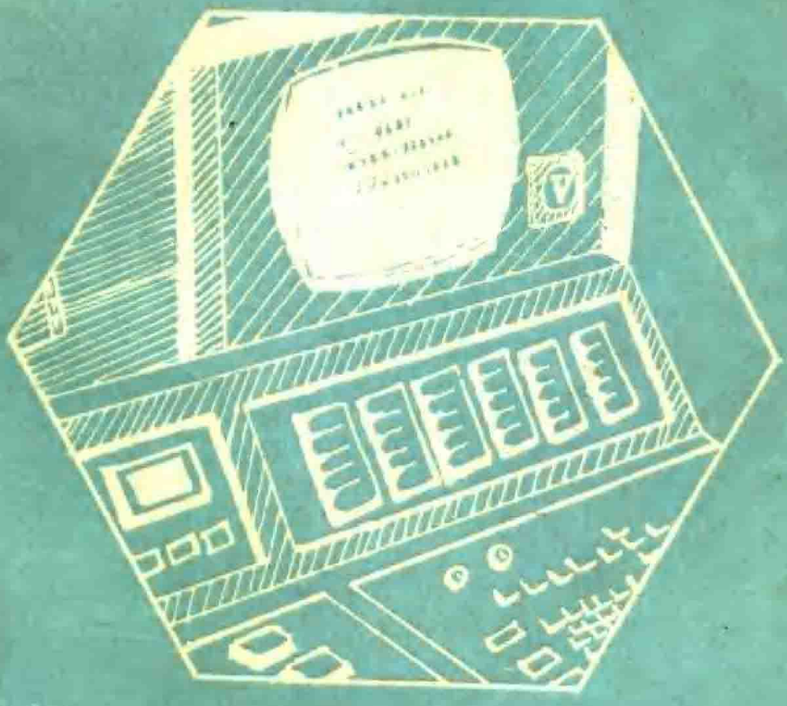
微处理器与微计算机

杨文福
朱家铿
朱慕荣
合编

何文兴
审校



沈阳市科协科技教育部编印

1980

前言

微处理器和微计算机的产生是大规模集成电路和计算技术发展的必然产物。它具有结构简单、价格低廉、型小体轻和可靠性高的特点。从1971年第一片微处理器问世以来，微型机得到了迅猛的发展。在短短的七、八年时间内，已发展到所谓“第四代”微计算机。目前微型机已达到16位字长，基本指令执行时间小于100ns，寻址能力达到1兆字节。微型机的发展使小型机生产厂受到很大冲击，促使他们走上小型机微型化的道路。与小型机NOVA、PDP—8、PDP—11具有同样功能的微型机也相继问世。

微型机的应用，在国外已广泛应用到生产过程控制、数据处理、监视检测等方面，也已深入到家庭的日常生活中去。国外也正在研制由微型机组成大型计算机系统。由1台NOVA机和10—20台16位字长的Intel—3000系列微型机组成1台IBM370/168大型机，它的运算时间大大缩短，价格也从400万美元降到8万美元。在此同时，用于微型机的软件和小型外围设备的研制工作也已取得可喜的成果。

在我们国内也正在急起直追，在上海、西安、沈阳等地已有微型机的样机问世，同时也从国外引进了多台微型机，很多单位和部门在从事这方面的工作。全国已成立微计算机技术情报协作组。但目前，国内微型机的价格太贵，小型外部设备也没有配套。如不解决这些问题，就很难广泛推广和应用微型机。在这方面还要进行相当艰巨的工作。

为了帮助大家熟悉微型机，我们选择了几种典型的机种作一介绍。本书是在东北工学院何文兴副教授讲课的基础上作了较大的补充和修改而编写的；由于我们没有实践基础，只是阅读一些有关资料而编写的，错误和缺点难免，希广大读者提出批评和意见。

本书第一章、第四章由朱家铿老师编写，第二章由朱慕荣老师编写，第三章由杨文福老师编写，第五章由朱慕荣、杨文福老师合编。全书由何文兴副教授最后审校。

目　录

第一章　8080A系列微计算机

第二章　M6800微处理机

第三章　微型计算机Z—80

第四章 Intel—300微型计算机

第五章 应用举例

第一章　Intel8080A系列微计算机

一、Intel8080A中央处理器

Intel8080是在1973年问世的第二代微处理器。它与8008相比，速度差不多提高了十倍。这是因为CPU采用n沟道MOS电路，并且采取了16根地址总线与8根数据总线分开的双总线结构的缘故。另外由于堆栈不在CPU芯片之内，而是利用存储器的一部分，这样可以不加限制地构成嵌套子程序。之后，Intel公司又在此基础上改为增强型8080A。

1、Intel8080ACPU的总体结构

8080A是八位并行微处理器，有一条16位的地址总线，一条8位双向数据总线。对存储器的寻址范围为0～64K，能选择256个输入口和256个输出口。8080A有基本指令78条，四种寻址方式（立即寻址、直接寻址、寄存器寻址及间接寻址），采用2相非重迭的时钟，主频2 MHZ，最短指令执行时间为2 μS。

8080A是由四个功能部件所组成：寄存器阵列及地址控制逻辑，算术及逻辑运算单元、指令寄存器及控制部分、双向三态数据总线缓冲器，它们是通过内部数据总线而联系的。其内部结构框图如图1—1所示。

