第 2.3 节

寻址技术

2.3、寻址技术

- 知识点
  - 形式地址、有效地址
  - 基本寻址技术
  - 复合寻址方式
- 重点
  - 基本寻址技术

2.3、寻址技术

- 回顾程序的执行
  - 顺序执行

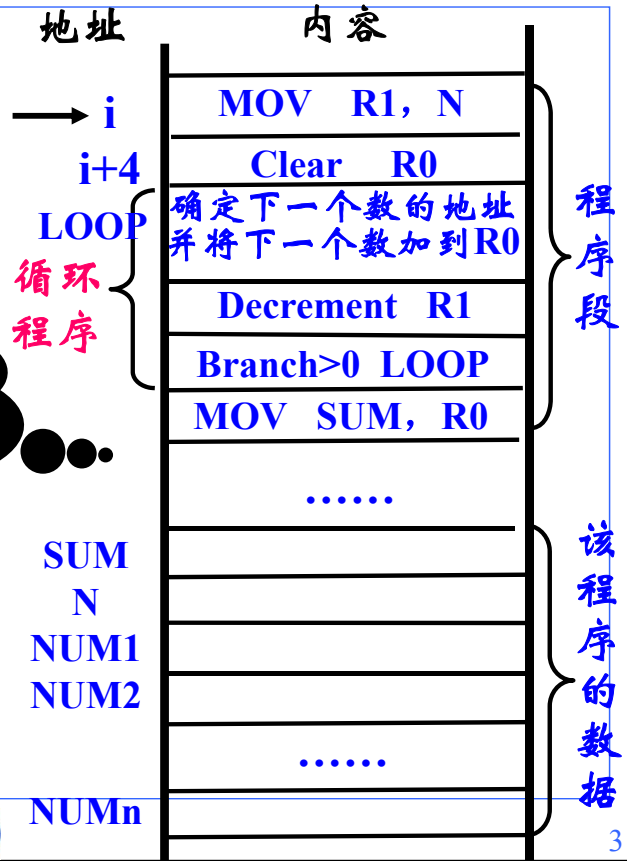
    X=A+B；

    Y=X-C；
  - 分支执行
  - 循环执行

    for(I=1 ； I=I+1； I<N)

      X(I)=A（I）+B（I）；

需要用灵活方式去指明下一条指令地址和要处理的操作数地址



2.3、寻址技术

- 寻址技术（Addressing）是确定操作数地址的技术。它是计算机设计中硬件技术对软件最早提供支持的技术。
- 技术出现的目的
  - 扩大访存区间
  - 提高访问数据的灵活性和有效性
  - 支持软件技术的发展：多道程序设计

