8088/8086
汇编语言程序设计
学习指导书

刘晓星　潘晓萍　编

8088

8086

中央广播电视大学出版社

8088/8086 汇编语言程序设计学习指导书

刘晓星　潘晓萍　编

中央广播电视大学出版社

(京)新登字 163 号

8088/8086 汇编语言程序设计学习指导书

刘晓星　潘晓萍　编

*

中央广播电视大学出版社出版
新华书店北京发行所发行
北京印刷三厂印装

*

开本 787×1092　1/16　印张15　千字374
1993 年 10 月第 1 版　1994 年 10 月第 2 次印刷
印数　12001～22000
定价 8.95 元
ISBN 7-304-00837-7/TP・41

前　言

本书是中央广播电视大学计算机专业的教材《8088/8086 汇编语言程序设计》的配套教材。本书对主教材内容按章节进行了系统性指导，明确了各章节中应掌握的内容及其深度，并通过例题讲解了习题的解题思路、方法和过程。每章内容包括两大部分：第一部分为内容要点和基本要求；第二部分为部分例题与习题选解，对程序设计的实例进行了由浅入深的分析和讲解，给出了典型的分析方法、常用算法和基本编程技巧。希望本教材对读者的学习有所帮助。本书第一、五、六、七、八、九、十章由刘晓星同志编写，第二、三、四章由潘晓萍同志编写。张立红同志也参加了部分内容的编写工作。

编　者

1993 年 2 月

目　　录

第一章 概 述

内容要点和基本要求

本章是本课程的总体概述，包括一些基础知识、基本概念和术语。其中基本概念和术语要求熟练掌握，其余内容要求一般掌握。

第一节 引 言

在这一节中，主要讲述了指令、程序和语言等基础知识，学员必须熟练掌握其中的几个基本概念。

机器指令：机器指令是能为计算机所接受的一组代码。它指出计算机所要进行的操作及其操作的对象。机器指令是由二进制代码组成的。

机器指令程序：一系列机器指令的集合称为机器指令程序。

汇编指令：用来表示机器指令的助记符称为汇编指令。每一条机器指令对应一条汇编指令。

汇编语言程序：用汇编指令编写的程序称为汇编语言程序。

汇编语言：汇编指令系统和汇编伪指令加上编程的语法规则称为汇编语言。

高级语言：高级语言是一种脱离具体计算机、面向过程、更符合人们思维和更易为人们所理解与学习的语言。

高级语言与汇编语言的比较

高级语言的优点：容易理解、便于书写、调试简单、通用性强、便于交流和移植以及具有较强的文件处理功能等。其缺点是：不能直接执行、目标程序冗长、存贮空间大、执行速度较慢、效率低。

汇编语言的优点：比机器语言容易理解、记忆和便于交流；程序精炼、执行速度快、占用存贮空间小、便于实时控制等。其缺点是：比高级语言难理解、易出错、不通用等。

第二节 汇编语言程序设计

这一节主要讲述了汇编语言程序设计的特点和方法，讲述了学习汇编语言程序设计中应当注意的七个方面以及进行汇编语言程序设计的步骤。这些内容对于初学者以后的学习、理解和使用汇编语言有着很好的指导作用。因此，要认真阅读，充分理解。

学习汇编语言应当注意的七个方面是：

1. 充分了解 CPU 和有关接口电路的结构、性能及其相关部件的功能。

2. 深刻理解和熟练掌握最常用指令的功能和作用。

3. 熟练掌握流程图的使用方法，并养成使用流程图的习惯。

4. 在条件允许的情况下，争取将大部分程序在计算机上实际通过。

5. 多编程、多上机是快速、熟练掌握汇编语言程序设计必不可少的组成部分。

6. 通过学习掌握一些较成熟的子程序。

7. 通过上机和程序设计，掌握微型计算机操作系统的功能、使用方法、使用条件以及系统的功能调用方法等。

有关程序设计的步骤，主教材中已有详尽的说明，可以大致分为以下四步：

建立数学模型→确定计算方法→编写程序→调试运行

第三节　汇 编 程 序

这一节讲述了什么是汇编程序，汇编程序的作用，汇编过程以及 IBM－PC 的汇编程序。其中，应掌握机器指令操作代码表、伪指令操作表和符号表的作用，汇编程序的作用，汇编过程，ASM 汇编与 MASM 汇编的区别以及下列几个概念：

