计算机，而只是其中央处理器。有时为区别大、中、小型中央处理器 CPU（Central Processing Unit）与微处理器，而称后者为 MPU（Microprocessing Unit）。通常，在微型计算机中直接用 CPU 表示微处理器。

（二）微型计算机

微型计算机（Microcomputer），简称 μC 或 MC，是指以微处理器为核心，配上存储器、输入/输出接口电路及系统总线所组成的计算机（又称主机或微电脑）。当把微处理器、存储器和输入/输出接口电路统一组装在一块或多块电路板上或集成在单片芯片上，则分别称之为单板、多板或单片微型计算机。

（三）微型计算机系统

微型计算机系统（Microcomputer system），简称 μCS 或 MCS，是指以微型计算机为

中心，配以相应的外围设备、电源和辅助电路（统称硬件）以及指挥微型计算机工作的系统软件所构成的系统。

以上三者的含义及相互关系如图 1.1 所示。

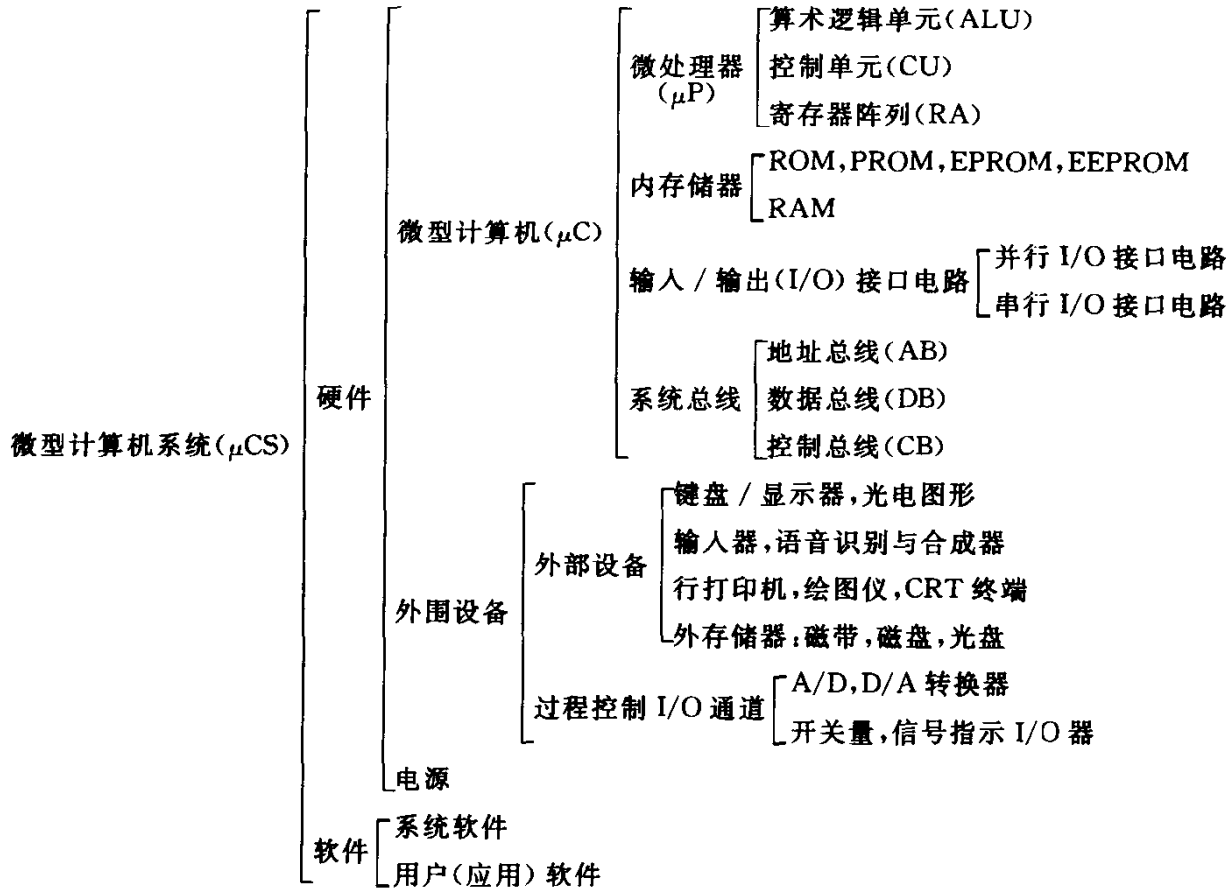


图 1.1　μCS,μC,μP 的相互关系

二、微型计算机系统的组成

微型计算机系统与任何其他计算机系统一样，由硬件和软件两个主要部分组成。

（一）硬件

微机硬件系统的组成如图 1.2 所示。图中，微处理器是微机的运算、控制中心，用来实现算术、逻辑运算，并对全机进行控制。存储器（简称主存或内存）用来存储程序或数据。输入/输出（I/O）芯片是微机与输入输出设备之间的接口。

（二）软件

计算机软件通常分为两大类：系统软件和用户软件。系统软件是指不需要用户干预的能生成、准备和执行其他程序所需的一组程序。用户软件是各用户为解题或实现检测与实时控制等不同任务所编制的应用程序。究竟应配置多少系统软件才能满足特定计算机系统的需要，这取决于具体的用途。程序的分级结构如图 1.3 所示。

