
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS
北京邮电大学

第4章 指令系统

4.1 指令系统的发展与性能要求

基本概念

- 指令（Instruction）是指要计算机执行某种操作的命令
- 从计算机组成的层次结构来说，计算机的指令有：
  - 微指令：微程序级的命令，它属于硬件
  - 宏指令：由若干条机器指令组成的软件指令，它属于软件
  - 机器指令：介于微指令与宏指令之间，每条指令可完成一个独立的算术运算或逻辑运算
- 指令系统ISA（Instruction Set Architecture）也称指令集架构
  - 一台计算机中所有机器指令的集合
  - 表征一台计算机性能的重要因素
  - 指令的格式与功能不仅直接影响到机器的硬件结构，也直接影响到系统软件，影响到机器的适用范围

计算机指令系统的发展过程

- 50年代：指令系统只有定点加减、逻辑运算、数据传送、转移等十几至几十条指令
- 60年代后期
  - 增加了乘除运算、浮点运算、十进制运算、字符串处理等指令，指令数目多达一二百条，寻址方式也趋多样化
  - 出现系列计算机（指基本指令系统相同、基本体系结构相同的一系列计算机）
- 70年代末期
  - 大多数计算机的指令系统多达几百条。这种计算机称为复杂指令系统计算机（CISC Complex Instruction Set Computer）
  - 相对于CISC，又出现了便于VLSI技术实现的精简指令系统计算机RISC（Reduced Instruction Set Computer）

指令系统的要求（1）

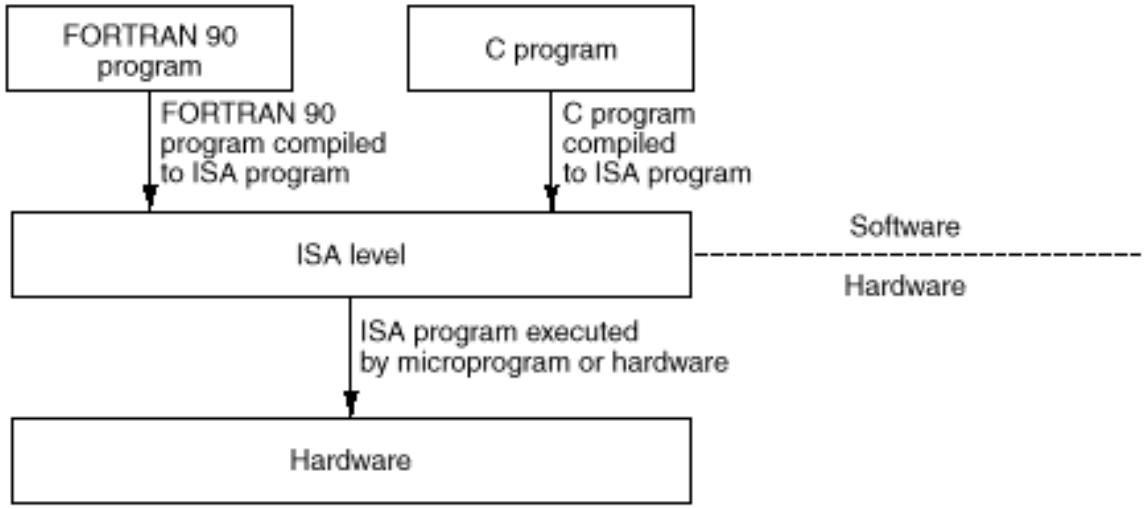
- 完备性：指令系统丰富、功能齐全、使用方便
- 有效性：编写的程序能够高效率的运行
  - 高效率主要表现在程序占据存储空间小、执行速度快
- 规整性
  - 对称性：在指令系统中所有寄存器和存储器单元都可同等对待，所有指令都可使用各种寻址方式
  - 匀齐性：指一种操作性质的指令可以支持各种数据类型
  - 指令格式和数据格式的一致性：指令长度和数据长度有一定的关系，以方便处理和存取

指令系统的要求（2）

- 兼容性
  - 系列机各机种之间具有相同的基本结构和共同的基本指令集，即指令系统是兼容的，各机种上基本软件可以通用
  - 由于新旧机种在结构和性能上有差异，做到所有软件都完全兼容是不可能的，通常要做到“向后兼容”，即在旧处理器上运行的软件仍然可以在处理器上运行

低级语言与硬件结构的关系

- 高级语言：如C，Java等，其语句和用法与具体机器的指令系统无关
- 低级语言：是面向机器的语言，和具体机器的指令系统密切相关
  - 机器语言（二进制语言），用指令代码编写程序
  - 汇编语言（符号语言），用指令助记符来编写程序



高级语言与低级语言比较

比较内容	高级语言	低级语言
对程序员的训练要求 (1) 通用算法 (2) 语言规则 (3) 硬件知识	 有 较少 不要	 有 较多 要
对机器独立的程度	独立	不独立
编制程序的难易程度	易	难
编制程序所需时间	短	较长
程序执行时间	较长	短
编译过程中对计算机资源的要求	多	少

4.2 指令格式

基本概念

- 指令字（简称指令）即表示一条指令的机器字
- 指令格式则是指令字用二进制代码表示的结构形式
- 指令的构成
  - 操作码字段：表示指令的操作特性与功能
  - 操作数地址字段：指定参与操作的操作数的地址

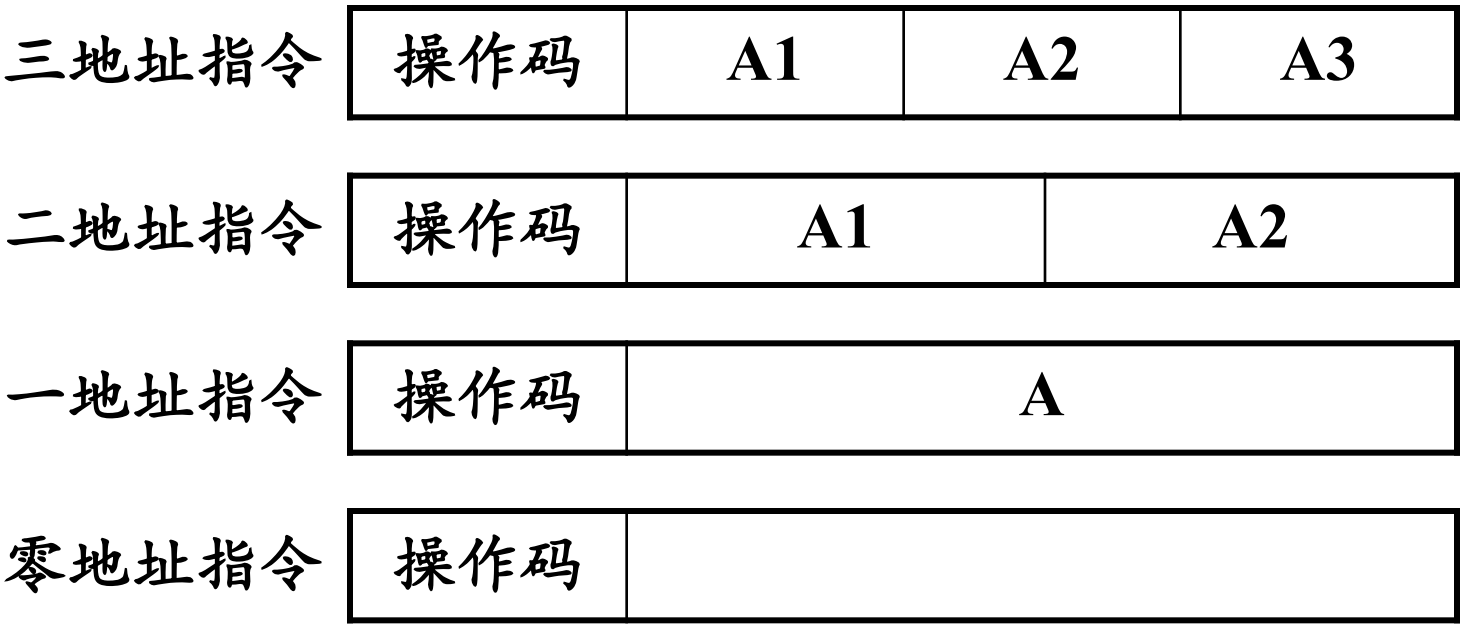
操作码

- 指令系统的每一条指令都有一个操作码，它表示该指令应进行什么样的操作
- 操作码字段的位数一般取决于计算机指令系统的规模。例如：
  - 如果某指令系统只有8条指令，则用3位操作码就可表示
  - 如果有32条指令，那么就需要5位操作码
- 对于一个机器的指令系统，指令字中操作码字段长度可以是固定的，也可以是变长的

