第三章 指令系统

本章主要内容

课前思考：

1) 计算机程序如何构成？执行过程具体由哪些因素决定？

2) 什么是指令？怎么样表示一条指令？

3) 一个处理器需要多少指令？

本章主要讲述指令系统。包括指令系统的结构、指令的格式和寻址方式，指令系统的优化设计，包括指令操作码的优化设计和地址码的优化设计。介绍指令系统发展两个方向构成的计算机，即CISC计算机和RISC计算机。学习时重点掌握指令的功能表示、格式与寻址方式。

第三章 指令系统

- 3.1 引言
- 3.2 指令格式
- 3.3 寻址技术
- 3.4 典型的指令系统
- 3.5 指令系统的优化设计*

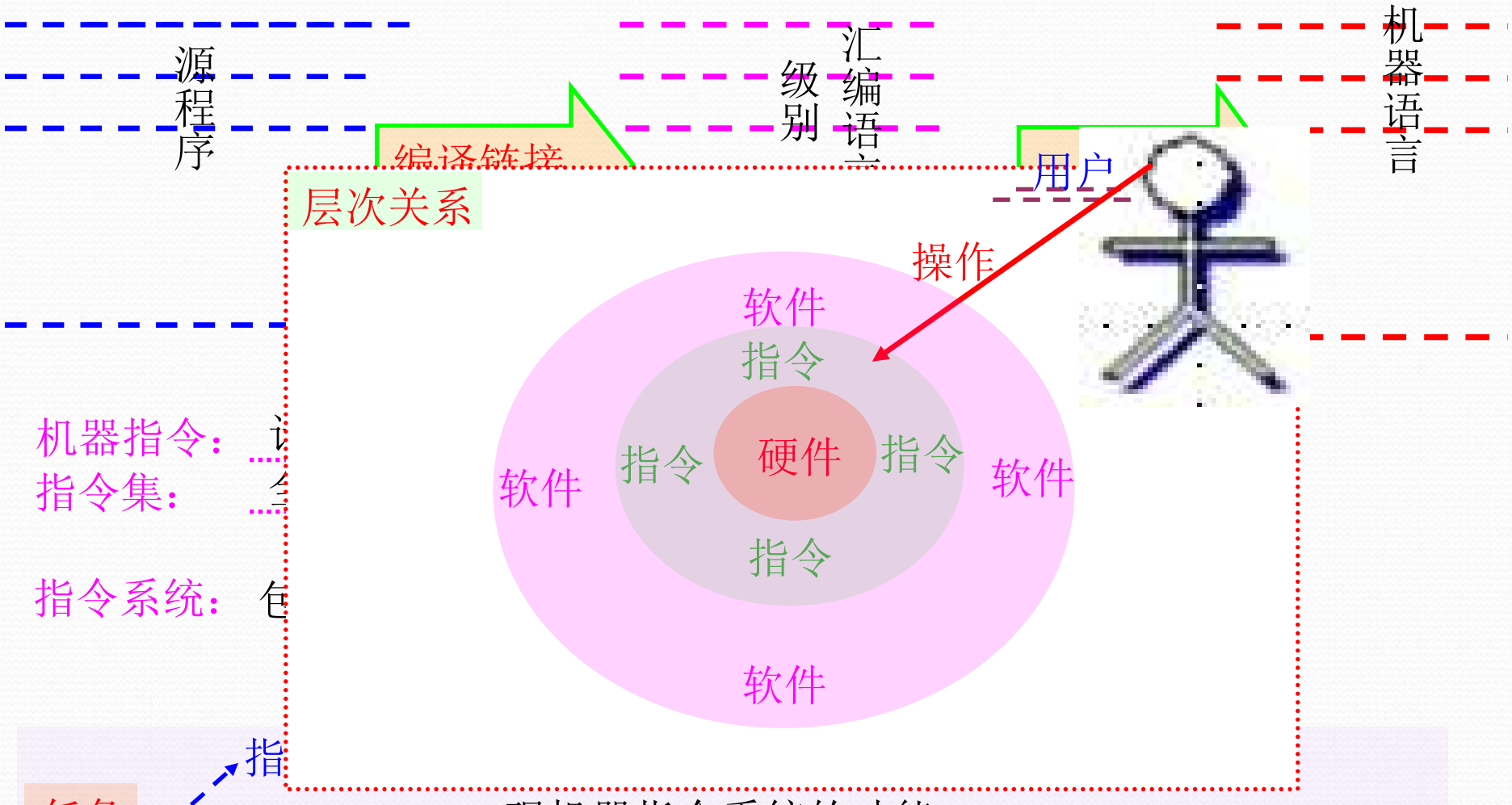
引言

传统的计算机指令系统设计时主要考虑以下几个因素：

（1）计算机面向的应用领域

（2）如何继承软件资产

概 述


源程序
汇编语言
级别
机器语言
编译链接
层次关系
用户
操作
软件
指令
硬件
指令
指令
软件
软件
指令
软件
机器指令：
指令集：
指令系统：
指
任务
现机器指令系统的功能。
指令使用者：使用指令来编制程序，完成预定任务

RISC与CISC

RISC 的产生和发展：

RISC（Reduced Instruction Set Computer）

CISC（Complex Instruction Set Computer）

80 — 20 规律　　—— RISC技术

- 典型程序中 80% 的语句仅仅使用处理机中 20% 的指令
- 执行频度高的简单指令，因复杂指令的存在，执行速度无法提高

? 能否用 20% 的简单指令组合不常用的 80% 的指令功能

RISC的主要特征

- 选用使用频度较高的一些简单指令，
  复杂指令的功能由简单指令来组合
- 指令 长度固定、指令格式种类少、寻址方式少
- 只有 LOAD / STORE 指令访存
- CPU 中有多个通用寄存器
- 采用流水技术，一个时钟周期内完成一条指令
- 采用组合逻辑实现控制器
- 采用优化的编译程序

CISC的主要特征

- 系统指令复杂庞大，各种指令使用频度相差大
- 指令长度不固定、指令格式种类多、寻址方式多
- 访存指令不受限制
- CPU 中设有专用寄存器
- 大多数指令需要多个时钟周期执行完毕
- 采用微程序控制器
- 难以用优化编译生成高效的目的代码

RISC与CISC比较

1. RISC更能充分利用 VLSI 芯片的面积

2. RISC 更能提高计算机运算速度

   指令数、指令格式、寻址方式少，
   通用寄存器多，采用组合逻辑，
   便于实现指令流水

3. RISC便于设计，可降低成本，提高可靠性

4. RISC有利于编译程序代码优化

5. RISC不易实现指令系统兼容

第三章 指令系统

- 3.1 引言
- 3.2 指令格式
- 3.3 寻址技术
- 3.4 典型的指令系统
- 3.5 指令系统的优化设计*

指令格式

指令的基本格式如下：

其中：

操作码指明了指令的操作性质及功能；

地址码则给出了操作对象的地址，也就是操作数的地址。

指令字长度

- 指令的长度是指一条指令中所包含的二进制代码的位数，它取决于操作码字段的长度、操作数地址的个数及长度
- 如果指令系统中所有指令的长度都是一样的，称为固定字长指令结构
- 如果各种指令的长度随指令的不同而有所不同，则成为变长指令字结构

操作码

- 每条指令都有不同于其它指令的操作码编码。
- 操作码占用的二进制位数一般取决于计算机指令系统的规模，实际上也就是与指令的条数有关。
- 一般来讲，对于共有m条指令的指令系统，指令的操作码字段为N位，有如下关系式成立：

$$N \geq \log_2 m$$

操作码分类

指令操作码的编码可以分为固定长度的定长编码和长度可变的变长编码两种。

(1) 定长编码(规整型)

操作码字段的长度和位置是固定的

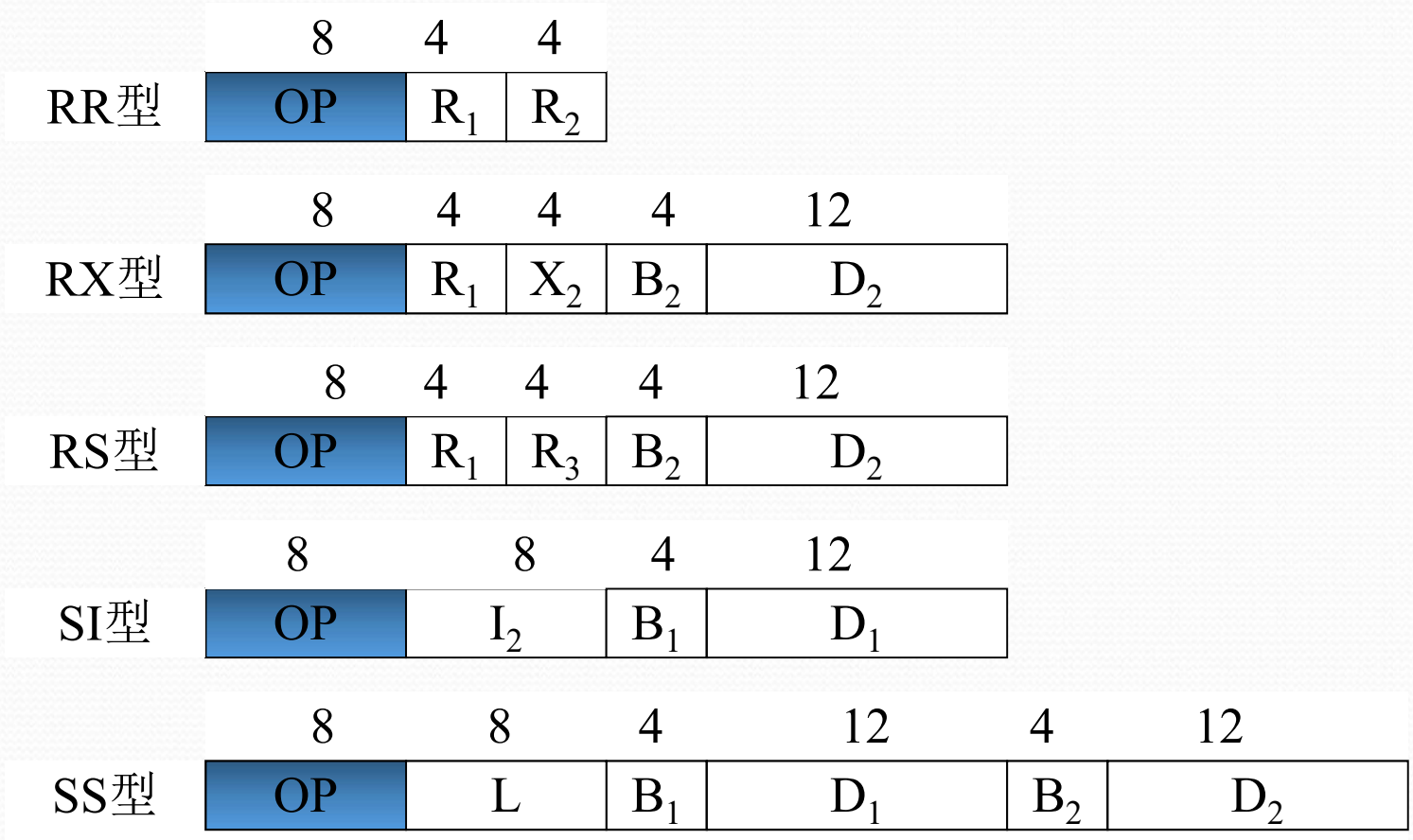
(2) 变长编码(非规整型)

