



# 计算机组成与体系结构

## 第8章

# 总线与输入输出系统

张健


ZJSS@BIT.EDU.CN

北京理工大学计算机学院

SCHOOL OF COMPUTER SCIENCE & TECHNOLOGY


BEIJING INSTITUTE OF TECHNOLOGY






### 第8章总线与输入输出系统

- 8.1总线设计
- 8.2外部设备
- 8.3主机与外设的连接
- 8.4输入输出信息传送控制方式
- 8.5中断系统
- 8.6 DMA的实现
- 8.7通道处理机




2025年12月




### 8.1总线设计

#### 8.1.1总线概述

总线是一组能为多个部件分时共享的公共信息传送线路。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；分时是指同一时刻总线上只能传送一个部件发送的信息。总线的优点是成本低、简单；缺点是总线的带宽形成了信息交换的瓶颈，从而限制了系统中总的I/O吞吐量。




2025年12月




### 8.1总线设计

#### 1.总线事务

通常把在总线上一对设备之间的一次信息交换过程称为一个“总线事务”，把发出总线事务请求的部件称为主设备，与主设备进行信息交换的对象称为从设备。例如CPU要求读取存储器中某单元的数据，则CPU是主设备，而存储器是从设备。总线事务类型通常根据它的操作性来定义，典型的总线事务类型有“存储器读”、“存储器写”、“I/O读”、“I/O写”、“中断响应”等，一次总线事务简单来说包括两个阶段：地址阶段和数据阶段。



2025年12月




### 8.1总线设计


#### 2.总线使用权

总线是由多个部件和设备所共享的，必须有一个总线控制机构，对总线的使用进行合理的分配和管理。

主设备发出总线请求并获得总线使用权后，就立即开始向从设备进行一次信息传送。这种以主设备为参考点，向从设备发送信息或接收从设备送来的信息的工作关系，称为主从关系。主设备负责控制和支配总线，向从设备发出命令来指定数据传送方式与数据传送地址信息。各设备之间的主从关系不是固定不变的，只有获得总线使用权的设备才是主设备，如CPU等。但主存总是从设备，因为它不会主动提出要与谁交换信息的要求。




2025年12月



### 8.1总线设计

#### 3.总线的数据宽度

数据宽度是I/O设备取得I/O总线后所传送数据的总量，它不同于前述的数据通路宽度。数据通路宽度是数据总线的物理宽度，也就是数据总线的线数。而两次分配总线期间所传送的数据宽度可能要经过多个时钟周期分次传送才能完成。数据宽度有单字（单字节）、定长块、变长块、单字加定长块和单字加变长块等。



2025年12月

8.1总线设计

4.总线的性能指标

(1)总线宽度  
总线宽度指的是总线的线数，它决定了总线所占的物理空间和成本。

(2)总线带宽  
总线带宽定义为总线的最大数据传输率，即每秒传输的字节数。在同步通信中，总线的带宽与总线时钟密不可分，总线时钟频率的高低决定了总线带宽的大小。总线的带宽公式为：  
$$B=W \times F / N$$
其中，W为数据总线宽度，通常以字节为单位；F为总线的时钟频率；N为完成一次数据传送所用的时钟周期数。

2025年12月

8.1总线设计

(3)总线负载  
总线负载指连接在总线上的最大设备数量。

(4)总线复用  
总线分时复用是指在不同时段利用总线上同一个信号线传送不同信号，例如地址总线和数据总线共用一组信号线。采用这种方式的目的是减少总线数量，提高总线的利用率。

(5)总线猝发传输  
猝发（突发）式数据传输是一种总线传输方式，即在一个总线周期中可以传输存储地址连续的多个数据。

2025年12月

8.1总线设计

8.1.2总线定时控制

1. 同步定时方式  
所谓同步定时方式，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I/O操作，所以发送者不必等待接收者有什么响应，当这个时间段结束后，就自动进行下一个操作。

2025年12月

8.1总线设计

2. 异步定时方式  
异步定时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。  
通常，把交换信息的两个部件或设备分为主设备和从设备，主设备提出交换信息的“请求”信号，经接口传送到从设备；从设备接到主设备的申请后，通过接口向主设备发出“回答”信号，整个“握手”过程就是一问一答中进行的。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的节拍硬性规定的，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。

2025年12月

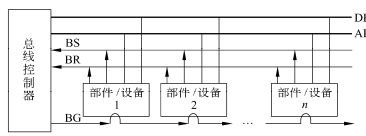
8.1总线设计

8.1.2总线仲裁

1. 链式查询方式  
链式查询方式的总线控制器使用三根控制线与所有部件和设备相连：  
•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，表示总线正在被某部件或设备使用。  
•总线批准（BG）：该线有效，表示总线控制器响应总线请求。

2025年12月

8.1总线设计



2025年12月

8.1总线设计



链式查询的优点是只用3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，所以当优先级较高的部件出现频繁的总线请求时，优先级较低的部件就可能难以得到响应。

2025年12月

8.1总线设计

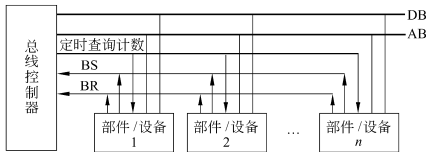


2.计数器定时查询方式

计数定时查询方式的总线上的每个部件可以通过公共的BR线发出请求，总线控制器收到请求之后，在BS为“0”的情况下，让计数器开始计数，定时地查询各个部件以确定是谁发出的请求。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”，获得了总线使用权，并中止计数查询，直至该部件完成数据传送之后，撤消BS信号。

2025年12月

8.1总线设计



2025年12月

8.1总线设计



这种计数可以从“0”开始，也可以从中止点开始。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将相等。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。定时查询方式的控制线数较多，对于*n*个部件，共需 $2 + \lceil \lg n \rceil$ 根线。

2025年12月

8.1总线设计



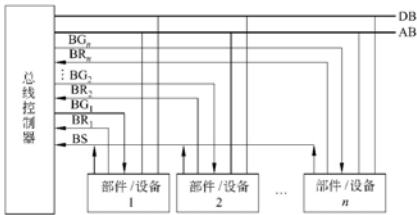
3.独立请求方式

在独立请求方式中，每一个共享总线的部件均有一对控制线：总线请求BR<sub>i</sub>和总线批准BG<sub>i</sub>。当某个部件请求使用总线时，便发出BR<sub>i</sub>，总线控制器中有一排队电路，根据一定的优先次序决定首先响应哪个部件的请求BR<sub>i</sub>，然后给该部件送回批准信号BG<sub>i</sub>。

独立请求方式的优点是响应时间快，然而这是以增加控制线数和硬件电路为代价的。对于*n*个部件，控制线的数目将达 $2n + 1$ 根。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。

2025年12月

8.1总线设计



2025年12月

第8章总线与输入输出系统



8.1 总线设计

8.2 外部设备

8.3 主机与外设的连接

8.4 输入输出信息传送控制方式

8.5 中断系统

8.6 DMA的实现

8.7 通道处理机

2025年12月

8.2 外部设备



8.2.1 外存储器

1. 硬盘存储器

(1) 硬盘的信息分布

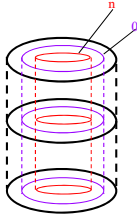
在硬盘中信息分布呈以下层次：记录面、圆柱面、磁道和扇区。一台硬盘驱动器中有多个盘片，每个盘片有两个记录面，每个记录面对应一个磁头，记录面号就是磁头号。在记录面上，一条条磁道形成一组同心圆，通常将一条磁道划分为若干个段，每个段称为一个扇区，每个扇区存放一个定长信息块（如512个字节）。在一个盘组中，各记录面上相同编号（位置）的诸磁道构成一个圆柱面。例如，某驱动器有4片8面，则8个0号磁道构成0号圆柱面，8个1号磁道构成1号圆柱面……硬盘的圆柱面数就等于一个记录面上的磁道数，圆柱面号即对应的磁道号。

