

I/O系统

笔记源文件: [Markdown](#), [长图](#), [PDF](#), [HTML](#)

1. I/O系统概述

1.1. I/O系统组成

- 1 I/O硬件: I/O设备+接口电路
- 2 I/O软件: 用I/O指令编写, 管理I/O电路和I/O接口, aka I/O驱动

1.2. I/O编址: 给每台I/O一个唯一编号

编址	本质特征	有关名词	CPU对二者访问
I/O-MM编址	I/O+MM地址空间统一	总线空间= I/O+MM地址	不区分
I/O独立编址	二者地址空间分开	设备号=独立的I/O地址	区分

1.3. I/O指令

- 1 隐式I/O指令: 适用于I/O-MM编址机器, 所有访存指令都可以直接访问I/O
- 2 显式I/O指令: 指令系统中有专门的I/O指令, 格式如下

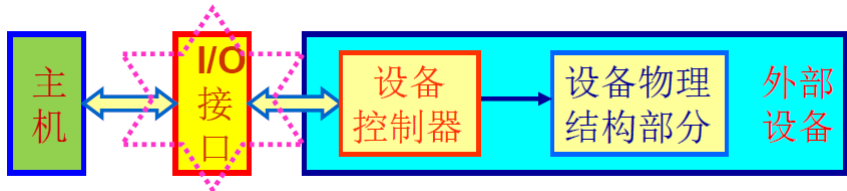
操作码	命令码	设备码
告诉CPU这是I/O指令	执行何种I/O操作	执行指令的是哪个I/O设备

+ I/O寻址过程: I/O指令 $\xrightarrow{\text{要访问的I/O地址}}$ I/O接口 $\xrightarrow[\text{设备选择电路}]{\text{选中目标地址}}$ CPU开始读写操作

2. I/O设备(aka外设)

2.1. 外设概述

- 1 基本组成

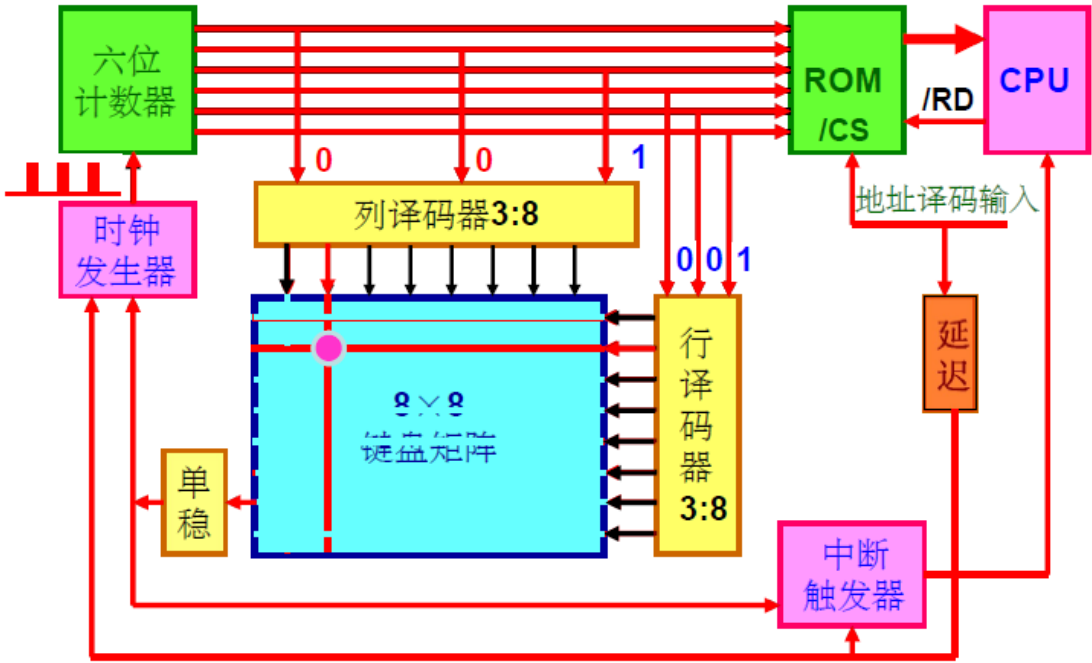


- 1. 设备控制器: 由电子电路实现, 控制设备具体行为
- 2. 设备的物理结构: 由光磁等实现, 和主机迥异

- 2 类型

类型	含义	示例
人机交互	信息交流 操作者 \longleftrightarrow 计算机	键盘/鼠标/打印机/显示器
信息驻留	保存信息	磁盘/磁带/光盘
机机通信	实现计算机系统间通信的设备	调制解调器/直流交流转换

