王道考研——组成原理

WWW.CSKAOYAN.COM

第四章 指令系统

本章总览

本节内容

指令系统

指令格式

本节总览

机器指令

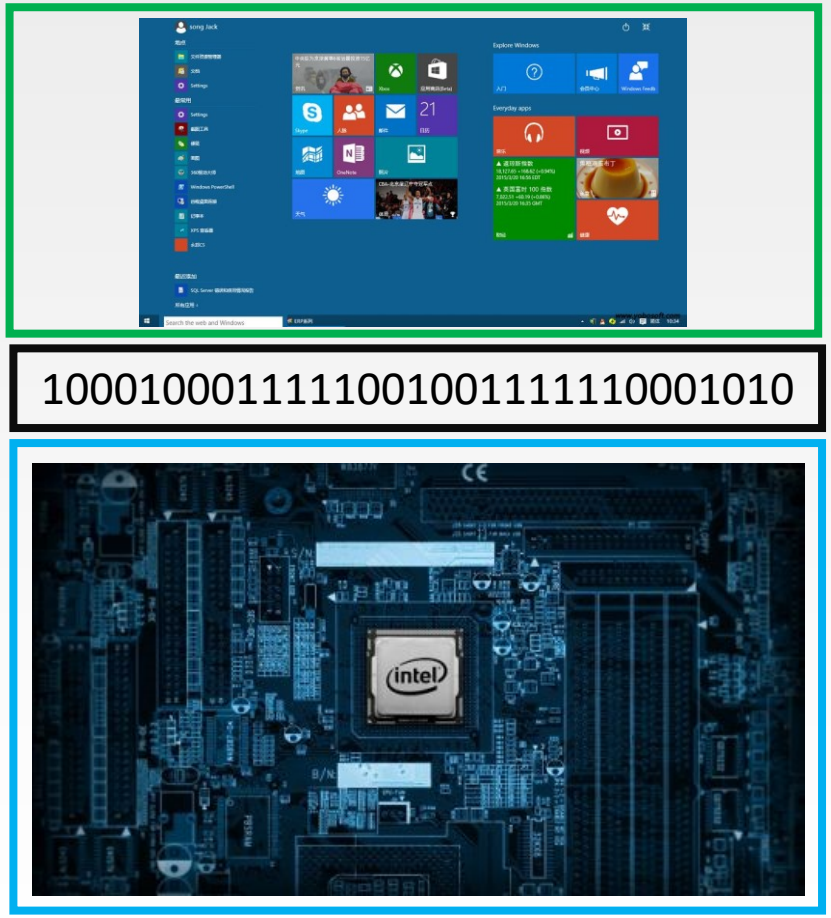
机器指令：

是指示计算机执行某种操作的命令，是计算机运行的最小功能单位。

一台计算机的所有指令的集合构成该机的指令系统，也称为指令集。

注：一台计算机只能执行自己指令系统中的指令，不能执行其他系统的指令。

指令系统在计算机中的地位



指令的一般格式

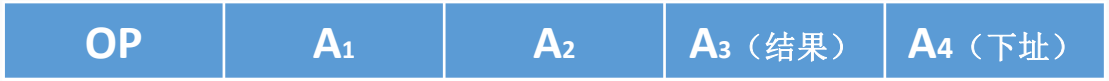
反映机器做什么操作　　对谁进行操作

(1) 长度固定

用于指令字长较长的情况 ， 如 RISC

如 IBM 370 操作码 8 位

(2) 长度可变



指令含义：$(A_1)OP(A_2)\rightarrow A_3$，$A_4$=下一条将要执行指令的地址

本节内容

指令系统

地址码的数目

指令格式-地址码

假设是+

	00000000	000001	000010	000011	000100
四地址指令	OP	A_1	A_2	A_3（结果）	A_4（下址）

指令含义：$(A_1)OP(A_2)\rightarrow A_3$，$A_4$=下一条将要执行指令的地址

设指令字长为32位，操作码占8位，4个地址码字段各占6位

设存储字长为32位，即4B

- A_i可直接表示2^6=64 个不同的位置
- 一条指令的执行(假设每个地址都是主存地址)：

  取指令 访存1次(假设指令字长=存储字长)

  取两个操作数 访存2次

  存回结果 访存1次

  共访存4次

000000	000420C4H
000001	12344321H
000010	43211234H
000011	55555555H
000100	22343234H
000101	
…	
111101	
111110	
111111	

指令格式-地址码

假设是+

00000000	111111	111110	111101	000100
OP	A_1	A_2	A_3（结果）	A_4（下址）

四地址指令

指令含义：$(A_1)OP(A_2)\rightarrow A_3$，$A_4$=下一条将要执行指令的地址

设指令字长为32位，操作码占8位，4个地址码字段各占6位

设存储字长为32位，即4B

- A_i可直接表示2^6=64 个不同的位置
- 一条指令的执行(假设每个地址都是主存地址)：

  取指令 访存1次(假设指令字长=存储字长)

  取两个操作数 访存2次

  存回结果 访存1次

  共访存4次

000000	00FFEF44H
000001	22BCE142H
000010	
000011	
000100	
000101	
…	
111101	55555555H
111110	43211234H
111111	12344321H

程序计数器PC [000000] 取出00FFEF44H后 [000001]

三地址指令

OP	A_1	A_2	A_3（结果）

若 PC 代替 A_4

指令格式-地址码

设指令字长及存储字长均为32位，操作码占8位

四地址指令

OP	A_1	A_2	A_3（结果）	A_4（下址）

指令含义：$(A_1)OP(A_2)\rightarrow A_3$，$A_4$=下一条将要执行指令的地址

4个地址码字段各占6位，指令操作数直接寻址范围为2^6=64；完成一条指令需要访存4次

三地址指令

OP	A_1	A_2	A_3（结果）

指令含义：$(A_1)OP(A_2)\rightarrow A_3$

3个地址码字段各占8位，指令操作数直接寻址范围为2^8=256；完成一条指令需要访存4次

二地址指令

OP	A_1（目的操作数）	A_2（源操作数）

指令含义：$(A_1)OP(A_2)\rightarrow A_1$

2个地址码字段各占12位，指令操作数直接寻址范围为2^{12}=4K；完成一条指令需要访存4次

一地址指令

OP	A_1

指令含义：1. $OP(A_1)\rightarrow A_1$，如加1、减1、取反、求补等　　完成一条指令需要访存3次

2. $(ACC)OP(A_1)\rightarrow ACC$，隐含约定的目的地址为ACC　完成一条指令需要访存2次

1个地址码字段占24位，指令操作数直接寻址范围为2^{24}=16M

指令格式-地址码

设指令字长及存储字长均为32位，操作码占8位

四地址指令

OP	A_1	A_2	A_3（结果）	A_4（下址）

指令含义：$(A_1)OP(A_2)\rightarrow A_3$，$A_4$=下一条将要执行指令的地址

4个地址码字段各占6位，指令操作数直接寻址范围为2^6=64；完成一条指令需要访存4次

三地址指令

OP	A_1	A_2	A_3（结果）

指令含义：$(A_1)OP(A_2)\rightarrow A_3$

3个地址码字段各占8位，指令操作数直接寻址范围为2^8=256；完成一条指令需要访存4次

二地址指令

OP	A_1（目的操作数）	A_2（源操作数）

指令含义：$(A_1)OP(A_2)\rightarrow A_1$

2个地址码字段各占12位，指令操作数直接寻址范围为2^{12}=4K；完成一条指令需要访存4次

一地址指令

OP	A_1

指令含义：1. $OP(A_1)\rightarrow A_1$，如加1、减1、取反、求补等　　完成一条指令需要访存3次

2. $(ACC)OP(A_1)\rightarrow ACC$，隐含约定的目的地址为ACC　　完成一条指令需要访存2次

1个地址码字段占24位，指令操作数直接寻址范围为2^{24}=16M

零地址指令

OP

指令含义：1. 不需要操作数，如空操作、停机、关中断等指令

2. 堆栈计算机，两个操作数隐含存放在栈顶和次栈顶，计算结果压回栈顶

指令字长

指令字长决定于
- 操作码的长度
- 操作数地址的长度
- 操作数地址的个数

1.指令字长 固定

指令字长 = 存储字长

2.指令字长 可变

按字节的倍数变化

小结

当用一些硬件资源代替指令字中的地址码字段后

- 可扩大指令的寻址范围
- 可缩短指令字长
- 可减少访存次数

本节内容

指令系统

扩展操作码

定长操作码

指令字的最高位部分分配固定的若干位表示操作码

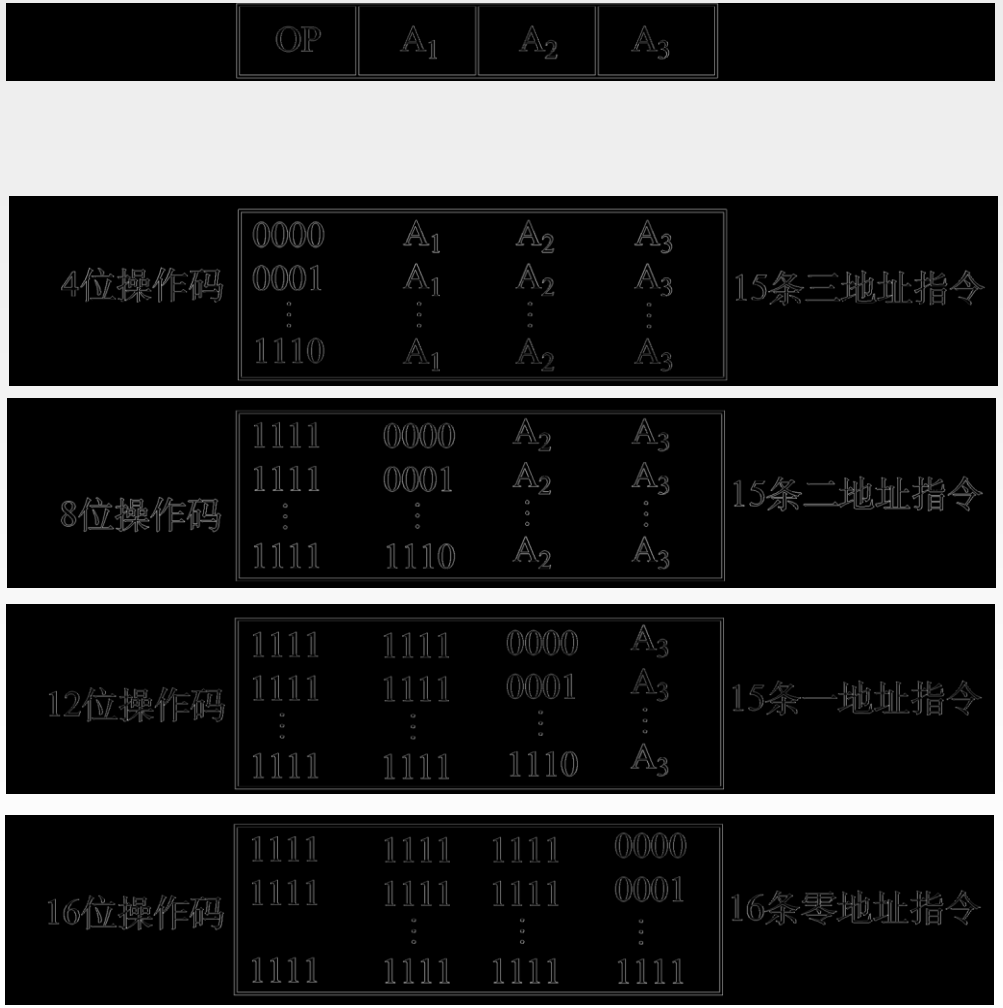
n位 → 2^n条指令

简化计算机硬件设计
提高指令译码和识别速度

用于指令字长较长的情况

扩展操作码

扩展操作码举例



- 在设计扩展操作码指令格式时，必须注意以下两点：

- 1）不允许短码是长码的前缀，即短操作码不能与长操作码的前面部分的代码相同。
- 2）各指令的操作码一定不能重复。

通常情况下，对使用频率较高的指令，分配较短的操作码；对使用频率较低的指令，分配较长的操作码，从而尽可能减少指令译码和分析的时间。

还有其他扩展操作码设计方法。

扩展操作码

扩展操作码举例

设指令字长固定为16位，试设计一套指令系统满足：

a) 有15条三地址指令
   共2^4=16种状态
   留出16-15=1种
b) 有12条二地址指令
   共1×2^4=16种
   留出16-12=4种
c) 有62条一地址指令
   共4×2^4=64种
   留出64-62=2种
d) 有32条零地址指令
   共2×2^4=32种

设地址长度为n，上一层留出m种状态，下一层可扩展出m× 2^n种状态

指令操作码

操作码指出指令中该指令应该执行什么性质的操作和具有何种功能。

操作码是识别指令、了解指令功能与区分操作数地址内容的组成和使用方法等的关键信息。例如，指出是算术加运算，还是减运算；是程序转移，还是返回操作。

操作码分类：

定长操作码： 在指令字的最高位部分分配固定的若干位（定长）表示操作码。

- 一般n位操作码字段的指令系统最大能够表示2^n条指令。
- 优：定长操作码对于简化计算机硬件设计，提高指令译码和识别速度很有利；
- 缺：指令数量增加时会占用更多固定位，留给表示操作数地址的位数受限。

扩展操作码(不定长操作码)：全部指令的操作码字段的位数不固定，且分散地放在指令字的不同位置上。

- 最常见的变长操作码方法是扩展操作码，使操作码的长度随地址码的减少而增加，不同地址数的指令可以具有不同长度的操作码，从而在满足需要的前提下，有效地缩短指令字长。
- 优： 在指令字长有限的前提下仍保持比较丰富的指令种类；
- 缺 ： 增加了指令译码和分析的难度，使控制器的设计复杂化。

本节内容

指令系统

操作类型

操作类型

1. 数据传送

源　　目的

LOAD　作用：把**存储器**中的数据放到**寄存器**中

STORE　作用：把**寄存器**中的数据放到**存储器**中

数据传送类：进行主存与CPU之间的数据传送

2. 算术逻辑操作

算术：加、减、乘、除、增 1、减 1、求补、浮点运算、十进制运算

逻辑：与、或、非、异或、位操作、位测试、位清除、位求反

运算类

3. 移位操作

算术移位、逻辑移位、循环移位(带进位和不带进位)

