第二章　80X86微处理器

第一节8086/8088 结构

一. 微型计算机结构

二. 8086/8088 结构

第二节PC机的总线结构和时序

第三节32位微处理器

第一节 8086/ 8088结构

一、微型计算机结构

CPU+适当容量M+输入输出接口+设备

=MICRO_COMPUTER +软件

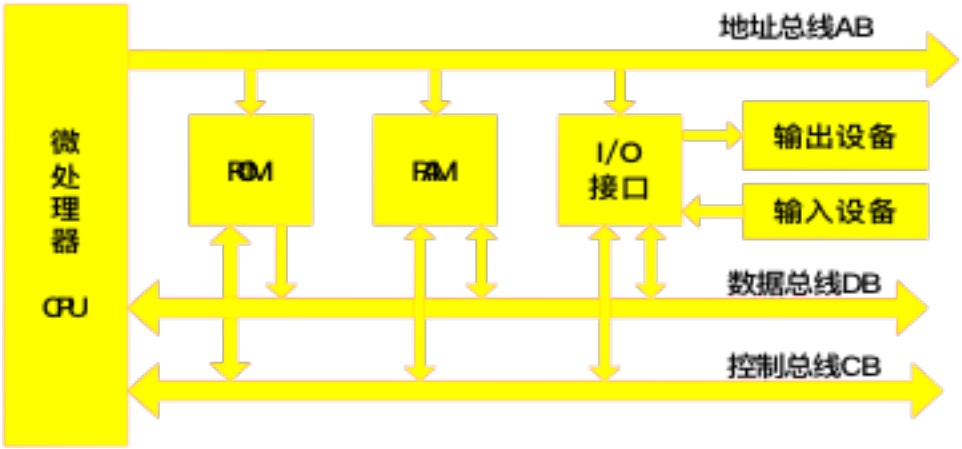
=MICRO_ COMPUTER SYSTEM

1、微型计算机外部结构

2、微处理器内部结构

1、微型计算机外部结构

微型计算机外部结构如下图所示(三总线结构)



微型计算机外部结构

2、 微处理器内部结构

由于CPU受成品率，成本，集成在单片上等原因限制

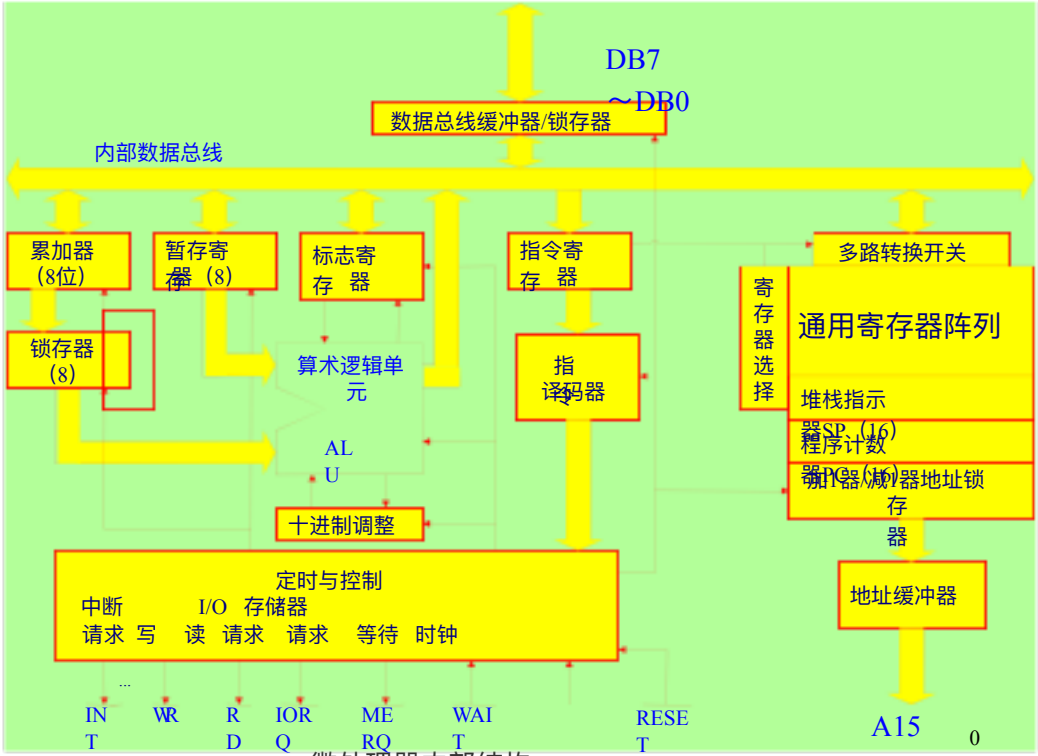
严格规定40条引脚，引脚数限制了总线的数量。

外部——采用三总线结构AB、DB、CB。

内部——采用单总线，

内部所有单元电路都挂在内部总线上，分时使用总线。

通常微处理器内部结构及外部连接方法如下页图所示。


DB7
～DB0
数据总线缓冲器/锁存器
内部数据总线
累加器
(8位)
暂存寄存器 (8)
标志寄存器
指令寄存器
多路转换开关
寄存器选择
通用寄存器阵列
堆栈指示器SP (16)
程序计数器PC (16)
加1器/减1器
地址锁存器
锁存器
(8)
算术逻辑单元
ALU
指令译码器
十进制调整
定时与控制
中断请求
写
读
I/O 请求
存储器请求
等待
时钟
地址缓冲器
...
INT
WR
RD
IORQ
MERQ
WAIT
RESET
A15
～A0
0

微处理器内部结构

微处理器内部主要由四部分组成：

（1） 内部寄存器阵列

- 用来寄存参与运算的数据（8位），

经常可以连成寄存器对（16位）用来存放操作数地址

- 16位专用寄存器

如： 程序计数器PC，堆栈指针SP

（2） 累加器和算术逻辑单元

对数据进行算术运算、逻辑运算场所，

运算结果 标志触发器记忆

或送某个寄存器等等。

(3) 指令寄存器、指令译码器、定时及各种控制信号产生电路

把用户程序中的指令一条条译出来，

然后以一定时序发出相应的控制信号。

(4) 内部采用单总线结构

在任何时候只有一个内部设备能使用内部总线，

各设备之间必须分时使用内部总线，因而速度受到影响。

∴只有要求速度高的位片机（通常用双级型集成电路）中， 采用内部多总线结构。

二、 8086/ 8088结构

INTEL 8086，16位机.

INTEL 8088 是一种准16位微处理器，

在INTEL8080与8085的基础上发展起来的。

结构特点：

(1) 内部结构 是16位的

（内部寄存器，内部运算部件，内部操作按 16位设计）；

(2)外部数据总线8条，能处理16位数据,也能处理8位数据；

（具有16位运算指令，包括*、/指令）

(3) 汇编语言与8080/8085兼容，能执行整套8080/8085的指令.

