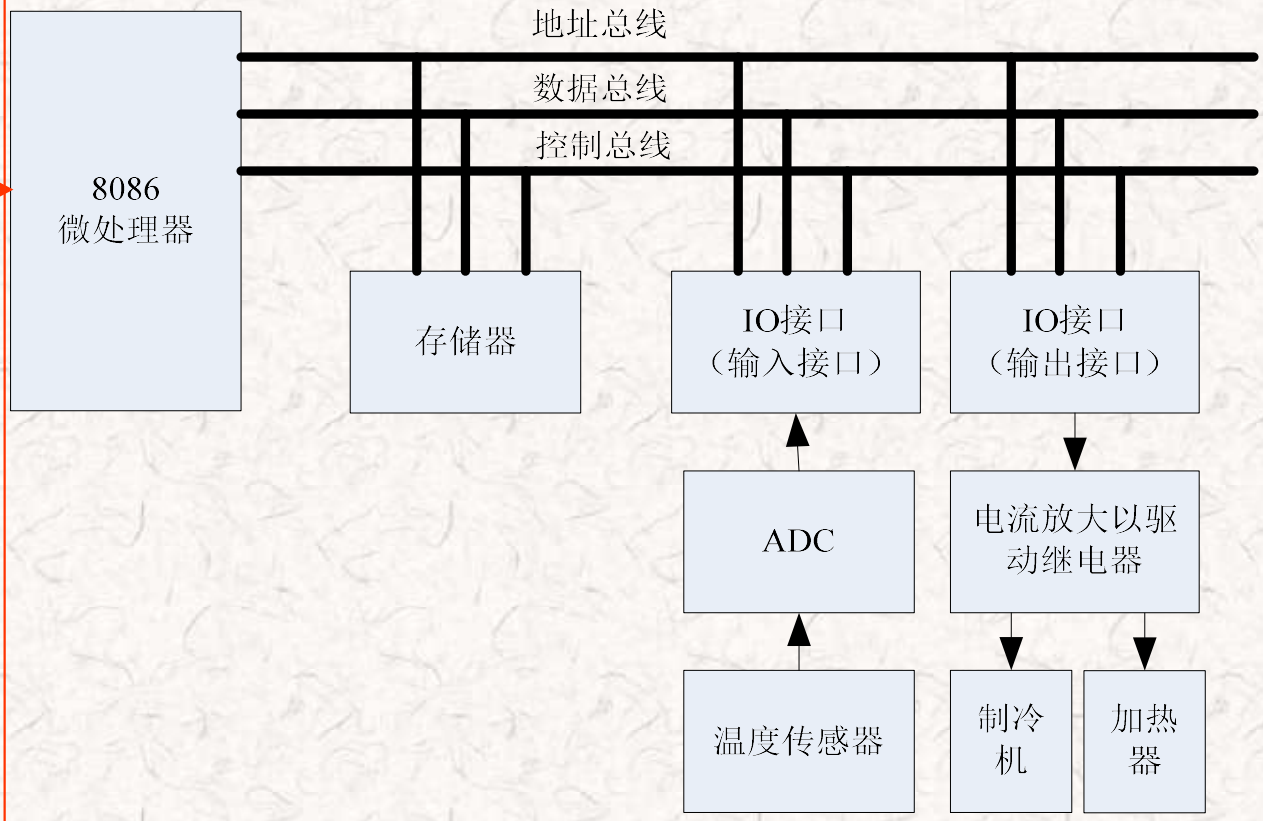
微机原理与接口技术

第二章 8086系统结构

课程内容之8086微处理器结构

第2章
8086CPU和系统的整体介绍

8086功能与外部总线
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？
存储器/IO的总线周期是怎样的？

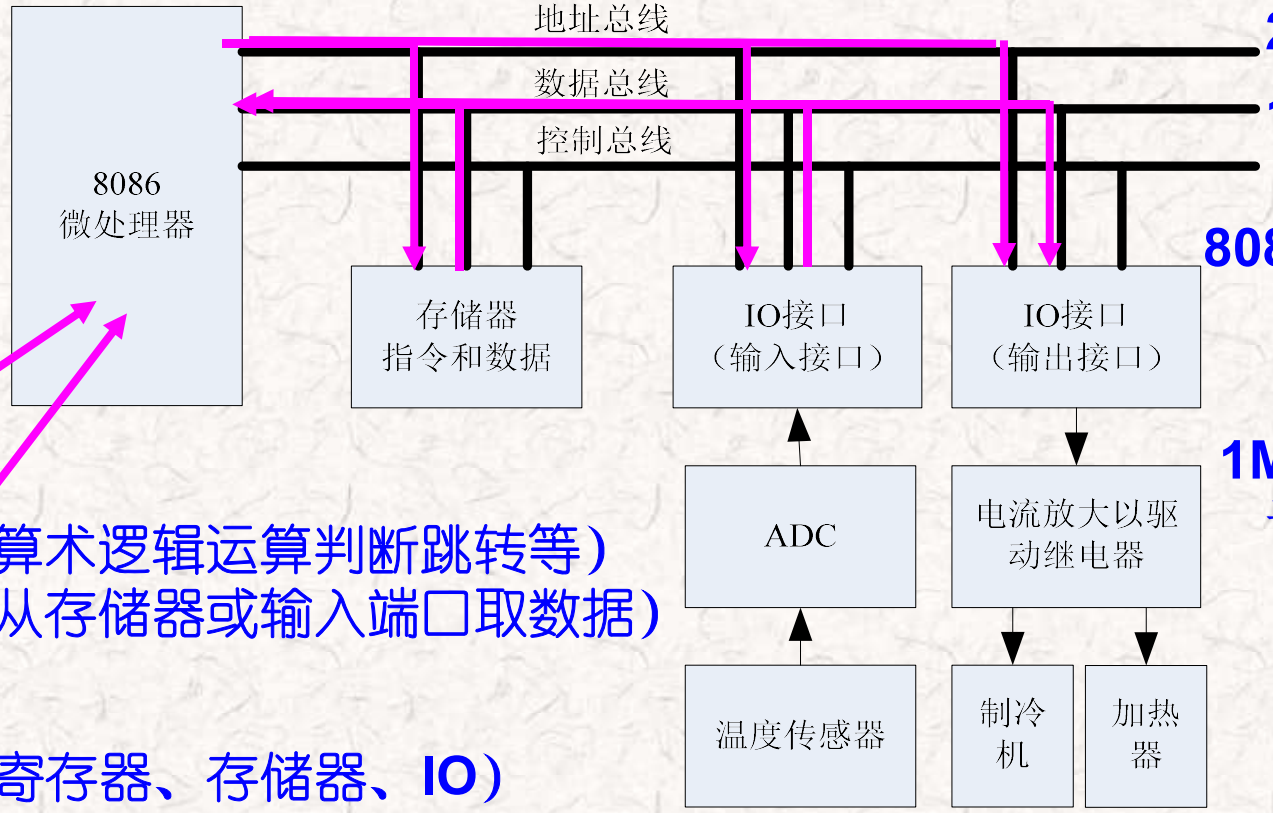


1.8086内部结构

- **8086CPU**内部分为几部分？各自的功能是什么？为什么这样设计？
- **8086**指令队列有多少字节？为什么要设置指令队列？
- **8086**地址总线有多少根？
- **8086**内部数据总线多少位？外部数据总线多少位？为什么说**8086**是真**16**位机？
- **ALU**能实现什么功能？

带着上述问题阅读课本 第二章引言及2.1.1节

8086外部总线及功能



20根地址线

16根数据线

8086地址空间是1MB

为什么不是1M*16bit呢？请关注8086存储系统

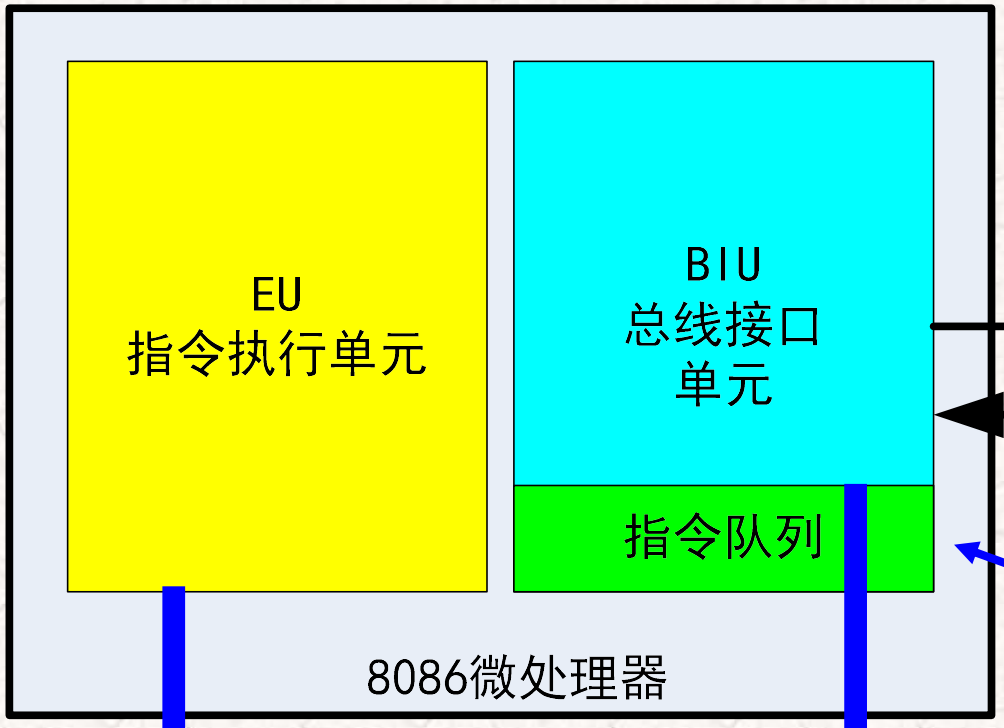
取指

分析指令

执行指令（算术逻辑运算判断跳转等）
（可能需要从存储器或输入端口取数据）

输出结果
（可能写回寄存器、存储器、IO）

8086结构框图



分时复用（最小系统的总线生成部分会有介绍）

16bit数据总线

20bit地址总线

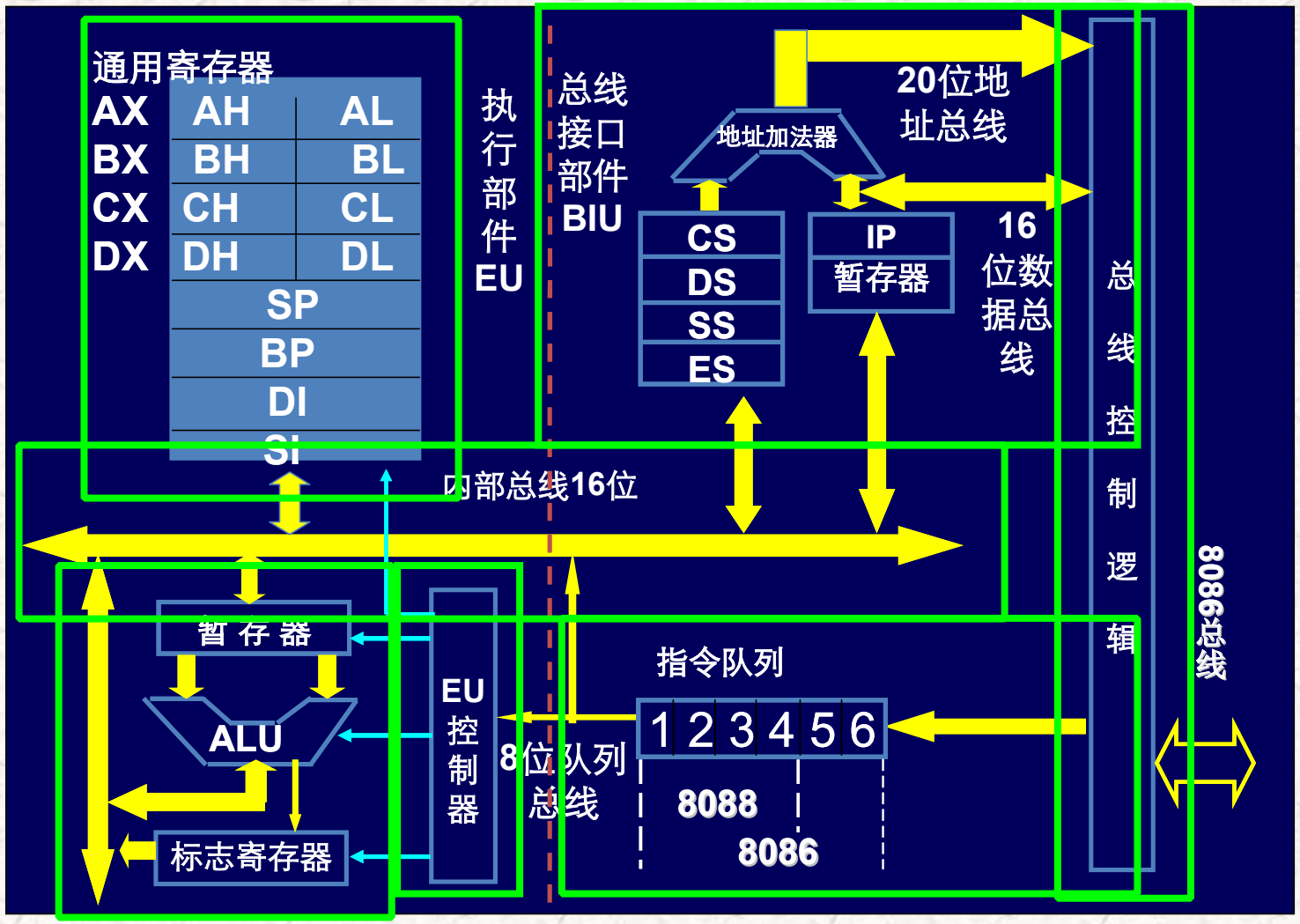
控制信号

通过指令队列实现并行工作

执行指令，包含：
（1）指令译码逻辑
（2）算术逻辑单元
（3）寄存器

读指令，读/写数据，包含：
（1）地址计算
（2）总线信号生成

认识8086内部几大模块



2.8086内部寄存器

- 寄存器的用途？**8086 CPU**内部数据寄存器有些？
- 地址指针和变址寄存器有哪些？作为地址指针的时候，寄存器存放的值表示什么？
- **BX、BP、SI、DI、SP**各是什么寄存器？
- **CS、SS、DS、ES**各是什么寄存器？
- 寄存器**IP**的用途？程序员可以直接给**IP**寄存器赋值吗？
- 标志寄存器中有哪些标志位和程序控制位？

带着上述问题阅读课本 2.1.2节

寄存器

寄存器用触发器或者锁存器构成，用于存放nbit数据

地址指针寄存器

- 寄存器在汇编指令中的两种使用方式：
- 使用方式1：将寄存器中的数据直接拿来运算
- 若BX=0，ADD AX, BX 表示 AX + 0 -> AX
- 使用方式2：寄存器中存放的数据表示的是存储单元的地址，参与运算的是该地址指向的存储单元中的数据
- 若BX=0，并且假设地址为0的存储单元中存放数据为100，则ADD AX, [BX] ，表示：AX + 100-> AX
- 8086汇编指令中可作为地址指针使用的寄存器有哪些？
- SI、DI、BX、BP

标志寄存器中的标志位

- 标志位的用途
- 记录**ALU**运算的结果的状态：例如运算结果的正负、运算是否溢出，是否产生进位，等
- 高级语言中的循环、分支都是通过判断运算标志实现的
- 循环：**for (i=0; i<50 ; i=i++)**

  ……
- 比较 **i** 和**50**（执行**i-50**，但不修改 **i)**，判断结果是否为负
- 分支：**If**（**score > 90)**

  ……

  比较**score**和**90**（执行**score-90**，但不修改 **score**），判断结果是否为正，且不等于**0**

标志寄存器中的标志位和控制位

- ZF -- zero flag， ALU运算结果全零，则为1
- SF -- sign flag， ALU运算结果最高位D7或D15为1，则为1
- CF -- carry flag， 有进位或者借位，则为1
- OF -- overflow flag， 后面详解
- PF -- parity flag， ALU结果低8位有偶数个1，则为1
- AF -- auxiliary carry flag， D3向D4有进位或借位，则为1（最低位为D0）

- IF -- interrupt flag， IF为1，允许响应可屏蔽外中断请求
- TF -- trap flag， TF为1，每条指令执行后均进入单步中断
- DF -- direction flag， DF为1，串操作指令每次地址减少

溢出标志位OF判断标准

- 如果为**8**位数（**D7:0**）加减运算，判断 **D6 -> D7**的进位或者借位，以及 **D7 -> CF** 的进位或者借位：
- 二者均有，或者均没有，则**OF=0**；
- 一个有，一个没有，则**OF=1**；
- 如果为**16**位（**D15:0**）加减运算，判断 **D14 -> D15**的进位或者借位，以及 **D15 -> CF**的进位或者借位：
- 二者均有，或者均没有，则**OF=0**；
- 一个有，一个没有，则**OF=1**；
- 举例：

```
  1111 1111        0111 1111
+ 1111 1111      + 0111 1111
-----------      -----------
1 1111 1110      0 1111 1110
   OF=0             OF=1
```

加减运算的溢出判断

- 什么是溢出？运算结果超出了数据的表示范围
- 以**8bit**为例，
- 无符号数表示范围 **0~255　（0~0FFH）**
- 有符号数补码表示范围 **-128~127　（80H~7FH）**
- 溢出和溢出标志位**OF**是不同的含义
- 用于无符号数加减运算判断溢出的标志位：**CF=1**
- 用于有符号数加减运算判断溢出的标志位：**OF=1**

```
  1111 1111
+ 1111 1111   CF=1
-------------  OF=0
1 1111 1110
```

```
  0111 1111
+ 0111 1111   CF=0
-------------  OF=1
0 1111 1110
```

