计算机系统结构
第二章 指令系统

主　讲：刘 超
中国地质大学（武汉）计算机学院

第二章 指令系统

- 2.1 指令系统结构的分类
- 2.2 寻址技术
- 2.3 指令格式的优化设计
- 2.4 指令系统的功能设计
- 2.5 指令系统的发展方向和优化
- 2.6 小结
- 2.7 习题

2.3 指令格式的优化设计

- 指令格式优化主要目标和研究内容
- 指令的组成
- 操作码的优化
- 指令字格式的优化
- 指令格式设计举例

1) 指令格式优化的主要目标和研究内容

- **指令格式的优化设计：是指如何用最短的二进制位数来表示指令的操作信息和地址信息，使指令的平均字长最短。**
- **主要目标**
  - 节省程序的存储空间
  - 指令格式尽量规整，减少硬件译码的难度
- **研究内容**
  - 操作码的优化表示
  - 地址码的优化表示

2) 指令的组成

- **一般的指令主要由两部分组成：操作码和地址码**
- **操作码主要包括两部分内容：**
  - **操作种类**：加、减、乘、除、数据传送、移位、转移、输入输出
  - **操作数的数据类型**：
    - 定点数、浮点数、复数、字符、字符串、逻辑数、向量等
    - 若采用自定义数据表示法，则操作码不必指出操作数的数据类型
- **地址码通常包括三部分内容：**
  - **操作数的地址**：直接地址、间接地址、立即数、寄存器编号、变址寄存器编号
  - **地址的附加信息**：如偏移量、块长度、跳距等
  - **寻址方式**：直接寻址、间接寻址、立即数寻址、变址寻址、相对寻址、寄存器寻址

3) 操作码的优化表示

操作码的优化编码方法通常有三种：

- ① 定长操作码；
- ② Huffman编码；
- ③ 扩展编码。

① 定长操作码

- 所有指令的操作码长度都是相等的。
- 规整简单，但浪费空间。

  如果操作码有n个

  那么操作码的位数至少应该有

  $l = \lceil Log_2 n \rceil$位
- 例：已知 操作码的个数$n = 15$，求定长编码的最小平均码长。
- 解：

$$l = \lceil Log_2 15 \rceil = 4$$

② Huffman编码

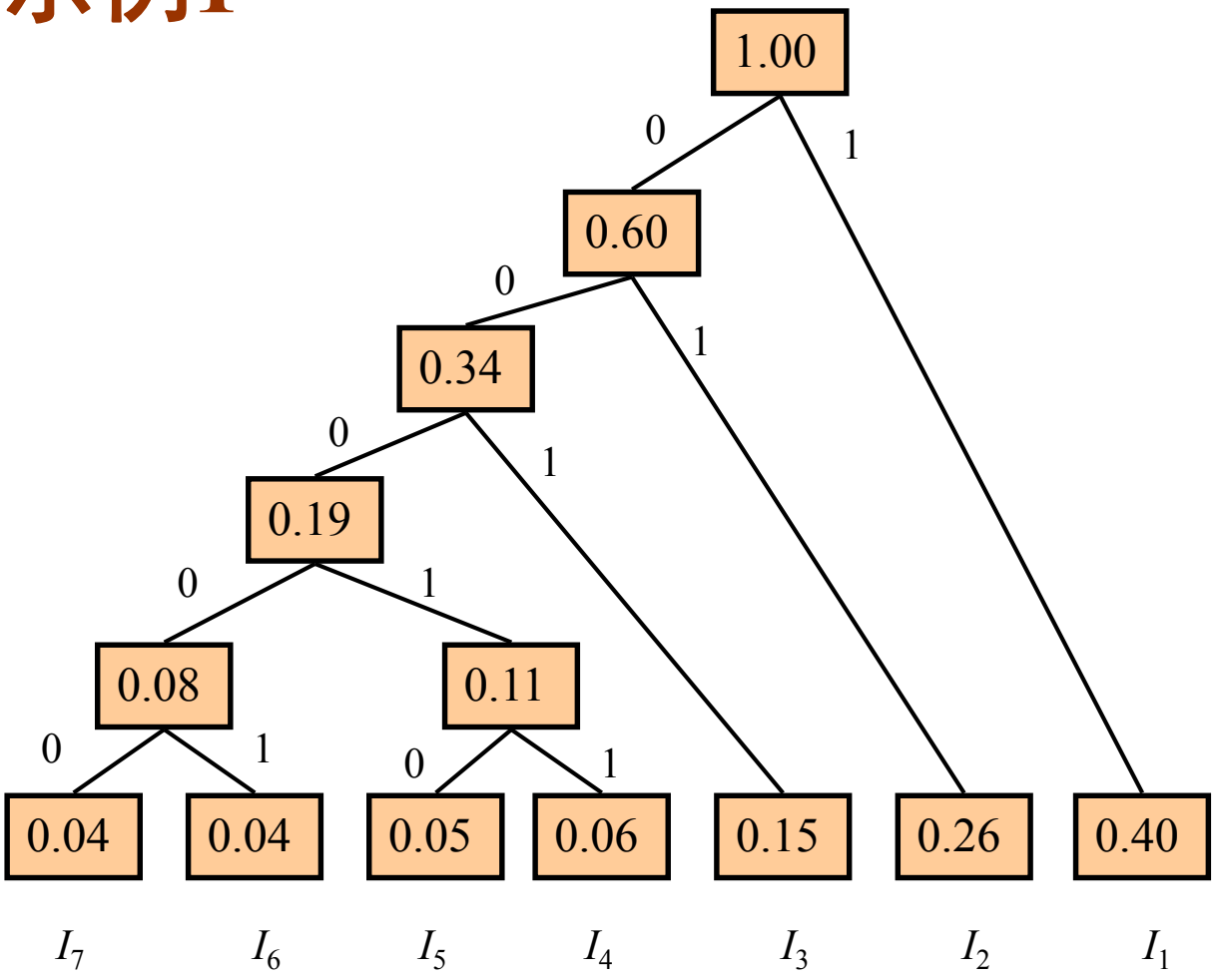
- 对使用频度最高的对象，用最短位数的编码表示，对使用最低的对象，用最长位数的编码表示。
- 构造方法：使用哈夫曼树来构造

