

Lecture 12: 知识 - 寻址方式

指令寻址

概念： 什么是指令寻址

- 确定下一条需要指向的指令的地址；

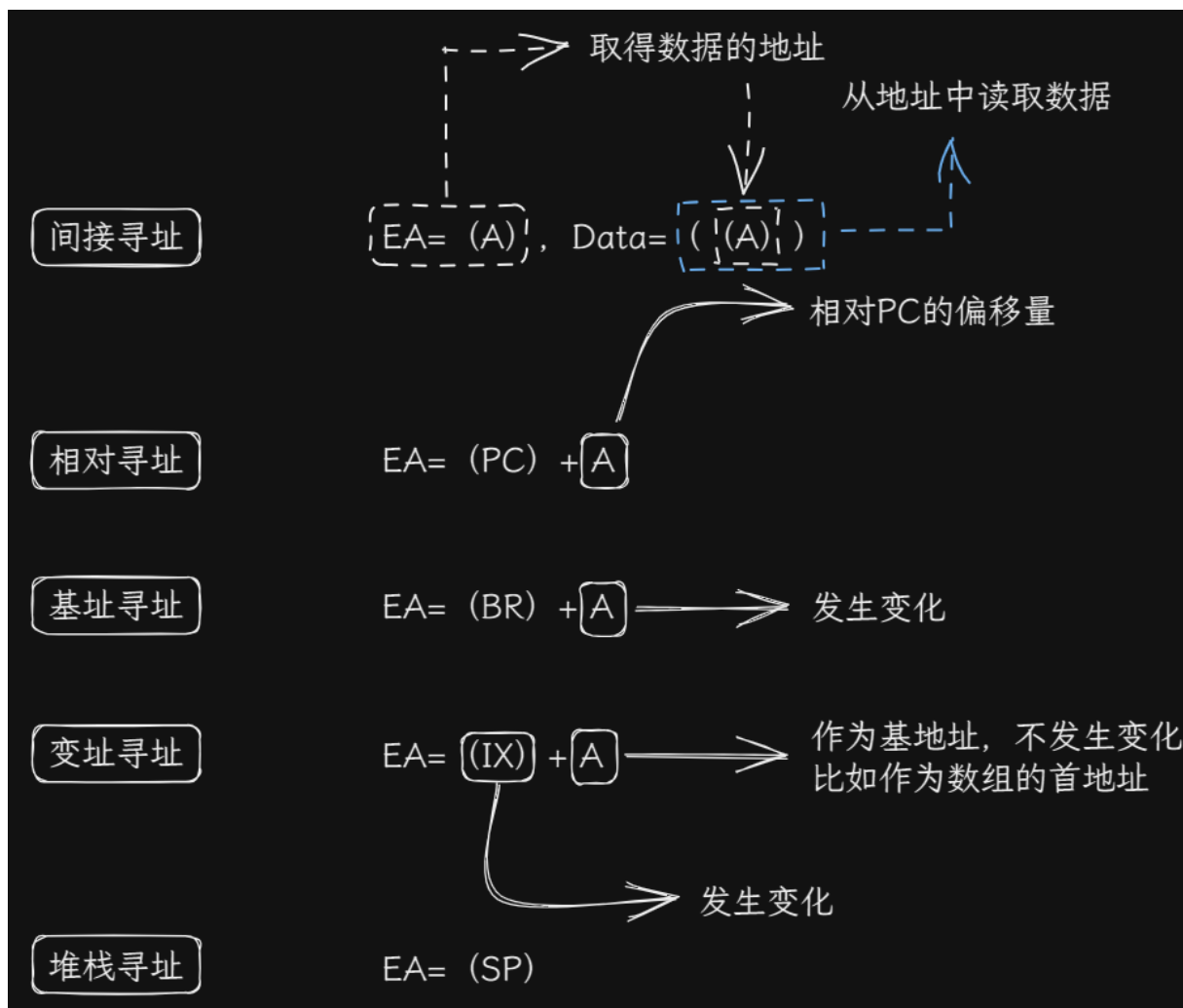
概念： PC 与下一条地址

- 概念：
 - **PC** 指令寄存器永远是指向下一条**打算指向的指令的地址**
- 顺序存储时：
 - 如果是**按照字**进行编制，则每次执行完之后， $PC=PC+1$ 就可以指向下一条地址；
 - 如果主存是**按照字节**编址（一字=两字节），则此时每个指令需要两个地址， $PC=PC+2$ ；
 - 如果主存是**按照变长指令**结构，则此时 **CPU** 只能一次先读入一个字，根据操作码判断这条指令的总字节数 **n**，按照指令的字节大小，修改 **PC** 的值，即 $PC=PC+n$ ；
- 跳跃寻址时：
 - 根据转移指令指出其寻址的地址；

数据寻址

概念： 什么是数据寻址

- 概念：
 - 确定本条指令的地址码所指明的真实地址；
- 总结：
 - 不同的寻址方式



• 分析:

- 相对寻址: 相对于 PC 进行偏移;
- 基址寻址: 基于 BR 寄存器的地址进行偏移, BR 不变;
- 变址寻址: IX 寄存器数值任意改变, A 不变 (数组首地址);

总结: 访存次数

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

概念: 数据寻址分类

• 分类:

