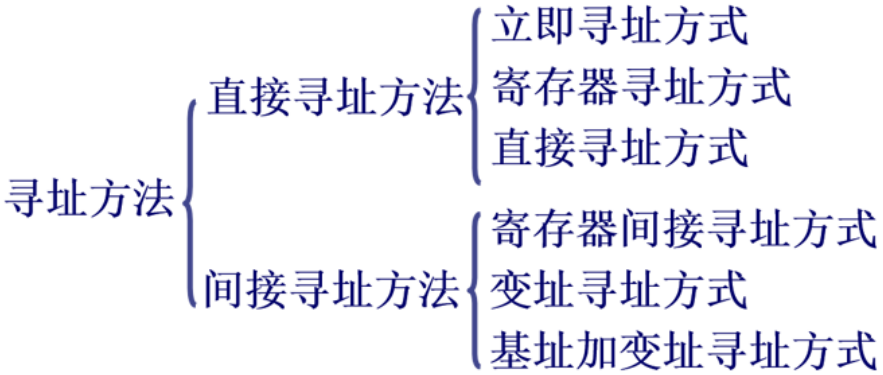
2.1 简述

本章的学习重点：

（1）6种寻址方式的使用格式及语法规定；
（2）6种寻址方式地址表示的含义及应用；



> 还有一种分法，可以分成3种
>
> 1. 寄存器方式
> 2. 立即方式（立即数）
> 3. 储存器方式
>   - 寄存器间接寻址
>   - 变址寻址
>   - 基址加变址寻址
>   - 直接寻址
>
>   > 为什么叫储存器方式

> 因为是存了操作数地址，再通过地址取找操作数
>
> 像C语言里的指针
>
> 注意，存储器方式里的这四种，都是无类型的，因为相当于就只告诉你一个指针，你要取一格，两格，还是四格，要看另一个操作数或者PTR指定。

学习寻址方式的3点思考：

（1）语法问题：寻址方式的使用格式及语法规定；
（2）类型问题：寻址方式表示出的数据类型；
（3）段的确定问题：寻址方式指明操作对象在主存中时，它在主存中的什么段中?

2.2 立即寻址方式

使用格式：n（n为常数或数值表达式，称为**立即数**）
功能：n本身就是操作对象，它作为指令的一部分，与指令一起在主存的代码段中。

```
例:
MOV    BX，10
MOV    AH，'A'
ADD    EAX，-12345678H
```

说明：
（1）**立即数**只能作为指令的**源操作数**，不能作为目的操作数；
（2）立即数不能作为单操作数指令的操作数；
（3）立即数本身只有大小，**没有类型**，所以另一边需要隐含类型或者指明类型。
如: MOV 100H, AX ；ERROR
INC 50 ；ERROR
（4）立即寻址方式**主要用于给寄存器或存储单元赋值。**

2.3 寄存器寻址方式

使用格式：R ；R为CPU中寄存器名。
功能：操作对象在CPU的寄存器R中。（不是在主存中）

```
例:
    MOV   BX，AX          ；(AX) → BX
    ADD   EDX，EAX        ；(EDX)+(EAX)→ EDX
```

说明：
（1）操作数的**类型**由**寄存器的位数决定**

> 8位寄存器是字节类型；16位寄存器是字类型；32位寄存器是双字类型.

（2）在双操作数的指令中，当两个操作数**类型均明确时，必须一致。**
如:

```
MOV   BL，AX    ；ERROR    BL是字节类型，AX是字类型，类型不一致
MOV   AX，EAX   ；ERROR    AX是字类型，EAX是双字类型，类型不一致
```

2.4 直接寻址方式

使用格式：**段寄存器名：[EA]**　或　**变量（例如BUF值之类的）** 或　**变量+常量（BUF+2这种）**

> (“段寄存器:”称为段跨越前缀)

其中：偏移地址EA的形式为**常数**或者**数值表达式。**
功能：操作对象在主存指定的段中。EA值与包含它的指令码一起存放在内存的代码段中。

```
例:
MOV  DS:[2000H]，AX;
执行前，设(DS)＝3000H.
在实方式下执行:
计算PA=(DS)+2000H=30000H+2000H=32000H.

(AX)→PA，即(32000H)=(AX);
```