操作系统是一套复杂的系统程序，用于提供人机接口和管理、调度计算机的所有硬件与软件资源。它所包含的系统程序的具体分类尚不完全统一。其中，最为重要的核心

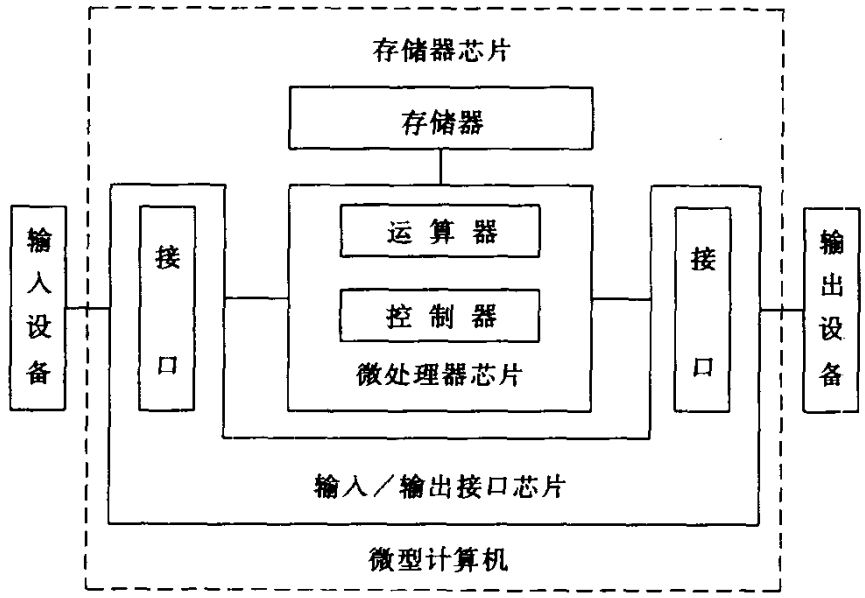


图 1.2　微机硬件系统组成

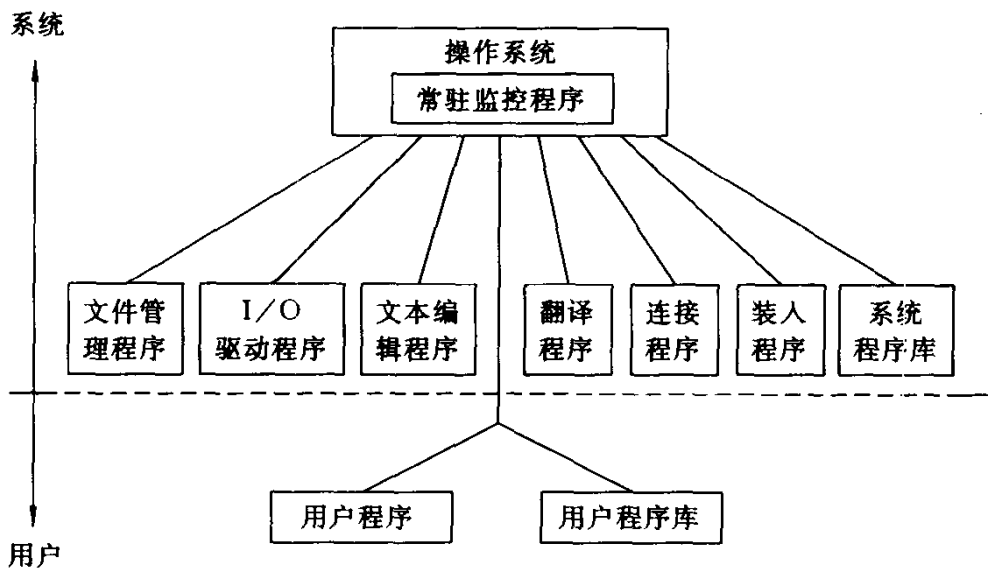


图 1.3　软件的分级结构

部分是常驻监控程序。计算机开机后，常驻监控程序始终存放在内存中，它通过接收用户命令，并启动操作系统执行相应的操作。

操作系统还包括 I/O 驱动程序和文件管理程序。前者用于执行 I/O 操作；后者用于管理存放在外存（或海量存储器）中的大量数据集合。每当用户程序或其他系统程序需要使用 I/O 设备时，通常并不是由该程序执行操作，而是由操作系统利用 I/O 驱动程序来执行任务。文件管理程序与 I/O 驱动程序配合使用，用于文件的存取、复制和其他处理。此外，系统软件还可包括各种高级语言翻译程序、汇编程序、文本编辑程序以及辅助编写其他程序的程序。程序设计分为 3 级：

（1）机器语言程序设计；

（2）汇编语言程序设计；

（3）高级语言程序设计。

机器语言程序是计算机能理解和直接执行的程序。汇编语言程序是用助记符语言表

示的程序，计算机不能直接“识别”，需经过称之为汇编程序的翻译把它转换为机器语言方能执行。机器语言指令与汇编语言指令基本上一一对应，都面向机器。而高级语言是不依赖于具体机型只面向过程的程序设计语言，由它所编写的程序，需经过编译程序或解释程序的翻译方能执行。

文本编辑程序是供输入或修改文本(字母、数字和标点等组成的一组字符或代码序列)用的程序，存于海量存储器中；它有几种用途，主要可用来生成程序。

在编写程序时，还可能需要另外两种系统程序：系统程序库；连接程序与装入程序。一般操作系统都有一个通用的系统程序库，用户还可以建立自己的程序库(一组子程序)。程序库中的子程序可附在任何系统程序或用户程序上以供调用。把待执行的程序与程序库及其他已翻译好的程序连接起来所用的准备程序称为连接程序或连接编辑程序；另一种准备程序是用来把待执行的程序送入内存，称为装入程序。有时，连接与装入功能可合成为一个程序。

应当指出，硬件系统和软件系统是相辅相成的，共同构成微型计算机系统，缺一不可。现代的计算机硬件系统和软件系统之间的分界线并不明显，总的趋势是两者统一融合，在发展上互相促进。

