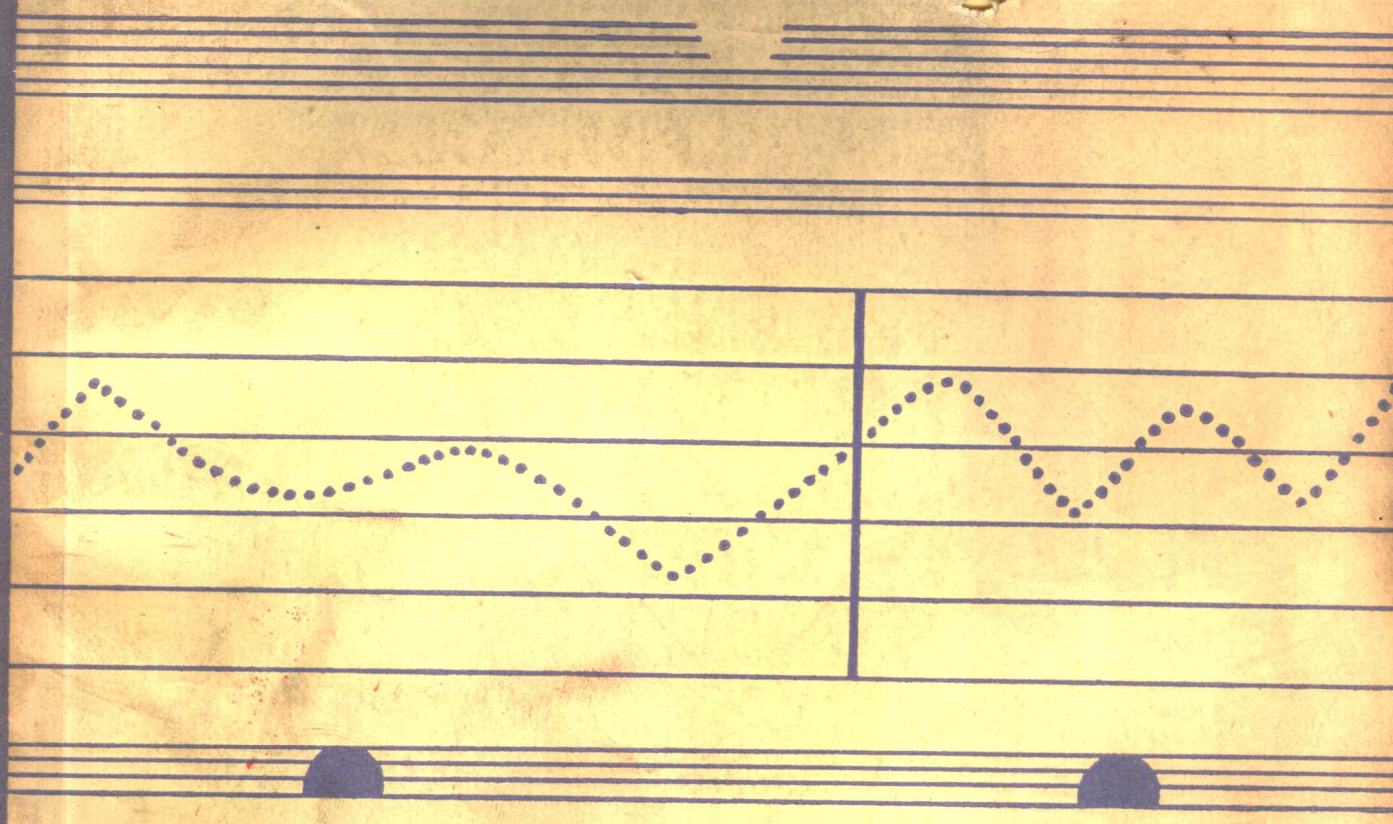
郑桂林
汪合生
编著

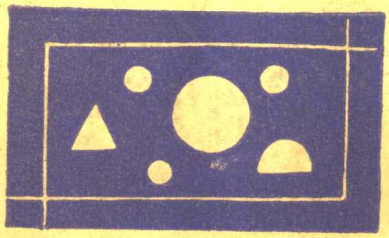
·电子天地·

微型计算机程序设计



電子書刊出版發行中心

封面设计：朱祥　开琪



微型计算机程序设计

著　　者　郑桂林　汪合生
编　　辑　王有春　张　政
责任编辑　廖汇芳　张　政
出　　版　《电 子 天 府》
印　　刷　成都科技大学印刷厂
发　　行　电子书刊出版发行中心
四川省期刊登记证291号
地　　址　成都市草堂寺南侧
《电子报》编辑部
定价：3.50元

