# 3.2 指令系统

设计CPU的一般过程：

指令系统→ 数据通路→ 控制器→ CPU定型

本节主要分析：√指令格式

√指令涉及的寻址方式

√向用户的指令类型

– *基于MIPS32架构*

⊙指令：*instruction*，计算机执行某类操作的信息的集合，是CPU工作的主要依据。

⊙指令集：*instruction set*，处理器能执行的全体指令的集合（CISC、RISC）

- √ *决定了计算机的硬件功能*
- √ *计算机中软硬件的分界面*

⊙ 指令字：用来表示指令的一组二进制代码。

⊙ 指令字长：指令中包含的二进制代码位数

⊙ 机器字长：计算机能够直接处理的二进制数据的位数 = 寄存器的宽度。

# 3.2.1 指令集类型

## 1、CISC（Complex Instruction Set Computing）

**复杂指令集计算**

早期计算机部件昂贵、速度慢，为了扩展硬件功能，不得不将更多更复杂指令加入到指令系统，以提高计算机的处理能力 → 复杂指令集

## 2、RISC（Reduced Instruction Set Computing）

**精简指令集计算**

随着半导体技术进步，80年代开始逐渐直接通过硬件方式，而不是扩充指令来实现复杂功能，指令规模逐渐缩小、指令进一步简化 → 精简指令集

# ※CISC→RISC的技术背景

## ①2—8规律

CISC中的不同指令使用频率悬殊：

简单指令(约占20%的)约占80%的使用频率，复杂指令(约占80%)只占大约20%的使用频率。

## ②不利于VLSI工艺

为实现大量的复杂指令，控制逻辑极不规整，给VLSI工艺造成很大困难。

## ③ 主存技术的发展

一般通过保存在控制存储器中的微程序来实现复杂指令，70年代后期开始用DRAM做主存，使主存与控制存储器的速度相当，因此很多复杂指令不必再用微程序来实现，可用简单指令构成的子程序实现等效功能。