人是通过软件系统与硬件系统发生关系的。通常，由人使用程序设计语言编制应用程序，在系统软件的干预下使用硬件系统。

1.2 微机硬件系统结构

所谓微机硬件系统结构是指按照总体布局的设计要求将各部件构成某个系统的连接方式。一种典型的微机硬件系统结构如图 1.4 所示。图中，用系统总线将各个部件连接起来。

系统总线是用来传送信息的公共导线，它们可以是带状的扁平电缆线，也可以是印刷电路板上的一层极薄的金属连线。所有的信息都通过总线传送。通常，根据所传送信息的内容与作用不同，可将系统总线分为 3 类：数据总线 DB(Data Bus)，地址总线 AB(Address Bus)，控制总线 CB(Control Bus)。系统中各部件均挂在总线上，所以，有时也将这种系统结构称为面向系统的总线结构。

在微型计算机中有两股信息流(数据信息流和控制信息流)在流动。在总线结构中，通过总线实现微处理器、存储器和所有 I/O 设备之间的信息交换。

采用总线结构时，系统中各部件均挂在总线上，可以使微机系统的结构比较简单，易于维护，并具有更大的灵活性和更好的可扩展性。

根据总线结构组织方式的不同，目前采用的总线结构可分为单总线、双总线和双重总线 3 类，如图 1.5 所示。

图 1.5(a)所示的是单总线结构。上面介绍的图 1.4 实际上就是这种结构。在单总线结构中，系统存储器 M 和 I/O 接口均使用同一组信息通路，因此，CPU 对 M 的读/写和对 I/O 接口的输入/输出操作只能分时进行。目前大部分中低档微机都采用这种结构，因为它的结构简单，成本低廉。

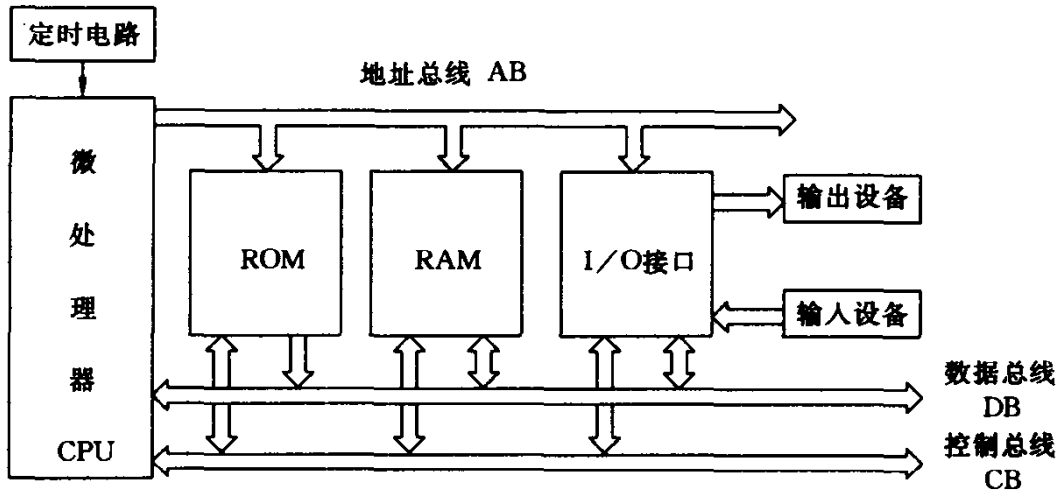
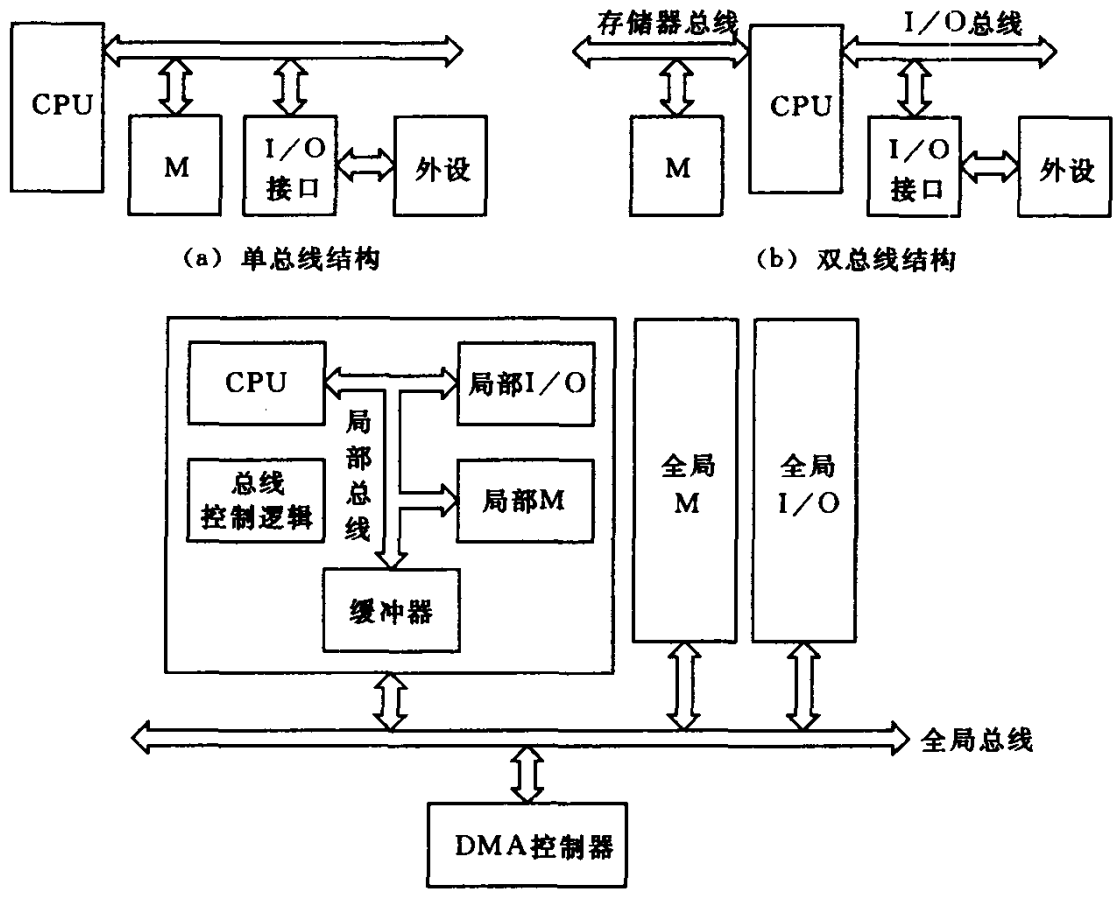


