

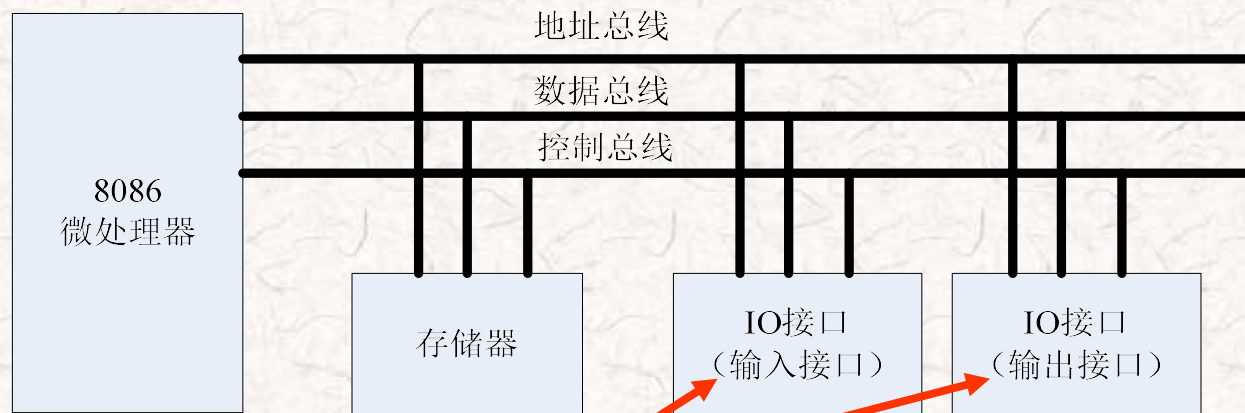


# 微机原理与接口技术

## •第六章 IO接口与8255芯片

# 课程内容之IO接口

继续理解指令执行与  
IO接口电路信号的对应关系



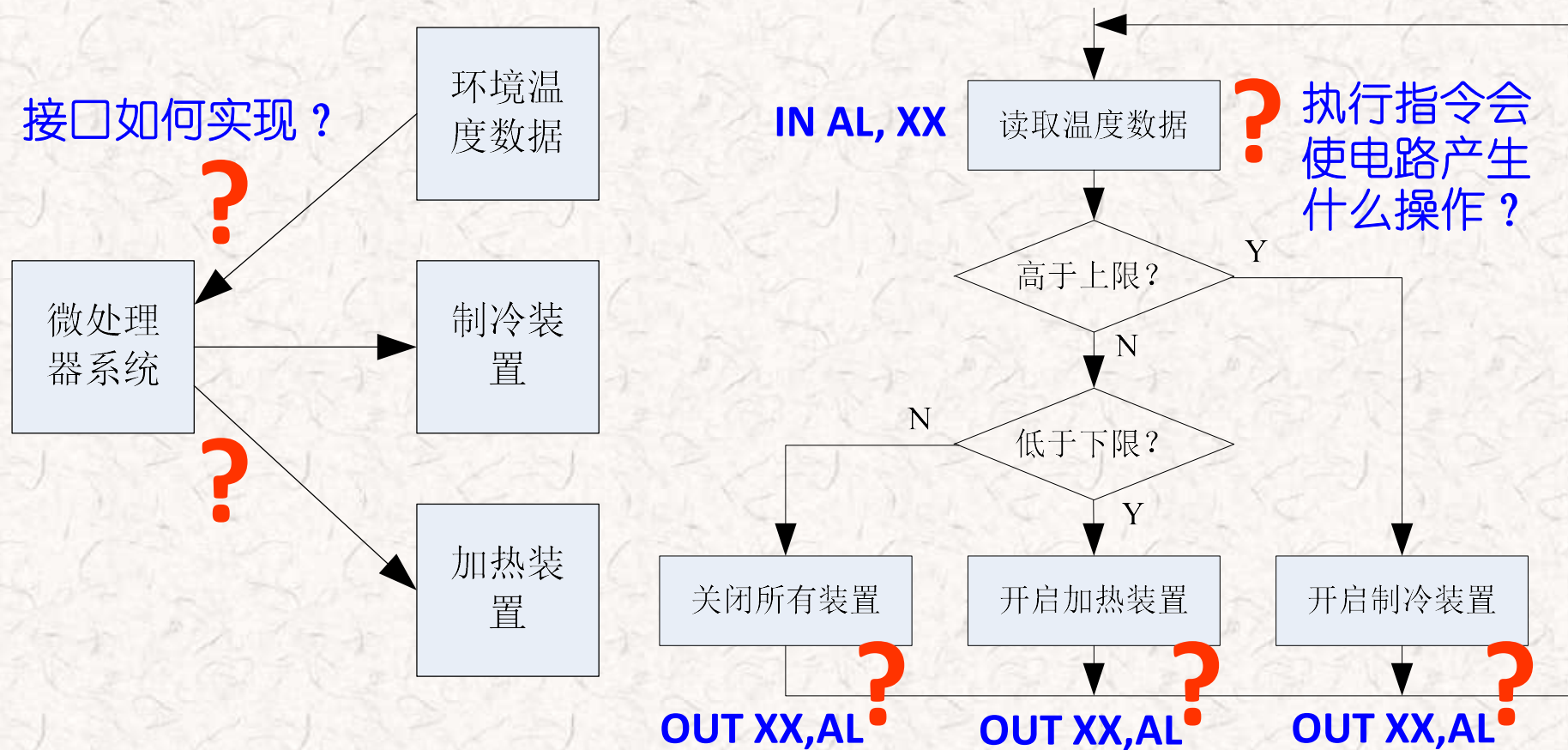
## 第6章

简单例子：如何设计温度控制系统的输入和输出接口电路？

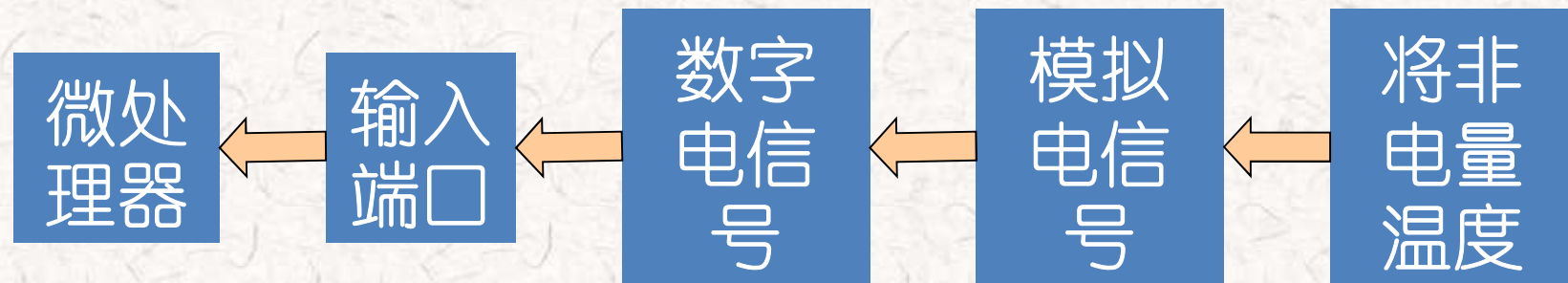
几种常见的IO接口方式

复杂例子：如何用通用器件或8255构造键盘和数码管显示IO电路？

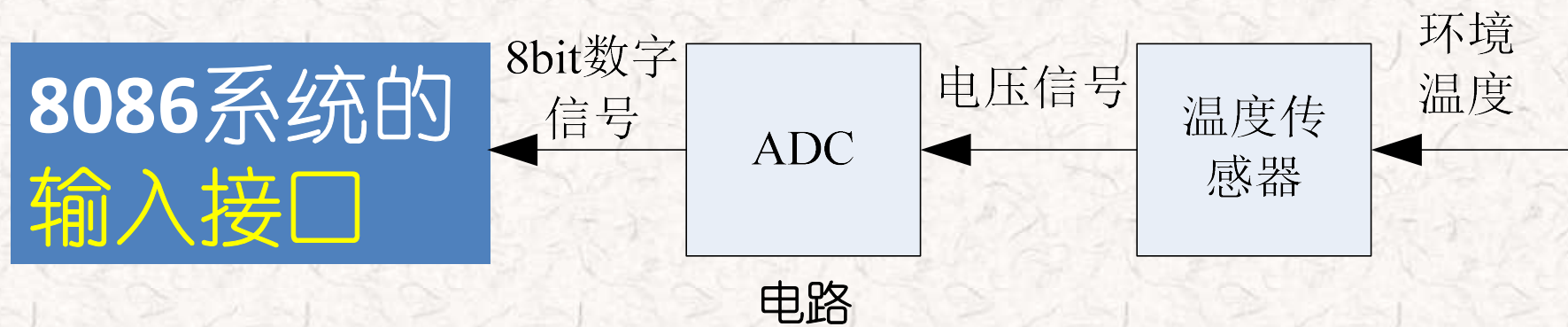
## 例子--如何构造一个温度控制系统？



# 如何采集环境温度？



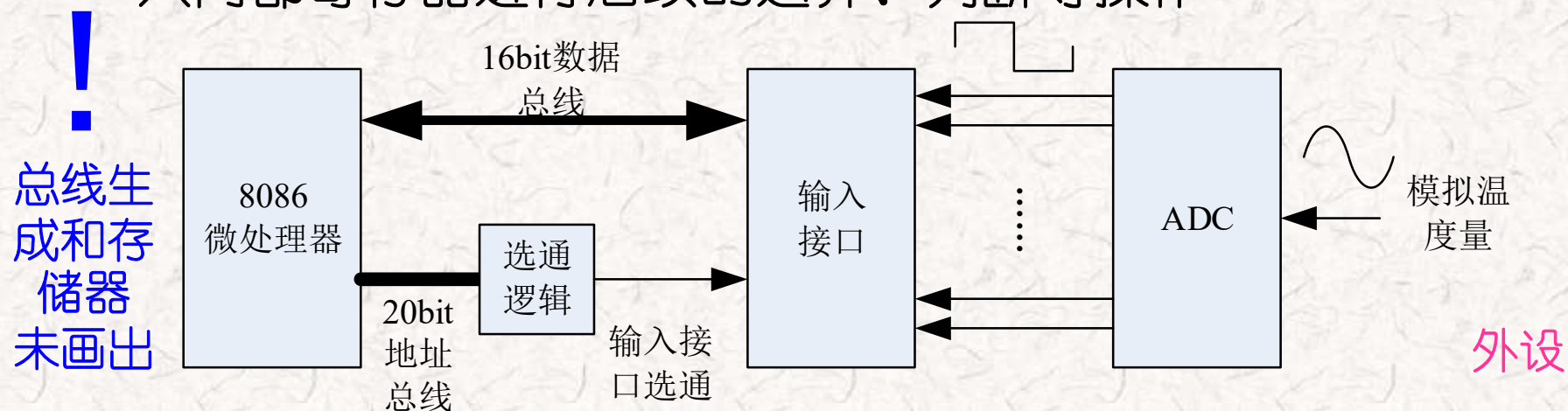
处理流程





# 8086系统的输入接口-温度采集

- 温度采集通过输入接口（连接外设与微处理器）实现
- 外设：温度传感器+ADC
- 微处理器：8086微处理器+存储器
- 输入接口：使8086能够通过总线将ADC输出的数字信号读入内部寄存器进行后续的运算、判断等操作

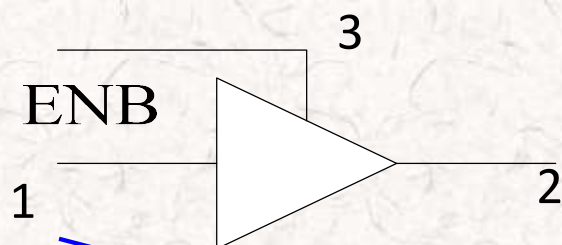


# 温度输入接口如何构成？

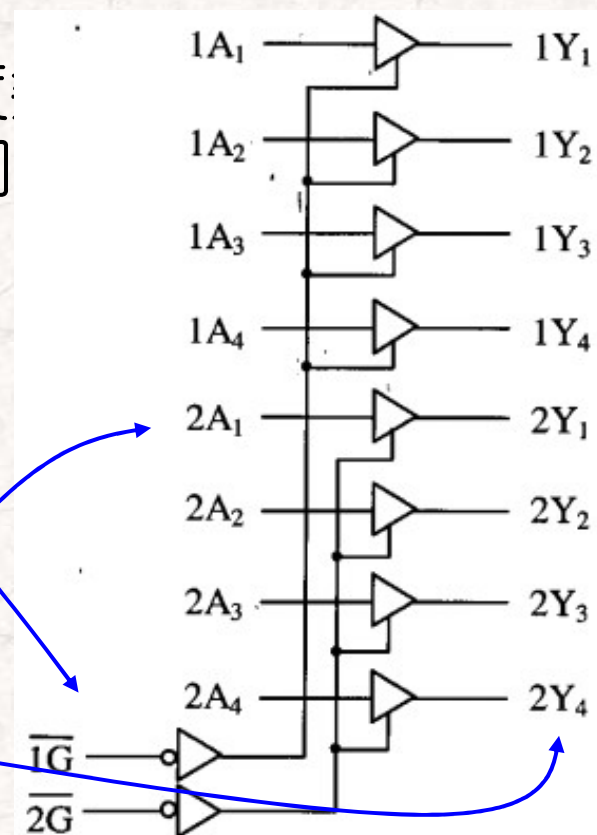
- 功能分析
- 在CPU执行读此IO端口操作时，将温度，从而进入CPU内部寄存器；其它时间
- 用三态缓冲器实现，如何连接？
- 具体构成：可使用哪些缓冲器芯片？

IO端口地址译码及读信号


ADC输出  
数字信号

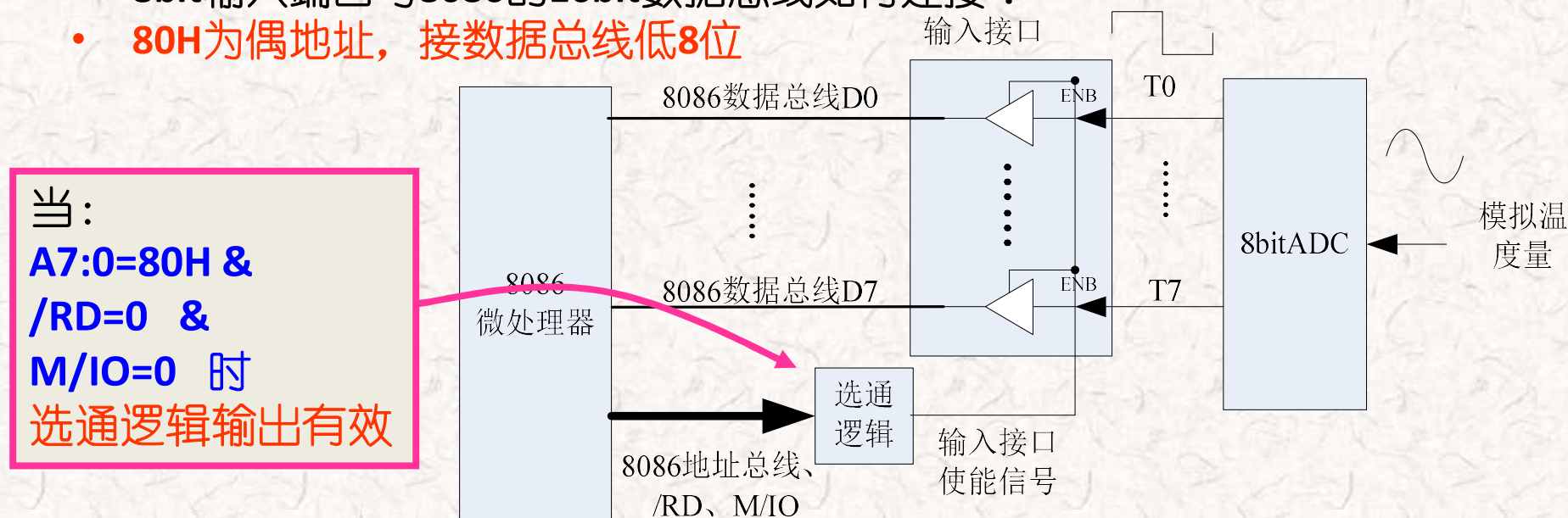


74LS244



## 温度输入接口连接图

- 问题：如何在8086系统中加入244构成的温度采集输入接口？设系统分配给温度端口的地址为80H，选通逻辑电路应接入哪些信号？CPU执行什么操作的时候电路产生有效选通信号以得到温度数据？
  - 地址、/RD，M/IO，执行指令 IN AL, 80H 时
  - 8bit输入端口与8086的16bit数据总线如何连接？
  - 80H为偶地址，接数据总线低8位
- 





# 8086执行指令 IN AL, 80H时选通逻辑输出

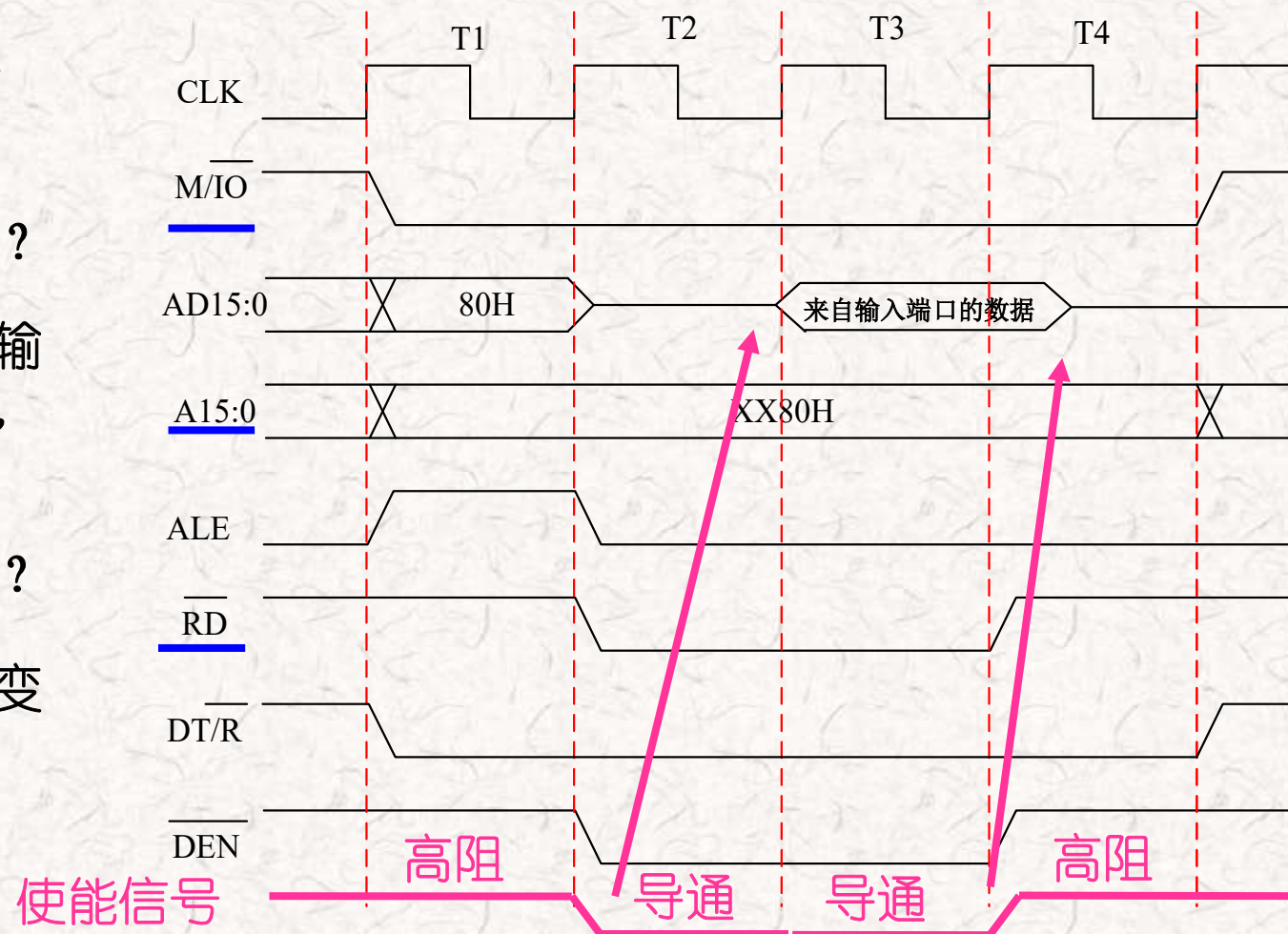
## 变化情况

8086总线信号如何？

设选通逻辑输出为输入接口的使能信号，  
低电平有效

输入接口如何变化？

8086数据总线如何变化？





## 简单输入端口总结

- (1) 功能？(2) 用什么逻辑电路构造？如何接入系统？
- (3) 选通信号功能？(4) 选通信号实现逻辑？
- 输入端口用于将外设准备好的数字信号通过数据总线送给微处理器
- 采用三态缓冲器实现，其输入为外设希望送给微处理器的数字信号，输出连接微处理器数据总线，使能端受选通信号控制
- 选通信号功能：当微处理器对该输入端口执行读操作时，选通信号有效，使该端口处于导通驱动状态，其它时间为高阻状态，放弃数据总线控制权
- 选通信号生成逻辑：地址总线输出地址=端口地址， $\text{/RD}$ =低电平， $\text{M/IO}$ =低电平，三者同时有效