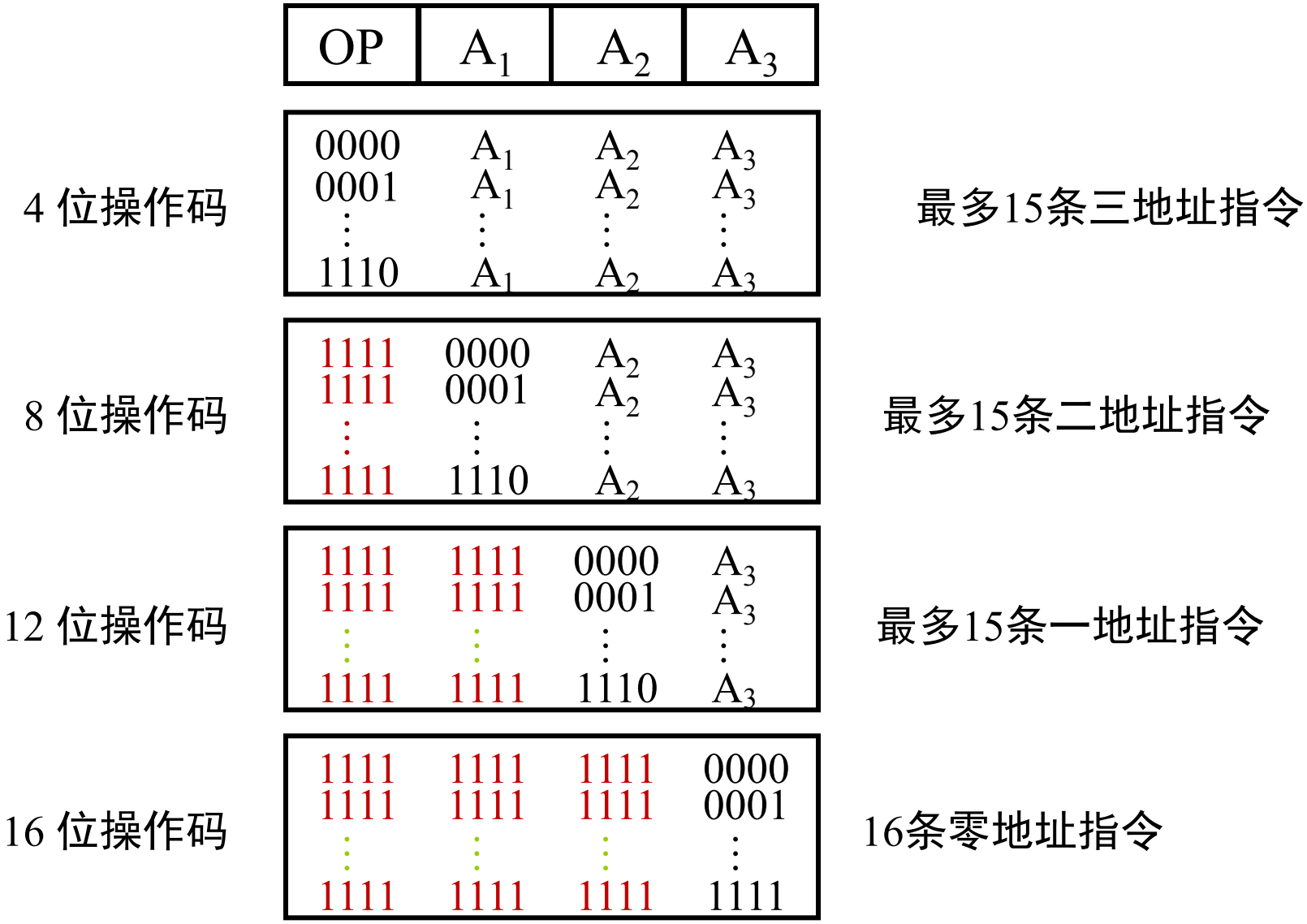
操作数地址

- 按照一条指令中有几个操作数地址，可将该指令称为几操作数指令或几地址指令
  - 三地址指令格式
  - 二地址指令格式
  - 一地址指令格式
  - 零地址指令格式

定长操作码指令格式



扩展操作码指令格式


OP
A1
A2
A3
4 位操作码
0000 A1 A2 A3
0001 A1 A2 A3
1110 A1 A2 A3
最多15条三地址指令
8 位操作码
1111 0000 A2 A3
1111 0001 A2 A3
1111 1110 A2 A3
最多15条二地址指令
12 位操作码
1111 1111 0000 A3
1111 1111 0001 A3
1111 1111 1110 A3
最多15条一地址指令
16 位操作码
1111 1111 1111 0000
1111 1111 1111 0001
1111 1111 1111 1111
16条零地址指令

地址码（1）

- 零地址指令
  - 指令字中只有操作码，而没有地址码
- 一地址指令
  - 称为单操作数指令，指令中只有一个操作数
  - 或者是，指令中默认以运算器中累加寄存器AC中的数为操作数1，而地址码字段所指明的数为操作数2，操作结果又放回累加寄存器AC中

地址码（2）

- 二地址指令常称为双操作数指令
  - 它的两个地址码字段分别指明参与操作的两个数在内存中或运算器中通用寄存器的地址，A1作存放操作结果的地址

    (A1)　OP　(A2)　→　A1
- 三地址指令
  - 指令字中有三个操作数地址

    (A1)　OP　(A2)　→　A3
  - A1为被操作数地址，也称源操作数地址；
  - A2为操作数地址，也称终点操作数地址；
  - A3为存放结果的地址。
  - 其中，A1、A2、A3可以是内存中的单元地址，也可以是运算器中通用寄存器的地址

地址码（3）

- 按照各操作数的物理位置不同，可分为三种类型：
  - 存储器–存储器（SS）型指令：参与操作的数都放在内存里，因此执行这种指令需多次访问内存
  - 寄存器–寄存器（RR）型指令：从寄存器中取操作数，把操作结果放到另一寄存器。机器执行寄存器-寄存器型指令的速度很快，因为执行这类指令，不需要访问内存，但需要多个寄存器
  - 寄存器–存储器（RS）型指令：执行此类指令时，既要访问内存单元，又要访问寄存器

指令字长度（1）

- 指令字长度：一个指令字中包含二进制代码的位数
- 机器字长：计算机能直接处理的二进制数据的位数，它决定了计算机的运算精度
  - 机器字长度通常与寄存器的位数一致
- 指令分类
  - 单字长指令：指令字长度等于机器字长度
  - 半字长指令：指令字长度等于半个机器字长度
  - 双字长指令：指令字长度等于两个机器字长度

指令字长度（2）

- 使用多字长指令，目的在于提供更大的地址空间
  - 主要缺点是必须两次或多次访问内存以取出一整条指令，降低了CPU的运算速度
- 等长指令字结构：各种指令字长度是相等的。这种指令字结构简单，且指令字长度是不变的
- 变长指令字结构：各种指令字长度随指令功能而异。结构灵活，代码密度高，能充分利用指令长度，但指令的控制较复杂

指令助记符

- 由于硬件只能识别1和0，所以采用二进制操作码是必要的，但不适合人阅读和书写程序
- 因此，每条指令通常用3个或4个英文缩写字母来表示。这种缩写码叫做指令助记符

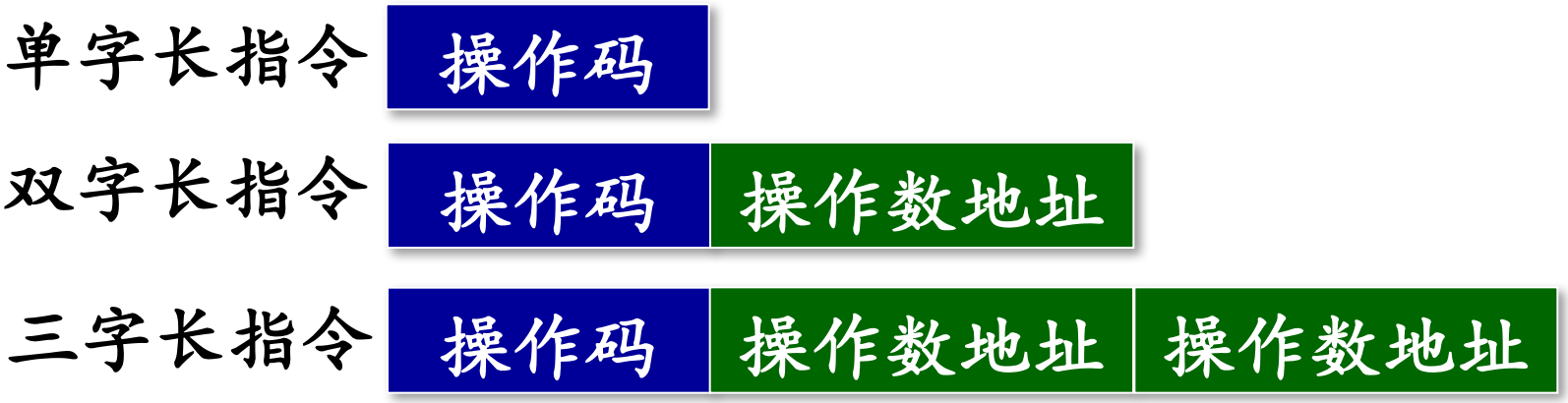
典型指令助记符

典型指令	指令助记符	二进制操作码
加法	ADD	001
减法	SUB	010
传送	MOV	011
跳转	JMP	100
转子	JSR	101
存储	STR	110
读数	LDA	111

- 注意：在不同的计算机中，指令助记符的规定是不一样的。把人认识的指令助记符转换成与之对应的二进制码后机器才能“理解”。这种转换借助汇编程序自动完成，汇编程序相当于一个“翻译”

举例1：八位微型计算机的指令格式

- 8位微型机字长只有8位，指令结构是一种可变字长形式，包含单字长、双字长、三字长指令等多种格式
- 内存按字节编址



举例2：PDP/11系列机指令格式

- PDP/11系列机是DEC（数字设备公司）在20世纪60年代末推出的小型机
- 被美国工业研究所评选为“1970年最有影响的技术产品”，后来发展出20余种产品
- 1977年，32位的VAX-11/780小型机走下生产线
- 1998年1月，康柏电脑公司以96亿美元的价格收购DEC
- 2001年9月惠普公司宣布以250亿美元，按换股方式收购康柏电脑公司

举例2：PDP/11系列机指令格式

- PDP/11系列机指令字长为16位
- 操作码字段是不等长的。这样做可以扩展操作码以定义更多的指令。但操作码字段不固定，使控制器的设计复杂化

指令类型 \ 指令位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
单操作数指令	操作码（10位）										目标地址(6位)					
双操作数指令	操作码(4位)				源地址(6位)						目标地址(6位)					
转移指令	操作码（8位）								位移量（8位）							
转子指令	操作码（7位）							寄存器号								
子程序返回指令	操作码（13位）															
条件码操作指令	操作码（11位）											S	N	Z	V	C

举例3： Pentium指令格式

- 指令字长度是可变的，从 1 字节到12字节，还可以带前缀
- Pentium指令是典型的CISC指令，之所以选择可变长，主要是为了向后兼容
- 指令的前缀是可选项，其作用是对其后的指令本身进行显示约定，每个前缀占 1 个字节

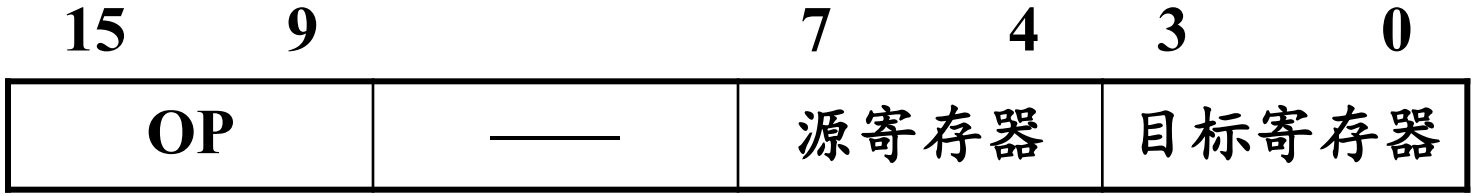
指令格式

8086；　80386+

例1

指令格式如下所示，其中OP为操作码，试分析指令格式的特点

解：



(1)16位长二地址指令

(2)操作码字段OP可以指定128条指令

(3)源寄存器和目标寄存器都是通用寄存器（可分别指定16个），所以是RR型指令，两个操作数均在寄存器中

(4)这种指令结构常用于算术逻辑运算类指令

4.3指令和数据的寻址方式

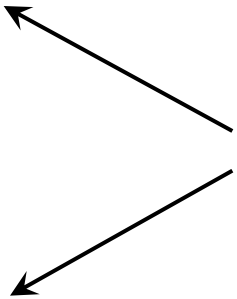
基本概念（1）

- 存储器既可用来存放数据，又可用来存放指令
- 操作数或指令在存储器中的地址：某个操作数或某条指令存放在某个存储单元时，其存储单元的编号
- 在存储器中，写入或读出操作数/指令字的方式有：
  - 地址指定方式
  - 相联存储方式（按内容寻址方式）
  - 堆栈存取方式