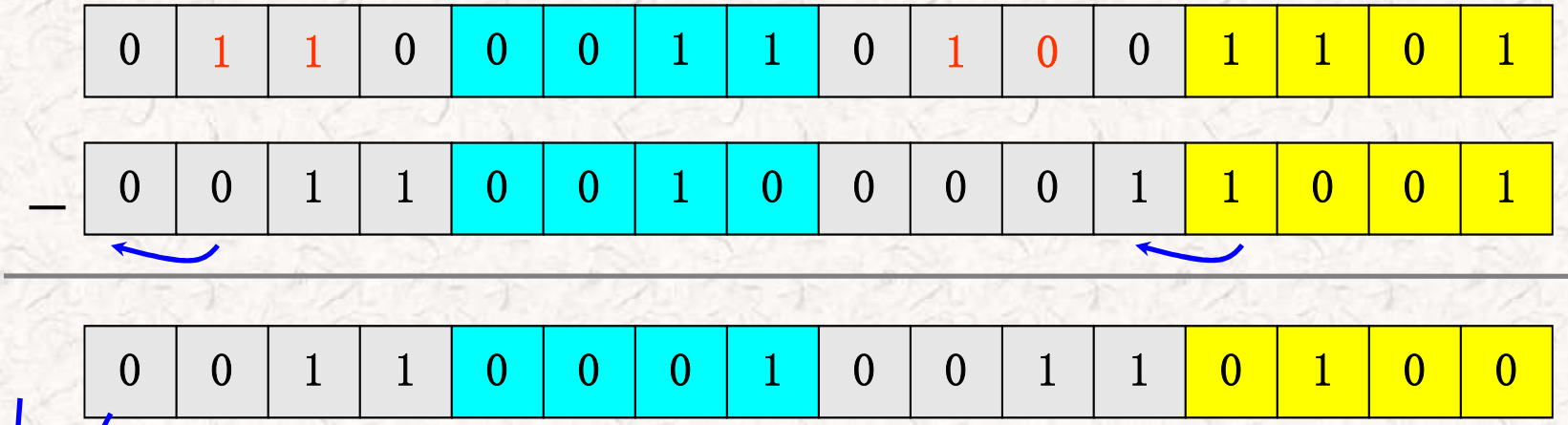
有符号数溢出会发生什么？

- 分析：次高位、最高位（符号位）与进位位之间连续的进位或者借位不会改变正确的符号位的运算结果，否则，符号位被修改，发生错误
- 结论：两个有符号数相加减，如果溢出将影响结果最高位，即符号位，使符号位取反
- 例如：两个数相加，结果应为正数的情况
- **7F+2=81H，127+2=-127，错误！正数变成负数**
- 例如：两个数相减，情况应为负数的情况
- **80H-01H=7FH，-128-1=127，错误！负数变成正数**
- 延伸思考：两个8位数相加减，结果最少用多少位表示不会溢出？四个8位数相加减呢？8个16位数相加减呢？
- 再思考：对结果位宽扩展还是对加数位宽扩展？如何扩展？

如何进行位扩展？

- 方式1：对结果进行扩展
- 举例：**8bit + 8bit = 9bit**
- 等价于 **{1'b0,8bit} + {1'b0,8bit} = 9bit**
- 仅适用于无符号数相加（增加位宽不影响数值）
- 方式2：对加数进行扩展
- 无符号数相加：高位扩充**0**
- 有符号数相加：高位扩充符号位
- 举例：将**8**位有符号数 **-1** 扩充为**16**位有符号数 **-1**
- **1111 1111 B → 1111 1111 1111 1111 B**（数值不变）

标志位练习1



SF
ZF
PF
CF
AF
OF

运算结果最高位为0　　∴SF=0;

运算结果本身≠0　　∴ZF=0;

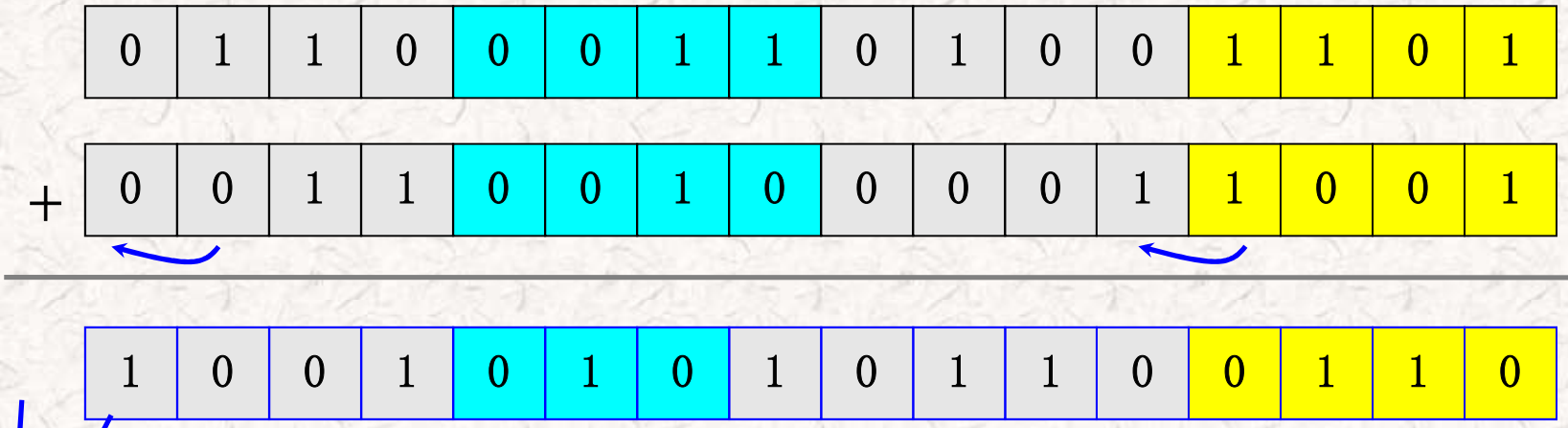
低8位中1的个数为奇数个　　∴PF=0;

最高位没有借位　　∴CF=0;

第三位向第四位没有借位　　∴AF=0;

次高位向最高位没有借位，最高位向前没有借位，OF=0

标志位练习2



SF
ZF
PF
CF
AF
OF

运算结果最高位为1　　∴SF=1；

运算结果本身≠0　　∴ZF=0；

低8位中1的个数为偶数个　　∴PF=1；

最高位没有进位　　∴CF=0；

D3向D4有进位　　∴AF=1；

次高位向最高位有进位，最高位向前没有进位，∴OF=1

使用标志位需要区分有符号 or 无符号

- **ALU**：二进制数加减，不区分有无符号（物理概念）
- 程序员：决定有符号数 **or** 无符号数（逻辑概念）
- 有符号数--补码
- 有符号位宽扩展方法（**CBW**，**CWD**）
- 有符号数运算指令
  - 加减指令：**ADD, SUB, ADDC, SBB**
  - 乘除指令：**MUL, IMUL, DIV, IDIV**
- 有符号数标志判断和转移指令（选择正确的标志位）

标志位的应用之：循环举例

- 举例：循环次数控制 **for i=1; i<50; i++**
- 通过 无符号数大小比较 + 标志判断 来实现
- 用某寄存器存放 i 值，如**AL**：
- **MOV AL, 1** ；初始化，**AL**赋值**1**
- **labelX:** 循环体
- **INC AL** ；**AL**的值加**1**
- **CMP AL, 50** ；比较**AL** 与**50**的大小
- **JC lableX** ；如果小于（**CF=1**），则跳转到**lableX**， **lableX**为循环体开始的指令标号

标志位的应用之：比较大小

- 如何判断两个数相等？
- **ZF**　形式举例：**CMP AX,BX　　JZ xxx**
- 如何判断两个有符号数加减运算后结果是正数还是负数？
- 如无溢出，根据**SF**；如有溢出，根据**SF**的非
- 如何判断两个数相加后是否溢出？
- **CF 或者 OF**
- 如何比较两个无符号数大小？
- 如何比较两个有符号数大小？

两个无符号数相减
如果ZF=1，相等；
否则，
CF=0，无借位，被减数大
CF=1，有借位，减数大；

两个有符号数相减
如果ZF=1，相等；
否则
如OF=0，SF=0，结果为正，被减数大；
　　SF=1，结果为负，减数大；
如OF=1，SF=1，结果为正，被减数大；
　　SF=0，结果为负，减数大；

标志位的应用之：位判断及其它

- 如何判断寄存器（例如AL）中某一位（例如D0）为 0 或者为 1 ？
- 方法1：右移 1 位，判断 CF
- 方法2：与0000 0001B 相与，判断 ZF
- 其它1 顺序执行,则　MOV AL, 3
- 　　SUB AL, 4　CF= ？　1
- 　　ADD AL, 4　CF= ？　1
- 其它2 如何对字节（假设放在AL寄存器中）生成偶校验位（假设放在BL中）？
- 利用 PF
- 其它3 如果允许CPU响应外中断，应如何操作
- （1）令TF=1，（2）令ZF=1，（3）令IF=1，（4）令DF=1
- 3　STI

```
SHR AL,   1
JC    Label1
```

```
AND AL,   01H
JNZ  Label1
```

```
MOV  BL,     0
ADD   AL,     0
JP      SETBL0
MOV  BL,     1
```

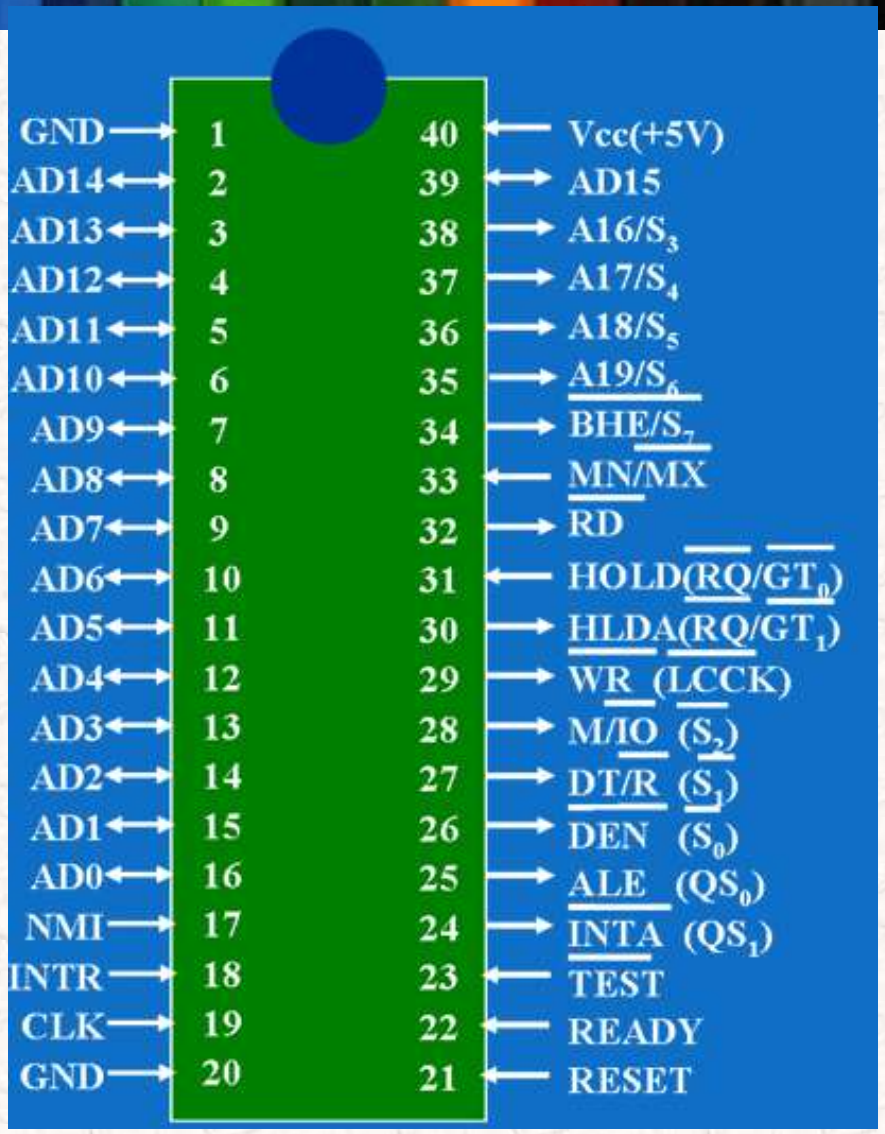
3.8086引脚功能

- **8086**有多少引脚？
- **AD0~AD15**的含义？
- 为什么数据总线和地址总线复用？二者如何区分的？与哪个信号相关？
- 什么是最小模式，最大模式？
- 说明信号定义**$\overline{\text{BHE}}$、$\overline{\text{RD}}$、$\overline{\text{WR}}$**
- **INTR、INTA、NMI**
- **RESET、CLK、 M/$\overline{\text{IO}}$**
- **$\overline{\text{DEN}}$、DT/$\overline{\text{R}}$**
- **MN/$\overline{\text{MX}}$**

带着上述问题阅读课本 2.1.3

8086引脚

- **40 pin DIP封装**
- 分时复用的引脚
- 最大模式与最小模式复用的引脚



复位RESET-系统一切工作的起点

寄存器	初始状态	寄存器	初始状态
状态标志寄存器	清0	IP	0000H
CS	FFFFH	DS	0000H
SS	0000H	ES	0000H
指令队列寄存器	清空	其他寄存器	0000H

- 复位后第一条指令：CS:0FFFFH IP:0000H → 0FFFF0H
- 标志寄存器：0000H，即禁止中断和单步方式

4.8086的存储器组织

- 8086可寻址空间是多大？每一个地址能够存放的数据位宽是多少？
- 8086存储空间为什么要分逻辑段？一个逻辑段的大小？
- 8086的逻辑段有几种类型？用途？段寄存器？
- 各个逻辑段之间相对位置怎样？
- 什么是逻辑地址？什么是物理地址？
- 逻辑地址如何转成物理地址？
- 说一说寄存器和存储器的区别，以及8086如何对二者进行访问

阅读课本 2.1.4

逻辑地址与物理地址

物理地址：
送到地址总线上的**20bit**地址

8086在与存储器进行实际
数据读写时使用物理地址

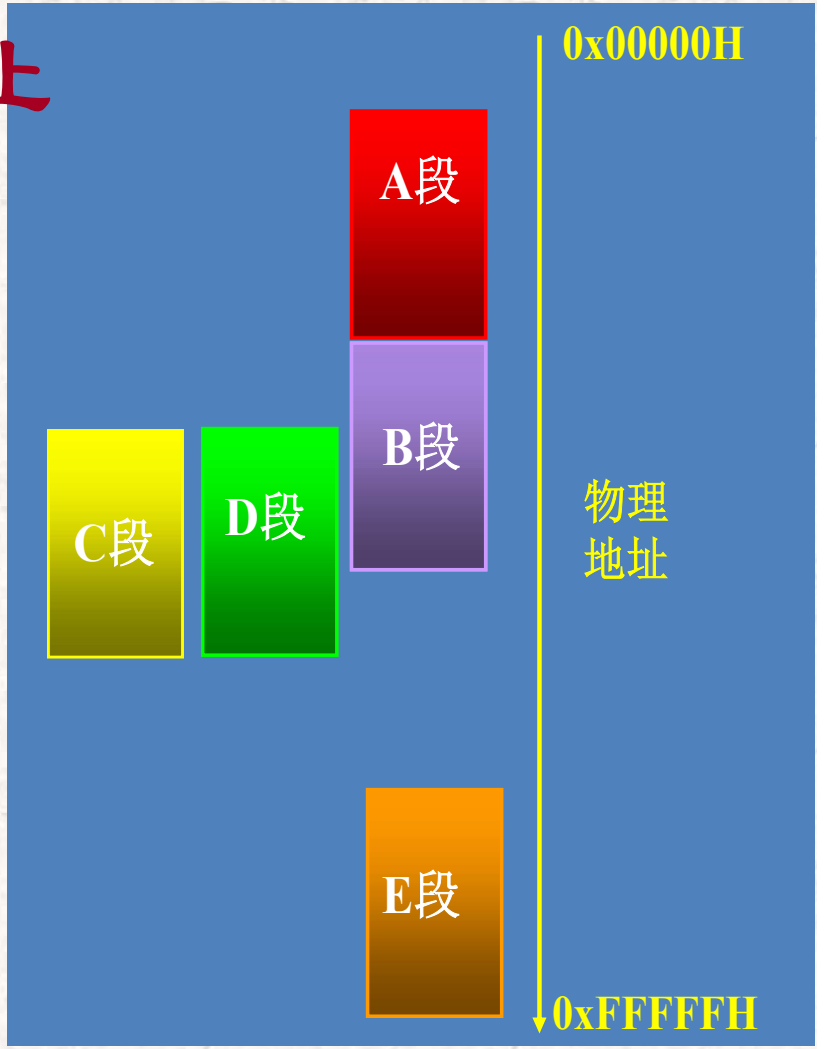
逻辑地址：段首地址，段内偏移地址

程序中使用逻辑地址
（受限16bit数据位宽）

物理地址=段首地址*16+段内偏移地址

BIU进行上述计算

存放之处：段寄存器　与段和寻址方式有关



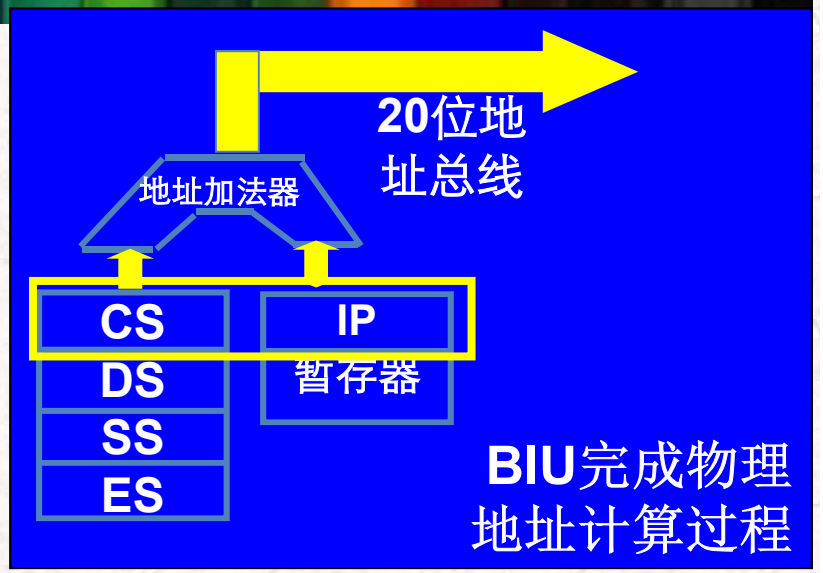
举例：取指时逻辑地址与物理地址的换算

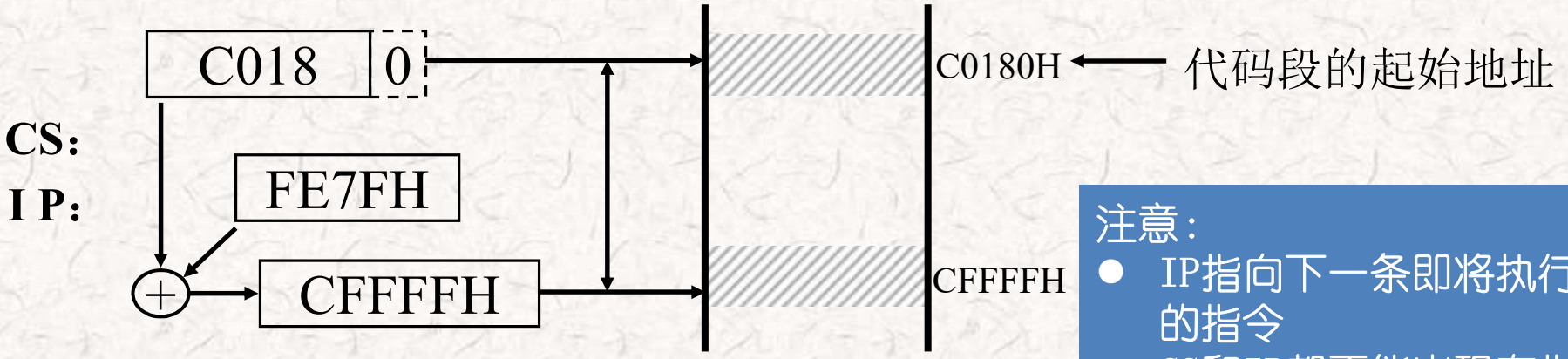
求出某指令所在的存储单元

代码段寄存器 CS=C018H，

段内偏移量 IP=FE7FH，

则该存储单元的实际地址是CFFFFH





即 C018 0 H+FE7F H=CFFFFH

注意：

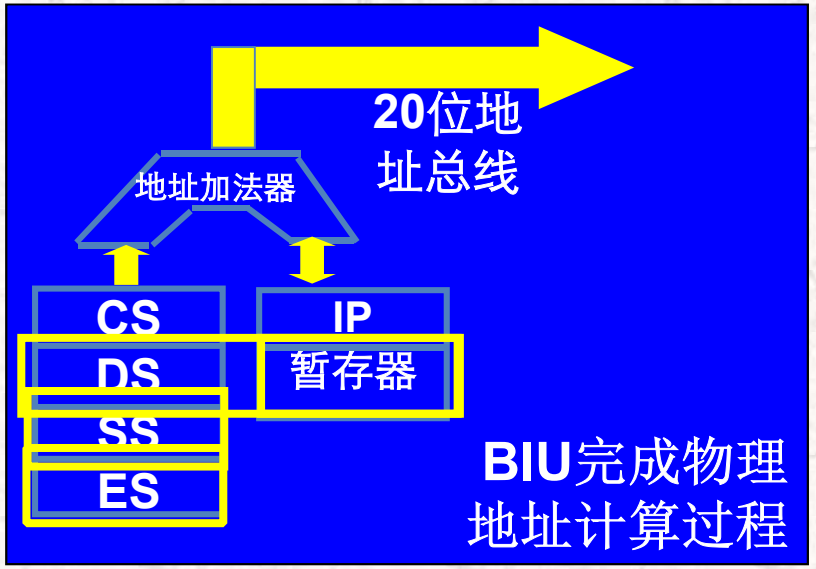
- IP指向下一条即将执行的指令
- CS和IP都不能出现在指令中

举例-数据访问物理地址计算

- 段寄存器：DS 或 ES
- 举例：若期望读取20100H单元的字节到AL
- MOV AX,2000H ；立即数赋值
- MOV DS,AX ；数据段寄存器赋值
- MOV BX,100H ；
- MOV AL, [BX] ；