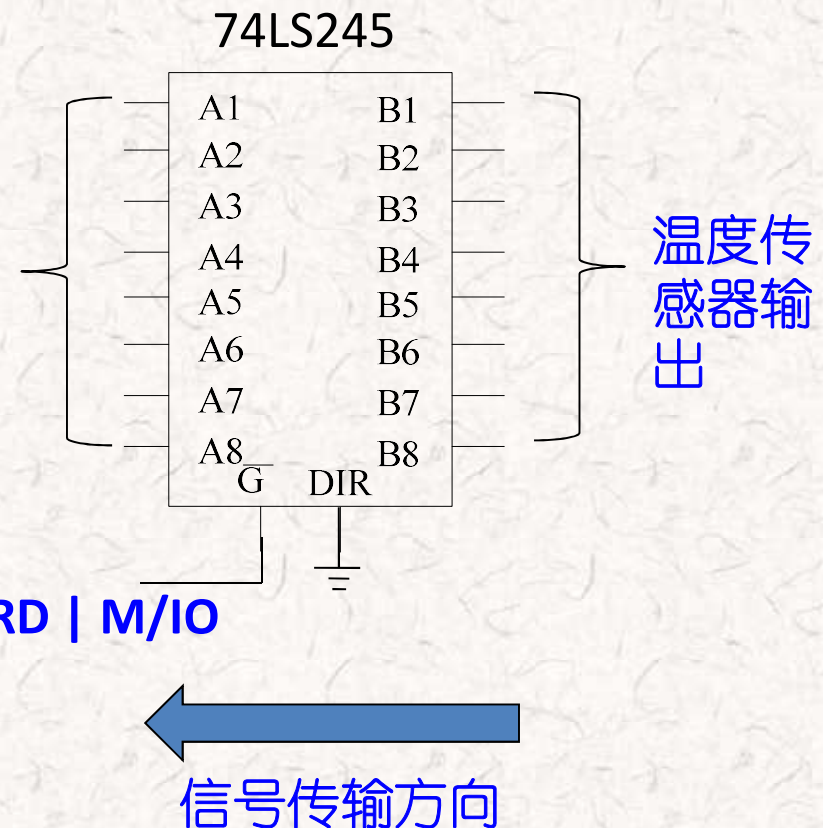
# 举例一如何用74LS245构造采集温度的输入接口

**/G:** 输出使能, 驱动 **or** 高阻  
**DIR:** 方向控制, 输出 **or** 输入

微处理器  
数据总线

$\overline{G}$	DIR	功能
0	0	B→A
0	1	A→B
1	X	高阻

选通信号  
地址译码 | /RD | M/IO

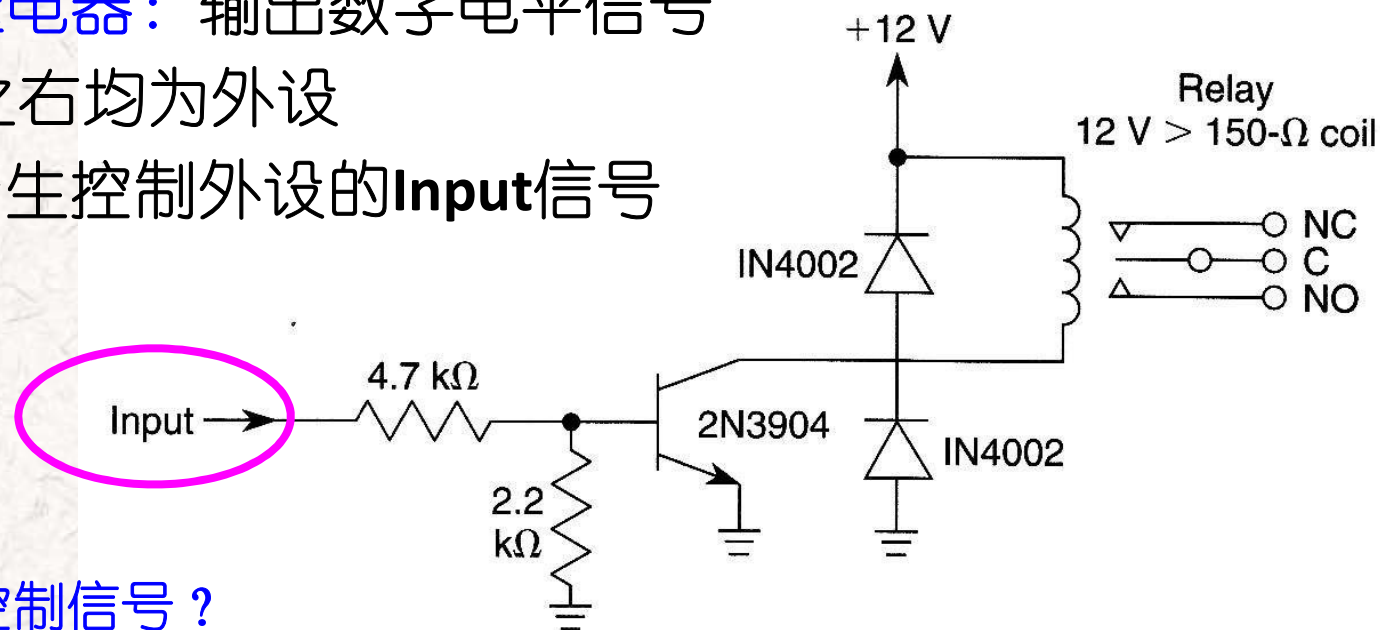


## 输出控制--如何实现加热或制冷？

- 所需设备：启动/停止可控的加热设备和制冷设备
- 如何启动/停止：继电器接通或断开
- 如何控制继电器：输出数字电平信号
- 下图**Input**之右均为外设
- 输出接口产生控制外设的**Input**信号

该信号需要能够  
维持高或低电平

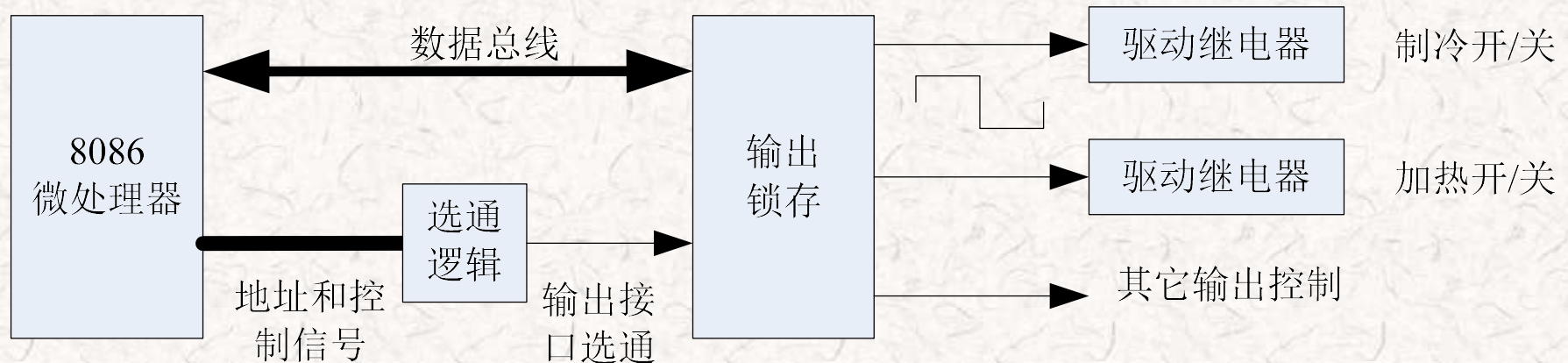
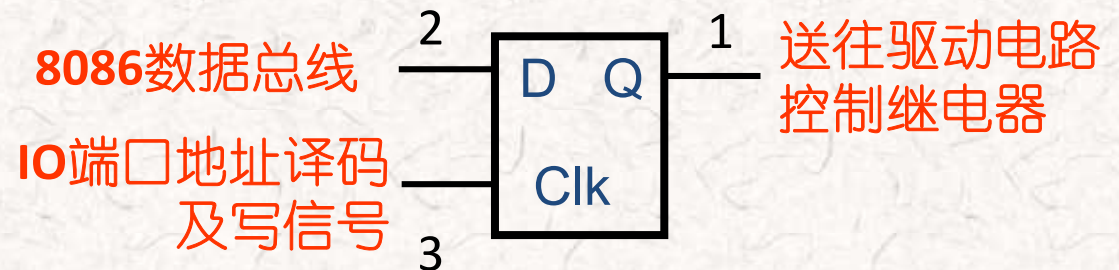
如何产生该输出控制信号？





# 如何保持输出信号？

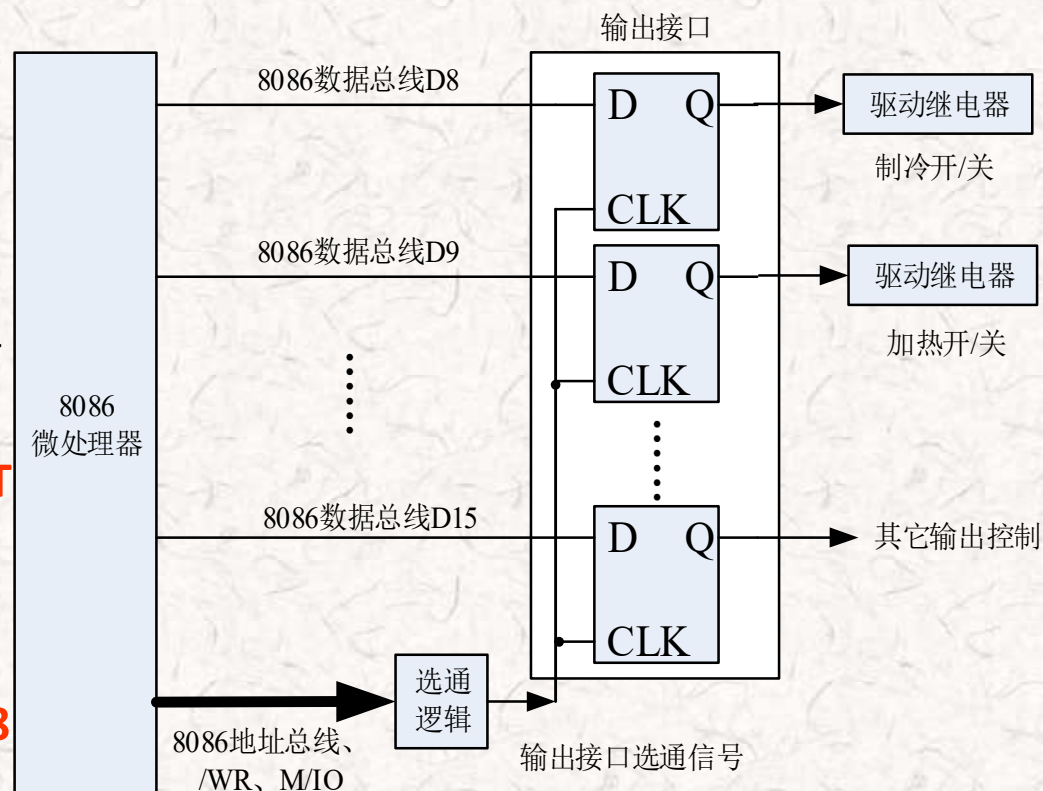
- 8086 IO输出: **OUT ADDRESS\_IO, AL**, 数据总线上的数据能保持吗？
- 锁存器
- 各端口如何连接？
- 如何接入8086系统？





# 输出控制接口连接图

- 问题：在8086系统中加入输出控制接口，设系统分配给输出控制接口的地址为81H，选通逻辑应接入哪些信号？CPU执行什么操作的时候电路产生该输出接口的有效选通信号？
- 地址、/WR，M/IO，执行指令 OUT 81，AL 时
- 8bit输出端口与8086的16bit数据总线如何连接？
- 81H为奇地址，接8086数据总线高8位



选通信号逻辑：  
地址=81H & /WR=0 & M/IO=0

# 8086执行指令 OUT 81H,AL时选通逻辑输出

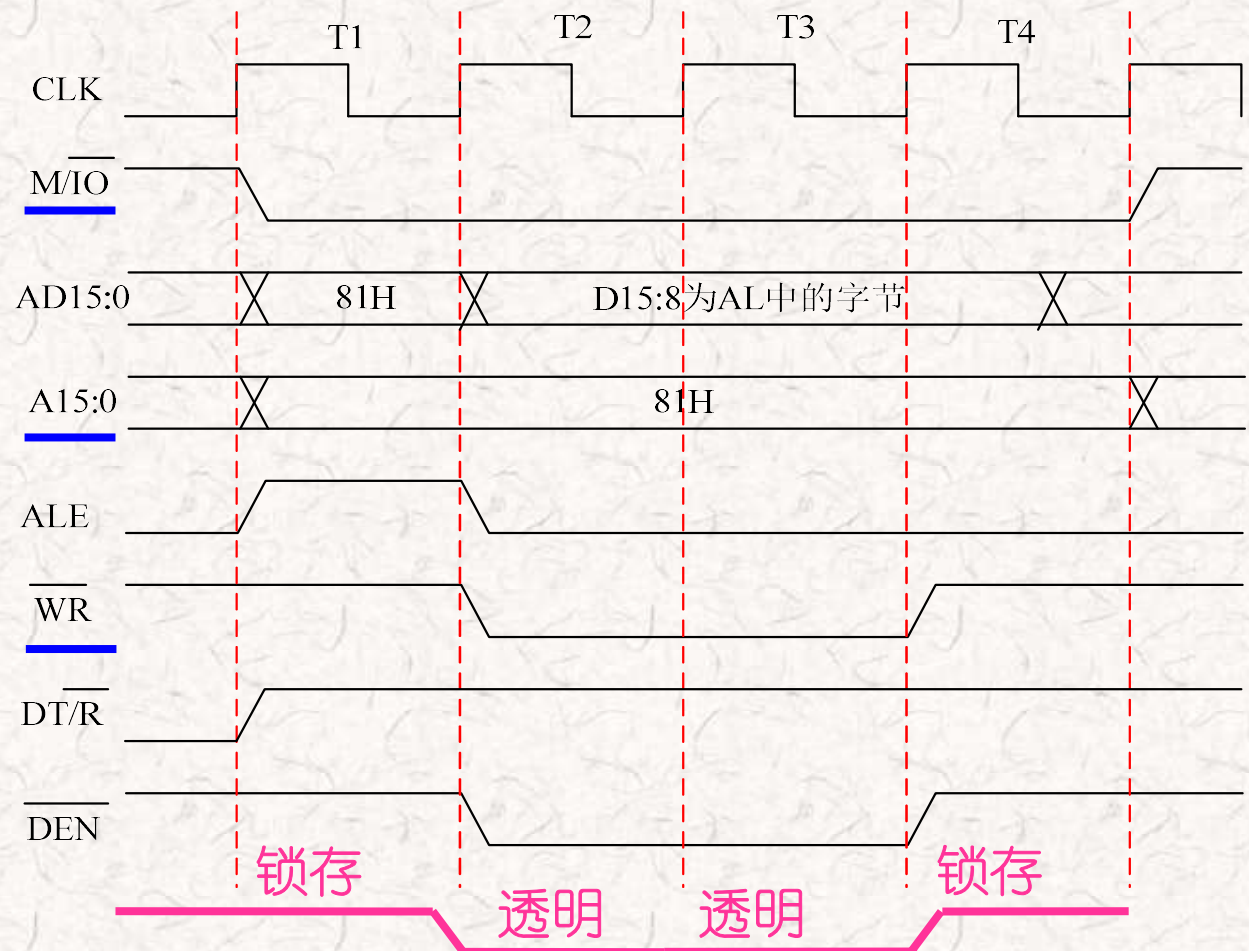
## 变化情况

8086总线信号如何？

设选通逻辑输出为输出接口的导通使能信号，低电平透明，高电平锁存

输出接口工作状态如何变化？

输出接口的输出数据如何变化？



# IO端口选通信号生成举例

设输入端口地址为  
**80H**，输出端口地址  
为**81H**，设选通信号  
低电平有效：



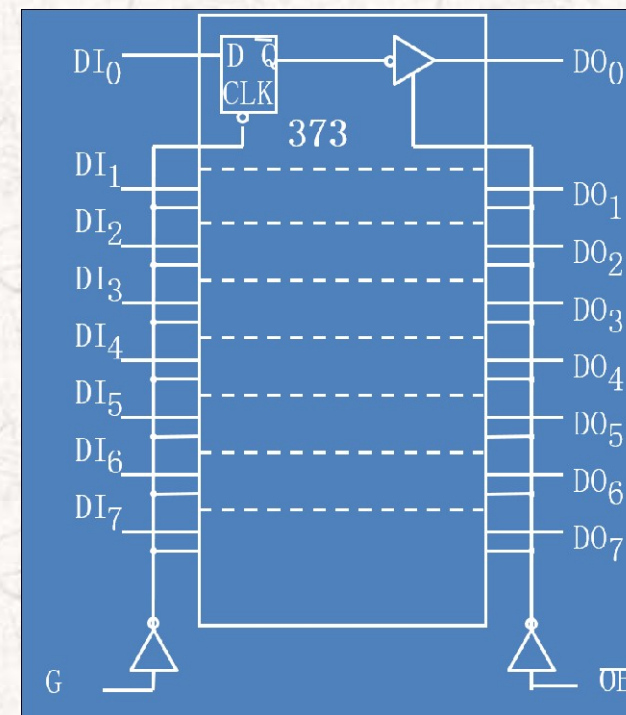
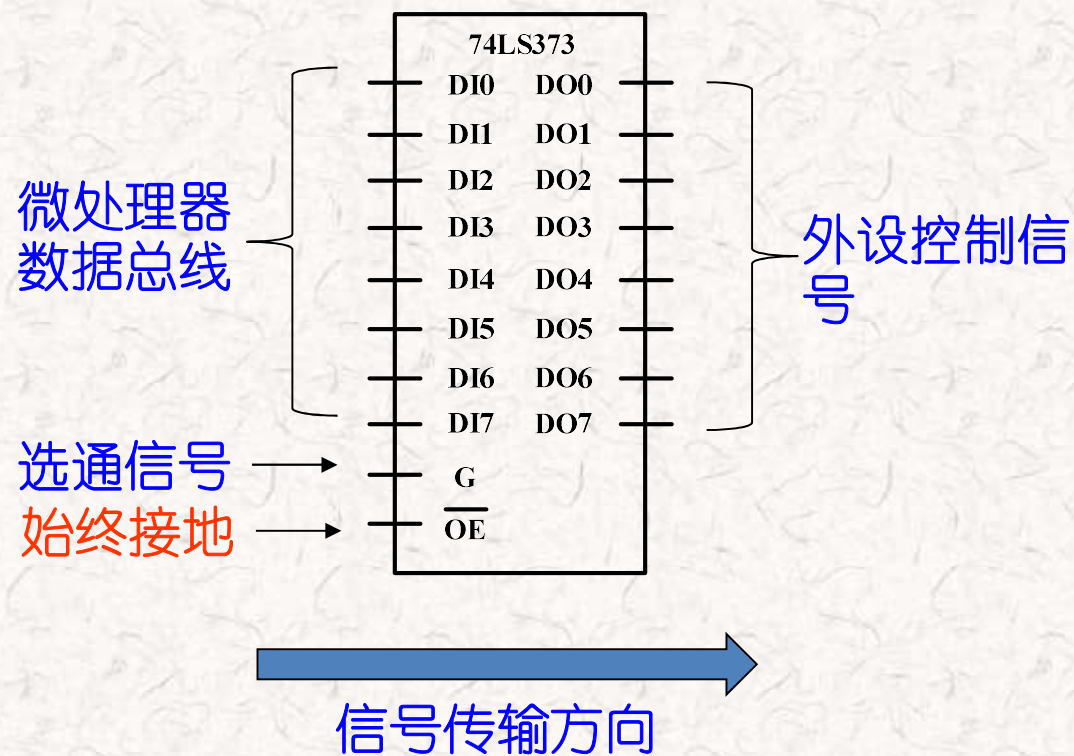


## 简单输出端口总结

- (1) 功能？(2) 用什么逻辑电路构造？如何接入系统？
- (3) 选通信号功能？(4) 选通信号实现逻辑？
- 输出端口用于将微处理器期望输出的数据锁存并保持，以便持续控制外设工作状态
- 采用锁存器（或触发器）实现，其输入为微处理器数据总线传递来的希望控制外设的命令，输出连接被控制外设，时钟端受选通信号控制
- 选通信号功能：当微处理器对该输出端口执行写操作时，选通信号有效，使该端口处于透明状态，接受微处理器传递来的数据，其它时间为锁存状态，输出保持不变
- 选通信号生成逻辑：地址总线输出地址=端口地址， $\overline{WR}$ =低电平， $M/\overline{IO}$ =低电平，三者同时有效



# 举例一如何用74LS373构造输出接口



# 温度控制电路连接练习

- 已经实例化了**244**，**373**和**138**器件；设计电路实现**80H**接口读取温度数据，**88H**接口输出控制信息；
- 需要按照要求完成电路连线；
- 运行代码，自动测试部分通过判断**80H**接口读取数据和**88H**接口输出数据是否正确判断电路设计的正确性
- 仿真结束**OK\_in**和**OK\_out**应为高电平

## 2.IO的一般特点与数据传送方式

- 还有哪些IO端口的例子？
- 与外设之间的无条件输入或者输出接口适用于什么情况？
- 简述查询式发送数据的过程，需要实现什么IO端口
- 简述查询式接收数据的过程，需要实现什么IO端口
- 简单说明中断方式进行数据传输优缺点是什么？什么情况下使用中断方式传输比较好？
- 什么是DMA？为什么DMA传输方式效率高？



# IO接口用于解决什么问题？

## 速度不匹配

**CPU**速度比外设的速度高很多，且不同外设速度差异甚大  
例如：打印机、扫描仪、音响等

## 信号电平不匹配

**CPU**都用**TTL**电平，而外设大多是复杂的机电设备，往往不能为**TTL**电平所驱动，有自己的电源系统和信号电平

例如：前述的制冷和加热控制设备

差分信号传输标准：**LVDS**，等



# IO接口用于解决什么问题？

## 信号格式不匹配

**CPU**传送的通常是**8位**、**16位**或**32位**并行数据，而外设使用的信息格式各不相同。

有模拟量、数字量或开关量；

有电流量、电压量；

有些采用串行方式，有些用并行方式

## 时序不匹配

外设都有各自的定时和控制逻辑，与**CPU**的时序不一致，如何与**CPU**协调工作？

# IO如何解决这些问题？

## 设置数据缓冲解决速度不匹配问题

事先把要传送的数据准备好，在需要的时刻完成传送

经常使用锁存器和缓冲器，并配以适当的握手交互信号来实现功能

## 设置电平转换电路解决电平不一致问题

如计算机和外设间进行串行通信时，可采用**MAX232**和**MAX422**等芯片来实现电平转换

## 设置信息转换逻辑满足各自格式要求

将外设传送的模拟量，经**A/D**转成数字量，送到**CPU**处理  
**CPU**送出的数字信号经**D/A**转为模拟信号，驱动外设

# IO如何解决这些问题？

## 设置时序控制电路同步CPU和外设的工作

接口电路接收CPU送来的命令或控制信号、定时信号，实施对外设的控制与管理，外设的工作状态和应答信号也通过接口及时返回CPU，以握手联络(handshaking)信号来保证主机和外部I/O操作实现同步

## 提供地址译码电路

计算机中存在多个外设，包含若干接口，其I/O地址译码电路用于CPU分别访问各个端口而不会发生冲突

## 中断控制等逻辑

实现外设与CPU的一种高实时的响应机制



# 化繁为简

- 外设都是通过缓冲器（输入）和锁存器（输出）挂接在CPU总线上，实现与CPU的数据、状态和控制命令交互的



# CPU与外设的数据传输方式

- 程序控制方式
  - 无条件传输
  - 查询传输
- 中断传输
- DMA传输 (DMA: **D**irect **M**emory **A**ccess)

# 程序控制之无条件传输的应用

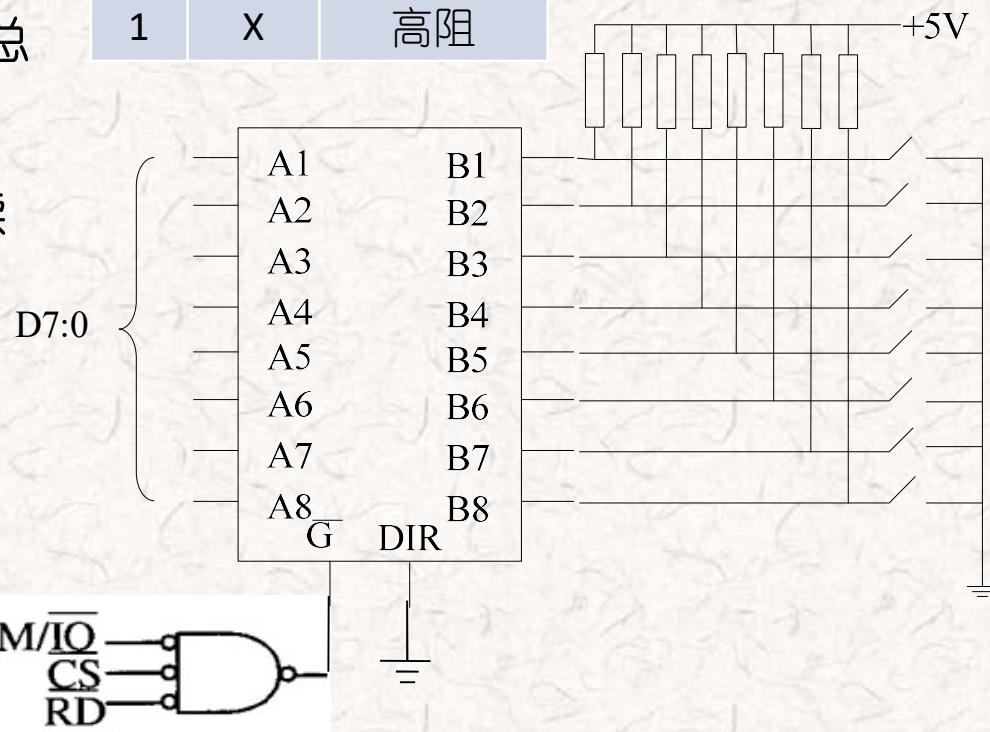
- 适用于简单外设
- 输出端口通常具有锁存功能，当CPU选中该端口输出数据时，端口接收并保持来自数据总线的数据
- 无条件输出端口：随时可以接收CPU设置的数据
- 当选通输入端口时，相应的数据或者状态出现在数据总线上，以便被CPU读取
- 无条件输入端口：输入数据随时准备好，可供CPU读取

