

微机原理与接口技术

第一章 基础知识

前言

- 自我介绍
- 教学内容：微型计算机原理与接口技术（周荷琴 第六版）1-8章
- 课程安排：48课时课堂教学，32课时上机练习
- 学习目标：使用汇编语言编程，嵌入式系统硬件设计入门
- 教学重点：加强练习，少量关联新技术发展
- 教学方法：预习，课堂讲解，课堂练习，课后练习，作业
- 第一二章后和第三四章后各有一次小测验

教学方法说明

- 第三章（指令）和第四章（编程）自学
 - 课堂编程练习，使用虚拟机
 - （注意：练习是点，自学是面）
 - 硬件学习需要采用**Modelsim**进行电路设计和调试
 - 设计代码请发邮箱sanghs@hust.edu.cn
-
- 课堂纪律：鼓励讨论，鼓励提问
 - 上课期间除了拍PPT，不能用手机做与课程无关的事，发现没收，下课归还
 - 课件问题

基本概念-1

- 数的表示：十进制、二进制、十六进制
- 注意数制的后缀，D（也可省略），B，H
- 十进制、二进制、十六进制之间的转换
- 字节、字、双字的概念

BCD码

- 用4bit二进制数表示数字0~9
- 例如：518 的 BCD 码 0101 0001 1000
- 计算机中BCD码分为：压缩BCD码、非压缩BCD码
- 例如：39的压缩BCD为 0011 1001
- 非压缩BCD码为00000011 00001001

ASCII码

- 用7bit表示字符和控制符，
- 例如：'A' 41H, '1' 31H,
- 回车0DH, 换行0AH, 删除18H, 等
- TXT文本文件中所包含的就是ASCII码
- 键盘送出的键值也是ASCII码

有符号数的表示方法

- 一般用最高位表示符号位
- 原码，反码，补码
- 正数的表示方法相同，负数各有不同

原码

- 符号位 + 绝对值

+1: 0 000 0001

-1: 1 000 0001

- 如何进行带符号加减运算？
- 根据符号位与运算符，分别执行加法或减法
- $+1 + (-1) = 1 - 1 = 0\ 000\ 0000 = 0$
- $+1 - (-1) = 1 + 1 = 0\ 000\ 0010 = 2$
- 能否只用一种运算电路来实现加减运算？


反码

- 符号位 + 绝对值/绝对值取反

+1: 0 000 0001

-1: 1 111 1110

- 如何进行带符号加减运算？
- 加法直接相加，减法转变为加法执行（带符号取反）
- +1 + (-1) = 1 111 1111 = -0 (0的表达不唯一)
- +1 - (-1) = 1 + (-(-1)) = 0 000 0010 = 2
- 特殊操作：符号位的进位应加入最低位

$$\begin{array}{r} 0\ 111\ 1111 \\ +\ 1\ 111\ 1111 \\ \hline 1\ 0\ 111\ 1110\ +1 \end{array}$$


补码

- 符号位 + 绝对值 / (绝对值取反+1)

+1: 0 000 0001

-1: 1 111 1111

- 正负数的转化: 带符号取反加1
- 通过正负数转换, 可以将减法转变为加法执行

- $+1 + (-1) = 0\ 000\ 0000 = 0$ (0的表达唯一)

0001 0001

+1111 0000

1 0000 0001

- $+1 - (-1) = 1 + (-(-1)) = 0\ 000\ 0010 = 2$

- 符号位和数据位一样加入运算, 例: $17-16=17+(-16)=1$

- 补码广泛用于微处理器中

进一步了解补码

- 运算规则： $[X+Y]_{\text{补}} = X_{\text{补}} + Y_{\text{补}}$ ， $[X-Y]_{\text{补}} = X_{\text{补}} + [-Y]_{\text{补}}$
- 优点1：通过加法器可以完成减法运算
- 优点2：用补码表示的有符号数加减和无符号数加减可通过相同的加法器实现

- 例如： $16 + (-128) = -112$

✓

- $16 + 128 = 144$

✓

$$\begin{array}{r} 0001 \ 0000 \\ +1000 \ 0000 \\ \hline 1001 \ 0000 \end{array}$$

- 8bit表示范围： $-128 \sim 127$ ， $1 \ 000 \ 0000\text{B} \sim 0 \ 111 \ 1111\text{B}$
- 16bit表示范围： $-32768 \sim 32767$ $8000\text{H} \sim 7\text{FFFH}$

基本概念-2

- 微处理器的功能
- 微型计算机的基本结构
- 存储器的功能
- 对存储器执行的操作
- 时序的概念

微处理器是怎么工作的？

- 简单来说：微处理器在**指令**的控制下进行**数据处理**工作
- 指令（**机器码**）和数据均以二进制方式表示，存放在**半导体存储器**中
- 举例：**高级语言**， `print (“Good morning”)` ；
- 通过**编译程序**将高级语言转化为微处理器可以执行的**机器码**
- 假设机器码和以**二进制ASCII码**方式表示的字符串 “Good morning” 已经存放在**内存（存储器）**里
- 微处理器基本功能之一：从存储器**顺序读取指令**
- 微处理器基本功能之二：**执行指令**
- 微处理器读取指令，根据指令依次读取字符，并依次写入连接显示器（**IO设备**）的**输出接口（IO接口）**，判断字符串是否全部显示
- 显示器控制电路在屏幕（**IO设备**）上显示**Good morning**

从哪里开始学习微处理器的知识？

- 从“微处理器的基本功能之一：从存储器读取指令（和数据）”开始了解：
- 什么是存储器，如何对存储器进行读写？
- 什么是总线，如何通过总线连接存储器？
- 微处理器硬件系统包括哪些部分？
- 微处理器的数据读写操作是如何在系统上进行的？

基本概念之： 存储器是什么

- 物理上：存放二进制数据的电路
- 逻辑上：存放一定位宽二进制数据的一维阵列
- 每个单元存放相同位宽的二进制数据
- 每个单元有索引号，即地址
- 地址范围：**000.....00b~111.....11b**

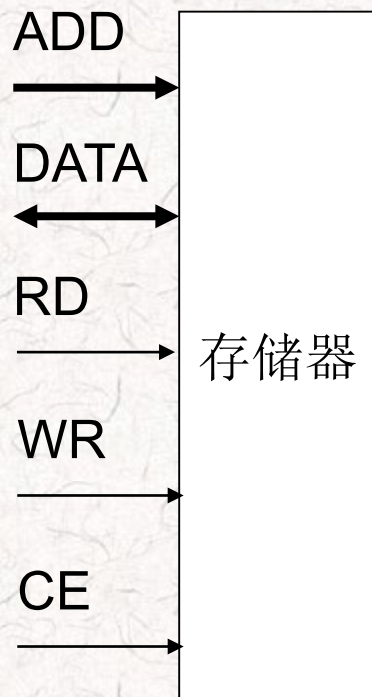
地址	内容
0000H	B8 H
0001H	02 H
0002H	00 H
0003H	3C H
~	~
1200H	8F H
1201H	A2 H
1202H	34 H
1203H	12 H
~	~
FFFFH	C9 H

基本概念之：如何对存储器



存储器实体

- 如何对存储器进行操作？
- 通过**引脚**实现对（由地址指定的）某个单元进行**读**或者**写**
- 由**地址端□**给出某个单元的地址（**n根信号**）
- 由**读/写信号**指明进行读或者写操作
- 由**数据端□**对该指定单元**读出**或者**写入**（**m根双向信号**）
- 还有**片选信号**指定该存储器是否可以被访问
- **读操作**：给出有效片选信号、单元的地址、有效读信号，则数据端□会出现该存储单元的内容（**存储器输出**）
- **写操作**：给出有效片选信号、单元的地址、有效写信号，需要写入的数据，则数据写入该单元（**存储器输入**）
- **总结：地址信号、数据信号、控制信号**
- 这也决定了**微处理器**的基本的**对外（CPU之外）访问模式**



