

计算机组成原理

2017年修订

西南交通大学信息科学与技术学院

唐慧佳 hjtang@home.swjtu.edu.cn



第2章 数据的机器层表示

§ 2.0 数制及进制转换

§ 2.1 数的定点表示

§ 2.2 数的浮点表示

§ 2.3 非数值数据的表示

§ 2.4 十进制数和数串的表示

§ 2.5 现代微机系统数据表示举例

§ 2.6 数据校验码



第2章 数据的机器层表示

数据：

- 1) 数值型 （如：无符号数和带符号数）
- 2) 非数值型 （如：符号、文字、校验码等）

数据表示： 数据在计算机中使用的二进制编码表示形式。
(机器数)

为什么 ？

- 1) 硬件基础：电容的冲放电、开关的闭合与开启
- 2) 运算基础：基数少，表示数的符号少，运算规则简单
- 3) 二进制的“1”、“0”与逻辑推理中的真假相对应，为实现逻辑运算和逻辑推理提供了便利。



§ 2.0 数制及进制转换

2.0.1 数制

1. 二进制

只有0和1两个数字符号，“逢二进一，借一当二”。

$$\begin{array}{ccccccc} 1 & 1 & 0 & 1 & . & 0 & 1 \\ 2^3 & 2^2 & 2^1 & 2^0 & & 2^{-1} & 2^{-2} \end{array} = 1 \times 2^3 + 1 \times 2^2 + 0 + 1 \times 2^0 + 0 + 1 \times 2^{-2}$$

2. 八进制与十六进制（用于阅读和书写）

八进制：数码有八个：0, 1, 2, 3, 4, 5, 6, 7 .

“逢八进一，借一当八”

十六进制：数码有十六个：0, 1, ..., 9, A, B, C, D, E, F



§ 2.0 数制及进制转换

2.0.1 数制

3. 几个概念

权值/位权：某个固定位置上的计数单位。例：

$$\begin{array}{ccccccc} 1 & 1 & 0 & 1 & . & 0 & 1 \\ 2^3 & 2^2 & 2^1 & 2^0 & & 2^{-1} & 2^{-2} \end{array} = 1*2^3 + 1*2^2 + 0 + 1*2^0 + 0 + 1*2^{-2}$$

权值

基数：指某个进制的值R。

例如：二进制数的基数为2，十进制数的基数为10。



4. 进制数的表示方式

1) 用下标加以标注。例如： $(1010)_2$, $(1010)_{10}$

2) 用后缀字母表示不同的进制。

B — 二进制

Q — 八进制

H — 十六进制

D — 十进制

例如： 375Q, A17H, 101B

常用几种进制的对应关系

十进制	二进制	八进制	十六进制
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

? 思考: 下列数中哪些可能为十六进制数、八进制数和十进制数?

108 , 907, A35, 780, 10, 11 , 675

2.0.2 进制转换

1. 任意进制数转换为十进制数

规则：按权展开后相加。

J进制数 $\pm (a_{n-1}a_{n-2}\dots a_0.a_{-1}a_{-2}\dots a_{-m})$ 的十进制值为：

$$N = \pm (a_{n-1} \times J^{n-1} + a_{n-2} \times J^{n-2} + \dots + a_0 \times J^0 \\ + a_{-1} \times J^{-1} + a_{-2} \times J^{-2} + \dots + a_{-m} \times J^{-m})$$

2. 十进制→二进制转换

1) 整数部分转换——除2取余法

例：十进制数11转换为二进制数

$\begin{array}{r} 2 \overline{) 11} \\ \underline{2} \\ 2 \overline{) 5} \\ \underline{4} \\ 2 \overline{) 2} \\ \underline{2} \\ 1 \end{array}$	$\begin{array}{l} \text{余数 } K_0 = 1 \\ \text{余数 } K_1 = 1 \\ \text{余数 } K_2 = 0 \\ \text{余数 } K_3 = 1 \end{array}$
--	---

$$\therefore (11)_{10} = (K_3 K_2 K_1 K_0)_2 = (1011)_2$$

2. 十进制→二进制转换

2) 小数部分转换——乘 2 取整法

例：十进制数 0.3 转换为二进制数

$$\begin{array}{rcl}
 0.3 & & \\
 \times 2 & & \\
 \hline
 0.6 & \text{整数 } K_{-1}=0 & \\
 \times 2 & & \\
 \hline
 1.2 & \text{整数 } K_{-2}=1 & \\
 \times 2 & & \\
 \hline
 0.4 & \text{整数 } K_{-3}=0 & \\
 \times 2 & & \\
 \hline
 0.8 & \text{整数 } K_{-4}=0 & \\
 \end{array}$$

...

$$\therefore (0.3)_{10} = (0.K_{-1}K_{-2}K_{-3}K_{-4}\dots)_2 = (0.0100\dots)_2$$

3. 二进制→八进制、十六进制的转换

二进制: 101110011.110 → 八进制: 563.6₈

$2^6 \ 2^3 \ 2^0 \ 2^{-3}$

即 $8^2 \ 8^1 \ 8^0 \ 8^{-1}$

二进制: 101110011.1100 → 十六进制: 173.C H

$2^8 \ 2^4 \ 2^0 \ 2^{-4}$

即 $16^2 \ 16^1 \ 16^0 \ 16^{-1}$

 手工转换捷径:

十进制→八进制→二进制

例：十进制数 11 转换为二进制数

$$\begin{array}{rcl}
 8 & \overline{) 11} & \text{余数 } K_0 = 3 \\
 & \underline{1} & \text{余数 } K_1 = 1 \\
 \therefore (11)_{10} & = (13)_8 & = (1011)_2
 \end{array}$$

 手工转换捷径:

十进制 \rightarrow 八进制 \rightarrow 二进制

例：十进制数 0.3 转换为二进制数

$$\begin{array}{r} 0.3 \\ \times 8 \\ \hline 2.4 \end{array} \quad \text{整数 } K_{-1}=2$$

$$\begin{array}{r} \times 8 \\ \hline 3.2 \end{array} \quad \text{整数 } K_{-2}=3$$

...

$$\therefore (0.3)_{10} = (0.23\dots)_8 = (0.010011\dots)_2$$

课堂练习:

1. 十进制转二进制

① 48, 103

② 0.375, 0.2

③ 4.62

④ $3 \frac{5}{8}$

2. 二进制转十进制和十六进制

① 1011, -100100

② 0.1101, -0.100101

③ 11101.1101

§ 2.1 数的定点表示

定点数：操作数数据格式中小数点的位置是固定的。

计算机中的定点数只采用纯整数或者纯小数表示。

D_3	D_2	D_1	D_0
0	1	0	0

0100 整数4

•

D_0	D_1	D_2	D_3
0	1	0	0

0100 小数0.5

•

定点数包括 1) 带符号数（最高位表示符号）

2) 不带符号数

带符号数可用原码、补码、反码或移码等编码表示。

（下面以小数为例）

2.1.1 原码表示法

编码规则：最高位为符号位（0—正；1—负），
数值部分与真值的绝对值相同。

例： 真值(X)	原码($[X]_{\text{原}}$)
0.0010	0.0010
-0.1010	1.1010
0.0000	0.0000
0.0000	1.0000

对于纯小数，原码定义为：

$$[X]_{\text{原}} = \begin{cases} X & 0 \leq X < 1 \\ 1-X = 1+|X| & -1 < X \leq 0 \end{cases}$$

原码表示数的范围（以 $n=5$ 为例）

二 进 制 真 值		原 码
零	0.0000	0.0000
	0.0000	1.0000
正数	+0.0001	0.0001

	+0.1111	0.1111
负数	-0.0001	1.0001

	-0.1111	1.1111

∴ 正数有 $2^{n-1}-1$ 个

负数有 $2^{n-1}-1$ 个

零 2 个 (0的原码有两种表示: 00...0和100...0)

