

## 第二章 80x86计算机组织

- ❖ 2.1 基于微处理器的计算机系统构成
- ❖ 2.2 80X86 CPU
- ❖ 2.3 80x86 CPU的存储器管理
- ❖ 2.4 外部设备



## 本章教学目标与要求

- 学习目标
  - 了解计算机系统的主要组成部分；
  - 掌握存储器地址分段的方法以及存储单元物理地址的形成方法；
  - 熟悉**8086**各类寄存器的用途；
  - 熟悉标志寄存器各标志位的意义。
- 难点与重点
  - 存储器分段；
  - 存储器单元物理地址的形成；
  - 各寄存器的用途。

## 本章知识点

- 2.1 基于微处理器的计算机系统构成
- 2.2 80X86 CPU
  - 80x86基本结构
  - 80x86寄存器组：通用、专用、段寄存器
- 2.3 80x86 CPU的存储器管理
  - 存储单元的地址和内容
  - 实模式存储器寻址
  - 保护模式存储器寻址
- 2.4 外部设备
  - I/O地址
  - DOS和BIOS功能调用

## 2.1 基于微处理器的计算机系统构成 - MPU

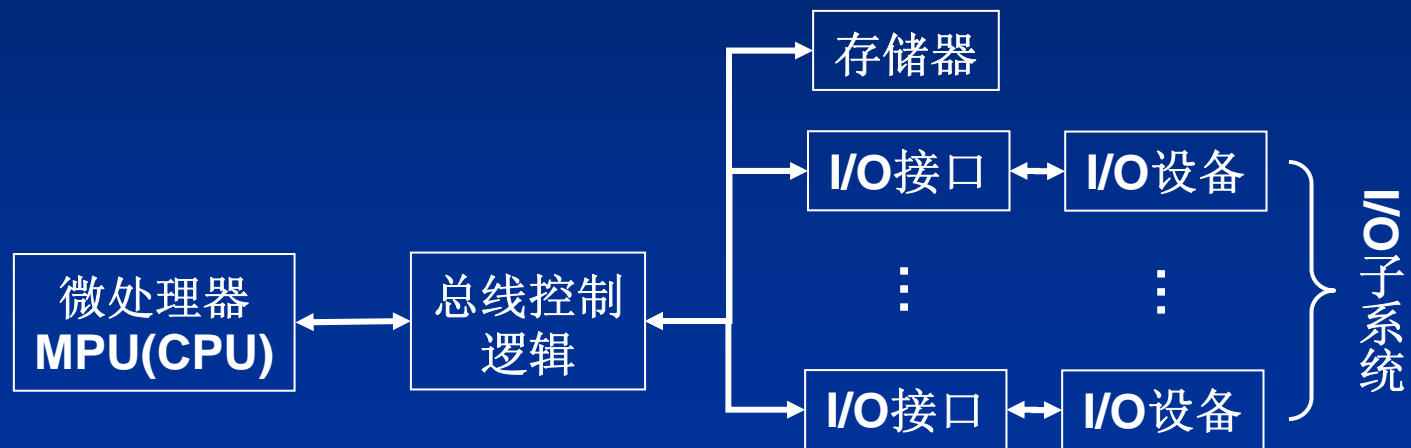
### 1. 微处理器MPU

- 计算机的硬件系统构成：
  - 运算器
  - 控制器
  - 存储器
  - 输入设备
  - 输出设备
- 微处理器：
  - 集成运算器、控制器及其它功能的超大规模集成芯片。

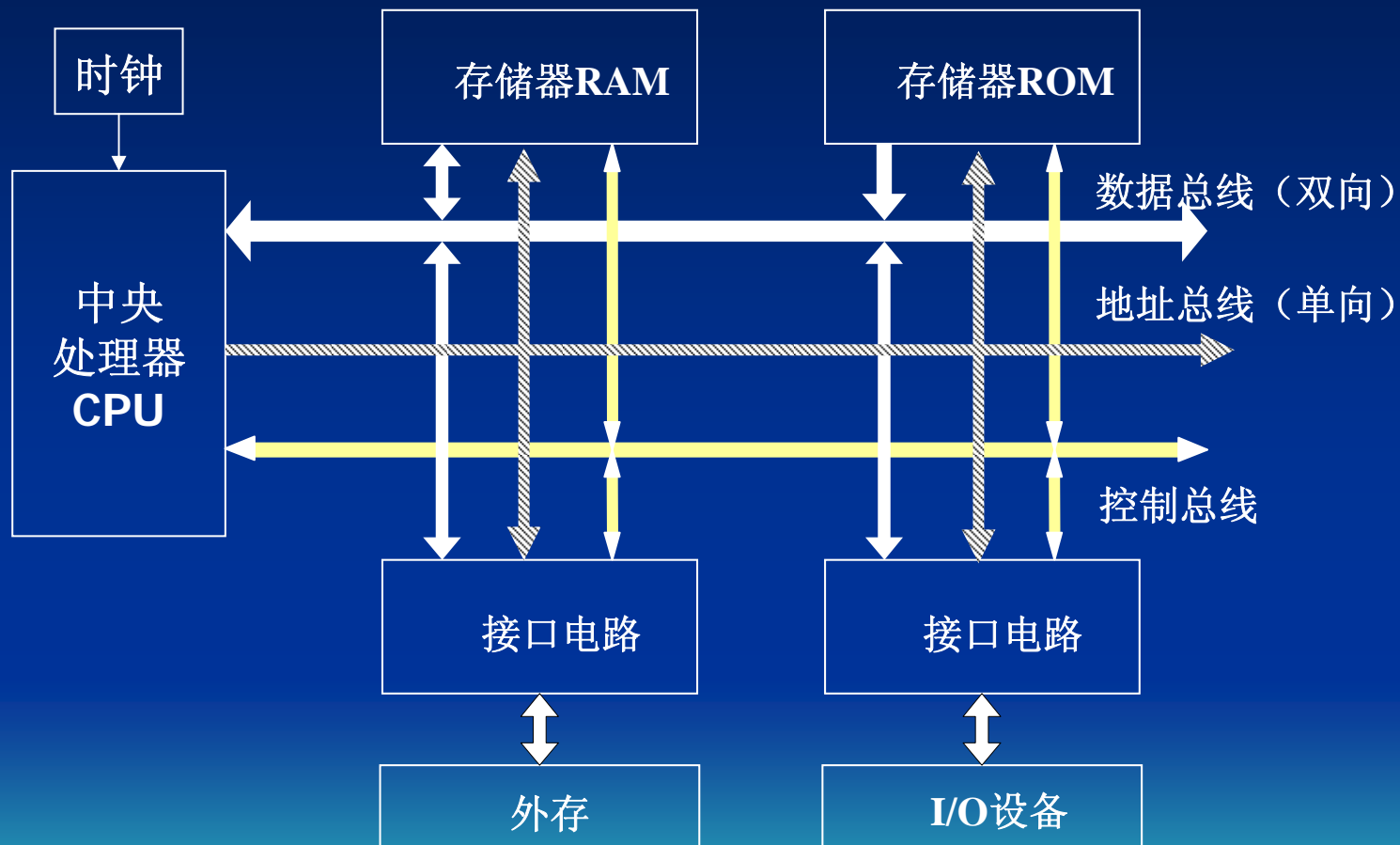
## 2.1 基于微处理器的计算机系统构成 – 硬件构成

### 2. 基于微处理器的计算机结构

- 以**MPU**为核心的计算机系统结构图：



## 2.1 基于微处理器的计算机系统构成 – 硬件构成



# 存储器

- **CPU** 是计算机的核心部件. 它控制整个计算机的运作并进行运算, 要想让一个**CPU**工作, 就必须向它提供指令和数据。
- 指令和数据在存储器中存放, 也就是平时所说的内存。

# 存储器

- 在一台**PC**机中内存的作用仅次于**CPU**。
- 离开了内存，性能再好的**CPU**也无法工作。



# 存储器

- 磁盘不同于内存，磁盘上的数据或程序如果不读到内存中，就无法被**CPU** 使用。

# 指令和数据

- 指令和数据是应用上的概念。
- 在内存或磁盘上，指令和数据没有任何区别，都是二进制信息。

## 指令和数据

- 二进制信息:

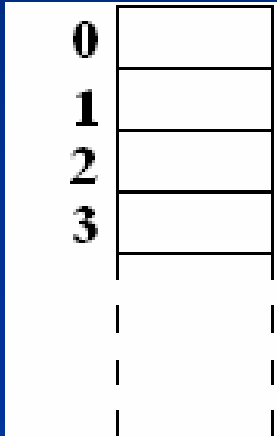
**1000100111011000**

**—> 89D8H (数据)**

**1000100111011000**

**—> MOV AX,BX (程序)**

# 存储单元

- 存储器被划分为若干个存储单元，每个存储单元从**0**开始顺序编号；
  - 例如：  
一个存储器有**128**个存储单元，  
编号从**0~127**。  
如右图示：
- 
- The diagram illustrates a vertical column of memory cells. On the left side of the column, the addresses 0, 1, 2, and 3 are listed vertically. Each address corresponds to a rectangular cell. The cells for addresses 0, 1, and 2 are empty. The cell for address 3 contains a single character, 'A'. Below the cell for address 3, there are three dashed lines, indicating that the sequence of memory cells continues beyond the shown range.

0	
1	
2	
3	
124	
125	
126	
127	

## CPU对存储器的读写

- **CPU**要想进行数据的读写，必须和外部器件（标准的说法是芯片）进行三类信息的交互：
  - 存储单元的地址（地址信息）
  - 器件的选择，读或写命令（控制信息）
  - 读或写的数据（数据信息）

