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。

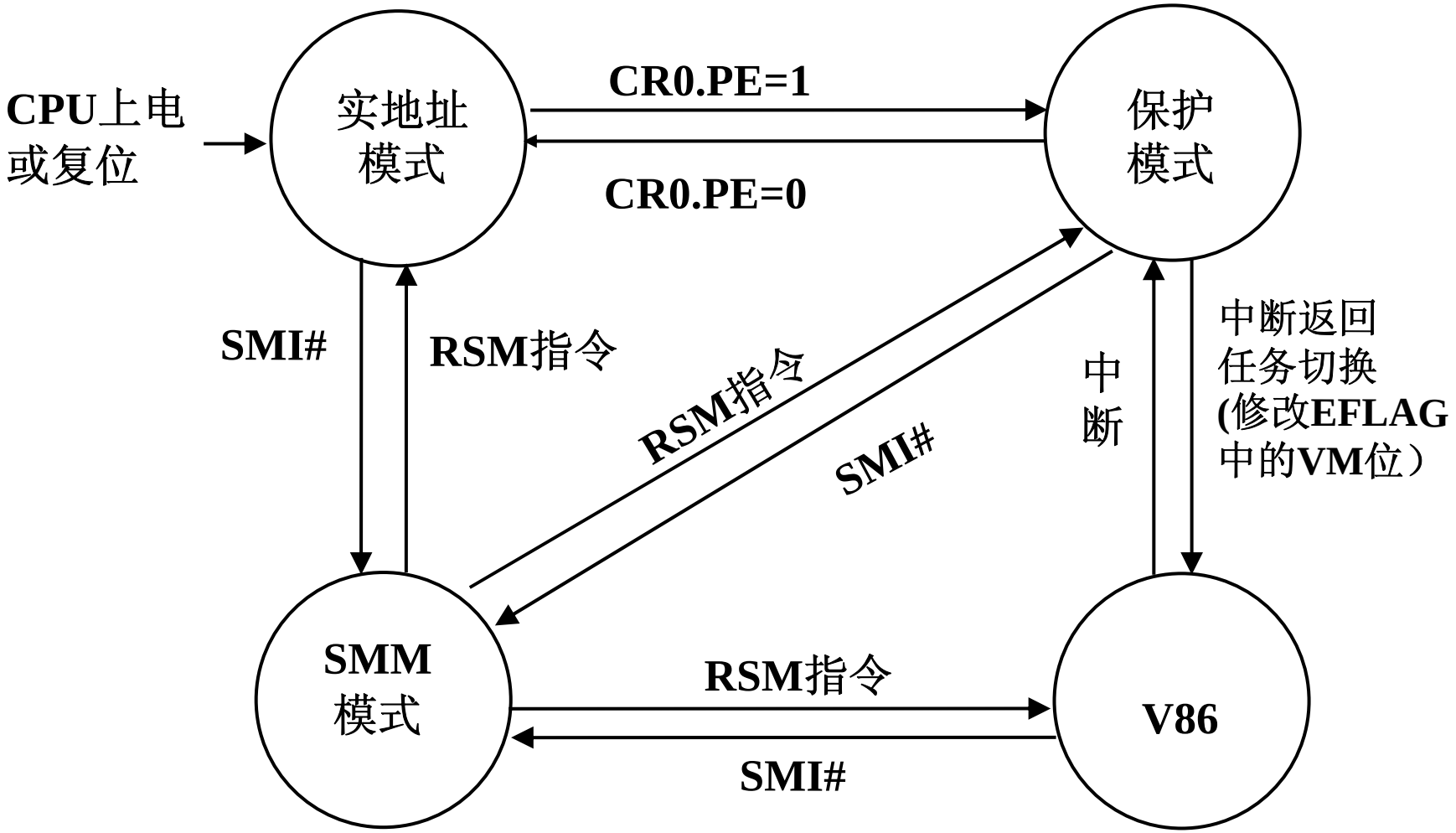
（3）虚拟8086模式

- 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

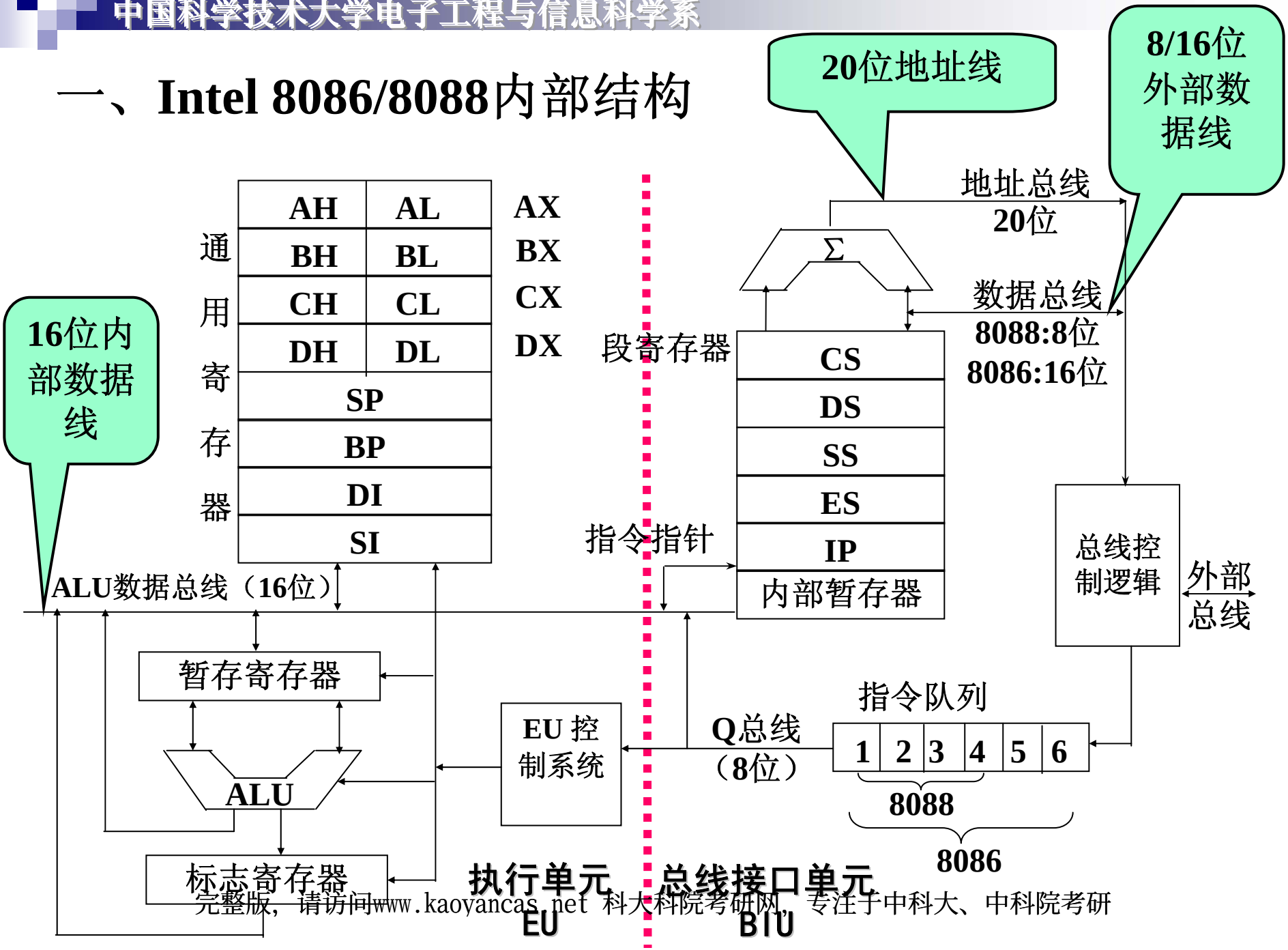


SMI#－系统管理中断信号；RSM指令是系统管理中断服务程序返回指令；修改控制寄存器CR0的PE=1（0）进入（退出）保护模式。

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。

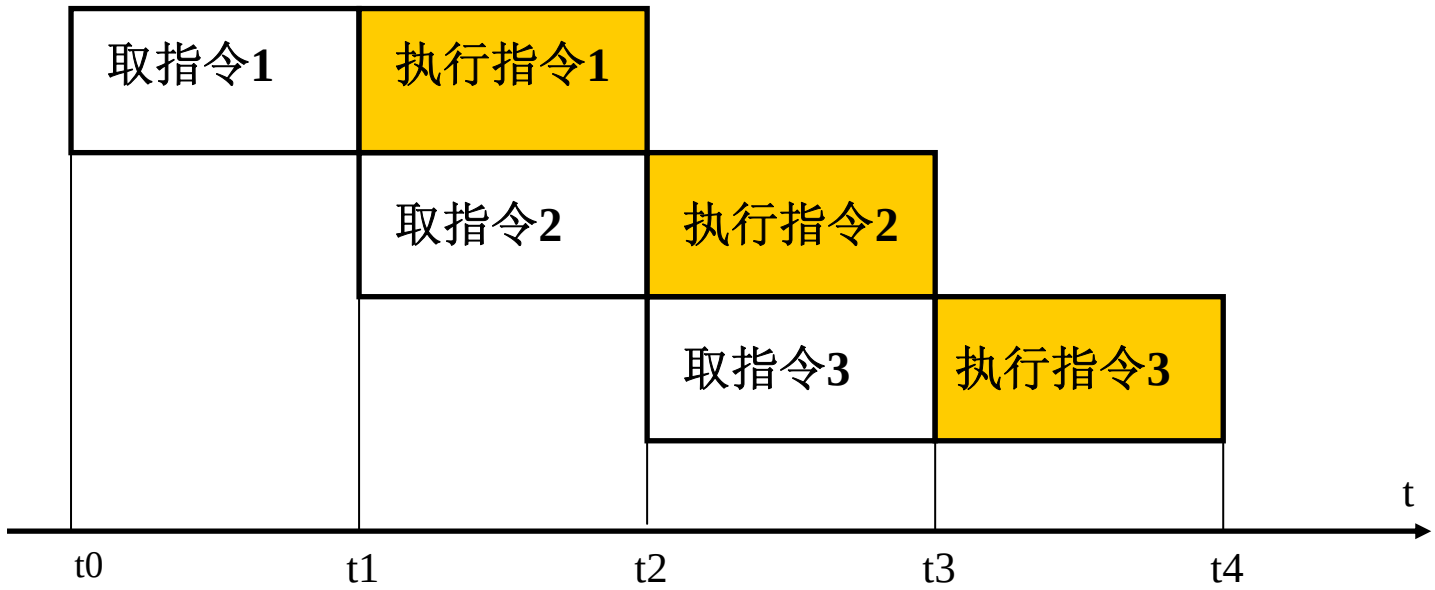
执行部件EU－Execute Unit

- 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。

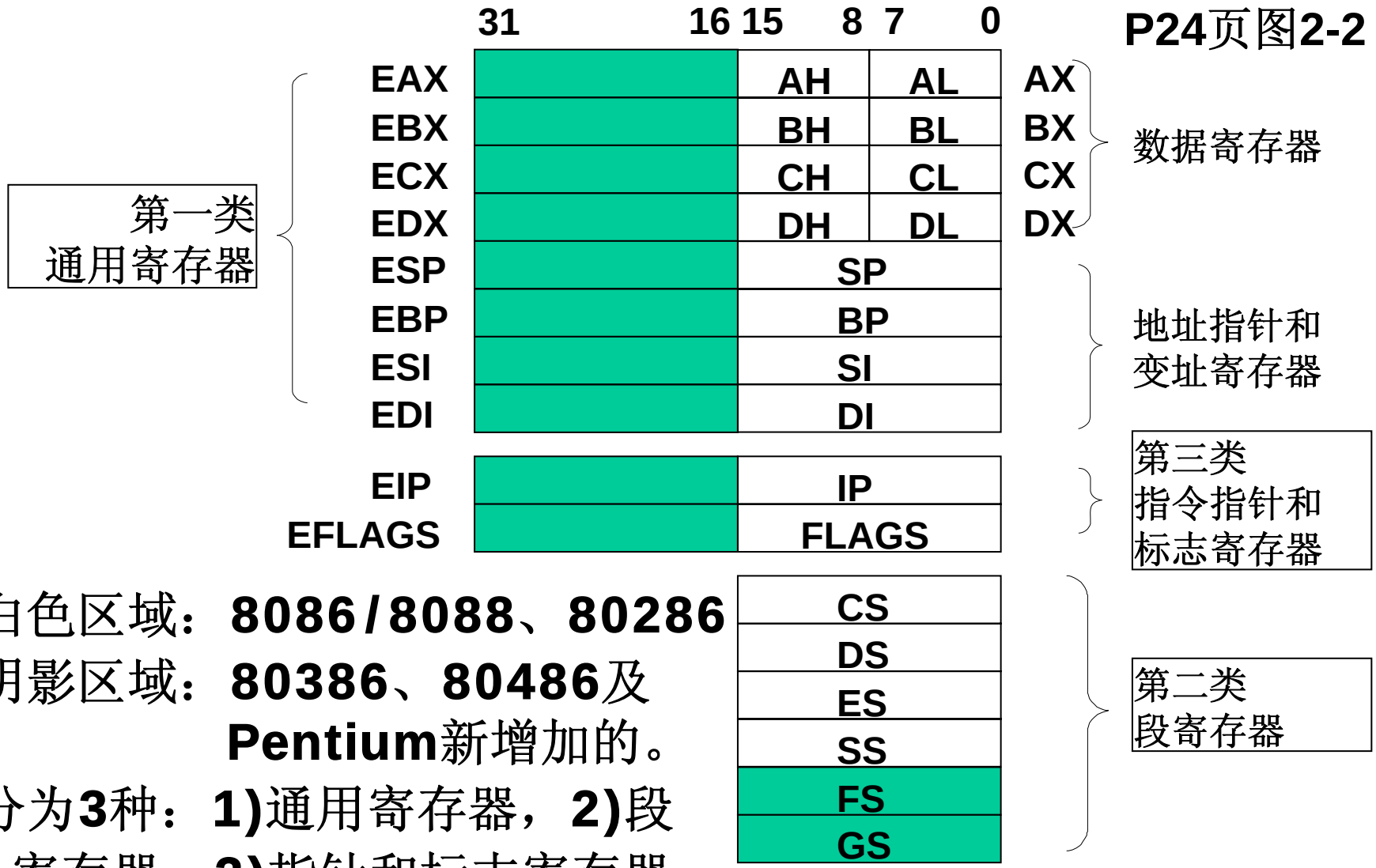
  2)系统级寄存器：80286以后新增加的，包括控制寄存器和支持存储器管理的段表寄存器。控制寄存器主要供操作系统使用，操作系统设计者要熟悉这些寄存器；段表寄存器在应用程序设计时不能直接访问，但能被系统软件访问或被间接引用，因此又称为程序不可见寄存器。

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。

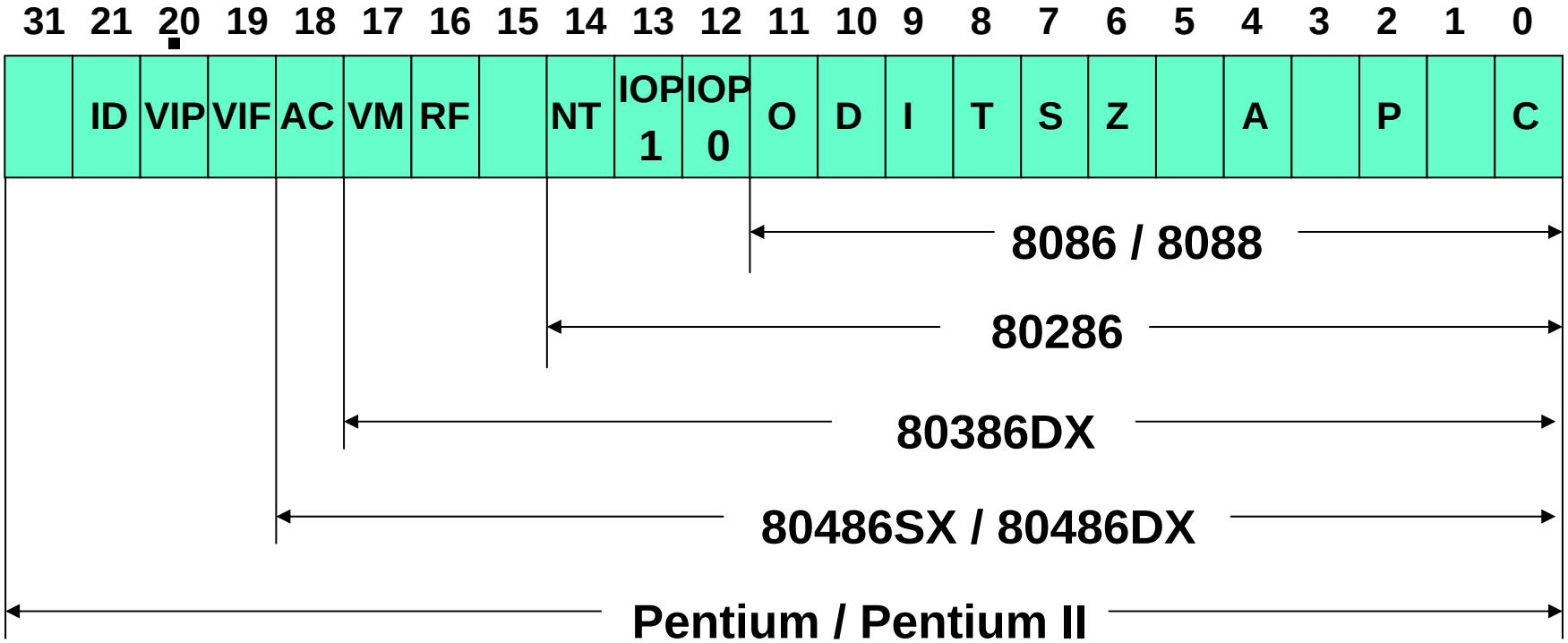
（三）指针和状态标志寄存器

- 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

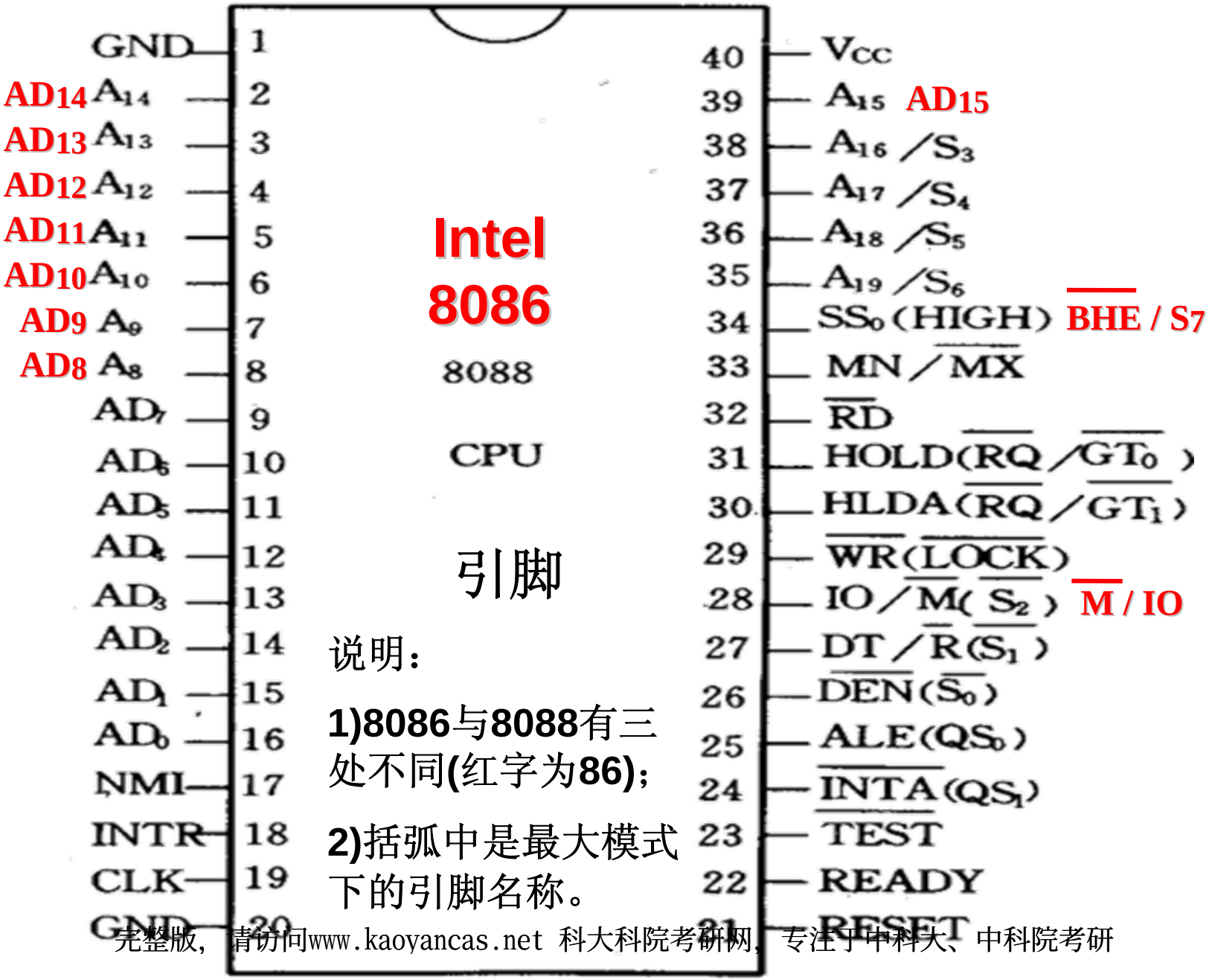
    1) 地址总线信号；

    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。



Intel 8086

8088 CPU

引脚

说明：

1)8086与8088有三处不同(红字为86)；

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**	不可屏蔽中断请求线，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（T1～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）