原码的特点：

简单、易懂（实质是表示数的符号和绝对值）

乘除法规则较简单

加减法实现比较复杂（需要对符号位进行判断）



2.1.2 补码表示法

1 . 取模的概念

(1) 模

可理解为一个计量器的容量，也可理解为一个计数系统的计数范围，当数值到达该计数系统的模时，计数会重新回到0。

模一般用MOD 表示。

数学中的取模操作是求余数的操作，结果是一个大于或等于0，且小于模M的数。

例如： $15 \text{ MOD } 12 = 3$

或写成： $15 = 3 \text{ (MOD } 12)$

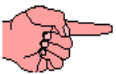
 对于二进制数，进行 2^n 取模运算，其结果就是去掉 2^n 位权及其左边的位后所剩下的值。

例如： $(11\ 0\ .1011)_2 = (0.1011)_2 \text{ (MOD } 2^1)$

2^1

2^0

0	1	0	1	1
---	---	---	---	---

 计算机中硬件（如运算器、寄存器）能表示的数据位数是有限的，所以其运算都是有模运算，当运算结果超过最大表示范围（也就是模）时，就会溢出，并自动舍弃溢出量。

(2) 补数

假定M为模，若数a、b满足 $a+b=M$ ，则称a、b互为补数。

在有模运算中，减去一个数a等于加上该数的补数b

$$X-a = X+b \pmod{M}$$

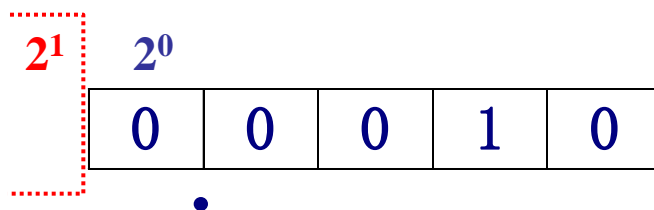
补码正是利用补数概念，把负数映射到正数域中（平移模值，小数的模为2，n位整数的模为 2^n ），从而将数的正负符号数码化。



2. 补码的定义及编码规则

对于纯小数，补码定义为：

$$[X]_{\text{补}} = X + 2^1 \pmod{2}$$



例：	真值(x)	补码($[x]_{\text{补}}$)
	0.0010	0.0010 (= $2^1 + 0.0010 \pmod{2^1}$)
	-0.1010	1.0110 (= $2^1 - 0.1010 \pmod{2^1}$)

? 思考：整数的补码定义式子？

同一数值，不同位数的补码之间的关系？

补码的编码规则：

(1) 当 $X \geq 0$ 时， $[X]_{\text{补}} = X$

例如： $X = 0.11010$,
 $[X]_{\text{原}} = 0.11010$,
 $[X]_{\text{补}} = 0.11010$

(2) 当 $X < 0$ 时，方法有两种：

- a) 各位取反，末位加“1”。
- b) 自低位向高位，尾数的第一个“1”及其右部的“0”保持不变，左部的各位取反。

例如： $X = -0.11010$,
 $[X]_{\text{补}} = 1.00110$

补码表示数的范围（以 $n=5$ 为例）：

二 进 制 真 值		补 码
零	0.0000	0.0000
正数	+0.0001	0.0001

	+0.1111	0.1111
负数	-0.0001	1.1111

	-0.1111	1.0001
	-1.0000	1.0000

∴ 正数有 $2^{n-1}-1$ 个

负数有 2^{n-1} 个

零 1 个

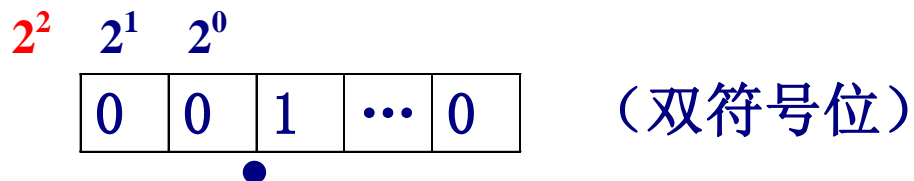


补码最小负数1.00...0对应的真值为-1.00...0,

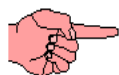
3. 变形补码

采用双符号位或更多的符号位，可扩大数的表示范围，通常在运算时用，以判别运算是否溢出。

变形补码定义： $[x]_{\text{补}} = x + 2^2 \pmod{4}$



例：	X	$[X]_{\text{补}}$	$[X]_{\text{补}}$ (变形补码)
	0.1010	0.1010	00.1010
	-0.1010	1.0110	11.0110



当变形补码的双符号位不同时，其真值超出了一般补码的表示范围。

2.1.3 反码表示法

反码表示法与补码表示法有许多类似之处，对于正数，数值部分与真值形式相同；对于负数，将真值的数值部分按位取反。

反码运算不方便，一般很少用于作算术运算。

例：设一机器字长为**8**位，求所给真值的原码、补码、反码。

$$X1 = 1011$$

$$X2 = -1011$$

$$X3 = 0.1011$$

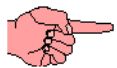
$$X4 = -0.1011$$

$[X1]_{\text{原}} = 00001011$	$[X2]_{\text{原}} = 10001011$	$[X3]_{\text{原}} = 0.1011000$	$[X4]_{\text{原}} = 1.1011000$
$[X1]_{\text{补}} = 00001011$	$[X2]_{\text{补}} = 11110101$	$[X3]_{\text{补}} = 0.1011000$	$[X4]_{\text{补}} = 1.0101000$
$[X1]_{\text{反}} = 00001011$	$[X2]_{\text{反}} = 11110100$	$[X3]_{\text{反}} = 0.1011000$	$[X4]_{\text{反}} = 1.0100111$

例：设一机器字长为**8**位，确定原码、补码、反码所能表示的数的范围？

提示： 若采用纯整数形式：

 若采用纯小数形式：

注意：

真值是没有经过编码的直观数据表示方式，一般用X表示，其值可带正负号；



编码后的数据（如原码、补码、反码等）

已经把正负符号数字化了，

其位数有规定，不能随便忽略任何位置上的0或1。

2.1.4 三种码制的比较与转换

1. 比较

- 原码、反码表示0及正、负数的范围是对称的，而补码0的表示形式是唯一，负数能多表示一个数（绝对值最大的负数），其值等于 -2^n （纯整数）或 -1 （纯小数）。
- 补码的符号位可以和数值位一起参加运算，但原码的符号位和数值位须分开处理；

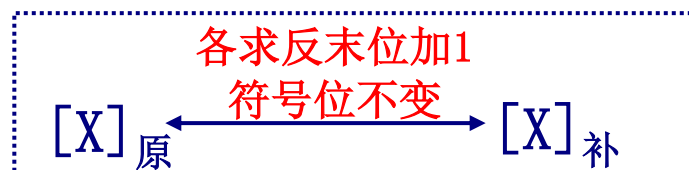
2.1.4 三种码制的比较与转换

2. 转换

(1) $X > 0$ 时, $[X]_{\text{原}} = [X]_{\text{反}} = [X]_{\text{补}} = X$;

(2) $X < 0$ 时,

① 原码与补码之间

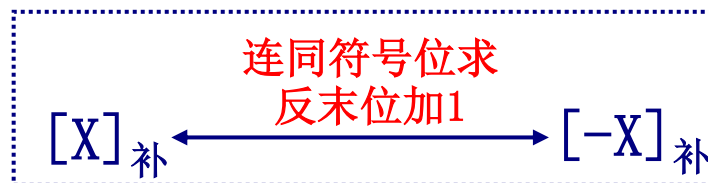


② 原码与反码之间的转换类似, 只是末位不加 1 。

2.1.5 补码机器数的求补(变补)和移位

1. 求补

又称为变补、求补数、求机器负数，即，由 $[X]_{\text{补}}$ 求 $[-X]_{\text{补}}$



例： $[X]_{\text{补}} = 0.0010$, $[-X]_{\text{补}} = 1.1110$;