图 1.4　典型的微机硬件系统结构



(a) 单总线结构　　(b) 双总线结构

(c) 双重总线结构

图 1.5　微机的 3 种总线结构

图 1.5(b)所示的是双总线结构。这种结构的 M 和 I/O 接口各具有一组连通 CPU 的总线，故 CPU 可以分别在两组总线上同时与 M 和 I/O 交换信息，因而拓宽了总线带宽，提高了总线的数据传输效率。目前的单片机和高档微机即采用这种结构。由于双总线结构中的 CPU 要同时管理 M 和 I/O 的通信，故加重了 CPU 的负担。为此，现在通常采用专门的处理芯片即所谓的智能 I/O 接口来负责 I/O 的管理任务，以减轻 CPU 的负担。

图 1.5(c)所示的是双重总线结构。它有局部总线与全局总线这双重总线。当 CPU

通过局部总线访问局部 M 和局部 I/O 时，其工作方式与单总线的情况相同。当系统中某微处理器需要对全局 M 和全局 I/O 访问时，则必须由总线控制逻辑统一安排才能进行，这时该微处理器就是系统的主控设备。比如，当 DMA(直接存储器存取)控制器作为系统的主控设备时，则全局 M 和全局 I/O 之间便可通过系统总线进行 DMA 操作；与此同时，CPU 还可以通过局部总线对局部 M 和局部 I/O 进行访问。这样，整个系统便可在双重总线上实现并行操作，从而提高了系统数据处理和数据传输的效率。目前各种高档微机和工作站基本上都采用这种双重总线结构。

1.3 微处理器组成

图 1.6 给出了一个简化的微处理器结构。由图中可知，微处理器由运算器、控制器和内部寄存器阵列 3 部分组成。现将各部件的功能简述如下。

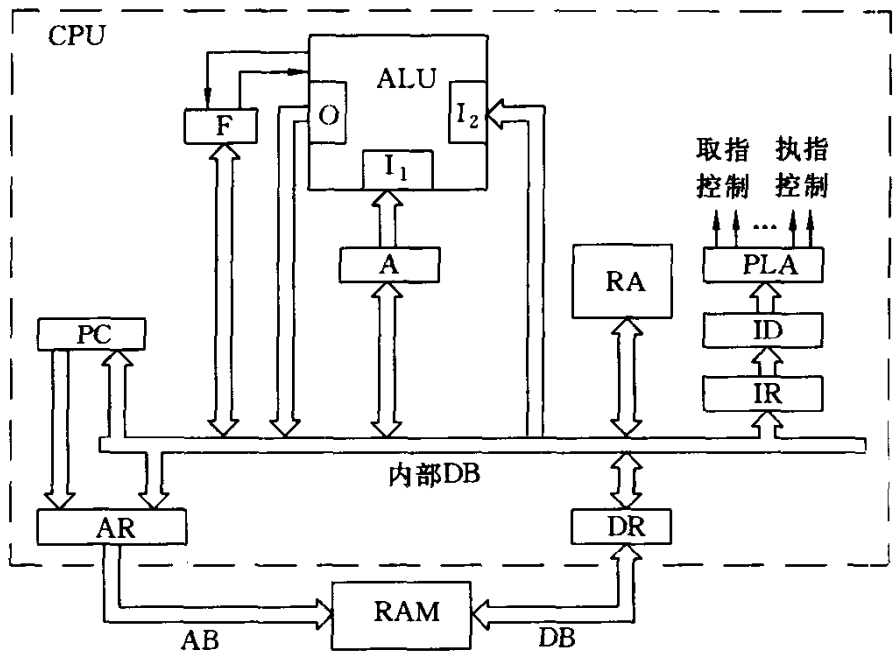


图 1.6 微处理器结构

一、运算器

运算器又称为算术逻辑单元 ALU(Arithmetic Logic Unit)，用来进行算术或逻辑运算以及位移循环等操作。参加运算的两个操作数，通常，一个来自累加器 A(Accumulator)，另一个来自内部数据总线，可以是数据寄存器 DR(Data Register)中的内容，也可以是寄存器阵列 RA 中某个寄存器的内容。运算结果往往也送回累加器 A 暂存。

二、控制器

(一) 指令寄存器 IR(Instruction Register)

指令寄存器 IR 用来存放从存储器取出的将要执行的指令(实为其操作码)。

（二）指令译码器 ID(Instruction Decoder)

指令译码器 ID 用来对指令寄存器 IR 中的指令进行译码，以确定该指令应执行什么操作。

（三）可编程逻辑阵列 PLA(Programmable Logic Array)
（也称为定时与控制电路）

可编程逻辑阵列用来产生取指令和执行指令所需的各种微操作控制信号。由于每条指令所执行的具体操作不同，所以，每条指令将对应控制信号的某一种组合，以确定相应的操作序列。

三、内部寄存器阵列

通常，内部寄存器阵列包括若干组寄存器。

（一）累加器 A

累加器是用得最频繁的一个寄存器。在进行算术逻辑运算时，它具有双重功能：运算前，用来保存一个操作数；运算后，用来保存结果。

（二）数据寄存器 DR

数据寄存器 DR 用来暂存数据或指令。从存储器读出时，若读出的是指令，经 DR 暂存的指令通过内部数据总线送到指令寄存器 IR；若读出的是数据，则通过内部数据总线送到有关的寄存器或运算器。