2025年12月

8.2 外部设备



在存入文件时，应首先将一个文件尽可能地存放在同一圆柱面中。这是因为如果选择同一圆柱面上的不同磁道，则由于各记录面的磁头已同时定位，换道的时间只是磁头选择电路的译码时间，相对于定位操作可以忽略不计。而如果选择同一记录面上的不同磁道，则每次换道时都要进行磁头定位操作，速度较慢。



2025年12月

8.2 外部设备



(2) 磁盘地址

主机向磁盘控制器送出有关寻址信息，磁盘地址一般表示为：

驱动器号、圆柱面（磁道）号、记录面（磁头）号、扇区号（块号）。

(3) 硬盘的主要性能指标

硬盘标称的容量是指格式化容量，即用户实际可以使用的存储容量，而非格式化容量是指磁记录介质上全部的磁化单元数。格式化容量一般约为非格式化容量的60%~70%左右。

2025年12月

8.2 外部设备



硬盘的存取时间主要包括4部分：第一部分是磁头从原先位置移动到目的磁道所需要的时间，称为**定位时间或寻道时间**；第二部分是磁头在到达目的磁道以后，等待被访问的记录块旋转到磁头下方的等待时间，称为**旋转时间或等待时间**；第三部分是信息的**读写操作时间**，也称为传输时间；最后是**磁盘控制器的开销**。由于寻找不同磁道和等待不同记录块所花的时间不同，所以通常取它们的平均值。传输时间和控制器的开销相对平均寻道时间 $T_s$ 和平均等待时间 $T_w$ 来说要小得多，可以忽略不计，所以硬盘的平均存取时间 $T_a$ 约等于：

$$T_a = T_s + T_w = \frac{T_{smax} + T_{smin}}{2} + \frac{T_{wmax} + T_{wmin}}{2}$$

2025年12月

8.2 外部设备



2. 磁盘阵列RAID

RAID将一组磁盘驱动器用某种逻辑方式联系起来，作为逻辑上的一个磁盘驱动器来使用，一般情况下，组成的逻辑磁盘驱动器的容量要小于各个磁盘驱动器容量的总和。

RAID的优点如下：

① 成本低，功耗小，传输速率高。在RAID中，可以让很多磁盘驱动器同时传输数据，而这些磁盘驱动器在逻辑上又是一个磁盘驱动器，所以使用RAID可以达到单个磁盘驱动器几倍、几十倍甚至上百倍的速率。

2025年12月

8.2外部设备

② 提供容错功能。这是使用RAID的第二个原因，如果不考虑磁盘上的循环冗余校验（CRC）码，普通磁盘驱动器无法提供容错功能。RAID的容错是建立在每个磁盘驱动器的硬件容错功能之上的，所以它提供更高的安全性。

③ RAID比起传统的大直径磁盘驱动器来，在同样的容量下，价格要低许多。

8.2外部设备

8.2.2常见输入设备

1.键盘

键盘是计算机系统不可缺少的输入设备，人们通过键盘上的按键直接向计算机输入各种数据、命令及指令，从而使计算机完成不同的运算及控制任务。

2.鼠标

鼠标是控制显示器光标移动的输入设备，由于它能在屏幕上实现快速精确的光标定位，可用于屏幕编辑、选择菜单和屏幕作图，鼠标器已成为计算机系统中必不可少的输入设备。

8.2外部设备

8.2.3常见输出设备

1．显示器

显示器是将电信号转换成视觉信号的一种装置，在计算机系统中，显示器被用作输出设备和人机对话的重要工具。显示器输出的内容不能长期保存，当显示器关机或显示别的内容时，原有内容就消失了，所以显示器属于**软拷贝输出设备**。

为了不断提供刷新画面的信号，必须把字符或图形信息存储在一个显示缓冲区中，这个缓冲区又称为视频存储器（VRAM）。显示器一方面对屏幕进行光栅扫描，另一方面同步地从VRAM中读取显示内容，送往显示器件。因此，对VRAM的操作是显示器工作的软、硬件界面所在。

8.2外部设备

显示存储器VRAM的容量由分辨率和灰度级决定，分辨率越高，灰度级越高，VRAM的容量就越大。同时，VRAM的存取周期必须满足刷新率的要求。

分辨率由每帧画面的像素数决定，而像素具有明暗和色彩属性。黑白图像的明暗程度称为灰度，明暗变化的数量称为灰度级，所以在单色显示器中，仅有灰度级指标。彩色图像是由多种颜色构成的，在彩色显示器中能显示的颜色种类称为颜色数。

8.2外部设备

显示方式从功能上分为两大类：字符方式和图形方式。

在字符显示方式中，将一屏中可显示的最多字符数称为分辨率，例如80列×25行。字符方式的VRAM通常分成两部分：字符代码缓存和显示属性缓存。字符代码缓存中存放着显示字符的ASCII码，每个字符占1个字节；显示属性缓存中存放着字符的显示属性，一般也占1个字节。

在图形显示方式中，将一屏中可显示的像素点数称为分辨率。对VRAM的需求随显示分辨率的大小和颜色数的多少而不同，公式如下：

VRAM的容量=分辨率×颜色深度

其中，颜色深度与颜色数的对应关系为：

颜色深度=log<sub>2</sub>颜色数

8.2外部设备

2．打印机

打印机的功能是将计算机的处理结果以字符或图形的形式印刷到纸上，转换为书面信息，便于人们阅读和保存。由于打印输出结果能永久性保留，故称为**硬拷贝输出设备**。

按照打印的工作原理不同，打印机分为击打式和非击打式打印机两大类。

打印机按照输出工作方式可分为串式打印机、行式打印机和页式打印机3种。

8.2外部设备



打印机通常有两种工作模式，即文本模式（字符模式）和图形模式。在文本模式下，主机向打印机输出字符代码（ASCII码）或汉字代码（国标码），打印机则依据代码从位于打印机上的字符库或汉字库中取出点阵数据，在纸上“打”出相应字符或汉字。与图形模式相比，文本模式所需传送的数据量少，占用主机CPU的时间少，因而效率较高，但所能打印的字符或汉字的数量受到字库的限制。在图形模式下，主机向打印机直接输出点阵图形数据，从而可打印出字符、汉字、图形、图像等，但图形模式所需传送的数据量大。

2025年12月

第8章总线与输入输出系统



- 8.1总线设计
- 8.2外部设备
- 8.3主机与外设的连接
- 8.4输入输出信息传送控制方式
- 8.5中断系统
- 8.6 DMA的实现
- 8.7通道处理机

2025年12月

8.3主机与外设的连接



8.3.1接口的功能和基本组成

1. 接口的功能

- (1) 实现主机和外设的通信联络控制  
接口中的同步控制电路用来解决主机与外设的时间配合问题。
- (2) 进行地址译码和设备选择  
当CPU送来选择外设的地址码后，接口必须对地址进行译码以产生设备选择信息，使主机能和指定外设交换信息。

2025年12月

8.3主机与外设的连接



- (3) 实现数据缓冲  
在接口电路中，一般设置有一个或几个数据缓冲寄存器，用于数据的暂存，以避免因速度不一致而丢失数据。在传送过程中，先将数据送入数据缓冲寄存器中，然后再送到输出设备或主机中去。
- (4) 数据格式的变换  
在输入或输出操作过程中，为了满足主机或外设的各自要求，接口电路中必须具有完成各类数据相互转换的功能。