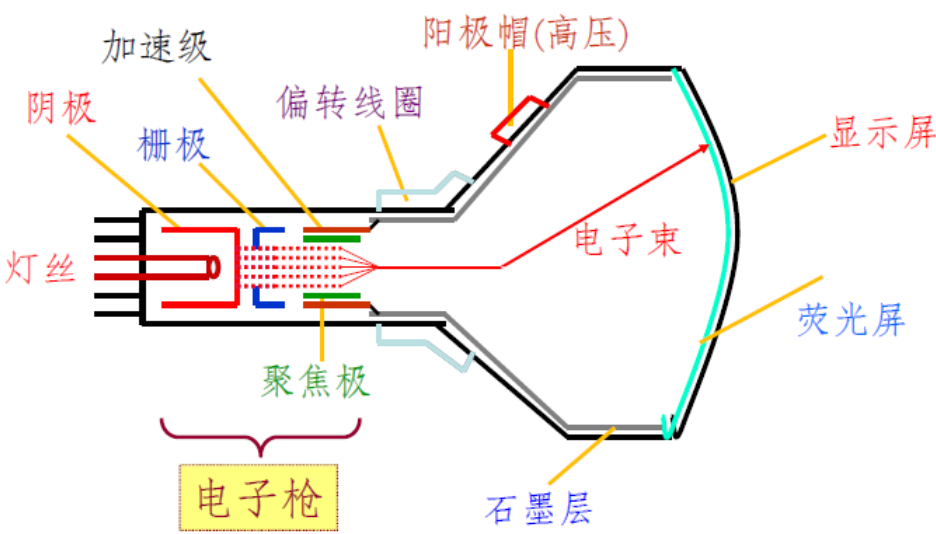
2.2. 键盘



- 1 结构：键盘以阵列方式排布，横竖交叉处放置一个键开关
- 2 键开关：位于键盘表面之下，用于将键盘敲击转化为电信号

2.3. 阴极射线管CRT显示器

2.3.1. 概述



- 1 技术指标

技术指标	含义	格式
像素	显示的亮点	#
分辨率	总像素数	每行像素数×每列像素数
灰度级(黑白)	像素点的亮暗级差	#
灰度级(彩色)	像素点的颜色种数	#

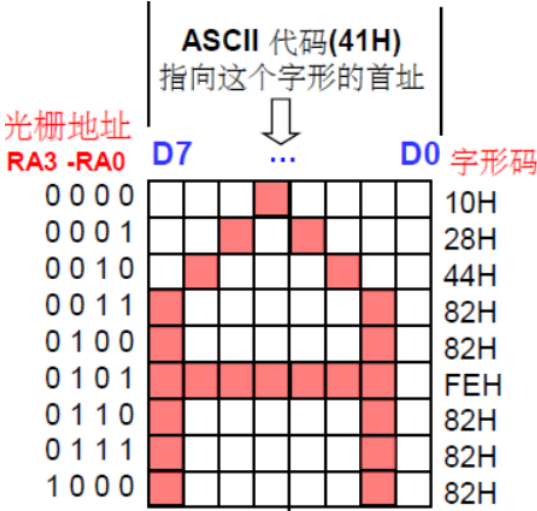
2 光栅扫描

扫描方式	含义	特点
随机扫描	图像和字符在哪，就扫哪	显示快速，驱动复杂
光栅扫描	周期性全屏扫描，分逐行和隔行两种	旧式电视普遍采用，但显示质量不佳

3 刷新：周期性地对屏幕重复进行扫描，有赖于刷新存储器/视频存储器/VRAM

2.3.2. 字符显示

2.3.2.1. 字符显示的点阵结构

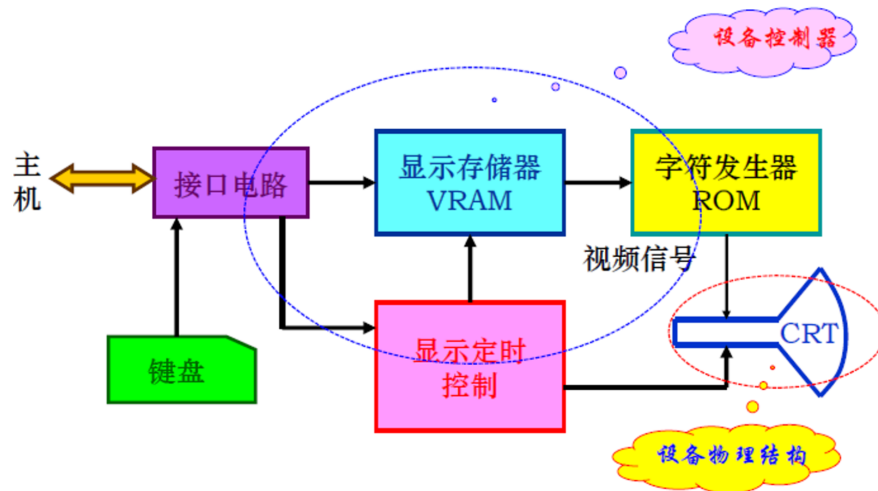


1 字符窗口：如上图中8×9的范围

2 光栅地址：就第几行

3 字形码：把每行八位分为两个四位，点阵占用为1，例如第三行为
0010 1000B=28H

2.3.2.2. CRT字符显示器构造

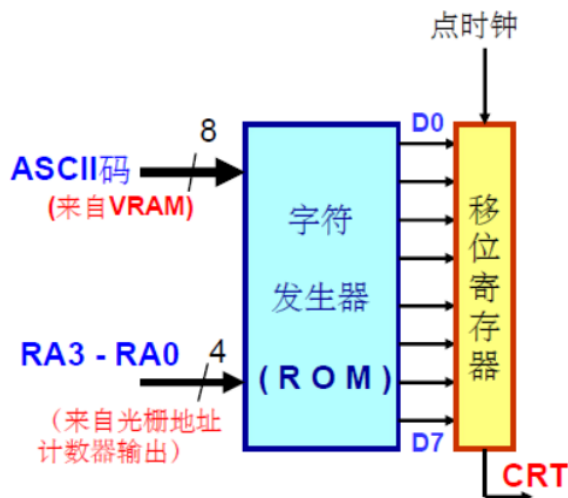


1:显示存储器VRAM:

1. 功能：存放了一屏中所有字符的ASCII信息
2. VRAM字符地址：注意在这里，排代表字符层面的行，行代表像素层面的行

高位地址	低位地址	VRAM字符地址
字符在屏幕中的排	字符在屏幕中的列	字符所在行×一行有多少字符+字符所在列

2 字符发生器ROM



1. 功能：存放字符点阵的二进制编码矩阵，根据字符的ASCII读出字符点阵
2. ROM地址

地址	高位地址	低位地址/光栅地址
含义	字符的ASCII码	字符点阵的行号
来源	VRAM输出	光栅地址计数器输出

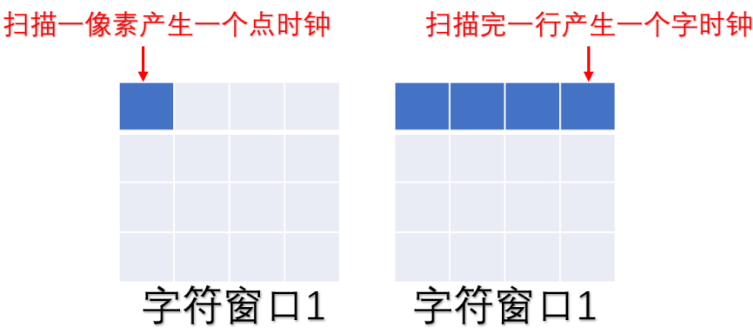
2.3.2.3. 字符显示的定时控制

1 各个计数器简介

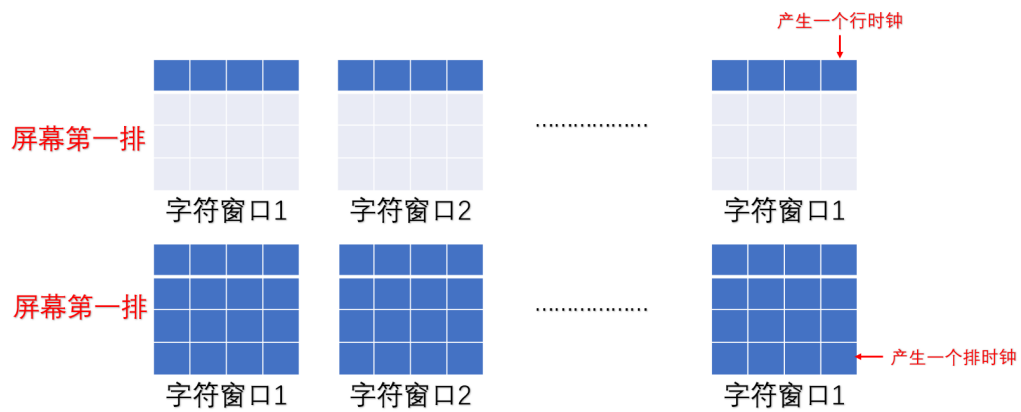
部件	功能	模长	输出
点振荡器	1. 产生点时钟，频率 \propto 分辨率/帧率	1	点时钟 →点计数器
点计数器	1. 计算窗口每行多少像素 2. 计完一行输出一个字符时钟	字符窗口宽度	字符时钟 →字计数器 字符时钟 →移位寄存器 加载
字计数器 水平地址计数器	1. 计算屏幕每行多少字符 2. 计完一行输出一个行时钟	一排字符数 + 水平回扫时间	行时钟 →行计数器 计数值 →VRAM 低址
行计数器 光栅地址计数器	1. 计算窗口多少行像素 2. 计完一个窗口输出一个排时钟	字符窗口高度	排时钟 →排计数器 排时钟 →ROM低 计数值 址
排计数器 垂直地址计数器	1. 计算屏幕多少行字符 2. 计完一次垂直扫一次(帧时钟)	屏幕字符行数 + 垂直回扫时间	计数值 →VRAM 高址