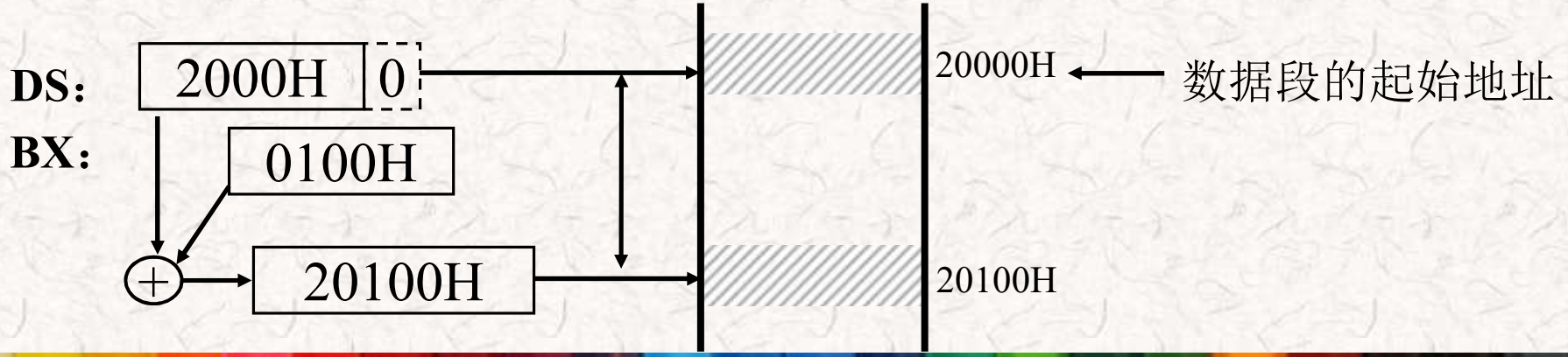
MOV AL,[100H]亦可

- 物理地址=2000H*10H+100H=20100H



存储器

进出栈操作使用SS和SP



物理地址与可能对应多个逻辑地址

例如，物理地址**01245H**可以从两个不同的相互重叠的段中得到，其中一个段的基址是**0123H**，另一个段的基址是**0124H**

8086的地址空间与逻辑段总结

- 存储空间：总**1MB**，在逻辑上采用 段 + 偏移地址 方式进行访问
- 段地址：存放于寄存器，分别是 **CS、SS、DS、ES**
- 偏移地址：
- 取指： **IP**--指令专用
- 堆栈： **SP**--堆栈专用
- 数据：由指令中的寻址方式决定
- 如果需要访问的存储空间超过**64KB**怎么办？
- 重新给段寄存器赋值

- **IO**空间： **64KB**
- 无需分段管理，指令中可以直接指明**IO**端口地址

- 如何区分**8086**总线上进行的是存储器访问还是**IO**访问？

通过**M/IO**信号进行区分

练习1-段寄存器和地址指针寄存器

- 代码段用来存放_____,访问代码段的段地址和偏移地址分别存放在_____和_____寄存器中
- 程序指令的机器码，CS，IP
- 数据段用来存放_____,访问数据段的段地址存放在_____中，偏移地址除指令中直接给出外，还可以存放在以下寄存器中_____
- 数据，DS，SI、DI、BX、BP（前需加 DS:）
- 堆栈段用来存放_____,访问堆栈段的段地址存放在_____中，偏移地址可以存放在____和____寄存器中
- 堆栈，SS，SP，BP
- 扩展段用来存放_____,访问扩展段的段地址存放在_____中，偏移地址可以存放在_____寄存器中
- 数据，ES，SI、DI、BX、BP（均需加 ES:）

练习2—物理地址计算

- 取指令时8086的20位存储器地址是如何计算得到的？ **4**
- **（1）DS*16+暂存器（2）IP（3）CS（4）CS*16+IP**

- 取数据时8086的20位存储器地址是如何计算得到的？ **1**
- **（1）DS*16+偏移量（2）DS*16+IP（3）CS（4）CS*16+IP**

练习3—段空间与物理地址计算

- 假设**DS=0100H**，请写出**DS**段的物理地址范围

  01000H - 10FFFH

- 假设**ES=1234H**，请写出**ES**段的物理地址范围

  12340H - 2233FH

- 假设**CS=2100H**，**IP=12FFH**，则读取指令时出现在地址总线上的地址是？

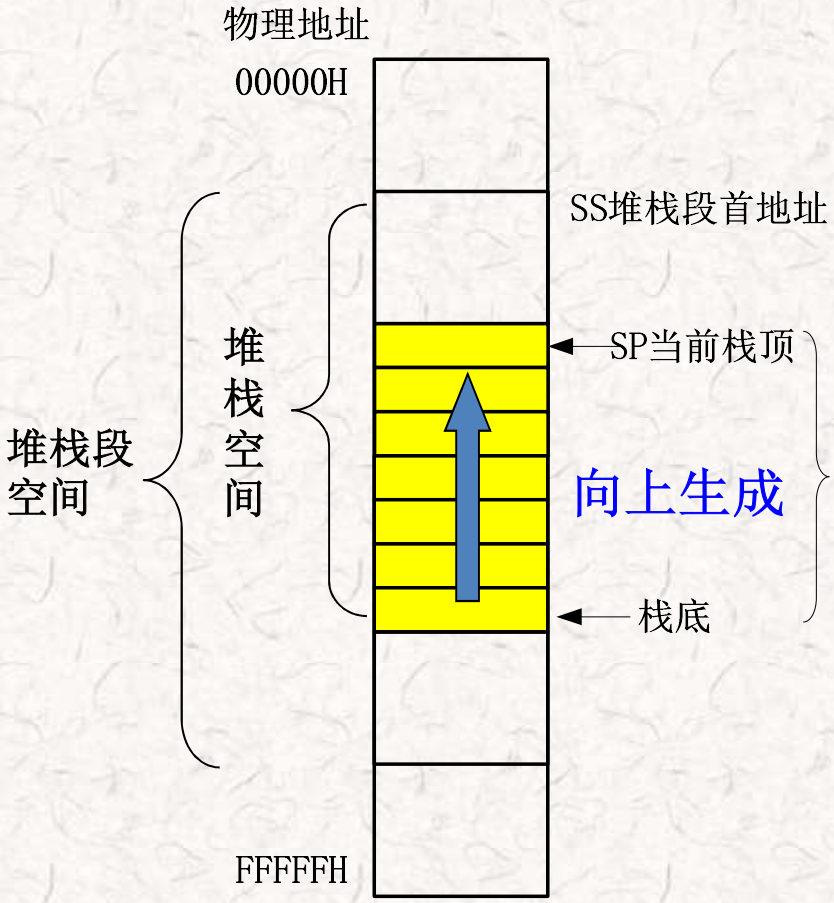
  CS * 16 + IP = 222FFH

5.8086 堆栈操作

- 什么是堆栈？堆栈的操作特点？主要用途？
- **8086**的堆栈设在哪个段？段寄存器和堆栈指针寄存器分别是什么？
- 进出栈的数据位宽是多少？高低字节如何放置？
- 进栈指令是什么？产生什么操作？
- 出栈指令是什么？产生什么操作？

阅读课本 2.1.4

堆栈的操作和生成方向



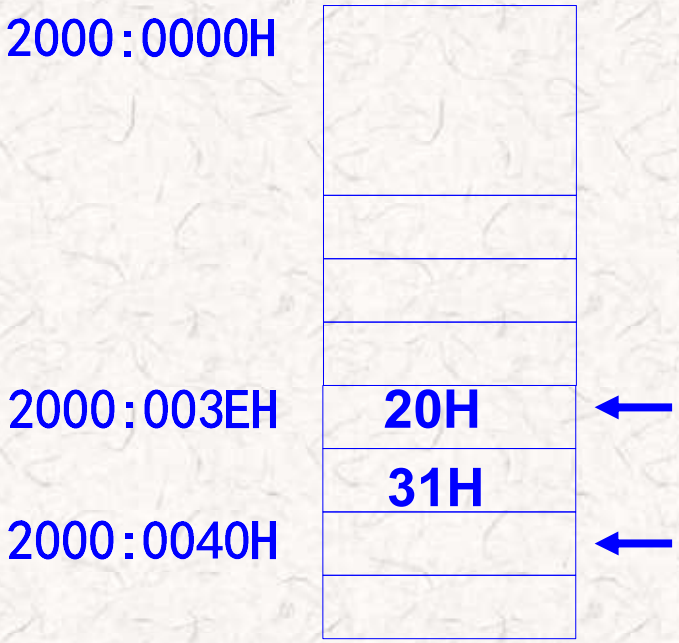
堆栈是一块特殊的数据区，后进先出

进栈：向栈顶压入两个字节，栈顶指针-2
出栈：从栈顶弹出两个字节，栈顶指针+2

初始化时，给**SP**一个初值，即定义栈底，堆栈的最大容量就确定了

8086进出栈操作

- 8086为硬件堆栈
- 进出栈均为2字节，16bit
- 进栈举例 PUSH BX
- SP<-SP-2
- 准备进栈的数据低字节->SP
- 准备进栈的数据高字节->SP+1
- 出栈举例 POP AX
- SP -> 目的操作数的低字节
- SP+1 -> 目的操作数的高字节
- SP<-SP+2
- 低字节在低地址（偶数地址），高字节在高地址（奇数地址），即SP须定义为偶数



AX= 3120H
BX=3120H

堆栈定义与操作举例

设初始化
SS=C000H
SP=2000H
则堆栈区如图

堆栈段地址范围：C0000H~CFFFFH

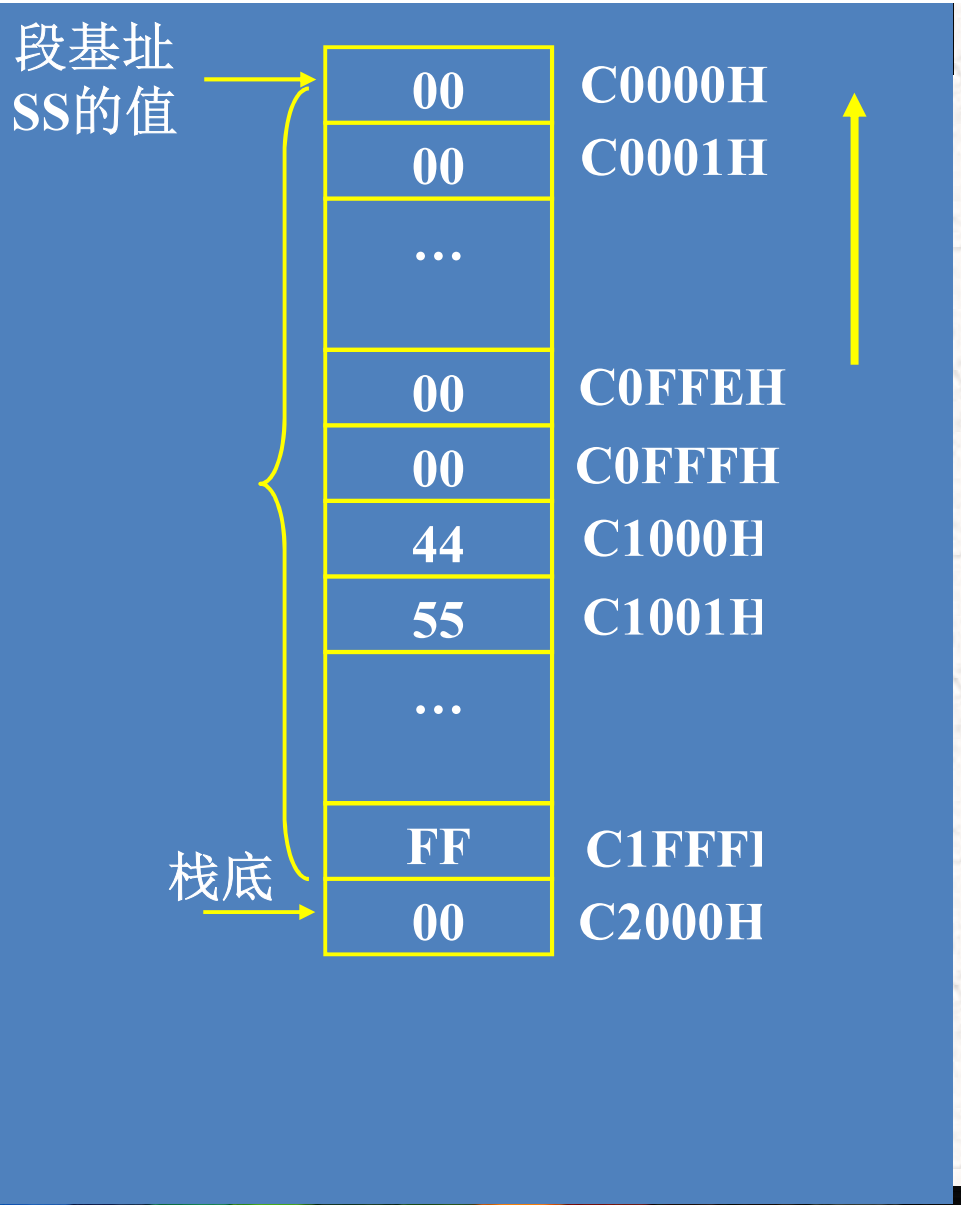
栈底物理地址是：C2000H

堆栈长度是：2000H

压入一个字后SP指针为：1FFEH

若SP=1000H，则当前栈顶物理地址是：C1000H

POP AX的执行结果：AX=5544H

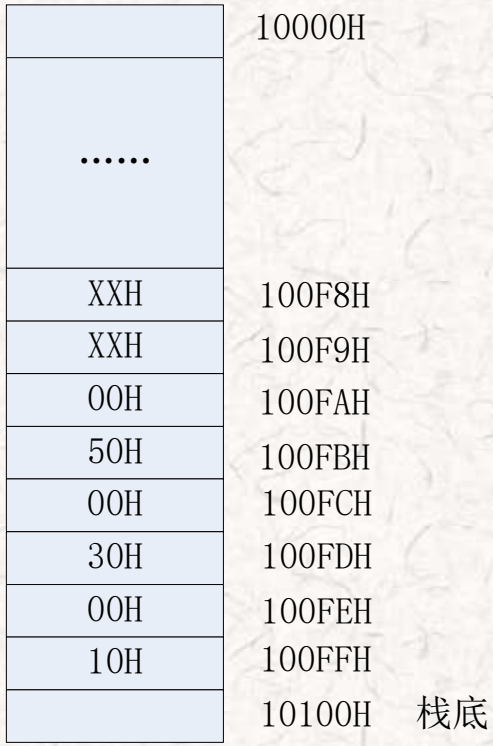


举例注意进出栈顺序对数值的影响

- 执行下列程序后，画出堆栈存储中的内容，并写出AX、BX、CX、SP的值
- 设SS=1000H
- MOV SP, 0100H
- ……
- MOV CX, 1000H
- MOV BX, 3000H
- MOV AX, 5000H
- PUSH CX
- PUSH BX
- PUSH AX
- POP CX
- POP BX
- POP AX

AX=1000H
BX=3000H
CX=5000H

SP=0100H



堆栈的用途和应用场景

- 用于暂存数据，保存和恢复一组数据（上下文/现场）
- 进出栈指令：**PUSH XX、POP XX （XX: reg or mem)**
- 子程序调用**CALL XXX**指令：（**CS**）、**IP**进栈
- 子程序返回**RET**指令：栈顶弹出到 **IP**、（**CS**）
- 进入中断服务程序（硬件自动）：**CS**、**IP**、**FLAGS**进栈
- 退出中断服务程序 **IRET**：栈顶弹出到**FALGS**、**IP**、**CS**