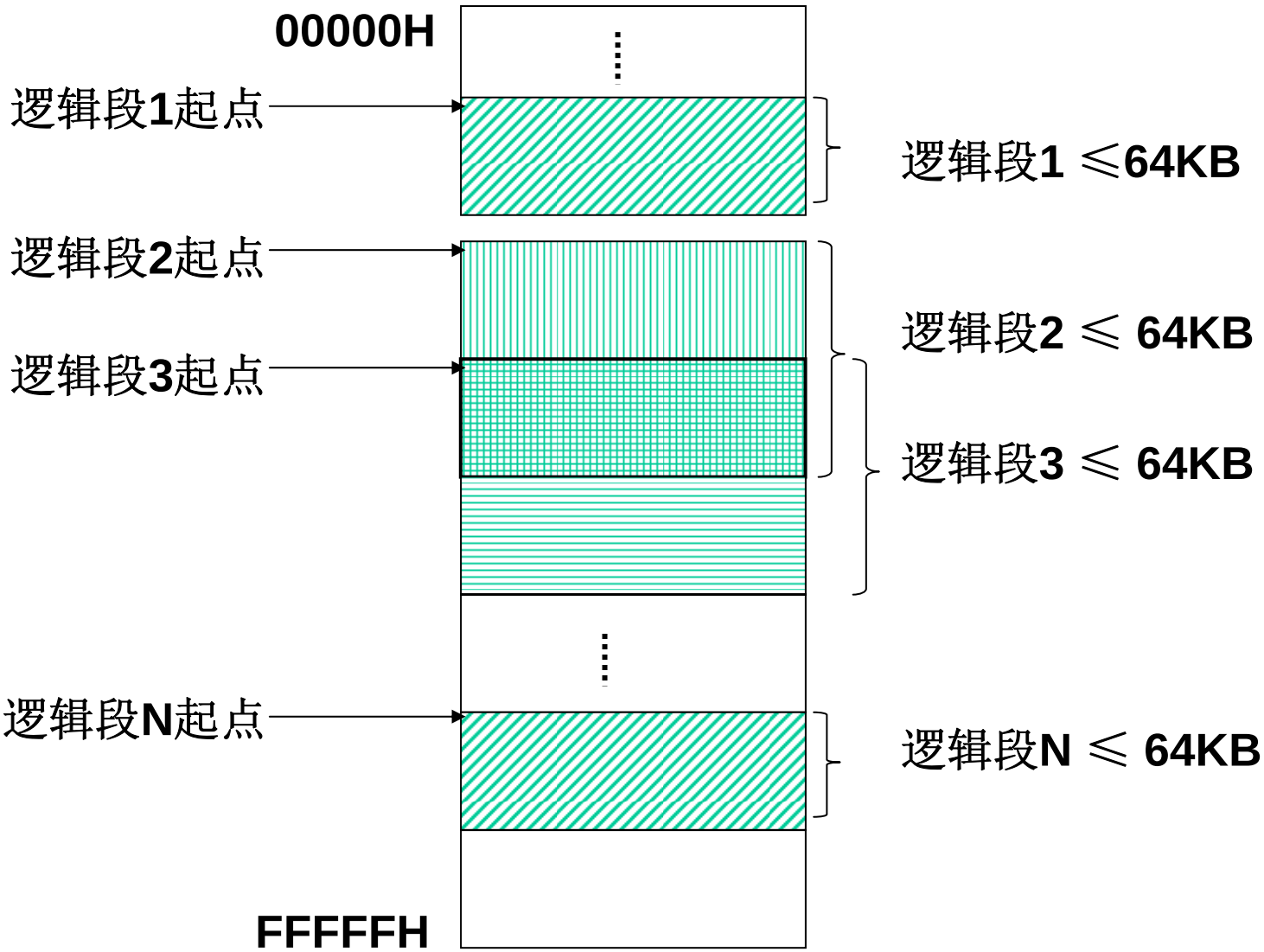
2-3、8086存储器组织

一、存储器的分段

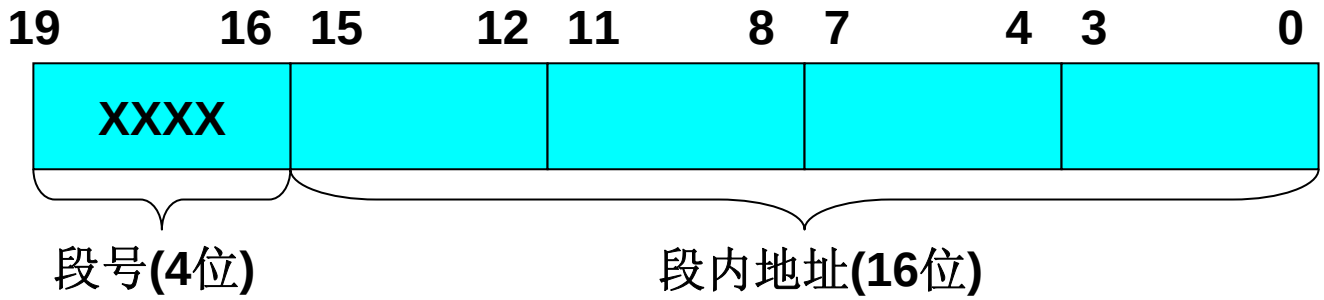
1)*为什么要采用存储器“分段”技术？*

- 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”。

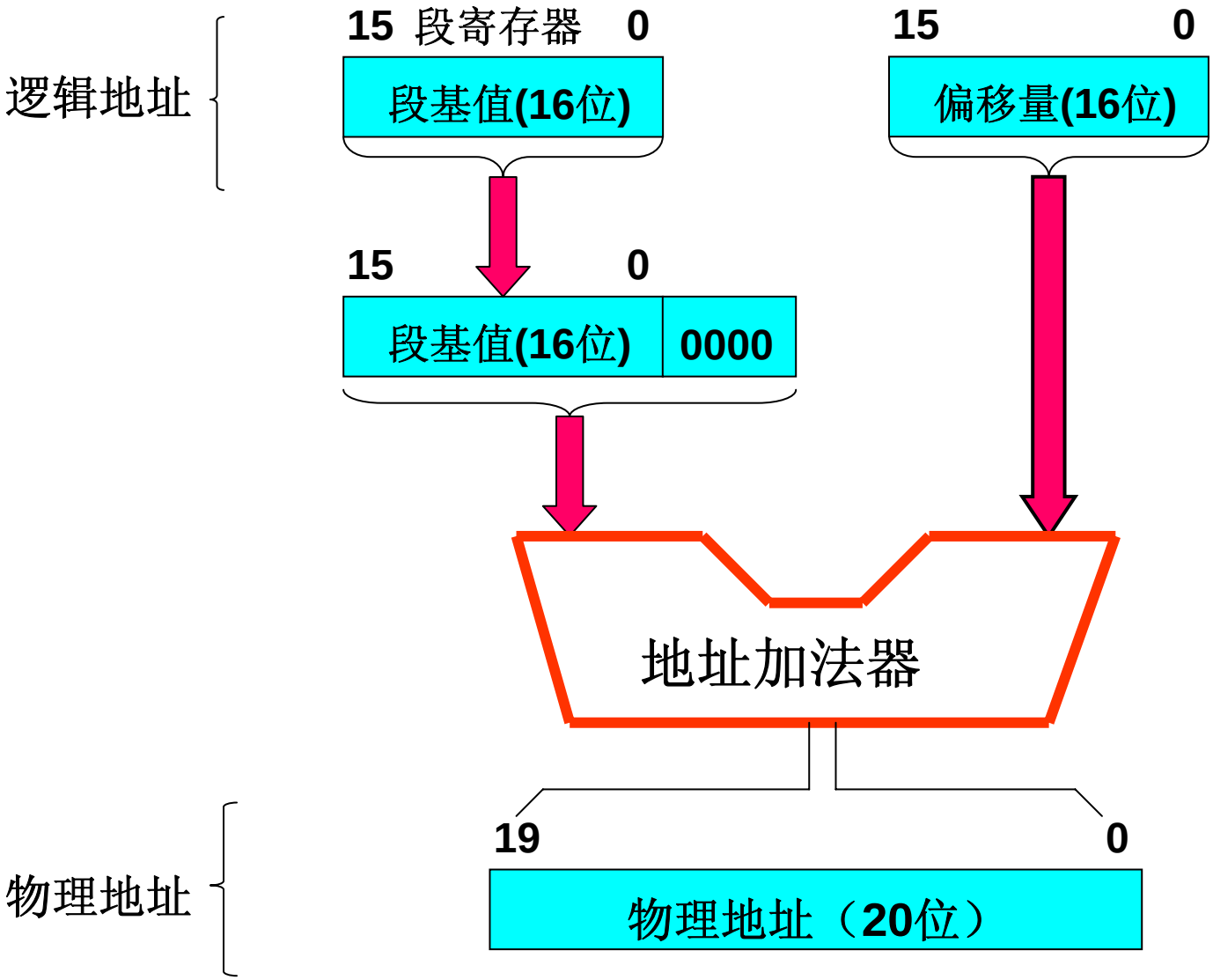
段基值和偏移量

- “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。

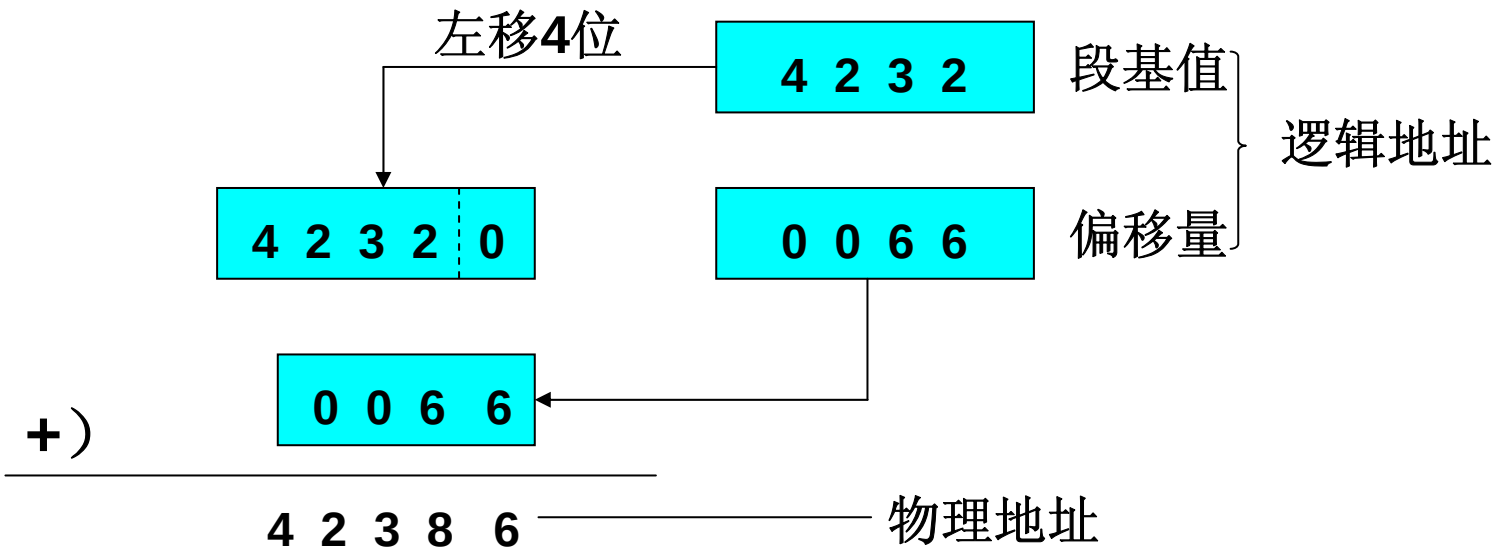
物理地址的形成

- 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

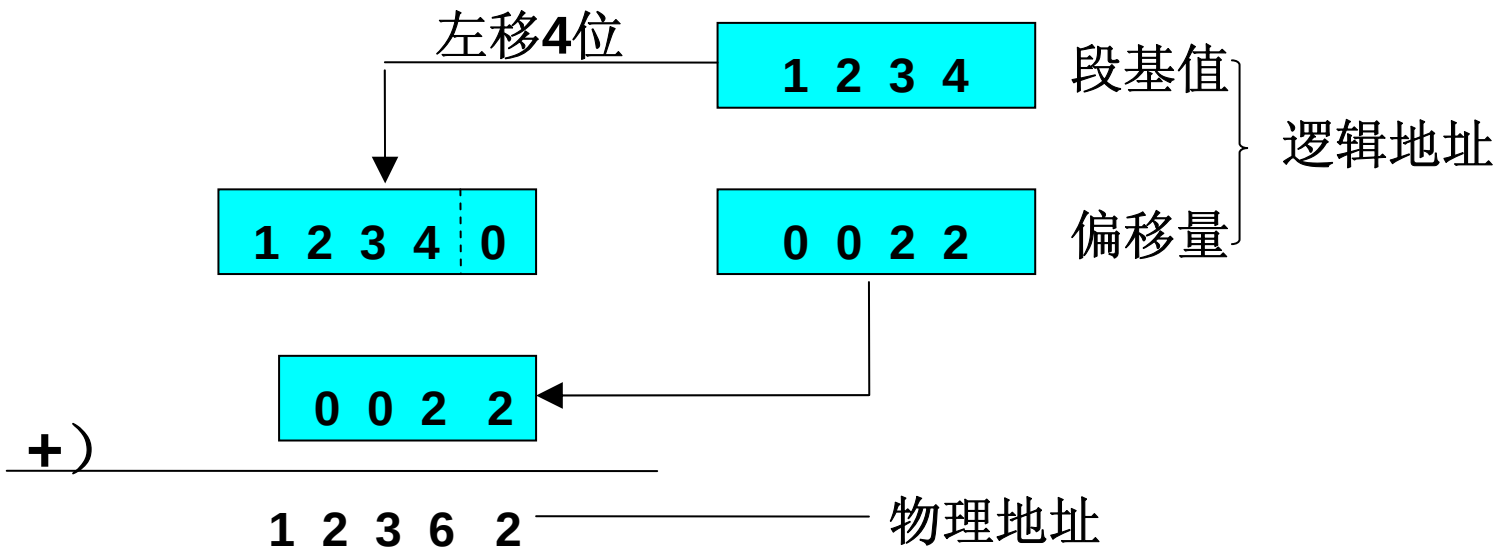
- 教材P36图2-8



- 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逻辑地址为CS:IP=4232H:0066H；其物理地址计算如下：

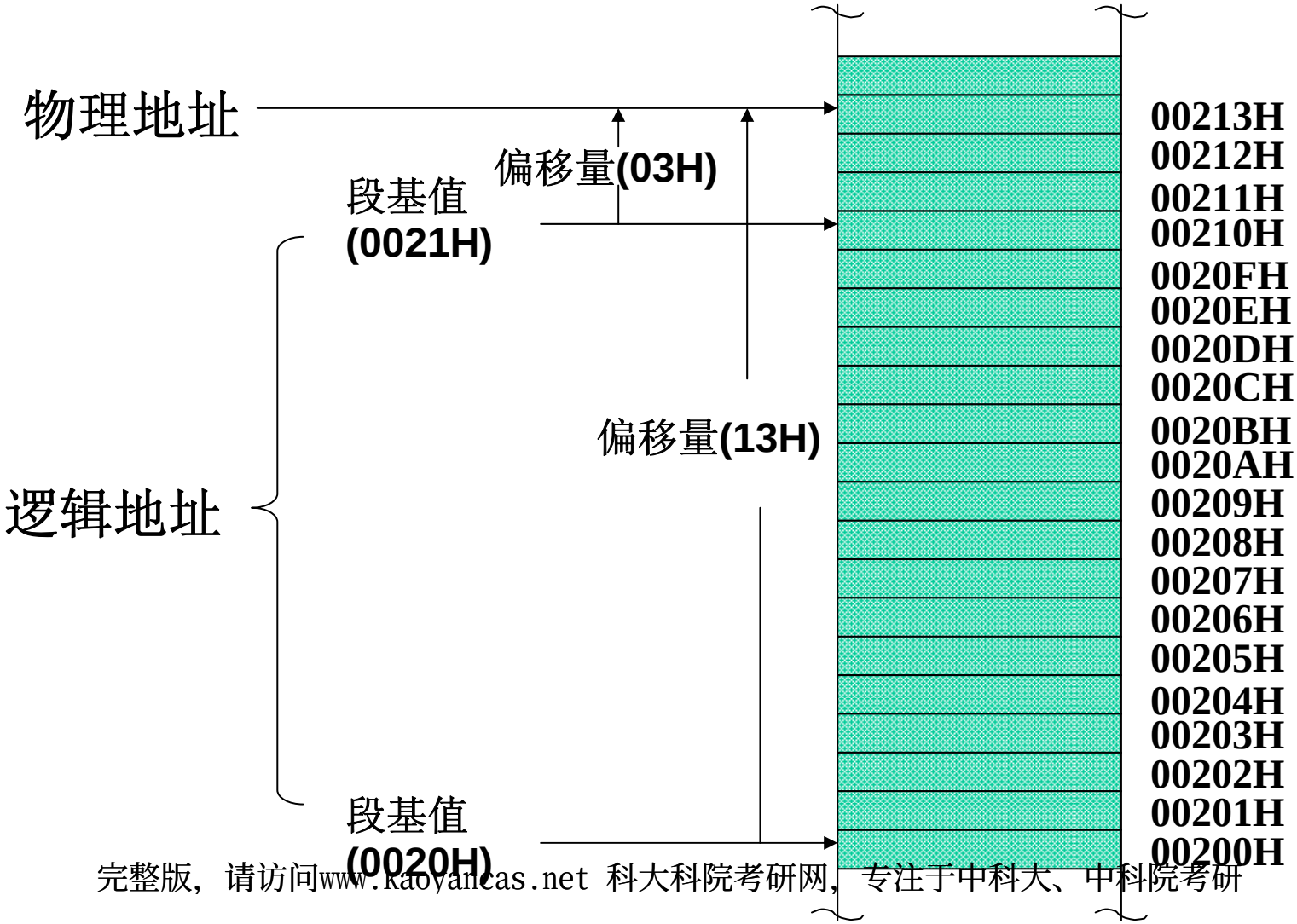


- 例2 设数据段寄存器DS的内容为1234H，基址寄存器BX的内容为0022H，即DS＝1234H，BX＝0022H，则访问数据段存储单元的逻辑地址表示为DS:BX=1234H:0022H，物理地址计算如下：



- 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



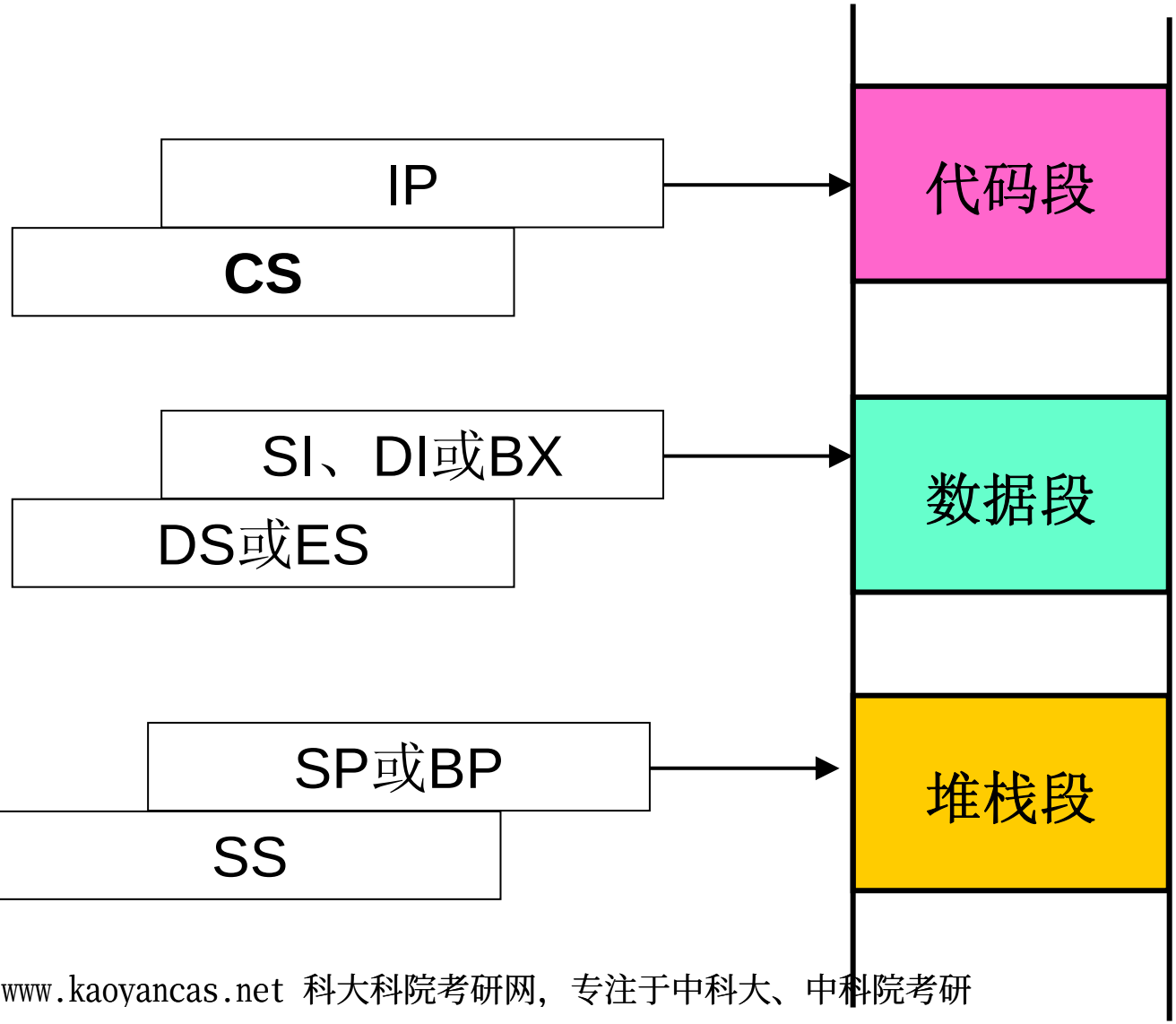
- 在这种“段加偏移”的寻址机制中，由于是将段寄存器的内容左移4位（相当于乘以十进制数16）来作为段的起始地址的，所以实模下各个逻辑段只能起始于存储器中16字节整数倍的边界。
- 这样可以简化实模式下CPU生成物理地址的操作。通常称这16字节的小存储区域为“分段”或“节”(paragraph)。
- 在“段加偏移”的寻址机制中，微处理器有一套用于定义各种寻址方式中段寄存器和偏移地址寄存器的组合规则。

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

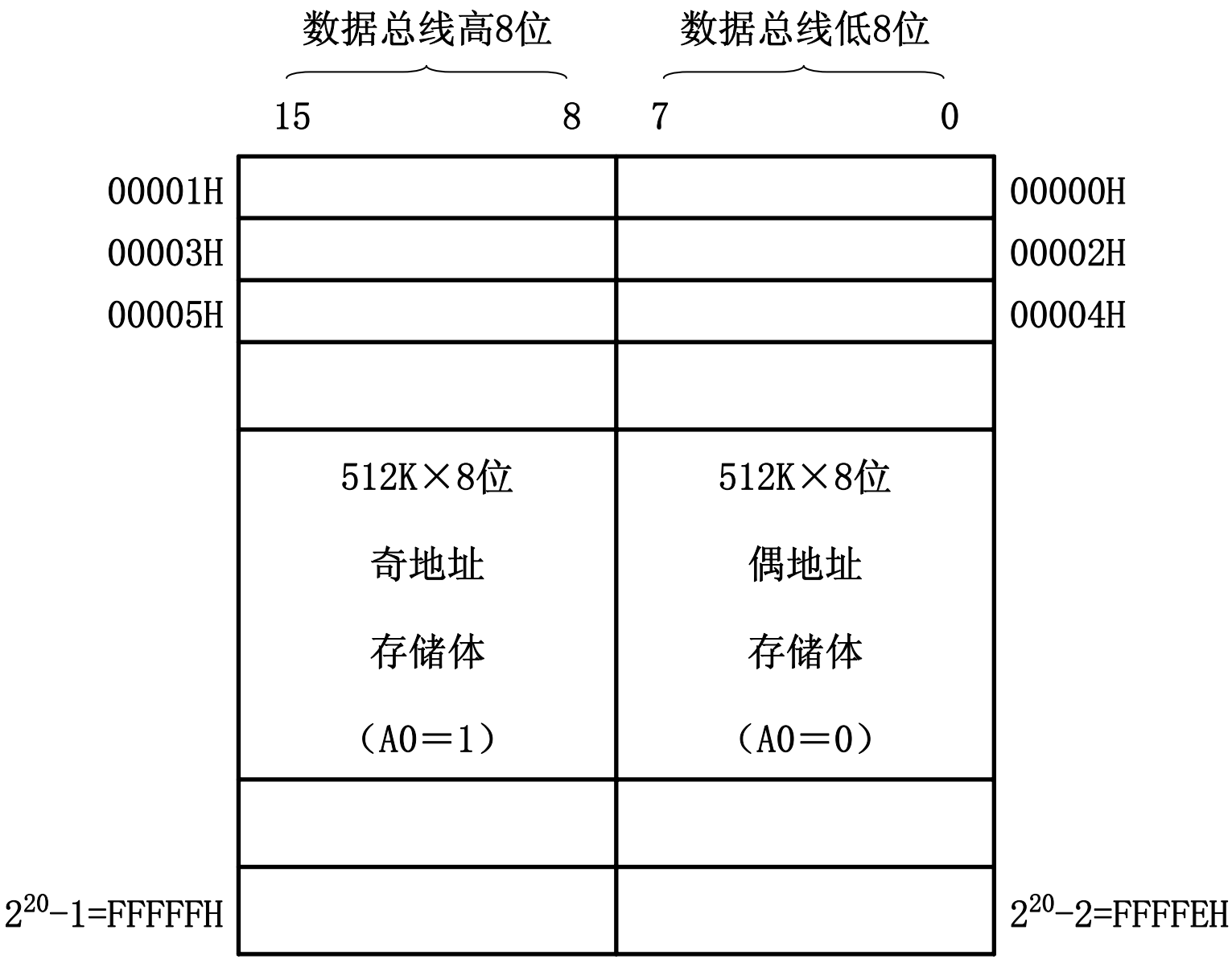


80x86默认的32位“段+偏移”寻址组合

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

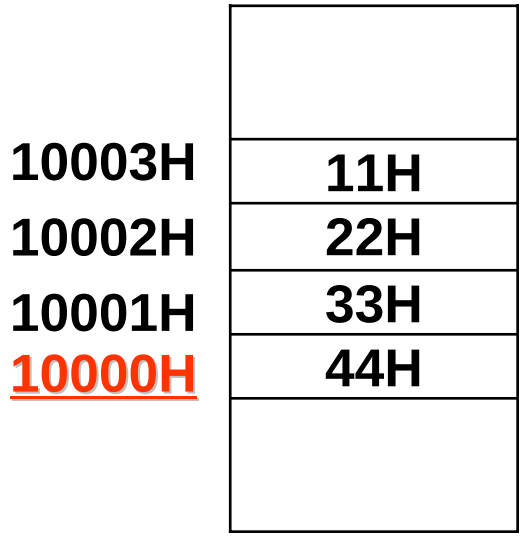
二、8086存储器的分体结构

- 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）**



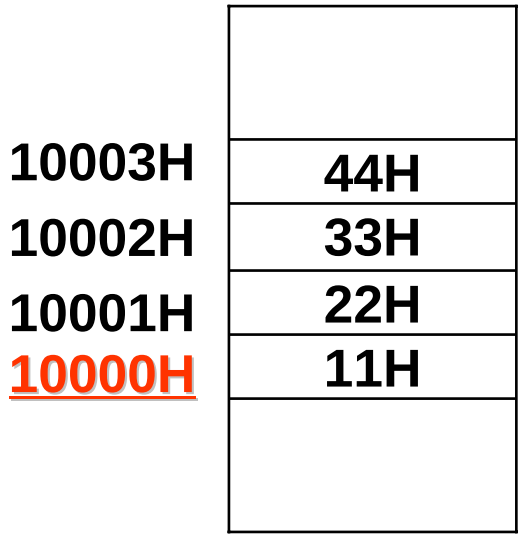
多字节存储字的两种不同存放方式

- 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**。此格式又称为**“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。此格式又称为**“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

