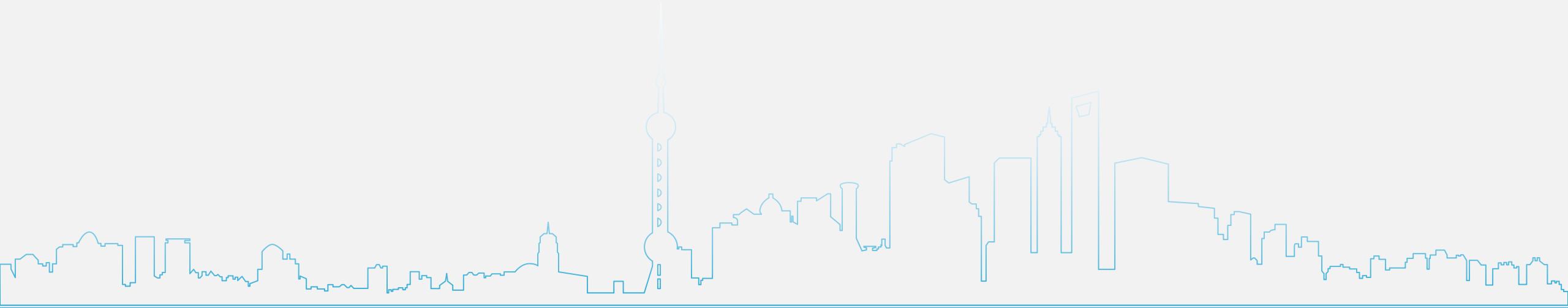


第三章　数据表示

秦磊华　计算机学院

1. 机器数

- 机器数内部使用的数据（与真值对应）
- 二进制数
- 是一种将数据符号数值化的一数据表示方法
- 符号数值化的方法?

用0、1表示
- 表示简单
- 识别容易
- 处理简单!

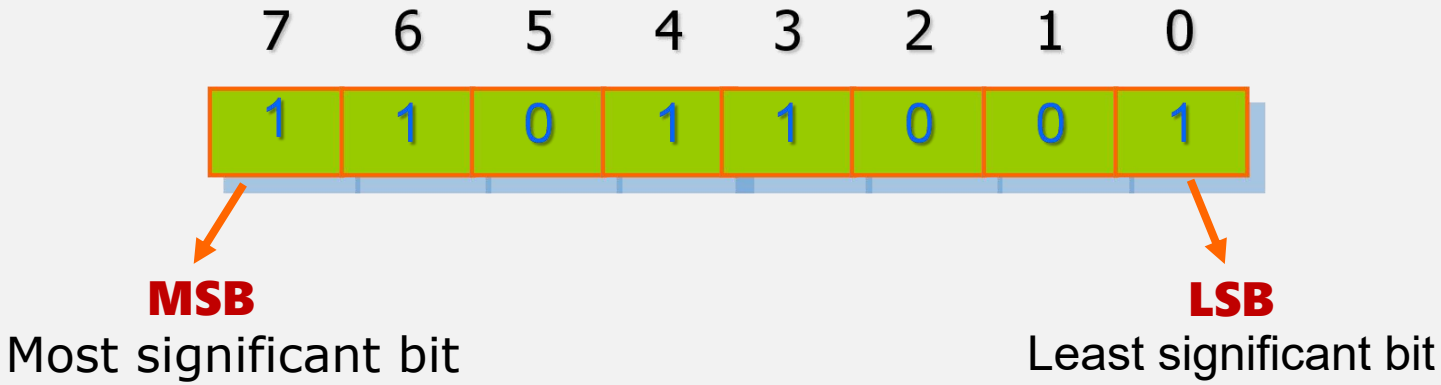
2. 定点数

（整数） $X_nX_{n-1}X_{n-2}...X_1X_0$.

（小数） $X_n.X_{n-1}X_{n-2}...X_1X_0$

$X_i \in \{0, 1\}$

- 表示n+1位二进制定点数(含符号位)需要多少位二进制?
- 符号位放在哪里比较好?



3. 原码

$X_nX_{n-1}X_{n-2}...X_1X_0$ （字长n+1位，数值位为n位）

1) 表示方法： 符号位: 0表示正数，1表示负数，数值位与真值相同

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

整数

x = + 1011001 ⇢ $[x]_{原}$ = 01011001

x = - 1011001 ⇢ $[x]_{原}$ = 1 1011001

3. 原码

$$X_n . X_{n-1}X_{n-2}\cdots X_1X_0$$

1) 表示方法： 符号位: 0表示正数，1表示负数，数值位与真值相同

$$[X]_{原}=\begin{cases} X & 0\le X<1 \\ 1-X & -1<X\le 0 \end{cases}$$

小数

$x = +0.1011001 \dashrightarrow [x]_{原} = 0.1011001$

$x = -0.1011001 \dashrightarrow [x]_{原} = 1.1011001$

3. 原码

2) 原码数据表示范围

$$X_nX_{n-1}X_{n-2}\cdots X_1X_0$$

111…1 ~ 01111…1 ⇢ $[-(2^n-1),\ 2^n-1]$

11111111 ~ 01111111 ⇢ [-127, 127]

1.1111…1 ~ 0.1111…1 ⇢ $[-(1-2^{-n}),\ 1-2^{-n}]$

1.1111111 ~ 0.1111111 ⇢ $[-(1-2^{-7}),\ 1-2^{-7}]$

3. 原码

3) 原码数据表示的特点

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

$$[X]_{原} = \begin{cases} X & 0 \le X < 1 \\ 1 - X & -1 < X \le 0 \end{cases}$$

- 表示简单: 只需要对符号位处理
- 0的表示不唯一: 0000...0 , 1000...0
- 符号位不参加运算
- 加减运算较为复杂，不能直接运算
- 数据表示区间对称

$$[-(1-2^{-n}) , 1-2^{-n}]$$

$$[-(2^n-1) , 2^n-1]$$

4. 反码

$X_nX_{n-1}X_{n-2}...X_1X_0$（字长n+1位，数值位为n位）

1) 表示方法： 0表示正数，数值位与真值或原码相同
1表示负数，数值位与真值或原码相反

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

整数

$x = +1011001 \dashrightarrow [x]_{反} = 0\ 1011001$

$x = -1011001 \dashrightarrow [x]_{反} = 1\ 0100110$

如何实现？

非门阵列

4. 反码

$$X_n.X_{n-1}X_{n-2}\cdots X_1X_0$$

1) 表示方法： 0表示正数，数值位与真值或原码相同
1表示负数，数值位与真值或原码相反

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

小数

x = + 0.1011001 ⇢ $[x]_{反}$ = 0.1011001

x = - 0.1011001 ⇢ $[x]_{反}$ = 1.0100110

4. 反码

2) 反码数据表示范围

$X_nX_{n-1}X_{n-2}\cdots X_1X_0$

反码：10000…0 ~ 01111…1 ⇢ $[-(2^n-1),\ 2^n-1]$

（真值：-1111…1 ~ 01111…1）

10000000 ~ 01111111 ⇢ [-127, 127]

（真值：-1111…1 ~ 01111…1）

1.0000…0 ~ 0.1111…1 ⇢ $[-(1-2^{-n}),\ 1-2^{-n}]$

1.0000000 ~ 0.1111111 ⇢ $[-(1-2^{-7}),\ 1-2^{-7}]$

4. 反码

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

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

3) 反码数据表示的特点

- 表示相对原码复杂: 负数需对数据位取反
- 0的表示不唯一： 0000...0 , 1111...1
- 运算相对复杂： 加运算需将进位位与最低位相加

```
    1101   (-2)
+   1010   (-5)
  1 0111
      1
  ------
    1000   (-7)
```