向存储器写入数据时，数据是经数据寄存器 DR，再经数据总线 DB 写入存储器的。

（三）程序计数器 PC(Program Counter)

程序计数器 PC 中存放着正待取出的指令的地址。根据 PC 中的指令地址，准备从存储器中取出将要执行的指令。通常，程序按顺序逐条执行。任何时刻，PC 均指示要取的下一个字节或下一条指令（对单字节指令而言）所在的地址。因此，PC 具有自动加 1 的功能。

（四）地址寄存器 AR(Address Register)

地址寄存器 AR 用来存放正要取出的指令的地址或操作数的地址。

在取指令时，将 PC 中存放的指令地址送到 AR，根据此地址从存储器中取出指令。

在取操作数时，将操作数地址通过内部数据总线送到 AR，再根据此地址从存储器中取出操作数；在向存储器存入数据时，也要先将待写入数据的地址送到 AR，再根据此地址向存储器写入数据。

（五）标志寄存器 F(Flag Register)

标志寄存器 F 用来寄存执行指令时所产生的结果或状态的标志信号。关于标志位的具体设置与功能将视微处理器的型号而异。根据检测有关的标志位是 0 或 1，可以按不同条件决定程序的流向。

此外，图中还画出了寄存器阵列 RA(Register Array)，也称为寄存器组 RS(Register Stuff)。

1.4 存储器概述

一、基本概念

存储器用来存放数据和程序。在计算机内部，数据和程序都用二进制代码的形式表示。

在微机中，一般用 8 位二进制代码作为一个字节(Byte)。用一个或几个字节组成一个字(Word)。如果用字表示一个数，称为数据字；表示一条指令，称为指令字。数据字和指令字也可以用双倍字长或多倍字长表示。

微机的字长多为 8 位和 16 位，高档微机的字长可达 32 位。目前，用于工业控制的微机其典型的字长为 8 位。

一个存储器可划分为很多存储单元。存储单元中的内容为数据或指令。为了能识别不同的单元，我们分别赋予每个单元一个编号。这个编号称之为地址。显然，各存储单元的地址与该地址中存放的内容是完全不同的意思，不可混淆。

二、存储器组成

现假定存储器由 256 个单元组成，每个单元存储 8 位二进制信息，即字长为 8 位，其结构简图如图 1.7 所示。这种规格的存储器，通常称为 256×8 位的读/写存储器。

