策划编辑　高小红　彭　敏
责任编辑　龙　敏
封面设计　文翰誉诚
上架编码　Wjyl-00077

微机原理与单片机技术

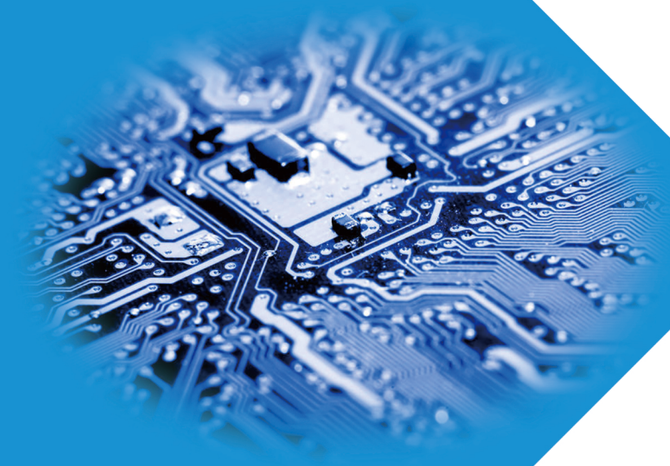
主编　陈　里　鞠现银　陈星光　周　莹

电子科技大学出版社

理论＋实践一体化教材

微机原理与单片机技术

主编　陈　里　鞠现银　陈星光　周　莹



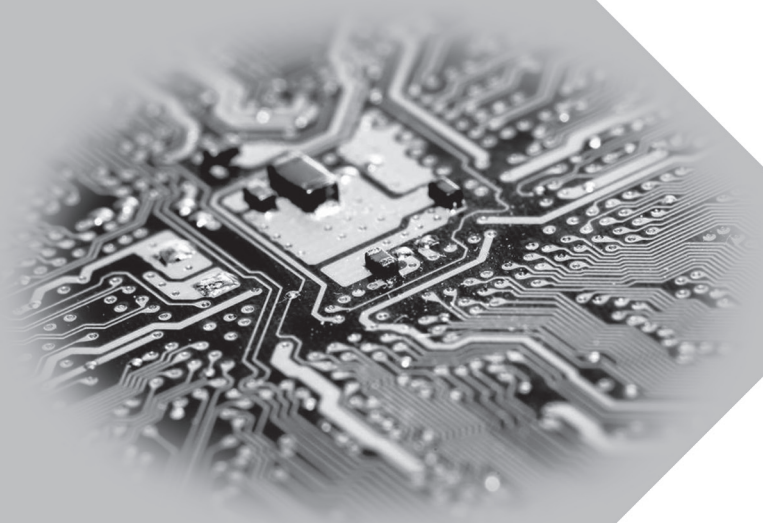
ISBN 978-7-5770-1962-8
9 787577 019628 >
定价：59.80元

电子科技大学出版社
University of Electronic Science and Technology of China Press

理论＋实践一体化教材

微机原理与单片机技术

主　编　陈　里　鞠现银　陈星光　周　莹
副主编　韩　萌　张雷伟　陈向诗瑶　刘璐玲
魏广飞　冉　波　徐　航



电子科技大学出版社
University of Electronic Science and Technology of China Press
·成都·

图书在版编目(CIP)数据

微机原理与单片机技术 / 陈里等主编. -- 成都 : 成都电子科大出版社, 2025. 9. -- ISBN 978-7-5770-1962-8

Ⅰ. TP368.1

中国国家版本馆 CIP 数据核字第 20253K0Z20 号

微机原理与单片机技术

WEIJI YUANLI YU DANPIANJI JISHU

陈　里　鞠现银　陈星光　周　莹　主编

策划编辑　高小红　彭　敏
责任编辑　龙　敏
责任校对　胡月莲
责任印制　梁　硕

出版发行　电子科技大学出版社
　　成都市一环路东一段 159 号电子信息产业大厦九楼　邮编 610051
主　　页　www. uestcp. com. cn
服务电话　028-83203399
邮购电话　028-83201495

印　　刷　湖北鄂南新华印刷包装股份有限公司
成品尺寸　185 mm×260 mm
印　　张　22.5
字　　数　495 千字
版　　次　2025 年 9 月第 1 版
印　　次　2025 年 9 月第 1 次印刷
书　　号　ISBN 978-7-5770-1962-8
定　　价　59. 80 元

目　录

上篇　微型计算机原理与接口技术

下篇 单片机原理及接口技术

上　　篇

微型计算机原理与接口技术

第 1 章
微型计算机概述

半个多世纪以来，计算机的产生、发展带动着人类在科学大道上迅猛前进。1946 年，美国宾夕法尼亚大学研制出世界上第一台电子计算机——电子数字积分计算机（Electronic Numerical Integrator And Calculator，ENIAC）。随着采用器材（件）从电子管、晶体管、中小规模集成电路发展到大规模集成电路、超大规模集成电路，20 世纪 70 年代初出现了第一台微型计算机（Microcomputer）。微型计算机与其他大、中、小型计算机的区别在于，其中央处理器（Central Processing Unit，CPU）采用了大规模 / 超大规模集成电路技术，使整个 CPU 集成在一块芯片上，这种芯片被称为微处理器。

微处理器包括算术逻辑单元（Arithmetic Logic Unit，ALU）、控制单元（Control Unit，CU）和寄存器组（Register Set，RS）三个基本部分，通常由一片大规模或超大规模器件组成。以微处理器为核心，加上内部存储器、输入 / 输出（Input/Output，I/O）接口和系统总线组成的裸机，称为微型计算机（简称“微机”）。微型计算机配上相应的外围设备（简称“外设”）、其他专用电路、电源及足够的软件，就构成微型计算机系统（Microcomputer System）。

1.1 微型计算机的发展与应用

1.1.1 微型计算机的发展概况

微型计算机的发展与微处理器的发展是同步的。微处理器集成度几乎每隔两年增加 1 倍，产品每 2～4 年更新换代一次。各代的划分通常以微处理器的字长和速度为主要依据。

1. 第一代：4 位和低档 8 位微处理器和微型计算机

第一代微处理器和微型计算机的发展时间为 1971—1973 年，这也是微型计算机的问世阶段。1971 年，美国 Intel 公司研制出了 4 位微处理器 4004，并于 1972 年生产了 8 位微处理器 8008。这一代微型计算机的特点是采用 P 型金属氧化物半导体

（P-channel Metal Oxide Semiconductor，PMOS）工艺，集成度为每片 2 300 个晶体管，字长分别为 4 位和 8 位，运算速度较慢（基本指令执行时间为 4～10 μs），指令系统简单，运算功能较差，采用机器语言或简单汇编语言。

2. 第二代：中、高档 8 位微处理器和微型计算机

第二代微处理器和微型计算机的发展时间为 1974—1977 年。该微处理器问世后，由于其体积小、使用方便等特点，受到用户的普遍欢迎。众多公司纷纷研制相应的产品，逐步形成以 Intel 公司、Motorola 公司、Zilog 公司产品为代表的三大系列微处理器。1974—1975 年，中档微处理器以 Intel 8080、Motorola 的 MC6800 为代表。1976—1977 年，出现高档 8 位微处理器，典型产品为 Intel 8085、Z80 和 MC6809。

第二代微处理器与第一代微处理器相比有了较多的改进，集成度提高了 1～4 倍，运算速度提高了 10～15 倍，指令系统相对比较完善，已具有典型的计算机体系结构，以及中断功能、直接存储器访问（Direct Memory Access，DMA）功能。软件除汇编语言外，还可使用 BASIC、Fortran 等高级语言。后期，第二代微处理器开始配上操作系统，如 CP/M 操作系统，它运用于以 8080A/8085A、Z80、MC6502 为 CPU，并带有磁盘及各种外设的微型计算机系统。

3. 第三代：16 位微处理器和微型计算机

第三代微处理器和微型计算机的发展时间为 1978—1984 年。1978 年前后，超大规模集成电路工艺的研制成功，使 1 个硅片上可以容纳 10 万个以上的晶体管。这一代微型计算机采用高性能金属氧化物半导体（High Performance Metal Oxide Semiconductor，HMOS）工艺，基本指令执行时间约为 0.5 μs，典型的微处理器是 8086、Z8000 和 MC68000。这类 16 位微型计算机具有丰富的指令系统，采用多级中断系统、多种寻址方式、多种数据处理形式、分段式存储器结构，功能大为增强。软件方面可以使用多种语言，有常驻的汇编程序、完整的操作系统、大型的数据库，并可构成多处理器系统。此外，这一阶段还出现了准 16 位微处理器，典型产品有 Intel 公司的 8088 和 Motorola 公司的 MC6809，其特点是外部数据总线为 8 位，内部数据总线为 16 位，工作速度和处理能力介于 8 位机和 16 位机之间。Intel 公司在 8086 的基础上又研制了 80186 和 80286 等性能更为优越的微处理器，其特点是从单元集成过渡到系统集成，以获得尽可能高的性能价格比。

4. 第四代：32 位微处理器和微型计算机

第四代微处理器和微型计算机的发展时间为 1985—1993 年。20 世纪 80 年代初，在每个单片硅片上可集成几十万个晶体管，产生了第四代的 32 位微处理器。其典型产品有 Intel 公司的 80386、Motorola 公司的 68020 和 National Semiconductor 公司的 16032 等。在 32 位微处理器中，具有支持高级调度、调试及系统开发的专用指令。由于集成度高，32 位微处理器中系统的运行速度和性能大为提高，可靠性增加，成本进一步降低。

5. 第五代：64 位高档微处理器和微型计算机

第五代微处理器和微型计算机的发展时间为 1994 年至今。其典型产品是 Intel 公司的奔腾系列芯片及与之兼容的 AMD 的 K6 系列微处理器芯片。64 位高档微处理器内部采用超标量指令流水线结构，并具有相互独立的指令和数据高速缓存。随着采用多媒体扩展（Multimedia Extensions，MMX）技术的微处理器的出现，微型计算机的发展在网络化、多媒体化和智能化等方面迈上了更高的台阶。

在微型计算机的每一个发展阶段，其集成度、性能等方面都有非常大的提升，微型计算机在今后将会有更快、更惊人的发展。

1.1.2 微型计算机的应用领域

微型计算机的诞生与发展使计算机的应用日益广泛和深入。它以极高的性能价格比、性能体积比，以及极大的使用方便性、灵活性，使计算机迅速推广应用到国防事业和国民经济的各个行业与领域，引起了社会、经济的巨大变革。归纳起来，微型计算机的应用主要有以下几个方面。

（1）科学计算与数据处理

在科学研究、工程设计和社会经济规划管理中存在大量复杂的数学计算问题，如卫星轨道的计算、大型水坝的设计、航天测控数据的处理、中长期天气预报、地质勘探与地震预测、社会经济发展规划的制定等，利用计算机可快速得到较理想的结果。

（2）生产与试验过程控制

在工业、国防、交通等领域，利用计算机对生产和试验过程进行自动实时监测、控制和管理，可提高工作效率和产品质量，降低生产成本。

（3）自动化仪器仪表及装置

在仪器仪表装置中使用微处理器或微型计算机，可明显增强功能，提高性能，减小质量和体积。

（4）信息管理与办公自动化

现代企事业单位和政府各部门需要管理的内容很多，如财务管理、人事档案管理、仓库材料管理、生产计划管理、信贷业务管理、购销合同管理等。采用计算机和目前迅猛发展的计算机网络技术，可实现信息管理自动化和办公自动化。

（5）计算机辅助设计

在航空航天器结构设计、建筑工程设计、机械产品设计和大规模集成电路设计等复杂设计活动中，目前普遍借助计算机进行设计，即计算机辅助设计（Computer Aided Design，CAD）。CAD 技术发展迅速，应用范围不断拓宽，目前又派生出计算机辅助测试（Computer Aided Test，CAT）、计算机辅助制造（Computer Aided Manufacture，CAM），以及将设计、测试、制造融为一体的计算机集成制造系统（Computer Integrated Manufacturing System，CIMS）等新的技术分支。

（6）计算机仿真

对一些复杂的工程问题、工艺过程、运动过程、控制行为等进行研究时，在建立数学模型的基础上，用计算机仿真的方法对相关的理论、方法、算法和设计方案进行综合、分析和评估，可以节省大量的人力、物力和时间。

（7）人工智能

人工智能是用计算机系统模拟人类某些智能行为的新兴学科，它包括声音、图像、文字等模式识别，自然语言理解，问题求解，定理证明，程序设计自动化和机器翻译，专家系统等。

（8）文化、教育、娱乐和日用家电

计算机辅助教学（Computer Aided Instruction，CAI）早已成为一种重要的教学手段。目前，电影、电视片的设计与制作，多媒体组合音像设备的推出，许多全自动、半自动“家电”产品的出现，以及许多智能型儿童玩具，无一不是微型计算机在发挥着作用。

1.2　微型计算机系统的分类和主要性能指标

1.2.1　微型计算机系统的分类

微型计算机的分类方式有很多。按微处理器的位数，可分为4位、8位、16位、32位和64位机等；按组装方式，可分为单片机、单板机、多板机，现介绍一下此类机型。

（1）单片机

利用大规模集成电路工艺将微型计算机的主要组成部分（CPU、内存和I/O接口电路等）集成在一片硅片上，这就是单片机。使用简单的开发装置可以对它进行在线开发。单片机在工业过程控制、智能化仪器仪表和家用电器中得到广泛的应用。

（2）单板机

将微型计算机的CPU、内存、I/O接口电路等多个芯片集成在一块印制电路板上就组成了单板机。

（3）多板机

根据需要把微型计算机的CPU、内存、I/O接口电路、电源等组装在不同的印制电路板上，然后装进同一个机箱，就构成了一个多板机。它可配置键盘、CRT（Cathode-Ray Tube，阴极射线管）显示器、打印机、硬磁盘驱动器等多种外设和足够的软件，成为一个完整的微型计算机系统。

1.2.2　微型计算机系统的主要性能指标

微处理器是微型计算机的心脏，它的职能是执行各种运算和信息处理，控制各

个计算机部件自动协调地完成系统规定的各种操作。它的性能与其内部结构和硬件配置直接相关，并反映出其所配置的微型计算机的特有的状态。不同型号微型计算机性能的差别主要在于微处理器的性能不同。

微处理器的性能指标主要体现在以下几个方面。

（1）字长

字长是计算机内部一次可以处理的二进制数的位数。一般一台计算机的字长取决于它的通用寄存器、内存储器、ALU 的位数和数据总线的宽度。字长越长，一个字所能表示的数据精度就越高；在完成同样精度的运算时，数据处理速度就越快。

一般情况下，微处理器的内、外数据总线宽度是一致的。但有的微处理器为了改进运算性能，加宽了微处理器的内部总线宽度，致使内部字长和对外数据总线宽度不一致。如 Intel 8088 的内部数据总线宽度为 16 位，外部数据总线宽度为 8 位。这类芯片被称为“准 × × 位”微处理器，因此 Intel 8088 被称为“准 16 位”微处理器。

（2）存储器容量

存储器容量是衡量计算机存储二进制信息量大小的一个重要指标。微型计算机中通常以字节 B（Byte 的缩写）为单位表示存储容量，并且将 1 024 B 称为 1 KB，1 024 KB 称为 1 MB（兆字节），1 024 MB 称为 1 GB（千兆字节），1 024 GB 称为 1 TB（兆兆字节）。

