

# 计算机组成原理

PRINCIPLES OF COMPUTER ORGANIZATION

第22次课：指令系统-下

杜国栋

信息科学与工程学院计算机科学与工程系

gddu@ysu.edu.cn



燕山大学  
YANSHAN UNIVERSITY

设计某指令系统时，假设采用16位定长指令字格式，操作码使用扩展编码方式，地址码为6位，包含零地址、一地址和二地址3种格式的指令。若二地址指令有12条，一地址指令有254条，则零地址指令的条数最多为（ ）。

- A 0
- B 2
- C 64
- D 128

 提交

设计某指令系统时，假设采用16位定长指令字格式，操作码使用扩展编码方式，地址码为6位，包含零地址、一地址和二地址3种格式的指令。若二地址指令有12条，一地址指令有254条，则零地址指令的条数最多为（ ）。

- A 0
- B 2
- C 64
- D 128

 提交



设计某指令系统时，假设采用**16位定长指令字格式**，操作码使用扩展编码方式，**地址码为6位**，包含零地址、一地址和二地址3种格式的指令。若二地址指令有12条，一地址指令有254条，则零地址指令的条数最多为（）。

- 地址码占6位，指令字长16位，二地址指令指令码占 $16-6\times 2=4$ **位**，最多可以表示 $2^4=16$ **条二地址指令**，二地址指令有12条，剩余 $16-12=4$ 种情况，此时对于一地址指令，二地址指令的高位地址码融入二地址指令指令码作为一地址指令的指令码，最多可以表示 $4\times 2^6=256$ 条一地址指令，一地址指令有254条，剩余 $256-254=2$ 种情况，此时对于零地址指令，低位地址码融入一地址指令指令码作为零地址指令的指令码，最多可以表示 $2\times 2^6=128$ 条零地址指令。





# 课程目标

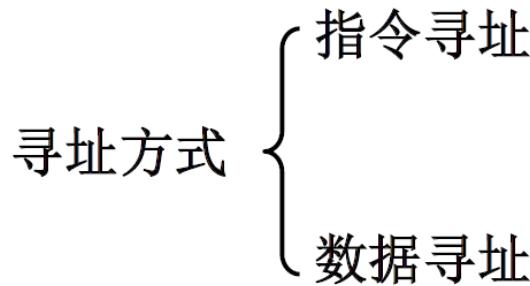
- 掌握被广泛采用的基本寻址方式；
- 熟悉按照完成的功能指令的分类；
- 了解指令系统的兼容性和精简指令系统。





# 寻址方式

- 形成操作数地址或寻找操作数的方式；
- 确定本条指令的操作数地址，下一条要执行指令的指令地址；
- 1条指令，可能会涉及多种寻址方式；





# 指令寻址

顺序  $(PC) + 1 \longrightarrow PC$

跳跃 由转移指令指出

	指令地址	指令	指令地址寻址方式
+1	PC		
0	LDA	1000	顺序寻址
1	ADD	1001	顺序寻址
2	DEC	1200	顺序寻址
3	JMP	7	顺序寻址
4	LDA	2000	
5	SUB	2001	
6	INC		
7	STA	2500	跳跃寻址
8	LDA	1100	顺序寻址
9	:		





# 指令寻址

## 1.顺序寻址方式

- 当程序按顺序执行时的指令寻址方式；
- 必须用程序计数器记录所要执行指令的存放单元地址；
  - ◆一般做顺序加1的操作；
  - ◆程序计数器又称指令指针寄存器；

## 2.跳跃寻址方式

- 当程序转移执行时的指令寻址方式；
- 程序计数器的内容由本条指令给出，而不是顺序改变。





# 数据寻址

操作码

寻址特征

形式地址 A

形式地址

指令字中的地址

有效地址

操作数的真实地址

约定

指令字长 = 存储字长 = 机器字长





# 数据寻址-立即寻址

指令中直接包含了操作数。

定长格式: 操作码 0 | ... | 立即数 |

数在指令中，  
其长度固定、  
位数少。

变长格式: 基本操作  
                |  
                | 立即数 |

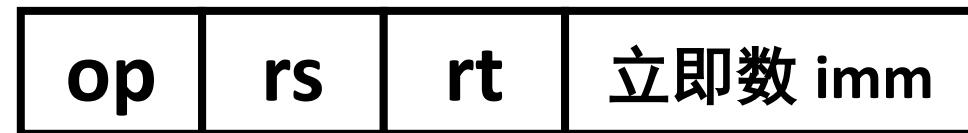
数在基本指令之后，其  
长度可与指令等长

用来提供偏移量、常数、设置初值等。





# 数据寻址-立即寻址



指令



addi \$rt, \$rs, 5

指令功能：  $\$rt \leftarrow \$rs + imm$ (符号扩展)