从图中可见，随机存取存储器由存储体、地址译码器和控制电路组成。

存储体共有 256 个存储单元，其编号从 00H(十六进制表示)到 FFH，即从 00000000 到 11111111。

地址译码器接收从地址总线 AB 送来的地址码，经译码器译码选中相应的某个存储单元，以便从中读出信息或写入信息。

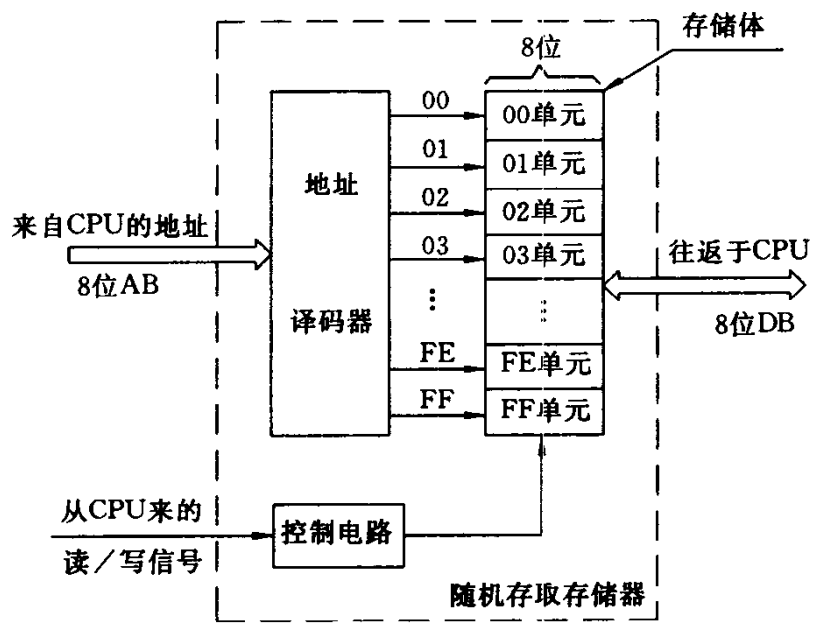


图 1.7 随机存取存储器结构简图

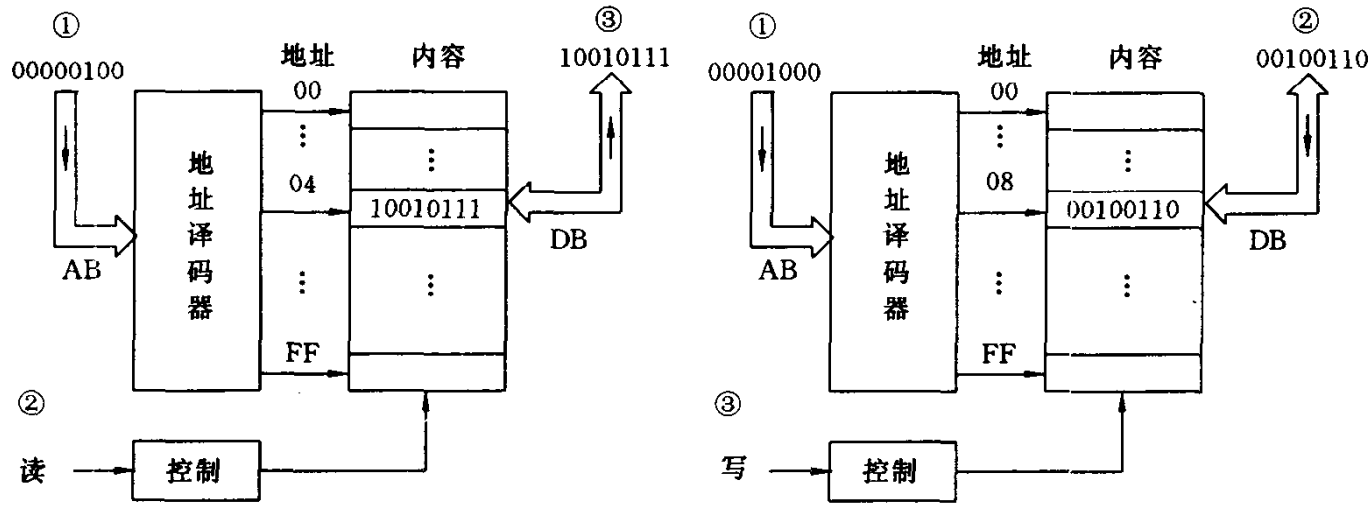
控制电路用来控制存储器的读/写操作过程。

三、读/写操作过程

从存储器读出信息的操作过程如图 1.8(a)所示。假定 CPU 要读出存储器 04H 单元 的内容 10010111 即 97H,则:(1) CPU的地址寄存器 AR 先给出地址 04H 并将它放到地址总线上,经地址译码器译码选中 04H 单元;(2)CPU 发出“读”控制信号给存储器,指示它准备把被寻址的 04H 单元中的内容 97H 放到数据总线上;(3) 在读控制信号的作用下,存储器将 04H 单元中的内容 97H 放到数据总线上,经它送至数据寄存器 DR,然后由 CPU 取走该内容作为所需要的信息使用。

应当指出,读操作完成后,04H 单元中的内容 97H 仍保持不变,这种特点称为非破坏性读出(NDRO—Non Destructive Read Out)。这一特点很重要,因为它允许多次读出同一单元的内容。

向存储器写入信息的操作过程如图 1.8(b)所示。假定 CPU 要把数据寄存器 DR 中的内容 00100110 即 26H 写入存储器 08H 单元,则:(1) CPU 的地址寄存器 AR 先把地址 08H 放到地址总线上,经地址译码器选中 08H 单元;(2) CPU 把数据寄存器中的内容 26H 放到数据总线上;(3) CPU 向存储器发送“写”控制信号,在该信号的控制下,将内容 26H 写入被寻址的 08H 单元。



(a) 存储器读操作过程示意图　　(b) 存储器写操作过程示意图

图 1.8　存储器读/写操作过程示意图

应当注意,写入操作将破坏该单元原来存放的内容,即由新内容 26H 代替了原存内容,原存内容将被清除。

上述类型的存储器称为随机存取存储器 RAM(Random Access Memory)。所谓“随机存取”即所有存储单元均可随时被访问,既可以读出也可以写入信息。

1.5　微机工作过程

微机的工作过程就是执行程序的过程,而程序由指令序列组成,因此,执行程序的过

程，就是执行指令序列的过程，即逐条地执行指令；由于执行每一条指令，都包括取指令与执行指令两个基本阶段，所以，微机的工作过程，也就是不断地取指令和执行指令的过程。微机执行程序过程示意图如图 1.9 所示。

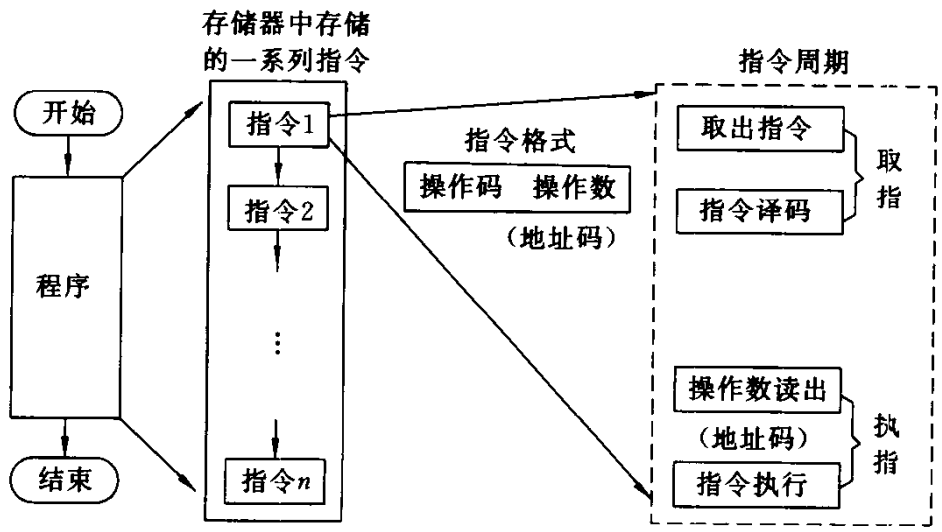


图 1.9　程序执行过程示意图

假定程序已由输入设备存放到内存中。当计算机要从停机状态进入运行状态时，首先应把第 1 条指令所在的地址赋给程序计数器 PC，然后机器就进入取指阶段。在取指阶段，CPU 从内 存中读出的内容必为指令，于是，数据寄存器 DR 便把它送至指令寄存器 IR；然后由指令译码器译码，控制器就发出相应的控制信号，CPU 便知道该条指令要执行什么操作。在取指阶段结束后，机器就进入执指阶段，这时，CPU 执行指令所规定的具体操作。当一条指令执行完毕以后，就转入了下一条指令的取指阶段。这样周而复始地循环一直进行到程序中遇到暂停指令时方才结束。

取指阶段都是由一系列相同的操作组成的，所以，取指阶段的时间总是相同的，它称为公操作。而执指阶段将由不同的事件顺序组成，它取决于被执行指令的类型，因此，执指阶段的时间从一条指令到下一条指令变化相当大。

