第5章　中央处理器

知识目标

◎了解CPU各个组成部分及其功能。

◎理解指令周期的概念、时序的产生及其功能、CPU的控制方式。

◎理解微程序及其相关的概念。

◎了解流水线、CPU多核等一些典型的CPU技术。

技能目标

◎能够理解指令执行的具体过程，并能画出指令具体执行的流程图。

冯·诺依曼体系结构计算机的五大部件中，运算器和控制器是信息处理的中心部件，所以它们合称为“中央处理单元”(central processing unit，CPU)。下面主要对CPU的组成、指令周期、微程序控制器以及CPU的新技术进行介绍。

5.1　CPU概述

在计算机系统中，CPU由控制器和运算器两大部分组成。控制器是整个系统的指挥中心，在控制器的控制下，运算器、存储器和输入/输出设备等部件构成一个有机整体，完成指令要求的工作。

对于冯·诺依曼结构的计算机而言，一旦将程序写入主存储器后，就可由计算机自动完成取指令和执行指令的任务。控制器就是专门完成此项工作的，它负责协调并控制计算机各部件执行程序的指令序列，其基本功能是取指令、分析指令和执行指令。

1.取指令

控制器必须具备能自动地从存储器中取出指令的功能。为此，要求控制器能自动形成指令的地址，并能发出取指令的命令，将对应此地址的指令取到控制器中。第一条指令的地址可以人为指定，也可由系统设定。

2.分析指令

分析指令包括两部分内容：其一，分析此指令要完成什么操作，即控制器发出什么操作命令；其二，分析参与这次操作的操作数地址，即分析操作数的有效地址。

3.执行指令

执行指令就是根据分析指令产生的操作命令和操作数地址的要求，形成操作控制信号

序列(不同的指令有不同的操作控制信号序列),通过操作运算器、存储器以及 I/O 设备,执行每条指令。

此外,控制器还必须能控制程序的输入和运算结果的输出(控制主机与 I/O 设备交换信息)以及对总线的管理,甚至能处理机器运行过程中出现的异常情况(如断电)和特殊请求(如打印机请求打印一行字符)。

总之,CPU 必须具有控制程序的顺序执行(指令控制)、产生完成每条指令所需的控制命令(操作控制)、对各种操作加以时间上的控制(时间控制)、对数据进行算术运算和逻辑运算(数据加工)以及处理中断、DMA 等事务的功能。

5.2　CPU 的组成

CPU 是数字计算机的主要设备之一,其主要功能是解释计算机指令以及处理计算机软件中的数据。计算机的可编程性主要是指对 CPU 的编程。CPU 是计算机中的核心部件,是一台计算机的运算核心和控制核心。计算机中所有操作都由 CPU 通过读取指令,对指令译码并执行实现的。

5.2.1　构成 CPU 的主要部件

根据 CPU 的功能,不难设想,要取指令,必须有一个寄存器专用于存放当前指令的地址;要分析指令,必须有存放当前指令的寄存器(IR);要执行指令,必须有一个能发出各种操作命令序列的控制命令产生部件(CU);要完成算术运算和逻辑运算,必须有存放操作数的寄存器和实现算术/逻辑运算的部件算术逻辑单元(ALU);为了处理异常情况和特殊请求,还必须有中断系统。CPU 的结构如图 5-1 所示。图 5-1 中包含了由 DR、IR 寄存器构成的指令通路和指令暂存器,由微指令或硬布线构成的控制命令产生部件,算术逻辑运算单元(ALU)和直接接入 IR 的中断和直接数据通道信号 INT、DMA。

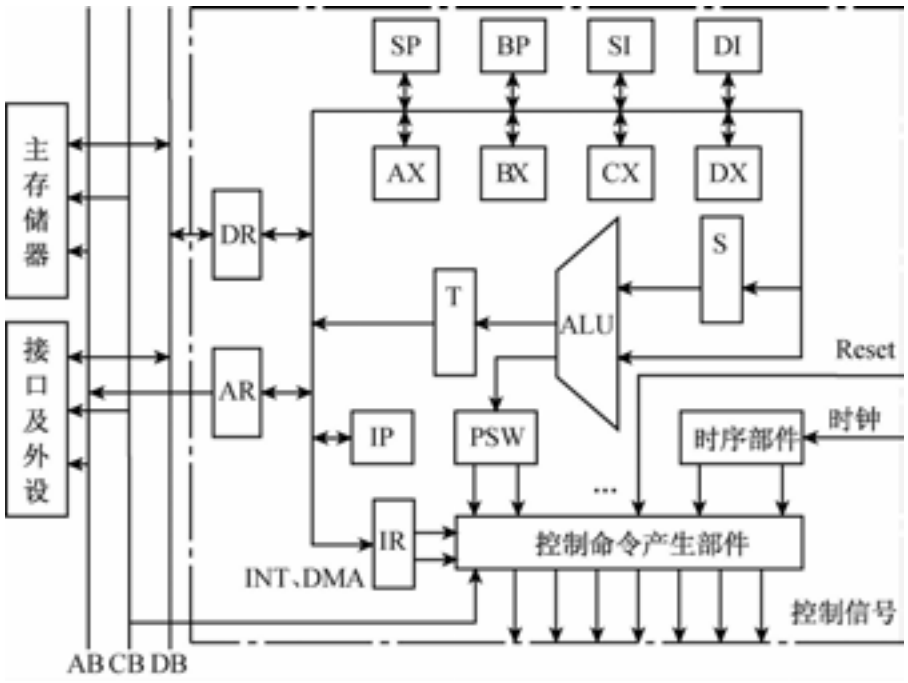

图 5-1　使用系统总线的 CPU 结构图

5.2.2 CPU 中的寄存器

寄存器是 CPU 内部重要的数据存储资源，是汇编程序员能直接使用的硬件资源之一。寄存器一般用来保存程序的中间结果，为随后的指令快速提供操作数，从而避免把中间结果存入内存，再读取内存的操作。

1.寄存器基本电路

寄存器基本组成及控制电路如图 5-2 所示。由 3 片 8 位字长的 74LS374 组成寄存器组。3 个寄存器输入接口的 8 位连至数据总线(DB)输入接口，而 3 个寄存器输出接口的 8 位连至 DB 输出接口。图 5-2 中，R0、R1、R2 经二进制控制开关译码产生数据输出选通信号，在同一时刻只能有一个选中工作，LDR0、LDR1、LDR2 为数据写入允许信号，控制 74LS374 的写入和读出，均为高电平有效；T4 信号为寄存器数据写入脉冲，上升沿有效。

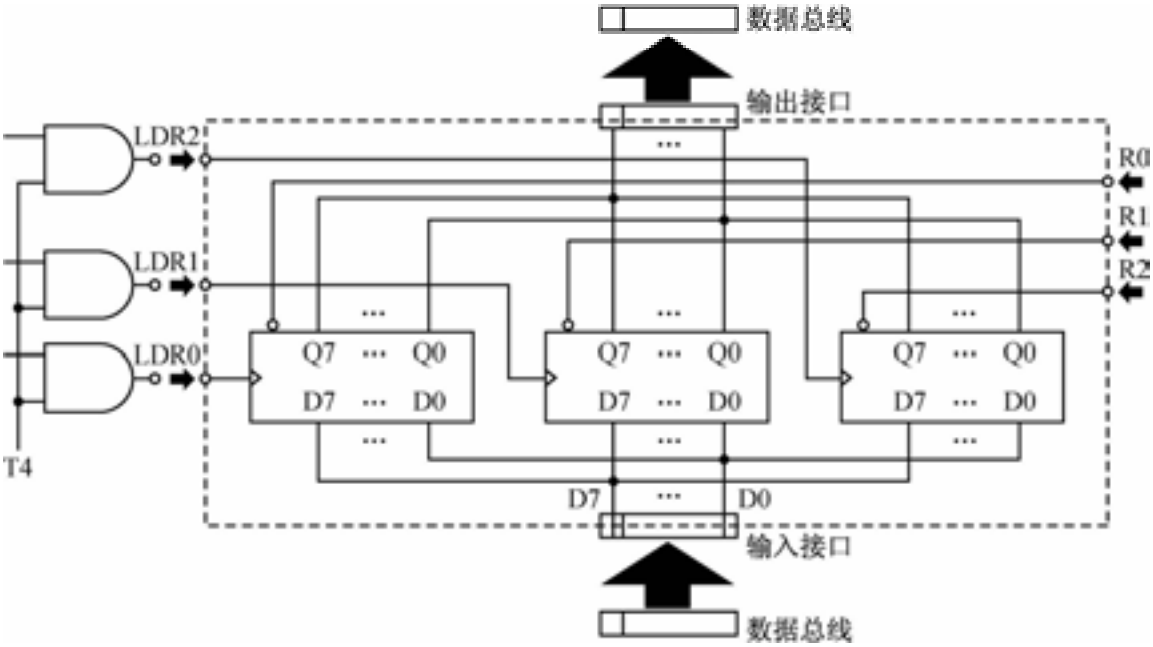

图 5-2 寄存器基本组成及控制电路

在真实 CPU 中，寄存器组包含多个寄存器，可为 8 位、16 位、32 位等，此处给出的电路图仅说明底层硬件的工作原理。

2.CPU 中用户可操作的寄存器

通常 CPU 执行机器语言访问的寄存器为用户可操作的寄存器，按其特征又可分为以下几类。

1)通用寄存器

通用寄存器可由程序设计者指定许多功能，既可用于存放操作数，也可作为满足某种寻址方式所需的寄存器。通用寄存器用于存放操作数，其位数应满足多数数据类型的数值范围，有些机器允许使用两个连续的寄存器存放双倍字长的值。通用寄存器可代替基址寻址所需的基址寄存器、变址寻址所需的变址寄存器和堆栈寻址所需的栈指针，也可作为计数器使用。寄存器间接寻址时还可用通用寄存器存放有效地址的地址。

2)地址指针寄存器

将专用寄存器作为基址寄存器、变址寄存器或栈指针，在设计指令格式时，只需将这类专用寄存器隐含在操作码中，而不必占用指令字中的位。

3)段地址寄存器

为了扩大计算机寻址范围，在一些 16 位机器中设置了段寄存器，它们与地址指针寄存

器合成物理地址，使寻址空间大于 16 位，如 8086/8088 系统中扩大至 20 位地址。可寻址范围为 $2^{20}=1$ MB。当然，若为 32 位机器，地址总线为 32 位，可满足最大的地址范围为 $2^{32}=4$ GB，则不再需要段地址寄存器。

4）条件码寄存器

条件码寄存器中存放条件码，它们对用户来说是部分透明的。条件码是 CPU 根据运算结果由硬件设置的位，例如，算术运算会产生正、负、零或溢出等结果。条件码可被测试，作为分支运算的依据。此外，有些条件码也可被设置，例如，对于最高位进位标志 CF，可用指令对它置位和复位。将多个条件码放到一个寄存器中，就构成了条件码寄存器。

4 种寄存器的内部结构如图 5-3 所示。

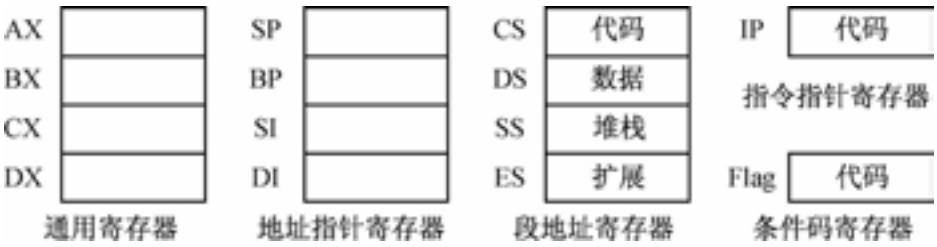

图 5-3　CPU 中的寄存器

计算机在中断或调用子程序前，必须将有可能在子程序中用到的寄存器的内容保存起来。这种保存由中断处理过程或指令交由 CPU 依据预定相关寄存器值自动完成，也可由程序员编程保存，视不同情况进行不同处理。

3. 控制和状态寄存器

CPU 中还有一类寄存器用于控制 CPU 的操作或运算。在一些机器中，大部分这类寄存器对用户是透明的。以下 4 种寄存器在指令执行过程中发挥着重要的作用。

(1) AR：存储器地址寄存器，用于存放将被访问的存储单元的地址。

(2) DR：存储器数据寄存器，用于存放欲存入存储器中的数据或最近从存储器中读出的数据。

(3) PC(有些机器中将 PC 描述成 IP)：程序计数器，存放现行指令的地址，通常具有计数功能。当遇到转移类指令时，PC 的值可被修改。

(4) IR：指令寄存器，存放当前欲执行的指令。

通过这 4 种寄存器，CPU 和主存可交换信息。例如，将现行指令地址从 IP 送至 MAR，启动存储器进行读操作，存储器就可将指定地址单元内的指令读至 MDR，再由 MDR 送至 IR。

在 CPU 内部必须给 ALU 提供数据，因此 ALU 必须可直接访问 MDR 和用户可见寄存器，ALU 的外围还可以有另一些寄存器，这些寄存器用于 ALU 的输入/输出以及用于和 MDR 及用户可见寄存器交换数据。

在 CPU 的控制和状态寄存器中，还有些用来存放程序状态字 PSW 的寄存器，该寄存器用来存放条件码和其他状态信息。在具有中断系统的机器中还有中断标记寄存器。

5.2.3　CPU 中的运算器

运算器主要完成对二进制数据的定点算术运算、逻辑运算以及移位操作。在某些 CPU 中还有专门用于处理移位操作的移位器。

1. 运算器的功能

运算器的主要功能是对数据进行加工和处理。它是在控制器的控制下工作的，是一个

加工处理部件。其主要功能如下：

(1)对数据进行加工处理，主要包括对数值数据的算术运算，如加、减、乘、除运算，变更数据的符号等。

(2)对各种数据的逻辑运算。

(3)它是传递数据的一条重要途径。

2.运算器的基本结构

运算器的主要部件如下：

(1)存放待加工的信息或加工后的结果信息的通用寄存器组。

(2)按操作要求控制数据输入的部件：多路开关或数据锁存器。它可以接收来自外部设备或存储器中的数据，也可以是暂存通用寄存器中的数据。

(3)按操作要求控制数据输出的部件：输出移位和多路开关，它们可以将 ALU 的输出，根据要求进行左移、右移、直送、半字交换，并从中选择之一进行输出，经总线送往其他部件，或作为中间结果送给通用寄存器，再次作为 ALU 的输入，进行下次运算。

(4)计算器与其他部件进行信息传送的总线以及总线接收器与发送器，总线接收器与发送器通常是由三态门构成的。

3.电路实现

1)运算实现部件

运算器由以下部分构成：由两片 74LS181 构成的并/串型 8 位 ALU；两个 8 位寄存器 DR1 和 DR2 作为暂存工作寄存器，用于保存参数和中间运算结果；ALU 的输出由三态 74LS245 连接到数据总线上。运算器的电路图如图 5-4 所示。

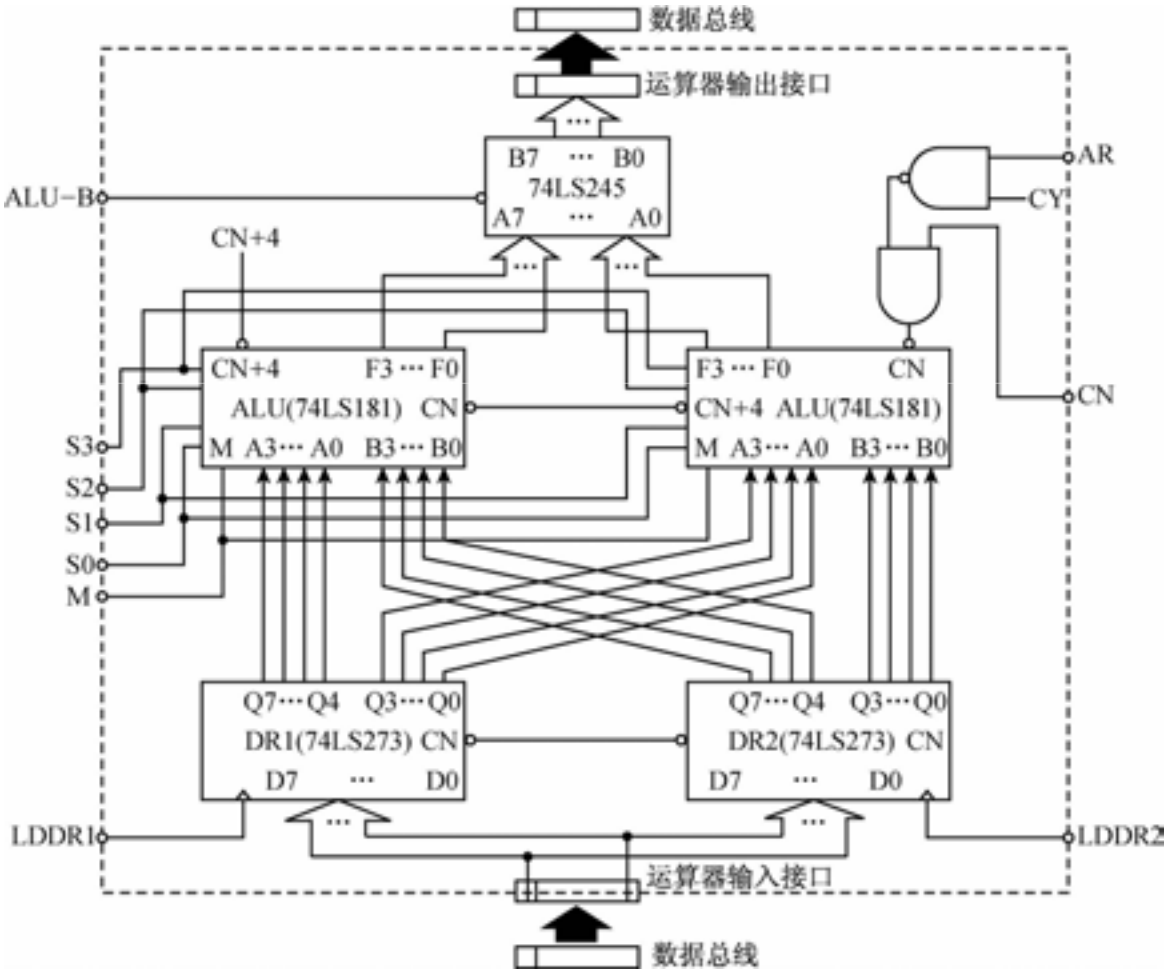

图 5-4 运算器电路图

电路中两片 74LS181 由 S0、S1、S2、S3 和 M 信号构成可完成 32 种算术运算和逻辑运算的电路，S0、S1、S2、S3 和 M 信号来自于控制命令产生部件。两片 74LS273 芯片构成了数据输入暂存器，74LS245 双向总线收发器构成了 ALU 输出门控。

工作时，分时给出 LDDR1 和 LDDR2 信号（来自控制命令产生部件），装入 DR1 数据和 DR2 数据，给出 S0、S1、S2、S3 和 M 信号完成规定运算，给出 ALU-B 信号，将结果送数据总线。

2）移位电路

如图 5-5 所示，一片 74LS299 作为移位寄存器，其中 8 位输入/输出端线接口连接至数据总线，进行移位操作时先装入数据，移位后再输出到数据总线。电路中 299-B 信号控制其使能端（0 有效），T4 为时序节拍脉冲，由 S0、S1、M 控制信号设置寄存器的运行状态，其控制功能如表 5-1 所示。

