福建省高校计算机等级考试规划教材(三级)

Intel 80X86微机原理与接口技术

Foundation and Application of Intel 80X86 Microcomputer

福建省高校计算机等级考试三级（偏硬）考试指导书

福建省高校计算机教材编写委员会 组织编写

余朝琨 编著

厦门大学出版社
XIAMEN UNIVERSITY PRESS

福建省高校计算机等级考试规划教材

Intel 80X86 微机原理与接口技术

——福建省高校计算机等级考试三级(偏硬)

余朝琨　编著

厦门大学出版社

图书在版编目(CIP)数据

Intel 80X86 微机原理与接口技术:福建省高校计算机等级考试三级(偏硬)考试指导书. 第 3 分册/余朝琨编著. —厦门:厦门大学出版社,2009.10
ISBN 978-7-5615-3363-5

Ⅰ.I… Ⅱ.余… Ⅲ.①微处理器,Intel 系列-理论-高等学校-教材②微处理器,Intel 系列-接口-高等学校-教材 Ⅳ.TP332

中国版本图书馆 CIP 数据核字(2009)第 176334 号

厦门大学出版社出版发行
(地址:厦门市软件园二期望海路 39 号 邮编:361008)
http://www.xmupress.com
xmup @ public.xm.fj.cn
南平市武夷美彩印中心印刷
2009 年 10 月第 1 版 2009 年 10 月第 1 次印刷
开本:787×1092 1/16 印张:15
字数:352 千字 印数:1~2000 册
定价:22.00 元

内容简介

本书是根据福建省高校计算机等级考试指导委员会审核批准的《福建省高等学校计算机应用水平等级考试三级(偏硬)考试大纲》(2008 年版)要求编写而成的。内容包括:80X86 微处理器的 PC 微机系统基础知识,涉及 80X86 汇编语言程序设计的编程基础,CPU 中的寄存器组成与应用,存储器的分段,数的转换,寻址方式,指令系统,伪指令以及编程方法。

在 8086 微机系统扩展与接口应用中,主要介绍计算机输入信息、输出信息、存储信息以及处理信息。介绍 8086CPU 的结构,引脚功能,8086 的两种基本组态及系统总线的基本时序。

在 80X86 系统的存储器体系结构中主要介绍存储器的分类,存储器的扩展,高速缓存以及虚拟存储器的工作原理。

对 PC 机的 IO 接口、中断以及可编程芯片的工作原理与具体应用都有实例分析。

全书涵盖考试大纲中 Intel 80X86 编程的基础知识及常用的接口应用,是参加福建省高等学校计算机应用水平等级考试三级(偏硬)应试者必备的辅导材料。本书从围绕实际应用所需的知识结构进行编写,突出实用性,可作为电子工程、自动化控制、机电工程等相关专业学生学习汇编语言程序设计用书,也可供从事微机及其应用系统设计开发的广大工程技术人员学习参考。

前 言

计算机应用中有一类称为“计算机(偏硬)”的应用,这类应用的特点是:将计算机作为智能处理核心,针对应用对象的信息处理需要,构造精简的专用的计算机硬件和编制针对性很强的专用软件。这类应用的可靠性要求极高,成本价格敏感,外形尺寸千变万化,一般不能将通用的计算机,如 PC 微机,搬到设备中,往往需要根据应用量体裁衣,设计、制作专用的计算机硬件系统,并且根据应用需要编写精炼的专用程序。

福建省计算机等级考试委员会专门设置了“计算机三级(偏硬)等级考试”,所考核和要求的知识构成紧密围绕计算机的这类应用。这个考试的宗旨是为了鼓励和促进大学生注意自己在本领域内的知识与能力的培养,引导大学相关计算机应用教学课程的改革与发展。将众多的信息领域相关知识划分为“计算机(偏硬)”与“计算机(偏软)”,前者包含必要的电子器件知识、典型微型计算机系统组成及工作原理、汇编语言与高级语言编程、计算机应用系统扩展和接口设计;后者包含数据结构、计算方法、数据库理论、操作系统、软件工程与程序设计等知识。这种分类仅是为了便于学习和设置考试。