2025年12月

8.3主机与外设的连接



- (5) 传递控制命令和状态信息  
当CPU要启动某一外设时，通过接口中的控制命令寄存器向外设发出启动命令；当外设准备就绪时，则有状态信息送回接口中的状态寄存器，为CPU 提供反馈信息，告诉CPU，I/O设备已经具备和CPU交换数据的条件。当外设向CPU提出中断请求和DMA请求时，CPU也有相应的响应信号反馈给外设。

2025年12月

8.3主机与外设的连接



- 2. 接口的基本组成  
接口中要分别传送数据信息、控制信息和状态信息，数据信息、控制信息和状态信息都通过数据总线来传送。大多数计算机都把I/O设备的状态信息视为输入数据，而把控制信息看成输出数据，并在接口中分设各自相应的寄存器，赋以不同的端口地址，各种信息分时地使用数据总线传送到各自的寄存器中。

2025年12月

8.3主机与外设的连接

```
graph LR
    Host[主机] -- 地址信息 --> Interface[接口]
    subgraph Interface
        ID[外设识别]
        DCS[数据/命令/状态]
        CC[控制电路]
    end
    Interface -- 数据(并) --> Host
    Host -- 控制联络信息 --> Interface
    Interface -- 数据(并/串) --> Device[外设]
    Device -- 控制联络 --> Interface
```

2025年12月

8.3主机与外设的连接

接口与端口是两个不同的概念。**端口是指接口电路中可以进行读/写的寄存器**，若干个端口加上相应的控制逻辑电路才组成接口。

通常，一个接口中包含有数据端口、控制端口和状态端口。存放数据信息的寄存器称为数据端口，存放控制命令的端口称为命令端口，存放状态信息的寄存器称为状态端口。CPU通过输入指令可以从有关端口中读出信息，通过输出指令可以把信息写入有关端口。**对状态端口只进行输入操作**，将设备状态标志送到CPU中去；**对命令端口只进行输出操作**，CPU将向外设发送各种控制命令。因此，在有的接口电路中状态信息和控制信息共用一个寄存器，称之为设备的控制状态寄存器。

2025年12月

8.3主机与外设的连接

8.3.2外设的识别与端口寻址

1.端口地址编址方式

(1) 独立编址

在这种编址方式中，主存地址空间和I/O端口地址空间是相对独立的，分别单独编址。例如，在8086中，其主存地址范围是从00000H到FFFFFFH，其I/O端口地址范围是从0000H到FFFFH，它们互相独立，互不影响。CPU访问主存时，由主存读写控制线控制；访问外设时，由I/O读写控制线控制，所以在指令系统中必须设置专门的I/O指令。当CPU使用I/O指令时，其指令的地址字段直接或间接的指示出端口地址。这些端口地址被接口电路中的地址译码器接收并且进行译码，符合者就是CPU所指定的外设寄存器，该外设寄存器将被CPU访问。

2025年12月

8.3主机与外设的连接

(2) 统一编址

在这种编址方式中，I/O端口地址和主存单元地址是统一编址的，把I/O接口中的端口作为主存单元一样进行访问，不设置专门的I/O指令。当CPU访问外设时，把分配给该外设的地址码（具体到该外设接口中的某一寄存器号）送到地址总线上，然后各外设接口中的地址译码器对地址码进行译码，如果符合即是CPU指定的外设寄存器。

2025年12月

8.3主机与外设的连接

2.独立编址方式的端口访问

Intel 80x86最多可直接寻址256个**字节端口**，可间接寻址65536个**字节端口**。

任意两个连续的8位端口可作为16位端口处理；四个连续的8位端口可作为32位端口处理。因此，I/O地址空间最多能提供64K个8位端口、32K个16位端口、16K个32位端口或总容量不超过64KB的不同端口的组合。

2025年12月

第8章总线与输入输出系统

8.1总线设计

8.2外部设备

8.3主机与外设的连接

8.4输入输出信息传送控制方式

8.5中断系统

8.6 DMA的实现

8.7通道处理机

2025年12月

8.4输入输出信息传送控制方式

8.4.1程序查询方式

1. 程序查询方式的基本思想

程序查询方式的输入和输出完全是通过CPU执行程序来完成的。一旦某一外设被选中并启动之后，CPU将查询这个外设的某些状态位，看其是否准备就绪？所谓准备就绪是指CPU在执行输入操作时，外设一定是“准备好”的；在执行输出操作时，外设一定是“缓冲器空”的。若外设未准备就绪，CPU就循环等待，只有当外设已做好准备，CPU才能执行I/O指令进行一次数据传送，这就是程序查询方式。

这种方式控制简单，但外设和主机不能同时工作，各外设之间也不能同时工作，系统效率很低，因此，仅适用于外设的数目不多、对I/O处理的实时要求不那么高、CPU的操作任务比较单一且并不很忙的情况。

2025年12月

8.4输入输出信息传送控制方式

2. 程序查询方式的工作流程

(1) 预置传送参数

在传送数据之前，由CPU执行一段程序，预置传送参数。传送参数包括存取数据的主存缓冲区首地址和传送数据的个数。

(2) 向I/O接口发命令字

当CPU选中某台外设时，执行输出指令向I/O接口发出命令字，启动外设，为接收数据或发送数据的操作做准备。

2025年12月

8.4输入输出信息传送控制方式

(3) 从I/O接口取回状态字

CPU执行输入指令，从I/O接口中取回状态字并进行测试，判断数据传送是否可以继续进行。

(4) 查询外设标志

CPU不断查询状态标志，如果外设没有准备就绪，CPU就踏步进行等待，一直到这个外设准备就绪，并发出“准备就绪”信号为止。

2025年12月

8.4输入输出信息传送控制方式

(5) 传送数据

只有外设准备好，才能实现主机与外设间的一次数据传送。输入时，CPU执行输入指令，从I/O接口的数据缓冲寄存器中接收数据；输出时，CPU执行输出指令，将数据写入I/O接口的数据缓冲寄存器。

(6) 修改传送参数

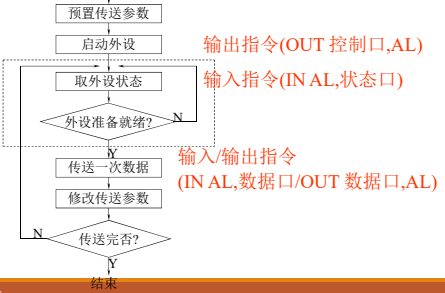
每进行一次数据传送，需要修改传送参数，其中包括主存缓冲区地址加1，传送个数减1。

(7) 判断传送是否结束

如果传送个数不为0，则转第3步，继续传送，直到传送结束为止。

2025年12月

8.4输入输出信息传送控制方式



```
graph TD
    A[预置传送参数] --> B[启动外设]
    B --> C[取外设状态]
    C --> D{外设准备就绪?}
    D -- N --> C
    D -- Y --> E[传送一次数据]
    E --> F[修改传送参数]
    F --> G{传送完否?}
    G -- N --> C
    G -- Y --> H[结束]
```

输出指令(OUT 控制口,AL)  
输入指令(IN AL,状态口)  
输入/输出指令  
(IN AL,数据口/OUT 数据口,AL)

2025年12月

8.4输入输出信息传送控制方式

8.4.2程序中断方式

1. 中断的提出

程序查询方式虽然简单，但却存在着下列明显的缺点：

① 在查询过程中，CPU长期处于踏步等待状态，使系统效率大大降低；

② CPU在一段时间内只能和一台外设交换信息，其他设备不能同时工作；

③ 不能发现和处理预先无法估计的错误和异常情况。

2025年12月



8.4输入输出信息传送控制方式

为了提高输入输出能力和CPU的效率，程序中断方式被引进计算机系统。程序中断方式的思想是：CPU在程序中安排好在一时刻启动某一台外设，然后CPU继续执行原来程序，不需要像查询方式那样一直等待外设的准备就绪状态。一旦外设完成数据传送的准备工作（输入设备的数据准备好或输出设备的数据缓冲器空）时，便主动向CPU发出一个中断请求，请求CPU为自己服务。在可以响应中断的条件下，CPU暂时中止正在执行的程序，转去执行中断服务程序为中断请求者服务，在中断服务程序中完成一次主机与外设之间的数据传送，传送完成后，CPU仍返回原来的程序，从断点处继续执行。