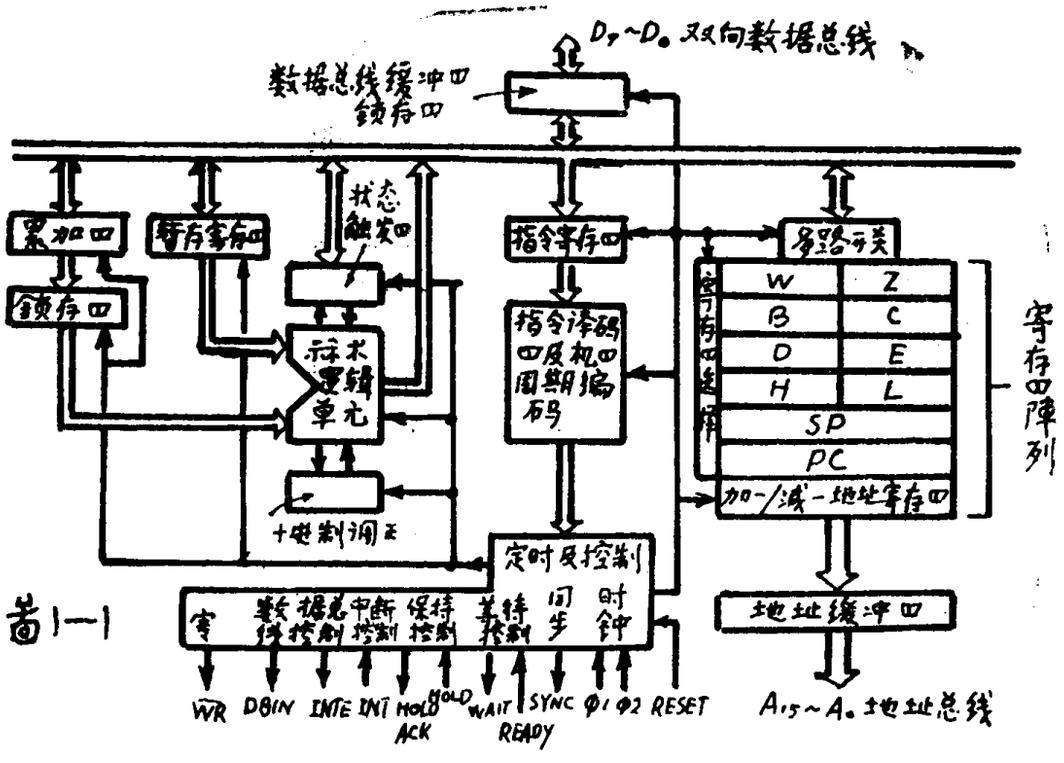


图1—1

8080逻辑框图

（1）寄存器阵列与地址控制逻辑

寄存器阵列是一个静态读写存储器（RAM）的阵列。它由6个16位的寄存器构成。其中可供程序员使用的工作寄存器有：

程序计数器（PC），它保存现行程序指令的存储器地址，每当取出指令后，它就自动加1。

堆栈指示器（SP），它保存下一个可供使用的存储器中堆栈单元的地址。数据压入堆栈时，（SP）—1，数据弹出堆栈时（SP）+1；

六个八位通用寄存器，B、C、D、E、H、L，它们与累加器A一起可用来存储数据或地址。它们既可作单个寄存器使用，也可作寄存器对（16位）来使用。它们由B—C，D—E，H—L组成三对寄存器对。写“寄存器对B”即表明是寄存器对B—C。寄存器对用来存放16位数据或地址时，高八位存在B、D、H中，低八位存在C、E、L中。

由寄存器选择线路来确定6个8位寄存器中的一个，并通过多路转换开关将选中的寄存器与内部数据总线间互相进行传送8位字长的数据。

16位的地址锁存器可从三个寄存器对B、D、H中任选一对，接收16位字长的数据。并经由它送到16位的地址缓冲器，或送到16位的加1/减1器上。加1/减1器从地址锁存器接收数据，加1或减1后将数据送回到寄存器阵列。因此，16位的数据能够在寄存器对、地址锁存器、加1/减1器之间进行传送。地址缓冲器输出到外部地址总线上。

另外，暂存寄存器对WZ不能用程序来选址，只有在CPU内部执行指令时才用到它。

（2）算术及逻辑运算单元（ALU）

ALU包含有下面这些单元：一个八位的累加器A，一个八位的暂存累加器ACT，一个八位暂存寄存器TMP，一个八位的运算器ALU，一个五位条件标志寄存器（进位CY、零位Z、符号位S、辅助进位AC及奇偶位P）及十进制调整线路。

算术、逻辑、移位或循环操作由ALU来执行。ALU的运算数据从TMP、ACT及进位触发器输入，运算结果送内部数据总线或累加器A，同时也将其一些状态送条件标志寄存器，以便作转移测试条件用。

累加器A从ALU及内部数据总线接收数据，并输出数据到ACT及内部数据总线上。在进行十进制调整时，通过检测条件标志进位CY，辅助进位AC，由ACU完成对累加器A中之数进行十进制调整。

