

微型计算机及其在 医学中的应用

刘兆毓 编

重庆大学出版社

微型计算机及其在医学中的应用

刘兆敏 编

责任编辑 王孝梓

*

重庆大学出版社出版

新华书店重庆发行所发行

中国人民解放军第三军医大学印刷厂印刷

*

开本: 787×1092 1/16 印张: 16 字数: 399千 插页: 1

1987年10月第一版

1987年10月第一次印刷

印数: 1-2500

ISBN7-5624-0002-4 统一书号: 15408·15

标准书号:

R·1

定 价: 2.70元

内 容 简 介

本书以微型计算机原理为基础, 而着重讲述微型计算机在医学中应用的基本知识。全书分为两部分, 第一部分为微型机原理, 共分三章。第一章扼要地讲述微型机的基本结构和工作原理。这一部分的重点是第二章接口技术和第三章微型机系统, 这也是应用微型机的人所最关心的。

第二部分有五章, 重点是第四章医学信号处理技术、第六章数理诊断和第七章数学模型与计算机模拟。医学信号处理技术一章介绍了单板机在医学上的应用和特点。第六章和第七章都属于数理医学范畴。其中, 数理诊断一章介绍了常用的一些统计方法; 数学模型和计算机模拟一章则应用BASIC语言程序讲述了用计算机模拟生理过程的方法。第五章简要地介绍了医学影像处理的基本原理, 第八章在介绍微型机数据库基本知识的基础上, 讨论了医院计算机管理和医生办公室自动化等问题。

附录中列出了目前常用的8位和16位微处理器芯片的基本性能、Z80 CPU指令系统说明和一些程序清单, 供读者参考。

本书可作为工科和医科院校有关专业本科及研究生的教材, 也可作为工程技术人员和医务工作者的参考书。

前 言

微型计算机由于其体积小、重量轻、功能强和价格日趋低廉而获得越来越广泛的应用。在生物医学工程领域内也不例外，无论是医学仪器或是医院的管理，微型计算机都是一种其它设备无法取代的强有力的工具。所以，从事生物医学工程的人，必须学习、掌握微型计算机的有关知识，并把它应用到生物医学学科的科研、生产以及教学中去，使这门学科取得新的飞跃。

本教材是为生物医学电子工程专业研究生编写的（亦可作为医电专业高年级学生的选修课教材或作为从事微型计算机应用的工程技术人员和医务工作者的参考书）。全书共分八章，前三章为微型计算机基本原理，后五章为微型计算机在医学中的应用。学习本教材所需的基本知识，除了大学电子类专业的基础课和技术基础课知识以外，尚需掌握统计学原理、数字信号处理、计算机组成原理、BASIC语言以及程序设计等方面的知识。

鉴于目前讲述微型计算机原理的书籍很多，而内容基本上大同小异，故第一章“微处理器”中，仅对目前广为流行又具有代表性的8位微处理器——Z80的CPU进行归纳性的介绍，这样既可作为初学者了解微型机的基本结构和工作原理之用，又可供已学过微型机的人复习之用。

微型计算机的应用，概括来讲主要有两方面的内容：一是应用软件的设计，二是微型机与应用对象之间的连接技术——接口技术。第二章在讨论了接口技术的一般原理后，着重介绍了常用的三种接口：并行接口、串行接口、模/数和数/模转换接口。

第三章介绍了人们容易忽略但又容易混淆的微型计算机总线技术。微型机系统则介绍了常用的单板机和苹果Ⅱ型微计算机，并介绍了在数据采集系统中使用苹果Ⅱ微机的例子。

微型机在医学中应用最多的是医学信号处理，因为各种微型机控制的医疗电子仪器以及临床监护系统等，都是首先进行医学信号的采集，然后进行处理。第四章较为详细地讨论了医学信号处理的基本方法，并以心电信号的处理作为实例，说明其基本原理。这章最后还介绍了单板机在医学信号处理中的应用方法。

第五章概略地介绍了医学影象处理技术。

最后三章的内容与微型机硬件关系不大，主要是与统计学、计算方法及计算机软件的知识有密切关系的一些内容。其中第六章介绍了数理诊断和统计分类的基本方法。第七章介绍了医学中的数学模型和微型计算机模拟的方法，并给出了BASIC语言的模拟实例，这对从事生理模拟研究的人来说，有一定的参考价值。第八章是在介绍计算机数据库的基本概念的基础上，讲述了微型计算机在医院管理方面的应用。

由于微型机在医学上的应用非常广泛和灵活，而且微型机的软、硬件技术也在不断发展中，本教材不可能详尽地介绍微型机在医学上的所有应用，只能对基本原理加以阐述，并由此出发，尽可能结合一些实例进行介绍。不过读者只要掌握了一般原理，微型机应用在医学方面的具体问题是不难解决的。

由于编者水平有限，经验不足，书中难免存在缺点和不足之处，恳请读者批评指正。

编 者

目 录

第一章 微处理器

第一节 微处理器与微型计算机	(1)
第二节 8 位微处理器的指令系统—Z80 指令系统	(3)
一、Z80CPU的基本结构	(3)
二、Z80 的寻址方式	(8)
三、Z80 的指令系统	(10)
第三节 汇编语言程序	(17)
一、汇编语言程序的基本格式	(17)
二、伪指令	(18)