2025年12月

8.4输入输出信息传送控制方式

The diagram illustrates the interrupt control process. It shows an '外部设备' (External Device) and a 'CPU' block. The CPU contains '中断服务程序' (Interrupt Service Routine) and '现行程序' (Current Program). The process flow is: 1. '外部设备' starts '工作' (Work). 2. When '工作' is '完成' (Completed), it sends an '中断请求' (Interrupt Request). 3. The CPU's '现行程序' sends a '启动' (Start) signal to the '中断服务程序'. 4. The '中断服务程序' sends a '响应' (Response) signal back to the '现行程序'. 5. The '中断服务程序' then performs its own '工作' (Work). 6. When its '工作' is '完成' (Completed), it sends an '中断返回' (Interrupt Return) signal to the '现行程序'. 7. The '现行程序' then resumes its '工作' (Work). This cycle repeats for multiple interrupts.

2025年12月

8.4输入输出信息传送控制方式

中断的处理过程实际上是程序的切换过程，即从现行程序切换到中断服务程序，再从中断服务程序返回到现行程序。CPU每次执行中断服务程序前总要保护断点、保护现场，执行完中断服务程序返回现行程序之前又要恢复现场、恢复断点。这些中断的辅助操作都将会限制数据传送的速度。

2025年12月

8.4输入输出信息传送控制方式

2. 程序中断与调用子程序的区别

表面上看起来，计算机的中断处理过程有点类似于调用子程序的过程，这里现行程序相当于主程序，中断服务程序相当于子程序。但是，它们之间却是有着本质上的区别的。

- (1)子程序的执行是由程序员事先安排好的（由一条调用子程序指令转入），而中断服务程序的执行则是由随机的中断事件引起的；
- (2)子程序的执行受到主程序或上层子程序的控制，而中断服务程序一般与被中断的现行程序毫无关系；
- (3)不存在同时调用多个子程序的情况，而有可能发生多个外设同时请求CPU为自己服务的情况。

2025年12月

8.4输入输出信息传送控制方式

3. 中断的基本类型

(1) 自愿中断和强迫中断

自愿中断又称程序自中断，它不是随机产生的中断，而是在程序中安排的有关指令，这些指令可以使机器进入中断处理的过程，如80x86指令系统中的软中断指令INT n。

强迫中断是随机产生的中断，不是程序中事先安排好的。当这种中断产生后，由中断系统强迫计算机中止现行程序并转入中断服务程序。

2025年12月

8.4输入输出信息传送控制方式

(2) 程序中断和简单中断

**程序中断**就是我们前面提到的中断，主机在响应中断请求后，通过执行一段中断服务程序来处理更紧迫的任务。

简单中断就是外设与主存间直接进行信息交换的方法，即DMA方式。这种“中断”不去执行中断服务程序，故不破坏现行程序的状态。主机发现有简单中断请求（也就是DMA请求）时，将让出一个或几个存取周期供外设与主存交换信息，然后继续执行程序。

2025年12月

8.4输入输出信息传送控制方式

(3)内中断和外中断

内中断是指由于CPU内部硬件或软件原因引起的中断。  
**外中断**是指CPU以外的部件引起的中断。

(4) 向量中断和非向量中断

**向量中断**是指那些中断服务程序的入口地址是由中断事件自己提供的中断。中断事件在提出中断请求的同时，通过硬件向主机提供中断服务程序入口地址，即向量地址。  
**非向量中断**的中断事件不能直接提供中断服务程序的入口地址，而由CPU 查询之后得到。

2025年12月

8.4输入输出信息传送控制方式

(5) 单重中断和多重中断

**单重中断**在CPU执行中断服务程序的过程中不能被再打断。  
**多重中断**在执行某个中断服务程序的过程中，CPU 可去响应级别更高的中断请求，又称为中断嵌套。

2025年12月

8.4输入输出信息传送控制方式

8.4.3直接存储器存取（DMA）方式

直接存储器存取（DMA）方式是在外设和主存之间开辟一条“直接数据通道”，在不需要CPU干预也不需要软件介入的情况下在两者之间进行的高速数据传送方式。这样不仅能保证CPU的高效率，而且能满足高速外设的需要。

2025年12月

8.4输入输出信息传送控制方式

1. DMA方式的特点

DMA方式具有下列特点：  
① 它使主存与CPU的固定联系脱钩，主存既可被CPU访问，又可被外设访问。  
② 在数据块传送时，主存地址的确定、传送数据的计数等都由硬件电路直接实现。  
③ 主存中要开辟专用缓冲区，及时供给和接收外设的数据。  
④ DMA传送速度快，CPU和外设并行工作，提高了系统的效率。  
⑤ DMA在传送开始前要通过程序进行预处理，结束后要通过中断方式进行后处理。

2025年12月

8.4输入输出信息传送控制方式

2. DMA和中断的区别

① 中断方式是程序切换，需要保护和恢复现场；而DMA方式除了开始和结尾时，不占用CPU的任何资源。  
② 对中断请求的响应只能发生在每条指令执行完毕时；而对DMA请求的响应可以发生在每个机器周期结束时。

取指令 取源操作数 取目的操作数 执行

DMA断点 中断断点

2025年12月

8.4输入输出信息传送控制方式

③ 中断传送过程需要CPU的干预；而DMA传送过程不需要CPU的干预，故数据传送速率非常高，适合于高速外设的成组数据传送。  
④ DMA请求的优先级高于中断请求。  
⑤ 中断方式具有对异常事件的处理能力；而DMA方式仅限于完成传送信息块的I/O操作。

2025年12月

8.4输入输出信息传送控制方式

3. DMA方式的应用

DMA方式一般应用于主存与高速外设间的简单数据传送。如磁盘、磁带、光盘等辅助存储器以及其他带有局部存储器的外设、通信设备等都是高速外设。

对磁盘的读写是以数据块为单位进行的，一旦找到数据块起始位置，就将连续地读写。从磁盘中读出数据或往磁盘中写入数据时，一般采用DMA方式传送，即直接将数据由主存经数据总线输出到磁盘接口，然后写入盘片；或将数据由盘片读出到磁盘接口，然后经数据总线写入主存。

2025年12月

8.4输入输出信息传送控制方式

4. DMA控制器

在DMA传送方式中，对数据传送过程进行控制的硬件称为DMA控制器。当外设需要进行数据传送时，通过DMA控制器向CPU提出DMA传送请求，CPU响应之后将让出系统总线，由DMA控制器接管总线进行数据传送。

在DMA传送过程中，DMA控制器将接管CPU的地址总线、数据总线和控制总线，CPU的主存控制信号被禁止使用。而当DMA传送结束后，将恢复CPU的一切权力并开始执行其操作。由此可见，DMA控制器必须具有控制系统总线的能力，也就是说能够像CPU一样输出地址信号，接收或发出控制信号，输入或输出数据信号。

