# 第二章　指令系统原理与实例

- 2.1 简介
- 2.2 指令集系统结构的分类
- 2.3 存储器寻址
- 2.4 操作数的类型
- 2.5 指令系统的操作
- 2.6 控制流指令
- 2.7 指令系统的编码
- 2.8 编译器的角色
- 2.9 MIPS系统结构
- 2.10 谬误和易犯的错误
- 2.11 结论

# 2.7指令系统的编码

前面讨论的问题都会直接影响指令如何编码。

- 指令编码包括：操作、寻址方式编码。
- 指令编码 ->处理器实现和编译后程序大小。
- 如何将寻址方式和操作通过编码结合到一起：

  *如图A.6，1-5个操作数，10种可能寻址方式，则每个操作数需要一个独立寻址标识符字段（显式）。

  * 如load/store结构，一个存储器操作数，1-2种寻址方式，则寻址方式由操作编码隐含表示。

- 对指令编码，需要在以下因素之间找到一个最佳平衡点：

  1、尽可能多的寄存器和寻址方式。

  2、寄存器字段、寻址方式字段尽量少，以缩短指令长度。

  3、指令长度易于流水线处理。

# 哈夫曼编码

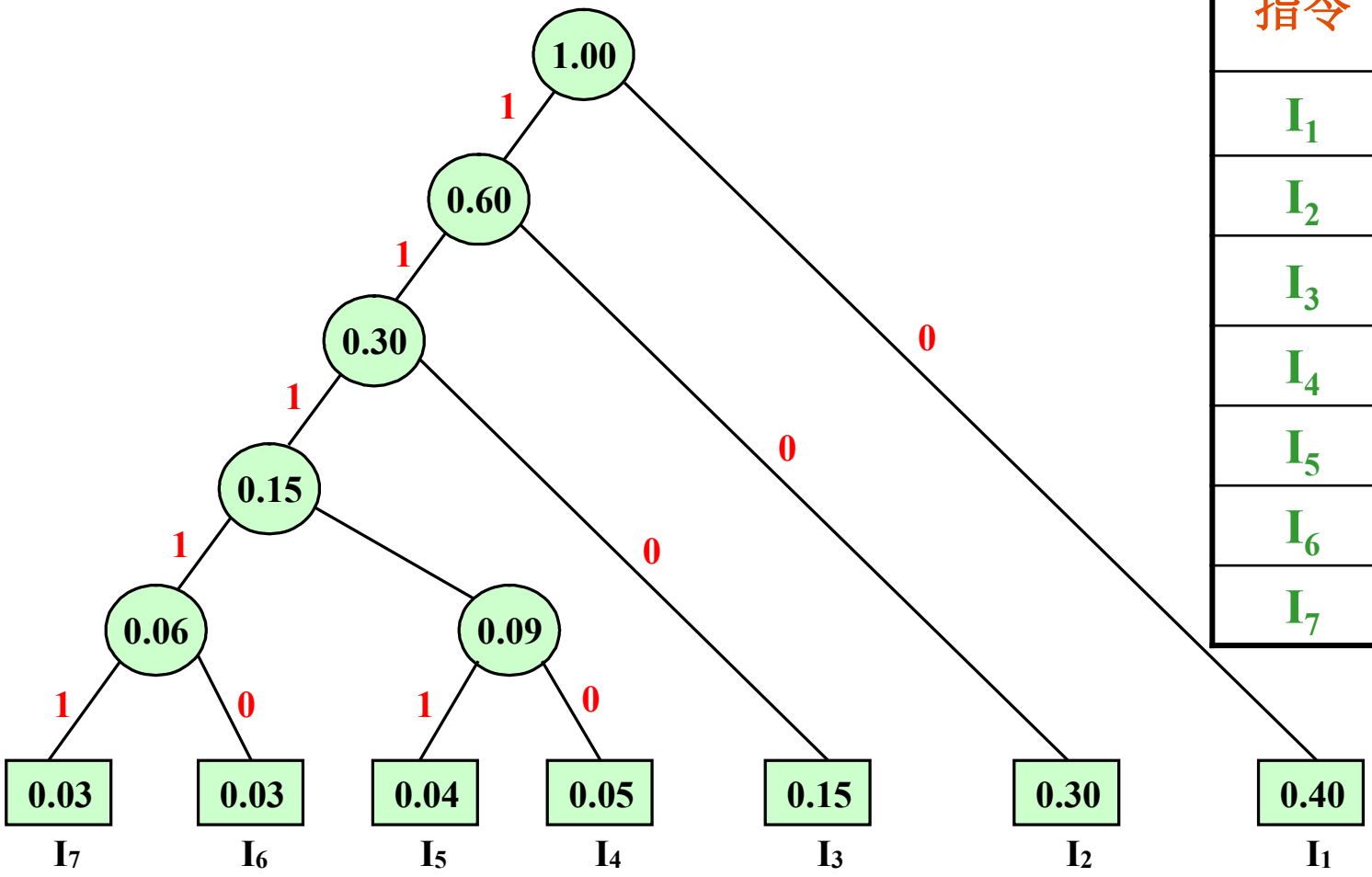
- 基本思想：当各种事件发生的概率不均等时，可以对发生概率最高的事件用最短的位数（时间）来表示（处理），而对于出现概率较低的事件，则可以用较长的位数（时间）来表示（处理），从而使总的平均位数（时间）缩短。