第二章 微型计算机的接口技术

第一节 接口的基本类型	(20)
第二节 接口的寻址方式	(21)
第三节 CPU与I/O接口的数据传送方式	(23)
一、同步传送(无条件传送)	(23)
二、异步传送(条件传送或查询方式)	(24)
三、中断传送方式	(25)
四、直接数据通道传送(DMA)	(25)
第四节 中断	(25)
一、中断的基本功能	(25)
二、Z80 的中断方式	(32)
第五节 并行接口电路	(35)
一、Z80—CTC	(35)
二、Z80—PIO	(39)
第六节 串行接口电路	(44)
一、串行通讯	(44)
二、Z80—SIO	(47)
第七节 模/数(A/D)和数/模(D/A)转换器	(53)
一、数据采集系统和A/D转换	(53)
二、采样—保持电路	(55)
三、A/D转换器	(57)
四、D/A转换器	(60)
五、A/D与D/A应用举例——数据采集和控制信号的输出系统	(62)
第八节 微型机的外设和接口	(70)

第三章 总线结构与微机系统

第一节 微型计算机的总线	(71)
一、内总线	(71)

二、IEEE 488 标准总线	(73)
三、EIA RS-232C 标准总线	(88)
第二节 微型计算机——单板机	(91)
一、TP801结构	(92)
二、TP801监控程序 (TPBUG)	(93)
第三节 微型机系统——苹果机 (Apple I)	(106)
一、Apple I 的基本结构	(107)
二、Apple I 的系统软件	(110)
三、Apple I 在数据采集系统中的应用	(116)

第四章 医学信号处理

第一节 医学信号处理概述	(118)
一、医学信号的采集	(118)
二、医学信号的处理	(118)
第二节 生理电信号——心电信号的计算机处理	(130)
一、心电信号和心电图	(130)
二、心电信号的计算机处理和分析	(132)
第三节 单板机在医学信号处理中的应用	(147)
一、医用单板机的特点	(147)
二、单板机I/O接口设计举例	(149)
三、单板机的程序设计	(154)

第五章 医学影像处理概述

第一节 基本概念	(158)
一、数字图象处理基本概念	(158)
二、医学影像的检出、记录和识别	(158)
第二节 计算机化的 X 线层面扫描系统 (CT)	(162)
一、CT 基本原理	(162)
二、重建技术	(165)

第六章 数理医药学 (一) ——数理诊断

第一节 概述	(171)
第二节 贝叶斯 (Bayes) 条件概率模型——逻辑诊断	(173)
第三节 贝叶斯公式的简化——评分法	(175)
第四节 二级判别	(176)
第五节 序贯判别	(179)
第六节 多元逐步回归分析	(181)
一、多元逐步回归分析的基本概念	(181)
二、逐步回归的计算步骤	(184)
三、逐步回归分析算法实例	(186)
四、逐步回归分析的优缺点	(188)
第七节 聚类分析	(189)

第七章 数理医药学(二)——数学模型与计算机模拟

第一节 概述	(193)
第二节 室动力学——应用流体力学模型	(194)
第三节 动脉压模型之一	(196)
一、模型	(197)
二、计算机模拟	(197)
三、BASIC 程序	(199)
第四节 动脉压模型之二	(200)
第五节 药物动力学模型	(203)

第八章 医院计算机系统

第一节 概述	(205)
第二节 有关计算机数据库的基本知识	(207)
一、文件系统和数据库	(207)
二、数据库的基本功能	(209)
三、数据结构及数据库类型	(210)
四、dBASE-Ⅰ数据库简介	(212)
第三节 病历管理系统和病历索引	(216)
第四节 医院信息管理系统的设计	(219)
一、模块化设计方法	(219)
二、医学语言Paralanguage	(221)
三、医院信息管理系统举例——PRAKTICE高级医疗信息系统	(222)

附录

附录1 常用的8位和16位微处理器芯片性能	(227)
附录2 Z80指令功能表	(229)
附录3 CLKREAD采样程序清单	(235)
附录4 两连续室应用流体力学模型模拟程序清单	(239)
附录5 动脉压模型模拟程序清单	(242)

第一章 微处理器

第一节 微处理器与微型计算机

自七十年代初，美国Intel公司研制的第一个微处理器Intel4004问世以来，短短的十几年间，微型计算机几乎席卷整个世界，并在新的技术革命中起着越来越重要的作用。虽然微型计算机的发展和应用主要是以微处理器为核心的，但微型计算机与微处理器却有不同的含意。

微处理器（Microprocessor）或称微处理机，是指由一片或几片超大规模集成电路或大规模集成电路芯片组成的中央处理部件，即CPU（Central Processing Unit）。它主要包括累加器、算术逻辑部件、寄存器、指令部件、时序和控制部件以及内部数据通路——内部数据总线。有的还包括时钟电路。