前　言

近年来，我国的微机应用发展很快。微型计算机以其价廉物美，小巧玲珑开始渗入到我国的社会主义建设的各个领域。为了适应第三次浪潮，第四次工业革命的时势，我国的社会主义现代化事业要求更快地发展微型计算机，更为广泛地应用微型计算机。为了适应这种形势的需要，各大专院校、各生产科研单位和部门正在大力培训微机应用的软件人员。本书是为大专院校计算机技术及应用专业本科班编写的微型计算机程序设计教材。本书也可作为培训微机软件人员的软件学习班的教材，它的基本部分（一至六章）还可作为微机普及班的教材。全书授课学时为80学时左右。

本书是着重于汇编语言程序设计的方法、理论和应用的角度来分析问题的。框图法贯穿于全书之中。本书收集、整理、提练了大量有关微型机的资料和论著，根据作者多年从事微机研究和微机软件开发，多次在成都科技大学教师班、本科班及各种类型的软件训练班讲述微型计算机程序设计及应用的手稿、体会而写成。

承蒙多年从事微机研究和应用的成都科技大学计算机应用教研室龚荣武、成都科技大学计算中心李宏信同志为本书的编写提出了许多宝贵意见；成都科技大学计算中心张政、孙刚春同志为本书的编著提出了许多建设性的意见，并给予了大力支持。在此，表示衷心的感谢。

本书在成都科技大学计算机系计算机应用教研室副主任郑桂林同志的指导下，由郑桂林、汪合生同志编写。其中郑桂林编写第四、五、七、八章，汪合生编写第一、二、三、六、九、十、十一章。

由于时间仓促，编者水平有限，书中一定存在着不少缺点和错误，敬请读者批评指正。

成都科技大学　郑桂林
汪合生

于一九八四年六月

内 容 简 介

本书以Z80指令系统为背景对汇编语言的程序设计作了系统、全面、由浅入深的介绍。全书共分十一章。第一、二章介绍∠80的指令系统和基本伪指令，以及有关程序的基本概念和简单程序设计。第三章介绍分支程序设计。第四章介绍循环程序设计。第五章介绍子程序设计。第六章介绍汇编语言的程序设计。第七章介绍简单的数据结构及字符串和数据表操作。第八章介绍输入／输出和中断。第九章介绍各种应用实例，它是前述各章的综合应用。第十章介绍汇编语言程序的上机操作过程和运行、调试。第十一章总结程序设计的步骤、方法和技巧。书末附有∠80指令系统表。

本书是以教材的形式编写，内容尽量做到由浅入深，循序渐进。

本书可作为微机应用方面的工程技术人员的自学教材，也可供从事微型计算机软件教学、研究及应用等方面的大专院校师生、科技人员参考。

目　录

第一章 Z80指令系统

§1 CROMCO Z80微型机的概念模型

汇编语言是与具体计算机相关的一种低级语言。在使用汇编语言来设计程序的过程中，不可避免地要与硬件系统打交道，而高级语言不依赖于计算机的结构和指令系统，这是它们两者用来进行程序设计时的区别之一。我们使用汇编语言（或机器语言）来编制程序的过程中，程序设计者所要动用的硬件称为概念模型，而不是指整个实际的硬件系统。

如图1—1所示，为我们今后讨论汇编语言程序设计时的Z80概念模型。

一、CPU寄存器

Z80 CPU 的寄存器阵列包括程序员可以用指令访问的208位读／写存贮器，所有这些寄存器均采用静态 RAM 实现。图1—1中所示208位存贮器组成18个8位寄存器：A、F、B、C、D、E、H、L、A′、F′、B′、C′、D′、E′、H′、L′，以及4个16位寄存器PC、SP、IX、IY。这些寄存器中包含一组主寄存器和一组交换寄存器。它们又可分为两组累加器 A、A′ 和两组状态标志寄存器 F、F′。两组通用寄存器，每组六个组成，它们分别是B、C、D、E、H、L 和 B′、C′、D′、E′、H′、L′。这两组通用寄存器又可以成双地合并成16位寄存器使用，它们是 BC、DE、HL和BC′、DE′、HL′。

（一）累加器 A 与状态标志器 F

CPU 中包括两个独立的8位累加器 A 和 A′。累加器 A 用来存放8位算术和逻辑运算结果，A′ 是 A 的交换（备用）寄存器，程序员只要用交换指令就能实现 AF 与 AF′ 交换，则既可用 AF′ 保存 AF，又可选用你想用的一对累加器和标志寄存器。