（3）运算速度

计算机的运算速度通常用每秒钟所能执行的指令条数表示。

（4）外设扩展能力

外设扩展能力主要是指计算机系统配接各种外设的可能性、灵活性和适应性。一台计算机允许配接多少外设，对系统接口和软件研制都有重大影响。

（5）软件配置情况

软件是计算机系统必不可少的重要组成部分，其配置是否齐全，直接关系到计算机性能的好坏和效率的高低。是否有功能很强、能满足应用要求的操作系统和高级语言、汇编语言，是否有丰富的、可供选用的应用软件等，都是在购置计算机系统时需要考虑的。

1.3 微型计算机系统的组成

1.3.1 微型计算机系统的硬件结构

1. 结构特点

目前的各种微型计算机系统，从硬件体系结构来看，基本上采用的仍是计算机

的经典结构——冯·诺依曼结构。这种结构的特点如下。

① 由运算器、控制器、存储器、输入设备和输出设备五大部分组成。

② 数据和程序以二进制代码形式不加区别地存放在存储器中。

③ 控制器是根据存放在存储器中的指令序列即程序来工作的，并由一个指令计数器控制指令的执行。

2. 主要组成部分的结构及功能

（1）CPU

CPU 包括运算器、控制器和寄存器组三个主要部分。运算器也称为“算术逻辑单元”，顾名思义，运算器的功能是完成数据的算术运算和逻辑运算。控制器通常由指令寄存器、指令译码器和控制电路组成。控制器根据指令的要求，对微型计算机各部件发出相应的控制信息，并使它们协调工作，从而完成对整个计算机系统的控制。寄存器组用来存放经常使用的数据。

（2）存储器

存储器，又称为“主存”或“内存”，是微型计算机的存储和记忆装置，用以存放数据和程序。微型计算机内存通常采用半导体存储器。

①内存单元的地址和内容。

内存中存放的是数据和程序，从形式上看，均为二进制数。在微型计算机中，通常以一个字节作为一个存储单元。内存容量就是其所能包含的内存单元的数量，以字节为单位。

不同存储单元有不同的地址，CPU 通过地址来识别不同的内存单元，如图 1-1 所示。显然，内存单元的地址和内存单元的内容是两个完全不同的概念。如在图 1-1 中，第 0 个内存单元的地址是 0000H，其内容是 11001111B，即 0CFH。

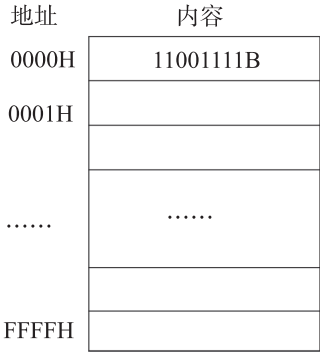


图 1-1　内存单元的地址和内容

②内存的操作。

CPU 对内存的操作有读和写两种。读操作是 CPU 将内存单元的内容读入 CPU 内部，写操作是 CPU 将其信息传送到内存单元并保存起来。写操作的结果改变了被写内存单元的内容，是破坏性的；而读操作是非破坏性的，即该内存单元的内容在

信息被读之后仍不变。

3. I/O 设备和 I/O 接口

I/O 设备是微型计算机系统的重要组成部分。程序、数据及现场信息要通过输入设备输入给微型计算机。CPU 计算的结果通过输出设备输出到外设。常用的输入设备有键盘、鼠标、扫描仪、A/D（Analog/Digital，数 / 模）变换器等。常用的输出设备有显示器、打印机、绘图仪等。磁盘既是输入设备，又是输出设备。

外设种类繁多（有机械式、电动式、电子式等），CPU 与其交换信息是比较复杂的。因此，微型计算机与 I/O 设备间的连接及信息的交换不能直接进行，而需要设计一个“接口电路”作为微型计算机与外设之间的桥梁。

4. 总线及其分类

CPU、存储器及 I/O 设备必须有机地连接在一起，才能相互协调地工作。在微型计算机中，各部分之间是采用“总线”（Bus）结构连接的，如图 1-2 所示。

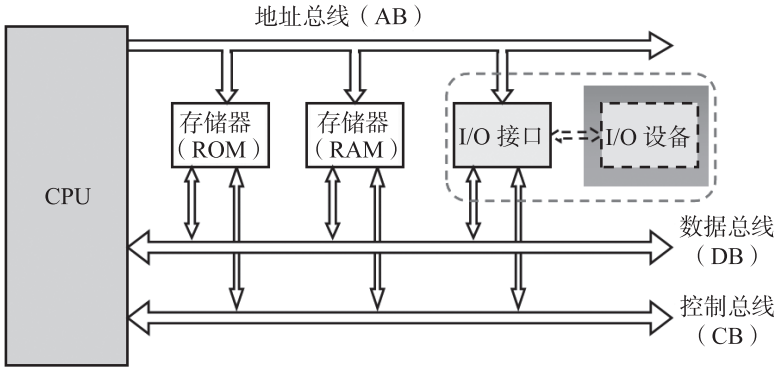


图 1-2　微型计算机结构框图

总线是微型计算机系统中多个部件之间公用的一组连线，它是芯片、插件或系统之间的标准信息通路。按功能划分，总线分为地址总线、数据总线和控制总线三类。

（1）地址总线（Address Bus，AB）

AB 用来传送地址信息，是单向总线。CPU 通过 AB 传出地址信息，用于选择存储单元和外设。AB 的宽度表明 CPU 最大可直接寻址的存储空间的大小。例如：16 位机的 CPU 的 AB 一般为 20 位，所以最大寻址空间为 2^{20}，即 1 MB。

（2）数据总线（Data Bus，DB）

DB 用来传送数据信息，它是双向的。CPU 既可通过 DB 从内存或输入设备读入数据，又可通过 DB 将 CPU 内部的数据传送至内存或输出设备。

（3）控制总线（Control Bus，CB）

CB 用来传输各种控制信号，这些控制信号有的是由 CPU 向内存及外设发出的信息，如读信号、写信号等信号；有的是外设等发送给 CPU 的信息，如终端请求信

号、准备好信号等信号。因此，CB 中每一条线的传送方向是一定的，有的从 CPU 送往其他部件，有的从其他部件送入 CPU。

采用总线结构可以减少机器中信息传输线的数目，从而提高机器的可靠性；方便对存储器芯片及 I/O 接口芯片进行扩充。

1.3.2　微型计算机的软件系统

微型计算机软件是为了运行、管理和维护微型计算机而编制的各种程序和文档的总和。软件和硬件是微型计算机系统不可分离的两个重要组成部分。没有软件的微型计算机就无法工作。微型计算机软件包括系统软件和应用软件。

1. 系统软件

系统软件主要包括操作系统（Operating System，OS）和系统应用软件。操作系统的主要作用是控制和管理微型计算机的各种软硬件资源，如 CPU、存储器、I/O 设备和其他各种系统及应用软件；同时，它还提供微型计算机的用户接口，使用户能方便自如地使用计算机。系统应用软件很多，如各种语言的汇编、解释、编译程序、诊断和调试程序等。

2. 应用软件

应用软件就是用户所编制的应用程序和文档的集合。

1.4　微型计算机的基本工作原理

1.4.1　指令、指令系统和程序简介

计算机能够自动进行计算，是由于人们将实现计算的一步步操作用命令的形式预先送入存储器，在执行时，机器把这些命令一条条地取出来，加以翻译和执行。

例如，进行“3+5=?”这个简单运算时，通常需要进行以下操作（假定 3、5 已在存储器中）。

第一步：取数 3。

第二步：取数 5 与 3 相加，保存。

第三步：停机。

所有这些取数、相加、存数等都是一种基本操作，将计算机执行的各种操作用命令的形式写下来，这就是指令。计算机所能执行的各种指令的集合，就是计算机的指令系统。计算机的指令系统，是在设计 CPU 时由设计人员规定的。在使用计算机时，必须将要解决的问题编成一条条指令，这些指令的有序集合就称为程序。

指令由操作码和操作数两大部分组成。操作码表示计算机执行什么操作，操作数表示参加操作的数本身或操作数所在的地址（即存放的位置）。

因为计算机只认识二进制数码，所以在计算机的指令系统中所有指令都必须以二进制编码的形式来表示，这称为机器指令。为了便于记忆和理解，人们用助记符（便于记忆的符号）代替操作码，用符号表示操作数，这样形成的指令称为汇编语言指令。例如：

```
MOV   AL, 03H
```

其中，MOV 是操作码助记符，AL 和 03H 是两个操作数。用汇编语言指令编写的程序称为汇编语言源程序。源程序必须翻译成机器能识别的二进制数码，计算机才能执行。

1.4.2 指令的执行过程

微型计算机执行程序的过程是从内存取出一条指令并执行这条指令，紧接着又是取指令、执行指令……如此周而复始，直到遇到停止指令。

为了进一步了解取指令、执行指令的过程，下面以计算机求解“3+5=?”为例进行说明，见表 1-1 所列。

表 1-1 求解“3+5=?”对应的汇编语言指令

操作	汇编语言指令	机器指令	
取数 3	MOV AL, 03H	10110000 00000011	B0H 03H
取数 5 与 3 相加	ADD AL, 05H	00000100 00000101	04H 05H
停机	HLT	11110100	F4H

在程序执行前，必须先将程序的机器代码装入内存储器中，同时将第一条指令的起始地址赋予指令计数器（Program Counter，PC）。程序启动后，计算机根据 PC 的值从第一条指令开始执行。

取第一条指令的过程如下。

①PC 的值 100H 送入地址寄存器（Address Register，AR）。

②PC 的内容自动加 1，即 100H 变为 101H。

③AR 将地址码 100H 通过地址总线送到地址译码器，经译码后选中 100H 单元。

④CPU 发出读控制信号。

⑤所选中的 100H 单元内容 B0H 通过数据总线送到数据寄存器（Data Register，DR）。

⑥因是取指令操作阶段，取出的是指令操作码 B0H，故 DR 将它送到指令寄存器（Instruction Register，IR）。

⑦IR 中的操作码经指令译码器 ID 译码后，通过微操作控制电路发出该指令的

有关控制命令。然后就转入第一条指令的执行阶段。

在取指阶段，把第一条指令从存储器取出并译码。这时微处理器已经知道取出的是 MOV　AL, 03H 指令。在执行指令阶段，应读出存储器下一个单元的内容，并把它送入累加器（Accumulator，AL），这条指令执行阶段的操作如下。

①PC 的值 101H 送入 AR。

②PC 的内容自动加 1，指向下一个单元。

③AR 将地址码 101H 通过地址总线送到地址译码器，经译码后选中 101H 单元。

④CPU 发出读控制信号。

⑤所选中的 101H 单元内容 03H 通过数据总线送到 DR。

⑥执行指令操作阶段，读出的是操作数，操作数 03H（在 DR 中）经内部数据总线送入 AL。

至此，第一条指令执行完毕，数据 03H 已经放入 AL 中。

同理，第二条指令的取指阶段取出加法指令；执行阶段取出操作数 05H，送入 ALU 的另一个输入端，操作数 05H 和 AL 中的 03H 相加，和数 08H 送入 AL 中。也就是说，第二条指令执行完毕后，AL 中为和数 08H。

取第三条指令，取出的是停机指令。执行这条指令后，微处理器停止全部操作。

1.5　计算机的运算基础

1.5.1　常用数制

日常生活中，人们习惯用十进制来表示数据，而在计算机内部，数据是用二进制来表示的。用二进制表示数据或地址时，书写太长，易出错，因此人们常用十六进制表示二进制数。

1. 二进制数

二进制的特点为逢二进一，由 0、1 两个数码组成，基数为 2，各个位权为 2^i，其中，i 为各数码在数中的位置。

任何一个二进制数 B（Binary number）都可以按权展开：

$$\begin{aligned} B &= b_{n-1}b_{n-2}\cdots b_1b_0b_{-1}\cdots b_{-m} \\ &= b_{n-1}\cdot 2^{n-1}+b_{n-2}\cdot 2^{n-2}+\cdots+b_1\cdot 2^1+b_0\cdot 2^0+b_{-1}\cdot 2^{-1}+\cdots+b_{-m}\cdot 2^{-m} \\ &= \sum_{i=-m}^{n-1}\left(b_i\cdot 2^i\right) \end{aligned}$$

$b_i \in \{0, 1\}$，m、n 为正整数，分别为小数部分和整数部分的位数。

2. 十六进制数

十六进制的特点为逢十六进一，由 0、1、…、9、A、B、…、F 共 16 个数码组成，基数为 16，各位权为 16^i。

任何一个十六进制数 H（Hexadecimal number）都可按权展开：

$$\begin{aligned} H&=h_{n-1}h_{n-2}\cdots h_1h_0h_{-1}\cdots h_{-m}\\ &=h_{n-1}\cdot16^{n-1}+h_{n-2}\cdot16^{n-2}+\cdots+h_1\cdot16^1+h_0\cdot16^0+h_{-1}\cdot16^{-1}+\cdots+h_{-m}\cdot16^{-m}\\ &=\sum_{i=-m}^{n-1}(h_i\cdot16^i) \end{aligned}$$

$h_i \in \{0, 1, \cdots, 9, A, B, \cdots, F\}$，m、n 为正整数。

其中，字母数码 A、B、C、D、E、F 分别与十进制数 10、11、12、13、14、15 相对应。

在编写计算机程序时，数据的书写可以用各种计数制来表示，为了区别不同的计数制，可以在数的右下角用数字标注该数的数制。例如，二进制数 1101.11 可以写成（1101.11）$_2$，还可以在数据末尾加后缀来表示：

B 后缀表示为二进制（Binary），如 01110101B 表示二进制数 01110101；

D 后缀表示为十进制（Decimal），如 68396D 表示十进制数 68396；

H 后缀表示为十六进制（Hexadecimal），如 96A7B2H 表示十六进制数 96A7B2。

省略后缀时，一般约定为十进制数，如 58740 表示十进制数 58740。

1.5.2 常用数制之间的转换

1. 二进制数与十进制数之间的转换

（1）二进制数转换为十进制数

这种转换比较简单，直接按权展开并按十进制数求和，即得到相应的十进制数。

【例 1-1】 将二进制数 110101.001B 转换为十进制数。

解：110101.001B

$=1\times2^5+1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+0\times2^{-2}+1\times2^{-3}=53.125$

（2）十进制数转换为二进制数

将十进制数分成整数部分和小数部分两个部分。对整数部分采用“除 2 取余”的方法进行转换，即将整数部分除以 2，可得整数商和余数，对商再除以 2，又得到整数商和余数，用同样的方法继续下去，直到商等于零为止，再将所得的一系列余数按逆序排列，即得到整数部分对应的二进制数；对小数部分采用“乘 2 取整”的方法进行转换，即将小数部分乘以 2，乘积保留整数部分，再将小数部分乘以 2，保留整数部分，用同样的方法继续下去，直到小数部分等于零或达到有效位数为止。然后将每次所得整数部分按顺序排列，得到小数部分对应的二进制数。最后将两部

分转换的结果合起来，便得到相应的二进制数。

【例 1-2】 将十进制数 29 转换为二进制数。

解：2 | 29 ····· 余 1 （最低位）

2 | 14 ····· 余 0

2 | 7 ····· 余 1

2 | 3 ····· 余 1

2 | 1 ····· 余 1 （最高位）

0

则 29=11101B。

【例 1-3】 将十进制数 0.495 转换为二进制数。

	0.495	0.99	0.98	0.96
	× 2	× 2	× 2	× 2
	0.990	1.98	1.96	1.92
	↓	↓	↓	↓
整数部分	0	1	1	1

最高位）　　（最低位）

则 0.495=0.0111B。

【例 1-4】 将十进制数 29.495 转换为二进制数。

解：在例 1-2 和例 1-3 中，已分别求出 29 和 0.495 的二进制数，这时，只要将它们相加即可得到最后结果，则 29.495=11101.0111B。

