计算机组成原理

第二章 计算机中的信息表示
（指令系统）

2025-8-22

信息与软件工程学院
School of Information and Software Engineering

主 要 内 容

1. 指令格式
2. 指令寻址方式（难点、重点）
3. 指令类型

计算机是处理信息的工具，即能实现各种**算术运算**与**逻辑运算**等，要了解其工作原理，首先要清楚计算机中信息的表示方式。

- 信息
  - 数据型
    - 数值型
    - 非数值型
  - **控制型**
    - **指令**
    - **微命令**

指令系统

计算机的工作，体现为指令的执行，实际上包含以下三个步骤：

①取指：从主存储器中取指令；

②执行指令：在CPU内部执行该条指令（在此过程中，CPU一般还需从主存储器中获取如操作数等相关的信息）；

③存结果：结果送回存储器存放。

指令系统

指令：一系列按照某种规律有序排列的，能被CPU识别、并执行的二进制代码。

指令系统（或指令集）：一台计算机所能执行的全部指令。

指令系统 

CPU硬件功能

2.1 指令格式

- 01. 指令基本格式
- 02. 指令字长
- 03. 操作码结构
- 04. 指令中的地址结构

一、指令基本格式

一条指令（最多）提供两方面的信息：

①与CPU操作有关的信息----操作码（OP）；

②与操作数有关的信息----地址码（AD）。

OP:Operation

AD:Address

最基本的指令格式：

操作码OP	地址码A

注意：一条指令中的操作码OP有且仅有一个，而地址码A可能有0、1、2、3个。

一、指令基本格式

在指令格式设计时相应地需要考虑以下一些问题：

①指令字长需多少位？

是固定字长还是可变字长？

②操作码构成需多少位？

位数与位置固定还是可扩展？

一、指令基本格式

在指令格式设计时相应地需要考虑以下一些问题：

③**地址码**：一条指令的执行涉及到哪些地址？在指令中给出哪些地址？有几个？哪些地址是**系统隐含约定**的？

扩展

④**寻址方式**：根据地址码如何获取**操作数地址**或**操作数**？是直接给出还是间接给出？或是经过某种变换（包括计算）获取的？（**难点、重点**）

二、指令字长

指令字长越长：

优点：指令功能越丰富，

完成工作越多；

缺点：占用存储空间大，

从主存中取指时间越长，

指令执行速度越慢。（特别是早期）

在计算机中，根据需要，指令长度可以变换。

二、指令字长

例:某计算机主频2GHz

CPU内的一次（大约）处理时间：

$$2GHz=1/(2\times10^{9})=5\times10^{-10}(s)$$

访存时间：

$$5ns=5\times10^{-9}(s)$$

CPU处理时间与访存时间相差一个数量级，这就是计算机“瓶颈”问题。

三、操作码结构

操作码的位数决定了操作类型的多少，

位数越多所能表示的操作种类也就越多。

操作码分类
- 定长操作码
- 扩展操作码
- 方式码

三、操作码结构

①定长操作码：

指令长度比较长时，位置、位数固定，位置在指令的前几位；

②扩展操作码：

指令长度比较短时，位置、位数不固定，用扩展标志表示。

三、操作码结构

扩展操作码示例

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。设计符合这些条件的指令:即三地址15条、二地址15条、一地址15条、零地址16条。

③ 方式码：

操作码分为几部分，每部分表示一种操作。

例：某机算逻指令

四、指令中的地址结构

地址结构：在指令中明确给出几个地址，给出哪些地址。

- 操作数的位置
  - 堆栈
    - 主存开辟（软堆栈）
    - CPU中的寄存器组组成（硬堆栈）
  - 寄存器
    - CPU中的寄存器
    - 外设接口中的寄存器
  - 存储器
    - 主存（包括cache）
    - 外存

四、指令中的地址结构

指令给出操作数地址方式：

- 显式：直接、间接、变址、基址等
- 隐式：隐含约定寄存器号、主存储器单元号

①显地址：

如果在指令中明显地给出地址，如写明主存储单元号或CPU的寄存器编号，则这种地址表达称为显地址。

显地址又分为：

三地址指令、二地址指令、

一地址指令、零地址指令。

四、指令中的地址结构

②隐地址：

如果在指令中的地址是以隐含的方式约定，如系统事先隐含约定操作数在CPU某个寄存器中，或是在堆栈中，因此在指令中并不给出地址码，这种隐含约定的地址就称为**隐地址**。

