第二章
微机系统结构

从Intel8086到Pentium 4

- 本章主要教学内容
  - 80x86微处理器的基本性能指标、组成及其寄存器结构
  - 8086/8086微处理器的引脚特性
  - 8086微处理器的存储器、I/O组织、时钟和总线周期
  - 80286、80386、80486、Pentium、Pentium 4简介
- 教学目的
  - 掌握8086微处理器的基本原理，了解80286～Pentium 4CPU的结构
- 教学重点：
  - 80x86微处理器的组成及其寄存器结构； 8086的存储器组织；8086系统配置；8086的总线周期
- 教学难点
  - 8086微处理器的系统配置和总线周期

2-0、80x86微处理器的工作模式

- 80386以上系统中有四种工作模式：

（1）实地址模式

（2）保护模式

（3）虚拟8086模式

（4）系统管理模式

- 8086/88系统只有一种工作模式
- 80286有（1）～（3）三种模式

（参见教材P446页）

（1）实地址模式

- 80286以上的微处理器所采用的8086的工作模式；
- 在实模式下，采用类似于8086的体系结构，其寻址机制、中断处理机制均和8086相同；
- 寻址空间为1MB，并采用分段方式，每段大小为64KB；
- 实模式是80x86处理器在加电或复位后立即出现的工作方式，系统初始化或引导程序必须先运行实模式；
- 实模式是为建立保护模式做准备的工作模式。

实模式的内存地址保留区域

- 在实模式下，存储器中保留两个专用区域
- 初始化程序区：
  - **FFFF0H～FFFFFH**，存放进入**ROM**引导程序的一条跳转指令；
- 中断向量表区：
  - **0000H～003FFH**，在这**1K**字节的存储空间中存放**256**个中断服务程序的入口地址，每个入口地址占**4**个字节（与**8086**的情形相同）。

（2）保护模式

- 支持多任务的工作模式。
- 提供了一系列的保护机制，如任务地址空间的隔离，设置特权级（0～3共4个特权级），设置特权指令，进行访问权限（如只读、只执行）及段限检查等。
- 80386以上的微处理器在保护模式下可以访问4G字节的物理存储空间，段的长度在启动分页功能时是4G字节，不启动分页功能时是1M字节，分页功能是可选的。
- 可以引入虚拟存储器的概念，以扩充编程者所使用的地址空间。

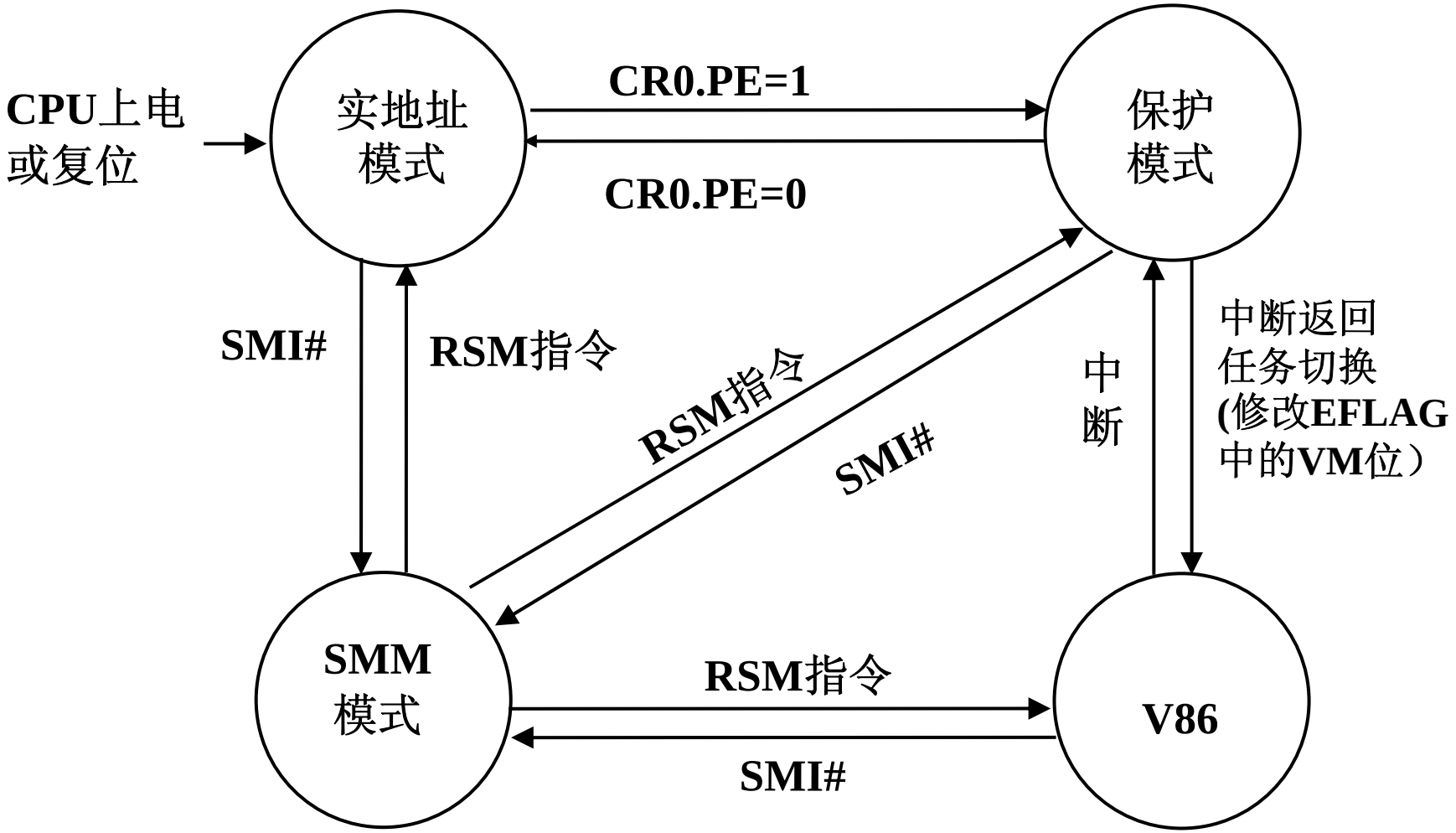
（3）虚拟8086模式

- 虚拟8086模式又称“V86模式”。
- 它是既有保护功能又能执行8086代码的工作模式，是一种动态工作模式。
- V86模式可以在多任务环境下运行多个用8086编写的程序（针对8086编写的程序虽然能够在实模式下运行，但是实模式不支持多任务）。
- 处理器能够迅速反复地在V86模式和保护模式之间切换，从保护模式进入V86模式执行8086程序，然后离开V86模式，进入保护模式继续执行原来的保护模式程序。

（4）系统管理模式（SMM）

- Intel 80386 SL开始引入的模式，标准的IA-32结构特点。
- 为操作系统和正在运行的应用程序提供透明的电源管理和系统安全平台功能。
- 当外部系统管理中断（SMI#）引脚被触发，或者从APIC（高级中断控制器）接收到一个系统管理中断（SMI#），处理器进入进入本模式。
- 进入SMM模式后，处理器首先保存当前运行的程序或状态，再转到一个独立的地址空间运行系统管理模式指定的代码。
- 从SMM模式返回时，处理器回到响应SMI#中断前的工作状态。

4种工作模式的转换关系

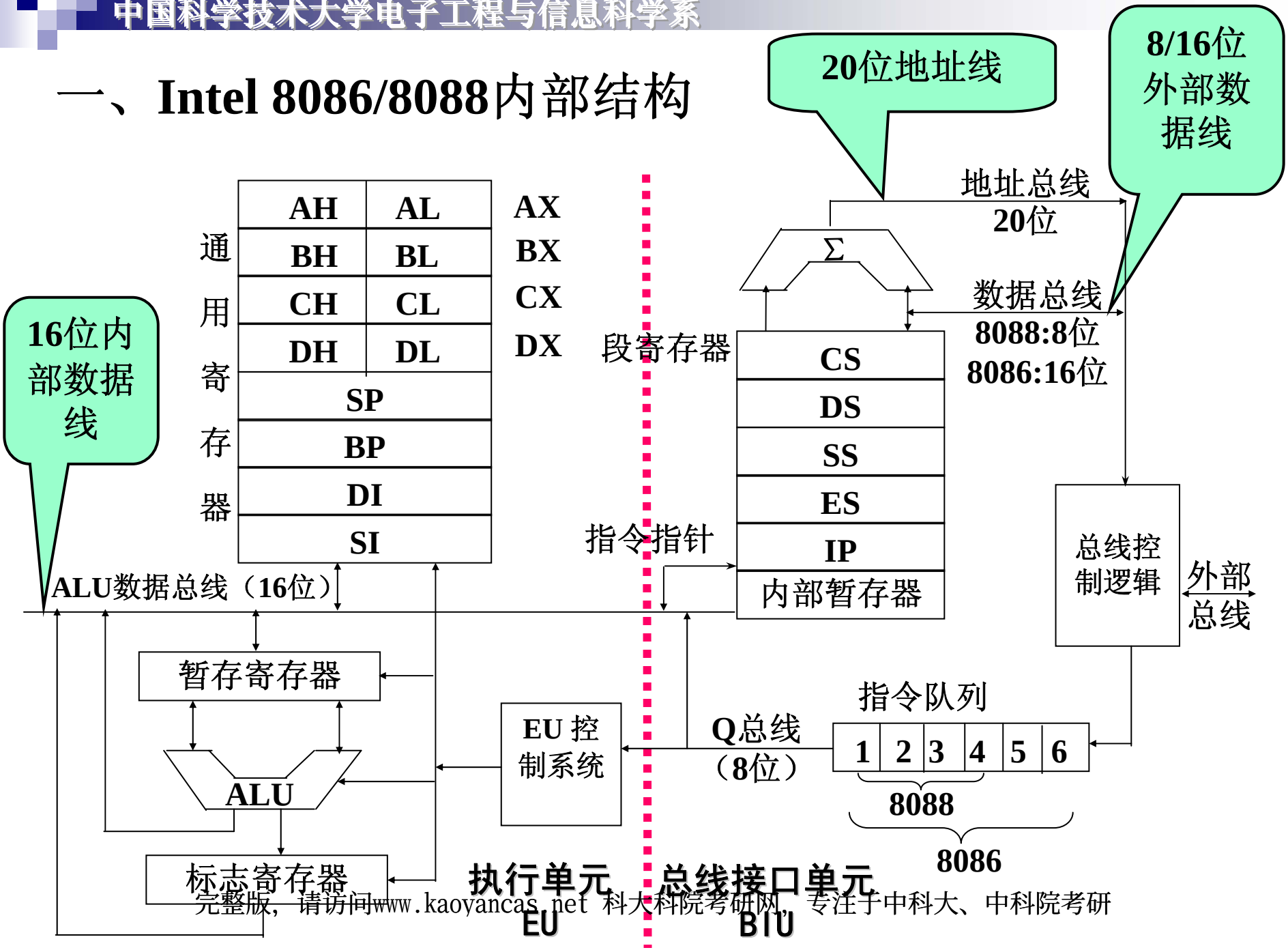


SMI#－系统管理中断信号；RSM指令是系统管理中断服务程序返回指令；修改控制寄存器CR0的PE=1（0）进入（退出）保护模式。

2-1、8086/8088 CPU

- 基本性能指标
  - **16位微处理器（8088的DB只有8位，称为准16位）；**
  - **采用HMOS工艺制造，芯片上集成了2.9万只晶体管；**
  - **单一+5V电源，40pin DIP（双列直插）封装；**
  - **时钟频率为5MHz~10MHz，基本指令执行时间为0.3us~0.6us，最长执行时间超过10us。**
  - **16根数据线和20根地址线，可寻址地址空间1MB。**
  - **8086有一个初级流水线结构，内部操作与对外操作具有并行性。**
  - **8086/8088可以和浮点运算器（协处理器）、I/O处理器或其它处理器组成多处理器系统，以提高系统的数据吞吐能力和数据处理能力。**