2. 二进制数与十六进制数之间的转换

二进制数与十六进制数之间的转换十分简单，因为一位十六进制数 0、1、…、9、A、B、…、F 与四位二进制数之间建立了一一对应的关系，见表 1-2 所列。

表 1-2　一位十六进制数与四位二进制数之间的关系

二进制数	十六进制数	二进制数	十六进制数
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D

续表

二进制数	十六进制数	二进制数	十六进制数
0110	6	1110	E
0111	7	1111	F

（1）二进制数转换为十六进制数

将二进制数转换为十六进制数时，以小数点为界线，分别向左或向右按四位二进制数进行分组，不足四位者，在前面或后面以零补足四位，再将每一组分别用相应的一位十六进制数表示，即实现转换。如果转换后的十六进制数是以字母开头的，往往在其前面添加“0”作标志。

【例 1-5】 将二进制数 10011101111.0110101B 转换为十六进制数。

解：将 10011101111.0110101B 分组并转化为对应十六进制数的过程如下。

```
100  1110  1111.0110  101B
 ↓    ↓     ↓   ↓     ↓
 4    E     F . 6     A
```

则 10011101111.0110101B=010011101111.01101010B=4EF.6AH。

（2）十六进制数转换为二进制数

将十六进制数转换为二进制数更直观，即根据表 1-1 的对应关系，将每一位十六进制数用相应的四位二进制数表示即可。

【例 1-6】 将十六进制数 3AF.85H 转换为二进制数。

解：3AF.85H 可以转换为

```
  3      A      F    .    8      5H
  ↓      ↓      ↓         ↓      ↓
0011   1010   1111      1000   0101
```

则 3AF.85H=001110101111.10000101B=1110101111.10000101B。

3. 十进制数与十六进制数之间的转换

（1）十六进制数转换为十进制数

将十六进制数转换为十进制数时，直接按权展开并求和，即得到相应的十进制数。

【例 1-7】 将十六进制数 5B3.A8H 转换为十进制数。

解：5 B3.A8H=$5\times16^{2}+11\times16^{1}+3\times16^{0}+10\times16^{-1}+8\times16^{-2}$

=1459.65625

（2）十进制数转换成十六进制数

①直接法。

类似于十进制数转换为二进制数的方法，对整数部分采用“除 16 取余”、小数

部分采用“乘 16 取整”的方法进行转换，然后进行合并。由于除 16、乘 16 的运算十分复杂，一般情况下人们很少采用此方法进行转换。

②间接法。

十进制→二进制→十六进制

【例 1-8】 将十进制数 56.75 转换为十六进制数。

解：　　56.75=111000.11B=38.CH

1.5.3　二进制数的运算

电子计算机具有强大的运算能力，它可以进行算术运算和逻辑运算两种运算。

1. 二进制数的算术运算

二进制数的算术运算包括加、减、乘、除四则运算。

（1）二进制数的加法

根据“逢二进一”规则，二进制数加法的法则如下：

①0 + 0 = 0

②0 + 1 = 1 + 0 = 1

③1 + 1 = 10（逢二进一）

（2）二进制数的减法

根据“借一当二”的规则，二进制数减法的法则如下：

①0 − 0 = 1 − 1 = 0

②0 − 1 = 1（借一当二）

③1 − 0 = 1

（3）二进制数的乘法

二进制数乘法过程可仿照十进制数乘法进行，二进制数乘法的法则如下：

①0×0 = 0

②0×1 = 1×0 = 0

③1×1 = 1

在计算过程中，由低位到高位，用乘数的每一位去乘被乘数，若乘数的某一位为 1，则该次部分积为被乘数；若乘数的某一位为 0，则该次部分积为 0。某次部分积的最低位必须和本位乘数齐，所有部分积相加的结果则为相乘得到的乘积。

（4）二进制数的除法

二进制数除法与十进制数除法类似，法则如下：

①0÷0 = 0

②0÷1 = 0

③1÷1 = 1

2. 二进制数的逻辑运算

二进制数的逻辑运算包括“或”“与”“非”和“异或”运算。

（1）逻辑“或”运算

逻辑“或”运算用符号“∨”来表示，其运算规则如下：

$$0 \vee 0=0$$
$$0 \vee 1=1$$
$$1 \vee 0=1$$
$$1 \vee 1=1$$

由此可见，两个相“或”的逻辑变量中，只要有一个为 1，“或”运算的结果就为 1。仅当两个变量都为 0 时，“或”运算的结果才为 0。

（2）逻辑“与”运算

逻辑“与”运算常用符号“·”或“∧”来表示，其遵循如下运算规则：

$$0 \cdot 0=0 \quad 或 \quad 0 \wedge 0=0$$
$$0 \cdot 1=0 \quad 或 \quad 0 \wedge 1=0$$
$$1 \cdot 0=0 \quad 或 \quad 1 \wedge 0=0$$
$$1 \cdot 1=1 \quad 或 \quad 1 \wedge 1=1$$

由此可见，两个相“与”的逻辑变量中，只要有一个为 0，“与”运算的结果就为 0。仅当两个变量都为 1 时，“与”运算的结果才为 1。

（3）逻辑“非”运算

逻辑“非”运算又称“逻辑否定”，实际上就是将原逻辑变量的状态求反，其运算规则如下：

$$\overline{1}=0$$
$$\overline{0}=1$$

由此可见，在变量的上方加一横线表示“非”。当逻辑变量为 0 时，“非”运算的结果为 1；当逻辑变量为 1 时，“非”运算的结果为 0。

（4）逻辑“异或”运算

逻辑“异或”运算常用符号“⊕”或“∀”来表示，其运算规则如下：

$$0 \oplus 0=0 \quad 或 \quad 0 \forall 0=0$$
$$0 \oplus 1=1 \quad 或 \quad 0 \forall 1=1$$
$$1 \oplus 0=1 \quad 或 \quad 1 \forall 0=1$$
$$1 \oplus 1=0 \quad 或 \quad 1 \forall 1=0$$

由此可见，两个相“异或”的逻辑运算变量取值相同时，“异或”的结果为 0；两个相“异或”的逻辑运算变量取值相异时，“异或”的结果为 1。

以上仅就逻辑变量只有一位的情况得到了逻辑“与”“或”“非”和“异或”运算的运算规则。当逻辑变量为多位时，可在两个逻辑变量对应位之间按上述规则进行运算。特别注意，所有的逻辑运算都是按位进行的，位与位之间没有任何联系，即不存在算术运算过程中的进位或借位关系。

1.5.4　计算机中数的表示方法

在日常生活中，人们常用十进制数、二进制数、八进制数、十六进制数等来表示数，而计算机内部表示数有特定的方式。

1. 机器数与真值

在计算机中，只能表示 0 和 1 两种数码，所以计算机中任何信息都是采用 0 和 1 的组合序列来表示的。

机器数：一个数在机器（计算机）中的表示形式称为机器数，形式上为二进制数。

真值：机器数的实际值叫真值。真值往往是面向人的，既可以用二进制数表示，也可以用其他进制数表示，但根据习惯，常用十进制数表示，如图 1-3 所示。

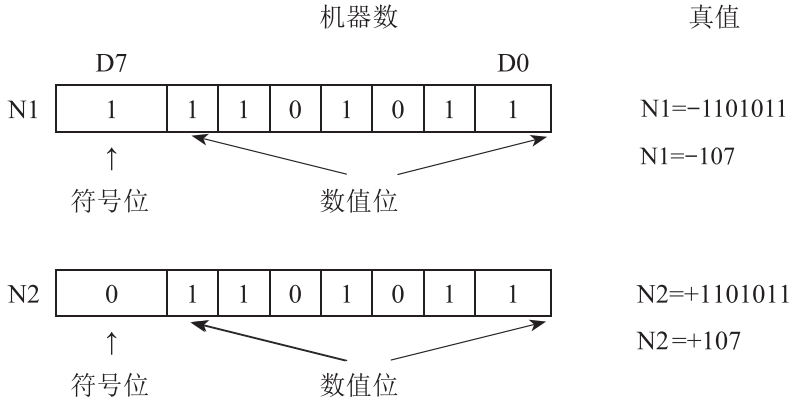


图 1-3　机器数与真值

机器数与日常生活中使用的二进制数比较，有以下特点。

①机器数为二进制形式，用 0 和 1 组合来表示数据。

②机器数符号数值化，数的最高位为符号位。当符号位为 0 时，表示正数；当符号位为 1 时，表示负数。

③小数点不直接出现。机器数通过一定的方式来表示数的小数点位置，有定点表示法和浮点表示法。

④机器数使用时需要明确采用的位数，相应有一定的表示范围。

2. 计算机中符号数的表示方法

①无符号数：与其二进制形式一样，最高位不再是符号位，而是数值位。

②有符号数：采用原码、反码和补码三种编码来表示，分别记作 $[X]_{原码}$、$[X]_{反码}$ 和 $[X]_{补码}$。

以下举例时采用数的位数为 8 位，但规则和方法也适用于 n 位机器数。

①原码。

正数的符号位用“0”表示，负数的符号位用“1”表示，其余数值位表示数值

本身，这种表示法称为原码。例如：

$$[+82]_{原码}=01010010$$

$$[-49]_{原码}=10110001$$

对于“0”，可以认为它是“+0”，也可以认为它是“−0”。因此在原码中，“0”有下列两种表示方法：

$$[+0]_{原码}=00000000$$

$$[-0]_{原码}=10000000$$

原码的表示方法很简单，只需在二进制真值的基础上，将符号位用“0”和“1”表示即可。8 位二进制数原码的数值表示范围为 −127～+127，16 位二进制数原码的数值表示范围为 −32 767～+32 767。

②反码。

正数的反码与原码相同，而负数的反码则是在原码的基础上，符号位不变（仍为“1”），数值位按位求反。例如：

$$[+82]_{反码}=01010010$$

$$[-49]_{反码}=11001110$$

而“0”的反码也有两种表示方法：

$$[+0]_{反码}=00000000$$

$$[-0]_{反码}=11111111$$

8 位二进制数反码的数值表示范围为 −127～+127，16 位二进制数反码的数值表示范围为 −32 767～+32 767。

③补码。

正数的补码与原码、反码相同，负数的补码则是在其反码的基础上加“1”（符号位保持不变）。例如：

$$[+82]_{补码}=01010010=[+82]_{原码}=[+82]_{反码}$$

$$[-49]_{补码}=[-49]_{反码}+1=11001110+1=11001111$$

补码中“0”的表示是唯一的，即

$$[+0]_{补码}=[-0]_{补码}=00000000$$

由补码的定义可以得到以下结论：

$$[\text{正数}]_{原码}=[\text{正数}]_{反码}=[\text{正数}]_{补码}$$

$$[\text{负数}]_{补码}=[\text{负数}]_{反码}+1$$

8 位二进制数补码的数值表示范围为 −128～+127，16 位二进制数补码的数值表示范围为 −32 768～+32 767。

表 1-3 所列为用 8 位二进制数表示的原码、反码和补码的相应关系。

表 1-3　有符号数的原码、反码和补码表示

十进制数	原码	反码	补码
127	01111111	01111111	01111111
126	01111110	01111110	01111110
125	01111101	01111101	01111101
……	……	……	……
2	00000010	00000010	00000010
1	00000001	00000001	00000001
0	00000000	00000000	00000000
−0	10000000	11111111	无此值
−1	10000001	11111110	11111111
−2	10000010	11111101	11111110
……	……	……	……
−126	11111110	10000001	10000010
−127	11111111	10000000	10000001
−128	无此值	无此值	10000000

补码的运算规律如下。

已知两个数 X、Y 的补码分别为 $[X]_{补码}$、$[Y]_{补码}$，则

$$[X+Y]_{补码}=[X]_{补码}+[Y]_{补码}$$

$$[X-Y]_{补码}=[X]_{补码}+[-Y]_{补码}$$

即任意两数和的补码等于两数补码之和；在求任意两数差的补码时，将减数 Y 转换为 −Y，减法运算就转换为补码加法运算。

总之，采用补码形式表示数据后，符号位可以与数值位一样参加运算，而且补码的减法运算可以通过加法运算来实现。也就是说，在计算机中只用一个加法电路，既能做加法运算，又能做减法运算，从而大大简化了计算机中运算器的结构。而一个有符号数的机器数究竟采用哪种形式，必须事先约定。对于微型计算机来说，一般采用补码形式来表示机器数。

1.5.5　常用的编码方法

在计算机内部采用二进制计数制，但在实际应用中，需要计算机处理的信息是多种多样的，如各种进位制的数据、不同语种的文字符号和各种图像信息等。为此，计算机中必须有一套遵从某种公共约定的编码系统，以便用相应的二进制编码

来表示不同的信息。最常用的两种编码方法为字母与字符的二进制编码和十进制数的二进制编码。

1. 字母与字符的二进制编码

由于计算机硬件只能识别二进制数，因此英文字母、字符和标点符号等也必须用二进制码来表示。目前，用来表示字母与字符的二进制编码方式有多种，最常用的是美国标准信息交换码（American Standard Code for Information Interchange，ASCII）。其编码方法如下：把所有可显示的字符（数字 0～9、大小写英文字母等）和控制字符（换行、回车等）共计 128 个，排列成 16 行 ×8 列的表格，字符与其位置一一对应，编码用其所在的列号、行号组成的 7 位二进制编码来表示。ASCII 码见表 1-4 所列。由表 1-4 可知，一些符号的 ASCII 码具有以下特点。

①数字符号 0～9 的 ASCII 码与其本身都相差 30H。如“0”的 ASCII 码为 30H，与数字“0”相差 30H。

②十六进制数字符号 A～F 的 ASCII 码与其本身均相差 37H 。如“B”的 ASCII 码为 42H，与“B”的值 0BH 相差 37H。

③大写字母 A～Z 的 ASCII 码与其小写字母 a～z 的 ASCII 码相差 20H。如“B”的 ASCII 码为 42H，“b”的 ASCII 码为 62H，两者相差 20H。

表 1-4　ASCII 码

低 4 位（$b_3b_2b_1b_0$）	高 3 位（$b_6b_5b_4$）							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	”	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}

续表

低4位（$b_3b_2b_1b_0$）	高3位（$b_6b_5b_4$）							
	000	001	010	011	100	101	110	111
1110	SO	RS	·	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

2. 十进制数的二进制编码

计算机中的数采用二进制形式表示，但人们常常习惯用十进制数来进行数据的输入/输出，BCD（Binary Coded Decimal，二进制编码的十进制码）就是专门用来解决用二进制数表示十进制数的问题。最常用的是8421BCD编码，其方法是采用4位二进制数来表示1位十进制数，从左至右每个二进制位对应的位权是8、4、2、1。

4位二进制数有0000～1111共16种状态，而十进制数只取0000～1001的10种状态，其余6种状态闲置不用，称为非法码。一般情况下，BCD码有压缩BCD码和非压缩BCD码两种表示形式。

（1）压缩BCD码

压缩BCD码指的是每一位十进制数用4位二进制数来表示，即一个字节表示2位十进制数。例如，十进制数59，采用压缩BCD码表示为二进制数是01011001。

（2）非压缩BCD码

非压缩BCD码指的是每2位十进制数用8位二进制数来表示，即一个字节表示1位十进制数，而且只用每个字节的低4位来表示0～9，高4位没有意义（通常为“0000”）。

由于机器内部对BCD码进行处理时，仍按二进制数运算，例如，对加法来说，二进制数是“逢二进一”，4位二进制数是“逢十六进一”，而实际BCD码是“逢十进一”，故机器运算的结果可能会出错，出现一些非法BCD码。为使计算机的运算结果和十进制数的运算结果一致，当各十进制数位的和大于9或大于等于16时，必须进行加“6”调整。在计算机系统中，这种调整过程是由BCD码调整指令自动完成的。

