TP译文

Microcomputer

Z-80的彻底研究

2

北京工业大学自动化系研究室
北京微型电脑应用协会

Z80的彻底研究

（日）森宏等

张学志 译

陈 平 校

参加本文审校的还有：方滨 芦乃谦 沈扬等同志

第一章　Z80系统的构成及方法

本章详细介绍使用Z80系列构成Z80系统的设计 方法以及Z80所独具的特点——优先链和方式 2 中断。

1.1 系统的构成

微计算机（以下称微型机）系统根据用途不同，结构也不一样。基于深入追求经济性和工作特性的要求，必须针对具体对象确定系统的最佳结构。由于微型机具有一定的自由度及可挠性，所以系统结构即使固定下来，在具体应用上也还有一定的伸缩性。

以个人计算机为例，这种结构可以用于家务，可以用于办公室进行事务处理，还可以用于研究室搞实验。所以，同一种结构能满足多种不同的用途。因此象这种通用的机器在结构上都有一定程度的伸缩性。存储器容量也是这样，虽然一般情况不会使用全部存储器容量，但多数还都是充分安装，其目的就是为适应不同用途的需要。

1.2 系统的扩展

系统扩展的问题是一个需要甚重考虑的问题。有人认为既然小规模系统容易扩展，大规模系统也不会有什么问题。事实并不如想象的那样简单。因为小规模系统与大规模系统是不一样的。如果把小规模系统扩展到大规模系统，该系统就必然会复杂得多。

譬如，使用 2 K字节存储器的系统可以不使用译码器。但是扩展到64K字节，至少 也要用 2～3 个译码器。如果再把I/O和程序等都包括进去，那么系统结构必然变得繁杂。

因此，在系统设计的最初阶段就要充分考虑结构的复杂性和扩展性问题，否则以后就 被动了。

1.3 系统的分类

按系统的用途分类，可分为成批处理和实时处理。

在着重计算同人对话的语言处理系统中是以成批处理为主。而在控制机器或采集信息的系统中就必须进行实时处理。

在进行成批处理的系统中，重视CPU的运算与数据操作功能和存储器与存储器 之 间，存储器与磁盘等辅助设备之间的数据传送功能。家用计算机与办公室用计算机就是一个很好的例子。

在实时处理系统中，I/O接口及其中断功能很重要。在检测、控制方面经常 使 用I/O，特别是处理多重作业时，更必须加强多重中断功能。

关于软件，通用操作系统（OS）是CP/M，它主要用于成批处理。实时的OS与监控程序适用与实时处理。在16位系列中有RSX11，iRMX86，AMOS8000等。

1.4 Z80系列的设计思想

Z80汇集了CPU和外围设备（简称外设），称之为Z80系列。理由是，各设备被设计成在系统内能够进行紧密的协调动作（特别是中断处理）。具体如图1—1所示。从CPU输出的系统控制信号，除一、二个以外，均能与外设的输入端连接起来，并且外设本身能把中断矢量直接送给CPU（图1—2）。以前这项工作是由专门的中断控制装置来完成。

由于Z80系列具有这种紧密性，因此减少了附加电路，缩小了空间并且降低了成本。

Z80的特点如下。

（1）CPU的指令系统很强，因而减轻了程序员的负担。

（2）CPU的硬件加强了，因此外部附加的电路就减少了。

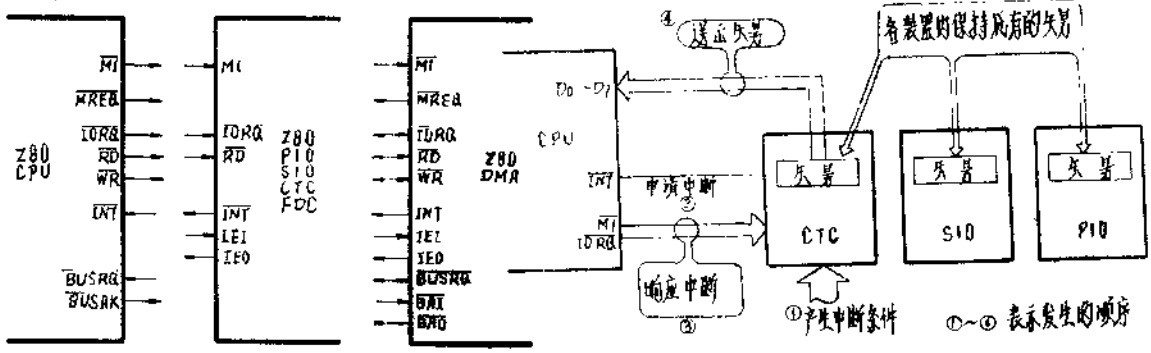
（3）外设的功能强，可以减少外部附加电路。

（4）外设是可编程的，使用起来很灵活。

由于以上的优点，大大减轻了开发软件和硬件的工作量，并且降低了安装成本。

从上述的系统分类来看，在成批处理的系统中，主要是有效利用CPU的数据操作和传送功能，而在实时处理的系统中则是发挥Z80系列整体的有效响应。

<图1-1> Z80系列装置的输入输出端

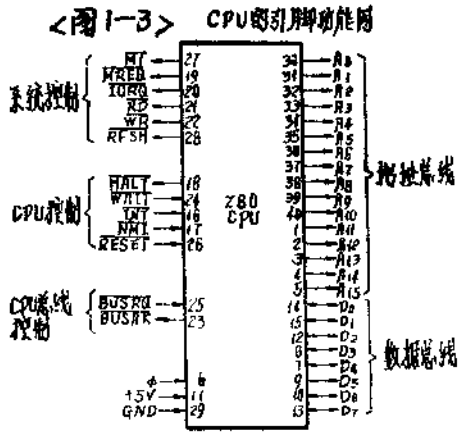


<图1-2> Z80外设保持、送出中断矢量

1.5 Z80 CPU的功能及特点

图1—3示出Z80CPU的引脚功能。下面以此为基础加以论述，这里把重点放在硬件上。

<图1-3> CPU的引脚功能图



CPU的主要作用是，从程序存储器读取指令代码，在CPU内译码，产生各种内部、外部用的控制信息（也包括数据）。靠这些控制信号对CPU内的数据进行运算，在数据存储器与I/O口之间进行数据传送。