暂存寄存器TMP从内部数据总线接收信息。并能将它的信息的全部或部分送到ALU、条件标志寄存器或内部数据总线。

（3）指令寄存器及控制部分

在取指令周期，指令的操作码从内部数据总线传送到8位的指令寄存器，指令寄存器又将操作码送指令译码器。译码器的部分输出结合各种定时信号，用组合逻辑（寄存器选择与控制、地址总线控制电路）形成寄存器阵列所需要的控制信号。译码器的另一部分输出，结合定时信号，用组合逻辑（运算器控制电路）形成算术及逻辑运算单元所需要的控制信号。

定时及控制线路根据指令译码器的输出和从外部来的控制信号产生机器的状态及周期的定时信号，并向外部输出信号或回答信号。

（4）双向三态数据总线缓冲器/锁存器。

8位双向三态总线缓冲器用于隔离CPU内部数据总线和外部数据总线D_0～D_7。在输出方式时，内部数据总线的内容被送进一个8位锁存器，再送至数据总线的输出缓冲器。在输入方式时，数据从外部数据总线传送到内部数据总线。当没有传送操作时，输出缓冲器被关闭。

2、8080A的机器周期和状态

任何CPU的工作都是周期性的，它由时钟提供定时信号，使CPU能取出一条指令，执行本指令的操作，再取下一条指令等等。我们称一条指令的取出和执行所需的时间为“指令周期”，完成某一确定动作（如取指令或存储器读出）的时间称为“机器周期”，每个机器周期又包含3—5个状态，每个状态的时间，宽度为时钟脉冲的间隔，称为“时钟周期”。它是处理动作的最小单位，如PC＋1，给出地址、取数等。

为了完成取指令及执行指令的动作，一个指令周期要求有一到五个机器周期，用M_1，M_2……M_5来表示，而每个机器周期又要求三到五个时钟周期，用T_1，T_2……T_5来表示。

一条指令需要多少个机器周期，主要取决于CPU在一条指令中需要访问存储器的次数或可寻址的外部设备的次数，因为每个机器周期仅发送一次地址。如取和执行一条指令需要访问存储器两次，则此指令就包括两个机器周期。

8080A的每一个机器周期对应于一个基本操作。一共有十个基本操作。任何一条指令的第一个机器周期一定是“取指令”操作。CPU执行的十种基本操作，是由每个机器周期的第一个时钟周期期间发出的8位状态信息（表1—1）来区分。给定的状态信息在SYNC（同步）时间间隔内出现在数据总线上。这个信息保存在外部锁存器中，可以用它来产生对外部电路的控制信号。

数据总线位	周期信息	取指令	存储器读	存储器写	堆栈读	堆栈写	输入读	输出写	中断响应	暂停响应	暂停时中断响应
		1	2	3	4	5	6	7	8	9	10
D_0	INTA	0	0	0	0	0	0	0	1	0	1
D_1	WO	1	1	0	1	0	1	0	1	1	1
D_2	STACK	0	0	0	1	1	0	0	0	0	0
D_3	HLTA	0	0	0	0	0	0	0	0	1	1
D_4	OUT	0	0	0	0	0	0	1	0	0	0
D_5	M_1	1	0	0	0	0	0	0	1	0	1
D_6	INP	0	0	0	0	0	1	0	0	0	0
D_7	MEMR	1	1	0	1	0	0	0	0	1	0

表1—1 8位周期信息表

除暂停、中断、保持三种基本操作外，一个机器周期可分为3—5个状态（T_1，T_2……T_5）。

T_1：用ϕ_2的上跳沿产生一个同步信号SYNC，它表示每个机器周期的开始，并送

外部电路作选通信号用（此同步信号在 T_2 的 ϕ_2 上跳沿时结束）。此时，在数据总线 $D_7 \sim 0$ 上出现本机器周期的信息码，并送到外部的锁存器中，以产生外电路的控制信号。如存储器读、写，I/o读、写等信号。同时在 ϕ_2 的上跳沿，CPU 的地址总线接收程序计数器PC来的指令地址信息。这一信息一直保持到T_3状态。

T_2：为了使CPU能与慢速存储器或I/o设备同步工作，当CPU向存储器或I/o发出地址信息后，就要进行查询。确认有无READY和HOLD信号，并检测是否为暂停指令。

Tw：如果READY 是低电平或已检测到是执行暂停指令，则CPU进入等待状态。此时通过WAIT线输出高电平，说明CPU处于等待状态。一直到READY变为高电平。Tw的延续时间为时钟周期的整数倍。HOLD信号和暂停指令在下面将详细讨论。

T_3：当READY是高电平，或者没有HOLD 信号或暂停指令时，即进入T_3状态。在此时钟周期内，将指令字节（当是“取指令”机器周期时）或数据字节（当是“存储器读出”、“堆栈读出”、或“输入读出”周期时）或中断指令（当是“中断响应”周期时）从数据总线输入CPU；或者当“存储器写入”、“堆栈写入”或“输出写入”周期时，CPU把数据字节输出到数据总线上去。

T_4、T_5：如果执行一条特定指令，则需要有 T_4和 T_5状态。如果不需要，CPU可以跳过它们中的一个或两个。T_4与T_5仅用于 CPU的内部操作。例如完成对寄存器对的内容加 1 后，送回寄存器对中去。