本书是计算机三级(偏硬)等级考试指导书的第三分册,内容包括 80X86 微处理器的 PC 微机系统的基础知识,涉及汇编语言编程的基础知识与编程实例;对接口技术及常用接口芯片进行介绍;对存储器的分类与扩展、高速缓存与虚拟存储器的基本概念介绍;对中断技术做了较详细叙述;在键盘技术中主要介绍 LED 的应用。全书主要突出与考试大纲相关的知识,是应用性的介绍,实例较多,在各章的习题中都附有相应的题型,以便于同学们复习时参考,书末附有主要习题的参考答案。读者要进一步深入学习,可参考其他教科书或参考书。

本书是在福建省高校计算机等级考试指导委员会的具体指导下编写的,得到吴锤红、陈庆强老师以及厦门大学出版社的大力支持。由于编者的水平有限,书中难免存在不妥甚至谬误之处,敬请读者不吝赐教为感。

余朝琨

2009 年夏

目 录

第 1 章　80X86 微处理器的 PC 微机系统基础

了解 8086CPU 的构成，为熟悉计算机的内部结构，以及与机器结构紧密相关的指令系统，这对编写汇编语言程序都是很重要的。

1.1　Intel 8086 微处理器的功能结构

1.1.1　8086CPU 的基本结构

图 1-1 是 8086CPU 中结构示意图。

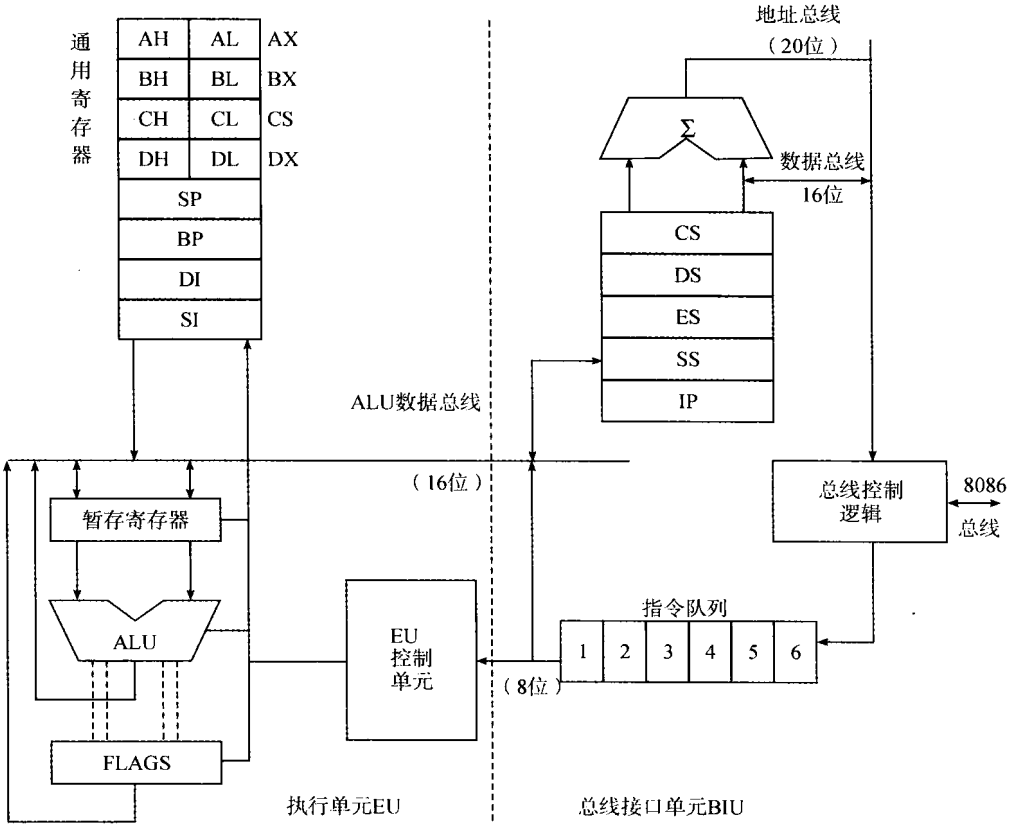


图 1-1　8086CPU 的结构框图

汇编语言程序设计者必须了解地址空间、寄存器组、寻址方式、指令系统等。

8086CPU 是 16 位微处理器，采用 16 位数据总线。8088CPU 是准 16 位的微处理器，采用 8 位数据总线，而使用 16 位内部总线。8086 具有 6 个字节指令流队列，8088 则是 4 个字节，

