第二章　IBM-PC微机的功能结构

本章主要内容：

- IBM-PC微机基本结构
- 8086/8088寄存器结构及其用途
- 8086/8088系统的存储器组织结构
- 8086/8088系统的堆栈及其操作方法

2.1 IBM-PC微机基本结构

一. 微机的一般构成

一般计算机应包括五大部件：

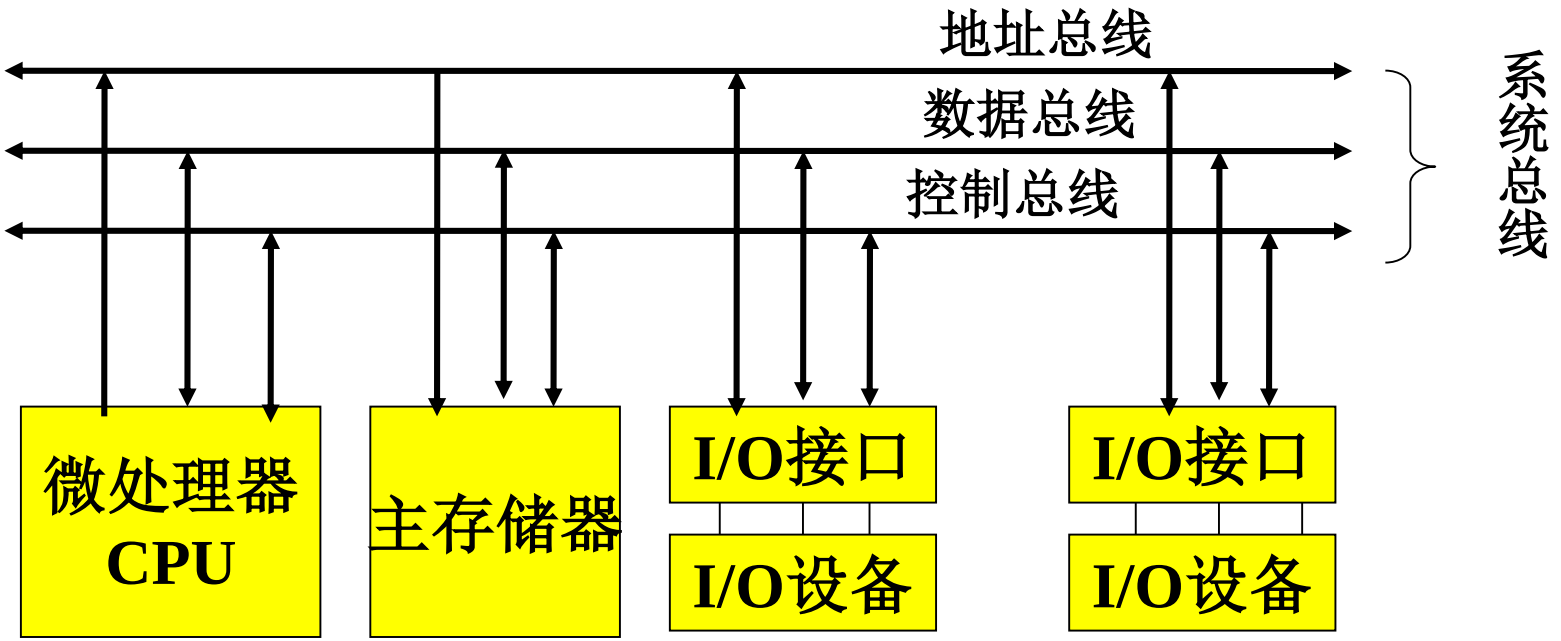
运算器、控制器、存储器、输入设备和输出设备。

由于微机的主要特点是其体积很小，因此在系统设计上就有一些特殊考虑。

➢将运算器和控制器两大部件集成在一个集成电路芯片上,称为中央处理器 ,简称CPU,也叫微处理器.

- 系统采用总线结构，具有较大的灵活性和扩展性。

微机硬件系统基本组成框图



1、中央处理器CPU

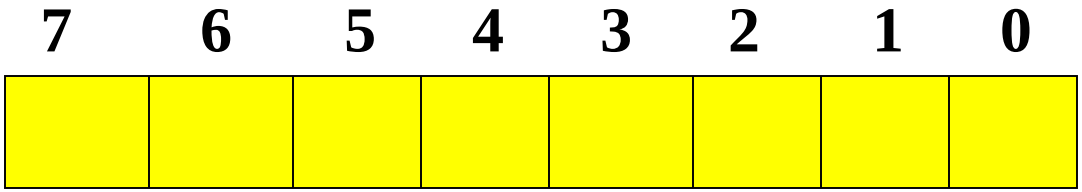
微型计算机中的中央处理器也叫微处理器。它包括运算器和控制器。

功能： 分析从主存储器取来的各条指令的功能，控制计算机各部件完成指定功能的各项操作。

2、主存储器

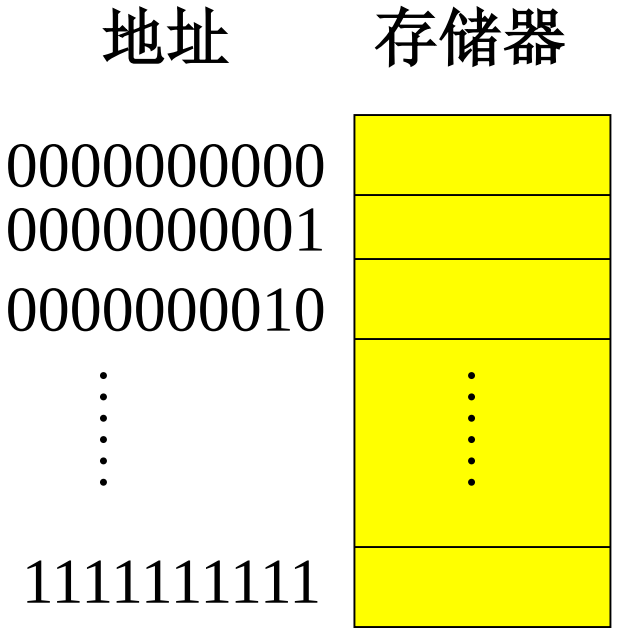
- 主存储器是用来存放程序和数据的部件。它由若干个存储单元构成。
- 存储单元的多少表示存储器的容量。每个存储单元使用一个唯一的编号来标识，称为存储单元的地址。
- 对每个存储单元内容的存和取是按照地址进行访问的。

计算机存储信息的基本单位是一个二进制位，一位可存储一个二进制数0或1。每8位组成一个字节（BYTE)。



在大多数计算机中，存储器的组织都是以字节为基本单位。每一个基本单位称为一个存储单元。

指示存储单元编号的地址长度决定了存储器的最大容量，例如一个10位二进制数表示的地址，可以用来区分2^{10}＝1024＝1K个单元。



习惯上将CPU与主存储器合称为主机

在计算机中，除了主存储器之外，一般还配置有辅助存储器，简称辅存。由于它的位置是在主机之外，因此也叫做外存。

3、输入输出设备及接口

- 输入设备将外部信息（程序、数据和命令）送入计算机。包括键盘、鼠标等。

- 输出设备将计算机处理后的结果转换为人或其它系统能识别的信息形式向外输出。如显示器、打印机等。

- 有的设备既具有输入功能又具有输出功能。如磁盘、磁带、触摸显示屏等。

- 由于I/O设备的工作速度、工作原理以及所处理的信息格式等与主机相差很大，因此I/O设备要通过I/O接口才能与系统总线连接。
- I/O接口是主机与I/O设备之间设置的逻辑控制部件。通过它实现主机与I/O设备间的信息传送。

4、系统总线

- 系统总线将CPU、存储器和I/O设备连接起来，实现各大部件之间的各种信息传送。
- 系统总线包括地址总线、数据总线和控制总线三组。它们分别用于传送不同的信息。

