第二章 计算机中的信息表示

- 数据信息
  - 数值型数据
  - 非数值型数据
- 控制信息 — 指令信息等

第一节 数据信息的表示

2.1.1 表示数据的大小

二进制、八进制、十六进制、二-十进制

2.1.2 表示数据的符号

原码、补码、反码

2.1.3 表示小数点

定点、浮点

1. 定点表示法

类型

无符号数　　00000000　　11111111

（0）　　（255）

定点整数

11111111 原　　01111111 原

（-127）　　（127）

10000000 补　　01111111 补

（-128）　　（127）

定点小数

1.1111111 原　　0.1111111 原

$-(1-2^{-7})$　　$(1-2^{-7})$

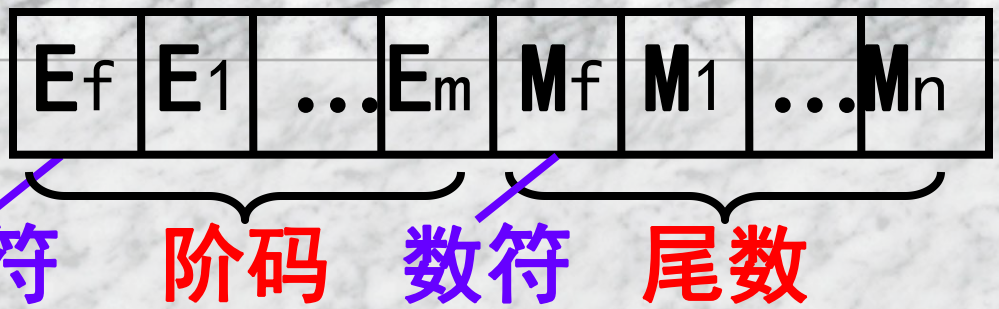
1.0000000 补　　0.1111111 补

（-1）　　$(1-2^{-7})$

2. 浮点表示法

浮点数真值：$N = \pm R^{E} \times M$

浮点数机器格式：



R：阶码底，隐含约定。

E：阶码，为定点整数，补码或移码表示。

其**位数**决定**数值范围**；**阶符**表示**数的大小**。

M：尾数，为定点小数，原码或补码表示。

其**位数**决定**数的精度**；**数符**表示**数的正负**。

尾数规格化：$1/2 \leqslant |M| < 1$ 最高有效位绝对值为1

例:若某浮点数阶码（连同一位符号位）共8位，移码表示，表示范围－128≤X≤127，则$X_{移}=2^7+X$。真值、移码、补码对应表如下：

真值X（十进制）	真值X（二进制）	$X_{移}$	$X_{补}$
－128	－10000000	00000000	10000000
－127	－01111111	00000001	10000001
…			
－1	－00000001	01111111	111111111
0	00000000	10000000	00000000
+1	00000001	10000001	00000001
…			
+127	01111111	11111111	01111111

移码的特点如下：

a. 最高位为符号位，但其取值与原码、补码正好相反。

b. 除符号位相反之外，移码的其余各位与补码相同。这是由于移码平移了2^7，而补码则平移了2^8（模值）。

c. 让X从－128逐渐增至+127，相应地$X_{移}$从00…00逐渐增至11……11，呈递增状。可见采用移码能更直观地比较正负阶码的大小，例如+1与－127之间的比较。