缩短指令长度

简化地址结构的基本途径：尽量使用**隐地址**。

四、指令中的地址结构

（1）四地址指令

格式：

操作码　操作数地址1　操作数地址2　结果存放地址　下条指令地址

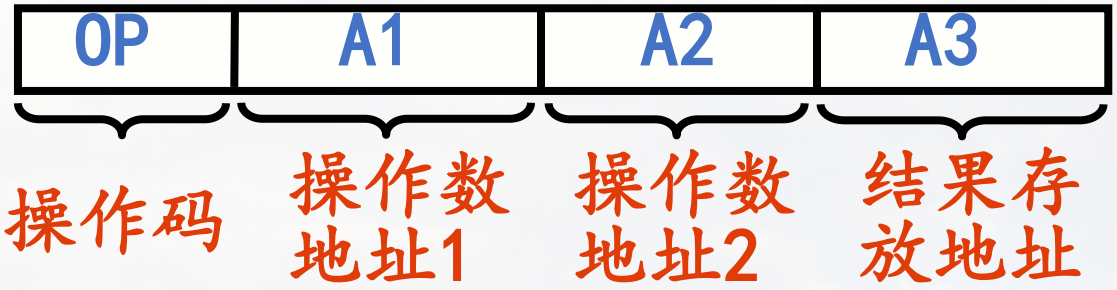
功能：　（A1）OP（A2）→ A3

下条指令地址 → A4

（2）三地址指令

四地址指令 ⟶ 三地址指令：使用隐地址

格式：



功能：（A1）OP（A2）⟶ A3

下条指令地址PC：（PC）+ n ⟶ PC

（A4隐含由CPU中的寄存器PC给出；在模型机中，为简化起见，令n=1）

（3）二地址指令

在绝大多数情况下，两个操作数运算后至少有一个今后不再使用，因而不需要保留，可将运算结果放入其中之一的地址中。

格式：

功能：（A1） OP （A2）⟶ A1/A2

下条指令地址PC：（PC）+ n ⟶ PC

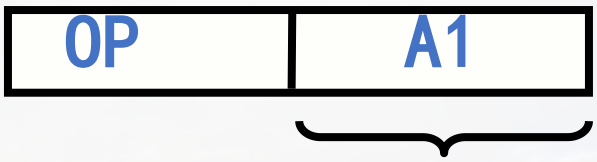
如： ADD AX, BX

(4) 一（单）地址指令

一（单）地址指令有以下两种常用的表示方式。

a. 隐含约定目的地的双操作数指令

格式：



功能：(A1) OP (AC) → AC

下条指令地址PC：（PC）+ n → PC

（4）一（单）地址指令

例：无符号乘法

8位寄存器

1）字节（8b）乘法：OPRD×AL→AX

如：MUL DL ；DL×AL→AX

16位寄存器

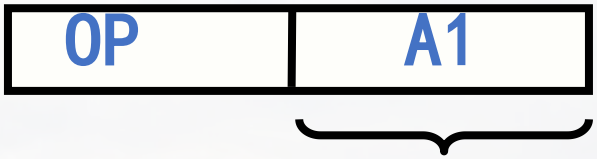
2）字(16b)乘法：OPRD×AX→DX：AX

如：MUL BX ；BX×AX→DX：AX

（4）一地址指令

b. 只有目的操作数的单操作数指令

格式：



功能：　OP（A1）⟶ A1

下条指令地址PC：（PC）+ n ⟶ PC

例：NEG　BL　；求负

NOT　BL　；求非

（5）零地址指令

格式：

如果指令中只给出操作码而没有显地址，则这种指令被称为零地址指令。

(5) 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：PUSHF ；FLAGS→堆栈栈顶

POPF ；堆栈栈顶→FLAGS

LAHF ；FLAGS的低8位→AH

SAHF ；AH→FLAGS的低8位

(5) 零地址指令

b. 不需要操作数的指令。

例： NOP ；空操作指令

它本身没有实质性运算操作，执行这种指令的目的就是消耗时间以达到延时的目的。

例： HLT ；停机指令

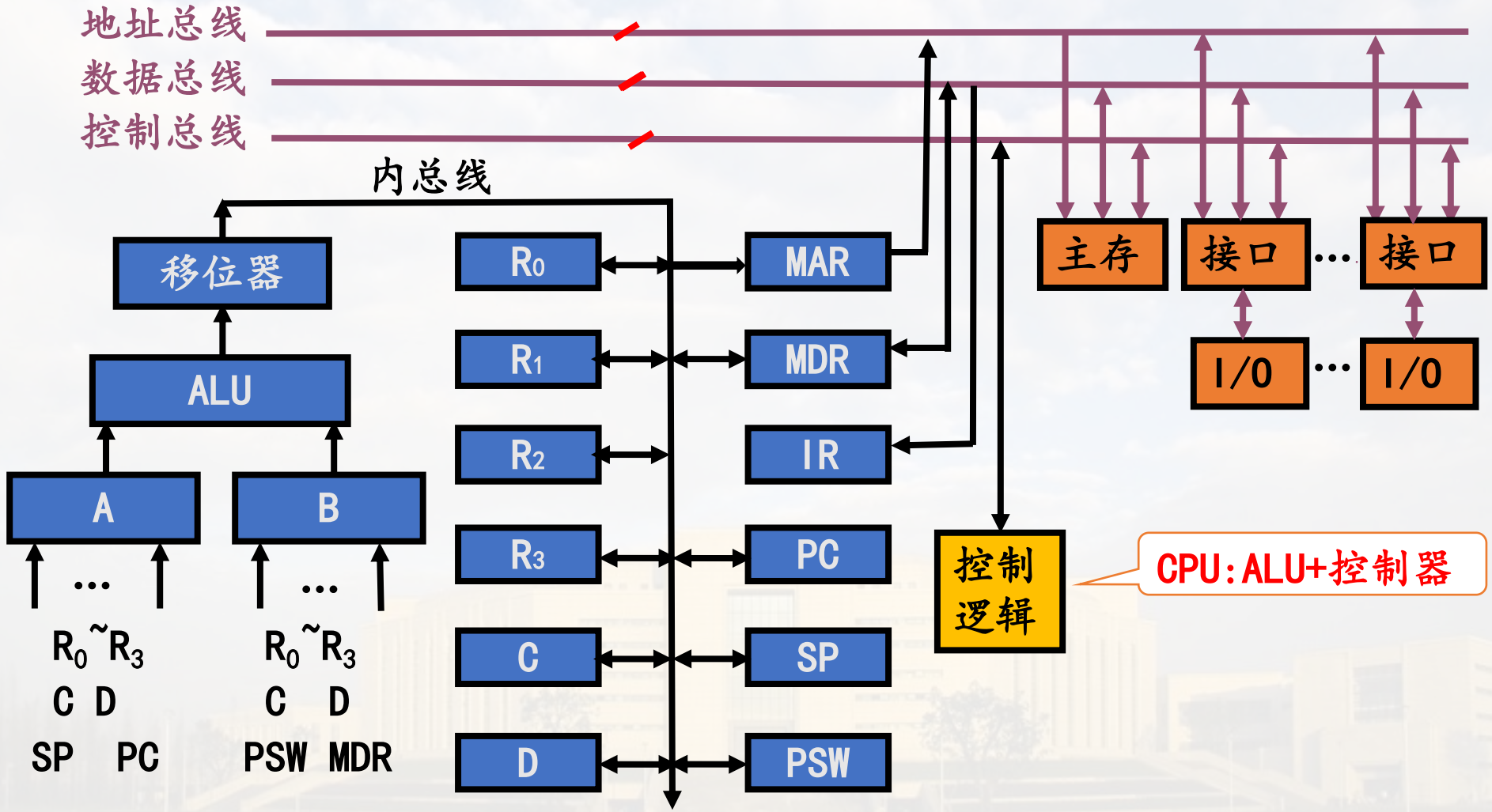
它也不需要操作数。

c. 对堆栈栈顶单元内容进行操作，如指令PUSH（压入堆栈）、POP（弹出堆栈）。

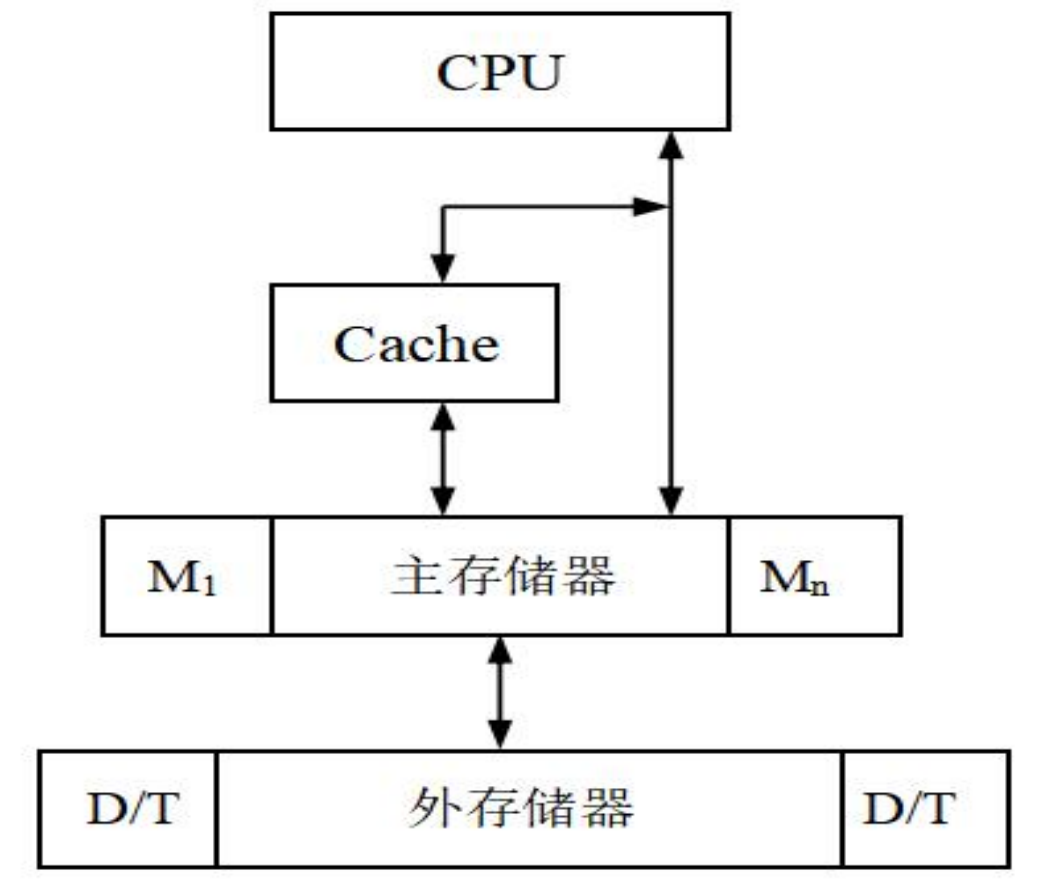
2.2 指令寻址方式

- 01. 操作数存储位置
- 02. 寻址方式

模型机数据通路框图


地址总线
数据总线
控制总线
内总线
移位器
ALU
A
B
R0~R3
C D
SP PC
R0~R3
C D
PSW MDR
R0
R1
R2
R3
C
D
MAR
MDR
IR
PC
SP
PSW
控制逻辑
主存
接口
接口
I/O
I/O
CPU:ALU+控制器

存储器的层次结构图



一、操作数存储位置

寻址方式是规定如何对地址字段作出解释，以找到操作数或确定操作数位置。

- 操作数的位置
  - 堆栈
    - 主存开辟（软堆栈）
    - CPU中的寄存器组组成（硬堆栈）
  - 寄存器
    - CPU中的寄存器
    - 外设接口中的寄存器
  - 存储器
    - 主存（包括cache）
    - 外存

一、操作数存储位置

能被CPU直接访问的操作数位置 { CPU内的R；$M_{主}$（包括Cache、接口中的R）

结论：

①CPU能够直接访问的操作数只能存放在主存储器（包括Cache、接口中的R）或CPU内的寄存器中；

②由于主存储器的容量远远大于CPU内的寄存器的容量，因此CPU能够直接访问的操作数主要存放在主存储器中。

二、寻址方式

寻址方式可分为四大类，其它的寻址方式则是它们的变型或组合。

四类
- 立即寻址
- 直接寻址类：直接寻址、寄存器寻址
- 间接寻址类：存储器间接寻址、多重间接寻址、寄存器间接寻址、自增型寄存器间接寻址、自减型寄存器间接寻址、堆栈
- 变址类：变址寻址、基址寻址、基址加变址寻址、相对寻址、页面寻址

二、寻址方式

寻址方式可分为四大类，其它的寻址方式则是它们的变型或组合。

① 立即寻址

在读取指令的同时也就从指令之中获得了操作数，即操作数包含在指令中。（操作数此时在IR中）

② 直接寻址类

直接给出主存地址或寄存器编号，从主存单元内或CPU的寄存器内读取操作数。

③ 间接寻址类

先从某寄存器/主存中读取地址，再按这个地址访问主存以读取操作数。

④ 变址类

指令给出的是形式地址（不是最终地址），经过某种变换（例如相加、相减、高低位地址拼接等），才获得有效地址，据此访问主存以读取操作数。

(1) 立即寻址方式

立即寻址是一种特殊的寻址方式。

指令中在操作码字段后面的部分不是通常意义上的操作数地址，而是操作数本身，也就是在取出指令的同时也就取出了可以立即使用的操作数。

例： MOV AX，1234H

（2）直接寻址方式（绝对地址）：两种

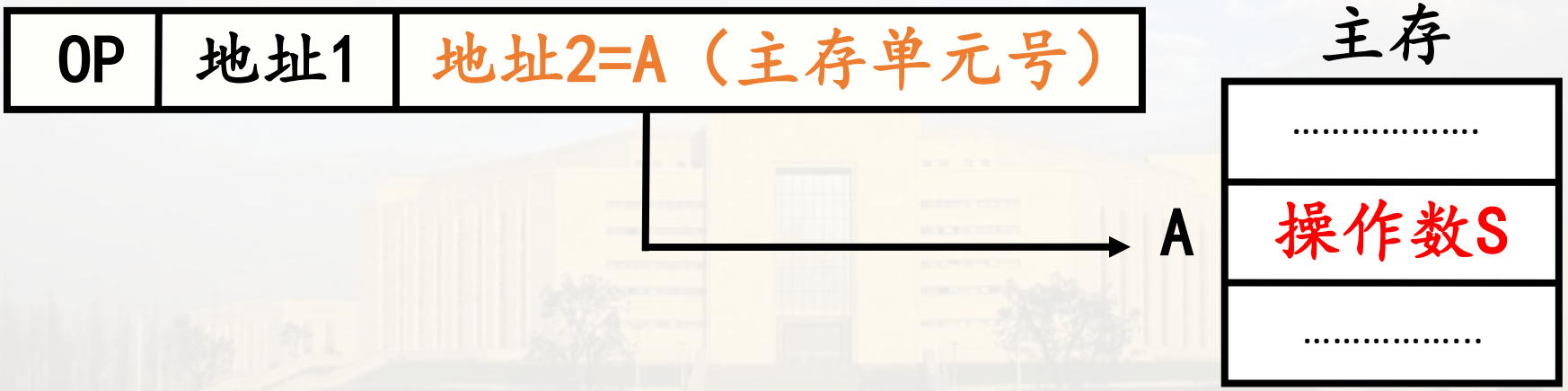
由指令直接给出操作数地址，根据该地址可读取或写入操作数，这种方式称为直接寻址方式。

① 直接寻址（主存直接寻址）方式

若指令中给出的地址码是主存的某个单元号，操作数存放在该指定的主存单元中，这种寻址方式称为直接寻址或主存直接寻址方式。

（2）直接寻址方式（绝对地址）

假定主存储器是按字编址（16位），指令正好占一个字存储空间，操作数为S，主存单元地址码为A；指令包含操作码OP和地址码A。



(2) 直接寻址方式（绝对地址）

例：若主存储器数据区的地址与数据之间对应关系如下，指令给出地址码A=2000H，按直接寻址方式读取操作数。

地址	数据
1000H	1A00H
2000H	1B00H
3000H	1C00H

操作数地址 → 2000H

操作数 → 1B00H

寻址过程：操作数地址 $\xrightarrow{M}$ 操作数

操作数S与地址码A的关系为：S=(A)

例如：MOV AX, [2000H]

(2) 直接寻址方式（绝对地址）

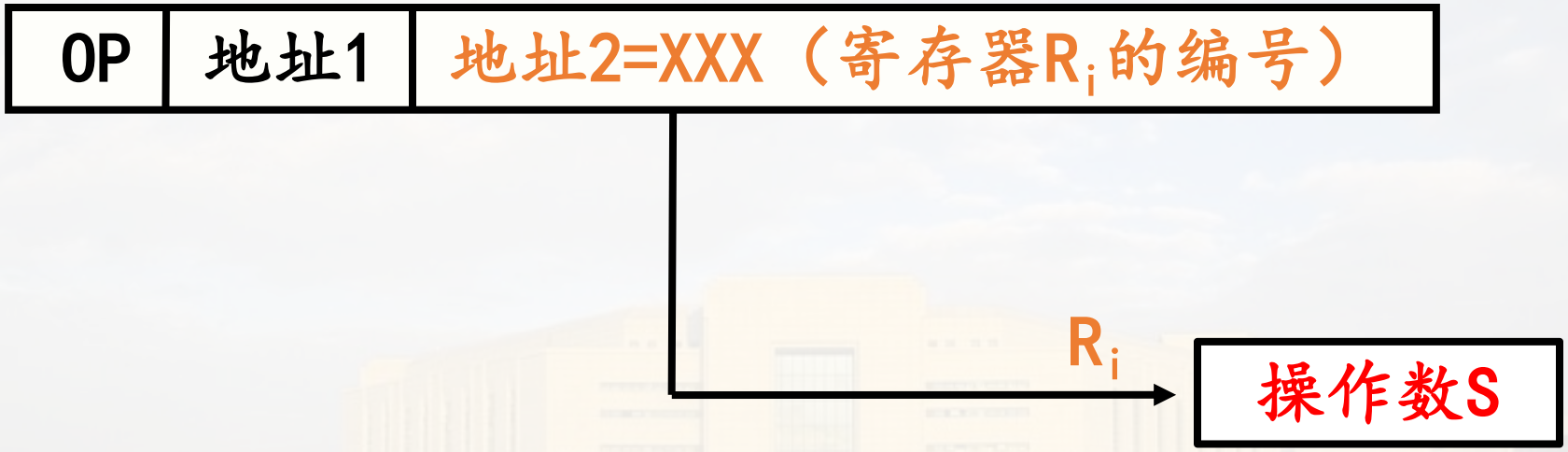
②寄存器寻址（寄存器直接寻址）方式

若指令中给出的地址码是CPU内某寄存器编号，操作数存放在该指定的寄存器中，这种寻址方式称为寄存器寻址或寄存器直接寻址方式。

在CPU中有若干寄存器，其中的一些是可编程访问的，称为可编址寄存器，设计时为它们分配不同的寄存器编号，例如：R_0=000，R_1=001，R_2=010，R_3=011，SP=100，PSW=101，PC=111等。

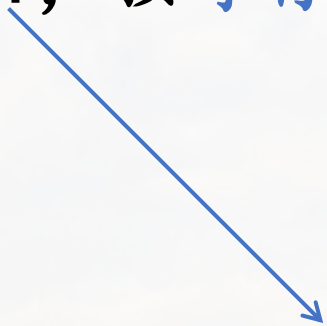
（2）直接寻址方式（绝对地址）

指令中给出的寄存器号是R_i（按上述编码则代码为XXX ），从寄存器R_i中可直接读取操作数S。



(2) 直接寻址方式（绝对地址）

例：若CPU中寄存器内容如下，现指令中给出寄存器号为011，按寄存器寻址方式读取操作数。



R_0	1000H
R_1	2000H
R_2	3A00H
R_3	3C00H

寻址过程：　寄存器号 $\xrightarrow{R_i}$ 操作数

操作数S与寄存器R_i的关系为：$S=(R_i)$

例如：MOV AX, BX

(2) 直接寻址方式（绝对地址）

直接寻址与寄存器寻址方式的比较：

a. 直接寻址是访问一次主存才能读取所需操作数；寄存器寻址是从CPU的寄存器中读取操作数，不需访问主存，所需时间大约是从主存中读数时间的几分之一到几十分之一，因而寄存器寻址比直接寻址快得多。

故在CPU中设置足够多的寄存器，以尽可能多地在寄存器之间进行运算操作，已成为提高工作速度的重要措施之一。

（2）直接寻址方式（绝对地址）

b. 由于寄存器数远少于主存储器的单元数，所以指令中存放寄存器号的字段位数也就大大少于存放主存地址码所需位数。采用寄存器寻址方式或其它以寄存器为基础的寻址方式（例如寄存器寻址、寄存器间址方式），可以大大减少指令中一个地址的位数，从而有效地缩短指令长度。这也使读取指令的时间减少，提高了工作速度。