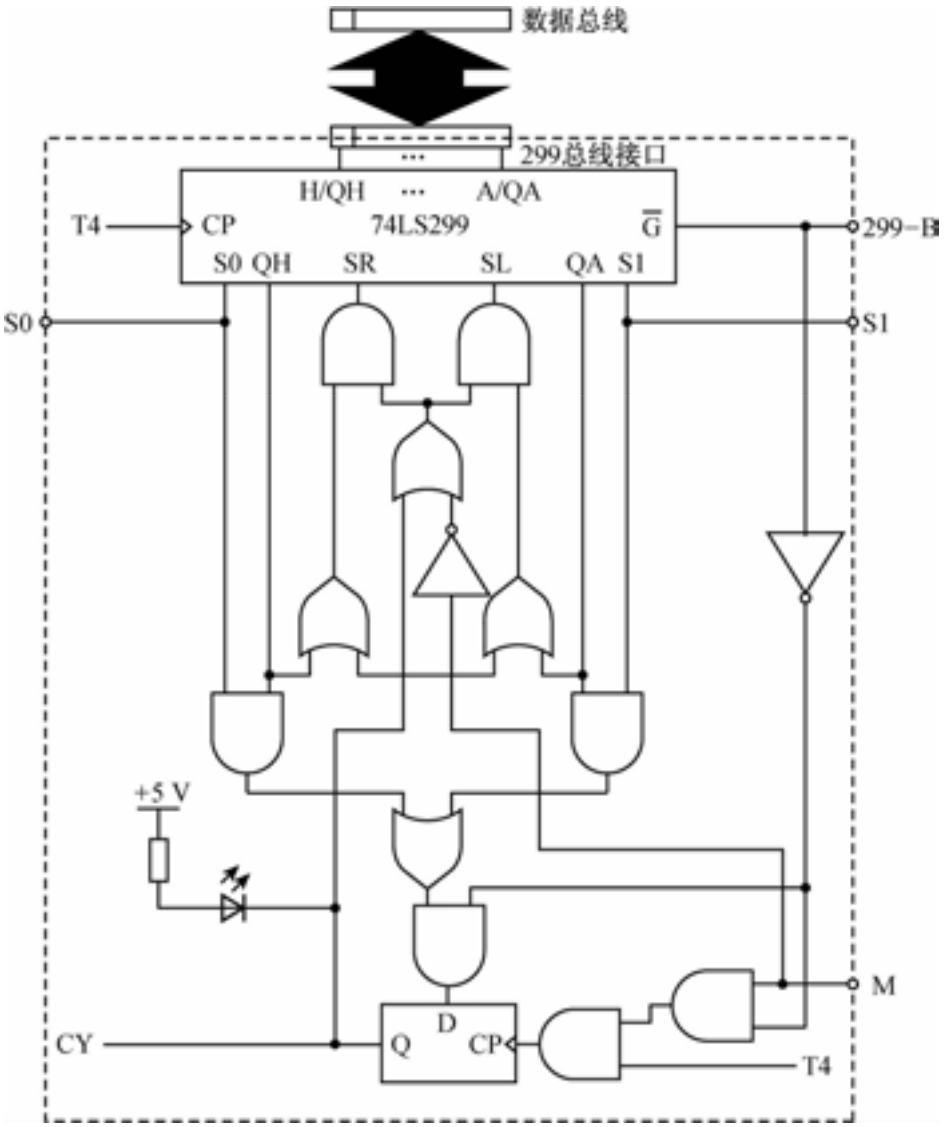

图 5-5　移位寄存器电路图

表 5-1　移位寄存器控制功能表

299-B	S1	S0	M	功　能
0	0	0	任意	保持
0	1	0	0	循环右移

续表

299-B	S1	S0	M	功　能
0	1	0	1	带进位循环右移
0	0	1	0	循环左移
0	0	1	1	带进位循环左移
任意	1	1	任意	装数

4.数据通路

上述部件都是通过数据总线进行数据交换的，在模型 CPU 内部只有一条数据总线进行数据传送。在真实 CPU 中必须要考虑数据交换通路，现实 CPU 中的架构设计就是以数据通路的传送速率、传送效率为设计依据的。

1)数据通路的概念

数据通路是指数据在 CPU 各功能部件之间传送的路径。这里的数据是广义的。例如，某通用寄存器的数据经内部总线加到 ALU 上，是将总线作为数据通路。同样，IP 送出的地址信号也通过总线传送到地址寄存器 AR 上。而 CPU 从内存读取的指令码也会通过数据通路加到指令寄存器 IR 上。

数据通路描述了信息从什么地方发出，中间经过什么部件，最后传送到哪个部件。信息每传送一步都需要一定的控制信号(或控制命令)才能做到。执行指令的控制信号是由控制命令产生部件产生，用于建立执行指令的数据通路。

数据通路的功能是实现 CPU 内部运算器、寄存器、控制器等各功能部件间的数据传送。

2)数据通路的基本结构

数据通路的基本结构主要有如下两种形式：

(1)总线结构。连接两个以上数字器件的信息通路称为总线。由图 5-1 可以看到，在 CPU 内部采用单总线结构，CPU 的各部件均连接在此内部总线上。同时，由于 CPU 是 16 位的，其内部总线的宽度也是 16 位的，总线上连接的各寄存器也是 16 位的。此外，为简单起见，规定主存按 16 位编址，每一主存地址存放 16 位二进制数。

在这种单总线结构中，所有寄存器、ALU、暂存器等部件的输入和输出端均连接在这条单一的 CPU 内部总线上，它们共享一条内部总线。这就产生一个非常重要的问题：任何时刻只允许连接在此总线上的一个部件输出其数据到总线上，其他剩余的部件绝不允许有输出(可以输入)。因为，只要有两个或两个以上的部件将它们的数据同时输出到此总线上，必然使总线产生竞争(也称为争用或冲突)。一旦产生竞争就一定会出错。因此，在单总线结构的 CPU 中，避免总线竞争是应该特别注意的。

在模型计算机中，数据传送是通过单总线分时传送方式进行的，部件分别在不同时间段执行指令，避免了总线冲突。因此执行指令时，必须按照指令功能安排好每一步的操作，结合时钟分步产生控制信号送到硬件的控制端进行相应的操作。

为了防止总线竞争，在 CPU 执行指令的过程中，连接在单总线上的部件在向总线输出数据时必须分时操作，即一个部件输出结束置为高阻状态后，另一部件才能向总线输出数据。当部件没有输出时，其输出端必须置高阻状态。

单总线结构易发生总线竞争，为防止竞争就需要分时操作，分时操作必定降低传输数据的性能。为此，可采用双总线或三总线结构。在双总线及三总线结构中，总线竞争的机会要

小得多，总线的传输性能可得到提高。显然，在总线构成上也会更加复杂。

(2)专用数据通路。在这种数据通路下，CPU 内部各功能部件之间不采用总线相连接，而是设置专用的数据通路，让各部件的数据、地址等信号走自己专用的连接线。这样就可以避免共享总线容易产生的总线竞争，可以更好地提高 CPU 的性能。

构成专用数据通路的工作很复杂，工作量很大，要动用更多的硬件资源。

5.3　指 令 周 期

CPU 从内存取出一条指令并执行完这条指令的时间总和称为指令周期。对于 CISC，指令周期一般由若干机器周期组成，是从取指令、分析指令到执行完指令所需的全部时间。指令不同，所需的机器周期数也不同。对于一些简单的单字节指令，在取指令周期中，指令取出到指令寄存器后，立即译码执行，不再需要其他的机器周期。对于一些比较复杂的指令，如转移指令、乘法指令，则需要两个或者两个以上的机器周期。从指令的执行速度看，单字节和双字节指令一般为单机器周期和双机器周期，三字节指令一般都是双机器周期，只有乘、除指令占用 4 个机器周期。在编程时要注意选用具有同样功能而机器周期数少的指令。

5.3.1　指令周期的概念

计算机之所以能自动地工作，是因为 CPU 能从存放程序的内存中取出一条指令并执行这条指令；紧接着又去取指令，执行指令……如此周而复始，构成了一个封闭的循环。除非遇到停机指令，否则这个循环将一直进行下去。

CPU 每取出并执行一条指令，都要完成一系列的操作，这一系列操作所需的时间通常称为一个指令周期。简单地说，指令周期是取出并执行一条指令的时间。由于各种指令的操作功能不同，有的简单，有的复杂，所以各种指令的指令周期是不相同的。

指令周期常常用若干机器周期数来表示，机器周期也称 CPU 周期，即完成一个基本操作为一个机器周期。例如，取指令、访问存储器、访问 I/O 接口、一次中断处理过程等都可以看成是一个基本机器操作。由于 CPU 内部的操作速度较快，而 CPU 访问一次内存所花的时间较长，因此通常用从内存中读取一个指令字的最短时间来规定机器周期。这就是说，一条指令的取出阶段(通常称为取指)需要一个机器周期时间。而一个机器周期时间又包含若干时钟周期(通常称为节拍脉冲或 T 周期，是处理操作的最基本单位)。这些时钟周期的总和规定了一个机器周期的时间宽度。一般情况下，任何一条指令的指令周期至少需要一个机器周期，而复杂一些的指令周期，则需要更多的机器周期。但必须指出，对机器周期的规定在各种计算机中不尽相同。

当计算机加电时，半导体存储器 RAM 以及各寄存器将处于随机状态。为了保证机器的正常工作，在计算机中一般都设置有存放初始化程序的只读存储器(ROM)，加电时由硬件产生的一个复位信号(RESET)使计算机处于初始状态，并从初始化程序入口开始执行。这是通过将初始化程序的入口(即程序第一条指令地址)装入程序计数器而实现的，也可以直接在指令寄存器 IR 中置入一条无条件转移指令，转到初始化程序入口处执行。初始化程序完成的任务随机器设计的不同而不同。

1.基本工作过程

计算机的基本工作过程主要是指执行指令的过程。

1)取指令过程

取指令阶段是将现行指令从主存储器中取到指令寄存器 IR 的过程,可分为下列几个步骤:

(1)将 IP 中的内容发送至存储器的地址寄存器 AR,并对存储器发出控制命令"读"(READ)。

(2)利用指令尚未读出的这段时间,控制器将 IP 的内容递增,为取下一条指令做好准备。

(3)在存储器中进行读操作,即将 AR 所指定的地址单元的内容读出,送到数据寄存器 DR 中,并发出信号通知控制器,表示存储器的读操作功能已完成。

(4)把存放在 DR 中的指令代码送到指令寄存器 IR 中,随后,指令译码器对操作码字段 OC 开始译码。

2)分析指令过程

指令译码器会识别和区分不同的指令类别及各种获取操作数的方法。由于各指令功能不同,寻找操作数的操作复杂程度也不一样。

对于无操作数指令的分析较简单,只要译码器识别出指令即可进入执行指令阶段。而对于带操作数指令的分析就需要读取操作数,因此,要先计算出操作数的有效地址。若操作数存放在通用寄存器中,则不需要再访问主存;如果操作数放在主存中,则要到主存中去取数。

对于不同的寻址方式,有效地址的计算方法也不同,有时可能要多次访问主存才能取出操作数来。

例如,假设目前在 IR 中的指令如下:

ADD(R0),R1

其中,R0 和 R1 是通用寄存器,事先已经由其他指令送入了内容。

分析指令阶段能得到两个结果:

(1)这是一条加法指令。

(2)源操作数是寄存器间接寻址方式,操作数在内存中,有效地址是(R0),目标操作数是寄存器直接寻址方式,操作数就是 R1 寄存器的内容。

又例如,若目前在 IR 中的指令如下:

SUB D(R0),(R1)

其中,R0、R1 是通用寄存器,事先已经由其他指令送入了内容。

分析指令阶段能得到两个结果:

(1)这是一条减法指令。

(2)源操作数是寄存器变址寻址方式,操作数在内存中,有效地址是(R0)+D,目的操作数是通用寄存器间接寻址方式,有效地址是 R1 寄存器的内容。

3)执行指令过程

执行指令阶段的任务是完成指令所规定的各种操作,具体实现指令的功能。为了执行一条指令,需要有各种控制信号在不同时刻来执行各个最基本的控制操作。单靠一个操作是不可能实现指令所要求的功能的,需将多个控制操作编排成秩序井然的序列。

例如,上述加法指令"ADD (R0),R1",经操作译码得到的这条加法指令的控制电位被送至微操作控制器,在时序部件配合下,产生一组按时序节拍发生的微操作控制信号,一步步地执行指令所要求的工作,直到指令的功能全部完成。

按加法指令“ADD (R0),R1”的要求,控制器发送相应的控制信号并执行下列动作:

从 R0 取出内容,送入 AR 中,发送读内存命令,取得源操作数并传输至 DR,再将 DR 的内容通过总线传输至运算器 ALU 的一端;从 R1 寄存器中取得内容,传输至 ALU 的另一端;在 ALU 中进行加法运算;最后把结果传输至 R1 寄存器中。

对于减法指令“SUB D(R0),(R1)”,在执行指令阶段控制器发送下列控制信号:

将形式地址 D 从内存单元(本条指令的第二个指令字即是 D)中取出,和 R0 的内容相加,形成有效地址,送入 AR 寄存器中,发送读内存命令,取得源操作数并传输至 DR,再将 DR 的内容通过总线传输至运算器 ALU 的一端,然后将源操作数保存在寄存器组中的源操作数暂存器中;从 R1 寄存器中取得内容,取得目的操作数有效地址,送入 AR 寄存器中,发送读内存命令,取得目的操作数并传输至 DR,再将 DR 的内容通过总线传输至运算器 ALU 的一端;再将源操作数从源操作数暂存器中读出,传输至 ALU 的另一端;在 ALU 中进行减法运算;最后把结果送内存。内存的地址即是目的操作数所在位置,由于在送结果之前对内存的操作是取目的操作数,故 AR 中已经是存放结果的内存地址了,无须再次计算,直接使用即可。

若无意外事件(如结果溢出)发生,机器就接着从 IP 中取得下一条指令地址,开始一条新指令的控制过程。

2.模型机结构

为讨论方便起见,这里建立如图 5-6 所示的单总线结构的模型机。

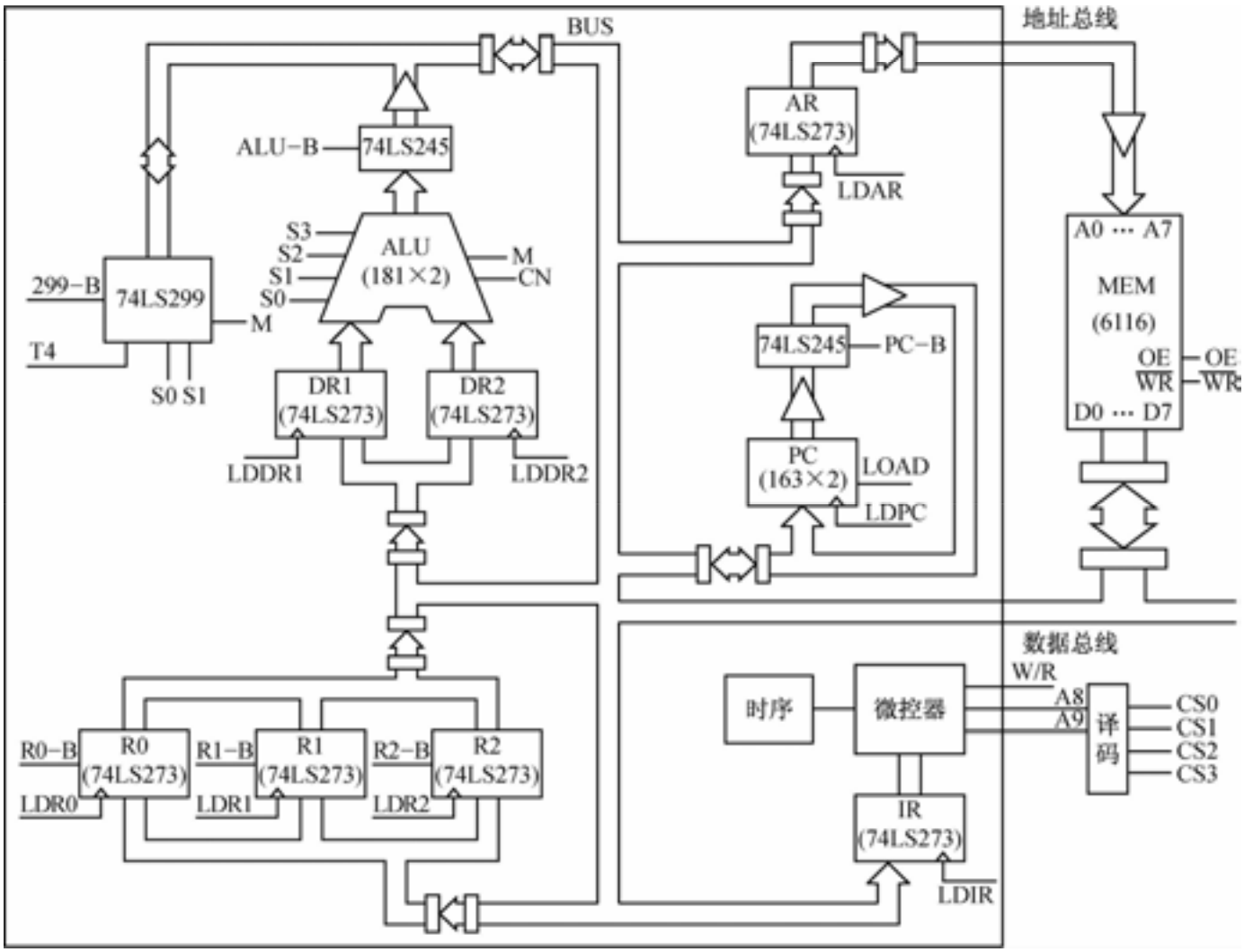

图 5-6　单总线结构的模型计算机框图

1)模型机硬件结构

模型机中,R0、R1、R2 是通用寄存器,DR1、DR2 为暂存器,是为运算器中 ALU 的工作服务的,同时 DR1 和 DR2 两个暂存器也作为 ALU 的两个输入多路开关使用,采用锁存器的方式实现。地址寄存器 AR 和指令暂存器 IR 均由 8 位 D 触发器构成,74LS299 构成移位寄存器。

2)模型机控制信号

模型机控制信号来自于模型机的微控制器,它是在指令的控制下产生控制信号的部件。在微程序方式下,微控制器是由多位存储器构成的。本模型机是由 32 位输出来构成控制信号集的,硬件结构为 6116 静态存储器。当然,当指令集确定后,可由掉电后能保持信息的只读存储器来实现,结合时序给出各控制点信号,完成指令的执行。下面介绍主要的控制信号。

(1)PC-B(program counter-bus):程序计数器送总线信号。程序计数器是计算机内部用于程序执行过程中表述执行地址的寄存器,它通过 PC←PC+1 进行程序的顺序执行。在本模型机中,PC 是由两片 74LS163(二进制同步计数器电路)构成的,每来一个 LDPC,执行 PC←PC+1。LOAD 用于非顺序程序结构的新 PC 值装入。

(2)LDAR(load address):加载地址控制信号,地址数据在取指令阶段来自 PC,在其他寻址方式下经 ALU 计算得到或来自于其他寄存器。地址输出用于主存储器的寻址(此处为 MEM 6116(2 KB×8)的静态存储器),取出存放于主存储器的指令或数据。

(3)OE、$\overline{WR}$:6116 主存储器片选,写入信号。OE 有效,6116 可输出对应单元的数据到数据总线;$\overline{WR}$有效时可写入数据。

(4)LDDR1、LDDR2:从数据总线装入数据到 DR1 或 DR2 信号。由于同一时刻只能有一个数据来自总线,所以 LDDR1 和 LDDR2 一定是互斥信号。

(5)LDR0、LDR1、LDR2:为 R0～R2 寄存器的数据装入信号。

(6)R0-B、R1-B、R2-B:为 R0～R2 寄存器输出数据到总线的控制信号。同样由于同一时刻只能有一个数据来自总线,所以 R0～R2 的输出控制信号 R0-B、R1-B、R2-B 一定是互斥信号。

(7)S0～S3、M、CN:74LS181 运算器输入控制信号。

(8)ALU-B:74LS181 运算器输出控制信号,运算完成后由 74LS245 总线驱动器输出至总线。

(9)T4:从总线打入数据至 74LS299,经 S0、S1 和 M 信号完成移位。完成移位后可由信号 299-B 输出至总线。

3.模型机的指令格式

1)模型机的数据格式

为了简化模型机,规定采用定点补码表示数据,且字长为 8 位,其格式如下:

D7	D6…D0
符号位	数值位

整型数值的范围为[－128,127]时,若为小数则表达的范围为[－1,1)。

2)指令格式

模型机设计四大类指令共 16 条,包括算术/逻辑指令、I/O 指令、访问指令、转移指令和停机指令。

(1)算术/逻辑指令。涉及 9 条算术/逻辑指令并用单字节表示,寻址方式采用寄存器直接寻址,其格式如下:

7 6 5 4	3 2	1 0
OP-Code	Rs	Rd

其中,OP-Code 为操作码,Rs 为源寄存器,Rd 为目的寄存器,并规定 00、01、10 对应寄存器 R0、R1、R2,即 00 表示使用 R0,01 表示使用 R1,10 表示使用 R2。该代码经微控制器产生 LDR0～LDR2 信号。

(2)I/O 指令。输入和输出指令采用单字节指令,其格式如下:

7 6 5 4	3 2	1 0
OP-Code	Addr	Rd

其中,Addr=10 时,选中“INPUT DEVICE”中的开关组作为输入设备;Addr=11 时,选中“OUTPUT DEVICE”中的 LED 作为输出设备。

(3)访问指令及转移指令。模型机设计两条访问指令,即存数(STA)、取数(LDA);两条转移指令,即无条件转移(JMP)、结果为 0 或有进位转移指令(BZ/C)。两种指令的指令格式如下:

7	6 5	4	3 2	1 0
0	M	0	OP-Code	Rd
D				

其中,Rd 为目的寄存器地址(LDA、STA 指令使用)。D 为位移量(正负均可),M 为寻址模式,其定义如下:

寻址模式 M	有效地址 E	说明
00	E=D	立即寻址
01	E=(D)	直接寻址
10	E=(PC)+D	变址寻址
11	E=(RI)+D	相对寻址

(4)停机指令。其指令格式如下:

7 6 5 4	3 2	1 0
OP-Code	00	00

4.微操作序列

控制器在实现一条指令的功能时,总是把每一条指令分解成一串时间上先后有序的最

基本、最简单的控制操作动作。

在计算机中，一个最基本、最简单的操作称为微操作。执行指令的一串微操作动作，称为指令的微操作序列。

针对以上模型机，介绍以下典型的微操作序列。

1)“从主存中取出数”的微操作序列

要从存储器中读出数的信息，必须先将该字的地址由 PC 送入 AR，PC←PC+1，发送读内存命令 OE 和 W/$\overline{R}$=0，若该字是指令字，则将数据通过总线送往 IR 中，即完成取指令的操作。

若是数据，可根据指令的要求，送到不同的目的地。现假设访问的是数据，并且要访问的存储单元的地址存放在 R1 寄存器中，将取得的数据装入 R0 寄存器中，则实现该功能的微操作序列如下：