两者有相同的指令系统。

1.1.2 8086CPU 结构

8086CPU 由指令执行部件 EU 与总线接口部件 BIU 两部分组成，其结构示意图见图 1-1。

EU 部件(Execution Unit)控制和执行指令，主要由算术逻辑部件 ALU、EU 控制部件、8 个 16 位寄存器和一个标志状态寄存器 FLAGS 组成。

BIU 部件(BUS Interface Unit)负责从存储器预取指令和数据，以及所有 EU 需要的总线操作，实现 CPU 与存储器和外设之间信息传递。BIU 主要由指令队列、指令指针寄存器、段寄存器、地址加法器(形成 20 位的物理地址)组成。

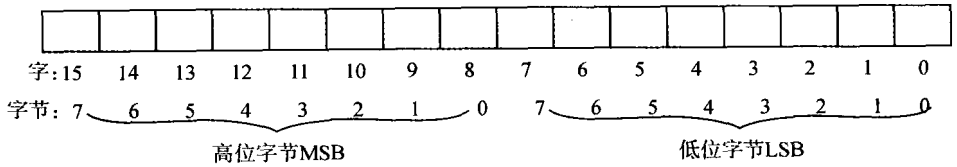
EU 和 BIU 能独立运行，在一条指令的执行过程中，就可取下一条指令送入指令队列，实现流水操作，提高指令运行速度，因为 EU 与 BIU 可实现并行操作。

1.2 8086 的存储器

存储器是计算机的记忆部件，用来存储信息。访问主存是物理地址，而编程者使用逻辑地址编写程序，这就存在逻辑地址转换成物理地址的过程，为此要理解与掌握存储器的分段，这也是汇编语言程序设计的重要基础。

1.2.1 存储单元的地址和内容

计算机存储信息的基本单位称为存储元件，每个存储元件是一个二进制位，一位可存储一个二进制数 0 或 1。每 8 位组成一个字节。由于 8086 为 16 位数据总线，则字长为 16 位，由 2 个字节组成，位编号如下：



一般存储器(Memory)是以字节为单位存储信息。在按字节编址的计算机中，为了正确地存、取信息，每个字节单元都给予一个存储器地址。

尽管存储器是由字节单元组成的，但是任何两个相邻字节可以构成一个字(Word)，即 16 位二进制代码。显然一个字有二个地址，约定用地址值较小的那个字节单元地址作为这个字单元地址的代表。一个字有两个字节，其中低字节数在相应的低地址中，高字节数在高地址中。

1.2.2 存储器地址的分段

8086/8088 CPU 只有 20 根地址总线 $A_{19}A_{18}\cdots A_1A_0$，可访问的存储器最大容量为：$2^{20}$B=1 048 576 B=1 024 KB=1 MB

8086 CPU中用来存放地址的寄存器都是16位的。那么在16位字长的机器里，用什么办法来提供20位物理地址呢？在IBM-PC机里采用了存储器地址分段的办法来解决。

把IMB的存储空间划分成若干个段(Segment)，每个段可由1～64 KB(即65 536B)个连续的字节单元组成。每个段是一个可独立寻址的逻辑单位。在8086/8088的程序设计中，需要设立几个段，每个段有多少个字节以及每个段的用途完全由用户自己确定。同时每个段中存储的代码或数据，可以存放在段内任意单元中。

一个存储器可以划分为若干个段，一般每个段的起始单元的首地址不是任意的，而是有所限制，它必须从任意的小段(Paragraph)的首地址开始。机器规定，从0地址开始，每16个字节划分为一个小段。IMB可分成65536个小段。观察下面每小段的首地址的特性：

第1小段：00000H，00001H，00002H，……0000EH，0000FH

第2小段：00010H，00011H，00012H，……0001EH，0001FH

第3小段：00020H，00021H，00022H，……0002EH，0002FH

…… …… ……

第65535小段：FFFE0H，FFFE1H，FFFE2H，……FFFEEH，FFFEFH

第65536小段：FFFF0H，FFFF1H，FFFF2H，……FFFFEH，FFFFFH

从每一小段的首地址特点可知其 $A_3A_2A_1A_0$ 都为0，这样，我们在分段中段寄存器CS，DS，SS，ES只须保存地址总线高16位的值，即 A_{19}～A_4 值。因 $A_3A_2A_1A_0$ 肯定为全零，也就是16进制的一个0，每一个小段物理存储器(Physical memory)中都有一个这样的起始地址，我们称为段基址(Segment Base Address)。每个段的段基址只能是上述64K个小段的首地址中之一。