2025年12月

8.4输入输出信息传送控制方式

DMA控制器在外设与主存之间直接传送数据期间，完全代替CPU进行工作，它的主要功能有：

① 接受外设发出的DMA请求，并向CPU发出总线请求。

② 当CPU响应此总线请求，发出总线响应信号后，接管对总线的控制，进入DMA操作周期。

③ 确定传送数据的主存单元地址及传送长度，并能自动修改主存地址计数值和传送长度计数值。

④ 规定数据在主存与外设之间的传送方向，发出读写或其他控制信号，并执行数据传送的操作。

⑤ 向CPU报告DMA操作的结束。

2025年12月

8.4输入输出信息传送控制方式

8.4.4通道控制方式

1.通道控制方式与DMA方式的区别

通道控制方式是DMA方式的进一步发展，实质上，通道也是实现外设和主存之间直接交换数据的控制器。与DMA控制器相比，两者的主要区别在于：

① DMA控制器是通过专门设计的硬件控制逻辑来实现对数据传送的控制；而通道则是一个具有特殊功能的处理器，它具有自己的指令和程序，通过执行通道程序来实现对数据传送的控制，故通道具有更强的独立处理数据输入输出的功能。

② DMA控制器通常只能控制一台或少数几台同类设备；而一个通道则可以同时控制许多台同类或不同类的设备。

2025年12月

8.4输入输出信息传送控制方式

2. 通道的功能

通道能独立地执行通道程序，产生相应的控制信号，实现对外设的统一管理和外设与主存之间的数据传送。但它不是一个完全独立的处理器，不能完全脱离CPU工作。通道还要受到CPU的管理，如启动、停止等，而且通道还应该向CPU报告自己的状态，以便CPU决定下一步的处理。所以它是从属于CPU的一个专用处理器。

2025年12月

第8章总线与输入输出系统

8.1总线设计

8.2外部设备

8.3主机与外设的连接

8.4输入输出信息传送控制方式

8.5中断系统

8.6 DMA的实现

8.7通道处理机

2025年12月

8.5中断系统

中断系统是计算机实现中断功能的软、硬件总称。一般在CPU中配置中断机构，在外设接口中配置中断控制器，在软件上设计相应的中断服务程序。

**8.5.1中断请求和中断判优**

**1.中断源和中断请求信号**

中断源是指中断请求的来源，即引起计算机中断的事件。通常，一台计算机允许有多个中断源。由于每个中断源向CPU发出中断请求的时间是随机的，为了记录中断事件并区分不同的中断源，可采用具有存储功能的触发器来记录中断源，这个触发器称为中断请求触发器（INTR）。当某一个中断源有中断请求时，其相应的中断请求触发器置成“1”状态，表示该中断源向CPU提出中断请求。

8.5中断系统

**2. 中断优先级与判优方法**

当多个中断源同时发出中断请求时，CPU在任何瞬间只能接受一个中断源请求。究竟首先响应哪一个中断请求呢？通常，把全部中断源按中断的性质和处理的轻重缓急安排优先级，并进行排队。

确定中断优先级的原则是：对那些提出中断请求后需要立刻处理，否则就会造成严重后果的中断源规定最高的优先级；而对那些可以延迟响应和处理的的中断源规定较低的优先级。例如，故障中断一般优先级较高，其次是简单中断，接着才是I/O设备中断。

8.5中断系统

中断判优的方法可分为软件判优和硬件判优两种。软件判优法，就是用程序来判优先级。软件判优方法简单，可以灵活地修改中断源的优先级；但查询、判优完全是靠程序实现的，不但占用CPU时间，而且判优速度慢。

```
graph TD; INT1{INT1?} -- Y --> S1[对1号中断源服务]; INT1 -- N --> INT2{INT2?}; INT2 -- Y --> S2[对2号中断源服务]; INT2 -- N --> INT3{INT3?}; INT3 -- Y --> S3[对3号中断源服务]; INT3 -- N --> INT4{INT4?}; INT4 -- Y --> S4[对4号中断源服务]; INT4 -- N --> INTn{INTn?}; INTn -- Y --> Sn[对n号中断源服务]; INTn -- N --> INT1;
```

8.5中断系统

采用硬件判优电路实现中断优先级的判定可节省CPU时间，判优速度快，但是成本较高。根据中断请求信号的传送方式不同，有不同的优先排队电路。这些排队电路的共同特点是：优先级别高的中断请求将自动封锁优先级别低的中断请求。硬件排队电路一旦设计连接好之后，将无法改变其优先级别。

8.5中断系统

**8.5.2中断响应和中断处理**

**1.CPU响应中断的条件**

(1) CPU接收到中断请求信号

首先中断源要发出中断请求，同时CPU还要接收到这个中断请求信号。

(2) CPU允许中断

CPU允许中断即开中断。CPU内部有一个中断允许触发器，只有当其被置位时，CPU才可能响应中断源的中断请求（中断开放）。如其被复位，CPU处于不可中断状态，即使中断源有中断请求，CPU也不响应（中断关闭）。

8.5中断系统

(3) 一条指令执行完毕

一般情况下，CPU在一条指令执行完毕，且没有更紧迫的任务时才能响应中断请求。

8.5中断系统

2. 中断隐指令

CPU响应中断之后，经过某些操作，转去执行中断服务程序。这些操作是由硬件直接实现的，我们把它称为中断隐指令。  
**中断隐指令并不是指令系统中的一条真正的指令，它没有操作码，所以中断隐指令是一种不允许、也不可能为用户使用的特殊指令。**  
其所完成的操作主要有：

8.5中断系统

(1) 保存断点

将原来程序的断点（即程序计数器PC的内容）保存起来。

(2) 暂不允许中断

为了在用软件保护中断现场（即CPU 的主要寄存器状态）时，不被新的中断所打断，从而保证被中断的程序在中断服务程序执行完毕之后能接着正确地执行下去。

(3) 引出中断服务程序

引出中断服务程序的实质就是取出中断服务程序的入口地址送程序计数器。

8.5中断系统

3. 中断周期

中断周期需完成如下操作：  
(1)将特定地址“0”送至存储器地址寄存器，记作 $0 \rightarrow MAR$ ；  
(2)将PC的内容（断点）送至MDR，记作 $(PC) \rightarrow MDR$ ；  
(3)向主存发写命令，启动存储器做写操作，记作Write；  
(4)将MDR的内容通过数据总线写入到MAR所指示的主存单元（0号）中，记作 $MDR \rightarrow M(MAR)$ ；

8.5中断系统

(5)向量地址形成部件的输出送至PC，为进入中断服务程序作准备，记作向量地址 $\rightarrow PC$ ；  
(6)关中断，将中断允许触发器清0，记作 $0 \rightarrow EINT$ 。  
如果断点存入堆栈，只需将上述(1)改为堆栈指针 $SP \rightarrow MAR$ 。

8.5中断系统

```
graph LR; A((中断源  
工作完成  
允许中断  
发中断请求)) -- INTR --> B((CPU  
中断优先级  
判定选择  
中断响应)); B -- INTR --> C((识别中断源  
向量地址  
形成)); C -- 向量地址 --> D((现场处理  
启动中断  
服务程序));
```

8.5中断系统

向量地址通常有以下两种情况。

(1) 向量地址是中断服务程序的入口地址  
如果向量地址就是中断服务程序的入口地址，则CPU不需要再经过处理就可以进入相应的中断服务程序，各中断源在接口中由硬件电路形成一条含有中断服务程序入口地址的特殊指令（重新启动指令），从而转入相应的中断服务程序。

(2) 向量地址是中断向量表的指针  
如果向量地址是中断向量表的指针，则向量地址指向一个中断向量表，从中断向量表的相应单元中再取出中断服务程序的入口地址，此时中断源给出的向量地址是中断服务程序入口地址的地址。

8.5中断系统



5. 中断现场的保护和恢复

中断现场指的是发生中断时CPU的主要状态，其中最重要的是断点，另外还有一些通用寄存器的状态。之所以需要保护和恢复现场的原因是因为CPU要先后执行两个完全不同的程序（现行程序和中断服务程序），必须进行两种程序运行状态的转换。一般来说，在中断隐指令中，CPU硬件将自动保存断点，有些计算机还自动保存程序状态寄存器的内容。但是，在许多应用中，要保证中断返回后原来的程序能正确地继续运行，仅保存这一、二个寄存器的内容是不够的。