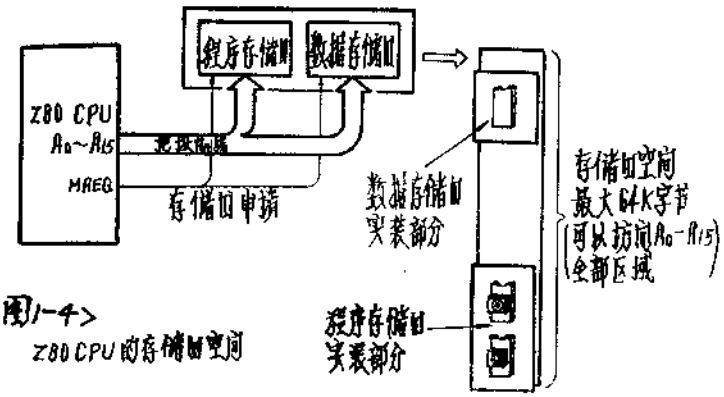
·存储器区

以前的存储器把程序区同数据区分开考虑，这样就比较容易理解，但许多通用的微型机却不是这样（图1—4）。因为这些微

型机使用相同的地址线A0～A15访问，在时间上虽然分开，但在空间上却是重叠的，为此，在使用时把区域分开，以便彼此不至互相重叠。

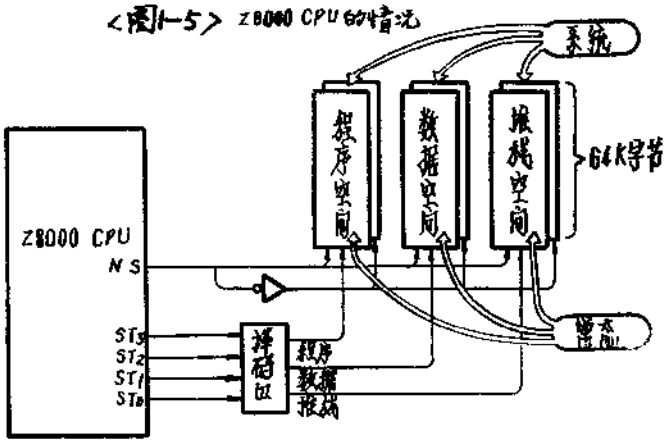
举一个例子来看，16位的Z8000CPU可以把二个区域（程序与数据）分开，如果发出一个控制信号也可以把堆栈分开（图1—5）。

Z80CPU有一个程序计数器（PC）作为程序指针，还有一个地址寄存器（AR）作为数据指针（图1—6）。但是此地址寄存器是假设的，实际上是由HL寄存器对和IX，IY变址寄存器或译码器输出承担着地址寄存器的任务。



<图1-4> Z80 CPU的存储回空间

<图1-5> Z8000 CPU的情况



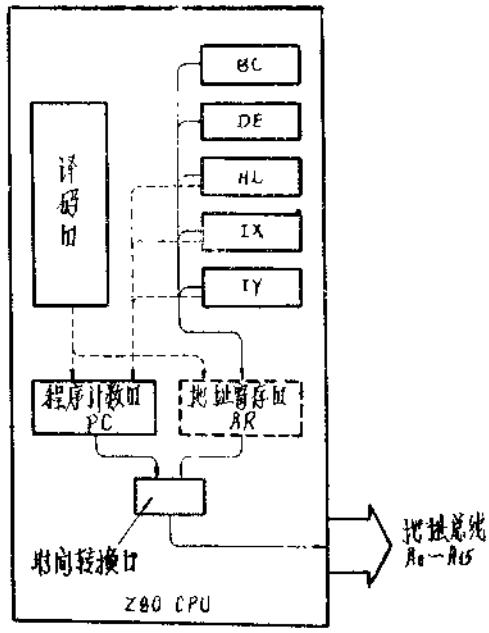
• 存储器的规模

Z80CPU的PC，AR都是16位的，它们的内容都是由地址总线A_0～A_{15}给出，可以访问（指定）64K的存储器。如果需要更多的存储器，那就将输出口分段使用，这样就能将存储器增加到64K以上。

实际上在组织系统时，存储器容量的大小是一个必须考虑的问题。笼统地回答这个问题比较困难。但在成批处理系统中，大多是解决计算处理和语言处理程序，因此一般都使用32K～64K字节。也有在主存储器（CPU能够直接访问的存储器）以外，增设辅助存储设备（磁盘和磁带）。

在控制系统中，如果是小规模系统，其程序存储器一般是2K～8K字节，大规模系统多使用20K～40K字节。数据存储器字节比较少，几百～几K字节，但在采集数据的系统中需要处理大量数据，因此一开始就要考虑使用外部辅助存储器。

<图1-6>
地址总线的构成



顺便说一下，辅助存储器是设置在普通I/O接口的外面，这是因为存取数据是通过I/O口，因此有人把微型机的外设也看作是系统的一个部分。

• I/O口

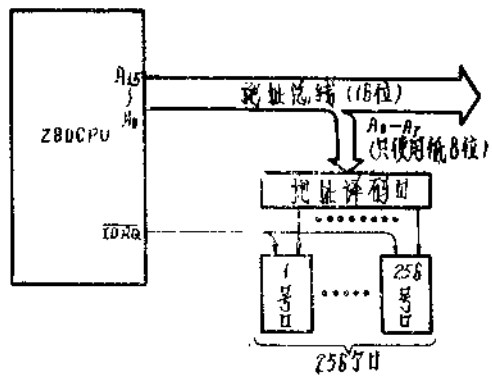
如图1—7所示，Z80CPU可以访问256个I/O口（Z80PIO是访问64个）。但是后面将要介绍的方式2中断，由于中断矢量的关系，其可能访问的数目最多为128个口。