习　题

一、选择题

1. 下列无符号数中，最小的数是（　　）。

A. 01A5H　　B. 110110101B　　C. 2590D　　D. 3764

2. 在下列有符号数中，最大的数是（　）。

A. 1002H　　B. 1001001100101100B　　C. 25700　　D. 9614D

3. 在机器数（　）中，零的表示形式是唯一的。

A. 补码　　B. 原码　　C. 补码和反码　　D. 原码和反码

4. 8 位二进制数补码的数值表示范围为（　）。

A. −128～+127　　B. −127～+127　　C. −129～+128　　D. −128～+128

5. B9H 可以看成（　）。

A. 无符号数 185　　B. 带符号数 −71 的补码

C. 十进制数 99 的组合 BCD 码　　D. 带符号数 −57 的原码

二、填空题

1. _____、_____ 和 _____ 集成在一块芯片上，被称作微处理器。

2. 总线按其功能可分为 _____、_____ 和 _____ 三种。

3. 写出下列二进制数的原码、反码和补码（设字长为 8 位）。

（1）$(+1010110)_{\text{二进制真值}}$ =（　　）$_{\text{原码}}$ =（　　）$_{\text{反码}}$ =（　　）$_{\text{补码}}$

（2）$(-1010110)_{\text{二进制真值}}$ =（　　）$_{\text{原码}}$ =（　　）$_{\text{反码}}$ =（　　）$_{\text{补码}}$

4. $[X]_{\text{补码}}$=78H，则 $[-X]_{\text{补码}}$ =（　　）H。

5. 已知 X_1=+0010100，Y_1=+0100001，X_2=−0010100，Y_2=−0100001，试计算下列各式（设字长为 8 位）。

（1）$[X_1+Y_1]_{\text{补码}}$ = _____

（2）$[X_1-Y_2]_{\text{补码}}$ = _____

（3）$[X_2-Y_2]_{\text{补码}}$ = _____

（4）$[X_2+Y_2]_{\text{补码}}$ = _____

6. 将下列数据转换成十进制数和 BCD 码。

（1）（5D.BA）H=（　　）D=（　　）$_{\text{BCD}}$。

（2）（1001.01011）B=（　　）D=（　　）$_{\text{BCD}}$。

三、简答题

1. 简述微处理器、微型计算机及微型计算机系统三个术语的内涵。

2. 什么叫总线？为什么各种微型计算机系统中普遍采用总线结构？

3. 微型计算机系统总线从功能上分为哪三类？它们各自的功能是什么？

第 2 章
8086/8088 微处理器及其结构

微处理器是微型计算机的核心。在计算机技术快速发展的今天，不同性能、不同档次、不同品牌的微处理器涌入市场，这些产品各具特色，形成一种激烈竞争的势态。其中 Intel 公司、AMD 公司和 Cyrix 公司的产品占有市场上绝大部分份额，是当今微处理器的主流。

Intel 公司是第一家推出微处理器的公司，它的微处理器一直处于微处理器发展的前沿，是世界上第一大微处理器生产商；AMD 公司和 Cyrix 公司是 Intel 公司的两大竞争对手，它们分别是世界上第二大和第三大 CPU 生产商。

Intel 系列中各种型号的微处理器向上兼容，是它获得巨大成功的重要原因之一，所以了解 Intel 系列 CPU 的结构及工作原理对学习微型计算机大有帮助。Intel 系列微处理器种类繁多，本章仅对 8086/8088 这两种类型的微处理器作较详细的介绍。

2.1 8086/8088 微处理器的内部结构

2.1.1 Intel 8086/8088 微处理器简介

8086 是 Intel 公司于 1978 年推出的第三代微处理器，数据总线为 16 位。为方便原 8 位机用户，大约一年后 Intel 公司又推出了 8088 芯片。8088 芯片是一种准 16 位微处理器产品，内部结构与 8086 基本相同，指令系统与 8086 完全兼容，内部数据总线为 16 位，外部数据总线为 8 位。

在 8086/8088 的设计中，引入了存储器分段和指令流水线技术，这种系统结构在以后的 Intel 系列微处理器中一直被沿用与发展。8086/8088 CPU 的主要性能如下。

①字长：16 位 / 准 16 位。

②时钟频率：标准主频为 5 MHz。

③数据总线、地址总线复用。

④内存容量：20 位地址总线，可直接寻址 1 MB 存储空间。

⑤端口地址：16 位 I/O 地址总线，可直接寻址 64 KB 个端口。

⑥中断功能：可处理内部软件中断和外部硬件中断，中断源可多达 256 个。

⑦两种工作模式：支持单片 CPU 和多片 CPU 系统工作。

2.1.2 Intel 8086 微处理器内部结构

8086 微处理器从功能上可以划分为执行部件（Execution Unit，EU）和总线接口部件（Bus Interface Unit，BIU）两个逻辑单元，其内部结构如图 2-1 所示。

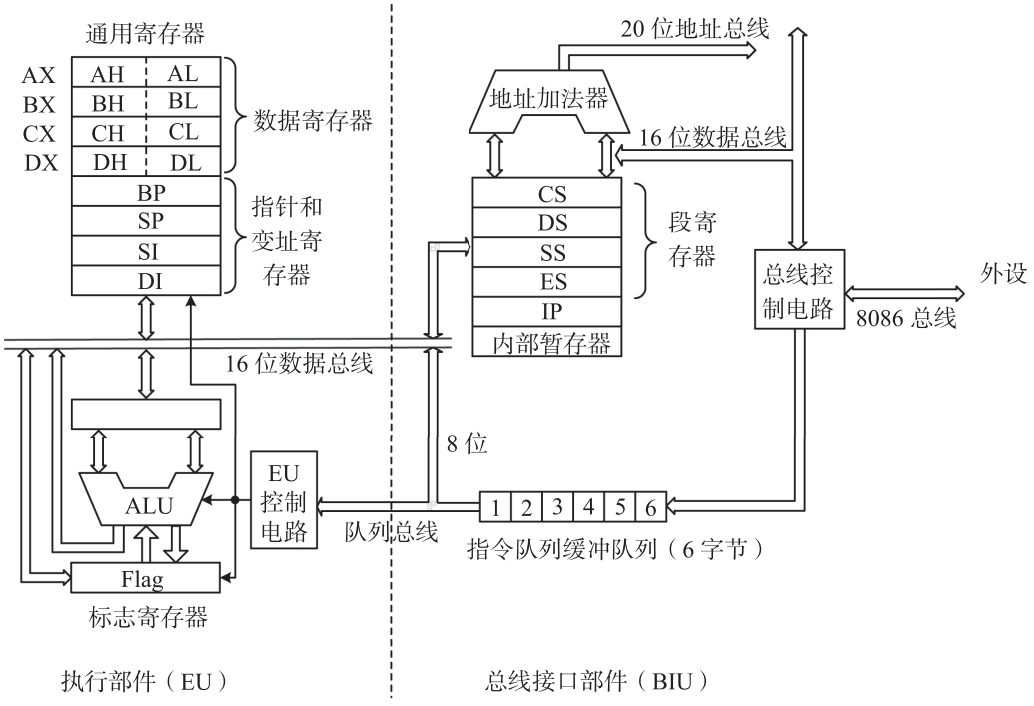


图 2-1　8086 CPU 内部结构框图

从图 2-1 可以看出：EU 主要由 1 个 16 位 ALU、1 个 16 位标志寄存器 Flag、8 个 16 位通用寄存器组和 EU 控制电路等 4 个部件组成，其主要功能是执行指令和形成访问存储器的有效地址（Effective Address，EA）；BIU 主要由 1 个 20 位地址加法器、4 个段寄存器、1 个 16 位指令指针寄存器 IP、6 个字节的指令队列缓冲器和总线控制电路等 5 个部件组成，其主要功能是形成访问存储器的物理地址、访问存储器读取指令并暂存到指令队列缓冲器中（等待执行），以及访问存储器或者 I/O 端口读取操作数并参加 EU 中的运算或存放运算结果。

1. EU

EU 只负责执行指令和形成访问存储器的有效地址。一般情况下，指令按顺序执行，EU 可源源不断地从指令队列中取得指令，而省去访问存储器取指令所需的时间。如果在执行指令过程中需要访问存储器取操作数，那么 EU 将访

问存储单元的偏移地址并送给 BIU 后，等待操作数到来后才能继续操作；如果遇到转移类指令或中断，要将指令队列中的后续指令作废，等待 BIU 重新从存储器取出目标地址中的指令代码送入指令队列缓冲器后，EU 才能继续执行指令。这种情况下，EU 和 BIU 的并行操作会受到一定的影响，这是采用并行操作不可避免的现象，只要转移指令或中断出现概率不是很高，两者的并行操作依然会取得良好的效果。

EU 中主要组成部件的功能分析如下。

（1）ALU

它可用于算术运算和逻辑运算，也可按指令的寻址方式计算出寻址单元的 16 位偏移地址（有效地址 EA），并将其送到 BIU 中（以形成 1 个 20 位的实际地址，实现对 1 MB 的存储空间寻址）。在算术运算和逻辑运算时，数据先传送到暂存寄存器中，再经 ALU 运算处理。运算后的结果经内部总线送回到累加器或其他寄存器、存储单元中。

（2）标志寄存器 Flag

它用来反映 CPU 最近一次运算结果的状态特征和存放控制标志。标志寄存器的各状态标志位记录了指令执行后的各种状态。

（3）通用寄存器

它包括 4 个 16 位数据寄存器 AX、BX、CX、DX，可以用来寄存 16 位或 8 位数据；以及 4 个 16 位指针和变址寄存器 SP、BP、SI、DI。

（4）EU 控制电路

它是控制、定时与状态逻辑电路，接收从指令队列缓冲队列中取来的指令，经过指令译码形成各种定时控制信号，对 EU 的各个部件实现特定的定时操作。

2. BIU

BIU 负责与存储器或 I/O 端口打交道。正常情况下，BIU 通过地址加法器形成指令所在存储器单元的物理地址后，访问存储器，从给定地址中取出指令代码并送到指令队列缓冲器中等待执行，一旦指令队列中空出两个字节，BIU 将自动进入读指令操作以填满指令队列。只要收到 EU 送来的操作数的有效地址，BIU 将立即形成操作数的物理地址，完成读 / 写操作数的功能。遇到转移指令时，BIU 将指令队列中尚未执行的指令作废，重新从存储器目标地址中取出指令送到指令队列。

BIU 中主要组成部件的功能分析如下。

（1）指令队列缓冲队列

8086 的指令队列可存放 6 个字节的指令代码，按“先进先出”的原则进行存取操作。一般情况下，应保证指令队列中是填满的，使 EU 连续不断地得到待执行的指令。

（2）地址加法器和段寄存器

它们用于形成存储器的物理地址，完成从 16 位存储器逻辑地址（包括段基址

和段内偏移地址两部分）到 20 位实际存储器地址（物理地址）的转换运算。

（3）指令指针寄存器 IP

它用于存放 BIU 要取的下一条指令的段内偏移地址。程序不能直接对 IP 进行存取，但能在程序运行中自动修正，使之指向要执行的下一条指令。某些转移、调用、中断和返回等指令能使 IP 的值改变。

（4）总线控制电路

它将 8086 CPU 的内部总线与外部总线相连，是 8086 CPU 与外部交换数据的通道。该电路包括 16 条数据总线、20 条地址总线和若干条控制总线。CPU 通过这些总线与外设取得联系，从而实现对数据的控制与管理。

2.1.3 8086 的指令流水线

传统的微处理器在执行程序时先从存储器中取出一条指令，然后执行指令，即取指令和执行指令是串行进行的，取指令期间 CPU 必须等待，其过程如图 2-2 所示。

图 2-2 传统微处理器的指令执行过程

在 8086 中，EU 和 BIU 的操作独立进行，两者可并行工作，使取指令和执行指令两个操作过程重叠进行，减少了 CPU 为取指令而等待的时间，从而加快了系统的运算速度，如图 2-3 所示。EU 执行指令时，不必去访问存储器取指令，而是从指令队列中取得指令代码，并分析执行它。若在指令执行过程中需要访问存储器或 I/O 端口，EU 只需向 BIU 送出有效地址。若是访问存储器，BIU 根据 EU 的要求形成访问存储器的物理地址；若是访问 I/O 端口，则 EU 送来的即为 I/O 端口的物理地址，然后根据物理地址去访问存储器或 I/O 端口，取得操作数并送到 EU 中去参加运算，或将运算结果写回到存储器或 I/O 端口中去，所以 EU 单元实际上不与外界打交道，所有与外部的操作都是在 BIU 的控制下完成的。

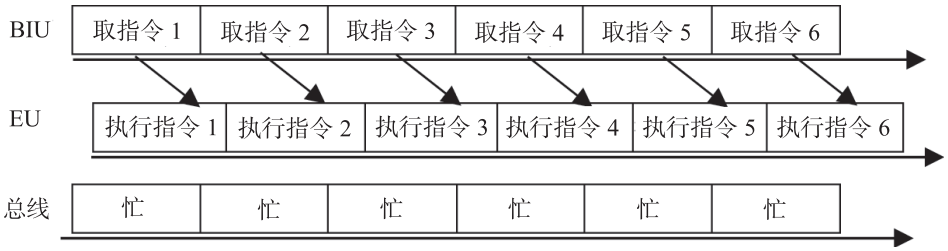


图 2-3 8086 微处理器的指令执行过程

2.1.4　8086 微处理器的寄存器结构

微处理器中的寄存器用来暂时存放参加运算的操作数和运算过程中的中间结果。寄存器的使用可减少程序执行过程中微处理器访问存储器的次数，提高微处理器的效率。

8086 微处理器中可供编程使用的有 14 个 16 位寄存器，按其用途可分为通用寄存器、段寄存器和控制寄存器三类，如图 2-4 所示。

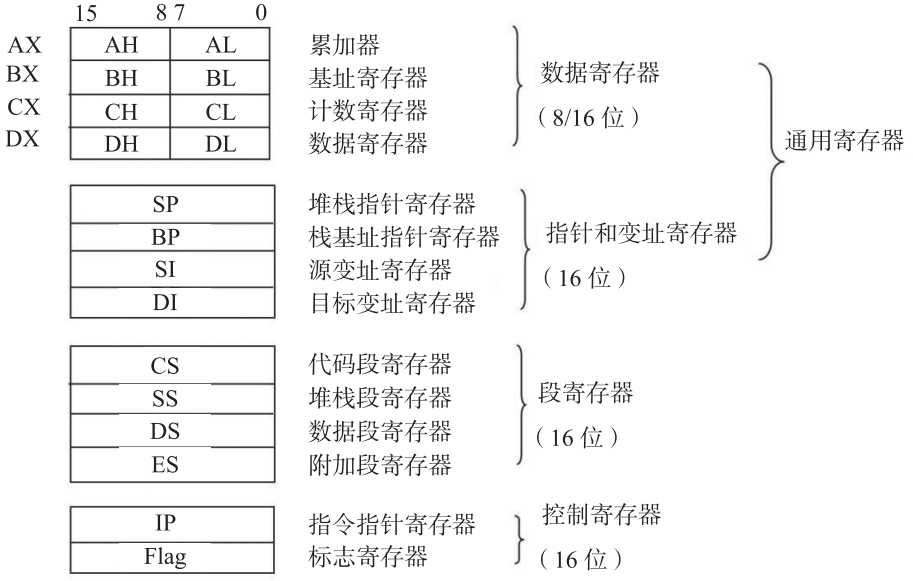


图 2-4　8086 微处理器的寄存器结构

1. 通用寄存器

8086 微处理器中设置了一些通用寄存器，操作数可以直接存放在这些寄存器中，既可减少访问存储器的次数，又可缩短程序的长度，占用内存空间少，提高了数据处理速度。

8086 的通用寄存器分为数据寄存器、指针和变址寄存器 2 组。

（1）数据寄存器

它用来存放操作数或中间结果，包括 4 个 16 位寄存器 AX、BX、CX 和 DX，还可将其分成独立的 8 位寄存器来使用，即 AH、BH、CH、DH 和 AL、BL、CL、DL 两组。16 位数据寄存器主要用于存放数据和地址，而 8 位数据寄存器只能用于存放数据。在某些指令中，这 4 个寄存器有隐含的专门用法，根据其专门用法称 AX 为累加器，BX 为基址寄存器，CX 为计数寄存器，DX 为数据寄存器。