2025年12月

8.5中断系统



为此，在中断服务程序开始时，应由软件去保存那些硬件没有保存，而在中断服务程序中又可能用到的寄存器（如某些通用寄存器）的内容，在中断返回之前，这些内容还应该被恢复。

现代计算机一般都先采用硬件方法来自动快速的保护和恢复部分重要的现场，其余寄存器的内容再由软件完成保护和恢复，这种方法的硬件支持是堆栈。

2025年12月

8.5中断系统



8.5.3多重中断与中断屏蔽

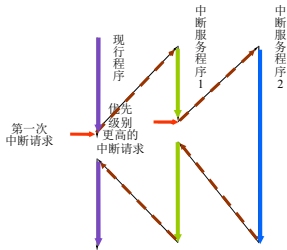
1. 中断嵌套

中断嵌套的层次可以有多层，越在里层的中断越急迫，优先级越高，因此优先得到CPU的服务。

要使计算机具有多重中断的能力，首先要能保护多个断点，先发生的中断请求的断点，先保护后恢复；后发生的中断请求的断点，后保护先恢复，堆栈的先进后出特点正好满足多重中断这一先后次序的需要。在CPU进入某一中断服务程序之后，系统必须处于开中断状态，否则中断嵌套是不可能实现的。

2025年12月

8.5中断系统



2025年12月

8.5中断系统



2. 允许和禁止中断

允许中断还是禁止中断是用CPU中的中断允许触发器控制的，当中断允许触发器（EINT）被置“1”，则允许中断；当中断允许触发器（EINT）被置“0”，则禁止中断。

允许中断即开中断，下列情况应**开中断**：

① 无论是单重中断还是多重中断，在中断服务程序执行完毕，恢复中断现场之后。

② 在多重中断的情况下，保护中断现场之后。

禁止中断即关中断，下列情况应**关中断**：

① 当响应某一级中断请求，不再允许被其他中断请求打断时。

② 在中断服务程序的保护和恢复现场之前。

2025年12月

8.5中断系统



3. 中断屏蔽

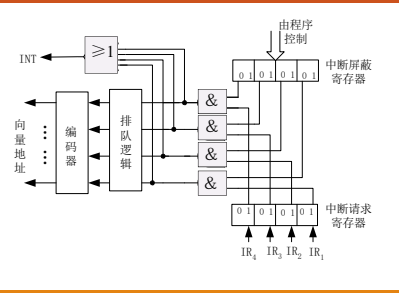
中断源发出中断请求之后，这个中断请求并不一定能真正送到CPU去，在有些情况下，可以用程序方式有选择地封锁部分中断，这就是中断屏蔽。

在中断接口电路中，多个屏蔽触发器组成一个屏蔽寄存器，其内容称为屏蔽字或屏蔽码，由程序来设置。屏蔽字某一位的状态将成为本中断源能否真正发出中断请求信号的必要条件之一。

这样，就可实现CPU对中断处理的控制，使中断能在系统中合理协调地进行。中断屏蔽寄存器的作用：用程序设置的方法将屏蔽寄存器中的某一位置“1”，则对应的中断请求被封锁，无法去参加排队判优；若屏蔽寄存器中的某一位置“0”，才允许对应的中断请求送往CPU。

2025年12月

8.5中断系统



2025年12月

8.5中断系统

如一个中断系统有16个中断源，每一个中断源按其优先级赋予一个屏蔽字。“0”表示开放，“1”表示屏蔽。

| 中断源的优先级 | 屏蔽字（16位）  |
|---------|-----------|
| 1       | 111...111 |
| 2       | 011...111 |
| 3       | 001...111 |
| ⋮       | ⋮         |
| 15      | 000...011 |
| 16      | 000...001 |

第1级中断源的优先级最高，它禁止本级和更低级的中断请求；第16级中断源的优先级最低，它仅禁止本级的中断请求，而对其他高级的中断请求全部开放。

2025年12月

8.5中断系统

#### 4.中断升级

中断屏蔽字的另一个作用是可以改变中断优先级，将原级别较低的中断源变成较高的级别，我们称之为中断升级。这实际上是一种动态改变优先级的方法。

这里所说的改变优先次序是指改变中断的处理次序。中断处理次序和中断响应次序是两个不同的概念，**中断响应次序是由硬件排队电路决定的，无法改变**。但是，**中断处理次序可以由屏蔽码来改变的**，故把屏蔽码看成软排队器。中断处理次序可以不同于中断响应次序。

2025年12月

8.5中断系统

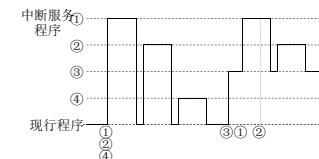
例如，某计算机的中断系统有4个中断源，每个中断源对应一个屏蔽码。中断响应的优先次序为1→2→3→4。中断的处理次序和中断的响应次序是一致的。

| 程序级别 | 屏蔽码 |    |    |    |
|------|-----|----|----|----|
|      | 1级  | 2级 | 3级 | 4级 |
| 第1级  | 1   | 1  | 1  | 1  |
| 第2级  | 0   | 1  | 1  | 1  |
| 第3级  | 0   | 0  | 1  | 1  |
| 第4级  | 0   | 0  | 0  | 1  |

2025年12月

8.5中断系统

根据这一次序，可以看到CPU运动的轨迹，当多个中断请求同时出现时，处理次序与响应次序一致；当中断请求先后出现时，允许优先级高的中断请求打断优先级低的中断服务程序，实现中断嵌套。



2025年12月

8.5中断系统

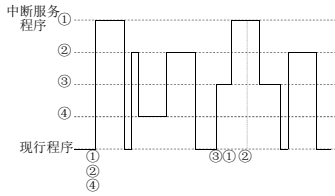
在不改变中断响应次序的条件下，通过改写屏蔽码可以改变中断处理次序，例如，要使中断处理次序改为1→4→3→2。

| 程序级别 | 屏蔽码 |    |    |    |
|------|-----|----|----|----|
|      | 1级  | 2级 | 3级 | 4级 |
| 第1级  | 1   | 1  | 1  | 1  |
| 第2级  | 0   | 1  | 0  | 0  |
| 第3级  | 0   | 1  | 1  | 0  |
| 第4级  | 0   | 1  | 1  | 1  |

2025年12月

8.5中断系统

在同样中断请求的情况下，CPU的运动轨迹发生了变化。



8.5中断系统

8.5.4中断全过程

中断全过程是指从中断源发出中断请求开始，CPU响应这个请求，现行程序被中断，转至中断服务程序，直至中断服务程序执行完毕，CPU再返回原来的程序继续执行的整个过程。

中断全过程分为五个阶段：

中断请求、中断判优、中断响应、中断处理、中断返回。

8.5中断系统

其中中断处理就是执行中断服务程序，中断服务程序基本上由三部分组成，第一部分为准备部分，其基本功能是保护现场，对于非向量中断方式则需要确定中断源，最后开放中断，允许更高级的中断请求打断低级的中断服务程序。第二部分为处理部分，即真正执行为某个中断源服务的中断服务程序。第三部分为结尾部分，首先要关中断，以防止在恢复现场过程中被新的中断打断，接着恢复现场，然后开放中断，以便返回原来的程序后可响应其它的中断请求。

8.5中断系统



8.5中断系统

8.5.5中断和异常

内部异常可分为故障、自陷和终止3类。  
故障是在引起故障的指令启动后，执行结束前被检测到的一类异常事件。  
自陷与故障等意外发生的异常事件不同，它是预先安排的一种CPU异常事件，就像预先设定的“陷阱”一样。  
终止是指在指令执行过程中发生了使机器无法继续执行的硬件故障，如电源掉电等，此时需要调出中断服务程序来重启系统。这种异常与故障和自陷不同，不是由特定指令产生的，而是随机发生的。