增加了许多16位操作指令；

(4) 20条地址总线，直接寻址能力1M字节；

(5) 40条引线封装；

(6) 单相时钟；

(7) 电源为5V。

8086与8088的主要区别在于8086的外部数据总线16位。

分三部分讨论

（一）8086/8088的编程结构

（二）8086/8088的寄存器结构

（三）存储器结构

（一）8086/8088编程结构

编程结构：从程序员和使用者的角度来看的结构。

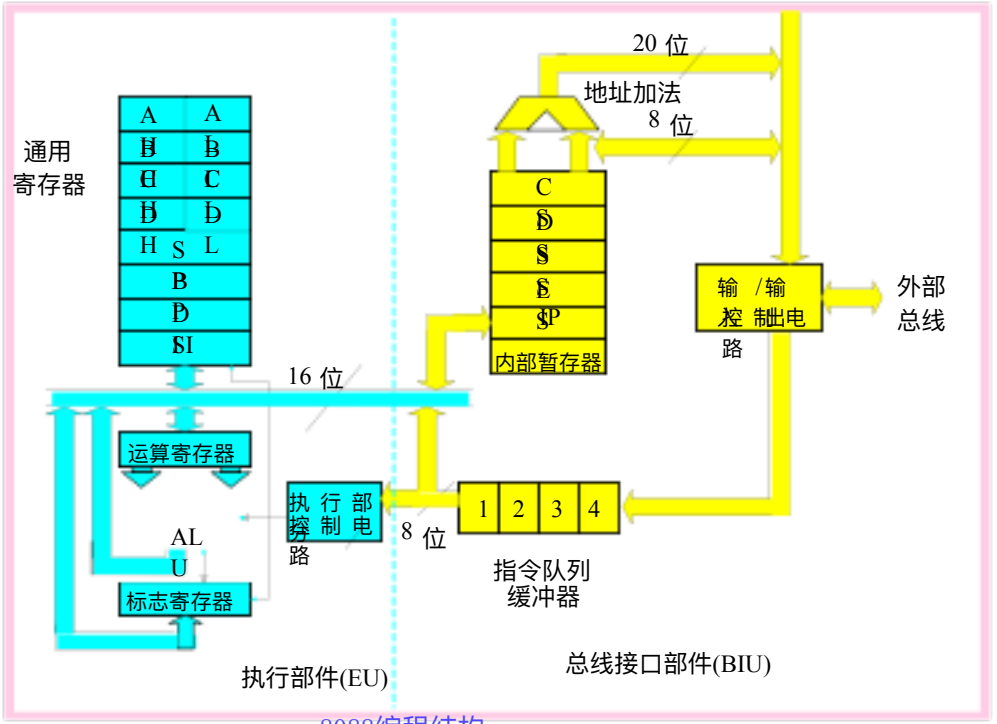
这种结构与CPU内部的物理结构和实际布局有区别。

8086/8088编程结构如下页图所示（P13G2.1）

分两部分：

1、总线接口部件 BIU（Bus Interface Unit）

2、执行部件 EU （Execution Unit）


20 位
地址加法
8 位
通用
寄存器
A
B
C
D
H
L
S
B
D
I
16 位
运算寄存器
AL
U
标志寄存器
执行部件(EU)
C
D
S
S
S
P
内部暂存器
输入/输出控制电路
外部
总线
执行部件控制电路
8 位
1
2
3
4
指令队列
缓冲器
总线接口部件(BIU)

8088编程结构

1、总线接口部件 BIU（Bus Interface Unit）

（1）．功能：负责与 M、I/O 端口传送数据。

具体讲：

总线接口部件要从内存 取指令送到指令队列；

CPU执行指令时，要配合执行部件从指定的内存单元
或者外设端口中取数据，将数据传送给执行部件；

或把执行部件的操作结果传送给指定的M或I/O口。

（2）．组成

- 4 个段地址寄存器（CS、DS、ES、SS），
- 16位指令指针寄存器IP,
- 20位的地址加法器(16D□段地址+偏移地址＝物理地址),
- 6字节（8086）或4字节（8088）的指令队列,
- 总线控制电路:

处理器与外界总线联系的转接电路。

包括三组总线：

20 位地址总线， 8 位双向数据总线，一组控制总线。

(3). 两点说明

- 指令队列

  8086 的指令队列为6个字节,

  8088 的指令队列为4个字节。

不论是8086还是8088都会在执行指令的同时从内存中取下一条或几条指令，取来的指令放在指令队列中，使 BIU 具有预取指令的功能，是一种先进先出（FIFO）的数据结构。

- 指令执行顺序

顺序指令执行：指令队列存放紧接在执行指令后面的那一条指令。

执行转移指令：BIU 清除指令队列中的内容，从新的地址取入指令，立即送往执行单元，然后再从新单元开始重新填满队列。

2、EU （Execution Unit）执行部件

（1）、功能：负责指令执行。

（2）、组成：

- 4个通用寄存器：AX、BX、CX、DX
- 4个专用寄存器：BP、SP、SI、DI，
- 标志寄存器（FLAGS）：

9个标志位，其中6个条件标志位用于存放结果状态，

- 算术逻辑单元：

16 位加法器，用于对寄存器和指令操作数进行算术或逻辑运算，

- EU 控制系统：

接受从总线接口单元的指令队列中取来的指令代码，

对其译码和向 EU 内各有关部分发出时序命令信号，

协调执行指令规定的操作。

8086/8088取指部分与执行部分是分开的。

- 在一条指令的执行过程中可以取出下一条（或多条）指令，指令 在指令队列中排队；
- 在一条指令执行完成后,就可以立即执行下一条指令，减少CPU为取指令而等待的时间，提高CPU的利用率和整个运行速度。

8086/8088微处理器:

BIU和EU分开，取指和执行可以重迭，

大大减少了等待取指所需的时间，提高CPU的利用率。

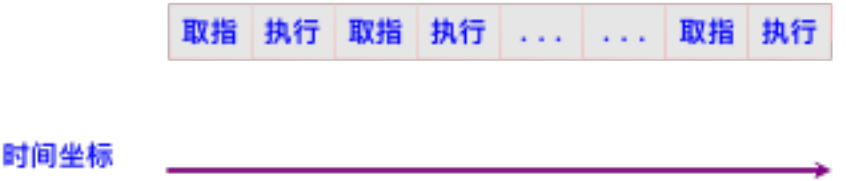
重迭操作技术：一方面提高了整个执行速率，

另一方面降低了与之相配的存储器的存取速度的要求。

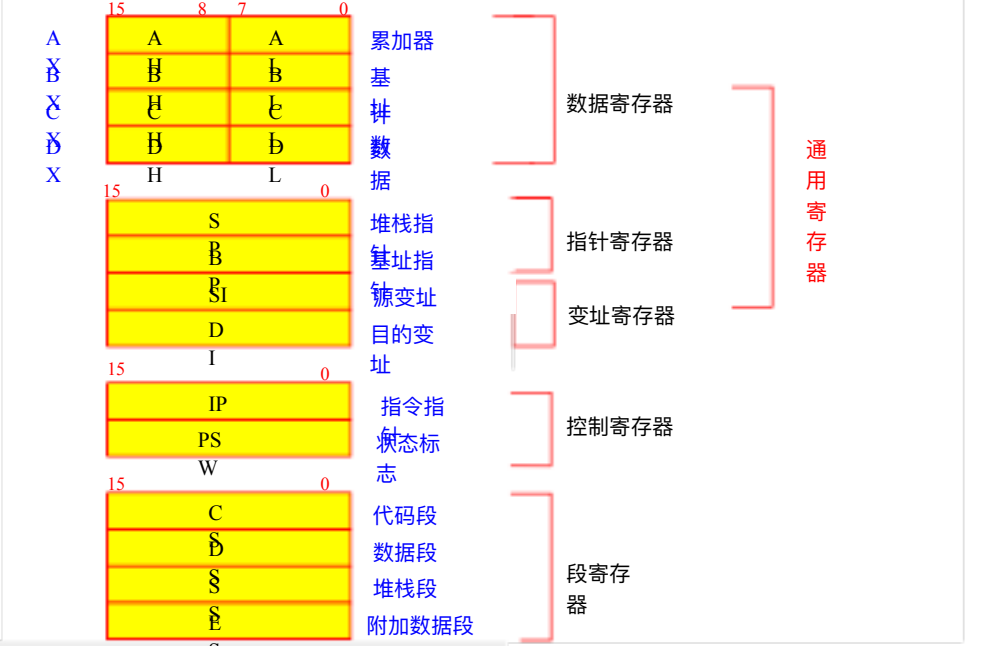
8086指令执行时序图

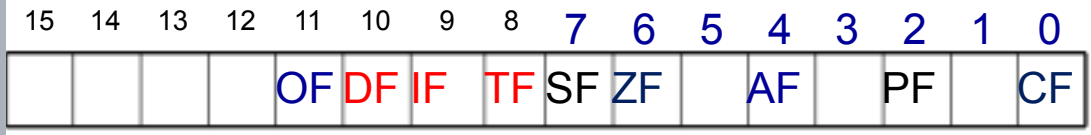
对于8080与8085及较早的8位微处理器:

程序执行由取指令和执行指令的循环来完成的，
每条指令执行完后CPU必须等待到下条指令取出来后才能执行。



（二）、 8086/8088的寄存器结构 (P15G2.3、P17G2.4)



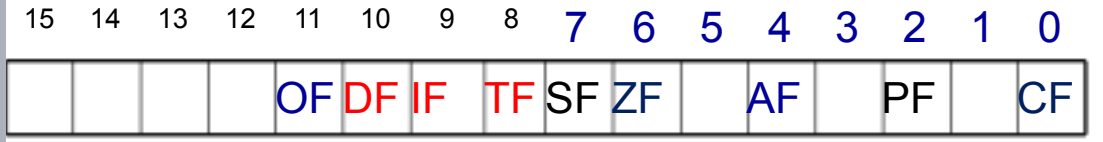



```
for( i=0;i<10;i++)
    s=s+i;
```

```
    MOV  CX, 10
A1: ……; 开始计算
    ……
    DEC  CX; CX减1
    JNZ   A1;不为0,执行A1
    ……
```

语句标号

指令中程序的分支，转向均是通过判断标志寄存器的标志位实现的。



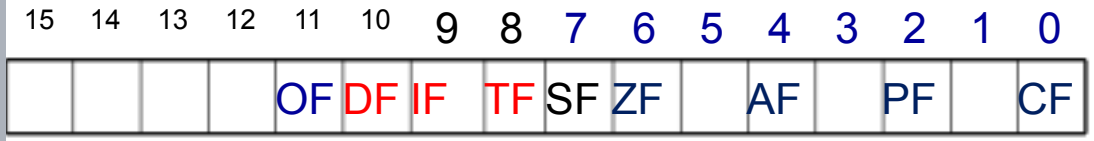
CF: 进位标志，运算器在加减法中出现进位或借位时该标志位置1。

PF: 奇偶标志，运算结果的低8位中1的个数为偶数时该标志位置1。

AF: 半进位标志，加法或减法时，位3向位4进位或借位时该标志位置1。用于BCD(二进制编码的十进制数)运算。

ZF: 零标志，运算结果为零时该标志位置1。

SF: 符号标志位，当运算结果的最高位为1(负数)时该标志位置1。



OF: 溢出标志，在算术运算中，带符号数的运算结果超出8位或16位带符号数所能表达的范围时，该标志位置1。

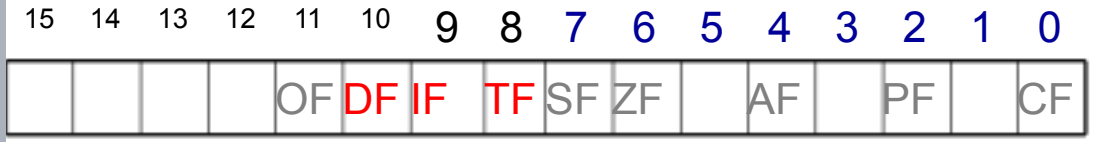
例如：8位数运算，当100 + 34 时，结果为134，超过8位正数的最大值 127，此时，该标志位置1。

100： 0110 0100

34： 0010 0010

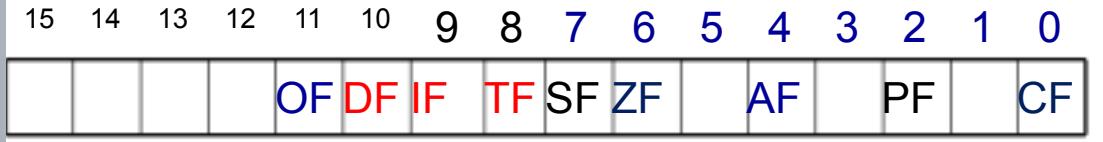
```
  0 1 1 0 0 1 0 0
+ 0 0 1 0 0 0 1 0
-----------------
  1 0 0 0 0 1 1 0
```

两个正数相加，结果为负数，这是由正数表示范围溢出造成的。



是控制位，不是反映目前CPU的状态，而是由程序进行设置，使CPU根据设置的状态运行。

其他标志位则是反映CPU的当前运行状态，程序可以通过判断这些标志进行转向、循环等。



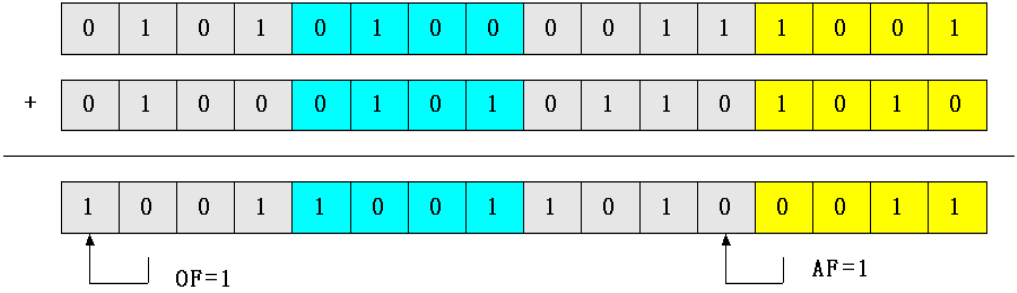
TF: 单步标志，该位置1时，8088/8086进入单步运行方式，即每条指令执行完后都产生中断，供用户检查各个寄存器及内存的当前状况，通常用于调试。

IF: 中断标志，该位置1时，8088/8086的CPU可以响应外部可屏蔽中断，否则，将外部可屏蔽中断屏蔽，即使有中断申请信号也不予理会。

DF: 方向标志，该位置1时，串操作指令为自动减量指令，即从高地址向低地址处理字符串。串操作指令是对将内存中连续的数据进行逐个自动操作。

运算对标志位的影响的例

例：执行两个数的加法，分析对标志位的影响。



SF=1； ZF=0； CF=0； OF=1； PF=1； AF=1

1、通用寄存器

数据寄存器、地址指针寄存器、变址寄存器。

数据寄存器: AX 、BX 、CX 、DX 。

地址指针寄存器: SP 、 BP 。

变址寄存器: SI 、 DI 。

2、段寄存器

CS 、 SS 、 DS 、 ES 。

3、控制寄存器

IP 、FLAGS。

1、通用寄存器

（1）数据寄存器

AX、BX、CX、DX 作为通用寄存器。

用来暂存计算过程中所用到的操作数，结果或其它信息。

访问形式: 可以用16位的访问;

或者可以用字节（8位）形式访问,

它们的高8位记作：AH 、 BH 、 CH 、 DH 。

它们的低8位记作：AL 、BL 、CL 、DL 。

AX——（Accumulator）累加器

它是算术运算的主要寄存器，

所有I/O指令都使用AL、AX与外部设备交换数据。

例：　　IN　AL，20H

　　OUT　30H，AX

BX——（Base）基址寄存器

在计算内存储器地址时，经常用来存放基址。

例：　　MOV　AX，[BX+03H]