2 图示计数器工作过程

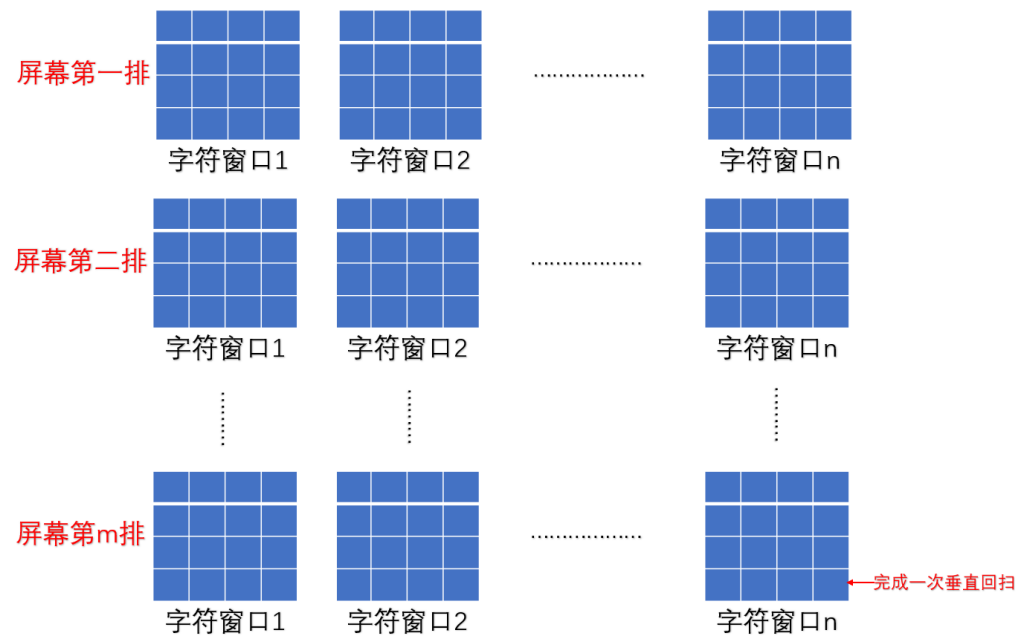
1. 点振荡器(左)&点计数器(右)



2. 字计数器&行计数器(最右边1改成n)



3. 排计数器



3 各个工作频率的关系

频率	含义	大小
帧频	扫完所有排耗时	帧频
排频	扫完单排耗时	帧频 × 排计数器模
行频	扫描单行耗时	帧频 × 排计数器模 × 行计数器模
字频	扫描窗口内一行耗时	帧频 × 排计数器模 × 行计数器模 × 字计数器模
点频	扫描完一像素耗时	帧频 × 排计数器模 × 行计数器模 × 字计数器模 × 点计数器模

2.3.2.4. 字符显示例题

0 条件

1. 某CRT显示器可显示64种ASCII字符
2. 每帧可显示64字×25排
3. 每个字符字形采用7×8点阵
 - 横向7点，字间间隔1点，这一点间隔也一起存在ROM中
 - 纵向8点，排间间隔6点
4. 采用逐行扫描：帧频50Hz/行频24.5KHz/点频14.896MHz

1 VRAM至少有多大？

1. VRAM缓存了一屏中所有字符的ASCII信息，VRAM也就是所谓的缓存
2. 64字×25排的意思，一共有25每行有64个字，一屏最多64×25个字符
3. 而字符使用ASCII编码，故一个字符需要8位，故大小至少为 $64 \times 25 \times 8 \text{bit} = 1600\text{B}$

2 ROM至少有多大？

1. ROM存放了字符的所有点阵
2. 一共有64中字符，每个字符点阵大小(加上间隔)为8×8，所以大小为 $64 \times 8 \times 8 \text{bit} = 512\text{B}$

3 VRAM缓存地址与屏幕显示位置如何对应

显示字符	左→右	上→下
缓存地址	步长为1 小→大	步长为1行所显示字符的数量 小→大

4 计算各个计数器分频

1. 首先：分频是指通过模长将频率降低多少倍，模=k ⇔ k:1分频
2. 其次：根据模大小定义，先算出连个计数器的分频

计数器	模大小	模值	分频
点计数器	字符窗口宽度	8(点阵宽)	8 : 1
字计数器	屏幕一行字符数+水平回扫时间	水平回扫时间不可知	#
行计数器	字符窗口高度	8+6(点阵高+排间距)	14 : 1
排计数器	屏幕字符行数+垂直回扫时间	竖直回扫时间不可知	#

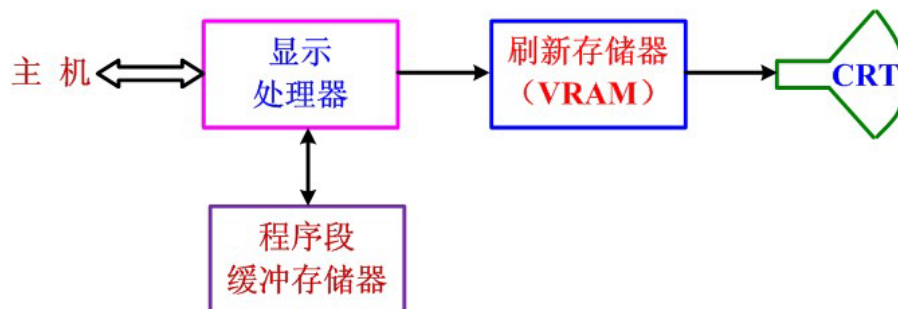
3. 再代入如下式子

点频	=	帧频	×	排模	×	行模	×	字模	×	点模
14.896MHz	=	50Hz	×	X	×	14	×	Y	×	8

点频	=	帧频	×	排模	×	行模	×	字模	×	点模
行频	=	帧频	×	排模	×	行模				
24.5KHz	=	50Hz	×	X	×	14				

$$\text{解得} \begin{cases} X = 35 \\ Y = 76 \end{cases}$$

2.3.3. 图像显示

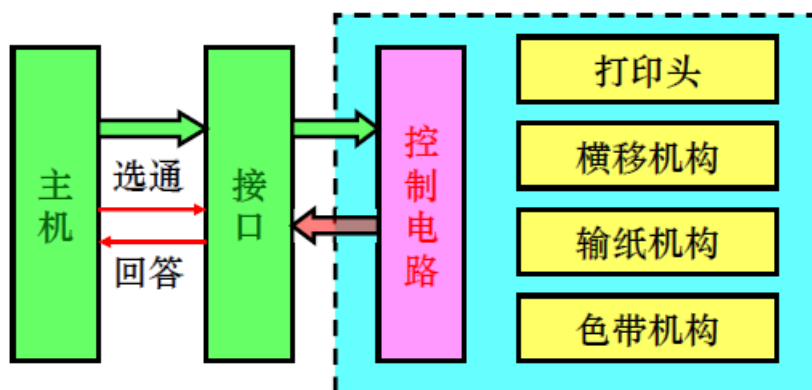


1 刷新存储器容量 = 像素数 * \log_2 (灰度级 aka 像素颜色数)