一、Intel 8086/8088内部结构



总线接口部件BIU －Bus Interface Unit

- 主要功能：
  - 根据执行部件EU的请求，负责完成CPU与存储器或I/O设备之间的数据传送。
- 内部构成
  - 四个16位段地址Reg：代码段CS、数据段DS、堆栈段SS和附加段ES；
  - 20位地址生成电路（地址加法器）
  - 一个16位指令指针IP，存放下一条要执行的指令地址；
  - 一个6/4字节指令队列缓冲器；
  - 总线控制电路。

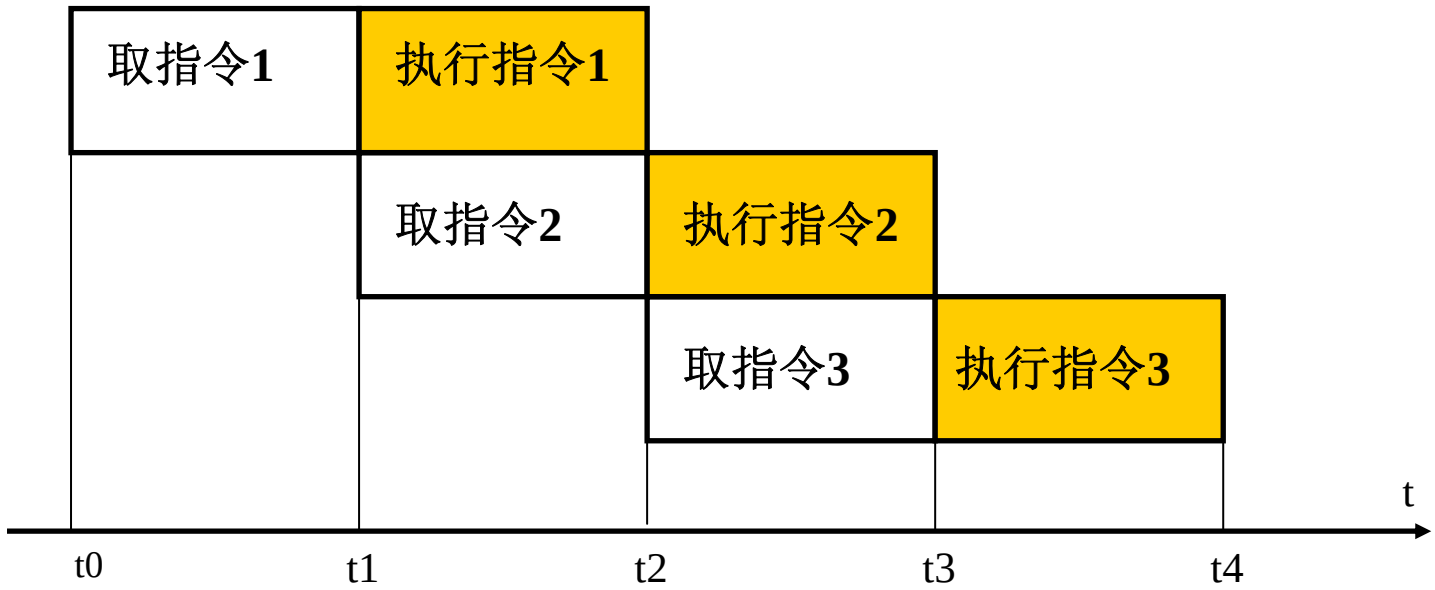
执行部件EU－Execute Unit

- 功能：
  - 从BIU的指令队列中取出指令代码，经指令译码器译码后执行指令。
  - 指令执行结果或指令执行所需的数据，都由EU向BIU发出命令，对存储器或I/O接口进行读/写操作。
- 执行部件中包含：
  - 一个16位的算术逻辑单元（ALU）；
  - 八个16位的通用Reg；
  - 一个16位的状态标志Reg和一个数据暂存Reg；
  - 执行部件的控制电路。

EU与BIU的并发操作—初级流水线

- EU与BIU可独立工作，BIU在保证EU与片外传送操作数前提下，可进行指令预取，与EU可重叠操作。
- 8086指令队列出现2个空字节，且EU未占总线，BIU自动取指令填充队列。

8086流水线操作



流水线的优点：在t0～t4时间间隔中，理想情况下，8086可执行3条指令

（注：此处的t0～t4不是机器时钟周期）

二、80x86微处理器的寄存器结构

- 80286～Pentium微处理器为保持与8086的兼容性，内部寄存器是在8086基础上的扩充。8086的寄存器可以看作是它们的寄存器组的一个子集。

- 80286以上微处理器的寄存器分为3大类：

  1)用户级寄存器，也称为程序可见寄存器。这类寄存器在进行汇编语言程序设计时必须掌握。也是本章介绍的重点。

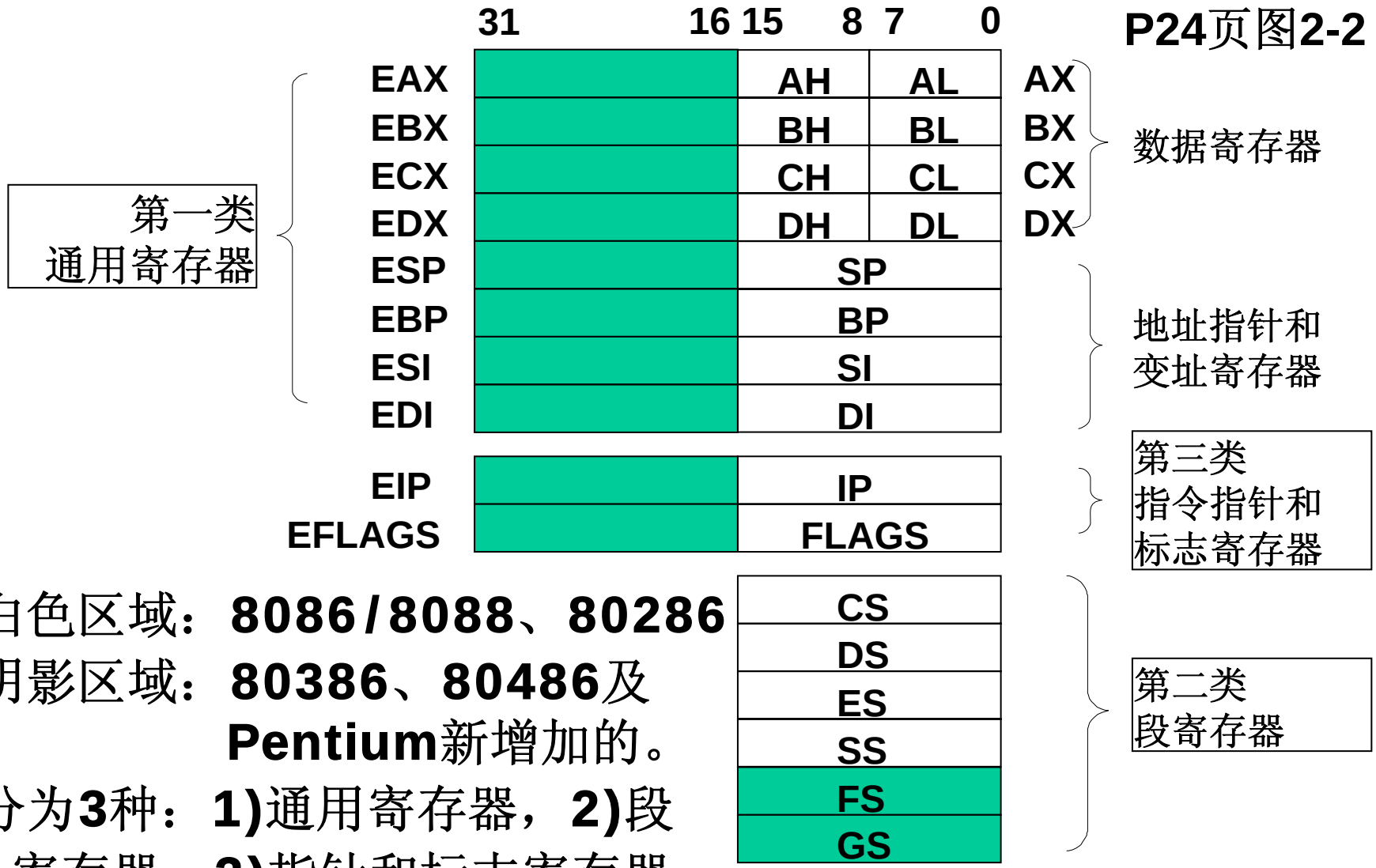
  2)系统级寄存器：80286以后新增加的，包括控制寄存器和支持存储器管理的段表寄存器。控制寄存器主要供操作系统使用，操作系统设计者要熟悉这些寄存器；段表寄存器在应用程序设计时不能直接访问，但能被系统软件访问或被间接引用，因此又称为程序不可见寄存器。

3)程序调试寄存器：80386以后陆续增加的寄存器。如80386增加10个32位DR0～DR7、TR6和TR7，PII增加的mm0～mm7寄存器，PIII又增加了xmm0～xmm7单精度浮点寄存器，在SIMD体系结构中使用汇编语言编程时将用到这些寄存器。