4. 转移操作

程序控制类：改变程序执行的顺序

无条件转移 JMP

条件转移　JZ：结果为0；JO：结果溢出；JC：结果有进位

调用和返回 CALL和RETURN

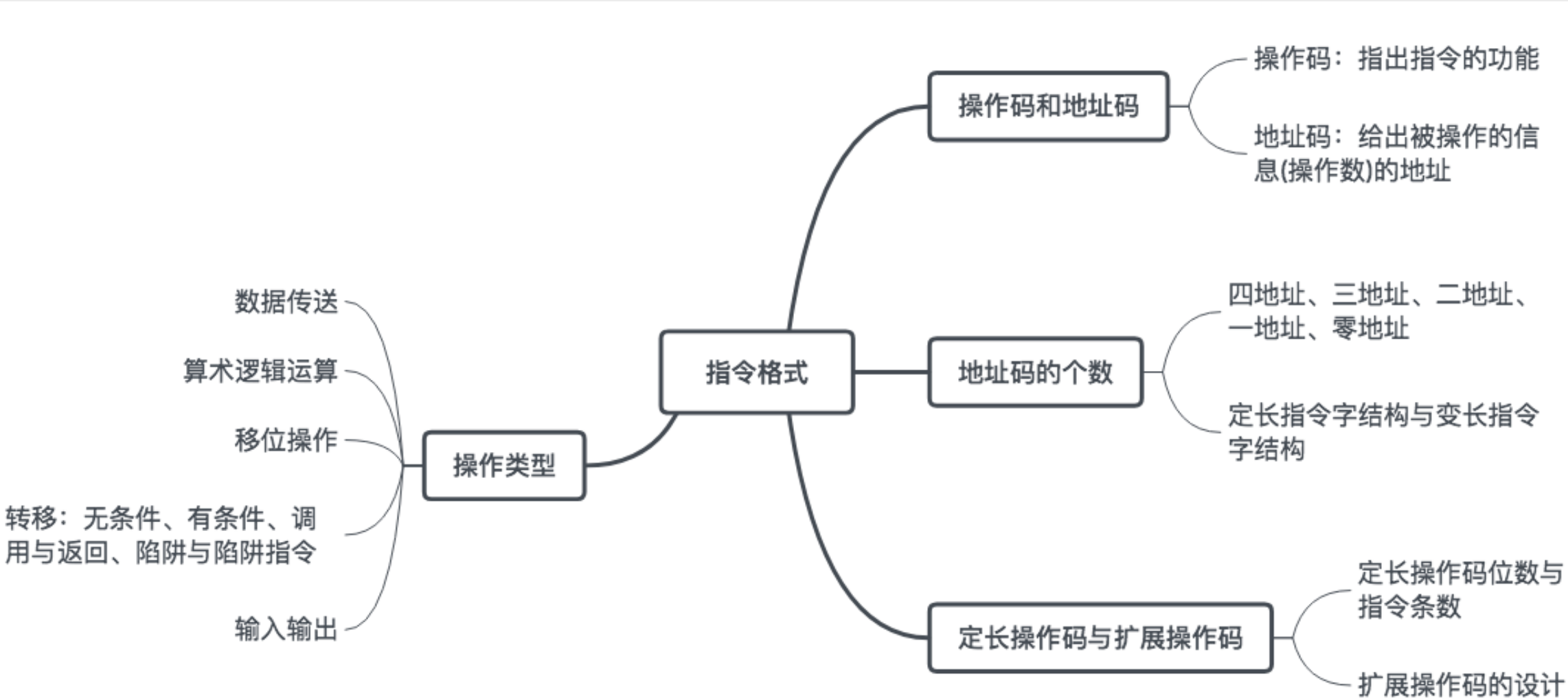
陷阱(Trap)与陷阱指令

5. 输入输出操作

输入输出类（I/O）：进行CPU和I/O设备之间的数据传送

CPU寄存器与IO端口之间的数据传送(端口即IO接口中的寄存器)

本节回顾



本节内容

指令系统

操作数类型

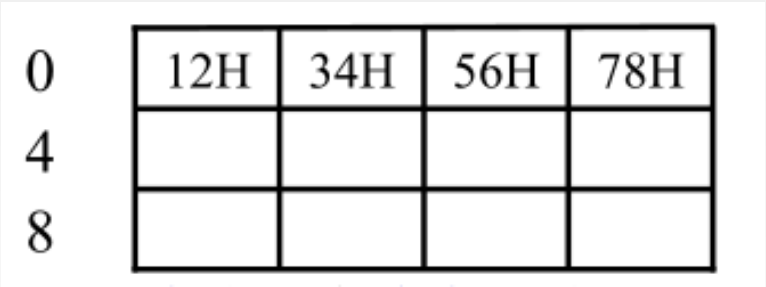
数据存放

操作数类型

地址	无符号整数
数字	定点数、浮点数、十进制数
字符	ASCII
逻辑数	逻辑运算

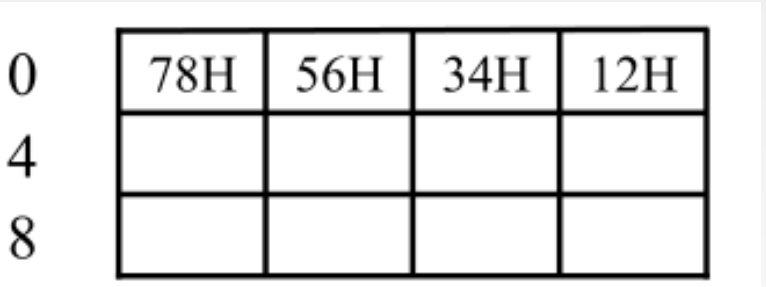
存放方式

12345678H的存放方式



字地址 为 高字节 地址

大端方式

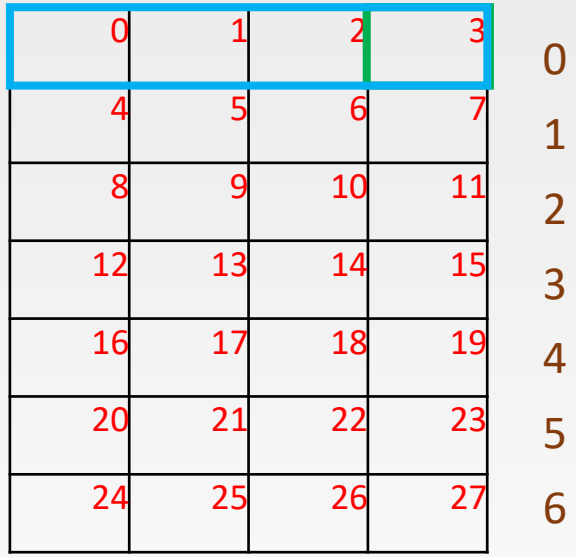


字地址 为 低字节 地址

小端方式

数据存放

字存储单元　　字节存储单元



按字地址寻址：给出一个字地址，可以取出长度为一个字的数据

字地址：
0～3为一个字，4～7为一个字…
每个字中最小的字节地址作为字地址

假设某数据长度为4B，则需要给出1个字地址

按字节编址：每个字节存储单元都有一个地址编号

按字节地址寻址：给出一个字节地址，可以取出长度为一个字节的数据

假设某数据长度为4B，则需要给出4个字节地址

按字编址：每个字存储单元对应一个地址编号

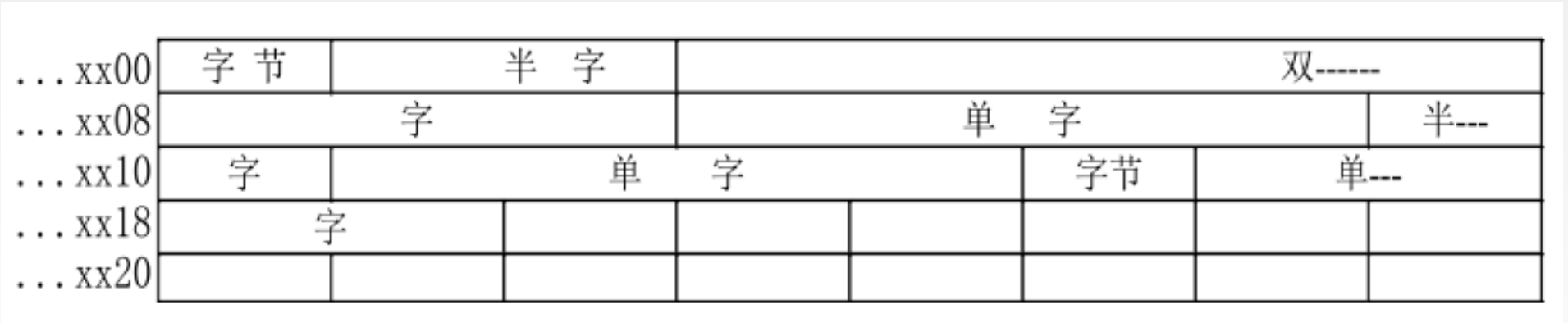
三个字长：
机器字长：CPU一次能处理的二进制数据的位数。　一般等于内部寄存器的位数。
指令字长：一个指令字中包含二进制代码的位数。　单字长指令：指令长度=机器字长
存储字长：一个存储单元存储二进制代码的长度。　半字长指令、双字长指令

注：这些长度都是字节的整数倍　通常一次并行取出一个存储字，按字节寻址时通过移位截取存储字中的一个字节

存放方式

字节编址，数据在存储器中的存放方式（存储字长64位，机器字长32位）

从任意位置开始存储

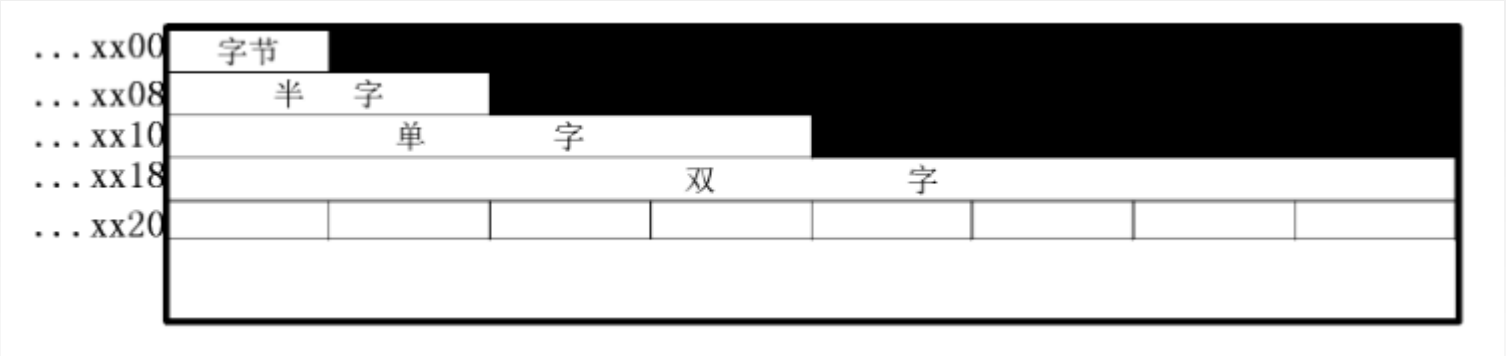


优点：不浪费存储资源

缺点：除了访问一个字节之外，访问其它任何类型的数据，都可能花费两个存储周期的时间。读写控制比较复杂。

存放方式

从一个存储字的起始位置开始访问

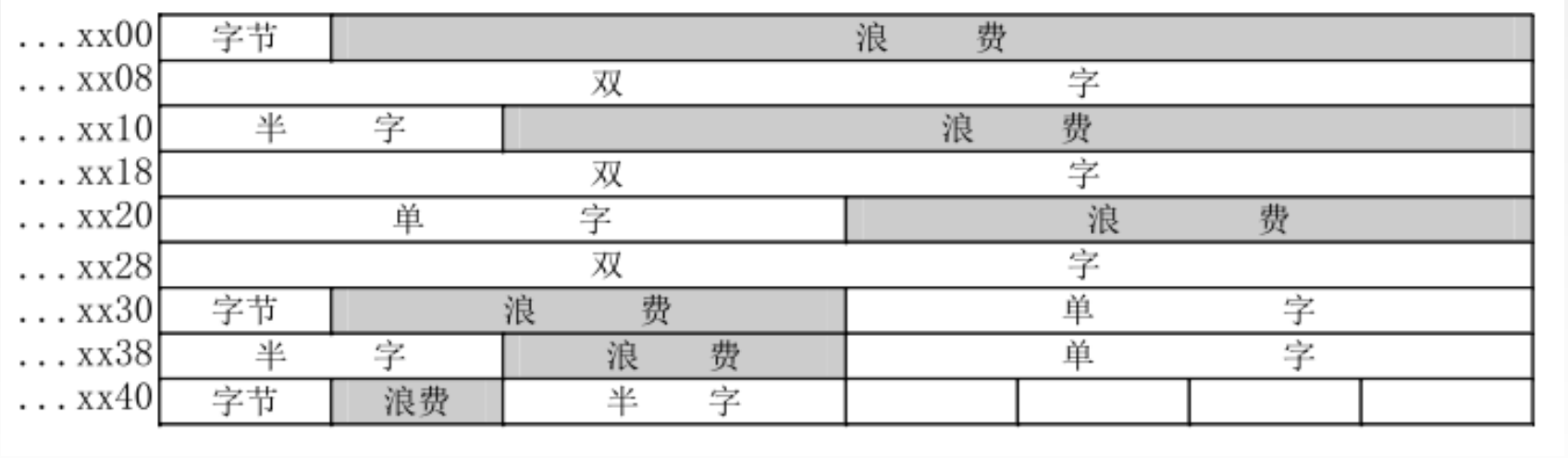


优点：无论访问何种类型的数据，在一个周期内均可完成，读写控制简单。
缺点：浪费了宝贵的存储资源

存放方式

由于不同的机器数据字长不同，每台机器处理的数据字长也不统一，为了便于硬件实现，通常要求多字节的数据在存储器的存放方式能满足“边界对准”的要求。

边界对准方式—— 从地址的整数倍位置开始访问



数据存放的起始地址是数据 长度（按照编址单位进行计算）的 整数倍

本方案是前两个方案的折衷，在一个周期内可以完成存储访问，空间浪费也不太严重

本节内容

指令系统

寻址方式
指令寻址

寻址方式

寻址方式

确定 本条指令 的 操作数地址

下一条 欲执行 指令 的 指令地址

寻址方式 { 指令寻址 / 数据寻址

指令寻址

始终由程序计数器PC给出

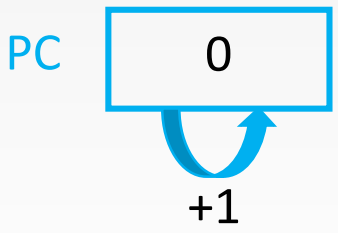
顺序寻址 (PC)+1 ⟶ PC

跳跃寻址 由转移指令指出

指令寻址

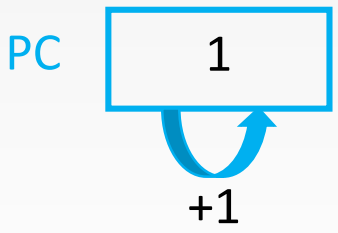
0001001111101000
0011001111101001
0010010010110000
1001000000000111
0001011111010000
0100011111010001
0101011111010001
0001100111000100
...

