计算机组成原理

2017年修订

西南交通大学信息科学与技术学院
唐慧佳　hjtang@home.swjtu.edu.cn

第3章　指令系统

第3章　指令系统

本章重点：

1. 理解指令的基本格式和基本操作种类，理解扩展操作码方法；
2. 掌握基本数据寻址方式和有效地址的确定方法，搞清楚每一种基本寻址方式的特点；
3. 存储器堆栈的概念及堆栈的进、出栈操作；
4. 精简指令集计算机的设计思想及其特点。

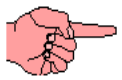
第3章　指令系统

指令：　执行某种基本操作的命令（如：加、减）。

指令是计算机硬件能够识别并直接执行的操作命令，
指示计算机硬件完成指定的基本操作。

程序：　由一系列有序指令构成。

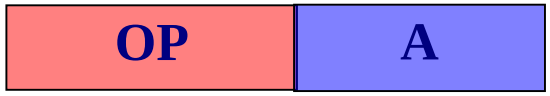
指令系统：一台计算机能执行的全部的指令的集合。



指令系统是软件和硬件的主要界面。
是设计计算机硬件的一个基本依据，
是软件设计者则编制程序的基础。

§3.1 指令格式

3.1.1 机器指令的基本格式



OP：操作码字段，指出所要进行的操作；

A：地址码字段，指出操作数和操作结果的地址。

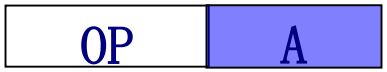
指令的长度：

1）定长．指令系统中所有的指令其长度都一样。

2）变长．各指令的长度可以不同。

3.1.2 地址码结构

地址码字段可以是单地址、双地址、三地址或零地址等格式。

单地址指令 

如：　AC←(AC) OP (A)　　AC为累加器，隐含方式
　　A←OP (A)

多地址指令 

如：　$A_3 \leftarrow (A_1)\ OP\ (A_2)$

零地址指令：操作数在栈顶和次栈顶中，或隐含指定。

例：完成 (X)+(Y) → Z的操作

用一条三地址指令即可----

ADD X,Y,Z;

用二条二地址指令实现----

ADD X,Y;　　(X)+(Y) → X

MOV Z,X;　　(X) → Z

用三条一地址指令实现----

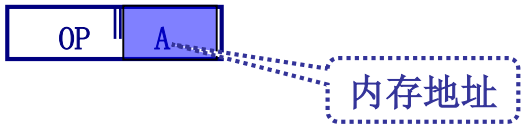
LDA X　　;　(X) → AC　　LDA 意为Load AC

ADD Y　　;　(AC) + (Y) → AC

STA Z　　;　(AC) → Z　　STA意为Store AC

地址段A_i的长度(N)与存储器容量(M)的关系：

$M=2^N$

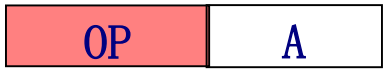


存在问题：① 地址段位数增长→指令过长；

② 程序设计的灵活性差。

∴ 需要采用好的寻址技术！

3.1.3 指令的操作码



1. 规整型(定长操作码、变长指令码)

n位操作码最多可表示2^n 种计算机指令.

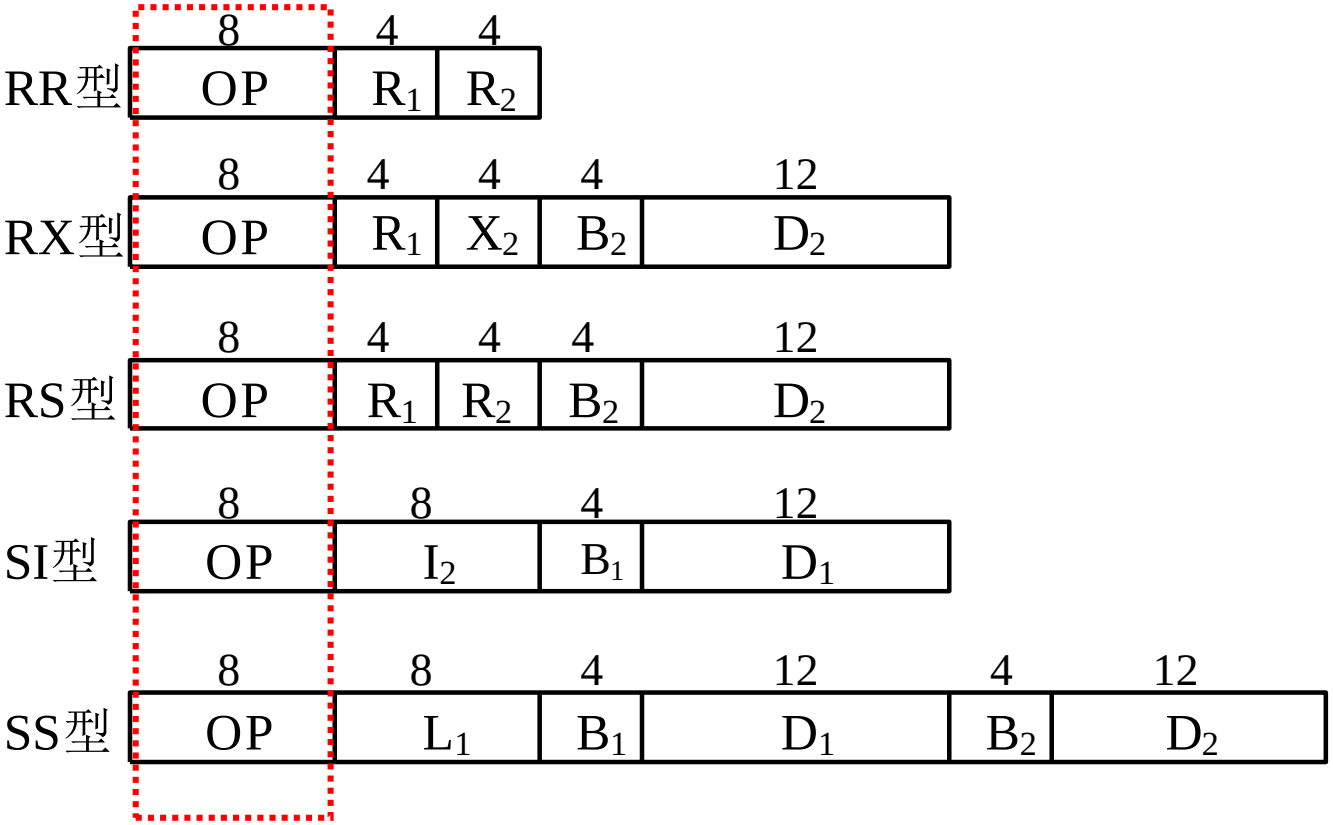
特点：操作码字段规整，译码简单、迅速。

指令的长度随操作数个数的不同而变化。

（适于大、中、小型机）

例：IBM 370机的指令格式

指令可分为几种不同的长度，不论指令的长度为多少位，其中操作码字段一律都是8位。



2. 非规整型(变长操作码、定长指令码)

可采用扩展操作码技术。

0000 xxxx xxxx xxxx
0001 xxxx xxxx xxxx
... ...
1110 xxxx xxxx xxxx

4 位操作码
15 条三地址指令

1111 0000 xxxx xxxx
1111 0001 xxxx xxxx
... ...
1111 1110 xxxx xxxx

8 位操作码
15 条二地址指令

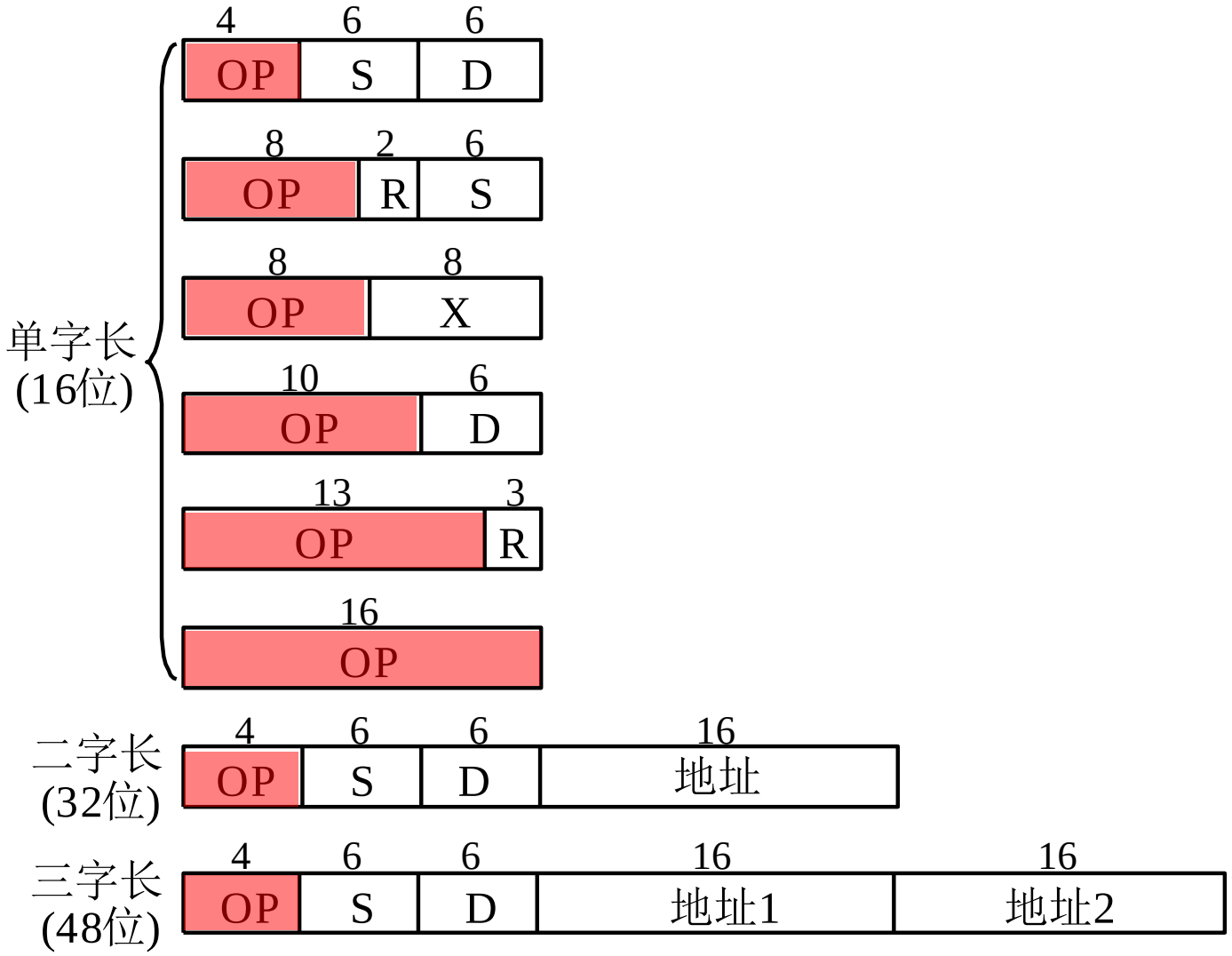
1111 1111 0000 xxxx
1111 1111 0001 xxxx
... ...
1111 1111 1110 xxxx

12 位操作码
15 条一地址指令

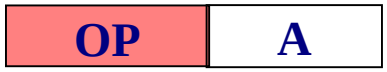
1111 1111 1111 0000
1111 1111 1111 0001
... ...
1111 1111 1111 1111

16 位操作码
16 条零地址指令

例：PDP-11机的指令格式



§3.2 指令类型



与机器的用途、性能的总体要求有关。

通用型计算机其基本的操作种类有：

- 数据传送类指令
- 运算类指令
- 程序控制类指令
- 输入输出类指令（非必备）

其它种类指令还有：

- 串操作指令
- 数据转换指令
- 处理机控制指令
- 特权指令

§3.2 指令类型

3.2.1 数据传送类指令

1. 一般传送指令（复制）

把数据从源地址复制到目的地址中去。

常用助记符：MOV ， LOAD(LD) ， STORE

类型：寄存器→寄存器

寄存器→主存

主存→寄存器

主存→主存

§3.2 指令类型

3.2.1 数据传送类指令

2. 堆栈操作指令

进栈PUSH、出栈POP，在程序中它俩往往成对出现。

3. 数据交换指令

常见的有字节交换、字交换、高低半字节之间交换等。

常用助记符：XCHG

3.2.2 运算类指令

1. 算术运算类指令

＋，－，×，／，加1，减1，向量运算等。

ADD, SUB, MUL, DIV, INC, DEC, . . . （各种运算不一定都具备）

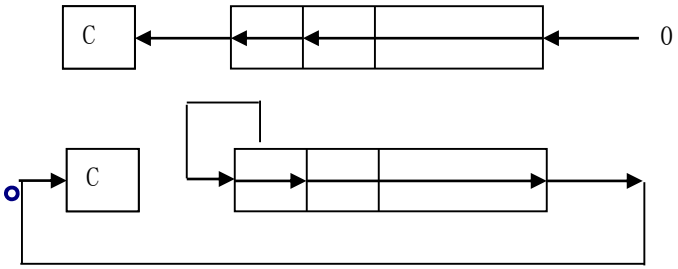
2. 逻辑运算类指令

与、或、取反、异或等（AND，OR，NOT，XOR，...）。

3. 移位类指令

算术移位(×2，÷2)