(1)R1→PC，LDPC 有效，完成 R1→PC。

(2)PC→BUS，PC←PC+1，LDAR 有效，完成 PC→AR。

(3)AR→地址总线，OE 有效，W/$\overline{R}$←$\overline{R}$，完成存储器读所需地址和控制信号。

(4)控制器等待存储器完成数据输出后给出 LDR0 有效，数据装入 R0。

2)“向主存中写数”的微操作序列

要从存储器中读出数的信息，必须先将该字的地址由 PC 送入 AR，PC←PC+1，发送读内存命令 OE 和 W/$\overline{R}$=1，此时将数据送到总线，即完成存数操作。

假设存数的存储单元的地址存放在 R1 寄存器中，将 R0 寄存器的数据装入其中，则实现该功能的微操作序列如下：

(1)R1→PC，LDPC 有效，完成 R1→PC。

(2)PC→BUS，PC←PC+1，LDAR 有效，完成 PC→AR。

(3)控制器 R0-B 信号，R0 送出数据到 BUS。

(4)AR→地址总线，OE 有效，W/$\overline{R}$←W，完成存储器读所需地址和控制信号，存储器完成存数操作。

3)ADD R0，D

这条指令是将寄存器 R0 与立即数 D 相加，结果送 R0。它是立即寻址方式。

取指令时，PC→地址总线，PC←PC+1，微控制器送 OE=1 和 W/$\overline{R}$=W=1，LDIR 有效，从内存取指令由数据总线送 IR。经由微控制器译码完成如下操作：

(1)R0→BUS，LDDR1 完成寄存器 R0 送 DR1。

(2)PC→地址总线，PC←PC+1，微控制器送 OE=1 和 W/$\overline{R}$=$\overline{R}$=0，LDDR2 有效，从内存取数由数据总线送 DR2。

(3)微控制器通过 S0～S3、M、CN 发出加法信号。

(4)ALU→BUS，LDR0 有效，运算结果送 R0。

加减和逻辑指令只要改变 S0～S3、M、CN 就可完成 32 种运算或优选的数种运算。乘除法运算可结合运算分析，由 74LS299 进行移位完成。

4)SUB R0，(R1)

本条指令是将寄存器 R0 中的操作数与 R1 所指的地址中的操作数相减，结果送 R0。它是寄存器间接寻址方式。

PC→地址总线，PC←PC+1，微控制器送 OE=1 和 W/$\overline{R}$=W=1，LDIR 有效，从内存取

指令由数据总线送 IR。经由微控制器译码完成如下操作：

(1)R0→BUS，LDDR1 完成寄存器 R0 送 DR1。

(2)R1→BUS，LDPC 有效，即 R1→PC。

(3)PC→地址总线，PC←PC＋1，微控制器送 OE＝1 和 W/$\overline{R}$＝$\overline{R}$＝0，LDDR2 有效，从内存取数由数据总线送 DR2。

(4)微控制器通过 S0～S3、M、CN 发出加法信号。

(5)ALU→BUS，LDR0 有效，运算结果送 R0。

5)转移指令 JMP D

本条指令是跳到当前 PC＋D 所指地址执行下一条指令。

转移类指令是从取指周期进入执行周期的。有两种转移指令：一种是无条件转移指令，另一种是条件转移指令。程序转移执行的去向地址是通过指令格式中的位移量(offset)来表示的，此处 offset＝D。

由于 PC 通常在取指令阶段已递增 1，故上述 PC 值已经指向本条转移指令的下一个单元。

从操作也可看出，完成的实际是 PC＋1＋D。

PC→地址总线，PC←PC＋1，微控制器送 OE＝1 和 W/$\overline{R}$＝W＝1，LDIR 有效，从内存取指令由数据总线送 IR。经由微控制器译码完成如下操作：

(1)PC→BUS，PC←PC＋1，微控制器送 OE＝1 和 W/$\overline{R}$＝$\overline{R}$＝0，LDDR2 有效，从内存取数据 D 由数据总线送 DR2。

(2)PC→BUS，LDDR1 有效，PC 送向 DR1。

(3)微控制器发出通过 S0～S3、M、CN 发出加法信号。

(4)ALU→BUS，LDPC 有效，运算结果送 PC。

如果是条件转移，则在步骤(3)中，满足条件完成 DR1＋DR2，不满足条件则完成 DR1＋0 操作，然后再送 PC。

5.微操作序列总结

指令在计算机硬件执行时，总是分了若干步进行，数据通路控制传送数据和运算每一步都可看成是微操作。总结所需的微操作可得到：

(1)取指令操作：PC→AR，PC←PC＋1，RAM→BUS，BUS→IR。

(2)读存储器数据：PC→AR，PC←PC＋1，RAM→BUS，BUS→Rd。

(3)写存储器数据：PC→AR，PC←PC＋1，Rs→BUS，BUS→RAM。

(4)运算操作：大部分运算操作可在数据送达 ALU 的两输入端时由 S0～S3 给出操作命令，但在间接寻址时也可执行操作 DR1＋DR2→BUS，BUS→AR。

(5)其他必需的基本微操作，任何一台计算机，若其指令系统中的每条指令功能都由硬件实现，则均有一个严格有序的微操作步序列。这些微操作控制信号由微操作控制器提供。从这里也可以清楚地看到，运算器中的数据通路是如何在控制信号作用下加工处理及传输数据的。

5.3.2 指令运行过程中的事务处理

在指令执行过程中，通常有许多事务需要及时处理。例如，程序正在运行时，控制台有停机要求，或电源掉电，或外设有输出数据请求，或程序执行中发生错误需要及时处理等。这些事务可能与现行程序的执行有关，也可能无关，但都要求能及时得到处理。如何在指令执行流程中安排适当的时刻来处理这些事务呢？目前多数机器是按照以下原则进行的：

(1)对于来自控制台的停机要求，当现行指令执行完后由 CPU 控制停机。

(2)对于高速外设要求占用总线控制权而与主存进行数据传输的请求，属于直接传输请求，优先级最高，现行指令执行中，在当前一个 CPU 周期结束后，就冻结指令执行，CPU 转让总线控制权，挪用一个存储器周期给这类高速外设进行数据传输处理。

(3)电源掉电，这直接威胁程序的运行，可用电源设备技术来设法延缓几毫秒后才真正断电，可在这几毫秒内紧急进行掉电前的现场保护处理，通常是当现行指令执行完成后，进行掉电事务处理。

(4)对运行中的故障事务处理，视其紧迫程度，分别在当前 CPU 周期结束或现行指令执行完后进行故障和错误处理。

(5)对一般性外部设备的数据传输请求，通常安排在 CPU 每个机器周期状态的开始进行采样扫描接收，在现行指令执行完后，CPU 视其请求判断优先等级并进行事务处理。

一个机器周期状态结束，下一个机器周期状态开始的转换瞬间是查询有无各种事务处理要求的时刻，其操作流程如图 5-7 所示。

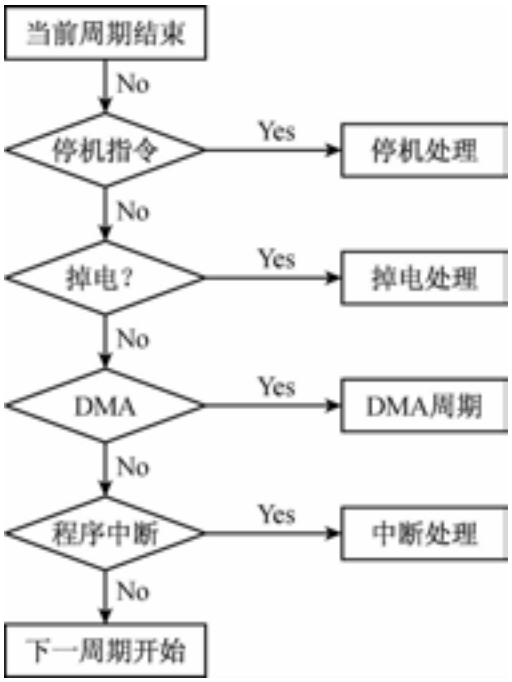

图 5-7 指令执行过程中的事务处理

5.3.3 时序及功能

时序系统的功能是为指令的执行提供各种操作定时信号。指令系统中的每条指令均有一个微操作序列，当用这些微操作信号去控制硬件线路产生相应动作时，应注意微操作控制信号彼此间严格的时间顺序。即使在同一操作步骤中，由于被控制操作的硬件线路、元器件

性能不同，各个操作控制信号在时序作用宽度上的要求也不尽相同，有的需要宽脉冲作用，有的需要窄脉冲作用。在机器中必须有产生这些定时信号的部件，即时序部件。

时序部件就是计算机的机内时钟。用其产生的周期状态、节拍电位以及时标脉冲对指令周期进行时间划分和标定。把这些有时序关系且符合线路控制要求的时序信号连同指令码和操作中反馈条件信号等要求综合加工，就可以产生各种微操作控制信号。

在计算机中，由于各种指令的操作功能不同，所以各种指令的指令周期也是不同的。指令周期通常由若干机器周期表示，机器周期也称为 CPU 周期。由于 CPU 内部的操作速度较快，而 CPU 访问一次内存所花的时间较长，因此许多计算机系统中通常用存储读写周期为量度来规定 CPU 周期。

不同的指令，其周期状态组合也是不尽相同的。例如，单操作数指令是由取指周期、取操作数周期和执行周期 3 个机器周期组成的，而双操作数指令由取指周期、取源操作数周期、取目的操作数周期和执行周期 4 个周期组成。

周期是在一个控制阶段内持续起作用的信号，通常用周期状态触发器来标识并指明某个周期控制。周期状态触发器之间的关系是互斥的。假设 1 为有效状态，则当某个触发器为 1 时，表示机器进入处理指令阶段，并执行该阶段的微操作序列。例如，取指周期触发器为 1 时，由它控制取指阶段的各个微操作。

在一个机器周期内要完成一系列微操作。一般在计算机系统中，把一个机器周期分成若干相等的时间段，每一个时间段对应一个电位信号，称为节拍电位。在计算机中，一般都以能保证 ALU 进行一次运算操作作为一拍电位的时间宽度。

一个机器周期究竟划分为多少个节拍最合适，取决于该周期时间内需要顺序完成的动作步数，而一步动作通常是指一次加法或一次寄存器传输。

如果指令在不同周期内需要完成的动作内容差别很大，则为它们选定统一的节拍数是很困难的。通常在照顾多数指令的要求下，选取最少的节拍数，然后对少数执行时间长或不确定的指令采取一些变通的延长周期的措施。

通常，在一个节拍内还设置一个或几个具有一定宽度的时标脉冲。时标脉冲的宽度一般为节拍电位宽度的 1/N，只要能保证所有的触发器都能可靠稳定翻转即可。

一台计算机内的控制信号一般由若干周期状态、若干节拍电位及若干个标脉冲三级控制时序信号定时完成。周期、节拍及时标脉冲的个数按实现指令的需要而定。三级控制时序信号的组合关系如图 5-8 所示。

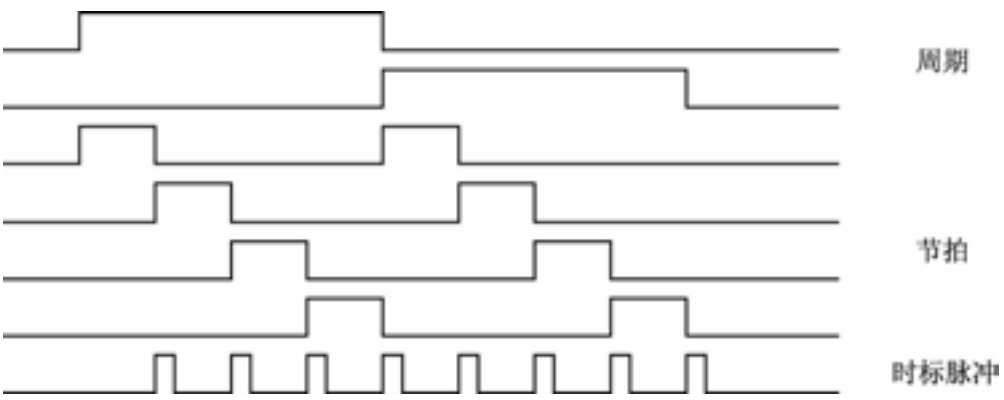

图 5-8　三级控制时序信号的组合关系

三级控制时序信号的宽度均成正整数倍同步关系。

周期状态之间、节拍电位之间、时标脉冲之间既不容许有重叠交叉，又不容许有空白间隙，应该是能一个接一个地准确连接，一个降落、另一个升起准确切换的同步信号。这意味着对整个指令周期中的每一个瞬间，都能用周期状态、节拍电位和时标脉冲唯一、准确地给予定义和标定。

不同的指令格式及寻址方式，对时序系统的影响有很大不同。例如，在直接寻址方式中，若指令格式是一地址指令，则该指令执行时只访问存储器一次，故时序系统仅需向存储器发送一次读/写命令；在二地址指令格式中，时序系统要向存储器发送两次读/写命令；多地址指令则需要更多。同时，由于操作数的寻址方式多种多样，因此在时序系统中，必须考虑各种寻址方式所带来的不同操作命令的安排与产生。因此在设计时序系统时，必须首先对机器指令系统的全部指令执行过程所需要的微操作、经过的数据传输途径及所需要的时间进行全面分析，在此基础上再确定控制方式，然后进行时序系统的设计。

5.3.4 CPU 的控制方式

控制器控制一条指令运行的过程是依次执行一个确定的微操作序列的过程。由于不同指令所对应的微操作步骤及其繁简程度有很大的差别，因此每条指令和每个微操作所需的执行时间也不相同。例如，无条件转移指令的执行，仅需要把形成的转移去向地址传输至程序计数器 PC，即可完成其功能；而算术运算、逻辑运算则一般需要的操作步骤比较多，执行的时间比较长。

控制不同微操作序列的时序控制信号的方法称为控制器的控制方式。控制方式通常分为同步控制方式、异步控制方式和同异步联合控制方式 3 类。

1. 同步控制方式

同步控制方式又称固定时序控制方式或无应答控制方式。任何指令的执行或指令中每个微操作的执行都受事先安排好的时序信号的控制，每个时序信号的结束就意味着一个微操作或一条指令已经完成，随即开始执行后续的微操作或自动转向下一条指令的执行。

在同步控制方式中，每个周期状态中产生统一数目的节拍电位及时标工作脉冲。不同的指令，微操作序列和操作时间也不一样。对同步控制方式要以最复杂指令的实现需要作为基准，进行控制时序设计。

同步控制方式设计简单，操作控制容易实现。但大多数指令实现时，会有较多空闲节拍和空闲工作脉冲，形成较大数量的时间浪费，影响和降低指令执行的速度。

2. 异步控制方式

异步控制方式又称可变时序控制方式或应答控制方式。执行一条指令需要多少节拍，不作统一规定，而是根据每条指令的具体情况而定，需要多少时标信号，控制器就产生多少时标信号。这种控制方式的特点是：每一条指令执行完毕后都必须向控制时序部件发回一个回答信号，控制器收到回答信号后，才开始下一条指令的执行。

这种控制方式的优点是每条指令都可以在最短的、必需的节拍时间内执行完毕，指令的运行效率高；缺点是由于各指令功能不一样，微操作步序列长、短、繁、简不一致，节拍个数不同，控制器需根据情况加以控制，故控制线路比较复杂。

异步工作方式在计算机中得到了广泛的应用。例如，CPU 对内存的读写操作，I/O 设备与内存的数据交换等一般都采用异步工作方式以保证执行时的高速度。

在单总线结构的计算机中，通过总线进行数据交换，一般采用主从关系，异步工作方式。占用总线控制权的设备称为主设备，与主设备进行数据交换的设备称为从设备，这种以主设备为参考点，向从设备发出信息或接收从设备送来的信息的工作关系，称为主从关系。异步工作方式一般采用两条定时控制线来实现。人们把这两条控制线称为“请求”线和“回答”线。当系统中两个部件 A 和 B 进行数据交换时，若 A 发出“请求”信号，则必须有 B 的“回答”信号进行应答，这次操作才是有效的，否则无效。

3.同异步联合控制方式

现代计算机系统中一般采用的方式是同步控制和异步控制相结合的方式，即联合控制方式。对不同指令的各个微操作实行大部分统一、小部分区别对待的方法。一般的设计思想是，在功能部件内部采用同步控制方式，而在功能部件之间采用异步控制方式，并且在硬件实现允许的情况下，尽可能多地采用异步控制方式。

例如，在一般微型机中，CPU 内部基本时序节拍关系采用同步控制方式，按多数指令的需要设置节拍数目与顺序，但对某些指令的控制要求可能不够用，这时采取插入节拍、延长节拍或延长周期时间的方式，使之满足各指令的需要。这些控制时序均体现了基本同步控制、局部异步协调控制的思想。

再例如，当 CPU 要访问存储器时，在发送读/写命令后，存储器进入异步工作方式，当存储器访问完毕以后，会向 CPU 发回一个信号，表示解除对同步时序的冻结，机器又按同步时序运行（或发出一个 WAIT 信号冻结，不发信号时解除冻结）。

指令种类往往直接影响到控制方式的确定。如果机器指令系统中，各类指令所需要的微操作序列以及执行时间比较一致，则应该采用同步控制方式，用统一的时序信号控制所有指令的执行；如果各类指令的微操作序列及其执行时间相差很大，则可采取同异步联合控制方式，对微操作序列比较接近的指令采用同步控制方式，对差别不是很大的指令采用插入节拍或延长节拍或周期的方法，还有些指令，若与其他指令很难同步，则可以采用异步工作方式处理。

5.4　微程序设计技术和微程序控制器

微程序控制技术在现今计算机设计中得到了广泛的采用，其实质是用程序设计的思想方法来组织操作控制逻辑。将微操作控制信号按一定规则进行信息编码，形成控制字。一条机器指令的执行过程，就是对若干控制字进行程序设计，此程序存放在专用存储器中。存放微程序的存储器称为“控制存储器”。

早在 20 世纪 50 年代初，英国剑桥大学的 Maurice V. Wilkes 教授就提出了“微程序设计”概念，但直到微电子技术的发展提供了高速的 ROM、RAM 之后，微程序设计技术才被大量使用。

5.4.1　微程序设计技术

由于微程序实现计算机的机器指令功能时，是存储在控制存储器中的，所以，改变控制存储器的内容，就可以方便地改变指令特性，增删指令，甚至改变指令系统。这给计算机设

计者和用户提供了相当大的灵活性。

同时这种控制技术能使机器逻辑设计规整，提高了可靠性、可利用性及可维护性（简称RAS技术），大大优化了硬件控制技术。

另外，这种控制技术有利于机器设计时的仿真。也就是说，在M1机器上使用M2机器语言编写程序并运行，从用户角度来看，M1和M2无区别。要做到这一点，只有计算机具有控制存储器的微程序设计结构才行。

微程序控制方法与组合逻辑控制方法相比，在诸多方面有着显著的差别。从实现方式上来讲，微程序控制器的控制功能是在存放微程序的控制存储器和存放当前正在执行的微指令寄存器直接控制下实现的，而组合逻辑控制方法则由逻辑门电路组合实现。前者比较规整，各条指令控制信号的差别反映在控制存储器的内容上，因此无论是增加或修改指令，只要增加或修改控制存储器中的内容即可。组合逻辑控制器的控制信号先用逻辑表达式列出，经过简化后用电路实现，因而显得零乱且复杂。修改指令或增加指令非常麻烦，有时甚至不可能。因此微程序控制得到了广泛的应用，尤其是指令系统复杂的计算机，一般都采用微程序控制方式来实现控制功能。

从性能上比较，在同样的半导体工艺条件下，微程序控制的速度比组合逻辑控制方式的速度慢。这是因为执行每条微指令都要从控制存储器中读取一次，影响了速度，而组合逻辑控制方式的速度取决于电路延迟。因而在超高速计算机中，对影响速度的关键部分，如CPU，往往采用组合逻辑控制方法。近年来，在一些新型计算机结构中，如RISC结构，一般选用组合逻辑方法。

微程序设计方法增强了诊断能力。在组合逻辑控制的机器中，如果要诊断机器故障，只能编写软件诊断程序，但大多数的故障会使机器无法执行，从而无法诊断。在微程序控制的机器中，可将诊断程序写成微程序，这些微程序可动态地传输至控制存储器，或常驻在控制存储器内。诊断程序可以实现一步步沿着一定的路径对整个系统进行扫描，它比用软件诊断有更多的运行机会。其次，由于微程序与系统之间有更密切的关系，更能准确地指出故障的原因。

此外，微程序控制还具有可靠性高、成本较低的优点。微程序开发在许多方面类似于软件开发，所以软件工程中行之有效的一系列开发手段都可应用于微程序的开发上。

1.基本概念

1）控制字

在每个节拍中，安排有许多微操作控制信号。如果把每个微操作信号用一个独立的二进制信息码表示，则这些二进制位的总和就组成了一个字。这是一个表征微操作控制要求的字，称为控制字（CW）。而指令的微操作时序表就相应地变成该指令的控制字序列。

2）微命令

在硬件联合控制中，把用来完成像打开和关闭控制门这样一类最基本操作动作的信号称为操作控制信号。而在微程序控制中，把这些控制信号称为微命令。微操作是微命令在时序的配合作用下的操作过程。