在程序设计中所设置的段称为逻辑段，各个逻辑段在物理存储器中可以是邻接的(Contiguous)，互为间隔(Disjoint)，部分重叠(Party overlapped)和完全重叠的(Full overlapped)等4种情况。

虽然存储器可以划分成若干个段，但在任何时刻，8086的一个程序只能访问4个段中的内容，这4个段分别是：

代码段(Code Segment)：源程序必须在代码段中编写，立即数也存放在代码段中；

堆栈段(Stack Segment)：存放数据、给程序调用或中断存放返回地址、传递数据等信息，用SP指示的堆栈操作原则是“先进后出”，用BP为地址可访问堆栈中任何单元；

数据段(Data Segment)和附加段(Extra Segment)：都是存放数据，在串操作中，数据段中默认的是源串，而附加段中一定是目标串。

它们的段首地址的段基值分别由对应的4个段寄存器CS，SS，DS，ES指明。它们分别保存各自段首地址的高16位值(A_{19}～A_4)，由4个段寄存器指向的那些段叫当前段(Current Segment)。所以当前段至多可容纳64 KB的程序代码，64 KB的堆栈和128 KB的数据(分别由DS、ES指向的当前段)。在规模不是很大的应用程序中，这些容量是足够使用。如果应用规模较大，可以在程序中通过修改相应段寄存器的内容，从而访问其他段，如可用LDS，LES指令等方式来改变当前段。

1.2.3 逻辑地址与物理地址

CPU 访问主存必须是物理地址，用户编程使用逻辑地址，于是在 8086/8088 系列微型机中，每个存储单元都有两种形式的地址：物理地址(Phsysical Address)和逻辑地址(Logical Address)以及 CPU 内如何把程序员使用的逻辑地址转成物理地址(真实地址)的问题。

每一个存储单元的物理地址是唯一的，就是这个单元的地址编码。CPU 与存储器之间的任何信息交换，都必须使用 20 位的物理地址，经地址译码器后形成开门信号，把被访问的存储单元的“门”打开，方能进行数据交换。

一个逻辑地址是由段基值和偏移量(OFFSET)两部分组成，而且都是无符号的 16 位二进制数。段基值是一个段首地址的高 16 位，它存放在某一个 16 位段寄存器中。段首址的低 4 位为 0，即 $A_3A_2A_1A_0=0000$。偏移量表示了某存储单元与它所对应的段的段首地址之间的字节距离。如当偏移量为'0'时，就在这个段的起始单元，当偏移量为 0FFFFH 时，就是这个段(最大)最末一个字节单元，表明与首地址相差的字节数为 65 535 个。

当 CPU 访问存储器时，总线接口部件 BIU 便把逻辑地址转换成物理地址。转换方法是：首先把逻辑地址中的段寄存器保存的段基值左移 4 位，或称把段寄存器的内容乘以 16，其本质是恢复 $A_3A_2A_1A_0$ 四位二进制的 0 或一位 16 进制的 0，使之形成一个完整的段首地址。这也就是为什么段首址要从小段的首地址开始的原因。形成了 20 位的段起始地址(段基址)之后，再加上 16 位的无符号偏移量，就形成 CPU 访问主存单元的 20 位物理地址，从地址总线输出。

物理地址＝段基值×16＋偏移地址

```
15              0
┌──────────────┐
│ 16 位段地址  │  0000
└──────────────┘
        15              0
 ＋)   ┌──────────────┐
       │ 16 位段地址  │
       └──────────────┘
19                      0
┌──────────────────────┐
│ 20 位物理地址        │
└──────────────────────┘
```