微型计算机（Microcomputer）是以微处理器为核心，加上读写存储器（RAM）、只读存储器（ROM）、输入/输出接口和联结这几部分的总线——元件级总线、内部总线和外部总线所组成。有的还应包括部分外部设备和一些软件，如监控程序等。

微型计算机系统（Microcomputer System）是在微型计算机上配以相应的软件和外部设备，构成一个功能比较完备的计算机系统。这里所说的软件是指系统软件，如操作系统、诊断程序、汇编程序、高级语言的解释或编译程序等。外部设备是指常规的一些设备，如键盘、字符/图形显示终端、打印机、磁盘驱动器、磁带机（一般是盒式磁带机）等。

图 1 - 1 为微处理器、微型计算机和微型计算机系统的基本组成示意图。

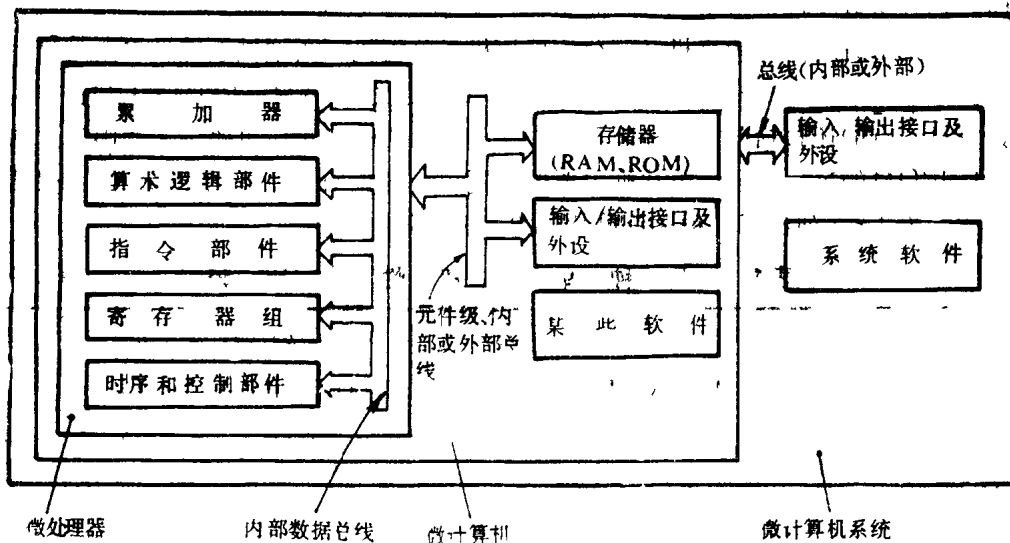


图1-1 微处理器、微型计算机及微型计算机系统

对于微型计算机系统，有多机系统和单机系统之分。前者是指用多个微型计算机或多个微处理器有机地组织起来的、能相互协调地工作的系统。后者则仅为一台微型计算机。

微处理器的分类方法很多，若按器件工艺来分，可分为MOS电路的、TTL电路的和ECL电路的；若按字长来分，可分为4位、8位、16位和32位的。另外还有位片式（bit-slice）

的。附录 1 列出了目前使用较多的微处理器芯片及其基本特性。

从采用的集成电路工艺来看，在MOS电路中主要有N沟道 NMOS，互补CMOS和高密度短沟道HMOS三种。这三种电路的技术性能差别见表 1 - 1。表中也给出了TTL电路和ECL电路的主要性能。

表1-1 微 处 理 器 工 艺 及 性 能

性 能 工 艺	集成度	功耗	速度	价格	与TTL 兼容性
NMOS	高	大	低	低	不
CMOS	较低	小	中	较贵	可
HMOS	高	小	高	贵	可
TTL	最低	大	高	贵	可
ECL	低	最大	最高	最贵	不

TTL电路主要用于位片式微处理器中。所谓“位片”，是指一个字长只有 2 位或 4 位的CPU芯片，用户可根据需要用多个这样的芯片组成一定字长的微型计算机。如用 4 片 2 位 CPU可做成 8 位 微 型 计 算 机。这类微型机多采用微程序控制，并且常用来作为计算机、通讯和信号处理方面的高速控制器。

微型计算机可按字长、结构和用途进行分类。

4 位机和 8 位机多用于控制方面，如工业过程控制、仪器仪表智能化、计算器及家用电器等。也有少数16位机用于工业控制。16位机和32位机则多用于科学计算、数据处理、商业等方面。也有少数 8 位机用于这一方面。

从结构上来看，微型计算机又可分为单片机、单板机、微型机系统及多微型机系统。

1. 单片机：是在一个芯片上做成既包括CPU，又有RAM、ROM和I/O接口的完整的微型计算机。有的还包括A/D转换器电路。这类单片机由于集成电路工艺的限制，它所包含的RAM和ROM的容量很小。单片机主要做控制用，可象一个电路元件那样放到仪器或控制设备中去。表 1 - 2 列出了几种 8 位单片机的主要技术指标。