## CPU对存储器的读写

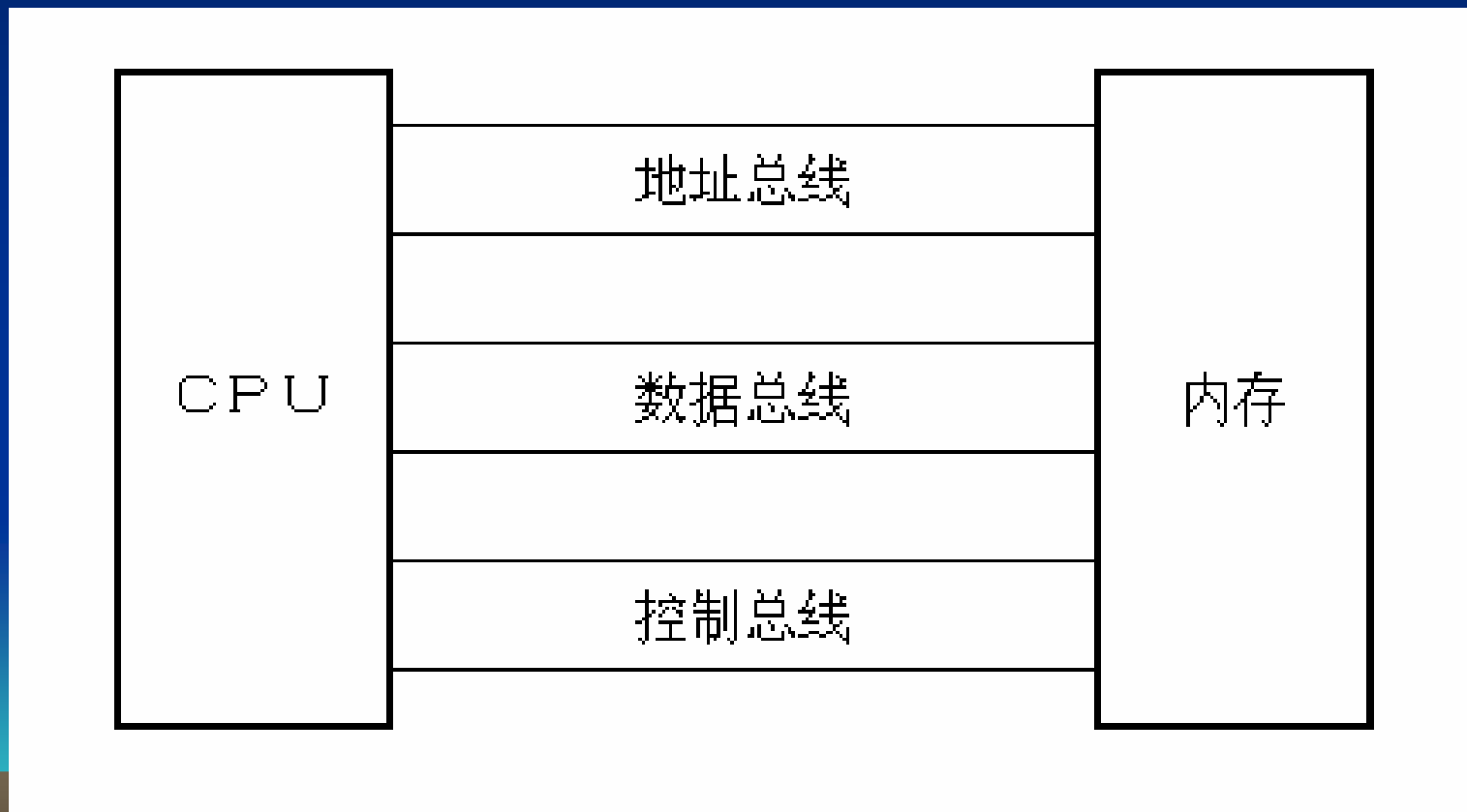
- 那么**CPU**是通过什么将地址、数据和控制信息传到存储芯片中的呢？
- 电子计算机能处理、传输的信息都是电信号，电信号当然要用导线传送。

# CPU对存储器的读写

- 在计算机中专门有连接**CPU**和其他芯片的导线，通常称为总线。
  - 物理上：一根根导线的集合；
  - 逻辑上划分为：
    - 地址总线
    - 数据总线
    - 控制总线
  - 图示

# CPU对存储器的读写

- 总线在逻辑上划分的图示：

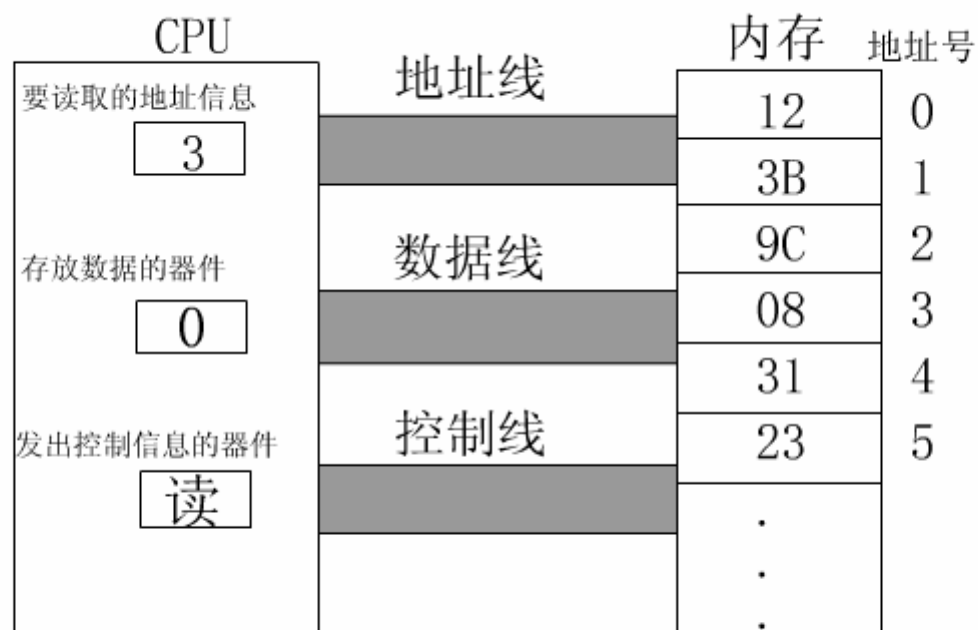




# CPU对存储器的读写

- CPU在内存中读或写的数据演示：
  - 读演示
  - 写演示
- 从上面我们知道CPU是如何进行数据读写的。可是我们如何命令计算机进行数据的读写呢？

# CPU对存储器的读写

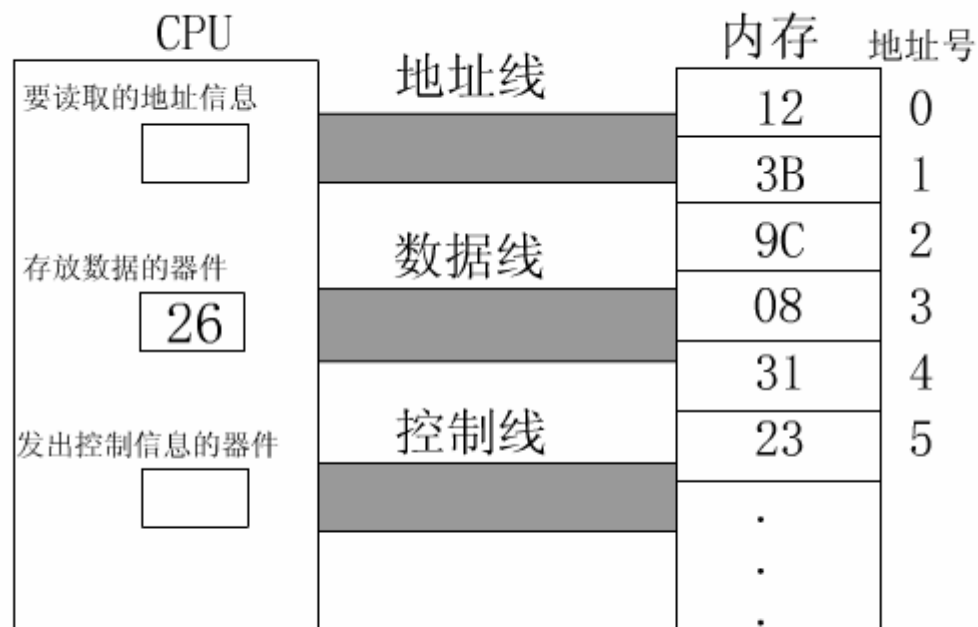


CPU从内存中3号单元处读取数据的过程

▶ play ◀ step ▶ step ◻ stop



# CPU对存储器的读写



CPU向内存中3号单元写入数据26的过程

▶ play ◀ step ▶ step ◻ stop



## CPU对存储器的读写

- 对于**8086CPU**，下面的机器码能够完成从**3号单元**读数据：
  - 机器码：**1010000000000001100000000**
  - 含义：从**3号单元**读取数据送入寄存器**AX**
  - **CPU**接收这条机器码后将完成上面所述的读写工作。

## CPU对存储器的读写

- 机器码难于记忆，用汇编指令来表示，情况如下：
  - 机器码：**1010000000000001100000000**
  - 对应的汇编指令：**MOV AX,[3]**
  - 含义：传送**3**号单元的内容到**AX**

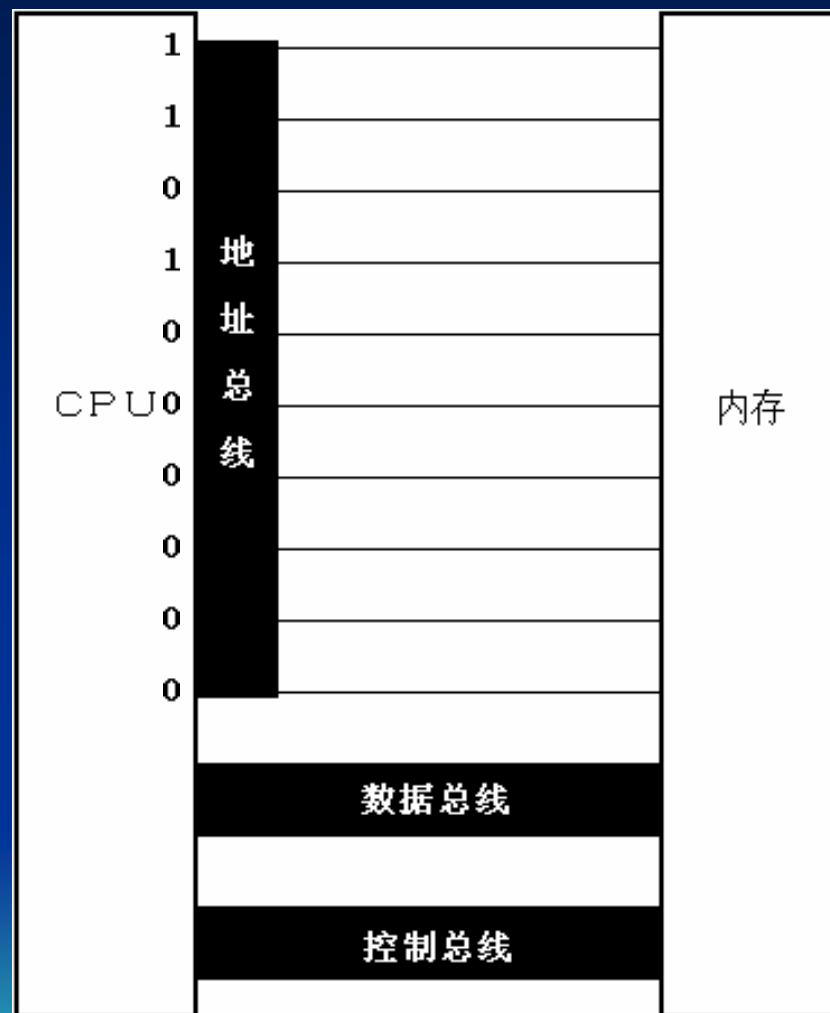
# 地址总线

- **CPU**是通过地址总线来指定存储单元的。
- 地址总线上能传送多少个不同的信息，**CPU**就可以对多少个存储单元进行寻址。

# 地址总线

- 地址总线发送地址信息演示

# 地址总线





## 地址总线

- 一根地址线可以寻址一个二进制位，是指一根地址线对应了地址总线中的一个二进制位
- 一根线上有两种状态0或1，可寻址为  $2^1 = 2$  (bit)
- 一个CPU有N根地址总线，则可以说这个CPU的地址总线的宽度为N。
- 这样的CPU最多可以寻找  $2$  的N次方个内存单元。

理解“CPU最多可以寻找  $2^N$  个内存单元”

- 因为一个单独的二进制位单元所能表示的信息太少了，所以用一个字节（**8bit**）来作为存储的最小单元
- 所以一个字节里的所有的位(**bit**)用同一个内存地址来访问
- 由于一个内存地址对应一个以字节为单位的内存单元，总线宽度为N的地址线可以表示 $2^N$ 个地址，那么总共就对应了 $2^N$ 个以字节为单位的内存单元（**8bit**）