$[Y]_{\text{补}} = 1.0110$, $[-Y]_{\text{补}} = 0.1010$

 **注意：** 求补与求补码概念的不同。

求补码通常指由真值 X 求 $[X]_{\text{补}}$ ，求补是指对机器数（补码、原码等）求补数。

2.1.5 补码机器数的求补(变补)和移位

2. 右移 (由 $[X]_{\text{补}}$ 求 $[X/2]_{\text{补}}$)

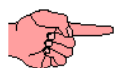
连同符号位一起各位向右移一位, **符号位不变**, 总位数不变。

$$\begin{aligned}\text{例: } [X]_{\text{补}} &= 0.0010, & [X/2]_{\text{补}} &= 0.0001; \\ [Y]_{\text{补}} &= 1.0110, & [Y/2]_{\text{补}} &= 1.1011\end{aligned}$$

3. 左移 (由 $[X]_{\text{补}}$ 求 $[2X]_{\text{补}}$)

各位按位向左移一位, 末位补0。

显然, 只有当 $[X]_{\text{补}}$ 的最高两位相同时, 左移后才是正确的 $[2X]_{\text{补}}$, 否则就溢出了。



无特殊指定时, 定点数认为用补码表示。

课堂练习:

3. 设字长 $M=8$, 求下列补码所对应的 $[X]_{\text{补}}$ 及 $X_{\text{真}}$

$$(1) [2X]_{\text{补}} = 80\text{H}, \quad (2) [X/2]_{\text{补}} = \text{C0H}, \quad (3) [-X]_{\text{补}} = \text{FFH}$$

习题: 1 (1) (3) (5) (6), 3, 4, 7, 9

补充习题2-1:

已知 $[X]_{\text{补}} = 3\text{EH}$, $[Y]_{\text{补}} = \text{DCH}$,

求: $[2X]_{\text{补}}$, $[2Y]_{\text{补}}$, $[1/2 X]_{\text{补}}$, $[1/4 Y]_{\text{补}}$,

$[X]_{\text{原}}$, $[Y]_{\text{原}}$, $[X]_{\text{反}}$, $[Y]_{\text{反}}$,

$[X]_{\text{移}}$, $[Y]_{\text{移}}$

阅读: 英文材料 2-2, 2-3

§ 2.2 数的浮点表示

2.2.1 浮点数及其一般表示格式

引例: $(0.0001011)_2 = (0.1011000)_2 \times 2^{-3}$

调整指数的值相当于改变小数点的位置

$$\text{浮点数 } N = M \times r^E$$

M—尾数(Mantissa)，表示有效数字的带符号纯小数
常用原码或补码表示；

E—阶码(Exponent)，表示小数点位置的带符号整数
常用移码或补码表示；

r—基数(radix)，通常取值2，也可取4，8，16等。

在每台机器中，浮点数的基数r是固定的常数，不必在数码中表示出来。

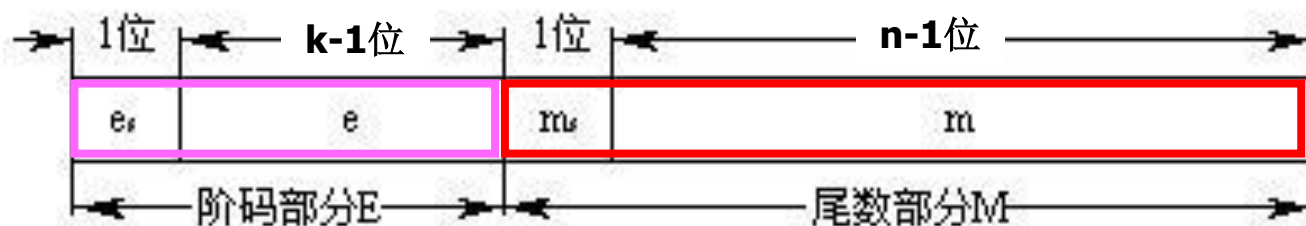
§ 2.2 数的浮点表示

2.2.1 浮点数及其一般表示格式

$$\text{浮点数 } N = M \times r^E$$

∴ 浮点数可用一对定点数表示：〈尾数M； 阶码E〉

其一般格式为：



精度主要由M的位数和r决定

数值范围主要由r和E决定

? 思考：浮点数的表示范围多大？（可对照上述引例来思考）

2.2.2规格化浮点数

1. 浮点数规格化形式

尾数的最高数位必须是一个有效值，以提高数据表示精度。

引例: $(0.0001011)_2 = (0.1011000)_2 \times 2^{-3}$ 规格化数!

规格化数!

设 $r=2$:

$$= (0.0101100)_2 \times 2^{-2} \quad \text{非规!}$$

非規！

若尾数M用原码表示，则规格化尾数的绝对值应满足

$$1/2 \leq |M| < 1$$

这样尾数的最高有效位总是1;

若尾数M用补码表示，则规格化尾数应满足

$$-1 \leq M < -1/2 \quad \text{或} \quad 1/2 \leq M < 1$$

即，尾数最高两位必须相异。

0.1xx...x 和 1.0xx...x ——规！

——规！

0.0xx...x 和 1.1xx...x ——非规！

——非规！



2. 浮点规格化数的数据表示范围

若阶码和尾数均用补码表示，阶码k位，尾数n位，则规格化数典型值为：

典型值	阶码	规格化尾数	真值
最小正数	$10\cdots0$	$0.10\cdots00$	$2^{-1} \times 2^{-2^{k-1}}$
最大正数	$01\cdots1$	$0.11\cdots11$	$(1-2^{-n+1}) \times 2^{2^{k-1}-1}$
绝对值最小负数	$10\cdots0$	$1.01\cdots11$	$-(2^{-1}+2^{-n+1}) \times 2^{-2^{k-1}}$
绝对值最大负数	$01\cdots1$	$1.00\cdots00$	$-1 \times 2^{2^{k-1}-1}$

