第一章　计算机基础

西南交通大学 Southwest Jiaotong University

❖ 本章内容提要:

1. 微机的发展历史,微机系统组成,三总线概念;
2. 微处理器构造及其指令执行过程;
3. 进位计数制(2、8、10、16进制)及其相互转换,二进制数的运算规则;
4. 编码(BCD码、ASCII码、汉字码,图形信息编码);
5. 无符号数和带符号数的表示方法;机器数和真值;带符号数的编码及运算;定点数和浮点数。

❖本章难点:

➢机器数与真值、补码与反码、溢出的判断方法。

第一节　计算机概述

1. 计算机发展历史简介;
2. 微型计算机结构简介;
3. 存储器的基本组成。

电子计算机的定义

西南交通大學 Southwest Jiaotong University

- 电子计算机是一种能够自动而又精确地处理信息的现代化电子设备。

电子计算机的起源

西南交通大學 Southwest Jiaotong University

- 为了加快计算速度与精度，人类一直在尝试制造计算机——从机械计算机到电子计算机；
- 第一台具有现代意义的电子计算机——ENIAC (Electronic Numerical Integrator And Computer)

电子计算机理论的创立

西南交通大学 Southwest Jiaotong University

- 冯·诺依曼(Von Neumann)确立了现代计算机的基本结构理论。
  - 由5个基本部件组成：输入器、输出器、运算器、存储器和控制器；
  - 采用了二进制进行运算；
  - 引入存储器存储程序和数据，计算机自动、高速地从存储器取出指令并执行指令。

这些基本原则至今仍然被现代计算机所采用，因此现代计算机一般被称为冯·诺依曼结构计算机。

计算机的发展史

西南交通大學 Southwest Jiaotong University

- 第一代: 电子管计算机时代(1946—1957年);
- 第二代: 晶体管计算机时代(1958—1964年);
- 第三代: 集成电路计算机时代(1964—1972年);
- 第四代:大规模集成电路计算机时代(1972年～至今);
- 第五代计算机。

特大规模集成电路ULSI(UltraLarge Scale Integration)
超大规模集成电路SLSI(Super large Scale Integration)
巨大规模集成电路GLSI(Great Large Scale Integration)

微型机的发展史

- 计算机发展到第四代时出现了微处理器，它的发展代表了微型机的发展历史，可大致划分为5代。
  - 第一代微处理器(1971—1973年)；
  - 第二代微处理器(1974—1978年)；
  - 第三代微处理器(1978—1981年)；
  - 第四代微处理器(1981—1992年)；
  - 第五代微处理器(1992年～至今)。

微型机的发展史

西南交通大學 Southwest Jiaotong University

- **第一代微处理器(1971—1973年)**

  微处理器发展的初级阶段,代表产品有Intel公司的Intel 4004,集成度为1200～2000只晶体管/片,基本指令的执行时间为10～20 ms,引脚数为16/24。

微型机的发展史

- **第二代微处理器(1974—1978年)**

  主要标志为8位中档和高档微处理器:

  Intel公司的8080;

  Motorola公司的M6800;

  Intel公司的8085;

  Zilog公司的Z80;

  Motorola公司的M6809。

微型机的发展史

- **第三代微处理器(1978—1981年)**

  主要标志是推出了16位微处理器:

  Intel的8086;

  Zilog的Z8000;

  Motorola的M68000;

  指令执行时间为0.5 μs,使微型计算机功能达到了小型计算机水平。

微型机的发展史

西南交通大學 Southwest Jiaotong University

- **第四代微处理器(1981—1992年)**

  主要标志为推出了32位微处理器:

  Intel于1991年初推出80386;

  HP公司推出HP9000;

  Motorola公司推出68020;

  Zilog公司推出Z80000。

  有准32位微处理器与真32位微处理器之分。

微型机的发展史

西南交通大學 Southwest Jiaotong University

- **第五代微处理器(1992年以后)**

  1993年,Intel公司推出了Pentium(奔腾),它拥有超标量结构,16KB一级缓存。 AMD和Cyrix推出了与其兼容的处理器K5和6x86;

  1996年,Intel公司推出了Pentium Pro(高能奔腾),增加256KB或512KB二级缓存;

  1996年,Intel公司推出了Pentium Pro(高能奔腾),一级缓存提高到32KB;

微型机的发展史

1997年推出PⅡ,它是对Pentium Pro的改进;

1998年推出赛扬(Celeron),它是简化版的PⅡ,价格较低;

1999年又推出了开发代号为Coppermine的PⅢ;

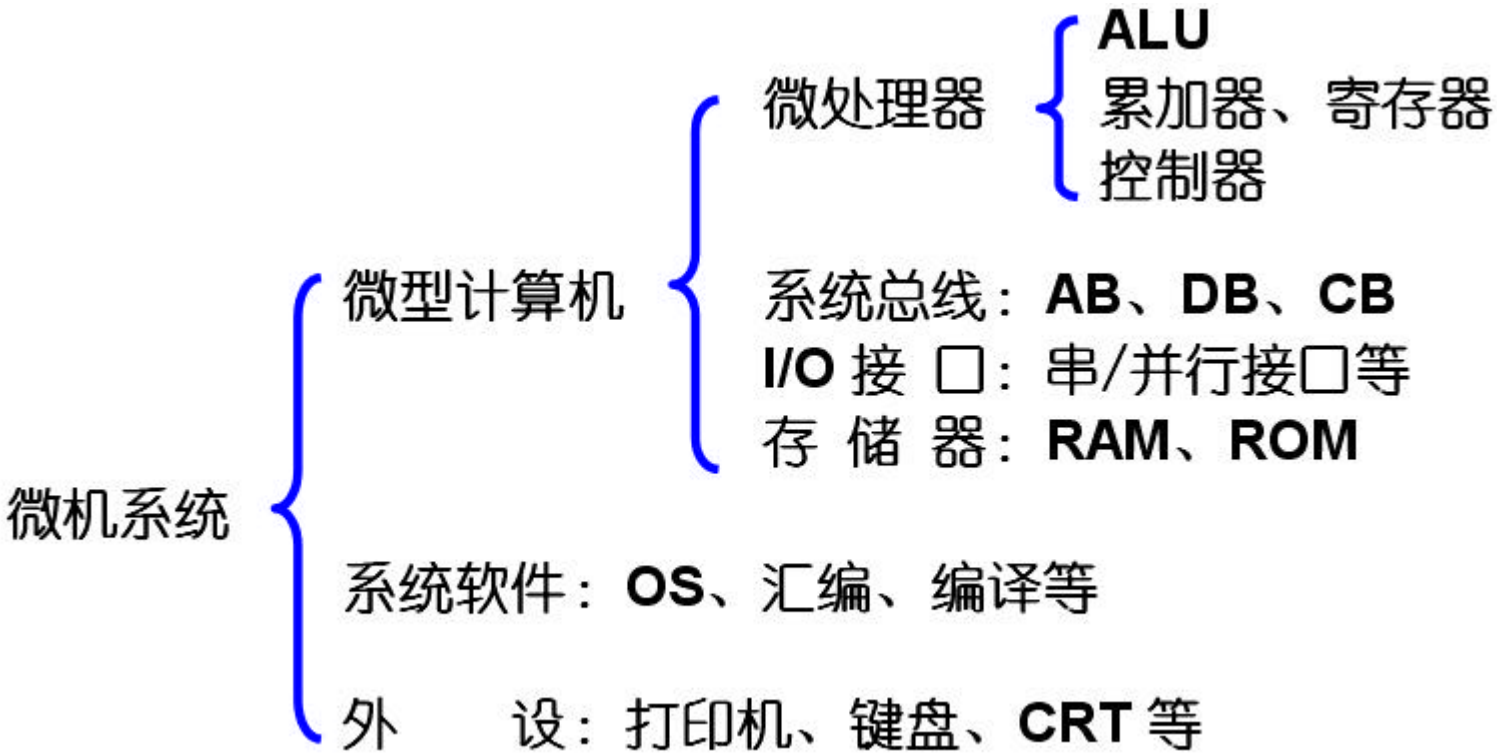
以后又陆续推出了赛扬Ⅱ、P4等性能更加先进的CPU(如使用超线程技术的P4、用于笔记本计算机的讯驰CPU);