取指令后，直接截取指令中的低16位代码，  
就能立即得到真值为5的操作数。

因此，得到“5”的方式，就是立即寻址。





# 数据寻址-直接寻址

指令中直接给出操作数的地址码。

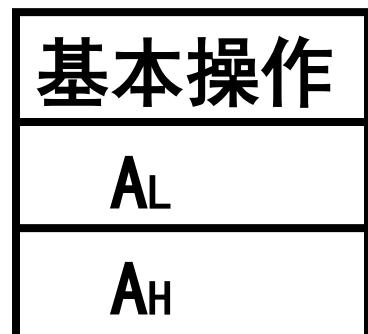
- 存储单元地址 (数在M中)
- 寄存器编号 (数在R中)

## ● 主存直接寻址(绝对寻址)

{ 定长格式  
变长格式



A的位数有限，  
限制访存范围



}

A的位数可等于指令字长，  
覆盖整个存储空间

指令



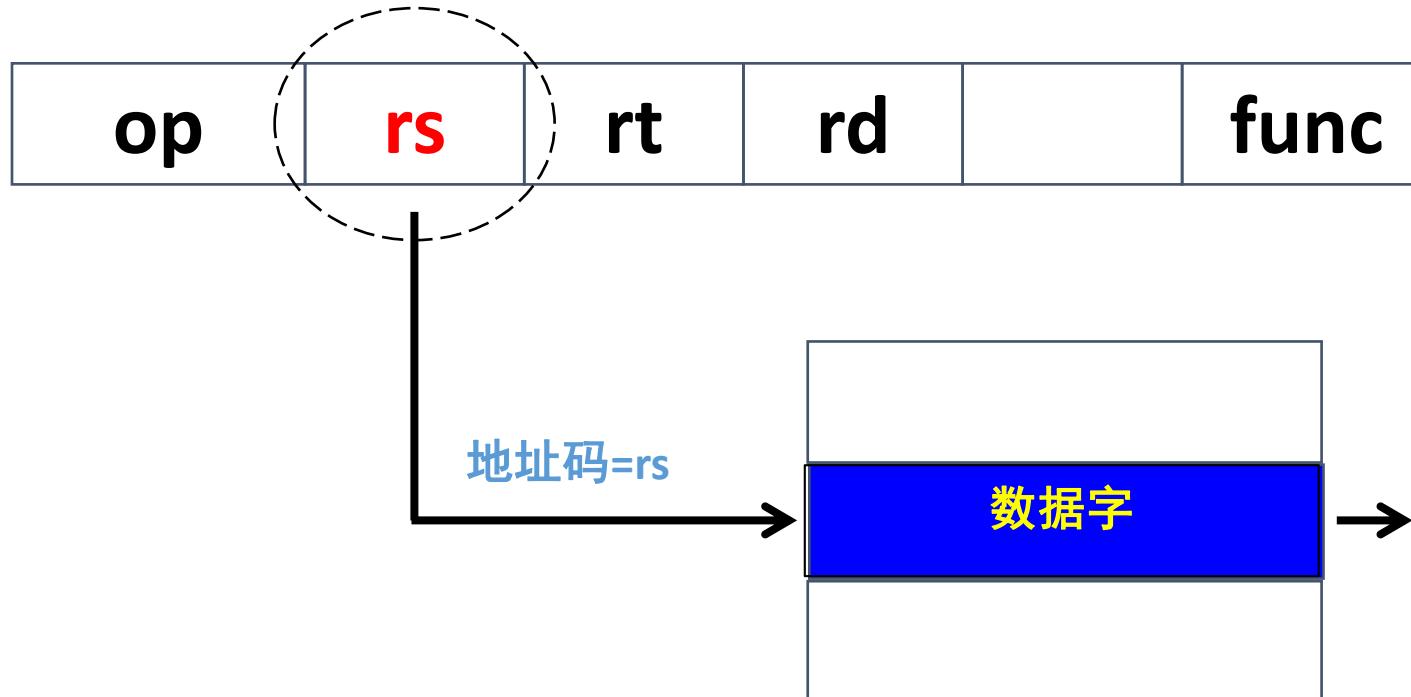
主存

操作数



# 数据寻址-直接寻址

- 寄存器直接寻址(Register Addressing)
  - 针对操作数在寄存器中的情况





# 数据寻址-间接寻址

指令给出操作数的间接地址。

- 存储单元地址 (数在M中)
- 寄存器编号 (数在R中)