还有逻辑移位、循环移位等。



注：算逻运算除了产生运算结果外，还产生一些状态信息记录在状态寄存器PSW的标志位中。

例如：进位、溢出、全零、正负和奇偶标志等。

3.2.3 程序控制类指令

1. 转移指令

(1) 无条件转移

JMP ‖ A

功能：PC←A　　（PC为程序计数器）

(2) 条件转移

如：JC A　　（进位位标志C=1时转）

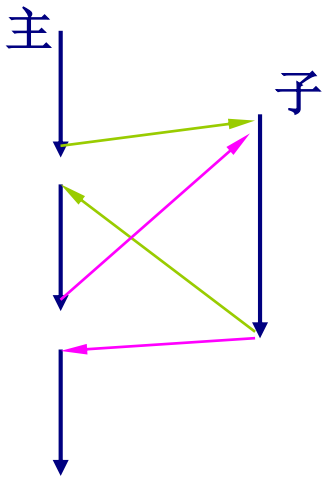
2. 子程序调用指令

CALL SUB_A

3. 返回指令

RET　返回主程序

RETI　中断返回

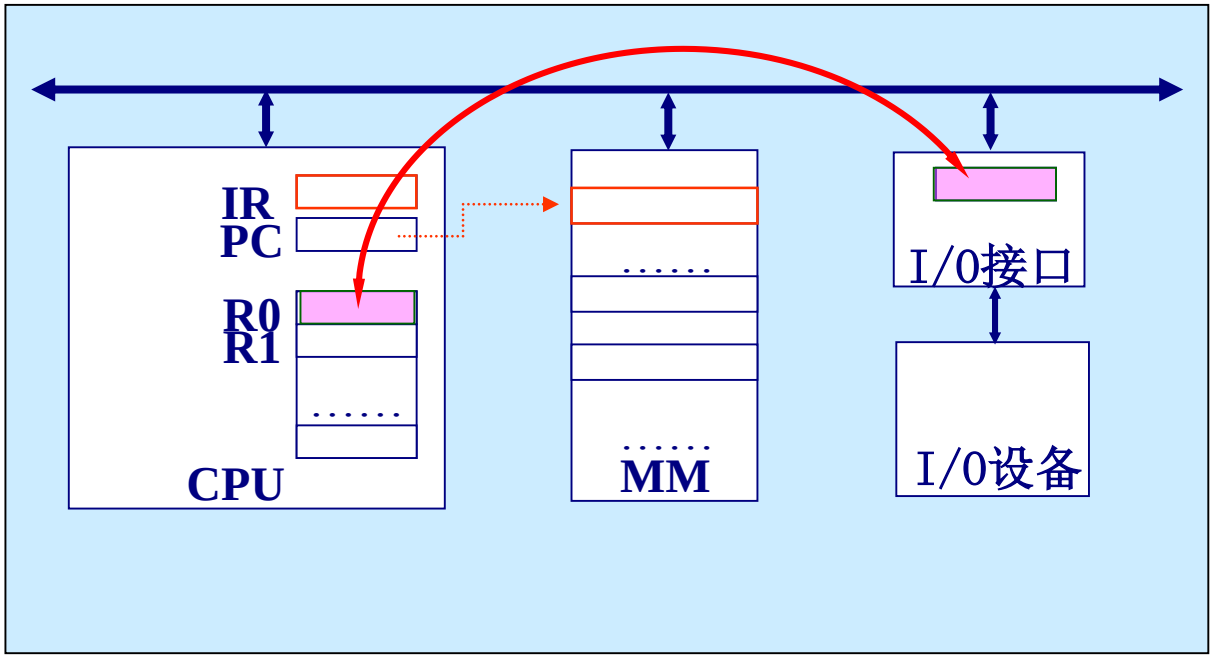


转子时要先保存好当前的PC值(通常是PC值压入堆栈)，然后再PC←SUB_A；

返回时要恢复PC，把栈中原PC的值弹出送给PC。

3.2.4 输入输出类指令

实现CPU与I/O设备之间的信息传送。



注：输入、输出都是对CPU而言的。

例如输入，指从I/O到CPU的数据传送。

3.2.4 输入输出类指令

输入指令如：

```
IN  A,[端口地址]        ; A是CPU中的寄存器
```

输出指令如：

```
OUT [端口地址],A        ; A是CPU中的寄存器
```

在I/O设备独立编址的计算机中，指令系统设有专门的输入/输出指令（IN/OUT）；而在I/O设备与内存统一编址的计算机中，输入输出由传送指令来实现，不设专门的输入/输出指令。

思考题：　P85 1，2，5，6

习题：　　P85　3，4

思考题：　P75　1，2，5，6，7，8，16

习题：P75　　3，4，10，12

§3.3 寻址技术

寻址：1)指令寻址
　　2)数据寻址

指令寻址：寻找下一条将要执行的指令地址。
　　通常采用顺序寻址(顺序执行)或跳跃寻址(转移)

数据寻址：寻找操作数的地址。
　　（本章重点）

寻址与存储单元的编址有关

3.3.1 编址

1. 编址单位

常见的编址单位有：

按字编址：编址单位=计算机字长

按字节编址：编址单位=1个字节

按位编址：编址单位=1bit

2. 指令中地址码的位数

主存容量越大，所需的地址码位数越长；

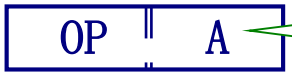
编址单位越小，所需的地址码位数越长。

3.3.2 基本的数据寻址方式

数据寻址方式：由指令中形式地址确定有效地址的方法。

以单操作数为例

无寻址技术时：



存储器地址（有效地址）

有寻址技术时：

形式地址

寻址方式

形式地址：指令中地址字段给出的地址。

（通常不能直接用来访问存储器）

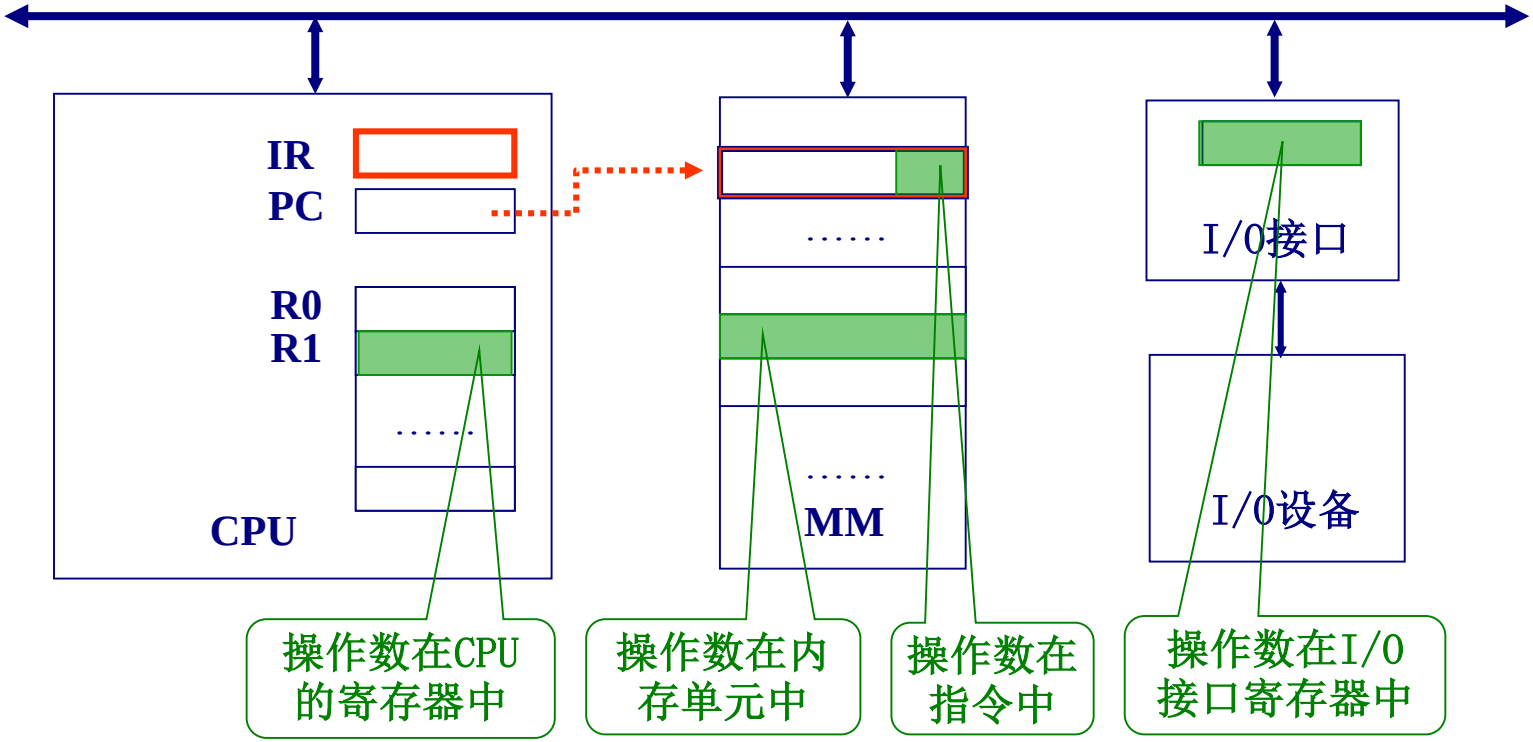
有效地址：形式地址经过一定的计算而得到的能直接访问存储器的地址。

形式地址 —寻址方式→ 有效地址

采用寻址方式的原因：

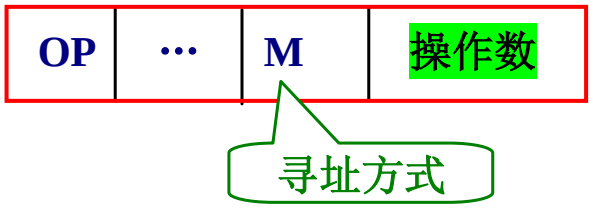
① 操作数地址表示多样化需要；

② 压缩操作数地址字段的长度。

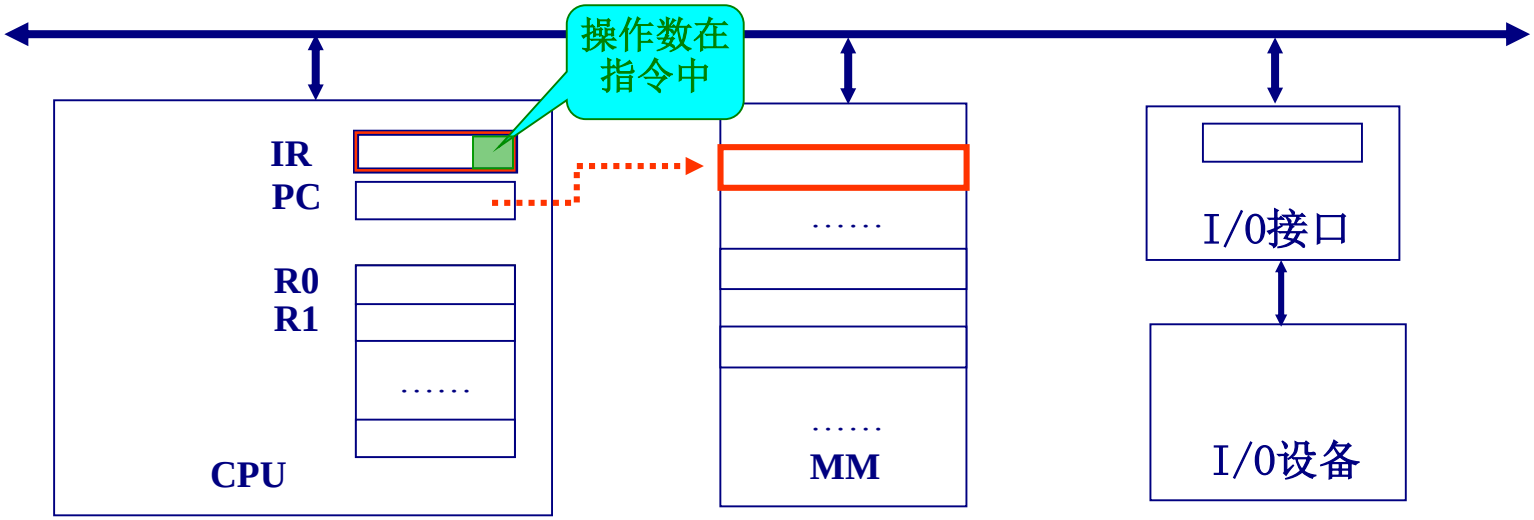


(1)立即寻址（Immediate Addressing）

指令中的地址字段存放的就是操作数。

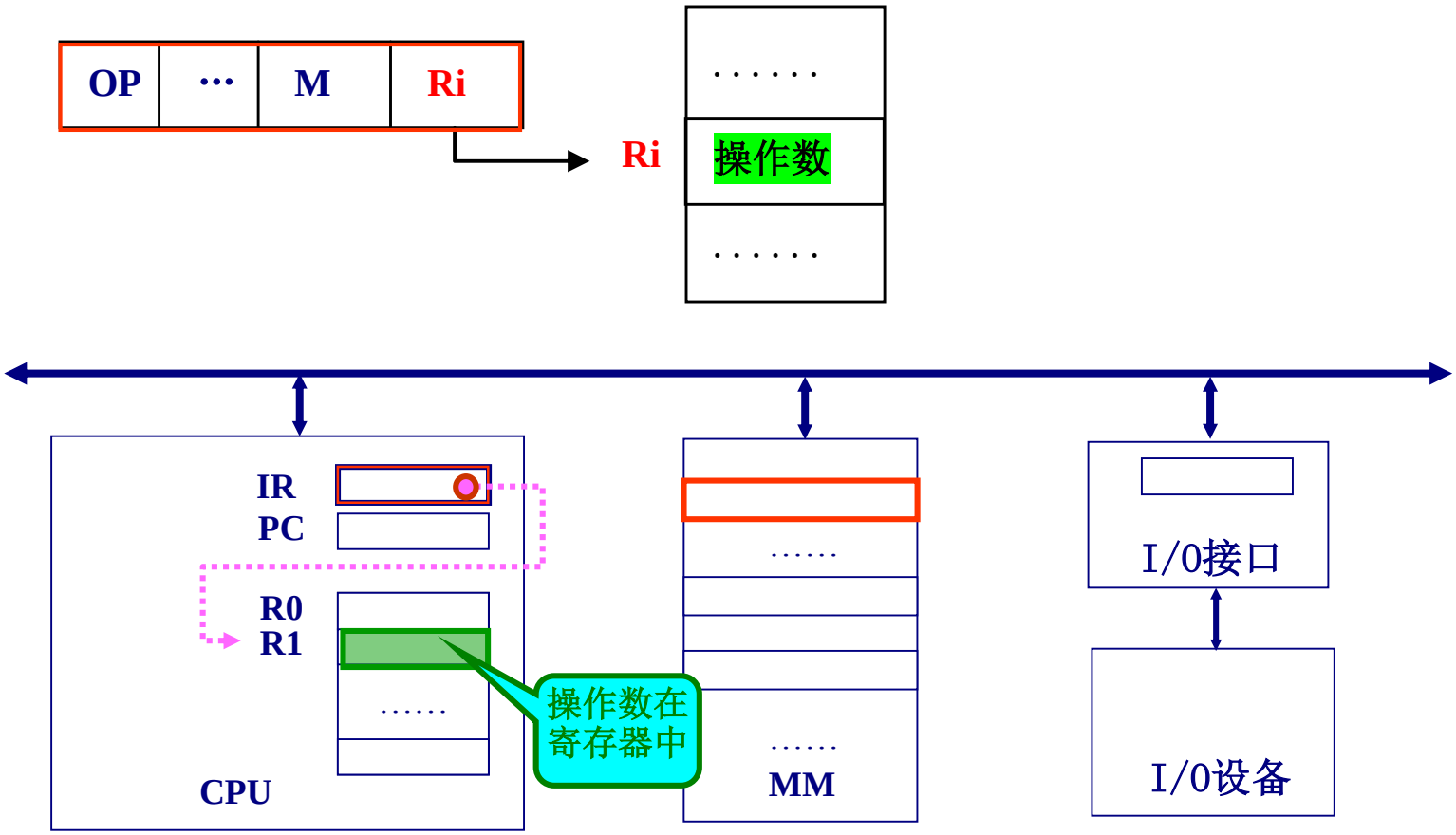


（取指令的同时取出数据）



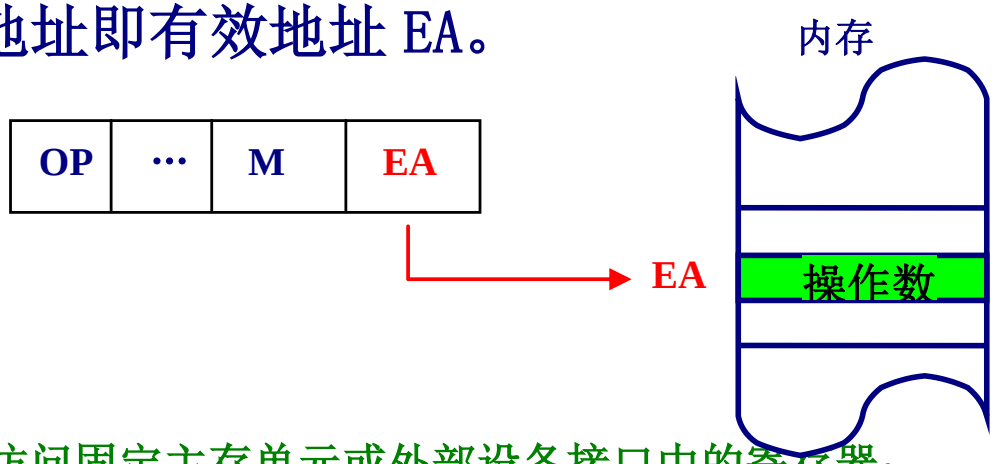
(2)寄存器寻址(Register Addressing)

