



微计算机丛书

INTEL 8086 程序设计基础

张开善 熊第衡 葛本修 编
徐爱卿 朱淑桃 裴 珉

电子工业出版社

INTEL8086程序设计基础

张开善 熊第衡 葛本修 编
徐爱卿 朱淑桃 裴 珉

电子工业出版社

内 容 简 介

本书对8086微型计算机系统的硬件和软件作了较全面地分析。着重阐述了8086汇编语言程序设计的方法。对SDK—86（国产型号TP—86）单板机及其监控程序进行了分析，并对8086宏处理语言作了简要介绍。

本书可供从事微型计算机科研、生产和应用的工程技术人员参考，亦可作为应用8086微型计算机系统的工程技术人员；有关院校及培训班的教材或参考书。

INTEL 8086程序设计基础

张开善 熊第衡 葛本修 编
徐爱卿 朱淑桃 裴 琨

责任编辑：王惠民

电子工业出版社出版（北京市万寿路）
新华书店北京发行所发行 各地新华书店经售
北京市密云县印刷厂印刷

开本：787×1092¹/₁₆ 印张：22.125 字数：510千字

1987年8月第1版 1987年8月第1次印刷

印数：6,000 定价：4.70元

统一书号：15290·433

前 言

计算机技术的迅速发展,使得计算机已渗透到工业、农业、国防、企业管理以及日常生活的各个领域,并已成为国家的经济实力与现代化水平的重要标志。计算机的生产、推广和应用已成为我国四个现代化的战略任务。

自七十年代以来,由于大规模集成电路技术的发展,微型计算机(简称微型机)的发展速度更为迅猛,其中央处理器(CPU)的集成度几乎是每两年翻一番,且性能增长一个数量级。例如,1971—1972年出现的Intel 4004和4040(4位),其集成度为2000晶体管/片;1974—1975年生产的8085(8位),其集成度为9000晶体管/片;1976—1978年生产的8086(16位),其集成度为29,000晶体管/片;而在1980年生产的iAPX 43201(32位),其集成度则达到100,000晶体管/片。现在,各种微型机的年产量已达到百万台,渗透并占领了各个技术领域,为计算机的普及推广和应用开创了广阔的领域。

例如,各种类型的数控机床,过去是使用小、中型计算机系统来完成其数控功能的,这种控制部件不仅成本昂贵、体积庞大,且维护保养也很麻烦。而现在就可以采用微型机——通常仅用一个单板机来实现控制,成本只要几千元,操作、维护也大为简化。

又如,在检测系统中,把微型机与测量仪器、仪表连接起来,它不仅可以存储测量数据,而且可以对测量的数据进行各种处理,大大提高了测量效率,实现了测量自动化和仪表智能化。

微型机不仅已广泛应用于国防工业、重工业和轻工业,同时也为通信技术的发展提供了有利的条件。

微型机的应用为实现仓库管理、旅馆管理、交通管制等信息处理和管理工作提供了现代化的手段。

近年来,我国微型机的科研、生产和应用获得了迅速发展。8位微型机的应用已较为广泛;16位微型机亦已开始科研和生产等领域中应用。16位微型机的功能可与小型机(如PDP-11/45, 70以及IBM 370/145)相媲美,而成本却只有小型机的几分之一,体积也小得多。

为了在工程技术人员中普及推广16位微型机的基础知识,我们收集了近年来国内外有关8086系统的资料,并对它们进行了分析、综合和整理。在对其中某些问题进行探讨和验证的基础上编写成这本书。本书从8086微处理器和8086微型机的系统结构出发,着重阐述8086的汇编语言程序设计。本书的对象是既考虑到具有一定微型机知识的读者,也照顾到那些自学或使用8086微型机的工程技术人员。对于初学者来说,在掌握了8086微处理器的结构及其指令系统,并对程序设计的一些特点、基本方法有所了解的基础上,逐步自行设计一些程序,由简到繁,通过上机实践,不难掌握8086的汇编语言程序设计。对已在使用或设计8086系统的读者,本书提供了各种程序达一百多个,且其中的大部分程序均已验证,可供读者选用或参考。

