《汇编语言》课件

第2章 寄存器(CPU工作原理)

- 2.1 通用寄存器
- 2.2 字在寄存器中的存储
- 2.3 几条汇编指令
- 2.4 物理地址
- 2.5 16位结构的CPU
- 2.6 8086CPU给出物理地址的方法
- 2.7 “段地址×16+偏移地址=物理地址”的本质含义
- 2.8 段的概念
- 2.9 段寄存器
- 2.10 CS和IP
- 2.12 代码段

CPU概述

- 一个典型的CPU由**运算器**、**控制器**、**寄存器**等器件组成。
- **内部总线**实现CPU内部各个器件之间的联系。
- **外部总线**实现CPU和主板上其它器件的联系。

寄存器概述

- 8086CPU有14个寄存器 它们的名称为：

  AX、BX、CX、DX、

  SI、DI、

  SP、BP、IP、

  CS、SS、DS、ES、

  PSW

- 这些寄存器以后会陆续介绍

2.1 通用寄存器

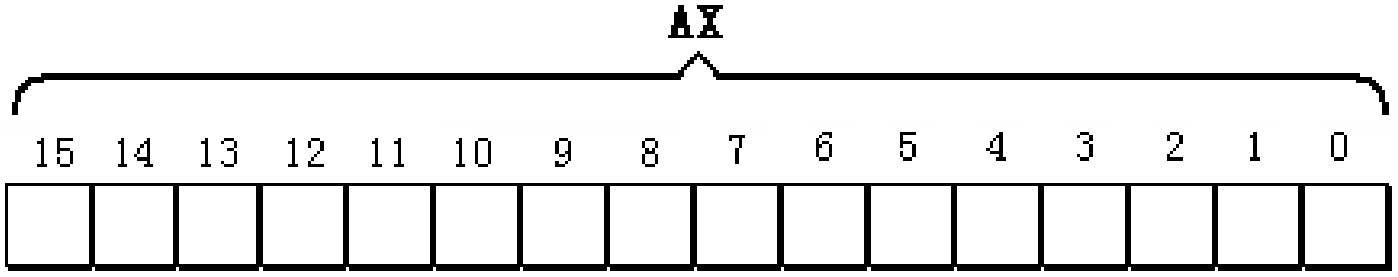
- 8086CPU所有寄存器都是16位，可存放两个字节。
- AX、BX、CX、DX 通常用来存放一般性数据被称为**通用寄存器**。

2.1 通用寄存器

- 下面以AX为例，了解寄存器的逻辑结构。
- 一个16位寄存器可存储一个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？