二、 Intel 8086/8088 CPU的功能结构

汇编语言程序是由一系列的指令(指令序列)构成。

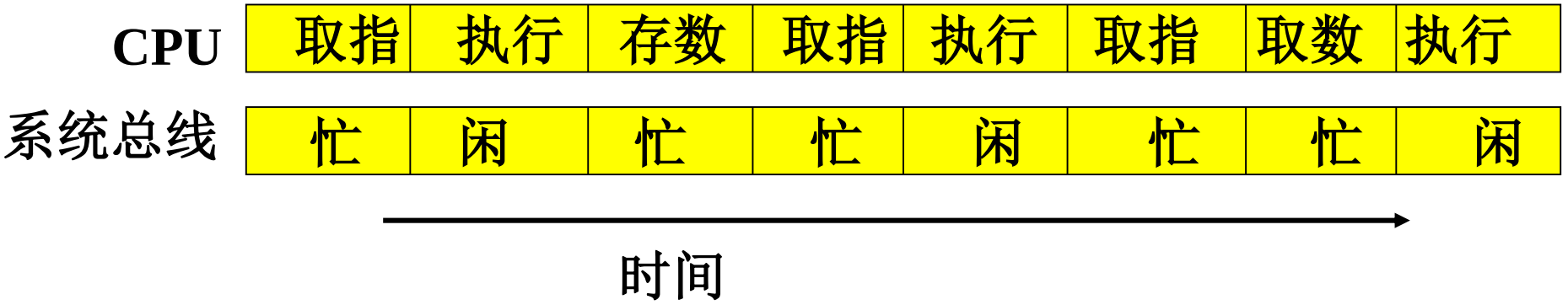
指令是构成汇编语言程序的最基本单位，就象高级语言中的语句。

CPU执行指令序列就是重复执行以下两个步骤：

- 从存储器中取指令
- 执行指令规定的操作

这两个步骤的执行又分为串行方式和指令流水线方式。

1. 串行方式



特点：

（1）当CPU在指令执行阶段，不需要占用系统总线，但此时总线也不工作，因此系统总线的空闲时间比较多 。

（2）在从存储器取指令、取数据或存数据时，总线处于忙状态，其所占用的时间也较长。而CPU却只需要花很短的时间去处理，因此大部分时间处于闲置状态。

采用串行工作方式的计算机其运行速度较慢

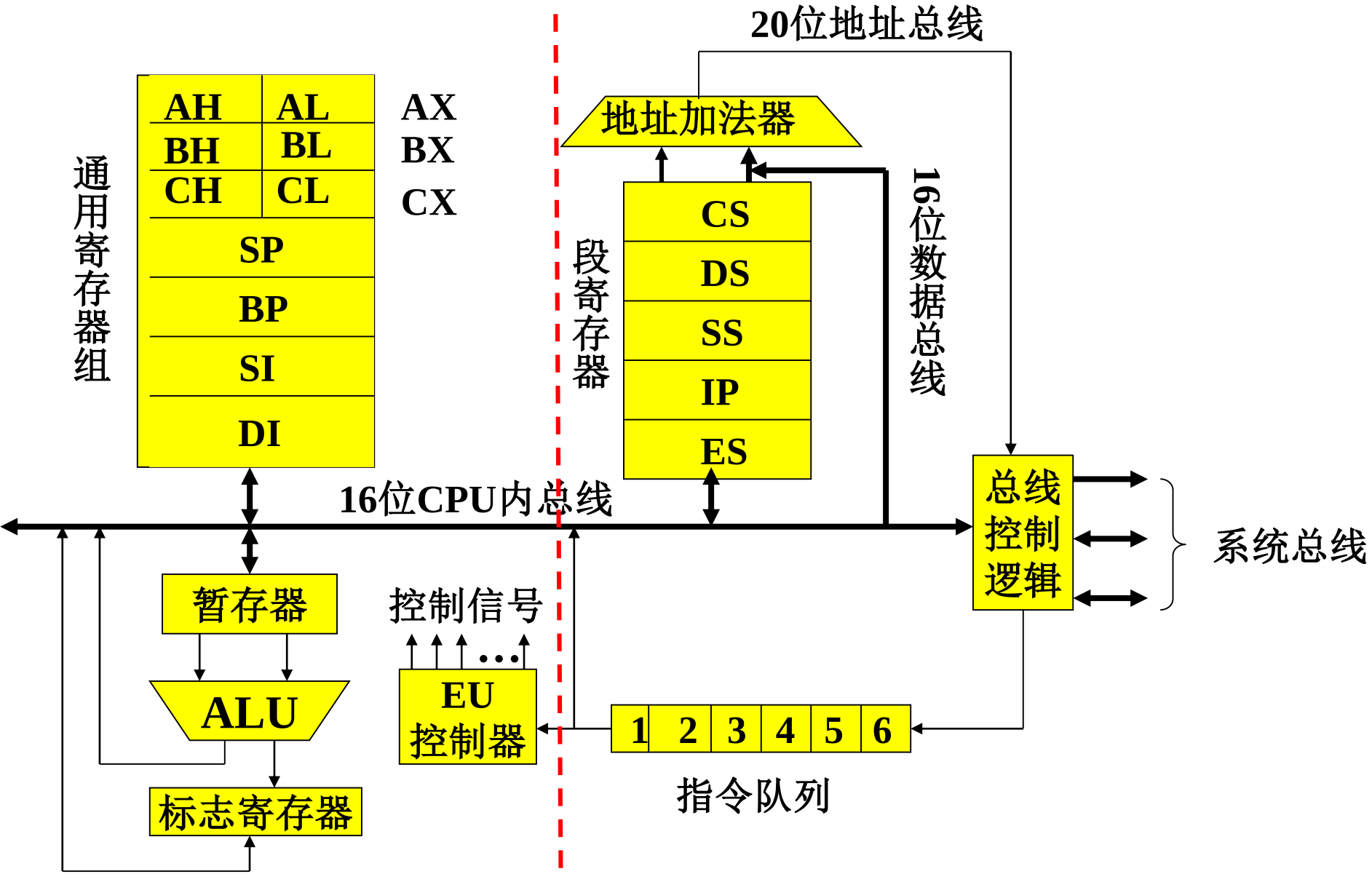
2.指令流水线方式

采用指令流水线工作方式的计算机具有较高的工作效率。CPU内部采用了一种先进的指令流水线结构，这种结构可以有效而充分地利用各主要硬件资源。

指令流水线结构最先出现在Intel公司的8086/8088 CPU中

要实现指令流水线方式，从CPU组成结构上要划分成多个单元。8086/8088CPU被划分成两个单元。

8086CPU结构


20位地址总线
AH
AL
AX
BH
BL
BX
CH
CL
CX
SP
BP
SI
DI
通用寄存器组
地址加法器
CS
DS
SS
IP
ES
段寄存器
16位数据总线
16位CPU内总线
总线控制逻辑
系统总线
暂存器
ALU
标志寄存器
控制信号
EU控制器
1
2
3
4
5
6
指令队列
执行单元（EU）
总线接口单元（BIU）