2.3、寻址技术

形式址址（逻辑地址）

- 寻址能力
- 地址计算的复杂度

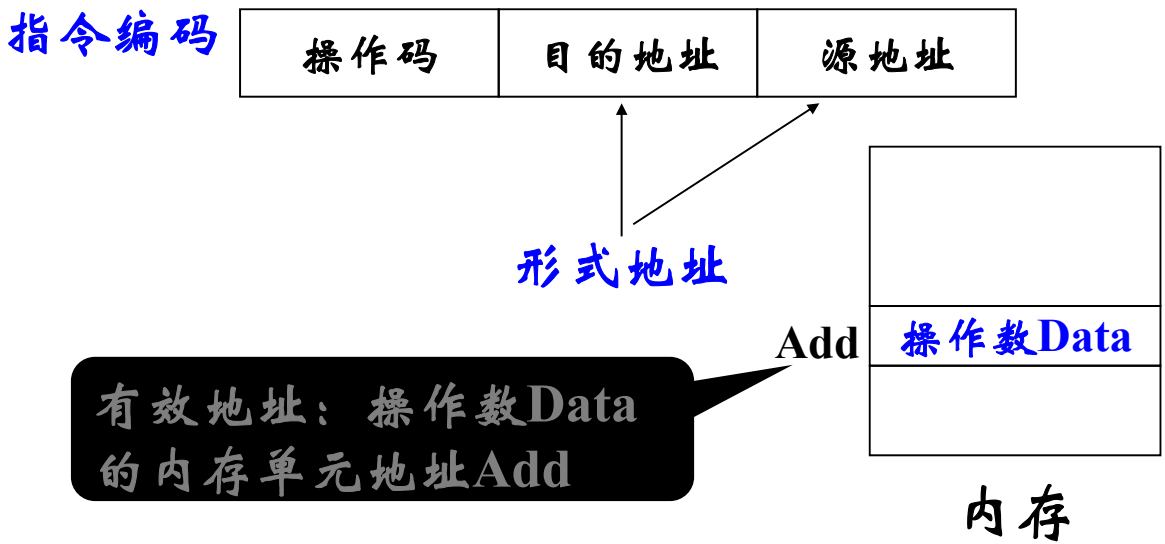
指令中直接给出的地址编码

有效地址（物理地址）

根据形式地址和寻址方式计算出来的操作数在内存单元中的地址

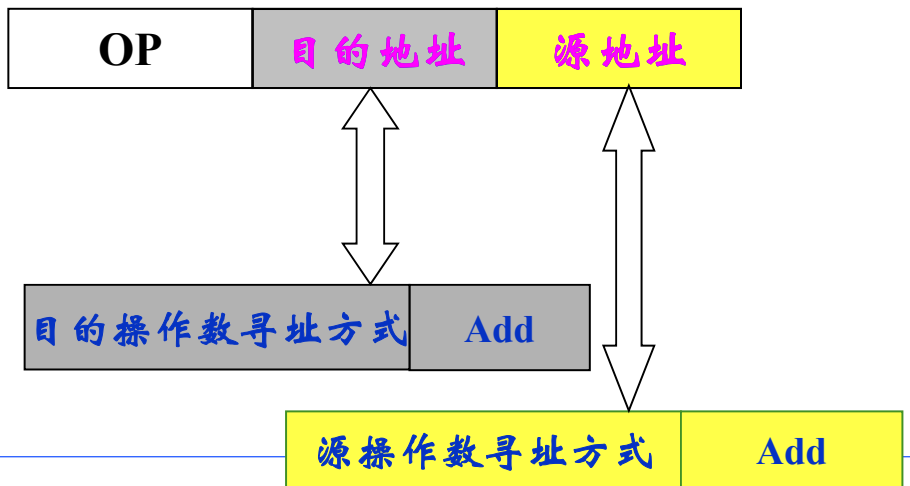
2.3、寻址技术

- 寻址：根据形式地址查找到操作数的过程



2.3、寻址技术

- 寻址方式/技术
  - 定义：指令代码中地址字段的一部分，指明操作数的获取方式或操作数地址的计算方式
  - 指令中每一个地址字段均有其寻址方式编码（或隐含寻址方式）



2.3.1、基本寻址技术

- 立即数寻址
- 直接寻址
- 间接寻址
- 寄存器寻址
- 存储器间接寻址
- 偏移寻址
- 堆栈寻址

2.3.1、基本寻址技术

- 约定
  - A—— 形式地址，指令地址字段的内容
  - EA—— 有效地址，包含有被访问操作数的实际单元
  - (X)—— 表示单元X中的内容

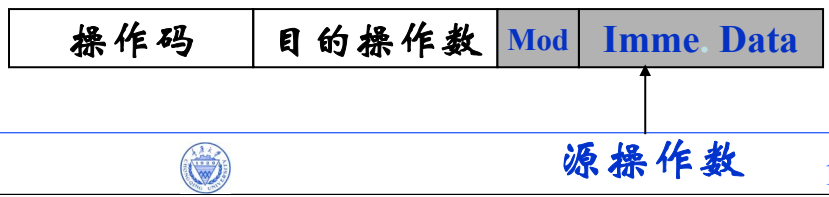
2.3.1、基本寻址技术

（1）立即数寻址

- 操作数直接在指令代码中给出
- 立即寻址只能作为双操作数指令的源操作数
- Operand = Imme. Data

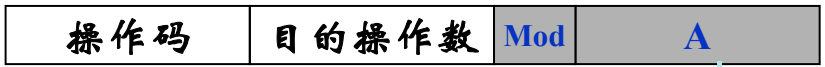
例：MOV AX，1000H

- 指令执行时间很短
- 操作数的大小受地址字段长度的限制
- 广泛使用

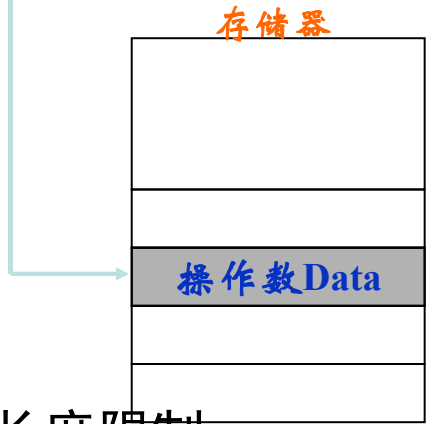


2.3.1、基本寻址技术

（2）存储器直接寻址



- 操作数在存储器中，指令地址字段直接给出操作数在存储器中的地址
- EA = A，Operand = (A)

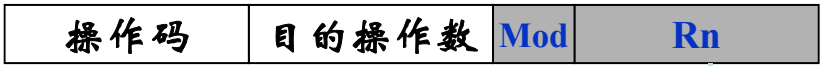


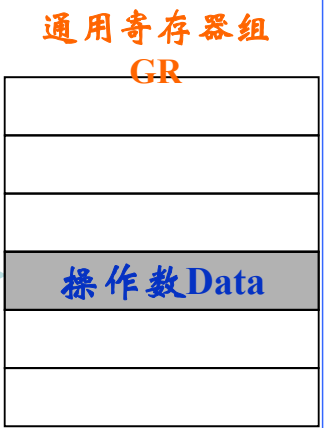
例：MOV AX，[1000H]