几乎所有的计算机中均采用这种方式

基本概念（2）

- 当采用地址指定方式时，形成操作数或指令地址的方式，称为寻址方式
- 寻址方式分为两类
  - 指令寻址方式
    - 顺序寻址方式
    - 跳跃寻址方式
  - 数据寻址方式



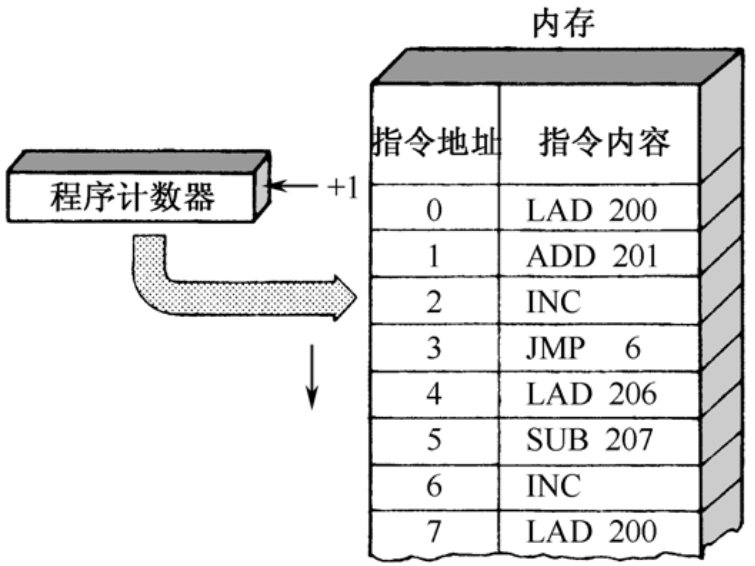
前者比较简单
后者比较复杂

指令寻址方式

- 顺序寻址方式
  - 指令地址在内存中按顺序安排，当执行一段程序时，通常是一条指令接一条指令的顺序执行
  - 从存储器取出第一条指令，然后执行这条指令；接着从存储器取出第二条指令，在执行第二条指令；接着再取出第三条指令……这种程序顺序执行的过程，称之为指令的顺序寻址方式

顺序寻址方式

- 必须使用程序计数器（又称指令指针寄存器）PC来计数指令的顺序号，该顺序号就是指令在内存中的地址
- 指令顺序寻址方式示意图

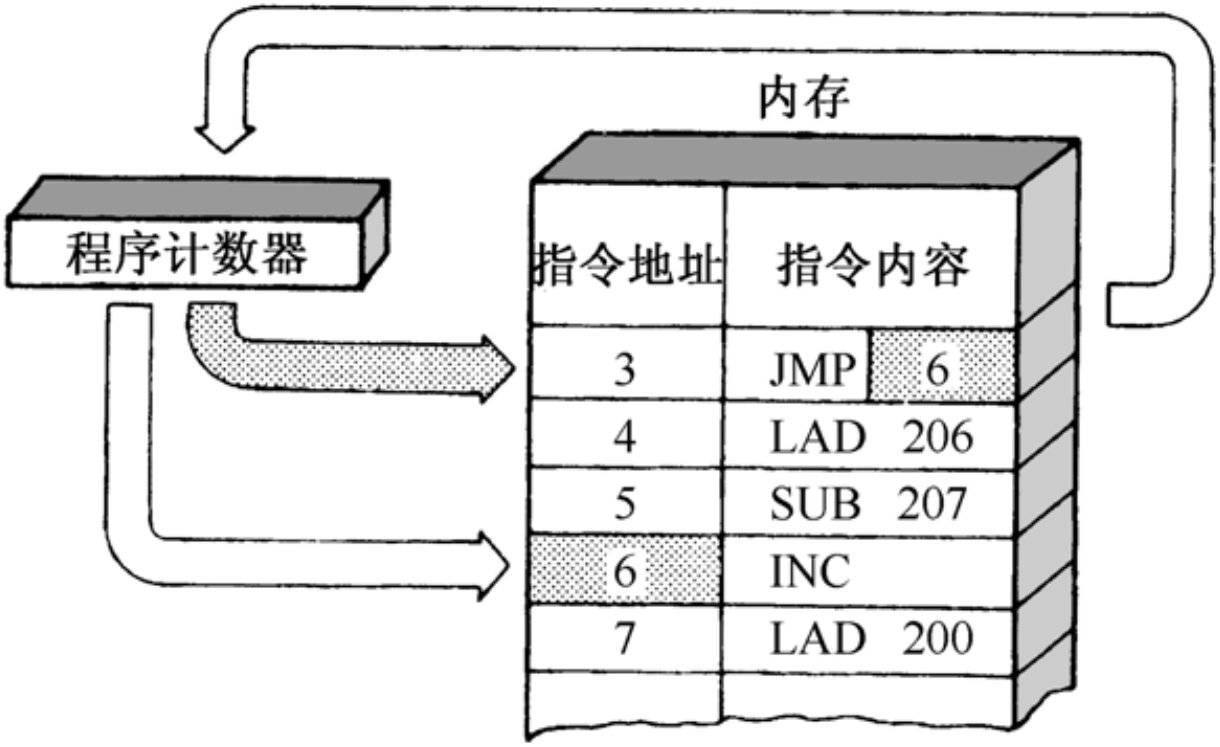


(a) 指令的顺序寻址方式

跳跃寻址方式

- 所谓跳跃寻址，是指下条指令的地址码不是由程序计数器给出，而是由本条指令给出
- 程序跳跃后，按新的指令地址开始顺序执行，指令计数器的内容也必须相应改变，以便及时跟踪新的指令地址
- 采用指令跳跃寻址方式，可以实现程序转移或构成循环程序，从而能缩短程序长度，或将某些程序作为公共程序引用
- 指令系统中的各种条件转移、无条件转移和子程序调用指令等，就是为了实现指令的跳跃寻址而设置的

指令跳跃寻址方式示意图



(b) 指令的跳跃寻址方式

操作数寻址方式

- 形成操作数的有效地址的方法，称为操作数的寻址方式
- 一种单地址指令的结构如下所示，其中用X，I，A各字段组成该指令的操作数地址

操作码(OP)	变址(X)	间址(I)	形式地址(A)

- 指令中操作数字段的地址码是由形式地址（也称偏移量）和寻址方式特征位等组合形成。
  - 寻址过程就是把操作数的形式地址，变换为操作数的有效地址的过程

1. 隐含寻址

- 在指令中不显式给出而是隐含（默认）给出操作数的地址
- 例如，一种单地址的指令格式，没有在地址字段中指明第2个操作数地址，而是默认累加寄存器作为第2个操作数地址，所以累加器对单地址指令格式来说是隐含地址

例：乘法指令 mul bh

ax ← (al) x (bh)

累加器，隐含寻址

2. 立即寻址

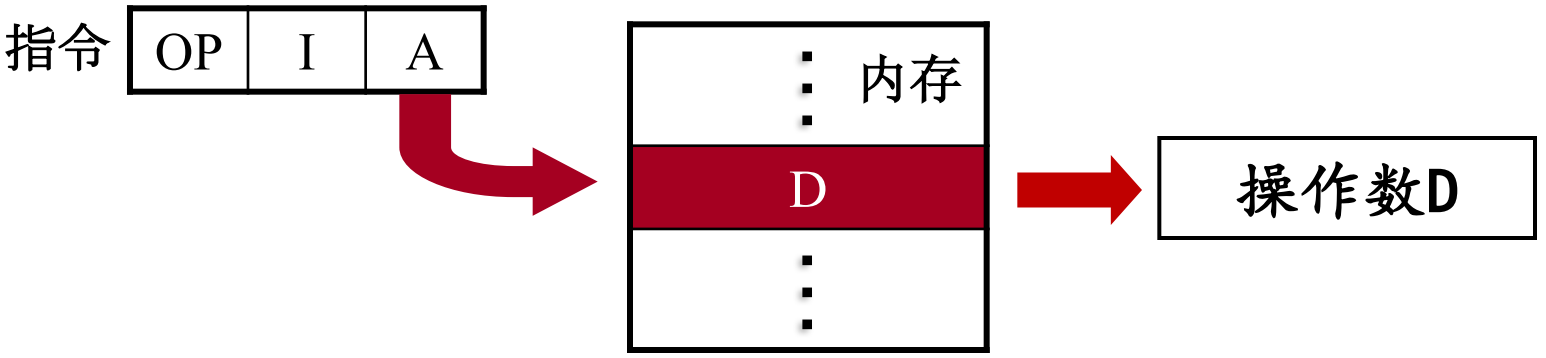
- 指令的地址字段指出的不是操作数的地址，而是操作数本身。这种方式的特点是指令执行速度较快，取指令的同时就取到了操作数，不需要访问内存取操作数
- 例如：单地址的移位指令格式为

- 地址字段不是操作数的地址，而是一个操作数

3. 直接寻址

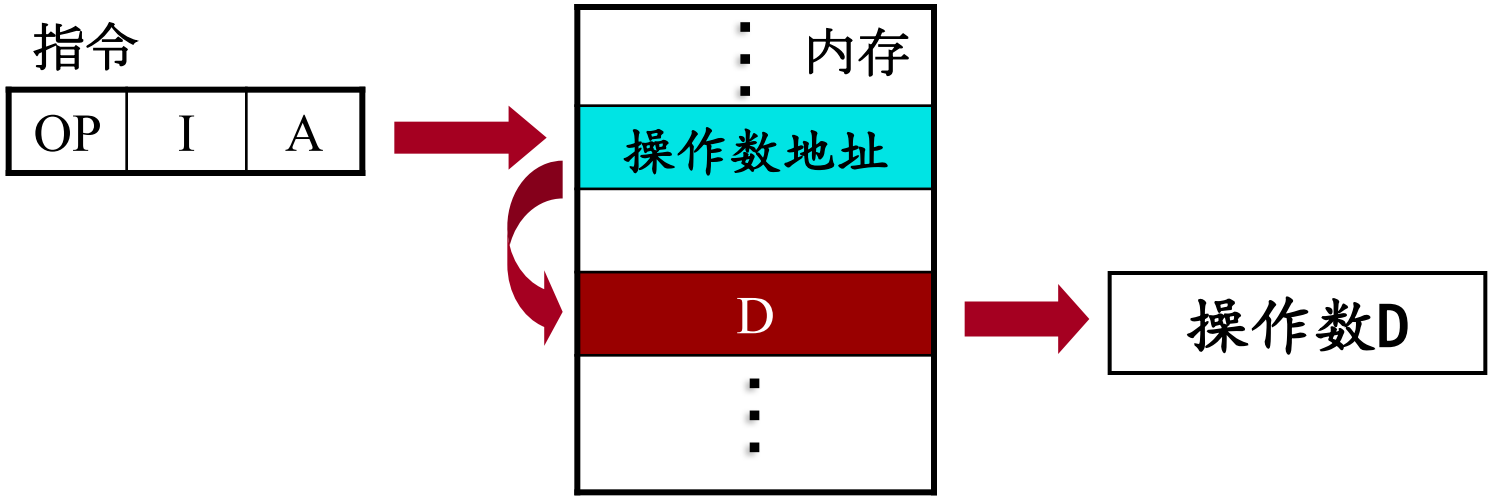
- 在指令的地址字段中直接指出操作数在内存的地址A
- 用直接寻址方式时，指令字中的形式地址A就是操作数的有效地址EA。因此通常把形式地址A又称为直接地址
- 如果用D表示操作数，那么直接寻址的逻辑表达式为