从CPU来看，I/O同存储器同样都是传送数据。因为都是用同样的地址线进行访问，所以在处理上也是在64K之中设置程序区、数据区以及I/O区，这种方式称作存储器间隔I/O。这种方式把所有的数据传送指令都当作输入输出使用，这样做当然是方便了，但就得把64K的一部分空间用作I/O口。可是，I/O访问时间同存储器的访问时间又不一样，这样从外部存取数据的时间就必定要拖长。为了解决这个问题，就必须改变I/O和存储器的访问方式。

改变访问方式的办法是，把I/O当作另外一个空间（称作间隔I/O）。在Z80系统中，对时钟周期和中断响应信号的处理比较困难，因此不使用存储器间隔I/O这种方式是非常明智的。

Z80为了把地址线上重叠的存储器空间同I/O空间分开，采用从CPU发出$\overline{MREQ}$（存储器请求）和$\overline{IORQ}$（I/O请求）两个控制信号这样一种办法。

<图1-7>
I/O口



• Z80 I/O设备

Z80有二个I/O设备，一个是PIO（并行I/O），一个是SIO（串行I/O）

Z80PIO有二个8位并行的输入输出口，有四根联络线。各口不只是8位并行，而且每1位都能用程序设定是输入还是输出。每个口都有各自的中断矢量。

其他的系列装置虽然也是这样，但是Z80的外围系列装置却是以设计程序（软件）来代替电路接线（硬件）。

Z80 SIO是2个通道的串行输入输出口。此设备也是可编程方式。其特点是能够适用于从单纯的非同步（起止同步）方式到高度的IBM—SDLC方式的各种串行传送方式。

有人说SIO的程序设计比较困难，实际上使用过后就会感到非常方便。

最近，系统与系统之间的通讯越来越受到重视，因此使用SDLC等方式的也就多起来了。这些都是由IC组成的电路，由于LSI技术的进步，有时使用一个SIO就能把问题解决了。所以说，SIO是一个很有使用价值的装置。

• CPU的系统控制

Z80CPU有六根输出系统控制用的信号线，即MREQ，$\overline{IORQ}$，$\overline{RD}$，$\overline{WR}$，$\overline{M1}$以及$\overline{RFSH}$。其中的$\overline{MREQ}$（存储器请求）和$\overline{IORQ}$（I/O请求）是为把存储器和I/O分开而设置的信号，这在前面已经介绍过了。$\overline{RD}$（读），$\overline{WR}$（写）二个信号都是以CPU为主体的信号，$\overline{RD}$是CPU读进数据时发出的信号。

M1信号

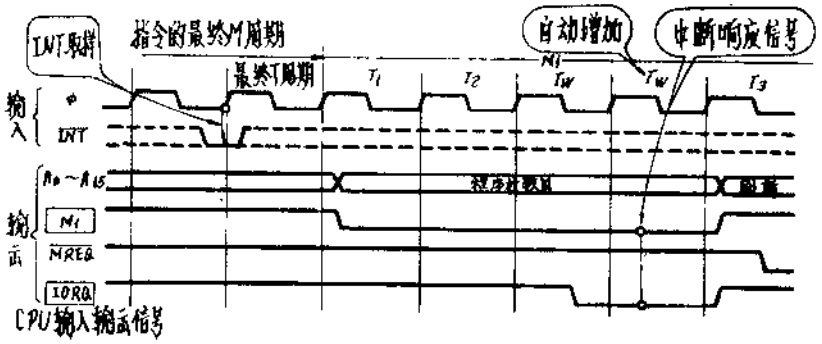
M1（机器周期1）是Z80系列所特有的信号，直接供给各外设使用。

M1虽然是在CPU取指令时发出的信号，但外设利用这个同步信号一边把中断矢量送给CPU，一边在监视中断服务程序的结束。

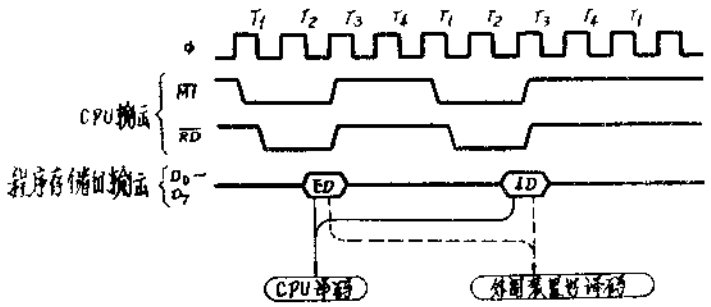
具体的例子如图1—8所示，对于外设的中断请求，CPU发出响应信号$\overline{M1}$和$\overline{IORQ}$（通常在$\overline{M1}$期间只发出$\overline{MREQ}$）。根据M1和$\overline{IORQ}$，外设把矢量送给CPU。同时，接受中断服务的外设在$\overline{M1}$期间监视数据总线上的指令代码，如果是“EDH”和“4DH”代码，就将其译码，中断程序到此结束。

