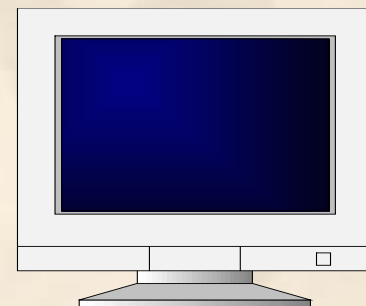


第2章

Intel x86微处理器



主要内容：

■ 8086/8088微处理器

- 特点
- 主要引线功能和内部结构
- 内部寄存器
- 存储器寻址
- 总线时序

■ IA-32微处理器

- ◆ IA-32微处理器结构
- ◆ IA-32微处理器工作方式
- ◆ 保护模式下的存储器访问



一、8086/8088特点及工作模式

1. 8088/8086 CPU的特点

■ 采用并行流水线工作方式 · · ·

CPU内部结构

—— 将CPU划分成两个功能部分并设置指令预取队列，实现流水线工作

■ 对内存空间实行分段管理 · · ·

存储器寻址部分

—— 将内存分为多个段并设置4个段地址寄存器，实现对1MB空间的寻址

■ 支持多处理器系统 · · ·

工作模式

—— 可以包含主处理器和协处理器

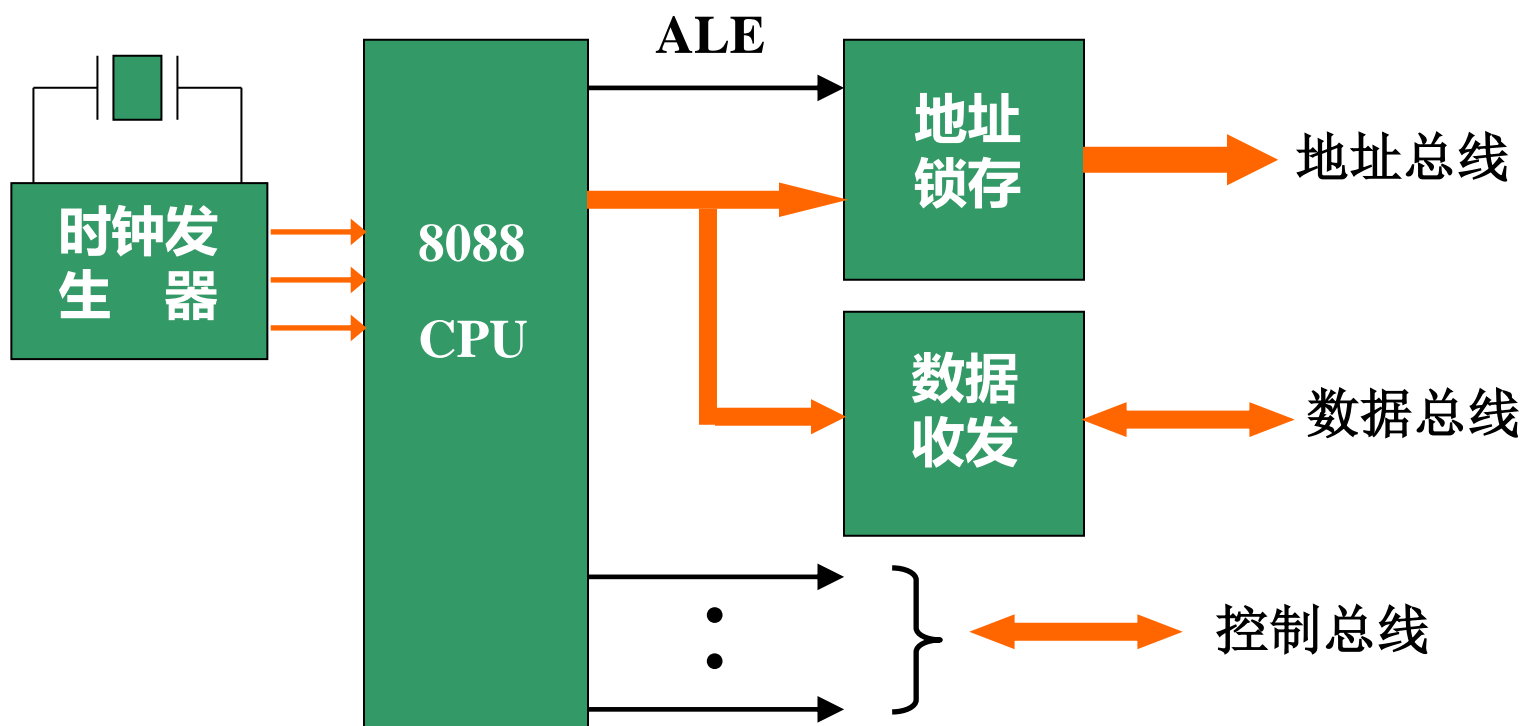
2. 8088CPU的两种工作模式

- 8088可工作于两种模式下

 **最小模式**
最大模式

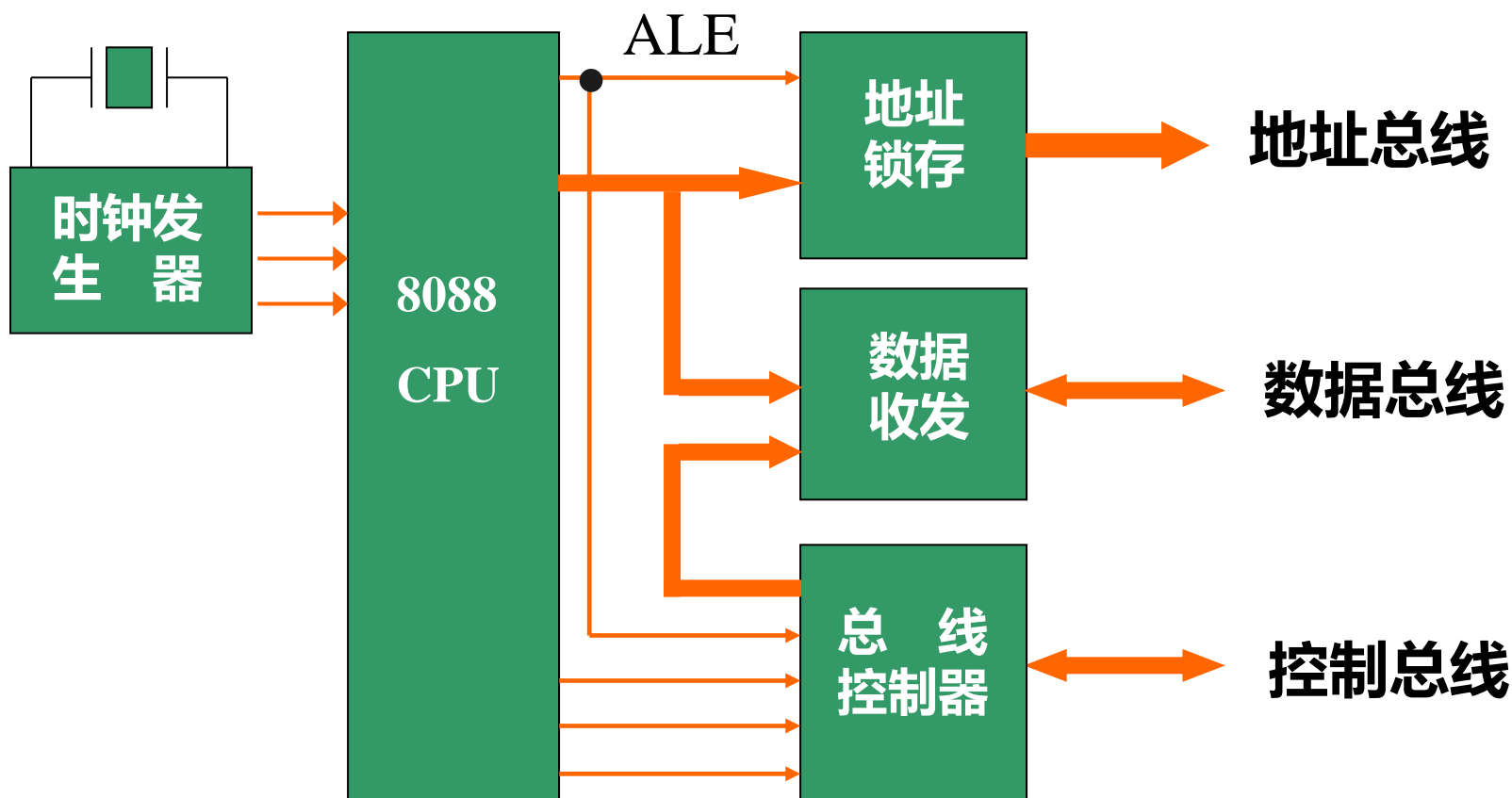
- **最小模式为单处理器模式，控制信号较少，一般可不必接总线控制器，CPU直接输出控制信号到总线。**

最小模式下的总线连接示意图



- 最大模式为多处理器模式，即CPU+协处理器。
- 控制信号较多，须通过总线控制器与总线相连。

最大模式下的总线连接示意图



两种工作模式的选择

- 8088是工作在最小还是最大模式由CPU上的 $\overline{MN}/\overline{MX}$ 引脚的输入决定。
 - $\overline{MN}/\overline{MX}=0$ ——工作于最大模式
 - $\overline{MN}/\overline{MX}=1$ ——工作于最小模式



二、8088/8086的引线及功能

8086和8088CPU都有40条引脚，8086外部数据线有16条，而8088只有8条，因此它们与存储器或外设接口连接时一次可传送的数据位数分别是16位和8位。

1. 主要引线——最小模式下的8088引线

■ 地址线和数据线：

- AD_0-AD_7 ：低8位地址和低8位数据信号分时复用。在传送地址信号时为单向，传送数据信号时为双向。
- A_8-A_{15} ：8位地址信号（8086为地址与数据复用线）
- $A_{16}-A_{19}/S_3-S_6$ ：高4位地址信号，与状态信号分时复用。

主要的控制和状态信号线

- $\overline{\text{WR}}$: 写信号;
- $\overline{\text{RD}}$: 读信号;
- $\text{IO}/\overline{\text{M}}$: 为“0”表示访问内存,
为“1”表示访问接口;
- $\overline{\text{DEN}}$: 低电平有效时, 表示数据总线上数据有效, 允许进行读/写操作;
- $\text{DT}/\overline{\text{R}}$: 数据收发器的传送方向控制, 为“1”时 CPU向存储器或I/O传送, 否则为反向;

- **ALE：地址锁存信号，当其为高时表示地址线上地址有效。一般用它将地址锁存到一个锁存器中；**
- **RESET：复位信号。当其为高时将完成CPU内部复位。复位后CPU内部寄存器的值如下表。**

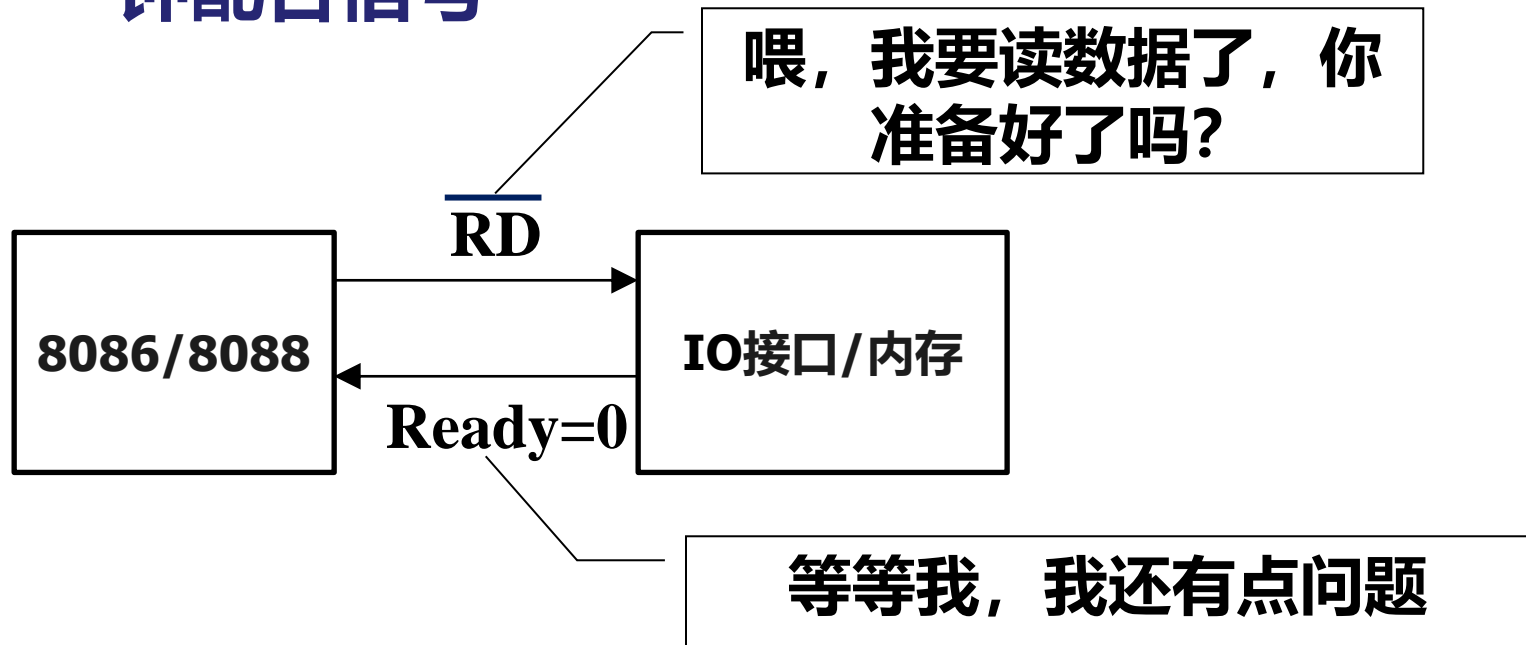
内部寄存器	内容	内部寄存器	内容
CS	FFFFH	IP	0000H
DS	0000H	FLAGS	0000H
SS	0000H	其余寄存器	0000H
ES	0000H	指令队列	空