D＝（A）



4. 间接寻址（1）

- 在指令的地址字段中的形式地址A不是操作数的真正地址，而是操作数地址的地址
- 间接寻址方式是早期计算机中经常采用的方式，但由于两次访存，执行速度慢，现在已不太使用



间接寻址（2）

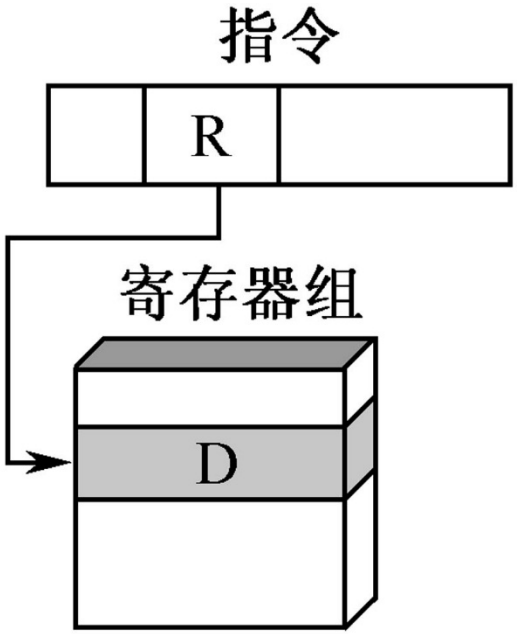
- 把直接寻址和间接寻址结合起来，指令格式如下：

操作码	I	A

- 寻址特征位I＝0，表示直接寻址，有效地址EA＝A
- 寻址特征位I＝1，表示间接寻址，有效地址EA＝(A)

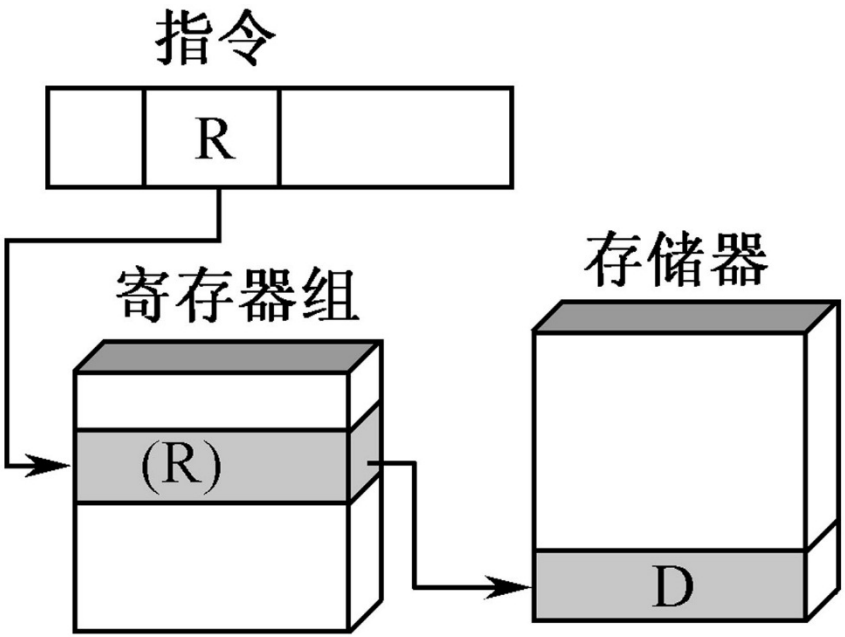
5. 寄存器寻址

- 寄存器寻址方式
  - 在指令的地址字段中给出的操作数地址不是内存的地址，而是CPU中通用寄存器的编号



6. 寄存器间接寻址

- 寄存器间接寻址方式
  - 在指令的地址字段中指定的寄存器的内容不是操作数，而是操作数的地址



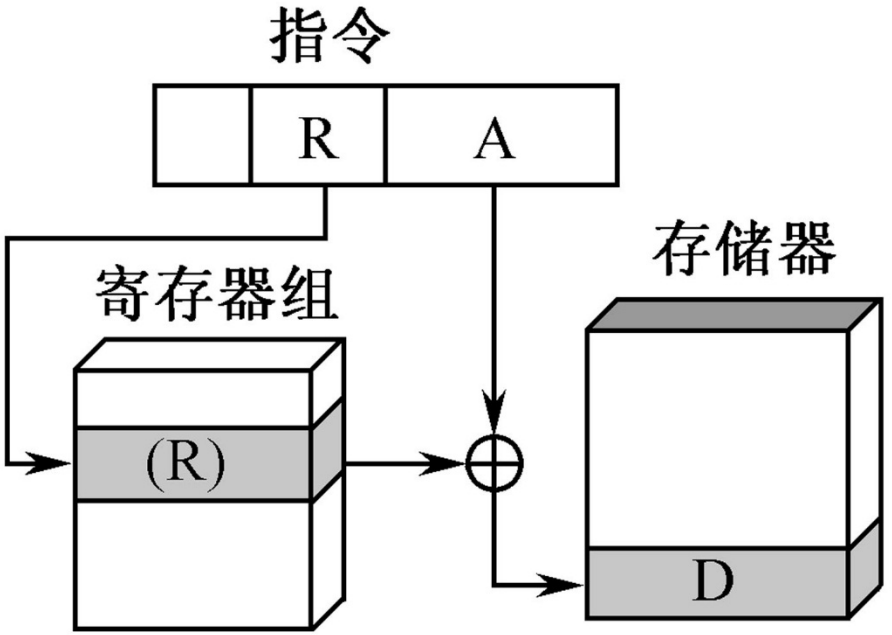
7. 偏移寻址

是直接寻址和寄存器间接寻址方式的结合

$$EA = (R) + A$$

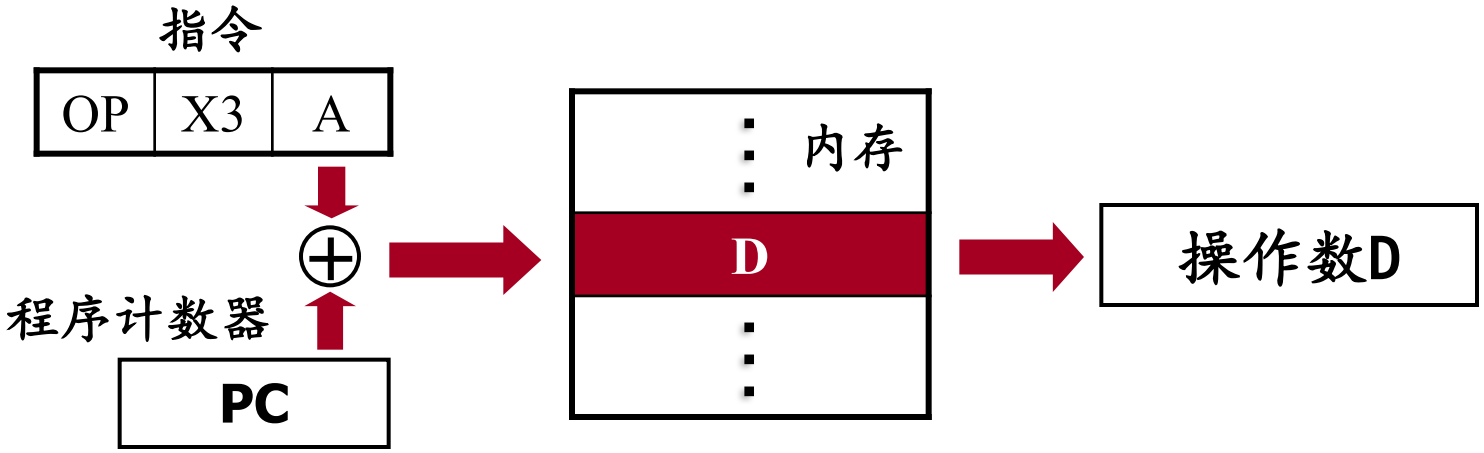
其中：R代表寄存器，A为偏移量（形式地址），是一个有符号数

- 相对寻址
- 基址寻址
- 变址寻址



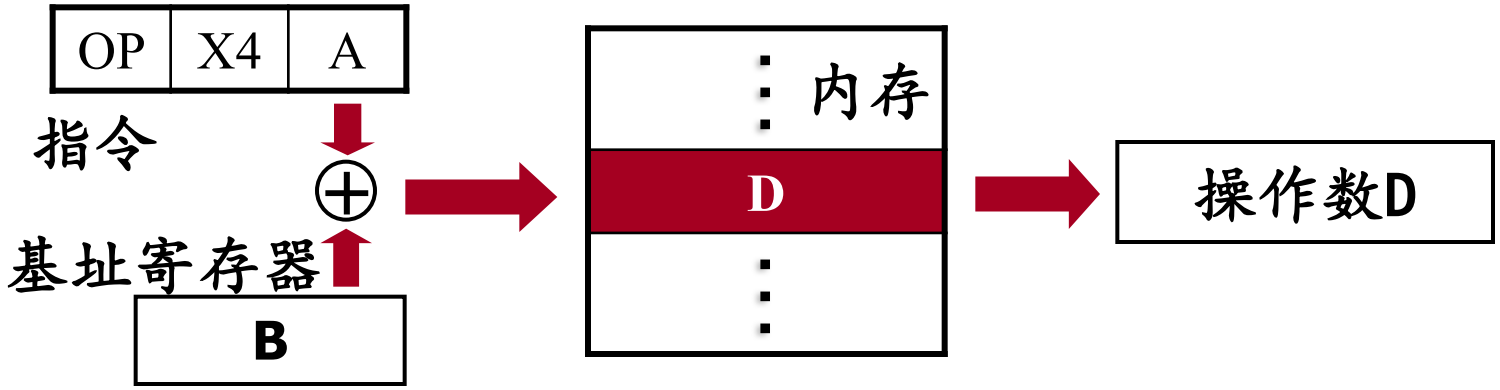
相对寻址

- 把程序计数器PC的内容加上指令格式中的偏移量A（有符号数）而形成操作数的有效地址
- 程序计数器的内容就是当前指令的地址，“相对”寻址，就是相对于当前的指令地址而言
- 好处是程序员无须用指令的绝对地址编程，所编程序可以放在内存任何地方