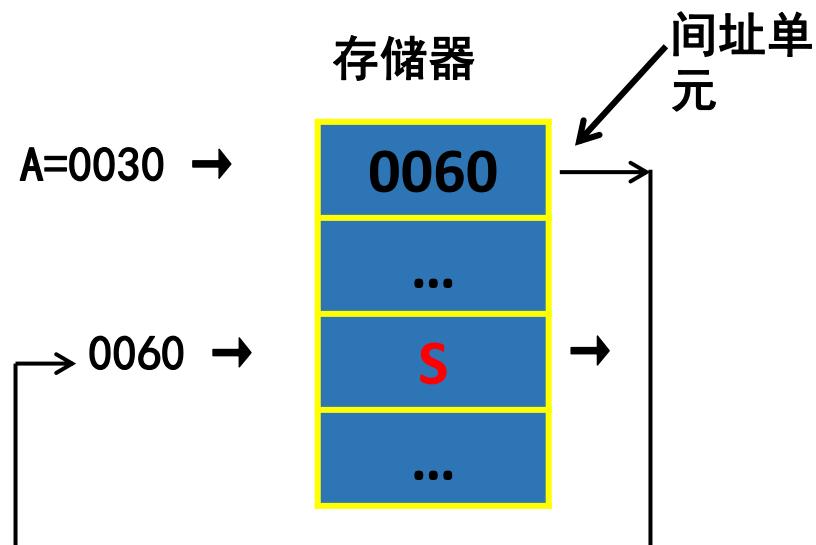
一般只在CISC中使用， RISC中一般不用；

- 主存间接寻址



操作数：  $S = ((A))$

1次间址





# 数据寻址-间接寻址

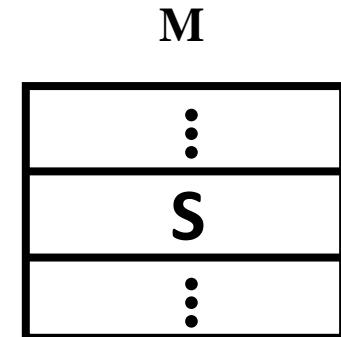
- 寄存器间接寻址



R=02

0040

0040→



$S = ((R))$  R的地址位数少，R可提供全字长地址码；  
修改R内容比修改M更快。

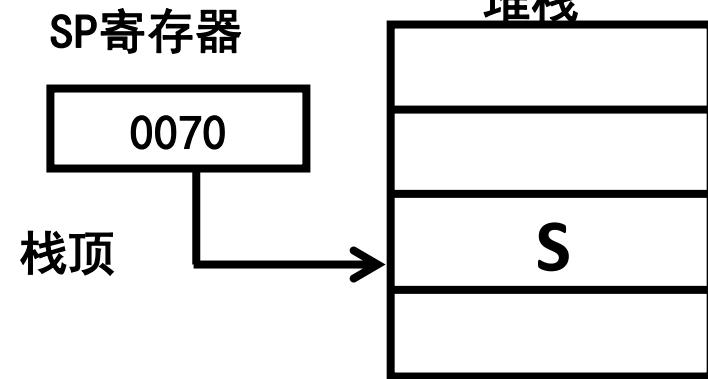
指针不变(由指令指定)，指针内容可变，使同一指令  
可指向不同存储单元，以实现程序的循环、共享，  
并提供转移地址。

- 堆栈间接寻址



厚德·博学·求是

$S = ((SP))$

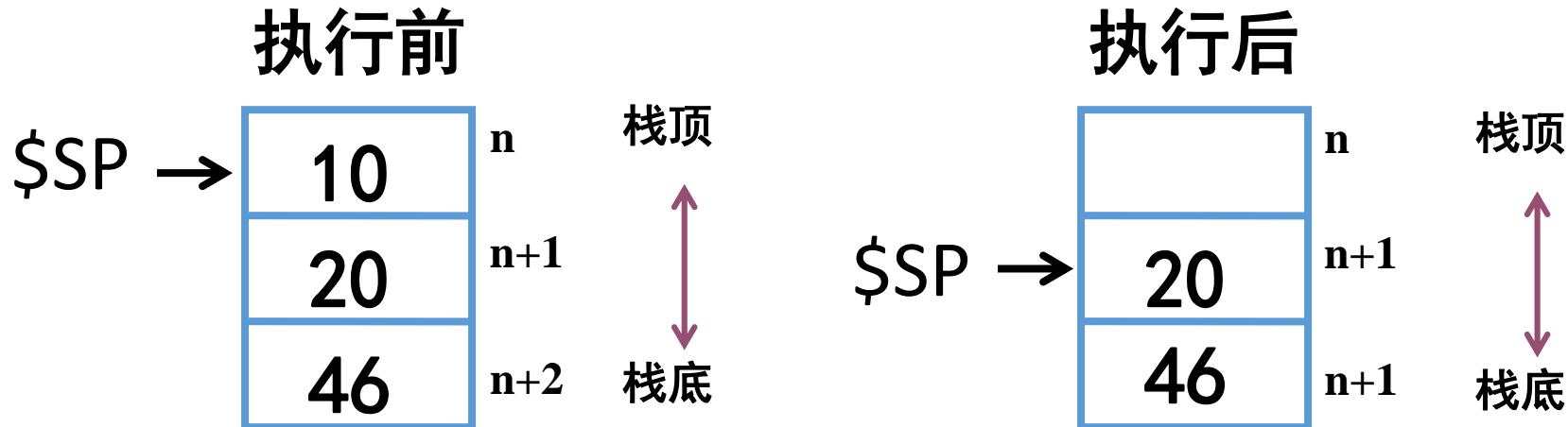




# 数据寻址-间接寻址

[例] POP (\$P)      LW \$t0, 0(\$SP)