## 数据总线

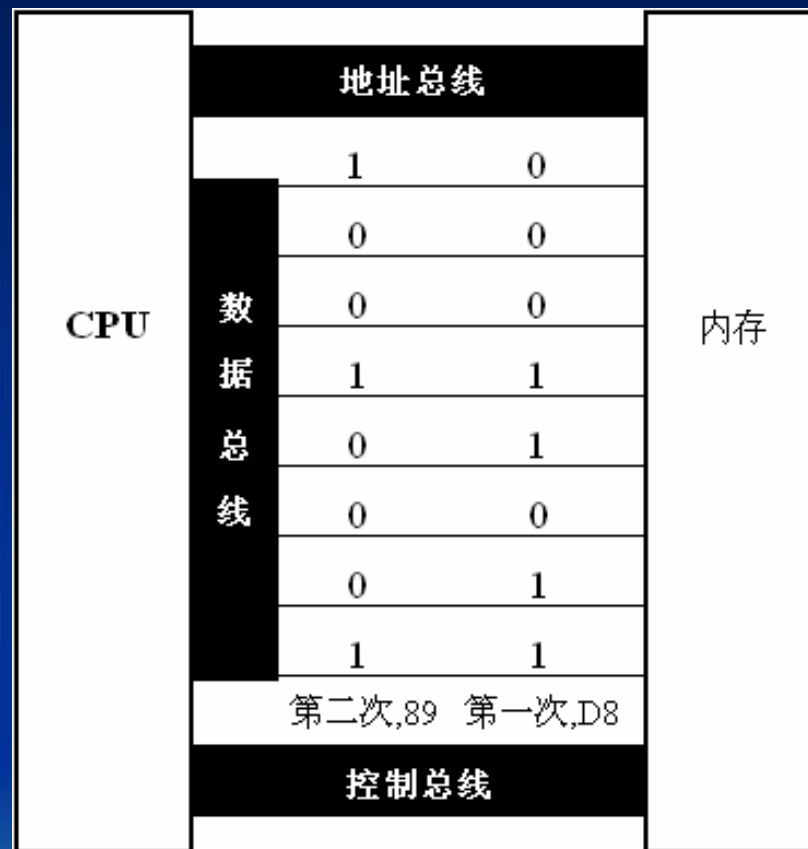
- **CPU**与内存或其它器件之间的数据传送是通过数据总线来进行的。
- 数据总线的宽度决定了**CPU**和外界的数据传送速度。

# 数据总线

- 我们来分别看一下它们向内存中写入数据 **89D8H** 时，是如何通过数据总线传送数据的：
  - 8088CPU数据总线上的数据传送情况
  - 8086CPU数据总线上的数据传送情况

# 数据总线示例

## 向内存写入数据89D8H

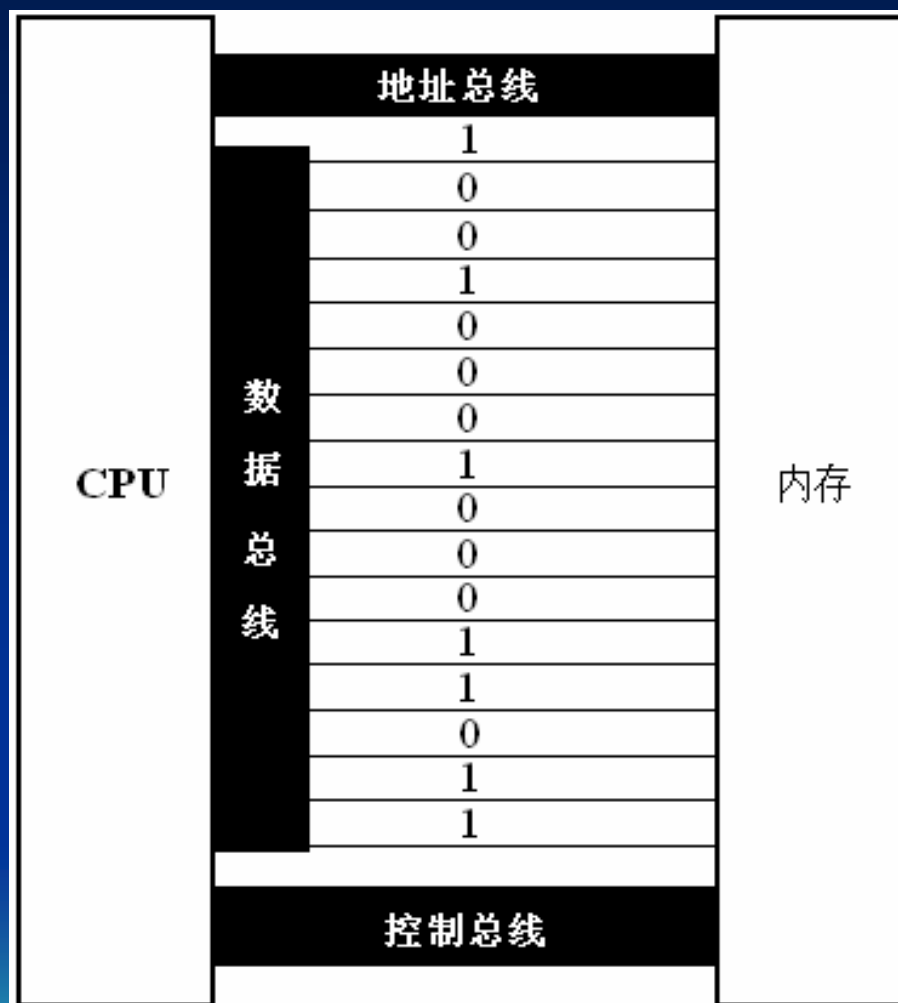


8位数据总线上传送的信息



# 数据总线示例

## 向内存写入数据89D8H



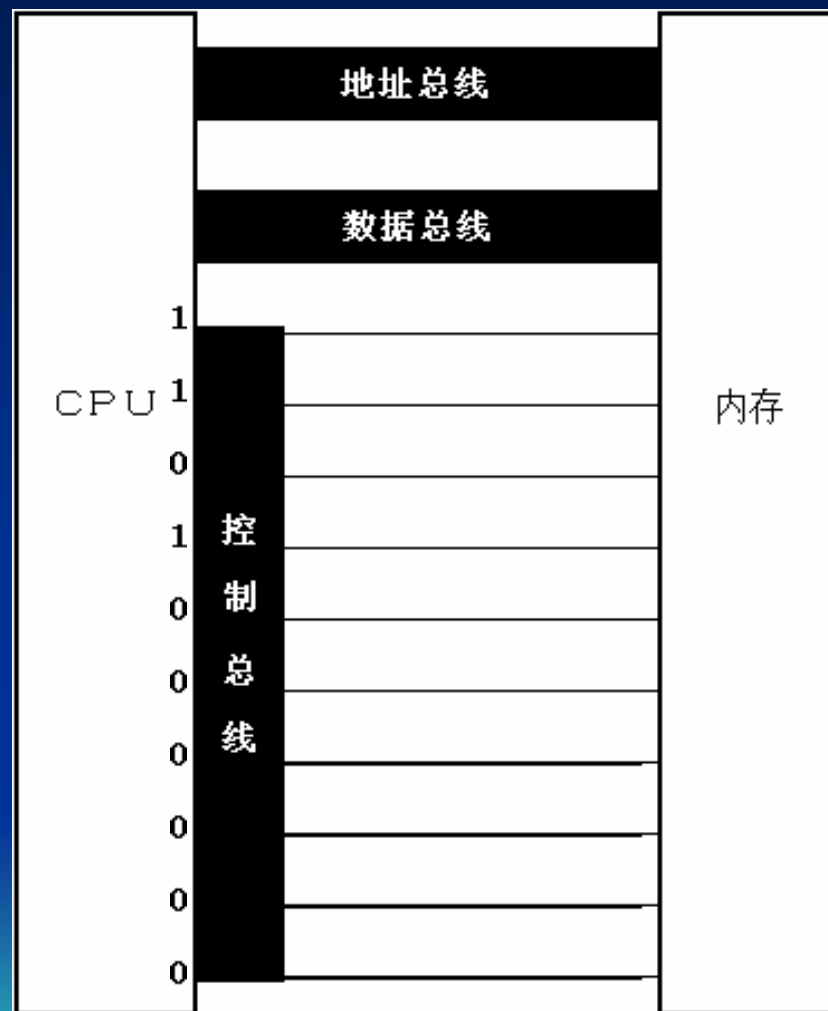
16位数据总线上传送的信息



# 控制总线

- **CPU**对外部器件的控制是通过控制总线来进行的。在这里控制总线是个总称，控制总线是一些不同控制线的集合。
- 有多少根控制总线，就意味着**CPU**提供了对外部器件的多少种控制。  
所以，控制总线的宽度决定了**CPU**对外部器件的控制能力。
- 控制总线上发送的控制信息

# 控制总线





# 控制总线

- 内存读或写命令就是由几根控制线综合发出的：
  - 其中有一根名为读信号输出控制线负责由**CPU** 向外传送读信号，**CPU** 向该控制线上输出低电平表示将要读取数据；
  - 有一根名为写信号输出控制线负责由**CPU**向外传送写信号。

# 小 结

- 一个存储单元可以存储 8 个 **bit**（用作单位写成“**b**”），即 8 位二进制数
- 一个**CPU**可以引出三种总线的宽度标志了这个**CPU**的不同方面的性能：
  - 地址总线的宽度决定了**CPU**的寻址能力；
  - 数据总线的宽度决定了**CPU**与其它器件进行数据传送时的一次数据传送量；
  - 控制总线宽度决定了**CPU**对系统中其它器件的控制能力。

## Questions

- 1个CPU的寻址能力为8KB，那么它的地址总线宽度为13。  
 $2^N = 8192 \text{ Byte}$
- 1KB的存储器有1024个存储单元。  
 $8\text{bit} = 1\text{Byte}, 1\text{KB} = 1024\text{Byte}$
- 8088的地址总线宽度为20根，则它的寻址能力为1(MB)。  
 $2^{20}$
- 80386的数据总线宽度为32根，则它一次可以传送的数据为4(B)。  
数据总线8根为8bit=1Byte，16根为16bit/8=2Byte，32根为32bit/8=4Byte
- 在存储器中，数据和程序以二进制形式存放。