本书共分十章，其中第一、二、三章由熊第衡、葛本修编写；第四、八、十章由徐爱卿编写；第五章由朱淑桃编写；第六、七章由葛本修、裴珉编写；第九章由葛本修、朱淑桃编写。全书由张开善主持编写，熊第衡、葛本修审核整理。

本书在编写过程中，杨亚藩等同志曾给了我们许多帮助，谨此致谢。

由于编者水平有限，加之时间仓促，书中的缺点、错误在所难免，热诚欢迎广大读者予以指正并提出宝贵意见。

编 者

一九八四年一月于北京

目 录

前 言	(i)
第一章 8086 的结构	(1)
第一节 存储器	(1)
一、存储器的结构	(1)
二、存储器的分段	(3)
第二节 输入/输出的结构	(5)
第三节 寄存器结构	(5)
一、通用寄存器	(5)
二、指针和变址寄存器	(6)
三、分段寄存器	(7)
四、标志寄存器	(8)
五、8086内部结构的特点	(8)
第四节 指令操作数和操作数寻址方式	(10)
一、单地址指令的寻址	(10)
二、双地址指令	(13)
第五节 关于操作数寻址方式的说明	(15)
第二章 8086指令系统	(19)
第一节 数据传送指令	(19)
一、通用数据传送指令	(20)
二、累加器专用传送指令	(22)
三、地址目标传送指令	(23)
四、标志传送指令	(24)
第二节 算术运算指令	(26)
一、算术运算的数据格式	(26)
二、标志寄存器	(28)
三、加法指令	(29)
四、减法指令	(30)
五、乘法指令	(31)
六、除法指令	(33)
第三节 逻辑运算指令	(34)
一、逻辑指令	(34)
二、移位/循环指令	(36)
第四节 字符串操作指令	(37)
一、基本串操作	(38)
二、说明	(38)
第五节 转移指令	(39)

一、无条件转移、调用和返回指令	(39)
二、条件转移指令	(43)
三、迭代控制转移指令	(44)
四、中断指令	(45)
第六节 处理器控制指令	(47)
一、标志位操作指令	(47)
二、HLT: 暂停指令	(47)
三、NOP: CPU空操作	(47)
四、WAIT: 等待指令	(47)
五、ESC处理器交权指令	(47)
六、LOCK: CPU总线封锁指令	(48)
第七节 标志位置位状态小结	(49)
第三章 8086系统设计	(51)
第一节 8086系统结构	(51)
一、总线结构	(51)
二、地址锁存	(52)
三、双向总线驱动器	(53)
四、时钟发生器	(54)
五、存储器部件	(54)
六、输入/输出端口	(58)
七、中断服务	(59)
八、8086的外围芯片简介	(63)
第二节 8086系统介绍	(63)
第三节 8086的定时	(69)
一、指令周期、处理器总线周期和T状态	(69)
二、8086处理器的定时	(70)
三、分析处理器时序的目的	(71)
四、系统时序分析	(73)
第四章 8086宏汇编语言及程序设计方法	(75)
第一节 8086汇编语言的特点	(75)
第二节 标记与表达式	(76)
一、标识符	(76)
二、保留字	(76)
三、定义符	(77)
四、表达式	(77)
第三节 数据的定义、初始化和访问	(79)
一、数据项及其属性	(79)
二、常数的定义方法	(79)
三、变量的定义方法	(79)
四、变量的访问	(82)
五、标号	(84)