SP既可出现在指令中，也可由操作码隐含约定



堆栈有三种方式（向下，向上，栈顶固定）



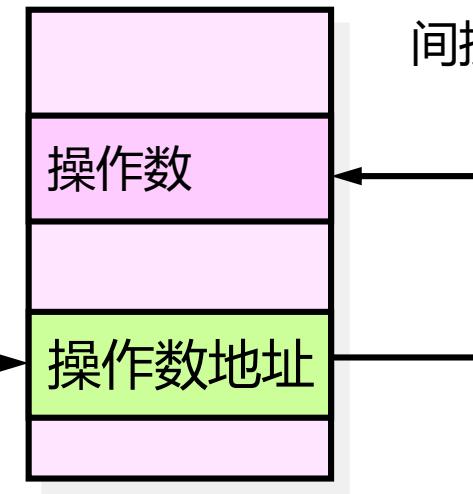


# 数据寻址-间接寻址

指令



主存



间接寻址

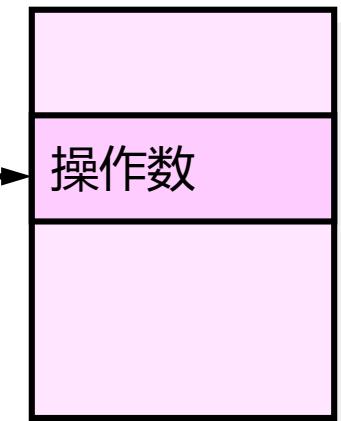
指令



寄存器组



主存



寄存器间接寻址

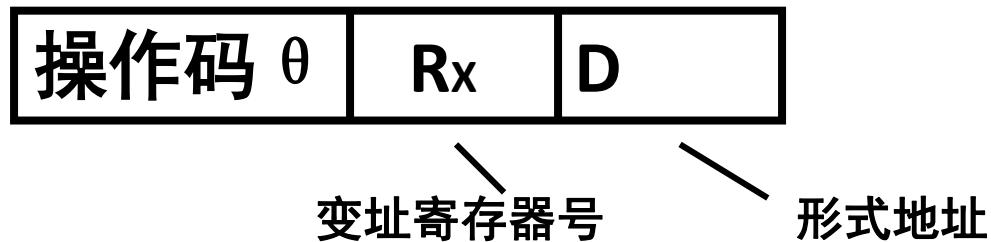




# 数据寻址-变址寻址

指令给出一个寄存器号和一个地址量，寄存器内容与地址量之和为有效地址。

格式



$$S = ((R_x) + D)$$

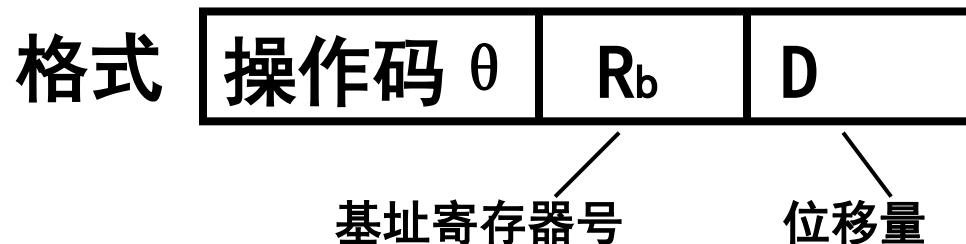
修改量                    基准地址





# 数据寻址-基址寻址

指令给出一个寄存器号和一个地址量，寄存器内容与地址量之和为有效地址（二维数组的读写）。



$$S = ((R_b) + D)$$

基准地址      相对于基址的位移





# 数据寻址-基址寻址

※ 变址与基址的区别：

