中国人民大学 信息学院
RENMIN UNIVERSITY OF CHINA SCHOOL OF INFORMATION

程序设计1荣誉课程

1. 计算机与程序运行

授课教师：游伟 副教授

授课时间：周一08:00 – 09:30，周四16:00 – 17:30（明德新闻楼0201）

上机时间：周四18:00 – 21:00（理工配楼二层5号机房）

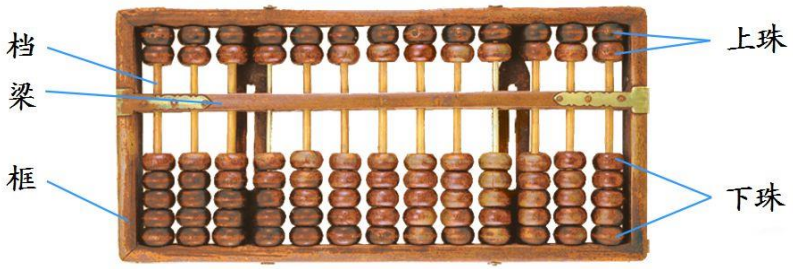
课程主页：https://rucsesec.github.io/programming

目录

1.1 计算模型与体系结构

- 计算：对目标对象的数据或信息进行表示、加工、变换和展示
  - 数值计算
  - 逻辑计算
  - 更复杂的计算

- 计算模型：刻画计算的抽象的形式系统或数学系统
  - 图灵机模型（TM）：通用计算模型，可描述任何理论上可计算的问题
  - 随机存取机模型（RAM）：与图灵机模型等价，但操作更简单



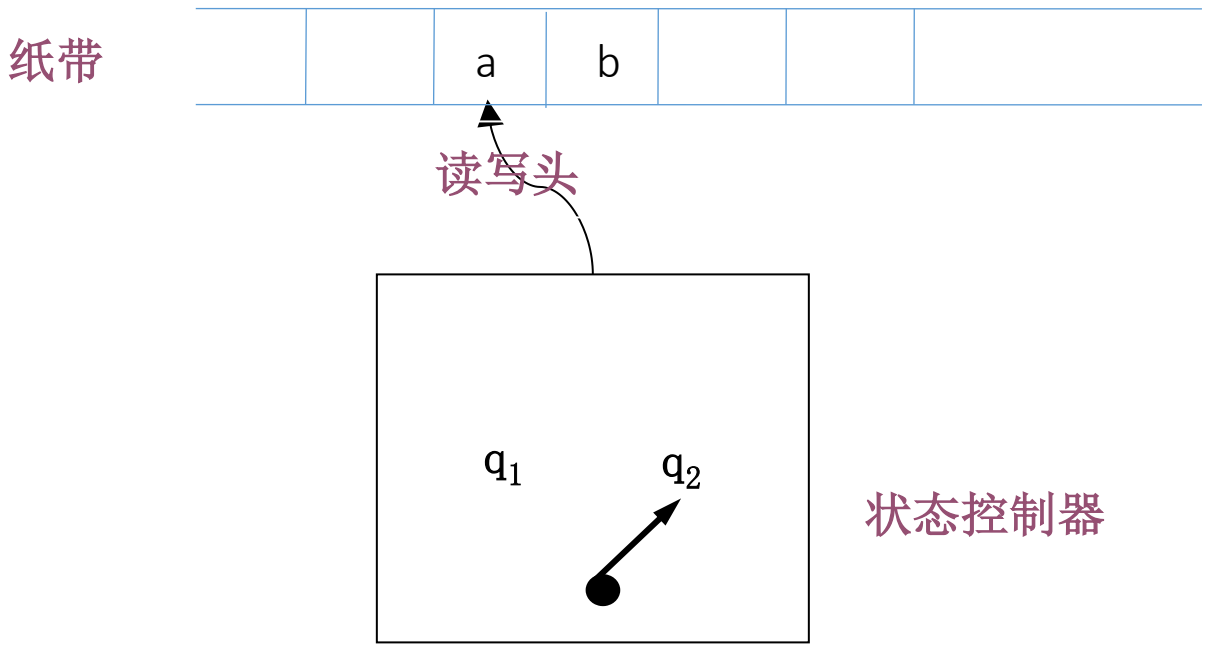
加法口诀（+3）

- 三上三
- 三下五去二
- 三去七进一

更多内容：《计算理论导论荣誉课程》（第6学期）

图灵机模型（TM）

- 直观描述：
  - 3个部件：有限状态控制器、无穷线性带和读写头
  - 3个动作：改变控制器状态、改写当前格符号、将读写头左移/右移一格
  - 1套规则：根据当前格局（当前状态+当前格符号），
    决定采取什么动作（当前状态改变+当前格符号+读写头移动）



规则

当前格局	采取的动作
(q2, a)	(q1, b, R)
(q2, b)	(q1, a, R)
(q1, b)	(q2, a, L)
(q1, a)	(q2, b, L)

图灵机模型（TM）

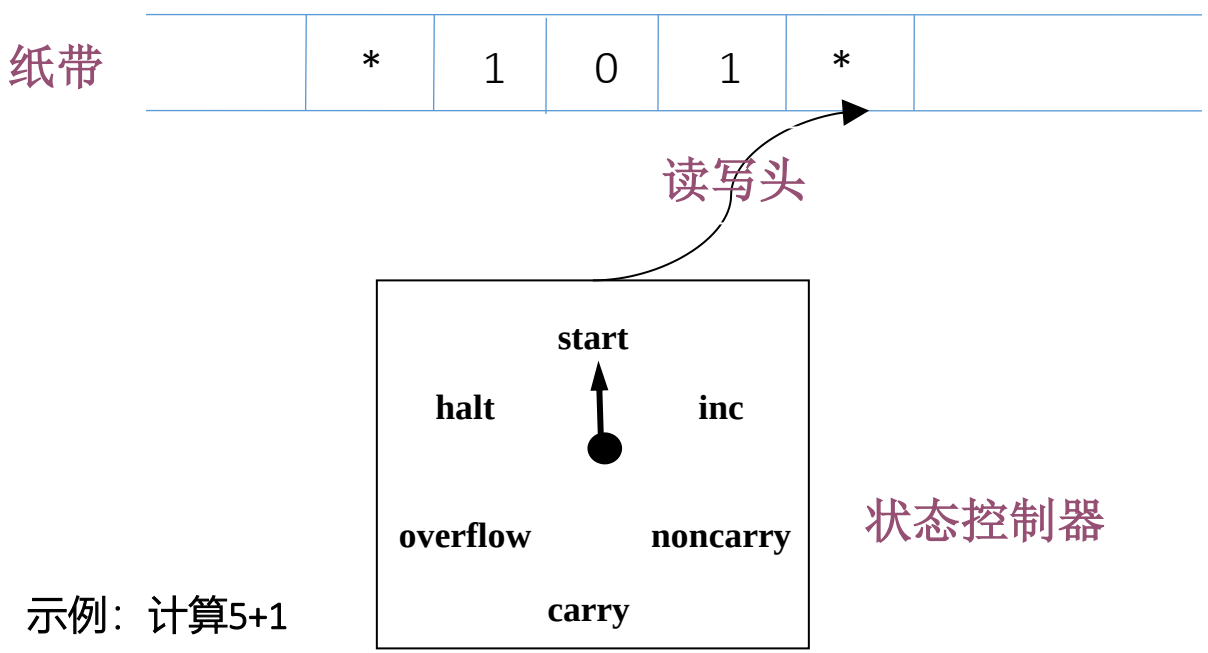
思考：

- 如果用十进制表示x，情况如何？
- 如何用图灵机模型计算x+y？

- 示例：用TM模型计算x+1
  - 编码：用二进制表示x，用*与符号带上的其它符号分隔开
  - 符号集：0, 1, *
  - 状态集：start（初始），inc（增加），noncarray（不进位），carray（进位），overflow（溢出），halt（停止）

规则

当前格局	采取的动作
(start, *)	(inc, *, L)
(inc, 0)	(noncarry, 1, L)
(inc, 1)	(carry, 0, L)
(inc, *)	(halt, *, -)
(noncarray, 0)	(noncarray, 0, L)
(noncarray, 1)	(noncarray, 1, L)
(noncarray, *)	(halt, *, -)
(carry, 0)	(nocarray, 1, L)
(carry, 1)	(carry, 0, L)
(carry, *)	(overflow, 1, L)
(overflow, 0/1/*)	(halt, *, -)

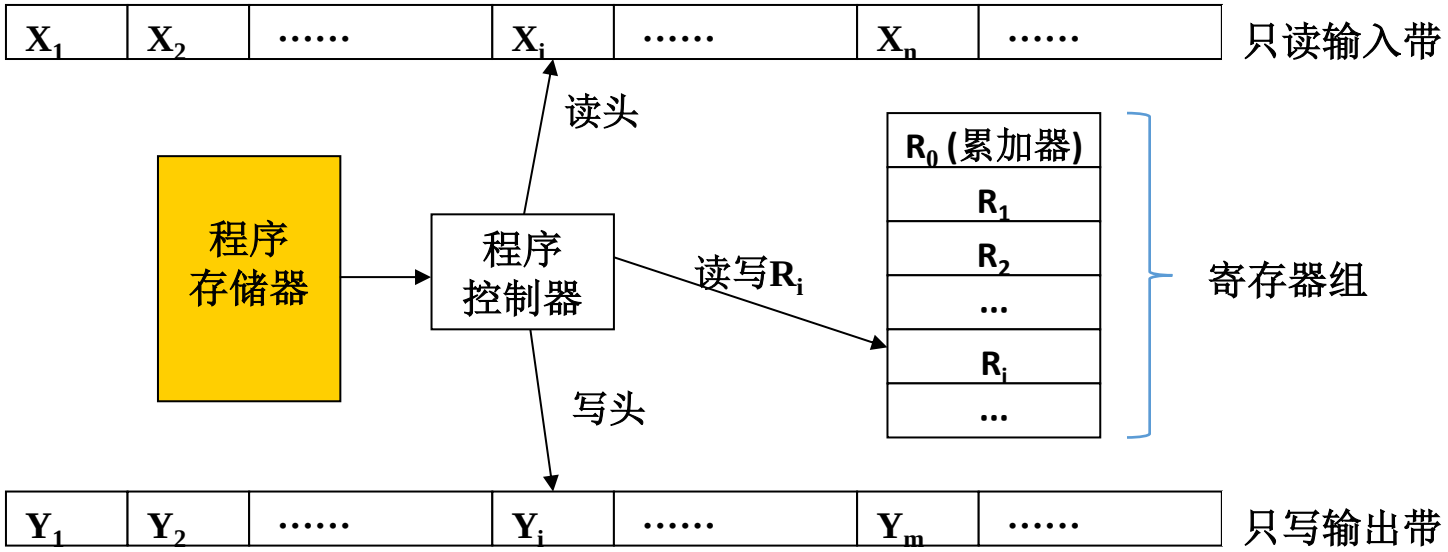


示例：计算5+1

随机存取机模型（RAM）

思考：RAM和TM的联系与区别

- 组成部件：
  - **只读输入带**由一系列方格和带头组成，每输入一次，带头右移一格
  - **只写输出带**由一系列方格和带头组成，每输出一次，带头右移一格
  - **程序控制部件**由指令计数器、指令译码器构成，用来控制程序走向执行
  - **程序存储部件**用来存储程序
  - **寄存器组**由个数无限、按序编号的寄存器组成