- 注：
  - SIMD体系结构是由一个控制器、多个处理器、多个存贮模块和一个互连网络组成。所有“活动的”处理器在同一时刻执行同一条指令，每个处理器执行这条指令时所用的数据是从它本身的存储模块中读取的。对操作种类多的算法，当要求存取全局数据或对于不同的数据要求做不同的处理时，SIMD无法独立胜任。
  - 与之相对应的是MIMD体系，就是通常所指的多处理机，典型MIMD系统由多台处理机、多个存储模块和一个互连网络组成，每台处理机执行自己的指令，操作数也是各取各的。MIMD结构中每个处理器都可单独编程，因而这种结构的可编程能力是最强的。但是硬件利用率不高。

80x86处理器*用户级*寄存器模型

参见P450页图13-4，对比P24页图2-2



白色区域：**8086/8088、80286**

阴影区域：**80386、80486及Pentium新增加的。**

分为**3**种：**1)**通用寄存器，**2)**段寄存器，**3)**指针和标志寄存器。

（一）通用寄存器

- 分为通用数据寄存器与指针和变址寄存器两组。

（1）通用数据寄存器：

- 共有4个通用数据寄存器，EAX、EBX、ECX和EDX（8086/8088和80286只有16位）。
- 一般用来存放8位、16位或者32位操作数，大多数算术运算和逻辑运算指令都可以使用这些寄存器。
- 每一个可根据需要作为32位寄存器(ERX)、16位寄存器(RX)或8位寄存器(RH或RL)引用，它们均可独立寻址、独立使用。（其中R代表A或B或C或D）
- 8086/8088和80286中的16位寄存器（AX、BX、CX和DX）当作是32位的子集。

EAX（Accumulator—累加器）

- EAX可以作为32位寄存器(EAX)、16位寄存器(AX)或8位寄存器(AH或AL)引用。
- 当累加器用于乘法、除法及一些调整指令时，它具有专门的用途（参见教材P25表2-1）。
- 在I/O指令中存放输入/输出数据。
- 在80386及更高型号的微处理器中，EAX寄存器也可以用来存放访问存储单元的偏移地址。
- 8086/8088和80286中的累加器AX只有16位。

EBX（Base—基址）

- **EBX是个通用寄存器，它可以作为32位寄存器(EBX)、16位寄存器(BX)或8位寄存器(BH或BL)引用。**
- 在查表转换和间接寻址时，存放访问存储单元的偏移地址（基址）。

ECX（Count—计数）

- ECX是个通用寄存器，它可以作为32位寄存器(ECX)、16位寄存器(CX)或8位寄存器(CH或CL)引用。
- ECX可用来作为多种指令的计数值。用于计数的指令是重复的串操作指令、移位指令、循环移位指令和LOOP/LOOPD指令。
- 移位和循环移位指令用CL计数，重复串操作指令用CX计数，LOOP/LOOPD指令用CX或ECX计数。
- 在80386及更高型号的微处理器中，ECX也可用来存放访问存储单元的偏移地址。

EDX（Data，数据）

- EDX是个通用寄存器，它可以作为32位寄存器(EDX)、16位寄存器(DX)或8位寄存器(DH或DL)引用。
- 在乘法指令中用于保存乘法运算产生的乘积的高位部分
- 在除法运算前存放被除数高位部分或余数。
- 在I/O操作时，用于对IO端口的间接寻址。
- 对于80386及更高型号的微处理器，这个寄存器也可用来寻址存储器数据

（2）地址指针和变址寄存器

- 另外4个通用寄存器，分别是：

1)ESP(Stack Pointer，堆栈指针)：ESP通常用于寻址一个称为“堆栈”的存储区，ESP存放的是访问堆栈所需的“堆栈指针”。这个寄存器作为16位寄存器引用时为SP；作为32位寄存器引用时则为ESP。

2)EBP(Base Pointer，基址指针)：EBP通常用来存放访问堆栈段的一个数据区的“基地址”（偏移量）。它作为16位寄存器引用时为BP；作为32位寄存器引用时，则是EBP。

3)ESI(Source Index，源变址)：ESI用于寻址串操作指令的源数据串。它的另一个功能是作为32位(ESI)或16位(SI)的数据寄存器使用。

4)EDI(Destination Index，目的变址)：EDI用于寻址串操作指令的目的数据串。如同ESI一样，EDI也可用为32位(EDI)或16位(DI)的数据寄存器使用。

- 以上8个通用寄存器中某些寄存器还有专门用途，并且有些寄存器被某些指令缺省指定使用（隐含寻址）。下表（教材P25页表2-1）列出了8086或者8088中各个通用寄存器的一些专门用途（注意到80x86系列CPU的兼容性，32位以上处理器可以类推）。

8086/8088通用寄存器的特定用法

Reg	特殊用途	Reg	特殊用途
AX, AL	I/O指令的数据寄存器 乘法指令存放被乘数或积(隐含) 除法指令存放被除数或商(隐含)	DX	字乘法/除法指令存放乘积高16位或被除数高位或余数(隐含)； 间接寻址I/O指令中的地址寄存器
AH	LAHF指令的目标寄存器(隐含)	SI	字符串运算指令的源变址寄存器(隐含) 间接寻址的变址寄存器
AL	十进制运算指令和XLAT指令中的累加器(隐含)	DI	字符串运算指令的目标变址寄存器(隐含) 间接寻址的变址寄存器
BX	间接寻址的基址寄存器 XLAT指令的基址寄存器(隐含)	BP	间接寻址的基址指针
CX	串操作和LOOP指令的计数器(隐含)	SP	堆栈操作的堆栈指针
CL	移位和循环指令的移位次数寄存器		

（二）段寄存器（Segment register）

- 8086～80286有4个16位的段寄存器，每个用来确定一个存储区(段)的起点，与其它寄存器联合生成存储器地址：
  （1）代码段寄存器CS （2）数据段寄存器DS
  （3）堆栈段寄存器SS （4）附加段寄存器ES
- 80386以上CPU又增加了两个16位的附加段寄存器：FS和GS。
- 对于同一个微处理器而言，在不同的模式（实模式和保护模式）下段寄存器的功能是不相同的。

- 实模式下，段寄存器用于确定一个64K字节存储器（段）起始地址，习惯上称为段址寄存器。
- 在保护方式下，段寄存器内保存着的16位数据，该数据用于索引（指示）一个被称为“描述符表”中的某一个表项，保护模式下的段寄存器又称为段选择符（selector）。
- 被索引的描述符表项被装入与段寄存器对应的“描述符寄存器”，每个表项定义某个段的起始地址、长度以及其它一些必要的属性信息（如可读、可写或可执行等）。保护模式下，80286最大段长度为16M，80386以上CPU最大段长度为4GB，程序可访问的虚拟地址空间可达64TB。

段寄存器的作用

1) 代码段寄存器**CS(Code Segment)**：代码段用来存储微处理器使用的代码（程序或过程）的一个存储区域，**CS**用于定义代码段的起始地址。

2) 数据段寄存器**DS(Data Segment)**：数据段是包含程序所使用的大部分数据的存储区。**DS**用于定义数据段的起始地址。

3) 附加段寄存器**ES(Extra Segment)**：附加段是为某些串操作指令存放目的操作数而附加的一个数据段。**ES**用以定义附加段的起始地址。

4) 堆栈段寄存器SS(Stack Segment)：

- 堆栈是存储器中的一个特殊存储区，用以暂时存放程序运行中的一些数据和地址信息。
- 堆栈段寄存器**SS**定义堆栈段的首地址。由堆栈段寄存器**SS**和堆栈指针寄存器**(ESP/SP)**共同确定堆栈段内的存取地址。
- 另外，**EBP/BP**寄存器也可以寻址堆栈段内的数据（操作数）。

5) 附加段寄存器FS和GS：

- 这两个段寄存器仅对**80386**及更高型号的微处理器有效，以便程序访问相应的两个附加的存储器段。

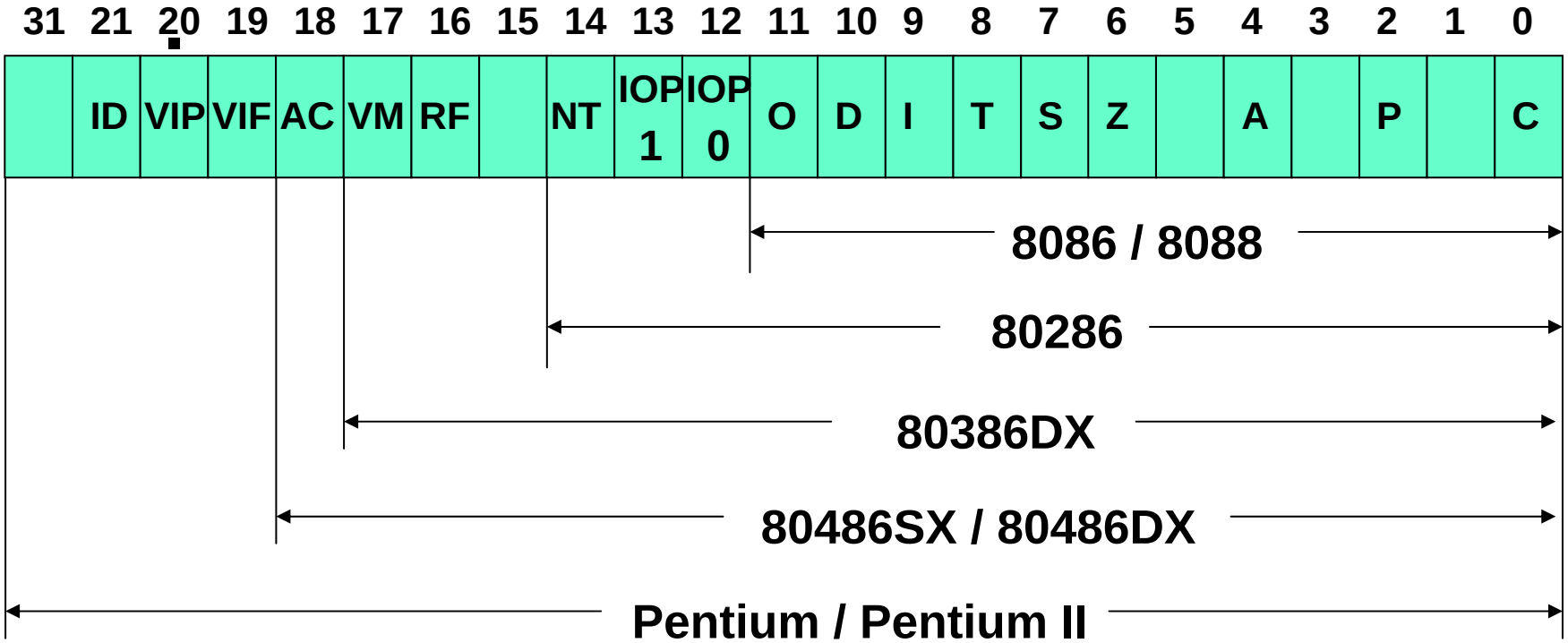
（三）指针和状态标志寄存器

- 32位指令指针EIP保存了下一条要执行的指令相对于现行代码段（CS）的段基地址的偏移量。偏移量加上当前段的地址，形成了下一条指令的地址。实模式下，EIP中的低16位与8086和80286 CPU中的IP相同，用于16位寻址，称为指令指针IP（Instruction Pointer）寄存器。