EDH，4DH代码同RETI指令是相对应的（图1—9）。也就是说，即使不从CPU专门发出指令，只要提供$\overline{M1}$信号，外设也会自动解除中断。

<图1-8> 中断响应



<图1-9> RETI(中断返回)代码



刷新信号

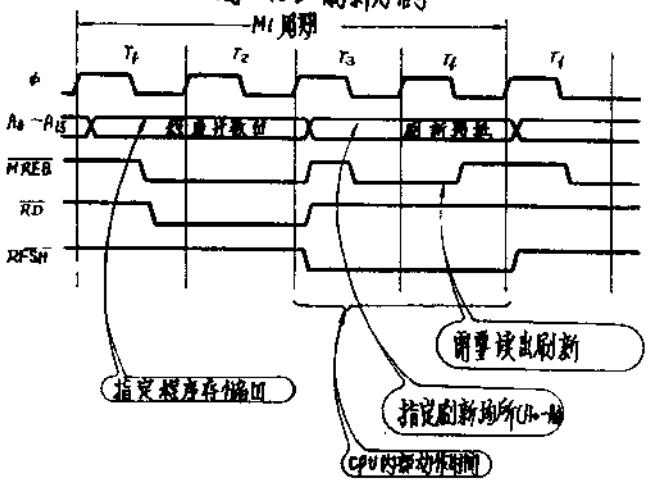
最近，DRAM（动态RAM）越来越向大容量化发展。不用说64K位的，甚至连256K位的现在都已经成为研究的课题。

Z80CPU访问的存储器空间如果使用的是64K位的动态RAM，最多是8个。为了方便地使用这样的动态RAM，Z80CPU增加了动态RAM刷新功能（图1—10）。

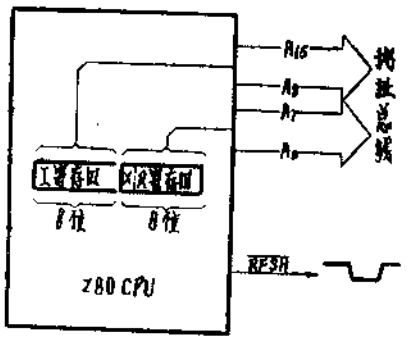
刷新装置并不复杂，就是在地址总线的低7位，顺序发出应当进行刷新的存储器的地址信号，但发出信号的时间必须准确。也就是说，CPU刚刚读进指令代码后，利用内部译码的一段空白时间进行刷新地址并发出刷新信号（$\overline{RFSH}$）。

刷新地址信号把CPU内的刷新计数器的内容送到地址总线A_0～A_6上，此计数器的内容每实行1条指令，就自动加1。而且送出刷新地址时，A_7是原来的值，I寄存器的内容送到高8位A_8～A_{15}上（图1—11）。

<图1-10> 刷新时序



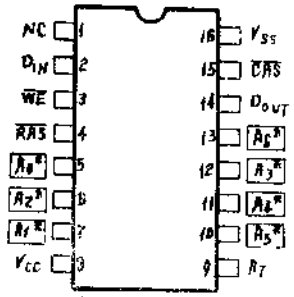
<图1-11> 刷新时的地址总线输出



X是初始值（普通设为"0"）

<图1-12>

64K DRAM刷新用地址



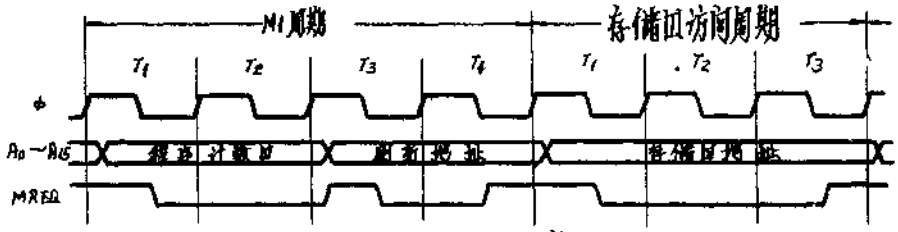
*是刷新地址

使用第7位的理由是，1个周期执行128条指令，以每条指令执行时间为8μs计算，1个周期就是1ms，因此，这在动态RAM的最长容许周期2ms之内是没有问题的。最近64K动态RAM也以第7位刷新为其标准方式了（图1—12）。

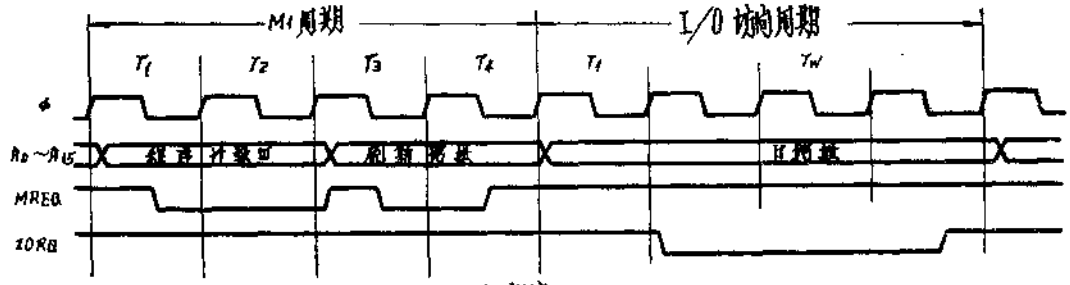
地址信号线的分时使用

CPU的地址输出在一个周期内分为如图1—13那样的（a），（b）两种情况。首先是程序区，其次是刷新区，最后指定和读出数据区或I/O区。

<图1-13> 地址线的分时使用



(a) 存储器指定



(b) I/O 指定

CPU中不设置区别程序区和数据区的信号。

CPU的控制输入

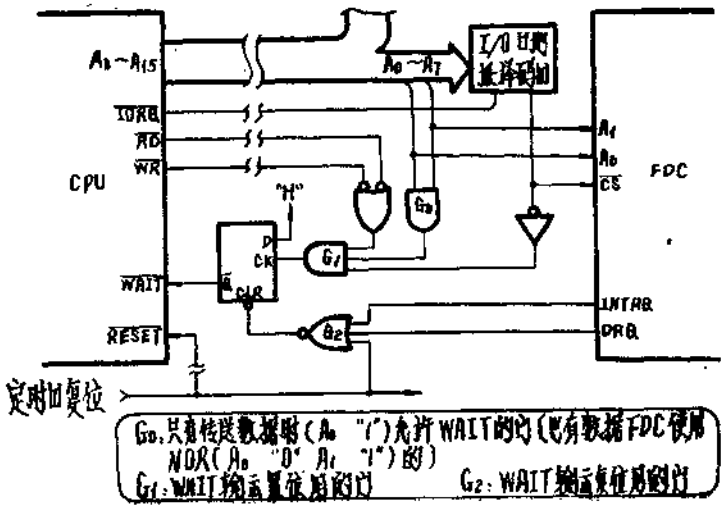
Z80CPU有4个控制输入信号，即：$\overline{\text{WAIT}}$，$\overline{\text{RESET}}$，$\overline{\text{INT}}$，$\overline{\text{NMI}}$。