## →CISC的特点：

①指令数量多；

②指令长度可以不固定，指令格式和寻址方式多样；

③很多指令会涉及存储器读写操作，指令周期长；

④一般在通用处理器中使用；

## →RISC的特点：

①指令数量少；

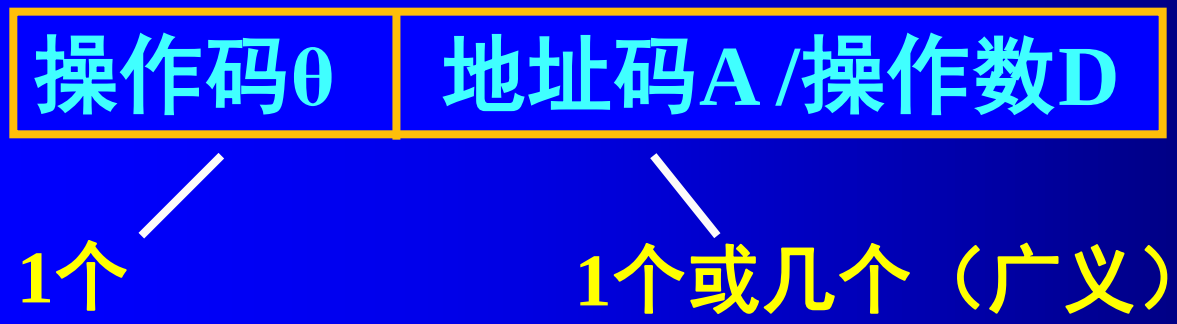
②指令长度固定，指令格式和寻址方式种类也少；

③一般只有少量指令（如取数/存数）才会读写存储器，其余指令只涉及CPU内部寄存器，指令周期短；

④一般在高端服务器CPU中使用；

# 3.2.2 指令格式

## 指令的基本格式



## 1. 指令字长

- 定长指令格式　规整、便于控制
- 变长指令格式　合理利用存储空间、提高取指令的效率，如超长指令集

## 2. 操作码结构

(1) 定长操作码

各指令θ 的位置、位数固定相同。

(2) 扩展操作码

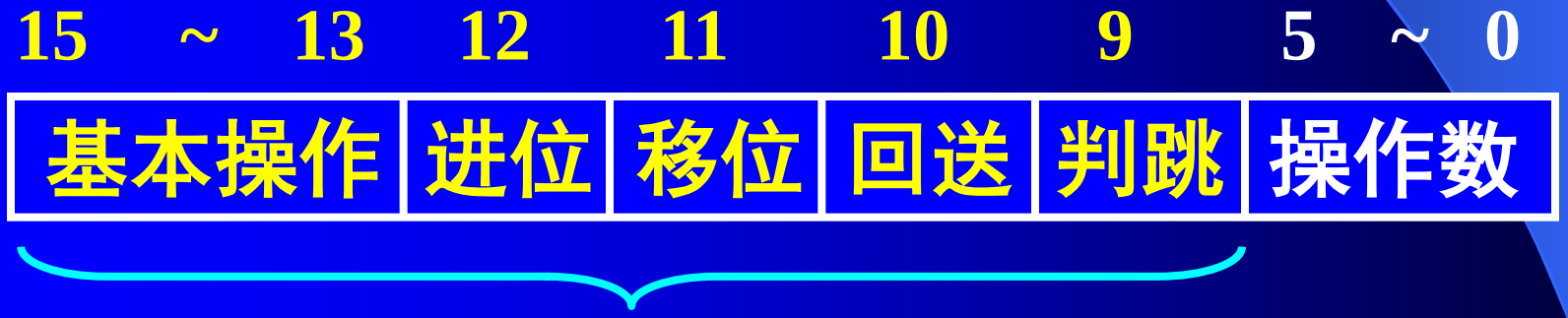
各指令θ 的位置、位数不固定，根据需要变化（设置扩展标志）。

## (3) 复合型操作码

**基本特征：**

**操作码分为几段，每段表示一种二级操作。**

[例]某机的算术逻辑运算指令格式



## 3. 地址结构

指令中提供的地址 < 地址偏移量/立即数
寄存器编号

（1）指令中提供地址的方式

√ 显式地址方式

→ 指令中明确指明地址码（直接/间接给出）