有效地址=寄存器内容(R)+指令中的立即数D

变址寻址： 指令提供基准量， 寄存器提供偏移量；

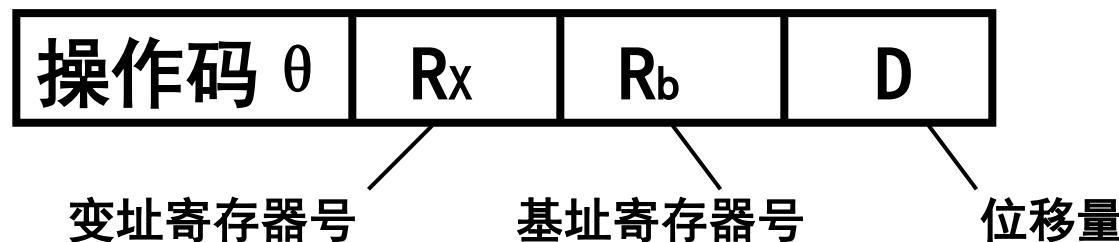
基址寻址： 指令提供偏移量， 寄存器提供基准量；





# 数据寻址-基址寻址

指令给出两个寄存器号和一个地址量，寄存器内容与地址量之和为有效地址（处理三维数组）。



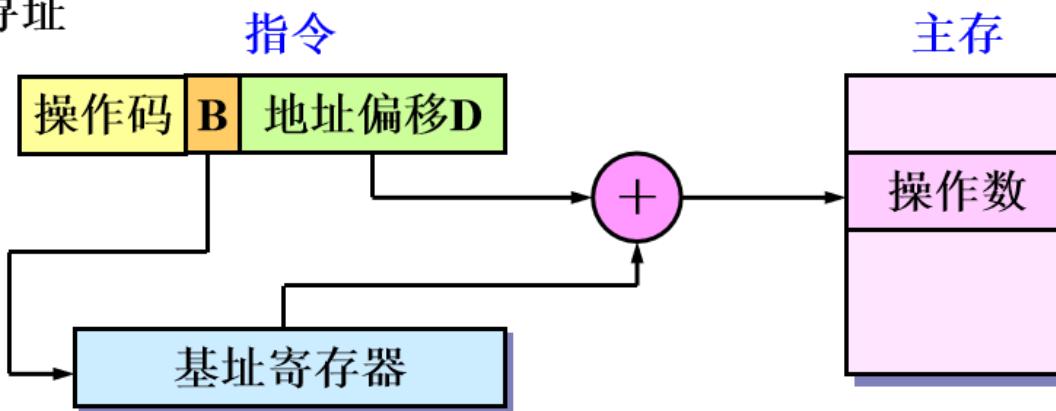
$$S = ((R_x) + (R_b) + D)$$

便于处理三维数组。

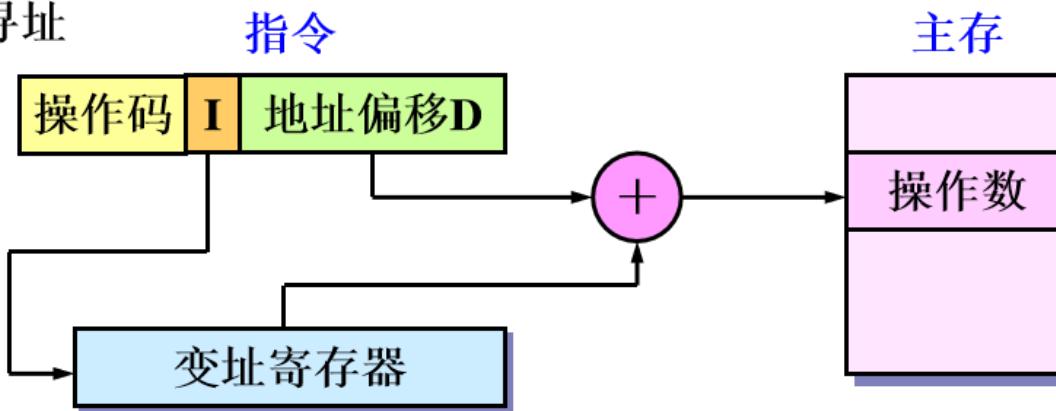


# 数据寻址-基址寻址

基址寻址



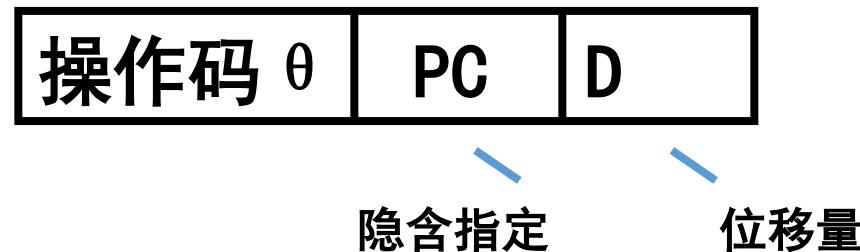
变址寻址





# 数据寻址-PC相对寻址

指令给出偏移量，**PC当前值与偏移量相加得到有效地址。**

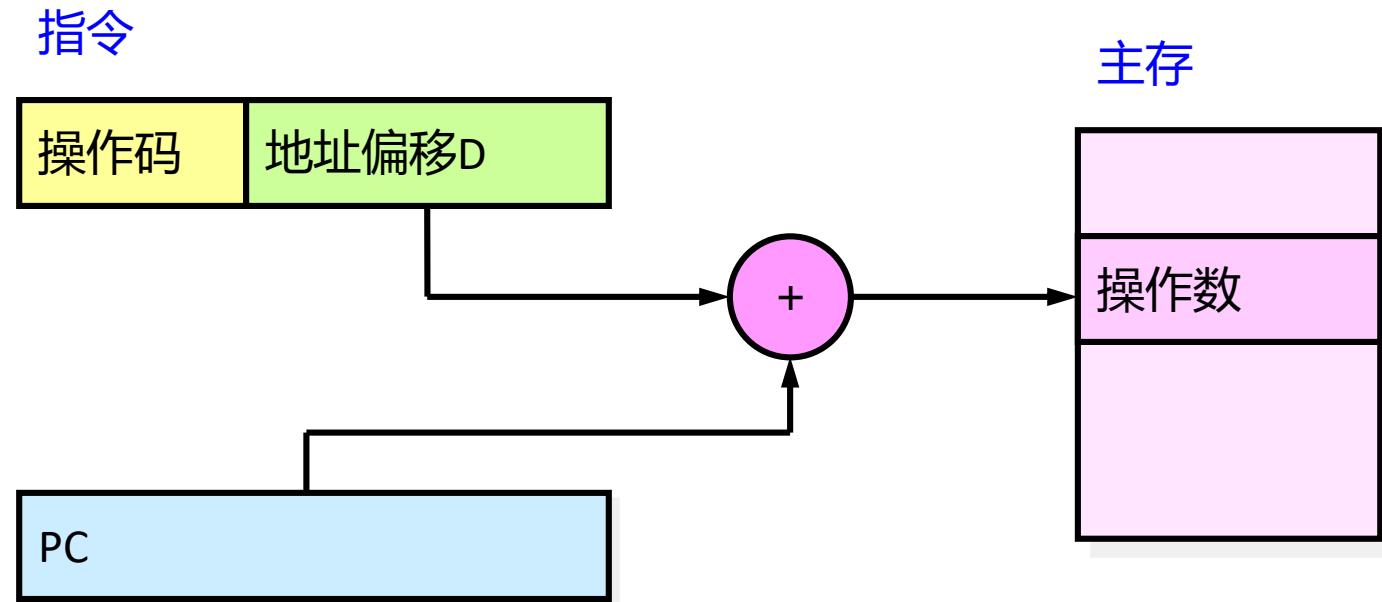


$$S = ((PC) + D)$$

是一种特殊的基址寻址方式  
有效地址相对于PC浮动, 编程方便。