64位微处理器。

微型机系统的组成

西南交通大学 Southwest Jiaotong University

❖ 完整的微机系统包括**硬件**与**软件**两部分。



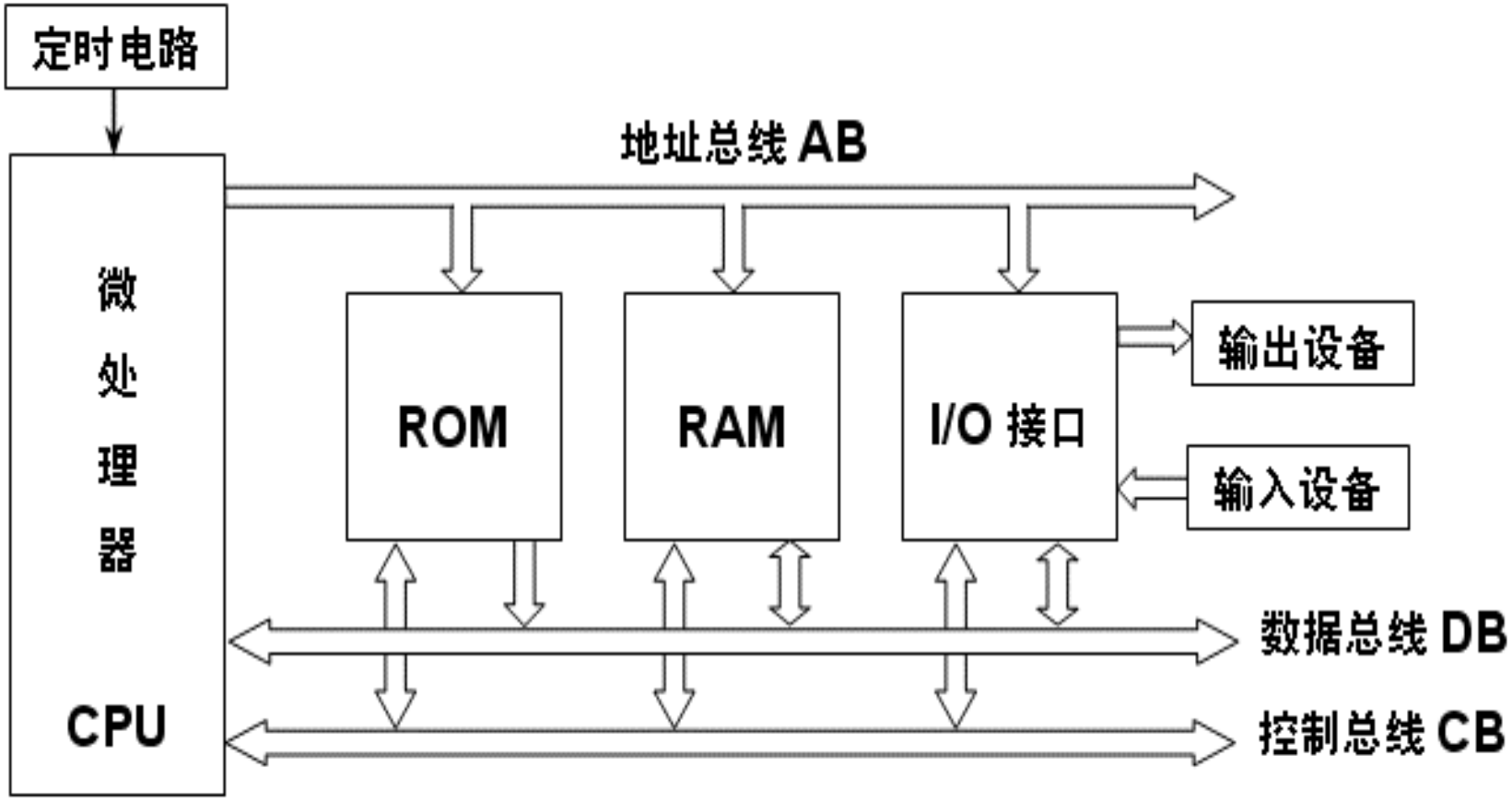
微机系统结构图

微机系统软件

西南交通大学 Southwest Jiaotong University

- 微型计算机的软件分为*系统软件*和*用户软件*;
- 系统软件是指不需要用户干预的能生成、准备和执行其它程序所需的一组程序;
- 用户软件是各用户为解题或实现检测与实时控制等不同任务所编制的应用程序。
- 程序设计层次
  - 机器语言程序设计;
  - 汇编语言程序设计;
  - 高级语言程序设计。

微机系统硬件结构图

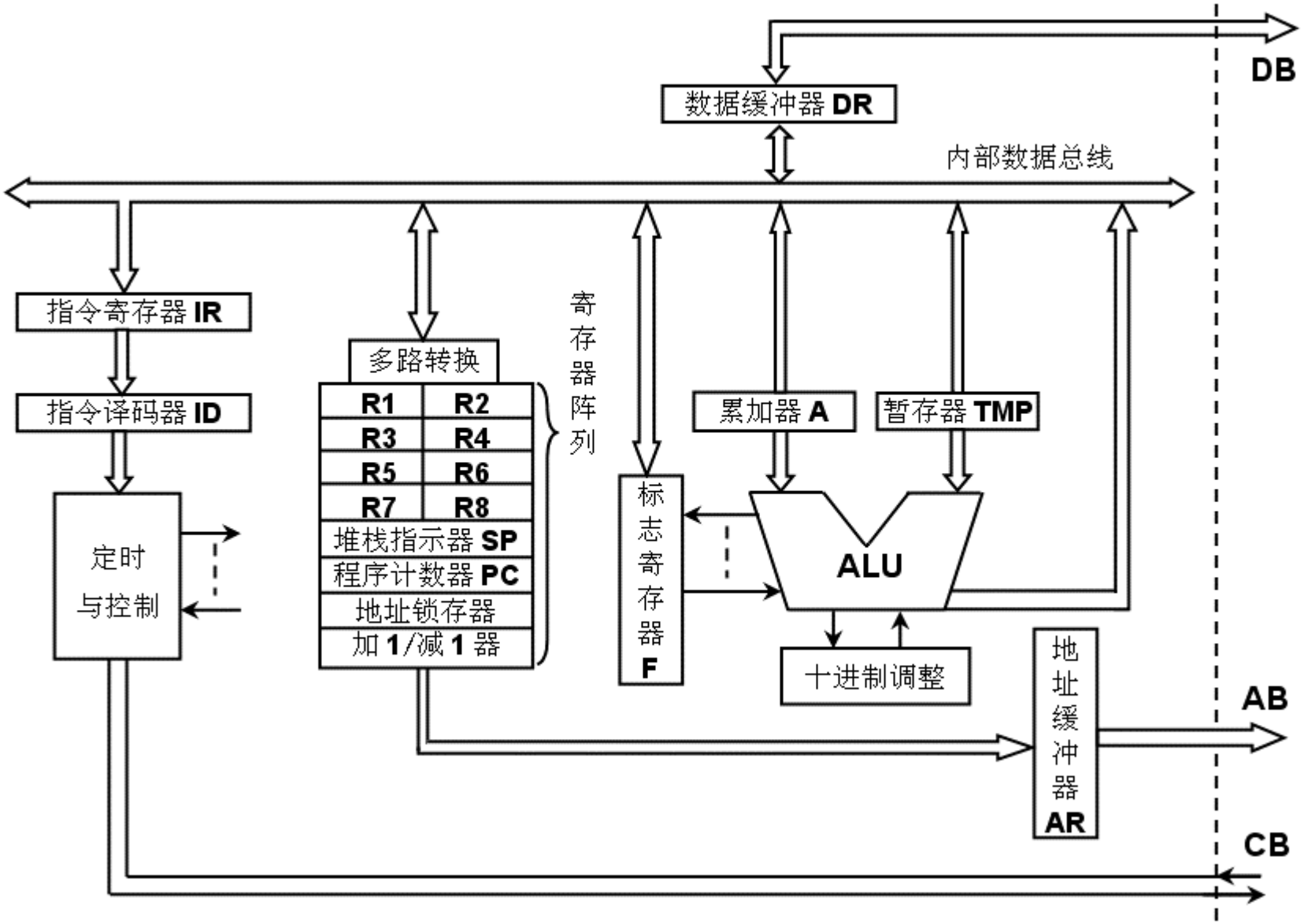


总线的概念

西南交通大学 Southwest Jiaotong University

- 总线定义：*所谓总线，是连接多个功能部件或多个装置的一组公共信号线。*
- 按所传送信息的类型，总线可分为：
  - 地址总线：传送计算机中的地址信息的信号线；
  - 数据总线：传送计算机中数据信息的信号线（双向、三态）；
  - 控制总线：传送计算机中的控制信号的一组总线。
- 按在系统中的不同位置，总线还可分为：
  - 内部总线；
  - 外部总线

CPU基本结构


DB
数据缓冲器 DR
内部数据总线
指令寄存器 IR
指令译码器 ID
定时
与控制
多路转换
R1
R2
R3
R4
R5
R6
R7
R8
堆栈指示器 SP
程序计数器 PC
地址锁存器
加 1/减 1 器
寄存器阵列
标志寄存器 F
累加器 A
暂存器 TMP
ALU
十进制调整
地址缓冲器 AR
AB
CB

