中山大學 软件工程学院

SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

1924-2024 中山大學 世纪华诞 100th ANNIVERSARY SUN YAT-SEN UNIVERSITY

设备管理

SSE202/204: 操作系统原理

苏玉鑫

suyx35@mail.sysu.edu.cn

助教：龙玉丹 单诗雯 毛晨希 沈志轩 郑灿峰 胡伟峰

版权信息

- 部分内容来自：上海交通大学并行与分布式系统研究所操作系统课件
  - https://ipads.se.sjtu.edu.cn/courses/os/
- 其它参考资料：
  - 清华大学操作系统公开课
    - https://open.163.com/newview/movie/courseintro?newurl=ME1NSA351
    - 介绍标准内容，适合考研
  - 南京大学计算机软件研究所
    - http://jyywiki.cn/OS/2025/
    - https://space.bilibili.com/202224425/channel/collectiondetail?sid=192498
    - 比较有趣

大纲

- 设备连接
- 设备类型抽象
  - 字符设备
  - 块设备
  - 网络设备
- 设备与操作系统的交互
  - 可编程I/O
  - 直接内存访问
- 操作系统如何响应设备：中断
  - 中断优先级
  - 硬中断
  - 软中断
- 操作系统如何管理设备
  - 驱动程序
    - 驱动模型
  - 设备树

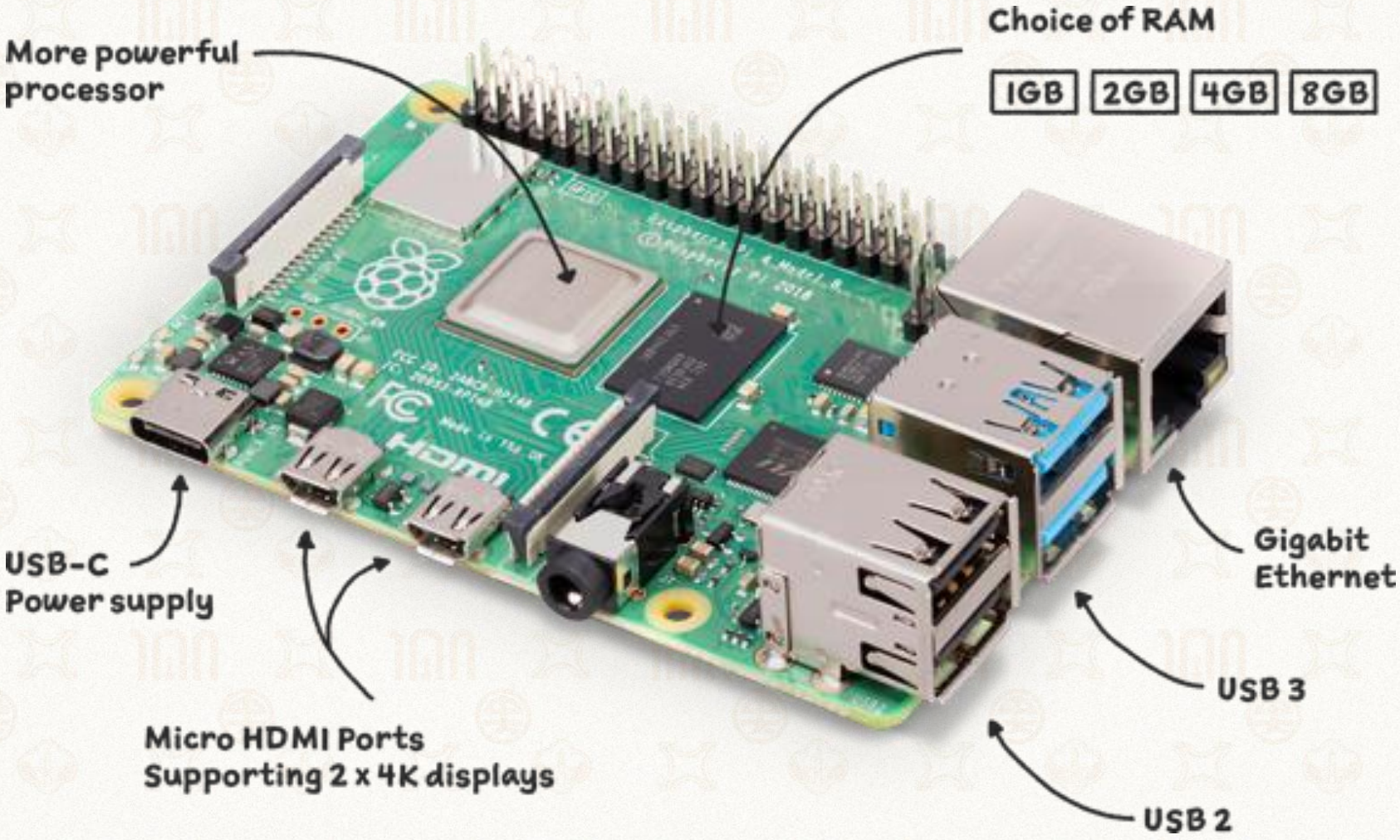
大纲

- 设备连接
- 设备类型抽象
  - 字符设备
  - 块设备
  - 网络设备
- 设备与操作系统的交互
  - 可编程I/O
  - 直接内存访问
- 操作系统如何响应设备：中断
  - 中断优先级
  - 硬中断
  - 软中断
- 操作系统如何管理设备
  - 驱动程序
    - 驱动模型
  - 设备树

操作系统的管理/服务

- 根据不同需求和场景，人们发明了大量专用设备
  - 通信、存储、智能计算、安全协处理器等
- 每种设备有自己的协议、规范
  - 如何标准化外设接口？
- 设备也可能产生错误
  - 如何得知外设的状态并修复错误？

树莓派上的外设与接口

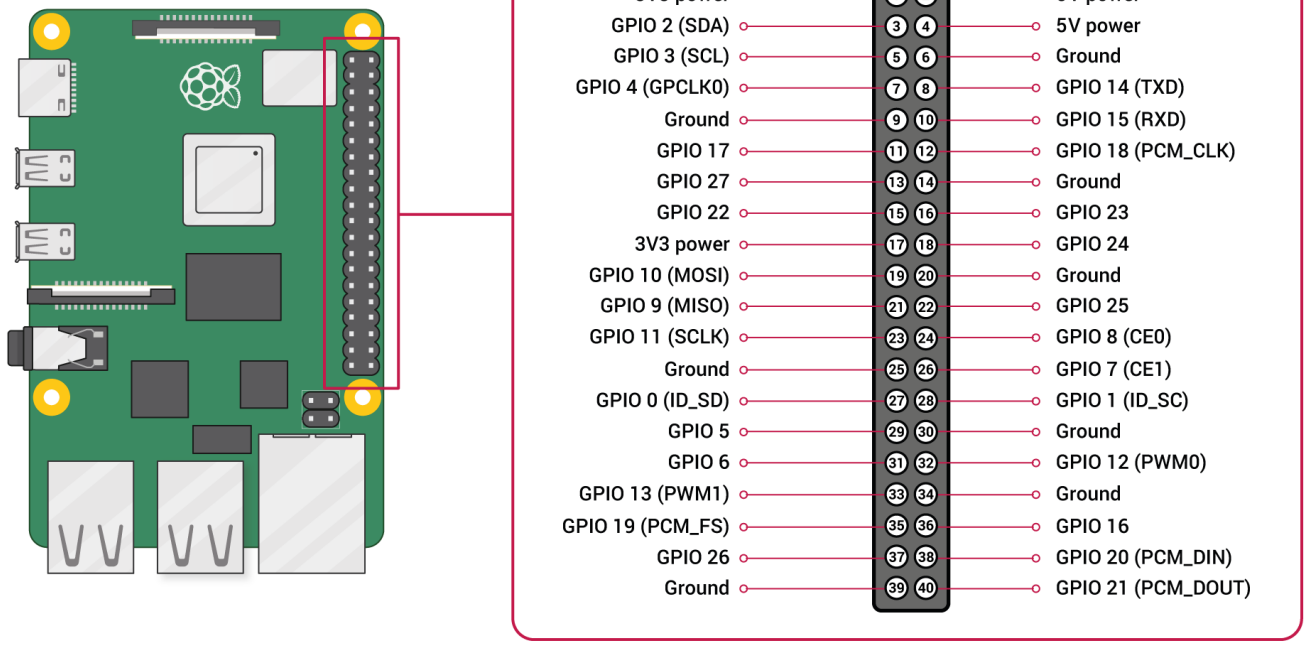


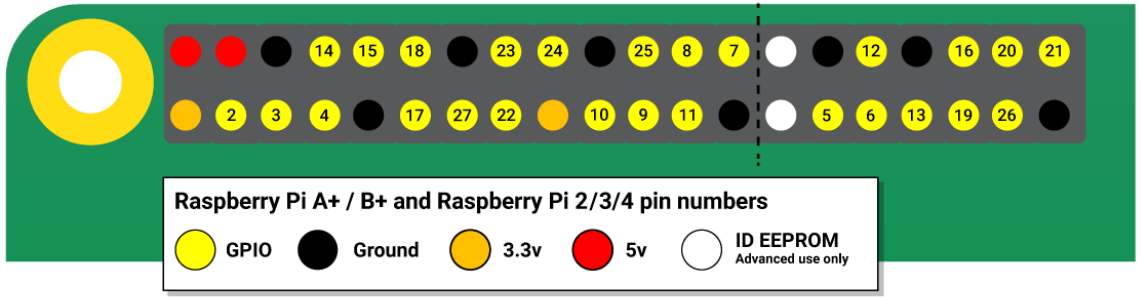
- 种类多
- 功能不同
- 接口、协议、规范不同
- 操作系统的担当：
  - 把复杂留给自己，把简单留给他人