- 注意堆栈操作的对偶性质，须保证进出栈顺序相反和数量相同

还有问题吗？

作业

P52　1(8086部分), 2~11

6.8086 存储器分体结构

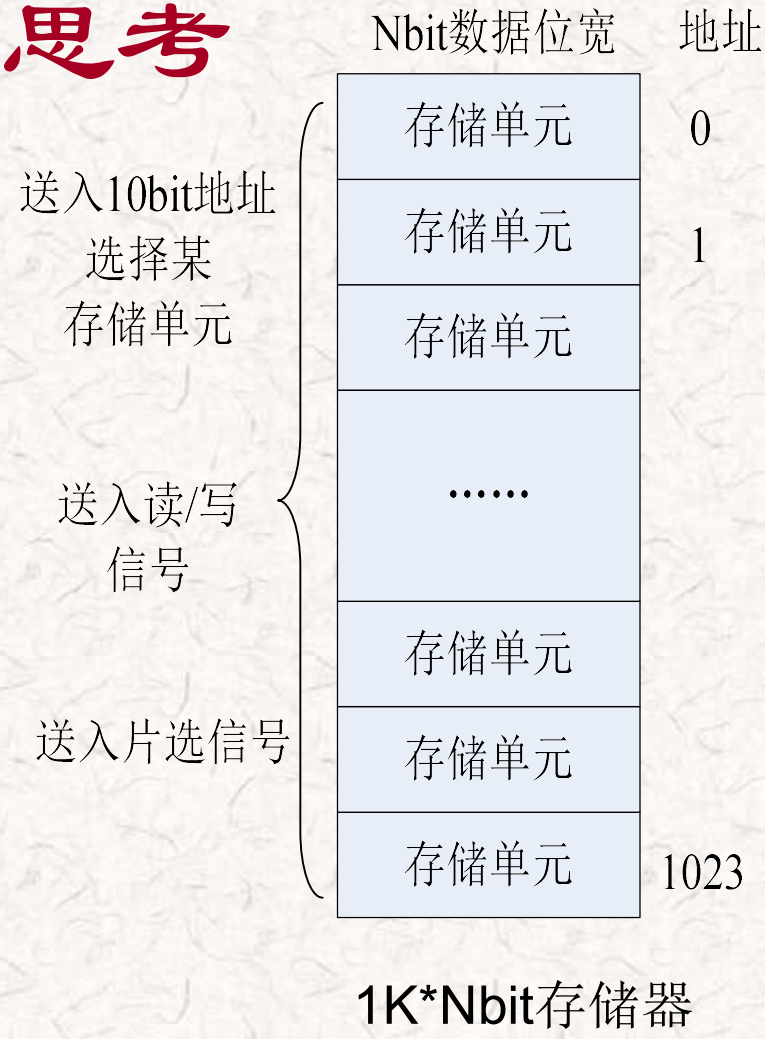
- 为什么**8086**要采用奇偶存储体的存储器组织模式？
- **8086**奇偶存储体各自的最大容量和地址范围？
- 一个**16**位的字在**8086**存储器中占据几个存储单元？
- 奇偶存储体与数据总线、地址总线和控制信号之间如何连接？
- **8086**通过哪些信号选通偶存储体进行字节读写访问？被读写的数据通过数据总线的哪些信号在**8086**和存储器之间传输？

阅读课本 2.1.4

存储器访问--回顾与思考

- 存储块每个单元位宽相同，均为Nbit
- 存储块内部单元地址连续
- 读操作：
- 给出存储单元地址、/RD、/CS，然后得到该单元的Nbit的数据
- 写操作：
- 同时给出存储单元地址、/WR、/CS、Nbit数据，写入相应单元

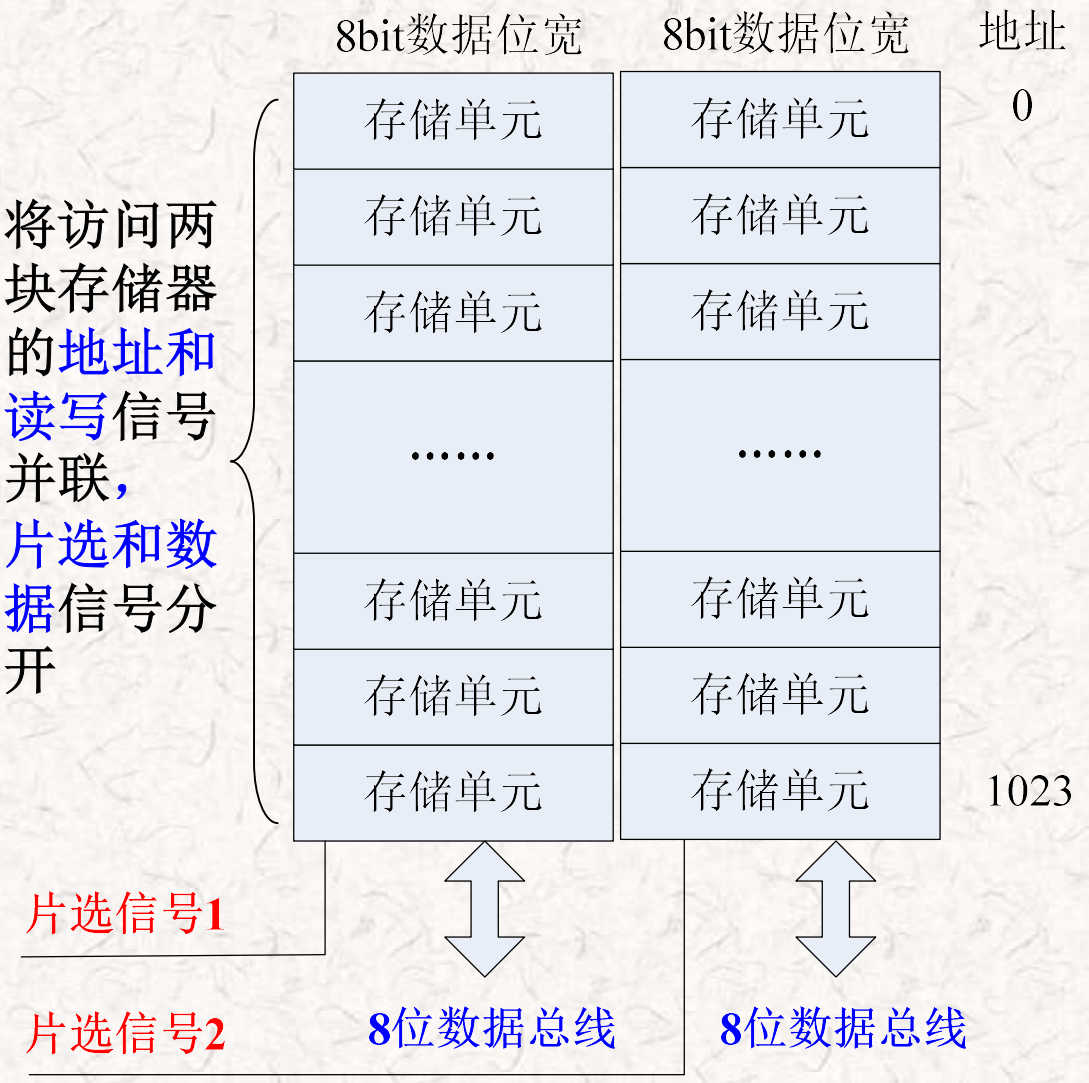
- 8086可以进行8bit和16bit访问，并且支持非对齐访问，如何实现呢？



1K*Nbit存储器

概念演示：

如何通过8bit存储器得到8/16bit数据



通过控制片选1和片选2可以得到8bit和16bit数据

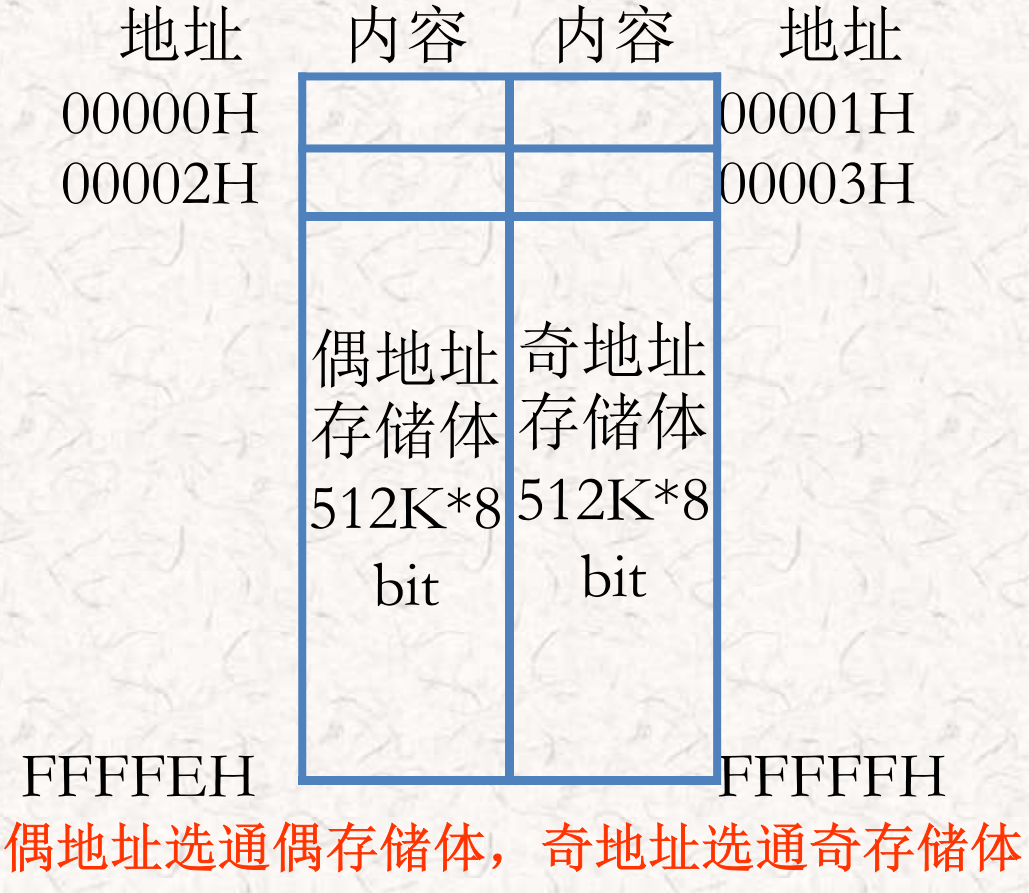
8086存储器分体结构的概念

1MB空间分为两个存储体
每个存储体地址空间512KB
每个地址存放8bit数据

地址和读写信号并联
片选和数据信号分开

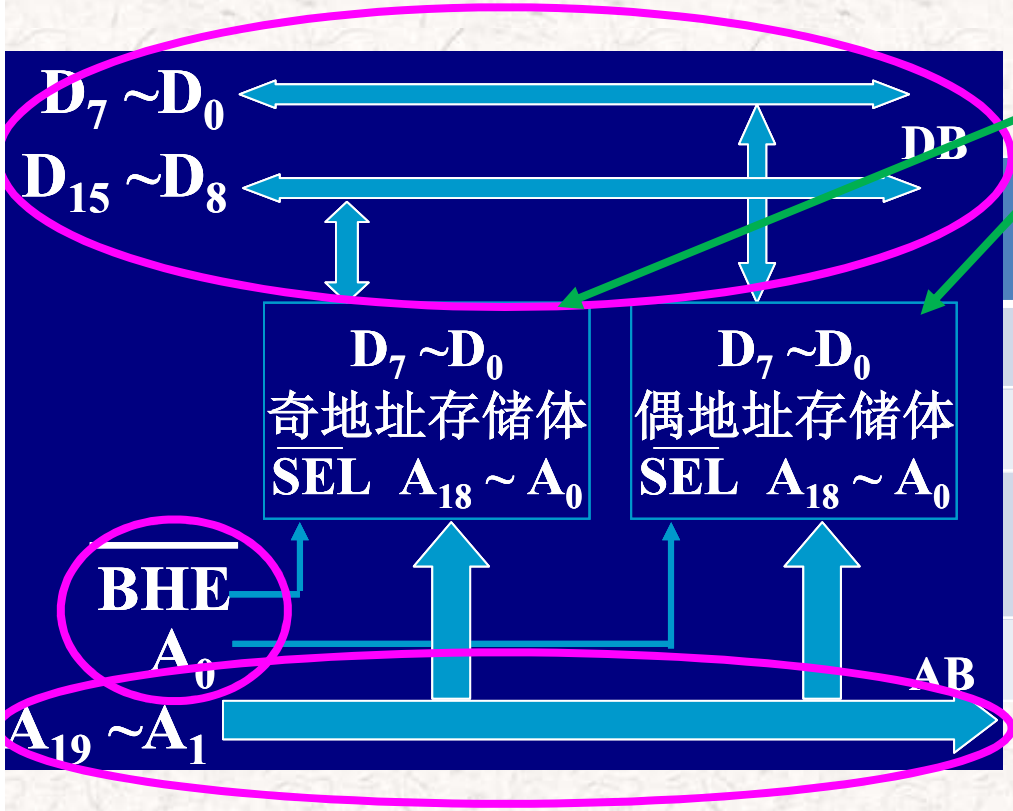
字节访问：奇地址访问选通奇地址存储体，偶地址访问选通偶地址存储体

字访问：同时选中两个存储体



偶地址选通偶存储体，奇地址选通奇存储体

奇偶存储体分体结构的实现



512KB

/BHE	A0	功能	数据总线
1	0	选中偶地址存储体	D7:0
0	1	选中奇地址存储体	D15:8
0	0	选中两个存储体，读写一个字	D15:0
1	1	无操作	-

通过送出偶地址+/BHE=0实现字访问

/BHE配合奇地址字节访问和字访问出现 /BHE=! (A0 | WORD)

举例：对齐与非对齐字访问

所存放数据	物理地址
11H	00075H
22H	00076H
33H	00077H

总结：只有偶地址起始的字（也称为对齐字）可以一次性读取

设DS=0H

MOV AX, [0076H]；低字节0076H，高字节0077H

对地址00076H开始的16位字可以同时读出吗？ AX=3322H

	A19	A18	……	A8	A7	A6	A5	A4	A3	A2	A1	A0	$\overline{BHE}$	
0076H	0	0	……	0	0	1	1	1	0	1	1	1	0	0077H

单次访问无法同时得到00075H奇地址和00076H存储单元的内容

MOV AX, [0075H]；低字节0075H，高字节0076H

对地址00075H可以同时读出这16位吗？

	A19	A18	……	A8	A7	A6	A5	A4	A3	A2	A1	A0	$\overline{BHE}$	
0075H	0	0	……	0	0	1	1	1	0	1	0	0	0	0074H

奇地址开始的字如何访问？

所存放数据	物理地址
11H	00075H
22H	00076H
33H	00077H

MOV AX, [0075H]；低字节0075H，高字节0076H

对地址00075H如何进行字读取？分两次存储器读操作
（8086硬件自动完成，对编程人员透明）

A19	A18	……	A8	A7	A6	A5	A4	A3	A2	A1	A0	BHE
0	0		0	0	1	1	1	0	1	0	0	0
0	0		0	0	1	1	1	0	1	1	0	0

1. 读00074H，舍弃低字节
2. 读00076H，舍弃高字节

将两次访问的字节组合成字

执行后，AX的值为：2211H

总结8086存储器分体结构特点

- 整个存储空间映射到奇偶两个存储体
- 每个存储体**1**字节位宽，分别连接到数据总线的**D15:8, D7:0**
- 偶存储体用**A0**选通，奇存储体用**/BHE**选通
- 对偶地址的字节访问通过数据总线**D7:0**进行
- 对奇地址的字节访问通过数据总线**D15:8**进行
- 读/写起始地址为偶地址的字（对齐的字）通过一次存储器访问可以实现
- 读/写起始地址为奇地址的字（非对齐的字）通过两次存储器访问可以实现

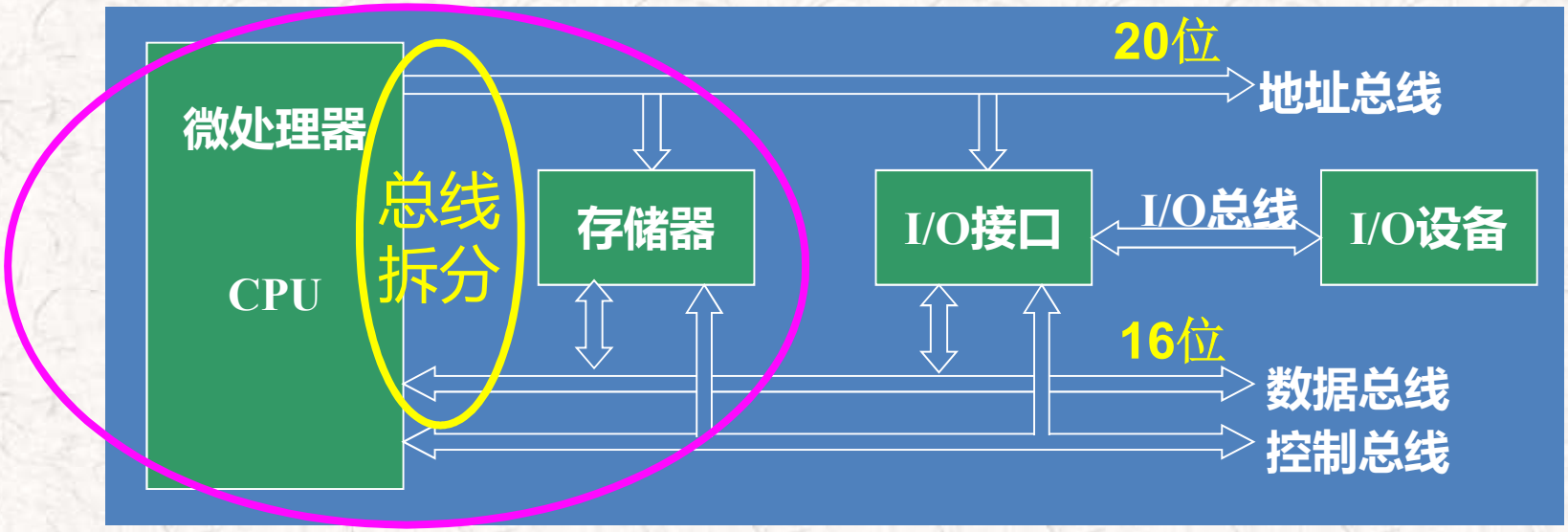
7.8086 系统组成与总线周期

- 时钟周期
- 总线周期
- 指令周期

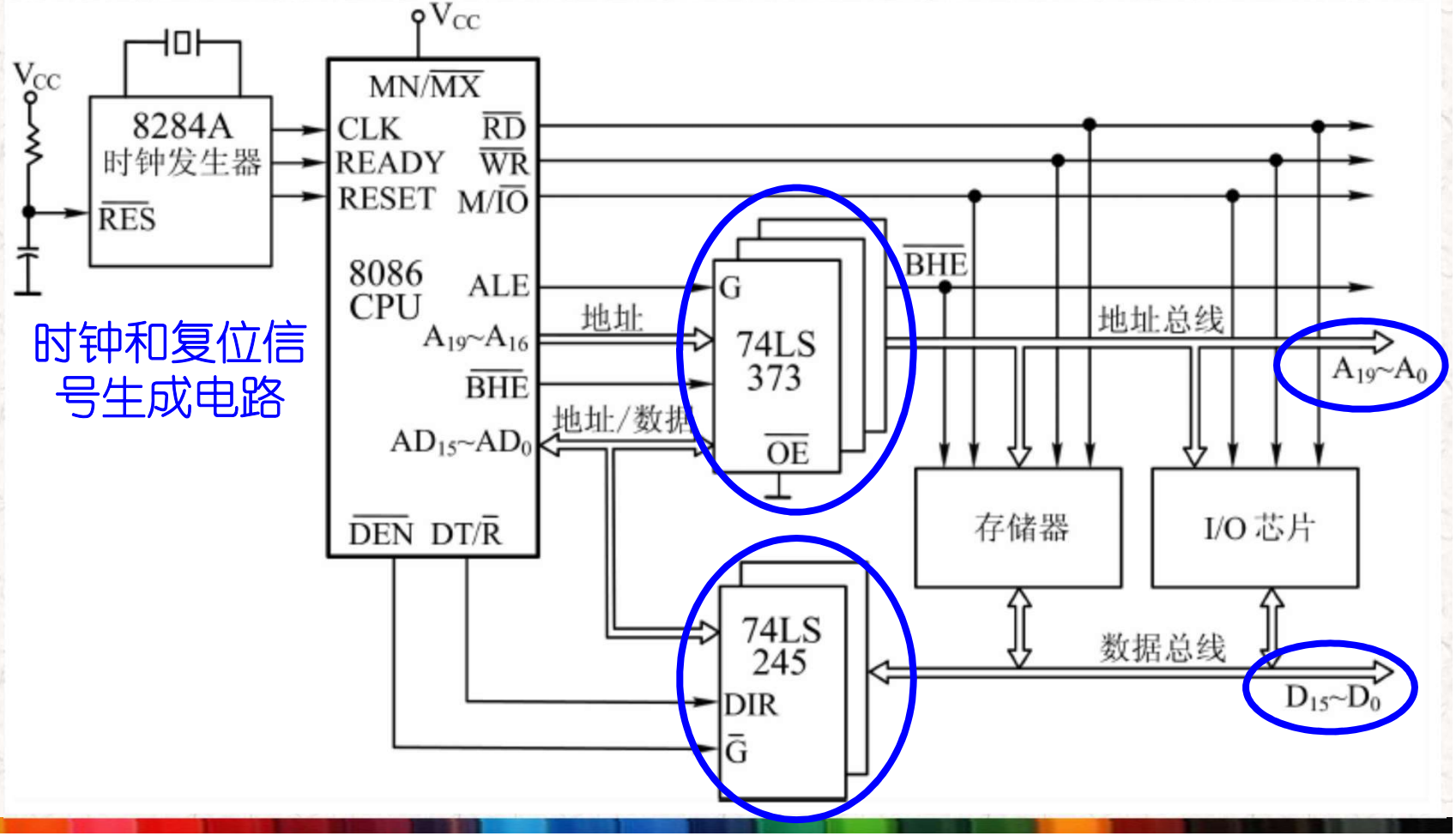
阅读课本 2.2引言和2.2.1（不含4. 时钟发生器）， 2.2.3（不含3.最大模式下的读/写总线周期）