CPU基本结构

- 通用CPU由运算器和控制器组成;
- 内部采用总线、累加器结构,广泛采用三态电路等;
- 引脚采用总线**分时复用**技术可节约成本,但系统的复杂性增加且降低了系统性能,现代高性能CPU一般不采用;

CPU基本结构

- 寄存器阵列RA
  - 通用寄存器R_1～R_8
  - 程序计数器PC(Program Counter)
  - 堆栈指针SP(Stack Pointer)

CPU基本结构

西南交通大学 Southwest Jiaotong University

- **运算器**
  - **累加器A(Accumulator)　它有两个功能: 运算前寄存第一操作数,是ALU的一个操作数的输入端; 运算后存放ALU的运算结果。它既是操作数寄存器又是结果寄存器。**
  - **暂存器 TMP(Temporary)。**
  - **算术逻辑单元ALU(Arithmetic Logic Unit)　由并行加法器和其它逻辑电路(如移位电路、控制门等)组成。完成各种算术逻辑运算及其它一些操作。**
  - **标志寄存器F(Flag)或称程序状态字(PSW)。**

CPU基本结构

- **控制器:**
  - **指令寄存器IR(**Instruction Register**)**
  - **指令译码器ID(**Instruction Decoder**)**
  - **定时与控制电路(**Timing and Control**)**

CPU基本结构

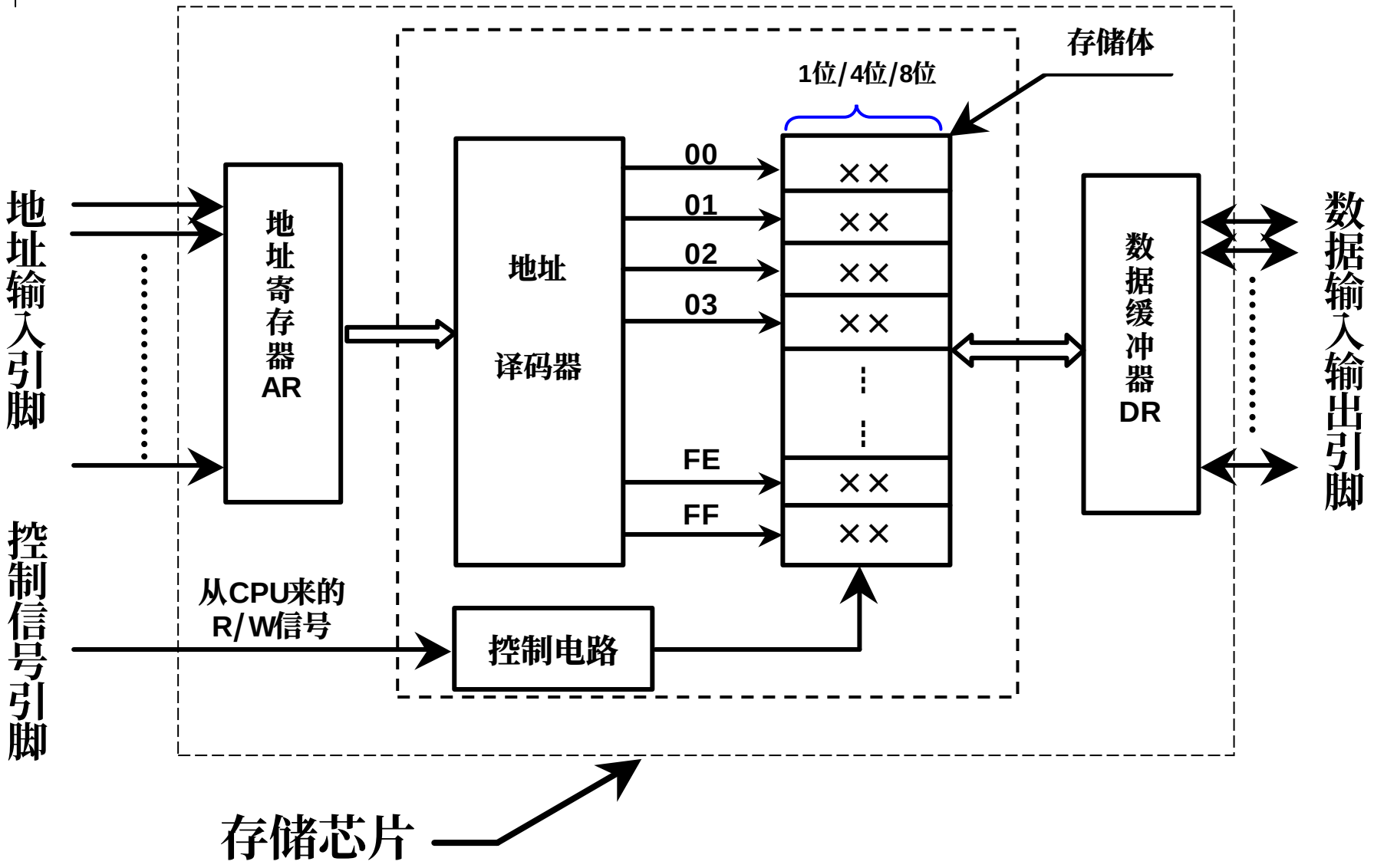
- **数据和地址缓冲器**

  简称总线缓冲器，是数据或地址信号的进出口。用来隔离微处理器内部总线和外部总线，并提供附加的驱动能力及信号整形功能。

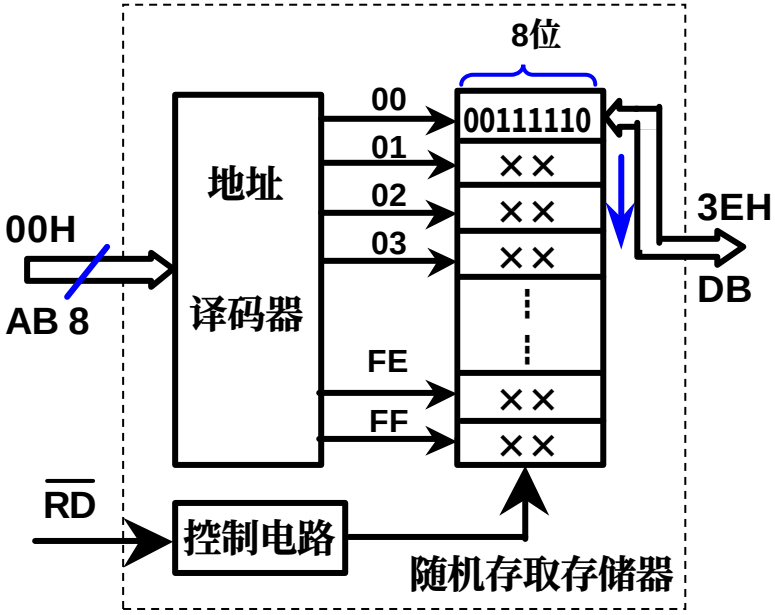
存储器概述

- 存储器是用来存放**程序**和**数据**的。在机器内部，程序和数据都是用二进制代码的形式表示。
- 存储单元地址与存储单元内容。
- 常用的存储容量单位有：
  - 位(bit)：记作b
  - 字节(Byte)：记作B
  - 千字节(2^{10}字节)：记作KB，1KB＝1024B
  - 兆字节(2^{20}字节)：记作MB，1MB＝1024KB
  - GB(2^{30}字节)：1GB＝1024MB
  - TB(2^{40}字节)：1TB＝1024GB

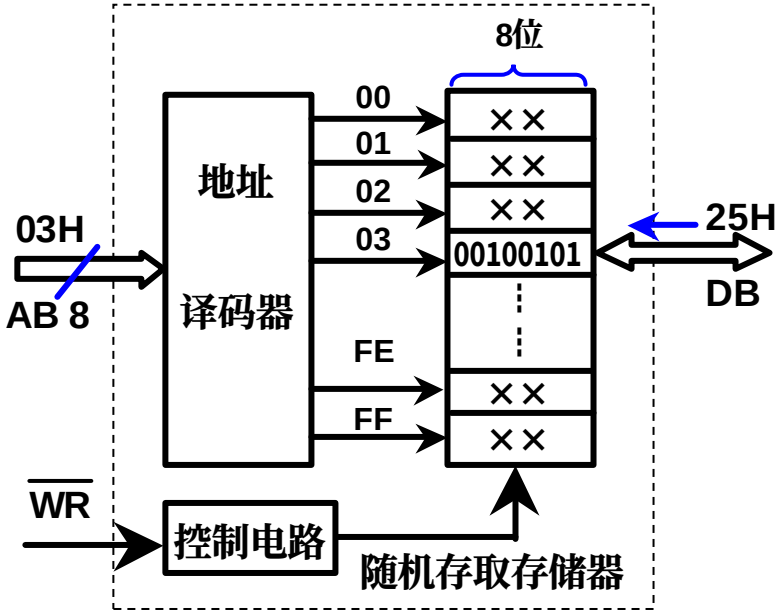
存储器结构图


存储体
1位/4位/8位
地址输入引脚
地址寄存器AR
地址
译码器
00
01
02
03
FE
FF
××
数据缓冲器DR
数据输入输出引脚
控制信号引脚
从CPU来的R/W信号
控制电路
存储芯片