## 练习：使用74LS245构成检测开关量的无条件输入接口并编程

设/ $\overline{CS}$ 为A7:0=82H的译码结果，且74LS245的端口A1~A8分别连接到数据总线D7~D0，请编程实现以下功能：读取B6连接的开关状态，若开关**闭合**，则跳转到标号“XXX”执行，否则继续向下执行

```
IN AL,82H
TEST AL, 04H
JZ XXX
```

$\overline{G}$	DIR	功能
0	0	B->A
0	1	A->B
1	X	高阻





## 练习：使用74LS373构成LED输出接口

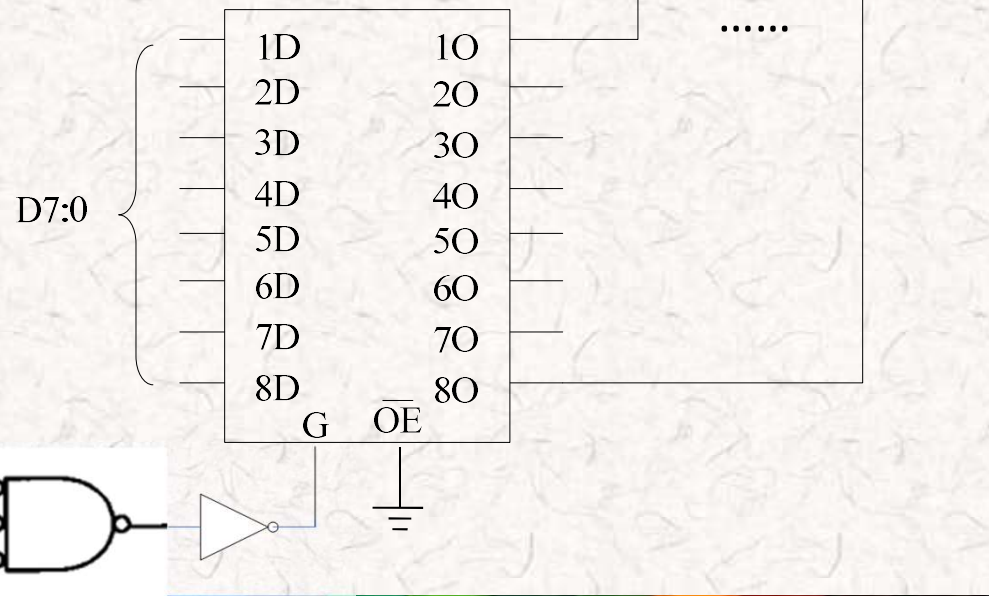
设计接8盏LED灯的简单输出接口，并编程控制LED灯全亮

如何使LED全部点亮：

```
MOV AL, 00H  
OUT 80H, AL
```

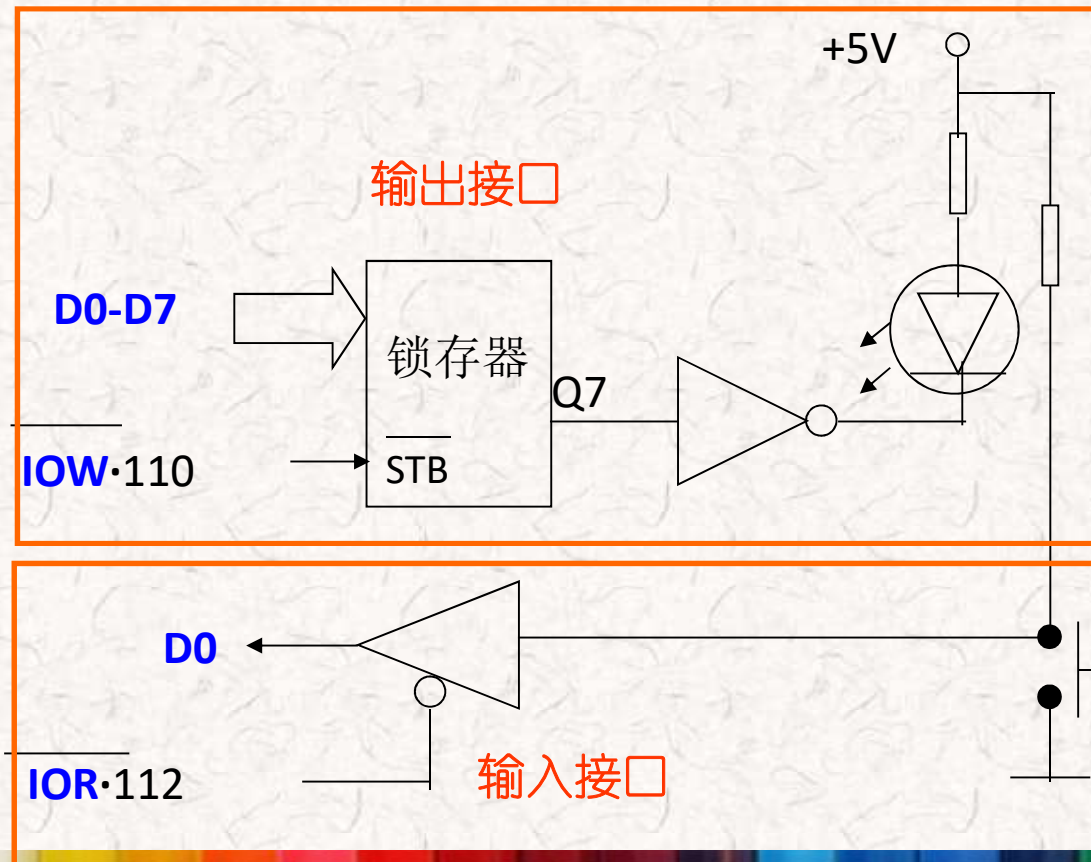
设/CS为A7:0=80H的译码结果

G	OE	功能
1	0	D->O
0	0	O保持
X	1	高阻



# 理解电路并编程

编程：8086 CPU读入按钮的状态，若按钮按下，LED亮，未按下则灭，循环执行



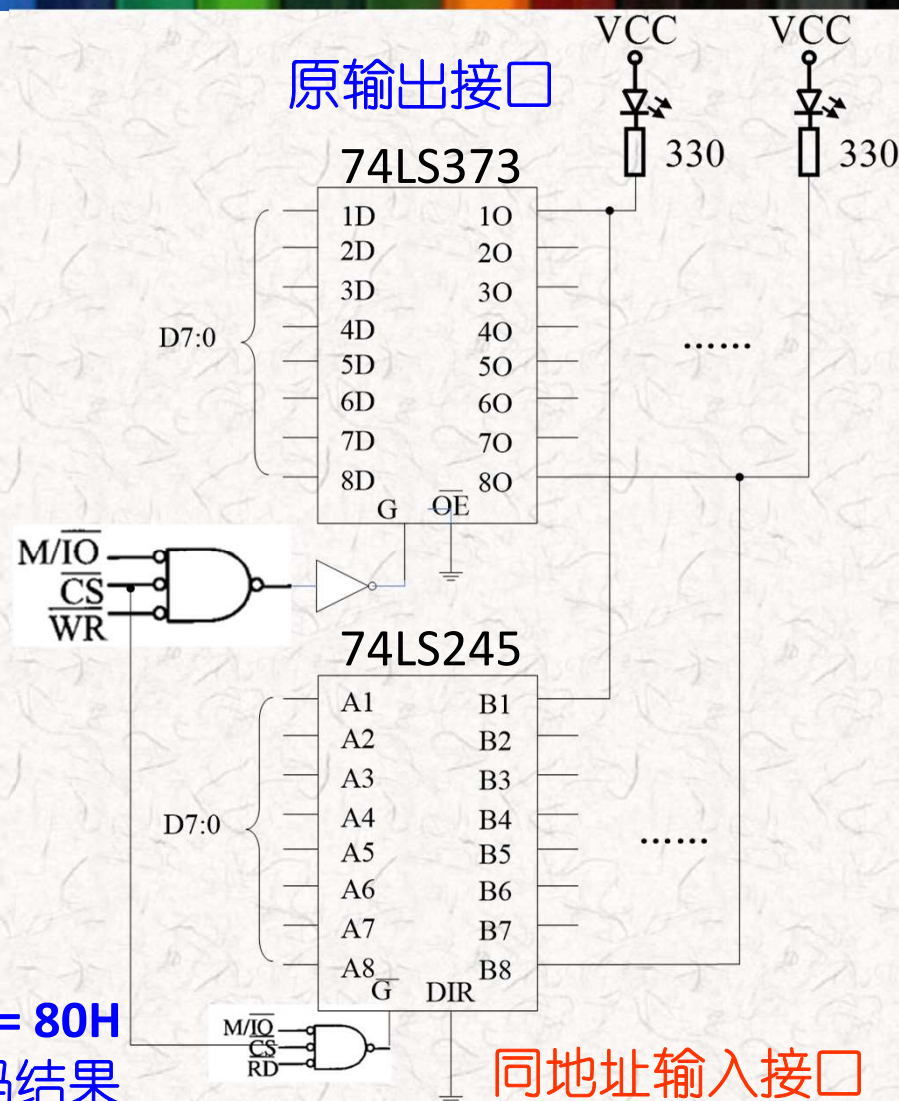
代码:

```
STA: IN  AL, 112
      AND AL, 01H
      (TEST AI, 01H亦可)
      JZ  OPE
      MOV AL, 00H
      OUT 110, AL
      JMP STA
OPE: MOV AL, 80H
      OUT 110, AL
      JMP STA
```

# 可读写的端口设计

- 如：读取某端口的值，将其中D0位清零后再输出
- **IN AL,80H**
- **AND AL,0FEH** ;清D0
- **(OR AL, 01H)** ;置D0
- **OUT 80H,AL**
- 读回输出的数据需要硬件支持

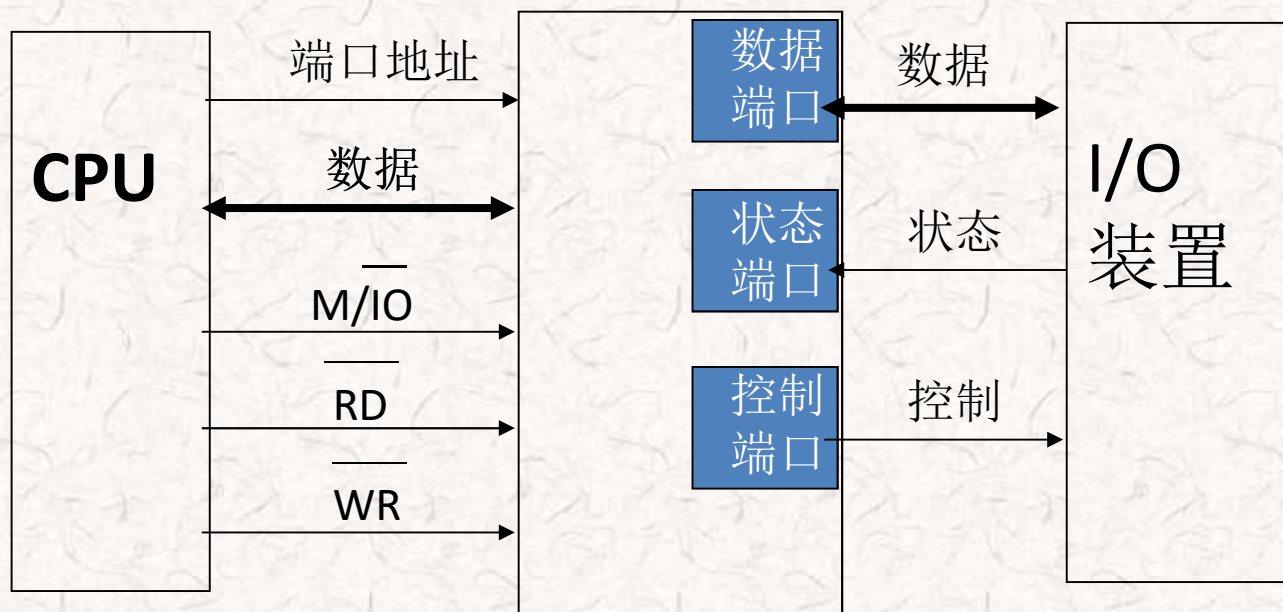
/CS是A7:0= 80H  
的地址译码结果





# 条件传送方式

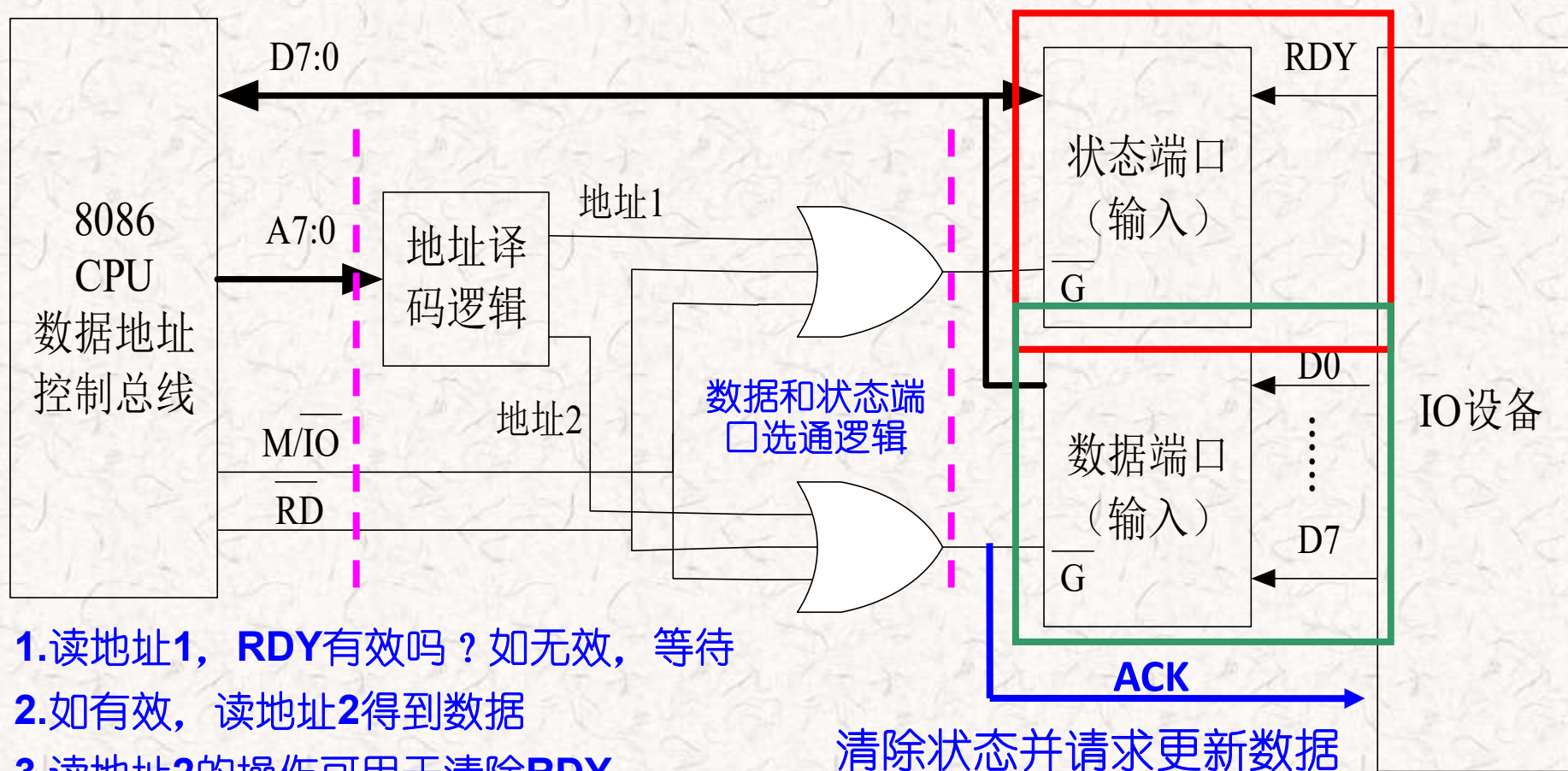
- 握手的概念：查询--通信--应答
- 硬件 - 信号，状态



# 常见的状态和应答信号

- 输入端口
  - 数据准备好：输入端口有外设数据可供**CPU**读取
  - 数据已读取：表明**CPU**已读取数据，用于清除数据准备好，提示外设可以发送新数据
- 输出端口
  - 设备就绪：输出端口准备好，可以接收**CPU**写入数据
  - 新数据写入：**CPU**向输出端口写入数据，用于清除设备就绪，并提示外设进行处理
- 错误：外设出错，具体错误类型不同，**CPU**的处理也不同

# 查询式输入端口的形式和操作

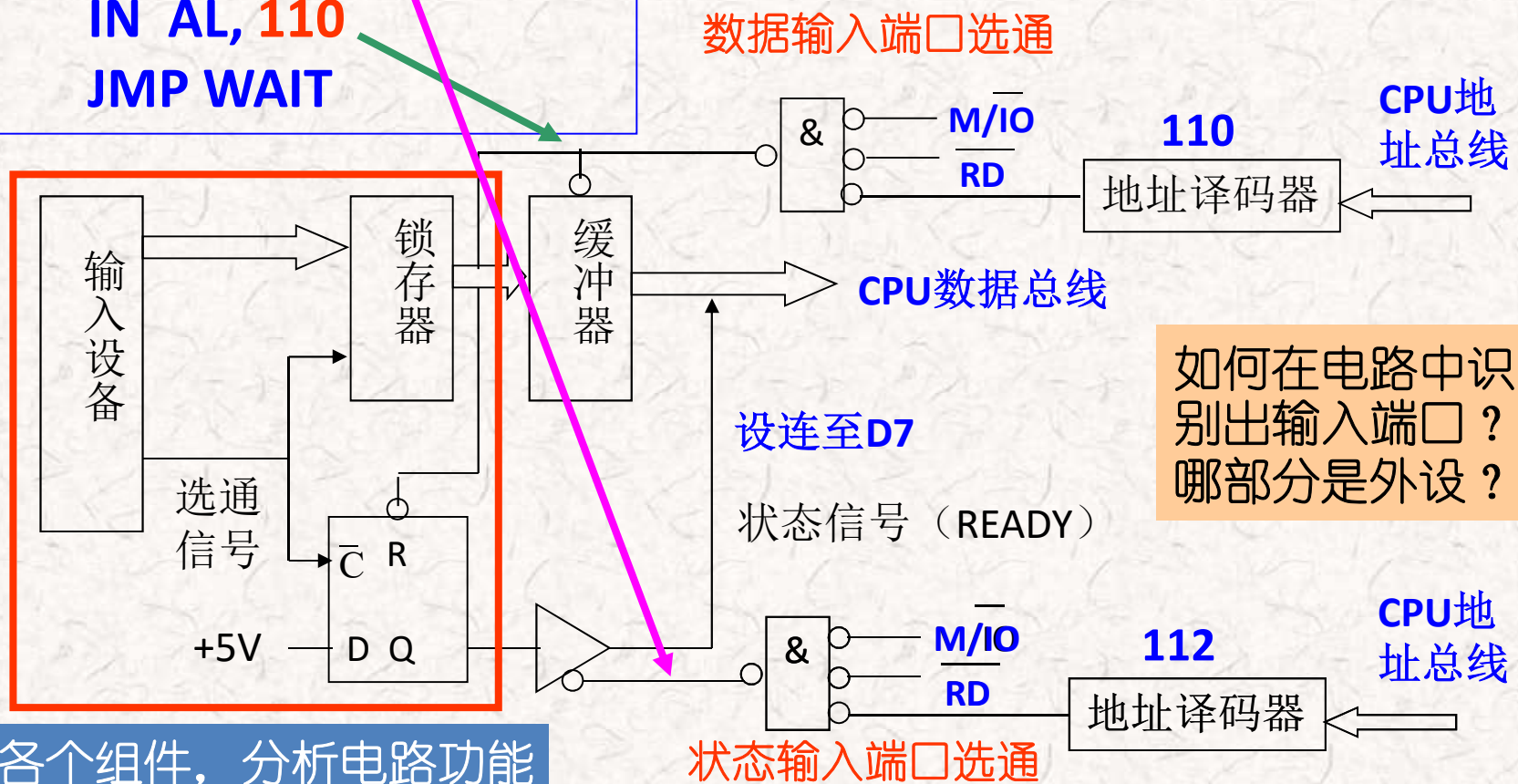




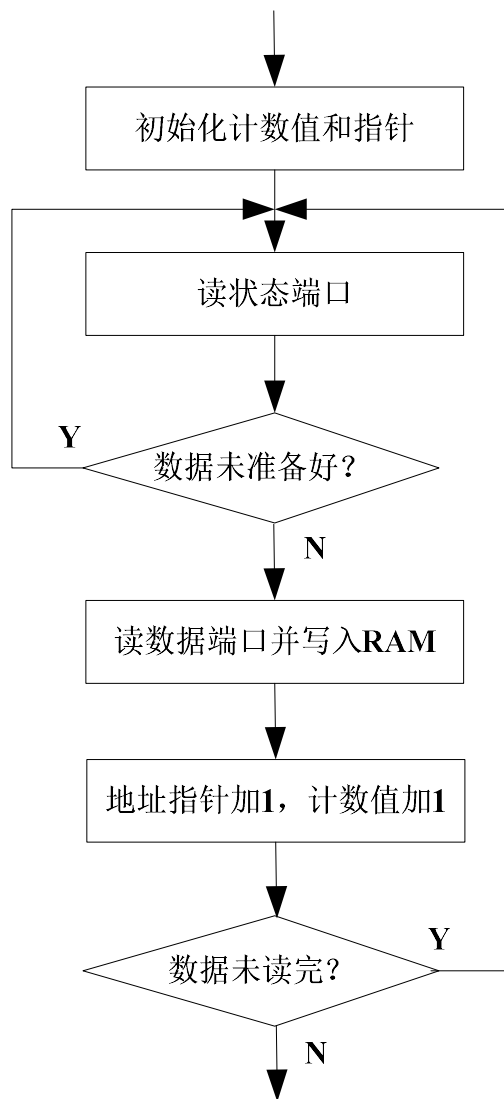
```

WAIT: IN AL, 112
      TEST AL, 80H
      JZ WAIT
      IN AL, 110
      JMP WAIT
  