尾数为0时，不论阶码为何值，一般都当做机器0处理。此时应把阶码置成最小值（绝对值最大的负数）。

2.2.3 阶码的移码表示法

移码：在真值X上加一个常数（偏置值），使数据的正负符号数字化的一种编码方法。

（相当于X在数轴上向正方向平移了一段距离，通常用于表示整数）

$$[X]_{\text{移}} = \text{偏置值} + X$$

对于n位定点整数，偏置值可取 2^{n-1} ，此时其编码规则较简单；也可取 $2^{n-1}-1$ 。

移码的编码规则：（对于 n 位定点整数，偏置值取 2^{n-1} ）

$[X]_{\text{移}}$ 与 $[X]_{\text{补}}$ 符号位(最高位)相反，其它各位都相同。

例 1: $X = 1011101$

$$[X]_{\text{移}} = 2^7 + X = 10000000 + 1011101 = 11011101$$

$$[X]_{\text{补}} = 01011101$$

例 2: $Y = -1011101$

$$[Y]_{\text{移}} = 2^7 + Y = 10000000 - 1011101 = 00100011$$

$$[Y]_{\text{补}} = 10100011$$

用移码表示阶码的优点：

(1) 移码可视为无符号数，全为0时对应的真值最小，全为1时对应的真值最大，有助于两个浮点数进行阶码的大小比较；

(2) 简化机器中的判零电路。当阶码全为0，尾数也全为0时，表示机器零。

2.2.4 定点、浮点表示法与定点、浮点计算机

1. 定点、浮点表示法的比较

数值范围：浮点表示法远远大于定点表示法；

数据分布：定点数分布均匀，浮点数越靠近数轴的原点
分布密度大；

数据精度：一般认为浮点数的精度低于定点数；

数学运算：浮点运算要比定点运算复杂得多。

2. 定点机与浮点机

通常可以将计算机分为几档：

(1) 定点机

以定点运算为主，浮点运算是通过软件来实现的。

（低档微、小型机）

(2) 定点机+浮点运算部件

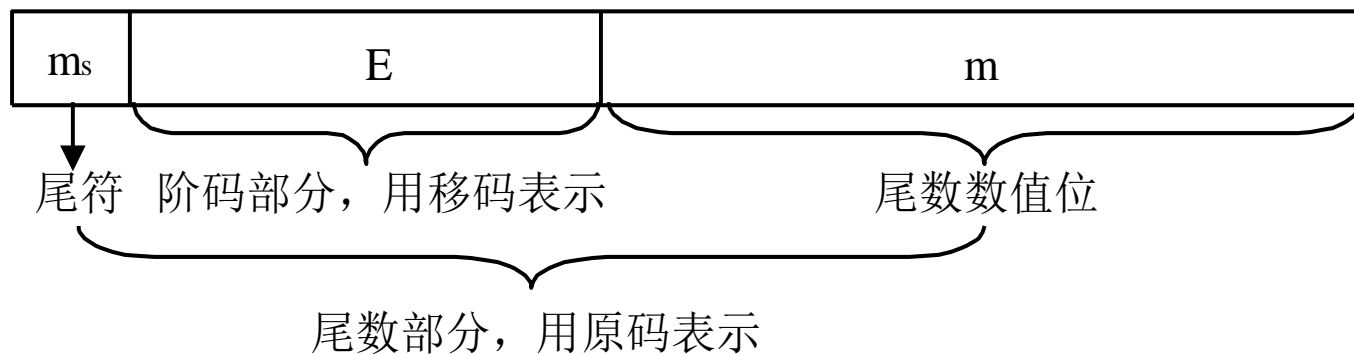
浮点运算部件是专门用于对浮点数进行运算的部件。

（微、小型机）

(3) 浮点机

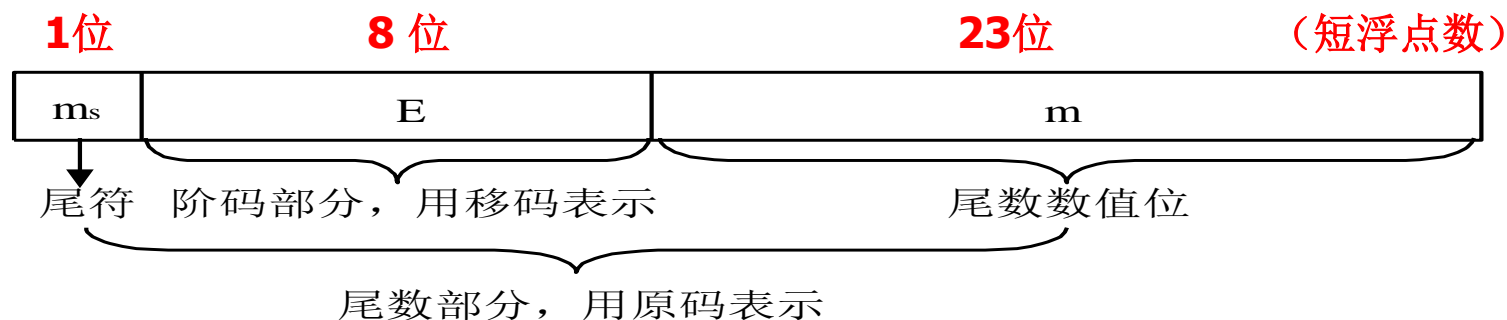
具有浮点运算指令和基本的浮点运算器。（大、中型机）

2.2.5 实用浮点数举例---IEEE754标准



类 型	数符 m_s	阶码 E	尾数 m	总位数	偏置值	
短浮点数	1	8	23	32	7FH	127
长浮点数	1	11	52	64	3FFH	1023
临时浮点数	1	15	64	80	3FFFH	16383

2.2.5 实用浮点数举例---IEEE754标准



短浮点数格式定义：

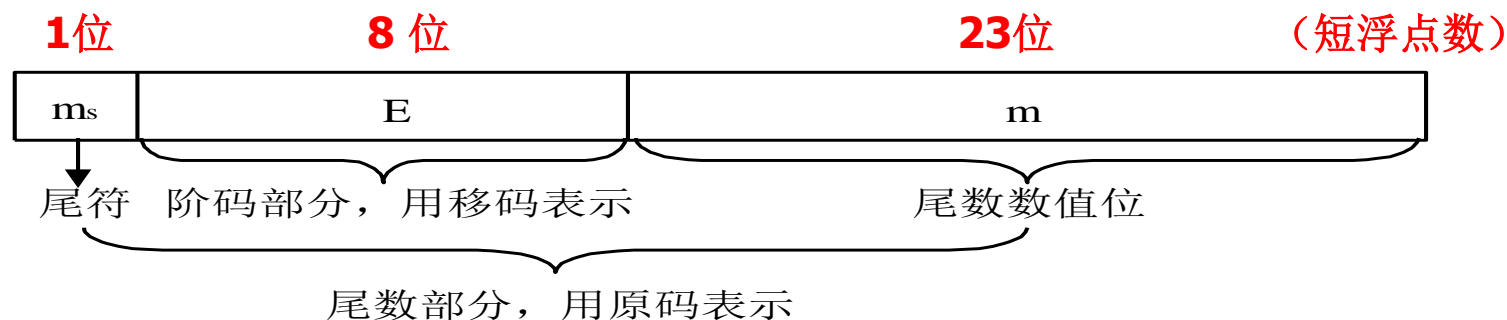
最高位为数符位；

其后是8位阶码，以2为底，阶码的偏置值为127；

尾数实际上是24位。隐含尾数最高数位1，且这一位1的位权为 2^0 ，其余23位是尾数的纯小数，用原码表示。

 **注意：IEEE754隐含尾数最高数位1，这一位1的位权为 2^0 ，在浮点数中不表示出来，因此尾数实际上是24位。**

2.2.5 实用浮点数举例---IEEE754标准



例：将 $(100.25)_{10}$ 转换成短浮点数格式 (P31例13)

$$(100.25)_{10} = (1100100.01)_2 = (1.10010001)_2 \times 2^6$$

符号位 = 0

阶码的移码 = $110 + 1111111 = 10000101$

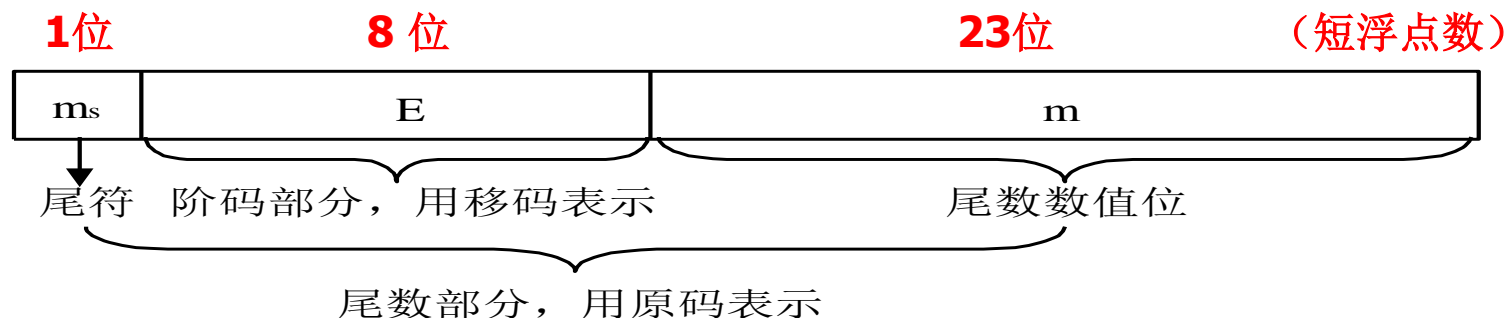
尾数纯小数部分 = 100100010000000000000000

∴ 短浮点数代码为：

0;100 0010 1;100 1000 1000 0000 0000 0000

即：42C88000H

2.2.5 实用浮点数举例---IEEE754 标准



例：把短浮点数C1C90000H转换成为十进制数（P31例14）

C1C90000H = 1;10000011;100100100000000000000000

数符：负 (0正1负)