假设：

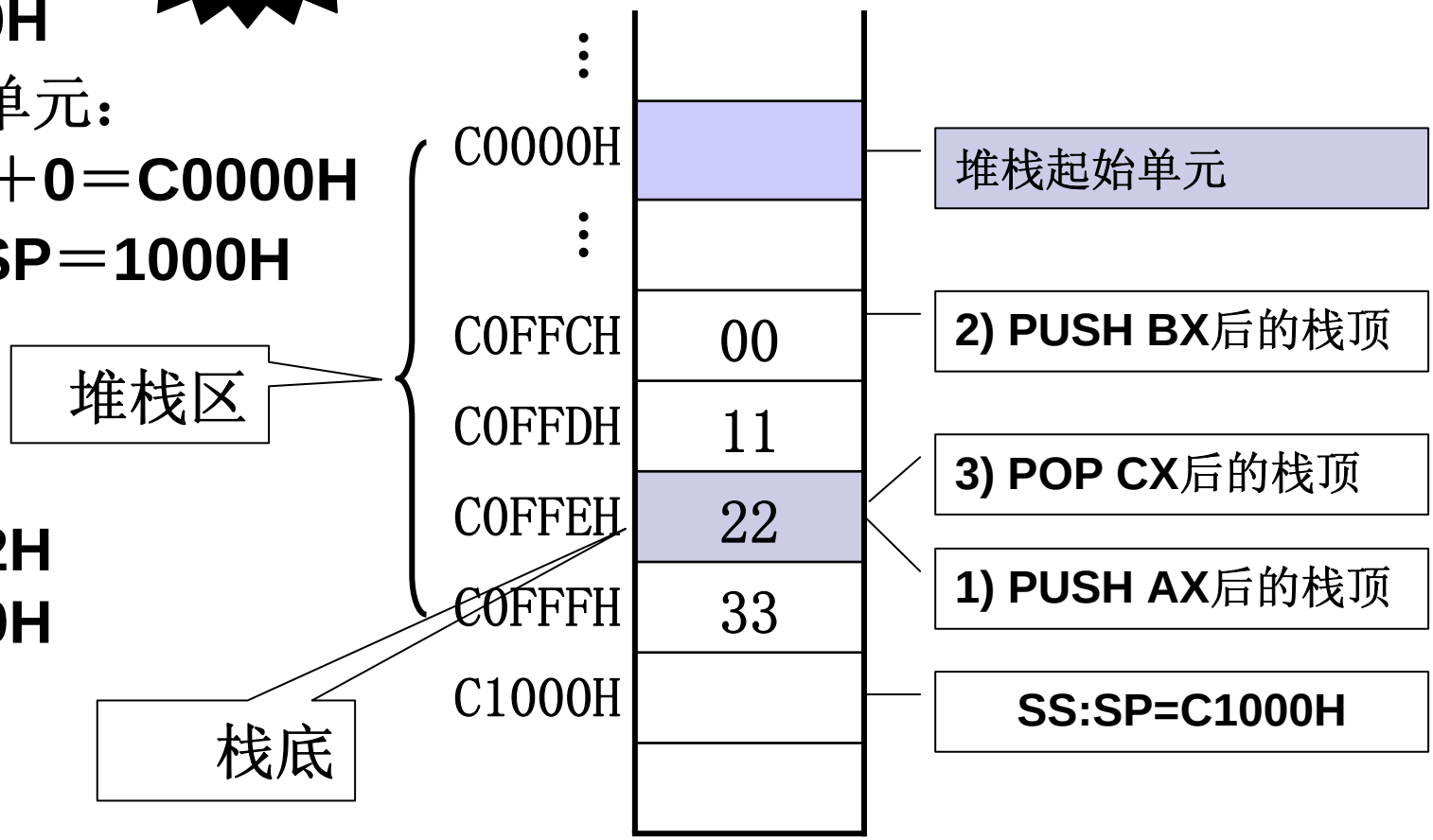
SS＝C000H

SP＝1000H

堆栈起始单元：

SS×10H＋0＝C0000H

堆栈长度**SP＝1000H**



再假设：

AX＝3322H

BX＝1100H

执行

PUSH AX

PUSH BX

POP　CX

后的堆栈区状态如图

- 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

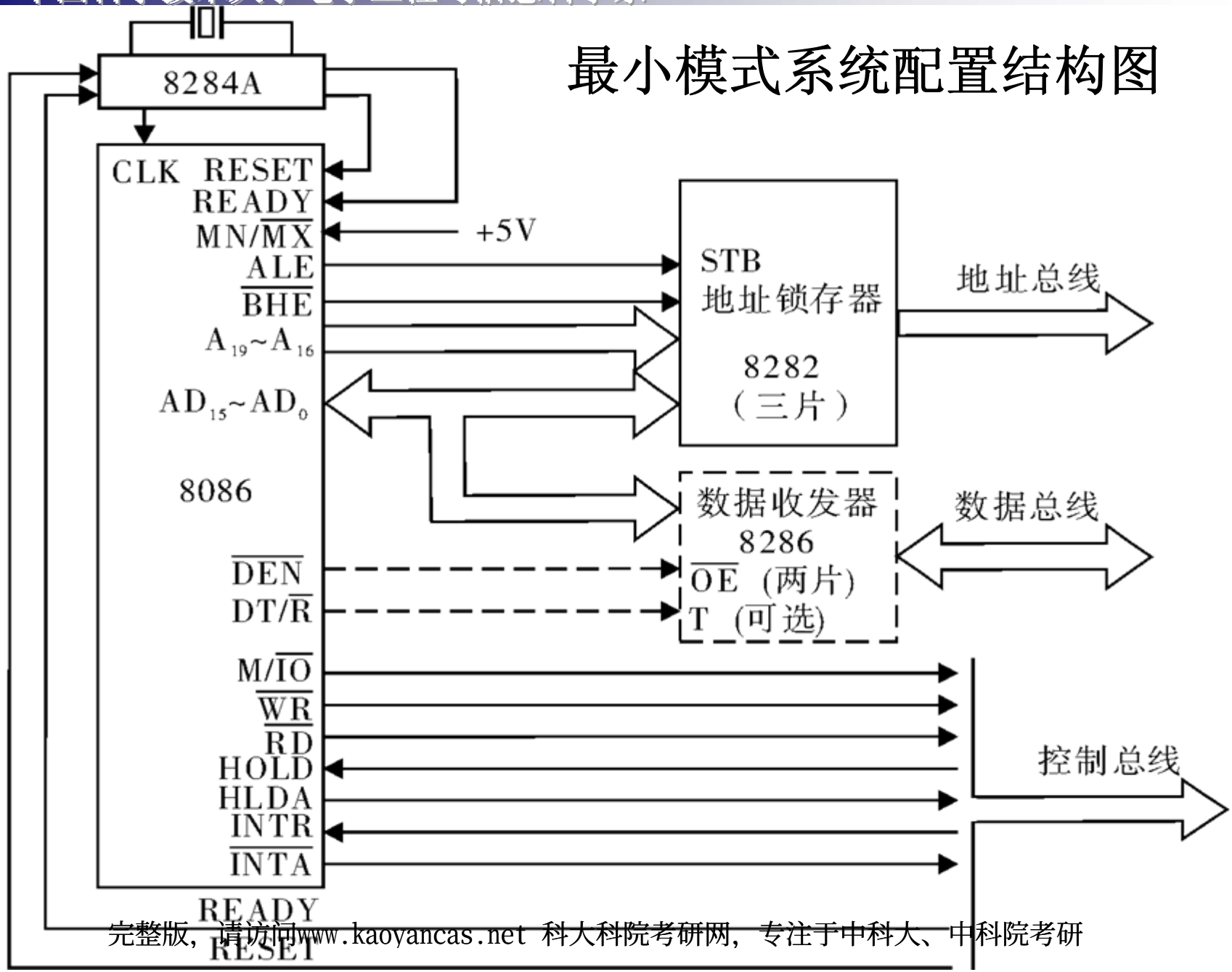
    教材P40例2-6

2-4 8086/8088系统配置

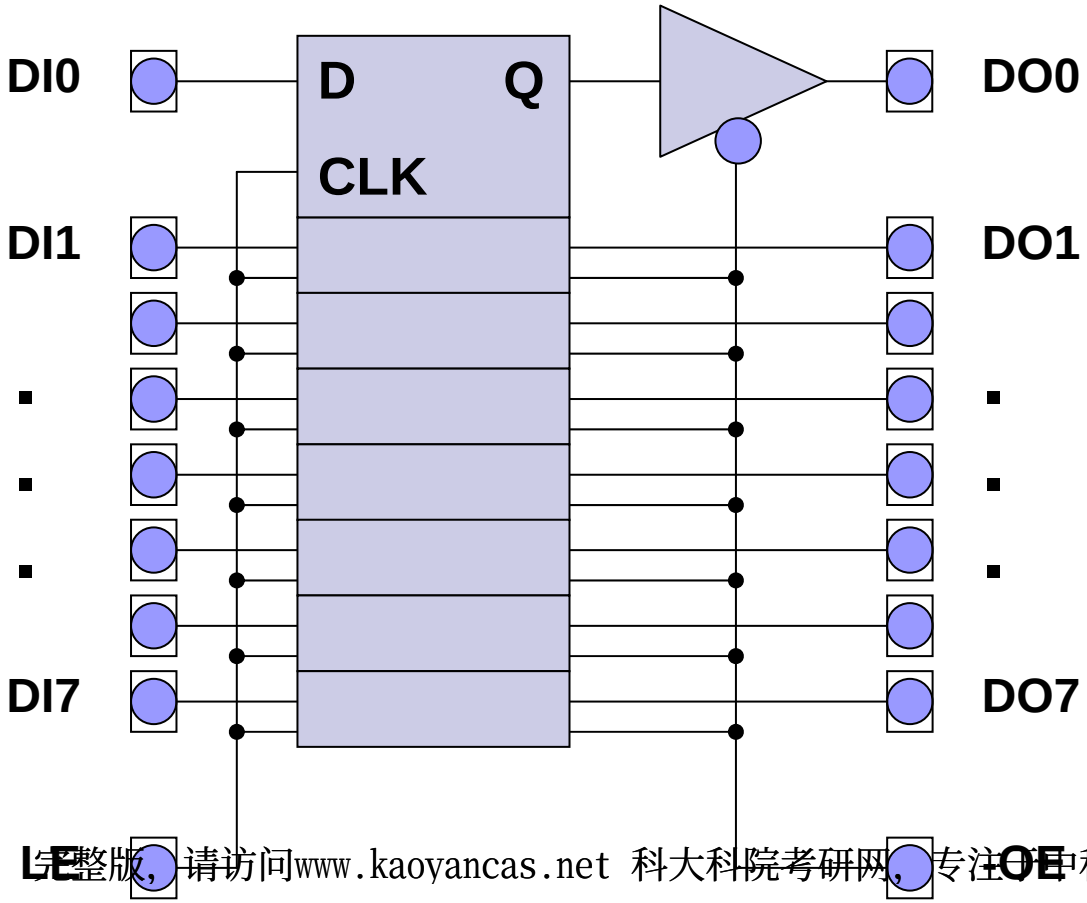
一、最小模式：MN/-MX引脚接＋5V

- 所有总线控制信号由CPU产生
- 外围电路：
  - Intel 8284—时钟发生器
  - 3片Intel 8282/8283或74xx373—地址锁存器
  - 2片Intel 8286/8287或74xx245—双向数据总线驱动器

最小模式系统配置结构图



- **Intel 8282/8283、74xx373**
  - 8单元三态缓冲/锁存器
  - Intel 8282与74xx373功能完全相同，8282输入与输出同相，8283输入输出反相。

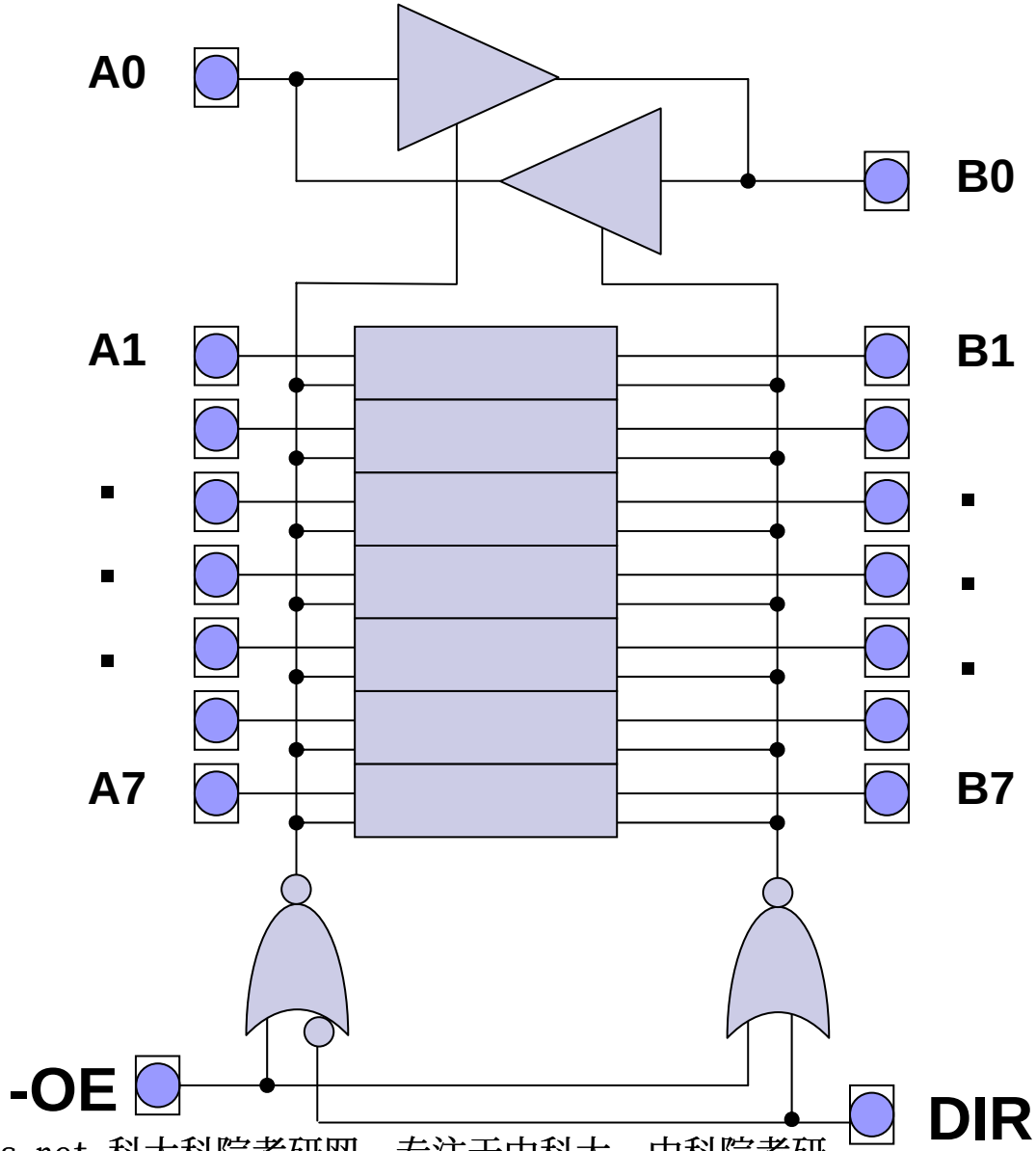


Intel 8282和74xx373内部功能结构图

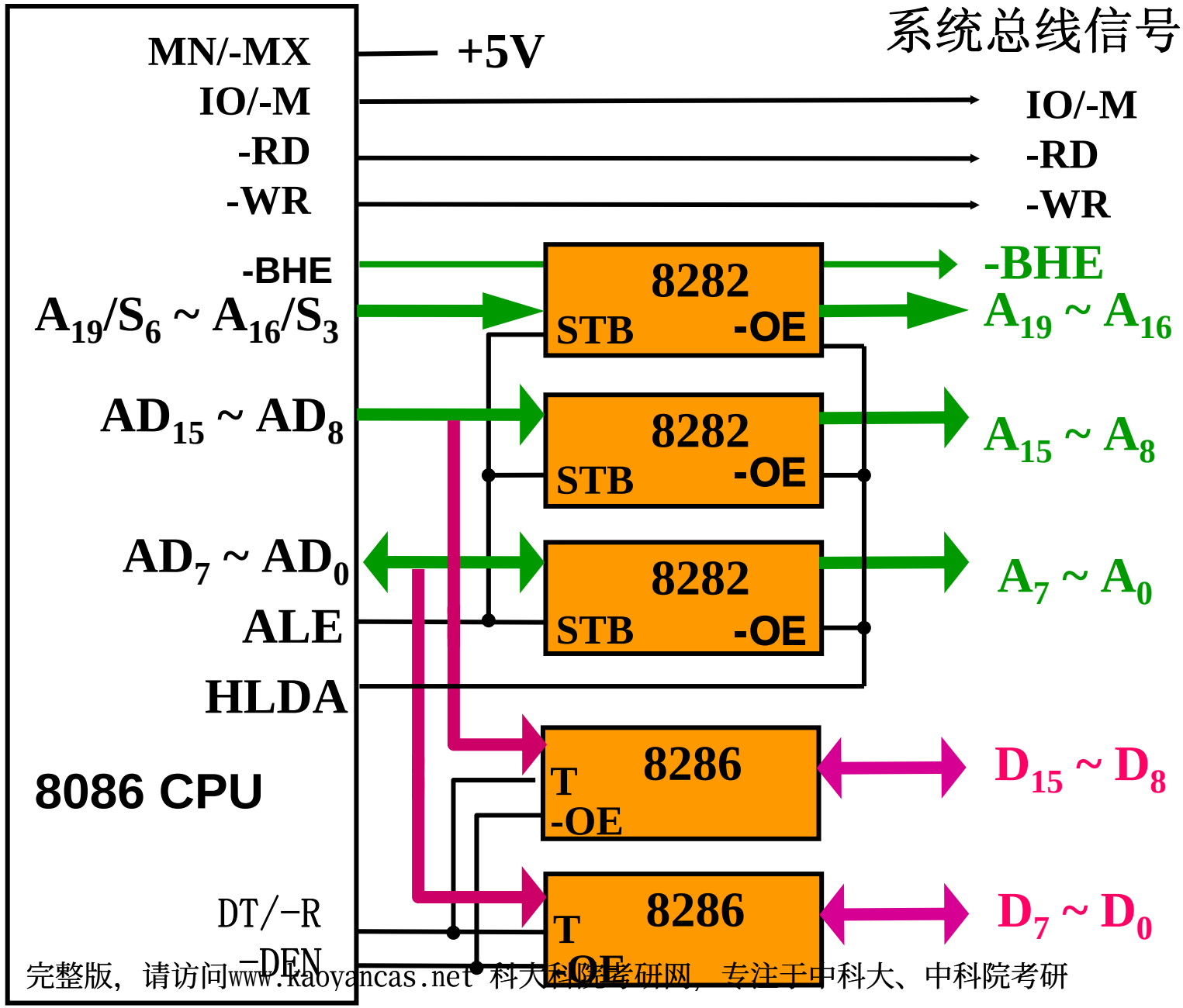
- **Intel 8286/8287、74xx245**
  - 8位三态双向总线驱动器
  - 8286与74xx245功能完全相同，8286输入与输出同相，8287输入输出反相。
  - -OE=H时，两侧三态高阻。
  - -OE=L时