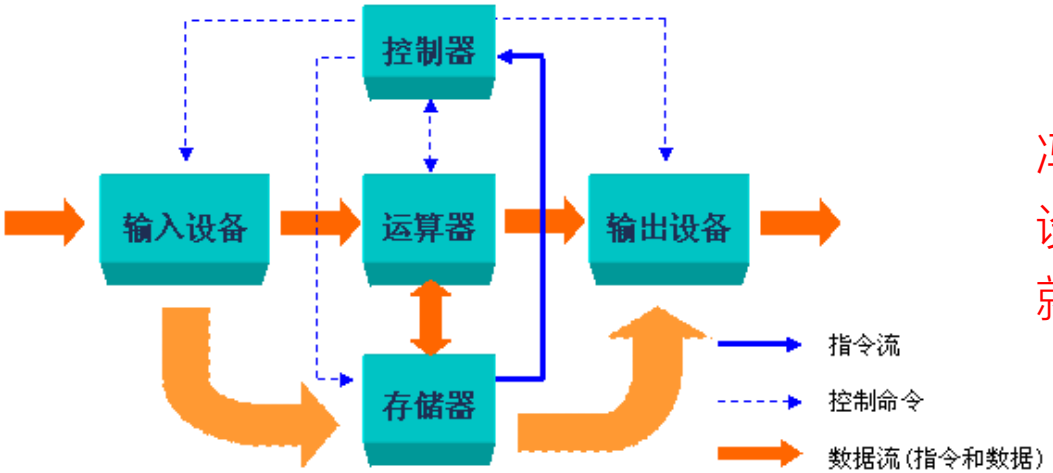
随机存取机模型（RAM）

- 指令集：
  - Input：输入数据
  - Output：输出数据
  - Load：将普通寄存器中的值载入累加器
  - Store：将累加器中的值存入普通寄存器
  - Add：将指定寄存器的值与累加器的值相加，结果存入累加器
- 示例：用RAM模型计算x+y

```
1. Input R1
2. Input R2
3. Load R1      //R0 = R1
4. Add R2       //R0 = R0 + R2
5. Store R3     //R3 = R0
6. Output R3
```

1.1 计算模型与体系结构

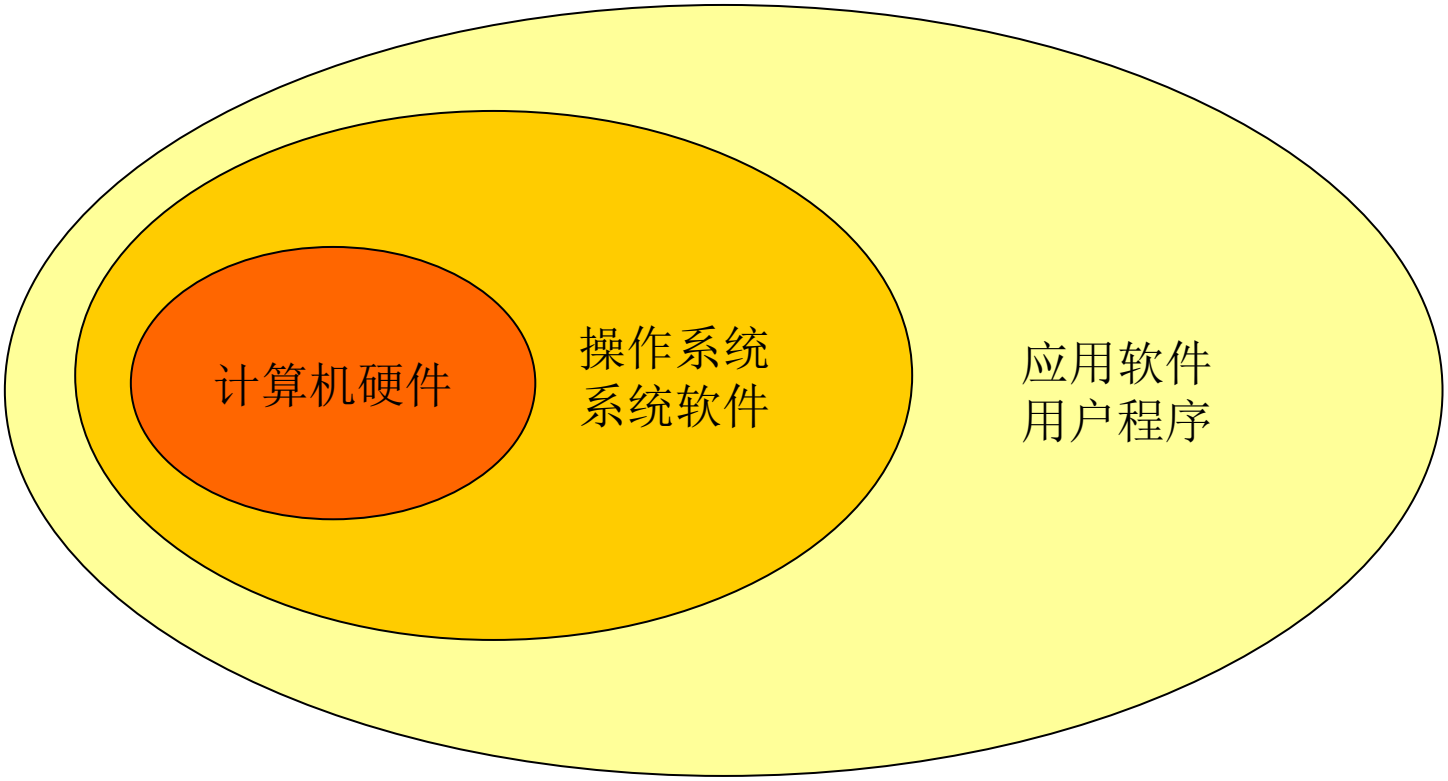
- 体系结构：计算模型的物理实现
- 冯·诺依曼体系结构
  - 计算机硬件由运算器、控制器、存储器、输入/输出设备组成
  - 数据和程序以二进制代码形式存放
  - 控制器根据存放在存储器中的程序来工作



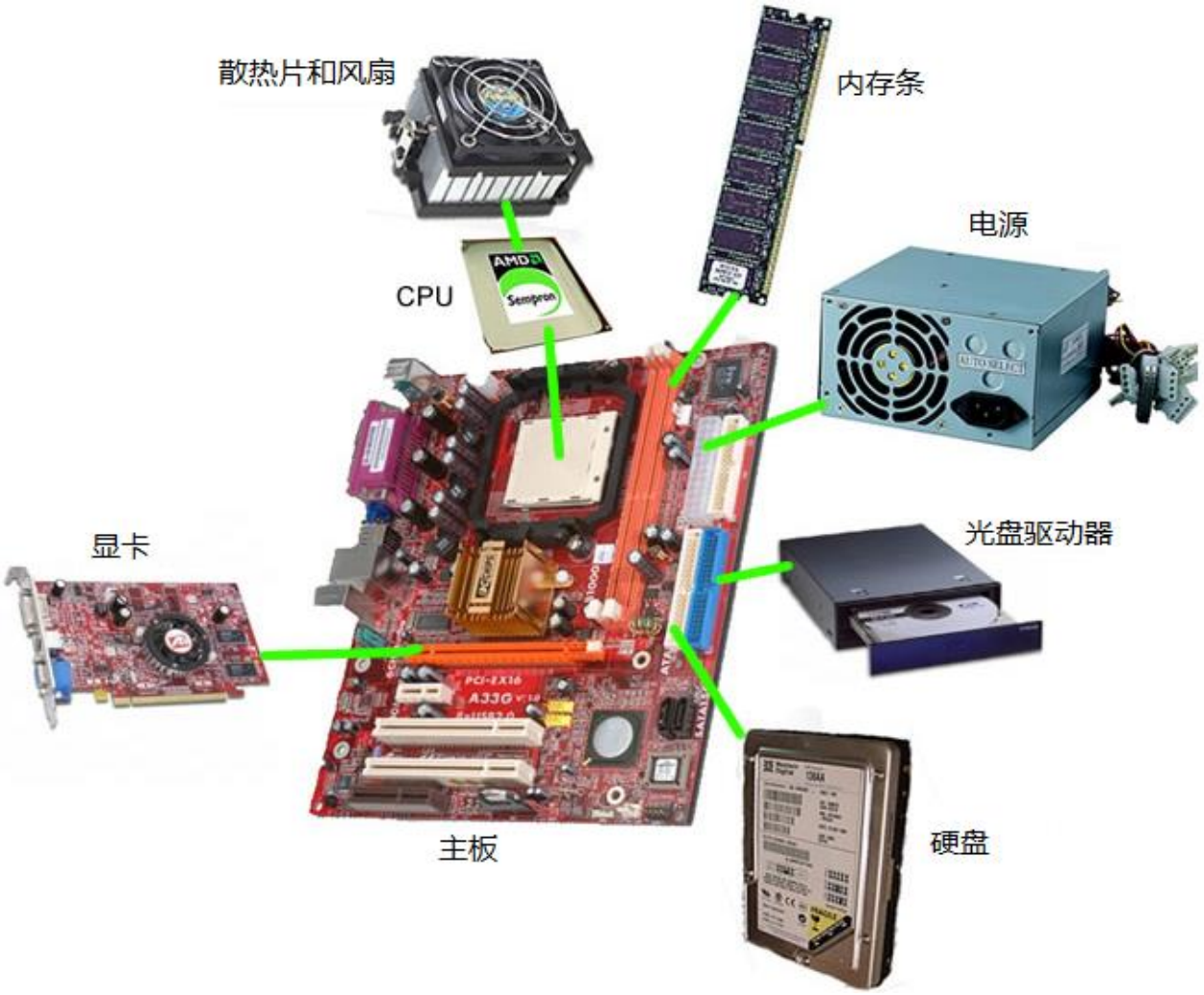
冯·诺依曼体系结构计算机是一种通用设计，只需修改存储器中的指令序列，就能控制计算机做不同的事情