## 2.1 基于微处理器的计算机系统构成 – 软件构成

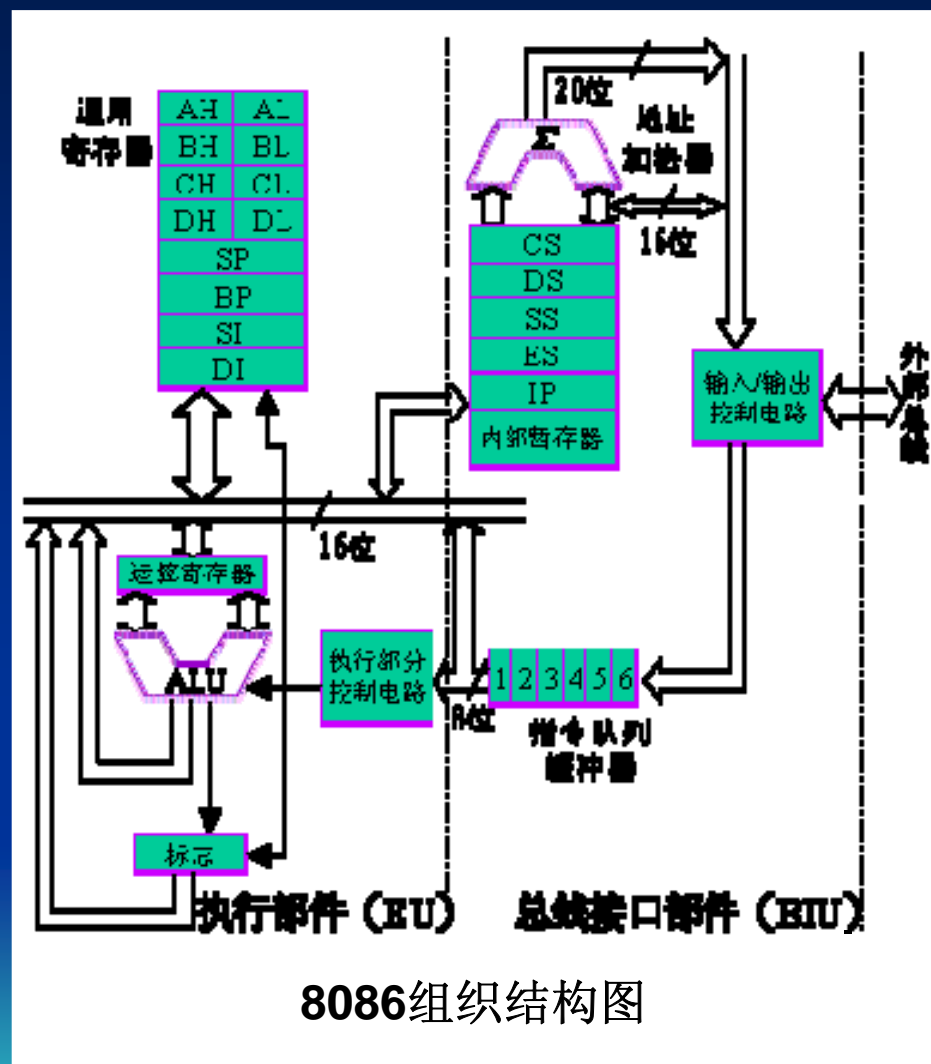
### 3. 软件系统构成

- 系统软件
  - 操作系统
    - 常驻监督程序
    - 驱动程序
    - 文件管理程序
    - 装入程序等
  - 编译、链接、调试程序
- 应用软件
  - 字处理软件、多媒体软件等

## 2.2 80X86 CPU – 结构

### 2. 80x86 CPU结构

- **BIU**: 负责与存储器、I/O 端口传送数据。
- **EU**: 负责指令的执行。
- **ALU**: 负责算术与逻辑运算。
- **寄存器**: 存储数据



## 2.2 80x86 CPU – 通用寄存器

### 3. 80x86通用寄存器

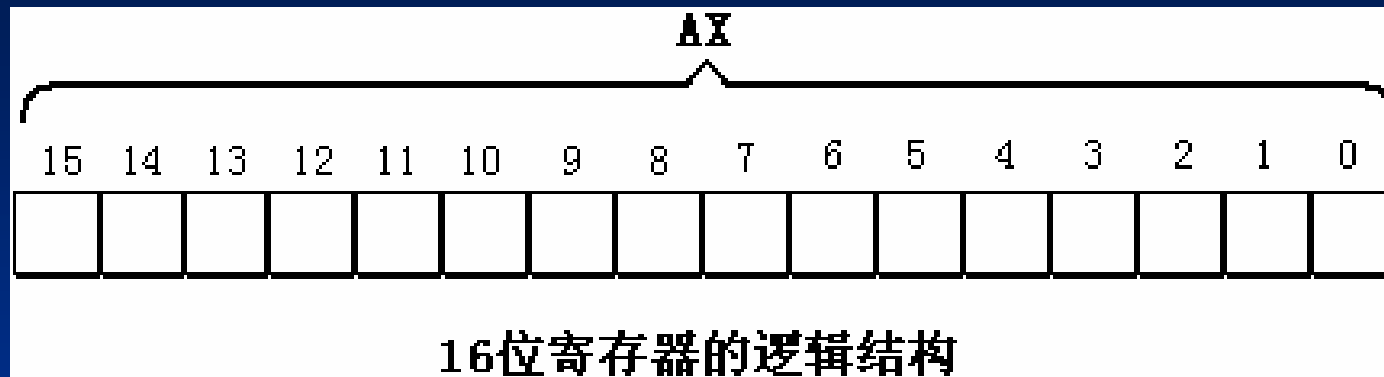
表：80x86通用寄存器

32位名称		8位名称	8位名称	名称
EAX		AH	AL	累加器(AX)
EBX		BH	BL	基址(BX)
ECX		CH	CL	计数(CX)
EDX		DH	DL	数据(DX)
ESP		SP		堆栈指针
EBP		BP		基址指针
EDI		DI		目的变址
ESI		SI		源变址

## 通用寄存器

- **8086CPU**所有的寄存器都是**16**位的，可以存放两个字节。
- **AX、BX、CX、DX** 通常用来存放一般性数据被称为通用寄存器。
- 下面以**AX**为例，寄存器的逻辑结构。

# 通用寄存器



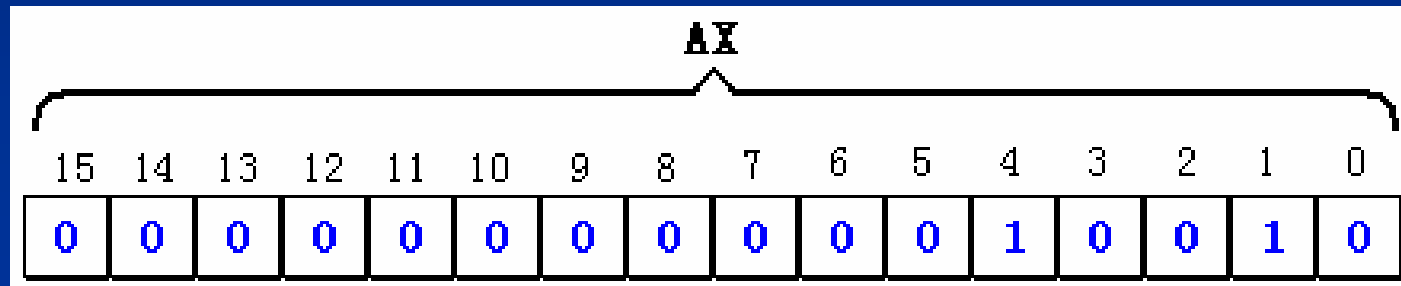
- 一个**16**位寄存器可以存储一个**16**位的数据。（[数据的存放情况](#)）
- 一个**16**位寄存器所能存储的数据的最大值为多少？

答案： **$2^{16}-1$**  ( $2^0+\dots+2^{15}=65535$ )



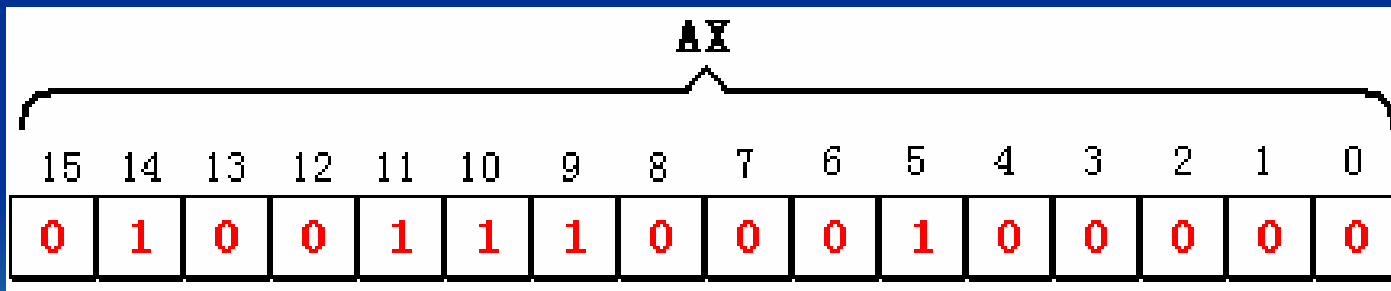
## 16位数据在寄存器中的存放情况

- 数据：18
- 二进制表示：10010
- 在寄存器AX中的存储：



## 16位数据在寄存器中的存放情况

- 数据: **20000**
- 二进制表示: **0100111000100000**
- 在寄存器**AX**中的存储:

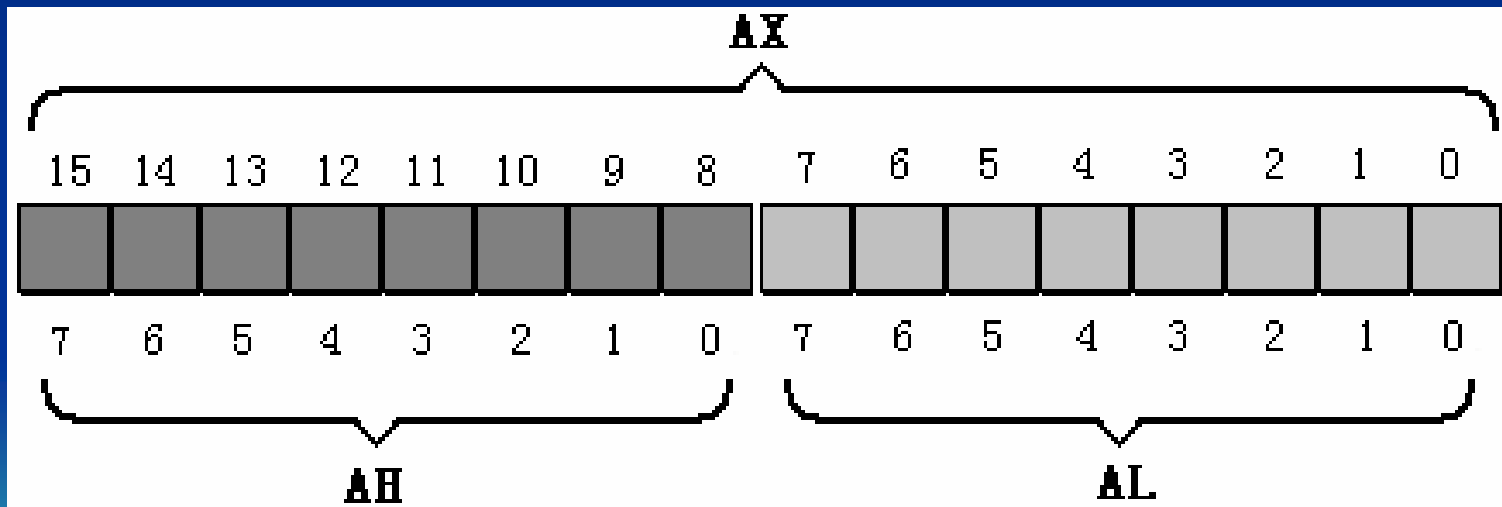


# 通用寄存器

- 8086上一代CPU中的寄存器都是8位的；
- 为保证兼容性，这四个寄存器都可以分为两个独立的8位寄存器使用。
  - AX可以分为AH和AL；
  - BX可以分为BH和BL；
  - CX可以分为CH和CL；
  - DX可以分为DH和DL。
- 8086CPU的8位寄存器存储逻辑