基址寻址

- 将CPU中基址寄存器的内容加上指令格式中的偏移量A得到的操作数有效地址
- 优点是可以扩大寻址能力。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



变址寻址

- 将CPU中变址寄存器的内容加上指令格式中的偏移量A得到的操作数有效地址。与基址寻址方式类似。但变址寄存器的内容自动递增或递减。

  EA＝（R）＋A

  R ←（R±1或2）

- 使用变址寻址方式的目的不在于扩大寻址空间，而在于实现程序块的规律性变化（自动加1减1、或加2减2）

例

某机器字长16位，主存按字节编址，转移指令采用相对寻址，由两个字节组成，第一字节为操作码字段，第二字节为相对位移量字段。假定取指令时，每取一个字节PC自动加1。若某转移指令所在主存地址为2000H，相对位移量字段的内容为06H，则该转移指令成功转以后的目标地址是

A.2006H

B.2007H

C.2008H

D.2009H

例

某计算机有16个通用寄存器，采用32位定长指令字，操作码字段（含寻址方式位）为8位，Store指令的源操作数和目的操作数分别采用寄存器直接寻址和基址寻址方式，若基址寄存器可使用任一通用寄存器，且偏移量用补码表示，则Store指令中偏移量的取值范围是

A -32768~+32767

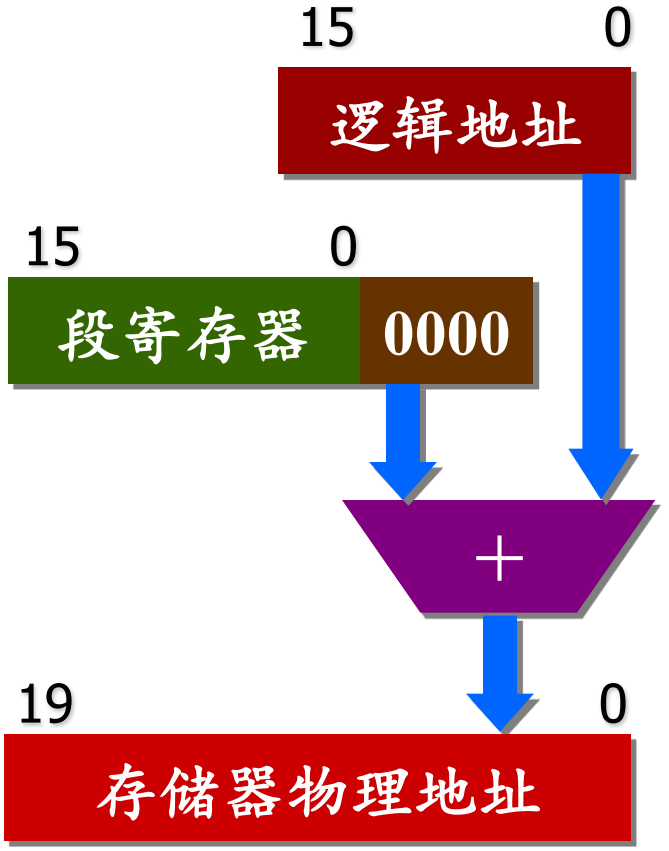
B -32767~+32768

C -65536~+65535

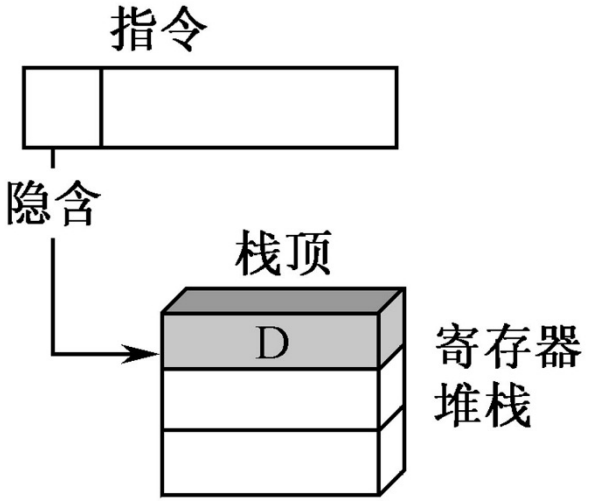
D -65535~+65536

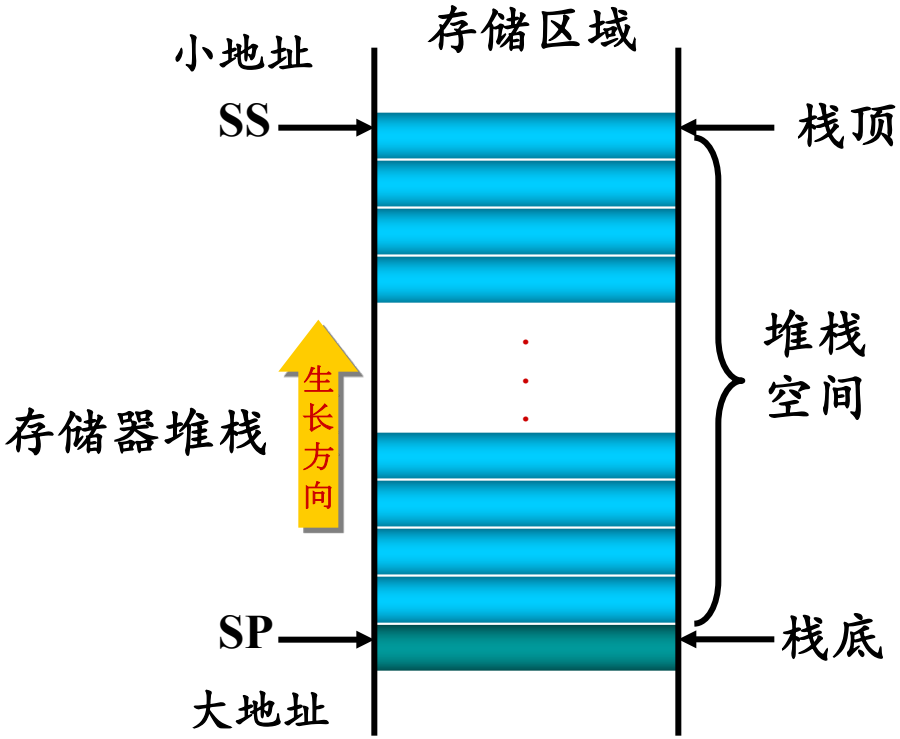
8. 段寻址方式

- 某微机有20位的地址，可直接寻址1M字节的存储空间
- 将整个1M的存储器以64K为单位划分成若干段
- 寻址时，段寄存器中的16位数会自动左移4位，然后和16位偏移量（逻辑地址）相加，即可形成所需的20位物理地址



9. 堆栈寻址





- 堆栈是一块能存储数据的存储区域，且数据的存取一般只能通过栈顶进行，其特点是“后进先出”，即：LIFO

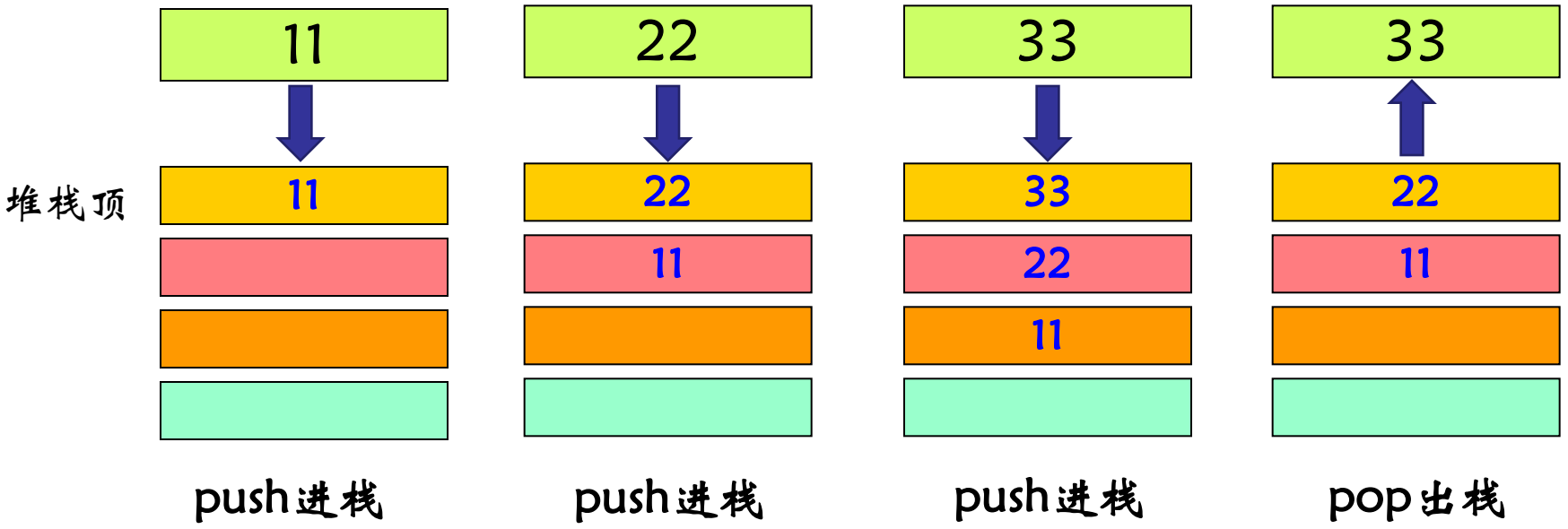
堆栈类型

- 寄存器堆栈
  - 存储区域用若干个寄存器组成
- 存储器堆栈
  - 存储区域是主存的一部分区域

寄存器堆栈

- CPU中有一组专门的寄存器，有16个或更多，它们称为串联堆栈，其中每一个寄存器能保存一个字的数据
- 数据的进出是通过栈顶实现的
- CPU通过“进栈”指令把数据送入堆栈，而通过“出栈”指令把数据从堆栈中取出
- 特点
  - 入栈和出栈时，栈顶不变，数据移动
  - 访问速度快
  - 缺点：
    - 寄存器的数目有限，所以堆栈大小受限
    - 数据的读出是破坏性的