（1）执行单元EU

EU的主要任务是分析与执行指令，具体包括：

A、从指令队列中取出指令代码，由控制器译码后产生相应的控制信号，控制各部件完成指令规定的操作。

B、对操作数执行各种指定的算术或逻辑运算。

C、向总线接口单元BIU发送访问主存或I/0的命令，并提供相应的地址和传送的数据。

（2）总线接口单元BIU

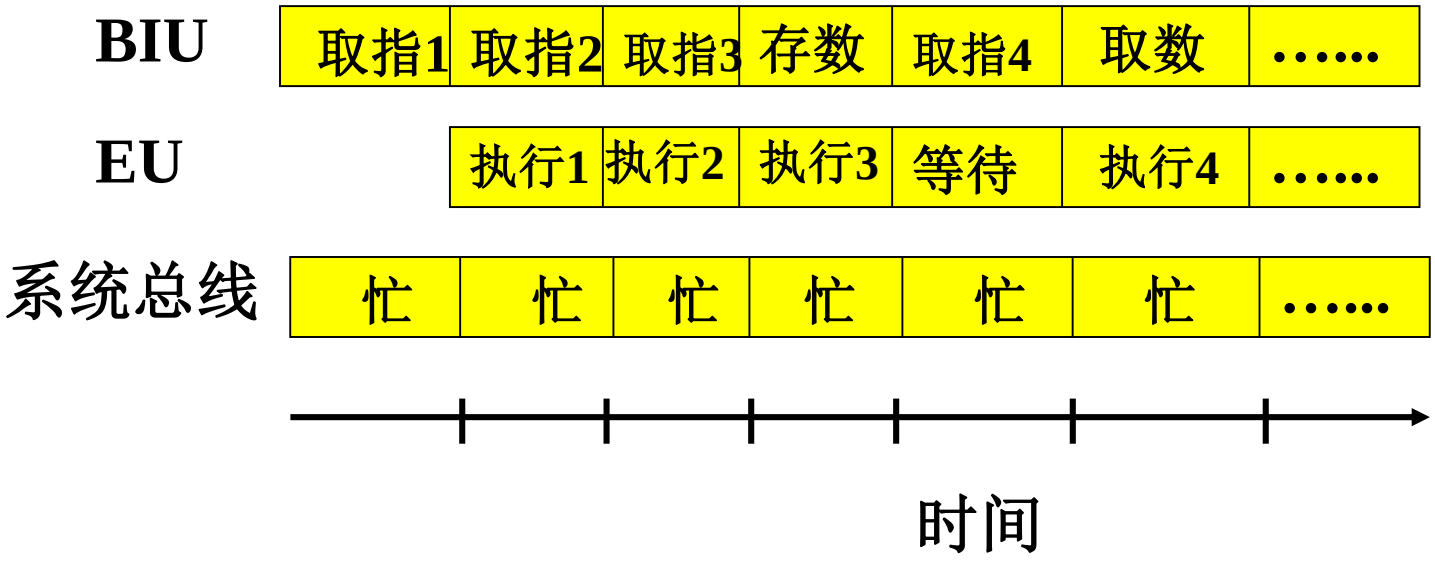
BIU负责CPU与存储器、I/O的信息传送。具体包括：

A、取指令——根据CS寄存器和指令指针IP形成20位的物理地址，从相应的存储器单元中取出指令，暂存到指令队列中，等待EU取走并执行。

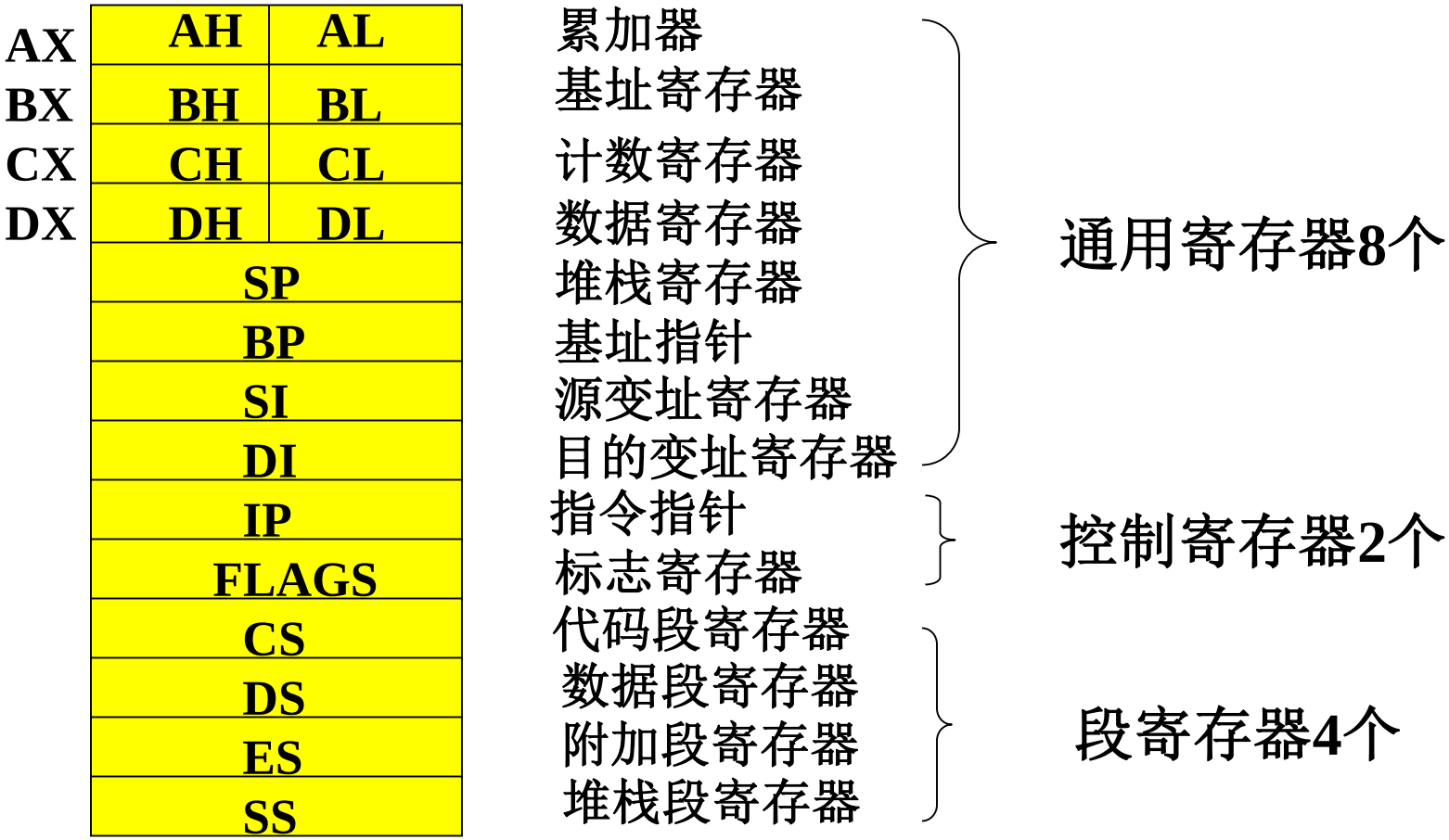
B、存取数据——在EU执行指令的过程中，需要与存储器或I/O端口传送数据时，由EU提供的数据和地址，结合段寄存器，通过外部总线与存储器或I/0进行数据的存取。