- 符号位参加运算
- 数据表示区间对称

$[-(1-2^{-n}),\ 1-2^{-n}]$

$[-(2^n-1),\ 2^n-1]$

5. 补码

$$X_nX_{n-1}\cdots X_1X_0$$

1)模的概念：符号位进位位的权值

- 机器字长不同，模(符号位进位位的权值)不同
- 机器字长为n，则权值为2^n

5. 补码

$X_nX_{n-1}X_{n-2}...X_1X_0$（字长n+1位,整数位为n）

2)表示方法： 0表示正数，数值位与真值或原码相同

1表示负数，数值位在反码最低位加1

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

x = + 1011001 ⇢ $[x]_{补}$ = 01011001

x = - 1011001 ⇢ $[x]_{补}$ = 10100111

x = - 1011001

如何实现？

↓

求补电路

5. 补码

$$X_n . X_{n-1}X_{n-2}\cdots X_1X_0$$

2)表示方法：　0表示正数，数值位与真值或原码相同

1表示负数，数值位在反码最低位加1

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

$x = +0.1011001$ ⟶ $[x]_{补} = 0.1011001$

$x = -0.1011001$ ⟶ $[x]_{补} = 1.0100111$

5. 补码

3) 补码数据表示范围

$X_nX_{n-1}X_{n-2}\cdots X_1X_0$

10000…0 ~ 01111…1 ⇢ $[-2^n, 2^n-1]$

- (11111…1 +1)

10000000 ~ 01111111 ⇢ [- 128, 127]

1.0000…0 ~ 0.1111…1 ⇢ $[-1, 1-2^{-n}]$

1.0000000 ~ 0.1111111 ⇢ $[-1, 1-2^{-7}]$

5. 补码

4) 补码数据表示的特点

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

- 表示更为复杂
- 0的表示唯一： 0000...0
- 运算简单： 符号位进位位按模舍去
- 数据表示区间不对称,大于同字长的原码和反码

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

$$[-1,\ 1-2^{-n}]$$

$$[-2^n,\ 2^n-1]$$

5. 补码　　6) 变形补码

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

MOD 模2^{n+2}

$$[X]_{补}=\begin{cases} X & 0\le X<1 \\ 4+X & -1\le X<0 \end{cases}$$

MOD 模2^2

x = + 1011001 ⇢ $[x]_{补}$ = 00 1011001

x = - 1011001 ⇢ $[x]_{补}$ = 11 0100111

5. 补码

7) 补码符号位扩展

$[X]_{补}$= 0101010111000011　　$[Y]_{补}$= 10011100　　求 $[X]_{补}+[Y]_{补}$= ?

```
  0101010111000011
+         10011100
------------------
  ??????????????????
```

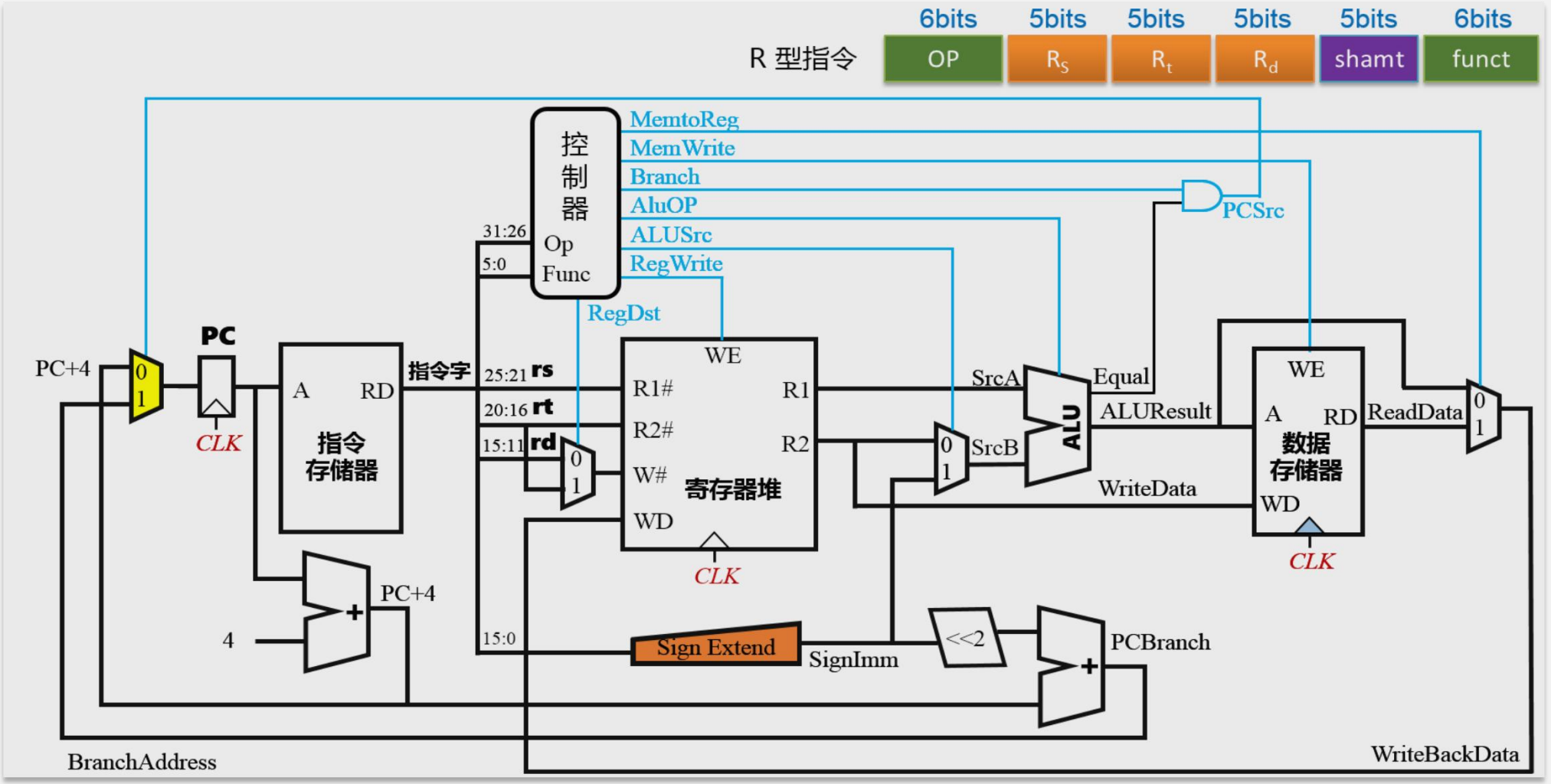
```
  0101010111000011
+ 1111111110011100
------------------
  0101010101011111
```

结论:

- 不同位数的整数补码相加时,位数少的向位数多的数进行符号扩展
- 通过符号扩展电路实现

5. 补码

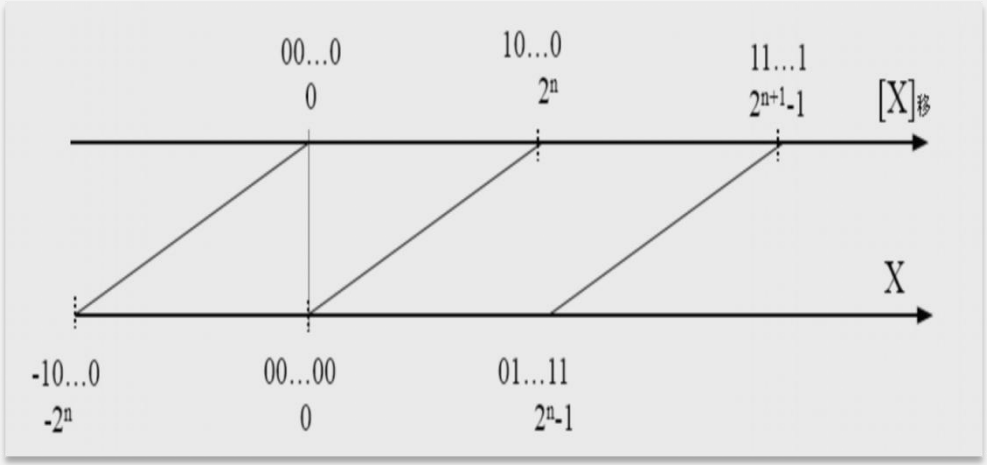
7) 补码符号位扩展



6. 移码(增码)

