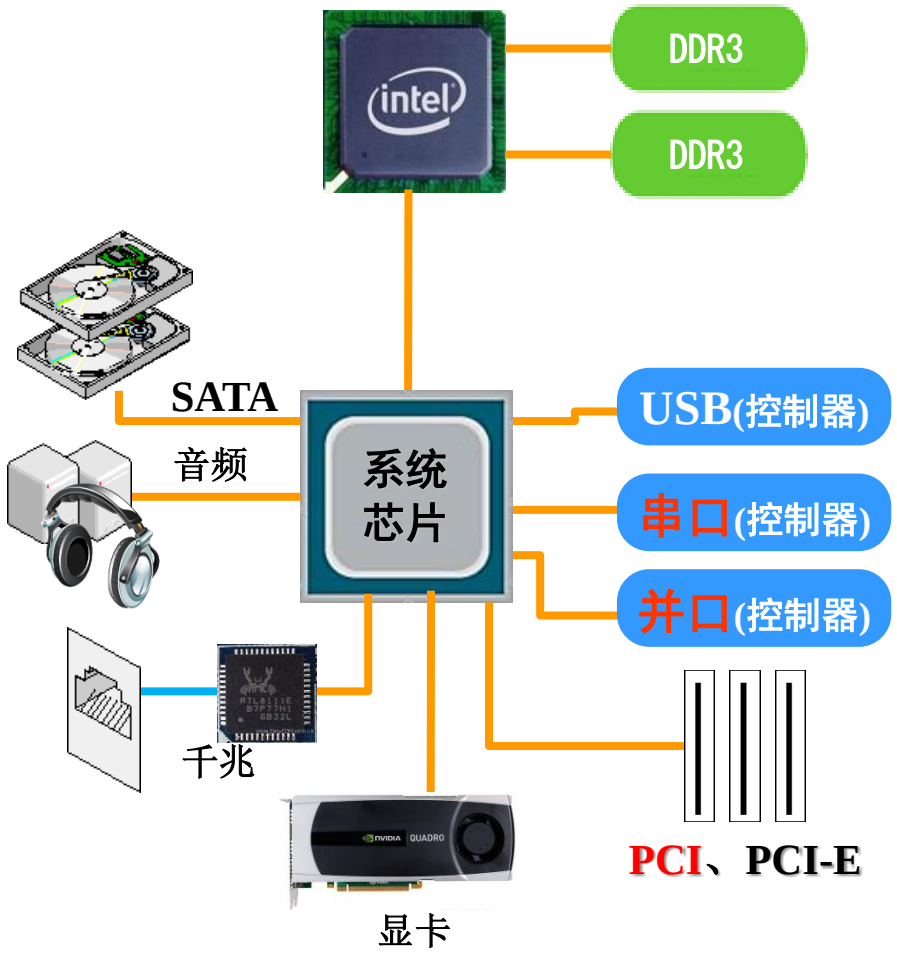
接口基础知识

高小鹏

系统结构研究所
北京航空航天大学计算机学院

计算机内部结构

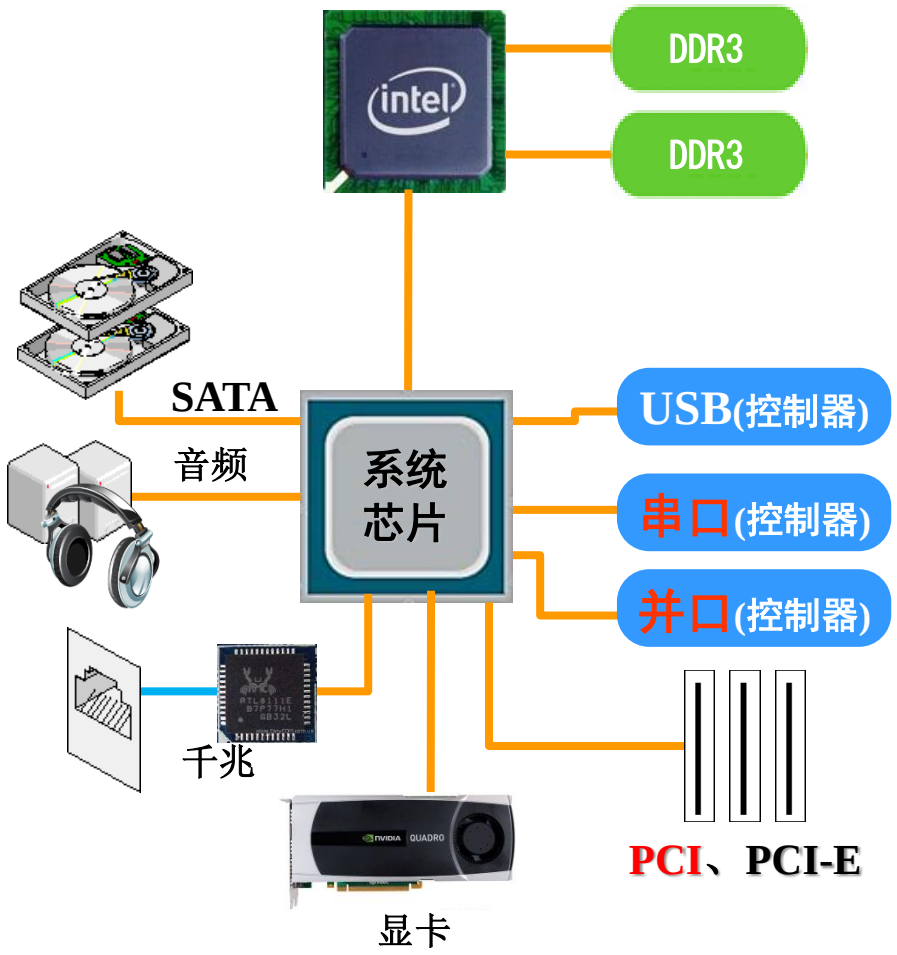
- CPU
  - 内存控制器(MC)
  - 集成显卡
- 系统芯片(主板芯片)
  - SATA、音频
  - PCI-E/PCI总线控制器
    - 32位PCI总线
  - USB2.0/3.0
  - 串口/并口
  - 定时器、中断控制器
- 网卡芯片(主板芯片)