并不是所有机器周期都有这些状态，它取决于被执行的指令类型和在这个指令周期内具体的机器周期。CPU的任何一个机器周期，只要其处理活动完成了，即直接进行到下一个机器周期，而不需要每次都进入到T_4及T_5状态。上述五个状态示于图 1—2。

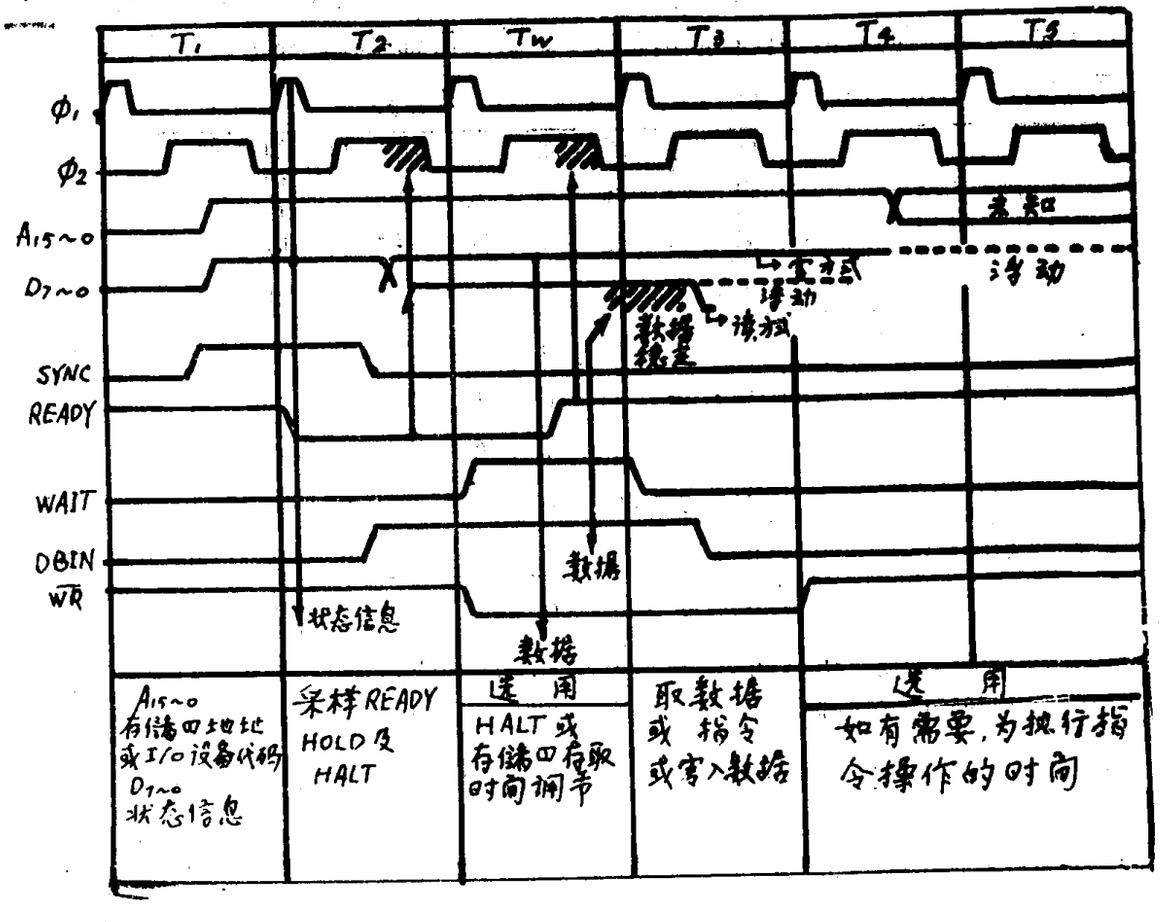


图 1—2 8080A五个状态时序图

下面举两条指令操作过程的例子：

①MOVr1,r2：这条指令的意思是“寄存器r 2 的内容转移到r 1 去。”这是一条一字节指令，因此只需要一个机器周期。如用SSS代表源寄存器r 2 的代码，则DDD代表目的寄存器r 1 的代码。用表格形式表示它的操作过程：

助记码	操作码	M_1				
	$D_7D_6D_5D_4D_3D_2D_1D_0$	T_1	T_2	T_3	T_4	T_5
MOV r1, r2	0 1 D D D S S S	PC 输出状态	PC = PC + 1	指令→指令寄存器	(SSS) →TMP	(TMP) →DDD

②SHLD addr：这是一条三字节指令，addr是两字节长的操作数。这条指令的意思是“L寄存器的内容存在以操作数为地址的存储单元内，H寄存器的内容存在下一个较高地址的存储单元内。”

助记码	操作码	M_1				M_2		
	$D_7D_6D_5D_4D_3D_2D_1D_0$	T_1	T_2	T_3	T_4	T_1	T_2	T_3
SHLD addr	0 0 1 0 0 0 1 0	PC 输出状态	PC = PC + 1	指令→指令寄存器	(1) ×	PC 输出状态	PC = PC + 1 B2	(2) →Z