构造哈夫曼树的方法：

- 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。

# 例

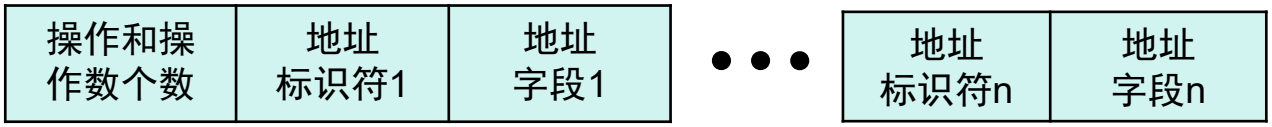
- 假设某模型机有7条指令，这些指令的使用频度如下




| 指令 | 频度$p_i$ |
|---|---|
| $I_1$ | 0.40 |
| $I_2$ | 0.30 |
| $I_3$ | 0.15 |
| $I_4$ | 0.05 |
| $I_5$ | 0.04 |
| $I_6$ | 0.03 |
| $I_7$ | 0.03 |

# 2.7指令系统的编码

- 三种常见编码方式　Figure A.18




（a）变长编码（如VAX和Intel 80x86）

- **变长编码**
  - 当指令系统的寻址方式和操作种类很多时，这种编码格式是最好的。
  - 用最少的二进制位来表示目标代码。
  - 可能会使各条指令的字长和执行时间相差很大。

# 2.7指令系统的编码

## 三种常见编码方式　Figure A.18

| 操作 | 地址字段1 | 地址字段2 | 地址字段3 |
|---|---|---|---|

（b）定长编码（如Alpha，ARM，MIPS，PowerPC，SPARC，SuperH）

## 定长编码

- 将操作类型和寻址方式一起编码到操作码中。
- 当寻址方式和操作类型非常少时，这种编码格式非常好。
- 可以有效地降低译码的复杂度，提高译码的速度。
- 大部分RISC的指令系统均采用这种编码格式。

# 2.7指令系统的编码

## 三种常见编码方式　Figure A.18

| 操作 | 地址标识符 | 地址字段 |
|---|---|---|

| 操作 | 地址标识符1 | 地址标识符2 | 地址字段 |
|---|---|---|---|

| 操作 | 地址标识符 | 地址字段1 | 地址字段2 |
|---|---|---|---|

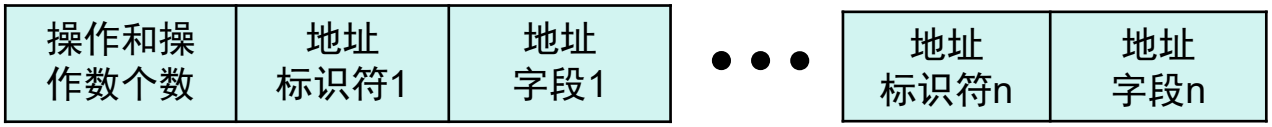
（c）混合编码（如IBM360/370，MIPS16，Thumb, TI TMS320C54x）

