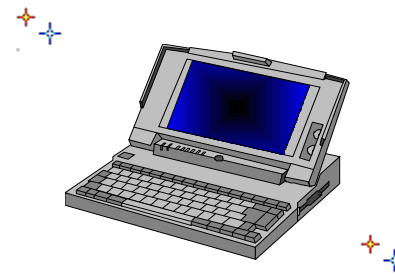
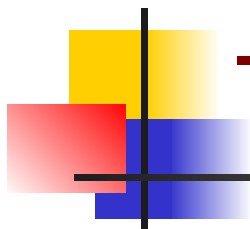


# 一般数据传送指令

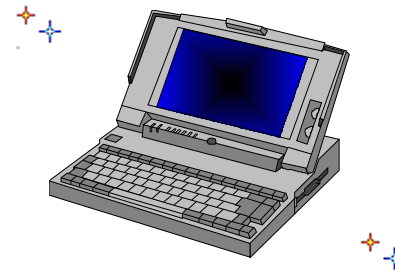


## ▪ 注意点:

- 两操作数字长必须相同;
- 两操作数不允许同时为存储器操作数;
- 两操作数不允许同时为段寄存器;
- 在源操作数是立即数时, 目标操作数不能是段寄存器;
- **IP**和**CS**不作为目标操作数, **FLAGS**一般也不作为操作数在指令中出现。



# 一般数据传送指令例



## ▪ 判断下列指令的正确性：

▪ **MOV AL, BX**

错，两操作数字长不相等

▪ **MOV AX, [SI]05H**

对，源操作数为相对寻址

▪ **MOV [BX][BP], BX**

错，目标操作数寻址方式错误

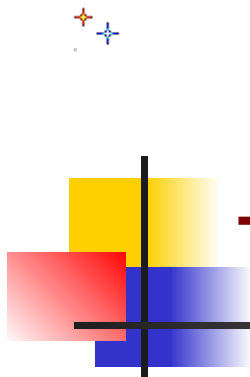
▪ **MOV DS, 1000H**

错，不能用立即寻址方式为段寄存器赋值

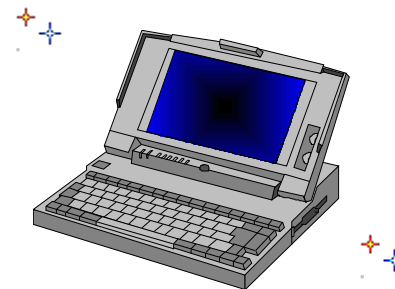
▪ **MOV DX, 09H**

▪ **MOV [1200], [SI]** 对，

错，两操作数不能同时为存储器操作数



# 寻址方式练习



- 注意点：设DS=6000H，ES=2000H，SS=1500H，SI=00A0H，BX=0800H，BP=1200H，字符常数VAR为0050H.说明以下各类指令源操作数的寻址方式及存储器操作数的物理地址

- **MOV AX, BX**
- **MOV DL, 80H**
- **MOV AX, VAR[BX][SI]**
- **MOV AL, 'B'**
- **MOV DI, ES: [BX]**
- **MOV DX, [BP]**
- **MOV BX 20H[BX]**

寄存器寻址

立即寻址

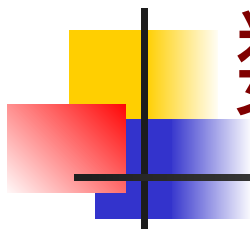
基址变址寻址 物理地址：608F0H

立即寻址

寄存器间接寻址 物理地址：20800H

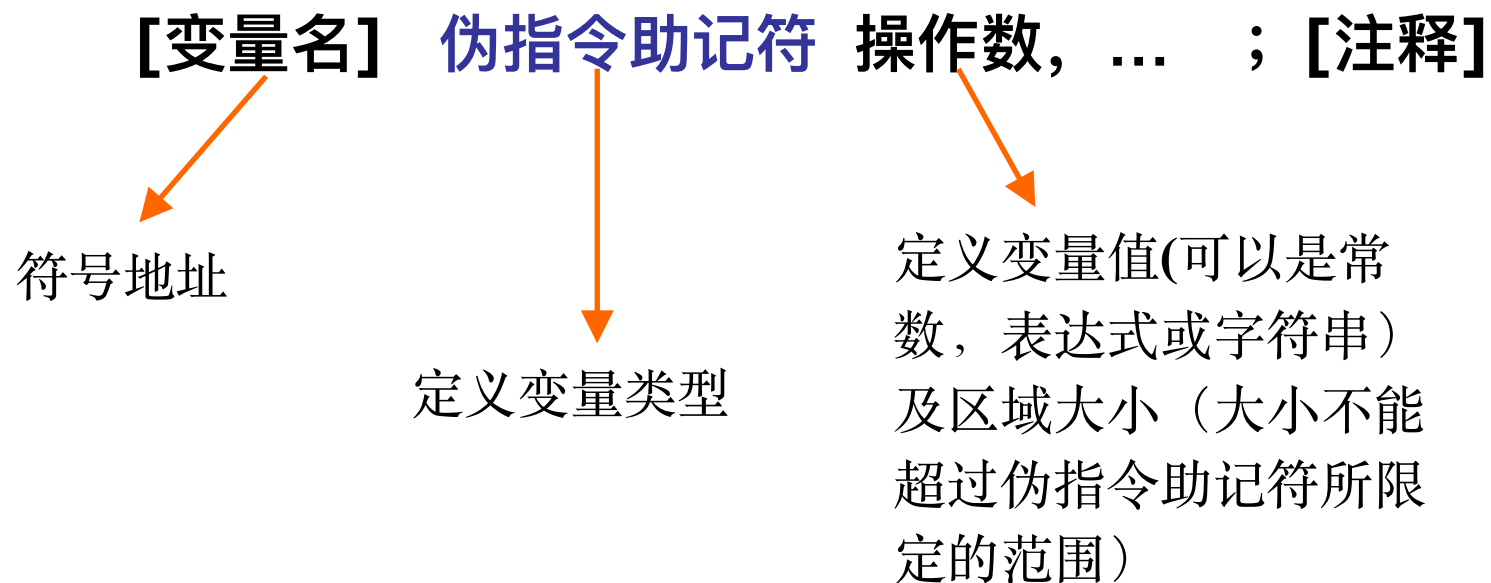
寄存器间接寻址 物理地址：16200H

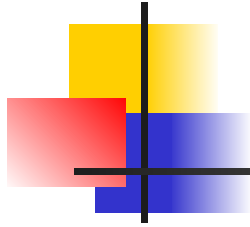
寄存器相对寻址 物理地址：60820H



# 数据定义伪指令

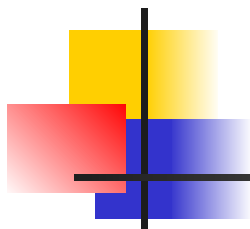
- 用于定义数据区中变量的类型及大小
- 格式：





# 数据定义伪指令助记符

- **DB** 定义的变量为字节型 (指向的每一个操作数占**1个字节**单元)
- **DW** 定义的变量为字类型 (双字节)
- **DD** 定义的变量为双字型 (4字节)
- **DQ** 定义的变量为4字型 (8字节)
- **DT** 定义的变量为10字节型

