Welcome

数据科学与大数据技术专业

计 算 机 系 统 基 础

上海体育学院经济管理学院

Wu Ying

回顾要点（I）

理解计算机

1. 个人计算机的硬件组成
2. 存储程序 原理
3. 可编程
4. 冯·诺依曼体系结构
5. 总线
6. 0 - 1 序列 指挥电路动作
7. 高级语言、汇编语言、机器码
8. 计算机发展阶段、特点以及应用
9. 总线概述、总线分类、系统总线分类、总线仲裁
10. 位、字节、地址、存储单元、地址总线位数与寻址范围

理解运算

1. 电路怎样实现运算，与或非逻辑运算
2. 传统逻辑、布尔代数、香农开关
3. CPU的演化
4. 与主存一起完成自动加法计算

深入 CPU 和 主存

1. 一条指令的执行过程
2. 程序，多条指令的连续执行
3. 冯·诺依曼机的基本工作原理
4. 指令和机器码（指令的分类、格式）
5. 模型机
6. 指令周期（Instruction Cycle）、机器周期（CPU/Machine Cycle）、时钟周期（Clock Cycle）
7. 微操作
8. 抽象

指令集和系统抽象层次

1. 指令集 + 指令集体系结构 = 指令系统
2. 计算机系统的抽象层次、不同用户
3. 操作系统、用户接口、编译与解释

回顾要点（II）

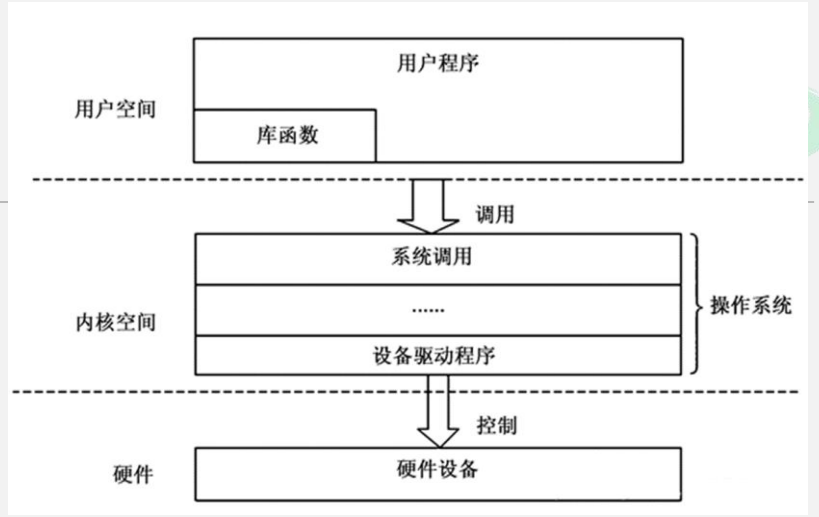
计算机系统

1. 硬件系统及软件系统的抽象层次
2. 软件系统及其分类（系统软件、支持软件、应用软件）
3. 应用操作：Windows tips、文字处理软件（Word、记事本、Markdown等）
4. 系统性能评价

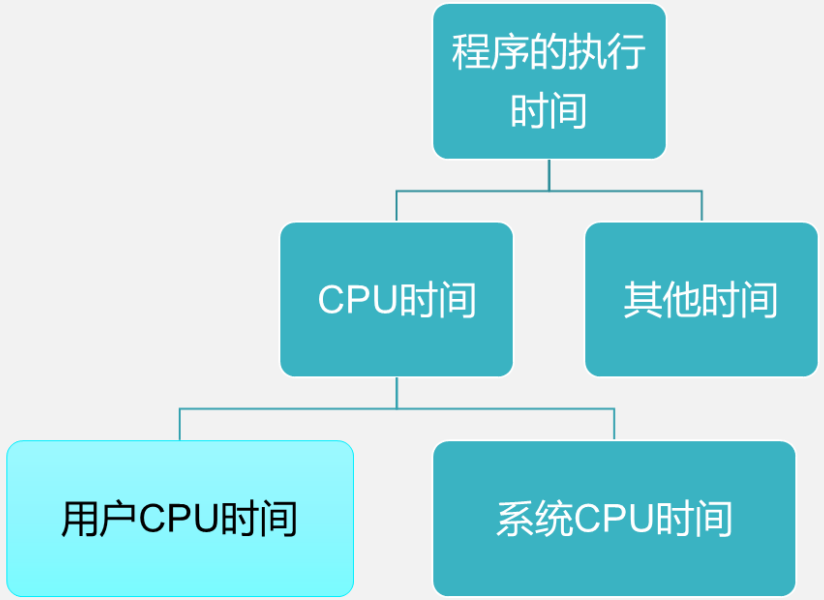
   程序执行时间、MIPS、Amdahl定律

数据的机器表示与处理

```
Welcome to Alibaba Cloud Elastic Compute Service !

Updates Information Summary: available
    3 Security notice(s)
        3 Moderate Security notice(s)
Run "dnf upgrade-minimal --security" to apply all updates.
Last login: Mon Nov  1 13:35:34 2021 from 118.31.243.210
[root@iZuf6an56zjdutq92u57jkZ ~]# time seq 1000000 | wc -l
1000000

real    0m0.020s
user    0m0.012s
sys     0m0.006s
[root@iZuf6an56zjdutq92u57jkZ ~]# time seq 1000000 | wc -l
1000000

real    0m0.019s
user    0m0.015s
sys     0m0.002s
[root@iZuf6an56zjdutq92u57jkZ ~]#
```