- 处理简单、直接
- 寻址空间受到指令的地址字段长度限制
- 较少使用，8位计算机和一些16位计算机

2.3.1、基本寻址技术

（3）寄存器直接寻址





- 操作数在寄存器中，指令地址字段给出寄存器的地址（编码）
- EA = Rn， Operand = (Rn)

例：MOV BX，AX

- 只需要很短的地址字段
- 无需访问存储器，指令执行速度最快
- 地址空间有限，可以编程使用的通用寄存器不多
- 使用最多，提高性能的常用手段

2.3.1、基本寻址技术

- 间接寻址与指针
  - 指令中没有明确给出操作数或者操作数的地址？
  - C语言程序中的指针

A=*B

MOV R1,B

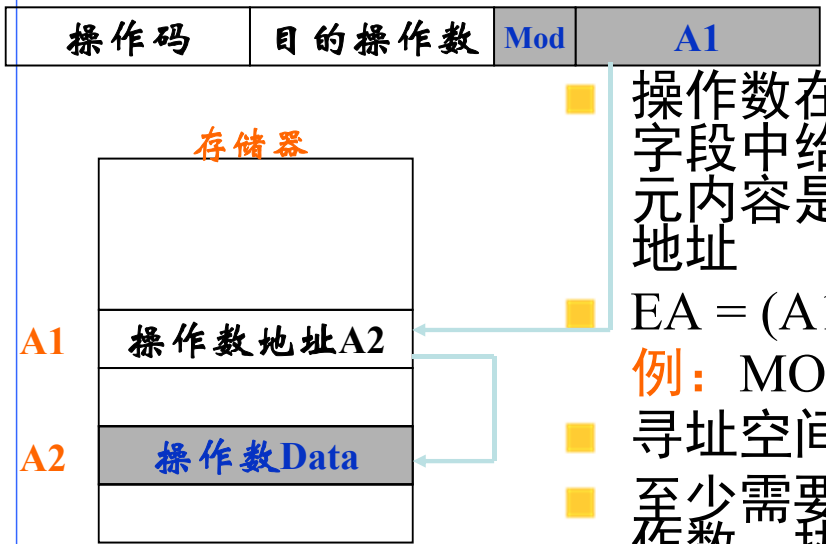
MOV A，(R1)

包含一个操作数地址的寄存器或内存单元

  - 间接寻址和指针使用在程序设计中是一个重要和强有力的概念

2.3.1、基本寻址技术

(4) 存储器间接寻址



- 操作数在存储器中，指令地址字段中给出的存储器地址的单元内容是操作数在存储器中的地址
- EA = (A1)，Operand = ((A1))

  例：MOV R1, @(1000H)
- 寻址空间大，灵活，便于编程
- 至少需要两次访存才能取到操作数，执行速度慢

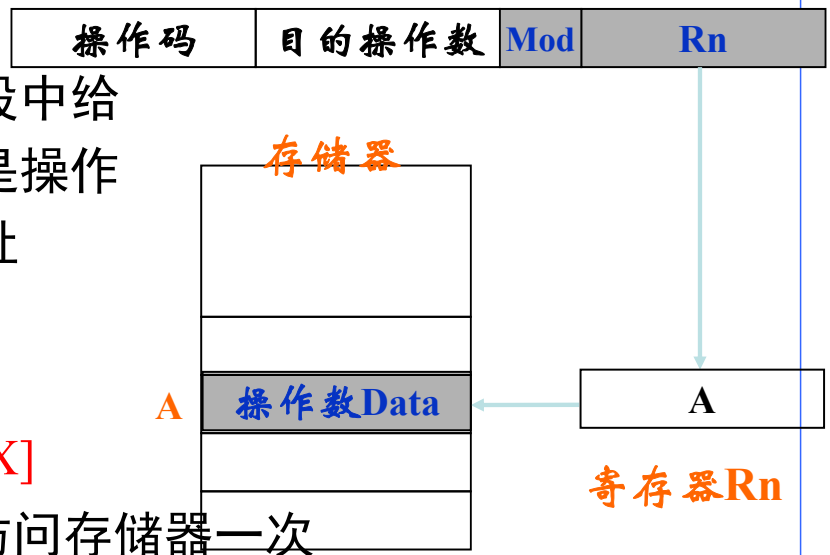
2.3.1、基本寻址技术

（5）寄存器间接寻址

- 操作数在存储器中，指令地址字段中给出的寄存器的内容是操作数在存储器中的地址
- EA = (Rn)，
  Operand = ((Rn))