M_3			M_4			M_5		
T_1	T_2	T_3	T_1	T_2	T_3	T_1	T_2	T_3
PC 输出状态	PC = PC + 1 B 3 —	(2) →W	WZ 输出状态	(L) —— WZ = WZ + 1	→数据总线	WZ 输出状态	H —— WZ = WZ + 1	→数据总线

注：（1）在取指令周期，T_3时指令送进指令寄存器，随之要译码和付诸执行，这至少要一个机器状态T_4。

（2）B_2，B_3为指令的第二、三字节，一般都是操作数。

3、8080A引线功能说明

8080A共有40条引线，其中电源、地占4条，即+5 V，-5 V，+12V及地，另有输入输出线共36条：

（1）A_{15-0}地址总线（输出三状态）

它可提供64K 8 位字节的存储器地址，或者表达256个输入或256 个输出设备的设备代码。

（2）D_{7-0}数据总线（输入输出三状态）

它提供CPU与存储器和I/o设备之间进行指令与数据传送的双向通路，在每个机器周期的第一个时钟周期，在数据总线上输出周期信息码。

（3）SYNC同步信号（输出）

它表示一个机器周期开始的信息。

（4）READY准备就绪（输入）及WAIT等待（输出）

它常用来使CPU与慢速度的存储器或I/O设备同步工作。当CPU送出一地址后，在READY输入端如是低电平，则CPU进入“等待”状态，此时，在输出线 WAIT上为高电平。因此，WAIT输出线是用来识别CPU是否处于“等待”状态之中。

（5）HOLD保持（输入）及HLDA保持响应（输出）

当外部设备向HOLD输出高电平时，即向CPU申请“保持”，CPU在T_3状态后完成了对地址及数据总线的使用，即响应此“保持”申请，CPU的地址总线及数据总线都处于高阻态。同时，通过HLDA输出高电平来表示它已响应“保持”申请。在“保持”期间，地址总线与数据总线就处于发出申请的那个外部设备的控制之下，进行存储器的信息传送。当外部设备已完成它的数据传送后，HOLD申请结束，HLDA 恢复为 低电平。CPU又恢复到上一个已执行了的周期后面的一个机器周期。

（6）RESET复位（输入）

当它输入高电平时，可使程序计数器PC清零，但不清除其它工作寄存器，因此复位后，CPU从零单元起开始执行程序。

（7）DBIN数据总线允许输入（输出）

此信号送给外部电路，说明数据总线处于输入工作状态，用这个信号来开放从存储器或I/O输入到CPU数据的门控电路。

（8）$\overline{WR}$写（输出）

当$\overline{WR}$为低电平时，CPU为写状态。

（9）INTE中断允许（输出）

当I NTE为高电平时，才允许外部来的中断申请。当接受中断申请后，INTE变为低电平，禁止下一个中断。在复位或执行DI（关中断）指令后，此 输出线为 低电平，而禁止中断。当执行EI（开放中断）指令后，INTE为高电平，允许中断申请。

(10) INT中断申请（输入）

CPU在执行完一条指令后，或CPU处于暂停时，可在此输入线上识别中断。当中断允许线处于高电平时，则CPU能响应中断。如果CPU处于HOLD状态或 INTE处于低电平，则CPU不能响应此中断申请

(11) ϕ_1、ϕ_2时钟（输入）

CPU工作的定时信号

4、中断时序

假如计算机正处理大量数据，其中有一部分数据要输出打印。CPU只要三个 基本机器周期就可输出一个字节数据。打印机却很慢，造成CPU“空等”。采用中断 处理后，CPU输出一个字节数据后，仍继续运算。当打印机打印完一个字节 数据后，准备好接收新的数据，它就向CPU申请中断。如CPU响应这一中断，就暂停主 程序，自动转到输出下一字符。

8080A是这样来处理中断的：当外部设备发出一中断 请求时，即把高电平 输入到CPU的INT端，只要CPU的中断允许端INTE为高电平时，在现行指令 执行完后，就

可响应中断申请，即完成下述工作：

（1）将INTE端变为低电平，禁止再次中断申请。

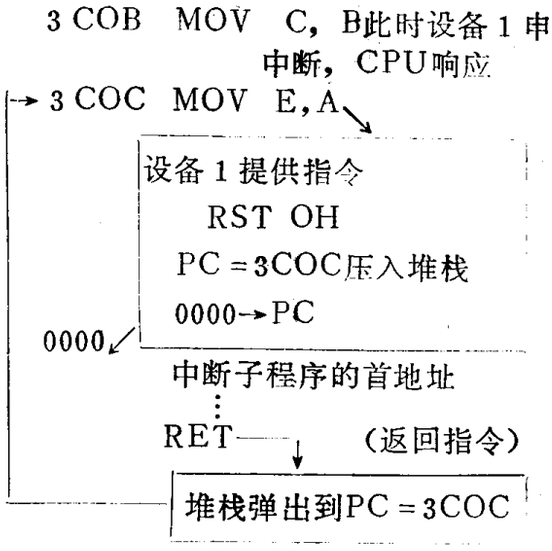
（2）由中断设备给出一条适应中断处理的特殊的一字节调用子程序指令，即重新启动指令RST。