表1-2 几 种 单 片 机 性 能 比 较

	I 8048/8748	I 8049	M6801/68701	Z`8	I 8022
ROM (字节)	1K ROM/EPROM	2K ROM	2K ROM/EPROM	2 K	2K ROM
RAM (字节)	64	128	128	144	64
存储器可扩充容量 (字节)	4K	4K	64K	124K	/
并行 I/O 接口线(条)	27	27	29	32	28 (包括 2 通道 8 位 A/D)
串行 I/O 接口	/	/	UART	UART	/
基本指令数	96条 与8080不兼容	96条 与8080不兼容	72条 与MC6800兼容	129条	8048子集
指令执行时间 (μs)	2.5~ 5	2.5~ 5	2~12	1.5~2.5	8~16

2. 单板机：是在一块印刷电路板上装有微处理器、RAM、ROM、I/O 接口及其它附属元、器件，构成一个可以执行机器语言指令的完整微型计算机。单板机的具体结构及其应用将在第三、四章内详述。

3. 微型机系统：微型机系统多以它所用的CPU和字长来分类。通常，系统中所含CPU的基本性能和结构决定了微型机系统的主要性能指标。当然，对微型机系统性能起重要作用的还有系统软件的设计和配置等因素。

4. 多微型机系统：多微型机系统是将多个微型机采用一定的方式联结起来（有的组成了计算机网络），其目的主要有：

（1）提高系统的性能。例如，分布式阵列处理机可以提高运算速度，而网络则可以实现资源共享。

（2）增加系统的灵活性。可根据需要扩充或压缩系统的规模。尤其是采用总线结构时更为方便。

（3）提高系统的可靠性。很明显，当系统中某一台微型机出现故障时，整个系统仍可运行。

第二节 8位微处理器的指令系统——

Z80指令系统

由于8位微型机既可用于控制又可用于小型数值计算和数据处理，因此本书选定8位机为主要微型机机型，重点讨论其中的典型产品——Z80CPU的基本原理。在后面的几章中也将以Z80外围接口电路为主进行讨论。

一、Z80 CPU的基本结构

Z80微处理器是美国Zilog公司在Intel 8080微处理器的基础上，经过改进设计而成的，它采用N沟道硅栅耗尽型工艺、40脚双列直插式封装，具有以下几个特点：

1. 具有与TTL电路相兼容的电路参数；

2. 使用+5V单一电源及单一时钟脉冲。它有两种时钟频率：Z80为2.5MHz，Z80A为4MHz；

3. 内部设有中断处理部件，可允许用户使用四种中断方式（非屏蔽中断及屏蔽中断方式0、1、2）；

4. 具有动态存储器刷新逻辑电路，用户在使用动态存储器时，只要给出刷新地址即可，而不必考虑刷新时序。

图1-2为Z80 CPU内部结构图。其各部分的功能如下：

1. 算术逻辑部件ALU (Arithmetic Logic Unit)：执行算术和逻辑运算、二—十进制运算、各种移位等操作。需要进行算术或逻辑运算的数须先送到累加器A（带锁存器）和暂存寄存器TMP中，然后再参加运算。运算结果通过内部总线再送回到累加器A中，同时还要根据运算结果对相应状态寄存器置位。

