



计算机组成与体系结构

第5章 指令系统

赵庆行、张骏鹏
人工智能学院

复习

- 指令及指令系统作用
 - 指令（最小功能单元）、指令系统/指令集、指令系统结构层/指令集体系结构(ISA)
- 指令系统结构层及定义（即设计）
 - 存储模式：边界对齐、堆栈、冯诺依曼/哈佛结构、加载/存储结构
 - 寄存器组织：寄存器文件、典型寄存器
 - 数据类型
 - I/O模式：统一编址、独立编址及其优缺点
 - 指令系统：I/O操作方式

本次课重点

- 指令格式
- 地址码设计
 - 地址字段数量
 - 基本寻址方式
- 操作码设计
 - 定长操作码
 - 变长操作码
- 指令长度设计



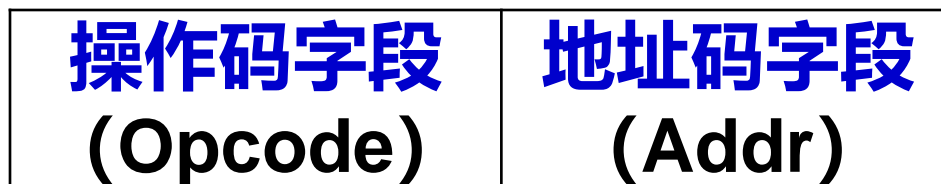
指令格式

5.3 指令设计


- ▶ 在确定了指令功能、数据类型、寄存器组织等与指令系统架构相关的信息之后，具体的指令设计就成为指令系统**设计的核心**。

5.3.1 指令格式

- 为了使指令能够有效地指挥计算机完成各种操作，一条指令应包含两个基本要素：操作码和地址码，其基本格式为：



指定指令要完成的功能，即计算机要完成的某个基本操作。



提供指令的操作对象。可以直接提供操作数，也可以提供操作数的存放地址



指令设计

--地址码字段设计

5.3.2 地址码设计


- ▶ 指令功能不同，需要的操作数数量也不同。
- ▶ 地址码字段为指令指定源操作数、目的操作数和下条指令地址等信息，格式如下：

地址码字段


目的操作数

源操作数


下条指令地址



指示指令执行结束时结果数据的存放地，目的地可以是主存、寄存器或I/O端口。
0到1个



指示指令执行开始时所需原始数据的来源，来源于指令本身、主存、寄存器或I/O端口。
0到多个



指示CPU欲执行的下一条指令从主存中何地址获取，仅在程序控制类指令或部分系统控制类指令中使用，在指令执行顺序发生改变时，由该字段提供转移地址（**顺序指令地址由程序计数器PC提供**）

5.3.2 地址码设计

- 按照地址码字段的数量，指令可以分为四地址、三地址、两地址、单地址和零地址指令，格式为：

四地址指令	Op	Addr1	Addr2	Addr3	Addr4
三地址指令	Op	Addr1	Addr2	Addr3	
二地址指令	Op	Addr1	Addr2		
一地址指令	Op	Addr1			
零地址指令	Op				



- ✓ 四地址指令： **op rd, rs1, rs2, ni**
; $rs1 \text{ op } rs2 \rightarrow rd$, ni 提供顺序或转移地址
- ✓ 三地址指令： **op rd, rs1, rs2**
; $rs1 \text{ op } rs2 \rightarrow rd$, PC 提供顺序地址
- ✓ 二地址指令： **op rd, rs1**
; $rd \text{ op } rs1 \rightarrow rd$ 或 $rs1 \text{ op } ACC \rightarrow rd$, PC 提供顺序地址
- ✓ 一地址指令： **op rd**
; $rd \text{ op } ACC \rightarrow ACC$ 或 rd 自身操作 $\rightarrow rd$, PC 提供顺序地址；或者 rd 提供转移地址。 ACC 为累加器或操作码隐含指定的操作数
- ✓ 零地址指令： **op**
; 由操作码指定操作数（隐含寻址）或无需操作数

5.3.2 地址码设计

- ▶ **地址码数量**和机器的字长有密切的关系：
 - 8位机通常采用一地址指令格式
 - 16位机更多采用二地址指令格式
 - 32位以上的计算机有条件选择三地址结构
- ▶ 地址字段愈多，完成同样功能所需的指令条数愈少。
- ▶ 对于相同的操作，不同地址码结构的计算机有不同的实现方案。
- ▶ 出于指令兼容性的考虑，往往在一个指令系统中会出现几种地址结构指令混合的情况。