$X_nX_{n-1}X_{n-2}...X_1X_0$

$[x]_{移} = 2^n + x \qquad -2^n \leqslant x < 2^n$



- 与补码的符号位相异，数据位相同

  $X=+10101 \quad [X]_{移} = 2^5 + 10101 = 110101$

  $X=-10101 \quad [X]_{移} = 2^5 - 10101 = 001011$

- 仅用于表示整数，通常表示浮点数的阶码

保持数据原有大小顺序，便于比较

7. 定点数据表示小结

将十进制值X(-127，-1，0，1，127)用四种机器数表示

X	真值	$[X]_{原}$	$[X]_{反}$	$[X]_{补}$	$[X]_{移}$
-127	-0111111	11111111	10000000	10000001	00000001
-1	-0000001	10000001	11111110	11111111	01111111
0	+0000000	10000000	11111111	00000000	10000000
		00000000	01111111		
1	+0000001	00000001	00000001	00000001	10000001
127	+1111111	01111111	01111111	01111111	11111111

7. 定点数据表示小结

n+1位定点数，数据位n位不同机器数的数据表示范围

	定点整数		定点小数	
原码/反码	$[-(2^n-1), 2^n-1]$	$(-2^n, 2^n)$	$[-(1-2^{-n}), 1-2^{-n}]$	$(-1, 1)$
补码	$[-2^n, 2^n-1]$	$[-2^n, 2^n)$	$[-1, 1-2^{-n}]$	$[-1, 1)$
移码	$[-2^n, 2^n-1]$	$[-2^n, 2^n)$	小数无移码	

8. C语言中的定点数

1)无符号整数

- unsigned char
- unsigned short
- unsigned int
- 一般用于地址运算，编号表示

2)有符号整数

- char short int long
- 采用补码表示

3)关注无符号数与有符号数取值范围的不同(以8位补码为例)

-128 ~ 127 VS 0 ~ 255

3.4 数值数据表示

4)32位机器上的程序

```
main()
{
    int x=-1;
    unsigned u = 2147483648;
    printf ("x = %u = %X = %d\n",x,x,x);
    printf ("u = %u = %X = %d\n",u,u,u);
    return;
}
```

真值赋值

真值输出

机器码输出

x = 4294967295 = FFFFFFFF = -1

u = 2147483648 = 80000000 = -2147483648

9. 汇编语言中的数据类型 ？

- 寄存器、内存数据没有数据类型
- 具体类型取决于指令操作符

ISA	无符号运算	有符号运算	浮点运算
X86	ADD/SUB 加减		FADD/FSUB 加减
	MUL/DIV 乘除	IMUL/IDIV 乘除	FMUL/FDIV 乘除
MIPS32	ADDU/SUBU 加减	ADD/SUB 加减	ADD.S ADD.D /SUB.S SUB.D 加减
	MULTU/DIVU 乘除	MULT/DIV 乘除	MUL.S MUL.D / DIV.S DIV.D 乘除
RISC-V32	ADD/SUB 加减		FADD.S FADD.D / FSUB.S FSUB. D 加减
	MULHU/DIVU 乘除	MULH/DIV 乘除	FMUL.S FMUL.D / FDIV. S FDIV. D 乘除

10. 浮点数据表示

1) 为什么要引入浮点数?

- 问题：三位十进制数能表示的最大数?
- 表示太阳质量 2×10^{30}kg需要多少十进制位? 3 or 5
- 机器字长确定时，浮点数能表示更大范围或更高精确度的数据；

  如电子的质量 9×10^{-28}g

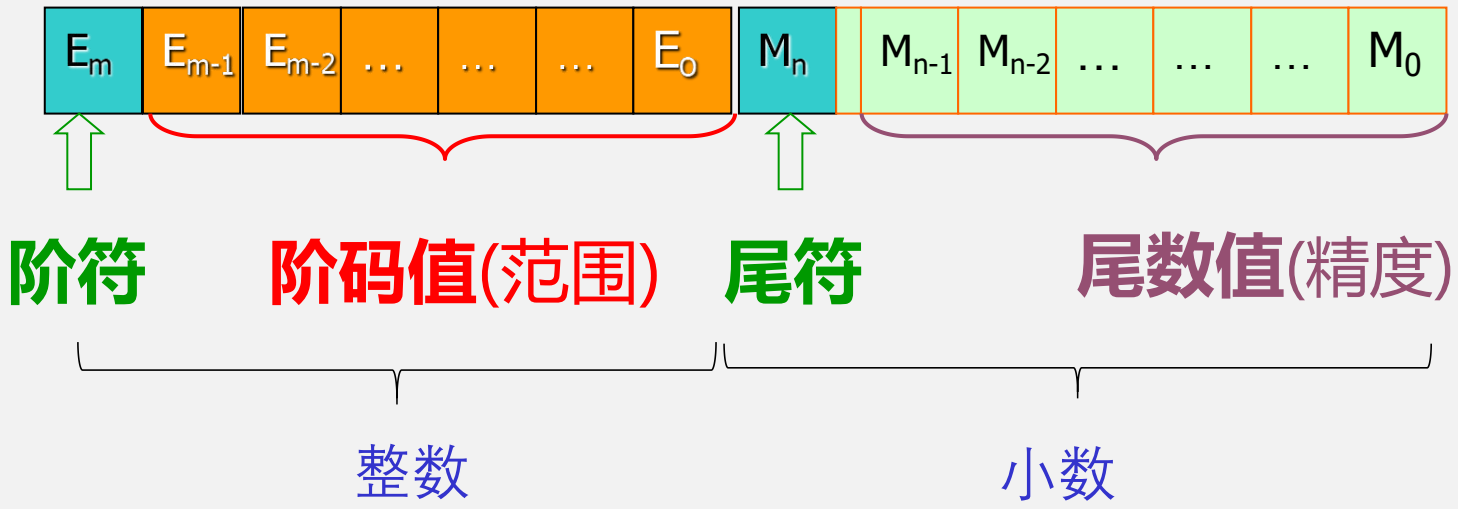
- 科学记数法利用幂和尾数来表示浮点数，浮点数采用阶码和尾数来表示。
- 另外，浮点数可表示纯整数和纯小数以外的实数。