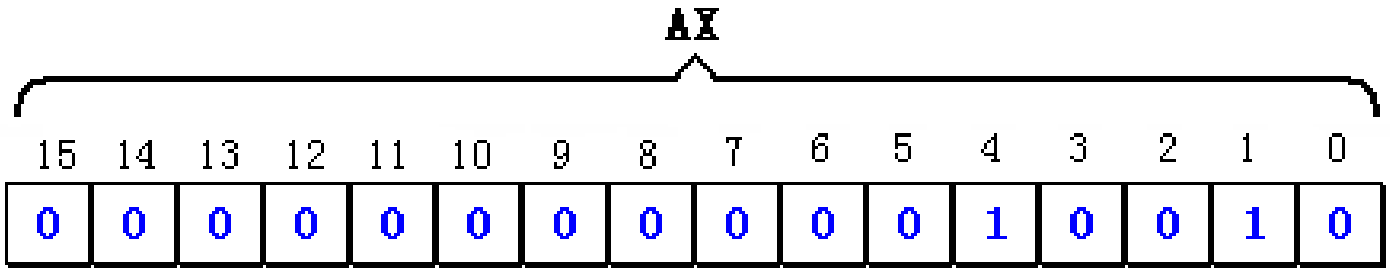
  答案：$2^{16}-1$。



16位寄存器的逻辑结构

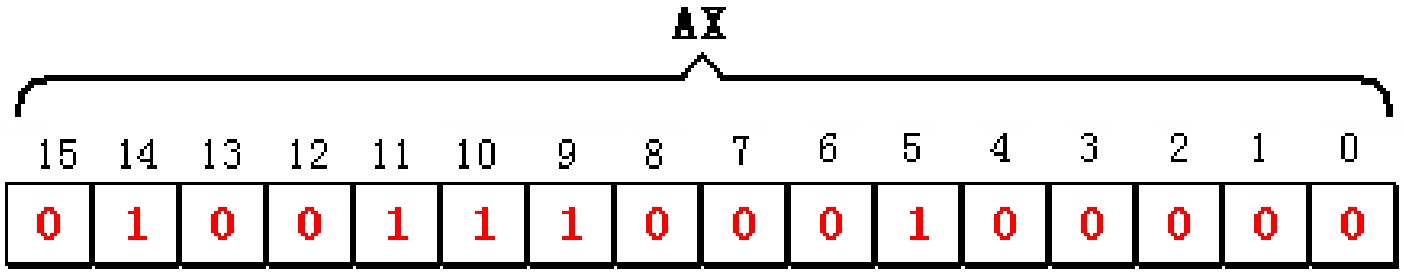
16位数据在寄存器中的存放情况

- 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：



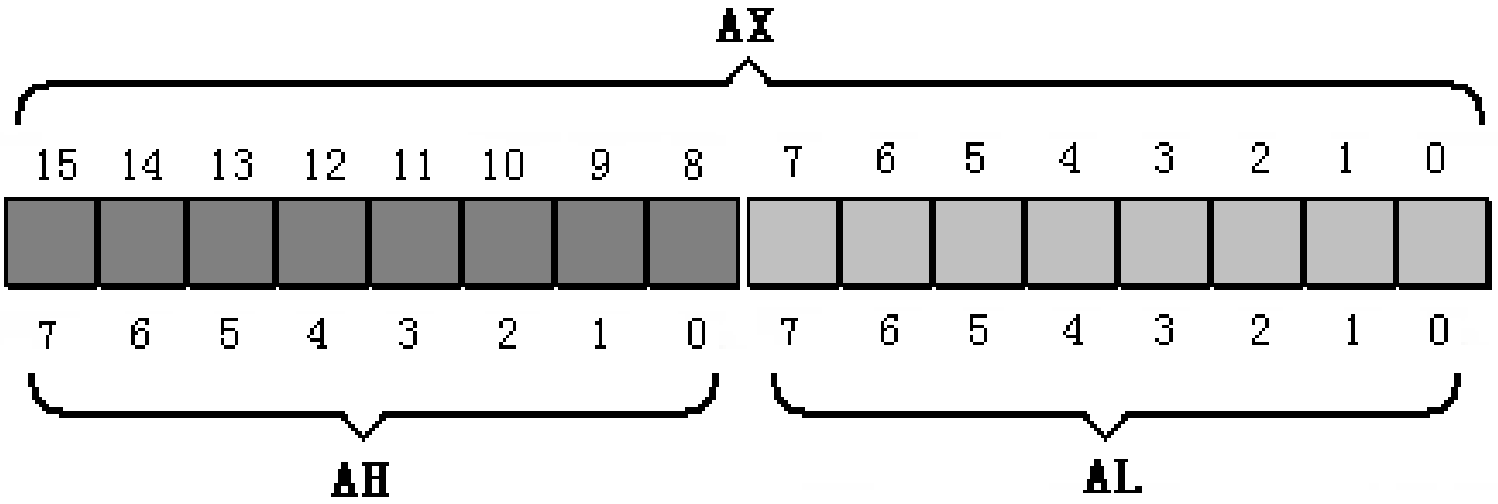
2.1 通用寄存器

- 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。

2.1 通用寄存器

- 8086CPU的8位寄存器存储逻辑
  - 以AX为例，8086CPU的16位寄存器分为两个8位寄存器的情况：



  - AH和AL是**可独立使用**的8位寄存器。

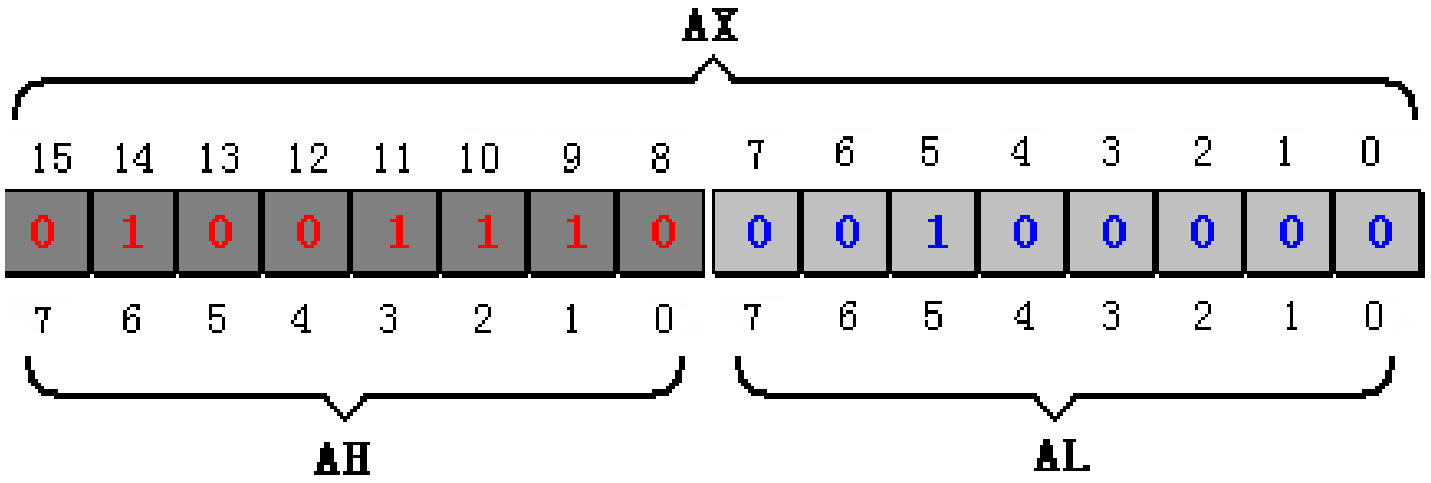
2.1 通用寄存器

- 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。

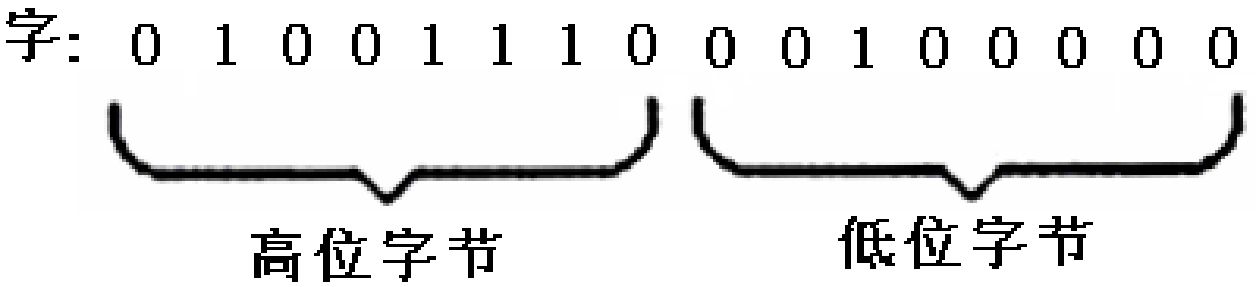
2.1 通用寄存器



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）

2.2 字在寄存器中的存储

- 一个字可以存在一个16位寄存器中
  - 字的高位字节 保存在 寄存器的高8位寄存器
  - 字的高位字节 保存在 寄存器的低8位寄存器



关于数制的讨论

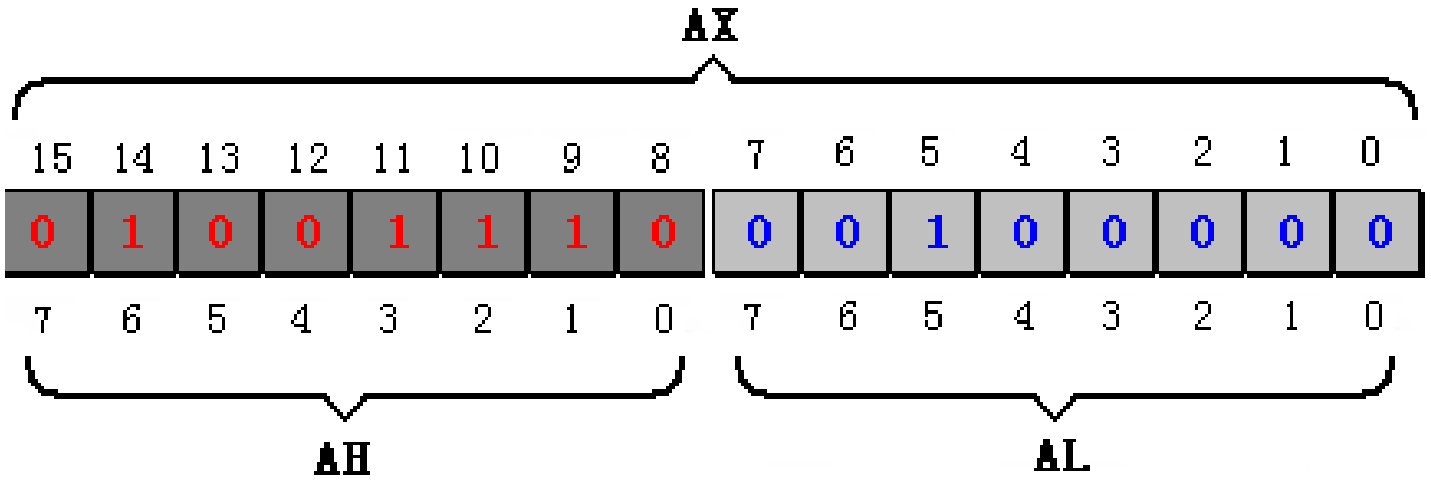
- 数据的表示：
  - 10进制
  - 2进制
  - 16进制

计算机中的数据多是由1~n个8位数据构成的，如

- 一个内存单元可以存放 8位数据，
- CPU中的寄存器又可存放 n 个 8位数据。

用16**进制来**表示数据可以直观的看出这个数据是由哪些8位数据构成的。

关于数制的讨论



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）

2.3 几条汇编指令

汇编指令示例

汇编指令	控制CPU完成的操作	用高级语言的语法描述
mov ax,18	将18送入AX	AX = 18
mov ah,78	将78送入AH	AH = 78
add ax,8	将寄存器AX中的数值加上8	AX = AX + 8
mov ax,bx	将寄存器BX中的数据送入寄存器AX	AX = BX
add ax,bx	将AX,BX 中的内容相加，结果存在AX中	AX = AX + BX

汇编指令举例

汇编指令不区分大小写

2.3 几条汇编指令

- CPU执行下表中的程序段的每条指令后，对寄存器中的数据进行的改变。

程序段中指令执行情况之一 （原AX中的值：0000H，原BX中的值：0000H）

程序段中的指令	指令执行后AX中的数据	指令执行后BX中的数据
mov ax,4E20H		
add ax,1406H		
mov bx,2000H		
add ax,bx		
mov bx,ax		
add ax,bx		

2.3 几条汇编指令

- CPU执行下表中的程序段的每条指令后，对寄存器中的数据进行的改变。

程序段中指令执行情况之一 （原AX中的值：0000H，原BX中的值：0000H）

程序段中的指令	指令执行后AX中的数据	指令执行后BX中的数据
mov ax,4E20H	4E20H	0000H
add ax,1406H	6226H	0000H
mov bx,2000H	6226H	2000H
add ax,bx	8226H	2000H
mov bx,ax	8226H	8226H
add ax,bx	?	8226H

2.3 几条汇编指令

程序段中指令执行情况之二 （原AX中的值：0000H，原BX中的值：0000H）

程序段中的指令	指令执行后AX中的数据	指令执行后BX中的数据
mov ax,001AH	001AH	0000H
mov bx,0026H	001AH	0026H
add al,bl	0040H	0026H
add ah,bl	2640H	0026H
add bh,al	2640H	4026H
mov ah,0	0040H	4026H
add al,85H	00C5H	4026H
add al,93H	?	4026H

2.3 几条汇编指令

- 指令

add al, 93H

是看作8位的寄存器上的加法，

此时CPU把al 和 ah 看作8位的寄存器上的独立的寄存器，**产生的进位不会存储在ah中**。

这个进位CPU丢弃了？此问题在后面的课程中讨论。

2.3 几条汇编指令

- 进行数据传送或运算时，两个操作对象位数要一致。
  - 以下哪些正确？

```
mov ax, bx
mov bx, ax
mov ax, 2000
add ax, 100H
```

对

```
mov ax, bl
mov al, bx
mov al, 2000
add al, 100H
```

错

特别提示

- 检测点2.1（Page 19）
- 没有通过检测点请不要向下学习！

2.4 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。
  - 每个内存单元在这个空间中都有唯一的地址——**物理地址**。