应当指出的是，指令通常包括操作码（Operation Code）和操作数（Operand）两大部分。操作码表示计算机执行什么具体操作，而操作数表示参加操作的数的本身或操作数所在的地址，也称之为地址码。在 8 位机中，由于 1 个存储单元只能存放 1 个字节，而指令根据其所含内容不同而有单字节、双字节、3 字节乃至最多 4 字节之分，因此，在执行 1 条指令时，就可能要处理 1～4 个不等字节数目的代码信息，包括操作码、操作数或操作数的地址。

为了进一步说明微机的工作过程，我们来具体讨论一个模型机怎样执行一段简单的程序。例如，计算机如何具体计算 3+2=？虽然这是一个相当简单的加法运算，但是，计算机却无法理解。人们必须要先编写一段程序，以计算机能够理解的语言告诉它如何一步一步地去做，直到每一个细节都详尽无误，计算机才能正确地理解与执行。

在编写程序之前，必须首先查阅所使用的微处理器的指令表（或指令系统），它是某种微处理器所能执行的全部操作命令汇总。不同系列的微处理器各自具有不同的指令表。假定查到模型机的指令表中可以用 3 条指令求解这个问题。表 1.1 示出了这 3 条指令及

其说明。

表中第 1 列为指令的名称。编写程序时，写指令的全名是不方便的，因此，人们给每条指令规定了一个缩写词，或称作助记符。第 2 列即助记符。第 3 列为机器码，机器码用二进制和十六进制两种形式表示，计算机和程序员用它来表示指令。最后一列，确切地说明了执行一条指令时所完成的具体操作。

表 1.1　模型机指令表之一

名　称	助　记　符	机　器　码		说　　明
立即数取入累加器	MOV A,n	10110000 n	B0 n	这是一条双字节指令，把指令第 2 字节的立即数 n 取入累加器 A 中。
加立即数	ADD A,n	00000100 n	04 n	这是一条双字节指令，把指令第 2 字节的立即数 n 与 A 中的内容相加，结果暂存 A。
暂　　停	HLT	11110100	F4	停止所有操作

现在我们来编写 3＋2＝？ 的程序。根据指令表提供的指令，用助记符形式和十进制数表示的加法运算的程序可表达为：

```
MOV A,3
ADD A,2
HLT
```

但是，模型机却并不认识助记符和十进制数，而只认识用二进制数表示的操作码和操作数。因此，必须按二进制数的形式来写程序，即用对应的操作码代替每个助记符，用相应的二进制数代替每个十进制数。

```
MOV A,3   变成      1011 0000;操作码(MOV A,n)
                    0000 0011; 操作数 (3)
ADD A, 2  变成      0000 0100; 操作码(ADD A, n)
                    0000 0010; 操作数(2)
HLT       变成      1111 0100; 操作码(HLT)
```

注意，整个程序是 3 条指令 5 个字节。由于微处理器和存储器均用 8 位字或 1 个字节存放与处理信息，因此，当把这段程序存入存储器时，共需要占 5 个存储单元。假设我们把它存放在存储器的最前面 5 个单元里，则该程序将占有从 00H 至 04H 这 5 个单元，如图 1.10 所示。

还要指出：每个单元具有两组和它有关的 8 位二进制数，其中方框左边的一组是它的地址，框内的一组是它的内容，切不可将两组数的含义相混淆。地址是固定的，在一台微机造好以后，它的地址也就确定了；而表示的内容则可以随时由于存入新的内容而改变。

当程序存入存储器以后，我们来进一步讨论微机内部执行程序的具体操作过程。

开始执行程序时，必须先给程序计数器 PC 赋以第 1 条指令的首地址 00H，然后就进入第 1 条指令的取指阶段，其具体操作过程如图 1.11 所示。

地址		指令的内容	助记符内容
十六进制	二进制		
00	0000 0000	1011 0000	MOV A,n
01	0000 0001	0000 0011	03
02	0000 0010	0000 0100	ADD A,n
03	0000 0011	0000 0010	02
04	0000 0100	1111 0100	HLT
⋮	⋮	⋮	
FF	1111 1111		