## 混合编码

- 提供若干种固定的指令字长。
- 以期达到既能够减少目标代码长度又能降低译码复杂度的目标。

# 2.7指令系统的编码

- 三种常见编码方式　Figure A.18




（a）变长编码（如VAX和Intel 80x86）


（b）定长编码（如Alpha，ARM，MIPS，PowerPC，SPARC，SuperH）


（c）混合编码（如IBM360/370，MIPS16，Thumb, TI TMS320C54x）

# 2.7指令系统的编码

- 变长编码：允许所有的操作使用所有的寻址方式，适合寻址方式和操作比较多的情形。最少的位数表示程序，译码复杂。
- 定长编码：把操作和寻址方式组合在操作码里，通常所有的指令长度都相同。这种方式适合于寻址方式和操作比较少的情况。译码简单，代码量大。
- 混合方法，减少过多的指令，以减轻多种结构的指令带来的工作负担，但仍提供多种指令长度以减少代码长度。

# Fixed-Length versus Variable-Length Encodings

## Schaum 5.11

**A given processor has 32 registers, uses 16-bit immediates, and has 142 instructions in its ISA. In a given program, 20 percent of the instructions take one input register and have one output register, 30 percent have two input registers and one output register, 25 percent have one input register, one output register, and take an immediate input as well, and the remaining 25 percent have one immediate input and one output register.**

**a. For each of the four types of instructions, how many bits are required? Assume that the ISA requires that all instructions be a multiple of 8 bits in length.**

**b. How much less memory does the program take up if a variable-length instruction set encoding is used as opposed to a fixed-length encoding?**

# Fixed-Length versus Variable-Length Encodings

## Schaum 5.11

A given processor has 32 registers, uses 16-bit immediates, and has 142 instructions in its ISA. In a given program, 20 percent of the instructions take one input register and have one output register, 30 percent have two input registers and one output register, 25 percent have one input register, one output register, and take an immediate input as well, and the remaining 25 percent have one immediate input and one output register.

a. For each of the four types of instructions, how many bits are required? Assume that the ISA requires that all instructions be a multiple of 8 bits in length.

b. How much less memory does the program take up if a variable-length instruction set encoding is used as opposed to a fixed-length encoding?

## Solution

**With 142 instructions, 8 bits are required to determine which instruction an instruction is (128 < 142 < 256). 32 registers means that 5 bits are required to encode register ID, and we know that 16 bits are required for each immediate. Given that, it's just a matter of adding up the fields required for each type of instruction.**

**One register input, one register output: 8 + 5 + 5 bits = 18 bits, which rounds up to 24.**

**Two input registers, one output register: 8 + 5 + 5 + 5 = 23 bits, which rounds up to 24.**

**One input register, one output register, and an immediate: 8 + 5 + 5 + 16 bits = 34 bits, which rounds up to 40 bits.**