例：根据信号的取值判断工作状态

- 当 $\overline{WR}=1$, $\overline{RD}=0$, $IO/\overline{M}=0$ 时,
表示CPU当前正在进行读存储器操作

READY

- 外部同步控制输入信号，高电平有效
- 8088与内存/外设之间在一个总线周期内的时钟配合信号



中断请求和响应信号

- **INTR**: 可屏蔽中断请求输入端
- **NMI**: 非屏蔽中断请求输入端
- **$\overline{\text{INTA}}$** : 中断响应输出端

2. 8088和8086CPU引线的差异

- 数据总线宽度不同

- 8088的外部数据总线宽度是8位，8086为16位。

- 访问存储器和I/O控制的信号高低有效不同

- 8088—— $\overline{IO}/\overline{M}=0$ 表示访问内存；
- 8086—— $\overline{IO}/M=1$ 表示访问内存。



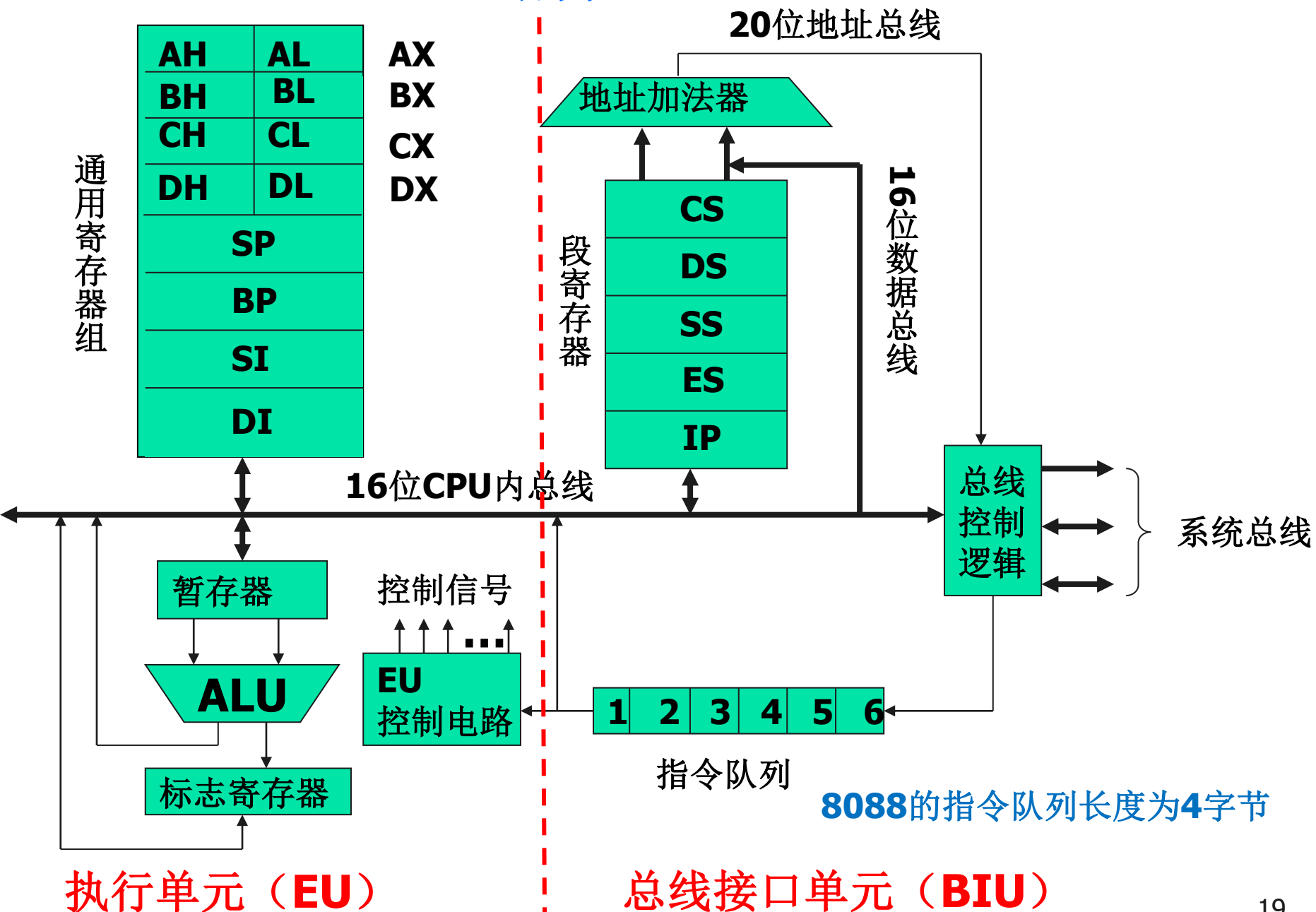
三、8088/8086的内部结构

1. 组成

- 8088/8086内部由两部分组成:

 执行单元 (EU)
总线接口单元 (BIU)

8086CPU结构



2. 执行单元

运算器(ALU)

8个通用寄存器

1个标志寄存器

EU部分的控制电路

■ 功能 → 指令的执行

- 指令译码

- 指令执行 → 在**ALU**中完成

- 暂存中间运算结果 → 在通用寄存器中

- 保存运算结果特征 → 在标志寄存器**FLAGS**中

3. 总线接口单元

地址加法器

4个段寄存器

指令指针IP

总线控制逻辑

功能:

- 从内存中取指令到指令队列
 - 指令队列是并行流水线工作的基础
- 负责与内存或I/O接口之间的数据传送
- 在执行转移程序时，BIU清除指令队列，从指定的新地址取指令，并立即传给执行单元执行。

结论


- **指令队列的存在使EU和BIU两个部分可并行工作，即：**
 - **实现指令的并行执行**
- **目的：**
 - **提高了CPU的效率；**
 - **降低了对存储器存取速度的要求**



四、内部寄存器

内部寄存器的类型

- 含14个16位寄存器，按功能可分为三类

 8个通用寄存器
4个段寄存器
2个控制寄存器

要求： 深入理解每个寄存器的作用

1. 通用寄存器

 **数据寄存器 (AX, BX, CX, DX)**
地址指针寄存器 (SP, BP)
变址寄存器 (SI, DI)

数据寄存器

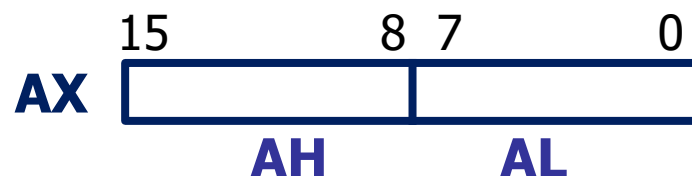
- 8088/8086含4个16位数据寄存器，每一个又可拆分为2个8位寄存器，即：

- AX → AH, AL

- BX → BH, BL

- CX → CH, CL

- DX → DH, DL



数据寄存器特有的固有用法

- **AX：**累加器。所有I/O指令都通过AX与接口传送信息，中间运算结果也多放于AX中；
- **BX：**基址寄存器。在间接寻址中用于存放基地址；
- **CX：**计数寄存器。用于在循环或串操作指令中存放计数值；
- **DX：**数据寄存器。在间接寻址的I/O指令中存放I/O端口地址；在32位乘除法运算时，存放高16位数。

地址指针寄存器

- **SP**: 堆栈指针寄存器, 其内容为栈顶的偏移地址;
- **BP**: 基址指针寄存器, 常用于在访问内存时存放内存单元的偏移地址。

BX与BP在应用上的区别

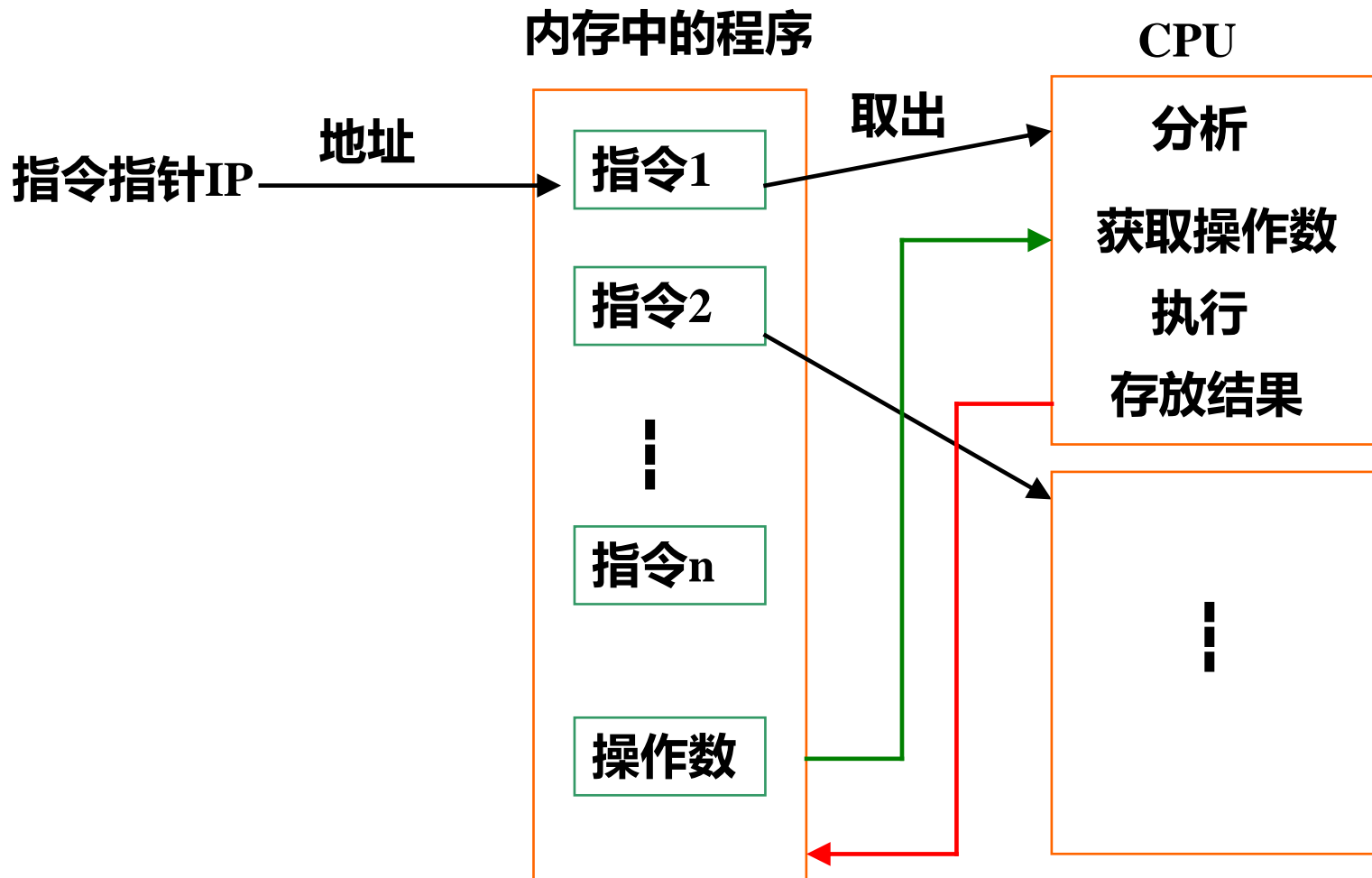
- 作为通用寄存器，二者均可用于存放数据；
- 作为基址寄存器，用BX表示所寻找的数据在数据段；用BP则表示数据在堆栈段。