EU和BIU是既分工又合作的两个独立部分。它们的操作在一定程度上是并行工作的，分别完成不同的任务，从而大大加快了指令执行速度。

Intel 8086/8088 运行时执行过程大致如下图所示。



2. 2 Intel 8086/8088CPU寄存器结构及其用途



一、通用寄存器

Intel 8086/8088有8个16位通用寄存器，它们具有良好的通用性，并且还可以用作某个特定的功能，可以由程序设计人员进行编程访问。

1. 数据寄存器

它包括AX、BX、CX和DX四个寄存器。它们中的每一个既可以是16位寄存器，也可以分成两个8位寄存器使用。即可以当作8个独立的8位寄存器使用。

数据寄存器既可以用来存放参加运算的操作数，也可以存放运算的结果。在多数情况下，使用这些寄存器时必须在指令中明确指示。

例：MOV AX , BX；将BX的内容送到AX中

ADD CH , DH；将DH和CH的内容相加，结果送到CH

在有些指令中，不需要明确指出使用的寄存器名，即隐念使用了某寄存器，称为隐含使用。

例如，在循环指令 LOOP中，CX被隐含指定作循环次数计数用。

个别指令对寄存器有特定的使用，并且又必须在指令中指明它的名字，这类寄存器的使用称为特定使用。

例如，移位指令 SHL AX，CL

CL被固定用作移位次数。

2. 指针寄存器

指针寄存器有堆栈指针SP和基址指针BP

它们一般被用来存放16位地址，在形成20位的物理地址时常被作为偏移量使用。

SP指针——在进行堆栈操作时，被隐含使用，被用来指向堆栈顶部单元。

BP指针——被用来指向堆栈段内某一存储单元。BP除用作地址指针外也可以象数据寄存器一 样，存放参加运算的操作数和运算的结果。

3. 变址寄存器

有两个16位的变址寄存器SI和DI，一般被用来作地址指针。

SI——源变址寄存器　　DI——目的变址寄存器

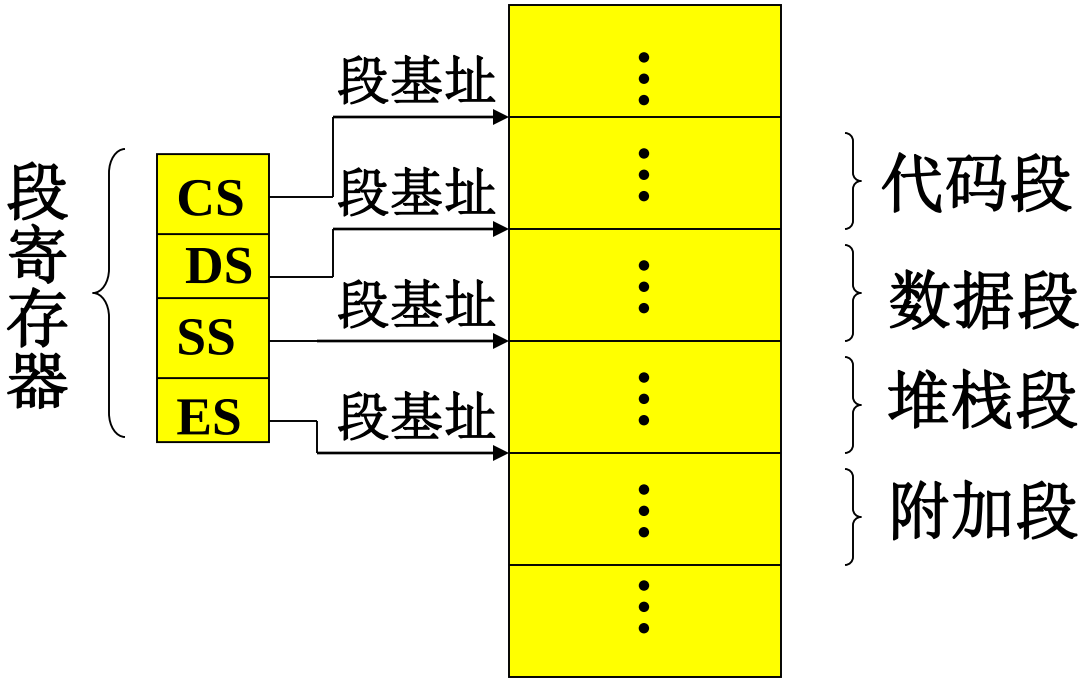
同BP寄存器一样，SI和DI也可以用作通用数据寄存器存放操作数和运算结果。

二、段寄存器

✓8086/8088CPU在使用存储器时，将它划分成若干个段。

✓每个段用来存放不同的内容，如程序代码、数据等等。

✓每个存储段用一个段寄存器来指明该段的起始位置（也叫段基址）。



CPU在访问存储器时必须指明两个内容：

（1）所访问的存储单元属于哪个段，即指明使用的段寄存器。

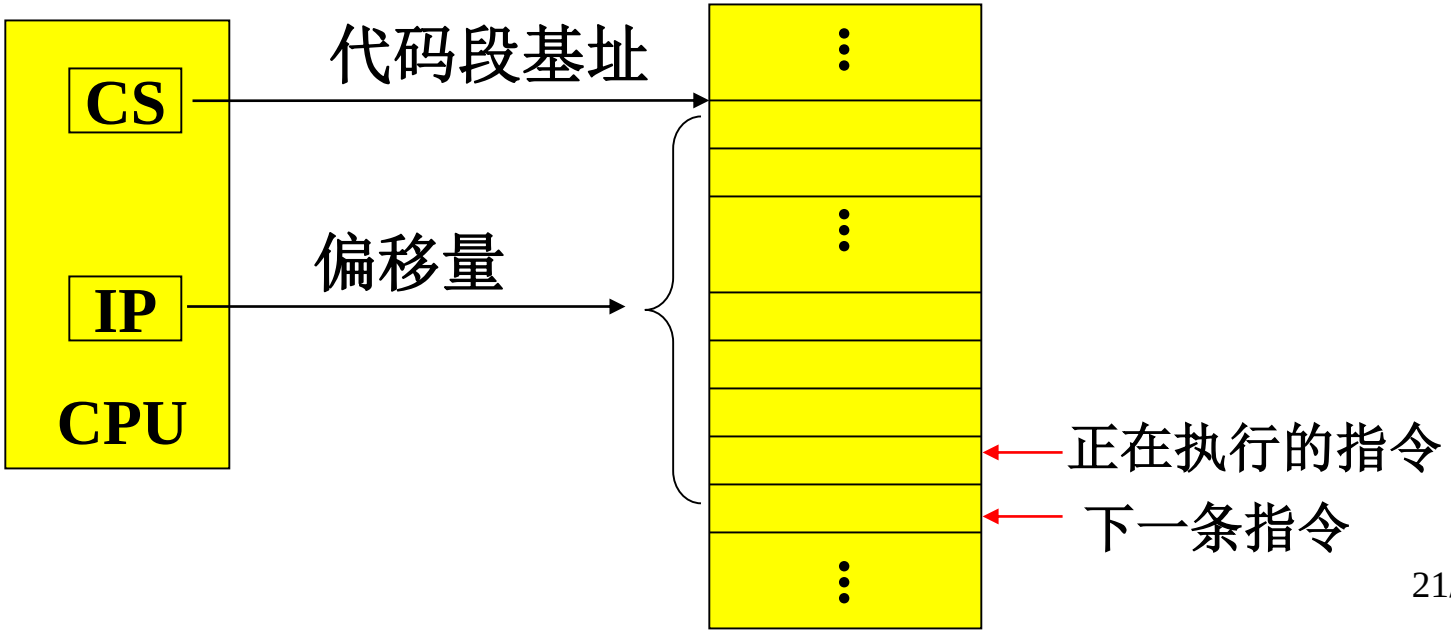
（2）该存储单元与段起始地址（段基址）的相距多少，即偏移量。

在程序设计中，一个程序将存储器划分成多少个存储段是任意的。但在程序运行的任何时刻最多只有4个段，分别由CS、DS、ES和SS指定。

三、指令指针IP

CPU在从存储器取指令时，以段寄存器CS作为代码段的基址指针，以IP的内容作为偏移量，共同形成一条指令的存放地址。

当CPU从内存中取出一条指令后，IP内容自动修改为指向下一条指令。



注意：IP的内容不能被直接访问，既不能用指令去读IP的值，也不能用指令给它赋值。但是可以通过某些指令的执行而自动修改IP的内容。

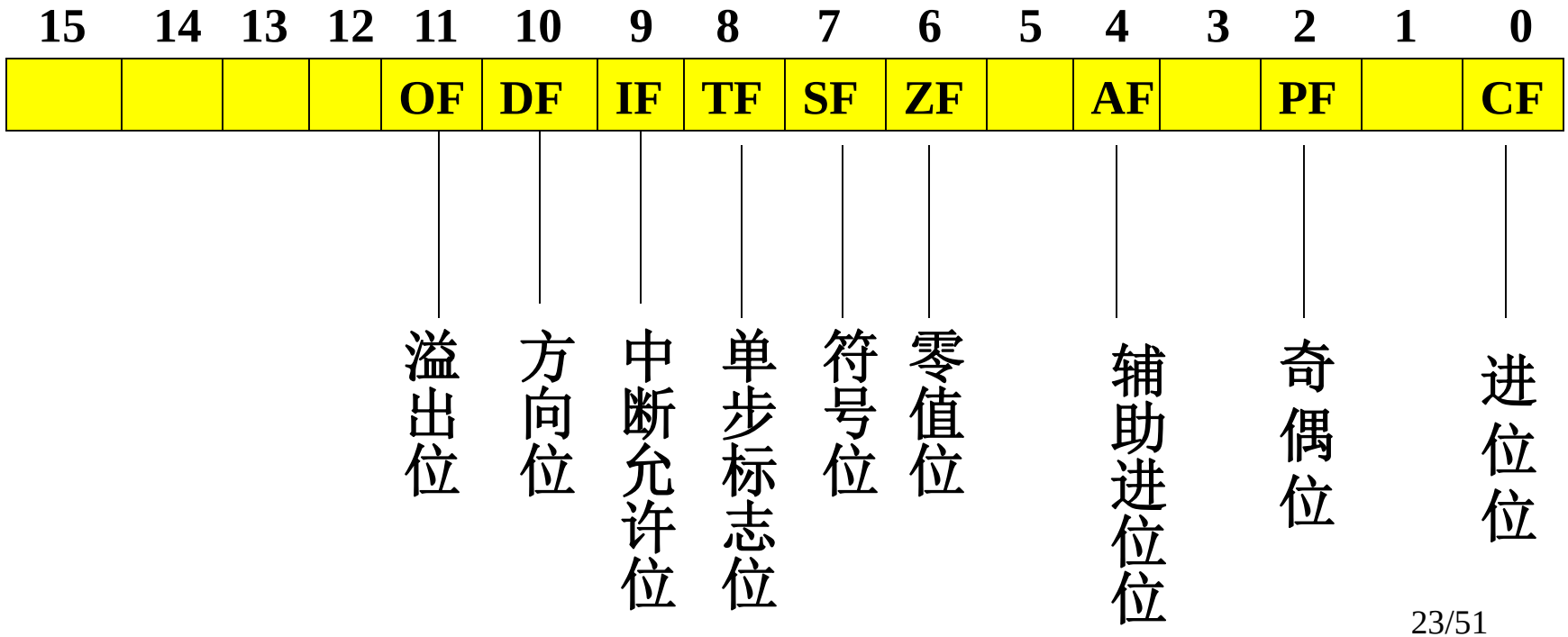
例如，下面两种指令就可以自动改变IP寄存器的内容。

- **转移指令将指令中的目的地址的偏移量送入IP**
- **子程序调用指令CALL，将IP原有内容自动压入堆栈，而将子程序的入口地址偏移量自动送入IP，而返回指令RET，又自动从堆栈中弹回原有IP的内容。**

四、标志寄存器

标志寄存器是用来反映CPU在程序运行时的某些状态，如是否有进位、奇偶性、结果的符号、结果是否为零等等。

8086/8088CPU中标志寄存器的长度为16位，但只定义了其中的9位。



标志位分为
- 状态标志：CF, PF, AF, ZF, SF, OF