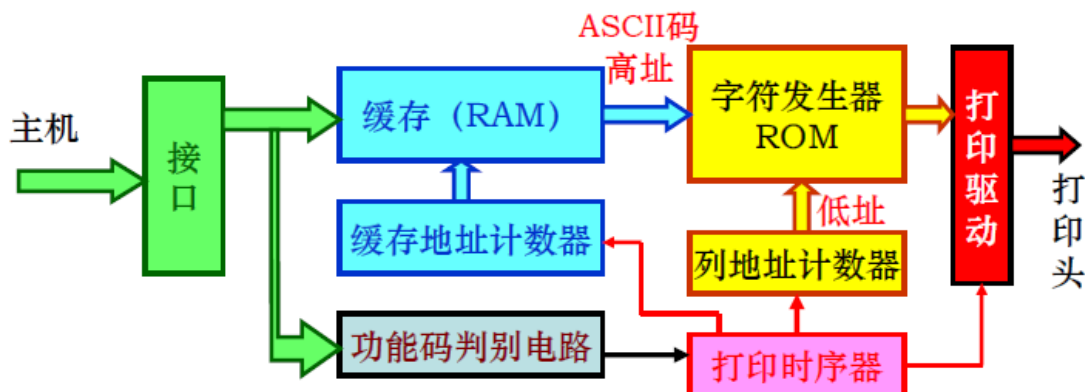
2 客观图像/摄像机取景 $\xrightarrow[\text{算法}]{\text{画出点线阴影等}}$ 主观图像/计算机图像

2.4. 点阵打印机

1 总体结构



2 控制电路结构：和CRT字符显示器都采用点阵，所以结构相似



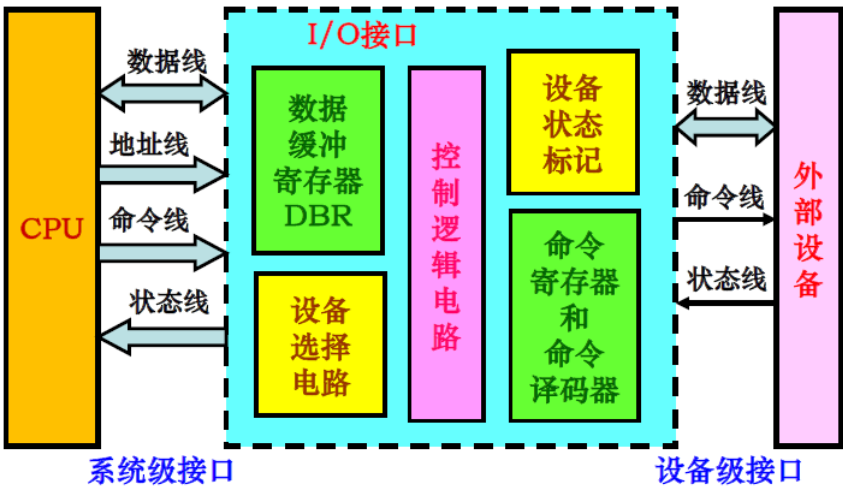
3. I/O接口：I/O设备^{I/O接口}↔CPU

3.1. I/O接口作用

功能	技术基础
CPU $\xleftrightarrow[\text{I/O接口}]{\text{速度匹配}}$ I/O设备	数据缓冲
CPU $\xleftrightarrow[\text{I/O接口}]{\text{数据格式转换}}$ I/O设备	串并转换电路
CPU $\xleftrightarrow[\text{I/O接口}]{\text{电气转换}}$ I/O设备	电平匹配
CPU $\xrightarrow[\text{I/O接口}]{\text{单向控制}}$ I/O设备	接收/传达控制命令
CPU $\xrightarrow[\text{I/O接口}]{\text{单向查询}}$ I/O设备	保存/传送I/O状态
CPU $\xrightarrow[\text{I/O接口}]{\text{I/O寻址}}$ I/O设备	设备选择电路

3.2. I/O接口结构：双连界面

1 接口内部总线与部件

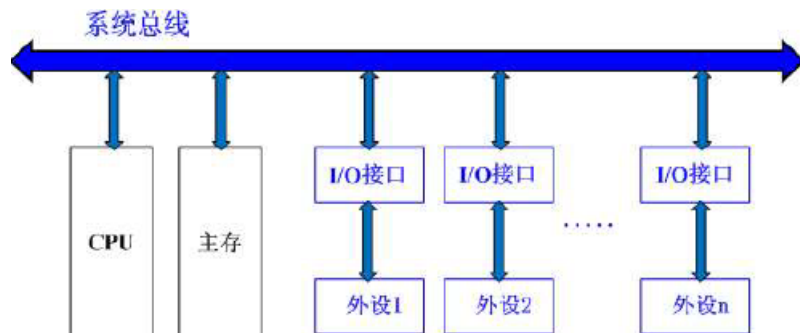


I/O接口有关总线	功能	连接部件
数据线	传输数据，以字/字节为单位	数据缓冲寄存器DBR
地址线/设备选择线	传送设备地址，位数等于设备地址位数	设备选择电路
命令线	传输控制命令，如外设读/写/启动/清除	命令寄存/译码器
状态线	传输设备状态，如忙/闲	设备状态标记(触发器)

