

汇 编 语 言

与

接 口 技 术

第6章 输入输出接口及数据传输控制方式

主编：王让定 朱莹

宁波大学信息学院



本章主要内容录

01



接口概述

02



CPU与外设数据的
传输控制方式



接口概述

PART 01



接口的概念

接口（Interface）包括软件接口和硬件接口

- **软件接口**

其一是软件本身的狭义“接口”，是软件不同部分之间的交互接口。比如API——应用程序编程接口

其二是指人与软件之间的交互界面——“用户界面”

- **硬件接口，通常称为I/O接口**

就是把外围设备同微型计算机连接起来的电路称为外设接口电路，简称外设接口。

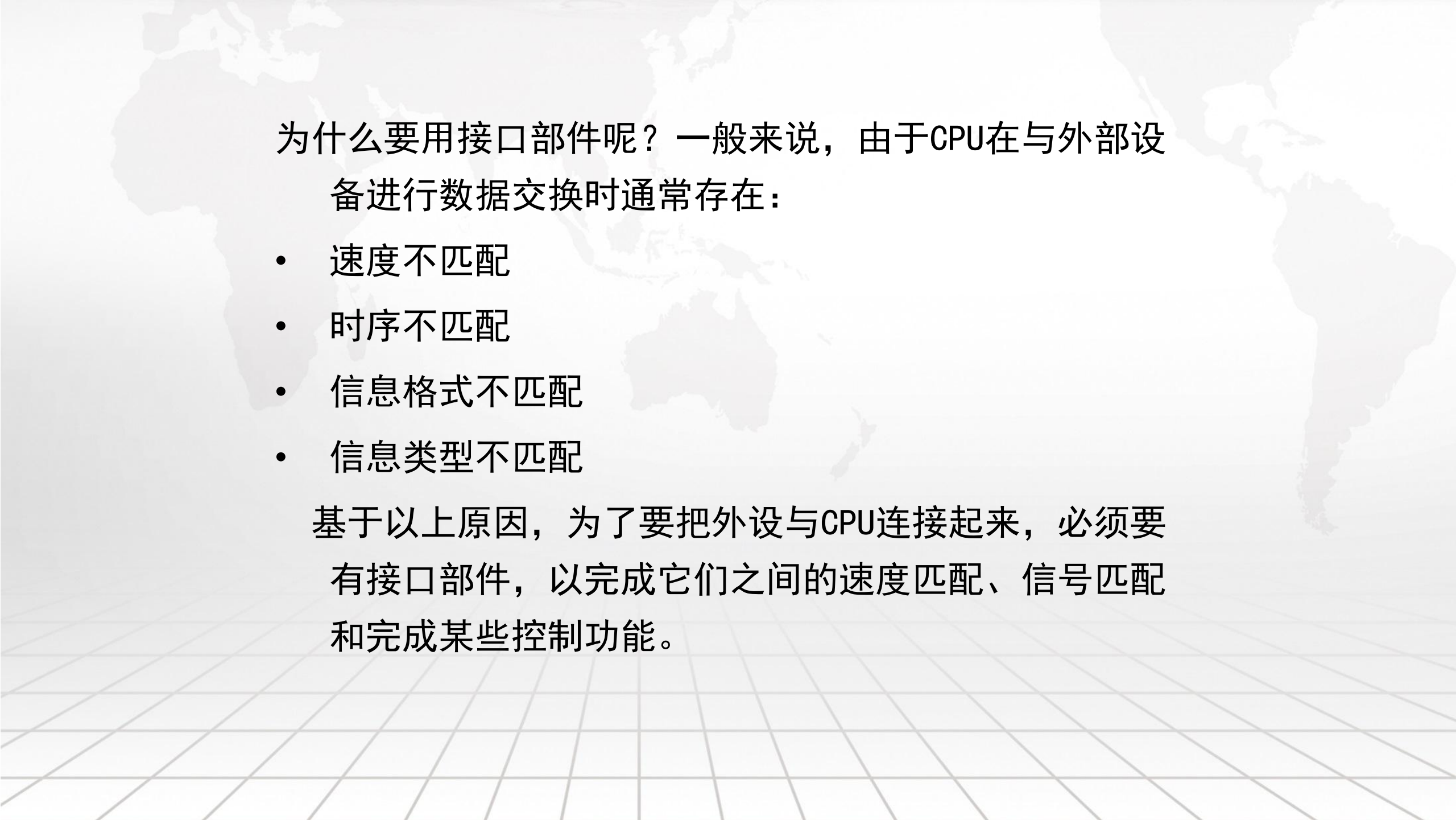


接口的功能

微型计算机是由大规模集成电路组成的、体积较小的电子计算机。

它是以微处理器为基础，配以内存储器及输入输出(I/O)接口电路和相应的辅助电路而构成的裸机。



A faint, light gray world map is visible in the background of the slide, centered behind the text.