更多内容：《计算机系统基础I》（第3学期）

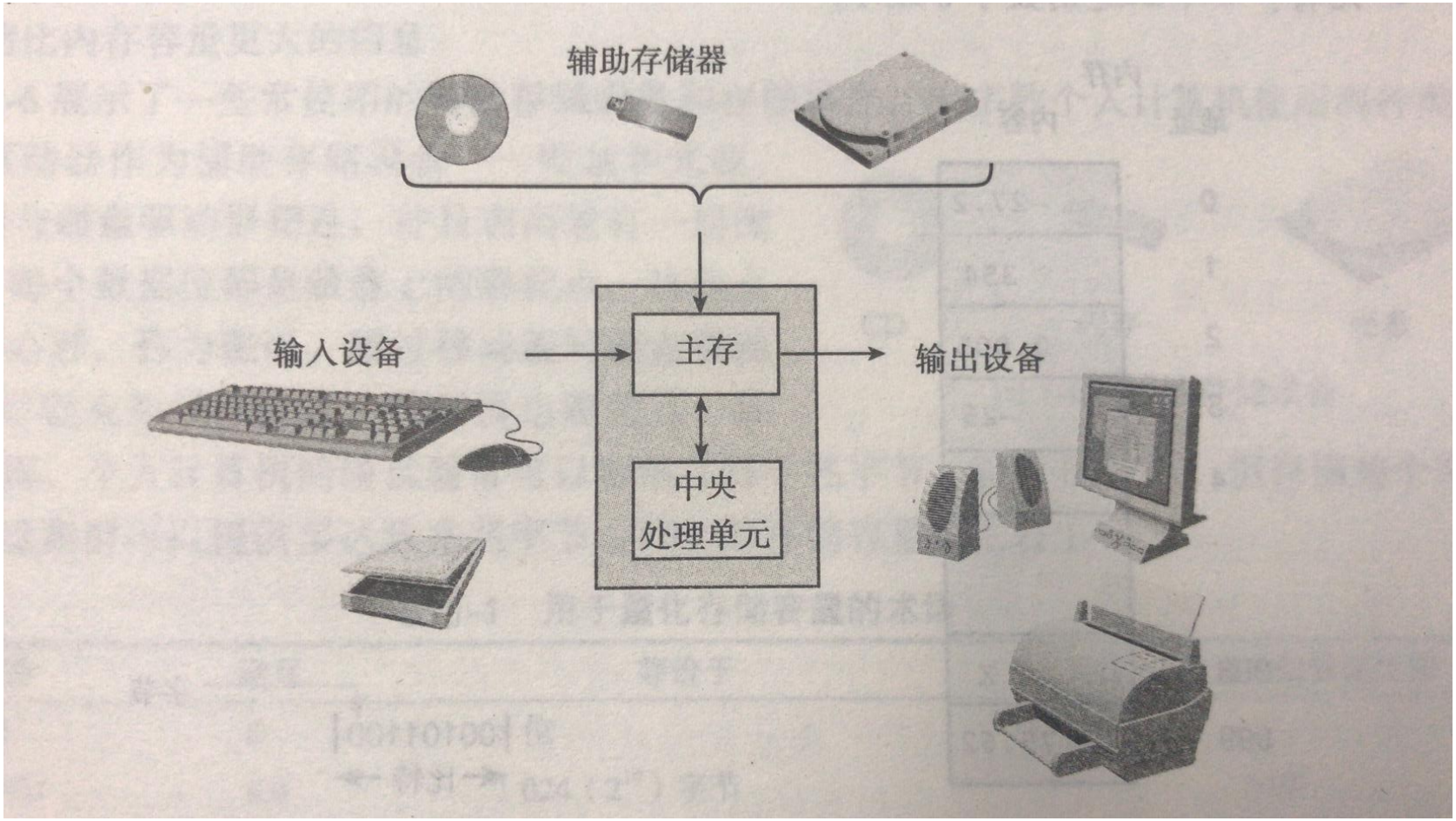
1.2 计算机系统的组成



1.2.1 计算机硬件



1.2.1 计算机硬件



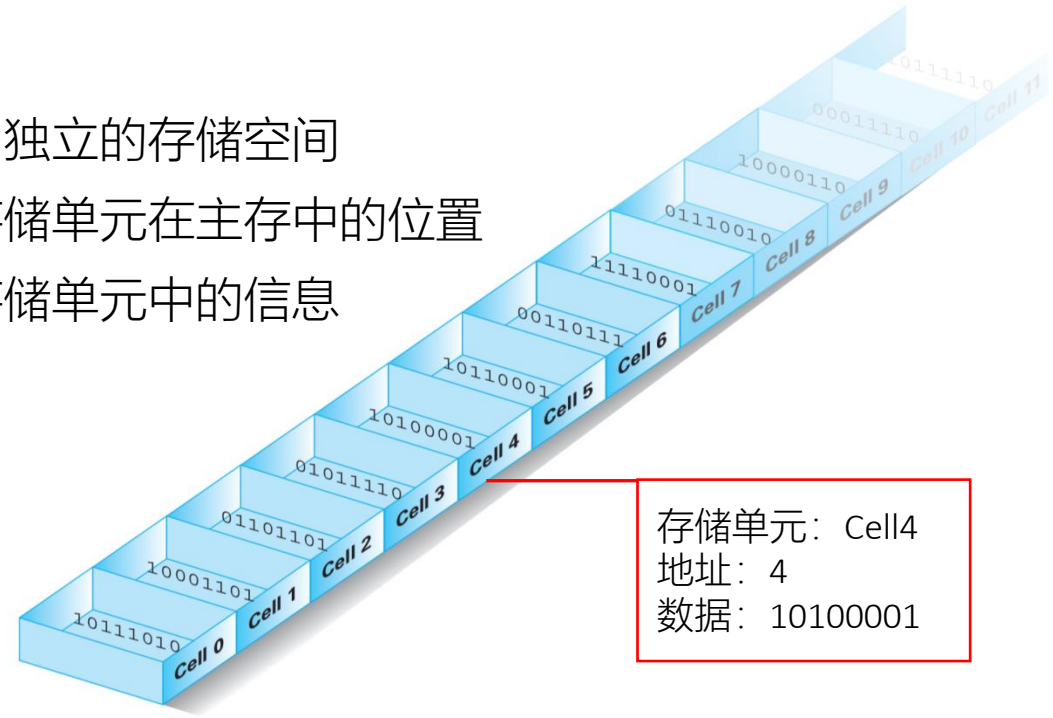
更多内容：《计算机系统基础I》（第3学期）、《计算机系统实现I》（第5学期）

中央处理器（CPU）

- 中央处理器是计算机中执行处理数据指令的器件
- CPU主要由两部分组成：运算器和控制器
  - 运算器进行加减法等算术运算以及比较两个数之类的逻辑运算
  - 控制器包括指令控制器（识别和翻译指令代码）、时序控制器（为每条指令按时间顺序提供控制信号）、总线控制器（控制CPU的内外部总线）和中断控制器（控制各种各样的中断请求）
- 字长：CPU可以同时处理的位数
  - 字长较长的处理器在每个处理周期内可以处理更多的数据
  - 当前的个人计算机通常都是32位或64位处理器

主存储器（Memory）

- 主存用于存放正在运行的程序和需要立即处理的数据
- 主要有两类主存：随机存储器（RAM）和只读存储器（ROM）
  - RAM用于存储临时的数据（如程序运行时的数据），断电后数据消失
  - ROM用于存储永久的数据（如开机引导程序），断电后数据不会消失
- 存储结构
  - 存储单元：主存中独立的存储空间
  - 存储单元地址：存储单元在主存中的位置
  - 存储单元内容：存储单元中的信息



1.2.2 计算机软件

- 系统软件：计算机工作时必须配备的软件
  - 操作系统：管理计算机硬件资源，控制程序运行，为用户提供交互界面
  - 语言处理程序：将高级语言源程序翻译成计算机能识别的目标程序
  - 数据库管理系统：存储和管理海量数据，方便进行查询和增删改操作
- 应用软件：为某种特定用途而开发的软件
  - 文本编辑：UltraEdit, Notepad++, Word, ……
  - 图像处理：Photoshop, ACDSee, 美图秀秀, ……
  - 视频播放：暴风影音, 优酷视频, 腾讯视频, ……
  - 网页浏览：Chrome, Edge, Firefox, Surfari, ……
  - ……

更多内容：《计算机系统基础II》（第4学期）、《计算机系统实现II》（第6学期）

1.3 信息在计算机中的表示

- 信息与数据
  - 信息：各种事物的变化和特征的反映
  - 数据：信息的载体，如数值、文字、图像等
  - 人理解的是信息，计算机处理的是数据
- 计算机中的数据
  - 数值数据：表示量的大小、正负，如整数、小数等
  - 字符数据：表示一些符号、标记，如英文字母A~Z、a~z、数字0～9、各种专用字符+、-、/、（）……及标点符号等
  - 其他数据：声音数据、图像数据等（需要特定的编码/解码规则）

更多内容：《计算机系统基础I》（第3学期）

1.3 信息在计算机中的表示

- 数据单位
  - 信息在计算机中用二进制表示，最小的数据单位是一个数位（bit）
  - 8个数位被称为1个字节（Byte），是计算机中表示存储容量的基本单位
  - 存储容量的计量单位有字节B、千字节KB、兆字节MB、十亿字节GB及万亿字节TB等，换算关系如下：

    1 B = 8 bit

    1 KB = 1024 B = 2^{10} B

    1 MB = 1024 KB = 2^{20} B

    1 GB = 1024 MB = 2^{30} B

    1 TB = 1024 GB = 2^{40} B

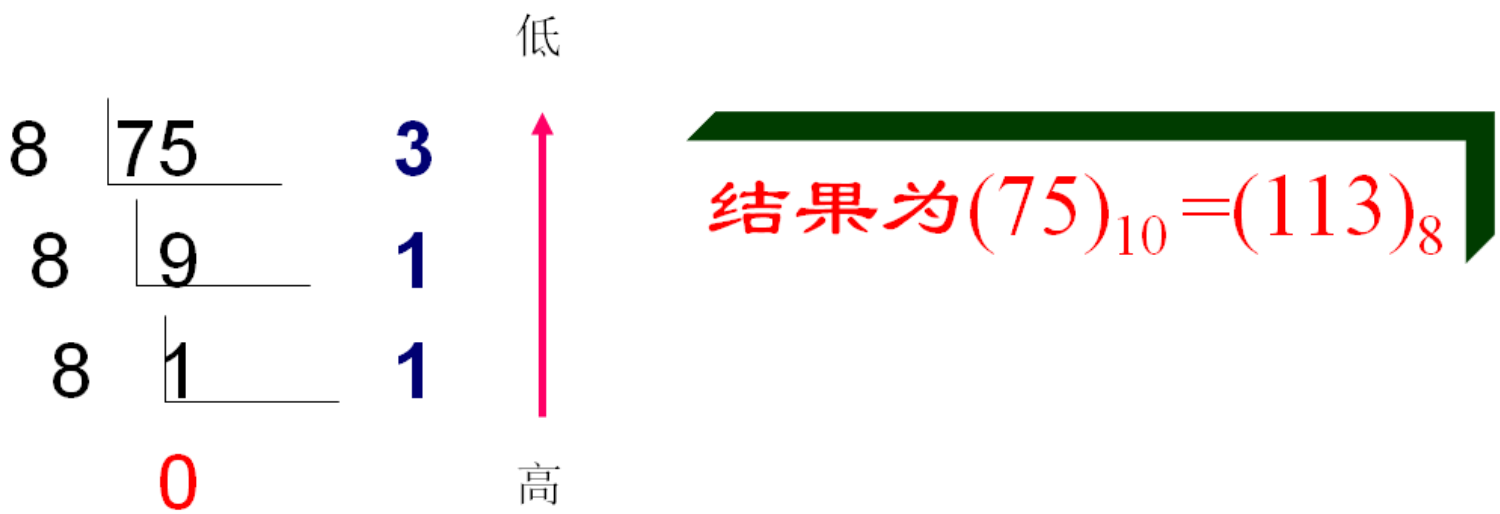
1.3.1 进制及其转换

- 基本概念
  - 数制：用一组固定的数码和一套统一的规则来表示数目的方法
  - 进位计数制：按照进位方式计数的数制
  - 基数：某种进制中允许使用的基本数码的个数
  - 位权：一个数码处在不同位置上代表的值不同。每个数码所代表的数值等于该数码乘以一个与数码所在位置相关的常数。这个常数称为味权。
- 计算机科学常用的进制
  - 二进制（Binary）
  - 八进制（Octonary）
  - 十六进制 (Hexadecimal)