例：MOV AX，[BX]

- 它比间接寻址少访问存储器一次
- 使用比较普遍



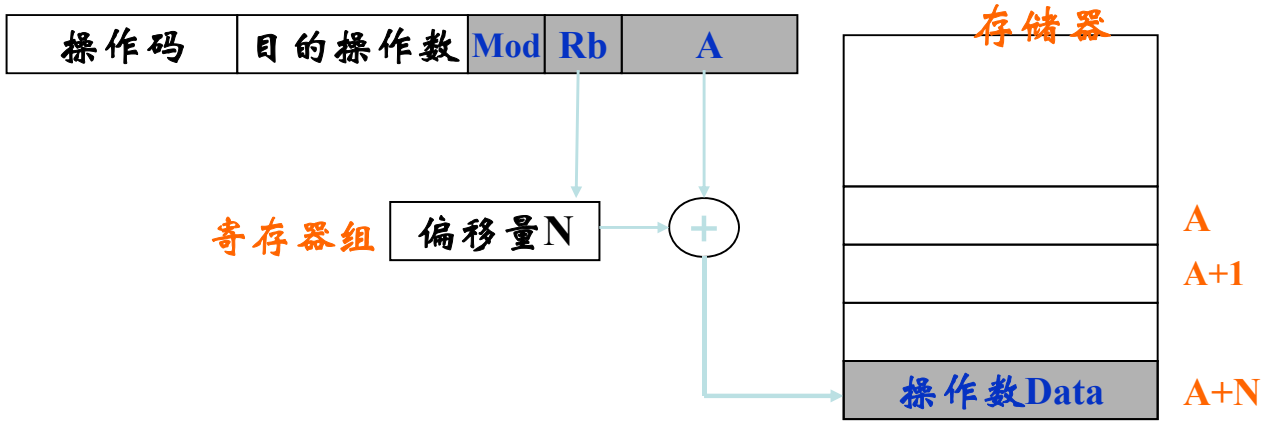
2.3.1、基本寻址技术

（6）偏移寻址

- 一种复合寻址技术，组合了直接寻址和寄存器间接寻址两种方式
- 功能强大，在指令不作任何修改的情况下，它可以通过修改寄存器的内容来实现访问单元地址的位移，因此称之为偏移寻址
- EA = （R） + A

2.3.1、基本寻址技术

（6）偏移寻址



2.3.1、基本寻址技术

（6）偏移寻址

- 根据间接寻址所使用的寄存器的不同，偏移寻址的方法很多。常见的三种偏移寻址
  - 相对寻址（Relative Addressing）
  - 基址寻址（Base-Register Addressing）
  - 变址寻址（Index Addressing）

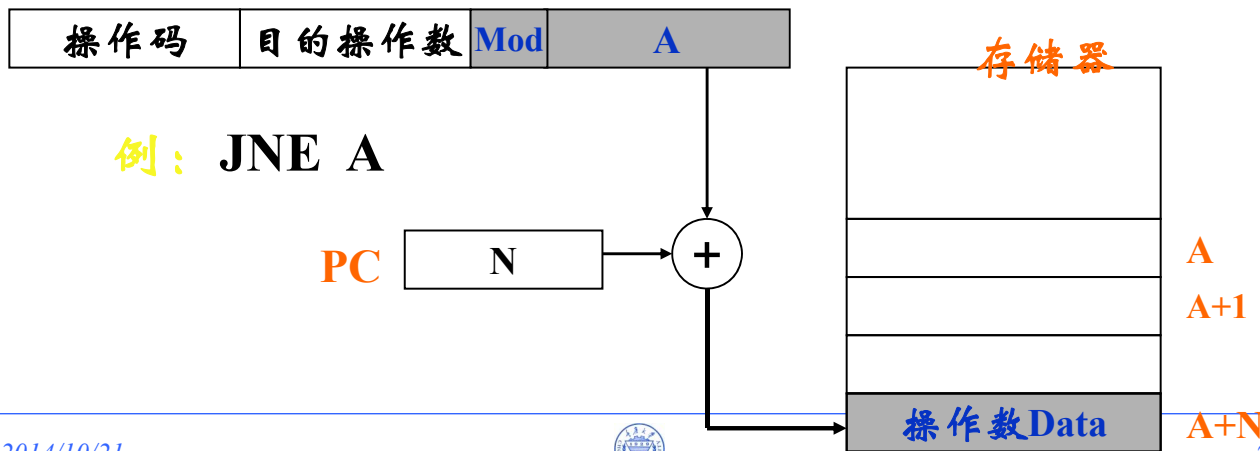
2.3.1、基本寻址技术

- 相对寻址（Relative Addressing）
  - 是把程序计数器PC作为偏移寻址中间接寻址所使用的寄存器，就是相对于当前指令地址
  - EA =（PC）+ A