利用Huffman树进行操作码编码

- **利用Huffman树进行操作码编码的方法，又称为最小概率合并法。**
  - 将各事件按其使用频度从小到大依次排列；
  - 每次从中选择两个频度值最小的结点，将其合并成一个新的结点，并把新结点画在所选结点的上面，
  - 然后用两条边把新结点分别与那两个结点相连。
    - 新结点的频度值是所选两个结点的频度值的和。
  - 把新结点与其他剩余未结合的结点一起，再以上面的步骤进行处理，反复进行，直到全部结点都结合完毕、形成根结点为止。

Huffman编码示例1

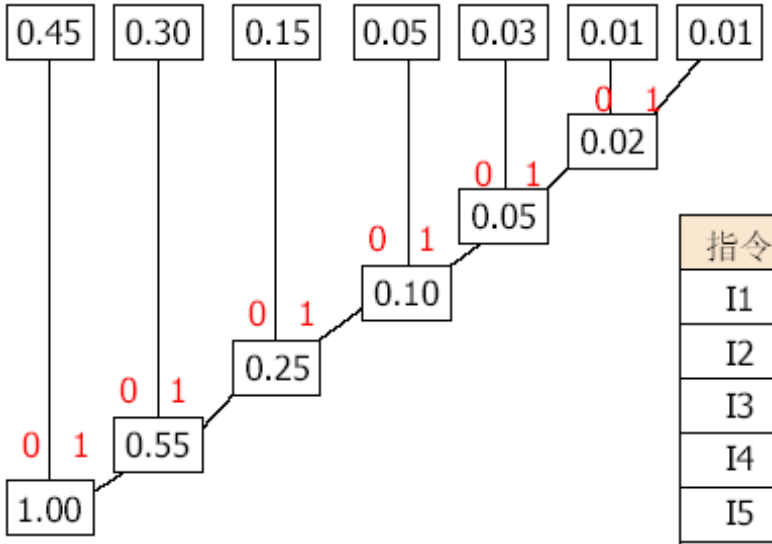
指令	频率	Huffman编码	长度
I_1	0.40	1	1
I_2	0.26	01	2
I_3	0.15	001	3
I_4	0.06	00011	5
I_5	0.05	00010	5
I_6	0.04	00001	5
I_7	0.04	00000	5



