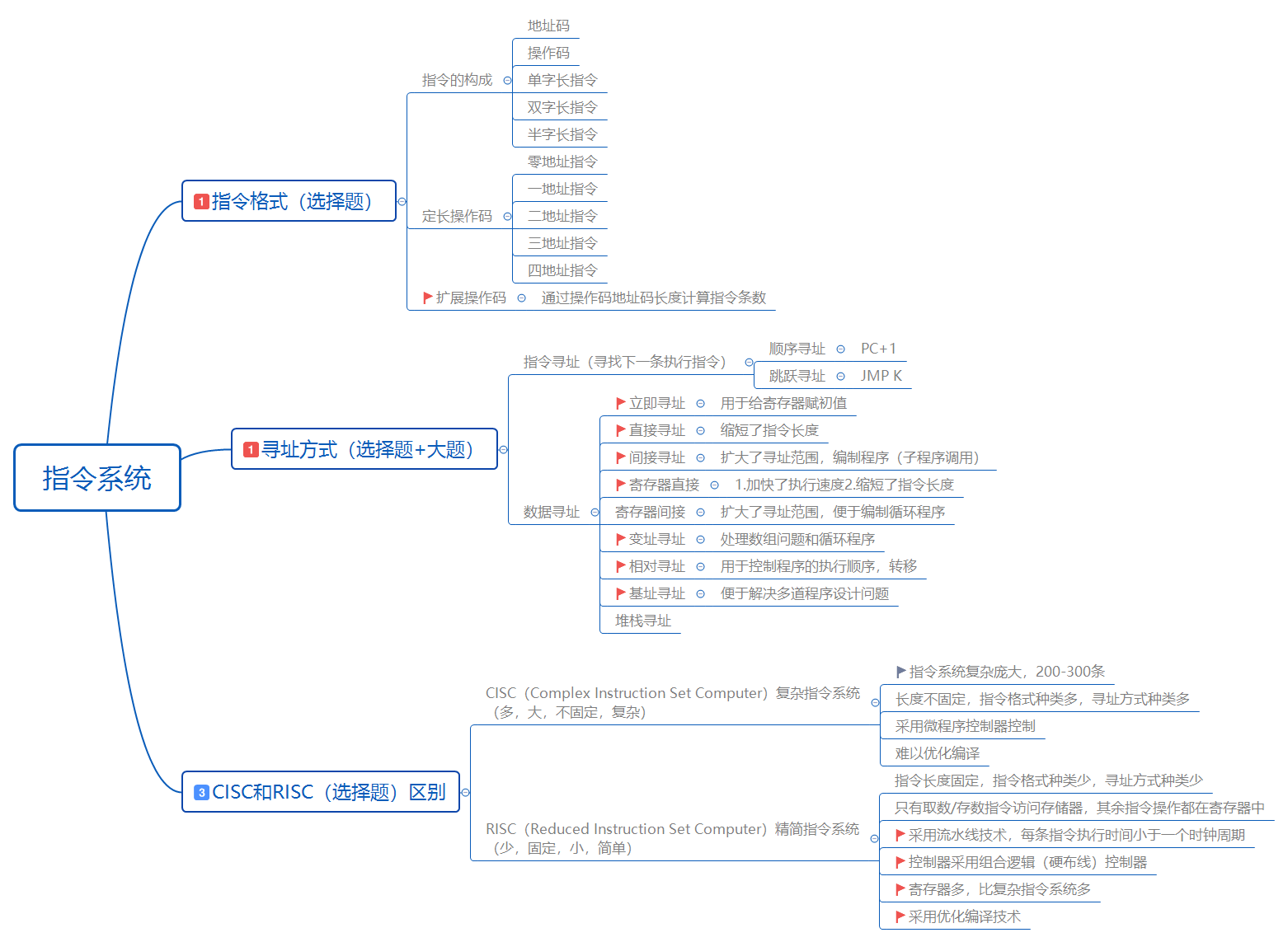
# 第四章

1. 知识体系



1. 本章重点
2. 各种寻址方式的原理和应用（重点）
3. 定长操作码和扩展操作码（概念+扩展操作码理解）
4. CISC和RISC的特点和区别（概念）
5. 几个问题：