2.3.1、基本寻址技术

相对寻址

- 基址寻址的特例，由程序计数器PC作为基址寄存器，指令中给出的形式地址作为偏移量，二者之和是操作数的内存地址
- EA = (PC)+A，Operand = ((PC)+A)



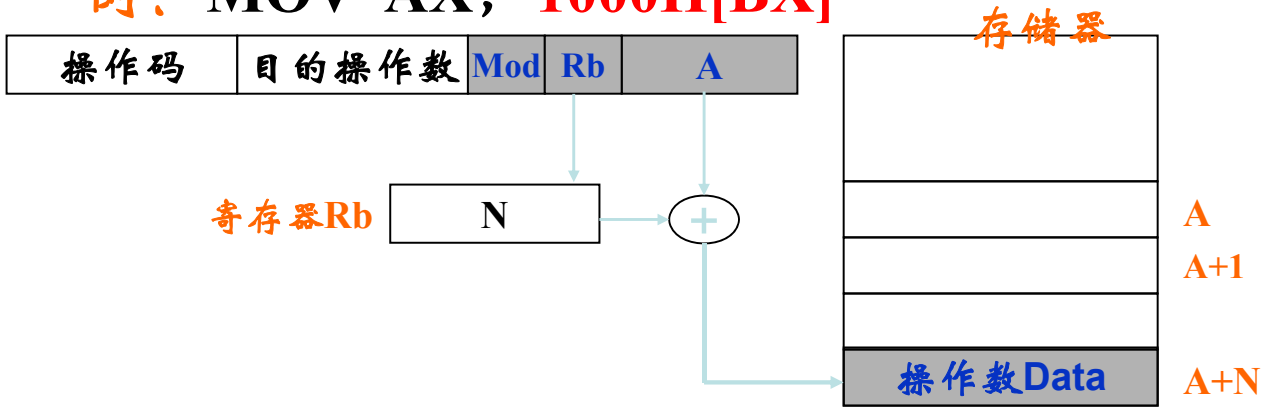
2.3.1、基本寻址技术

- 基址寻址（Base-Register Addressing）
  - 相对于基址寄存器BR的偏移寻址。基址寄存器存放的是一个存储器单元地址（基地址）
  - EA=（BR）+A
  - 采用基址寻址的程序，可以不用考虑程序加载到主存中的位置
  - 段式访存

基址寻址

- 操作数在存储器中，指令地址字段给出一基址寄存器和一形式地址，基变址寄存器的内容与形式地址之和是操作数的内存地址
- EA = (Rb)+A， Operand = ((Rb)+A)

例：MOV AX，1000H[BX]



基址寻址的作用：较短的形式地址长度可以实现较大的存储空间的寻址

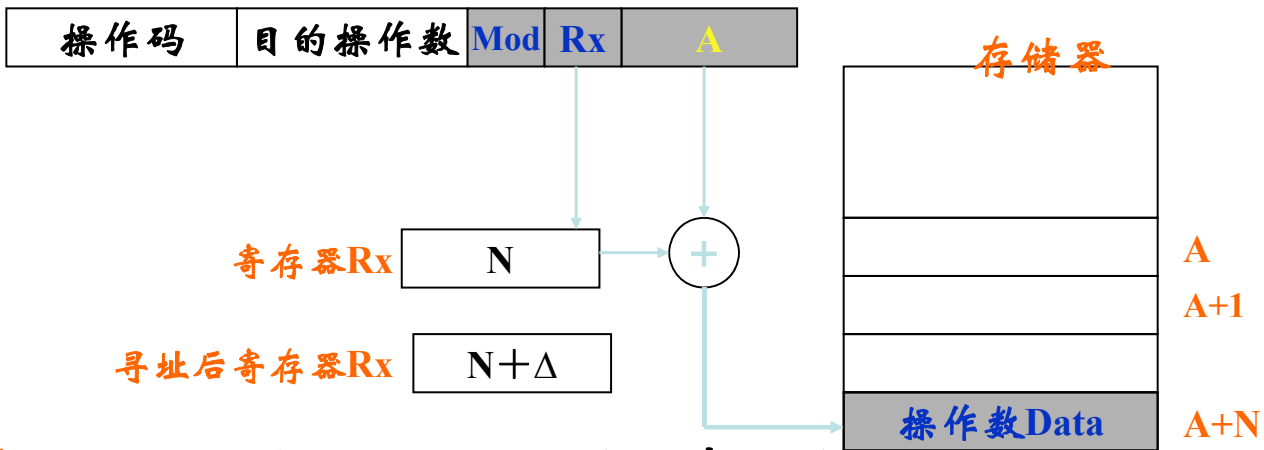
2.3.1、基本寻址技术

- 变址寻址（Index Addressing）
  - 相对于变址寄存器（Index Register，记为X）的偏移寻址
  - EA=（X）+A
  - 形式地址不变，程序可以通过变址寄存器内容的变化来访问一片主存地址
  - 自变址寻址（Autoindexing）
    - EA = A +（X）
    - X = X + 1

