3.4 8086/8088的存储器组织

一. 存储器地址空间和数据存储格

8086/8088的存储器都是以字节(8位）为单位组织的。它们具有20条地址总线，所以可寻址的存储器地址空间容量为 2^{20}（约 1 M）字节。每个字节对应一个唯一的地址，地址范围为 0 ~ 2^{20} -1（用16进制表示为00000 ~ FFFFFH），如下图3.4.1所示。

3.4 8086/8088的存储器组织

十六进制地址	二　进　制　地　址
00000	0000 0000 0000 0000 0000
00001	0000 0000 0000 0000 0001
00002	0000 0000 0000 0000 0010
00003	0000 0000 0000 0000 0011
FFFFE	1111 1111 1111 1111 1110
FFFFF	1111 1111 1111 1111 1111

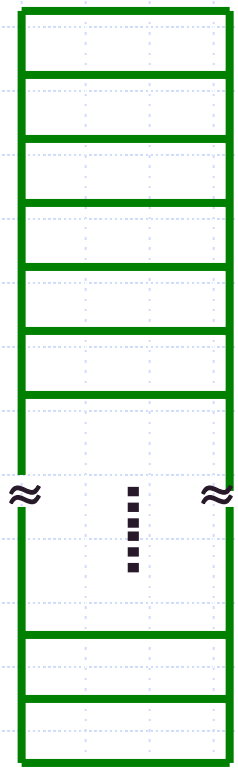


图3.4.1

3.4 8086/8088的存储器组织

存储器内两个连续的字节，定义为一个字，一个字中的每个字节，都有一个字节地址，每个字的低字节（低8位）存放在低地址中，高字节（高8位）存放在高地址中。字的地址指低字节的地址。各位的编号方法是最低位（LSB）为位0，一个字节中，最高位（MAS）编号为位7；一个字中最高位的编号为位15。