- 控制标志：TF , DF, IF

1. 进位标志位CF

在进行算术运算时，若最高位（对字操作是第15位，字节操作是第7位）产生进位或借位时CF被自动置“1”，否则置“0”

在移位类指令中，CF也被用来存放从最高位（左移时）或最低位（右移时）移出的数值（0或1）。

2. 奇偶标志位PF

当指令操作结果的低8位中含有1的个数为偶数时，则PF被置1，否则PF被置0。

注意：PF只反映操作结果的低8位的奇偶性，与指令操作数的长度无关。

3. 辅助进位标志位AF

在进行算术运算时，若低字节的低四位向高4位产生进位或借位，即第3位产生进位或借位时，AF位被置1，否则置0。AF标志位用于十进制运算的调整。

注意：AF只反映运算结果低八位，与操作数长度无关。

4. 零值标志位ZF

若运算结果各位全为0，则ZF被置1，否则置0。

5. 符号标志位SF

将运算结果视为带符号数，当运算结果为负数时SF被置1，为正数时，则置0。

由于第7位是字节操作数的符号位，而第15位是字操作数的符号位，因此，SF位与运算结果的最高位（第7位或第15位）相同。

6.溢出标志位OF

当运算结果超过机器用补码所能表示数的范围时，则OF置1，否则置0.

字节数据，机器用补码所能表示的数范围为-128～+127。
字数据的表示范围为：-32768～+32767

注意：溢出与进位是两个完全不同的概念，不能相互混淆。
下面通过几个例子来说明

例如：计算 -85D+（-1D）= - 86D

进位被丢弃

```
    10101011  B
  +）11111111  B
-------------------
 1 ←10101010  B    -86D
```

CF=1,OF=0, 结果正确。

计算 100 D+100 D =200 D

```
    01100100 B
  +）01100100 B
-------------------
    11001000 B  -56 D
```

CF=0，OF=1, 结果发生溢出，即结果出错。

计算 -85 D +-117 D = -202 D

10101011 B

+）10001011 B

1←00110110 B 54 D

CF=1, OF=1,结果发生溢出，即结果出错。

7. 单步标志位TF（Trace Flag）

单步标志也叫跟踪位，该标志为控制标志位。单步标志位供调试程序使用。

当TF位被设置为1时，每执行一条指令后，CPU暂停运行，即产生单步中断。

8. 中断允许标志位IF

该标志位为控制标志位。当IF被设置为1时，CPU可以响应可屏蔽中断，否则不允许响应可屏蔽中断。

9. 方向标志位DF

DF也是控制标志位。它被用来规定串操作指令的增减方向。

当DF=0时，串操作指令自动使变址寄存器（SI和DI）的内容递增。当DF=1时，串操作指令自动使变址寄存器的内容递减。

2.3 存储器组织结构

一、存储器的组成

1. 存储器是由若干个存储单元构成

存储单元的多少就表示了存储器的容量。

2. 每个存储单元存放相同长度的二进制数

一个存储单元的长度一般为8位二进制数，即一个字节。

3. 每个存储单元有一个唯一的地址编号——地址

8086/8088CPU有20根地址线，即它可以产生20位的地址码，它的存储器寻址能力为2^{20}，即1兆字节空间。

这一兆字节存储单元的地址范围为：00……0~11……1。

20位　　20位

十六进制数地址	二进制数地址	存储单元（字节）
00000H	00000000000000000000	
00001H	00000000000000000001	
00002H	00000000000000000010	
⋮	⋮	⋮
FFFFEH	11111111111111111110	
FFFFFH	11111111111111111111	

