4.6 指令系统

主讲教师：黄兰秋

指令系统

从用户的角度看，指令是用户使用与控制计算机的最小功能单位，从计算机本身的组成看，指令直接与计算机的运行性能、硬件结构密切相关，它是设计一台计算机的起始点和基本依据。通过本章的学习，要求理解指令中包含的信息，掌握常用的指令和指令格式，深入理解指令的寻址方式及用途，了解常见的指令功能和种类

教学内容：

1、指令系统的基本概念

2、指令格式

3、指令和数据的寻址方式

4、典型指令

重点：

本章的重点是指令格式，指令和数据的寻址方式

难点：

本章的难点是指令和数据的寻址方式

指令系统概述

1. 计算机的程序：是解决某一实际问题的指令序列；

2. 指令：就是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、机器指令和宏指令之分。

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。

3. 指令系统：一台计算机中所有机器指令的集合，它是表征一台计算机性能的重要因素，其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。

计算机指令系统特性

完备性：指令丰富，功能齐全，使用方便。

有效性：程序占空间小，执行速度快。

规整性：

对称性、匀齐性：

指令格式和数据格式的一 致性：

兼容性：系列机软件向上兼容

指令格式

指令的格式就是机器语言的语法；

每条指令规定机器完成一定的功能。

一台计算机的所有的指令集合称为该机的指令系统或指令集。它是程序工作者编制程序的基本依据，也是进行计算机逻辑设计的基本依据 。

一条指令应包含如下信息：

1）进行何种操作：即操作性质。体现在指令中被称为操作码。

2)操作的对象：数据来源以及如何寻找操作数。体现在指令中被称为地址码。

3)操作结果：结果存放在何处。

4)下一条指令又如何寻找?

一、机器指令的基本格式

指令由操作码和地址码两部分组成，用二进制代码表示指令的基本格式如下：

- 操作码是说明指令操作性质的二进制数代码。
- 操作码所占的二进制位数决定了一台计算机所能允许的指令条数。

$$L_{op}=\log 2^{n}$$

例如，操作码占用六位二进制码时，这台计算机最多允许有：$2^6=64$ 条指令

地址码则指明操作数的地址。（被操作数，操作数，操作结果）

指令字长度与机器字长度的关系

(1) 机器字长

- 机器字长度简称字长，指计算机能直接处理的二进制数据的位数。
- 字长还能反映指令的直接寻址能力，若字长n位全用来寻址，可直接寻址2^n个字节。
- 为了便于处理字符数据及尽可能地利用存储空间，一般把机器字长定为字节长度（8位）的整数倍，即是8位、16位、 32位或64位。
- 微型、小型机的字长多为8位、16位和32位，中、大型机的字长多为32位和64位。

因此，一个字中可以存储1个、2个、4个或8个字符。

(2) 指令字长

指令字的长度取决于操作码的长度、操作数地址的长度和操作数地址的个数。

由于操作码的长度、操作数地址的长度以及所采用操作数地址数目不同，各种指令的长度不是固定的，当然也不是任意的。

为了充分利用存储空间，指令字的长度也定为字节长度的整数倍。例如INTEL 8086/80586系列机的指令长度分别为8位、16位、24位、32位、48位等,而最长的指令可达120位。

指令字长度与机器字长度没有固定的关系，它可以等于机器字长，也可以大于或小于机器字长。

采用多字长指令的目的

解决寻址较大存储空间的问题

取指令要多次访内,影响速度，占用存储空间较大。

几个基本概念

在一个指令系统中，若所有指令的长度都是相等的，称为定长指令字。定长结构指令系统控制简单，但不够灵活。

若各种指令的长度随指令功能而异，就称为变长指令字。变长结构指令系统灵活，能充分利用指令长度，但指令的控制较复杂。

指令字长度等于机器字长度的指令，称为单字长指令；

指令字长等于两个机器字长的指令，称为双字长指令；

指令字长度只有半个机器字长度的指令，称为半字长指令。

指令分类方法

1、按计算机系统的层次结构分类

宏指令　机器指令　微指令

2、按操作数物理位置分类

访问内存　　存储器－存储器（MM）型

访问寄存器　　寄存器－寄存器（RR）型

访问内存和寄存器型　寄存器－存储器（RM）型

3、按操作数个数分类

四地址指令　三地址指令　二地址指令　一地址指令　零地址指令

二、地址码

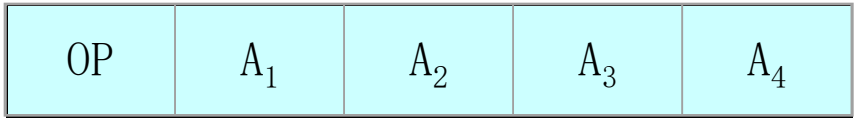
指令中的地址码用来指出该指令的源操作数地址(一个或两个)、结果地址及下一条指令的地址。

这里的地址可以是主存地址，也可以是寄存器地址，甚至可以是I/O设备的地址。

下面以主存地址为例，分析指令的地址码字段。

(1) 四地址指令

指令格式：



OP：操作码；

A_1：第一地址码，存放第一操作数；

A_2：第二地址码，存放第二操作数；

A_3：第三地址码，存放操作结果；

A_4：第四地址码，存放下条要执行指令的地址。

操作：（A_1）OP（A_2）→ A_3

- 这种指令直观易懂，后续指令的地址可任意填写。
- 可直接寻址的地址范围与地址字段的位数有关。

  例如：指令字长32位，操作码占8位，4个地址段各占6位，

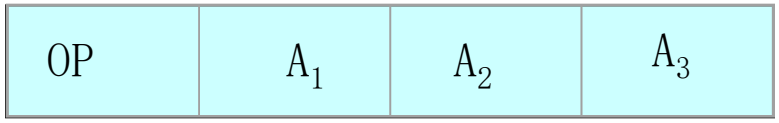
  则指令的直接寻址范围为：　$2^6 = 64$

- 如果地址字段均指示主存的地址，则完成一条四地址指令，共需**访问四次**存储器（取指令一次，取两个操作数两次，存结果一次）。

- 因为程序中大部分指令都是顺序执行的，当采用指令计数器后，A_4地址可以省去。

(2) 三地址指令

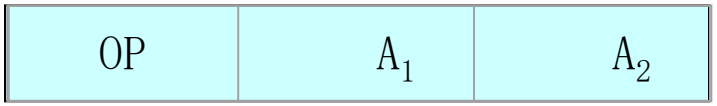
指令格式：



- 三地址指令中各项含义与四地址指令相同。由于采用了指令计数器 （又称程序计数器，简称PC），省去了A_4地址；
- 用三地址指令编写的程序，其指令在内存中必须依次存放，才能利用程序计数器自动增量的办法顺序执行。若程序要转向时，必须用转移指令改变程序的执行顺序。
- 如果指令字长不变(32位)，操作码仍为8位，故三地址指令直接寻址范　　围可达：　$2^8=256$
- 若地址字段均为主存地址，则完成一条三地址指令也需四次访问存储器。

(3) 二地址指令

指令格式：



OP：操作码；

A_1：既作第一操作数地址，又作目的地址；

A_2：第二操作数地址。

操作：（A_1） OP（A_2） → A_1

- 有的机器也表示（A_1） OP（A_2） → A_2，A_2既作第一操作数地址，又作目的地址；

- 在不改变字长和操作码位数的前提下，二地址指令可直接访问的主存地址数为：$2^{12}=4K$

- 使用二地址指令编写的程序，其指令在内存中也要依次存放，才能用程序计数器自动增量使之顺序执行。若程序发生转向时，也必须用转移指令改变程序的执行顺序。

- 当二地址指令执行之后，A_1中的内容被修改了。

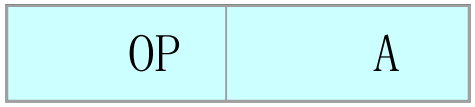
有的机器规定A_2为目的地址，这时则是A_2的内容被修改了。

- 若地址字段均为主存地址，则完成一条二地址指令也需四次访问存储器。

(4) 一地址指令

指令中只给出一个操作数地址，另一个操作数地址和目的地址则是隐含的。这个隐含的地址就是运算器的累加寄存器AC。

指令格式：



操作：（AC）OP（A）→ AC

- 在不改变字长和操作码位数的前提下，二地址指令可直接访问的主存地址数为：$2^{24} = 16M$
- 完成一条指令只需两次访存；

- 采用一地址指令编写的程序，其指令在内存中也要顺序存放，由程序计　数器自动增量控制其顺序执行。程序转向时，也用转移指令改变程序的执行方向。
- 在程序执行前，必须用一条“取数指令”把其中一个操作数放到累加寄存器中。

程序结束后，累加寄存器的内容已被修改。若要将累加寄存器中的结果送回内存，则必须使用“存数指令”。

(5) 零地址指令

- 没有操作数地址的指令称为零地址指令。
- 执行零地址指令时，被运算的操作数地址全部是隐含的，指令格式中只说明作什么操作。
- 如停机指令就是零地址指令。

三、操作码

操作码字段分两种：

(1) 固定长度操作码

指操作码所占的二进制位数固定不变，且集中放在指令字的一个字段中。

有利于简化硬件设计，减少指令译码时间，广泛用于字长较长的大、中型计算机和超级小型计算机中。

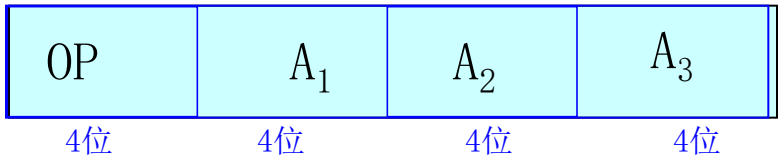
IBM370和VAX－11系列机中，操作码的长度都是8位。

(2) 可变长度操作码

操作码的长度是可变的，且分散地放在指令的不同字段中。有利于压缩程序中操作码的平均长度，在字长较短的微型机中被广泛应用。

PDP-11，INTEL 8086/80386等，其操作码的长度均是可变的。

一种可变长度操作码示例

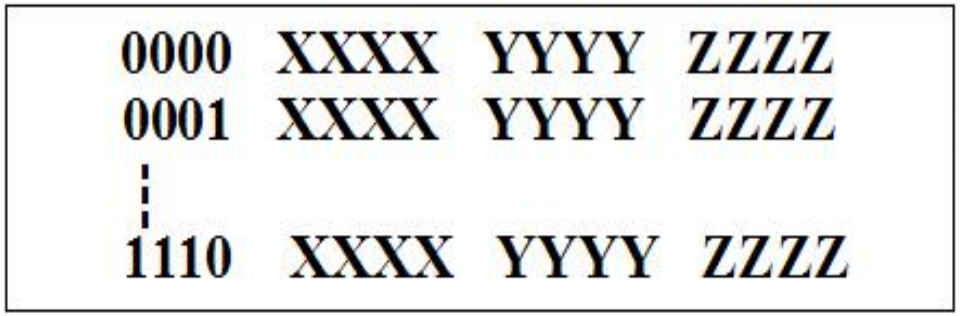


这是一个16位字长的指令码，包括4位基本操作码字段和三个4位长的地址字段。4位基本操作码，若全部用于三地址指令，则有16条。

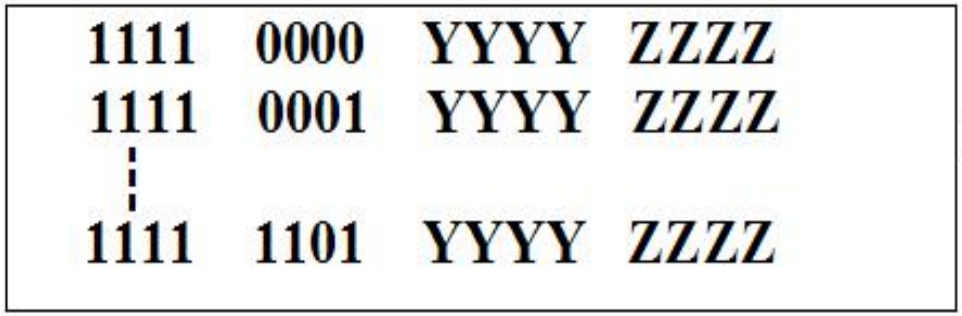
显然，4位基本操作码是不够的，必须向地址码字段扩展操作作码的长度。其扩展方法及步骤如下：

① 15条三地址指令的操作码由4位基本操作码0000～1110所给定，剩下一个1111则用

于把操作码扩展到X地址码字段，即由4位扩展到8位；



15条三地址指令



14条二地址指令

② 14条二地址指令的操作码由8位操作码的1111,0000～1111,1101 给定，剩下的1111,1110和1111,1111又可用于把操作码扩充到 Y地址字段，即从8位又扩充到12位；