为什么要用接口部件呢？一般来说，由于CPU在与外部设备进行数据交换时通常存在：

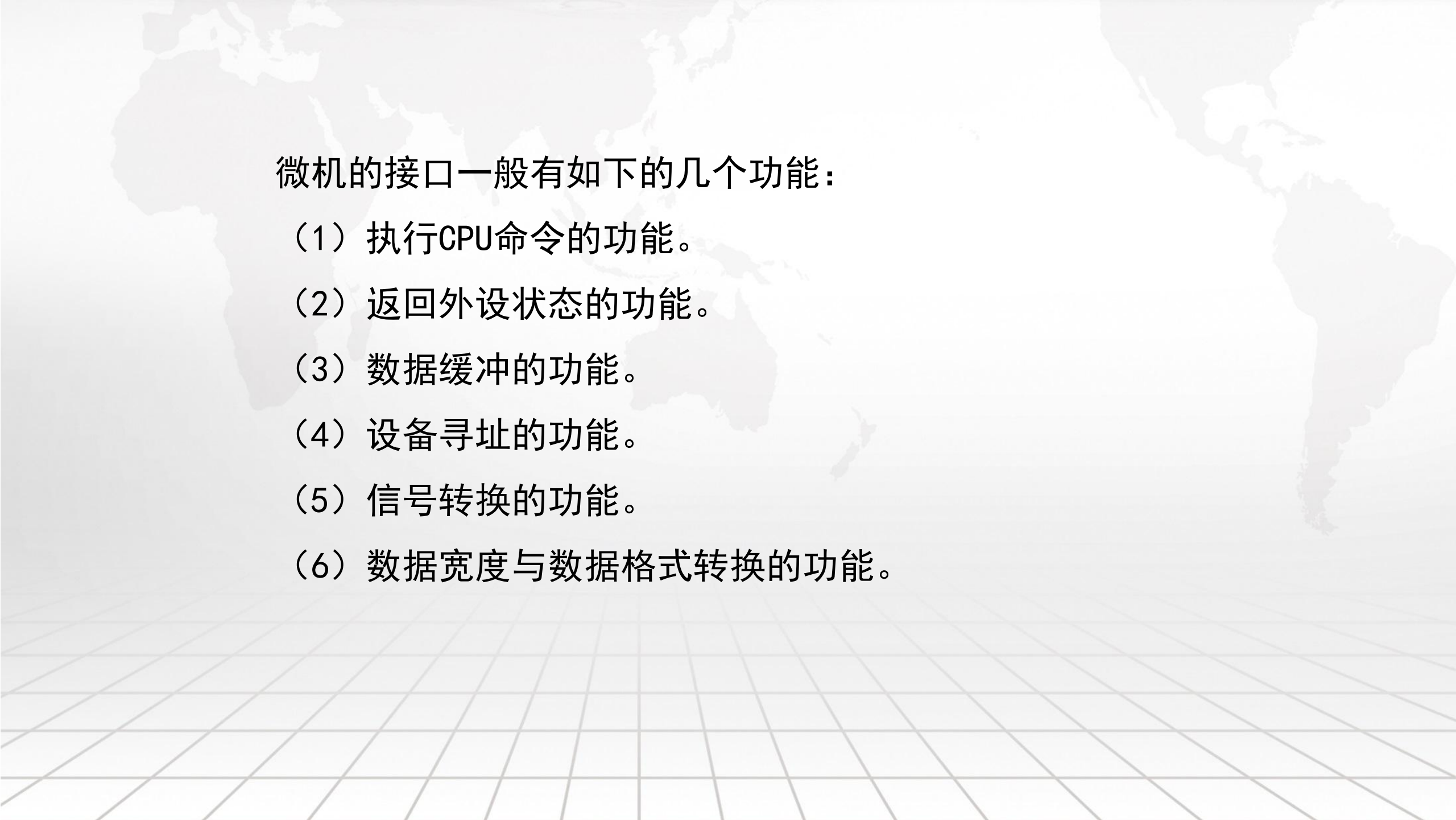
- 速度不匹配
- 时序不匹配
- 信息格式不匹配
- 信息类型不匹配

基于以上原因，为了要把外设与CPU连接起来，必须要有接口部件，以完成它们之间的速度匹配、信号匹配和完成某些控制功能。

CPU与外设之间所传送的信息类型

CPU与I/O端口之间所交换的信息，可以有下列几种类型：

- (1) 数据信息：包括数字量、模拟量、开关量等，可以输入、也可以输出。
- (2) 状态信息：这是I/O端口送给CPU的有关本端口所对应的外设当前状态的信息。供CPU进行分析、判断和决策等。
- (3) 控制信息：这是CPU送给I/O端口的控制命令，使相应的外部设备完成特定的操作。



微机的接口一般有如下的几个功能：

- (1) 执行CPU命令的功能。
- (2) 返回外设状态的功能。
- (3) 数据缓冲的功能。
- (4) 设备寻址的功能。
- (5) 信号转换的功能。
- (6) 数据宽度与数据格式转换的功能。



I/O端口及其编址方式

1. I/O端口和I/O操作

端口（Port）是指接口电路中能被微处理器直接访问的寄存器的地址。微处理器通过这些地址（即端口）向接口电路中的寄存器发送命令、读取状态和传送数据。

I/O操作就是指对I/O端口的操作，即CPU所访问的是与I/O设备相关的端口，而不是I/O设备本身。

2. I/O端口编址方式

一般来说，I/O端口有存储器映像编址和独立编址两种方式。

(1) 存储器映像编址的I/O端口

将I/O端口地址置于存储器空间，和存储单元统一编址。

优点：对端口访问非常灵活，存储器的各种寻址方式都可用来寻址端口。而且I/O接口与CPU的连接方法和存储器芯片与CPU的连接方法类似。

缺点：端口占用了一部分存储空间，而且端口地址的位数和存储器单元地址位数一样，比独立编址的I/O端口地址长，因而访问速度较慢。

(2) 独立编址的I/O端口

把接口中的端口地址单独编址。这样，在一个计算机系统中可形成两个独立的地址空间，即存储器地址空间和I/O地址空间。

优点：I/O端口地址不占用存储器空间，使用专门的I/O指令对端口进行操作，I/O指令短，执行速度快，并且由于专门I/O指令与存储器访问指令有明显的区别，使程序中I/O操作和存储器操作层次清晰，程序的可读性强。