# 数据寻址-PC相对寻址



“与地址无关的程序设计”





立即寻址	$S=D$	快, 便于程序设计, 赋初值, 初值大小受限
寄存器寻址	$S=R$	快, 便于程序设计, 不能访问主存
直接寻址	$E=D$	慢, 便于程序设计, 提供访存, 范围受限
间接寻址	$E=(D)$	很慢, 解决直接寻址访存范围受限的问题
寄存器间接	$E=(R)$	慢, 便于程序设计, 提供访存, 范围增大
相对寻址	$E=(PC)+D$	慢, 提供访存, 不能在循环中使用
变址寻址	$E=(X)+D$	慢, 便于程序设计, 提供访存, 可在循环中用
基址寻址	$E=(B)+D$	慢, 提供更大的范围的访存能力, 不能在循环中用

变址寻址方式非常适合于循环问题, 原因在于指令的偏移量(形式地址)保持不变, 使得执行循环时, 只需要改变IX的内容即可。

假若使用基址寻址的方式, 意味着循环过程中不断需要新的偏移量, 也就是需要更多的指令字加以控制。

而变址寻址只需要一条指令即可完成相关操作, 可以大量缩短指令编码的长度, 提高指令字的可用性。

使用寻址方式的好处: 有利于缩短指令字长、方便程序设计、扩展访存空间!