操作数在某一通用寄存器（在 CPU）中。

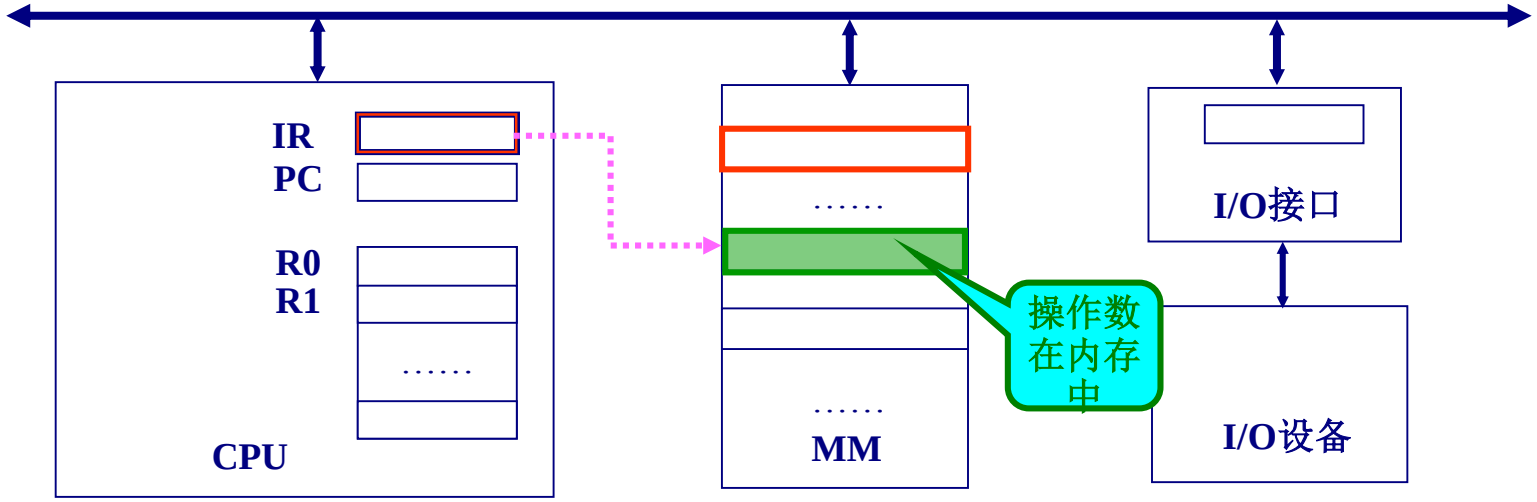


(3) 直接寻址(Direct Addressing)

形式地址即有效地址 EA。



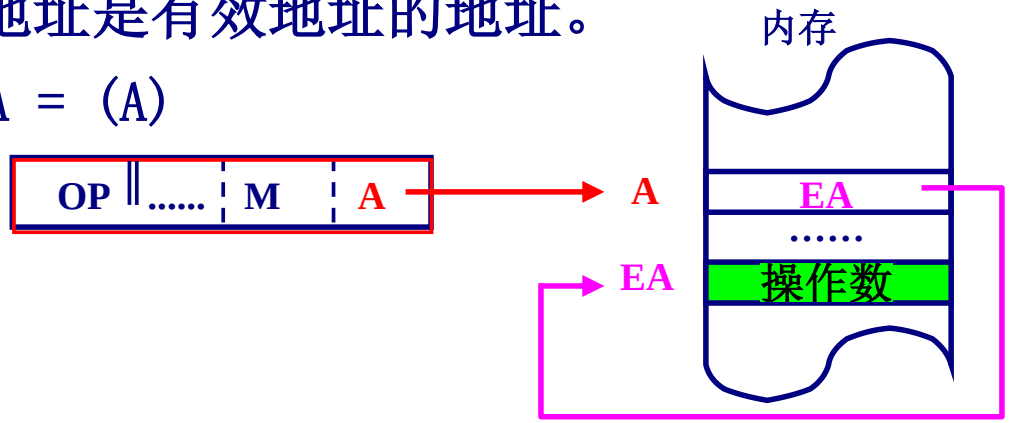
适合于访问固定主存单元或外部设备接口中的寄存器；
地址码字段较长。例如，64K 寻址空间需 16 位。



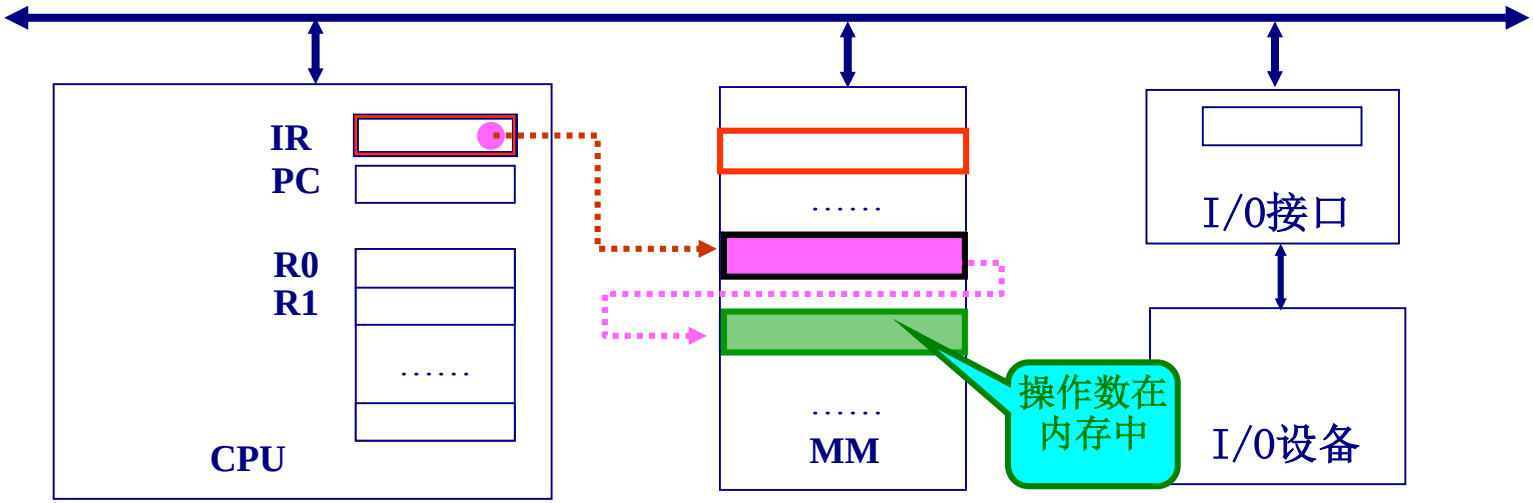
(4) 间接寻址(Indirect Addressing)

形式地址是有效地址的地址。

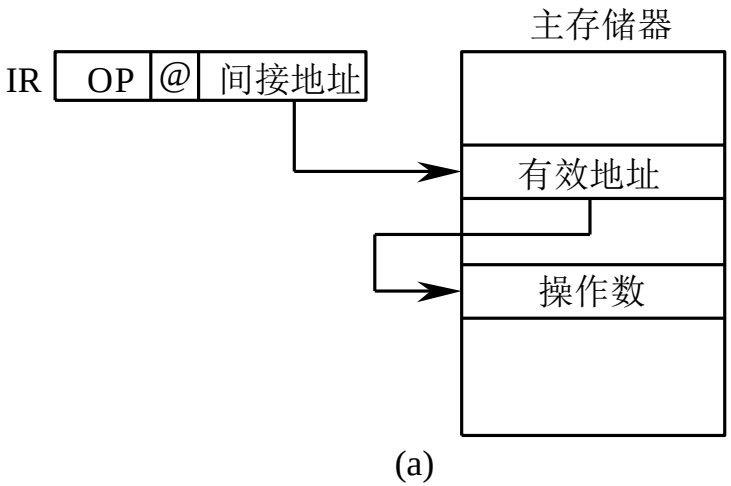
EA = (A)



需访问两次主存才能取出操作数。



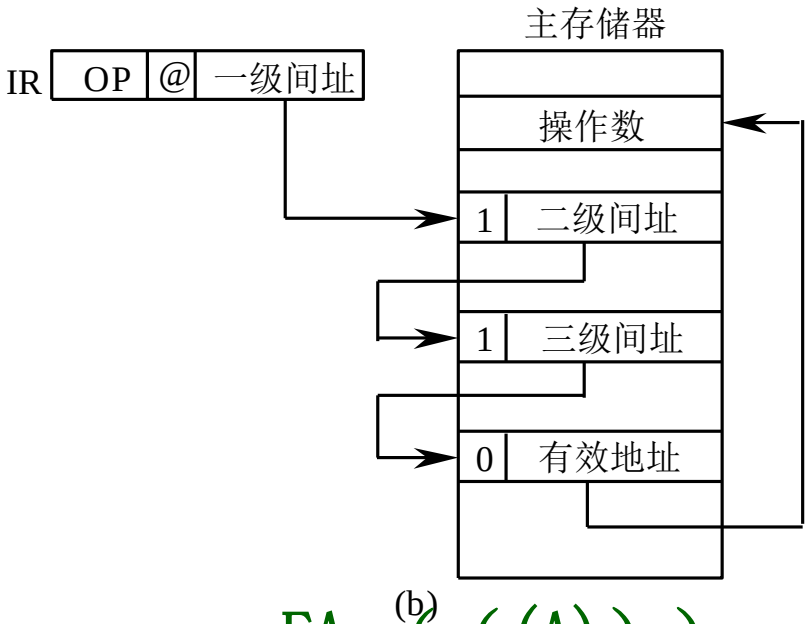
一级间接寻址



EA=(A)

S=((A))

三级间接寻址



EA=(((A)))

S=((((A))))

特点：

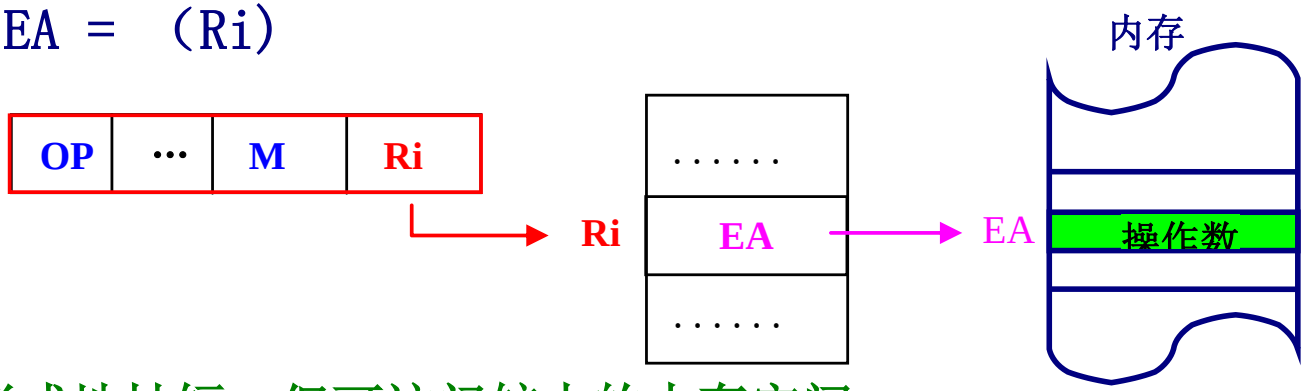
扩大了寻址范围，可用指令的短地址访问大容量主存空间；

访问速度较慢。

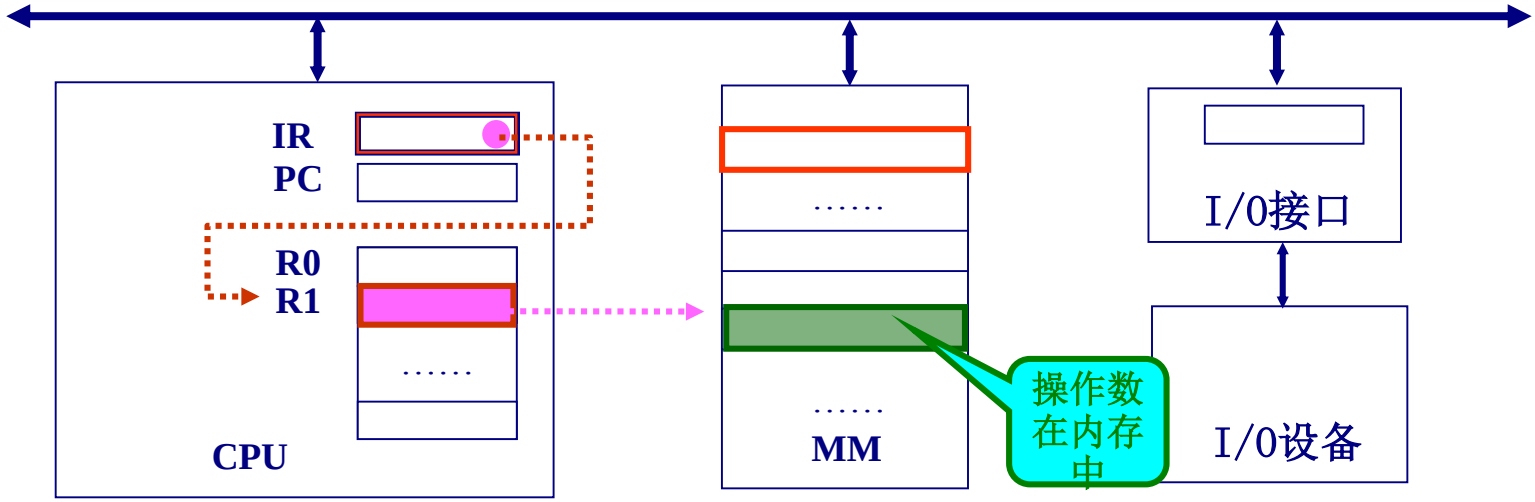
(5)寄存器间接寻址(Register Indirect Addressing)