其中的一个 F 寄存器用来作为状态标志寄存器使用，以表征 CPU 执行指令后的有关的状态标志，供程序测试、判断、实现分支和转移。因此，F 寄存器使用的好坏关系到程序的质量。F 寄存器有八个二进制位，但只用了其中的六位作为标志位。这六位是进位标志（C），减法运算标志（N），奇偶／溢出标志（P／V），半进位标志（H），零标志（Z），符号标志（S）。其各位分配如下图1—2所示：

7	6	5	4	3	2	1	0
S	Z	未用	H	未用	P/V	N	C

图1—2 状态标志寄存器 F

各位标志的作用如下：

1. 进位器（C）

如果加法指令在累加器的最高位得到一个进位，或在减法运算中得到一个借位，则此标志置位（即 C 位为1），否则就复位（即 C 位为0），在移位操作时，对 C 位也有影响。

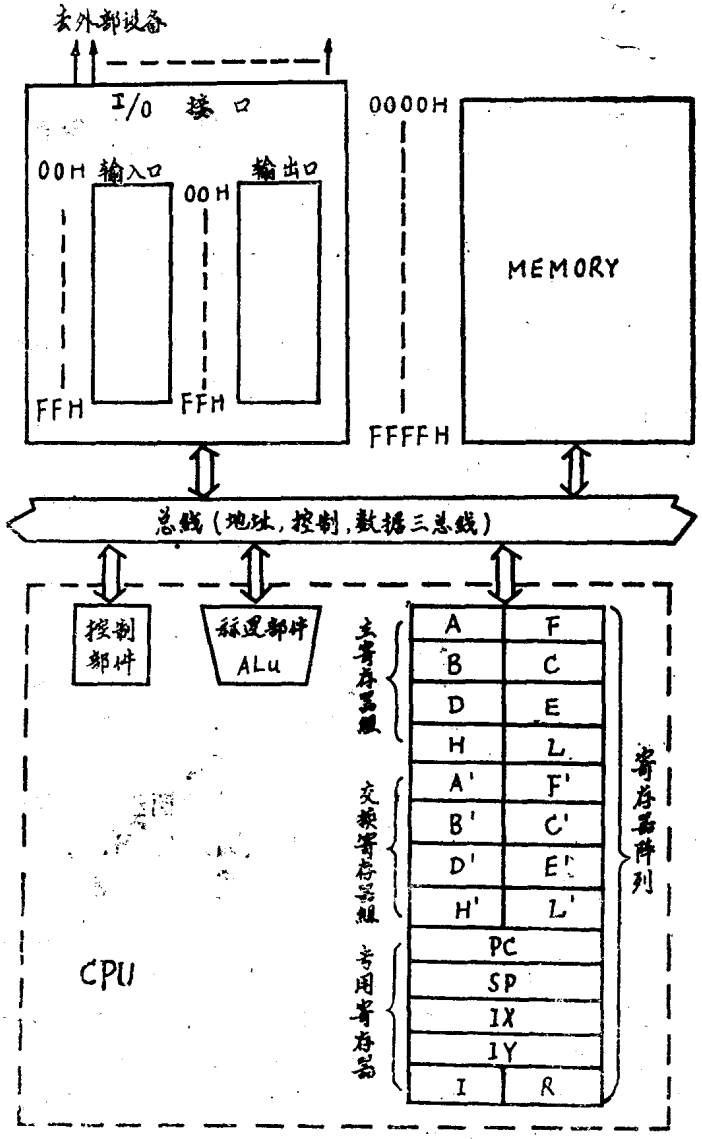


图1—1 Z80概念模型

2. 减法运算标志位（N）

如果前一次运算是减法运算，则N位置位，DAA指令就是根据N位判别前一次运算是加法运算，还是减法运算来进行十进制调整的。

3. 零标志（Z）

如果执行一条指令的结果之值为0，则此标志置位，否则复位。

5. 奇偶／溢出标志位（P／V）

该标志有二个作用：在逻辑运算时，是表示结果的奇偶性，当运算结果为偶时则置位。而在算术运算中，结果溢出时则置位，即当累加器值超过了能够表示2的补码数的上限（+127）或者小于下限（-128）时就置位。