变址寻址

- 操作数在存储器中，指令地址字段给出一变址寄存器和一形式地址，变址寄存器的内容与形式地址之和是操作数的内存地址
- EA = (Rx)+A，Operand = ((Rx)+A)

例：MOV AX，1000H[DI]



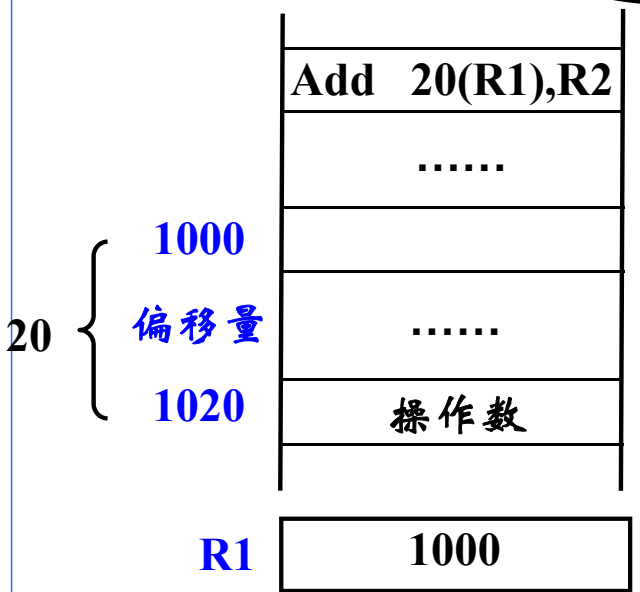
变址寻址的作用：数组操作，串操作

2.3.1、基本寻址技术

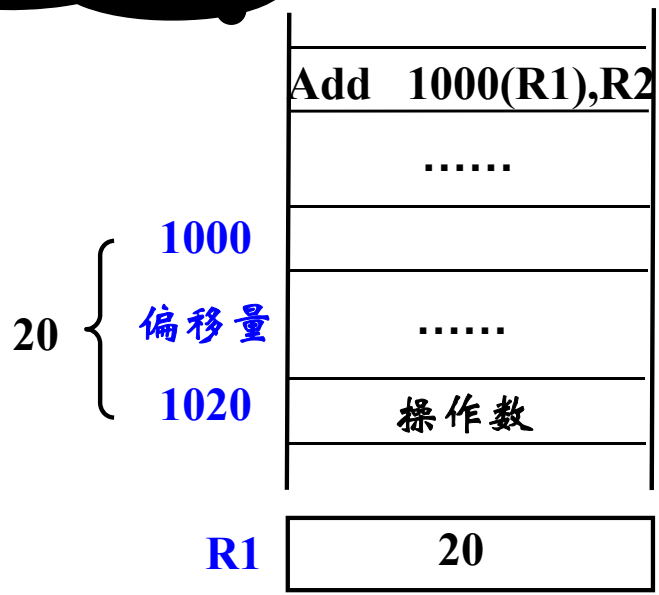
- 变址与数组

X（Ri）

两种使用变址的方法



偏移量作为一个常数给出



偏移量在变址寄存器中

2.3.1、基本寻址技术

- 基址寻址、变址寻址的比较
  - 对于一道程序，基址是不变的，程序中的所有地址都是相对于基址来变化的。对于变址寻址，形式地址给出的是一个存储器地址基准，变址寄存器X存放的是相对于基准地址的偏移量
  - 在基址寻址中，偏移量位数较短。而在变址寻址中，偏移量位数足以表示整个存储空间
  - 基址寻址立足于面向系统，主要是解决程序逻辑空间与存储器物理空间的无关性；而变址寻址立足于用户，主要是为了能编写出高效的访问一片存储空间的程序
- 软件设计的重要支持基础