3. 表示范围与精度

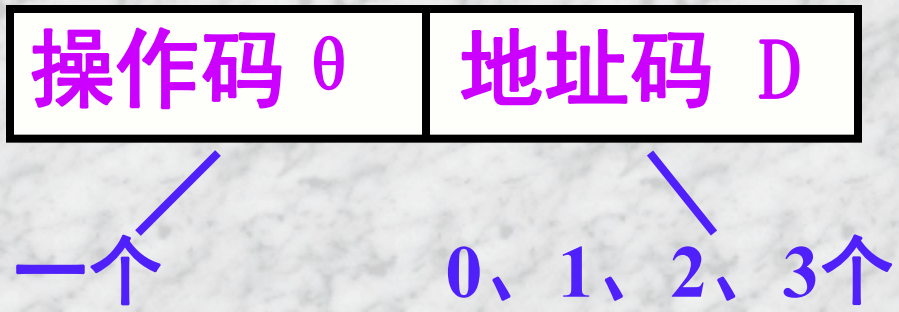
第二节　指令信息的表示

指令：指示计算机执行某类操作的信息的集合。

本节主要讨论：一般指令格式
常用寻址方式
面向用户指令类型

2.2.1 指令格式

指令基本格式



1. 指令字长

- 定长指令格式　便于控制
- 变长指令格式　合理利用存储空间

2. 操作码结构

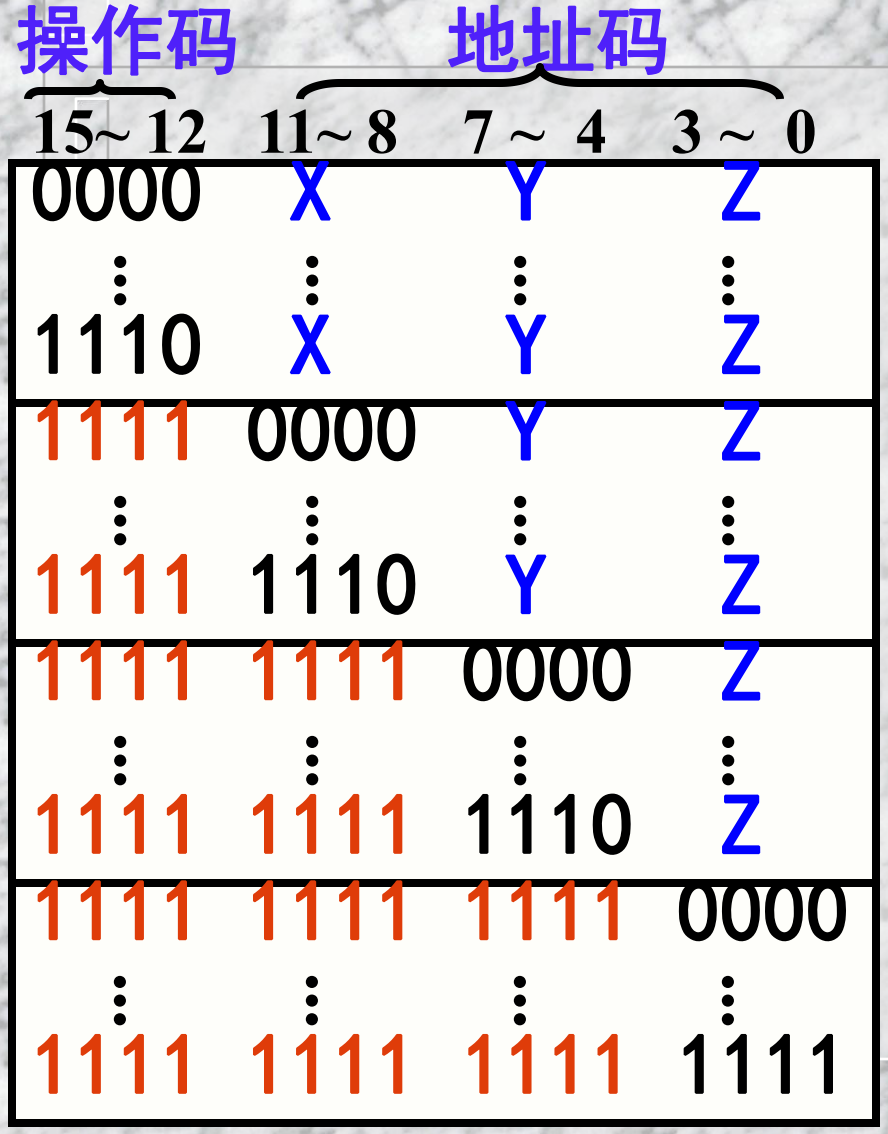
（1） 定长操作码

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。　关键在设置扩展标志。

例. 指令字长16位，可含有3、2、1或0个地址，每个地址占4位。



三地址指令　15条

二地址指令　15条

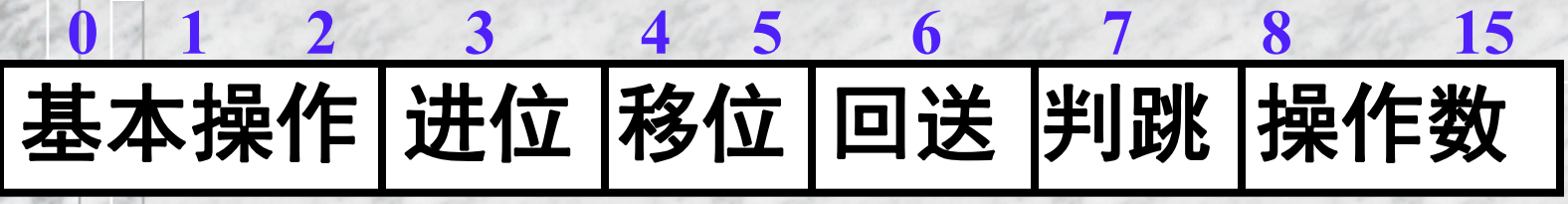
一地址指令　15条

零地址指令　16条

（3）复合型操作码

操作码分为几部分，每部分表示一种操作。

例. 某机算逻指令



3. 地址结构

指令中提供的地址 < 存储单元地址码
寄存器编号

（1）指令提供地址的方式 直接或间接给出

显地址方式:指令中明显指明地址。

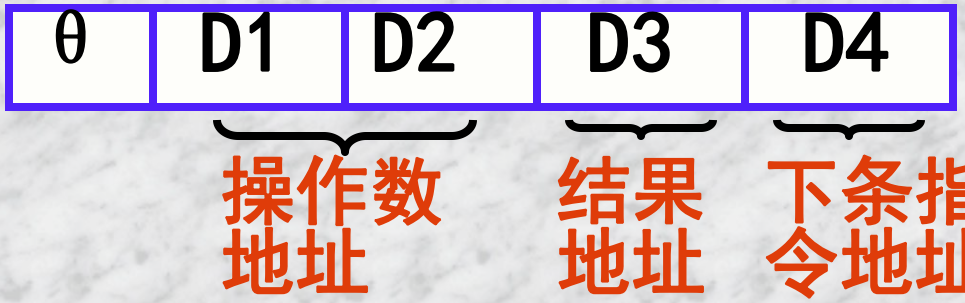
隐地址方式:地址隐含约定,不出现在指令中。

使用隐地址可以减少指令中的地址数，简化地址结构。

（2） 地址结构的简化

● 四地址结构指令

格式：



功能：(D1) θ (D2)⟶D3

(D4)⟶下条指令

用指令计数器PC指示指令地址。

●三地址结构指令

格式：

操作数地址　　结果地址

功能：(D1) θ (D2) ⟶ D3