当使用响应缓慢的存储器访问CPU时，必须考虑叫CPU稍似等待，因此设置等待输入。

以前有后备电池使用C—MOS存储器时，CPU要经常等待，但是最近都使用高速C—MOS存储器了，所以使用$\overline{\text{WAIT}}$的就少了。

使用倍密度软磁盘装置时，应很好的使用$\overline{\text{WAIT}}$（图1—14）倍密度软磁盘寻找每一

<图1-14>软磁盘控制口的等待电路



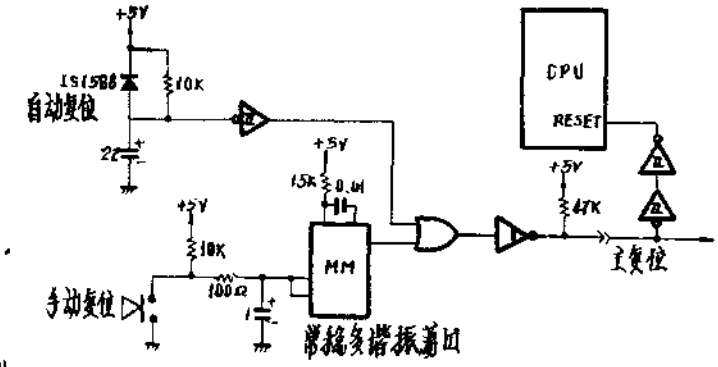
个字节并传送数据需要16μs，但这速度是接近极限速度（不只是数据传送。标志的检查与字节数的计算也是如此）。

在FDC中，当一个字节的数据备齐以前，使用$\overline{WAIT}$信号叫CPU处于等待状态。这种情况也有使用DMA的，但用DMA传送需要3—4μs（时钟2.5MHz时），给CPU留下做其他工作的时间就只有12～13μs（可以实行2～4指令）。

$\overline{RESET}$信号是把程序计数器（PC）的内容都置零，也就是说PC指向程序存储器的0地址。用这个信号可使中断允许标志复位，I,R寄存器为零，把中断方式置于方式0。而且数据寄存器无变化。

通常输入该端的信号，一个是接通电源时从复位电路自动发出的信号，如图1—15所示。另一个是手动复位信号。

<图1-15> 复位电路例



中断控制

一个系统在一定的时间内完成1～2项任务，在这种情况下，不使用中断而仅使用一种简单的方法是可以的（例如使用$\overline{NMI}$）。但是在进行多种操作的情况下，就必须使用中断方式。特别是在控制系统中，反复进行检测输入和控制输出，这时就必须使用中断。

并行处理必须分时进行，互相转换需要高速性。因此在分析中断因素和决定优先权方面就得借助于硬件。通常不是靠CPU用程序判别中断条件，而是采用调用外部处理程序，通过送出中断矢量转移到中断服务程序的方法。

为此目的，以前都是设置外部电路，但在Z80系列中是由外设来完成此项工作。另外，决定优先权也是采用与外设有关的优先链方式自动进行。

中断输入

Z80CPU有二个输入中断的信号$\overline{NMI}$（非屏蔽中断）和$\overline{INT}$（屏蔽中断）。

$\overline{NMI}$的中断优先权最高，它不能象$\overline{INT}$那样，而是直接从存储器的66H地址开始执行。

这种方法多半是在成批处理系统中，从操作台发出中断申请或在控制系统中出现异常的情况下使用。

$\overline{INT}$有三种中断方式，即：方式0，1、2。在由i8080方式外加中断控制电路构成的系统中是使用方式0，而在由Z80系列外设构成的系统中就不能使用方式0了。

如果中断因素简单，容易分析，也就是说容易识别中断是从哪个外设发出的，这种情况下多使用方式1。例如在小系统中使用定时中断时就经常使用方式1。

在Z80系列中，方式2是一种最有效的方式。后面还要详细介绍。

总线控制输入输出

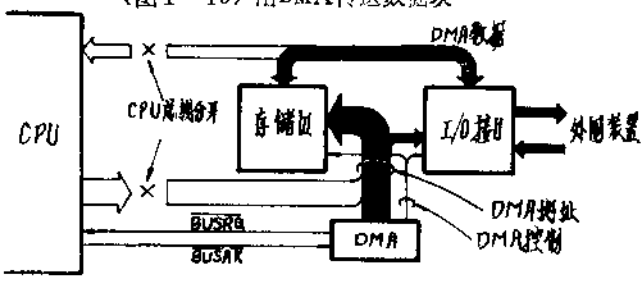
如果是普通的系统，其数据块的传送，使用数据块传送指令就可以了，但是，如果是处理图象数据和磁带、磁盘数据的系统就需要进行高速传送。

通常，程序传送数据都是通过CPU进行，因为每传送1个数据就要进入取指令周期。这样，如果数据块是连续传送，传输效用是不高的。因此就要考虑DMA（直接存储器访问）了。LSI化的DMA称作DMAC（DMA控制），在Z80中简称DMA。