指令寻址



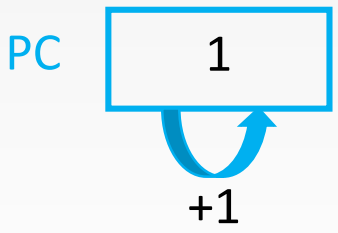
指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8	…	

指令寻址



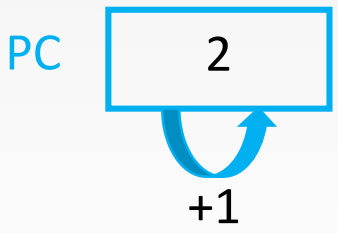
指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8	…	

指令寻址



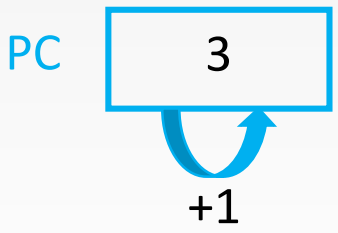
指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8	…	

指令寻址



指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8	…	

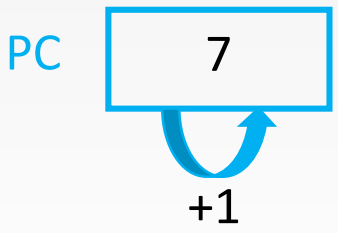
指令寻址



指令地址	操作码	地址码
0	LDA	1000
1	ADD	1001
2	DEC	1200
3	JMP	7
4	LDA	2000
5	SUB	2001
6	INC	
7	LDA	1100
8	…	

JMP：无条件转移
把PC中的内容改成7

指令寻址



指令地址		操作码	地址码
	0	LDA	1000
顺序寻址	1	ADD	1001
顺序寻址	2	DEC	1200
顺序寻址	3	JMP	7
	4	LDA	2000
	5	SUB	2001
	6	INC	
跳跃寻址	7	LDA	1100
	8	…	

JMP：无条件转移
把PC中的内容改成7

指令寻址

始终由程序计数器PC给出

顺序寻址 (PC)+1 ⟶ PC

跳跃寻址 由转移指令指出

本节内容

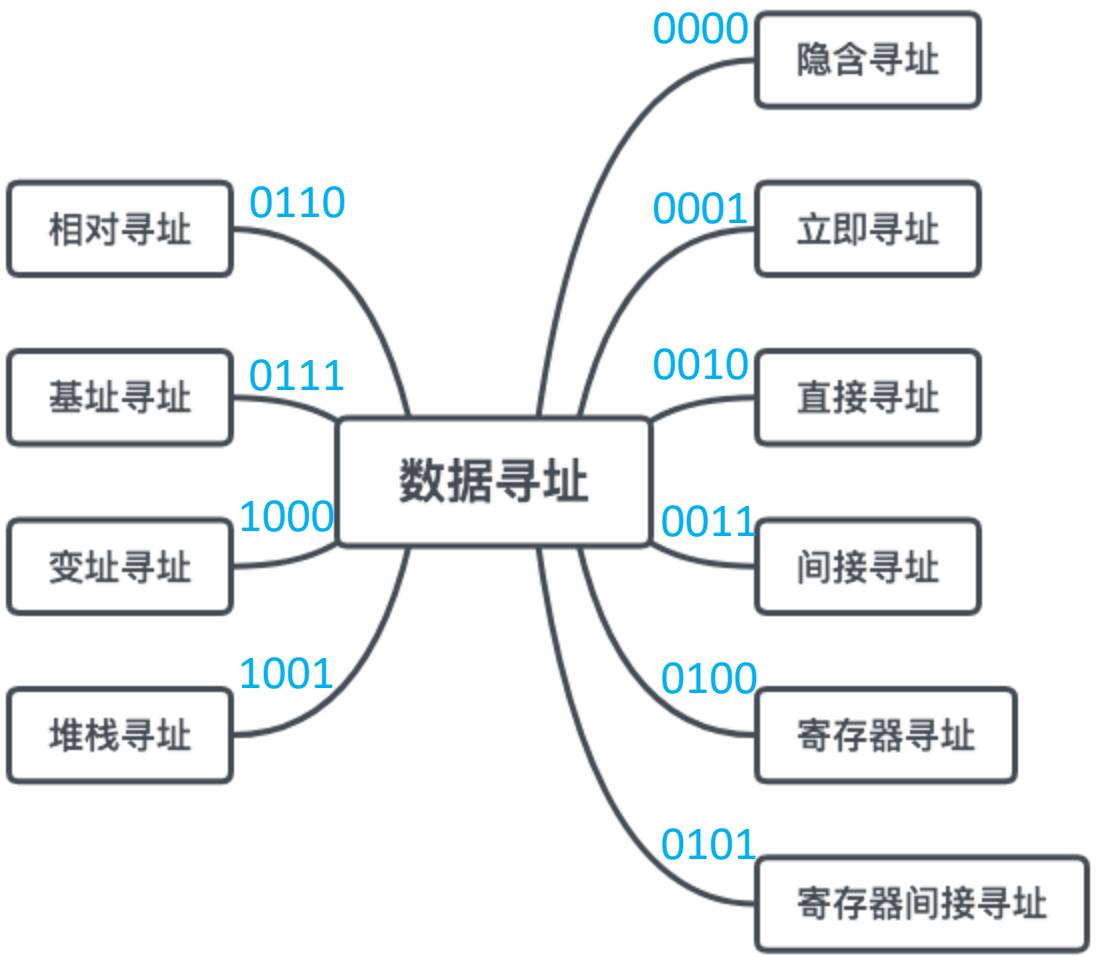
指令系统

数据寻址-1

数据寻址

一地址指令

操作码（OP）	地址码(A)