操作码字段位数是不相同的

定长编码（规整型）

- 固定长度编码对于简化指令结构，减少指令译码时间是非常有利的，在字长较长的大、中型机及超级小型机上被广泛采用。IBM 370机采用的就是这种结构。
- IBM 370是一种32位机，机器字长32位。有三种结构的指令，分别是半字长指令(指令字长度16位)、单字长指令(指令字长度32位)和1.5倍字长指令(指令字长度48位)。不同类型的指令编码格式有所不同，但是操作码字段的长度都是固定的8位长。下图给出了这种指令系统的格式(图中用OP表示操作码字段)。

IBM 370机的指令系统



IBM 370机的指令系统格式

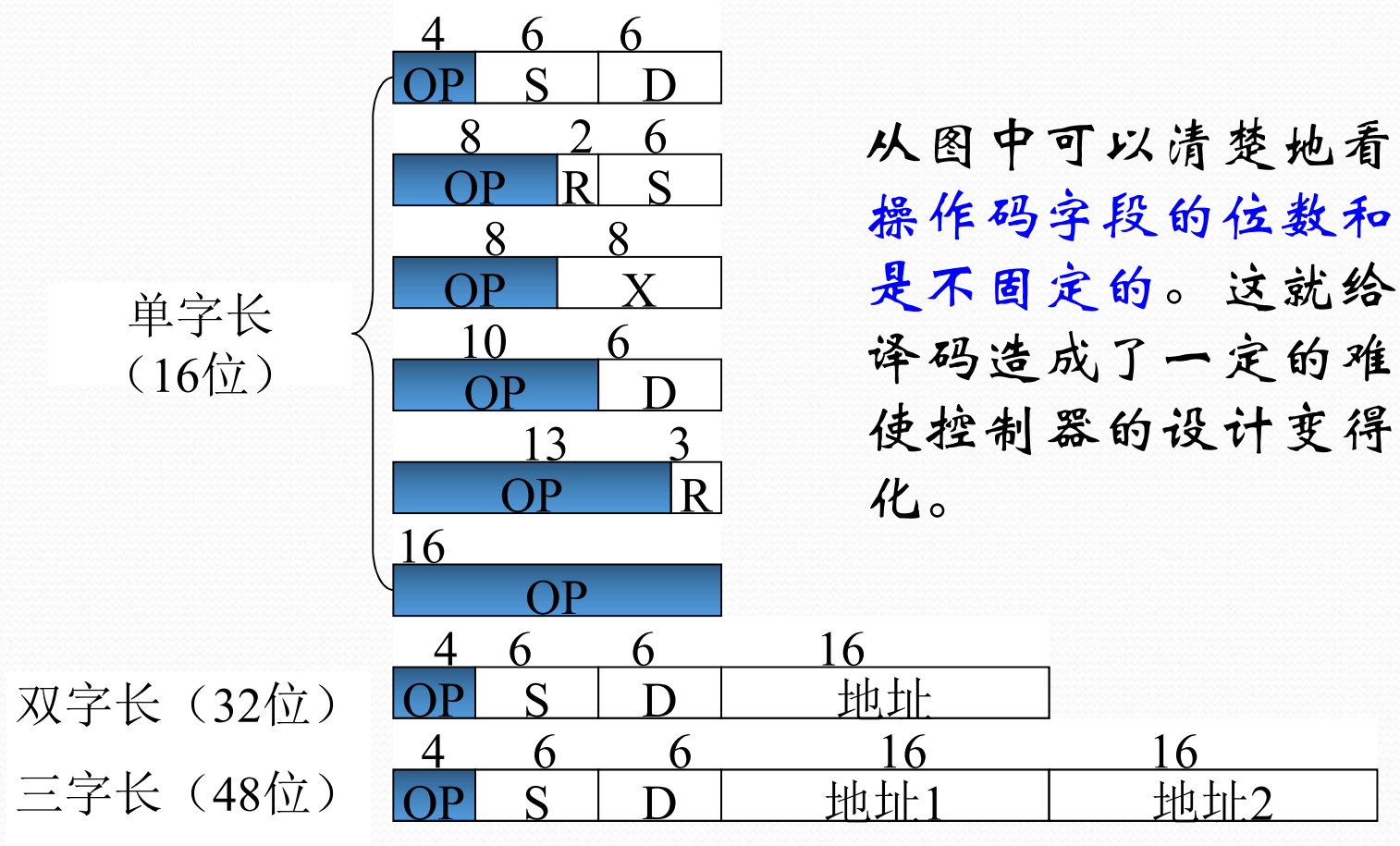
IBM 370机指令系统特点

从图中可以看出，不论哪种指令，操作码字段的长度都是固定的8位。8位操作码可以容纳256条不同的指令，实际上在IBM 370机中仅有183条指令，存在着一定的信息冗余，这种冗余的信息实际上是一种非法的操作码。

变长编码

- 变长编码的操作码字段位数是不相同的。
- 采用这种方式实际上是对指令系统进行优化的结果。这种方式可以有效地压缩指令中操作码字段的平均长度，在字长较短的小型、微型计算机系统中被广泛采用。如小型机PDP-11机，这种机型的字长是16位，它采用的就是操作码字段变长编码方式。
- PDP-11机的指令分为单字长、双字长、三字长指令三种，操作码字段占4～16位不等，可以遍及整个指令字长度范围。

变长编码



从图中可以清楚地看出，操作码字段的位数和位置是不固定的。这就给指令译码造成了一定的难度，使控制器的设计变得复杂化。

图 PDP-11机的指令系统格式

地址码

地址码给出操作对象的地址，也就是操作数的地址。

地址码一般用下列几种形式表示：

(1) 第一操作数地址，用A1表示；

(2) 第二操作数地址，用A2表示；

(3) 操作结果存放地址，用A3表示；

(4) 下一条将要执行的指令地址，用A4表示。

如果以上四项信息在指令中显式地给出，称为显地址指令；如果这些信息采用事先的约定形式，没有在指令中显式地给出，则称为隐地址指令。

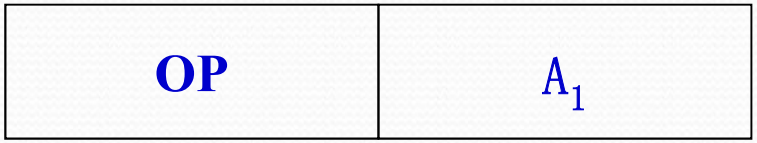
零地址指令

- 在指令各式中只有操作码字段，没有地址码字段。其格式为：

- 只有操作码字段的指令可能有两种情况，一种是本身就没有操作数，称为无操作数指令，如停机指令、空操作指令、等待指令等；另一种是操作数是按照某种约定隐含的，例如堆栈操作类指令。

一地址指令

- 在指令格式中只包含一个显地址字段，其指令格式为：



- 这种指令完成的功能可能有两种情况，一种是完成单操作数运算，如加1、减1等。由于这类指令仅需要一个操作数，所以这个地址既是操作数地址，也是存放结果的地址。指令完成的功能可以表示为：

$OP(A_1) \rightarrow A_1$

$(PC)+1 \rightarrow PC$（隐含完成）

- 其中，A_1表示地址，(A_1)表示存放于该地址单元的内容。

一地址指令

另一种情况是双操作数。那么，另一个操作数来自何方呢？来自事先的约定。另一个操作数虽然未在指令中显式地给出，但是按照事先的约定，另一个操作数必须存放在事先约定的专门的寄存器中，一般这个寄存器是累加器Acc（Accumulator）。这种指令的含义是：

$(A_{CC})OP(A_1)\rightarrow A_{CC}$

(PC)+1→PC(隐含完成)

如，Intel 80x86的指令系统中的乘法、除法指令。MUL AL,DIV BL等

二地址指令

二地址指令一般是运算类指令，指令中显式地给出参加运算的两个操作数。两个操作数中往往包含一个源操作数和一个目的操作数，运算结果存放在目的操作数中。指令格式为：

OP	A_1	A_2

指令的含义为：

$(A_1)OP(A_2)\rightarrow A_1$

$(PC)+1\rightarrow PC$（隐含完成）

三地址指令

指令中包含三个地址段，其中的两个地址段用来存放源操作数地址，第三个操作数用来存放目的操作数地址。三地址指令的格式如下：

OP	A_3	A_1	A_2

指令的含义：

$(A_1)OP(A_2)\rightarrow A_3$

$(PC)+1\rightarrow PC$（隐含完成）

四地址指令

四地址指令将指令中地址码的全部信息都显式地给出，四段地址信息中包含了两个源操作数地址，一个目的操作数地址和下一条将要执行的指令地址。

OP	A_1	A_2	A_3	A_4

指令的含义：

$(A_1)OP(A_2)\rightarrow A_3$

A_4=下一条将要执行的指令地址

可以看出，四地址指令会占用较长的指令字长度，虽然结构比较清楚，但实际很少使用。

在指令系统的设计过程中要考虑诸多因素，最终选择合适的地址码段数。要根据指令字长度，程序长度，指令执行时间等方面考察指令系统的效率，选择最优的指令地址码段数，这部分内容在“指令格式的优化”中有阐述。

操作码与地址码

常用的指令编码方式往往使用操作码扩展方式。事实上，如果指令长度一定，则地址码与操作码字段的长度是相互制约的。为了解决这一矛盾，设计时是让操作数地址个数多的指令，其操作码字段短些，操作数地址个数少的指令，其操作码字段长些。这样就可以在不增加指令字长度的情况下，扩展操作码的位数，使其能够表示更多的指令。

例：设某计算机系统的指令字长度为16位，操作码字段为4位，有3段地址码，每段4位。其指令格式如下：

OP（4）	A_1（4）	A_2（4）	A_3（4）

操作码与地址码

- 如果按照定长格式编码的话，4位的操作码最多可以表示16条不同的三地址指令。现假设指令系统中除去有三地址指令之外，还有二地址指令、一地址指令和零地址指令。使用扩展的操作码编码方法可以使指令的条数大大增加。
- 如果基于上述指令格式，要求设计指令编码方法，使指令系统中包含的三地址指令15条，二地址指令15条，一地址指令15条，零地址指令16条，共61条指令。要求给出其编码方法。
- 需要指出的是，指令的编码格式、操作码的扩展方法，可以有不同的方式，在实际编码时要避免出现操作码重复的情况。这一点是必须要做到的，不能有两条不同的指令，操作码却是相同的。另一点就是编码结果要使指令格式尽量规整，这样可以为译码提供方便。