2.3.1、基本寻址技术

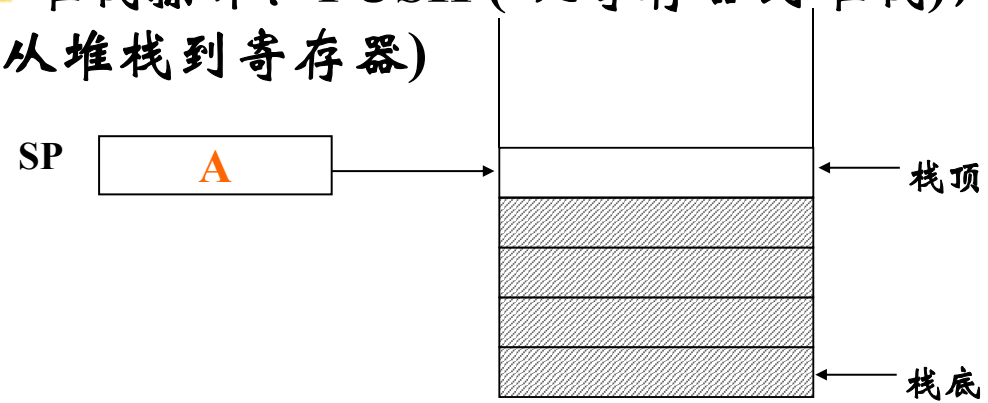
（7）堆栈寻址

- 以堆栈寻址方式工作的指令，一般都不是显示地给出操作数的地址，而是隐含着操作数的地址，这个地址就是栈顶。所以又称这种寻址为隐含寻址（Implied Addressing）

2.3.1、基本寻址技术

堆栈寻址

- 堆栈的结构：一段内存区域
- 栈底，栈顶
- 堆栈指针(SP)：是一个特殊寄存器部件，指向栈顶
- 堆栈操作：PUSH(从寄存器到堆栈)，POP(从堆栈到寄存器)

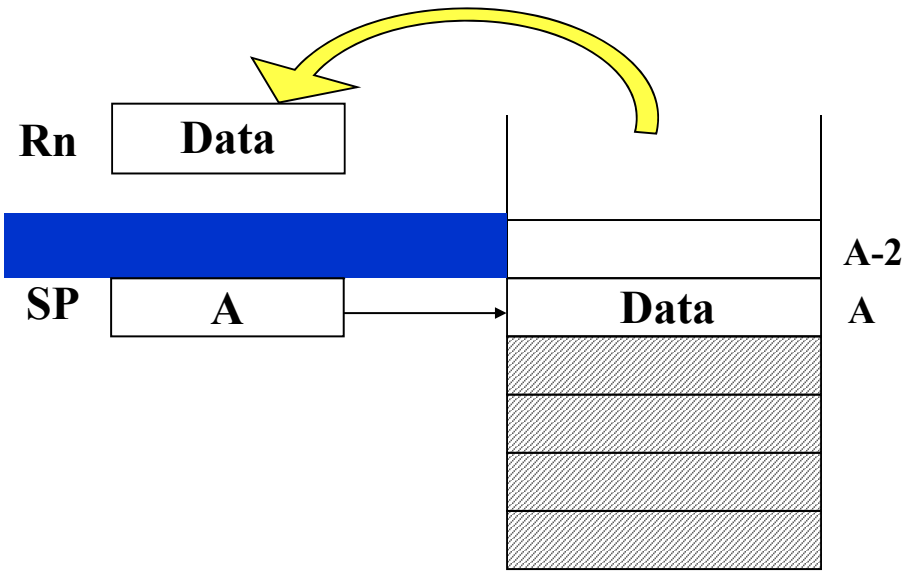


2.3.1、基本寻址技术

堆栈寻址

- 出栈操作：POP Rn

  SP ← (SP) + 2， Rn ← ((SP))



2.3.1、基本寻址技术

（8）基本寻址方式的小结

表2- 6　　基本寻址方式的比较

寻址方式	规则	主要优点	主要缺点
立即数寻址	操作数＝A	无需访存储器	操作数受限
存储器直接	EA＝A	简单	寻址空间受限
存储器间址	EA＝(A)	寻址空间大	多次访问主存
寄存器直接	EA＝R	无需访存储器	寻址空间受限
寄存器间址	EA＝(R)	寻址空间大	多访主存一次
偏移寻址	EA＝(R)＋A	灵活	复杂
堆栈寻址	EA＝(SP)	缩短指令字长	应用范围有限

2.3.2、复合寻址方式

- 复合寻址方式是把间接寻址方式同相对寻址方式或变址寻址方式相结合而形成的寻址方式。它分为先间接方式与后间接方式两种
  - 变址间接式　EA＝（（
  - 间接变址式　EA＝（X）＋（A）
  - 相对间接式　EA＝（（PC）＋A）
  - 间接相对式　EA＝（PC）＋（A）

在寻址的灵活性和硬件的复杂性之间权衡

2.3、寻址技术

- 面向不同对象的寻址方式
  - 面向存储器寻址主要访问存储器，少量访寄存器
  - 面向寄存器寻址主要访问寄存器，少量访问存储器。减少访存，速度快，适合于向量、矩阵运算
  - 面向堆栈寻址主要访问堆栈，少量访问寄存器和存储器。有利于减轻对高级语言编译的负担，不用考虑寄存器的优化分配和使用，有利于支持子程序嵌套、递归调用时的参数、返回地址及现场等的保存和恢复。可节省许多地址字段，节省程序空间，存储效率高，免去复杂地址计算

寻址方式举例

例 一种三地址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为偏移量字段，通过I，X，D的组合，可构成下表所示的寻址方式。请写出6种寻址方式的名称。