如果DMA参加操作，就可以代替CPU的数据传送功能。这样，CPU就脱离地址总线和数据总线而让给DMA使用。

在Z80CPU上设立了两个信号来完成这项工作，一个是$\overline{BUSRQ}$，一个是$\overline{BUSAK}$（图1—16）。前者接受要求脱离总线的申请，后者发出这一申请。

〈图1—16〉由DMA传送数据块



除使用DMA外，CPU脱离总线的例子还不少，例如使用多CPU系统，它也是靠硬件弥补CPU的不足来提高系统的传输效率。

状态表示

在CPU的控制信号中，有一个叫作$\overline{HALT}$的输出信号，执行HALT指令将中止CPU工作（实际上是实行NOP指令），并从这个引脚发出信号。

例如，在成批处理系统中，停止人的工作时或在控制系统中等待中断时，都要使CPU停止工作，其状态用红灯表示，输出$\overline{HALT}$信号。

1.6 Z80系列中的中断结构

在设计Z80系列时，已经充分考虑到使其具备适应多重中断的功能。从前是在外部附加中断控制器，现在只增设发出中断的I/O设备就可以了。而且矢量也由程序来设定。这样，硬件不作太大变动就能改变系统的用途。

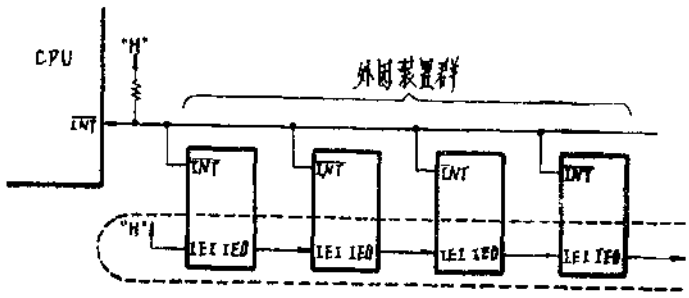
现在对中断结构详细说明如下。

优先链

根据中断进行多重处理，必须把各中断按先后顺序排队。当然也有的方式不用排顺序，但在处理上比较麻烦，因此把各中断排出先后顺序是最方便的。确定外设发出中断的顺序有好几种方式，Z80采用的是优先链方式，具体说，就是外设都有其自身的设定功能。

从Z80外设的引脚图上可以看出由IEI，IEO引脚构成的优先链（图1—17）。

<图1-17> 决定优先权顺序的优先级



把优先权最高的装置的IEI（输入）接上高电平（5V），把此装置的IEO接在后面优先权低的装置的IEI上。

这样，按IEO→IEI顺序连接下来，一直连接到最低位的IEO。

另一方面，$\overline{INT}$输出都是线“或”，同CPU的$\overline{INT}$输入连接。

工作起来，IEI为高电平的装置，任何时候均可申请中断，而申请中断的外设的IEO变成低电平。接下去后面的装置便不能申请中断。这样，低电平一直传递到最低位装置。这里存在着一个传递延迟的问题，后面还要详细谈这个问题。

发出中断申请的外设在得到CPU响应的时候送出自己的矢量，转向中断服务程序的处理过程，接受服务处理。

此时，该装置为了把服务处理状态记录下来，就在装置的内部设置一个中断服务锁存器。如果此锁存器在中途被优先权更高的中断所打断，此打断一经结束，该锁存器就又回到原来的状态（图1－18）。

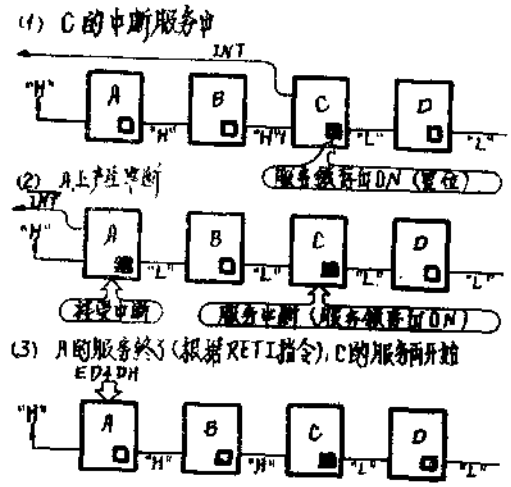
中断处理程序的最后必须设置一个“RETI”指令，当此代码“EDH，4DH”在$\overline{M1}$周期中送到数据总线时，外设就对此代码译码，根据此指令，CPU结束该中断服务程序。

外设进行译码后出现以下动作。

（a）接受服务的装置在读完“EDH，4DH”时，IEO从低电平上升到高电平，并且服务锁存器复位。

（b）没有接受服务（服务锁存器复位）的装置，在读“EDH”时，IEO必定是高电平（如果优先权更高的装置结束服务后，IEO从低电平上升到高电平）。

（c）虽然接受了服务，但中途被优先权更高的装置所打断，这时因为原来的装置的锁存器处于置位状态，IEO就仍为原来的低电平。



<图1-18> 中断服务的中断和重新开始

以上，根据优先顺序，服务—中断—再服务这样有条不紊地进行着。特别是，即使在超越优先顺序的情况，也还能顺利地返回原来的位置。这是Z80系列的一大特点。

优先链传送的延迟

上面介绍的Z80优先链的操作非常巧妙灵活，但唯一的缺点是传送时间的延迟，也就是说，从IEI到IEO在装置内的传送时间过长，大约需要130～190ns。如果是多段连接，优先权低的IEO从高电平返回到低电平的时间就更晚了。

如果相离较远的两个设备同时发出中断，在优先权高的中断控制信号（IEO输出）还没有传递到第二位置时CPU就会发出回答信号，这样就有可能双方都发出矢量。

为了避免发生这种情况，这里有二种方法。一种是使CPU推迟发出回答信号，一种是缩短传送时间。后一种方法如图1—19所示，把各装置的IEI和IEO的AND门输出到下一装置的IEI。这种方法简单方便，使用起来可以增加互相连接的段数。在这以前连接4段就到了极限。