十进制整数 ➔ 非十进制整数

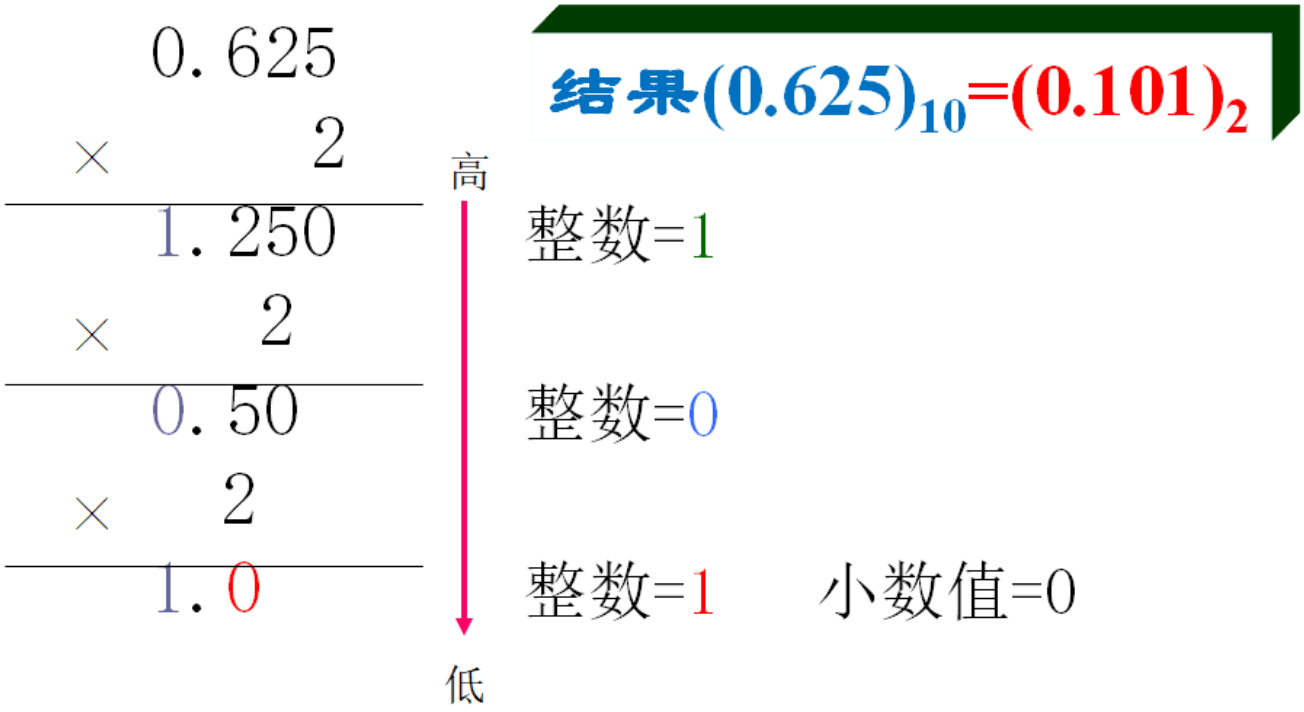
余数法： **除基数取余数。**

示例： $(75)_{10}=(\ ?\ )_8$



十进制小数 ➔ 非十进制小数

进位法： 乘基数取整。用十进制小数乘基数，当积为0或达到所要求的精度时，将整数部分由上而下排列。 示例： $(0.625)_{10}=(?)_2$



十进制数 ➔ 非十进制数

例：$(81.65)_{10}=(?)_2$，要求精度为小数六位。

整数与小数分别转换

（0.65）$_{10}$ =（ ? ）$_2$，*进位法*—乘基数取整。

得：$(0.65)_{10}=(0.101001)_2$

（81）$_{10}$ =（ ? ）$_2$，*余数法*—除基取余法

得：（81）$_{10}$ = （1010001）$_2$

综合得：$(81.65)_{10}=(1010001.101001)_2$

注意：小数转换不一定能算尽，只能算到一定精度的位数为止，故要产生一些误差。当位数较多时，这个误差就很小了。

非十进制数 ➔ 十进制数

利用多项式表示法（**位权多项式法**）把各非十进制数按权展开求和。

转换公式：

$$(s)_N = \sum_{n}^{-m} k_i N^{i-1}$$

$$(S)_N = (k_n k_{n-1} \cdots k_1 . \quad k_0 k_{-1} \cdots k_{-m})_N$$

例：$(1011.1)_2 = 1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}$

$= 8+0+2+1+0.5$

$= (11.5)_{10}$

二进制与八进制之间的转换

从小数点开始，将二进制数的整数和小数部分每三位分为一组，不足三位的分别在整数的最高位前和小数的最低位后加“0”补足，然后每组用等值的八进制码替代，即得目的数。

例：（11010111.0100111）$_B$ = （？）

0

0 11010111.0100111 00

3 2 7 2 3 4

三位合一位

得：（11010111.0100111）$_B$ = （327.234）$_O$

（327.234）$_O$ = （？）$_B$

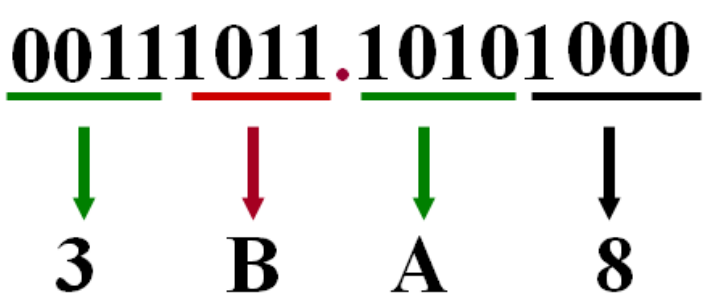
小数点为界

一位拆三位

二进制与十六进制之间的转换

从小数点开始，将二进制数的整数和小数部分每四位分为一组，不足四位的分别在整数的最高位前和小数的最低位后加“0”补足，然后每组用等值的十六进制码替代，即得目的数。

例9：111011.10101 B = 3B.A8 H



四位合一位

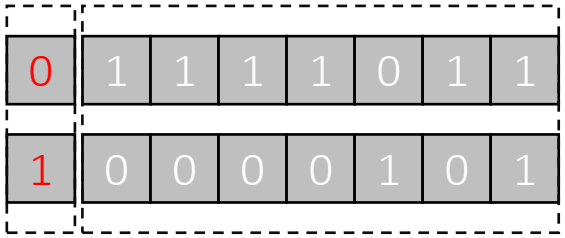
一位拆四位

1.3.2 数值的表示

- 真值与机器数
  - 真值：在计算机外部，用正负号表示的实际数值
  - 机器数：在计算机内部，将正负号数字化后得到的数值编码结果

$(+123)_D$

$(-123)_D$



符号位

- 定点数与浮点数
  - 定点数：约定所有数值数据的小数点隐含在某一个固定位置上
  - 浮点数：小数点的位置可按需要浮动

定点整数的表示

注意：
- $B2U$、$B2T_\omega$、 $B2F$是双射
- $B2S_\omega$、$B2O_\omega$是满射，但不是单射

- 假设一个二进制整数有ω位，位向量$\vec{x} = [x_{\omega-1}, x_{\omega-2}, \ldots, x_0]$

- 无符号整数

$$B2U(\vec{x}) \stackrel{\text{def}}{=} \sum_{i=0}^{\omega-1} x_i \cdot 2^i$$ 值的范围：$[0, 2^\omega - 1]$

- 有符号整数
  - 原码（Sign-Magnitude）：除符号位以外，其余位保持原样

$$B2S_\omega(\vec{x}) \stackrel{\text{def}}{=} (-1)^{x_{\omega-1}} \cdot \left(\sum_{i=0}^{\omega-2} x_i \cdot 2^i \right)$$ 值的范围：$[-2^{\omega-1}+1, 2^{\omega-1}-1]$

  - 反码（Ones'-Complement）：反码 + |负数| $= 2^\omega - 1$（ω位二进制所能表示的最大无符号整数）

$$B2O_\omega(\vec{x}) \stackrel{\text{def}}{=} -x_{\omega-1} \cdot (2^{\omega-1} - 1) + \sum_{i=0}^{\omega-2} x_i \cdot 2^i$$ 值的范围：$[-2^{\omega-1}+1, 2^{\omega-1}-1]$

  - 补码（Two's-Complement）：补码 + |负数| $= 2^\omega$（模数），补码=反码+1

$$B2T_\omega(\vec{x}) \stackrel{\text{def}}{=} -x_{\omega-1} \cdot 2^{\omega-1} + \sum_{i=0}^{\omega-2} x_i \cdot 2^i$$ 值的范围：$[-2^{\omega-1}, 2^{\omega-1}-1]$