- 立即寻址
- 直接寻址
- 隐含寻址

- 间接寻址
- 寄存器寻址
- 寄存器间接寻址
- 相对寻址
- 基址寻址
- 变址寻址
- 堆栈寻址

• 格式:



• 分析:

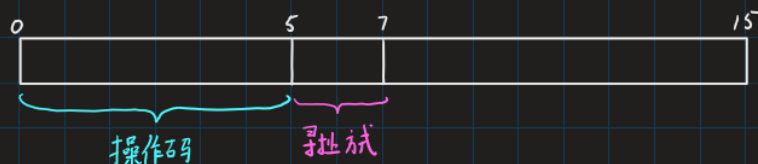
【2020 统考真题】某计算机采用 16 位定长指令字格式，操作码位数和寻址方式位数固定，指令系统有 48 条指令，支持直接、间接、立即、相对 4 种寻址方式。在单地址指令中，直接寻址方式的可寻址范围是 ()。

A. 0 ~ 255 B. 0 ~ 1023 C. -128 ~ 127 D. -512 ~ 511

■ 解析

由于当前指令系统共有 48 条指令，因此至少需要 $\lceil \log_2 48 \rceil = 6$ 位来表示操作码

由于当前有四种寻址方式，因此需要两位来表示寻址方式；



所以有 8 位可以用于表示直接寻址的地址。由于地址码必须为正数，因此寻址范围为： $0 \sim 2^8 - 1 = 0 \sim 255$

直接寻址

直接寻址

• 概念:

- 指令字中的形式地址 A 就是操作数的真实地址 EA，即 $EA=A$ ；

• 执行过程:

- 一条指令的执行 $\rightarrow$ 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:

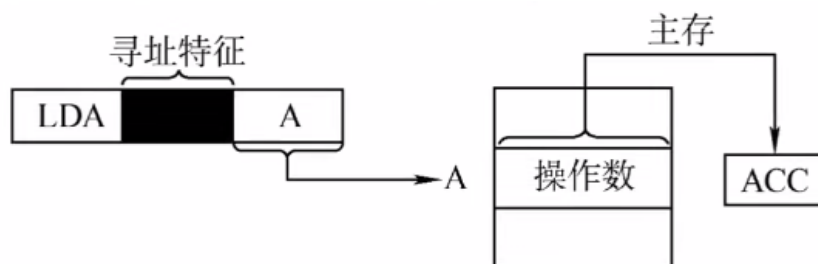
• 优点:

- 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；

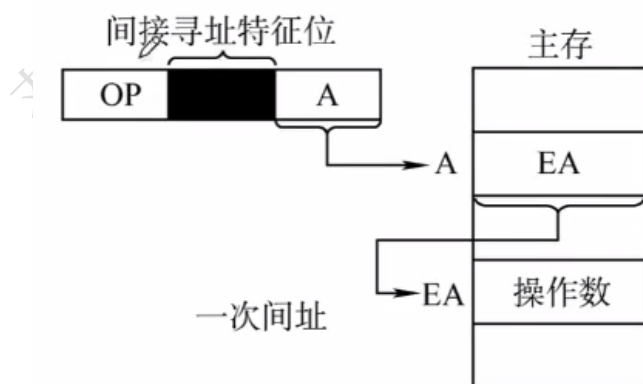
• 图示：



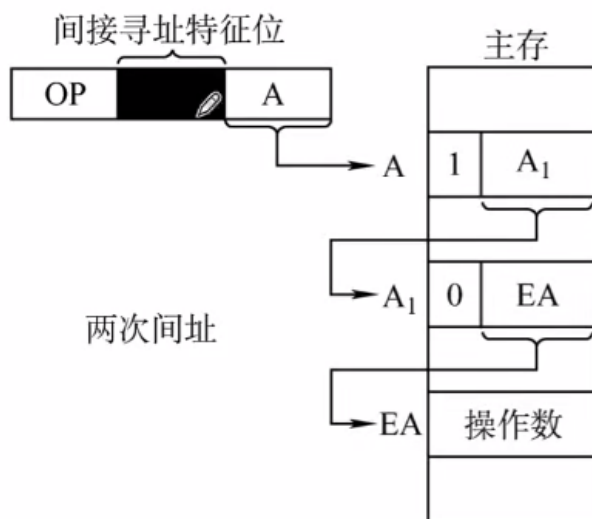
间接寻址

概念： 间接寻址

- 概念：
 - 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 $EA = (A)$ ；
- 过程分析：
 - 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；
- 特点：
 - 优点：
 - 1. 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)；
 - 2. 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)；
- 图示：
 - 一次间接寻址：



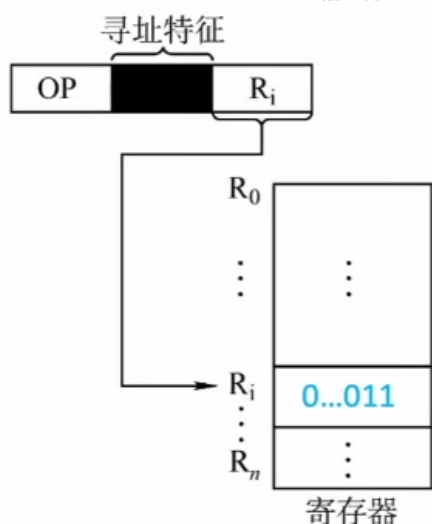
- 二次间接寻址：



寄存器寻址

概念：寄存器寻址

- 概念：
 - 在指令字中**直接给出操作数所在的寄存器编号**，即 $EA=R$ ，其操作数在由 R 所指的寄存器内；
- 特点：
 - 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。
- 图示：