寻址方式位

数据寻址

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

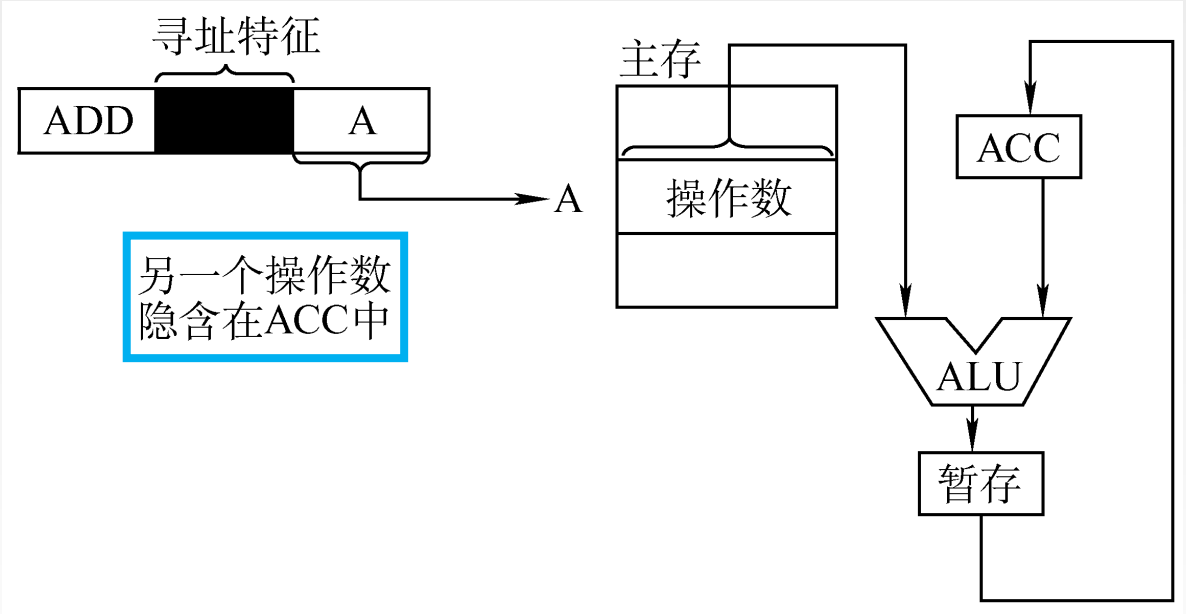
一地址指令

求出操作数的真实地址，称为有效地址(EA)。

二地址指令

隐含寻址

隐含寻址：不是明显地给出操作数的地址，而是在指令中隐含着操作数的地址 。



优点：有利于缩短指令字长。
缺点：需增加存储操作数或隐含地址的硬件。

数据寻址

假设操作数为3

立即寻址

假设操作数为3

操作码（OP）	#	0…011

立即寻址：形式地址A就是操作数本身，又称为立即数，一般采用补码形式。
#表示立即寻址特征。

一条指令的执行：
取指令 访存1次
执行指令 访存0次
暂不考虑存结果
共访存1次

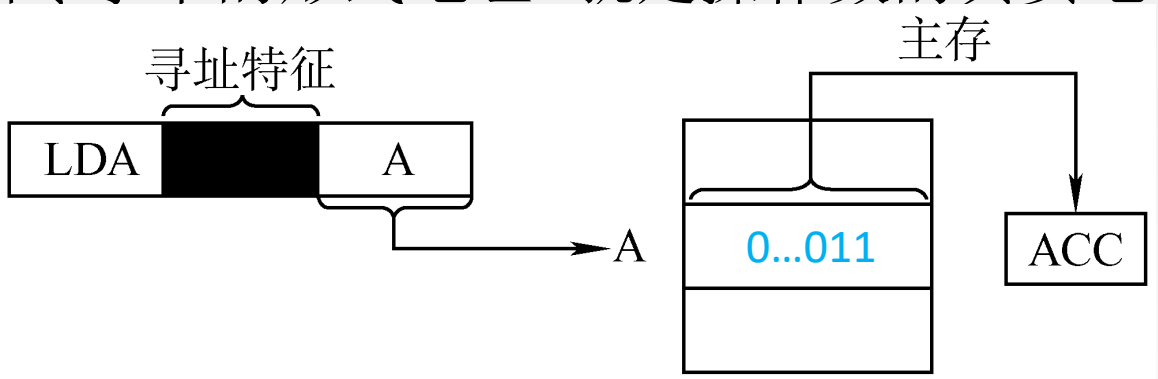
优点：指令执行阶段不访问主存，指令执行时间最短

缺点：
A的位数限制了立即数的范围。
如A的位数为n，且立即数采用补码时，
可表示的数据范围为$-2^{n-1}\sim 2^{n-1}-1$

直接寻址

假设操作数为3

直接寻址：指令字中的形式地址A就是操作数的真实地址EA，即EA=A 。



一条指令的执行：
取指令 访存1次
执行指令 访存1次
暂不考虑存结果
共访存2次

优点：简单，指令执行阶段仅访问一次主存，
不需专门计算操作数的地址。

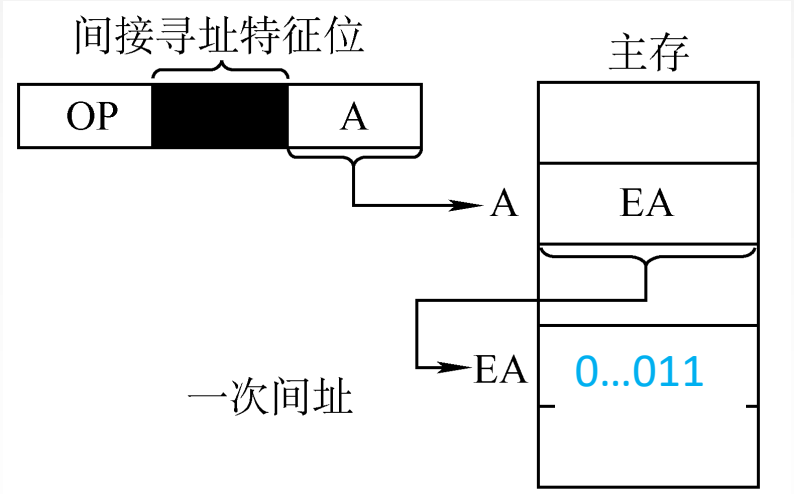
缺点：
A的位数决定了该指令操作数的寻址范围。
操作数的地址不易修改。

间接寻址

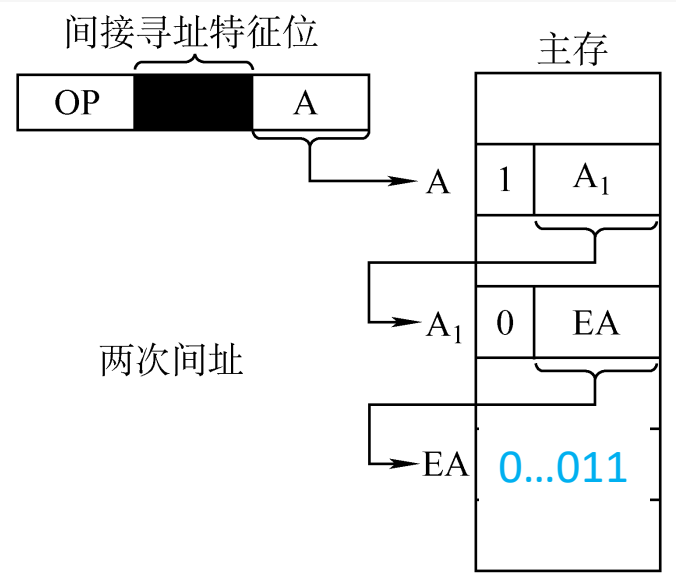
假设操作数为3

操作码（OP）		1001…0111

间接寻址：指令的地址字段给出的形式地址不是操作数的真正地址，而是操作数有效地址所在的存储单元的地址，也就是操作数地址的地址，即EA=(A)。



一条指令的执行：
取指令 访存1次
执行指令 访存2次
暂不考虑存结果
共访存3次

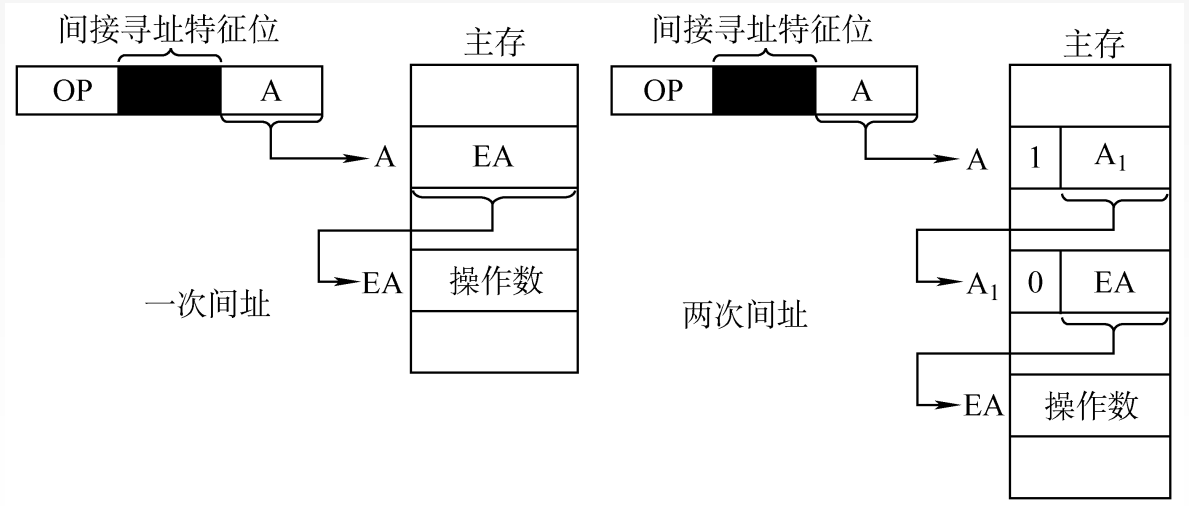


间接寻址

假设操作数为3

操作码（OP）		1001…0111

间接寻址：指令的地址字段给出的形式地址不是操作数的真正地址，而是操作数有效地址所在的存储单元的地址，也就是操作数地址的地址，即EA=(A)。



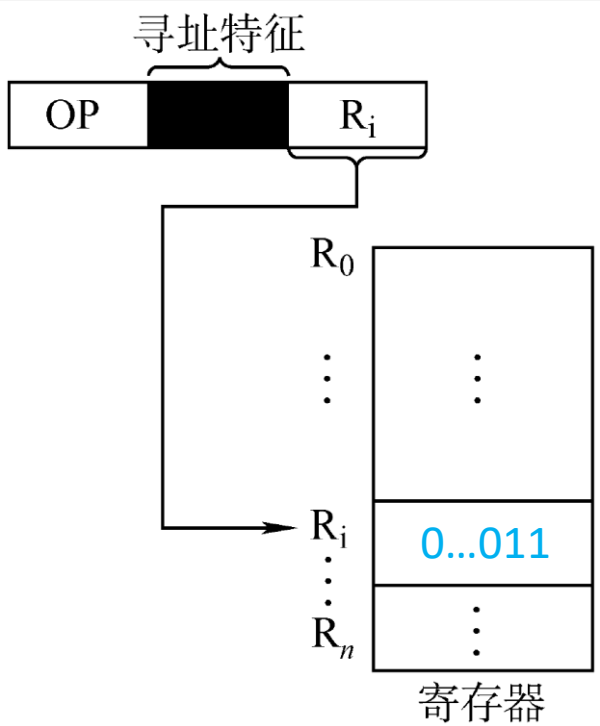
优点：
可扩大寻址范围(有效地址EA的位数大于形式地址A的位数)。
便于编制程序(用间接寻址可以方便地完成子程序返回)。

缺点：
指令在执行阶段要多次访存(一次间址需两次访存，多次寻址需根据存储字的最高位确定几次访存)。

寄存器寻址

假设操作数为3

寄存器寻址：在指令字中直接给出操作数所在的寄存器编号，即EA =R_i，其操作数在由R_i所指的寄存器内。



一条指令的执行：
取指令 访存1次
执行指令 访存0次
暂不考虑存结果
共访存1次

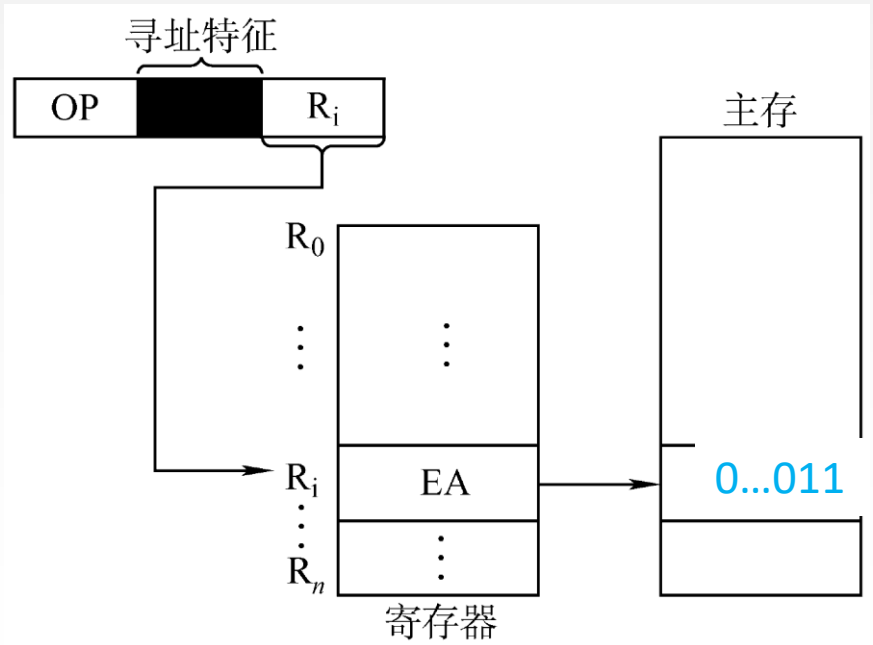
优点：
指令在执行阶段不访问主存，只访问寄存器，指令字短且执行速度快，支持向量/矩阵运算。

缺点：
寄存器价格昂贵，计算机中寄存器个数有限 。

寄存器间接寻址

假设操作数为3

寄存器间接寻址：寄存器R_i中给出的不是一个操作数，而是操作数所在主存单元的地址即EA=(R_i)。



一条指令的执行：
取指令 访存1次
执行指令 访存1次
暂不考虑存结果
共访存2次

特点：
与一般间接寻址相比速度更快，
但指令的执行阶段需要访问主存
(因为操作数在主存中)。

本节回顾

寻 址 方 式	有 效 地 址	访 存 次 数(指令执行期间)
隐含寻址	程序指定	0
立即寻址	A即是操作数	0
直接寻址	EA=A	1
一次间接寻址	EA=(A)	2
寄存器寻址	EA=R_i	0
寄存器间接一次寻址	EA=(R_i)	1

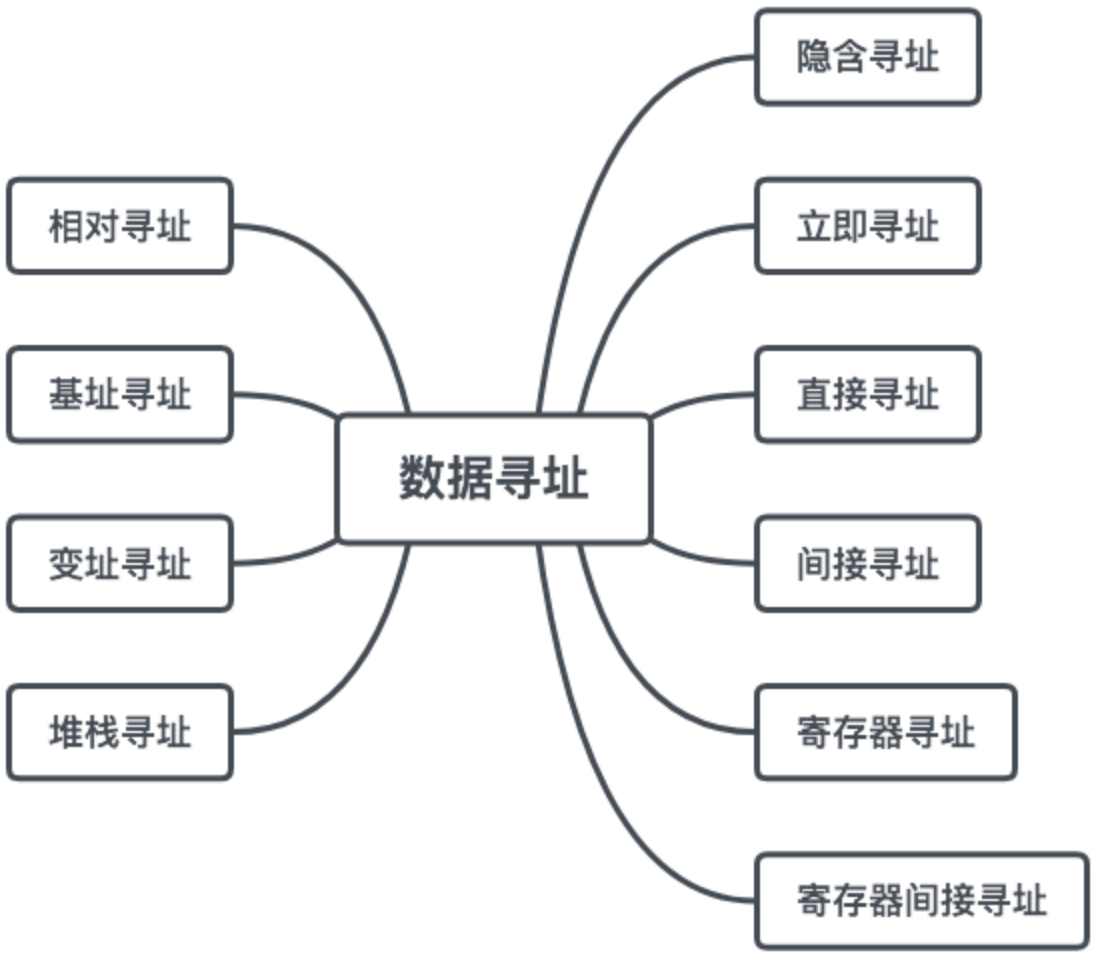
本节内容

指令系统

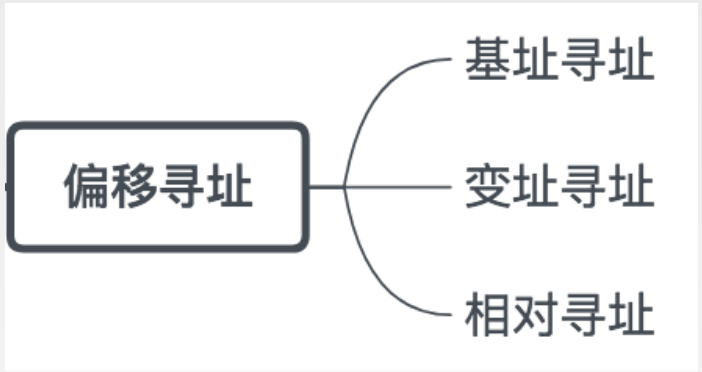
数据寻址-2
偏移寻址

本节总览

操作码（OP）	寻址特征	形式地址(A)