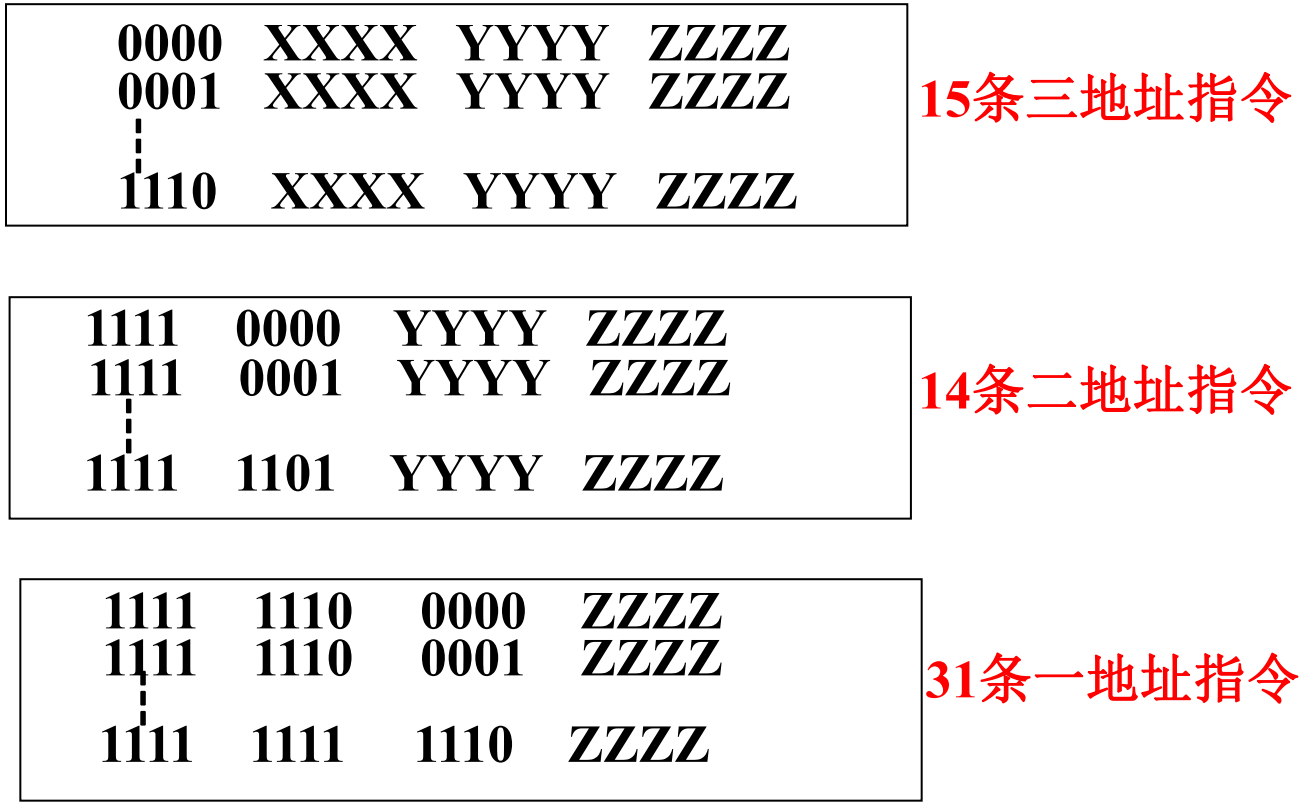
1111	0000	YYYY	ZZZZ
1111	0001	YYYY	ZZZZ
⋮			
1111	1101	YYYY	ZZZZ

14条二地址指令

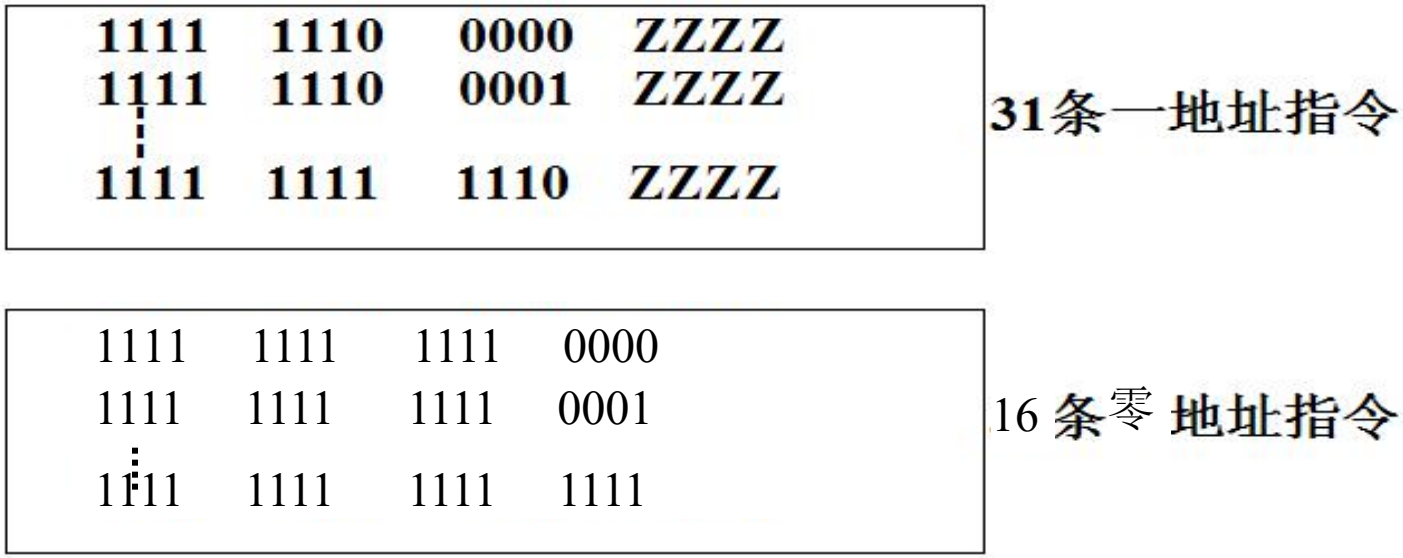
1111	1110	0000	ZZZZ
1111	1110	0001	ZZZZ
⋮			
1111	1111	1110	ZZZZ

31条一地址指令

③ 31条一地址指令的操作码由12位操作码的1111, 1110, 0000～1111, 1111, 1110给定。



由此类堆，还可以把剩下的1111, 1111, 1111扩充到Z地址码字段而形成的16位操作码，这时还可由1111, 1111, 1111, 0000~1111, 1111, 1111, 1111给出16条零地址指令。



除了这种安排外，还有多种其它安排方法。

如：形成15条三地址指令，12条二地址指令，31条一地址指令，16条零地址指令，共74条指令。

实际的机器可以采用各种灵活的扩展方式，其思路与此类似。

上述方法是在不增加指令长度的情况下，采用对地址少的指令使用较长的操作码，对地址数多的指令使用较短的操作码。其实质是增加了指令的数量，丰富了指令的功能。

在可变长操作码的指令系统设计中，究竟使用何种扩展方法为好，指令的使用频度（即在程序中出现的概率）是非常重要的依据。即频度高的指令应分配短的操作码，频度低的指令则分配较长的操作码。这样，既可有效地缩短操作码在程序中的平均长度，节省存储空间，又可缩短常用指令的译码时间以提高程序的运行速度。

其缺点是译码系统比固定操作码复杂，增加了设计控制器的难度，需要更多的硬件作支持。

指令格式举例

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。

单字长指令

双字长指令

三字长指令

内存按字节编址，所以单字长指令每执行一条指令后，指令地址加 1 。双字长指令或三字长指令每执行一条指令时，指令地址要加2或加3，可见多字长的指令格式不利于提高机器速度。

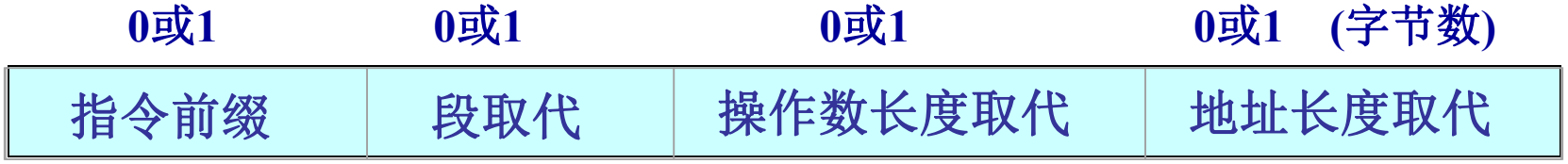
PDP/11系列机指令格式

PDP/11系列机指令字长16位，其指令格式如下表所示。

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	

在PDP/11中，操作码字段是不固定的，其长度也是不相同的。这样做可以扩展操作码以包含较多的指令。但是操作码字段不固定，对控制器的设计来说必将复杂化。

Pentium指令格式



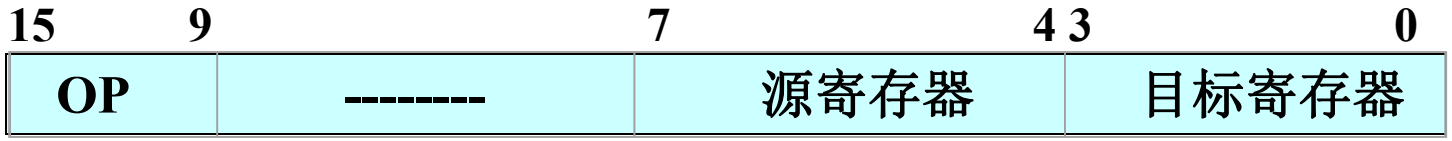
(a) 前缀

(b) 指令

Mod-R/M字段： 规定了存储器操作数的寻址方式，给出了寄存器操作数的存器地址号。

SIB字段： 和Mod-R/M字段一起，对操作数来源进行完整的说明。

例：指令格式如下所示，OP为操作码，分析指令格式的特点。



解：

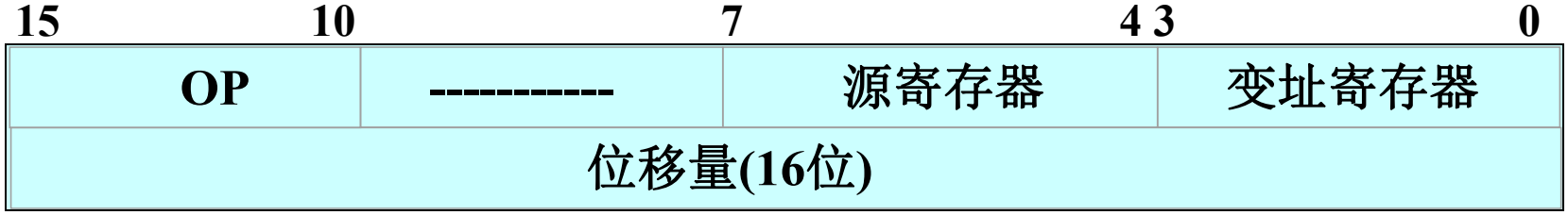
(1) 单字长二地址指令。

(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。

例 指令格式如下所示，OP为操作码字段，分析指令格式特点。



解：

(1) 双字长二地址指令，用于访问存储器。

(2) 操作码字段OP为6位，可以指定64种操作。

(3) 一个操作数在源寄存器（共16个），另一个操作数在存储器中（由变 址寄存器和位移量决定），所以是RS型指令。

寻址方式

存储器既可以存放数据又可以存放指令。因此当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，就是该操作数或指令在存储器中的地址。

- 在存储器中，操作数或指令字写入或读出的方式，有地址指定方式、相联存储方式和堆栈存取方式。
- 几乎所有的计算机，在内存中都采用地址指定方式
- 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，即指令寻址方式和数据寻址方式，前者比较简单，后者比较复杂。

1. 指令的寻址方式

指令的寻址方式有两种，一种是顺序寻址方式，另一种是跳跃(转移)寻址方式。

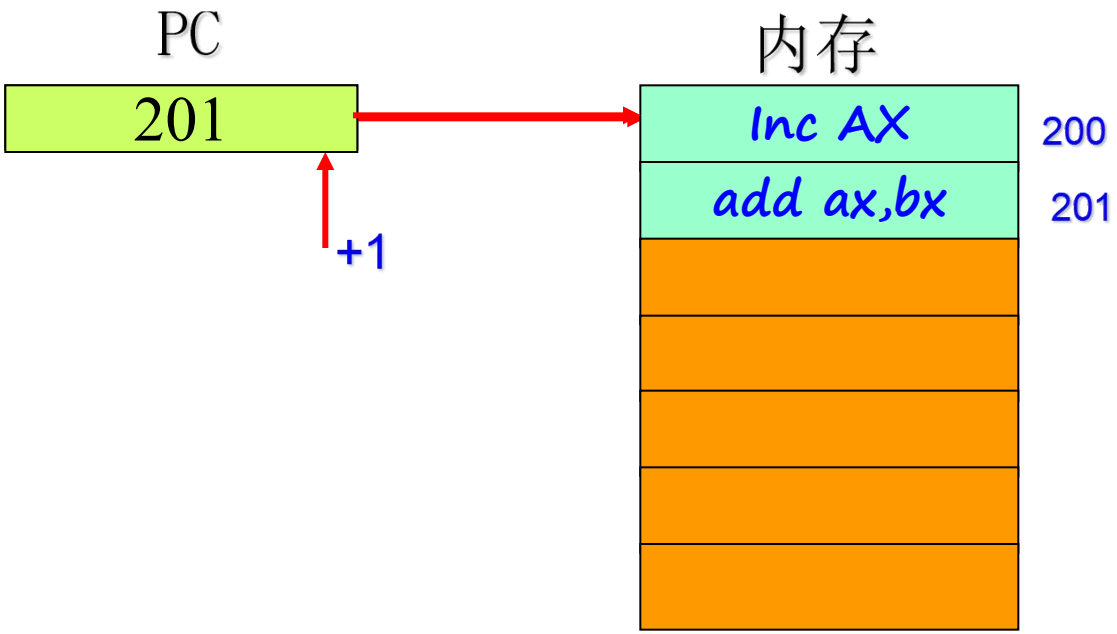
(1) 顺序寻址方式

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，再执行第二条指令；接着再取出第三条指令……这种程序顺序执行的过程，我们称为指令的顺序寻址方式。