🌈 一条指令/地址通路



2 接口的外部连接



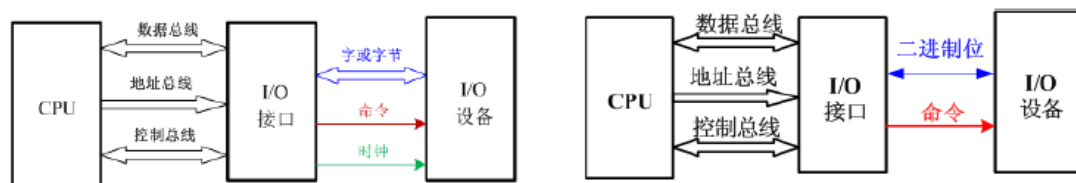
1. I/O接口↔主机(CPU): 总线结构

2. I/O接口↔I/O设备: 一字/一字节的多位同时并行传输, 或者一位一位串行传输

3.3. I/O接口^{通讯}↔I/O设备

1 同步通信: 由时钟同步(框定啥时候互相要干啥), 通讯双方无需应答信号

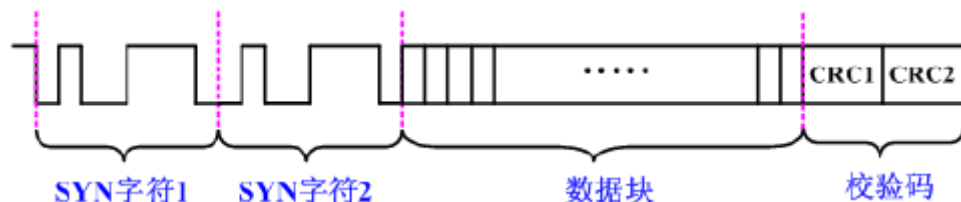
1. 串行/并行的连接方式: 区别在于一次传一字/还是一位



同步并行接口

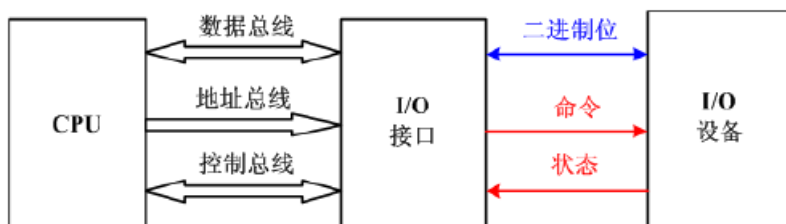
同步串行接口

2. 同步串行数据格式: 先传送两SYN作为起始标志, 然后一数据块一数据块传数据, 最后校验



2 异步通信: 二者间用应答方式联络

1. 连接方式: 去掉时钟, 但加上状态线(用于将I/O设备状态发往I/O接口和CPU)



2. 数据格式: 没有同步时钟后, 需要在格式里加入同步信息(起始位/终止位)

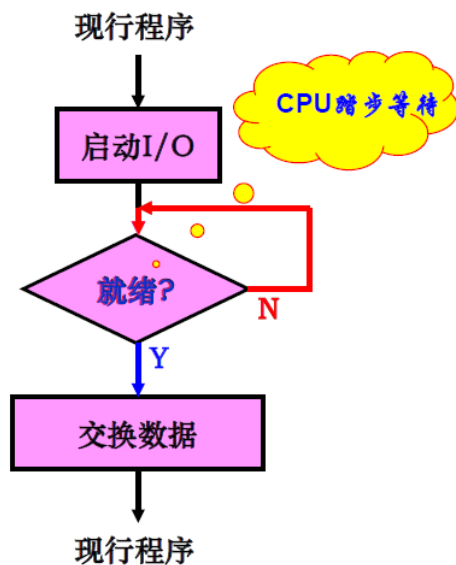
4. I/O数据传送方式

4.1. 程序查询方式

4.1.1. 基本思想

1 原理：CPU要执行I/O时，往现行程序里塞一段I/O指令编制的程序，运行塞入程序以I/O

2 处理流程：CPU干瞪眼



4.2. 中断程序方式

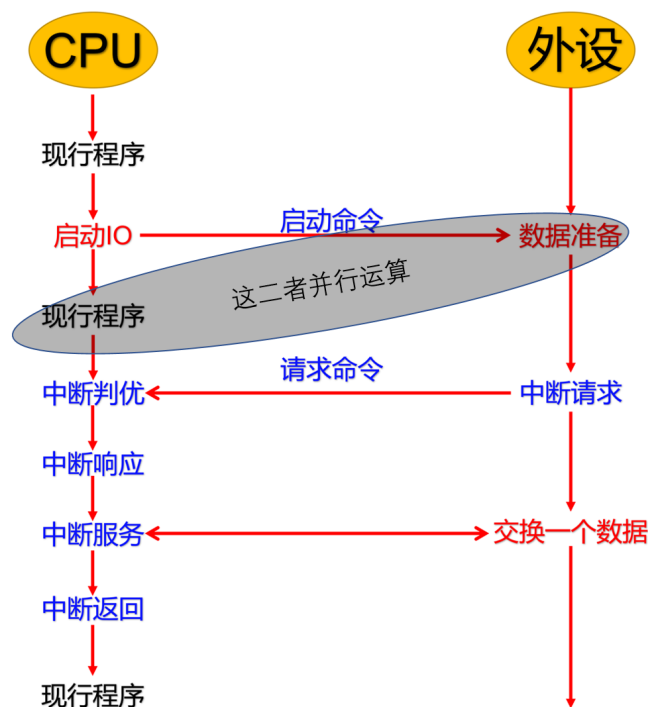
4.2.1. 中断概念

1 中断/程序中中断：计算机执行被紧急事件打断，CPU暂停程序去处理事件，处理完后继续执行

2 中断技术：软硬件结合

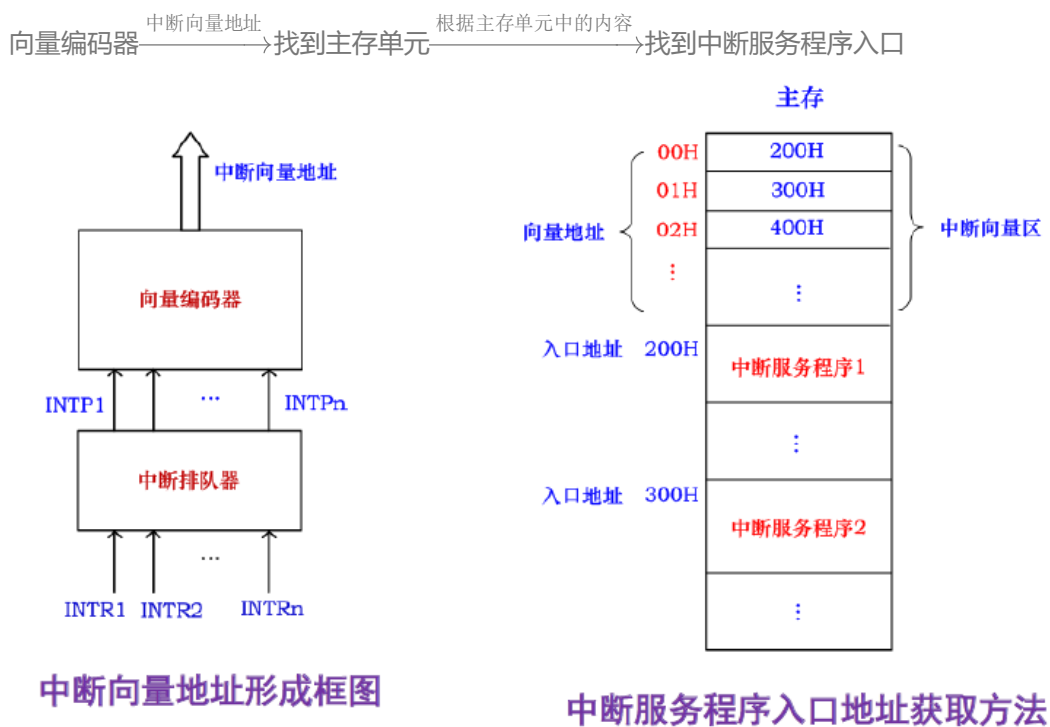
3 I/O中断：由I/O操作引发的中断

4.2.2. 中断处理流程



步骤	描述
请求	<div> <div> <div>外设(INTR触发器)</div> <div>请求信号 ↓ 存储/发出</div> <div>→</div> <div>CPU(EINT触发器)</div> <div>允许信号 ↓</div> <div>→ CPU决定理不理这个中断</div> </div> </div>
判优	<div> <div>INTR1/INTR2...INTRn(请求队列)</div> <div>得到优先队列 ↓ 串行/并行排队器</div> <div>→ INTP1/INTP2...INTPn</div> </div>
响应	CPU完成: 关中断, 入栈 程序断点和PSW, 找到中断服务入口(软件查询/硬件向量)
服务	保护现场, 中断处理(数据交换), 现场恢复
返回	CPU完成: 出栈 断点和PSW, 开中断