源程序：用汇编语言编写的程序称为该语言的源程序。源程序不能在计算机上直接运行。

目标程序：源程序经过汇编程序处理后得到的机器指令代码，称为目标程序。目标程序可以直接加载运行。

汇编程序：将汇编语言源程序转换为计算机能够识别并可执行的目标程序的处理程序称为汇编程序。

机器指令操作代码表：供汇编程序查询要汇编的助记符指令的指令代码和指令长度。

伪指令操作表：给出伪指令及其处理程序地址，提供汇编程序完成伪指令定义功能的处理。

符号表：在汇编过程中建立的一种表格。其中内容为程序中使用的所有符号（或标号）及每个符号（或标号）所对应的值或相对地址，提供汇编程序在第二次扫描时查阅该表，以获取每个符号的实际值代入要转换的机器指令代码中。

汇编过程：简单地说，汇编程序一般都采用二次扫描完成对源程序的汇编任务。第一遍扫描主要完成符号表的建立；第二遍扫描将每个程序行转换为机器指令代码或数据，从而生成目标程序，同时还可根据需要生成列表文件和索引文件。两次扫描都以遇到 END 伪指令而结束。

第二章　8088/8086 系统结构

内容要点和基本要求

本章共有四节内容，其中第一节和第二节内容是深入理解指令功能的基础，必须熟练掌握。第三、第四节内容作为一般掌握和了解。下面将每节内容的要点分述如下。

第一节　8088 微处理器的结构

由于汇编语言是面向机器的语言，可以节约内存，提高速度，充分挖掘硬件潜力，因此在进行汇编语言程序设计时，要求程序设计员必须对所用机型的内部结构有一个比较全面的了解。

一、8088 微处理器

8088 微处理器具有 8 位数据通道，20 条地址线，直接寻址能力达 1MB(即 2^{20})。从结构上说可以分成两大部分：执行单元 EU(Exeution Unit)和总线接口单元 BIU(Bus Interface Unit)。如图 2-1 所示。