```

## 查询式输入电路与编程



认识各个组件，分析电路功能



## 查询式输入一组数据

**MOV BX, 0** ;初始化地址指针

**MOV CX, COUNT\_1** ;传送字节数

**READ\_S1:**

**IN AL, PORT\_S1** ;读入状态位

**TEST AL, 01H** ;数据准备好?

**JZ READ\_S1** ;否, 循环检测

**IN AL, PORT\_IN** ;准备好, 读数据

**MOV [BX], AL** ;存入内存缓冲区

**INC BX** ;修改地址指针

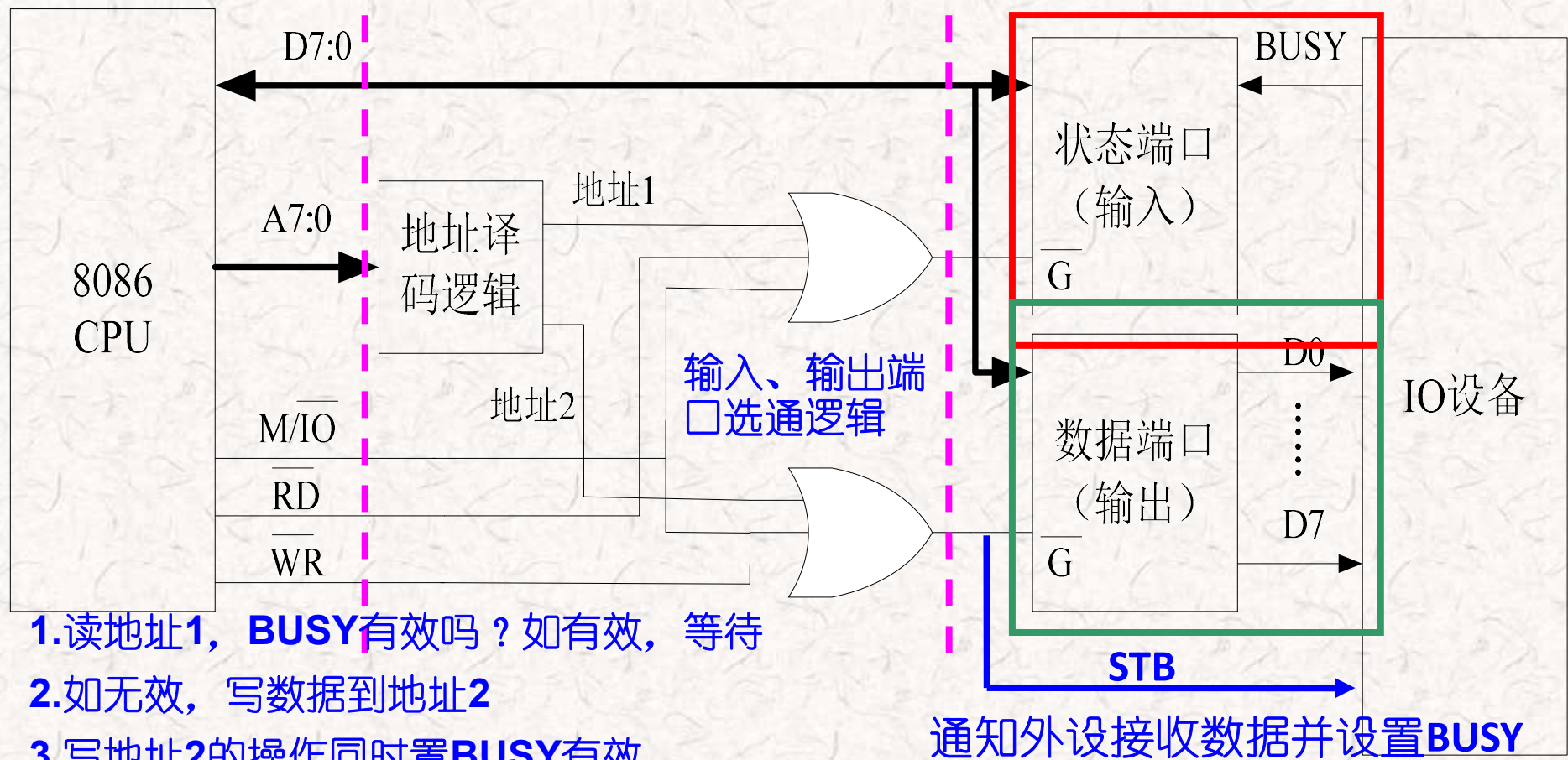
**DEC CX**

**JNZ READ\_S1** ;未完, 继续

...

;已传送完

# 查询式输出端口的形式和操作



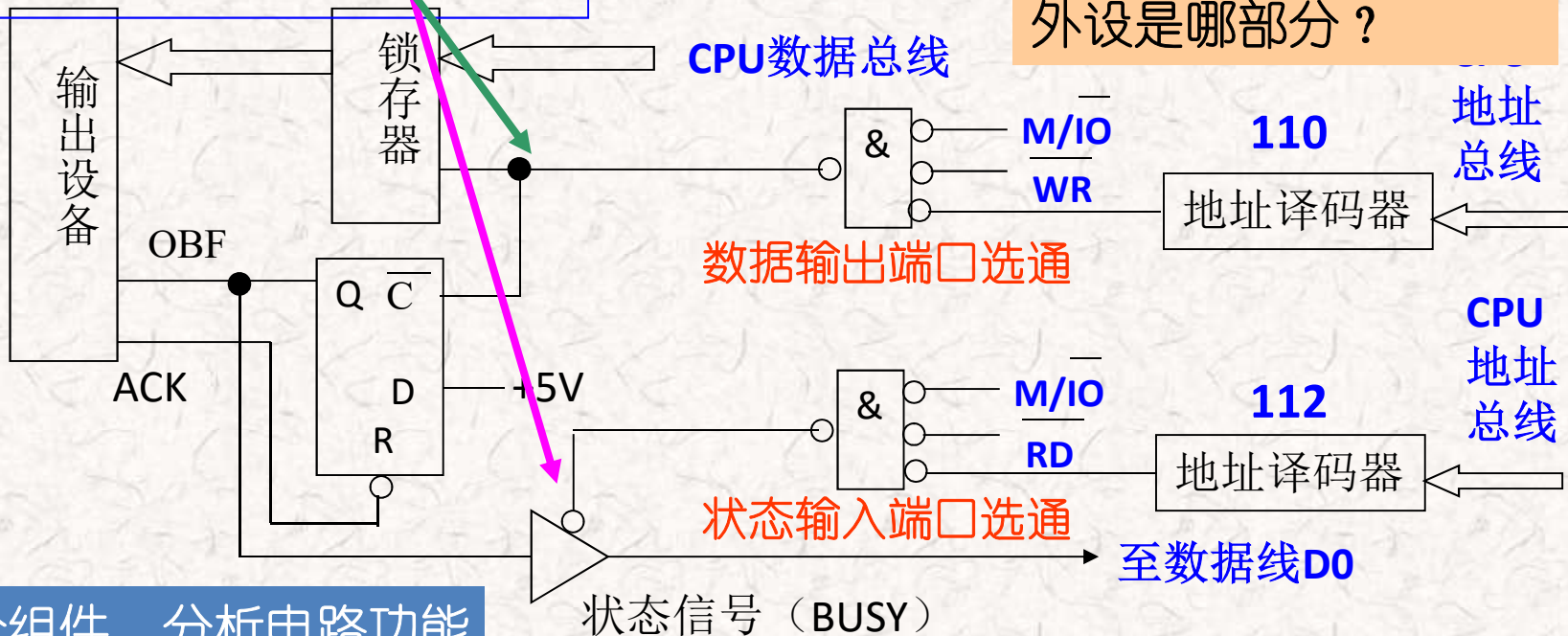


```

WAIT: IN AL, 112
      TEST AL, 01H
      JNZ WAIT
      MOV AL, BL
      OUT 110, AL
      JMP WAIT
  
```

## 查询式输出电路 与编程

如何在电路中识别出  
输入/输出端口？  
外设是哪部分？



认识各个组件，分析电路功能

# 查询式输出一组数据

查询式传输

优点

接口简单

缺点

- 1 浪费CPU时间
- 2 外设请求可能没有及时响应

```
MOV    CX, COUNT_2    ;传送的字节数
READ_S2:
IN      AL, PORT_S2    ;读入状态位
TEST    AL, 02H        ;忙否?
JNZ     READ_S2        ;忙，循环检测
MOV     AL, 待输出数据 ;不忙
OUT     PORT_OUT, AL    ;输出数据
DEC     CX
JNZ     READ_S2        ;未传送完，循环
...
;已传送完
```

# 中断传送方式-提高CPU利用率

过程：

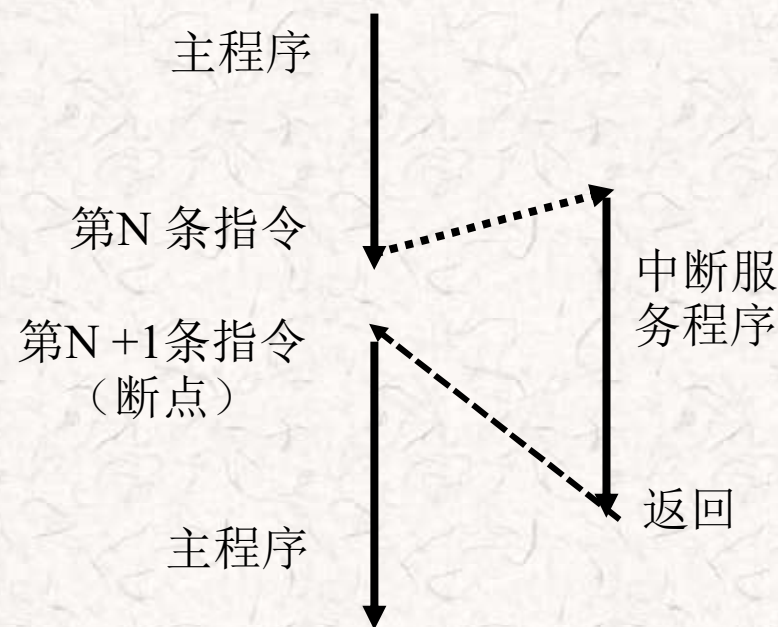
I/O端口就绪则发起中断请求，CPU在中断服务程序中读取或者输出数据

优点：

- (1) 实时性高
- (2) 避免反复查询浪费CPU时间

缺点：

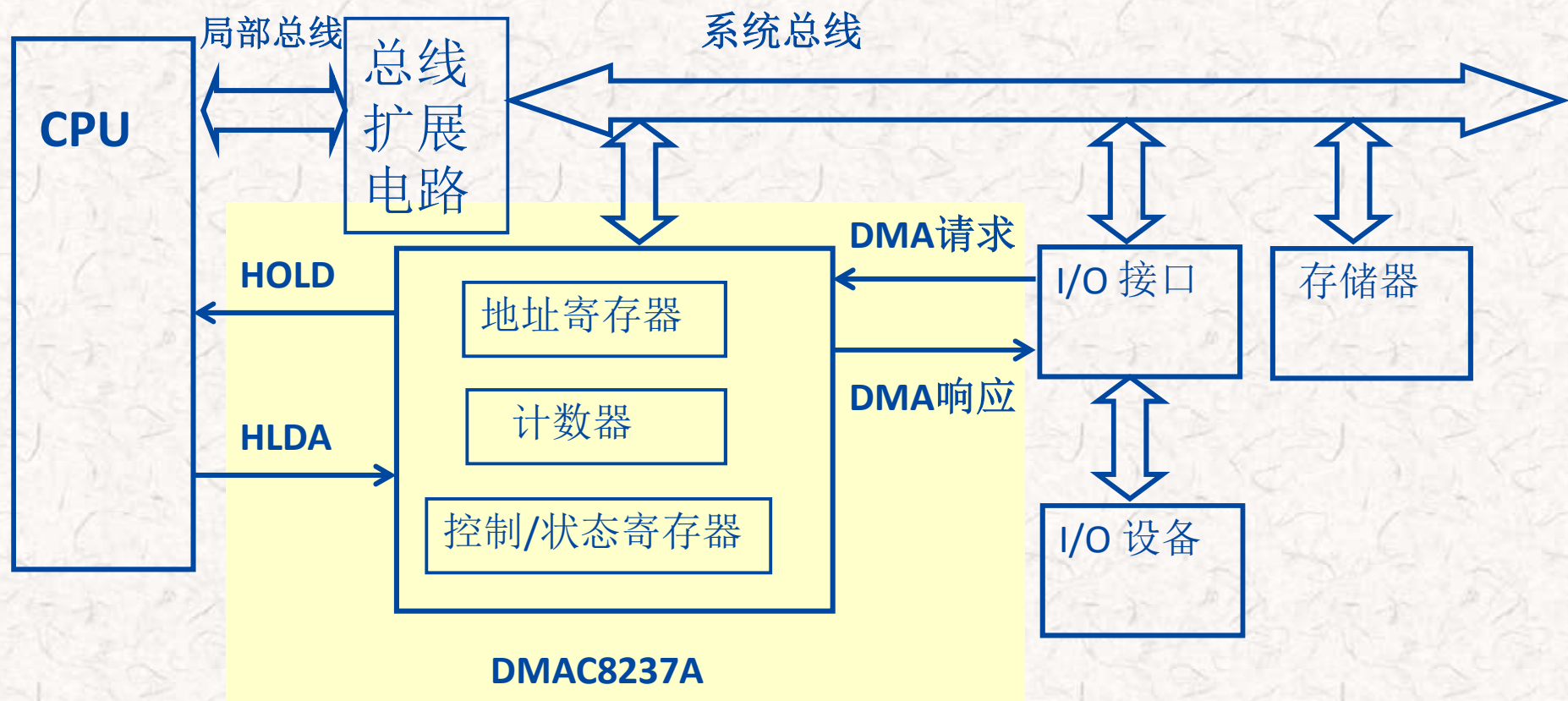
批量传输每次都中断则时间开销大  
(相比DMA而言)



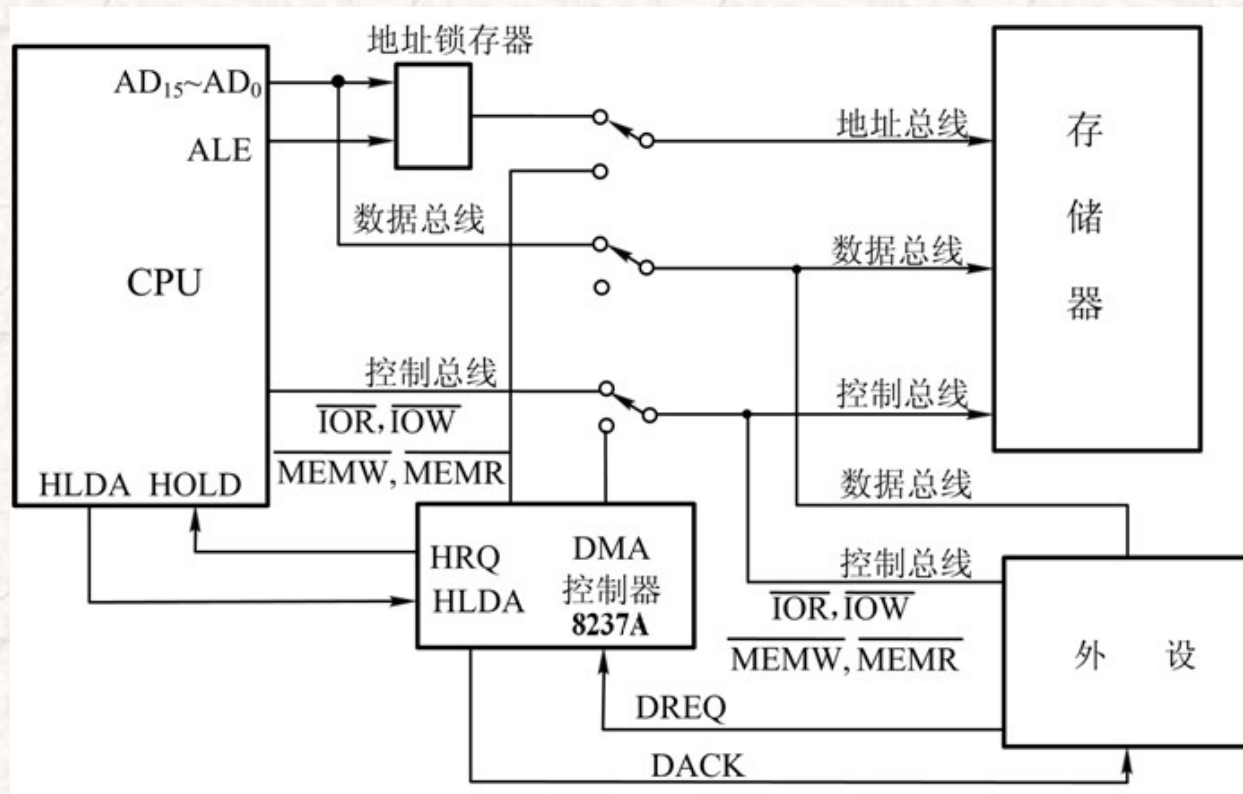
保存断点、获取中断类型号，获取中断向量，保存现场，**执行**，恢复现场，恢复断点



# DMA(Direct Memory Access)传输方式



# DMA对总线的控制



**DMA获取总线控制权后：**

- 生成访问内存所需的地址信息，并在每次传输后自动修改地址指针
- 生成访问外设和存储器的读写控制信号
- 传输次数计数，直到等于设定值

# DMA传送过程

主程序中完成**DMA**模块配置，包括：

源地址

目的地址

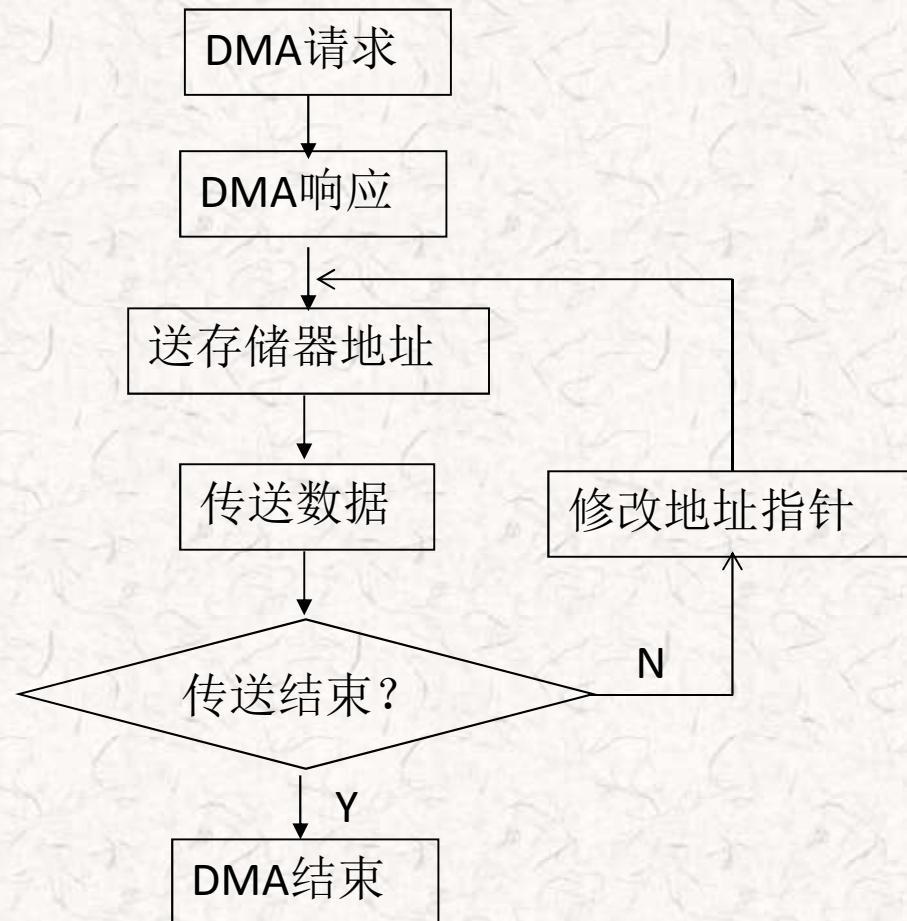
传送的字节数

地址修改方式

启动触发方式

.....，等

然后等待**DMA**启动信号来到





# DMA传输的特点

优点:

- (1) 批量数据传输效率高
- (2) 节省CPU时间

缺点:

硬件自动修改指针、计数值、产生读写信号, 电路复杂度高

举例: 通过指令实现数据块搬移

MOV DI, OFFSET SOUCE 4

MOV SI, OFFSET DESTIN 4

MOV CX,200 4

REPEAT: MOV AL, [DI]; 进行总线操作 8

MOV [SI],AL; 进行总线操作 8

INC DI 2

INC SI 2

DEC CX 2

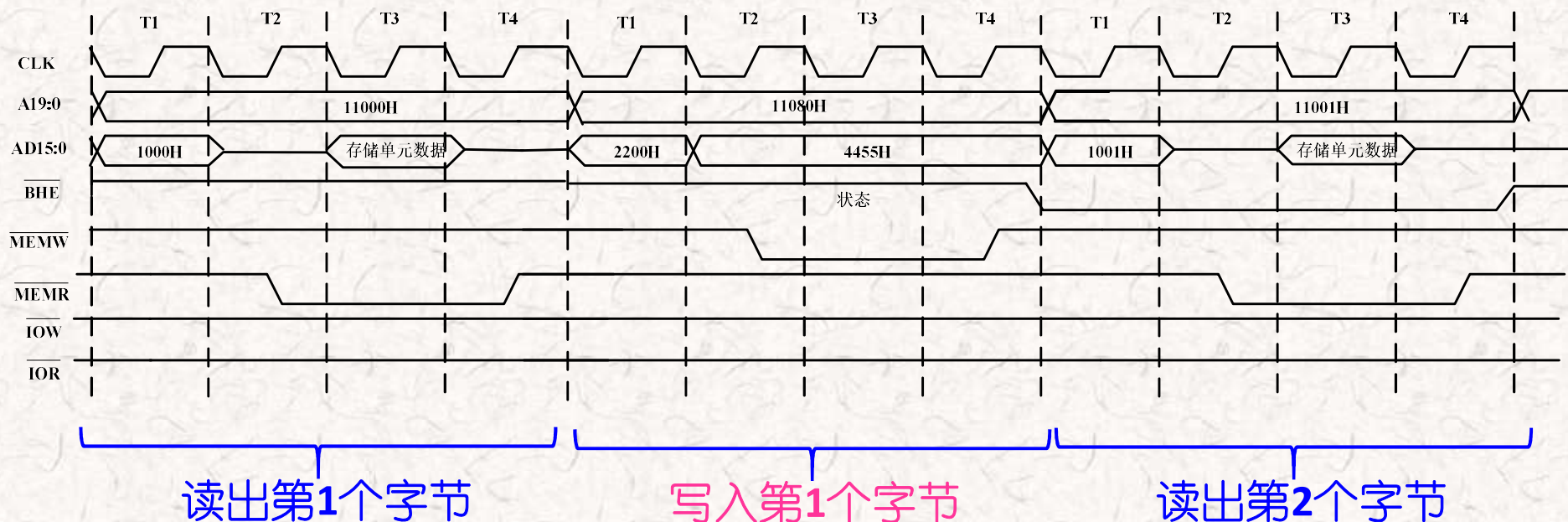
JNZ REPEAT 16

典型指令的执行时间  
(时钟周期) [5]

指令	register- register	register immediate	register- memory	memory- register	memory- immediate
mov	2	4	8+EA	9+EA	10+EA
ALU	3	4	9+EA,	16+EA,	17+EA
jump	register => 11 ; label => 15 ; condition, label => 16				
整数 乘法	70~160 (取决于操作数 data 以及大小) 加上EA				
有符 号整 数除 法	80~190 (取决于操作数 data 以及大小) 加上EA				