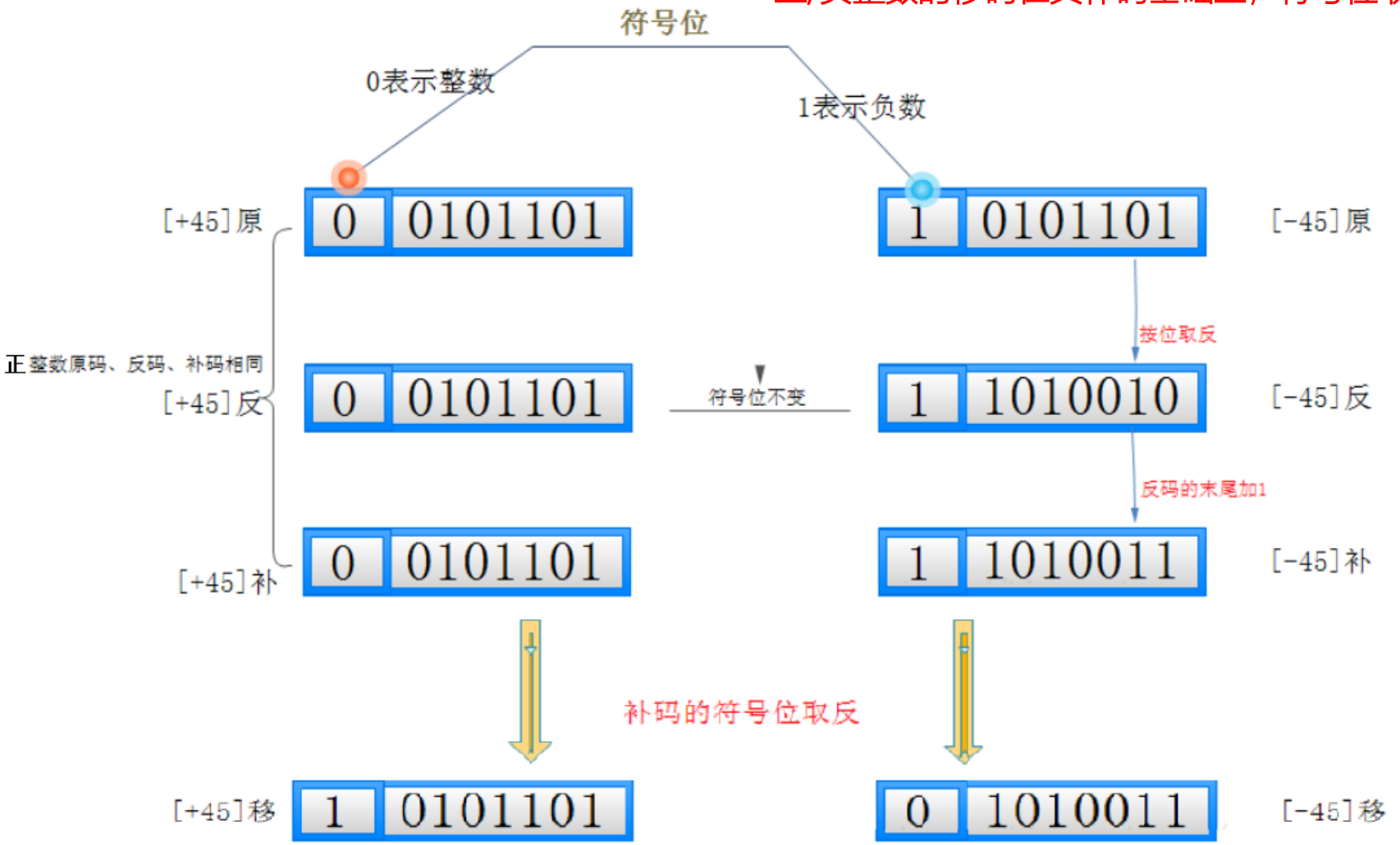
  - 移码（Frame Shift）：移码 = 补码 + $2^{\omega-1}$

$$B2F(\vec{x}) \stackrel{\text{def}}{=} (-x_{\omega-1} \cdot 2^{\omega-1})^{(1-x_{\omega-1})} + \sum_{i=0}^{\omega-2} x_i \cdot 2^i$$ 值的范围：$[-2^{\omega-1}, 2^{\omega-1}-1]$

定点整数的表示

规律：

- 正整数的原码、反码、补码相同
- 负整数的原码符号位为1，其余位不变，
- 负整数的反码在其原码基础上，按位取反，
- 负整数的补码在其反码基础上，末位加1
- 正/负整数的移码在其补码基础上，符号位取反



编码的意义

- 硬件局限：只有加法器，没有减法器
- 编码目标：
  - 运算的操作数和运算结果使用相同的编码
  - 能简单判断正负和比较值的大小
  - 运算过程不需要判断正负和比较值的大小
  - 所有的减法操作都能转换成加法操作
- 在绝大多数计算机中，使用补码来表示定点整数

有符号整数几种编码方式的比较（假定$\omega=4$）

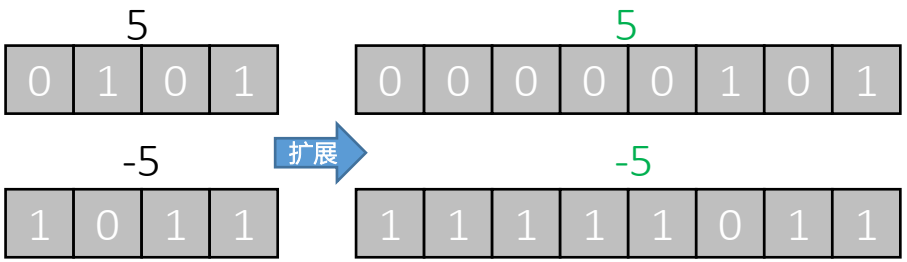
- 原码
  - 在反码表示中，数值0有两种表示形式（$[0000]_{原}$和$[1000]_{原}$）
  - 计算$(+1)_D+(+1)_D$：$[0001]_{原}+[0001]_{原}=[0010]_{原}=(+2)_D$
  - 计算$(+1)_D+(-1)_D$：$[0001]_{原}+[1001]_{原}=[1010]_{原}=(-2)_D$
  - 计算$(-1)_D+(-1)_D$：$[1001]_{原}+[1001]_{原}=[0010]_{原}=(+2)_D$
- 反码
  - 在反码表示中，数值0有两种表示形式（$[0000]_{反}$和$[1111]_{反}$）
  - 计算$(+1)_D+(+1)_D$：$[0001]_{反}+[0001]_{反}=[0010]_{反}=(+2)_D$
  - 计算$(+1)_D+(-1)_D$：$[0001]_{反}+[1110]_{反}=[1111]_{反}=(-0)_D$
  - 计算$(-1)_D+(-1)_D$：$[1110]_{反}+[1110]_{反}=[1100]_{反}=(-3)_D$

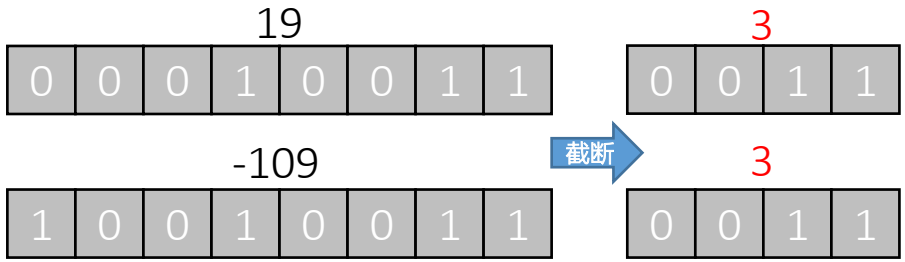
有符号整数几种编码方式的比较（假定$\omega=4$）

- 补码
  - 在补码表示中，数值0只有一种表示形式（$[0000]_{补}$表示0，$[1000]_{补}$表示-8)
  - 计算$(+1)_D+(+1)_D$：$[0001]_{补}+[0001]_{补}=[0010]_{补}=(+2)_D$
  - 计算$(+1)_D+(-1)_D$：$[0001]_{补}+[1111]_{补}=[0000]_{补}=(0)_D$
  - 计算$(-1)_D+(-1)_D$：$[1111]_{补}+[1111]_{补}=[1110]_{反}=(-2)_D$
- 移码
  - 移码不使用符号位判断正负，而是通过检查是否大于$2^{\omega-1}$来判断正负
  - 计算$(+1)_D+(+1)_D$：$[1001]_{移}+[1001]_{移}=[0010]_{移}=(-6)_D$
  - 计算$(+1)_D+(-1)_D$：$[1001]_{移}+[0111]_{移}=[0000]_{移}=(-8)_D$
  - 计算$(-1)_D+(-1)_D$：$[0111]_{移}+[0111]_{移}=[1110]_{移}=(+6)_D$

定点整数的扩展与截断

- 假设一个二进制整数有ω位，位向量$\vec{x} = [x_{\omega-1}, x_{\omega-2}, \dots, x_0]$，现欲将其变成$\omega'$位，位向量$\vec{x}'$
- 扩展（当$\omega' > \omega$时）：从数值角度考虑（保值）
  - 无符号整数的零扩展：$\vec{x}' = [0, \dots 0, x_{\omega-1}, x_{\omega-2}, \dots, x_0]$
  - 有符号整数补码表示形式的符号扩展：$\vec{x}' = [x_{\omega-1}, \dots, x_{\omega-1}, x_{\omega-1}, \dots, x_0]$
- 截断（当$\omega' < \omega$时）：从位级角度考虑（不保值）
  - 截断无符号整数和有符号整数：$\vec{x}' = [x_{\omega'-1}, x_{\omega'-2}, \dots, x_0]$
  - 规律：$x' = x \bmod 2^{\omega'}$





有符号整数和无符号整数之间的转换

- 从数值角度考虑
  - 对于在两种形式中都能表示的值，保持不变
  - 将负数转换成无符号数，可能会得到0
  - 待转换的无符号数超出补码能表示的范围，可能会得到所能表示的最大值
- 从位级角度考虑（大多数C语言的实现）
  - 规则：保持位值不变，只是改变了解释这些位的方式，从而数值可能改变
  - 无符号整数转换为有符号整数的补码表示形式

$$U2T_\omega(x) = \begin{cases} x, & x \le 2^{\omega-1} - 1 \\ x - 2^\omega, & x > 2^{\omega-1} - 1 \end{cases} \qquad (0 \le x \le 2^\omega - 1)$$

  - 有符号整数的补码表示形式转换为无符号整数

$$T2U_\omega(x) = \begin{cases} x + 2^\omega, & x < 0 \\ x, & x \ge 0 \end{cases} \qquad (-2^\omega \le x \le 2^\omega - 1)$$

无符号整数：19

0	0	0	1	0	0	1	1

有符号整数的补码形式：19

无符号整数：147

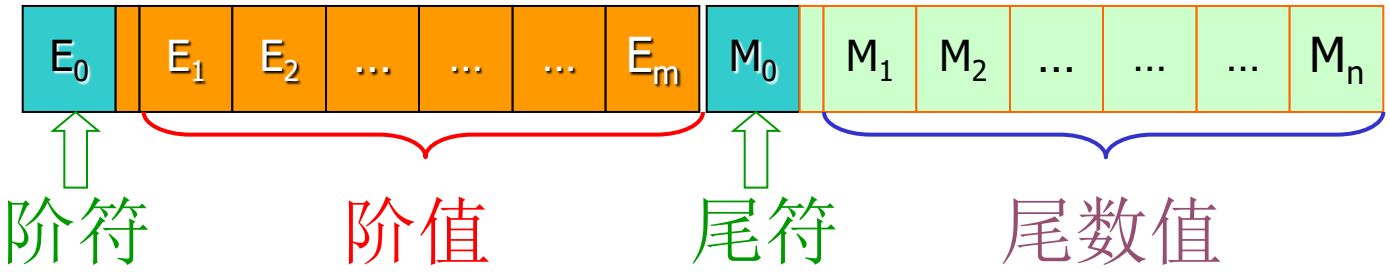
1	0	0	1	0	0	1	1

有符号整数的补码形式：-109

浮点数的表示

- 科学计数法：$N = M \times R^E$
  - M：尾数，R：基值，E：阶码
  - 示例：$0.00303 = 3.03 \times 10^{-3} = 0.303 \times 10^{-2}$