6. 半进位（H）

这是从低四位操作而产生的进位或借位。当执行 DAA 指令（累加器的十进制调整指令）时，此标志用来供 CPU 校正先前所进行十进制加、减运算的结果。

（二）通用寄存器

Z80 CPU 中有两套配好的通用寄存器组，每组有六个8位寄存器，即B、C、D、E、H、L与 B′、C′、D′、E′、H′、L′、。你可以单独地使用 8 位寄存器 B、C、D、E、H、L，也可把它们成对地当作16位的寄存器使用，即配成 BC、DE、HL、BC′、DE′、HL′使用。在任何时候，程序员通过简单的整组交换指令，就能够选择其中的一套寄存器工作。在要求快速中断的计算机系统中，保留一套通用寄存器组及累加器／标志寄存器用来处理很快的例行子程序。例行子程序之间的运行只要执行简单的交换指令即可。这样处在中断或子程序处理时，就不存在由于存取外部堆栈中的内容所要求的时间，因而可以大大缩短中断服务时间。程序员对这些通用寄存器有广泛的使用范围。它们还可以简化程序设计，特别是用在外部读／写存贮器容易极小的 ROM 系统中。

（三）专用寄存器

1. 16位的程序计数器 PC：

PC 用来记录程序运行的运态轨迹，即作为程序运行时，指示要取指令的地址。

PC 用来保存正从存贮器取出的现行指令的16位地址，当地址码发送到地址线 A_{15}～A_0 上以后，PC 的内容自动加“1”。当程序出现分支时，新的地址码由转移指令置入 PC，此时 PC 不自动增量。再有当调用子程序时，在转子指令的控制下将原来 PC 中的内容（主程序的断点）保护进栈之后，再打入子程序的入口地地。当子程序执行到返回指令时，该指令将使保存在堆栈中的主程序断点退回 PC。

2. 16位堆栈指示器 SP：

SP 用来保持现行栈顶的16位地址，该地址可处在外部 RAM 中任何位置上。外部堆栈的工作方式是后进先出的（LIFO）。能通过执行“进栈”PUSH 和“退栈”POP 指令使 CPU“寄存器对”及 IX、IY的内容进栈和退栈。堆栈可以简化多级中断，使子程序嵌套不受限制而且可直接间接的递归调用，由于 Z—80有丰富的栈指令，所以其栈是广义的栈。它简化了各种数据的处理过程。并使程序容易实现。

3. 两个16位的变址寄存器 IX 与 IY：

两个独立的变址寄存器保存了变址寻址方式中用的16位基地址。参看图1—3及图1—4。

±d 带符号的8bit的位移量。范围是＋127～－128

M是16位的基地址

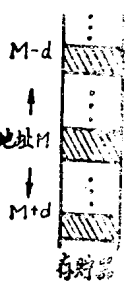
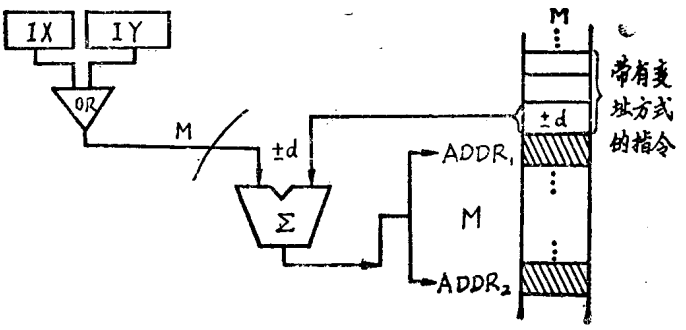


图1—3 变址范围



$ADDR_2 = M + d$　　$ADDR_1 = M - d$

图1—4 Z—80变址的形成

在这种寻址方式中，IX、IY 中存放基准地址，变址指令中包含了一个字节的带符号数的2的补码形式表示的位移量 d，经变址寻址而得出存／取数据的存贮区指针，因而大为简化了程序设计，尤其是对数据表格上的操作特别有效。

4． 8位中断页面地址寄存器I：

Z—80 CPU 在响应屏蔽中断方式2时，能够完成对任何存贮器位置的间接访问，为此目的，I寄存器存放间接地址的高8位，外设的中断装置提供间接地址的低8位。这个特点允许把中断子程序动态地安置在存贮器的任何位置上且对子程序的访问时间是极小的。如图1—5所示。