寄存器中存放操作数的有效地址。

EA = (Ri)



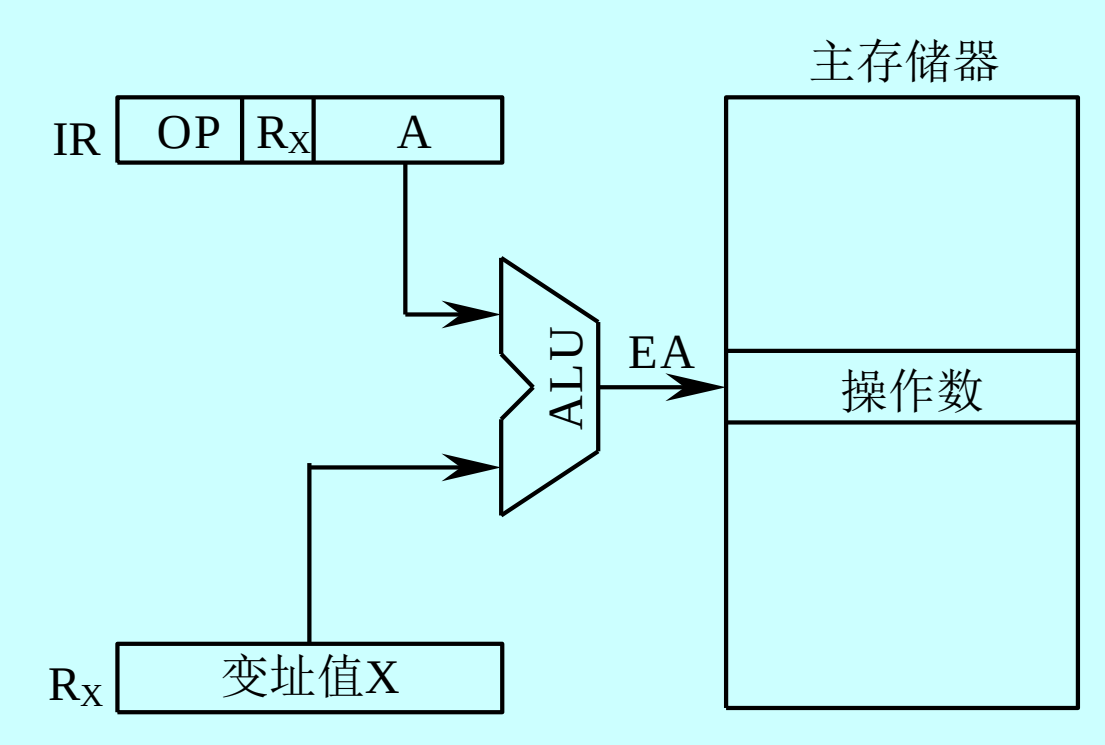
形式地址短，但可访问较大的内存空间。



(6)变址寻址(Indexed Addressing)

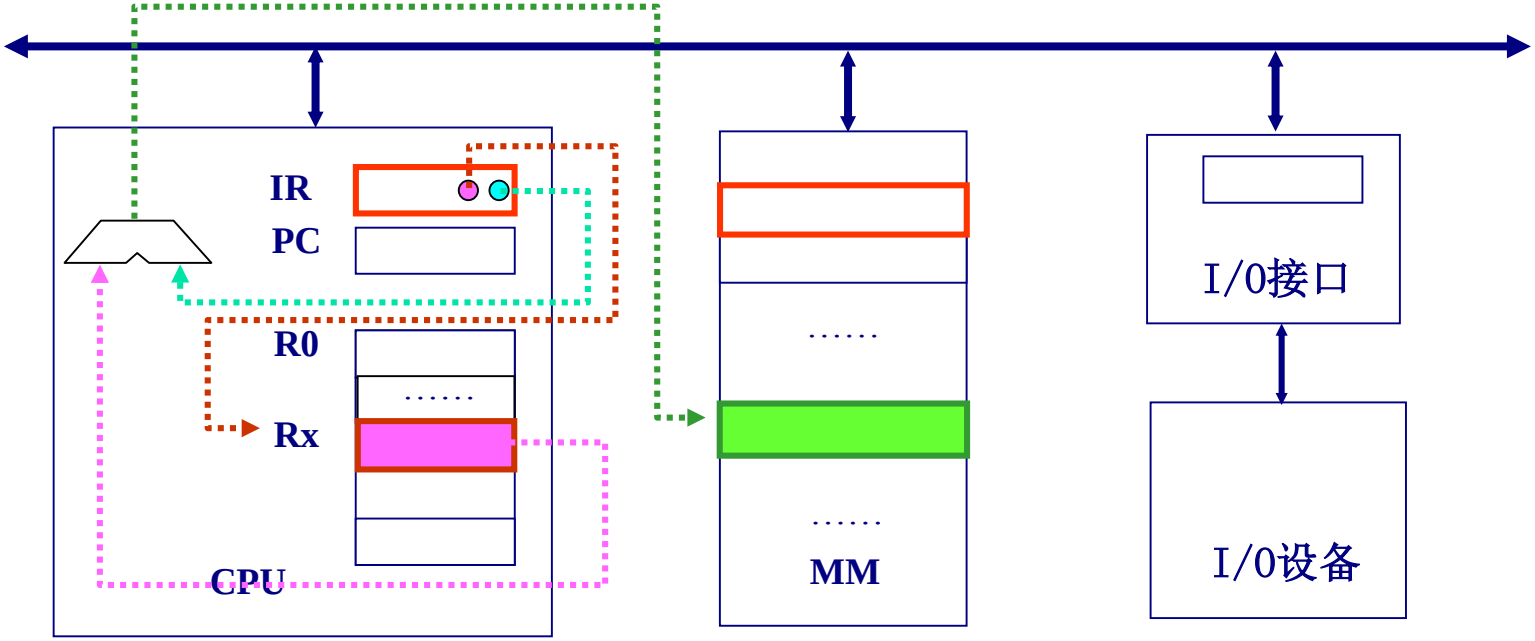
有效地址 = 变址寄存器的内容 + 形式地址(位移量)。

EA = (Rx) + A



适用于对数据块操作

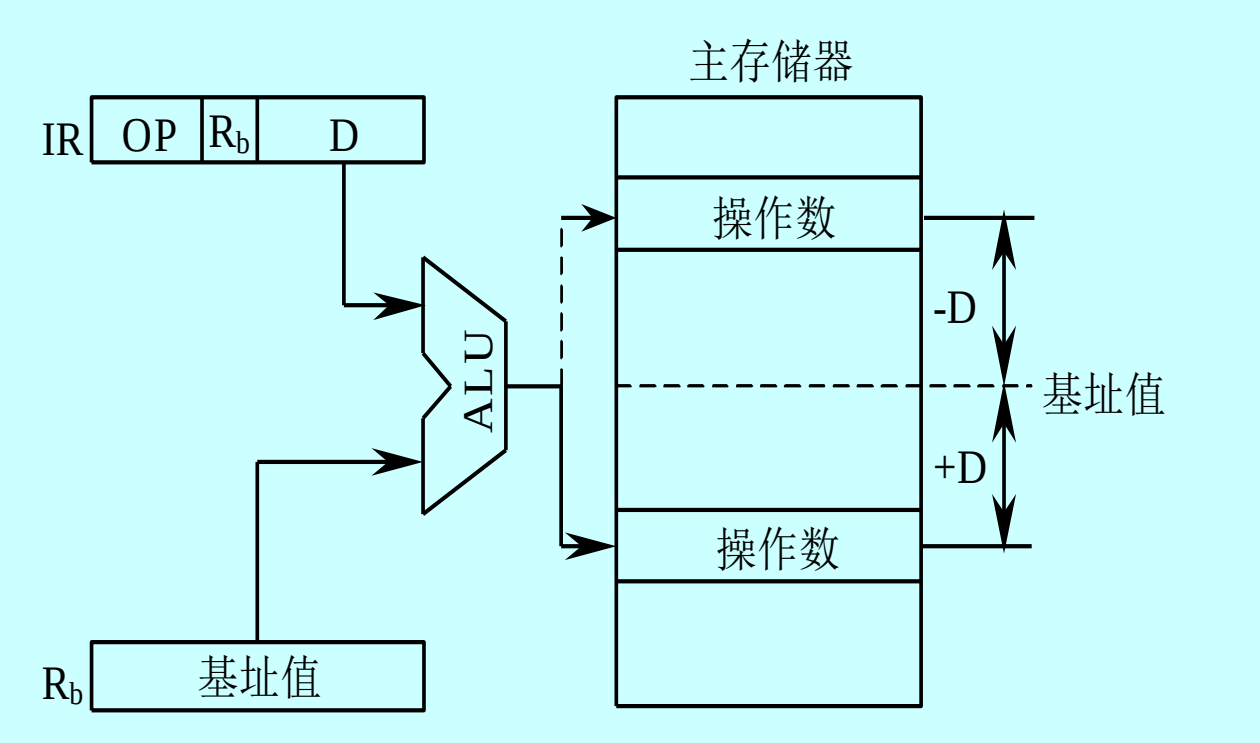
EA = (Rx) + A



(7)基址寻址(Based Addressing)