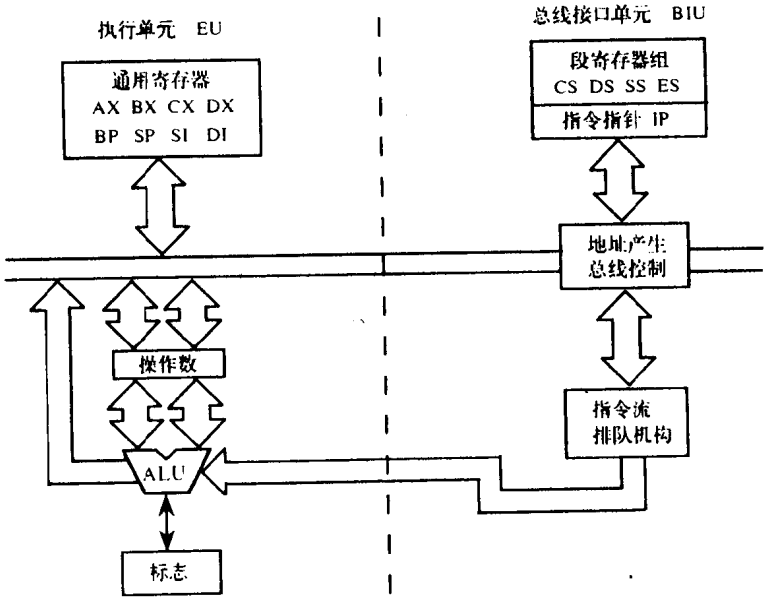

图 2-1　8088CPU 功能结构

1. EU：EU 由通用寄存器、数据暂存器，算术逻辑单元 ALU，标志寄存器 Flag 和 EU 控制单元组成。其所有寄存器和数据通路都是 16 位的，便于数据快速传送。它主要用来完成指令的执行，并进行算术逻辑运算。

2. BIU：BIU 由段寄存器，指令指针寄存器 IP，地址加法器，总线控制逻辑和指令队列组成。

它主要用来实现 EU 的所有总线操作，负责 CPU 与存贮器或输入/输出设备之间的信息交换。

二、程序执行过程

CPU 通常是重复执行下列步骤来完成一个程序的运行：

(1)从存贮器中取出待执行的指令存入指令队列中并译码；

(2)EU 从指令队列中取指令并执行，同时 BIU 从内存中取第二条(或更多条)指令至指令队列中；

(3)正执行的指令如果用到内存中的数据，则通过总线返回 BIU 读取指定的内存操作数；

(4)完成操作码的规定动作；

(5)EU 执行下一条指令，然后返回(1)处继续执行上述过程。

由于 8088 CPU 取指令和执行指令分别由 BIU 和 EU 并行完成，不占用总线时间，从而减少了 CPU 为取指令而等待的时间，使指令的执行可以不间断地进行，提高了 CPU 的利用率，加快了系统的运行速度。

第二节　8088 微处理器的寄存器

8088 共有 14 个寄存器，分为三大组。第一组为通用寄存器共 8 个，第二组为控制寄存器共 2 个，第三组为段寄存器共 4 个。如图 2-2。在编制汇编程序时会频繁地用到这些寄存器，因此对它们的掌握是非常重要的。

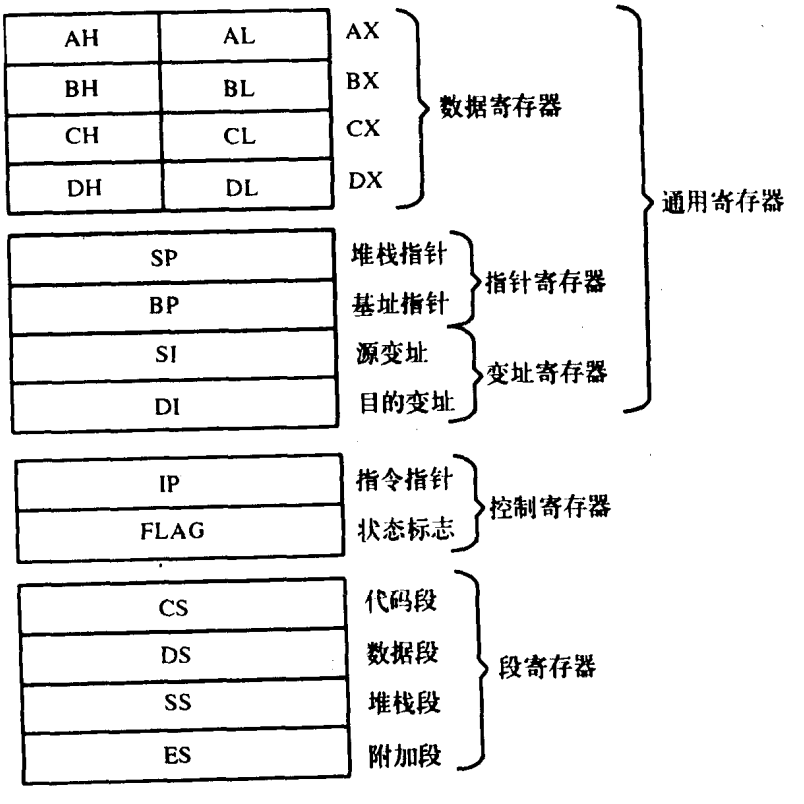

图2-2 8088寄存器结构

一、通用寄存器

通用寄存器包括数据寄存器、指针寄存器和变址寄存器。

1. 数据寄存器为：

AX(Accumulator)累加器

BX(Base Register)基址寄存器

CX(Count Register)计数寄存器

DX(Data Register)数据寄存器

AX、BX、CX、DX 均为 16 位寄存器，可用来存放 16 位的数据，也可将它们每一个按高字节和低字节分为两个 8 位的寄存器使用，分别称为 AH，AL，BH，BL，CH，CL，DH，DL 寄存器。例如 AX 的高 8 位为 AH 寄存器，AX 的低 8 位为 AL 寄存器。其中作为累加器的是 AX，AL(8 位)，其它用以存放操作数和中间结果，一般均可参与算术与逻辑运算，而且可以互换。另外，一些寄存器在一般使用时虽可互换，但作为某些特殊用途使用时则必须使用其中某些寄存器(这是规定)。例如用 BX 作为间接寻址的基址寄存器。这些在今后的学习中会逐步遇到，我们必须给予足够的重视。

