# 4.4 指令和数据的寻址方式

# 指令和数据的寻址方式

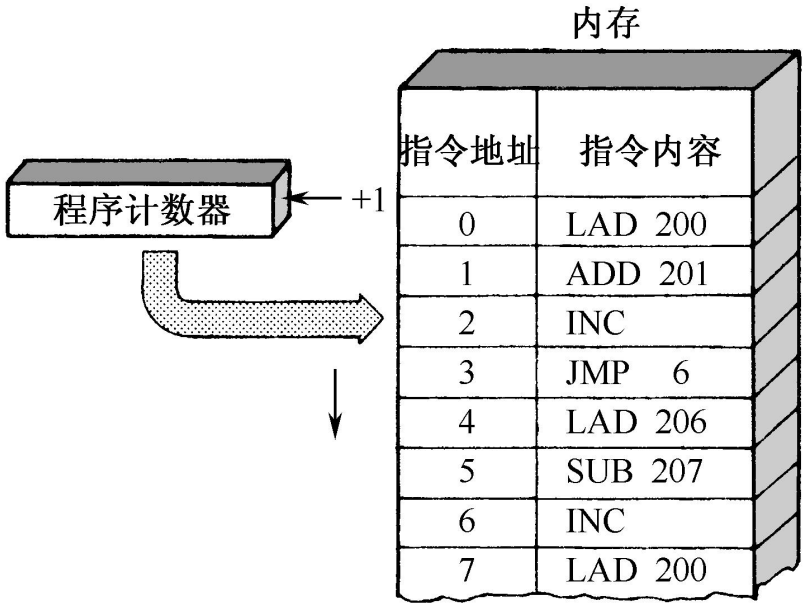
- 问题
  - 确定本条指令中各操作数的地址
  - 下一条指令的地址

- 寻址方式是指CPU根据指令中给出的地址码字段寻找相应的操作数的方式，它与计算机硬件结构紧密相关，而且对指令的格式和功能有很大的影响

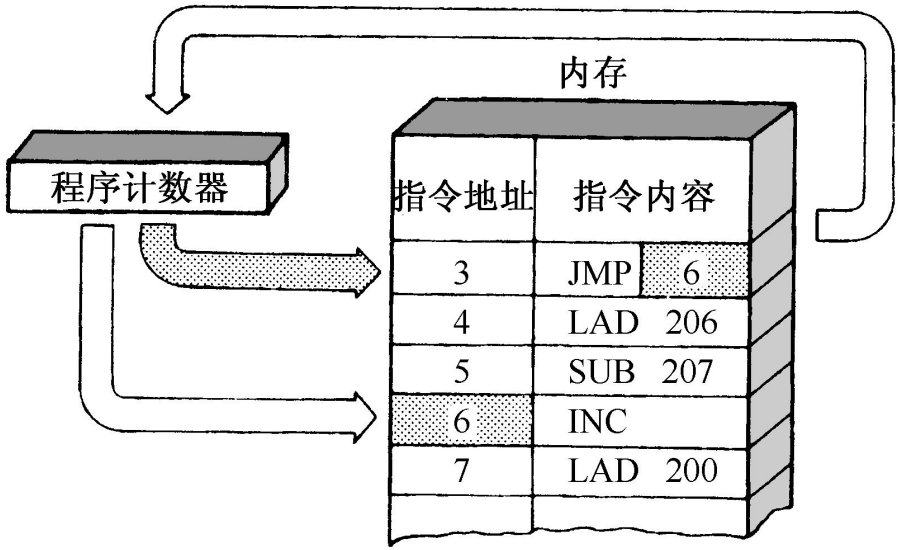
# 提纲

# 4.4.1 指令的寻址方式

- 顺序方式
  - PC+1
- 跳跃方式




(a) 指令的顺序寻址方式




(b) 指令的跳跃寻址方式

# 4.4.2 操作数的寻址方式

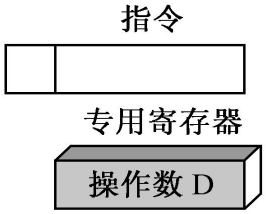
- 形成操作数有效地址的方法，称为寻址方式
- 操作数通常放在哪儿呢?

# 4.4.2 操作数的寻址方式

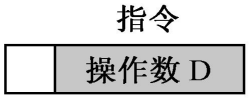
- 操作数包含在指令中；
- 操作数包含在CPU的某一个内部寄存器中；
- 操作数包含在主存储器中；
- 操作数包含在I/O设备的端口中