传输200个字节的所需时钟周期数:  
(8+8+2+2+2+16)\*200=7600

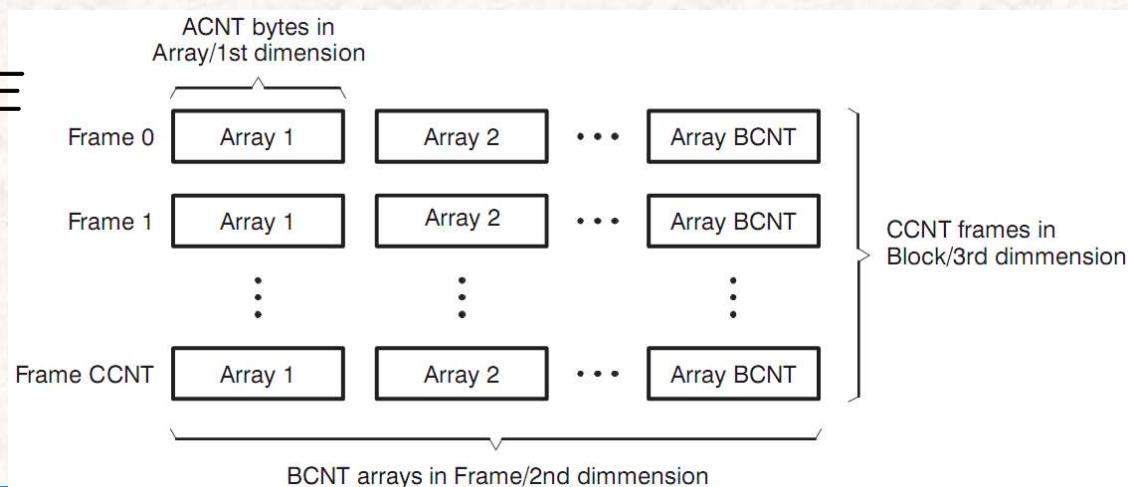
# DMA传输-提高传输效率



DMA传输开销:  $(4+4)*200=1600$ , 传输200个字节所需总周期数减少79%  
传输过程无需CPU参与

# DMAC的今生--EDMA

- **EDMA**是高性能微处理器必备模块
- 具有多个队列并且可以同时维持多个传输通道
- 支持二维、三维数据传输
- 不同传输事件可以相互关联，传输完成可以触发中断
- .....
- **CPU**可与**DMA**并行工作



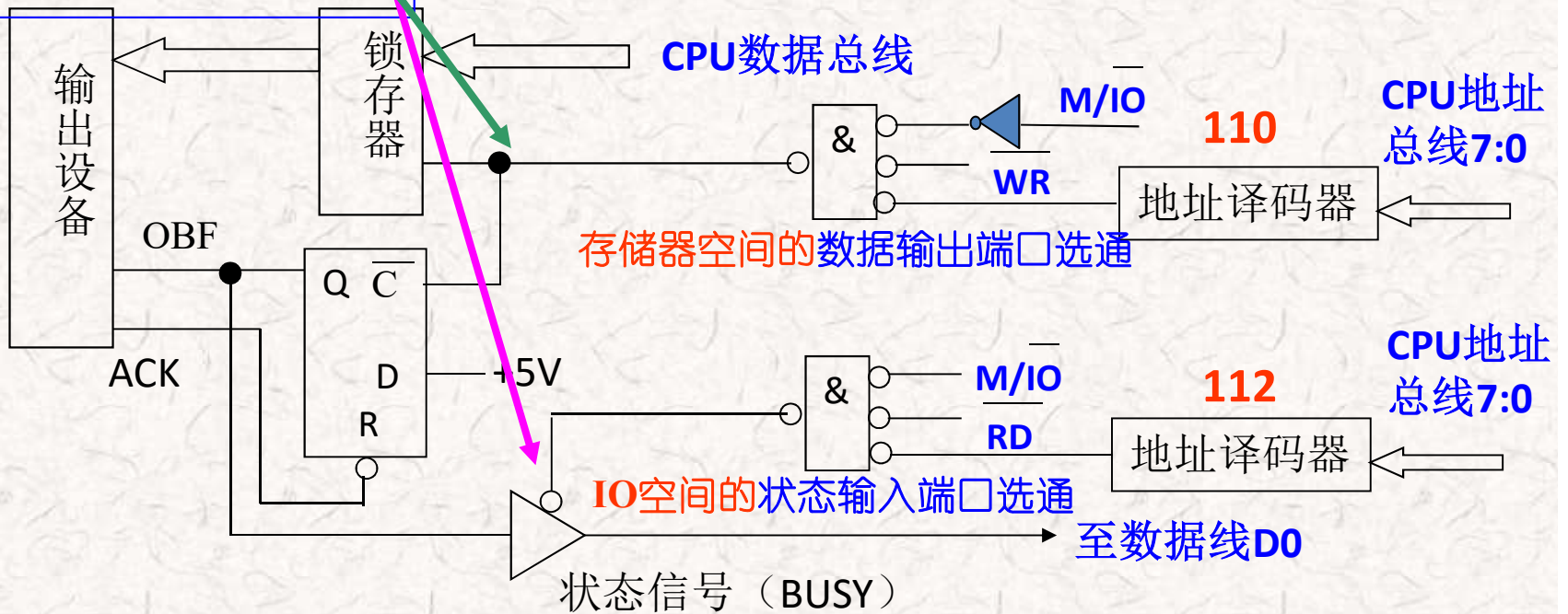


```

WAIT: IN AL, 112
      TEST AL, 01H
      JNZ WAIT
      MOV AL, BL
      MOV [110], AL
      JMP WAIT
  
```

## 关于接口地址译码的进一步讨论—分析电路并编程

查询式输出接口



# IO端口分配在8086不同地址空间的比较

- IO空间译码

- $\overline{M}/\overline{IO}$ 为低电平
- 最多16根地址线可用, 64KB地址空间
- 只有IO访问指令可用
- 输入指令: `IN AL, 20H;` `MOV DX, 220H;` `IN AL, DX`
- 输出指令: `OUT 80H, AL;` `MOV DX, 1A0H;` `OUT DX, AL`

- 存储器空间译码

- $\overline{M}/\overline{IO}$ 为高电平
- 最多20根地址线可用, 1MB地址空间
- 存储器访问指令均可使用, 寻址方式丰富
- 直接寻址, 寄存器间接寻址, 寄存器相对寻址, 基址变址寻址, 相对基址变址寻址

## 3.8255可编程并行接口

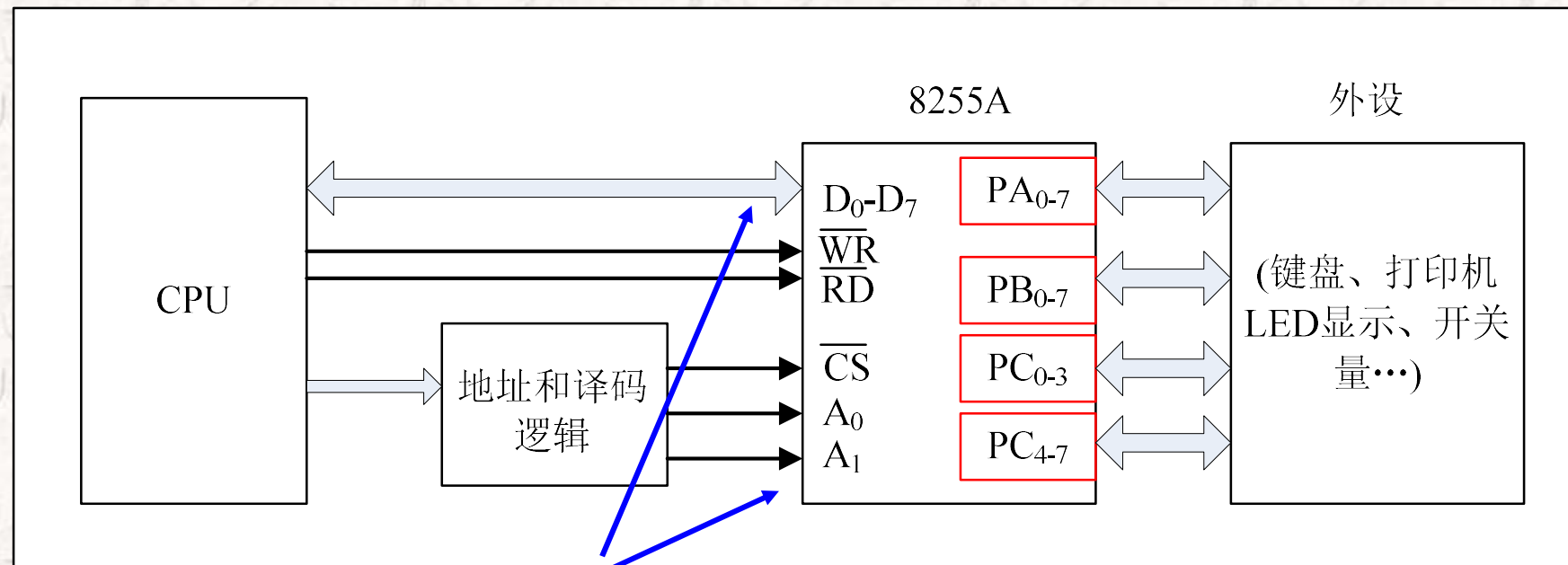
- 说明对8255控制口进行写操作时施加在8255引脚上的信号是怎样的
- 若为8255分配的基地址为20H（设未使用的低位地址为0），D7:0接数据总线D15:8，则8255各端口地址可以设计为：
- (1) 20H, 22H, 24H, 26H
- (2) 21H, 23H, 25H, 27H
- (3) 22H, 24H, 26H, 28H
- (4) 23H, 25H, 27H, 29H

学习课本 第6章6.2节



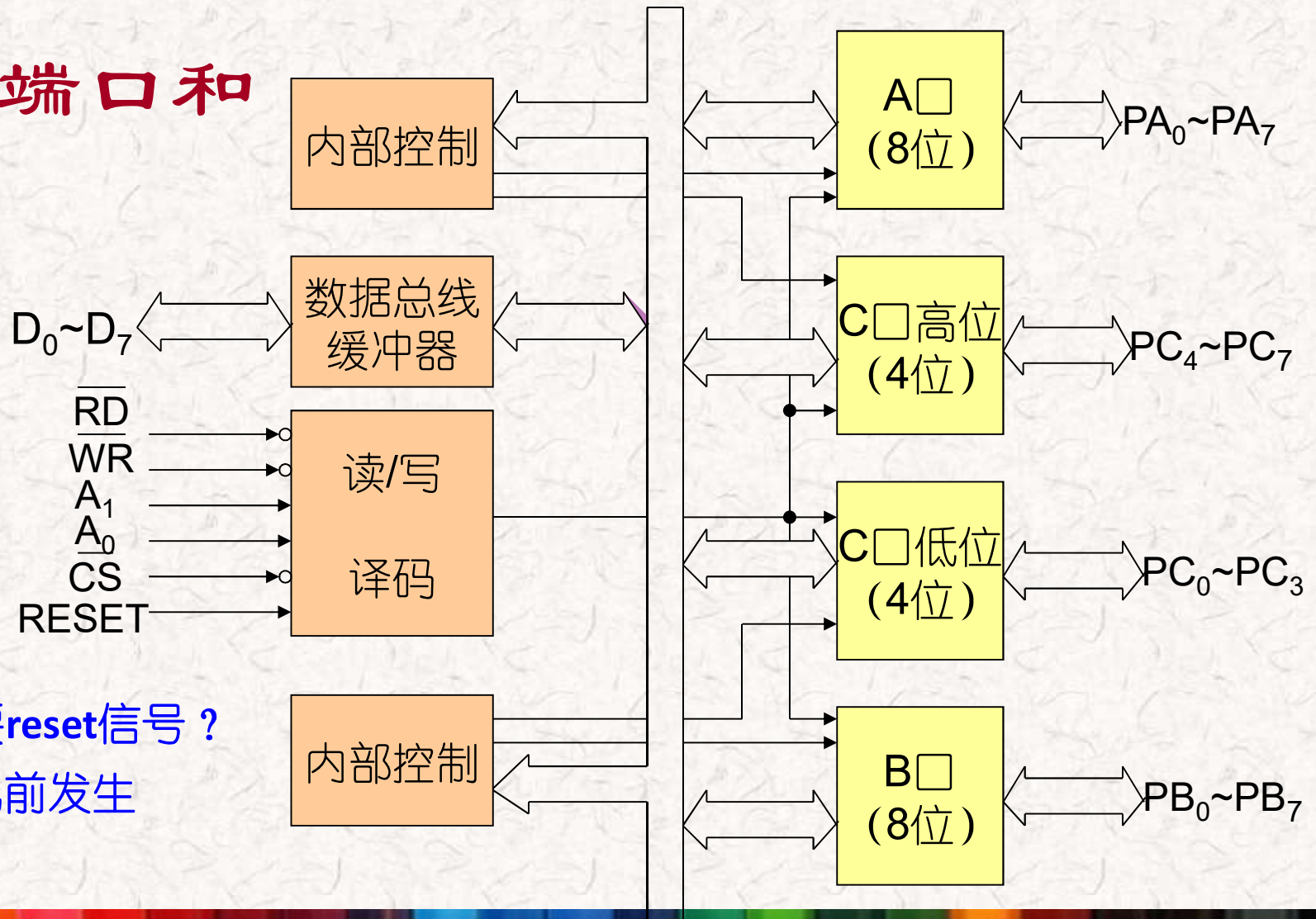
# 8255在微处理器系统中的应用

## 可编程并行接口



**2<sup>2</sup>\*8bit 占据4个地址**

# 8255端口和结构



为什么需要reset信号？  
避免初始化前发生  
输出冲突

# 8255端口操作

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	$A_1 A_0$
0	1	0	0 0
0	1	0	0 1
0	1	0	1 0
0	1	0	1 1
0	0	1	0 0
0	0	1	0 1
0	0	1	1 0
0	0	1	1 1
1	×	×	×
0	1	1	×

## 功 能

D7:0写入端口A (数据输出)

D7:0写入端口B (数据输出)

D7:0写入端口C (数据输出)

D7:0写控制端口 (控制命令字)

读端口A的数据到D7:0 (数据输入)

读端口B的数据到D7:0 (数据输入)

读端口A的数据到D7:0 (数据输入)

无操作 (控制寄存器不支持读)

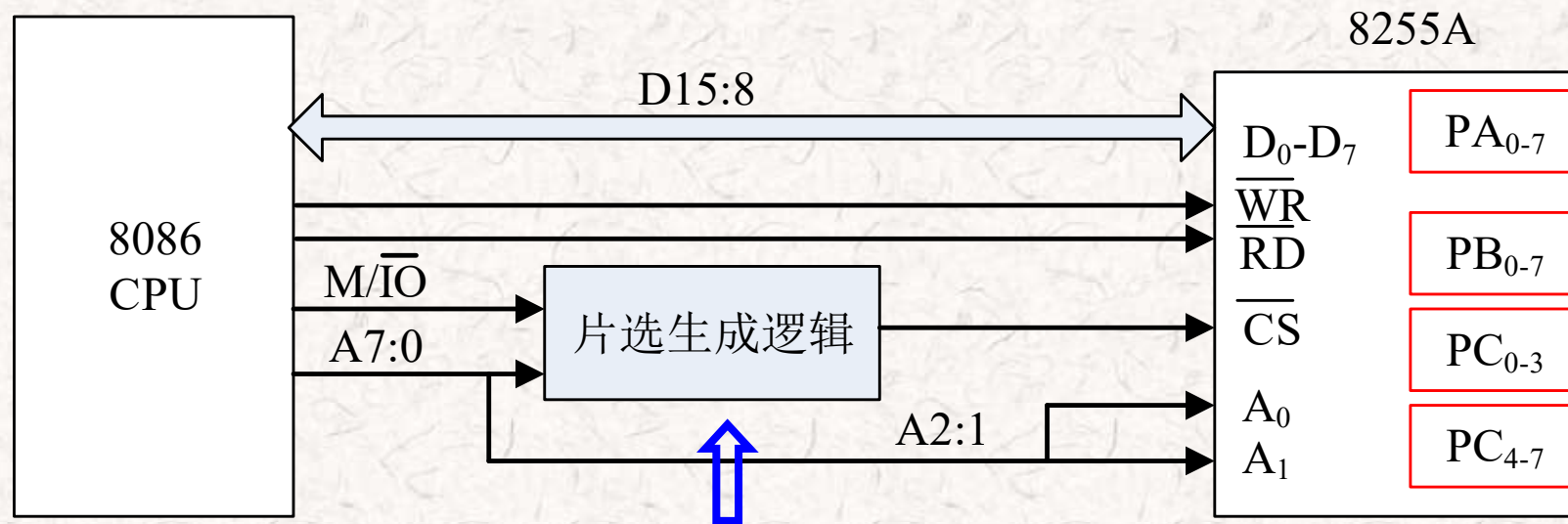
未选通本器件

无操作



# 8255与8086 CPU的连接

若为**8255**各端口分配地址为**21H,23H,25H,27H**，如何实现？

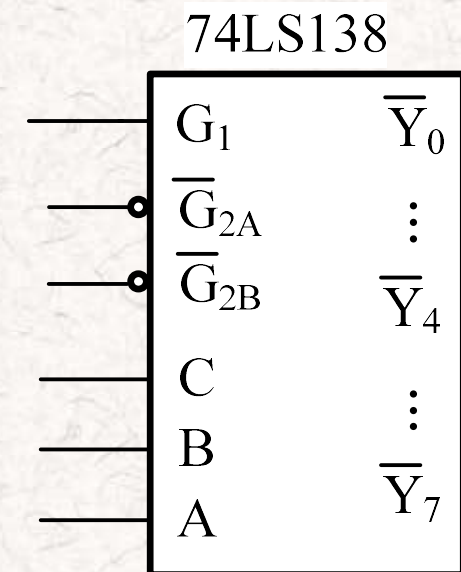


$A7:3=00100B$  &  $M/\overline{IO}=0$  &  $A0=1$

如果直接将8255的A0、A1连接到8086的A0、A1，需要考虑什么？

## 8255与8086的连接练习

- 若8255数据端  $\square \leftrightarrow$  8086数据总线 **D7:0**
- 8255地址线 **A1:0**  $\leftarrow$  8086地址总线 **A2:1**
- 8255片选连接 **/Y7**
- 则8086的**A0**信号可以接入以下哪个信号？
- (1)  $A0 \rightarrow A$
- (2)  $A0 \rightarrow G1$
- (3)  $A0 \rightarrow /G2A$  ✓
- (4)  $A0 \rightarrow B$



# 8255工作方式

- 三种工作方式
- 方式0：基本输入输出方式（输出锁存，输入缓冲）
  - A□，B□，C□均支持，其中C□可按位控制
- 方式1：选通输入/输出方式（查询式输入输出）
  - A□、B□支持，C□作为握手信号
- 方式2：双向数据传输
  - 只有A□支持，C□作为握手信号，B□工作在方式0



## 8255方式设置

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
特征位 = 1	<b>A口方式</b> 00=方式0 01=方式1 10=方式2 11=不用		<b>PA</b> 0=输出 1=输入	<b>PC<sub>4~7</sub></b> 0=输出 1=输入	<b>B口方式</b> 0=方式0 1=方式1	<b>PB</b> 0=输出 1=输入	<b>PC<sub>0~3</sub></b> 0=输出 1=输入

要把A口指定为方式1输入，C口上半部为输出；B口指定为方式0输出，C口下半部为输入，写初始化代码。设控制口地址为**306H**

工作方式命令代码是：**10110001B或0B1H**

```
MOV DX, 306H      ; 8255A命令口地址
MOV AL, 0B1H      ; 初始化命令
OUT DX, AL         ; 送到命令口
```

## 8255 C口按位输出控制

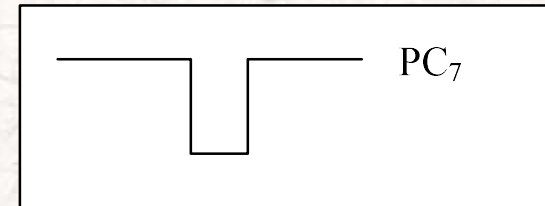
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
特征位 = 0	不用 (写0)			位 选 择 000=C□0位 001=C□1位 ... 111=C□7位			1=置位 (高电平) 0=复位 (低电平)

方式0下C□可以进行按位操作

其它方式下C□未使用的引脚也可以按位操作

## 编程在C口生成负脉冲举例

- 如利用8255A的PC7作打印机接口电路的数据选通信号，请编程在该引脚产生一个负脉冲，设8255端口地址600H~606H

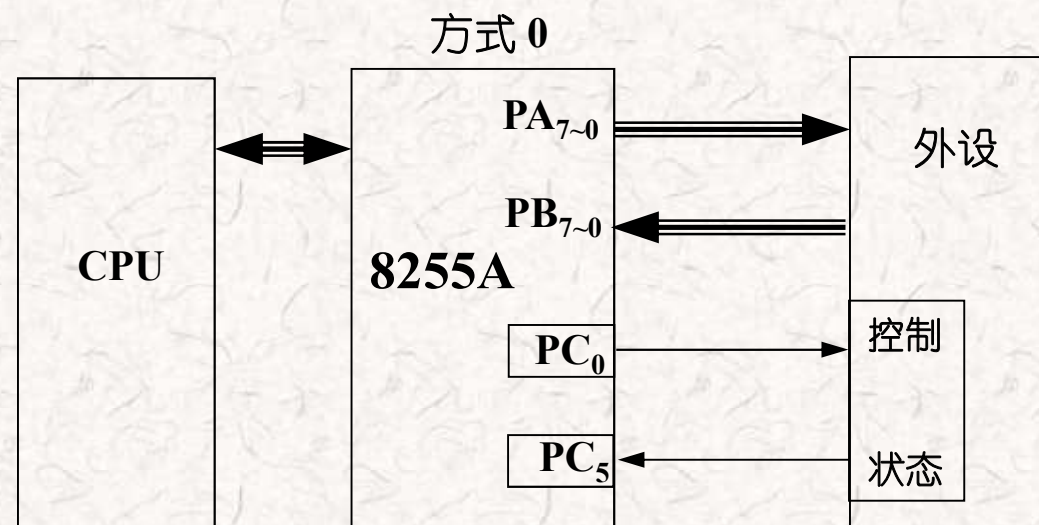


```
MOV DX, 606H           ; 8255A命令口
MOV AL, 00001110B      ; 置PC7=0, 输出低电平
OUT DX, AL
NOP                     ; 空操作
NOP                     ; 延时, 维持低电平
MOV AL, 00001111B      ; 置PC7=1, 输出高电平
OUT DX, AL
```



## 方式0的特点

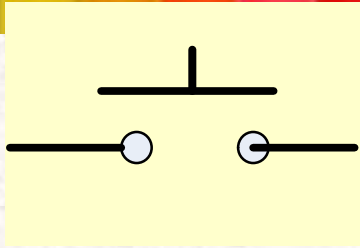
- 基本输入或输出，输入缓冲，输出锁存
- 适用于不需要应答的简单输入输出
- 延伸考虑：可以在方式0下实现有应答的输入输出么？
- 注意：C口只有在方式0下才可自由设置