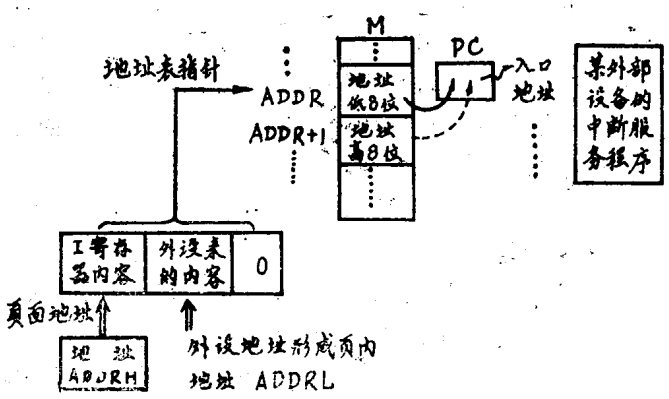


图1—5 I 寄存器动能示意

5． 8位动态 RAM 再生寄存器 R：

用于 RAM 的动态刷新（再生），一般为特殊目的才使用R或改变其内容。

二、算术与逻辑运算单元 ALU：

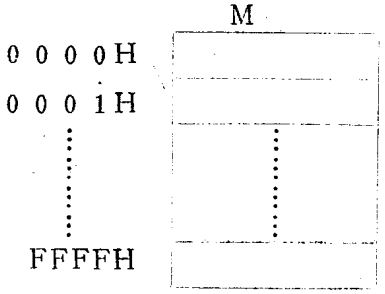
一切算术和逻辑运算在 CPU 中的 ALU 中进行。Z80内部的 ALU 是通过内部数据总线与寄存器和外部数据总线相联系的。

三、内存：

我们以64K存贮器容量为背景，讨论程序设计。对内存来说，是我们在程序设计中要常打交道的部分。对程序设计来说，关心内存的两个问题：一是内存的编址，二是字长。

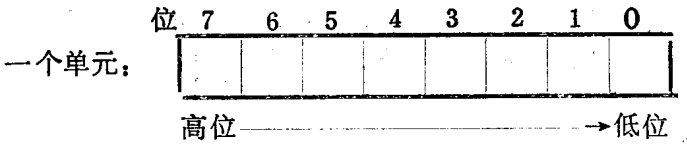
1． 内存编址：

内存地址用16位编址，我们用十六进制表示为0000H—FFFFH，如下图所示：



2． 字长：

Z80是一种八位机，内存每个单元存放一个8位二进制位的信息，即为一个字节。每个单元中，一个字节的各位编码如下所示：



四：I／O部分

Z80配置有0到255共256个I／O口地址，口地址可由编程时选择。

Z80有组成系统的各种接口电路，如：PIO 并行的输入输出接口，SIO 串行输入输出接口，CTC 接口，DMA 直接存贮器访问接口等。

§2 Z80 寻址方式

一、寻址问题的提出：

微处理器的指令基本格式是由其功能和结构所决定的。

每一条指令都由两部分组成（一些特殊专用指令除外），即分为操作码和地址码两部份组成。

操作码指明该条指令应执行的特定操作，例如传送数据、数据的逻辑操作、算术运算、转移等。

地址码应指明操作数的地址、结果数的地址以及下一条指令的地址。

具有上述功能的指令格式如下图所示：

操作码	第一操作数地址	第二操作数地址	结果的地址	下一条指令的地址

从图可见，地址码段分为四段，每一段指示一个地址。指令清晰。但是，为能访问到内存的每一个单元，每段地址应占有的位数（bit）为 $N=\log_2 M$，其中 M 为内存的容量。例如，若 $M=1024$，则 $N=\log_2 1024=10$。故一条指令的地址码总长度占40bit。随着存贮容量的增大，其他址码便增长，再加上操作码段的长度，，使得一条指令远远超过 8bit 或16bit。而一般微机的处理器是8bit 和16bit 字长的居多。为了解决这个矛盾，可以采取：

1． 把指令中指出下一条指令的地址的功能交给地址计数器去完成。

2． 在 CPU 中增设累加器，把一个操作数始终放在累加器中，运算所得的结果也放于累加器中。

这样处理之后，当然有使指令的地址长度减少。上述四地址指令格式可变成如下的单地址指令：

操作码	操作数地址

如果这样处理之后，指令编码长度仍超过微处理机字长时，可以采用指令分段的方法来

处理。例如，某个微处理器的字长为8bit，即一个字节长，则一条指令长为24bi的话，可以将其分为三段，每段长为8bit。从这种意义上讲，微处理器的指令格式又分为单字节长、双字节长和三字节长。

地址码主要指明操作数地址和结果数地址等，因此，为了缩短地址码长度，也可以在寻找操作数地址的方式上想办法。目前，在微处理器中常采用的隐含寻址和间接寻址等都可缩短地址的长度。

二、Z80的寻址方式：

存贮器用来存贮数据和指令。数据和指令在存贮器中存放的位置，称为地址。存放指令的地址，称为指令地址；存放数据的地址，称为操作数地址。

指令地址是顺序安排的，也是顺序执行的。如果若改变指令执行顺序，可以用转移指令，转移之后仍然是顺序执行的。换句话说，指令地址是在程序的执行过程中自动形成的，而且是有规律的。