缺点：需设置专门的I/O指令和控制信号，增加了系统的开销。



I/O端口及其地址译码

- I/O地址译码电路：指CPU为了对I/O端口进行读写操作需要把来自地址总线上的地址代码翻译成所需要访问的端口的过程。
- I/O端口地址译码的方法灵活多样，可按地址和控制信号不同的组合进行译码。一般情况下，可有两种译码方案。
 - 1) 高位地址线与CPU的控制信号进行逻辑组合，经过译码电路产生I/O接口芯片的片选信号，实现系统中的片间寻址；
 - 2) 低位地址线不参与译码，直接连接I/O接口芯片，进行I/O接口芯片的片内端口寻址，即寄存器寻址，此时，低位地址线又称为接口电路中的寄存器寻址线，低位地址线的条数决定于接口中寄存器的个数。若从系统的角度考虑，低位地址线的条数应由系统中含有寄存器数目最多的接口芯片来决定。
- I/O端口地址译码电路的形式随设计任务的复杂度而变化，一般可分为固定式单端口地址译码电路、固定式多端口地址译码电路和可选式地址译码电路。

1. 固定式单端口地址译码电路

固定式单端口地址译码电路是指该译码电路只能产生一个不可更改的端口地址。由于形式比较简单, 故一般用门电路实现。

例 假设某微处理器共有12条地址线, 即为 $A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$, 试用74LS20/30/32和74LS04设计I/O端口地址为2FFH (片选信号为低电平有效)。

解: 依题意, 要产生2FFH端口地址, 则译码电路的输入地址线 $A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$ 的值应为001011111111, 采用门电路的译码电路如图1. 4所示。

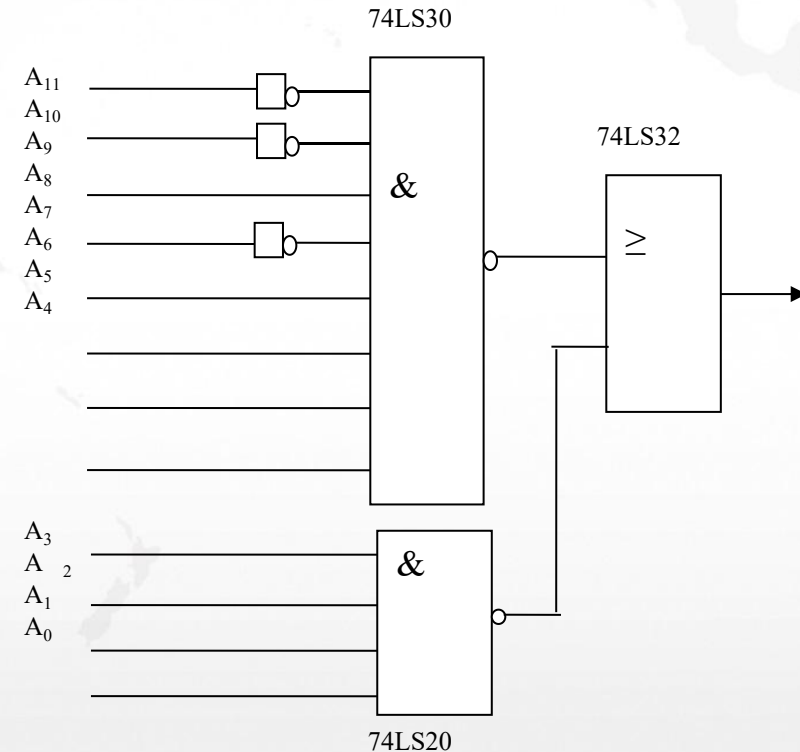


图1. 4 固定式单端口地址译码电路

2. 固定式多端口地址译码电路

固定式多端口地址译码电路能同时译出多个地址，但每个地址是固定不变的，一般采用译码器译码比较方便。译码器的型号很多，74LS138译码（3-8译码器）、74LS154（4-16译码器）和74LS139（2-4译码器）等。

3. 可选式地址译码电路

如果用户要求接口电路的端口地址能适应不同的地址分配场合，或为系统以后扩充留有余地，则可采用可选式地址译码电路。其电路可由地址开关、译码器、异或门等组成，随着PLD器件的普及，甚至可使用GAL、PAL等PLD器件来构成可选式地址译码电路。

例 分析下图所示I/O端口地址译码电路，假设该微处理器共有12条地址线。

解：依题意，若 $S_3S_2S_1S_0$ 的状态为全闭合，当CPU地址线的高4位 $A_{11}A_{10}A_9A_8$ （CPU地址线的高4位 $A_{11}A_{10}A_9A_8$ 与比较器74LS85的 $A_3A_2A_1A_0$ 引脚相连）=0000时，74LS85的A=B引脚输出为逻辑1。

由于74LS138的 G_{2A} 、 G_{2B} 引脚接CPU地址线的 A_7A_6 ，要使74LS138能正常译码， A_7A_6 必须为00，因此当比较器在A=B时，I/O端口地址对应的CPU地址线的高6位

$A_{11}A_{10}A_9A_8A_7A_6=000000$ ，此时，74LS138按 $A_5A_4A_3$ 的值进行译码。另外，由于CPU地址线的低3位没有进行连接，故这3位地址线的取值可能在000~111之间的某一个值。

依上述分析，当74LS138译码器译码后，若译码输出端为 Y_0 有效（即为低电平），表示 $A_5A_4A_3=000$ ，此时，对应的端口地址范围为可能是000H~007H；若译码输出端为 Y_5 有效，表示 $A_5A_4A_3=101$ ，此时，对应的端口地址范围为可能是028H~02FH，等等。