> 就是说32000H 只是相当于一个地址，把AX里的内容，(AX)放到（32000H）中。

说明:

1. 段属性问题:
   “段寄存器名:[EA]”格式中，操作对象所在**段寄存器名指定的段中。**
   “含有变量的地址表达式”格式中，操作对象在**变量所定义的段中。**

   > 比如BUF在哪个段定义，就是哪个段的。

2. 类型属性问题
   “段寄存器名:[EA]”格式表示的操作数无类型，若**漏写段跨越前缀**，则汇编程序会错误的认为其是立即寻址。

   > 直接写 [20H]就相当于以（20H）作为偏移地址。注意与寄存器间接寻址区分。
   >
   > > […] 表示以…的内容做偏移地址，如[AX] 即以（AX）做偏移地址[1]
   >
   > MOV AX, ES:[20H]　；源操作对象在内存ES段中，无类型。
   > LEA　AX, [20H]　　;等价MOV AX, 20H, 即源操作数为立即寻址

   “含有变量的地址表达式”格式中，表示的操作数有类型，且类型由变量的类型决定。

   > MOV BUF, AX　　;
   >
   > 目的操作数为直接寻址（是个变量），且类型和段属性由BUF决定。（其中BUF为已定义的字类型的变量）

2.5 寄存器间接寻址方式

使用格式：[R]

- R为16位寄存器**BX,BP,SI,DI**之一。注意16位的只能是这四个

  > BX，基址寄存器
  > BP，堆栈基址寄存器
  > SI， 源变址寄存器
  > DI， 目的变址寄存器

- 或是32位寄存器:EAX,EBX,ECX,EDX　,EDI,ESI,EBP,ESP之一
- 不能是8位寄存器。

功能：操作对象在主存中，操作对象的EA在CPU的R中。

例：`MOV AX，[SI]`；SI为规定的寄存器

- 假定执行前：(AX)=5, (SI)=20H, (DS:[20H])=0FFFFH.
  执行:

  (1)CPU计算DS: EA=(SI)=0020H
  (2)CPU依"DS:EA", 按工作方式计算源操作数的PA， PA=(DS)+EA, 再根据PA值取出操作对象0FFFFH → AX
- 执行后: (AX)=0FFFFH; (SI), (DS), DS:(20H)内容未变

说明：

（1）段属性
默认情况：当R是**BP,EBP,SP,ESP,** 则操作对象在当前**堆栈**中，即操作数地址为"**SS:[R]**"；
R为其它规定的寄存器，则操作对象在当前**数据段**中，即操作数为"**DS:[R]**"。
指定情况：在[R]前可加上段跨越前缀，则操作对象在前缀指定的段中。

例： MOV DS:[EBP], CX; 目的操作数表示操作对象在DS段中，而不是在默认的堆栈段中。
（2）类型属性：此种寻址**无类型**，所以要用PTR BYTE 这种来指定类型
（3）注意[R]中的R必须是规定的寄存器。

```
如: MOV  [AX], CX   ;  ERROR , 16位寄存器里面不能用AX
    MOV   DX, [BL]  ;  ERROR ,  不能用8位寄存器
```

```
例：执行下列程序段后，EAX,EBX的内容为多少？

MOV  EAX, -1      ; -1 → EAX, 即(EAX)=0FFFFFFFFH
MOV  [ESP], EAX   ; (EAX) →([ESP]),  即把(EAX)送入SS 的栈顶但ESP指针未移动。
                  ; 相当于ESP指针指向的内容变成(EAX)了。
                  ; 因为它不是PUSH进去的，它只是简单替换了栈顶的值。所以ESP指针不动。
POP  EBX          ; (ESP) →EBX,即(EBX)=0FFFFFFFFH, (ESP)+4 →ESP
```