（2）直接寻址方式（绝对地址）

例如：1MB的内存（即2^{20}），其地址所占的位数为20位

即：

（2）直接寻址方式（绝对地址）

注意：减少指令中地址数目与减少一个地址的位数是两个不同的概念。

采用隐地址可以减少指令中地址的数目；（参看本节PPT的P20）

采用寄存器寻址方式、寄存器间址方式等可以使指令中的地址位数减少。

其实，均减少了指令长度。

（3）间接寻址及其变形

地址段提供的不一定就是操作数地址，如间接寻址方式。

① 间接寻址（主存间接寻址）方式

1、若操作数存放在主存某个存储单元中，则该主存单元的地址被称为操作数地址。

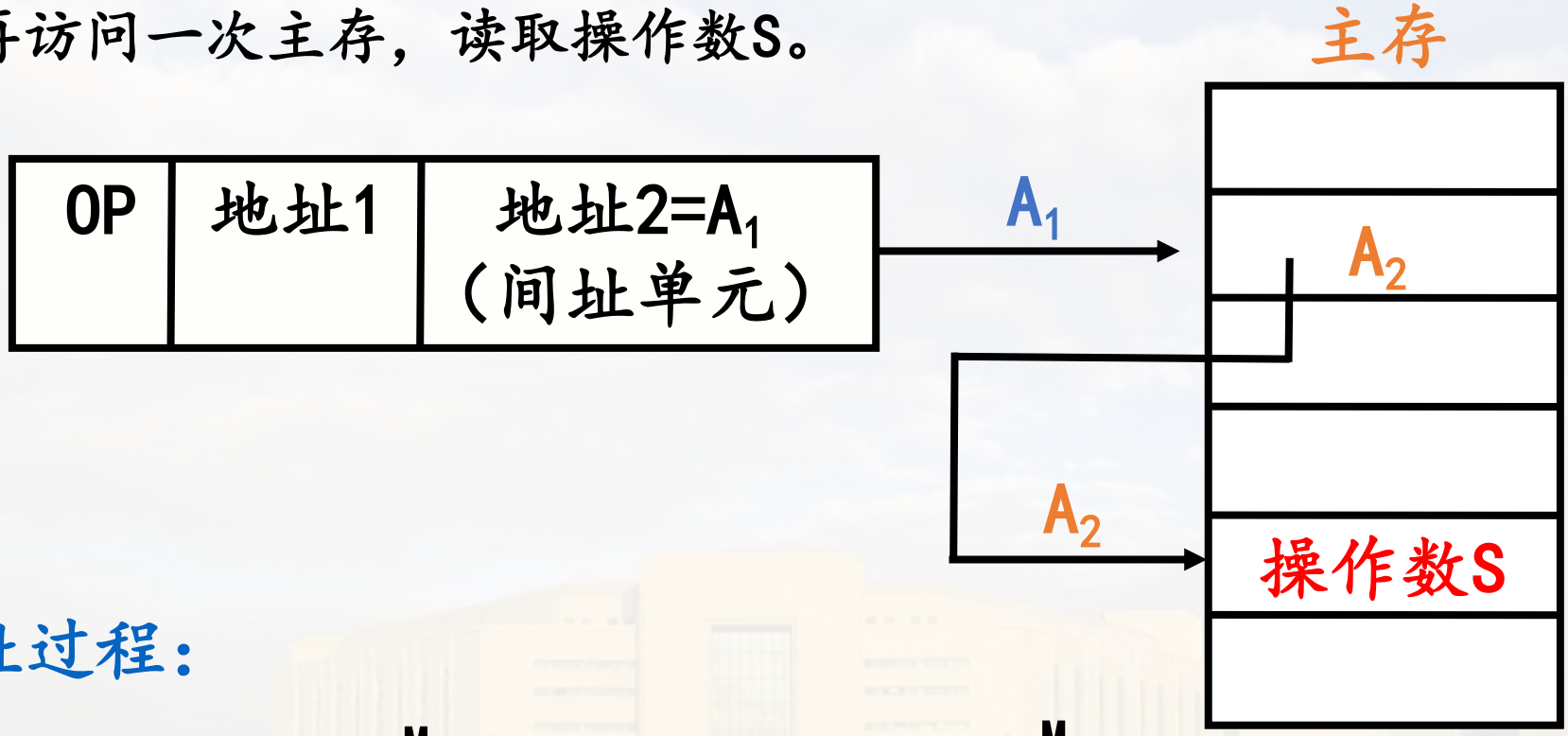
2、若操作数地址存放在另一主存单元之中（不是由指令直接给出），则该主存单元被称为间址单元，间址单元本身的地址被称为操作数地址的地址。

（3）间接寻址及其变形

3、若指令中地址给出的是间址单元地址（即操作数地址的地址，而不是操作数地址，且在主存），从中读取操作数地址，按照操作数地址再次访问主存，从相应单元中读写操作数，这种寻址方式称为间接寻址或主存间接寻址方式。主存器间址方式常用助记符（M）表示。

（3）间接寻址及其变形

指令中给出地址A_1，据此访问间址单元，从中读取操作数地址A_2，按A_2再访问一次主存，读取操作数S。



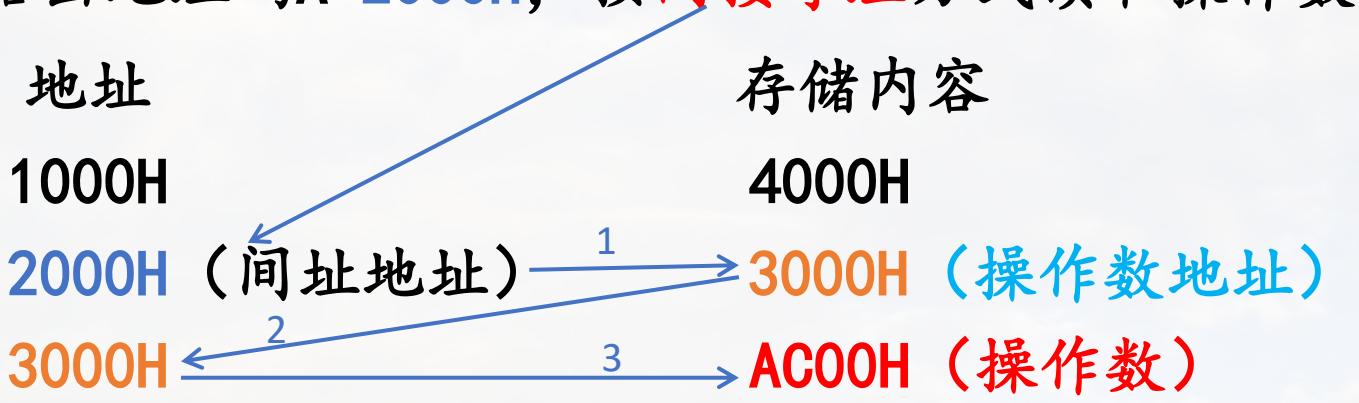
寻址过程：

间址单元地址A_1 $\xrightarrow{M_1}$ 操作数地址A_2 $\xrightarrow{M_2}$ 操作数S

操作数S与地址A_1、A_2的关系为：$S=((A_1))=(A_2)$

（3）间接寻址及其变形

例：若主存储器数据区的地址与单元内容之间对应关系如下，指令给出地址码A=2000H，按间接寻址方式读取操作数。



指令给出间址单元地址A=2000H；

据此访问主存储器，则操作数地址(A)=3000H；

按此地址再次访问主存储器，则操作数S=((A))=ACOOH。

（3）间接寻址及其变形

地址段提供的不一定就是操作数地址，如间接寻址方式。

② 寄存器间接寻址方式

若指令中给出的地址码是CPU的寄存器编码，被指定的寄存器中存放的是操作数地址，按照该地址访问主存某单元，该单元的内容为操作数，这种寻址方式称为寄存器间接寻址。寄存器间址方式常用助记符（R）表示。

（3）间接寻址及其变形

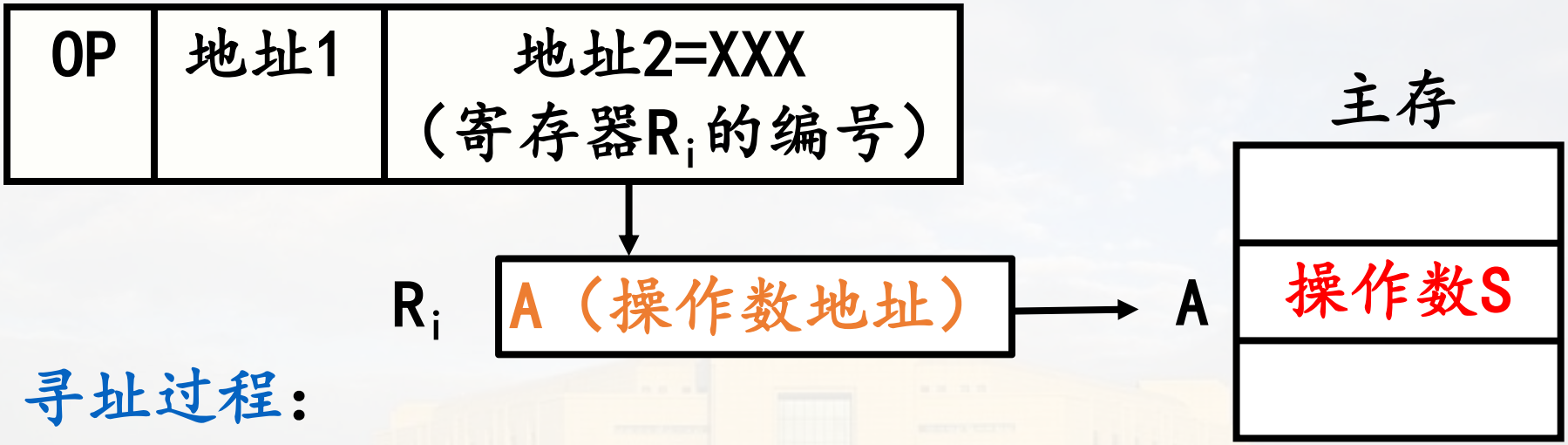
② 寄存器间接寻址方式

在模型机中，CPU内的7个可编址寄存器编号，为：

R_0=000，R_1=001，R_2=010，R_3=011，SP=100，PSW=101，PC=111等。

（3）间接寻址及其变形

指令在地址段给出的是寄存器R_i编号，从R_i中读出的是操作数地址A，按地址码A访问主存，从相应单元中读取操作数S。



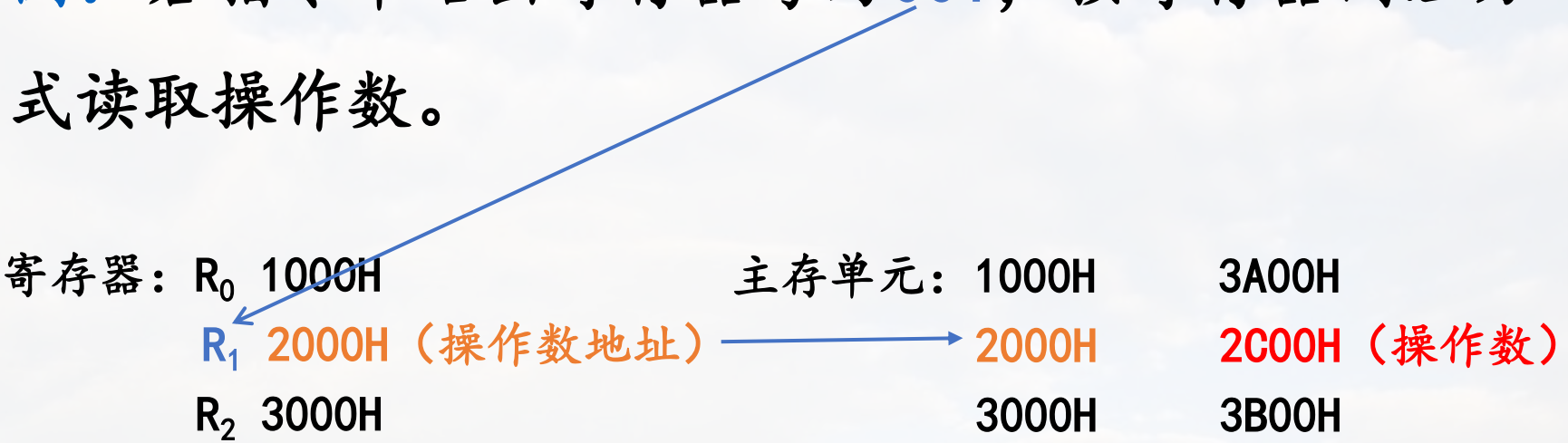
寻址过程：

寄存器号 $\xrightarrow{R_i}$ 操作数地址A $\xrightarrow{M}$ 操作数S

操作数S与寄存器R_i、A的关系为：$S=((R_i))=(A)$

（3）间接寻址及其变形

例：若指令中给出寄存器号为001，按寄存器间址方式读取操作数。



指令指定的寄存器为R_1，则操作数地址（R_1）=2000H，据此访问主存储器，则操作数为S=((R1))=(2000H)=2C00H。

(3) 间接寻址及其变形

采用寄存器间址方式，还可以大大的缩短地址段位数。

寄存器间接寻址方式(8086/8088)：

BX 、SI、DI（隐含DS段），BP（隐含SS段）

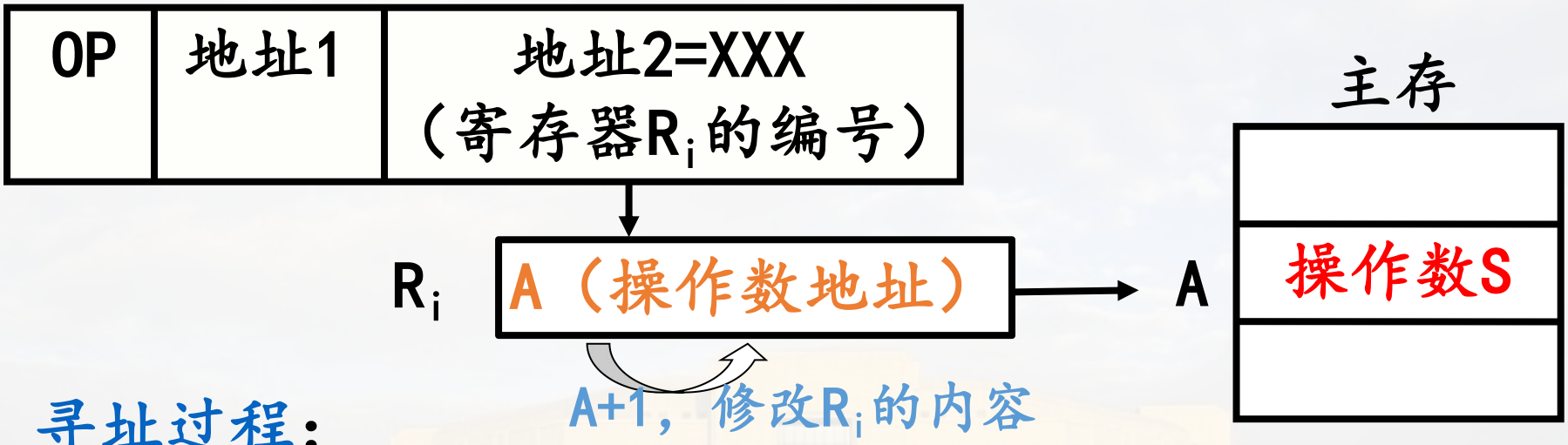
例如：MOV AX，[BX] ；等价于MOV AX，DS：[BX]