存储器的操作过程



读操作



写操作

存储器的特点

- 读操作完成后，原存储单元中的内容仍保持不变，它允许多次读出同一单元的内容。
- 对存储单元执行写入操作将破坏该存储单元原存储的内容，即由新内容代替了原来存储的内容。
- 随机存取存储器RAM(Random Access Memory)。所谓“随机存取”即所有存储单元均可随时被访问，既可以读出也可以写入信息。
- 只读存储器ROM(Read only Memory)，只能读出其内容的存储器。

第二节　二进制与信息在计算机中的编码

1. 进位计数制与二进制；
2. 进位计数制之间的转换；
3. 信息在计算机中的编码。

进位计数制

❖ r进制计数的一般表达式

$$A_r = \pm[a_{n-1} \times r^{n-1} + a_{n-2} \times r^{n-2} + \cdots + a_0 \times r^0 + a_{-1} \times r^{-1} + \cdots + a_{-m} \times 10^{-m}]$$

$$= \pm \sum_{i=-m}^{N-1} a_i \times r^i$$

r— r进制计数制的基;

r^i — r进制计数制第i位的权

a_i—第i位取值, $a_i = 0, 1, \cdots r-1$

进位计数制的实质是位置计数法,不同数位的权值不同。

常用计数制及其标记方法

- 常用计数制有十、二、八与十六进制。
- 标记方法1:

  把数加上方括号,并在方括号右下角标注数制代号,如$[101]_{16}$、$[101]_{8}$、$[101]_{2}$和$[101]_{10}$
- 标记方法2:

  是在被标记数后面分别用大写英文字母B、Q、D和H标记二进制、八进制、十进制和十六进制数。十进制数中的D可以省略。为避免与字符串混淆,规定以字符打头的十六进制数前面必须加0。

  例如: 0F789H,0FACEH

十进制计数

西南交通大学 Southwest Jiaotong University

- 基数为10;
- 第i位权值为10^i;
- 第i位取值范围是0～9;
- 例：$123.45 = 1\times10^2+2\times10^1+3\times10^0+4\times10^{-1}+5\times10^{-2}$

二进制计数

西南交通大學 Southwest Jiaotong University

- 基数为2;
- 第i位权值为2^i;
- 第i位取值范围是0、1;
- 例:

10110.11B

$=1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+1\times2^{-2}$

$=1\times2^4+1\times2^2+1\times2^1+1\times2^{-1}+1\times2^{-2}$

$=22.75$

- 目前绝大部分计算机内部都使用二进制。

8进制计数

- 基数为8;
- 第i位权值为8^i;
- 第i位取值范围是0～7;
- 例：$35.71Q = 3\times8^1+5\times8^0+7\times8^{-1}+1\times8^{-2}$

$= 29$

十六进制计数

- 基数为16;
- 第i位权值为16^i;
- 第i位取值范围是0～9、A～F;
- 例： $70F.B1H = 7\times16^2 + F\times16^0 + B\times16^{-1} + 1\times16^{-2}$
  $= 1807.6914$
- 16进制数在本课程中常用。

不同计数制之间的转换

- 任意进制数转换为十进制数：按权展开相加法。例如：
  - $70F.B1H = 7\times16^{2} + F\times16^{0} + B\times16^{-1} + 1\times16^{-2}$
    $= 1807.6914$
  - $35.71Q = 3\times8^{1} + 5\times8^{0} + 7\times8^{-1} + 1\times8^{-2}$
    $= 29$
  - $10110.11B = 1\times2^{4} + 0\times2^{3} + 1\times2^{2} + 1\times2^{1} + 0\times2^{0} + 1\times2^{-1} + 1\times2^{-2}$
    $= 22.75$

不同计数制之间的转换

- 十进制数转换为任意进制数(整数与小数分别处理)
  - 整数部分采用除基取余法

设任意十进制正整数A,可表示为r进制:

$A = a_{n-1} \times r^{n-1} + a_{n-2} \times r^{n-2} + \ldots + a_1 \times r^1 + a_0$

上式两边除以r,则有:

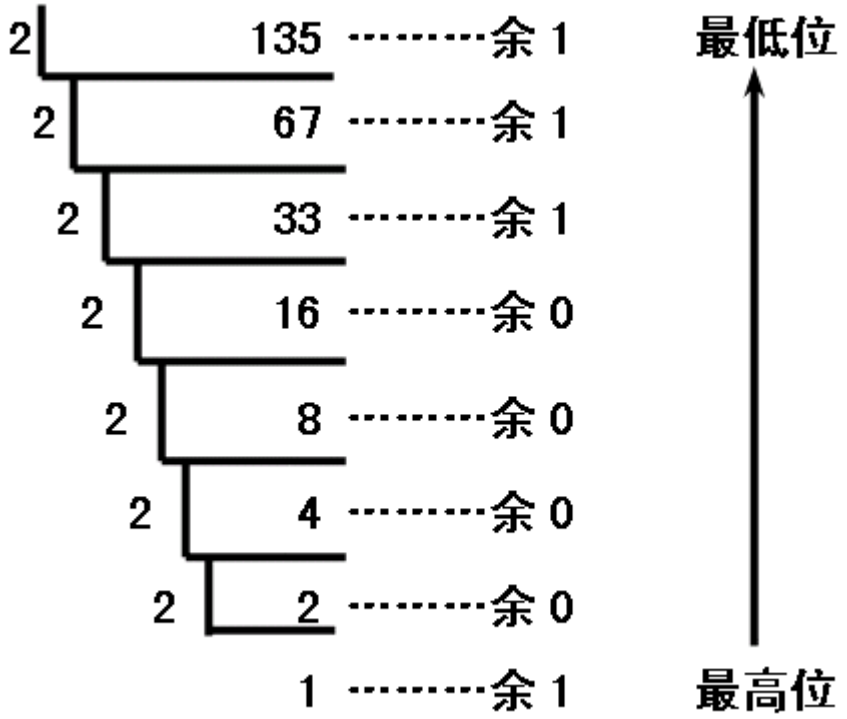
$A/r = a_{n-1} \times r^{n-2} + a_{n-2} \times r^{n-3} + \ldots + a_1 + a_0/r$

因$0 \le a_0 \le r-1$且为整数,a_0即为除法后的余数。因此,数A第一次除以基数r后的余数即为r进制数的最低位。重复上述过程直至商为0,可依次得到r进制数的系数。

不同计数制之间的转换

- 【例1】 试求出十进制数135的二进制数。

解法如下。

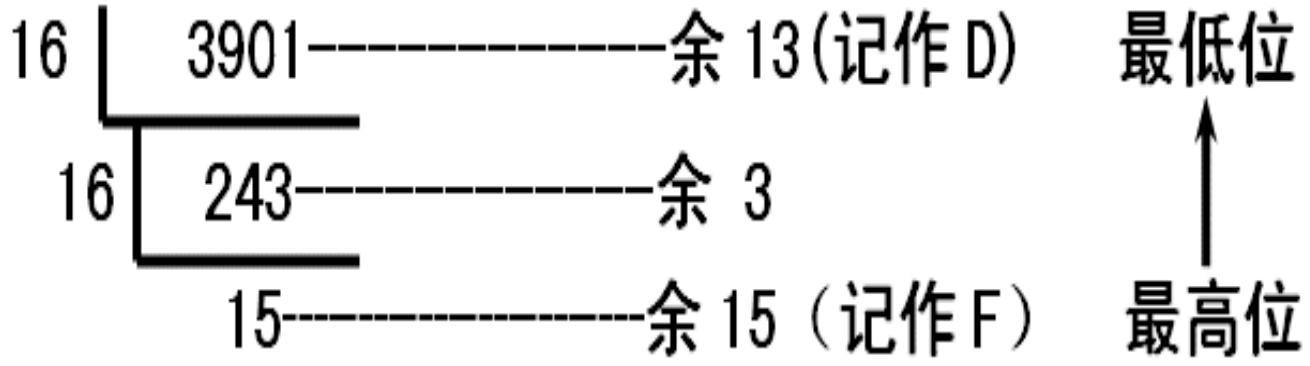


结果为: 135＝10000111B

不同计数制之间的转换

- 【例2】 求3901所对应的十六进制数。

解：相应竖式为如下



于是得到: 3901＝0F3DH

不同计数制之间的转换

- 小数部分采用**乘基取整法**

  设任意十进制小数B,可表示为r进制小数

  $B = b_{-1} \times r^{-1} + b_{-2} \times r^{-2} + \ldots + b_{-m} \times r^{-m}$