数据在存贮器中一般也是按一定顺序存放的，但是在运算过程中（即程序执行过程中），有些数据可能要反复使用，无一定规律可循，这就产生了操作数地址的形成方式问题。我们把形成操作数地址的方式，称为寻址方式。形成操作数地址的过程，称为寻址过程。

目前，微处理器有多种寻址方式，Z80有下列十种：

1. 寄存器寻址

由指令操作码中的几位来指定CPU中的某一个寄存器参加运算的一种寻址方式。

Z80中八位寄存器 r 的编号为：

000：B
001：C
010：D
011：E
100：H
101：L
111：A

例如：1NC A 指令，是把累加器 A 的内容加1后再送回到累加器 A。这是一个一字节指令，由指令中的 b5b4b3 这三位指定了累加器 A。功能如图1—6。

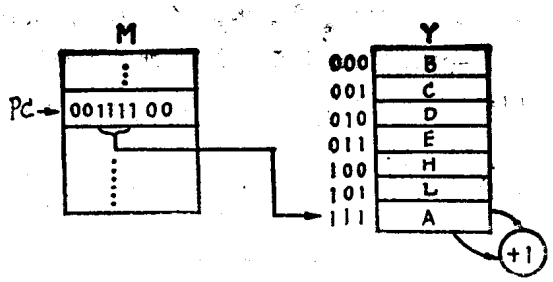


图1—6 寄存器寻址示意图

2. 隐含寻址

在这种寻址方式中，指令操作码中自动隐含一个或多个保存操作数的 CPU 寄存器。在算逻运算中，常常隐含累加器A作另一操作数。

例如，SUB B，指令格式为10010000。寻址功能见图1—7所示。

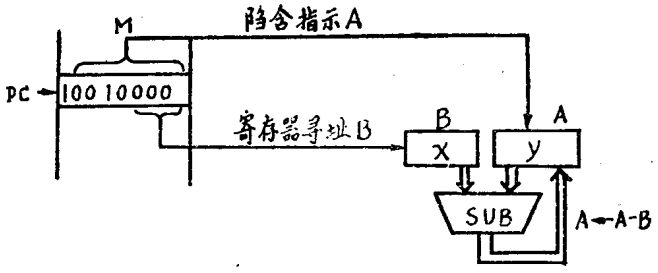
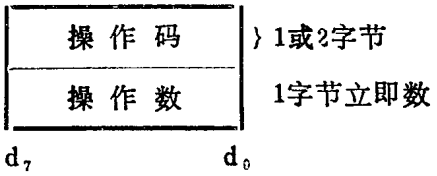


图1—7　隐含寻址示意图

3．立即寻址

在这种寻址方式中，存贮器内操作码后面的字节就是实际的操作数。



例如，AND 7。这条指令的功能是把7这个立即数与A中的内容相与，结果留在A中。寻址功能如图1—8所示。

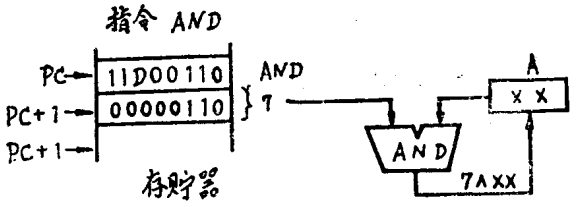
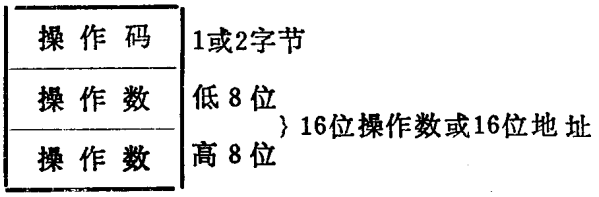


图1—8　立即寻址示意图

4．立即扩展寻址

这种方式仅仅是立即寻址的扩展，跟在操作码后面的两个字节是操作数。



这种型式指令的例子是16位（2字节）数据送到“寄存器对”（16位寄存器）中去。见图1—9。

指令，LD BC，200H

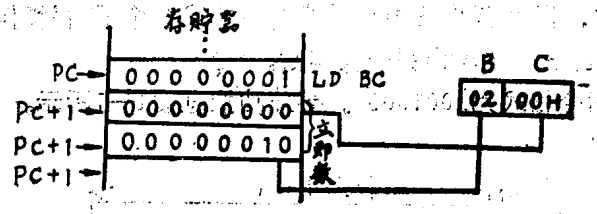


图1—9 立即扩展寻址示意图

5. 扩展寻址

扩展寻址在指令中的地址配备了两个字节（16位）。此两个字节的地址可以是程序跳转的地址，或者是存放一个操作数的地址。扩展寻址亦即大型机中的直接寻址。它能使得程序在存贮器的任何位置间跳转或者在任何位置上存取数据。