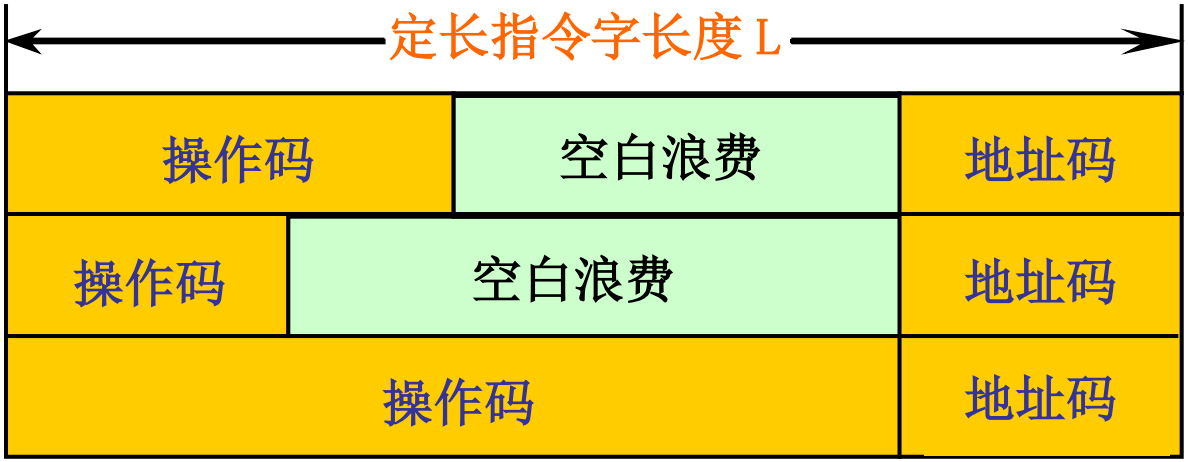
**One input immediate, one output register: 8 + 16 + 5 bits = 29 bits, which rounds up to 32 bits.**

**Since the largest instruction type requires 40-bit instructions, the fixed-length encoding will have 40 bits per instruction.**

**Each instruction type in the variable encoding will use the number of bits given in Part a. To find the average number of bits per instruction in the variable-length encoding, we multiply the number of bits for each instruction type by that type's frequency and add the results. This gives 20% * 24 bits + 30% * 24 bits + 25% * 40 + 25% * 32 = 4.8 + 7.2 + 10 + 8 = 30 bits on average. Therefore, the variable-length encoding requires 25 percent less space than the fixed-length encoding for this program.**

# 2.7指令系统的编码

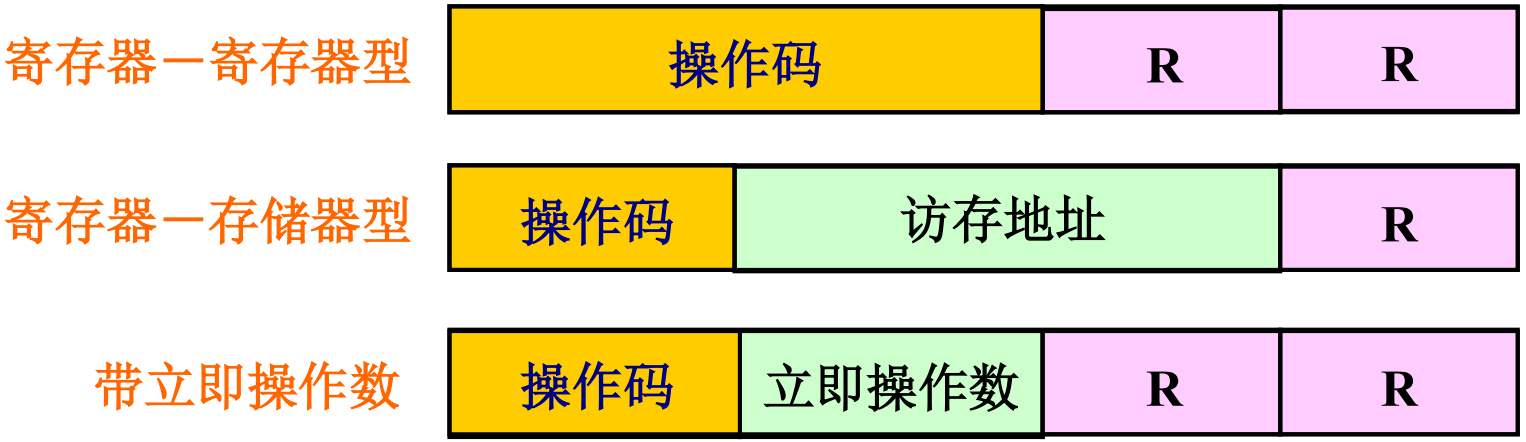
- 定长编码：如果指令字的宽度、地址码的长度和个数都固定，则操作码的缩短并不能带来好处，会使指令字中出现空白浪费。如下图所示。