  上式两边乘以r,则有：

  $r \times B = b_{-1} + b_{-2} \times r^{-1} + \ldots + b_{-m} \times r^{-m+1}$

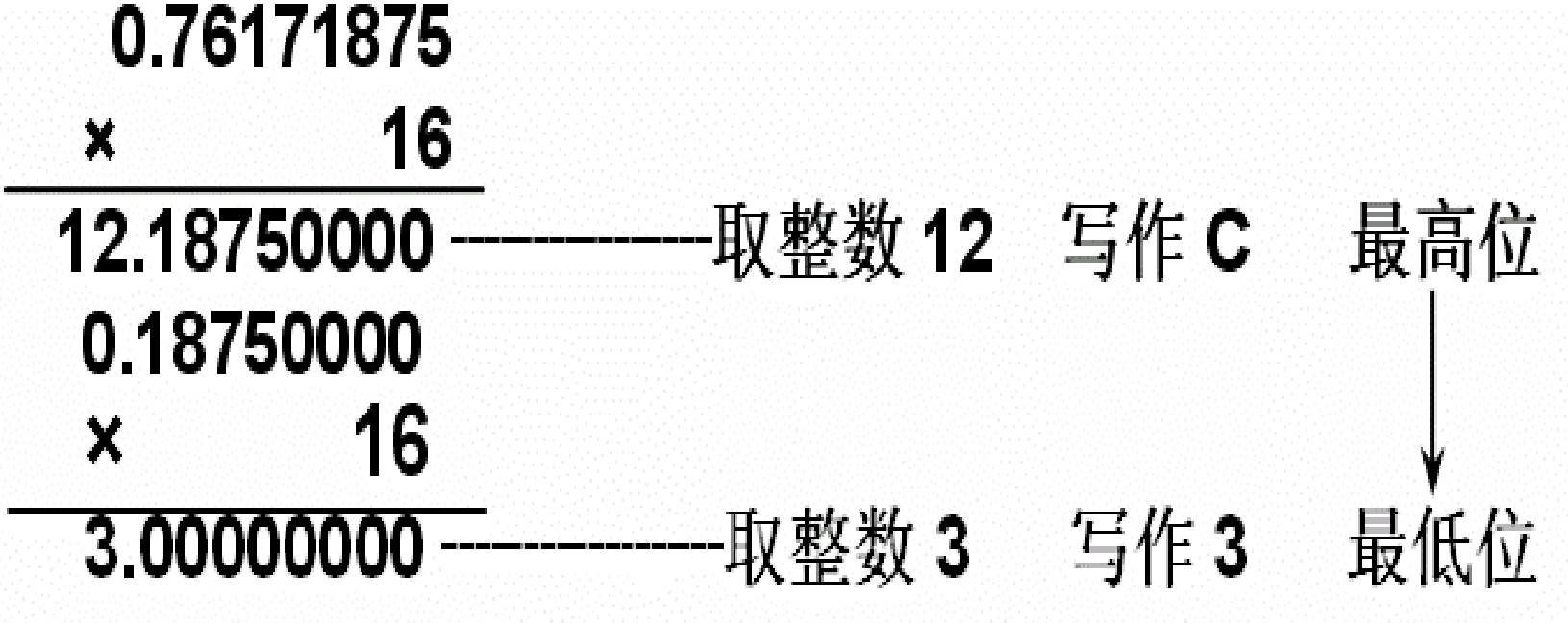
  可见,十进制小数乘以一次基数r后,乘积的整数部分即为r进制小数的最高位。重复上述运算,可依次得到r进制小数的各位系数。

- 任何十进制整数都可以精确转换成一个r进制整数,但十进制小数却不一定可以精确转换成一个r进制小数,小数转换位数视精度要求确定。

不同计数制之间的转换

- 【例3】求0.76171875的十六进制数。

  解: 相应竖式如下

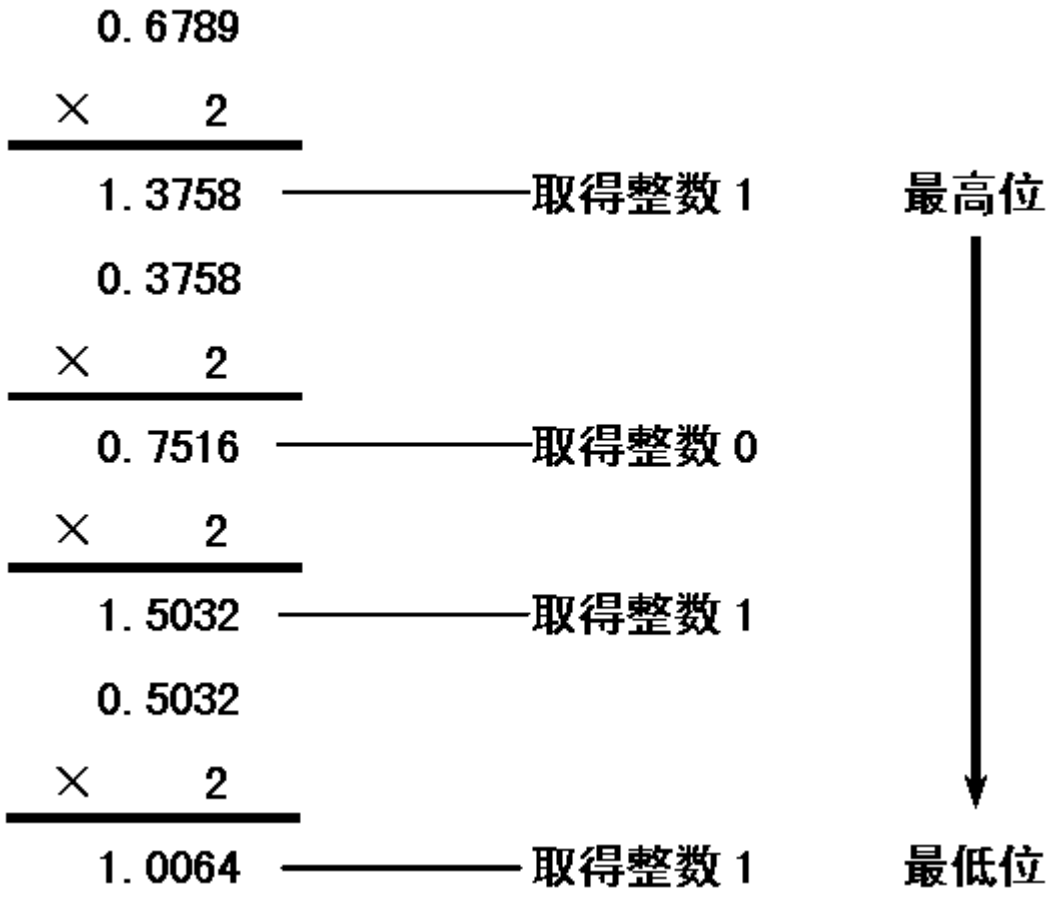


于是得到: 0.76171875＝0.C3H

不同计数制之间的转换

❖ 【例4】 试把十进制小数0.6879转换为二进制小数。

解: 相应算式如下



0.6879≈0.1011B

由此可见，0.6879并不能精确转换为二进制小数，但0.1011B＝0.6875，已有相当精度。

不同计数制之间的转换

西南交通大学 Southwest Jiaotong University

- **二进制与八进制数的转换**

  可采用“三位合一位法”。“三位合一位法”法则是：从二进制数的小数点开始，向左或向右每三位分成一组，不够三位以0补足之(整数部分不足3位，左边补0；小数部分不足3位，右边补0)，然后每组用八进制数码表示，并按序相连。

- **八进制转换成二进制数**

  这种转换方法是把八进制数的每位分别用三位二进制数码表示，然后把它们连成一体。

不同计数制之间的转换

- **二进制数转换成十六进制数**

  可采用“四位合一位法”。“四位合一位法”法则是：从二进制数的小数点开始，向左或向每四位分成一组，不够四位以0补足之（整数部分不足4位，左边补0；小数部分不足4位，右边补0），然后每组用十六进制数码表示，并按序相连。

- **十六进制转换成二进制数**

  这种转换方法是把十六进制数的每位分别用四位二进制数码表示，然后把它们连成一体。

二进制数的算术运算

- 加法运算法则：

  0 + 0＝0

  1 + 0＝0 + 1＝1

  1 + 1＝0　　(向邻近高位有进位)