设备：信息传输的通道

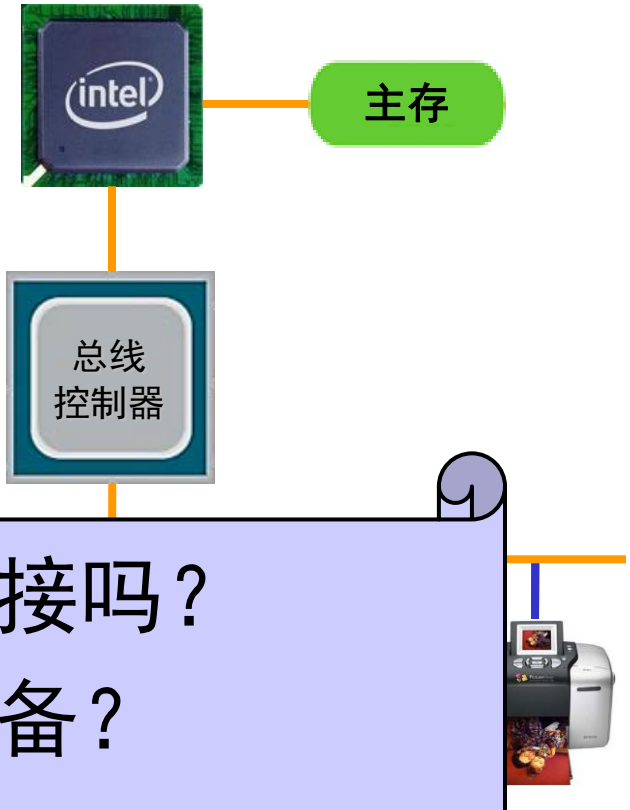
- CPU：信息处理
- 内存：程序/数据
- 设备：输入输出

- 设备：CPU与外界交换的通道
  - 信息从设备输入
  - 信息通过设备输出



计算机硬件结构的一般抽象结构

- CPU
- 系统桥/总线
- 主存
- 各类设备：磁盘/U盘/…



Q：设备能直接与主机连接吗？

Q：程序如何访问这些设备？

。 。 。

设备与主机相连需要解决的问题

- 硬件
  - 目标：CPU与外设(被控对象)在硬件上连接构成一个有机整体
  - 方法：I/O接口电路(接口、接口控制器)
- 软件
  - 目标：控制设备工作方式，完成信息传送
  - 方法：接口控制程序(或驱动程序)

- 接口技术 = 硬件 + 软件

需要回答的问题

- Q：设备如何接入计算机？
  - A：接口控制器(或接口电路，简称接口)
- Q：接口控制器呈现给软件何种界面？
  - A：接口控制器中的寄存器(或存储器)
- Q：程序如何访问接口控制器？
  - A：读写映射到CPU地址空间的接口控制器的寄存器
- Q：如何在CPU地址空间中分配地址？
  - A：每个设备占据地址空间中不冲突的区域
- Q：接口控制器如何判断程序是否读写自身？
  - A：对当前总线地址进行译码

问题1

设备如何接入计算机？

实例1：硬盘

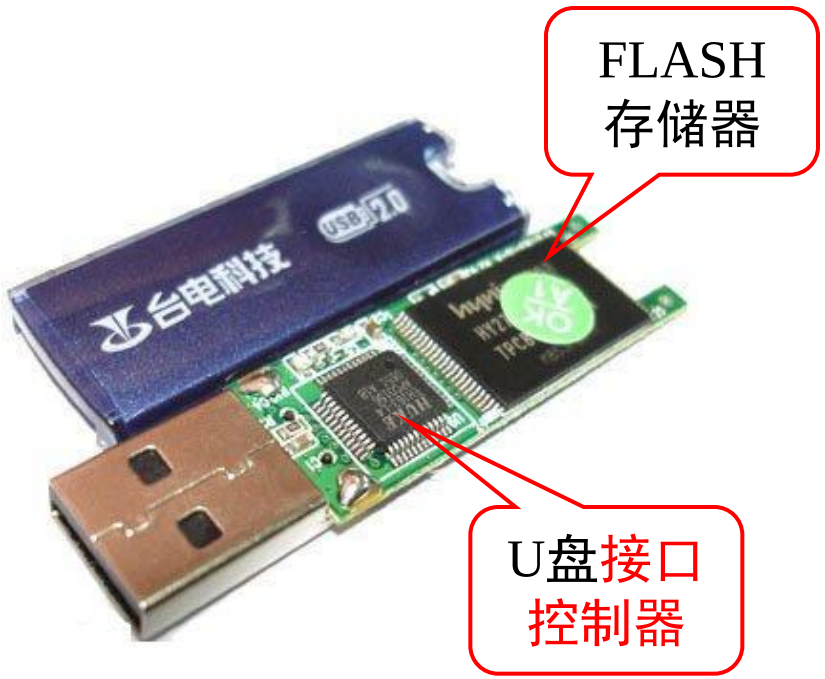
- SATA硬盘接口控制器
  - SATA总线接口
  - 磁盘接口
  - …
- 盘体(盘片/磁头/电机)

=

+

实例2：U盘

- USB接口控制器
  - USB总线接口
  - FLASH存储器接口
  - …
- 大容量FLASH存储器



设备通过接口控制器与主机连接

- 设备实际上包括两部分
  - 接口控制器（也称为接口芯片）
  - 设备主体
- 设备主体不直接与主机连接，而是通过接口控制器与主机连接

为什么设备必须通过接口控制器与主机相连？

- 速率不匹配
  - 主机：Intel i7 四核 3.2GHz
  - 键盘：100Hz
- 协议及时序不匹配
  - 主机：FSB、PCI-E、PCI
  - 硬盘：SATA
  - WLAN：802.11a/b/g
- 数据格式不匹配
  - 主机：32位/16位/8位，Bps（byte per second）
  - 网络：1位，bps（bit per second）
- 信号电平不匹配
  - 主机：TTL、CMOS、。。。
  - 电机：24V

接口控制器的功能1/2

- 地址译码
- 接受CPU编程
  - 设置模式、控制过程、查询状态
- 缓冲(或缓存)数据
  - 解决CPU与设备间速度不匹配的矛盾
- 时序控制
  - 匹配主机端总线工作要求
  - 匹配设备端信号工作要求