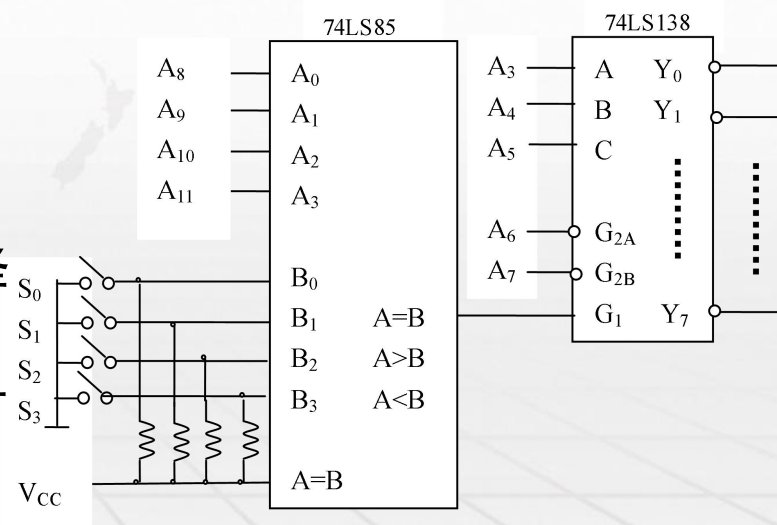


图 用比较器组成的可选式译码电路



I/O操作指令

1. I/O端口寻址

I/O端口寻址是对输入输出设备的端口地址寻址，可分为两种形式，即直接端口寻址和间接端口寻址。

(1) 直接端口寻址

由指令直接给出端口地址，端口地址范围为0 ~ 255。

例 `IN AL, 32H` ; 32H为8位端口地址

(2) 间接端口寻址

由DX寄存器指出端口地址，这种方式给出的端口地址范围为0 ~ 65535。

例 `IN AL, DX` ; DX寄存器的内容为端口寻址

2. I/O操作指令

(1) 输入指令

格式：IN 累加器，端口

功能：把一个字节/字由输入端口传送到AL/AX中。

例

IN AL, 21H ; 将端口21H的8位数读到AL

MOV DX, 201H

IN AX, DX ; 将端口201H和202H的16位数读到AX中

OUT DX, AX ; 将AX的数据输到511H和512H端口

(2) 输出指令

指令格式：OUT 端口，累加器

指令功能：把AX中的16位数或AL中的8位数输出到指定端口。

例

OUT 22H, AL ; 将AL中的数据传到端口22H

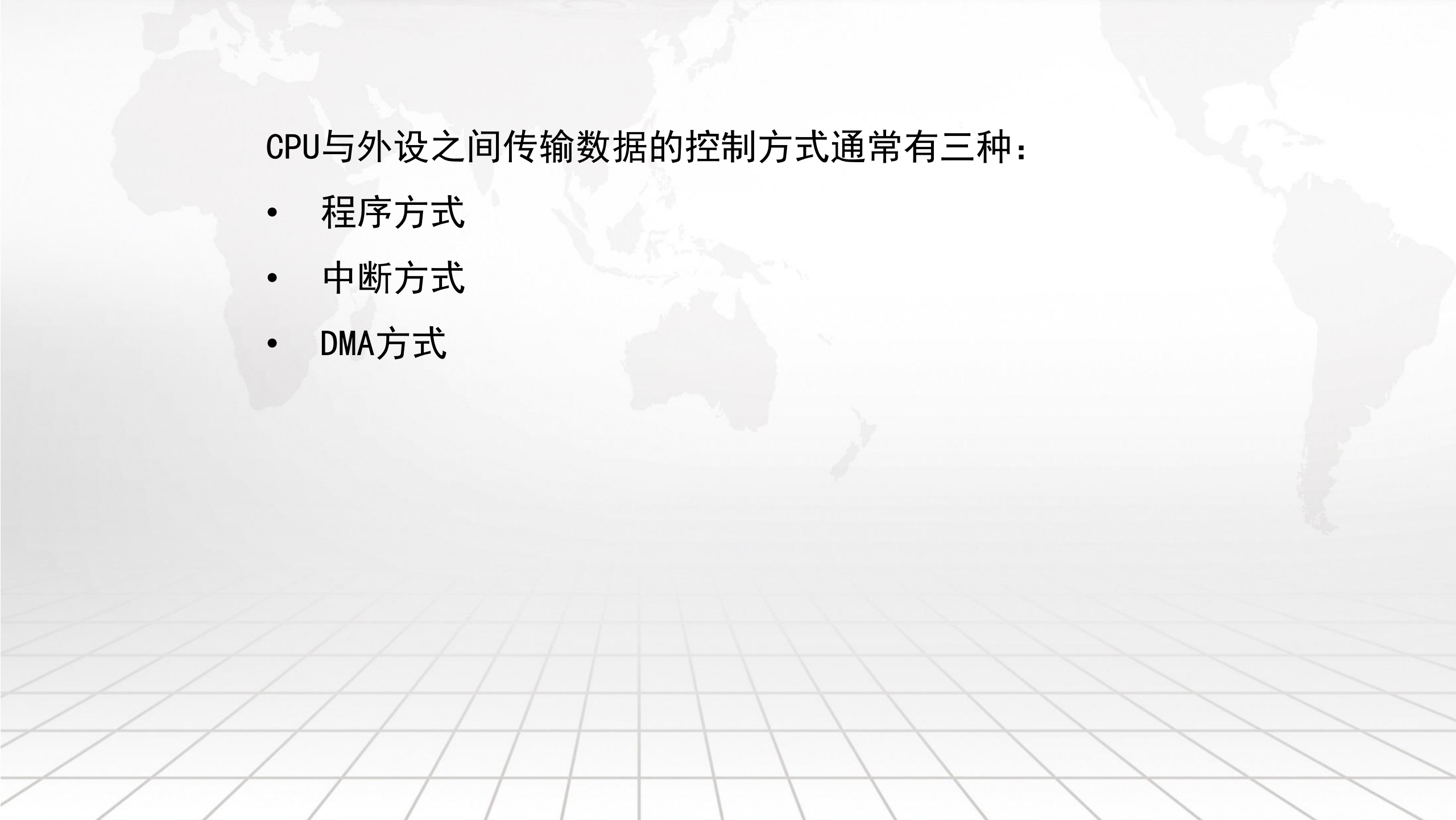
MOV DX, 511H

OUT DX, AX ; 将AX的数据输到511H和512H端口



CPU与外设数据 的传输控制方式

PART 02