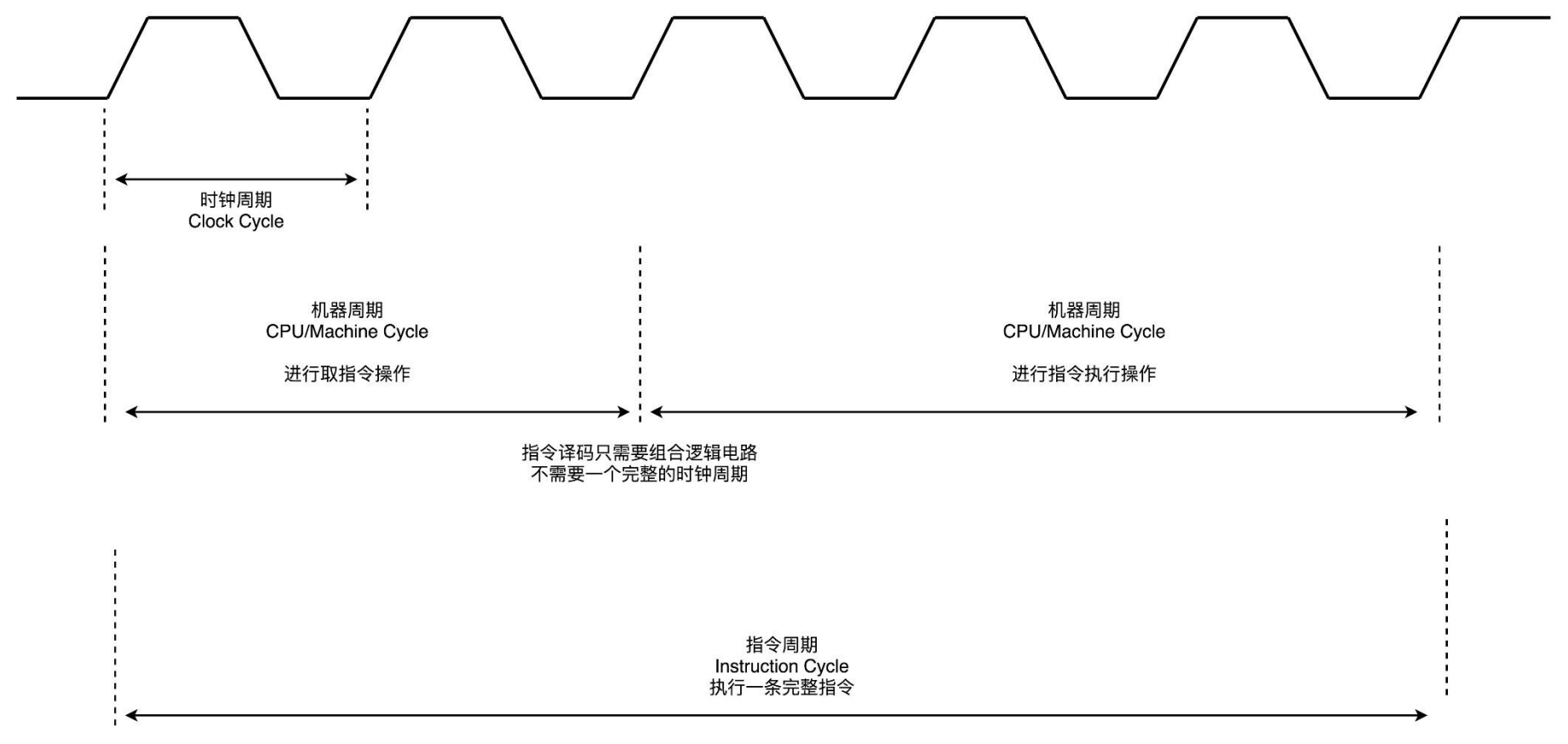
wc -l -(line) 显示行数



程序的用户CPU时间 = 程序指令所需要的总的时钟周期数 Clock Cycles × Clock Cycle Time

每条指令的平均时钟周期数（Cycles Per Instruction，简称 CPI）

- 一条指令的执行时间包含一个或多个时钟周期



程序的 CPU 执行时间 = 程序指令所需要的总的时钟周期数 Clock Cycles × Clock Cycle Time

【1.4.2　P20 】

程序的用户CPU时间 = 程序的总指令数 × CPI（指令执行所需平均周期数） × 时钟周期时间 Clock Cycle Time

【例1.1P20】

假设某个频繁使用的程序P在机器M1上运行需要10秒， M1的时钟频率为2GHz

设计人员想开发一台与M1具有相同ISA的新机器M2， **采用新技术可使M2的时钟频率增加，但同时也会使CPI增加**

假定程序P在M2上的时钟周期数是在M1上的1.5倍，则M2的时钟频率至少达到多少才能使程序P在M2上的运行时间缩短为6秒？

【解】

程序P在机器M1上耗用的时钟周期数为：

运行时间×时钟频率 = 10s * 2GHz = 20G

因此，程序P在M2上的时钟周期数为：1.5 * 20 G = 30 G

要使程序P在M2上的运行时间缩短到 6s,则 M2 的时钟频率至少应为：

程序总的时钟周期数 ÷ CPU时间 = 30G / 6s = 5G Hz

由此可见，M2的时钟频率是M1的2.5倍，但M2的速度却只是M1的1.67倍。

1.4.3 用指令执行速度进行性能评估

- MIPS (Million Instructions Per Second)　平均**每秒**执行**百万条**指令

机器A每秒能执行200万条指令，记做2 MIPS，机器B每秒能执行500万条指令，记做5 MIPS

- 程序的平均执行时间：

  每种指令所占的比例 * 该种指令执行所需要的时钟周期数

1.4.3 用指令执行速度进行性能评估

【例1.3 P22】

假设某程序P编译后生成的目标代码由A/B/C/D四类指令组成

它们在程序中所占的比例分别为：43%、21%、12%、24%

已知四类指令的CPI分别为1　2　2　2

重新对程序P进行编译优化，生成的新目标代码中A类指令条数减少了50%，其他类指令的条数没有变。

(1)编译优化前和优化后，程序的CPI各是多少？

(2)假定程序在一台主频为50MHz的计算机上运行，优化前后的MIPS各是多少？

1.4.3 用指令执行速度进行性能评估

【例1.3 P22】

假设某程序P编译后生成的目标代码由A/B/C/D四类指令组成

它们在程序中所占的比例分别为：43%、21%、12%、24%

已知它们的CPI分别为1/2/2/2，重新对程序P进行编译优化，生成的新目标代码中A类指令条数减少了50%，其他类指令的条数没有变。

(1)编译优化前后程序的CPI各是多少？

(2)假定程序在一台主频为50MHz的计算机上运行，优化前后的MIPS各是多少？

解：
优化后A类指令的条数减少了50%，因而各类指令所占比例分别计算如下：
A类指令：21.5 /（21.5+21+12+24） = 27%
B类指令：21 /（21.5+21+12+24）= 27%
C 类指令：12 /（21.5+21+12+24）= 15%
D 类指令：24/（21.5+21+12+24）= 31%

(1)
优化前程序的CPI为： 43% 1 + 21% *2 + 12%*2 + 24%*2 = 1.57
优化后程序的CPI为： 27% 1 + 27% *2 + 15%*2 + 31%*2 = 1.73
(2)
优化前的MIPS 为： 50MHz / 1.57 = 31.8MIPs
优化后的MIPS 为：50MHz / 1.73 =28.9MIPS

从MIPS数来看，优化后的程序执行速度反而变慢了。
这显然是错误的，优化后只减少了A类指令条数而其他指令数没有变，所以程序执行时间一定减少了。
从这个例子可以看出，用MIPS数来衡量性能是不可靠的

用MIPS进行性能评估，有时不准确不客观

优化后，从MIPS数来看，执行速度反而变慢了

但只减少了A类指令条数，其他指令条数未变，程序执行时间一定是减少了

计算机的运算速度与许多因素有关，如机器的主频、执行的操作类型等

MIPS—执行定点指令的速度，MFLOPS每秒完成浮点操作次数

- MFLOPS——Million FLOating-point operations Per Second　Mflop/s
  基于完成的浮点操作次数衡量

MFLOPS	**M**flop/s	10^6	mega 兆，百万
GFLOPS	**G**flop/s	10^9	giga 吉[咖]，千兆,十亿
TFLOPS	**T**flop/s	10^{12}	tera 太[拉]，万亿
PFLOPS	**P**flop/s	10^{15}	peta 拍
EFLOPS	**E**flop/s	10^{18}	exa 艾
Z			zetta 泽
Y			jotta 尧

Site:	National Super Computer Center in Guangzhou
Manufacturer:	NUDT
Cores:	3,120,000
Linpack Performance (Rmax)	33,862.7 TFlop/s
Theoretical Peak (Rpeak)	54,902.4 TFlop/s
Power:	17,808.00 kW
Memory:	1,024,000 GB
Interconnect:	TH Express-2
Operating System:	Kylin Linux
Compiler:	icc
Math Library:	Intel MKL-11.0.0
MPI:	MPICH2 with a customized GLEX channel

超星练习（15分钟）

1. 用一台40MHz处理机执行标准测试程序，若程序在处理机上运行所获得的有效CPI为1.55CPI，则相应的MIPS速率为　　MIPS。

25.8

超星热身

2. 多选 以下哪些术语是用来评价CPU的性能

A. CPI　　B. MDR　　C. FLOPS　　D. MIPS

A C D

3. 单选 将高级语言程序翻译成机器语言程序需借助于

A. 汇编程序　　B. 编译程序　　C. 连接程序　　D. 编辑程序

B

4. 单选计算机的运算速度与许多因素有关，如机器的主频、执行什么样的操作等。MIPS是衡量计算机运行速度普遍采用的计量单位。假想，机器A每秒能执行200万条指令，记做2MIPS，机器B每秒能执行500万条指令，记做5MIPS。在这两台机器上，运行同一程序时，两条机器上所用时间

A. B机器用的时间短　　B. A机器B机器所用时间相同　　C. A机器用的时间短　　D. 无法确定

D

5. 计算机中哪一部分负责指令译码?

A. 算术逻辑单元　B. 控制单元　　C. 存储器译码电路　　D. 输入输出译码电路

B

超星热身

6. 计算机的通用性使其可以求解不同的算术和逻辑问题，这主要取决于计算机的 _____ 。
A. 高速运算
B. 指令系统
C. 可编程性
D. 存储功能

D

超星热身

7. 在计算机存储单元中，一个 ASCII 码值占用的字节数为 _____ 。

A.1
B.2
C.4
D.8

A

超星热身

8. 单选 计算机系统中的软件系统包括系统软件和应用软件。下面关于软件的正确说法是 _____ 。
A. 软件就是没有固定形状的容器
B. 软件是计算机系统中运行的程序、及其使用的数据以及相应的文档的集合
C. 软件是指人力资源、人员素质等构成的系统
D. 计算机系统中看不见、摸不着的都是软件

B

9．下列描述中_____是正确的。
A．控制器能理解、解释并执行所有的指令及存储结果；
B．一台计算机包括输入、输出、控制、存储及算逻运算五个单元；
C．所有的数据运算都在CPU的控制器中完成；
D．以上答案都正确。

B

10．以下叙述中_____是错误的。
A．取指令操作是控制器固有的功能，不需要在操作码控制下完成；
B．所有指令的取指令操作都是相同的；
C．在指令长度相同的情况下，所有指令的取指操作都是相同的；
D．一条指令包含取指、分析、执行三个阶段

B

11、计算机系统的层次结构从内到外依次为（）。

A、硬件系统、系统软件、应用软件 B、系统软件、硬件系统、应用软件

C、系统软件、应用软件、硬件系统 D、应用软件、硬件系统、系统软件

A

12、【单选题】冯·诺依曼机工作的基本方式的特点是（）。

A、多指令流单数据流

B、按地址访问并顺序执行指令

C、堆栈操作

D、存贮器按内容选择地址

B

13.计算机操作的最小单位时间是（）

A

A. 时钟周期
B. 指令周期
C. CPU周期
D. 机器周期

14. MDR的位数反映了

B

A. 机器字长
B. 存储字长
C. 存储单元的个数
D. 存储容量

15.汇编语言与机器语言的对应关系为

A. 一对多
B. 多对一
C. 多对多

D

D. 一对一

1. (判断题)一般机器级，也称为机器语言级,属于计算机系统的第一级。

2. (判断题)计算机系统的操作系统以下层级编写程序采用的语言，基本是二进制数字化语言，机器执行和解释容易。编写程序采用的语言，基本是二进制数字化语言，机器执行和解释容易。

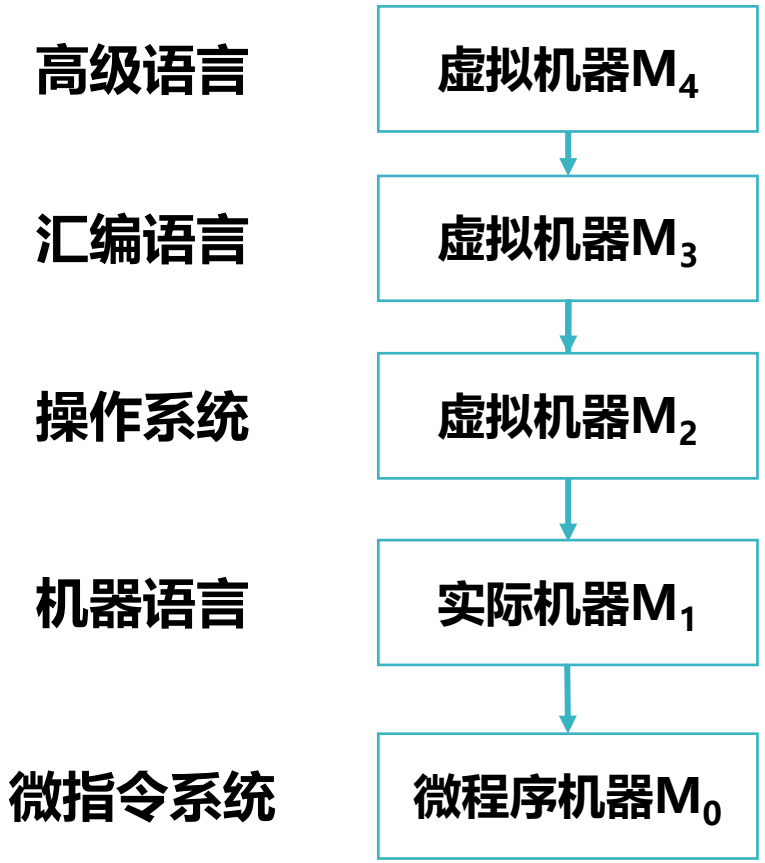
3. (判断题)随着大规模集成电路和计算机系统结构的发展，实体硬件机的功能范围在不断扩大。