# 2.7指令系统的编码

- **定长编码：采用地址个数可变和/或地址码长度可变的方案**
  - 利用操作码缩短所带来的好处
  - 最常用的操作码最短，其地址字段个数最多。

    能够使指令的功能增强，从总体上减少所需的指令条数 。



# 2.7指令系统的编码

- 嵌入式应用中RISC指令系统的变化

  *ARM Thumb,MIPS 16---要求代码量小，混合编码方式，包括16位和32位指令。较短的指令支持较少的操作，较小的地址和较少的寄存器，同时还放弃了典型RISC的三地址指令格式而采用两地址格式。

  *IBM CodePack---将标准指令压缩，压缩的指令留在存储器、ROM和磁盘中。添加新硬件来解压，指令Cache保存解压的32位指令。编译器无须修改就可以处理不同的指令系统，而且指令的解压也很容易。

  *SuperH---为配合较短的指令格式和较少的操作，这种指令系统只有16个寄存器而不是32个，但其它方面均与典型的RISC系统类似。

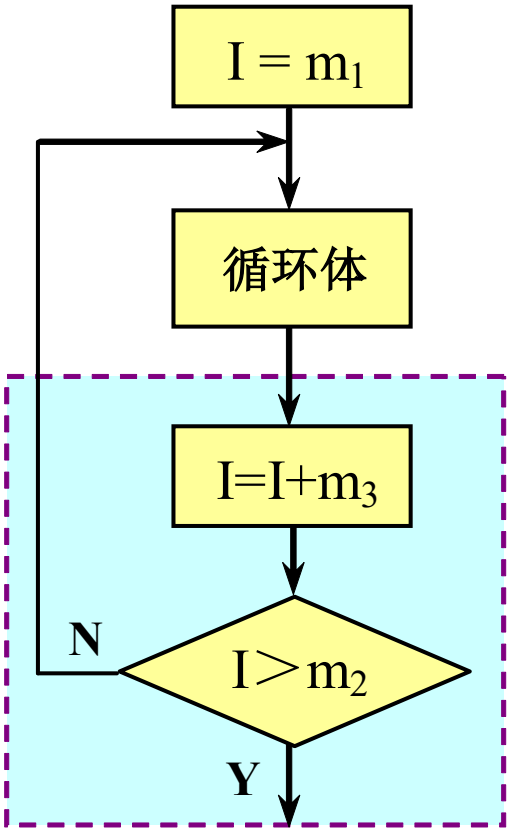
# 指令系统的发展和改进

- **增强指令功能主要是从以下3个方面着手：**
  - 面向目标程序增强指令功能
  - 面向高级语言的优化实现来改进指令系统
  - 面向操作系统的优化实现改进指令系统

# 面向目标程序增强指令功能

**例如：循环在程序中占有相当大的比例，所以在指令上提供专门的支持。**

- 循环控制部分通常用3条指令完成：
  - 一条加法指令
  - 一条比较指令
  - 一条分支指令
- 设置循环控制指令，用一条指令完成上述3条指令的功能。




一般循环程序的结构

# 面向高级语言的优化实现来改进指令系统

- 对高级语言中使用频度高、执行时间长的语句，增强有关指令的功能，加快这些指令的执行速度，或者增加专门的指令，可以达到减少目标程序的执行时间和减少目标程序长度的目的。
- 增强系统结构的规整性，减少系统结构中的各种例外情况。

# 面向操作系统的优化实现改进指令系统

- 操作系统和计算机系统结构是紧密联系的，操作系统的实现在很大程度上取决于系统结构的支持。

- 指令系统对操作系统的支持主要有：
- 处理机工作状态和访问方式的切换；
- 进程的管理和切换；
- 存储管理和信息保护；
- 进程的同步与互斥，信号灯的管理等。