十进制调整累加器DAA (Decimal Adjust Accumulator)：是为进行十进制运算而设立的。

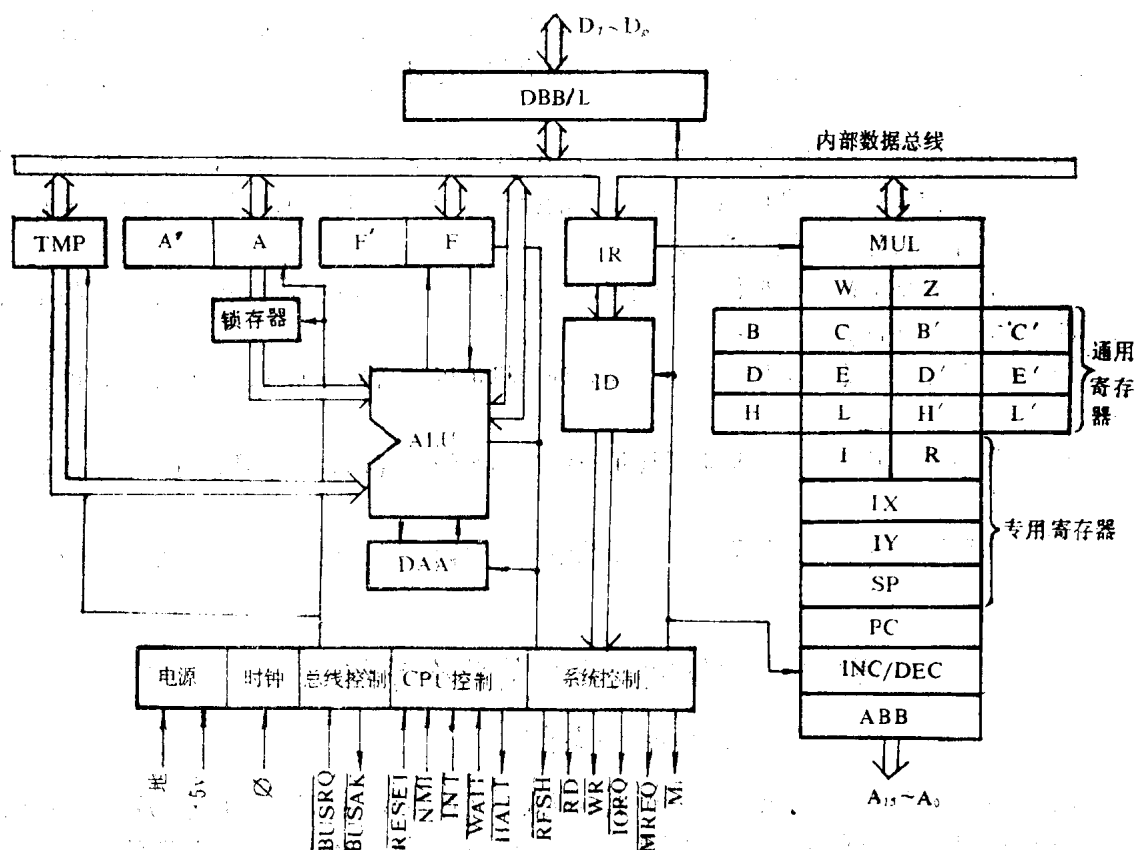


图1-2 Z80 CPU内部结构

2. 累加器A (Accumulator)：实际上是一个寄存器，它本身没有运算能力，但可以和ALU一起完成各种运算，并由它保存运算结果。累加器A'是辅助累加器。

3. 指令寄存器IR (Instruction Register) 和指令译码器ID (Instruction Decoder)：当指令从存储器中取出以后，就放到指令寄存器IR中，然后送到指令译码器中进行译码，即将指令翻译成相应的控制信号，以产生所需的动作。

4. 程序计数器PC (Program Counter)：目前的计算机都采用冯·诺依曼“存储程序”的思想，即将编好的程序，按存储器单元的编号（地址），从小到大依次存放。例如，程序中第一条指令放在存储器0000号单元中，第二条放在0001中，……。在执行程序时，也应从0000, 0001, ……单元中依次取出每条指令，将指令译成相应的控制信号，然后去执行所需的动作。而负责记录程序所在的存储器单元号码并不断增加其值的部件，就叫程序计数器。在图1-2中，PC经加1/减1寄存器INC/DEC (Increment/Decrement) 对PC的内容进行修改以后，由地址缓冲器ABB (Address Bus Buffer) 送到地址总线AB (Address Bus) 上。

5. 通用寄存器：共有两组完全相同的8位通用寄存器，一组是B、C、D、E、H、L，另一组是B'、C'、D'、E'、H'、L'，称为辅助寄存器。这些寄存器可以单独作为8位寄存器使用，也可以构成寄存器对（16位）使用，这时，BC、DE、HL以及B'C'、D'E'和H'L'各为一对。

6. 专用寄存器:

(1) 变址寄存器IX和IY (Index Register X,Y): 这两个都是16位寄存器, 用于存放变址寻址方式的变址值。

(2) 堆栈指示器或称堆栈指针 SP (stack Pointer): 堆栈是存储器中的一个专用存储区。在Z80 CPU中, SP保存当前堆栈栈顶的16位地址, 当执行堆栈操作时, 是按SP的内容去访问存储器, 访问完了以后, 要对SP的内容进行修改。堆栈操作是按“后进先出”的原则进行的。这种操作对于中断处理 (包括多级中断处理)、多重子程序嵌套等操作是非常有用的。

(3) 中断向量寄存器 I (Interrupt Vector Register) 或称中断页面地址寄存器: 用于存放中断向量的高8位地址。

(4) 刷新寄存器R (Refresh register) 或称刷新计数器: 当采用动态MOS存储器时, 必须定期 (一般为2ms) 对存储器进行刷新, 以防止信息被泄漏掉。Z80 CPU是利用取指令周期的后两个状态来进行刷新的。刷新时, 每次刷新内存的一行, 而行地址由寄存器R提供, 且在每刷新一行之后, R的内容自动加1, 指向下一行。

从图1-2可以看出, 由于通用寄存器和专用寄存器组成的寄存器阵列, 共同使用一条内部数据总线, 故每次只能选中一个寄存器, 而且要专门设置一个多路器MUL, 由它来控制寄存器的选择。在指令执行过程中, 寄存器阵列中W、Z这两个寄存器是用来暂存中间信息的, 程序员不能使用。

(5) 标志寄存器F (Flag register): 标志寄存器是用于记录指令执行后的一些特征的。这些特征反映了运算的部分结果并用来判断程序是否要转移等。这样的特征标志称为测试标志。还有一类标志主要是用于BCD运算, 叫做非测试标志。Z80的标志寄存器共有六个标志位。图1-3为标志寄存器F中各标志位的安排以及与运算结果的对应关系。

1) 进位标志Cy (Carry) 或简写为C: 在做加法运算时, 从最高位 D_7 产生进位时, 则 $C_y = 1$ 。在做减法运算时, 最高位 D_7 产生借位时, 此位也置“1”。各种移位操作也对此位有影响。