(寄存器寻址)

(变址寻址)

(基址寻址)

S=操作数  
E=有效地址  
D=地址字段值  
R=寄存器  
 $(x)=x$ 中的内容

### 各种数据寻址方式获得数据的速度(由快到慢):

- (1) 立即数寻址: 直接获得;
- (2) 寄存器寻址: 存取一次寄存器;
- (3) 直接寻址: 存取一次主存;
- (4) 寄存器间接寻址: 存取一次寄存器+一次主存;
- (5) 变址寻址(基址寻址、相对寻址): 存取一次寄存器+加法操作+存取一次主存;
- (6) 一级间接寻址: 存取两次主存;
- (7) 多级( $n$ )间接寻址: 存取 $n+1$ 级主存;



根据操作数所在位置，指出其寻址方式：

操作数在寄存器中，为[填空1] 寻址方式；

操作数地址在寄存器中称为[填空2] 寻址方式；

操作数在指令中称为 [填空3] 寻址方式；

操作数地址(主存)在指令中为[填空4] 寻址方式。

操作数的地址为某一寄存器中的内容与位移量之和则可以是  
[填空5]、[填空6]、[填空7] 寻址方式。

供选择的答案：直接； 寄存器； 寄存器间接； 基址； 变址；  
相对； 堆栈； 立即数。

作答

指令系统中采用不同寻址方式的主要目的是（ ）。

- A 实现存储程序和程序控制
- B 缩短指令长度，扩大寻址空间，提高编程灵活性
- C 可以直接访问外存
- D 提供扩展操作码的可能并降低指令编码难度

 提交

基址寻址的主要作用是（ ）。

- A 支持程序的动态定位
- B 支持访存地址的越界检查
- C 支持向量、数组的运算寻址
- D 支持程序在存储器中的定位和扩大寻址范围

 提交



# 指令的功能和类型

- (1) 按指令格式 PDP-11：单、双操作数指令等；
- (2) 按操作数寻址方式

IBM 370：

RR型（寄存器-寄存器）

RX型（寄存器-变址寄存器）

- (3) 按指令功能  
传送、访存、I/O、算数逻辑运算、程序控制、  
处理机控制等指令。



# 指令的功能和类型

- 数据传送类指令

源地址对应的  
存储单元  
主要包括：



目的地址对应的  
存储单元

取数指令、存数指令、数据传送(单字、成组)、  
数据交换和堆栈操作等。

主要用来实现：

寄存器之间、存储器单元之间以及寄存器-存储器  
单元的数据传送。



# 指令的功能和类型

- 输入/输出(I/O)指令



设计时需考虑：

(1) I/O指令对设备的适应性

如何用通用I/O指令实现对各种具体设备的控制？

✓ I/O指令中留有扩展余地

指令中某些字段事先不定义，需要时再约定其含义。

用于外设种类、数量不多的场合。

✓ 把设备抽象化、透明化处理（接口中设置控制/状态/数据寄存器）



# 指令的功能和类型

- 算术\逻辑运算指令

## (1) 算术运算指令

设计时需考虑操作数类型、符号、进制等，运算结束后设置CPU相应状态标志寄存器。

如基本的加法(add, addi)，减法(sub)等指令

## (2) 逻辑运算指令

如与(and)，或(or)，异或(xor)等指令

常用来对码位的设置和条件判断等操作。



# 指令的功能和类型

- 程序控制类指令 主要作用：控制指令的执行流程。

## (1) 转移指令

- 无条件转移 : 操作码+转移地址
- 条件转移 : 操作码+转移地址+转移条件
- 循环 : 转移条件为循环计数值

## (2) 转子指令与返回指令

转子：即调用，操作码+子程序入口

返回：操作码+返回地址（堆栈的顶单元中）

同一条返回指令应能提供多个不同的返回地址（条件返回），一般用堆栈存放返回地址。





# 指令的功能和类型

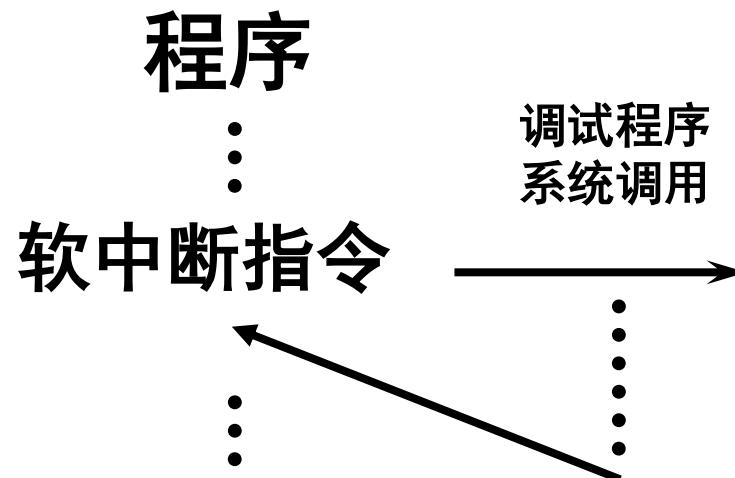
- 程序控制类指令 主要作用：控制指令的执行流程。

(3) 软中断指令 早期主要用于程序的调试。

现在常常用于系统功能调用。

常以INT n的指令形式出现在程序中。

↓  
n表示不同  
功能号





# 指令的功能和类型

- **控制处理机的专用指令**

如CPU状态字标志位的清楚、修改，空操作指令NOP、暂停HLT、等待WAIT、总线锁定LOCK等。

- **面向操作系统的指令**

提供给操作系统专用，如访问系统寄存器、检查保护属性、存储管理等。



# CISC vs RISC

(按照指令复杂程度进行划分)