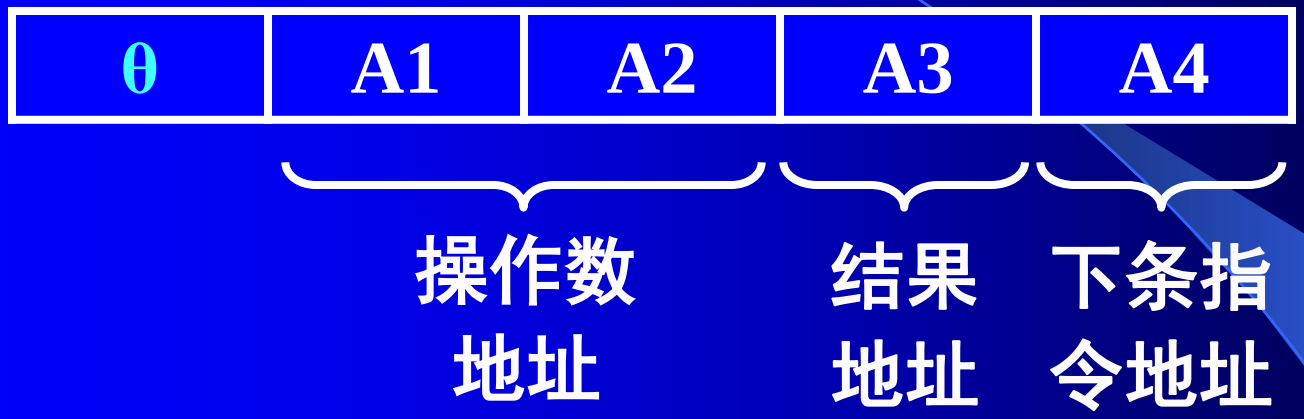
√ 隐式地址方式

→ 地址码隐含约定，不在指令中出现。

使用隐式地址，可以减少指令中的地址数量，从而简化地址结构。

## （2） 常见的地址结构类型

✓ 四地址结构指令



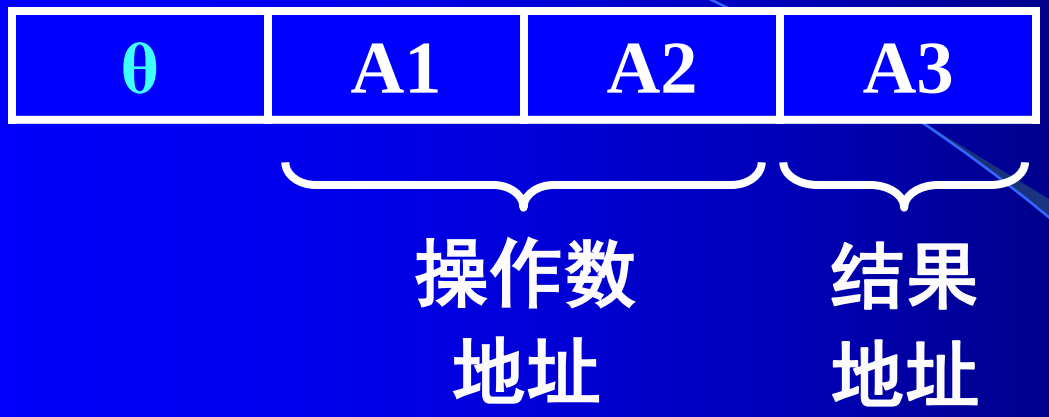

功能：（A1）θ（A2）→A3

A4→PC

一般用PC寄存器指示下条指令的地址。

四地址结构指令在RISC中很少会使用。

## ✓三地址结构指令




**功能：**

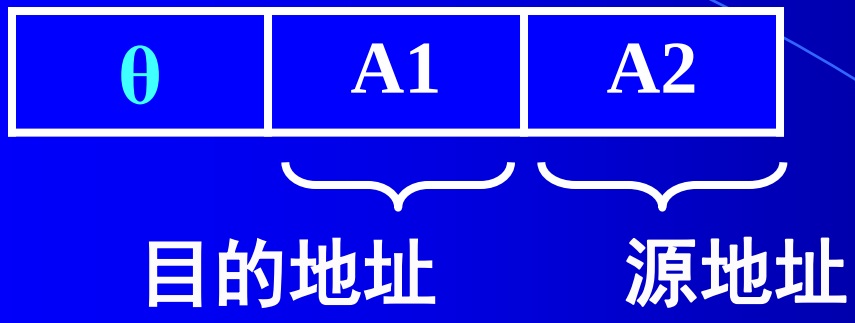
(A1) θ (A2) → A3

(PC) + → PC　　如：ADD rd, rs, rt

↓

自动修改PC的内容，使PC指向下一条指令

## ✓二地址结构指令




**功能：**

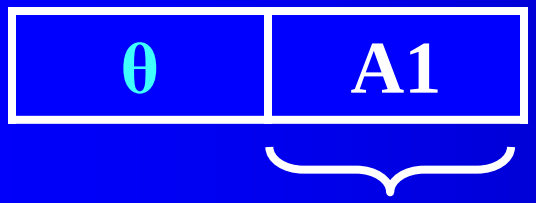
(A1) θ (A2) → A1

(PC)+ → PC　　如：ADD R1, R0

↓

自动修改PC的内容，使PC指向下一条指令

## ✓ 一地址结构指令




隐含约定

双操作数：A1 U $[PC]_H$→PC　如：J addr

单操作数：

θ(A1) → A1

$(PC)^+$ → PC　如：INC R0

自动修改PC的内容，使PC指向下一条指令

## ✓ 零地址结构指令

| θ |
|---|

功能：

①用于处理机的特殊控制（如HLT, NOP）。

②针对隐含约定的寄存器，如返回指令：

RST

↓

隐含操作：(SP)→PC；$SP^{+}$ →SP；

把堆栈栈顶单元保存的返回地址打入PC。

## 4. 指令可能会涉及到的操作数类型

① 地址码数据

寄存器编号或者存储器地址，无符号整数。

② 数值型数据

定点数、浮点数等，一般用补码表示。

③ 字符型数据

通常表示为ASCII码/汉字内码格式。

④ 逻辑型数据

常规二进制代码，不具有数值含义。

# 5. MIPS32架构的指令格式

- RISC
- 指令字长为：32位
- 寄存器数量：32个

| 指令类型 | 指令长度（32位定长） | | | | | |
|---|---|---|---|---|---|---|
| | 31 ~ 26 | 25~21 | 20~16 | 25~11 | 10 ~ 6 | 5 ~ 0 |
| R型 | op(6) | rs(5) | rt(5) | rd(5) | shamt | func |
| I型 | op(6) | rs(5) | rt(5) | address/imm (16) | | |
| J型 | op(6) | address (26) | | | | |