- 根据操作数放在不同的地方，从而派生各种不同的寻址方式，往往不同的计算机具有不同的寻址方式。

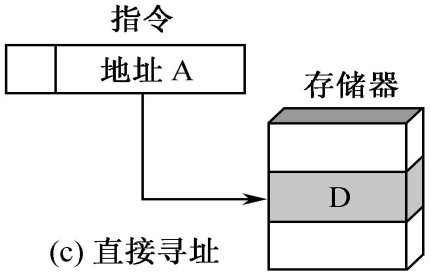
# 4.4.2 操作数的寻址方式



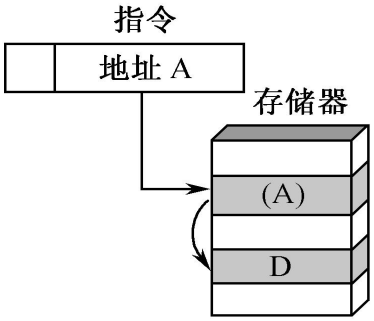

(a) 隐含寻址



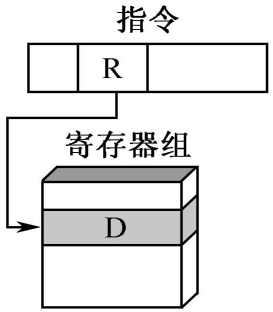

(b) 立即寻址



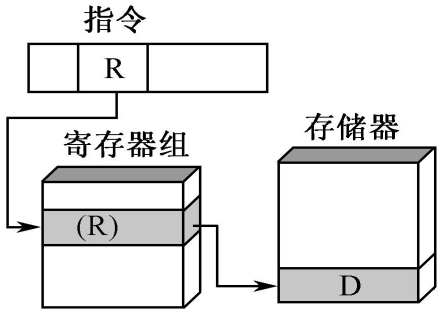

(c) 直接寻址



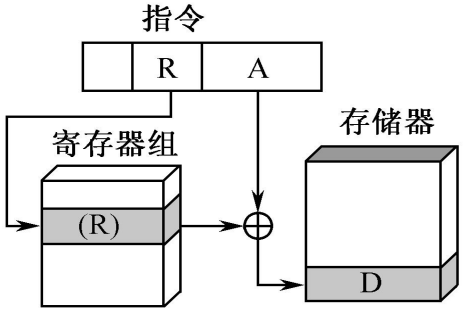

(d) 间接寻址



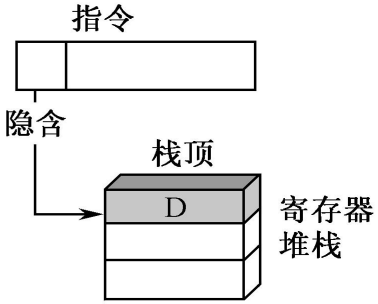

(e) 寄存器寻址




(f) 寄存器间接寻址




(g) 偏移寻址




(h) 堆栈寻址

# 4.4.2 操作数的寻址方式

- 1、隐含寻址

- 指令中隐含着操作数的地址
- 如某些运算，隐含了累加器AC作为源和目的寄存器
- 如8086汇编中的STC指令（标志寄存器操作指令），设置进位标志寄存器的CF为1

# 4.4.2 操作数的寻址方式

- 2、立即寻址

- 立即寻址是一种特殊的寻址方式，指令中在操作码字段后面的部分不是通常意义上的操作数地址，而是操作数本身，也就是说数据就包含在指令中，只要取出指令，就取出了可以立即使用的操作数，因此，这样的操作数被称为立即数
- 指令格式：操作码θ 操作数A

# 4.4.2 操作数的寻址方式

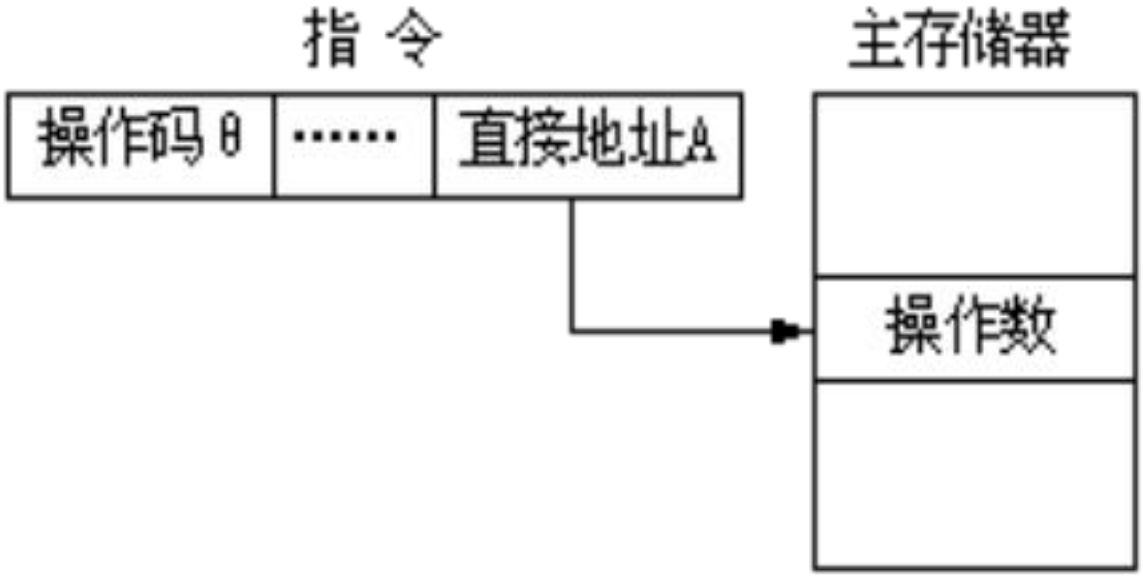
- 2、立即寻址

- 特点：在取指令时，操作码和操作数被同时取出，不必再次访问存储器，从而提高了指令的执行速度。
- 但是，因为操作数是指令的一部分，不能被修改
- 而且对于定长指令格式，操作数的大小将受到指令长度的限制，所以这种寻址方式灵活性最差
- 通常用于给某一寄存器或主存单元赋初值，或者用于提供一个常数