基本概念之：总线1

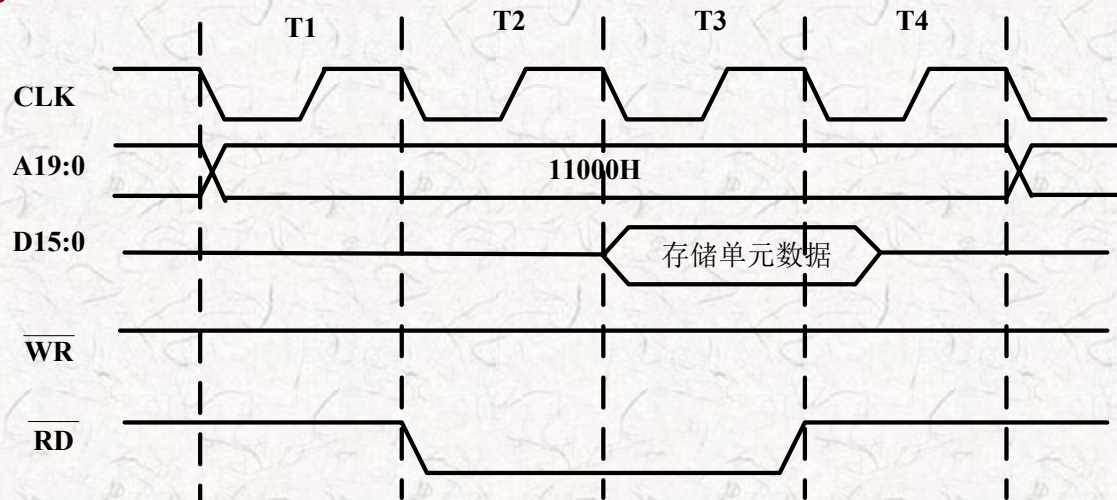
- 总线是用于微处理器与微处理器硬件系统中其它部分（存储器和IO端口）**交换信息**的一组信号，包括：
- **地址总线：单向传输**
 - 由微处理器发出，指明要**读（或写）**的存储器或IO端口的地址
 - 地址总线的根数决定**CPU**能寻址的范围，也称为地址空间
 - 举例 **8086：A19:0** 可寻址**1M**地址空间
 - 地址空间的单位是什么呢？需要结合其它访问存储器的信号分析，通常用字节表示
 - **8086** 地址空间为**1MB**

基本概念之：总线2

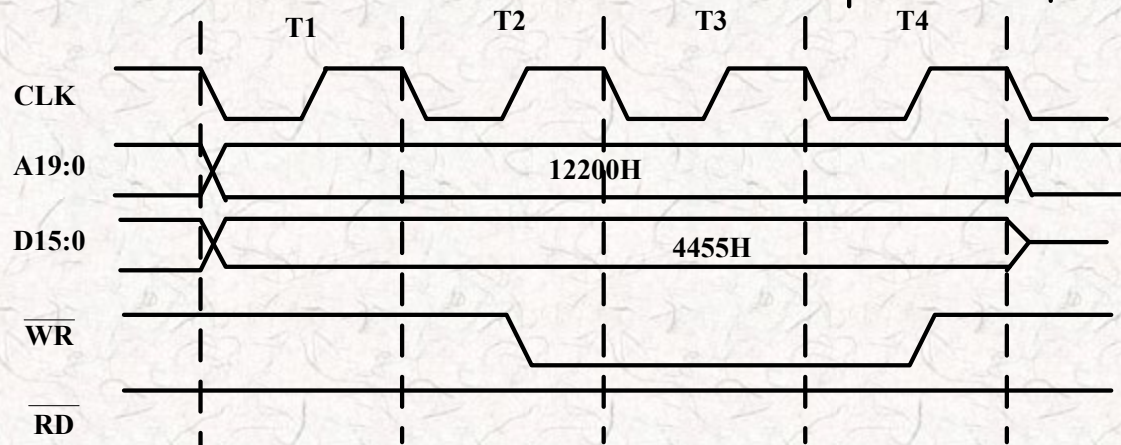
- 读、写、忙以及其它控制信号：单向传输
 - 读、写：由微处理器发出，指明对当前地址要进行读还是写操作
 - 忙：由存储器或IO端口发出，请求微处理器延长访问周期
- 核心--数据总线：分时双向传输
 - 微处理器 -> 存储器或IO端口（微处理器写操作）
 - 微处理器 <- 存储器或IO端口（微处理器读操作）
 - 数据总线根数决定CPU一次传输数据的最大位宽
 - 例如：8086 16bit，ARM 32bit，现代主流处理器 64bit
- 总结：微处理器系统中的主控方是微处理器
- 要实现正确的存储器访问，上述信号需按照一定的时序配合工作