存储单元只有唯一的一个物理地址，但它却可由不同的段基值和不同的偏移地址组成。

1.3 8086 的寄存器

寄存器是汇编语言编程的资源之一，只要充分了解寄存器的使用方法，才能正确使用寄存器资源，编好程序。

1.3.1 8086 寄存器结构及其用途

对 CPU 中各寄存器的使用及寄存器的默认用法的掌握至关重要。

Intel 8086/8088CPU 共有 14 个十六位寄存器，见图 1-2。

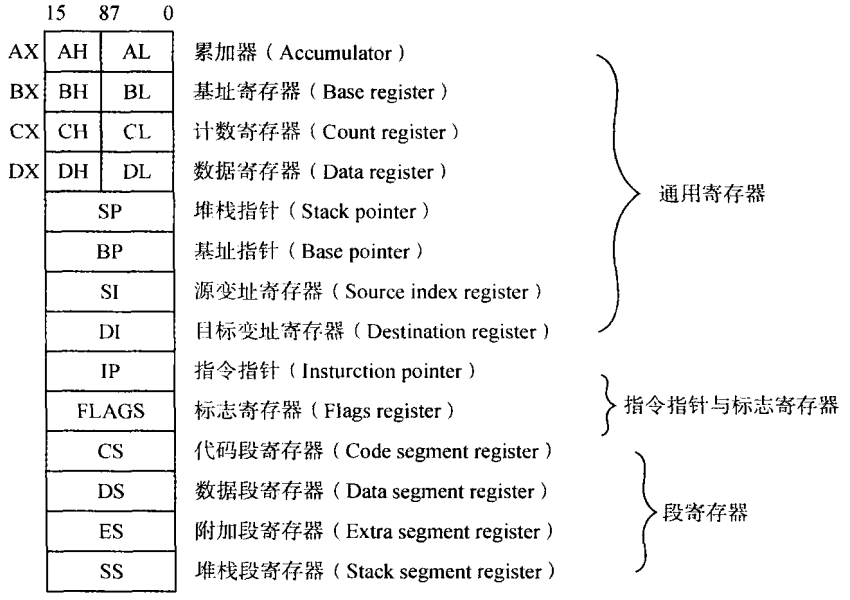


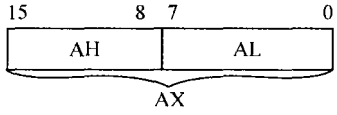
图 1-2　Intel 8086/8088CPU 内部寄存器

1.3.2　通用寄存器(General Register)

通用寄存器共有 8 个，根据使用情况又可分为三类。

1. 数据寄存器

AX、BX、CX、DX 四个寄存器中的每一个，既可以作为十六位寄存器来使用，同时也可以作为两个 8 位寄存器来使用，如：AX 可看成为：



因此，这四个十六位寄存器也可以看成 8 个独立的八位寄存器 AH、AL、BH、BL、CH、CL、DH、DL，它们分别由十六位寄存器的高八位和低八位构成。在汇编语言程序设计中，由编程人员根据需要任意选用，这些寄存器既可存放参加运算的操作数，也可存放运算的结果，具有良好的通用特性，这就是这些寄存器的共性。

寄存器在某些指令中，或在某种场合下又有其默认的用法，这就是个性。通常是：①AX 的默认用法有：AX 为十六位累加器，AL 为八位累加器；在 I/O 指令中必须用 AX 或 AL；AX 与 DX 配合组成 32 位数据寄存器，DX 中存放高 16 位数据等；②BX 常做为基地址寄存器；③CX 在串操作或用循环指令（如 LOOP 等）中的循环计数必须选用 CX；④DX 作为数据寄存器，在 I/O 端口操作中存放端口地址，与 AX 配合形成 32 位数据寄存器。还有一些隐含使用将在指令系统中作进一步说明。总之，寄存器的默认搭配必须认真记住，才不致于违反语法规则。

2. 指针寄存器

堆栈指针(SP)和基址指针(BP)通常用来作为十六位地址指针。SP 是指向堆栈段栈顶存储单元的偏移量,且总是指向栈顶,进栈与出栈的操作(字操作)皆由 SP 来指明偏移地址,堆栈指针 SP 就是这样的隐含使用。

用 BP 作地址指针时,默认的也在堆栈段,用 BP 作地址指针可以对堆栈中任何字节存储单元或字单元进行操作。但 BP 指明的存储单元可允许段跨越。

3. 变址寄存器

两个 16 位变址寄存器 SI、DI,在不同情况下的用法为:

(1)只有在串操作指令中,源串操作数必须用 SI 来提供偏移量,目的串操作数必须用 DI 提供偏移量。对于串操作指令,SI、DI 的作用绝对不能互换,在这种情况下,SI、DI 才是名副其实的源变址寄存器与目的变址寄存器,必须严格按规定使用 SI、DI。