CX——（Count）计数寄存器

在循环LOOP指令和串处理指令中用作隐含计数器。

例： MOV CX , 200H

AGAIN: ……

……

LOOP AGAIN ; (CX)-1□(CX),结果□0转AGAIN

DX——（Data）数据寄存器

一般在双字长乘除法运算时,

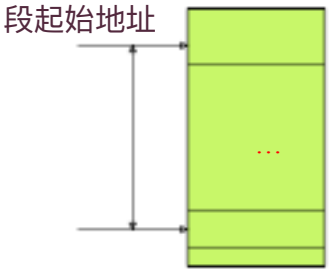
把DX和AX组合在一起存放一个双字长(32位)数，DX用来存放高16位;

对某些I/O操作DX可用来存放I/O的端口地址（口地址 □ 256）。

例： MUL BX ; (AX)□(BX)□(DX)(AX)

例: IN AL , DX

（2）地址指针与变址寄存器：



SP、BP、SI、DI 四个16位寄存器。

以字为单位在运算过程中存放操作数，

经常用以在段内寻址时提供偏移地址。

段内偏移地址

段地址 :只取段起始地址高16位值。

偏移地址:指在段内某内存单元物理地址相对段起始地址的偏移值。

地址指针寄存器(SP 、BP)

SP（stack pointer）——堆栈指针寄存器

用来指示栈顶的偏移地址, 必须与SS段寄存器联合使用确定实际地址。

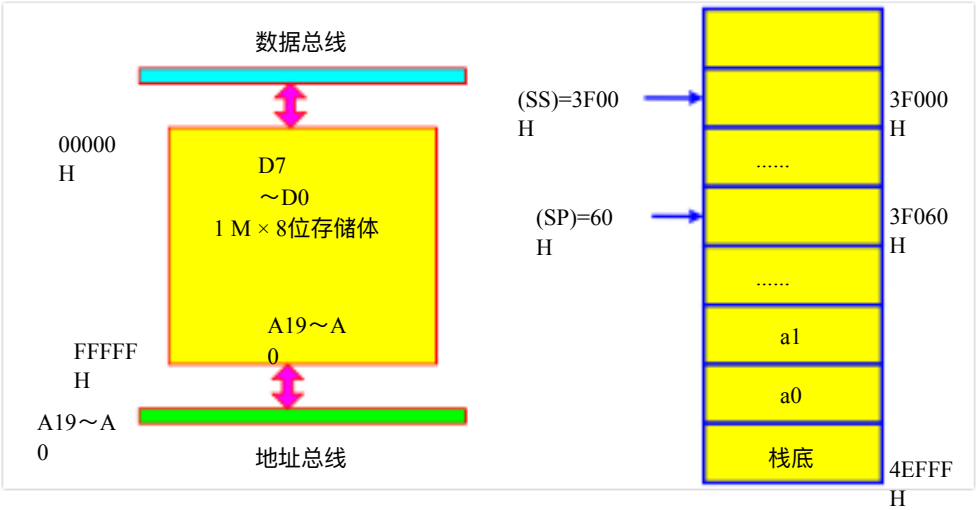
堆栈和指针如下页图所示。

BP（base pointer）——基址指针寄存器

可以与SS寄存器联合使用来确定堆栈段中某一存储器单元地址。

设：（SS)=3F00H,(SP)=0060H堆栈和指针如下图：

堆栈是内存开辟的一个特殊数据区，一端固定，一端浮动，严格按照后进先出的工作原则。



8086/8088系统存储器与总线连接

堆栈和指针

- 变址寄存器(SI 、 DI)

SI——Source Index Register 源变址寄存器。

DI——Destination Index Register 目的变址寄存器。

使用场合：常用于变址寻址。

一般与DS联用，用来确定数据段中某一存储单元的地址;

SI , DI具有自动增量和自动减量功能.

例： MOV AX, [SI]

在串处理指令中，SI、DI作为隐含的源变址和目的变址寄 存器分别达到在数据段和附加段中寻址的目的。

执行示意图如右图。

例：　……

MOV　SI，2000H

MOV　DI，3000H

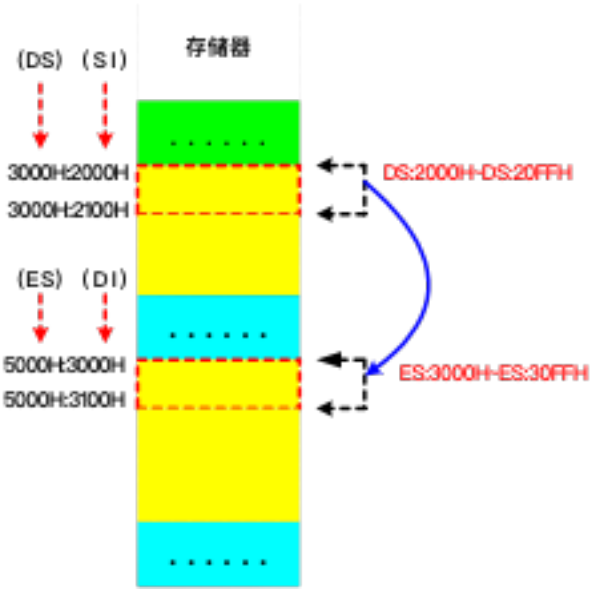
MOV　CX，100H

CLD

……

MOVSB

…….



串处理指令执行示意图

2、段寄存器

段寄存器: 4个16位段寄存器CS、DS、SS、ES。

用来识别当前可寻址的四个段，不可互换使用。

CS——Code Segment Register 代码段寄存器

用来识别当前代码段（程序一般放在代码段）。

DS——Data Segment Register数据段寄存器

用来识别当前数据段。

SS——Stack Segment Register堆栈段寄存器，

用来识别当前堆栈段。

ES——Extra Segment Register附加段寄存器，

用来识别当前附加段。

表 8086/8088段寄存器与提供段内偏移地址的寄存器之间的默认组合

段寄存器	提供段内偏移地址的寄存器
CS	IP
DS	BX、SI、DI或一个16位数
SS	SP或BP
ES	DI(用于字符串操作指令)

3、控制寄存器

控制寄存器：IP 、 FLAGS

IP—Instruction Pointer指令指针寄存器

用来存储代码段中的偏移地址;

程序运行过程中IP始终指向下一次要取出的指令偏移地址。 IP要与CS寄存器相配合才能形成真正的物理地址。

FLAGS — 标志寄存器，程序状态字寄存器（PSW，Processor States Word ）， 16位寄存器。

由条件码标志、控制标志构成。

只用了其中9位, 6位条件码标志 , 3位控制标志。如下表示。

●条件码标志:

用来记录程序中运行结果的状态信息作为后续条件转移指令的转移控制条件。∴称为条件码。

条件码包括6位：CF 、 PF 、 AF 、 ZF 、 SF 、 OF 。

① OF（Overflow Flag）溢出标志（一般指补码溢出）

OF=1：在运算过程中，如操作数超过了机器表示的范围称为溢出。

OF=0：在运算过程中，如操作数未超过机器能表示的范围称为 不溢出。

字节允许范围 -128—+127，

字运算范围 -32768—+32767 。

② SF（Sign Flag）符号标志

SF=1：记录运算结果的符号为负。

SF=0：记录运算结果的符号为正。

③ ZF（Zero Flag）零标志

ZF=1：运算结果为0。

ZF=0：运算结果不为0。

④ CF（Carry Flag）进位标志

CF=1：记录运算时从最高有效位产生进位值。

CF=0：记录运算时从最高有效位不产生进位值。

⑤ AF（Auxiliary Carry Flag）辅助进位标志

AF=1：记录运算时第3位（半个字节）产生进位值。

AF=0：记录运算时第3位（半个字节）不产生进位值。

⑥ PF(Parity Flag)奇偶标志

PF=1: 结果操作数低8位中有偶数个1。

PF=0: 结果操作数低8位中有奇数个1。

用来为机器中传送信息时可能产生的代码出现情况提供检验条件。

- 控制标志:

对控制标志位进行设置后,对其后的操作起控制作用。

控制标志位包括3位: TF、 IF 、 DF 。

跟踪（陷阱）标志TF、中断标志IF 、 方向标志 DF 。

① TF(Trap Flag)跟踪(陷阱)标志位

TF=1 ,每执行一条指令后，自动产生一次内部中断，

使CPU处于单步执行指令工作方式，便于进行程序调试，用户能检查程序。

TF=0, CPU正常工作，不产生陷阱。

1) IF(Interupt Flag)中断标志位

IF=1, 允许外部可屏蔽中断。CPU可以响应可屏蔽中断请求。

IF=0, 关闭中断。CPU禁止响应可屏蔽中断请求。

- IF的状态对不可屏蔽中断和内部软中断没有影响。

③ DF(Direction Flag)方向标志位

DF=1,每次串处理操作后使变址寄存器SI和DI减量，使串处理从 高地址向低地址方向处理。

DF=0,每次串处理操作后使变址寄存器SI和DI增量，使串处理从 低地址向高地址方向处理。

- DF方向标志位是在串处理指令中控制处理信息的方向用的。

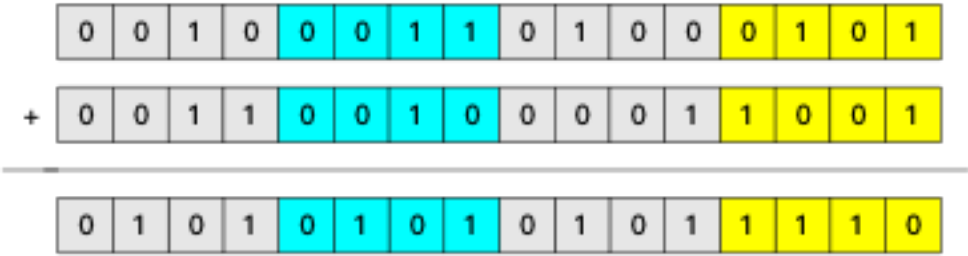
•控制信息:由系统程序或用户程序根据需要用指令来设置的。

• 状态信息:由中央处理器,根据计算结果自动设置的,

机器提供了设置状态信息指令,

必要时,程序员可以用这些指令来建立状态信息。

例1：执行两个数的加法，分析对标志位的影响。



标志:

运算结果最高位为0　　∴SF=0

运算结果本身≠0　　∴ZF=0

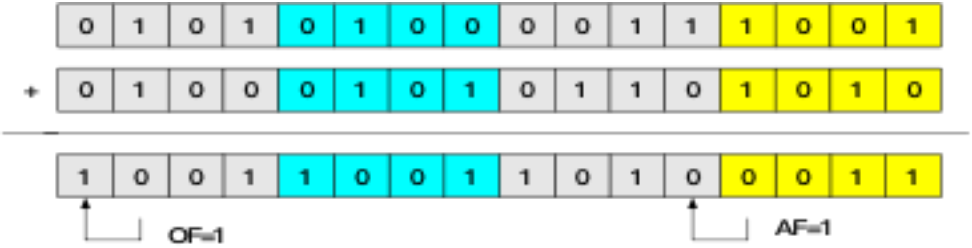
低8位中1的个数为奇数个　∴PF=0

最高位没有进位　　∴CF=0

第三位向第四位无进位　∴AF=0

次高位向最高位没有进位，最高位向前没有进位，∴OF=0。

例2：执行两个数的加法，分析对标志位的影响。



标志:

运算结果最高位为1， SF=1

运算结果本身不为0， ZF=0

最高位向前无进位， CF=0

次高位向最高位产生进位,而最高位向前没有进位， OF=1

结果低8位含偶数个1， PF=1

第三位向第四位有进位, AF=1 。

在绝大多数情况下，一次运算后并不影响所有标志，
程序也并不需要对所有的标志作全面的关注。
一般只是在某些操作后,对其中某个标志进行检测。

（三）存储器结构

1 存储单元的地址和内容

2 存储器地址分段

3 各段在存储器中分配

1 存储单元的地址和内容及与CPU之间数据传送

(1) 存储单元的地址和内容

存储器位编号：

8086/8088字长16位，由二个字节组成，位编号如下：

高位字节 （8～15位）　　低位字节（0～7位）

8086/8088内部的 ALU 能进行16 位运算。

有关地址寄存器如 SP、IP、BP、SI、DI 等都是16位的。

存储单元地址：按照字节编址

内存单元的地址和内容

存储单元的的内容：一个存储单元有效的信息。

机器字长是16位，

大部分数据以字节为单位表示，

- 一个字存入存储器占有相邻的二个单元：

低位字节存入低地址，高位字节存入高地址。

字单元的地址采用它的低地址来表示。

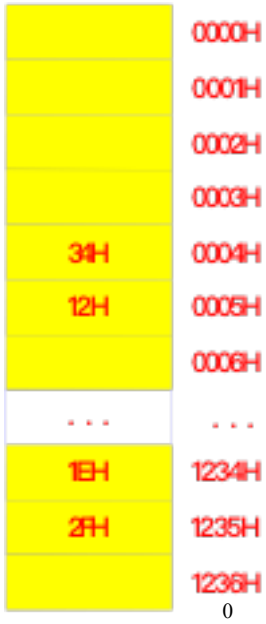
例： 字单元 ：（0004H）=1234H,

字节单元 ：（0004H）=34H

同一个地址既可以看作字节单元地址，

又可看作字单元地址，需要根据使用情况确定。

字单元地址：可以是偶数也可以是奇数。



(2) CPU和存储器或I/O端口之间传送数据

- 8086

8086数据总线是16位的。

讨论CPU和存储器或I/O端口之间传送数据方式：

- 字数据读/写操作
- 字节读/写操作

- 字数据读/写操作

(a) 操作数存放在偶地址开始两个存储单元或两个 I/O端口中：

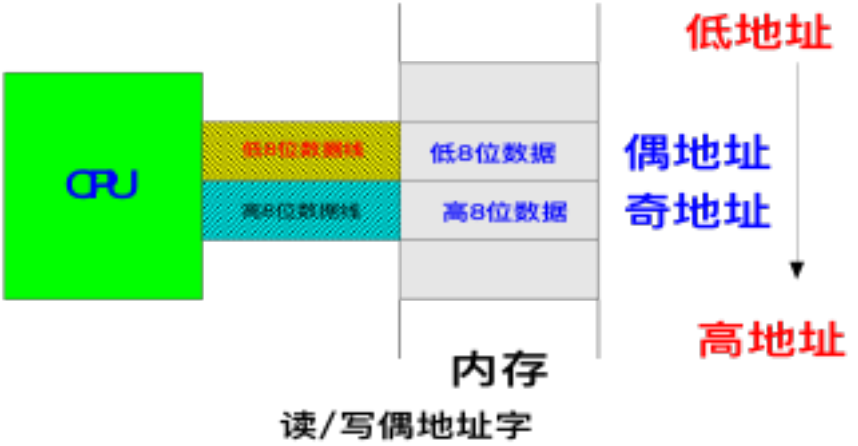
操作数字的低8位——在偶地址单元或在偶地址端口；

高8位——在奇地址单元或在奇地址端口。

在一个总线周期内完成（通常4个时钟周期）16位数据传送，

建议操作数存放在偶地址开始两个存储单元或两个 I/O端口中。

字数据读/写偶地址字操作如下图：



对应的偶地址单元/偶地址端口—数据通过数据总线低8位传输。

对应的奇地址单元/奇地址端口—数据通过数据总线高8位传输。

操作数存放在奇地址开始两个存储单元或两个 I/O端口中

如下图所示：

读/写奇地址字 0

在第一个总线周期中:

对应于奇地址单元或奇地址端口字节（操作字低8位）

通过数据总线高8位进行传输，而数据总线低8位处于空闲状态；

在第二个总线周期中:

对应于偶地址单元或偶地址端口字节（操作字高8位）

通过数据总线低8位进行传输，而数据总线高8位处于空闲状态.