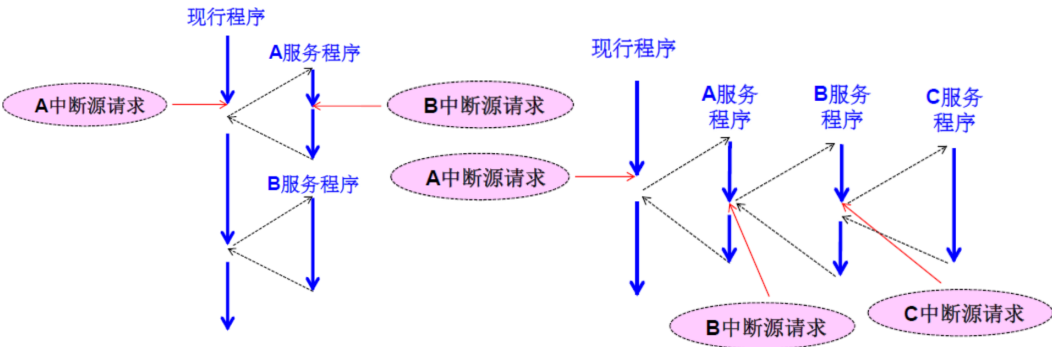
4.2.3. 中断向量



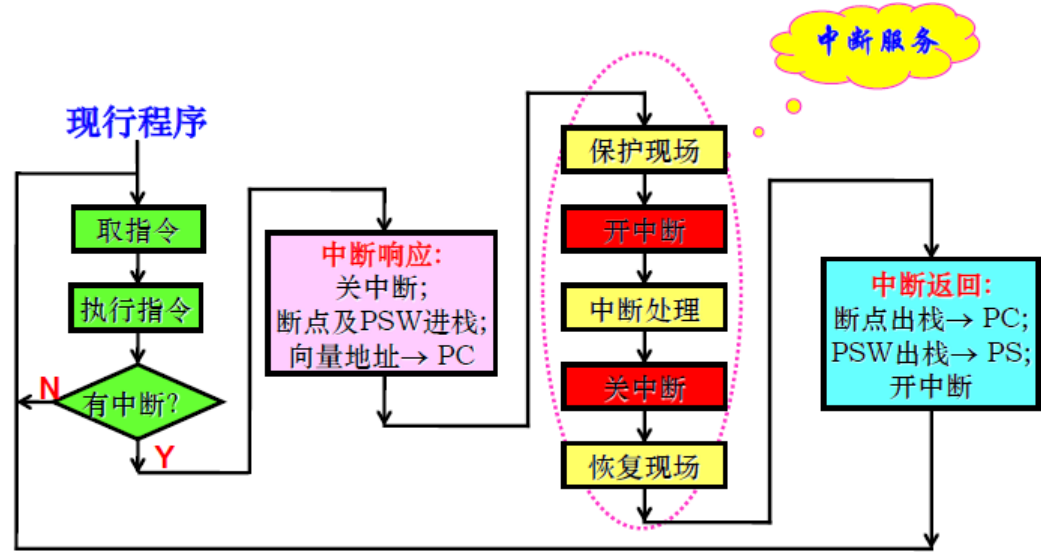
4.2.4. 多重中断

1 单重/多重中断：在处理当前A中断时，新的B中断来临，该如何处理

中断类型	优先级	执行操作
单重中断	$B \leq A$	执行完A中断后再去执行B中断
多重中断	$B > A$	中断A中断的处理，转而处理B中断，处理完后再回到A中断

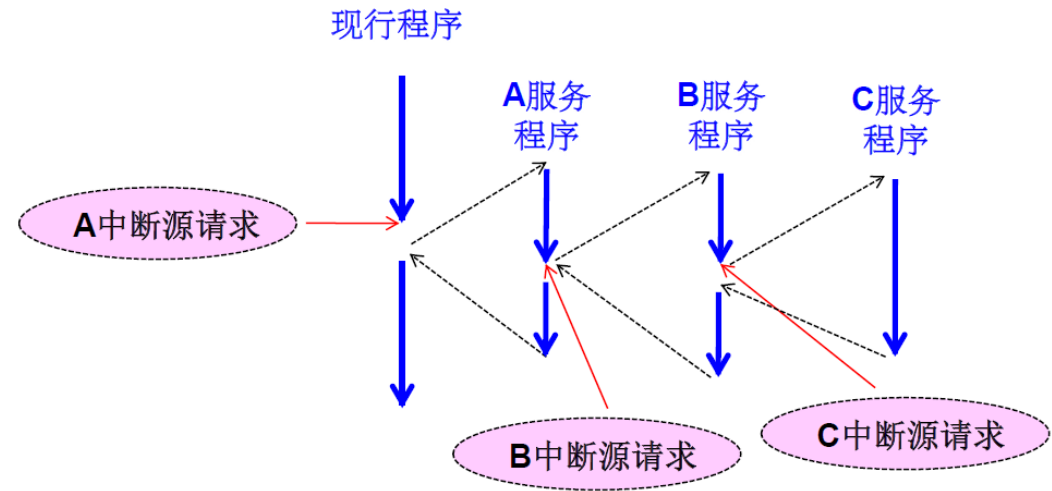


2 多重中断实现的条件：中断服务中插入中断的开关操作



4.2.5. 中断嵌套

1 嵌套规则：优先级低的中断，可以嵌套执行高的，如下图→是嵌套方向(A套B套C)



2 实现技术：

技术	决定了说明
优先级	中断的响应次序
屏蔽字	中断的处理次序

3 屏蔽字设置：假设预想的中断处理次序为A → B → C → D → E则屏蔽字为

中断源	E	D	C	B	A
A	1	1	1	1	1
B	1	1	1	1	0
C	1	1	1	0	0
D	1	1	0	0	0
E	1	0	0	0	0
现行程序	0	0	0	0	0

1. 规则1: $X \rightarrow Y$ 表示X优先级高于Y，则下表(为1表示可以屏蔽/默认自己可以屏蔽自己)