3）微地址和微指令

一条指令的控制序列存放于控制存储器中，为了读取这些控制字，必须给每个存放控制字的单元都赋予“地址”，此地址称为“微地址”。具有微地址的控制字称为“微指令”。

4）微程序

一系列微指令的有序集合构成微程序。一个微程序可以实现一条机器指令的功能。即

一条机器指令可以分解为若干条有序的微指令，将微程序存入控制存储器中，机器指令的执行过程实际上变成微指令的执行过程。

5)微周期

微周期就是从控制存储器中读出一条微指令并执行相应操作所需要的时间。

2.微指令编译法

指令的微程序是若干条指令的一种有序集合。要执行一条机器指令，只需从主存储器取出该指令码，再按照指令的要求给出该指令微程序的起始微地址，依次从CM中逐条取出微指令并实现其控制，在微程序编制上，有许多共同点。为优化控制，缩短微指令字长，完全可以把传统的较成熟的程序设计技术应用于计算机控制结构设计中。

同程序设计相比，微程序设计更依赖于机器结构组成和时序系统。不过，编制微程序的思想和方法却类似于编制机器指令程序，只是这两种程序实现的目标不一样而已。

影响编译方法的因素很多。大型机强调速度，要求编译尽量具有快速性；中、小型机是兼顾速度和价格，要求在确保一定速度的情况下，尽量缩短微指令字长；小、微型机则更多地注意经济、实惠，要求最大限度地缩短微指令字长。各类计算机从各自的特点出发，设计了各种各样的编译法。尽管如此，仍然有基本的、共同的设计准则。各种编译法的选择原则如下：

(1)缩短微指令的长度。

(2)提高微操作的并行性。

(3)提高机器的控制性能并降低价格。

(4)有利于微程序设计的灵活性。

这几项指标是互相制约的，选择时应进行全面的分析和权衡。

1)直接控制法

在这种形式的微指令字中，每一个独立的二进制位代表一个微命令。例如，该位为1表示要执行这个微命令，该位为0表示不执行这个微命令。每条指令都输出一组能控制的控制信号。从定义微命令到实现微操作不再需要把编码的信息译成控制信号，而只要在时序配合下就能进行直接控制。

采用这种方法，若微命令总数为n，则微指令字长就应有n位。n条微命令可以并行n个操作，从而控制实现多种信息的传输处理，故操作速度快。此法直观、方便，但问题在于对于一般中、小型机来说，其微命令就多达几百个，若采用此法，必然使微指令字长达几百位。这对控制存储器横向容量来说，已经到了难以容忍的程度。所以，这种方法仅适用于高速控制部件，在复杂系统中很少采用。按直接控制法编码的微指令，又称水平型微指令，它是面向数据通路的控制门(或控制点)。

水平型微程序设计的优点是，可以利用硬件能并行执行的特点，使编制机器指令的微程序所需的微指令条数较少，执行速度也比较快。但是，水平型微指令字长较长，增加了控制存储器的容量，同时微指令和机器指令差别较大，编制程序难度较大。

2)最短编译法

最短编译法的基本思想是：使每一条L位字长的微指令只定义一个微命令，即用L位二进制组合编码00…0～11…1中的每个码值来表示一条微命令。例如，若L取值为6。就有64种编码状态，则可定义的微指令最多有64条。由于最短编译法采用二进制编码状态，所以微指令的输出必须经过译码才能得到微命令。

按最短编译法编码的微指令又称垂直型微指令，它是面向算法来编码的。类似于传统

的程序设计方法，容易掌握，编程比较简单，且微指令字长较短，微指令字中各位都能得到充分利用。然而，由于每条垂直微指令只能对应一个微命令，因此并行控制能力差，微程序长度较长，执行速度慢。

3)字段编译法

实际上，一台机器的微指令往往不局限于水平型或垂直型微指令中的一种格式，而是采用折中的方法。字段编译法就是折中方案，具有两者的优点，且克服了两者的缺点。字段编译法又分为字段直接编译法和字段间接编译法。

(1)字段直接编译法。如图 5-9 所示，字段直接编译法的具体方法是把一条微指令分成几段，段与段间按水平法设计，每个段内分别按垂直法进行编码，每一段形成一个微命令，一条微指令可同时有并行的几个微命令。

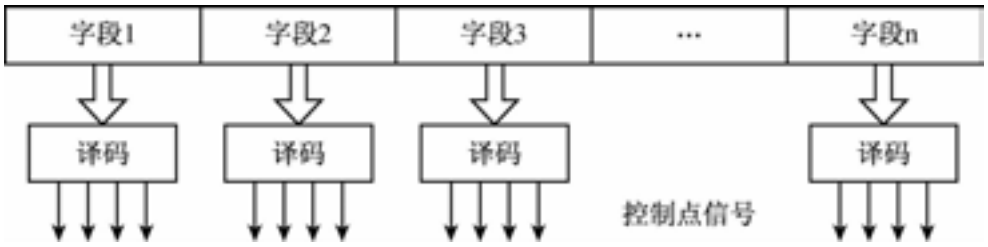

图 5-9　字段直接编译法

例如，设一个微指令字长为 20 位，分成 7 段，分别为：2 个 4 位一组，2 个 3 位一组，3 个 2 位一组，则可定义 60 条微命令。相比较，当指令字长为 20 位，用直接控制法则可同时产生 20 个微命令并行操作，但也只能定义 20 条微命令；若用最短编译法，每条微指令则只有 1 个微命令操作，可定义 2^{20} 条微命令。因此字段直接编译法得到了广泛的应用。

微指令字分段的原则如下：

在同一节拍内，需要互相配合起作用的微操作是并行操作，其微命令可以分在不同的字段内，以便配合进行微操作控制(组合性的操作控制)。这是微命令的相容性。

在同一节拍内，不允许同时出现具有排他性的微操作，只能是串行操作，其微命令可分在一个字段内，这是微命令的互斥性。

例如，在运算器中，两个锁存器接收数据，需要将锁控制信号清零，这个信号应分在不同的字段中，以便允许同时作用；而运算器的输出门控制，有直送、左移、右移及半字交换 4 种操作，但在同一时刻，只可能有一种传输方式，只允许一种微操作控制，这种互斥性微命令就可分在同一字段编码。一般来说，是把相互有关系而又互斥的微命令分在一个字段内编码，而把相互有关系而又需要并行操作的微命令分在不同字段中编码。例如，F2 字段占微指令字长两位，可以表示 4 条互斥的微命令，如下所示：

- 00：直送。
- 01：左移。
- 10：右移。
- 11：半字交换。

(2)字段间接编译法。如图 5-10 所示，字段间接编译法是在字段直接编译法的基础上，用来进一步缩短指令字长，组合零散微命令的一种编译法。若在字段直接编译法中再规定一个字段的某些微命令要由另一个字段中的某些微命令来解释，这样的编译法就称为字段间接编译法。在一个字段中除了已有明确定义的多个微命令之外，尚有若干微命令的定义

要由另一个字段的微命令或某个标志位加以明确解释。这种编译法适用于把那些不同类型的、不常用的，但数量又可观的零散的微命令编入少数几个字段之中，以减少微指令字的长度，组合编译更多的微命令。

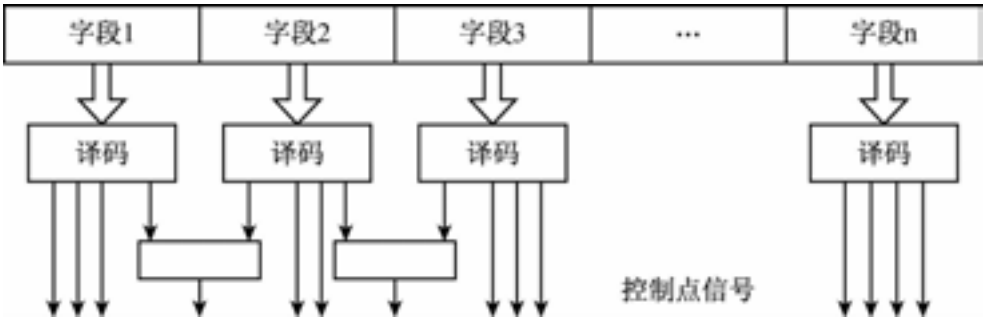

图 5-10　字段间接编译法

可见，字段间接编译法的译码系统较复杂一些，虽然可以进一步减少指令长度，但有可能削弱微指令的并行控制能力，通常只作为直接编译法的一种补充。

3.微程序流的控制

在计算机中，微程序以编码（微码）形式按给定的微指令的地址存放在控制存储器的相应单元中。微程序执行时，只要依次给出各个微指令的地址，就能使微程序连续执行，直至完成为止。因此，要使微程序连续执行，关键在于当前微指令执行完成以后，如何确定后续微指令的地址。

微程序的执行过程与一般的机器目标程序执行过程类似，也有执行顺序的控制问题。在 机器指令程序设计中有顺序程序设计、分支程序设计和循环程序设计之分，微程序技术亦有相应的设计技术问题。

每条机器指令均有一个与其对应的微程序，微程序执行顺序的控制，是通过指定微指令的微地址来进行的。微程序的执行顺序与其首地址如何给出，以及后继微地址如何形成有密切关系。

下面对微地址的形成进行分析。

1）初始微地址的形成

每条机器指令对应一段微程序，当执行公用的取指微程序，从主存中取出机器指令后，由机器指令的操作码指出微程序的首地址。这是一种多分支情况，通常有以下几种方式：

（1）操作码的位数与位置固定，这时可直接使操作码与微地址码的部分位相对应。例如，若微程序入口地址＝000C，则控制存储器第 0 页的一些单元被安排为各个微程序入口（首地址），再通过无条件微转移使这些单元与相应的后续微指令相连接。当然，指令的操作码也可以与微地址码的其他位相对应，此时，可有若干后续单元用于存放相应的微程序。当空出的单元不足以存放所有的微程序时，仍然可采用无条件转移的方法再与其他后续微指令相连接。

（2）当每类指令的操作码位数与位置固定，而各类指令之间的操作码位数与位置不固定时，可采用分级转移的方法。先按指令类型转移到某条微指令，区分出是哪一大类，然后进一步按机器指令的操作码转移，区分出是哪一种具体的机器指令。

（3）当操作码的位数与位置都不固定时，通常的方法是采用 PLA（可编程逻辑阵列，一种硬件的方法）实现。将每条机器的操作码翻译成相应的微程序入口，即 PLA 的输入是机

器指令的操作码，输出是相应的机器指令在控制存储器中微程序的入口。

2）后继微地址的形成

计算机得到微程序入口以后，就开始执行微程序，后继微地址的形成方法对微程序编制的灵活性影响很大。通常采用下面两种方法形成后继微地址。

（1）增量方式。这种方式和机器指令的控制方式类似。将微程序起始地址送入微程序计数器 μPC 中，由 μPC 的递增来控制微程序的顺序执行。这样，后继微地址平时来源于 μPC。把影响微地址的因素，如状态标志位、条件码、指令码以及从控制存储器中读出的微指令字的各测试位，均送到微地址产生器中进行逻辑分析、综合判别，形成新的有效转移微地址，最后再送到 μPC，就可实现微程序执行进程的转移。实现这些逻辑功能时利用了程序设计技术中的条件转移指令概念。

但微程序编制与一般程序编制有一个明显的不同点，那就是对一般程序设计（如汇编语言程序）来说，主要考虑的是如何使占用的内存单元少，执行程序所需的时间短，按此原则来巧妙地选择指令，完成程序设计的任务。一般程序中，顺序执行的指令占多数，转移类指令占少数。而在微程序设计中，由于控制存储器容量很小，其中保留的是完成指令系统所需要的全部微指令。编写微程序时，只能选用这些微指令，因而微地址相连顺序的情况较少，更多的是微地址转移类的微指令。因而，单纯采取微程序计数器及条件转移微指令并不好，这会使转移微指令数目的比重大大增多。实际上可采用更好、更简便的办法：给微指令字增加一个下地址字段（NAF），即指明下一条可能要执行的微指令的地址。

（2）在微指令字的格式中，增设下地址字段之后，就可以用微地址寄存器（μAR）取代程序计数器。下一条微指令地址在多数情况下可由现行微指令字的下地址字段直接给出，少数情况由微地址产生器对下地址字段进行修改后产生，这种方法也称为断定方式。微指令字格式如图 5-11 所示。

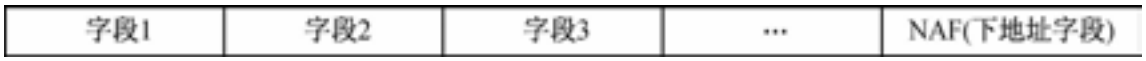

图 5-11　微指令字格式

综上所述，后继微地址的形成是设计微程序控制的关键问题之一。确定后继微指令地址有以下几种情况：

（1）顺序执行时，后继微地址可以由现行微指令字的下地址字段 NAF 或微程序计数器 μPC 直接确定。

（2）无条件转向的后继微地址，可以由现行微指令字的下地址字段确定。

（3）有条件转向的后继微地址由现行机器指令操作码，现行微指令执行时产生的状态特征或条件码的判别结果决定。

用带下地址字段的微指令去编写微程序时，若微程序是无分支的顺序执行或无条件转移，则下条微指令地址可以直接由 NAF 给出。若是有条件转移，而且条件是机器运行中的状态标志值或指令的寻址方式等外界因素，则后继微地址形成的方法是：以 NAF 的值为基本地址，把外界因素作为修改因子，修改 NAF 字段值以生成下一条微指令地址值。

5.4.2　微程序控制器

在微程序控制方式中，一条机器指令的执行过程就是其微程序的执行过程。类似于一

般程序的执行，控制器中需要增设一个微指令计数器。微程序控制器一般采用微地址寄存器(μAR)和下地址字段 NAF 来代替 μPC。

1.基本组成

微程序控制操作中，也可能依据执行中出现的标志状态，产生微程序的分支、转移问题。这样，微程序控制部件的基本结构如图 5-12 所示。

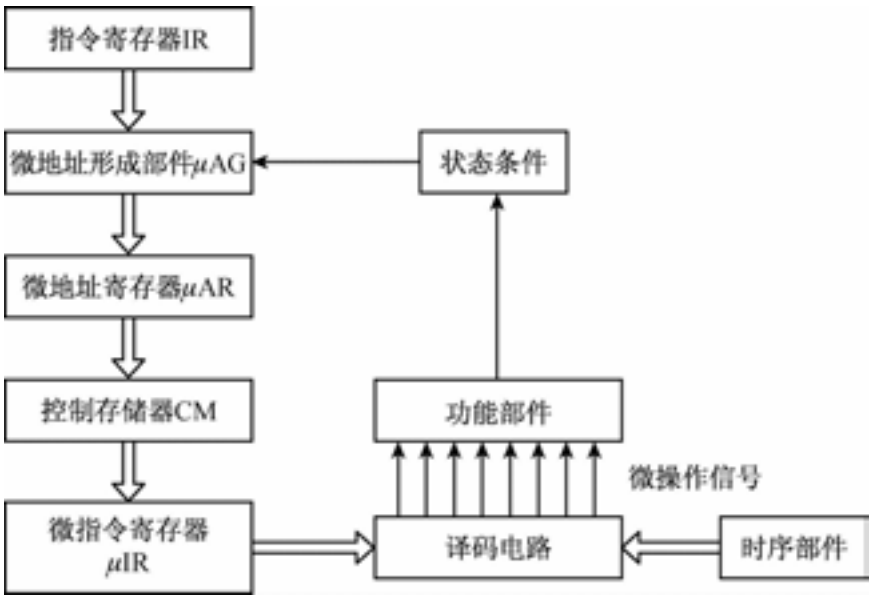

图 5-12　微程序控制器基本结构

(1)指令寄存器 IR。用于存放从计算机主存储器中取出的机器指令，直到该指令执行完毕。

(2)微地址形成部件 μAG。用来产生机器指令的首条微指令地址和后续地址。

(3)微地址寄存器 μAR。接收微地址形成部件送来的微地址。

(4)控制存储器 CM。用来存放微程序。

(5)微指令寄存器 μIR。用来存放从控制存储器中取得的微指令。

(6)译码电路。完成对 μIR 相应微字段的译码，形成微操作控制信号。

(7)时序部件。给译码电路提供时序，按指令要求为各功能部件操作顺序提供同步脉冲。

(8)功能部件。例如，ALU 在微操作控制信号作用下执行指令规定的运算，形成状态条件，如进位等。

2.工作过程

微程序控制器的工作过程实质上就是计算机在微程序控制器的控制下，执行机器指令的过程，具体如下：

(1)从控制存储器中运行取指令微程序，完成从主存储器中取得机器指令的工作。

(2)根据机器指令的操作码，得到相应机器指令的微程序入口。

(3)逐条取出微指令，完成相关微操作控制。

(4)执行下一条机器指令。

微程序是机器设计者事先编制好的，是支持实现工作程序的一种硬件手段。实质上，它是机器指令的解释器，对于一台具体的机器来说，按照确定的指令系统，所对应的微程序是固定的，也是有限的。

3.微指令的执行方式

执行一条微指令的过程与执行一条机器指令的过程类似。首先，将微指令从控制存储

器中取出，然后执行微指令所规定的各种操作。根据微指令的执行方式可分为串行执行和并行执行两种方式。

1）串行执行方式

采用串行方式，取微指令和执行微指令却是顺序执行的，即现行微指令执行完毕，再从控制存储器中读取下一条微指令执行。串行方式的微周期较长、工作效率低，但控制简单，因为在每个微周期中总要等到所有微操作都结束并建立了运算结果的状态之后，才确定后继微指令地址。微指令的串行操作过程如图 5-13 所示。

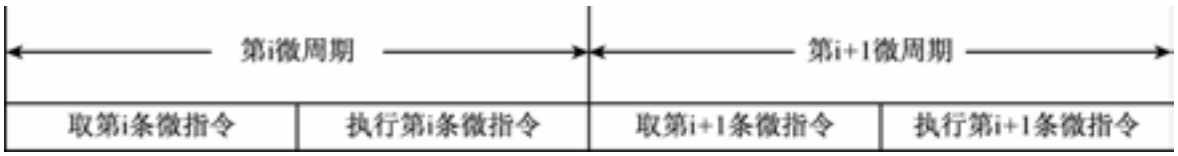

图 5-13　微指令的串行操作过程

2）并行执行方式

微周期长短是影响微程序控制速度最基本的因素。只要从机器系统结构及控制设计上缩短微周期，就能提高微程序控制效率。由于取微指令和执行微指令是在两个完全不同的部件中进行的，为了提高微指令的执行速度，可将这两部分操作在时间上重叠起来执行，以缩短微周期。即在执行现行微指令时，预取下一条微指令，现行微指令执行到适当时刻，但并不是等到它执行完毕，就开始执行已预取的下一条微指令。微指令的并行操作过程如图 5-14 所示。

图 5-14　微指令的并行操作过程

在并行执行方式中，由于执行本条微指令与取下一条微指令是同时进行的，通常在没有转移的情况下，这种并行操作是可行的。然而，如果现行微指令的后继微地址需要测试转移时，情况会比较复杂。当测试转移条件不依赖于现行微指令执行结果时，预取的微指令是正确的，当转移条件依赖于现行微指令的执行结果时，就必须在微指令执行结束后才能形成正确的后继转移地址，故在这种情况下需要等待。这是现代 CPU 设计中流水线不宜为提高工作效率而划分过细的原因。

这种工作方式的目的在于进一步使机器各部件更多地并行操作，但控制的逻辑组织相应地也复杂得多。这种工作方式给现代 CPU 带来的好处是：CPU 几乎可以在一个微周期内完成一条指令。

4.模型计算机微程序控制器

依据前面的内容，下面给出模型机微操作及微程序流程图，如图 5-15 所示。图 5-15 中给出了 4 种寻址方式在模型机中的操作过程和一些指令的执行过程。计算机一开始总是在 PC 控制下，从存储器中取出指令然后放入 IR，经微控制器执行存放于控制存储器中的微程序完成指令。从图 5-15 中可以看到当指令结束时，重新回到控制存储器 01 单元，取下一条指令。