5.3.2 地址码设计

- 为了用短地址码灵活方便地提供操作数来源和存放地信息，给程序设计提供更好的支持，所有计算机的指令系统无一例外对地址码字段使用专门设计的**寻址方式**进行编码。
- 影响指令**地址码长度**的因素包括：
 - 寻址方式的种类
 - 操作数数目
 - 寄存器数目
 - 主存地址范围
 - 主存可寻址最小单元等。

假设为加法指令ADD设计指令格式，下一条指令地址由程序计数器PC提供，则合适的选择是采用

- A. 零地址指令格式
- B. 一地址指令格式
- C. 二地址指令格式
- D. 三地址指令格式

假设为加法指令**ADD**设计指令格式，下一条指令地址由程序计数器PC提供，则合适的选择是采用

- A. 零地址指令格式
- B. 一地址指令格式 ✓
- C. 二地址指令格式 ✓
- D. 三地址指令格式 ✓





指令设计

--基本寻址方式

5.4.1 基本寻址方式

- 寻址方式就是指令获取操作数的方式。也即，地址码字段提供操作数、转移地址的编码方式。
- 寻址方式的**设计**要考虑两个方面：
 - 能够有效压缩地址码字段长度
 - 能够支持灵活的程序设计
- 寻址方式在指令中以**两种方式**呈现：
 - 由操作码决定其寻址方式，称为**隐式寻址**；
 - 指令中设置寻址方式字段，由寻址方式字段的**不同编码**来指定操作数的寻址方式，称为**显式寻址**。
- 一台计算机可以采用多种寻址方式。多数计算机都会首先采用公认的基本寻址方式，之后再增加某些独特的寻址方式，以此构成本系统的寻址方式。

5.4.1 基本寻址方式

1. 隐含寻址

- **特征**：操作数的存放地由操作码指定。
- 以Intel指令为例

DAA；加法十进制调整指令，

隐含规定调整对象为寄存器**AL**

MUL BL；乘法指令，

隐含规定乘数之一是8位寄存器**AL**、乘积存放在16位寄存器**AX**中。

5.4.1 基本寻址方式

2. 立即寻址

- **特征**：操作数在指令中。



- 以Intel指令为例

MOV AL, 02H ;源操作数为立即寻址

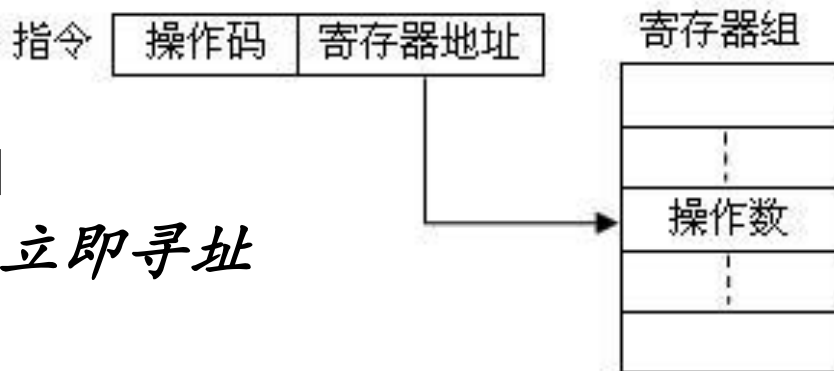
- **优点**：有利于加快指令的执行速度，
- **缺点**：会增加指令长度，不便于修改。
- **用途**：只适用于提供常数、设定初始值的场合。