    DIR=H时A－>B

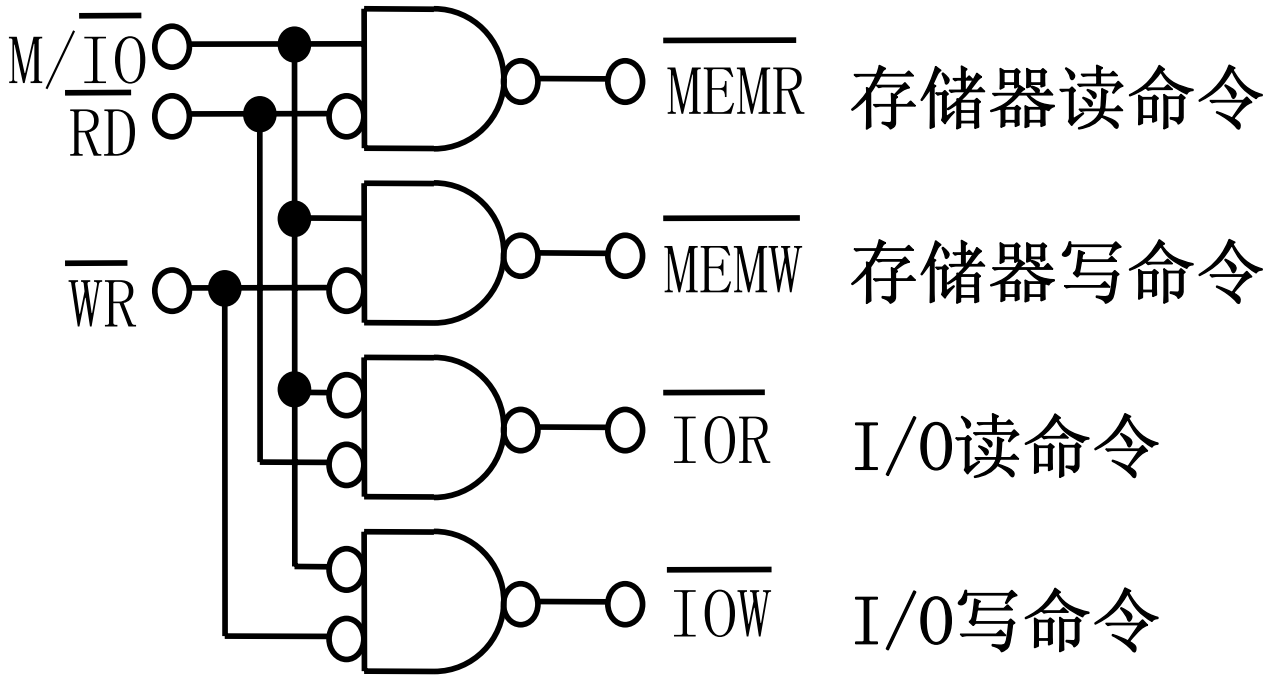
    DIR=L时B－>A



最小模式地址和数据总线的形成

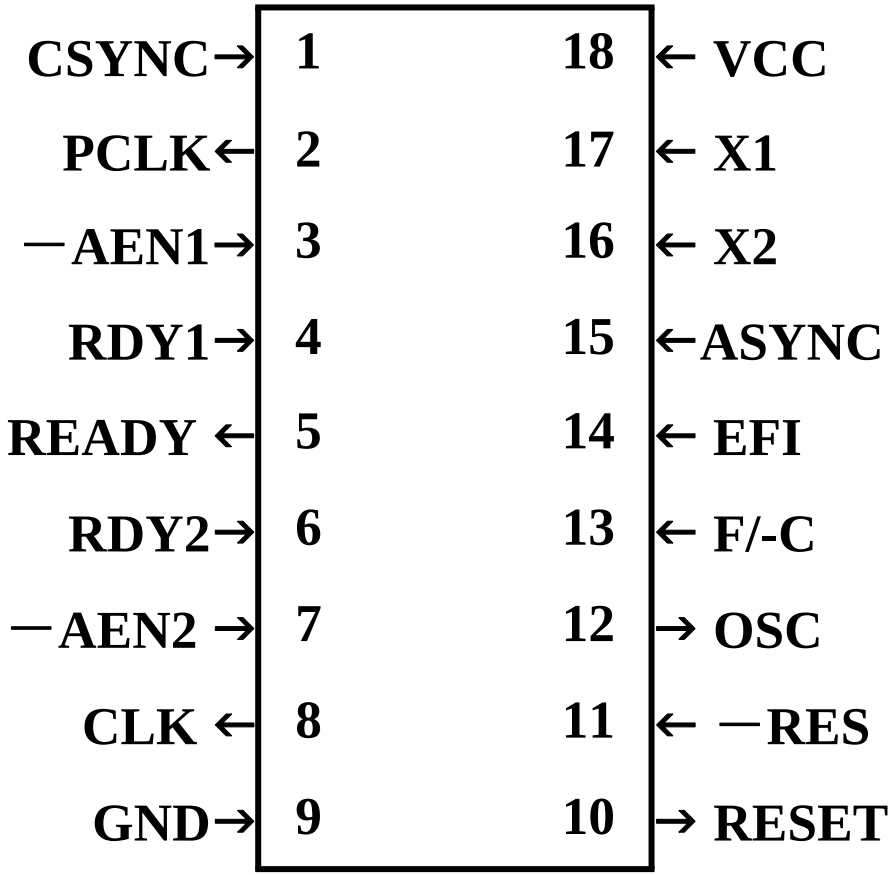


最小模式读写信号的生成

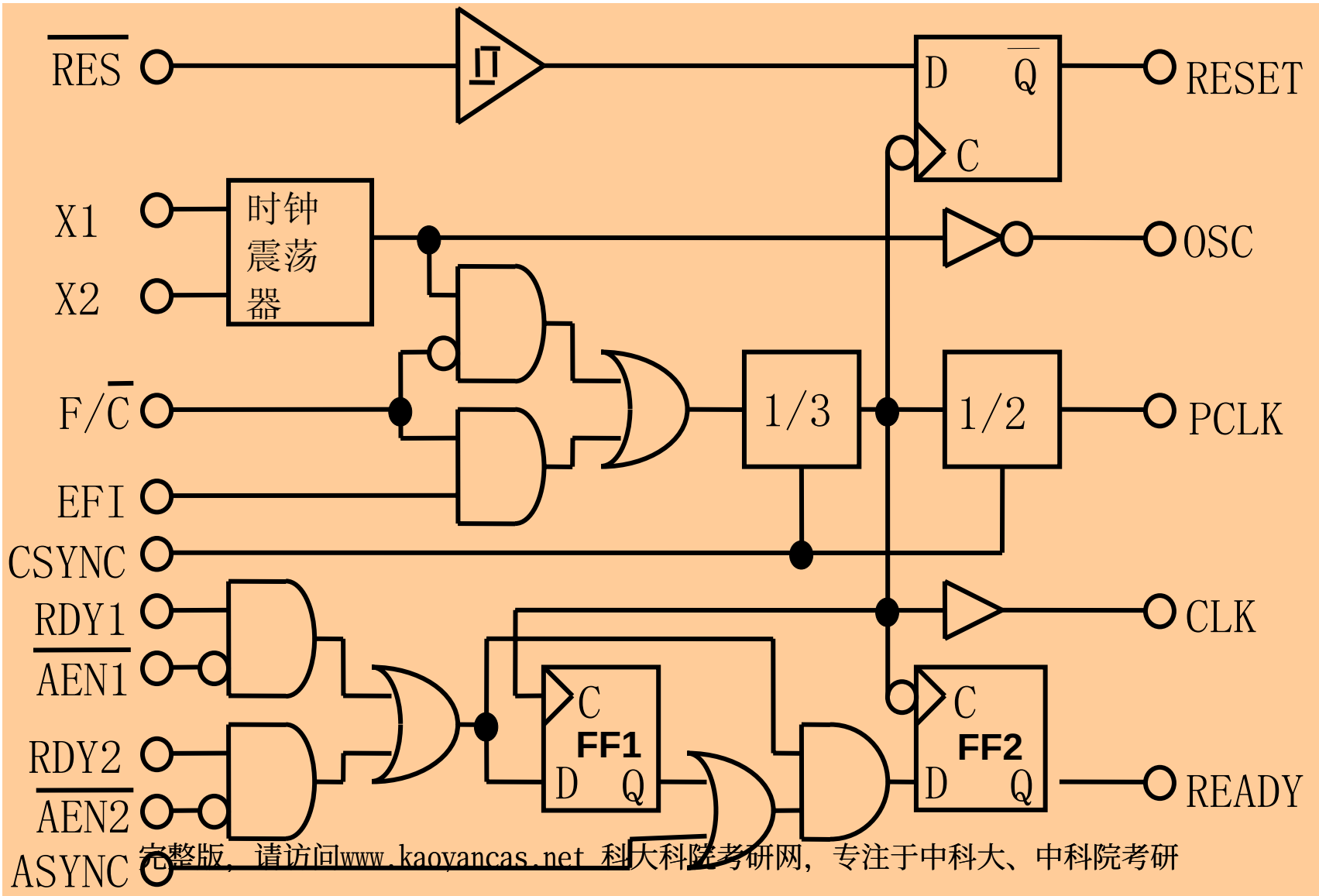


8284时钟信号发生器

- 功能：
  - 为CPU和外设提供同步时钟信号；
  - 为CPU提供准备好 (READY)信号和复位 (RESET)信号。



8284芯片内部逻辑结构图



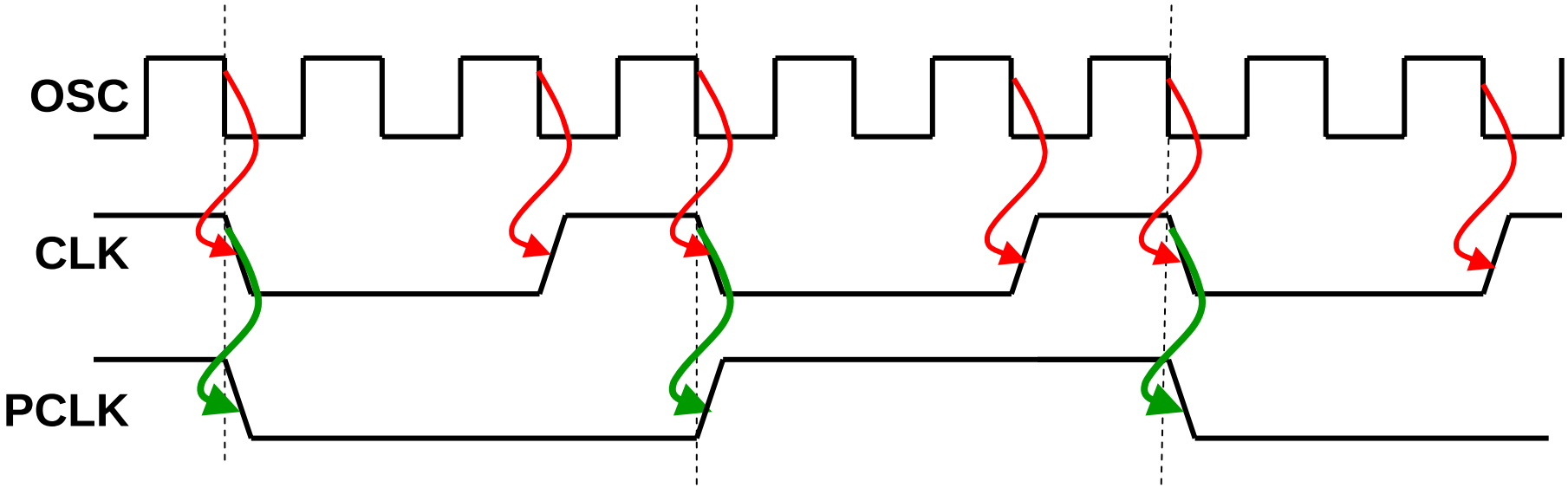
- 说明：

（1）时钟信号产生

- 内部震荡器X1和X2，外接晶体即可震荡。
- 外部频率信号（方波或矩形脉冲）输入端EFI。
- 时钟选择端F/-C， F/-C为高电平选择外时钟，为低电平选择内时钟。
- CSYNC：外部时钟的同步信号。使用内部时钟时，CSYNC接地。
- OSC：内部时钟同频信号。
- CLK：OSC的3分频信号，占空比为1/3，是8086 CPU的时钟。
- PCLK：OSC的6分频信号，占空比1/2。

8284产生的几种时钟之间的相位关系

- PC机晶振频率14.318MHz（非方波）；
- OSC频率＝晶振频率14.318MHz（方波）；
- CLK频率＝OSC频率三分频4.77MHz，占空比为1/3的矩形波；
- PCLK频率＝CLK频率二分频2.385MHz（方波）



（2）复位逻辑

- 输入-RES经斯密特触发器分频后，由系统时钟同步产生RESET信号，给计算机系统复位。

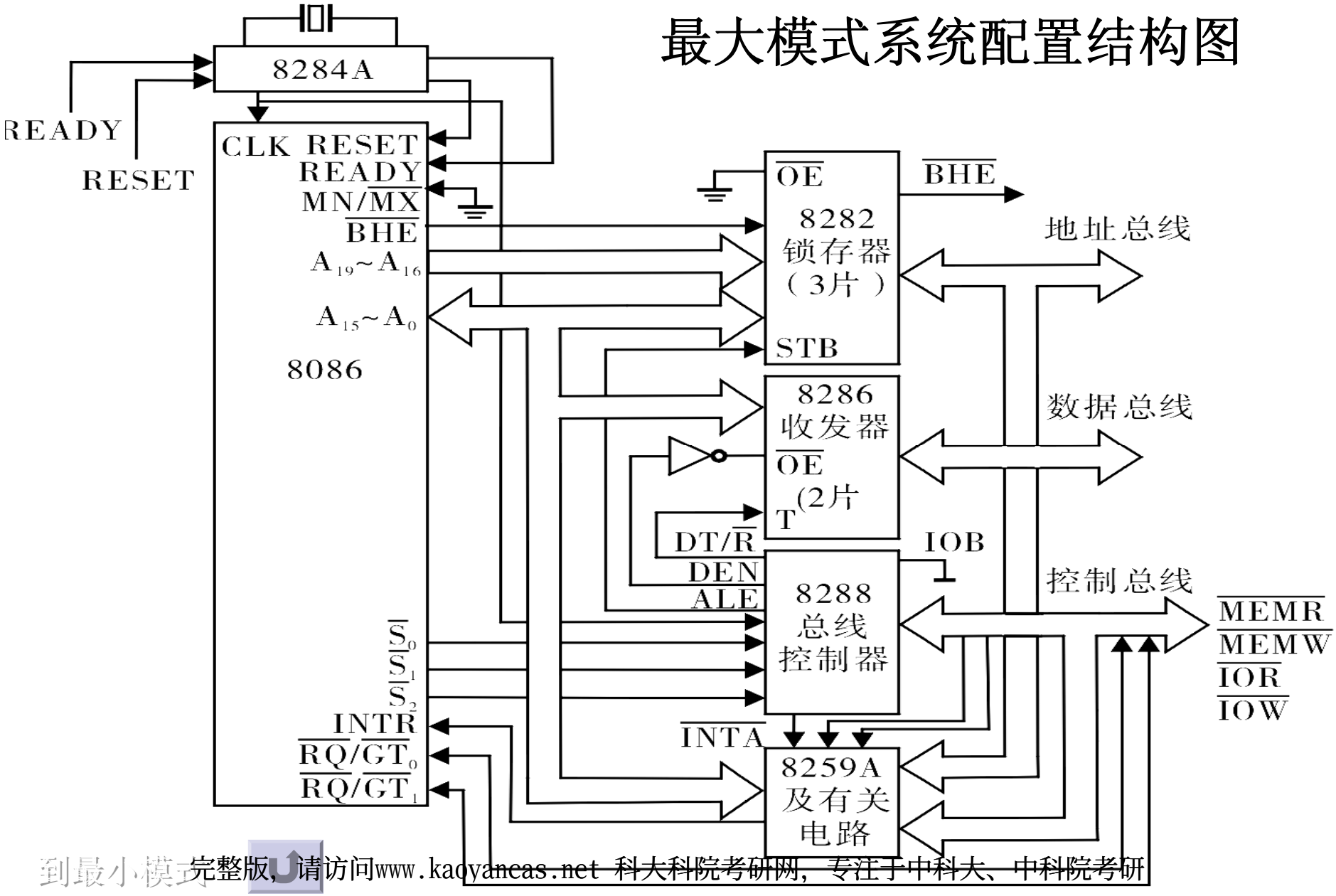
（3）准备好控制逻辑

- 准备好控制电路有两组控制信号
  - RDY1·（－AEN1）
  - RDY2·（－AEN2）
- ASYNC：异步设备同步控制。ASYNC为低时，READY信号多插入一个时钟周期。

二、最大模式：MN/-MX引脚接地（0V）

- CPU输出状态信号-S2、-S1和-S0；
- Intel 8288总线控制器根据状态信号译码产生总线控制信号；
- Intel 8289总线仲裁器根据状态信号以及总线使用权请求信号裁决总线使用权归属（PC机未使用）。
- 外围电路：Intel 8288—总线控制器

最大模式系统配置结构图



Intel 8288总线控制器

引脚功能：

1) 总线状态信号：

-S2、-S1和-S0；

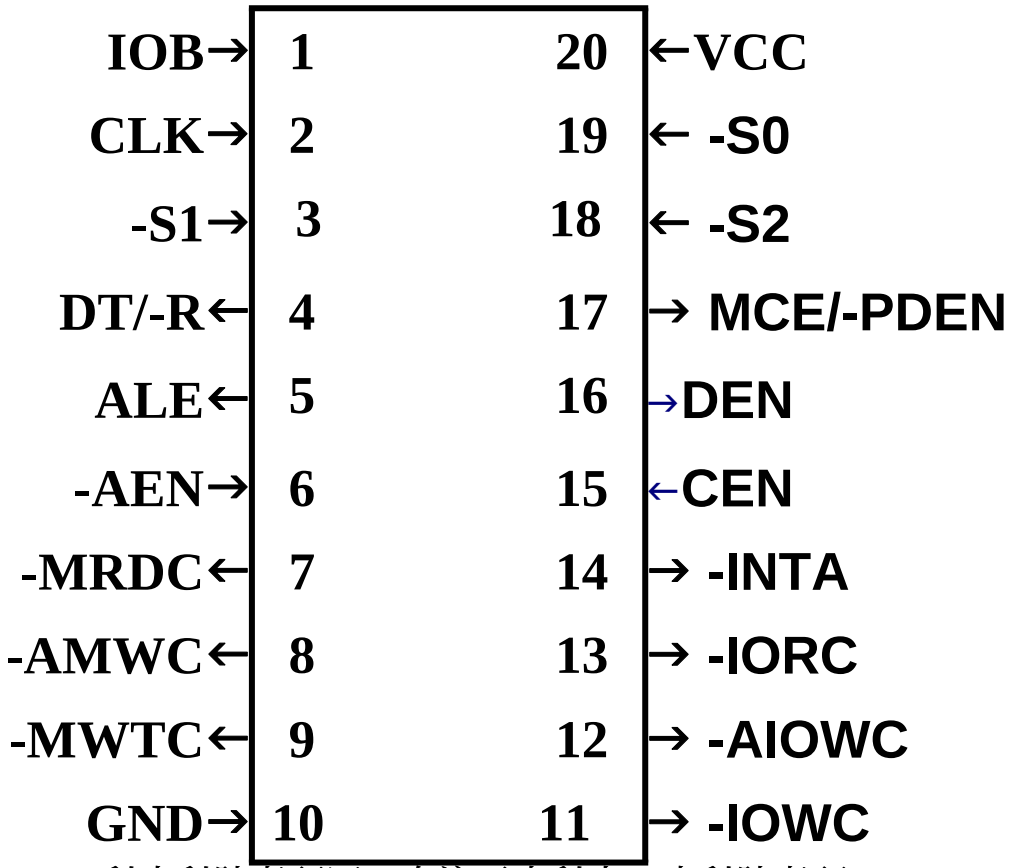
2) 控制输入信号：

CLK

-AEN

CEN

IOB



3) 总线命令信号（7条）

-INTA

-IORC

-IOWC和-AIOWC

-MRDC

-MWTC和-AMWTC

4) 总线控制信号（4条）

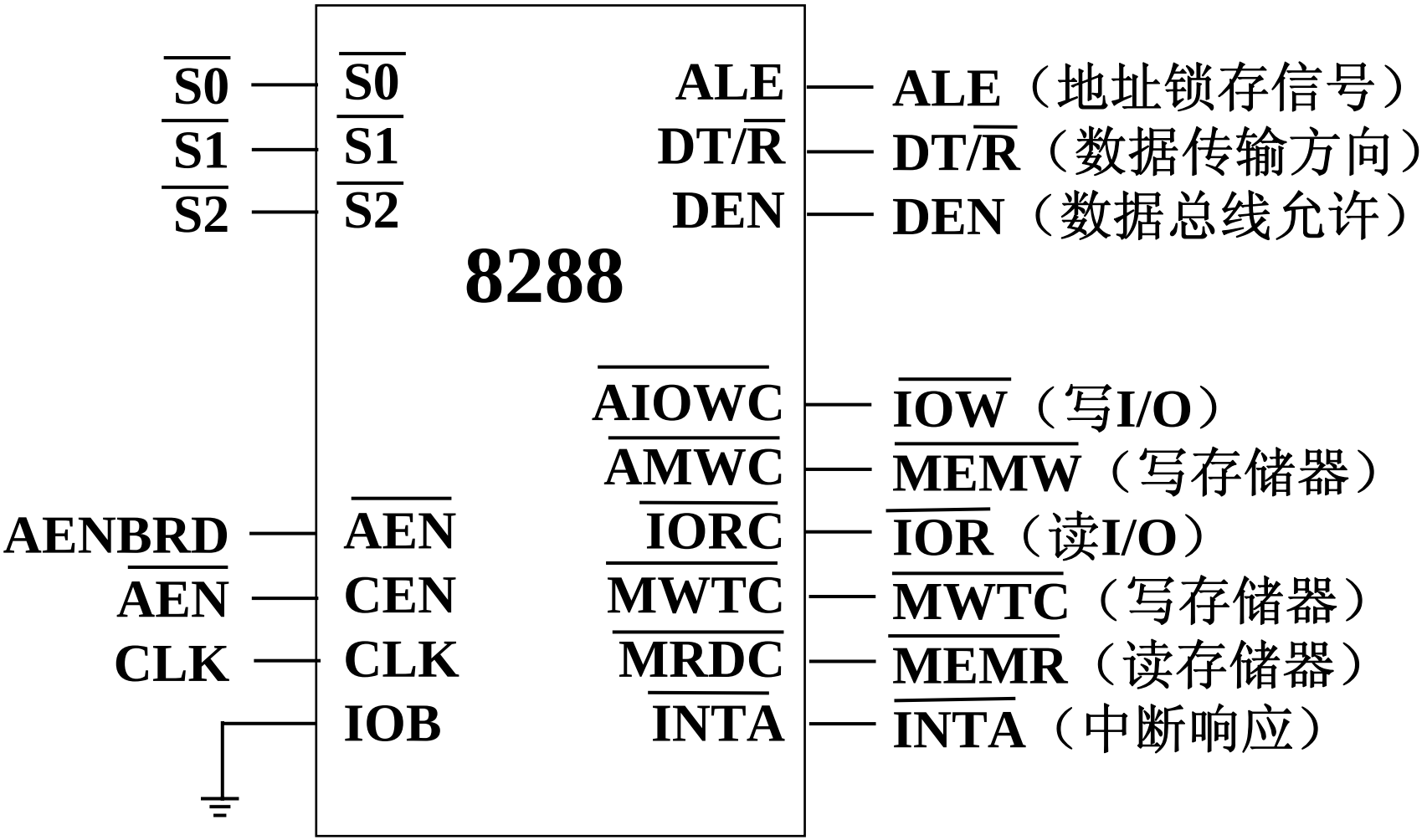
ALE

DEN

DT/-R

MCE/-PDEN

8288在PC/XT机中的连接

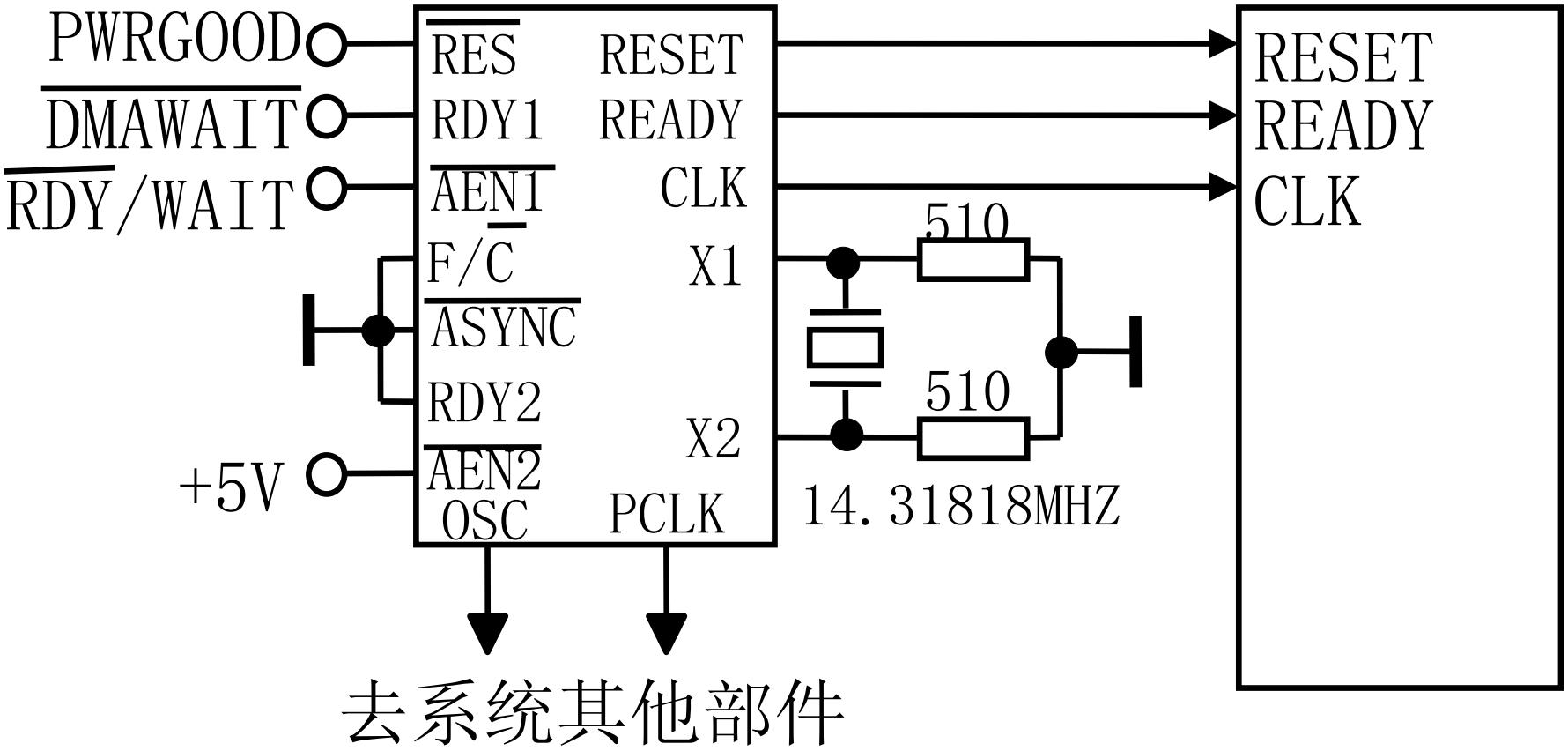


（8288工作在系统总线方式）

状态信号与总线命令和控制信号的对应关系

$\overline{S2}\,\overline{S1}\,\overline{S0}$	对应的操作	8288 产生的控制信号	相关的指令举例
000	发中断响应信号	$\overline{INTA}$	无
001	读 I/O 端口	$\overline{IROC}$	IN AL，DX
010	写 I/O 端口	$\overline{IOWC}$ 和 $\overline{AIOWC}$	OUT DX，AL
011	暂停	无	NOP
100	取指令	$\overline{MRDC}$	无
101	读内存	$\overline{MRDC}$	MOV AX，[1234H]
110	写内存	$\overline{MWTC}$ 和 $\overline{AMWC}$	MOV [DL]，AX
111	无效	无	无

最大模式下8284在系统中的连接方式



8086CPU时钟电路

2-5 8086/8088的总线时序

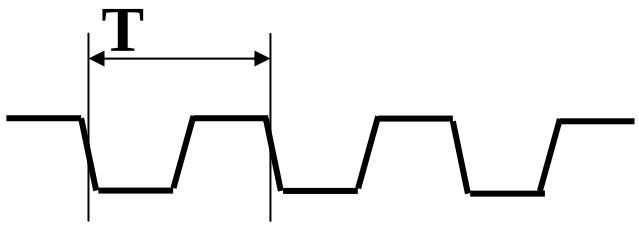
- 时序（Timing）是指信号高低电平（有效或无效）变化及相互间的时间顺序关系。
- CPU时序决定系统各部件间的同步和定时。
- 总线时序描述CPU引脚如何实现总线操作。

什么是总线操作？

- 总线操作：CPU通过总线对外的各种操作
- 8086/8088的总线操作主要有：
  - 存储器读、I/O读操作
  - 存储器写、I/O写操作
  - 中断响应操作
  - 总线请求及响应操作
  - CPU正在进行内部操作、并不进行实际对外操作的空闲状态（idle)
- 描述总线操作的微处理器时序有三级
  - 时钟周期 → 总线周期 →指令周期