# 4.4.2 操作数的寻址方式

- 3、直接寻址

- 指令中地址码字段给出的地址A就是操作数的有效地址EA(Effective Address)，即EA = A



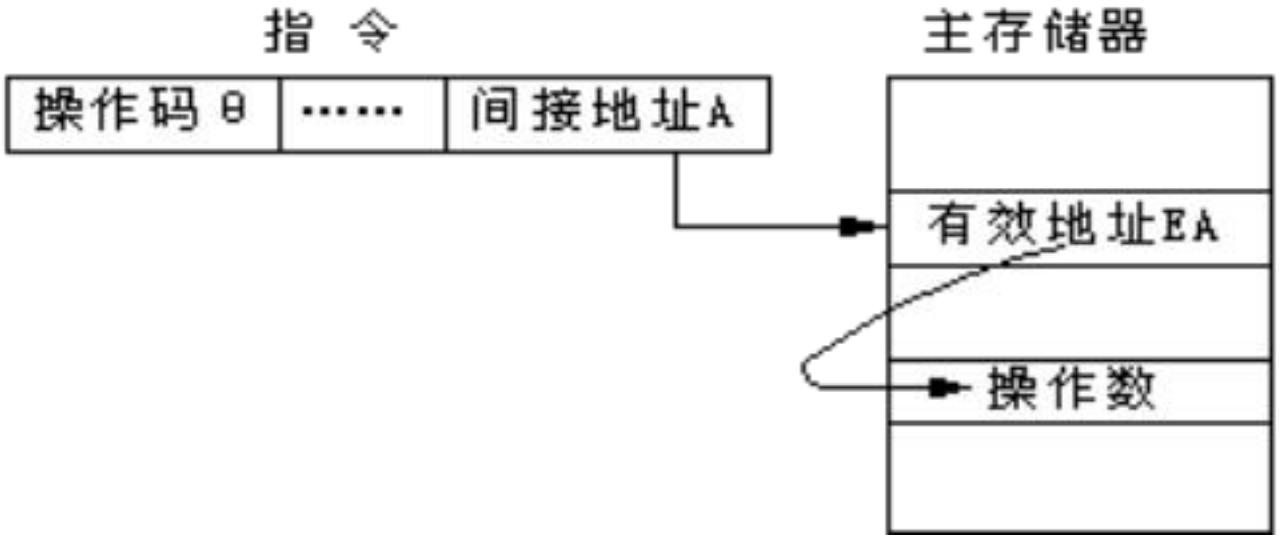
# 4.4.2 操作数的寻址方式

- 操作数地址是不能修改的，与程序本身所在的位置无关，所以又叫做绝对寻址方式
- 在早期的计算机中，主存储器的容量较小，指令中地址码的位数要求不长，采用直接寻址方式简单快速，也便于硬件实现，因此，常被作为主要的寻址方式
- 但在现代，随着计算机主存容量的不断扩大，所需的地址码将会越来越长。指令中地址码的位数将不能满足整个主存空间寻址的要求，因此直接寻址方式受到了很大的限制
- 另外，在指令的执行过程中，为了取得操作数，必须进行访存操作，降低了指令的执行速度

# 4.4.2 操作数的寻址方式

- 4、间接寻址

- 间接寻址意味着指令的地址码部分给出的地址A不是操作数的地址，而是存放操作数地址的主存单元的地址，简称操作数地址的地址。操作数的有效地址的计算公式为：EA = (A)

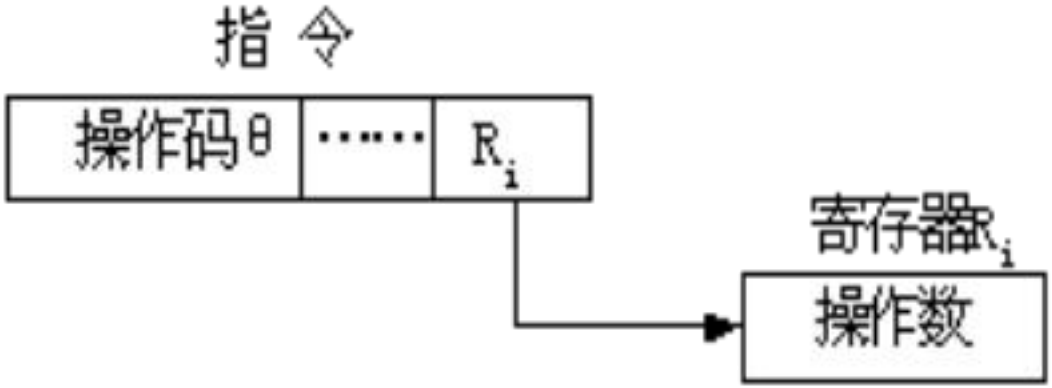


# 4.4.2 操作数的寻址方式

- 特点：因为操作数的有效地址在主存储器中，可以被灵活的修改而不必修改指令，从而使间接寻址要比直接寻址灵活得多
- 但是，间接寻址在指令执行过程中至少需要两次访问主存储器才能取出操作数，严重降低了指令执行的速度。