(2)在串指令以外的多数情况下,源和目的变址寄存器,可由用户随意选用,被用来作地址寄存器,在变址寻址中 SI、DI 的内容作为段内偏移量的组成部分。

(3)SI、DI 两寄存器除作地址寄存器外,也可以作为通用数据寄存器使用,存放操作数和运算结果。

1.3.3 段寄存器

8086CPU 有 4 个段寄存器。代码段 CS、数据段 DS、堆栈段 SS 和附加段 ES 寄存器。每个段寄存器可以确定一个段的起始地址,这些段各有用途。代码段主要存放运行的程序;数据段存放运行程序所用的数据,如果程序中使用了串处理指令,则其源操作数默认存放在数据段中;堆栈段定义了堆栈的所在区域,堆栈是一种数据结构,它开辟了一个比较特殊的存储区,并以"先进后出"的方式来访问这一区域,它只有一个出口,并以 SP 堆栈指针指明栈顶;附加段是附加的数据段,它是一个辅助的数据区,在串操作指令中用到目的串必定存放在附加段中。在编制程序时,应该按照上述规定把程序的各部分放在规定的区段之内。

除非专门指定,各段在存储区中的分配是由操作系统完成的。当 CPU 访问某存储单元,如取指令或存取操作数时,就必须指明该存储单元在哪个段寄存器指向的存储段中,同时给出该存储单元在这个存储段内的偏移地址,即偏移量。一个存储单元与它所在段的段基址之间的距离,即字节数叫作该存储单元的偏移量。

在程序运行的任何时刻,最多只能有四个当前段。为了切换当前段,可以用程序的办法更换相应段寄存器的内容。例如:某程序已经设定有两个数据段 DATA1 和 DATA2,当前数据寄存器 DS 正指向 DATA1 段基值时,DATA1 即为当前数据段。若在程序运行中要求访问数据段 DATA2 中某存储单元,这时必须先转换当前数据段,用几条指令让 DS 指向数据段 DATA2 的段基址,把 DATA2 切换为当前数据段。

1.3.4 指令指针与状态标志寄存器

包括 IP 和 PSW 两个 16 位寄存器。

1. 指令指针

IP是指令的地址指针，用来存放指令在代码段中的偏移地址。在程序运行的过程中，它始终指向下一条指令的首地址，称为当前IP，它与CS寄存器联用确定下一条指令的物理地址。8086CPU就是用IP寄存器来控制指令序列的执行流程。

2. 状态标志寄存器

状态标志寄存器PSW(Program Status Word)是一个16位寄存器。用来反映微处理器在程序运行时的某些状态。8086CPU使用PSW寄存器中9位，其中6个状态标志位：OF、SF、ZF、AF、PF、CF。3个控制标志位：DF、IF、TF，这要由编程者设定其值，在执行某些指令时起控制作用。而状态标志所提供的信息往往作为后续条件转移指令的转移条件，所以也称为条件码。状态标志和控制标志在PSW寄存器中的位置见图1-3。

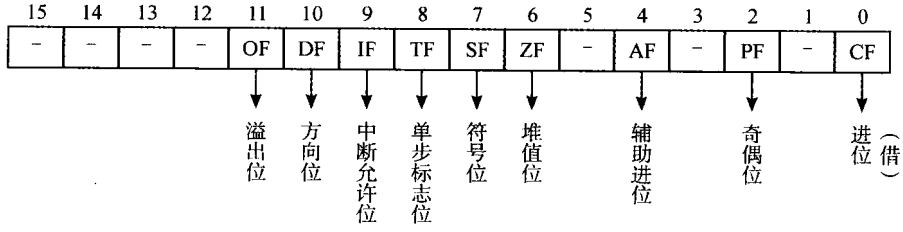


图1-3 PSW寄存器

6位状态标志位。

OF：(Overflow Flag)溢出标志，在运算过程中，当运算结果的数超出机器所能表示的范围则称为溢出，此时OF位置1，否则置0。如以补码表示的机器数范围为：－128～＋127(字节数据)或者－32 768～32 767(字数据)。INTO溢出中断指令就是靠检测OF位的状态而决定执行与否。仅当OF＝1时执行INTO指令。