7　　0

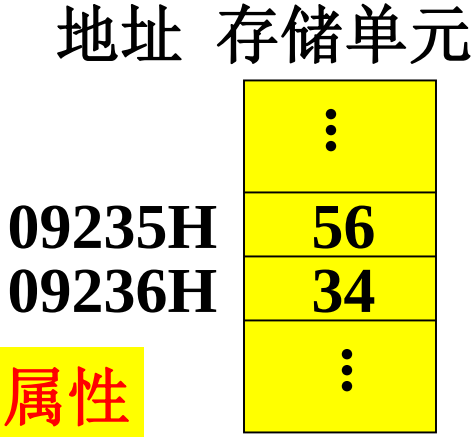
为了方便书写，在源程序中常用5位十六进制数或一个符号来表示一个存储单元的地址。

4. 任何两个相邻字节单元就构成一个字单元

一个字存储单元（WORD）的长度为16位二进制数，即两个字节。字单元的地址为两个字节单元中较小地址字节单元的地址。

16位长数据的存放规则是低8位放在较低地址字节单元中，高8位放在较高地址字节单元中。

例如，将数据3456H放在地址为09235H的存储单元中的存储分配。



5、在定义一个地址时必须指出是字节或字类型属性

由于存储单元可分为字单元和字节单元，因此8086/8088CPU访问内存的指令中，分为字节访问和字访问两种指令。

二、存储器的段结构

由于8086/8088可寻址的存储空间为1MB，需要提供20位长的地址码。而CPU内部的寄存器长度只有16位。能够直接访问的最大地址空间是64KB。

8086/8088系统的存储器段结构具有以下几个特点：

1. 8086/8088CPU将1MB的存储空间划分成若干个段，每个段最大长度为64K（65536）个字节单元组成。

在8086/8088的汇编语言源程序中，用户可以根据自己需要来设定段的个数、各个段长度和每个段的用途。并且代码或数据可以存放在段内任意单元中。

2. 每个段的基址（段基址）必须是一个小节的首址。

段基址——一个段的起始地址。

在存储器中规定从0地址开始，每16个字节单元称为一个小节（Paragraph）。因此，1MB内存就可划分为64K个小节。

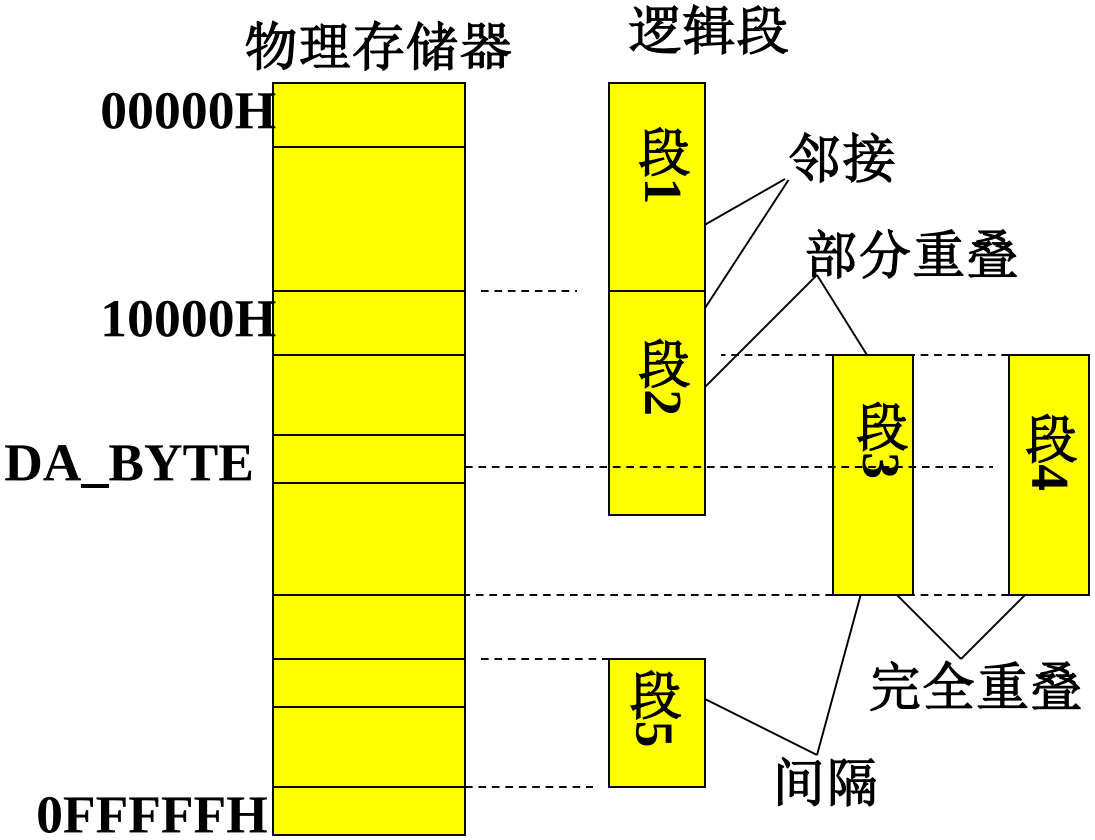
第 1 小节：00000H，00001H，00002H……0000FH
第 2 小节：00010H，00011H，00012H……0001FH
⋮ ⋮ ⋮ ⋮ ⋮
第65535小节：FFFE0H FFFE1H FFFE2H……FFFEFH
第65536小节：FFFF0H FFFF1H FFFF2H……FFFFFH

可以看出，每个小节的首地址最低位必为0（16进制数表示）。因此段基址只能是上述64K个小节首址之一。

3. 逻辑段在物理存储器中可以是邻接的、间隔的、部分重叠的和完全重叠的等4种情况。

逻辑段是指在汇编语言源程序中设置的段

内存中的一个物理存储单元可以映象到一个或多个逻辑段中



DA_BYTE物理单元可以映象到逻辑段2、段3和段4中。

4. 在任一时刻，一个程序只能访问4个当前段中的内容。

4个段分别是代码段、数据段、堆栈段和附加段，称为当前段 。4个段寄存器CS、DS、SS和ES分别保存了它们段基址的高16位地址，称为段基值。段基址的最低4位为0。（小节首址的低4位为全0 ）。