③ 自增型寄存器间址方式

寄存器间址的一种变型，若指令中给出存放操作数地址的寄存器号，从寄存器中读出操作数地址后，访问主存相应单元，读取操作数S，同时寄存器内容加1，这种寻址方式称为自增型寄存器间接寻址。自增型寄存器间址方式常用助记符(R)+表示。

(3) 间接寻址及其变形

指令中在地址段给出的是寄存器R_i编号，从R_i中读出的是操作数地址A，按地址码A访问主存，从相应单元中读取操作数S，同时，对寄存器R_i中的内容加1。



寻址过程：

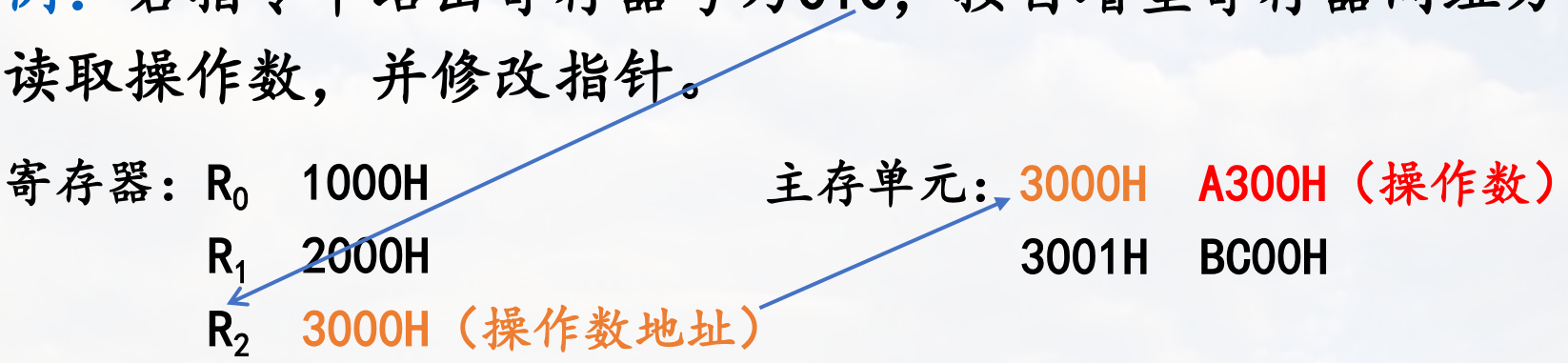
寄存器号 $\xrightarrow{R_i}$ 操作数地址A $\xrightarrow{M}$ 操作数S

R_i内容+1

操作数S与寄存器R_i的关系为：$S=((R_i))=(A)$

（3）间接寻址及其变形

例：若指令中给出寄存器号为010，按自增型寄存器间址方式读取操作数，并修改指针。



指令指定的寄存器为R_2，则操作数有效地址$(R_2)=3000H$；按照该地址访问主存储器，则操作数为$S=((R_2))=(3000H)=A300H$；R_2内容加1后，指针内容修改为$(R_2)=3001H$。

照此继续，通过重复执行这同一条指令就可以沿着地址码增加的方向，访问从3000H单元开始的一段连续区间。

（3）间接寻址及其变形

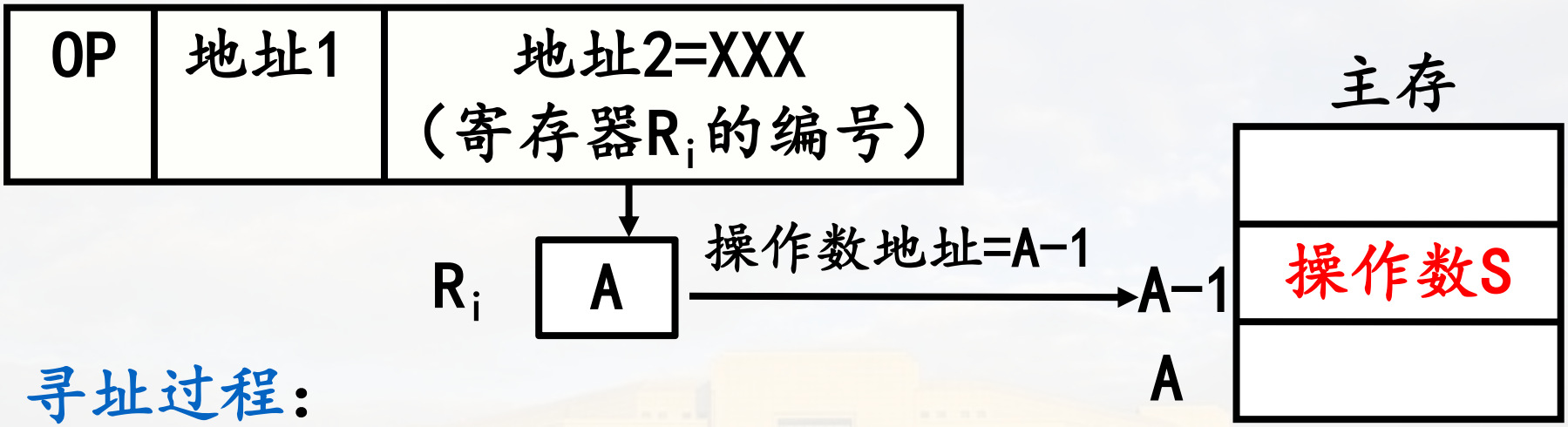
④ 自减型寄存器间址方式

寄存器间址的又一种变形。

若指令中给出寄存器号，被指定的寄存器内容减1后作为操作数地址，按照该地址访问主存储器，相应的主存单元内容为操作数，自减型寄存器间址方式常用助记符 -(R) 表示。

(3) 间接寻址及其变形

指令在地址段给出的是寄存器R_i编号，将R_i中的内容减1作为操作数地址A，按地址码A访问主存，从相应单元中读取操作数S。



寻址过程：

寄存器号 $\xrightarrow{R_i}$ 操作数地址=$(R_i)-1$ $\xrightarrow{M}$ 操作数

操作数S与寄存器R_i的关系为：$S=((R_i)-1)=(A)$

（3）间接寻址及其变形

例：若指令中给出寄存器号为010，按自减型寄存器间址方式修改指针，并读取操作数。

寄存器：	R_0	1000H	主存单元：2FFEH		A300H
	R_1	2000H	2FFFH（操作数地址）		16FFH（操作数）
	R_2	3000H	3000H－1	3000H	BC00H

指令指定的寄存器为R_2；

将R_2的内容减1后作为操作数地址（R_2－1）= 3000H－1=2FFFH；从地址2FFFH单元中读得操作数S=((R2) －1)=(2FFFH)=16FFH。照此继续，通过重复执行这同一条指令，就可以访问从2FFFH开始，沿地址码减小方向的一个连续数据区。

(3) 间接寻址及其变形

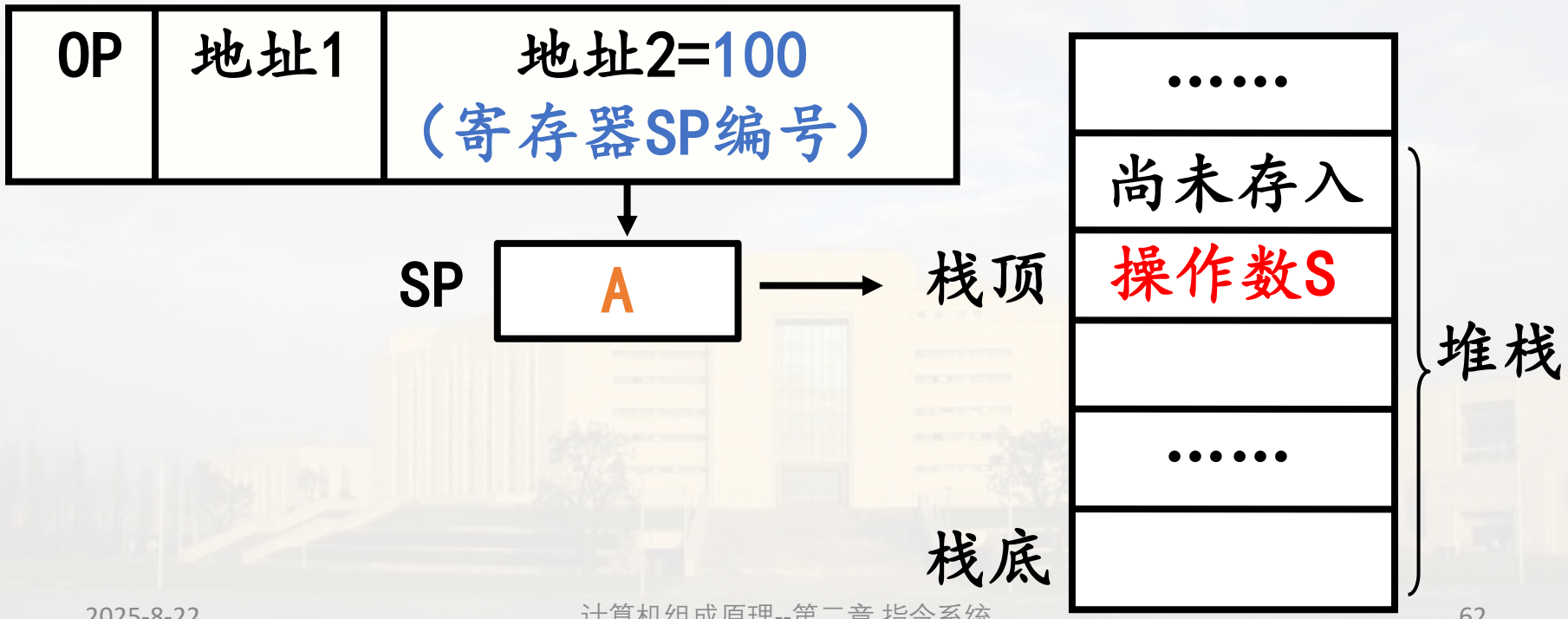
⑤ 堆栈寻址

堆栈寻址方式是指操作数在堆栈中，指令隐含约定由堆栈指针SP寄存器提供栈顶单元地址（SP也可以编码形式出现在指令中），进行读出或写入的一种寻址方式。

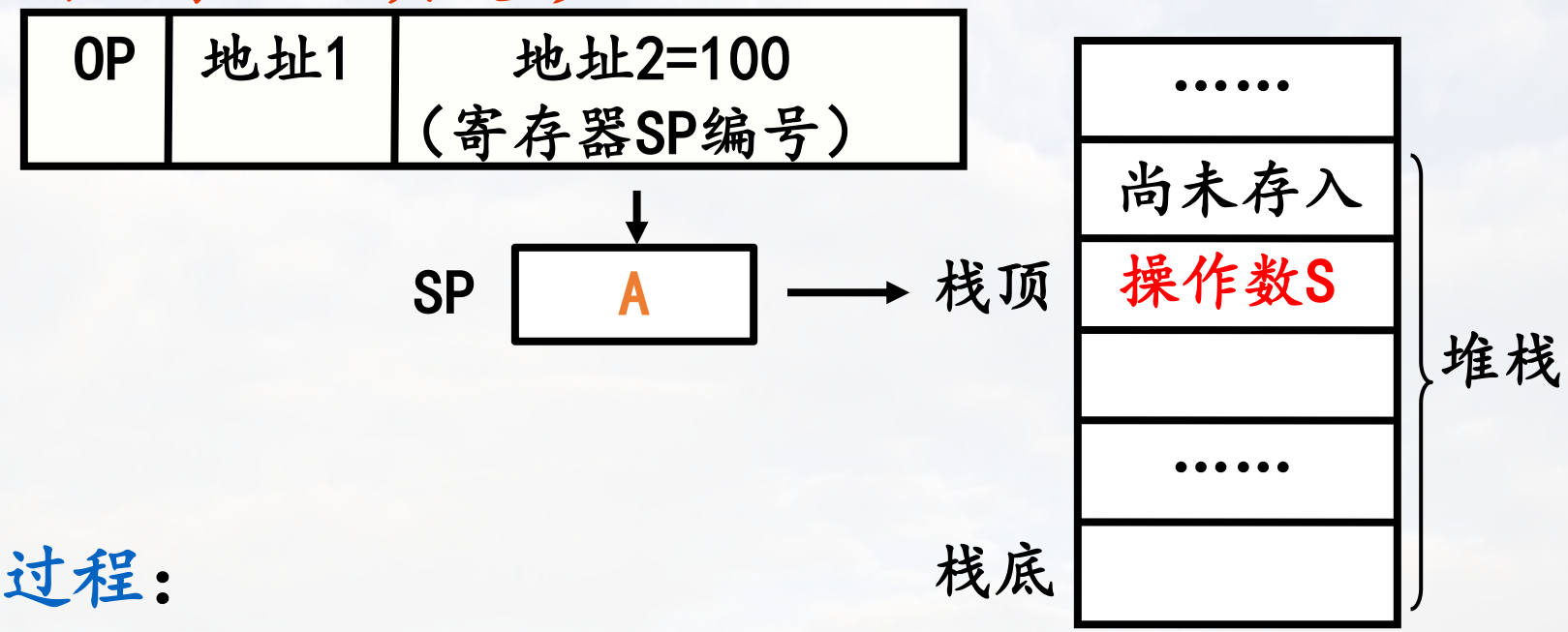
根据压入数据时栈顶单元的地址是减小还是增大或不变，可以将堆栈的工作方式大致分为向上生成方式、向下生成方式和栈顶固定方式三种。