(PC) + 1 ⟶ PC

下条指令地址

转移时，用转移地址修改PC内容。

●二地址结构指令

格式：

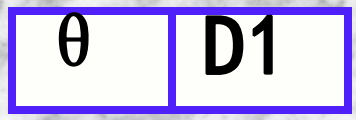
源/目的　　目的/源

功能：(D1) θ (D2) ⟶ D2/D1

(PC) + 1 ⟶ PC

●一地址结构指令

格式：



隐含约定

功能：双操作数：(D1) θ (A) ⟶ A

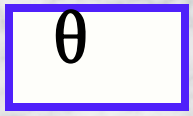
(PC) + 1 ⟶ PC

单操作数：θ (D1) ⟶ D1

(PC) + 1 ⟶ PC

●零地址结构指令

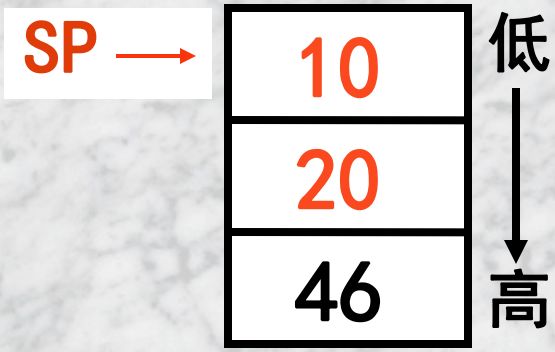
格式：



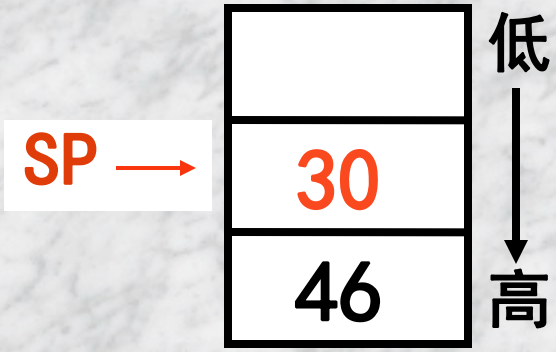
功能：用于堆栈或特殊指令操作。

例. ADD；

执行前：



执行后：



举例：

1.二地址指令

例如：MOV AX，BX

2.一地址指令

a. 隐含约定目的地的双操作数指令

例如：无符号乘法

字节乘法：OPRD×AL→AX

如：MUL DL ；DL×AL→AX

字乘法：OPRD×AX→DX：AX

如：MUL BX ；BX×AX→DX：AX

b. 只有目的操作数的单操作数指令

例如：**NEG BL ；求负**

NOT BL ；求非

3.零地址指令

a.对只有目的操作数的指令，隐含在指定寄存器内进行操作。

例如：**PUSHF ； FLAGS→入栈**
POPF ； FLAGS→出栈
LAHF ； FLAGS的低8位→AH
SAHF ； AH→FLAGS的低8位

b.不需要操作数的指令。

例如：NOP　；空操作指令
　　HLT　；停机指令