接口控制器的功能2/2

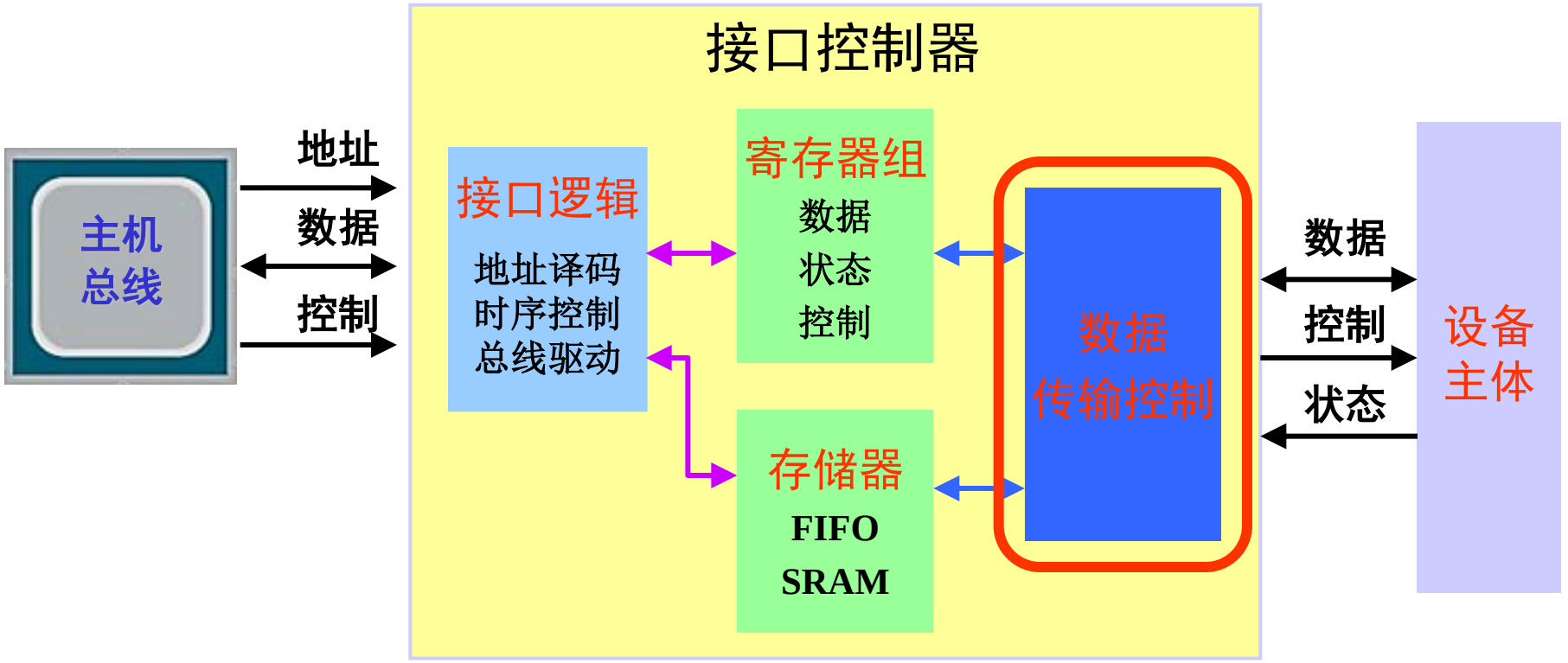
- 与设备主体交换数据
- 信号变换功能
  - 信号电平、逻辑极性
- 中断管理功能
  - 以中断方式接收和发送数据，降低CPU无效开销
  - 原因
    - 数据接收/发送过程必然涉及设备状态检查与判断
    - CPU工作频率极高。相对于设备一次数据传输时间，CPU能够执行大量指令

接口控制器的基本组成

- 总线接口逻辑
  - 地址译码
  - 时序控制
  - 总线驱动
- 寄存器组（或内部存储器）
  - 寄存器组：数据、状态、控制
  - 存储器：缓冲区、队列等
- 设备数据传输控制
  - 与特定设备的传输协议相关

总线–接口控制器–设备

- 接口逻辑、寄存器组是必须具备的
- 存储器是可选的
  - 主要用途：增强数据传输性能
- 不同类型设备的数据传输控制差异极大



接口控制器与总线间的接口信息

- 标准总线接口
  - 主机内部总线：PCI-E、PCI
  - 主机扩展总线：USB、1394
- 主机内部总线
  - 地址信号
  - 数据信号
  - 控制信号

接口控制器与设备间的接口信息

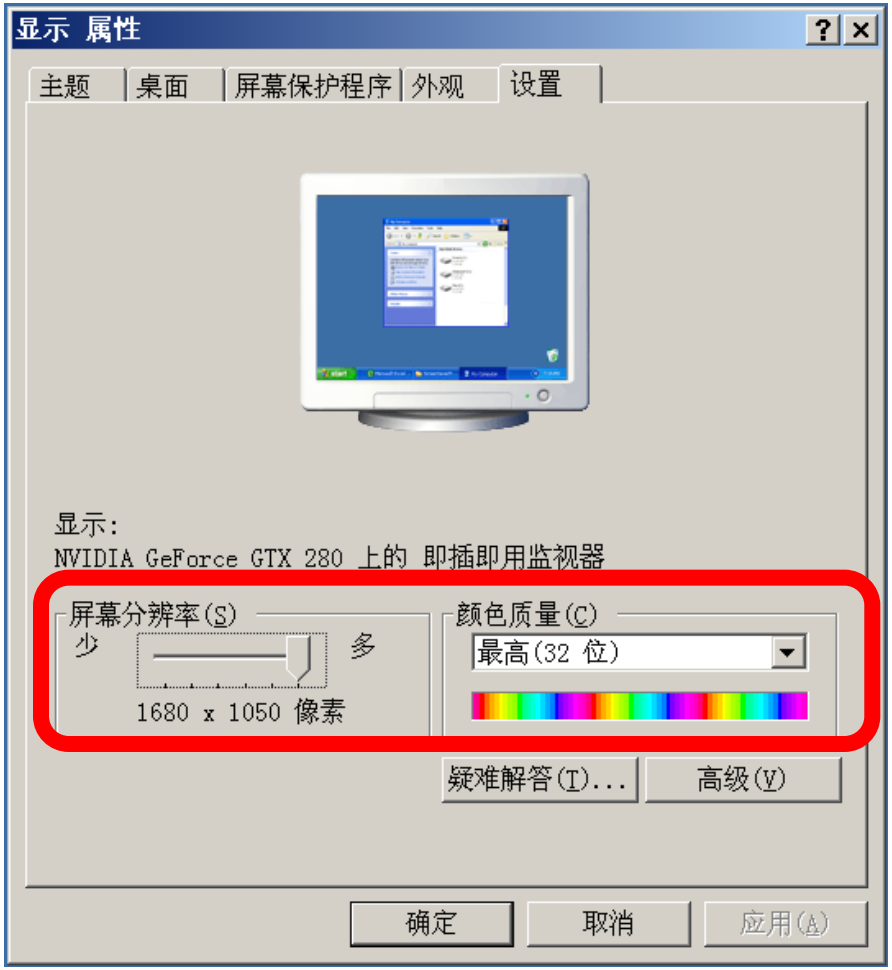
- 数据信息
  - 数字量、模拟量
- 状态信息
  - 反映设备当前工作状态（输入是否就绪，输出是否空闲）
  - 通常状态寄存器(SR)存储上述信息
- 控制信息
  - 按照预定工作时序工作，完成控制器设备间数据交互