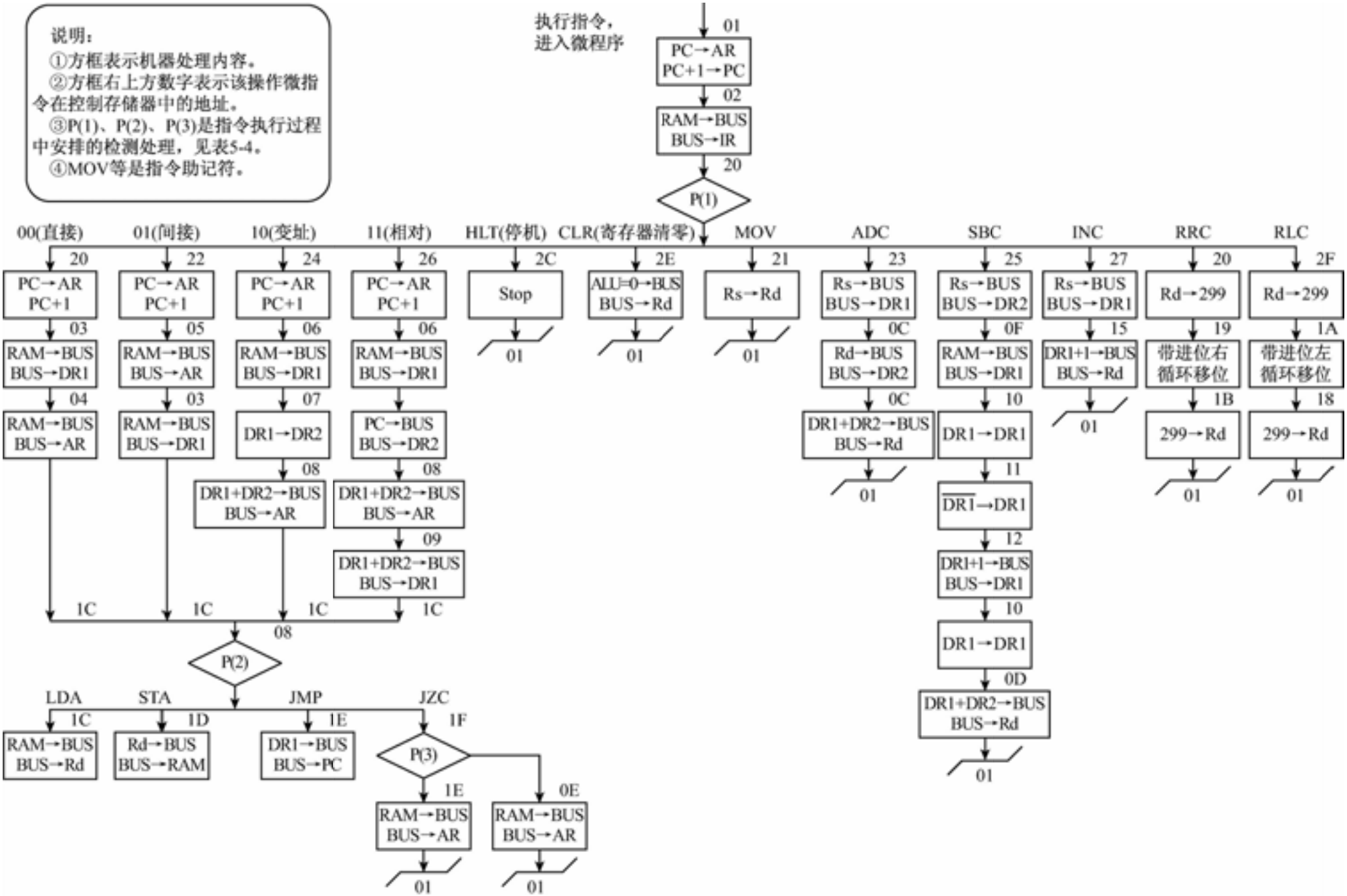

图 5-15　模型机微操作及微程序流程图

图 5-15 中每一个框图给出了具体微操作内容，可以根据微操作内容及模型计算机硬件结构，得出模型计算机硬件各个控制点的控制信号。假定各个控制信号为 1 时有效，把各个框图的操作内容所需的控制信号存放在一个控制存储器中，就完成了该框图微操作所需的微指令。在模型计算机中使用了 32 位的存储器作为控制存储器，用于存放控制信号，微指令格式及内容说明如表 5-2 所示。

表 5-2 微指令格式及内容说明

M25	M24	M23	M22	M21	M20	M19	M18
C	B	A	AR	保留	PX3	A9	A8
M17	M16	M15	M14	M13	M12	M11	M10
CE	LOAD	CN	M	S0	S1	S2	S3
M9	M8	M7	M6	M5	M4	M3	M2
PX2	LDAR	LDPC	LDIR	LDDR2	LDDR1	LDR0	WE
						M1	M0
UA0	UA1	UA2	UA3	UA4	UA5	PX1	SWB

在图 5-15 中，微指令采用了直接控制和字段编译混合方式。M19～M2 是直接控制，有两个字段要译码，包括了 CBA 字段和 PX 字段，具体含义如表 5-3 和表 5-4 所示。UA0～UA5 是下地址字段，是该微指令执行完毕后下一条微指令的地址。

表 5-3 CBA 字段操作

C	B	A	选　　择
0	0	0	—
0	0	1	PC→BUS
0	1	0	ALU→BUS
0	1	1	299→BUS
1	0	0	Rs→BUS
1	0	1	Rd→BUS
1	1	0	R1→BUS

表 5-4 PX 字段操作

M9	M1	选　　择	测 试 字
PX2	PX1		
0	0	—	关闭测试
0	1	P(1)	识别操作码
1	0	P(2)	判寻址方式
1	1	P(3)	条件测试

下面以 PC→AR，PC←PC＋1 为例，介绍微指令编码。

依据模型机的控制点，该操作要求 PC→BUS，LDAR；LDPC 及下一步的微地址。按照图 5-15 和表 5-3 以及表 5-4 有：CBA＝001，完成 PC→BUS，LDAR 完成 BUS→AR，给出 LDPC 完成 PC←PC＋1。把它们填入图 5-15，则有：

00100000，00000000，01100000，01000000 即 20H，00H，60H，40H。

同理可以得到框图 RAM→BUS，BUS→IR 的微指令：

00000000，10000000，00010000，00000110 即 00H，80H，10H，06H。

将上述分析应用于图 5-14 的每一个微操作，可得每一个微操作框图的微指令码，将这些微指令码写入控制存储器中，完成模型机的控制存储器。在控制存储器及时序部件的配合下就可以完成指令的执行。

【例 5-1】 以“ADC R2，R1”为例，说明指令的执行过程。

解：该指令完成 R1＋R2→R2，由微程序流程图可知微指令存储序列为 01H，02H，20H，23H，0CH，0DH，执行后完成“ADC R2，R1”，并回到 01H 开始下一条指令。01H～0DH 内容如下：

```
01H:20H,00H,60H,40H
02H:00H,80H,10H,06H      ;取指令
20H:20H,00H,60H,C0H      ;P(1)事务处理,条件不满足,向下执行
23H:80H,00H,04H,30H      ;R2→BUS→DR1
0CH:A0H,00H,08H,B0H      ;R1→BUS→DR2
0DH:40H,29H,02H,80H      ;DR1＋DR2→R2
```

由【例 5-1】可知，指令在微程序控制方式下的指令执行中硬件的操作过程，通过微程序执行 01H～0DH 的微指令，在时序电路的配合下各部分硬件的执行过程。此过程完成了计算机系统由机械的电子线路组成到可以用机器码编程的一步跨越，当用助记符来表示机器码时，汇编语言出现了。从此，随着人类继续进行的大量艰辛工作，在可编程的基础上，各种编译程序、操作系统相继出现。计算机在硬件和软件发展上越来越靠近人的习惯，使人们在利用计算机上越来越方便，大量的工作交由机器处理，造就了今天信息化社会的“无所不在的计算”时代，即后 PC 时代。

为了提高计算机硬件的工作速度和工作效率，人们进行了大量的工作，计算机 CPU 中采用了大量的新技术，使得硬件处理速度和效率极大地提高，从而使计算机技术得到广泛应用。

5.5　CPU 的新技术

如何加快指令的执行过程是计算机系统设计的基本任务。除了采用高速部件外，流水线控制方法也是一种常用的控制技术，其目的在于提高指令的并行性，从而加速指令的解释过程。流水线技术不只用于指令的解释过程，还广泛用于计算机系统结构的其他方面，如向量流水线处理等。计算机已广泛采用多核技术，在软件支持下可并行完成程序的进程。加快指令的执行还有多级高速缓存技术。

5.5.1 流水线技术

传统的串行执行方式不仅执行指令的周期长，同时硬件资源的利用率低，于是，引进了流水线技术。

1.基本概念

流水线技术是将一个重复的时序过程分解成若干子过程，而每一个子过程都可有效地在其专用功能段上与其他子过程同时执行。

描述流水线的工作过程，常采用时(间)—空(间)图的方法，如图 5-16 所示。

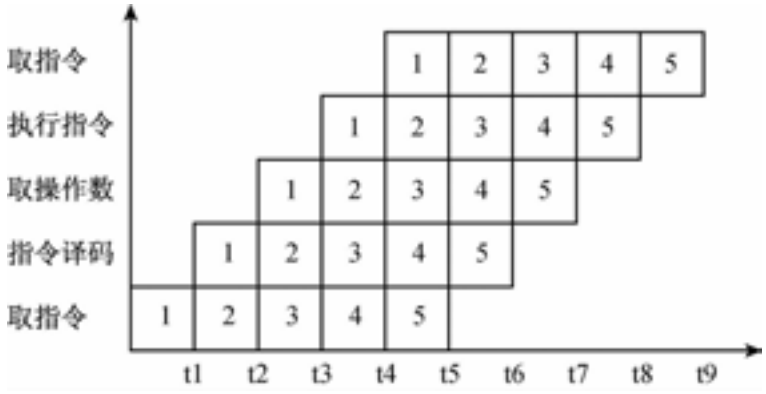

图 5-16 流水线的工作过程

如图 5-16 所示，如果执行 5 条指令，每一条指令有 4 个阶段，如果串行执行，则需 4×5＝20 个时间段 t。使用 4 级流水线，时间减为 9t。若条件理想，流水线基本上会实现每经过时间段 t 就执行完一条指令。

一般而言，流水线技术有如下特点：

(1)流水线可分成若干互有联系的子过程(功能段)。

(2)实现子过程的功能段所需时间应尽可能相等，以避免因时间不等而产生处理的瓶颈，形成流水线的断流。

(3)形成流水线处理，需要一段准备时间，称为“通过时间”，只有在此之后流水线过程才稳定。

(4)指令流不能顺序执行时，会使流水线过程中断，如果再想形成流水线过程，则需要“通过时间”，所以流水线过程不应常断流，否则效率不会很高。

为了使存储器的存取时间能与流水线的其他各过程段的速度相匹配，一般都采用多体交叉存储器。例如，IBM 360 计算机根据一个机器周期输出一条指令的要求、存储器的存取周期、CPU 访问存储器的频率，采用了模 8 交叉存储器。在现有的流水线计算机中，存储器几乎都是采用交叉存取的方式工作。

2.流水线的分类

流水线技术不仅在指令的执行过程中可以提高处理速度，还可以用于各种大量重复的时序过程，如浮点数相加等。因此就有按不同结构和不同观点产生的不同分类。

1)按完成的功能分类

(1)单功能流水线：只能完成一种固定功能的流水线，如只能实现浮点数加法运算。

(2)多功能流水线：同一个流水线可有多种连接方式来实现多种功能。

2)按同一时间内各段之间的连接方式分类

(1)静态流水线:同一时间内,流水线的各段只能按同一种功能的连接方式工作。

(2)动态流水线:同一时间内,流水线的各段可按不同运算的连接方式工作。例如,在流水线中有些段完成浮点加法,有些段实现定点乘法。

3)按级别分类

(1)部件级流水线:又称运算操作流水线,是指处理器的算术逻辑部件分段,使各种数据类型能进行流水线操作。

(2)处理器级流水线:又称指令流水线,是指在指令执行过程中划分成若干功能段,并按流水线方式组织起来。

(3)处理机间流水线:又称宏流水线,是指两台以上的处理器串行地对同一数据流进行处理,每台处理器完成一个任务。

4)按数据表示分类

(1)标量流水线:只能对标量数据进行流水线方式的处理。

(2)向量流水线:它具有向量指令,能对向量的各元素进行流水线方式的处理。

5.5.2　CPU 多核技术

多核 CPU 是指在一枚封装中集成两个或多个完整处理器内核,通过 cache 进行联系和数据交换;经特别设计的操作系统会利用所有相关的资源,将它的每个执行内核作为分立的逻辑处理器。通过在多个执行内核之间划分任务,多核处理器可在特定的时钟周期内执行更多任务。

目前双核和四核处理器之间还是有一个断层,相比四核处理器,三核处理器的成本更加低,但其效率却比双核心处理器的更高,特别是在游戏应用时,其中一个核心甚至可以进行专门的物理运算,如 AMD 三核 Pentium 处理器。多核技术能够使服务器并行处理任务,多核系统更易于扩充,并且能够在更纤巧的外形中融入更强大的处理性能,这种外形所用的功耗更低、计算功耗产生的热量更少。多核架构能够使目前的软件更出色地运行,并创建一个促进未来软件的编写更趋完善的架构。尽管认真的软件厂商还在探索全新的软件并发处理模式,随着向多核处理器的移植,现有软件无须修改就可支持多核平台。

多核处理器主要具有以下几个显著的优点:

(1)控制逻辑简单。相对超标量微处理器结构和超长指令字结构而言,单芯片多处理器结构的控制逻辑复杂性明显要低很多。相应的单芯片多处理器的硬件实现必然要简单得多。

(2)高主频。由于单芯片多处理器结构的控制逻辑相对简单,包含极少的全局信号,所以延迟对其影响比较小。因此,在同等工艺条件下,单芯片多处理器的硬件实现要比超标量微处理器和超长指令字微处理器获得更高的工作频率。

(3)低通信延迟。由于多个处理器集成在一块芯片上,且采用共享 cache 或者内存的方式,多线程的通信延迟会明显降低,这样也对存储系统提出了更高的要求。

(4)低功耗。通过动态调节电压/频率、负载优化分布等,可有效降低 CPU 功耗。

(5)设计和验证周期短。微处理器厂商一般采用现有的成熟单核处理器作为处理器核心,从而可缩短设计和验证周期,节省研发成本。

目前,虽然有一些桌面应用尚不支持多线程、多核处理器价格相对偏高、应用开发工具

不成熟等问题，但是，随着应用需求的扩大和技术的不断进步，多核必将展示出其强大的性能优势。

对于多核心的计算机，若在程序编写或编译时无法把程序线性化，就不能充分利用多核心的特色，结果程序只能在一个核心上运行，从而会浪费中央处理器的资源。

当核心到达某一个数目时，效能反而会下降。根据研究，八核心的处理器是效能的极限，当核心数目多于 8 个，其效能就会大幅下降。十六核心的处理器，其实际效能只等同于相同时钟频率的双核心处理器。

习 题 5

一、选择题

1. 下面关于同步控制方式的描述中正确的是(　　)。

A. 只适用于 CPU 控制的方式　　B. 由统一时序信号控制的方式

C. 只适用于外设控制的方式　　D. 所有指令执行时间都相同的方式

2. 异步控制方式常用于(　　)，并作为其主要控制方式。

A. 在单总线结构计算机中访问主存与外设时　　B. 微型机的 CPU 控制中

C. 组合逻辑控制的 CPU 中　　D. 微程序控制器中

3. 在一个微周期中(　　)。

A. 只能执行一个微操作

B. 能执行多个微操作，但它们一定是并行操作的

C. 能顺序执行多个微操作

D. 只能执行相斥性的操作

4. 指令周期是指(　　)。

A. CPU 从主存取出一条指令的时间

B. CPU 执行一条指令的时间

C. CPU 从主存取出一条指令加上执行这条指令的时间

D. 时钟周期时间

5. 在 CPU 中跟踪指令后继地址的寄存器是(　　)。

A. 主存地址寄存器　　B. 程序计数器

C. 指令寄存器　　D. 状态寄存器

6. 中央处理器是指(　　)。

A. 运算器　　B. 控制器

C. 运算器和控制器　　D. 运算器、控制器和主存储器

7. 计算机操作的最小时间单位是(　　)。

A. 时钟周期　　B. 指令周期

C. CPU 周期　　D. 外围设备

8. 微程序控制器中，机器指令与微指令的关系是(　　)。

A. 每一条机器指令由一条微指令来执行

B. 每一条机器指令由一段用微指令编成的微程序来解释执行

C. 一段机器指令组成的程序可由若干条微指令来执行

D. 一条微指令由若干条机器指令组成

9. 就微指令的编码方式而言，若微操作命令的个数已确定，则(　　)。

A. 直接表示法比编码表示法的微指令字长短

B. 编码表示法比直接表示法的微指令字长短

C. 编码表示法与直接表示法的微指令字长相等

D. 编码表示法与直接表示法的微指令字长关系不确定

二、名词解释

时序系统　指令周期　微指令　微周期

三、简答题

1. 微程序控制的基本思想是什么？

2. 计算机执行指令时，微控制器的基本功能是什么？基本组成部件包括哪些？

3. 时序部件的作用是什么？

4. 在计算机进行程序存储及控制中，PC、IR、AR 和 DR 各起什么作用？能否相互代替？

5. 说明机器指令和微指令的关系。

6. 根据图 5-15，针对本章的模型机，写出机器指令“ADD R0,(R1)”在执行时所需要的微操作控制信号序列。

7. 根据图 5-15，针对本章的模型机，写出机器指令“ADD (R0),R1”在执行时所需要的微操作控制信号序列。

8. 指令和数据都存放在主存中，如何识别从主存储器中取出的是指令还是数据？

第7章　系统总线

知识目标

◎掌握总线的基本概念、结构以及分类。

◎理解总线接口的基本概念和信息传送方式。

◎了解总线常用的控制方式以及常用的总线接口。

技能目标

◎能够画出总线的结构图。

◎能够理解同步通信和异步通信的过程。

总线是将计算机系统中各个部件连接起来的信息传输通道，通过总线可以传输数据信息、地址信息、各种控制命令和状态信息。总线就像人类的神经系统一样，用于协助CPU工作。CPU通过总线实现指令读取，并实现与内存、外设之间的数据交换。在CPU、内存与外设确定的情况下，总线速度是制约计算机整体性能的关键，先进的总线技术对于解决系统瓶颈、提高整个微型计算机系统的性能有着十分重要的影响。因此，在微型计算机近40年的发展过程中，总线结构一直不断地发展变化，总线结构方式已成为衡量微型计算机性能的重要指标之一。

7.1　总线概述

从物理上来看，总线是一组传输公共信息的信号线的集合，是在计算机系统各部件之间传输地址、数据、控制和状态信息的公共通路。它由一组导线和相关的控制电路、驱动电路组成。在处理器内部的各功能部件之间、在处理器与高速缓存和主存之间、在处理器与外围设备之间等，都是通过总线连接在一起的。

7.1.1　总线的基本概念

总线速度是制约计算机整体性能的关键。为了更好地理解这句话，应该从总线主要的性能指标入手开始介绍，这也是总线所涉及的基本概念。对于最终用户来说，所关心的总线指标主要有3个：总线宽度、总线频率和传输率。

(1)总线宽度。总线宽度是指一次并行传输的信息位数。例如，PCI为32位总线，一次可以传输32位数。一般来说，总线的宽度越宽，在一定时间内传输的信息量越大。但是，在

一个系统中，总线宽度不会超过 CPU 的数据宽度。

(2)总线频率。总线频率是指总线工作时每秒能传输数据的次数。例如，ISA 的频率为 8 MHz，PCI 的频率为 33 MHz。总线的频率越高，传输速度越快。

(3)传输率。传输率是指每秒能够传输的字节数，用 MB/s 表示。传输率和总线宽度、总线频率之间的关系是：

$$传输率 = 总线宽度/8 \times 总线频率$$

例如，总线宽度为 32 bit(位)，总线频率为 100 MHz，则

$$传输率 = 32\ bit \div 8 \times 100\ MHz = 400\ MB/s$$

所以，总线的传输率与总线宽度和总线频率成正比。

7.1.2 总线的工作原理

接到总线上的部件有两种工作方式：主方式和从方式。部件工作于主方式时称为总线主部件，此时它可以控制总线并启动信息传送；工作于从方式的部件只能按照主部件的要求工作。有些部件既可以工作于主方式，也可以工作于从方式，但不能同时工作于两种方式。

总线是以分时的方法来为多个部件服务的，但在任意时刻只为某两个部件或设备所占用。当总线上的一个部件要与另一个部件进行通信时，首先应该发出总线请求信号。在某一时刻，可能会有多个部件同时要求使用总线，总线控制机构根据一定的判决原则，决定先由哪个部件使用总线。只有获得了总线控制权的部件，才能开始传送数据。此时发送信息的总线主部件分时地将信息发往总线，再由总线将这些信息同时发往各个接收信息的总线从部件。究竟哪个部件接收信息，是由获得总线控制权的总线主部件给出的地址信息经过译码之后产生的控制信号来决定的。

对于同时出现的数据传输请求，总线应根据某种策略来决定先为谁服务，这个工作是由总线仲裁电路来实现的。总线仲裁电路可以根据设备的优先级别对总线请求进行排队，使发出请求的设备按其优先级的高低依次使用总线，以避免总线冲突。

7.1.3 总线的结构

在早期的计算机系统中并不采用总线结构，各个部件之间分别连接，这样，造成信息通路复杂，不易控制，也不易维护和扩展。微型计算机从诞生以来就采用了总线结构，使用总线结构以后，方便了系统的组装、维护和扩展。

具体说来，首先总线结构支持模块化设计。总线结构使得系统成为由总线连接的多个独立的子系统，每个子系统对应一个模块。这种模块化结构将系统设计分解成多个独立模块，因此整个设计可多轨并进，缩短了设计周期，而且便于故障检测和维护，也便于系统扩展。

其次，总线结构具有很好的开放性和通用性。由于每种总线都有固定的标准，而且其技术规范完全公开，所以，严格按照总线标准设计的部件都可以连接到相应的总线上，这使得大量生产商在此基础上能以很快的速度和很低的成本设计和生产大量的扩展卡，为系统功能的扩展和提高带来很好的效果。