2. 指针寄存器为：

BP(Base Pointer Register)基址指针寄存器

SP (Stack Pointer Register)堆栈指针寄存器

SP 和 BP 均为 16 位寄存器，都可以对堆栈中的数据进行操作。SP 始终指向当前堆栈的栈顶地址，而 BP 不具备这一功能，它是用来指向堆栈中某一数据区的基址。

堆栈是在内存中开辟的一端固定一端活动的存贮空间，固定端为堆栈的最高地址，叫栈底，活动端叫栈顶。堆栈存取必须以字为单位，一个字存入堆栈时 SP 减 2，再把字存入 SP 所指示的字单元中；从堆栈中取出一个字时，将 SP 所指示的字单元中的数据取出，然后 SP 加 2。

3. 变址寄存器为：

SI(Source Index Register)源变址寄存器

DI (Destination Index Register)目标变址寄存器

SI 和 DI 也是 16 位寄存器，它们既可作为一般通用寄存器使用，在数据串操作指令中还通常作为专用寄存器分别存放源数据串或目的数据串的基址。

二、控制寄存器

控制寄存器包括指令指针寄存器和标志寄存器。

1. 指令指针寄存器为 IP(Instruction Pointer Register)，它常用来存放待取指令的 16 位地址偏移量。

2. 标志寄存器为 FLAG(Flag Register)，用来反映系统的状态及运算结果的特点。它也是 16 位寄存器，共有 9 个标志位，其结构如图 2-3 所示。

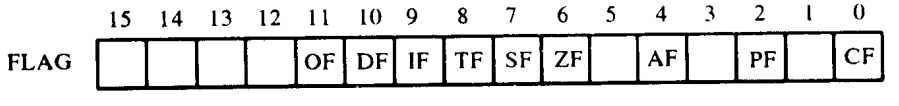

图2－3　8088标志寄存器

各标志位分别称作：

AF(Auxiliary Carry Flag)辅助进位标志

CF (Carry Flag)进位标志

DF (Direction Flag)方向标志

IF (Interrupt-enable Flag)中断允许标志

OF (Overflow Flag)溢出标志

PF (Parity Flag)奇偶标志

SF (Sign Flag)符号标志

TF (Trap Flag)追踪标志

ZF (Zero Flag)零标志

其中，DF、IF、TF 称作控制标志，其余 6 个为状态标志。各标志位的功能及作用见表 2-1。

表 2-1 FLAG 各标志的功能与作用

标志位	功　　能	作　　用
CF	当结果最高位产生一个进位或借位时，CF 置 1，否则置 0	多字节数的加减运算，移位和循环指令
PF	当操作结果中 1 的个数为偶数时，PF 置 1 否则置 0	检查数据传送过程中是否发生错误
AF	对字节操作若由低半字节(一个字节的低 4 位)向高半字节有进位或借位；对字操作若低字节向高字节有进位或借位，则 AF＝1，否则 AF＝0	多用于十进制运算的调整运算
ZF	当操作结果为 0 时，ZF 置 1，否则置 0	用于分支或循环程序的转移控制中
SF	当操作结果的最高位为 1 时，SF 置 1，否则置 0	表示带符号数结果的正负
TF	置 TF＝1 使处理进入单步方式，置 TF＝0 则 CPU 正常执行程序	便于程序调试
IF	当 IF＝1 时允许 CPU 接收外部可屏蔽中断请求，当 IF＝0 时屏蔽上述中断请求	对外部可屏蔽中断进行管理
DF	DF＝1 时串操作指令为自动减量指令，DF＝0 时串操作指令为自动增量指令	控制串操作指令增量或减量操作
OF	带符号数运算结果超出 8 位或 16 位带符号数表达范围时 OF＝1，否则 OF＝0	判断带符号数运算结果是否溢出

应注意，OF 与 CF 是有区别的。例如在字节运算时：

```
        MOV     AL,64H
        ADD     AL,64H
          01100100
即      +01100100
        ---------
          11001000
```