问题2

接口控制器
呈现给计算机何种界面？

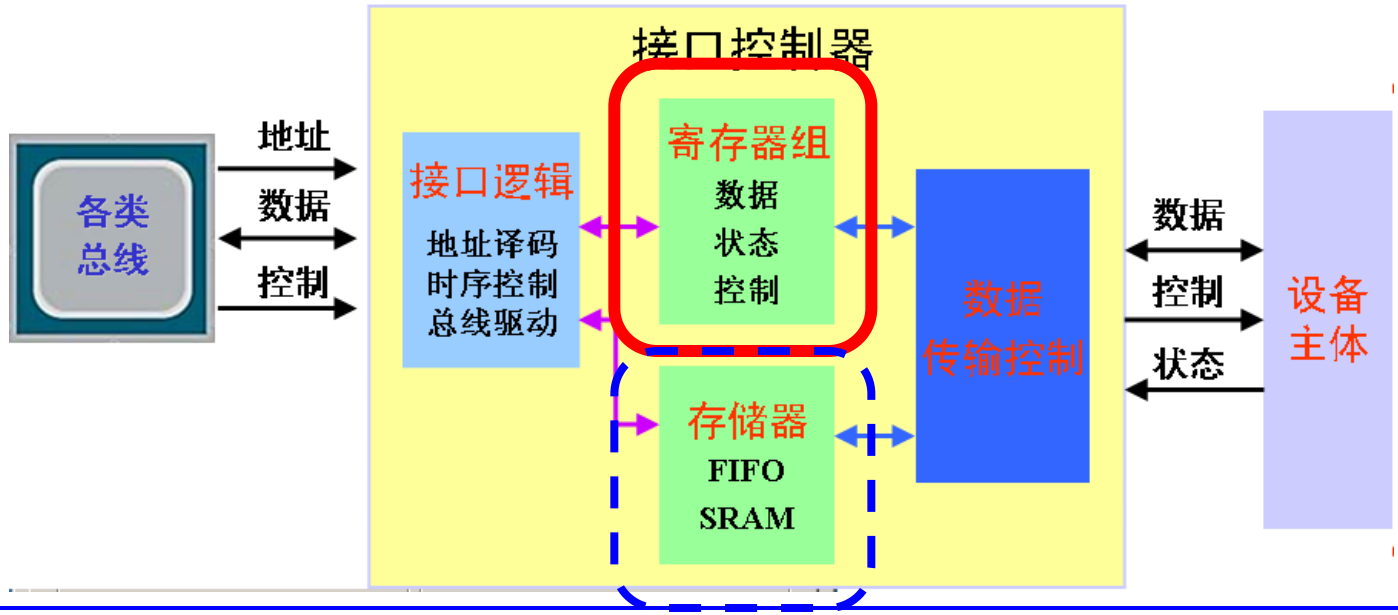
举例：调节显示器

- 例如通过显卡控制LCD
  - $R_{分辨率}$： 调节分辨率
  - $R_{色彩}$： 调节色彩



寄存器：主机与设备的交互界面

- 从信息处理的角度：主机 = 程序，即指令序列
- 指令的基础功能是读写某个单元
- 接口控制器的寄存器具备可读可写特性
  - 部分设备包括存储器。如显卡的帧缓存



问题3

程序中如何访问接口控制器？

通过例子认识程序员的视图

- Q：0x80000000是什么？
- Q：地址对应的是什么？
  - A1：某个内存单元
  - A2：显卡某个寄存器
- CPU必须通过地址访问内存(或设备)

```
unsigned int *p ;

p = (unsigned int *)
     0x80000000 ;

*p = 0xABCDEF12 ;
```

程序通过对某个地址访问设备

- CPU取指令及读写操作均需要给出地址
- 从程序员角度出发，地址 = 主存单元
  - 地址是CPU要访问主存单元的唯一定位信息
  - 当CPU发出某个地址，意味着CPU要读写对应主存单元
- 与主存类似，设备也必须拥有相应的地址

问题4

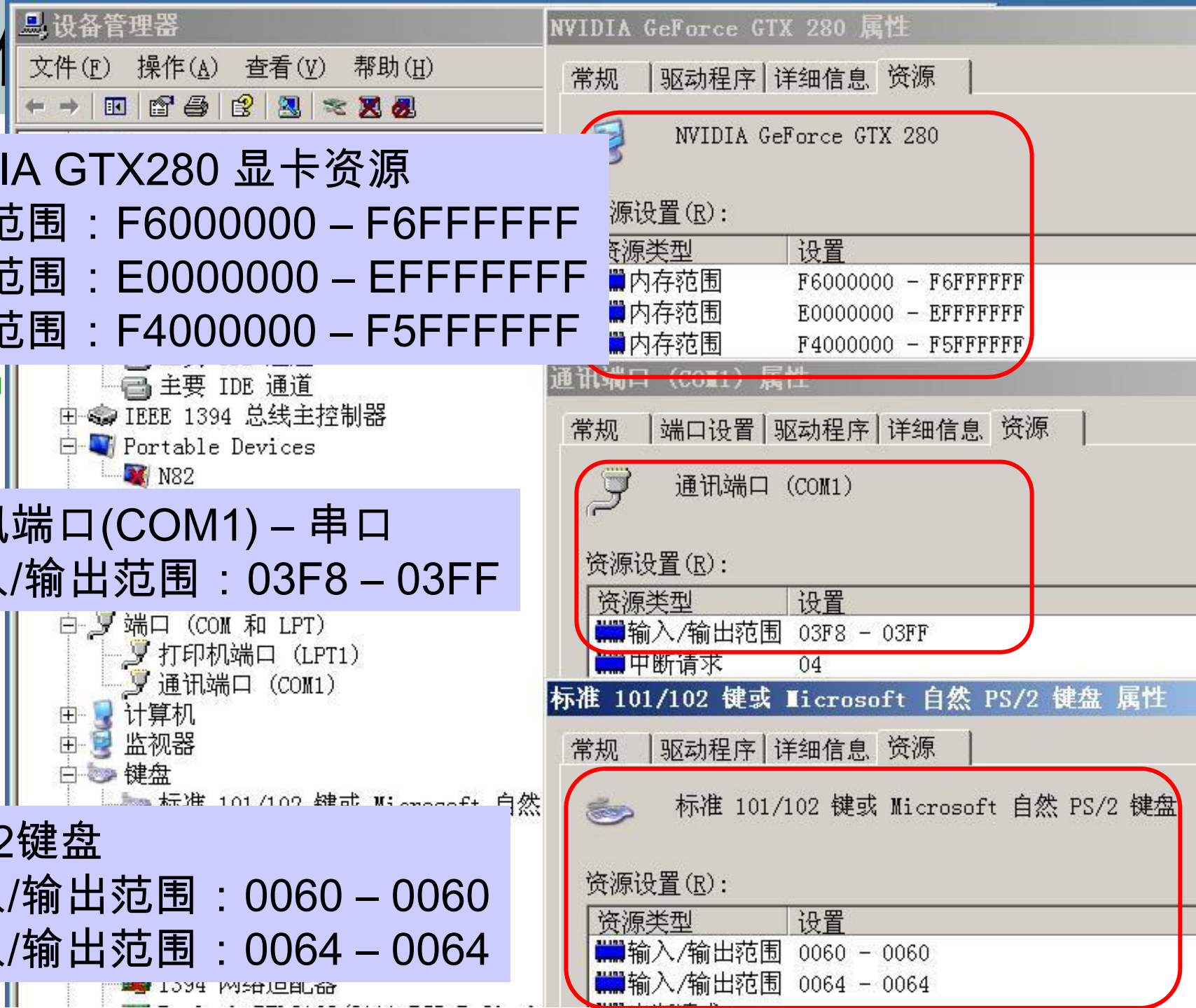
如何在CPU地址空间中分配地址？

实

NVIDIA GTX280 显卡资源
内存范围：F6000000 – F6FFFFFF
内存范围：E0000000 – EFFFFFFF
内存范围：F4000000 – F5FFFFFF