2) 零标志位Z (Zero): 当运算结果为0时, 将此标志位置“1”, 否则置“0”。

3) 符号标志S (Sign): 在做带符号的运算时, 当运算结果为负时, $S = 1$, 为正时 $S = 0$ 。

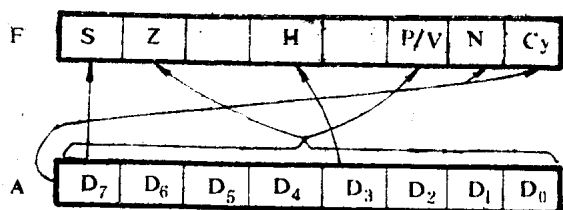
4) 减标志N (Negate): 在BCD运算时, 十进制调整指令DAA用此标志来区别加减法, 若执行DAA之前的指令是加法, 则置N为“0”, 是减法, 则置 $N = 1$ 。

5) 半进位标志H (Half carry): 当低位半字节向高位半字节产生BCD进位或借位时, 此位置“1”, 此位用于修正BCD加减法的结果。

6) 奇偶/溢出标志P/V (Parity/oVerflow): 此标志有两个用途。一是标识逻辑运算结果的奇偶性, 如果结果中有偶数个“1”, 则P/V为“1”, 否则为“0”。另一用途是标识是否发生2的补码溢出, 如果发生溢出, 则此标志位置“1”, 否则置“0”。

因为溢出是对带符号的数参加运算所产生的结果而言的, 所以在判断溢出时必须用到符号位S。

对于8位微处理器来讲, 溢出是表明运算结果超出了8位数的最大范围, 即运算结果小于负 $2^7 = -128$, 或大于 $2^7 - 1 = +127$ 。



若以 A_7 、 B_7 分别表示A、B两数的最高位，即两数的符号，用S表示和的符号，则可列出两数相加时的真值表如下：

图1-3 标志寄存器F

A_7	B_7	S	P/V	
0	0	0	0	两数同号，且与和的符号相同，无溢出。
1	1	1	0	
0	0	1	1	两数同号，且与和的符号不同，产生溢出。
1	1	0	1	
0	1	0	0	两数异号，不产生溢出（与和的符号无关）。
0	1	1	0	
1	0	0	0	
1	0	1	0	

对于减法则有：

A_7	B_7	S	P/V	
0	0	1	1	两数异号（在做减法时，变为同号相加），差的符号与被减数相反，则产生溢出。
1	1	0	1	
0	0	0	0	不产生溢出。
0	1	0	0	
0	1	1	0	
1	0	0	0	
1	1	1	0	
1	0	1	0	

综合上面四种产生溢出的情况，可以归纳出两种规则：

- 当 $A_7 = B_7 = 1$ 即 $Cy = 1$ 时， $S = 0$ 为有溢出，或
- 当 $A_7 = B_7 = 0$ 即 $Cy = 0$ 时， $S = 1$ 也有溢出。

最后可得

$$P/V = S \oplus Cy \quad (1-1)$$

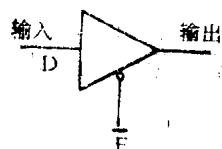
(6) 内部数据总线及缓冲器、锁存器

Z80CPU 内部数据的传送都是在内部数据总线上进行的，这种总线称为单总线。采用单总线的优点是减少了引线所占面积，提高了芯片的集成度。缺点是任何时刻只能有一个内部电路使用它。也就是CPU内部各单元电路只能分时使用内部总线，因而速度受到了影响。

Z80CPU 与其它部分各单元之间的联系是通过片间总线，或称器件级总线来实现的。Z80CPU有三组片间总线：地址总线 $A_{15} \sim A_0$ 共16条，双向数据总线 $D_7 \sim D_0$ 共8条以及控制总线14条。地址总线信号是经过地址缓冲器ABB (Address Bus Buffer) 与其它器件进行联系的，数据总线则经过数据总线缓冲/锁存器DBB/L (Data Bus Buffer/Latch) 与其它器件

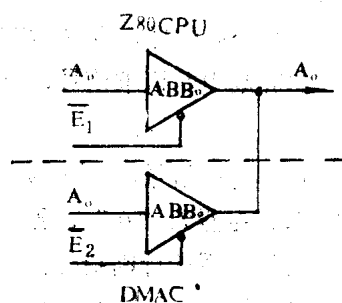
进行联系。ABB与DBB/L都是三态门电路。

三态门电路在总线接口电路中起着很重要的作用，一方面它可以起到多条互连线的相互隔离作用，另外也有传输速度快和驱动能力强的特性。图 1-4 (a) 示出了三态门的逻辑电路符号和真值表。当Z80 CPU与存储器直接访问控制器DMAC (Direct Memory Access Controller) 共同管理地址总线时，由于DMAC的地址总线也采用三态门输出，故可以简单地将这两个地址线连在一起，如图(b)所示。在使用时，如果DMAC要控制地址总线，只要使 $\overline{E}_1$ 为高电平，这时上面的三态门输出端呈高阻抗状态，对DMAC所给出的地址信号就不会产生影响。



输入		输出
$\overline{E}$	D	
低	低	低
低	高	高
高	X	高阻抗

(a)



(b)

图1-4 三态门

(a) 三态门逻辑电路符号和真值表 (X为无关值)