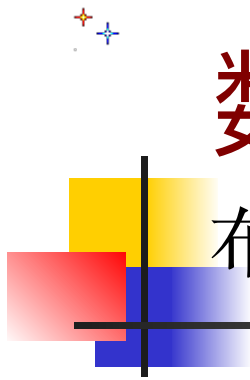


## 数据定义伪指令例

- **DATA1 DB 11H, 22H, 33H, 44H**
- **DATA2 DW 11H, 22H, 3344H**
- **DATA3 DD 11H\*2, 22H, 33445566H**

以上变量在内存  
中的存放形式





# 数据定义伪指令例\_变量在内存中的分布



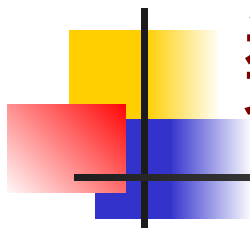
**DATA1**

11
22
33
44
11
00
22
00
44
33

**DATA2**

**DATA3**

22
0
0
0
22
0
0
0
66
55
44
33



# 数据定义伪指令的几点说明

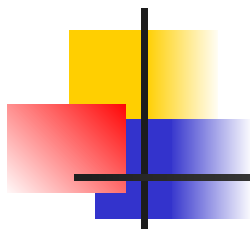


- 伪指令的性质决定所定义变量的类型；
- 定义字符串必须用**DB**伪指令
- 例：

**DATA1 DB 'ABCD', 66H**

41H	'A'
42H	'B'
43H	'C'
44H	'D'
66H	



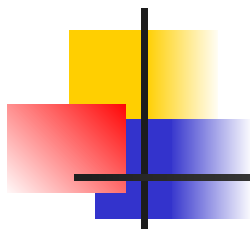


# 重复操作符

- 作用：
  - 为一个数据区的各单元设置相同的初值
- 目的：
  - 常用于声明一个数据区
- 格式：

[变量名] 伪指令助记符 n **DUP** (初值, ...)
- 例：

**DW 20 DUP (0)**  
**M1 DB 10 DUP (0)**



# “?”的作用

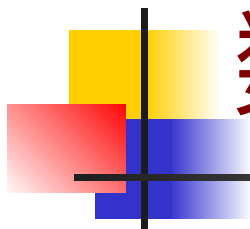
- 表示随机值，用于预留存储空间

▪ **MEM1 DB 34H, 'A', ?**

**DW 20 DUP ( ? )**

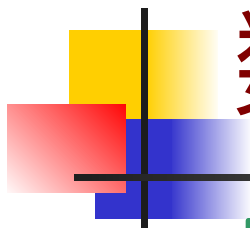
随机数  
占1个字节单元

预留40个字节单元，每单元为随机值



# 数据定义伪指令例

- **M1 DB 'How are you?'**
- **M2 DW 3 DUP(11H), 3344H**
- **DB 4 DUP ( ? )**
- **M3 DB 3 DUP (22H, 11H, ? )**



# 数据定义伪指令例



M1

'H'
'o'
'w'
' '
'a'
'r'
'e'
' '
'y'
'o'
'u'
'?'

M2

11H
00H
11H
00H
11H
00H
44H
33H
XX
XX
XX
XX

M3

22H
11H
XX
22H
11H
XX
22H
11H
XX

随机数



# 汇编语言

---

**DATA SEGMENT**

**X DW 2645H, 3576H**

**Y DW, 4328H, 2598H**

**Z DB 4 DUP (?)**

**DATA ENDS**

**CODE SEGMENT**

**ASSUME CS:CODE, DS:DATA**

**START:**

**MOV AX, DATA**

**MOV DS, AX**

**MOV BX, X**

**ADD BX, X[2]**

**MOV Z, BX**

**MOV BX, X**

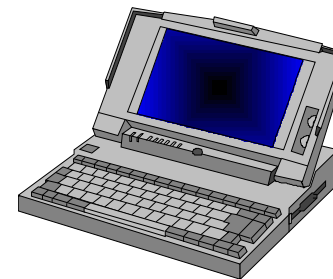
**ADD BX, Y**

**MOV Z, BX**

**MOV DX, X[2] ; [X+2]**

**ADD DX, Y[2] ; [Y+2]**

**MOV Z[2], DX**



# LEA指令

- 比较下列指令：

**MOV SI, DATA1**

执行结果：SI=1234H

执行结果：SI=DATA1

**MOV BX, [BX]**

执行结果：BX=7788H

执行结果：BX=1100H

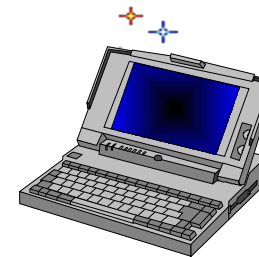
符号  
地址

DATA1

1100H

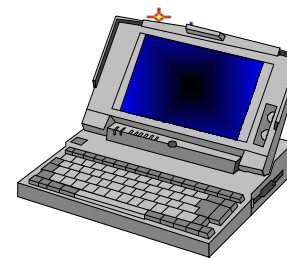
BX=1100H

...
34H
12H
...
88H
77H
...

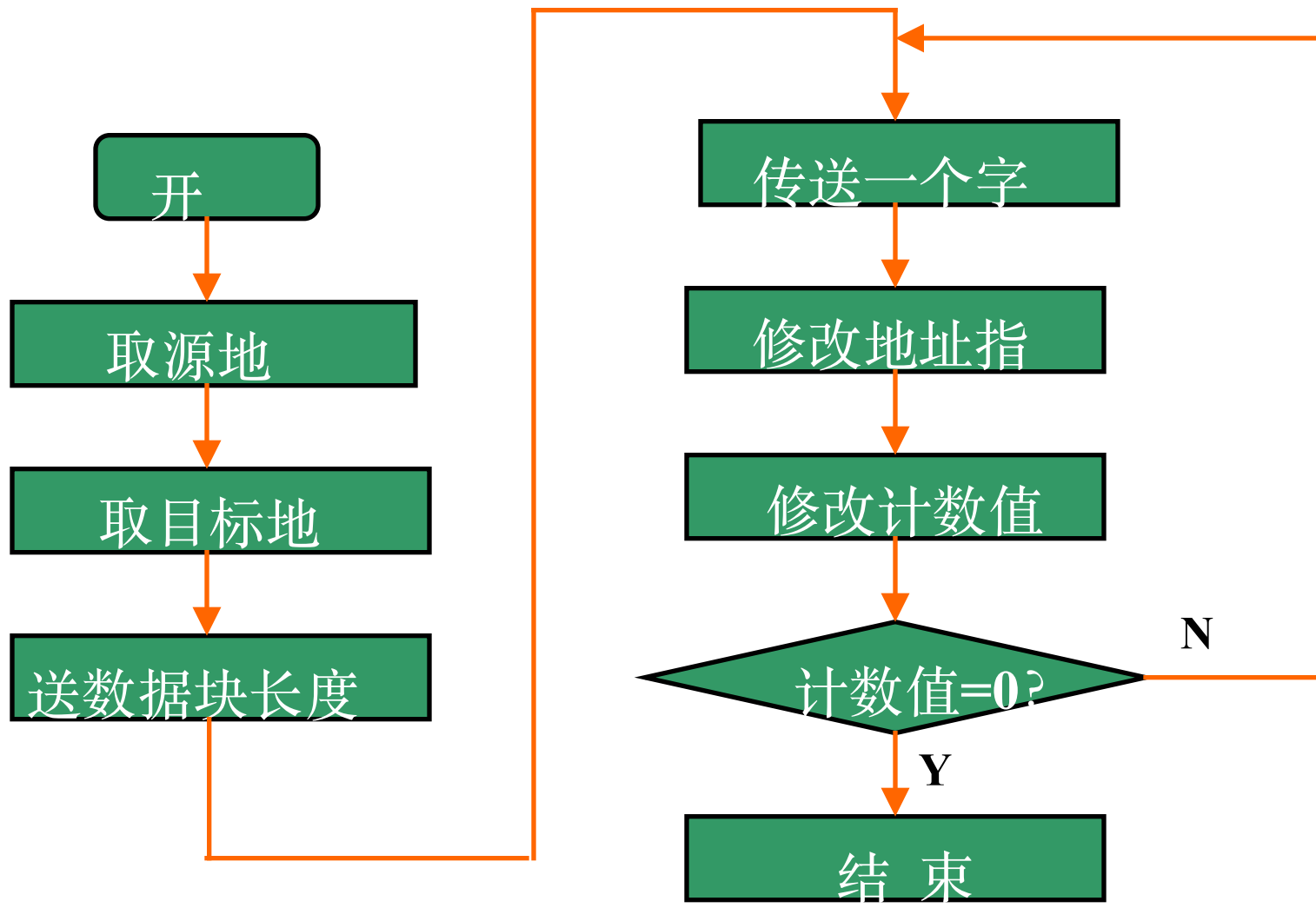


# LEA指令在程序中的应用

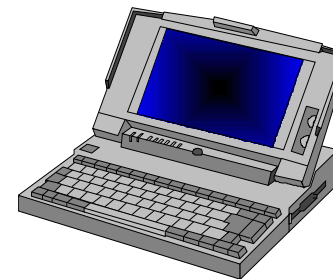
- 将数据段中首地址为**MEM1** 的**50**个字节的数  
据传送到同一逻辑段首地址为**MEM2**的区域  
存放。编写相应的程序段 。