有效地址 = 基址寄存器的内容+形式地址(位移量)。

EA = (Rb) + D

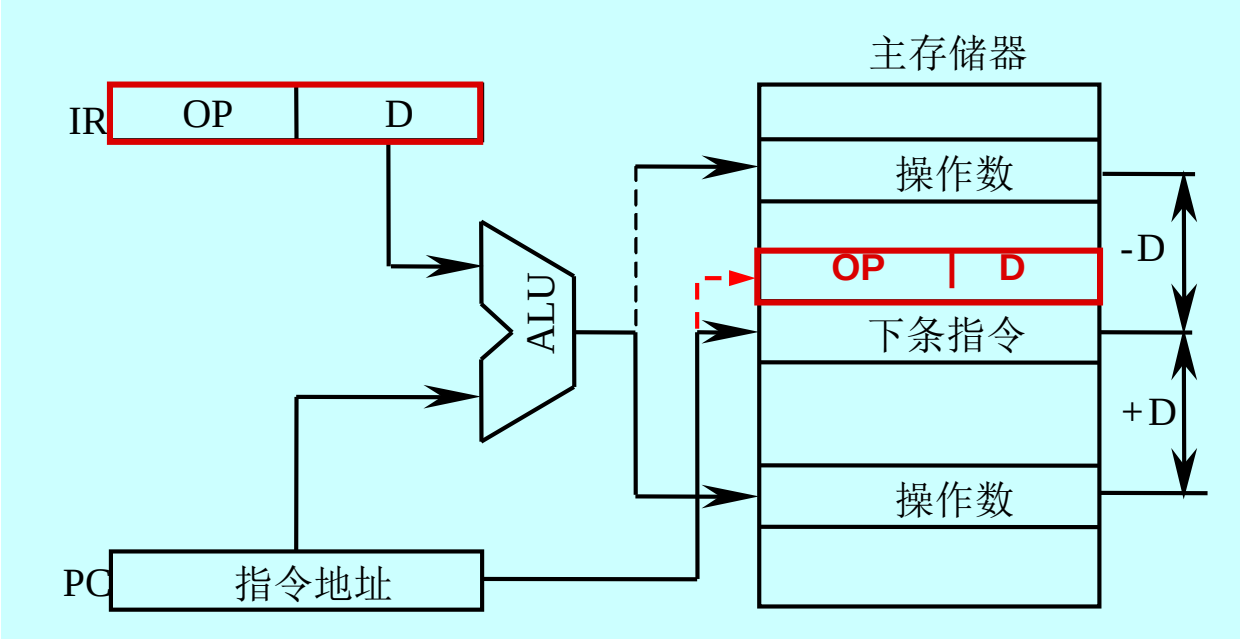


◆ 变址寻址与基址寻址的区别：

变：通常由变址寄存器提供修改量，
形式地址为基准地址，
面向用户(例如向量运算)；

基：基址寄存器提供基准地址，
形式地址为位移量，
面向操作系统(例如程序的动态重定位）。

(8)相对寻址



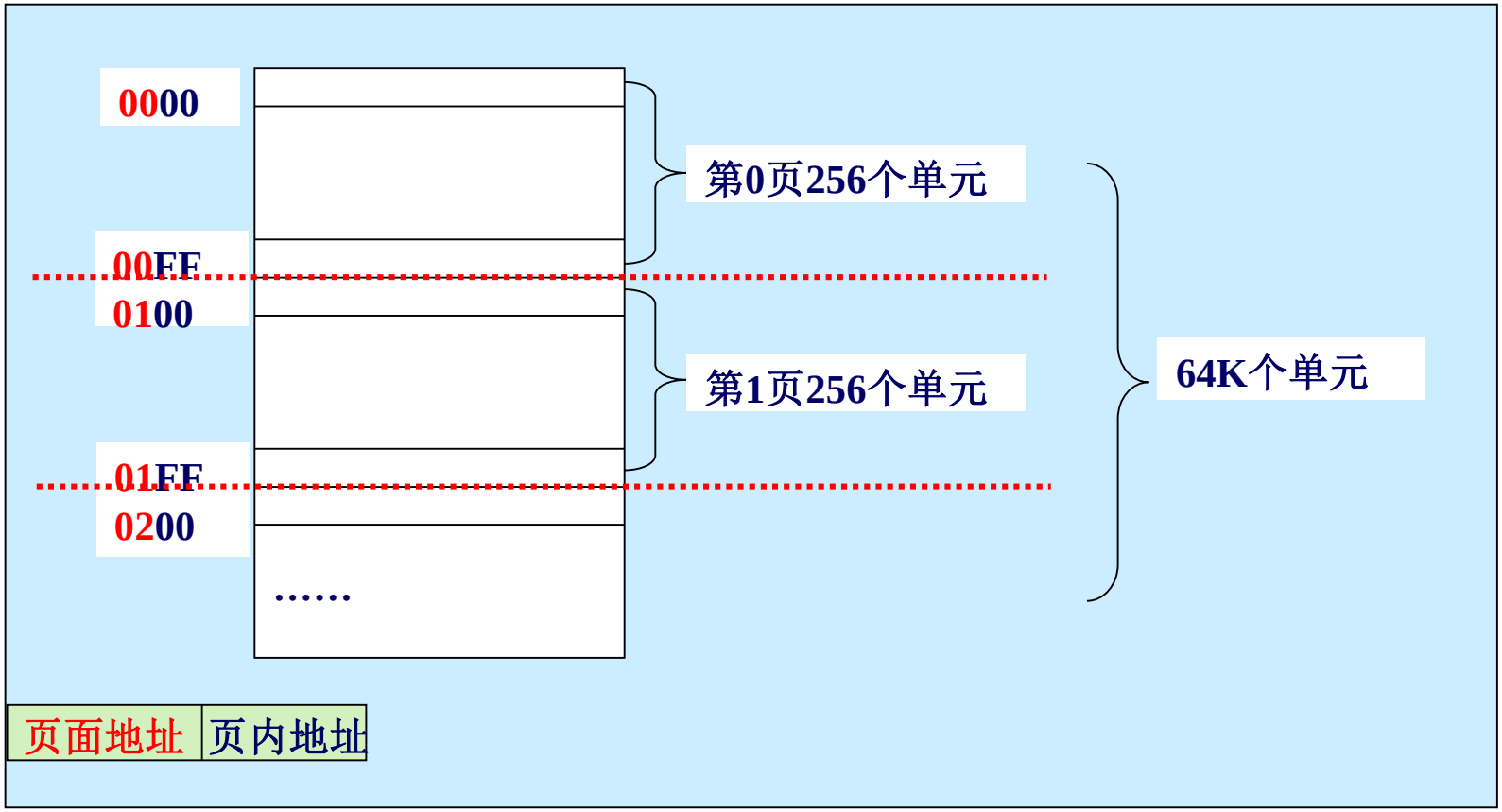
EA=(PC)+D

位移量D指出的是操作数和现行指令之间的相对位置。

(9)页面寻址

将主存空间分页，解决地址码位数与主存访问范围的矛盾。

例：某机主存为64K，将其划分为256页，则页面地址从00H～FFH；
每页有64K/256=256单元，页内单元序号从00H～FFH。



(9)页面寻址

页面寻址又可以分成三种不同的方式：

1）基页寻址

EA= 0∥A　操作数S在零页面中。实际上就是直接寻址。

2）当前页寻址

EA=$(PC)_H$∥A　操作数S与指令本身处于同一页面中。

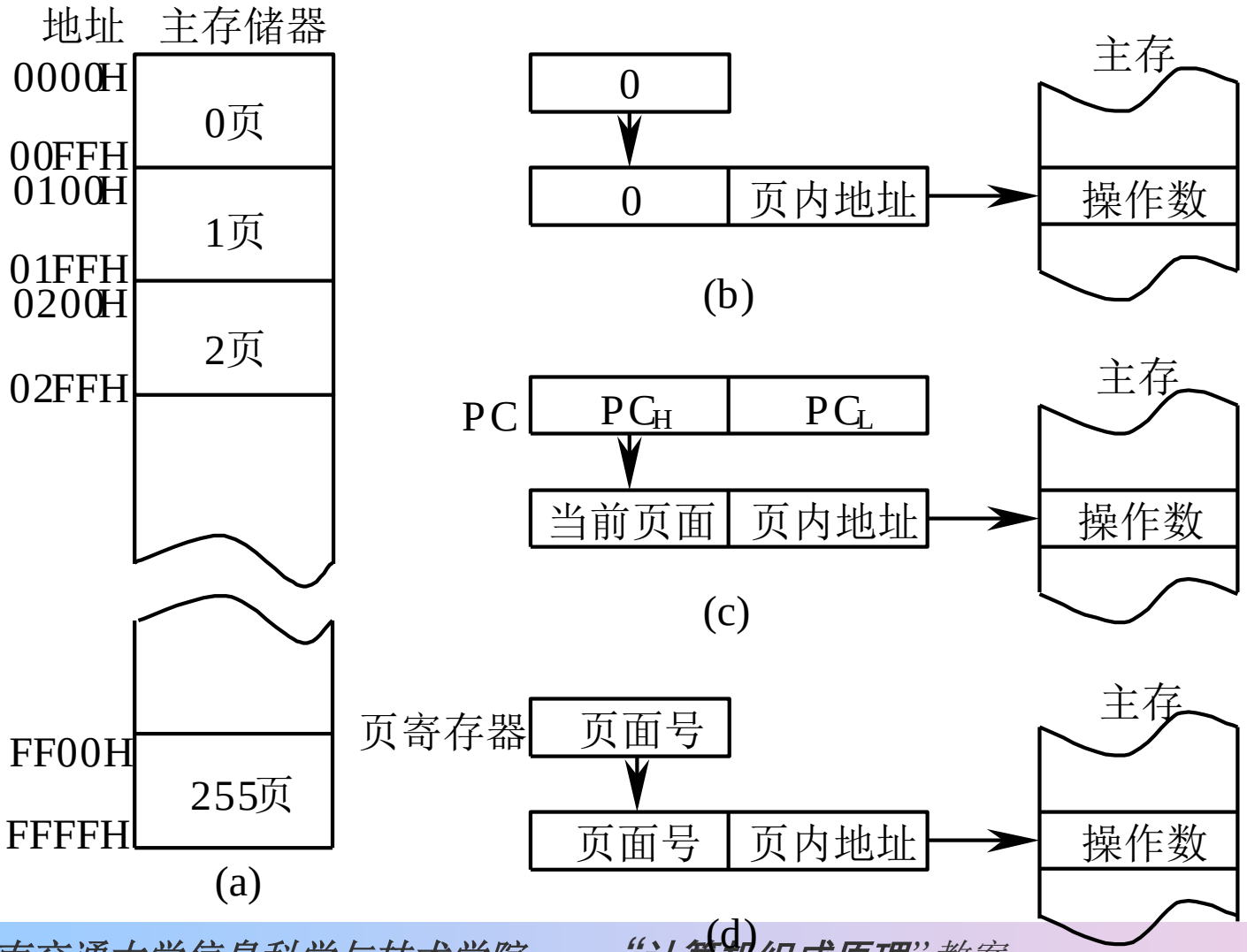
3）页寄存器寻址

页面地址取自页寄存器

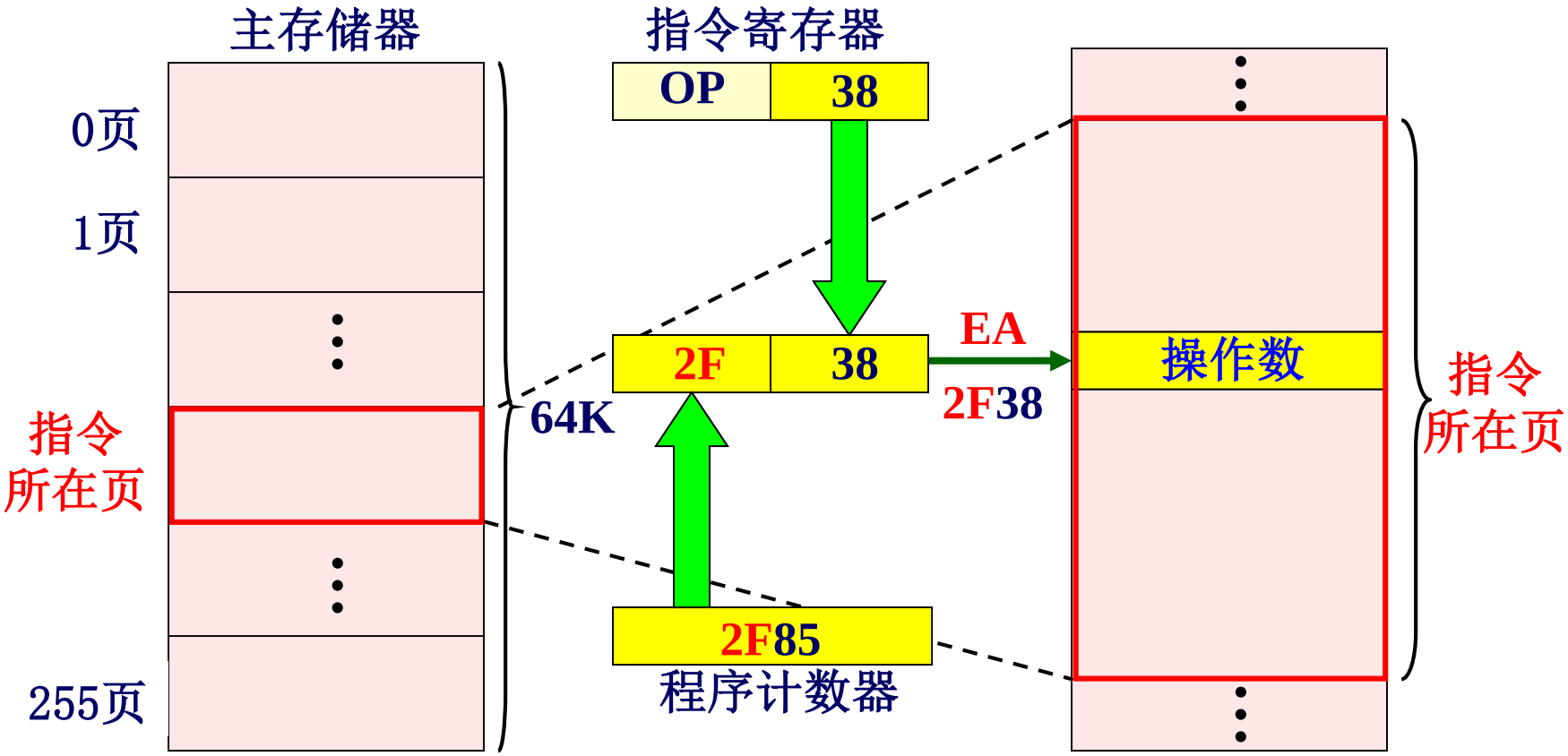
与形式地址相拼接形成有效地址。

(9)页面寻址

页面寻址又可以分成三种不同的方式：



以当前页寻址为例：


主存储器
指令寄存器
OP
38
0页
1页
指令
所在页
255页
64K
2F
38
EA
2F38
操作数
指令
所在页
2F85
程序计数器

(10)其它寻址

位寻址：能寻址到位，一般用于专门的位操作指令。

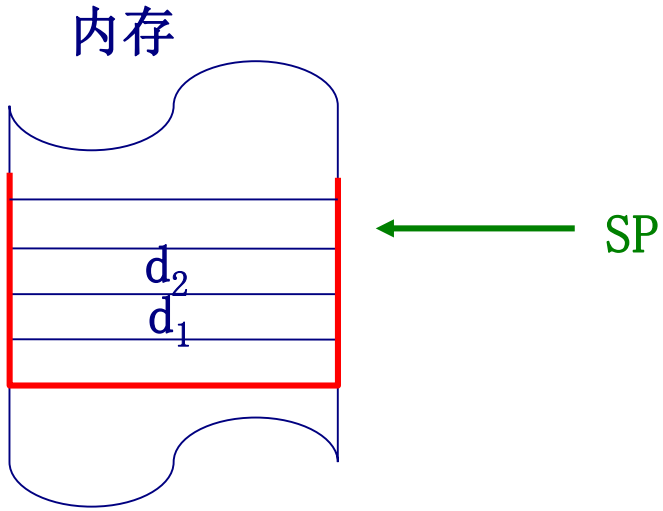
块寻址：对连续的数据块进行寻址。

一般要指明块首址和块长(或)末址。

堆栈寻址：由堆栈指针SP隐含指定,不需指令给出地址码。

存储器堆栈：内存中一个连续的存储区，

按后进先出方式存取。



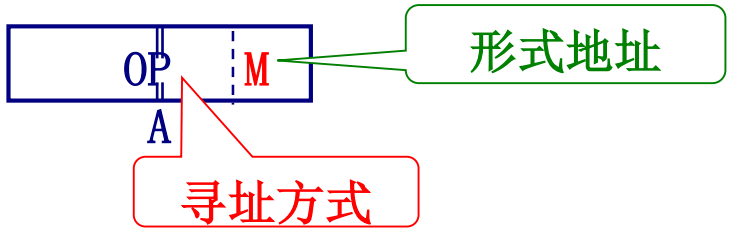
SP为栈顶指针

基本操作：1）压入

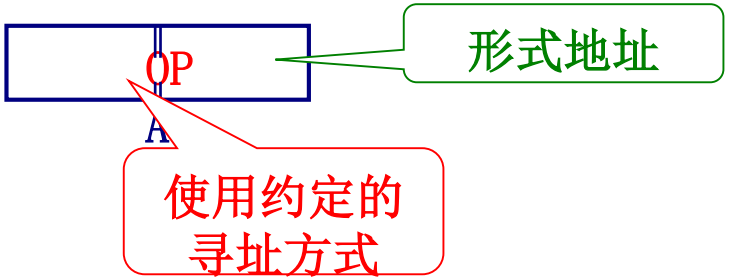
2）弹出

寻址方式的表示方式

1）显式：在指令中设置专门的寻址方式字段。



2）隐式：由指令的操作码隐含约定址方式。



一条指令中的各地址码可采用不同的寻址方式。

例如：MOV AX,[BX]

寄存器寻址

寄存器间接寻址

思考题：　P85　1，2，5，6，7，8，16

习题：P85　3，4，10，12

思考题：　P75　1，2，5，6，7，8，16

习题：P75　3，4，10，12

3.3.4 变型或组合寻址方式

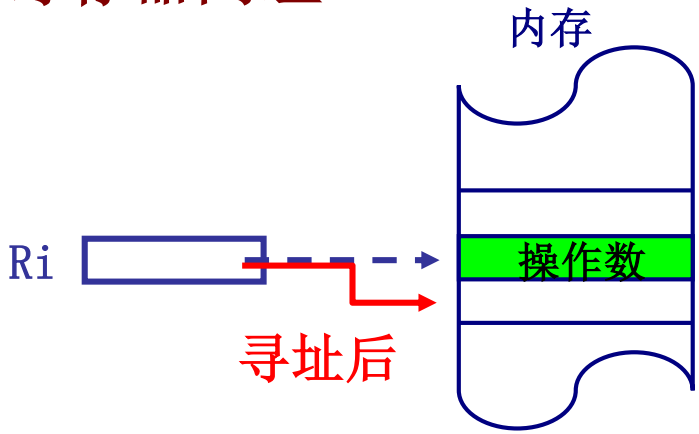
1. 自增型寄存器间址和自减型寄存器间址

(1) 自增寻址 (Ri)+

EA=(Ri)

Ri←(Ri)+d

先确定EA后递增Ri.