- 标志寄存器EFLAGS存放着微处理机当前状态的信息，EFLAGS从8086到Pentium保持兼容。

- 本章主要介绍8086 CPU中FLAGS寄存器的9个标志位。

80x86系列微处理器的标志寄存器



8086 CPU中的标志位－状态标志

- FLAGS寄存器中共有6个状态标志位
  - CF，进位标志。本次运算最高位有进位或借位发生，则CF＝1。STC（CLC）指令使CF＝1（＝0），CMC指令使之取反。
  - PF位，奇偶校验标志。本次运算结果的低8位有偶数（奇数）个1，则PE＝1（＝0）。该标志现已不用了。
  - AF，辅助进位标志。本次运算低4位向高4位有进位或借位发生时，AF＝1。常用于BCD码计算。
  - ZF，全零标志。本次运算结果为零时，ZF＝1。
  - SF，符号标志。本次运算最高位为1（补码表示负数）时，SF＝1。
  - 6个状态标志的速记方法：ACOPSZ

8086 CPU中的标志位－ 控制标志

- FLAGS寄存器中共有3个控制标志位
  - TF，单步标志。当其置1时，CPU每执行完一条指令，就产生1次内部中断，用户可以单步逐条跟踪程序运行，便于程序调试。
  - IF，中断标志。IF＝1（＝0）时，允许（禁止）CPU响应“可屏蔽中断”。 STI（CLI）指令使IF＝1（＝0）。
  - DF，方向标志。指示串操作指令的地址指针变化方向。若DF＝1（＝0），串操作指令的地址指针自动增量（减量）。STD（CLD）指令使DF ＝1（＝0） 。
  - 3个状态标志的速记方法：TID或DIT

2-2 8086/8088引脚及其功能

- 从最基本的16位开始，循序渐进。
- 8086 CPU的引脚特点
  - 40条引脚，DIP封装，在逻辑上可分为4类

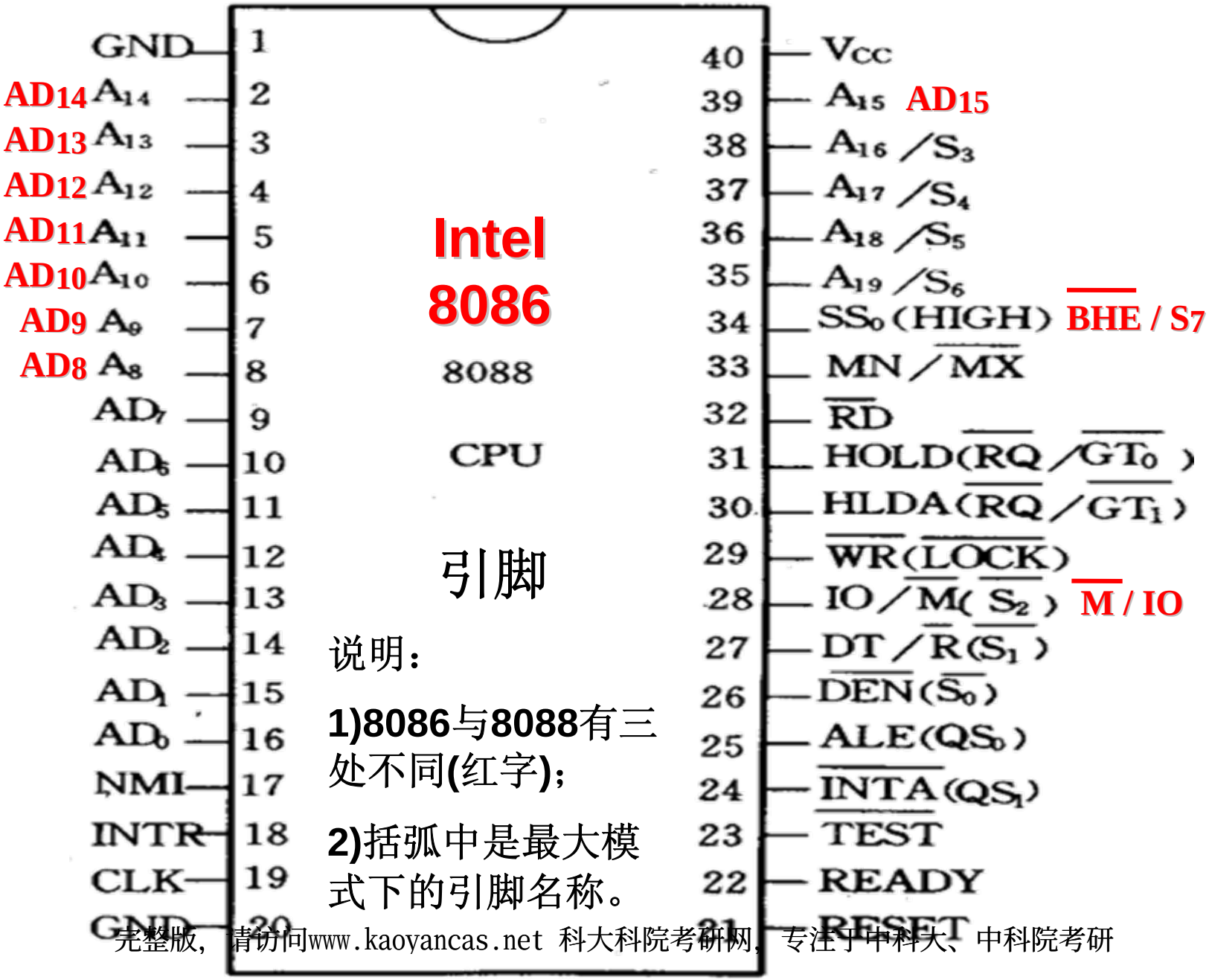
    1) 地址总线信号；

    2) 数据总线信号；

    3) 控制总线信号；

    4) 其它类，如：电源、地、时钟等。

- 8086/8088引脚采用分时复用技术，一条引脚在不同时间代表不同信号，解决引脚不够的问题。
- 8088内部结构与8086几乎完全相同，只是外部数据总线只有8位，引脚有所不同。



说明：

1)8086与8088有三处不同(红字)；

2)括弧中是最大模式下的引脚名称。

8086/8088 CPU的两种工作模式

- 8086/8088有两种工作模式：
  - 最小模式：所有总线控制信号均由CPU产生，适用于单个处理器系统。
  - 最大模式：由外部的总线控制器根据CPU输出的状态信号产生相应的总线控制信号，适用于多处理器系统。PC机采用的是最大模式。
  - 两种模式下CPU的引脚功能有所不同。图2-3中括弧内是最大模式下的引脚名称。

8086/8088的引脚功能介绍

（1）基本引脚信号－1		
名称	方向/特性	功能作用
AD15～AD0	I/O，3S	16位地址/数据复用引脚。HOLD期间为三态高阻。
A19/S6～A16/S3	O，3S	高4位地址/状态复用引脚。访问M时，A19～A0构成20位地址，可访问1MB空间；8086的I/O端口空间只有64K，访问I/O时，A19～A16为“0”。S6＝0指示CPU连在总线上；S5＝IF；S4和S3指示当前使用的段寄存器（表2-3)。
-BHE/S7	O，3S	高字节允许/状态复用引脚。在读写操作期间，－BHE有效时指示高8位数据总线上数据有效，AD0＝0指示低8位上数据有效。
-RD	O，3S	读选通信号，低电平有效。
-WR	O，3S	写选通信号，低电平有效。

（1）基本引脚信号－2

名称	方向/特性	功能作用
NMI	**I**	非屏蔽中断请求线，上升沿触发。**CPU**收到**NMI**后，在完成当前指令后进入中断处理程序。
INTR	**I**	可屏蔽中断请求线，高电平或上升沿触发。**CPU**在每条指令结束前的最后一个时钟周期检查该信号，若有效且**IF＝1**，本次指令结束后转入中断响应周期。
CLK	**I**	时钟信号，处理器基本定时脉冲。
RESET	**I**	复位信号，高电平并且至少保持**4**个时钟周期以上，**CPU**进入复位状态；重新变低后**CPU**脱离复位进入重启动，**8086**从**FFFF0H**开始执行指令。
READY	**I**	准备好信号，高有效，读写速度同步控制信号。
-TEST	**I**	测试信号，低有效，执行**WAIT**指令的控制信号。
MN/-MX	**I**	最小/最大工作模式选择信号，**MN/-MX**接高电平**(5V)**为最小模式， **MN/-MX**接地为最大模式。
VCC		处理器的电源引脚，接 **+5V**。
GND		处理器的地线引脚，接系统地线。

（2）最小模式下的有关控制信号－1		
名称	方向/特性	功能作用
-INTA	O	最小模式下对外部INTR中断请求的响应信号。CPU进入中断响应周期后，利用两个完整的总线周期连续发出两个-INTA脉冲（T2～T4）周期，读取中断类型码。
ALE	O	地址锁存允许信号。作为锁存器的锁存控制信号，用于分离分时复用的地址和数据总线。
-DEN	O,3S	数据总线缓冲器允许信号。
DT/-R	O,3S	双向数据总线缓冲器的方向控制信号。高电平对应CPU写操作，低电平对应CPU的读操作。
M/-IO	O,3S	存储器或I/O接口选择信号。高电平访问M，低电平访问I/O端口

（3）最大模式下的有关控制信号

名称	方向/特性	功能作用
-S2, -S1, -S0	O，3S	总线周期状态输出信号，外部总线控制器根据此信号产生各类总线命令和控制信号。CPU在每个总线周期的最后一个时钟（T4）输出下一个周期的状态信号。见P32表2-4。
-LOCK	O，3S	总线封锁信号。信号有效时不允许其它主控部件占用总线，例如在连续两个-INTA期间，以及前面加Lock前缀的指令执行期间。
-RQ/-GT0, -RQ/-GT1	I/O	最大模式总线请求/总线响应信号，双向。输入时表示其它主控设备向CPU申请总线使用权的RQ信号；输出时是CPU对于总线请求的应答（响应）GT信号。两条线可以分别连接两个主控设备，-RQ/-GT0比-RQ/-GT1的优先权高。主要用于CPU与数值协处理器以及IO处理机的总线使用权协调。
QS1, QS0	O	指令队列状态编码信号，表明8086当前指令队列的状态。外部（如Intel8087协处理器）可以通过这两个信号跟踪8086/8088内部指令队列的变化。