认知外设

- 设备的基本用途（能提供什么功能）
- 是否具备某些共同点（能否分门别类做简化）
- 如何驱使它们工作（设备的可编程接口长什么样）

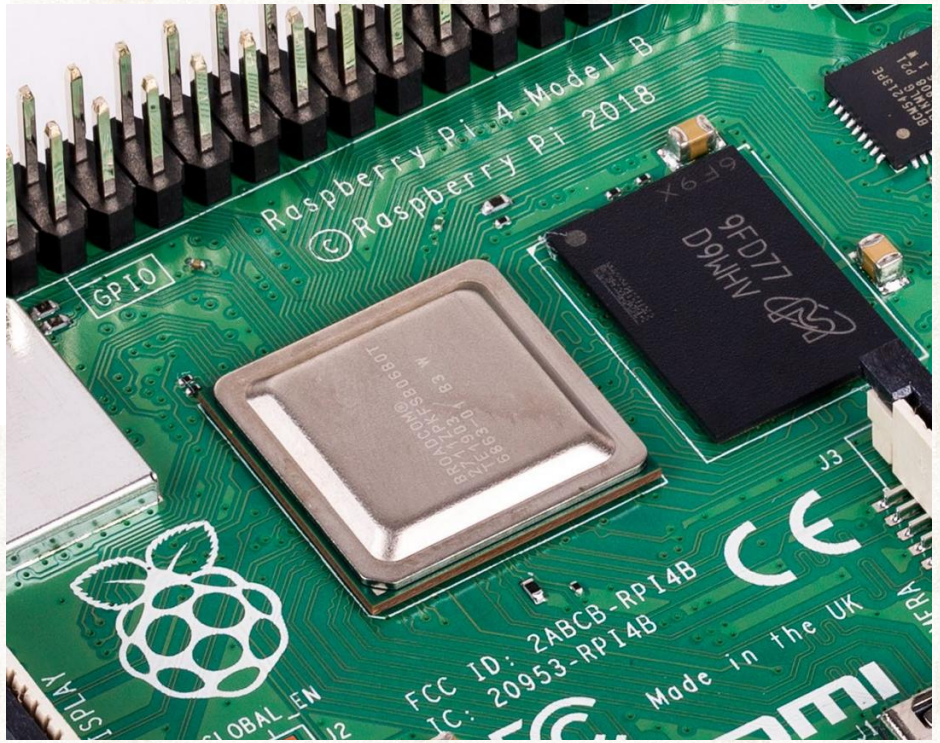
General Purpose I/O, GPIO





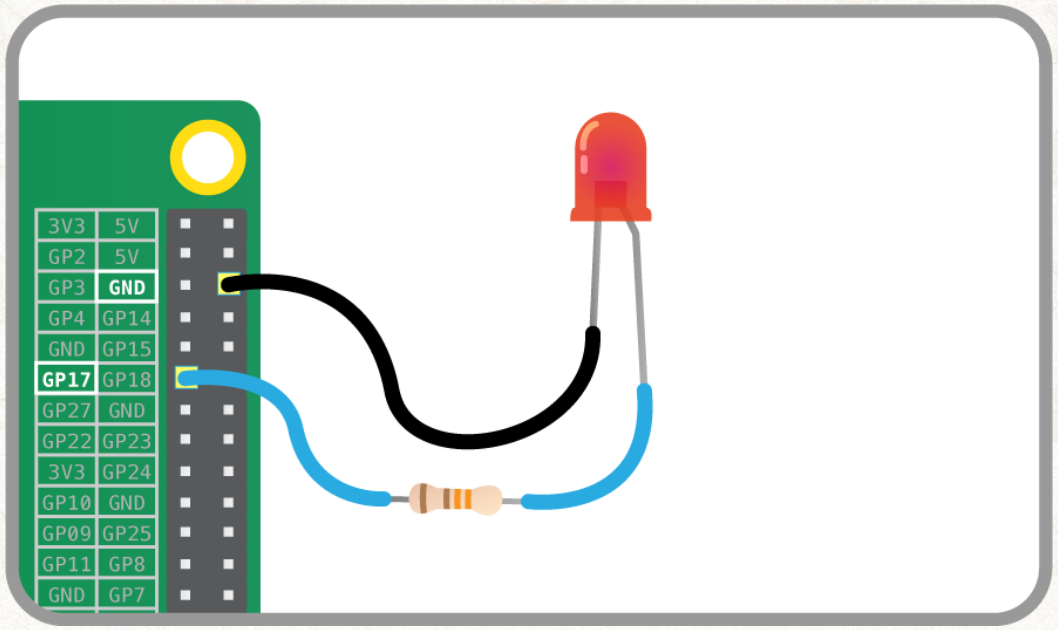
https://www.raspberrypi.com/documentation/computers/os.html#gpio-and-the-40-pin-header

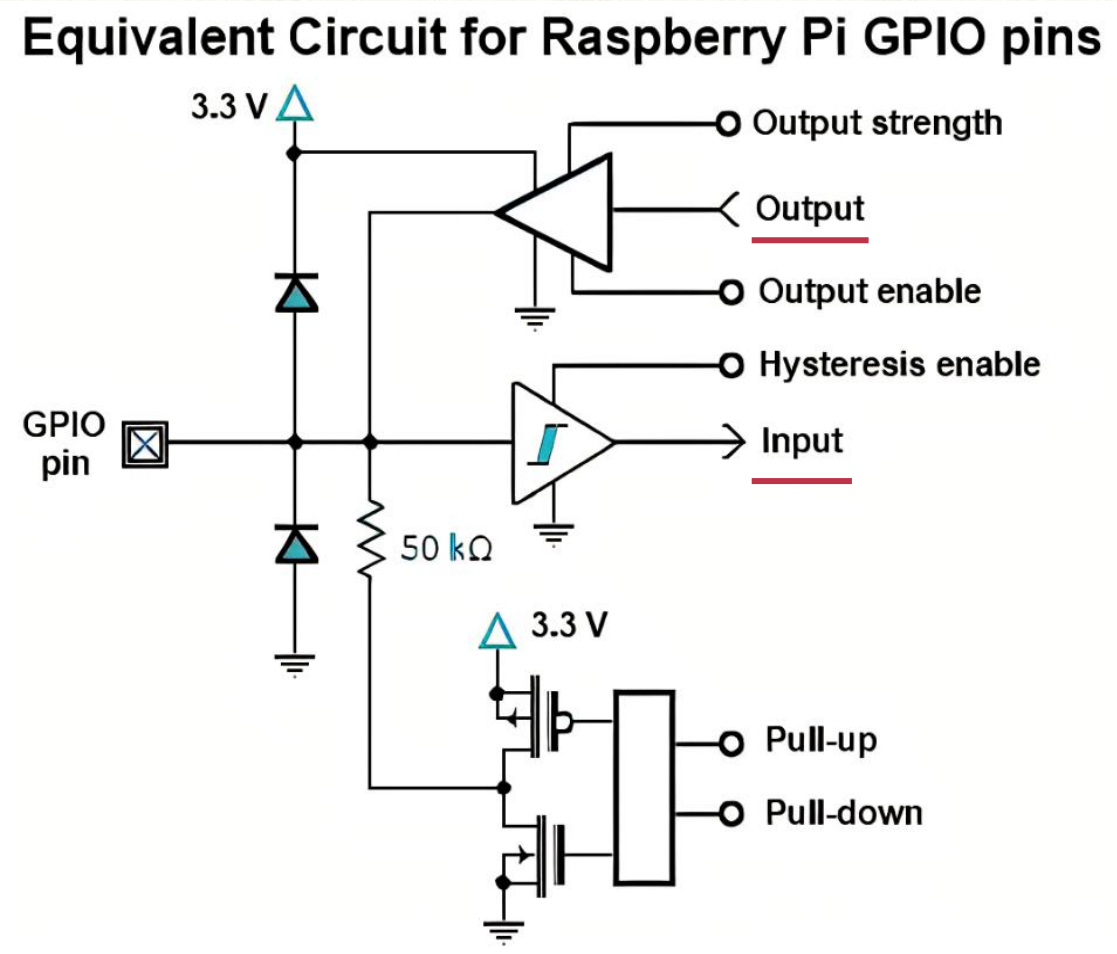
- 硬件接口、电线、管脚数目是有限的
- 而外接设备可以非常多



GPIO LED

- 有专门的 INPUT/OUTPUT 管脚
- 通过管脚进行控制
- 每个01组合只显示一种状态



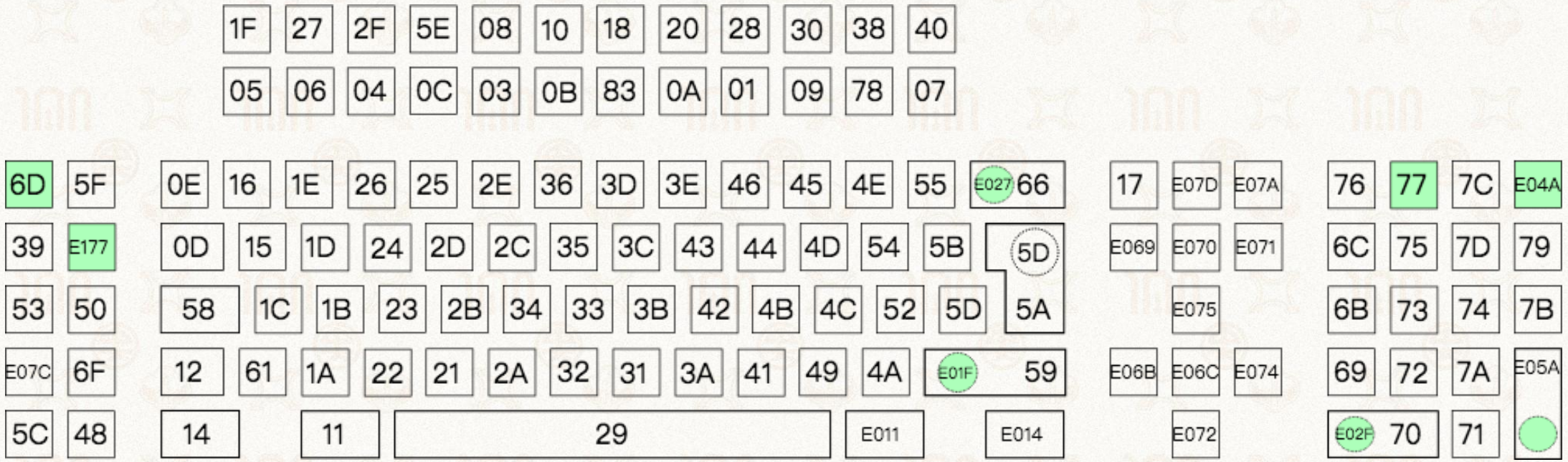


Intel 8042（PS/2 键盘控制器）

- 电信号 → 数字信号 → 编码(Scan Code)