链接的段数N必须满足以下公式。

$$t_{mr} - t_{SU} > N \times t_d$$

（这里，连接N段相同装置）

t_{mr}：从$\overline{M1}$上升沿到$\overline{IORQ}$下降沿的时间

$$t_{mr} = 2.5tc - 50ns$$

t_{SU}：在装置内的准备时间

$$t_{SU} = (t_{DM} - t_{DL}) + (ts - t_{DL})$$

t_d：器件的延迟

t_{DM}：从$\overline{M1}$下降沿到IEO下降沿的延迟时间

ts：从IEI下降沿的准备时间

t_{DL}：从IEI下降沿到IEO下降沿的延迟时间

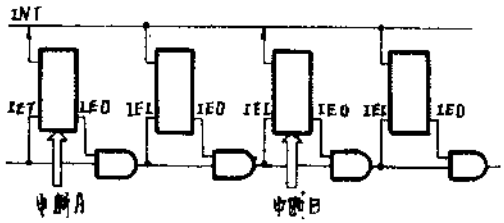
此公式的条件是，在中断回答期间即在$\overline{M1}$的下降沿到$\overline{IORQ}$的下降沿之间，设备必须完成准备工作。

这里的td按通常的连接是

$$t_d = t_{DL}$$

如果使用上述方法，应该把AND门的延迟时间包括在内。

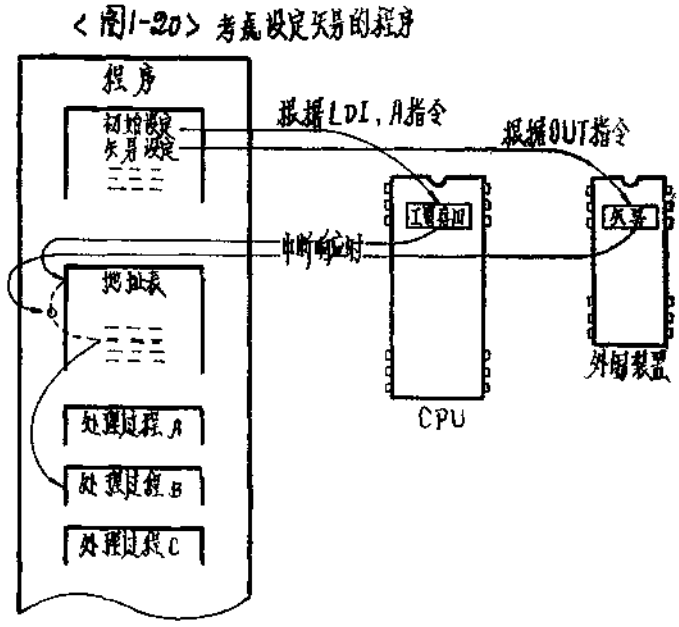
<图1-19> IEI-IEO传送延迟的改进方法（抑制同时发生中断）



中断方式2

最初有人说Z80的中断方式2比较难掌握，但事实并非如此。要点是，把所有的中断处理程序首地址排成一个表，向外设送入矢量，从表上找到所需地址，这些都搞清楚以后往下进行就简单了。

以下连同程序设计的顺序一起加以说明（图1－20）。



首先，利用DI指令禁止CPU响应中断（通常输入RESET后，CPU自动禁止中断，因此也可不用DI指令）。这样，外设也就不响应中断。

接着，对各外设的方式进行设定。方式的顺序根据设备而定，这点是要注意的。在方式的顺序中要设定产生中断的条件及设定中断矢量。该矢量是用OUT指令向外设写入。在Z80的PIO、CTC中，如果把矢量的最低位写为“0”，就认为它是矢量而保存起来。在SIO和DMA中，必须写入规定的“写寄存器”。