CPU与外设之间传输数据的控制方式通常有三种：

- 程序方式
- 中断方式
- DMA方式



程序控制方式

(1) 无条件传送方式

适用于总是处于准备好状态的外设

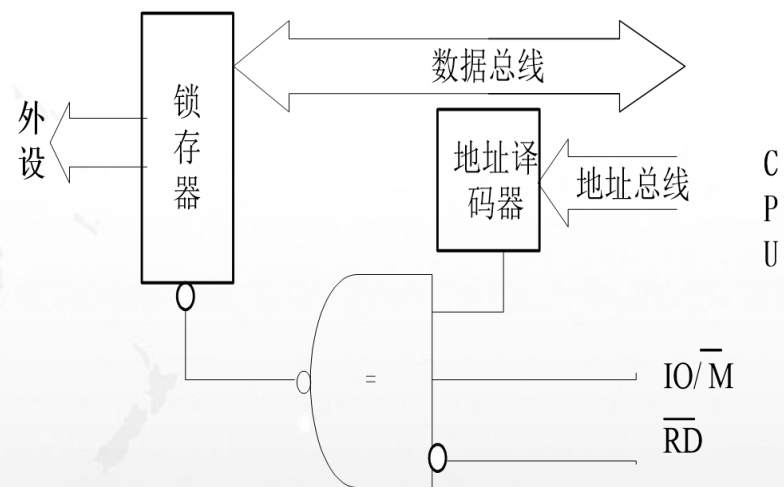
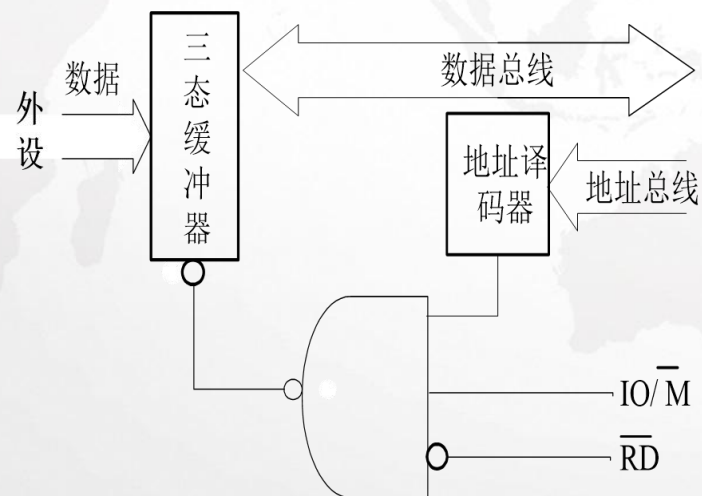
以下外设可采用无条件传送方式：

- 开关
- 发光器件(如发光二极管、7段数码管、灯泡等)
- 继电器
- 步进电机

优点：软件及接口硬件简单

缺点：只适用于简单外设，适应范围较窄

(a) 无条件传送的输入方式; (b) 无条件传送的输出方式



当进行输入时，由于数据保持时间比CPU的处理时间长，输入端必须用输入缓冲器与CPU的数据总线相连。当CPU执行输入指令时，I/O读信号IOR有效，来自输入设备的数据到达数据总线，传送给CPU。显然，CPU在执行输入指令时，要求外设的数据已经准备好，否则就会出错。

当进行输出时，由于外设速度较慢，要求接口有锁存功能，即CPU送给外设的数据应该在接口中保持一段时间。当CPU执行输出指令时，I/O写信号IOW有效，CPU输出的信息经过数据总线进入输出锁存器，输出锁存器保持这个数据，直到外设取走该数据。显然，CPU在执行输出指令时，必须保证锁存器是空闲的。 ■

从以上分析可以看出，无条件传送是最简便的传送方式，它所需的硬件和软件都较少。

(2) 查询方式

适用于外设并不总是准备好，而且对传送速率、传送效率要求不高的场合。

CPU在与外设交换数据前必须询问外设状态——“你准备好了吗？”