这些约定如图3.4.2所示

3.4 8086/8088的存储器组织

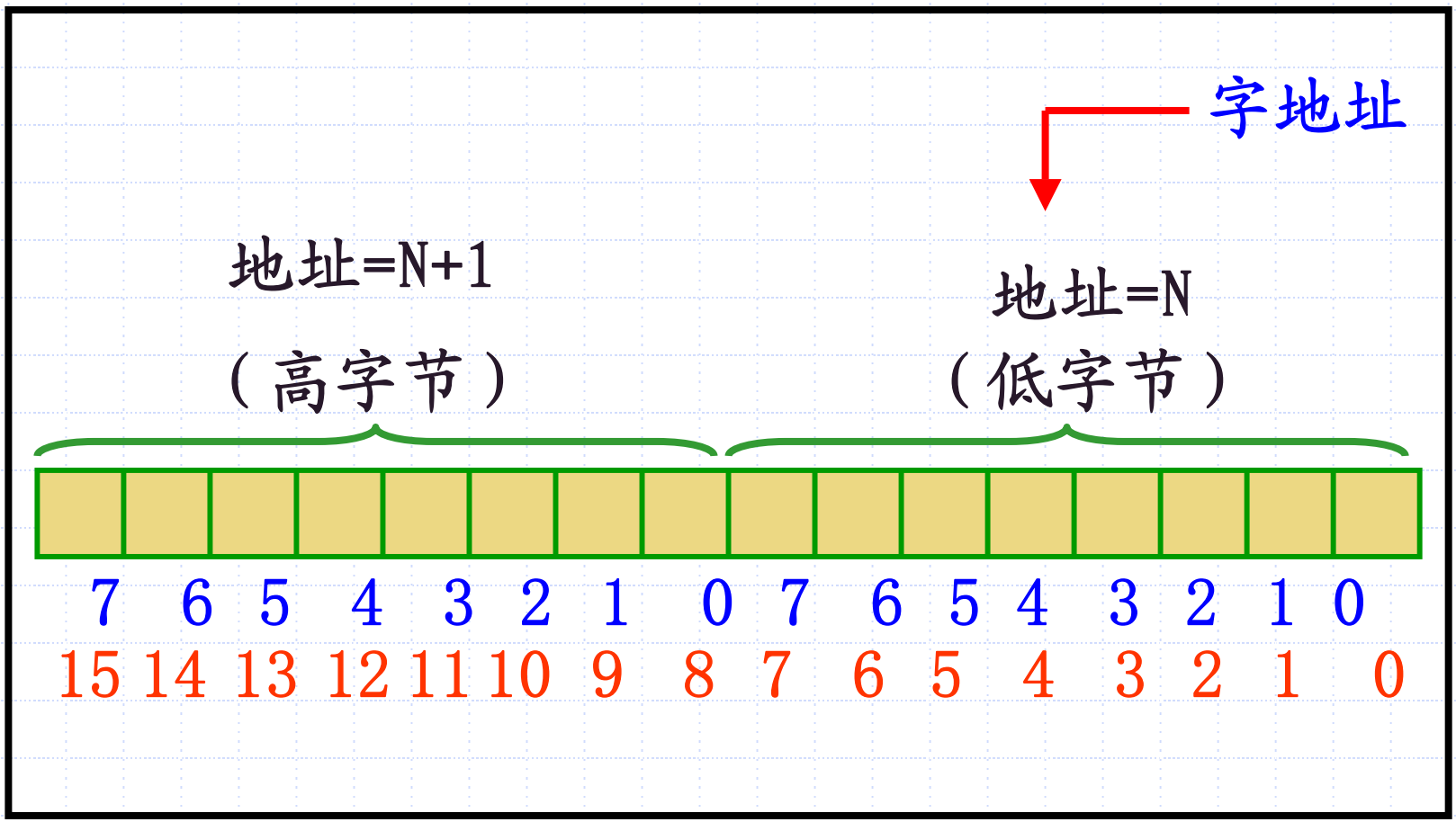


图3.4.2

3.4 8086/8088的存储器组织

字数据在存储器中存放的格式如图3.4.3所示

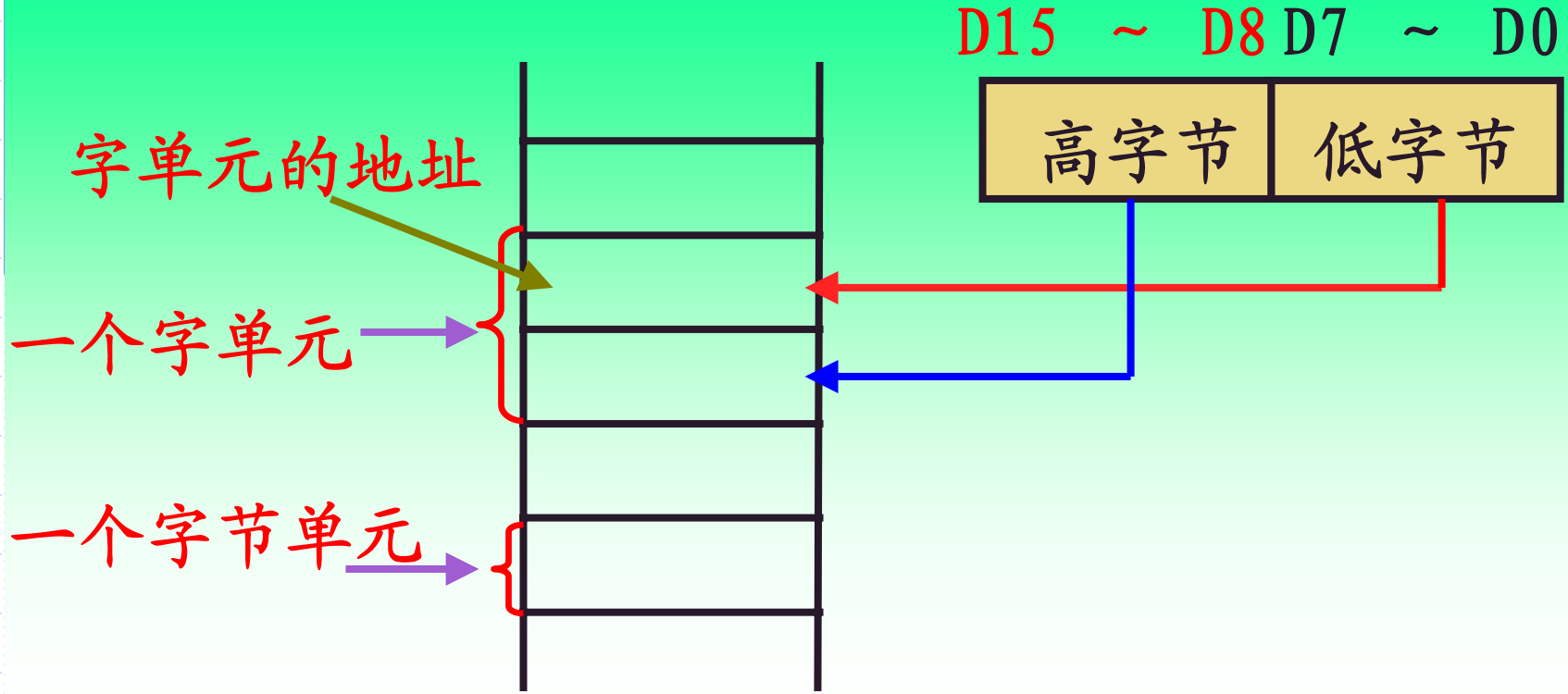


图3.4.3　字数据在存储器中存放格式示意图

3.4 8086/8088的存储器组织

8086/8088允许字从任何地址开始。字的地址是偶地址时，称字的存储是对准的，若字的地址是奇地址时，则称字的存储是未对准的。

8086 CPU数据总线16位，对于访问（读或写）字节的指令，需要一个总线周期。对于访问一个偶地址的字的指令，也是需要一个总线周期。而对于访问一个奇地址的字的指令，则需要两个总线周期（CPU自动完成）。

3.4 8086/8088的存储器组织

8088 CPU数据总线8位，无论是字，还是字节数据存取操作，也无论是偶地址的字，还是奇地址的字，每一个总线周期只能完成一个字节的数据存取操作。对字数据所组成的连续两个总线周期是由CPU 自动完成的。

3.4 8086/8088的存储器组织

二. 存储器的分段和物理地址的形式

1. 为什么要分段

从前面的介绍可知，

8086/8088CPU有20条地址线（A19∽A0），能寻址外部存贮空间为2^{20}=1MB，而在8088/8086CPU内部能向存贮器提供地址码的地址寄存器有六个，均为16位，所以用这六个16位地址寄存器任意一个给外部存贮器提供地址，只能提供2^{16}=64K个地址，所以，对1MB地址寻址不完。这六个16位地址寄存器分别为：

3.4 8086/8088的存储器组织

- BX → 基址寄存器
- BP → 基址指针寄存器
- SI → 源变址寄存器
- DI → 目的变址寄存器
- SP → 堆栈指针寄存器
- IP → 指令指针寄存器

为了使8088/8086CPU能寻址到外部存贮器1MB空间中任何一个单元，8088/8086巧妙地采用了地址分段方法（将1MB空间分成若干个逻辑段），从而将寻址范围扩大到了1MB。

3.4 8086/8088的存储器组织

2. 怎么分段

- 1MB的存贮空间中，每个存贮单元的实际地址编码称为该单元的物理地址(用PA表示)。
- 把1MB的存贮空间划分成若干个逻辑段，每段最多64KB。
- 各逻辑段的起始地址必须能被16整除，即一个段的起始地址(20位物理地址)的低4位二进制码必须是0。

3.4 8086/8088的存储器组织

- 一个段的起始地址的高16位自然数为该段的段地址. 显然, 在1MB的存贮空间中, 可以有 2^{16} 个段地址. 每个相邻的两个段地址之间相隔16个存贮单元。
- 在一个段内的每个存贮单元, 可以用相对于本段的起始地址的偏移量来表示,, 这个偏移量称为段内偏移地址, 也称为有效地址(EA)。
- 段内偏移地址也用16位二进制编码表示. 所以, 在一个段内有 2^{16} =64K个偏移地址(即一个段最大为64KB)。

3.4 8086/8088的存储器组织

- 在一个64KB的段内，每个偏移地址单元的段地址是相同的.所以段地址也称为段基址。
- 由于相邻两个段地址只相隔16个单元，所以段与段之间大部分空间互相覆盖(重叠)。
- 存贮器段的划分与段的覆盖示意图如下图3.4.4所示。