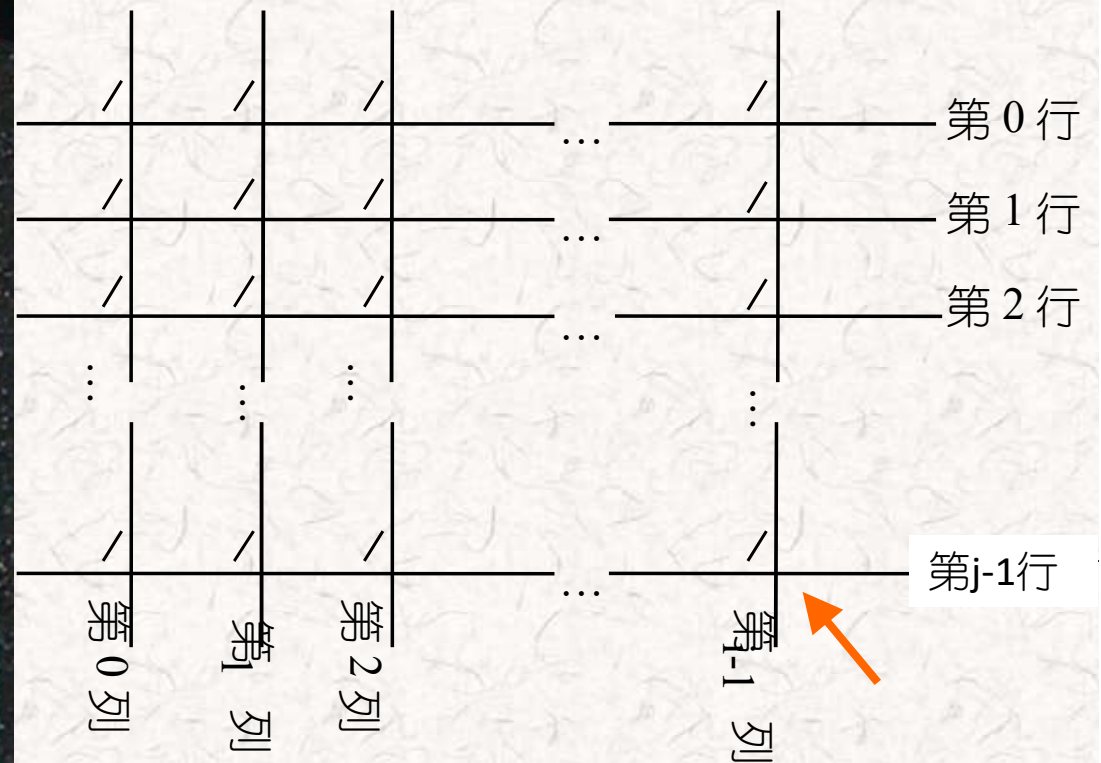
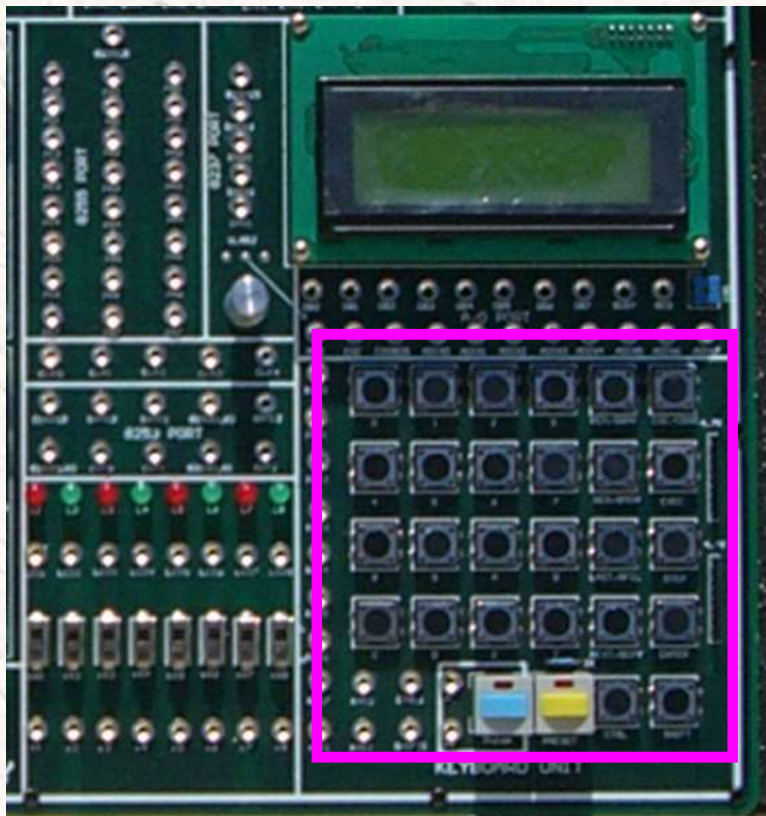
## 3.8255应用练习

- 画出P225键盘扫描程序流程图

学习课本 第六章6.3.2



## 并口典型应用--矩阵式键盘



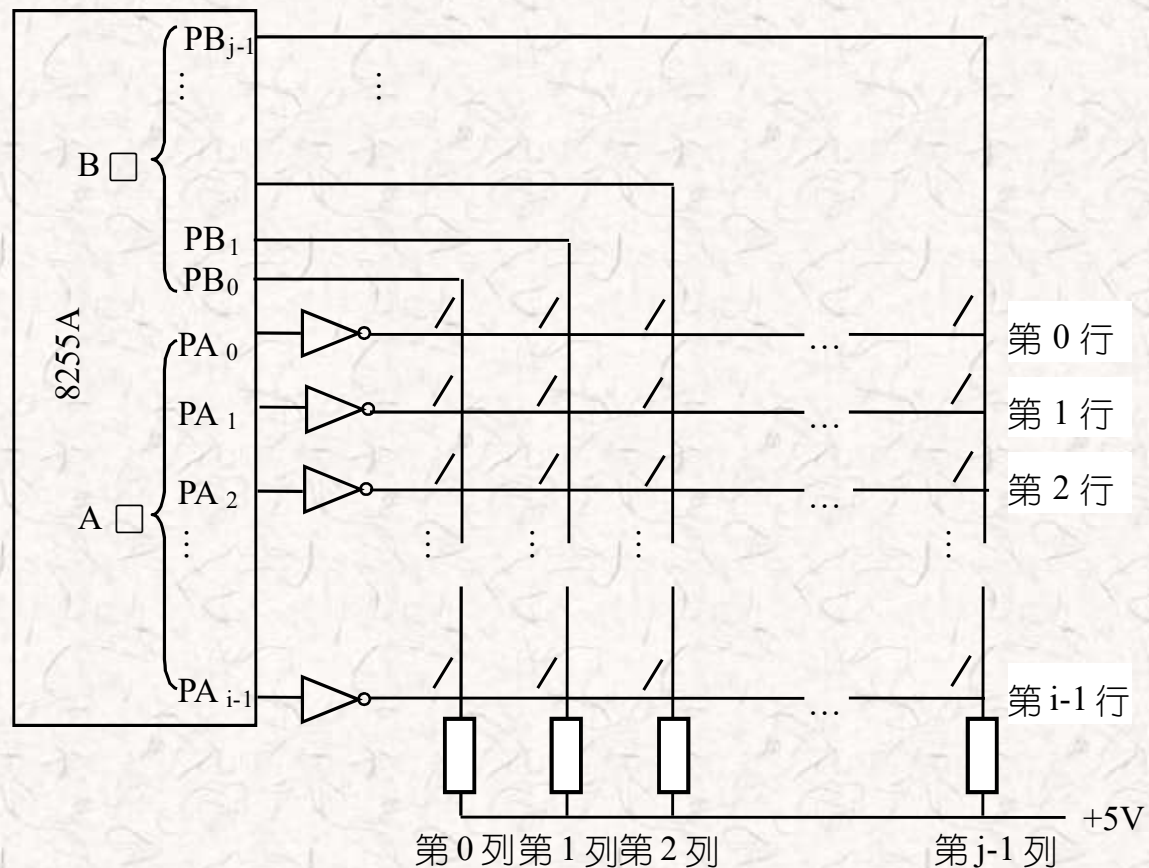
矩阵中每个按键有唯一的键值



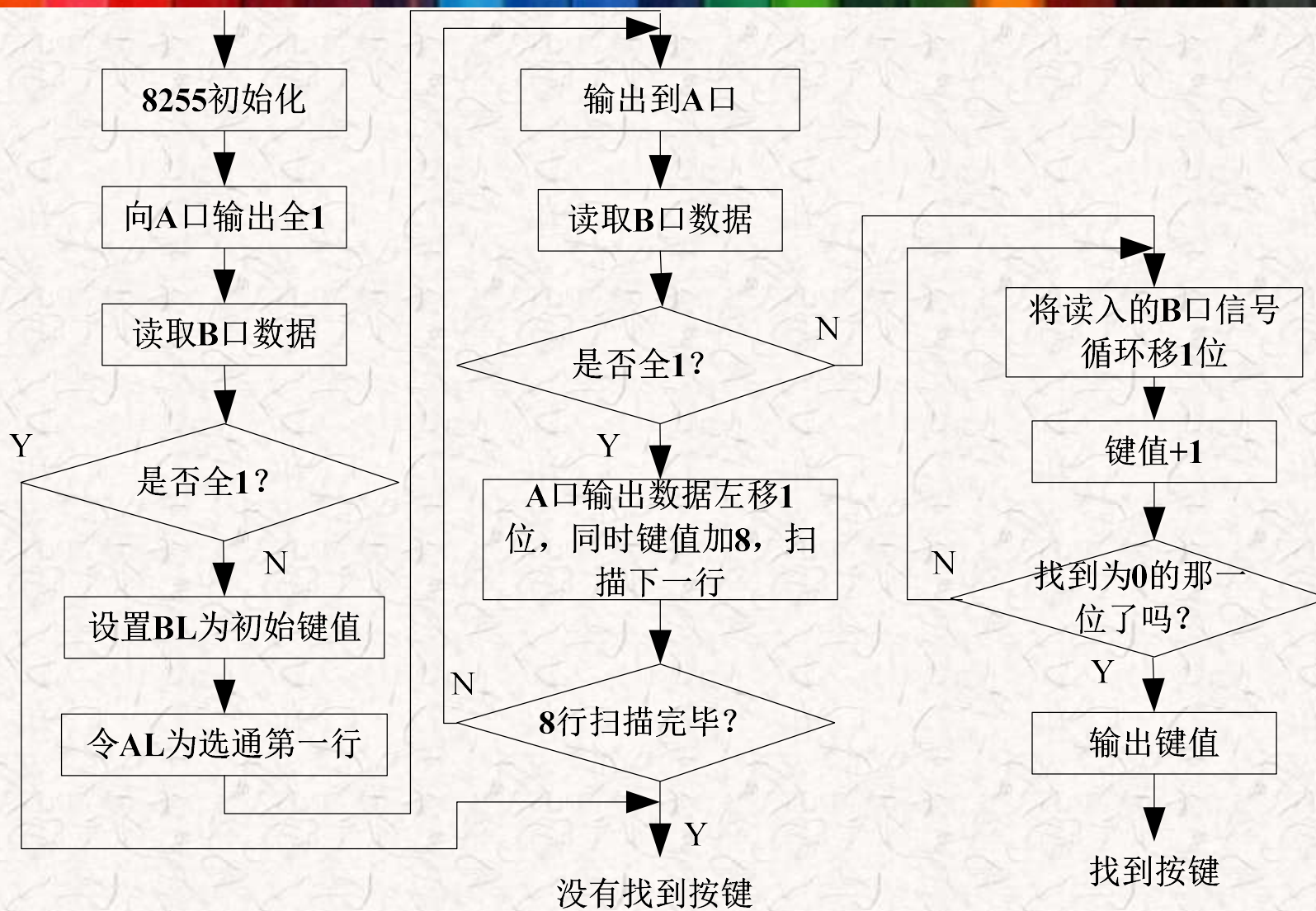
# 获取键值的方法-行扫描法

1. 判断有无键按下：  
所有行线置低电平，读入  
列线，是否全1？

2. 确定按下的键的键值：  
行线依次置为低电平，读入  
列线，判断是否有键按下，  
同时计算键值



# 行扫描流程图



# 行扫描法代码1

```
PORTA EQU    0FFF8H
PORTBEQU     0FFFAH
PORTCEQU     0FFFCH
CONTREQU     0FFFEH; 增强代码可读性
```

...

```
MOV    DX, CONTR
```

```
MOV    AL,10000011B ; 控制字
```

```
OUT    DX, AL
```

```
WAIT:  MOV    DX, PORTA
```

```
MOV    AL, 0FFH      ;选通各行
```

```
OUT    DX, AL
```

```
MOV    DX, PORTB
```

```
IN     AL, DX        ;检查各行
```

```
CMP    AL, 0FFH      ;有键压下吗 ?
```

```
JE     DONE          ;否, 跳走
```



# 行扫描法代码2

```
MOV    BL, 0           ;置键号初值
MOV    BH, 1           ;逐行扫描
MOV    CX, 8           ;设置行计数
FNDROW: MOV AL, BH
MOV    DX, PORTA
OUT    DX, AL
MOV    DX, PORTB       ;取列值
IN     AL, DX
CMP    AL, 0FFH        ;有键压下吗？
JNZ    FNDCOL          ;有，找键值
ROL    BH, 1           ;无，选通下一行
ADD    BL, 8           ;键值增加，跳过一行
DEC    CX
JNZ    FNDROW          ;循环扫描8行
```

# 行扫描法代码3

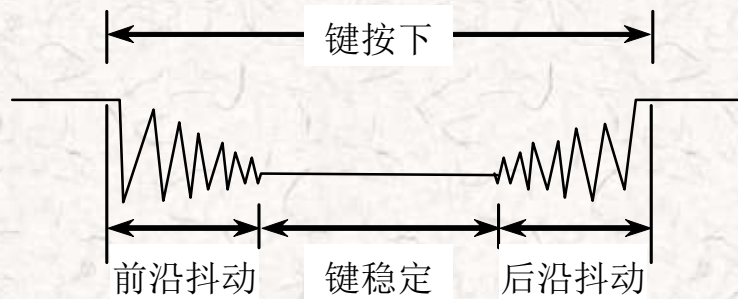
```
JMP     DONE           ;转无键压下处理
FNDCOL: ROR  AL, 1      ;处理被按键列值
        JNC     RIGHT   ;找到该列的键号
        INC     BL       ;BL中键号加1
        JMP     FNDCOL
RIGHT:  MOV     AL, BL    ;键号送AL
        ...
DONE:   转其他程序
```

问：这是一个多大的键盘？  
如第 5 行第 2 列按下，BL 中的键值是多少？  
(注意行列编号从 0 开始)

8\*8 42

## 增强设计稳定性

- 编写代码去除按键抖动



- 去抖应该加入键盘扫描程序的哪部分？

```
MOV DX, PORTB
WAITING: IN AL, DX
        CMP AL, 0FFH
        JE NOKEY

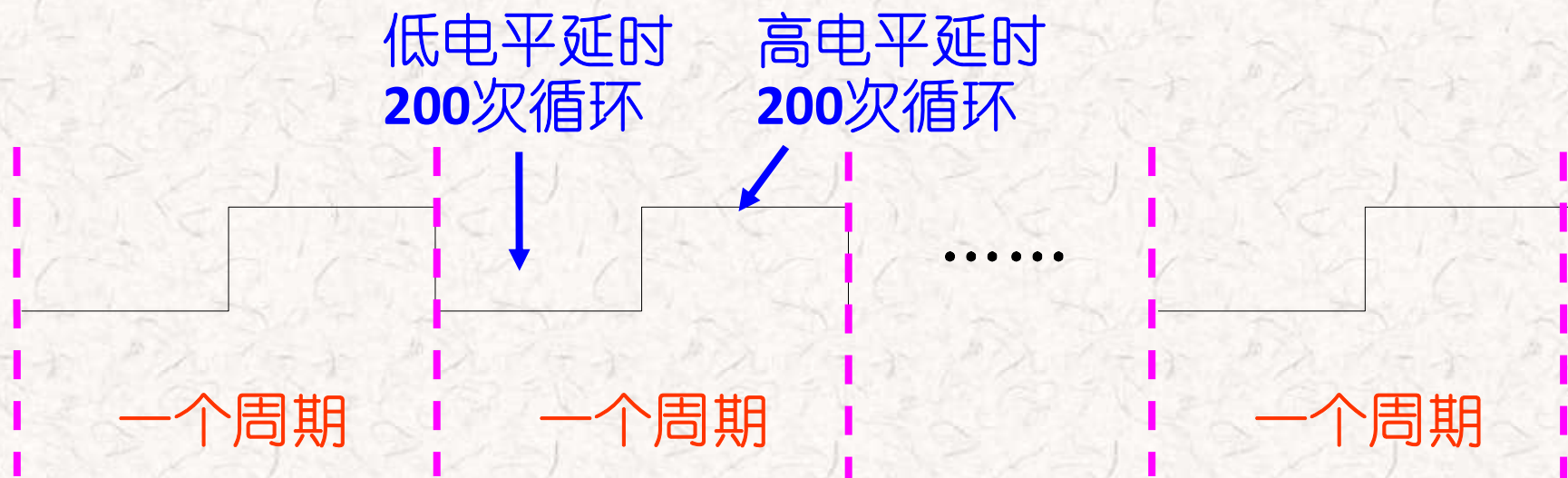
        MOV AH, AL
        MOV CX, 100
        DELAY: LOOP DELAY

        IN AL, DX
        CMP AL, AH
        JE STABLE
        JMP WAITING
```



## IO综合练习1--C口位输出编程

- 设置8255的A□、B□、C□高四位均工作在方式0，输入，C□低四位工作在方式0输出，编程在PC2产生10个周期的方波，周期为延时循环400次，设控制□地址为36H



## 参考代码

```
MOV AL, 9AH  
OUT 36H, AL
```

```
MOV BL, 10
```

```
NEXT: MOV AL, 05H  
OUT 36H, AL  
MOV CX, 200
```

```
DELAY1: DEC CX  
JNZ DELAY1
```

```
MOV AL, 04H  
OUT 36H, AL
```

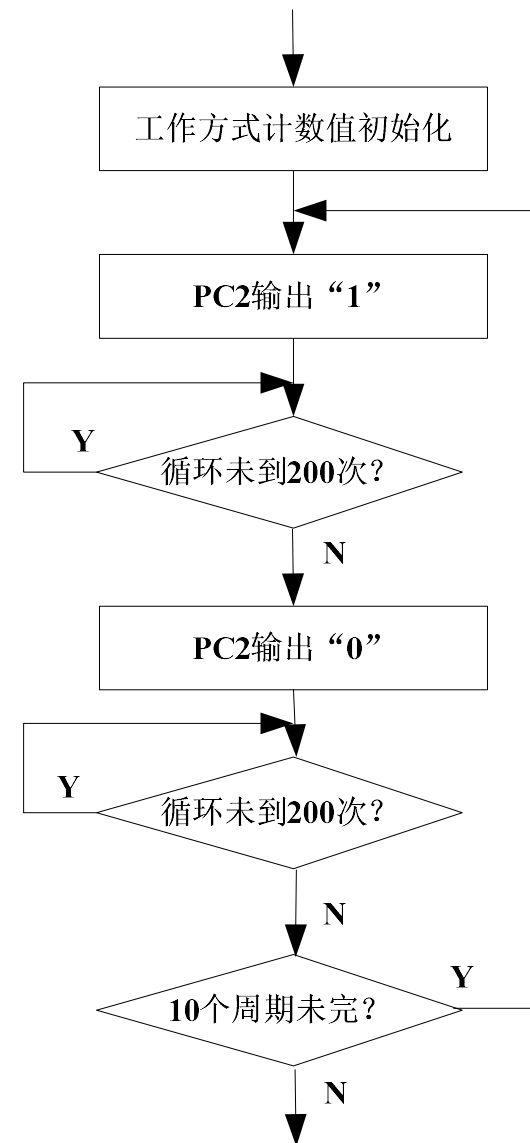
```
MOV CX, 200
```

```
DELAY2: DEC CX
```

```
JNZ DELAY2
```

```
DEC BL  
JNZ NEXT
```

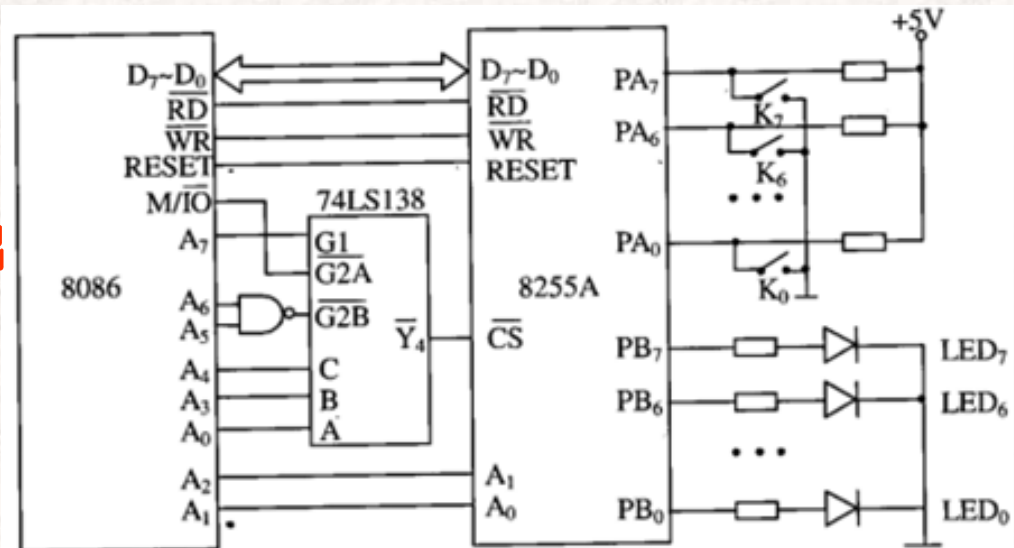
```
HLT
```



# IO综合练习2

A口方式0，输入，B口方式0，输出  
0F0H、0F2H、0F4H、0F6H  
IO空间

初始化设置命令字，读取A口数据，取反，写入B口



- (1) 图中8255的A口和B口工作在方式几？分别为输入还是输出？
- (2) 假设K7闭合，K6~K0断开，则读取PA口数据得到 **7FH**，向B口写入15H，LED0~7亮着的灯有 **LED0、LED2、LED4**
- (3) 8255的端口地址是多少？在IO空间还是存储器空间？
- (4) 编写程序对8255初始化，读取开关状态，如果有开关闭合，则令相应LED点亮。



## 参考代码--找出下面代码的错误

APORT	EQU	F0H
BPORT	EQU	F2H
CONTR	EQU	F6H

MOV BL, 99H	; 工作方式命令字
<u>IN APORT, BL</u>	; 初始化
REPEAT: MOV AX, APORT	
<u>IN AL, AX</u>	; 读A□
<u>NEG AL</u>	; 取反
<u>MOV BPORT, AL</u>	; 写入到B□
JMP REPEAT	

## 参考代码

```
APORT    EQU    0F0H
BPORT    EQU    0F2H
CONTR     EQU    0F6H
```

```
MOV  AL, 99H      ; 工作方式命令字
OUT  CONTR, AL
REPEAT: IN  AL, APORT
NOT  AL           ; XOR AL, 0FFH 亦可
OUT  BPORT, AL
JMP  REPEAT
```

## 10综合练习3

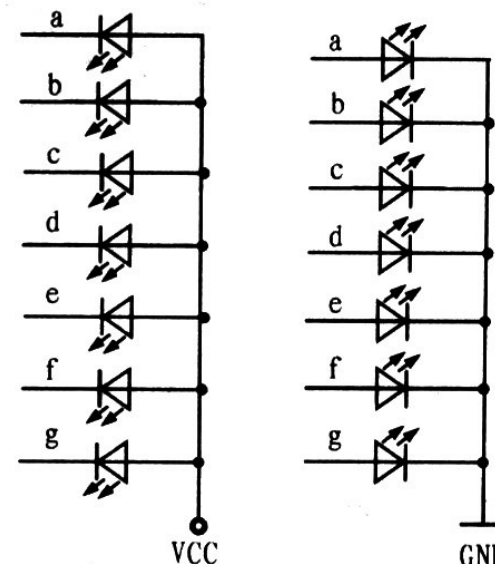
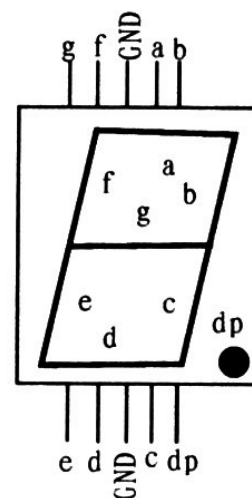
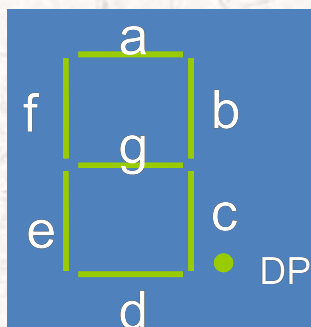
- 在**8086**系统中使用**8255**芯片实现设定温度值输入和当前温度值显示功能
- 要求**8255**连接**2**个**8**段数码管，分别用于显示温度设定值和当前值的个位和十位；
- 再构成一个**4\*4**的键盘，用于接收工作人员的温度设置
- 为**8255**四个端口分配地址为**70H**、**72H**、**74H**、**76H**
- 完成电路设计
- 完成温度设置和显示程序，具体要求如下：



## 编程要求

- (1) 对**8255**进行初始化；
- (2) 编写键盘扫描子程序，输出参数为键值，位于**AH**中，有效范围**0~15**；
- (3) 将键值**0~15**分别定义为数字**0~9**、设定、确定、退回，其它按键备用；
- (4) 实现目标温度设定和控制功能，按“设定”，则显示当前设定温度（设温度值均为正数），通过数字键输入，“退回”键修改，“确定”键完成输入。
- (5) 设温度值为两位十进制数

# 认识数码管



(a) 符号和引脚 (b) 共阳极连法 (c) 共阴极连法

图 1 LED 数码管外形及其内部结构

TABLE 3FH, 06H, 5BH, .....