**1)什么是指令字长，机器字长，存储字长?**

**2)半字，字，字节（byte），位（bit） 的关系**

字，半字，字节 大小是根据不同的操作系统来说的：

**32位系统**  
字 —>32bit=4byte =4B=4字节  
半字 —>16bit=2B  
字节 —>8bit（1B）

**64位系统**  
字 —>64bit=8byte=8B=8字节  
半字 —>32bit  
字节 —>8bit

1. **按字节遍址，按字编址**

**按字节1B编址**

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** | **3** | **4** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**按字编址：32系统 机器字长为多少位 1字=4字节**

|  |
| --- |
| **1** |
| **2** |
| **3** |
| **4** |

1. 知识点讲解

## 4.1 指令格式

**指令：**我们每次对计算机进行一系列的操作时候，都会将这些操作映射为计算机底层一个个基本的指令操作，这些指令操作是计算机执行最基本的单位比如说++操作，add操作，都是映射到计算机底层的基本操作来执行的

**操作码**：标识我们所要实现的功能

**地址码**：1.指出操作数的地址2.操作后结果放到哪里3.指令结束后该怎么办，寻找下一条指令

**指令系统（指令集）**：一台计算机所能执行的全部指令的集合。

**单字长指令**：指令长度等于存储字长。

**双字长指令**：指令长度等于两个存储字长。

**半字长指令**：指令长度等于半个存储字长。

1. **定长指令**

**优点**：设计简单，指令译码速度和执行速度快

**缺点**：指令数量太多时，地址码位数就会严重不足

**适用**：字长较长的计算机系统，32位或64位

|  |  |  |
| --- | --- | --- |
| OP | A1 | A2 |

**1+2**

**四地址，(**A1)OP(A2)->A3 A4存放下一条指令 访存四次

**三地址**，(A1)OP(A2)->A3 访存四次

**二地址**，(A1)OP(A2)->A2 访存四次

**一地址**，op(A1)->A1 自增，取反,访存三次，也可存放在**通用寄存器**中，访存两次

**零地址指令**： ①关机指令，return；访存一次

②操作数存放在堆栈中(堆栈寻址)

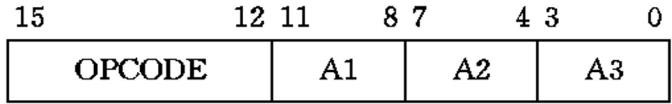
1. **不定长指令**

即操作码的长度可变，且分散地放在指令字的不同字段中。

**优点**：在指令字长较短的情况下，既能表示出更多的指令条数，又能满足地址码位数的要求；

**缺点**：操作码长度不固定将增加指令译码和分析的难度，使控制器的设计复杂化，

**适用**：字长较短的计算机系统，16位或16位下

若都是三地址指令，有 16 条。30条指令

0000XXX--1110XXXX,15 11110000--1111 1110 15位 1111 1111 0000-1111 1111 1111

1111 1111 1111 0000 ----1111 1111 1111 1111 45+16=61

指令有80条

0000--1000,（ 1100 1110 1111） (1100 0000---1100 1111)(1110 0000---1110 1111)(1111 0000---1111 1111)

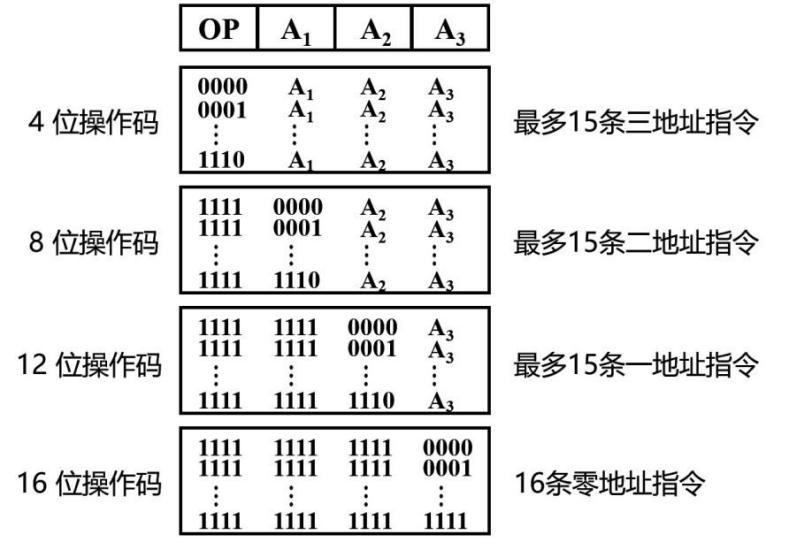
1. 15 条三地址指令的操作码由 4 位基本操作码从 0000～1110 给出，剩下一个码点（组合）1111 用于把操作码扩展到 A1，即 4 位扩展到 8 位；
2. 15 条二地址指令的操作码由 8 位操作码从 11110000～11111111 给出，剩下一个码点 11111111 用于把操作码扩展到 A2，即从 8 位扩展到 12 位；