（2）指针和变址寄存器

它用来存放地址的偏移量（被寻址存储单元相对于段起始地址的距离，或称偏

移地址），包括 16 位指针寄存器 SP、BP 和变址寄存器 SI、DI。

指针寄存器 SP 和 BP 用于堆栈段操作。其中，SP 用来存放现行堆栈段栈顶的段内偏移地址，称为堆栈指针寄存器；BP 用来存放现行堆栈段内任意一个数据单元的偏移地址，称为栈基址指针寄存器。

变址寄存器 SI 和 DI 常用于串操作，分别用来存放源操作数的段内偏移量和目标操作数的段内偏移量，故 SI 和 DI 分别称为源变址寄存器和目标变址寄存器。

通用寄存器的特定用法见表 2–1 所列。

表 2–1　通用寄存器的特定用法

寄存器名称	特定用法
AX、AL	在输入 / 输出指令中用作数据寄存器，在乘法指令中存放被乘数或乘积，在除法指令中存放被除数或商
AH、AL	AH 在 LAHF 指令中作为目标寄存器使用，AL 在 XLAT 指令中作累加器使用
BX	BX 在间接寻址方式中作基址寄存器使用，在 XLAT 指令中作基址寄存器使用
CX	在循环指令和字符串指令中作为循环次数计数器
CL	在移位 / 循环指令中作移位次数计数器使用
DX	在乘法 / 除法指令中存放乘积高位或被除数高位或余数，在间接寻址的输入 / 输出指令中作地址寄存器使用
BP	在间接寻址的指令中作基址指针使用
SP	在堆栈操作中作堆栈指针使用
SI	在字符串运算指令中作源变址寄存器使用，在间接寻址的指令中作变址寄存器使用
DI	在字符串运算指令中作目标变址寄存器使用，在间接寻址的指令中作变址寄存器使用

2. 段寄存器

8086 微处理器可直接寻址的存储器空间是 1 MB，直接寻址需要 20 位地址码，而所有的内部寄存器都是 16 位的，用这些寄存器只能寻址 64 KB 的空间。采用存储器分段技术可实现用 16 位地址寻访 1 MB 存储空间。

8086 微处理器把 1 MB 的存储空间划分为若干个逻辑段。每个逻辑段的最大长度为 64 KB，并规定每个逻辑段的 20 位起始地址的低 4 位为 0000B。这样，在 20 位段起始地址中只有高 16 位为有效数字，称这高 16 位有效数字为段的基地址（简称“段基址”），段寄存器是用来存放段基址的。

BIU 单元中有 4 个段寄存器分别用来存放 4 类段的基地址。

①代码段寄存器（Code Segment，CS）：用来存放当前程序段的段基址，要执行的指令代码均存放在当前代码段中。

②堆栈段寄存器（Stack Segment，SS）：用来存放当前堆栈段的基地址。

③数据段寄存器（Data Segment，DS）：用来存放当前数据段的基地址。

④附加段寄存器（Extra Segment，ES）：用来存放当前附加段的基地址。

3. 控制寄存器

8086 微处理器的控制寄存器主要有指令指针寄存器 IP 和标志寄存器 Flag。

（1）指令指针寄存器 IP

它是一个 16 位寄存器，存放 EU 要执行的下一条指令的偏移地址，用以控制程序中指令的执行顺序。正常运行时，BIU 可修改 IP 中的内容，使它始终指向 BIU 要取的下一条指令的偏移地址。IP 实际上是指令机器码存放单元的地址指针，不能用指令读取 IP 或给 IP 设置给定值，但可以通过某些指令修改 IP 的内容，如转移类指令就可以自动将转移目标的偏移地址写入 IP 中，实现程序转移。

（2）标志寄存器 Flag

它是一个 16 位的寄存器，共有 9 个有效标志，其中 6 个用作状态标志，3 个用作控制标志，如图 2-5 所示。

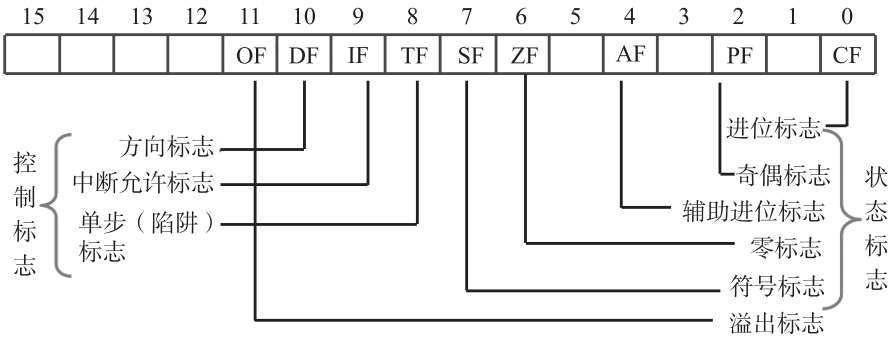


图 2-5　8086 微处理器的标志寄存器 Flag

Flag 中的状态标志用来反映 EU 执行算术运算和逻辑运算以后的结果特征，这些标志常常作为条件转移类指令的测试条件，以控制程序的运行方向；控制标志用来控制微处理器的工作方式或工作状态，它通常由程序设置或由程序清除。

①CF（Carry Flag）——进位标志位。算术运算指令执行之后，运算结果最高位（字节运算时为第 7 位，字运算时为第 15 位）若产生进位（加法时）或借位（减法时），则 CF=1；否则，CF=0。

②PF（Parity Flag）——奇偶标志位。运算指令执行后，如果运算结果的低 8 位中“1”的个数为偶数，则 PF=1；否则，PF=0。

③AF（Auxiliary Carry Flag）——辅助进位标志位。加法运算过程中，若第 3 位有进位，或者减法运算过程中，第 3 位有借位，则 AF=1；否则，AF=0。

④ZF（Zero Flag）——零标志位。运算指令执行之后，若结果为“0”，则 ZF=1；否则，ZF=0。

⑤SF（Sign Flag）——符号标志位。它和运算结果的最高位相同。在有符号运

算时最高位表示符号，若 SF=1，则运算结果为负数；若 SF=0，则运算结果为正数。

⑥OF（Overflow Flag）——溢出标志位。若本次运算结果有溢出，则 OF=1；否则，OF=0 。OF=CF ⊕ CD，CF 为进位标志位，CD 表示次最高位（即数值的最高位）的进借位，若有进借位，则 CD=1；否则，CD=0。

⑦DF（Direction Flag）——方向标志位。该标志位用于指定字符串处理指令的步进方向。当 DF=1 时，字符串处理指令以步进方式由高地址向低地址进行；当 DF=0 时，则相反。该标志位可用指令置位或者清零。

⑧IF（Interrupt Flag）——中断允许标志位。该标志位可用于控制 CPU 是否接受可屏蔽的中断请求，若 IF=1，则可以接受中断；若 IF=0，则中断被屏蔽，CPU 不接受中断。该标志位可用指令置位和复位。

⑨TF（Trap Flag）——单步（陷阱）标志位。当 TF=1 时，CPU 处于单步工作方式，此时 CPU 每执行完一条指令就自动产生一次内部中断，单步中断用于程序调试过程中。

例如：58H+3CH=94H，该运算完成后，各状态标志位的值分别为 CF=0，SF=1，ZF=0，OF=1，PF=0，AF=1。

2.2　8086/8088 微处理器的外部特性

8086 微处理器采用双列直插式的封装形式，具有 40 条引脚，如图 2-6 所示。为减少芯片上的引脚数量，8086 CPU 采用了总线分时复用技术。

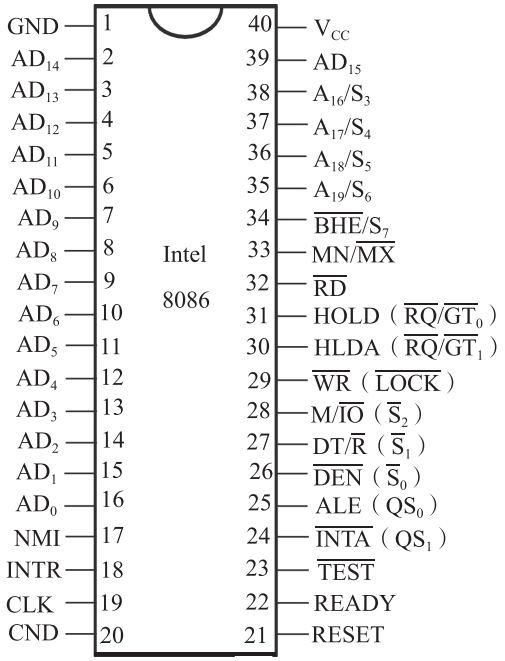


图 2-6　8086 微处理器引脚图

为适应不同的应用场合，8086/8088 微处理器有最小工作模式和最大工作模式两种工作方式。在最小工作模式下，系统没有协处理器，所有的控制信号都由 8086/8088 CPU 本身发出；在最大工作模式下，系统可包含两个或多个协处理器，其中 8086/8088 CPU 本身是主处理器，通过 8288 总线控制器提供控制信号。

2.2.1　8086 微处理器最小工作模式下的引脚

8086 微处理器的引脚按其作用可分为以下 4 类。

1. 地址 / 数据复用总线 AD_{15} ～ AD_0

AD_{15}～AD_0 是分时复用的地址 / 数据总线（Address/Data Bus）。传送地址时，单向三态输出；传送数据时，双向三态输入 / 输出。所谓“三态”是指除 0、1 两种状态外，还有一种悬浮（高阻）状态，通常采用三态门进行控制。

CPU 访问存储器或 I/O 端口一次所需的时间称为总线周期。一个总线周期，通常包括 T_1、T_2、T_3 和 T_4 状态，即 4 个时钟周期。作为地址和数据总线的复用引脚，在总线周期的第一个时钟周期（T_1 状态）传送地址，其他周期传送数据。当 CPU 处于保持响应状态时，这些引脚处于高阻隔离状态（即悬浮状态）。

2. 地址 / 状态复用总线 A_{19}/S_6 ～ A_{16}/S_3

A_{19}/S_6～A_{16}/S_3 是分时复用的地址 / 状态总线（Address/Status Bus）。在 T_1 周期作为地址线，A_{16}～A_{19} 与 AD_0～AD_{15} 一起组成访问存储器的 20 位地址线。当 CPU 访问 I/O 端口时，AD_0～AD_{15} 传送访问 I/O 端口的 16 位地址，A_{16}～A_{19} 保持为 0。在其他周期作为状态线，S_3～S_6 用来输出状态信息，其中 S_3 和 S_4 的状态组合指出当前正在使用的是哪个段寄存器，具体规定见表 2-2 所列；S_5 用来表示中断标志状态线，当 IF=1 时，S_5=1；S_6 恒保持为 0。

表 2-2　S_4、S_3 状态组合与当前段寄存器的关系

S_4	S_3	段寄存器
0	0	ES
0	1	SS
1	0	CS 或未使用任何段寄存器
1	1	DS

3. 控制总线

①$\overline{BHE}$ /S_7（Bus High Enable/Status）。$\overline{BHE}$对应总线高位有效信号，三态输出，低电平有效。在 T_1 周期，输出总线高字节允许信号$\overline{BHE}$，表示高 8 位数据线上的数据是否有效，其余时钟周期输出状态为 S_7。在 CPU 处于保持响应期间被设置为高阻状态。若$\overline{BHE}$ =1，则表示数据传送只在 AD_0～AD_7 上进行。$\overline{BHE}$和 AD_0 组合用来产生存储体的选择信号，见表 2-3 所列。

表 2-3　$\overline{BHE}$和 AD_0 组合的含义

$\overline{BHE}$	AD_0	总线使用情况
0	0	16 位数据总线上进行字传送
0	1	高 8 位数据总线上进行字节传送
1	0	低 8 位数据总线上进行字节传送
1	1	无效

②$\overline{RD}$（Read）。$\overline{RD}$ 对应读信号，三态输出，低电平有效。当 $\overline{RD}$ 有效时，表示 CPU 正在进行读存储器或 I/O 端口的操作。

③$\overline{WR}$（Write）。$\overline{WR}$ 对应写信号，三态输出，低电平有效。当 $\overline{WR}$ 有效时，表示 CPU 正在进行写存储器或 I/O 端口的操作。

④M/$\overline{IO}$（Memory/IO）。M/$\overline{IO}$ 对应存储器或 I/O 端口访问信号，三态输出。M/$\overline{IO}$ 为高电平时，表示 CPU 当前正在访问存储器；M/$\overline{IO}$ 为低电平时，表示 CPU 当前正在访问 I/O 端口。

⑤READY。READY 对应准备就绪信号，输入，高电平有效。当 READY 有效时，表示 CPU 访问的存储器或 I/O 端口已做好输入 / 输出数据的准备工作，马上可以进行读 / 写操作；当 READY 无效时，要求 CPU 插入一个或多个等待周期 T_w，直到 READY 有效为止。

⑥INTR（Interrupt Request）。INTR 对应可屏蔽中断请求信号，输入，高电平有效。CPU 在执行每条指令的最后一个时钟周期检测此引脚输入的信号，若为高电平，则表示 I/O 设备向 CPU 申请中断。如果这时 CPU 允许中断（中断允许标志位 IF=1），CPU 就会在结束当前指令后，响应中断请求，进入可屏蔽中断的处理程序。

⑦$\overline{INTA}$（Interrupt Acknowledge）。$\overline{INTA}$ 对应中断响应信号，输出，低电平有效。此引脚有效，表示 CPU 响应了某外设发来的 INTR 信号，在中断响应总线周期，可用作读选通信号。

⑧NMI（Non-Maskable Interrupt Request）。NMI 对应不可屏蔽中断请求信号，输入，上升沿触发。不受中断允许标志位 IF 的控制，当 NMI 引脚出现由低到高的电平变化时，CPU 就会在结束当前指令后，转向执行类型 2 的中断服务程序。

⑨RESET。RESET 对应复位信号，输入，高电平有效。为使 CPU 完成复位功能，该信号至少要保持 4 个时钟周期。复位后 CPU 内部寄存器的状态见表 2-4 所列。

表 2-4　复位后 CPU 内部寄存器的状态

内部寄存器	内容	内部寄存器	内容
CS	FFFFH	IP	0000H

续表

内部寄存器	内容	内部寄存器	内容
DS	0000H	Flag	0000H
SS	0000H	其余寄存器	0000H
ES	0000H	指令队列缓冲器	空

当 RESET 返回低电平时，CPU 将重新启动。

⑩ MN/$\overline{MX}$（Minimum / Maximum）。MN/$\overline{MX}$ 对应工作模式选择信号，输入。当此引脚接 +5 V 电压（高电平）时，CPU 工作在最小模式系统中；当此引脚接地（低电平）时，CPU 工作在最大模式系统中。