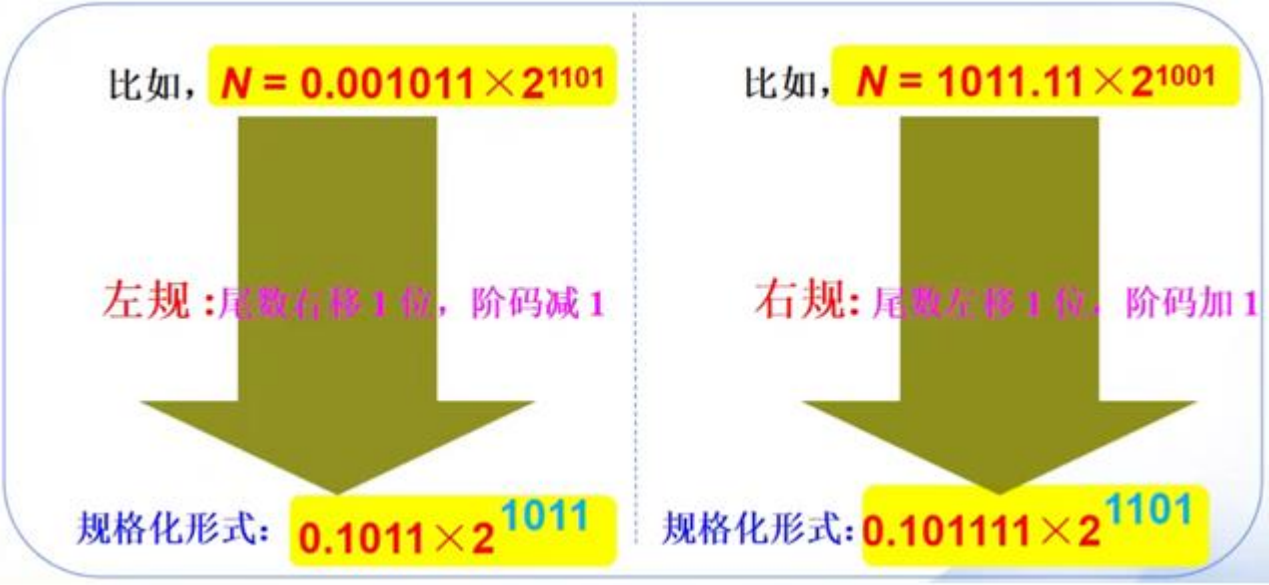
    $19880000 = 1.988 \times 10^{7} = 0.1988 \times 10^{8}$
- 一个机器浮点数由**尾数**和**阶码**及其**符号**位组成
  - 尾数：用定点小数表示，给出有效数字的位数，决定了浮点数的表示精度
  - 阶码：用定点整数表示，指明小数点所在位置，决定了浮点数的表示范围
  - 尾数越长，表示的精度越高；阶码越长，表示的范围越广
  - 在固定长度的浮点数格式内，表示范围和表示精度是一对矛盾



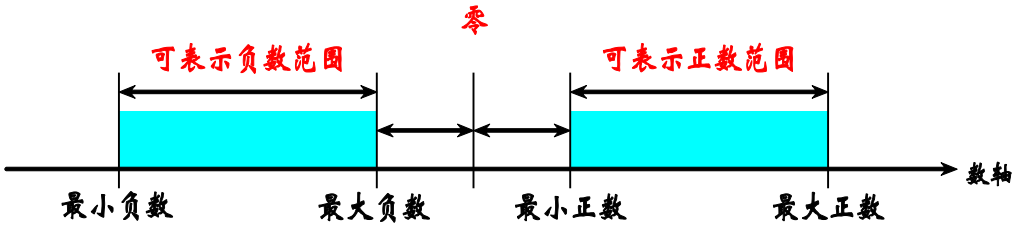
浮点数的规格化

- 目的：为了数据表示的唯一性，为了提高数据的表示精度
- 规格化要求：尾数的最高有效位为1，从而保证有n个有效数字
- 规格化通过尾数移位和修改阶码实现
  - 尾数的小数点每向左移动1位，阶码相应加1
  - 尾数的小数点每向右移动1位，阶码相应减1

- 二进制原码的规格化数：
  - 正数：0.1xxxxxxx
  - 负数：1.1xxxxxxx
- 二进制补码的规格化数：
  - 正数：0.1xxxxxxx
  - 负数：1.0xxxxxxx



浮点数的表示范围



非规格化形式

最大正数：$Max(+) = S_{max} \times r^{j_{max}}$
$= 0.11\ldots1 \times 2^{11\ldots1}$
$= (1-2^{-n}) \times 2^{2^m-1}$

最小正数：$Min(+) = S_{min} \times r^{j_{-max}}$
$= 0.00\ldots01 \times 2^{-11\ldots1}$
$= (2^{-n}) \times 2^{-(2^m-1)}$

最小负数：$Min(-) = -S_{max} \times r^{j_{max}}$
$= -0.11\ldots1 \times 2^{11\ldots1}$
$= -(1-2^{-n}) \times 2^{2^m-1}$

最大负数：$Max(-) = -S_{min} \times r^{j_{-max}}$
$= -0.00\ldots01 \times 2^{-11\ldots1}$
$= -(2^{-n}) \times 2^{-(2^m-1)}$

规格化形式

最大正数：$Max(+) = S_{max} \times r^{j_{max}}$
$= 0.11\ldots1 \times 2^{11\ldots1}$
$= (1-2^{-n}) \times 2^{2^m-1}$

最小正数：$Min(+) = S_{min} \times r^{j_{-max}}$
$= 0.10\ldots0 \times 2^{-11\ldots1}$
$= (2^{-1}) \times 2^{-(2^m-1)}$

最小负数：$Min(-) = -S_{max} \times r^{j_{max}}$
$= -0.11\ldots1 \times 2^{11\ldots1}$
$= -(1-2^{-n}) \times 2^{2^m-1}$

最大负数：$Max(-) = -S_{min} \times r^{j_{-max}}$
$= -0.10\ldots0 \times 2^{-11\ldots1}$
$= -(2^{-1}) \times 2^{-(2^m-1)}$

注：n为尾数位数，m为阶码位数

1.3.3 字符的表示

- 字符必须按特定规则变成二进制编码才能输入计算机
- 字符编码：规定用怎样的二进制码来表示字符
- 西文字符：ASCII编码
  - **ASCII**（American Standard Code for Information Interchange,美国信息交换标准码）。通用的是7位ASCII码，用7位二进制数表示一个字符的编码。共有$2^7=128$个不同的编码
  - 主要的字符及其编码：
    - 48 ~ 57为10个阿拉伯数字0 ~ 9
    - 65 ~ 90为26个大写英文字母A ~ Z
    - 97 ~ 122为26个小写英文字母a ~ z
    - 32为空格，13为回车

ASCII表

ASCII 值	控制字符
0	NUT
1	SOH
2	STX
3	ETX
4	EOT
5	ENQ
6	ACK
7	BEL
8	BS
9	HT
10	LF
11	VT
12	FF
13	CR
14	SO
15	SI
16	DLE
17	DCI
18	DC2
19	DC3
20	DC4
21	NAK
22	SYN
23	TB
24	CAN
25	EM
26	SUB
27	ESC
28	FS
29	GS
30	RS
31	US
127	DEL

可显示字符

控制字符

二进制	十进制	十六进制	图形	Binary	Decimal	Hex	Graphic	Binary	Decimal	Hex	Graphic
0010 0000	32	20	(空格) (sp)	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(	0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29	)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k
0010 1100	44	2C	,	0100 1100	76	4C	L	0110 1100	108	6C	l
0010 1101	45	2D	-	0100 1101	77	4D	M	0110 1101	109	6D	m
0010 1110	46	2E	.	0100 1110	78	4E	N	0110 1110	110	6E	n
0010 1111	47	2F	/	0100 1111	79	4F	O	0110 1111	111	6F	o
0011 0000	48	30	0	0101 0000	80	50	P	0111 0000	112	70	p
0011 0001	49	31	1	0101 0001	81	51	Q	0111 0001	113	71	q
0011 0010	50	32	2	0101 0010	82	52	R	0111 0010	114	72	r
0011 0011	51	33	3	0101 0011	83	53	S	0111 0011	115	73	s
0011 0100	52	34	4	0101 0100	84	54	T	0111 0100	116	74	t
0011 0101	53	35	5	0101 0101	85	55	U	0111 0101	117	75	u
0011 0110	54	36	6	0101 0110	86	56	V	0111 0110	118	76	v
0011 0111	55	37	7	0101 0111	87	57	W	0111 0111	119	77	w
0011 1000	56	38	8	0101 1000	88	58	X	0111 1000	120	78	x
0011 1001	57	39	9	0101 1001	89	59	Y	0111 1001	121	79	y
0011 1010	58	3A	:	0101 1010	90	5A	Z	0111 1010	122	7A	z
0011 1011	59	3B	;	0101 1011	91	5B	[	0111 1011	123	7B	{
0011 1100	60	3C	<	0101 1100	92	5C	\	0111 1100	124	7C	|
0011 1101	61	3D	=	0101 1101	93	5D	]	0111 1101	125	7D	}
0011 1110	62	3E	>	0101 1110	94	5E	^	0111 1110	126	7E	~
0011 1111	63	3F	?	0101 1111	95	5F	_				

1.3.3 字符的表示

- 中文字符：
  - **国标码**：常用汉字6763个，其中一级汉字3755个，二级汉字3008个。国标码用两个字节来表示一个汉字，每个字节的最高位为0
  - **机内码**：在计算机内部对汉字进行存储、处理的编码。一个汉字的内码用两个字节存储，每个字节的最高位为1，以区别于ASCII码

图：国标码 $(4D3C)_H$ $(0100110100111100)_B$
机内码 $(CDBC)_H$ $(1100110110111100)_B$

灵：国标码 $(4169)_H$ $(0100000101101001)_B$
机内码 $(C1E9)_H$ $(1100000111101001)_B$

1.4 程序与程序设计语言

- 程序设计语言：指挥计算机工作的命令。没有程序，计算机将无法进行工作
- 程序设计语言的分类
  - 按照人与机器的交互程度分类
  - 按照程序的运行方式分类
  - 根据程序语言解决问题的方法分类