4. (判断题)CPI表示每条指令的位数。

5. (判断题)FLOPS表示平均每秒执行多少百万条定点指令数。

6. (判断题)CPU的时钟频率越高，CPU的工作节奏越快，运算速度自然越高（ ）

7. (判断题)核数多的处理器运算速度比核数少的处理器要低（ ）


高级语言
虚拟机器M_4
汇编语言
虚拟机器M_3
操作系统
虚拟机器M_2
机器语言
实际机器M_1
微指令系统
微程序机器M_0

数据的机器级表示与处理

上周回顾——计算机系统的层次

- 软件系统
- 程序执行时间
- MIPS
- Amdahl定律

02

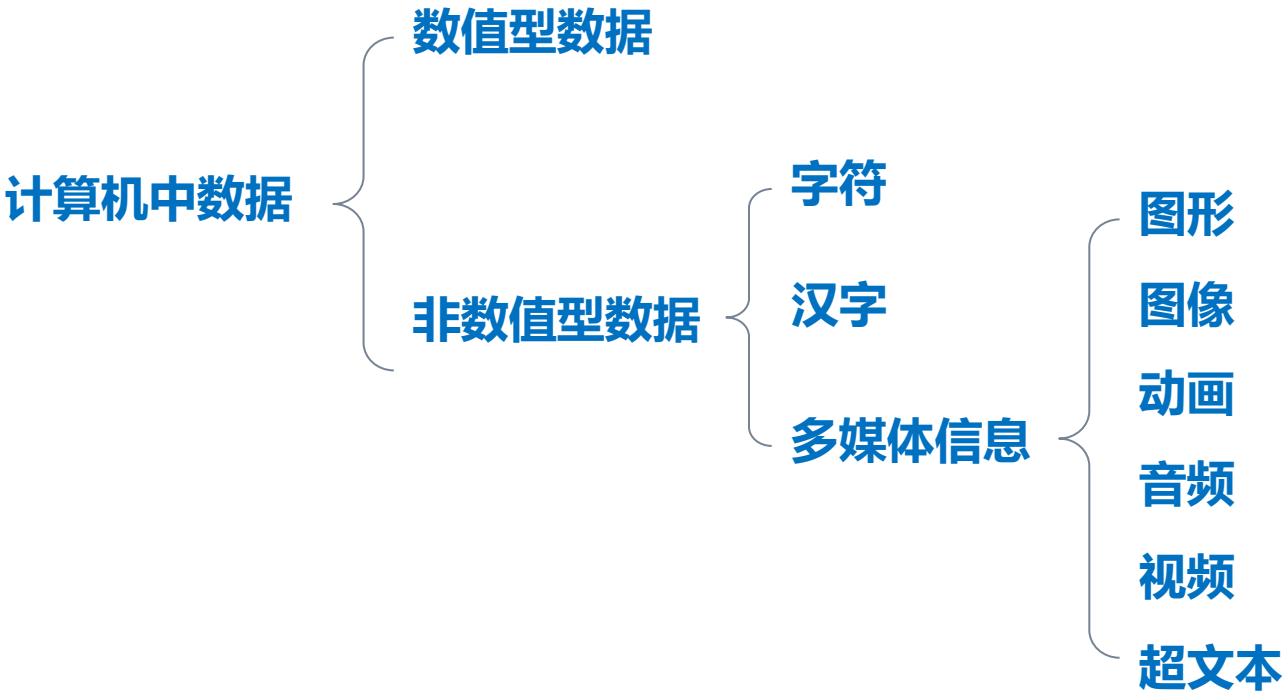
数制和编码

- 数字化编码

程序 = 算法 + 数据结构

对应到硬件层面
算法 — 计算机指令
数据 — 二进制：整数、浮点数、数组、字符、字符串……

数字化编码

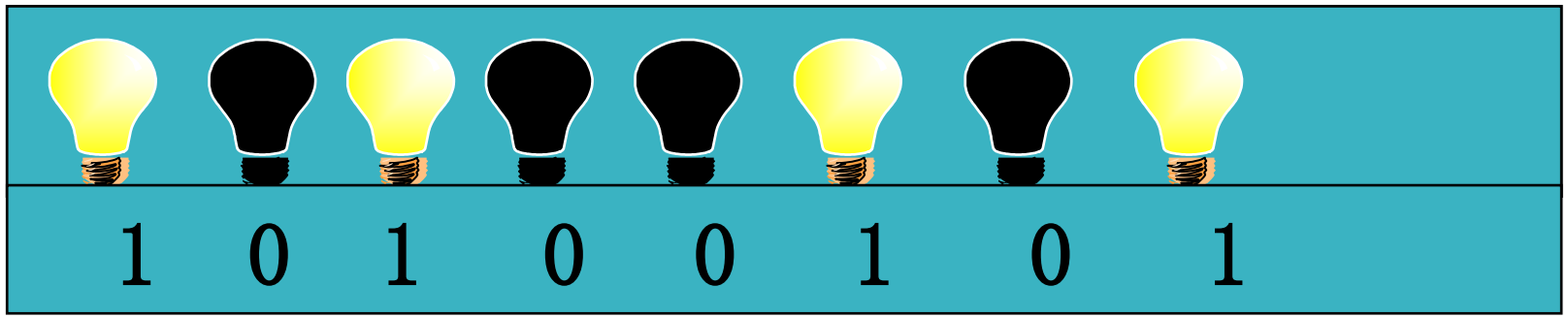


现实世界中的信息转换为计算机中的“离散”样本信息，然后对其用“0”“1”进行数字化编码

少量简单的基本符号，对大量复杂多样的信息进行一定规律的组合。

基本符号的种类和组合规则是信息编码的两大要素。

1’20”
Why 二进制



输入设备

数值　十 / 二进制转换

西文　ASCII 码

汉字　输入码 / 机内码转换

声音、图像　模 / 数转换

内存

数值型数据：

整数、浮点数

非数值型数据：

位串

输出设备

二 / 十进制转换　数值

西文字形码　西文

汉字字形码　汉字

数 / 模转换　声音、图像

指令系统能识别的基本类型数据

数制

数制也称为进位计数制。是指用一组固定的符号和统一的规则来表示数值的方法

常用的数制

- 十进制——符合人们习惯
- 二进制——计算机内部表示和存储数据，便于物理实现。
- 十六进制、八进制——便于书写，与二进制转换

数据表示

- 基数
  - 数制中所用到的数码的个数。
  - 基数简称“基”或“底”。常用字母R表示
  - Eg:

    十进制基数10：0,1,2,3,4,5,6,7,8,9

    二进制基数2：0,1

    八进制基数8：0,1,2,3,4,5,6,7

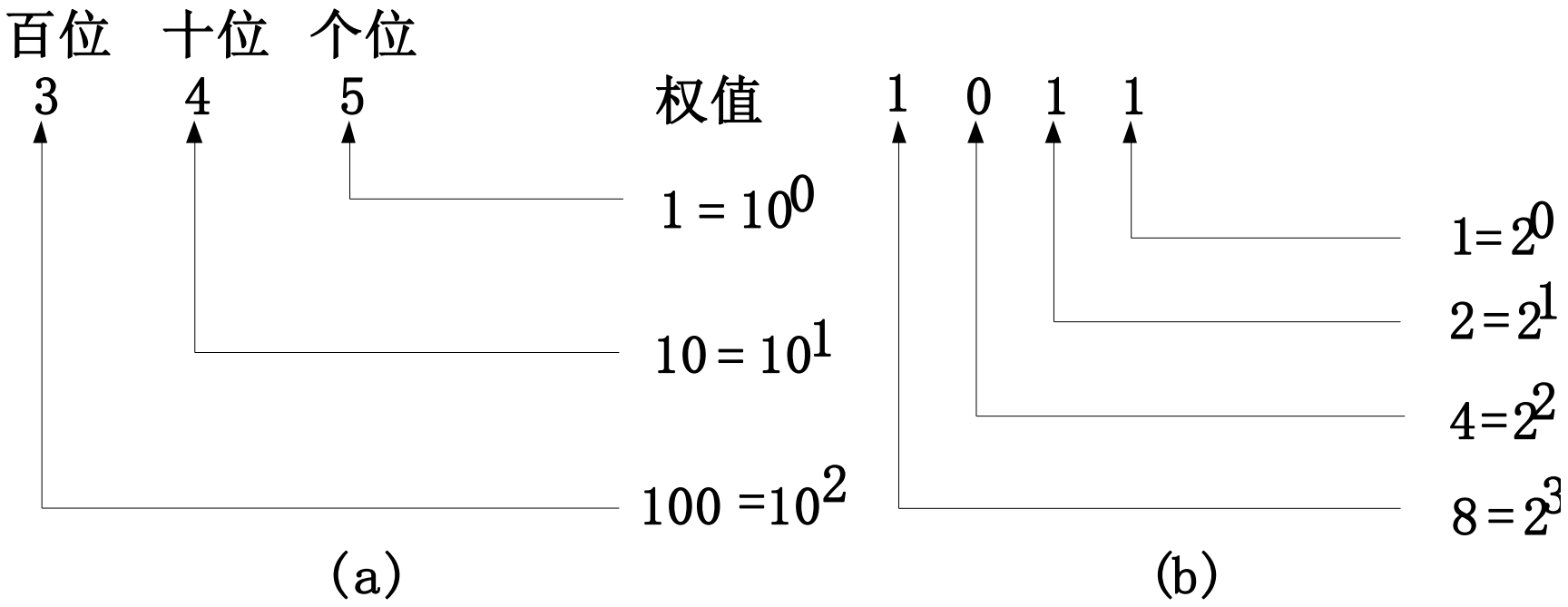
    十六进制基数16：0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