Huffman编码示例2

- 计算机具有7种操作码。固定长编码需要3位二进制。如何采用 Huffman编码。

指令	概率
I1	0.45
I2	0.30
I3	0.15
I4	0.05
I5	0.03
I6	0.01
I7	0.01



指令	概率	Huffman编码	编码长度
I1	0.45	0	1
I2	0.30	10	2
I3	0.15	110	3
I4	0.05	1110	4
I5	0.03	11110	5
I6	0.01	111110	6
I7	0.01	111111	6

注：

- **哈夫曼树构造过程可以总结为：**

> 从小到大排序，
> 最小两个合并，
> 重复上述过程，
> 只剩一个结束。

- **哈夫曼树不是唯一的**(因为相同的频率可以任取一个在前，且编码时又可任取左1或左0)，但所得的平均码长应是一样的。

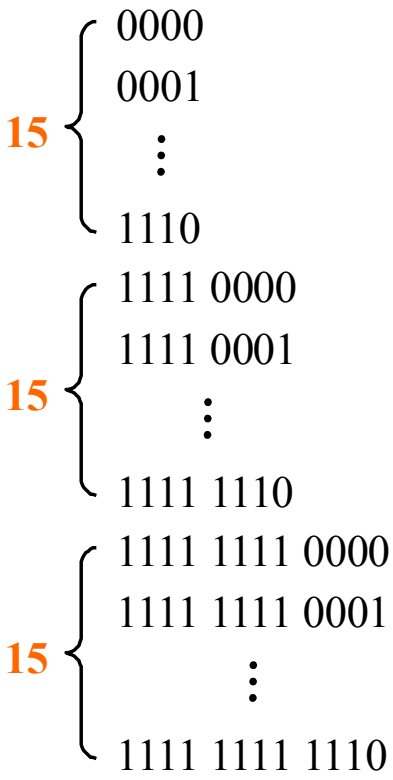
③ 扩展编码

- **Huffman编码有优点，但缺点也比较明显：**
  - 优点：平均信息长度短，冗余量小。
  - 缺点：操作码不规整，码长种类较多，不利于硬件的译码，也不利于软件的编译，另外，也很难与地址码配合形成长度规整的指令编码。
- **改进：扩展编码法**
  - 由定长操作码与Huffman编码法相结合形成的折中方案
  - 主要思想：限定几种码长，使用频度高的用短码，使用频度低的用长码，短码不能是长码的前缀。