（3）将 PC 中所存的返回地址压入堆栈，作为以后返回指令用的返回地址。

RST指令的格式如下：

7							0
1	1	N	N	N	1	1	1

NNN 为000～111 范围内的数，每一个数对应一个中断设备。执行 RST 指令，除将PC 的内容压入堆栈外，且使 PC 的内容为0000000000NNN000。因此，给出了相应中断的服务子程序的首地址，以保证CPU在响应中断后能自动指向相应 的服 务子程序。显然，对应于每一中断服务子程序的首地址分别为十进制的0、8、16、24、32、40、48、56。对不同的中断设备可事先预以编码。这样，每一个子程序只有8个存储单元。如果8个字节的子程序不够，可以在其中安排一条调用更长的子程序的指令来处理中断。下面举一个具体例子，说明处理中断的过程。



5、保持时序

微型机可以执行直接存储器存取（DMA）的操作。一般 CPU 控制全部 数据的传送，数据一定要经过CPU。如果有成批数据要在外部设备与存储器 间传送，都要经过CPU的话，那是很不经济的。因而8080A配有直接存取（DMA）电路。当外 部设备发出请求保持状态，CPU使它的HLDA（保持响应）输出端为高电平，CPU停止操 作，退出对地址总线和数据总线的控制。在HOLD期间，两条 总线处于请求保持的 设备控制之下，并直接与存储器进行数据传送。当完成数据传送时，HOLD结束，CPU 就恢复到原执行的机器周期的后面的机器周期。

从CPU的外部来看，一旦 CPU 响应保持，地址和数据总线变为高阻态，处理器就停止操作。然而从CPU内部来看，CPU的一定的功能仍可继续。如果一个HOLD申请在T_3被响应，而CPU又处在需要用T_4和T_5状态才完成的机器周期之中，则 CPU 进入保持之前，仍继续执行T_4T_5状态。这样，CPU的内部处理和外部的 DMA 传送相重叠，节约了CPU的时间。

6、暂停时序

当执行一条暂停指令（HALT）时，CPU在下个机器周期的T_2状态过后进入暂停状态（TWH）。8080A有三种方法脱离暂停状态。

（1）RESET（复位）端上输入一高电平，使8080A恢复到T_1状态，同时 PC 清零。CPU开始从存储单元“0”处执行。

（2）HOLD状态的输入会使8080A进入保持状态。当HOLD转为低电平，8080A在下一个ϕ_1时钟脉冲的上升沿重新进入暂停状态。

（3）中断响应将使8080A脱离暂停状态，并在下一个ϕ_1时钟脉冲上升沿进入T_1。CPU执行由外部进入数据总线的指令。注意，当进入暂停状态，允许中断（INTE）标志必须置“1”，否则8080A不能响应中断，要脱离暂停状态就只有经过 RESET信号。