六、记录的定义方法	(85)
七、记录的访问及应用	(88)
八、结构的定义和预置	(89)
九、结构的访问及应用	(90)
十、属性操作符	(92)
第四节 8086汇编语言程序的结构	(95)
一、程序的分段控制方法	(96)
二、段的寻址性	(97)
三、段寄存器的加载	(98)
四、段的前缀(段的修改)	(99)
五、段的隐含访问	(100)
六、GROUP(群)伪指令	(101)
七、PROCEDURE(过程)伪指令	(101)
八、程序模块的连接	(102)
九、END(结束汇编)伪指令	(104)
十、ORG(起始点)伪指令和程序计数器\$	(105)
十一、EVEN(偶)伪指令	(105)
第五节 几个程序结构	(105)
一、总的代码容量小于64K字节, 总的数据和堆栈容量也小于64K字节的程序结构	(105)
二、总的代码和堆栈容量小于64K, 而数据的容量大于64K的程序结构	(106)
三、代码容量大于64K, 而数据及堆栈的容量小于64K的程序结构	(107)
四、代码、数据段都大于64K的程序结构	(107)
第五章 中断技术	(109)
第一节 一般概念	(109)
一、中断的一般概念	(109)
二、中断类型和中断结构	(110)
三、中断的一般工作过程	(111)
四、CPU对中断的响应	(113)
第二节 8259A可编程中断控制器(PIC)	(113)
一、8259A的基本功能	(113)
二、8259A的编程	(115)
三、8259A的级联	(117)
第三节 8086的中断矢量结构和排队电路	(119)
一、预先确定的中断	(119)
二、用户确定的软件中断	(121)
三、用户确定的硬件中断	(121)
四、中断矢量表	(122)
五、8086的中断处理过程	(124)
六、断点中断	(124)
七、8086的中断控制逻辑	(125)
八、中断响应时序	(126)

第四节 中断程序设计和实例	(126)
一、断电保护	(126)
二、A/D转换器与系统的连接	(133)
三、实时钟中断程序	(138)
四、日历钟中断程序	(140)
第六章 输入输出接口技术	(141)
第一节 输入输出接口方法	(131)
一、累加器I/O方法	(141)
二、存储器映象I/O方法	(142)
第二节 8251与8255接口控制器的程序管理	(143)
一、8251可编程的串行接口控制器 (PSIC)	(143)
二、8255可编程的并行接口控制器 (PPIC)	(147)
第三节 累加器I/O程序举例	(153)
第四节 输入输出的应用程序举例	(164)
第七章 数据运算与代码转换	(175)
第一节 十进制数的运算程序	(175)
一、两个多位十进制数相加	(175)
二、对存储器中的十进制数取补	(176)
第二节 浮点数的运算	(178)
一、浮点数运算的方法	(178)
二、浮点数的基本操作子程序	(179)
第三节 初等函数运算程序举例	(190)
一、求三角函数的两个子程序	(190)
二、计算平方根的子程序	(196)
第四节 8087数据处理器 NDP	(198)
一、8087的结构以及与8086CPU的连接	(198)
二、NDP处理器的数据格式	(202)
三、NDP指令系统	(203)
四、8087NDP处理器应用程序举例	(203)
第五节 代码转换	(213)
一、二进制代码转换为八进制的ASCII码	(213)
二、4位十进制数转换为二进制代码	(214)
三、6位十进制数至二进制数的转换程序	(217)
第八章 实用程序及分析	(218)
第一节 过程调用及参数传送程序	(218)
一、通过寄存器传送参数	(218)
二、通过存储器传送参数	(221)
三、通过堆栈传送参数	(221)
四、结构的参数传送	(230)
第二节 控制程序转移的实用程序	(236)
一、外部输入控制开关量的方法	(236)

二、程序内部设置状态字的方法	(136)
第三节 延时程序	(241)
一、简单的延时和延时程序	(241)
二、嵌套的延时程序	(243)
三、关于计算指令和程序执行时间的几个问题	(246)
第四节 多精度数值运算程序	(247)
一、双精度数的求补	(247)
二、双精度数移位程序	(248)
三、多精度数的加减程序	(250)
四、长度不同的多精度数的加法程序	(251)
第五节 设置及清除单步工作方式的程序	(254)
第六节 设置断点的程序	(257)
第七节 字符串处理程序	(263)
一、数据块移动程序	(263)
二、码型转换程序	(267)
第八节 输入/输出程序	(270)
一、键盘输入显示输出程序	(271)
二、存储器映象的输入/输出程序	(271)
第九章 SDK-86单板计算机及其监控和调试程序分析	(279)
第一节 SDK-86单板机概述	(279)
第二节 SDK-86单板机的结构	(280)
一、时钟发生器	(280)
二、等待状态发生器	(280)
三、中央处理单元	(281)
四、并行I/O端口	(281)
五、随机存取存储器	(282)
六、可编程只读存储器	(283)
七、I/O译码器	(284)
八、离板译码器	(284)
九、键盘/显示	(286)
十、串行接口	(286)
十一、总线扩展	(288)
第三节 SDK-86监控程序分析	(289)
一、监控程序概况	(289)
二、键盘监控程序分析	(290)
第十章 宏处理语言及程序设计	(315)
第一节 概述	(315)
一、宏处理语言的概念	(315)
二、宏处理和宏处理程序	(315)
三、宏处理语言中的标识符	(316)
四、宏处理语言中的表达式和数	(316)