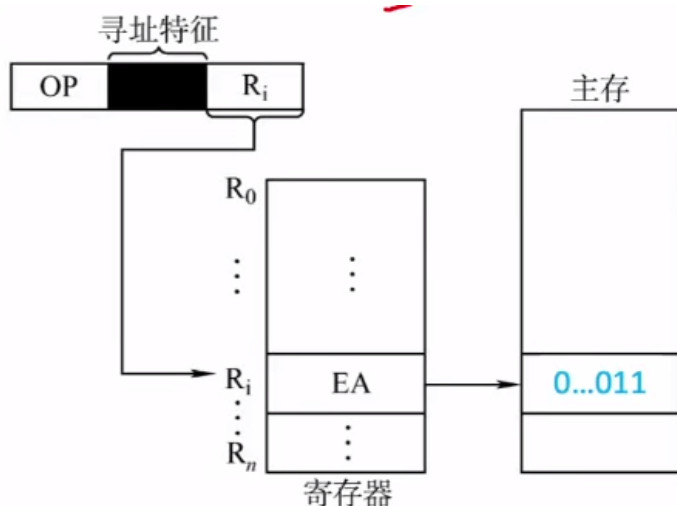
寄存器间接寻址

概念：寄存器间接寻址

- 概念：
 - 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 $EA = (R_i)$

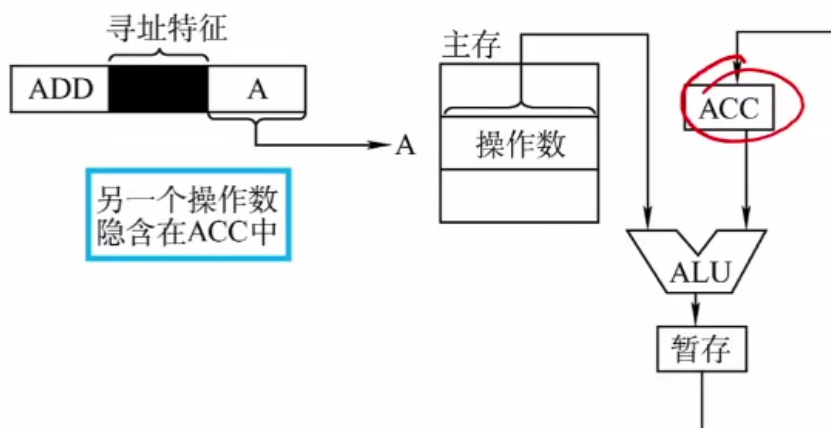
- 图示：



隐含寻址

概念： 隐含寻址

- 概念：
 - 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；
- 特点：
 - 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；
- 图示：



立即寻址

概念： 立即寻址

- 概念：
 - 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；
- 特点：

- 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 $-2^{n-1} \sim 2^{n-1} - 1$

偏移寻址

引入：三种偏移寻址

- 概念：
 - 相对寻址、基址寻址和变址寻址三种寻址方式非常类似，都将**某个寄存器的内容与一个形式地址相加**而生成操作数的有效地址，通常把这三种寻址方式称为偏移寻址；
- 分类：
 - 基址寻址：以程序的起始存放地址作为“起点”
 - 变址寻址：程序员自己决定从哪里作为“起点”
 - 相对寻址：以程序计数器 PC 所指地址作为“起点”；

总结：寻址方式

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

基址寻址

概念：基址寻址

- 基址寄存器：
 - BR
 - 基址寄存器是**面向操作系统的**，其内容由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；
- 基址寻址：

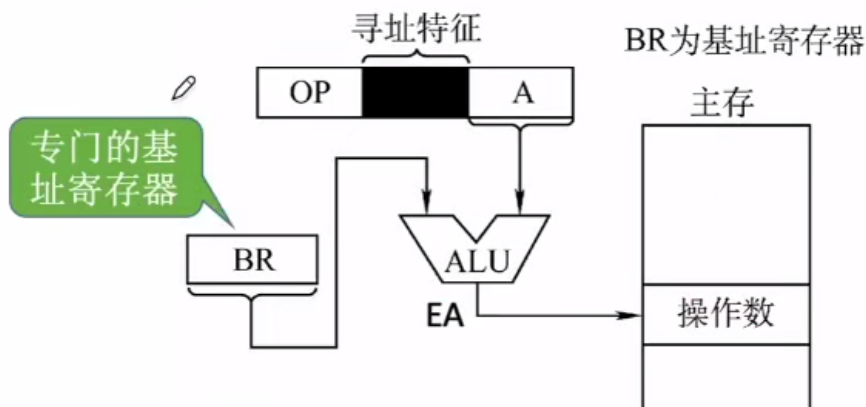
- 将 CPU 中基址寄存器 (BR) 的内容 (或者是某个指定的通用寄存器) 加上指令格式中的形式地址 A;
- 而形成操作数的有效地址, 即 $EA=(BR)+A$;