基本概念之：时序

- 不同信号之间的
时间先后关系
- 时序是实现正确
操作的要求



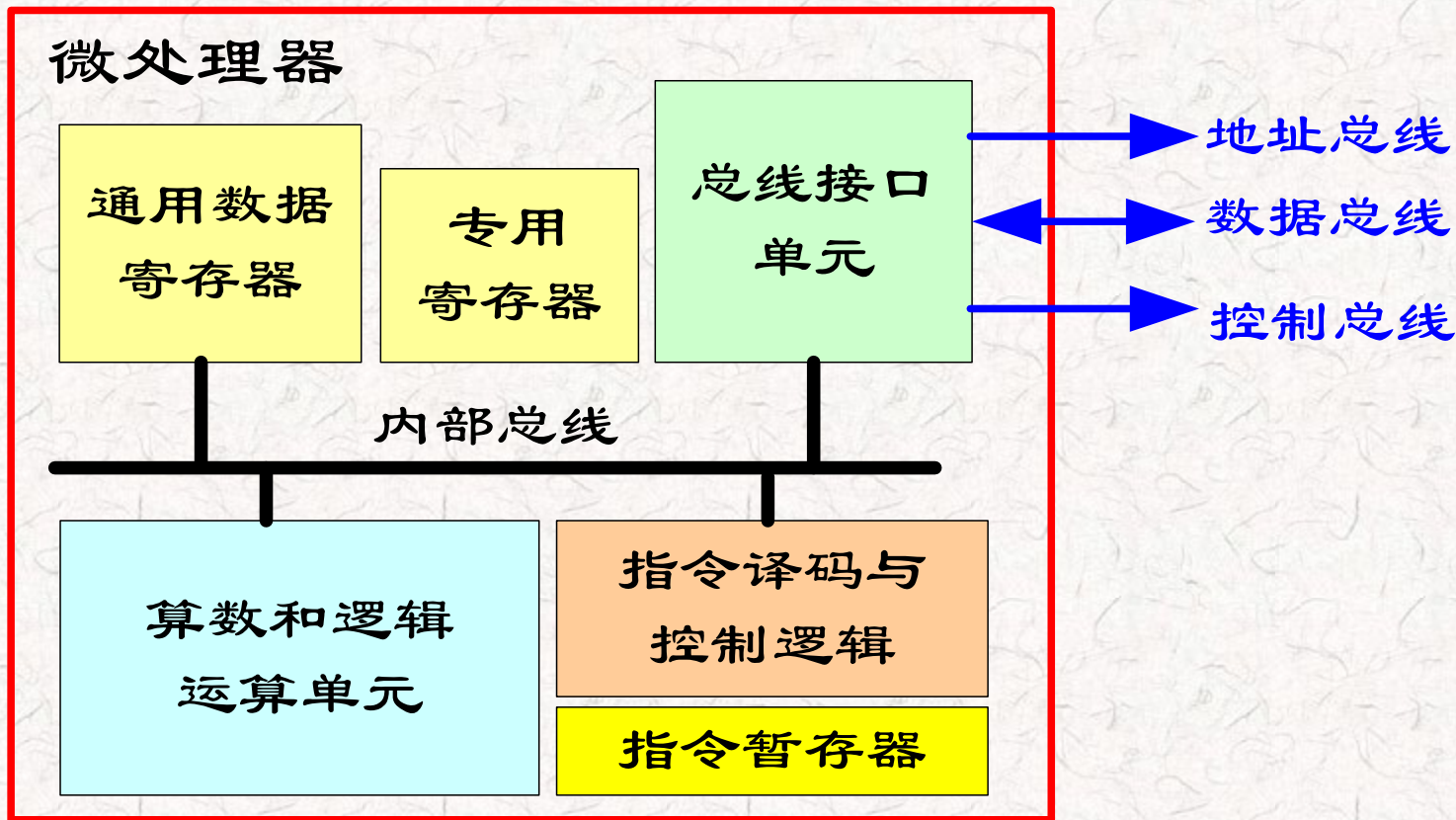
微处理器端读操作典型时序



微处理器端写操作典型时序

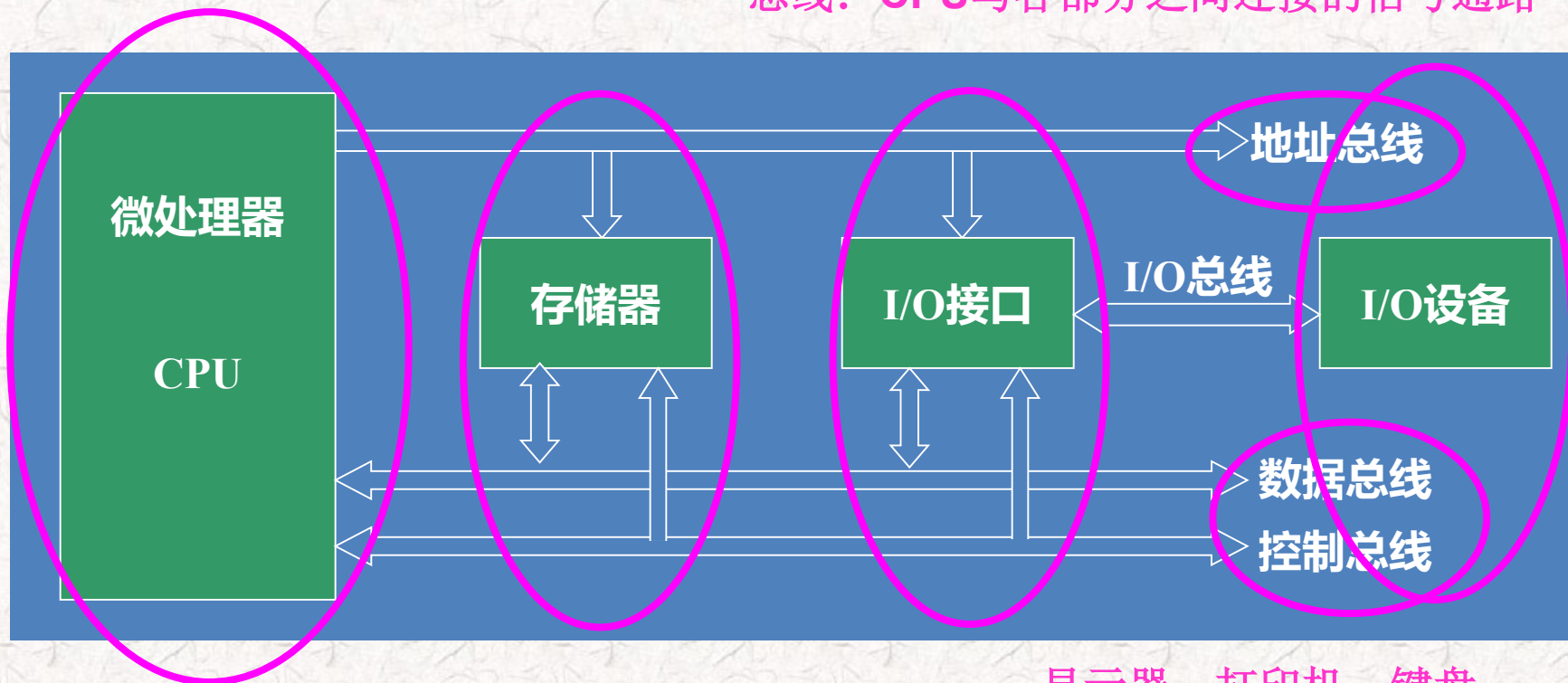
注意：一次读或者写操作
只能对一个地址进行

基本概念之：CPU内部基本组成



微型计算机硬件组成

总线：CPU与各部分之间连接的信号通路

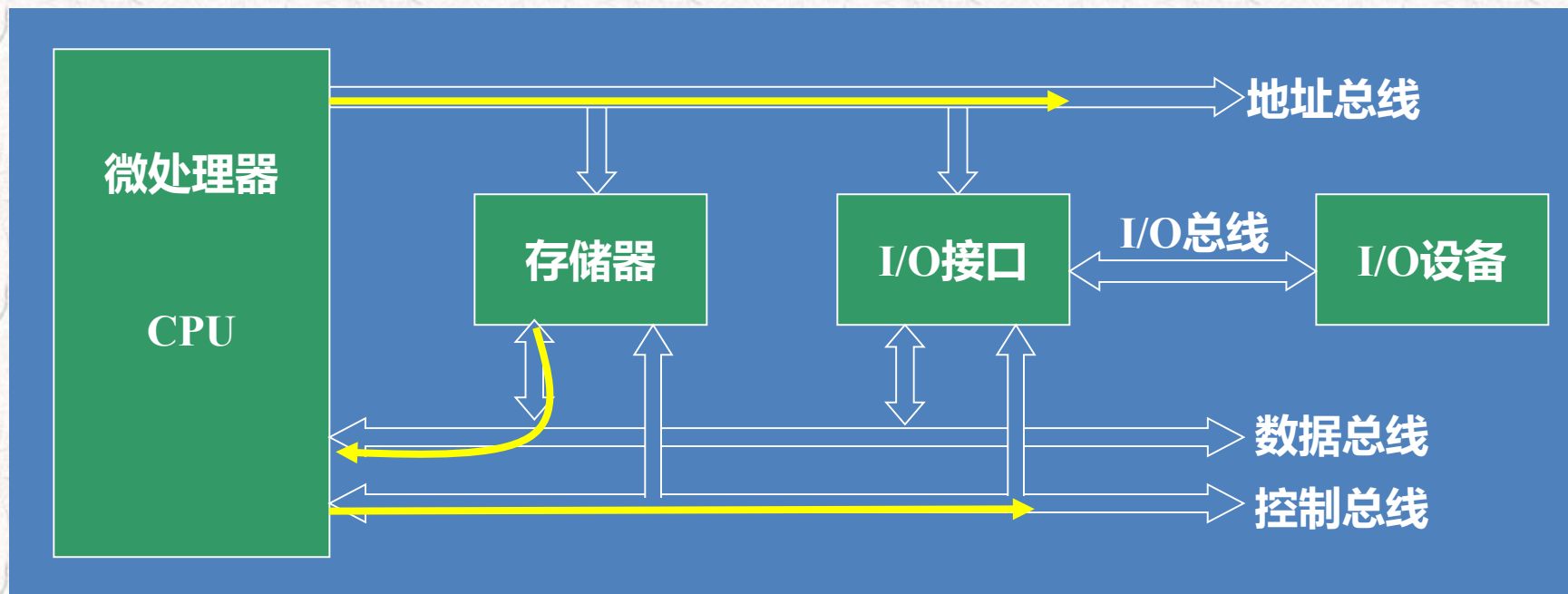


系统的核心控制部件 存放程序和数据

显示器、打印机、键盘、
显示灯、喇叭等

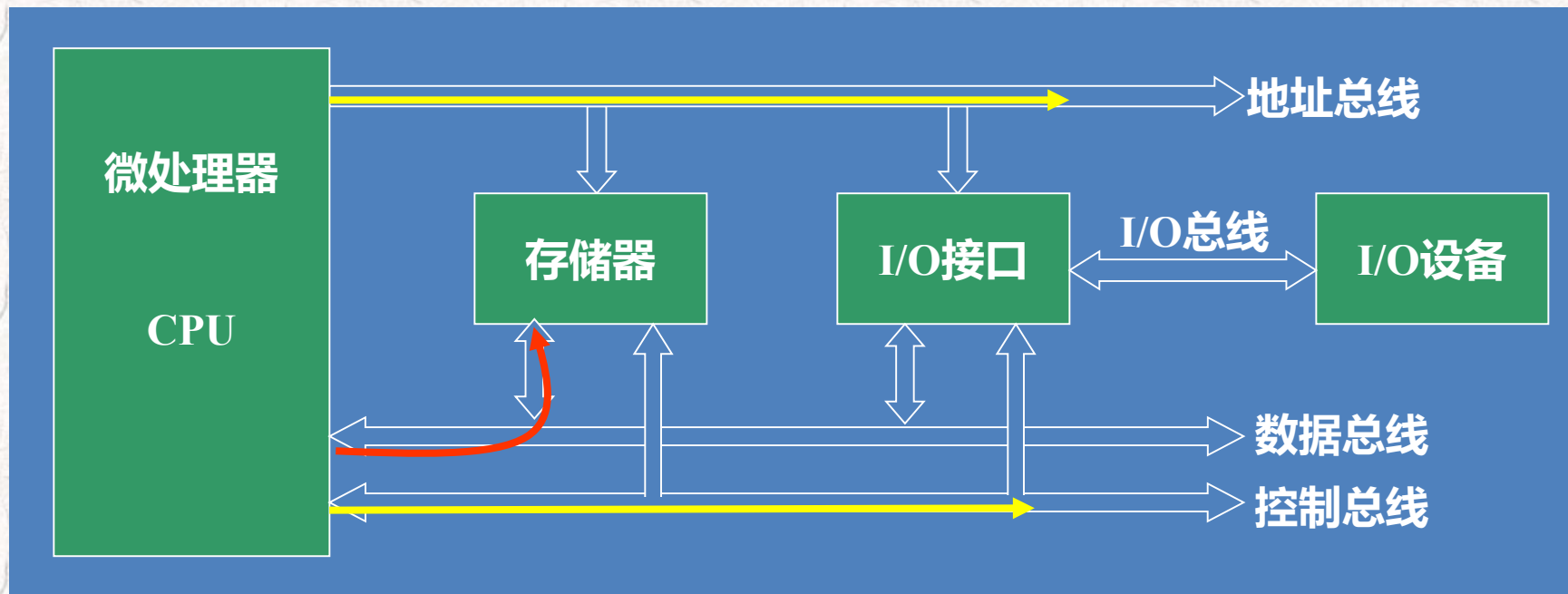
微处理器如何读取数据？

1. **CPU**向**地址总线**输出想要访问的存储器某单元（或**IO**端口）的**地址**
2. **CPU**向**控制总线**中的“**RD**”信号线发出低电平的**读请求**
3. **存储器**响应地址和读信号，向**数据总线**输出该存储单元（或**IO**端口）的**数据**