MOV AL,5 ; 待显示数字

MOV AH, 0;

MOV BX,AX

MOV AL, TABLE[BX]

OUT PORTB, AL

显示字符	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0
七段代码 (H)	06	5B	4F	66	6D	7D	07	7F	6F	77	7C	39	5E	79	31	3F

# 电路设计

- 如何实现4\*4键盘？如何实现2个数码管驱动？
- 键盘的（行线 or 列线）需要通过电阻上拉？为什么？
- 如何实现70H,62H,74H,76H地址分配？

• **A7 A6 A5 A4 A3 A2 A1 A0**

• **0 1 1 1 0 0 0 0**

• **0 1 1 1 0 0 1 0**

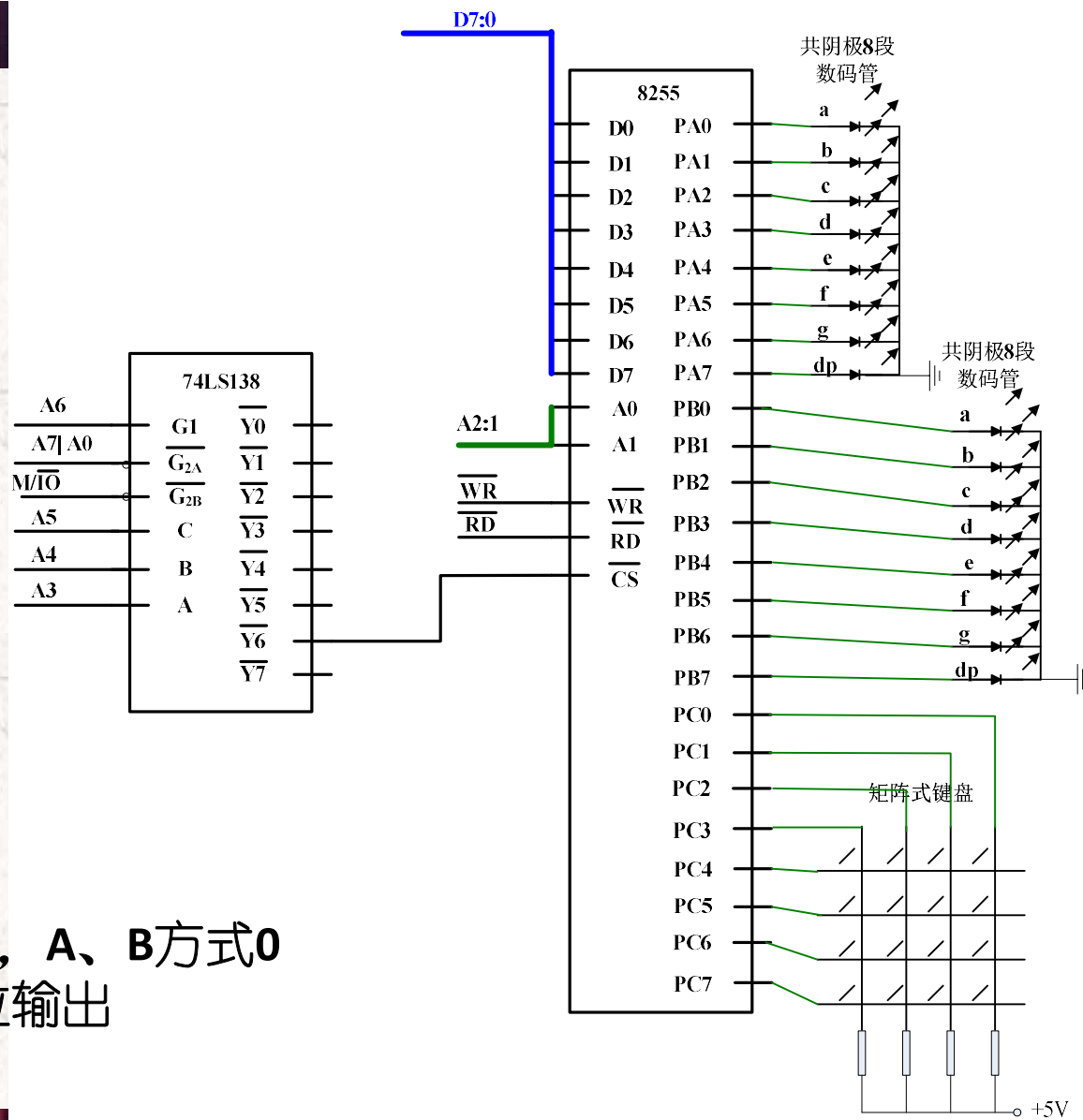
• **0 1 1 1 0 1 0 0**

• **0 1 1 1 0 1 1 0**

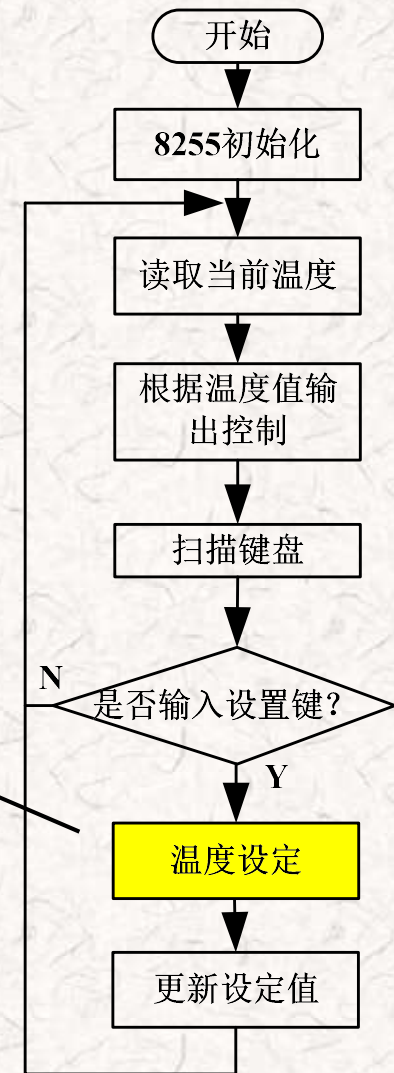
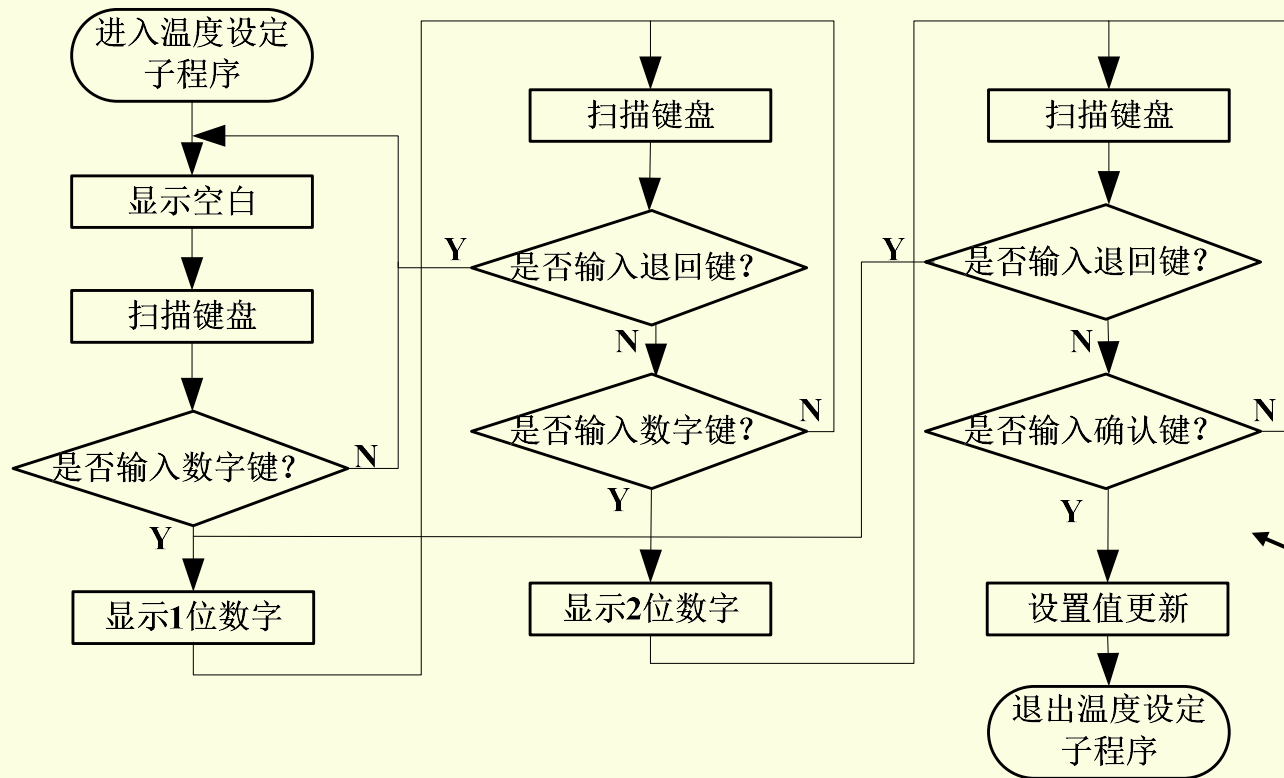


# 参考电路

工作方式：10001000B，88H，A、B方式0  
输出，C高四位输入，低四位输出



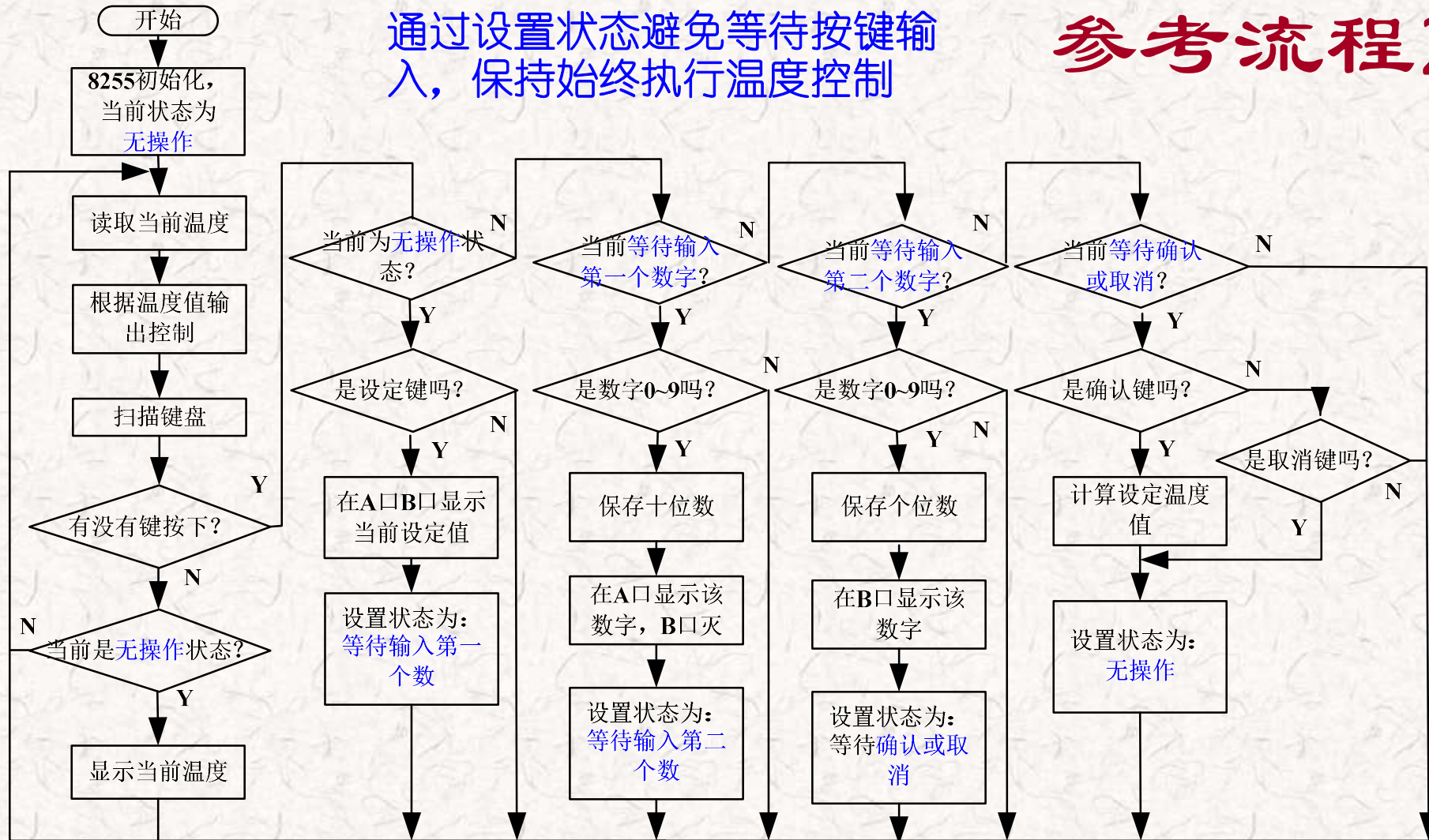
# 参考流程1



存在什么问题？ 等待输入导致其它响应暂停

通过设置状态避免等待按键输入，保持始终执行温度控制

## 参考流程2





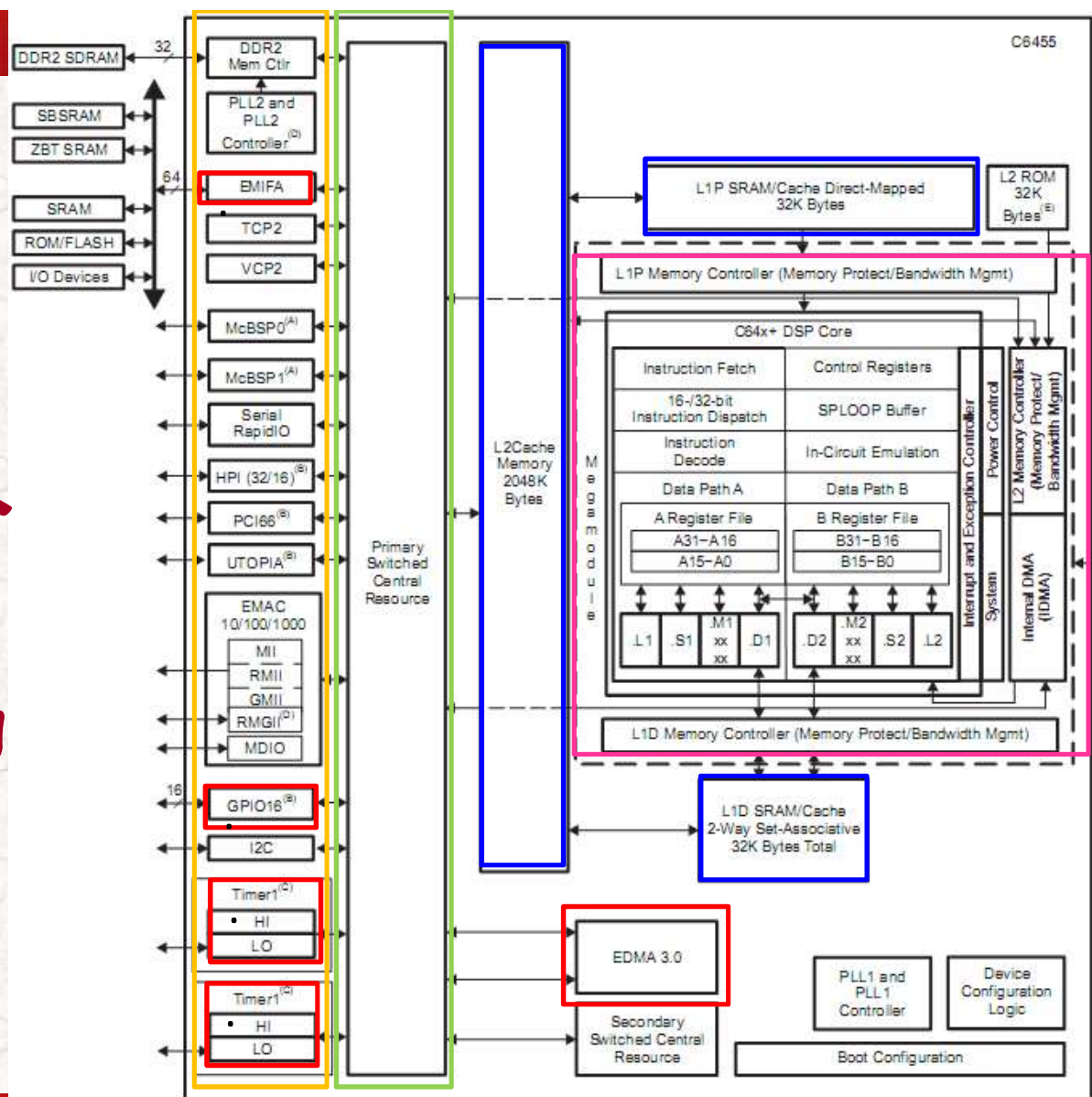
# 温度显示参考代码

- 温度显示代码

TABLE 3FH, 06H, 5BH, .....

XOR AH,AH	;清除高字节, 准备被除数 <b>AX</b>
MOV BH,AH	;预备 <b>BX</b> 用于间接寻址
MOV CL,10	;除数为 <b>10</b>
DIV CL	;商在 <b>AL</b> 中, 余数在 <b>AH</b> 中
MOV BL,AL	;准备显示商, 即十位数
MOV AL, TABLE[BX]	;转为七段码
OUT PORTB, AL	;输出显示十位
MOV BL,AH	;准备显示余数, 即个位数
MOV AL, TABLE[BX]	;转为七段码
OUT PORTA, AL	;输出显示个位

# 拓展： 简单认识 DSP 6455 系统结构



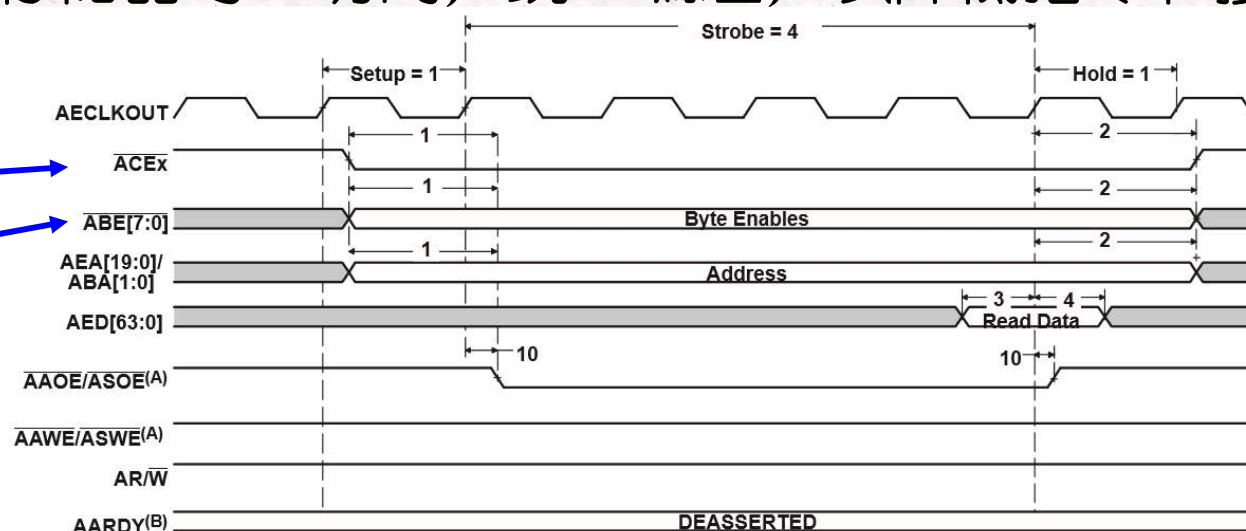
# 现代DSP的外部存储器接口-EMIF

特点1：支持异步和同步外部存储器访问

特点2：预先划分存储空间，输出片选信号, 降低板级电路复杂度

特点3：不区分存储器与IO访问，统一编址，以降低指令和接口电路复杂度

直接输出片选  
字节选通信号

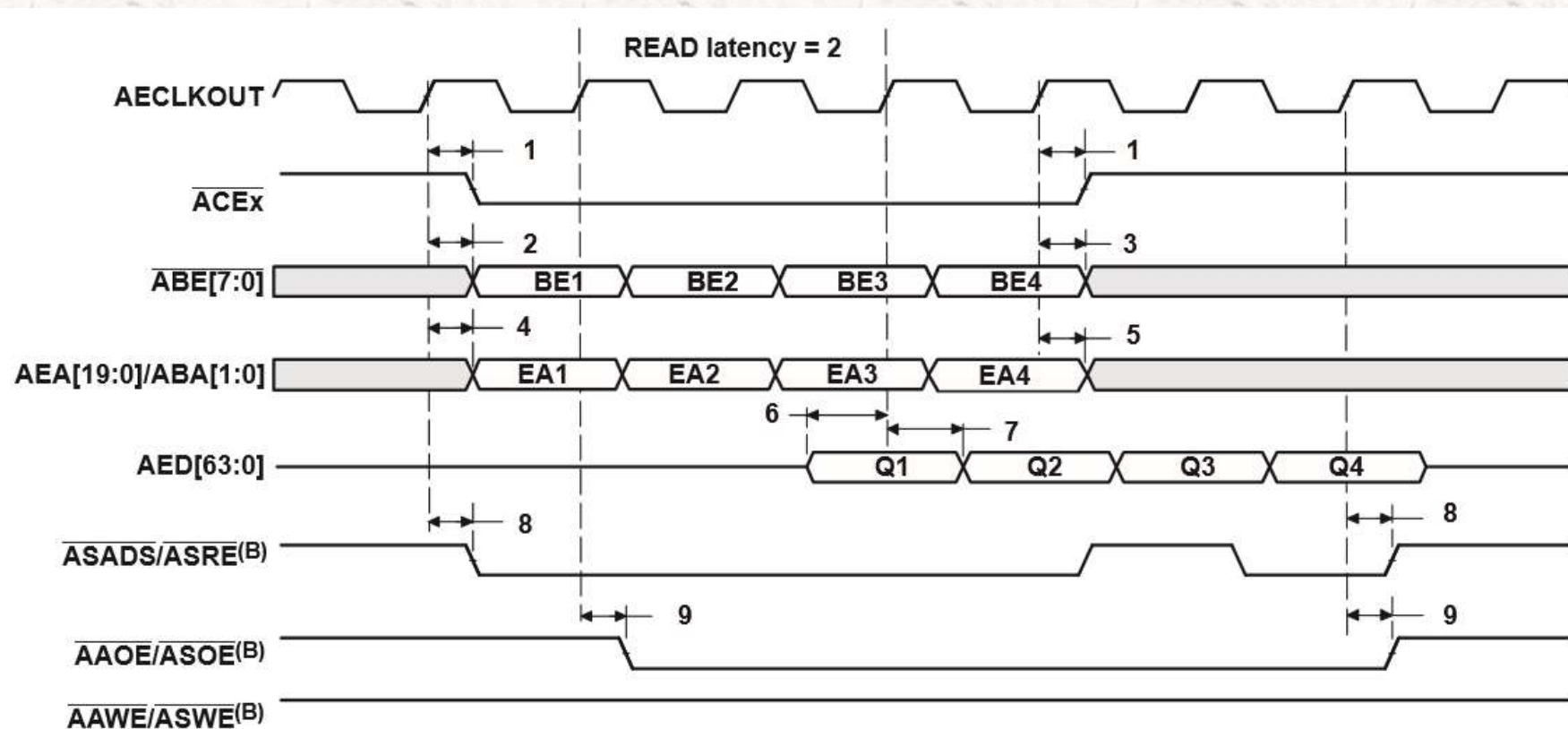


- A  $\overline{AAOE}/\overline{ASOE}$  and  $\overline{AAWE}/\overline{ASWE}$  operate as  $\overline{AAOE}$  (identified under select signals) and  $\overline{AAWE}$ , respectively, during asynchronous memory accesses.
- B Polarity of the AARDY signal is programmable through the AP field of the EMIFA Async Wait Cycle Configuration register (AWCC).