8.5中断系统

中断请求在当前指令执行完后进行检测，而异常事件则当前指令执行过程中进行检测，CPU对两者的处理过程也有所不同。中断的断点是当前指令后面一条指令的地址，而对于不同的异常事件，其断点是不一样的。例如，故障的断点是发生故障的当前指令的地址，自陷的断点则是自陷指令后面一条指令的地址，也就是说，断点的值由异常类型和发生异常时的PC的值决定。为了能在异常处理后正确返回到原被中断程序继续执行，数据通路必须能正确计算断点值，不过断点的保存方法和中断方式相同。



第8章总线与输入输出系统

8.1总线设计

8.2外部设备

8.3主机与外设的连接

8.4输入输出信息传送控制方式

8.5中断系统

8.6 DMA的实现

8.7通道处理机

2025年12月

8.6DMA的实现

8.6.1DMA接口

通常将DMA方式的接口电路称为DMA控制器。

1. DMA控制器的基本组成

(1) 主存地址计数器

用来存放主存中要交换数据的地址，该计数器的初始值为主存缓冲区的首地址，当DMA传送时，每传送一个数据，将地址计数器加“1”，从而以增量方式给出主存中要交换的一批数据的地址，直至这批数据传送完毕为止。

2025年12月

8.6DMA的实现

(2) 传送长度计数器

传送长度计数器用来记录传送数据块的长度。其初始值为传送数据的总字数或总字节数，每传送一个字或一个字节，计数器自动减1，当其内容为0时表示数据已全部传送完毕。也有些DMA控制器中，初始时将字数或字节数求补之后送计数器，每传送一个字或一个字节，计数器加1，当计数器溢出时，表示数据传送完毕。

(3) 数据缓冲寄存器

数据缓冲寄存器用来暂存每次传送的数据。输入时，数据由外设（如磁盘）先送往数据缓冲寄存器，再通过数据总线送到主存。反之，输出时，数据由主存通过数据总线送到数据缓冲寄存器，然后再送到外设。

2025年12月

8.6DMA的实现

(4) DMA请求触发器

DMA请求触发器的作用是每当外设准备好数据后给出一个控制信号，使DMA请求触发器置位。

(5) 控制/状态逻辑

它由控制和时序电路以及状态标志等组成，用于指定传送方向，修改传送参数，并对DMA请求信号和CPU响应信号进行协调和同步。

(6) 中断机构

当一个数据块传送完毕后触发中断机构，向CPU提出中断请求，CPU将进行DMA传送的结尾处理。

2025年12月

8.6DMA的实现

2.DMA控制器的引出线

(1)地址线

在DMA方式下，呈输出状态，可对主存进行地址选择；在CPU方式下，呈输入状态，可对DMA控制器中的有关寄存器进行寻址。

(2)数据线

在DMA方式下，用它进行数据传送；在CPU方式下，可对DMA控制器的有关寄存器进行编程。

2025年12月

8.6DMA的实现

(3)控制数据传送方式的信号线

存储器读信号、存储器写信号、外设读信号、外设写信号。

(4)DMA控制器与外设之间的联络信号线

DMA请求信号

DMA响应信号

(5)DMA控制器与CPU之间的联络信号线

总线请求

总线响应信号

2025年12月

8.6DMA的实现

3.DMA控制器的连接和传送

2025年12月

8.6DMA的实现

① 由外设向DMA控制器发出DMA请求信号DREQ。  
② DMA控制器向CPU发出总线请求信号HRQ。  
③ CPU向DMA控制器发出总线响应信号HLDA，此时DMA控制器获取了总线的控制权。  
④ DMA控制器向外设发出DMA响应信号DACK，表示DMA控制器已控制了总线，允许外设与主存交换数据。  
⑤ DMA控制器按主存地址计数器的内容发出地址信号作为主存地址的选择，同时主存地址计数器的内容加1。

2025年12月

8.6DMA的实现

⑥ DMA控制器发出 IOR 信号到外设，将外设数据读入数据缓冲寄存器，同时发出信号MEMW，将数据缓冲寄存器中的数据写入选中的主存单元。  
⑦ 传送长度计数器减1。  
重复⑤~⑦步骤，直到字节计数器减到0为止，数据块的DMA方式传送工作宣告完成。这时，DMA控制器的HRQ降为低电平，总线控制权交还CPU。

2025年12月

8.6DMA的实现

8.6.2DMA的传送方法

1.CPU停止访问主存法

这是最简单的DMA方法。这种方法是用DMA请求信号迫使CPU让出总线控制权。CPU在现行机器周期执行完成之后，使其数据、地址总线处于三态（浮空状态），并输出总线批准信号。每次DMA请求获得批准，DMA控制器获得总线控制权以后，连续占用若干个存取周期（总线周期）进行成组连续的数据传送，直至批量传送结束，DMA控制器才把总线控制权交回CPU。在DMA操作期间，CPU处于保持状态，停止访问主存，仅能进行一些与总线无关的内部操作。该方法只适用于高速外设的成组传送。

2025年12月

8.6DMA的实现

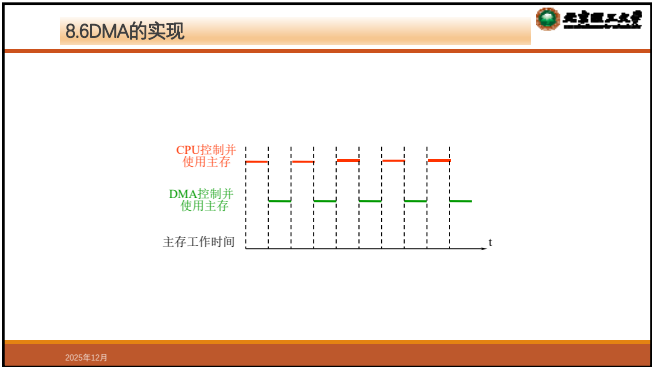
2025年12月

8.6DMA的实现

2.存储器分时法

把原来的一个存取周期分成两个时间片，一片分给CPU，一片分给DMA，使CPU和DMA交替地访问主存。这种方法无须申请和归还总线，使总线控制权的转移几乎不需要什么时间，所以对DMA传送来讲效率是很高的，而且CPU既不停止现行程序的运行，也不进入保持状态，在CPU不知不觉中便进行了DMA传送；但这种方法需要主存在原来的存取周期内为两个部件服务，如果要维持CPU的访存速度不变，就要求主存的工作速度提高一倍。另外，由于大多数外设的速度都不能与CPU相匹配，所以供DMA使用的时间片可能成为空操作，将会造成一些不必要的浪费。

2025年12月

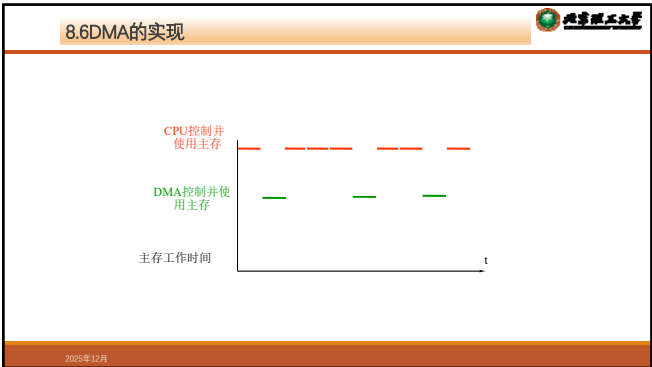


8.6DMA的实现

**3.周期挪用法**

周期挪用法是前两种方法的折中。当外设没有DMA请求时，CPU按程序要求访问主存；一旦外设DMA请求并获得CPU批准后，CPU让出一个周期的总线控制权，由DMA控制器控制系统总线，挪用一个存取周期进行一次数据传送，传送一个字节或一个字；然后，DMA控制器将总线控制权交回CPU，CPU继续进行自己的操作，等待下一个DMA请求的到来。重复上述过程，直至数据块传送完毕。如果在同一时刻，发生CPU与DMA的访问冲突，那么优先保证DMA工作，而CPU等待一个存取周期。若DMA传送期间CPU无须访存，则周期挪用对CPU执行程序无任何影响。