⑪ ALE（Address Latch Enable）。ALE 对应地址锁存允许信号，输出，高电平有效。在最小模式系统中用作地址寄存器 8282/ 8283 的选通信号。

⑫ DT/$\overline{R}$（Data Transmit / Receive）。DT/$\overline{R}$ 对应数据发送 / 接受控制信号，三态输出。在最小模式下用来控制总线收发器 8286/8287 的数据传送方向。当 CPU 输出（写）数据到存储器或 I/O 端口时，DT/$\overline{R}$ 输出高电平；当 CPU 输入（读）数据时，DT/$\overline{R}$ 输出低电平。

⑬ $\overline{DEN}$（Data Enable）。$\overline{DEN}$ 对应数据允许信号，三态输出，低电平有效。在 CPU 访问存储器或 I/O 端口的总线周期的后一段时间内和中断响应周期中，此信号为低电平。$\overline{DEN}$ 被用作总线收发器 8286/ 8287 的选通控制信号。在 DMA 方式时，$\overline{DEN}$ 为高阻状态。

⑭ HOLD（Hold Request）。HOLD 对应总线请求信号，输入，高电平有效。在最小模式系统中，当其他部件要求占用总线时，可通过对此引脚施加一个高电平请求信号，向 CPU 请求使用总线。

⑮HLDA（Hold Acknowledge）。HLDA 对应总线响应信号，输出，高电平有效。CPU 一旦测试到有 HOLD 请求时，就在当前总线周期结束时，使 HLDA 有效，表示响应这一总线请求，并立即让出总线使用权，CPU 中的指令执行部件（EU）可继续工作到下次要求使用总线为止。一直到 HOLD 无效，CPU 才将 HLDA 置成无效，并收回总线的使用权，继续操作。

⑯ CLK（Clock）。CLK 对应主时钟信号，输入。时钟频率随 CPU 芯片型号的不同而有较大差异，如 8086/8088 为 5 MHz，8086−2 为 8 MHz。

⑰ $\overline{TEST}$。$\overline{TEST}$ 对应测试信号，输入，低电平有效。当 CPU 执行 WAIT 指令时，每隔 5 个时钟周期对 $\overline{TEST}$ 进行一次测试，若 $\overline{TEST}$ 是高电平，则继续等待；若 $\overline{TEST}$ 是低电平，等待结束，CPU 继续执行下一条指令。$\overline{TEST}$ 是由协处理器或其他处理器提供的，用来协调它们之间的同步工作。

4. 电源线 V_{CC}、GND

8086 CPU 只需要单一的 +5 V 电源，由 V_{CC} 接 + 5 V，GND 为接地端。

2.2.2 8086 最小工作模式下的系统配置

图 2-7 所示为 8086 微处理器最小工作模式下的系统配置，这是一个以 8086 微处理器为主体的单机系统。系统所需要的控制信号均由 8086 微处理器直接提供。

从图 2-7 可以看出，在 8086 微处理器的最小工作模式中，硬件包括 1 片时钟发生器 8284 和 3 片地址寄存器 8282 或 8283。当系统中所连的存储器和外设较多时，需要增加两片 8286 或 8287 数字收发器以增加数据总线的驱动能力。

8282 是典型的 8 位寄存器芯片。8086 系统采用 20 位地址，加上 $\overline{BHE}$ 信号，所以共需 3 片 8282 作为地址寄存器。

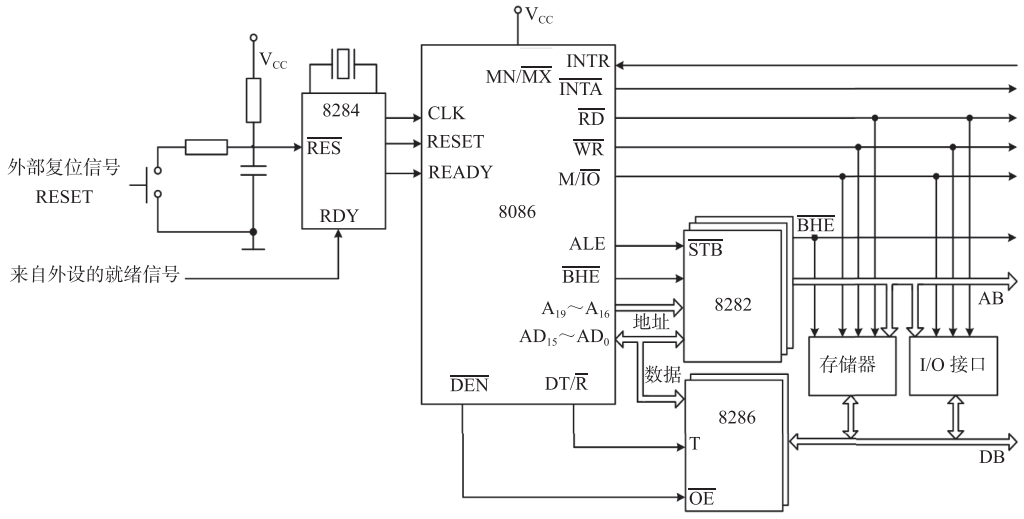


图 2-7　8086 微处理器最小工作模式下的系统配置

当一个系统中所连的存储器和外设较多时，数据总线上需要有发送器和接收器来增加数据总线的驱动能力。发送器和接收器简称“收发器”，也常常称为“总线驱动器”。8086 微处理器系统所使用的典型收发器为 8 位 8286/8287，所以该系统中需用两片 8286 或 8287。8286 和 8287 的区别仅在于 8 位数据输入或输出时，信号是反相（8287）还是不反相（8286）。

在 8086 最小工作模式系统中，8286/ 8287 作 8 位数据缓冲器用，8286/ 8287 的 OE 端与 8086 的数据允许信号 $\overline{DEN}$ 相连，当 8086 与存储器或 I/O 端口进行数据交换时，$\overline{DEN}$ 有效（$\overline{DEN}$=0），8286/8287 的$\overline{OE}$有效，允许数据输出；反之，$\overline{DEN}$ 无效，8286/8287 的 $\overline{OE}$ 无效，禁止数据输出。8286/8287 的 T 端与 8086 CPU 的数据发送 / 接收信号 DT/$\overline{R}$ 相连，当 8086 CPU 写存储器或 I/O 端口时，DT/$\overline{R}$ 为高电平，使 8286/ 8287 的 T 端为高电平，8 位数据由 CPU 向存储器或 I/O 接口输出；当 8086

读存储器或 I/O 端口时，DT/$\overline{R}$ 为低电平，使 8286 的 T 端为低电平，8 位数据由存储器或 I/O 接口向 CPU 输入。应该指出，在 8086 最小工作模式系统中，可以不用数据总线收发缓冲器 8286/8287，这时地址 / 数据总线可直接与存储器或 I/O 端口的数据线相连。

在 8086 最小工作模式配置中，除上述 8282 及 8286 之外，还有一个时钟发生器 8284。8284 的功能有产生恒定的时钟信号，对准备“好”信号（READY）及复位信号（RESET）进行同步。外界控制信号 RDY 及 $\overline{RES}$ 信号可以在任何时候到来，8284 能把它们同步在时钟后沿（下降沿）时输出 READY 及 RESET 信号到 8086 微处理器。

2.2.3　8086 微处理器最大工作模式下的引脚

当 MN / $\overline{MX}$ =0 时，8086 CPU 工作在最大模式系统中。此时，除引脚 24～31 外，其他引脚与最小模式完全相同。图 2-8 所示为 8086 微处理器最大工作模式的系统配置。

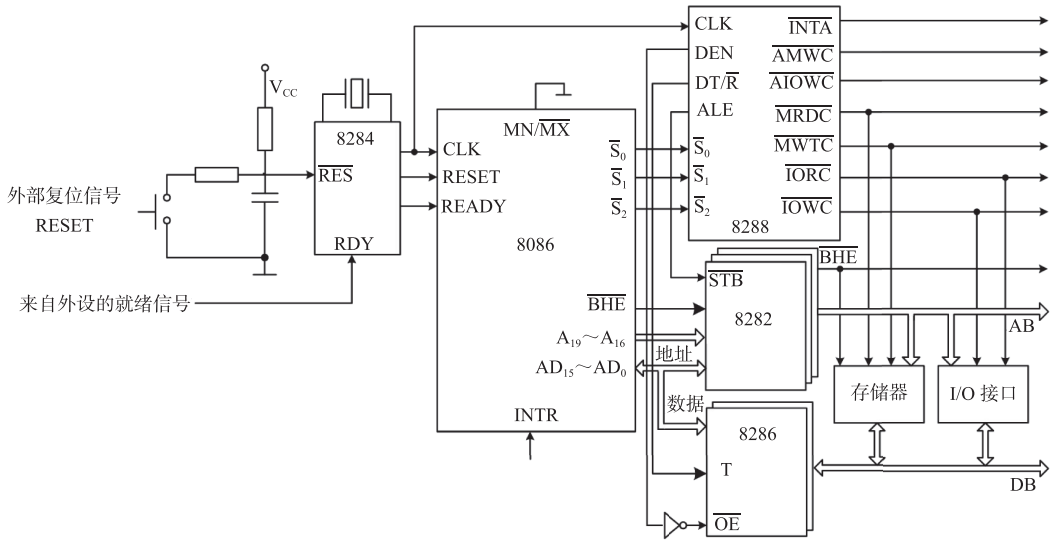


图 2-8　8086 微处理器最大工作模式下的系统配置

1. $\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$

$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$对应总线周期状态（Bus Cycle Status）信号，三态输出。$\overline{S_2}$、$\overline{S_1}$、$\overline{S_0}$用来指示当前总线周期所进行的操作类型，它们经由总线控制器 8288 进行译码，产生相应的访问存储器或 I/O 端口的总线控制信号。

2. $\overline{LOCK}$

$\overline{LOCK}$ 对应封锁信号，三态输出，低电平有效。当 $\overline{LOCK}$ 有效时，表明此时 CPU 不允许其他系统总线控制器占用总线。$\overline{LOCK}$ 是由软件设置的。在 8086 指令

系统中，有一条控制此信号的单字节总线封锁前缀指令 LOCK。当在一条指令前加上 LOCK 指令时，就能保证 CPU 在执行此指令过程中，$\overline{\text{LOCK}}$ 引脚始终是低电平，不会响应总线请求。当前面附加 LOCK 前缀指令的那条指令执行完毕时，$\overline{\text{LOCK}}$ 引脚变为高电平，撤销总线封锁，这时 CPU 允许响应总线请求。

3. QS_1、QS_0

QS_1、QS_0 对应指令队列状态（Instruction Queue Status），向外部输出。QS_1 和 QS_0 两个信号组合起来可指示 BIU 中指令队列的状态，以提供一种让其他处理器（如 8087）监视 CPU 中指令队列状态的手段。QS_1、QS_0 编码的含义见表 2-5 所列。

表 2-5　QS_1、QS_0 编码的含义

QS_1	QS_0	含义
0	0	无操作
0	1	从队列中取第一个字节
1	0	队列已空
1	1	从队列中取后续字节

4. $\overline{RQ}/\overline{GT_0}$、$\overline{RQ}/\overline{GT_1}$

$\overline{RQ}/\overline{GT_0}$、$\overline{RQ}/\overline{GT_1}$ 对应总线请求（输入）和总线请求允许（输出）（Request/Grant）信号，双向，低电平有效。$\overline{RQ}/\overline{GT_0}$ 和 $\overline{RQ}/\overline{GT_1}$ 为 8086 和其他处理器（如 8087 和 8089）使用总线时所提供的一种仲裁电路，以代替最小模式下的 HOLD/HLDA 两信号的功能。$\overline{RQ}/\overline{GT_0}$ 和 $\overline{RQ}/\overline{GT_1}$ 是专门为多处理器系统设计的。输入时表示其他处理器向 CPU 请求使用总线，输出时表示 CPU 对总线请求的响应信号，两条线可同时与两个处理器相连，内部保证 $\overline{RQ}/\overline{GT_0}$ 比 $\overline{RQ}/\overline{GT_1}$ 有较高的优先级。

2.2.4　8088 CPU 的引脚

8088 CPU 大部分引脚的名称及功能与 8086 相同，不同之处仅在于以下几点。

① 由于 8088 CPU 的外部数据总线只有 8 条，因此分时复用地址数据总线只有 AD_0～AD_7。8088 CPU 的 A_8～A_{15} 对应 8086 CPU 的 AD_8～AD_{15}，专门用来传送地址。

② 8086 CPU 中的第 28 号引脚是 M/$\overline{IO}$，在 8088 CPU 中改为 IO/$\overline{M}$，定义相反。

③ 8086 CPU 中的第 34 号引脚是 $\overline{BHE}$，在 8088 CPU 中改为 SS_0，它与 DT/$\overline{R}$、IO/$\overline{M}$ 一起表示 8088 CPU 在最小模式中的周期状态。

2.3　存储器组织

存储器是计算机存储信息的地方。程序运行所需要的数据、程序执行的结果及程序本身均保存在存储器中。

2.3.1　数据的存储格式

计算机存储信息的基本单位是一个二进制位（Bit），一个二进制位可以存储一位二进制数“0”或“1”。8 个二进制位组成一个字节（Byte），位编号由右向左从 0 开始递增。8086 的字长为 16 位，由两个字节组成，称为一个字（Word），位编号由右向左从 0 开始递增。

在存储器里，以字节为单位存储信息。为了正确地存储信息，每一个字节单元都被赋予一个地址，即存储单元地址。地址编号从 0 开始，顺序加“1”，是一个无符号的二进制整数，常用十六进制数表示。

一个存储单元中存放的信息称为该存储单元的内容，如图 2-9 所示。

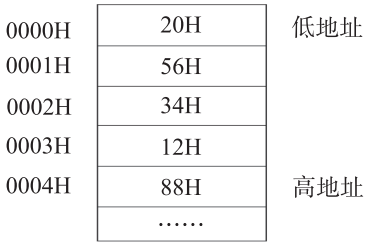


图 2-9　存储格式

在 0002H 地址单元存储的信息为 34H，即 0002H 单元的内容为 34H，表示为

[0002H]=34H　或　（0002H）=34H

每个存储单元的内容是一个字节，若存放的数据为一个字，则要占用连续两个存储单元。存放时字的低字节存放在低地址单元中，高字节存放在高地址单元中，并以低地址作为该字的地址。从偶地址开始存放的字，称为规则字或对准字；从奇地址开始存放的字，称为非规则字或非对准字。

在图 2-9 中，0002H“字”单元的内容为 [0002H]=1234H。

因此，同一个地址既可以看作字节单元的地址，又可以看作字单元的地址，这要根据具体情况来确定。

2.3.2　8086/8088 系统对存储器的组织

1. 8086 系统对存储器的组织

8086 微处理器有 20 条地址线，可直接寻址的存储器空间为 1 MB（2^{20} B），地

址范围为 0～（$2^{20}-1$）（00000H～FFFFFH）。1 MB 的存储空间被分成两个 512 KB 的存储体，分别叫高位库和低位库。低位库固定与 CPU 低位数据线 D_0～D_7 相连，该存储体中的每个存储单元的地址均为偶地址；高位库与 CPU 的高位数据线 D_8～D_{15} 相连，该存储体中的每个存储单元的地址均为奇地址。两个存储体之间采用字节交叉编址方式，如图 2-10 所示。