8086最小模式下的总线拆分

- 最小模式特点：系统的主控方是单个8086
- 表现为：总线控制权和控制权的切换由单片8086决定
- 最大模式：多8086处理器，由独立仲裁器决定谁获得控制权



8086系统最小模式系统



时钟和复位信号生成电路

锁存器74LS373

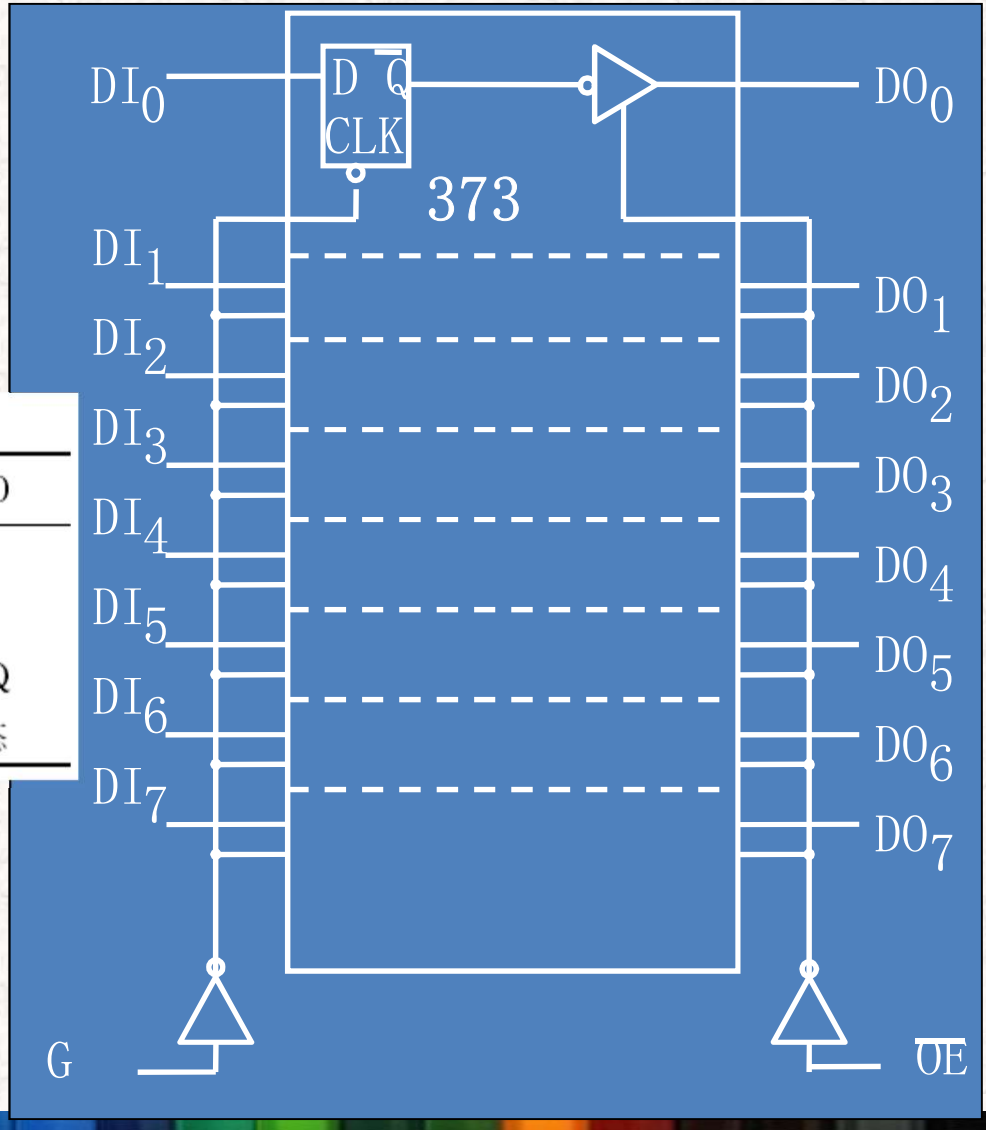
G： 锁存控制，透明 **or** 保持
/OE： 输出使能，驱动 **or** 高阻

表 2.5　74LS373 的真值表

输入使能端 G	输出允许端$\overline{OE}$	输入 D	输出 O
1	0	1	1
1	0	0	0
0	0	×	锁存 Q
×	1	×	高阻态

逻辑门输入或输出的圆圈表示取反

注意：课本P39，373应为锁存器，不是触发器

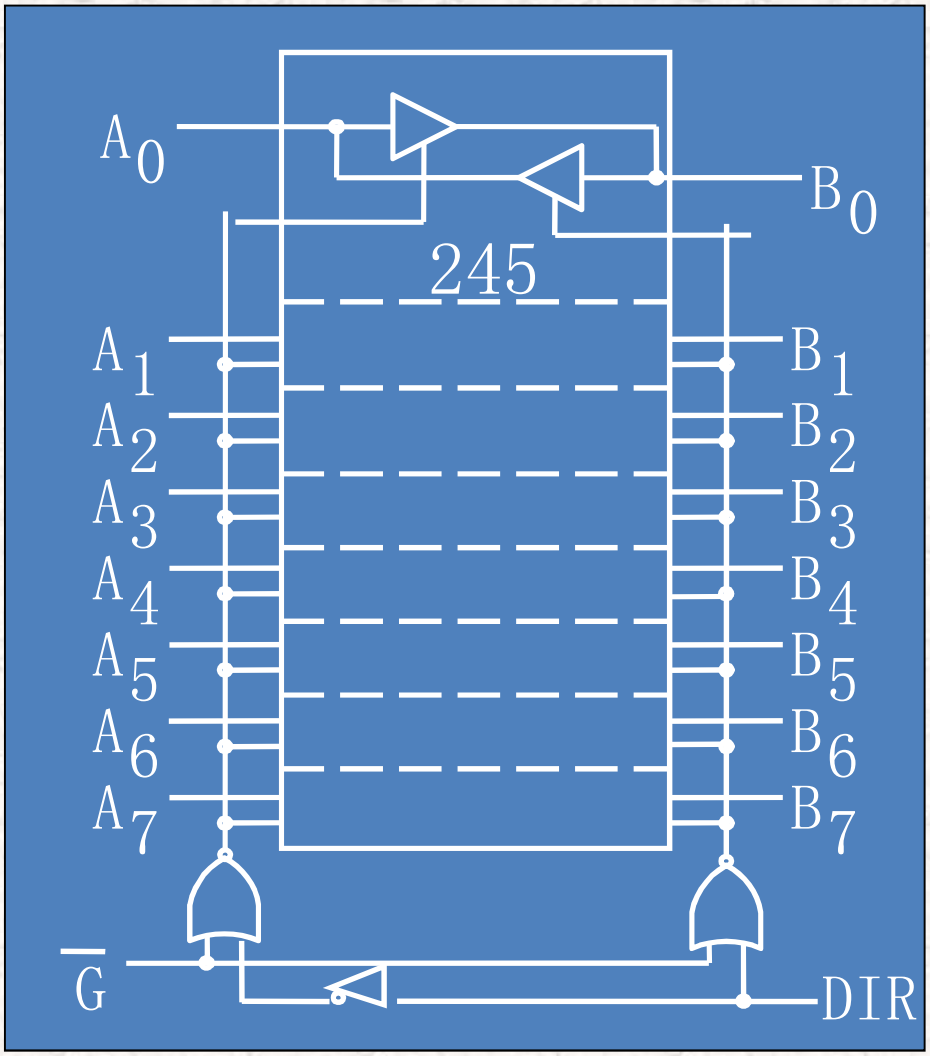


双向驱动器

74LS245

/G：输出使能，驱动 or 高阻
DIR：方向控制，输出 or 输入

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

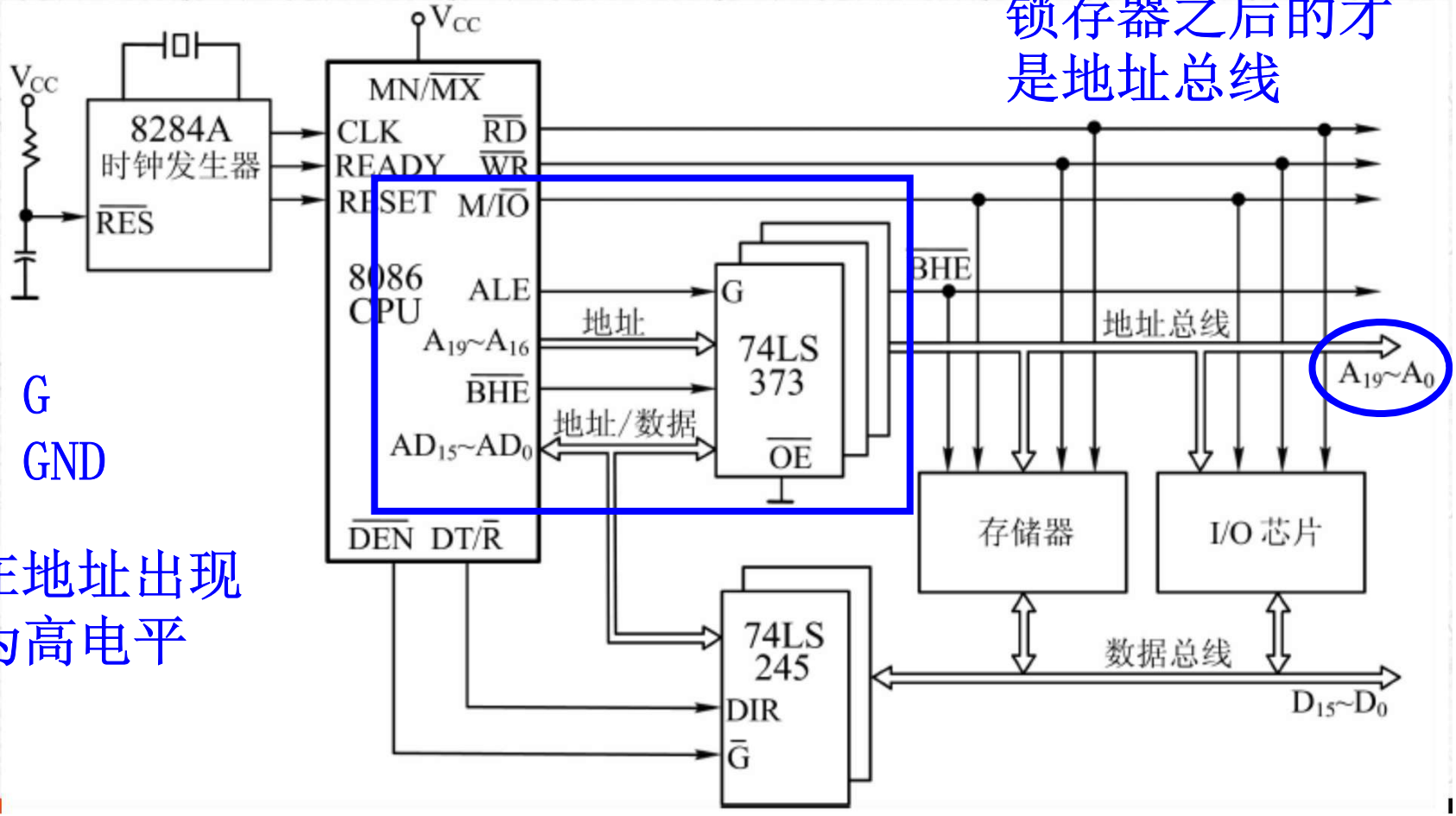


8086系统20根地址总线生成

锁存器之后的才是地址总线

373:
ALE – G
/OE – GND

ALE在地址出现期间为高电平

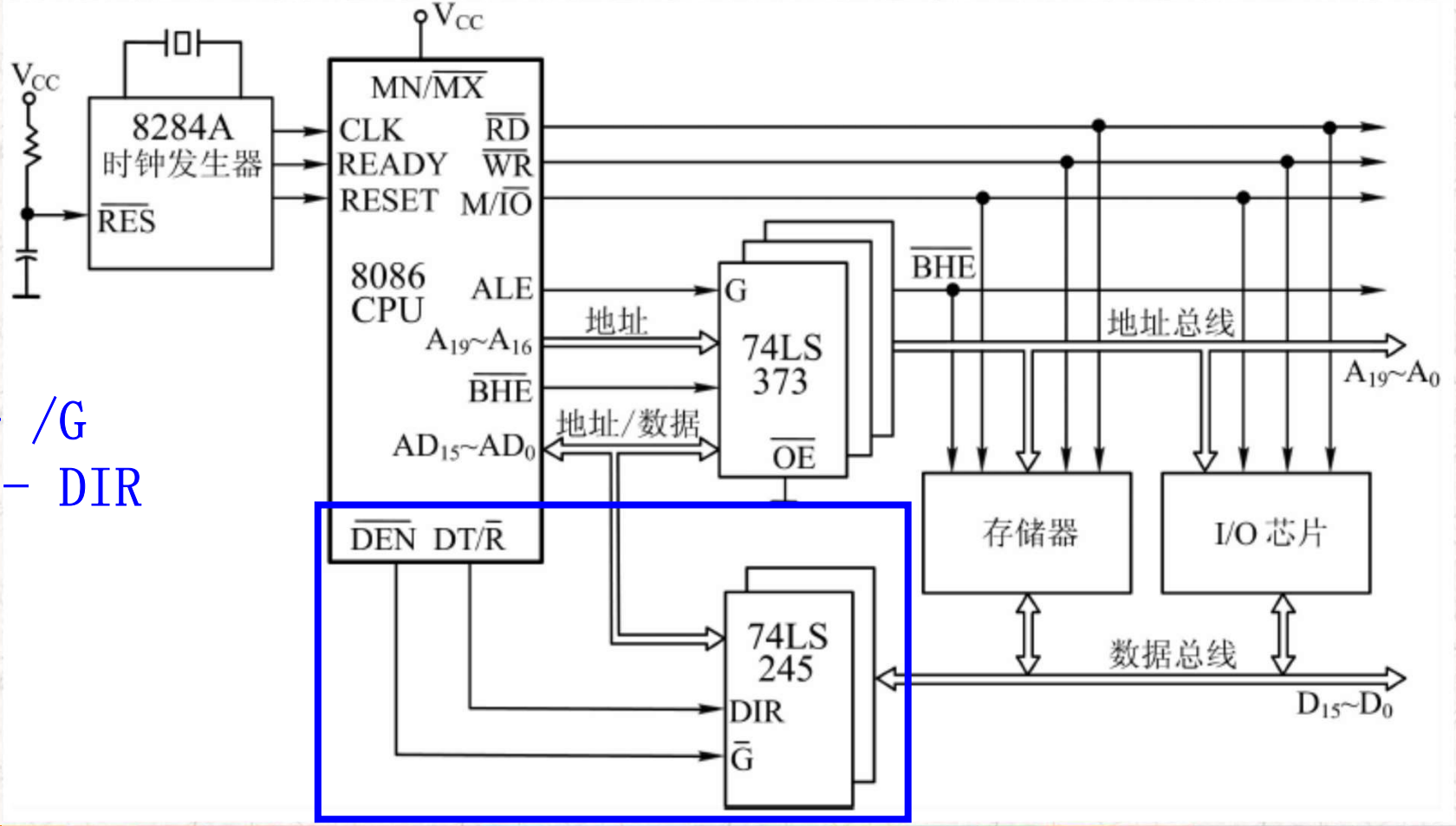


8086系统16根数据总线驱动

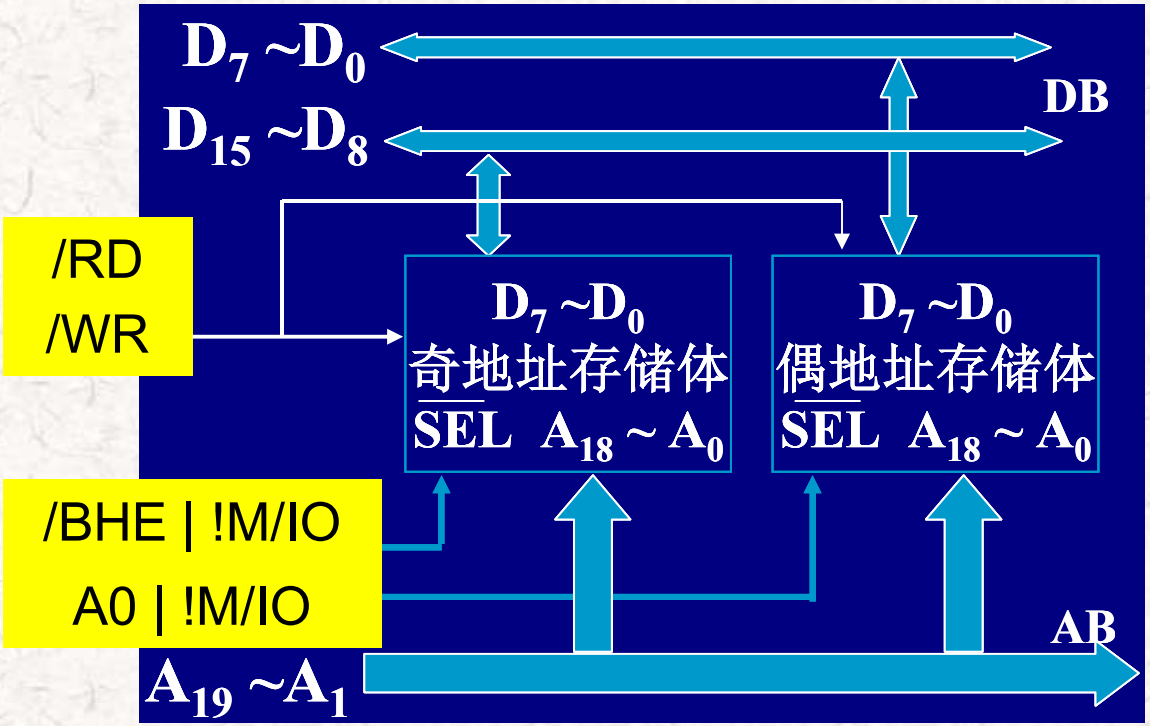
245:

DEN - /G

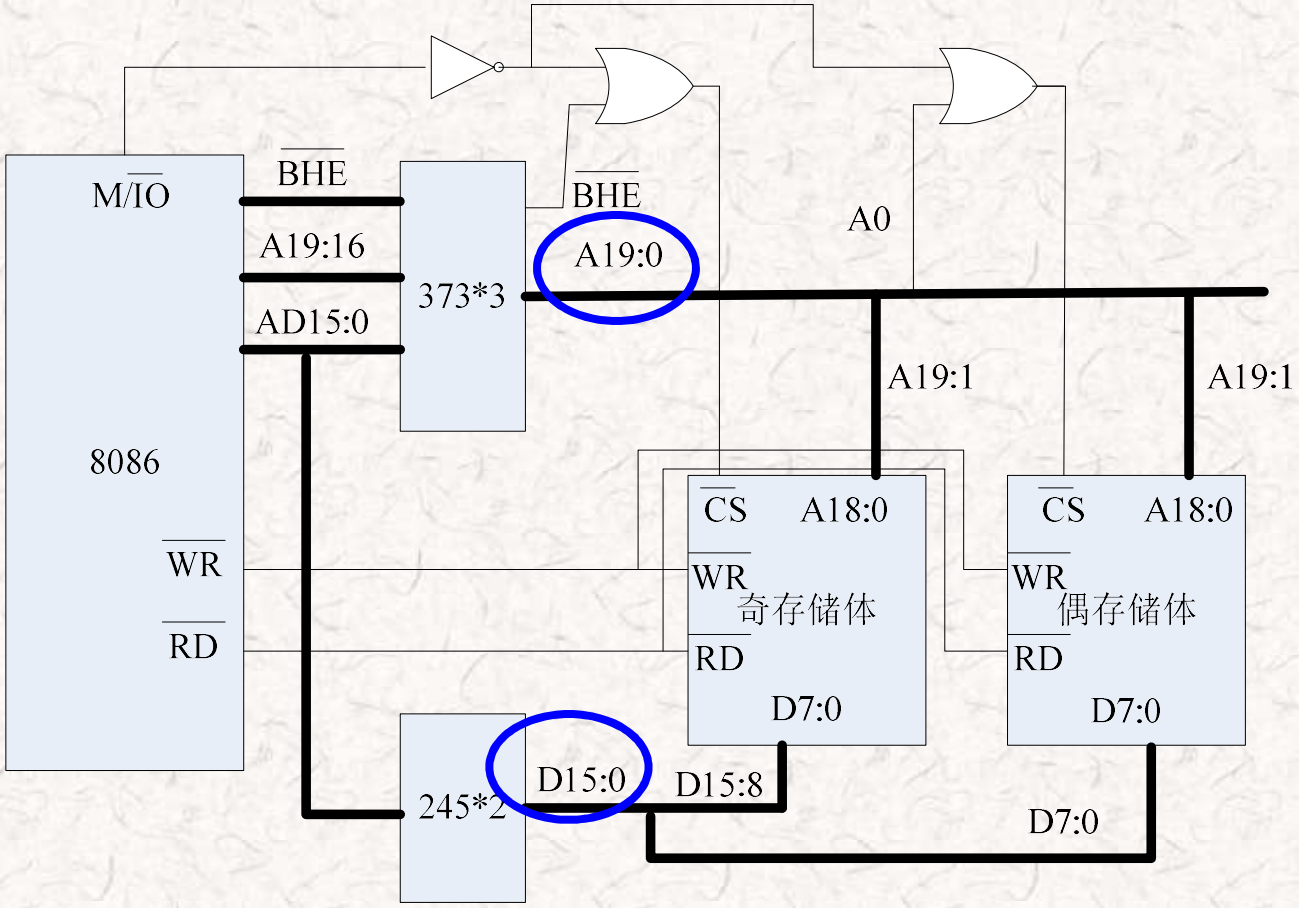
DT/R - DIR



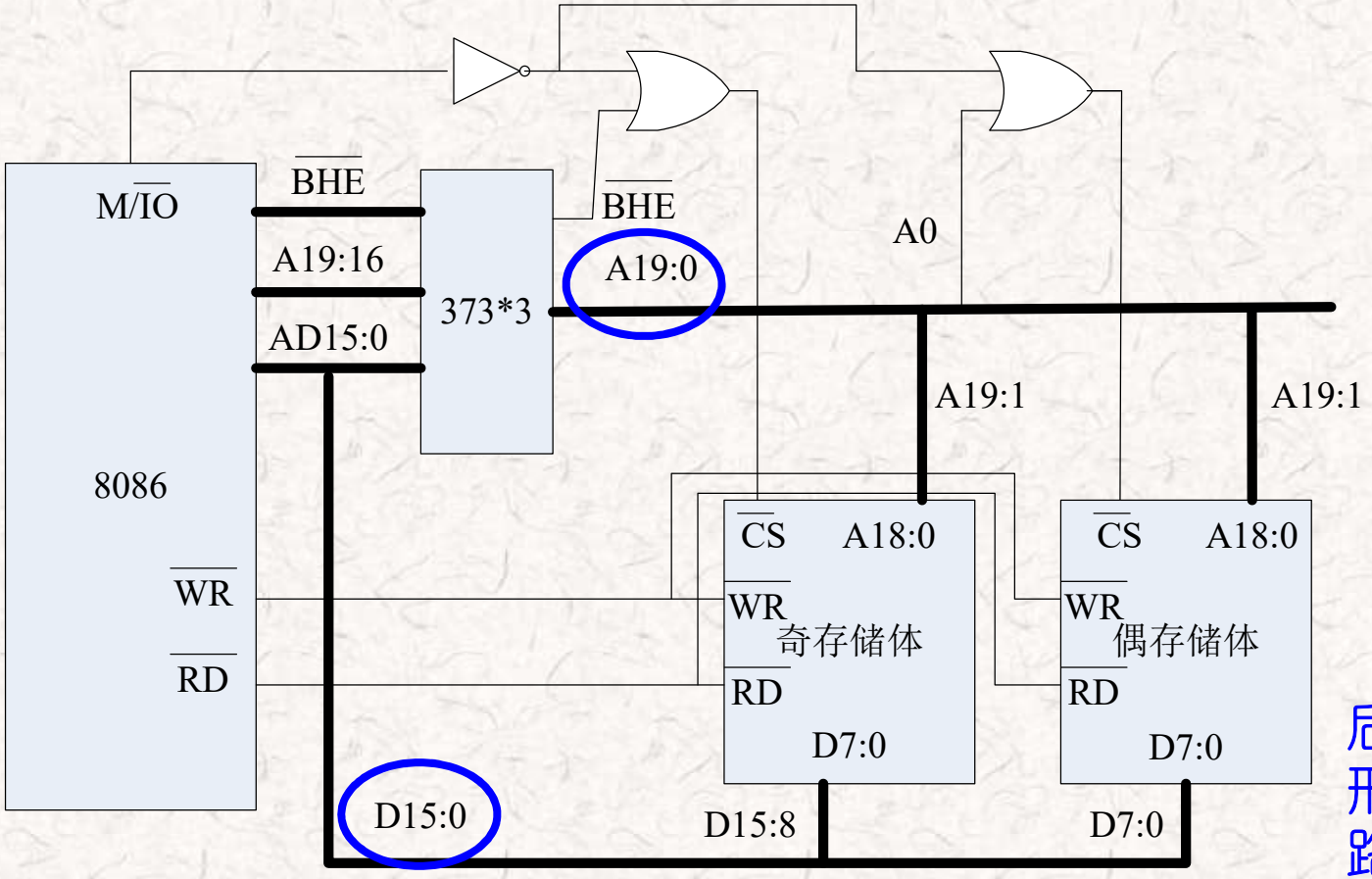
8086系统奇偶存储体接入



8086最小系统--有数据总线驱动



8086最小系统--无数据总线驱动

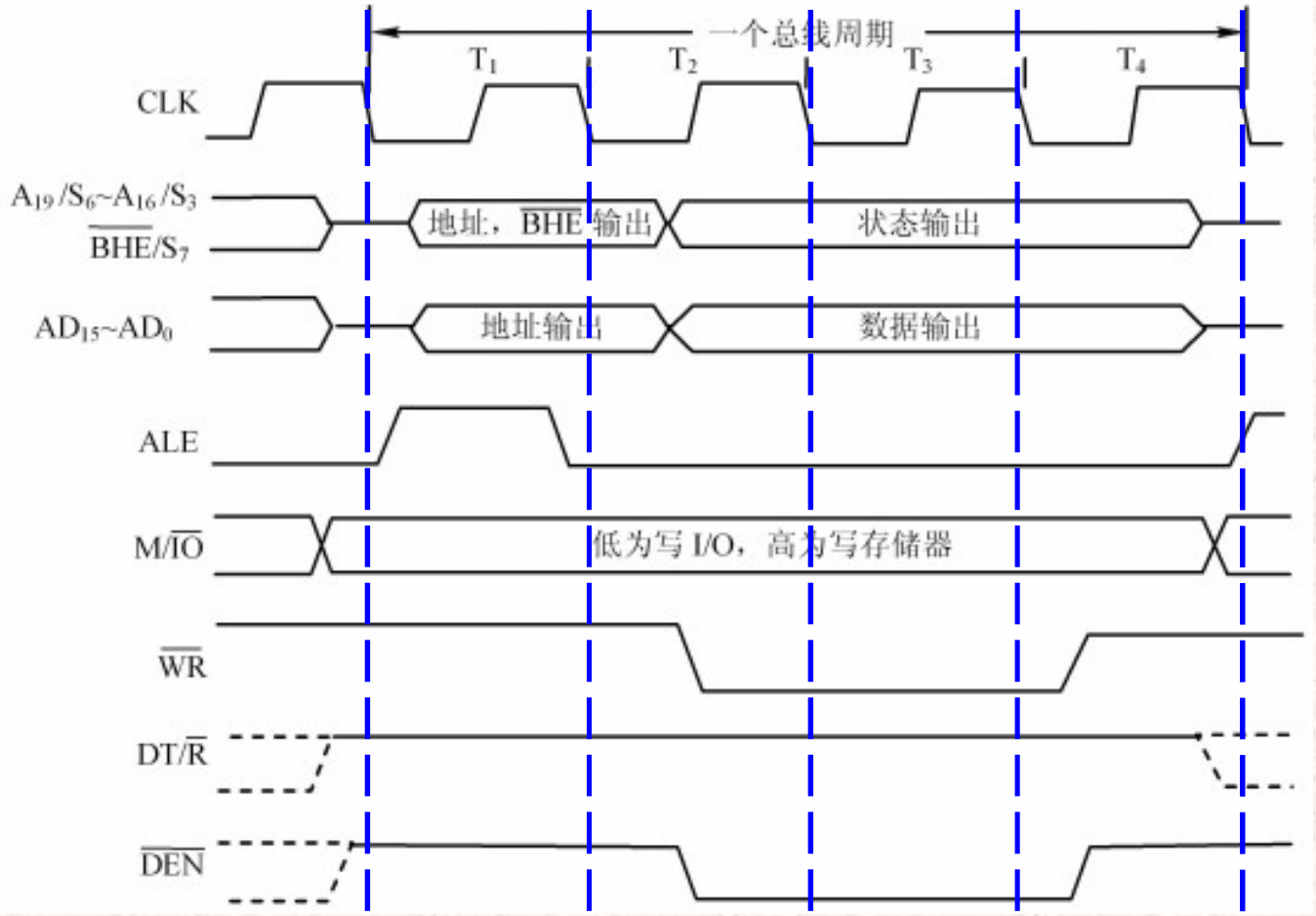


后面的总线波形假设在此电路上进行分析

总线操作之--写总线时序

时钟周期与总线周期

写操作
8086输出字或者字节到指定某地址的存储器或输出端口



写总线操作举例1-写偶地址字节

若DS=3000H, AL=55H，执行 MOV [2200H], AL时产生如下总线周期

地址锁存器输出的20根地址总线

A0=0选中偶地址存储体

偶地址数据通过D7:0传递

/BHE=1不选中奇地址存储体

地址锁存使能

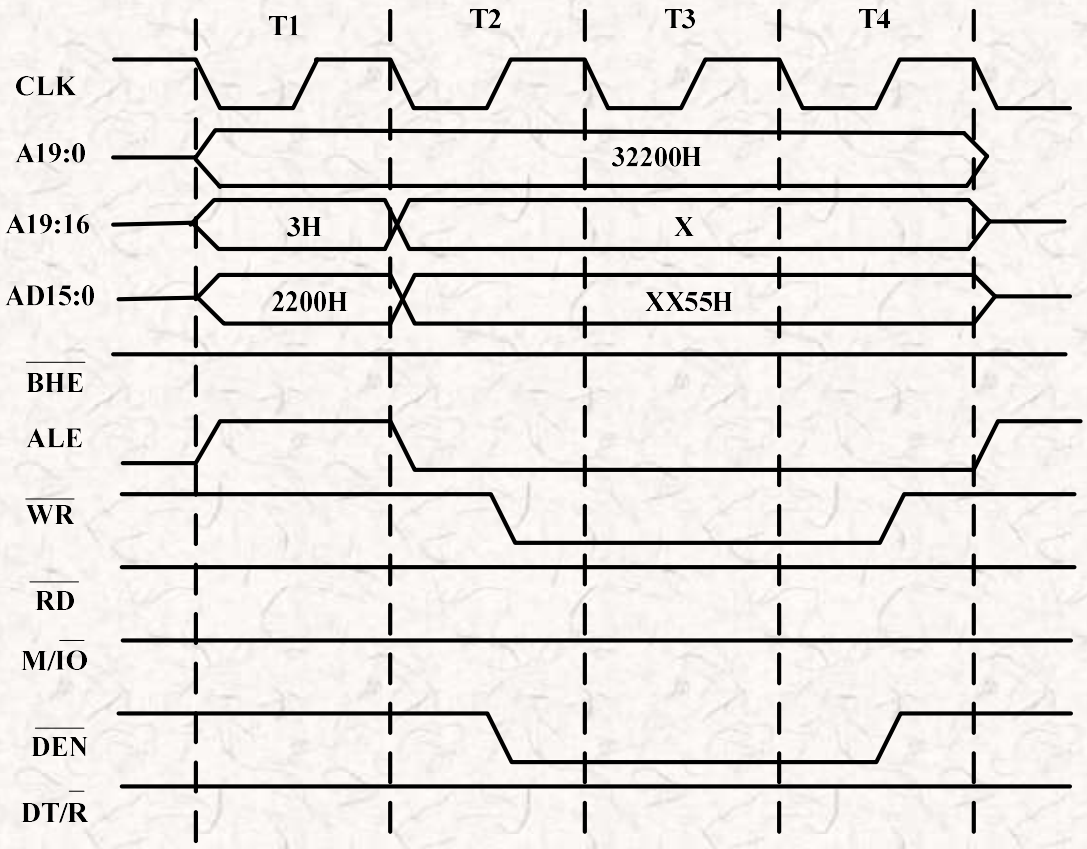
T2、T3中/WE=0表明执行写操作

/RE=1表明不执行读操作

M/IO=1表明当前访问存储器

数据总线驱动器输出使能

控制数据总线驱动器方向为输出



写总线操作举例2-写奇地址字节

若DS=3000H, AL=55H，执行 MOV [2201H], AL时产生如下总线周期