偏移寻址

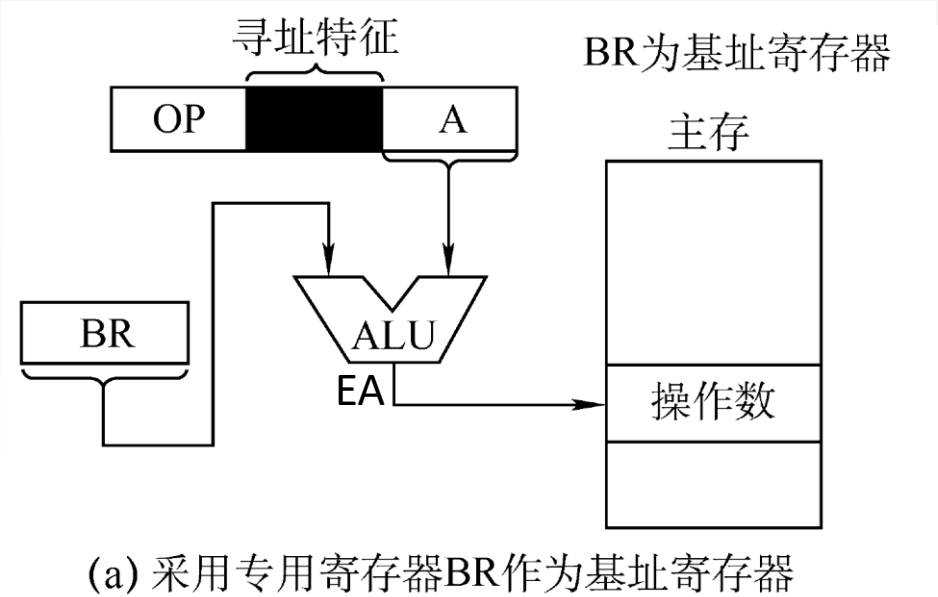


EA=(BR)+A

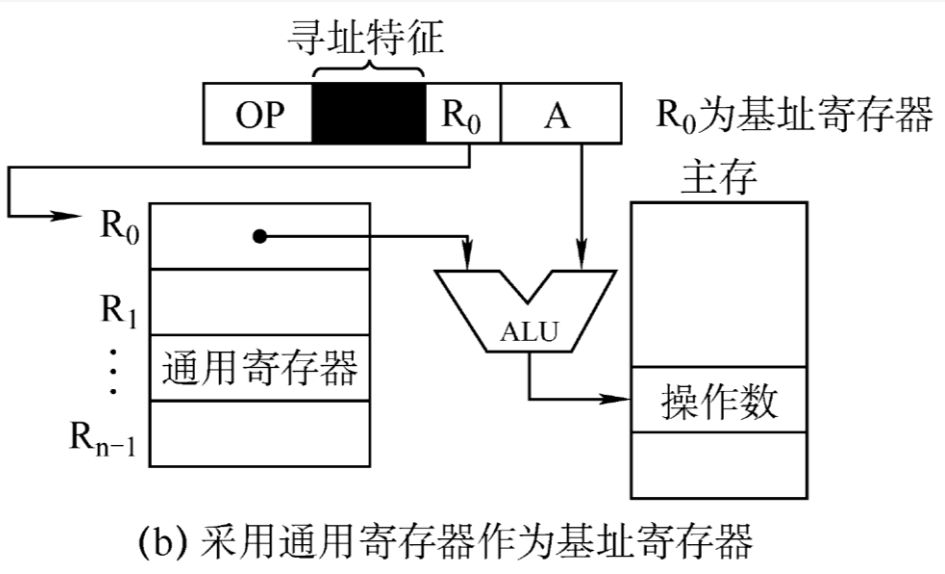
EA=(IX)+A

EA=(PC)+A

基址寻址：将CPU中基址寄存器（BR）的内容加上指令格式中的形式地址A，而形成操作数的有效地址，即EA=(BR)+A。



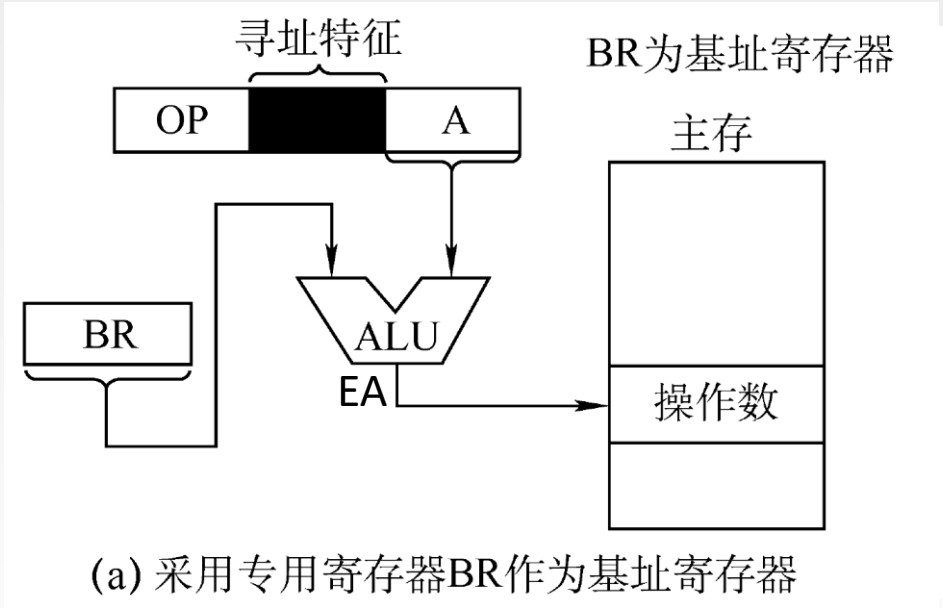
(a) 采用专用寄存器BR作为基址寄存器



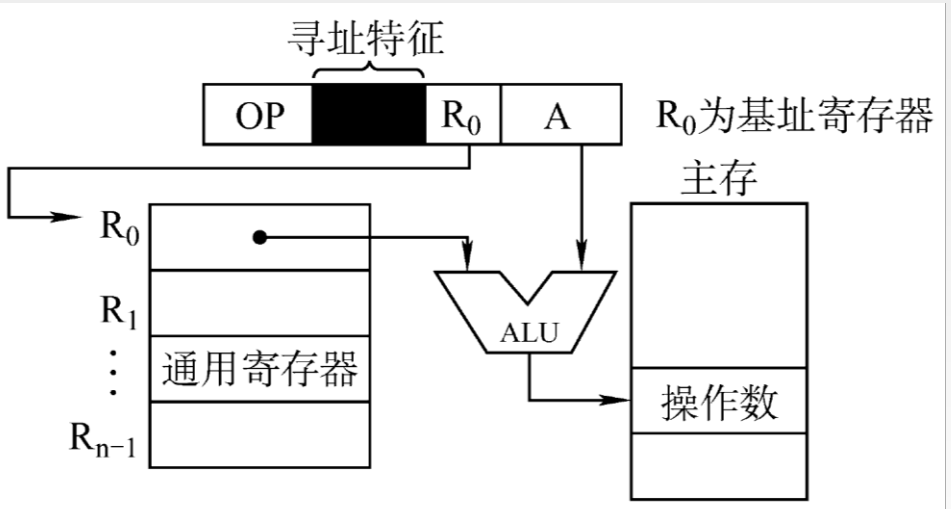
(b) 采用通用寄存器作为基址寄存器

基址寻址

基址寻址：将CPU中基址寄存器（BR）的内容加上指令格式中的形式地址A，而形成操作数的有效地址，即EA=(BR)+A。



(a) 采用专用寄存器BR作为基址寄存器



(b) 采用通用寄存器作为基址寄存器

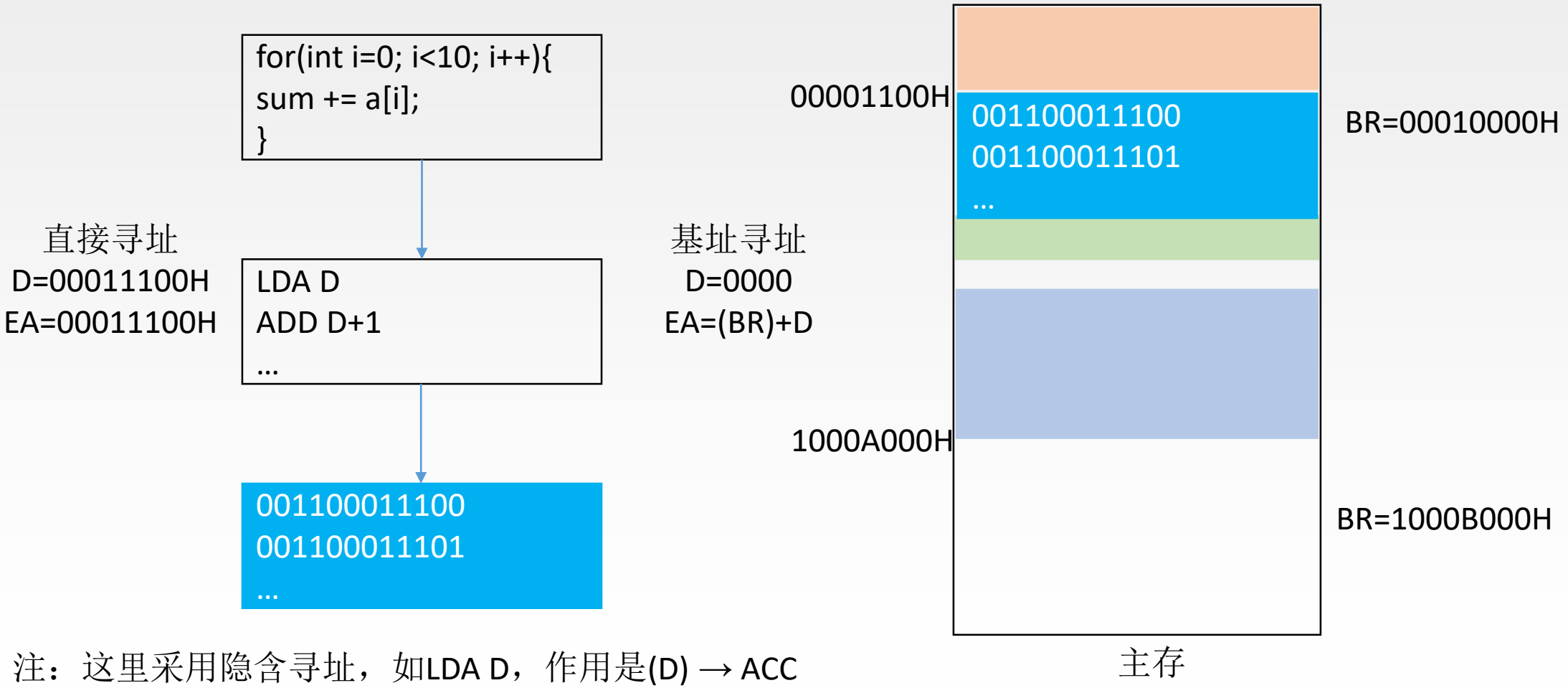
注：基址寄存器是**面向操作系统**的，**其内容由操作系统或管理程序确定**。在程序执行过程中，基址寄存器的内容不变（作为基地址），形式地址可变（作为偏移量）。

当采用通用寄存器作为基址寄存器时，可由**用户决定哪个寄存器作为基址寄存器**，但其**内容仍由操作系统确定**。

优点：可扩大寻址范围（基址寄存器的位数大于形式地址A的位数）；用户不必考虑自己的程序存于主存的哪一空间区域，故**有利于多道程序设计**，以及可用于编制浮动程序。

基址寻址

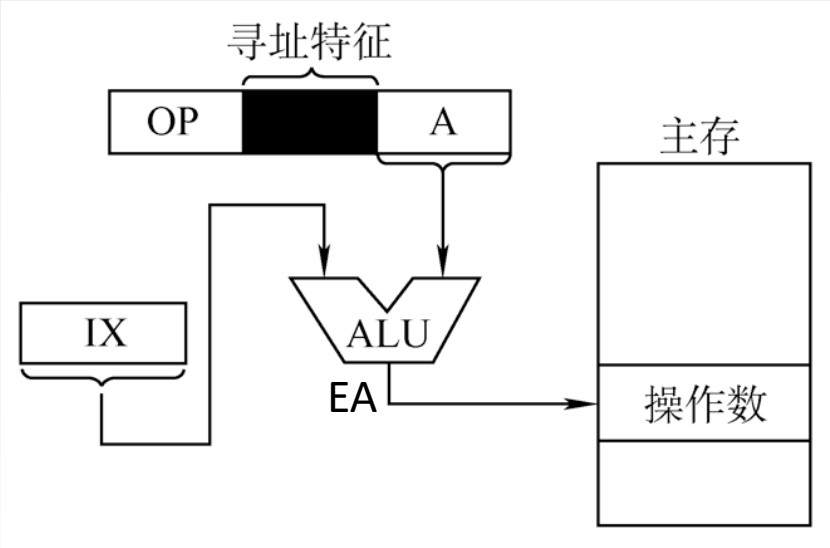
优点：可扩大寻址范围（基址寄存器的位数大于形式地址A的位数）；用户不必考虑自己的程序存于主存的哪一空间区域，故**有利于多道程序设计**，以及可用于编制浮动程序。



注：这里采用隐含寻址，如LDA D，作用是(D) → ACC

变址寻址

变址寻址：有效地址EA等于指令字中的形式地址A与变址寄存器IX的内容相加之和，即EA= (IX)+A，其中IX为变址寄存器（专用），也可用通用寄存器作为变址寄存器 。



注：变址寄存器是**面向用户**的，在程序执行过程中，**变址寄存器的内容可由用户改变**（作为偏移量），形式地址A不变（作为基地址）。

优点：可**扩大寻址范围**（变址寄存器的位数大于形式地址A的位数）；在数组处理过程中，可设定A为数组的首地址，不断改变变址寄存器IX的内容，便可很容易形成数组中任一数据的地址，特别**适合编制循环程序**。

基址寻址与变址寻址

优点：可**扩大寻址范围**（变址寄存器的位数大于形式地址A的位数）；在数组处理过程中，可设定A为数组的首地址，不断改变变址寄存器IX的内容，便可很容易形成数组中任一数据的地址，特别**适合编制循环程序**。

```
for(int i=0; i<10; i++){
sum += a[i];
}
```

基址寻址
D=0000
EA=(BR)+D

```
LDA D
ADD D+1
...
```

```
001100011100
001100011101
...
```

基址寻址与变址寻址

优点：可**扩大寻址范围**（变址寄存器的位数大于形式地址A的位数）；在数组处理过程中，可设定A为数组的首地址，不断改变变址寄存器IX的内容，便可很容易形成数组中任一数据的地址，特别**适合编制循环程序**。

```
for(int i=0; i<10; i++){
sum += a[i];
}
```

基址寻址
D=0000
EA=(BR)+D

```
LDA D
ADD D+1
…
ADD D+9
```

直接寻址
D=00011100H
EA=00011100H

采用变址寄存器X

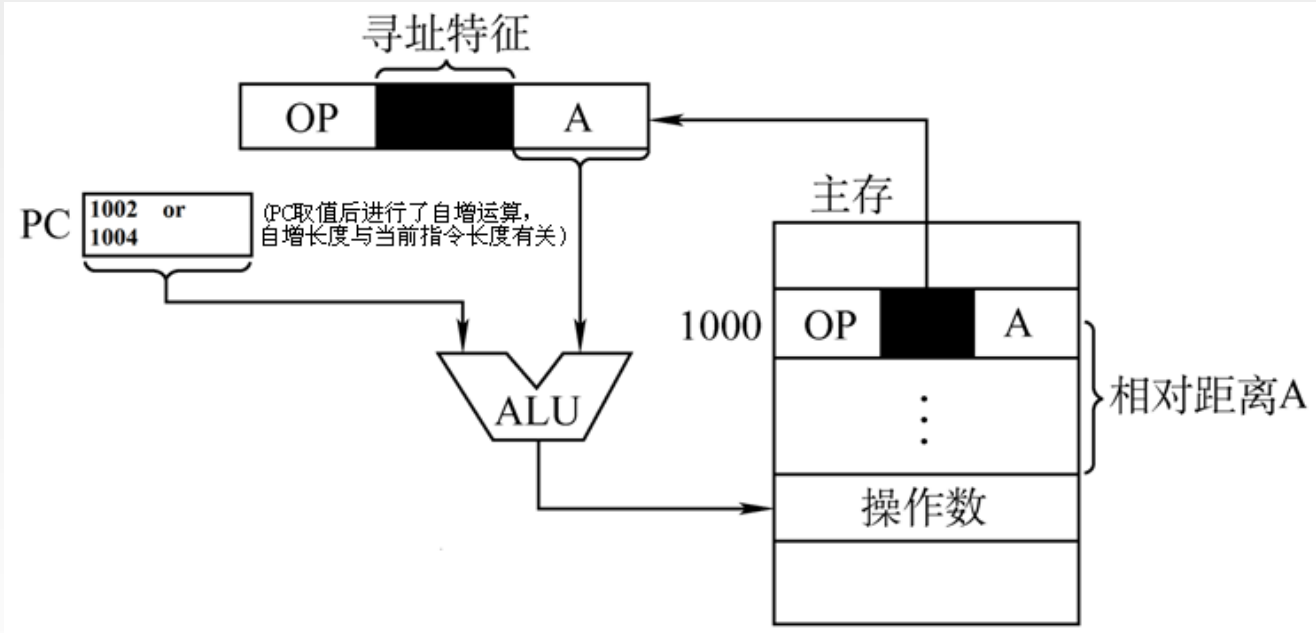
M-2	LDA #0	0 → ACC
M-1	LDX #0	0 → X
M	ADD X,D	(ACC)+(D+(X)) → ACC
M+1	INX	(X)+1 → X
M+2	CPX #10	(X)-10，若结果为0，则Z=1；若结果非0，则Z=0
M+3	BNE M	若Z=0，M → PC；若Z=1，(PC)+1 → PC

变址寻址与基址寻址配合使用：EA=A+(BR)+(IX)

变址寻址与间接寻址配合使用：
如先变址后间址，EA=(A+(IX))；
先间址后变址，EA=(A)+(IX)。

相对寻址

相对寻址：把程序计数器PC的内容加上指令格式中的形式地址A而形成操作数的有效地址，即EA=(PC)+A，其中A是相对于当前指令地址的位移量，可正可负，补码表示 。