# 4.4.2 操作数的寻址方式

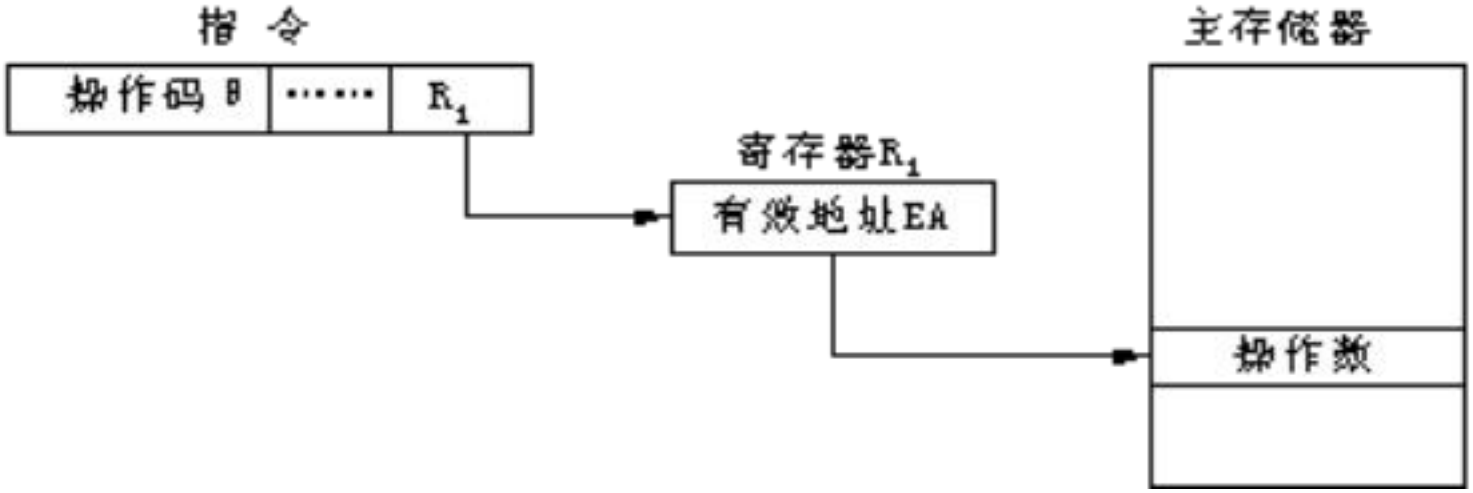
- 5、寄存器寻址
- 在指令的地址码部分给出CPU内某一通用寄存器的编号，指令的操作数存放在相应的寄存器中，即EA=$R_i$




- 优点：
  - 由于寄存器在CPU的内部，指令在执行时从寄存器中取操作数比访问主存要快得多；
  - 由于寄存器的数量较少，因此寄存器编号所占位数也较少，从而可以有效减少指令的地址码字段的长度

# 4.4.2 操作数的寻址方式

- 6、寄存器间接寻址
- 为了克服间接寻址中多次访存的缺点，可采用寄存器间接寻址，即将操作数放在主存储器中，而操作数的地址放在某一通用寄存器中，然后在指令的地址码部分给出该通用寄存器的编号，这时有EA=($R_i$)