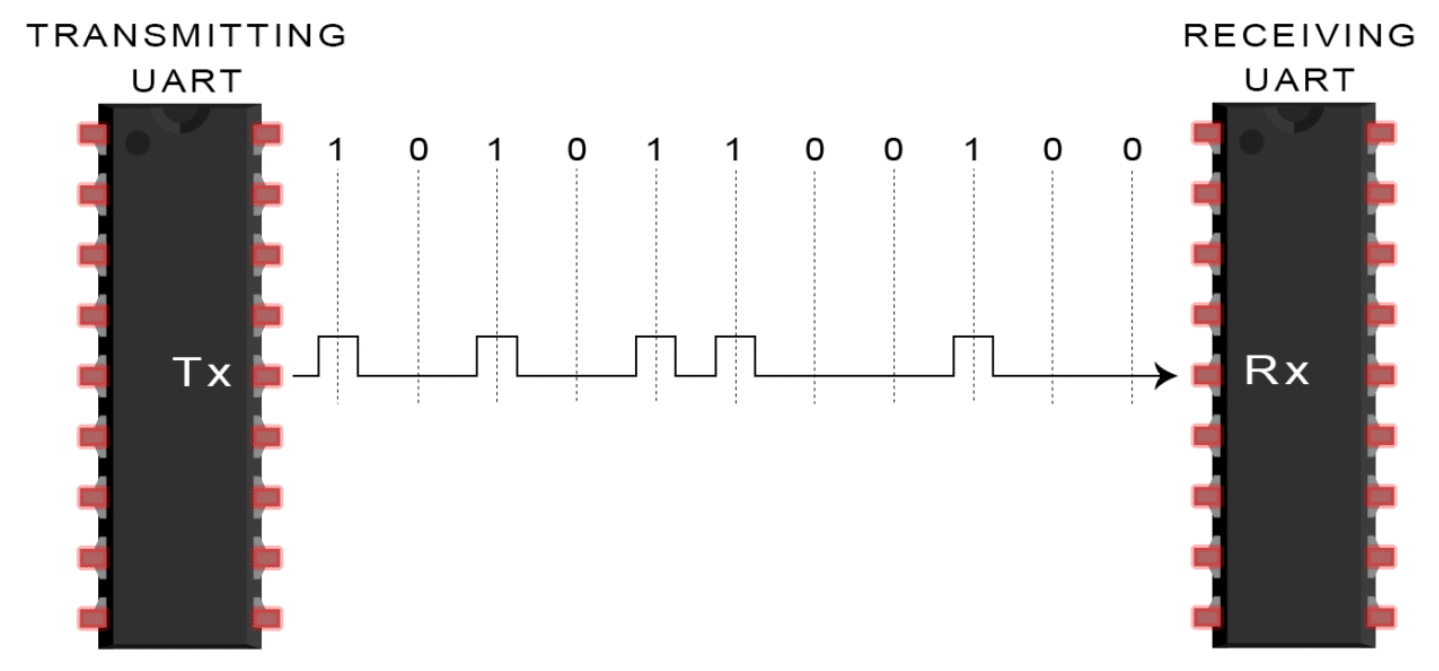
- 每次只能键入一个字符

如果长按会如何?



UART（串口）

- 通用异步收发传输器
  - Universal Asynchronous Receiver/Transmitter
- 半双工
- 每次只能传输一个字符



https://www.circuitbasics.com/basics-uart-communication/

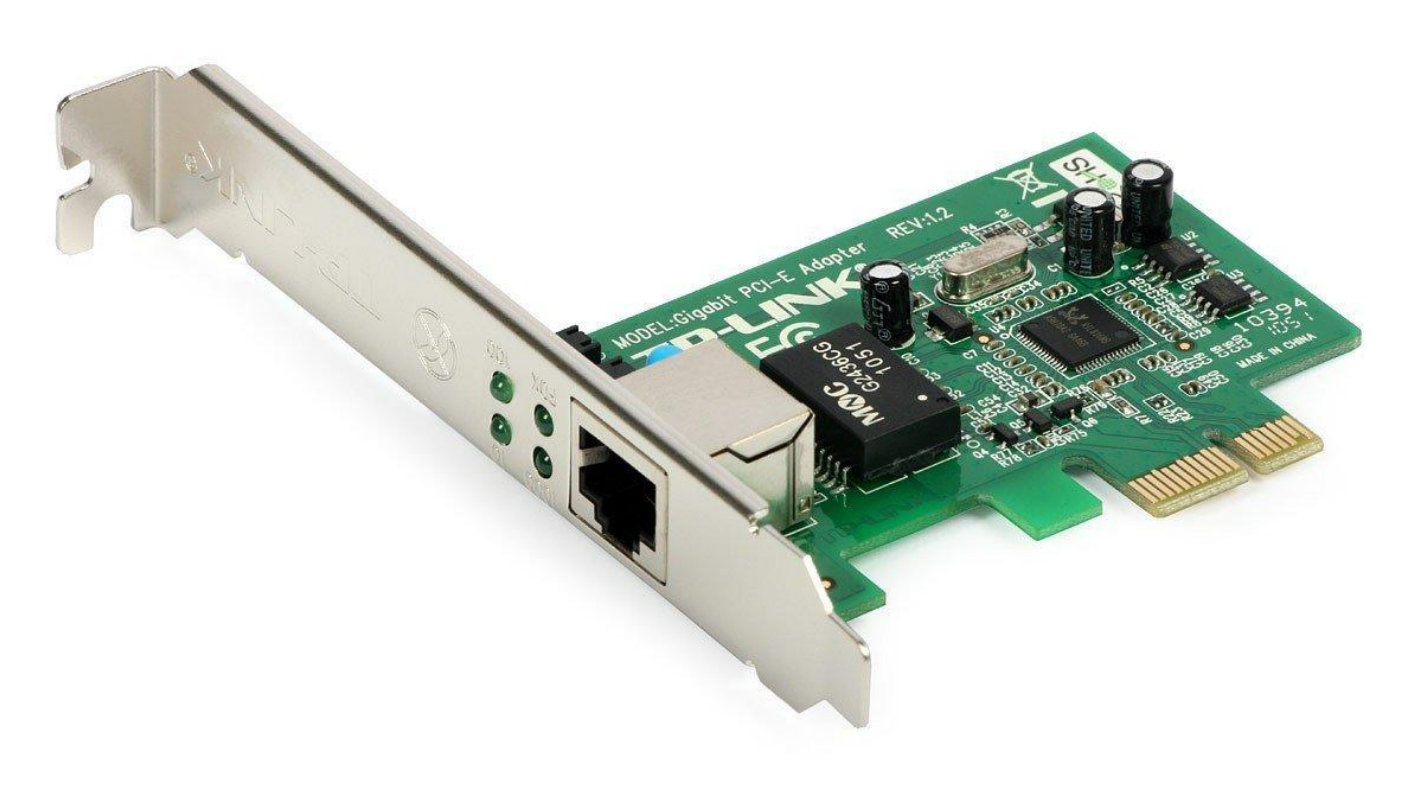
Flash闪存

- 按照页/块的粒度进行读写/擦除
- 支持页/块随机访问

Ethernet网卡

- 每次传输一块数据（以太网帧）
  - 每一块有严格的结构

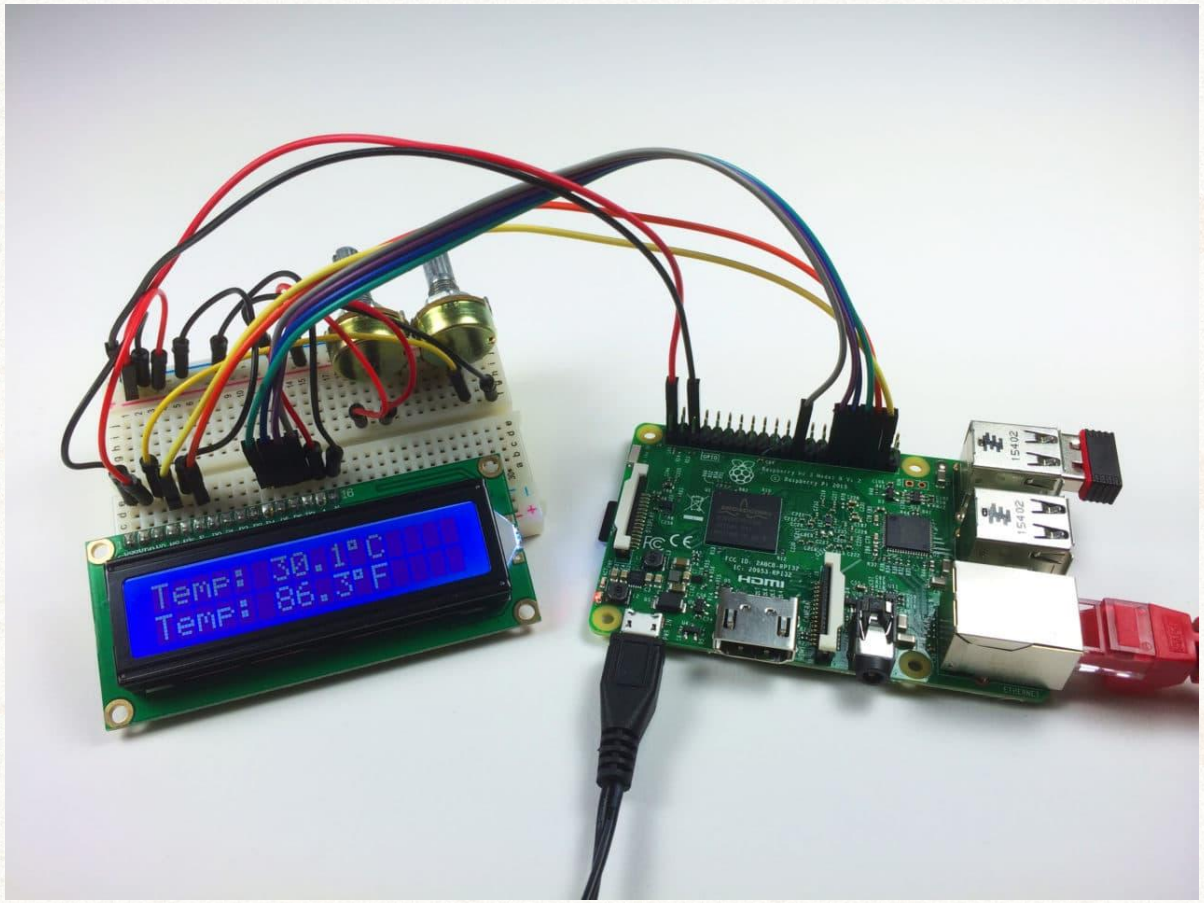
- Wifi、蓝牙与此类似



设备还有很多……

- 显示器
- 传感器
- 陀螺仪
- 磁力计
- 协处理器(GPU、TPU)
- …

- 如何统一管理?
  - 抽象



大纲

- 设备连接
- 设备类型抽象
  - 字符设备
  - 块设备
  - 网络设备
- 设备与操作系统的交互
  - 可编程I/O
  - 直接内存访问
- 操作系统如何响应设备：中断
  - 中断优先级
  - 硬中断
  - 软中断
- 操作系统如何管理设备
  - 驱动程序
    - 驱动模型
  - 设备树