另外，总线结构具有很好的灵活性。有了总线以后，系统的组合有一定的随意性，系统主板上有多组总线的扩展槽，每组对应于一种总线。在某一组总线槽的任何一个中插上符

合此总线规范的适配卡，就可以连接相应的设备。这个特点使用户可根据自己的需要选择设备，而且连接方便灵活。

总线技术的发展与微型计算机系统结构的发展密不可分，在很大程度上，微型计算机系统结构决定了所采用的总线结构。从计算机系统结构的观点来考察，总线结构大致可分为单总线结构、双总线结构、三总线结构和多总线结构。

1.单总线结构

在许多单处理器的计算机中，使用一条单一的系统总线来连接 CPU、主存和 I/O 设备。通过这条总线既可以访问主存，又可以进行 I/O 数据传输，所以称为单总线结构，如图 7-1 所示。

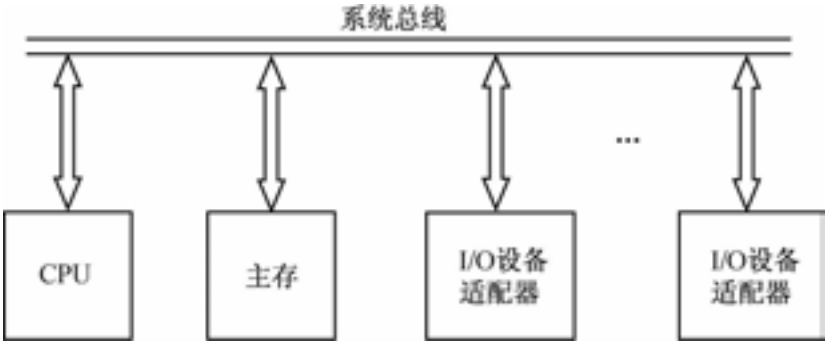

图 7-1　单总线结构

具有单总线结构的微型计算机系统的优点是：体系结构简单，软硬件开发方便，用户易于扩充系统 I/O 设备，而且在主存与 I/O 设备交换信息时还允许 CPU 继续工作。但由于微型计算机系统的所有设备均挂在单总线上，所以这种结构使得单总线只能分时工作，即同一时刻只能在两台设备之间传送数据，从而使系统总体数据传输的效率和速度受到限制，这是单总线结构的主要缺点。另外，这种结构将存储器与 I/O 设备同等看待（主存和 I/O 设备统一编址），在设计总线时就不得不考虑兼顾双方的特点，使 CPU 访问存储器时受到了很大的影响。

2.双总线结构

在单总线系统中，由于所有逻辑部件挂在同一个总线上，所以总线只能分时工作，即在某一时间只能允许一对部件之间传送数据，这使信息传送的吞吐量受到限制。为此出现了如图 7-2 所示的双总线结构。这种结构保持了单总线结构简单、易于扩充的优点，但又在 CPU 和主存之间增加了一组高速的存储总线。

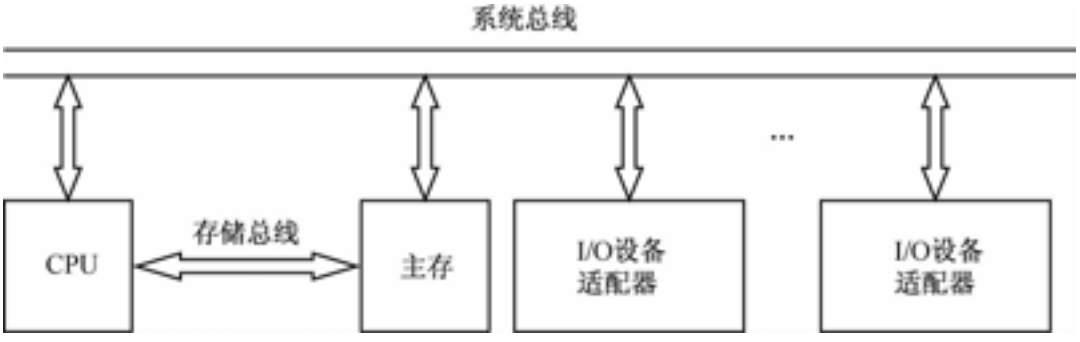

图 7-2　双总线结构

存储总线的设置可使 CPU 通过专用总线与主存交换数据，解决了高速主存访问与慢速

I/O 设备之间的差别，并减轻了系统总线的负担，同时主存仍可通过系统总线与外部设备之间实现直接存储器存取(DMA)操作，而不必经过 CPU。当然，与单总线结构相比，双总线结构是以增加硬件为代价的。高档微型机中广泛采用双总线结构。

3.三总线结构

三总线结构如图 7-3 所示，它是在双总线结构的基础上增加了 I/O 总线形成的。其中系统总线是 CPU、IOP 之间进行数据传送的公共通路，而 I/O 总线是多个外部设备与通道之间进行数据传送的公共通路。

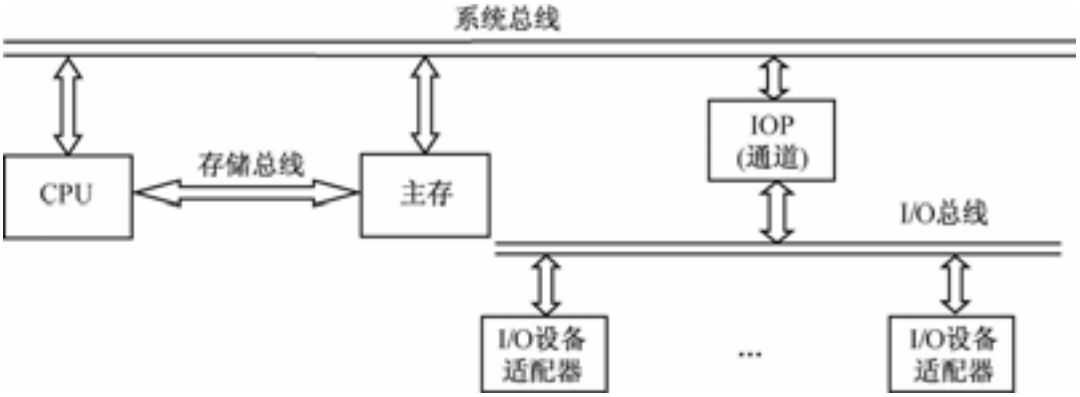

图 7-3　三总线结构

在 DMA 方式中，外设与存储器间是直接交换数据的，不经过 CPU，从而减轻了 CPU 对数据输入/输出的控制，而通道方式进一步提高了 CPU 的效率。通道实际上是一个具有特殊功能的处理器，又称为 IOP(IO 处理器)，它分担了一部分 CPU 的功能，以实现对外设的统一管理及外设与内存之间的数据传送。显然，由于增加了 IOP，整个系统的效率大大提高，然而这是以增加更多的硬件为代价换来的。三总线结构通常用在中、大型计算机中。

4.多总线结构

除了以上的总线结构外，现在的微型计算机系统中常见的还有多总线结构，即系统中拥有 3 条以上的总线。例如，Pentium 4 系统就有 3 条以上总线：前端总线、PCI、ISA、AGP 以及 USB 总线等，如图 7-4 所示。

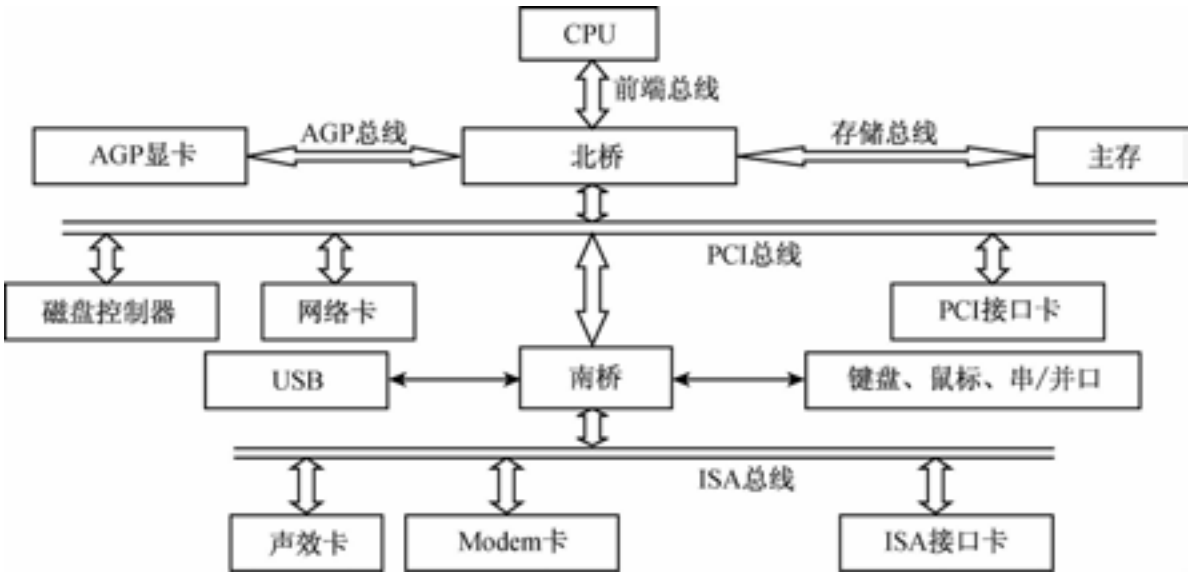

图 7-4　多总线结构

多总线结构将不同类型的设备划分到不同的总线层次中，这样做的目的是兼顾速度和成本的要求。速度越快的部件所连接的总线拥有的传输率就越高，但其生产成本很高，价格也很贵。反之，用于连接速度较低部件的总线对传输率的要求较低，这样相关电路的开发和制造就较为简单，从而降低了生产成本。

5.总线结构对系统性能的影响

在一个计算机系统中，采用哪种总线结构，往往会对计算机系统的性能产生很大影响。下面从3个方面来介绍这些影响。

(1)最大存储容量。一个计算机系统的最大存储容量似乎与总线无关，但实际上总线结构对最大存储容量也会产生一定的影响。例如，在单总线系统中，对内存和外设进行存取的差别，仅仅在于出现在总线上的地址不同，为此必须为外围设备保留某些地址。由于某些地址必须用于外围设备，所以在单总线系统中，最大存储容量必然小于由计算机地址总线所决定的地址总数。

在双总线系统中，对内存和外设进行存取的判断是利用各自的指令操作码。由于内存地址和外设地址出现在不同的总线上，所以存储容量不会受到外围设备多少的影响。

(2)指令系统。在双总线系统中，CPU对存储总线和系统总线必须有不同的指令系统。这是因为操作码规定了要使用哪一条总线，所以在双总线系统中，访问内存和输入/输出设备各有不同的指令。另外，在单总线系统中，访问内存和I/O设备使用相同的操作码，或者相同的指令，但它们使用不同的地址。

(3)吞吐量。计算机系统的吞吐量是指流入、处理和流出系统的信息的速率。它取决于信息输入内存、CPU取指令、数据从内存取出或存入，以及所得结果从内存送给一台外围设备的速率。这些步骤中的每一步都关系到内存，因此，系统吞吐量主要取决于内存的存取周期。

由于上述原因，采用双端口存储器可以增加内存的有效速度。这意味着，如果把每个端口连接到不同的总线上，那么内存可以在同一时间内对每个端口完成读操作或写操作。比起单端口来说，双端口存储器可以使更多的信息由内存输入或输出。

在三总线系统中，由于将CPU的一部分功能下放给通道，由通道对外围设备统一管理并实现外围设备与内存之间的数据传送，因而系统的吞吐能力比单总线系统强得多。

7.1.4 总线的分类

一个系统中常常包含了多种类型的总线，按照布局的范围，总线可以分为以下几类。

1.内部总线

内部总线是处于CPU内部，用来连接CPU内部的运算器和寄存器等各个功能部件的总线。由于内部总线是由芯片厂商设计的，并且封装在CPU片内，所以，大多数计算机系统设计人员和用户更加关注内部总线的对外引线，通常称这些对外引线为CPU总线。

2.局部总线

局部总线是微型机系统设计人员和应用人员最关心的一类总线，它是主机板上的信息通道，连接主机板上各个主要的部件，而且通过扩展槽连接各种适配器，如显示卡、图像卡、声卡和网卡等。局部总线的种类很多，而且不断翻新，目前，微型机系统中常用的局部总线

有 ISA(industry standard architecture)、EISA(extension industry standard architecture)和 PCI(peripheral component interconnect)。其中,PCI 是当前最先进的局部总线。实际上,局部总线是组成微型机系统的主框架,也因此备受重视。随着 CPU 的更新换代,局部总线也不断推陈出新。之所以用"局部"一词,是相对于高性能计算机系统而言的,因为在高性能超级计算机系统中,还有更高层次的总线作为系统总线。打开任何一台 PC,便可以看到主机板上并排着多个插槽,这就是局部总线扩展槽。一般情况下,一组扩展槽中的每一个扩展槽都是相同的,对应着一个局部总线,要添加某个外设来扩展系统功能时,只要在其中的任何一个扩展槽内插上该总线标准的适配器,再连接外设即可。

3.系统总线

系统总线是多处理器系统(高性能超级计算机系统)连接各 CPU 插件板的信息通道,用来支持多个 CPU 的并行处理。系统总线位于机箱底板上,各 CPU 插件板和其他总线主模块(如 DMA 模块)插件以此互相连接,并通过仲裁机制竞争对总线的控制权。在 PC 中,一般不使用系统总线。但是,在高性能超级计算机系统中,系统总线是系统设计中的关键技术。

4.外部总线

外部总线是微型机和外部设备之间或者几个微型机系统之间的通信总线。例如,微型机系统和打印机、硬盘、扫描仪之间的信息传输,就是通过外部总线实现的。外部总线的传输率低,对于不同的设备所使用总线标准也不同,常见的有串行总线 RS-232-C、专门用于硬盘连接的 IDE(integrated drive electronics)总线和 SCSI(small computer system interface)总线,用于与并行打印机连接的 Centronics 总线以及最新的通用串行总线 USB 等。这类总线不仅用于微型机系统,也用于其他系统中进行通信,所以常常被称为通信总线。

总线还有很多种不同的分类方法,按照功能的不同,总线又分为数据总线、地址总线和控制总线,分别用来传输数据、地址、命令和状态信号。

7.2 总线的控制与通信

总线是一组能为多个部件分时共享的公共信息传送线路。分时是指同一时刻总线上只能传送一个部件发送的信息,如果系统中有多个部件,它们是不能同时使用总线的。共享是指总线上可以挂接多个部件,各个部件之间相互交换的信息都可以通过这组公共线路传送。

7.2.1 总线的控制

总线是多个部件所共享的,在总线上某一时刻只能有一个总线主部件控制总线。为了正确地实现多个部件之间的通信,避免各个部件同时往总线上发送信息而造成冲突,必须有一个总线控制机构,对总线的使用进行合理分配和管理。

控制总线的部件称为主部件,主部件一旦获得总线控制权后,就立即开始向另一个部件进行一次信息传送。负责接收信息的部件称为从部件,它是与主部件进行信息交换的对象。这种以主部件为参考点,向从部件发送信息或接收从部件发来的信息的工作关系,称为主从关系。主部件负责控制和支配总线,向从部件发出命令来指定数据传送方式和地址信息。

各部件之间的主从关系不是固定不变的，只有获得总线控制权的部件才是主部件，如 CPU 和 DMA 控制器等。

根据总线控制部件的位置，控制方式可以分为集中方式和分散方式两类。总线控制逻辑集中在一处的，称为集中式总线控制；总线控制逻辑分散在总线各控制部件中的，称为分散式总线控制。下面介绍几种集中式总线控制方式。

1. 链式查询方式

链式查询方式如图 7-5 所示，总线控制器使用 3 根控制线与所有部件或设备相连。这 3 根控制线如下：

- 总线请求(BR)：该线有效，表示至少有一个部件或设备要求使用总线。
- 总线忙(BS)：该线有效，表示总线正在被某个部件或设备使用。
- 总线回答(BG)：该线有效，表示总线控制器响应总线请求。

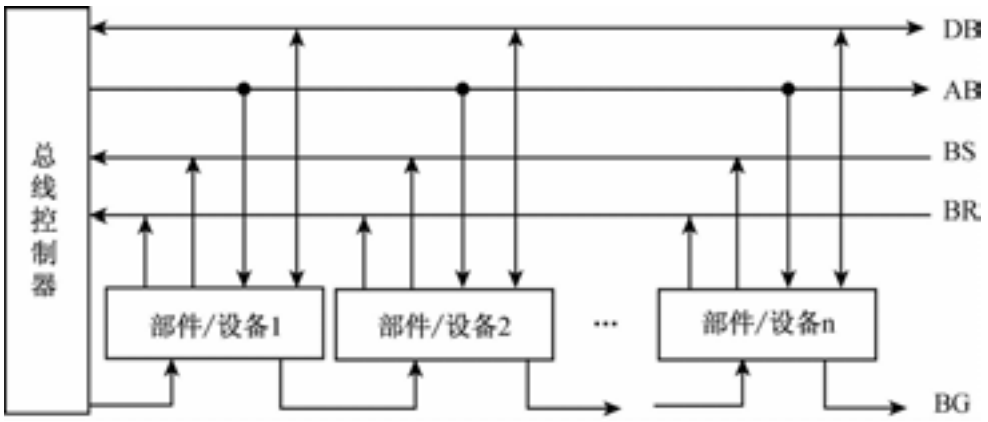

图 7-5　链式查询方式

与总线相连的所有部件经公共的 BR 线发出总线请求，只有在 BS 信号未建立之前，BR 才能被总线控制器响应，并送出 BG 回答信号。BG 信号串行通过每个部件，如果某个部件本身没有总线请求，则该信号传给下一个部件，如果这个部件有总线请求，就停止传输 BG 信号，获得总线控制权。这时该部件建立 BS 信号，表示它占用了总线，并撤销总线请求信号 BR，进行数据传送。BS 信号在数据传输完后撤销，BG 信号也随之撤销。

显然，链式查询方式的优先次序是由 BG 线上串接部件的先后位置来决定的，在查询链中，离总线控制器最近的设备具有最高优先权。

链式查询的优点是只用很少几根线就能按一定的优先次序来实现总线控制，并很容易扩充。缺点是对查询链的故障很敏感，如果第 i 个部件中的查询链电路有故障，那么第 i 个以后的部件都不能工作。另外，因为查询的优先级是固定的，所以若优先级较高的部件频繁地请求总线，优先级较低的部件就难以得到响应。

2. 计数器查询方式

计数器查询方式如图 7-6 所示。总线上任何设备要求使用总线时，都通过 BR 线发出总线请求。总线控制部件接到总线请求信号后，在 BS 线为 0 的情况下让计数器开始计数，定时查询各个部件，以确定是谁发出了请求。当查询线上的计数值与发出请求的部件号一致时，该部件就使 BS 线置 1，获得总线控制权，并终止计数查询，直至该部件完成数据传送之后，撤销 BS 信号。

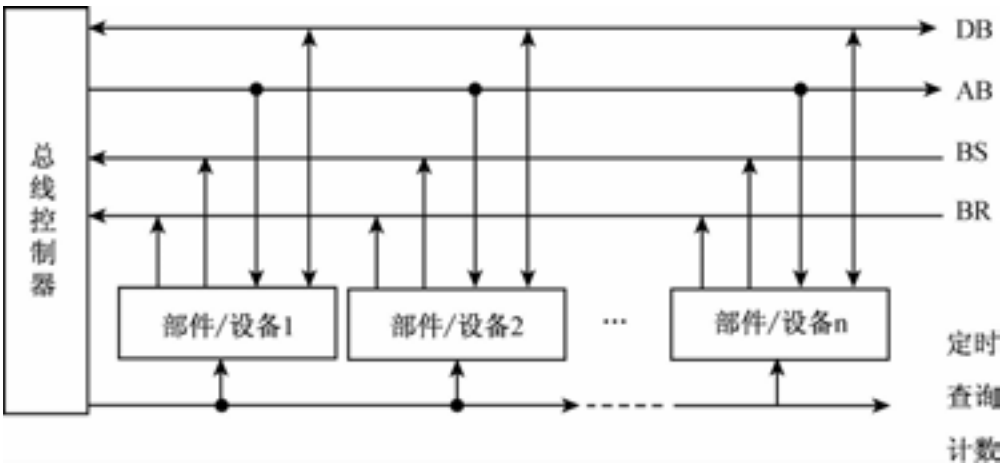

图 7-6　计数器查询方式

每次计数可以从 0 开始，也可以从上次计数的中止点开始。如果从 0 开始，各部件的优先次序和链式查询方式相同，优先级的次序是固定的。如果从中止点开始，即为循环优先级，各个部件使用总线的级别相等。计数器的初始值还可以由程序来设置，这就可以方便地改变优先次序，增加系统的灵活性。

3. 独立请求方式

独立请求方式如图 7-7 所示。在这种方式中，每一个共享总线的部件均有一对控制线：总线请求 BR_i 和总线回答 BG_i。当某个部件请求使用总线时，便发出 BR_i，总线控制器中有一个排队电路，它根据一定的优先次序决定首先响应哪个部件的请求，然后向该部件发出总线回答信号 BG_i。