- 支持操作系统的有些指令属于特权指令，一般用户程序是不能使用的。

# 2.7指令系统的编码

- 总结
  - * 关注代码量大小，选择变长编码。
  - * 关注程序的执行性能，选择定长编码。
  - * 折中可以选择混合编码。

# Exercise A.8

- 考虑以下情景：处理器的指令长度为12位，有32个通用寄存器， 所以寄存器地址字段的大小为 5位。是否有可能拥有如下指令编码？
  - 3个两地址指令
  - 30个单地址指令
  - 45个零地址指令

- 考虑以下情景：处理器的指令长度为12位，有32个通用寄存器， 所以寄存器地址字段的大小为 5位。是否有可能拥有如下指令编码？
  - 3个两地址指令
  - 31个单地址指令
  - 35个零地址指令

- 考虑以下情景：处理器的指令长度为12位，有32个通用寄存器， 所以寄存器地址字段的大小为 5位。假定已经拥有3个两地址指令和24个零地址指令。最多可以为这一处理器编码多少个单地址指令？

# Exercise A.8

- Consider the case of a processor with an instruction length of 12 bits and with 32 general-purpose registers so the size of the address fields is 5 bits. Is it possible to have instruction encodings for the following?
  - 3 two-address instructions
  - 30 one-address instructions
  - 45 zero-address instructions

- Assuming the same instruction length and address field sizes as above, determine if it is possible to have
  - 3 two-address instructions
  - 31 one-address instructions
  - 35 zero-address instructions

- Assume the same instruction length and address field sizes as above. Further assume there are already 3 two-address and 24 zero-address instructions. What is the maximum number of one-address instructions that can be encoded for this processor?

# Exercise A.8(a)

- **Consider the case of a processor with an instruction length of 12 bits and with 32 general-purpose registers so the size of the address fields is 5 bits. Is it possible to have instruction encodings for the following?**
  - **3 two-address instructions**
  - **30 one-address instructions**
  - **45 zero-address instructions**

- **Solution: Possible.**

| Instruction | addr[11:10] | addr[9:5] | addr[4:0] |
|---|---|---|---|
| 3 two-address instr. | ‘00’, ‘01’, ‘10’ | ‘00000’ to ‘11111’ | ‘00000’ to ‘11111’ |
| 30 one-address instr. | ‘11’ | ‘00000’ to ‘11101’ | ‘00000’ to ‘11111’ |
| 45 zero-address instr. | ‘11’ | ‘11110’ | ‘00000’ to ‘11111’ |
| | | ‘11111’ | ‘00000’ to ‘01100’ |

# Exercise A.8(b)

- **Consider the case of a processor with an instruction length of 12 bits and with 32 general-purpose registers so the size of the address fields is 5 bits. Determine if it is possible to have**
  - **3 two-address instructions**
  - **31 one-address instructions**
  - **35 zero-address instructions**

- **Solution: Impossible.**

| Instruction | addr[11:10] | addr[9:5] | addr[4:0] |
|---|---|---|---|
| 3 two-address instr. | ‘00’, ‘01’, ‘10’ | ‘00000’ to ‘11111’ | ‘00000’ to ‘11111’ |
| 31 one-address instr. | ‘11’ | ‘00000’ to ‘11110’ | ‘00000’ to ‘11111’ |
| 32 zero-address instr. | ‘11’ | ‘11111’ | ‘00000’ to ‘11111’ |

# Exercise A.8(c)

- **Consider the case of a processor with an instruction length of 12 bits and with 32 general-purpose registers so the size of the address fields is 5 bits.**
- **Further assume there are already 3 two-address and 24 zero-address instructions. What is the maximum number of one-address instructions that can be encoded for this processor?**

- **Solution: 31**

| Instruction | addr[11:10] | addr[9:5] | addr[4:0] |
|---|---|---|---|
| 3 two-address instr. | ‘00’, ‘01’, ‘10’ | ‘00000’ to ‘11111’ | ‘00000’ to ‘11111’ |
| 31 one-address instr. | ‘11’ | ‘00000’ to ‘11110’ | ‘00000’ to ‘11111’ |
| 24 zero-address instr. | ‘11’ | ‘11111’ | ‘00000’ to ‘10111’ |