• 优点:

- 1. 便于程序“浮动”, 方便实现 **多道程序并发运行**;
- 2. 扩展程序的寻址范围;

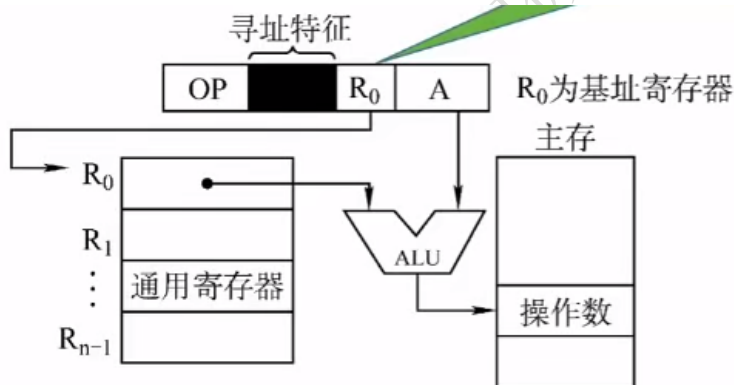
• 图示:

- 使用专用寄存器:



(a) 采用专用寄存器BR作为基址寄存器

- 使用通用寄存器:



变址寻址

概念: 变址寻址

- 变址寄存器:

- IX

- 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 可为变址寄存器 (专用), 也可用通用寄存器作为变址寄存器;

• 作用:

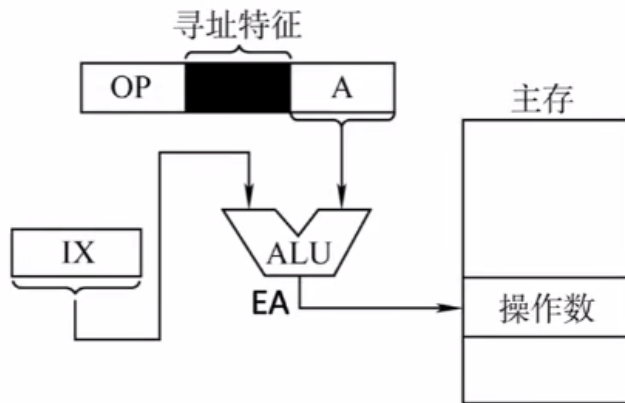
- 变址寄存器是**面向用户**的, 在程序执行过程中, 变址寄存器的内容可由用户改变, 形式地址 A 不变, 即 A 作为基地址;

• 优点:

- 变址寻址的优点是可扩大寻址范围 (变址寄存器的位数大于形式地址 A 的位数);

- 真题：2017年
- 在数组处理过程中，可设定形式地址 **A** 作为当前数组的首地址，通过不断改变变址寄存器 **IX** 的内容，便可很容易形成数组中任意一个数据的地址，特别适合编制循环程序；
- 偏移量（变址寄存器 **IX**）的位数足以表示整个存储空间；
- 由用户操控，适合编制循环程序；

• 图示：



• 分析：

【2013 统考真题】假设变址寄存器 R 的内容为 1000H，指令中的形式地址为 2000H；地址 1000H 中的内容为 2000H，地址 2000H 中的内容为 3000H，地址 3000H 中的内容为 4000H，则变址寻址方式下访问到的操作数是（ ）。

A. 1000H B. 2000H C. 3000H D. 4000H

★ 总结：分析寻址过程前，先确定当前提问的对象是操作数还是目标地址；

• 若是目标地址，则获得目标地址后不需要访问。若是操作数，则在获得形式地址EA后，还需要在EA所在位置进行访问、取得操作数；

■ 解析

变址寻址的形式地址： $EA = (IX) + A = \text{IX寄存器中的值} + \text{A地址的值}$

变址寻址的操作数 = (EA)