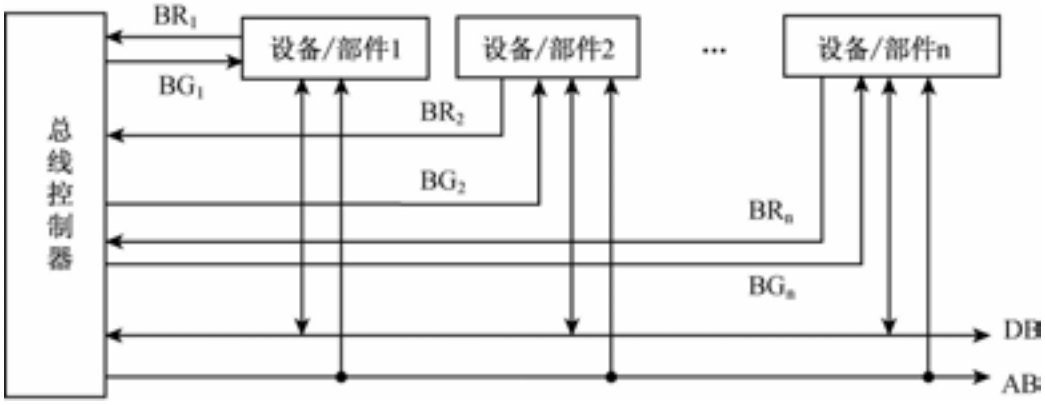

图 7-7　独立请求方式

独立请求方式的优点是响应速度快，即为确定优先响应设备所花费的时间少，不用逐个查询设备，然而这是以增加控制线数和硬件电路为代价的。此方式对优先次序的控制也是相当灵活的，它可以预先固定，也可以通过程序来改变优先次序。

对比以上 3 种方式，可见链式查询方式所需的控制线最少，仅用两根线就能确定总线控制权属于哪个设备；独立请求方式所需的控制线最多，需要采用 2n 根线；计数器查询方式居中，对于有 n 个部件的系统，共需要 $\log_2 n$ 根定时查询计数线。

7.2.2　总线的通信

数据在总线上传送时，送出数据的部件称为源部件，接收数据的部件称为目的部件。要

确保源部件和目的部件之间的数据传送协调一致，总线上的数据传输必须由定时信号控制，定时信号使源部件和目的部件之间同步，实现两部件间的协调和配合。总线的通信方式一般分为同步方式和异步方式两种。

1. 同步方式

所谓同步方式，是指系统采用一个统一的时钟信号来协调发送和接收双方的传送定时关系。时钟产生相等的时间间隔，每个间隔构成一个总线周期。在一个总线周期中，发送和接收双方可以进行一次数据传送。由于是在规定的时间段内进行输入/输出操作，所以，发送者不必等待接收者的反应，当这个时间段结束后，就自动进行下一个操作。由于总线上的数据传输用一个统一的时钟控制，发送和接收信号都在固定的时刻发出。图 7-8 是同步总线执行写操作的定时图。数据在源端于某一固定时刻在数据准备好信号 READY 控制下发出，在目的端由数据接收信号 ACK 控制接收。

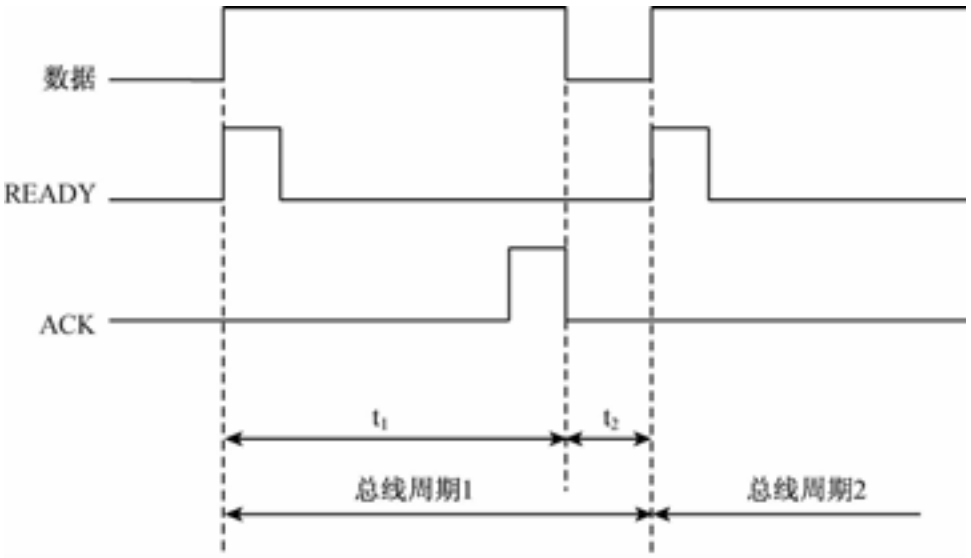

图 7-8　同步总线执行写操作的定时图

同步方式中的时钟频率必须适应在总线上最长的延迟和最慢的接口的需要。例如，在图 7-8 中，延迟时间 t_1 和 t_2 要根据连接到总线上最慢的设备来决定。因此，同步方式的效率较低，时间利用也不够合理。同步方式的另一个缺点是，源部件无法知道目的部件是否真正收到数据，目的部件也无法知道源部件的数据是否真正送至总线，故可靠性比较低。

2. 异步方式

异步方式也称为应答方式。在这种方式下，没有公用的时钟，也没有固定的时间间隔，完全依靠传送双方相互制约的“握手”信号来实现定时控制。

首先由源部件提出交换信息的“请求”信号，经接口传送到目的部件。当目的部件接收到源部件的申请后，通过接口向源部件发出“回答”信号，整个“握手”过程就是通过一问一答进行的。必须指出，从“请求”到“回答”的时间是由操作的实际时间决定的，而不是由 CPU 的节拍硬性决定的，所以具有很强的灵活性，而且对提高整个计算机系统的工作效率也是有好处的。异步方式能够保证两个工作速度相差很大的部件间可靠地进行数据交换，自动完成时间的配合。

异步方式根据“请求”和“回答”信号的撤销是否互锁，分为以下 3 种情况。

1)不互锁

“请求”信号和“回答”信号都有一定的时间宽度,“请求”信号的撤销和“回答”信号的撤销没有直接的联系,如图 7-9 所示。

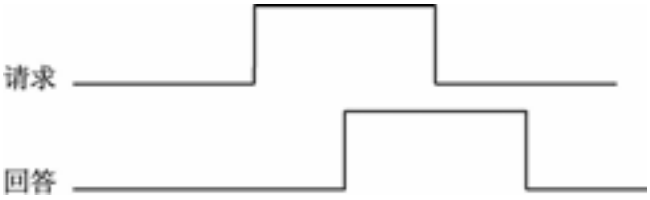

图 7-9 不互锁的情况

2)半互锁

“请求”信号的撤销取决于接收到“回答”信号,而“回答”信号的撤销由目的部件自己决定,如图 7-10 所示。

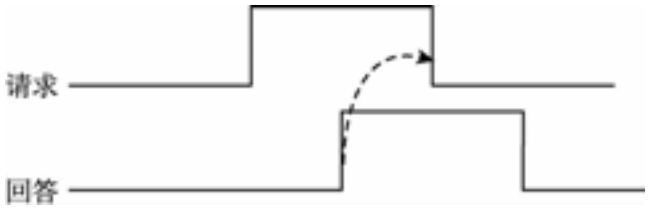

图 7-10 半互锁的情况

3)全互锁

“请求”信号的撤销取决于“回答”信号的来到,而“请求”信号的撤销又导致“回答”信号的撤销。全互锁方式(如图 7-11 所示)提供了最高的灵活性和可靠性。

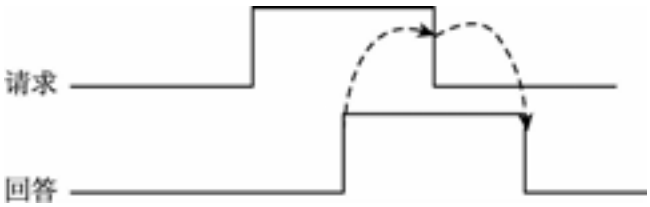

图 7-11 全互锁的情况

7.3 总线接口

任何一个微处理器都要与一定数量的部件和外围设备连接,但如果将各部件和每一种外围设备都分别用一组线路与 CPU 直接连接,那么连线将会很复杂,甚至难以实现。为了简化硬件电路设计和系统结构,常用一组线路,配置以适当的接口电路,与各部件和外围设备连接,这组共用的连接线路称为总线。采用总线结构便于部件和设备的扩充,尤其制定了统一的总线标准,使不同设备间实现互连变得容易。

7.3.1 总线接口的概念

广义上讲,“接口”这一术语是指 CPU 和内存、外围设备以及两种外围设备或两种机器

之间通过总线进行连接的逻辑器件。接口部件在它所连接的两个部件之间起着“转换器”的作用，以便实现彼此之间的信息传送。

1. 信息的传送方式

数字计算机使用二进制数，它们或用电位的高、低来表示，或用脉冲的有、无来表示。在前一种情况下，如果电位高时用数字 1 表示，那么电位低时用数字 0 表示。后一种情况下，有脉冲时用数字 1 表示，无脉冲时用数字 0 表示。

在计算机系统中，传输信息一般有 4 种方式：串行传送、并行传送、并串行传送和分时传送。出于速度和效率的考虑，系统总线上传送的信息通常采用并行传送方式。在一些微型计算机中，由于 CPU 引脚数的限制，系统总线传送信息时还采用并串行方式或分时方式。

1）串行传送

当信息以串行方式传送时，只有一条传输线，且采用脉冲信号传送。使用串行方式传送时，按顺序来传送表示一个数码的所有二进制位的脉冲信号，每次一位，通常以第一个脉冲信号表示数码的最低有效位，最后一个脉冲信号表示数码的最高有效位，如图 7-12 所示。

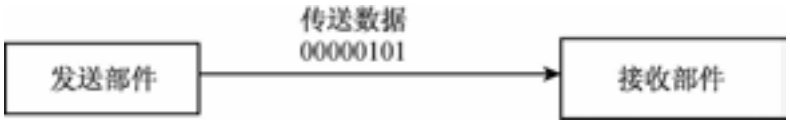

图 7-12　数据的串行传送

当串行传送数据时，有可能按顺序连续传送若干 0 或若干 1。如果在编码时用有脉冲代表二进制数 1，无脉冲代表二进制数 0，那么当连续出现几个 0 时，则表示某段时间间隔内传输线上没有脉冲信号。为了确定传送了多少个 0，必须采用某种时序格式，以便接收设备加以识别。通常采用的方式是指定“位时间”，即指定一个二进制位在传输线上占用的时间长度。显然，位时间是由同步脉冲来体现的。

假定串行数据是由位时间组成的，那么传送 8 bit 需要 8 个位时间。例如，如果接收设备在第一个位时间和第三个位时间接收到一个脉冲，而其余的 6 个位时间没有收到脉冲，那么就会知道所收到的二进制信息是 00000101。需要注意的是，串行传送时低位在前，高位在后。

串行传送的主要优点是只需要一条传输线，这一点对长距离传输显得特别重要，不管传送的数据量有多少，只需要一条传输线，成本较低。

串行传送可以分为同步串行传送和异步串行传送两种方式。

采用同步串行传送时，将许多字符组成一个信息组，这样，字符可以一个接一个地传输。但是，在每组信息的开始要加上同步字符，在没有信息要传输时，要加上空字符，因为同步传输不允许有任何间隙。在同步方式下，要求进行信息传输的双方必须使用同一个时钟进行协调，正是这个时钟确定了同步串行传送过程中每一位的位置。

采用异步串行传送时，两个字符之间的传输间隔是任意的，所以每个字符的前后都要用一些数位来作为分割位。每个串行字符由 4 个部分组成：1 个起始位、5～8 个数据位、1 个奇偶校验位、1～2 个终止位。起始位后面紧跟的是要传送字符的最低位，而每个字符的结束是一个高电平的终止位。两个相邻字符之间的间隔称为空闲位，它可以是任意长度，以便有能力处理实时的串行数据，然而下一个字符的开始必须以高电平变成低电平的起始位的前沿为标志。

2)并行传送

用并行方式传送二进制信息时,每个数据位都需要单独的一条传输线,信息由多少二进制位组成,就需要多少条传输线,从而使得二进制数 0 或 1 在不同的线上同时进行传送。如果要传送的数据由 8 个二进制位组成,那么就要使用 8 条线组成的扁平电缆,每一条线分别代表了二进制数的不同位值。

和串行传送方式相比,在同样的传输率下,并行传送的信息实际传输速度快。由于并行传送比串行传送所用的电缆要多,随着传输距离的提高,电缆的开销也会成为突出的问题,所以,并行传送总是用在数据传输率要求较高而传输距离较短的场合。

3)并串行传送

当信息在总线上以并串行方式传送时,如果一个数据字由两字节组成,那么传送一字节时采用并行方式,而字节之间采用串行方式。显然,并串行传送方式是并行方式与串行方式的结合,如图 7-13 所示。

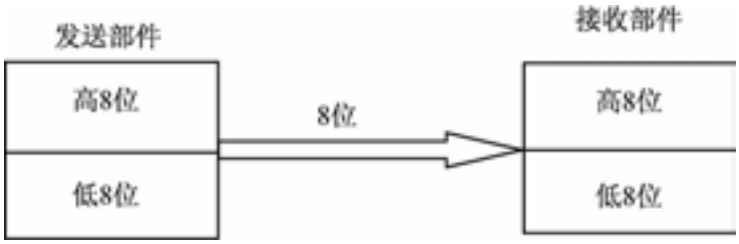

图 7-13 并串行传送

采用并串行传送信息是一种折中办法。当总线宽度(传输线数目)不是很宽时,采用并串行方式传送信息可以使问题得到较好解决。例如,在有的微型机中,CPU 内部的数据采用 16 位并行运算。但由于 CPU 芯片引脚数的限制,出入 CPU 的数据总线宽度不是 16 位而是 8 位。因此,当数据从 CPU 出入数据总线时,以字节为单位,采用并串行方式进行传送。

4)分时传送

分时传送有两种概念。一是在分时传送信息时,总线不明确区分哪些是数据线,哪些是地址线,而是统一用来传送数据或地址信息。由于传输线上既要传送地址信息,又要传送数据信息。所以必须划分时间,以便在不同的时间间隔中完成传送地址和传送数据的任务。例如,在有些微型机中,利用总线接口部件,分时在 16 位的 I/O 总线上传送数据和地址。分时传送的另一个概念是共享总线的部件分时使用总线。

2.接口的基本概念

一个典型的计算机系统具有各种类型的外围设备,因而具有各种类型的接口。CPU、接口和外围设备之间的连接关系如图 7-14 所示。外围设备本身带有自己的设备控制器,它是控制外围设备进行操作的控制部件。设备控制器通过接口接收来自 CPU 传送的各种信息,并根据设备的不同要求把这些信息传输到设备,或者从设备中读出信息传送到接口。由于外围设备种类繁多且速度不同,所以每种设备都有适应其工作特点的设备控制器。图 7-14 中将外围设备本体与它自己的控制器电路连接在一起,统称为外围设备。

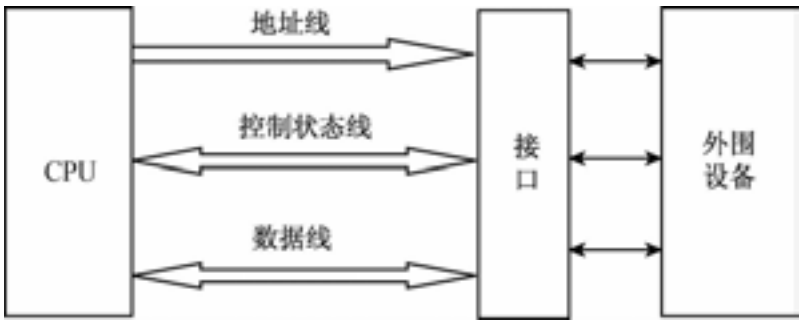

图 7-14　CPU、接口和外围设备的连接关系

为了使所有的外围设备能够兼容，并能在一起正常工作，CPU 规定了不同的信息传送控制方法。不管什么样的外围设备，只要选用某种数据传送控制方法，并按它的规定通过总线和主机连接，就可以进行信息交换。通常在总线和每个外围设备的设备控制器之间使用一个接口电路来解决这个问题，以保证外围设备用计算机系统特性所要求的形式发送和接收信息。接口逻辑通常都会标准化，对应不同的输入/输出控制方式，有不同的标准接口。当然，不同的 CPU，其标准接口也不一样。

7.3.2　常用的总线接口

随着微电子技术和计算机技术的发展，总线技术也在不断地发展和完善，而且计算机总线技术种类繁多，各具特色。下面介绍微机中主要的总线技术。

1. ISA 总线

ISA 总线是在早期 IBM-PC/XT 总线的基础上发展起来的。在早期的 PC 中，ISA 总线应用非常广泛，大多数计算机主机板上只提供 ISA 插槽。随着技术的进步，ISA 总线已逐渐被淘汰。现在，大多数 PC 主机板上只保留了一个 ISA 插槽，有些较新型的主板上甚至已不再提供 ISA 插槽。

ISA 由主槽和附加槽两部分组成，每个槽都有正反两面插脚。主槽有 62 个引脚，即 IBM-PC/XT 系统中的 62 芯总线槽，附加槽有 36 个引脚。ISA 总线在 62 芯主槽基础上增加了 36 芯附加槽后，使得数据宽度扩展为 16 位，地址线扩展为 24 位，寻址能力达到了 16 MB，工作频率为 8.33 MHz，数据传输率高达 16 MB/s，而且具有 11 个外部中断输入端和 7 个 DMA 通道。62 芯的主槽仍然可以独立使用，但只能限于 8 位数据宽度和 20 位地址。36 线附加槽增加了 8 位双向数据线、4 位地址线、6 根中断请求信号线、4 组 DMA 控制线和其他控制线。

2. EISA 总线

ISA 总线对于 286 和 386SX 等微型计算机系统来说是方便的，但对于 386DX 以上档次的具有 32 位地址和数据宽度的微型计算机系统来说，因其数据总线和地址总线宽度不够而影响了 32 位微处理器性能的发挥。为此，Compaq、HP、AST 等 9 家公司联合起来，在 ISA 总线的基础上于 1988 年推出了为 32 位微型计算机设计的“扩展工业标准总线”，即 EISA 总线。

EISA 总线在结构上与 ISA 总线有良好的兼容性，保护了厂商和用户的权益，同时又充

分发挥和利用32位微处理器的功能，使之在图形技术、光存储器、分布处理、网络、数据处理等需要高速处理能力的地方发挥作用。EISA总线可支持80486及以前的X86 CPU，但它不支持Pentium及以后的各类新型微处理器，因为从Pentium开始的微处理器已使用64位数据总线，而EISA总线并不支持64位数据总线。

EISA总线主要性能与特点如下：

(1)可支持CPU等总线主控制器的32位寻址能力和16位、32位的数据传输能力，对数据宽度具有变换功能。

(2)扩展和增强了DMA仲裁和传输能力，使DMA的数据传输率最高可达33 MB/s。EISA总线与系统主板交换数据的速率比ISA总线快4倍。

(3)可通过软件实现系统主板和扩充板的自动配置功能，无须借助DIP开关。EISA系统和ISA系统不同，任何挂接到EISA总线上的接口卡必须经配置后才能被系统所识别。EISA系统的配置是通过专门的软件(EISA configuration software)，将EISA总线上各设备的情况记录在一片8 KB×8的SRAM中，使系统对这些EISA资源进行有效的管理和使用。这个SRAM如同ISA系统中的CMOS一样，需要使用电池供电，以保持其中的内容不丢失，一旦遭遇信息丢失，或有新的EISA设备加入时，需要重新进行配置。

(4)可管理多个总线主控制器，并使用突发方式对系统存储器进行读/写访问。两个总线主控制器之间通过EISA总线也可以进行数据交换。另外，EISA总线的总线主控制器可不占用DMA通道。而ISA总线对于每一个总线主控制器都要使用一个DMA通道。

(5)可用程序来控制中断请求，采用边沿触发或电平触发方式。

(6)EISA总线插槽既可插ISA插卡，又可插EISA插卡。在插EISA卡时使用32位数据线，可达到33 MB/s的传输率。

(7)最大时钟频率为8.33 MHz。

3.PCI总线

PCI总线是一种与CPU隔离的总线结构，并能与CPU同时工作。这种总线适应能力强、速度快，数据传输率为133 MB/s，适用于Pentium以上的微型计算机。

随着图形处理技术和多媒体技术的广泛应用，计算机处理的信息除了传统的文字、图形信息外，还包括了音频和视频信息。与传统微型计算机处理的文本信息和简单的图形信息相比，音频和视频信息具有实时性强、复合性高、信息量大的特点。实时性是指播放声音、图像和视频时要求声音或画面能连续、平滑地变化。复合性是指在播放声音或图像时，声像必须保持同步与协调。与文字信息相比，音频和视频的信息量要大得多。仅仅存储1分钟的声音，其信息量可达到2～6 MB，存储1秒的视频图像，信息量高达9～10 MB。这些特点，要求现代的微型计算机应具有更快的处理速度、更大的存储空间和更高的总线带宽。事实上，微型计算机中的一些关键性部件，如CPU、内存、显卡、硬盘等经过多年来的不断改进，在性能上有了很大的提高，CPU的速度为千兆赫兹以上，硬盘与硬盘控制器之间的数据传输率可达33 MB/s以上，网卡的数据传输率达到了100 MB/s，图形控制器和显示器之间的数据传输率达到了200 MB/s以上。通常认为I/O总线的速度应为外设速度的3～5倍。显然原有的ISA、EISA总线已远远不能适应要求，成为整个系统的主要瓶颈。1991年下半年，由Intel公司首先提出的新的PCI总线标准便应运而生了。