几种周期的定义

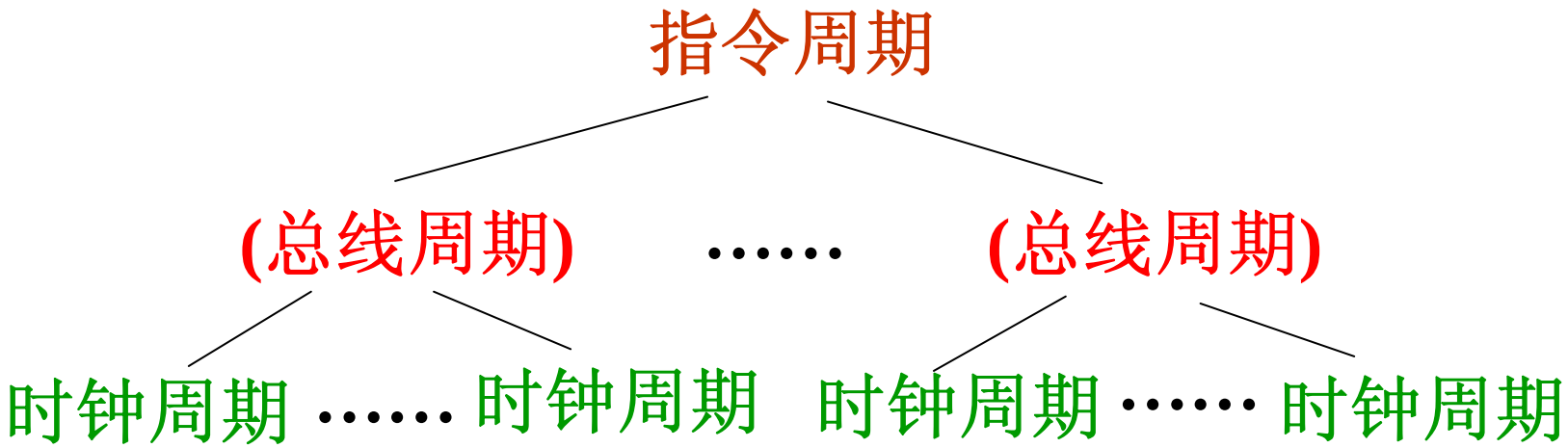
- 时钟周期（CC）：每两个相邻的时钟脉冲上升（下降）沿之间的时间间隔，也称为T状态。8086 CPU的时钟周期就是CPU输入时钟频率的倒数，是CPU操作的最小时间单位。



- 总线周期（BC）： CPU与存储器或输入/输出端口进行一次数据交换所花费的时间。

- 指令周期（IC）：指一条指令经取指、译码、读写操作数到执行完成的过程。

- 在CISC体系计算机中，一条指令周期由若干个总线周期组成，不同指令执行所需的时间不同。



- 而在单周期指令CPU（如RISC计算机）中，所有指令的执行时间相同，只包含一个时钟周期。

8086/8088的基本总线周期

- 8086/8088的基本总线周期至少4个时钟周期
  - 4个时钟周期编号为T_1、T_2、T_3和T_4
  - 总线周期中的时钟周期也被称作“T状态”
  - 时钟周期的时间长度就是时钟频率的倒数
- 当需要延长总线周期时插入等待状态Tw
- CPU进行内部操作，没有对外操作时，其引脚就处于空闲状态Ti

何时有总线周期？

- 任何指令的取指阶段都需要存储器读总线周期，读取的内容是指令代码。
- 任何一条以存储单元为源操作数的指令都将引起存储器读总线周期，任何一条以存储单元为目的操作数的指令都将引起存储器写总线周期。
- 只有执行**IN**指令才出现**I/O**读总线周期，执行**OUT**指令才出现**I/O**写总线周期。
- **CPU**响应可屏蔽中断时生成中断响应总线周期。

同步与异步时序

- 同步（动词）：正确区分动作或者数据单元的开始与结束。总线操作中如何实现时序同步是关键
- **CPU总线周期采用（半）同步时序：**
  - 各部件都以系统时钟信号为基准
  - 当某些部件与**CPU**的速度有差异时，快速部件（**CPU**）插入等待状态等待慢速部件（**I/O**或存储器）
- 主机的外设接口与外设之间常采用异步时序，它们通过应答联络信号实现同步操作。

一、系统的复位与启动

- RESET变成高电平（应至少维持4个时钟周期）时，8086 CPU结束现行操作，进入复位，除了CS＝FFFFH以外，其它寄存器均为0000H，指令队列清空。
- RESET回到低电平后，CPU脱离复位从（CS:IP）开始执行指令（FFFF0H）。
- 由于标志寄存器被清零，IF＝0不响应外部可屏蔽中断，所以应在系统初始化程序结束前安排一条STI指令，使IF＝1。
- RESET信号有效引起的8086 CPU复位时序参见P50图2-22。

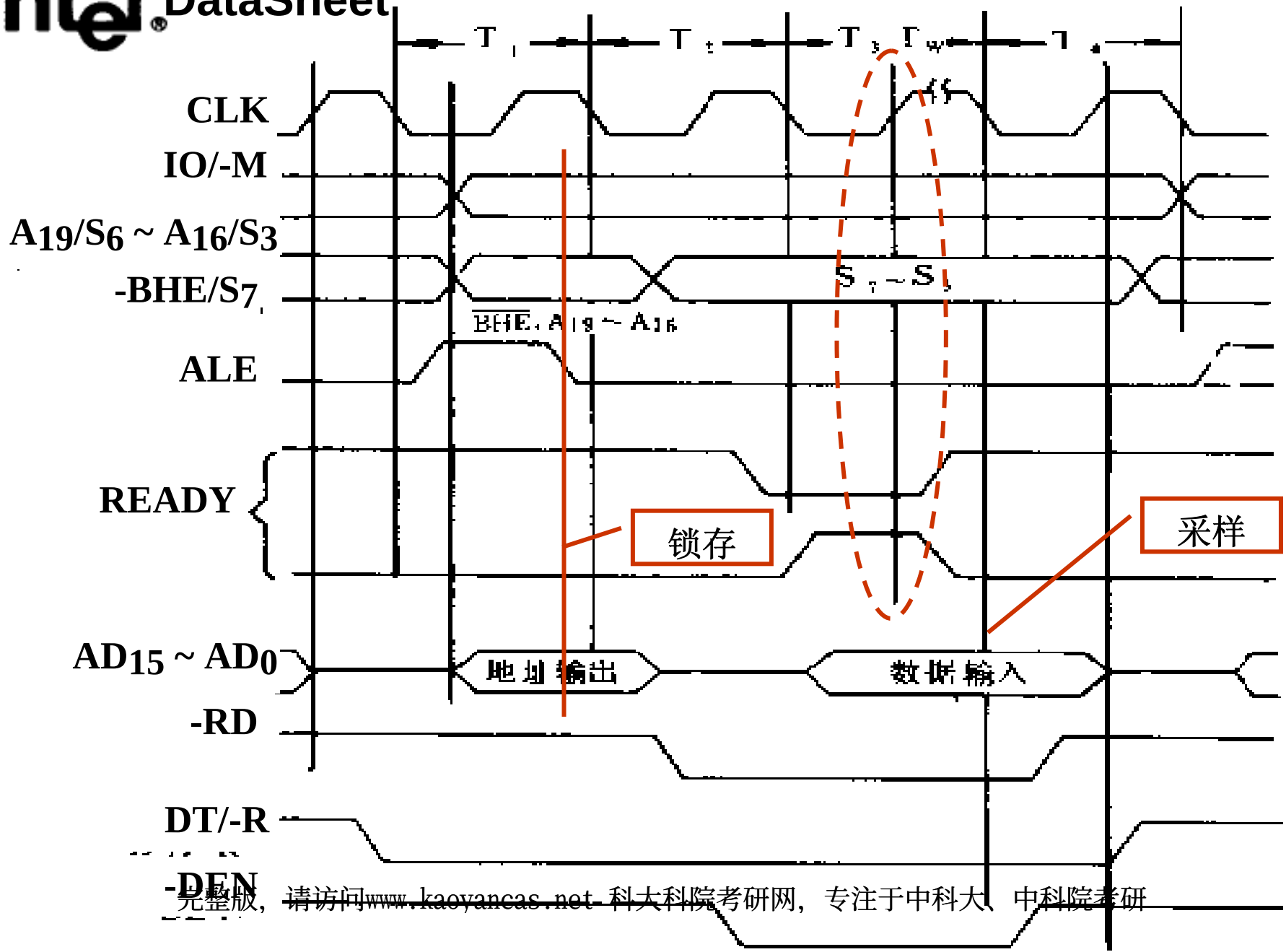
二、8086最小模式下的总线操作

- 8086/8088共有6种时钟状态：Ti、T1～T4和等待状态Tw。
- 一个总线周期包括T1~T4，还可能在T3后面插入1个或多个Tw。
- 当CPU不使用总线期间，或者在CPU两次总线操作之间的空闲期间，执行的是空闲状态Ti。

1、最小模式下读周期

- **T1状态：**
  - **M/-IO、地址信号、ALE、-BHE和DT/-R有效**
- **T2状态：**
  - **总线上的地址信号撤销，A19～A16输出状态，A15～A0进入三态，-RD和-DEN有效。**
- **T3状态**
  - **T3周期上升沿（注意：是在T3周期的2/3处，不是在T3的开始时刻）CPU采样READY，若无效（低电平），当前时钟周期后面插入1个Tw周期。**

intel DataSheet

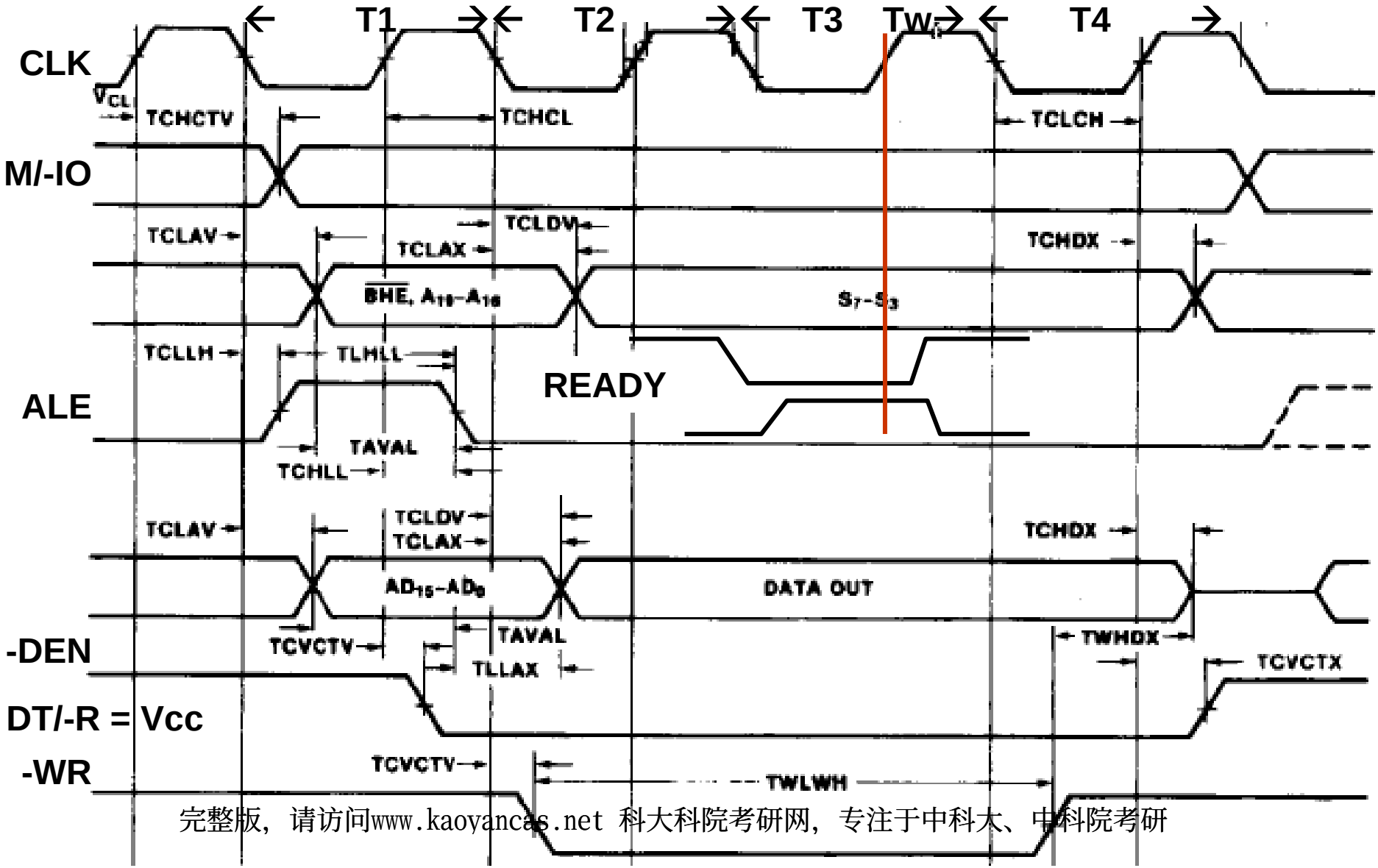


- 若READY有效，－RD信号将地址信号所选中的存储单元或者IO端口的数据读出至D15～D0总线上。

- **Tw状态**
  - 重复T3状态的动作，延长CPU的访问周期。
- **T4状态**
  - CPU在T4开始时刻采样D15～D0总线上的数据，T4后半段结束各个总线命令和控制信号无效，总线上的数据也消失，本次总线周期结束。

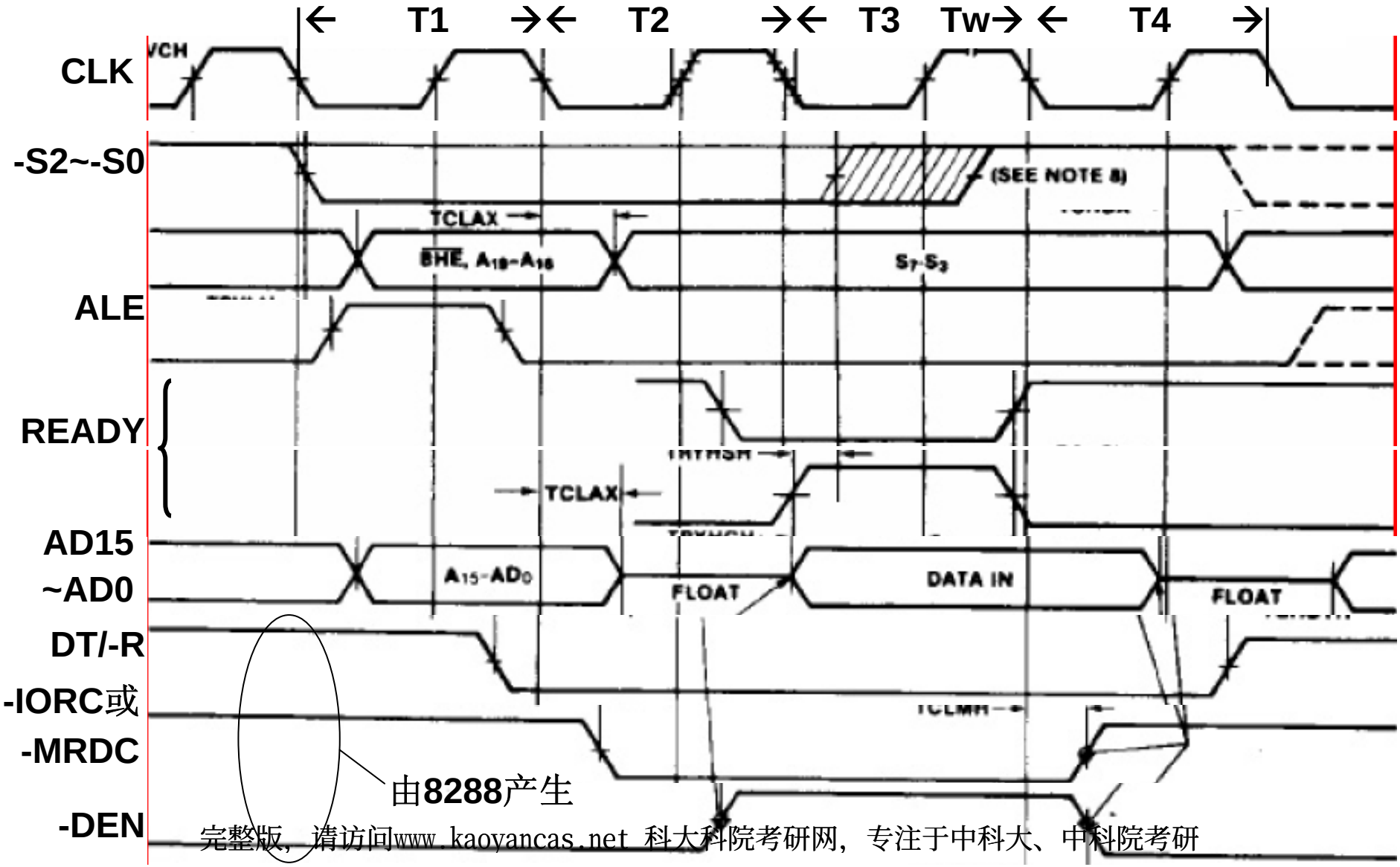
2、最小模式下写周期

intel DataSheet



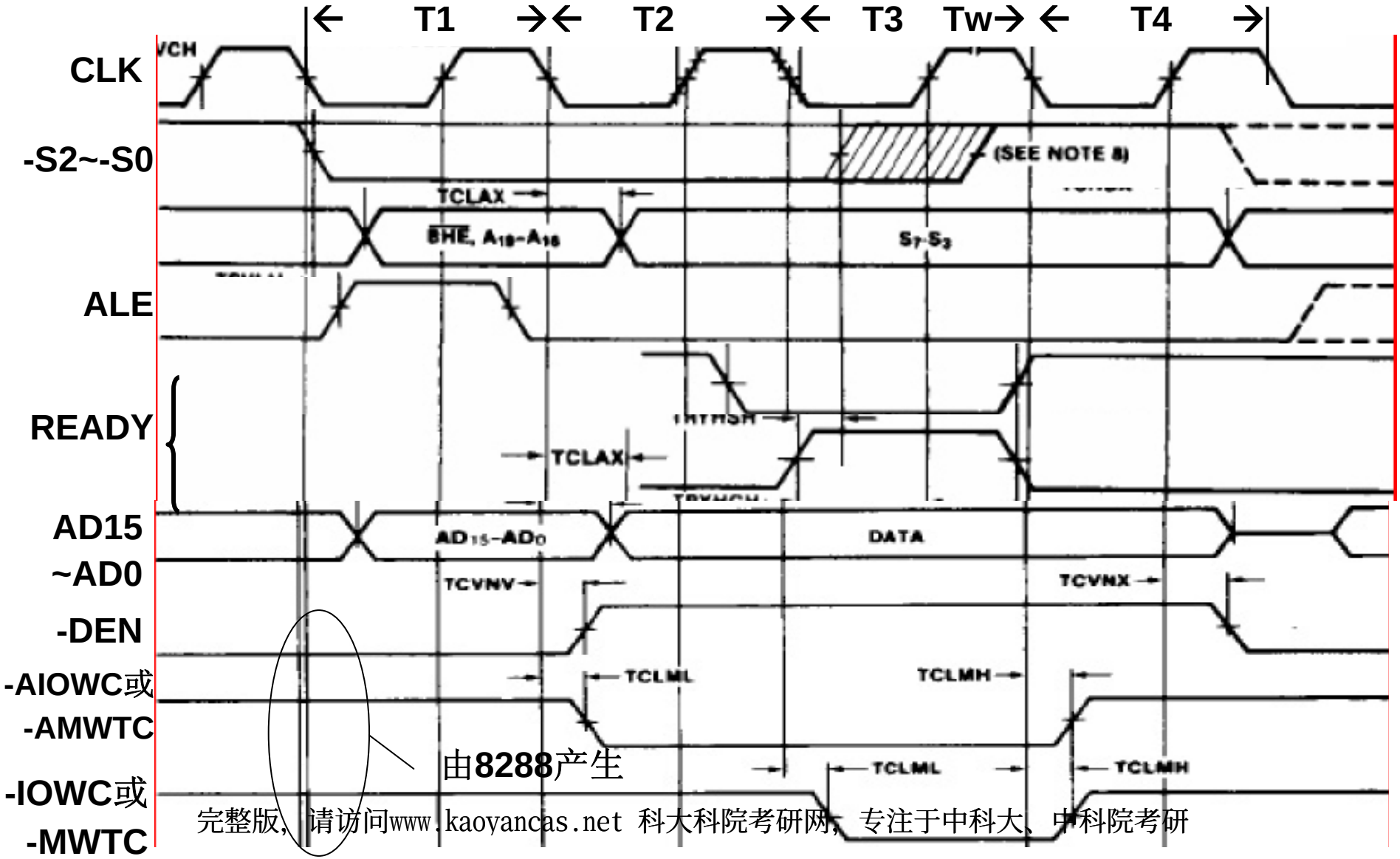
3、最大模式下读周期

intel DataSheet



4、最大模式下写周期

intel DataSheet



2-7 32位微处理器简介

- Intel公司的32位CPU分为三大类：
  1）从80286到Pentium，称为80x86处理器
  2）从Pentium Pro到Pentium III，称为IA－32结构。
  3）基于Netburst结构的Pentium和至强(Xeon)
- Intel公司每一代新的CPU的指令系统和IO端口都与以往产品兼容，在早期CPU上开发的程序在新一代的CPU上可以直接运行。

Intel 80286微处理器简介

1．80286的主要特性

（1）增加地址线，使内存容量提高。

（2）具有两种地址方式：实地址方式和保护虚地址方式。

（3）使用虚拟内存。

（4）寻址方式更加丰富 （24种）

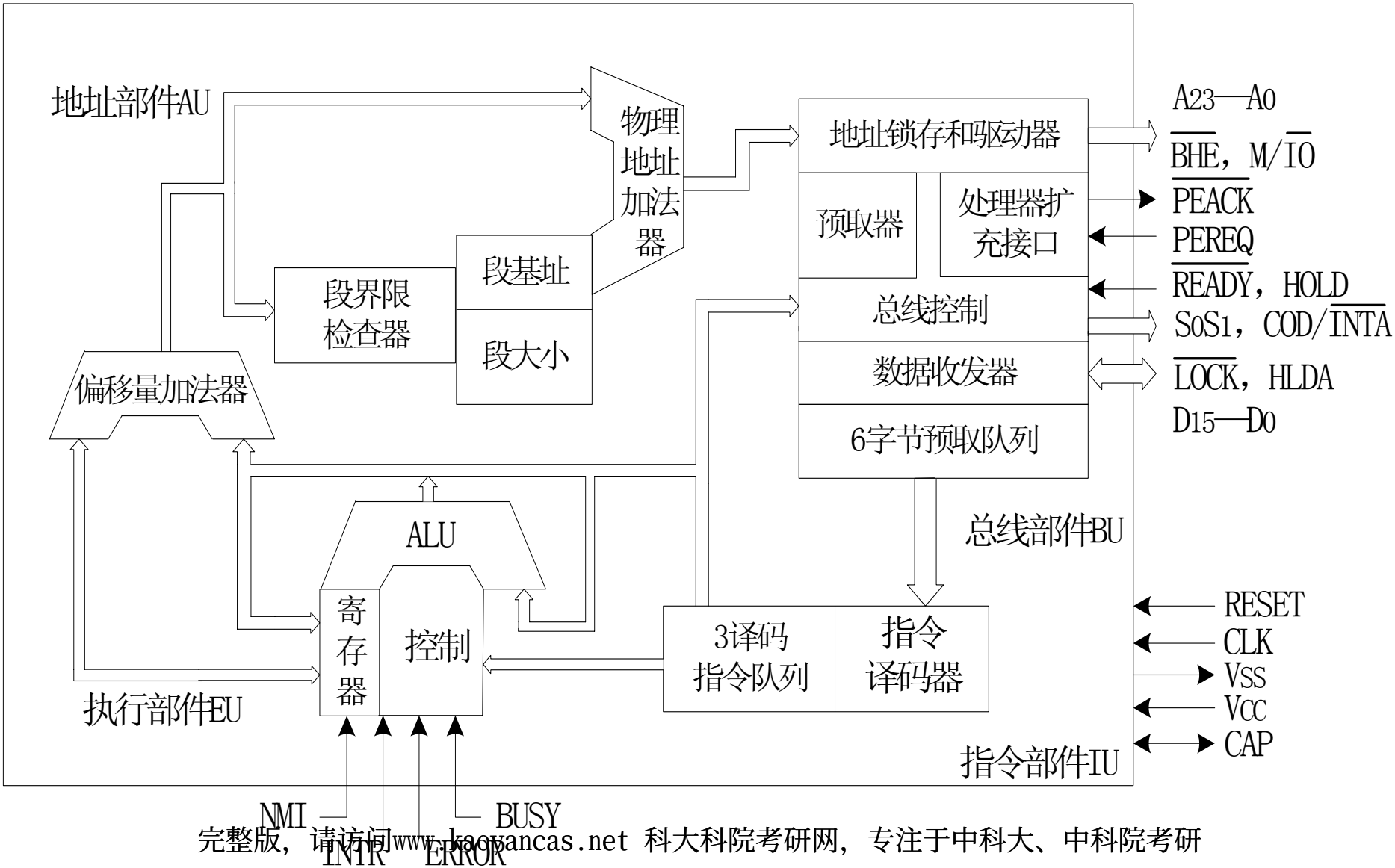
（5）可以同时运行多个任务 。

（6）三种类型中断 ：硬件中断、软件中断和异常中断。

（7）增加了高级类指令、执行环境操作类指令和保护类指令 。

（8）时钟频率提高

2．80286内部结构



3．80286的地址方式

- 80286访问存储器时，有两种方式即实地址方式和虚地址保护方式。

1) 实地址方式：80286加电后即进入实地址方式。在实地址方式下，80286与8086在目标码一级是向上兼容的，它兼容了8086的全部功能，8086的汇编语言源程序可以不做任何修改在80286上运行。