(b) 三态门的使用

锁存器实际上是一种寄存器，缓冲器/锁存器是寄存器带有三态输出门的电路。Z80CPU的双向数据总线就是这种电路。由于使用了三态门输出电路，所以很容易实现双向传输的功能，如图 1-5 所示。

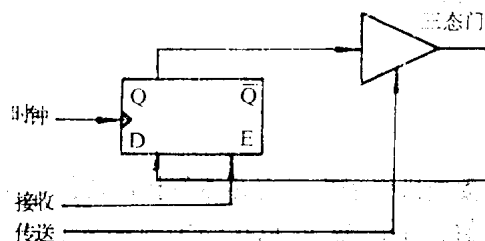


图1-5 双向缓冲器/锁存器

(7) 定时和控制单元部件：这一部件除电源外，可以分为四部份：

1) 系统控制部分，给出 6 个信号

① $\overline{M}_1$ ：指明现行的机器周期是取操作码周期。当操作码是两个字节时，每取一个字节，都发出 $\overline{M}_1$ 信号。 $\overline{M}_1$ 也与 $\overline{IORQ}$ 一起指示中断响应周期。

② $\overline{MREQ}$ ：存储器请求信号，指明地址总线上已给出了访问存储器的地址。

③ $\overline{IORQ}$ ：输入/输出 (I/O) 请求信号，指明地址总线低 8 位上已给出了访问 I/O 接口的地址。 $\overline{IORQ}$ 与 $\overline{M}_1$ 一起还用来表示中断响应，通知 I/O 接口把中断向量地址放到数据总

线上。在 M_1 期间可以进行中断响应操作,但绝不会发生I/O操作。

④ $\overline{RD}$: 读信号, CPU用它从存储器或I/O接口中读数据。

⑤ $\overline{WR}$: 写信号, CPU用它向存储器或I/O接口写数据。

⑥ $\overline{RFSH}$: 刷新信号, 指明地址总线的低7位上已给出动态存储器的刷新地址。刷新时还要用 $\overline{MREQ}$ 信号。

2) CPU控制部分: 有一个输出信号, 4个输入信号。

① $\overline{HALT}$: 暂停信号, 表示CPU已经执行了一条 $\overline{HALT}$ 指令并进入暂停状态, 这时CPU执行空操作NOP指令, 直到CPU接受 $\overline{NMI}$ 或 $\overline{INT}$ 信号, 操作才重新开始。在暂停期间, 仍维持动态存储器的刷新操作。

② $\overline{WAIT}$: 等待信号。由慢速存储器或I/O接口发出此信号, 告诉CPU再等待一段时间, 直到存储器或I/O已准备好而撤销此信号为止。CPU接到 $\overline{WAIT}$ 信号后, 通过插入等待周期进行等待。

③ $\overline{INT}$: 中断请求信号, 由I/O接口发出。

④ $\overline{NMI}$: 非屏蔽中断请求信号, 也由I/O接口发出。非屏蔽中断比一般中断的优先权高。

⑤ $\overline{RESET}$: 复位信号, 用来使CPU复位。其功能包括置PC为“0”, 清除中断允许触发器, 置I和R寄存器为“0”, 置中断方式0。将地址总线 and 数据总线处于高阻状态, 使所有控制信号无效, 不进行刷新等。

3) 总线控制部分, 当DMAC要控制CPU的地址总线、数据总线和控制信号线(称控制总线)时, 应向CPU发一个总线请求信号 $\overline{BUSRQ}$, 即提出占用总线的请求。一般来讲, CPU会在一个机器周期的最后节拍, 发出一个总线响应信号 $\overline{BUSAK}$, 告诉DMAC, CPU的地址总线、数据总线和三态输出的控制总线已处于高阻状态, I/O设备就可以使用这些总线了。

CPU控制总线信号有两个特点, 一是全部信号都是低电平有效, 这是因为负逻辑信号的抗干扰能力较强之故。另一个特点是, 除 $\overline{HALT}$ 、 $\overline{RFSH}$ 和 $\overline{BUSAK}$ 之外, 所有输出控制信号均为三态输出。

4) 时钟电路(又称节拍电路)。Z80CPU要求外部提供时钟信号 ϕ 。如前所述, Z80使用2.5MHz时钟, Z80A使用4MHz时钟。

二、Z80的寻址方式

“寻址”就是寻找操作数的地址。由于小型、微型机多采用多寄存器结构, 使得CPU操作很灵活。例如, 一条指令, 其操作数既可以在存储器中, 也可以在CPU内部某个寄存器中, 甚至还可以在I/O接口中。这种在指令中规定操作数地址的方式叫做寻址方式。在Z80中有十种寻址方式。

1. 立即寻址 (Immediate addressing)

这种寻址方式中, 指令的操作码之后, 就是操作数本身。例如欲将常数5送入累加器A, 可用指令LD A, 5来实现。操作码按规定应为3E, 操作数则为5。机器码指令格式应为:

第 1 字节	3 E	操作码
第 2 字节	5	操作数

2. 扩展的立即寻址 (Immediate extended addressing)