D7 位向前无进位，故运算后 CF＝0；但运算结果超过了＋127，此时溢出标志 OF＝1。

三、段寄存器

8088 的四个段寄存器是：

CS(Code Segment Register)代码段寄存器

DS(Data Segment Register)数据段寄存器

SS(Stack Segment Register)堆栈段寄存器

ES(Extra Segment Register)附加段寄存器

1. 段寄存器的产生原因

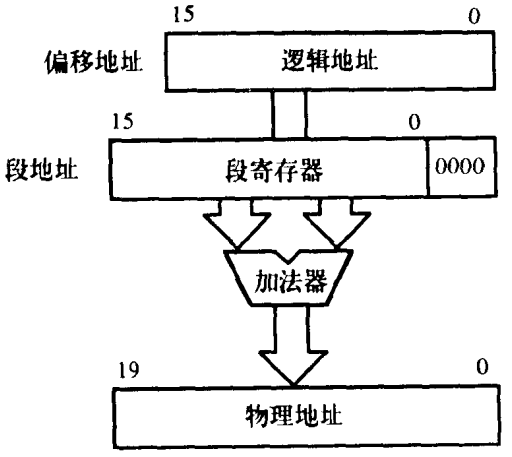

图2-4 8088中物理地址的形成

8088 的直接寻址能力达 1MB，即 2 的 20 次方。因此应有从 00000H 到 FFFFFH 的任一个 20 位地址方可从 1MB 的存贮器中取出所需指令和操作数，但 8088 CPU 内部的 ALU 只能进行 16 位运算，且与地址有关的寄存器如 SP、IP、BP、SI、DI 等也都是 16 位的，因此寻址范围只有 $2^{16}=64KB$。为使寻址达 1MB，CPU 将 1MB 存贮器以 64KB 为范围分成若干逻辑段，寻址时由段寄存器的内容给出段的基址(基本地址)，然后与 SP 或 IP 或 BP 或 SI 或 DI 等相对于段基址的 16 位偏移量(位移量)结合形成实际的 20 位物理地址。

2. 物理地址的形成

BIU 根据指令要求会自动选择相应的段寄存器，并自动左移 4 位，与一个 16 位偏移量相加，形成所需要的 20 位物理地址。如图 2-4 所示。

至于存贮器存取数据应自动选择什么段寄存器做段基，以一个来自何处的 16 位地址做偏移量，系统中有一个基本约定。只要指令中不特别说明要超越这个约定，则一般情况下就要按这个基本约定来执行，此即所谓 Default(默认)状态。如表 2-2。

表 2-2 8088 寻址约定和允许超越情况

存贮器存取方式	自动选择的段基	允许超越的段基	偏移量(相对位移)
取指令	CS	无	IP
堆栈指令	SS	无	SP
源数据串	DS	CS ES SS	SI
目的数据串	ES	无	DI
用 BP 作基址寄存器	SS	CS ES DS	有效地址 EA
通用数据读写	DS	CS ES SS	有效地址 EA

其中，有效地址 EA(Effective Address)为相对于段首的位移，即偏移量。

第三节 8088 微处理器的端脚功能

由于输入/输出及中断程序设计中常与系统硬件发生联系，所以对系统 CPU 各端脚及其功能应有所了解。

8088 为 40 个端脚双列直插式封装的 CPU 芯片，其工作模式由端脚 MN/$\overline{MX}$控制，有最大工作模式和最小工作模式两种；不同工作模式下的脚端功能有所不同。各端脚功能详见主教材。

第四节　8086 微处理器与 8088 微处理器的区别

8088 与 8086 均为第三代微处理器，8086 是 16 位 CPU，8088 称为准 16 位（因其内部虽是 16 位的，但对外数据总线是 8 位的，是 8086 的一个变形），二者内部结构相似，指令系统和汇编语言相同，在软件上完全兼容。二者在硬件的开发和使用上有一些差别，这是因为：

一、结构区别

8086 指令队列寄存器为 6 个字节长，而 8088 指令队列寄存器为 4 个字节长。

二、端脚区别

8086 对外有 16 位数据线，8088 仅有 8 位数据线；8086 中的 $M/\overline{IO}$ 线在 8088 中定义为 $IO/\overline{M}$ 线，8086 中的 BHE 线在 8088 中为 SS_0 线。