为此，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。

顺序寻址过程演示



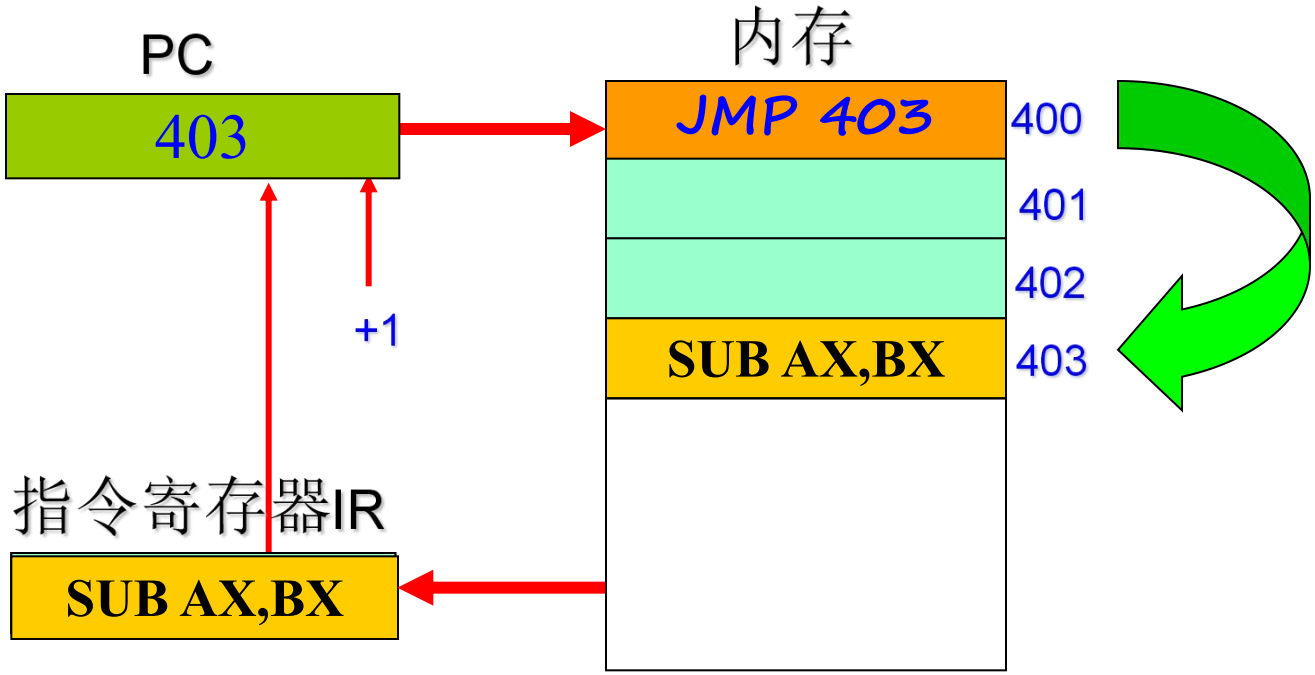
(2) 转移寻址方式

当程序转移执行的顺序时，指令的寻址就采取转移寻址方式。

所谓转移，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。程序转移后，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。

采用指令转移寻址方式，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或无条件转移指令，就是为了实现指令的转移寻址而设置的。

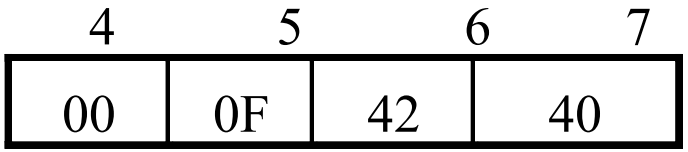
转移寻址过程演示



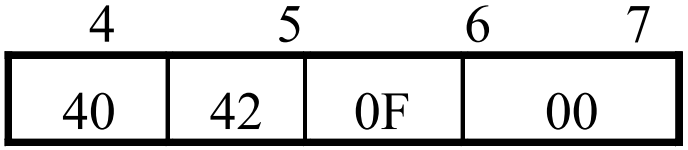
数据在存储器存储方式

1. 存储方式

- 大数端(Big Endian)：最低字节存储在高地址
- 小数端(Little Endian)：最低字节存储在低地址
- 32位字000F4240H



(a) 大数端存储方式



(b) 小数端存储方式

当在具有不同定义的机器之间，传输存储字时，必须变换字节的顺序，以保证数据的成功复制

2. 边界对齐问题

对齐访问地址

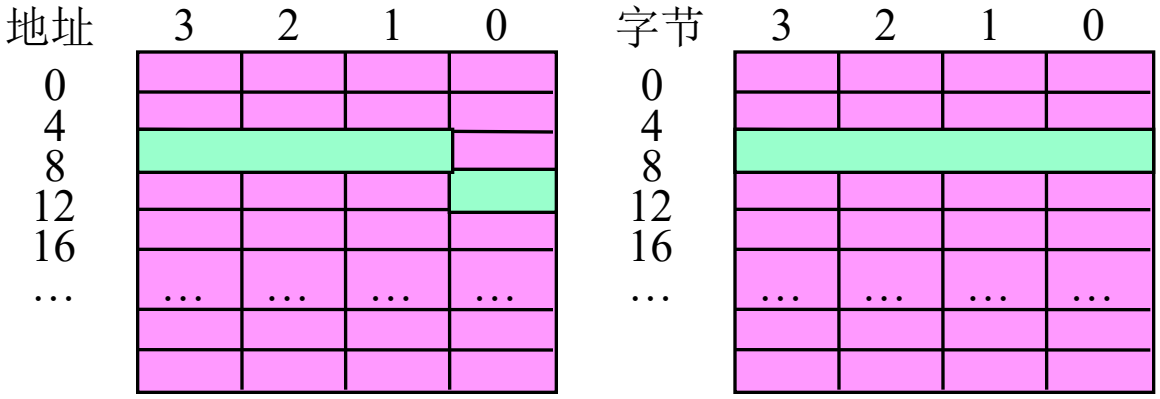
字节　　XXXXXXXXXXX

半字　　XXXXXXXXXX0

字　　XXXXXXXXX00

双字　　XXXXXXXX000

要求对象只能安放于其大小的倍数的地址上



(a) 字不对齐　　(b) 字对齐

存储器中字的对齐情况

2. 操作数的寻址方式

- 操作数的寻址就是寻找操作数的地址，其目的是寻找所需要的操作数。
- 寻址方式就是指寻找操作数地址所用的方法，它与计算机的硬件结构密切相关，对指令的格式和功能都有极大的影响。从使用的观点来看，寻址方式和汇编语言程序设计的关系更加密切。
- 由于各种计算机的硬件结构各不相同，指令格式亦多种多样，因而，寻址方式也就千差万别，这里仅对常见的单操作数地址的寻址方式作介绍。用这些最基本的寻址方式可以组合成各类计算机中更为复杂的寻址方式。

寻址方式的设计目的：

(1) 丰富程序设计手段，提高程序质量；

(2) 减少访问主存的次数，压缩程序占用的存储空间。保证指令的地址码字段尽可能缩短，而访问的存储空间尽可能地大。

区分不同的寻址方式

为了能区分出各种不同寻址方式，必须在指令中给出标识。标识的方式通常有两种：显式和隐式。

显式的方法就是在指令中设置专门的寻址方式字段，用二进制代码来表明寻址方式类型。

隐式的方式是由指令的操作码字段说明指令格式并隐含约定寻址方式。

显式

隐式

注意：一条指令若有两个或两个以上的地址码时，各地址码可采用不同的寻址方式。例如，源地址采用一种寻址方式，而目的地址采用另一种寻址方式。

MOV AX, (BX)

寻址方式

一种单地址指令的一般结构

操作码(OP)	寻址特征(X)	形式地址(D)

操作码OP	变址X	间址I	形式地址D

- 指令中操作数字段的地址码是由形式地址和寻址方式特征位等组合形成；

一般来说，指令中所给出的地址码，并不是操作数的有效地址。

- 因此寻址过程就是把操作数的形式地址D，变换为操作数的有效地址E的过程。

实际有效地址为E，实际操作数S

S= (E)

(1) 隐含寻址

在指令中不明显地给出操作数的地址，其操作数或操作数的地址隐含在某个通用寄存器中或指定的存储单元中。

例如:

① 单地址的指令格式，没有在地址字段中指明第二操作数地址，而是规定累加寄存器AC作为第二操作数地址，AC对单地址指令格式来说是隐含地址。

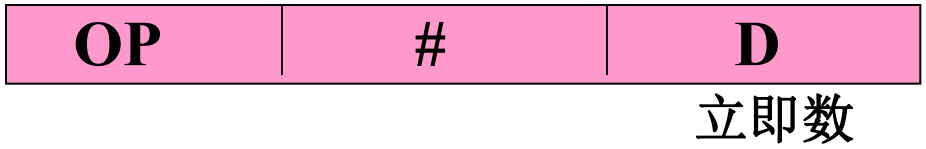
② IBM-PC机中的乘法指令：MUL　OPR

这种方式可以缩短指令的长度，在字长较短的微型机或小型机中被广泛采用。

(2) 立即寻址

所需的操作数由指令的直接给出，称为立即数寻址方式，简称立即寻址。

立即寻址特征



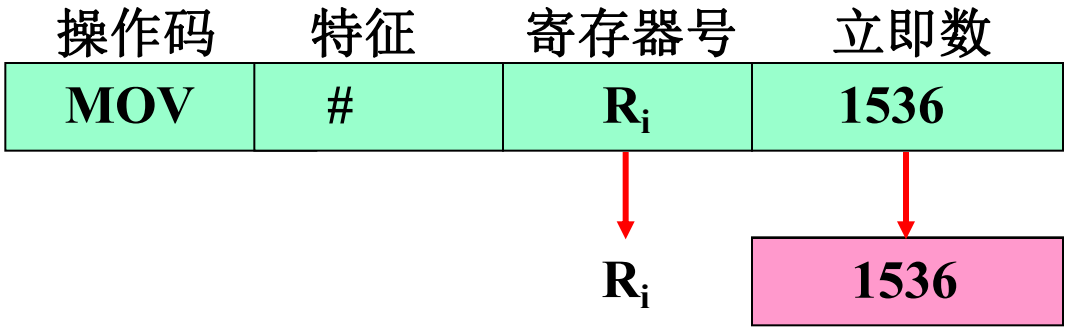
地址码字段是操作数本身

S=D

这种方式的特点是：

取指令时将操作码和一个操作数同时取出，不必再次访问存储器。提高了指令的执行速度，但操作数只是指令的一部分，其数值受到了限制。

例如：

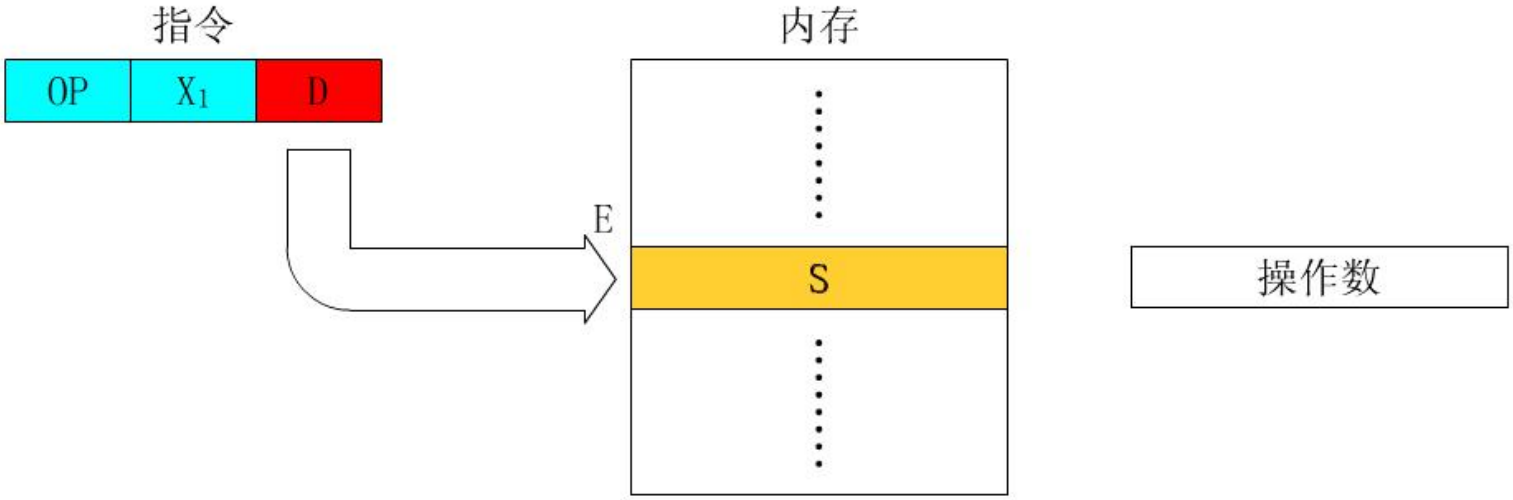


这条指令的执行结果是“把立即数1536传送到目的寄存器Ri中”。

其汇编符号记为：MOV R_i，#1536。

(3) 直接寻址

直接寻址是指指令中的地址码就是操作数的有效地址，按这个地址可直接在存储器中存入或取得操作数。



采用直接寻址方式时，指令字中的形式地址D就是操作数的有效地址E，既E＝D。因此通常把形式地址D又称为直接地址。此时，由寻址模式给予指示，如X_1=0。 如果用S表示操作数，那么直接寻址的逻辑表达式为 ：