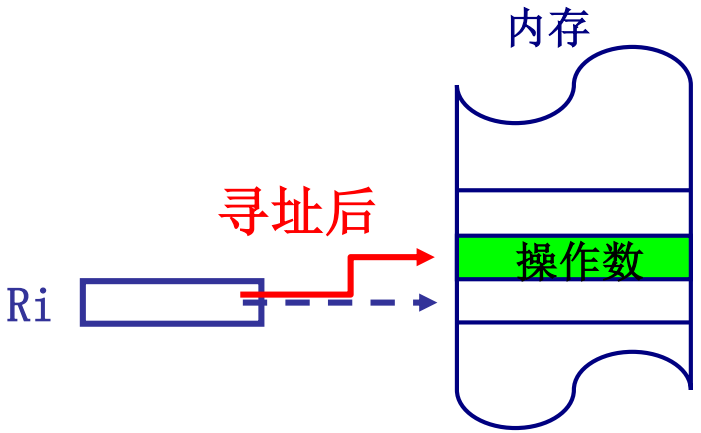


(2) 自减寻址 -(Ri)

Ri←(Ri)-d

EA=(Ri)

先递减Ri后确定EA.



2. 扩展变址方式

(1) 先变址后间址（前变址方式）

EA=(A+(Rx))，

操作数 S=((A+(Rx)))。

(2) 先间址后变址（后变址方式）

EA=(A)+(Rx)，

操作数 S=((A)+(Rx))。

3. 基址变址寻址

$$EA = (Rb) + (Rx) + D$$

其中：Rb为基址寄存器，Rx为变址寄存器，D为位移量。

位移量D在指令一旦确定后不能再修改；
基址和变址寄存器中的内容可以改变。

常见寻址方式小结

寻址方式		通俗解释
(1)	立即寻址	操作数在指令(的地址字段)中。
(2)	寄存器寻址	操作数在 CPU 的某一寄存器中。
(3)	寄存器间址	操作数在内存，其地址在 CPU 的某一寄存器中。
(4)	直接寻址	操作数在内存，其地址由指令直接给出。
(5)	间接寻址	操作数在内存，指令中给出操作数地址的地址（操作数的地址也在内存）。
(6)	变址、基址、相对寻址	操作数在内存，其内存地址为 Rx 或 Rb 或 PC 的值 + 指令中给的位移量。

本章重点：

1. 理解指令的基本格式和基本操作种类，理解扩展操作码方法；
2. 掌握基本数据寻址方式和有效地址的确定方法，搞清楚每一种基本寻址方式的特点；
3. 存储器堆栈的概念及堆栈的进、出栈操作

§3.4 堆栈与堆栈操作

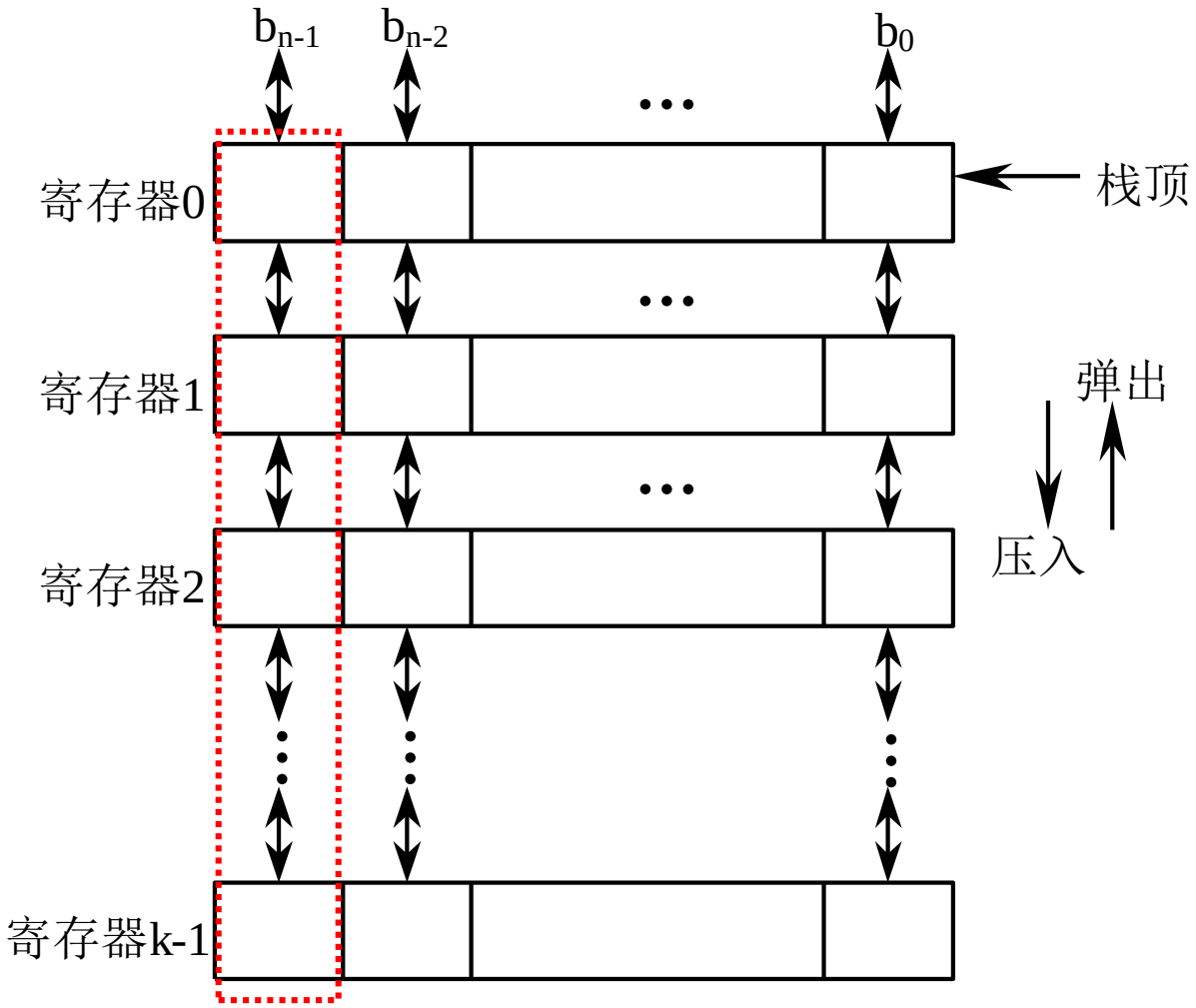
堆栈：“后进先出”(LIFO)的存储区。

3.4.1 堆栈结构

- 寄存器堆栈(硬堆栈)
- 存储器堆栈(软堆栈)

1. 寄存器堆栈(硬堆栈)

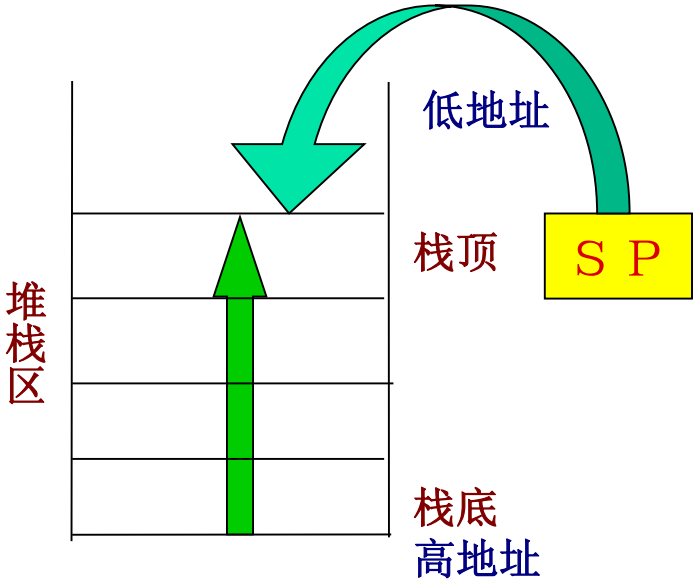
可由移位寄存器组构成。（从图中纵向看）



2. 存储器堆栈(软堆栈)