2.5 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。

2.6 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，表现出的寻址能力却只有64K。

2.6 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。

- 地址加法器合成物理地址的方法：

物理地址 = 段地址×16 + 偏移地址

在8086CPU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

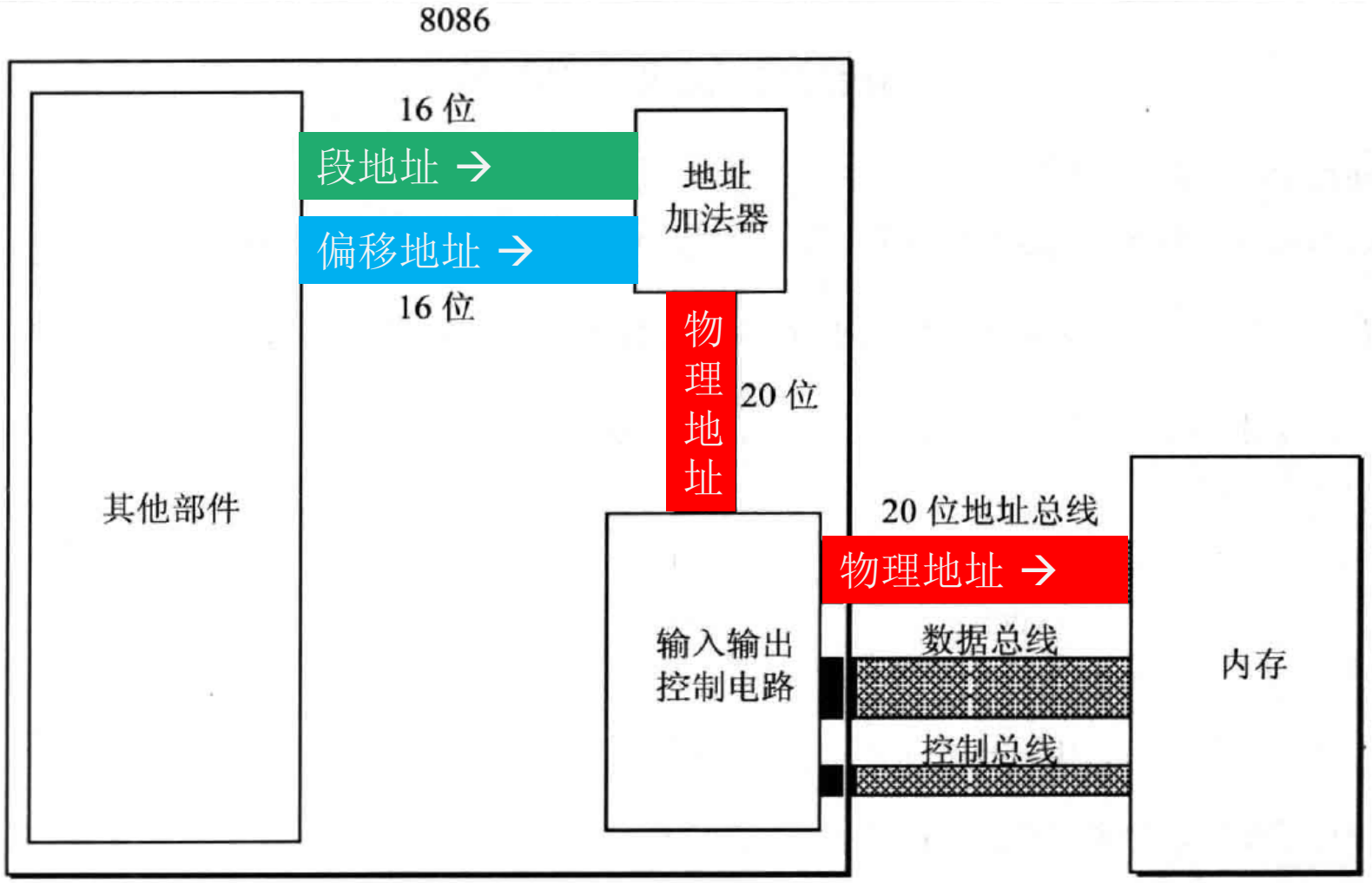


图 2.6　8086CPU 相关部件的逻辑结构

在8086CPU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

- 例如：

  8086CPU访问地址为123C8H的内存单元

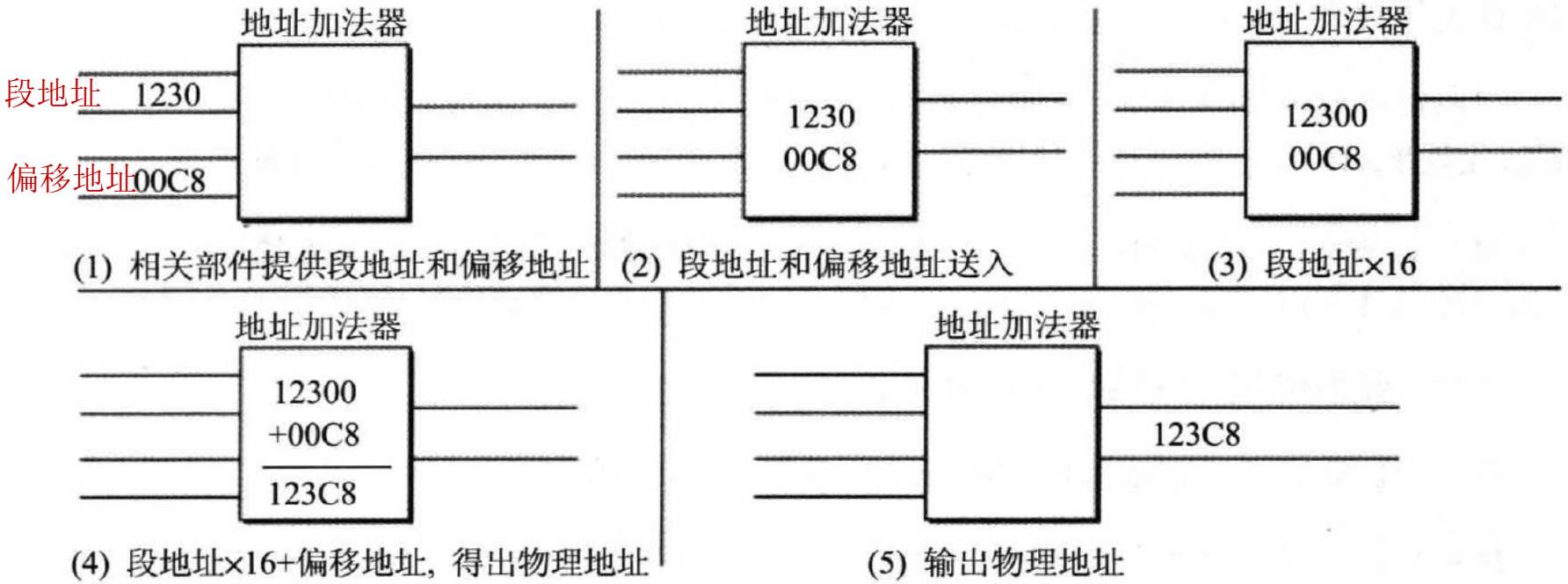


图 2.7　地址加法器的工作过程

由段地址×16引发的讨论

- 观察移位次数和各种形式数据的关系：
  - 地址加法器如何完成段地址×16的运算？

答：以二进制形式存放的段地址左移4位。

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

2.7“段地址×16+偏移地址=物理地址”的本质含义

- “段地址×16+偏移地址=物理地址”的本质含义

- 利用两个比喻说明：
  - 比喻1：

    说明“基础地址+偏移地址 = 物理地址” 的思想

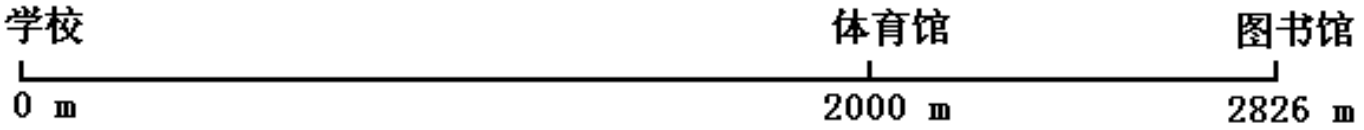
  - 比喻2：

    说明“段地址×16+偏移地址=物理地址”的思想

    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”

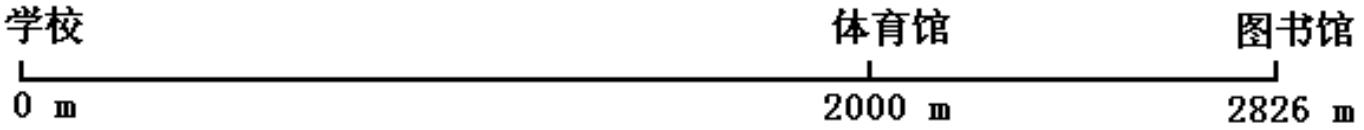
- 比喻1：



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”

- 比喻1:



- 描述方法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。

- 描述方法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×16+偏移地址=物理地址”

- 比喻2：

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×16+偏移地址=物理地址”