3) 15 条一地址指令的操作码由 12 位操作码从 111111110000～111111111110 给出，

剩下一个码点 111111111111 用于把操作码扩展到 A3，即从 12 位扩展到 16 位；

4) 16 条零地址指令的操作码由 16 位操作码从 1111111111110000～

1111111111111111 给出



【例题1】某计算机按字节编址，指令字长固定且只有两种指令格式，其中三地址指令29条，二地址指令107条，每个地址字段为6位，则指令字长至少应该是**24位**

操作码字段：5位32条107/3 6位

## 4.2 寻址方式

**寻址方式**：寻找指令或者操作数的有效地址

**有效地址**: 操作数的真实地址称为有效地址， 记作 EA= (500 操作数)

指令寻址：寻找下一条执行指令

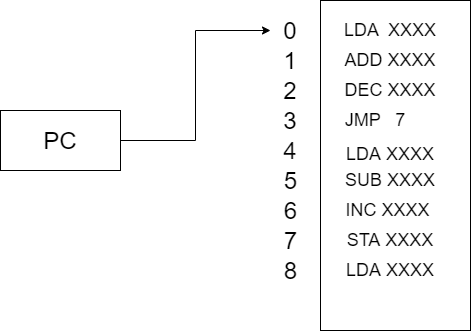
寻址方式

数据寻址:寻找地址码（操作数的地址）

## 一、指令寻址

**顺序** ( PC )+ 1→PC（指令字长=存储字长 指令字长=2倍存储字长）

**跳跃** 由转移指令指出



## **二、数据寻址**

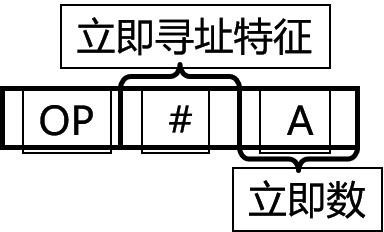
|  |  |  |
| --- | --- | --- |
| 操作码 | 寻址特征0001 | 形式地址 A |