# LEA指令在程序中的应用





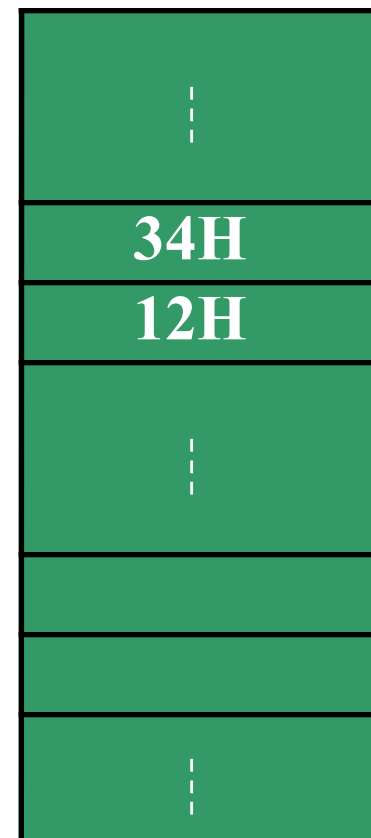


# LEA指令在程序中的应用

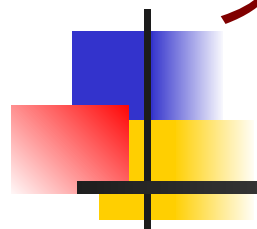
```
LEA SI, MEM1
LEA DI, MEM2
MOV CL, 50
NEXT: MOV AL, [SI]
      MOV [DI], AL
      INC SI
      INC DI
      DEC CL
      JNZ NEXT
      HLT
```

MEM1

MEM2

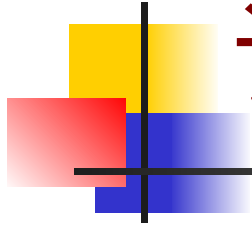


# 第5章 译码方式



(内存存储器设计)

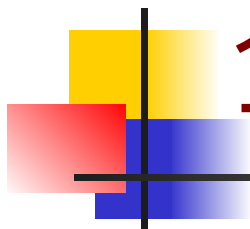
---



# 译码方式

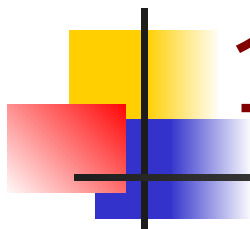
---

- 全地址译码
- 部分地址译码



# 全地址译码

- 用全部的高位地址信号作为译码信号，使得存储器芯片的每一个单元都占据一个唯一的内存地址。



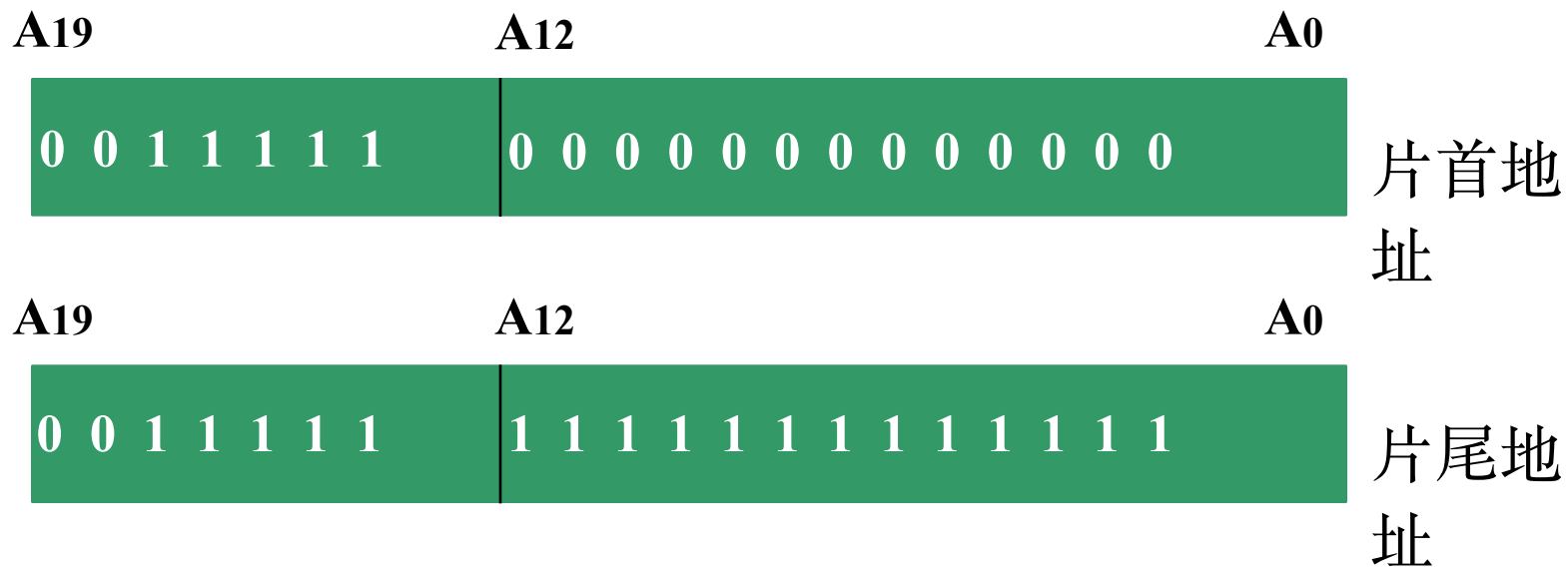
# 全地址译码例

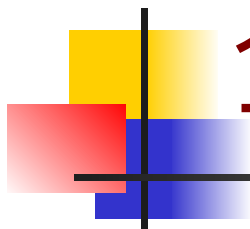
- 若已知某**SRAM 6264**芯片在内存中的地址为：  
**3E000H~3FFFFH**
- 试画出将该芯片连接到系统的译码电路。



# 全地址译码例

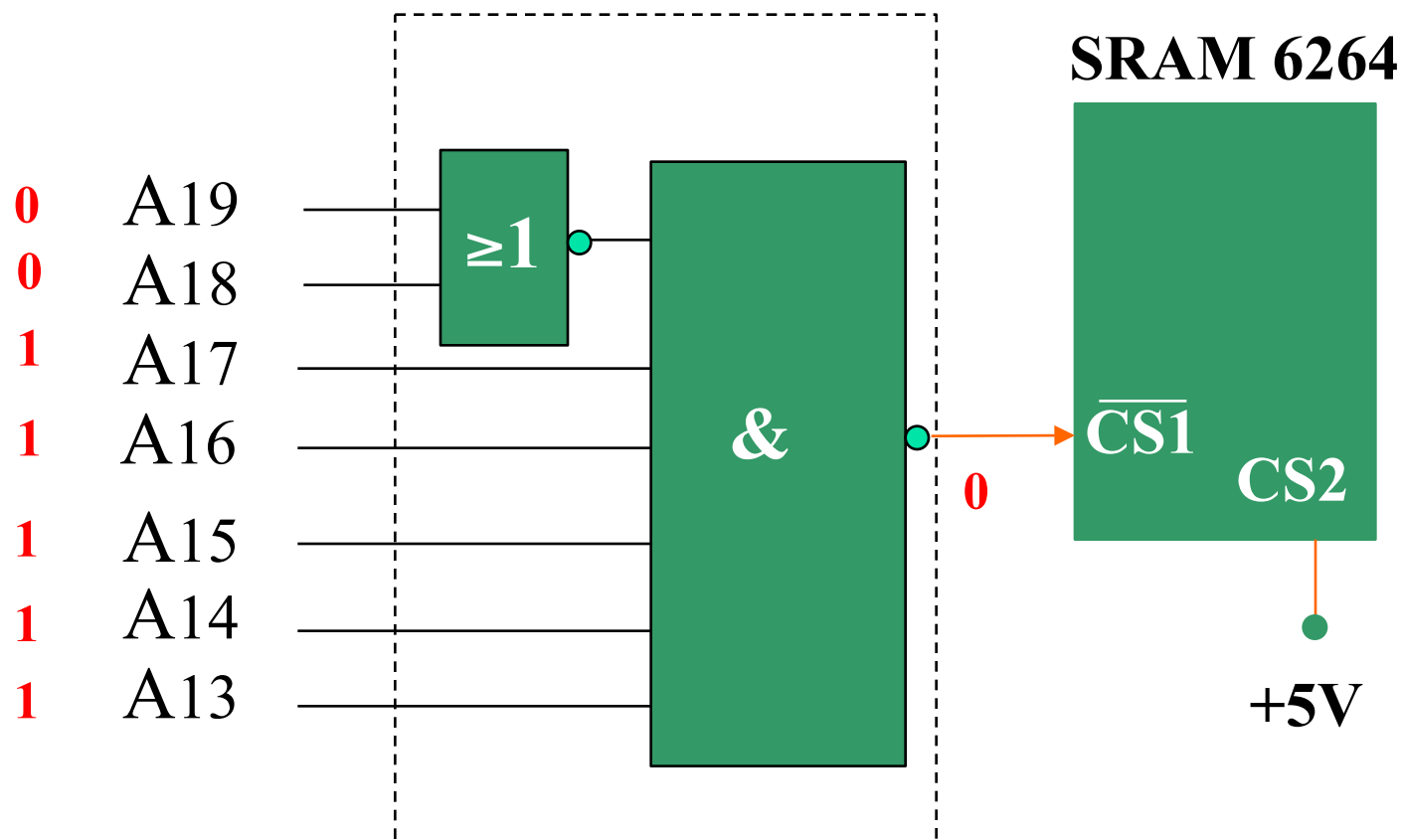
- 设计步骤：
  - 写出地址范围的二进制表示；
  - 确定各高位地址状态；
  - 设计译码器。