从主存中划出一段区域来作堆栈，栈底固定，栈顶浮动。

由堆栈指针寄存器SP指示当前栈顶的位置。



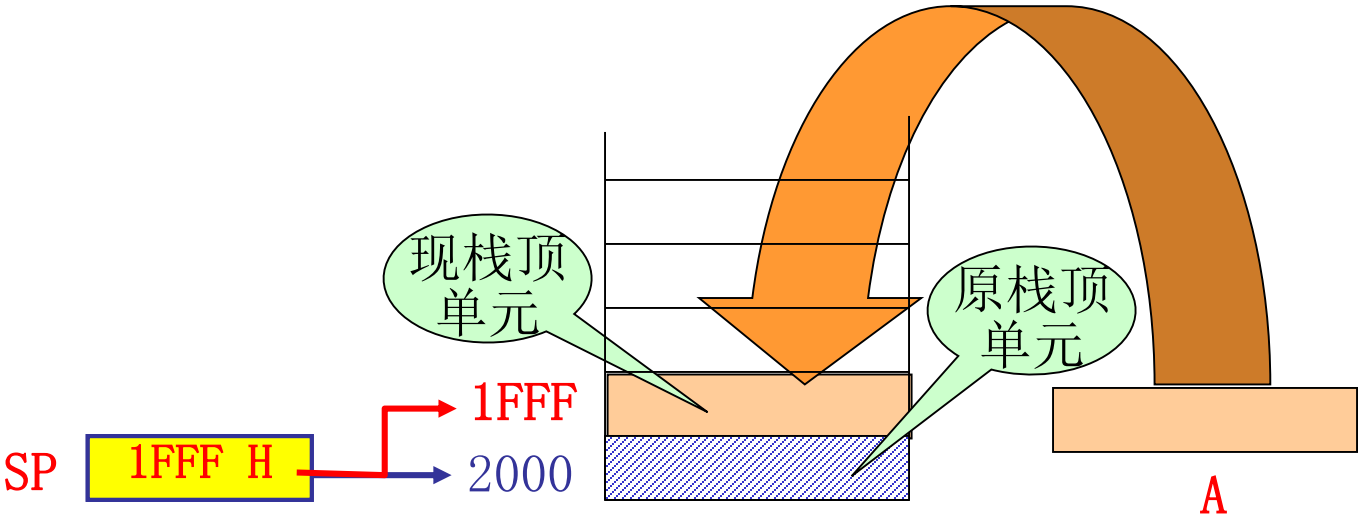
自底向上生成堆栈，栈底地址大于栈顶地址，通常栈指针始终指向栈顶的满单元。

3.4.2 堆栈操作

进栈操作（压入）：

（SP）－1→SP　修改栈指针

（A）→（SP）　将A中的数据压入堆栈

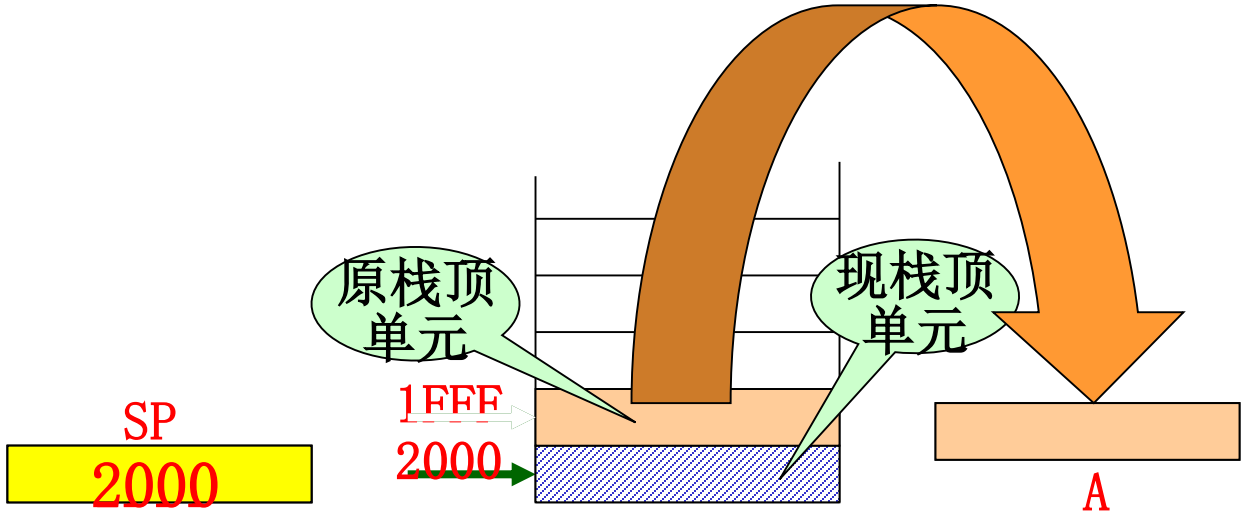


3.4.2 堆栈操作

出栈操作（弹出）：

（（SP））　→A　将栈顶内容弹出，送入A中

（SP）+ 1　→SP　修改栈指针



堆栈的应用：

在一般计算机中，堆栈主要用来暂存中断断点、子程序调用时的返回地址、状态标志及现场信息等，也可用于子程序调用时参数的传递等。

§3.5 指令系统实例（侧重于8086/8088寻址方式）

3.5.0 与8086/8088相关的预备知识

1. 8086/8088的结构

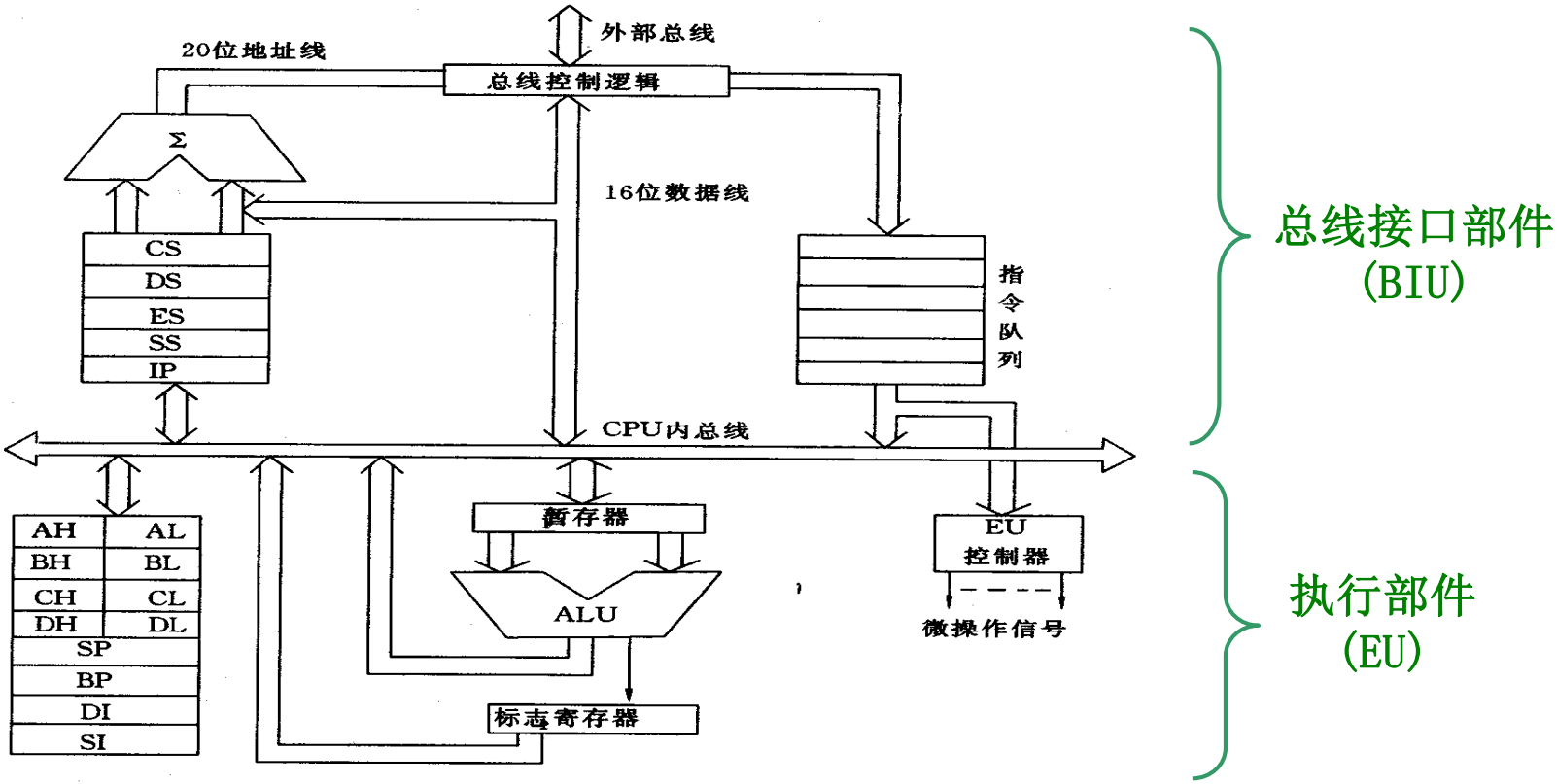


图 2-12　8086 内部结构框图

3.5.0 与8086/8088相关的预备知识

2. 8086/8088存储器的分段管理

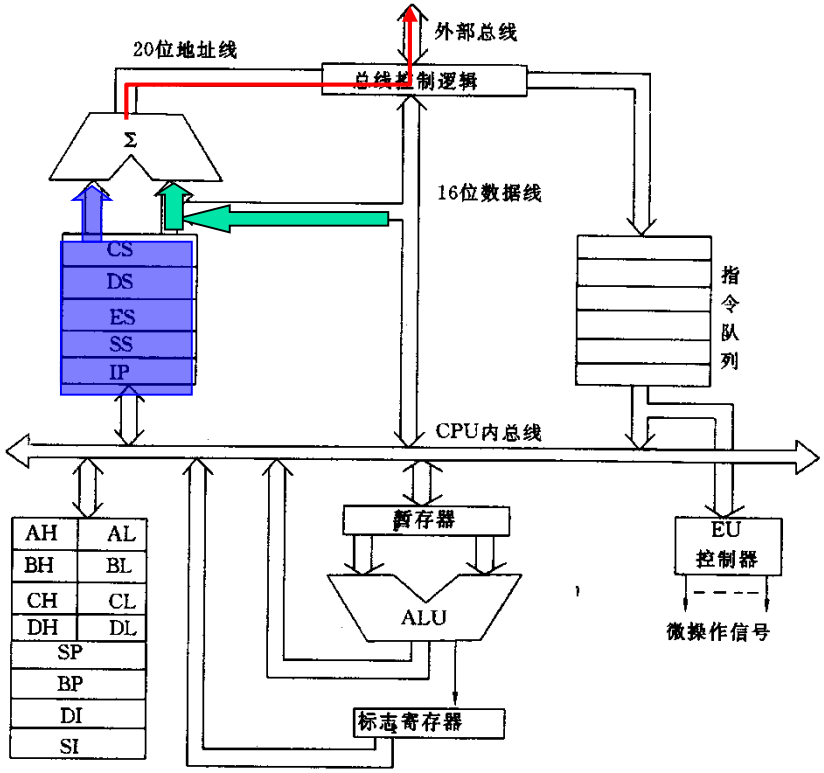


图 2-12　8086 内部结构框图

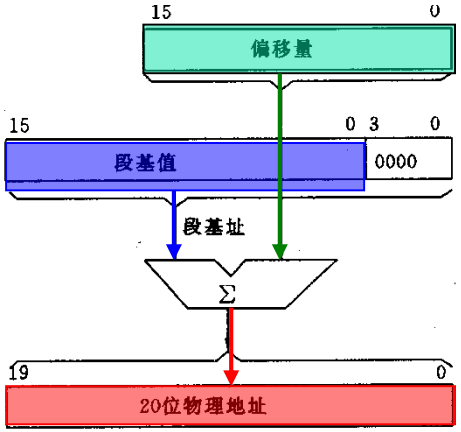


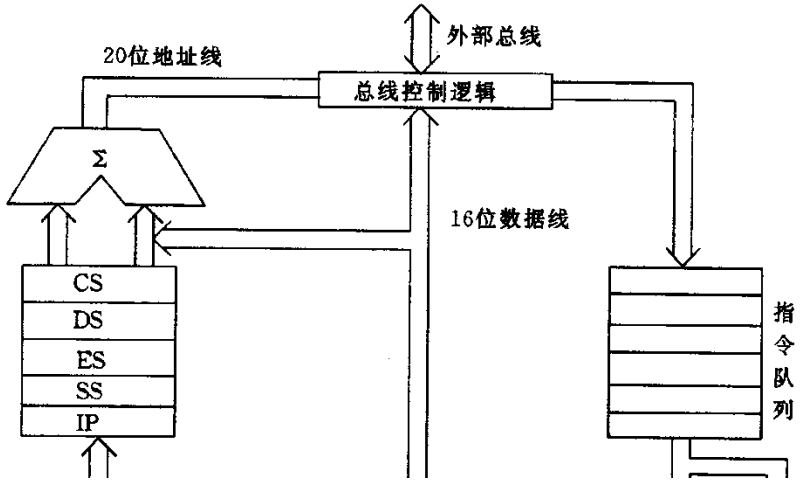
图 2-13　8086/8088 主存物理地址的形成

分段管理的目的是使指令可以访问较大的内存空间。

3.5.0 与8086/8088相关的预备知识

3. 段寄存器

CS：代码段寄存器
DS：数据段寄存器
SS：堆栈段寄存器
ES：附加段寄存器



通常将需执行的程序各部分(指令代码、数据、堆栈等)分别放在指定的某个段中。当CPU访问某个主存单元(如取指令或存取操作 数)时，就必须指明（或默认使用）哪个段寄存器提供该单元的段基值，并给出该单元在这个段内的偏移量。

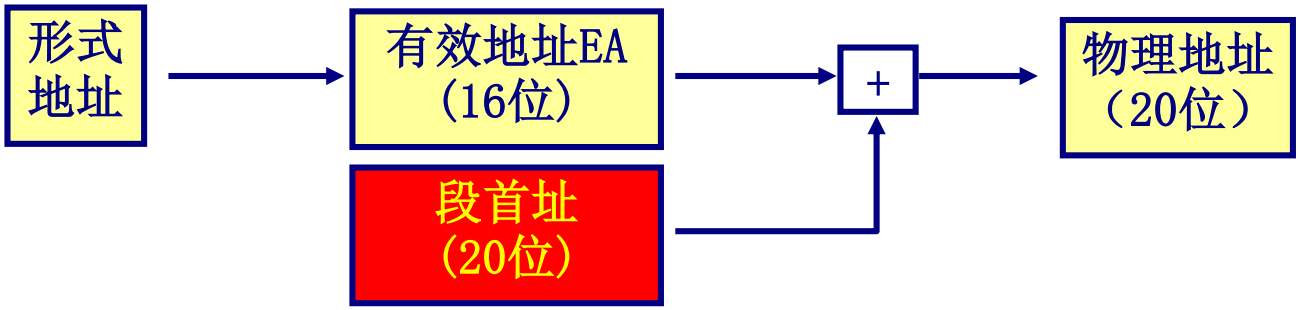
3.5.1 8086/8088指令系统特征

二地址指令系统。（一条指令里最多有两个地址）

操作码：定长操作码，变长指令码。

地址码：寄存器、立即、寄存器间址、基址、变址等寻址。

对于操作数在内存的寻址方式，有效地址EA≠ 物理地址！



对于二地址指令：其中一个寻址方式可选，

另一个只能是寄存器寻址。

3.5.2 寻址方式

以二地址指令为例，指令格式为

例如：　ADD　BX，5

基本寻址方式	汇编符号例子	寻找操作数(的地址)的过程
寄存器寻址	ADD　BX,5	EA = BX，即数据在寄存器中
寄存器间址	ADD [BX],AX	EA =（BX）
立即寻址	ADD　BX,5	操作数在指令中
直接寻址	ADD [100],AX	EA = 100
间接寻址	------	
变(基)址寻址	ADD [BX+20],AX	EA = (BX+20)
自相对	LOOP L1	EA = (IP)+ 位移量
……		

例：
```
MOV   AL, AH      ;   AL ← (AH)              8位
SUB   AX, BX      ;   AX ← (AX) - (BX)       16位
INC   CX          ;   CX ← (CX) + 1          16位
```

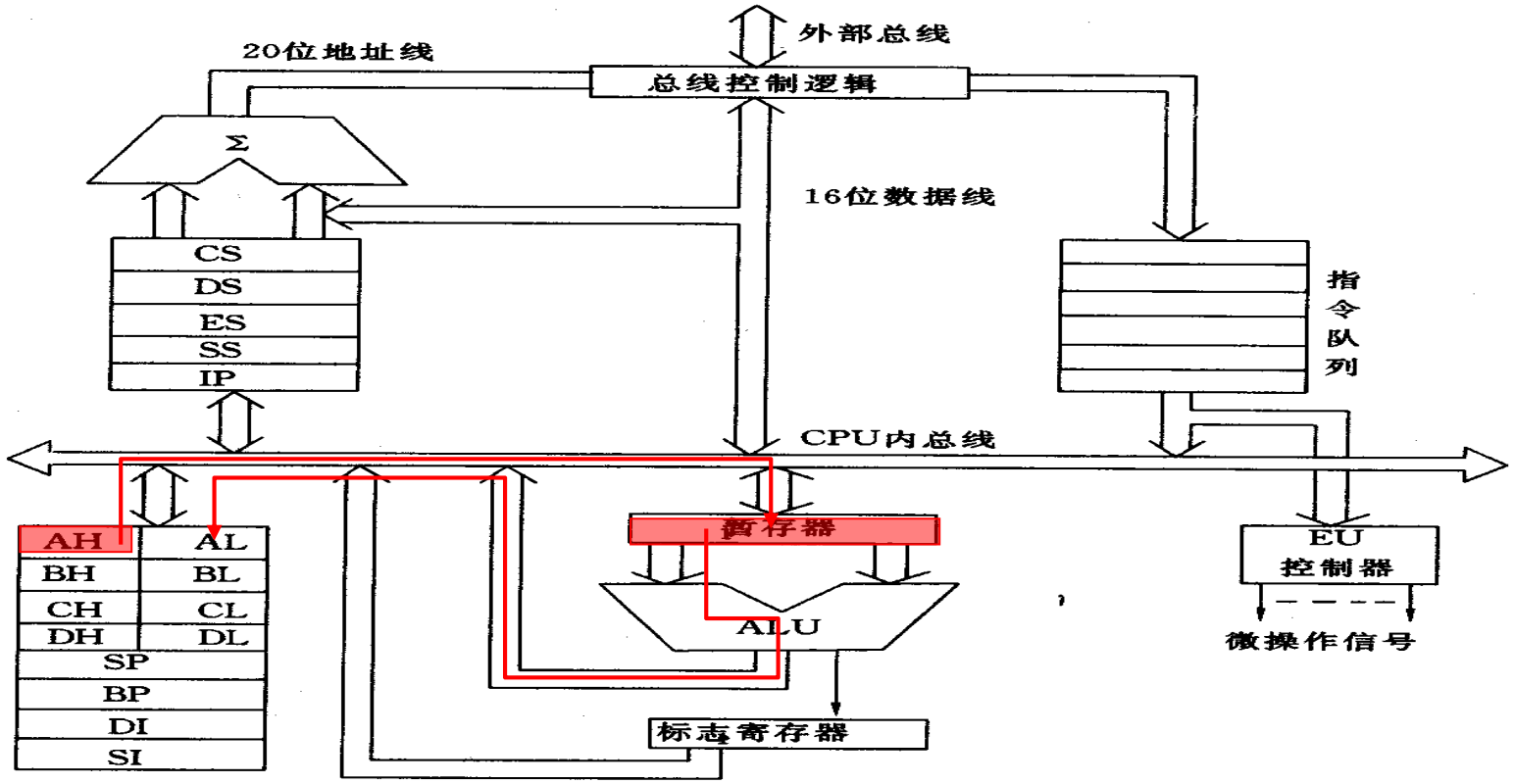
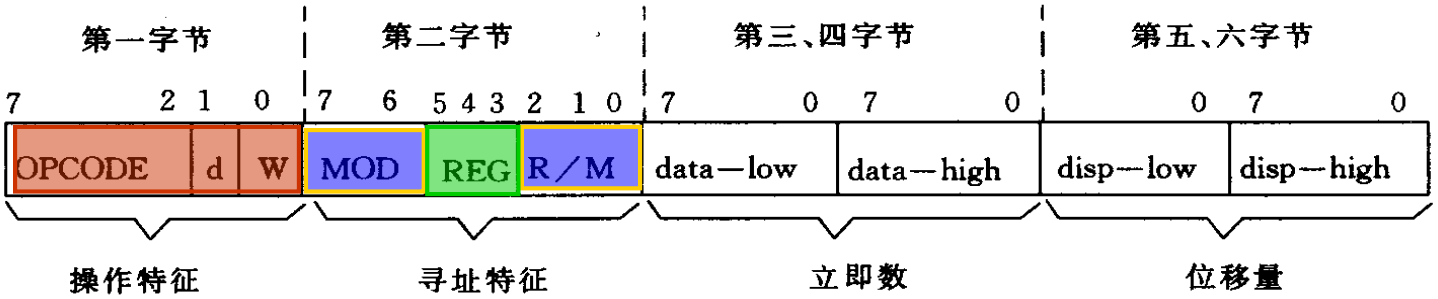


图 2-12　8086 内部结构框图

3.5.3 双操作数指令代码格式



OPCODE：操作码

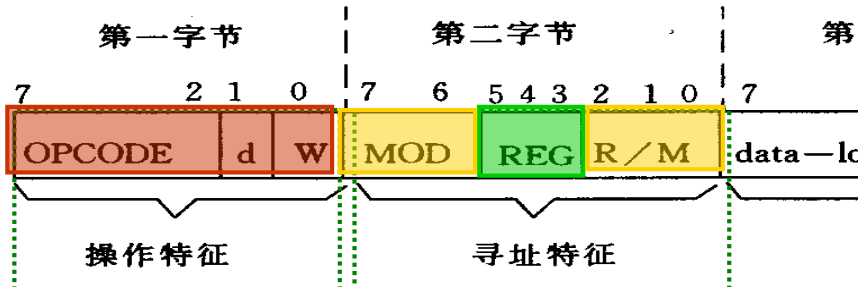
d ：方向。 1--目操取决于REG，源操取决于MOD和R/M；
0--目操取决于MOD和R/M，源操取决于REG

W ： 字/字节。0--字节操作； 1--字操作

REG ： 寄存器号 W=0 W=1

000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
……		

MOD,R/M：可指定多种寻址方式。例如MOD=11时为REG寻址（依据R/M，W）。



例：　MOV　AL,AH ;　100010 1 0　11 000 100

　　SUB　AX,BX ;　001010 1 1　11 000 011

附：寻址方式的应用例子（仿8086的模型机，没有段寄存器）

把2000—2099内存区域中的内容复制到3000—3099 内存区域中.