2) 虚地址保护方式：此方式是集实地址方式、存储器管理、对于虚拟存储器的支持和对地址空间的保护为一体而建立起来的一种特殊工作方式，使80286能支持多用户、多任务系统。

Intel 80386微处理器简介

Intel于1985年推出的第一个32位微处理机。

1．80386的主要特性

（1）灵活的32位微处理器，提供32位的指令。

（2）提供32位外部总线接口，最大数据传输速率为32MB。

（3）具有片内集成的存储器管理部件MMU，可支持虚拟存储和特权保护 。

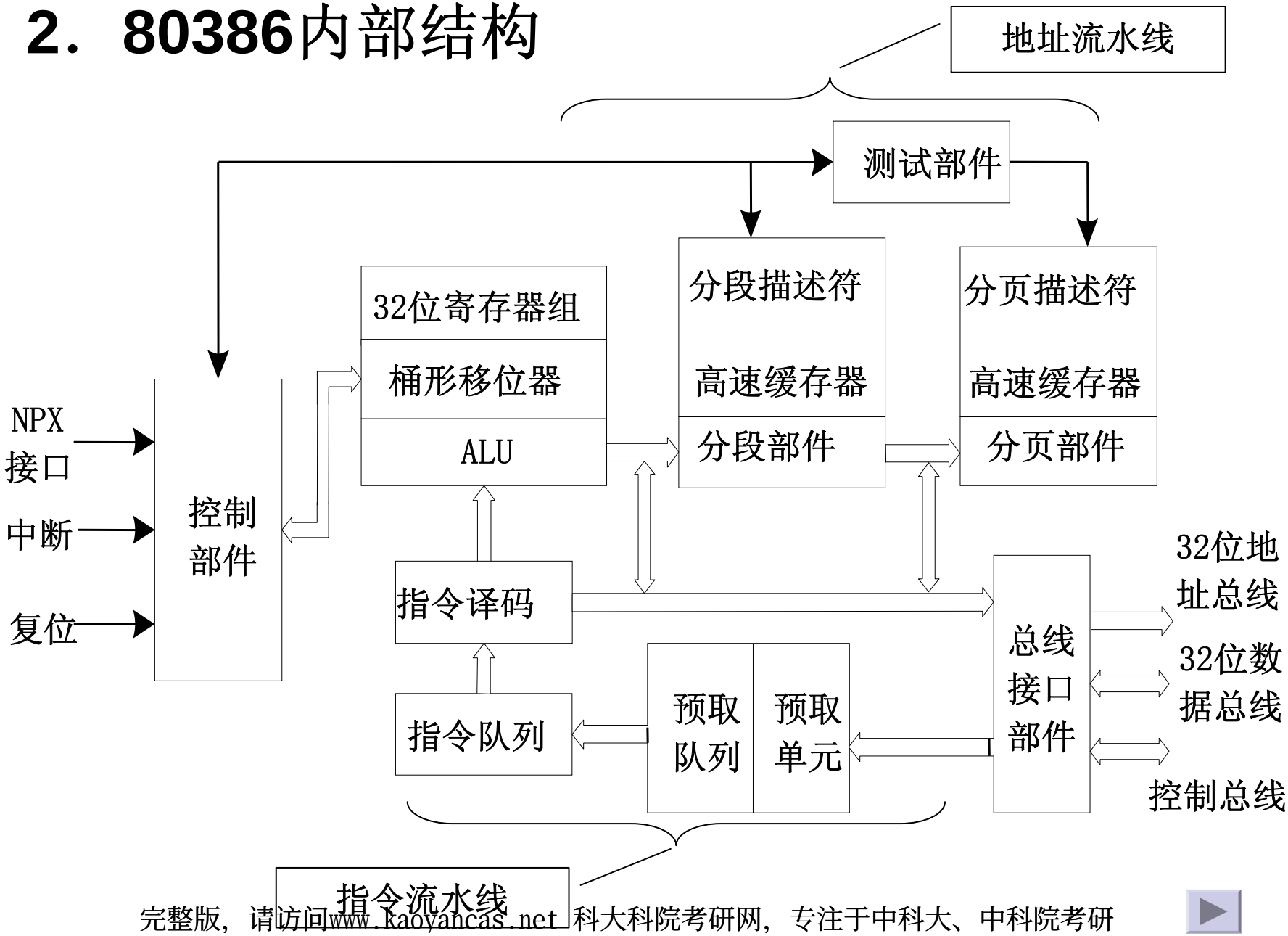
（4）具有实地址方式、保护方式和虚拟8086方式。

（5）具有极大的寻址空间。

（6）通过配用数值协处理器可支持高速数值处理。

（7）在目标码一级与8086、80286芯片完全兼容。

2．80386内部结构

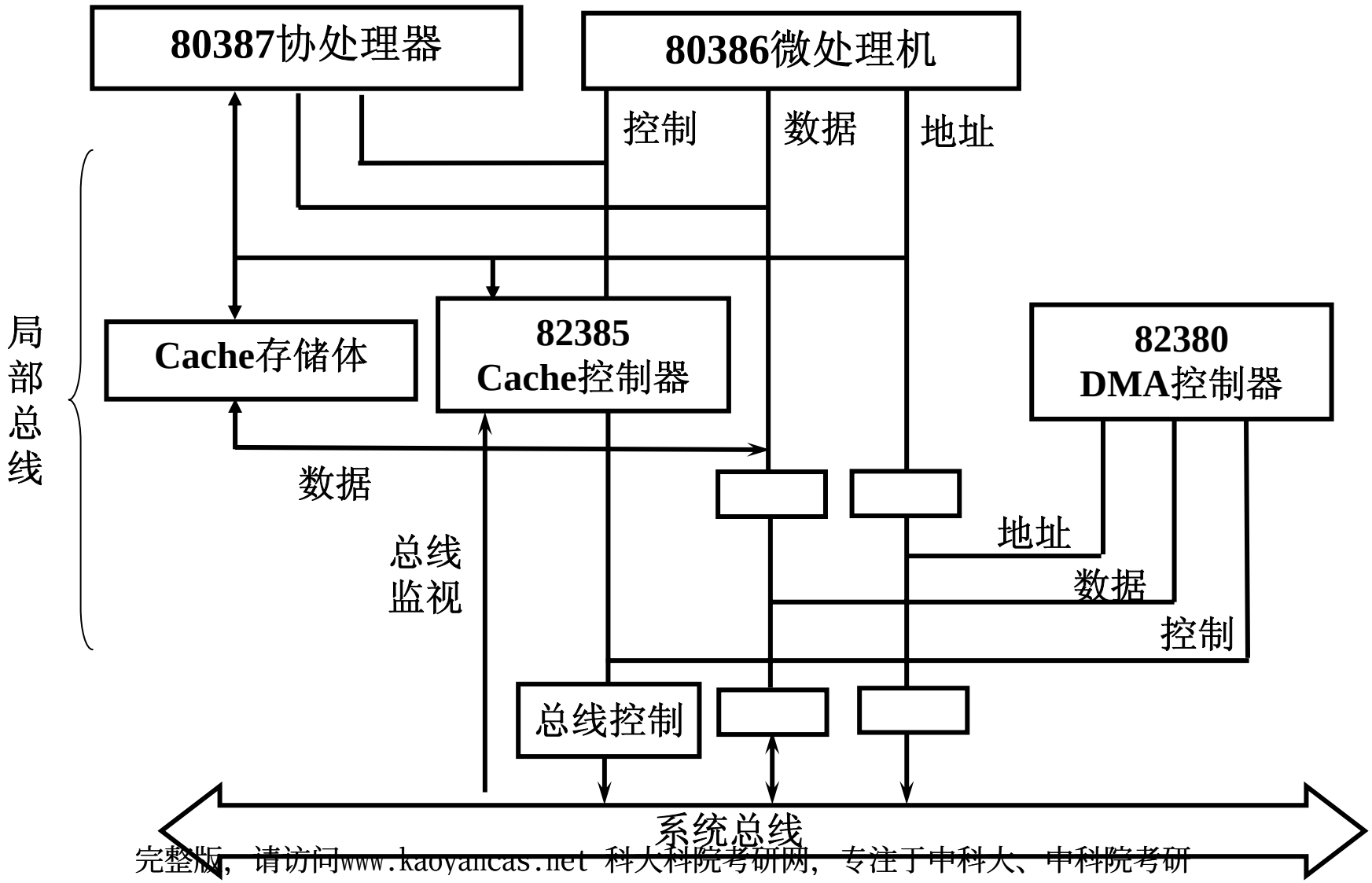


3．80386的寄存器结构

80386中共有34个寄存器，分为以下7类：

a) 通用寄存器（General-Purpose Register）
b) 段寄存器（Segment Register）
c) 指令指针和标志寄存器（IP & Flag Register）
d) 状态和控制寄存器（Status and Control Register）
e) 系统地址寄存器（System Address Register）
f) 调试寄存器（Debug Register）
g) 测试寄存器（Test Register）

4. 80386微机系统的核心



- 协处理器80387作用：加速浮点运算操作。
- Cache控制器82385作用：对容量为32K字节的高速缓冲存储器Cache实施控制，并带有总线监视（Bus Watching）这一高级特征。
- DMA控制器82380作用：集若干系统功能于一体，其中包括动态RAM刷新、中断与计时器和32位直接存储器存取控制。
- 80386的高性能系统核心配备有两种不同的总线：
  - 一是80386的系统总线；
  - 二是80386的局部总线。

5．80386的工作方式

（1）实地址方式：

- 系统启动后，80386自动进入实地址方式。此方式下，采用类似于8086的体系结构。

（2）保护方式：

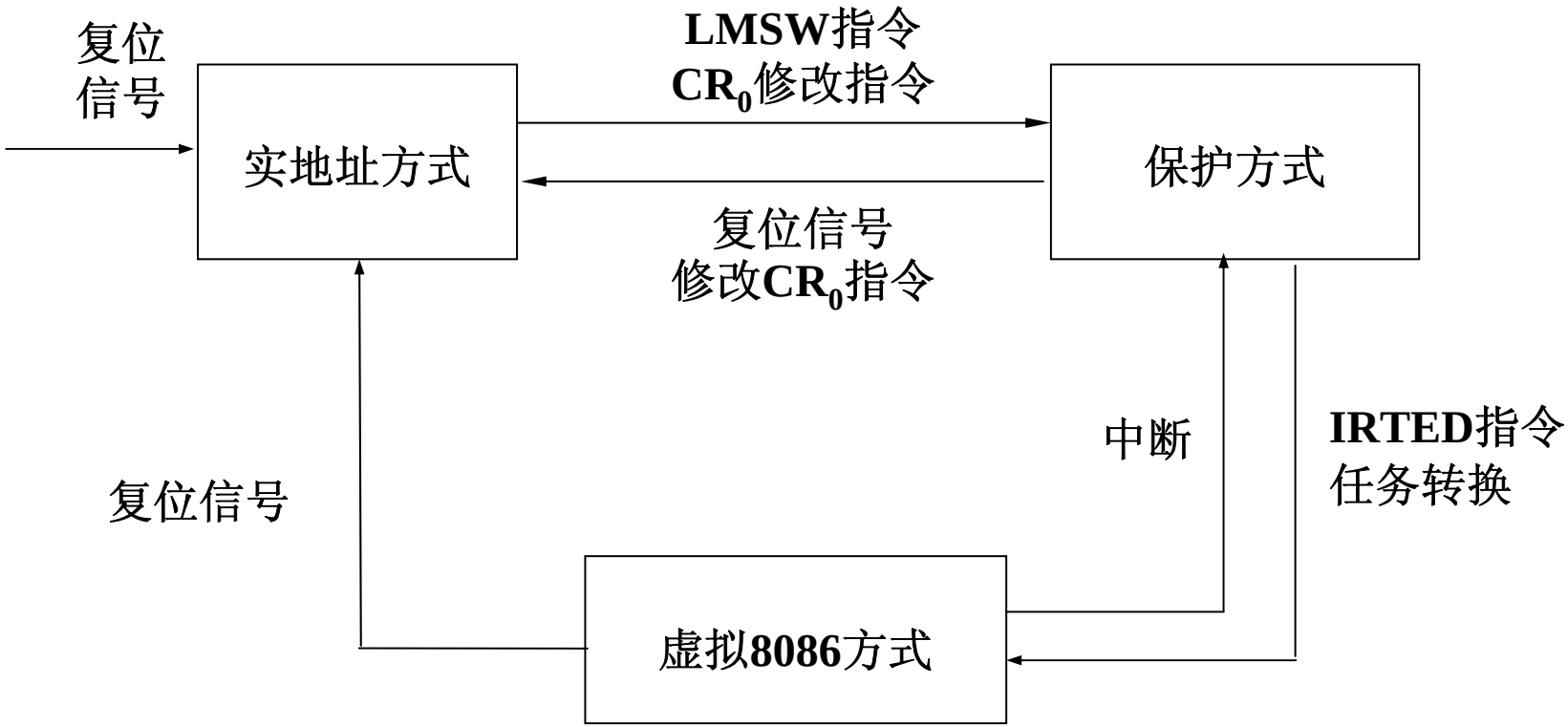
- 是指在执行多任务操作时，对不同任务使用的虚拟存储器空间进行完全的隔离，保护每个任务顺利执行。

（3）虚拟8086方式：

- 是指一个多任务的环境，即模拟多个8086的工作方式。

80386工作方式之间的转换

80386的3种工作方式（即实地址方式、保护方式和虚拟方式）之间的转换如下图所示。



6. 80386指令流水线（instruction pipeline）

- 80386的指令流水线由三个逻辑部件构成：

一、总线接口部件；

- 执行存储器周期并向预取部件传递指令代码。

二、预取部件；

- 有一个能容纳16字节的队列。预取部件把对齐了的指令存放在队列中，以便于指令译码部件有效译码。

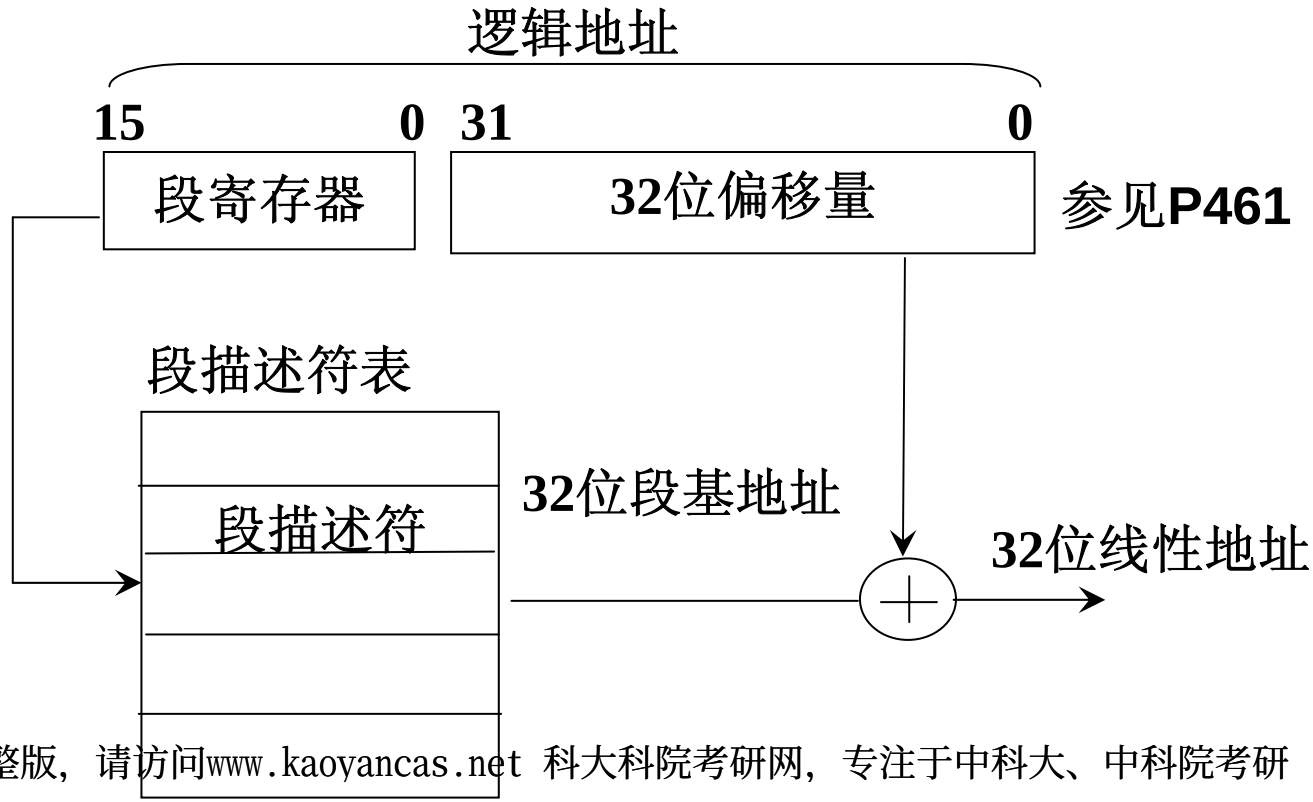
三、指令译码部件。

- 从16字节的指令代码队列中提取代码，形成最后指令，且以三个字为一条指令按序排好。

7. 80386的内存管理

- 自80386到Pentium，保护模式下的内存除了支持分段管理机制以外，还支持分页管理机制（P460～475）
- 分段机制：
  - 与实模式类似。不同的是段寄存器存放的一个段描述符表（Segment Descriptor Table）中某一描述符项在表中的索引值。
  - 80386以上CPU的“内存段描述符”包括32位的段基地址A31～A0，指示一个段的开始地址。

- 段基地址＋偏移地址＝ 32位线性地址（可寻址4GB）；
- 线性地址：逻辑地址转换为物理地址的中间过程。如果禁止分页，则线性地址就是物理地址。
- 线性地址的计算如下图所示：



分页机制

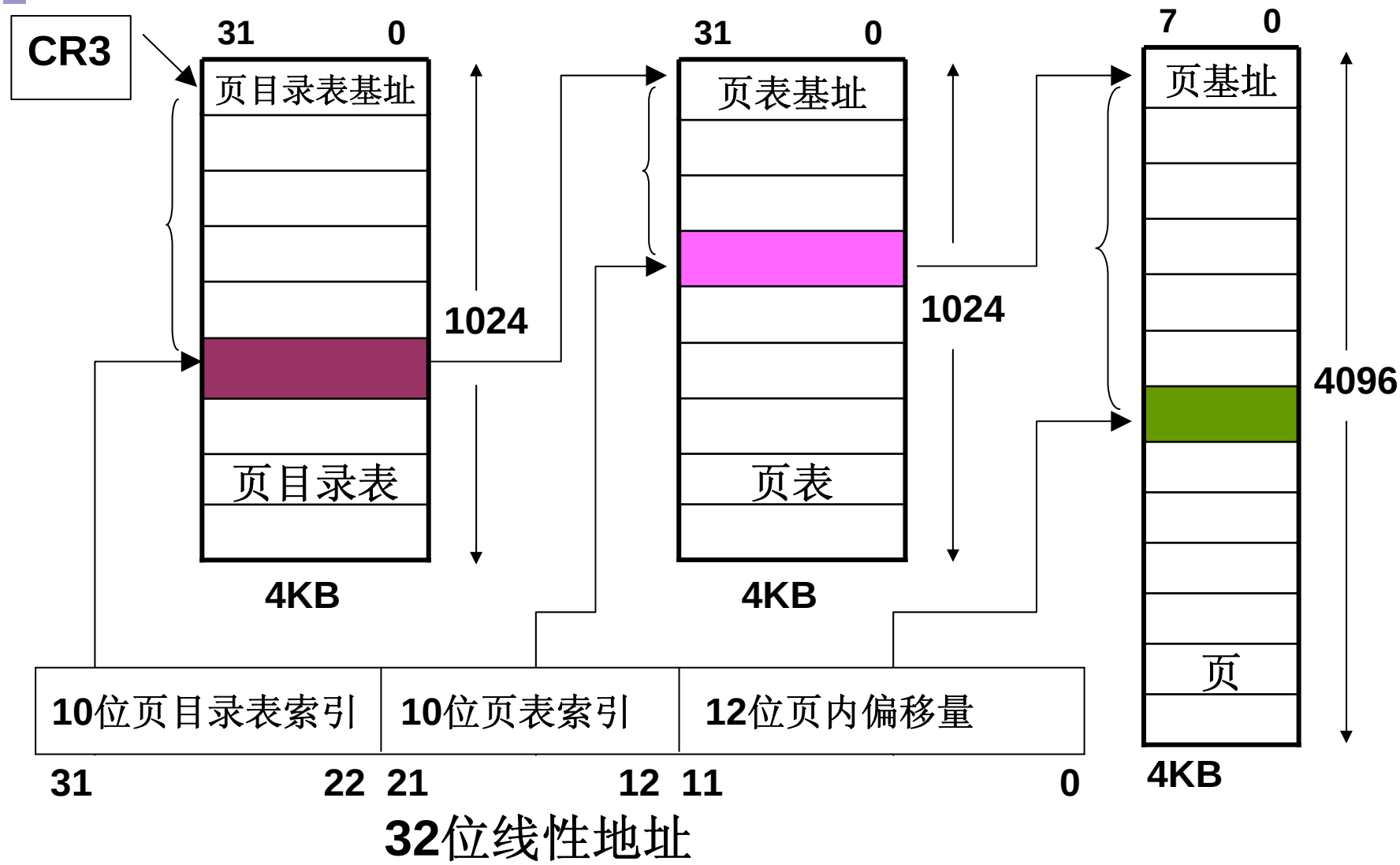
- 基本思想：
  - 将物理地址和线性地址分为同样大小的**4KB**的页，通过线性地址与物理地址之间灵活的映射关系，将线性地址转换成物理地址。分页也称为页变换。
- 分页的目的：
  - 对远大于物理地址空间的存储空间进行管理，可将磁盘等存储设备的空间映射到内存，从而实现虚拟内存。分页还可有效利用内存碎片。（教材**P460**）
- 问题：
  - **32**位物理地址可以管理的存储空间为**4GB**，等于**1M**个**4KB**的页，每个页起始地址是**4**个字节，**1M**个页的索引就需要**4MB**空间。分页应如何组织和管理？

两级页表的索引机制

- 页表：用于存放每页32位基地址（占4个字节）的表。在80386以后的处理器中，每张页表的大小为4KB，可存放1024个页的基地址。
- 页目录表：用于存放每个页表的32位基地址的表。页目录表的大小也是4KB，总共可存放1024个页表的基地址。
- 页目录表的基地址由CR3提供，CR3只给出页目录表基地址的高20位，低12位隐含全为0。
- 页目录表管理1024个页表 → 每个页表又管理1024个页 → 每一页内存放4KB的程序或数据。这样2级页表只需要使用8KB空间。

线性地址到物理地址的转换

- 当控制寄存器**CR0**的**PG**为**0**时，禁止分页，线性地址就是物理地址。
- 当控制寄存器**CR0**的**PG**为**1**时，允许进行分页（页变换）。页变换过程就是把线性地址转换成物理地址的过程。
- 页变换在段变换之后进行，段变换得到的线性地址被视为以下三部分：
  - 高**10**位——页目录表索引号
  - 中**10**位——页表索引号
  - 低**12**位——页内偏移量
- 页变换过程如图所示。



8. 80386的地址流水线（Addrss Pipeline）

- 地址流水线由分段部件（**Segmentation Unit**）、分页部件（**Paging Unit**）和与指令流水线共享的总线接口部件组成。
- 分段部件与分页部件集成于同一芯片，充分利用流水线与并行执行的优点，将逻辑地址到线性地址再到物理地址的各个步骤重叠起来，与其它总线周期重叠进行，并且能在当前总线周期结束之前完成转换。无需在总线周期上增加时钟个数。又因为存储管理功能是在芯片内进行的，所以它不是外部总线的一部分。在芯片之外也不再需要存储管理元件。