(3) 间接寻址及其变形

指令在地址段给出的是寄存器号SP编号，对SP中的内容进行相应操作（减1或加1，对应压栈或出栈），得到操作数地址A，按地址码A访问主存，从相应单元中读取操作数S。



（3）间接寻址及其变形



寻址过程：

1）压栈：寄存器号 $\xrightarrow{SP}$ 操作数地址=(SP)-1 $\xrightarrow{M}$ 操作数S

操作数S与寄存器SP的关系为：S=((SP) -1)

2）出栈：寄存器号 $\xrightarrow{SP}$ 操作数地址 $\xrightarrow{M}$ 操作数S

操作数S与寄存器SP的关系为：S=((SP))

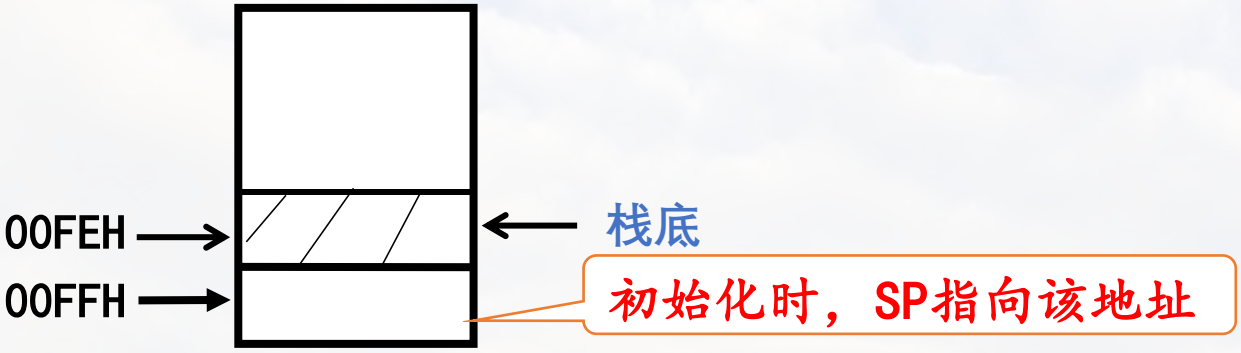
（3）间接寻址及其变形

例：对堆栈的连续压入与连续弹出（自底向上生长方式），SP内容为00FFH，压入第一个数据元素a，然后压入第二个数据元素b，最后弹出栈顶单元内容。

最基本的堆栈操作指令有两种：

1. 压入指令PUSH（进栈、压栈），将指定的操作数存入栈顶；
2. 弹出指令POP（出栈），将栈顶数据读出，送入指定目的地。

（3）间接寻址及其变形

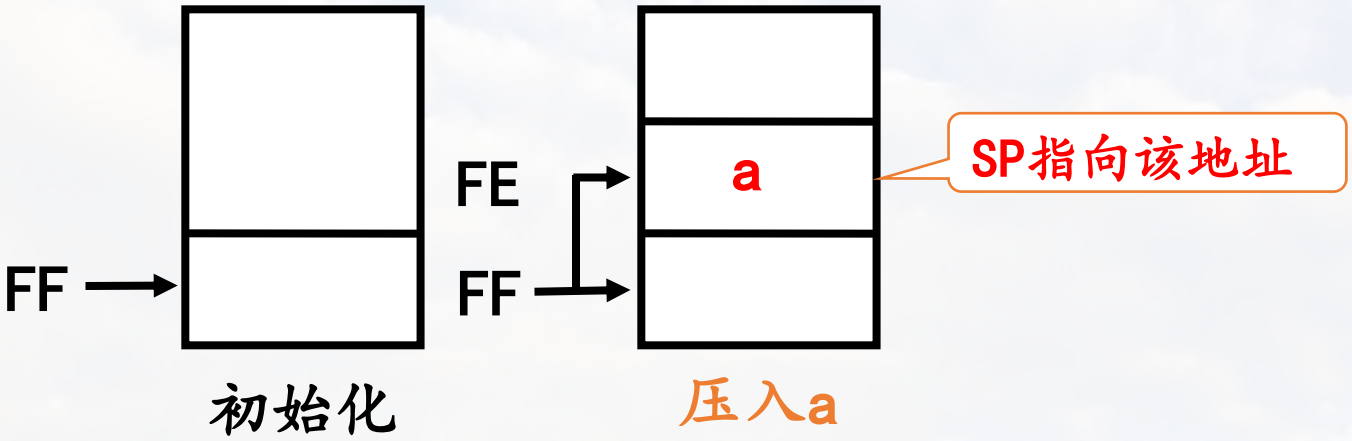


初始化

1）初始化

将初始值送入堆栈指针SP寄存器，本例中假定初始值为00FFH。（在模型机系统中，将压入数据的第一个堆栈单元称为栈底，SP为初始化为栈底地址+1）

（3）间接寻址及其变形



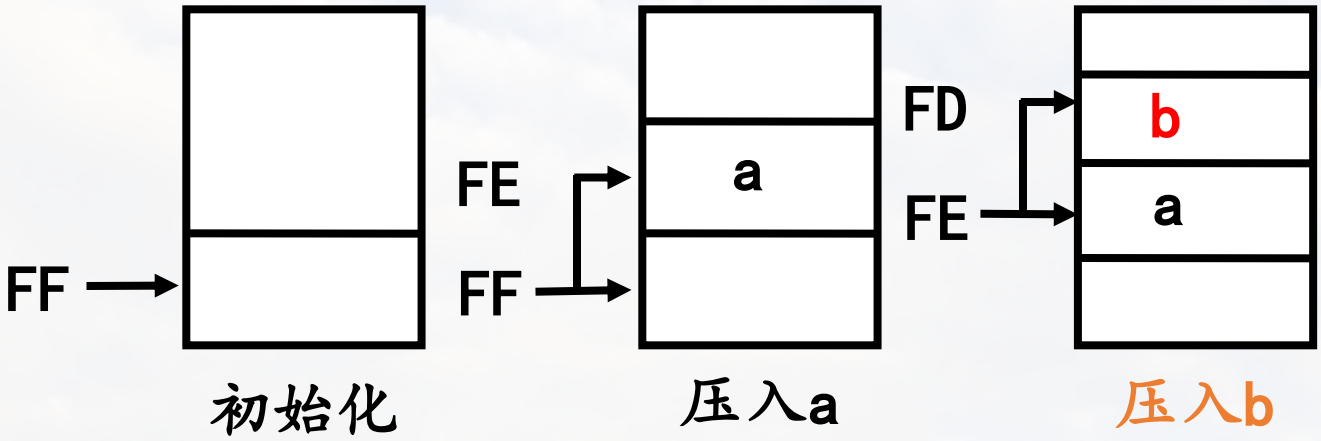
1）压入第一个数据元素a

a. (SP)－1→SP。先修改堆栈指针，指向待存入的新栈顶。SP内容00FFH减1后，修改为00FEH。

b. 压栈。将待存数据a送入00FEH单元，00FEH单元成为新栈顶。

（先修改堆栈指针后再压入数据）

（3）间接寻址及其变形

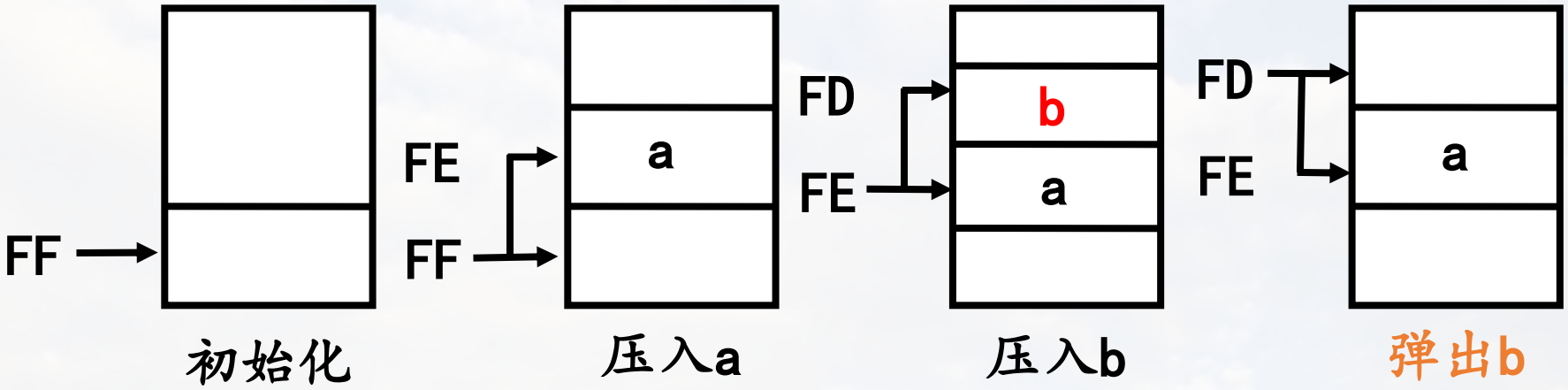


3）压入第二个数据元素b

a. SP－1→SP。SP内容由00FEH修改为00FDH。

b. 压栈。将待存数据b送入00FDH单元，00FDH单元成为新栈顶。

（3）间接寻址及其变形



4）弹出

a. 将栈顶单元最后压入的数据b读出，送入指定位置；

b. (SP)＋1→SP。弹出数据后再修改堆栈指针，让SP内容加1，由00FDH修改为00FEH，SP指向新栈顶。

（3）间接寻址及其变形

例如：程序定义堆栈区及压栈/出栈操作……

定义堆栈区长度

```
STACK1   SEGMENT PARA STACK
    DW   100  DUP（0）          ；长度100（64H）
STACK1   ENDS
```

……

压栈操作

```
PHSH   AX                    ；入栈
PUSH   DS
PUSH   DATA-WORD
PUSHF
```

……

出栈操作

```
POPF                         ；出栈
POP    DATA-WORD
POP    DS
POP    AX
```

（3）间接寻址及其变形

⑥多重间接寻址(主存多重间接寻址)方式(自学、不要求)

上述间址方式均只有一层间址。

有的机器允许多重间址，即根据指令找到间址单元，其中的内容还不是操作数地址，而是下一层间址单元的地址；根据该地址访问下一层间址单元，取出来的才是操作数地址(存放操作数的存储单元的地址码)。

(3) 间接寻址及其变形

怎么知道从存储单元中读出的是有效操作数地址还是间接地址呢?