8088与8086的区别

- 内部指令队列只有4个字节。当队列出现有一个空闲字节时，BIU就会自动访问存储器预取指令进行队列填充。
- BIU的总线控制电路与外部交换数据的数据总线是8位，与BIU内部的段寄存器和IP之间的数据总线也是8位的，16位指令和数据的读写要分两个总线周期才能完成（对比图2-6和图2-1中的BIU单元）
- 8088用IO/-M代替8086的M/-IO。
- 8088用-SS0代替-BHE，-SS0与IO/-M、DT/-R组合编码反映了最小模式下8088 CPU的总线操作周期（见表2-6）

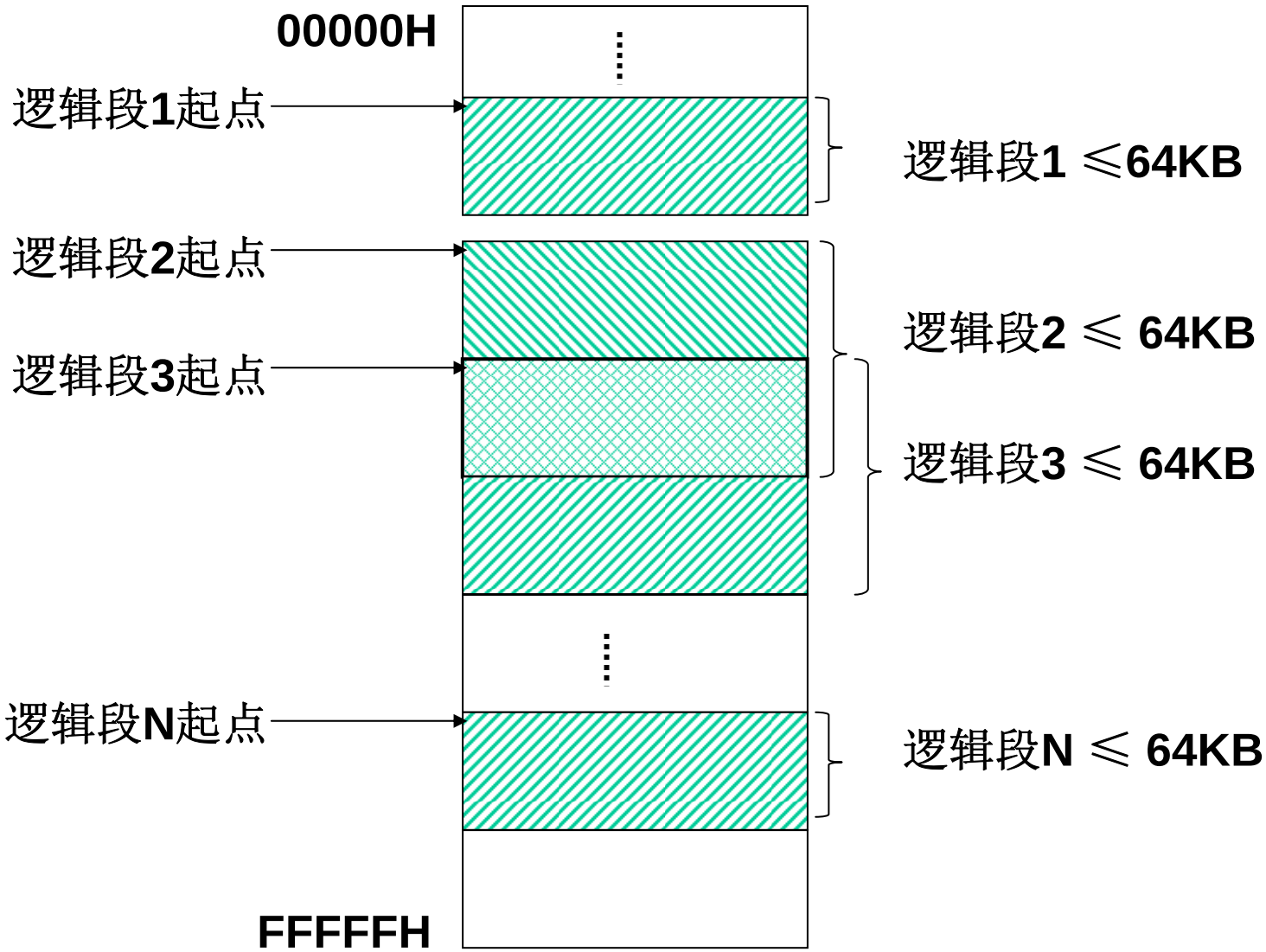
2-3、8086存储器组织

一、存储器的分段

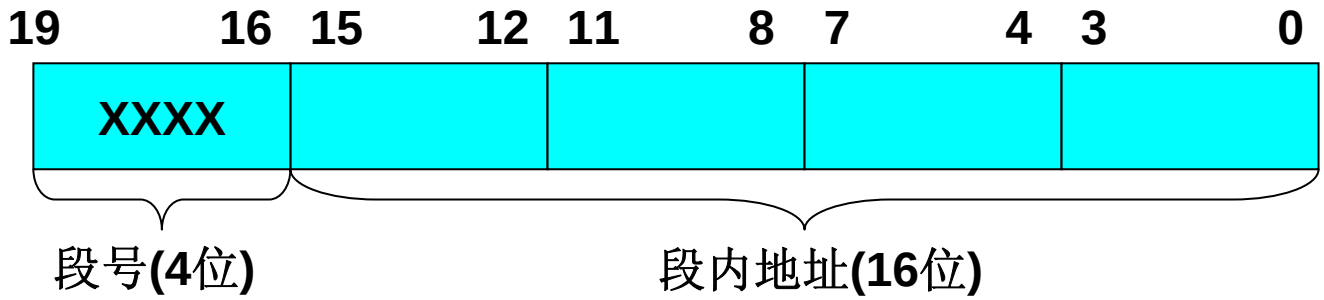
1)*为什么要采用存储器“分段”技术？*

- 实模式下CPU可直接寻址的地址空间为2^{20}＝1M字节单元。CPU需输出20位地址信息才能实现对1M字节单元存储空间的寻址。
- 实模式下CPU中所使用的寄存器均是16位的，内部ALU也只能进行16位运算，其寻址范围局限在2^{16}＝65,536(64K)字节单元。
- 为了实现对1M字节单元的寻址，80x86系统采用了存储器分段技术。

- 具体做法：将**1M**字节空间分成许多逻辑段，每段最长**64K**字节单元，可用**16**位地址码进行寻址。
- 每个逻辑段在实际存储空间中位置可以浮动，其起始地址由段寄存器的内容来确定。实际上，段寄存器中存放的是段起始地址的高**16**位，称之为：**“段基值” (segment base value)**。
- 各个逻辑段在实际的存储空间中可以完全分开，也可以部分重叠，甚至完全重叠。
- 段的起始地址的计算和分配通常是由操作系统完成的，并不需要普通用户参与。
- 逻辑段的物理存储位置如图所示。



- 事实上还有多种方法可将**1M**字节单元的物理存储器空间分成可用**16**位地址码寻址的逻辑段。
- 例如将**20**位物理地址分成两部分： 高**4**位为段号，可用机器内设置的**4**位长的“段号寄存器”来保存，低**16**位为段内地址，也称“偏移地址”，如下图所示：



这种分段方法的不足：

(1) **4**位长的“段号寄存器”与其它寄存器不兼容，操作麻烦。

(2) 每个逻辑段大小固定为**64K**字节单元，当程序中所需的存储空间不是**64K**字节单元的倍数时，就会浪费存储空间。

对比之下，**80x86**采用的分段机制则要灵活方便的多。

物理地址和逻辑地址

- 在有地址变换机构的计算机系统中，每个存储单元可以看成具有两种地址：物理地址和逻辑地址。
- 物理地址是信息在存储器中实际存放的地址，它是CPU访问存储器时实际输出的地址。
- 例如，实模式下的80x86/Pentium系统的物理地址是20位，存储空间为2^{20}＝1M字节单元，地址范围从00000H到FFFFFH。
- CPU和存储器交换数据所使用的就是这样的物理地址。

- 逻辑地址是编程时所使用的地址，或者说程序设计时所涉及的地址是逻辑地址而不是物理地址。
- 编程时不需要知道产生的代码或数据在存储器中的具体物理位置。这样可以简化存储资源的动态管理。
- 在实模式下的软件结构中，逻辑地址由“段基值”和“偏移量”两部分构成。
- 逻辑地址的表示形式为“段基值：偏移地址”