微处理器如何写出数据？

1. **CPU**向**地址总线**输出想要访问的存储器某单元（或**IO**端口）的**地址**
2. **CPU**向**控制总线**中的“**WR**”信号线发出低电平的**写请求**
3. **CPU**向**数据总线**输出将要写入该存储单元（或**IO**端口）的**数据**



微处理器指令执行过程

指令种类：
数据的搬移
加减或逻辑运算
数据移位
运算结果的判断
指令跳转，等

访问存储器，指令
以二进制数据的形
式保存在存储器中

在CPU内部对
指令进行分析
(译码)

取指令

分析指令

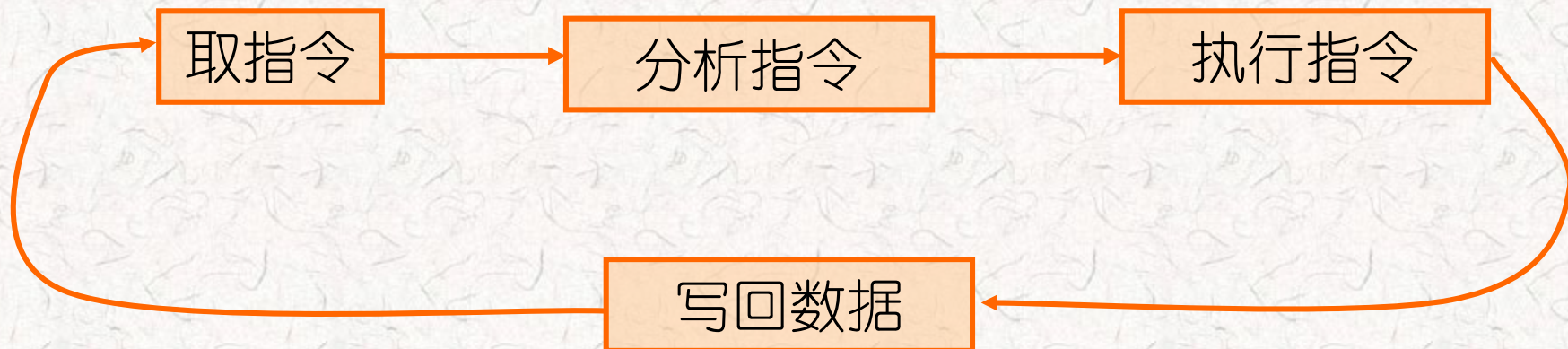
执行指令

写回数据

读取下一条指令

根据指令的要求将运算结
果保存到寄存器、存储器
或输出到IO端口

无数复杂功能是通过
上述过程重复完成的

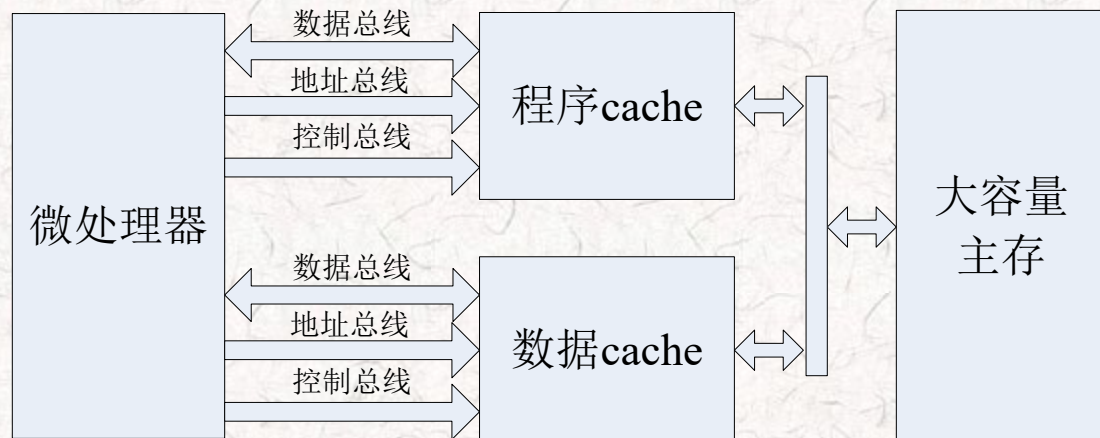


一点扩展知识--哈佛结构

- 冯诺依曼结构（单总线）：地址、数据、控制（三总线）
- 哈佛结构（双总线）：程序总线、数据总线
- 解决“存储墙”的方法之一



哈佛结构



现代应用更广泛的总线和存储结构

微处理器系统--计算机的组成

微处理器主
机硬件

- 计算核心：微处理器
- 存储器：内存，放程序和数据，二进制形式
- IO接口：连接微处理器与外设的接口
- 总线：连接微处理器、存储器、IO接口的共用的信号线
- 外设：硬盘、光驱、键盘、显示器、音响、显卡、网卡、优盘、移动盘，等
- 软件：运行在微处理器上的各种程序

微处理器
硬件系统

微处理
器系统

一个复杂的微处理器系统--桌面计算机系统

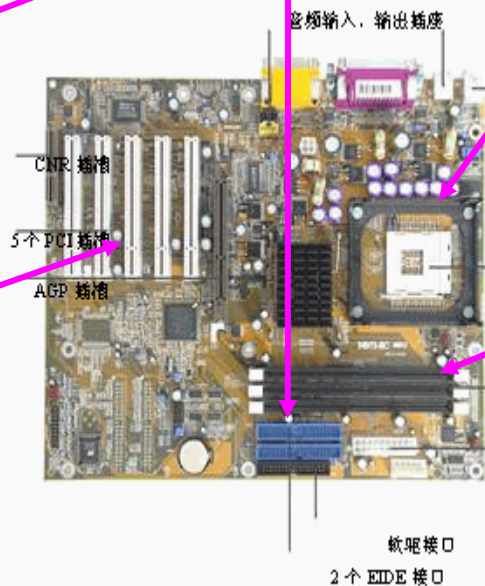


外设：
显示器
键盘
扬声器
鼠标
光驱
软驱

接口：
USB插口
耳机插口
(也是外设)