通讯端口(COM1) – 串口
输入/输出范围：03F8 – 03FF

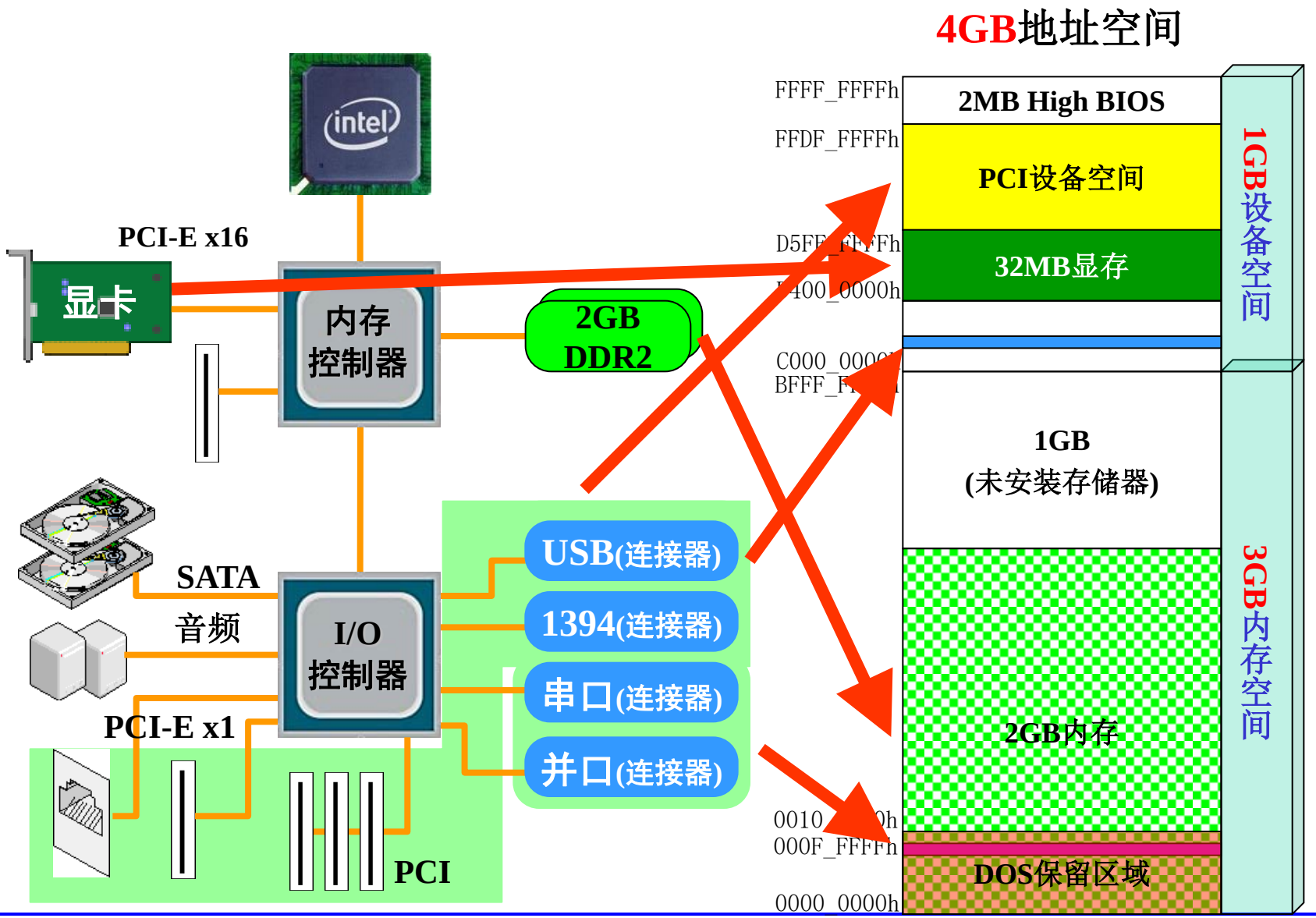
PS/2键盘
输入/输出范围：0060 – 0060
输入/输出范围：0064 – 0064



地址图

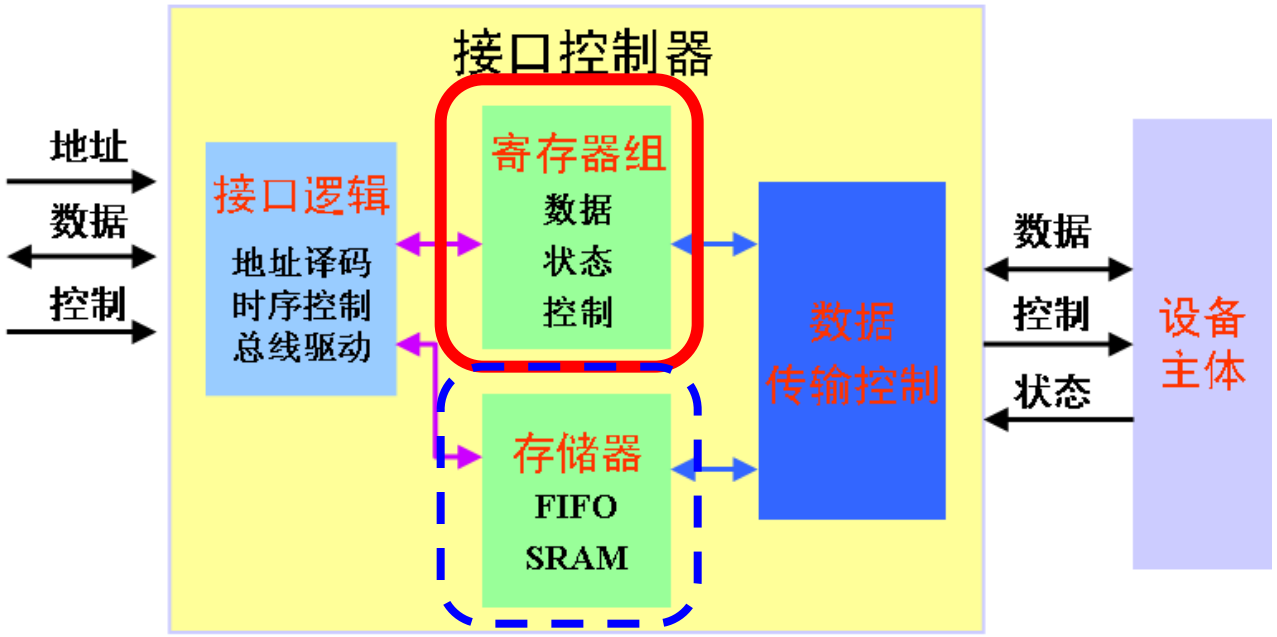
- 地址图：将CPU地址空间划分为若干区域
  - 区域←→部件
  - 主存：大量的地址范围
  - 显存：较大的地址范围
  - 串口：极少的地址范围
- 地址分配基本原则：任意两个部件的地址范围不能重叠