10. 浮点数据表示

2) 计算机内浮点数的一般格式

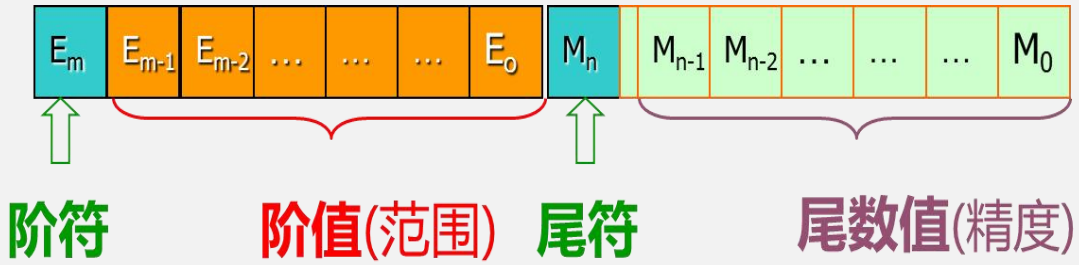
$$N=2^{E}\times M$$

$$N=2^{\pm e}\times(\pm m)$$



10. 浮点数据表示

3) 计算机内浮点数的一般格式特点分析

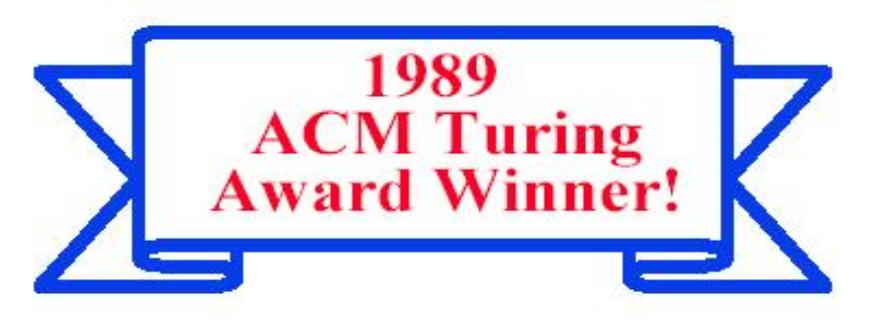


- 机器字长确定的情况下，尾数位越长则阶码越短，如何取舍?
- 早期各计算机公司不通型号计算机，有着千差万别的浮点数表示;
- 数据交换、软件可移植性、计算机协同工作等都受到影响。

10. 浮点数据表示

4) IEEE 754格式

上世纪70年代后期开始，1985年完成浮点数标准IEEE 754制定，至今仍为许多CPU与浮点运算器所采用 。



UC Berkeley math professor
William Kahan.

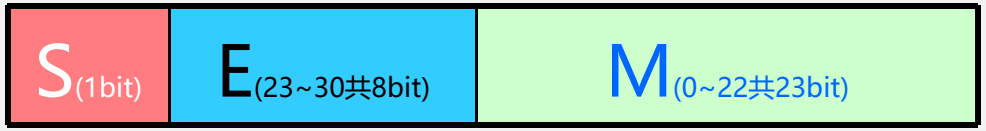
www.cs.berkeley.edu/~wkahan/ieee754status/754story.html

10. 浮点数据表示

What Every Computer Scientist Should Know About Floating-Point Arithmetic
DAVID GOLDBERG。 ACM Computing Surveys, Vol 23, No 1, March 1991

4) IEEE 754格式

32浮点数 *float*



64浮点数 *double*

- 机器数构成：阶码E，尾数M，符号位S.
- $N = (-1)^S \times 1.M \times 2^e$ (规格化尾数，1隐藏)
- 阶码E移码表示，偏移量127/1023，E=e + 127/1023 ;尾数原码表示
- 偏移值127/128：综合考虑表示更大数据范围和表示特殊值。

  e_{min} = -127， e_{max} = 126 (E=128)； e_{min} = -126， e_{max} = 127 (E=127)；

3.4 数值数据表示

符号位	阶码	尾数	表示的数据
0/1	255	1xxxx	NaN Not a Number *
0/1	255	非零 0xxxx	sNaN Signaling NaN **
0	255	0	+∞
1	255	0	-∞
0/1	1~254	M	$(-1)^S \times (1.M) \times 2^{(E-127)}$ (规格化)
0/1	0	M (非零)	$(-1)^S \times (0.M) \times 2^{(-126)}$ (非规格化)
0/1	0	0	+0/-0

** sNaN 是NaN的一种，例如可以用来检查变量是否被初始化

3.4 数值数据表示

返回NaN的运算有如下三种

1)至少有一个参数是NaN的运算

2)不定式

- ♦除法运算：0/0、∞/∞、∞/−∞、−∞/∞、−∞/−∞
- ♦乘法运算：0×∞、0×−∞
- ♦加法运算：∞ + (−∞)、(−∞) + ∞
- ♦减法运算：∞ - ∞、(−∞) - (−∞)

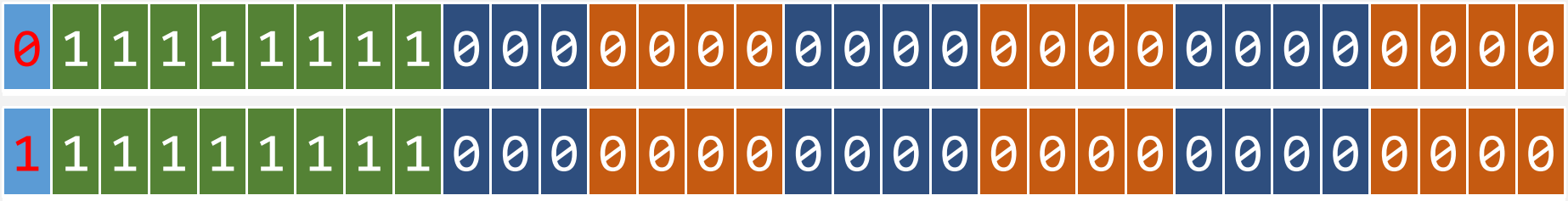
3)产生复数结果的实数运算。例如：

- ♦对负数进行开偶次方的运算
- ♦对负数进行对数运算
- ♦对正弦或余弦到达域以外的数进行反正弦或反余弦运算

3.4 数值数据表示


```
main()
{
  float a=1.0,b=-1.0;
  a=a/0;  b=b/0;
  printf("a=%f b=%f",a,b);
  return;
}
```