# 通用寄存器

- 以**AX**为例，**8086CPU**的**16位**寄存器分为两个**8位**寄存器的情况：

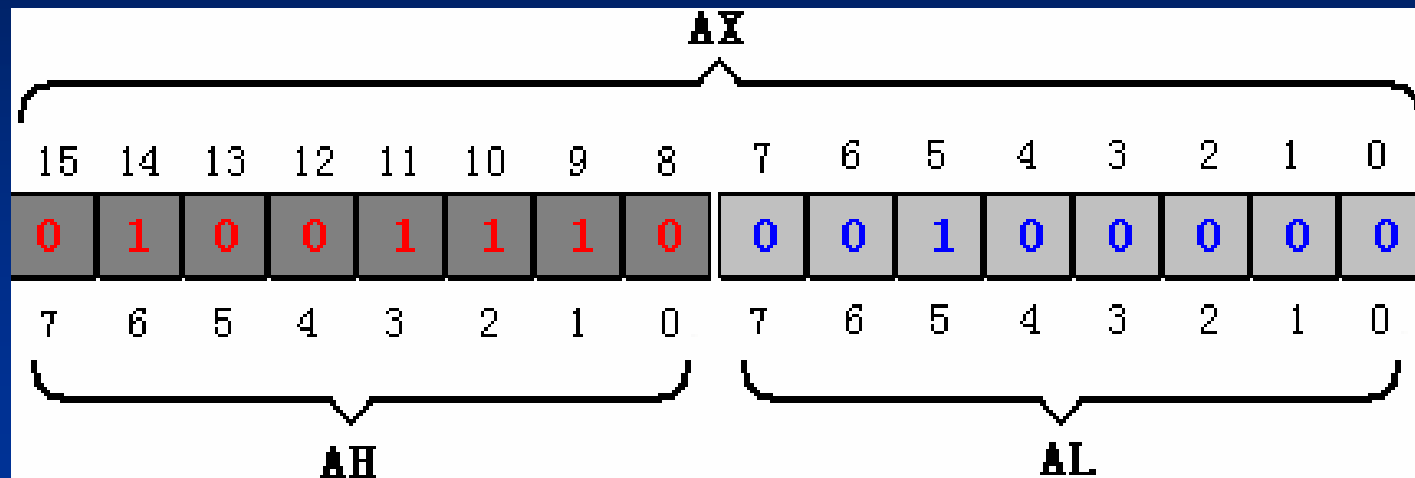


## 通用寄存器

- **AX**的低**8**位（**0**位~**7**位）构成了**AL**寄存器，高**8**位（**8**位~**15**位）构成了**AH**寄存器。
- **AH**和**AL**寄存器是可以独立使用的**8**位寄存器。
- 8086CPU的8位寄存器数据存储情况
- 一个**8**位寄存器所能存储的数据的最大值是多少？

答案： $2^8-1$ 。

# 通用寄存器



寄存器	寄存器中的数据	所表示的值
AX	0100111000100000	20000 (4E20H)
AH	01001110	78 (4EH)
AL	00100000	32 (20H)



## 字在寄存器中的存储

- 一个字可以存在一个**16**位寄存器中，这个字的高位字节和低位字节自然就存在这个寄存器的高**8**位寄存器和低**8**位寄存器中。



## 2.2 80x86 CPU – 通用寄存器

### 3. 80x86通用寄存器

- 数据寄存器：
  - 一般用于存放数据，包括**AX**、**BX**、**CX**、**DX**，可以分别访问其低端字节**AL**、**BL**、**CL**、**DL**和高端字节**AH**、**BH**、**CH**、**DH**，从**386**起扩充为**32**位，分别是**EAX**、**EBX**、**ECX**、**EDX**。
- 指针和变址寄存器：
  - 一般用来存放地址的偏移量，包括指针寄存器**SP**、**BP**和变址寄存器**SI**、**DI**，从**386**起扩充为**32**位，分别是**ESP**、**EBP**、**ESI**、**EDI**。



## 2.2 80x86 CPU – 通用寄存器

### 3. 80x86通用寄存器

- 通用寄存器具有通用性，但也有特定或隐含的用法。

寄存器	执 行 操 作
<b>AX</b>	整字乘法，整字除法，整字I/O。
<b>AL</b>	字节乘法，字节除法，字节I/O。翻译，十进制算术运算
<b>AH</b>	字节乘法，字节除法。
<b>BX</b>	翻译。
<b>CX</b>	字符串操作，循环。
<b>CL</b>	变量的移位和循环移位。
<b>DX</b>	整字乘法，整字除法，间接I/O。
<b>SP</b>	堆栈操作。
<b>SI</b>	字符串操作。
<b>DI</b>	字符串操作。

## 2.2 80x86 CPU – 指令指针寄存器 IP

### 4. 80x86指令指针寄存器IP:

- 指向下一条指令的首地址（代码段中的偏移）
- 控制器的取指单元根据IP的值，从存储器中读取下一条指令，同时修改IP的值。
- 从386起扩充为32位，称为EIP。

表：80x86指令指针寄存器

32位名称

EIP

16位名称

IP

名称

指令指针

## 2.2 80x86 CPU - PSW

### 5. 80x86程序状态与控制寄存器

- 又称标志寄存器**FLAGS**、程序状态字**PSW**。
- 用于存储**条件码**标志、**控制**标志、**系统**标志。
- 从**386**起扩充为**32**位，称为**EFLAGS**。
- 不同的**CPU**版本，标志的定义有所不同。

表：8086/8088/80286 标志寄存器

比特		14	13	12	11	10	9	8	7	6		4		2		0
标志		NT	IOPL		OF	DF	IF	TF	SF	ZF		AF		PF		CF

## 2.2 80x86 CPU – PSW: 条件标志

### 5. 80x86程序状态与控制寄存器

- 条件码标志:

- 记录程序运行结果的状态信息, 由**CPU**自动设置。
- 溢出标志**OF**: 1表示运算结果溢出。
- 符号标志**SF**: 1表示运算结果为负数。
- 零标志**ZF**: 1表示运算结果为0。
- 进位标志**CF**: 1表示运算时最高位有进位。
- 辅助进位标志**AF**: 1表示运算时第3位有进位
- 奇偶标志**PF**: 1表示运算结果中有偶数个1。

## 2.2 80x86 CPU – PSW: 控制、系统标志

### 5. 80x86程序状态与控制寄存器

- 控制标志——方向标志**DF**:
  - 在串处理指令中控制处理信息的方向。
  - 当**DF==1**时，每次操作后变址寄存器**SI**和**DI**减小，使串处理从高地址向低地址方向处理。
  - 当**DF==0**时，**SI**和**DI**递增，由低地址向高地址处理。
- 系统标志:
  - 陷阱标志**TF**: 用于调试时的单步执行方式。
  - 中断标志**IF**: 1表示允许CPU响应可屏蔽中断请求。
  - I/O特权级**IOPL**: 在保护模式下，控制对I/O地址空间的访问。

## 2.2 80x86 CPU - PSW

### 5. 80x86程序状态与控制寄存器

- 在调试程序**Debug**中提供了测试标志位的手段，它用符号表示某些标志位的值。

表：Debug 程序标志位符号表示

标志	解释	标志为1	标志为0
<b>OF</b>	溢出（是/否）	<b>OV</b>	<b>NV</b>
<b>DF</b>	方向（减量/增量）	<b>DN</b>	<b>UP</b>
<b>IF</b>	中断（允许/禁止）	<b>EI</b>	<b>DI</b>
<b>SF</b>	符号（负/正）	<b>NG</b>	<b>PL</b>
<b>ZF</b>	结果为零（是/否）	<b>ZR</b>	<b>NZ</b>
<b>AF</b>	辅助进位（是/否）	<b>AC</b>	<b>NA</b>
<b>PF</b>	奇偶（偶/奇）	<b>PE</b>	<b>PO</b>
<b>CF</b>	进位（是/否）	<b>CY</b>	<b>NC</b>

## 2.2 80x86 CPU – 段寄存器

### 6. 80x86段寄存器

- 直接或间接存放段基址，**16**位长。
- 代码段寄存器**CS**：存储当前代码段的基地址。
- 数据段寄存器**DS**：存储数据段基地址。
- 附加段寄存器**ES**：存储附加数据段基地址。
- 堆栈段寄存器**SS**：存储堆栈段基地址。
- **FS**、**GS**寄存器：附加数据段，从**386**起增添。

## 2.3 80x86 CPU的存储器管理 – 存储单元地址

### 1. 存储单元的地址

- 基于**80x86CPU**的**IBM PC**机以**字节**为单位存储信息，即每个存储单元存储一个字节的数据。
- 每个存储单元被分配一个**唯一**的地址。
- 物理地址从**0**开始，顺序编号。
- 若**CPU**地址总线宽度为**N**，则可访问的字节单元地址范围为  **$0 \sim 2^N - 1$** 。
- **8086/8088**的地址总线宽度为**20**位，可访问的地址空间为 **$0 \sim \text{FFFFFFH}$ （1MB）**。
- **80286**的地址总线宽度为**24**位，**386~Pentium**地址总线宽度为**32**位（**P15**）。



## 内存中字的存储

2000的十六进制是？

- 在0地址处开始存放20000: **4E20H**
- 0号单元是低地址单元，1号单元是高地址单元。

在存储器中如何存放？

接下来存放数据18

0	20H
1	4EH
2	12H
3	00H
4	
5	

内存中字的存储

5	
4	
3	00H
2	12H
1	4EH
0	20H

内存中字的存储

- 问题:

- (1) 0地址单元中存放的字节型数据是多少? 20H
- (2) 0地址字单元中存放的字型数据是多少? 4E20H
- (3) 2地址字单元中存放的字节型数据是多少? 12H
- (4) 2地址单元中存放的字型数据是多少? 0012H
- (5) 1地址字单元中存放的字型数据是多少? 124EH

0	20H
1	4EH
2	12H
3	00H
4	
5	

内存中字的存储

- 结论：
  - 任何两个地址连续的内存单元，**N**号单元和**N+1**号单元，可以将它们看成两个内存单元，也可以看成一个地址为**N**的字单元中的高位字节单元和低位字节单元。

## 2.3 80x86 CPU的存储器管理 – 基本数据类型

### 2. 存储单元的内容

- 数据类型及存储顺序:

- **位**（比特、**bit**）
- **字节**：**8**位，可存储于任何单元。
- **字**：**16**位，**2**字节，低位字节在低地址单元，高位字节在高地址，起始地址最好是偶地址。
- **双字**：**32**位，低位字存入低地址，高位字存入高地址，起始地址最好是**4**的倍数。
- **4字**：**64**位，低位双字存入低地址，高位双字存入高地址，起始地址最好是**8**的倍数。
- 例1：1号字节单元内容为**78H**。
- 例2：1号字单元内容为**5678H**。
- 例3：1号双字单元内容为**12345678H**

字节 地址

	00H
78H	01H
56H	02H
34H	03H
12H	04H
	05H
	06H
	07H
	08H
	09H
	0AH
...	...

同一地址可看作字节、字、双字、4字单元  
约定：

X表示单元地址，则 (X) 表示X单元的内容

如右图：

**X = 0004H**

**(X) = 5678H**

**((X)) = 2F1EH**

字节 地址

	00H
78H	0004H
56H	0005H
34H	0006H
12H	0007H
1EH	5678H
2FH	5679H
	567AH
	567BH
...	...