5.4.1 基本寻址方式

3. 寄存器寻址

➤ **特征：**操作数在指令指定的寄存器中。

MOV AL, 02H
;目的操作数为立即寻址



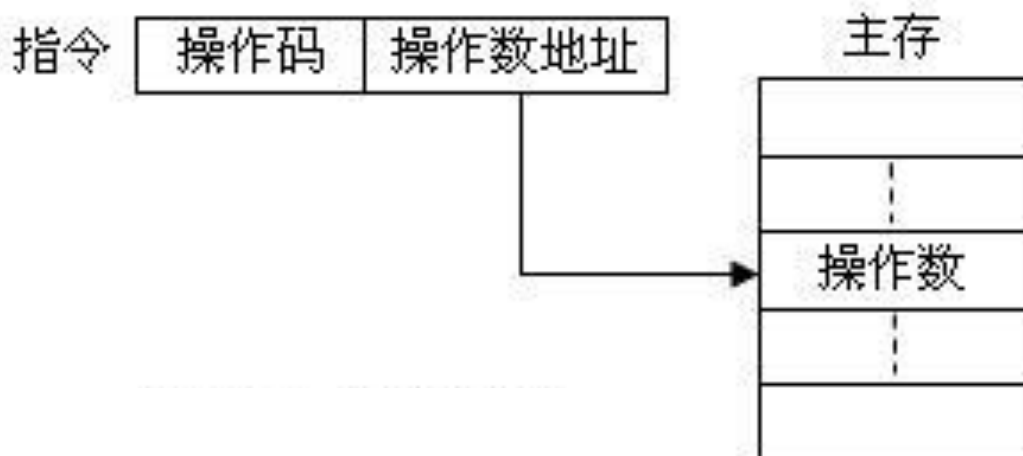
➤ **优点：**

- 操作数在不改变程序的情况下可以方便地修改
- 寄存器存取数据的速度比访问主存更快，功耗更小
- 寄存器地址比主存单元地址短，可以有效缩短指令长度
- 用寄存器存放基址值、变址值可派生出其他寻址方式，使编程更具有灵活性。

5.4.1 基本寻址方式

4. 直接寻址

- **特征**：操作数地址在指令中，操作数在主存单元中，访问一次主存便可获得操作数。



- **缺点**：直接提供的主存地址需要较多的二进位，造成指令字较长，同时也不便于操作数在主存中存放地址的灵活修改。

5.4.1 基本寻址方式

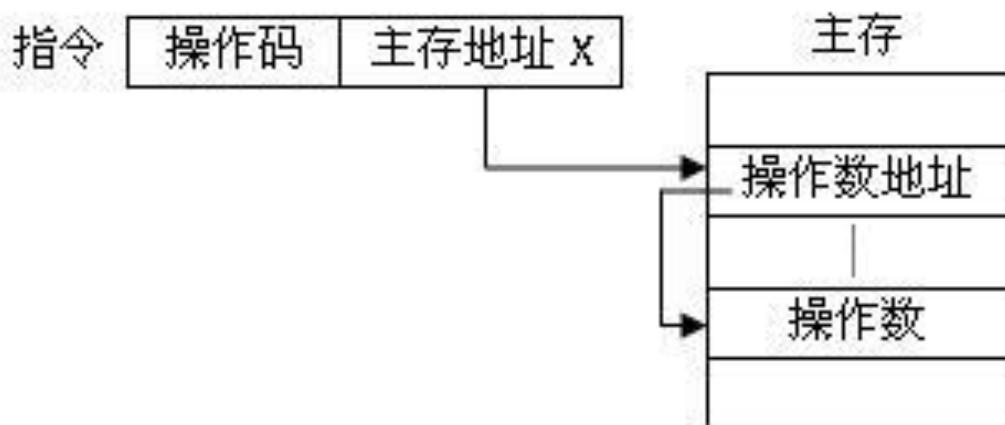
5. (存储器)间接寻址

- **特征**：操作数地址的地址在指令中，操作数在主存单元中。

操作数地址 = EA = (X)