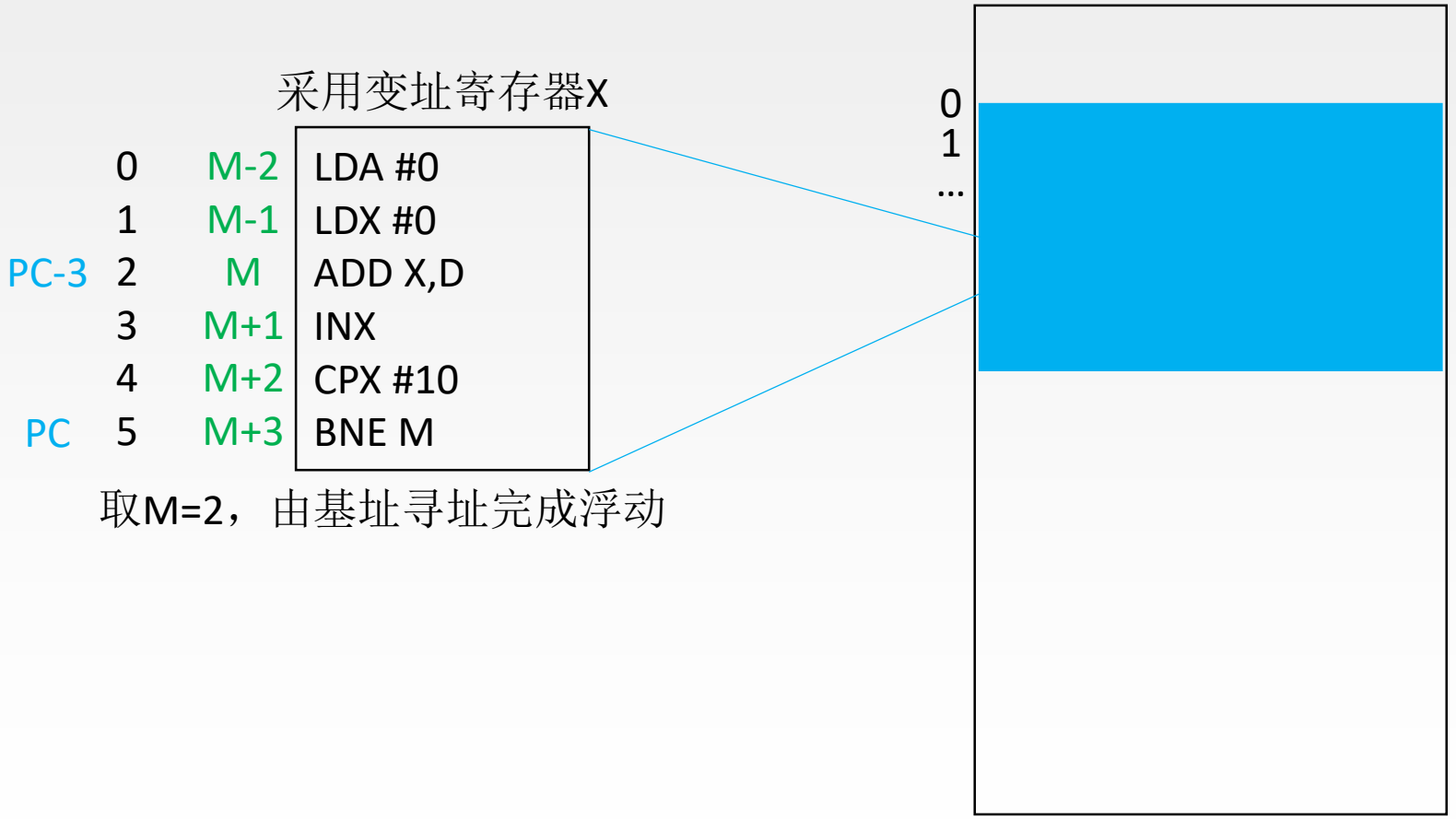
优点：操作数的地址不是固定的，它随着PC值的变化而变化，并且与指令地址之间总是相差一个固定值，因此便于程序浮动。

相对寻址广泛应**用于转移指令**。

相对寻址

优点：操作数的地址不是固定的，它随着PC值的变化而变化，并且与指令地址之间总是相差一个固定值，因此便于程序浮动。

相对寻址广泛应**用于转移指令**。

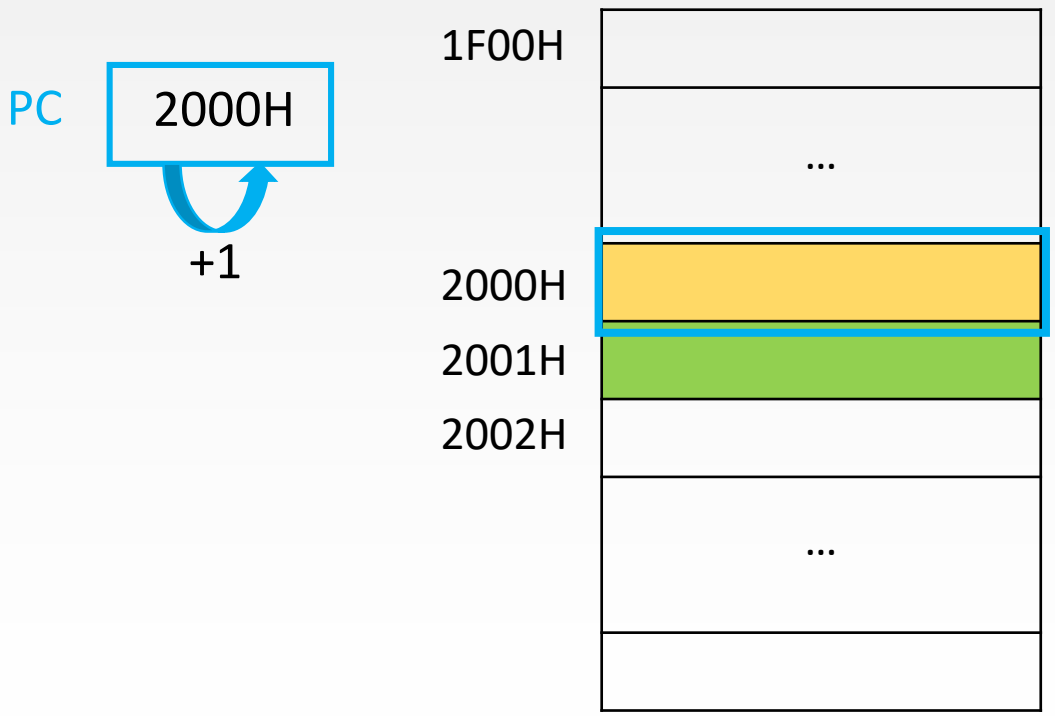


相对寻址举例

某机器指令字长为16位，主存按字节编址，取指令时，每取一个字节PC自动加1。当前指令地址为2000H，指令内容为相对寻址的无条件转移指令，指令中第一个字节为操作码，第二个字节为形式地址，当前形式地址为40H。

(1) 求取指令后及指令执行后PC内容。

(2) 若要求转移到1F00H， 求形式地址的内容。

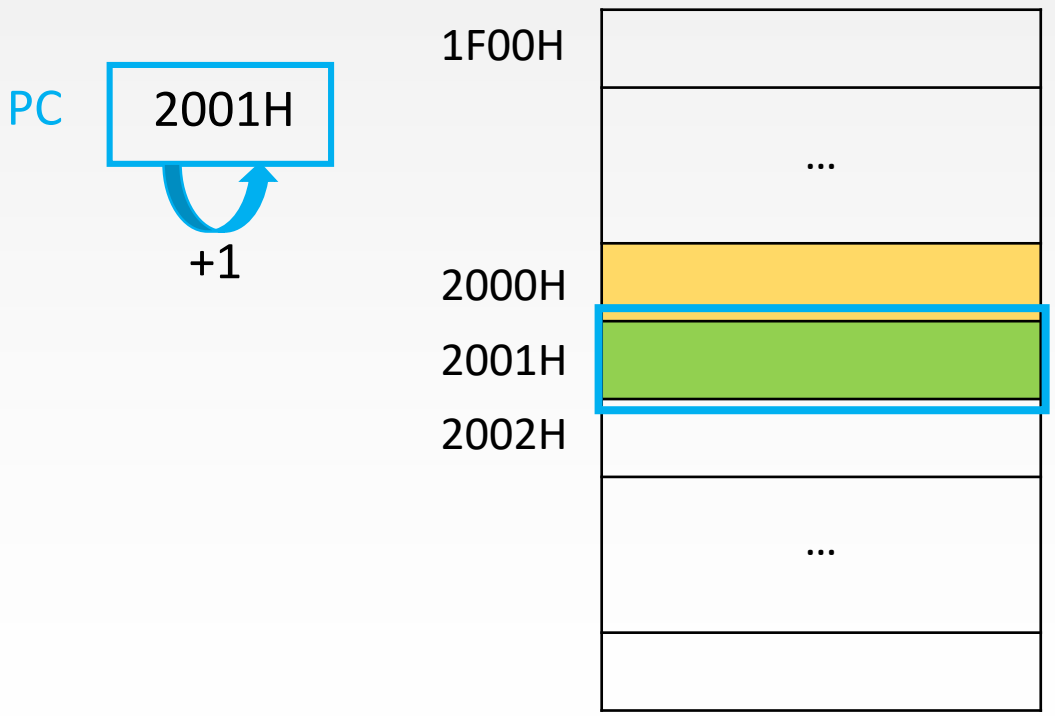


相对寻址举例

某机器指令字长为16位，主存按字节编址，取指令时，每取一个字节PC自动加1。当前指令地址为2000H，指令内容为相对寻址的无条件转移指令，指令中第一个字节为操作码，第二个字节为形式地址，当前形式地址为40H。

(1) 求取指令后及指令执行后PC内容。

(2) 若要求转移到1F00H， 求形式地址的内容。

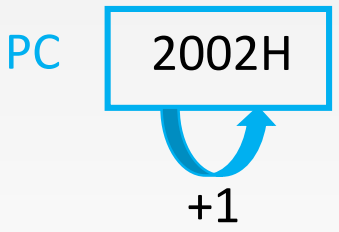


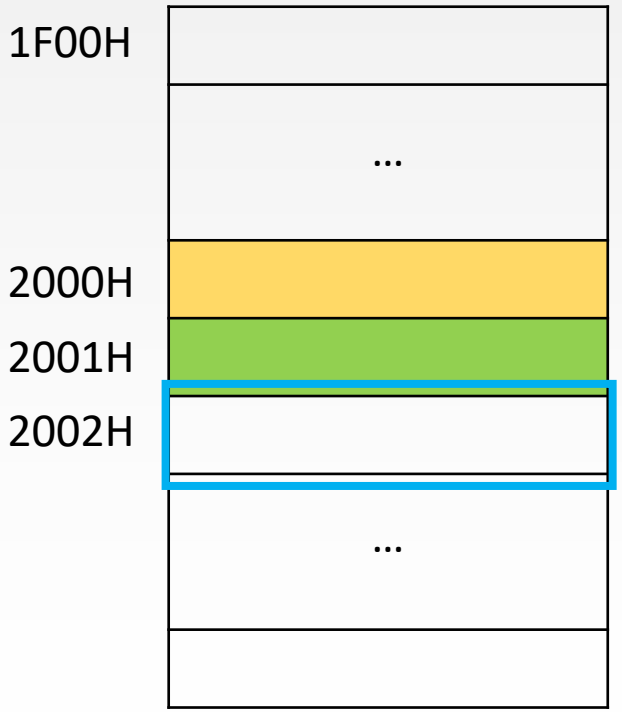
相对寻址举例

某机器指令字长为16位，主存按字节编址，取指令时，每取一个字节PC自动加1。当前指令地址为2000H，指令内容为相对寻址的无条件转移指令，指令中第一个字节为操作码，第二个字节为形式地址，当前形式地址为40H。

(1) 求取指令后及指令执行后PC内容。

(2) 若要求转移到1F00H， 求形式地址的内容。





CPU分析指令的结果：是一条无条件转移指令，

EA=(PC)+A

= 2002H + 40H

=2042H

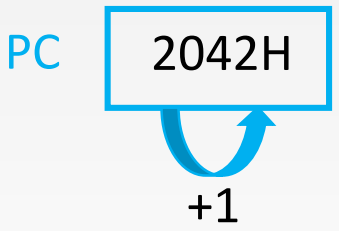
故取指令后PC内容为2002H，指令执行后PC内容为2042H

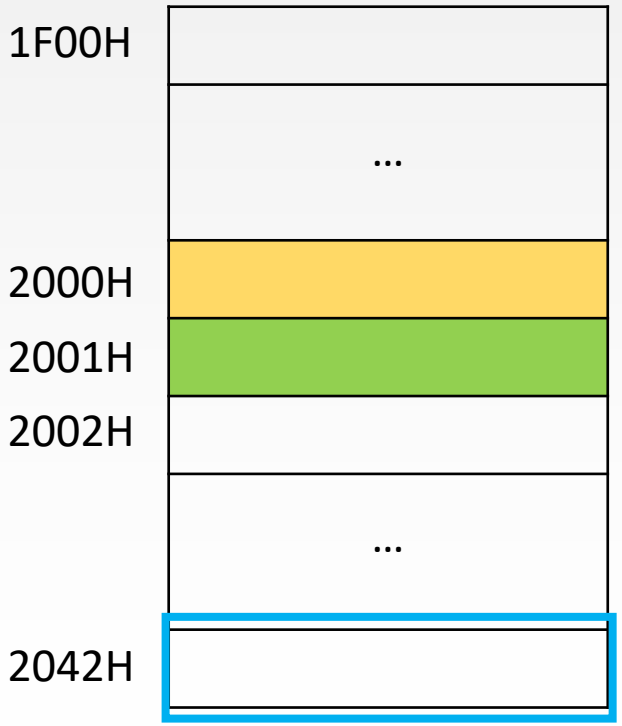
相对寻址举例

某机器指令字长为16位，主存按字节编址，取指令时，每取一个字节PC自动加1。当前指令地址为2000H，指令内容为相对寻址的无条件转移指令，指令中第一个字节为操作码，第二个字节为形式地址，当前形式地址为40H。