数据表示

- 位权
  - 一个数码处在不同位置所代表的值不同。每个数码所表示的数值等于该数码乘以一个与数码所在位置相关的常数，这个常数叫做位权。
  - 位权的大小：以基数为底、数码所在位置的序号为指数的整数次幂。
  - 例如：$345=3\times10^2+4\times10^1+5\times10^0$

数据表示

（a）十进制数345；（b）二进制数1011



数据表示

- 对于任意的 R 进制数
  - 位置计数法

$$(N)_R = a_{n-1}a_{n-2}\cdots a_1 a_0 . a_{-1}\cdots a_{-m}$$

  - 按权展开形式

$$(N)_R = a_{n-1}\times R^{n-1} + a_{n-2}\times R^{n-2} + \cdots + a_1\times R^1 + a_0\times R^0 + a_{-1}\times R^{-1} + \cdots + a_{-m}\times R^{-m}$$

(其中n为整数位数，m为小数位数，R为基数)

数据表示

- 按权展开形式
  - 十进制数

$(34958.34)_{10}=$

$3\times10^4+4\times10^3+9\times10^2+$

$5\times10^1+8\times10^0+3\times10^{-1}+4\times10^{-2}$

  - 二进制数

$(100101.01)_{2}=$

$1\times2^5+0\times2^4+0\times2^3+$

$1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}$

特点：

- 用十个数码表示——0、1、2、3、4、5、6、7、8、9
- 遵循“逢十进一”的规则

权展开式

对任意一个n位整数和m位小数的十进制数D，可表示为：

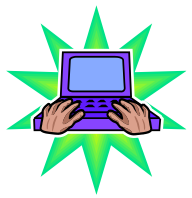
$$D_m = D_{n-1} \cdot 10^{n-1} + D_{n-2} \cdot 10^{n-2} + \cdots + D_0 \cdot 10^0 + D_{-1} \cdot 10^{-1} + \cdots + D_{-m} \cdot 10^{-}$$

例：将十进制数314.16写成展开式形式

解：

$$314.16 = 3\times10^2 + 1\times10^1 + 4\times10^0 + 1\times10^{-1} + 6\times10^{-2}$$

$$= 300+10+4+0.1+0.06$$



十进制数是人们最习惯使用的数制，在计算机中一般把十进制数作为输入输出的数据形式。

特点：

- 用两个数码表示——0、1
- 遵循“逢二进一”的规则

计算机可直接识别的进制

权展开式

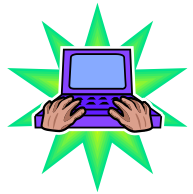
对任何一个n位整数m位小数的二进制数，可表示为：

$$D=B_{n-1}\cdot 2^{n-1}+B_{n-2}\cdot 2^{n-2}+\cdots+B_0\cdot 2^0+B_{-1}\cdot 2^{-1}+\cdots+B_{-m}\cdot 2^{-m}$$

例：将二进制数$(1101.01)_2$写成展开式形式，它代表多大的十进制数？

解：$(1101.01)_2 = 1\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0+0\times 2^{-1}+1\times 2^{-2}$

$= 8+4+0+1++0+0.25=(13.25)_{10}$



二进制数使用的数码少，只有0和1，用电器元件的状态来表示既方便又可靠，在计算机内部存储和运算中使用，运算简单，工作可靠。

特点：

- 用八个数码表示——0、1、2、3、4、5、6、7、8
- 遵循“逢八进一”的规则

权展开式

$$D=Q_{n-1}\cdot 8^{n-1}+Q_{n-2}\cdot 8^{n-2}+\cdots+Q_0\cdot 8^0+Q_{-1}\cdot 8^{-1}+\cdots+Q_{-m}\cdot 8^{-m}$$

例：八进制数$(317)_8$代表多大的十进制数？

解： $(317)_8 = 3\times 8^2 + 1\times 8^1 + 7\times 8^0$

$= 192+8+7=(207)_{10}$

八进制接近十进制，且与二进制转换方便，常用来对二进制数的“缩写”，如：将$(110111001101)_2$写成$(6715)_8$，便于对二进制数的表示和记忆。

特点：

- 用十六个数码表示——0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F

权展开式

对任何一个n位整数m位小数的十六进制数，可表示为：

$$D_m = H_{n-1} \cdot 16^{n-1} + H_{n-2} \cdot 16^{n-2} + \cdots + H_0 \cdot 16^0 + H_{-1} \cdot 16^{-1} + \cdots + H_{-m} \cdot 16^{-}$$

例：十六进制数$(3C4)_{16}$代表多大的十进制数？

解：$(3C4)_{16} = 3\times16^2 + 12\times16^1 + 4\times16^0$

$= (964)_{10}$

在表示同一量值时，十六进制数最短，如将$(110111001101)_2$写成$(DCD)_{16}$，且与二进制转换方便，因此十六进制数常用来在程序中表示二进制数或地址。

例：

$$(1011.01)_2 = 1\times2^3+0\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}$$

$$= 8+0+2+1++0+0.25=(11.25)_{10}$$

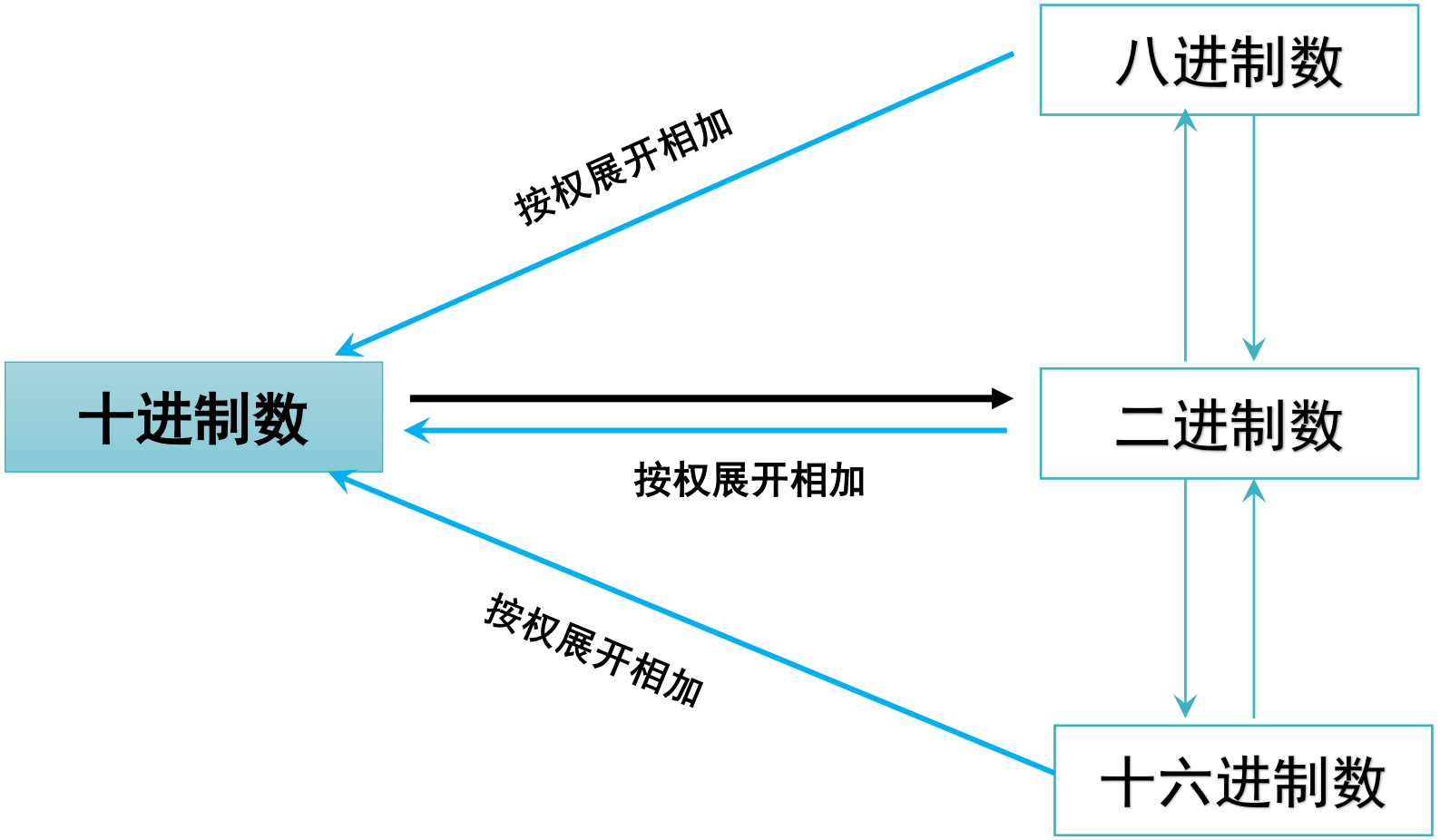
$$(157)_8 = 1\times8^2+5\times8^1+7\times8^0$$