↑ ↑ ↑

实际(物理)地址 形式地址 有效地址

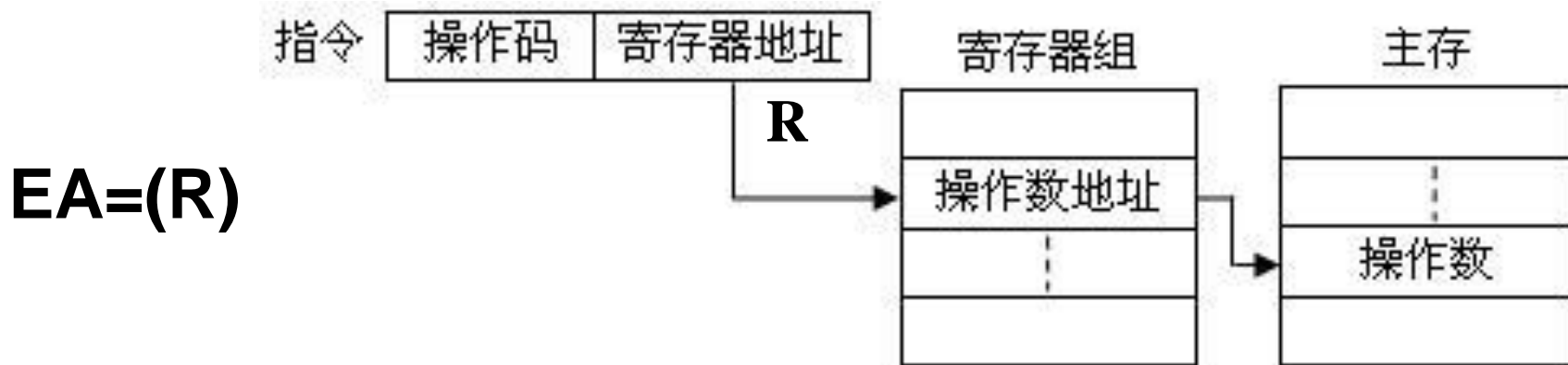


- **优点**：主存单元X作为数据地址的指针，使操作数和操作数地址均可在不修改指令的情况下加以修改，提供了灵活的编程手段。
- **缺点**：需要访问两次主存才能获得操作数，降低指令的执行速度。