地址图


4GB地址空间
FFFF_FFFFh
2MB High BIOS
FFDF_FFFFh
PCI设备空间
D5FF_FFFFh
32MB显存
E400_0000h
C000_0000h
BFFF_FFFFh
1GB
(未安装存储器)
2GB内存
0010_0000h
000F_FFFFh
DOS保留区域
0000_0000h
1GB设备空间
3GB内存空间
PCI-E x16
显卡
内存
控制器
2GB
DDR2
SATA
音频
I/O
控制器
USB(连接器)
1394(连接器)
串口(连接器)
并口(连接器)
PCI-E x1
PCI

设备地址的具体内容

- Q：设备中哪些组成要素需要分配地址？
- A：寄存器组
  - 某些设备也包括隶属于接口的存储器，如FIFO、帧缓存



设备编址的分类

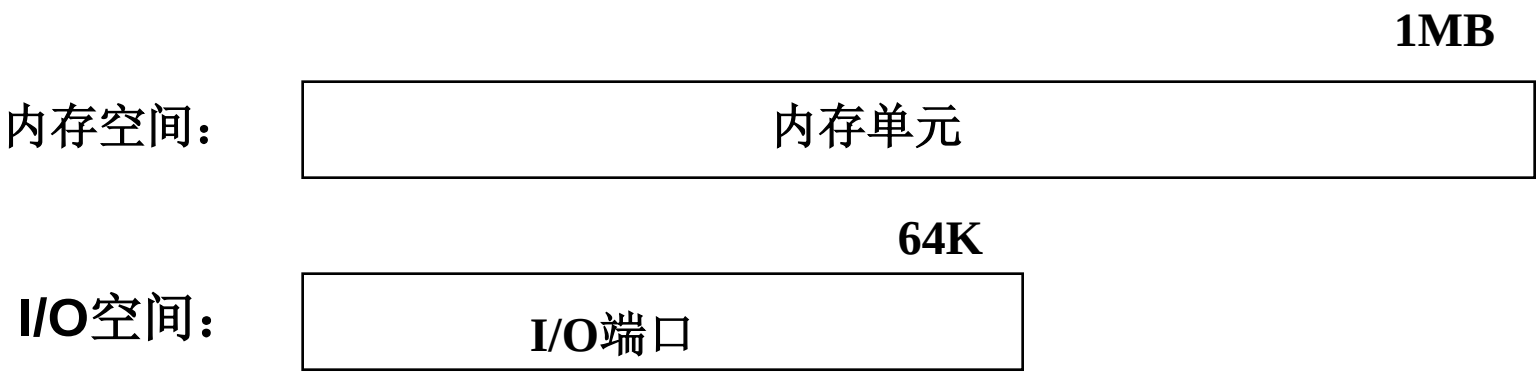
- 两大类设备编址方式
  - 存储器映射编址
  - 独立编址

存储器映射编址(Memory Mapping)

- 基本方法
  - 将设备的寄存器（或存储器）视为主存单元，并分配地址
  - 与主存单元一起在CPU地址空间统一分配地址
- 特点
  - 系统中只有一个寻址空间，即内存寻址空间
  - CPU所有的存储操作类指令均可应用于设备访问
  - 视图单一，模型简单

独立编址

- CPU有两个地址空间：存储器空间、I/O空间
  - 除存储类指令外，有专用的I/O类指令(IN，OUT)
  - 必须设置特殊信号来区分当前地址是访内存单元还是访问I/O端口(80x86 CPU：M/IO信号)



早期x86处理器地址空间模型

存储器映射编址 vs 独立编址

- 存储器映射编址：
  - 模型简单，软件可移植性好
  - 由于地址图略大，系统译码略微复杂些。但对于现代IC技术来说不是问题
  - 主流编址方式(PowerPC、MIPS …)
    - X86也采用。保留独立编址更多的是为了兼容

问题5

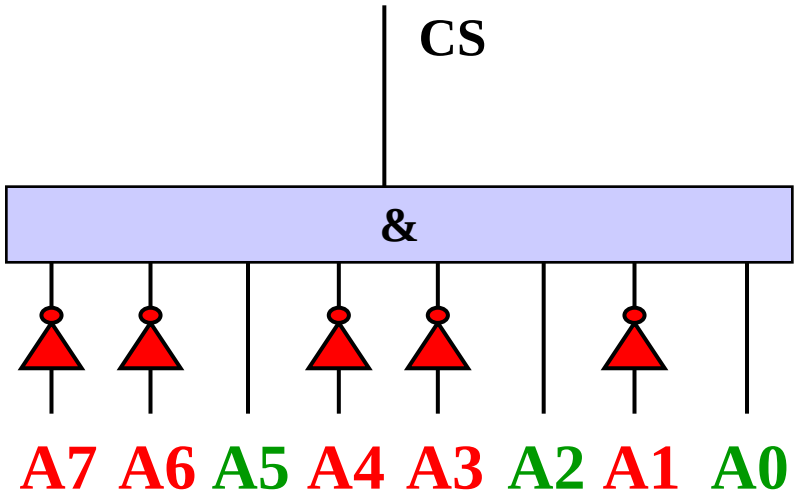
接口控制器如何判断程序是否读写自身？

地址译码

- 判断是否是某个特定地址
  - 假设8位地址：A =?= 25h
- 判断是否属于某个特定区域
  - 假设8位地址：A ∈ {20h～28h}

地址译码：与特定地址比较

- 判断A =?= 25h
- A = A7，A6，…，A0；25h = 0010_0101b
- CS = !A7 & !A6 & A5 & !A4 &
  !A3 & A2 & !A1 & A0



地址译码：与特定区域比较

- 判断A ∈ {28h～2Fh}
- 由于是判断是否属于属于某个区域，不需要判断具体是哪8个单元(28h~2Fh)，
  - 因此A2、A1、A0三位地址无需考虑
- 推广到一般：基地址(Base)+偏移范围(Offset)
  - 地址译码主要是比较Base部分
  - offset：用于内部寻址
- 地址区间{28h～2Fh}
  - 基地址：28h
  - 偏移范围：0～7，共计8个单元

地址译码：与特定区域比较

- A ∈ {28h～2Fh}，base = 28h = 0010_1000b
- 由于最低3位地址不参与比较，因此base = 0010_1xxxb
- 判断A ∈ {28h～2Fh} → A[7:3] =?= 0_0101b
- CS = !A7 & !A6 & A5 & !A4 & A3

例2：地址译码(与特定区域比较)