这种寻址与上一种相同，只是操作数为两字节16位数，而且要使用寄存器对。例如欲将常数1000H(H表示16进制数)送入HL寄存器对，可写为LD HL,1000H。机器指令格式应为

第 1 字节	21	操作码
第 2 字节	00	操作数低 8 位
第 3 字节	10	操作数高 8 位

注意，在操作码后面的第一个字节是16位数的低 8 位，第二个字节是高 8 位。

3. 寄存器寻址 (Register addressing)

指令中规定操作数就是CPU内部某一寄存器的内容，这就是寄存器寻址。

4. 寄存器间接寻址 (Register indirect addressing)

这种寻址方式与寄存器寻址不同的是寄存器的内容不是操作数，而是操作数所在的存储器单元地址。例如将地址2000H的内容7EH传送到累加器A中，则可用指令LD A, (HL)，但HL的内容必须为2000H，如图1-6所示。

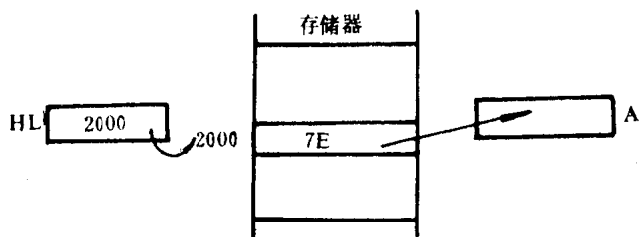


图1-6 寄存器间接寻址示意图

5. 扩展寻址(Extend addressing)

这种寻址中，操作码后面的两个字节是16位的操作数地址。当然，该地址也可作为程序的转移地址。

第 1 字节	操作 码
第 2 字节	低位地址
第 3 字节	高位地址

6. 隐含寻址 (Implied addressing)

隐含寻址是指令中没有明确指出操作数，但暗含着存放操作数的寄存器。例如，指令：ADD B，表示寄存器B的内容与累加器A的内容相加，并将其结果保存在累加器A中。此处A在指令中未出现。

7. 相对寻址

用操作码后面的一个字节来表示相对于当前程序地址的偏移量（用2的补码表示）。将此偏移量与程序计数器PC中的内容相加，就可得到程序偏移后的地址。由于PC之值在取指令之后已经修改为PC + 2，所以第二字节的偏移量应该减去2：

PC	操 作 码	
PC + 1	偏 移 量	
PC + 2		←自PC + 2算起的偏移量

例如转移指令JR e是将程序转移到PC+e-2处,这里的PC应修改(即加2)了的PC值。

由于大多数程序要转移的程序段都在转移指令的附近,因此使用相对寻址的转移指令很方便,且指令只有两个字节,可以节省存储器地址空间。另外,带有相对寻址的程序,也便于程序浮动。

8. 变址寻址 (Indexed addressing)

这是一种三字节指令寻址格式,其中两个字节为操作码,一个字节为位移量d(用补码表示),其实际地址是变址寄存器之值与位移量之和(图1-7)

变址寻址方式适合于对连续数据区(即对数据表)的访问,因只要用IX(或IY)所规定的数据表的起始地址,而令d从0逐渐增加,即可连续访问这一数据表。

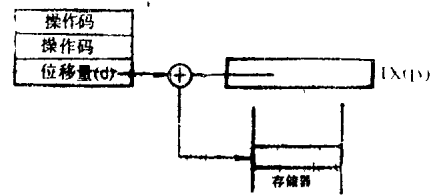


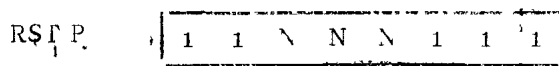
图1-7 变址寻址示意图

9. 按位寻址 (Bit addressing)

用于这类寻址的指令称为位操作指令,它可以实现对CPU内部寄存器、存储器以及I/O接口中的寄存器的某一位进行操作。

10. 修改零页寻址 (Modified page zero addressing)

这种寻址方式用于重新启动指令RST P,这是一条在中断响应(Z80中断方式0时)中,由I/O接口形成的一条单字节指令。指令格式为:



其中NNN为000~111八个数,由此八个数对应了存储器0页(Z80存储器以256个单元为一页)的00H、08H、10H、……38H八个地址,称为中断处理程序的入口地址。换句话说,就是由NNN决定了RST P指令中的地址P之值。

三、Z80的指令系统

Z80CPU共有158条基本指令。按照功能,这些指令可分四类,它们是

- (1)数据操作指令 (Data Manipulation Instructions)
- (2)数据传送指令 (Data Transfer Instructions)
- (3)程序操作指令 (Program Manipulation Instructions)
- (4)状态管理指令 (Status Management Instructions)

1. 数据操作指令

(1) 算术运算指令

在八位微型机中,算术运算指令主要是单字节的加、减法运算,也可以进行双字节的加、减法运算。在需要进行多字节加、减法运算时,则需进行几次加、减法操作。而且在对高位字节进行运算时,应考虑由低位字节来的进位值。这时使用带进位的加减指令就很合适。

加1增量 (INC)、减1减量 (DEC) 指令多用于循环程序中。因这两条指令可以使通