  1 + 1 + 1＝1　　(向邻近高位有进位)

- 减法运算法则：

  0 - 0＝0

  1 - 1＝0

  1 - 0＝1

  0 - 1＝1　　(向邻近高位借位)

二进制数的算术运算

- 二进制乘法法则:

  0×0＝0

  1×0＝0×1＝0

  1×1＝1

- 除法是乘法的逆运算。与十进制类似,也由减法、试商等操作逐步完成。

逻辑运算

- 逻辑“与”运算法则

  $0 \wedge 0 = 0$,$1 \wedge 0 = 0 \wedge 1 = 0$,$1 \wedge 1 = 1$

- 逻辑“或” 运算规则

  $0 \vee 0 = 0$,$1 \vee 0 = 0 \vee 1 = 1$,$1 \vee 1 = 1$

- 逻辑“非”运算规则

  $\overline{0} = 1$,$\overline{1} = 0$

- 逻辑“异或”运算规则

  $0 \oplus 0 = 1 \oplus 1 = 0$,$1 \oplus 0 = 0 \oplus 1 = 1$

计算机中的编码

西南交通大學 Southwest Jiaotong University

- 目前的计算机还无法直接识别人类社会使用的信息(如文字、声音、图像等)。
- 编码是信息的另一种表示方式,经过编码的信息适合计算机处理;

计算机中的编码

- **BCD码**

  用4位二进制数字编码表示一位十进制数字即为BCD(Binary-Coded Decimal)码。

- **常用编码(8421 BCD码)**

十进制数	8421BCD码	十进制数	8421BCD码
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

计算机中的编码

- 压缩与非压缩BCD码
  - 用一个字节存放两位BCD码来表示十进制数,即为压缩BCD码;
  - 一个字节只存放一位BCD码来表示十进制数,即为非压缩BCD码。

00100011
00000001

压缩BCD码表示的123

00000011
00000010
00000001

非压缩BCD码表示的123

计算机中的编码

西南交通大学 Southwest Jiaotong University

- ASCII码(美国标准信息交换码)
  - 用7位二进制数字编码的西文字符集;
  - 编码规则表(参见ASCII码表)。
  - 例: 0®30H,9®39H,A®41H,a®61H,
    $®24H,CR®0DH,LF® 0AH,SP®20H
- ASCII码字符串
  - 字符通常都是以ASCII码形式存放于内存。字符串是指一串连续的字符,占用存储器一片连续的空间。书写时用单引号括起来,如:´COMPUTER´。

计算机中的编码

西南交通大學 Southwest Jiaotong University

- 汉字编码
  - 字母、数字和各种常用符号682个;
  - 一级常用汉字3755个,按汉语拼音顺序排列;
  - 二级常用汉字3008个,按偏旁部首排列。
- 编码方法
  - GB2312国标字符集构成一个二维平面,分成94行94列,行号称为区号,列号称为位号,分别用七位二进制数表示。每个汉字或字符在码表中都有占用一个唯一的14位编码(7位区号在左,7位位号在右),用区号和位号作为汉字的编码就是汉字的区位码。

计算机中的编码

西南交通大學 Southwest Jiaotong University

❖GB2312—80字符集的二维分布图

区码 \ 位码	1）2）3）.....94）
1） ⋮ 9）	标准符号区：字母、数字、各种常用符号
10） ⋮ 15）	自定义符号区
16） ⋮ 55）	一级汉字（3755 个，按拼音顺序排列）
56） ⋮ 87）	二级汉字（3008 个，按偏旁部首排列）
88） ⋮ 94）	自定义汉字区（共 658 个）

（55 区的第 90~94 位未定义汉字）

计算机中的编码

西南交通大學 Southwest Jiaotong University

- 汉字编码种类
  - 区位码

    用编码表中的区号(高位数字)和位号(低位数字)构成的编码就是汉字的区位码。
  - 国标码(系统之间交换汉字信息时使用的编码)

    在区位码的区号和位号上各加32(即20H),就构成了该汉字的国标码(GB2312–80)。
  - 机内码(计算机内部存储和处理汉字时使用的编码)

    在国标码的高字节与低字节上各加128(即80H),就构成了该汉字的机内码。

计算机中的编码

- 汉字的处理
  - 汉字的输入——将汉字转换为机内码

    三大类输入法: 键盘输入法、字形识别法和语音识别法。
  - 汉字的输出——将机内码还原为汉字

    根据内码到汉字库中找出该汉字的字形描述信息，再送去显示或打印输出。

图形图像的数字化

- 图形表示法是用几何要素(点、线、图、体等)以及表面材质、光照位置等来描述画面或场景中的内容。
- 图象表示法用m×n个像点(Pixel,即像素)来表示画面内容,又称为位图表示法和点阵表示法。

第三节　机器数及其运算法则

❖ 本节主要内容

1. 有符号数的机器表示(原码、反码与补码);
2. 机器数与真值的转换;
3. 补码运算规则及运算溢出;
4. 数的定点与浮点表示。

❖ 难点

补码概念、补码运算溢出的判断方法。

无符号数与有符号数

- 无符号数及其计算机表示
  - 无符号数是指正整数;
  - 计算机字长的全部数位都可用来表示无符号数的数值大小;
  - 设机器字长为n位,则其无符号数的数值范围是:0～2^n-1。
- 有符号数及其计算机表示
  - 有符号数包括正数和负数;
  - 计算机表示有符号数时,需要占用字长中的一位表示数符;
  - 数值表达范围与采用的码制有关。

机器数与真值

- 将数符编码表示的有符号数称为机器数。通常约定二进制数的最高位为符号位，“0”表示正号，“1”表示负号。
- 用“+”、“–”加绝对值来表示数值大小，这种表示形式在计算机技术中称为真值。

原码

- 用二进制数字的最高位表示数的符号(常以“0”表示正数,“1”表示负数),其余各位表示数值绝对值,则称为该数的原码表示法。

设$|X|=X_{n-2}X_{n-3}\cdots X_1X_0$(即X的数值有n–1位二进制数)。

- X＞0时,$[X]_{原}=0X_{n-2}X_{n-3}\cdots X_1X_0$(正数的原码表示);
- X＜0时,$[X]_{原}=1X_{n-2}X_{n-3}\cdots X_1X_0$(负数的原码表示);
- 零的原码表示有两种:

$[+0]_{原}=000\cdots00$，$[-0]_{原}=100\cdots00$

- 原码的定义式如下:(设机器字长为n位)

$$[X]_{原}=\begin{cases}X & 0\le X\le 2^{n-1}-1\\ 2^{n-1}-X & -(2^{n-1}-1)\le X\le 0\end{cases}$$

反码

- 正数的反码与其原码相同，负数的反码是将其原码除符号位以外的各位按位取反。

设$|X|=X_{n-2}X_{n-3}\cdots X_1X_0$（即X有n−1位二进制数数值）。

- $X\geq 0$时，$[X]_{反}=0X_{n-2}X_{n-3}\cdots X_1X_0$（正数的原码表示）；
- $X\leq 0$时，$[X]_{反}=1\overline{X_{n-2}}\,\overline{X_{n-3}}\cdots\overline{X_1}\,\overline{X_0}$（负数的原码表示）；
- 零的反码表示有两种：

  $[+0]_{反}=000\cdots00$，$[-0]_{反}=111\cdots11$

- 反码的定义式如下：（设机器字长为n位）

$$[X]_{反}=\begin{cases}X & 0\leq X\leq 2^{n-1}-1\\ 2^n-1+X & -(2^{n-1}-1)\leq X\leq 0\end{cases}$$

补码的定义

- 在计算机中引入补码主要是基于下面两个原因:
  - 使符号位能和有效数值部分一起参加数值运算,从而简化运算规则,节省运行时间。
  - 使减法运算转化为加法运算,从而进一步简化计算机中运算器的线路设计。
- 模与同余
  - 计算机中字长总是一定的,能直接表示的最大数值有限,当运算结果超出其最大值时,就发生溢出,此时所产生的溢出量即为模(Module)。
  - 自动舍弃溢出量的运算称为模运算。

补码的定义

- 模运算实例: 钟表计时。
- ✓ 时钟走到12点时,计时又从零开始,12即为溢出量,也就是说其模为12;
- ✓ 钟表对时与模运算: 设标准时间是5点,但时钟却指在8点,为了校准至5点,可用倒拨3小时或顺拨9小时,这两种拨法可记为:

倒拨 $8-3=5$

顺拨 $8+9=12+5=5 \pmod{12}$

由此可见,在模为12的数字系统中,

$8-3=8+9 \pmod{12}$

$-3=9 \pmod{12}$

补码的定义

推而广之，

$[X-Y]=(X-Y)+12 \pmod{12}= X+(12-Y) \pmod{12}$（其中 $X>0, Y>0$）

若设 $[-Y]_{补}=12-Y$，则有：

$$X-Y=X+[-Y]_{补} \pmod{12}$$

从而，减法运算也可转化为补码加法运算。

需要注意的是，模为12的数字系统中，求$[-Y]_{补}$仍需做减法。但在二进制数字系统中，补码求法简便，容易由数字电路实现。

补码的定义

- 补码定义
  - 设n位二进制数的最高位为符号位，数值部分为n–1位，其补码定义为：

$$[X]_{补}=\begin{cases} X & 0\le X\le 2^{n-1}-1 \\ 2^{n}+X & -(2^{n-1}-1)\le X<0 \end{cases}$$

此时n位二进制数的模数为2^n

补码的求取

西南交通大學 Southwest Jiaotong University

- 补码的求法
  - 正数的补码与其原码相同;
  - 负数的补码等于其原码中除符号位保持不变外,其余各位按位求反,再在最低位加1。按照定义,

$$[X]_{补}=2^n+X=2^n-|X| \quad (X<0)$$

$$=2^{n-1}+(2^{n-1}-1-|X|)+1$$

$$=2^{n-1}+[(1-X_{n-2})\times 2^{n-2}+(1-X_{n-3})\times 2^{n-3}+\ldots+(1-x_1)\times 2^1+(1-x_0)]+1$$

$$=2^{n-1}+\overline{X_{n-2}}\,\overline{X_{n-3}}\ldots\overline{x_1}\,\overline{x_0}+1$$

符号位为1　数值位部分求反加1

补码的求取

- 0的补码只有一个:

  $[+0]_{补}=[-0]_{补}=000\cdots0$

- 补码与反码的关系

  $[X]_{补}=[X]_{反}+1$

- 简便的直接求补法1

(1)从最低位起,到出现第一个1以前(包括第一个1),原码中的数字不变,以后逐位取反,但符号位不变。

补码的求取

❖ [例] X_1＝-1011111B，X_2＝-1111000B，n＝8，求$[X_1]_{补}$及$[X_2]_{补}$。

解：① X_1＝-1011111B，$[X_1]_{原}$＝1 1 0 1 1 1 1 1B

符号位不变

按位取反

第一个1不变

由原码求补码：

因此，$[X_1]_{补}$＝10100001B

补码的求取

解：②X2＝-1111000B，[X2]原＝1 1 1 1 1 0 0 0B

符号位不变

按位取反

第一个1不变

由原码求补码:

因此，$[X_2]_{补}$＝10001000B

补码的求取

- **简便的直接求补法2**

  首先求出负数绝对值的n位(n为所需字长)二进制表示形式,再连同符号位按位取反加1。

[例]设X = –117，求 $[X]_{补}$ 。（mod 2^8）

解：|X|＝117＝01110101B

$[X]_{补}$＝10001010B+1＝10001011B

- **计算机中的数据,没有特别申明的带符号数一律用补码表示,把符号也看成数的一部分共同参与运算,运算结果自然也是补码。数据在输入过程中即由输入程序转化为补码。**

由机器数求真值

- 由原码求真值：将符号位变为“+”或“–”号，数值位不变。
- 由反码求真值：若符号位为0，其真值为“+”号加上数值位；若符号位为1，其真值为“–”号加上各数值位的按位求反。
- 由补码求真值：若符号位为0，其真值为“+”号加上数值位；若符号位为1，其真值为“–”号加上各数值位的按位求反再加1。

补码加、减运算规则

- 参加运算的两个操作数均用补码表示，运算结果仍为补码；
- 符号位作为数字的一部分直接参与运算；
- 若做加法，则两数直接相加，若做减法，则先将减数进行变补操作，再与被减数相加；

补码的运算规则

- $[X + Y]_{补} = [X]_{补} + [Y]_{补}$
  - 上式表明两数*和的补码*等于两数的*补码之和*。
- $[X-Y]_{补} = [X + (-Y)]_{补} = [X]_{补} + [-Y]_{补}$
  - 上式表明两数之差的补码，等于被减数的补码与减数的负数的补码之和，也就是说减法可转换为补码加法完成。

补码的运算规则

西南交通大学 Southwest Jiaotong University

- 如何由$[Y]_{补}$求$[-Y]_{补}$？

  计算机中有符号数总是用补码表示，因此在做减法前已知的是减数的补码（设为$[Y]_{补}$），而减法转换为加法需要求出$[-Y]_{补}$。

方法1: 先求出$[Y]_{补}$的原码，再求$[-Y]_{补}$

[例] $[Y_1]_{补}=11110001B$，$[Y_2]_{补}=00110001B$，求$[-Y_1]_{补}$及$[-Y_2]_{补}$。

解：$[Y_1]_{原}=10001111B$，$[-Y_1]_{原}=00001111B$

$[-Y_1]_{补}=00001111B$。

$[Y_2]_{原}=00110001B$，$[-Y_2]_{原}=10110001B$

$[-Y_2]_{补}=11001111B$

补码的运算规则

方法2：变补法，即将$[Y]_{补}$的每一位(包括符号位)都求反加1。记为：$[[Y]_{补}]_{变补}=[-Y]_{补}$

[例] $[Y]_{补}=11110001B$，则 $[-Y]_{补}=00001111B$。

实际上，计算机内部都是采用变补法计算，易于用电路实现。注意变补是一种运算，本质上是求该补码相应真值的负数的补码，与求补码的概念不同，多数微处理器都有求变补指令。

补码的运算规则

西南交通大学 Southwest Jiaotong University

- 补码运算举例
- 例1: 设X＝96,Y＝19,利用补码运算规则求X–Y (设字长n＝8)。

解: $[X]_{补}=[X]_{原}=01100000B$,

$[Y]_{补}=[Y]_{原}=00010011B$, $[-Y]_{补}=11101101B$

$[X-Y]_{补}=[X]_{补}+[-Y]_{补}$

$$
\begin{array}{ll}
 & 01100000B \quad [X]_{补} \\
+) & 11101101B \quad [-Y]_{补} \\
\hline
 & 1\,|\,01001101B=[X-Y]_{补}=[X-Y]_{原}
\end{array}
$$

丢失

计算结果: X–Y＝+77

补码的运算规则

例2: 设X=–18, Y=+15,利用补码运算规则求X+Y

解: |X|=00010010B, $[X]_{补}$=11101110B ,

$[Y]_{补}=[Y]_{原}$=00001111B

$[X+Y]_{补}=[X]_{补}+[Y]_{补}$

$$
\begin{array}{rll}
 & 11101110B & [X]_{补} \\
+) & 00001111B & [Y]_{补} \\
\hline
 & 11111101B & [X+Y]_{补}=[-3]_{补}
\end{array}
$$

计算结果: X+Y=–3

关于“0”的问题

表1–4列出了三种表示法表示的三位二进制数的真值，最高位是符号位，0为“+”，1为“–”，n＝3。

表1–4 带符号位二进制数的三种表示法的真值

二进制数	原码表示法	反码表示法	补码表示法
000	+ 0	+ 0	+ 0
001	+ 1	+ 1	+ 1
010	+ 2	+ 2	+ 2
011	+ 3	+ 3	+ 3
100	– 0	– 3	– 4
101	– 1	– 2	– 3
110	– 2	– 1	– 2
111	– 3	– 0	– 1

从表中看出，原码和反码两种表示法中，都出现了+0和–0。而补码表示法中，100表示–4而非–0。从下面的例子中可以看出这种表示法的正确性。

– 3	101	– 4	100	
+) – 1	+) 111	+) + 2	+) 010	
– 4	1 ¦ 100	– 2	110	为–2的补码

如果把100误认为–0，就会出现错误。例如：

– 0	100		
+) + 1	+) 001		
+ 1	101	为–3的补码	

溢出

- 有限字长补码的数据表达范围是有限的，一个n位二进制补码的表示范围为-2^{n-1}～$+(2^{n-1}-1)$。
- 所谓溢出是指当两个带符号数进行补码运算时，若运算结果超过运算装置的容量，数值部分便会发生溢出，占据符号位的位置，从而引起计算出错。溢出发生时，运算结果是错误的。
- 溢出的实质是运算结果超出有限字长计算机的数据表达范围。

溢出

- 运算溢出举例
  - [例1] 已知X=–1111111B,Y=–0000010B,用补码运算求X+Y (设字长n＝8)。

解: 运算式如下:

$$
\begin{array}{rll}
[X]_{补} = & 10000001B & (-127\text{的补码}) \\
+)\ [Y]_{补} = & 11111110B & (-2\text{的补码}) \\
\hline
[X]_{补}+[Y]_{补} = 1 & 01111111B & (+127\text{的补码})
\end{array}
$$