3.4 8086/8088的存储器组织

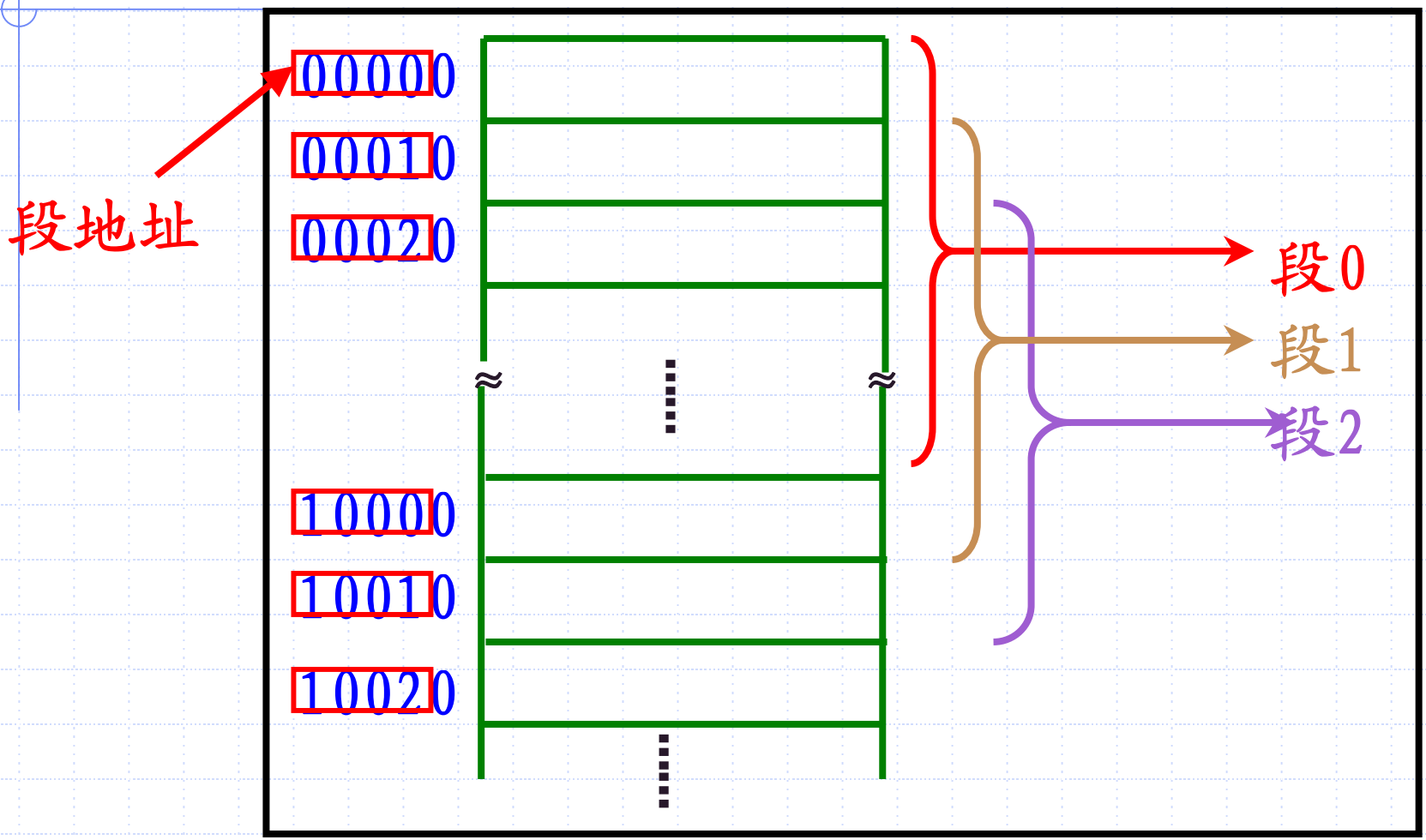


图3.4.4　存贮器段的划分与段的覆盖示意图

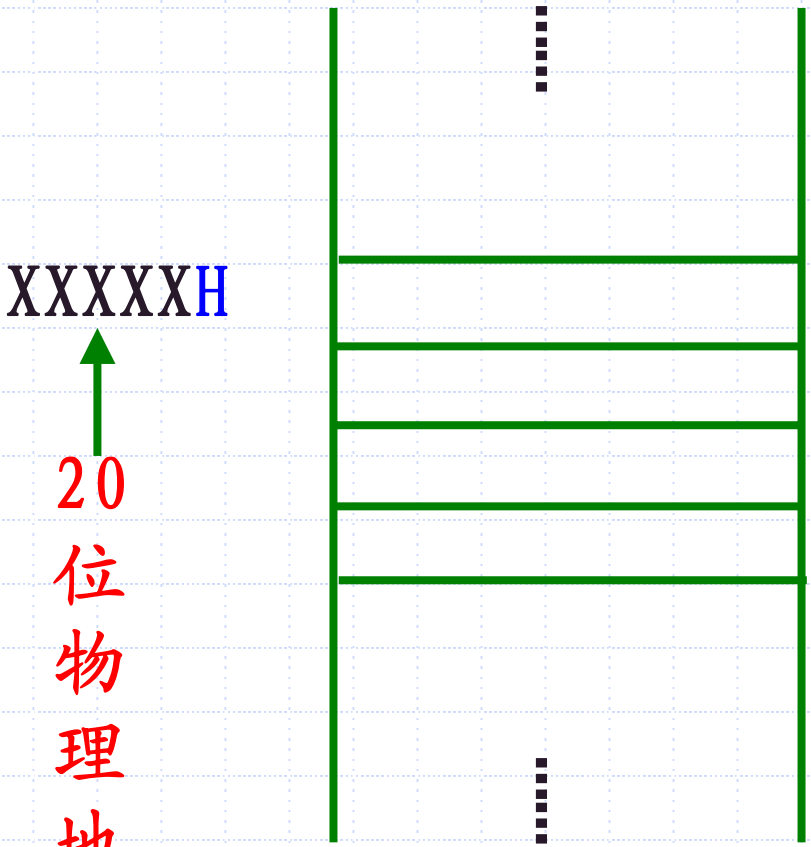
3.4 8086/8088的存储器组织

2. 物理地址(PA)的形成

把1MB的存贮空间分成若干个逻辑段以后，对一个段内的任意存贮单元，都可以用两部分地址来描述，一部分地址为段地址(段基址)，另一部分为段内偏移地址(有效地址EA)，段地址和段内偏移地址都是无符号的16位二进制数，常用4位十六进制数表示。这种方法表示的存贮器单元的地址称为逻辑地址。如下图3.4.5所示。

逻辑地址的表示格式为：段地址：偏移地址

3.4 8086/8088的存储器组织



段地址：段内偏移地址

XXXXH : XXXXH

逻辑地址

图3.4.5

3.4 8086/8088的存储器组织

一个存储单元用逻辑地址表示后，CPU 对该单元的寻址就应提供两部分地址 { 段地址 / 段内有效地址

其中段地址有段寄存器提供:

- CS —— 提供当前代码（程序）段的段地址
- DS —— 提供当前数据（程序）段的段地址
- ES —— 提供当前附加数据段的段地址
- SS —— 提供当前堆栈段的段地址

3.4 8086/8088的存储器组织

②段内偏移地址由下列地址寄存器提供:

BX
BP
SI
DI

→ CPU 对存储器进行数据读/写操作时，由这些寄存器以某种寻址方式向存储器提供段内偏移地址。

SP → 堆栈操作时，提供堆栈段的段内偏移地址

IP → CPU 取指令时，由IP提供所取指令代码所在单元的偏移地址。

3.4 8086/8088的存储器组织

- 已知某存储单元的逻辑地址,怎样求该单元的物理地址PA:

物理地址=段地址×10H+段内偏移地址

8086/8088 CPU中的BIU单元的地址加法器Σ用来完成物理地址的计算，其计算方法如图3.4.6所示。

3.4 8086/8088的存储器组织

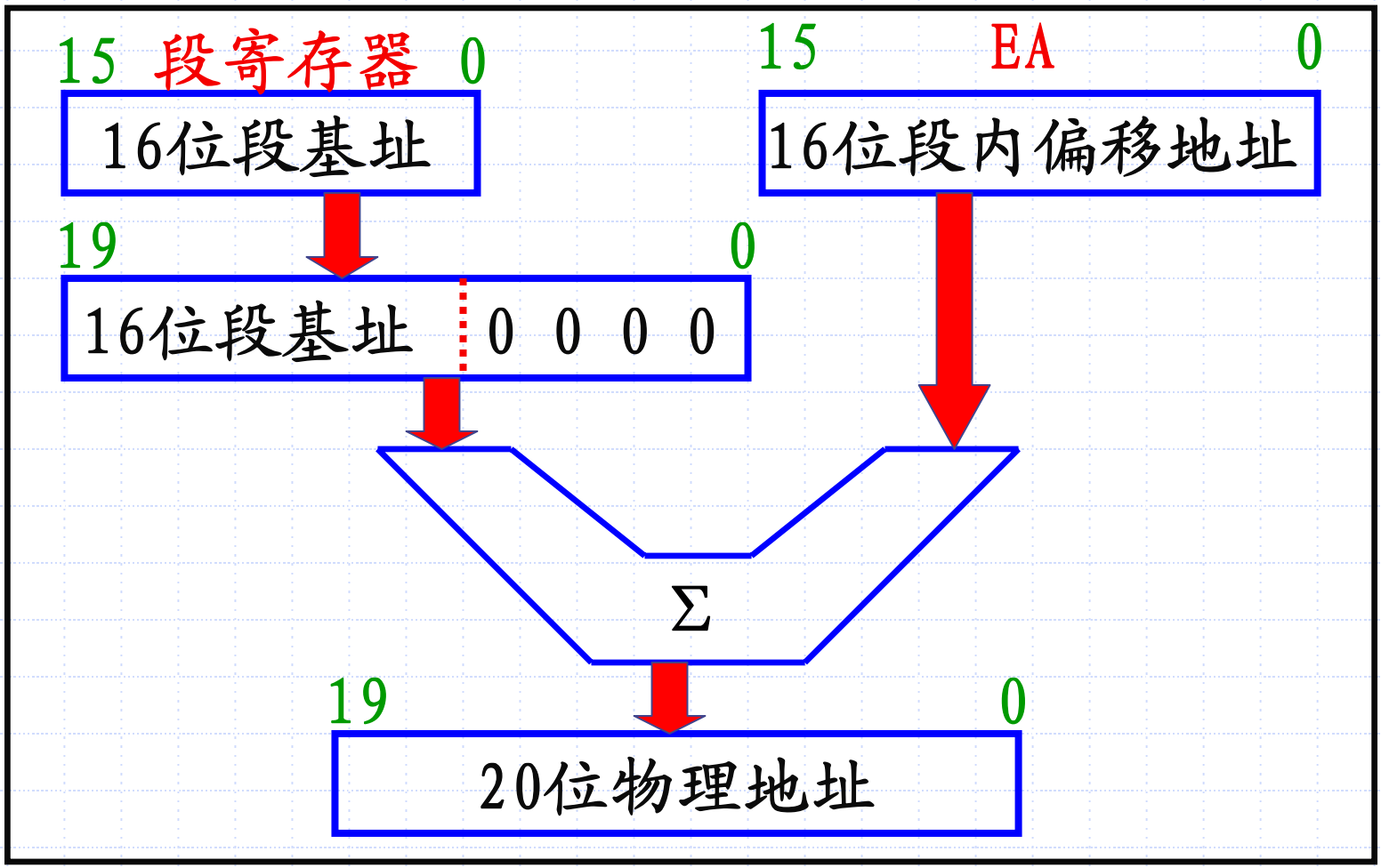


图3.4.6 物理地址的形成

3.4 8086/8088的存储器组织

例1：某单元的逻辑地址为4B09H:5678H，则该存储单元的物理地址为：

物理地址（PA）=段地址×10H+EA

=4B09H×10H+5678H

=4B090H+5678H

=50708H

3.4 8086/8088的存储器组织

例2：如图3.4.4中物理地址为00020H单元，其逻辑地址可以有：

[0000H:0020H]

00000H（段地址×16）

+ 0020H（EA）

00020H（PA）

[0001H:0010H]

00010H（段地址×16）

+ 0010H（EA）

00020H（PA）

3.4 8086/8088的存储器组织

[0002H: 0000H]

$$
\begin{array}{rl}
 & 00020\text{H}\ (\text{段地址}\times 16) \\
+ & 0000\text{H}\ (\text{EA}) \\
\hline
 & 00020\text{H}\ (\text{PA})
\end{array}
$$

由此可见，一个存储单元，若用不同的逻辑地址表示，其PA是唯一的。

3.4 8086/8088的存储器组织

在访问存储器时，段地址总是由段寄存器提供的。8086/8088微处理器的BIU单元设有4个段寄存器（CS、DS、SS、ES），所以CPU可以通过这4个段寄存器来访问4个不同的段。用程序对段寄存器的内容进行修改，可实现访问所有的段。

一般地，把段地址装入段寄存器的那些段（不超过4个）称为当前段。

3.4 8086/8088的存储器组织

三. 信息的分段存储与段寄存器的关系

特别要指出的是，用户用8086/8088汇编语言编写程序时，要把程序中的不同信息安排在不同的段，也就是说，用户源程序汇编后在存储器中存放是按照不同的信息放在不同的逻辑段。而程序中的信息包括:

- 程序（代码）信息
- 数据信息
- 堆栈信息

3.4 8086/8088的存储器组织

其中，代码信息 → 存放在代码段，其地址由CS:IP提供。

堆栈信息 → 存放在堆栈段，其地址由SS:SP提供。

数据信息 → 通常情况下，存放在数据段（段地址由DS提供），当然也可以存放在附加数据段（段地址由ES提供），其段内偏移地址依据寻址方式的不同来求得。

3.4 8086/8088的存储器组织

8086/8088CPU各种类型访问存储器时，其地址成分的来源见下表所示。

访问存储器类型	默认段地址	可指定段地址	段内偏移地址来源
取指令码	CS	无	IP
取堆栈操作	SS	无	SP
字符串操作源地址	DS	CS, ES, SS	SI
字符串操作目的地址	ES	无	DI
BP用作基址寄存器时	SS	CS, DS, ES	依据寻址方式求得有效地址EA
一般数据存取	DS	CS, ES, SS	依据寻址方式求得有效地址EA

3.4 8086/8088的存储器组织

一个段最大空间为64KB，实际使用时，不一定能用到64KB。理论上分段时，相邻段之间大部分空间是相互重叠的，但实际上不会重叠。汇编程序对用户源程序汇编时，会将用户程序中不同信息段独立存放。如图3.4.7所示。