- 这种寻址方式的指令较短，并且在取指后只需一次访存便可得到操作数，因此指令执行速度较前述的间接寻址方式要快，也是目前在计算机中使用较为广泛的一种寻址方式

# 4.4.2 操作数的寻址方式

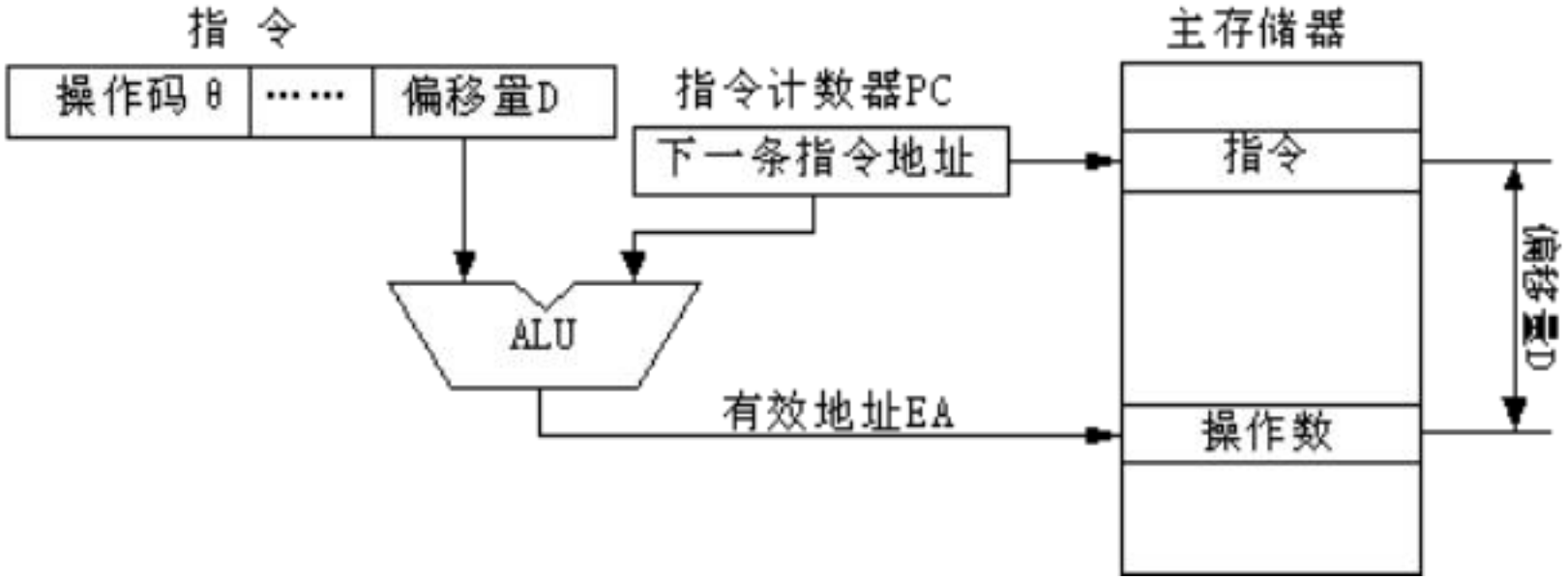
- 7、偏移寻址

$$EA = (R) + A$$

# 4.4.2 操作数的寻址方式

- （1）相对寻址

- 由程序计数器PC提供基准地址，而指令的地址码部分给出相对的位移量A，两者相加后作为操作数的有效地址，即：EA = (PC) + A



# 4.4.2 操作数的寻址方式

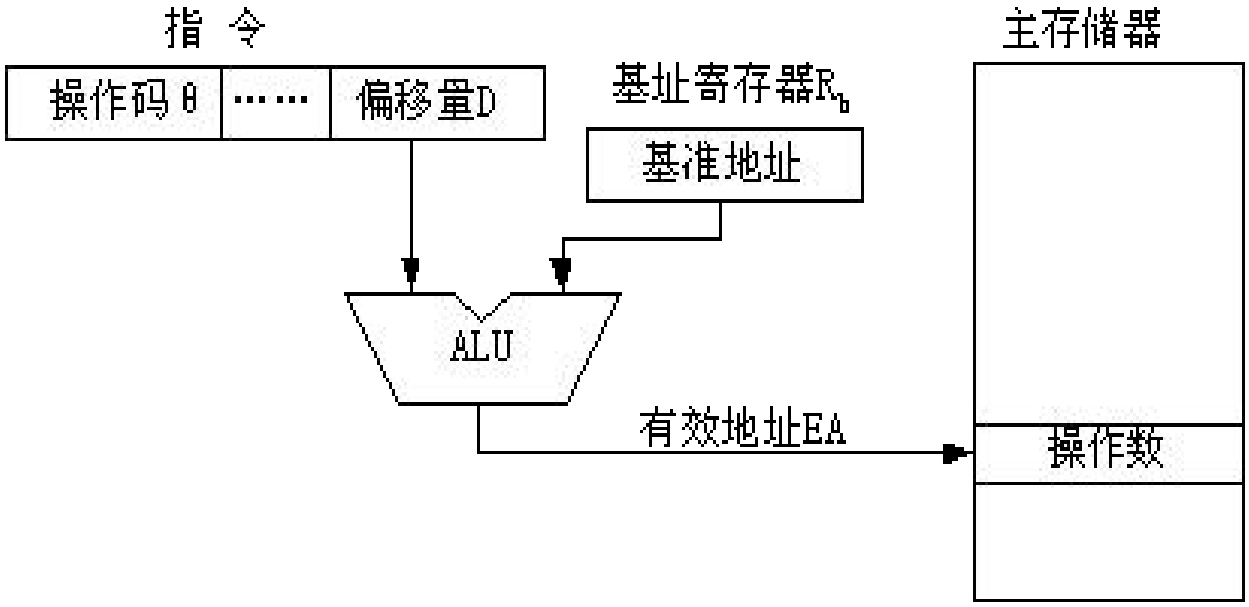
- 寻址方式使程序模块可采用浮动地址，编程时只要确定程序内部操作数与指令之间的相对距离，而无需确定操作数在主存储器中的绝对地址，这样，将程序安排在主存储器的任意位置都不会影响程序执行的正确性。

# 4.4.2 操作数的寻址方式

- （2）基址寻址

- 在基址寻址方式中，指令的地址码部分给出偏移量D，而基准地址放在基址寄存器$R_b$中，最后操作数的有效地址仍然是由基准地址A与偏移量D相加而成，即：$EA = (R_b) + D$
- 用哪一个寄存器作为基址寄存器也必须在硬件设计时就事先规定，基址寄存器$R_b$中的内容称为基准地址，该值可正可负