寄存器堆栈 进栈与出栈


堆栈顶
11
11
push进栈
22
22
11
push进栈
33
33
22
11
push进栈
33
22
11
pop出栈

存储器堆栈

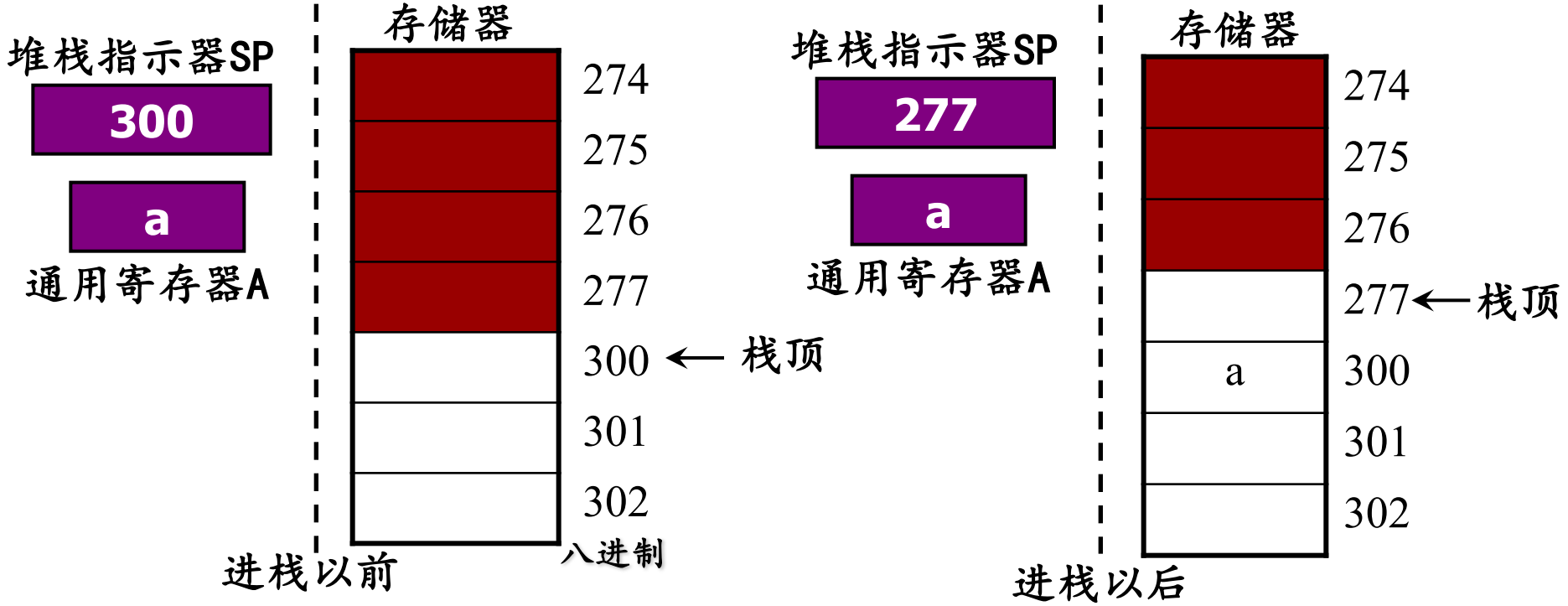
- 需要一个堆栈指示器来指示堆栈中栈顶的位置，它通常是CPU中一个专用的寄存器（SP：Stack Point）
- 比串联堆栈灵活
- 栈顶变化，由SP指示，而数据不动
- 优点：
  - 堆栈可以根据需要，任何长度
  - 堆栈个数可以根据需要而定
  - 可以用对存储器寻址的任何一条指令来对堆栈进行寻址

进栈操作

进栈操作：

（A）→Msp，（SP）－1→SP

其中（A）表示通用寄存器A的内容，SP表示堆栈指示器，Msp 表示SP所指示的堆栈栈顶单元

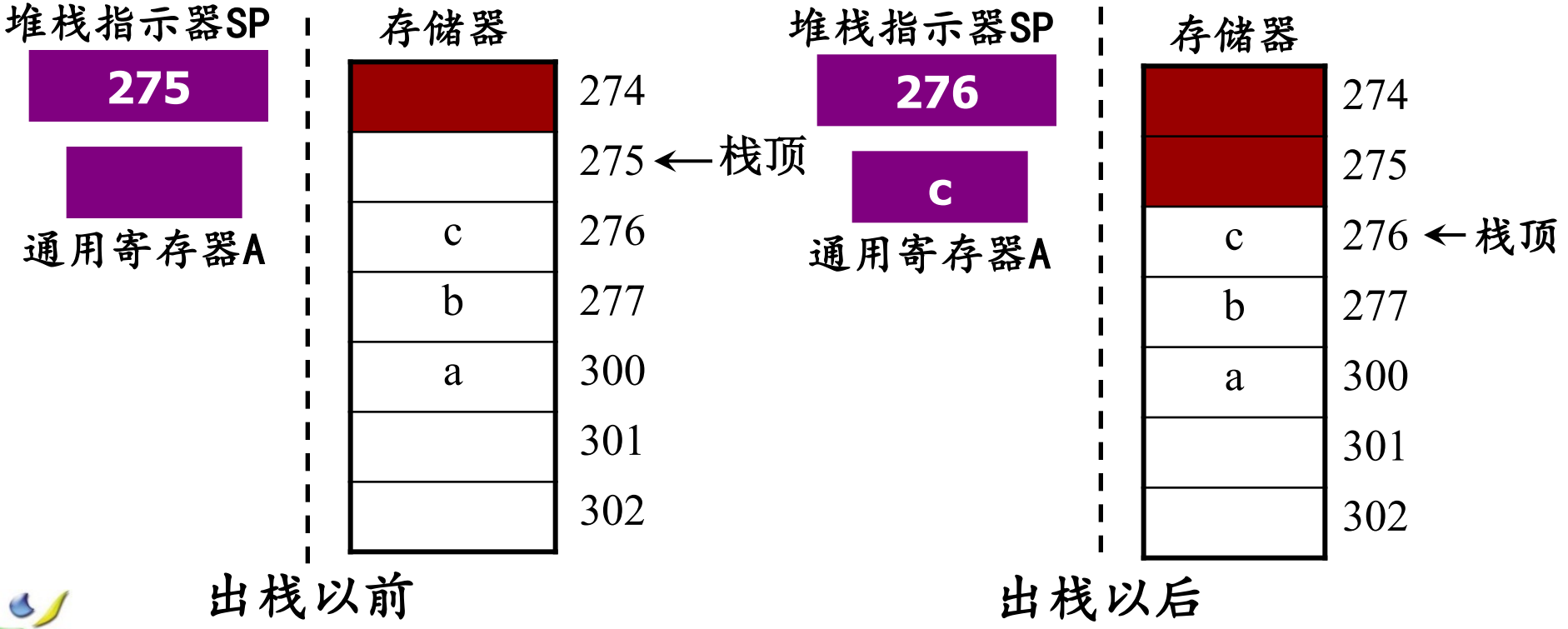


出栈操作

出栈操作：

（SP）＋1→SP，（Msp）→A

其中（A）表示通用寄存器A的内容，SP表示堆栈指示器，Msp 表示SP所指示的堆栈栈顶单元



说明

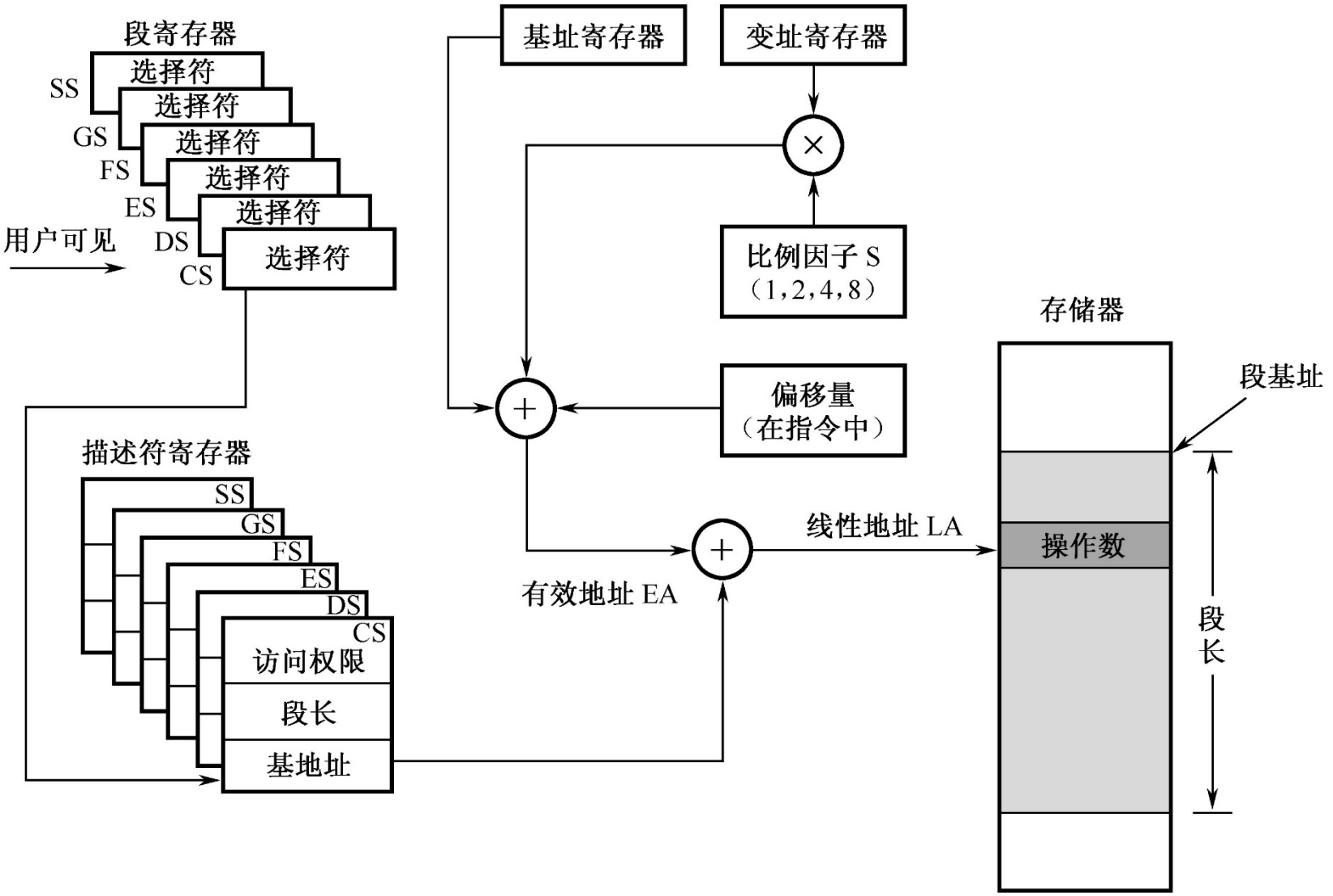
在设计处理器的堆栈操作时，有2种选择：

1. 堆栈指示器 SP始终指向空单元
   - 进栈时，先存入数据，后修改堆栈指示器SP
   - 出栈时，先修改堆栈指示器SP，然后取出数据
   - 这里修改SP可以是加操作，也可以是减操作
2. 堆栈指示器 SP始终指向满单元
   - 进栈时，？自己思考
   - 出栈时，？自己思考