整个CPU 的状态转换可以 用CPU的状态转换图来说明。见图1—3。

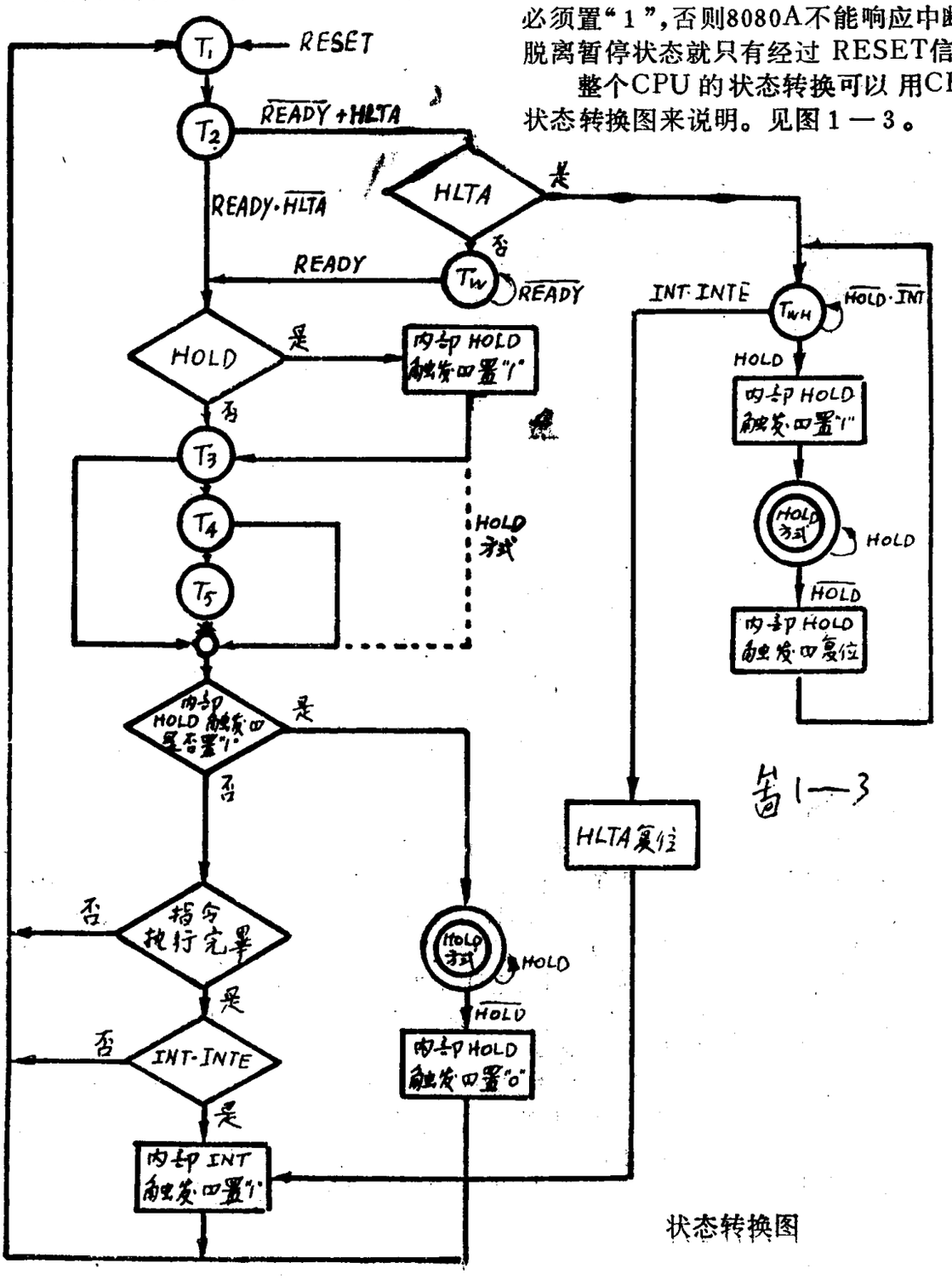


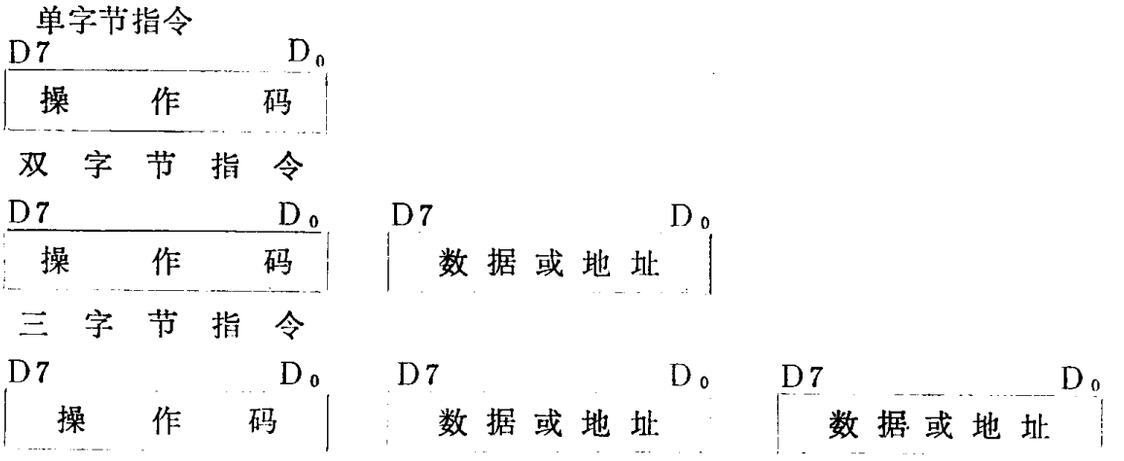
图1—3

状态转换图

二，Intel8080A的指令系统

1、指令的格式与寻址方式

每8位二进制数组成指令的一个字节。在8080中指令的长度可以有1、2和3个字节等三种形式。多字节指令应该存在相继的存储器单元中。第一字节的地址常常称作指令的地址。



指令的具体格式是根据指令的具体操作而定的。例如MOV r1,r2是单字节指令，SHLD addr是三字节指令。数据和指令一样也存放在存储器的相继单元中，数据的低位字节在前，随后是逐渐增高的高位字节。

8080A指令的寻址方式一共有四种。

（1）直接寻址：它是三字节指令，指令的第二、第三字节指出了指令操作地址。地址的低位是第二字节，第三字节就是地址的高位。例如：指令LDA 1F2A（H），指令的定义是“取存储地址1F2A之内容送到累加器A中”。这条指令在存储器内的形式如下：

存储器地址	存储器内容
05AE（任意）	3A（LDA操作码）
05AF	2A
05BO	1F

（2）寄存器对间接寻址：在指令中就指定了一对寄存器，用它来存放存储器的地址。对大多数8080A的指令，是用H和L这对寄存器来作地址寄存器用。只有两种指令才用寄存器对B、C或D、E作地址寄存器。例如：指令ADDM，它的含义是“以寄存器对H、L的内容为地址的存储器数据与累加器A的数相加后仍送入累加器。”它是单字节指令。

（3）立即寻址：指令的操作数据即包含在指令内，不必去寻址。它有双字节和三字节指令。指令的第二或第二、三字节为立即操作数。例如：指令LXIrp，data，它的含义是“将指令中的数data送入rp指定的寄存器对中去。”它是一条三字节指令。