变址寄存器

- **SI：源变址寄存器**
- **DI：目标变址寄存器**
- **变址寄存器在指令中常用于存放数据在内存中的地址。**

2. 控制寄存器—IP与FLAGS

- IP: 指令指针寄存器，其内容为下一条要执行指令的偏移地址。



标志寄存器FLAGS

作用：存放运算结果的状态特征和控制CPU的运行

6个状态标志位

- **CF (Carry Flag)**

- 进位标志位。加(减)法运算时，若最高位有进(借)位则CF=1

- **PF (Parity Flag)**

- 奇偶标志位。运算结果的低8位中“1”的个数为偶数时PF=1

- **AF (Auxiliary Carry Flag)**

- 辅助进位标志位。加(减)操作中，若Bit3向Bit4有进位(借位)，AF=1

- **ZF (Zero Flag)**

- 零标志位。当运算结果为零时ZF=1

- **SF (Sign Flag)**

- 符号标志位。当运算结果的最高位为1时，SF=1

- **OF (Overflow Flag)**

- 溢出标志位。当算术运算的结果超出了有符号数的可表达范围时，OF=1

状态标志位设置举例

- 给出以下运算结果及运算后各状态标志位的状态：
 - $10110110 + 11110100$

$$\begin{array}{r} 10110110 \\ + 11110100 \\ \hline \boxed{1} 10101010 \end{array}$$

CF = 1 OF = 0

AF = 0 PF = 1

SF = 1 ZF = 0

3个控制标志位

■ TF (Trap Flag)

- 陷阱标志位，也叫跟踪标志位。TF=1时，使CPU处于单步执行指令的工作方式。

■ IF (Interrupt Enable Flag)

- 中断允许标志位。IF=1使CPU可以响应可屏蔽中断请求。

■ DF (Direction Flag)

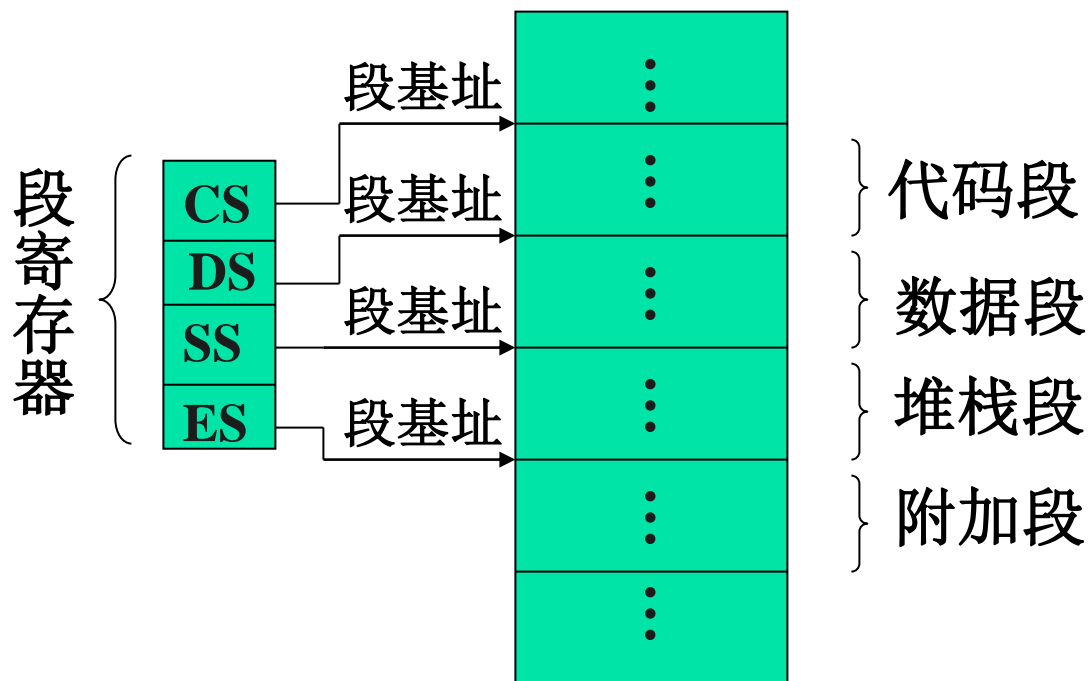
- 方向标志位。在数据串操作时确定操作的方向。

3. 段寄存器

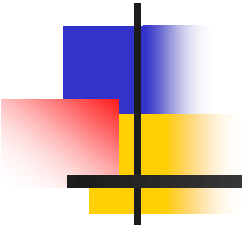
■ 作用：用于存放相应逻辑段的段基地址

- ✓ 8086/8088CPU在使用存储器时，将它划分成若干个逻辑段。
- ✓ 每个逻辑段用来存放不同目的内容，如程序代码、数据等等。
- ✓ 每个逻辑段用一个段寄存器来指明该段的起始位置（也叫段基址）。
- ✓ 8086/8088内存中逻辑段的数量最多为64K个
- ✓ 程序中同时可以使用4个段，分别由CS、DS、ES和SS四个段寄存器指示。

- **CS:** 代码段寄存器, 存放代码段的段基址。
- **DS:** 数据段寄存器, 存放数据段的段基址。
- **ES:** 附加段寄存器, 存放数据段的段基址。
- **SS:** 堆栈段寄存器, 存放堆栈段的段基址。



五、8086/8088的存储器组织



1. 内存单元的编址

内存单元的地址表示有物理地址和逻辑地址两种方式

■ 物理地址

- 指每个内存单元在整个内存空间中具有的惟一的地址。
- 8086/8088CPU有20根地址线，它可以产生20位的地址码，寻址范围为 2^{20} ，即1兆字节空间。

这一兆字节存储单元的地址范围为： $\underbrace{00\dots\dots0}_{20\text{位}}\sim\underbrace{11\dots\dots1}_{20\text{位}}$ 。

为了方便书写，在源程序中常用**5**位十六进制数或一个符号来表示一个存储单元的地址。

十六进制数地址	二进制数地址	存储单元（字节）
		70
00000H	000000000000000000000000	
00001H	000000000000000000000001	
00002H	000000000000000000000010	
⋮	⋮	⋮
FFFFEH	111111111111111111111110	
FFFFFH	111111111111111111111111	

- 任何两个相邻字节单元就构成一个字单元
- 字单元的地址为两个字节单元中较小地址字节单元的地址。
- 字数据的存放规则是低8位放在较低地址字节单元，高8位放在较高地址字节单元。 是大端还是小端方式? 小端

例如，将数据3456H放在地址为09235H的存储单元中的存储分配。

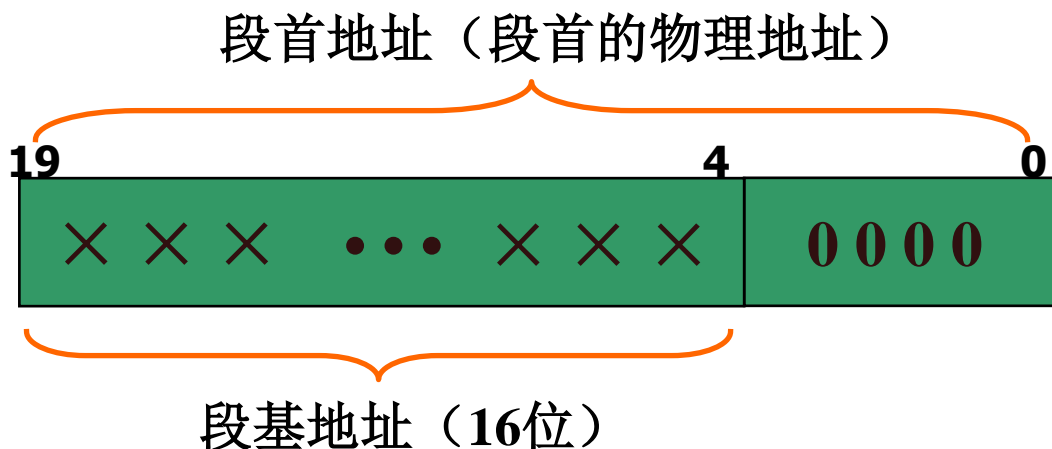
地址	存储单元
	⋮
09235H	56
09236H	34
	⋮

■ 逻辑地址

每个存储单元的逻辑地址由两部分组成

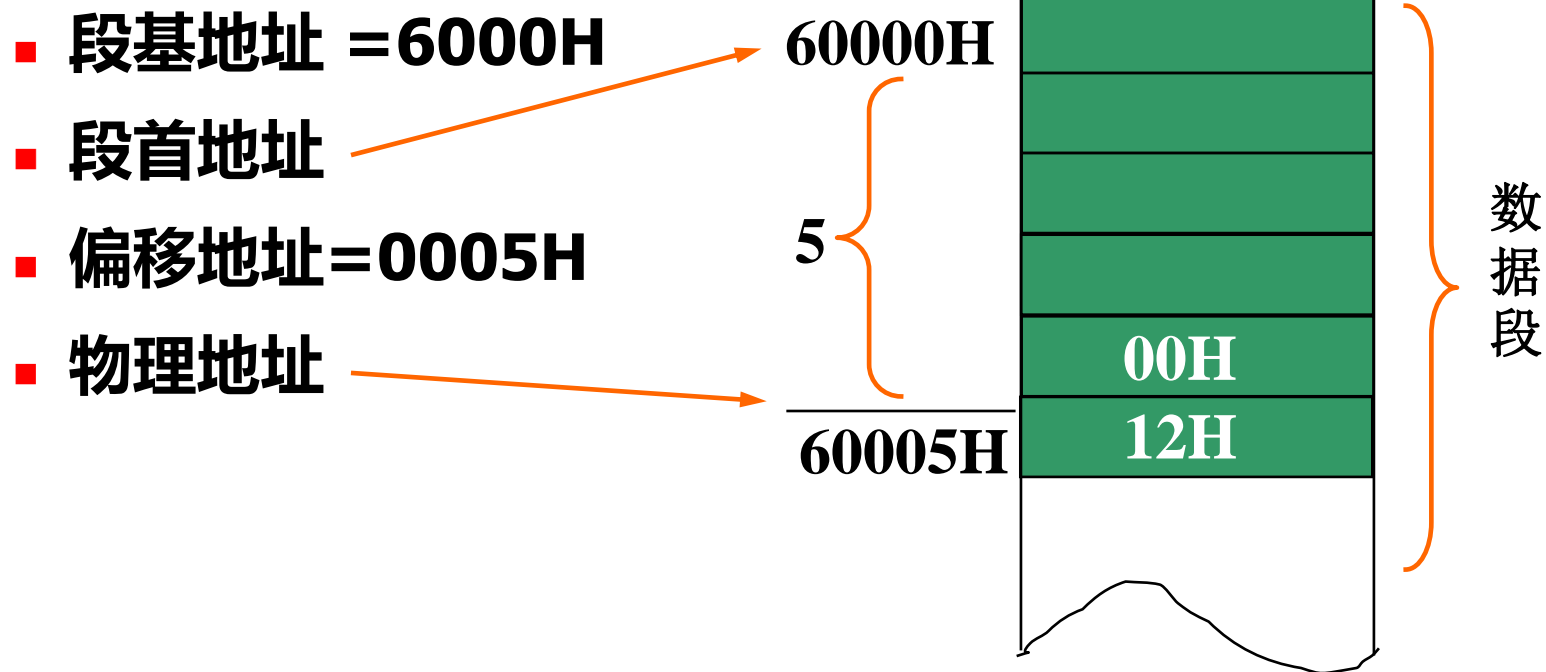
- 16位的段（基）地址
---决定该逻辑段在内存中的起始位置
- 16位的段内地址，也叫相对地址，或偏移地址
---决定该存储单元相对段起始单元的距离

逻辑段的起始单元称为段首，段首的偏移地址=0



- 将内存划分成多个逻辑段后，就可以使用逻辑地址来指示存储单元。
- 使用逻辑地址方便了程序的开发和对存储器进行动态管理。

例：



■ 8086/8088的存储器段结构的特点:

1. 每个段最大长度为64K（65536）个字节单元组成。

2. 每个段的起始地址（段首）必须是一个小节的首址。

从0地址开始，每16个字节单元称为一个**小节（Paragraph）**
1MB内存就可划分为64K个小节。

第 1 小节:	00000H,	00001H,	00002H.....0000FH
第 2 小节:	00010H,	00011H,	00012H.....0001FH
⋮	⋮	⋮	⋮
第65535小节:	FFFE0H	FFFE1H	FFFE2H.....FFFEFH
第65536小节:	FFFF0H	FFFF1H	FFFF2H.....FFFFFH