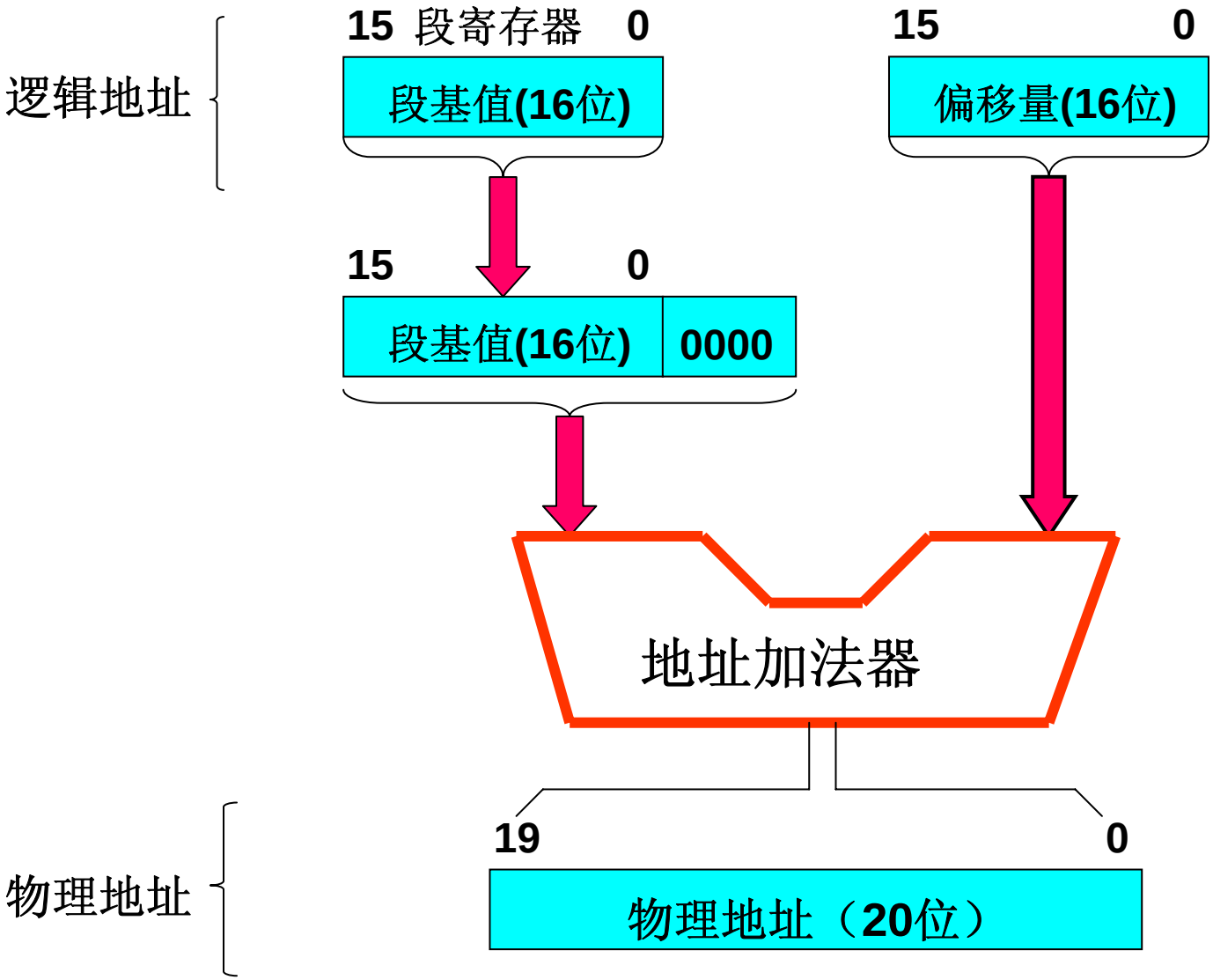
段基值和偏移量

- “段基值”是段的起始地址的高16位。
- “偏移量”(offset)也称偏移地址，它是所访问的存储单元与段起始地址之间的字节距离（跨度）。
- 给定段基值和偏移量，就可以在存储器中寻址所访问的存储单元。
- 在实模式下，“段基值”和“偏移量”均是16位的。
- “段基值”由段寄存器CS、DS、SS、ES以及FS和GS（80386以上CPU增加的寄存器）提供；
- “偏移量”由BX、BP、SP、SI、DI、IP或以这些寄存器的组合形式来提供。

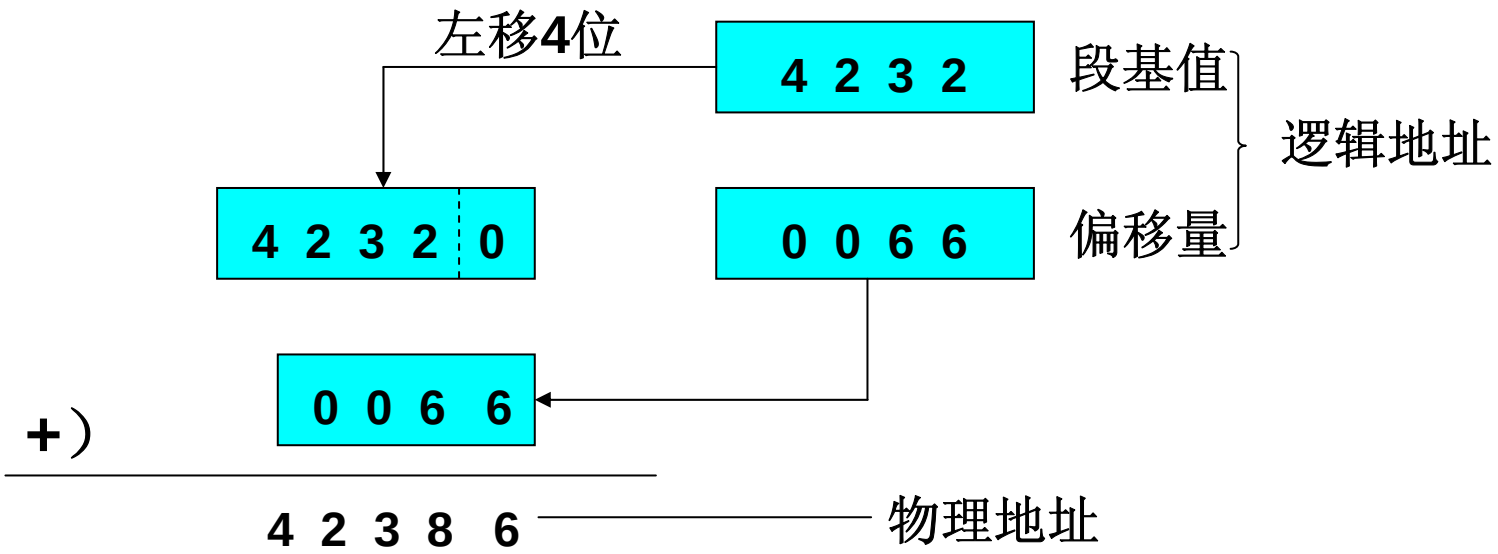
物理地址的形成

- 实模式下CPU访问存储器时的20位物理地址由逻辑地址转换而来。
- 具体方法：段寄存器中的16位“段基值”左移4位（低位补0），再与16位的“偏移量”相加，即：
- 物理地址＝段基值× 10H ＋ 偏移地址

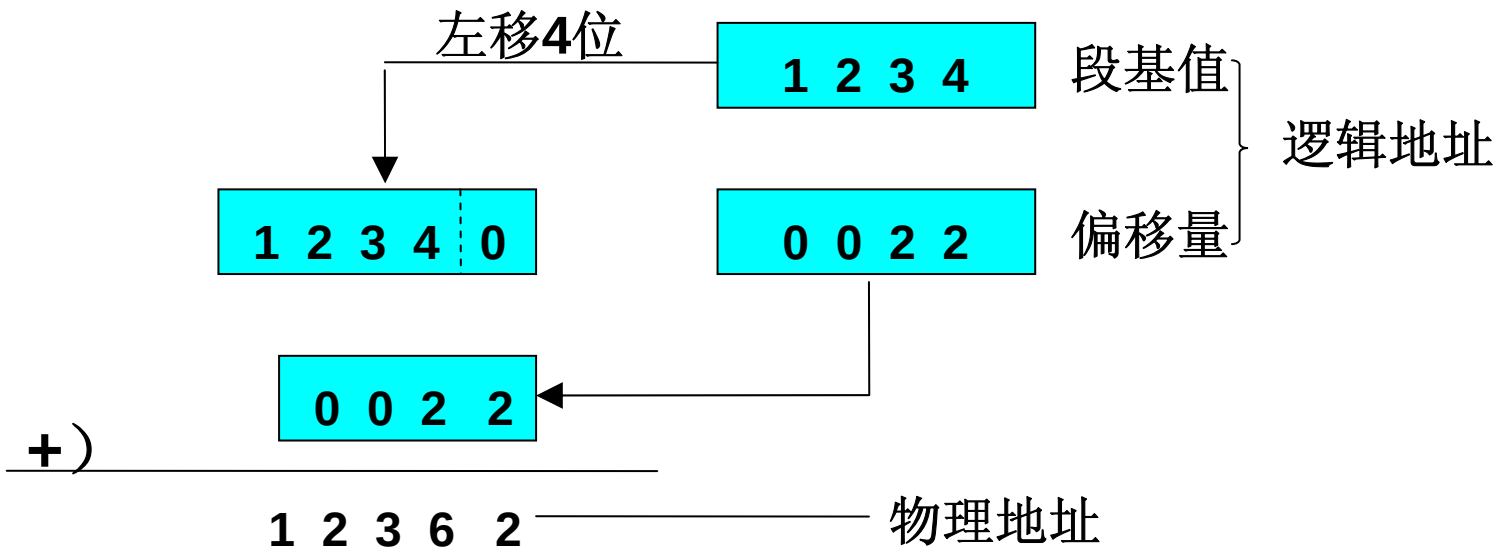
- 教材P36图2-8



- 例1：设代码段寄存器CS的内容为4232H，指令指针寄存器IP的内容为0066H，即CS＝4232H，IP＝0066H，则访问代码段存储单元的物理地址计算如下：