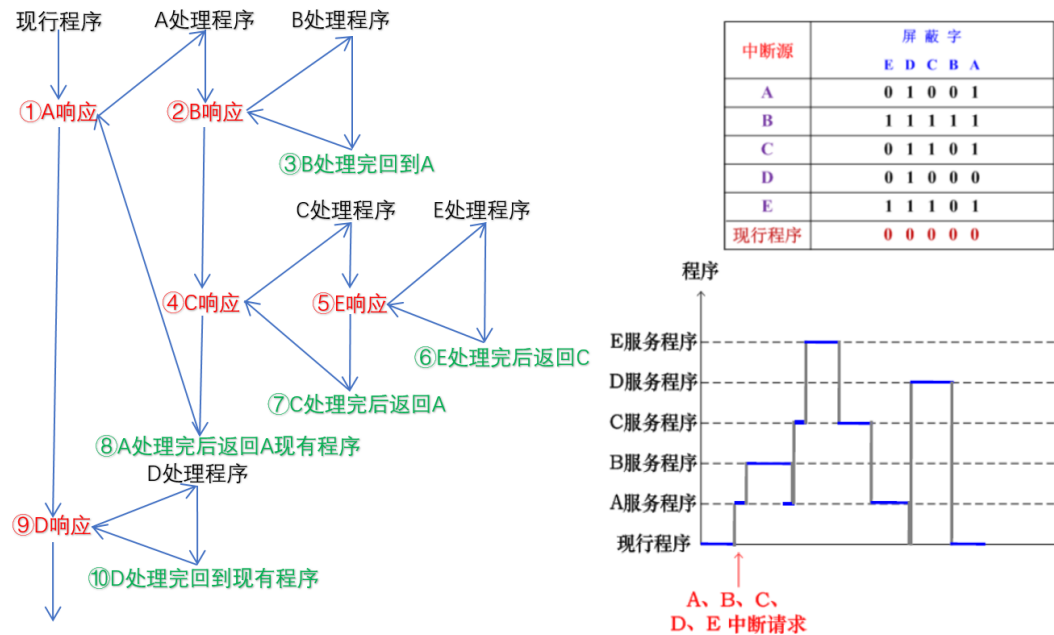
中断源	X	Y
X	1	1

2. 规则2: $Y \rightarrow Z$ 表示Z优先级下于Y，则下表(为0表示不可以屏蔽)

中断源	Y	Z
Z	0	1

3. 总的来说就是以处理次序为准，左边的都记0，右边的都记1，自己记1

4 嵌套实例：优先级ABCDE，处理次序BECAD



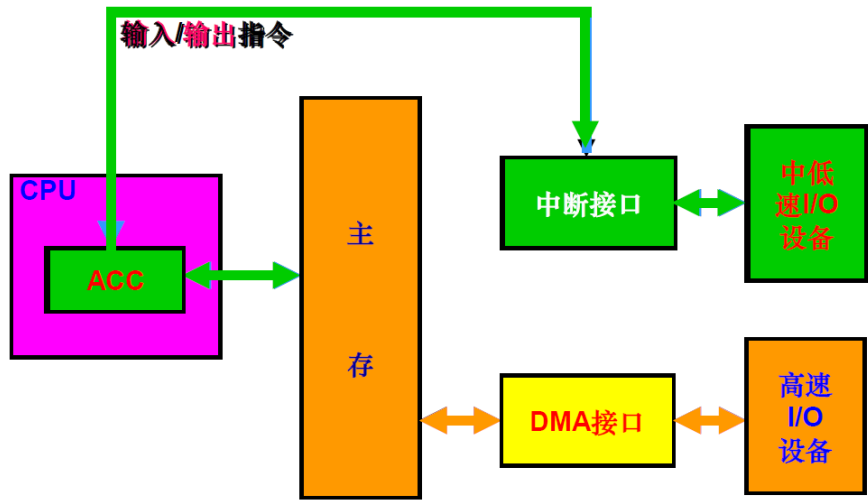
阶段	正在运行的程序	正在请求的中断(按优先级排序)	被屏蔽的中断	响应的中断
①	现有程序	ABCDE	#	A
②	A中断处理	BCDE	DA	B
③	B中断处理	BCDE	EDCBA	#
④	A中断处理	CDE	DA	C
⑤	C中断处理	DE	DCA	E
⑥	E中断处理	D	EDCA	#
⑦	C中断处理	D	DCA	#
⑧	A中断处理	D	DA	#
⑨	现有程序	D	#	D

4.3. DMA(直接存储器访问)方式

4.3.1. DMA功能: I/O $\xleftrightarrow[\text{DMA}]{\text{直接/自动/横批地传输数据}}$ 主存

1 目的: 而减少CPU干预

2 和中断方式结构对比如下



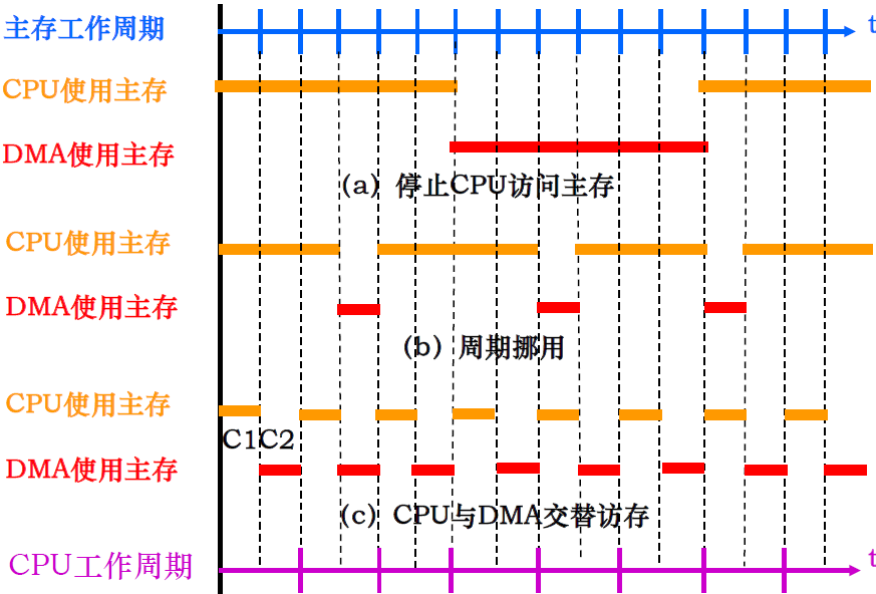
4.3.2. DMA数据交换流程

阶段	操作
预处理	CPU $\xrightarrow{\text{初始参数(主存地址/传送字数)}}$ DMA控制器, 然后CPU回去执行原程序
数据传输	1. I/O备好数据+CPU给出总线权 2. 主存 $\xleftrightarrow[\text{一个个地}]{\text{传输数据}}$ DMA, 传输过程无CPU参与 3. 传完一批后DMA控制器 $\xrightarrow{\text{中断}}$ CPU