第三章　8088/8086 指令系统

内容要点和基本要求

本章主要讲授了 8088/8086 的寻址方式和指令系统。各指令的功能、作用、使用条件、置位情况及指令的寻址方式是要求熟练掌握的内容。刚开始学习寻址方式和指令系统时不可能迅速消化，需要反复思考与练习才能逐步理解掌握。有些内容暂时用不到时可以先不记，待到需要时再返回复习不迟。

第一节　寻 址 方 式

一般地，输入计算机的数据将存于 CPU 内部寄存器、外部存贮器或 I/O(输入/输出)接口三处当中之一处，由 CPU 对这些数据(操作数)进行处理。为此，我们首先需要知道这些操作数是什么类型的，以及这些操作数各自在什么地方，即其地址如何。所谓寻址方式，就是寻找指令中操作数所在地址的方法。基本寻址方式各书归纳的种类略有不同，主教材共分为 9 种。各种寻址方式及其特点都必须熟练掌握，以有助于对指令的深入理解，为今后编写程序打好基础。

一、操作数的类型

操作数有三种类型：立即数操作数，寄存器操作数和存贮器(内存)操作数。

1. 立即数操作数：作为指令代码的一部分出现在双操作数指令中的操作数叫立即数操作数。它通常可作为源操作数(乘除法及字符串操作指令除外)，但不能作为目的操作数。

2. 寄存器操作数：存放在寄存器中的操作数叫寄存器操作数。通用寄存器在单操作数指令中作目的操作数，在双操作数指令中既可作源操作数，也可作目的操作数；段寄存器不能用一般传送指令将立即数送入，若需将立即数置入 DS 或 SS 或 ES，应首先将立即数送 AX 或其它通用寄存器，然后再由 AX 传送至 DS、SS 或 ES(CS 因与指令地址有关，一般不需用户干预)；标志寄存器一般不能作为操作数。另外，有些指令中虽未明确写出寄存器名，但却隐含着使用约定的通用寄存器作操作数。见表 3-1。

表 3-1　寄存器作为隐含操作数

指　令	隐含使用的寄存器
AAA,AAD,AAM,AAS	AL,AH
CBW,CWD	AL,AH,或 AX,DX
DAA,DAS	AL
IN,OUT	AL 或 AX
MUL,IMUL,DIV,IDIV	AL,AX 或 AX,DX
LAHF,SAHF	AH
LES	ES
LDS	DS
循环或移位指令	CL
字符串操作指令	CX,SI,DI
XLAT	AL,BX

3. 存贮器操作数：存放在存贮器中的操作数叫存贮器操作数。必须明确，存贮器操作数所在的存贮器地址是存贮器的物理地址。在汇编指令中，存贮器操作数所在段的段寄存器名一般不写，其约定隐含的原则如表 2-2，即我们曾提到过的所谓 Default(默认)状态。

二、寻址方式

1. 立即寻址(Immediate Addressing)

这种寻址方式所提供的操作数直接包含在指令中。它紧跟在操作码后面，与操作码一起放在代码段区域中，其中立即数 im 可以是 8 位的，也可以是 16 位的；若是 16 位数则低字节数 imL 在低地址字节，高字节数 imH 在高地址字节。例如：

```
MOV   AX,   0FFH    ;字立即数
MOV   AL,   53H     ;字节立即数
MOV   AL,   53      ;(十进制)字节立即数
```

立即寻址示意如图 3-1。

立即寻址主要用来处理常数，给存贮器或寄存器赋初值。立即寻址在取指令的同时也取出数据，故指令执行速度较快。但由于数据是固定的，因此使用起来显得不够灵活。

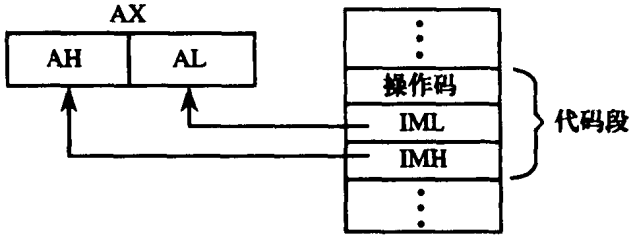

图3－1　立即寻址示意图

2. 直接寻址(Direct Addressing)