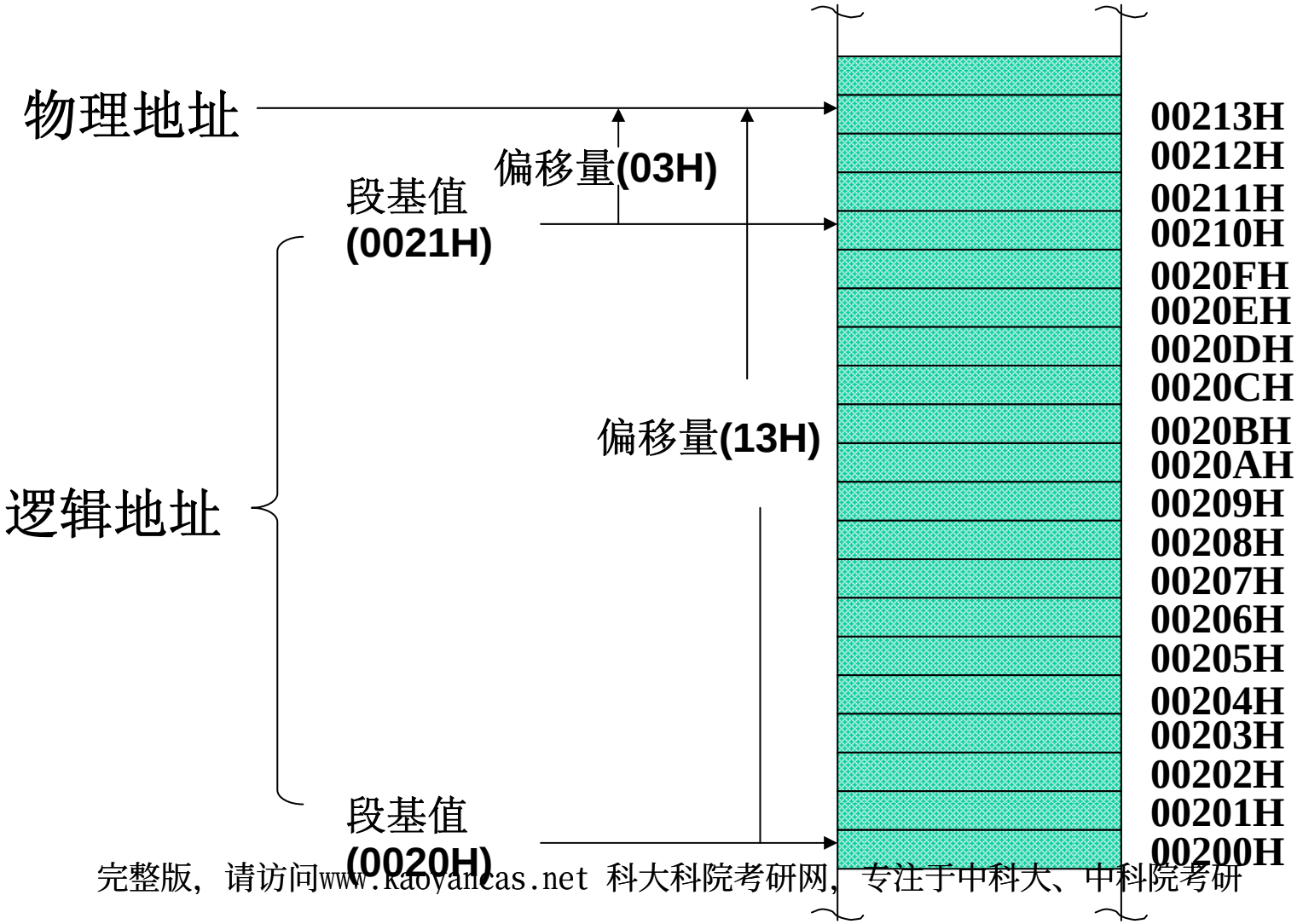


- 例2 设数据段寄存器DS的内容为1234H，基址寄存器BX的内容为0022H，即DS＝1234H，BX＝0022H，则访问数据段存储单元的物理地址计算如下：



- 注意：每个存储单元有唯一的物理地址，但它可以由不同的“段基值”和“偏移量”转换而来，这只要把段基值和偏移量改变为相应的值即可。
- 换言之，同一个物理地址可以由不同的逻辑地址来构成。或者说，同一个物理地址与多个逻辑地址相对应。
- 例如：
  - 段基值为**0020H**，偏移量为**0013H**，构成的物理地址为**00213H**；
  - 若段基值改变为**0021H**，配以新的偏移量**0003H**，其物理地址仍然是**00213H**，如下图所示。

示意图：一个物理地址对应多个逻辑地址



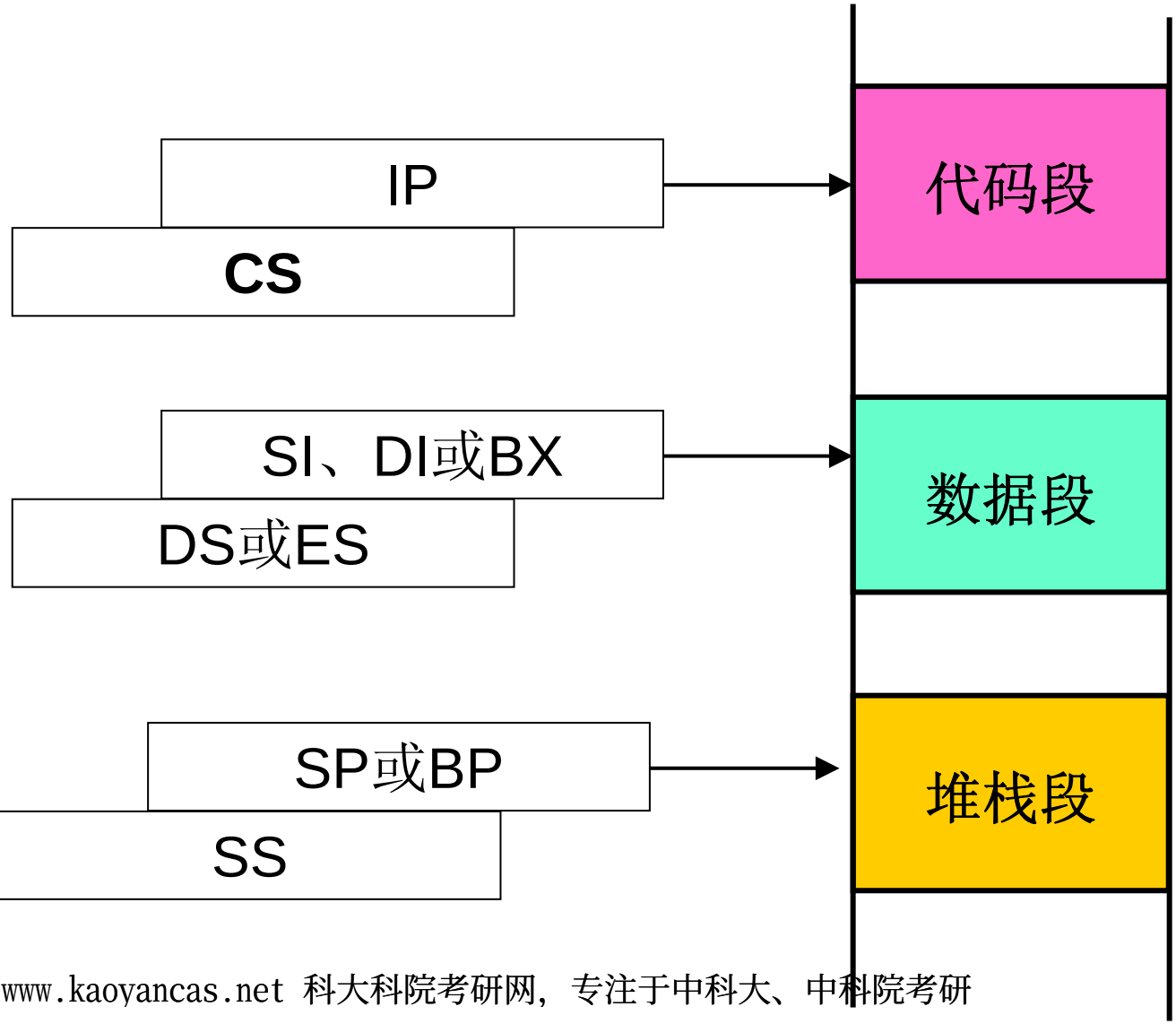
- 在这种“段加偏移”的寻址机制中，由于是将段寄存器的内容左移4位（相当于乘以十进制数16）来作为段的起始地址的，所以实模下各个逻辑段只能起始于存储器中16字节整数倍的边界。
- 这样可以简化实模式下CPU生成物理地址的操作。通常称这16字节的小存储区域为“分段”或“节”(paragraph)。
- 在“段加偏移”的寻址机制中，微处理器有一套用于定义各种寻址方式中段寄存器和偏移地址寄存器的组合规则。

80x86默认的16位“段+偏移”寻址组合

段寄存器	偏移地址寄存器	主要用途
CS	IP	指令地址
SS	SP或BP	堆栈地址
DS	BX、DI、SI或者 8位或16位立即数	数据地址
ES	串操作指令的DI	串操作目的地址

存储单元寻址示意图

P36：图2-9

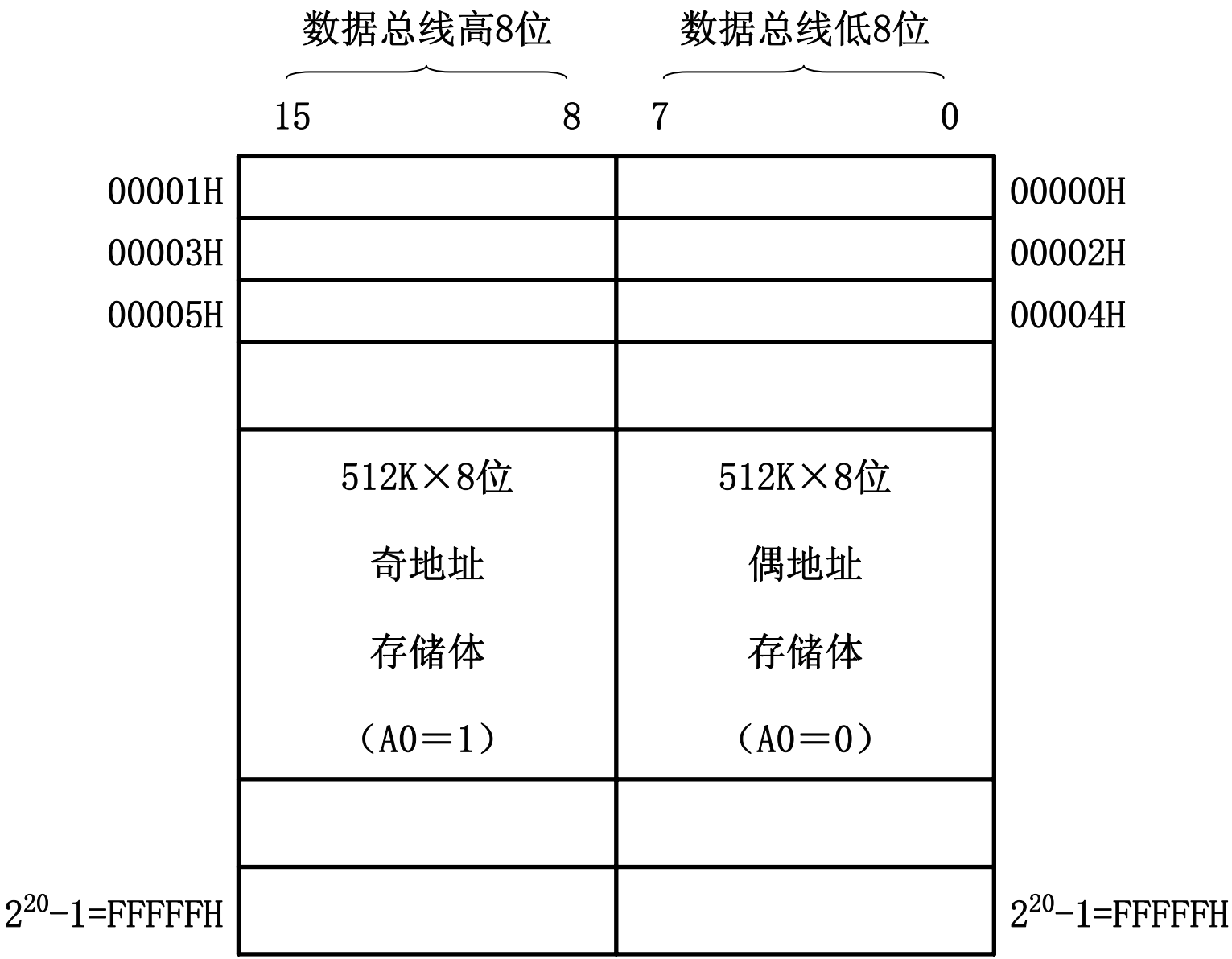


80x86默认的32位“段+偏移”寻址组合

段寄存器	偏移地址寄存器	主要用途
CS	EIP	指令地址
SS	ESP或EBP	堆栈指针
DS	EAX、EBX、ECX、EDX、 EDI、ESI、8位、16位或32位立即数	数据地址
ES	串操作指令的EDI	串操作目的地址
FS	无默认	一般地址
GS	无默认	一般地址