SF：(Sign Flag)符号标志。SF与运算结果的最高位D_7或D_{15}相一致。

ZF：(Zero Flag)零标志。若运算结果为0时，则ZF置'1'，结果非0则置'0'。

AF：(Auxiliary Carry Flag)辅助进位标志，当进行算术运算时，若低字节中的D_3位产生进位(或借位)时，则AF置成'1'，否则置'0'。它是计算机用于十进制运算调整中使用的，用户不能用。

PF：(Parity Flag)奇偶标志，若操作结果的低8位中含'1'的个数为偶数时，则PF置'1'，否则PF置成'0'。

CF：(Carry Flag)进位标志，当进行算术运算时，如果最高位(对字操作是D_{15}，对字节操作是D_7)产生进位(加法)或借位(减法)，则CF置'1'，否则置'0'。CF也可在移位类指令中使用，用它保存从最高位(左移时)或最低位(右移时)移出的代码(0或1)。

以上六个状态标志中，只有CF可用指令来设置其状态，如CLC指令使CF复位，STC使CF置位，CMC，可使CF求反。

3位控制标志位：

DF：(Direction Flag)方向标志，用在串处理指令中控制处理信息的地址增减方向。当DF位置'1'时(使用STD指令)，每次串操作后变址寄存器SI和DI自动减1(字节操作)或减2(字操

作),这样就使串处理从高地址向低地址方向处理。当 DF 为'0'时,则使变址寄存器 SI 和 DI 自动加 1(字节操作)或加 2(字操作),使串处理从低地址向高地址方向进行。增址方向是默认的。

IF:(Interrupt Flag)中断标志,这个标志位主要针对可屏蔽中断的开放或禁止。当 IF='1'时,CPU 中断开放,IF='0'时,不允许响应可屏蔽中断,用 STI 指令使 IF 置'1',用 CLI 使IF='0'。

TF:又称陷阱标志(Trap Flag),或称跟踪标志(Trace Tlag)或称单步标志位。用于单步方式操作,当 TF='1'时,在执行完一条指令后,产生单步中断。这在 DEBUG 调试程序状态下,可以使指令单步运行,可逐一检查各寄存器内容,标志状态、存储器的检查或修改等等。TF='1'时为调试程序时所用,当程序调试成功后让 TF='0',CPU 正常工作不产生单步中断。

1.4 8086 的 I/O 端口

计算机运行时的程序和数据都需要通过输入设备送给计算机,程序运行的结果也要通过输出设备送给用户,如在显示器上显示或在打印机上输出等,所以输入输出设备是计算机必不可少的组成部分。大容量的外存储器(如硬磁盘或软磁盘、光盘)能够存储大量的信息,也是现代计算机不可缺少的一部分。对于外部设备的管理是汇编语言的重要使用场合之一。

CPU 外连接的各种功能部件,可归纳为两种性质的部件:主存储器和 I/O 端口,I/O 端口也称外部设备,外部设备与主机(CPU 和主存)的通信是通过外设接口进行的。每个接口包括一组寄存器,这寄存器也称为端口或 I/O 端口。这些寄存器最主要的用途是:

(1)数据寄存器:用来存放要在外设和主机间传送的数据,这类寄存器实际上起缓冲器的作用。

(2)状态寄存器:用来保存外围设备或接口的状态信息,以便 CPU 在必要时测试外设状态,了解外设的工作情况,起联络作用。例如,每个设备都有忙闲位用来标志设备当前是否正在工作,是否有空接受 CPU 给予的新任务等,使 CPU 与外设实现异步控制就更为方便。

(3)命令寄存器:CPU 给外设或接口的控制命令通过此寄存器传送给外部设备。例如,CPU 要启动磁盘工作,必须发出启动命令等。

当然外部设备通过接口与主机相连,每个接口中究竟需要如何设置这些寄存器以及这些寄存器的具体数量都是取决于不同类型的接口电路而定。

为使主机访问外设方便起见,外设接口中的每个寄存器给予一个端口(Port)地址,又称为端口号,这样就组成了一个独立于内存空间的 I/O 地址空间。8086/8088 的 I/O 空间可达 64 K个端口地址,在输入输出指令中寻址的外设地址总线使用了 16 根,所以端口地址的范围是 0000H～FFFFH。

1.5 Intel 8086 指令系统

1.5.0 概述

一台计算机能执行什么样操作,能做多少种操作,是由该计算机的指令系统所决定的。因