(1) 求取指令后及指令执行后PC内容。

(2) 若要求转移到1F00H， 求形式地址的内容。





CPU分析指令的结果：是一条无条件转移指令，

EA=(PC)+A

= 2002H + 40H

=2042H

故取指令后PC内容为2002H，指令执行后PC内容为2042H

相对寻址举例

某机器指令字长为16位，主存按字节编址，取指令时，每取一个字节PC自动加1。当前指令地址为2000H，指令内容为相对寻址的无条件转移指令，指令中第一个字节为操作码，第二个字节为形式地址，当前形式地址为40H。

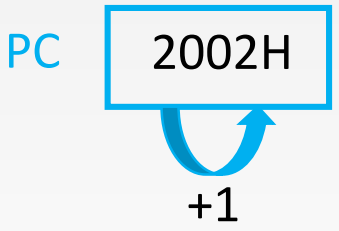
(1) 求取指令后及指令执行后PC内容。

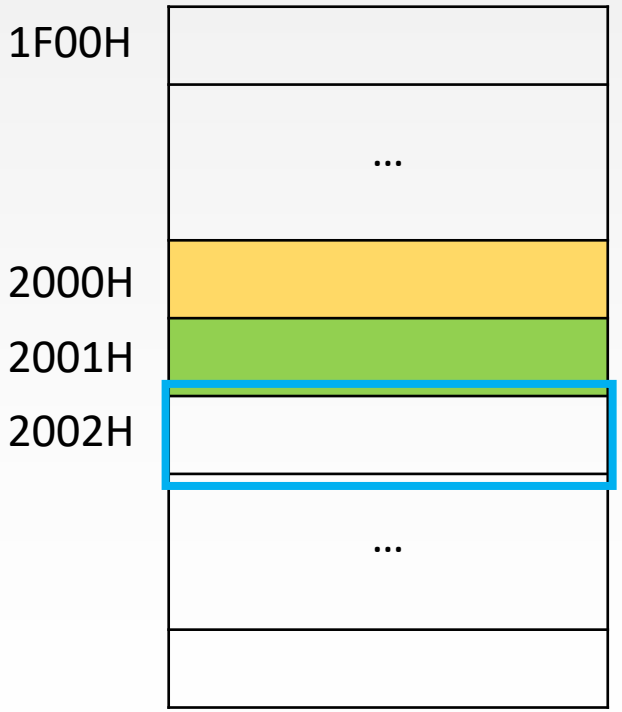
(2) 若要求转移到1F00H， 求形式地址的内容。

即 (PC) + A =2002H + A = 1F00H

A = 1F00H – 2002H = 1EFFH+1H – 2002H

=1EFFH – 2002H + 1H = FEFDH +1H = FEFEH





CPU分析指令的结果：是一条无条件转移指令，

EA=(PC)+A

= 2002H + 40H

=2042H

故取指令后PC内容为2002H，指令执行后PC内容为2042H

本节回顾

寻 址 方 式	有 效 地 址	访 存 次 数(指令执行期间)
隐含寻址	程序指定	0
立即寻址	A即是操作数	0
直接寻址	EA=A	1
一次间接寻址	EA=(A)	2
寄存器寻址	EA=R_i	0
寄存器间接一次寻址	EA=(R_i)	1
转移指令　相对寻址	EA=(PC)+A	1
多道程序　基址寻址	EA=(BR)+A	1
循环程序　变址寻址　数组问题	EA=(IX)+A	1

偏移寻址

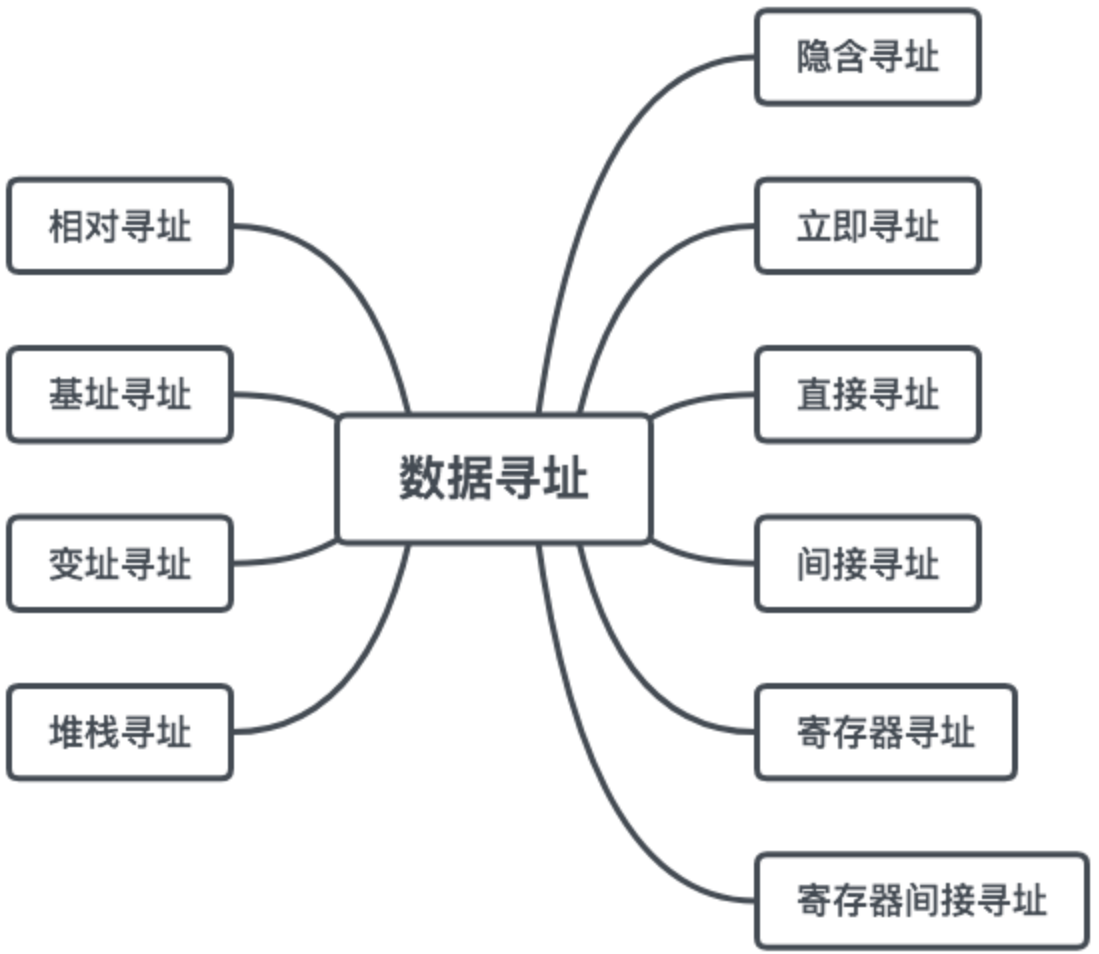
本节内容

指令系统

数据寻址-3
堆栈寻址

本节总览

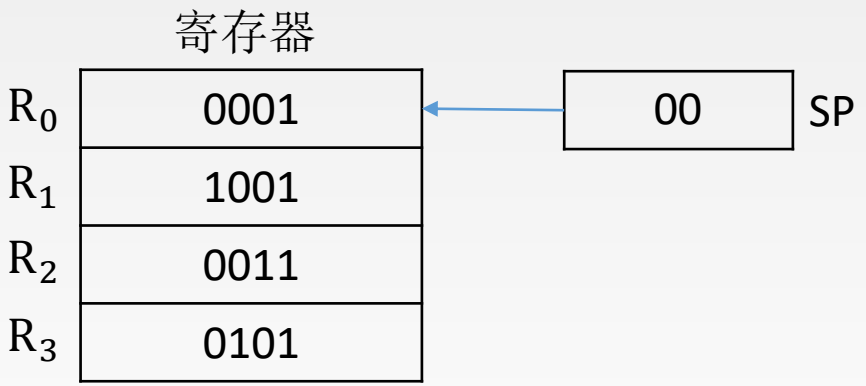
操作码（OP）	寻址特征	形式地址(A)



堆栈寻址

堆栈寻址：操作数存放在堆栈中，隐含使用堆栈指针(SP)作为操作数地址。

堆栈是存储器（或专用寄存器组）中一块特定的按“后进先出（LIFO）”原则管理的存储区，该存储区中被读/写单元的地址是用一个特定的寄存器给出的，该寄存器称为堆栈指针（SP）。

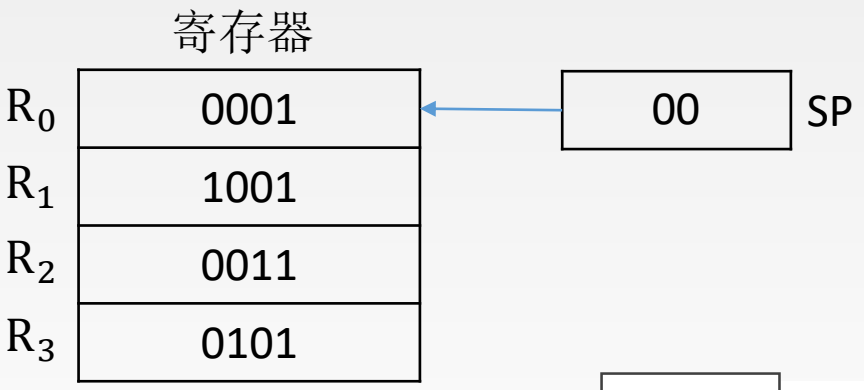


假设SP指向栈顶元素，R_0为栈顶

堆栈寻址

堆栈寻址：操作数存放在堆栈中，隐含使用堆栈指针(SP)作为操作数地址。

堆栈是存储器（或专用寄存器组）中一块特定的按“后进先出（LIFO）”原则管理的存储区，该存储区中被读/写单元的地址是用一个特定的寄存器给出的，该寄存器称为堆栈指针（SP）。



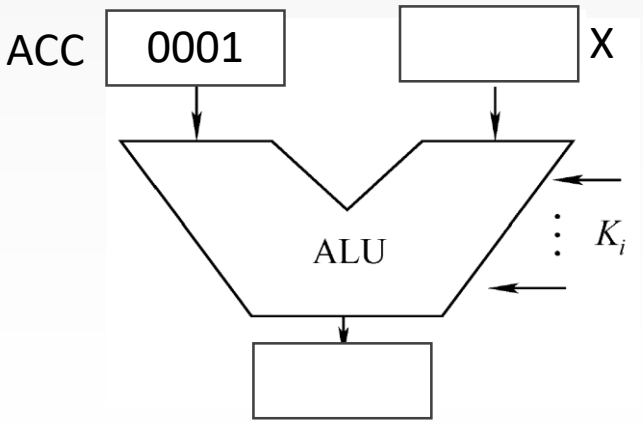
假设SP指向栈顶元素，R_0为栈顶

完成一次加法

记栈顶单元为Msp

POP (Msp) → ACC

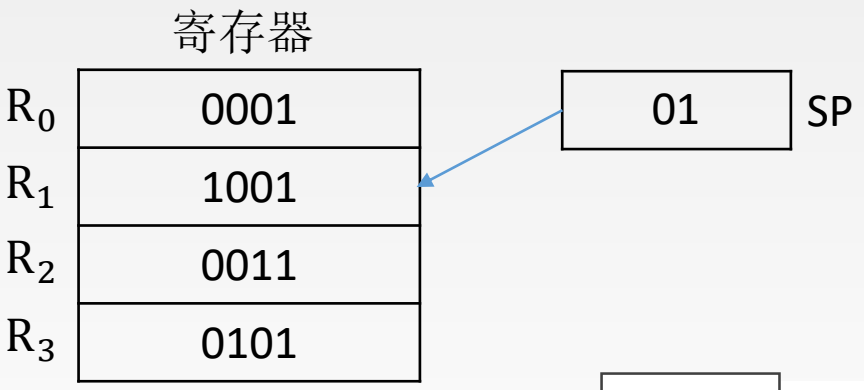
(SP)+1 → SP



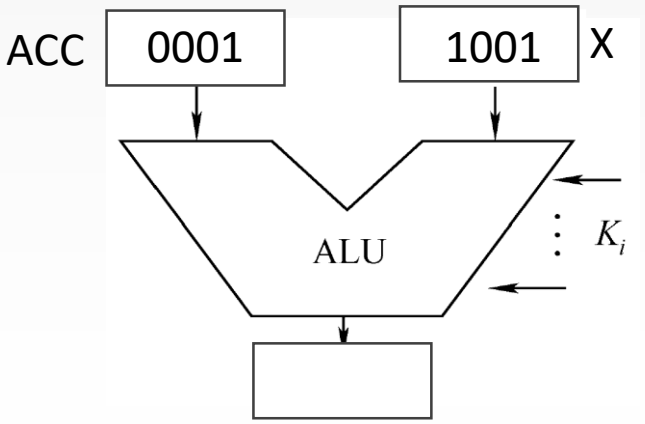
堆栈寻址

堆栈寻址：操作数存放在堆栈中，隐含使用堆栈指针(SP)作为操作数地址。

堆栈是存储器（或专用寄存器组）中一块特定的按“后进先出（LIFO）”原则管理的存储区，该存储区中被读/写单元的地址是用一个特定的寄存器给出的，该寄存器称为堆栈指针（SP）。



假设SP指向栈顶元素，R_0为栈顶



完成一次加法

记栈顶单元为Msp

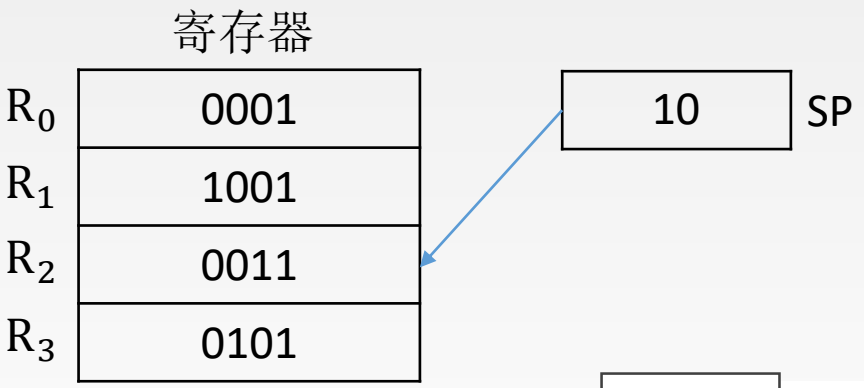
POP (Msp) → ACC
(SP)+1 → SP

POP (Msp) → X
(SP)+1 → SP

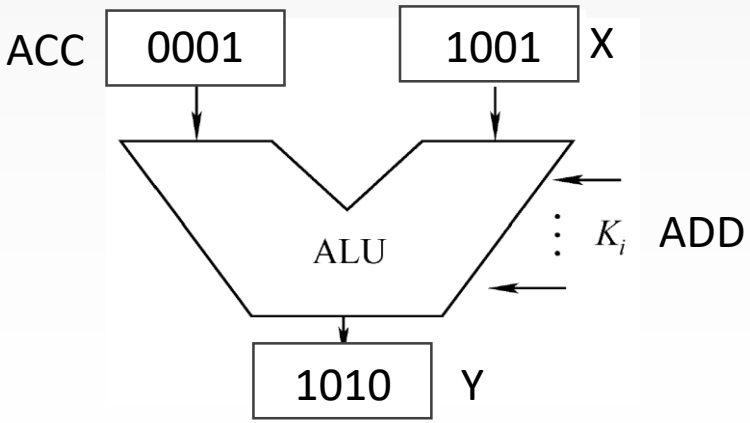
堆栈寻址

堆栈寻址：操作数存放在堆栈中，隐含使用堆栈指针(SP)作为操作数地址。

堆栈是存储器（或专用寄存器组）中一块特定的按“后进先出（LIFO）”原则管理的存储区，该存储区中被读/写单元的地址是用一个特定的寄存器给出的，该寄存器称为堆栈指针（SP）。