更多内容：《程序设计II荣誉课程》（第2学期）《编译原理》（第4学期）

1.4.1 按照人与机器的交互程度分类

- 机器语言：由“0”和“1”组成的二进制码构成
  - 优点：速度快，不需要翻译
  - 缺点：依赖机器，可读性差，难以掌握
- 汇编语言：用符号和助记符来代表机器语言
  - 优点：速度快，可读性较好
  - 缺点：依赖机器，较难掌握，需要翻译
- 高级语言：用类英语描述对机器的指令
  - 优点：可读性好，易于掌握；不依赖机器，可实现跨平台
  - 缺点：翻译需要时间开销

```
1. int i, sum = 0;
2. for (i = 0; i < 10; i++) sum = sum + i;
```

高级语言

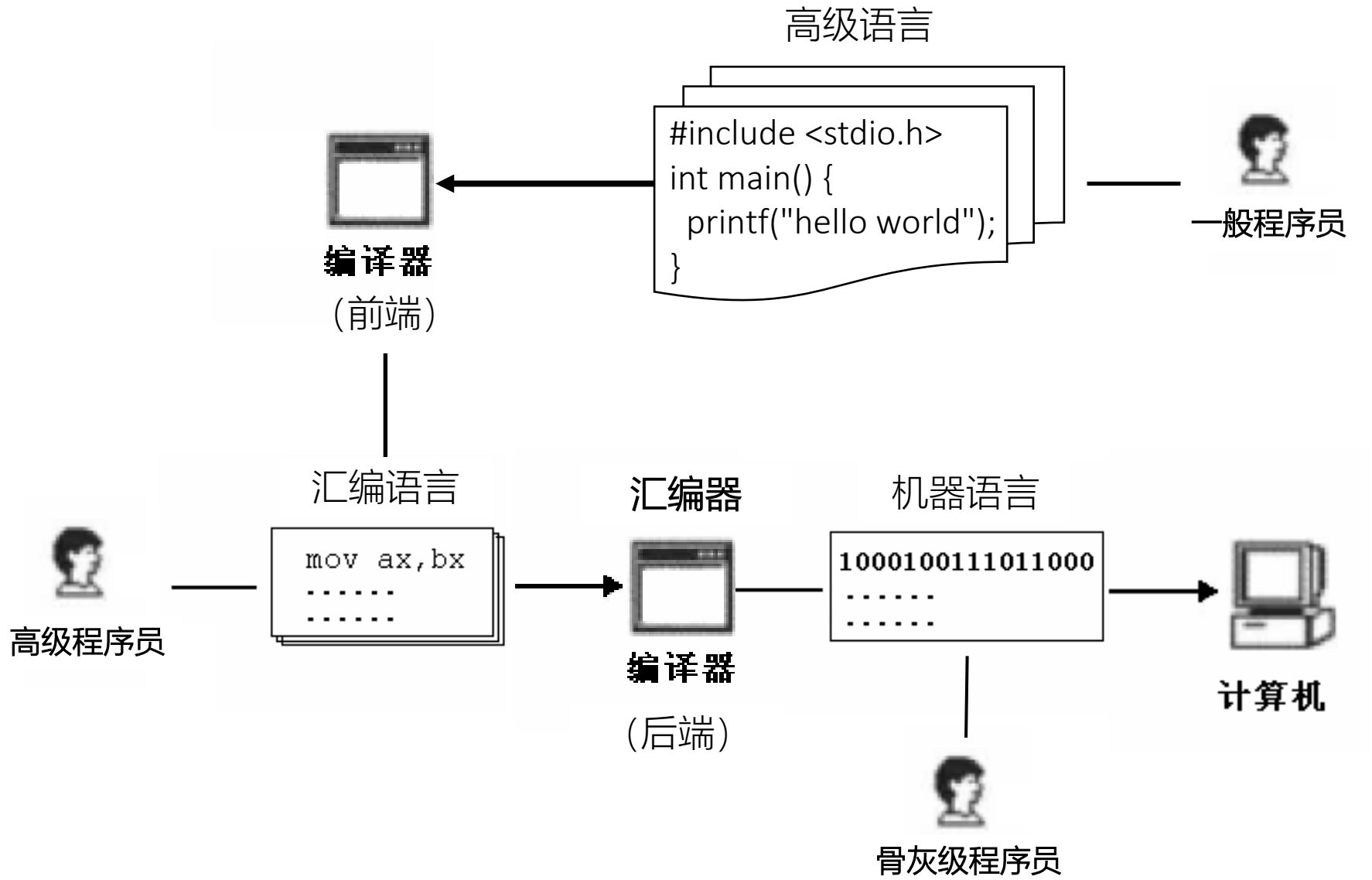
```
1. 1131: movl $0x0,-0x8(%rbp)
2. 1138: movl $0x0,-0x4(%rbp)
3. 113f: jmp 114b <main+0x22>
4. 1141: mov -0x4(%rbp),%eax
5. 1144: add %eax,-0x8(%rbp)
6. 1147: addl $0x1,-0x4(%rbp)
7. 114b: cmpl $0x9,-0x4(%rbp)
8. 114f: jle 1141 <main+0x18>
9. 1151: mov $0x0,%eax
```

汇编语言

83 45 fc 01

机器语言

1.4.1 按照人与机器的交互程度分类

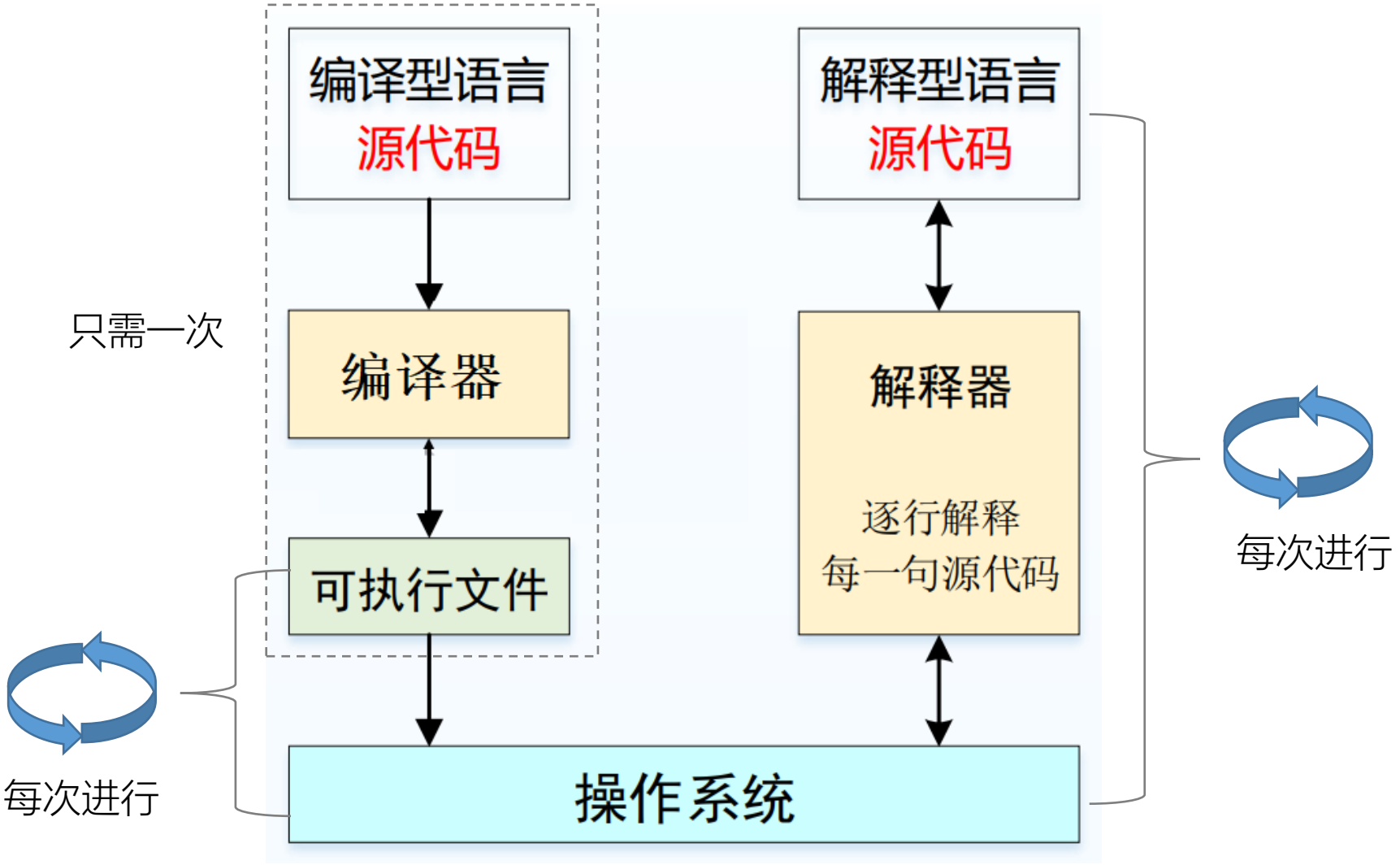


1.4.2 按照程序的运行方式分类

- 编译型语言：使用专门的编译器，针对特定的平台，将高级语言源代码一次性的编译成可被该平台硬件执行的机器码，并包装成该平台所能识别的可执行程序的格式
  - 优点：运行速度快，代码效率高
  - 缺点：代码需要编译才能运行，可移植性差，只能在兼容的系统上运行
  - 常见的编译型语言：C，C++，Pascal，……
- 解释型语言：每次运行时，都使用专门的解释器对源程序逐行解释成特定平台的机器码并立即执行
  - 优点：可移植性好，只要有解释环境，可以在不同的操作系统上运行
  - 缺点：需要解释环境，运行起来比编译慢，占用资源也要多一些，代码效率低，不仅要给用户程序分配空间，解释器本身也占用了系统资源
  - 常见的解释型语言：Python，JavaScript，MATLAB

1.4.2 按照程序的运行方式分类

思考：Java是编译型语言还是解释型语言?