2025年12月



8.6DMA的实现

**8.6.3DMA的传送过程**

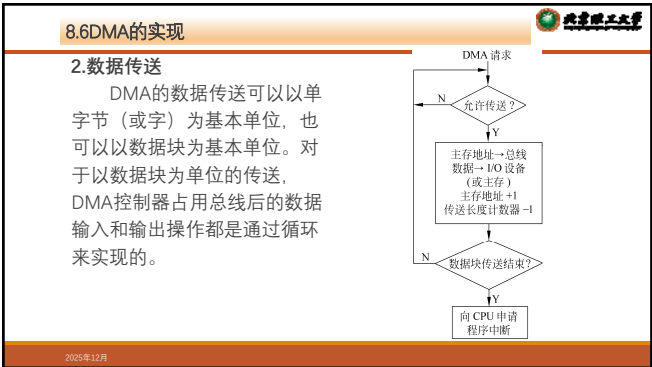
DMA的传送过程可分为3个阶段：

**1. DMA预处理**

在DMA传送之前必须要做准备工作，即初始化。这是由CPU来完成的。CPU首先执行几条I/O指令，用于测试外设的状态、向DMA控制器的有关寄存器置初值、设置传送方向、启动该外部设备等。

在这些工作完成之后，CPU继续执行原来的程序，在外设准备好发送的数据（输入）或接收的数据已处理完毕（输出）时，外设向DMA控制器发DMA请求，再由DMA控制器向CPU发总线请求。

2025年12月



8.6DMA的实现

**3. DMA后处理**

当传送长度计数器计到0时，DMA操作结束，DMA控制器向CPU发中断请求，CPU停止原来程序的执行，转去执行中断服务程序做DMA结束处理工作。

2025年12月

第8章总线与输入输出系统

8.1总线设计

8.2外部设备

8.3主机与外设的连接

8.4输入输出信息传送控制方式

8.5中断系统

8.6 DMA的实现

8.7通道处理机

2025年12月

8.7通道处理机

8.7.1通道工作过程

主要过程分为如下3步：  
① 在用户程序中使用访管指令进入管理程序，由CPU通过管理程序组织一个通道程序，并启动通道。  
② 通道执行CPU为它组织的通道程序，完成指定的数据输入/输出工作。  
③ 通道程序结束后向CPU发中断请求。CPU响应这个中断请求后，第二次调用管理程序对中断请求进行处理。

2025年12月

8.7通道处理机

8.7.2通道的类型

1.字节多路通道

字节多路通道是一种简单的共享通道，用于连接与管理多台低速设备，以字节交叉方式传送信息。

2025年12月

8.7通道处理机

一个字节多路通道，包括多个按字节方式传送信息的子通道。每个子通道服务于一个设备控制器，每个子通道都可以独立地执行通道程序。各个子通道可以并行工作，但是，所有子通道的控制部分是公共的，各个子通道可以分时地使用。  
通道不间断地、轮流地启动每个设备控制器，当通道为一个设备传送完一个字节后，就转去为另一个设备服务。当通道为某一设备传送时，其他设备可以并行地工作，准备需要传送的数据字节或处理收到的数据字节。这种轮流服务是建立在主机的速度比外设的速度高得多的基础之上的，它可以提高系统的工作效率。

2025年12月

8.7通道处理机

2.选择通道

对于高速设备，字节多路通道显然是不合适的。选择通道又称高速通道，在物理上它也可以连接多个设备，但这些设备不能同时工作，在一段时间内通道只能选择一台设备进行数据传送，此时该设备可以独占整个通道。

2025年12月

8.7通道处理机

每个选择通道只有一个以成组方式工作的子通道，逐个为多台高速外围设备服务。选择通道主要用于连接高速外设，如磁盘、磁带等，信息以成组方式高速传送。但是，在数据传送过程中还有一些辅助操作（如磁盘机的寻道等），此时会使通道处于等待状态，所以虽然选择通道具有很高的数据传输速率，但整个通道的利用率并不高。

2025年12月

## 8.7通道处理机



## 3.数组多路通道

数组多路通道是把字节多路通道和选择通道的特点结合起来的一种通道结构。它的基本思想是：当某设备进行数据传送时，通道只为该设备服务；当设备在执行辅助操作时，通道暂时断开与这个设备的连接，挂起该设备的通道程序，去为其他设备服务。数组多路通道有多个子通道，既可以执行多路通道程序，即像字节多路通道那样，所有子通道分时共享总通道，又可以用选择通道那样的方式成组地传送数据；既具有多路并行操作的能力，又具有很高的数据传输速率，使通道的效率充分得到发挥。

2025年12月

## 8.7通道处理机



每个选择通道只有一个以成组方式工作的子通道，逐个为多台高速外围设备服务。选择通道主要用于连接高速外设，如磁盘、磁带等，信息以成组方式高速传送。但是，在数据传送过程中还有一些辅助操作（如磁盘机的寻道等），此时会使通道处于等待状态，所以虽然选择通道具有很高的数据传输速率，但整个通道的利用率并不高。

2025年12月

## 8.7通道处理机



## 8.7.3通道的流量分析

通道流量是指通道在数据传送期内，单位时间里传送的字节数。它能达到的最大流量称为通道极限流量。

假设通道选择一次设备的时间为 $T_S$ ，每传送一个字节的时间为 $T_D$ ，通道工作时的极限流量分别如下。

## (1) 字节多路通道

字节多路通道每选择一台设备只传送一个字节，故其通道极限流量

$$f_{\max \cdot \text{byte}} = 1/(T_S + T_D)$$

2025年12月

## 8.7通道处理机



## (2) 选择通道

选择通道每选择一台设备就把 $N$ 个字节全部传送完，其通道极限流量

$$f_{\max \cdot \text{select}} = N/(T_S + NT_D) = 1/(T_S/N + T_D)$$

## (3) 数组多路通道

数组多路通道每选择一台设备只传送 $K$ 个字节，如果要传送 $N$ 个字节，就得经 $[N/K]$ 次传送才行，每次都要花去一个选择设备的时间 $T_S$ ，所以，其通道极限流量

$$f_{\max \cdot \text{block}} = K/(T_S + KT_D) = 1/(T_S/K + T_D)$$

2025年12月

## 8.7通道处理机



对于采用字节交叉方式工作的字节多路通道应为该通道所接各设备的字节传送速率之和，即

$$f_{\text{byte} \cdot j} = \sum_{i=1}^{p_j} f_{i \cdot j}$$

而对于其他两种类型的通道应为所接各设备的字节传送速率中之最大的那个，即

$$f_{\text{block} \cdot j} = \max_{i=1}^{p_j} f_{i \cdot j}$$

$$f_{\text{select} \cdot j} = \max_{i=1}^{p_j} f_{i \cdot j}$$

2025年12月

## 8.7通道处理机



为使通道所接外部设备在满负荷工作时仍不丢失信息，应使通道的实际最大流量不能超过通道的极限流量。

如果在I/O系统中有多个通道，各个通道是并行工作的，则I/O系统的极限流量应当是各通道或各子通道工作时的极限流量之和。

2025年12月