这种寻址方式为操作数地址的 16 位偏移量直接包含在指令中，与操作码一起存放在代码段区域。如不特指，操作数一般隐含在数据段区域中，即以数据段寄存器为基址，其物理地址为 DS 加上这个 16 位偏移量。其寻址示意如图 3-2。

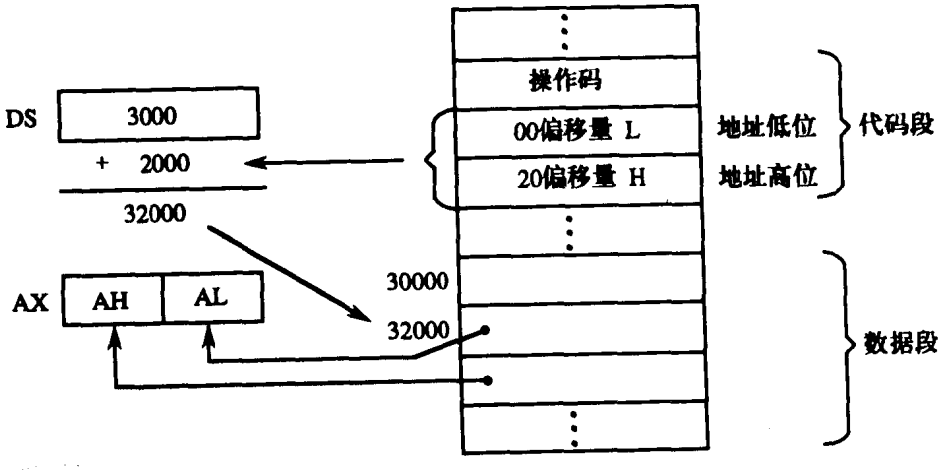

图3－2　直接寻址示意图

另外，直接寻址允许段超越，即允许段基由 SS、ES 或 CS 指出。此时操作数的地址为指令

指明的段前缀 SS 或 ES 或 CS 与 16 位偏移量之和。

例如　　MOV　AX,　ES:[22A0H]

其中 ES 即为段前缀。

直接寻址适用于处理单个变量。因其取出指令后只需访问一次存贮器便可取出操作数，所以执行速度还是比较快的。如要处理某个存放在存贮器里的变量，可用直接寻址方式将该变量先送至某一寄存器，然后再作进一步处理。为使指令字不要过长以免影响执行速度，系统规定：双操作数指令必须有一个操作数使用寄存器方式，这就是为什么一个变量常常要先送到寄存器去的原因。

3. 寄存器寻址(Register Addressing)

这种寻址方式提供的操作数在任意一个通用寄存器中。其寻址示意如图 3-3。

寄存器寻址方式由于操作数就在寄存器当中，不需要访问存贮器来取得操作数，因而可以取得较高的运算速度。虽然每一通用寄存器都可参加算术与逻辑运算，但使用累加器 AX 存放结果时指令码更短，执行速度会更快些。

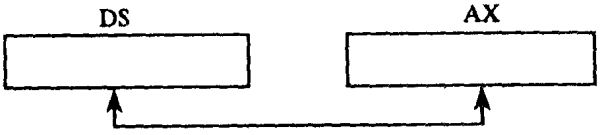

图3-3　寄存器寻址示意图

4. 寄存器间接寻址(Register Indirect Addrressing)

这种寻址方式提供的操作数在存贮器中，但操作数地址的 16 位偏移量却在 SI、DI、BX 或 BP 之一中。8088/8086 系统规定：若以 SI、DI 或 BX 间接寻址，则操作数在数据段区域中，即隐含段寄存器为 DS，操作数地址由 DS 加上 SI 或 DI 或 BX 中的 16 位偏移量形成；若以 BP 间接寻址，则操作数在堆栈段寄存器中，即隐含段寄存器为 SS，操作数地址由 SS 与 BP 相加而形成(若指令中指明段超越，则基址为所指明的段前缀寄存器)。

例如　MOV　AX,[SI]

　　MOV　AX,[BP]