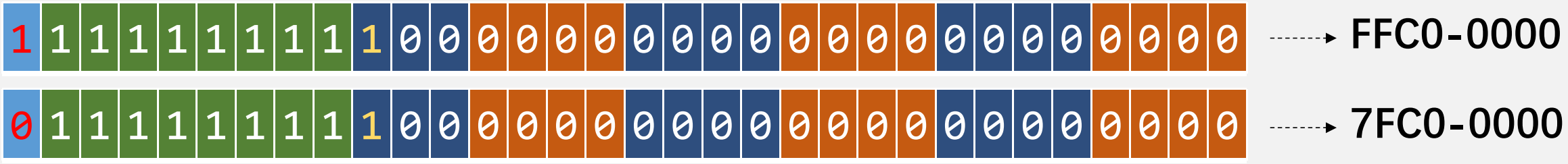
a=1.#INF00 b=-1.#INF00



3.4 数值数据表示

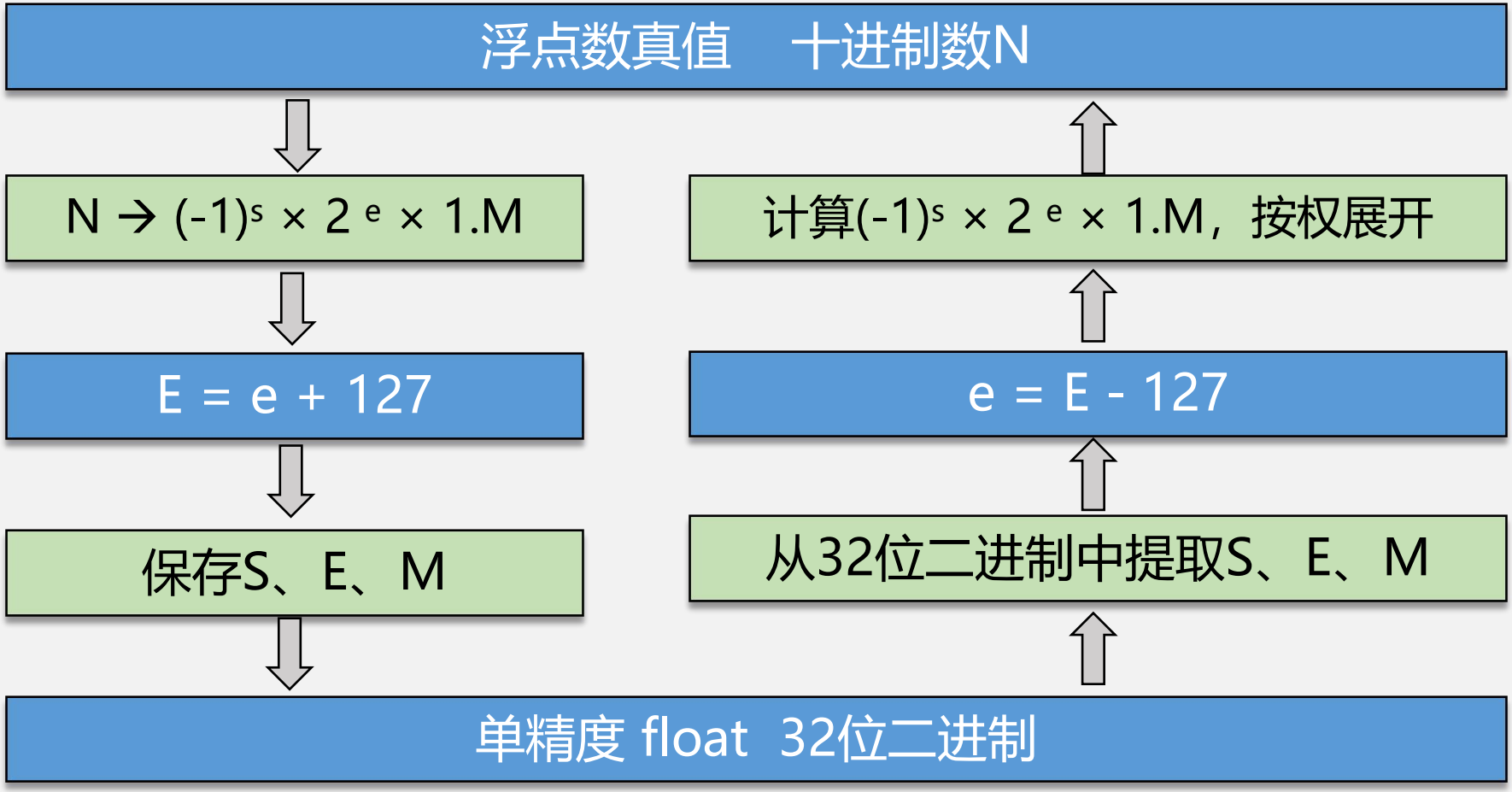

```
main()
{
  float a=0.0, b;
  a=a/0;  b=-sqrt(-1);
  printf ("a=%f b=%f",a,b);
  return;
}
```

a=1.#IND00　b=1.#QNAN0

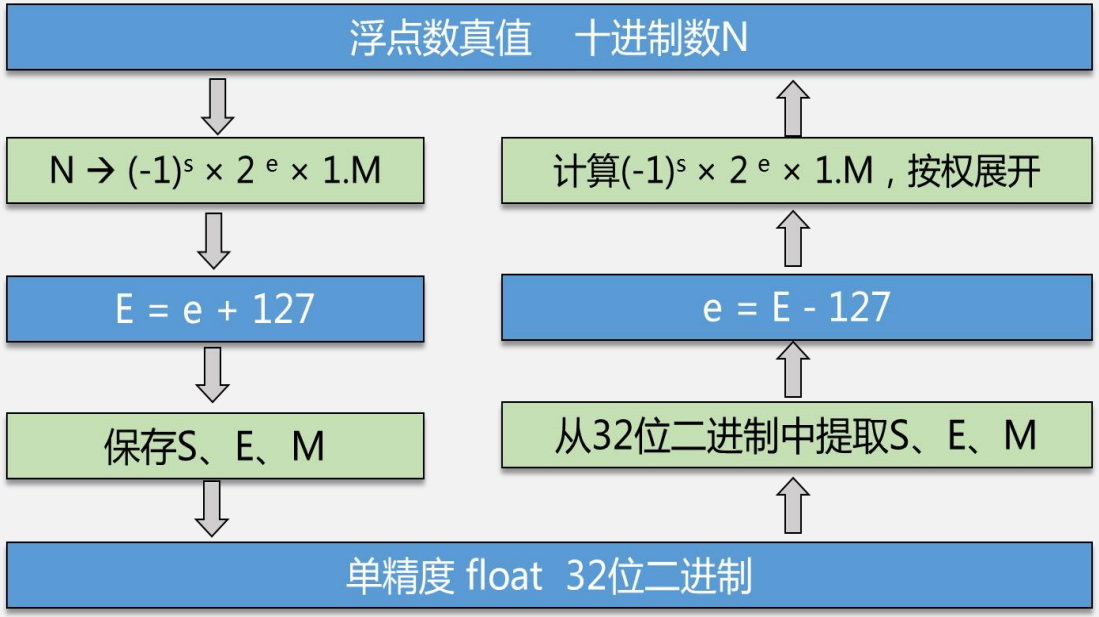


10. 浮点数据表示

5) IEEE 754格式与浮点数真值的转换



10. 浮点数据表示 5) IEEE 754



例 3.3_{10} → IEEE 754 float

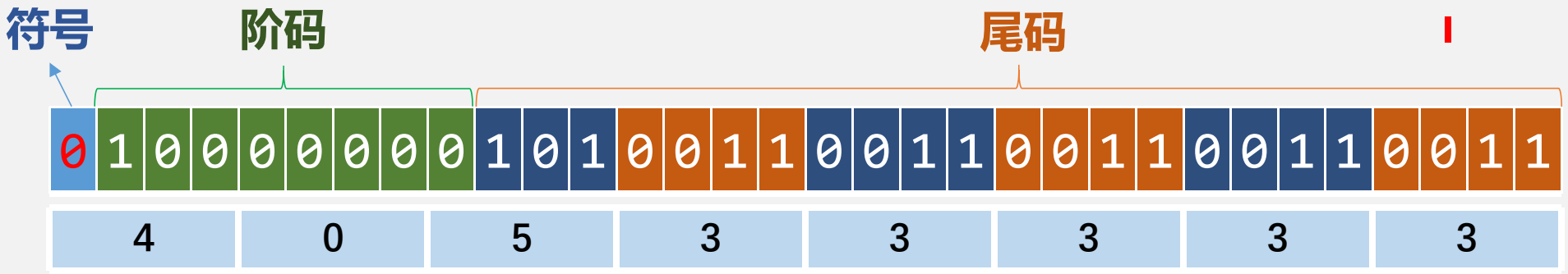
$3.3_{10} = 11.01001100110011001100110011..._2$

$= 1.10100110011001100110011 \times 2^1$

S = 0 , e = 1,

E = 01111111 +1= 100000000

M= 10100110011001100110011



3.4 数值数据表示


```
main()
{
  double a,b,c;   int d;
  b=3.3;   c=1.1;
  a=b/c;
  d=b/c;
  printf("%f,%d",a,d);
  if (3.0!=a)
  printf("\nReally? 3.0!=a");
}
```

```
3.000000,2
```

```
Really? 3.0!=a
```