**形式地址A** 指令字中的地址

**有效地址EA** 操作数的真实地址 EA=500 操作数

## 立即寻址

形式地址 A 就是操作数



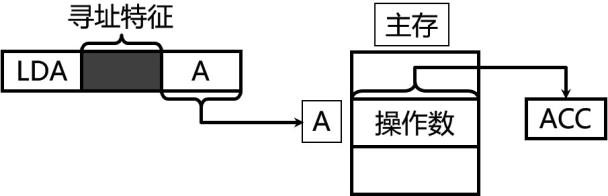
**优点**：简单，执行阶段不用访问存储器

**缺点**：A中存放操作数，指令的字长限制了数的大小，存放的补码，可正可负，

**应用场景**：for循环的循环次数，程序，数组的首地址，适用于对寄存器，或者内存单元赋初值

## 直接寻址

EA = A 有效地址由形式地址直接给出，

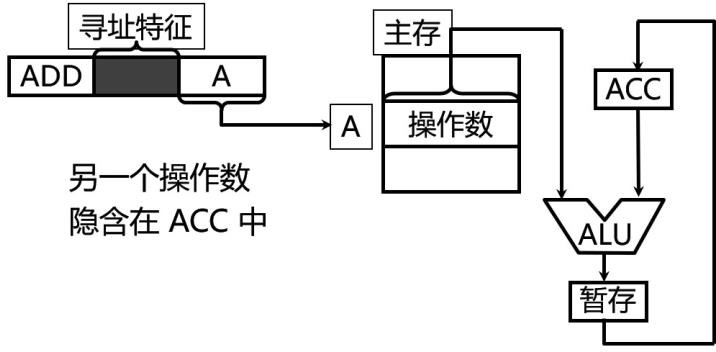


执行阶段访问一次存储器

**优点**：简单；

**缺点**：寻址范围比较小。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（必须修改 A）

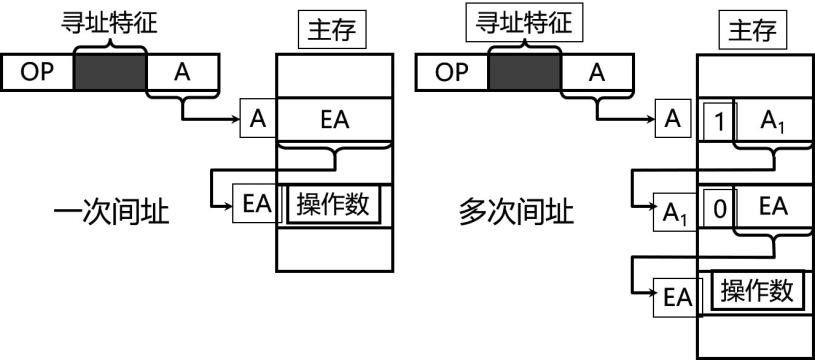
## 隐含寻址（了解）



一地址指令的算术运算，其中的一个操作数是隐含在ACC中

## 间接寻址

拿到形式地址A ——> 去主存中根据形式地址找到有效地址EA ——>根据EA 再次访问主存得到操作数

EA =（A）有效地址由形式地址间接提供，执行指令阶段 N+1 次访存

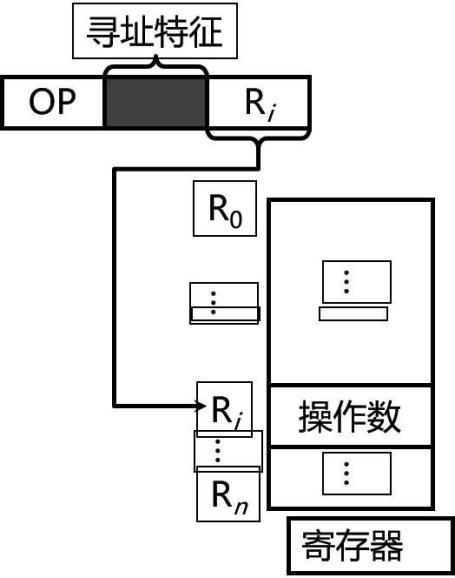
**优点**：可扩大寻址范围,便于编制程序（用于子程序调用返回）

**缺点**：多次访存

## 寄存器寻址

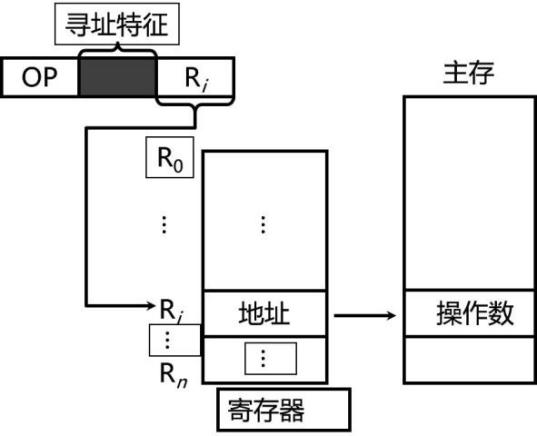
**在直接寻址中 ，地址码给出的是有效地址EA，在寄存器寻址中，地址码A给出的是寄存器的编号，通过寄存器编号去寻找操作数**