指令格式示例

下面是指令的编码格式：

指令	编码格式	说明
三地址指令	操作码（4位）地址码（三段）	共15条
	0000 **** **** ****	
	0001 **** **** ****	
	……	
	1110 **** **** ****	
二地址指令	操作码（8位）地址码（二段）	共15条
	1111 0000 **** ****	
	1111 0001 **** ****	
	… …	
	1111 1110 **** ****	
一地址指令	操作码（12位）地址码（一段）	共15条
	1111 1111 0000 ****	
	1111 1111 0001 ****	
	… …	
	1111 1111 1110 ****	
零地址指令	操作码（12位，无地址码）	共16条
	1111 1111 1111 0000	
	1111 1111 1111 0001	
	…	
	1111 1111 1111 1111	

指令格式设计

一般情况下，指令系统中的指令字长度或操作码长度都是可变的，如果一个机型指令系统共有N条指令，第i条指令的长度为l_i,每条指令出现的概率为p_i，则指令的平均长度为：

$$\overline{L} = \sum_{i=1}^{N} (l_i * p_i)$$

练习：

设机器字长12位，操作码字段3位，地址码字段每段3位，给出一种设计方案，使得三地址指令4条，二地址指令8条，一地址指令8条。零地址指令最多能够有多少条？

(1472)