2.2.2 寻址方式

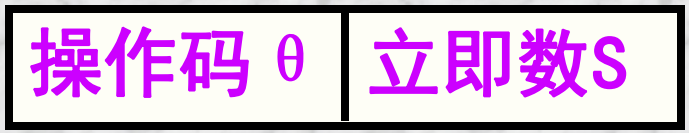
是指寻找操作数地址或操作数的方式。

1. 常见寻址方式

（1） 立即寻址

指令直接给出操作数。

定长格式：



数在指令中，其长度固定、有限。

变长格式：

数在基本指令之后，其长度可变。

用来提供常数、设置初值等。

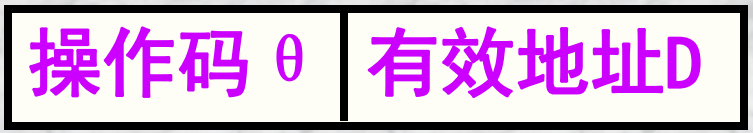
（2） 直接寻址

指令直接给出操作数地址。

- 存储单元号(数在M中)
- 寄存器号 (数在R中)

● 存储器直接寻址 （直接寻址）

定长格式



D的位数有限，
限制访存范围

变长格式

D的位数可覆盖
整个存储空间

S = (D)

● 寄存器直接寻址(寄存器寻址)

格式 | 操作码 θ | 寄存器号R |

R所占位数少；
访问R比访问M快

S = (R)

用于访问固定的存储单元或寄存器。

(3) 间接寻址

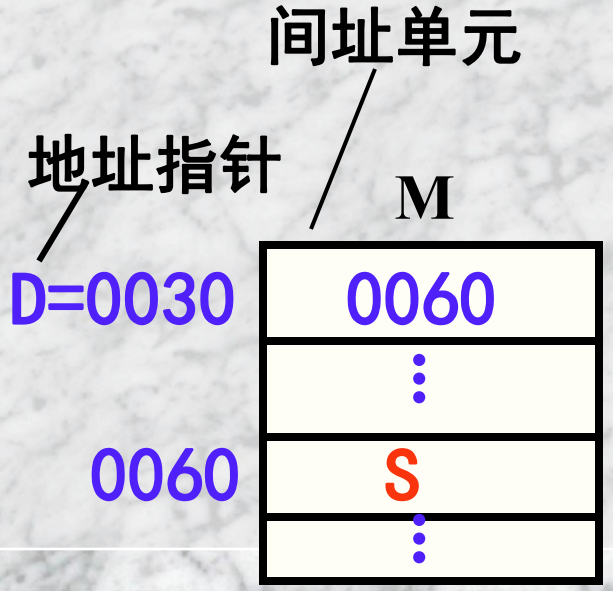
指令给出操作数的间接地址。

- 存储单元号 (数在M中)
- 寄存器号 (数在M中)

● 存储器间址

格式 | 操作码 θ | 间接地址D |

S =((D))



● 寄存器间址

格式 | 操作码 θ | 寄存器号R |

S =((R))

R所占位数少；R可提供全字长地址码；

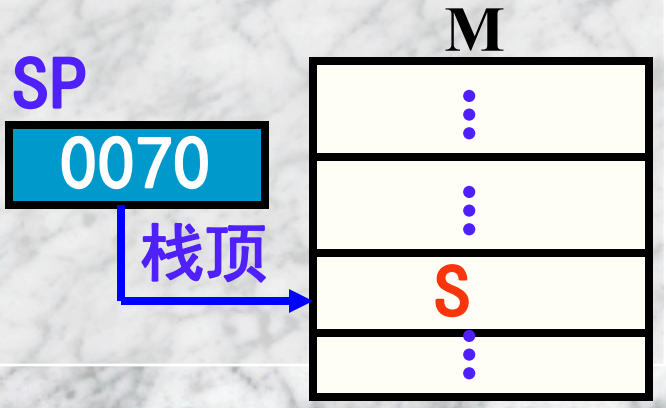
修改R内容比修改M内容快。

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。

● 堆栈寻址

格式 | 操作码 θ | 堆栈指针SP |

S =((SP))