每个小节的首地址最低位必为0（16进制数表示）。因此段首只能是上述64K个小节首址之一。

例：

已知 CS=1055H, DS=250AH,
ES=2EF0H, SS=8FF0H

画出各段在内存中的分布。

CS=1055H

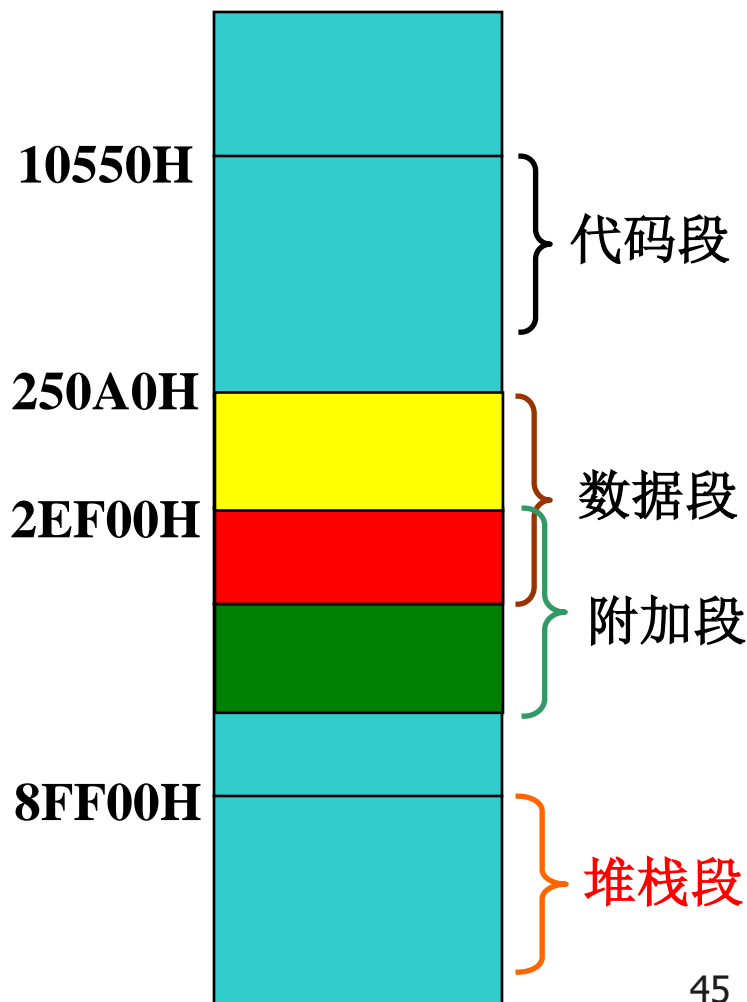
- 段首地址=10550H

DS=250AH

- 段首地址=250A0H

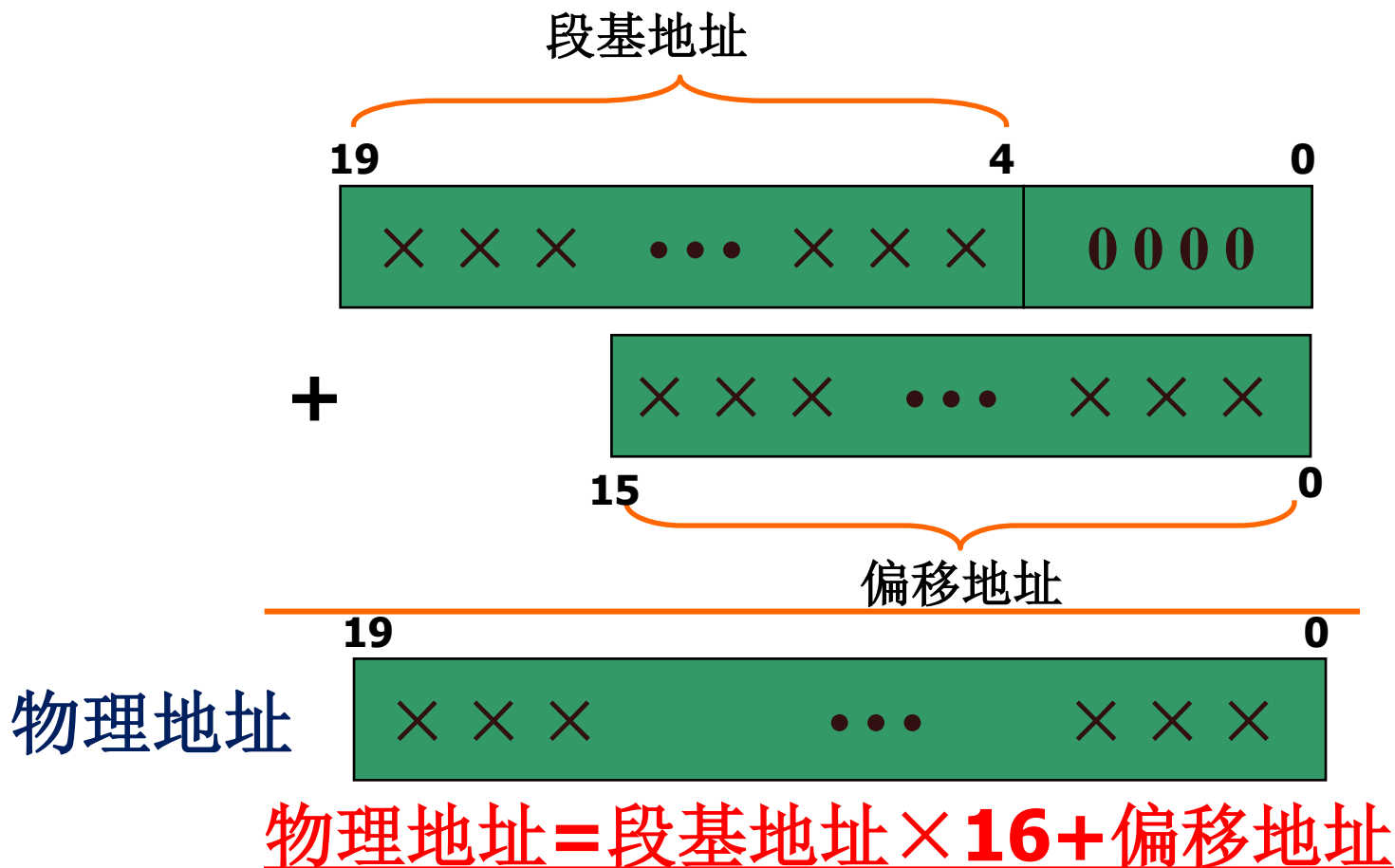
ES=2EF0H

SS=8FF0H



2. 逻辑地址与物理地址的转换

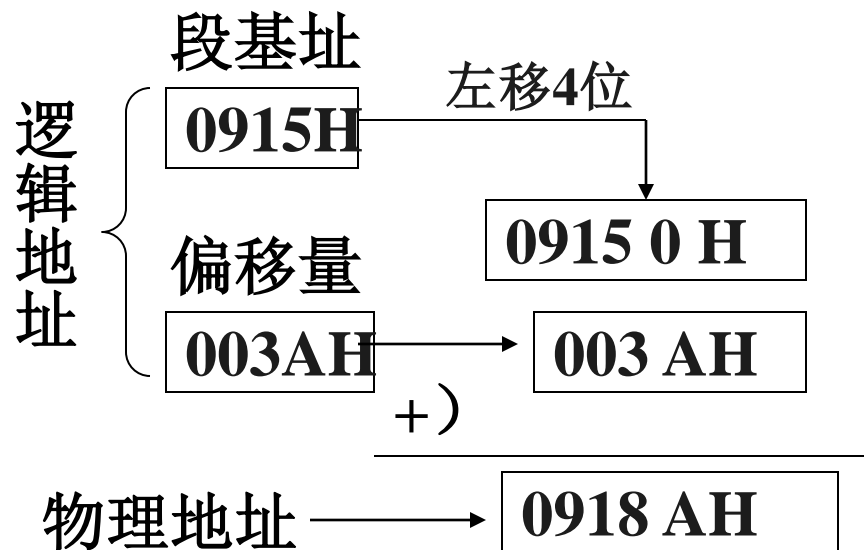
- 内存物理地址由段基地址和偏移地址组合而成



举例

- 例1，设某操作数存放在数据段，DS=250AH，数据所在单元的偏移地址=0204H。则该操作数所在单元的物理地址为：
 - $250AH \times 16 + 0204H = 252A4H$

例2:



例3: 同一个物理地址
002D3H被两个逻辑段中的
逻辑地址映射的情况。

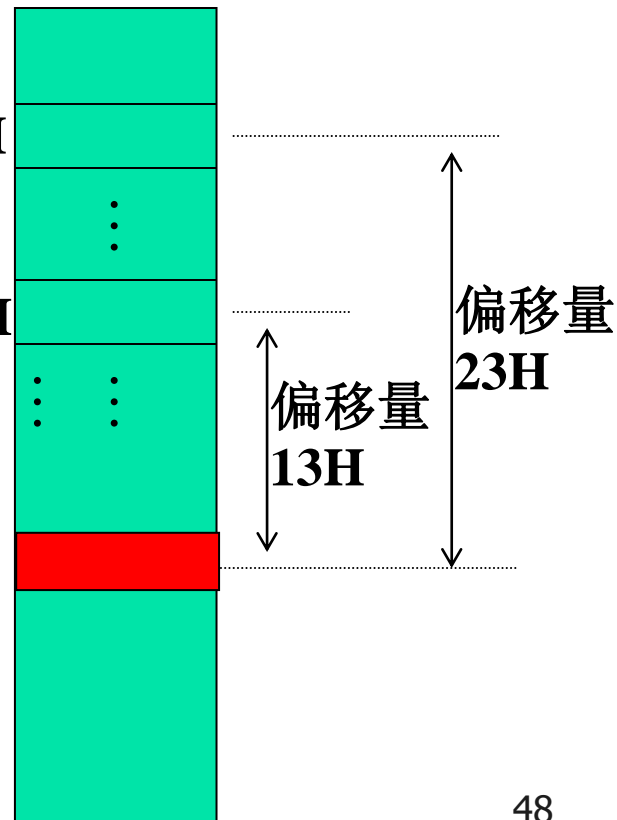
$$002B0H + 00023H = 002D3H$$

$$002C0H + 00013H = 002D3H$$

段1首址 **002B0H**

段2首址 **002C0H**

002D3H

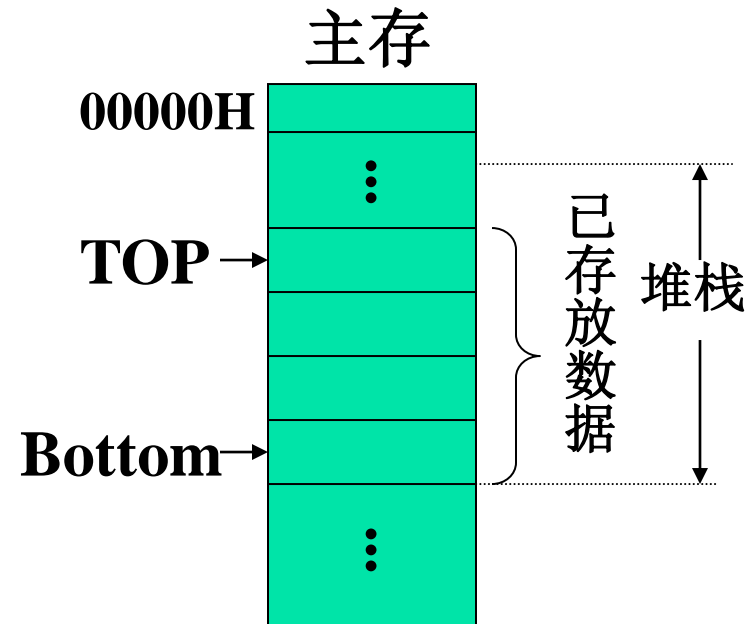


4. 堆栈及堆栈段的使用

■ 堆栈定义

- 堆栈是一个特定的存储区，访问该存储区一般需要按照专门的规则进行操作。
- 主要用于暂存数据以及在过程调用或处理中断时保存断点信息。

- 堆栈的一端固定，称为**栈底**。
栈底是堆栈存储区的最大地址单元。
- 堆栈的另一端浮动，称为**栈顶**。任何时候，栈顶是最后存入信息的存储单元。
- 栈顶位置随着堆栈中存放信息的多少而改变。