第三章 指令系统

- 3.1 引言
- 3.2 指令格式
- 3.3 寻址技术
- 3.4 典型的指令系统
- 3.5 指令系统的优化设计*

指令寻址

含义：寻址方式
- ①确定 本条指令 的 操作数地址
- ②下一条 欲执行 指令 的 指令地址

分类：寻址方式
- 指令寻址 → 寻找指令
- 数据寻址 → 寻找数据

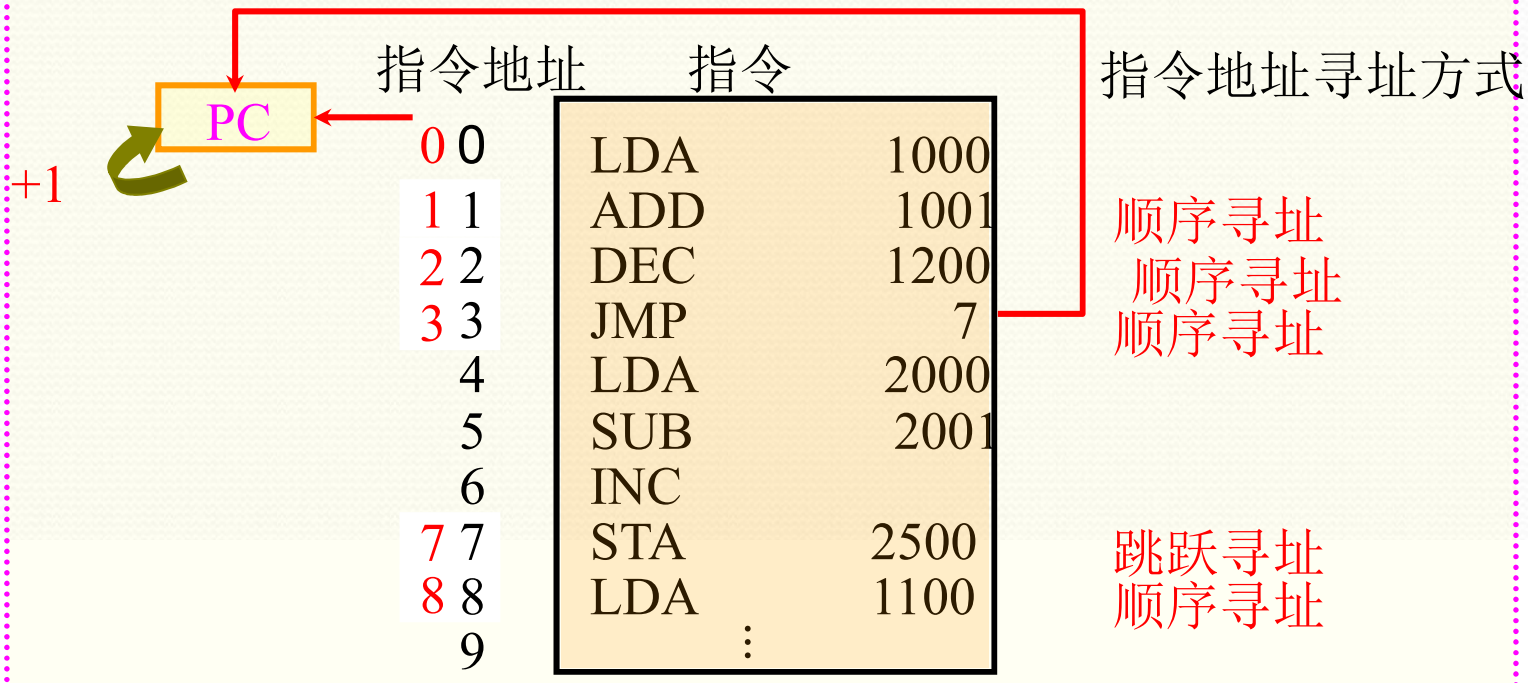
（1）顺序寻址

(PC) + 1

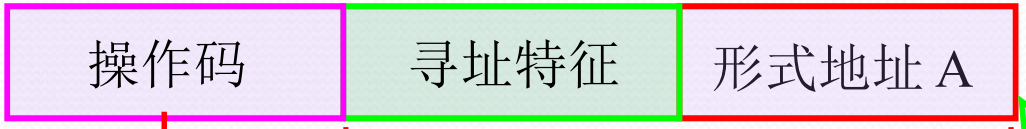
赋值

PC

（2）跳跃：由转移指令指出



数据寻址方式



描述指令功能

决定寻址方式

指令字长 约定 存储字长 = 机器字长

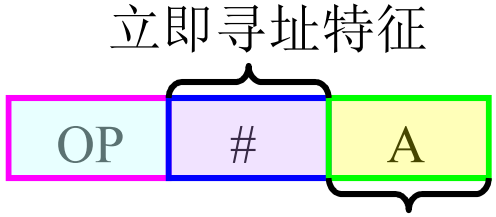
精髓：由形式地址找到有效地址的过程。

指令字中的地址

操作数的真实地址

立即寻址：

（1）格式



（2）结果：形式地址 A 就是操作数

且A可正可负 补码

（3）特点：

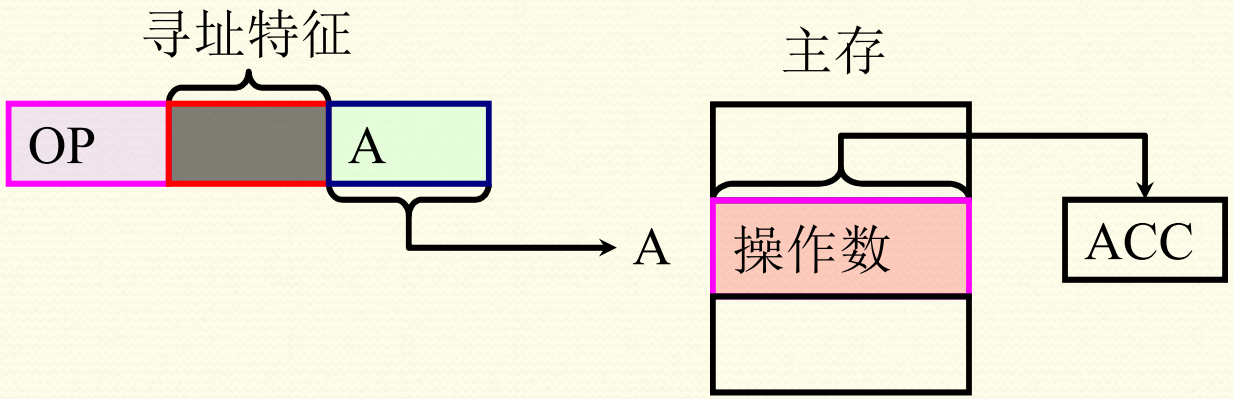
①指令执行阶段不访存