5.4.1 基本寻址方式

6. 寄存器间接寻址

- **特征**：操作数地址在指令指定的寄存器中，操作数在主存单元中，访问一次主存即可获得操作数。

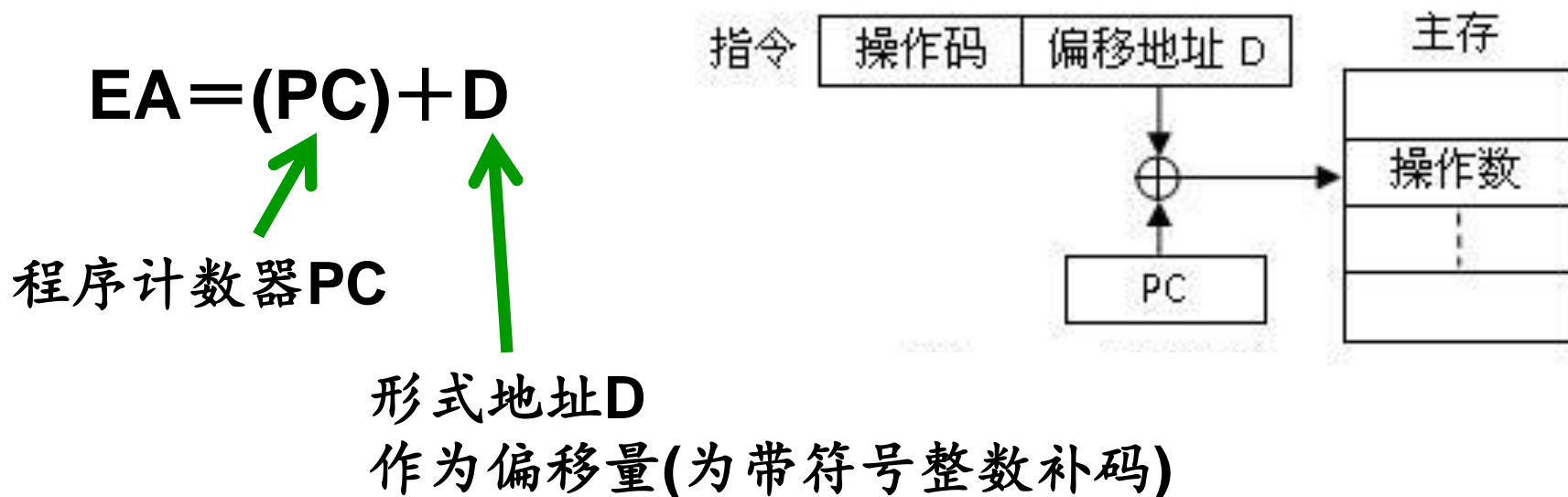


- **优点**：既可有效压缩指令长度，又可较大范围寻址主存；将寄存器作为地址指针，可实现用同一条指令访问不同主存单元的效果，方便循环程序编写。

5.4.1 基本寻址方式

7. 相对寻址

- **特征**：操作数地址由程序计数器和指令提供的地址偏移量决定，操作数在主存单元中，访问一次主存即可获得操作数。



- **优点**：实现“与地址无关的程序设计”。

5.4.1 基本寻址方式

8. 基址寻址

- **特征**：操作数地址由基址寄存器和指令提供的地址偏移量决定，操作数在主存单元中，访问一次主存即可获得操作数。

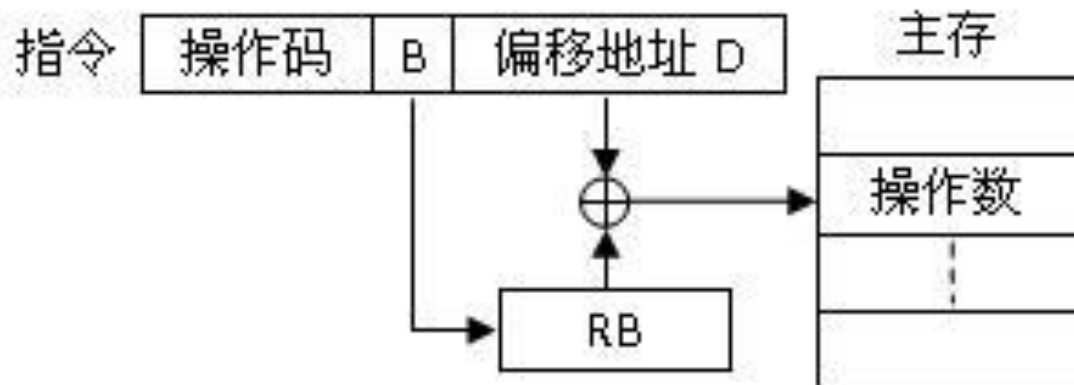
$$EA = (RB) + D$$



基址寄存器

(基址寄存器寻址位B

或寄存器地址选择RB)



- **优点**：既能缩短指令的地址字段长度，又可扩大寻址主存空间。

5.4.1 基本寻址方式

9. 变址寻址

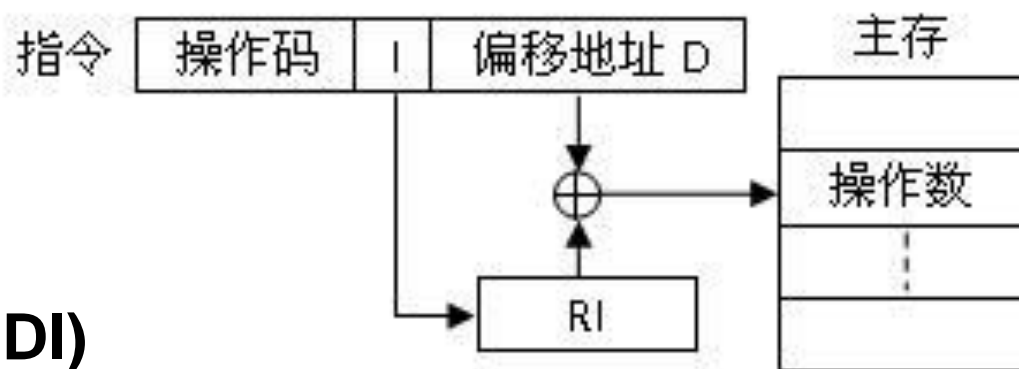
- **特征**：操作数地址由变址寄存器和指令提供的地址偏移量决定，操作数在主存单元中。

$$EA = (RI) + D$$



变址寄存器

寻址位I或地址选择RI(SI或DI)



- **特殊之处**：自动变址功能。
- 相对寻址、基址寻址、变址寻址方式的本质相同，差别仅在提供基准地址的寄存器不同。

5.4.1 基本寻址方式

10. 堆栈寻址

- 该寻址方式通常由指令操作码指定，用在涉及堆栈操作的指令中，所寻址的操作数在堆栈中。
- 堆栈指针SP隐含指定
 - 压栈、调用指令中的目的操作数地址
 - 弹出、返回指令中的源操作数地址

$EA = (SP)$ ，SP自动修改指向新的堆栈单元

- 除了上述基本寻址方式外，某些计算机系统还设计有位寻址、块寻址和页面寻址等方式。



指令设计

--操作码设计

5.3.3 操作码设计

- 操作码是用来指示计算机执行某种操作或完成某种功能的。
- 每一条指令有**唯一**确定的操作码，不同指令的操作码用不同的二进制编码表示。
- 操作码的**编码位数**决定了指令系统的规模或计算机操作的种类。
- 操作码字段的设计主要采用两类编码方式：
 - 操作码长度**固定**
 - 操作码长度**可变**