堆栈指针**SP**（**Stack Pointer**）：

设置的一个寄存器，用来指示**栈顶**的位置，其内容就象一个指针一样。

◆ **SP**的内容始终指向栈顶单元

◆ 堆栈中数据的进出都由**SP**来控制

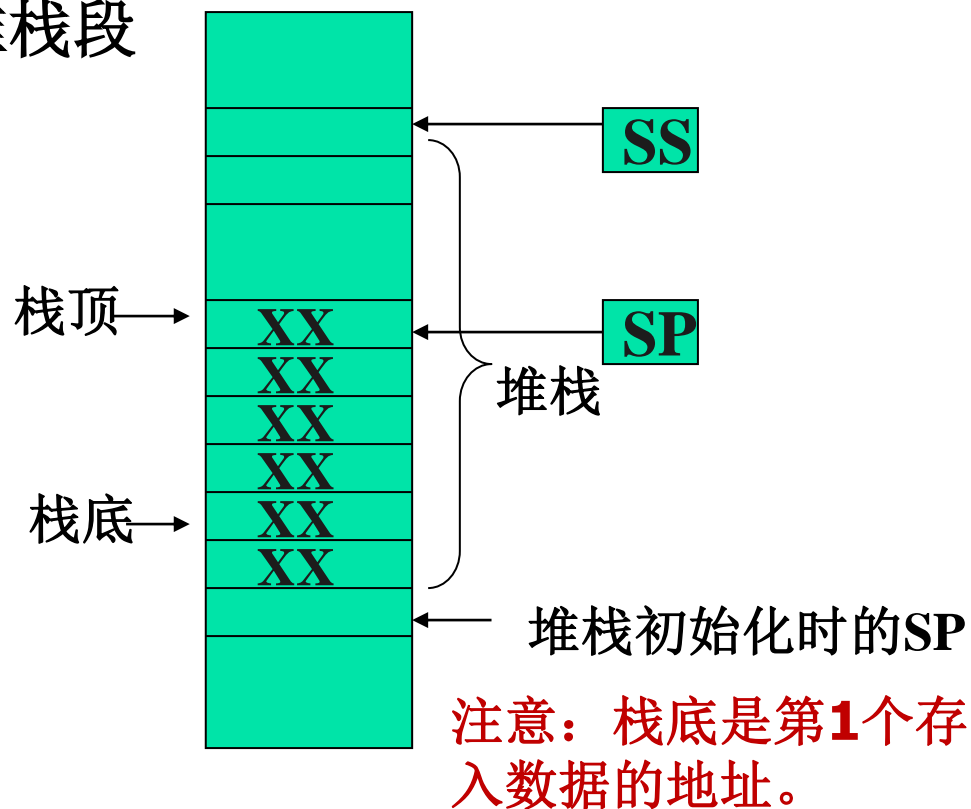
在堆栈中存取数据的规则：

“**先进后出FILO**”（**First-In Last-Out**）。即最先送入堆栈的数据要到最后才能取出，而最后送入堆栈的数据，最先取出。

8086/8088堆栈的组织

在8086/8088微机中堆栈是由堆栈段寄存器SS指示的一段存储区。

- 数据在堆栈中以字为单位存放，低8位放在较低地址单元，高8位放在较高地址单元。
- 堆栈初始化时SP指向栈底+2单元，其值就是堆栈的长度。由于SP是16位寄存器，因此堆栈长度 $\leq 64\text{K}$ 字节。



- SP始终表示堆栈段首与栈顶之间的距离（字节数）。
 - 当SP为最大(初始)值时，表示堆栈为空。
 - 当SP为0时，表示堆栈全满。

例：

■ 已知

■ $SS=1000H$, $SP=0100H$

■ 则：

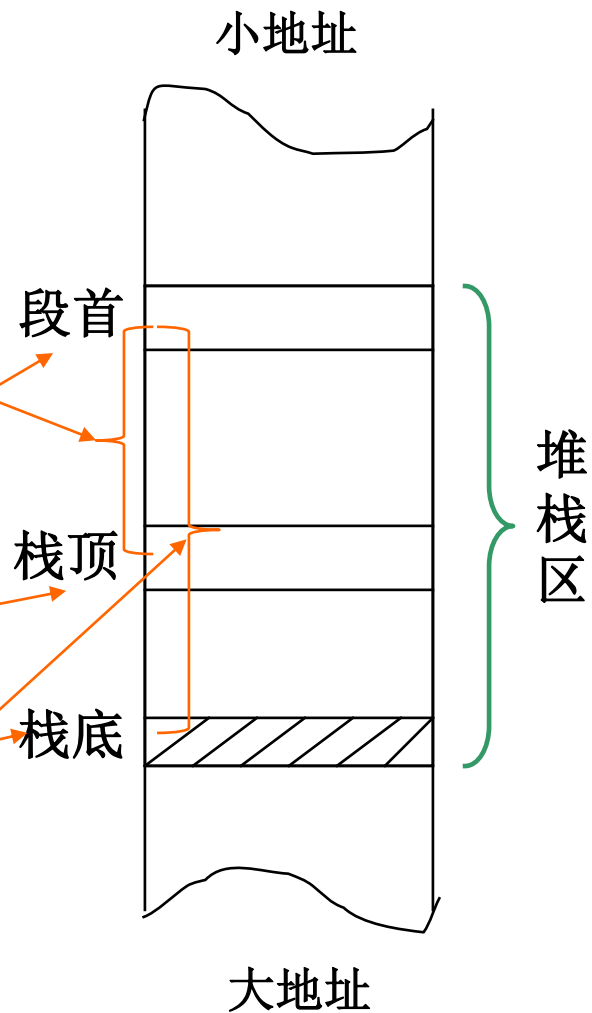
■ 堆栈段的段首地址 = $10000H$

■ 栈顶地址 = $SS * 16 + SP$
 $= 10100$

■ 若该段最后一个单元地址为
 $10200H$

则：栈底偏移地址 = $0200H$

堆栈初始化时 $SP = ?$ $202H$





六、8088系统总线

主要内容：

总线的基本概念；

最小总线模式；

最大总线模式；

最小模式总线时序。

1. 概述

■ 总线定义：

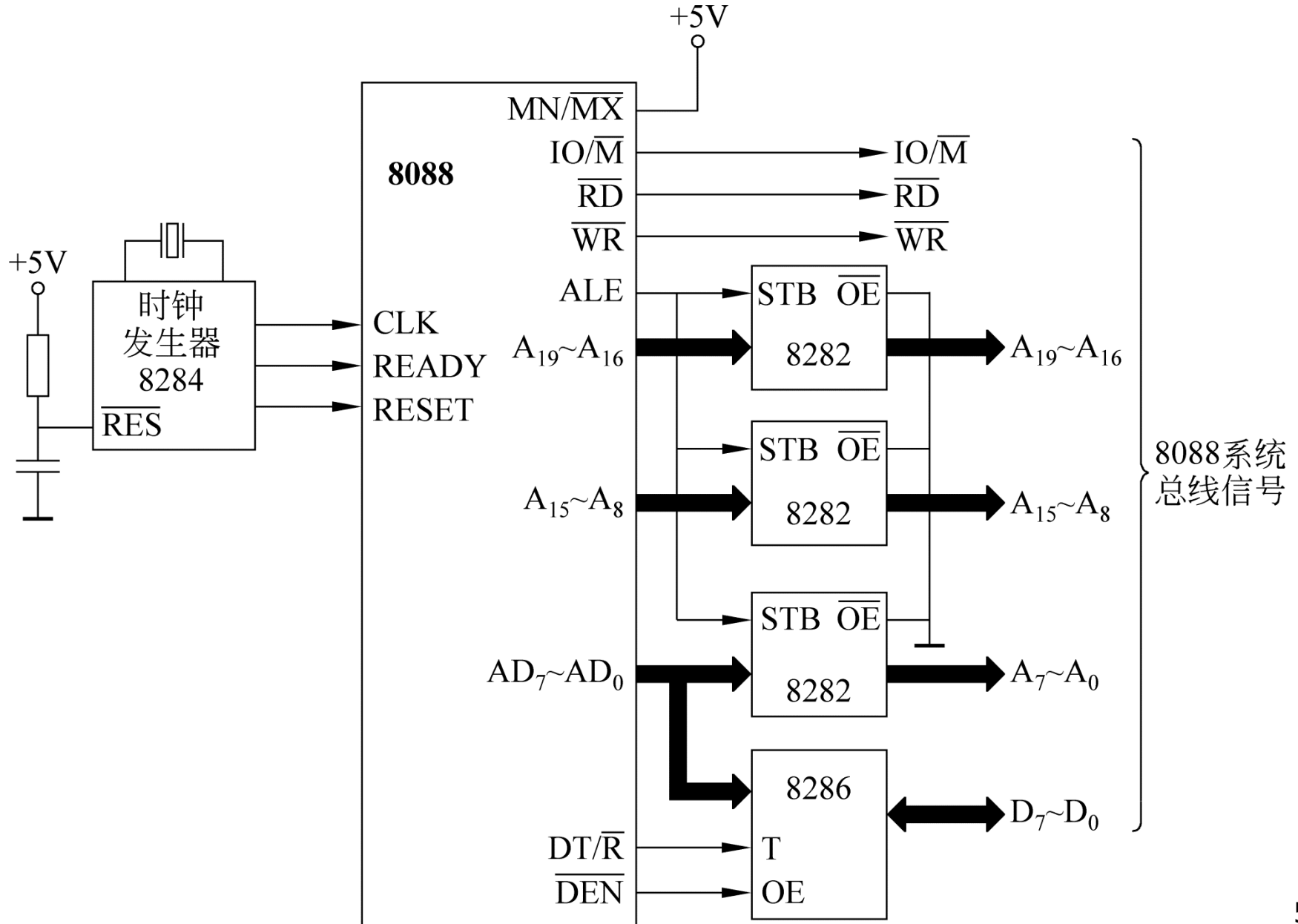
是一组导线和相关的控制、驱动电路的集合，它是计算机系统各部件之间传输地址、数据和控制信息的通道。

总线中可能包括：

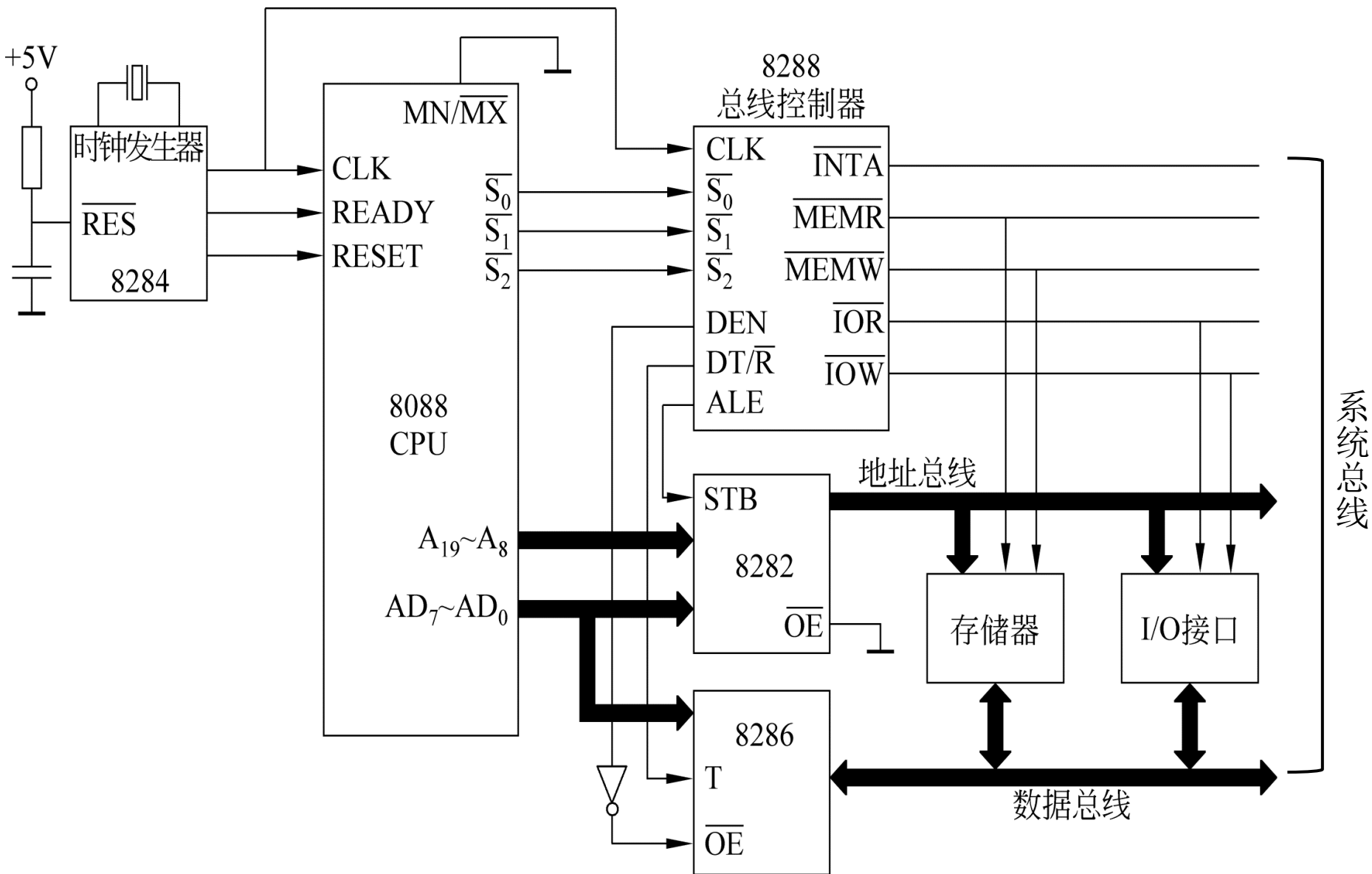


- 地址总线（AB）
- 数据总线（DB）
- 控制总线（CB）

2. 最小模式下的系统总线



3. 最大模式下的系统总线



4.总线时序

- 时序:

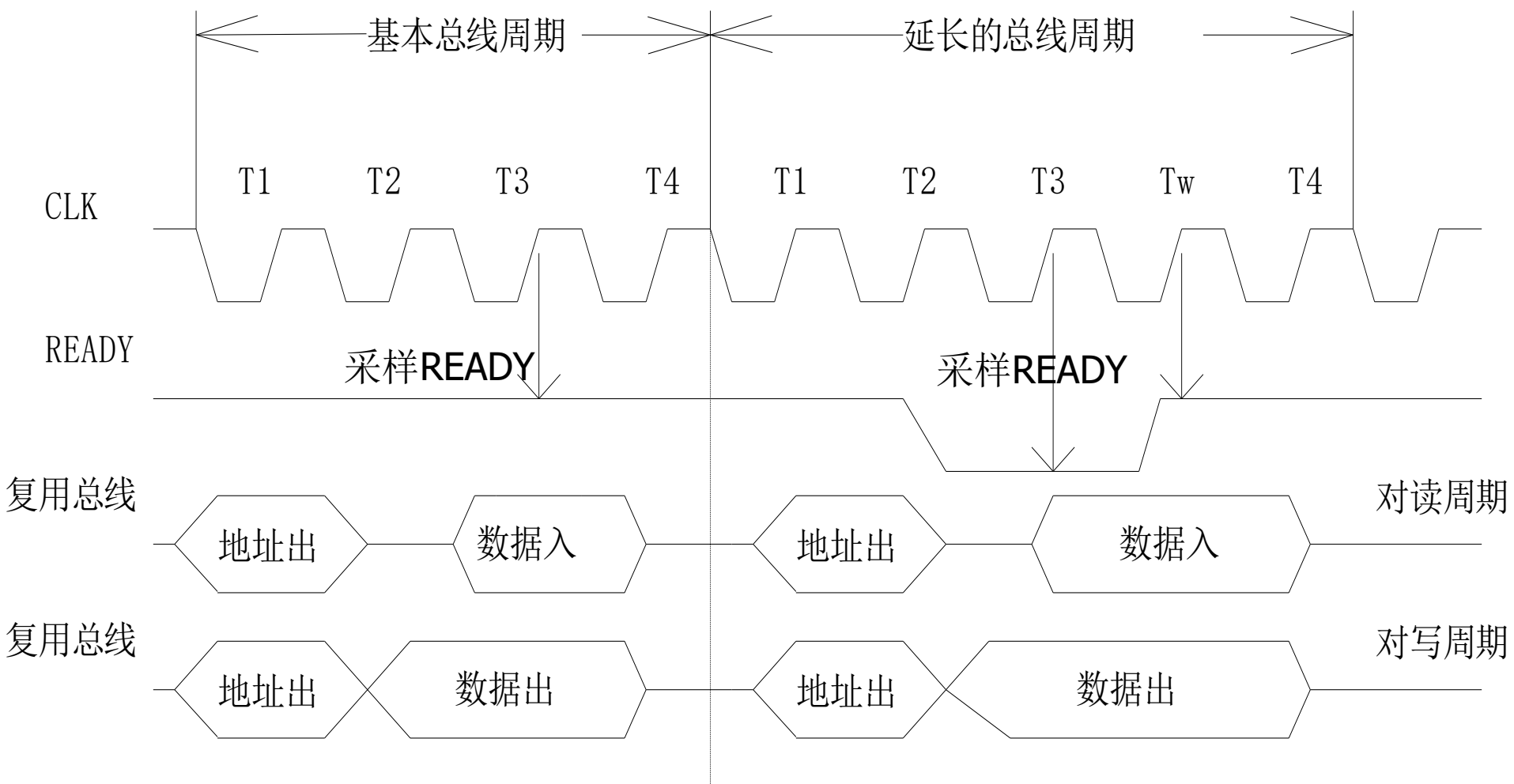
- CPU各引脚信号在时间上的关系

- 总线周期:

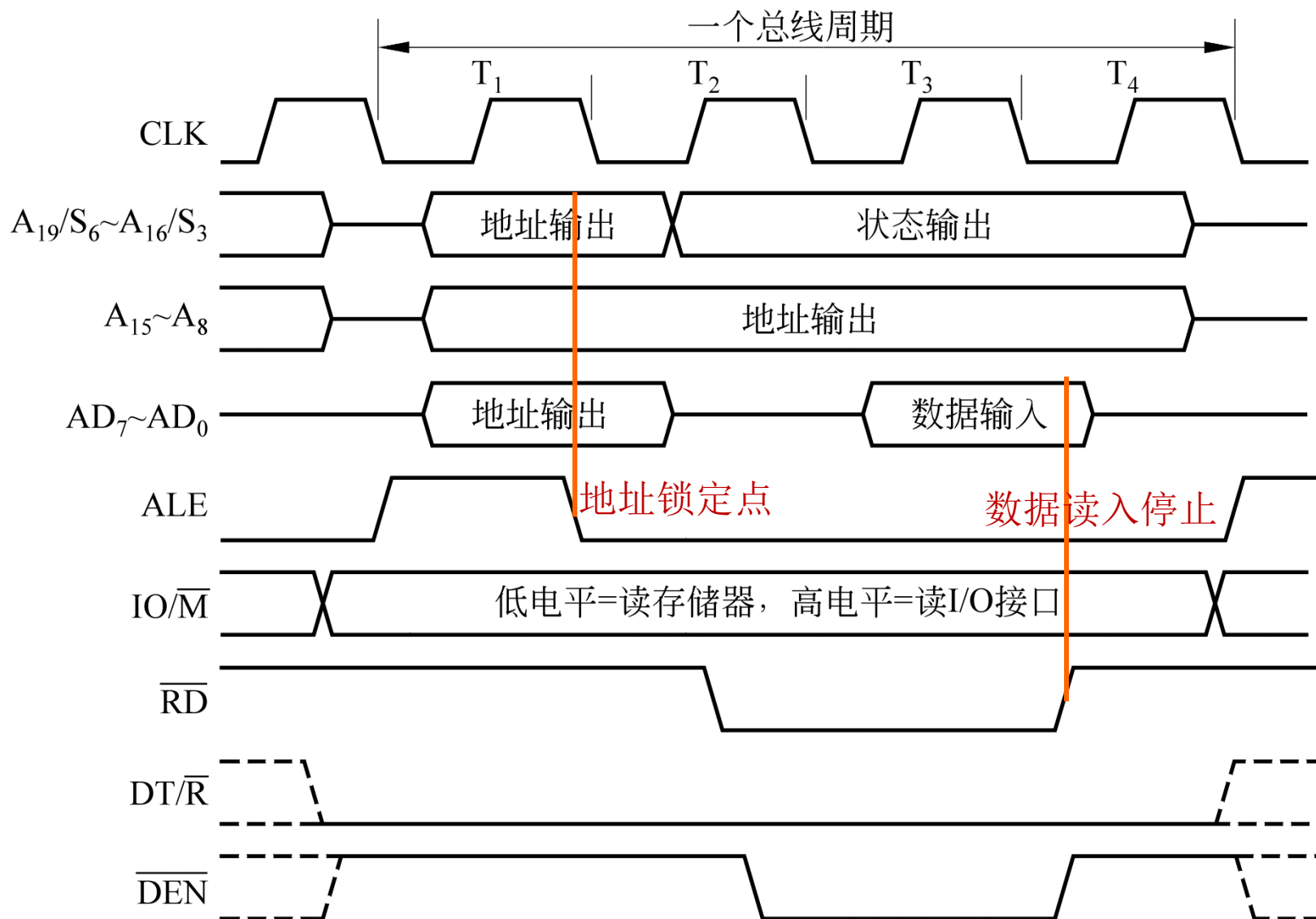
- CPU完成一次访问内存(或I/O接口)操作所需要的时间。
- 一个总线周期至少包括4个时钟周期。

典型的总线周期

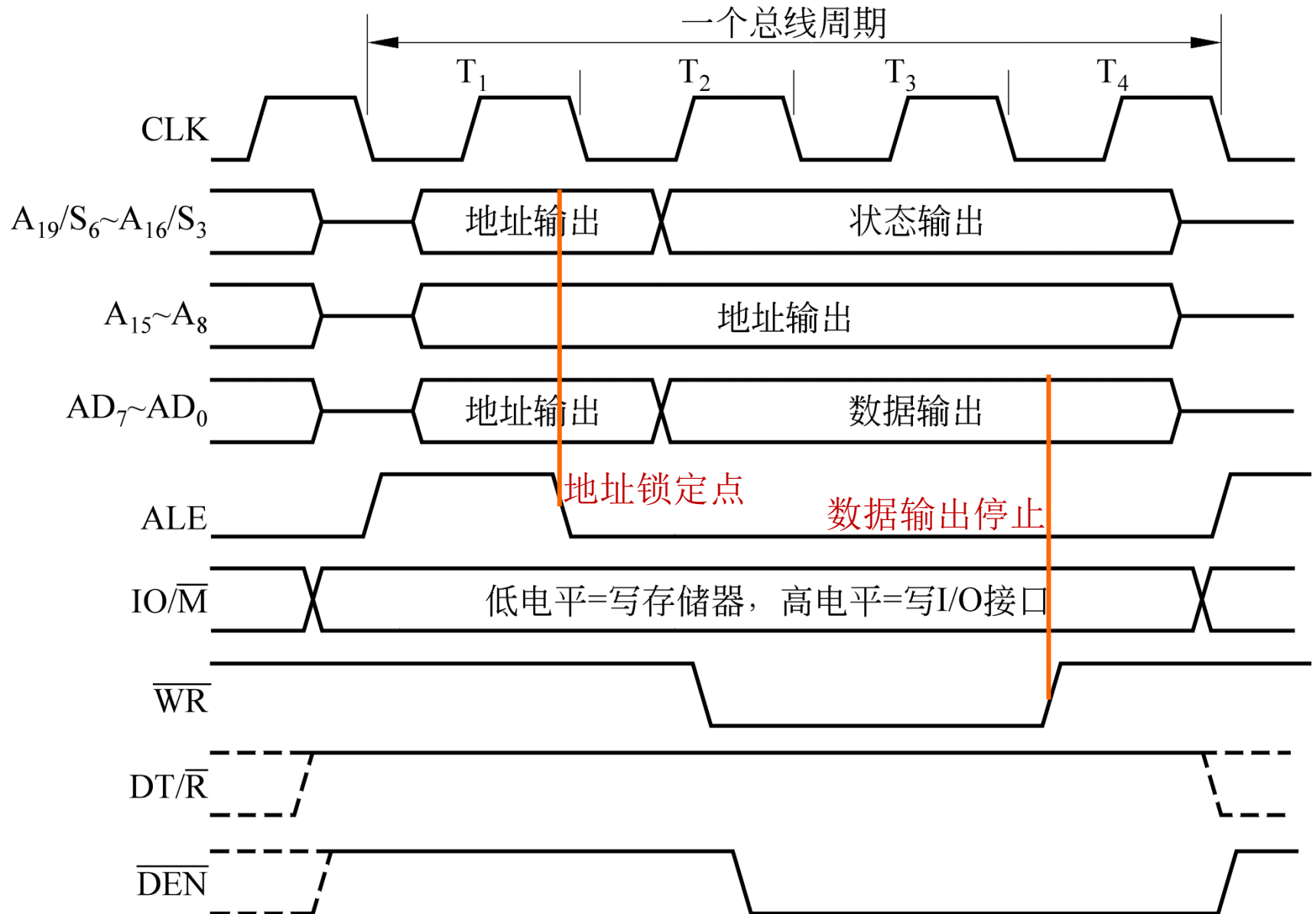
进行一次总线操作(访存或访**I/O**)至少需**4**个时钟周期**T**



8088最小模式下的读周期



8088最小模式下的写周期



七. IA-32微处理器及工作方式

本节内容包括：

- ◆ IA-32微处理器结构
- ◆ IA-32微处理器工作方式
- ◆ 保护模式下的存储器访问
- ◆ IA-32e微处理器结构简介

（一） IA-32微处理器简介

Intel公司将80286之后的80X86 32位微处理器称为IA(Intel Architecture)-32结构。

1. 80386微处理器

80386是与8086兼容的高性能的32位微处理器

CPU被划分为6个单元：总线接口单元BIU、指令预取单元IPU、指令译码单元IDU、执行单元EU、分段单元SU、分页单元PU。

主要特点：

- 具有4GB (2^{32}) 的物理地址空间和64TB (2^{46}) 的虚拟地址空间的存储器寻址能力
- 不仅有分段存储管理方式，还增加了分页存储管理方式。
- 多任务运行机制
- 优先级保护机制
- 具有实地址模式、保护模式和虚拟8086模式三种工作方式。

1989年Intel公司推出了80486微处理器芯片，它集成了一个80386和一个80387。

2. Pentium 微处理器

主要特性

➤ 与80X86系列微处理器兼容

➤ RISC型超标量结构

处理器包含有多个指令单元和多条指令流水线，其运行速度成倍提高。

➤ 高性能浮点运算器

采用了超级流水线技术，使其浮点运算速度比486快3~5倍。

➤ 双重分离式高速缓存

将指令缓存与数据缓存分离，工作速度大大提高。

➤ 64位数据总线

采用突发传送方式，大幅度提高了数据传输速度。

➤ 分支指令预测

采用分支指令预测技术，大大提高流水线执行效率。

➤ 常用指令固化与微代码改进

把常用的指令改用硬件实现，即引入了RISC技术，使程序运行速度进一步提高。

➤ 系统管理方式

提供了系统管理方式(SMM)，它向操作系统提供电源管理和系统安全等功能的机制。

(二) IA-32微处理器的主要寄存器

1. 通用寄存器

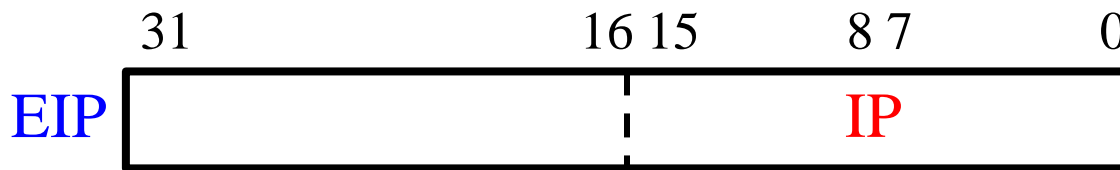
8个32位的通用寄存器

	31	16	15	8	7	0
EAX				AH	AX	AL
EBX				BH	BX	BL
ECX				CH	CX	CL
EDX				DH	DX	DL
ESI				SI		
EDI				DI		
EBP				BP		
ESP				SP		