地址锁存器输出的20根地址总线

A0=1不选中偶地址存储体

奇地址数据通过D15:8传递

/BHE=0选中奇地址存储体

地址锁存使能

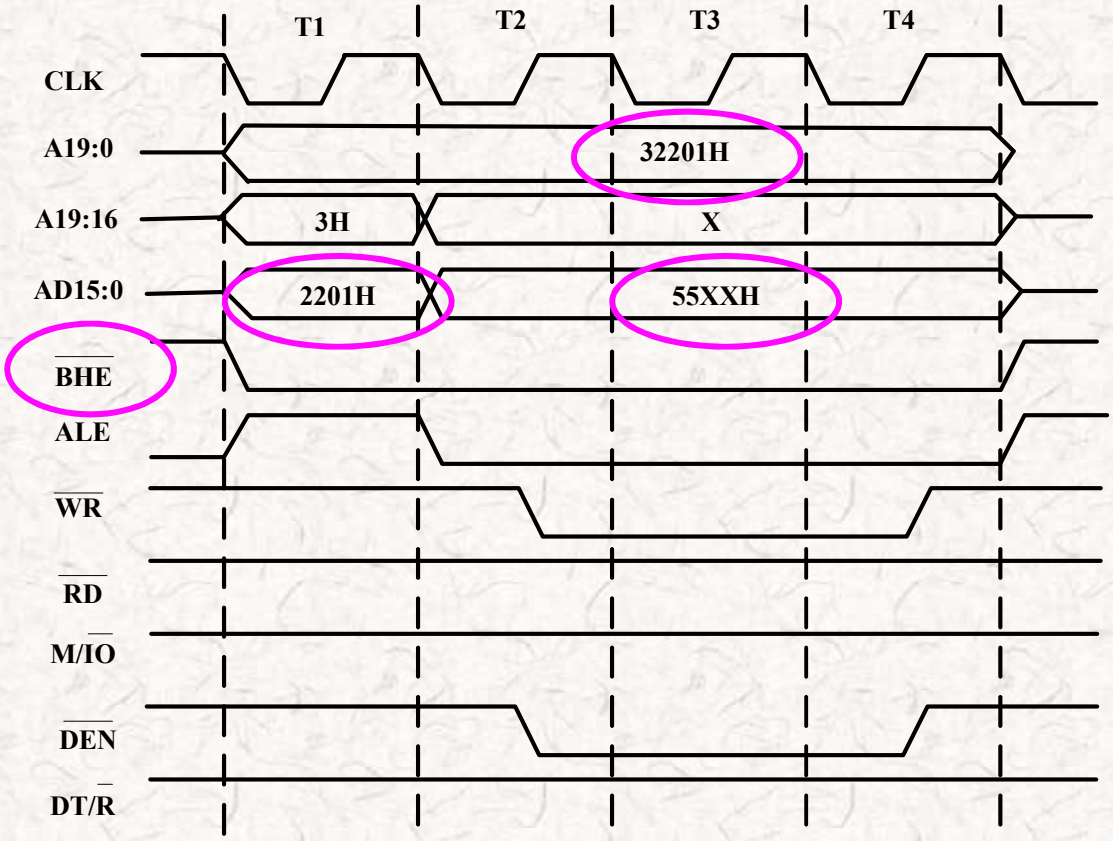
T2、T3中/WE=0表明执行写操作

/RE=1表明不执行读操作

M/IO=1表明当前访问存储器

数据总线驱动器输出使能

控制数据总线驱动器方向为输出



写总线操作举例3-写偶地址字

若DS=3000H, AX=4455H，执行 MOV [2200H], AX时产生如下总线周期

地址锁存器输出的20根地址总线

A0=0选中偶地址存储体

16位字通过D15:0传递

/BHE=0选中奇地址存储体

地址锁存使能

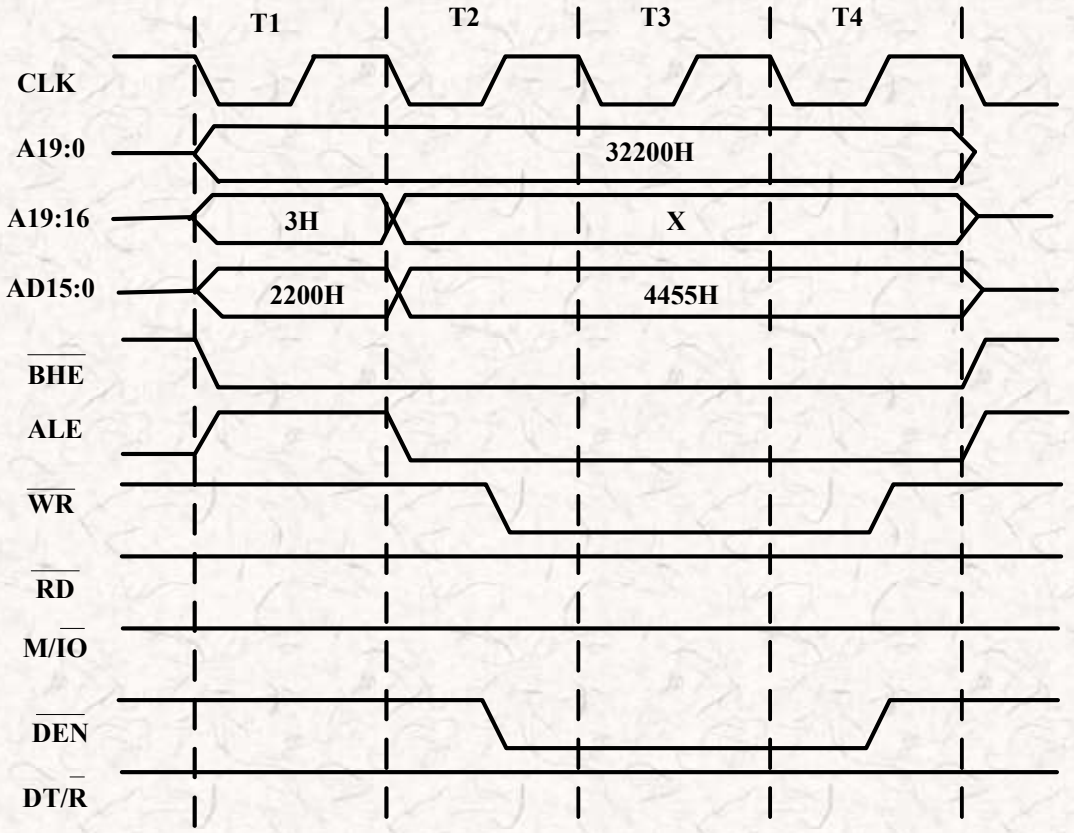
T2、T3中/WE=0表明执行写操作

/RE=1表明不执行读操作

M/IO=1表明当前访问存储器

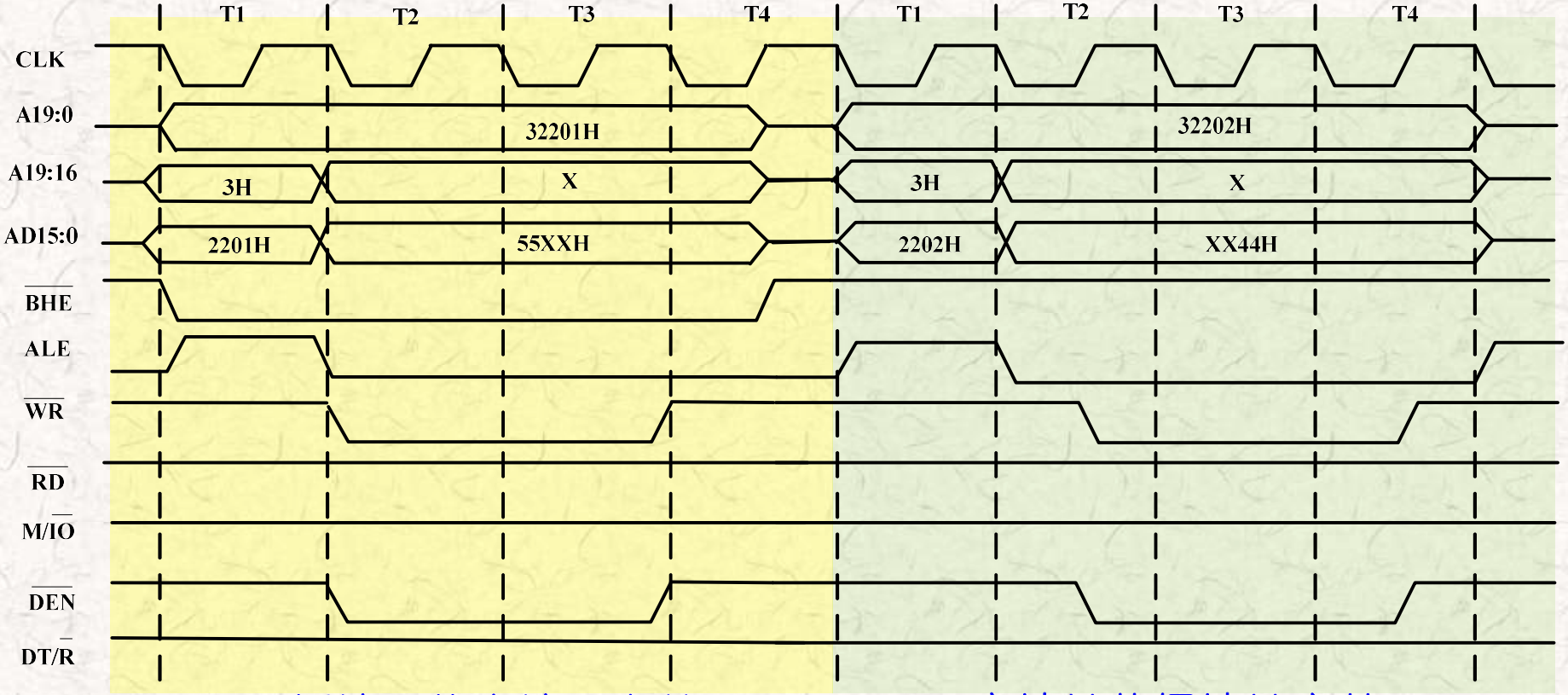
数据总线驱动器输出使能

控制数据总线驱动器方向为输出



写总线操作举例4-写奇地址字

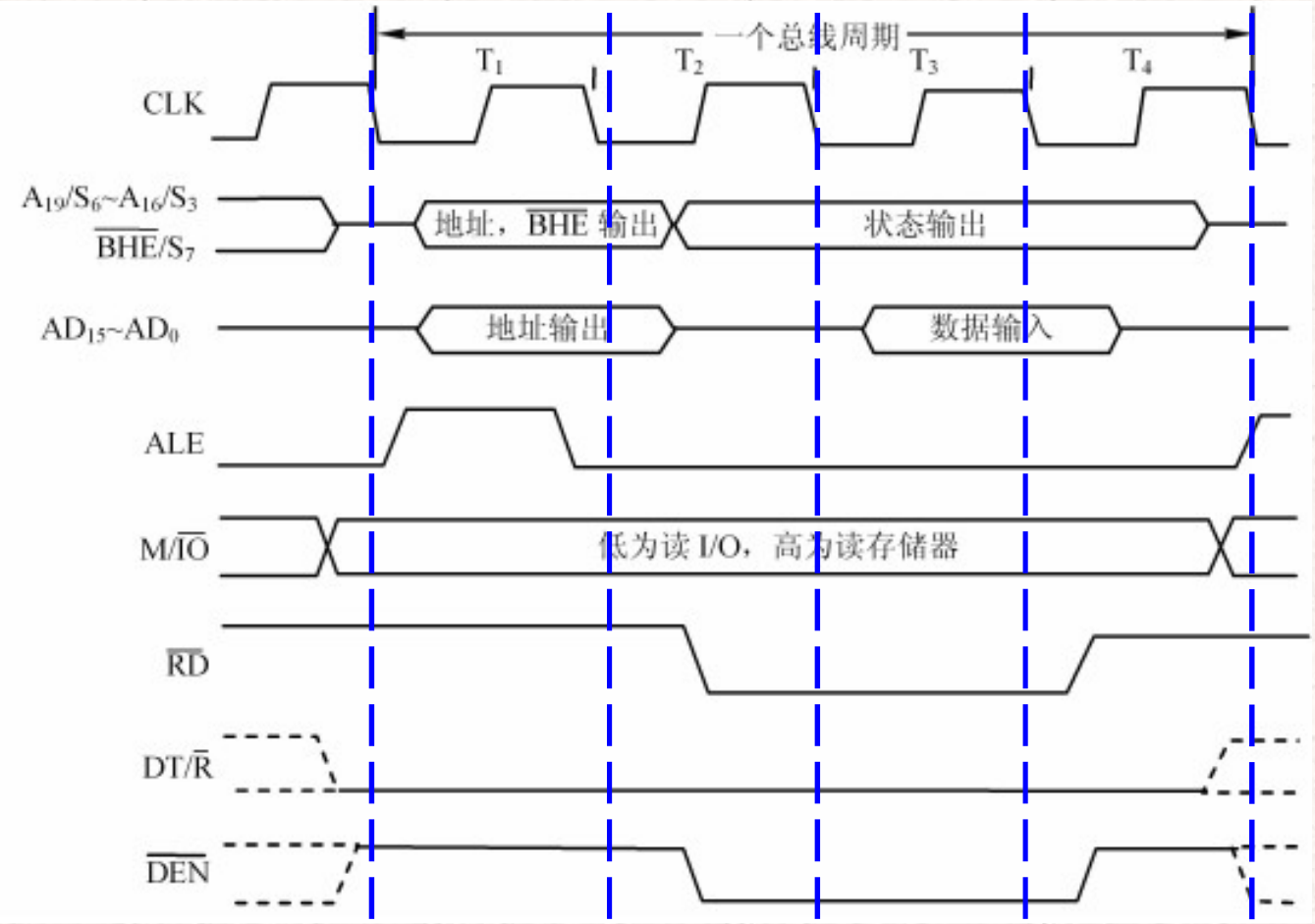
若DS=3000H, AX=4455H，执行 MOV [2201H], AX时产生如下总线周期