PCI是一种先进的局部总线，不依附于某个具体处理器。从总线结构上看，PCI是在CPU和原来的ISA总线之间插入的另一级总线，具体由一个桥接电路实现对这一层的管理，并实现上下之间的接口以协调数据的传送。管理器提供了信号的缓冲，使之支持10种类型的外设接口，并能在高时钟频率下保持高性能。PCI总线也支持总线主控技术，允许智能设备在需要时取得总线控制权，以加速数据传输。

一个PCI接口包括一系列的寄存器，它们位于PCI接口上的一个小容量的存储器中，其中包含了PCI接口的信息。根据这些寄存器中的信息，计算机就可以把PCI接口自动配置到系统中。这个特性被称为即插即用特性，这也是PCI总线在新的计算机系统中变得如此流行的原因之一。

PCI总线的主要性能与特点如下：

- 总线时钟频率为33 MHz/66 MHz。
- 最大数据传输率在时钟频率为33 MHz时为133 MB/s(32位)或266 MB/s(64位)。
- 时钟同步方式。
- 总线宽度为32位/64位。
- 能自动识别外设(即插即用功能)。
- 具有与处理器和存储器子系统完全并行操作的能力。
- 支持64位寻址能力。
- 完全的多总线主控能力。

4.AGP总线

AGP总线是一种为提高视频带宽而设计的总线规范。AGP插槽可以插入符合该规范的AGP显示插卡。其视频信号的传输速率可以从PCI的133 MB/s提高到266 MB/s(1×模式)、533 MB/s(2×模式)、1 066 MB/s(4×模式)或2 133 MB/s(8×模式)。严格地说，AGP不能称为总线，因为它仅在AGP控制芯片和AGP显卡之间提供点对点的传输。

在AGP出现以前，几乎所有图形显示卡都采用了PCI总线接口。随着图形显示卡3D图形处理性能的提高，显卡处理的数据越来越多，PCI接口就逐渐暴露出它的局限性。这种局限性主要表现在3D图形描述中，储存在PCI显示卡显示内存中的不仅有影像数据，还有纹理数据(texture data)、Z轴的距离数据及Alpha变换数据等，特别是纹理数据的信息量相当大。如果要描绘细致的3D图形，就必须要求显存容量很大，而且必须采用快速的显存，从而导致显示卡价格高昂。为此，3D显卡的制造厂商期望产品既能增加纹理数据的储存能力，又能降低生产成本。一个有效的办法就是将纹理数据从显示内存移到主存，以便减少显示内存的容量，从而降低显卡的成本。从整个系统来看，增加显示内存也不如增加主存划算，因为作为主存储器的DRAM其价格已不太昂贵，而且把纹理数据储存在主存储器比储存在显示存储器可以更加有效地利用内存。存储纹理数据所需的内存空间依据应用程序而定，也就是说，当应用程序结束后，它所占用的主存空间又可恢复，纹理数据不会永远占用主存的空间。

然而遗憾的是，当纹理数据从显示内存移到主存时，由于纹理数据传输量很大，数据传输的瓶颈就从显示卡的内存总线转移到PCI总线上。例如，显示1 024×768×16位真彩色的3D图形时，纹理数据的传输速率需要200 MB/s以上，但目前的PCI总线最高数据传输

速率仅为 133 MB/s，因而成为系统的主要瓶颈。

为了解决 3D 图形数据的传输问题，主要的微型计算机生产厂商联合推出了 AGP 图形接口。AGP 在主存与显卡之间提供了直接的通道，使得 3D 图形数据可以不经过 PCI 总线而直接进入显示子系统。这样就能突破由于 PCI 总线形成的系统瓶颈，从而实现了以相对低的价格来达到高性能 3D 图形的描绘功能。所以推出 AGP 接口主要是为了大幅度提高微型计算机的 3D 图形的处理能力，或者说 AGP 是用于加速图形显示的一个专用总线接口。采用 AGP 总线的系统结构如图 7-15 所示。

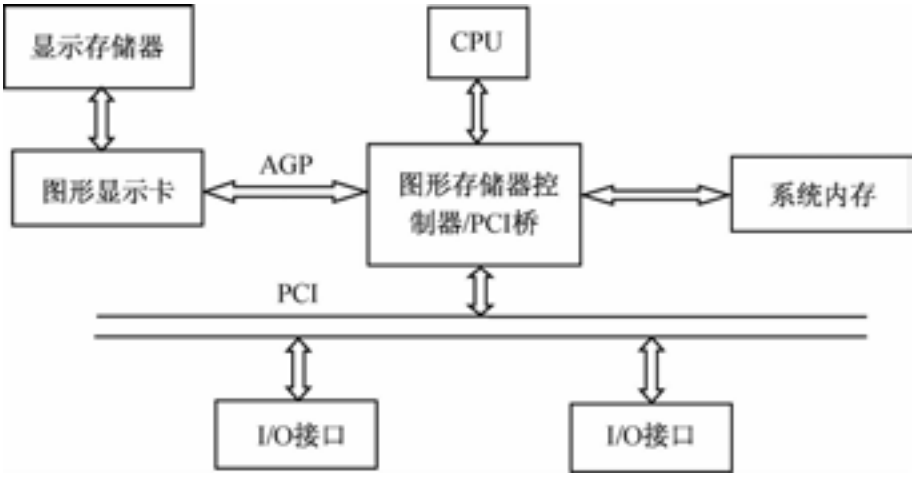

图 7-15　AGP 接口的系统结构

AGP 总线的主要性能与特点如下：

(1)数据读/写操作的流水线操作。流水线操作是 AGP 提供的仅针对主存的增强协议。由于采用了流水线操作而减少了内存等待时间，数据传输速率有了很大的提高。

(2)具有 2×、4×、8×的数据传输频率。AGP 使用了 32 位数据总线和多时钟技术的 66 MHz 时钟。因为时钟频率提高到了 66 MHz，所以带宽是 PCI 总线的两倍，达到了 266 MB/s。随后 AGP 2×问世，通过每周期传送两次 32 位数据将带宽提高到了 533 MB/s。以后又出现了每时钟周期处理 4 个 32 位数据的 AGP 4×模式，使 AGP 总线传输带宽突破 1 Gb/s，达到了 1 066 MB/s。最新的 8×模式使 AGP 带宽达到了 2 133 MB/s。

(3)直接内存执行。AGP 允许 3D 纹理数据不存入拥挤的帧缓冲区(即图形控制器内存)，而将其存入系统内存，从而让出帧缓冲区和带宽供其他功能使用。这种允许显卡直接操作主存的技术称为直接内存执行(direct memory execute，DME)。

(4)地址信号与数据信号分离。采用多路信号分离技术，并通过使用边带寻址总线来提高随机内存访问的速度。

(5)并行操作。在 CPU 访问系统 RAM 的同时允许 AGP 显示卡访问 AGP 内存，显示卡可以独享 AGP 总线带宽，从而进一步提高系统性能。

5.外部总线 RS-232-C

RS-232-C 是一种使用已久但一直保持生命力的串行总线标准。早在 1969 年，美国工业电子学会(Electronic Industries Association，EIA)和国际电报电话咨询委员会(Consultative Committee on International Telegraph and Telephony，CCITT)共同制定了 RS-232-C 标准，其传输距离可达 15 m。目前，PC 上的两个串口 COM1、COM2 接口即为 RS-232-C

接口。

当前微型机系统中，RS-232-C 接口用来连接调制解调器、串行打印机等设备。

目前，较为常用的 RS-232-C 串口有 9 针串口（DB-9）和 25 针串口（DB-25），如图 7-16 所示。

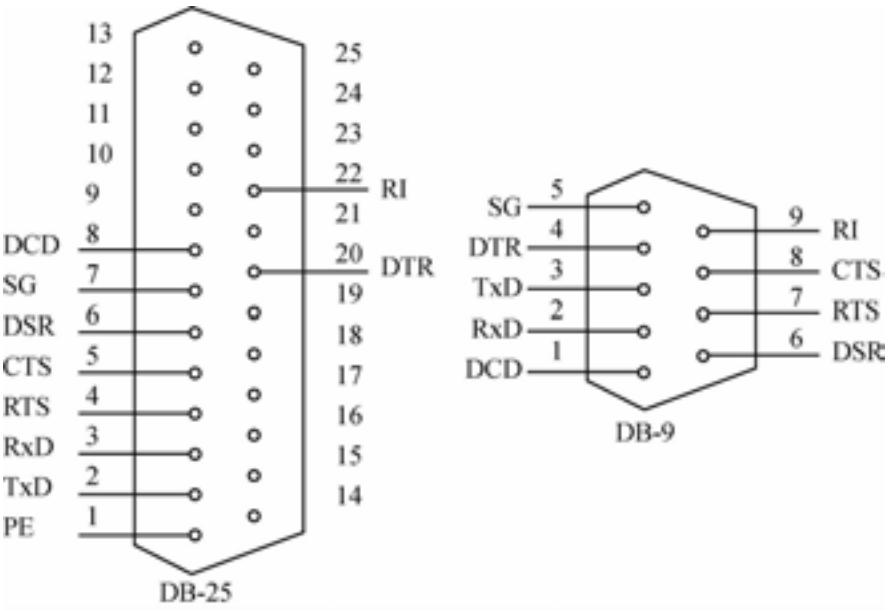

图 7-16　RS-232-C 两种常用的串口

RS-232-C 采用负逻辑规定逻辑电平，信号电平与通常的 TTL 电平也不兼容，RS-232-C 将－15～－5 V 规定为 1，＋5～＋15 V 规定为 0。

RS-232-C 是用正负电压来表示逻辑状态的，与 TTL 以高低电平表示逻辑状态的规定不同，因此，要实现串行接口与终端的 TTL 器件连接，必须在它们之间进行电平与逻辑关系的变换。一般有两种接口转换电路：图 7-17 为采用 MC1488 和 UN1489A 接口的转换电路，它使用±12 V 电压，功耗较大，不适宜在低功耗的系统中使用；图 7-18 为采用 MAX232 接口的转换电路。MAX232 接口电路包括两路驱动器和接收器。芯片内有一个电压转换器，可把输入的＋5 V 电压转换为 RS-232-C 接口所需的电压，适用于没有±12 V 的单电源系统，具有功耗低、价格低、外围电路简单等优点。

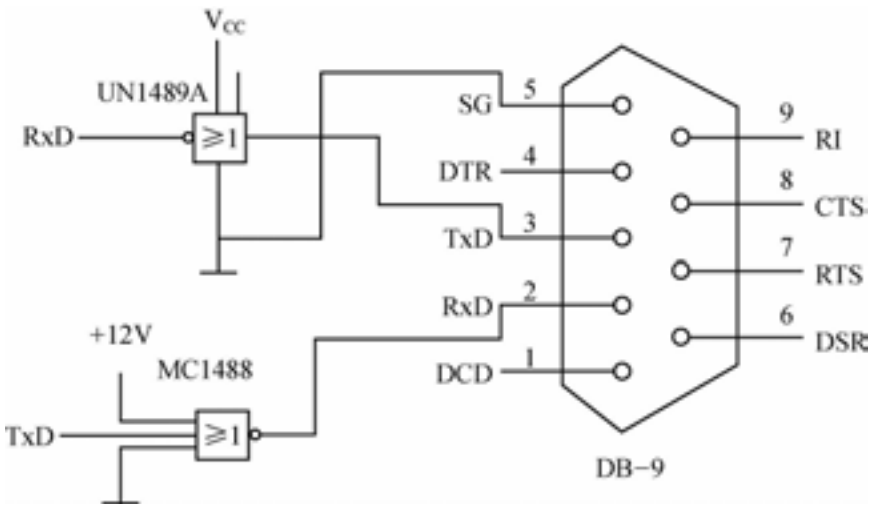

图 7-17　采用 MC1488、UN1489A 接口的转换电路

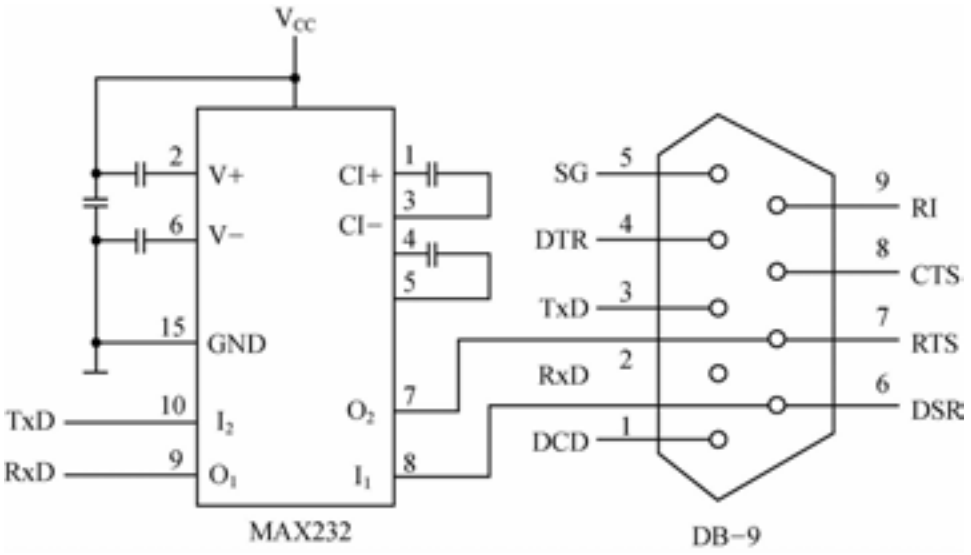

图 7-18　采用 MAX232 接口的转换电路

图 7-16 中，RS-232-C 两种常用串口部分针脚的功能如表 7-1 所示。

表 7-1　RS-232-C 常用串口部分针脚的功能

9 针串口(DB-9)			25 针串口(DB-25)		
针 脚 号	功　　能	缩　　写	针 脚 号	功　　能	缩　　写
1	数据载波检测	DCD	2	发送数据	TxD
2	接收数据	RxD	3	接收数据	RxD
3	发送数据	TxD	4	请求发送	RTS
4	数据终端准备好	DTR	5	清除发送	CTS
5	信号地	SG	6	数据设备准备好	DSR
6	数据设备准备好	DSR	7	信号地	SG
7	请求发送	RTS	8	数据载波检测	DCD
8	清除发送	CTS	20	数据终端准备好	DTR
9	振铃指示	RI	22	振铃指示	RI

6.通用串行总线

传统的计算机仅有少量的输入/输出接口，通常设置在主机箱的后面板上，用以连接键盘、鼠标、显示器与打印机等外部设备。随着计算机应用的日益广泛，需要连接的外设不断增加，而接口缺乏的矛盾日趋尖锐，于是通用串行总线(universal serial bus，USB)应运而生。USB 是 Intel、DEC、Compaq、Microsoft、IBM 等公司于 1996 年共同制定的串行接口标准，其设计初衷是串行总线能够用一个 USB 端口连接所有不带适配卡的外设，提供所谓的“万用”连接功能。同时，可以在不开机箱的情况下增减设备，支持即插即用功能。由于 Microsoft 从 Windows 98 开始加入了对 USB 的支持，使 USB 技术得到了飞速发展和极为广泛的普及。现在，USB 已成为微型计算机上普遍认同的一种事实上的接口标准，支持这一标准的各种新产品正大量涌现。

PC 接口中在很长时间内占统治地位的是串行接口和并行接口。它们有两个致命的弱点：一是传输速率低(串口传输速率最高为 115 kbit/s，并口传输速率最高为 1.5 Mbit/s)；二

是连接设备少。

USB具有以下优点。

1)自动配置

当用户连接USB外设到一个正在运行的系统时,Windows能自动检测外设,加载合适的驱动程序。外设第一次连接到系统时,Windows可能会提示用户插入驱动程序磁盘,但除此之外,其他安装步骤都是自动的。不需要定位并运行安装程序,也无须重启系统。USB不需要用户进行初始设置,如端口地址和中断请求(IRQ)线等,这给用户带来了很大的方便。

2)易于连接

有了USB,就不需要再打开计算机的机箱去为每个外设增加扩展卡。USB的连接器和电缆都有确定的规格,即使是没有经验的用户也不会接错。一个普通的PC有2～6个USB端口,需要的话,可以通过连接一个USB集线器来扩展端口的数量。集线器可以提供最多7个端口来连接更多的外设或集线器。一个USB最多可支持127个外设。USB电缆只有4根芯线:2根数据线、1根电源线和1根地线。电缆可长达5 m。通过集线器,连接还可以扩展到30 m。可以在任何时候连接和断开外设,不论系统和外设是否开机都不会损坏PC或外设。当外设连接到PC上时,操作系统会自动检测到并准备使用。另外,USB接口自带了电源线和地线,可以提供+5 V的电源。如果一个外设需要中等功率的电源供应(最多500 mA),则它完全可以从总线得到电源供应而不使用外置电源。

3)速度较快

一个全速USB接口可以以12 Mbit/s的速度进行通信。实际的数据传输率比这个数值要低一些,这是因为所有外设都共用总线,导致总线除传输数据外,还必须携带状态、控制和错误检测信号。当只有一个设备通信时,最大理论传输速率可达9.6 Mbit/s。如果这还不够快,USB2.0规范将允许以480 Mbit/s的速率传输数据。这使得USB对打印机和其他需要快速传递大容量数据的外设更具吸引力。USB也支持1.5 Mbit/s的低速传输,低速外设通常很便宜,而且它们的电缆可以更灵活,因为电缆不需要屏蔽。

4)节省硬件资源

PC上可供使用的IRQ线是一种宝贵的资源,无法给新的外设分配IRQ常常是使用USB的原因之一。如果外设都尽可能地使用USB,就可使IRQ线空闲出来供那些必须使用IRQ的外设使用。对于USB来说,它只需要若干端口地址和一根IRQ线,而挂接到USB上的设备不需要其他任何资源。对比之下,每个非USB外设都要求有自己的端口地址,通常还需要一根IRQ线,有时候还需要有一个扩展槽。

5)可靠性高

USB的可靠性来自于硬件设计和传输协议。USB驱动器、接收器和电缆的硬件规范消除了大多数可能引起数据错误的噪声。此外,USB协议采用了差错控制机制,当检测到错误时能通知发送方重发前面的数据。检测、通知和重发都是由硬件来完成的,不需要任何软件的介入。

6)成本低

虽然USB比以前的接口更复杂,但它的组件和电缆并不昂贵。带有USB接口的设备与带有相同功能的老式接口的设备所需要的费用几乎是相同的,甚至更低。对成本非常低的外设来说,可以选择低速传输以降低对硬件的需求,使成本控制在合理的范围内。

7)功耗低

当USB外设不使用时,省电电路和代码会自动关闭它的电源,但仍然能够在需要的时候作出反应。降低电源消耗的特征对于电源供应非常敏感的笔记本电脑尤其有用,还可带来保护环境的好处。

目前USB规范主要有3种:USB1.1、USB2.0和USB3.0.。USB1.1是较为普遍的USB规范,其高速方式的传输速率为12 Mbit/s,大部分MP3为此类接口类型。USB2.0是由USB1.1演变而来的,是目前使用最多的USB规范。它的传输速率达到了480 Mbit/s,足以满足大多数外设的速率要求。USB2.0定义了一个与USB1.1相兼容的架构,所有支持USB1.1的设备都可以直接在USB2.0的接口上使用而不必担心兼容性问题。USB3.0是最新的USB规范,可提供10倍于USB2.0的传输速率和更高的节能效率,可广泛用于PC外围设备和消费电子产品。

习 题 7

一、选择题

1.总线控制方式分为(　　)。

A.集中式控制和分布式控制　　B.中断控制和DMA控制

C.单向控制和双向控制　　D.并行控制和串行控制

2.在总线的链式查询方式中,所有部件都经(　　)向总线控制器发送总线请求信号。

A.总线忙　　B.总线空闲　　C.总线可用　　D.总线请求

3.在总线的链式查询方式中,部件使用总线的优先次序由它在(　　)信号线中的连接位置决定。

A.总线忙　　B.总线空闲　　C.总线可用　　D.总线请求

4.在总线的集中式定时查询方式中,若计数器每次清零,则部件使用总线的优先级次序确定为(　　)。

A.物理上离总线控制器越近,其优先级越高

B.循环优先级

C.0号部件优先级最高

D.N号部件优先级最高

5.在总线的集中式定时查询方式中,若计数器不清零,则部件使用总线的优先级次序确定为(　　)。

A.物理上离总线控制器越近,其优先级越高

B.循环优先级

C.0号部件优先级最高

D.N号部件优先级最高

二、简答题

1.什么是总线?它有哪几种类型?

2.比较单总线、双总线、三总线结构的性能特点。

3.总线数据传送有哪几种定时方式?各自的优缺点是什么?

4. 为什么要设立总线仲裁机构？集中式总线控制常用哪些方式？它们各有什么优缺点？

5. 什么是 AGP 总线？它的主要用途是什么？

6. USB 能做什么？列举日常生活中使用了 USB 总线的设备的名称。

7. 简述 USB1.1、USB2.0 和 USB3.0 的特点。