- 比喻2：

  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

2.8 段的概念

内存是否被划分成了一个一个的段，每一个段有一个段地址呢？

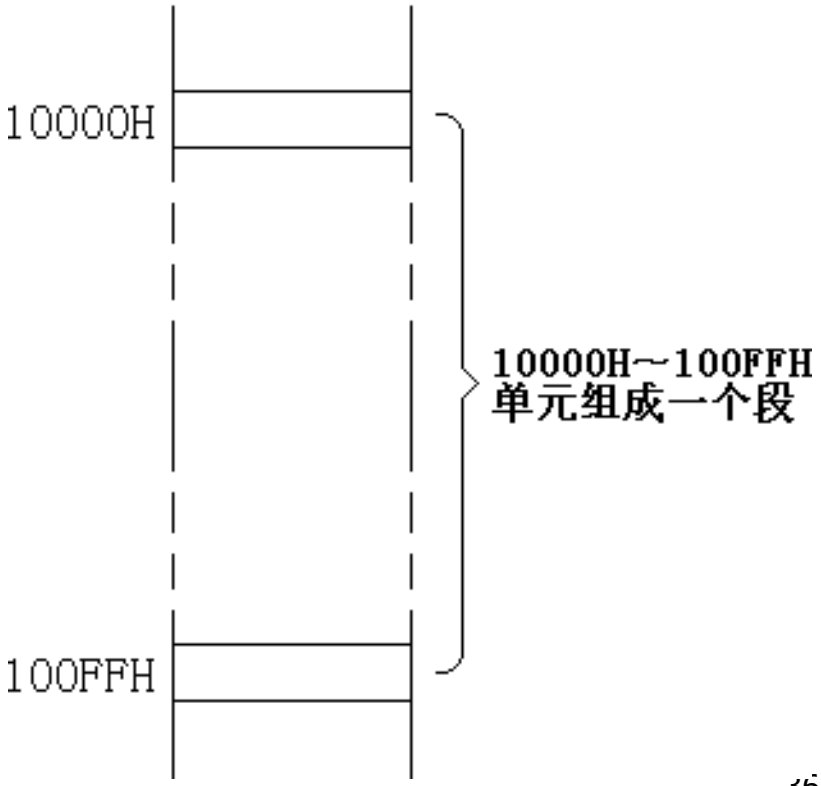
解答：内存并没有分段，段的划分来自于CPU，由于8086CPU用

“（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。

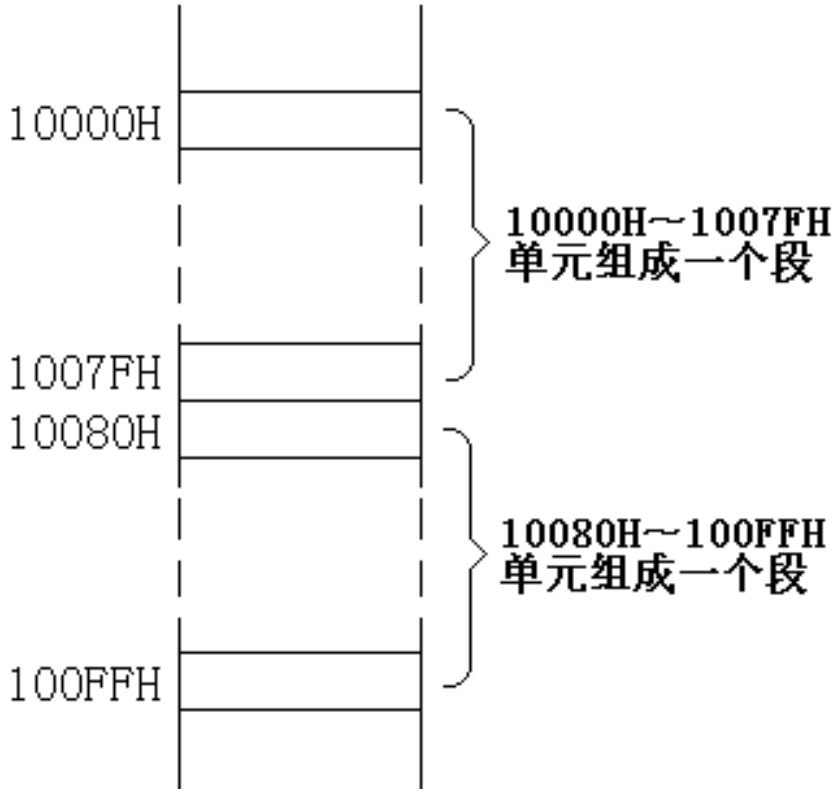
2.8 段的概念

- 若地址10000H~100FFH的内存单元组成一个段，
  - 段的起始地址（基础地址）为10000H，即段地址为1000H。
  - 大小为100H。



2.8 段的概念

- 也可以分出两个段
- 10000H~1007FH 、10080H~100FFH 的内存单元组成，它们的
  - 起始地址（基础地址）为10000H和10080H，
  - 段地址为1000H 和1008H
  - 大小都为80H



2.8 段的概念

- 在编程时，可根据需要将若干地址连续的内存单元看作一个段，
  - 用段地址×16定位段的起始地址（基础地址）
  - 用偏移地址定位段中的内存单元。

- 两点需要注意

  1. 段地址×16 必然是 16的倍数，所以一个段的起始地址也一定是16的倍数；

  2. 偏移地址为16位，16 位地址的寻址能力为 64K，所以一个段的长度最大为64K。

2.8 段的概念

- 内存单元地址小结
  - CPU访问内存单元时，必须向内存提供内存单元的物理地址。
  - 8086CPU在内部用段地址和偏移地址移位相加的方法形成最终的物理地址。

内存单元地址小结

问题1：观察下面的地址，有什么发现？

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

- 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。

内存单元地址小结

问题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。
- 示例：给定段地址1000H，用偏移地址寻址，CPU的寻址范围为：10000H~1FFFFH。

内存单元地址小结

- 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：
  - 数据存在内存2000:1F60单元中；
  - 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。