假设SP指向栈顶元素，R_0为栈顶



完成一次加法

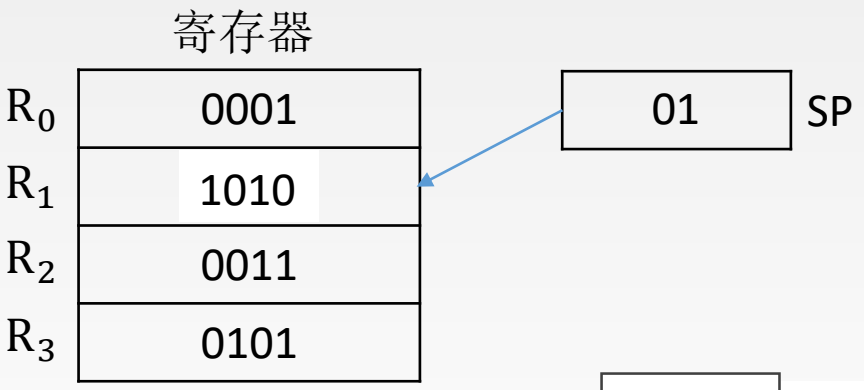
记栈顶单元为M_{sp}

POP	$(M_{sp}) \rightarrow ACC$
	$(SP)+1 \rightarrow SP$
POP	$(M_{sp}) \rightarrow X$
	$(SP)+1 \rightarrow SP$
ADD	$(ACC)+(X) \rightarrow Y$
PUSH	$(SP)-1 \rightarrow SP$

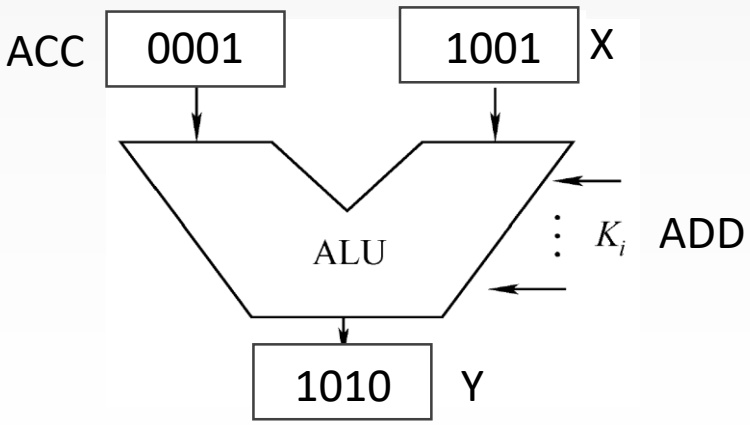
堆栈寻址

堆栈寻址：操作数存放在堆栈中，隐含使用堆栈指针(SP)作为操作数地址。

堆栈是存储器（或专用寄存器组）中一块特定的按“后进先出（LIFO）”原则管理的存储区，该存储区中被读/写单元的地址是用一个特定的寄存器给出的，该寄存器称为堆栈指针（SP）。



假设SP指向栈顶元素，R_0为栈顶



出栈：(SP)+1 → SP
(Msp) → ACC
入栈：(Y) → Msp
(SP)-1 → SP

出栈：(SP)-1 → SP
(Msp) → ACC
入栈：(Y) → Msp
(SP)+1 → SP

完成一次加法

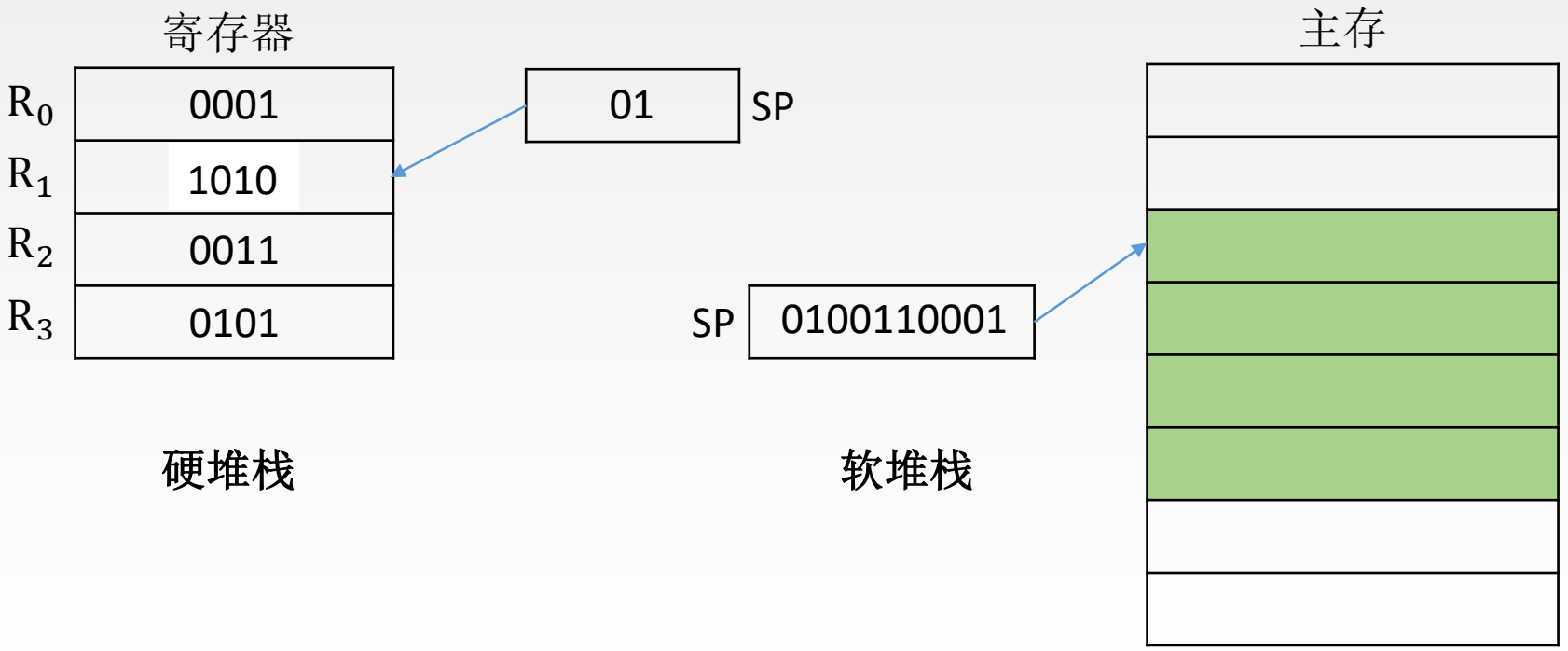
记栈顶单元为Msp

POP	出栈	(Msp) → ACC (SP)+1 → SP
POP		(Msp) → X (SP)+1 → SP
ADD		(ACC)+(X) → Y
PUSH	入栈	(SP)-1 → SP　　(Y) → Msp

堆栈寻址

堆栈寻址：操作数存放在堆栈中，隐含使用堆栈指针(SP)作为操作数地址。

堆栈是存储器（或专用寄存器组）中一块特定的按“后进先出（LIFO）”原则管理的存储区，该存储区中被读/写单元的地址是用一个特定的寄存器给出的，该寄存器称为堆栈指针（SP）。



本节回顾

寻 址 方 式	有 效 地 址	访 存 次 数(指令执行期间)
隐含寻址	程序指定	0
立即寻址	A即是操作数	0
直接寻址	EA=A	1
一次间接寻址	EA=(A)	2
寄存器寻址	EA=R_i	0
寄存器间接一次寻址	EA=(R_i)	1
转移指令　相对寻址	EA=(PC)+A	1
多道程序　基址寻址	EA=(BR)+A	1
循环程序　变址寻址　数组问题	EA=(IX)+A	1

偏移寻址

本节内容

指令系统

CISC和RISC

本章总览

CISC和RISC

CISC： Complex Instruction Set Computer

设计思路：一条指令完成一个复杂的基本功能。

代表：x86架构，主要用于笔记本、台式机等

RISC： Reduced Instruction Set Computer

设计思路：一条指令完成一个基本“动作”；
多条指令组合完成一个复杂的基本功能。

代表：ARM架构，主要用于手机、平板等

80-20规律：典型程序中 80% 的语句仅仅使用处理机中 20% 的指令

比如设计一套能输出单词的指令集：

CISC的思路：每个单词的输出由一条指令完成

一条指令可以由一个专门的电路完成：
17万个单词=17万个电路

→ 采用“存储程序”的设计思想，由一个比较通用的电路配合存储部件完成一条指令

RISC的思路：每个字母的输出由一条指令完成，
多条指令组合完成一个单词

26个字母=26个电路

“并行”、“流水线”

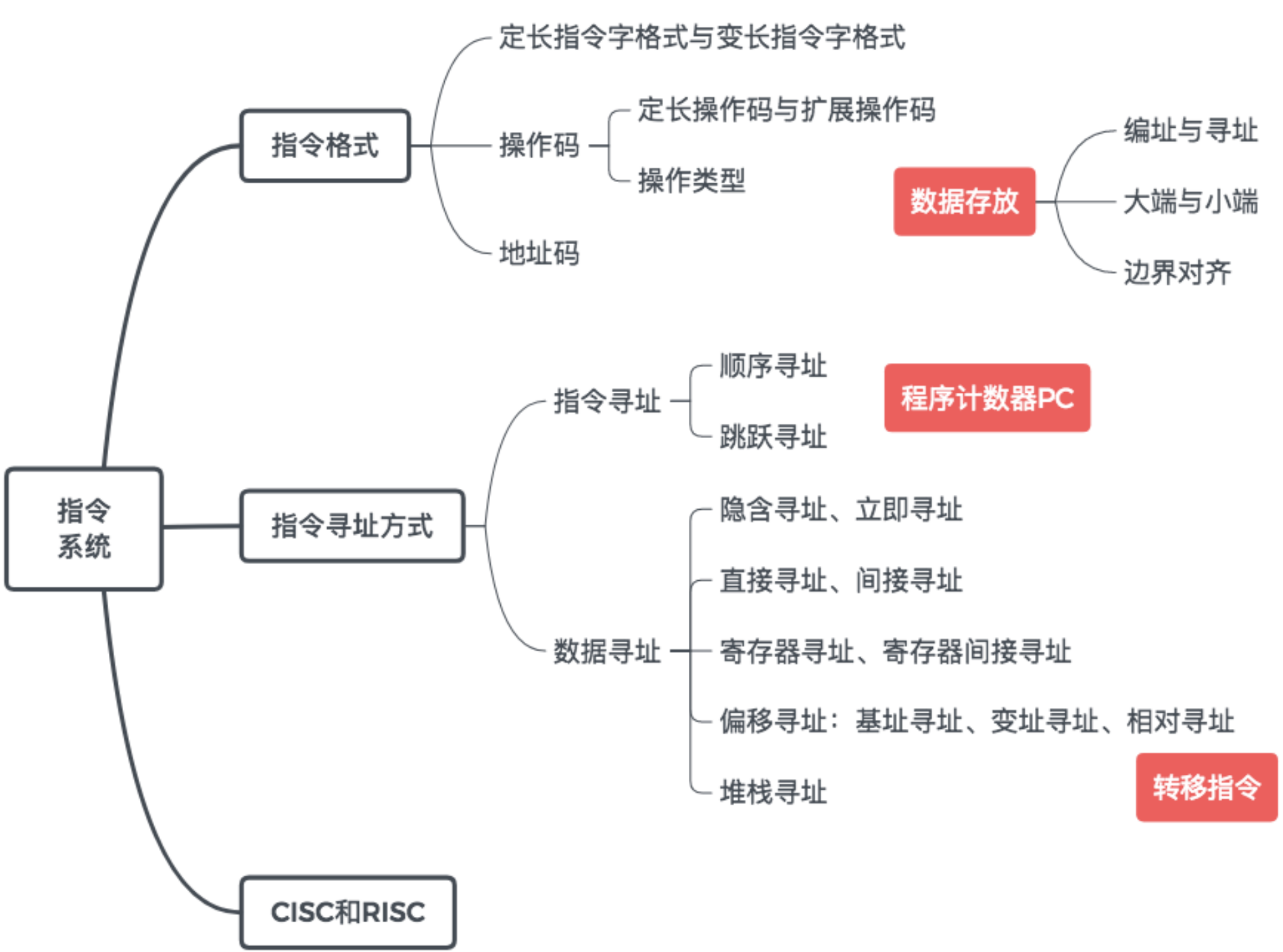
CISC和RISC

类别 / 对比项目	CISC	RISC
指令系统	复杂，庞大	简单，精简
指令数目	一般大于200条	一般小于100条
指令字长	不固定	定长
可访存指令	不加限制	只有Load/Store指令
各种指令执行时间	相差较大	绝大多数在一个周期内完成
各种指令使用频度	相差很大	都比较常用
通用寄存器数量	较少	多
目标代码	难以用优化编译生成高效的目标代码程序	采用优化的编译程序，生成代码较为高效
控制方式	绝大多数为微程序控制	绝大多数为组合逻辑控制
指令流水线	可以通过一定方式实现	必须实现

本节内容

指令系统

本章小结


指令系统
指令格式
定长指令字格式与变长指令字格式
操作码
定长操作码与扩展操作码
操作类型
地址码
数据存放
编址与寻址
大端与小端
边界对齐
指令寻址方式
指令寻址
顺序寻址
跳跃寻址
程序计数器PC
数据寻址
隐含寻址、立即寻址
直接寻址、间接寻址
寄存器寻址、寄存器间接寻址
偏移寻址：基址寻址、变址寻址、相对寻址
堆栈寻址
转移指令
CISC和RISC