在中断允许之前，要把转移地址表的首地址的高8位放入I寄存器。

最后，如果允许各外设向CPU发出中断申请（CPU根据EI指令），外设的初始化即告结束。

以后，程序的情况虽有所不同，但开始时把各中断处理程序的首地址编排成表是必不可少的。

中断服务地址表应同程序一起固化在ROM中。

表上有128个转移地址。

在外设中予先设置的矢量同转移地址表中的一个地址是对应的。例如，假定地址表的首地址为0003H，矢量就是2×（3－1）＝4(0004H)。

程序例子如图1－21所示。此例使用1个PIO（2个口）和1个CTC（只有2通道）。

JTBL和MASK是选择转移表首地址的低8位的一种记忆方法。

图1－21　程序举例

```
;
;              TEST PROGRAM
;
  * * *                                  * * *
```

```
                ;
    0060        TIME    EQU     ËoH
                ;
                        ORG     oH
0000  3E4F      INIT:   LD      A,4FH           ; PIO MODE1
0002  D3D1              OUT     (001H),A        ; SET PIOA
0004  3E0F              LD      A,0FH           ; PIO MODE0
0006  D3D3              OUT     (0D3H),A        ; SET PIOB
0008  3E05              LD      A,05H           ; CTC TIMER
                                                    MODE
000A  D3D6              OUT     (0D6H),A        ; SET CTC CH 0
000C  3E60              LD      A,TIME
000E  D3D6              OUT     (0D6H),A        ; SET TIME CONST
0010  3E41              LD      A,41H           ; CTC CONT MODE
0012  D3D7              OUT     (0D7H),A        ; SET CTC CH 1
                ;
                ; (* * VECTOR SETTING * *)
                ;
0014  3E03      VSET:   LD      A,3             ; UP-BYTE OF JT-
                                                    BL ADRS
0016  ED47              LD      I,A
0018  3E04              LD      A,JTBL&MASK     ; VECTOR FOR PIOA
001A  D3D1              OUT     (0D1H),A        ;
001C  3E06              LD      A,JTBL&MASK+2; VECTOR FOR BIOB
001E  D3D3              OUT     (0D3H),A
0020  3E08              LD      A,JTBL&MASK+4; VECTOR FOR CTC
0022  D3D6              OUT     (0D6H),A
                ;       ...
                ;       ...
                ; * *   JUMP ADDRESS TABLE ***
                ;
                        ORG     304H
0304  0054      JTBL:   DEFW    SPIOA
0306  0454              DEFW    SPIOB
0308  0854              DEFW    SCTC0
030A  0C54              DEFW    SCTC1
                ;       ...
                ;       ...
                ; * * *  INTERRUPT SURVICE ROUTINES * * *
```

```
                ;
                        ORG     5400H
                ;
5400  DBD0    SPIOA:  IN      A,(0D0H)
                ;       ...
5402  ED4D              RETI
                ;
5404  DBDR    SPIOB:  IN      A,(0D2H)
                ;       ...
5406  ED4D              RETI
                ;
5408  3E20    SCTC0:  LD      A,20H
                ;       ...
540A  ED4D              RETI
                ;
540C  3E25    SCTC1:  LD      A,25H
                ;       ...
540E  ED4D              RETI
                ;
5410                    END
```

外设的访问与矢量的设定

关于Z80的外设，后面还要专门介绍，这里先概括地说明一下。

Z80 PIO

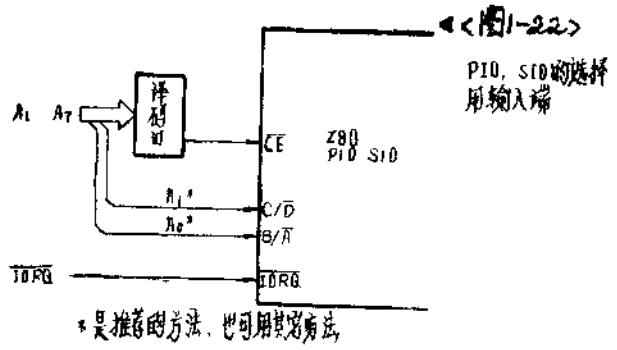
PIO有A，B2个口，根据B/A端选择A口和B口。在通常的数据传送以外，需要送入控制字时，要靠C／$\overline{D}$端（图1－22）。

例如，如果B/$\overline{A}$，C／$\overline{D}$是低电平，A口可以进行数据传送。如果B／$\overline{A}$是低电平，C／$\overline{D}$是高电平，则向A口写控制字。

B／$\overline{A}$，C/$\overline{D}$端一般连接地址线的A_0与A_1，高位地址线A_2～A_7经过译码接PIO的$\overline{CE}$端。

PIO的A口，B口各有数据，控制地址，所以PIO共有4个地址。

中断矢量虽然作为控制字写入，但为了使此矢量的最低位为零，就必须把前述的转移地址表的位置确定下来。



Z80CTC

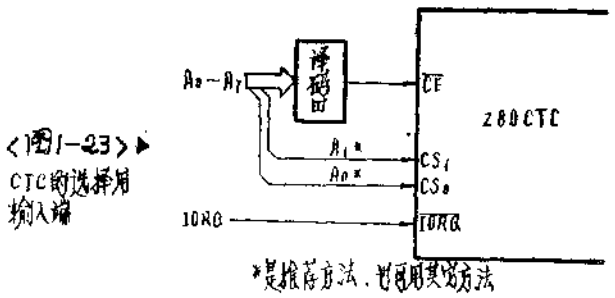
在CTC上有4条通道的计数器，定时器。为了对它们进行选择，设立了CS_0和CS_1输入（图1—23）。

CTC工作不同于PIO，它的工作是设定工作方式，向计数器或定时器写入时间常数，写入中断矢量以及读出计数器的内容。

工作方式的设定，最低位为“1”，此时的第2位也为“1”，而下面的8位控制字即为时间常数。

中断矢量只写入0通道，其他通道会自动写入。

CTC必须使用4通道8字节的转移地址时，0的通道矢量必须是0或8的倍数。



<图1-23>
CTC的选择用输入端

Z80 SIO

SIO有A，B两个通道，同PIO一样也是根据B/$\overline{A}$，C/$\overline{D}$输入进行口的选择和对数据，控制的选择。

条件设定等方法在以后章节里还要详述，这里要说明的是，在SIO内有几个“写寄存器（WR）”，就是根据这些写寄存器的写入来进行各种设定。

中断矢量的写入，虽然只向WR_2写入1个，但要根据各种状态修改后返回CPU。

如果禁止修正，矢量就以原样返回；如果允许修正则A通道有4种，B通道有4种，总共8种状态。

把写入矢量规定为偶数，其高4位为X，就会从通道B的X0H，X2H，X4H，X6H中的一个返回或从通道A的X8H，XAH，XCH，XEH中的一个返回。因此使用SIO的情况，在转移地址表上不管使用不使用从X0H开始的地址号，都应保证16个字节。

若中断矢量是奇数，最好还是把它变成偶数（就是最低位是“0”）。

Z80 DMA

DMA有一个口地址，矢量修正等也同SIO一样（图1－24）。但是DMA的矢量设定虽然是针对“写寄存器4（WR_4）”的第6段，但不能省略中途各段的设定，因此变动时就显得复杂一些。

中断矢量一写入偶数个1（奇数也行，但最好尽量避免），在禁止修正的场合就是1；允许修正的场合用以下4种返回（X0H，X2H，X4H，X6H）。

矢量的修正

CTC是把各通道设定的矢量送回CPU，而SIO和DMA是在申请中断的时候，根据产生中断的原因把最初设定的矢量送回CPU。实际上这也是Z80中断方式2的一大特点。