②A 的位数限制了立即数的范围

直接寻址

公式：EA = A　有效地址由形式地址直接给出

变换过程：



特点：

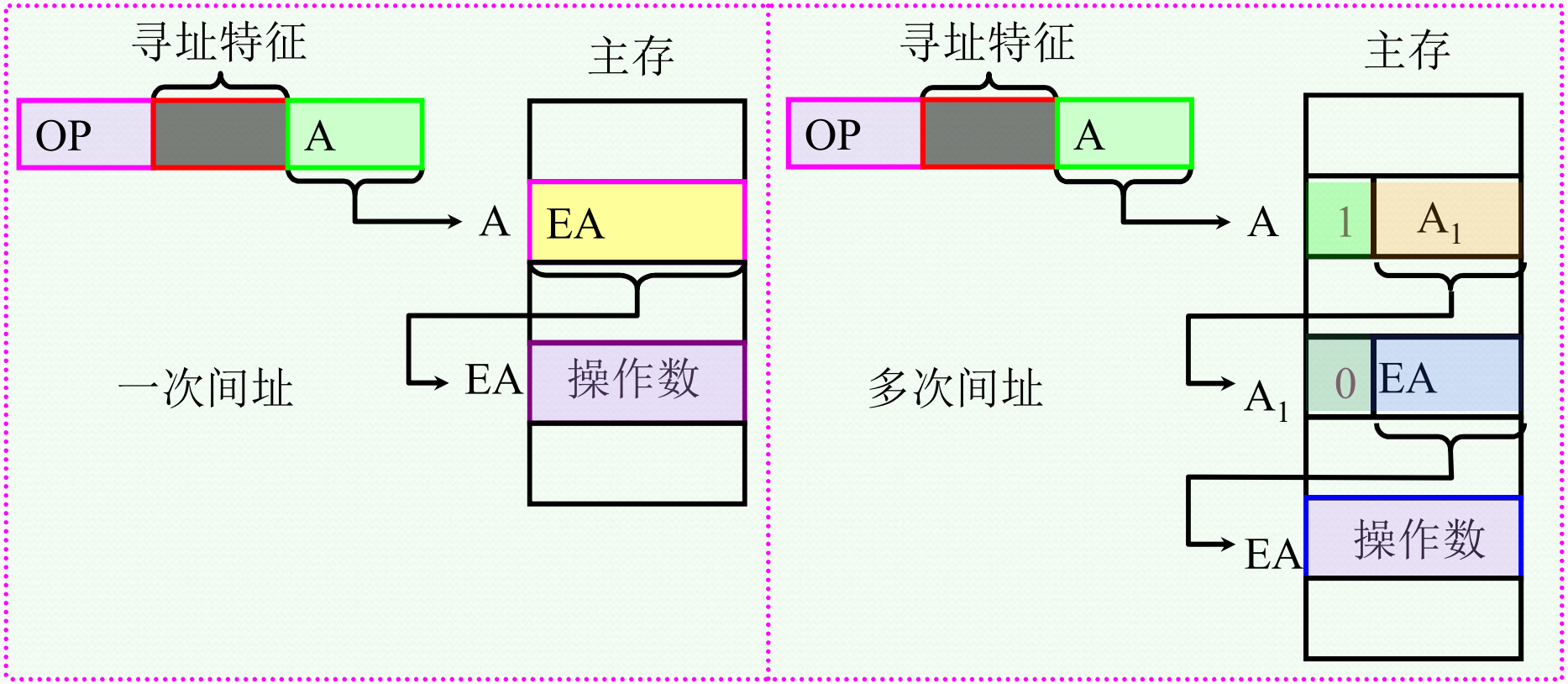
①执行阶段访问一次存储器

②A 的位数决定了该指令操作数的寻址范围

③操作数的地址不易修改（必须修改A）

间接寻址

公式： EA＝（A）有效地址由形式地址间接提供



特点：

①执行指令阶段 2 次访存/多次访存

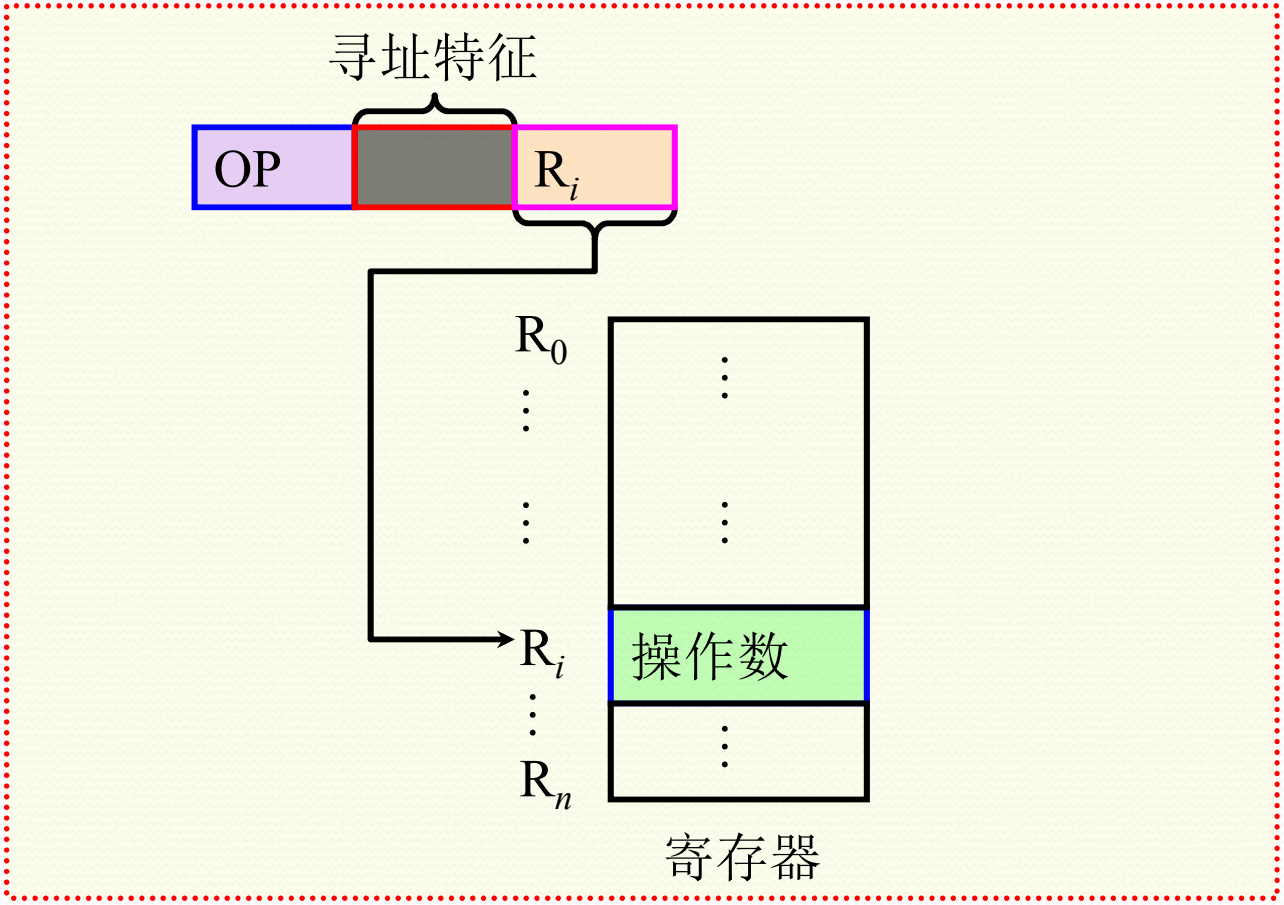
②可扩大寻址范围

③ 便于编制程序

寄存器直接寻址

公式：$EA = R_i$ 有效地址即为寄存器编号

变换过程：



特点：

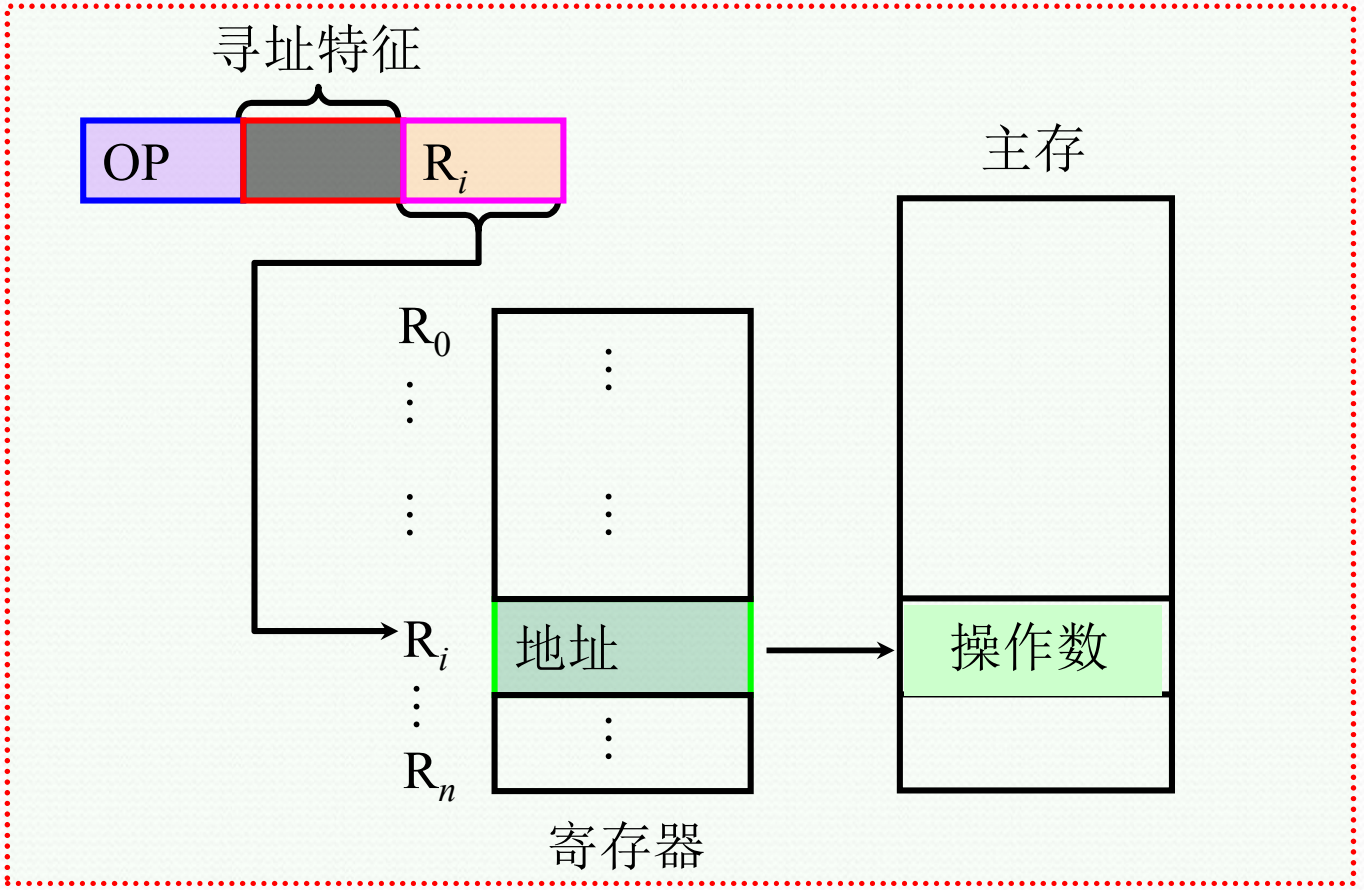
①执行阶段不访存，只访问寄存器，执行速度快

②寄存器个数有限，可缩短指令字长

寄存器间接寻址

公式：EA = (R_i)　有效地址在寄存器中

变换过程：



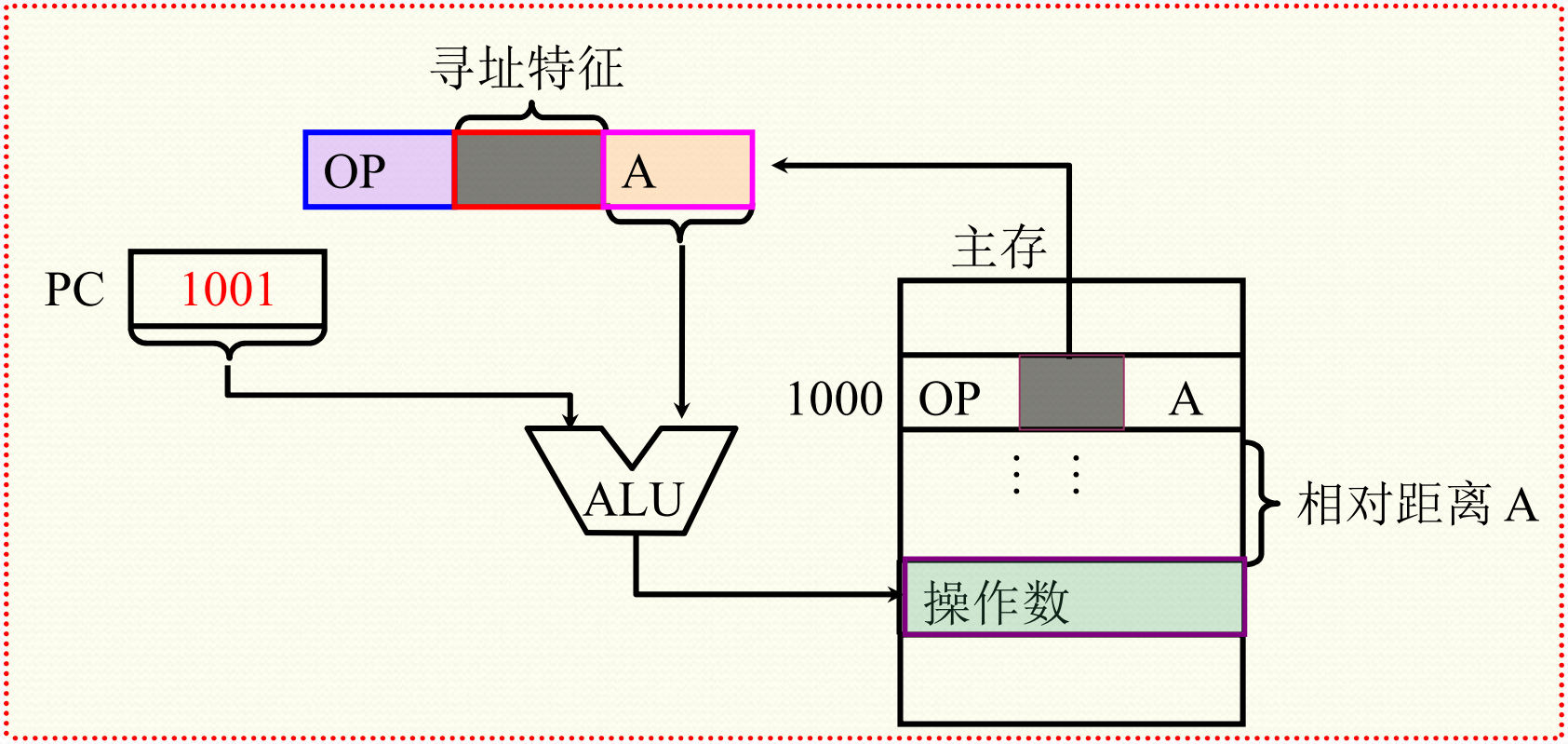
特点：
- ①有效地址在寄存器中，操作数在存储器中，执行阶段访存
- ②便于编制循环程序