设备管理：复用文件系统抽象

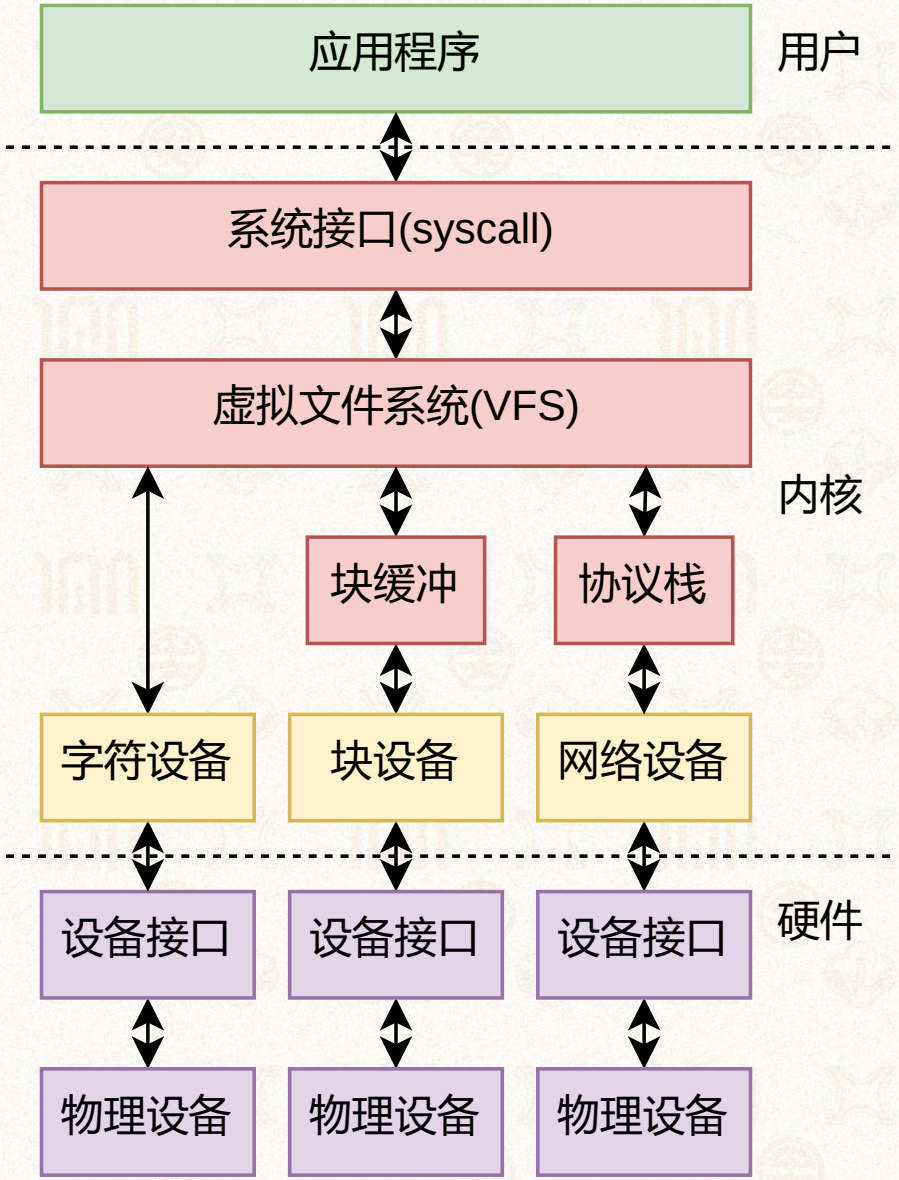
- 抽象的作用
  - 操作系统将外设细节和协议封装在接口的内部

- 如下示例代码可以运行在不同设备上
  - 为应用程序提供的相同的抽象接口（文件接口）

```
char buffer[256];
int read_num = -1;
int fd = open("/dev/something", O_RDWR);
write(fd, "something to device", 19);

while (read_num == -1) {
    read_num = read(fd, buffer, 256);
}
close(fd);
```

设备抽象（以Linux为例）

- 对设备进行分类
  - 字符设备（char）： LED、键盘、串口等
  - 块设备（block）： 闪存、硬盘等
  - 网络设备（network）： Ethernet网卡、蓝牙网卡等
- 对设备进行管理
  - 字符抽象：文件系统（read/write）
  - 块抽象：文件系统（read/write），mmap
  - 网络抽象：socket，文件系统兼容（用read/write也可读写socket）



字符设备(Char Device)

- 例子：
  - 键盘、鼠标、串口、打印机等
- 访问模式：
  - 顺序访问，每次读取一个字符
  - 调用驱动程序和设备直接交互
- 通常使用文件抽象：
  - open()
  - read()
  - write()
  - close()

```
yxsu@Dell-T6401:~$ cat /proc/devices
Character devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  5 ttyprintk
  6 lp
  7 vcs
 10 misc
 13 input
 21 sg
 29 fb
 89 i2c
 99 ppdev
108 ppp
128 ptm
136 pts
180 usb
189 usb_device
```

字符设备(Char Device)

- 命令行终端示例:
- 通过向字符设备写字符，实现操纵另一个设备

终端1(名称为pts/9)

```
yxsu@Dell-T6401:/dev$ ps
    PID TTY          TIME CMD
3140290 pts/9    00:00:00 bash
3141628 pts/9    00:00:00 ps
yxsu@Dell-T6401:/dev$ echo "hello world" > /dev/pts/15
```

终端2(名称为pts/15)

```
yxsu@Dell-T6401:~$ ps
    PID TTY          TIME CMD
3141309 pts/15   00:00:00 bash
3141506 pts/15   00:00:00 ps
yxsu@Dell-T6401:~$ hello world
```

块设备

- 例子：
  - 磁盘、U盘、闪存等（以存储设备为主）
- 访问模式：
  - 随机访问，以块粒度进行读写
  - 在驱动程序之上增加一层缓冲，避免和慢设备频繁交互
- 通常使用内存抽象：
  - 内存映射文件(Memory-Mapped File)：直接访问数据
  - 同样可以使用文件抽象，但内存抽象更受欢迎（灵活性更好）

```
yxsu@Dell-T6401:~$ cat /proc/devices
…
Block devices:
  7 loop
  8 sd
  9 md
 11 sr
 65 sd
 66 sd
 67 sd
 68 sd
 69 sd
 70 sd
 71 sd
128 sd
129 sd
130 sd
131 sd
132 sd
133 sd
```

SCSI disk 表示磁盘

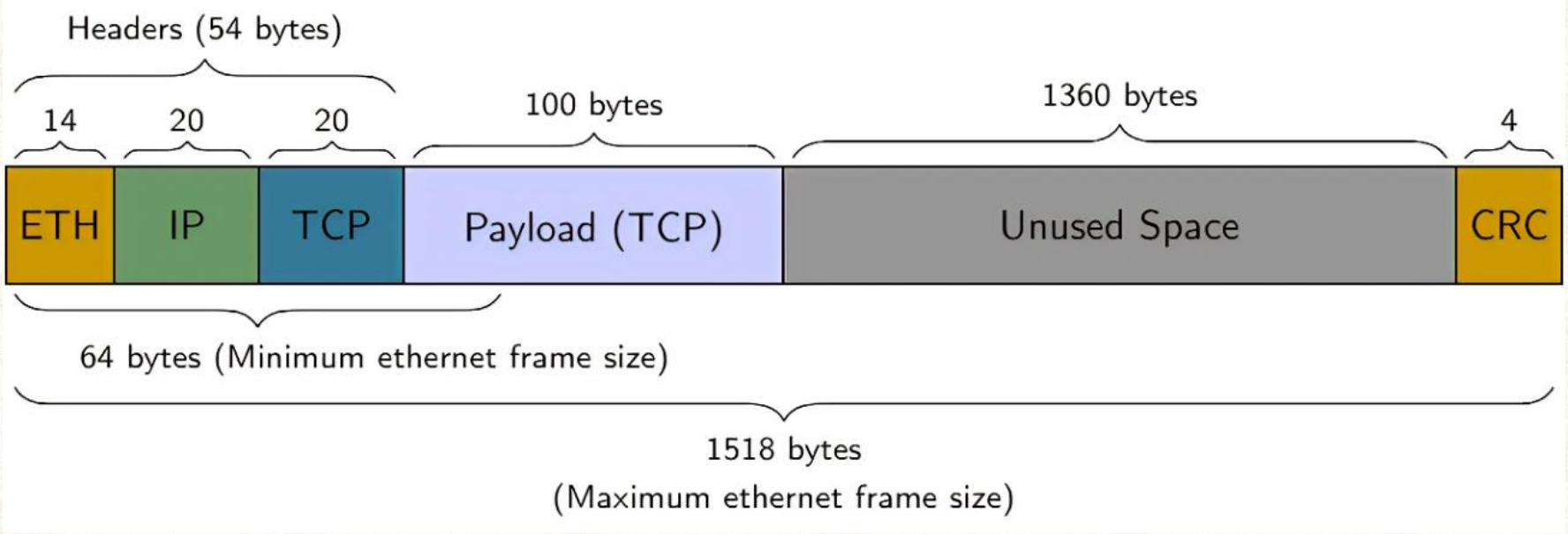
块设备

```
yxsu@Dell-T6401:/dev$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0    7:0    0     4K  1 loop /snap/bare/5
loop1    7:1    0  61.9M  1 loop /snap/core20/1405
loop2    7:2    0  55.5M  1 loop /snap/core18/2284
loop3    7:3    0  43.6M  1 loop /snap/snapd/15177
loop7    7:7    0  55.5M  1 loop /snap/core18/2344
loop8    7:8    0   9.6M  1 loop /snap/kubeadm/2417
loop10   7:10   0  44.7M  1 loop /snap/snapd/15534
loop11   7:11   0  54.2M  1 loop /snap/snap-store/558
loop12   7:12   0  65.2M  1 loop /snap/gtk-common-
themes/1519
loop13   7:13   0    51M  1 loop /snap/snap-store/547
loop14   7:14   0   9.6M  1 loop /snap/kubeadm/2461
loop15   7:15   0  61.9M  1 loop /snap/core20/1434
sda      8:0    0  21.9T  0 disk /data
sdb      8:16   0 893.8G  0 disk
├─sdb1   8:17   0   512M  0 part /boot/efi
└─sdb2   8:18   0 893.3G  0 part /
sr0     11:0    1  1024M  0 rom
```

```
yxsu@Dell-T6401:/dev$ sudo fdisk /dev/sdb

欢迎使用 fdisk (util-linux 2.34)。
更改将停留在内存中，直到您决定将更改写入磁盘。
使用写入命令前请三思。


命令(输入 m 获取帮助)： p
Disk /dev/sdb：893.77 GiB，959656755200 字节，1874329600 个扇区
Disk model: PERC H750 Adp
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：262144 字节 / 262144 字节
磁盘标签类型：gpt
磁盘标识符：2330DA30-1ADD-4C37-98B6-87F666805186

设备         起点       末尾       扇区   大小 类型
/dev/sdb1     2048    1050623    1048576   512M EFI 系统
/dev/sdb2  1050624 1874327551 1873276928 893.3G Linux 文件系统
```

网络设备

- 例子：
  - 以太网、WiFi、蓝牙等（以通信设备为主）
- 访问模式：
  - 面向格式化报文的收发
  - 在驱动层之上维护多种协议，支持不同策略
- 通常使用套接字抽象：
  - socket()
  - send()
  - recv()
  - close()