计算机内部

微处理器



键盘, 鼠标



Socket CPU 插座

SDRAM 内存插槽

电源接口

软驱接口
2个 EIDE 接口



电脑配置网
WWW.023DN.COM

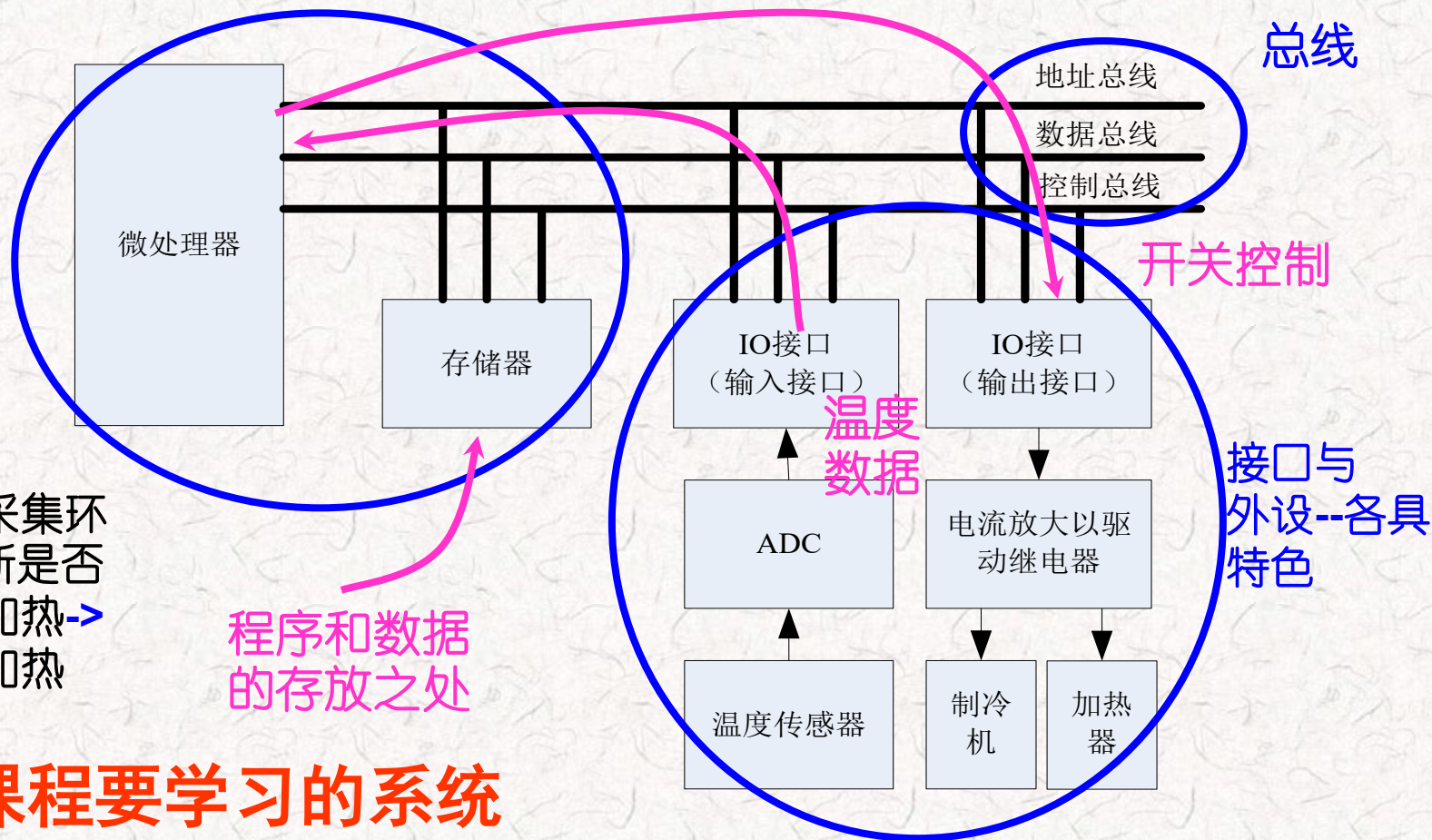
一个简单的微处理器系统

大棚温度调节系统

微处理器
与存储器--
核心组成

系统功能：采集环境
温度->判断是否需要
制冷或加热->控制
制冷或加热

这是本课程要学习的系统



学习方法-软硬件结合

课程内容与学习方法

第2章
8086CPU和系统的
整体介绍

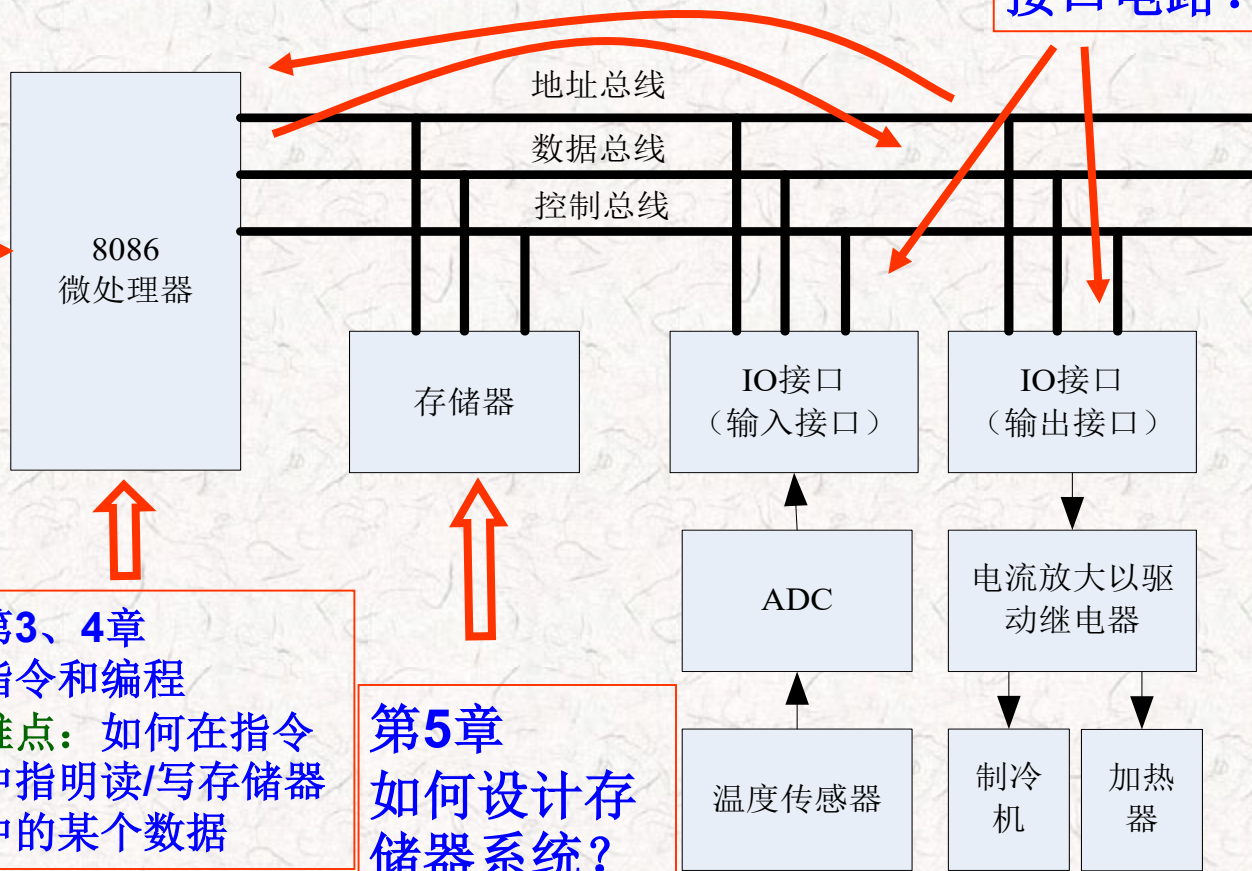
8086功能?
内部结构?
对存储器系统有什么要求?
怎么指定存储器的地址?
如何生成总线?
访问存储器/IO的总线周期是什么?

第3、4章
指令和编程
难点: 如何在指令中指明读/写存储器中的某个数据

第5章
如何设计存储器系统?

第8章
8086的中断管理系统

第6、7章
如何设计IO接口电路?



补充基本概念-3

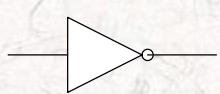
- 逻辑运算和逻辑门
- 取反、与、或、与非、或非、异或
- 驱动器（缓冲器）、三态驱动器（三态缓冲器）
- 输出冲突和地址选通的概念
- 译码器
- 编码器
- 锁存器

逻辑运算

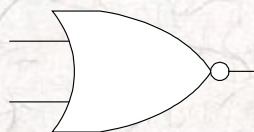
- 取反---表示法：NOT、上划线、！、/
- 与-----表示法：AND、&、 \wedge 、 \cdot
- 或-----表示法：OR、|、 \vee 、+
- 与非---表示法：NAND
- 或非---表示法：NOR
- 异或---表示法：XOR、 \otimes

常见逻辑门

注意信号传输方向



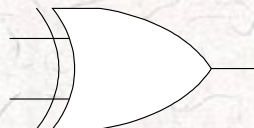
$$Y = \overline{A}$$



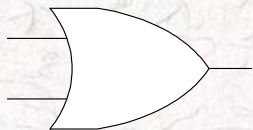
$$Y = \overline{A + B}$$



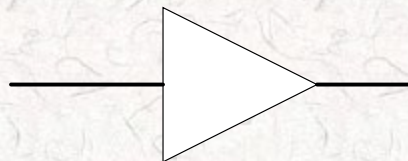
$$Y = A \cdot B$$



$$Y = A \otimes B$$



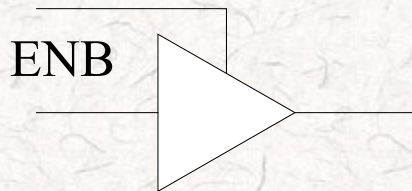
$$Y = A + B$$



驱动器

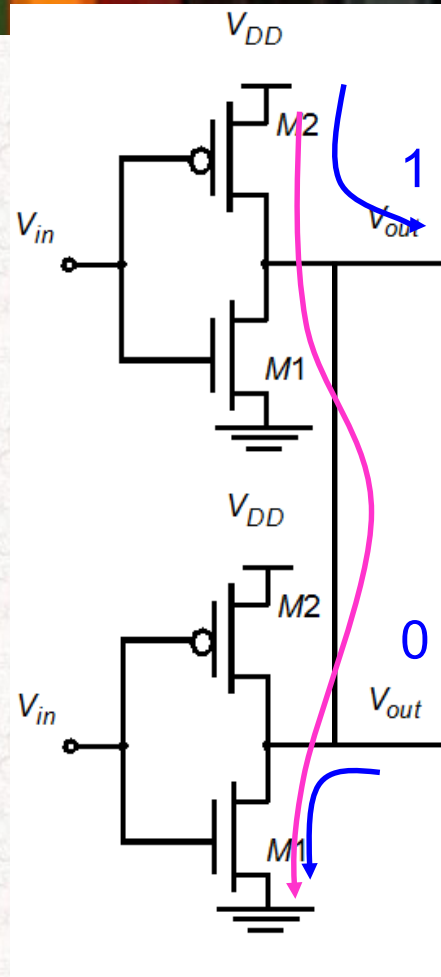
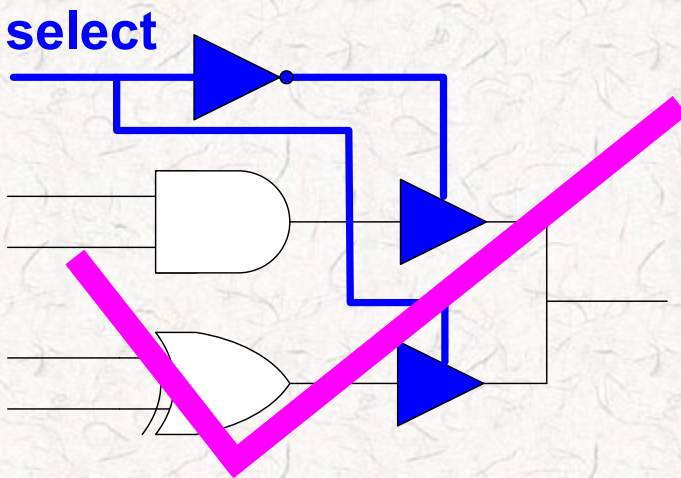
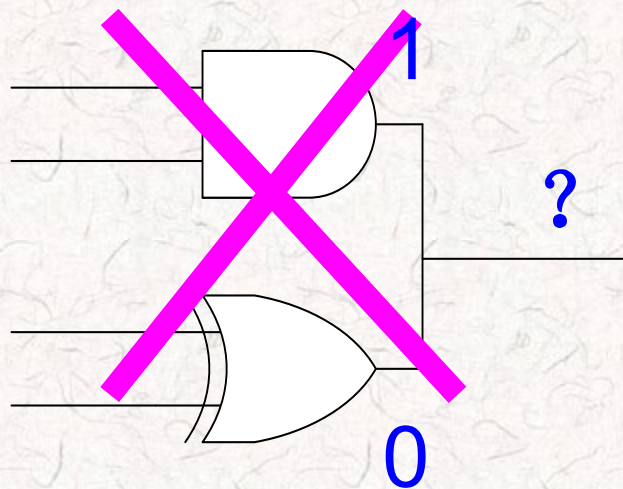