5.3.3.1 定长操作码

- 对所有指令的操作码用**相同位数**的二进制数进行编码即为定长操作码编码方式。
- 某计算机的指令系统需要设置**N**条指令，若所有指令的操作码均用**n**位二进制数表示，则应满足关系式：

$$N \leq 2^n$$

从 2^n 个编码中选出**N**个编码分配给**N**条指令，即可完成操作码设计。

- 操作码长度固定
 - **好处**：操作码构造简单，有利于简化硬件设计、提高指令译码和后续执行速度；
 - **缺点**：操作码占用指令空间较大，且指令规模（条数）扩充受到限制。
- **RISC**系统一般采用此方式，**RISC-V**，**SPARC**指令系统。

5.3.3.2 变长操作码

- 对不同类型的指令操作码用不固定长度的二进制数进行编码即为变长操作码或扩展操作码编码方式。
- 扩展操作码技术是指令优化技术，其**技术核心**是：
 - 使程序中指令的平均操作码长度尽可能短，以减少操作码在程序中的总位数；
 - 尽可能充分地利用指令的二进制数位，以增加指令字表示的操作信息。
- 扩展操作码的**设计原则**：
 - 如果**指令字长固定**，则长地址码对应短操作码，操作码长度随地址码长度缩短而增加；
 - 如果**指令字长可变**，则以指令使用频度（即在程序中出现的概率）作为设计依据，使用频度高的指令用短操作码，使用频度低的指令用长操作码，这便是霍夫曼编码原理；
 - **设计总是从短操作码开始**，并要保证当前使用的操作码编码与未来要扩展的操作码编码能够**有效地区分**。

5.3.3.2 变长操作码

1. 基于霍夫曼编码原理设计变长操作码

- 统计发现，在程序设计过程中，由于指令功能的不同和设计者对指令使用爱好的不同，致使一个指令系统中的各种指令使用频度大不相同。
- 右表是针对SPECint92测试程序对80x86指令使用情况的统计结果。
- 按照霍夫曼编码原理，使用频率高的指令应采用短操作码。

Rank	80x86 instruction	Integer average (% total executed)
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
Total		96%

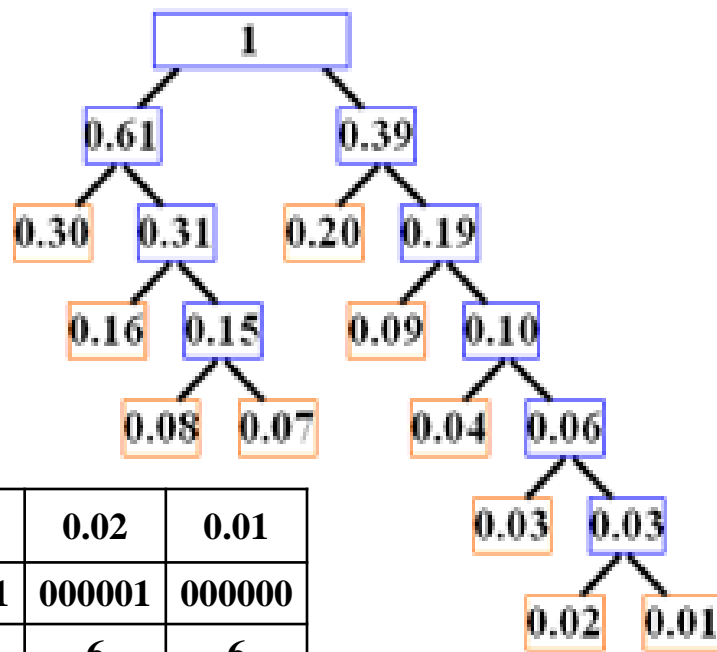
5.3.3.2 变长操作码

例5.2 某计算机有10条指令，它们的使用频率分别为：0.30、0.20、0.16、0.09、0.08、0.07、0.04、0.03、0.02、0.01。用Huffman树（最优二叉树）对它们的操作码进行编码，并计算平均代码长度。

解：右图为基于霍夫曼编码原理构造的Huffman树，

根据Huffman树得出的Huffman编码结果及各编码长度如下：

指令使用频度	0.30	0.20	0.16	0.09	0.08	0.07	0.04	0.03	0.02	0.01
操作码编码	11	01	101	001	1001	1000	0001	00001	000001	000000
编码长度	2	2	3	3	4	4	4	5	6	6