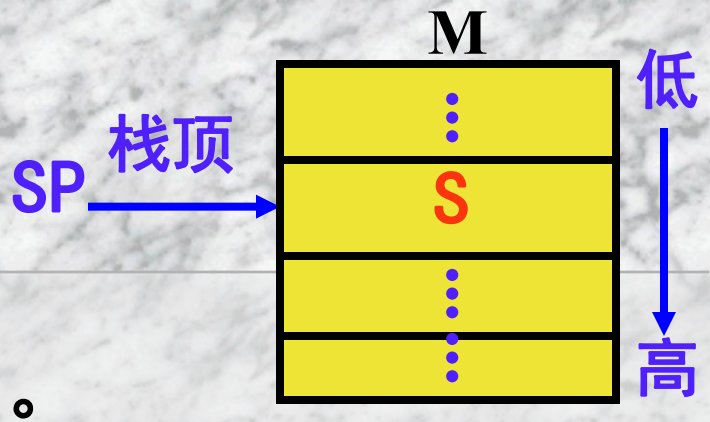
堆栈向上生成

压栈：SP自动减1，再存数。

-(SP)，自减型间址。

出栈：先取数，SP再自动加1。

(SP)+，自增型间址。



SP既可出现在指令中，也可隐含约定。

（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

操作码 θ	Rx	D

格式

变址寄存器号　　形式地址

$S = ((Rx) + D)$

修改量　　基准地址

例. 用变址方式访问一组连续区间内的数组元素。

D为存储区首址；

(Rx)为所访单元距离首址的长度；

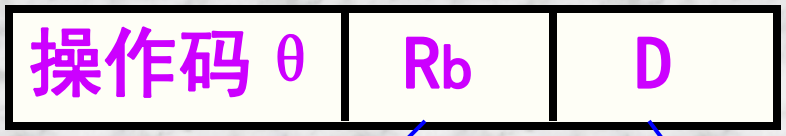
Rx初值为0，每访问一个单元，(Rx)+1。

D的位数有限，若不能提供全字长地址码，会使访存空间受到限制。

● 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。

格式



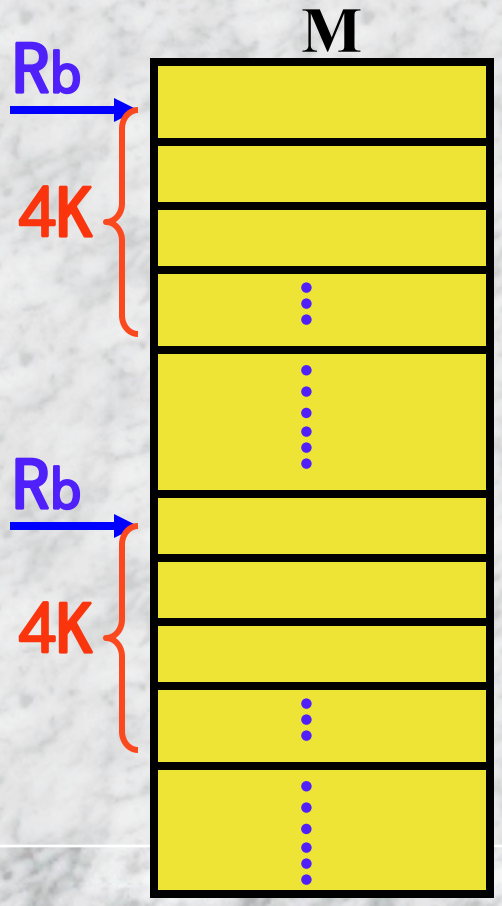
基址寄存器号　位移量

S =((Rb)+ D)