特别提示

- 检测点2.2（Page 25）
- 没有通过检测点请不要向下学习！

2.9 段寄存器

- 段寄存器就是提供段地址的。

8086CPU有4个段寄存器：

CS

DS

SS

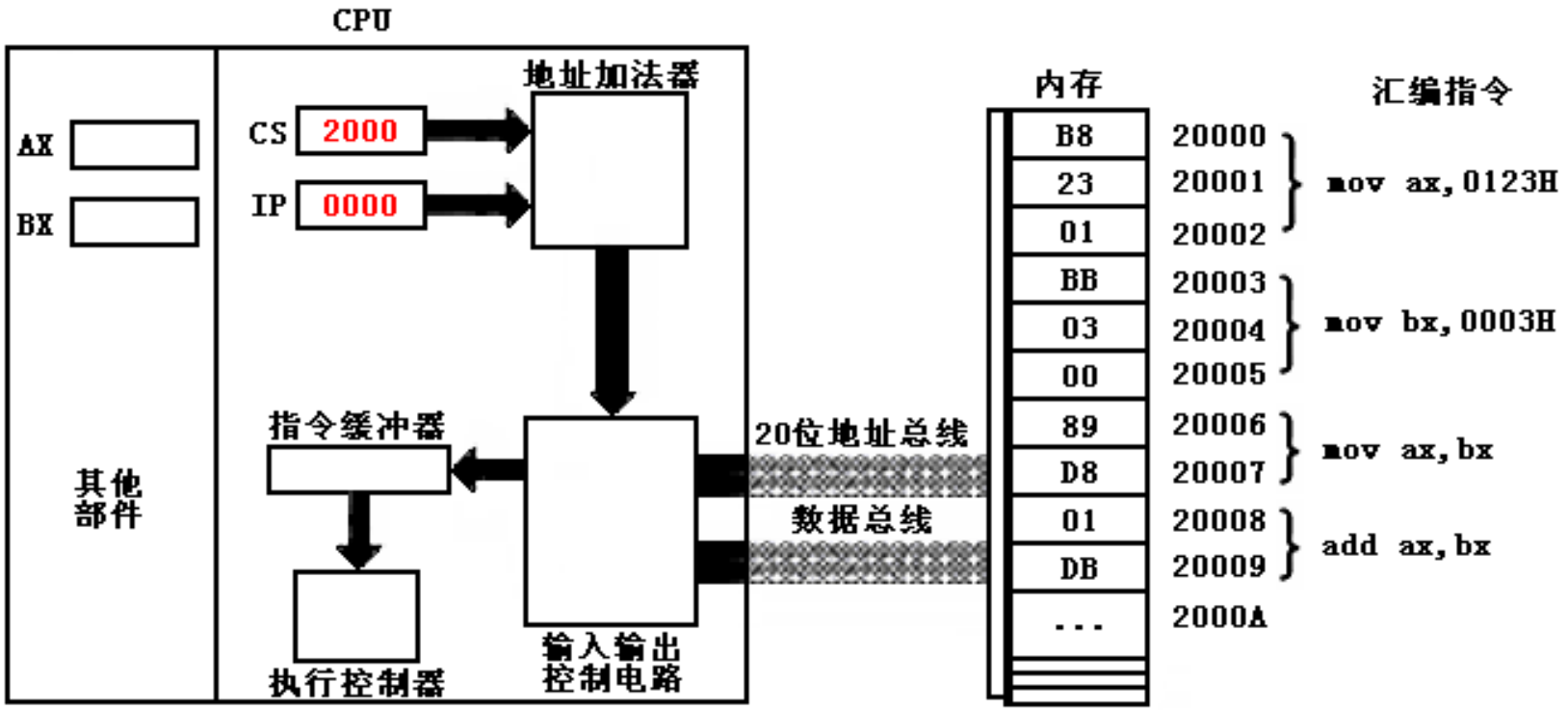
ES

当8086CPU要访问内存时，这4个段寄存器提供内存单元的段地址。

2.10 CS和IP

- CS（代码段寄存器）和IP（指令指针寄存器）是8086CPU中最关键的寄存器，它们指示了CPU当前要读取指令的**指令地址**。

8086PC读取和执行指令相关部件



读取和执行指令演示

- 8086PC读取和执行指令演示

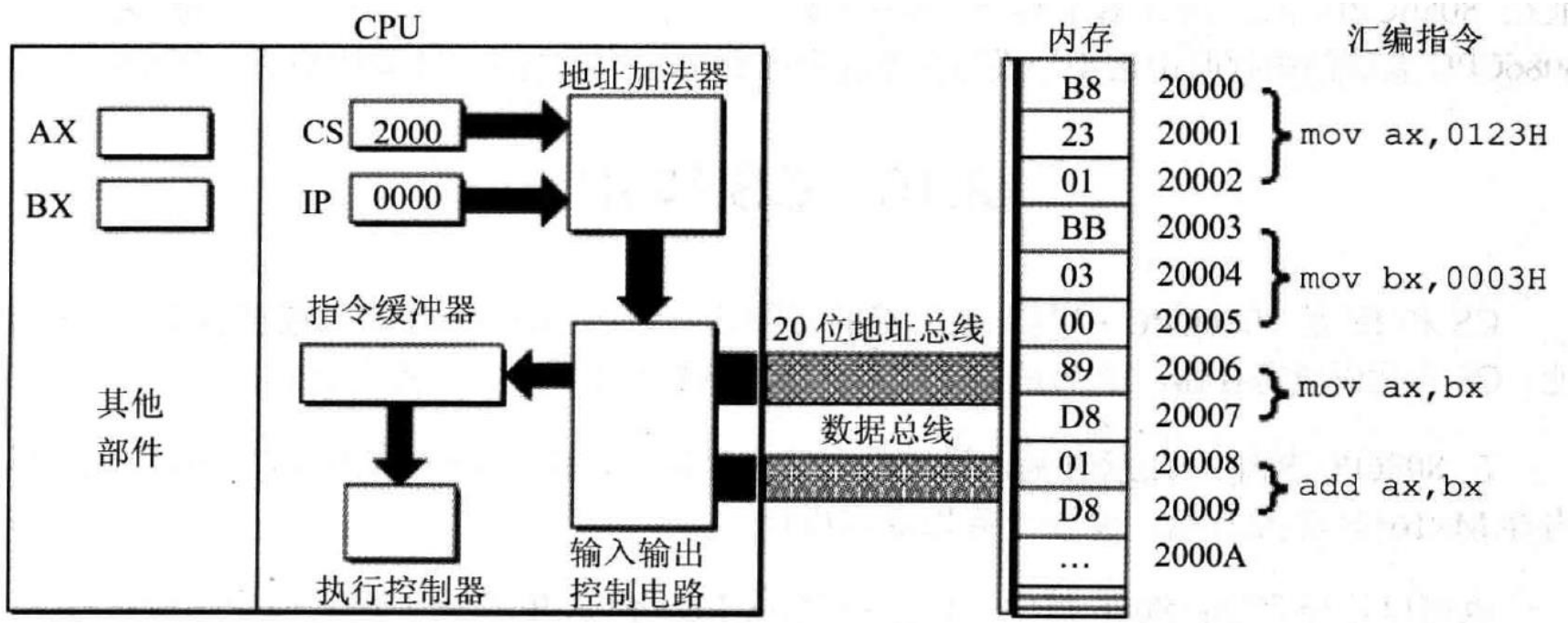


图 2.11　初始状态(CS:2000H，IP:0000H，CPU 将从内存 2000H×16+0000H 处读取指令执行)

读取和执行指令演示

- 8086PC读取和执行指令演示

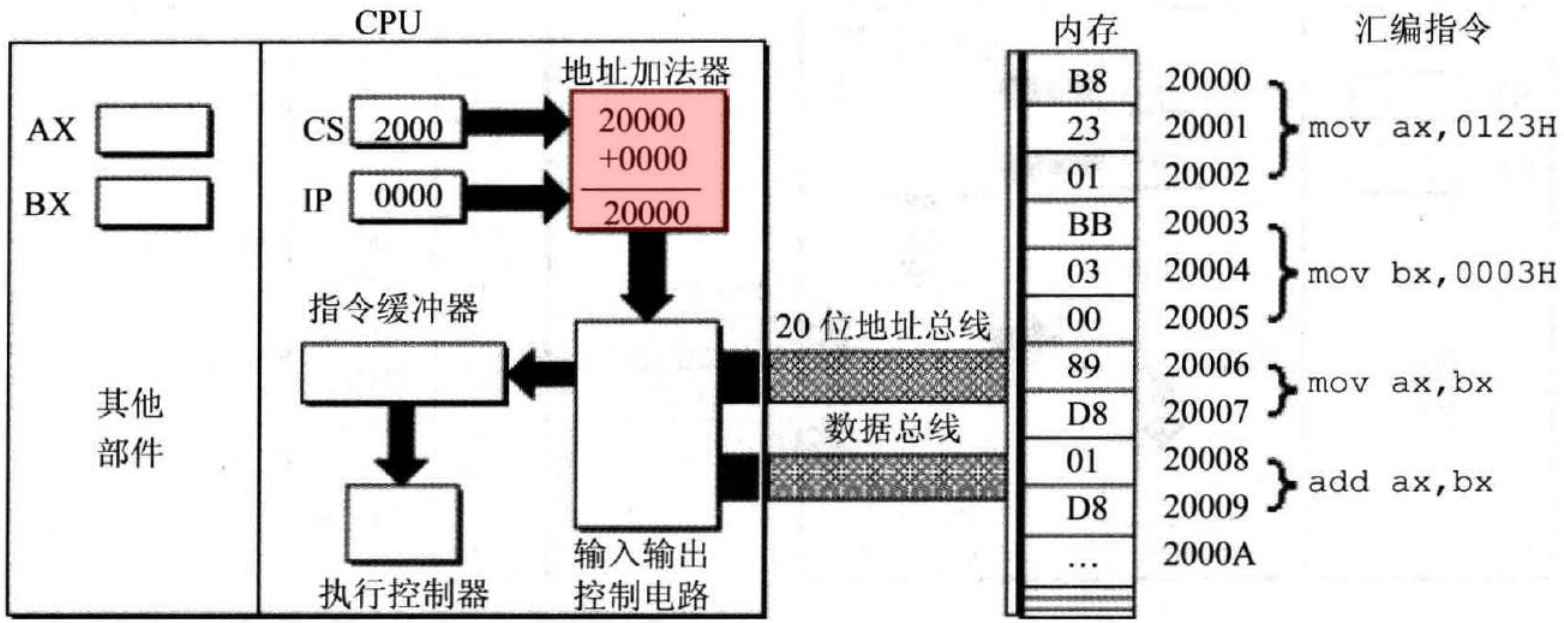


图 2.12　CS、IP 中的内容送入地址加法器(地址加法器完成：物理地址=段地址×16+偏移地址)