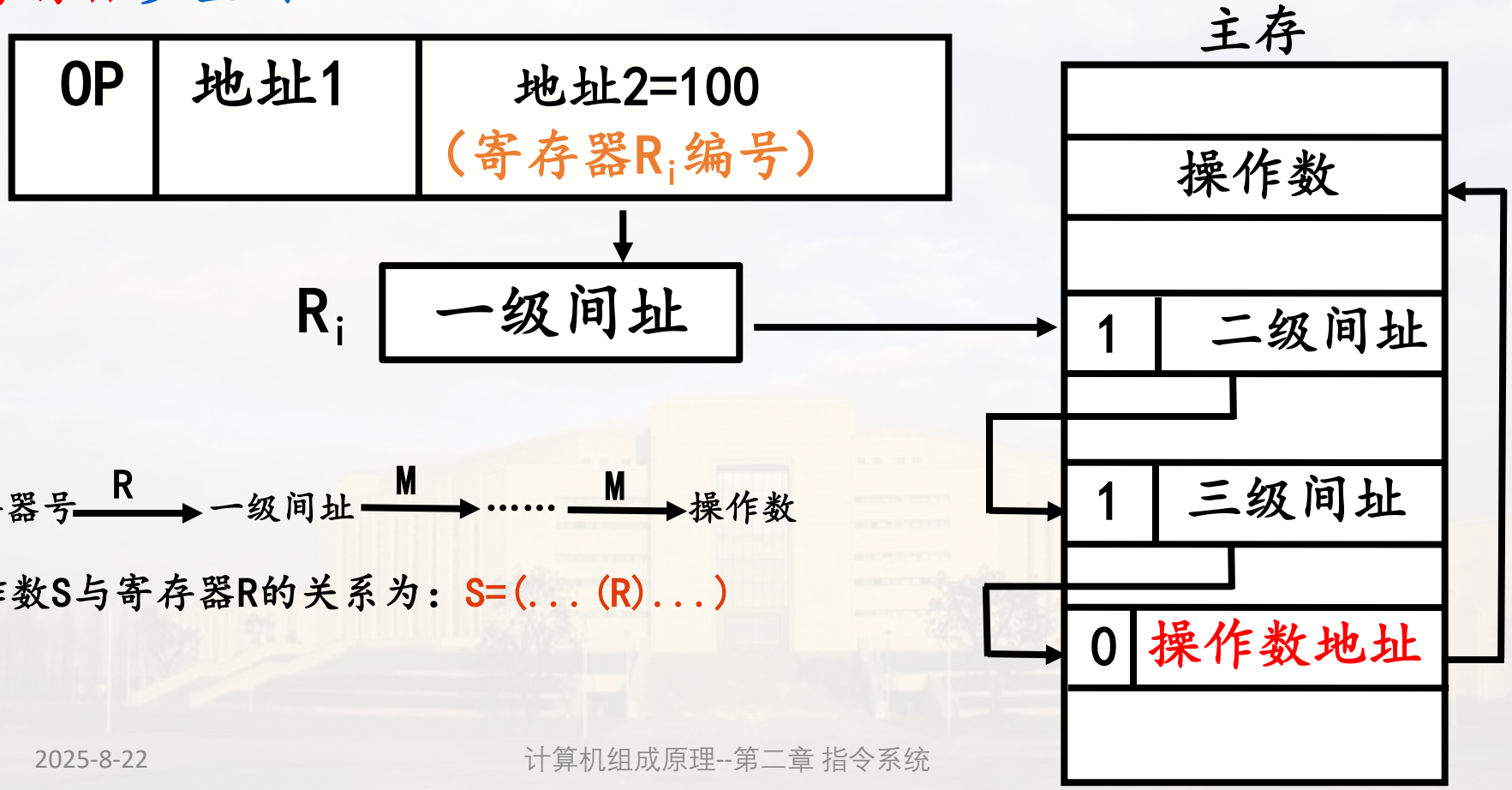
可在间址单元的存储内容中设置一位间址标志位,一般选取最高位。

当该位为1时,表明所读出的是间接地址,还需再次间址;直到该位为0,表明这次取出的是操作数的有效地址,按这个地址访问主存,读出的是操作数(即间址过程结束)。

（3）间接寻址及其变形

多重间址有分为寄存器多重间址与存储器多重间址。

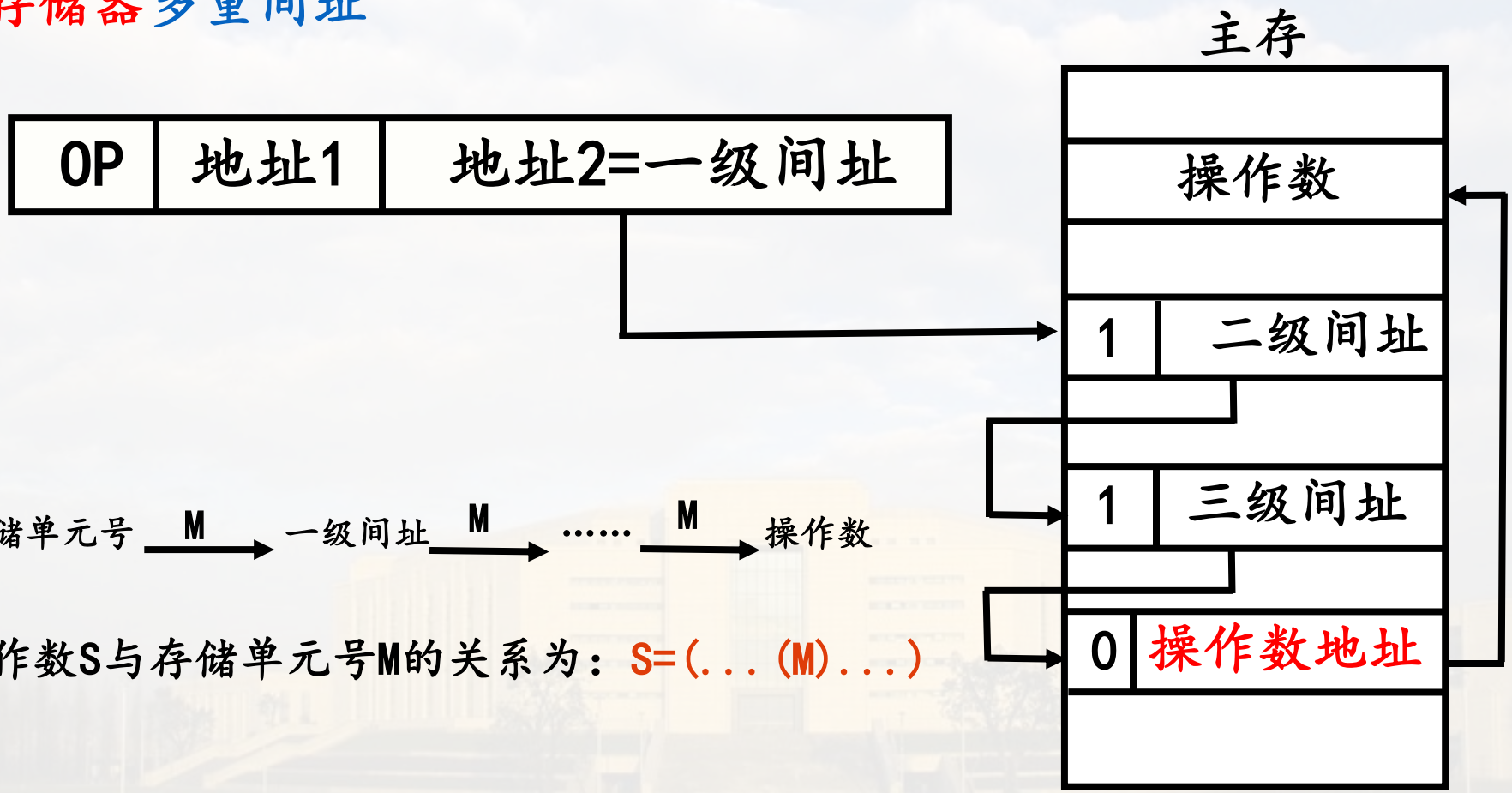
寄存器多重间址



操作数S与寄存器R的关系为：S=(...(R)...)

（3）间接寻址及其变形

存储器多重间址



操作数S与存储单元号M的关系为：S=(...(M)...)

(3) 间接寻址及其变形

1) 寄存器多重间址:

寄存器号 $\xrightarrow{R}$ 一级间址 $\xrightarrow{M}$ …… $\xrightarrow{M}$ 操作数

2) 存储器多重间址:

一级间址 $\xrightarrow{M}$ 二级间址 $\xrightarrow{M}$ …… $\xrightarrow{M}$ 操作数

多重间接寻址产生地址的方法提供了**编程的灵活性**,但多重间址方式**增加了访存次数**，因而极大的减慢了计算机工作速度,所以现在已很少采用。

(4) 变址、基址寻址及其变化

通过地址计算使地址灵活可变。

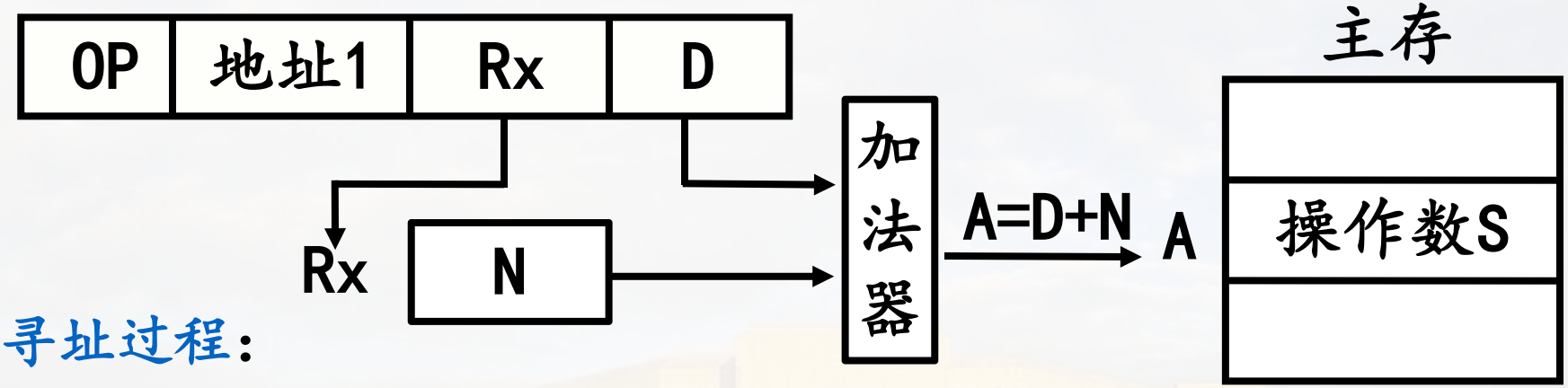
① 变址寻址：

若指令中给出变址寄存器号和一个形式地址，变址寄存器的内容(称为变址量)与形式地址相加，得到操作数有效地址(即操作数实际地址)，按照有效地址访问某主存单元，该单元的内容即为操作数，这种寻址方式称为变址寻址方式。变址方式常用助记符X(R)表示。

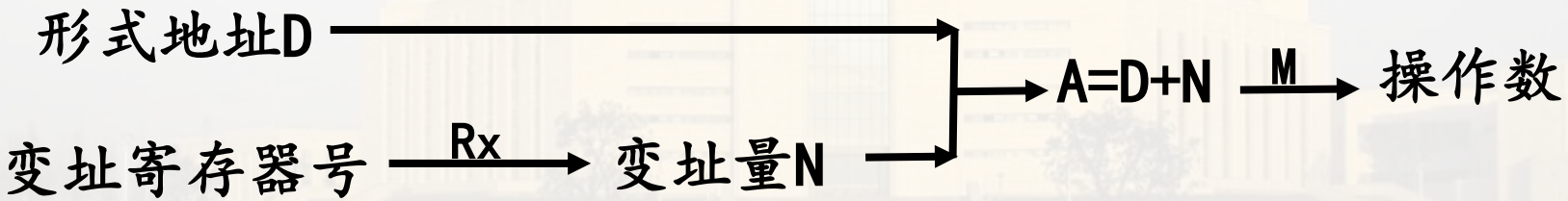
在8086/8088中使用SI、DI（隐含使用DS段）

（4）变址、基址寻址及其变化

指令中为获得某个操作数地址给出两个信息：形式地址D，变址寄存器Rx。有效地址A=D+(Rx)=D+N，根据A访问主存储器，读写操作数S。



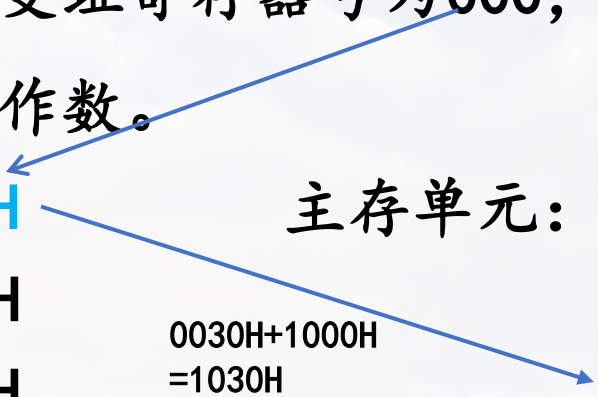
寻址过程：



操作数S与形式地址D、变址寄存器Rx的关系为：S=((Rx)+D)=(N+D)

（4）变址、基址寻址及其变化

例：若指令中给出变址寄存器号为000，形式地址为1000H，按变址方式读取操作数。



变址寄存器是R_0，则变址量为N=（R_0）=0030H；

形式地址D=1000H，则变址计算：

$A=D+(R_0)=D+N=1000H+0030H=1030H$；

据此访问主存储器，读得操作数S=(A)=2C00H。

（4）变址、基址寻址及其变化

例如：

```
MOV   AX, 10H[SI]
;等价于 MOV   AX, DS: 10H[SI]
MOV   AL, 20H[DI]
;等价于 MOV   AL, DS: 20H[DI]
```

（4）变址、基址寻址及其变化

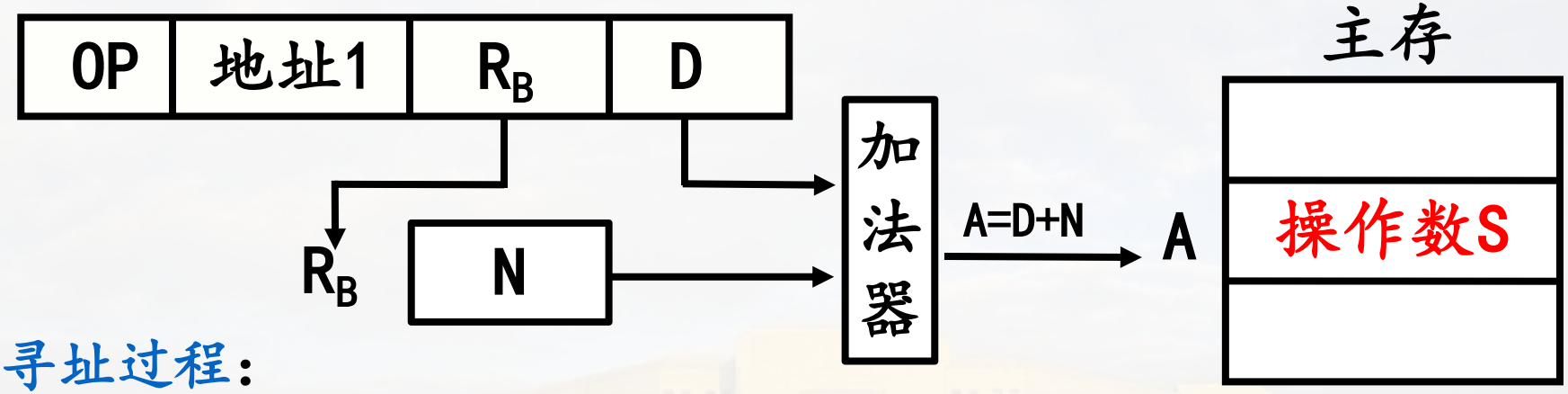
② 基址寻址：（计组部分不要求，但微机原理部分要求）

若指令中给出基址寄存器号和一个形式地址，基址寄存器内容（作为基准地址）与形式地址（作为位移量）相加，其和为操作数有效地址(即操作数实际地址)，按照该地址访问主存储器，该单元的内容即为操作数，这种寻址方式称为基址寻址。