- Q：判断A ∈ {C0h～DFh}
- S1：分析起始地址
  - 起始地址：1100_0000b
  - 结束地址：1101_1111b
- S2：确定不变地址位
  - A7，A6，A5 (注意：偏移区域是满的)
  - base = 110x_xxxxb
- 判断A ∈ {C0h～DFh} → A[7:5] =?= 110b
- CS = A7 & A6 & !A5

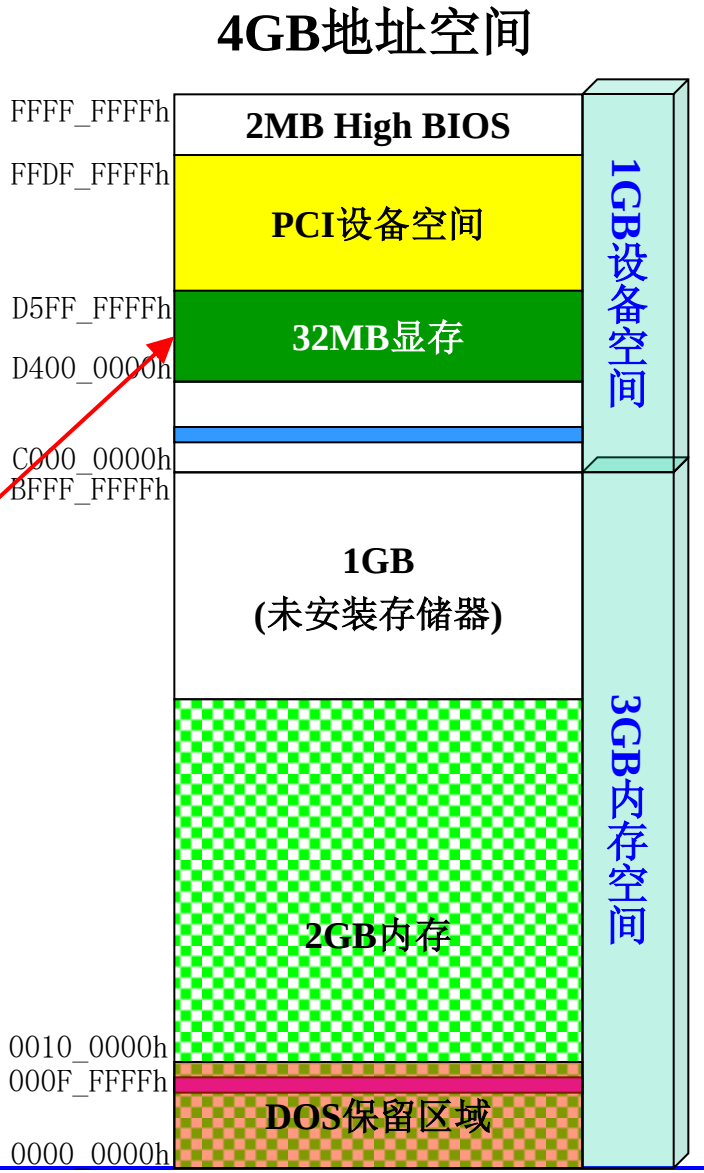
例3：地址译码(与特定区域比较)

- Q：判断A ∈ {C0h～D7h}
  - 起始地址：1100_0000b
  - 结束地址：1101_0111b
- 问题：偏移区域不是满的！
  - A7，A6，A5：不变；A4变；A3：不变
- 方法：拆分区域为若干满偏移区域
  - 等价于判断 A ∈ {C0h～CFh} | A ∈ {D0h～D7h}
- CS = A7 & A6 & !A5 & !A4 |
  A7 & A6 & !A5 & A4 & !A3
- CS = A7 & A6 & !A5 & (!A4 | A4 & !A3)

示例：程序写显卡寄存器过程分解

- 假设$R_{分辨率}$的地址是D500_0000h
- 通过编程$R_{分辨率}$设置分辨率
  - 1：800 * 600
  - 2：1024 * 768
  - …

```
unsigned int *p ;
p = (unsigned int *)
    0xD5000000 ;
*p = 0x00000002 ;
```

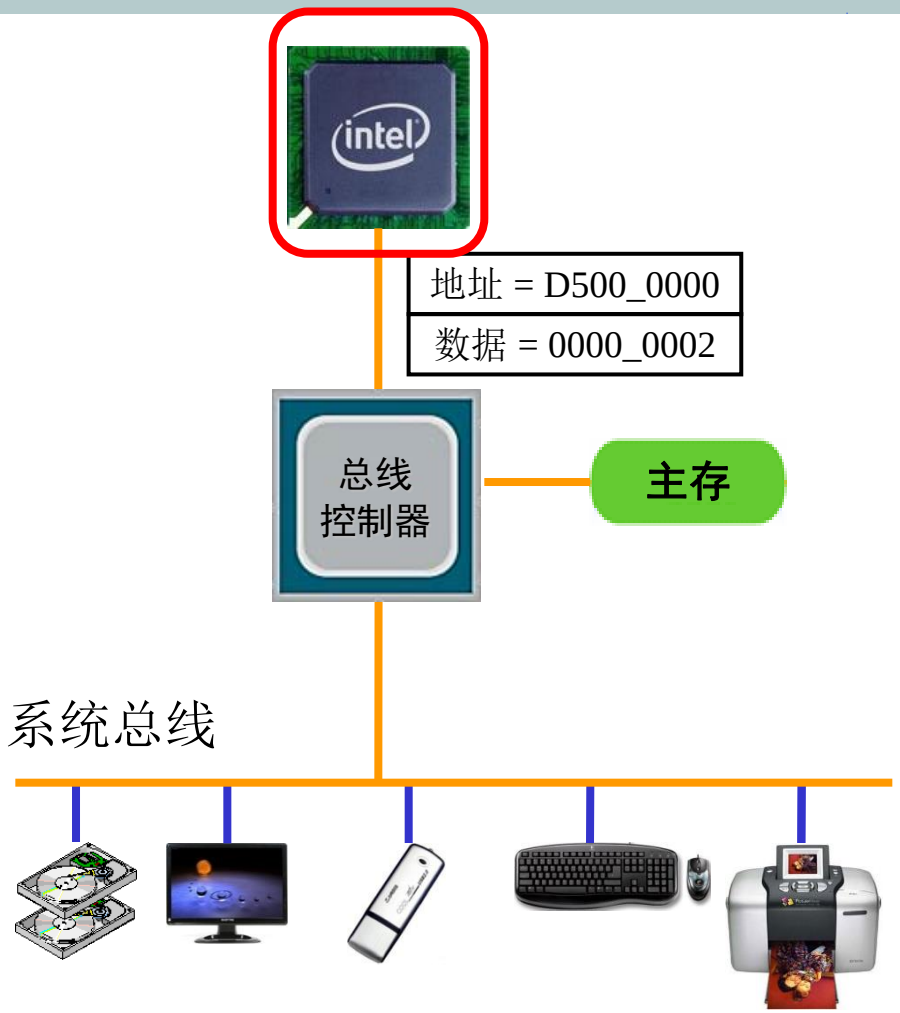


程序读写显卡寄存器过程分解

- S1：取指令
  - CPU输出地址，主存返回数据(实际为指令)
- S2：译码
- S3：取操作数
  - 操作数在寄存器中
- S4：ALU运算
  - 似乎没有需要运算的环节☺
- S5：写显卡寄存器
  - CPU输出地址及数据
  - 设备响应写入操作