平均代码长度

$$\begin{aligned} &= (0.30 + 0.20) \times 2 + (0.16 + 0.09) \times 3 + (0.08 + 0.07 + 0.04) \times 4 + 0.03 \times 5 + (0.02 + 0.01) \times 6 \\ &= 1 + 0.75 + 0.76 + 0.15 + 0.18 = 2.84 \text{ 位} \end{aligned}$$

5.3.3.2 变长操作码

例5.3 某计算机系统中有20条指令的使用频度是80%，80条指令的使用频度是15%，40条指令的使用频度是5%，试设计固定长度和可变长度的操作码。

解：（1）定长操作码：140条指令，8位操作码长度，从256种编码中选出140种编码分配给140条指令作为它们的操作码。

（2）扩展操作码：一种设计结果为

频度	条数	操作码编码			扩展位数应满足的关系
80%	20	00000			$20 < 2^5$
		↓			
		10011			
15%	80	10100	000		$80 < (2^5 - 20) \times 2^3$
		↓			
		11101	111		
5%	40	11110	000	00	$40 < (2 \times 8) \times 2^2$
		↓			
		11111	001	11	

平均操作码长度 = $5 \times 80\% + 8 \times 15\% + 10 \times 5\% = 5.7$ (位)

5.3.3.2 变长操作码

2. 基于特定规则扩展操作码

- 设计扩展操作码不仅要考虑操作码的可分辨性，还应考虑其是否便于快速译码。图5.9示意了两种以某种特定规则设计的扩展操作码。

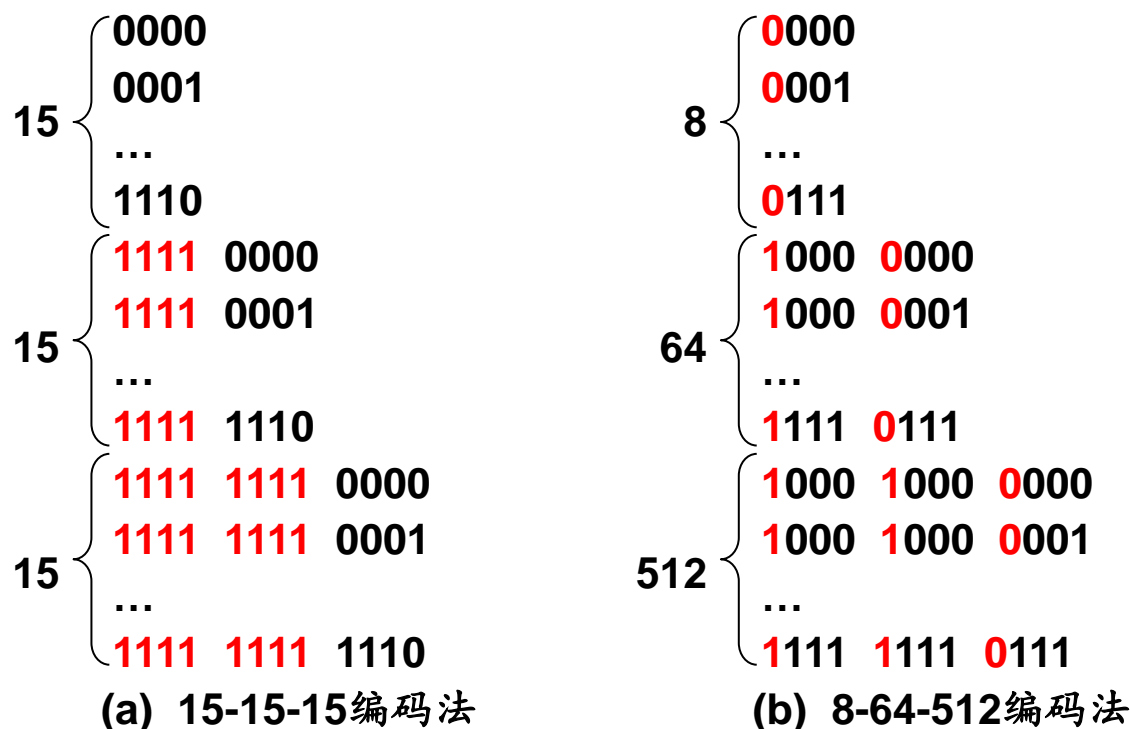


图5.9 特定规则的扩展操作码编码方法

5.3.3.2 变长操作码

3. 依据地址码数量扩展操作码

- 对于**指令字长度固定**的指令系统，为充分利用指令的二进制位，操作码长度可随地址码数量多少而变化。

例5.4 某指令系统的指令长度确定为32位，由三地址、两地址、一地址、零地址指令组成，其中各类指令中地址字段位数如下表，问各类指令的操作码可以设计为几位？各类指令数最多可以是多少？