$$S=(E)=(D)$$

例：

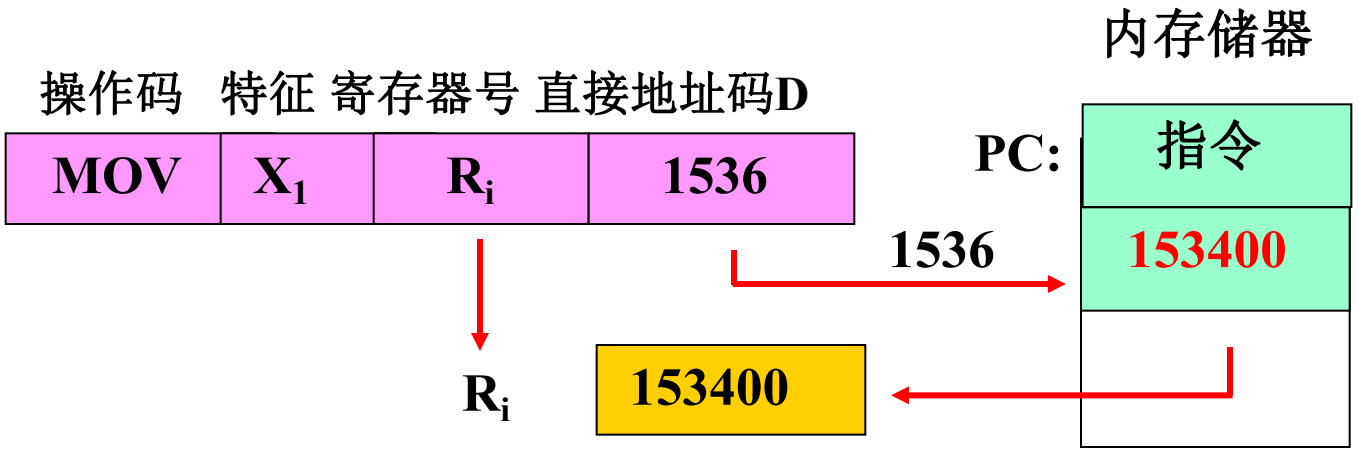
图中特征码X_1指明是采用直接寻址。

该指令表示“把1536号存储单元中存放的操作数153400取出，送入存器R_i中 ”。

其汇编符号记可能为：MOV R_i，(D)

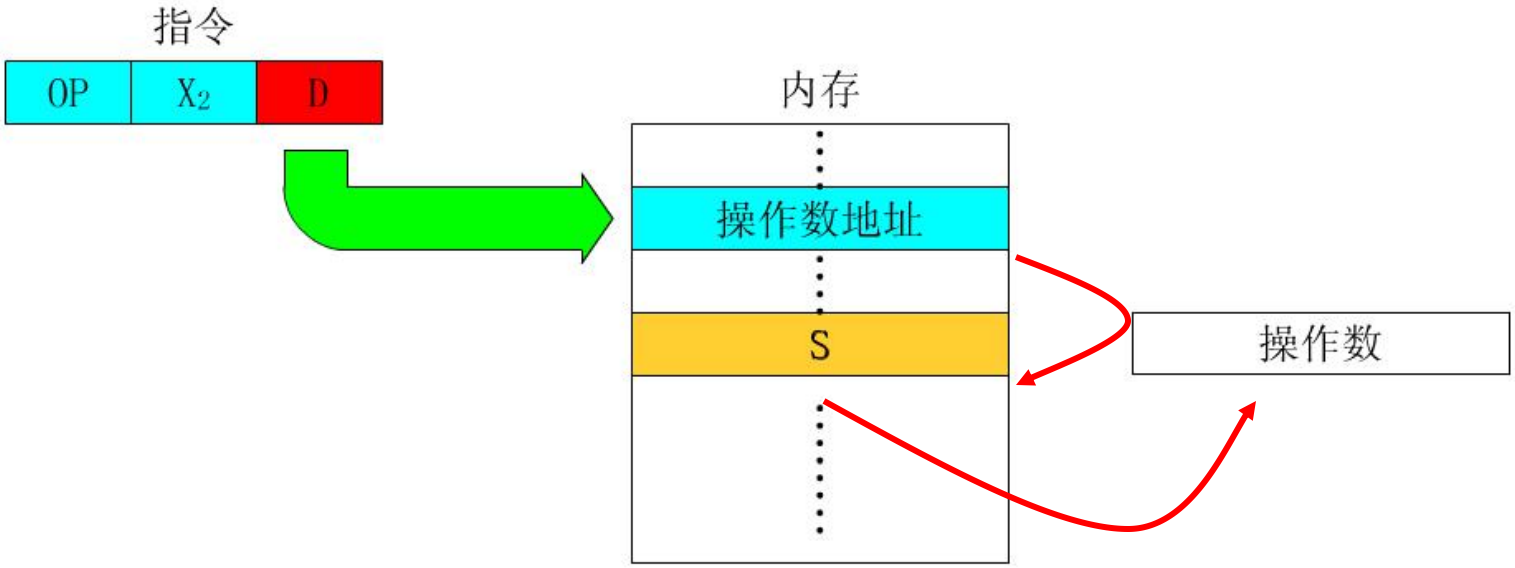
直接地址也可以用公式表示，若设有效地址为E，则可记为：

E=D （D就是有效地址）



(4) 间接寻址

指令中的地址码部分给出的既不是操作数，又不是操作数的地址，而是存放操作数地址的内存单元的地址，这个地址叫做间接地址。其寻址方式称为间接寻址方式，简称间址。

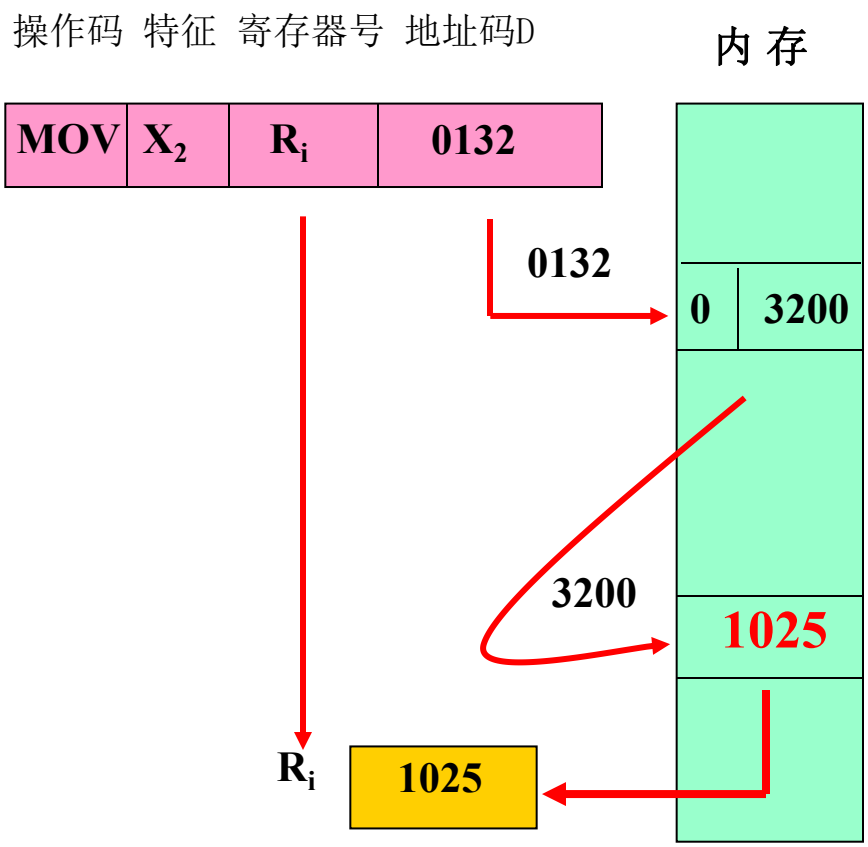


D单元的内容是操作数地址，D是操作数地址的地址

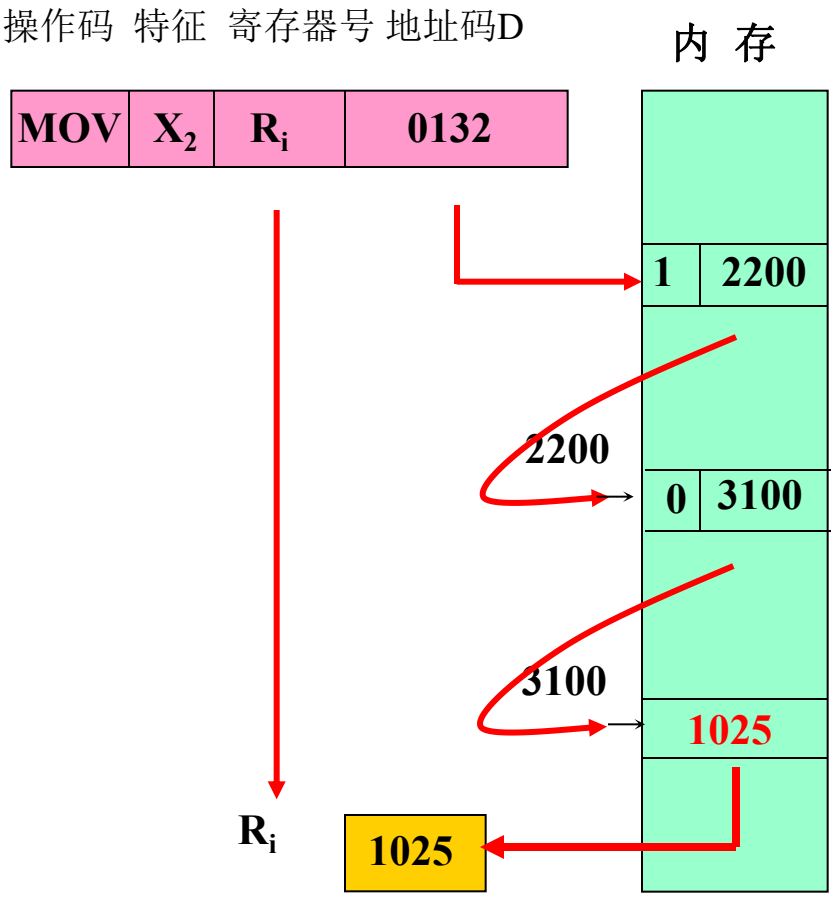
$$E=(D)$$

通常在间接寻址情况下，由寻址特征位给予指示。

例如：



(a) 一次间址 MOV R_i, ((D))



(b)二次间址 MOV R_i, (((D)))

需要多次访问内存，速度慢已被淘汰

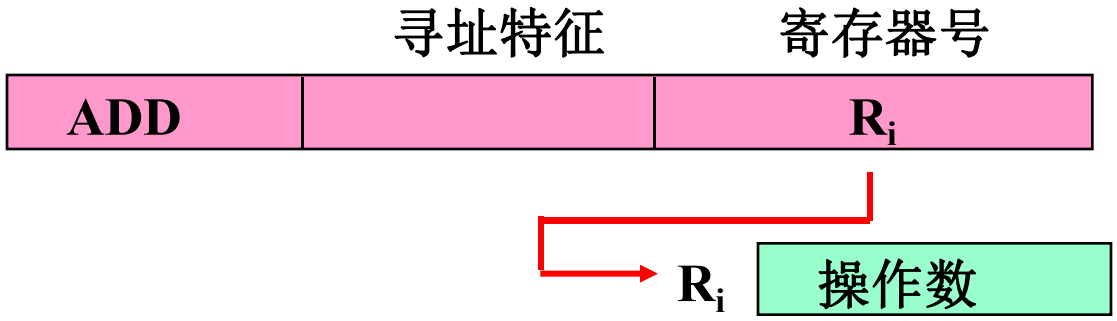
(5) 寄存器寻址方式和寄存器间接寻址方式

① 寄存器直接寻址

如果指令的地址码部分给出的是某通用寄存器编号R_i，R_i寄存器中存放着操作数，则称为寄存器直接寻址。

例如：执行ADD R_i指令，

该指令说明R_i是操作数的有效地址，从R_i中取出操作数与累加器的内容相加，其结果放在累加器中。

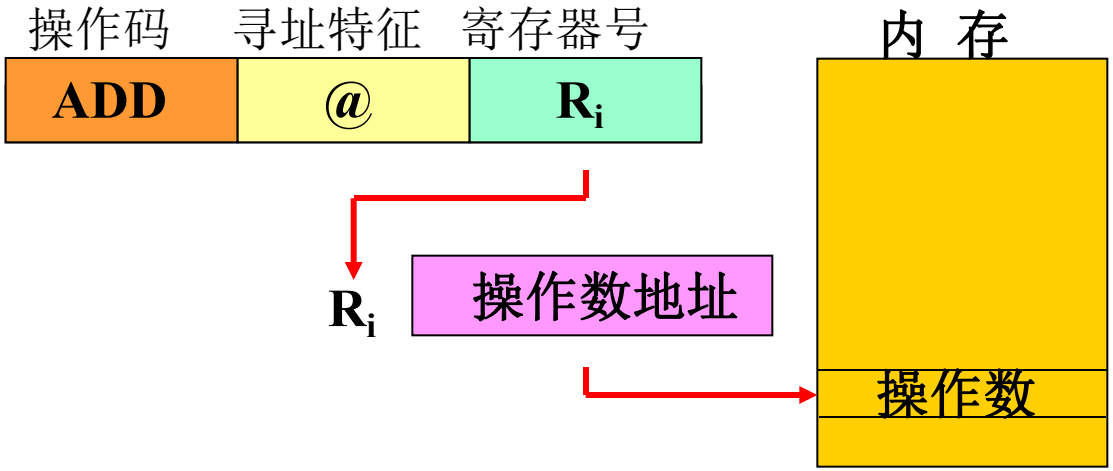


S=(R)