$(2^{-126}+10^{20})\ -10^{20}=?$　　$2^{-126}+(10^{20}-10^{20})=?$

由于精确度损失问题，浮点运算不满足结合律

3.5 非数值数据

计算机内如何使用二进制表示字符数据。

1. ASCII 码

位 数				w7	0	0	0	0	1	1	1	1
				w6	0	0	1	1	0	0	1	1
				w5	0	1	0	1	0	1	0	1
w4	w3	w2	w1	行 \ 列	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

1) ASCII: American Standard Code for Information Interchange (ANSI 7bits)

2)7bits, MSB=0, 共可表示128个字符

- 52 Letters : a-z, A-Z
- 10 Digits: 0-9
- 34 Symbols :! @ # $ % ^ & * ( ) …
- 32 Control characters : <CR> <BEL> <ESC> <LF>

EBCDIC-Extended Binary-Coded Decimal Interchange Code (IBM 8bits)

Unicode: Universal Multiple-Octet Coded Character Set", 简称为UCS （统一码、万国码、单一码）

3.5 非数值数据

Unicode:

Unicode通常用两个字节表示一个字符，原有的英文编码从单字节变成双字节，只需要把高字节全部填为0就可以。

以满足跨语言、跨平台进行文本转换、处理的要求。1990年开始研发，1994年正式公布。

2. 汉字编 码

1) GB2312简体中文编码表

- GB2312标准共收录6763个汉字(一级，二级)及682个全角字符拉丁字母、希腊字母、日文平假名及片假名字母、俄语字母等;
- 整个字符集分成94个区，每区有94个位;
  - 01-09区为特殊符号
  - 10-15区没有编码
  - 16-55区为一级汉字，按拼音排序，共3755个
  - 56-87区为二级汉字，按部首 / 笔画排序，共3008个
  - 88-94区没有编码

2. 汉字编 码

GB2312汉字编码字符集对照表

第01区	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
A1A0			、	。	·	ˉ	ˇ	¨	〃	々	―	～	‖	…	‘	’
A1B0	“	”	〔	〕	〈	〉	《	》	「	」	『	』	〖	〗	【	】
A1C0	±	×	÷	∶	∧	∨	∑	∏	∪	∩	∈	∷	√	⊥	∥	∠
A1D0	⌒	⊙	∫	∮	≡	≌	≈	∽	∝	≠	≮	≯	≤	≥	∞	∵
A1E0	∴	♂	♀	°	′	″	℃	$	¤	¢	£	‰	§	№	☆	★
A1F0	○	●	◎	◇	◆	□	■	△	▲	※	→	←	↑	↓	〓	
第02区	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
A2A0																
A2B0		1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
A2C0	16.	17.	18.	19.	20.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
A2D0	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	①	②	③	④	⑤	⑥	⑦
A2E0	⑧	⑨	⑩			㈠	㈡	㈢	㈣	㈤	㈥	㈦	㈧	㈨	㈩	
A2F0		Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ	Ⅶ	Ⅷ	Ⅸ	Ⅹ	Ⅺ	Ⅻ			
第03区	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F

3.5 非数值数据

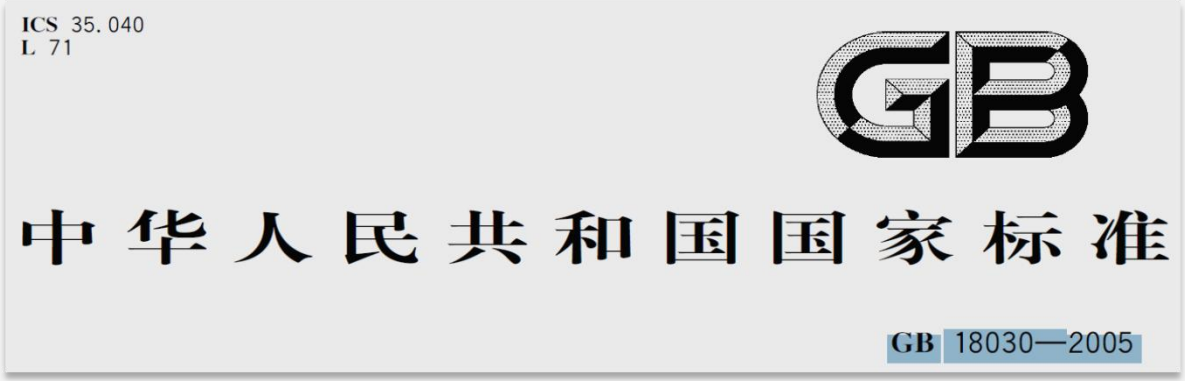
2. 汉字编 码

GB2312汉字编码字符集对照表

区位码　　机内码

A9F0																
第10区	AAA0-AAFF	没有编码														
第11区	ABA0-ABFF	没有编码														
第12区	ACA0-ACFF	没有编码														
第13区	ADA0-ADFF	没有编码														
第14区	AEA0-AEFF	没有编码														
第15区	AFA0-AFFF	没有编码														
第16区	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
B0A0		啊	阿	埃	挨	哎	唉	哀	皑	癌	蔼	矮	艾	碍	爱	隘
B0B0	鞍	氨	安	俺	按	暗	岸	胺	案	肮	昂	盎	凹	敖	熬	翱
B0C0	袄	傲	奥	懊	澳	芭	捌	扒	叭	吧	笆	八	疤	巴	拔	跋
B0D0	靶	把	耙	坝	霸	罢	爸	白	柏	百	摆	佰	败	拜	稗	斑
B0E0	班	搬	扳	般	颁	板	版	扮	拌	伴	瓣	半	办	绊	邦	帮
B0F0	梆	榜	膀	绑	棒	磅	蚌	镑	傍	谤	苞	胞	包	褒	剥	
第17区	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
B1A0		薄	雹	保	堡	饱	宝	抱	报	暴	豹	鲍	爆	杯	碑	悲

3. 汉字编码标准发展



- GB2312-1980(GB0)(简体)

  6763个汉字

- GB13000-1993

  20902个汉字 (Unicode 1.1版本)

- 汉字扩展规范GBK1.0 标准1995（非国家标准）：21003个字符（兼容GB2312）
- Big5-2003

  大五码,13,060个汉字，使用繁体中文社群中最常用的汉字字符集标准

- GB18030-2005 (1/2/4字节编码：可变长编码)