阶段	操作
后处理	CPU响应中断，执行中断程序以结束I/O

高亮部分可能会发生DMA接口和CPU同时访问主存的情况，冲突消解见下

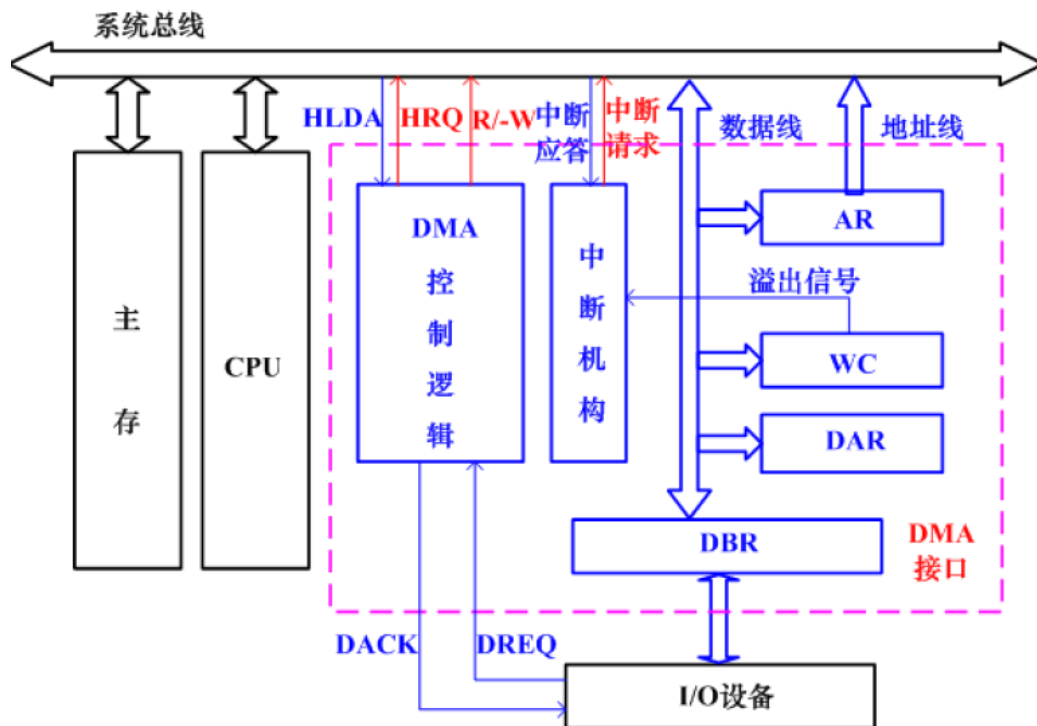
4.3.3. DMA接口和CPU的访存冲突消解



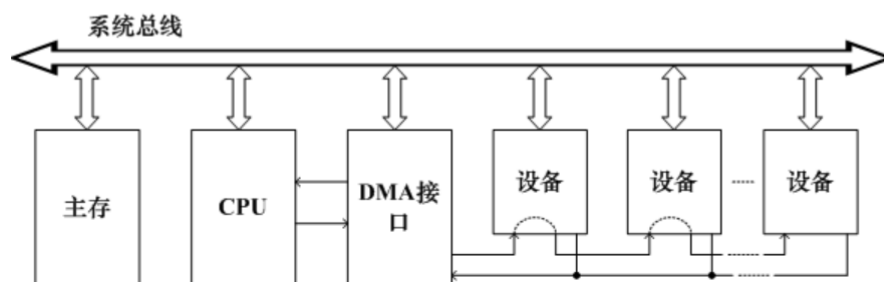
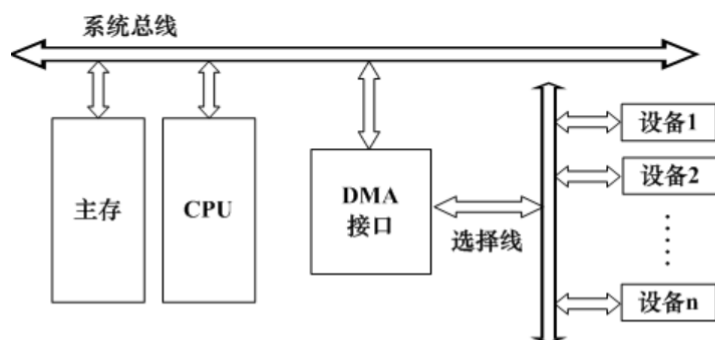
方式	操作
停止CPU访存	DMA接口要用主存时，直接暂停CPU现有的程序，让其踏步等待
周期挪用/窃取	CPU先不停，外设备好数据后发DMA请求，DMA接口然后才偷占小段周期
交替访存	将CPU工作周期划为两个主存周期，一个给CPU一个给DMA

4.3.4. DMA接口

1 结构：相当于DMA控制器+I/O接口，同时向下兼容了查询和中断接口的功能



2 种类:



种类	物理上	逻辑上
选择型(上图)	直连多个设备	同一时刻只为一个设备服务
多路型(下图)	直连多个设备	同一时刻可为多个设备服务

4.4. I/O方式例题

4.4.0. 大前提

1 CPU: 主频50MHz, 执行每条指令需要5个时钟周期

2 I/O: 外设只有键盘和硬盘

1. 用户敲击键盘速度时一秒五遍

2. 硬盘传输速率为5MBbs

4.4.1. 当外设都采用程序查询

1 CPU一秒执行的时钟数：50M

2 CPU每秒查询键盘所耗时钟

1. 键盘一共101键，每键查询一次要经历三个指令：读状态→判断是否按下→回送状态

2. 一秒内执行指令数101键×3指令×5遍，占用周期数

101键×3指令×5遍×5时钟每条指令

3. 比率 $\frac{101 \times 3 \times 5 \times 5}{50M} = 0.01515\%$

3 硬盘每秒查询所耗指令

1. 一秒要传输5MB大小也就是5M个存储单元，对每个单元需要执行两指令：命中→读出

2. 一秒内执行指令数5M存储单元×2指令，占用周期数

5M存储单元×2指令×5时钟每条指令

3. 比率 $\frac{5M \times 2 \times 5}{50M} = 100\%$ ，所以就不该采用软件查询

4.4.2. 当外设都采用中断方式

1 新前提

1. 中断服务：有9条指令，其他开销占1条指令时间

2. 硬盘：硬盘 $\xleftrightarrow{\text{记录块}}$ 主存，传输速率5MBbs，中断CPU一次交换一字节

2 求： $\frac{\text{CPU键盘数据传送时间}}{\text{键盘准备数据时间}}$

1. 处理一次中断，就是执行一次中断服务程序，耗时

(9指令+1间隔)×5时钟每指令=50时钟

2. 键盘一秒敲击5次，也就是每次敲的准备时间为0.2s=10M时钟

3. 比率为 $50/10^7 = 0.0005\%$

3 求： $\frac{\text{CPU硬盘数据传送时间}}{\text{硬盘准备数据时间}}$

1. 照样中断服务一次耗时50时钟

2. 一次交换一字节所以一秒钟要交换5M次，交换一次要准备1/5M秒也就是10时钟

3. 比率为 $50/10 = 5$ ，所以硬盘处理不能用中断

4.4.3. 当外设都采用DMA方式


1 新前提：只考虑硬盘

1. 每次DMA传送的数据为500B

2. DMA预处理和后处理一次，总开销为500个时钟

2 求CPU用于硬盘时间的占比

1. 将预处理→传输数据→后处理视为一次完整DMA传输，其中数据传输是不需要CPU插手的

- 
2. 数据传输耗时: $500\text{B}/5\text{MBps}=100\mu\text{s}$, 无需CPU
 3. 前后处理耗时: $500\text{时钟}/50\text{MHz}=10\mu\text{s}$, 需要CPU
 4. 所以占比为 $10\mu\text{s}/(10\mu\text{s}+100\mu\text{s})=9.09\%$