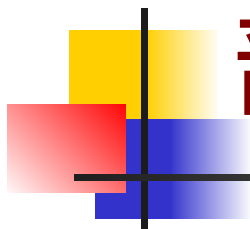




# 全地址译码例

高位地址: **0011111**



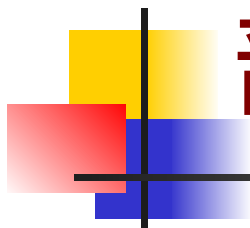


## 部分地址译码

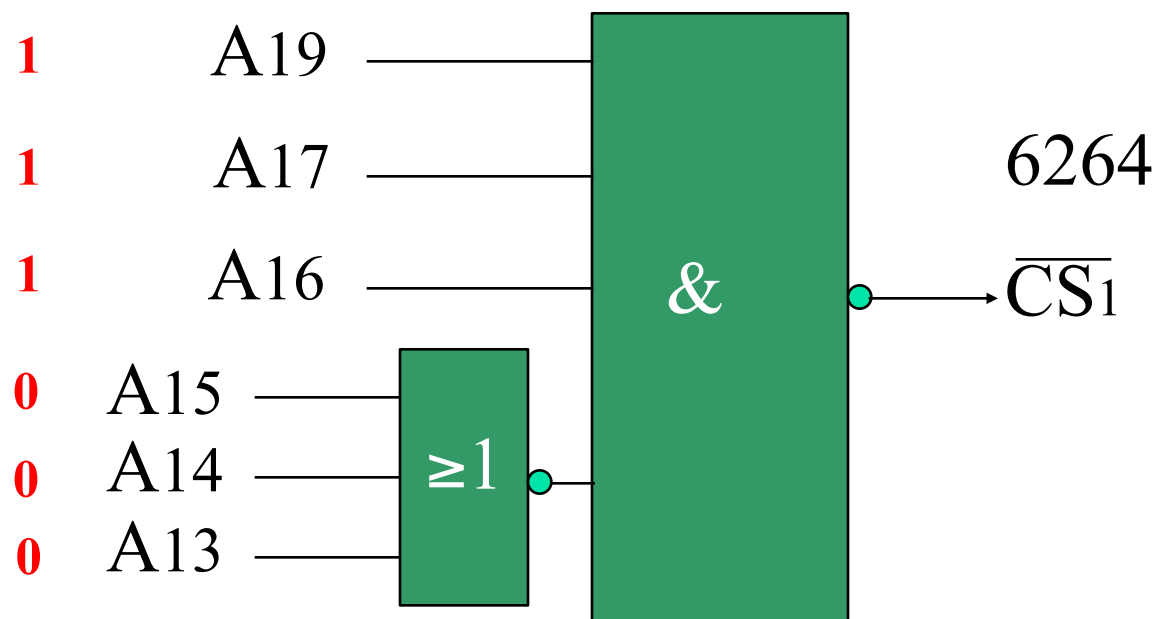
---

- 用部分高位地址信号（而不是全部）作为译码信号，使得被选中存储器芯片占有几组不同的地址范围。





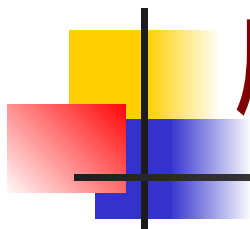
## 部分地址译码例



高位地址:  $1 \times 11000$  ———  $1011000$ ,  $1111000$

两组地址: **F0000H — F1FFFH**

**B0000H — B1FFFH**



## 应用举例

---

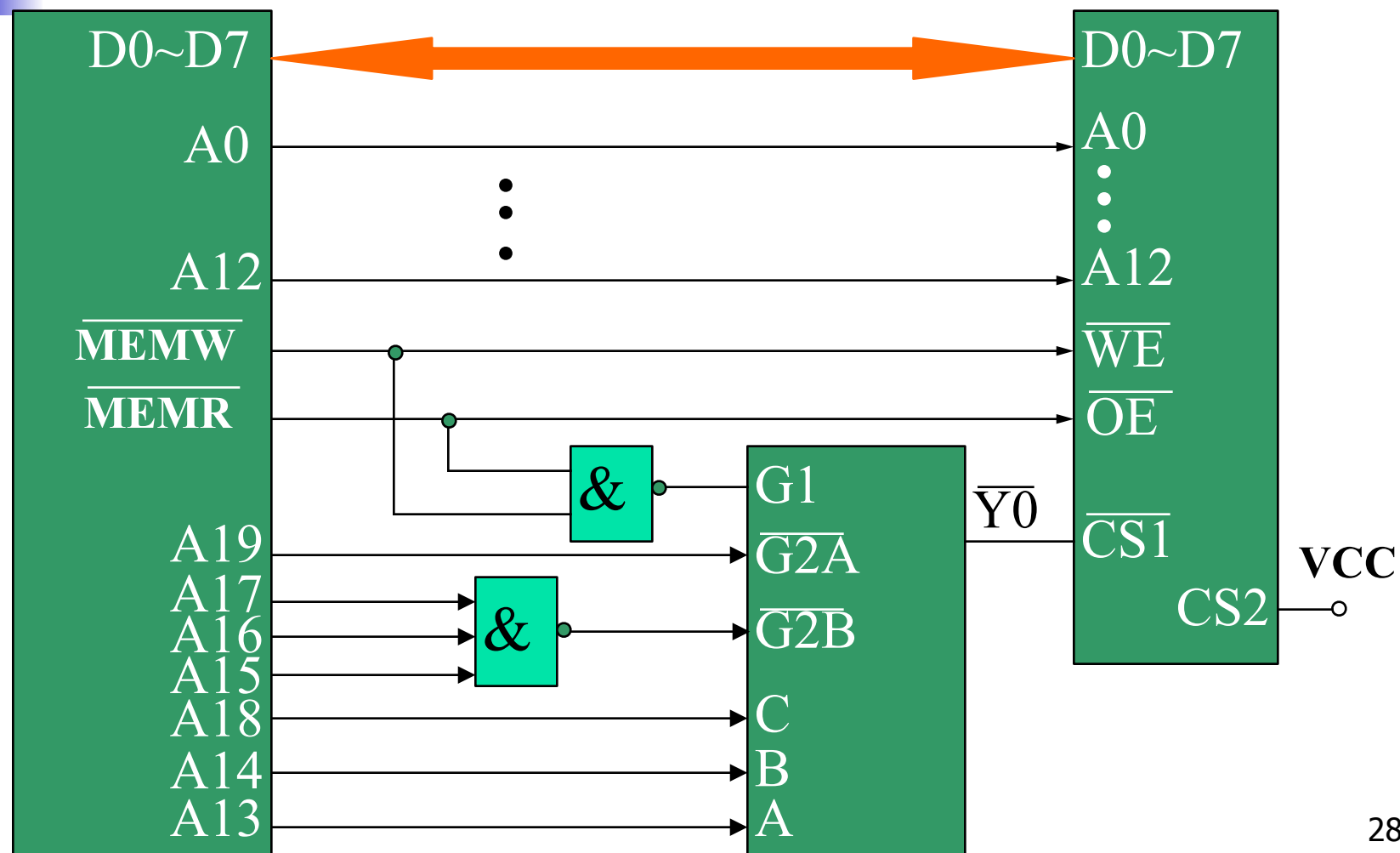
- 将**SRAM 6264**芯片与系统连接，使其地址范围为：**38000H~39FFFH**。
- 使用**74LS138**译码器构成译码电路。