由此可见，

操作数存放在奇地址开始两个存储单元/两个 I/O端口中进行

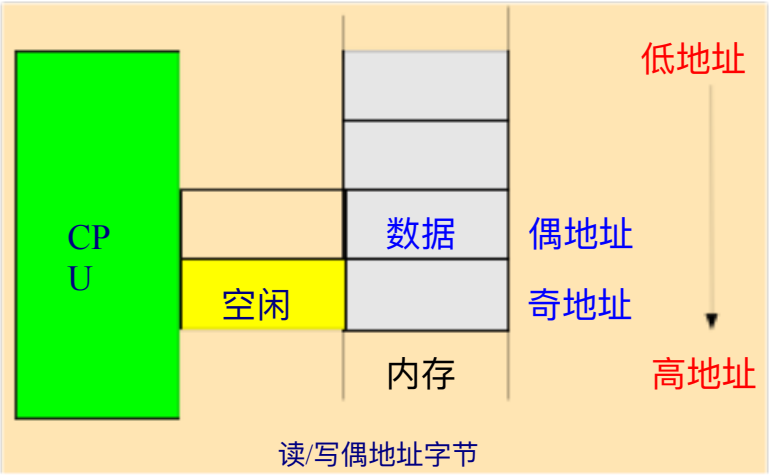
数据传输，8086需要二个总线周期。

- 字节读/写操作：

对偶地址单元/偶地址端口的字节数据进行读/写如下图所示：

在一个总线周期中，只有数据总线的低8位传输数据，

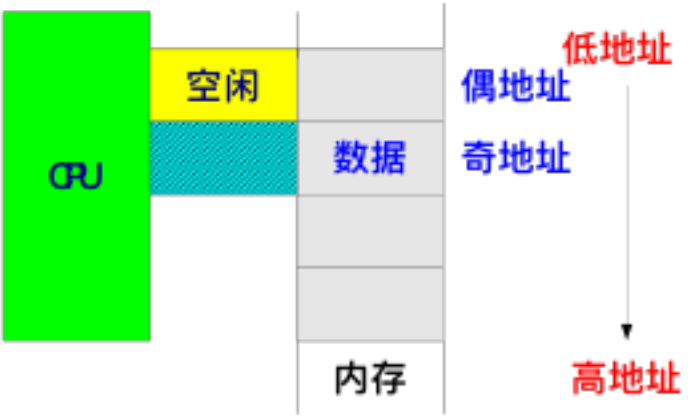
高8位处于空闲状态。



对奇地址单元/奇地址端口的字节数据进行读/写如下图所示：

在一个总线周期中，只有数据总线的高8位传输数据，

而低8位处于空闲状态。



- 8088

8088数据总线只有8位。

讨论CPU和存储器或I/O端口之间传送数据：

每个总线周期只传输1个字节，

所以对每个字操作要增加4个时钟周期。

（3）存储器特性

存储器的内容是取之不尽的。

即从某个单元取出其内容后，该单元仍保持原来的内容　不变，可以重复取出；

只有存入信息后，原有的内容自动消失。

2 存储器地址分段：

8086/8088有20条地址总线，直接寻址能力为220=1M字节。

用16进制数表示1M字节的地址范围应为00000H~FFFFFH。

（1） 8086/8088内部20位物理地址形成

（2）逻辑地址与物理地址

（1）8086/8088内部20位物理地址形成

- 存储器地址分段

8086/8088地址总线是20位的，CPU中的寄存器是16位的，
20位地址无法用16位寄存器表示，必须分段。

程序员在编制程序时把存储器划分成段。

段内地址16位,每个段的大小最大可达64KB；
实际可以根据需要来确定段大小，可以是1，100，1000，
在64K范围内的任意字节数。

IBM PC机对段的起始地址有限制，
即段不能从任意地址开始：
必须从任一小段（paragraph）的首地址开始。

小段的概念

从0地址开始每16字节为一小段，

对于16位地址总线，段内存储器小段地址如下：

如：0000H，0001H，0002H，……，000EH，000FH一个小段

0010H，0011H，0012H，……，001EH，001FH

0020H，0021H，0022H，……，002EH，002FH

…　　…

FFF0H, FFF1H, FFF2H, ……, FFFEH, FFFFH

其中：第一列就是每个小段的首地址。

每个小段首地址特征：

在16进制表示的地址中，最低位为0H

（即20位地址的低4位为0000B）。

在1M字节的地址空间，共有64K个小段，其首地址为：

0000 0H

0001 0H

…

4123 0H

4124 0H

…

FFFE 0H

FFFF 0H

□ 20位物理地址形成

物理地址：

在1M字节存储器里，每个存储单元都有一个唯一的20位地址作为该存储单元的物理地址。

CPU访问存储器时，必须先确定所要访问的存储单元的物理地址才能取出（或存入）该单元中的内容。

20位物理地址形成：由16位段地址和16位偏移地址组成。

段地址:只取段起始地址高16位值。

偏移地址:指在段内某内存单元物理地址相对段起始地址的偏移值。

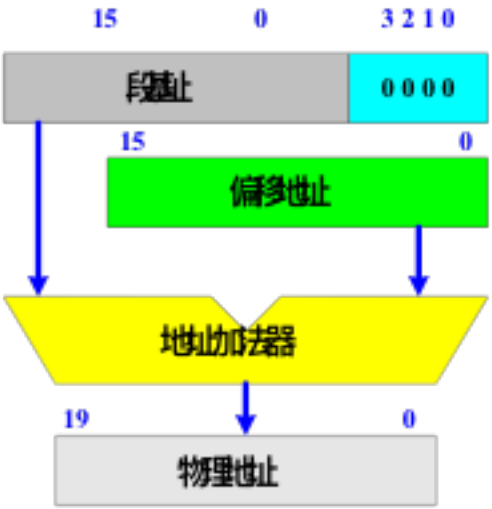
物理地址计算方法:

把段地址左移4位再加上偏移地址值形成物理地址，写成:
物理地址= 16d□段地址+偏移地址。

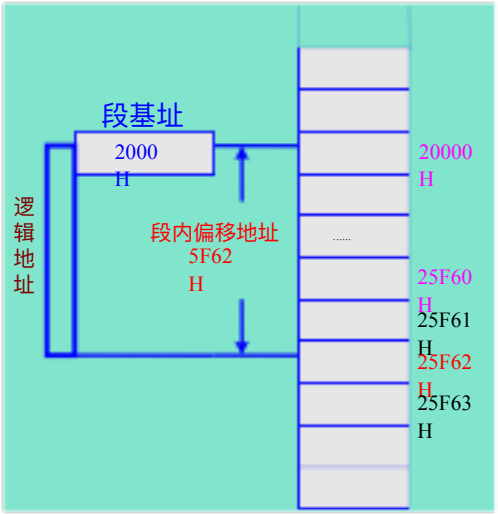
* 每个存储单元只有唯一的物理地址。
但可由不同的段地址和不同的偏移地址组成。

（2）逻辑地址与物理地址

- 逻辑地址与物理地址概念



物理地址的形成



逻辑地址与物理地址

逻辑地址：由段基址和段内偏移地址组成的地址，
段基址和段内偏移地址都是16位的无符号二进制数，在程序设计时使用。

物理地址：存储器的绝对地址（20位的实际地址），
范围从00000H～FFFFFH，
是由CPU访问存储器时由地址总线发出的地址。

存储器管理：将程序中逻辑地址转移为物理地址的机构。

□逻辑地址来源

操作类型	正常使用（隐含）段基址	可替换段地址	偏移地址	物理地址计算
取指令	CS	无	IP	(CS)□16D+(IP)
堆栈操作	SS	无	SP	(SS) □16D+(SP)
BP间址	SS	CS,DS,ES	有效地址EA	(SS) □16D+EA
存取变量	DS	CS,ES,SS	有效地址EA	(DS)□16D+EA
源字符串	DS	CS,ES,SS	SI	(DS) □16D+(SI)
目标字符串	ES	无	DI	(ES) □16D+(DI)

取指令：

自动选择代码段寄存器CS，

再加上由IP决定的16位偏移量，

计算得到要取的指令20位物理地址。

堆栈栈顶操作：

自动选择堆栈段寄存器SS，

再加上由SP决定的16位偏移量，

计算得到堆栈栈顶操作需要的20位物理地址。

涉及到操作数：

自动选择数据段寄存器DS或附加段寄存器ES，

再加上16位偏移量，计算得到操作数的20位物理地址。

其中：

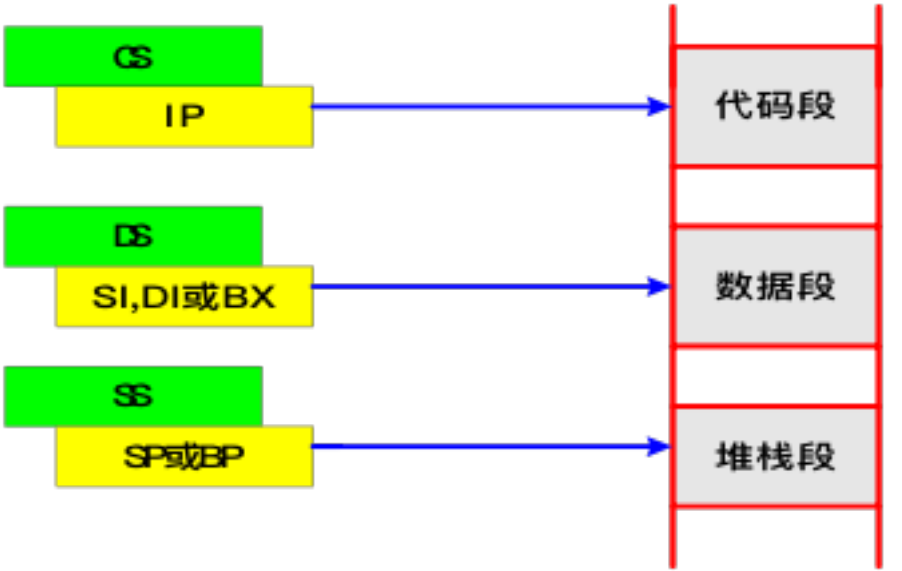
16位偏移量：　包含在　指令中的直接地址；

某个16位地址寄存器的值；

指令中的位移量+16位地址寄存器中值等。

16位偏移量关键取决于指令的寻址方式。

归纳段寄存器和其它寄存器组合指向存储单元示意图如下：



段寄存器和其他寄存器组合指向存储单元示意图

3 各段在存储器中分配

各段在存储器中分配分4种情况讨论：

（1）由操作系统负责分配

一般情况，各段在存储器中的分配是由操作系统负责。

每个段可以独立地占用64K存储区。

（2）各段也可以允许重迭

每个段的大小允许根据实际需要分配，不一定要占64KB。

每个存储单元的内容不允许发生冲突

（段可重迭，但使用时防止冲突）。

（3）在程序的首部设定各段寄存器的值

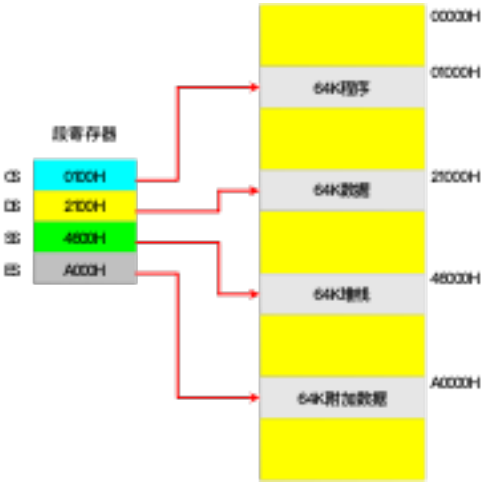
如果程序中的四个段都是64K的范围之内，

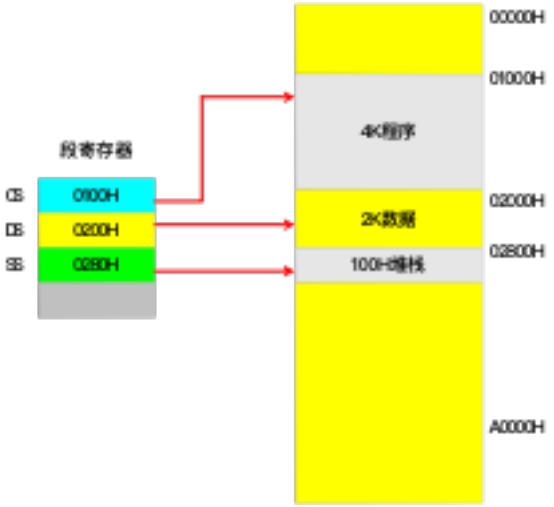
程序运行时所需要的信息都在本程序所定义的段区之内，

程序员只要在程序的首部设定各段寄存器的值就可以了。

（4）动态地修改段寄存器的内容

如果程序的某一段（如数据段）在程序运行过程中会超过64K空间，或者程序中可能访问除本身四个段以外的其它段区的信息，那么在程序中必须动态地修改段寄存器的内容。





64KB为一段

代码段范围:01000H~10FFFH

各段允许重迭:

代码段：01000H~01FFFH

数据段：02000H~027FFH

堆栈段：02800H~028FFH

物理地址: XXXX0H

偏移地址从0000H起

其中：XXXX为基地址

码段

堆栈段

基地址与偏移地址用两个寄存器分别存放，在同一逻辑段寻址时只对偏移地址进行变动即可。

数据段

附加数据段

0

例如：基地址：21ABH　偏移地址：1678H

物理地址：

$$\begin{array}{r} 21AB0 \\ +\quad 1678 \\ \hline 23128 \end{array}$$

23128 H

每一个逻辑段中，基地址是由计算机编译系统分配的，在这个逻辑段中不能改变，而偏移地址从0000 H开始，最长到FFFF H，所以一个逻辑段的长度不能超过64KB。

8086存储地址例

- 在内存中某数据区内，连续存放27个字数据。首字的存储地址为 BA00H：1BA0H。计算该数据区的首末存储单元的物理地址

- 8086的I/O组织

8086允许有65535个8位的I/O端口。

- 8086/8088 采用独立编址方式访问I/O端口
- 8086/8088 CPU使用(低)16位地址线寻址I/O端口
- 最多可达64K个端口地址
- 可访问64K个8位(字节数据)的端口
- 若访问16位(字数据)的端口时，最多可达32K个