**EA = R*i*** 有效地址即为寄存器编号

执行阶段不访存，只访问寄存器，执行速度快，寄存器个数有限，可缩短指令字长

优点：1.加快了执行速度2.缩短了指令长度

## 寄存器间接寻址

EA = ( R*i* ) 有效地址在寄存器中 操作数在存储器中，执行阶段访存（一次）

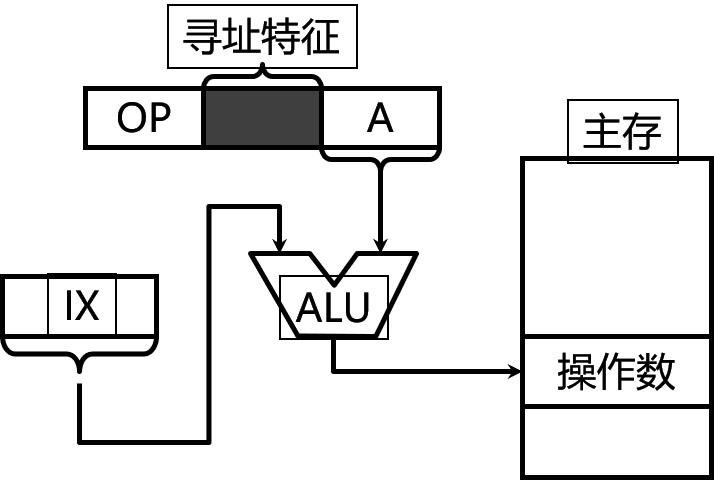
优点：扩大寻址范围（相对寄存器寻址来说）

缺点：需要访问主存（相对寄存器寻址来说）

## 基址寻址

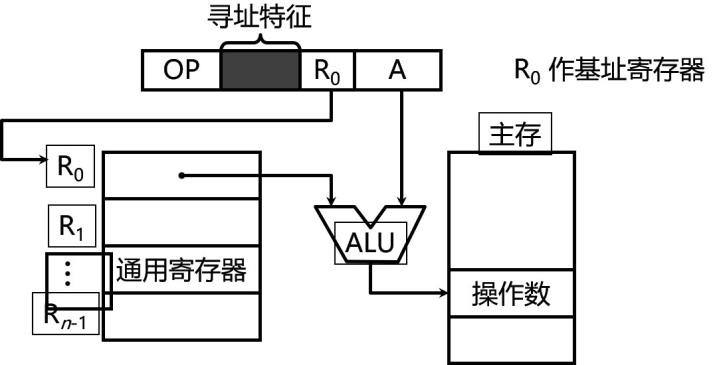
EA = ( BR ) + A BR 为基址寄存器

1. **采用专用寄存器作基址寄存器**



BR 内容由操作系统确定

**(2)采用通用寄存器作基址寄存器**

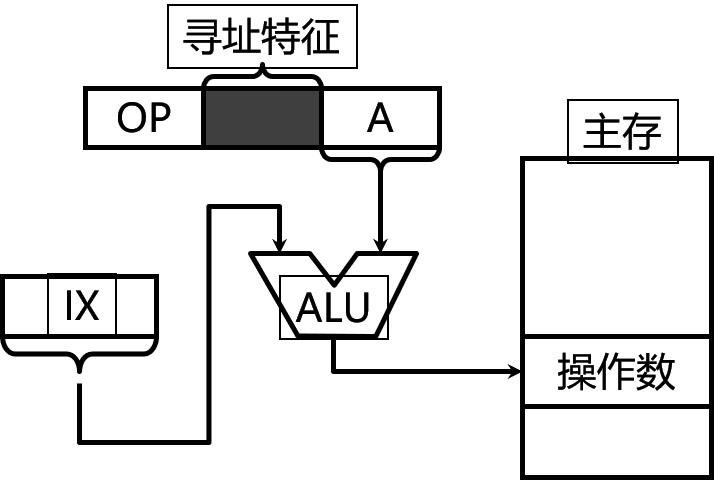
由用户指定哪个通用寄存器作为基址寄存器基址寄存器的内容由操作系统确定

在程序的执行过程中 R0内容不变，形式地址 A可变

**优点**：可扩大寻址范围，因为基址寄存器的位数可以大于形式地址位数，在程序的执行过程中 BR 内容不变，形式地址 A可变，有利于多道程序设计问题

## **变址寻址**

EA = ( IX ) +A IX 为变址寄存器（专用） 通用寄存器也可以作为变址寄存器



IX的内容由用户给定，在程序的执行过程中 IX内容可变，形式地址 A 不变

**优点**：扩大寻址范围，适合处理数组问题和循环程序

**总结：**

1）基址寻址和变址寻址的区别：两种方式有效地址的形成都是寄存器内容+偏移地址，

但在基址寻址中，程序员操作的是偏移地址，基址寄存器的内容由操作系统控制，在执行过

程中是动态调整的；而在变址寻址中，程序员操作的是变址寄存器，偏移地址是固定不变的。

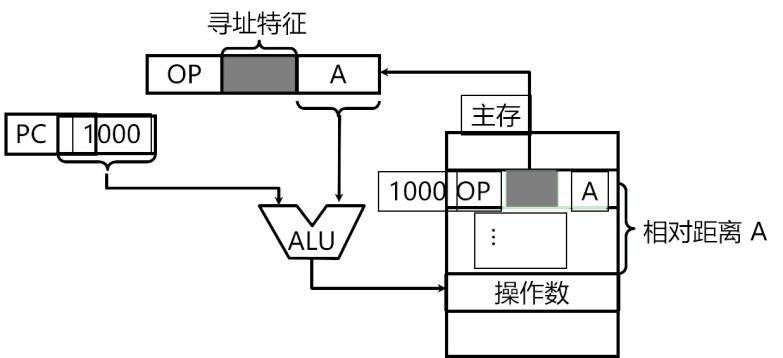
1. 先变址后间址，先间址后变址

## 相对寻址

EA = ( PC ) + A A 是相对于当前指令的位移量（可正可负，补码）

优点：用于转移指令JMP，控制指令的执行顺序

原理：基于程序局部性原理(时间局部性，空间局部性)，可以基于当前PC值，进行程序浮动，比如if判断语句，switch for 循环语句都是需要小范围的跳转的



## 堆栈寻址

**硬堆栈**：寄存器堆栈；**软堆栈**：指定一块内存

堆栈指示器

SP

|  |
| --- |
| R0 |
| R1 |
| R2 |
| R3 |

从堆栈中取数：(R0)-->ACC,(sp)+1-->sp;(R3)-->ACC (sp)-1-->sp ;