  70,244个汉字，对GB 2312-1980完全向后兼容，与GBK基本向后兼容，并支持Unicode（GB 13000）的所有码位。

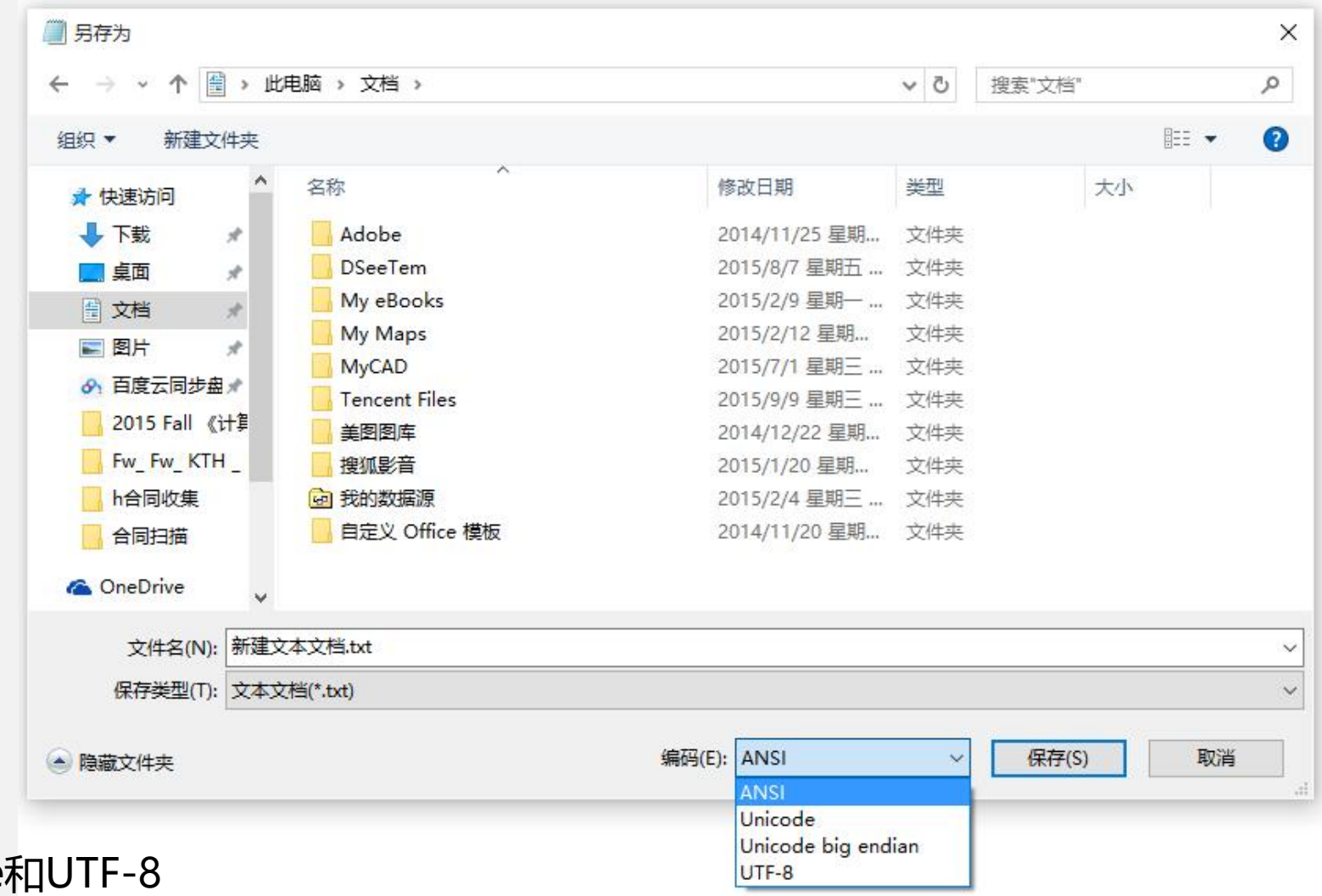
3. 汉字编码标准发展

- <META content="text/html; charset=gb2312"
  … http-equiv=Content-Type>
- charset=gb2312 简体中文
  charset=big5 繁体中文
  charset=EUC_KR 韩语
  charset=Shift_JIS 或 EUC_JP 日语
  charset=KOI8-R/Windows-1251俄语
  charset=iso-8859-2 中欧语系
  charset=utf-8 unicode多语言

1. 打开电脑，点击主菜单，找到并点击“控制面板”选项，之后找到并点击“时钟、语言和区域”选项。
2. 在打开的页面中找到并点击“语言”选项。
3. 点击并打开“添加语言”选项，即可找寻想要添加的语言包。也可以点击右上的搜索框，输入想添加的语言包名称。
4. 输入想添加的语言包并搜索之后，点击该语言包并点击下方的“添加”按钮，语言包便安装...

3. 汉字编码标准发展

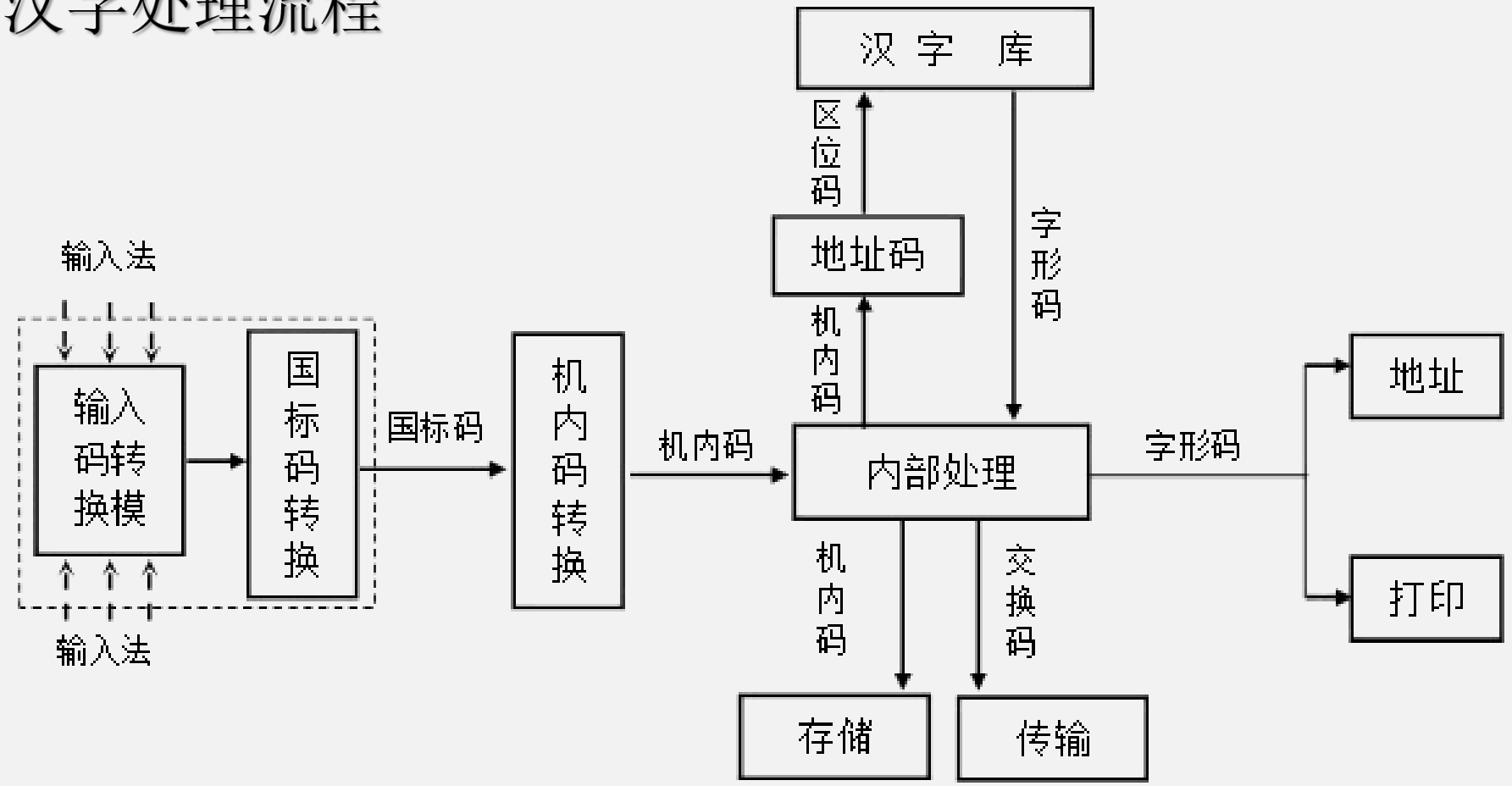
记事本编码区别?