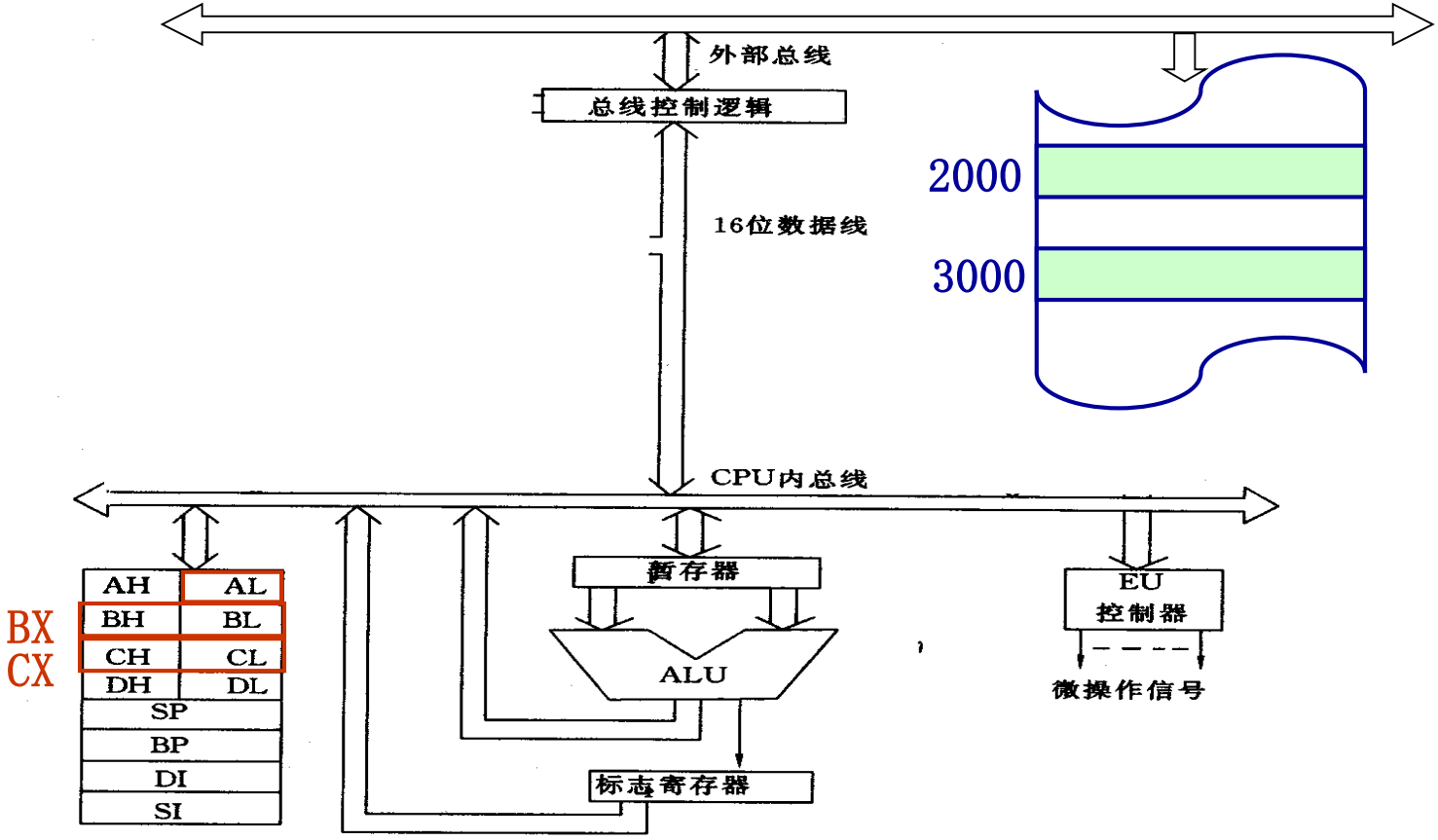


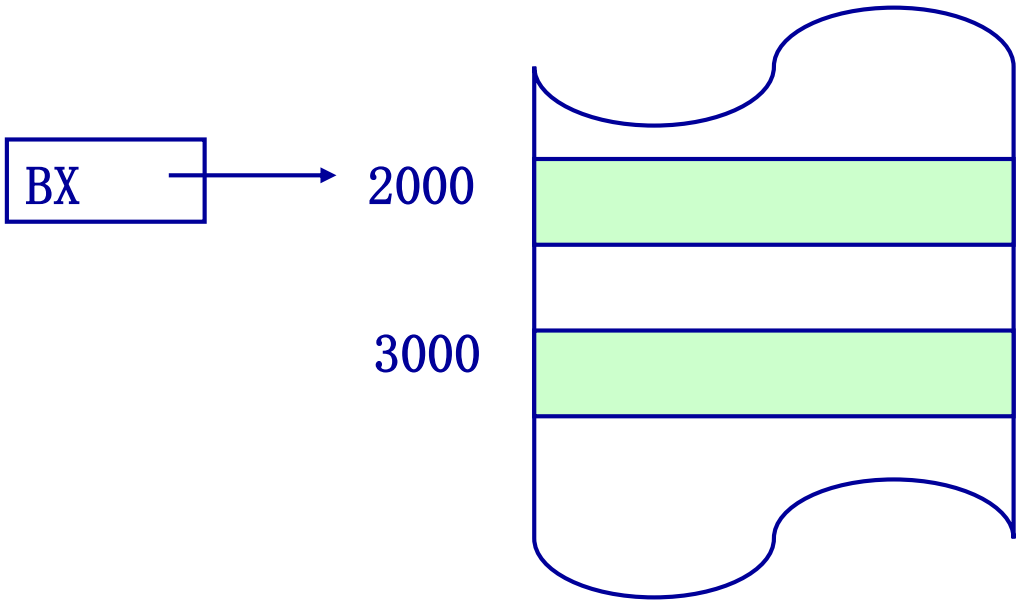
图 2-12　8086 内部结构框图

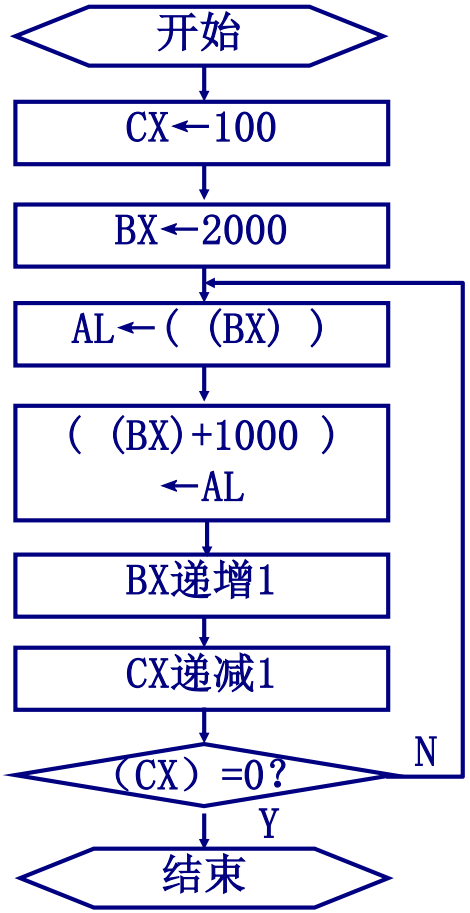
附：寻址方式的应用例子（仿8086的模型机，没有段寄存器）

把2000—2099内存区域中的内容复制到3000—3099 内存区域中.

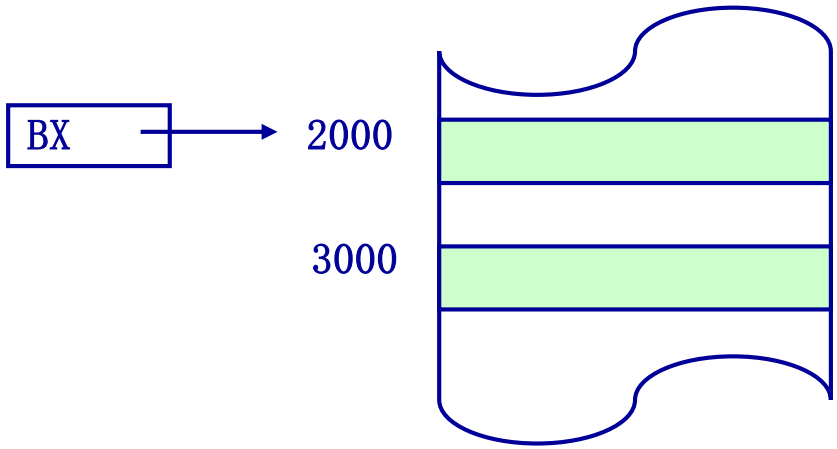
[解] CX作为传输数据的循环计数器（倒计数）；

BX作为被复制内存单元的指针（地址）。






```
    MOV CX，100   ；寄存器寻址、立即数寻址
    MOV BX，2000  ；寄存器寻址、立即数寻址
L1：MOV AL，[BX]  ；寄存器寻址、寄存器间址
    MOV [BX+1000],AL；变址寻址、寄存器寻址
    INC BX        ；寄存器寻址
    LOOP L1       ；隐地址、自相对寻址
                  ；    CX←（CX）-1，
                  ；    然后若（CX）≠0则转移
（END）
```



§3.5 精简指令系统计算机（RISC）

3.5.1 指令系统发展的演变

1．从简单到复杂

起因：VLSI技术的发展，使硬件成本降低

从指令系统的完备性和有效性考虑

（缩小机器语言与高级语言的语义差距）

系列机向下兼容的功能要求的影响

设计思想：计算机性能越强，其指令系统应越复杂。

这就是复杂指令集计算机(CISC)产生的背景

CISC---- Complex Instruction Set Computer

CISC特点：指令多达几百种，指令复杂化

指令功能增强，寻址方式也更多样

§3.5 精简指令系统计算机（RISC）

3.5.1 指令系统发展的演变

2. RISC的产生

RISC（Reduced Instruction Set Computer）

简化指令系统计算机1975年由IBM公司的John Cocke提出

起因：CISC研制周期长，增加设计失误的可能性；

复杂指令须复杂的操作，有时反而影响速度；

CISC指令使用频度的“20%----80%率”；

不利于指令执行的流水处理等。

2．RISC的产生

分析：程序的执行速度受三个因素的影响：

1）程序中的指令数N

2）每条指令执行所需的周期数C

3）每个周期的时间P

一段程序的执行时间T为：T＝N×C×P

CISC技术中：N↓，但同时C↑↑或P↑

RISC技术中：N↑，且同时C↓↓和P↓

P相同时：

在CISC中，C＝4～10、执行指令条数为N；

在RISC中，C＝1.3～1.7、

执行指令条数约为N+N×(20%～40%)；

结论：RISC的性能为CISC的2～5倍。

3.5.2 RISC的特点和优势

(1) 指令总数较少；

(2) 基本寻址方式种类少；

(3) 指令格式少，而且长度一致；

(4) 除取数和存数指令（Load/Store）外，大部分指令在单周期内完成；

(5) 只有取数和存数指令能够访问存储器，其余指令的操作只限于在寄存器之间进行；

(6) CPU中通用寄存器的数目应相当多；

(7) 为提高指令执行速度，绝大多数采用硬联线控制实现，不用或少用微程序控制实现；

(8) 采用优化的编译技术，力求以简单的方式支持高级语言。

3.5.3 RISC基本技术

1. RISC寄存器管理技术
2. 流水线技术
3. 延时转移技术

一、填空题

1. 根据操作数所在位置，指出其寻址方式：操作数在寄存器中，称为_____寻址方式；操作数地址在寄存器中，称为_____寻址方式；操作数在指令中，称为_____寻址方式；操作数地址在指令中，称为_____寻址方式。操作数的地址，为某一个寄存器中的内容与位移之和，则可以是_____、_____或_____寻址方式。

2. 在直接寻址方式中，操作数存放在_____中。

3. 存储器间接寻址方式指令中，给出的是_____所在的存储器的地址。

4. 计算机通常使用_____来指示指令的地址。

二、简答题

1. 对各种存储设备进行编址的目的是什么？

2. 何谓指令寻址？

3. 指令中地址码的位数与直接访问的主存容量和最小寻址单位有什么关系？

4. 何谓有效地址？

一、填空题答案

1. 寄存器，寄存器间接，立即，直接，基址，变址，相对

2.内存

3.操作数地址

4.程序计数器

二、简答题

1. 对各种存储设备进行编址的目的是什么？

答：存储器是存放指令和操作数的，编址的目的是为了能通过PC的内容找到指令存放的位置，通过指令的地址码字段指出操作数的来源和去向。

2. 何谓指令寻址？

答：指令寻址即寻找下一条将要执行的指令地址。通常采用顺序寻址(顺序执行)或跳跃寻址(转移)。

3. 指令中地址码的位数与直接访问的主存容量和最小寻址单位有什么关系？

答：主存容量越大，所需的地址码位数就越长。对于相同容量来说，最小寻址单位越小，地址码的位数就越长。

4. 何谓有效地址？

答：形式地址——经过一定的计算而得到的能直接访问操作数的地址。