Figure 7-33. Asynchronous Memory Read Timing for EMIFA



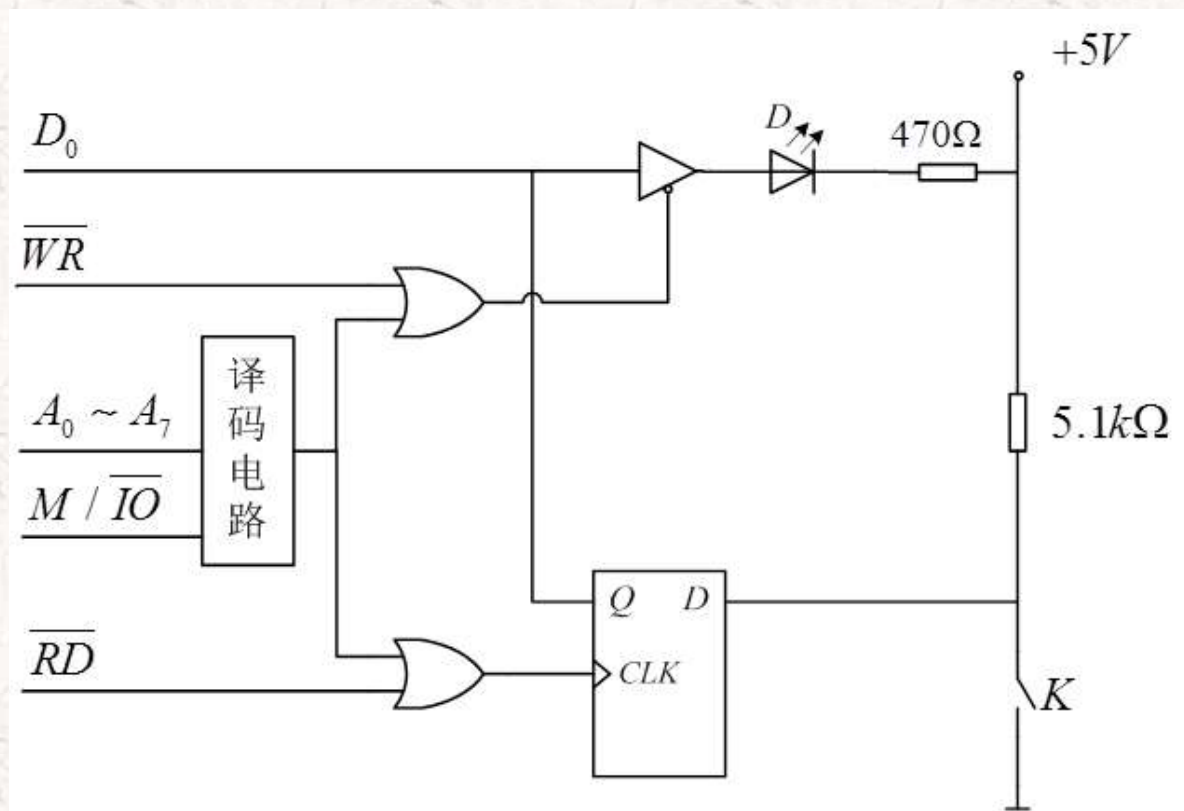
## 同步存储器读访问举例--高带宽传输



# 现代DSP的输入输出接口--GPIO

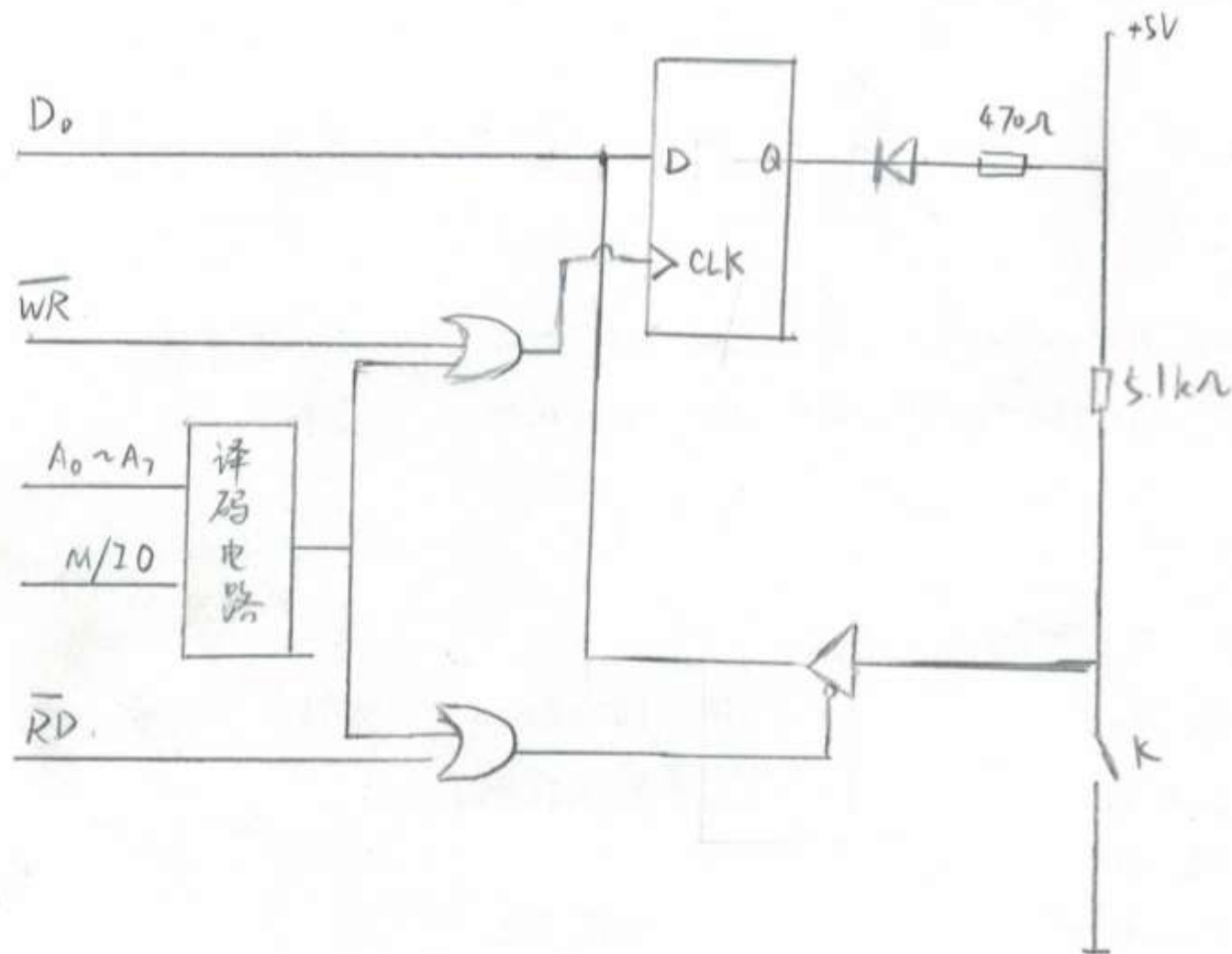
- 通常**8~16**位
- 可按位编程为输入或输出
- 与中断请求功能合并，检测跳变，触发中断（可以直接关联**DMA**传输）

# 协同练习1-电路找错与修正





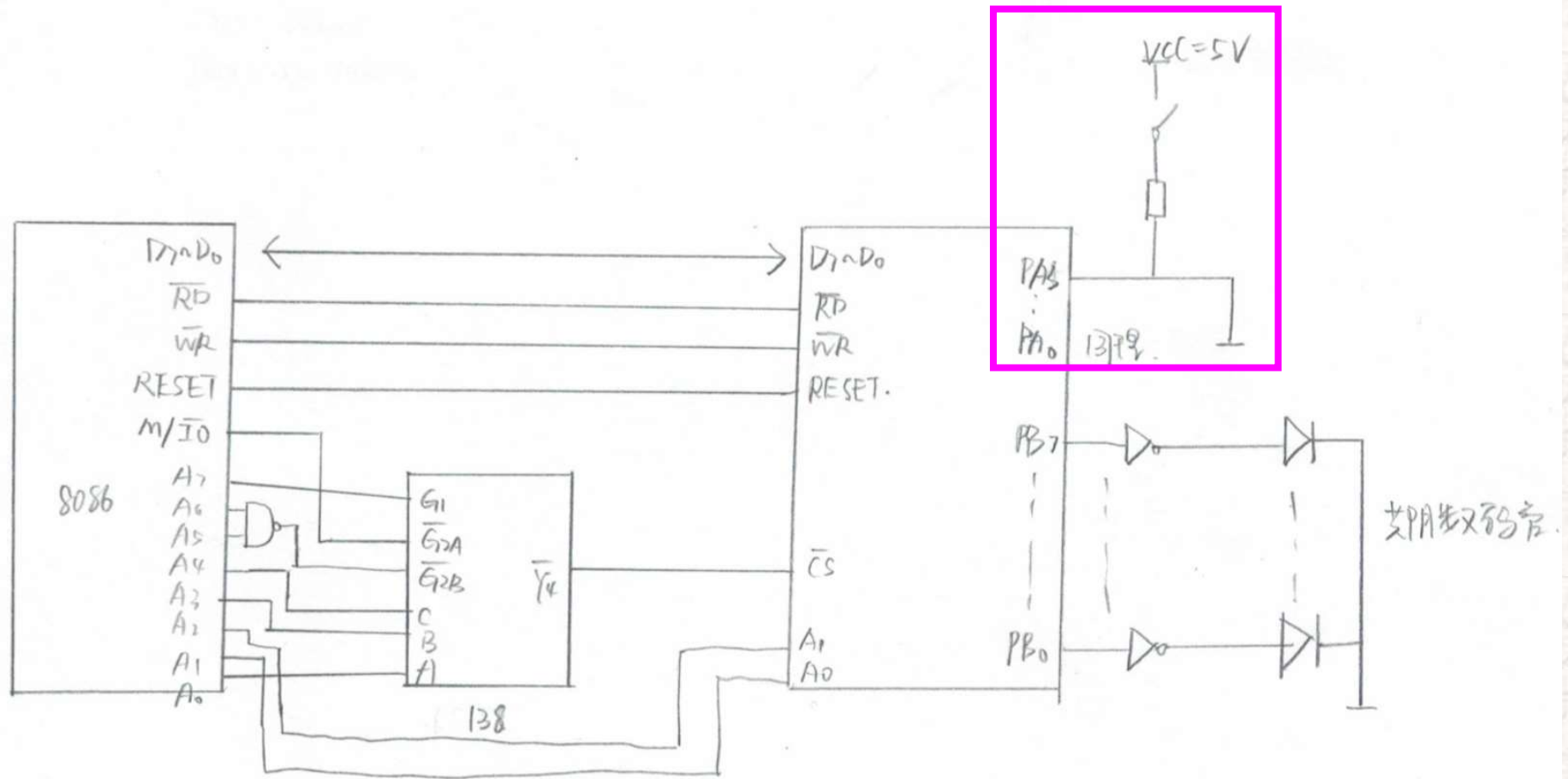
# 示例设计



## 协同练习2-开关与数码管

- 设计电路读取4位拨码开关的状态并将其组合作为十六进制数显示在七段数码管上。假设开关闭合代表1，断开代表0。
- 硬件电路设计：使用8255完成电路设计
- 软件编程：（1）编写8255初始化程序
- （2）编写读取开关状态并显示的程序

## 示例设计-电路1





# 示例设计-代码1

A□: 方式0输入  
B□: 方式0输出  
C□: 输出

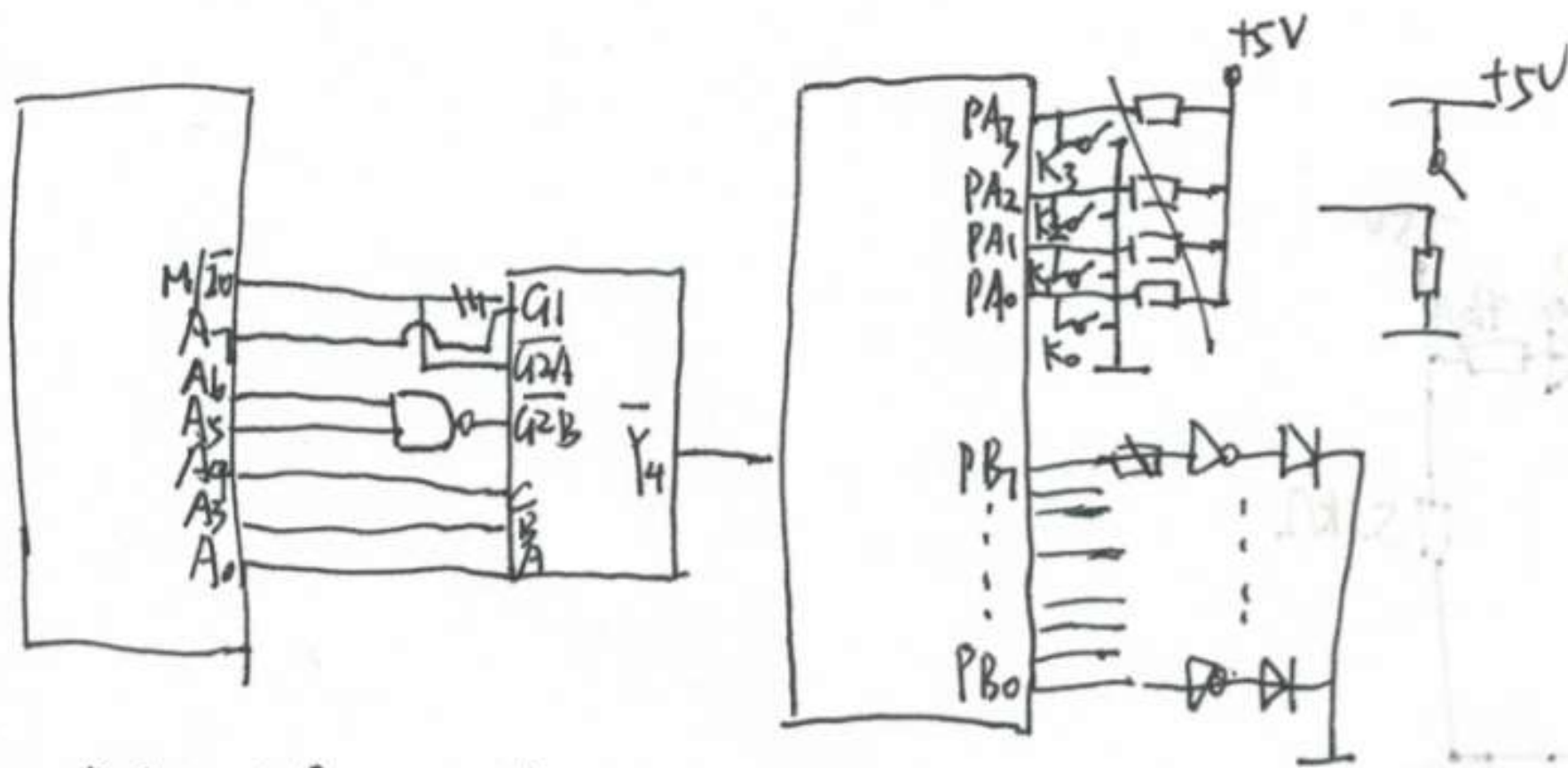
```
APORT EQU 0F0H
BPORT EQU 0F2H

DATA SEGMENT
TABLE DB ...
DATA ENDS

ASSUME MOV AL, 90H
OUT 63H, AL

IN-PORTA: IN AL, 0F0H
AND AL, 0FH
MOV BX, OFFSET TABLE
XLAT
OUT 0F2H, AL
CALL DELAY
JMP IN-PORTA
```

## 示例设计-电路2



# 示例 设计- 代码 2

```
APORT EQU 0F0H
BPORT EQU 0F2H
DATA \ CONTR EQU 0F6H
SEGMENT TABLE DB 40H, 79H, 24H, 30H, 19H, ... < DATA ENDS
MOV AL, 99H
OUT CONTR, AL
REPEAT: IN AL, APORT
AND AND AL, 0FH
MOV BX, OFFSET TABLE
MOV MOV AH, 0
ADD BX, AX
MOV AL [BX]
OUT BPORT, AL
JMP REPEAT
```

A□: 方式0输入  
B□: 方式0输出  
C□: 输入



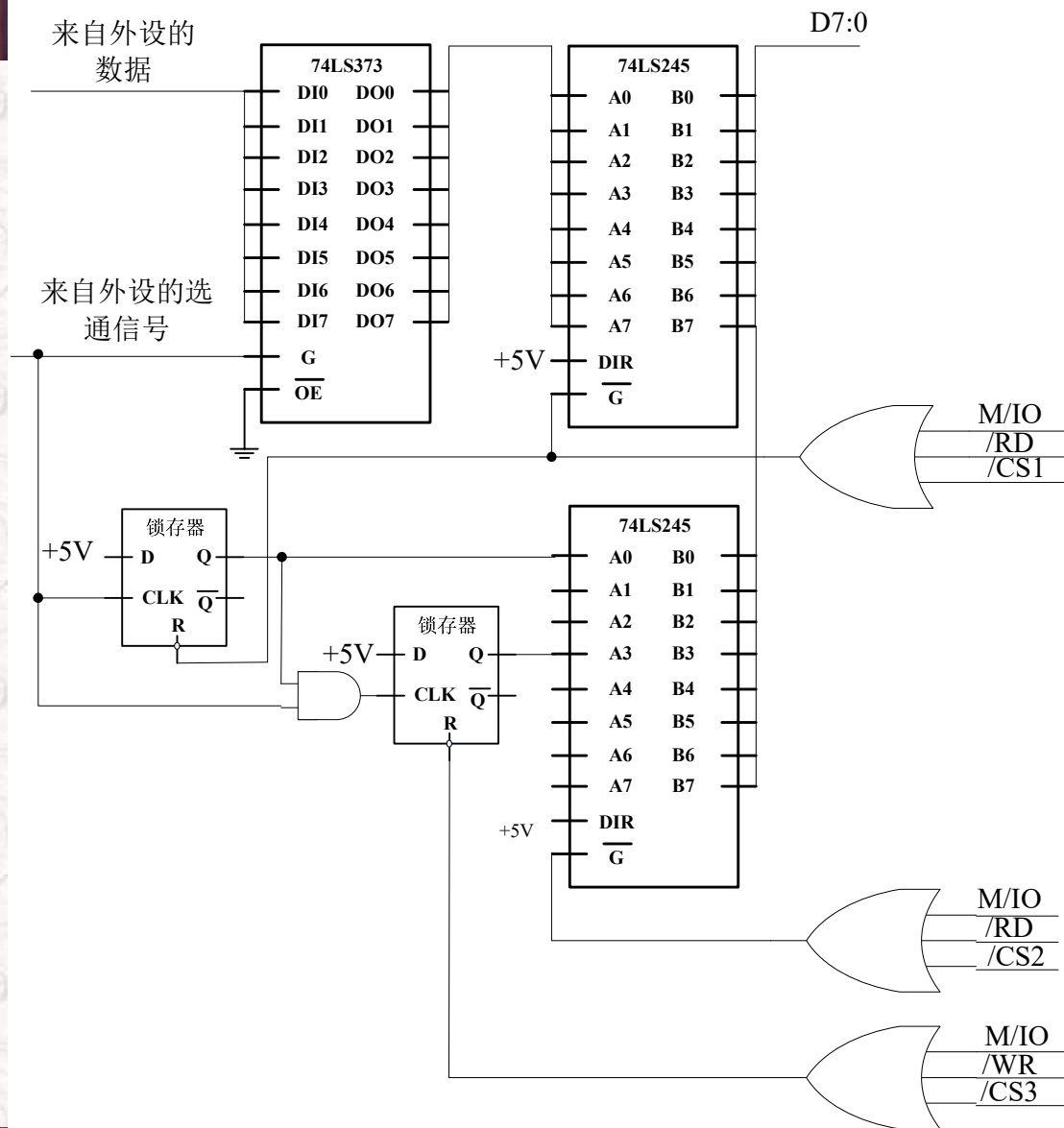


## 协同练习3-查询式数据传输

- (1) 为**8086**设计具有**3**个接口的外设，**8bit**数据输入端口、状态端口、命令端口。外设支持查询式输出功能，其中状态接口提供以下**2**个状态：
  - **输入端口数据就绪**：输入端口有数据待处理，**1**有效，**8086**读取后该标志清零；
  - **输入错误**：如果**8086**尚未读取数据时，有新的数据到达该输入接口，则置**1**，并保持，直到**8086**向**命令端口**执行写操作
- (2) 查询并执行：如果输入端口就绪则读取一个字节写入物理地址**80000H**的存储单元，接收字节数**+1**；如果有出错标志，则错误次数**+1**，并写命令端口清除该标志；

# 设计3参考- 电路

译码电路略



## 设计3参考-编程

```
DATAP EQU CS1  
STATEP EUQ CS2  
COMM EQU CS3
```

```
IN AL, STATEP  
TEST AL, 01H ; D0 READY  
JNE READ  
TEST AL, 08H ; D3 ERROR  
JNE ERROR  
JMP CONTINUE
```

```
READ: IN AL, DATAP  
MOV [SI], AL  
INC SI  
JMP CONTINUE  
ERROR: OUT COMM, AL  
CONTINUE: ...
```



# 作业

- P220
- 1~4
- 7~12
- 13 与8086连接，并将端口地址改为80H，82H，84H，86H，再进行后续的设计，需要考虑奇偶地址问题
- 14 与8086连接，并将端口地址改为61H，63H，65H，67H，再进行后续的设计，需要考虑奇偶地址问题



## 预告-第七章需要掌握的内容

- 定时计数器的概念
- **8253方式0和方式3**
- 其它工作方式不需要掌握