指令格式	操作码	地址1 (5位)	地址2 (5位)	地址3 (16位)
三地址指令	操作码 (?)			
二地址指令	操作码 (?)			
一地址指令	操作码 (?)			
零地址指令	操作码 (?)			

5.3.3.2 变长操作码

解: 三地址指令: 操作码6位, 指令数 $n_3 \leq 2^6 - 1$

双地址指令: 操作码11位, 指令数 $n_2 \leq (2^6 - n_3) \times 2^5 - 1$

单地址指令: 操作码16位,

指令数 $n_1 \leq ((2^6 - n_3) \times 2^5 - n_2) \times 2^5 - 1$

零地址指令: 操作码32位,

指令数 $n_0 \leq (((2^6 - n_3) \times 2^5 - n_2) \times 2^5 - n_1) \times 2^{16}$

指令格式	操作码	地址1 (5位)	地址2 (5位)	地址3 (16位)
三地址指令	操作码 (6)			
二地址指令	操作码 (11)			
一地址指令	操作码 (16)			
零地址指令	操作码 (32)			



指令设计

--指令长度设计

5.3.4 指令长度设计

➤ 指令长度设计的一般原则：

- 短的操作码与多地址码字段配合，长的操作码与简单地址码组合；
- 指令长度一般设计为总线宽度的整数倍；
- 指令长度为存储器最小可寻址单位的整数倍。

➤ 一条指令的长度可固定，也可变化。

- **定长指令的好处**：规则，便于指令获取；**缺点**：定长指令的长度是按最长的指令来设计的，所以指令中会出现许多二进制数位闲置而造成浪费的状况。
- **变长指令的好处**：可按指令的实际需要设计指令的长度，使指令中的每一位都尽可能成为有用信息；**缺点**：增加了指令获取、处理的复杂度。

5.3.4 指令长度设计

- **理论**上，可以按下式确定指令长度，即

$$\text{指令长度} = \text{操作码长度} + \sum_{i=1}^M \text{第} i \text{段地址码长度} \quad (5-2)$$

式中假设指令中有M个地址码字段，每个地址码字段的长度可以不相同。

- **实际**采取的指令长度**设计准则**：指令长度应为式(5-2)计算结果向上取**字节的整数倍**。
- 指令长度设计的**另一准则**是：在满足操作种类、寻址范围和寻址方式的前提下，指令应**尽可能短**。这是指令功能完备性与有效性的统一。
- 指令的长度与机器的字长没有固定的关系。

5.3.4 指令长度设计

- 在实际计算机系统中，有三种常用设计方案：
- **定长操作码，变长指令码**。操作码长度固定，且集中放在指令字的第一个字段中，指令的其余部分全部用于地址码。例如**8086**指令系统，指令采用**8位定长操作码**、**1~5字节变长指令码**的格式。
 - **变长操作码，定长指令码**。即在指令码长度固定的情况下，让操作码位数随地址码长度的减少而扩展，如例**5.4**所示。
 - **定长操作码，定长指令码**。例如**RISC-V**指令系统，指令采用**7位定长操作码**、**32位定长指令码**的格式。

某计算机有A、B两类指令。其中，A类指令共5条，使用频度为80%；B类指令共12条，使用频度为20%。试用扩展操作码编码方案，为两类指令分别确定操作码长度及编码，要求平均码长尽量短，并计算平均码长。

答案解析

扩展操作码设计：

设A类指令操作码长度为 x ，求得满足 $2^x > 5$ 的最小 $x=3$ 。

A类指令码长=3。

指令编码可选为000~100。

设B类指令操作码需在上述每个3位前缀的基础上扩展 y 位，求满足 $(2^3-5) \cdot 2^y \geq 12$ 的最小 y ，即 $3 \cdot 2^y \geq 12$ ， $y=2$ 。

B类指令码长=3+2=5。

指令编码为10100~11111

平均码长=3×0.8+5×0.2=3.4。