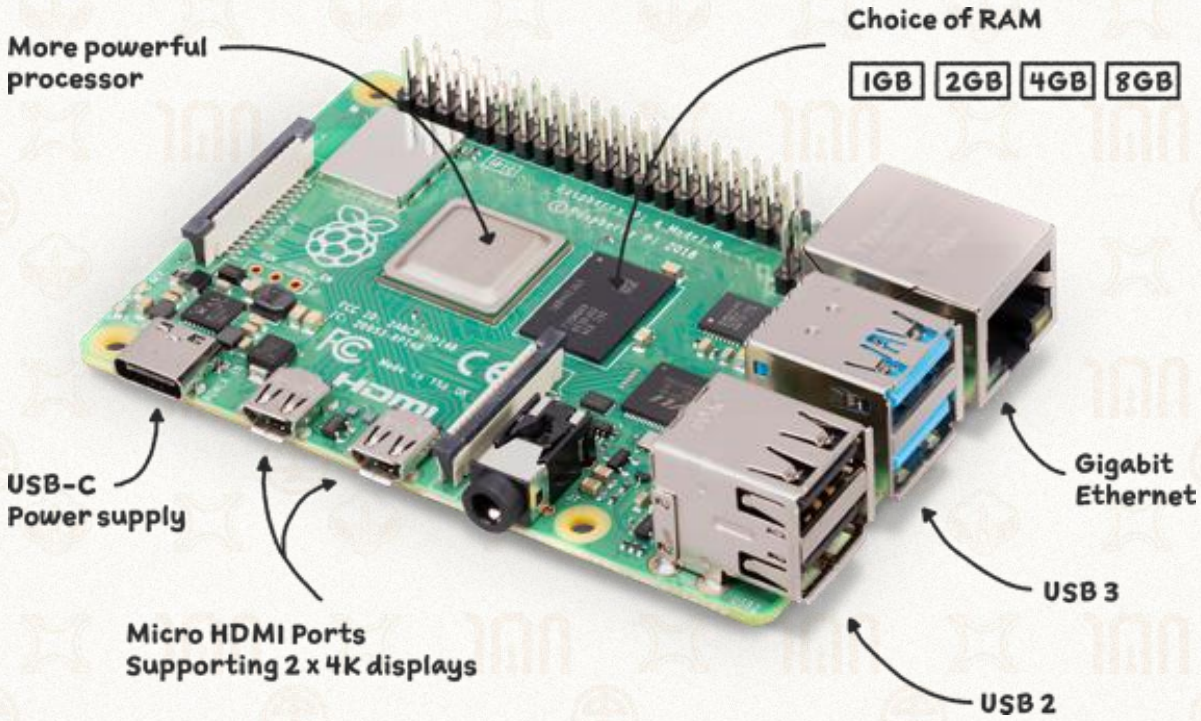
多选题 1分

设置

此题未设置答案，请点击右侧设置按钮

树莓派上的这些外设中，哪些属于字符设备？

A	UART串口
B	SD卡
C	RTC实时时钟
D	DS18B20温度传感器
E	USB
F	以太网
G	HDMI
H	GPIO
I	CSI摄像头
J	板载无线蓝牙



提交

大纲

- 设备连接
- 设备类型抽象
  - 字符设备
  - 块设备
  - 网络设备
- 设备与操作系统的交互
  - 可编程I/O
  - 直接内存访问
- 操作系统如何响应设备：中断
  - 中断优先级
  - 硬中断
  - 软中断
- 操作系统如何管理设备
  - 驱动程序
    - 驱动模型
  - 设备树

CPU与外设的数据交互

- 可编程 I/O（Programmable I/O）
  - 通过CPU in/out 或 load/store 指令
  - 消耗CPU时钟周期和数据量成正比
  - 适合于简单小型的设备

- 直接内存访问（Direct Memory Access, DMA）
  - 外设可直接访问总线
  - DMA与内存互相传输数据，传输不需要CPU参与
  - 适合于高吞吐量I/O

可编程 I/O

- PIO (Port IO)
  - IO设备具有独立的地址空间
  - 使用特殊的指令（如x86中的in/out指令）
- MMIO (Memory-mapped IO)
  - 将设备映射到连续物理内存中
  - 使用内存访问指令
  - 行为与内存不完全一样，读写会有副作用

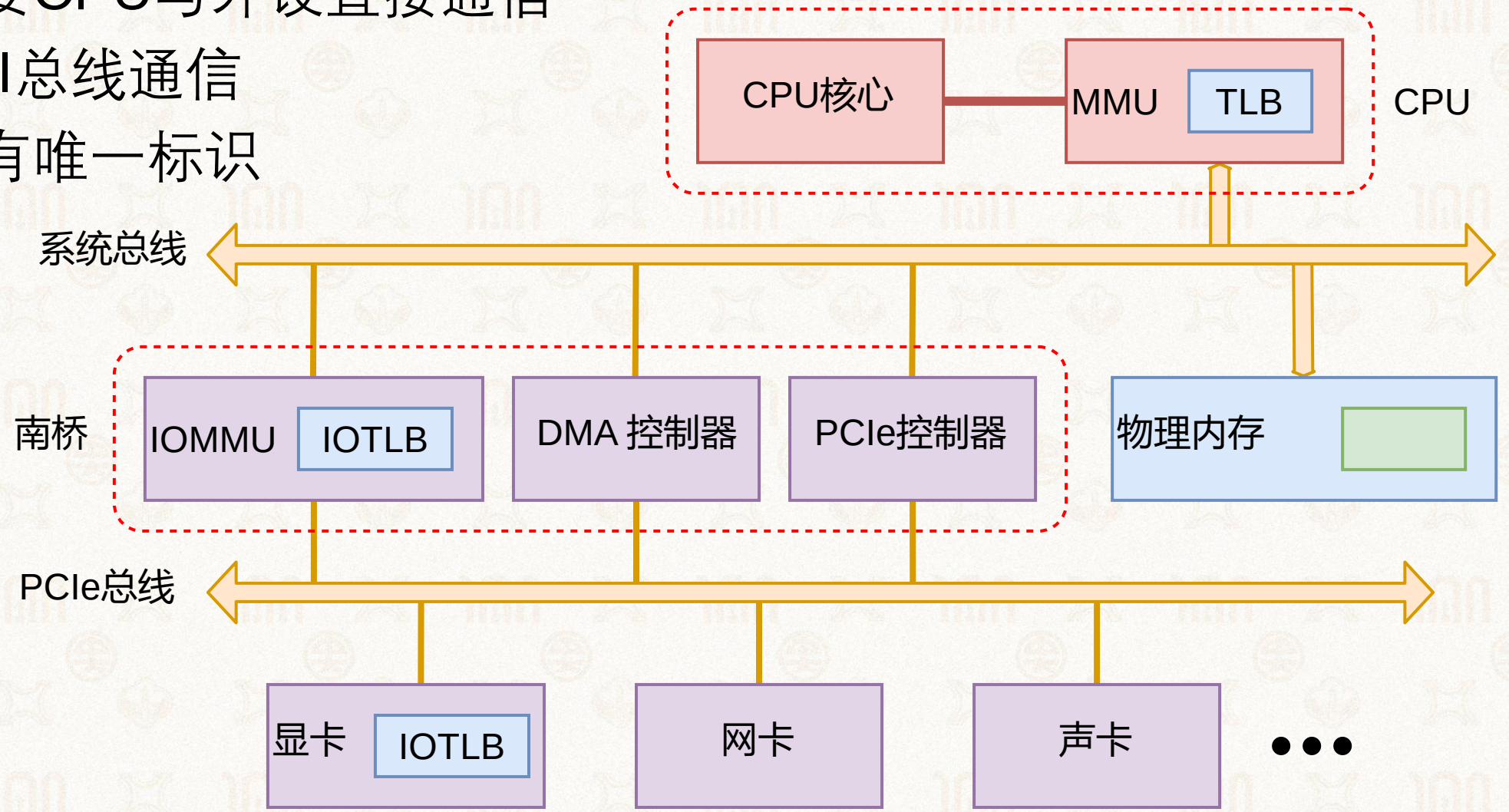
```
void early_uart_init(void) {
    unsigned int ra;

    ra = early_get32(GPFSEL1);
    ra &= ~(7 << 12);
    ra |= 2 << 12;
    ra &= ~(7 << 15);
    ra |= 2 << 15;
    early_put32(GPFSEL1, ra);

    early_put32(GPPUD, 0);
    delay(150);
    early_put32(GPPUDCLK0, (1 << 14) | (1 << 15));
    delay(150);
    early_put32(GPPUDCLK0, 0);
}
```

```
BEGIN_FUNC(early_put32)
    str w1, [x0]
    ret
END_FUNC(early_put32)
```

```
BEGIN_FUNC(early_get32)
    ldr w0, [x0]
    ret
END_FUNC(early_get32)
```

CPU与外设的数据交互

- 普通交互需要CPU与外设直接通信
- 外设通过PCI总线通信
- 每个设备都有唯一标识
  - 总线号
  - 设备号
  - 功能号
- 查看: lspci



https://www.intel.cn/content/www/cn/zh/products/sku/226254/intel-core-i71260p-processor-18m-cache-up-to-4-70-ghz/specifications.html