发现了么，[ ]这个功能很像指针

2.6 变址寻址方式

使用格式：`[R*F+V]（或写成：[R*F]+V，或V[R*F]）`

> 像一维数组方式，注意这个V写在前面也是表示加V，不是乘

功能：操作对象在主存中,其相对段首址的偏移地址　EA=(R)*F+V;

其中：

- R的规定同寄存器间接寻址中R的规定

> R为16位寄存器时，只能为**BX,BP,SI,DI**之一

- F为指定的比例因子，**可为1,2,4或8。**
- 当R为16位寄存器或ESP时，F只能取1并省略不写。 2

- V为**16位(R为16位寄存器)**或32位(R为32位寄存器)的二进制补码表示的有符号数；或者是数值表达式、变量或标号名。

例:

> MOV AL，[BX+5] ⇔ MOV AL, 5[BX] ；其中F为1，默认的段寄存器为DS.
>
> MOV AL, [EBX*2]+5 ；其中F为2，V为5，默认的段寄存器为DS.
>
> 执行前：(AL)=18H, (EBX)=1100H, (DS:[2205H])=55H.
> 执行: EA=(EBX)*2+5=1100H*2+5=2205H
> (DS:(2205))=55H → AL
>
> 例: ADD -2[BP], AX ; (AX)->SS:[(BP)-2]，默认的段寄存器为SS[3]

说明:
(1)段属性问题

当V为常数或数值表达式时，操作对象所在的段**由R（寄存器）决定**（与寄存器间接寻址方式相同）
当V为**变量或标号**时，操作对象所在的段是**变量或标号所在的段**（不由R决定）。
感觉像

(2)类型

当**V为常量时，此种寻址无类型**;
当**V为含有变量或者标号的表达式时，则有类型，其类型与变量或标号类型相同。**

如： MOV BUF[BX], CX ；设BUF是已定义的字类型变量， 则BUF[BX]类型为字类型。CX当然也是字类型。
假定BUF定义为字节类型，则BUF[BX]与CX类型不一致，此指令错误。

(3)注意R为规定的寄存器

MOV AX, -12[CX] ；ERROR，CX不是指定的4个16位寄存器之一。

(4) 注意R为ESP时F只能为1

2.7 基址加变址寻址方式

使用格式：`[BR+IR*F+V]`(或写成：`V[BR][IR*F]`, 或 `V[BR+IR*F]`)
功能：操作对象在内存中,其EA=(BR)+(IR)*F+V。此种寻址格式与变址寻址的区别仅仅多了一项**BR寄存器**，称为**基址器。**

> B代表Base基址寄存器，I代表index，变址寄存器

其中:

(1)V,F同变址寻址的规定。F总是与IR相乘。
(2)BR与IR选用与搭配关系：
当使用16位寄存器时：
BR只能选用**BX和BP之一**;
IR只能选用**SI,DI之一**与之搭配;
F只能为1。
当使用32位寄存器时：
BR可选用任一32位通用寄存器；(8个)
IR可选用**除ESP之外**的任一32位通用寄存器与之搭配。（7个）

例：MOV AX, 8[BX][SI]; 其中V=8,BR为BX,IR为SI,F为1
执行前:(AX)=45H,(BX)=30H,(SI)=20H,DS:[0058H]=99H
执行：EA=(BX)+(SI)+V=30H+20H+8H=0058H
(DS:[0058H])=99H → AX

说明：
(1)段属性与类型属性问题

- 段的默认情况：当**V为常数时**， 默认的段寄存器由基址器**BR决定。**
即BR为BP,ESP,EBP默认段为SS,其它默认段为DS。操作数无类型。
- 当**V为变量或标号表达式时**，操作对象所在的段就是变量或标号所在的段。
操作数有类型，且与变量或标号类型相同。

段的显式表示：

操作对象在显式给出的段中。
例：

- MOV AX, [BP+SI] ;源操作数无类型；由于BP，操作对象在SS中
- MOV AX, DS:[BP+SI] ;源操作数无类型；由于段超越前缀为DS，操作对象在DS中
- MOV AX, SUM[BP+SI] ;源操作数有类型，类型同SUM；操作对象在SUM所在的段中，SUM在DS段就用DS
- MOV AX, CS:SUM[BP+SI] ;源操作数有类型；由于段超越前缀为CS，操作对象在CS段中
- MOV EDX, ES:10[EBX+ESI*4] ;源操作数无类型；由于段超越前缀为ES，操作对象在ES中