自动丢失　符号位

可见,两负数相加结果变成了正数,结果显然错误,此时运算产生了溢出。

溢出

- [例2] 已知X=–0000010B,Y=–0000010B,用补码运算求X+Y (设字长n＝8)。

解: 运算式如下:

$$
\begin{array}{lll}
[X]_{补}=11111110B & & \text{(-2的补码)} \\
+)\quad [X]_{补}=11111110B & & \text{(-2的补码)} \\
\hline
[X]_{补}+[Y]_{补} = 1 & 11111100B & \text{(-4的补码)}
\end{array}
$$

模运算自动丢失　符号

负数相加,结果仍然为负,运算结果正确,没有溢出产生。

溢出

- [例3] 已知X=+1111111B,Y=+0000010B,用补码运算求X+Y (设字长n＝8)。

解: 运算式如下:

$$
\begin{array}{rll}
[X]_{补} = 01111111B & & (+127的补码) \\
+)\ [Y]_{补} = 00000010B & & (+2的补码) \\
\hline
[X]_{补}+[Y]_{补} = & 10000001B & (-127的补码) \\
 & \uparrow & \\
 & 符号位 &
\end{array}
$$

两正数相加结果变成了负数,结果显然错误,此时运算产生了溢出。

溢出判别法

❖双高位判别法:

引进两个附加符号:

- C_S: 它表征最高位(符号位)的进位情况,如有进位,$C_S=1$,否则,$C_S=0$;
- C_P: 它表征数值部分最高位的进位情况,如有进位,$C_P=1$,否则,$C_P=0$。

双高位判别法为: 溢出$OV = C_P \oplus C_S$

❖双高位判别法可说明如下(假定参与运算的为n位二进制有符号数)。

溢出判别法

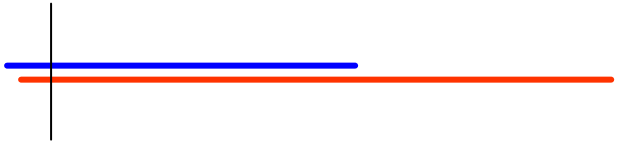
- 两正数相加，若和≥2^{n-1}，则C_P＝1，而C_S＝0，此时称为“正溢出”；若和＜2^{n-1}时，C_S＝0，C_P＝0，此时无溢出发生。
- 两负数相加，若和的绝对值＞2^{n-1}，则数值部分补码之和必小于2^{n-1}，C_P＝0，而C_S＝1。此时称为“负溢出”；若和的绝对值≤2^{n-1}时，C_S＝1，C_P＝1，无溢出发生。
- 一个正数和一个负数相加，和肯定不溢出。此时，若和为正数，则C_S＝1，C_P＝1；若和为负数，则C_S＝0，C_P＝0。

无符号数的运算

- 所谓不带符号数，是指参加运算的操作数X和Y都为正数，且整个字长的各位都用来表示数值大小。
- 8086 CPU中，两个不带符号数进行加减法，仍然采用补码运算。如果加法运算之和不超过整个字长的表示范围(0～2^n –1，n=8或16)，则不溢出，否则产生溢出(进位)，但并不表示错，只是表明给定字长无法表示，向更高位有进位而已。结果的最高位存放在进位位中。

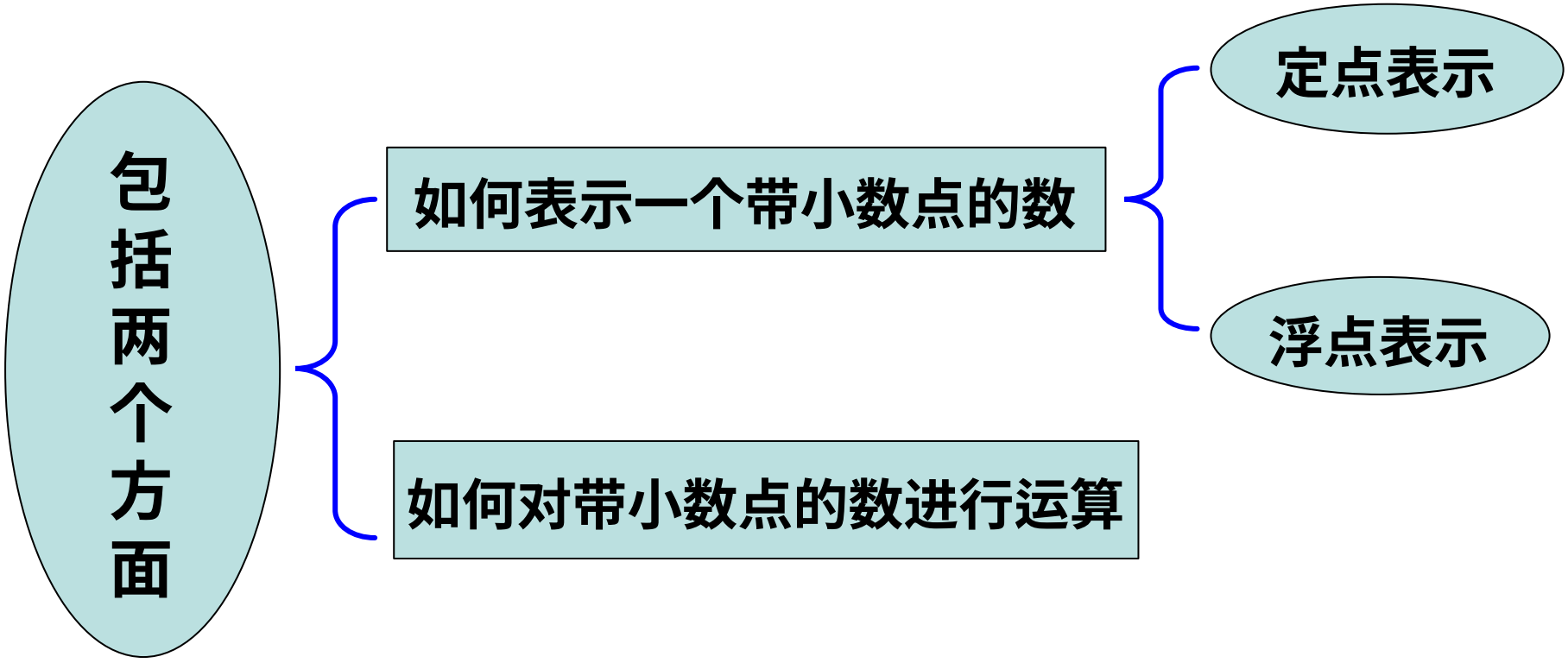
无符号数的运算

- 减法运算时，仍然采用补码加法进行计算。两个n位不带符号数进行减法运算，最高位有进位(此时称为借位)，表明被减数小于减数。当差的绝对值小于2^{n-1}时差值为负数(用补码表示)，否则结果溢出，得到的n位差值不正确(此时，如果将进位位看作该数的符号位，结果仍然正确)。因此，*不带符号数的减法运算通常只用于判断两数的大小关系。*



数的定点与浮点表示

对小数点的处理



定点法

所谓定点表示，就是小数点在数中的位置是固定的

采用定点法表示数时，小数点在数中的位置事先规定好，机器运行过程中一直保持不变。当小数点在数值位的最前面时，则为纯小数；当小数点在数值位的最后时，则为整数；还可以规定小数点在数中的任何位置（实际运用时一般不采用）。当机器采用定点运算，由于小数点位置固定而统一，对位问题已不存在。

定点计算的线路简单，但应用却不方便

浮点法

浮点表示，就是小数点在数中的位置是浮动的

如同十进制，任何一个二进制数可以表示成如下形式：

$$N = \pm S \times 2^{\pm J}$$

其中J称为阶码，S称为尾数。在计算机中若把一个二进制数也分成阶码和尾数两部分来表示，叫做浮点表示法。

数的浮点表示法源自十进制中的科学计数法

浮点数的机器表示法

阶符	阶码J	尾数符	尾数S

在大多数计算机中都把尾数规定为纯小数，即小数点在尾数的最前面。对于字长较短的微机而言，可用多个连续字节构成浮点数。

例子

浮点运算

浮点数比定点数运算复杂,运算规则也不尽相同。

- 同阶运算: 阶码不变,尾数直接相加减。
- 不同阶运算: 阶码不同,则两数就不能直接相加、减,必须对阶(即对齐小数点)后,才能做尾数间的加、减运算。
- 对阶原则: 对阶时,低阶向高阶看齐,即把阶小的小数点左移。
- 规格化(使0.5≤S<1)与计算精度(保留最多的有效数字)。