寻址方式	I	X	有效地址E算法	说明
（1）	0	00	E＝D	
（2）	0	01	E＝（PC）±D	PC为程序计数器
（3）	0	10	E＝（R_2）±D	R_2为变址寄存器
（4）	1	11	E＝（R_3）	
（5）	1	00	E＝（D）	
（6）	0	11	E＝（R_1）±D	R_1为基址寄存器

解：（1）直接寻址　（3）变址寻址　（5）间接寻址

（2）相对地址　（4）寄存器间接寻址　（6）基址寻址

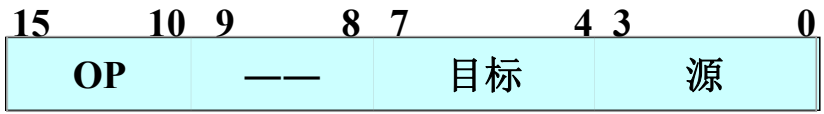
例 某16位机所使用指令格式和寻址方式如下所示。该机有两个20位基值寄存器，四个16位变址寄存器， 十六个16位通用寄存器。

指令格式中的 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)＝$(A)_H$， MOV是传送指令，

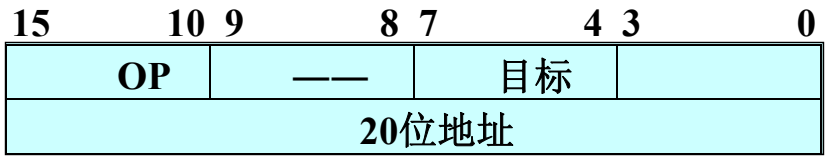
STA (OP)＝$(1B)_H$， STA为写数指令

LDA (OP)＝$(3C)_H$。 LDA为读数指令



MOV S, D

STA S, M



LDA M，D

要求：（1）分析三种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 $(1C2)_H$

解：(1) 第一种指令是单字长二地址指令，RR型；

第二种指令是双字长二地址指令，RS型，其中S采用基址寻址或变址寻址，R由源寄存器决定；

第三种也是双字长二地址指令，RS型，其中R由目标寄存器决定，S由20位地址（直接寻址）决定。

(2) 第一种指令所花时间最短，因为是RR型指令，不需要访问 存储器。

第二种指令所花时间最长，因为是RS型指令，需要访问存储器，同时要进行寻址方式的变换运算（基值或变址），这也需要时间。

第三种指令虽然也访问存储器，但节省了求有效地址运算的时间开销。

第二种指令的执行时间不会等于第三种指令的执行时间。

(3) 根据已知条件：

MOV(OP)＝001010，STA(OP)＝011011，LDA(OP)＝111100，

LDA的操作码　　目标寄存器编号

① $(F0F1)_H$ $(3CD2)_H$ = 111100, 00,1111,0001 $(3CD2)_H$

该指令代表LDA指令，编码正确，其含义是把主存$(3CD2)_H$地址单位元的内容取至15寄存器。

LDA M，D

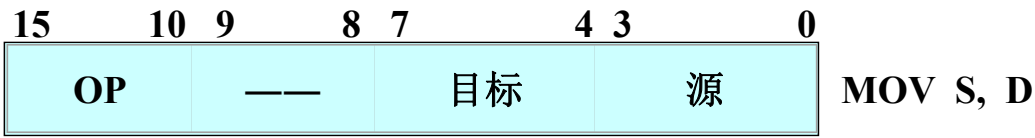
根据已知条件：

MOV(OP)＝001010，STA(OP)＝011011，LDA(OP)＝111100，

目标寄存器编号　源寄存器编号

② $(2856)_H$ = 001010, 00, 0101, 0110

代表MOV指令，编码正确，含义是把6号源寄存器的内容传送至5号目标寄存器。



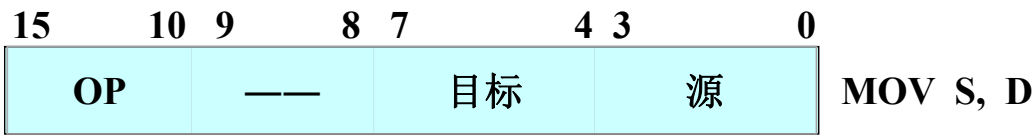
根据已知条件：MOV(OP)＝001010， STA(OP)＝011011， LDA(OP)＝111100，

③ $(6FD6)_H$ = 011011, 11, 1101, 0110

该指令是单字长指令，一定是MOV指令，但编码错误，可改正为 001010,00,1101,0110 = $(28D6)_H$

④ $(1C2)_H$ = 000000, 01, 1100, 0010

该指令是单字长指令，代表MOV指令，但编码错误，可改正为 001010, 00, 1100, 0010 = $(28C2)_H$。



敬请批评指正

谢　谢