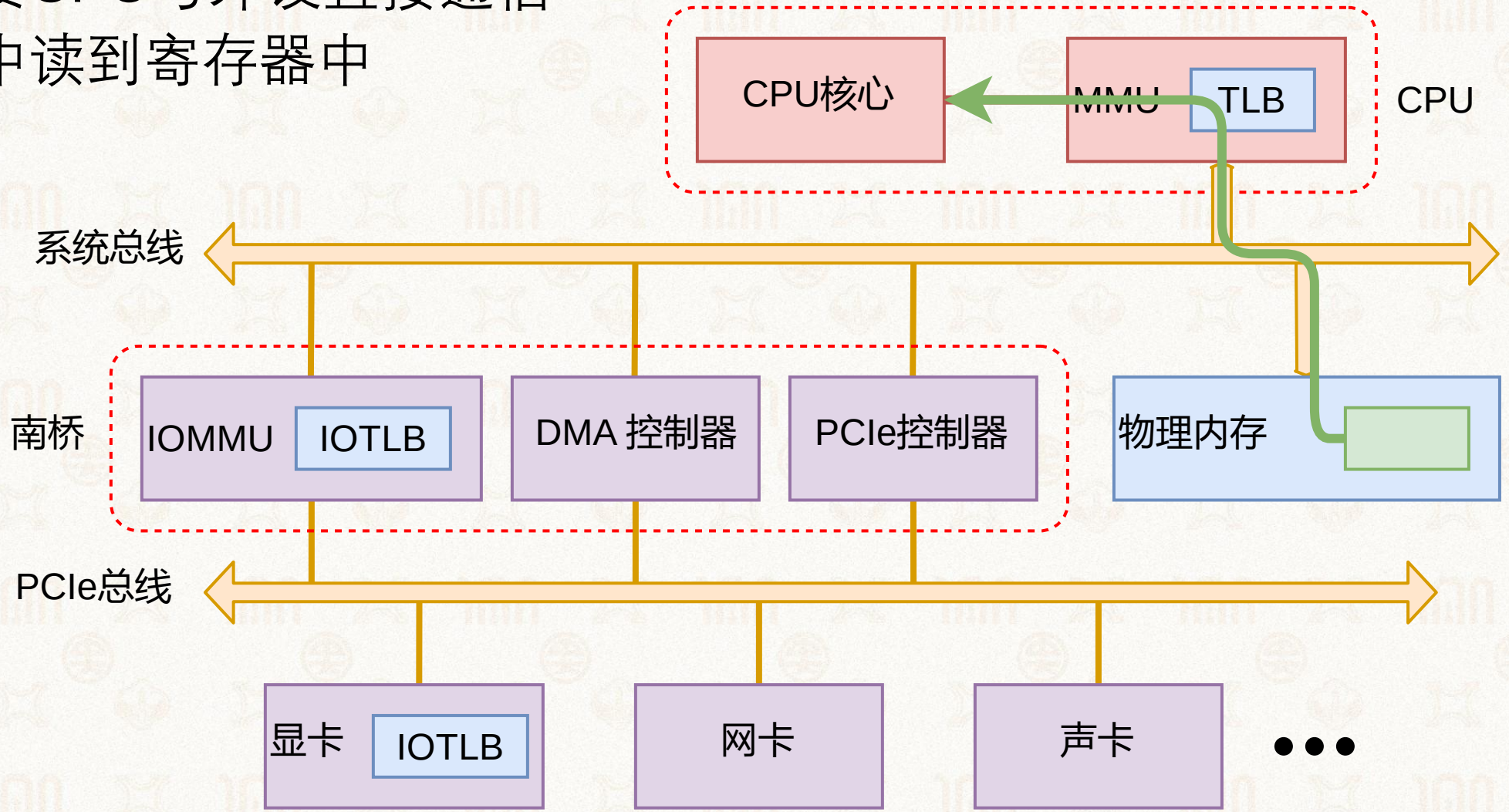
CPU与外设的数据交互

- 查看PCI设备：

```
yxsu@Dell-T6402:~$ lspci
00:00.0 Host bridge: Intel Corporation Sky Lake-E DMI3 Registers (rev 07)
00:05.0 System peripheral: Intel Corporation Sky Lake-E MM/Vt-d Configuration Registers (rev 07)
00:05.2 System peripheral: Intel Corporation Sky Lake-E RAS (rev 07)
00:05.4 PIC: Intel Corporation Sky Lake-E IOAPIC (rev 07)
00:08.0 System peripheral: Intel Corporation Sky Lake-E Ubox Registers (rev 07)
00:08.1 Performance counters: Intel Corporation Sky Lake-E Ubox Registers (rev 07)
00:08.2 System peripheral: Intel Corporation Sky Lake-E Ubox Registers (rev 07)
00:11.0 Unassigned class [ff00]: Intel Corporation C620 Series Chipset Family MROM 0 (rev 09)
00:11.5 SATA controller: Intel Corporation C620 Series Chipset Family SSATA Controller [AHCI mode] (rev 09)
00:14.0 USB controller: Intel Corporation C620 Series Chipset Family USB 3.0 xHCI Controller (rev 09)
00:14.2 Signal processing controller: Intel Corporation C620 Series Chipset Family Thermal Subsystem (rev 09)
00:17.0 SATA controller: Intel Corporation C620 Series Chipset Family SATA Controller [AHCI mode] (rev 09)
00:1c.0 PCI bridge: Intel Corporation C620 Series Chipset Family PCI Express Root Port #1 (rev f9)
00:1f.0 ISA bridge: Intel Corporation C621 Series Chipset LPC/eSPI Controller (rev 09)
00:1f.2 Memory controller: Intel Corporation C620 Series Chipset Family Power Management Controller (rev 09)
00:1f.4 SMBus: Intel Corporation C620 Series Chipset Family SMBus (rev 09)
00:1f.5 Serial bus controller [0c80]: Intel Corporation C620 Series Chipset Family SPI Controller (rev 09)
02:00.0 PCI bridge: PLDA PCI Express Bridge (rev 02)
03:00.0 VGA compatible controller: Matrox Electronics Systems Ltd. Integrated Matrox G200eW3 Graphics Controller (rev 04)
```