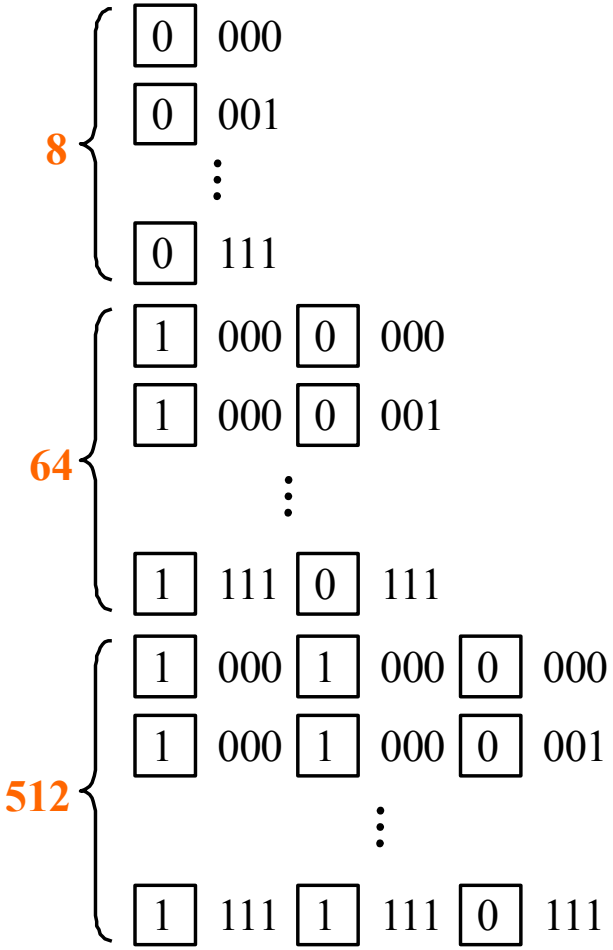
扩展编码的表示方法

- **用码长表示：用横线隔开不同码长，例如4-8-12法。**
- **用码点数表示：例如15/15/15法，8/64/512法**
  - 15/15/15法，每一种码长都有4位可编码位（前头可以有相同的扩展标识前缀），可产生16个码点（即编码组合），但是至多只能使用其中15个来表示事件，留下1个或多个码点组合作为更长代码的扩展标识前缀。已经用来表示事件的码点组合不能再作为其它更长代码的前导部分，否则接收者会混淆。这就是“非前缀原则”。
  - 8/64/512法，每一种码长按4位分段，每一段中至少要留下1位或多位作为扩展标识。各段剩下的可编码位一起编码，所产生的码点用来对应被编码事件。每一段中的标识位指出后面还有没有后续段。

扩展编码的表示方法



15/15/15 编码法



8/64/512 编码法

扩展编码的分类

- **等长编码法**是对出现频率较低的操作码用较长的编码表示时，每次扩展的编码位数相等，如2-4-6扩展法是指每次扩展时加长2位。
- **不等长编码法**是指每次扩展的编码位数不相等，如1-2-3-5扩展法的前两次扩展都加长1位，第3次扩展加长2位。

示例：扩展编码

I_i	P_i	1 - 2 - 3 - 5 编 码	2-4编码（3/4）	2-4编码（2/8）
I1	0.45	0	00	00
I2	0.30	10	01	01
I3	0.15	110	10	1000
I4	0.05	11100	1100	1001
I5	0.03	11101	1101	1010
I6	0.01	11110	1110	1011
I7	0.01	11111	1111	1100

编码方法性能指标

- **平均码长*l***

$$l = \sum_{i=1}^{n} (P_i \cdot l_i)$$

P_i是操作码i的使用概率

l_i是操作码i的长度

n为操作码个数

- **信息熵H（Entropy）：表示用二进制编码表示n个码点时，理论上的最短平均码长。任何实际编码得到的平均码长*l*都大于H。**

$$H = -\sum_{i=1}^{n} [P_i \cdot \log_2(P_i)]$$

- **信息冗余量：表明消息编码中“无用成分”所占的百分比，用来衡量代码优化的程度。**

$$R = \frac{l - H}{l} = (1 - \frac{H}{l})$$

例：定长编码的信息冗余量

- 定长编码的平均码长为

$$l = \lceil \log_2 n \rceil$$

- 定长编码的信息冗余量为

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

例：编码方法性能比较

- 某计算机有7种不同的操作码，其使用概率如图所示，试比较定长编码、 Huffman编码、1-2-3-5不等长扩展编码、2-4(3/4)等长扩展编码、2-4(2/8)等长扩展编码的熵以及平均码长、信息冗余量。

指令	概率
I1	0.45
I2	0.30
I3	0.15
I4	0.05
I5	0.03
I6	0.01
I7	0.01

解：

- 熵H=

–

$$(2\times 0.01\times \log_2 0.01+0.03\times \log_2 0.03+0.05\times \log_2 0.05+0.15\times \log_2 0.15+0.3\times \log_2 0.3+0.45\times \log_2 0.45)$$