注意:
(1)在决定段时，优先级关系：段跨越前缀>变量>BR
(2)注意BR与IR的搭配问题
例：MOV [BX+BP], AX ; **ERROR**

> 从AX看出是16位寄存器
>
> BR只能选用**BX和BP之一**;
> IR只能选用**SI,DI之一**与之搭配;

MOV [EBX+ESP*2+5], EAX ; **ERROR**

> 从EBX看出是32位寄存器
>
> IR可选用**除ESP之外**的任一32位通用寄存器与之搭配。

以下几种情况，缺省段不受超越前缀影响[4]

1. 取指令时只能用CS段
2. 压栈时的目的地址和出栈时的源地址只能用SS段
3. 串操作指令中的目的串只能用ES[5]

2.8 寻址方式的有关问题及应用举例

1.注意某些指令隐含操作数的寻址方式

80x86汇编指令按带操作数的个数分为3类：

- 不带操作数的指令；
- 单操作数的指令；

- 双操作数的指令；

但这些指令的执行都要涉及到源操作数和目的操作数的寻址问题。只不过缺省的操作数的寻址方式都是隐含的。
例：PUSH BUFA; 单操作数指令。其中目的操作数被隐含。显式给出的源操作数BUFA是字类型变量，寻址方式为直接寻址；目的操作使用SP/ESP的寄存器间接寻址。
　CBW　；隐含源和目的操作数。源操作数为AL,寄存器寻址；目的操作数为AX,寄存器寻址。

2.有的指令对寻址方式有特别的要求

例：　XCHG AX, 50; 一般源操作数可以为立即寻址；但在交换指令中源操作数不能为立即寻址。

3.双操作数指令中，源和目的操作数不能同为存储器寻址方式

例：MOV BYTE PTR[SI], [DI];　ERROR
　ADD [EBX+EDI*4+10], COUNT;ERROR COUNT为字变量

即双操作数的寻址方式只能是以下几种组合：
(1)寄存器对寄存器。
(2)寄存器与存储器。
(3)源操作数为立即寻址，目的操作数为寄存器或存储器之一。

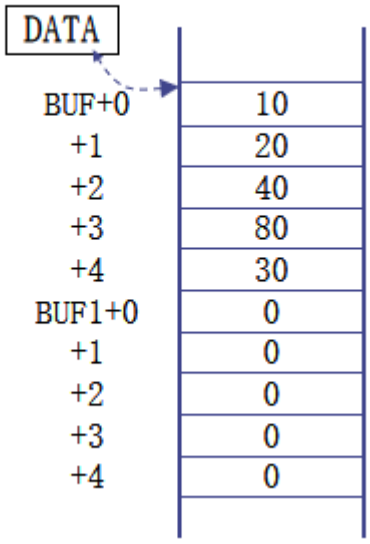
4.两个操作数的类型都不明确时必须使一个明确。

这个就不举例了

5.寻址方式的举例

例：已知数据段定义和存储示意图如下：

```
DATA SEGMENT
      BUF DB 10,20,40,80,30
      BUF1 DB 5 DUP(0)
DATA ENDS
      ...
      ...
```



(1)分别利用直接寻址，寄存器间接寻址，变址寻址和基址加变址寻址，将该段中**BUF+3单元中的内容送到AL中。**

- 直接寻址
  MOV AL, BUF+3

- 寄存器间接寻址
  LEA BX, BUF+3 ；将BUF+3相对于段首地址的偏移地址EA=3送入BX中。
  MOV AL, [BX]

  > 相当于 ([BX])->(AL)
  >
  > 其实平常的MOV AL, BL 也是(BL)->(AL)