3.4 8086/8088的存储器组织

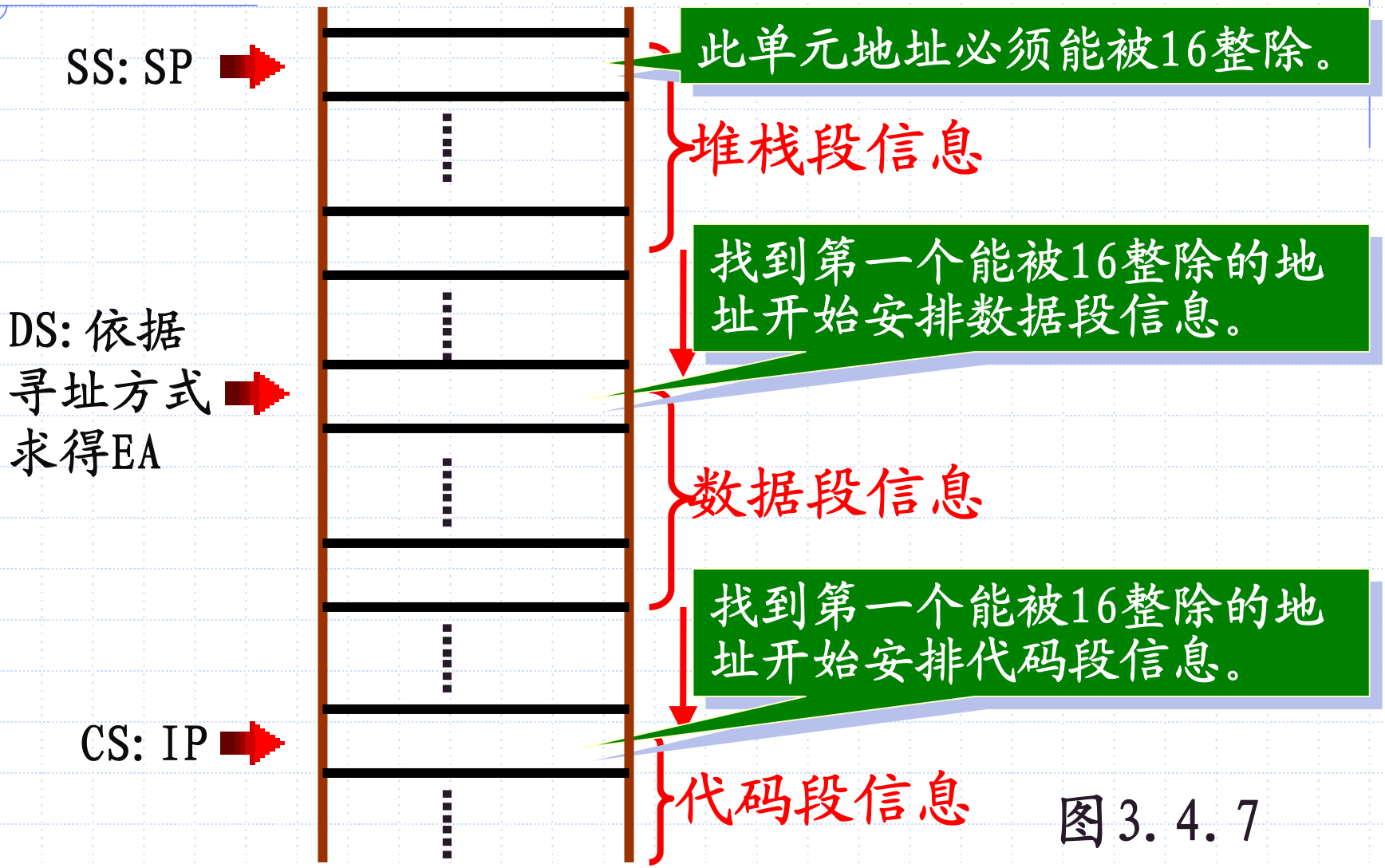


图3.4.7

3.5 8086/8088的I/O组织

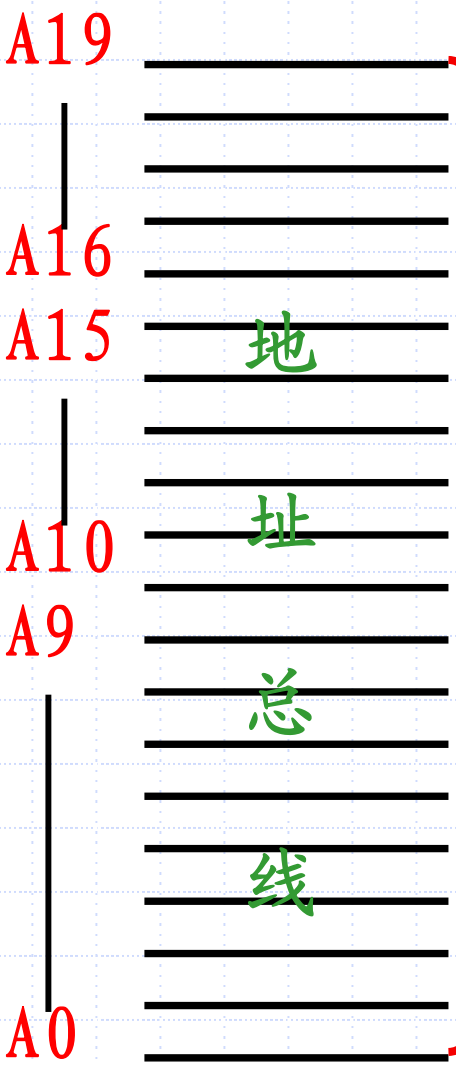
一.我们已经说明，I/O设备包括与外界通信和存储大容量信息用的各种外部设备。由于这些外部设备的复杂性和多样性，特别是速度比CPU低得多，因此I/O设备不能直接和总线相连接。 I/O接口是保证信息和数据在CPU和I/O设备之间正常传送的电路。

3.5 8086/8088的I/O组织

I/O接口与CPU之间的通信是利用称为I/O端口的寄存器来完成的，一个I/O端口有一个唯一的I/O地址与之对应。

二. 8086/8088CPU共有20条地址线，对存储器和I/O端口的寻址采用独立编址的方式，其关系如下：

3.5 8086/8088的I/O组织



全部用来给存储器编址，所以能寻址存储器空间。 $2^{20}=1MB$

低16条用来给I/O编址，所以能寻址 $2^{16}=64KB$ 的I/O空间。

PC系列机中，只用A9～A0 10条地址线给I/O编址，所以PC机的I/O空间为 $2^{10}=1KB$。

3.6 8086/8088的寻址方式

所谓寻址方式，就是指令中用于说明操作数所在地或者所在地地址的方法。

8088/8086的寻址方式分为两类：

- 关于寻找数据的寻址方式
- 关于寻找转移地址的寻址方式

3.6 8086/8088的寻址方式

下面讲关于数据的寻址方式时，均以数据传送指令MOV为例讲解。MOV指令格式如下：

指令完成的功能：　(DST)←(SRC)

3.6 8086/8088的寻址方式

一. 关于寻找数据的寻址方式（共8种）

数据的寻址方式就是告诉CPU存/取数据的地方。

1. 立即寻址

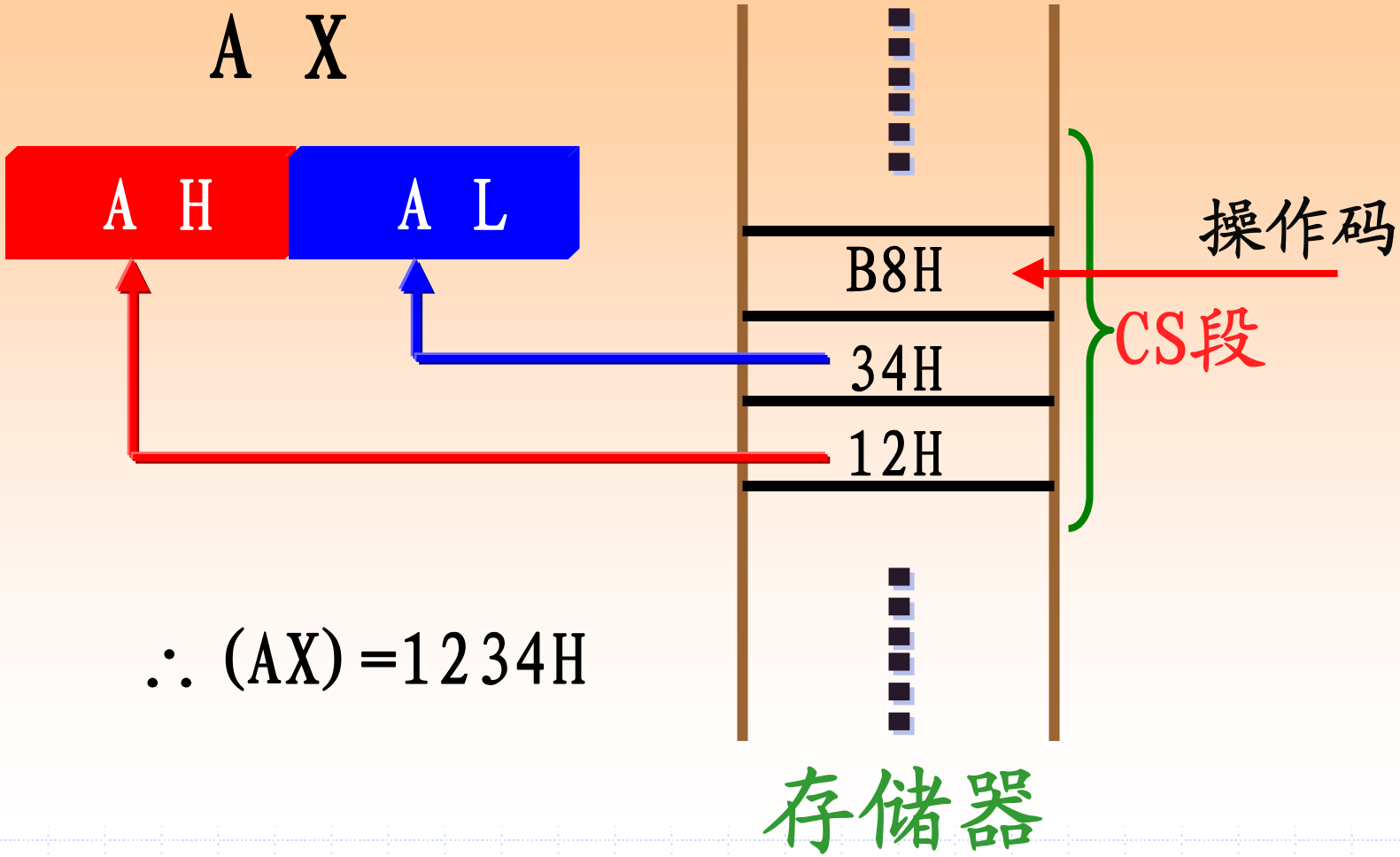
操作数直接存放在指令中，紧跟在操作码之后，作为指令的一部分，存放在代码段里，这种操作数称为立即数。