第二节 宏指令的定义、调用及其扩展.....	(317)
一、不带参数的宏指令.....	(317)
二、带参数的宏指令.....	(318)
三、宏体中有标号的宏指令.....	(320)
第三节 内部宏指令的调用及扩展.....	(321)
一、计算及设置数值的宏指令.....	(321)
二、字符串比较宏指令.....	(322)
三、控制程序流向的宏指令.....	(323)
四、串处理宏指令.....	(325)
五、控制台I/O宏指令.....	(327)
六、其他宏指令.....	(327)
附录 8086指令系统总表.....	(329)
参考书目.....	(343)

第一章 8086的结构

计算机的结构就是指组成这台计算机的功能部件以及它们之间的配合和联系。它所涉及的问题是：在计算机中有多少寄存器；这些寄存器具备哪些功能；能连接多少存储器；存储器如何寻址，以及能使用哪些类型的输入/输出设备。

8086是一个单片处理器，它具有四组寄存器。第一组是通用寄存器，它们是用来存放中间结果的；第二组是指针和变址寄存器，它们用来在存储器的规定段区中给信息定位；第三组是段寄存器，它们用来确定存储器的段区；第四组是指令指针。在8086中还有九个标志位，这些标志位用来记录处理器的状态和控制它的操作。8086可以访问1兆字节的存储器和65,000个输入或输出端口。本章的前半部分将对这些特性作详细地说明。

典型的计算机指令需要对被处理的操作数 (Operands)，即被处理的数据进行定位，并对这些操作数的数值执行规定的操作，还要将结果存放到某个规定的结果单元。操作数和结果的存放单元可以由指令指定在存储器或寄存器中。通过指令的说明，确定并找到这些存储单元有效地址的方法称为计算机的操作数寻址方式 (Operand-Addressing Modes)。8086操作数的寻址方式将在本章的后半部分介绍。有关对不同操作数进行处理的实际指令将在第二章阐述。

第一节 存储器

一、存储器的结构

存储器的每一个存储单元的长度为8位二进制代码，也就是一个字节(Byte)；8086系统的存储器可以具有 2^{20} (近似于1,000,000) 个字节的容量。每个字节指定一个唯一的地址 (无符号数)，因此它的寻址范围是从0至 $2^{20}-1$ (二进制表示为00000000000000000000至11111111111111111111；十六进制表示为00000H至FFFFFH)^①，如图1-1所示。

存储器中任何两个相邻的字节中存放一个16位的字 (Word)，在一个字中的每个字节有一个字节地址；这两个地址中数值较小的那个作为这个字的地址。有关字的例子如图1-2所示。

一个字包含16位。其中的低8位放在较低位的地址中，该地址加1后的单元存放这个字的高8位；如果在存储器中存放一个字节的序列，那么最低8位放在地址较低的单元中，以后依次存放，最高的8位则应放在最高位的地址中。这就是说8086存储器中字

① 以后我们用数字后面加字母B(Binary)表示二进制数；以字母Q (Octal)表示八进制；以字母D (Decimal)或不加字母表示十进制数，用字母H (Hexadecimal)表示十六进制数。