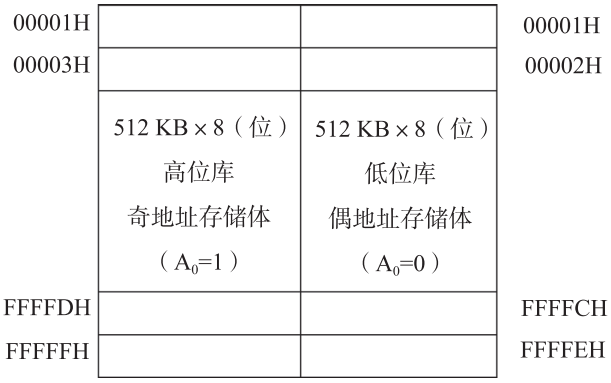


图 2-10　8086 存储器的分体结构

对于任何一个存储体，只需要 19 位地址线 A_1～A_{19} 就够了，最低位地址线 A_0 用以区分当前访问哪一个存储体。当 A_0=0 时，表示访问偶地址存储体；当 A_0=1 时，表示访问奇地址存储体。

8086 系统设置了一个总线高位有效控制信号 $\overline{BHE}$。$\overline{BHE}$ 与 A_0 相互配合，使得 CPU 可以同时访问两个存储体中的一个字信息。$\overline{BHE}$ 与 A_0 的组合控制作用见表 2-3 所列。

两个存储体与 CPU 总线之间的连接如图 2-11 所示。奇地址存储体的片选端受控于 $\overline{BHE}$ 信号，偶地址存储体的片选端受控于地址线 A_0。

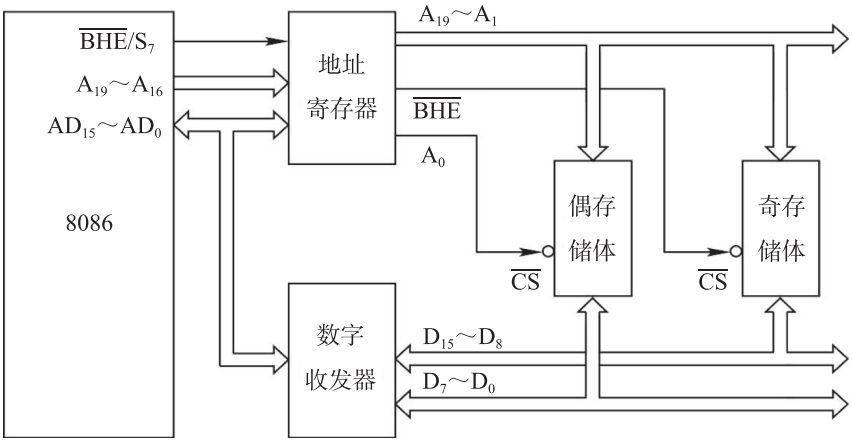


图 2-11　存储体与总线的连接

当访问存储器中某个字节单元时，指令中的地址码经变换后得到 20 位的物理地址。

如果是偶地址（$\overline{BHE}$=1 与 A_0=0），这时可由 A_0 选定偶地址存储体，由 A_0～A_{19} 决定偶地址存储体的某个字节单元，读 / 写该单元中一个字节的信息，通过数据总线的低 8 位传送数据；如果是奇地址（A_0=1），那么偶地址存储体不会被选中，系统将自动产生 $\overline{BHE}$=0 作为奇地址存储体的选择信号，与 A_1～A_{19} 一起选定奇地址存储体中的某个字节单元，并读 / 写该单元中一个字节的信息，通过数据总线的高 8 位传送数据。

当访问存储体中某个字单元时，一种情况是需要访问从偶地址开始存放的一个字（即高字节在奇地址中，低字节在偶地址中），可一次访问存储器读 / 写一个字信息，这时 $\overline{BHE}$=0，A_0=0；另一种情况是需要访问从奇地址开始的一个字（即高字节在偶地址中，低字节在奇地址中），这时需要访问两次存储器才能读 / 写这个字的信息，第一次访问存储器读 / 写奇地址中的字节，第二次访问存储器读 / 写偶地址中的字节。显然，为了加快程序的运行速度，希望访问存储器的字地址为偶地址（规则字）。

2. 8088 系统对存储器的组织

在 8088 系统中，可直接寻址的存储器空间同样为 1 MB，但是整个 1 MB 的存储空间同属于一个单一的存储体。对于 8088 CPU，由于外部数据总线是 8 位，无论是字还是字节的存取操作，也不管是规则字还是非规则字，8088 CPU 每访问一次存储器只能读 / 写一个字节的信息，任何字信息的存取都需要访问两次存储器才能完成，因此 8088 系统中的程序运行速度比 8086 系统中的慢些。

2.3.3　存储器分段

在 8086 系统中，可寻址的存储空间达 1 MB。要对整个存储器空间寻址，则需要 20 位的地址码，而 8086 CPU 内所有的寄存器均只有 16 位，只能寻址 64 KB（2^{16} B）的空间。因此把整个存储空间分成许多逻辑段，每个逻辑段容量最多为 64 KB。8086 微处理器允许它们在整个存储空间中浮动，各个逻辑段之间既可以紧密相连，又可以相互重叠，还可以分开一段距离，如图 2-12 所示。

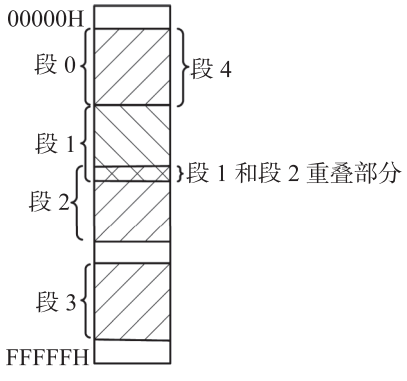


图 2-12　逻辑分段示意图

在 8086/8088 存储器中，16 字节的存储空间称为一节（Paragraph），且规定节的起始地址能被 16 整除。各个逻辑段应从节的起始边界开始，即保证段起始地址的低 4 位地址为“0”。

2.3.4 存储器地址

在 8086 系统中，访问 1 MB 的内存空间时，其 20 条地址线都有效，但其内部地址寄存器都是 16 位的，因此经常要用到下面的各类地址对存储器地址进行描述。

①物理地址（Physical Address，PA）。物理地址为无符号的 20 位二进制数，是存储单元的实际地址，是唯一代表存储空间每个字节单元的地址。当 CPU 与某个存储单元交换信息时，必须给出该存储单元的物理地址，才能进行存取操作。

②段的起始地址。段的起始地址为无符号的 20 位二进制数，是各类段的第一个存储单元的物理地址。段的起始地址要求能被 16 整除，即 20 位物理地址的低 4 位为“0”。

③段基址（Segment Address）。段基址为无符号的 16 位二进制数，是段的起始地址的高 16 位，当前段的段基址存放在段寄存器中。

④偏移地址（Offset Address）。偏移地址为无符号的 16 位二进制数，是要寻址的内存单元的地址与本段首地址的偏差。在编程中常被称为“有效地址”。

⑤逻辑地址（Logical Address）。逻辑地址为无符号的 16 位二进制数，是在程序中使用的地址，由段基址和偏移地址两部分组成。其表示形式为“段基址：偏移地址”。

逻辑地址到物理地址的转换由 BIU 中 20 位的地址加法器自动完成，如图 2-13 所示。实际上，物理地址是段基址左移 4 位（二进制数）加偏移地址形成的。

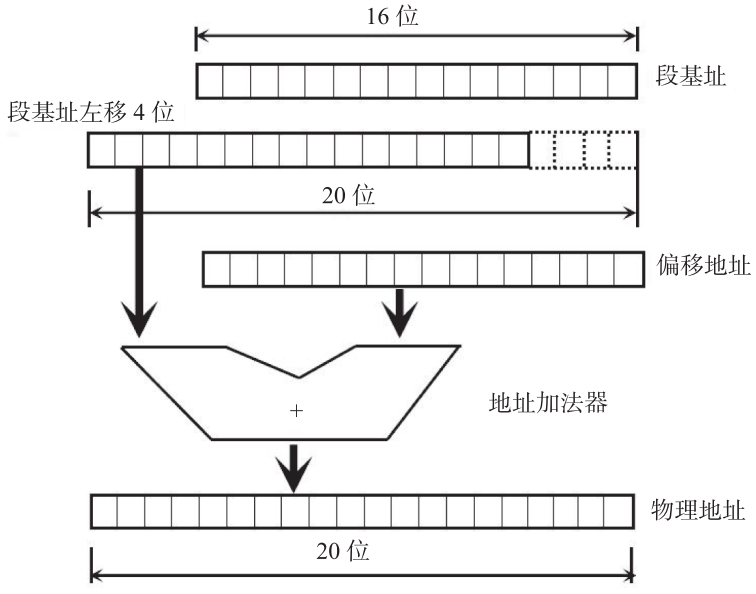


图 2-13 物理地址的形成过程

物理地址的计算公式为

$$物理地址 = 段基址 \times 10H + 偏移地址$$

例如：当（CS）=2000H，（IP）=0100H 时，其指向的物理地址为

$$\begin{aligned} 2000H \times 10H+0100H &= 2000H \times 16+0100H \\ &= 20000H+0100H \\ &= 20100H \end{aligned}$$

访问存储器时，段基址是由段寄存器提供的。8086 微处理器通过 4 个段寄存器来访问不同的段。不同的操作使用的段基址和偏移地址的来源不同，表 2-6 给出了各种访问存储器操作所使用的段寄存器和段内偏移地址的来源。

表 2-6　逻辑地址来源

操作指令	段基址		偏移地址
	正常来源	其他来源	
取指令	CS	无	IP
堆栈操作	SS	无	SP
存 / 取变量	DS	CS、ES、SS	有效地址 EA
取源串	DS	CS、ES、SS	SI
存目标串	ES	无	DI
通过 BP 间接寻址	SS	CS、ES、DS	有效地址 EA

一般情况下，段寄存器的使用由系统约定，有些操作除约定的段寄存器外，还可指定其他段寄存器。对于数据存取，除由约定的 DS 给出段基址外，还可指定 CS、SS 和 ES；有些操作只能使用约定的段寄存器，不允许指定其他段寄存器，如取指令只能使用 CS。

2.3.5　专用和保留的存储器单元

Intel 公司在 8086/8088 系统存储区的最低地址区和最高地址区保留了一些单元供 CPU 的某些特殊功能专用。其中主要有如下功能。

① 00000H～003FFH（共 1 KB）。存放中断向量表，中断向量即中断服务程序的入口地址，每个中断向量占 4 个字节，前 2 个字节存放中断向量的偏移地址，后 2 个字节存放中断向量的段基址。因此，1 KB 区域可以存放 256 个中断向量。这个区域被包含在系统的随机存取存储器（Random Access Memory，RAM）范围内。

② FFFF0H～FFFFFH（共 16 B）。系统上电或复位时，（CS）= FFFFH，（IP）= 0000H，故 FFFF0H 为系统的复位地址，从这个地址开始存放一条无条件转移指令，

使系统自动跳转到初始化程序。这个区域被包含在系统的只读存储器（Read-only Memory，ROM）范围内。

2.4 8086 微处理器的工作时序

2.4.1 总线周期

计算机的工作过程是执行指令的过程。为提高指令的执行效率，在 8086/8088 CPU 中设置了可独立操作的指令执行部件 EU 和总线接口部件 BIU，两者的分工各不相同。

总线接口部件 BIU 负责从存储器取出指令并送入指令队列中，或者从存储器或 I/O 端口取出操作数去参加 EU 中的运算，或者是将 EU 运算的结果写入存储器或 I/O 端口中。实际上，BIU 是负责与 CPU 外部（包括存储器和 I/O 端口）交换数据的，而这些操作均要经过系统外部总线来完成，所以在 8086/8088 CPU 中把 BIU 完成一次访问存储器或访问一次 I/O 端口操作所需要的时间称为一个总线周期。理想情况下，BIU 可处于连续工作状态，不断地访问存储器进行读取指令或读 / 写操作数或访问 I/O 端口的工作，即不断地执行总线周期。

指令执行部件 EU 负责执行指令。它只需要从指令队列中取得指令，并分析执行它。在指令执行过程中，可根据需要随时要求 BIU 访问存储器取操作数或写运算结果或访问 I/O 端口。EU 部件的操作与 BIU 访问外设的操作可并行进行，理想情况下，EU 也可处于连续工作状态，不断地执行从指令队列中得到的指令。

实际上，EU 和 BIU 均不可能完全处于连续工作状态。当 EU 执行某些复杂指令时，内部操作时间很长，且不需要访问外设时，BIU 处于空闲状态，可用 T_1 表示。同样，EU 有时需要等待 BIU 从外设取出操作数后才能进行计算，或遇到转移指令时，原来指令队列中已取出的指令全部作废，要等待 BIU 重新从存储器中取出目标地址中的指令后，EU 才能继续执行下一条指令。但是，由于 EU 的内部操作过程可被 BIU 的总线周期所覆盖，所以可以不考虑 EU 的内部操作时序。

在 8086/8088 CPU 中，每个总线周期至少包含 4 个时钟周期（T_1～T_4），如图 2-14 所示为典型的总线周期时序。在 T_1 状态下，BIU 把要访问的存储单元地址（20 位）或 I/O 端口地址（16 位）输出到地址总线上。若为读周期，在 T_2 状态中总线 AD_0～AD_{15} 处于悬浮（高阻）缓冲状态，以使 CPU 有足够的时间从输出地址信息方式转变为输入（读）数据信息的方式及等待存储器 I/O 端口从接收到地址信息到可靠地把数据送到数据线上。在 T_3～T_4 中，CPU 从总线 AD_0～AD_{15} 上读入数据；若为写周期，由于输出地址和输出数据都是输出（写）方式，CPU 无须改变输入 / 输

出方式的缓冲时间，CPU 可以在 T_2～T_4 中把数据输出在数据总线上。考虑到 CPU 与慢速的存储器或 I/O 接口之间传送速度间的配合，有时需要在 T_3 和 T_4 状态之间插入若干个附加的时钟周期 T_w，待存储器或 I/O 接口将准备好的数据送上数据总线或可靠地从数据总线上获取数据后，再通知 CPU 脱离等待状态，并立即进入 T_4 状态。故这种插入的附加时钟周期称为等待周期（T_w）。

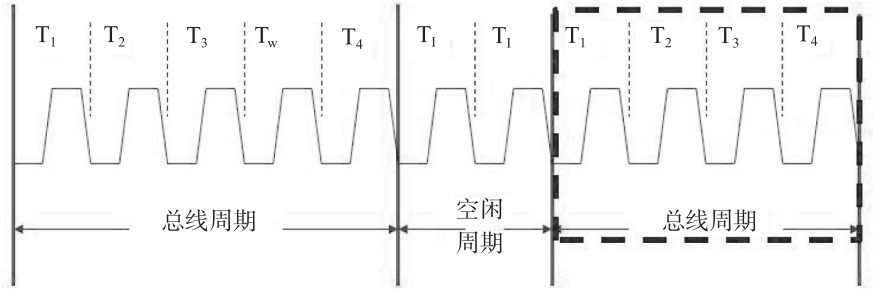


图 2-14　典型的总线周期时序

应当指出，仅当 BIU 需要补充指令队列中的空缺，或者当 EU 在执行指令过程中需要经外部总线访问存储单元或 I/O 端口时，BIU 才会进入执行总线周期的工作时序。也就是说，总线周期不是一直存在的，而时钟周期却是一直存在的。在两个总线周期之间，可能会出现一些没有 BIU 活动的时钟周期，处于这种时钟周期中的总线状态称为空闲状态（T_I）。如图 2-14 所示为总线周期中有空闲状态及等待周期的时序。通常，当 EU 执行一条占用很多时钟周期的指令（如乘、除法指令）时，或者在多处理器系统中，在交换总线控制权时就会出现空闲状态。