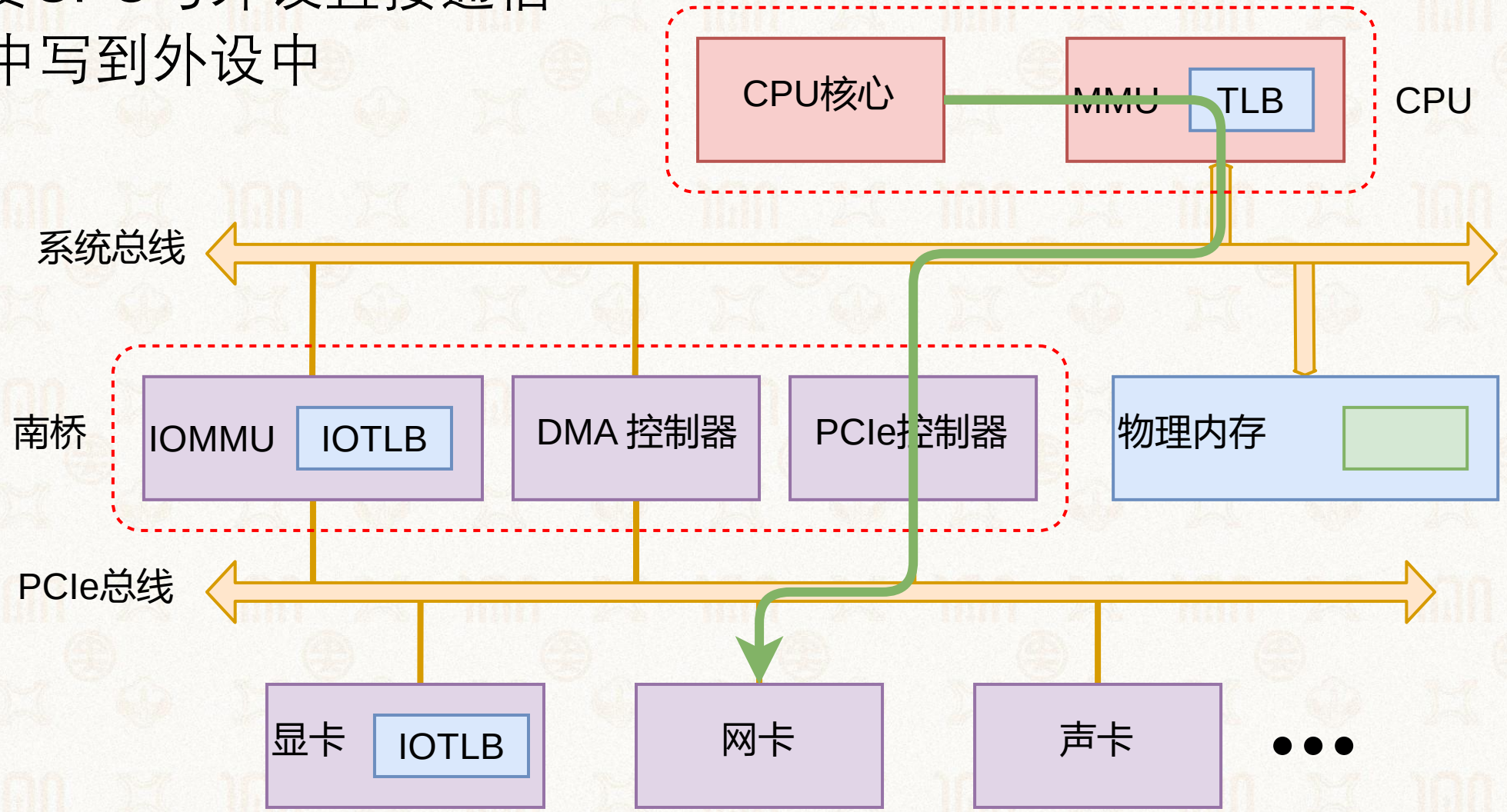
CPU与外设的数据交互

- 普通交互需要CPU与外设直接通信
- 数据从内存中读到寄存器中



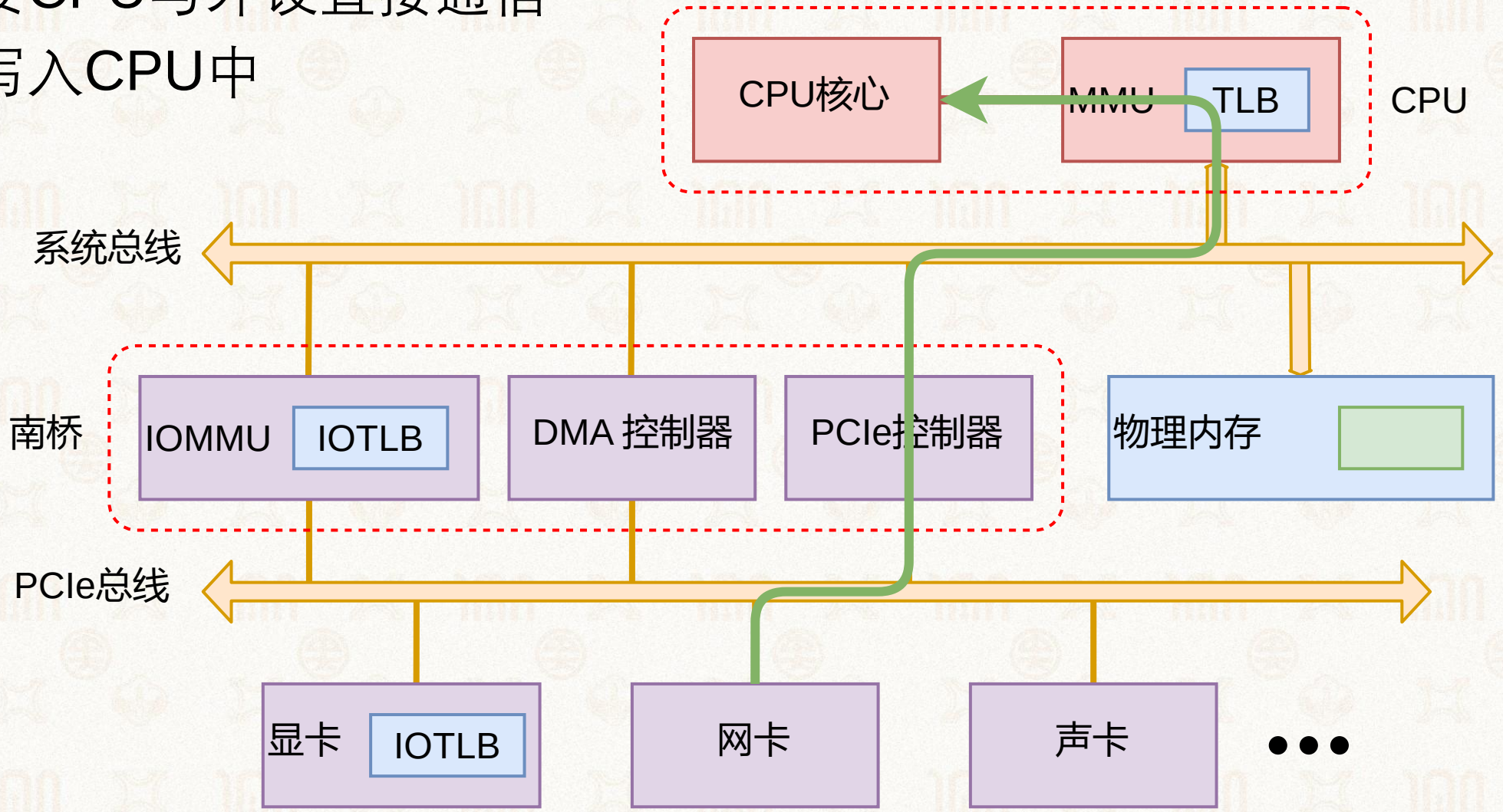
CPU与外设的数据交互

- 普通交互需要CPU与外设直接通信
- 数据从CPU中写到外设中



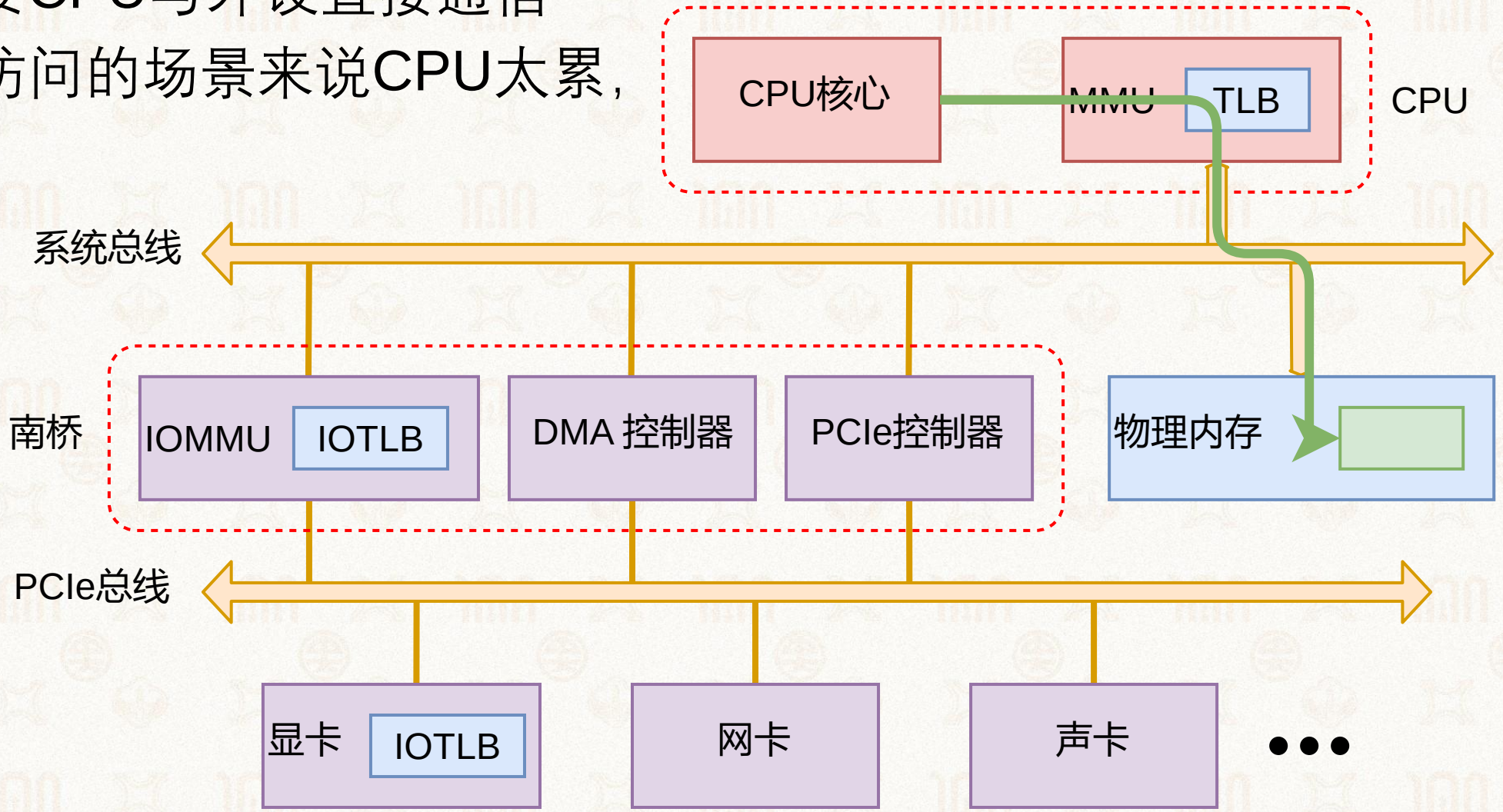
CPU与外设的数据交互

- 普通交互需要CPU与外设直接通信
- 数据从外设写入CPU中



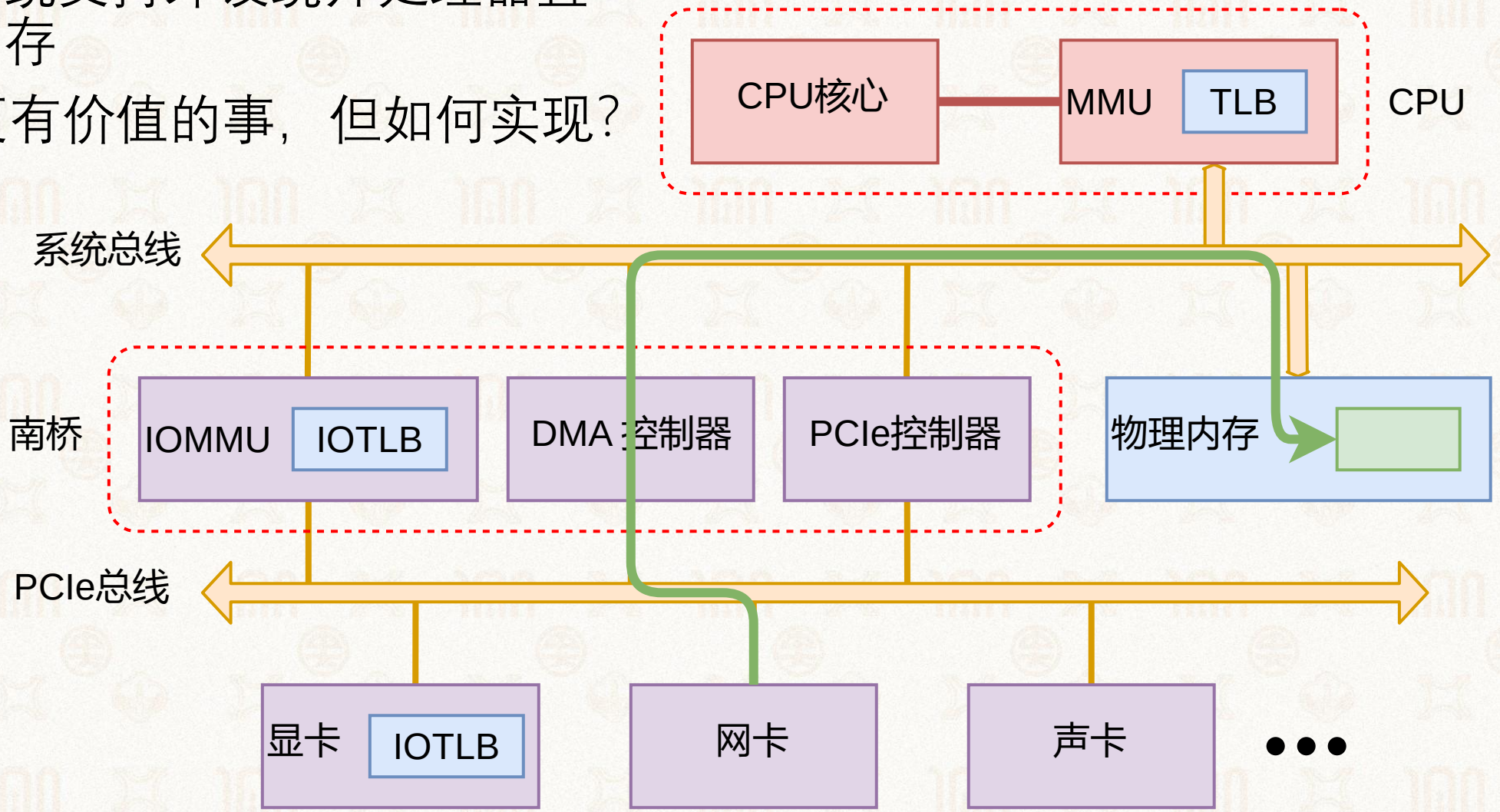
CPU与外设的数据交互

- 普通交互需要CPU与外设直接通信
- 对需要频繁访问的场景来说CPU太累，且无意义



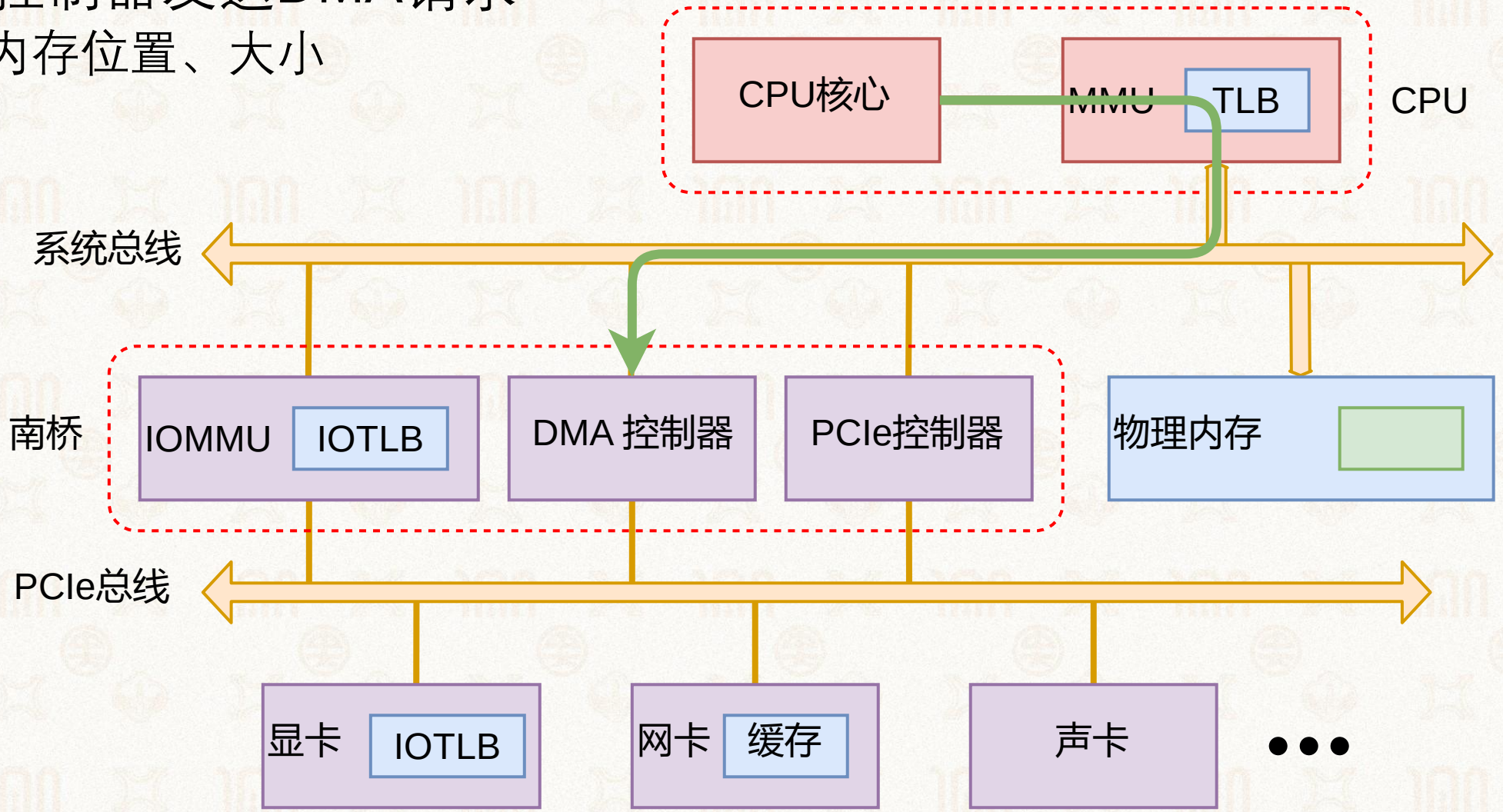
直接内存访问 DMA

- 现代计算机系统支持外设绕开处理器直接存取物理内存
- CPU可以做更有价值的事，但如何实现？



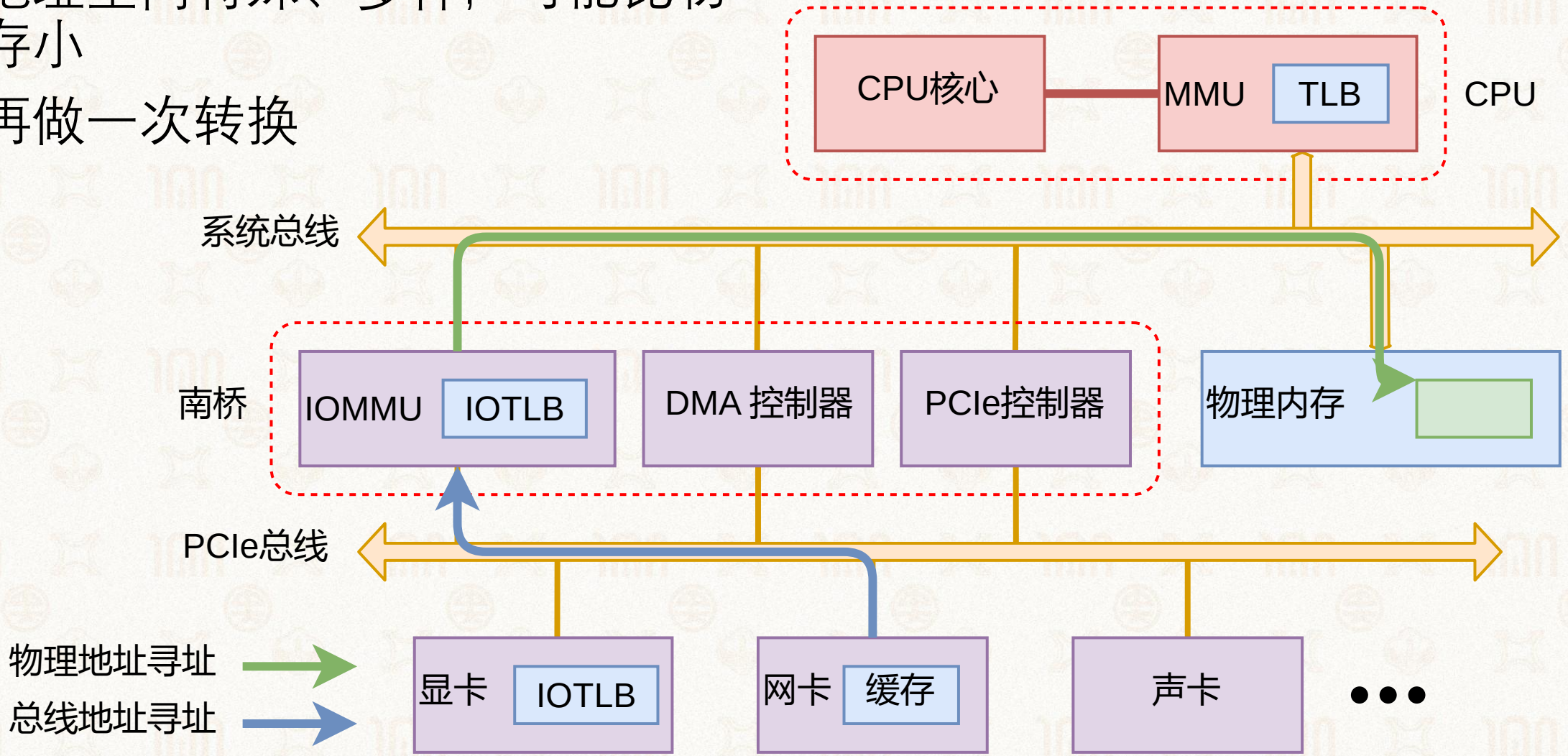
直接内存访问 DMA

- CPU向DMA控制器发送DMA请求
  - 方向、物理内存位置、大小



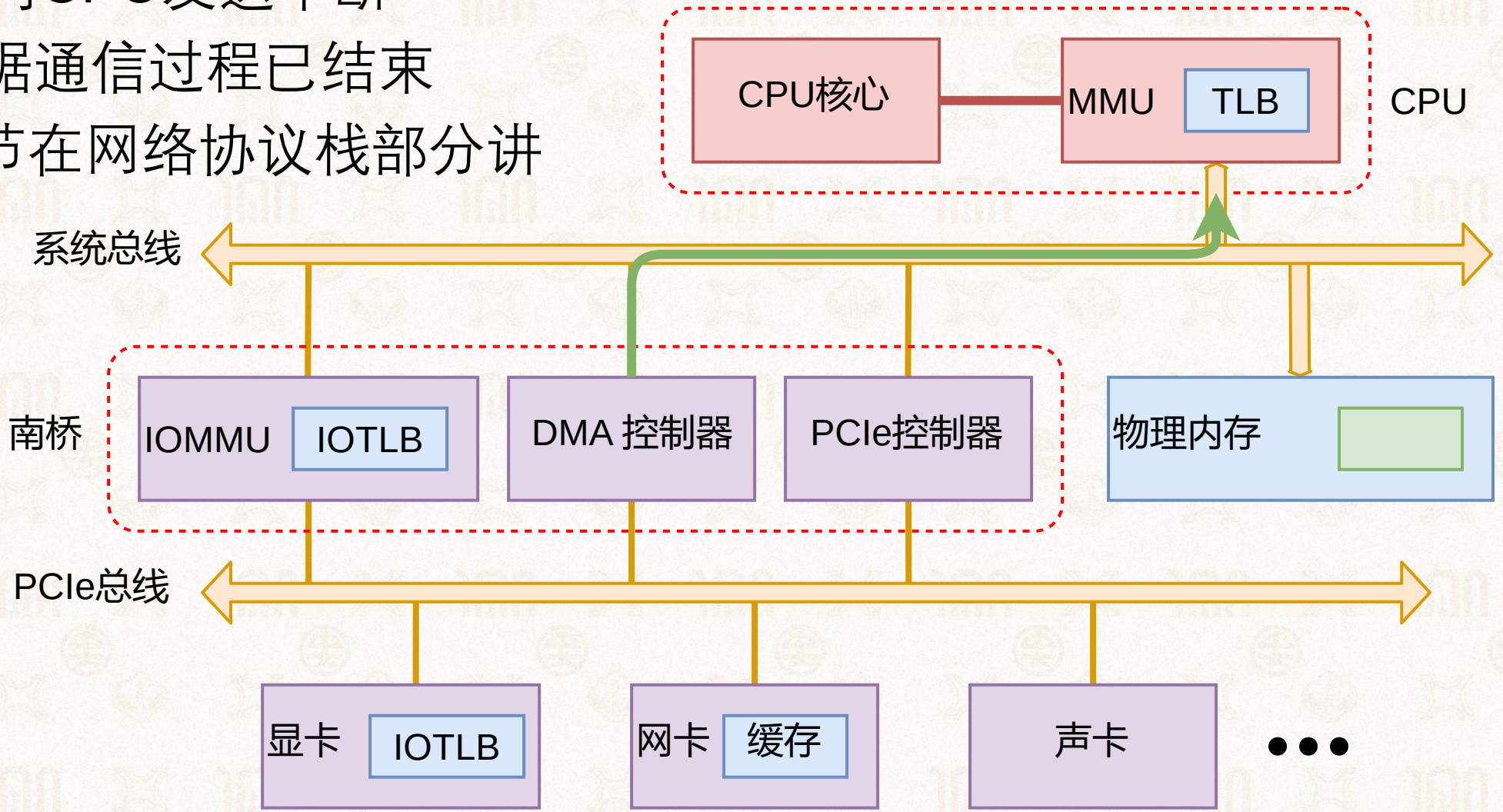
直接内存访问 DMA

- 外设地址空间特殊、多样，可能比物理内存小
- 需要再做一次转换



直接内存访问 DMA

- DMA控制器向CPU发送中断
- 通知CPU数据通信过程已结束
- 编程模型细节在网络协议栈部分讲



单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

进行数据传输时，DMA的特点在于

A DMA允许设备绕过处理器直接读写系统内存的数据

B DMA 通过读写内存指令就可以直接控制设备

C DMA可以高效的处理不需要I/O操作的数据传输

D DMA可以高效的处理网络传输的数据

提交

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

如果系统使用DMA模式将数据从硬盘读到内存，则该过程包括以下操作：

① DMA控制器发出中断请求

② 初始化DMA控制器并启动硬盘

③ 从硬盘传输一块数据到内存缓冲区

④ 执行“DMA结束”中断服务程序

那么正确的执行顺序是：

③ → ① → ② → ④

② → ③ → ① → ④

② → ① → ③ → ④

① → ② → ④ → ③

提交

谢谢

微信：suyuxin
钉钉：苏玉鑫
B站：https://space.bilibili.com/502854403
软工集市课程专区：https://ssemarket.cn/new/course
匿名提问箱：https://suask.me/ask-teacher/106/苏玉鑫