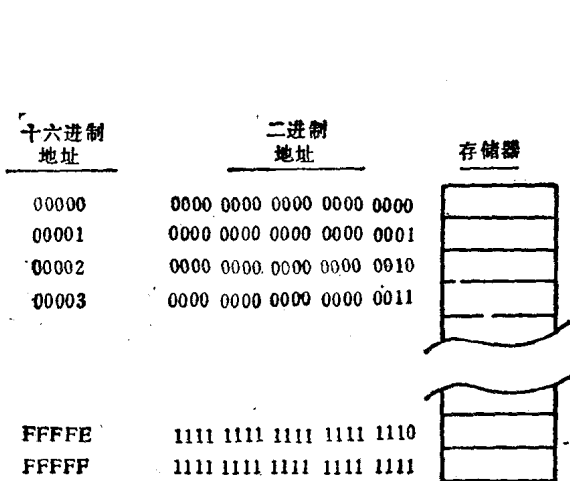


图1-1 存储器地址

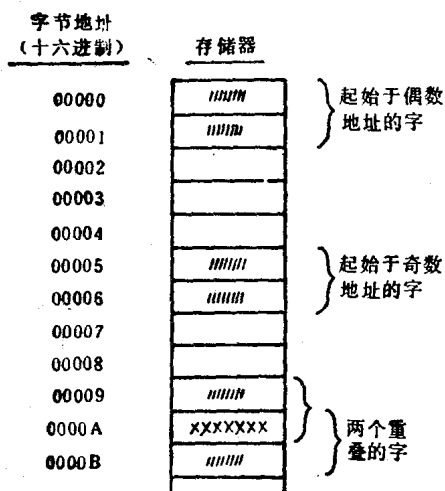


图1-2 存储器中字的例子

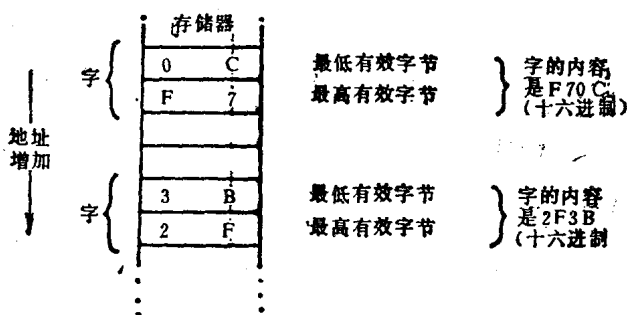


图1-3 存储器中“自下而上”存储的例子

的存放是反向的（或称自下而上存放），如图1-3所示。

8086访问内存的指令有两种方式，一种是字节访问指令，另一种是字访问指令，然而8086每次访问内存储器的信息量总是16位。在执行字节访问指令时，16位信息中仅有8位是有用的，可以是第一个字节，也可以是第二个字节，而其他的8位则视为无效，如图1-4(a)、(b)所示。这16位字在存储器中总是放在以偶数地址起始的两个相邻的字节中，这样在读、写一个以偶数为起始地址的字时，只需访问一次存储器；然而，对于以奇数为起始地址的字，就必须两次访问存储器，读、写两个以偶数为起始地址的两个字。忽略每个字中所不需要的那个字节，并对所须的两个字节进行字节调整，如图1-4(c)(d)所示。但是，它从存储器取出或存入的信息量每次总是16位。在字节指令的情况下，这些位数仅有8位是有用的，而其他的8位则为无效。16位字在存储器中总是存放在以偶数地址起始的两个相邻字节中，这就是说，读或写一个以偶数为起始地址的字的指令，只须访问一次存储器；然而，对于以奇数为起始地址的字的指令，就必须两次访问存储器中的两个偶数地址的字，忽视每个字中所不必要的那半个字，并对剩下的那两个半字进行某种字节调整，各种字节和字的读操作例子如图1-4所示。