写低地址的奇地址字节　　写高地址的偶地址字节

总线操作之--读总线时序

读操作
被8086指定某地址的存储器或输入端口输出数据到数据总线



读总线操作举例1-读偶地址字节

若DS=9000H, 存储单元93FFEH存放66H，执行 MOV BL, [3FFEH] 时产生如下总线周期

地址锁存器输出的20根地址总线

A0=0选中偶地址存储体

8086放弃总线驱动，存储器输出驱动，偶地址数据通过D7:0传递

/BHE=1未选中奇地址存储体

地址锁存使能

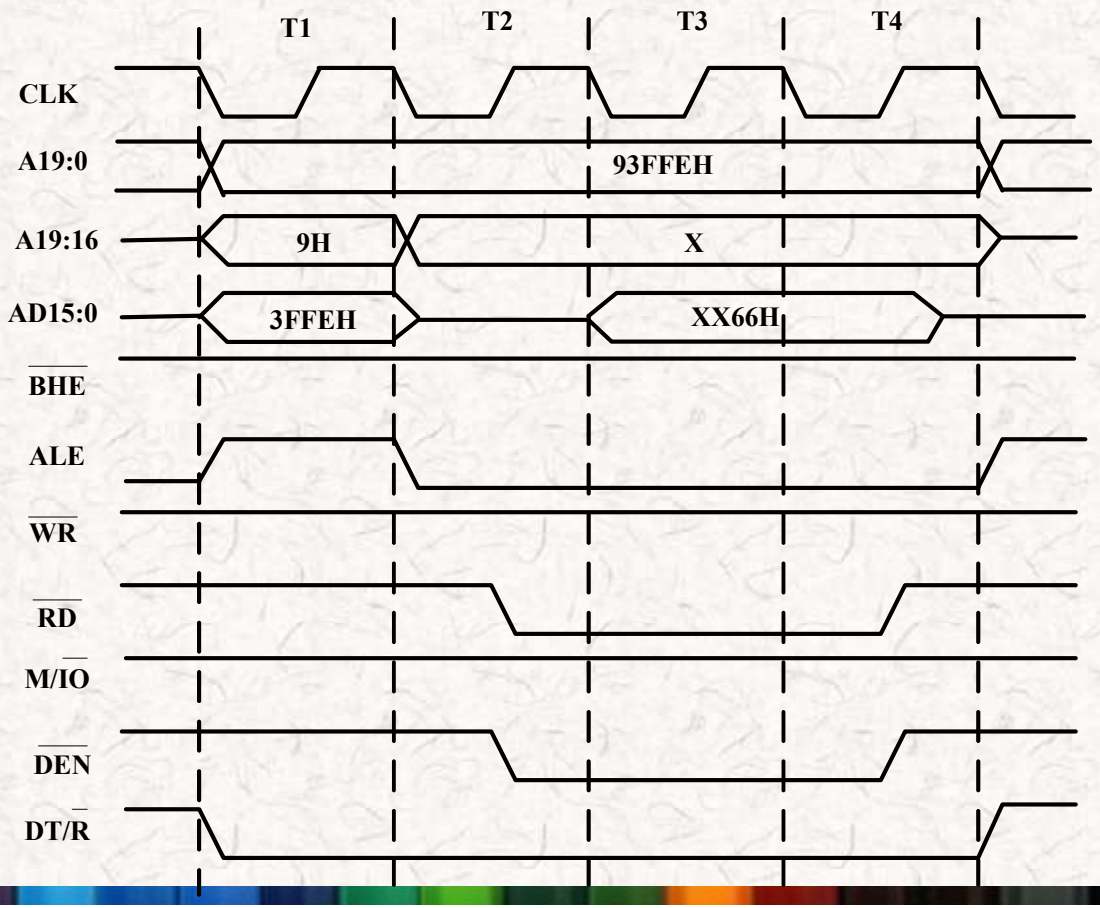
T2、T3中/WE=1表明不执行写操作

/RE=0表明执行读操作

M/IO=1表明当前访问存储器

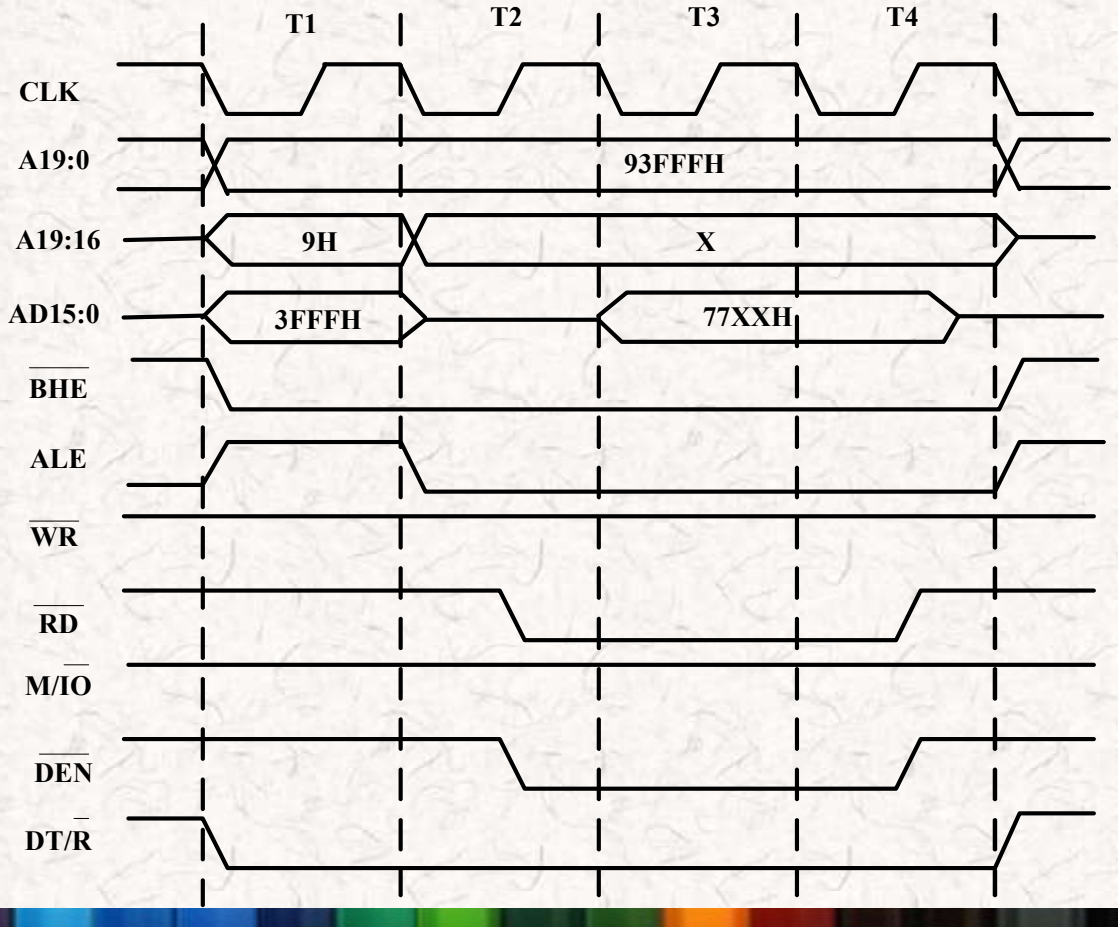
数据总线驱动器输出使能

控制数据总线驱动器方向为输入



读总线操作举例2-读奇地址字节

若DS=9000H, 存储单元93FFFH存放77H，执行 MOV BL, [3FFFH] 时产生如下总线周期



地址锁存器输出的20根地址总线

A0=1未选中偶地址存储体

8086放弃总线驱动，存储器输出驱动，奇地址数据通过D15:8传递

/BHE=0选中奇地址存储体

地址锁存使能

T2、T3中/WE=1表明不执行写操作

/RE=0表明执行读操作

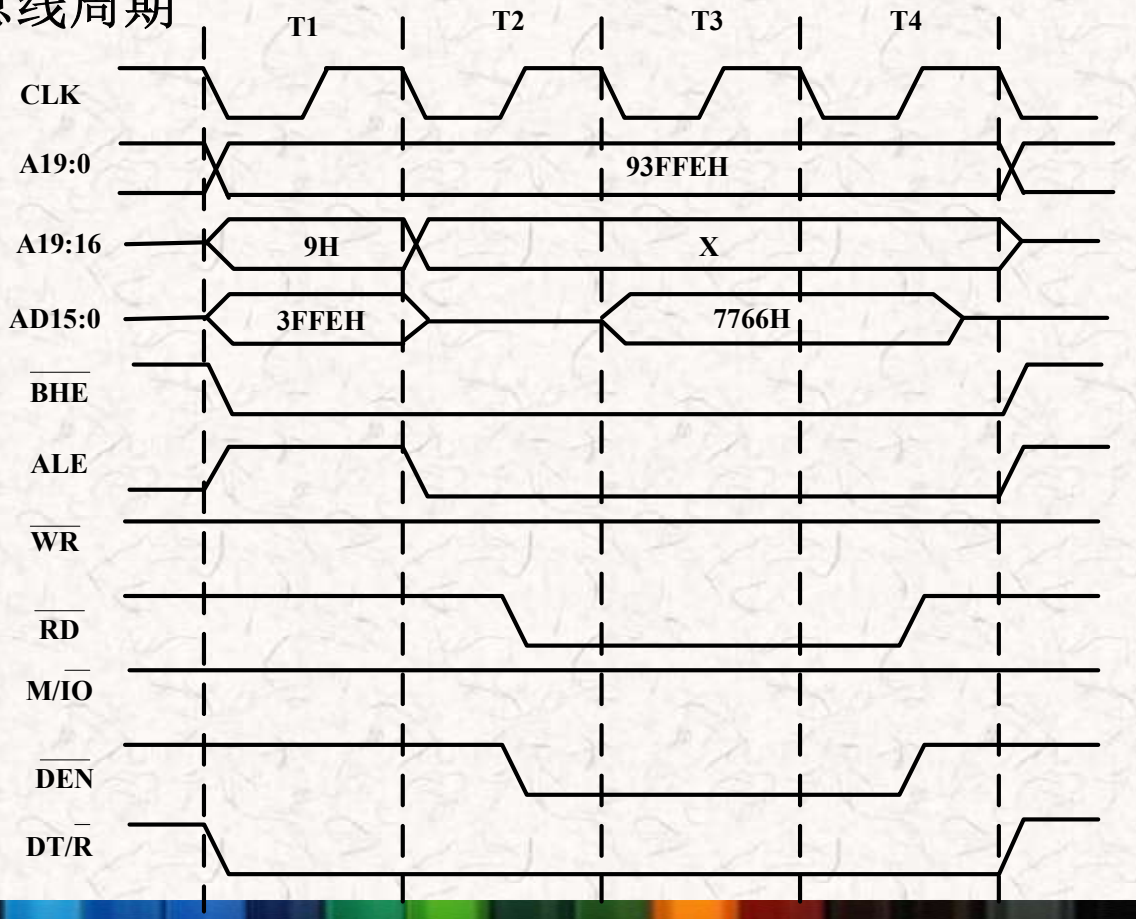
M/IO=1表明当前访问存储器

数据总线驱动器输出使能

控制数据总线驱动器方向为输入

读总线操作举例3-读偶地址字

若DS=9000H, 存储单元93FFEH存放66H， 93FFFH存放77H ， 执行 MOV BX, [3FFEH] 时产生如下总线周期



地址锁存器输出的20根地址总线

A0=0选中偶地址存储体

8086放弃总线驱动，存储器输出驱动，16位字通过D15:0传递

/BHE=0选中奇地址存储体

地址锁存使能

T2、T3中/WE=1表明不执行写操作

/RE=0表明执行读操作

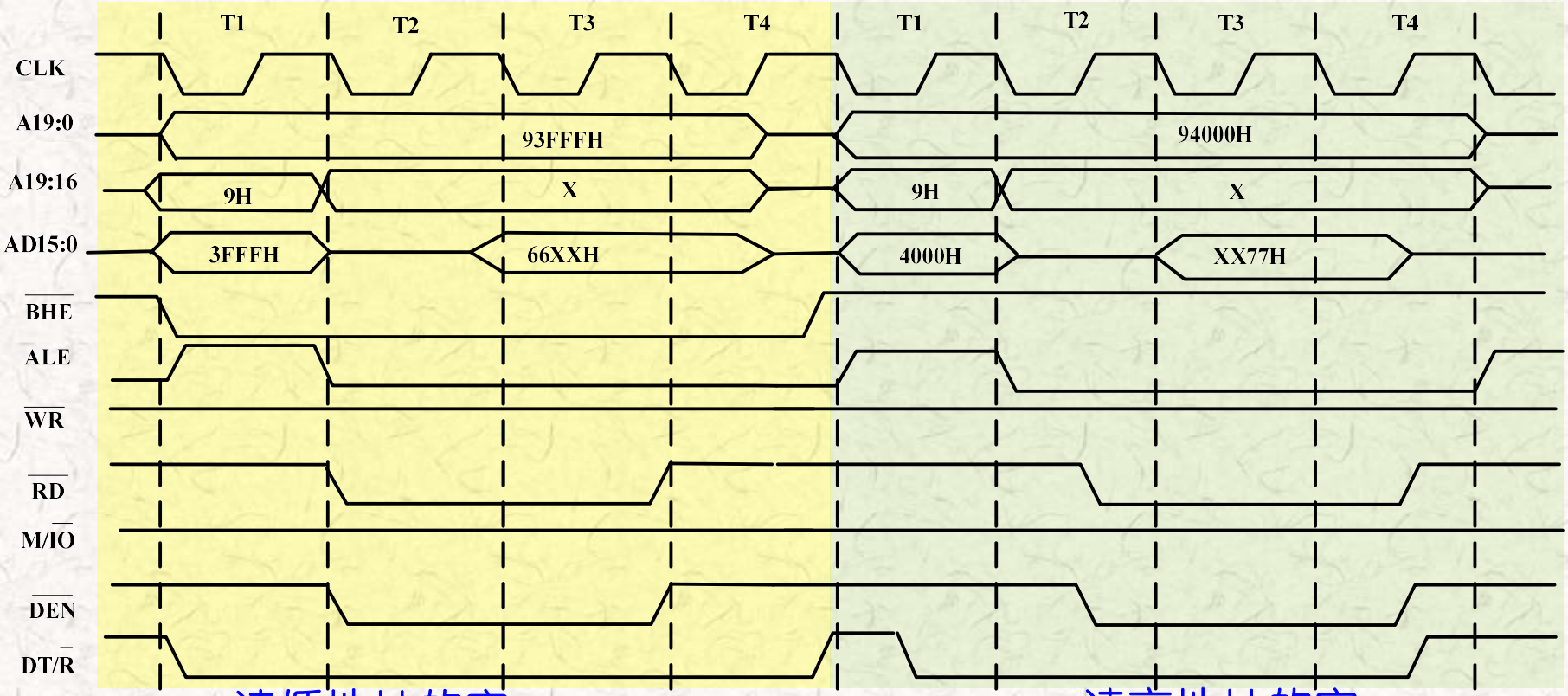
M/IO=1表明当前访问存储器

数据总线驱动器输出使能

控制数据总线驱动器方向为输入

读总线操作举例4-读奇地址字

若DS=9000H, 存储单元93FFFH存放66H， 94000H存放77H，执行MOV BX, [3FFFH] 时产生如下总线周期



练习1

- 1、8086的总线周期用于实现什么操作？
- 2、 8086CPU进行总线操作时，若M/$\overline{IO}$=1 , $\overline{WR}$=0, $\overline{RD}$=1, $\overline{BHE}$=0, A0=0，CPU此时对（存储器 or IO端口）的（奇 or 偶）地址进行（读 or 写）（字节 or 字）的操作
- 3、 8086一个最基本的总线周期包括____个T状态，最小模式下读总线周期中，数据在_____状态出现在数据总线上
- 1答案：对外部存储器和IO接口的读/写操作，如：取指令，或者读写数据
- 2答案：存储器 偶 写 字（16bit）
- 3答案：4 T3

总结1：本章学习了什么？

- 8086微处理器的内部结构：EU，BIU，通过指令队列并行工作
- 8086的运算部件是什么：ALU
- 8086外部数据总线、地址总线位宽：16位，20位
- 8086存储器空间大小，分段管理模式，物理地址和逻辑地址的概念与变换方式
- 8086标志位：ZF、CF、OF、SF、PF、AF
- 标志位受什么影响：当前ALU的运算结果
- 8086控制位：IF、DF、TF
- 可以作为地址指针的寄存器：BX、SI、DI、BP

总结2：本章学习了什么？

- **8086**堆栈的功能与特点：暂存数据、后进先出、硬件维护堆栈指针，**SS**，**SP**
- 为什么**8086**采用奇偶存储体的分体结构：可以实现字节或字的访问
- 奇偶存储体分别受什么信号选通：**/BHE=0**，**A0=0**
- 奇偶存储体如何与数据总线连接：奇存储体连接高**8**位，偶存储体连接低**8**位
- **/BHE**什么时候有效：需要选通奇存储体时，包括读/写奇地址字节，或读写字

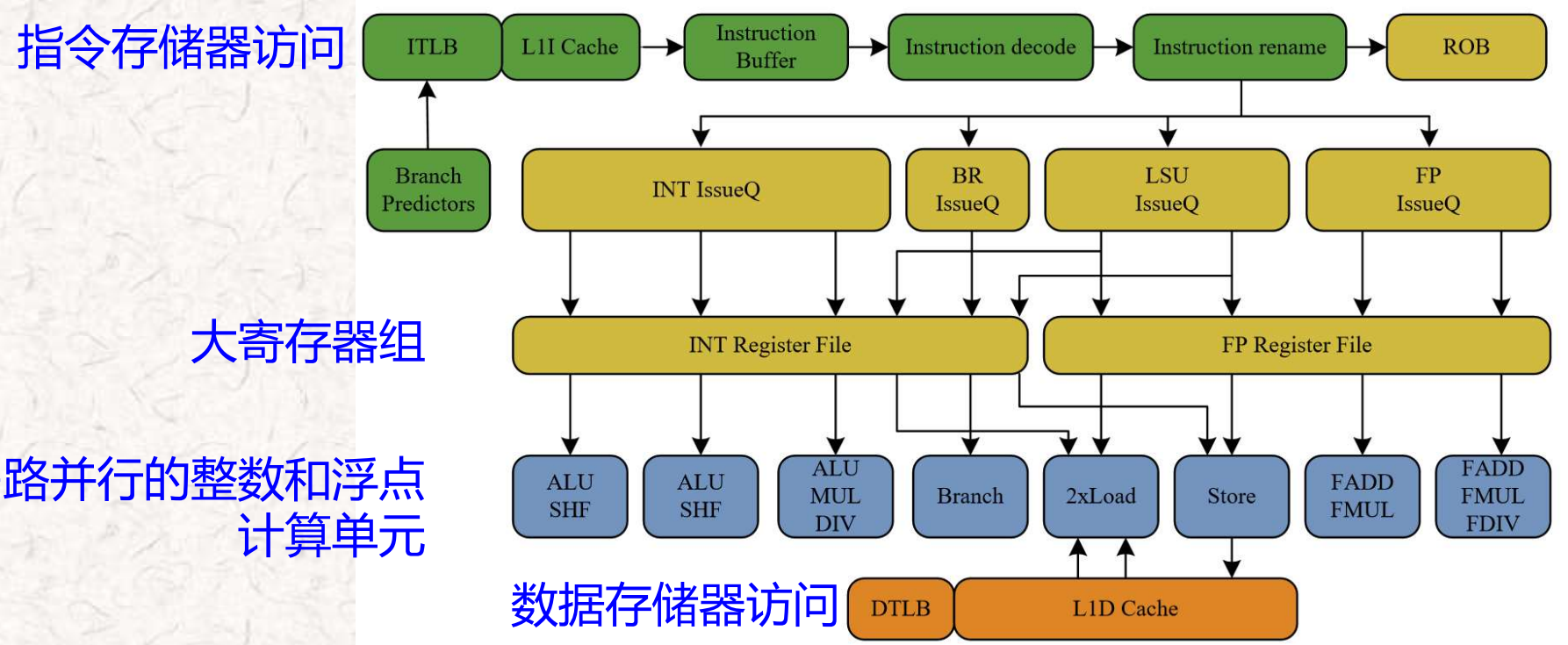
总结3：本章学习了什么？

- 8086如何生成外部的20bit地址总线和16bit数据总线；ALE控制地址锁存器，实现AD15:0 上地址和数据的分离，以及A19:16和/BHE状态的保持
- 总线周期实现什么操作：一个总线周期实现对一个地址的存储器或IO进行一次字节或字的读或写操作；
- 总线周期中各个时钟周期的具体操作：T1 A19:16、/BHE、AD15:0发送地址，T2、T3 AD15:0作为数据总线，传送数据，同时给出读或写信号，T4所有信号撤销；整个周期中M/IO信号为1，指示访问存储器，为0，指示访问I/O
- 读操作中来自存储器或IO的数据什么时候出现在数据总线上？T3

扩展知识1：现代微处理器的运算路径

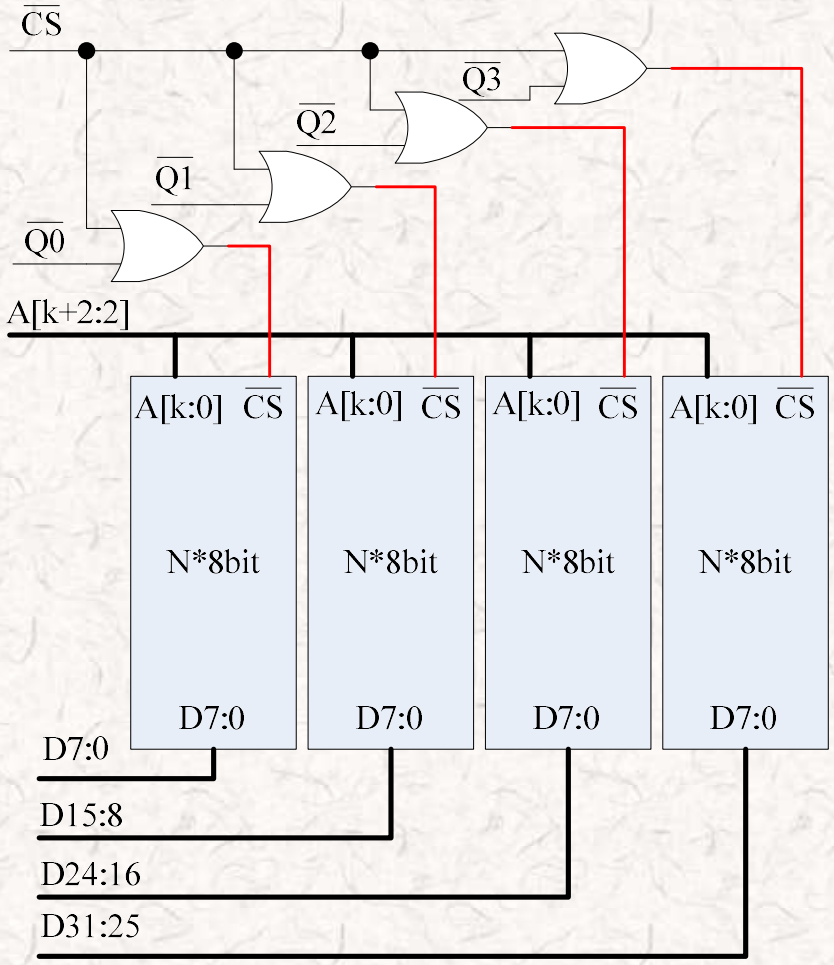
- 发展趋势：增强计算路径，降低流水线深度
- 更大的寄存器组，更多的计算路径

DMR：超标量处理器



扩展知识2：32位微处理器如何实现变位宽访问？

- 32位微处理器通常支持8位，16位和32位访问，因此地址空间以8bit字节为单位编址，增加字节选择信号/Q0~/Q3

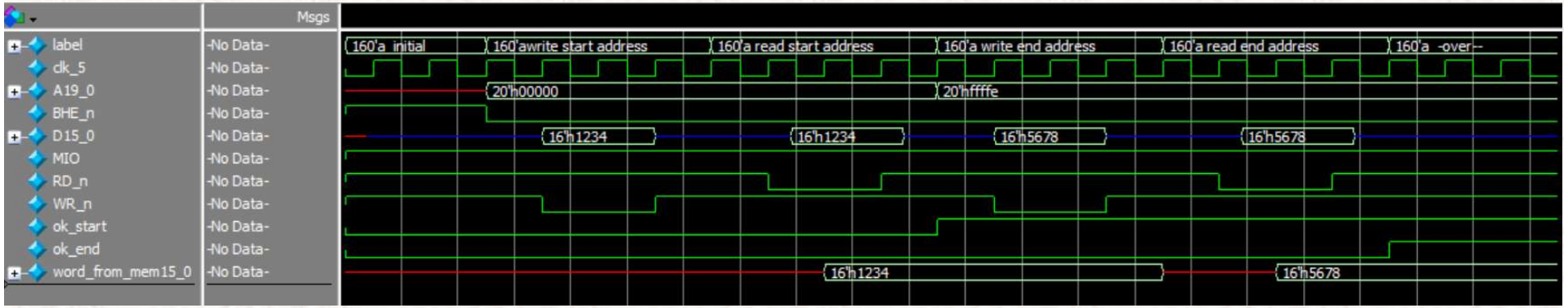


扩展知识3：非对齐存储器访问的实现方式

- 硬件实现：硬件自动实现两次访问并拼接结果
- 优点：快速
- 缺点：硬件复杂度高
- 异常中断：软件方式实现，检测到非对齐访问调用子程序实现两次访问和拼接
- 优点：硬件复杂度低
- 缺点：访问时间开销大
- 引申：微处理器不常用的复杂功能很多可以采用异常中断方式实现

练习2-存储体接入与总线波形

- **8086**奇偶存储体连接练习
- 在**Modelsim**中将已经定义的**512KB**的存储体与**8086**系统总线连接
- 运行代码，将可以观察到对地址**0**和**FFFF0H**的字读写的总线访问波形



还有问题吗？

作业

P20 1~12

13中有关**8086**的部分

预告-全面预习第三章和第四章

- **3.1** 寻址方式 重要
- **3.2** 机器码 了解
- **3.3** 指令系统 重要，
- 以下指令不需掌握：
- **XLAT、LAHF、SAHF、**
- 十进制调整指令：**AAA、DAA、AAS、DAS、AAM、AAD、**
- 串操作指令：**LOOP、LOOPE/LOOPZ、LOOPNE/LOOPNZ、JCXZ**
- **ESC、WAIT、LOCK**

预告-第四章需要掌握的内容

- **4.1** 掌握，其它运算符中只需了解（）和 **[ ]**
- **4.2 - 4.2.5** 掌握
- **4.3** 了解，常用**DOS**功能调用需掌握
- **4.4** 读懂程序，会画流程图
- **4.5** 不要求