读取和执行指令演示

- 8086PC读取和执行指令演示

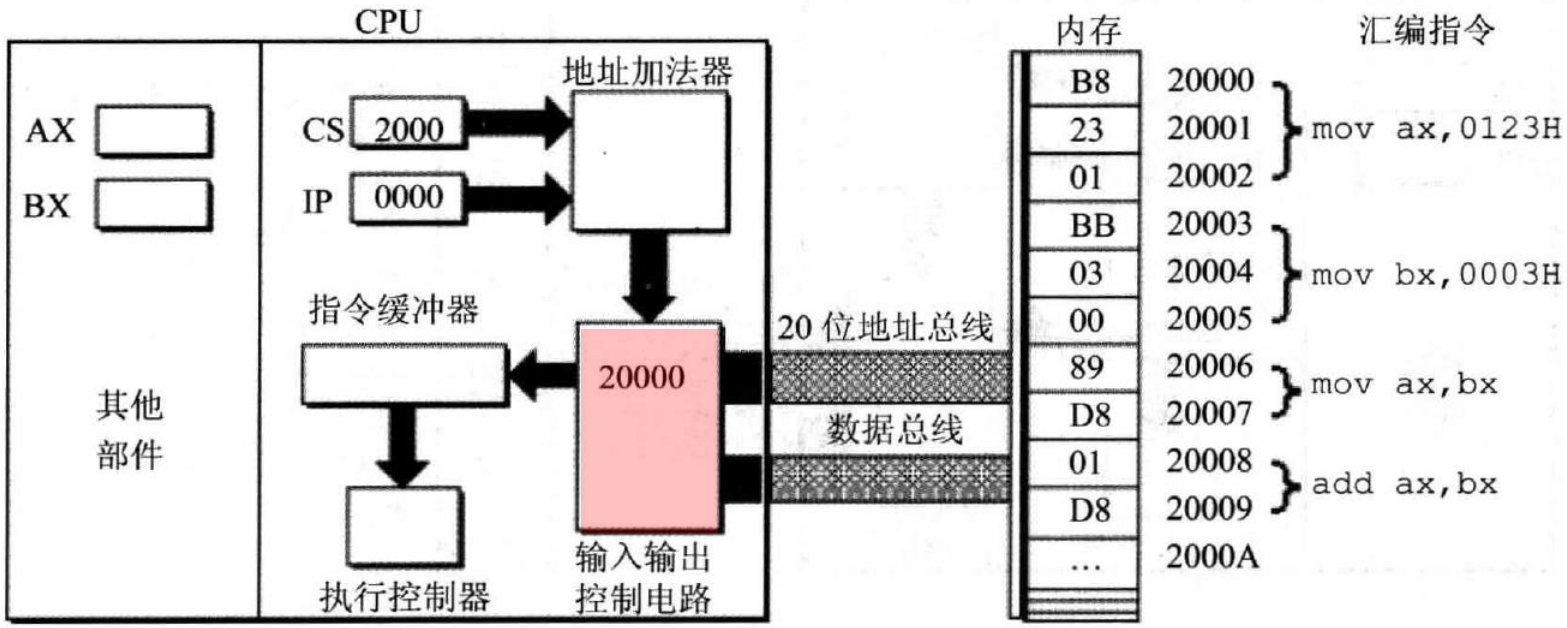


图 2.13　地址加法器将物理地址送入输入输出控制电路

读取和执行指令演示

- 8086PC读取和执行指令演示

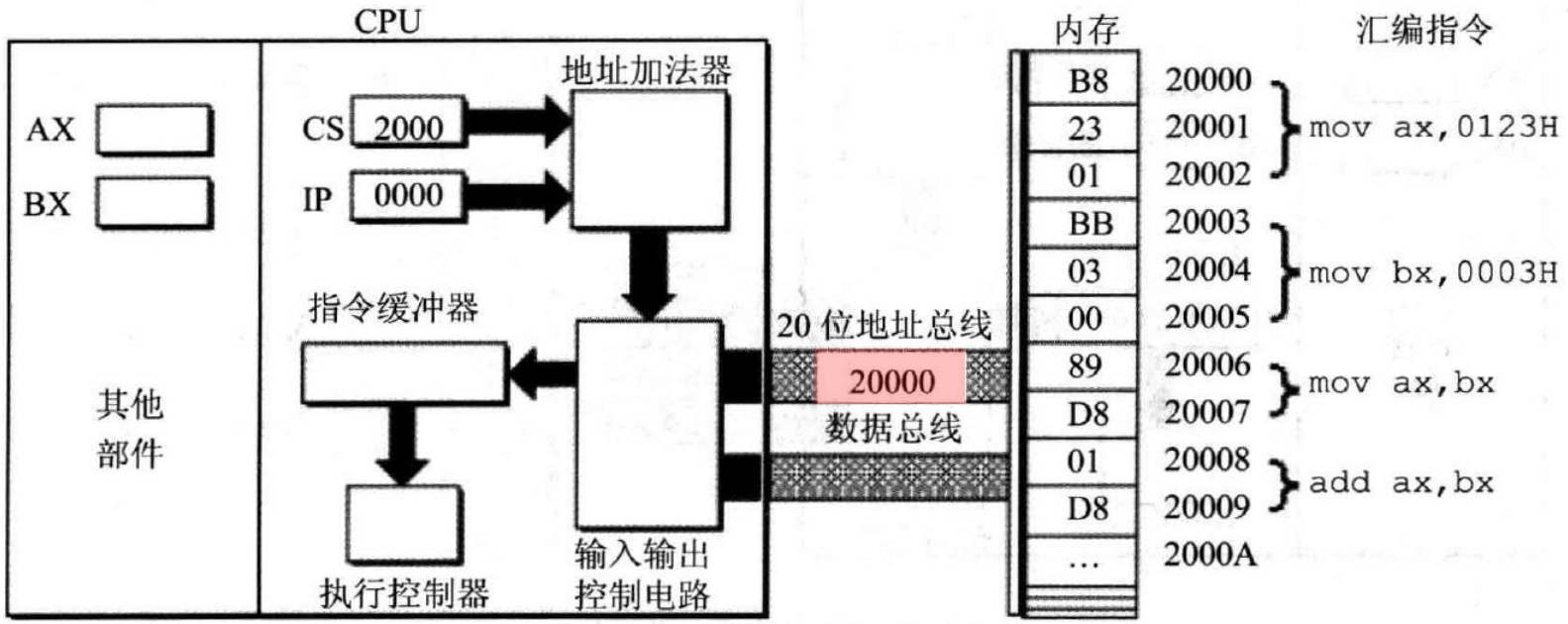


图 2.14　输入输出控制电路将物理地址 20000H 送上地址总线

读取和执行指令演示

- 8086PC读取和执行指令演示

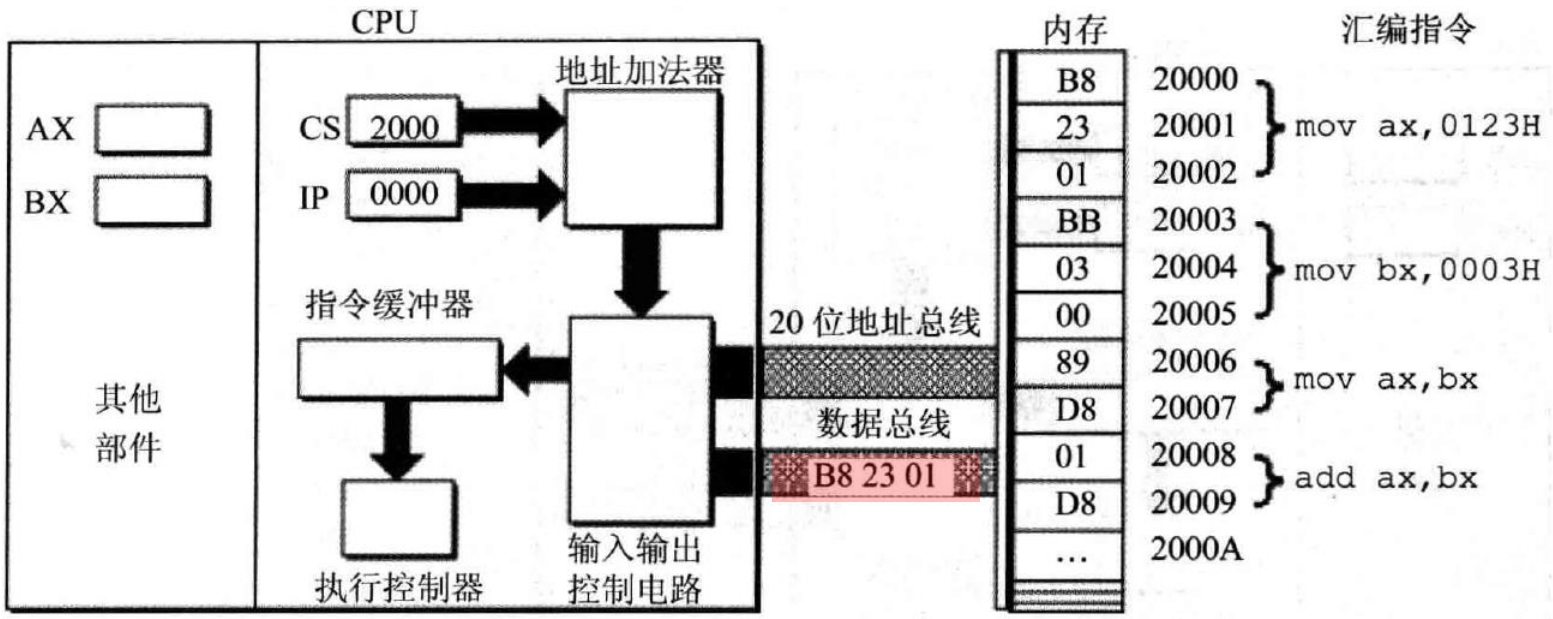


图 2.15　从内存 20000H 单元开始存放的机器指令 B8 23 01 通过数据总线被送入 CPU

读取和执行指令演示

- 8086PC读取和执行指令演示