相对寻址

公式：EA = (PC) + A

A 是相对于当前指令的位移量（可正可负，补码）

变换过程：



特点：

①A 的位数决定操作数的寻址范围

②程序浮动

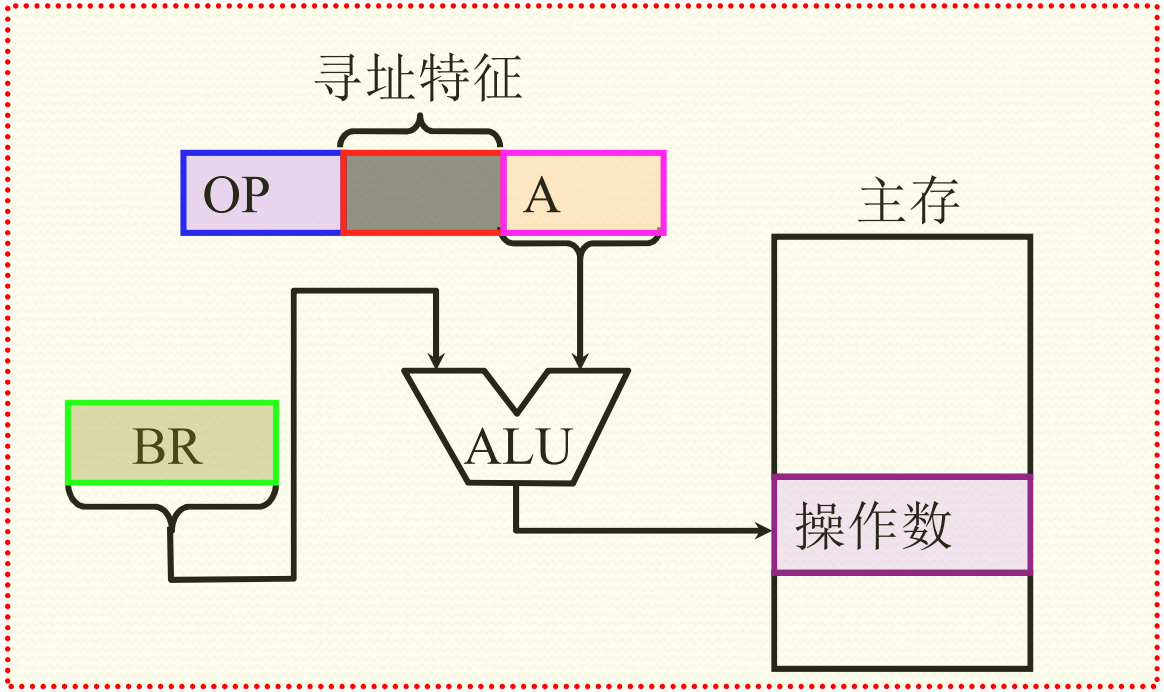
③广泛用于转移指令

基址寻址

(1) 采用专用寄存器作基址寄存器

公式：EA = (BR) + A　　BR 为基址寄存器

变换过程：



特点：

①可扩大寻址范围

②便于程序浮动

③BR 内容由操作系统或管理程序确定

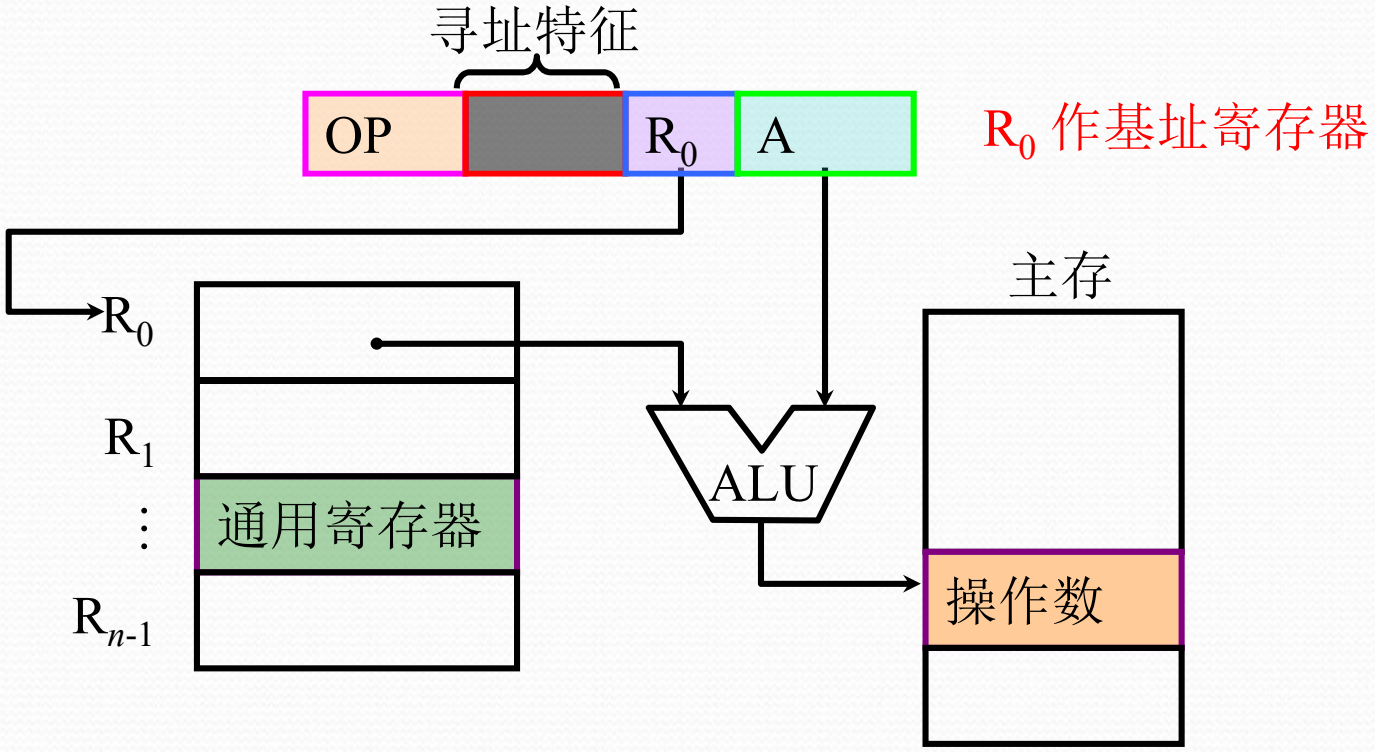
④在程序的执行过程中 BR 内容不变，形式地址 A 可变

基址寻址

(2) 采用通用寄存器作基址寄存器

公式： $EA = (R_i) + A$　　BR 为基址寄存器

变换过程：



特点：

① 由用户指定哪个通用寄存器作为基址寄存器

② 基址寄存器的内容由操作系统确定

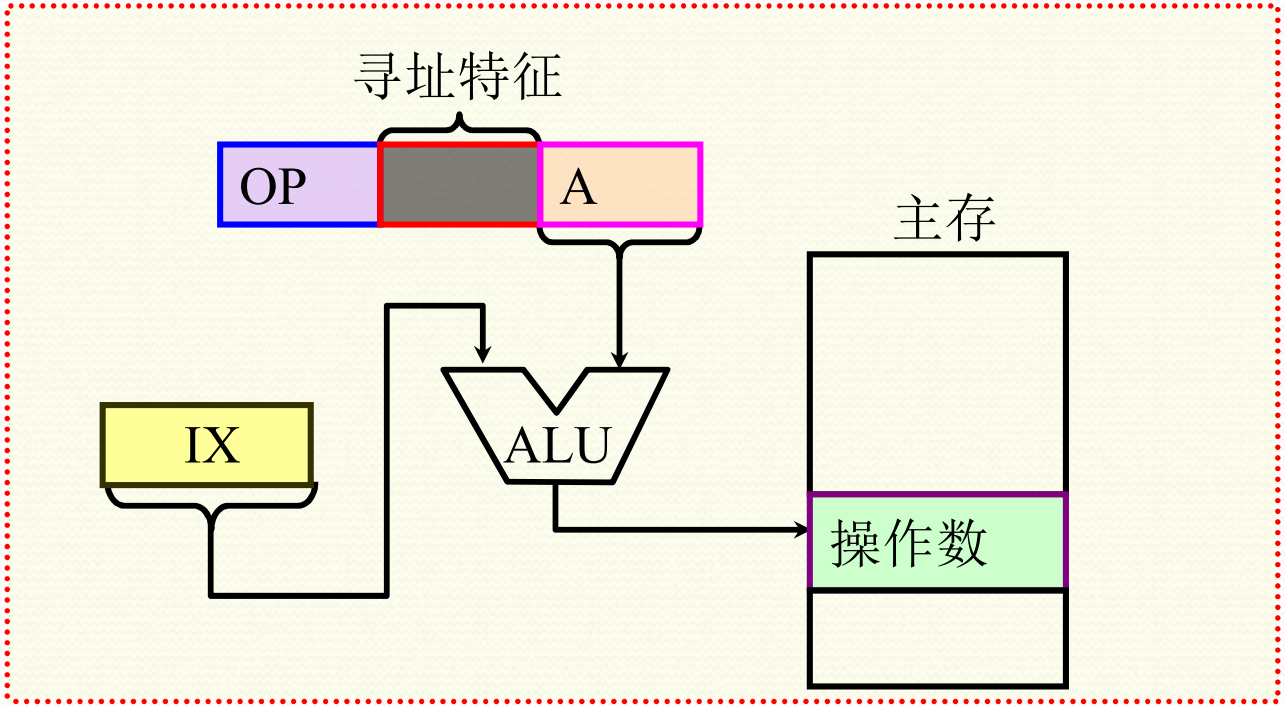
③ 在程序的执行过程中 R_0 内容不变，形式地址 A 可变

变址寻址

特点：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

变换过程：



特点：

①可扩大寻址范围

②IX 的内容由用户给定

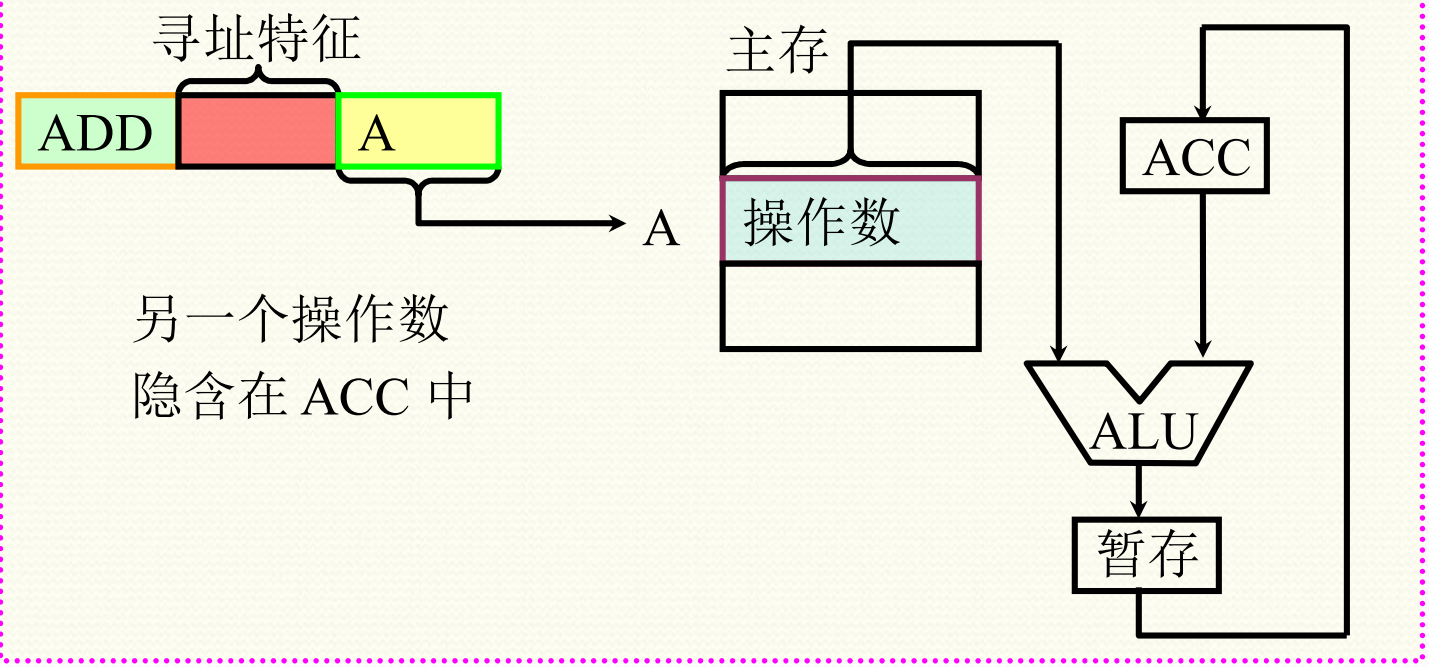
③在程序的执行过程中 IX 内容可变，形式地址 A 不变

④便于处理数组问题

隐含寻址

公式：操作数地址隐含在操作码中

变换过程：



举例：如 8086

MUL 指令　被乘数隐含在 AX（16位）或 AL（8位）中

MOVS 指令　源操作数的地址隐含在 SI 中

目的操作数的地址隐含在 DI 中

特点：指令字中少了一个地址字段，可缩短指令字长

块寻址 & 段寻址方式

块寻址方式

块寻址方式经常用在输入输出指令中，以实现外存储器或外围设备同内存之间的数据块传送。块寻址方式在内存中还可用于数据块搬移。块寻址时，通常在指令中指出数据块的起始地址（首地址）和数据块的长度（字数或字节数）。