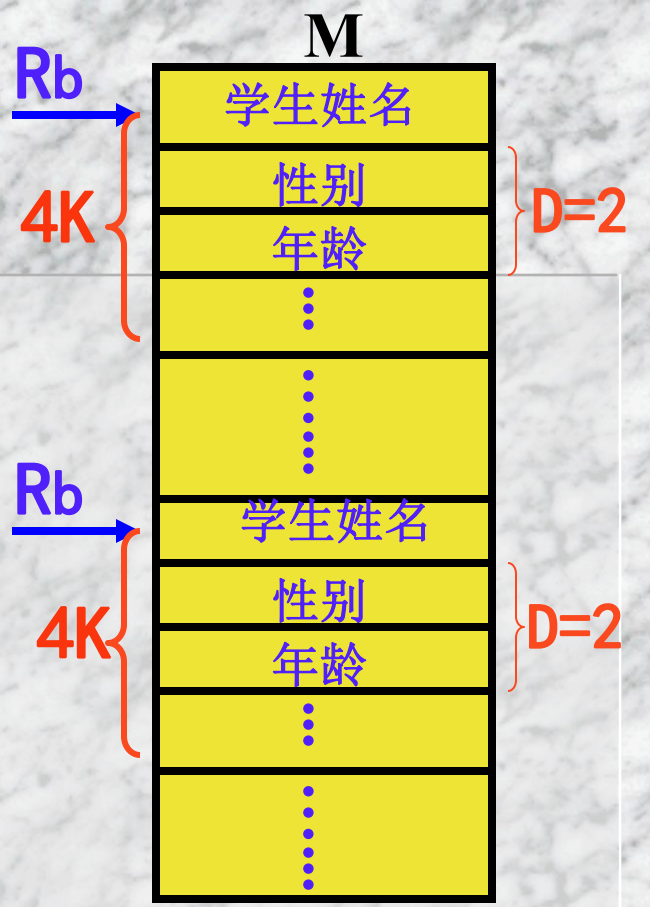
基准地址　相对于基址的位移

（D的位数只需覆盖一个较小的存储区间）

改变Rb的内容，程序能访问存储空间中任何一个定长区间(4K)。



便于访问两维数组中某类指定的元素。



变址与基址的区别：

变址：指令提供基准量(不变)，R提供修改量(可变)；适于处理一维数组。

基址：指令提供位移量(不变)，R提供基准量(可变)；用于扩大有限字长指令的访存空间。

● 基址加变址

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。

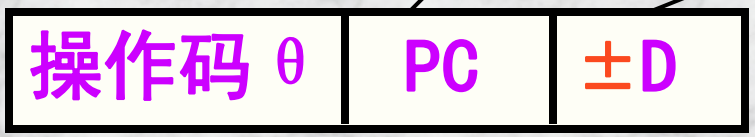
格式

S =((RX)+(Rb)+ D)　　便于处理两维数组。

● 相对寻址

指令给出位移量，PC内容与位移量之和为有效地址。

格式



S =((PC)±D)

有效地址相对PC上下浮动，给编程带来方便。

● 页面寻址

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。

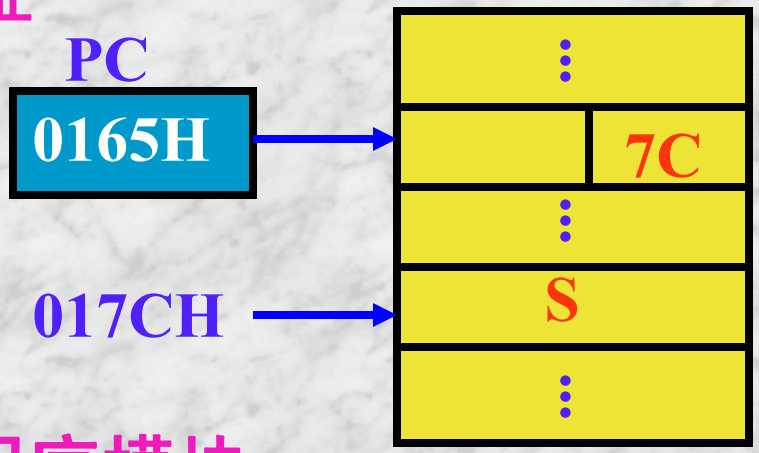
格式

或隐含指定　　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。



用于页式管理存储系统。

寻址速度快，适于组织程序模块，有效利用存储空间。

举例：

1.立即寻址方式

例如：MOV AX，1234H

;1234H是立即数

2.直接寻址（主存直接寻址）方式

例如：MOV AX，[1234H]

;[1234H]是主存储器地址

3.寄存器寻址（寄存器直接寻址）方式

例如：MOV AX，BX

;AX/BX是寄存器地址

4.寄存器间接寻址方式

例如：MOV AX，[BX]

;MOV AX，DS：[BX]

;[BX] 是寄存器间接地址

5.堆栈寻址

例如：。。。。。。。。。。。。。。。。。。。。。。。。。。。。

```
STACK1  SEGMENT PARA STACK
    DW  100  DUP（0）      ；长度100（64H）
STACK1  ENDS
。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。
PHSH  AX                  ；入栈
PUSH  DS
PUSH  DATA-WORD
PUSHF
。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。
。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。；主程序
POPF                      ；出栈
POP  DATA-WORD
POP  DS
POP  AX
```

6. 变址寻址：变址寄存器：SI、DI

例如：MOV AX，10H[DI]

;等价于MOV AX，DS：10H[DI]

MOV AL，20H[SI]

;等价于MOV AL，DS：20H[SI]

7.基址寻址：基址寄存器：BX、BP

例如：MOV DX，VAR[BP]

;等价于MOV DX，SS：VAR[BP]

8.基址加变址方式

例如：MOV AX，10[BX][SI]

;等价于MOV AX，DS：10[BX][SI]

MOV DX，VAR[BP][DI]

;等价于MOV DX，SS：VAR[BP][I]