## 2.3 80x86 CPU的存储器管理 – 三种工作模式

### 3. 实模式存储器寻址

#### – 80x86CPU工作模式概述：

- 实（地址）模式
- 保护（虚拟地址）模式
- 虚拟86模式

表： 80x86CPU支持的工作模式

工作模式	8086/8088	286	386及后续CPU
实模式	√	√	√
保护模式	×	√	√
虚拟86模式	×	×	√

## 物理地址

- **CPU**访问内存单元时要给出内存单元的地址。所有的内存单元构成的存储空间是一个一维的线性空间。
- 每一个内存单元在这个空间中都有唯一的地址，这个唯一的地址称为物理地址。
- **CPU**访问存储器时，必须先确定所要访问的存储单元的地址

## 16位结构的CPU

- 概括的讲，**16位结构**描述了一个**CPU**具有以下  
几个方面特征：
  - **1**、运算器一次最多可以处理**16位**的数据。
  - **2**、寄存器的最大宽度为**16位**。
  - **3**、寄存器和运算器之间的通路是**16位**的。



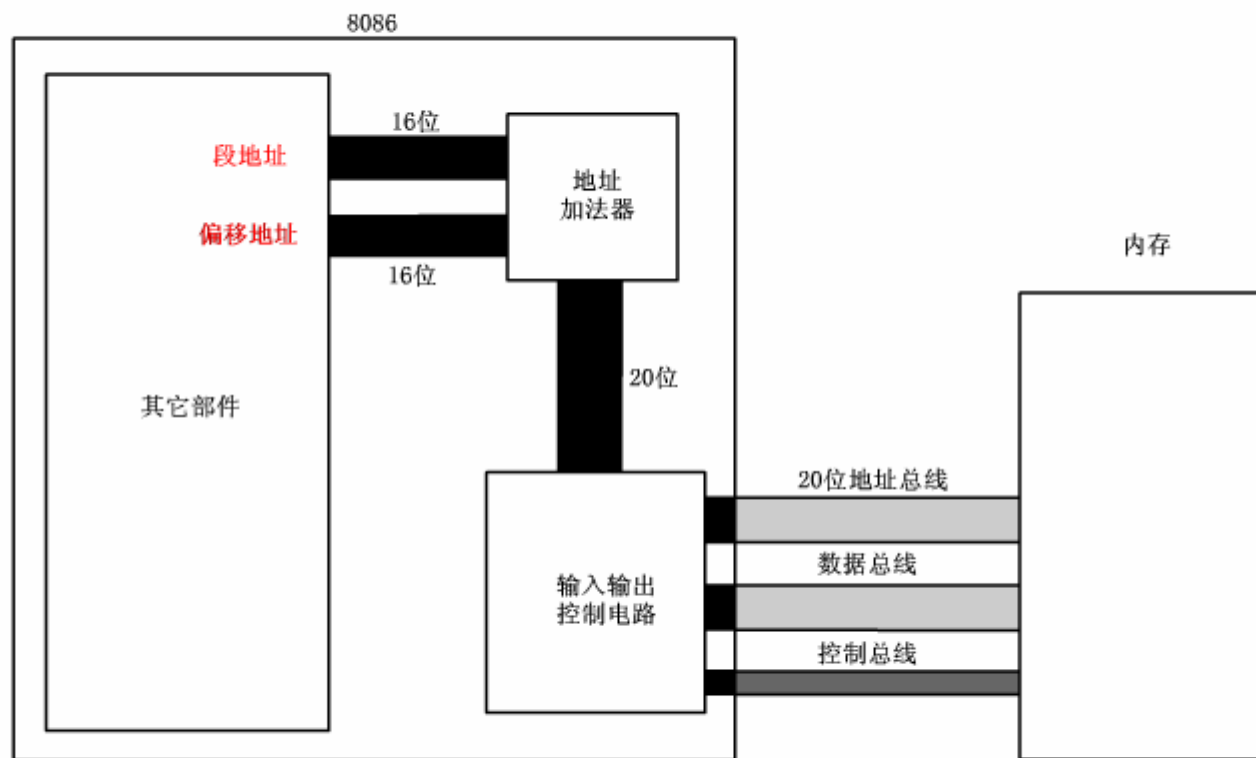
## 8086CPU给出物理地址的方法

- **8086有20位地址总线，可传送20位地址，寻址能力为1M。**
- **8086内部为16位结构，它只能传送16位的地址，表现出的寻址能力只有64K。**

# 8086CPU给出物理地址的方法

- **8086CPU**采用一种在内部用两个**16**位地址合成的方法来形成一个**20**位的物理地址。

# 8086CPU相关部件的逻辑结构



8086CPU给出物理地址的方法

play step step stop

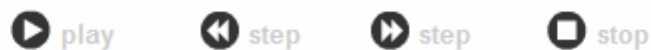
## 地址加法器

- 地址加法器合成物理地址的方法：  
物理地址 = 段地址  $\times$  16 + 偏移地址

# 8086CPU访问地址为123C8H的内存单元



地址加法器的工作过程



## 由段地址 $\times 16$ 引发的讨论

移位位数	二进制	十六进制	十进制
0	10B	2H	2
1	100B	4H	4
2	1000B	8H	8
3	10000B	10H	16
4	100000B	20H	32

- 观察移位次数和各种形式数据的关系：
  - 一个数据的二进制形式左移1位，相当于该数据乘以2；
  - 一个数据的二进制形式左移N位，相当于该数据乘以2的N次方；
  - 地址加法器如何完成段地址 $\times 16$ 的运算？

以二进制形式存放的段地址左移4位。

“段地址 $\times 16$ +偏移地址=物理地址”的本质含义

- 两个比喻说明：
  - 说明“基础地址+偏移地址 = 物理地址”的思想：第一个比喻
  - 说明“段地址 $\times 16$ +偏移地址=物理地址”的思想：第二个比喻

**8086CPU**就是这样一个只能提供两张3位数据纸条的**CPU**。

“基础地址+偏移地址 = 物理地址”



- 比如说，学校、体育馆同在一条笔直的单行路上（学校位于路的起点**0**米处）。
- 读者在学校，要去图书馆，问我那里的地址，我可以用几种方式描述这个地址？



“基础地址+偏移地址 = 物理地址”



- (1) 从学校走**2826m**到图书馆。这**2826**可以认为是图书馆的物理地址。
- (2) 从学校走**2000m**到体育馆，从体育馆再走**826m**到图书馆。
  - 第一个距离**2000m**是相对于起点的基础地址；
  - 第二个距离**826m**是将对于基础地址的偏移地址。



“段地址 $\times 16$ +偏移地址=物理地址”

- 比如我们只能通过纸条来通信，读者问我图书馆的地址，我只能将它写在纸上告诉读者。
- 显然我必须有一张可以容纳 4 位数据的纸条才能写下**2826**这个数据：

可以写下四位数据的纸条

2	8	2	6
---	---	---	---

“段地址 $\times 16$ +偏移地址=物理地址”

- 不巧的是，没有能容纳4位数据的纸条，仅有两张可以容纳3位数据的纸条。
- 这样我只能以这种方式告诉读者**2826**这个数据：

两张可以写下3位数据的纸条

2	0	0
---	---	---

8	2	6
---	---	---



# 段的概念

- 错误认识：
  - 内存被划分成了一个一个的段，每一个段有一个段地址。
- 其实：
  - 内存并没有分段，段的划分来自于**CPU**，由于**8086CPU**用“（段地址×16）+偏移地址=物理地址”的方式给出内存单元的物理地址，使得我们可以用分段的方式来管理内存。

## 2.3 80x86 CPU的存储器管理 – 存储器分段

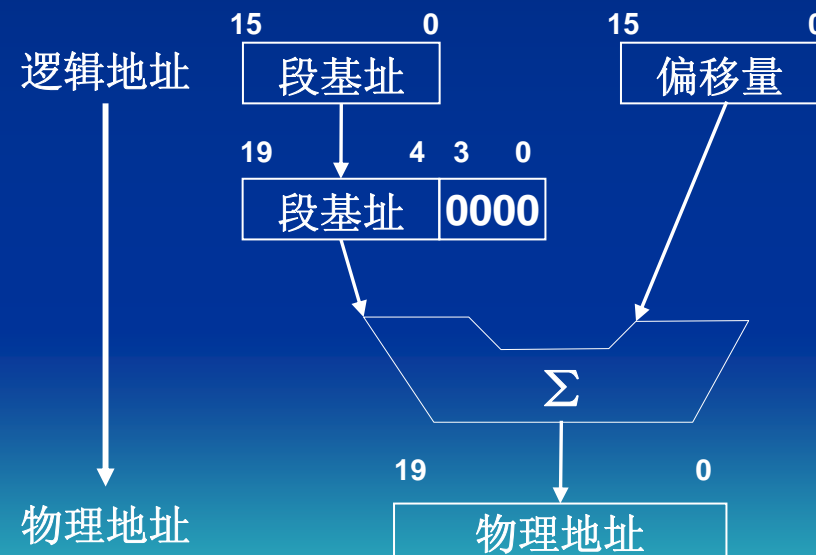
### 3. 实模式存储器寻址

- **段Segment**是按照程序的逻辑结构划分成的多个相对独立的部分。
  - 程序段可以是主程序、子程序、数据块、数组、表格等。
- 存储地址的分段
  - 实地址模式，**CPU**可访问的最大存储容量为**1MB**，需**20**位地址信号。
  - **1MB**的地址空间被划分为大小不等的段，每段最大长度为**64KB**，段基址为**16**的整数倍。

## 2.3 80x86 CPU的存储器管理 – 实模式物理地址计算

### 3. 实模式存储器寻址

- 逻辑地址的格式：段基址:段内偏移量
- 物理地址 = 段基址  $\times 16$  + 偏移量
  - 即段基址左移4位，与段内偏移求和。



参见：保护模式存储器寻址

## 2.3 80x86 CPU的存储器管理 – 段寄存器

### 3. 实模式存储器寻址

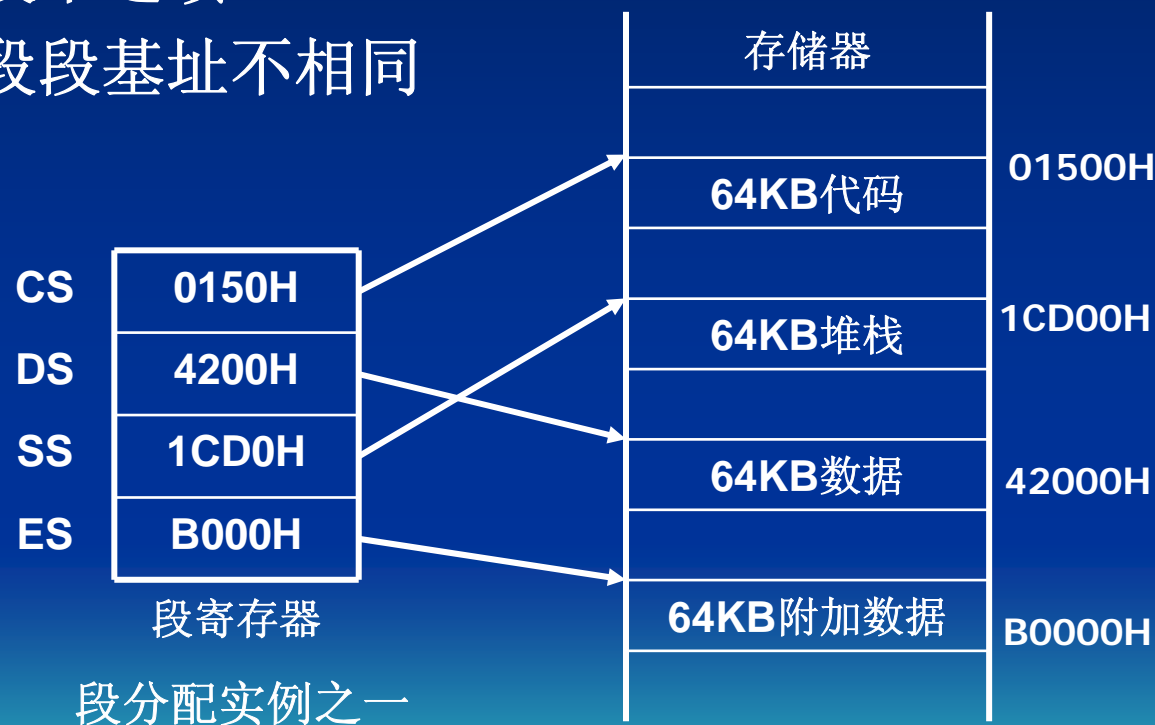
- 段寄存器：专门存放段基址的寄存器。
  - **CS**代码段、**DS**数据段、**ES**附加数据段、**SS**堆栈段
- 偏移量由**BX**、**BP**、**IP**、**SP**、**SI**、**DI**或根据寻址方式计算出的有效地址**EA**（**Effective Address**）提供。
- 注意事项：
  - 每个存储单元有唯一的物理地址，但可由不同逻辑地址表示。
  - 例：逻辑地址“**1200H:0345H**”和“**1100H:1345H**”，都对应于物理地址**12345H**。
  - 除非专门指定，一般情况下，段在存储器中的分配（即段基址的设置）是由操作系统负责的。