Pentium的寻址方式

- Pentium的外部地址总线宽度是36位，也支持32位物理地址空间
- 实地址模式下，逻辑地址形式为段寻址方式：将段寄存器内容（16位）左移4位，低4位补全0，得到20位段基地址，再加上16位段内偏移，即得20位物理地址
- 在32为操作系统的保护模式下，32位段基地址加上段内偏移得到32位线性地址，由存储管理部件将其转换成32位的物理地址
- Pentium提供了9种有效地址的获取方式
- 各种寻址方式得到的有效地址EA，都只是段内偏移，再加上段基地址后才得到物理地址

Pentium各种寻址方式



9种寻址方式

（1）立即寻址　　操作数在指令中

（2）寄存器寻址　　操作数在R中

（3）直接寻址　　E=Disp

（4）基址寻址　　E=(B)

（5）基址/变址＋偏移量寻址　　E=(B or I)+Disp

（6）基址＋变址＋偏移量寻址　　E=(B)+(I)+Disp

（7）比例变址＋偏移量寻址　　E=(I)×S+Disp

（8）基址＋比例变址＋偏移量寻址　E=(B)+(I)×S+Disp

（9）相对寻址　　E=(PC)+Disp

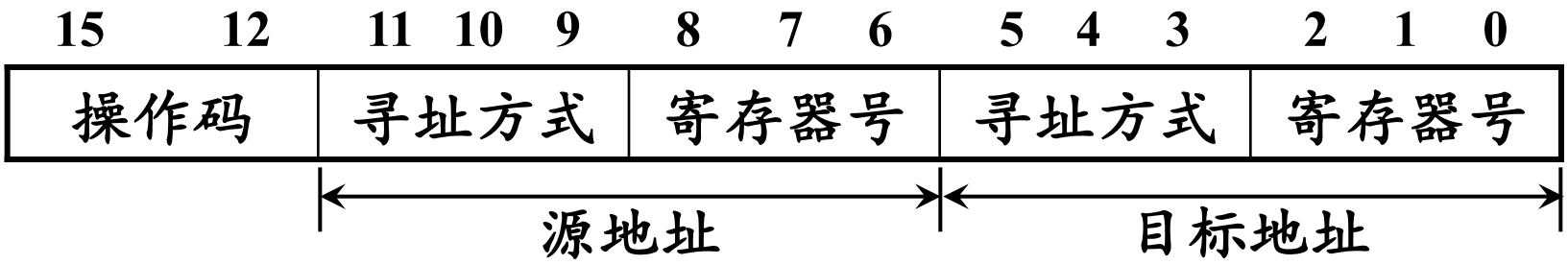
举例：PDP/11系列机寻址方式（1）

- PDP/11系列机指令字长16位
- 扩展操作码指令格式
- 操作数字段均为6位

指令位 / 指令类型	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
单操作数指令	操作码（10位）										目标地址(6位)					
双操作数指令	操作码(4位)				源地址(6位)						目标地址(6位)					
转移指令	操作码（8位）								位移量（8位）							
转子指令	操作码（7位）							寄存器号								
子程序返回指令	操作码（13位）															
条件码操作指令	操作码（11位）											S	N	Z	V	C

举例：PDP/11系列机寻址方式（2）

- 寻址方式
  - 寻址方式指示8种寻址方式，分为直接寻址和间接寻址两大类，还有4种PC寻址方式，是一种隐含寻址方式，以访问R7为标志
- 寄存器号
  - CPU内有8个寄存器：R0~R7，其中R7为程序计数器，R6为堆栈指示器，指令中用3位表示寄存器编号，形式地址存放在这些寄存器中



例3

一种二地址RS型指令的结构如下所示：

6位		4位	1位	2位	16位
OP	—	通用寄存器	I	X	偏移量D

其中I为间接寻址标志位，X为寻址模式字段，D位偏移量字段，通过I，X，D的组合，可构成下表所示的寻址方式， 请写出六种寻址方式的名称。

例3解

- 解：

	I	X	有效地址E算法	说明	寻址方式
(1)	0	00	E=D		直接寻址
(2)	0	01	E=(PC)+D	PC为程序计数器	相对寻址
(3)	0	10	E=(R2)+D	R2为变址寄存器	变址寻址
(4)	1	11	E=(R3)		寄存器间接寻址
(5)	1	00	E=(D)		间接寻址
(6)	0	11	E=(R1)+D	R1为基址寄存器	基址寻址

例4（1）

某16位机器所使用的指令格式和寻址方式如图所示，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）＝$(1B)_H$，LAD（OP）＝$(3C)_H$。MOV是传送指令，STO为写数指令，LAD为读数指令。

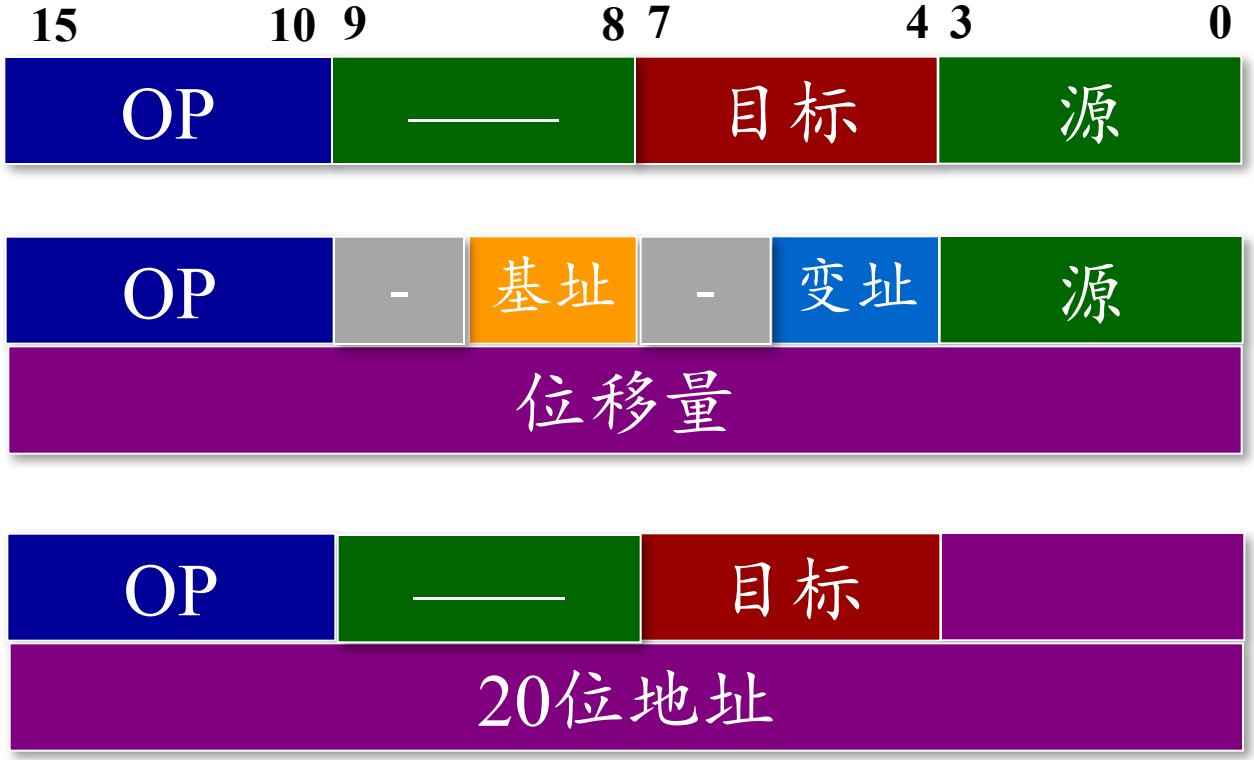
要求：

（1）分析三种指令的指令格式与寻址方式

（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？

例4（2）



MOV S, D

STO S, M

LAD M, D

解（1）指令1 单字长二地址指令 RR型

指令2 双字长二地址指令 RS型

M的寻址方式为：基址+变址+位移量

指令3 双字长二地址指令 RS型 M为直接寻址

例4（3）

（2）指令1执行速度最快

指令2执行速度最慢，因为访存地址是经过计算才能得到

指令3介于前两者之间，比指令2快

（3）①F0F1 3CD2

1111 0000 1111 0001 0011 1100 1101 0010

3C　　15号寄存器　　13CD2

把主存13CD2地址单元的内容装载（LDA）到15号寄存器

动手做

某指令格式如下所示。

其中M为寻址方式，I为变址寄存器编号，D为形式地址。若采用先变址后间址的寻址方式，则操作数的有效地址是

A I+D

B (I)+D

C ((I)+D)

D ((I))+D

动手做

- 某计算机按字节编址，指令字长固定且只有两种指令格式，其中三地址指令29条，二地址指令107条，每个地址字段为6位，问指令字长至少应该是多少位？

- 解：6*3+5=23 → 指令字长至少为24位，因为按字节编址

4.5 典型指令

指令的分类

- 数据传送指令
- 算术运算指令
- 逻辑运算指令
- 程序控制指令
- 输入输出指令
- 串处理指令
- 特权指令
- 其他指令

数据传送指令

- 完成主存和寄存器之间，或寄存器和寄存器之间的数据传送

主要包括：

- 一般传送指令　**MOV　AX，BX**
- 数据交换指令　**XCHG**
- 堆栈操作指令　**PUSH/POP**

算术运算指令

- 完成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

主要包括：

- 定点/浮点的加、减、乘、除指令 ADD、SUB…
- 加1、减1指令　INC　DEC
- 求反、求补指令 NOT、NEG
- 算术移位指令、算术比较指令 SAL CMP
- 十进制加、减运算指令

逻辑运算指令

- 完成无符号数的位操作、代码的转换、判断及运算

主要包括：

- 逻辑或指令 OR
- 逻辑与指令 AND
- 逻辑异或（按位加）指令 XOR
- 逻辑移位指令 SHR

程序控制指令

- 程序控制类指令用于控制程序的执行方向，并使程序具有测试、分析与判断的能力。也称为转移指令

主要包括：

- 条件转移指令 JNZ、JC
- 无条件转移指令 JMP
- 转子程序指令 CALL
- 返回主程序指令 RET
- 中断返回指令IRET
- …… ……

输入输出指令

- 完成CPU与外设之间的数据传送

主要包括：

- 输入指令 IN
- 输出指令 OUT

注：有的处理器没有专门的输入输出指令

串处理指令

- 对内存中连续存放的字节或字序列进行处理的指令

主要包括：

- 串传送指令
- 串存储指令
- 串加载指令
- 串比较指令
- 串扫描指令

特权指令

- 特权指令是指具有特殊权限的指令。一般不直接供给用户程序使用，是由系统程序使用的指令。
- 这类指令主要完成：
  - 系统资源的分配和管理
  - 系统工作方式的设置
  - 用户访问权限的检测
  - 段表、页表的修改
  - 任务的创建和切换
  - …