对外设的要求：应提供设备状态信息

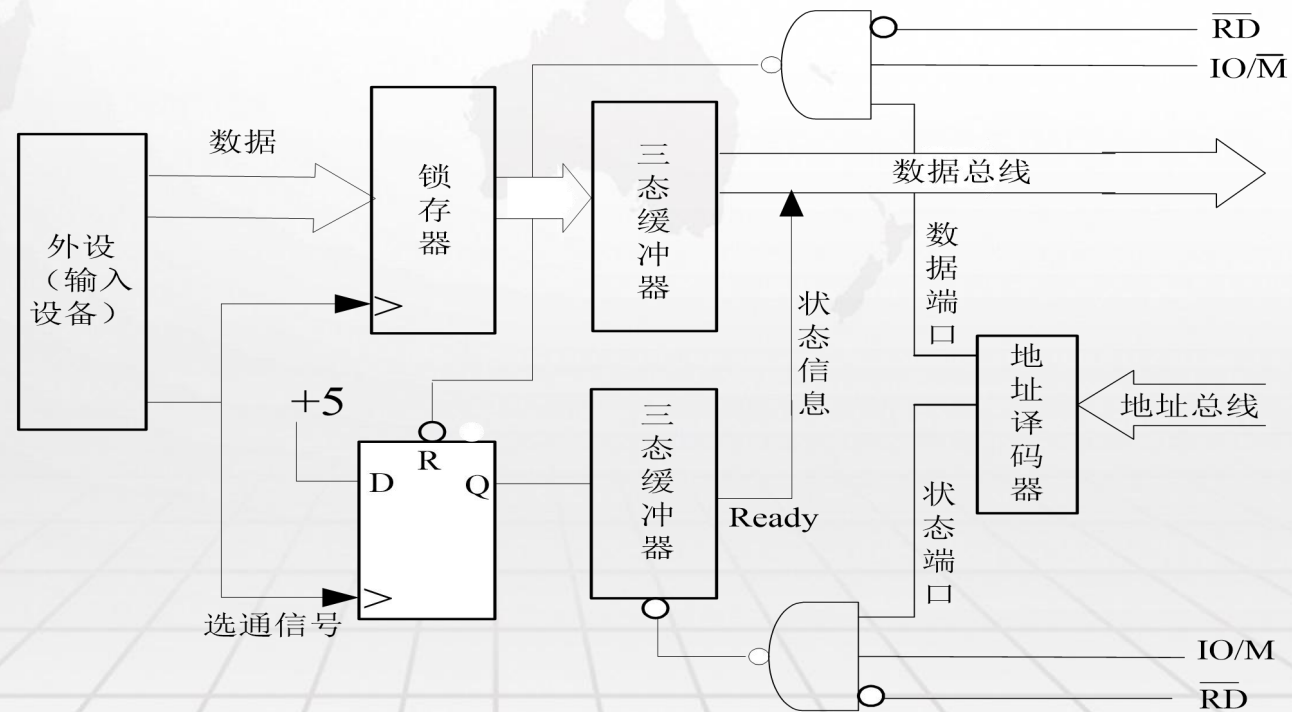
对接口的要求：需要提供状态端口

优点：软件比较简单

缺点：CPU效率低，数据传送的实时性差，速度较慢

1) 查询式输入

下图所示为查询式输入的接口电路，该电路有两个端口寄存器，即状态口寄存器和数据口寄存器。



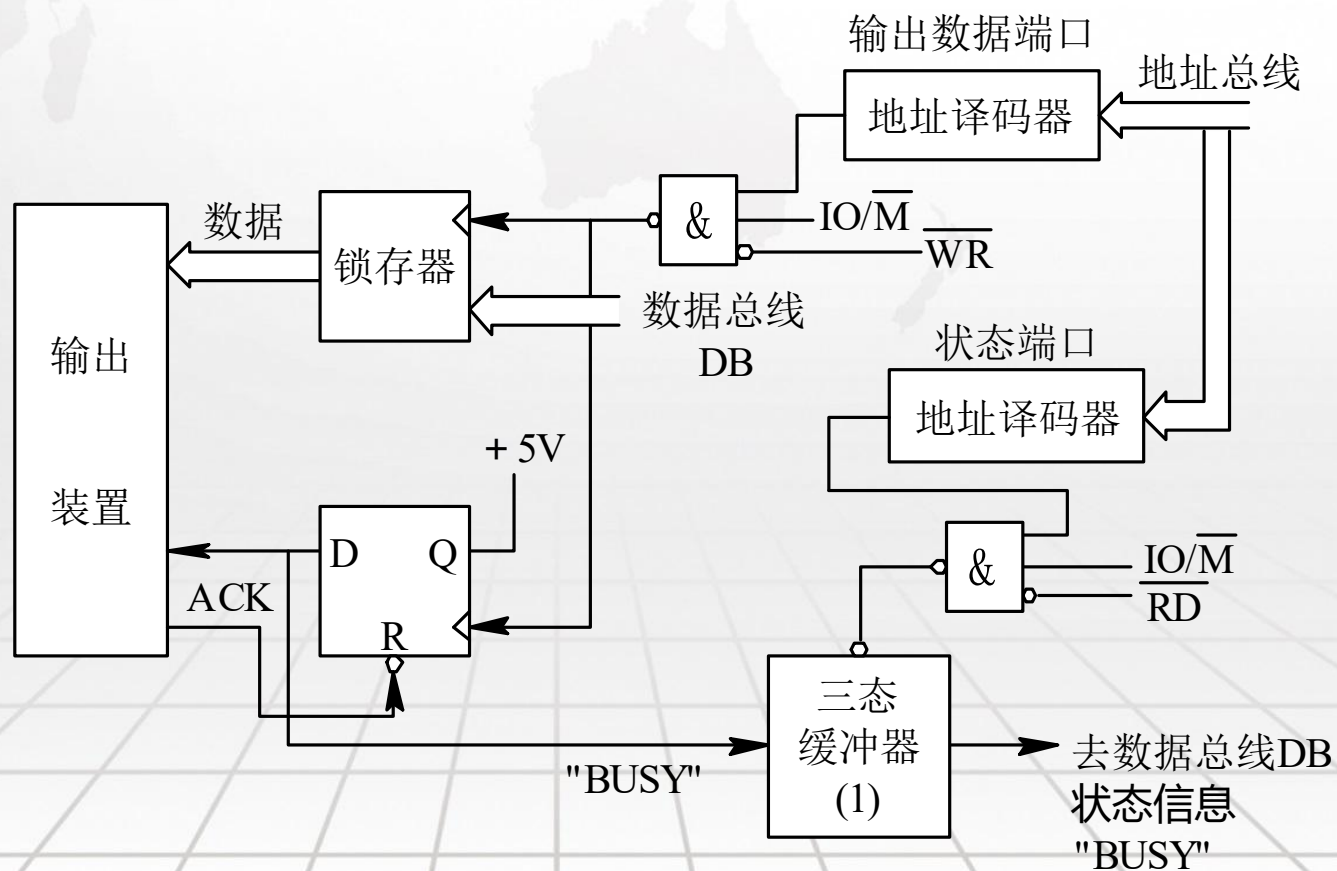
当输入设备准备好数据之后，发出选通信号。它一方面把输入数据锁存到数据锁存器中，另一方面使状态标志触发器置1。状态标志是一位信号，通过缓冲器后，接到CPU数据总线的某一位上，假设接至D7位。CPU先读状态口，查询D7是否为1。若D7=1，表示输入数据已经准备好，再读数据口，取走输入数据，同时使状态标志触发器复位。