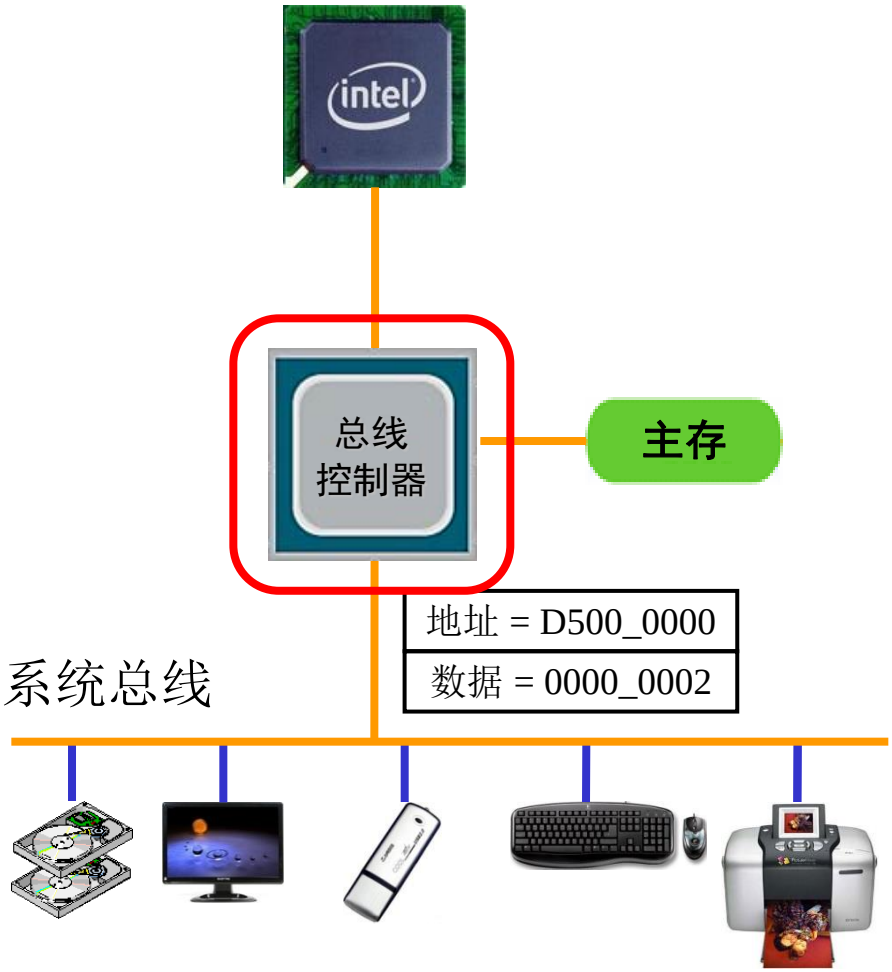
CPU

- CPU产生CPU的写周期
  - 地址总线：0xD5000000
  - 数据总线：0x00000002
  - 控制总线：R/W#为L



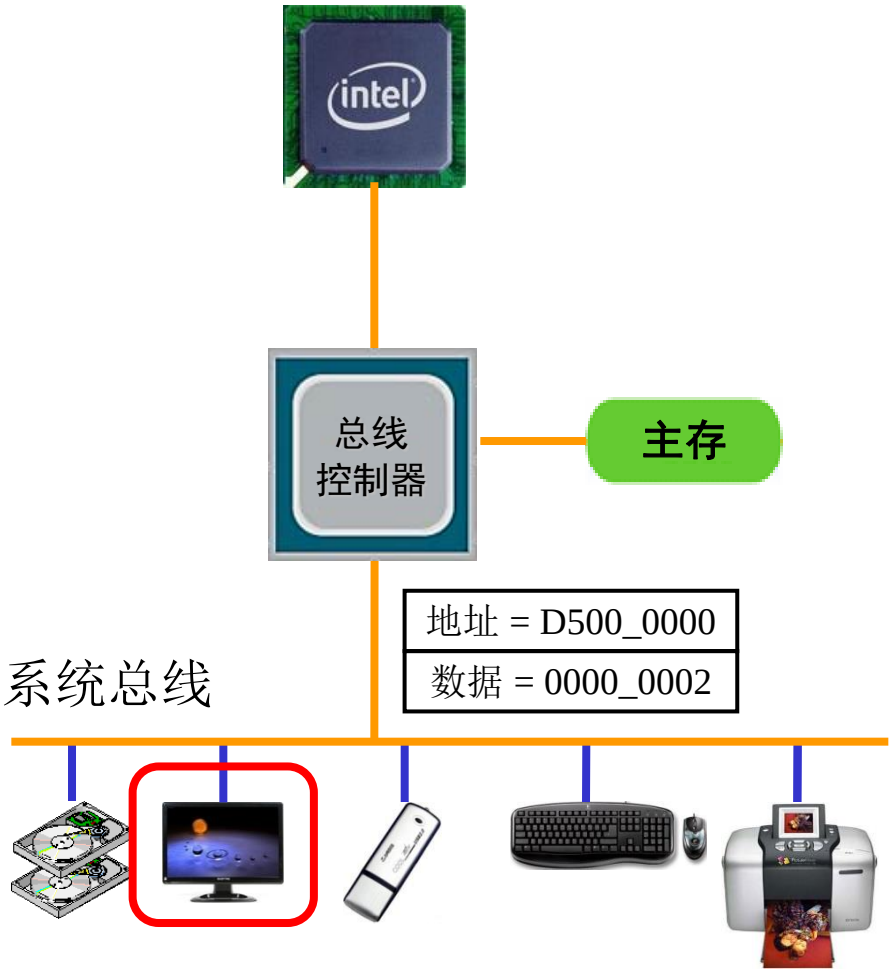
总线控制器

- 响应CPU总线写周期
- 查看地址图
  - 本质上也是译码
- 产生系统总线写周期
  - 地址总线：0xD5000000
  - 数据总线：0x00000002
  - 控制总线：R/W#为L



显卡

- 假设显卡已经知道自己的地址范围
  - 后面会继续讨论
- 与其他所有设备同时侦听总线，分析地址
  - 从地址总线获取地址
  - 地址 =?= D500_0000h
  - 将数据写入内部寄存器



小节

- 设备通过接口电路接入计算机
- 设备接口电路中的寄存器是软硬件界面
- 寄存器被映射到CPU地址空间，程序中对相应地址的读写指令在执行时，最终被系统生成相应的总线周期
- 设备通过地址译码判断当前总线周期
- 设备会在CPU地址空间中占有不同的区域，任意两个设备间的区域不能重叠