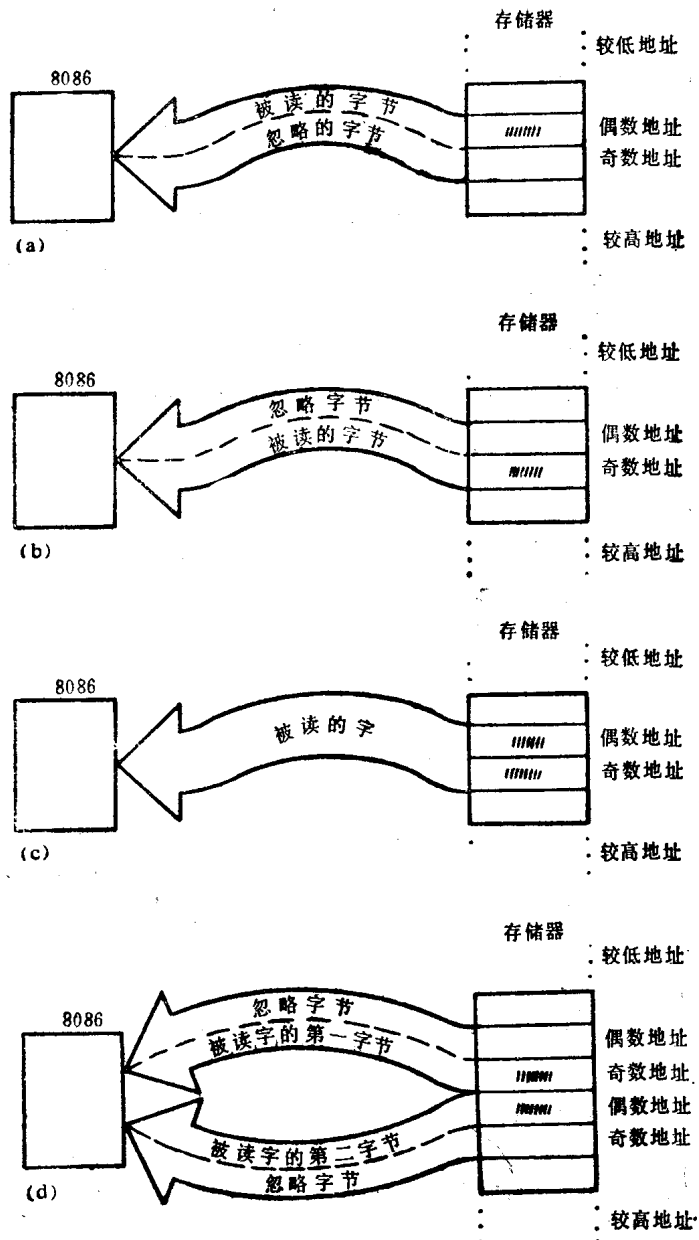


图1-4 从8086存储器的偶数和奇数地址读字节和字。

(a) 读偶数地址中的字节；(b) 读奇数地址中的字节；(c) 读偶数地址中的字；(d) 读奇数地址中的字，要求两次存储器访问

在8086（或8088）程序中，指令仅要求指出对某个字节或字进行访问，而对存储器访问的方式不必说明，无论执行哪种访问，都是由处理器来完成的。

二、存储器的分段

因为8086可以寻址存储器的容量为 2^{20} 个字节。这就是说，在8086处理器中，字节和字的地址必须由20位来表示。但，8086是字长为16位的机器，对地址的运算也只能有

16位长。因此，要求有一个附加的装置进行地址计算。

我们可以将1兆字节的存储器分成若干段；每段包括 2^{16} （近似65,000）个字节。每段的首地址都是一个能被16整除的数字（即字节地址的最低4位都是0）。在任何给定的时刻，程序能够立即访问这样四个段中的内容。这四个段称为：现行代码段（Current Code Segment）、现行数据段（Current Data Segment）、现行堆栈段（Current Stack Segment）和现行附加段（Current Extra Segment），附加段是一个通用区，通常作为一个附加的数据段来看待。我们将这四个现行段的第一个字节的地址的16位最高有效位放入四个专用寄存器中的一个，作为这个现行段的标记。这些寄存器分别称为分段寄存器（Segment Registers）。CS:代码段寄存器；DS:数据段寄存器；SS:堆栈段寄存器；ES:附加段寄存器。各个段不需相等，而且它们可以重叠。有关段的例子如图1-5所示。