其他指令

主要包括：

- 状态寄存器置位/复位指令 STC、CLC…
- 测试指令 TEST
- 停机指令 HLT
- 空操作指令 NOP
- 系统控制用的特殊指令 WAIT…

基本指令系统（1）

- 复杂指令系统计算机（CISC）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
  - 例如Pentium有191条指令，9种寻址方式
- 最常使用的是一些最简单最基本的指令，仅占指令总数的20%，但在程序中出现的频率却占80%

基本指令系统（2）

- 教材第122页表4.9
  - 几乎所有计算机的指令集中都能找到这些基本指令

复杂指令系统

CISC指令系统的特点：

- 指令系统庞大、指令条数200~300条
- 指令长度不固定，指令格式种类多，寻址方式种类多
- 可以访问存储器的指令不受限制
- 各种指令的使用频度相差较大
- 各种指令的执行时间相差很大，不利于流水线处理
- 控制器一般采用微程序控制方式
- 难以用优化编译生成高效的目标代码

RISC计算机的出现

- RISC思想首先由IBM公司提出，IBM801处理器是公认的体现RISC思想的机器，但那时不叫RISC
- 1980/1年伯克利大学的Patterson教授和斯坦福大学的Hennessy教授各自领导的研究组先后提出了RISC概念，并分别设计了RISC I和MIPS处理器芯片
- 推出商业芯片
  - SPARC处理器　SUN公司
  - MIPS处理器　MIPS公司


BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS
北京邮电大学

精简指令系统

RISC指令系统的特点：

- 选取使用频率最高的一些简单指令，指令条数少
- 指令长度固定，指令格式种类少，寻址方式种类少
- 只有取数／存数指令访问存储器，其余指令的操作都在寄存器之间进行
- CPU中通用寄存器数量较多，减少访存次数
- 重叠窗口技术，减少过程调用中保护现场和回复现场时间
- 大部分指令都能在一个机器周期内完成，便于采用流水线技术，
- 采用硬布线控制逻辑，控制器简单，留出更多芯片面积便于实现大容量Cache
- 注重编译的优化

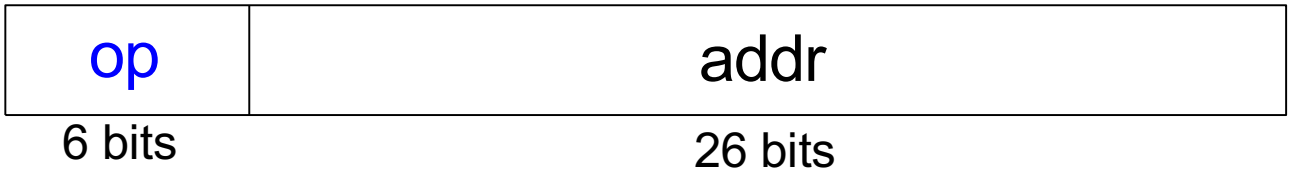
实例 MIPS Instruction Formats

- 32-bit instructions
- 3 instruction formats:

R-Type

I-Type

J-Type



R-Type

- Register-type
- 3 register operands:
  - rs, rt: source registers
  - rd: destination register
- Other fields:
  - op: the operation code or opcode (0 for R-type instructions)
  - funct: the function with opcode, tells computer what operation to perform
  - shamt: the shift amount for shift instructions, otherwise it's 0

R-Type

R-Type Examples

Assembly Code

```
add $s0, $s1, $s2

sub $t0, $t3, $t5
```

Field Values

op	rs	rt	rd	shamt	funct
0	17	18	16	0	32
0	11	13	8	0	34
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Machine Code

op	rs	rt	rd	shamt	funct	
000000	10001	10010	10000	00000	100000	(0x02328020)
000000	01011	01101	01000	00000	100010	(0x016D4022)
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	

Note the order of registers in the assembly code:

```
add rd, rs, rt
```

I-Type

- Immediate-type
- 3 operands:
  - rs, rt: register operands
  - imm:16-bit two's complement immediate
- Other fields:
  - op: the opcode
  - Simplicity favors regularity: all instructions have opcode
  - Operation is completely determined by opcode

I-Type

I-Type Examples

Assembly Code

```
addi $s0, $s1, 5
addi $t0, $s3, -12
lw   $t2, 32($0)
sw   $s1,  4($t1)
```

Field Values

op	rs	rt	imm
8	17	16	5
8	19	8	-12
35	0	10	32
43	9	17	4
6 bits	5 bits	5 bits	16 bits

Note the differing order of registers in assembly and machine codes:

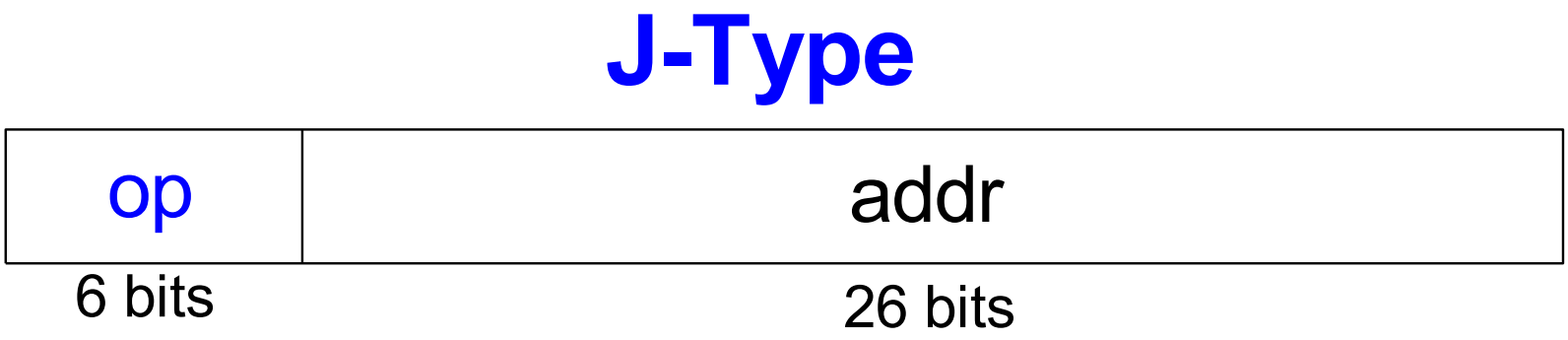
```
addi rt, rs, imm
lw   rt, imm(rs)
sw   rt, imm(rs)
```

Machine Code

op	rs	rt	imm	
001000	10001	10000	0000 0000 0000 0101	(0x22300005)
001000	10011	01000	1111 1111 1111 0100	(0x2268FFF4)
100011	00000	01010	0000 0000 0010 0000	(0x8C0A0020)
101011	01001	10001	0000 0000 0000 0100	(0xAD310004)
6 bits	5 bits	5 bits	16 bits	

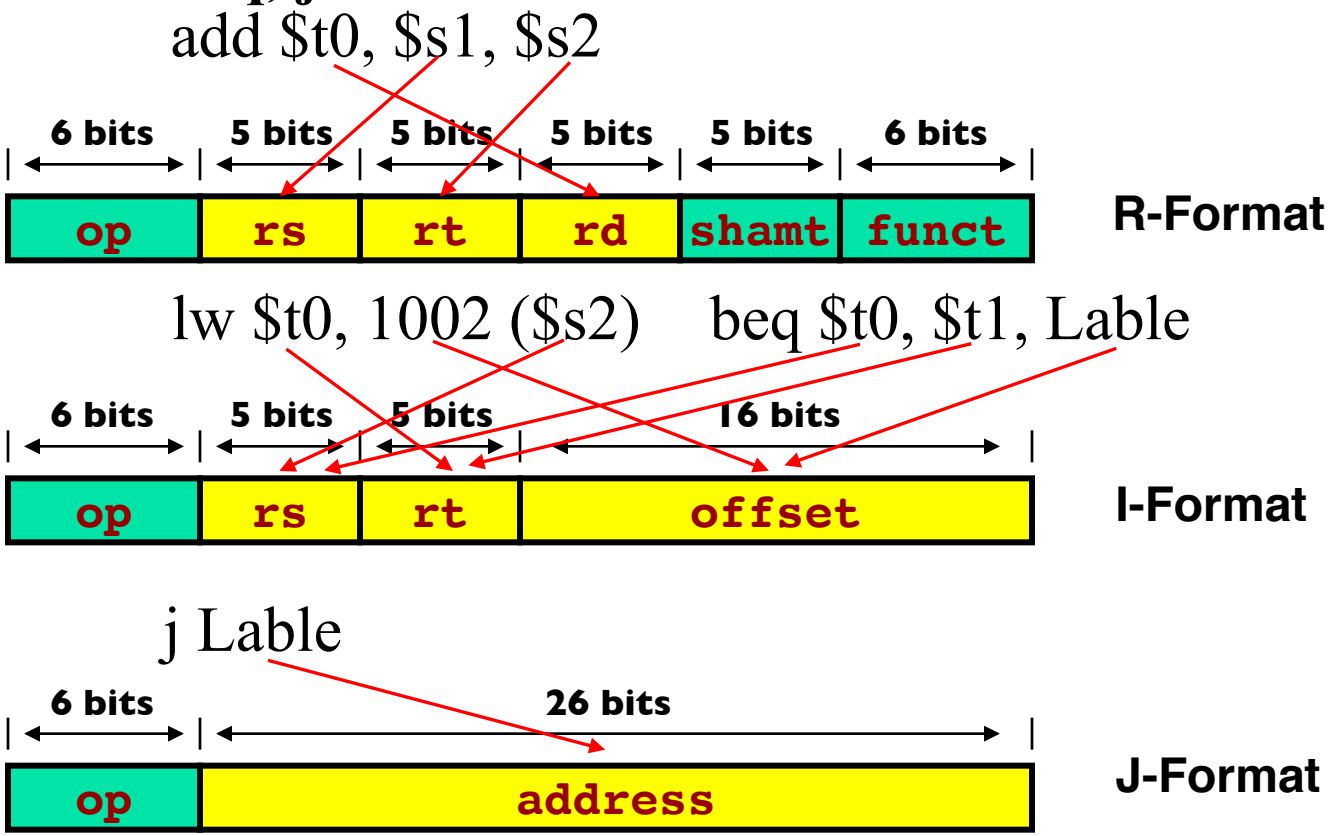
J-Type

- Jump-type
- 26-bit address operand (addr)
- Used for jump instructions (j)



MIPS指令格式

- 算术逻辑指令：add, sub, and, or, slt
- 取数/存数指令：lw, sw
- 控制流指令：beq, j



Powre PC指令类型

- 整数算术运算 / 逻辑运算 / 移位指令
- 浮点运算指令
- 取数（LOAD） / 存数（STORE）指令
- 条件寄存器指令
- 控制转移指令

指令等长，为32位

指令格式

教材第123页图4.5