三、 逻辑地址与物理地址及对应关系

1. 物理地址

在1MB的存储空间中，每个存储单元的物理地址是唯一的，它就是该存储单元的20位地址。

8086/8088的物理地址范围：00000H~0FFFFFH

CPU与存储器之间的任何信息交换都使用物理地址。

2. 逻辑地址

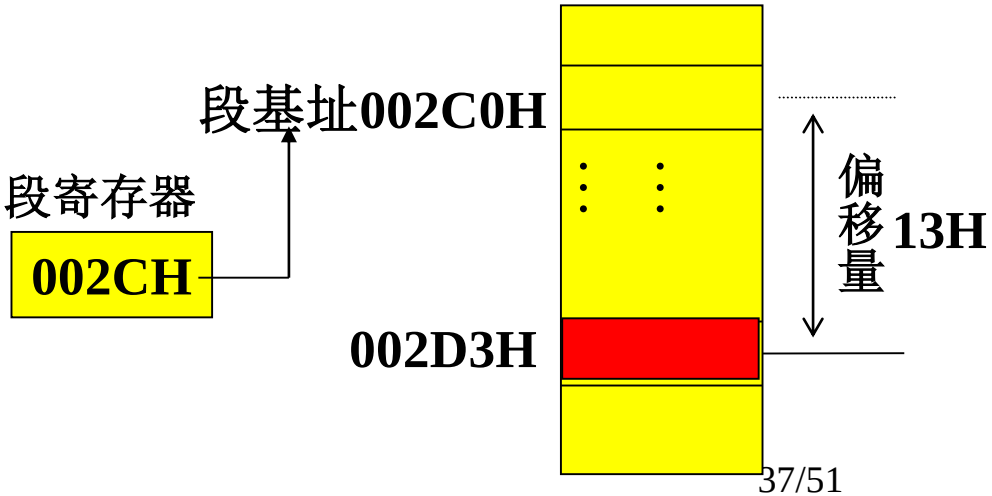
在程序设计中，为了便于程序的开发和对存储器进行动态管理，使用了逻辑地址。

一个逻辑地址包括两个部分：段基值和偏移量

段基值：存放在某一个段寄存器中，是一个逻辑段的起始单元地址（段基址）的高16位。

偏移量：表示某个存储单元与它所在段的段基址之间的字节距离。

对于一个64K的段，当偏移量为0时，就是这个段的起始单元，而偏移量为0FFFFH时，就是这个段的最后一个字节单元。



逻辑地址的表示方法　　段基值：偏移量

例如，3267H：0A0H

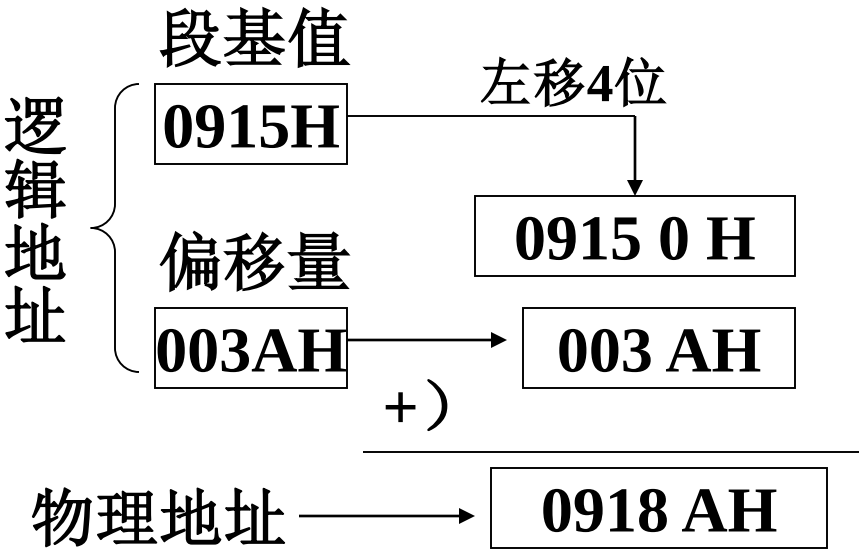
表示该逻辑单元位于段起始地址为32670H，段内偏移量为0A0H个字节。

3.逻辑地址转换为物理地址

当CPU要访问存储器时，需要由总线接口单元BIU将逻辑地址转换成物理地址。

转换方法：将逻辑地址的段基值左移4位，形成20位的段基址（低位为0）然后与16位的偏移量相加，结果即为20位的物理地址。

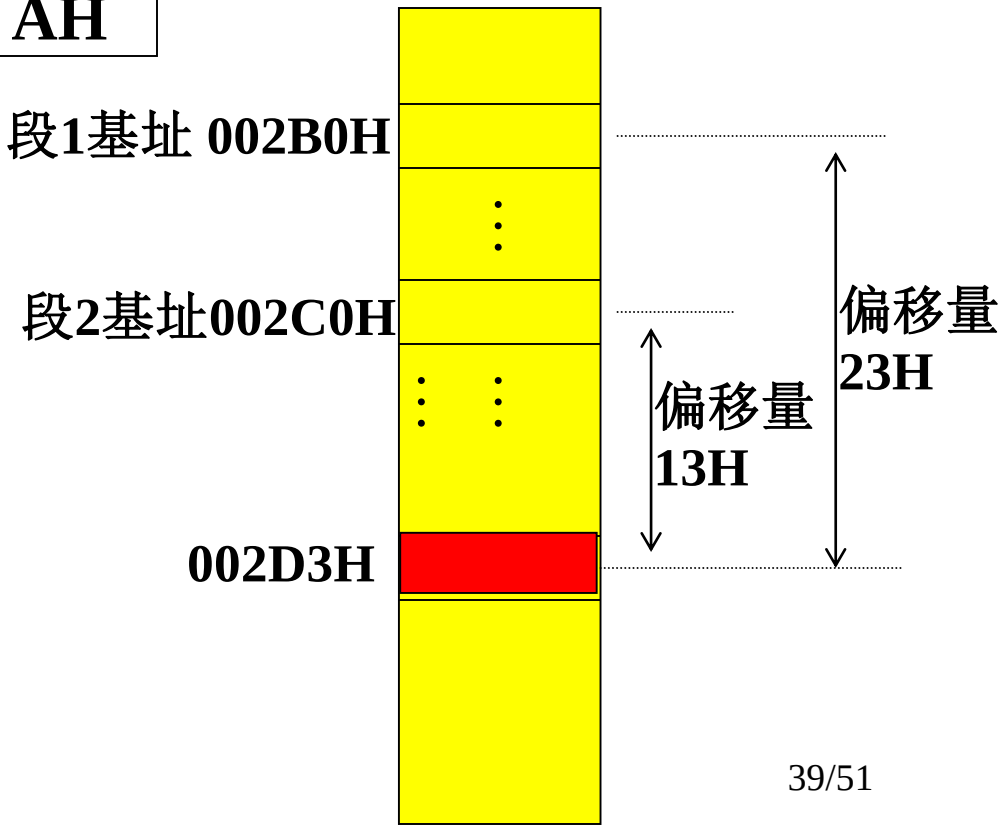
例1：



例2：同一个物理地址002D3H被两个逻辑段中的逻辑地址映射的情况。

002B0H+00023H=002D3H

002C0H+00013H=002D3H



4.逻辑地址的来源

在程序的执行过程中，CPU根据不同操作类型访问存储器，其逻辑地址中段基值和偏移量的来源是不一样的。下表是各种操作类型所对应的逻辑地址的来源。

序号	操作类型	逻辑地址		
		段基值		偏移量(OFFSET)
		隐含来源	允许替代来源	
1	取指令	CS	无	IP
2	堆栈操作	SS	无	SP
3	取源串	DS	CS，SS，ES	SI
4	存目的串	ES	无	DI
5	以BP作基址	SS	CS，DS，ES	有效地址EA
6	存取一般变量	DS	CS，SS，ES	有效地址EA