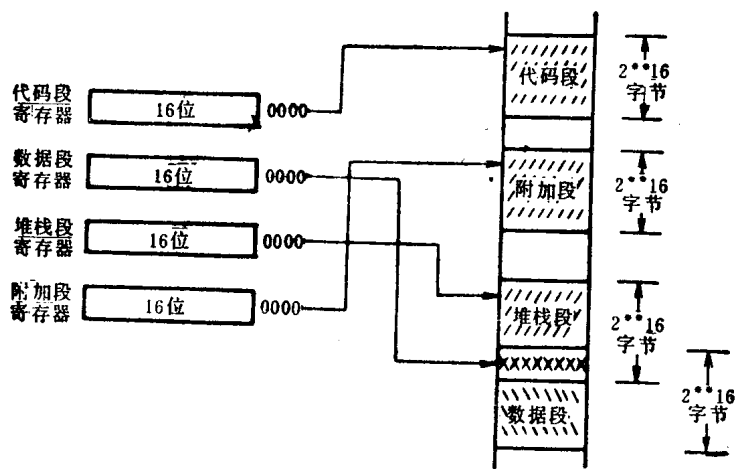


图1-5 段的例子。注意在这个例子中堆栈段和数据段重叠

例如，假设16位的代码分段寄存器包含的数值是C018H，这样代码段的起始字节地址是C018H，而段长度最大可达 2^{16} （10000H）个字节。因此，该代码段的最后字节地址是D017FH。在一个段内，地址则可以由一个16位的偏置量指出。处理器将这个16位偏置地址与分段寄存器的内容相加，并在它的低四位添加四个零后，与它原来的数值相加，构成20位字节或字地址，如图1-6所示。

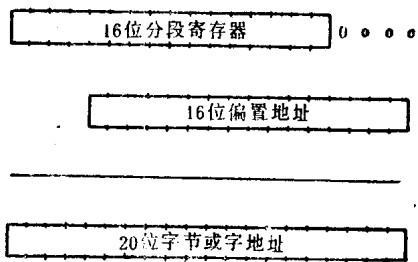


图1-6 字节或字地址结构

因此，在上例中，这个字节在现行代码段中的字节地址是CFFFFH，它在代码段中有一个FE7FH的偏置地址（CFFFFH—C0180H），如图1-7所示。

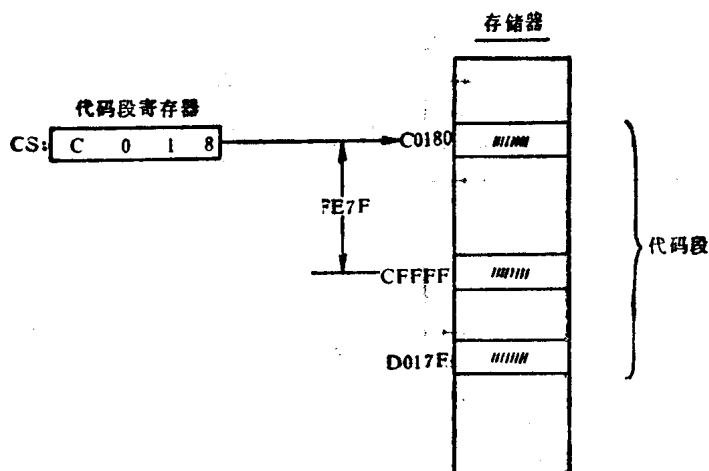


图1-7 字节地址结构的例子

第二节 输入/输出的结构

8086系统与外部环境连接的装置称为端口 (Port)。通过这些端口，8086可以接收外部的信息，并且可以发出信号来控制其他的过程。

8086可以访问类似于存储器字节的 2^{16} (近似于65,000) 个8位端口，每个8位端口由不同的地址确定，它的范围是从0至 $2^{16}-1$ 。任何两个相邻的8位端口可以作为一个相当于存储器字的16位端口来处理，16位奇数地址端口 (对于8088则是所有16位端口) 则要求进行两次访问。事实上，除了它们没有端口分段寄存器以外，端口寻址与存储器字节或字的寻址方式是相同的；换句话说，全部端口可以认为是在一个段中。

第三节 寄存器结构

8086处理器包括十三个16位寄存器和9位标志位。如前所述，这些寄存器可分为四组。其中三组，每组包括四个寄存器，第十三个寄存器称为指令指针，它不能由程序员直接使用，而是由它自身进行置位。8086的寄存器和标志位如图1-8所示。

三组可编程的寄存器是通用寄存器，指针和变址寄存器以及分段寄存器。通用寄存器主要是用来保存算术和逻辑运算的操作数；指针和变址寄存器用来保存段内的偏置地址；分段寄存器用来确定段的起始地址。