② 寄存器间接寻址

寄存器间接寻址，是指令中地址字段所指的寄存器中存放的是操作数的地址。

例如：执行ADD @R_i 指令时,说明R_i不是操作数的有效地址,而是存放操作有效地址的寄存器号。



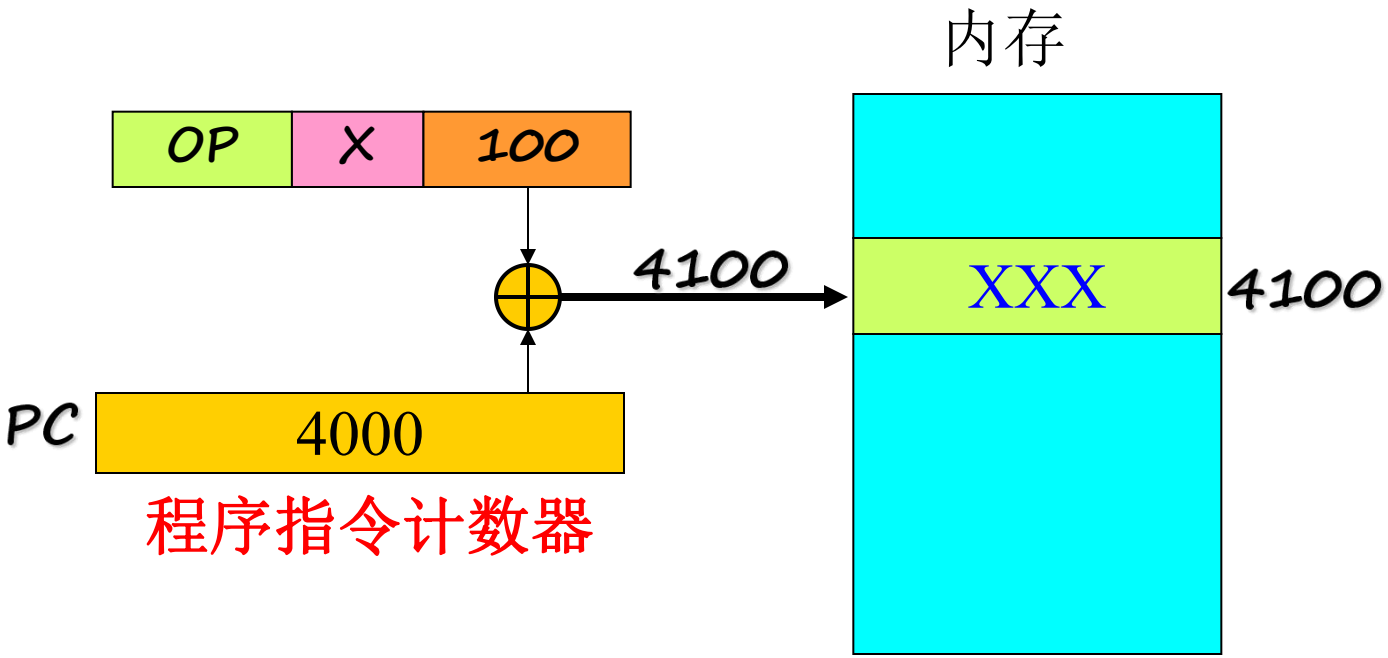
E=(R)

(6) 相对寻址方式

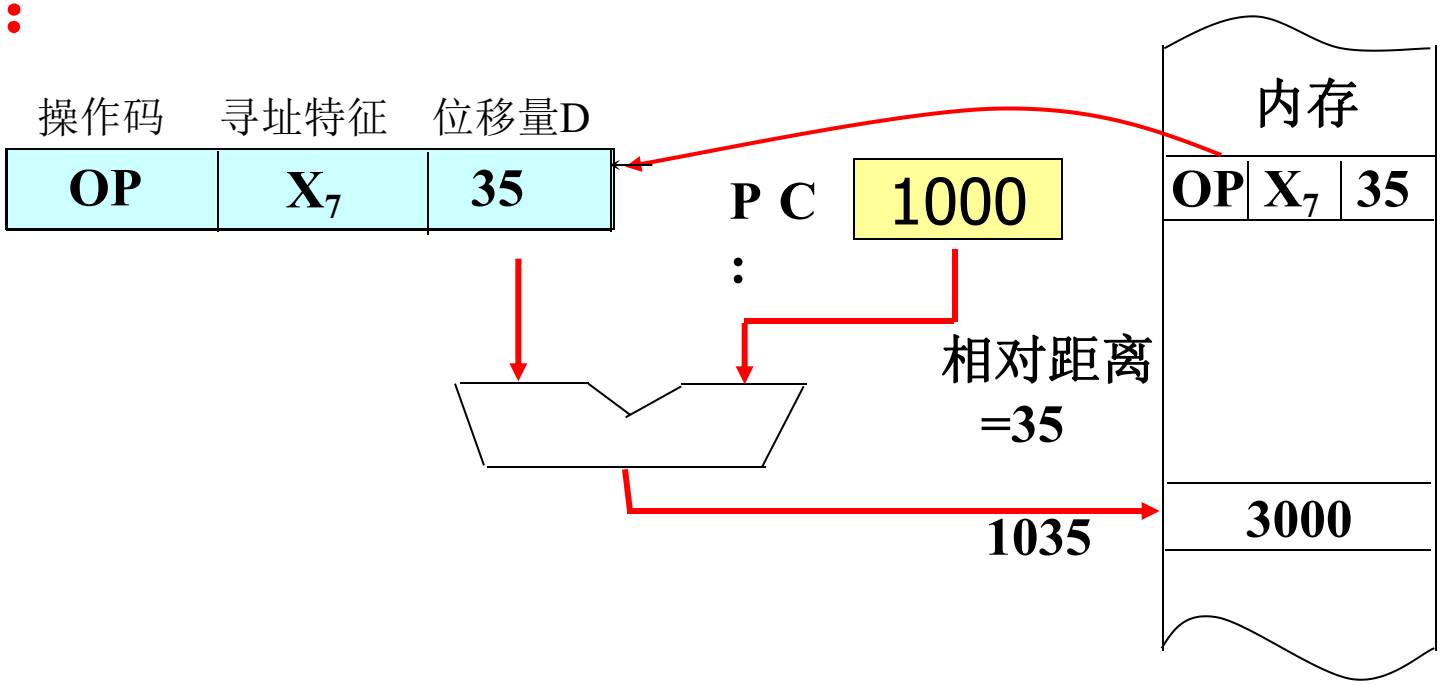
指令中的形式地址D 加上PC的内容作为操作数的地址.

E=D+(PC)

形式地址D通常称为偏移量，其值可正可负，相对于当前指令地址进行浮动。



例如：

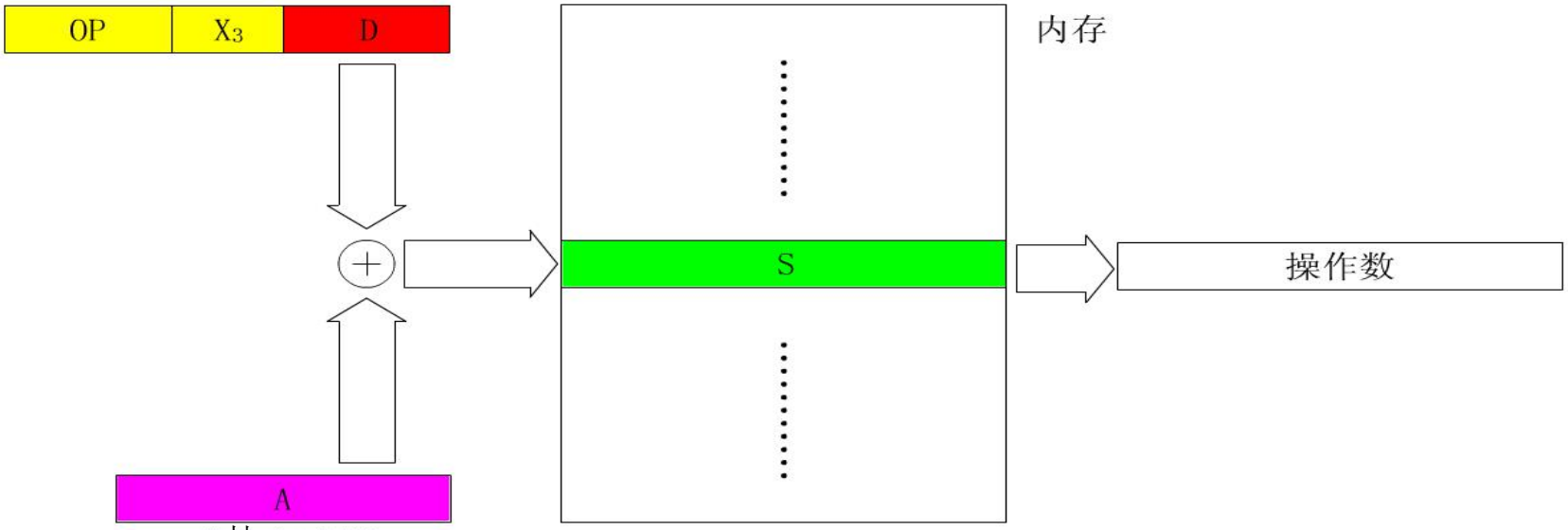


PC的内容为1000，指令的位移量（相对距离）为35。故其相对地址为1035。这个地址是不固定的，随PC的值变化而变化，并且相对地差一个固定值。因此，无论程序装入存储器的任何区域，只要这个差值不变，程序均能正确运行。由于程序在内存空间里是浮动的，又称浮动寻址。

(7) 基址寻址方式

当存储容量较大，所需地址码的长度大于字长时，指令中的地址码部分直接给出的地址不可能直接访问存储器的所有单元。因此，把整个存储空间分成若干段，每段的首地址存放在一个基址寄存器中，段内的位移量由指令直接给出。