阶码真值： 10000011-1111111=100 (移码-偏置值)

$$\begin{aligned}
 \therefore \text{该浮点数} &= -(1.1001001)_2 \times 2^4 \\
 &= (-11001.001)_2 \\
 &= -25.125
 \end{aligned}$$

课堂练习:

4. 某浮点数字长32位，格式如下。其中阶码部分8位，以2为底，补码表示；尾数部分一共24位（含1位数符），补码表示。现有一浮点代码为 $(8C5A3E00)_{16}$ ，试写出它所表示的十进制真值。（P50 15题）



5. 将十进制数28.75转换为IEEE754短浮点数。（P50 17(1)题）



习题： 12, 15, 17(1) (3) (5),
18(1) (3), 20, 21

阅读：英文材料 2-4

§ 2.3 非数值数据的表示

2.3.1 字符和字符串的表示

1. ASCII字符编码

ASCII----**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)

用7位二进制表示一个字符，它包括数字0~9、A~Z、a~z等128个字符。计算机中通常用1字节存放1个字符。

(见P34表2-5)

2. 字符串的存放

字符串采用向量存放法，即在存储器中占用一片连续的空间，每个字节存放一个字符的ASCII码。

2.3.2 汉字的表示

1. 汉字输入编码

用于将汉字输入到计算机内部。

要求：操作简单、容易记忆、码位短、输入速度快。

类型：拼音编码、字形编码、数字编码、整字编码、其他(如语音识别和手写输入)等。

例如，简拼、全拼、五笔、区位、智能ABC 等。

2.3.2 汉字的表示

2. 国标码和汉字内码（汉字机内码）

(1) 区位码:

区位码用四位十进制数表示**6763**个汉字。前两位叫做区码，分为**94**个区（**01-94**）；后两位叫做位码（**01-94**），每个区中包含**94**个位（汉字）。

(2) 国标码**GB2312-80**:

国家标准“信息交换用汉字编码字符集(基本集)”的简称。共收集常用汉字6763个、各种图形符号682个。
一个汉字用两个字节表示，每个字节的最高位为0。

(3) 汉字机内码: 计算机内部汉字信息存储、交换和处理的代码。

汉字机内码每个汉字也是占**2**个字节，但为了保证计算机中汉字处理系统的中西文兼容，机内码的每个字节最高位为**1**，而**ASCII**码的字节最高位为**0**。



2.3.2 汉字的表示

区位码、国标码和汉字内码之间的关系

$$\text{国标码} = \text{区位码 (十六进制)} + 2020\text{H}$$

$$\text{机内码} = \text{国标码} + 8080\text{H}$$

$$= \text{区位码 (十六进制)} + \text{A0A0H}$$

例：“中”字

区位码：5448（在54区的48位上）

（在36H区的30H位上）

国标码 5650 H

机内码 D6D0 H

36H	30H
+20H	+20H
<hr/> 56H	<hr/> 50H

2.3.2 汉字的表示

3. 汉字字模码

用于汉字的字形显示输出。有向量编码和点阵码。

点阵码：

每个汉字采用 $N \times N$ 的点表示，形成若干字节的二进制编码。

例如， 16×16 汉字点阵每个汉字用32字节表示。

? 思考： 32×32 汉字点阵每个汉字用多少字节表示？

向量编码：

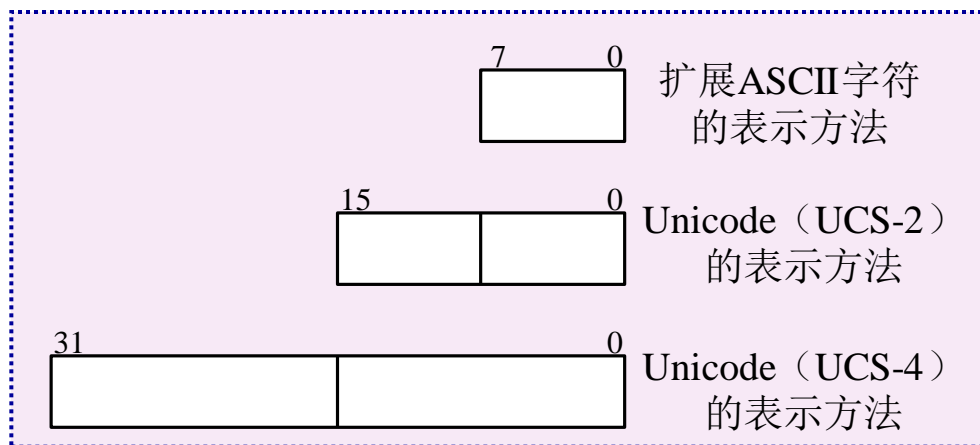
2.3.3 统一代码(Unicode)

能表示6800种语言中任意一种语言里使用的所有符号的一种全新的编码方法。这种字符集被称为基本多语言平面（BMP）

USC-2: 用2字节表示Unicode字符;

USC-4: 用4字节表示Unicode字符。

PC机中表示语言符号的3种方法:



§ 2.4 十进制数和数串的表示

2.4.1 十进制数的编码

用四位二进制数来表示一位十进制数，称为二进制编码的十进制数（Binary-Code Decimal），简称BCD码。

常见的BCD编码有8421码、2421码、余3码等。

常见的BCD编码有8421码、2421码、余3码等。

十进制数	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

例：两位十进制数39

可表示为 $(0011\ 1001)_{8421}$ 或 $(0110\ 1100)_{\text{余3码}}$

2.4.2 十进制数串

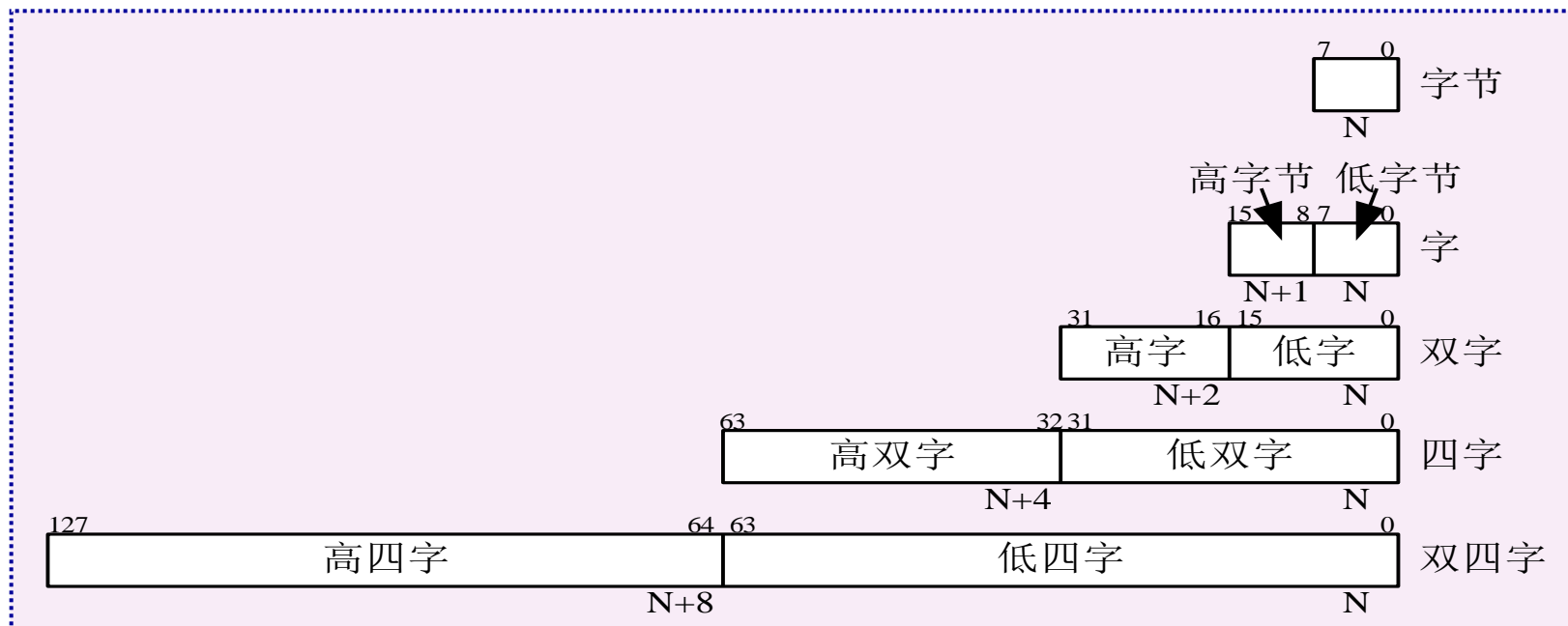
非压缩的十进制数串表示法：一个字节存放一个十进制数或符号的ASCII-7码；

压缩的十进制数串表示法：一个字节存放两位BCD码表示的十进制数。

§ 2.5 现代微机系统数据表示举例

现代的微机系统大多采用**Intel**系列的微处理器，近年来，**Intel**的微处理器有了极大的发展，形成了**IA-32**结构。最新的Itanium处理器是第一款真正64位产品（IA-64）。

IA-32结构的基本数据类型是字节、字、双字、四字和双四字。



§ 2.5 现代微机系统数据表示举例

1. 无符号整数

包含字节、字、双字和四字的无符号的二进制数。

2. 带符号整数

包含字节、字、双字和四字的带符号的二进制定点整数。

3. 浮点数

采用IEEE 754标准所规定的格式。

4. 指针数据

指针是主存单元的地址。

IA-32 结构定义了两种类型的指针：近指针（32位）和远指针（48位）。

.....

? 思考：它们的数据的表示范围？

§ 2.6 数据校验码

能够发现错误或能够自动纠正错误的数据编码。

2.6.1 奇偶校验码

在若干位有效信息(如一个字节)上增加一个二进制位(校验位)，组成校验码。常用于检验内存数据存取过程中是否出现错误。

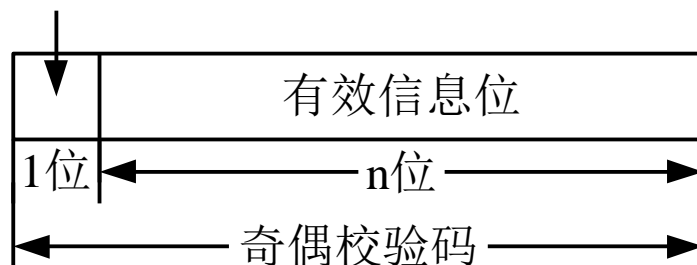
奇校验——整个校验码（有效信息位和校验位）中“1”的个数为奇数（常用）。

偶校验——整个校验码中“1”的个数为偶数。

§ 2.6 数据校验码

2.6.1 奇偶校验码

例：奇偶校验位



有效信息 (8 位)	奇校验码 (9 位)	偶校验码 (9 位)
00000000	100000000	000000000
01010100	001010100	101010100

注：校验位的具体位置可以事先设定。

§ 2.6 数据校验码

2.6.1 奇偶校验码

1. 编码

由有效信息产生1校验位形成校验码

以有效信息1字节为例，

$$\text{奇校验位} = \overline{D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0}$$

§ 2.6 数据校验码

2.6.1 奇偶校验码

2. 校验（解码）

由校验码校验该码是否出错，可以检测出一位错误（或奇数位错误）。

以有效信息1字节为例（上例）：

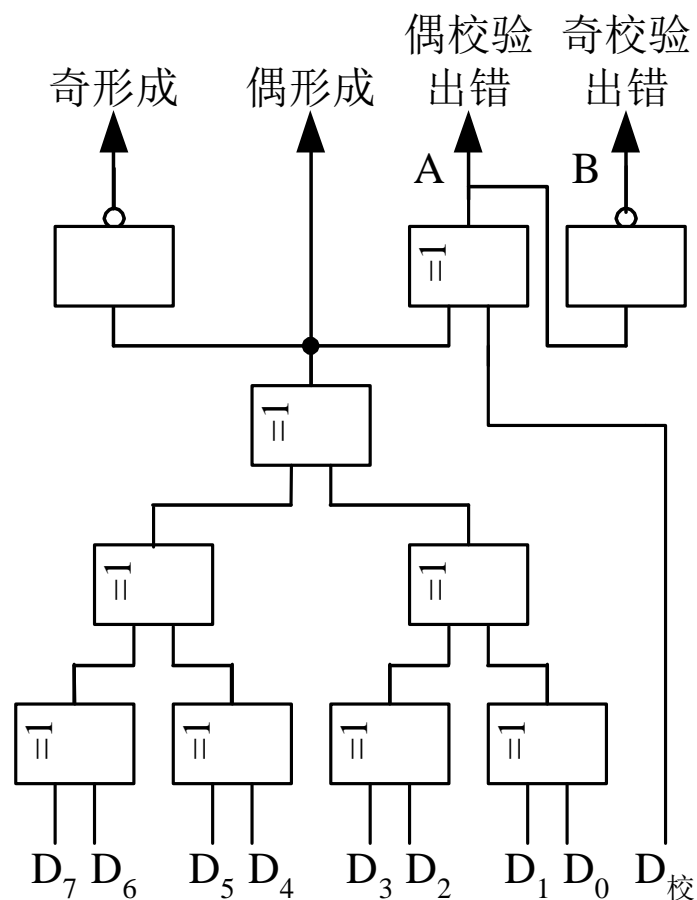
$$\text{奇校验出错} = \overline{D_{\text{校}} \oplus D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0}$$

§ 2.6 数据校验码

2.6.1 奇偶校验码

3. 编解码的电路实现

可用异或门构成。



习题：P50 12, 15, 17(1)(3)(5),
18(1)(3), 20, 21

阅读：英文材料 2-5, 2-6

补充题：

计算机存储程序概念的特点之一是把数据和指令都作为二进制信号看待。今有一计算机字长32位 ($D_{31} \sim D_0$)，数符位是第31位。对于二进数

1000 1111 1110 1111 1100 0000 0000 0000 ,

- (1) 表示一个补码整数，其十进制值是多少？
- (2) 表示一个无符号整数，其十进制值是多少？
- (3) 表示一个IEEE 754标准的单精度浮点数，其值是多少？

本章重点:

1. 进制之间的相互转换是最基本的, 应当熟练掌握;
2. 定点数的原码、补码、变形补码和反码表示, 补码还应特别注意:
 - ① 最小的补码很特殊, 没有对应的原码和反码;
 - ② “变补”和“求补码”概念的不同;
 - ③ 补码乘以2(算术左移)和乘以1/2(算术右移)的结果;
3. 浮点数的规格化概念, 真值 \longleftrightarrow 浮点数表示形式;
4. 常见的字符编码方法(ASCII码)、区位码、汉字国标码、机内码、BCD码的特点。
5. 奇偶校验码的校验原理及校验位形成方法;
6. 学会看原文资料, 并熟悉其专业名词和术语。

习题：P47 12, 15, 18, 23, 24（第3版）

补充题：

计算机存储程序概念的特点之一是把数据和指令都作为二进制信号看待。今有一计算机字长32位 ($D_{31} \sim D_0$)，数符位是第31位。对于二进数

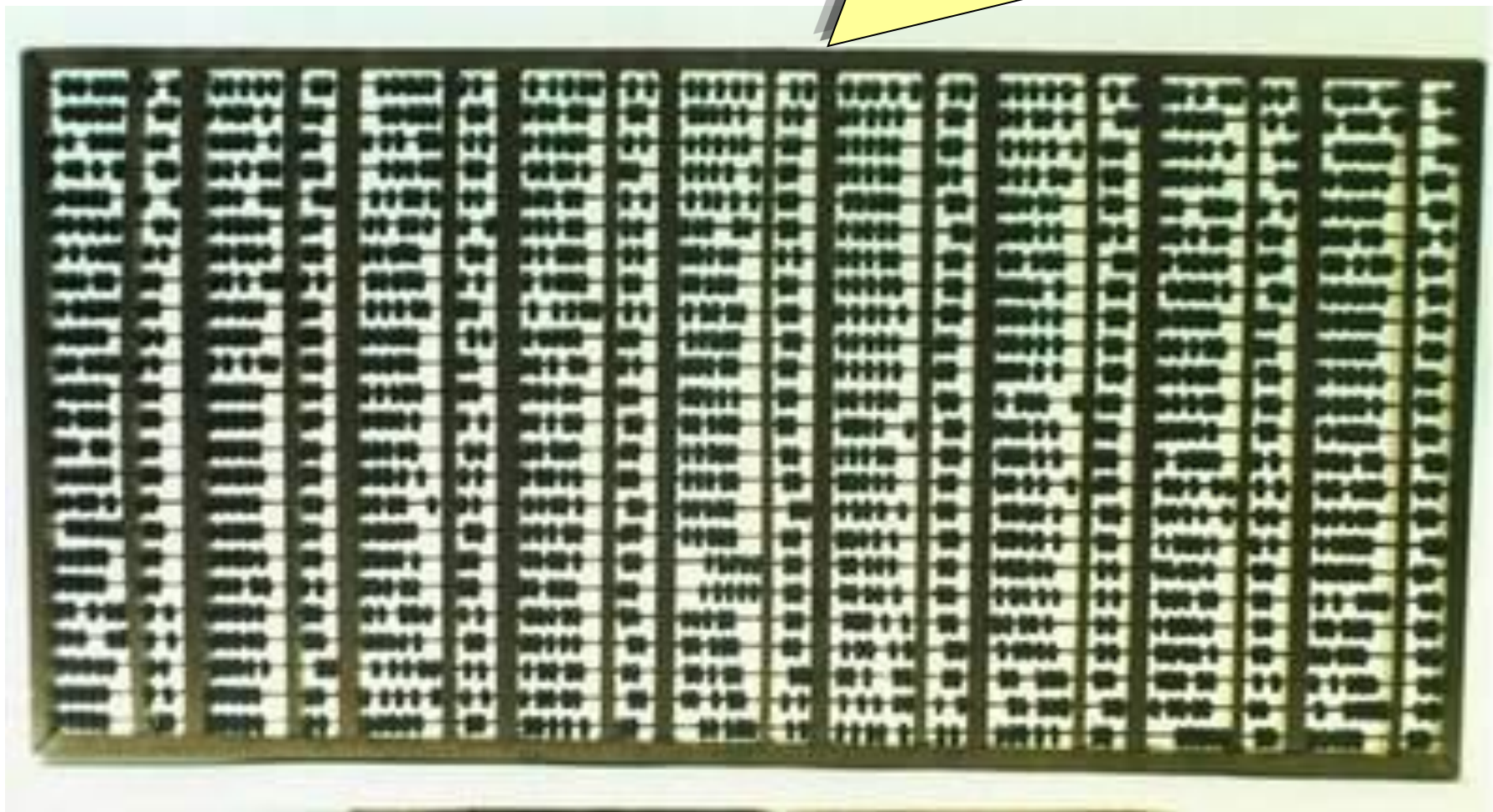
1000 1111 1110 1111 1100 0000 0000 0000 ,

- (1) 表示一个补码整数，其十进制值是多少？
- (2) 表示一个无符号整数，其十进制值是多少？
- (3) 表示一个字长为32位的浮点数，其中阶码8位，移码表示（偏移值27），尾数24位，补码表示，其十进制值是多少？