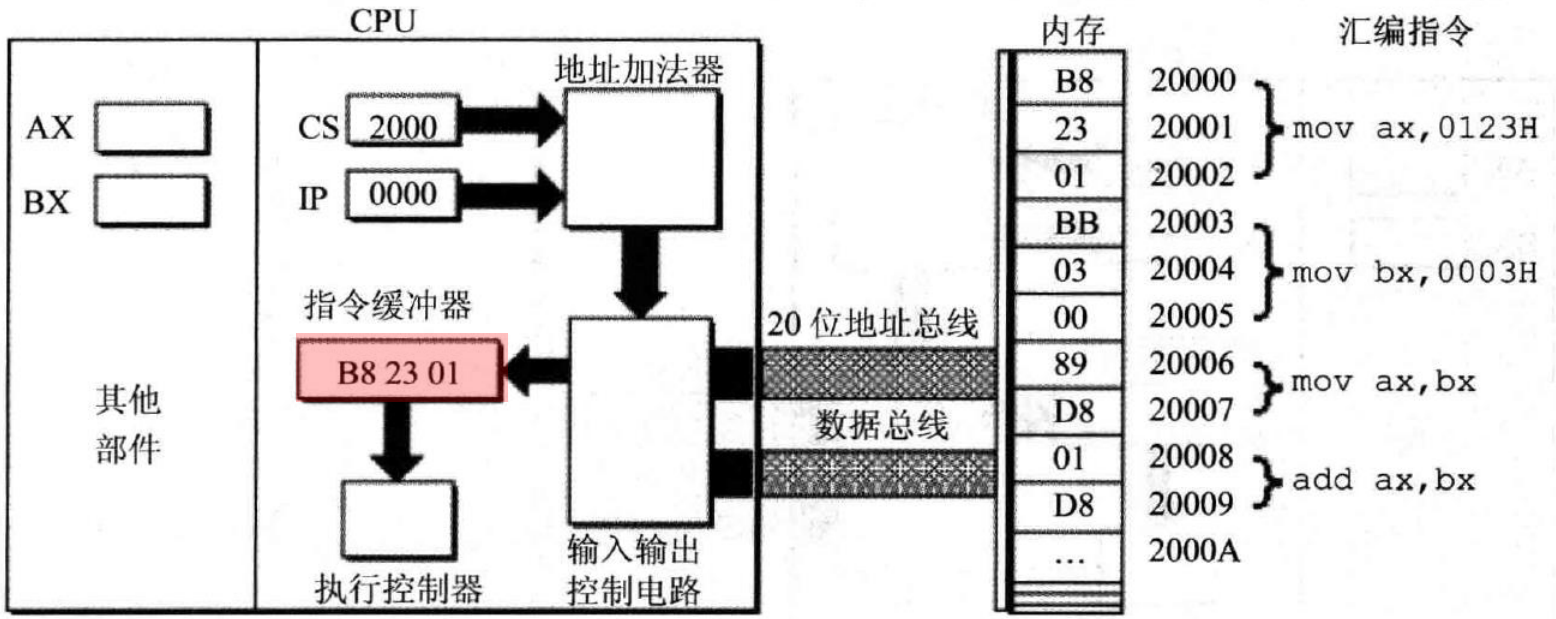


图 2.16　输入输出控制电路将机器指令 B8 23 01 送入指令缓冲器

读取和执行指令演示

- 8086PC读取和执行指令演示

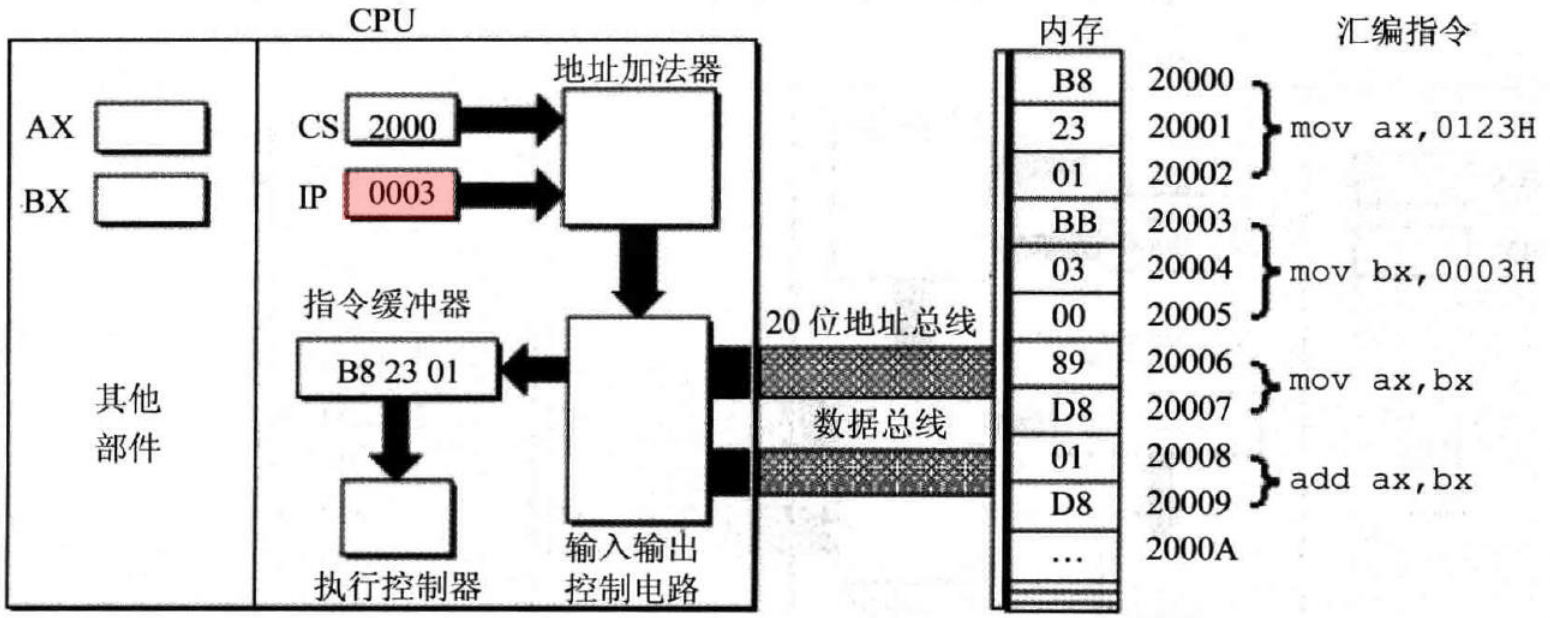


图 2.17　IP 中的值自动增加

(读取一条指令后，IP 中的值自动增加，以使 CPU 可以读取下一条指令。因当前读入的指令 B82301 长度为 3 个字节，所以 IP 中的值加 3。此时，CS：IP 指向内存单元 2000:0003。)

读取和执行指令演示

- 8086PC读取和执行指令演示

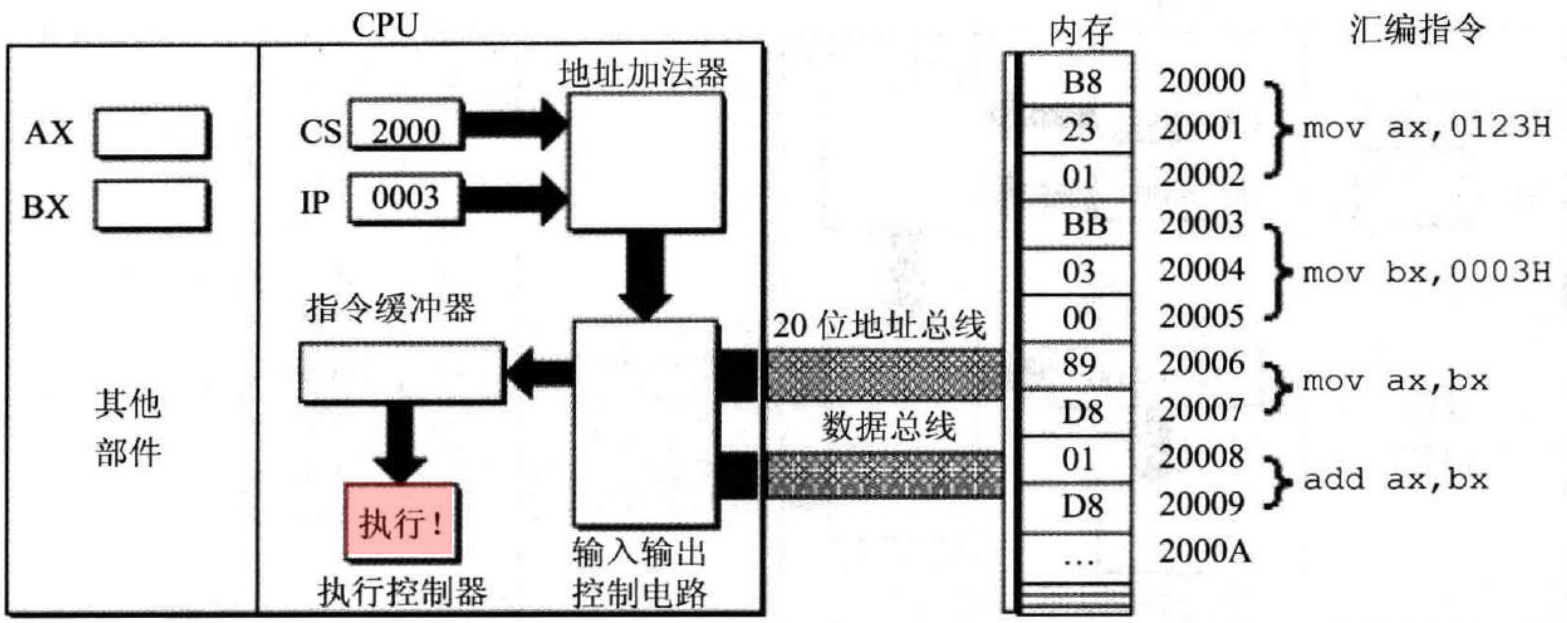


图 2.18　执行控制器执行指令 B8 23 01(即 mov ax,0123H)