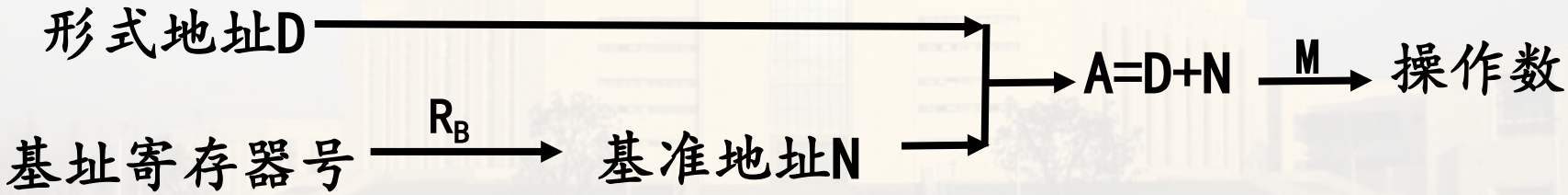
在8086/8088中使用BX（隐含使用DS段）、BP（隐含使用SS段）

（4）变址、基址寻址及其变化

指令中为获得某个操作数地址给出了两个信息：形式地址D，基址寄存器R_B。有效地址A= D+(R_B)= D+N，根据A访问主存储器，读写操作数S。



寻址过程：

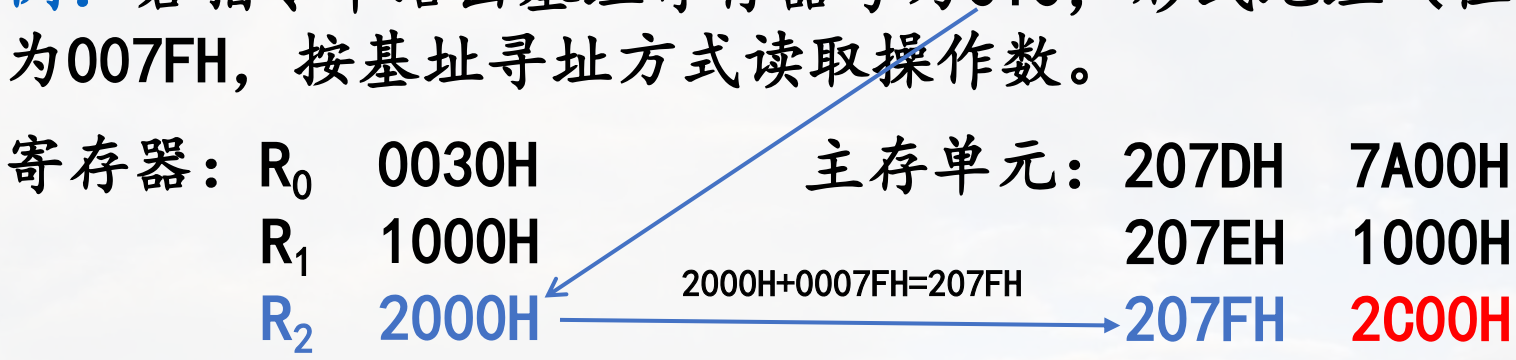


操作数S与形式地址D、变址寄存器R_B的关系为：S=((R_B)+D)=(N+D)

（4）变址、基址寻址及其变化

在CPU中有专用的基址寄存器，或者由程序指定某个通用寄存器担任基址寄存器。

例：若指令中给出基址寄存器号为010，形式地址（位移量）为007FH，按基址寻址方式读取操作数。



基址寄存器为R_1，则基准地址为$N=(R_1)=2000H$；
位移量为$D=007FH$，则基址计算：

$A=D+(R_1)=D+N=007FH+2000H=207FH$；

据此访问主存储器，读得操作数$S=(A)=2C00H$。

(4) 变址、基址寻址及其变化

在某些大型机中，基址寄存器只能由特权指令来管理，用户指令无权操作和修改。在某些小，微型计算机中，基址和变址寻址实际上是合二为一的(如：Intel 8086/8088)。

例如：MOV DX, VAR[BP]

; 等价于MOV DX, SS: VAR[BP]

MOV AX, 10H[BX]

;等价于 MOV AX, DS: 10H[BX]

（4）变址、基址寻址及其变化

	变址寻址	基址寻址
相同点	有效地址计算方法相同； 在一些计算机中，都是由相同硬件实现。	
不同点	变址寄存器提供修改量（可变的），而指令中提供基准量（固定的）	基址寄存器提供基准量（固定的），而指令中提供位移量（可变的）
	面向用户的，用于访问字符串，向量和数组等成批数据	面向系统，主要用于逻辑地址和物理地址的转换，用以解决程序在主存中的再定位和扩大寻址空间等问题

（4）变址、基址寻址及其变化

③ 基址加变址方式(计组部分不要求，但微机原理部分要求)

基址寻址方式的目的是扩大有限字长指令的寻址空间，变址寻址方式的目的是为了灵活修改地址以适应连续区间(程序循环)的操作。

如果在同一条指令中要兼有这两种功能，可以采取复合型的寻址方式，即基址加变址方式。

(4) 变址、基址寻址及其变化

例：某商场的销售金额汇总表如表所示，采用基址加变址方式查询某天的销售额。

表 商场销售金额统计示意表

月 \ 日 / 金额（万）	1	2	……	17	……	30	31
1	100	100		80		60	80
2	50	60		100		60	70
·····	·····	·····					
6	60	80		90		80	70
·····	·····	·····					
12	100	100		100		90	100

（4）变址、基址寻址及其变化

例：某商场的销售金额汇总表如表所示，采用基址加变址方式查询某天的销售额。假定：存储首址为1000H（内容为1月1日销售金额），每天的销售金额存放在一个主存单元之中，为每个月份分配31个单元。

表 商场销售金额统计示意表

月 \ 日 / 金额（万）	1	2	……	17	……	30	31
1	100	100		80		60	80
2	50	60		100			
6	60	80		90		80	
12	100	100		100		90	100

运用基址加变址的寻址方式查找6月17日的销售金额。

（4）变址、基址寻址及其变化

若：

指定R_0为基址寄存器，其中存放表格的首址1000H，作为基准地址；

指定R_1为变址寄存器，其中存放的变址量为：从表的首址到六月份这一行的起始单元之间的距离，即

$(5\times31)_{10}=(155)_{10}=(10011011)_2=009BH$；

位移量为从该行起点到17日这一列的距离，即0011H。

则有：操作数有效地址=1000H+009BH+0011H=10ACH

（4）变址、基址寻址及其变化

通过修改基址，还可以实现程序段在存储空间中的重新定位。

现在，变址加基址的寻址方式广泛应用于许多计算机中，包括微型计算机。

例如：MOV　AX，10[BX][SI]

；等价于MOV　AX，DS：10[BX][SI]

MOV　DX，VAR[BP][SI]

；等价于MOV　DX，SS：VAR[BP][SI]

（4）变址、基址寻址及其变化

④ 相对寻址（浮动编址）（自学，不要求）

变址寻址的变形，助记符为X(PC)

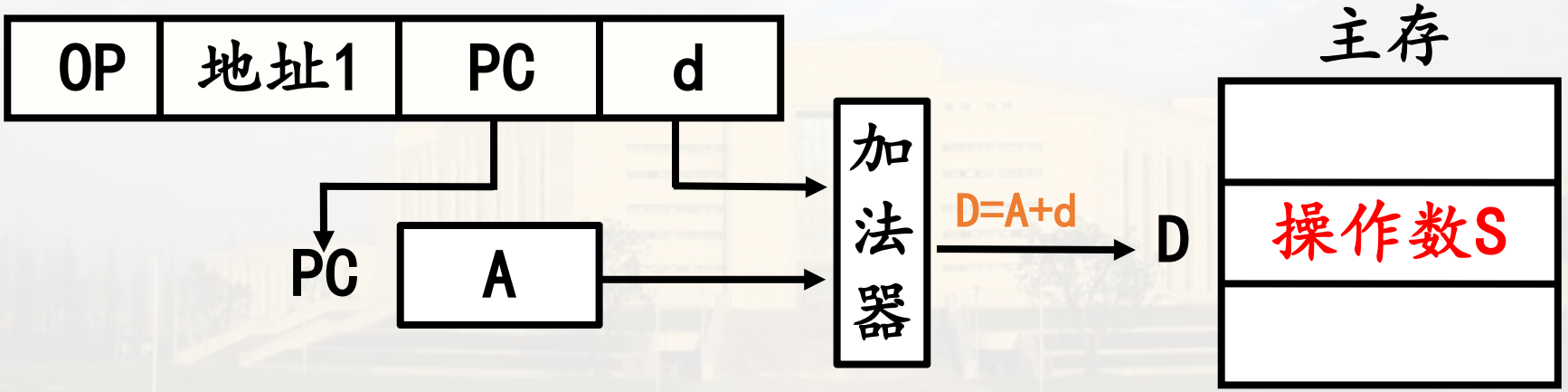
若指令中选定程序计数器PC作为变址寄存器，或是隐含地指定PC，指令中给出的形式地址作为位移量（可正、可负），二者相加后形成操作数的有效地址。这种寻址方式实际上就是以当前指令位置为基准，相对它进行位移定位（往前或往后），所以称为相对寻址。

(4) 变址、基址寻址及其变化

程序计数器PC的内容为现行指令地址A，按地址A从主存中读取指令；

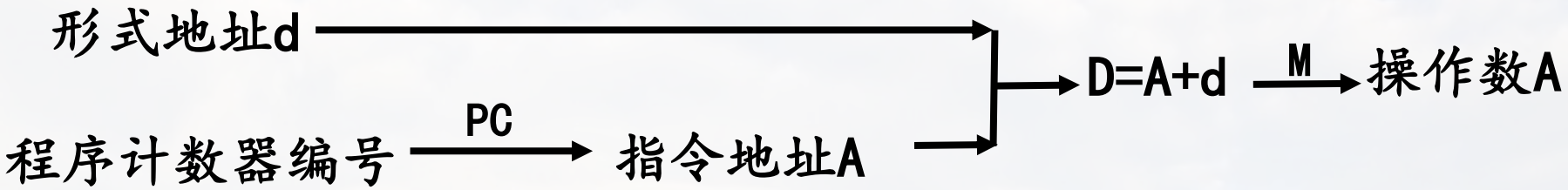
指令中形式地址段给出位移量d，它是从现行指令位置到操作数S所在单元之间的距离（单元数）；

操作数有效地址D=A+d，据此访问主存储器，从D单元中读取操作数。



（4）变址、基址寻址及其变化

寻址过程：



操作数S与形式地址d 、变址寄存器PC的关系为：

$$S=((PC)+d)$$

（PC）即变址量A作为基准地址来看待。

（4）变址、基址寻址及其变化

例：若从1000H单元中取出一条指令，该指令采用相对寻址方式读取操作数，形式地址（位移量）为0003H。

寄存器：PC 1000H　　主存单元：0FFDH BC00H

R_0 2000H　　1003H AF00H

R_1 3000H

现行指令存放在（PC）= 1000H单元中，则基准地址为1000H；

指令给出位移量为d=0003H，

则操作数有效地址=(PC)+d=1000H+0003H=1003H，

据此访问主存，操作数为S=（（PC）+d）=(1003H)=AF00H。

（4）变址、基址寻址及其变化

例：若从1000H单元中取出一条指令，该指令采用相对寻址方式读取操作数，形式地址（位移量）为-0003H。

寄存器：		主存单元：	
PC	1000H	OFFDH	BC00H
R_0	2000H	1003H	AF00H
R_1	3000H		

现行指令存放在（PC）=1000H单元中，则基准地址为1000H；指令给出位移量为d=-0003H，

则操作数有效地址=(PC)+d=1000H+（-0003H）=0FFDH，

据此访问主存，操作数为S=（（PC）+ d）=(0FFDH)=BC00H。

（4）变址、基址寻址及其变化

⑤ 页面寻址(自学，不要求)