## 2.3 80x86 CPU的存储器管理 – 段分配实例之一

### 3. 实模式存储器寻址

#### – 段分配实例之一：

- 各段不连续
- 各段段基址不相同



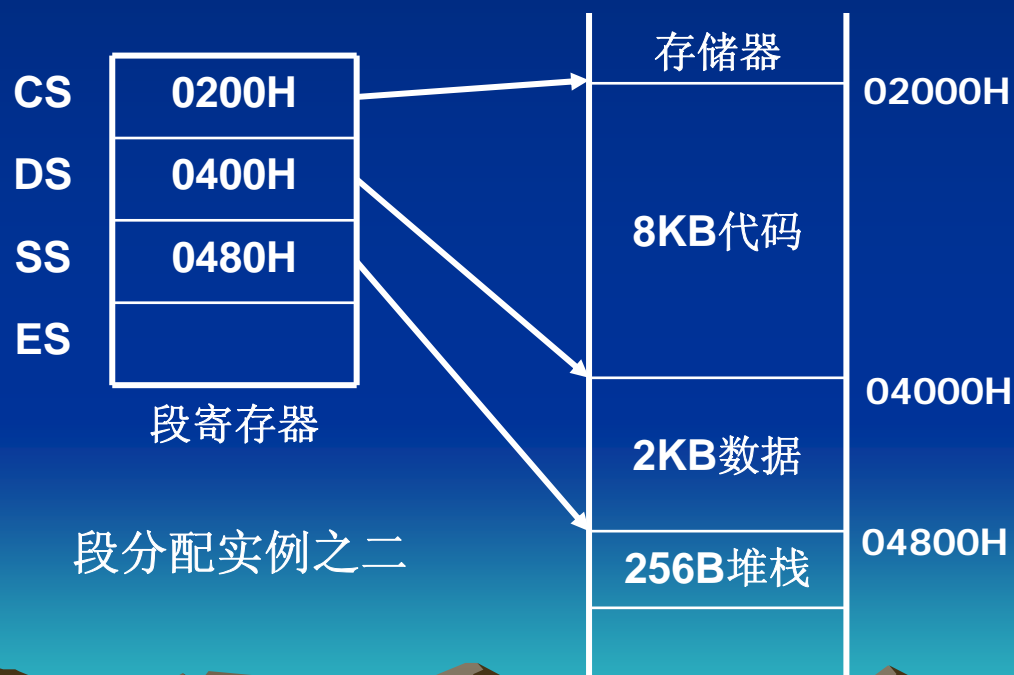


## 2.3 80x86 CPU的存储器管理 – 段分配实例之二

### 3. 实模式存储器寻址

#### — 段分配实例之二：（400H=1K，2000H=8K）

- 各段连续
- 各段段基址不同

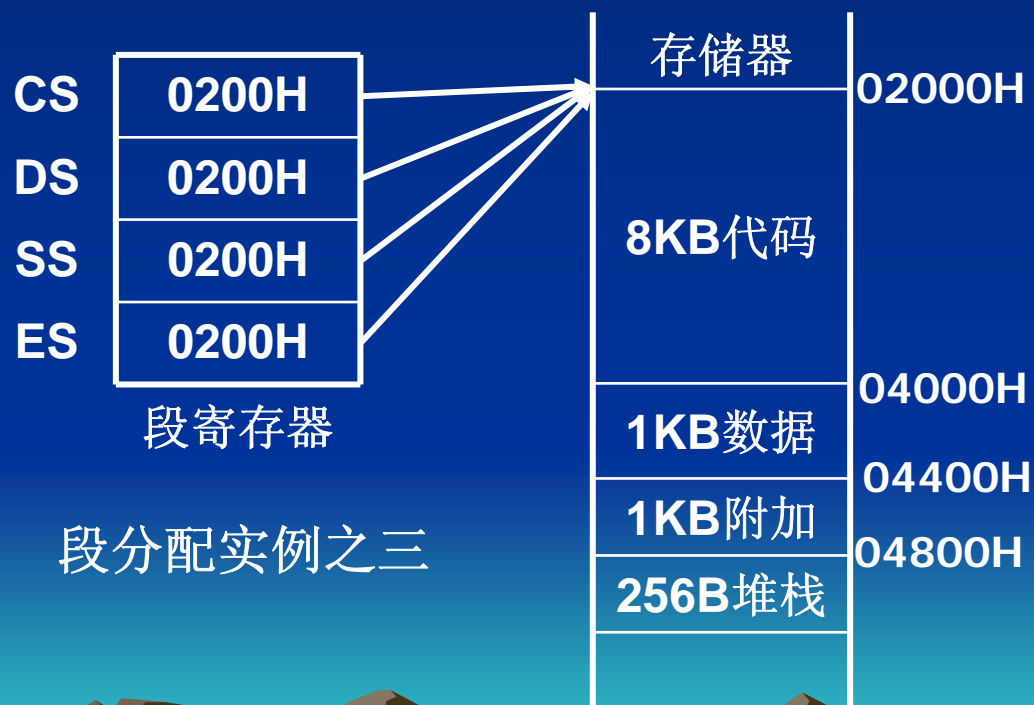


## 2.3 80x86 CPU的存储器管理 – 段分配实例之三

### 3. 实模式存储器寻址

#### – 段分配实例之三

- 各段连续
- 各段段基址相同



## 内存单元地址

- (1) 观察下面的地址，读者有什么发现？

物理地址	段地址	偏移地址
<b>21F60H</b>	<b>2000H</b>	<b>1F60H</b>
	<b>2100H</b>	<b>0F60H</b>
	<b>21F0H</b>	<b>0060H</b>
	<b>21F6H</b>	<b>0000H</b>
	<b>1F00H</b>	<b>2F60H</b>

- 结论：**CPU**可以用不同的段地址和偏移地址形成同一个物理地址。

## 内存单元地址

- (2) 如果给定一个段地址，仅通过变化偏移地址来进行寻址，最多可以定位多少内存单元？
  - 结论：偏移地址**16**位，变化范围为**0~FFFFH**，仅用偏移地址来寻址最多可寻**64K**个内存单元。
  - 比如：给定段地址**1000H**，用偏移地址寻址，**CPU**的寻址范围为：**10000H~1FFFFH**。

## 内存单元地址

- 在**8086PC**机中，存储单元的地址用两个元素来描述。即段地址和偏移地址。
- “数据在**21F60H**内存单元中。”对于**8086PC**机的两种描述：
  - (a) 数据存在内存**2000:1F60**单元中；
  - (b) 数据存在内存的**2000**段中的**1F60H**单元中。
- 可根据需要，将地址连续、起始地址为**16**的倍数的一组内存单元定义为一个段。



## 2.3 80x86 CPU的存储器管理 – 保护模式存储管理概述

### 4. 保护模式存储器寻址

- **80x86的地址线条数及最大寻址空间**
  - **8086/8088: 20根, 1MB空间**
  - **286: 24根, 16MB空间**
  - **386及后续CPU: 32根, 4GB**
- **保护模式:**
  - 解决了实模式只能寻址**1MB**的问题。
  - **段页式**存储管理, 每段最大**4GB**, 每段划分为等长的页 (**4KB**)。
  - 支持多任务处理。
  - 保护模式下的三种地址:
    - 虚拟地址 (逻辑地址)、线性地址、物理地址

实模式只能访问  
**1MB**的存储空间!!

## 2.3 80x86 CPU的存储器管理 – 逻辑地址

### 4. 保护模式存储器寻址

- 逻辑地址

- 实模式逻辑地址:

- 地址形式: **段基址:偏移量**
    - 段基址: 存放于段寄存器中。

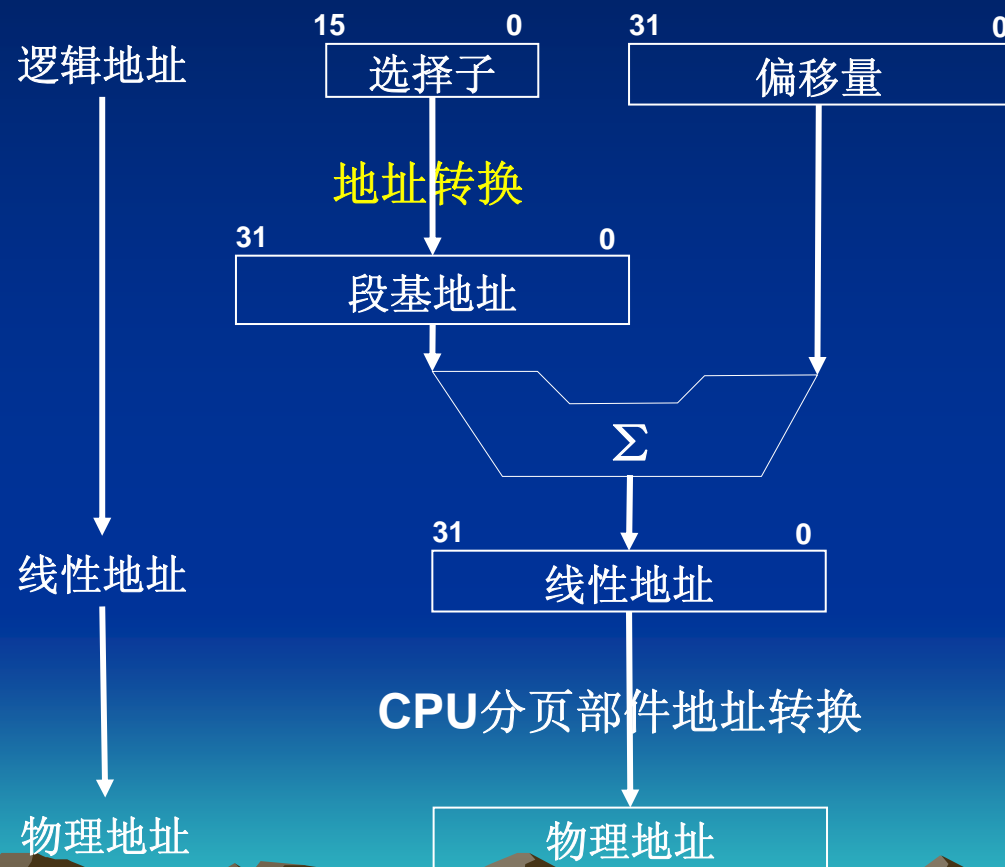
- 保护模式逻辑地址:

- 地址形式: **选择器 (选择子):偏移量**
    - 选择子: 存放于段寄存器中, **16**位长, 不能直接表示段基地址, 而是由操作系统根据复杂算法获得**32**位段基地址。
    - 偏移量: **32**位长, 即保护模式每段最长可达**4GB**.