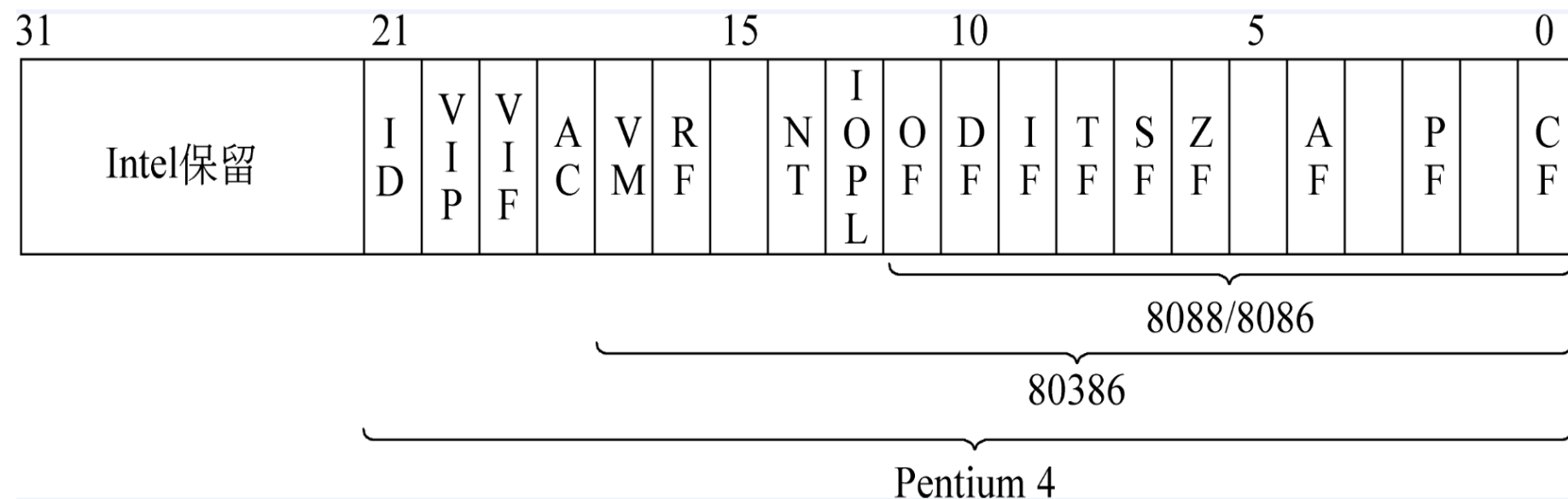
- EAX, EBX, ECX和EDX可以作为32位、16位或8位寄存器使用。
- 使用16位和8位寄存器时，与8086中的命名（AX, BX, CX, DX, AH, AL, BH, BL等）一致。
- ESP、EBP、EDI和ESI被扩展为32位，SP、BP、DI和SI仍然与8086一致。

2. 指令指针和标志寄存器

指令指针EIP为32位，其低16位与8086同名为IP。



标志寄存器EFLAGS为32位，其中第0~11位共有9个标志位，与8086一致。



3. 段寄存器

- ◆ 共有6个16位的段寄存器，CS、SS、DS和ES与8086相同，增加了两个数据段寄存器FS和GS。
- ◆ IA-32中段基址和偏移量都是32位，段寄存器只有16位不直接作段基址，其内容称为段选择器，需间接访问才能得到段基地址。

4. 系统地址寄存器

两个48位的寄存器：{ 全局描述符表寄存器GDTR
中断描述符表寄存器IDT

两个16位的寄存器：{ 局部描述符表寄存器LDTR
任务状态段寄存器TR

这四个寄存器都是用来实现对存储器的访问，GDTR、IDT和LDTR的作用将在后面介绍。

5. 控制寄存器

- ◆ IA-32内部有4个32位的控制寄存器CR0、CR1、CR2和CR3，其中CR1保留未使用。
- ◆ 它们用来设置和保存机器的各种全局性状态，比如是否有浮点部件，是否处于保护模式，是否采用页式存储管理等等。

(三) IA-32处理器的工作方式

IA-32有两种主要的工作方式:

- 实地址方式—实模式
- 保护虚地址工作方式—保护模式

◆ 在保护模式下又增加了一种虚拟8086方式

另外还有一种称为系统管理模式(**SMM**), 它只是给操作系统提供管理电源和系统安全的机制, 一般不归类到工作方式。

1. 实模式

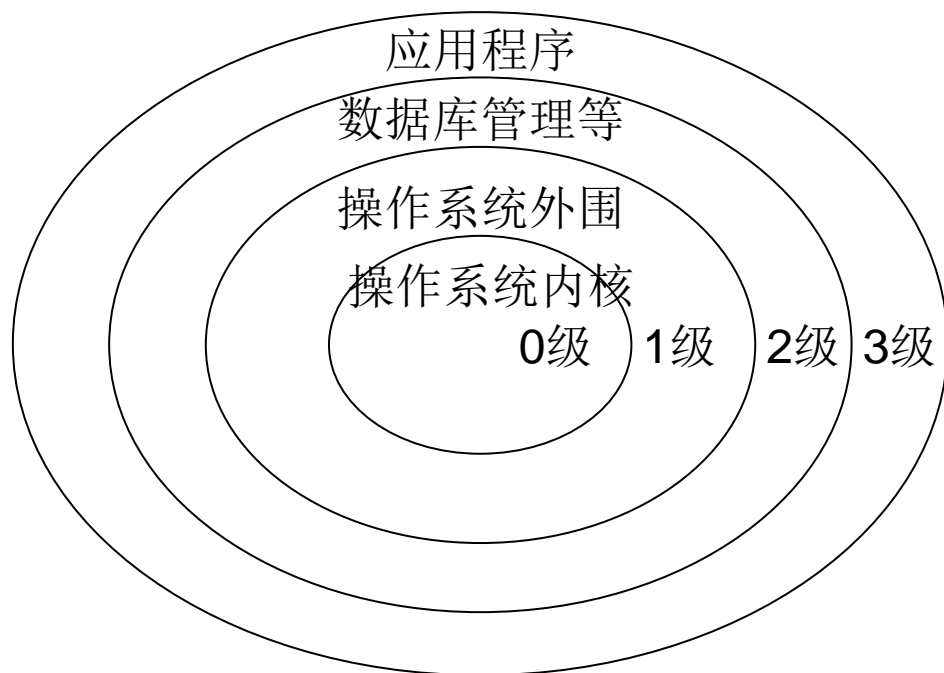
- 实模式是为了与8086兼容而设置的工作方式。
- IA-32的30多条地址线中只有低20条地址线起作用, 可寻址1MB的物理地址空间。
- IA-32 CPU相当于一个快速的8086, 虽然具有32位的数据线和数据寄存器, 但无法发挥其全部功能。
- 实模式下微处理器不能实现多任务的处理。

2. 保护模式

保护虚地址方式是IA-32微处理器的主要工作方式

特点：

- 32位地址可寻址4GB的线性存储器空间；
- 支持虚拟存储器功能。每个任务运行可以有16K个段，每个段最大为4GB，一个任务最大可使用64TB虚拟地址空间；
- 程序运行分为4个特权等级，操作系统核心运行在最高特权级0，用户程序运行在最低特权级3。



3. 虚拟86模式

为了在IA-32系统上运行MS-DOS支持下的应用程序，微处理器可以工作在实地址方式，但其功能十分有限。

- 寻址空间只有1MB；
- 不能实现多任务、多用户处理；
- 没有特权机制；
- 不能发挥IA-32微处理器的分页式存储管理机制。

在虚拟8086方式下，IA-32微处理器总体上是工作在保护模式，支持多用户多任务操作系统。其中，有的任务可以工作在虚拟8086方式，运行DOS应用程序。

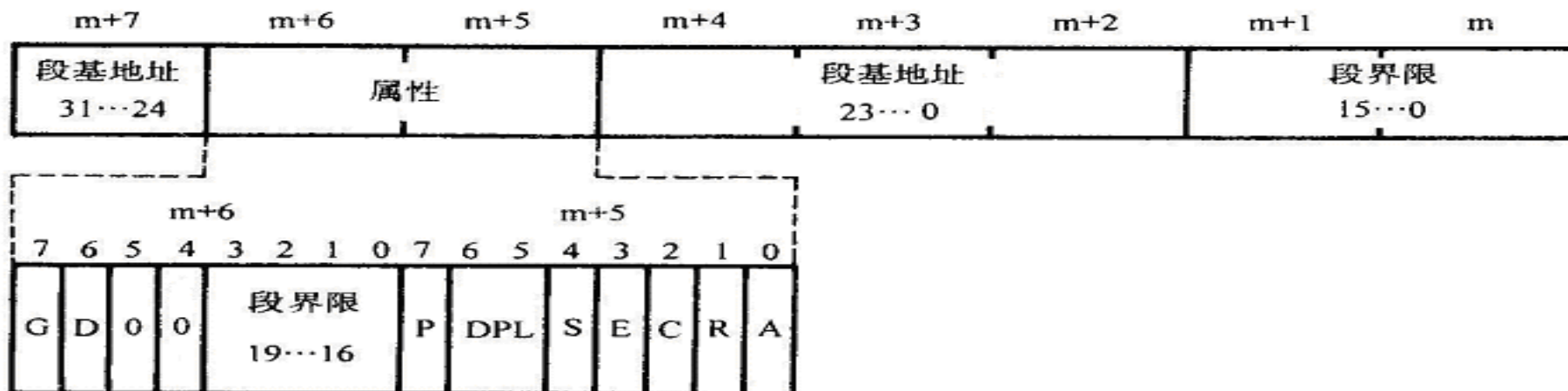
在本课程后面的章节中我们把**8086/8088**的工作模式和**IA-32**处理器的实模式简称为**16位模式**；将**IA-32**处理器的保护模式简称为**32位模式**。

(四) 保护模式下的存储器访问

- ◆ 在**8086**或实模式下**CPU**只能访问**1MB**的存储空间，用户可随意访问任何地址单元。
- ◆ 保护模式下可访问的地址空间达**4GB**，但不是随意使用，而是要受到一定的限制，这种保护就是由**段描述符**来完成。

1. 段描述符

- ◆ 一个段描述符指示了这个段在内存中的位置、大小以及使用限制。段描述符有**64**位，包括段基址**32**位、段界限**20**位、特权级等属性**12**位。



- ◆ 段基址：**32** 位，它指示该段在 **4GB** 线性地址空间中的起始地址。
- ◆ 属性：**12**位，包括特权级、段类型、段存在标志、粒度标志等。
 - **特权级DPL**：每个段都有一个特权级，取值范围 **0 ~ 3**，**0** 级为最高级，一般用于操作系统核心代码。如果特权级数值高的程序试图访问特权级数值低的段，则发生处理器故障。
 - **段类型**：指定段的访问类型（只读、可读可写、只执行等）以及段生长的方向（向上或向下）。
 - **段存在标志**：指示该段当前是否在物理内存中
 - **粒度标志G**：确定对段界限字段的解释。**G=0**:段界限以字节为单位。**G=1**:则段界限的解释单位为 **4096** 字节。。
- ◆ 段界限：**20** 位，指定段的大小。按照粒度标志，有两种解释：
 - **G=0**: 允许段大小的范围为 **1B~1MB**。
 - **G=1**:允许段大小的范围为 **4KB~4GB**。

2. 描述符表

- ◆ 把多个段的段描述符顺序地存放在一起就构成描述符表。分为全局描述符表**GDT**和局部描述符表**LDT**。
- ◆ 系统中所有任务都可能会用到的段描述符所构成的描述符表称为全局描述符表**GDT**。系统中只有一个**GDT**!
- ◆ 用全局描述符表寄存器**GDTR**来指示**GDT**在内存中的存放位置。
- ◆ 每个任务的代码段、数据段和堆栈段的段描述符，以及该任务所使用的门描述符组成该任务的局部描述符表**LDT**。
- ◆ 每个任务的**LDT**在内存中的存放空间形成一个段，这个段的描述符叫**LDT描述符**。
- ◆ **LDT描述符**作为系统描述符也存放在**GDT**中。
- ◆ 局部描述符表寄存器**LDTR**用来指示**LDT描述符**在**GDT**中的位置。