# 3.2.3 指令中的寻址方式

※指形成操作数地址或寻找操作数的方式；

※1条指令，可能会涉及多种寻址方式；

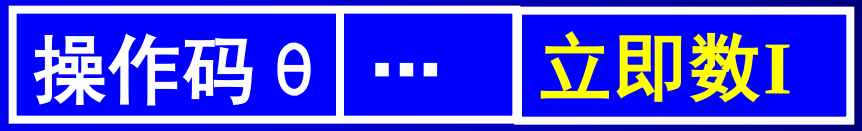
## 1、常见寻址方式

### ①立即寻址(Immediate addressing)

指令中直接包含了操作数。

数在指令中，其长度固定、位数少。

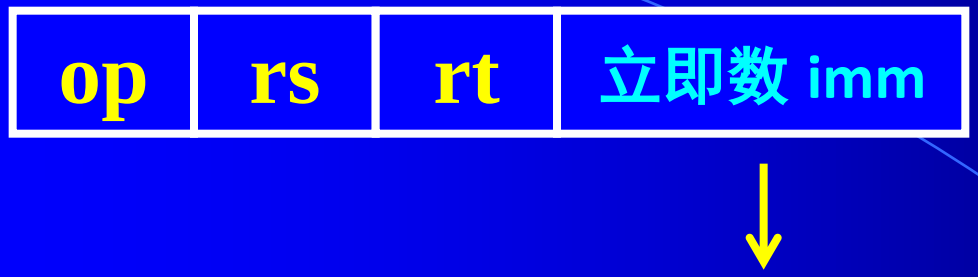
定长格式：




变长格式：


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。




addi $rt, $rs, 5

指令功能： $rt ← $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。

## ②直接寻址

**指令中直接给出操作数的地址码。**

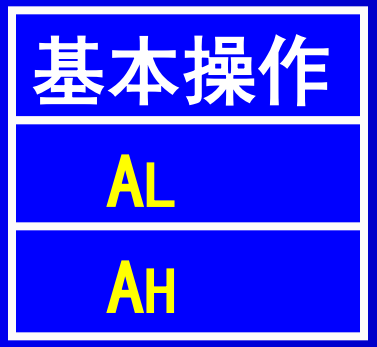
- 存储单元地址 （数在M中）
- 寄存器编号 （数在R中）

- 主存直接寻址（绝对寻址）

定长格式


A的位数有限，
限制访存范围

变长格式




A的位数可等于指令字长，
覆盖整个存储空间

操作数：S =（D）

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

## ③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M中）

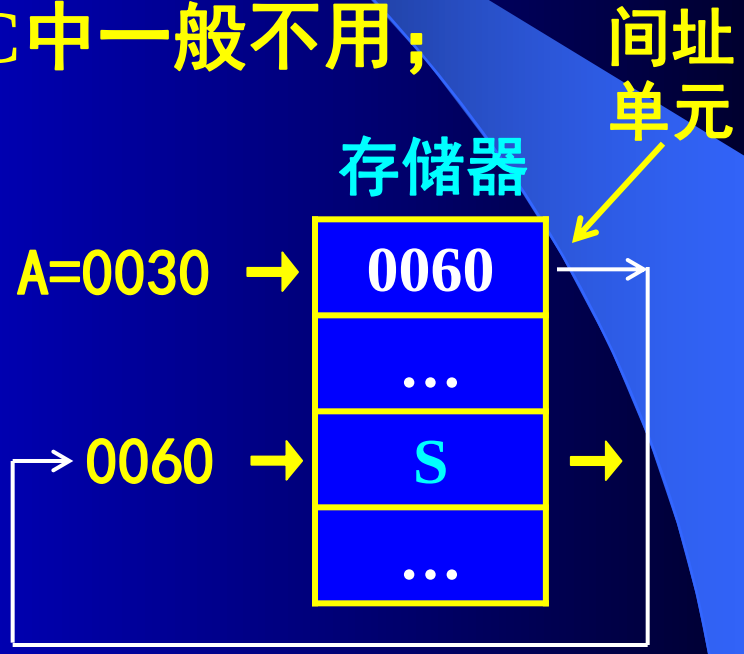
**一般只在CISC中使用，RISC中一般不用；**

- 主存间接寻址

| 操作码θ | 间接地址A |
|---|---|

操作数：S =((A))

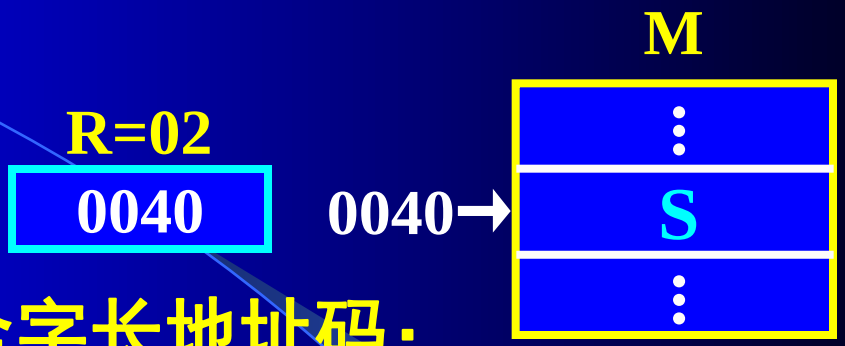
1次间址



- 寄存器间接寻址

| 操作码θ | 寄存器号R |
|---|---|

S =((R))