1.4.3 根据程序语言解决问题的方法分类

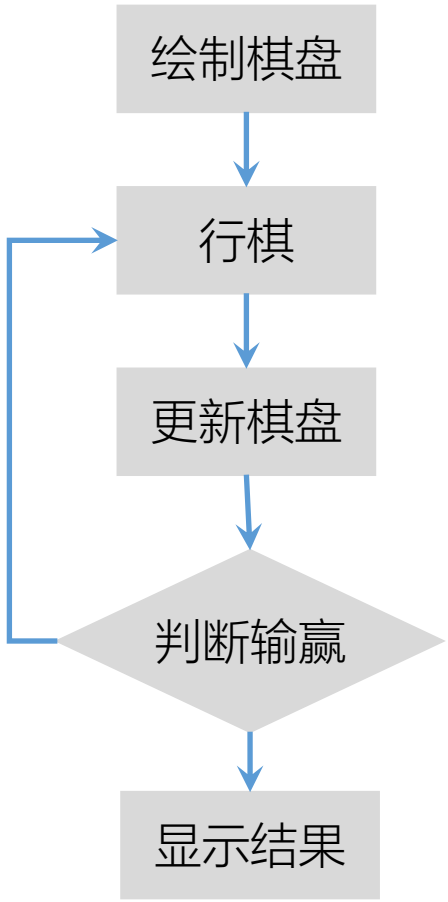
- 面向过程的程序语言
  - 按功能划分为若干个基本模块，形成一个树状结构
  - 各模块间的关系尽可能简单，功能上相对独立
  - 模块内部由顺序、选择和循环三种基本结构组成
  - 其模块化实现的具体方法是使用子函数
- 面向对象的程序语言
  - 将数据及其操作方法封装在一起，作为一个整体——对象
  - 对同类型对象抽象出其共性，形成类
  - 类通过一个简单的外部接口，与外界发生关系
  - 对象与对象之间通过消息进行通信
- 函数式语言
- 逻辑式语言
- 专用语言

1.4.3 根据程序语言解决问题的方法分类

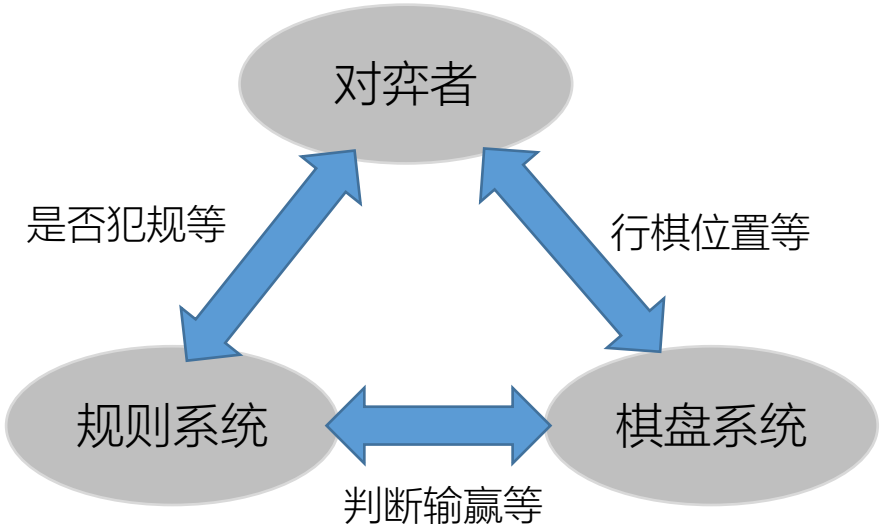
- 以下棋程序为例
- 面向过程：描述下棋的过程（逻辑）
  - 步骤：1. 绘制棋盘；2. 行棋；3. 更新棋盘；4. 判断输赢；
    5. 返回步骤2；6. 输出结果
  - 特点：步骤清楚，设计简单；但程序是一个整体，修改麻烦
- 面向对象：描述下棋涉及的对象及其行为关系
  - 对象：对弈者、棋盘系统、规则系统
  - 关系：对弈者获得用户的行棋位置后告知棋盘系统，棋盘系统在屏幕上更新棋局，同时利用规则系统判定棋局（犯规、输赢）
  - 特点：设计复杂；但框架确定后，修改和替换方便

1.4.3 根据程序语言解决问题的方法分类

面向过程



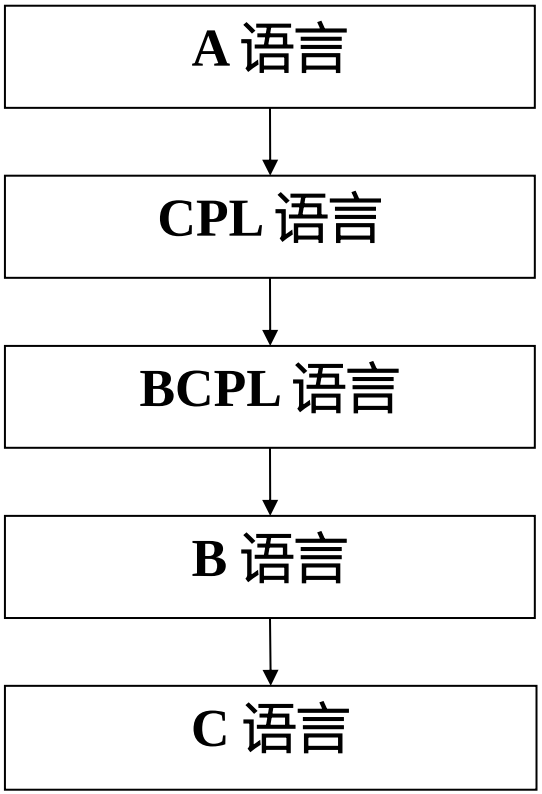
面向对象



1.5 C语言介绍

- C语言是一门面向过程的编译型“高级语言”。它既有高级语言的特点，又具有低级汇编语言的特点
- C语言可以作为系统设计语言，编写工作系统应用程序，也可以作为应用程序设计语言，编写不依赖计算机硬件的应用程序
- 为什么学习C语言?
  - C语言相比其它高级语言如（C++，Java，C#）是低级语言，通过它可以更好地了解计算机底层工作原理。比如数据在内存中是如何存储的，如何直接访问内存中的数据等等。
  - C语言是其他任何高级语言的基础，学好C语言，就可以更容易掌握其他语言。语言都是相通的，C更专注于语言的实质，而不需要分散更多精力在集成开发环境的使用和抽象的数据概念上。

1.5.1 C语言发展简史



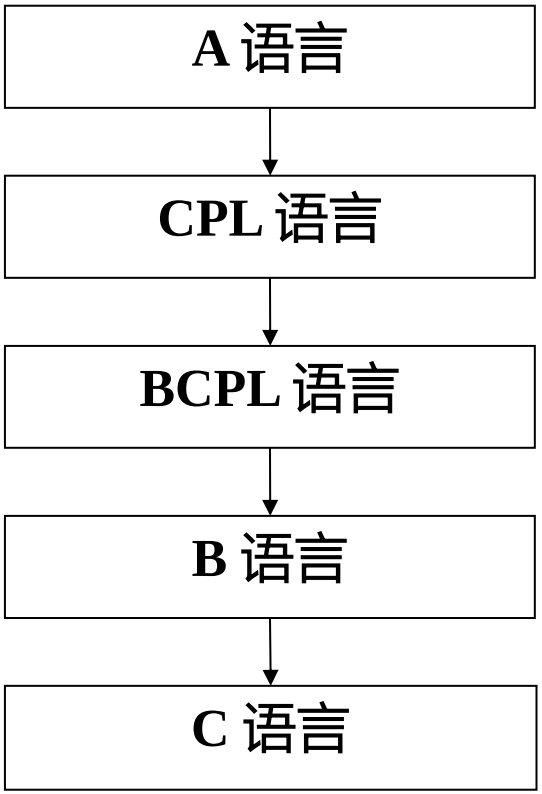
第1阶段：A语言。C语言的原型ALGOL 60语言，也成A语言。ALGOL60是一种面向问题的高级语言，它离硬件比较远，不适合编写系统程序。其特点是局部性、动态性、递归性和严谨性。

第2阶段：CPL语言。1963年，剑桥大学将ALGOL 60语言发展成为CPL语言（Combined Programming Language），CPL语言在ALGOL60的基础上与硬件接近了一些，但规模仍然比较宏大，难于实现。

第3阶段：BCPL语言。1967年，剑桥大学对CPL语言进行了简化，推出了BCPL语言（Basic Combined Programming Language）语言。BCPL语言是一种结构化程序设计语言，能够直接处理与机器本身数据类型相近的数据，具有与内存地址对应的指针处理方式。

第4阶段：B语言。1970 年代初期，美国贝尔实验室将BCPL进行了修改，设计出比较简单且接近硬件的语言，取名B语言。B语言只有一种数据类型，计算机字。大部分运算操作使用整数，其余操作使用一个复引用的内存地址。在许多方面B语言更像是一个早期版本的C语言，它还包括了一些库函数，其作用类似于C语言中的标准输入/输出函数库。

1.5.1 C语言发展简史



第5阶段：C语言。由于B语言过于简单，数据类型和功能有限，美国贝尔实验室在B语言的基础上最终设计出了一种新的语言，取名C语言，并试着以C语言重写Unix。1972年，丹尼斯·里奇（Dennis Ritchie）完成C语言的设计，并成功地用其编写操作系统，降低了代码修改难度。

C语言发展后续：

- 1978年，C语言先后移植到大、中、小、微型计算机上，风靡世界，成为最广泛的几种计算机语言之一。
- 1983年，美国国家标准委员会（ANSI）对C语言进行了标准化，于1989通过了一个C语言标准草案ANSI C。1990年，国际标准化组织（ISO）和国际电工委员会（IEC）把ANCI C标准定为C语言的国际标准，简称为C90标准。
- 随后ISO和IEC又陆陆续续地修改标准，如C99, C11, C17，最新的标准C2x预计将于2022年12月1日完成。新标准引入新的特性，丰富C语言的功能，方便用户的使用。

例：C99开始支持变长数组

```
1. #define N 5
2. int arr[N];
```

```
1. int n;
2. scanf("%d", &n);
3. int arr[n];
```

C99之前

C99及之后

1.5.2 C语言的特点

- 功能强大、适用范围大、可移植性好
  - 许多著名的系统软件都是由C语言编写的。C语言可以像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元。
  - C语言适合于多种操作系统，如DOS、UNIX等。对于操作系统、系统使用程序以及需要对硬件进行操作的场合，用C语言明显优于其它解释型高级语言，一些大型应用软件也是用C语言编写的。
- 运算符丰富
  - C语言的运算符包含的范围很广泛，共有34种运算符。C语言把括号、赋值、强制类型转换等都作为运算符处理。
  - C语言的运算类型极其丰富，表达式类型多样化。灵活使用各种运算符可以实现在其它高级语言中难以实现的运算。
- 数据结构丰富
  - C语言的数据类型有：整形、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等。能用来实现各种复杂的数据结构的运算。
  - 引入了指针概念，使程序效率更高，甚至可以直接访问内存地址。

1.5.3 C语言程序示例

```
1.  #include <stdio.h>                    /*包含标准输入输出头文件*/
2.  int main(int argc, char **argv)       /*主函数*/
3.  {                                     /*函数体开始*/
4.    printf("Hello World");              /*调用库函数，输出字符串常量*/
5.    return 0;                           /*返回值*/
6.  }                                     /*函数体结束*/
```

```
1.  #include <stdio.h>                    /*包含标准输入输出头文件*/
2.  #define C 1                           /*定义常量C的值为1*/
3.  int main(int argc, char **argv)       /*主函数*/
4.  {                                     /*函数体开始*/
5.    int x;                              /*定义整数类型变量x*/
6.    scanf("%d", &x);                    /*调用库函数，读入一个整数值，存入变量x中*/
7.    x = x + C;                          /*计算x+C的值，结果存入变量x中*/
8.    printf("Hello World: %d", x         /*调用库函数，输出结果*/
9.    return 0;                           /*返回值*/
10. }
```