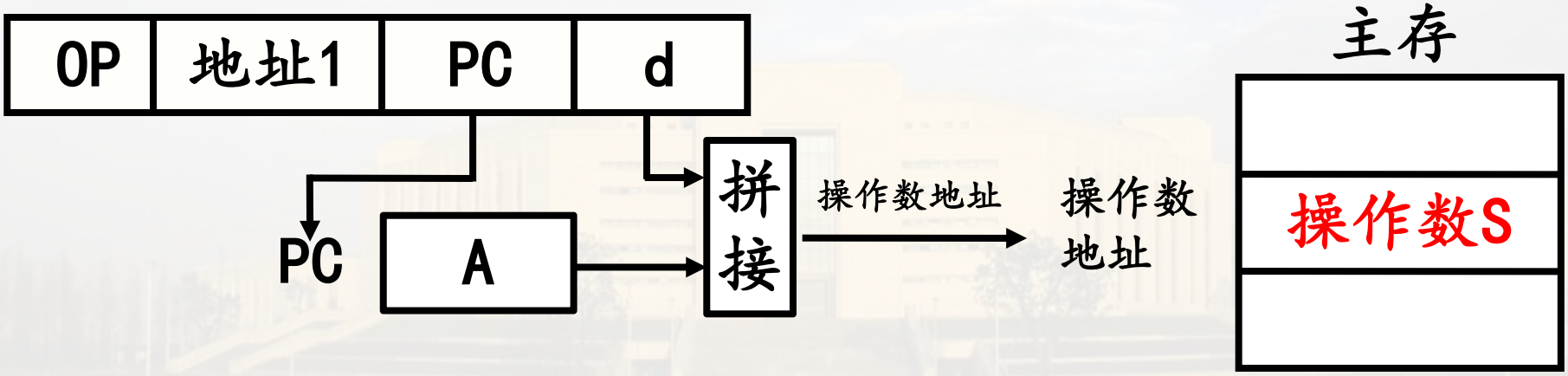
若不是将PC内容与位移量进行算术加，而是将PC内容的高位段与位移量相拼接，相对寻址就演变成页面寻址。

（4）变址、基址寻址及其变化

程序计数器PC的内容为A，指令中形式地址段给出位移量d。

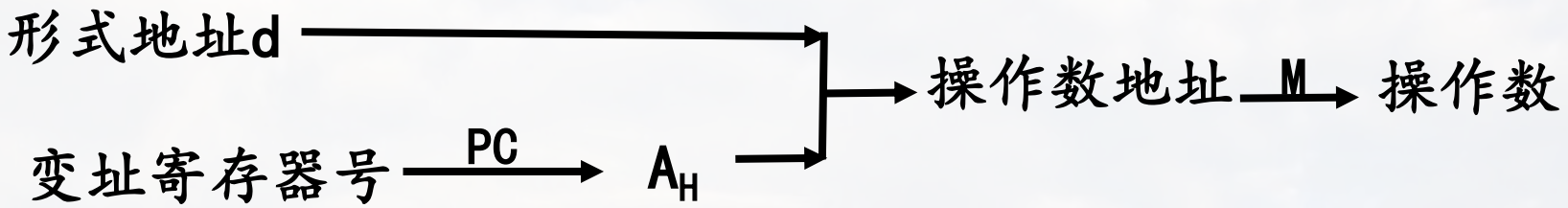
按页面寻址方式，操作数有效地址=（PC）$_H$，d；

据此访问主存储器，从单元中读取操作数。



（4）变址、基址寻址及其变化

寻址过程：



操作数S与形式地址d、变址寄存器PC的关系为：

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

（4）变址、基址寻址及其变化

例：若从1030H单元中取出一条指令，该指令采用页面寻址方式读取操作数，形式地址为FFH。

寄存器：PC 1030H　　主存单元：10FFH ACOOH

R_0 2000H　　1100H 7FC0H

R_1 3000H

PC的内容为1030H，其高8位为10（低8位为30），与形式地址FFH相拼接，得到操作数有效地址10FFH，从主存储器中得到操作数ACOOH。

（4）变址、基址寻址及其变化

页面寻址方式适合于采取页面管理的存储组织。

例如:某机主存容量1MB，分为 1K页,每页1KB,则取PC内容的高10位作为页面号(它也指明了现在程序运行在哪个页面），指令中提供10位的位移量(相对于页的起点），也就是页内单元地址。二者拼接为20位有效地址。

(5) 小结

以上四类十余种寻址方式，关键点是“数在哪里”（在指令中、在CPU寄存器中、在主存中）。

① 如果操作数在主存中，指令直接给出有效地址还是通过“多次读取”间接获得有效地址(通过寄存器间址、通过存储单元间址)？

② 如何通过计算使地址量可变（与变址寄存器内容加、与基址寄存器内容加、与程序计数器内容加或拼接)？

2.3 指令类型

- 01. 指令分类
- 02. 传送类指令
- 03. 输入/输出（I/O）指令

一、指令分类

RISC（Reduced Instruction Set Computer）
CISC（Complex Instruction Set Computer）

对指令的分类方法归纳起来大致有以下三类：

①按指令格式分类

按指令格式分为双操作数指令、单操作数指令、程序转移指令等。

②按操作数寻址方式分类

RR型（寄存器—寄存器型） RX型（寄存器—变址存储器型）

RS型（寄存器—存储器型） SI型（存储器—立即数型）

SS型（存储器—存储器型）

③按指令功能分类

现在的大部分微处理器，将指令分为：

传送指令

输入/输出（I/O）指令

算术运算指令

逻辑运算指令

程序控制类指令

处理机控制类指令等

（1）一般传送指令

是计算机中最基本的指令。机器的绝大部分操作，不管是访问存储器的操作，或是算术/逻辑运算操作，或是输入/输出操作等等，从广义的角度来看，基本上都可以归结为信息的传送，或在传送过程中作了某种处理（包括算术/逻辑运算）。

单纯的传送指令将数据从一个位置（源地址）传送到另一个位置（目的地址）。

（1）一般传送指令

在具体设置传送指令时，一般应当对以下三个方面做出说明：

①传送范围，即指令允许数据在什么范围内传送。如前所述，操作数的来源与目的地主要是：CPU寄存器、主存储器。

1）CPU寄存器之间的数据传送。将源寄存器的内容传送到另一个目的寄存器，即$R_i \rightarrow R_j$。

如：MOV R_j，R_i。

（1）一般传送指令

2）主存储器单元之间的数据传送。将主存单元的内容传送到主存的另一目的单元，即$M_1 \rightarrow M_2$。

如：MOV　mem2，mem1。（实际上未有该指令）

3）从CPU寄存器传送到主存单元。将源寄存器的内容传送到主存某一目的单元，即存入数据操作，即$R_i \rightarrow M$。如：MOV　mem，reg。

在有些计算机中，该指令用助记符STORE表示。

（1）一般传送指令

4）从主存单元传送到CPU寄存器。将主存某一单元的内容传送到目的寄存器，即读出数据操作，即$M \rightarrow R_j$。

如：MOV reg , mem。

在有些计算机中，该指令用助记符LOAD表示。

②传送单位，即数据可以按字节、字、双字或数组为单位进行传送，因此传送指令应该用某些方式指明数据传送的单位。

（1）一般传送指令

③ 设置寻址方式。

有的计算机为各种寻址方式分类编号，如本书模型机的0型、1型、2型、3型、4型、5型、6型等。

（2）堆栈指令

堆栈指令实际上是数据传送指令的一种特殊情况，分为压栈（PUSH）和出栈（POP）两种，在编写程序时，它们一般是成对出现的。若是在主存某一段区域开辟的一个堆栈区，则：

压栈：将数据压入堆栈栈顶，可视为存入数据到主存某一单元的一个特例。

出栈：从堆栈栈顶弹出数据，可视为读出主存某一单元中数据的一个特例。

（3）数据交换指令

前述的指令传送方向都是单向的，然而，数据传送也可以是双向的，即将源操作数与目的操作数（一个字节或一个字）相互交换位置。

如：XCHG　AX，BX

　　XCHG　[2000H]，CL

三、输入/输出(I/O)指令

从广义的角度来看，I/O指令也是一种传送指令，只是传送范围的一方固定为输入/输出（I/O）设备，如键盘、鼠标、显示器、打印机等。主机对外围设备的访问一般就是对有关接口寄存器的访问。

三、输入/输出(I/O)指令

I/O指令在设置上是比较灵活的一类，其原因如下：

①主机所连接的外围设备种类和数量都可能发生变化，因此，I/O指令应有足够的扩展余地，以适应不同的应用场合。

② 外围设备种类甚多，它们所需的具体控制命令，以及在工作中所产生的状态信息，差别很大；但指令系统往往是通用的。

(1) 外围设备编址

访问外围设备，首先要知道其地址，这就涉及到对设备的编址问题。对外围设备的编址方法一般有两类三种。

①对外围设备单独编址

②外围设备与主存储器统一编址

① 对外围设备单独编址：两种

1）单独编址到设备级：

早期的做法是为每台外围设备分配一个设备码，该设备的接口中设置有限的几个寄存器，例如寄存器A、B、C。

在I/O指令中给出设备码，并指明是哪个寄存器。

① 对外围设备单独编址

2）单独编址到寄存器级：

现在普遍采用的方法是：为各I/O接口中的有关寄存器分配一种I/O端口地址，即编址到寄存器一级。

各台设备有自己的接口，一个接口可以占有若干个I/O端口地址，各接口所占有的端口地址数目可以不同，因此，只要送出某个端口地址，就能知道选中了哪一个接口中的哪一个寄存器，也就知道选中了哪台设备。

① 对外围设备单独编址

2）单独编址到寄存器级：

端口地址：8位，即2^8，0∽255，00H∽FFH，采用直接寻址

16位，即2^{16}，256∽64K，0100H∽FFFFH，采用间接寻址

例1：IN　AX，0CH　　；

0CH表示端口号：即地址

采用直接寻址，输入端口地址0CH的一个字到AX中

例2：MOV　DX，02ECH

IN　AX，DX　　；　（IN　AL，DX ）？对否

DX表示端口地址02ECH

采用间接寻址，输入端口地址02ECH的一个字到AX中

② 外围设备与主存储器统一编址

统一编址到寄存器级，具体做法是将每个外围设备接口中的有关寄存器视作一个主存单元，分配一个存储单元地址(总线地址)。

一个接口视其需要可占用一个或多个总线地址，根据指令给出的地址码，可以判明是访问主存还是访问外围设备，是访问哪一个接口的哪一个寄存器。

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等信号线	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

(2) I/O指令的设置方法

通常有三类常见的I/O指令设置方法，一台计算机可以选取其中的一种或数种。

① 设置专用的I/O指令：IN，OUT

② 采用通用的数据传送指令实现I/O操作：MOV指令

③ 通过I/O处理器（或I/O处理机）控制I/O操作

三、输入/输出(I/O)指令

① 设置专用的I/O指令：IN，OUT

如果外围设备单独编码（设备码、或端口地址），指令系统中有专门的I/O指令，这是明显的，又称为显式I/O指令。

1）如采用设备编码方式，则I/O指令的地址码部分应给出所要访问的外围设备的编码，并指定所访问的寄存器。

三、输入/输出(I/O)指令

① 设置专用的I/O指令：IN，OUT

2）如果采用I/O端口地址编码方式，则I/O指令的地址段给出端口地址，并指定CPU中寄存器号，操作码规定输入或输出操作类型。

端口地址：8位，即2^8，0∽255(十进制)，00H∽FFH(十六进制)，采用直接寻址

16位，即2^{16}，256∽64K(十进制)，0100H∽FFFFH(十六进制)，采用间接寻址

```
直接寻址：OUT   20H,AL      ；20H表示端口号
间接寻址：MOV   DX,83FCH    ；83FCH表示端口号
          IN    AL,DX       ；采用间接寻址，输入
                            一个字节到AL中
```

② 采用通用的数据传送指令实现I/O操作

目前，许多计算机采用通用的数据传送指令实现I/O操作，相应地将外围设备（I/O接口寄存器）与主存单元统一编址。

如果传送指令的源地址是CPU寄存器，而目的地是接口寄存器，则这条传送指令就是一条输出指令。例如传送指令

MOV n，R_0；

如果传送指令的源地址是接口寄存器，而目的地是CPU中的寄存器，则是一条输入指令，例如MOV R_0，n。

② 采用通用的数据传送指令实现I/O操作

若将外围设备接口中的寄存器与主存单元统一编址，由同样的总线地址访问。这样，外设便可看作是总线地址所覆盖的存储空间的一部分，因而主机可以通过传送指令用相同的方式访问存储器和外围设备。对存储器的各种寻址方式同样适合于对外设的寻址，使编制程序灵活方便。这种I/O指令是隐含在传送指令之中的，所以又称为隐式I/O指令。

③ 通过I/O处理器（或I/O处理机）控制I/O操作

CPU的I/O指令（不管是显式还是隐式）都是通用的，它们一般并不专门针对某一台具体的设备或某一种具体的操作。因为一台计算机系统中究竟连接哪些外围设备、有哪些具体操作，变化是很多的，而且是计算机系统设计者事先无法确定的。

解决的办法就是用I/O指令向接口发送命令字，再转化为与具体设备相匹配的命令。对不同的设备，命令字的约定不同。

③ 通过I/O处理器（或I/O处理机）控制I/O操作

对命令字/状态字的设置问题：

命令字：启动位、校验位、维护位、允许中断位等。

状态位：忙位、完成位、空闲位、故障位、效验出错位等。

第二章 复习提纲

一. 指令信息的表示

1. 操作数的位置

2. CPU能直接访问的操作数位置

3. 指令给出操作数地址方式：显式、隐式

4. 寻址方式

4. 寻址方式

大致可将众多的寻址方式归纳为以下四大类，其它的寻址方式则是它们的变型或组合。

① 立即寻址。在读取指令时也就从指令之中获得了操作数，即操作数包含在指令中。

② 直接寻址类。直接给出主存地址或寄存器编号，从CPU内或主存单元内读取操作数。

③ 间接寻址类。先从某寄存器中或主存中读取地址，再按这个地址访问主存以读取操作数。

④ 变址类。指令给出的是形式地址（不是最终地址），经过某种变换（例如相加、相减、高低位地址拼接等），才获得有效地址，据此访问主存储器以读取操作数。

二. 指令的分类

1. 传送类指令

2. 输入/输出（I/O）指令

谢谢观看

计算机组成原理

2025-8-22

信息与软件工程学院
School of Information and Software Engineering