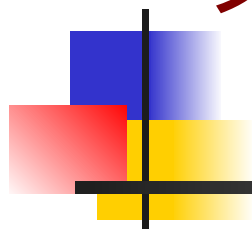
- A19** **A12** **A0**
- 0 0 1 1 1 0 0 0** ... ... **0**

## 高位地址

# 应用举例

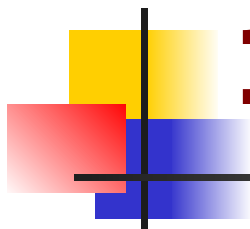


# 第5章 存储器扩展技术



(内存存储器设计)

---



# 1. 存储器扩展

- 用多片存储芯片构成一个需要的内存空间；
- 各存储器芯片在整个内存中占据不同的地址范围；
- 任一时刻仅有一片（或一组）被选中。
- 存储器芯片的存储容量等于：

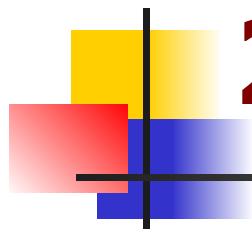
扩展  
单元

单元数 × 每单元的位数

字节数

字长

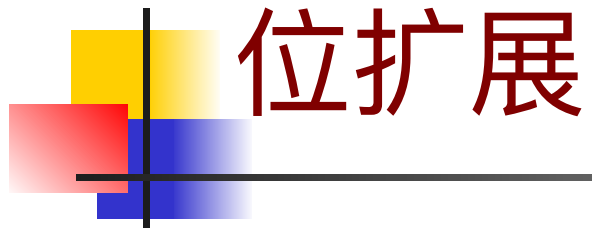
扩展  
字长



## 2. 存储器扩展方法

---

- 位扩展 → 扩展字长
- 字扩展 → 扩展单元数
- 字位扩展 → 既扩展字长也扩展单元数



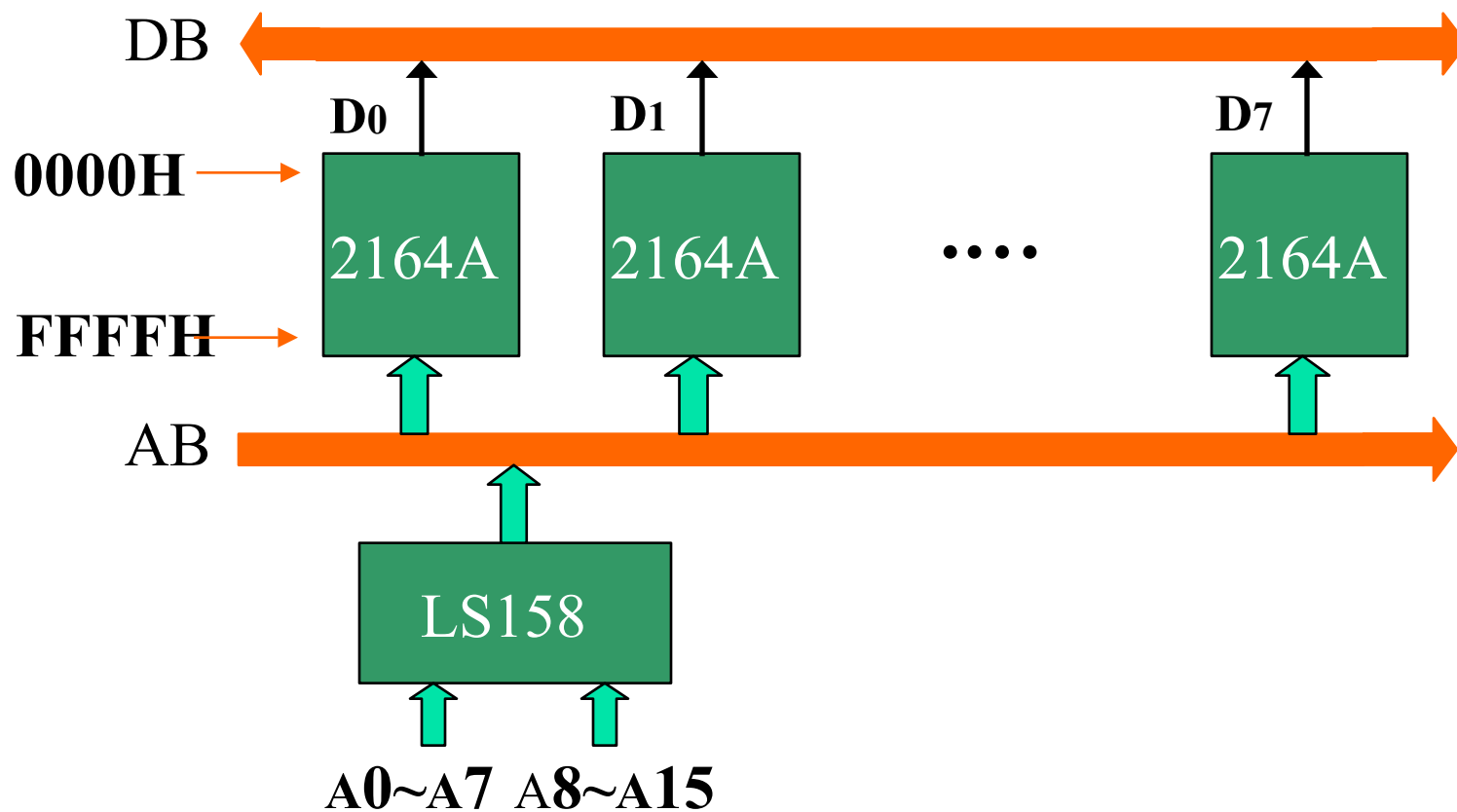
# 位扩展

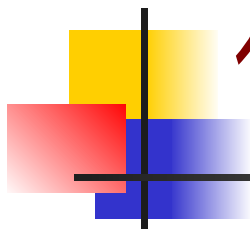
- 构成内存的存储器芯片的字长小于内存单元的字长时——需进行位扩展。
- 位扩展：每单元字长的扩展。



# 位扩展例

- 用8片2164A芯片构成64KB存储器。
  - 64K×1bit

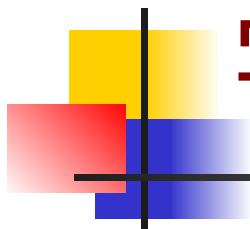




## 位扩展方法：

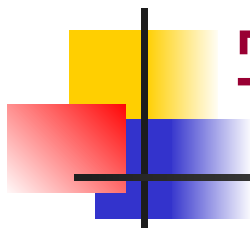
- 将每片的地址线、控制线并联，数据线分别引出。
- 位扩展特点：
  - 存储器的单元数不变，位数增加。