$$= 64+40+7=(111)_{10}$$

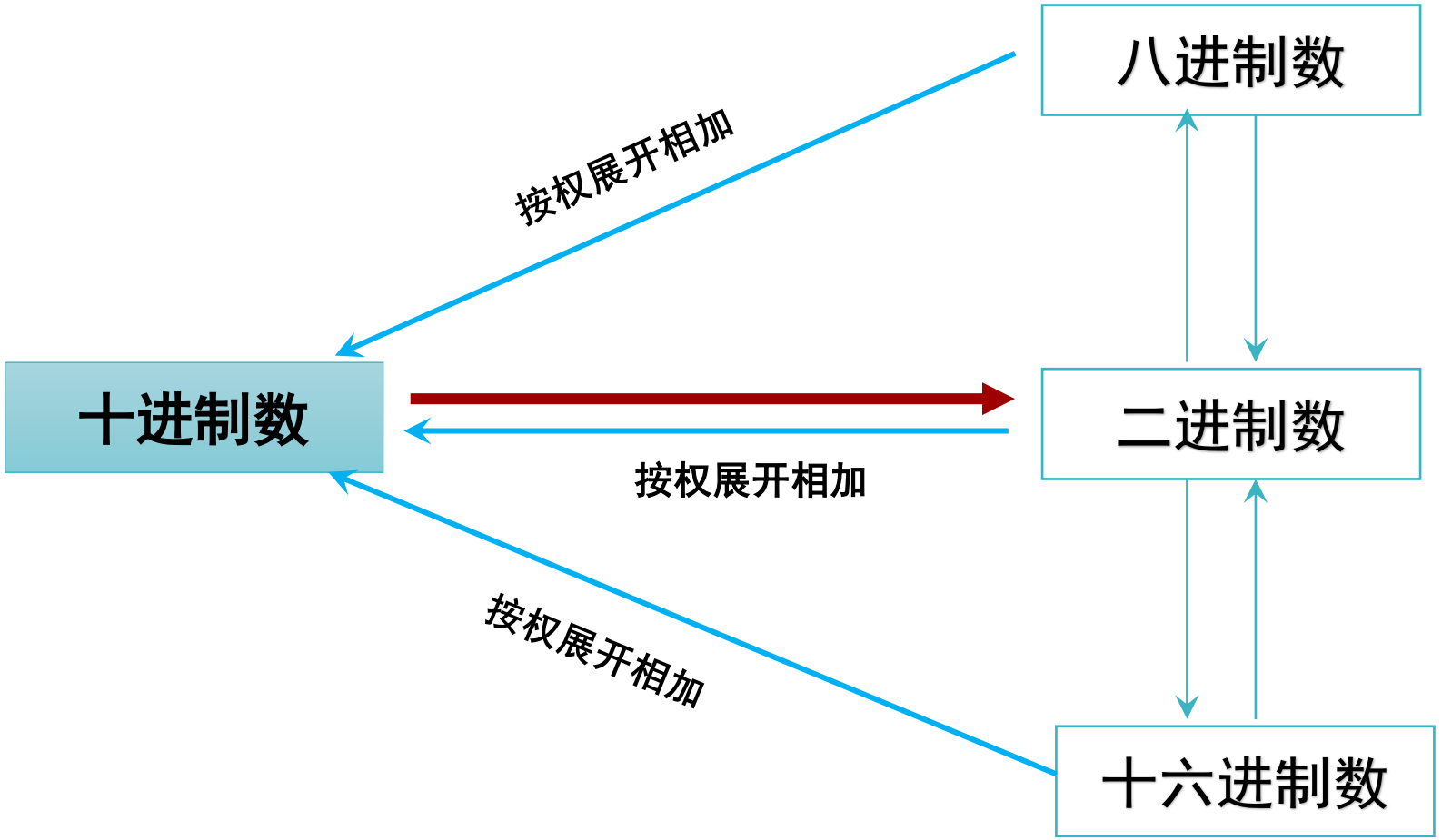
$$(2A4)_{16} = 2\times16^2+10\times16^1+4\times16^0$$

$$= 512+160+4=(676)_{10}$$

数制之间的转换



数制之间的转换



数制之间的转换

- 十进制转二进制

  方法:

  - 整数部分：除2倒取余，**至商为零**；所得的余数倒序排列。
  - 小数部分：乘2顺取整，**直到小数点后为零或者达到精度要求为止**；乘积的整数部分顺序排列。

数制之间的转换

- 十进制整数转二进制整数
  - 例 求$(19)_{10}$的二进制数值。

解:

因此，$(19)_{10} = (10011)_2$

数制之间的转换

- 十进制小数转二进制小数
  - 例 求$(0.6875)_{10}$的二进制数值。

解:

	整数	
0.6875 × 2		
1.3750	1	(高位)
0.3750 × 2		
0.7500	0	
× 2		
1.5000	1	
0.5000 × 2		
1.0000	1	(低位)

因此，$(0.6875)_{10} = (0.1011)_2$

数制之间的转换

- 十进制小数转二进制小数
  - 十进制小数转换为二进制小数过程中，有时会出现乘积的小数部分总不等于0的情况，或者出现循环小数的情况
  - 如：$(0.2)_{10}$ =(0.001100110011…)$_2$
  - 这样的情况下，乘2过程的结束由所要求的转换精度确定。
  - 一般当要求二进制数取m位小数时，可求出m+1位，然后对最低位作0舍1入处理。

数制之间的转换

- 十进制小数转二进制小数
  - 例 求$(0.323)_{10}$的二进制数值。 (保留4位小数)

解:

$$
\begin{array}{r}
0.323 \\
\times\quad 2 \\
\hline
0.646 \\
\times\quad 2 \\
\hline
1.292 \\
\times\quad 2 \\
\hline
0.584 \\
\times\quad 2 \\
\hline
1.168 \\
\times\quad 2 \\
\hline
0.336
\end{array}
$$

高位

低位

因此，$(0.323)_{10} = (0.0101)_2$

数制之间的转换

- 十进制数转二进制数
  - 例 将$(237.625)_{10}$转换成二进制数。

解:

整数除2取余

2	237	1
2	118	0
2	59	1
2	29	1
2	14	0
2	7	1
2	3	1
2	1	1
	0	

小数乘2取整

	0.625	
	× 2	
1	1.250	
	0.25	
	× 2	
0	0.50	
	× 2	
1	1.0	

则，$(237.625)_{10} = (11101101.101)_2$

数制之间的转换

练习：

将十进制整数$(105)_{10}$转换为二进制整数。

$(105)_{10} = (1101001)_2$

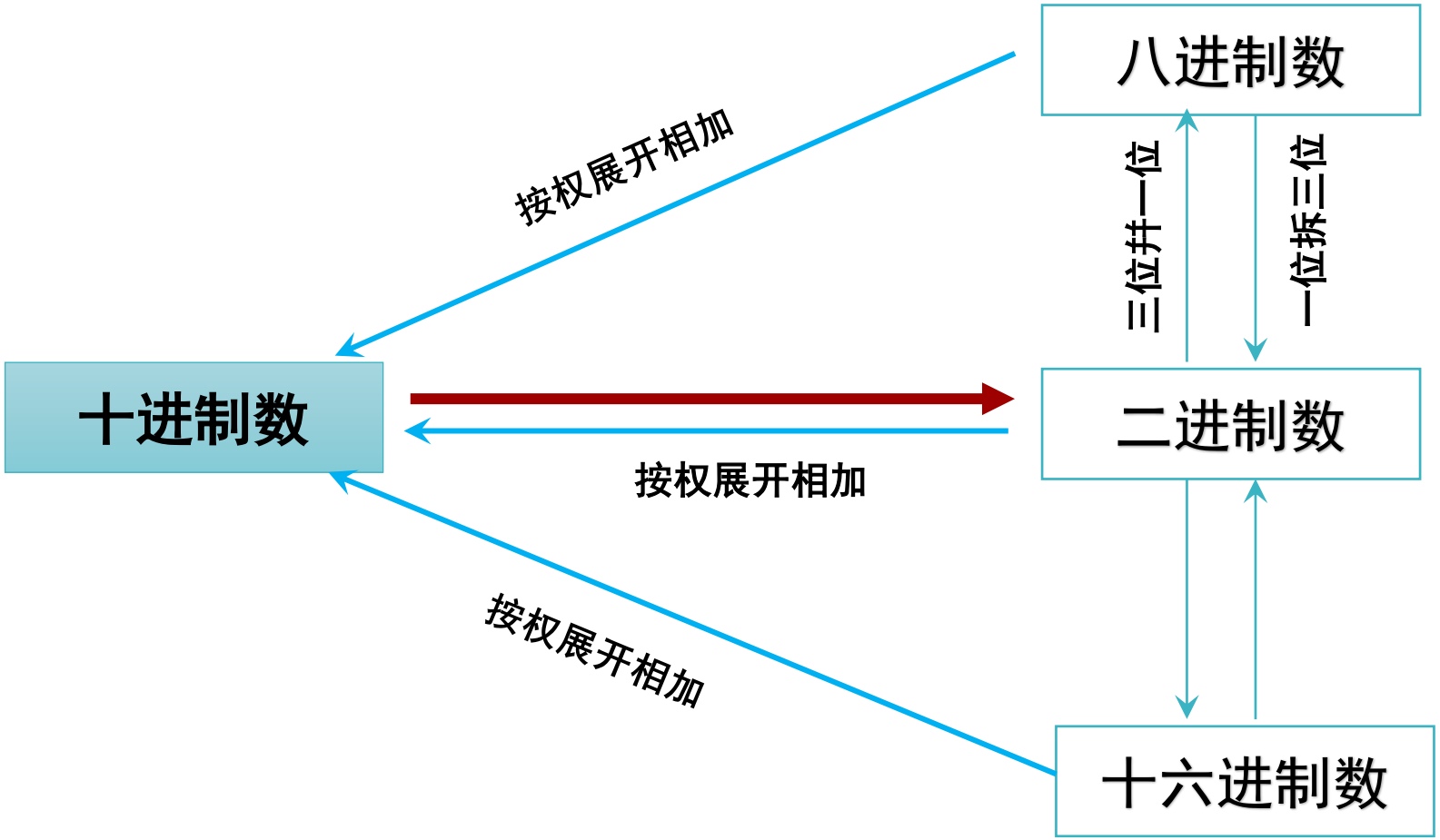
将十进制小数$(0.8125)_{10}$转换为二进制小数。

$(0.8125)_{10} = (0.1101)_2$

数制之间的转换

- 十进制数转R进制数
  - 整数部分：除R倒取余，**直到商为零。**
  - 小数部分：乘R顺取整，**直到小数点后为零或者达到精度为止。**
- 练习：
  - 将十进制数301.6875转换为十六进制数，和八进制数。