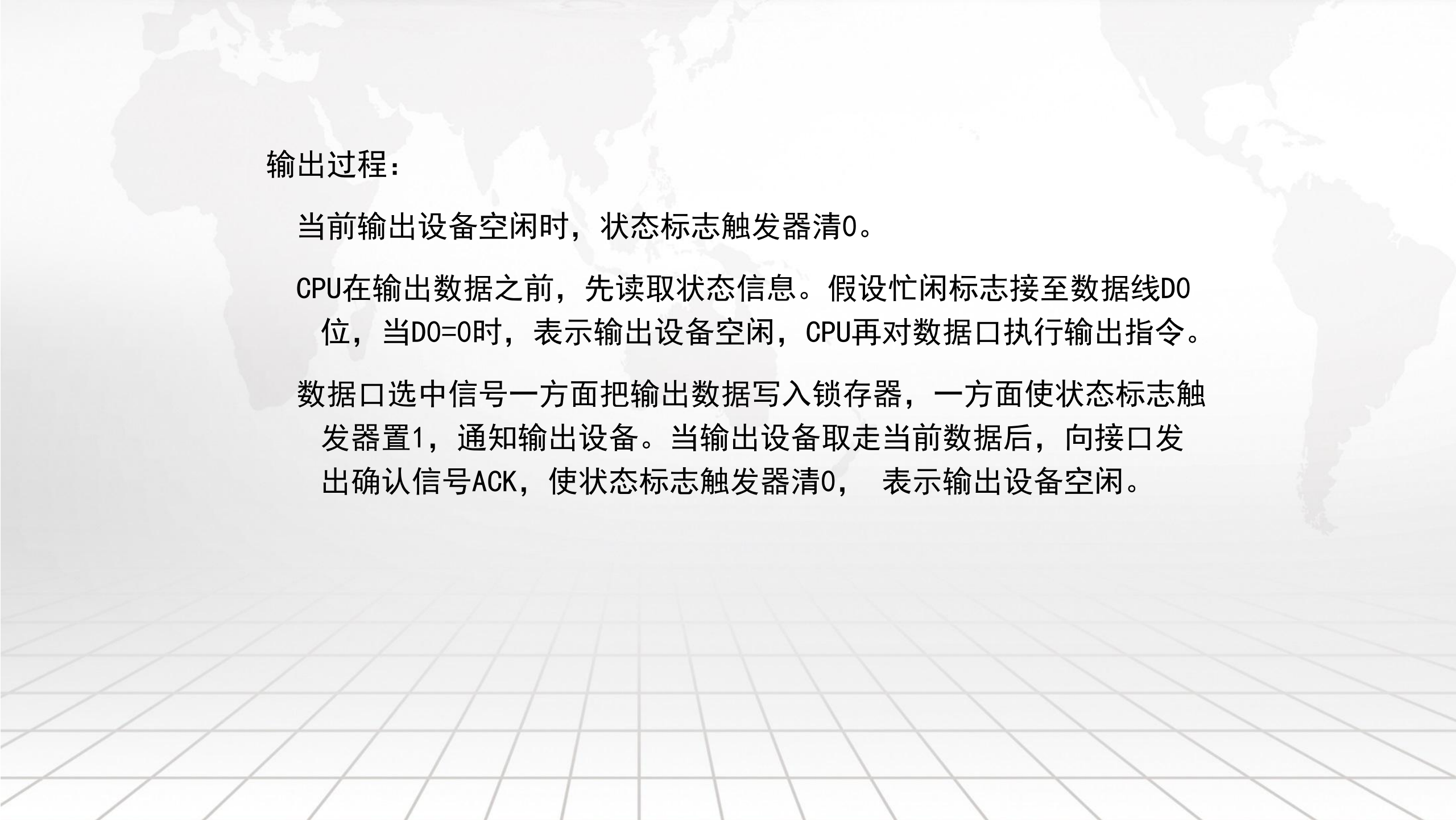
数据传送过程的3个步骤：

- ①CPU先读取状态字
- ②检查状态字是否表明数据准备就绪
- ③如果准备就绪，则执行输入指令读取数据，且使状态位清零。这样便开始下一个数据传输过程。

2) 查询式输出

当CPU要往一个外设输出数据时，先读取接口中的状态字，如果状态字表明外设有“空”或“不忙”，可以往外设输出数据，此时CPU才执行输出指令，否则CPU等待。接口电路如图：





输出过程：

当前输出设备空闲时，状态标志触发器清0。

CPU在输出数据之前，先读取状态信息。假设忙闲标志接至数据线D0位，当D0=0时，表示输出设备空闲，CPU再对数据口执行输出指令。

数据口选中信号一方面把输出数据写入锁存器，一方面使状态标志触发器置1，通知输出设备。当输出设备取走当前数据后，向接口发出确认信号ACK，使状态标志触发器清0，表示输出设备空闲。