**位扩展：** 确保所有芯片具有**完全相同**的地址范围

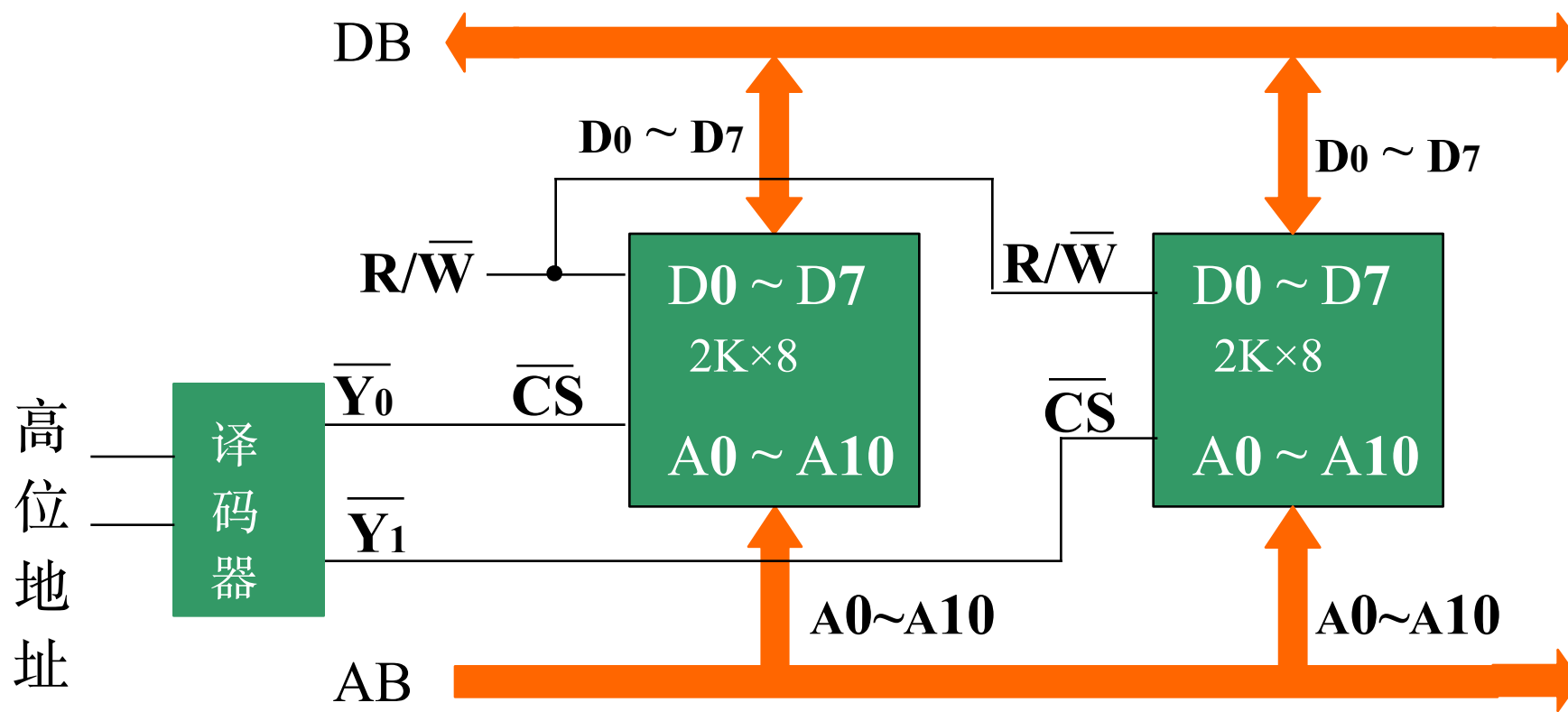


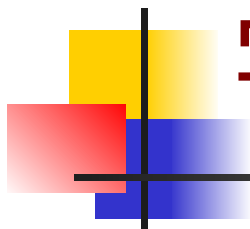
# 字扩展

- 地址空间的扩展
  - 芯片每个单元中的字长满足，但单元数不满足。
- 扩展原则：
  - 每个芯片的地址线、数据线、控制线并联。
  - 片选端分别引出，以使每个芯片有**不同**的地址范围。



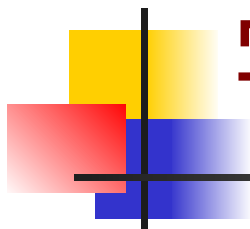
## 字扩展示意图





## 字扩展例

- 用两片64K×8位的SRAM芯片构成容量为128KB的存储器
- 两芯片的地址范围分别为：
  - **20000H~2FFFFH**
  - **30000H~3FFFFH**

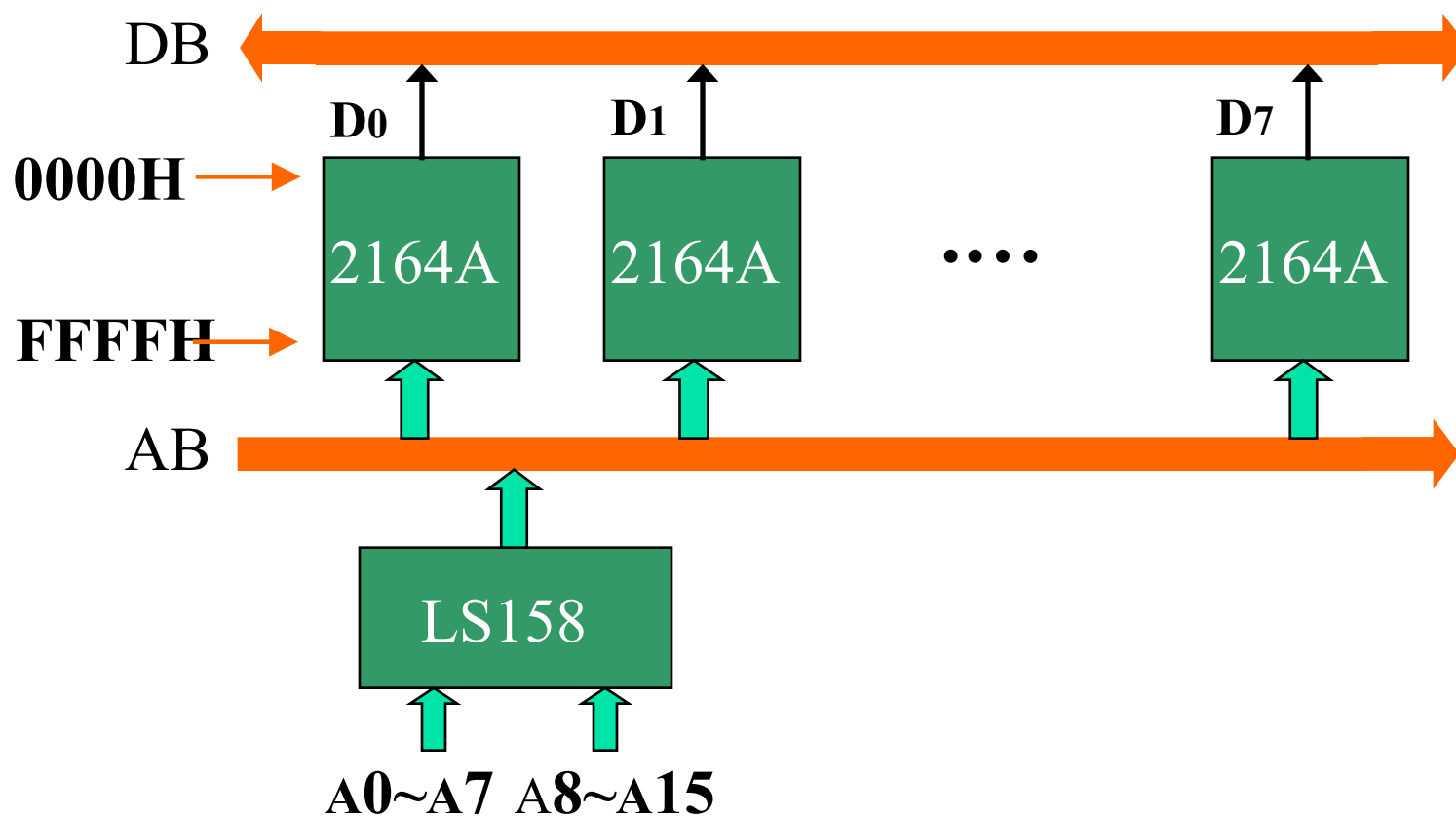


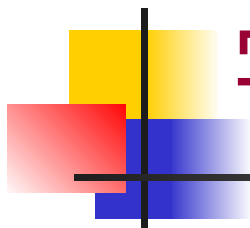
# 字扩展例

- 设计：两芯片的地址范围得
  - 需要2片芯片
- 存储器地址范围：
  - 20000H~2FFFFH  
**0010**0000000000000000~**0010**111111111111
  - 30000H~3FFFFH  
**0011**0000000000000000~**0011**111111111111