存储器的实际地址就等于基址寄存器的内容加上段内位移量。这就叫做基址寄存器寻址方式，简称基址寻址。

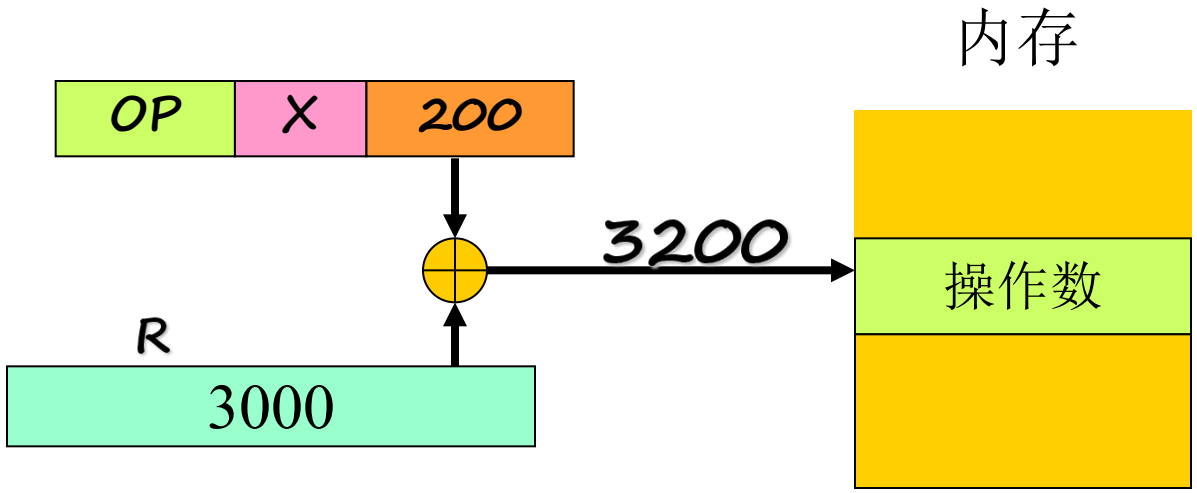


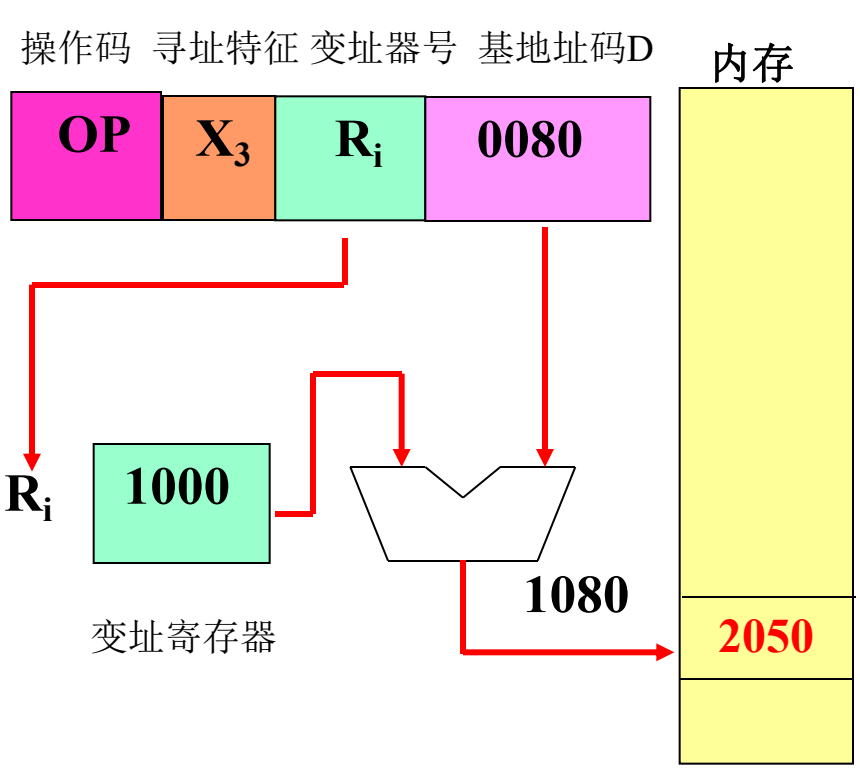
(8) 变址寻址方式

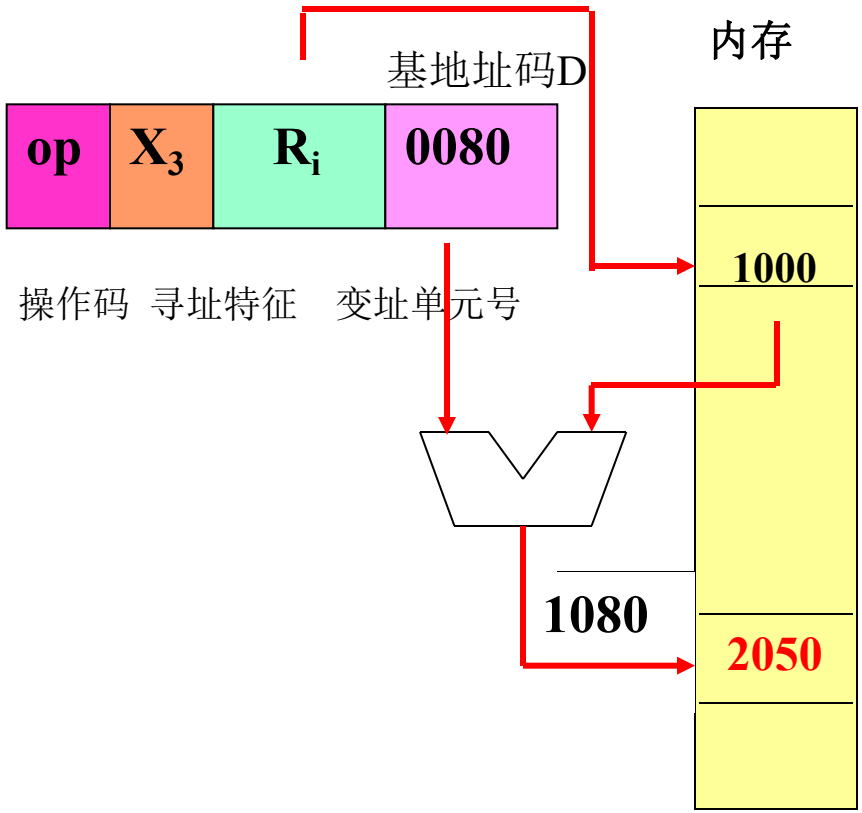
变址寻址是将指令中的基地址和一个“变址值”相加后形成操作数的有效地址。“变址值”存放在一个专用的变址寄存器R中或专用的内存单元中。

E=D+(R)

MOV AX, 200[SI]　　**SI,DI** 都称为变址寄存器







X_3=1 EA=(R_i)+D

有效地址 EA=1000+0080=1080

操作数=(1080)=2050

(a) R_i为变址寄存器时的变址寻址

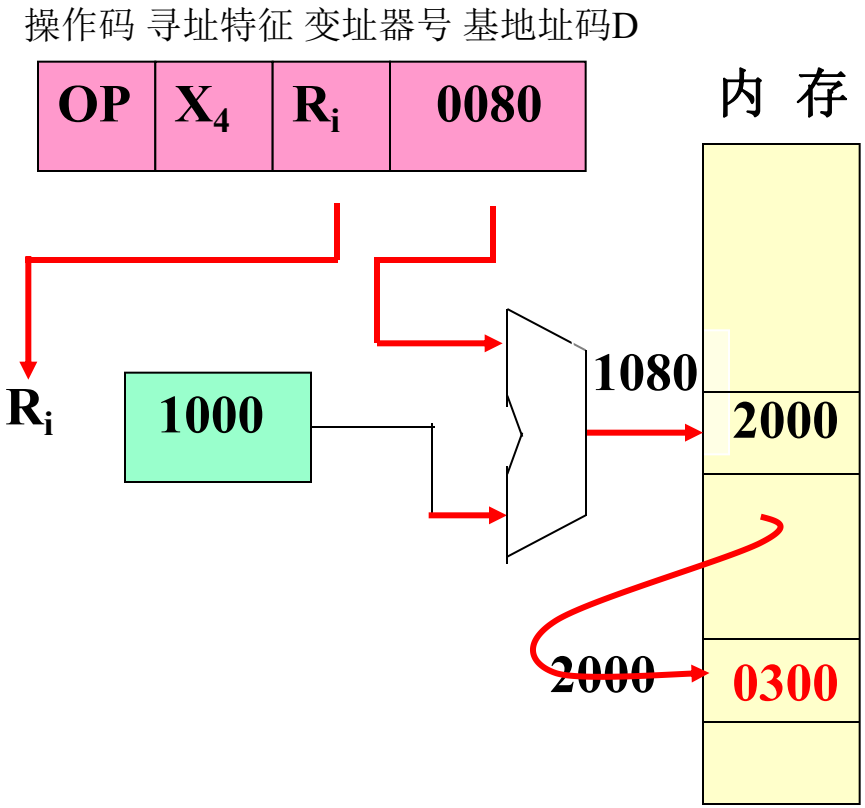
X_3=1 EA=(R_i)+D

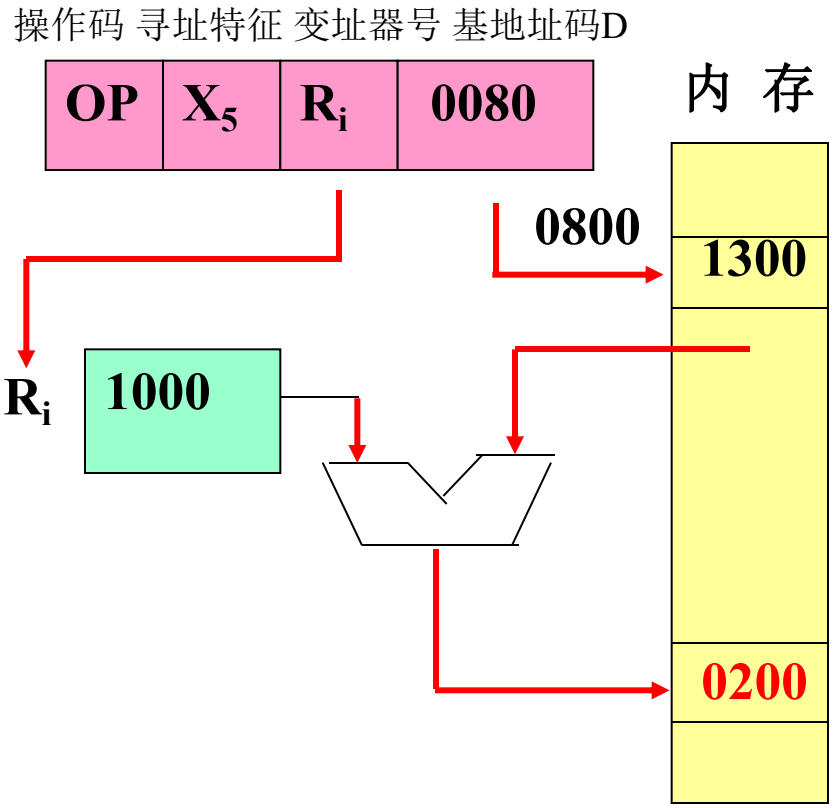
有效地址 EA=1000+0080=1080

操作数=(1080)=2050

(b)R_i为内存变址单元时的变址寻址

复合型寻址方式（变址+间址）





X_4=1 EA=((R_i)+D)

有效地址 EA=(1000+0080) =2000

操作数 = (2000) = 0300

(a)先变址方式

X_5=1 EA=(D)+(R_i)

有效地址 EA=1300+1000=2300

操作数 = (2300) = 0200

(b)后变址方式

基址和变址的应用方向

基址寻址面向系统，主要用于逻辑地址到物理地址的变换，用来解决程序在主存储器中的再定位和扩大寻址空间等问题。在一些大型机中，基址寄存器是由管理程序利用特权指令来使用，用户程序无权修改它，从而确保了系统的安全性。

变址寻址则面向用户，用于访问字符串、向量和数组等成批数据，没有逻辑地址到物理地址的变换功能。在一些小型机或微型机中，基址寻址和变址寻址实际上已经合二为一了。

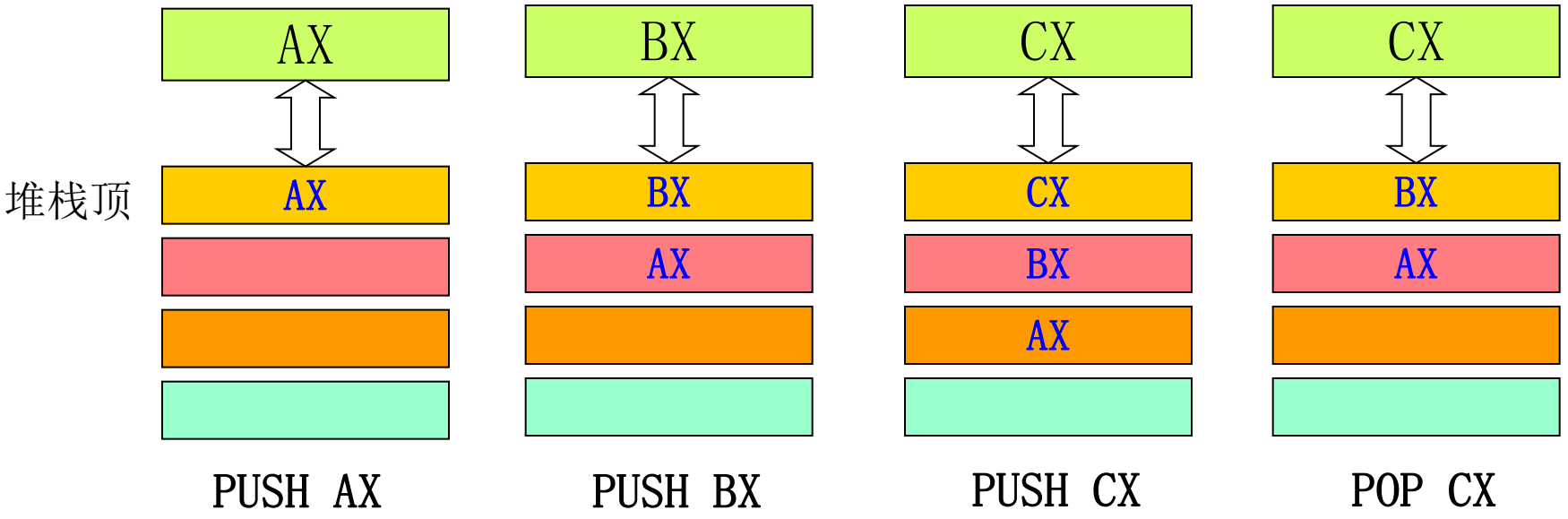
(9) 堆栈寻址方式

堆栈-------一组能存取数据的暂时存储单元。

串联堆栈

- 一组专门的寄存器，一个R保存一个数据。
- 数据的传送在栈顶和通用寄存器之间进行。
- 快速：在CPU内部实现
- 串行：进栈和出栈涉及到栈内所有其它数据的移动；
- 破坏性读出：读数据的同时也离开了堆栈；
- 栈容量有限：取决于CPU内堆栈专用寄存器的数量；
- 栈顶不动，数据移动。

堆栈寻址方式



这种堆栈由硬件支持，又称硬堆栈。

存储器堆栈

用一部分主存空间作堆栈称为存储器堆栈。

- 堆栈的数目、长度可随意指定
- SP---堆栈指示器(栈指针)， CPU中一个专门寄存器，SP内容是栈顶单元地址。改变SP内容即可移动栈顶的位置。
- 堆栈操作期间，堆栈中数据不动，栈顶移动
- 非破坏性读出
- 存储器堆栈称为软堆栈
- 速度慢

当建立存储器堆栈时，可用程序来设置。把一个主存地址送入堆栈指示器(SP)，就可确定堆栈的顶。

软堆栈的两种生成方式

软堆栈按堆栈指示器中的初值进行相加或相减，可分成两种生成方式。堆栈指示器SP通常指向栈顶（或栈底）：

- 若采用SP的内容由初值不断向大的地址方向推进时（即（SP）+Δ→SP），称为自顶向下生成方式，简称“向下生成”方式;
- 若采用SP的内容由初值不断向小的地址方向推进时（即（SP）－Δ→SP），称为自底向上生成方式，简称“向上生成”方式。