$$Y = \overline{A \cdot B}$$



三态驱动器

输出冲突的概念

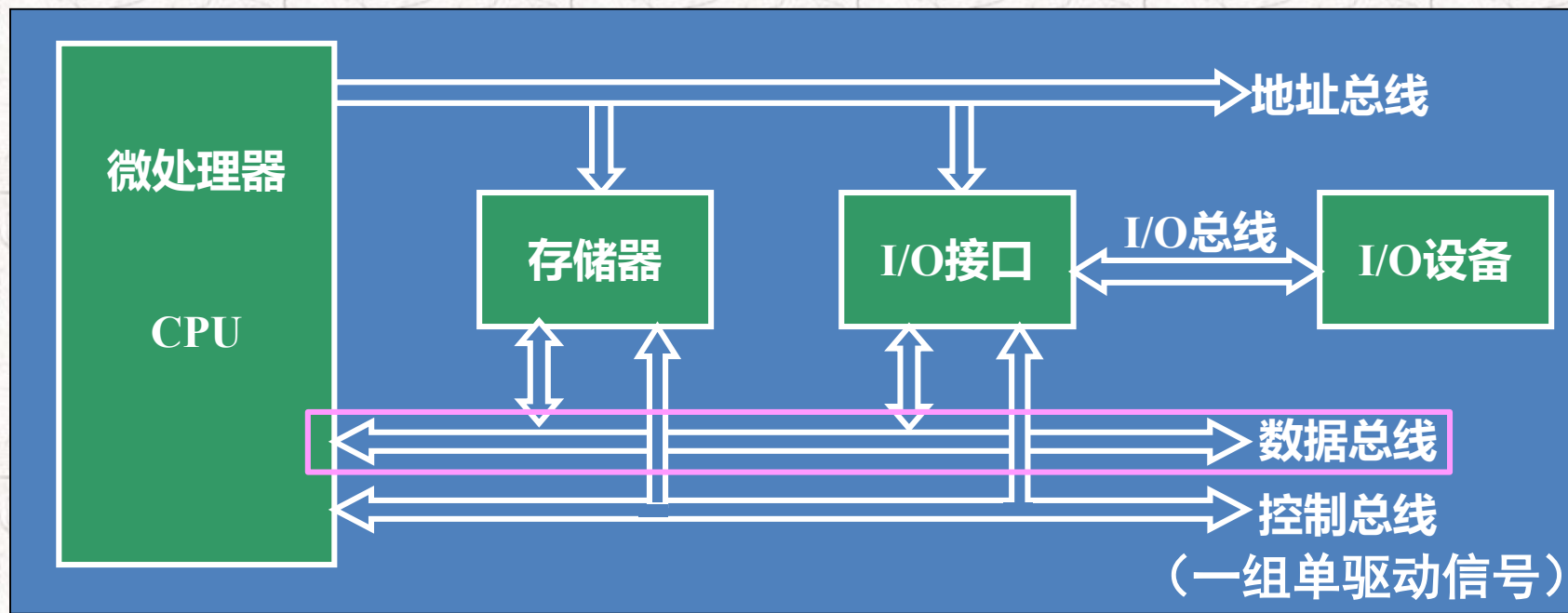


连接在一起的输出端同一时间只能有1个有效

微处理器系统哪里可能有输出冲突？

地址和控制线单向传输，一个信号只有一个驱动方；
数据总线双向传输是多输出驱动的，可能的输出有：

(1) CPU (2) 存储器 (3) IO接口



如何避免数据总线冲突

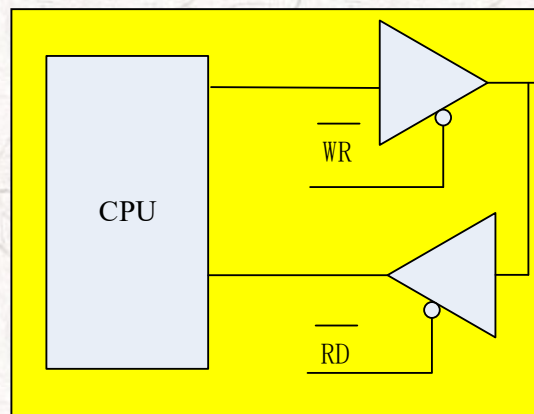
注意：输入是
没必要高阻的

两个保障措施：

1. 采用三态驱动器

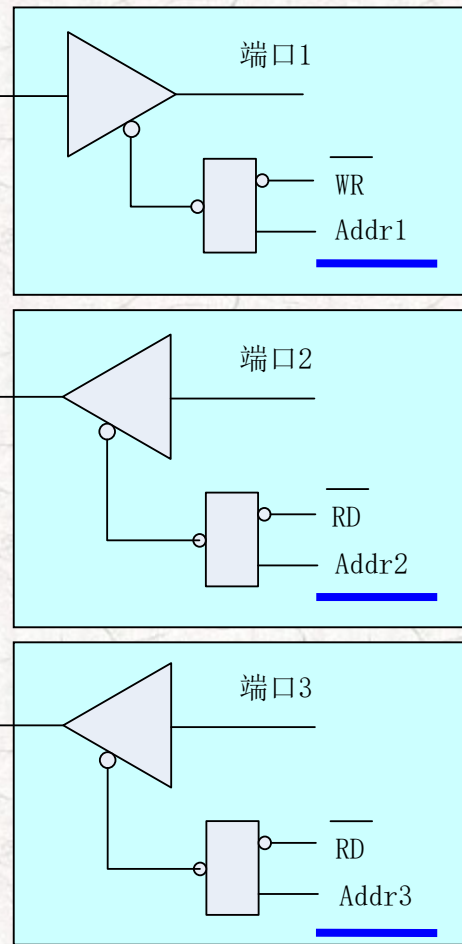
2. 读写信号+分配地址空间，同一时间最多允许一个端口输出驱动，其它高阻

地址译码
器实现



数据总线

总线上连接了几组信号？
哪些是输出驱动？



译码器的功能

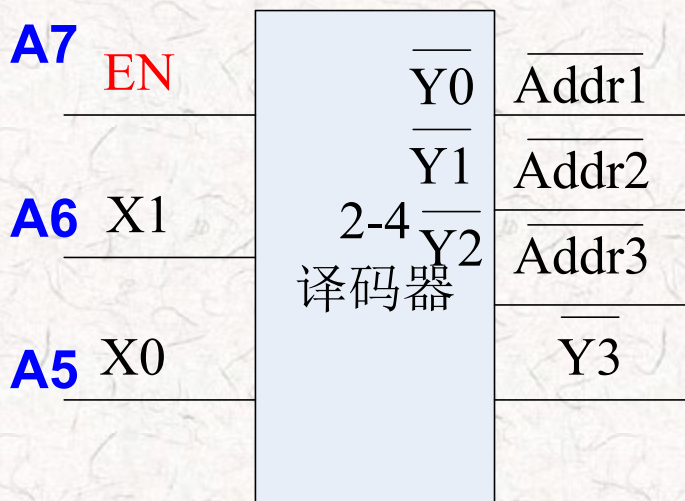
- 将二进制数据的每一个组合赋予特定的含义，给出对应的信号，实现这个“翻译”过程的叫做译码器
- 例如：2bit数据的每个组合：00，01，10，11，各表示一种含义