立即寻址主要用来给REG或M赋初值。

注意：只能用于源操作数字段，不能用于目的操作数字段。如：**MOV 12H , AL** ✗ **(语法错误)**

3.6 8086/8088的寻址方式

例：MOV AX，1234H



∴ (AX)=1234H

3.6 8086/8088的寻址方式

2. 寄存器寻址

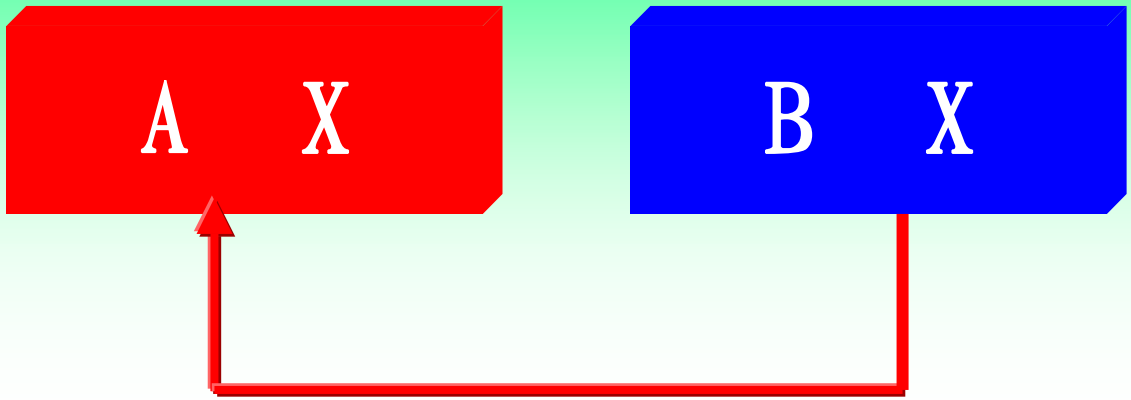
数据放在指令规定的寄存器中，对16位数据，REG可以是AX、BX、CX、DX、SI、DI、SP、BP以及段寄存器，而对于8位数据， REG可以是AH、AL、BH、BL、CH、CL、DH、DL。

在程序设计中，一般存放数据时，寄存器选择通用寄存器，而存放结果时尽可能的使用AX累加器，因为使用AX累加器要比用其它寄存器存放结果，指令执行时间要短一些。

寄存器寻址既可以作DST,也可以作SRC。

3.6 8086/8088的寻址方式

例：MOV　AX，BX



若（AX）=1234H，(BX)=5678H，则CPU执行上条指令后，（AX）=5678H，而（BX）不变。

又如：MOV　CX，DL ✗ (语法错误)

错误原因：类型不一致。

3.6 8086/8088的寻址方式

3. 存储器寻址

这类寻址方式，操作数在存储器中，而存储器单元的地址由以下五种寻址方式的任何一种均可以找到。但在指令中给出的只是要寻找的操作数所在单元的段内偏移地址，而操作数所在单元的段地址除非指令中用段超越前缀特别指明，否则是默认的。