堆栈操作

堆栈操作使用一种特殊的数据传送指令，即压入指令（PUSH）和弹出指令（POP）。

若采用“向上生成”的堆栈，其操作过程如下：

- 压入指令：PUSH　OPR，是把OPR（设长度为一个字）压入堆栈。其操作是：

  OPR →（SP）；　（SP）－1 →SP 。

- 弹出指令：POP OPR，是从堆栈弹出一个数据（长度为一个字）送OPR，操作是：

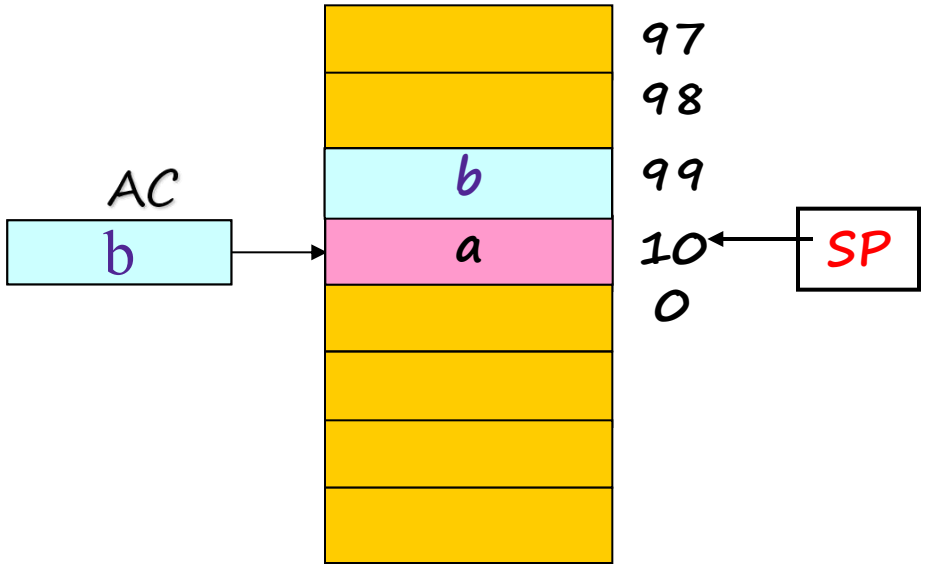
  （SP）+1 → SP　　（SP）→OPR。

压栈操作

压栈---累加器中的数送堆栈保存.

例：

(AC) →堆栈MSP　　堆栈指针(sp) －1 →sp



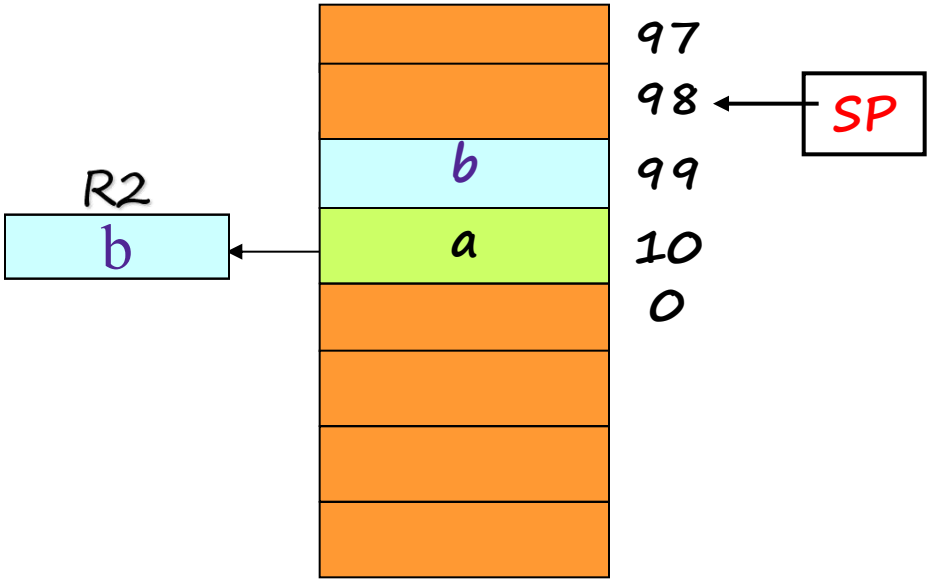

```
mov   AC,  a
PUSH  AC
mov   AC, b
PUSH  AC
POP   R2
```

出栈操作

出栈---------将堆栈中的数取出送累加器

堆栈指针 (sp)＋1 → sp （堆栈MSP）→ R2

POP R2



寻址方式举例

1.PDP / 11系列机寻址方式

(1) 寄存器

PDP / 11计算机CPU中有8个程序可访问的寄存器，其编号为R_0－R_7：

- R_7作为程序计数器（PC）使用，
- R_6作为堆栈指示器（SP）使用，
- 其他6个寄存器R_0－R_5可作为通用寄存器。

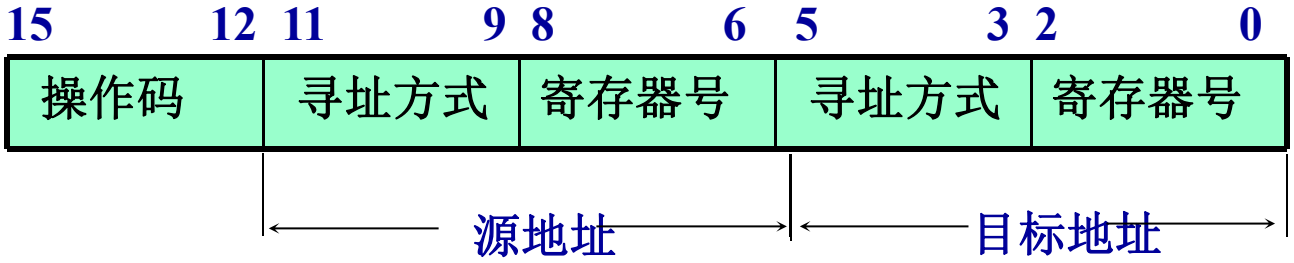
(2) 指令格式

PDP／11系列机指令字长为16位；

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	

- 指令系统中有单操作数指令和双操作数指令；
- 操作数字段即地址部分均由6位二进制构成。

例如双操作数指令为：



- 其中3位对应寄存器编号，形式地址就存放在这些寄存器中,以充分利用指令位数.
- 还有3位作为寻址特征位，指示8种寻址方式。
- 此外有4种指令计数器型寻址方式是一种隐含寻址方式，它 们以访问R_7(程序计数器)为标志 。

(3)PDP/11系列机寻址方式

	寻址名称	寻址特征值	有效地址E	汇编格式	说明
直接型	寄存器型	000	E=R	R_n	寄存器R_n的内容是操作数，相当于直接地址
	自增型	010	E=(R); (R)+2→R	$(R_n)+$	寄存器R_n的内容是操作数地址，然后递增寄存器内容
	自减型	100	(R)−2→R; E=(R)	$-(R_n)$	先递减寄存器R_n的内容，然后作为操作数的地址
	变址型	110	(PC)+2→PC; E=((PC))+(R)	$X(R_n)$	寄存器R_n的内容与本指令下一单元所存的数相加，作为操作数的地址

	寻址名称	寻址特征值	有效地址E	汇编格式	说明
间接型	寄存器间接	001	E＝ (R)	@ R_n或(R_n)	寄存器R_n的内容作为操作数地址，相当于间接地址
	自增间接	011	E＝(R)； (R)＋2→R	@(R_n)＋	寄存器R_n的内容作为操作数地址的地址，然后递增寄存器内容
	自减间接	101	(R)－2→R； E＝((R))	@ －(R_n)	寄存器R_n的内容先递减，然后作为操作数地址的地址
	变址间接	111	(PC)＋2→PC； E＝(((R))＋((PC)))	@ X(R_n)	寄存器R_n的内容与本指令下一单元中的内容相加，作为操作数据地址的地址

	寻址名称	寻址特征值	有效地址E	汇编格式	说明
程序计数器型	立即型	010	(PC)+2→PC; E=（PC）	#n	指令下一个单元是操作数
	绝对值	011	(PC)+2→PC; E=((PC))	@#A	指令下一个单元内容是操作数的地址
	相对值	110	(PC)+2→PC; ((PC))+(PC)+2=E	A	指令下一个单元内容与指令地址加4的数相加，其和作为操作数地址
	相对间接型	111	(PC)+2→PC; E=(((PC))+(PC)+2)	@ A	指令下一单元内容与指令地址加4的数相加，其和作为操作数地址的地址

PDP／11的寻址方式特征是采用寄存器进行寻址：

- 直接型和间接型寻址方式是在通用寄存器中寻址；
- 当在寄存器R_7中寻址时，变为四种指令计数器型的寻址方式。

汇编语言：

- 当指令的操作码用助记符表示，而地址及其寻址特征也用符号表示时，就成为汇编语言。
- 这些符号称为汇编符号。
- 用汇编符号表示指令格式，就称为汇编格式。
- 各种机器的汇编语言对机器的依赖性太强，彼此间不能通用。

2.Pentium的寻址方式

(1) Pentium的工作模式

实地址模式

在实地址模式下，逻辑地址形式为段寻址方式：

将段名所指定的段寄存器内容（16位）左移4位，低4位补全0, 得到20位段基地址, 再加上段内偏移, 即得20位物理地址。

保护虚拟地址模式

在保护模式下, 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。

无论是实地址模式还是保护模式，段基地址的获取方式已是固定的方式。我们主要讨论有效地址的获取方式。

(2) Pentium指令格式及寻址方式

序号	寻址方式名称	有效地址E算法	说明
(1)	立即		操作数在指令中
(2)	寄存器		操作数在某寄存器内，指令给出寄存器号
(3)	直接	E=Disp	Disp为偏移量
(4)	基址	E=(B)	B为基址寄存器
(5)	基址＋偏移量	E=(B)+Disp	
(6)	比例变址＋偏移量	E=(I)×S+Disp	I为变址寄存器，S为比例因子(1,2,4,8)
(7)	基址＋变址＋偏移	E=(B)+(I)+Disp	
(8)	基址＋比例变址＋偏移量	E=(B)+(I)×S＋Disp	
(9)	相对	指令地址=(PC)+Disp	PC为程序计数器或当前指令指针寄存器

(字节数)

1或2		0或1			0或1		0,1,2,4	0,1,2,4
操作码	Mod	Reg或操作码	R/M	比例S	变址I	基址B	位移量	立即数
	2位	3位	3位	2位	3位	3位		

(1) 立即寻址：立即数可以是8位、16位、32位。

(2)寄存器寻址：

一般指令：

或使用8位通用寄存器(AH, AL, BH, BL, CH, CL, DH, DL);

或使用16位通用寄存器(AX, BX, CX, DX, SI, DI, SP, BP);

或使用32位通用器(EAX, EBX, ECX, EDX, ESI, EDI, ESP， EBP);

对64位浮点数操作，要使用一对32位寄存器。

少数指令

以段寄存器(CS, DS, ES, SS, FS, GS)来实施寄存器寻址方式。

(3) 直接寻址：也称偏移量寻址方式，偏移量长可以是8位、16位、32位。

(4) 基址寻址：基址寄存器B可以是上述通用寄存器中任何一个。
基址寄存器B的内容为有效地址。

(5) 基址＋偏移量寻址：基址寄存器B是32位通用寄存器中任何一个。

(6) 比例变址＋偏移量寻址：

也称为变址寻址方式，变址寄存器I是32位通用寄存器中除ESP外的任何一个，而且可将此变址寄存器内容乘以1，2，4或8的比例因子S，然后再加上偏移量得到有效地址。

(7) 基址＋变址＋偏移

(8) 基址＋比例变址＋偏移量

} 是(4)，(6) 两种寻址方式的组合，此时偏移量可有可无。

(9) 相对寻址： 适用于转移控制类指令。用当前指令指针寄存器EIP或IP的内容（下一 条指令地址）加上一个有符号的偏移量，形成CS段的段内偏移。

寻址方式举例

例 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

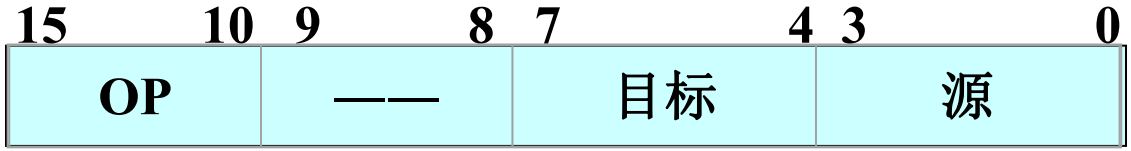
例 某16位机所使用指令格式和寻址方式如下所示。该机有两个20位基值寄存器，四个16位变址寄存器， 十六个16位通用寄存器。