计算机发展

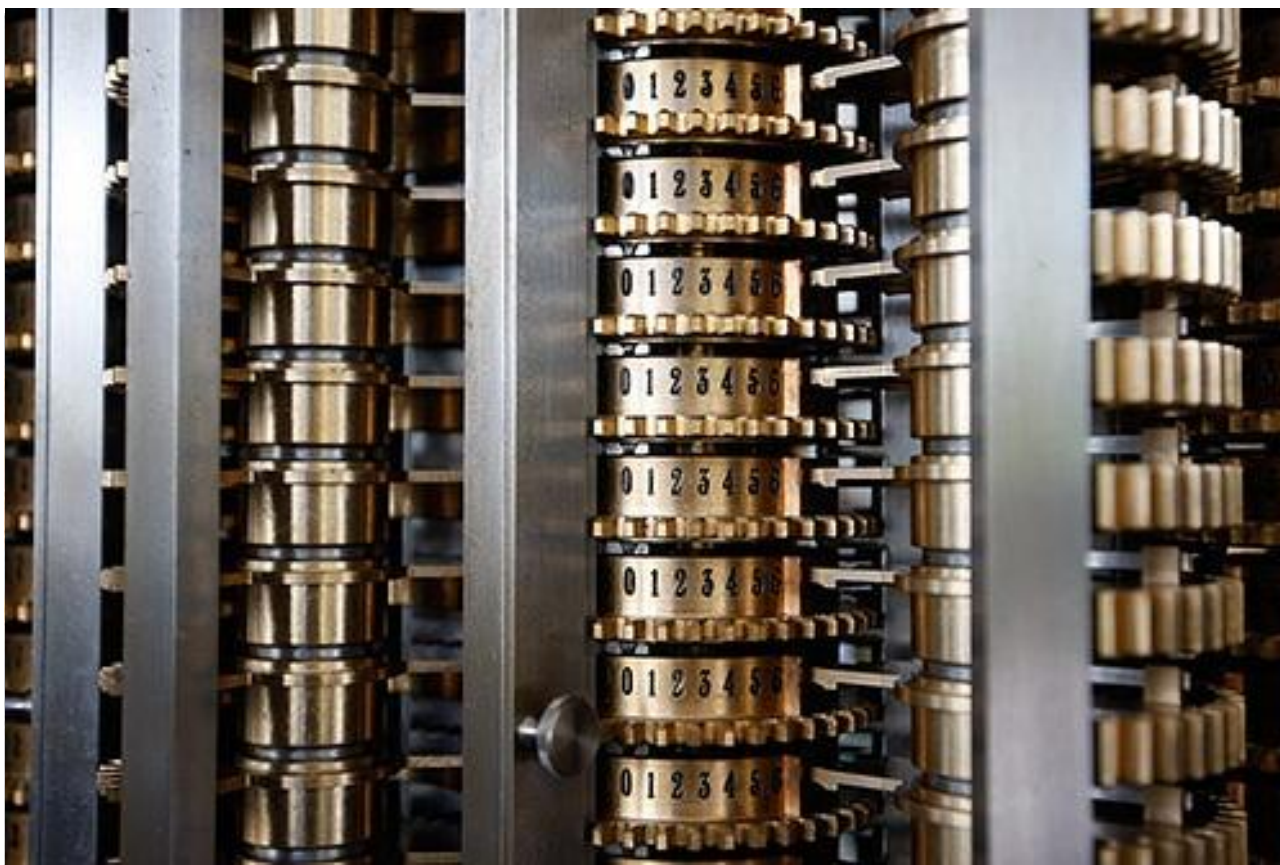
1. 算盘

特大型清代算盘，边长60厘米，宽40厘米，全木结构，内分9层，每层均为标准、完整的上2下5珠24档算盘，全盘算珠总数多达1512个。



计算机发展

2. 机械计算机的诞生



计算机发展

3. 电子计算机问世

1946年，第一台电子计算机ENIAC (Electronic Numerical Integrator And Calculator)



 5000次加法/秒

 体重28吨

 占地170m²

 18800只电子管

 1500个继电器

 功率150KW

主要用于计算弹道和氢弹的研制！

计算机发展

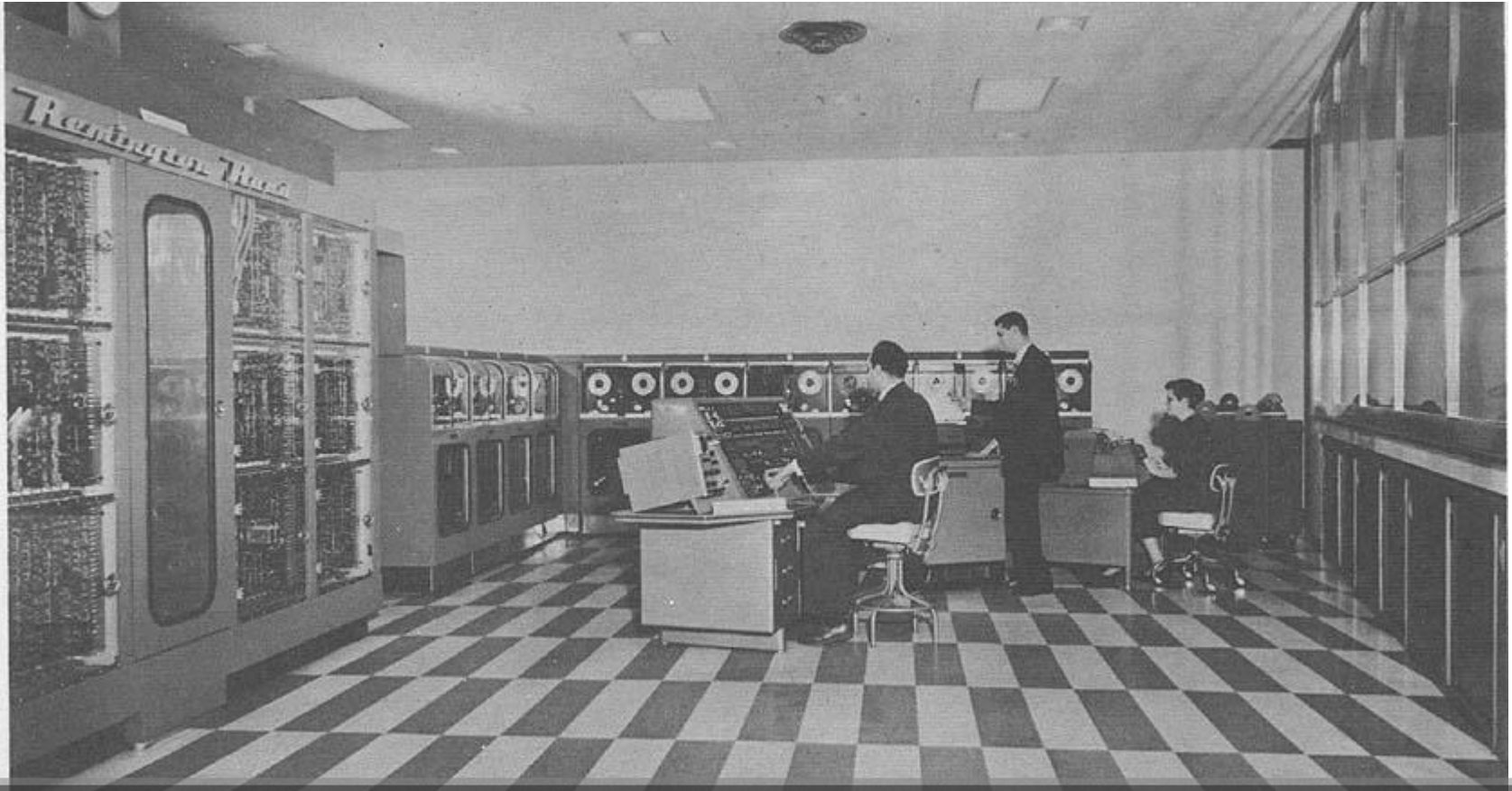
4. 晶体管计算机

1947 年: Bell 实验室的 William B. Shockley 、 John Bardeen 和 Walter H. Brattain 发明了晶体管, 开辟了电子时代新纪元。



1947年, 贝尔实验室发明晶体管。参与这项研究的约翰·巴丁 (John Bardeen)、威廉·肖克利 (William Shockley)、华特·豪舍·布拉顿 (Walter Houser Brattain) 于1956年获诺贝尔物理学奖。

计算机发展



1951年：UNIVAC-1 ——第一台商用计算机系统诞生，设计者是J. Presper Eckert 和John Mauchly 。被美国人口普查部门用于人口普查，标志着计算机进入了商业应用时代。UNIVAC-1是使用晶体管的计算机，是第二代计算机的代表。

计算机发展

5. 集成电路为现代计算机铺平道路

1958年9月12日，德州仪器的工程师Jack Kilby Jack Kilby展示了第一个集成电路，或叫微晶片。这块原始的芯片不是很耐看，在此后几十年里这位2000年的诺贝尔物理学奖得主说，早知道这块世界上第一个可工作集成电路会不断向人展示，他应该把它打磨得漂亮点。从现在的标准看，它确实非常原始，电线突出，只包含一个晶体管，几个电阻器和一个电容器，但在演示中它成功的在示波屏上产生了一个正弦波，从那时起，技术发生了翻天覆地的变化



计算机发展

6. 当代计算机技术渐入辉煌

随着超大规模集成电路和微处理器技术的进步，计算机进入寻常百姓家的技术障碍逐渐被突破。特别是在Intel公司发布了其面向个人用户的微处理器8080 之后，这一浪潮终于汹涌澎湃起来，同时也催生出了一大批信息时代的弄潮儿，如Stephen Jobs(史蒂芬·乔布斯)、Bill Gates(比尔·盖茨)等，至今他们对整个计算机产业的发展还起着举足轻重的作用。在此时段，互联网技术和多媒体技术也得到了空前的应用与发展，计算机真正开始改变我们的生活。



计算机技术

计算机技术（ computer technology ）是计算机领域中所运用的技术方法和技术手段。计算机技术具有明显的综合特性，它与**电子工程、应用物理、机械工程、现代通讯技术、和数学**等紧密结合。