## 2.3 80x86 CPU的存储器管理 – 保护模式存储器寻址

### 4. 保护模式存储器寻址

– 逻辑地址 → 线性地址 → 物理地址



参见：实模式存储器寻址



## 2.3 80x86 CPU的存储器管理 – 段描述符

### 4. 保护模式存储器寻址

- 段描述符：
  - 描述段的**大小**、**位置**、**控制**与**状态**信息。
  - 由基地址、界限、访问权限、附加字段组成
    - 基地址：指定段的起始地址。
    - 界限：指定段的长度。
  - 保护模式中，每个段都有一个段描述符。
  - 逻辑地址→线性地址：
    - ① 根据段选择子找到段描述符；
    - ② 从段描述符中提取段基地址与长度；
    - ③ 段基址与偏移量求和，得线性地址。

## 2.4 外部设备 – 外设接口寄存器

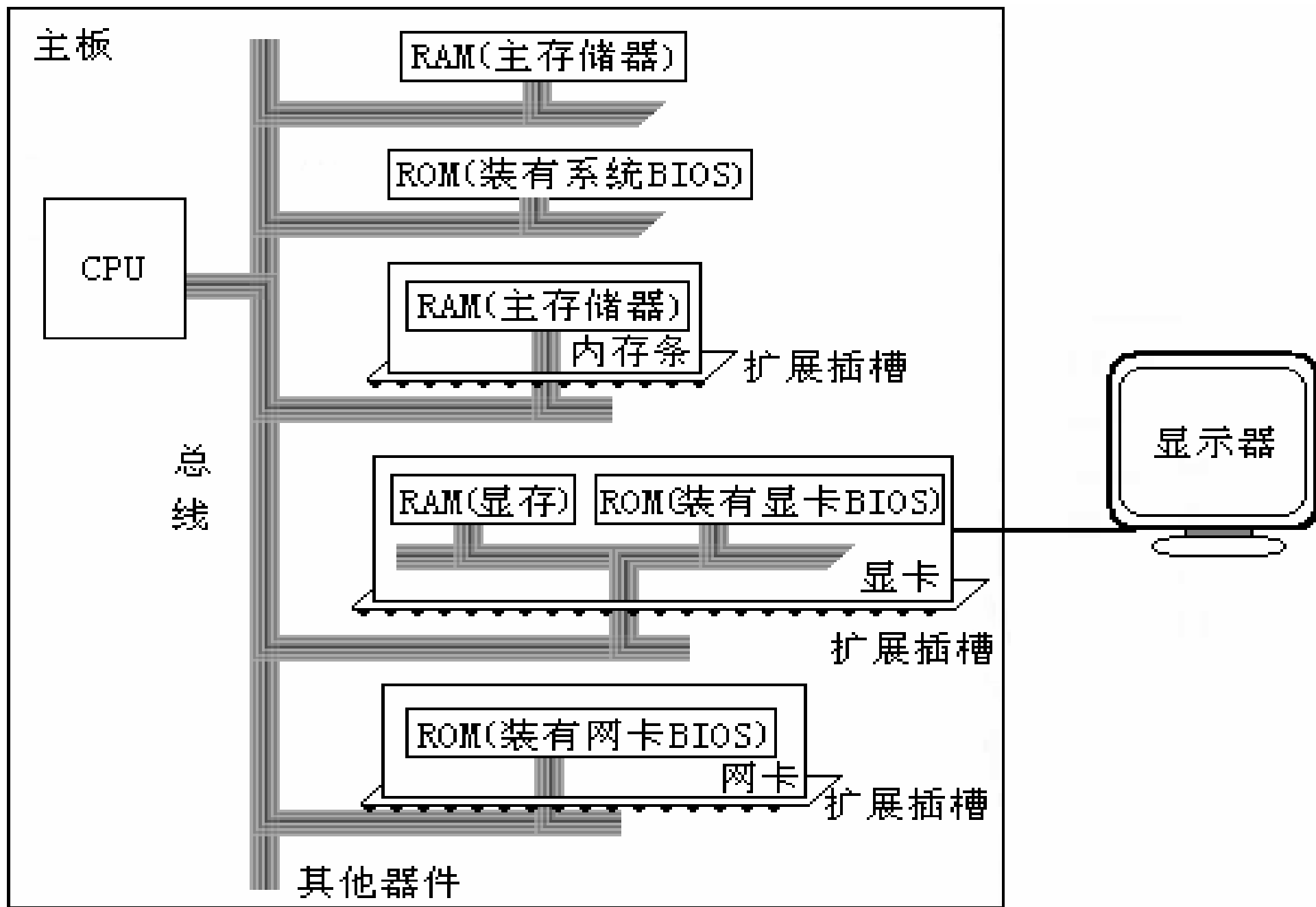
- 在每一台**PC**机中，都有一个主板，主板上  
有核心器件和一些主要器件。
- 这些器件通过总线（地址总线、数据总  
线、控制总线）相连。

## 接口卡

- 计算机系统中，所有可用程序控制其工作的设备，必须受到**CPU**的控制。
- **CPU**对外部设备不能直接控制，如显示器、音箱、打印机等。直接控制这些设备进行工作的是插在扩展插槽上的接口卡。

## 各类存储器芯片

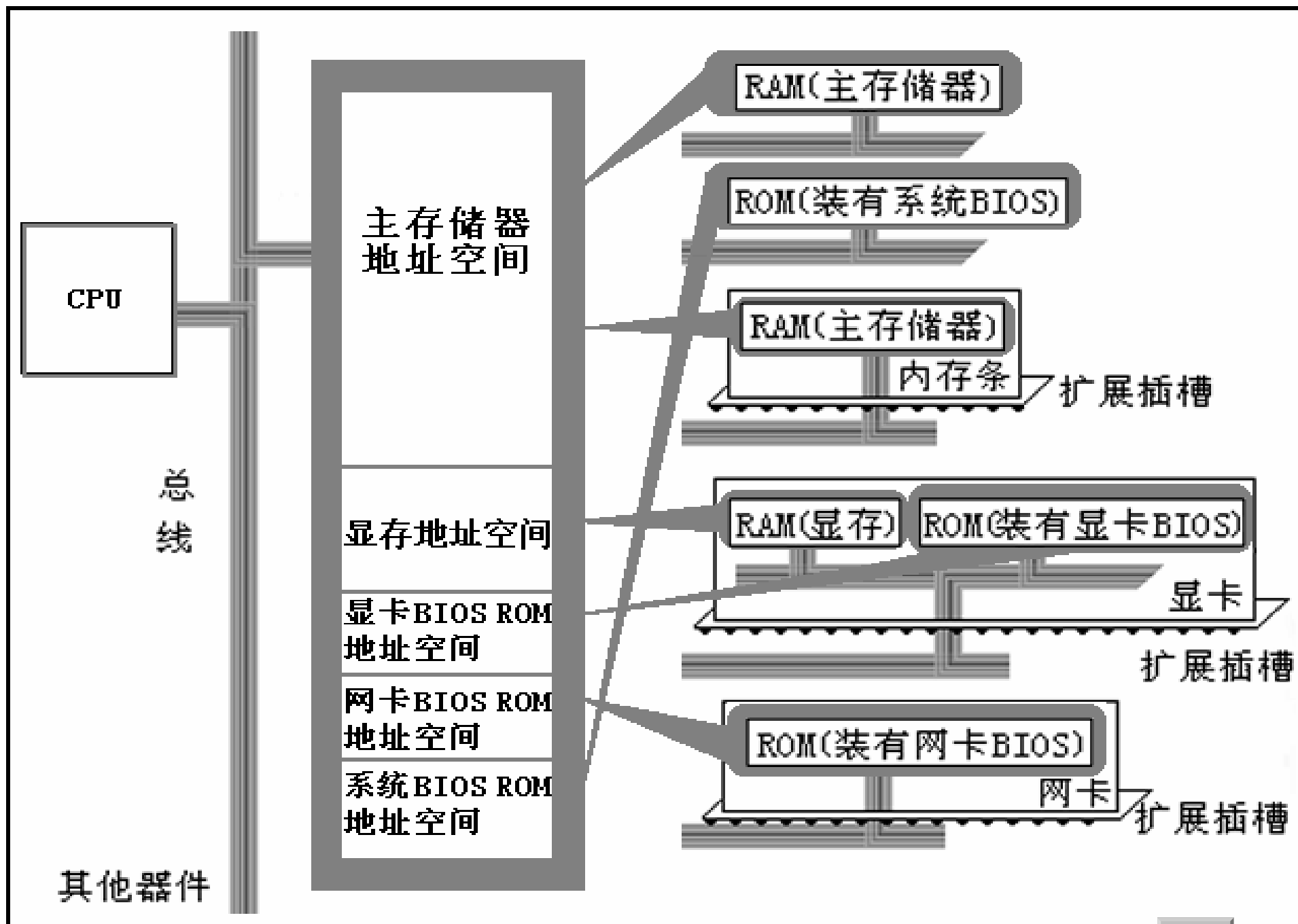
- 从读写属性上看分为两类：  
随机存储器（**RAM**）和只读存储器（**ROM**）
- 从功能和连接上分类：
  - 随机存储器**RAM**
    - 用于存放供**CPU**使用的绝大部分程序和数据
    - 有装在主板上的**RAM**和插在扩展槽上的**RAM**
  - 装有**BIOS**的**ROM**（主板、显卡、网卡的**BIOS**）
    - 主板和各类接口卡厂商提供的软件系统
    - 通过它利用该硬件设备进行最基本的输入输出
  - 接口卡上的**RAM**（如显卡上的**RAM**----显存）
- **PC**机中各类存储器的逻辑连接情况



PC集中各类存储器的逻辑连接

- 上述的那些存储器在物理上是独立的器件。
- 但是它们在以下两点上相同：
  - 1、都和**CPU**的总线相连。
  - 2、**CPU**对它们进行读或写的时候都通过控制线发出内存读写命令。

- 将各类存储器看作一个逻辑存储器：
  - 所有的物理存储器被看作一个由若干存储单元组成的逻辑存储器；
  - 每个物理存储器在这个逻辑存储器中占有一个地址段，即一段地址空间；
  - **CPU**在这段地址空间中读写数据，实际上就是在相对应的物理存储器中读写数据。



将各类存储器看作一个逻辑存储器





## 2.4 外部设备 – 外设接口寄存器

- 外部设备与主机（**CPU**和存储器）的通信通过外设接口进行
- 每个接口包括一组寄存器：
  - **数据**寄存器
    - 存放主机与外设间传送的数据
  - **状态**寄存器
    - 存放外设或接口的状态（如忙闲标志位）
  - **命令（控制）**寄存器
    - 存放主机给外设或接口的命令（如启动磁盘命令）

## 2.4 外部设备 – I/O地址

- 独立于内存存储器的I/O地址空间

- 外设中的每个寄存器给予一个端口地址（端口号）
- 80x86的I/O地址空间可达64KB，所以地址范围为0000~FFFFH，用16位二进制代码表示

### 8086PC机的内存地址空间分配



## 2.4 外部设备 – I/O编程接口

- 对外设的管理及信息传送是汇编语言最经常使用、也是最复杂的一部分程序
- **80x86**提供两种类型的例行程序供用户使用
  - 基本输入输出系统（**BIOS**）功能调用
    - 存储于系统**ROM**中，工作层次低，更贴近硬件。
    - 例： **INT 10H**：显示功能调用；  
**INT 13H**：磁盘功能调用
  - 磁盘操作系统（**DOS**）功能调用
    - **DOS**操作系统的一部分，开机时由磁盘装载到内存中，工作层次较高。
    - 例： **INT 21H**： **DOS**系统功能调用