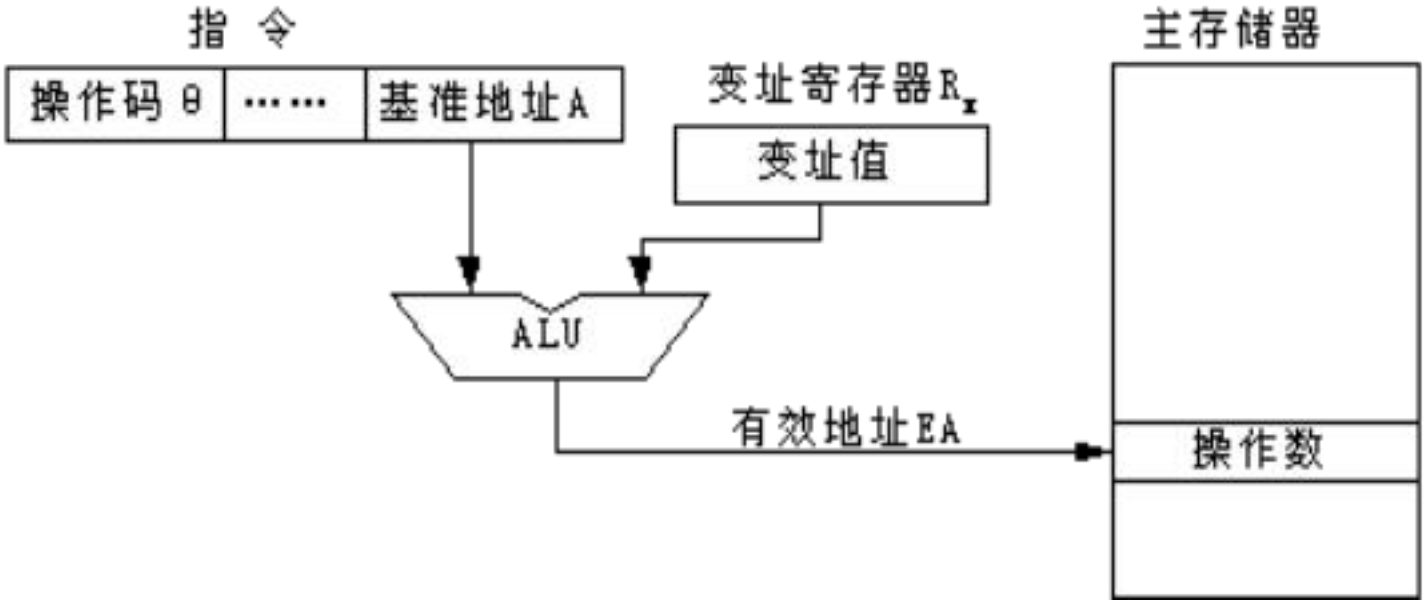
# 4.4.2 操作数的寻址方式




- 基址寄存器的位数可以设置得很长，从而可以在较大的存储空间中寻址

# 4.4.2 操作数的寻址方式

- （3）变址寻址
- 变址寻址就是将指令的地址码部分给出的基准地址A与CPU内某特定的变址寄存器Rx中的内容相加，以形成操作数的有效地址，即：EA = A + (Rx)。
- 用哪一个寄存器作为变址寄存器必须在硬件设计时就事先规定，变址寄存器Rx中的内容称为变址值，该值可正可负。

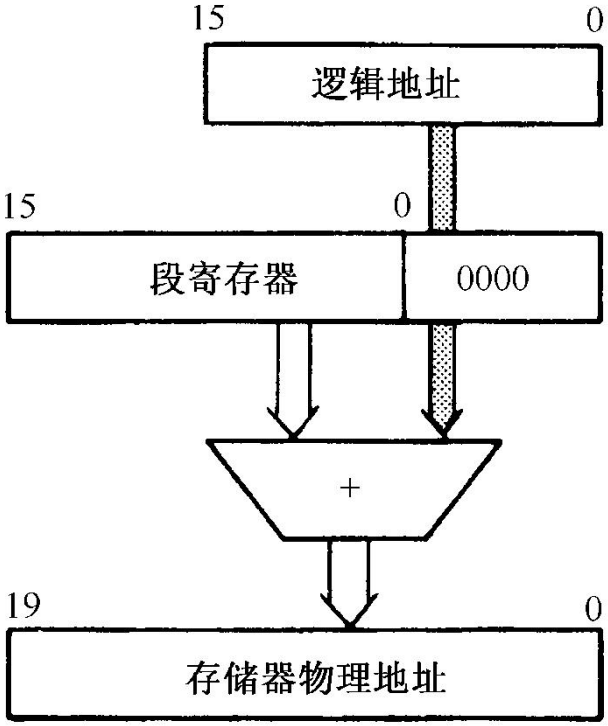


# 4.4.2 操作数的寻址方式

- 变址寻址方式是一种被广泛采用的寻址方式，最典型的应用就是将指令的地址码部分给出的地址A作为基准地址，而将变址寄存器Rx中的内容作为修改量
- 在遇到需要频繁修改操作数地址时，无须修改指令，只要修改Rx中的变址值就可以了，这对于数组运算、字符串操作等一些进行成批数据处理的指令是很有用的

# 4.4.2 操作数的寻址方式

- 8、段寻址方式




- 9、堆栈寻址　　POP, PUSH

# 4.4.2 操作数的寻址方式

例：一种二地址 RR 型，RS 型指令结构如下所示

| 6 位 | 4 位 | 4 位 | 1 位 | 2 位 | 16 位 |
|---|---|---|---|---|---|
| OP | 源寄存器 | 目标寄存器 | I | X | D（偏移量） |