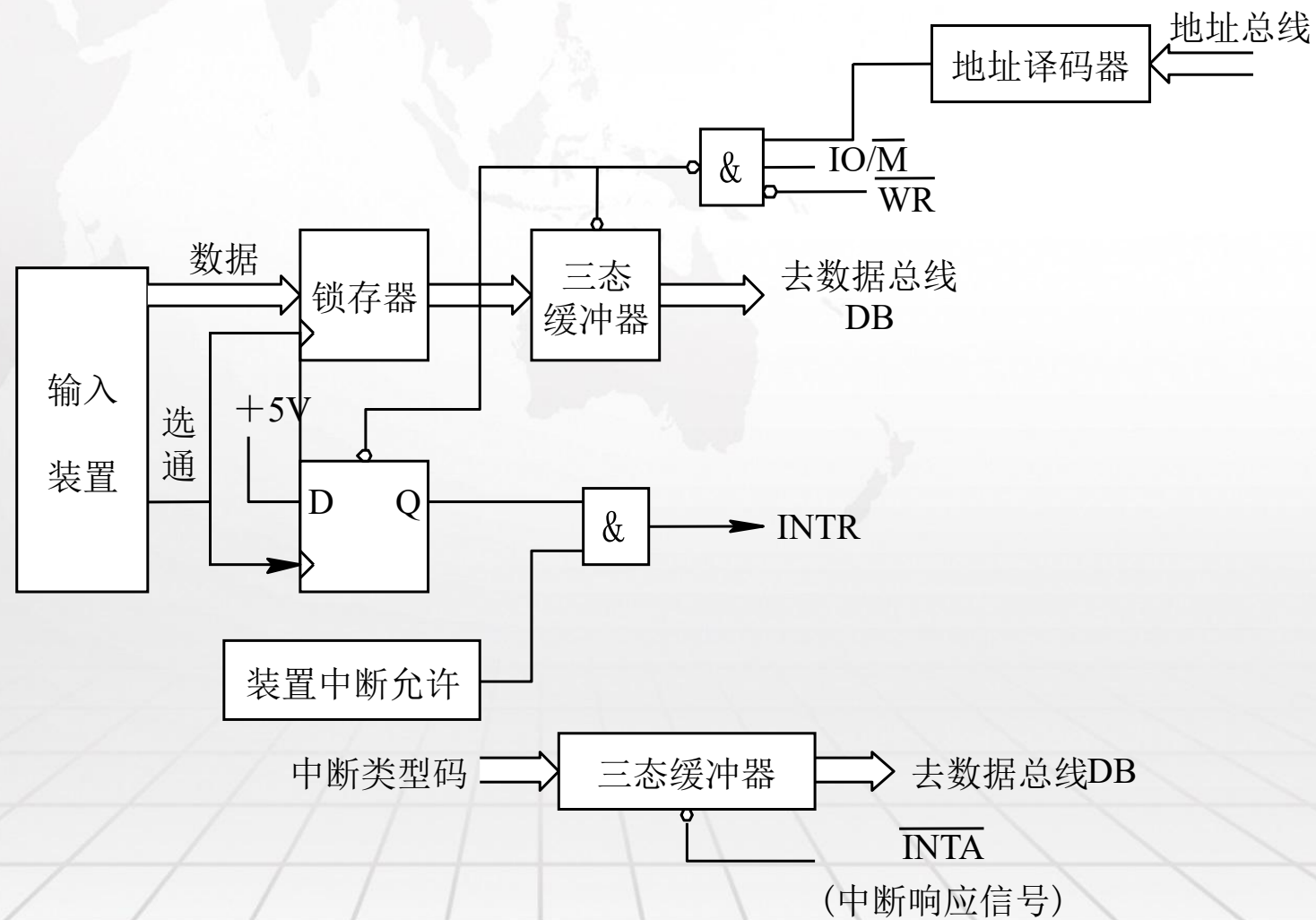
中断控制方式

2、中断控制的输入/输出

中断控制的输入和输出方式，也称中断传送方式，即当外设的输入数据准备好或接收数据的锁存器为空时，主动向CPU发出中断请求，使CPU中断原来执行的程序(主程序)，转去执行为外设服务的输入或输出操作，服务完毕，CPU再继续执行原来的程序。 ■

中断传送方式中， CPU和外设(甚至多个外设)可同时工作， 从而大大提高了CPU的效率和控制程序执行的实时性。必须经过 (1) 暂停主程序，实现程序的转移，即中断响应。(2) 保护和恢复有关寄存器内容。(3) 执行I/O操作，并实现内存到累加器再到端口之间的传送。(4) 实现中断返回。

中断传送时的接口电路如图所示：





直接存储器存取(DMA: Direct Memory Access)控制方式

数据在I/O接口与存储器之间的传送，**不经CPU的干预，而是在专用硬件电路的控制下直接传送**。这种方法称为DMA。在这种方式下，传送的速度就只取决于存储器和外设的工作速度。这大大提高了数据传送速度。

DMA传送主要应用于高速度大批量数据传送的系统中，如磁盘存取、图像处理、高速数据采集系统等，以提高数据的吞吐量。

DMA传送一般有三种形式

- ①存储器与I/O设备之间的数据传送；
- ②存储器与存储器之间的数据传送；
- ③I/O设备与I/O设备之间的传送。

为实现DMA工作方式而设计的专用接口电路，称为DMA控制器(DMAC)。

例如，Intel公司的8257、8237，Zilog公司的Z 8410(Z80 DMAC)，Motorola公司的MC6844等，都是能实现DMA方式的可编程DMAC芯片。DMA控制器必须有以下功能：

- ①能接收外设发出的DMA 请求信号，然后向CPU 发出总线接管请求信号。
- ②当CPU发出总线请求允许信号并放弃对总线的控制后，DMAC能接替对总线的控制，进入DMA方式。
- ③DMAC得到总线控制权后，要往地址总线发送地址信号，能修改地址指针，并能发出读 / 写控制信号。
- ④能决定本次DMA传送的字节数，判断DMA传送是否结束。
- ⑤DMA过程结束时，能发出DMA 结束信号，将总线控制权交还给CPU。



数据传送控制方式的发展

数据传送控制方式的发展大体上经历了四个阶段，

- 早期阶段
- 接口模块
- 中断阶段
- 通道结构阶段
- I/O处理机阶段