从寄存器中存数：(sp)-1-->sp (ACC)-->R0;(sp)+1-->sp (ACC)-->R3 ;

【练习1】根据操作数所在的位置，指出指令的寻址方式：

1）操作数在指令中，可能是什么寻址方式?（**立即寻址**）

2）操作数的有效地址在指令中，可能是什么寻址方式?（**直接寻址**）

3）操作数在寄存器中，可能是什么寻址方式？（寄存器寻址）

4）操作数有效地址在寄存器中，可能是什么寻址方式？（寄存器间接寻址）

5）操作数在存储器中，可能是什么寻址方式?（直接，间接，寄存器间接，基址，变址，间址，堆栈）

6）操作数有效地址在存储器中，可能是什么寻址方式？（间接）

7）操作数有效地址为某一奇存器中的内容与位移量之和，可能是什么寻址方式?（基址，变址，间址）

## 4.3 CISC和RISC

## 4.4习题

1.某机器字长为32位，存储器按半字2B编址，每取出一条指令后PC的值自动+2，说明

其指令长度是（B）。指令长度4B=2存储字长2B 1B=8bit

A.16位

B.32位

C.128位

D.256位

3.一条双字长的取数指令（LDA）存于存储器的200和201单元，其中第一个字为操作码

OP和寻址特征M，第二个字为形式地址A.假设PC当前值为200，变址寄存器IX的内容为100.基址寄存器的内容为200，存储器相关单元的内容如下表所示：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 201 | 300 | 400 | 401 | 500 | 501 | 502 | 700 |
| 内容 | 300 | 400 | 700 | 501 | 600 | 700 | 900 | 401 |

下表的各列分别为寻址方式、该寻址方式下的有效地址以及取数指令执行结束后累加器AC

的内容，试补全下表：

|  |  |  |
| --- | --- | --- |
| **寻址方式** | **有效地址EA** | **累加寄存器ACC** |
| 立即寻址 | ----------------------- | 300 |
| 直接寻址 | 300 | 400 |
| 间接寻址 | 400 | 700 |
| 相对寻址(PC)+A | 200+300 | 600 |
| 变址寻址(IX)+A | 100+300 | 700 |
| 基址寻址(BR)+A | 500 | 600 |
| 先变址后间址(IX)+A-->A(400) | 700 | 401 |
| 先间址后变址 | 500 | 600 |

前面变址求的有效地址=后面间址的形式地址

先间址后变址：最初的形式地址A=300 去间址找到有效地址400 ，对于后面的间址，形式地址A1就为400 ，在变址100+400