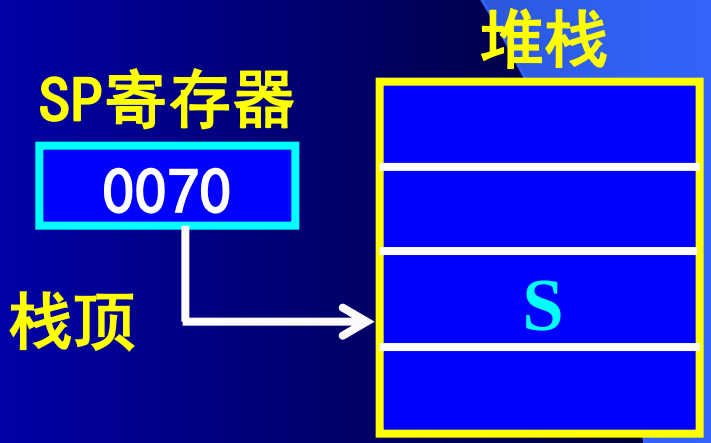

R的地址位数少，R可提供全字长地址码；
修改R内容比修改M更快。

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

| 操作码θ | 堆栈指针SP |
|---|---|

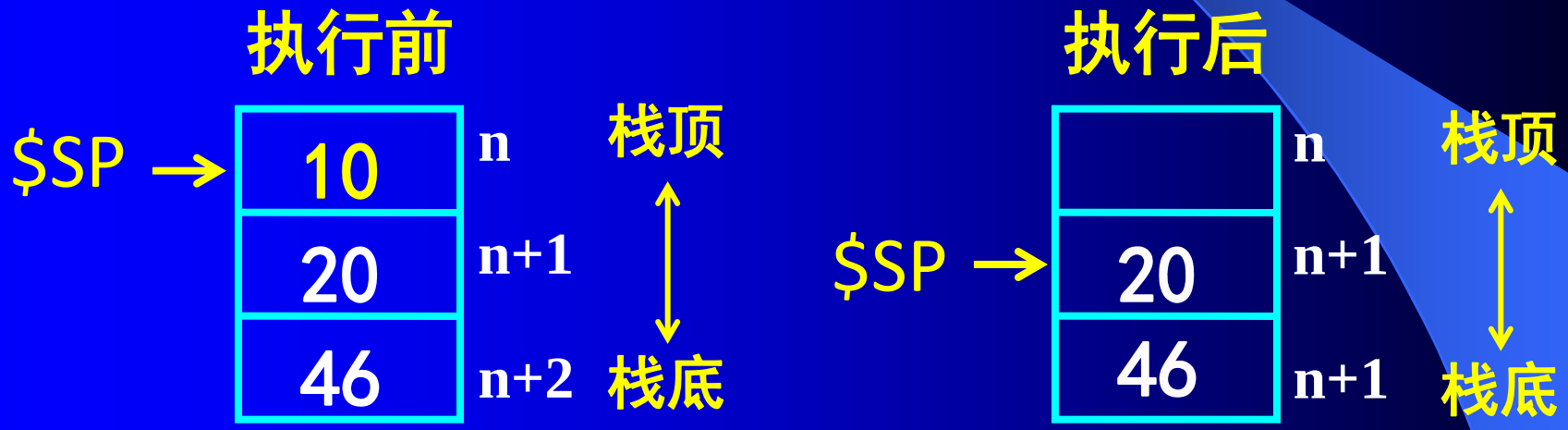
S =((SP))



［例］ POP (SP)

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）

# （4）变址、基址寻址及其变化

## ● 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


变址寄存器号　　形式地址

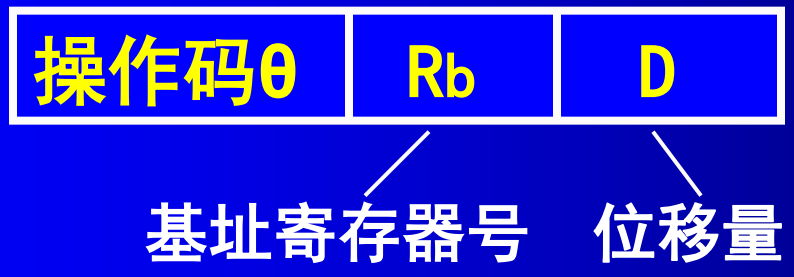
S =((RX)+ D)

修改量　　基准地址

## ● 基址寻址(base addressing)

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）。

格式




S =((Rb)+ D)