说明：

（1）允许替代来源也叫做段超越，它表示了段基值除使用隐含的段寄存器外是否可以指定其它段寄存器来提供。

（2）有效地址EA，它表示根据指令所采用的寻址方式（下一章介绍）计算出来的段内偏移量。

2.4 堆栈及其操作方法

堆栈是一个特定的存储区，访问该存储区一般需要按照专门的规则进行操作。

堆栈的用途：主要用于暂存数据以及在过程调用或处理中断时保存断点信息。

一、堆栈的构造

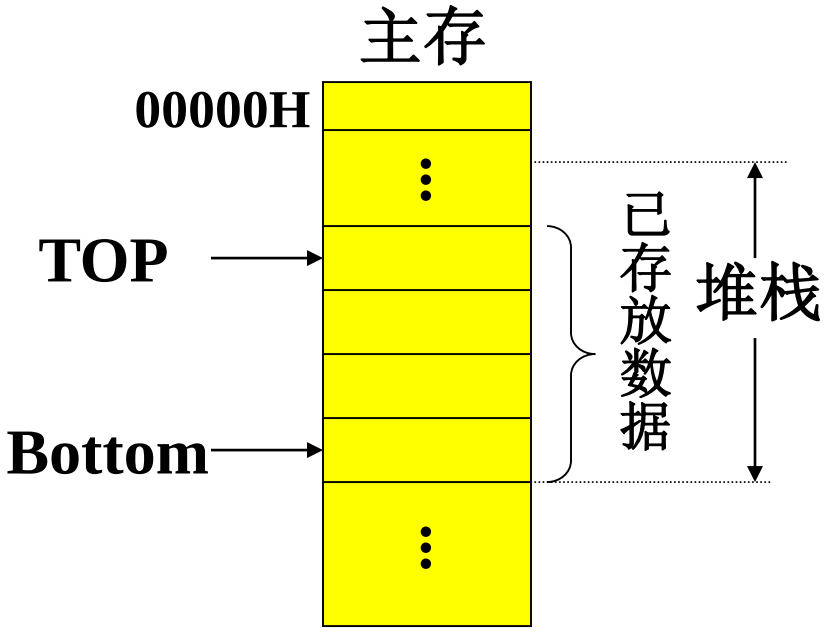
堆栈一般分为：专用堆栈存储器 和 软件堆栈

专用堆栈存储器 按堆栈的工作方式专门设计的存储器

软件堆栈 由程序设计人员用软件在内存中划出的一块存储区作为堆栈来使用。8086/8088采用这种方式。

堆栈的一端是固定的，称为栈底。栈底是堆栈存储区的最大地址单元。

另一端是浮动的，称为栈顶。在任何时刻，栈顶是最后存入信息的存储单元。栈顶是随着堆栈中存放信息的多少而改变。



为了指示现在堆栈中存放数据位置，通常设置一个寄存器来指示栈顶位置。其内容就象一个指针一样，因此被称为堆栈指针SP（Stack Pointer）。

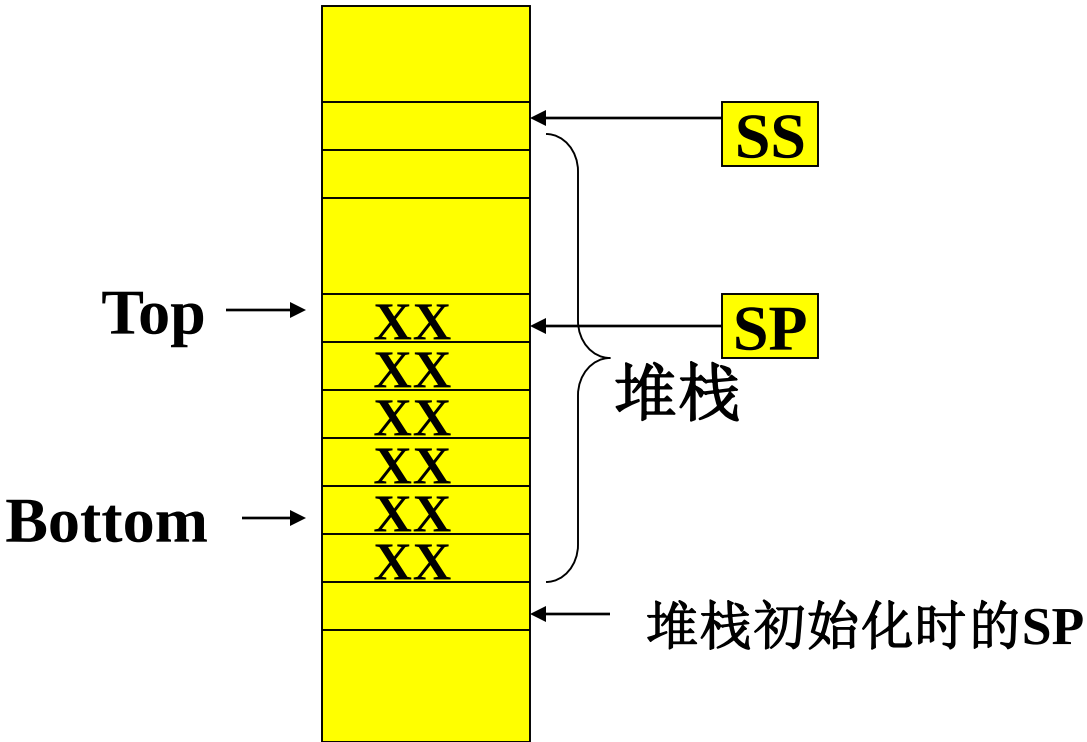
SP的内容始终指向栈顶单元

堆栈中数据进出都由SP来控制

在堆栈中存取数据的规则是：“先进后出FILO”（First-In Last-Out）。即最先送入堆栈的数据要到最后才能取出，而最后送入堆栈的数据，最先取出。

二、8086/8088堆栈的组织

在8086/8088微机中堆栈是由堆栈段寄存器SS指示的一段存储区。



顶由堆栈指针SP指示。SP中内容始终表示堆栈段基址与栈顶之间的距离（字节数）。当SP内容为最大(初始)值时，表示堆栈为空。而当（SP）=0时,表示堆栈全满.

当SP被初始化时，指向栈底+2单元，其值就是堆栈的长度。由于SP是16位寄存器，因此堆栈长度≤ 64K字节。

数据在堆栈中的存放格式是：以字为单位存放，数据的低8位放在较低地址单元，高8位放在较高地址单元。

当用户程序中要求的堆栈长度超过一个堆栈段的最大长度64KB时，可以设置几个堆栈段。

通过改变堆栈段寄存器SS的内容，即可改变到另一个堆栈段，当改变了堆栈段寄存器SS的内容后，必须紧接着赋予SP新值。

三、 堆栈操作

1.设置堆栈

设置堆栈主要是对堆栈段寄存器SS和堆栈指针SP赋值。

例如：

```
STACK1  SEGMEMT  PARA   STACK
        DB  100   DUP（0）
STACK1   ENDS
```

第一行中的PARA STACK就是用来说明本段为堆栈段。

当程序经过汇编、连接并装入内存时，系统将自动为其分配一个存储区作为堆栈段，将这个段的段基址的高16位送入SS中，将程序指定的字节单元数100赋值给SP。

2.进栈PUSH

进栈就是把数据存入堆栈。由指令PUSH或者由机器自动实现，可以将通用寄存器、段寄存器或字存储单元的内容压入堆栈顶部。

例：PUSH AX ;将寄存器AX的内容压入堆栈
PUSH DS ;将段寄存器DS的内容压入堆栈
PUSH DATA-WORD ;将字存储单元DADA-WORD压入
; 堆栈
PUSHF ;将标志寄存器内容压入堆栈。

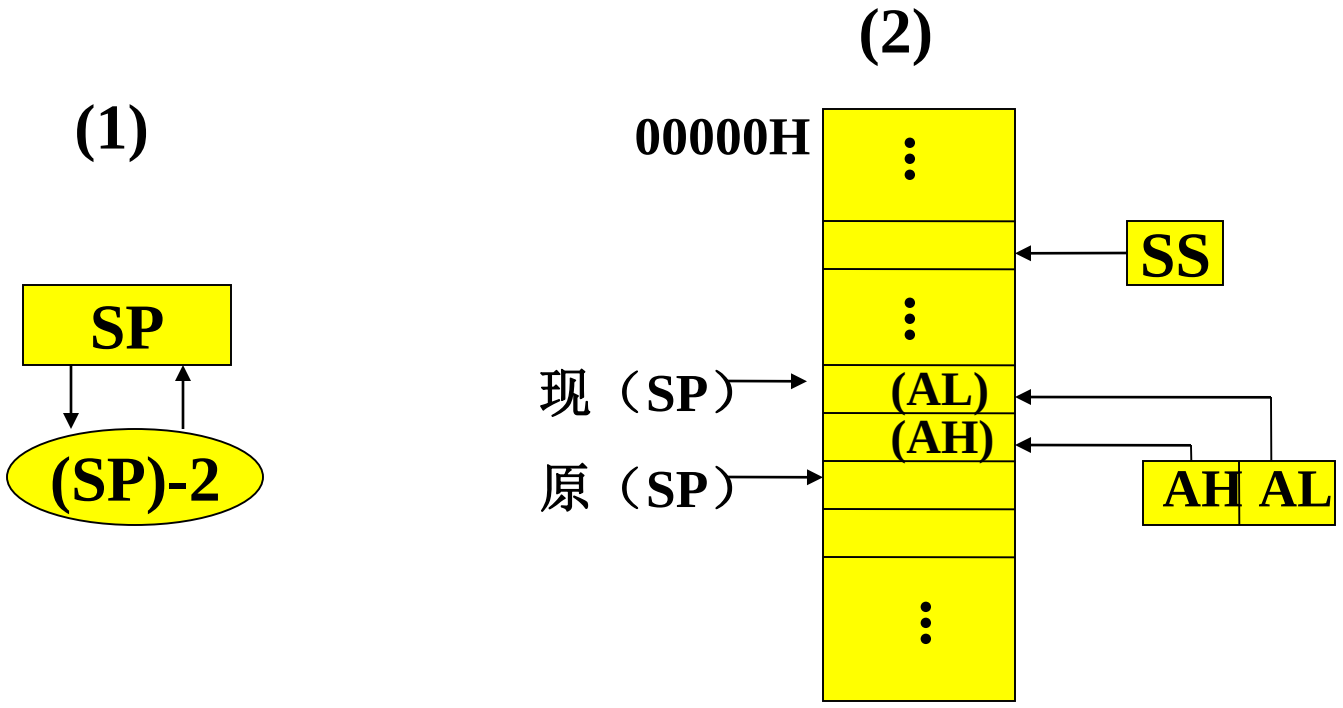
进栈的执行过程：

（1）首先将堆栈指针SP减2，即指向一个空的堆栈字单元

SP<=(SP)-2

（2）将要储存的内容（寄存器或存储单元的内容）送入SP指向的字单元中。（SP）<=数据

例如，指令PUSH AX的执行过程如下图所示：



3.出栈POP

出栈操作由POP指令或机器自动实现，它从堆栈顶部弹出一个字到通用寄存器、段寄存器或字存储单元。

例如：POP AX；将栈顶字单元内容弹出到AX
POP DS；将栈顶字单元内容弹出到DS
POP DATA-WORD；将栈顶字单元内容弹出到DATA-WORD存储。
POPF；将栈顶字单元内容送回标志寄存器F。

出栈的操作过程：

（1）将SP指向的字单元（即栈顶字单元）内容送往指定的寄存器或存储器。

即：寄存器/存储器<=((SP))

（2）堆栈指针SP内容加2，即：SP<=（SP）+2

例如，指令POP AX的操作过程如下图所示。