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。

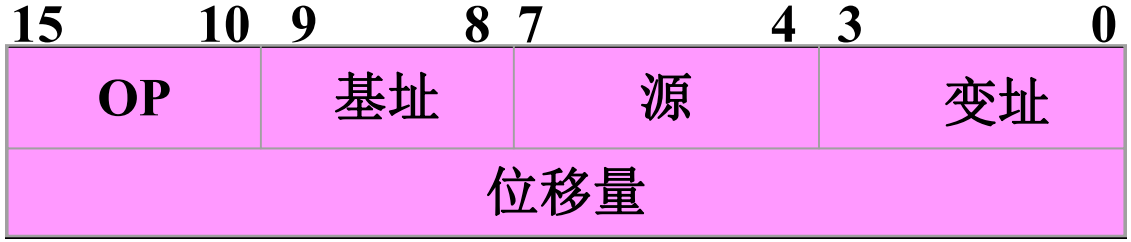
三种指令的操作码分别是 MOV (OP)＝$(A)_H$， MOV是传送指令，

STA (OP)＝$(1B)_H$， STA为写数指令

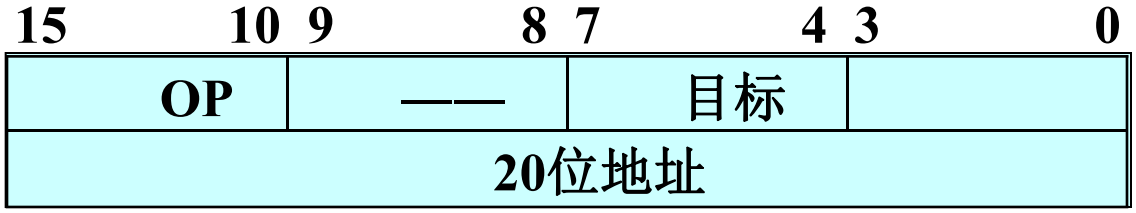
LDA (OP)＝$(3C)_H$。 LDA为读数指令



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：

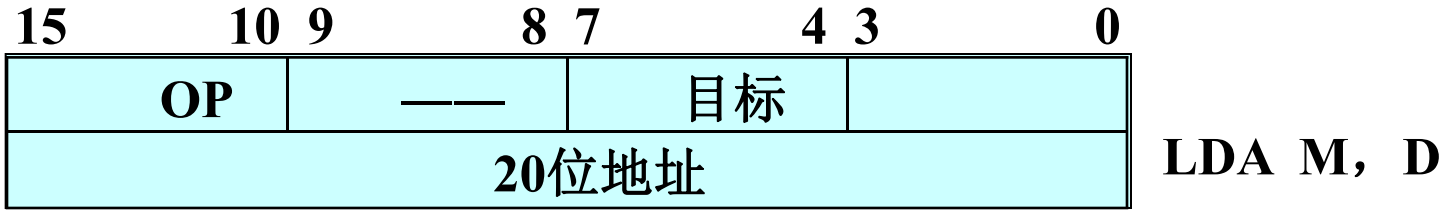
MOV(OP)＝001010，STA(OP)＝011011，LDA(OP)＝111100，

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

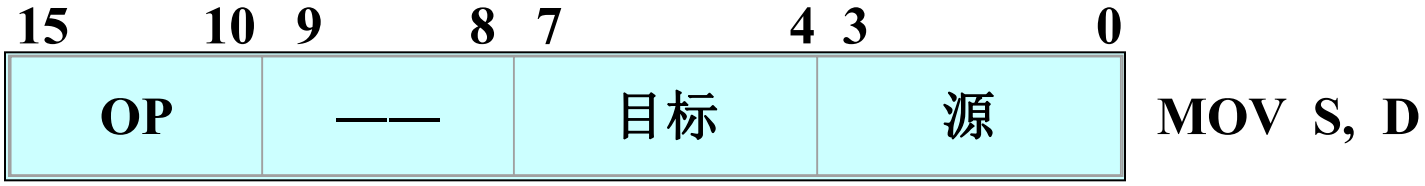
根据已知条件：

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。



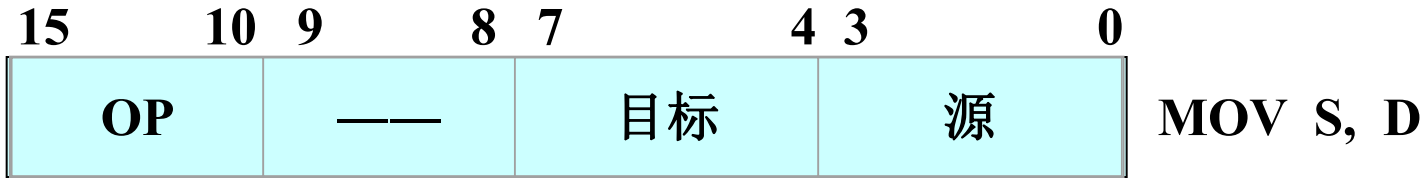
根据已知条件：MOV(OP)＝001010， STA(OP)＝011011， LDA(OP)＝111100，

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 = $(28C2)_H$。



指令的分类

一台计算机的指令系统通常有上百条或几百条指令，从它们所完成的功能来看，一个较为完善的指令系统，应具备以下各类指令。

1.数据传送类指令

这类指令的功能是实现寄存器与寄存器，寄存器与存储单元以及存储单元与存储单元之间的数据传送。

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。

2. 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，算术比较指令，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（如IBM 370机、国产银河机、CRAY-1机、CDC的STAR－100机等）中还设置有向量运算 指令，可直接对整个向量或矩阵进行求和、求积运算。

3. 逻辑运算指令

这类指令包括逻辑加、逻辑乘、逻辑比较、测等指令、按位加、逻辑移位等指令，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. 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。

转子指令是转向本指令操作地址所指出的子程序入口，并将原程序中下一条指令地址存入内存某单元（或指定的某个寄存器）中，以便为返回主程序时提供返回地址。

5. 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。

6. 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。

7. 陷阱(TRAP)指令

陷阱是一种意外事故中断，它的目的不是请求CPU正常处理中断，而是为了把发生的故障通知CPU，并根据故障的情况转入相应的故障处理程序。陷讲指令就是为实现这个任务而设置的。

在一般计算机中，陷阱指令是一种隐含指令（它不出现在指令系统中，其功能是以指令的形式隐含地完成的），用户不能使用，只有当故障出现时，才由CPU自动产生并执行之，故又叫自中断指令。

但是，有些计算机中又设置有可供用户使用的陷阱指令，可用它来实现调用系统程序的请求。

例如，INTEL 8086的软件中断指令，实质上是一种直接提供用户使用的陷阱指令，它可完成调用系统子程序的过程。其汇编格式为：INT TYPE，其中TYPE是一个8位常数，表示中断类型。执行时，根据中断类型可以找到相应系统子程序的入口地址。

8. 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。

9. 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。

精简指令系统计算机

RISC的产生和发展情况

RISC的主要特点

RISC的产生和发展情况

1.RISC产生的时代背景

- 早期的计算机结构简单，指令条数少，指令系统功能弱，计算机的性能较差。
- 随着**VLSI**技术的迅速发展，硬件成本不断下降，软件成本不断上升，促进人们在指令系统中增加更多的指令和更复杂的指令，以适应不同应用领域的需要。
- 特别是系列机问世之后，为了能做到程序兼容，新设机型或高档机除了要继承老机器的指令系统中的全部指令外，还要增加若干新的指令，从而导致同一系列计算机的指令系统越来越复杂，机器结构也越来越复杂。

目前，大多数计算机的指令系统多到几百条指令。 这就体现了计算机性能越高，其指令系统越复杂的传统设计思想。这类计算机称为“复杂指令系统计算机”，简称CISC（Complex Instruction Set Computer）。

例如：

VAX11/780计算机有303条指令，18种寻址方式；

Pentium机有191条指令，9种寻址方式。

复杂指令需进行复杂的操作，从而降低了机器的执行速 度。

对CISC指令系统所作的运行统计分析表明，各种指令使用频率相差悬殊，常用的较为简单的指令，仅占指令总数的20%，但在程序中使用的频率却占80%。下表是HP公司研究了IBM 370计算机的高级语言运行情况后所得到的统计数据。

指令类别	COBOL	FORTRAN	PASCAL
转移	24.2%	18.0%	18.4%
逻辑操作	14.6%	8.1%	9.9%
存数取数	40.2%	48.7%	54.0%
存储单元送存储单元	12.4%	2.1%	3.8%
整数运算	6.4%	11.0%	7.0%
浮点运算	0.0%	11.9%	6.8%
十进制运算	1.6%	0.0%	0.0%
其它	0.6%	0.2%	0.1%

国外一些计算机公司和大学开展了对指令系统合理性的研究，1975年IBM公司的John Cocke提出了精简指令系统的想法，后来出现了各种各样的“精简指令系统计算机”，简称RISC（Reduced Instruction Set Computer）。也称为RISC技术。

RISC是一种设计思想。其目标是设计出一套能在高时针频率下单周期执行简单而有效的指令集。设计的重点在于降低由硬件执行指令的复杂度。

2.RISC的发展概况

RISC技术的实质是要求指令系统简化，尽量使用寄存器—寄存器操作指令，指令操作在一个周期内完成，指令格式力求一致，以利于提高编译的效率。

(1)第一代RISC处理器

RISC I机

- 1982年美国加州伯莱克大学研究的RISC I机(以后又有RISC II)：
- 采用VLSI CPU芯片, 片上晶体管数为44000个, 线宽3μm，字长32位，
- 有128个寄存器（但程序员只能看到32个）这些寄存器均为32位，
- 它只有31条指令，两种寻址方式。其中，只有两种存储器访问指令，LOAD（取数）和STORE（存数）指令。

其功能超过VAX－11/780或M68000，而速度比VAX－11/780几乎快一倍。

同时，斯坦福大学推出MIPS机，IBM公司推出IBM 801机，显示了RISC的生命力，并推动了RISC技术的不断发展。

这些机种称第一代RISC处理器，具有32位数据通路，支持Cache，但软件支持较少，性能与CISC处理器相当。

(2)第二代RISC处理器

第二代RISC处理器在第一代的基础上提高了集成度和时钟频率，建立了比较完善的Cache分层存储体系。它们已具有单指令流水线，每次顺序执行多条指令，每个周期发出一条指令。例如，MIPS公司的R3000处理器，则采用了25MHZ和33MH$_Z$的时钟频率，芯片集成度是11.5万晶体管，字长32位。

(3)第三代RISC处理器

第三代RISC处理器则采用了巨型计算机或大型计算机的设计技术——超级流水线（Superpipelining）技术和超标量（Superrscalar）技术。提高了指令级的并行处理能力，使RISC处理器的整体性能得到改善。

例如，MIPS公式的R4000处理器采用50MH_Z和75MH_Z的外部时钟频率（内部流水线时钟是100MH_Z和150MH_Z），芯片集成度高达110万晶体管，字长是64位，并包含有16KB的Cache在芯片上。它具有R4000PC、R4000 SC、R4000 MC三种版本，分别提供给台式系统、高性能服务器和多处理器环境下使用。

自1983年开始出现商品化RISC机以来，比较有名的是RISC计算机有：

IBM公司的IBM RT系列；

HP公司的精密结构计算机（HPPA）；

MIPS R3000；

Motorola M88000；

Intel 80960；

INMOS Transputer；

AMD Am29000；

Fairchild clipper等。

其中Clipper兼顾了RISC和CISC两方面的特点，又称为类RISC机。

RISC的主要特点

精简指令系统计算机不仅是要简化指令系统，而且要通过简化指令系统而使计算机的硬件结构变得简单合理，以提高运算速度，最有效的办法是减少指令的执行周期数。

1.P、C、I

- 若设高级语言程序经编译后在机器上运行的机器指令数为I;
- 每条机器指令执行时所需要的平均机器周期数是C;
- 每个机器周期的执行时间为T，

计算机执行程序的时间P可用下式计算：

$$P=I\times C\times T$$

RISC/CICS 的I、C、T统计表

	I	C	T
RISC	1.2 ~ 1.4	1.3 ~1.7	<1
CICS	1	4 ~ 6	1

其中，I、T为比值，C为实际周期数）

- 由于RISC指令比较简单，原CISC机中比较复杂的指令可用RISC中一段子程序代替，因此，RISC中的I比CISC的多20%—40%;
- 但是，RISC的大多数指令只用一个机器周期实现，C的值比CISC的小得多;
- 又因为RISC结构简单，完成一个操作所经过的数据通道较短，因此，T的值大为减少。

2.RISC主要特点

(1) 选取使用频率最高的一些简单指令和很有用但不复杂的指令;

(2) 指令长度固定，指令格式种类少，寻址方式种类少;

(3) 访问存储器指令极少，只有取数/存数指令（Load/Store），其余指令的操作都在寄存器之间进行。

(4) 大部分指令在一个机器周期内完成，复杂指令可化为简单指令序列，把它当成子程序使用;

(5) CPU中通用寄存器很多;

(6) 控制器采用组合逻辑部件，而不采用微程序控制;

(7) 取指令和执行指令采用流水线重叠操作，提高了运行速度和信息处理能力;

(8) 使存储器靠近CPU，以减少传送数据的时间延迟，并在主存和外存之间设置中间速度存储器，以加速主存和外存之间的信息传送;

(9) 以简单有效的方式支持高级语言。

1.典型RISC机指令系统的基本特征

型号	指令数	寻址方式	指令格式	通用寄存器数	主频 /MH_Z
RISC-I	31	2	2	78	8
RISC-II	39	2	2	138	12
MIPS	55	3	4	16	4
SPARC	75	4	3	120-136	25-33
MIPSR3000	91	3	3	32	25
i860	65	3	4	32	30


THANKS FOR ALL