基准地址　相对于基址的位移

## ※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：

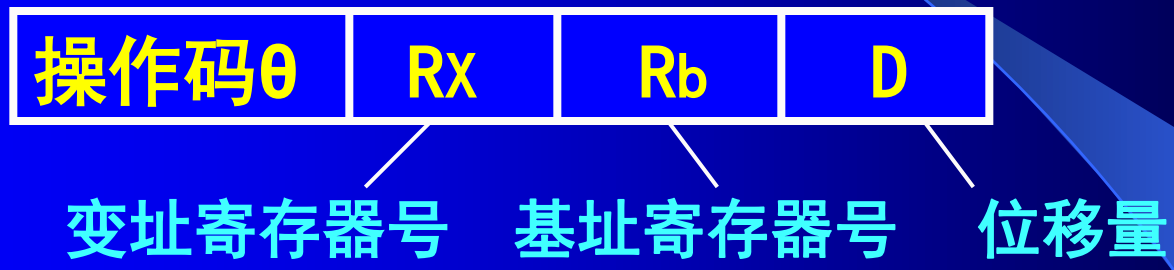
指令提供基准量，寄存器提供偏移量；

基址寻址：

指令提供偏移量，寄存器提供基准量；

## ● 基址+变址

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）。

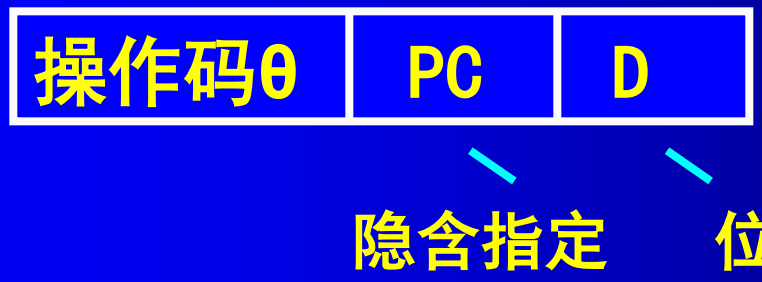



S =((Rx)+(Rb)+ D)

便于处理三维数组。

## （5）PC相对寻址(PC-relative addressing)

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)