当扩展寻址用来说明操作数的源地址或目的地址时，标记(nn)用来表示存贮器中 nn 位置中的内容，这里 nn 表示指令中16位地址码。这就表示两个字节的地址码 nn 用来作为存贮器位置的指示器。采用括弧通常表示在括弧中的数值作为存贮器位置的指针。

指令形式为：

操 作 码	1字节或2字节
低 8 位地址	} 操作数的地址
高 8 位地址	

例如，LD A，（2000H）。是把2000H 单元中的内容送至累加器 A。设2000H单元中的数为EFH，其功能如图1—10所示。

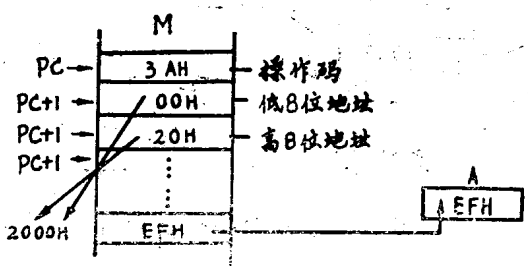


图1—10 扩展寻址示意图

6. 寄存器间接寻址

这种寻址形式指定某一16位 CPU “寄存器对” 如 HL，用作任何存贮器位置的指针。这种指令型式很有功效，且用得很广泛，实现访问存贮器也很简单。在 Z—80中数据块传送和查找命令是这种寻址型式的扩充，即附了寄存器自动加 1、减 1 和比较的功能。为了表示寄存器间接寻址，在作为指示器的寄存器名字外加括弧。

例如：（HL）。

微型机8080A、Z80中用的是“寄存器对”间址，与大型机的存贮单元间地不同，因为微型机字长短形成16位地址不易，所以用“寄存器对”间址。

例：ADD A，（HL）。以 HL 中的内容为地址，从该地址单元中取出的内容X才是

操作数，与累加器A中的内容y相加，结果放在A中。如图1—11所示。

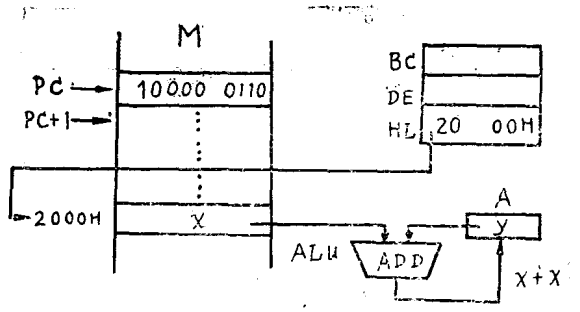


图1—11寄存器间接寻址示意图

7. 零页寻址

这是一个一字节指令，它指定零页（0000H—00FFH为零页）中的八个特定单元为子序的入口地址。Z80有一种专门的单字节的重新启动指令 RST n，它可以转到零页上的八个特定地址之一启动常用的子程序或中断子程序。其指令形式为：

7	6	5	4	3	2	1	0
1	1	b5	b4	b3	1	1	1

b5b4b3形成0—7之间的整数，根据其形成的值，指令调用规定入口处的子程序。

执行这条指令时，把 PC 的内容压入堆栈保护起来，然后把二进制代码 0000000000b5 b4 b3000传送于 PC。利用这条指令可以实现程序转移至下述8个地址之一：

0 0 0 0H
0 0 0 8H
0 0 1 0H
0 0 1 8H
0 0 2 0H
0 0 2 8H
0 0 3 0H
0 0 3 8H

b5b4b3的值与规定的入口地址调用如表1—1所示。

表1—1　RST n 入口地址表

b5 b4 b3	调用地址	指令	符号指令
0 0 0	0000H	C7 H	RST 0
0 0 1	0008H	CF H	RST 8
0 1 0	0010H	D7 H	RST 10H 或 RST 16
0 1 1	0018H	DF H	RST 18H 或 RST 24
1 0 0	0020H	E7 H	RST 20H 或 RST 32
1 0 1	0028H	EF H	RST 28H 或 RST 40
1 1 0	0030H	F7 H	RST 30H 或 RST 48
1 1 1	0038H	FF H	RST 38H 或 RST 56