举例：a+b=c ：2+3=5

汇编语言程序设计：

```
DATA   SEGMENT     ;定义数据段起始
    a  DB  2                ;立即寻址赋初值
    b  DB  3
    c  DB  ?
DATA   END              ;定义数据段结束

STACK1  SEGMENT  PARA  STACK ;定义堆栈段起始
    DW  20H   DUP(0)
STACK1   ENDS           ;定义堆栈段结束

CODE  SEGMENT           ;定义代码段起始
  ASSUME   CS:CODE,DS:DATA,SS:STACK1
```

```
BEGIN:MOV  AX,DATA      ;程序开始的地址
      MOV  DS,AX        ;初始化DS段
      MOV  AL,a
      ADD  AL,b
      MOV  c,AL         ; 前三句为c=a+b
      MOV  AH,4CH       ;调用4CH号功能
      INT  21H          ;返回DOS操作系统
CODE  ENDS
   END  BEGIN           ;汇编结束标志
```

2. 对寻址方式的说明

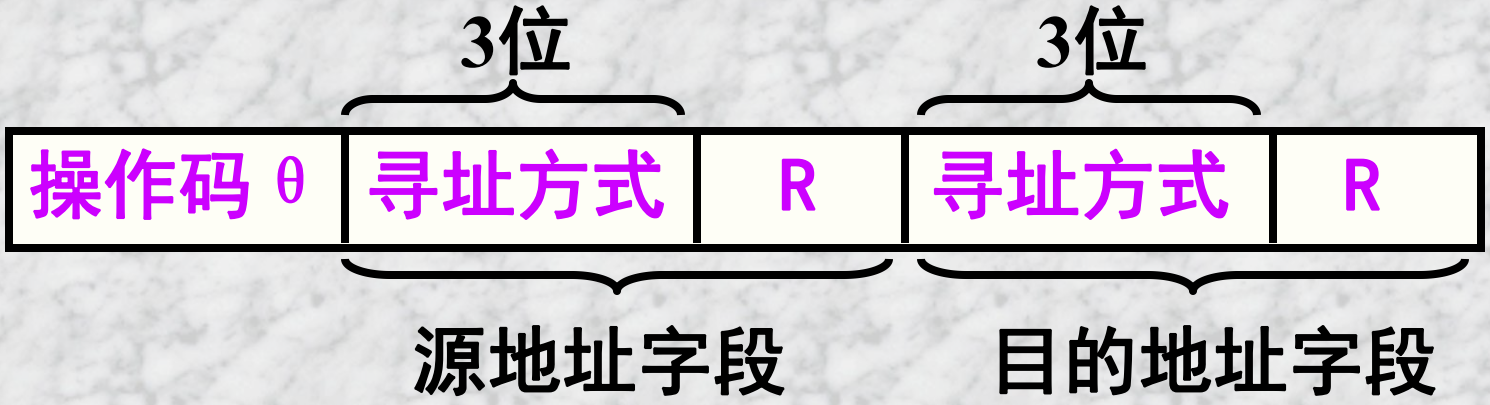
（1）操作码隐含说明不同寻址方式

例. 某机指令操作码最高两位

- 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

（2）指令中设置专门字段说明寻址方式

例. 某机指令的每个地址字段中各设置一个3位的寻址方式字段。



2.2.3 指令类型

1. 传送指令

源地址 —数→ 目的地址

设置时需考虑：

（1）规定传送范围

例. DJS-100系列： R ⟷ M

80X86： R ⟷ M, R ⟷ R

IBM370：R ⟷ M, R ⟷ R, M ⟷ M

（2）指明传送单位

例. 用操作码说明(VAX-11)：MOVB MOVW MOVL

8 16 32

用地址量说明(80X86)：MOV AL, BL 8

MOV AX, BX 16

MOV EAX, EBX 32

例. 80X86的串传送指令：REP MOVSW 传送次数由计数器控制

（3）设置寻址方式

在寻址方式的设置上几乎不受限制，能比较集中地反映指令系统各种寻址方式的实现。

2. 输入/输出指令

各种信息

主机 ⟷ 外设

设置时需考虑：

（1）I/O指令的功能扩展

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

指令中某些字段编码事先不定义，需要时再约定其含义。

用于外设种类、数量不多的场合。

● I/O接口中设置控制/状态寄存器

主机用输出指令或传送指令将具体设备的控制命令按约定的代码格式送往接口中的控制寄存器，向外设发出命令。

外设的状态信息也以某种格式放在接口的状态寄存器中，主机用输入指令或传送指令从状态寄存器中取出有关信息进行查询、分析。

如何设置控制/状态寄存器是接口设计的关键。

（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

如何为I/O端口分配地址？

单独编址

编址到寄存器：为每个寄存器(I/O端口)分配独立的端口地址；

I/O指令中给出端口地址。

I/O地址空间不占主存空间，可与主存空间重叠。