其中源寄存器，目标寄存器都是通用寄存器，I 为间接寻址标志位，X 为寻址模式字段，D 为偏移量字段。通过 I, X, D 的组合，可构成一个操作数的寻址方式，其有效地址 E 的算法及有关说明列于下表：

| 寻址方式 | I | X | 有效地址 E 算法 | 说明 |
|---|---|---|---|---|
| (1) | 0 | 00 | E=D | D 为偏移量 |
| (2) | 0 | 01 | 指令地址=(PC)+D | PC 为程序计数器 |
| (3) | 0 | 10 | E=($R_x$)+D | $R_x$ 为变址寄存器 |
| (4) | 1 | 11 | E=(R) | R 为通用寄存器 |
| (5) | 1 | 00 | E=(D) | |
| (6) | 0 | 11 | E=($R_b$)+D | $R_b$ 为基址寄存器 |

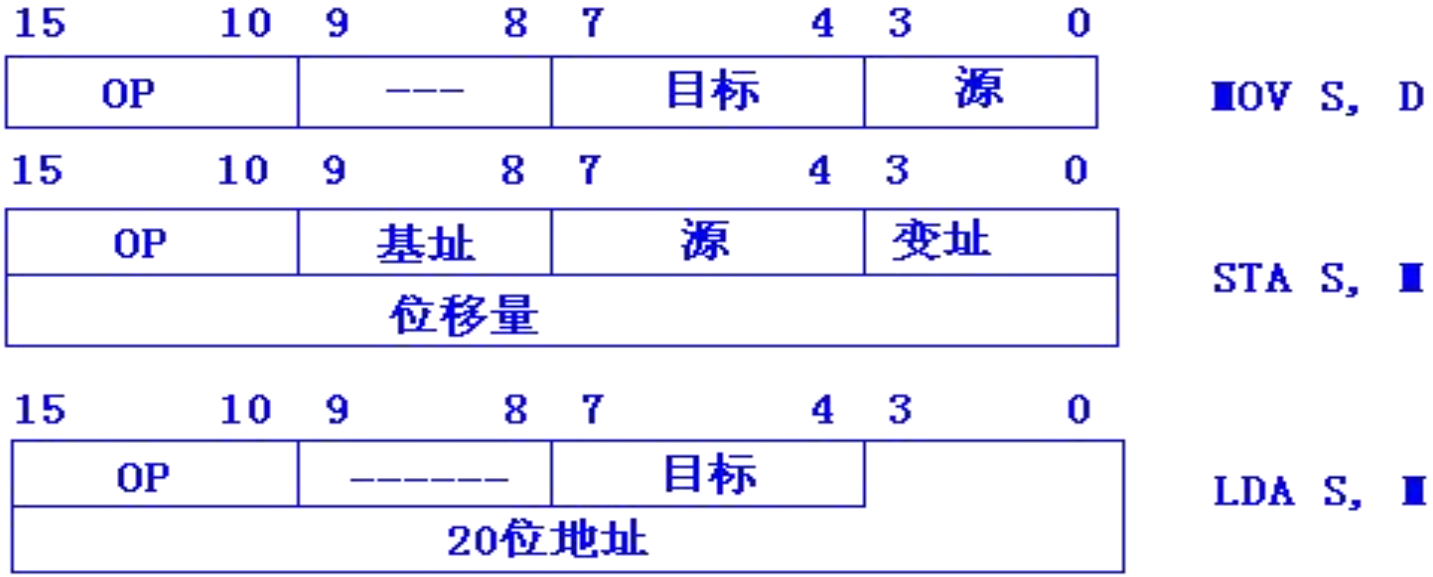
请写出表中 6 种寻址方式名称，并说明主存中操作数的位置。

# 4.4.2 操作数的寻址方式

- 解：
  - （1）直接寻址，操作数在有效地址E=D的存储单元中
  - （2）相对寻址
  - （3）变址寻址，操作数在E=(RX) + D的存储单元中
  - （4）寄存器间接寻址，通用寄存器的内容指明操作数在主存中的地址
  - （5）间接寻址，用偏移量做地址访主存得到操作数的地址指示器，再按地址指示器访主存的操作数，因此间接寻址需两次访问主存.
  - （6）基址寻址，操作数在E=(Rb) + D 的存储单元中

# 4.4.2 操作数的寻址方式

- 例：某16位机器所使用的指令格式和寻址方式如下所示，该机有两个20位基址寄存器，四个16位变址寄存器，十六个16位通用寄存器，指令汇编格式中的S（源），D（目标）都是通用寄存器，M是主存中的一个单元。三种指令的操作码分别是MOV（OP）=（A）H，STA（OP）=（1B）H，LDA（OP）=（3C）H。MOV是传送指令，STA为写数指令，LDA为读数指令。



# 4.4.2 操作数的寻址方式

- 要求：
  - （1）分析三种指令的指令格式与寻址方式特点。
  - （2）CPU完成哪一种操作所花时间最短？哪一种操作所花时间最长？第二种指令的执行时间有时会等于第三种指令的执行时间吗？
  - （3）下列情况下每个十六进制指令字分别代表什么操作？其中如果有编码不正确，如何改正才能成为合法指令？