3.6 8086/8088的寻址方式

①直接寻址

指令中直接给出了要寻找操作数所在单元的16位偏移地址。

- 指令中直接给出的操作数所在单元的16位偏移地址默认在数据段。也可以通过增加段超越前缀来改变操作数所在的段地址。

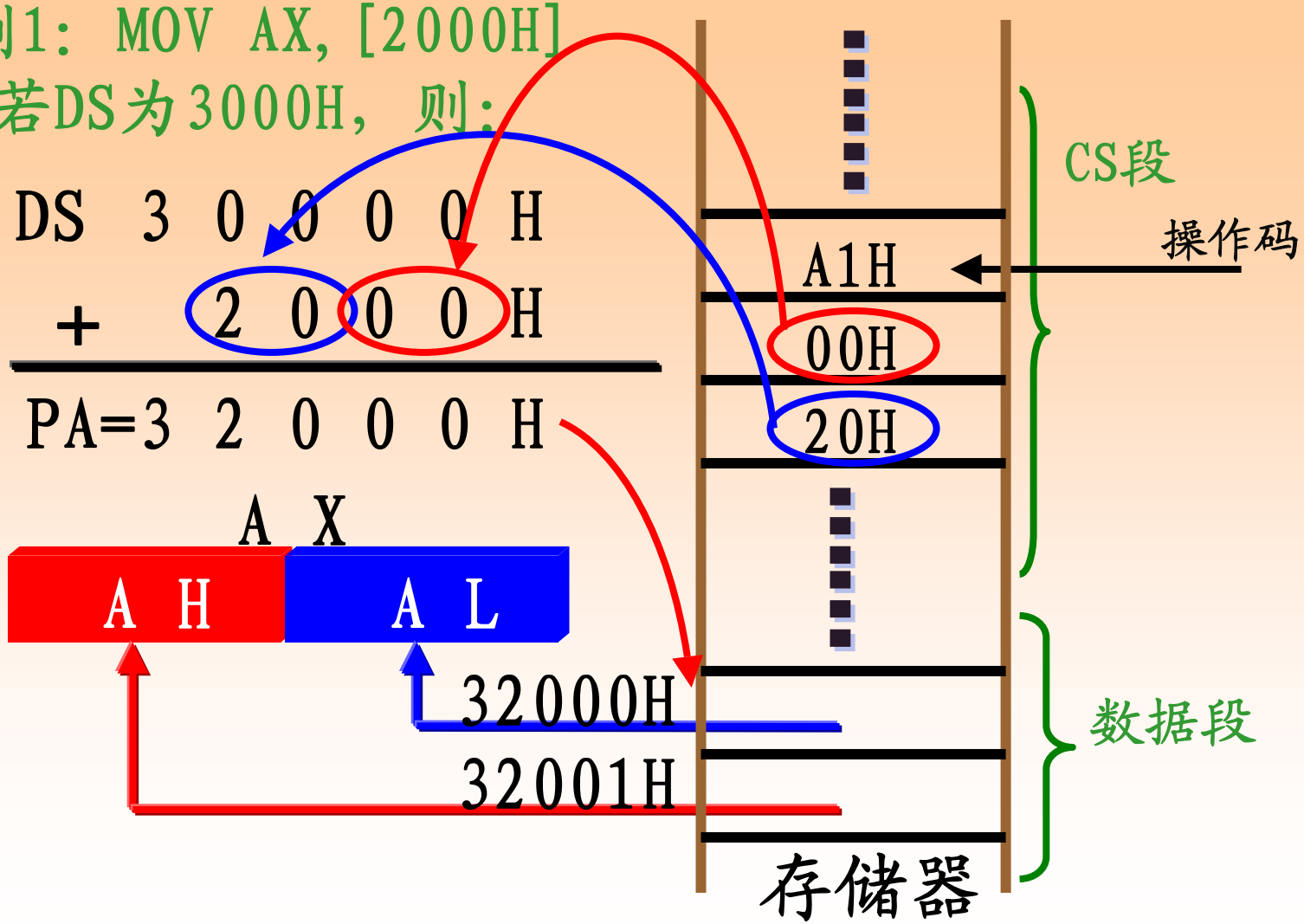
- 操作数所在单元的物理地址：

PA = (段寄存器) × 16 +指令中给出的偏移地址

3.6 8086/8088的寻址方式

例1：MOV AX,[2000H]