$$\approx 1.95$$

解（续）：

- 定长编码的平均码长

$$l = \lceil \log_2 7 \rceil = 3$$

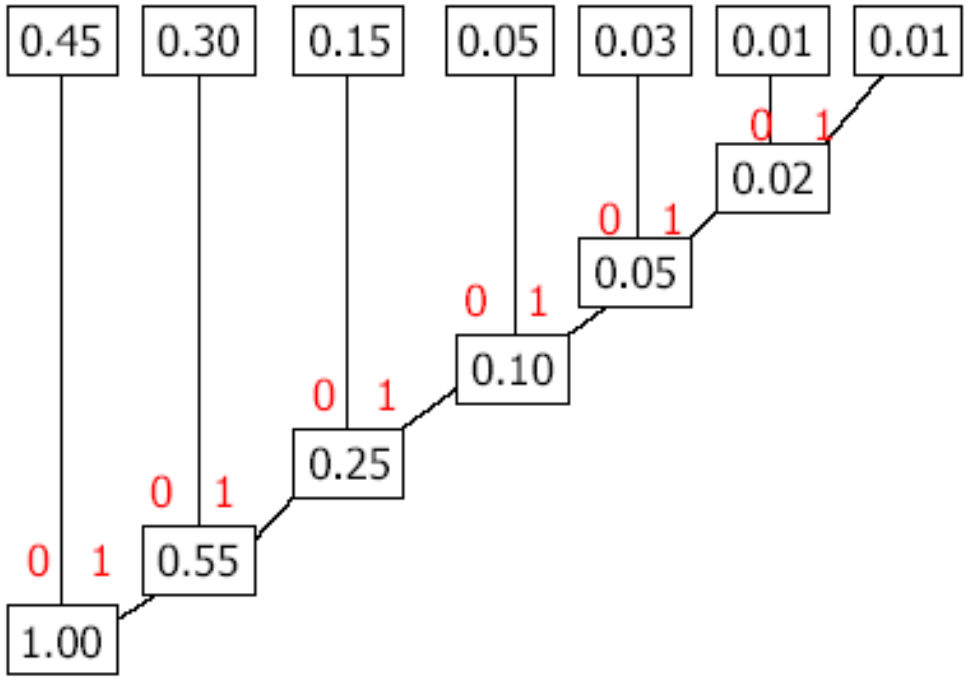
- 定长编码的信息冗余量

$$R = \frac{l - H}{l} = 1 - \frac{H}{l} = 1 - \frac{1.95}{3} \approx 35\%$$

解（续）：

- **Huffman编码**

指令	概率
I1	0.45
I2	0.30
I3	0.15
I4	0.05
I5	0.03
I6	0.01
I7	0.01



解（续）：

指令	概率	Huffman编码	编码长度
I1	0.45	0	1
I2	0.30	10	2
I3	0.15	110	3
I4	0.05	1110	4
I5	0.03	11110	5
I6	0.01	111110	6
I7	0.01	111111	6

- **Huffman编码的平均码长**

$$l = \sum_{i=1}^{n} (P_i \cdot l_i) =$$

$$0.45 \times 1 + 0.30 \times 2 + 0.15 \times 3 + 0.05 \times 4 + 0.03 \times 5 + 0.01 \times 6 + 0.01 \times 6$$

$$= 1.97$$

- **Huffman编码的信息冗余量**

$$R = \frac{l - H}{l} = (1 - \frac{H}{l}) = (1 - \frac{1.95}{1.97}) = 1.0\%$$

- **可见， Huffman编码明显优于定长编码**

解（续）：

- 扩展编码

I_i	P_i	1-2-3-5编码	2-4编码（3/4）	2-4编码（2/8）
I1	0.45	0	00	00
I2	0.30	10	01	01
I3	0.15	110	10	1000
I4	0.05	11100	1100	1001
I5	0.03	11101	1101	1010
I6	0.01	11110	1110	1011
I7	0.01	11111	1111	1100

解（续）：

- **采用1-2-3-5不等长扩展编码的操作码平均长度为：**
  - l=(0.45×1+0.30×2+0.15×3+(0.05+0.03+0.01+0.01) ×5 =2.00位
  - 信息冗余量为：R =1-1.95/2.00=2.5%
- **采用2-4(3/4)等长扩展编码的操作码平均长度为：**
  - l=(0.45+0.30+0.15)×2+(0.05+0.03+0.01+0.01)×4 =2.20位
  - 信息冗余量为：R =1-1.95/2.20=11.4%
- **采用2-4(2/8)等长扩展编码的操作码平均长度为：**
  - l=(0.45+0.30)×2+(0.15+0.05+0.03+0.01+0.01)×4 =2.50位
  - 信息冗余量为：R =1-1.95/2.50=22%
- **由上可知， 1-2-3-5不等长扩展编码优于后两种等长扩展编码。**

2.6 小结

- **本章主要内容有指令优化、指令系统发展方向（CISC，RISC）两个部分。具体细节如下：**
- **指令优化**
  - 操作码优化
    - 定长编码、Huffman编码、扩展编码方法
    - 编码方法性能指标（熵H，平均码长L，信息冗余量R）
  - 操作数优化
    - 地址表示形式、寻址方式、地址制；
    - 如何使操作数地址部分和操作码配合，使指令最短、最有效
- **指令系统发展方向**
  - CISC、RISC定义；CISC的缺点及RISC的由来；
  - 设计RISC的一般原则
  - 设计RISC机器的关键技术