8086/8088 CPU 在最小工作模式和最大工作模式中控制信号不完全相同，因此时序也有所不同，本节只介绍最小工作模式下的部分时序。

2.4.2　最小工作模式读 / 写总线周期

1. 8086 CPU 读总线周期

图 2-15 所示为最小工作模式下的 8086 CPU 读总线周期时序图。一个典型的总线周期包含 4 个时钟周期，即 T_1、T_2、T_3 和 T_4。在存储器和外设速度较慢时，要在 T_3 和 T_4 之间插入一个或几个等待周期 T_w。在各时钟周期中，所完成的操作功能如下所述。

（1）T_1 周期

BIU 将被访问存储单元或 I/O 端口的 20 位物理地址 AD_0～AD_{15} 和 A_{16}/S_3～A_{19}/S_6 连同总线高位有效信号 $\overline{BHE}$（访问 I/O 端口时有效位数为 16 位的 AD_0～AD_{15}，A_{16}/S_3～A_{19}/S_6 和 $\overline{BHE}$ 为零）一起送上总线，在地址锁存允许信号 ALE 的控制下，将地址锁存到 8282/8283 地址寄存器中，然后输出到地址总线上，由M / $\overline{IO}$信号确定是访问存储器还是访问 I/O 端口。

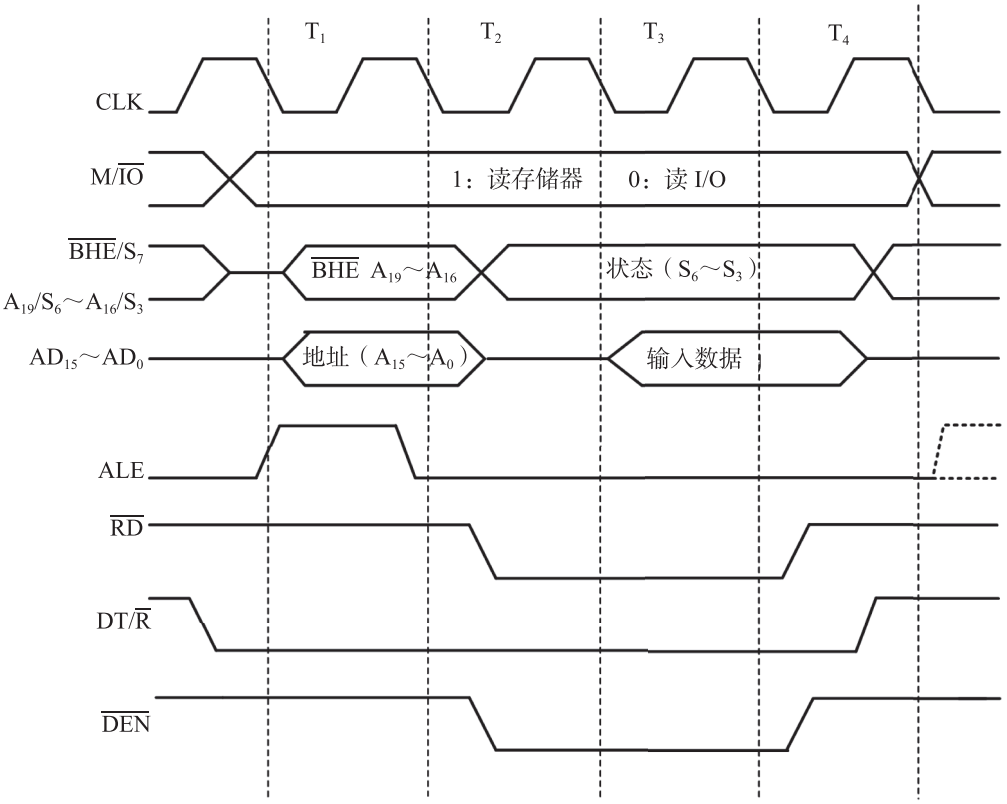


图 2-15　8086 CPU 读总线周期时序图（最小工作模式下）

另外，当系统接有数字收发器时，要用到 DT/$\overline{R}$ 和 $\overline{DEN}$ 作为控制信号。前者作为数据传输方向的控制，后者实现数字收发器的选通。在 T_1 周期，若 DT/$\overline{R}$ 端输出为低电平，则表示本总线周期为读周期，即数字收发器是从数据总线上接收数据。

（2）T_2 周期

CPU 撤销输出的地址信息，结束 AD_0～AD_{15} 上的地址信息输出，AD_0～AD_{15} 成为高阻浮空状态，A_{16}/S_3～A_{19}/S_6 和 $\overline{BHE}/S_7$ 立即成为状态信息输出。与此同时，$\overline{RD}$ 信号有效，启动被选的存储器或 I/O 端口。此时，DT/$\overline{R}$ 为低电平，$\overline{DEN}$ 也为低电平，8286 处于反向传送（即信息由数据总线传向 CPU）。

（3）T_3 周期

CPU 继续提供状态信息，并且继续维持 $\overline{RD}$、M/$\overline{IO}$ 及 DT/$\overline{R}$、$\overline{DEN}$ 信号为有效电平。

在总线周期 T_3 与 T_4 之间是否要插入等待周期 T_w，是由系统中专门设置的 READY 信号来控制的。当系统所采用的存储器或 I/O 接口工作速度较慢，来不及在 T_3 周期送出所需要的信息时，则应利用 READY 电路产生 READY 信号并经 8284 系统时钟电路同步后加到 CPU 的 READY 引脚上。若 CPU 在 T_3 周期开始时检测到

READY 为无效，则 8086 CPU 在 T_3 周期结束后不应立即进入 T_4 周期，而应插入一个 T_w 周期。以后在每一个 T_w 周期的开始都要检测 READY 引脚电平，只有检测到 READY 为高电平时，才在这个 T_w 周期后进入 T_4 周期。

（4）T_4 周期

在 T_4 周期和前一个周期交界的下降沿处，CPU 将数据总线上出现的稳定数据送入 CPU 中。

总线周期在 T_4 周期后结束，其他各控制信号和状态信号也进入无效状态。下一个总线周期可能在 T_4 周期结束后立即开始，也可能在 T_4 周期结束后出现若干个 T_I 空闲状态才开始，这取决于 BIU 何时需要进入下一个总线周期。

2. 8086 CPU 写总线周期

8086 CPU 在最小工作模式下的写总线周期时序图如图 2-16 所示，与前述的读总线周期有许多相似之处，下面只说明它们的不同之处。

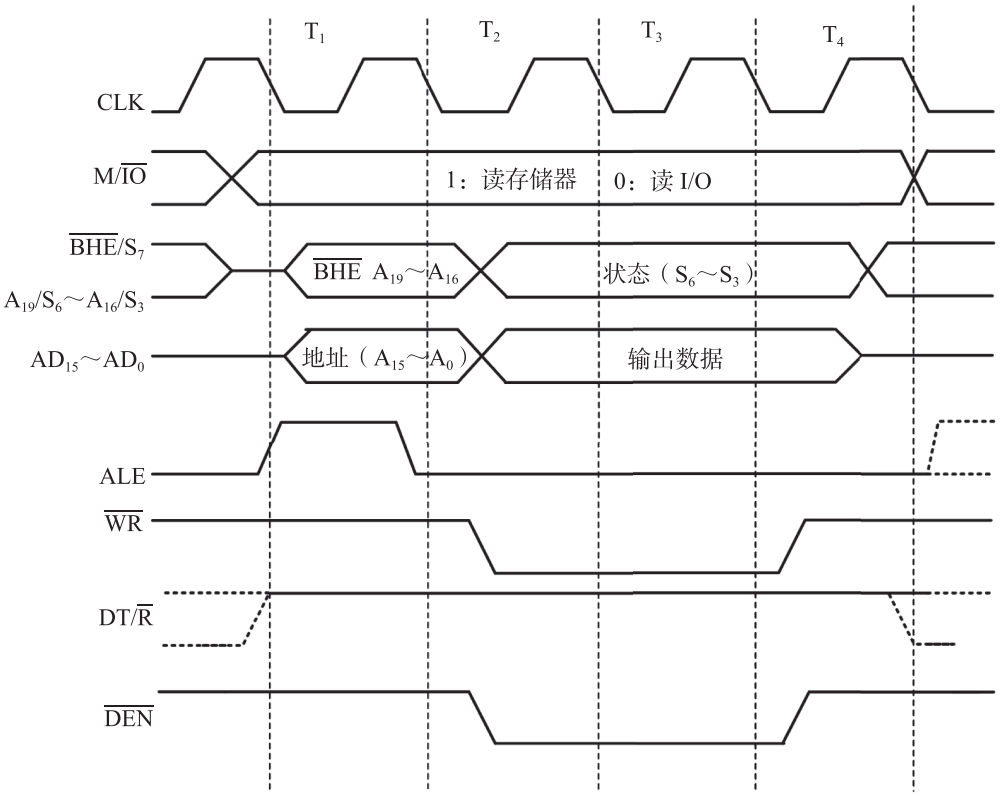


图 2-16　8086 CPU 写总线周期时序图（最小工作模式下）

写总线周期的操作是将从 CPU 输出的 8/16 位数据写到存储器或 I/O 端口的指定地址中。写总线周期中，地址的传送过程与读总线周期完全相同，只是在地址 / 数据复用总线 AD_0～AD_{15} 上，一旦输出地址被锁存后，立即输出数据，并使数据$\overline{WR}$有效，向存储器或 I/O 端口发出写命令，要求将数据总线上的数据写入指定

的存储单元或 I/O 端口中。同样，根据存储器或 I/O 端口的速度，可通过 READY 线要求 CPU 在 T_3 和 T_4 周期之间插入 T_w 周期，以延长写入过程。此外，在写总线周期中，DT/$\overline{R}$ 线应输出高电平，以便使数字收发器 8286/8287 呈输出状态。

8088 CPU 在最小工作模式下的读 / 写总线周期时序与 8086 CPU 读 / 写总线周期时序基本相同，所不同的仅在于 8088 每次访问存储器或 I/O 端口时，只能读 / 写一个字节，通过 AD_0～AD_7 传送，不存在总线高位有效信号 $\overline{BHE}$。

习　　题

一、选择题

1. 某微型计算机最大可寻址的内存空间为 16 MB，其 CPU 的地址总线至少应有（　）条。

A. 26　　B. 28　　C. 20　　D. 22　　E. 24

2.8086/8088 CPU 的 RESET 引脚至少应维持（　）个时钟周期的正脉冲宽度才能有效复位。

A. 4　　B. 5　　C. 2　　D. 3

3. 当 RESET 信号进入高电平状态时，将使 8086/8088 CPU 的（　）寄存器初始化为 0FFFFH。

A. SS　　B. DS　　C. ES　　D. CS

4. 8086/8088 CPU 与慢速存储器或 I/O 接口之间进行数据传输时，为了使传送速度匹配，有时需要在（　）状态之间插入若干等待周期 T_w。

A. T_1 和 T_2　　B. T_2 和 T_3

C. T_3 和 T_4　　D. 随机

5. 在 8086/8088 CPU 中，标志寄存器的主要作用是（　）。

A. 检查当前指令的错误　　B. 纠正当前指令执行的结果与错误

C. 决定是否停机　　D. 产生影响或控制某些后续指令所需的标志

6. 8086 最小模式下的存储器读周期中地址锁存发生在总线周期的（　）时刻。

A. T_1　　B. T_2　　C. T_3　　D. T_4

7. 指令指针寄存器 IP 的作用是（　）。

A. 保存将要取的下一条指令的偏移地址

B. 保存 CPU 要访问的内存单元地址

C. 保存运算器运算结果的内容

D. 保存正在执行的一条指令

8. 8086 CPU 有两种工作模式，最小模式的特点是（　　）。

A. CPU 提供全部控制信号　　B. 由编程进行模式设定

C. 不需要 8282 收发器　　D. 需要总线控制器 8288

二、填空题

1. 在 8086/8088 CPU 中，由 _____ 计算出 _____ 位偏移量部分并送 _____，由 _____ 最后形成一个 _____ 位的内存单元物理地址。

2. 8086/8088 CPU 在总线周期的 T_1 时刻，用 A_1/S_6～A_{16}/S_3 输出 _____ 位地址信息的最高 _____ 位，而在其他时钟周期，则输出 _____ 信息。

3. 8086/8088 CPU 复位后，从 _____ H 单元开始读取指令字节，一般这个单元在 _____ 区中，在其中设置一条 _____ 指令，跳转到初始化程序，使 CPU 对系统进行初始化。

4. 在 8086 系统的存储体系结构中，1 MB 存储体分 _____ 个库，每个库的容量都是 _____ 字节，其中和数据总线 D_{15}～D_8 相连的库全部由 _____ 单元组成，称为高位字节库，并用 _____ 作为此库的选通信号。

5. 在 8086/8088 系统中，可以有 _____ 个段起始地址，任意相邻的两个段起始地址相距 _____ 个存储单元。

6. 用段基值及偏移量来指明内存单元地址的方式称为 _____ 。

7. 通常，在 8086/8088 CPU 中，当 EU 执行一条占用很多时钟周期的指令时，或者在多处理器系统中在交换总线控制时总线会出现 _____ 状态。

8. 8086 CPU 使用 _____ 条地址线访问 I/O 端口，最多可访问 _____ 个字节端口，使用 _____ 条地址线访问存储单元，最多可访问 _____ 个字节单元。

9. CPU 取一条指令并执行该指令的时间称为 _____ 周期，它通常包含若干个 _____ 周期，而后者又包含若干个 _____ 周期。

10. 设内存中一个数据区的起始地址是 1020H：0A1CBH，在存入 5 个字数据后，该数据区的下一个单元的物理地址是 _____ 。

11. 在 8086 系统中，对指令寻址由寄存器 _____ 和 _____ 决定，而默认方式下堆栈段中的偏移量可由寄存器 _____ 或 _____ 来指示。

12. 8086 的中断向量表位于内存的 _____ 区域，它可以容纳 _____ 个中断向量，每一个向量占 _____ 个字节。

13. 在 8086 CPU 中，典型总线周期由 _____ 个时钟周期组成。其中 T_1 期间，CPU 输出 _____ 信息；必要时，可以在 _____ 两个时钟周期之间插入 1 个或多个 T_w 等待周期。

三、简答题

1. 8086/8088 CPU 分为哪两个部分？如何协调工作？

2. 8086/8088 CPU 的地址总线有多少位？其寻址范围是多少？

3. 8086/8088 CPU 使用的存储器为什么要分段？怎么分段？

4. 8086/8088 CPU 中有几个通用寄存器？有几个变址寄存器？有几个指针寄存器？通常，哪几个寄存器也可作为地址寄存器使用？

5. Intel 8086 CPU 与 8088 CPU 有何区别？

6. 8086/8088 CPU 工作在最小模式时，当 CPU 访问存储器时，要利用哪些信号？当 CPU 访问外设接口时，要利用哪些信号？且说明各引脚的有效状态。

7. 试指出下列运算后的各个状态标志，并说明进位标志和溢出标志的区别。

（1）1278H+3469H；

（2）54E3H−27A0H；

（3）3881H+3597H；

（4）01E3H−01E3H。

8. 什么是逻辑地址？什么是物理地址？它们之间有什么联系呢？各用在何处？

9. 设现行数据段位于存储器 0B0000H～0BFFFFH 单元，问 DS 段寄存器的内容为多少？

10. 给定一个存放数据的内存单元的偏移地址是 20C0H；(DS) = 0C0E0H，求出该内存单元的物理地址。