如果数据块是变长的，可用三种方法指出它的长度：

（1）指令中划出字段指出长度；

（2）指令格式中指出数据块的首地址与末地址；

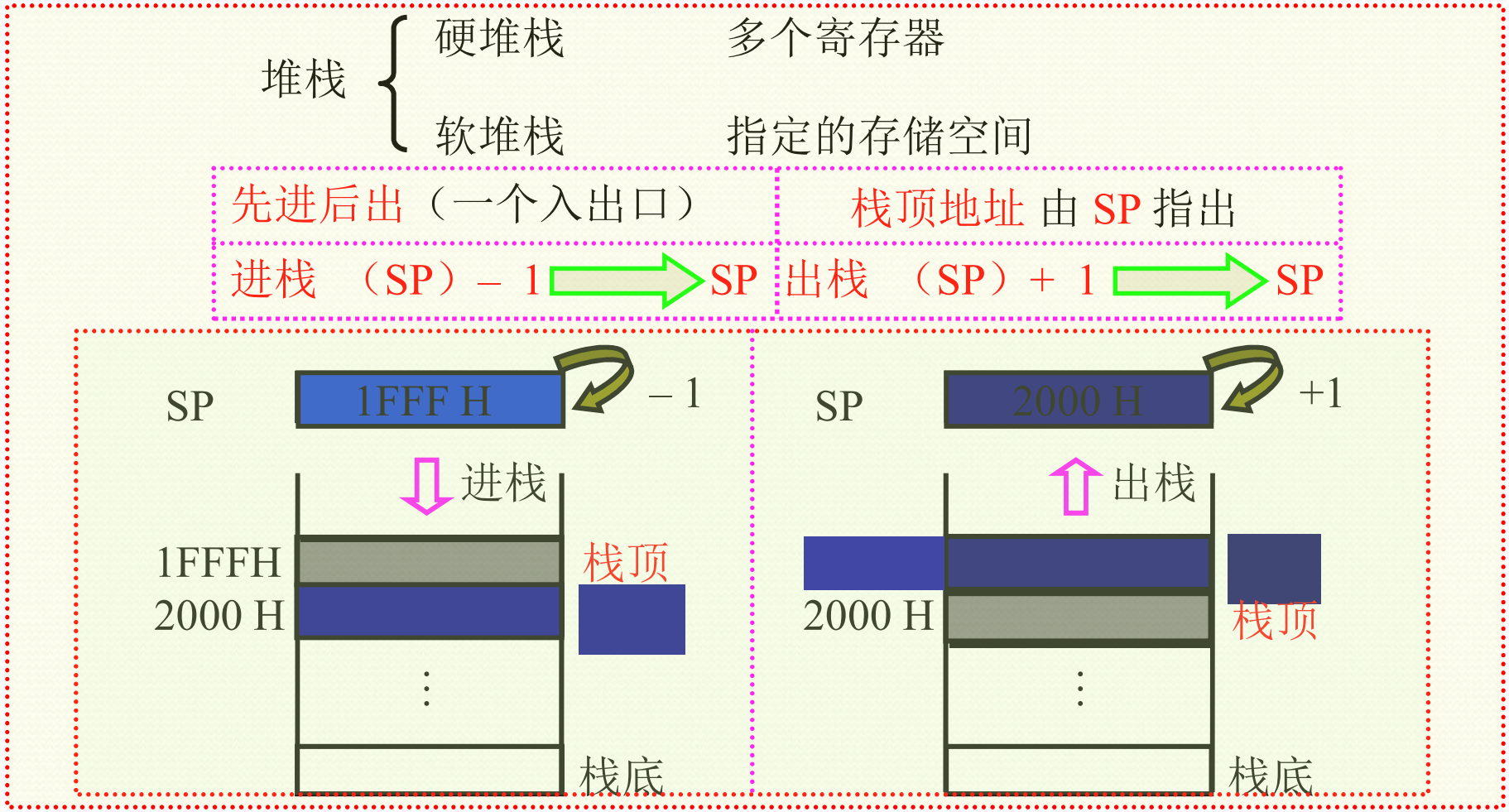
（3）由块结束字符指出数据块长度。

段寻址方式

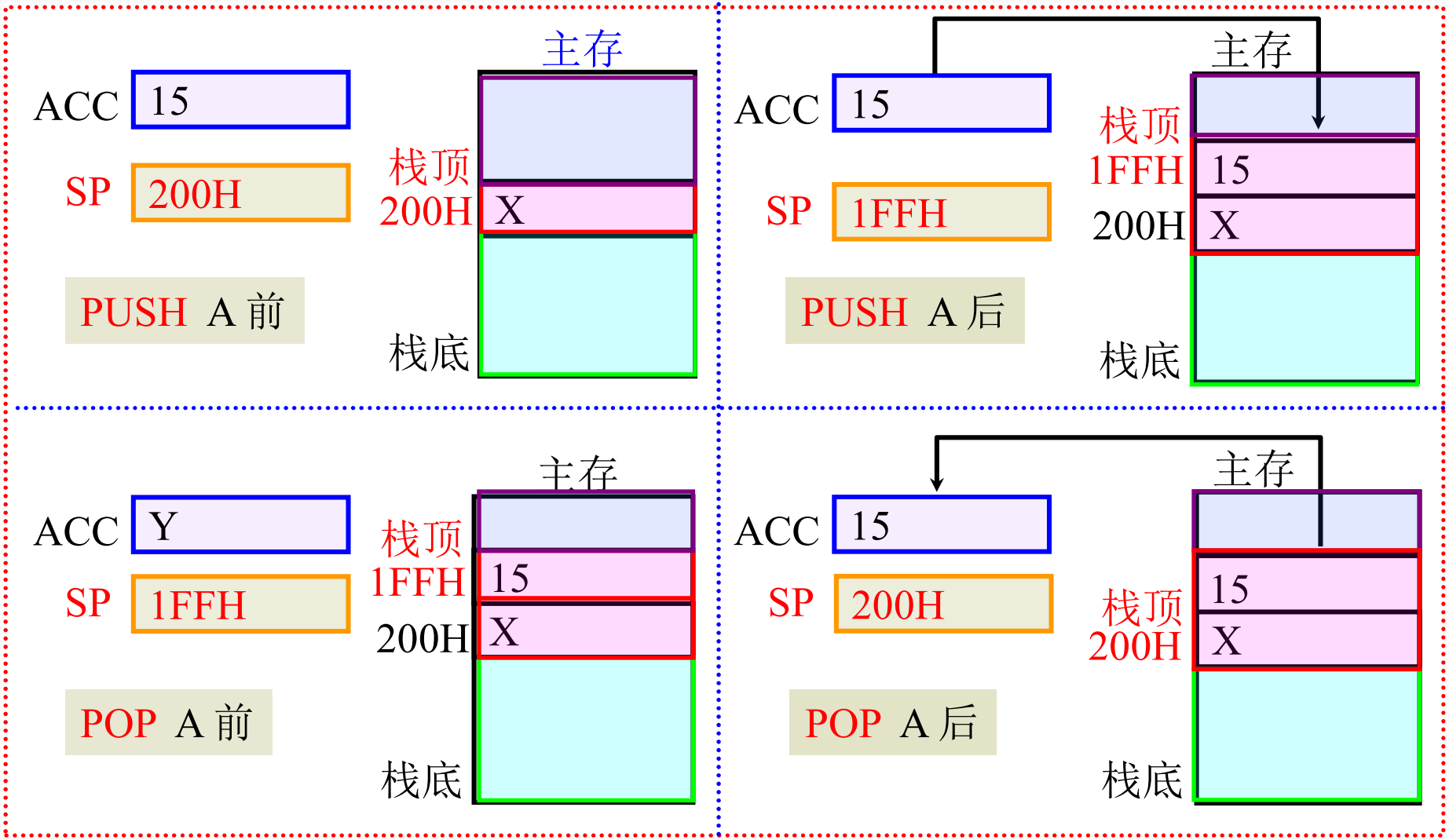
微型机中采用了段寻址方式 。例如它们可以给定一个20位的地址，从而有1M存储空间的直接寻址能力。

堆栈寻址

(1) 堆栈的特点


堆栈
硬堆栈
多个寄存器
软堆栈
指定的存储空间
先进后出（一个入出口）
栈顶地址 由 SP 指出
进栈 （SP）– 1 SP
出栈 （SP）+ 1 SP
SP
1FFF H
– 1
进栈
1FFFH
2000 H
栈顶
栈底
SP
2000 H
+1
出栈
2000 H
栈顶
栈底

堆栈寻址举例


主存
ACC
15
SP
200H
栈顶
200H
X
栈底
PUSH A 前
主存
ACC
15
SP
1FFH
栈顶
1FFH
15
200H
X
栈底
PUSH A 后
主存
ACC
Y
SP
1FFH
栈顶
1FFH
15
200H
X
栈底
POP A 前
主存
ACC
15
SP
200H
栈顶
200H
15
X
栈底
POP A 后

SP 的修改与主存编址方法

① 按 字 编址	进栈	（SP）– 1 ⇨ SP
	出栈	（SP）+ 1 ⇨ SP
② 按 字节 编址 (存储字长 16 位)	进栈	（SP）– 2 ⇨ SP
	出栈	（SP）+ 2 ⇨ SP
② 按 字节 编址 (存储字长 32 位)	进栈	（SP）– 4 ⇨ SP
	出栈	（SP）+ 4 ⇨ SP

例 题

例：某机器存储容量为64K×16位，该机访存指令格式如下：

OP		M		I	X	A	
15	12	11	10	9	8	7	0

其中M为寻址模式：0为直接寻址，1为基址寻址，2为相对寻址，3为立即寻址；I为间址特征(I=1间址);X为变址特征(X=1变址)

设PC为程序计数器，Rx为变址寄存器，Rb为基址寄存器，试问：

(1)该指令能定义多少种操作？

(2)立即寻址操作数的范围是多少？

(3)在非间址情况下，除立即寻址外，写出每种寻址方式计算有效地址的表达式。

(4)设基址寄存器为14位，在基址寻址时，指令的寻址范围是多少？

(5)间接寻址时，寻址范围是多少？若允许多重间址，寻址范围是多少？

例 题

解：

OP		M		I	X	A	
15	12	11	10	9	8	7	0

(1) 根据题意，操作码部分4位，可定义16种操作

(2) 立即寻址操作数的范围是: -128~+127

(3) 直接寻址：EA=A；基址寻址：EA=(R_B)+A;变址寻址：EA=(R_x)+A;
相对寻址：(PC)+A

(4)设基址寄存器为14位，在非变址基址寻址时，EA= (R_B)+A，因R_B为14位，因此寻址范围16K。

(5)间接寻址时，寻址范围64K;允许多重间址，寻址范围 是32K

第三章 指令系统

- 3.1 引言
- 3.2 指令格式
- 3.3 寻址技术
- 3.4 典型的指令系统
- 3.5 指令系统的优化设计*

指令的分类

(1). 数据传送指令

数据传送指令主要包括取数指令、存数指令、传送指令、成组传送指令、字节交换指令、清累加器指令、堆栈操作指令等等。这类指令主要用来实现主存和寄存器之间，或寄存器和寄存器之间的数据传送。

(2). 算术运算指令

这类指令包括二进制定点加、减、乘、除指令，浮点加、减、乘、除指令，求反、求补指令，算术移位指令，算术比较指令，十进制加、减运算指令等。这类指令主要用于定点或浮点的算术运算，大型机中有向量运算指令，直接对整个向量或矩阵进行求和、求积运算。

指令的分类

(3). 逻辑运算指令

这类指令包括逻辑加、逻辑乘、按位加、逻辑移位等指令，主要用于无符号数的位操作、代码的转换、判断及运算。移位指令用来对寄存器的内容实现左移、右移或循环移位。

(4). 程序控制指令

程序控制指令也称转移指令。执行程序时，有时机器执行到某条指令时，出现了几种不同结果，这时机器必须执行一条转移指令，根据不同结果进行转移，从而改变程序原来执行的顺序。这种转移指令称为条件转移指令。除各种条件转移指令外，还有无条件转移指令、转子程序指令、返回主程序指令、中断返回指令等。转移指令的转移地址一般采用直接寻址和相对寻址方式来确定。

指令的分类

(5). 输入输出指令

输入输出指令主要用来启动外围设备，检查测试外围设备的工作状态，并实现外部设备和CPU之间，或外围设备与外围设备之间的信息传送。

(6). 字符串处理指令

字符串处理指令是一种非数值处理指令，一般包括字符串传送、字符串转换（把一种编码的字符串转换成另一种编码的字符串）、字符串替换（将某一字符串用另一字符串替换）等。这类指令可以在文字编辑中对大量字符串进行处理。

指令的分类

(7). 特权指令

特权指令是指具有特殊权限的指令。这类指令只用于操作系统或其他系统软件，一般不直接提供给用户使用。在多用户、多任务的计算机系统中特权指令必不可少。它主要用于系统资源的分配和管理。

(8). 其他指令

除以上各类指令外，还有状态寄存器置位、复位指令、测试指令、暂停指令，空操作指令，以及其他一些系统控制用的特殊指令。

以上几种类型的指令涵盖了计算机系统的全部指令系统。可能有的资料采用不同的描述，但是作为指令系统来讲，其主要内容也不外乎这几种类型的指令。

精简指令系统

精简指令系统计算机是目前发展迅速的计算机系统，由于其特有的优点深受系统设计者和用户的青睐。这里简要介绍一下RISC指令系统的类型及特点。

RISC指令系统的最大特点是：

(1)选取使用频率最高的一些简单指令，指令条数少；

(2) 指令长度固定，指令格式种类少；

(3) 只有取数／存数指令访问存储器，其余指令的操作都在寄存器之间进行。

典型RISC机指令系统基本特征

型号	指令条数	寻址方式	指令格式	通用寄存器数	主频/MHZ
RISC-I	**31**	**2**	**2**	**78**	**8**
RISC-II	**39**	**2**	**2**	**138**	**12**
MIPS	**55**	**3**	**4**	**16**	**4**
SPARC	**75**	**4**	**3**	**120-136**	**25-33**
MIPSR3000	**91**	**3**	**3**	**32**	**25**
i860	**65**	**3**	**4**	**32**	**50**
Power PC	**64**	**6**	**5**	**32**	

MIPS指令系统

以MIPS为例。MIPS-C共有55条指令。

指令类型和指令格式 (www.mips.com)

MIPS共有三种指令格式，其格式列于下表。

MIPS中32个寄存器

寄存器编号	MIPS助记符	释义	备注
0	$Zero	固定值为0	硬件置位
1	$at	汇编器保留	
2~3	$v0~$v1	函数调用返回值	
4~7	$a0~$a3	函数调用参数	4个参数
8~15	$t0~$t7	暂存寄存器	8个参数
16~23	$s0~$s7	通用寄存器	调用之前需保存
24~25	$t8~$t9	暂存寄存器	2个
26~27	$k0~$k1	操作系统保留	
28	$gp	全局指针	
29	$sp	堆栈指针	
30	$fp	帧指针	
31	$ra	函数返回地址	

MIPS-32 指令格式(R型指令)

- ✓ OP：指令的基本操作---操作码
- ✓ Rs：第一个源操作数寄存器
- ✓ Rt：第二个源操作寄存器
- ✓ Rd：存放结果的目的操作寄存器
- ✓ Shamt：偏移量，用于移位指令
- ✓ Funct：函数，对操作码进行补充

MIPS指令格式(R型指令)