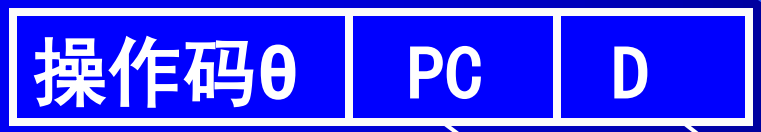
是一种特殊的基址寻址方式

有效地址相对于PC浮动，编程方便。

# （6）页面寻址（伪直接寻址）

指令给出位移量，PC的高位部分与位移量拼接，形成有效地址。

格式

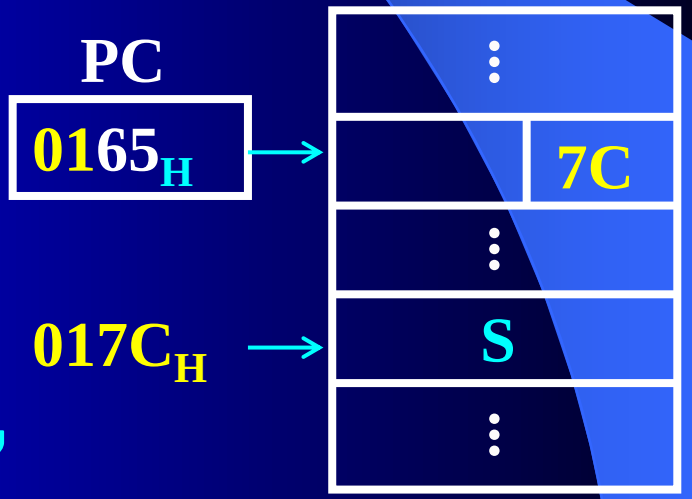



隐含指定　　位移量

$S = ((PC)_H , D)$

页号　　页内地址

[例]M为64KB，划分为256页，每页256B，按字节编址。




用于页式存储系统。

寻址速度快，适于组织程序模块，有效利用存储空间。

## 2. 指令中的寻址方式约定

### (1) 操作码可隐含说明不同寻址方式

[例] 某机指令操作码的最高两位

- 00：RR型指令，寄存器-寄存器寻址
- 01：RX型指令，寄存器-变址寻址
- 10：SI型指令，基址-立即寻址
- 11：SS型指令，基址-基址寻址

MIPS指令，一般都是采用这种方式。

## （2）指令中可设置寻址方式字段

[例]某机指令的每个地址字段中各设置一个3位的寻址方式说明字段。

## 3.2.4 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。

## ● 数据传送类指令

源地址对应的存储单元 —数 据→ 目的地址对应的存储单元

**主要包括:**

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。

设计传送指令时需要考虑：

（1）规定传送范围

（2）指明传送单位

（3）设置寻址方式

MIPS32中

只有lw和sw这两个指令可以访问存储器

- 输入/输出(I/O)指令

主机 

外部设备

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）

## （2）主机对外设的寻址方式

寻找I/O接口中的寄存器的方式。

↓

I/O端口

该类寄存器的编号，也称为I/O端口的地址（简称端口地址）

如何为I/O端口分配地址？

- ✓单独编址
- ✓统一编址

## （3）I/O指令的设计思路

**思路1：设计专用的I/O指令　　（显式I/O指令）**

针对端口自行单独编址，用I/O指令访问I/O端口。
指令中说明I/O类型，并给出端口地址。

**思路2：用传送指令替代I/O指令（隐式I/O指令）**

针对端口和主存统一编址，用访存指令访问I/O端口，
故不必再设计专用的I/O指令。

**思路3：通过IOP进行I/O操作控制**

需设计两级I/O指令
- CPU控制IOP(如启动、停止等)
- IOP控制具体I/O操作(如保存等)

- 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。

## ● 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）

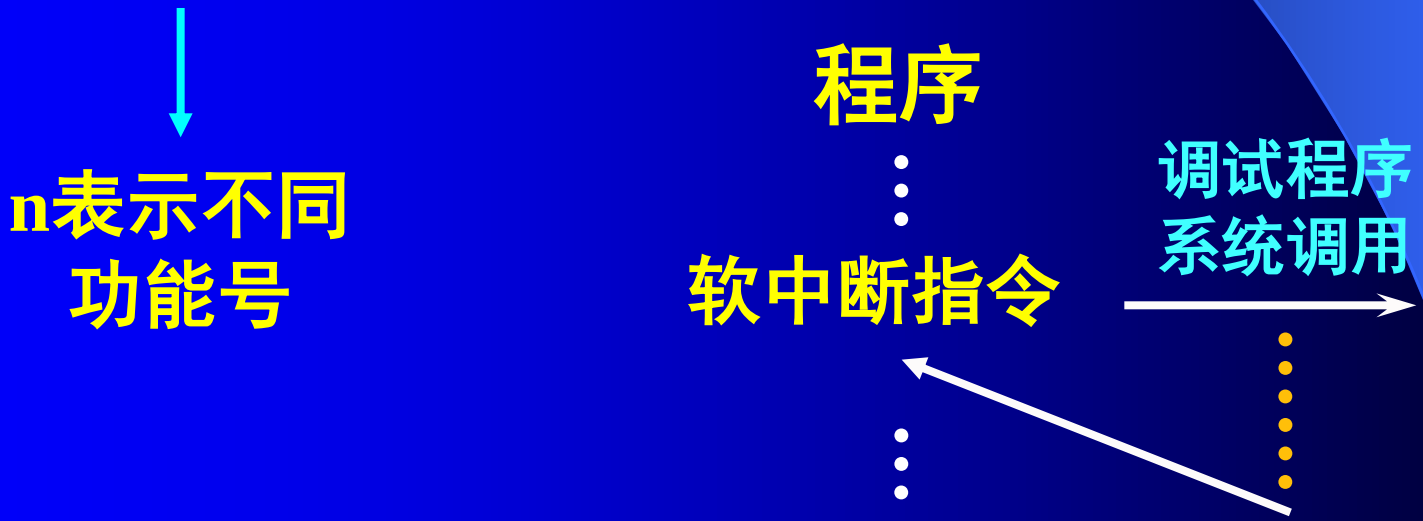
同一条返回指令应能提供多个不同的返回地址（条件返回），*一般用堆栈存放返回地址。*

## （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。



- **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。