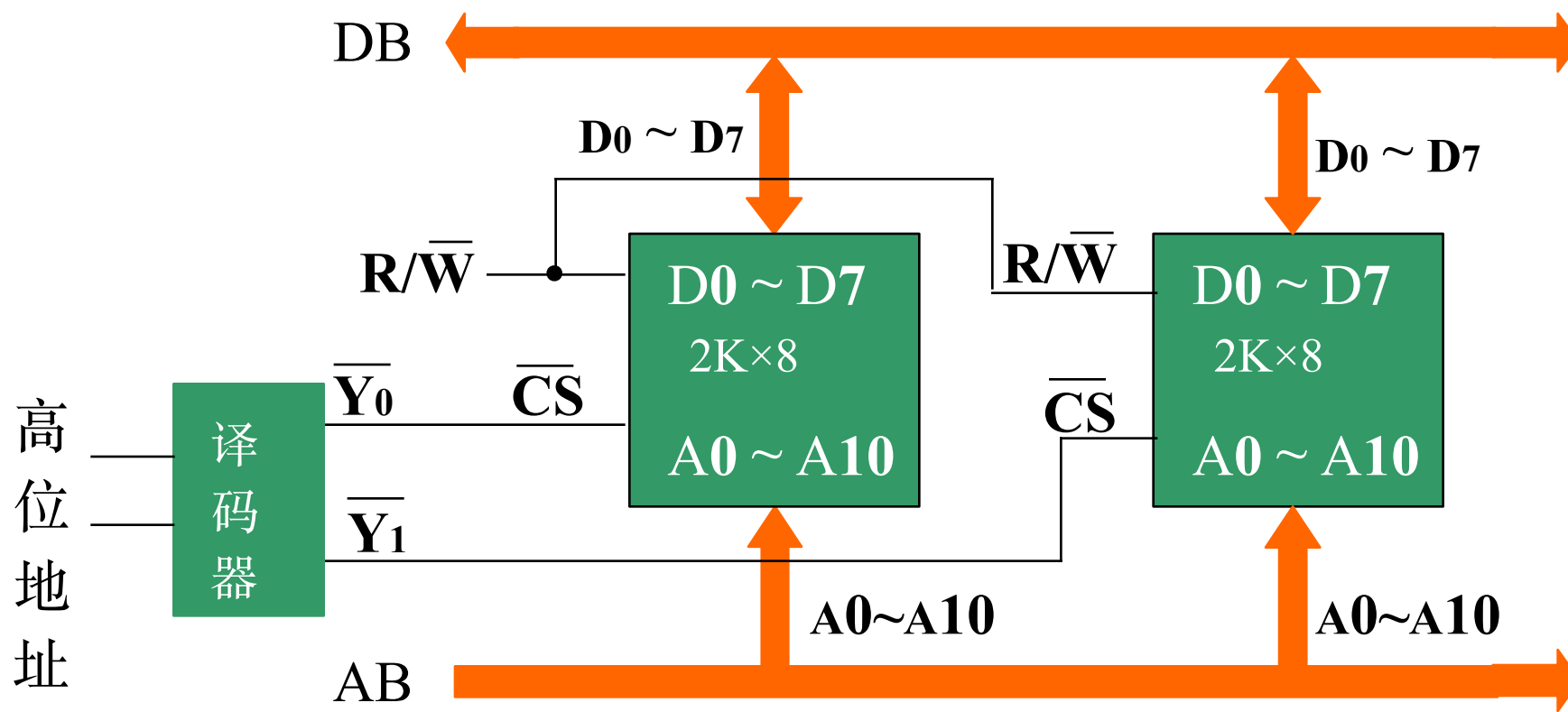
# 存储器位扩展例

- 用8片2164A芯片构成64KB存储器。





## 字扩展示意图







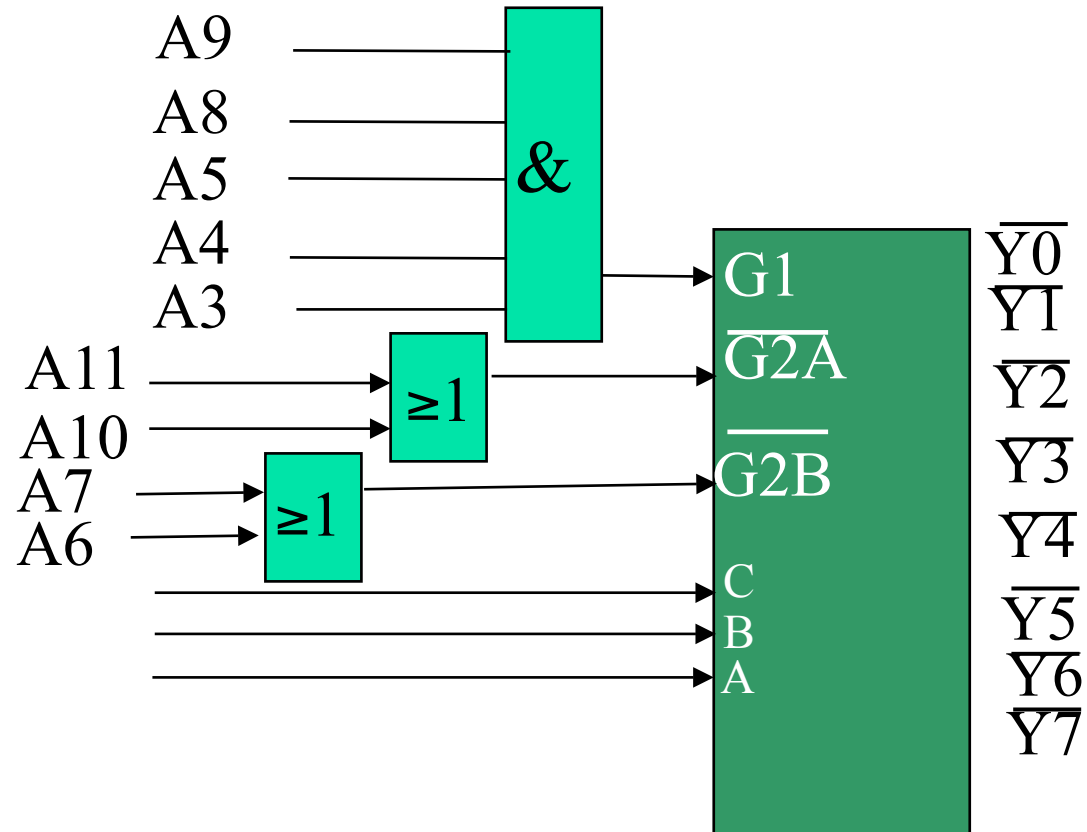
# I/O端口举例

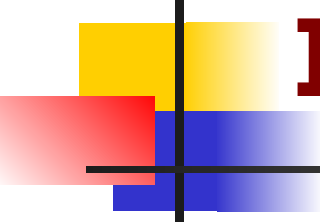
---

- **8088微机系统中，某外设接口所选端口地址为：338H~33FH。**
- **使用74LS138译码器构成译码电路。**



0 0 1 1 0 0 1 1 1    0 0 0





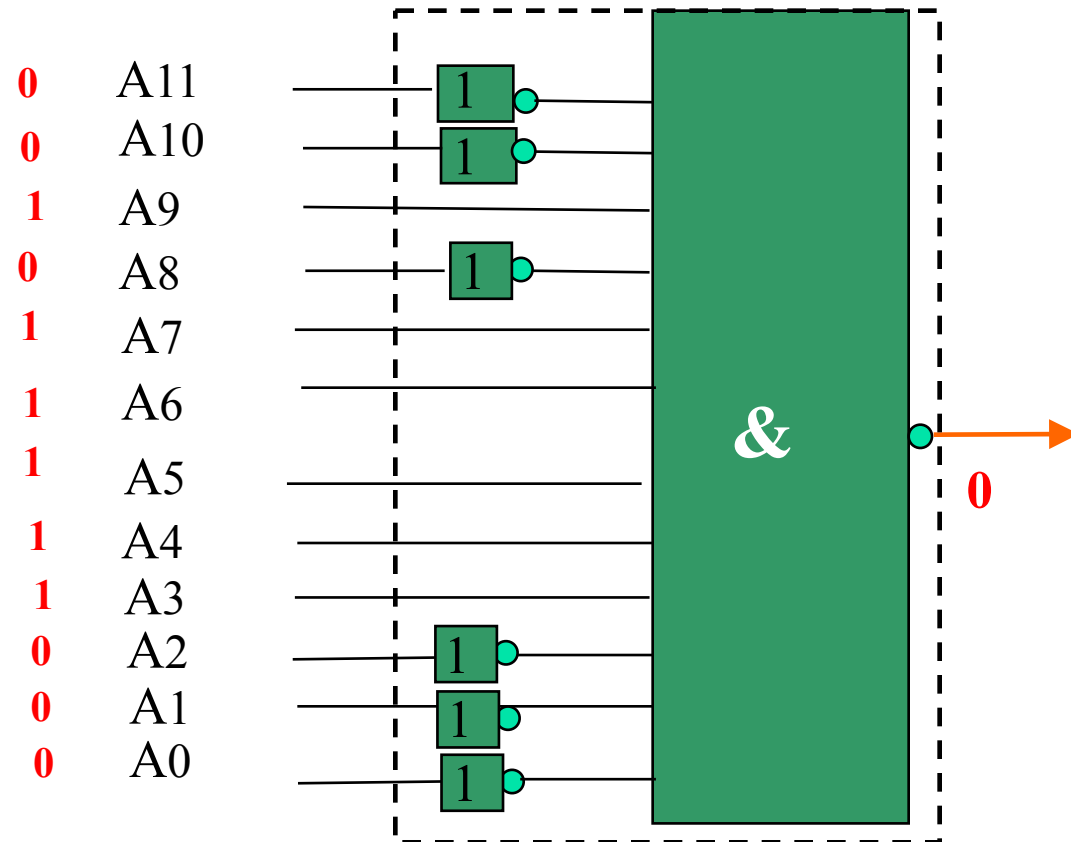
# I/O地址译码例

---

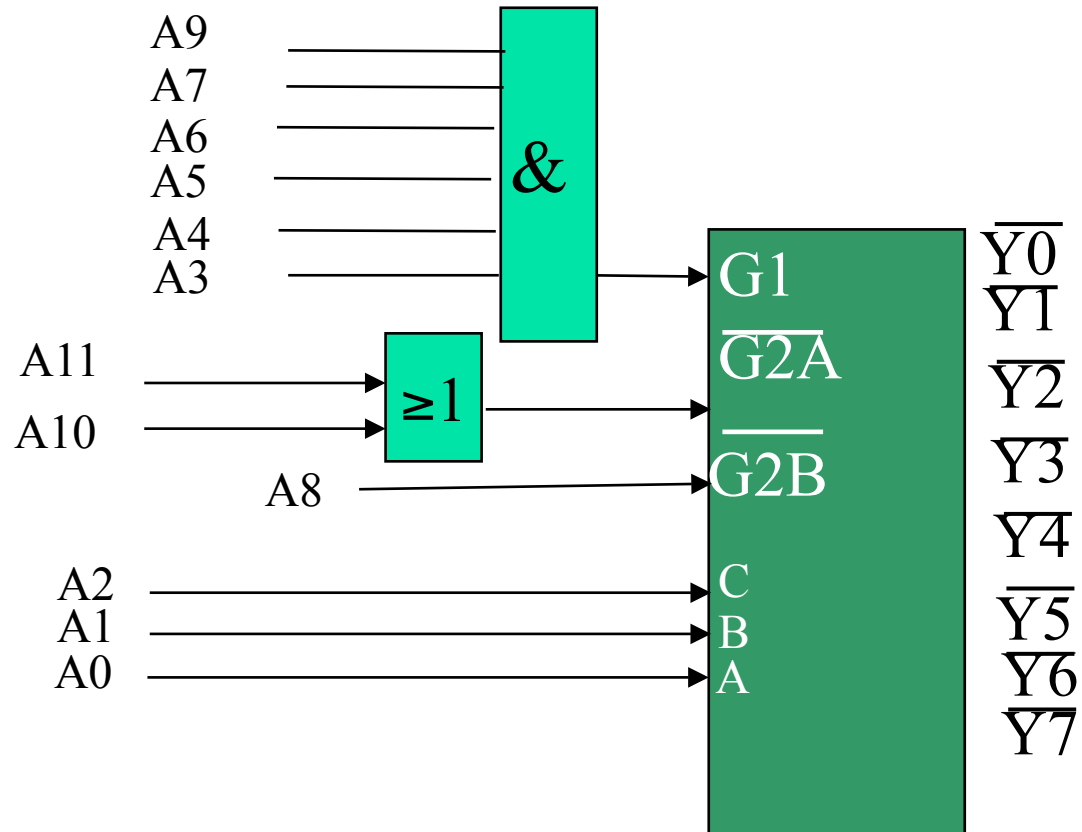
- 某外设接口有个端口，地址为**2F8H**，设计进行读写译码电路

# 译码

**2F8: 001011111000B**



0 0 1 0 1 1 1 1 1 0 0 0



# 8253应用例

- 采用8253作定时/计数器，其接口地址为0120H~0123H。
- 输入8253的时钟频率为2MH。要求：
  - CNT0每10ms输出一个CLK周期宽的负脉冲
  - CNT1输出10KHz的连续方波信号
  - CNT2在定时5ms后产生输出高电平