X1	X0	$\overline{Y3}$	$\overline{Y2}$	$\overline{Y1}$	$\overline{Y0}$
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

译码器：一组输入最多只能有一个输出有效，输出有效信号是互斥的

译码器如何将地址转为使能信号？

举例：设8bit地址



使能信号EN有效时，输出是输入译码得到的

使能信号EN无效时，输出全部无效

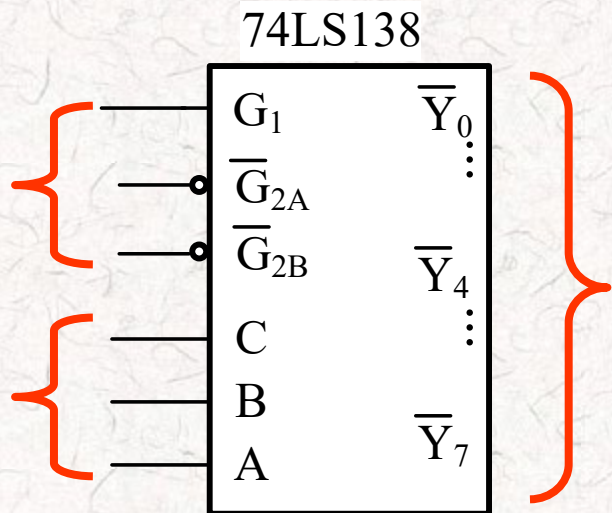
译码器常用于分配地址空间，输入为地址，输出为该地址对应的选通信号

A7	A6	A5	A4	A3	A2	A1	A0	输出	地址范围
1	0	0	X	X	X	X	X	/Addr1	80H~9FH
1	0	1	X	X	X	X	X	/Addr2	0A0H~0BFH
1	1	0	X	X	X	X	X	/Addr3	0C0H~0DFH
1	1	1	X	X	X	X	X	/Y3	0E0H~0FFH

微处理器地址总线每次读/写操作只能输出一个地址，
因此同一时间最多只有一个译码输出信号有效

本课程中常见的译码器

74LS138译码器在使能有效情况下的输入和输出



C	B	A	\overline{Y}_7	\overline{Y}_6	\overline{Y}_5	\overline{Y}_4	\overline{Y}_3	\overline{Y}_2	\overline{Y}_1	\overline{Y}_0
0	0	0	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	0	1
0	1	0	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	1	1	0	1	1	1	1
1	0	1	1	1	0	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1

74ls138译码器的应用举例

- 如某微处理器有8根地址线，其可寻址地址空间有多大？
- 如果8bit地址用于译码，A7 ~A2分别接G1到A，请问/Y0、/Y3所对应的地址空间？

• 答案

• 2的8次方，256个

• G1 /G2A /G2B C B A

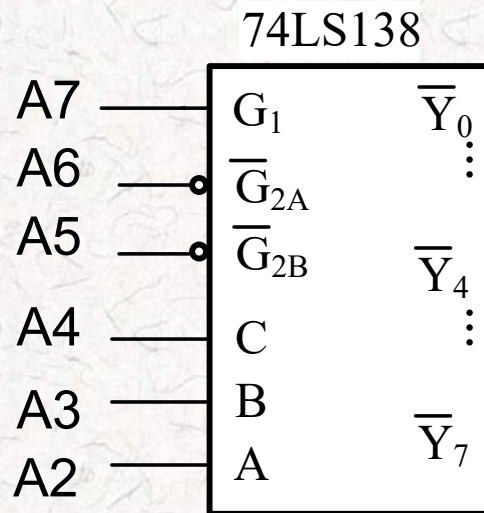
• A7 A6 A5 A4 A3 A2 A1 A0

• 1 0 0 0 0 0 X X

• 1 0 0 0 1 1 X X

• Y0对应80H~83H

• Y3对应8CH~8FH



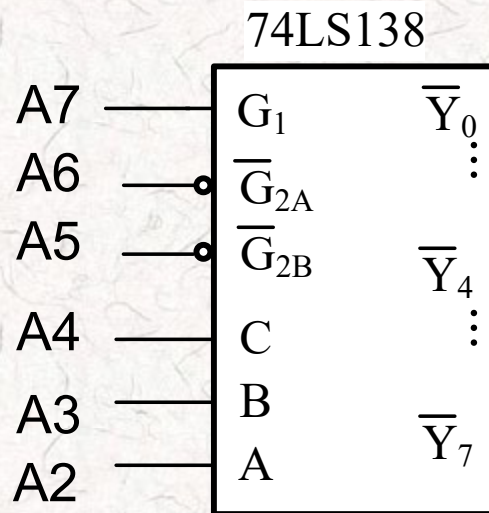
引申: Verilog描述y0,y3生成逻辑

```
always @(*)  
  if(A[7:2]==6'b100000)  
    y0=1'b0;  
  else y0=1'b1;
```

```
always @(*)  
  if(A[7:2]==6'b100011)  
    y3=1'b0;  
  else y3=1'b1;
```

```
assign y0=(A[7:2]==6'b100000) ? 1'b0 : 1'b1;  
assign y3=(A[7:2]==6'b100011) ? 1'b0 : 1'b1;
```

```
assign y0=(A[7:0]>=8'h80 && A[7:0]<=8'h83) ? 1'b0 : 1'b1;  
assign y3 =(A[7:0]>=8'h8c && A[7:0]<=8'h8f) ? 1'b0 : 1'b1;
```



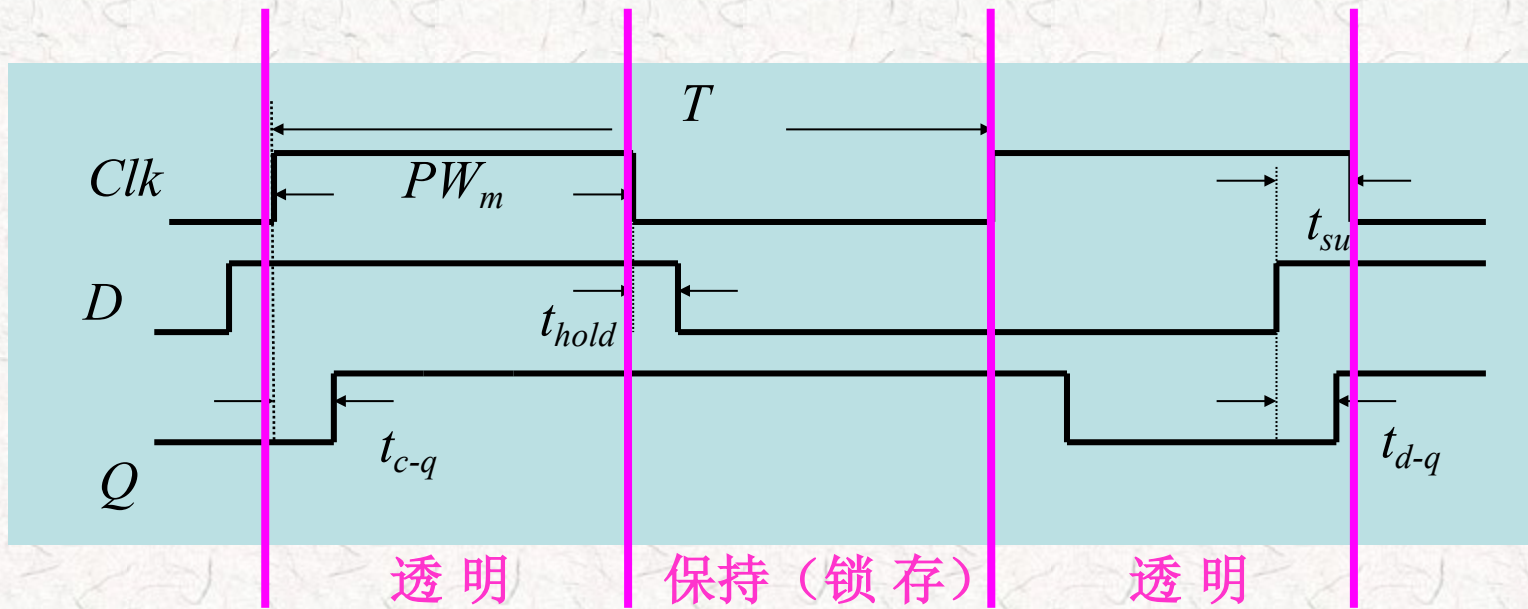
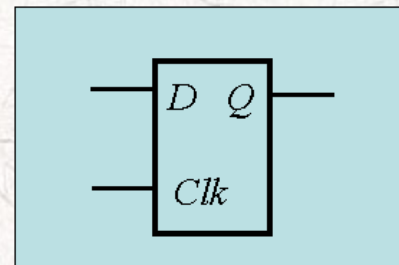
编码器

- 与译码器相反，对输入的某个信号赋予唯一的编码

$\overline{Y3}$	$\overline{Y2}$	$\overline{Y1}$	$\overline{Y0}$	A1	A0
1	1	1	0	0	0
1	1	0	1	0	1
1	0	1	1	1	0
0	1	1	1	1	1