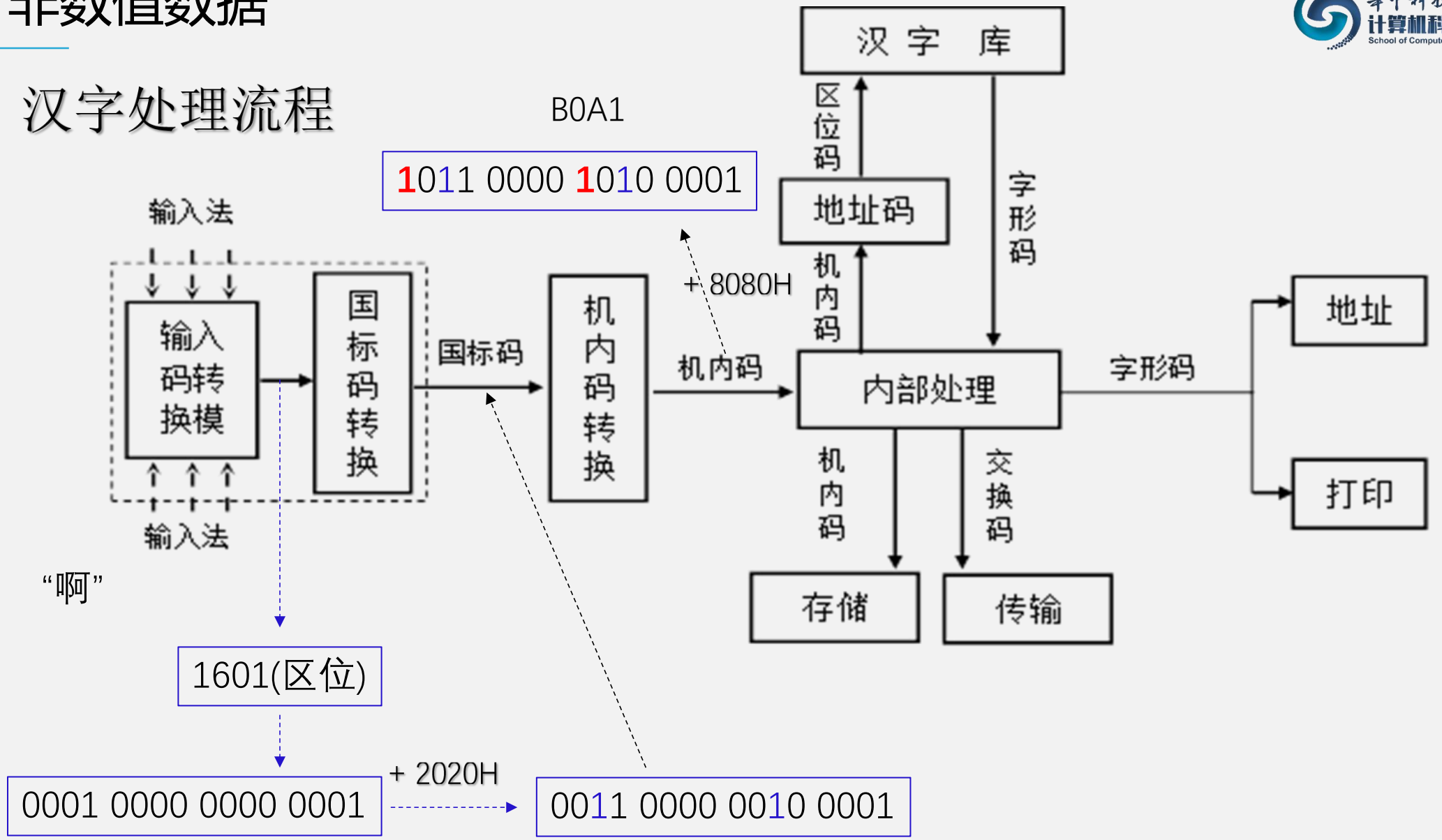
字符编码笔记：ASCII，Unicode和UTF-8

http://www.ruanyifeng.com/blog/2007/10/ascii_unicode_and_utf-8.html

4. 汉字处理流程


汉 字 库
区位码
地址码
字形码
机内码
输入法
输入码转换模
国标码转换
国标码
机内码转换
机内码
内部处理
字形码
地址
打印
机内码
交换码
存储
传输
输入法

4. 汉字处理流程


汉 字 库
区位码
地址码
字形码
B0A1
1011 0000 1010 0001
+ 8080H
机内码
输入法
输入码转换模
国标码转换
国标码
机内码转换
机内码
内部处理
字形码
地址
打印
机内码
交换码
存储
传输
输入法
“啊”
1601(区位)
0001 0000 0000 0001
+ 2020H
0011 0000 0010 0001

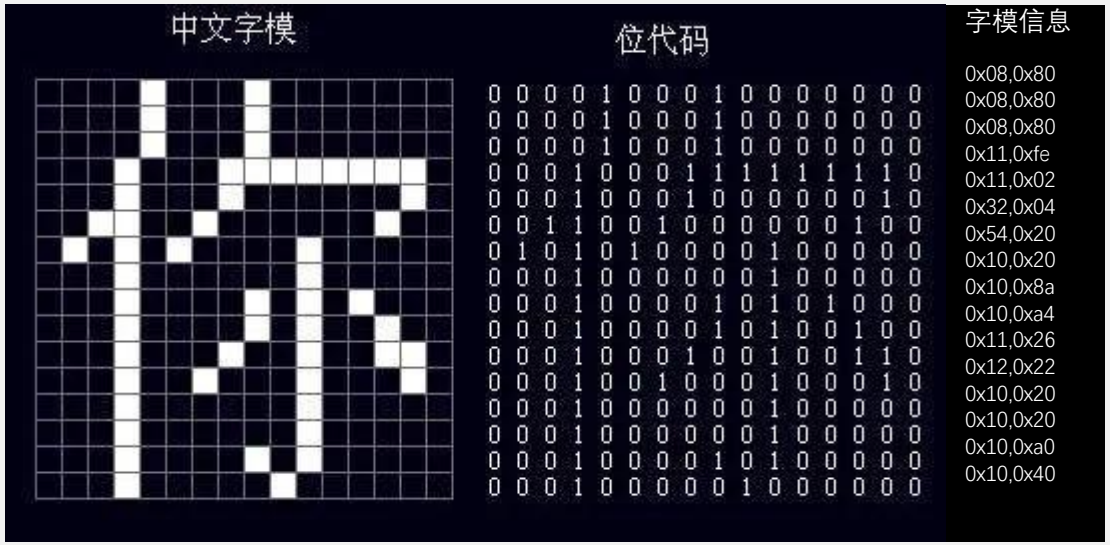
4. 汉字处理流程

位码+ 2020H + 8080H 变成机内码的原因

- GB2312依然保留26个英文字母和一些特殊符号的编码，覆盖ASCII中的相同部分，同时将ASCII中前32个控制字符则继续沿用(即十六进制20H)。因此，区位码要加上20H才能得到国标码。
- 汉字国标码中用2个字节表示一个汉字，且每个字节的高位为0(见上一条以及一、二级汉字总数6763，128×128 = 16384)
- 上述覆盖而非兼容ASCII码的设计方案应用中出现了乱码。原因是两字节表示的1个汉字与2个ASCII字符无法区分。为此，在国标码的基础上，将字节的最高位设为1，因为ASCII中使用7位，最高位为0。这样就区分开了ASCII和GB2312。这是国标码还需加上8080H得到机内码的原因。

4. 汉字处理流程

♦点阵字体(位图字体)把每个字分成M×N个点(16×16、24×24),每个字形都以一组二维像素信息表示。



♦存储1024个24*24点阵的汉字字型码需要的字节数是多少？ 24*3 = 72B

♦位图字体难以缩放，强行放大会导致失真字形，产生 马赛克式的锯齿边缘。

本章结束
再见！