# Classroom Test

- 某机器的指令字长为 16 位, 设有单地址指令和两地址指令。若每个地址字段均为 6 位, 且两地址指令有 A 条, 问单地址指令最多可以有多少条?
- 某处理机的指令系统要求有: 三地址指令 4 条, 单地址指令 255 条, 零地址指令 16 条。设指令字长为 12 位, 每个地址码长度为 3 位。问能否用扩展编码为其操作码编码? 如果要求单地址指令为 254 条,能否对其操作码扩展编码? 说明理由。

# Classroom Test 1

- 某机器的指令字长为 16 位, 设有单地址指令和两地址指令。若每个地址字段均为 6 位, 且两地址指令有 A 条, 问单地址指令最多可以有多少条?

# Classroom Test 2

- 某处理机的指令系统要求有: 三地址指令 4 条, 单地址指令 255 条, 零地址指令 16 条。设指令字长为 12 位, 每个地址码长度为 3 位。问能否用扩展编码为其操作码编码? 如果要求单地址指令为 254 条,能否对其操作码扩展编码?

# Classroom Test 1

- **某机器的指令字长为 16 位, 设有单地址指令和两地址指令。若每个地址字段均为 6 位, 且两地址指令有 A 条, 问单地址指令最多可以有多少条?**
- **解：**根据题意,两地址指令格式如下图所示。

| 操作码 | 地址码 1 | 地址码 2 |
|---|---|---|
| 4 位 | 6位 | 6 位 |

其中, 4 位操作码可表示 16 个 $(2^4)$ 短操作码。两地址指令共有 $A$ 条, 占用了 16 个短码点中的 $A$ 个,剩余的 $(16-A)$ 个码点均可用作扩展标志。一个扩展标志都可使用一个 6 位的地址字段 进行扩展, 从而得到 $2^6$ 个扩展操作码。所以, 单地址指令最多可有 $(16-A)\times 2^6$ 条。

# Classroom Test 2

- 某处理机的指令系统要求有: 三地址指令 4 条, 单地址指令 255 条, 零地址指令 16 条。设指令字长为 12 位, 每个地址码长度为 3 位。问能否用扩展编码为其操作码编码? 如果要求单地址指令为 254 条,能否对其操作码扩展编码?
- 解：根据题意,三地址指令格式如下图所示。

| 操作码 | 地址码 1 | 地址码 2 | 地址码 3 |
|---|---|---|---|
| 3 位 | 3位 | 3 位 | 3 位 |

操作码为 3 位长, 可表示 8 个码点, 用 4 个码点表示 4 条三地址指令, 剩下的 4 个码点作为扩展标志。

# Classroom Test 2

❖ **解：** 根据题意,三地址指令格式如下图所示。

| 操作码 | 地址码 1 | 地址码 2 | 地址码 3 |
|---|---|---|---|
| 3 位 | 3位 | 3 位 | 3 位 |

操作码为 3 位长, 可表示 8 个码点, 用 4 个码点表示 4 条三地址指令, 剩下的 4 个码点作为扩展标志。

单地址指令只需用地址码 3 字段表示地址, 其余 6 位地址码可用于表示操作码。6 位长的扩展部分可表示 64 个码点, 有 4 个 3 位长的扩展标志, 所以, 共可表示 $4 \times 2^6 = 256$ 个。若用 255 个码点表示 255 条单地址指令操作码, 则还余下一个码点作为扩展标志。

零地址指令不需要地址码, 地址码 3 的 3 位可用于表示零地址指令的操作码。 3 位长的扩展部分可表示 8 个码点。但是, 9 位长的码点只剩下一个作为扩展标志, 因此, 只能表示 8 条零地址指令操作码, 不能满足题目中的数量要求。

如果单地址指令为 254 条, 则 9 位长的码点余下 2 个码点作为扩展标志, 再扩展 3 位后正好表示 16 条零地址指令操作码。