Intel 80486微处理器简介

1．80486的主要特性

（1）首次增加RISC技术。

（2）芯片上集成部件多。数据高速缓存、浮点运算部件、分页虚拟存储管理和80387数值协处理器等多个部件。

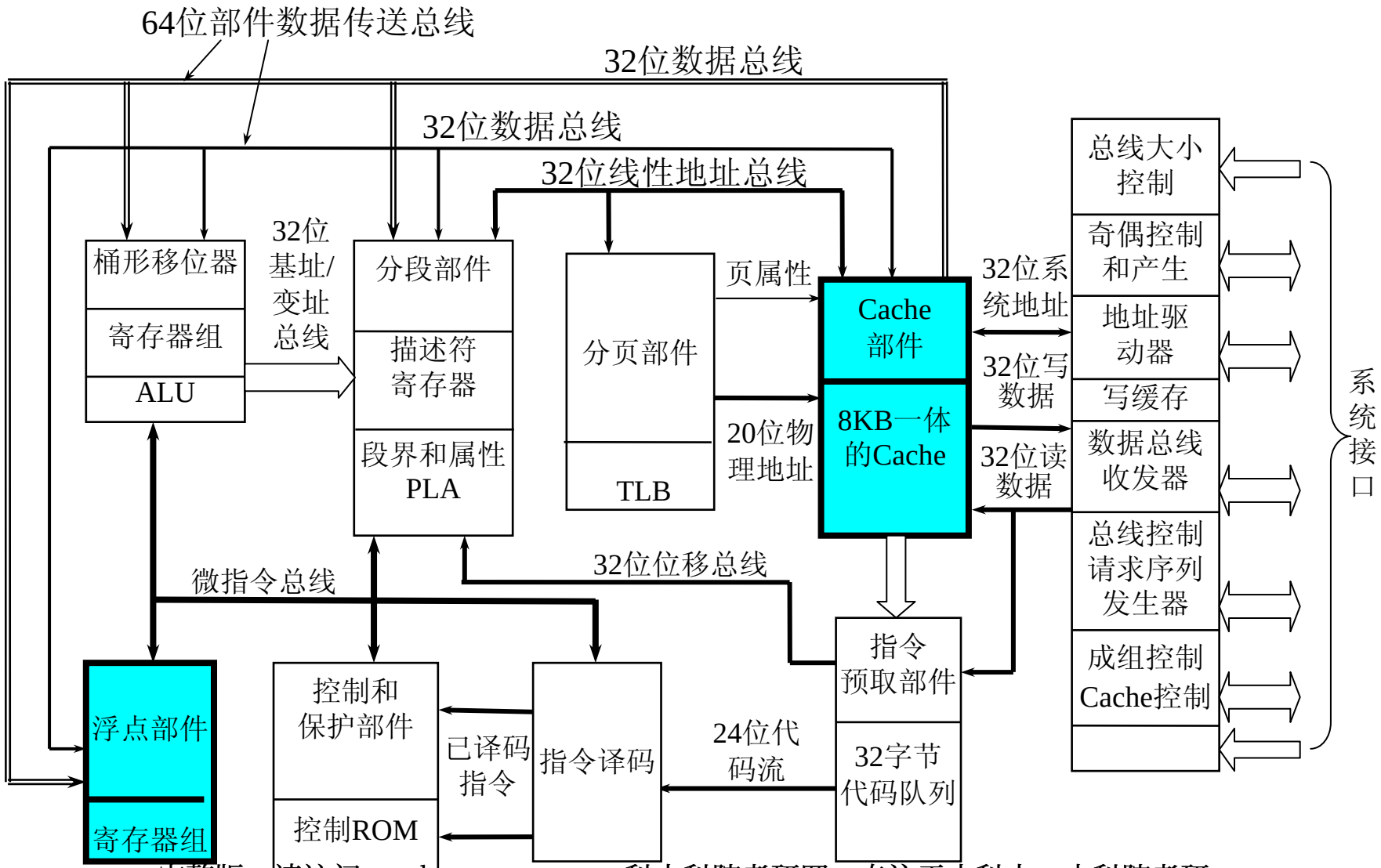
（3）高性能的设计。

（4）完全的32位体系结构。

（5）支持多处理器。

（6）具有机内自测试功能，可以广泛地测试片上逻辑电路、超高速缓存和片上分页转换高速缓存

2．80486的内部结构



- 在以80386为基础的系统中，协处理器80387和内部的寄存器组、Cache控制器和存储体等属于不同的独立芯件，不在CPU芯片之内。
- 而在80486中，浮点部件（增强后的80387，内部数据总线64位）FPU、Cache控制部件以及8KB的指令和数据共用Cache被集成到CPU内部。
- 80486其余部件与80386非常相似，但指令流水线的预取部件、译码部件，以及控制逻辑部件有改进和优化，Cache被集成到指令执行流水线中。
- 引入RISC计算机的优点，一些最常用、最简单的指令在一个时钟周期完成，平均每条指令的执行周期为1.2个时钟周期，达到了小型计算机的水平。

Pentium 微处理器简介

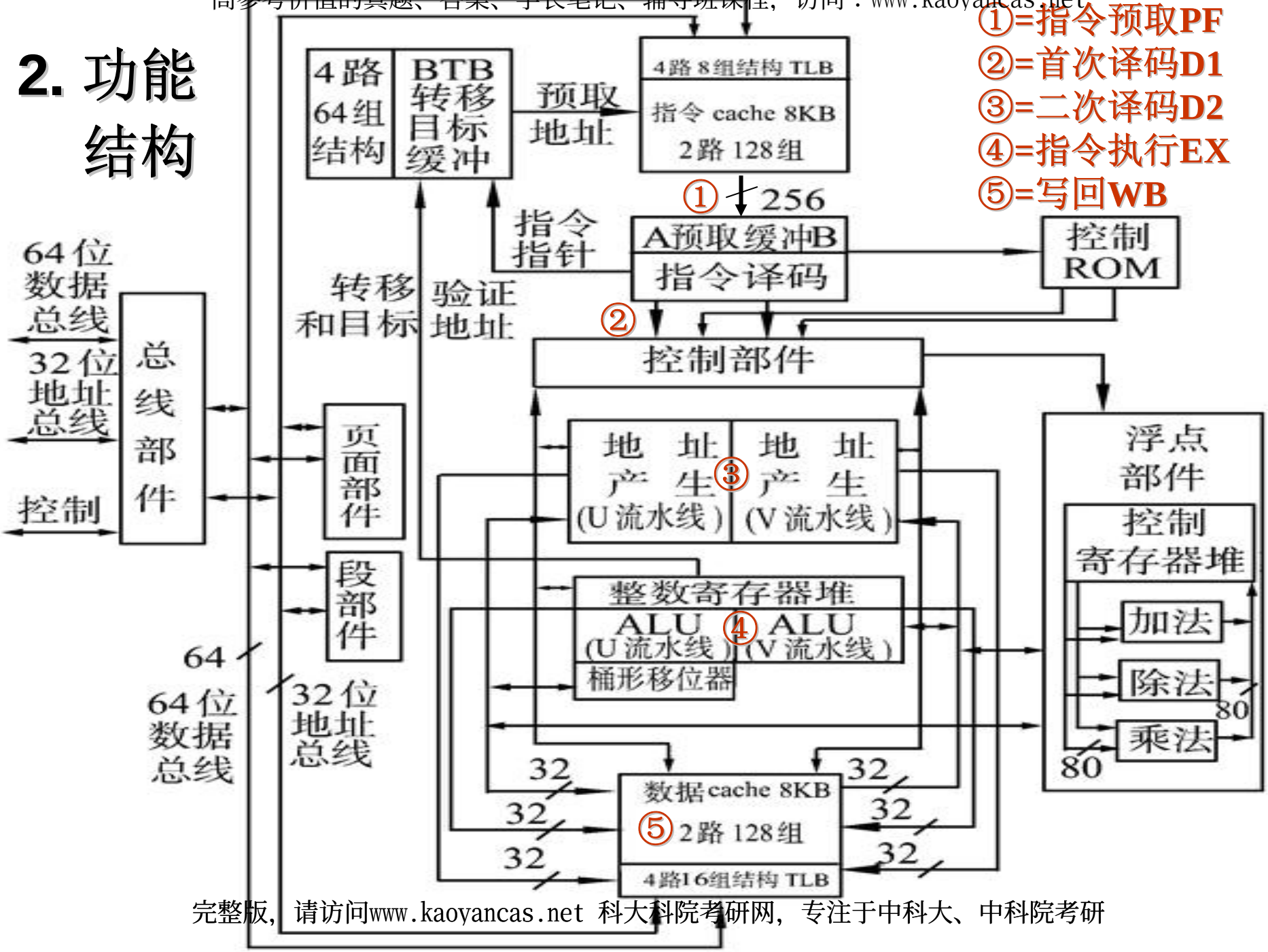
1. 特点：

- 80x86系列微处理器兼容
- 有64位数据总线、 32位地址总线，寻址空间4GB。
- RISC型超标量结构，两条并行的5级整数指令流水线，一条8级浮点流水线。
- 具有超级流水线技术的高性能浮点运算器。
- 数据-代码分离式高速缓存。
- 利用片上分支目标缓冲器提高分支指令预测准确性。
- 常用的指令不采用微程序设计，而改用硬件实现。
- 支持64位外部数据总线突发传输方式。
- 增加了SMM模式，增强的错误检测和报告功能。
- 通过APIC总线支持多处理器系统。

1995年生产的Pentium CPU

采用296引脚的引脚栅格阵列（PGA）陶瓷封装技术

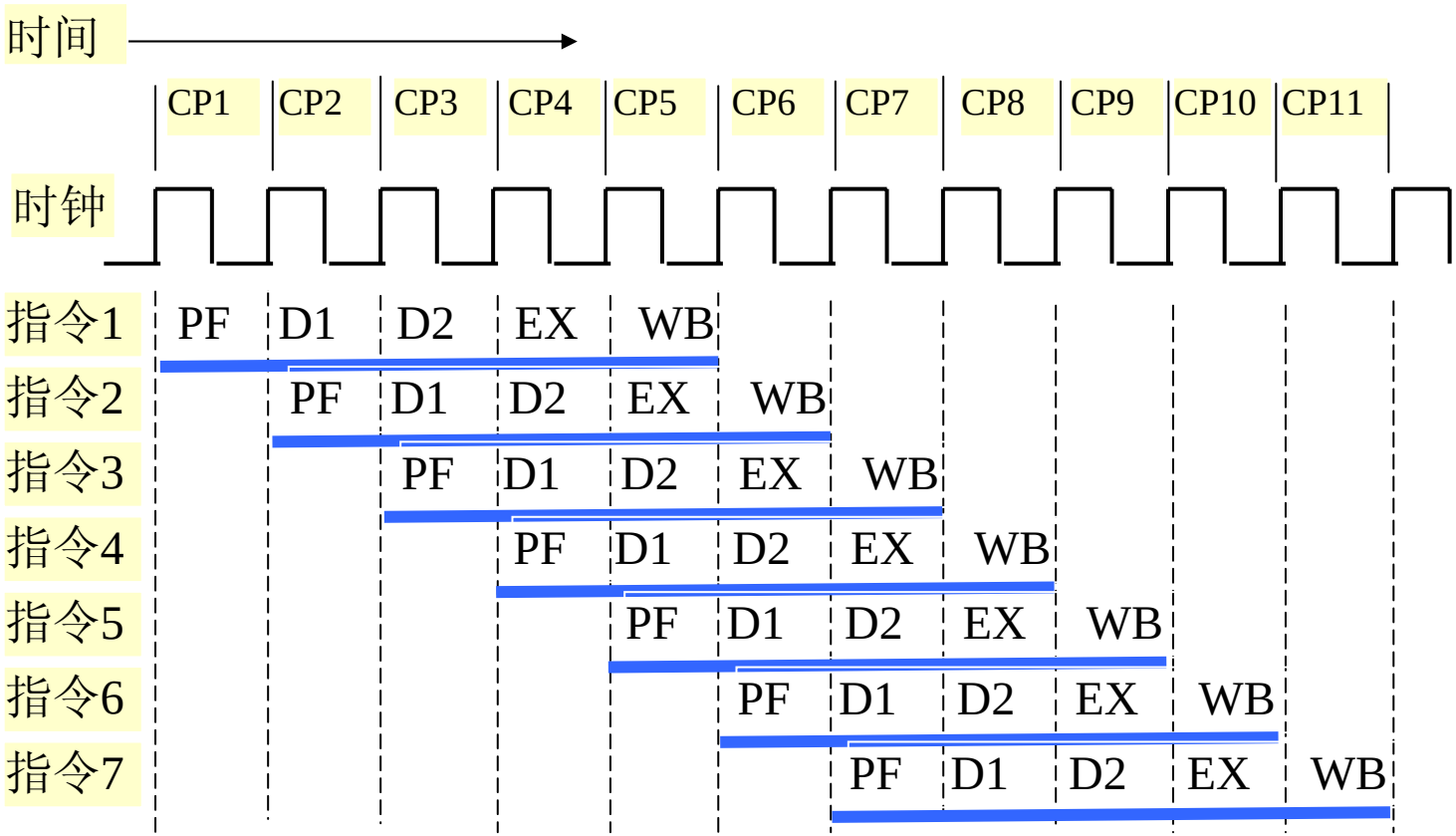
2. 功能结构



3. 流水线与超标量

- 流水线：
  - 在**CPU**中把一条指令分解成多个可单独处理的操作，每个操作在一个专门的硬件站（**stage**）上执行。这样一条指令需要顺序地经过流水线中多个站的处理才能完成，但是前后相连的几条指令可以依次流入流水线中，在多个站间重叠执行，可实现指令的并行处理。
  - 在**80486**中，每条指令被分解分预取（**PF**）、一次译码（**D1**）、二次译码（地址生成**D2**）、指令执行（**EX**）和结果回写（**WB**）**5**个步骤。指令在流水线上执行过程如图所示：

- 采用流水线方式，每条流水线每个时钟周期可执行一条指令。流水线以时间并行方式提高CPU的指令执行速度。

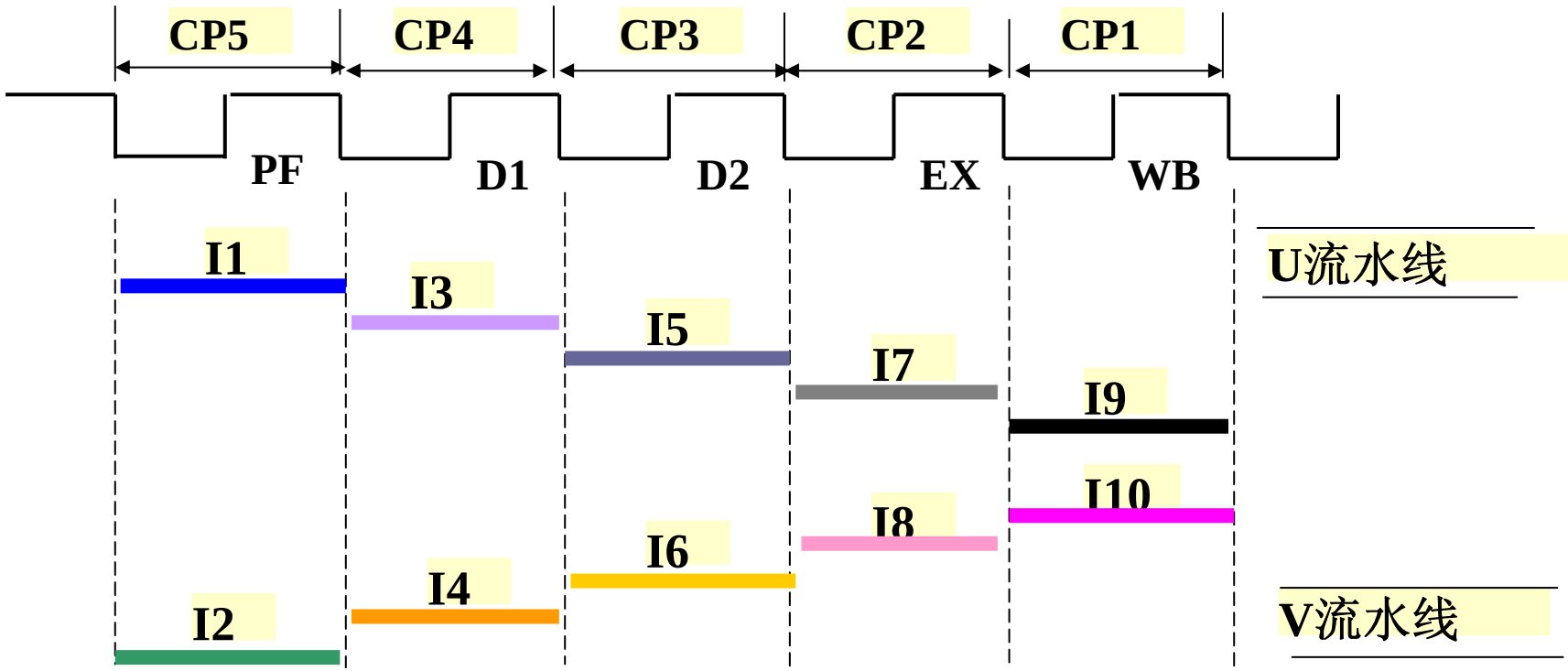


- 超标量结构：
  - 配置多个执行部件和指令译码电路，能同时执行多条指令。
- **Pentium**由三个执行单元组织而成：
  - 一个执行浮点指令（**FPU**）
  - 另两个执行整型指令（**U**流水线和**V**流水线）
  - 这意味着**Pentium**可以同时执行三条指令。
- 超标量结构的本质是利用空间上的并行处理方式提高**CPU**的性能，也是**Pentium**的核心技术。

Pentium微处理器的整数流水线

- Pentium内部有U和V两条整数流水线，每条都有独立的ALU、地址生成和Cache接口电路。
- 每条整数指令被分解成以下五个执行步骤：
  - PF(预取)：处理器从代码cache中预取指令
  - D1(1次译码：进行指令译码，确定操作码和寻址信息。此阶段还进行指令的“成对性检查”和“分支预测”。
  - D2(2次译码)：产生访问存储器的地址。
  - EX(执行)：处理器或者访问数据cache，或者利用ALU、筒型移位器或其他功能单元计算结果。
  - WB(写回)：根据指令执行结果更新相应的寄存器和标志寄存器。

- Pentium微处理器将两条整数指令分别发送到U、V两条流水线上，以并行方式同时执行（即所谓的双发射技术）。每条指令按照流水线方式分为5个步骤执行。Pentium每个时钟周期可以执行两条整数指令。



4. 高性能的FPU—浮点流水线

- Pentium CPU内部集成的FPU有一条8级浮点流水线，每条浮点指令由预取PF、首次译码D1、二次译码D2、读取操作数EX（从存储器或寄存器读取）、首次执行X1、二次执行X2、写浮点数WF和出错报告ER共8个操作步骤组成。
- 对于诸如浮点加法、浮点减法、浮点乘法以及比较之类的“基本”浮点指令来说，能以每个时钟一条指令的速率执行。
- Intel系列浮点部件不仅支持单精度（32位）浮点计算、同时也支持双精度（64位）和80位的扩展精度浮点运算。

5. 流水线方式的一些问题

- 要使流水线具有良好的性能，必须使流水线畅通流动，不发生断流和停滞。但由于指令在流水线执行过程中会出现以下三种相关冲突，实现流水线的不断流是相当困难的，这三种相关是：
  - 资源相关
  - 数据相关
  - 控制相关。

（1） 资源相关

- 所谓资源相关，是指多条指令进入流水线后在同一机器时钟周期内争用同一个功能部件所发生的冲突。
- 当数据和指令放在同一个存储器且只有一个访问端口时，便发生两条指令争用存储器资源的相关冲突。
- 解决冲突的办法：

  a）第二条指令停顿一拍后再启动，躲避竞争。

  b）增设一个存储器，将指令和数据分别放在两个存储器中（现代CPU普遍采用的方法）。

（2）数据相关

- 在一个程序中，如果必须等前一条指令执行完毕后，才能执行后一条指令，那么这两条指令就是数据相关的（如读后写**WAR**、写后读**RAW**）。
- 在流水计算机中，指令的处理是重叠进行的，前一条指令还没有结束，第二、三条指令就陆续地开始工作。由于多条指令的重叠处理，当后继指令所需的操作数，刚好是前一指令的运算结果时，便发生数据相关冲突。
- 为了解决数据相关冲突，流水**CPU**的运算器中特意设置若干运算结果缓冲寄存器，暂时保留运算结果，以便于后继指令直接使用，这称为“向前”或定向传送技术。

（3）控制相关

- 控制相关冲突是由转移指令引起的。当执行转移指令时，依据转移条件的产生结果，可能顺序取下条指令；也可能转移到新的目标地址取指令，从而使流水线发生断流。
- 上述断流被称为流水线“气泡”，是影响流水线效率的主要因素。
- 为了减小转移指令对流水线性能的影响，可以采用“延迟转移法”。但是目前最普遍采用的是**分支转移预测和推测执行**技术。

（4）指令配对问题

- 能够在U、V两条流水线并行执行的整数指令要满足一定的前提条件，Pentium数据手册定义了如下配对规则：
  1. 两条指令都是简单指令（即完全可由硬件执行而无需任何微码控制，在一个时钟周期内执行的指令）。
  2. 没有写后读和读后写的依赖关系。
  3. 一条指令不能同时既包含位移量又包含立即数。
  4. 带前缀的指令只能出现在U流水线中。
  5. 此外，条件分支转移指令和非条件分支转移指令，只有当它们作为配对中的第二条指令出现时才可以配对。
- Pentium CPU在一次译码阶段进行指令配对检查，不满足上述条件的只能在单条流水线上执行。

（5）分支转移预测和推测执行技术

- 为什么要预测和推测？
  - 例：外科医生手术时与器械护士的配合问题。
- 程序中必不可少的包含各种类型的转移指令。在进行流水线操作时，由于这些转移指令的出现，导致通过预取操作从**Cache**中预取到预取部件中的指令和已经预译码的指令作废，再装入正确的指令序列重新处理。这样执行效率反而比没有指令预取还要慢。

- 转移指令在经过**D1**阶段的指令译码即可被处理器发现，但转移指令的转移条件多数是指令执行后的寄存器标志位状态。也就是说转移的方向要等到前面的指令经过**WB**步骤更新了标志寄存器后才能确定。

- 通常情况下，一旦处理器发现条件转移指令，则停止后续指令的预取，要等到前面的指令执行完毕并更新了标志寄存器，确定了转移方向以后再重新开始流水线操作，而这时流水线已经空置，流水线出现了气泡，执行部件被迫停顿，所以条件转移指令对流水线效率的影响很大。

- 如果在D1（一次译码）阶段能够对程序转移方向进行正确的预测，就可以根据预测结果提前按照正确的方向预取指令和进行译码，避免流水线出现气泡，这就是分支转移预测和推测执行所要研究的内容。

- 分支转移预测的算法研究和推测执行技术是新一代CPU研发领域的热点问题之一。

- 分支转移预测和推测执行是CPU动态执行技术中的主要内容，而动态执行是目前最新一代CPU的标志性的先进技术之一。

Pentium微处理器的分支转移预测策略

- 思想：
  - 根据最近执行的分支指令的历史状况动态地预测程序分支。
- 策略：
  - 设置一个分支目标缓冲器**BTB（Branch Target Buffer）。BTB**保存最近所遇到的条件转移指令信息，每当在指令流中发现条件转移指令时，就对**BTB**进行检查，若该指令的信息在**BTB**已存在，则依据它的历史信息进行预测，并按预测的转移方向继续预取指令和译码，如果预测正确，可以保证流水线不会空置，如果预测错误，流水线只能刷新，重新读取指令，这样处理器有**3~4**个周期的延迟。

- **Pentium所采用的预测算法：**
  - **BTB**中所保存的每条转移指令附加了**2bit**的历史信息，用于表示可能发生转移的可能性程度，总共可以表示**4**种状态：**11**是最可能发生转移，**10**其次，**00**是最不可能发生转移。
  - 新进入**BTB**的指令其历史信息为**11**，此后出现一次未发生转移的情况，则将其减**1**，直到减为**00**，而发生一次转移，则将其加**1**，直到为**11**。
  - 如果状态为**11**或**10**，则预测为发生转移；如果状态为**00**或**01**，则预测为不发生转移。
- 分支预测对循环的意义最为显著，只有在进入和退出循环时才会发生预测错误，而在循环中间过程中进行的分支预测都是正确的。

（6）哈佛结构－指令和数据Cache分设

- 486片内8KB的Cache即存放数据又存放指令。
- Pentium则设置2个8KB的Cache，一个作为代码Cache（I－Cache），另一个作为数据Cache（D－Cache），这种结构又称为“哈佛结构”。
- 指令和数据使用不同的Cache，避免了指令预取和数据操作的冲突，提高了Pentium的性能。
  - Pentium在流水线的指令预取阶段，指令是从代码I-Cache中取出，与D-Cache的操作无关。
  - 对于486而言，由于数据和指令合用一个Cache，指令预取与前面指令的数据操作之间发生冲突的几率较大，从而影响了CPU的执行效率。

Pentium Pro（P6）高能奔腾简介

- Intel首次将256KB或512KB的L2 Cache（SRAM）与CPU集成在一个芯片内部。L2 Cache 通过一条64位宽且与CPU等时钟速率的专用总线，实现与CPU间的通信，从而提高了CPU性能。
- Pentium Pro还采用3路超标量体系结构和14级的超级流水线。超级流水线有更多的Stage，可以支持超长指令字技术，进一步提高了CPU的并行处理能力。
- Pentium Pro其内部的数据总线和地址总线均为64位。