（4）堆栈指示器寻址：存储单元可以用16位堆栈指示器来指示地址。通过堆栈指示器寻址的操作称为堆栈操作，送数据到堆栈称为推入（压入），从堆栈取出数据称为弹出。

当堆栈推入操作时，将一个寄存器对或程序计数器的16位数据传送到存储器的某一区域内。此存储区域的地址由堆栈指示器确定。

16位数据的高 8 位存放在比堆栈指示器的内容少 1 的存储器地址，数据的低 8 位存放在少 2 的存储器地址。操作终了时堆栈指示器自动减 2 。

当堆栈弹出操作时，16位数据是从存储器的某一区域传送到某一寄存器对或16位程序计数器。此时，将堆栈指示器所指示的存储单元的内容送往寄存器对的第二个寄存器或程序计算器的低有效位 8 位。而堆栈指示器加 1 后指示的存储单元的内容，送往寄存器对的第一个寄存器或程序计数器高有效位 8 位。然后堆栈指示器再自动加 1 。

2 、指令的符号

为了阅读指令方便起见，下面介绍一下指令系统中所用到的符号和标志。

符号

addr　　16位地址数

data　　8 位数据数

data16　　16位数据数

B 2 , B 3　　指令的第二、三字节

r_1, r_2, r　　寄存器A、B、C、D、E、H、L中的一个，它的格式为：（DDD目的寄存器，SSS源寄存器）

DDD或SSS	寄 存 器 名 称
000	B
001	C
010	D
011	E
100	H
101	L
111	A

RP　　指定某个寄存器对

RP	寄 存 器 对
00	B—C
01	D—E
10	H—L
11	SP

rh　　指定寄存器对中第一个寄存器。

rL　　指定寄存器对中第二个寄存器

CCC	表示状态标志位的状态：	
000	Z = 0	Z—零标志
001	Z = 1	
010	Cy = 0	Cy—进位标志
011	Cy = 1	
100	P = 0	P—奇偶性标志
101	P = 1	
110	S = 0	S—符号标志
111	S = 1	

（　）括孤中所示为存储器单元或寄存器内容。

← 数据传送符号，读作“传送到”，

∧　逻辑“与”

∨　“或”

⊻　“异或”

+　加

−　2的补码减法

*　乘

$\bar{A}$　A的反码

NNN　为二进制数000到111中的任意一数，NNN000为重新启动指令（RST）的转移地址。

M　存储器地址

3、指令系统

8080A的指令系统按其功能可分下面5大类：

（1）数据传送类指令：这一类指令传送数据出入寄存器和存储器。这类指令不影响条件标志状态。

例如：MOVr1，r2这是一条在寄存器间传送数据的指令。r2为源寄存器，r1为目的寄存器。即为r1←（r2），寄存器r_2的内容送入寄存器r1。它是单字节指令，在1个机器周期或5个时钟周期内完成。该指令的操作码为：01DDDSSS

如对MOV、H，B这一条具体指令的操作码为：01100000

它的操作功能为H←（B）

（2）算术类指令

这类指令对存于寄存器及存储器中的数据，进行算术操作。除了另行规定外，所有这类指令的操作结果按规则对零、符号、奇偶性、进位、辅助进位等标志产生影响。所有减法操作都是按2的补码运算，如果有借位时进位标志置1，否则置0。

例：ADI data，这条指令是将指令的第二字节内容与累加的内容相加，其结果放在累加器里。即A←（A）+（B2）。它是二字节指令，在2个机器周期或7个时钟周期内完成。该指令的第一字节为操作码，第二字节为数据：

第一字节	1	1	0	0	0	1	1	0
第二字节	数					据		

例：DAA，这是一条累加器的十进制调整指令。它是把累加器的8位二进制数码调整为二位二—十进制数码。

我们知道一个四位二进制数要转换成二—十进制数有两种情况，当二进制数≤9时，就不需要转换，当二进制数＞9时，就要加6即为二—十进制数，并对高位有进位。例如二进制数是12，则要在这四位数中留下2，并向高位进1。而四位二进制数是逢16进1，因而当12＋6后，16进1到高位，剩下2留在本位。

十进制调整的步骤如下：

①如果累加器的低4位，其值大于9或AC标志置位，只要有其中一种情况，累加器低4位加6，并向高4位进位。

②如果累加器的高4位，其值大于9或CY标志置位，则累加器的高4位加6。

这是一条单字节指令，在一个机器周期或4个时钟周期内完成。它的操作码为：

0	0	1	0	0	1	1	1

（3）逻辑类指令

这类指令对寄存器和存储器中的数据以及条件标志执行逻辑操作。可以用来比较两数的大小，检查一个数是否为零等操作。除了另有规定外，这类指令的结果按规则置位或复位Z、S、P、AC、Cy等标志状态。

例：CMP M，这条指令把累加器的内容减去地址存在寄存器H和L的存储器单元内容。累加器的内容保持不变。用符号来表示为：（A）—（H、L）。条件标志的置位根据减的结果而定。如（A）＝（H、L）则Z＝1，如（A）＜（H、L），Cy＝1。它是单字节指令，在二个机器周期或7个时钟周期内完成。该指令的操作码为

1	0	1	1	1	1	1	0

（4）转移类指令

这类指令改变程序流程的正常次序，它不影响条件标志。转移指令有两种类型：无条件转移和条件转移。无条件转移只对程序计数器的内容进行改变。例如指令JMP addr是一3字节指令，它将第2第3字节的数据送入程序计数器。下一条指令就从新地址开始执行。而条件转移指令要检测4个条件标志中的某一个的状态来决定是否执行转移指令。这些条件是：