		6bits	5bits	5bits	5bits	5bits	6bits
指令	格式	OP	rs	rt	rd	shamt	funct
add	R	0	Reg	Reg	Reg	0	32_{10}
sub	R	0	Reg	Reg	Reg	0	34_{10}
and	R	0	Reg	Reg	Reg		36
or	R	0	Reg	Reg	Reg		37
nor	R	0	Reg	Reg	Reg		39
sll	R	0	Reg	Reg	Reg		0
srl	R	0	Reg	Reg	Reg		2
jr	R	0	REG	0	0	0	8

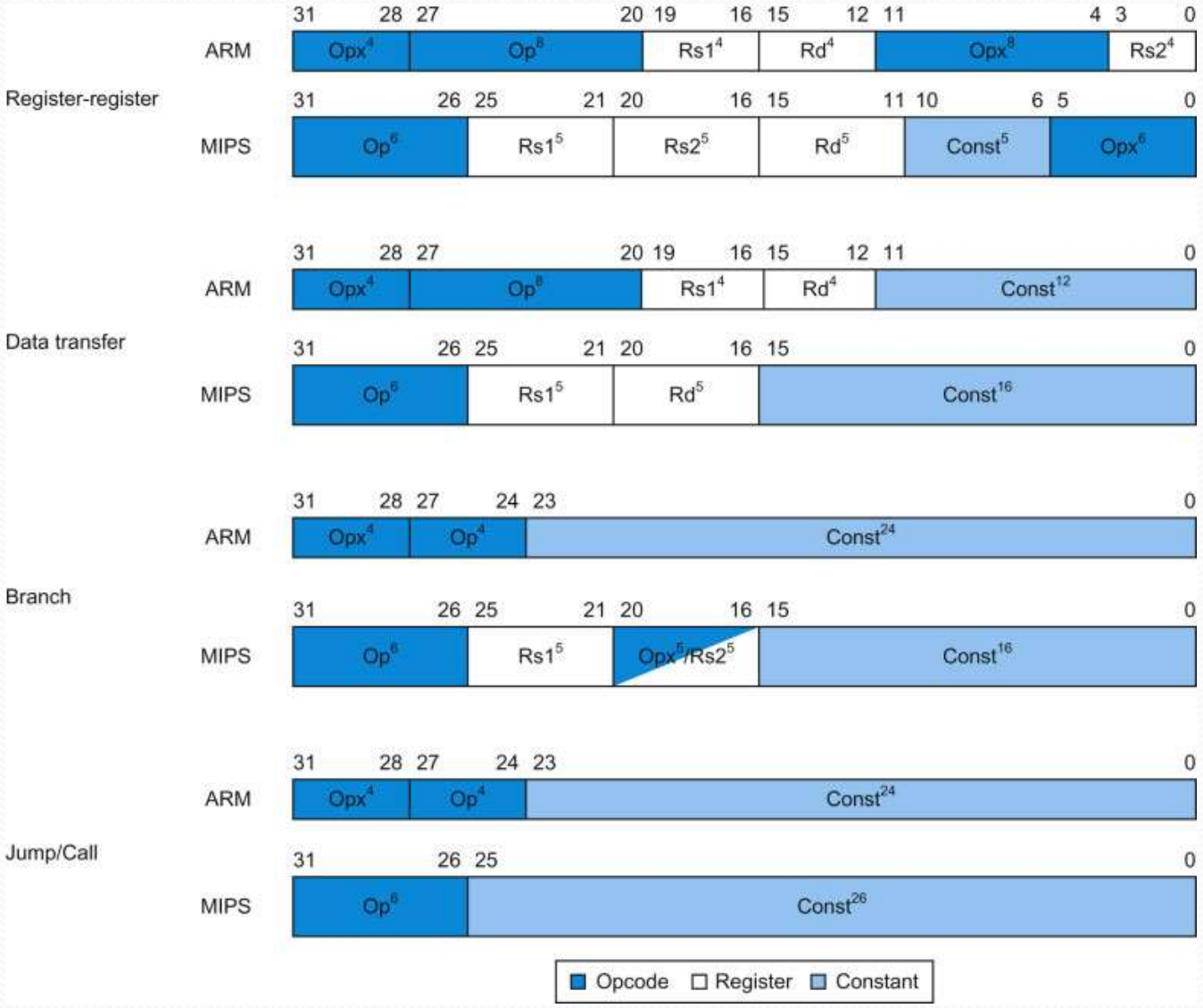
add	**R**	**0**	18	19	17		**32**

add $s1,$s2,$s3 # machine code 0x2538820

MIPS指令格式(I、J型指令)

		6bits	5bits	5bits	5bits	5bits	6bits
指令	格式	OP	rs	rt	rd	shamt	funct
add	R	0	Reg	Reg	Reg	0	32_{10}
addi	I	8	Reg	Reg	16bits 立即数		
lw	I	35	Reg	Reg	16bits 立即数		
sw	I	43	Reg	Reg	16bits 立即数		
andi	I	12	Reg	Reg	16bits 立即数		
ori	I	13	Reg	Reg	16bits 立即数		
beq	I	4	Reg	Reg	16bits 立即数（相对寻址）		
bne	I	5	Reg	Reg	16bits 立即数（相对寻址）		

j	**J**	**2**	26bit 立即数(伪直接寻址)
jal	**J**	**3**	26bit 立即数(伪直接寻址)



Instruction formats, ARM and MIPS. The differences result from whether the architecture has 16 or 32 registers.

例 题

例：某16位机器所使用的指令格式和寻址方式如下所示，该机有两个20位基址寄存器，四个16位变址寄存器，十六个16位通用寄存器。指令汇编格式中的S(源)，D(目标)都是通用寄存器，M是主存中的一个单元。

15 10	9 8	7 4	3 0
OP	--	目标	源

MOV S，D

15 10	9 8	7 4	3 0
OP	基址	源	变址
位移量			

STO S，M

15 10	9 8	7 4	3 0
OP	--	目标	
20位地址			

LAD M，D

三种指令的操作码分别是MOV(OP)=$(A)_H$，STO(OP)=$(1B)_H$，LAD(OP)=$(3C)_H$。MOV是传送指令，STO为存数指令，LAD为取数指令。要求：

(1) 分析三种指令的指令格式与寻址方式特点。

(2) CPU完成哪一种操作所花时间最短?哪一种操作所花时间最长?第二种指令的执行时间有时会等于第三种指令的执行时间吗?

(3) 下列情况下每个十六进制指令字分别代表什么操作?其中如果有编码不正确，如何改正才能成为合法指令?

$(F0F1)_H(3CD2)_H$; $(2856)_H$; $(6FD6)_H$; $(1C2)_H$

解：(1) 第1种指令为单字长指令，RR型；第二种双字长二地址指令，RS型，其中S采用基址或变址寻址，R由源寄存器决定；第三种指令双字长二地址指令，RS型，R由目标寄存器决定，S由20位地址决定。

(2) 第一种操作所花时间最短；第二种操作所花时间最长；第二种指令的执行时间不等于第三种指令的执行时间。

(3) $(F0F1)_H(3CD2)_H$ ---双字长指令

=111100 00 1111 0001 0011 1100 1101 0010

LAD指令，1111(15号寄存器)，即$R_{15}\leftarrow (13CD2)_H$

$(2856)_H$ =0010 10 00 0101 0110---单字长; $R_5\leftarrow(R_6)$

$(6FD6)_H$---单字长，一定是MOV指令，但编码错误，可改为$(28D6)_H$

$(1C2)_H$---单字长，一定是MOV指令，但编码错误，可改为$(28C2)_H$

第三章 指令系统

- 3.1 引言
- 3.2 指令格式
- 3.3 寻址技术
- 3.4 典型的指令系统
- 3.5 指令系统的优化设计*

操作码的优化设计

1．Huffman编码法

根据Huffman编码法的原理，采用最优Huffman编码法表示的操作码的最短平均长度可以通过如下公式计算得到：

$$H = -\sum_{i=1}^{n} p_i * \log_2 p_i$$

其中pi表示第i种指令在程序中出现的概率，一共有n种操作码。

相对于最优Huffman编码法，固定长度操作码编码法的信息冗余量可以表示为：

$$R = 1 - \frac{-\sum_{i=1}^{n} p_i \cdot \log_2 p_i}{\lceil \log_2 n \rceil}$$

Huffman编码的基本思想

当各种事件发生的概率不均时，采用优化技术对发生概率最高的事件用最短的位数（时间）来表示（处理），而对出现概率较低的，用较长的位数（时间）来表示（处理），就会导致表示（处理）的平均位数（时间）的缩短。

Huffman编码的基本步骤

（1）将要编码的字符按出现频率的次序排列，频率相等的符号可任意排列；

（2）把出现频率最小的两个符号合并，并将其频率相加，按相加后的频率次序重新排序；

（3）继续过程（2），直至只剩下两个频率，此后以相反过程进行编码；

（4）对最后两个频率分别指定代码0和1；

（5）若某一频率由两个频率相加而成，则分别指定这两个频率的下一个代码为0或1；

（6）继续过程(5)，直到所有符号均已指定不同代码为止。

平均编码长度

采用Huffman编码法所得到的是操作码的平均长度，计算方法为：

$$H = \sum_{i=1}^{n} p_i \cdot l_i$$

其中p_i代表第i种操作码在程序中出现的概率，l_i表示第i种操作码的二进制编码位数，n表示操作码的数量。

扩展编码法

- 最终目的是使操作码在冗余量尽量小的基础上，努力使操作码规整一些。采用扩展编码法，可以有多种结果。
- 对于等长扩展法，根据采用不同的扩展标志还可以有多种不同的扩展方法。例如，对于4-8-12等长扩展法，有采用保留一个码点标志的15/15/15……扩展法，采用每次保留一个标志位的8/64/512……扩展法。当然也可以在扩展过程中每次采用不同的扩展标志，形成其它的扩展方法。
- 对于4-8-12等长扩展法的15/15/15……扩展法和8/64/512……扩展法，操作码的具体编码方法如下表所示

扩展编码法

15/15/15……扩展法		8/64/512……扩展法	
操作码编码	说明	操作码编码	说明
0000 0001 … 1110	4位长度的操作码 共15种	0000 0001 … 0111	4位长度的操作码 共8种
1111 0000 1111 0001 … 1111 1110	8位长度的操作码 共15种	1000 0000 1000 0001 … 1111 0111	8位长度的操作码 共64种
1111 1111 0000 1111 1111 0001 … 1111 1111 1110	12位长度的操作码 共15种	1000 1000 0000 1000 1000 0001 … 1111 1111 0111	12位长度的操作码 共512种

地址码的优化设计

1) 地址码个数的选择

- 对于程序存储量的计算
- 对于程序执行速度的计算

2) 缩短单个地址码长度的方法

- 用间接寻址方式缩短地址码长度
- 用变址寻址方式缩短地址码长度
- 用寄存器间址寻址方式缩短地址码长度

作业

P.335-336：1，4，6，7，8，18，21

谢谢！