读取和执行指令演示

- 8086PC读取和执行指令演示

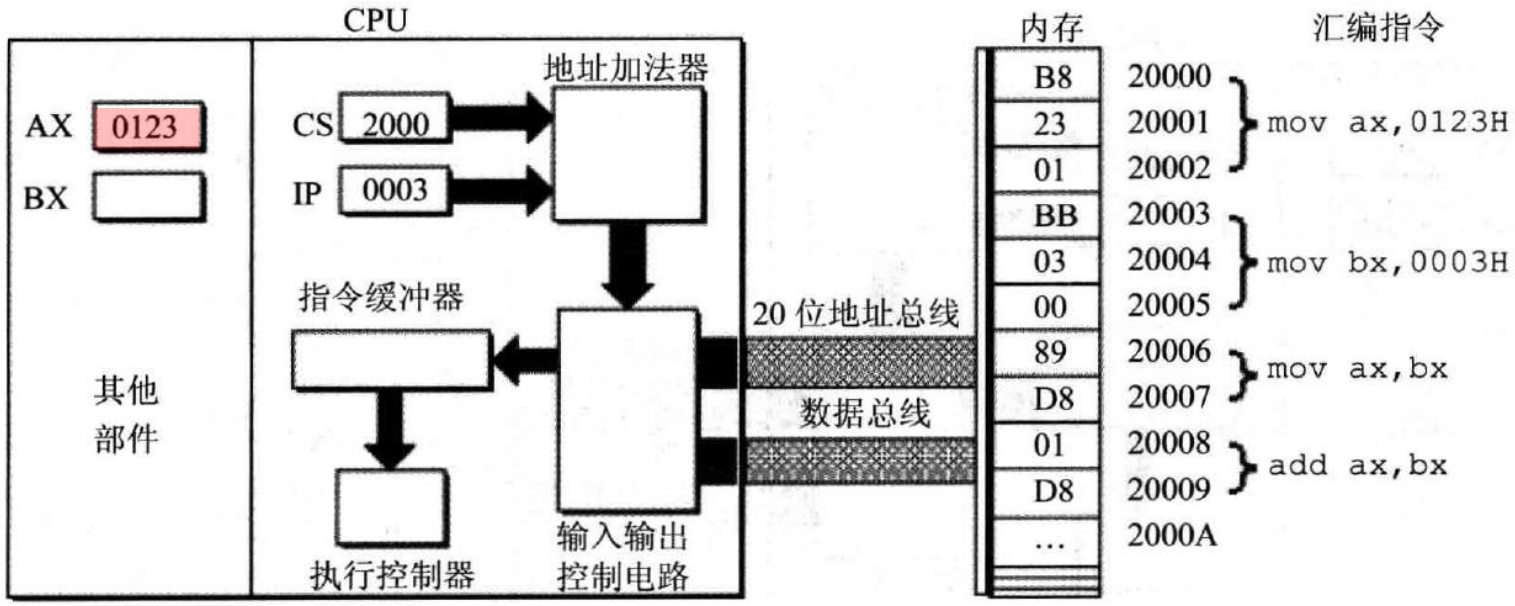


图 2.19　指令 B8 23 01 被执行后 AX 中的内容为 0123H

(此时，CPU 将从内存单元 2000:0003 处读取指令。)

读取和执行指令演示

- 8086PC读取和执行指令演示

……

后续指令的执行过程，请阅读教材p29-p31，注意IP的变化

8086PC工作过程的简要描述

8086PC工作过程的流程

1. 从CS:IP指向内存单元读取指令，读取的指令进入指令缓冲器；

2. IP = IP + 所读取指令的长度，从而指向下一条指令；

3. 执行指令。 转到步骤1，重复这个过程。

8086PC工作过程的简要描述

- 在 8086CPU 加电启动或复位后（即 CPU刚开始工作时）

  CS=FFFFH

  IP=0000H，

  所以CPU从内存FFFF0H单元中读取指令执行，这是开机后执行的第一条指令。

2.10 CS和IP

- 内存中指令和数据没有任何区别，都是二进制信息，CPU在工作的时候把有的信息看作指令，有的信息看作数据。

CPU根据什么将内存中的信息看作指令？

答：CPU将CS:IP指向的内存单元中的内容看作指令。

- 在任何时候，CPU将CS、IP中的内容当作指令的段地址和偏移地址，用它们合成指令的物理地址，到内存中读取指令码，执行。

2.11 修改CS、IP的指令

- 在CPU中，程序员能够用指令**读写的部件**只有寄存器，程序员可以通过改变寄存器中的内容实现对CPU的控制。

- CPU从何处执行指令是由CS、IP中的内容决定的，可以通过改变CS、IP中的内容来控制CPU执行目标指令。

2.11 修改CS、IP的指令

- 同时修改CS、IP的内容：

jmp 段地址:偏移地址

功能：用指令中的段地址修改CS，偏移地址修改IP。

例如：

jmp 2AE3:3

jmp 3:0B16

2.11 修改CS、IP的指令

- 仅修改IP的内容：

jmp 某一合法寄存器

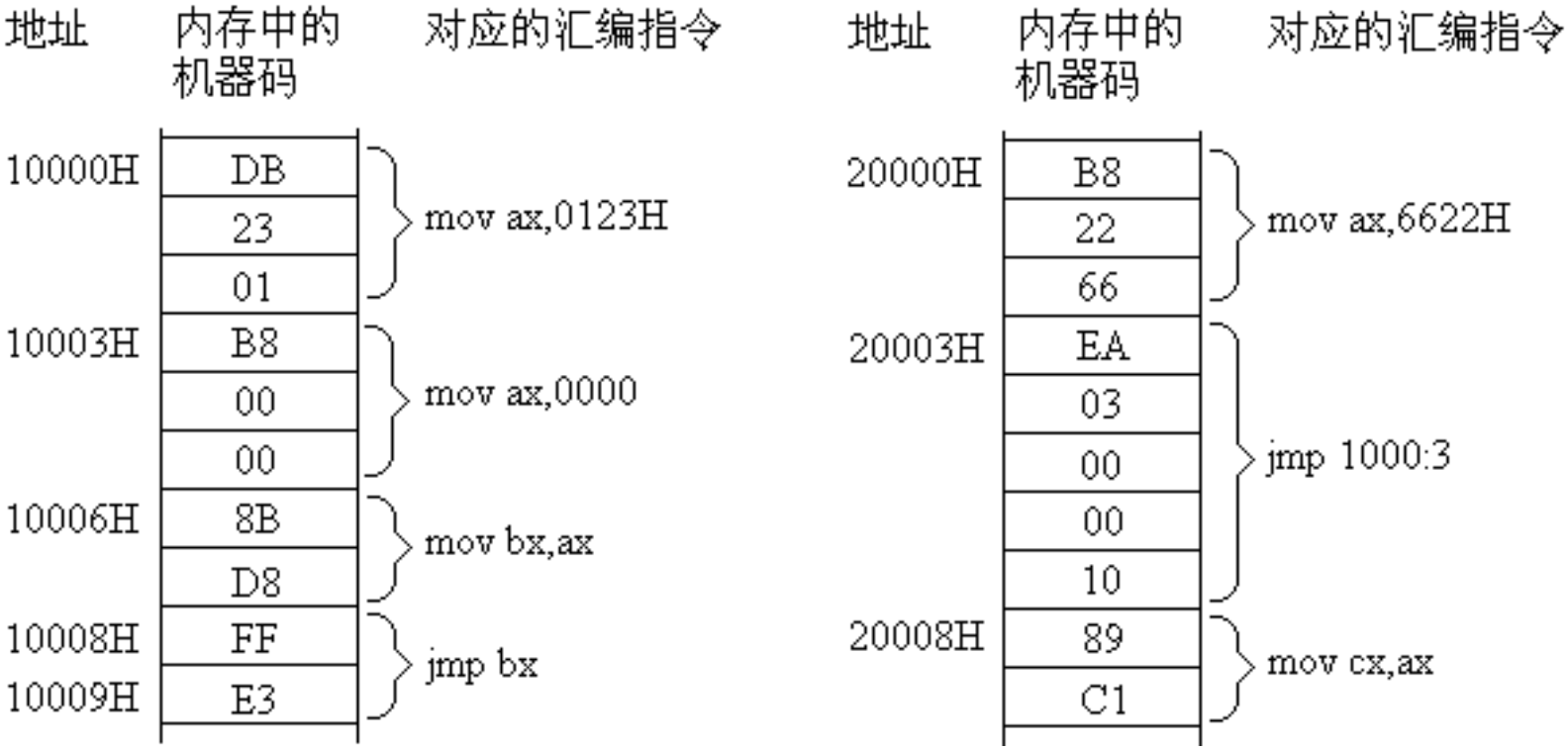
功能：用寄存器中的值修改IP。

例如

```
jmp ax
jmp bx
```

问题分析

- 内存中存放的机器码和对应汇编指令情况：（初始：CS=2000H，IP=0000H）



- 请写出指令执行序列：

问题分析结果：

请写出指令执行序列：

（1）mov ax,6622
（2）jmp 1000:3
（3）mov ax,0000
（4）mov bx,ax
（5）jmp bx
（6）mov ax,0123H
（7）转到第（3）步执行

2.12 代码段

- 对于8086PC机，在编程时，可以根据需要，将一组内存单元定义为一个段。

- 可以将长度为N（N≤64KB）的一组代码，存在一组地址连续、起始地址为16的倍数的内存单元中，这段内存是用来存放代码的——代码段。

2.12 代码段

- 代码段示例

mov ax,0000	(B8 00 00)
add ax,0123	(05 23 01)
mov bx,ax	(8B D8)
jmp bx	(FF E3)

若这段指令（长度：10 字节），存在地址为123B0H~123B9H的一组内存单元中，可认为123B0H~123B9H这段内存单元是用来存放代码的，是一个代码段，它的段地址为123BH，长度为10字节。

2.12 代码段

- 如何使得代码段中的指令被执行呢？

将一段内存当作代码段，仅仅是我们在编程时的一种安排，CPU 并不会由于这种安排，就自动地将我们定义的代码段中的指令当作指令来执行。

CPU 只认被 CS:IP 指向的内存单元中的内容为指令。

所以要将CS:IP指向所定义的代码段中的第一条指令的首地址。

CS = 123BH，IP = 0000H。

2.9节~2.12节 小结

1、段地址在8086CPU的寄存器中存放。当8086CPU要访问内存时，由段寄存器提供内存单元的段地址。8086CPU有4个段寄存器，其中CS用来存放指令的段地址。

2、CS存放指令的段地址，IP存放指令的偏移地址。

8086机中，任意时刻，CPU将CS:IP指向的内容当作指令执行。

2.9节~2.12节 小结（续）

3、8086CPU的工作过程：

1. 从CS:IP指向内存单元读取指令，读取的指令进入指令缓冲器；
2. IP指向下一条指令；
3. 执行指令。（转到步骤（1），重复这个过程。）

4、8086CPU提供转移指令修改CS、IP的内容。

特别提示

- 检测点2.3（Page 33）
- 没有通过检测点请不要向下学习！