## 一、RISC 的产生和发展

RISC (Reduced Instruction Set Computer)

CISC (Complex Instruction Set Computer)

80 — 20 规律

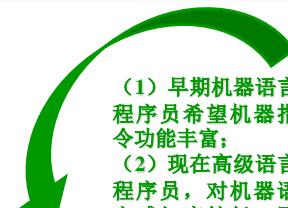
— RISC技术

- 典型程序中 80% 的语句仅仅使用处理机中 20% 的指令
- 执行频度高的简单指令，因复杂指令的存在，执行速度无法提高
- ? 能否用 20% 的简单指令组合不常用的 80% 的指令功能

RISC 的目标绝不是简单的缩减指令系统，而是使处理器的结构更简单，更合理，具有更高的性能和执行效率（高效能），并降低处理器的开发成本，非常适用于移动设备。

厚德·博学·求是

面向CISC指令集的CPU控制器设计难度越来越高、成本越来越大、开发周期越来越长。



- (1) 早期机器语言程序员希望机器指令功能丰富；
- (2) 现在高级语言程序员，对机器语言感知度较低，因为编译器优化技术的提高。

**CISC** Complex Instruction Set Computer,  
复杂指令集计算机

- 指令类型多、格式多、功能复杂，寻址方式多
- 多数指令需要多个CPU周期才能完成几乎都可以访存
- 寄存器数目较少，主要采用微程序控制方式
- 目标代码难以优化
- 各种指令的使用频率相差很大——“二-八”规律
- Intel X86系列，AMD X-86、X86-64系列等

**RISC** Reduced Instruction Set Computer,  
精简指令集计算机

- 指令类型少、格式统一、功能简单，寻址方式简单
- 各条指令的执行时间都差不多，只有极少数指令可以访存
- 寄存器数量较多
- 采用“超标量和超流水线”，CPU并行处理能力提高
- MIPS、IBM Power PC、SUN SPARC、ARM等

- (1) 硬布线实现；
- (2) 编译器优化；
- (3) 流水线；
- (4) 多寄存器；

- 相互融合：
  - (1) CISC借鉴RISC的流水技术，用于微程序优化，借以降低开发复杂度。
  - (2) RISC也在增加复杂的指令集（应用频率高），借以提高性能。



RISC的主要问题在于指令集简单，因此在处理一些较为复杂的应用时，存储器需要读入的指令总数耗时更多，部分场合下性能表现不理想也是RISC的硬伤。

基址寄存器的内容为2000H (H表示十六进制) , 变址寄存器内容为03A0H, 指令的地址码部分是3FH, 当前正在执行的指令所在地址为2B00H。

(1) 请求出变址编址 (考虑基址) 和相对编址两种情况的访存有效地址 (即实际地址) 。

(2) 设变址编址用于取数指令, 相对编址用于转移指令, 存储器内存放的内容如下:

地址 内容

003FH 2300H

2000H 2400H

203FH 2500H

233FH 2600H

23A0H 2700H

23DFH 2800H

2B00H 063FH

请写出从存储器中所取的数据以及转移地址。

(3) 若采用直接编址, 请写出从存储器取出的数据。

作答



基址寄存器的内容为2000H (H表示十六进制) , 变址寄存器内容为03A0H, 指令的地址码部分是3FH, 当前正在执行的指令所在地址为2B00H。

- (1) 请求出变址编址 (考虑基址) 和相对编址两种情况的访存有效地址 (即实际地址) 。
- (2) 设变址编址用于取数指令, 相对编址用于转移指令, 存储器内存放的内容如下:

地址 内容

003FH	2300H
2000H	2400H
203FH	2500H
233FH	2600H
23A0H	2700H
23DFH	2800H
2B00H	063FH

请写出从存储器中所取的数据以及转移地址。

- (3) 若采用直接编址, 请写出从存储器取出的数据。

(1) 变址编址访存有效地址为:  $2000H + 03A0H + 3FH = 23DFH$

相对编址访存有效地址为:  $2B00H + 3FH = 2B3FH$

(2) 取出数据为2800H, 转移地址为2B3FH

(3) 若机内设有基址寄存器, 所取数据为2500H

若机内没有基址寄存器, 所取数据为2300H



厚德·博学·求是



# 有问题欢迎随时跟我讨论

办公地点：西校区信息馆504

邮 箱：[gddu@ysu.edu.cn](mailto:gddu@ysu.edu.cn)