    ①（F0F1）H（3CD2）H　　②（2856）H

    ③（6FD6）H　　④（1C2）H

# 4.4.2 操作数的寻址方式

- 解:
  - （1）MOV:单字长，二地址指令属于RR指令
    STA:双字长，二地址指令，RS指令 S为基址寄存器或变址寄存器
    LDA:双字长，二地址指令，RS指令 S为20位地址。
  - （2）MOV单字长取出只需要一次访存，第一个不访问存储器要快。第二种还需要计算有效地址并对存储器进行访问。第二种指令所花费的时间不等于第三种，第三种无需进行有效地址的计算。
  - （3）根据条件。MOV(OP)=(A)h=001010
    STA(OP)=(1B)h =011011
    LDA(OP)=(3C)h=111100
  - (F1F1)h(3CD2)h=(1111 0000 1111 0001 0011 1100 1101 0010)b，正确，表明把主存(13cd2)h地址单元的内存取至15号寄存器
  - （2856）H=（0010 1000 0101 0110）b，单字长指令表明把寄存器6的内容传送到寄存器5号
  - （6FD6）H=0110 1111 1101 0110b操作码不对 应修改为（28D6）h
  - （1C2）H=00000001 1100 0010b操作码不对 应修改为（28C2）h

# 4.5 典型指令

# 提纲

# 4.5.1 指令的分类

- 数据传送指令
  - 数据传送指令主要包括取数指令、存数指令、传送指令、成组传送指令、字节交换指令、清累加器指令、堆栈操作指令等等。这类指令主要用来实现主存和寄存器之间，或寄存器和寄存器之间的数据传送。
- 算术运算指令
  - 这类指令包括二进制定点加、减、乘、除指令，浮点加、减、乘、除指令，求反、求补指令，算术移位指令（带符号移位），比较指令，十进制加、减运算指令等。这类指令主要用于定点或浮点的算术运算，大型机中有向量运算指令，直接对整个向量或矩阵进行求和、求积运算。
- 逻辑运算指令
  - 这类指令包括逻辑加、逻辑乘、按位加、逻辑移位（不带符号移位）等指令，主要用于无符号数的位操作、代码的转换、判断及运算。 移位指令用来对寄存器的内容实现左移、右移或循环移位。

# 4.5.1 指令的分类

- 程序控制指令
  - 程序控制指令也称转移指令。执行程序时，有时机器执行到某条指令时，出现了几种不同结果，这时机器必须执行一条转移指令，根据不同结果进行转移，从而改变程序原来执行的顺序。这种转移指令称为条件转移指令。除各种条件转移指令外，还有无条件转移指令、转子程序指令、返回主程序指令、中断返回指令等。 转移指令的转移地址一般采用直接寻址和相对寻址方式来确定。
- 输入输出指令
  - 输入输出指令主要用来启动外围设备，检查测试外围设备的工作状态，并实现外部设备和CPU之间，或外围设备与外围设备之间的信息传送。

# 4.5.1 指令的分类

- 字符串处理指令
  - 字符串处理指令是一种非数值处理指令，一般包括字符串传送、字符串转换（把一种编码的字符串转换成另一种编码的字符串）、字符串替换（把某一字符串用另一字符串替换）等。这类指令在文字编辑中对大量字符串进行处理。
- 特权指令
  - 特权指令是指具有特殊权限的指令。这类指令只用于操作系统或其他系统软件，一般不直接提供给用户使用。 在多用户、多任务的计算机系统中特权指令必不可少。它主要用于系统资源的分配和管理。
- 其他指令
  - 除以上各类指令外，还有状态寄存器置位、复位指令、测试指令、暂停指令，空操作指令，以及其他一些系统控制用的特殊指令。

# 4.5.2 基本指令系统的操作

- CISC（复杂指令系统计算机）的指令系统一般多达二三百条，例如VAX11/780计算机有303条指令，18种寻址方式。Pentium机也有191条指令，9种寻址方式

- 20%和80%规律：CISC中大约有20%的指令使用频率高，占据了80%的处理机时间，而有80%的不常用指令只占用处理机的20%时间

- 131页的表4.9给出了基本指令系统的操作（从教学目的考虑）

# 4.5.3 精简指令系统

- CISC中，通过增强指令系统的功能，简化了软件，增加了硬件的复杂程度。然而指令复杂了，指令的执行时间必然加长，从而使整个系统的执行时间反而增加，因而在计算机体系结构设计中，软硬件的功能分配必须恰当

- VLSI技术发展引起的问题
  - VLSI工艺要求规整性，而大量复杂指令控制逻辑极其不规整，给VLSI工艺造成了很大的困难
  - 现在用微程序实现复杂指令与用简单指令组成的子程序相比，没有多大的区别。因为现在控制存储器和主存的速度差缩小

# 4.5.3 精简指令系统

- 特点（采用流水线技术）
  - 简单而统一格式的指令译码
  - 大部分指令可以单周期执行
  - 只有LOAD/STORE可以访问存储器
  - 简单的寻址方式
  - 较多的寄存器
  - 对称的指令格式（指令系统中所有的寄存器和存储器单元都可同等对待，所有的指令都可使用各种寻址方式）