图 1.10　存储器中的指令

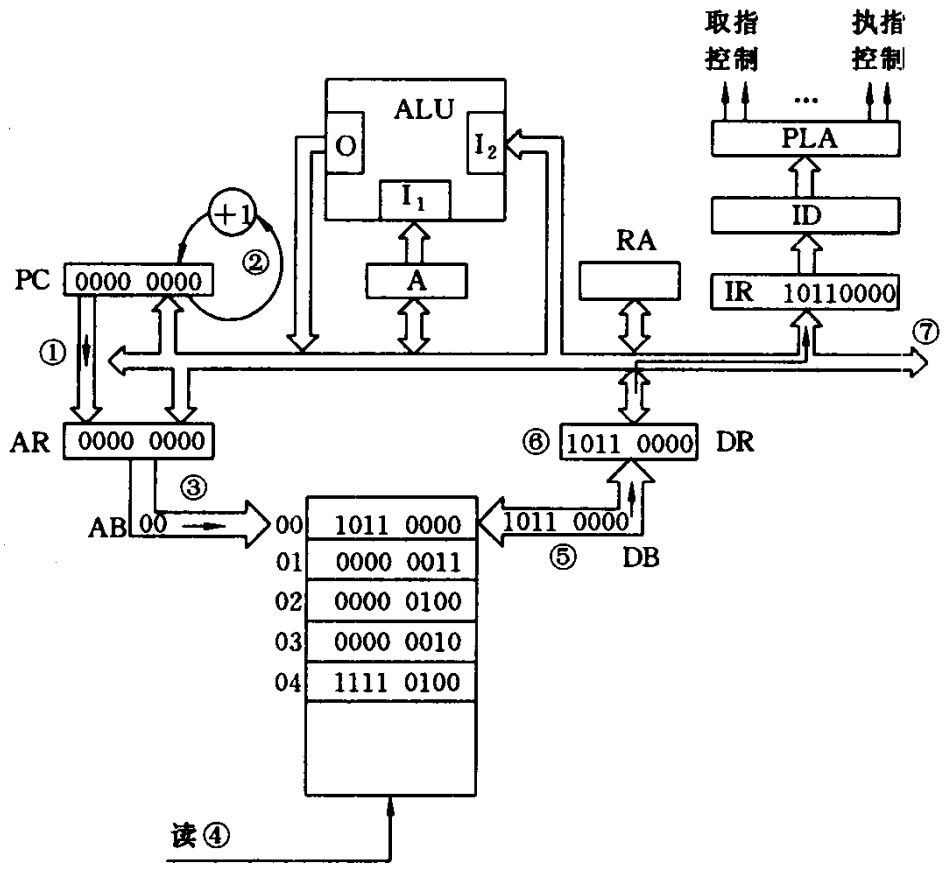


图 1.11　取第 1 条指令的操作示意图

① 把 PC 的内容 00H 送到地址寄存器 AR。

② 一旦 PC 的内容可靠地送入 AR 后，PC 自动加 1，即由 00H 变为 01H。注意，此时 AR 的内容并没有变化。

③ 把地址寄存器 AR 的内容 00H 放在地址总线上，并送至存储器，经地址译码器译码，选中相应的 00H 单元。

④ CPU 发出读命令。

⑤ 在读命令控制下，把所选中的 00H 单元中的内容即第 1 条指令的操作码 B0H 读到数据总线 DB 上。

⑥ 把读出的内容 B0H 经数据总线送到数据寄存器 DR。

⑦ 取指阶段的最后一步是指令译码。因为取出的是指令的操作码，故数据寄存器 DR 把它送到指令寄存器 IR，然后再送到指令译码器 ID，经过译码，CPU“识别”出这个操作码 B0H 就是 MOV A，n 指令，于是，它“通知”控制器发出执行这条指令的各种控制命令。这就完成了第 1 条指令的取指阶段。

然后转入执行第 1 条指令的阶段。经过对操作码 B0H 译码后，CPU 就“知道”这是一条把下一单元中的操作数取入累加器 A 的双字节指令 MOV A，n，所以，执行第 1 条指令就必须把指令第 2 字节中的操作数 03H 取出来。

取指令第 2 字节的过程如图 1.12 所示。

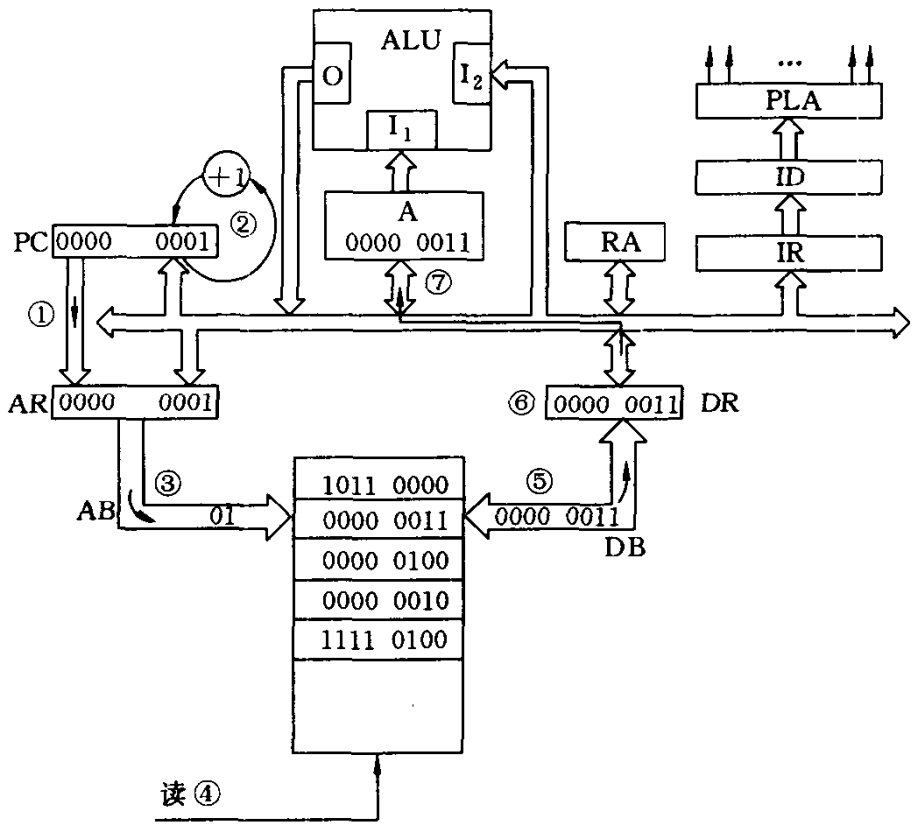


图 1.12　取立即数的操作示意图

① 把 PC 的内容 01H 送到地址寄存器 AR。

② 当 PC 的内容可靠地送到 AR 后，PC 自动加 1，变为 02H。但这时 AR 中的内容 01H 并未变化。

③ 地址寄存器通过地址总线把地址 01H 送到存储器的地址译码器，经过译码选中相应的 01H 单元。

④ CPU 发出读命令。

⑤ 在读命令控制下，将选中的 01H 单元的内容 03H 读到数据总线 DB 上。

⑥ 通过 DB 把读出的内容送到数据寄存器 DR。

⑦ 因 CPU 根据该条指令具有的字节数已知这时读出的是操作数，且指令要求把它送到累加器 A，故由数据寄存器 DR 取出的内容就通过内部数据总线送到累加器 A。于