数制之间的转换

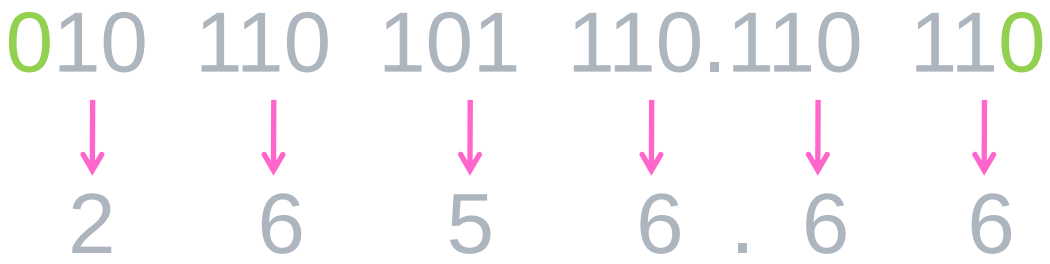


数制之间的转换

- 二进制转八进制
  - “三位一并”法
  - 方法：从小数点开始分别往两边，整数部分自右向左，小数部分自左向右，按每三位为一组，不足三位用0补齐，每组用相应的八进制数写出。
- 八进制转二进制
  - “一分为三”法
  - 方法：每位八进制数用三位二进制数代替。

数制之间的转换

- 二进制转八进制
  - 例 将（10110101110.11011）$_2$转换为八进制数。



则（10110101110.11011）$_2$ = （2656.66）$_8$

数制之间的转换

- 八进制转二进制
  - 例 将（6237.431）$_8$转换为二进制数。

6	2	3	7	.	4	3	1
↓	↓	↓	↓		↓	↓	↓
110	010	011	111	.	100	011	001

则（6237.431）$_8$ = （110010011111.100011001）$_2$

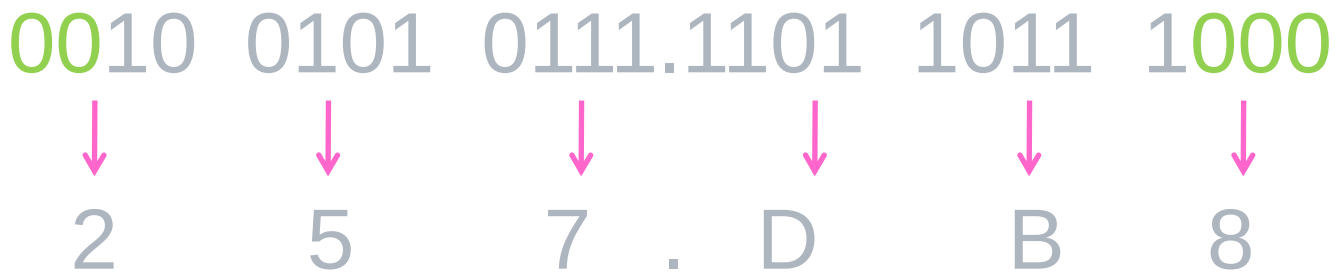
注意：一定要写够三位二进制。

数制之间的转换

- 二进制转十六进制
  - “四位一并”法
  - 方法：从小数点开始分别往两边，整数部分自右向左，小数部分自左向右，按每四位为一组，不足四位用0补齐，每组用相应的十六进制数写出。
- 十六进制转二进制
  - “一分为四”法
  - 方法：每位十六进制数用四位二进制数代替。

数制之间的转换

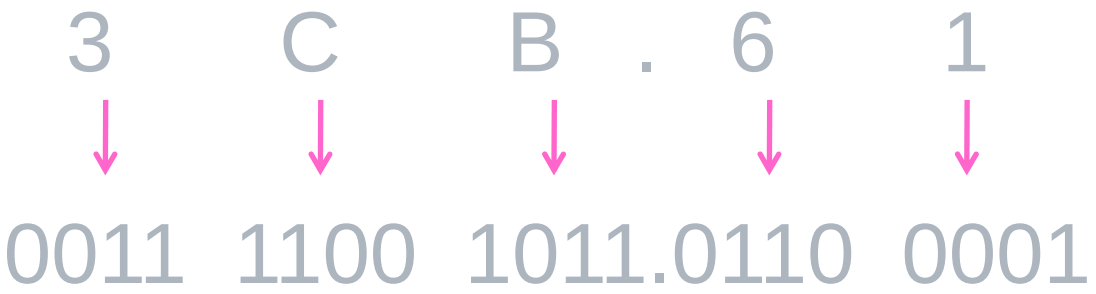
- 二进制转十六进制
  - 例 将（1001010111.1101101111）$_2$转换为十六进制数。



则（1001010111.1101101111）$_2$ = （257.DB8）$_{16}$

数制之间的转换

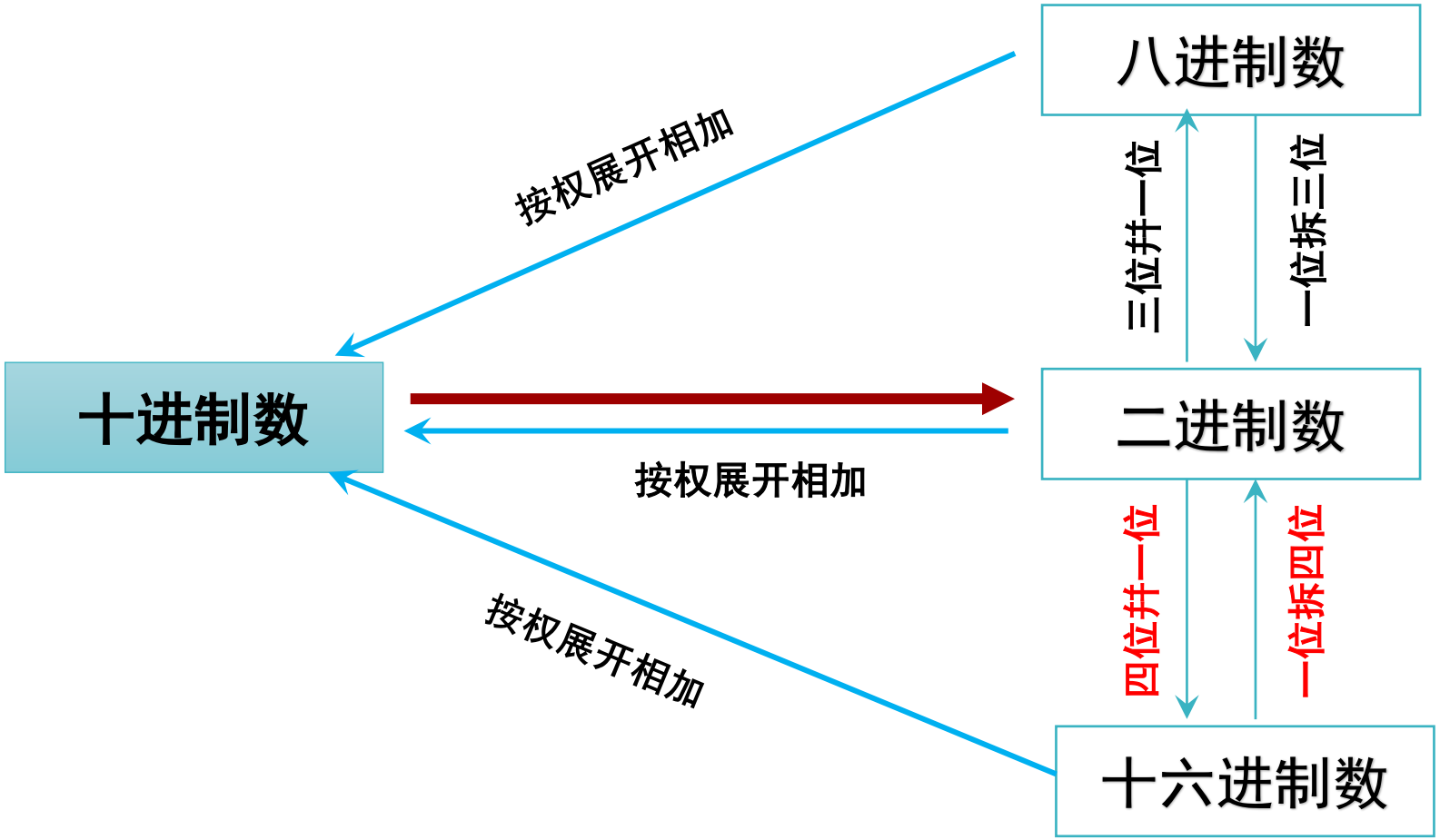
- 十六进制转二进制
  - 例2-9 将（3CB.61）$_{16}$转换为二进制数。



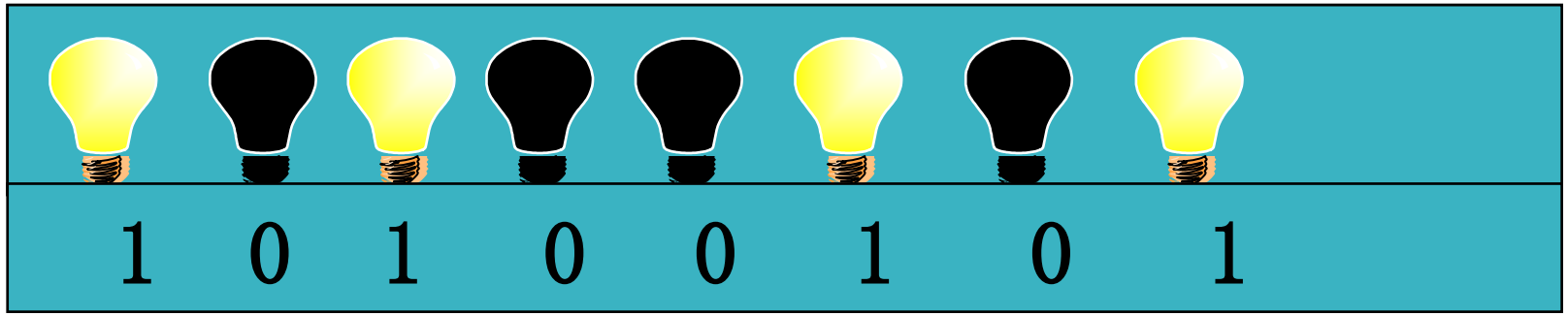
则（3CB.61）$_{16}$ = （1111001011.01100001）$_{2}$

注意：一定要写够四位二进制。

数制之间的转换


八进制数
三位并一位
一位拆三位
按权展开相加
十进制数
二进制数
按权展开相加
四位并一位
一位拆四位
按权展开相加
十六进制数

数据的机器级表示与处理



输入设备　　输出设备

数值　十 / 二进制转换

西文　ASCII 码

汉字　　输入码 / 机内码转换

声音、图像　模 / 数转换

内存

数值型数据:

整数、浮点数

非数值型数据:

位串

二 / 十进制转换　　数值

西文字形码　　西文

汉字字形码　　汉字

数 / 模转换　　声音、图像

数据的机器级表示与处理

01

数制和编码

- 编码
- 数制（基数、位权）
- 其他进制转十进制
- 十进制转二进制
- 八进制、十六进制与二进制互相转换

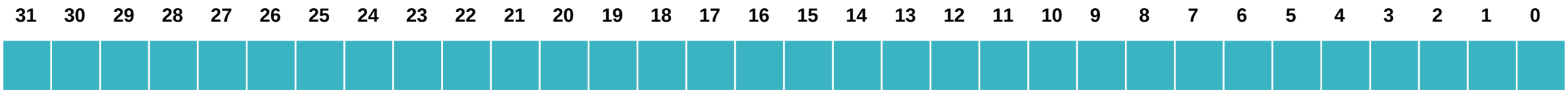
02

定点与浮点表示

- 定点小数与定点整数
- 浮点表示
- 定点数的编码表示
- 原码表示、补码表示

定点与浮点表示

	1	0	0	1	.	0	1
位序	3	2	1	0	.	1	2



32位的二进制小数，小数点位置有 ？ 种可能

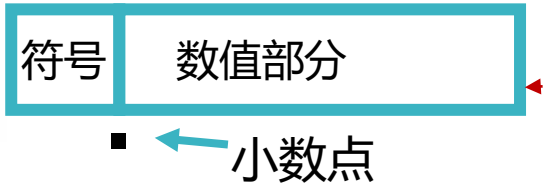
记录这个位置信息，至少需要？ 位

2.1.3 定点与浮点表示

在计算机中，小数点的位置事先约定好

定点数 小数点位置约定在固定位置

定点小数
(纯小数)



定点整数
(纯整数)

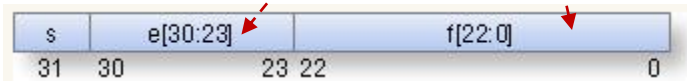
小数点位置约定为可浮动的数 **浮点数**

尾数 M

阶码 E

$$X = (-1)^s \times M \times R^E$$

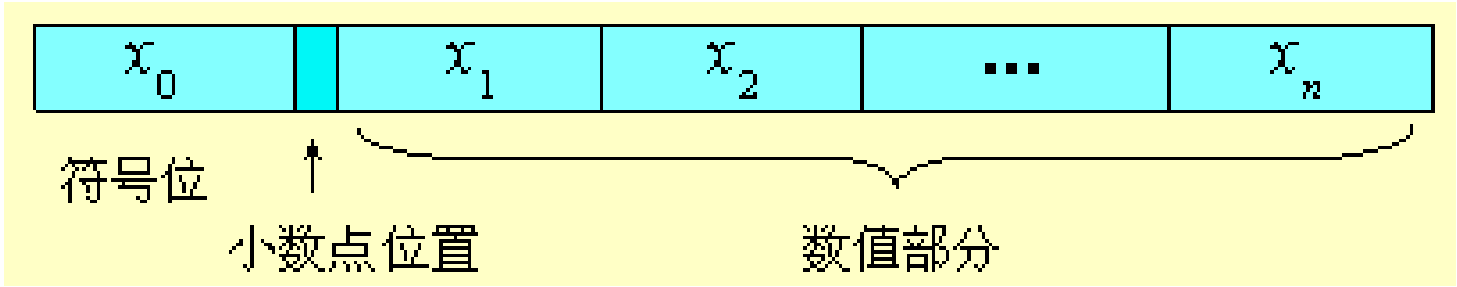
s 为 0或1



IEEE单精度浮点数存储：23位小数f、8位偏置指数e、1位符号s

定点小数

定点小数是纯小数，约定小数点位于符号位后（带符号的定点小数）



定点小数，绝对值最小的非零数 ？

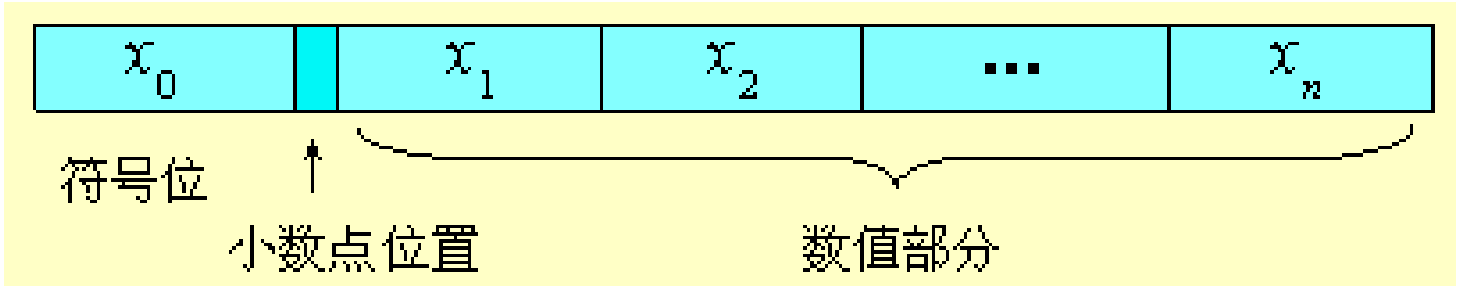
符号	1	0	0	0
位序 .	1	2	3	4

定点小数，绝对值何时最大？

定点小数的表示范围？

定点小数

定点小数是纯小数，约定小数点位于符号位后（带符号的定点小数）



定点小数，绝对值最小的非零数 ？ $|x|_{min} = 2^{-n}$

符号	1	0	0	0
位序 .	1	2	3	4

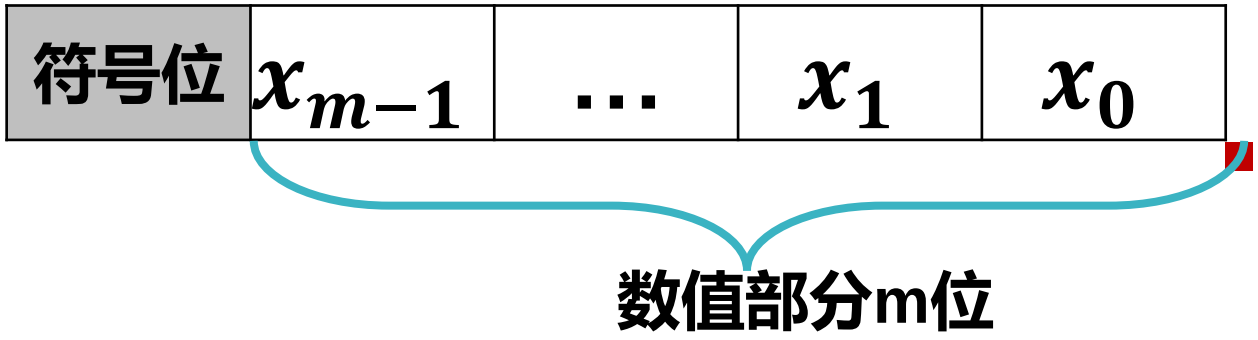
定点小数，绝对值何时最大？ $|x|_{max} = 1-2^{-n}$

定点小数的表示范围

$$2^{-n} \le |x| \le 1 - 2^{-n}$$

定点整数

定点整数是纯整数，约定小数点位于有效数值部分最低位之后



定点整数，绝对值最小的非零数 ？ $|x|_{min} = ?$

符号	0	0	0	1
位序	3	2	1	0 .

定点整数，绝对值何时最大？ $|x|_{max} = ?$

定点整数的表示范围？

定点整数

定点整数是纯整数，约定小数点位于有效数值部分最低位之后

符号位	x_{m-1}	…	x_1	x_0

数值部分

定点整数，绝对值最小的非零数 ？ $|x|_{min} = 1$

符号	0	0	0	1
位序	3	2	1	0 .

定点整数，绝对值何时最大？ $|x|_{max} = 2^m - 1$

定点整数的表示范围

$$1 \le |x| \le 2^m - 1$$

$$-(2^m - 1) \le x \le 2^m - 1$$

- 当数据小于定点数能表示的最小值时，计算机将它们作0处理，称为下溢
- 大于定点数能表示的最大值时，计算机将无法表示，称为上溢
- 上溢和下溢统称为溢出

数据的机器级表示与处理

01

数制和编码

- 编码
- 数制（基数、位权）
- 其他进制转十进制
- 十进制转二进制
- 八进制、十六进制与二进制互相转换

02

定点与浮点表示

- 定点小数与定点整数
- 浮点表示，规格化，IEEE754
- 定点数的编码
  - 原码表示
  - 补码表示、特殊数据的补码表示
- 整数的表示

2.1.3 定点与浮点表示

在计算机中，小数点的位置事先约定好

定点数 小数点位置约定在固定位置

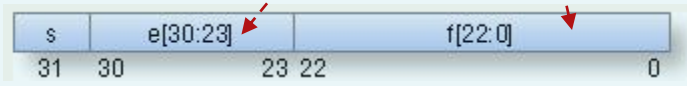
- 定点小数(纯小数)：符号 | 数值部分 ■（小数点）
- 定点整数(纯整数)：符号 | 数值部分 ■（小数点）

小数点位置约定为可浮动的数 **浮点数**

- 尾数 M
- 阶码 E

$$X = (-1)^s \times M \times R^E$$

s 为 0或1



IEEE单精度浮点数存储：23位小数f、8位偏置指数e、1位符号s

浮点表示

任意一个 R 进制数 X，总可以写成

s 为 0或1

$$X = (-1)^{s} \times M \times R^{E}$$

阶码 / 指数 / 阶
E
R进制定点纯整数

阶码决定小数点在数据中的位置，因而决定了浮点数的表示范围

尾数 M
定点纯小数

有效数字的位数，决定了浮点数的表示精度

25.65×10^{1}	$\mathbf{0.01101 \times 2^{-3}}$
2.565×10^{2}	$\mathbf{0.00001101 \times 2^{0}}$
0.2565×10^{3}	$\mathbf{1.101 \times 2^{-5}}$