若DS为3000H，则：



3.6 8086/8088的寻址方式

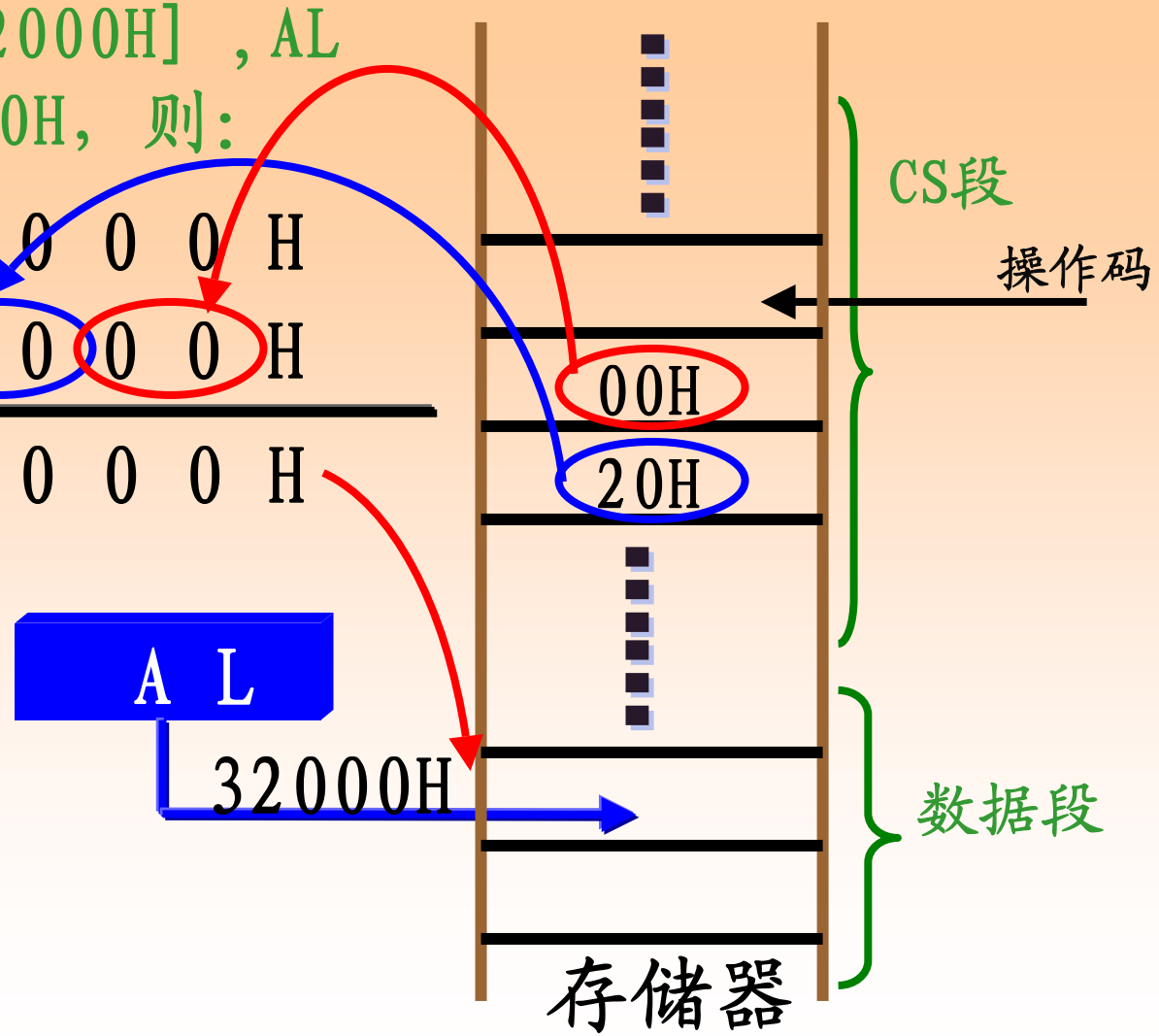
例2：MOV [2000H] ,AL

若DS为3000H，则：

DS 3 0 0 0 0 H

\+ 2 0 0 0 H

PA=3 2 0 0 0 H



3.6 8086/8088的寻址方式

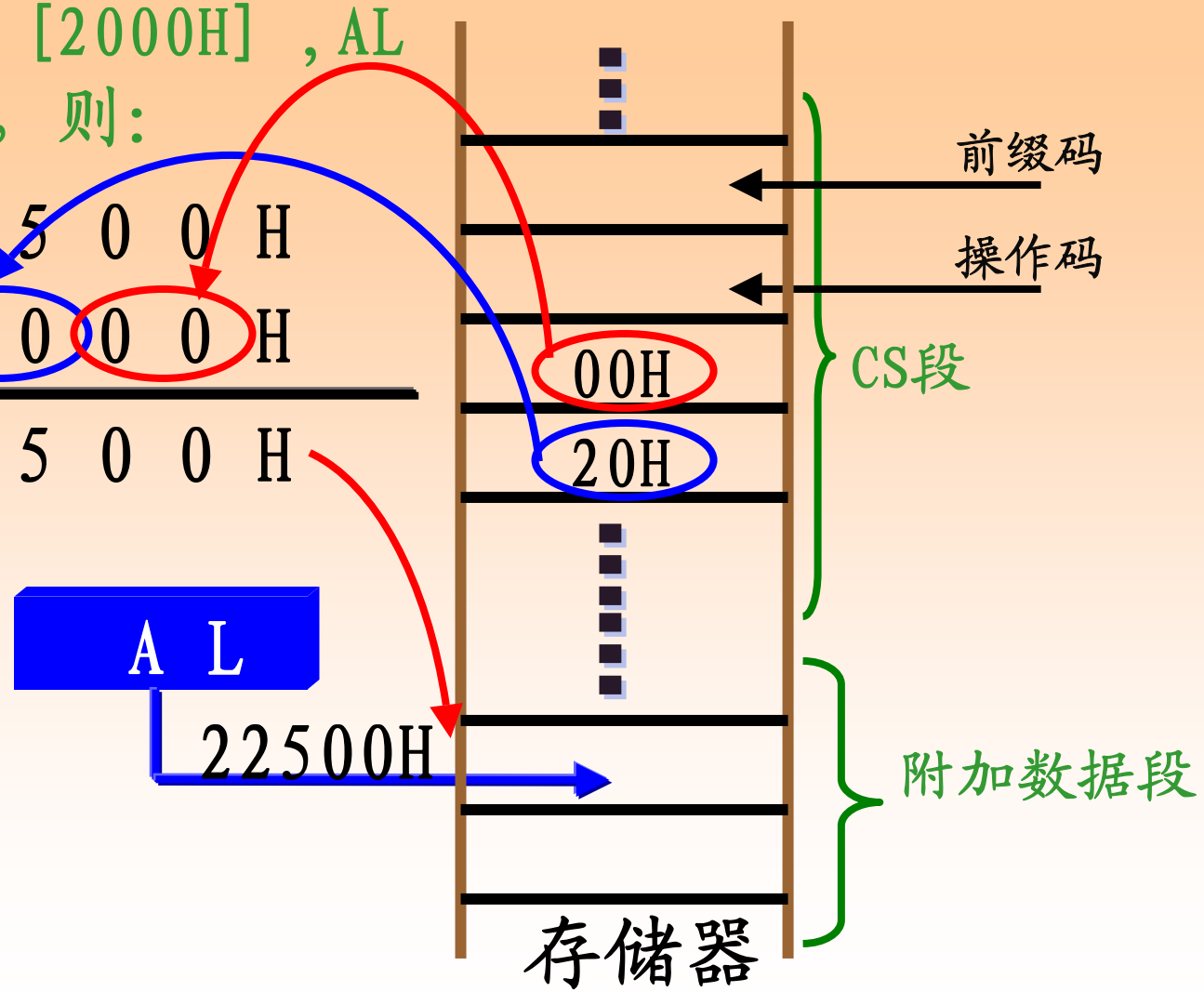
例3: MOV ES:[2000H] ,AL

若ES为2050H，则:

ES 2 0 5 0 0 H

\+ 2 0 0 0 H

PA=2 2 5 0 0 H



3.6 8086/8088的寻址方式

在实际的汇编语言程序设计中，如果程序比较复杂，而用到的存放数据的单元又很多，那么在直接寻址方式当中，用户就要记住存放数据的每个单元的地址，同时还要记住该地址单元存放的数据的意义，这样对设计程序带来了很大的困难。所以在实际的汇编语言程序设计中，常常采用给存放数据的单元，定义一个符号地址名，即变量名/变量。

3.6 8086/8088的寻址方式

- 变量名一但定义了，就具有了：

该单元的段地址
该单元的偏移地址
类型
长度
大小

} 五个属性

这样，在程序设计中就可以用这个变量名代替原来的存储器单元的实际地址。

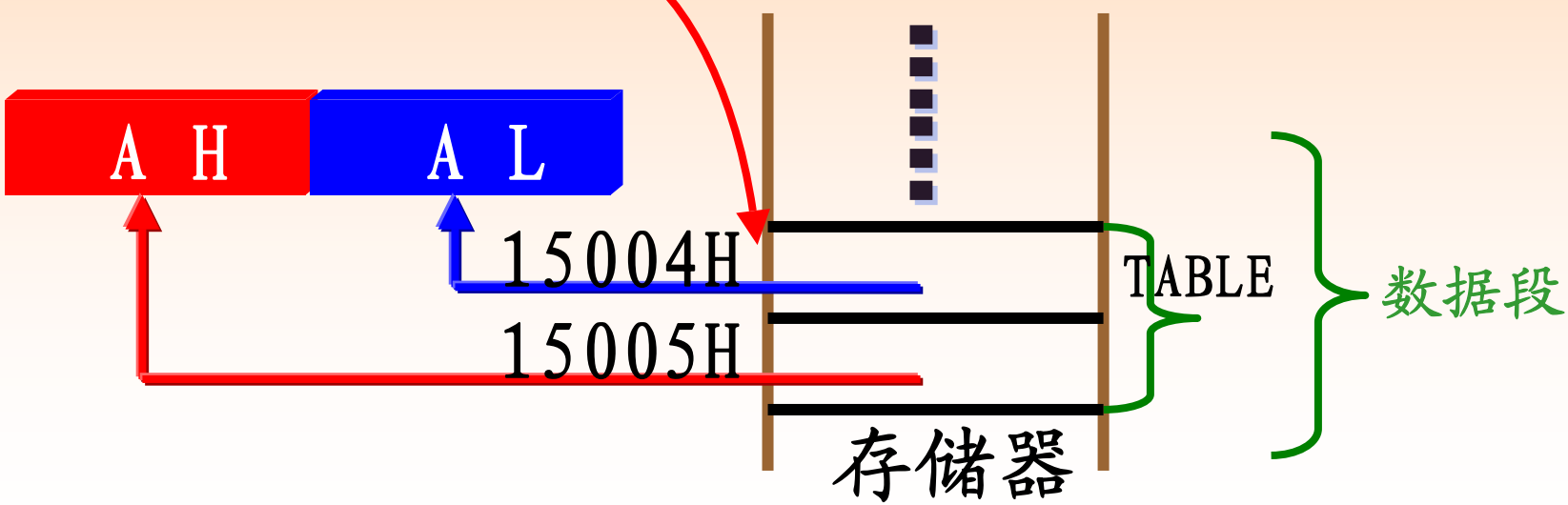
3.6 8086/8088的寻址方式

例4：若(DS)=1500H,TABLE为在DS段定义的一个字变量，且偏移地址为0004H。则CPU执行 MOV AX ,TABLE 指令完成的操作如下：

DS 1 5 0 0 0 H

\+ 0 0 0 4 H

PA=1 5 0 0 4 H



3.6 8086/8088的寻址方式

例5：若VAR1为字变量， VAR2和VAR3为字节变量，判断下列指令的书写格式是否正确，正确的说出SRC和DST的寻址方式，不正确说出错误原因。

MOV AX , VAR1	✓	SRC为直接寻址 DST为寄存器寻址
MOV AX , VAR2	✗	类型不一致
MOV VAR2, VAR3	✗	两存储器单元之间不能直接传送数据
MOV [0200H] ,12H	✗	类型不明确

3.6 8086/8088的寻址方式

例6：将例5中语法不正确的语句改对。

MOV AX , VAR2 ✕ 类型不一致

改：MOV AL , VAR2

MOV VAR2, VAR3 ✕ 两存储器单元之间不能直接传送数据

改：
MOV AL ,VAR3
MOV VAR2 ,AL

MOV [0200H] ,12H ✕ 类型不明确

改：MOV BYTE PTR [0200H] ,12H

或者：MOV WORD PTR [0200H] ,12H

注：PTR为临时属性修改符。