寻址示意如图 3-4(1)和图 3-4(2)所示。

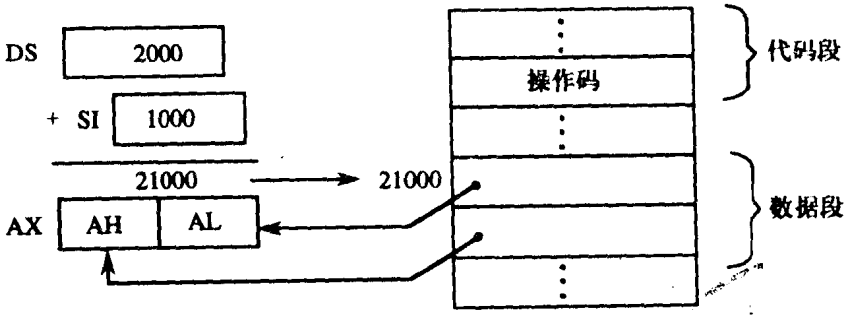

图3-4(1)　寄存器间接寻址示意图

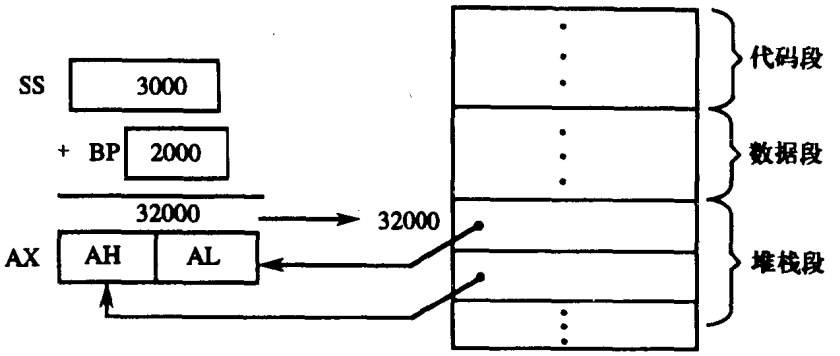

图3－4(2) 以BP间接寻址示意图

寄存器间接寻址方式可以用于表格处理，适应大批量数据传送的需要。执行完一条指令后，只需修改寄存器内容便可取出下一个数据。

5. 变址寻址(Register Index Addressing)

这种寻址方式提供的操作数地址由三项构成：一个基址或变址寄存器的内容与指令中给出的 8 位或 16 位直接位移量(displacement)之和(即操作数的有效地址 EA)，再加上隐含使用的段寄存器值。与寄存器间接寻址相似，系统规定若以 SI、DI 或 BX 作变址，则隐含段寄存器为 DS；若以 BP 作变址，则隐含段寄存器为 SS(允许使用段超越操作，以别的段寄存器作基址)。

例如　　MOV　AX,count[SI]

也可表示成　MOV　AX,[count+SI]

其中 count 为 16 位位移量的符号地址。其寻址过程如图 3-5 所示。

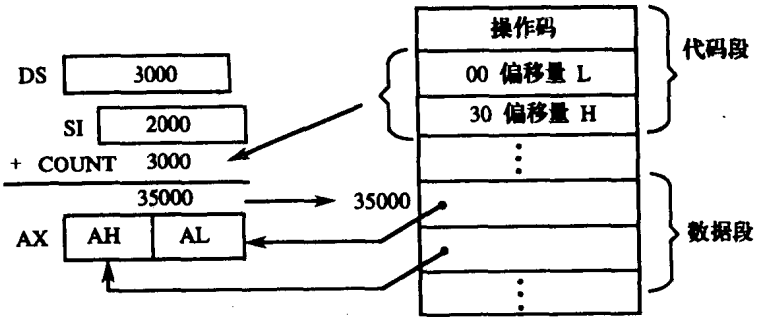

图3－5 变址寻址示意图

变址寻址方式的执行速度与上几种相比要稍慢些，因为它必须先执行一次加法以后才能得到操作数地址。但它的寻址方式很灵活，也同样可用于表格处理等。表格的首地址可设为 count，利用修改基址或变址寄存器的内容来取得表格中的值。

6. 基址加变址寻址(Based Indexed Addressing)

这种寻址方式提供的操作数有效地址 EA 由基址寄存器(BX 或 BP)加上变址寄存器(SI 或 DI)，再加上指令中给出的 8 位或 16 位直接位移量形成。其物理地址为 EA 加上隐含的段寄存器内容。隐含规定为：用 BX 作基址则隐含使用的段寄存器为 DS；用 BP 作基址则隐含使