若不对浮点数的表示作出明确规定，同一个浮点数的表示就不是唯一的。

浮点数的规格化 【2.3.2 P45】

浮点数尾数的位数决定了浮点数的有效位数

有效位数越多，数据的精度越高

尽可能多地保留有效数字的位数，即有效数字尽量占满尾数数位

尾数部分最高位为非零数字

浮点数的表示唯一

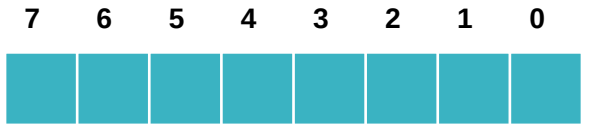
规格化

0.011011×2^{-3}

0.000011011×2^{0}

0.11011×2^{-4}

1.1011×2^{-5}



尾数右移一位（小数点左移一位），阶码加1——右规　右规时阶码会增加，因此阶码可能溢出

尾数左移一位（小数点右移一位），阶码减1——左规　$\pm 0.0\ldots0bb\ldots b$ ，左规

IEEE754 浮点数标准【2.3.3 P45-46】

规定：小数点前总是“1”，可隐含。

隐含一位后，使得单精度格式的23位尾数能够表示24位有效数字，

双精度52位尾数能够表示53位有效数字

单精度，偏置常数为127

0.011011×2^{-3}

0.000011011×2^{0}

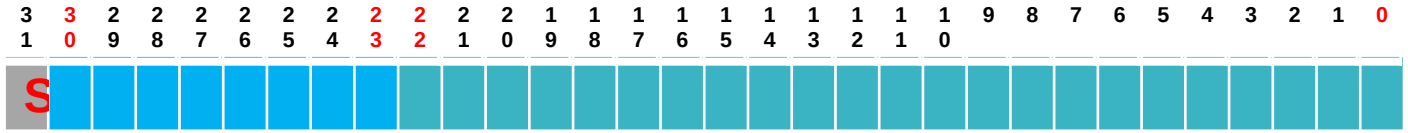
0.11011×2^{-4}

1.1011×2^{-5}

$X = (-1)^{s} \times M \times R^{E}$

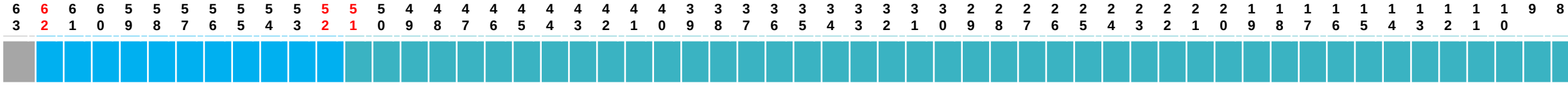
$(-1)^{S} \times 1.M \times 2^{(E-127)}$

32位单精度格式



1 bit　8 bits 阶码　23 bits 尾数

64位双精度格式

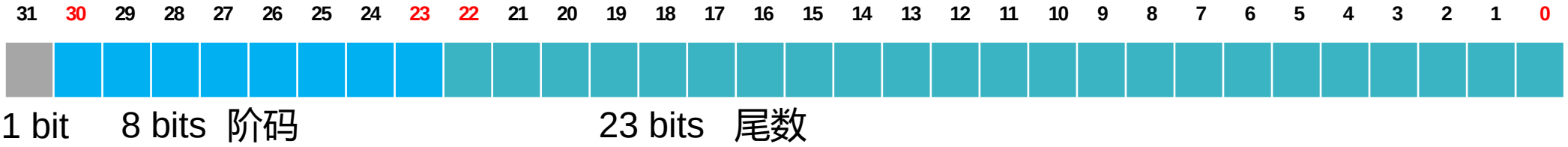


1 bit　11 bits 阶码　52 bits 尾数

IEEE754 浮点数标准【2.3.3 P45-46】

BEE00000H is the hex. Rep. Of an IEEE 754 SP FP number

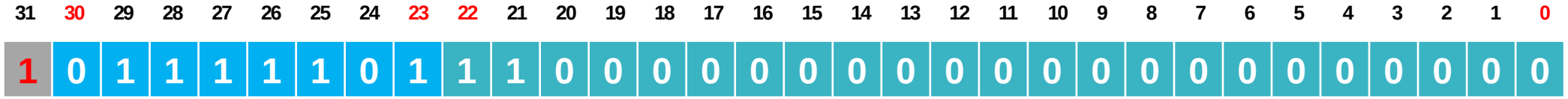
32位单精度格式



IEEE754 浮点数标准【2.3.3 P45-46】

BEE00000H is the hex. Rep. Of an IEEE 754 SP FP number

32位单精度格式



1 bit　8 bits 阶码　23 bits 尾数

符号位: 1 => 负数　-1

阶码:

$(01111101)_2 = (125)_{10}$

单精度偏置常数为127: 125 - 127 = -2

尾数: $(1.11000000000000000000)_2$

该单精度数二进制数对应的十进制值为:

$-(1.11000000000000000000)_2 \times 2^{-2} = -(0.0111)_2 = -(1 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}) = -0.4375$

“Father” of the IEEE 754 standard

直到80年代初，各个机器内部的浮点数表示格式还没有统一，因而相互不兼容，在机器之间传送数据时带来麻烦

1970年代后期，IEEE成立委员会着手制定浮点数标准

1985年完成浮点数标准IEEE 754的制定

This standard was primarily the work of one person, UC Berkeley math professor William Kahan.

现在所有计算机都采用IEEE 754来表示浮点数

www.cs.berkeley.edu/~wkahan/ieee754status/754story.html

Prof. William Kahan

1989
ACM Turing
Award Winner!

浮点数X的范围

s 为 0或1

阶码 E
定点纯整数

$$X = (-1)^{s} \times M \times R^{E}$$

尾数 M
定点纯小数

绝对值最小的非零数 ？ $|x|_{min}$ ？

绝对值最大的数 ？ $|x|_{max}$ ？

定点小数的表示范围

$$2^{-n} \le |x| \le 1 - 2^{-n}$$

定点整数的表示范围

$$1 \le |x| \le 2^{m}-1$$

$$-(2^{m}-1) \le x \le 2^{m}-1$$

浮点数 X 的范围

s 为 0或1

阶码 E
定点纯整数

$$X = (-1)^{s} \times M \times R^{E}$$

尾数 M
定点纯小数

绝对值最小的非零数 ？ $|x|_{min}$

$$0.0\ldots01 \times 2^{-111\ldots1} = 2^{-n} \times 2^{-(2^m-1)}$$

$$= \frac{2^{-n}}{2^{(2^m-1)}}$$

绝对值最大的数 ？ $|x|_{max}$

$$0.1\ldots11 \times 2^{111\ldots1} = (1-2^{-n}) \times 2^{(2^m-1)}$$

浮点数X的表示范围

$$\frac{2^{-n}}{2^{(2^m-1)}} \leq |X| \leq (1-2^{-n}) \times 2^{(2^m-1)}$$

例如，$n=7$，$m=7$

浮点数 X 的范围

s 为 0或1

阶码 E
定点纯整数

$$X = (-1)^{s} \times M \times R^{E}$$

尾数 M
定点纯小数

绝对值最小的非零数 ？ $|x|_{min}$

$$0.0\ldots01 \times 2^{-111\ldots1} = 2^{-n} \times 2^{-(2^m-1)} = \frac{2^{-n}}{2^{(2^m-1)}}$$

绝对值最大的数 ？ $|x|_{max}$

$$0.1\ldots11 \times 2^{111\ldots1} = (1-2^{-n}) \times 2^{(2^m-1)}$$

浮点数X的表示范围

$$\frac{2^{-n}}{2^{(2^m-1)}} \leq |X| \leq (1-2^{-n}) \times 2^{(2^m-1)}$$

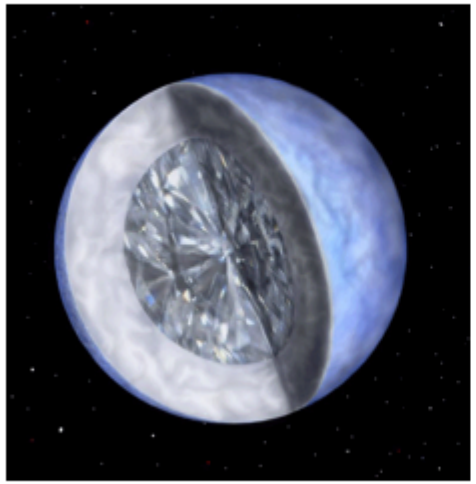
例如，$n=7,\ m=7$

$$\frac{2^{-7}}{2^{(2^7-1)}} \leq |X| \leq (1-2^{-7}) \times 2^{(2^7-1)}$$

$$\frac{2^{-7}}{2^{(127)}} \leq |X| \leq (1-2^{-7}) \times 2^{127}$$

$$\frac{0.0078125}{1.70\times10^{38}} \leq |X| \leq 0.9921875 \times 1.70 \times 10^{38}$$

$$10^{38} = 10^4 \times 10^8 \times 10^8 \times 10^8 \times 10^8$$



宇宙的钻石星球，代码为BPM37093，是一颗白矮星。位于靠近南十字座的半人马座，距离地球约50光年。

这颗星球的外层覆盖着氢氦气体，宝石藏在气体下面。它原本是一颗太阳大小的恒星，这颗恒星燃烧尽了所有的核燃料，最终由于恒星内部强大的压力使碳结晶，形成钻石。有着一个直径3000千米的钻石内核，是迄今人类已知的银河系中最大的钻石，**10的34次方克拉**（即重量达100**亿亿亿亿克拉,2000亿亿亿吨**）。

1 克拉=0.2克或200毫克=200×10^{-9}吨=2e-7 吨

$$10^{34} \times 2 \times 10^{-7} = 2 \times 10^{27} = 2 \times 10^3 \times 10^8 \times 10^8 \times 10^8$$