由上可知： $EA = (IX) + A = 1000H + 2000H = 3000H$

所以操作数 = (EA) = (3000H) = 4000H

分析：变址寻址与 **for** 循环语句

- 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；

分析：基址&变址寻址的复合

- 举例：先进行基址寻址，再进行变址寻址；
- 基址寻址： $EA = (BR) + A$
- 变址寻址： $EA = (IX) + A$
- 先基址后变址寻址： $EA = (IX) + ((BR) + A)$

分析：真题 - 2016年

- 题目:

【2016 统考真题】某指令格式如下所示。

OP	M	I	D
----	---	---	---

其中 M 为寻址方式, I 为变址寄存器编号, D 为形式地址。若采用先变址后间址的寻址方式, 则操作数的有效地址是 ()。

- A. $I + D$ B. $(I) + D$ C. $((I) + D)$ D. $((I)) + D$

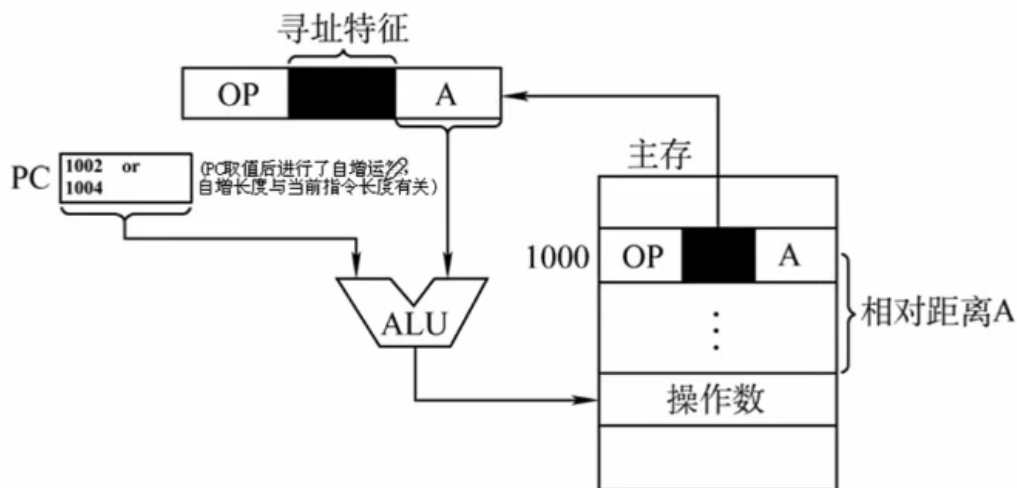
- 分析:

- 按步骤分析, 上一个步骤的结果为下一个步骤的来源;
- 第一步: 变址寻址
 - 由于变址寄存器中的值作为变址, 因此需要先取出其值: (I)
 - 将变址寄存器中取出的值加上形式地址 D , 得到变址寻址的结构: $(I) + D$
- 第二步: 间接寻址
 - 间接寻址的特点是: 其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。所以需要对第一步得到的形式地址再进行访问, 才可以得到目标地址;
 - 即目标地址 $EA = ((I) + D)$
- 补充:
 - 问题:
 - 请问采用先变址再间址的寻址方式时, 得到的操作数是?
 - 解答:
 - 操作数 = $(EA) = (((I) + D))$
 - 就是对取出的有效地址再进行一次访存;

相对寻址

概念: 相对寻址

- 概念:
 - 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 PC 所指地址的位移量, 可正可负, 补码表示;
 - A 是基于下一条指令的 PC 地址所计算的偏移量;
- 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 (一段代码在程序内部的浮动);
 - 相对寻址广泛应用于转移指令;
- 图示:



分析:

【2009 统考真题】某机器字长为 16 位，主存按字节编址，转移指令采用相对寻址，由 2 字节组成，第一字节为操作码字段，第二字节为相对位移量字段。假定取指令时，每取一字节 PC 自动加 1。若某转移指令所在主存地址为 2000H，相对位移量字段的内容为 06H，则该转移指令成功转移后的目标地址是 ()。

A. 2006H B. 2007H C. 2008H D. 2009H

解析

由于当前机器字长为 16 位 = 2B，所以每执行一条指令后需要自动加 2B；

• 第一步：取指令

• 因为转移指令的主存地址为 2000H，所以当 CPU 从 2000H 取出 jmp 指令后，此时 $PC = (PC) + 2B$ ，所以当前的 PC 值为 2002H

• 第二步：执行指令

• 相对寻址的结果为 $EA = (PC) + A$

• 由于 $(PC) = 2002H$ ， $A = 06H$ ，所以 jmp 需要跳转到 $2002H + 06H = 2008H$



★ 总结：涉及指令执行 + 相对寻址时，注意指令执行顺序是先取指令、再执行指令。因此在执行相对寻址之前，PC 的值已经完成了一次自加；

堆栈寻址

概念：硬堆栈

概念：

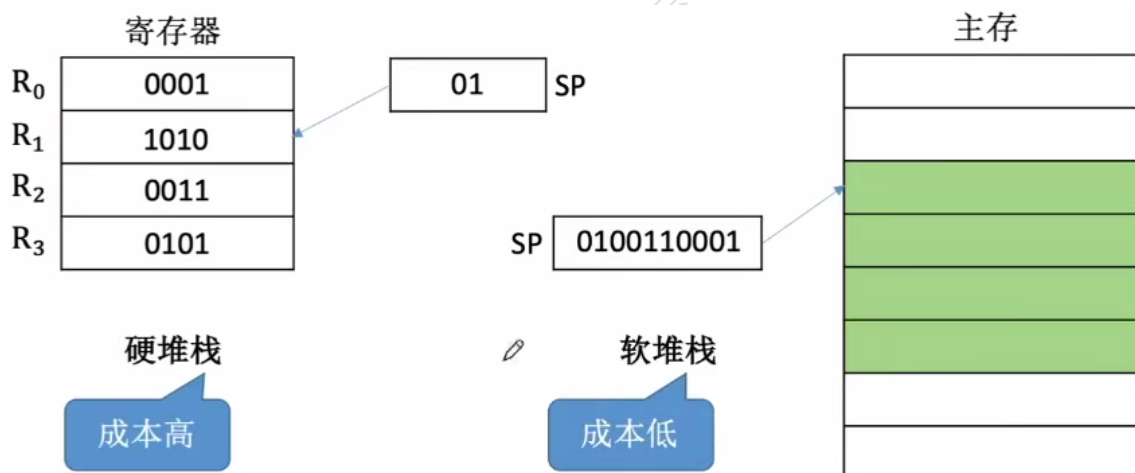
- 操作数存放在堆栈中，隐含使用堆栈指针 (SP) 作为操作数地址；
- 堆栈指针是专门的寄存器；
- 即其地址的指令中地址是隐含的；
- 硬堆栈全部在寄存器中实现；

堆栈指令：

- PUSH、PULL
- 实现：
 - 栈顶在小地址方向
 - 出栈：
 - 1. $(M_SP) \rightarrow ACC$
 - 2. $(SP) = (SP) + 1$
 - 入栈：
 - 1. $(SP) = (SP) - 1$
 - 2. $(Y) \rightarrow M_SP$

概念：软堆栈

- 概念：
 - 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）；
 - 软堆栈需要在主存的栈区实现；
- 图示：



总结

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
偏移寻址 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
堆栈寻址	入栈/出栈时EA的确定方式不同	硬堆栈不访存, 软堆栈访存1次

获取笔记持续更新可加微信: WandD222