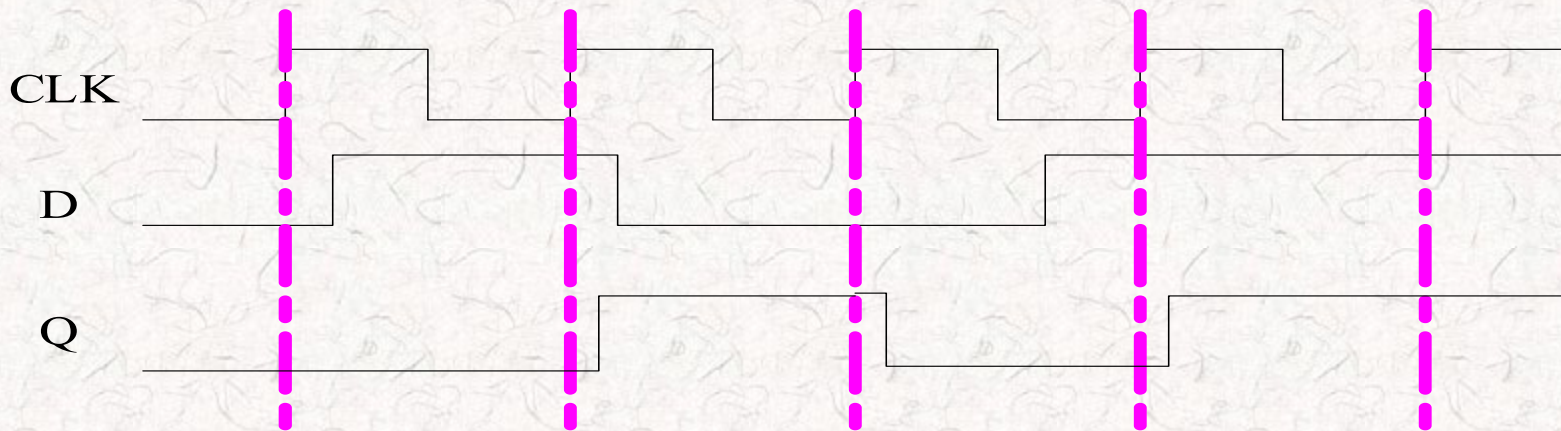
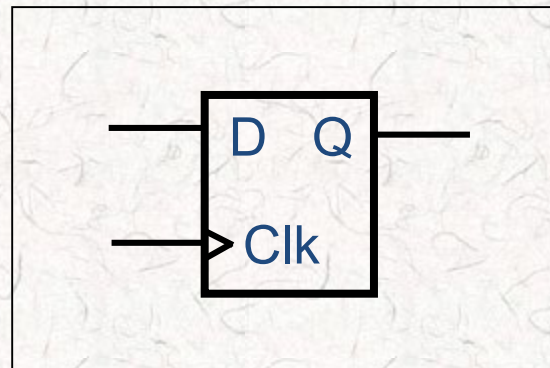
时序元件-锁存器

- 时序元件：能够保存数据的元件
- 锁存器：受CLK 电平控制
- 透明 / 保持 两种状态



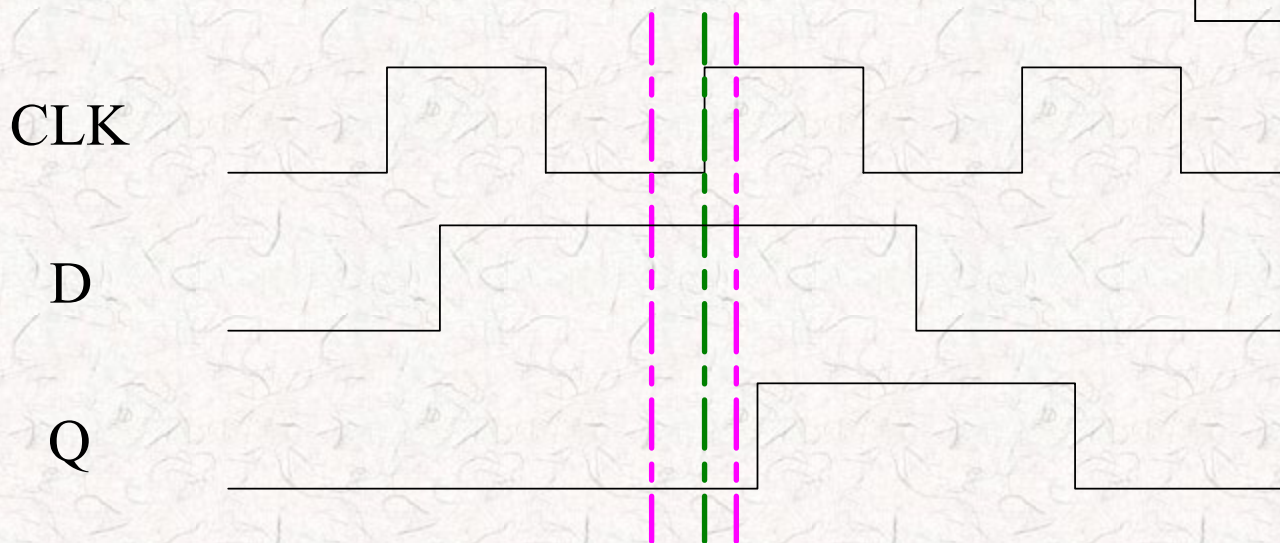
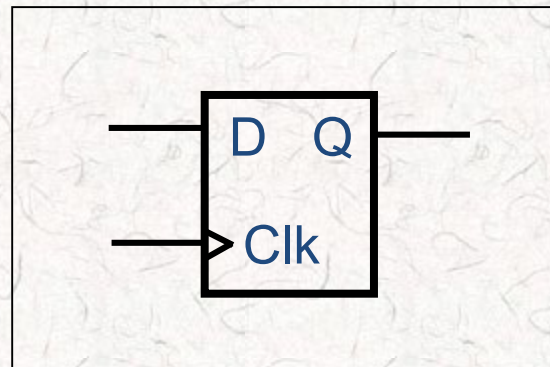
时序元件-触发器

- 触发器：时钟 **上跳沿** 更新状态
- 在时钟沿之后，输出更新为输入在时钟沿之前的状态



引申：时序元件的时序要求

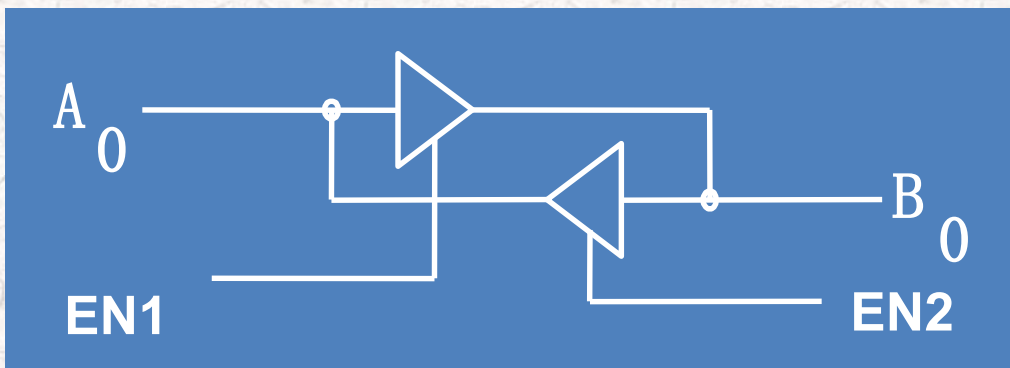
- 时序元件锁存数据时，在时钟沿附近，输入D应保持稳定，否则Q端出现亚稳态，数据锁存错误



读图练习

- EN1、EN2高电平有效，试分析以下输入时电路的功能
- EN1=1, EN2=0 A0 --> B0
- EN1=0, EN2=1 B0 --> A0
- EN1=EN2=0 B0 A0 相互隔离
- EN1=EN2=1 非法状态

双向缓冲器



总结：本章学习了什么？

- 计算机内部有符号数编码方式：补码
- 微处理器硬件系统的组成：微处理器，总线，存储器，IO接口和IO设备
- 微处理器基本功能：顺序读取指令，执行指令
- 微处理器指令执行的基本流程：取值，执行，结果写回
- 如何解决数据总线多输出并联冲突：（1）所有输出通过三态驱动器连接到数据总线；（2）三态驱动器不能同时输出驱动，微处理器输出或者微处理器根据地址指定某个端口输出有效；

还有问题吗？

作业

**P53 12, 13, 14(8086部分),
17, 18**



预告：第二章需要掌握的内容

- 学习8086处理器概述，最小模式系统的组成和特点
- 除以下内容外都需要掌握
 - 1. 最大模式相关内容
 - 2. 时钟发生器8284A
 - 3. 8088处理器与8086的区别