乱序执行—Pentium Pro的标志性新技术

- 此前的x86处理器是按照指令在程序中的本来顺序执行的，这种执行方式经常会陷入到一个费时的指令执行状态中，任何引起延时的指令（如执行等待指令）都会影响到流水线的吞吐量。
- Pentium Pro采用指令动态执行技术，将多条指令译码后存放在一个指令池（Instruction Pool）中，指令池打开一个足够大的指令窗口（30条指令），在这个指令窗口中进行多分支指令预测和数据流分析，然后再以一个优化的顺序预测执行，已形成操作数的指令先行派送到流水线中执行，保证流水线高效不停顿地运行，这一技术打破了原有的顺序，可将处理器停滞时间限制到最小。该技术称为“**乱序执行**”（Out of order execution)。是Pentium Pro与此前CPU的**根本区别**。

Pentium II

- 随着多媒体技术的发展，面向个人和一般商务应用的CPU开始从“如何增加计算能力”向“如何增强多媒体信息处理能力”的方向转变。
- P Ⅱ是在上述背景下研发的一种新体系结构的微处理器，实质上是将多媒体扩展MMX集成到了Pentium Pro中。
- PⅡ内核约有750万支晶体管（Pentium Pro为550万支，Pentium为310万支），采用深0.35微米加工工艺制作而成，全面改善了PC机系统在整数、浮点计算及多媒体信息处理方面的性能，给用户带来新一级更高性能的可视计算能力。
- 双重独立总线结构、内置MMX技术、动态执行、单边接触插盒（Slot 1）是PⅡ的4大主要特征。

（1）双重独立总线

- 以往的x86 CPU仅有一条数据总线与内存、L2 Cache以及PCI总线相连，任何时候只能访问一个设备。
- DIB（Dual Independent Bus-DIB）是指CPU带有两条独立的总线，其中一条（后端）总线连接L2 Cache；另一条（前端）总线连接PCI和内存。很好地解决了处理器到内存总线带宽受限的瓶颈问题，从而提高了通信带宽。

（2）内置MMX多媒体扩展技术

- MMX的主要技术特点有：

（1）SIMD（单指令多数据）技术是MMX的基础，它使得多条信息可由一条单一指令来处理。

（2）MMX指令具有很强的通用性，不仅能满足建立在当前及未来算法上的PC机应用程序的大部分需求，而且可用于编码译码器、算法及驱动程序等。

（3）增加了8个64位MMX寄存器。

（4）新增加了57条指令。用这些指令完成音频、视频、图形图像数据处理，使多媒体、通信处理能力得到大幅度提高。

（5）引入了适合视频信号处理的“饱和”运算。

（3）动态执行技术的。单位时间内能处理更多的数据，提高处理器的性能。

（4）新的单边接触式插盒

- **SEC（ Single Edge Contact）封装有242个触点，其插槽被命名为Slot 1。与传统的296引脚的Socket 7大相径庭。**
- **Socket 7采用单64位总线，当时钟为66.6MHz时，最大可达533Mbps的传输带宽，即使时钟增加至100MHz，带宽达到800Mbps也难以满足高端系统的需求。**
- **Slot1则是使用双64位总线，后端总线（Back end Bus）独立地与二级缓存交换数据，使CPU的数据吞吐量得到大幅度提高。**

Pentium 4 CPU简介

- 物理特性（三个版本）
  - 2000.8 Willamette
    - 0.18u铝工艺，1.4GHz ～ 2.0GHz
    - Socket 423/3400万晶体管，Socket 478/4200万晶体管
    - FSB 400/533MHz，12K μ Ops + 8KB L1 + 256KB L2
  - 2001.2 Northwood
    - 0.13u铝/铜工艺，1.6GHz ～ 3.06GHz
    - Socket 478，5500万晶体管，FSB 533/800MHz
    - 12K μ Ops + 8KB L1 + 512KB L2 + 2MB L3 （XE版本）
  - 2004.2 Prescott
    - 0.09u铜工艺，2.8GHz ～ ？
    - Socket 478/SocketT，12,500万晶体管
    - FSB 800MHz，16K μ Ops + 16KB L1 + 1MB L2

Netburst体系结构

- 超级流水线，20级
- 高级动态执行
  - 乱序执行：一条指令暂时不能执行时，后面的指令可继续执行
  - 推测执行：为保证流水线不间断，先执行再判断，发现预测错误时再返回错误点重新开始。
  - 同时执行126条指令，同时执行48个读取操作和24个存储操作。
  - 增强的分支预测能力，分支目标缓存BTB，可追踪4096个分支目标地址。

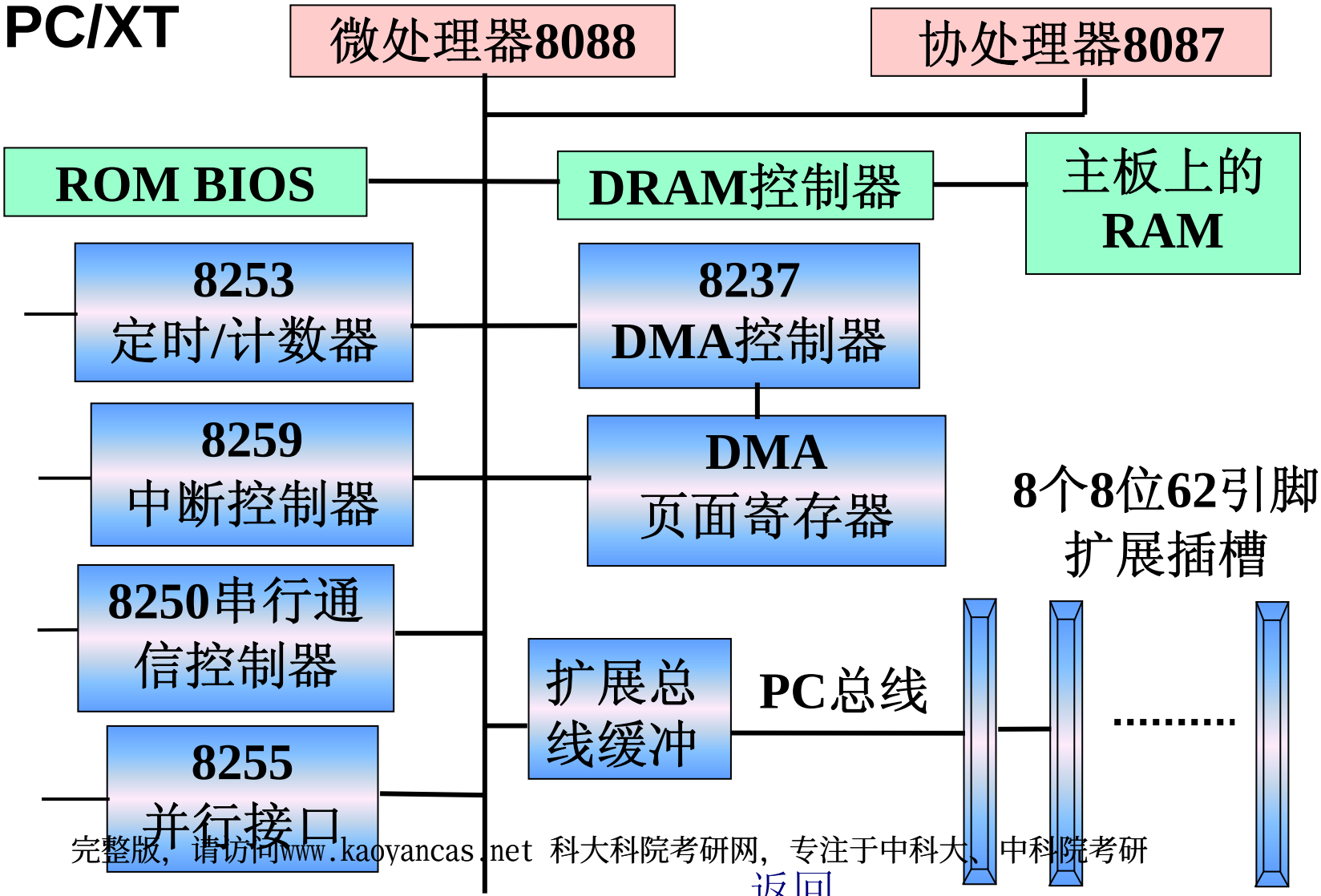
- 新型缓存体系结构
  - 12K微指令追踪缓存，存放已执行过的指令，以便分支预测错误时能够立即恢复到错误的分支点。
  - 8KBL1数据缓存，256～512KB L2高速传输缓存。
- 4倍速总线接口
  - FSB总线频率：100MHz（133MHz）×4 ＝ 400MHz（533MHz）
- 增强的单指令多数据流指令集SSE2
  - 144条128位多媒体指令
  - 128位整数运算、128位双精度浮点运算、数据类型可以灵活定义。
  - 改善了视频、音频、3D图形、网络等领域的数据处理能力。

超线程(Hyper-Threading, HT)技术

- 允许物理上单个的处理器采用共享执行资源的方法同时执行两个或更多的分离的代码流（线程）。
- 结构上HT技术由单处理器上的2个或者多个逻辑处理器组成，每个逻辑处理器都有自己的IA-32结构状态。
- 每个逻辑处理器都有自己的IA-32通用寄存器、段寄存器、控制寄存器、调试寄存器等。
- 逻辑处理器共享的资源包括执行引擎和系统总线接口。

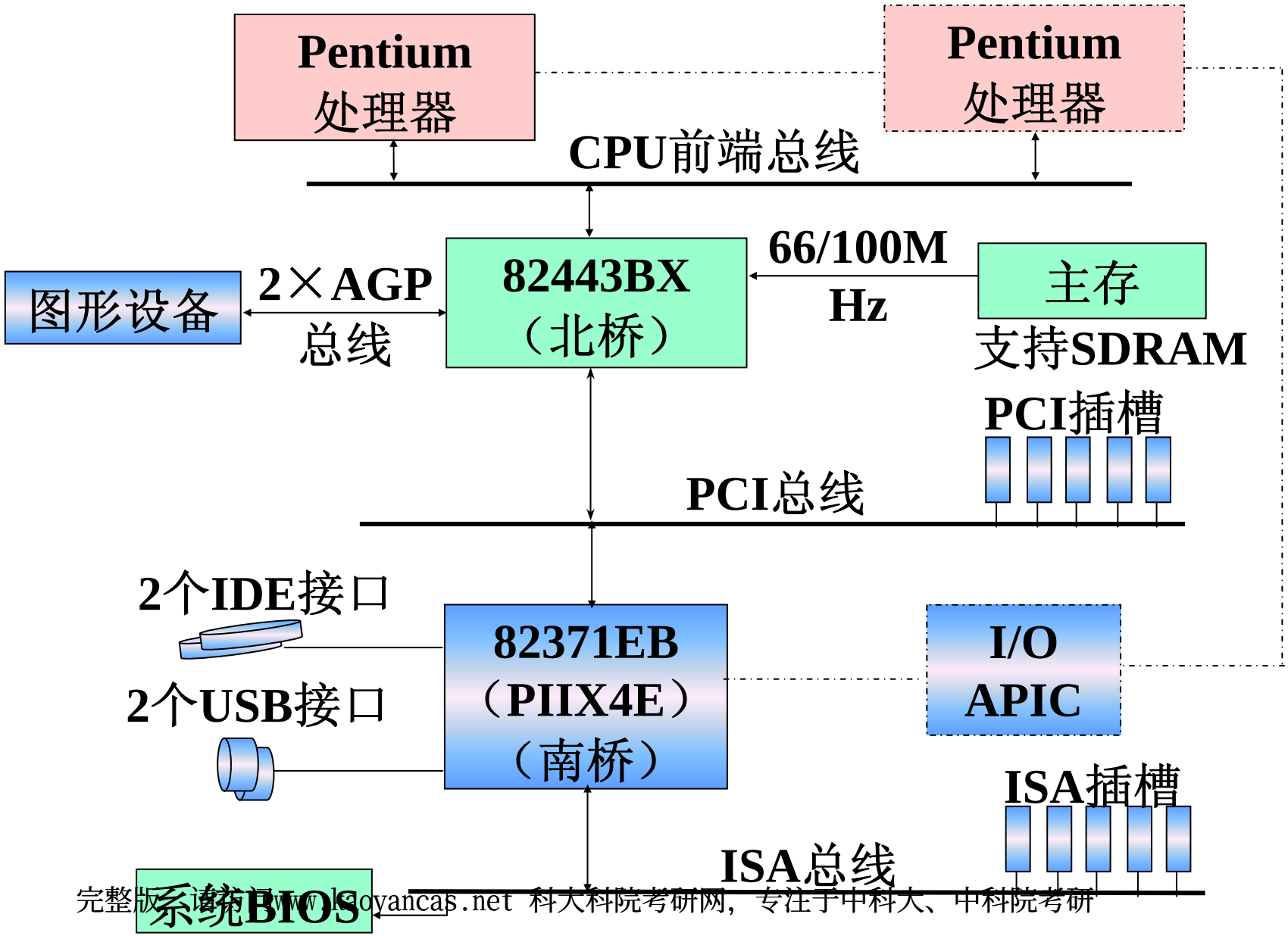
2-8 PC系列微机的基本结构

1、PC/XT

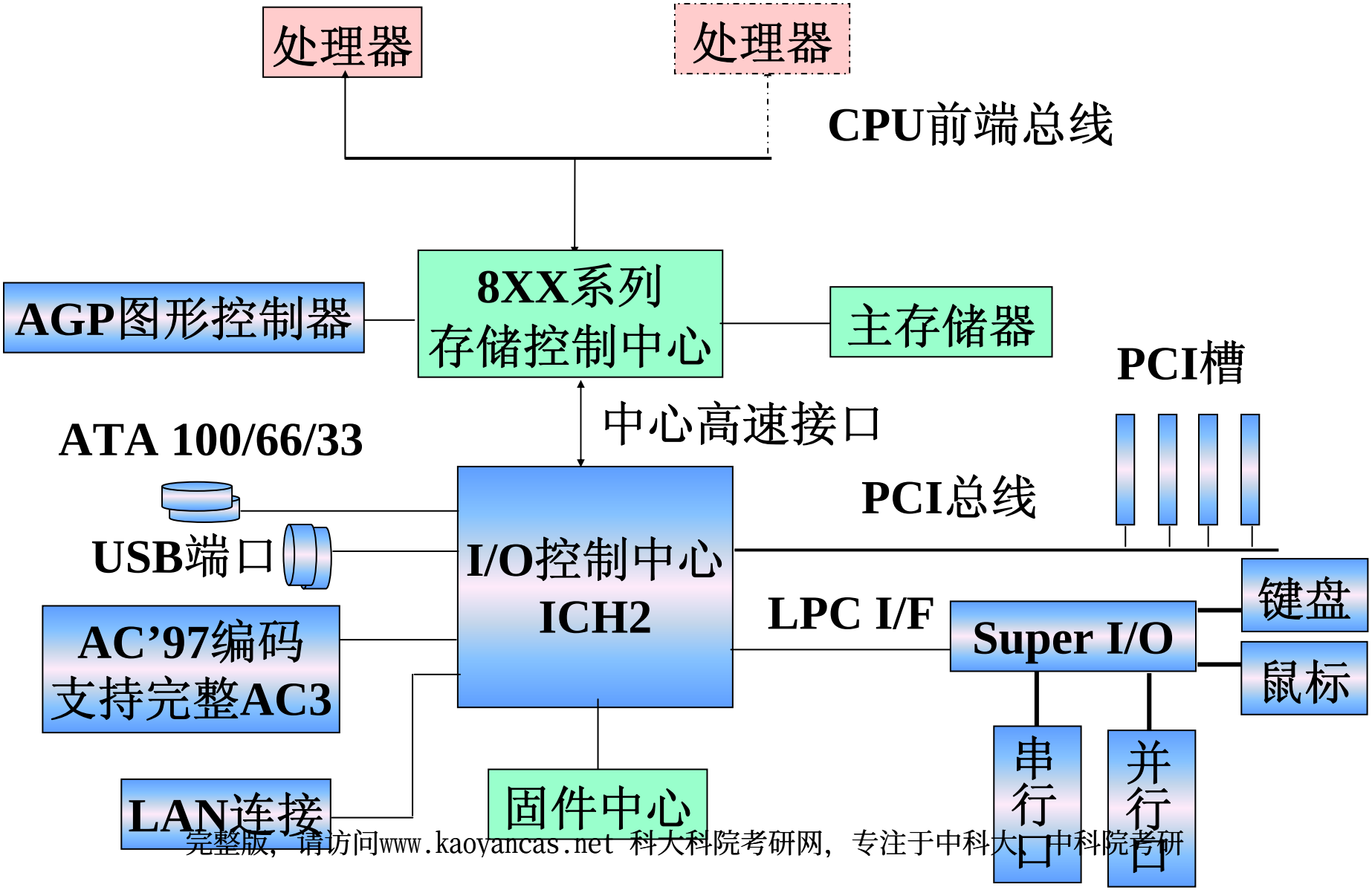


返回

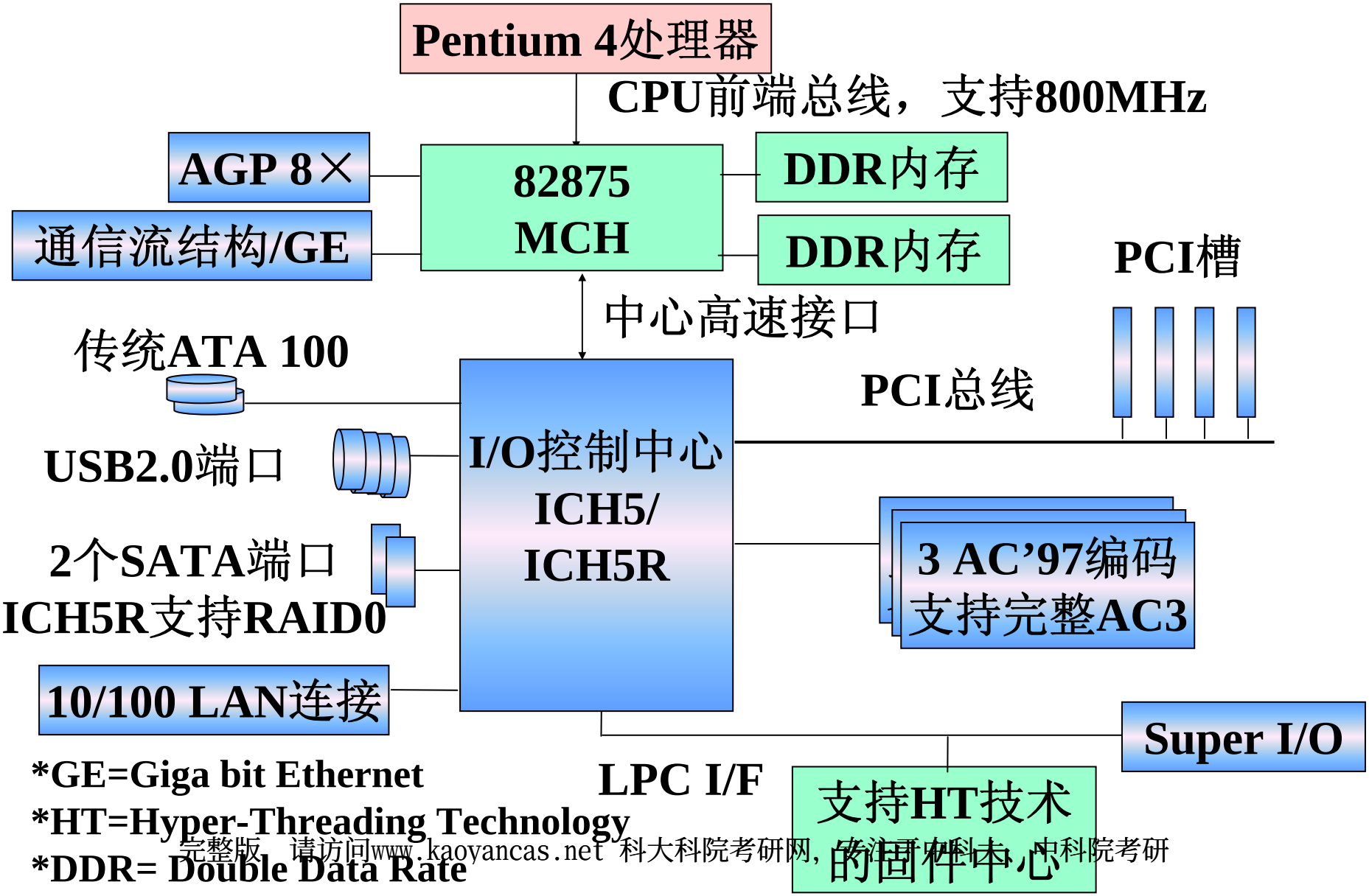
2、南北桥结构的Pentium II 微机逻辑框图



3、中心结构的Pentium III微机逻辑框图

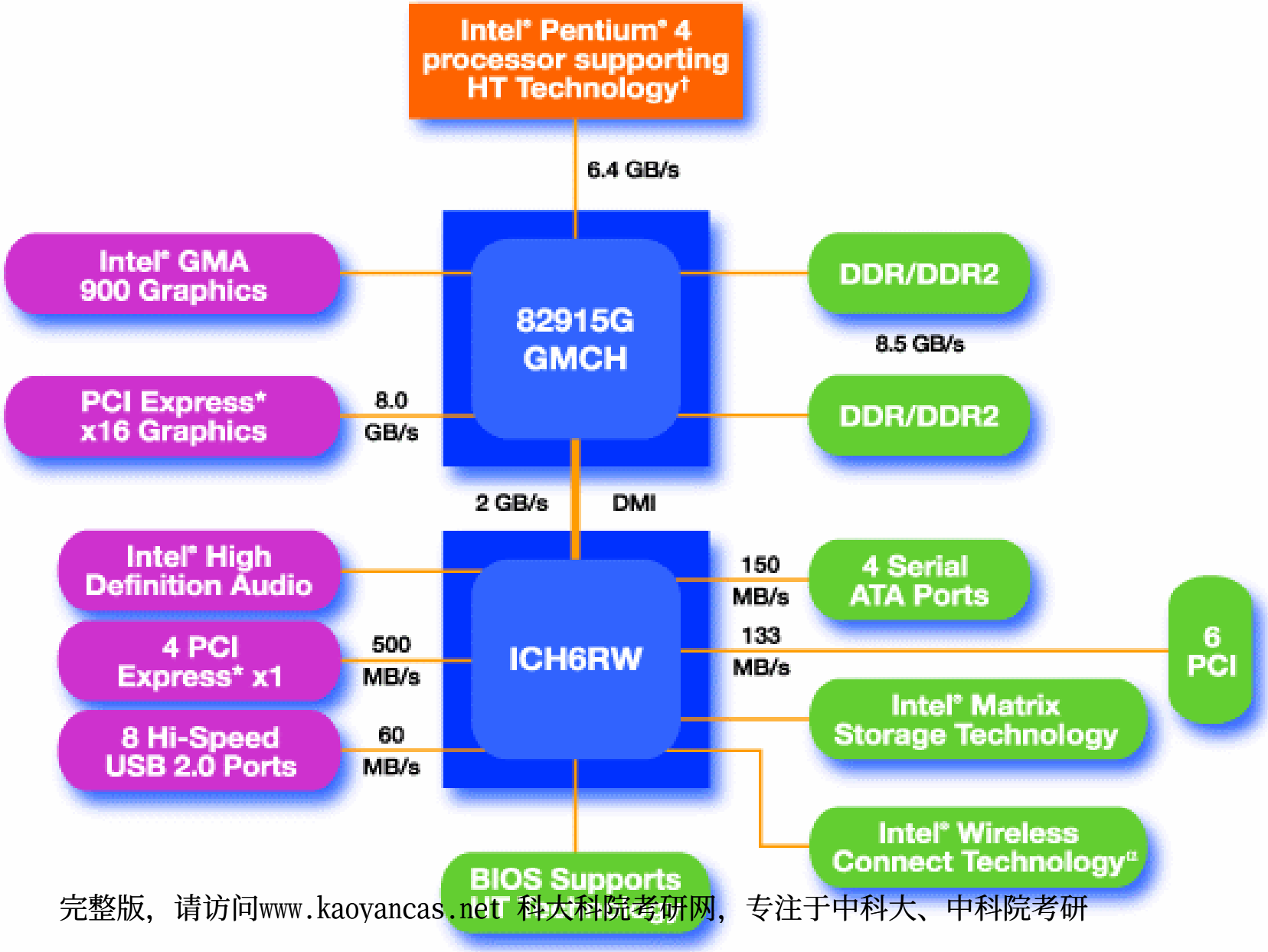


使用875芯片组的Pentium 4微机



*GE=Giga bit Ethernet
*HT=Hyper-Threading Technology
*DDR= Double Data Rate

使用915芯片组的Pentium 4微机



习题：P55～P56

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