- 寄存器变址寻址
  LEA BX, BUF ；取BUF的EA → BX
  MOV AL, [BX+3]; ((BX)+3) → AL
- 基址加变址寻址
  LEA BX, BUF;
  MOV SI, 3;
  MOV AL, [BX+SI];

(2)利用寄存器间接寻址，变址寻址和基址加变址寻址，将**BUF为首址的连续5个字节单元的内容**分别送到以BUF1为首址的连续字节单元中。

```
a. 寄存器间接寻址
   …
      LEA SI, BUF
      LEA DI, BUF1;
      MOV CX, 5;
A:    MOV AL, [SI];
      MOV [DI], AL;
      INC SI;
      INC DI;
      DEC CX;
      JNZ A;
    …
b. 变址寻址
   …
      MOV SI, 0;
      MOV CX, 5;
A:    MOV AL, BUF[SI];BUF是字节内容
      MOV BUF1[SI], AL;
      INC SI;
      DEC CX;
      JNZ A;
    …
c. 基址加变址寻址
   …
      LEA BX, BUF
      LEA BP, BUF1
      MOV SI, 0;
      MOV CX, 5;
A:    MOV AL, [BX][SI];
      MOV DS:[BP][SI], AL;
      INC SI;
      DEC CX;
      JNZ A;
    …

```

说明：
(1)寄存器间接寻址，变址寻址和基址加变址寻址都能用来传送一片连续存储区的内容。
(2)a,b,c三种连续寻址中，b的寻址最直观，可读性最好。

小结

本章介绍了6种寻址方式。除了寄存器寻址和立即寻址外，**其他4种寻址方式**所表示的操作对象都**存放在内存中**，**都要计算偏移地址EA和确定段寄存器。**（就是存储器方式）

> 寄存器对象放在CPU里

1. 直接寻址方式的格式：段寄存器：[EA]：EA的值在格式中直接给出。或由含有变量地址表达式算出。
2. 寄存器间接寻址格式：[R]；　　EA=(R)
3. 变址寻址格式：[R*F+V]；　EA=(R)*F+V
4. 基址变址格式：[BR+IR*F+V]；EA=(BR)+ (IR)*F+V

操作数使用格式是要求重点掌握的内容之一，其中要注意：

- 系统对格式的**R，BR，IR**规定：16位寄存器，BX，BP，DI，SI之一和32位寄存器。
- 格式中段属性问题：
  - 默认情况：BP，EBP，ESP指SS段，其他指DS。
  - 显式情况：是显式给的段寄存器。
- 格式中的类型属性问题：凡在格式**有变量，标号或含有变量表达式，则操作数都有类型**，其类型就是变量或标号的类型。
- 在实际应用中，若要连续存取内存中一片连续单元的内容时，则可供选用的寻址方式是：寄存器间接寻址，变址寻址，基址加变址。

额外补充注意事项

其他有什么不清楚的指令直接去用DOS测试一下就好，通不过会报错的

- 两个段寄存器之间不能直接相互传值 MOV DS,SS是不行的
- MOV ES, AL: 源不能比目标长度短，报错：Wrong type of register
- MOV AL, BX: 可行但mov的源比目标长度大，会导致数据丢失。警告：Operand types must match
- MOV CS, AX: 错误，CS不能作为目标寄存器，CS系统自动放。Illegal use of CS register
- MOV DS, 1230H: 错误，立即数不能直接送段寄存器。Immediate mode illegal
- POP CS: 不能POP CS段寄存器。报错：Ilegal use of CS register
- PUSH CS: 正确
- PUSH BL: BL的大小不符合。警告：Illegal size for operand。PUSH只能推16位和32位的
- CMP 12H, CL：错误，CMP第一个操作数不能是立即数。报错：Immediate mode illegal

---

1. 书P3的1.1.3 ↩
2. 基本上都是学的16位的 ↩
3. 当R是**BP,EBP,SP,ESP**, 则操作对象在当前**堆栈**中，即操作数地址为“**SS:[R]**”； ↩
4. 超越前缀就是显示指定 DS:[ ] 这种，见书P42 ↩
5. 见5.1节 ↩