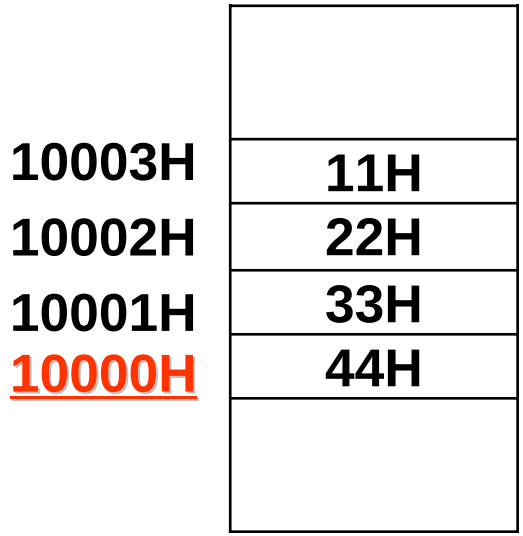
二、8086存储器的分体结构

- 存储器是按字节组织的，两个相邻字节称为一个“字”。
- 存放的信息若是以字节（8位）为单位，在存储器中按顺序排列存放。
- **8086存储器的1M字节存储空间分成两个512KB字节的存储体。低位存储体与8086CPU的低8位数据线D7~D0相连，该存储体中的每个地址均为偶地址。高位存储体与8086CPU的高8位数据线D15~D8相连，该存储体中的每个地址均为奇地址。(P37图2-10）**



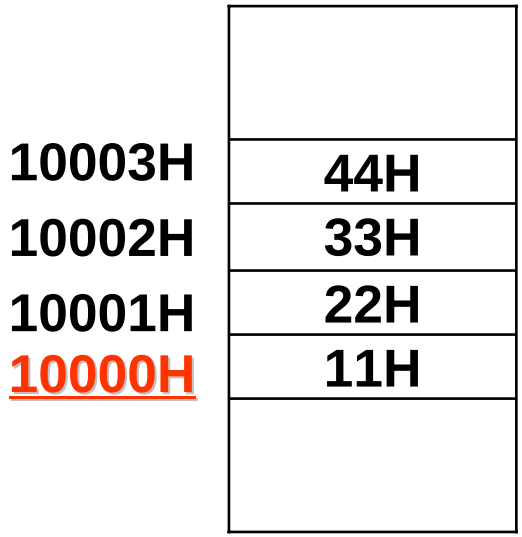
多字节存储字的两种不同存放方式

- Intel 80x86方式：
  - 一个多字节的存储字的地址是多个连续字节单元中最低端字节单元的地址，而此最低端存储单元中存放的是多字节存储字中最低字节。
  - 例如：一个**32**位（**4**字节）的存储字**11223344H**在内存中的存放情况如右图所示，该**32**位存储字的地址即是**10000H**。此格式有人称其为**“小尾存储格式”**（**little endian memory format**）。



Intel 80x86系统存储格式示意图

- **Motorola的680x0方式：**
  - 与**Intel80x86**正好相反。一个多字节的存储字的地址是多个连续字节单元中最高端字节单元的地址，
  - 例如： **32**位（**4**字节）的存储字**11223344H**在内存中的存放情况如右图所示。该存储字的地址也是**10000H**，但是地址指向的是存储字的最高字节（**11H**）的存储单元。此格式有人称其为**“大尾存储格式”**（ **big endian memory format**）。



Motorola的680x0系统存储格式示意图

“对准存放”

- 多字节数据可以从偶地址开始存放，也可以从奇地址开始存放。但是在8086系统中，存储器的访问都是以字或为单位进行的，并从偶地址开始。
- 若多字节数据从偶地址开始存放，当CPU读写一个字时，只需要访问一次存储器。否则CPU要两次访问存储器，分别读取高低两个字节。
- 为减少不必要的开销，编程时注意从偶地址开始存放数据。这种方式称为：

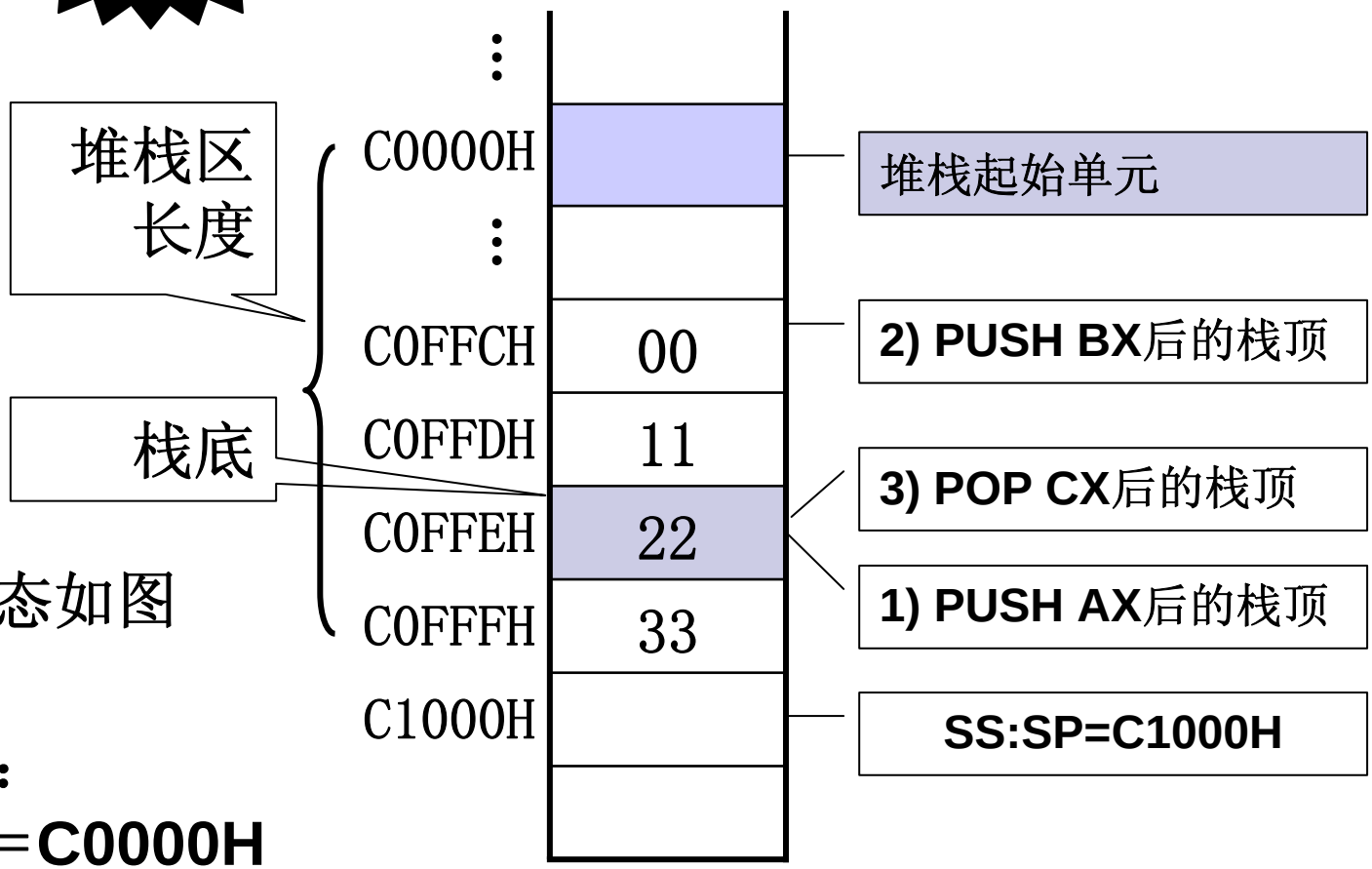
三、堆栈的概念

- 堆栈是在内存中用来存放需要暂存数据的一个特定区域，堆栈段是由段定义语句定义的最大空间为64KB的一个段。
- 堆栈操作以字为单位对准存放，并且按照先入后出（FILO）原则。
- 堆栈地址增长方向是地址逐步减小，栈底地址大于栈顶地址。
- 段基址由SS指定，栈顶由SP指定，SP可以指向当前栈顶单元，也可以指向栈顶上的一个空单元。80x86系统采用的是SP指向当前栈顶单元。

- 堆栈区**最高地址－1的单元为栈底**，最后进栈数据所对应的地址单元为**栈顶**，SS指向栈的**起始单元**（注意：起始单元不是栈底，见下张幻灯片图）。
- SP寄存器**动态**跟踪栈顶位置，初始化时SP的值为堆栈的长度，即指向栈底＋2单元。
- 将数据送入堆栈叫**压栈**或入栈，PUSH），从栈中取出数据叫**弹出**（或出栈POP），PUSH和POP均以字为单位。
- 当执行PUSH指令时，CPU自动修改栈顶指针SP－2→SP，使之指向新的栈顶，再将入栈数据低位数据压入SP单元，高位数据压入SP＋1单元。
- 当执行POP指令时，CPU先将当前栈顶SP（低位数据）和SP＋1（高位数据）弹出，然后再自动修改栈顶指针SP＋2→SP，使之指向新的栈顶。

例

AX＝3322H
BX＝1100H
SS＝C000H
SP＝1000H
执行
PUSH AX
PUSH BX
POP　CX
后的堆栈区状态如图



堆栈起始单元：
SS×10H＋0＝C0000H
堆栈长度：
SP＝1000H

- 堆栈使用要点：
  - 在进入中断服务子程序或调用子程序之前，利用堆栈保存当前CPU的运行现场（指令指针IP和有关寄存器的内容），子程序运行结束后再从堆栈恢复保存的信息。
  - FILO原则

    教材P40例2-5
  - 匹配原则

    教材P40例2-6

习题：P55～P56

- 2
- 5
- 7
- 9
- 10：a）、c）
- 11：b）、d）
- 12
- 13
- 14
- 16