- ◆ **LDTR**只有**16**位，它无法直接指向**LDT**的地址，但可以间接地访问到局部描述符表**LDT**。
- ◆ 在多任务的**Windows**系统中，**LDTR**只有一个，它用来指示当前任务，当前任务切换时就会改变**LDTR**的内容。

3. 段选择器

- ◆ **IA-32**有**CS**、**DS**、**ES**、**SS**、**FS**和**GS** 6个段寄存器
- ◆ 段寄存器的内容称为**段选择器（或段选择子）**，用来从**GDT**或**LDT**中选择一个段描述符，即用来选择一个段。

	15		3	2	1	0
段选择器	索引 值			TI	CPL/ RPL	

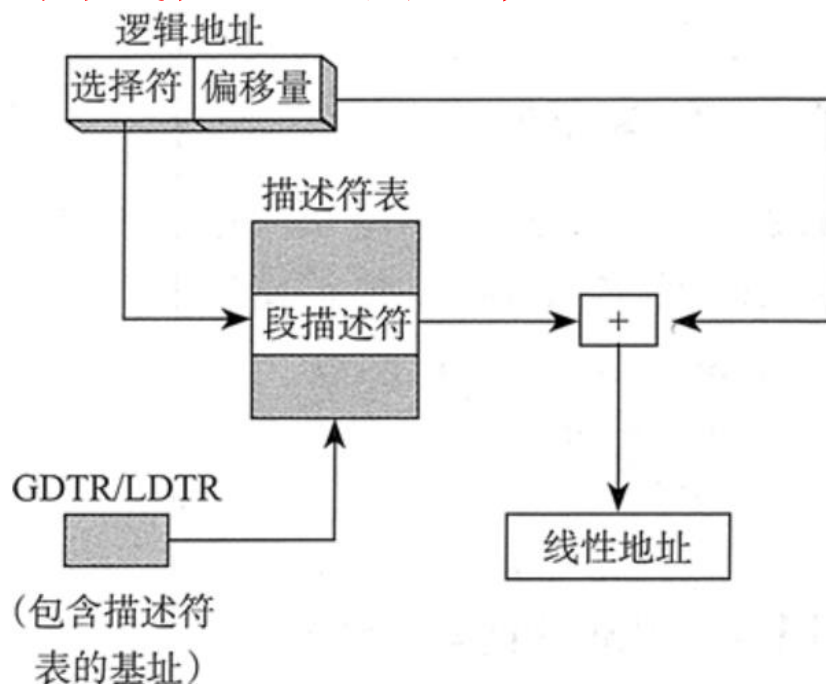
- ✓ **索引值**：**13**位，指示在描述符表中的相对位置。
- ✓ **TI**：**1**位，为**0**则描述符在**GDT**中，为**1**则在**LDT**中。
- ✓ **CPL/RPL**：**2**位，表示当前的代码或数据的优先级。
CS中就是**CPL**，其他段寄存器中为**RPL**。

保护模式下的逻辑地址表示仍然是： 段寄存器：段内地址

段寄存器--为CS、DS、ES、SS、GS、FS之一

段内地址--为32位，段的长度可达 $2^{32}=4\text{GB}$

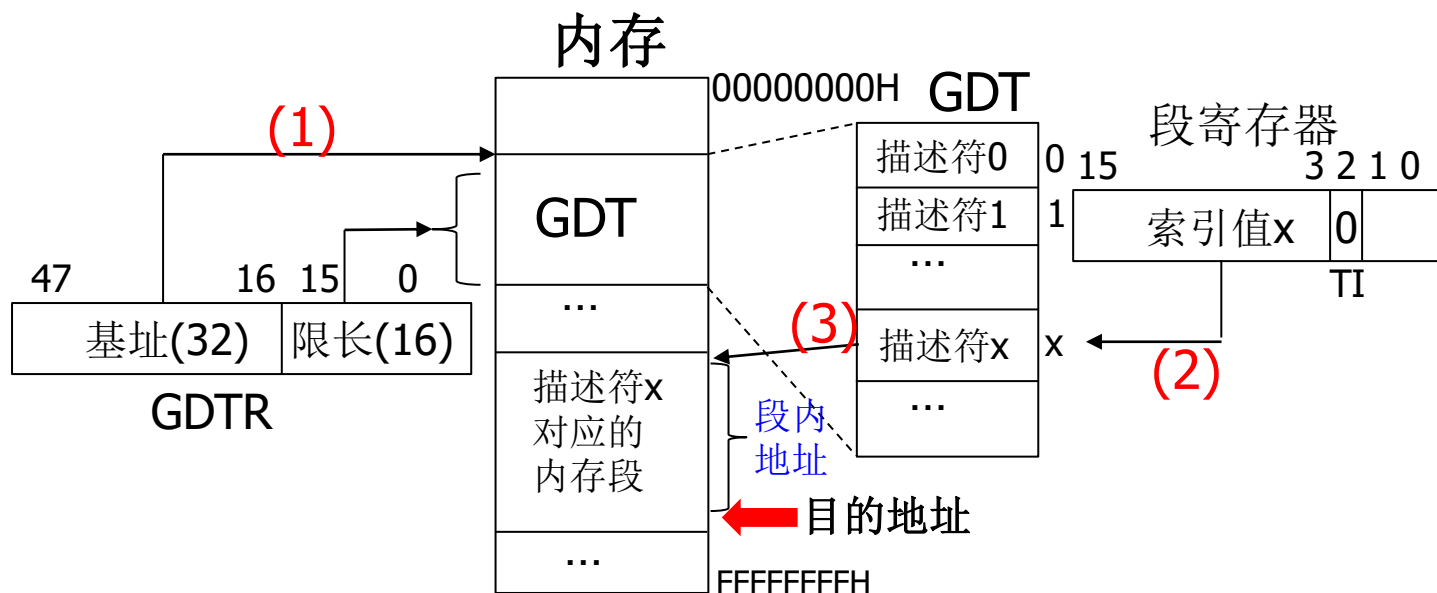
4. 逻辑地址转换为线性地址的过程



- ◆ 在指令执行时要将逻辑地址转换成一个32位的线性地址才能访问到物理存储器。
- ◆ 根据段寄存器中的TI位是0还是1，有两种不同的转换过程。

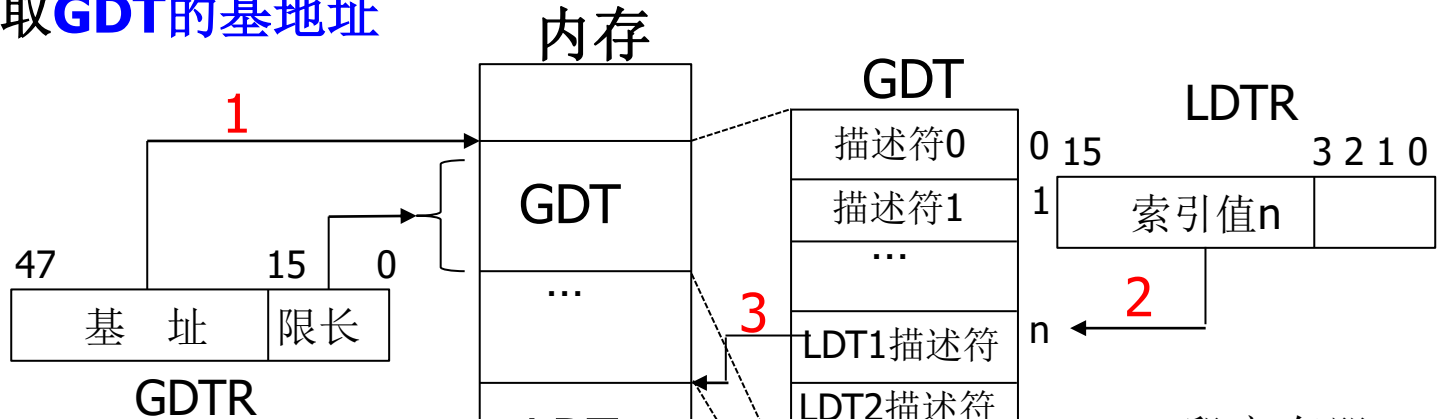
TI=0:段描述符在全局描述符表GDT中

- (1)从GDTR中获取GDT的基地址
- (2)以段选择器的高13位做索引值在GDT中找到对应的段描述符。
- (3)从段描述符中获得段基址，即找到段的起始地址。
- (4)从段基址偏移段内地址即是最终的目的地址-线性地址



TI=1:段描述符在局部描述符表LDT中

(1)从GDTR中获取GDT的基地址



(2)以LDTR的高13位做索引值在GDT中找到对应的LDT段描述符

段内地址
目的地址 →

(3)从LDT段描述符中获得该LDT的起始地址

(4)以段寄存器的高13位做索引值在LDT中找到对应的段描述符

(5)从段描述符中获得段基址，即找到段起始地址。

(6)从段基址偏移段内地址即是最终的线性地址

线性地址与物理地址的对应关系

- ◆ 当处理器屏蔽了分页机制时，线性地址与物理地址一一对应，即线性地址就是物理地址。
- ◆ 当处理器启用分页机制时，存储地址空间采用分页管理，每个页面为**4KB**。
- ◆ 物理地址空间的各个不相邻的页可以映射成一个连续的线性地址空间，这方便了对存储器的碎片管理。
- ◆ 分页机制下物理地址空间可以大于线性地址空间，例如**Pentium4**处理器线性地址空间为**4GB**，但它有**36**条地址线，可以有**64GB(2^{36} 字节)**的物理地址空间。
- ◆ 分页机制还可以实现虚拟内存机制，它让程序可使用的内存容量大于实际的物理内存容量。
- ◆ 分页机制对用户透明，用户看不到页面是如何映射，只需使用线性地址即可。

平坦存储模式--flat

- ◆ 在**8086CPU**中，通过段寄存器对数据和代码进行分段管理是一种安全机制。
- ◆ 在**80386**以后的**CPU**中采用了有效的保护模式机制，所以分段管理的安全作用就不需要，段寄存器的分工就可以没有了。
- ◆ 在**Windows 32**位模式下将所有的段寄存器都设置为指向相同的线性地址**0**，代码段、数据段、堆栈段都重合于同一个**4GB**的线性地址空间。这种工作模式称为**平坦存储模式（或平面存储模式）**。
- ◆ 平坦存储模式下完全可以不管段寄存器的使用，访问存储器只使用**32**位的偏移地址，程序中不会出现**CS、DS、SS...**等段寄存器。
- ◆ 虽然数据、堆栈和代码都共用一个**4GB**的线性地址空间，但在存储分配时仍然是相对集中，不是交叉随机分配。

（五）IA-32e架构及工作模式

- ◆ IA-32e是向后兼容IA-32 的64 位x86 处理器架构，该架构最早由AMD公司提出，将其称为AMD-64。
- ◆ Intel 推出的面向个人电脑的IA-32e架构处理器有Core i5、Core i7 和Core i9 等64 位处理器。用于用于服务器和工作站的处理器为Xeon（至强）。
- ◆ AMD 推出的处理器有Opteron 和Athlon64 等。

1. 寄存器结构

- ◆ 16个64位的通用寄存器RAX、RBX、RCX、RDX、RSI、RDI、RBP、RSP 以及新增加的R8-R15。
- ◆ 64 位状态标志寄存器RFLAGS，但只定义了其低32 位，与IA-32 的32位状态标志寄存器EFLAGS 相同。
- ◆ 64 位指令指针寄存器RIP。
- ◆ 8 个64 位MMX 寄存器，8 个80 位浮点寄存器，与IA-32 相同。
- ◆ 16 个128 位XMM 寄存器，比IA-32 增加了8 个。

2. 工作模式

IA-32e 架构处理器有兼容模式和**64** 位模式两种工作模式。

(1) 兼容模式

- ◆ 线性地址为**32** 位，最大线性地址空间为**4GB**。
- ◆ 可直接运行原有的**16** 位和**32** 位应用程序。但Microsoft 的**64** 位Windows操作系统不能运行**16** 位Windows 或DOS 的应用程序。

(2) 64 位模式

- ◆ 线性地址为**64** 位，即编程地址空间达到了 **2^{64} B**。
- ◆ 处理器执行的指令可以混合使用**32** 位和**64** 位操作数。
- ◆ 在指令前加前缀**REX**就可以直接使用CPU 中的**64** 位寄存器提供**64** 位操作数。

本章小结

- 微处理器的一般构成
- 8088CPU的主要引线及其功能
- 8088CPU的内部结构
- 内部寄存器功能
 - 寄存器中数据的含义
 - 8位寄存器中存放的均为运算的数据
- 存储器组织
 - 逻辑地址，段基地址，偏移地址，物理地址
- 堆栈
 - 栈顶地址，栈底地址，堆栈段基地址
- 系统总线
 - 最小模式和最大模式的总线，总线时序
- 80386、pentium微处理器结构特点
- IA-32的三种工作模式
- 保护模式下的存储器访问方法

作业

- 习题二 2.7、2.8、2.10、2.22