注意：

以8086/8088CPU的IBMPC系统中，存储器首尾地址的用途固定。

- 00000H~003FFH共1K内存单元用于存放中断向量。

- FFFF0H~FFFFFH是存储器底部的16个单元。

  系统加电复位时，会自动转到FFFF0H单元执行，

  而在FFFF0H处存放一条无条件转移指令，转向系统初始化程序。

小结与问题思考：

微型计算机外部、内部结构 特点：

外部——采用三总线结构AB、DB、CB。

三组总线：

地址总线 : AB(ADDRESS BUS):

单向，由CPU发出的，

地址总线 AB的位数决定指令能直接寻址内存单元范围和外设接口。

序号从0开始。如：A19~A0

数据总线 DB(DATA BUS):

双向，实现CPU 与 M、或I/O 传送数据。

序号从0开始。如：D15~D0

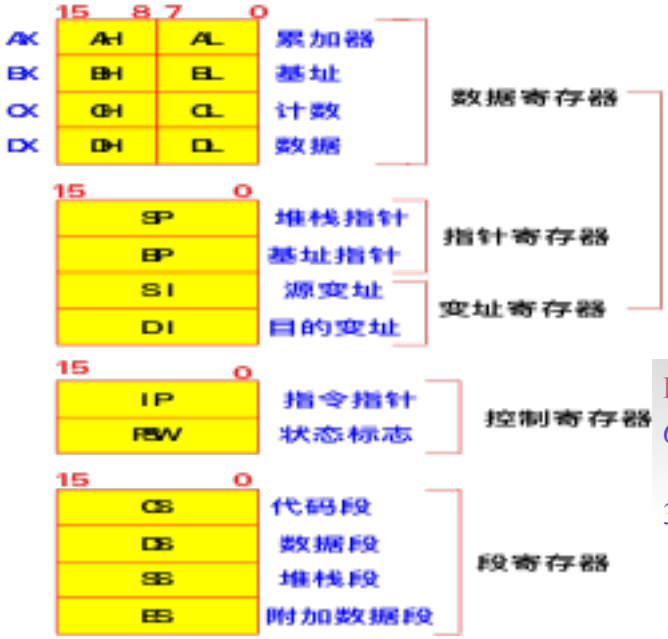
控制总线 CB(CONTROL BUS):传送控制信号。

内部——采用单总线，

即内部所有单元电路都挂在内部总线上，分时使用总线。

2. 8086/8088内部有14个16位寄存器,按功能分为三类：

通用寄存器
段寄存器
控制寄存器



数据寄存器:AX,BX,CX,DX
地址指针寄存器:SP,BP
变址寄存器:SI,DI

FLAGS中：
6 位条件标志：OF,SF,ZF,CF
PF,AF
3位控制标志：TF,IF,DF

3. 学习寄存器注意事项（问题思考）：

（1） AX , BX ,CX ,DX是数据寄存器

可以作为16位寄存器使用，又可以作为8位寄存器使用。
但每个寄存器在使用时各有特殊之处。

问题： AX,BX,CX,DX能用来存放偏移地址吗？哪些可以？哪些不可以?

（2）16位的寄存器

问题： BX,BP,SP,SI,DI,IP存放的偏移地址在构成物理地址时，与段地址是如何配合使用?

（3）FLAGS中

问题：条件码标志、控制标志各位的含义是什么?

4. 逻辑地址与物理地址是如何定义的？差别在那里？

5. 存储器是怎样编址的？同一个地址既可以看作字节单元地址，又可看作字单元地址，怎样理解？

1. IBM PC机存储器分段有哪些规定？
2. 系统加电复位时，会自动转到那个单元执行？
3. 重迭操作技术有什么好处？
4. 如何理解每个存储单元只有唯一的物理地址。但可由不同的段地址和不同的偏移地址组成？

8086/8088存储器结构

8086的1MB存储空间实际上分为两个512 KB的存储体，又称存储库，分别叫高位库和低位库，如图所示。

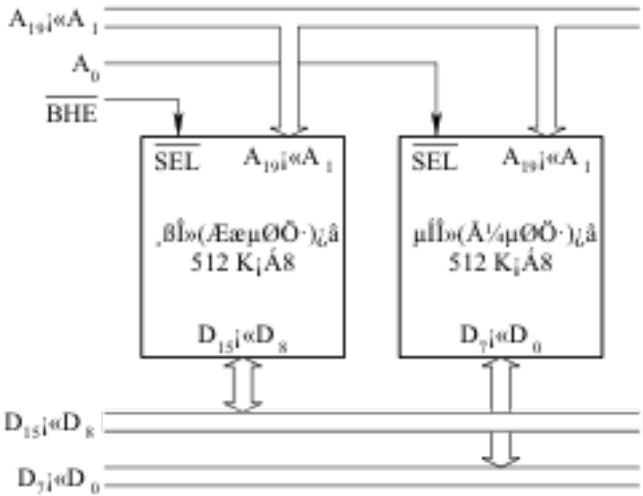


图　8086存储器高低位库的连接

低位库与数据总线D7□D0相连，该库中每个存储单元的地址为偶数地址；高位库与数据总线D15□D8相连，该库中每个存储单元的地址为奇数地址。

地址总线A19□A1可同时对高、低位库的存储单元寻址，A0和$\overline{BHE}$用于对库的选择，分别连接到库选择端SEL卜。如表所示。

表 **8086**存储器高低位库选择

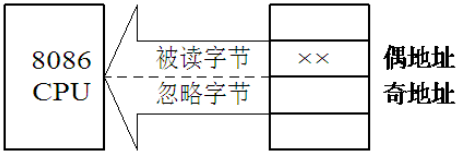
$\overline{BHE}$	A_0	对 应 操 作
0	**0**	同时访问两个存储体，读/写一个字的信息
0	**1**	只访问奇地址存储体，读/写高字节的信息
1	**0**	只访问偶地址存储体，读/写低字节的信息
1	**1**	无操作

规则字: 从偶地址开始的两个连续存储单元，即字的低字节在偶地址单元，高字节在奇地址单元。

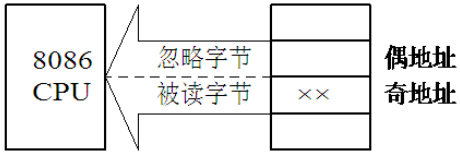
非规则字: 从奇地址开始的两个连续存储单元，即字的低字节在奇地址单元，高字节在偶地址单元。

对于规则字可通过一个总线周期完成读/写操作，这时A0=0，BHE=0；对于非规则字需要通过两个总线周期才能完成读/写操作，即第一次访问存储器时读/写奇地址单元中的字节，第二次访问存储器时读/写偶地址单元中的字节。

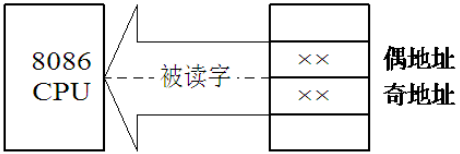
为了加快程序的运行速度，希望字型数据在存储器中规则存放。



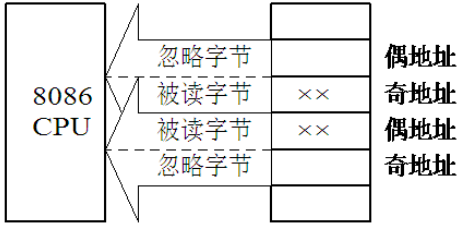
(a)读偶地址单元中的字节



(b) 读奇地址单元中的字节



(c) 读偶地址单元中的字



(d) 读奇地址单元中的字