一、通用寄存器

处理器中，如果没有通用寄存器，则各个指令就必须从存储器取出它的操作数并将运算结果存入存储器。但是，存储器访问比较费时，若将经常使用的操作数和结果保存在一个能快速存取的地方，就可减少这个存取时间。为此，8086在处理器中设有四个通用寄存器。

8086的16位通用寄存器是AX，BX，CX和DX。每个通用寄存器可分为两个8位寄

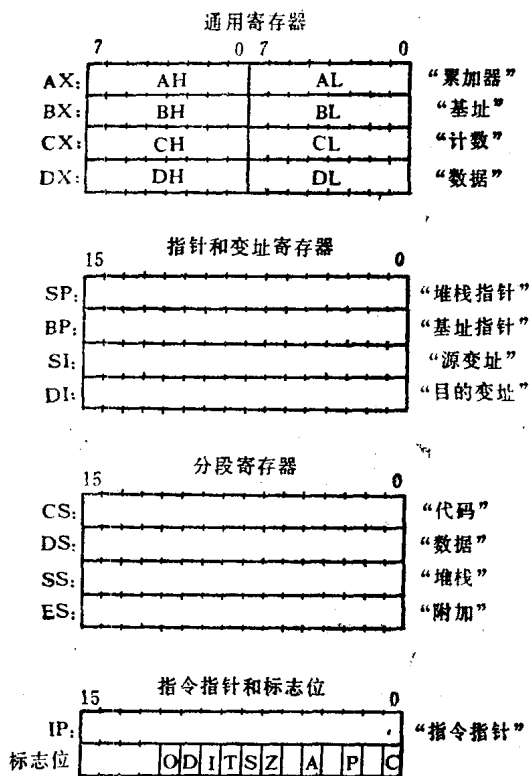


图1-8 8086的寄存器和标志位

寄存器；也可以将这两部分组合成为一个16位寄存器。因此，每个通用寄存器的两个8位寄存器给定了它自己的名称：其低位有效部分的名称分别为AL, BL, CL和DL；而高位有效部分的名称是AH, BH, CH和DH。这些寄存器的双重性使得它们很容易地处理字节和字的信息。

通常，通用寄存器的内容可以交互地参与8086的算术和逻辑运算。例如，加法指令（ADD）可以是两个具有相同位数的通用寄存器的内容进行相加，而将其结果存放在其中的某个寄存器中。但是，某些指令规定某个通用寄存器有某种专门的用途。例如，在串操作指令中要求CX寄存器用来作为计数器，记录数据串中的元素的数目。而AX, BX或DX寄存器都不能用于这个目的。CX寄存器的这种特殊用途就可以把它称为计数（Count）寄存器，而对于AX, BX和DX

寄存器可以分别称为累加器（Accumulator），基址（Base）和数据（Data）寄存器。

通用寄存器的这些特殊规则在程序较长的情况下，可以使程序缩短。在编程的过程中要将特殊类型的数据放在特殊的寄存器中，如果我们选择CX寄存器来记录数据串中元素的数目；那么这个数据就总存放在其中，不必将数据串的大小传送到CX。如果不设计这种特殊用途，则各通用寄存器的功能是相同的，每个字符串指令也就必须指出该特殊字段存放在哪个寄存器中，这就使得每条串指令变长，由一字节指令变为两字节指令；或者需要采用多字节的串指令，这都使得程序变长，因此对某些指令要使用这些专用寄存器，这在减少程序的长度方面是具有切实效果的。

二、指针和变址寄存器

在一条访问存储器单元的指令中，可以直接指出这个单元的地址。这个地址占据指令的一定空间，因而增加了代码的长度。如果将常用单元的地址存放在特定的寄存器中，那么访问这些单元的指令只需指出该寄存器的地址即可。这可以大大缩短指令的长度。这种寄存器往往称为指针或变址寄存器。

8086的指针和变址寄存器由四个16位寄存器SP, BP, SI和DI组成。对于在一个段内的寻址，这些寄存器通常存放的是偏置地址。例如，一条加法指令（ADD），可以规定该指令的一个操作数的地址是在存储器的现行数据段中，而其段内的偏置包含在指定的指针或变址寄存器（譬如SI）中。