- 画线路连接图，并编写初始化程序。

启动方式

工作方式

计数脉冲频率



# 8253应用例

---

- 计算计数初值:

**CNT0:  $10\text{ms}/0.5\mu\text{s}=20000$**

**CNT1:  $2\text{ MHz}/10\text{KHz}=200$**

**CNT2:  $5\text{ms}/0.5\mu\text{s}=10000$**

- 确定控制字:

**CNT0: 方式2, 16位计数值**

————→ **00110100**

**CNT1: 方式3, 低8位计数值**

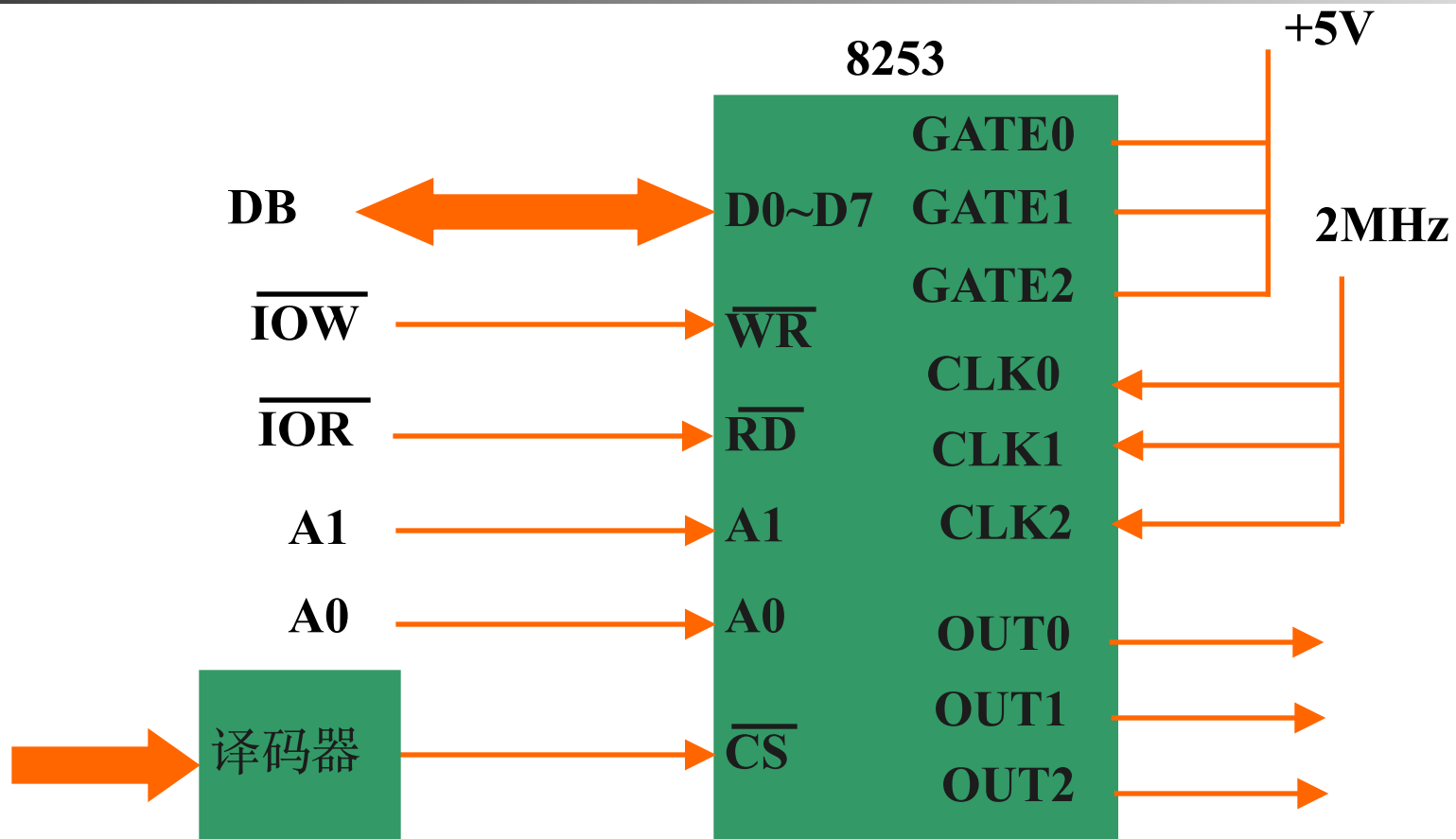
————→ **01010110**

**CNT2: 方式0, 16位计数值**

————→ **10110000**



# 8253应用例





# 8253应用例——初始化程序



**CNT0:**

**MOV DX, 0123H**

**MOV AL, 34H**

**OUT DX, AL**

**MOV DX, 0120H**

**MOV AX, 20000**

**OUT DX, AL**

**MOV AL, AH**

**OUT DX, AL**

**CNT1:**

**.....**

**CNT2:**

**.....**