零页寻址主要用于形成中断矢量，此时 RST n 指令由申请中断的中断源提供。

8. 相对寻址

这种寻址用于转移指令中。它以 PC 的内容作为基地址，向前或向后移动一定的位移量作为转移地址。

指令格式为：

A	相对转移
A+1	位移量 e
A+2	

其中位移量 e 是8位带符号数对 2 的补码（－128到＋127）。若现行的相对寻址指令的操作码地址为A，下一条指令操作码的地址为A＋2，则执行完这一条指令之后，程序转移到的地址B为：

$$B = A + 2 + \text{位移量 } e。$$

也就是说，如以相对转移指令的操作码的地址作为基地址，则总位移的范围在＋129到－126之间。若以相对转移指令的下一条指令的操作码地址 A＋2为基地址，则总的位移量范围在＋127到－128之间。由于计算相对地址的方法不同，故有 e 和 e—2之差。如图1—12所示。在汇编语言中，e 可用绝对地址或用标号来书写，可避免程序员在编程序时用手工计算位移量，而由汇编程序将其化为相对地址。

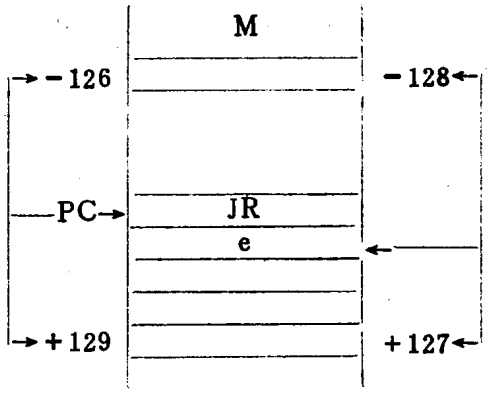


图1—12 相对寻址示意图

9. 变址寻址

在这种寻址方式中，把操作码后的一个数据字节作为位移量与变址器 IX 或 IY 相加而形成存贮器指针。

变址寻址的指令形式为：

操作码	} 二个字节操作码
操作码	
位移量 d	

位移量 d 是带符号的对 2 的补码（－128到＋127）。

因为Z80有两个变址寄存器 IX、IY，为表示变址采用记号：

（Ix＋d）或（Iy＋d）。这里，d是操作码后的位移量。所计算出的括号内的值作为存贮器的指针。

例：若 IY 中的内容为2000H，则 ADD A，（IY＋20H）是把 IY 中的内容2000H 与指令中给定的位移量20H相加而求得操作数的地址，由此地址中取出内容X与累加器A中的

内容y相加，结果送入A，如图1—13所示。

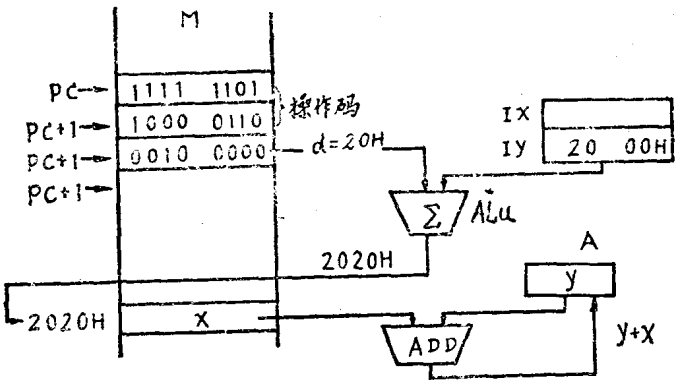


图1—13 变址寻址示意图

10。 位寻址

Z80中具有很多“位置1”，“位置0”和“位测示”指令。这些指令通过寄存器寻址，寄存器间接寻址和变址寻址，三种寻址方式之一，使任何存贮器单元或者 CPU 的任何寄存器中的某一位进行操作。并用操作码中的三位来阐明8位中的哪一位进行操作。见图1—14所示。

例：BIT b, r,　1 1 0 0 1 0 1 1

0 1 0 0 1 0 0 1

b　r

指出为bit，即第二位↑　↑指出为寄存器C

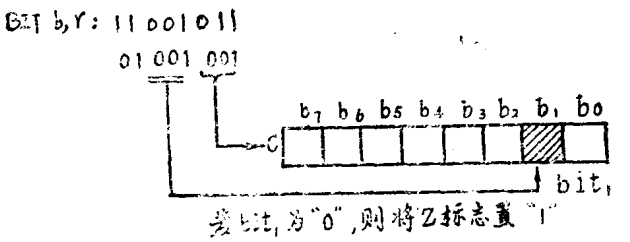


图1—14

§3 Z80指令的分类和编码格式

Z80 CPU 的指令系统功能较强，它既含有数据处理指令，又有数据传送，程序控制和状态管理，CPU 控制指令。我们从功能和性质上概括起来可以分成以下八种类型的指令。

1。 数据传送与交换指令

2。 成组传送和检索指令