需设置标志区分访问对象，如

$M/\overline{IO}$ { =1 访问存储器
=0 访问I/O端口

● 统一编址

编址到寄存器：为每个寄存器（I/O端口）分配总线地址；

访问外设时，指令中给出总线地址。

I/O端口占据部分主存空间。

常将存储空间的低端分配给主存单元，高端分配给I/O端口，以示区分。

（3）I/O指令设置方式

● 设置专用I/O指令 —— 显式I/O指令

针对单独编址，用I/O指令访问I/O端口。

指令中说明输入/输出操作，并给出端口地址。

单独编址与统一编址的比较

	单独编址方式	统一编址方式
优点	I/O指令和传送指令容易区分，外设地址线少，译码简单，主存空间不会闲置	可用传送指令代替专用I/O指令，通过地址总线访问外设接口中的寄存器（如同通过地址总线访问主存单元一样）
缺点	控制类总线中增加了I/O Read 和 I/O Write 信号线	接口中的寄存器占用主存一部分地址，减少了主存的可用空间

例. 80X86I/O指令设置

输入：IN AL, n; (n)⟶AL (直接端口寻址)

端口地址

IN AL, DX; ((DX))⟶AL(间接端口寻址)

间接端口地址

输出：OUT n, AL; (AL)⟶n (直接端口寻址)

OUT DX, AL;(AL)⟶(DX)(间接端口寻址)

端口地址：8位，即0—255，采用直接寻址

16位，即256—1023，采用间接寻址

例子1：IN　AX，0CH

；采用直接寻址，输入一个字到AX中

例子2：MOV　DX，02ECH

IN　AX，DX

；采用间接寻址，输入一个字到AX中

● 用传送指令实现I/O操作 —— 隐式I/O指令

针对统一编址，用传送指令访问I/O端口。

不设专用I/O指令。

例. 某机I/O接口中设置

控制/状态寄存器CSR，其总线地址为177550（8进制）

数据缓冲寄存器DBR，其总线地址为177552

控制/状态字格式：

15	14	12	7	6	2	1	0
出错	故障	忙	完成	允许中断	维护	校验	启动

主机调用输入机：

启动：主机 —控制字（启动位为1）/传送指令→ 177550

测试：主机 ←状态字/传送指令— 177550

取数：主机 ←数据/传送指令— 177552

- 通过I/O处理机进行I/O操作

两级I/O指令：
- CPU执行简单I/O指令（启动、停止、查询、清除）
- I/O处理机执行I/O操作指令（输入、输出……）

3. 算术逻辑运算指令

（1）算术运算指令

设置时需考虑操作数类型、符号、进制等；

运算结束后设置相应状态标志。

（2）逻辑运算指令

实现对代码位的设置（或）、测试、清除（与）、修改（异或）等。

4. 程序控制指令

控制程序流程。

（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：操作码　子程序入口

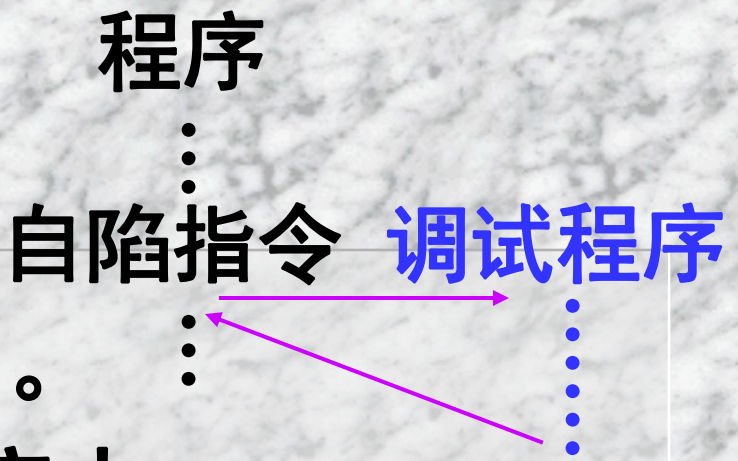
返回：操作码　返回地址

同一条返回指令应能提供多个不同的返回地址。

返回地址的存取：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。

以 INT n 的形式出现在程序中。

表示不同的功能调用

第二章复习提纲

1. IEEE754格式：32位短浮点格式
2. I/O指令的功能扩展(目的、方法)，外设编址方式和指令设置方式。
3. 基本概念：扩展操作码(扩展方法)、地址结构(简化方法)、隐地址、显地址、基本寻址方式(立即、直接、间址、变址)的含义与应用场合。