

# 信息表示

王厚峰

wanghf@pku.edu.cn

EECS, PKU

# 内容

➤ **编码与数值**

➤ **符号与文字表示**

➤ **多媒体（模态）表示**

# 一个最基本的问题

- 什么是计算机科学？4种观点
  - 关于计算机的科学（强调硬件与体系结构）
  - 关于软件的（软件设计与开发）
  - 关于数据的（最终的目的是对数据加工）
  - 关于算法的（数据加工的方法）

**关于信息的科学**

IT (Information Technology)

**你认为呢？**

# 计算机系统与构成

- 软件与数据在计算机内的表示：广义的信息表示
  - 程序代码表示
  - 数据表示
    - 数值表示
    - 非数值表示（符号、声音、图像、...）
  - 不同信息的表示在形式上没有区别



# 计算机中的信息表示与编码

- 什么是信息编码
  - 采用约定的符号，按照特定的组合规则，表示日常各种复杂的信息
- 信息编码要考虑三个因素
  - 编码符号集合
  - 外在信息与编码之间的转换关系
  - 内外运算规则的一致性（**数值表示**）
- 计算机上所有信息用**2进制**编码（**0/1** 两个符号）
  - 优点：**1) 正好反映两种稳定的物理态；**
  - 2) 算术/逻辑运算简便**

# 计算机中的信息编码类型

## • 数值编码

- 计算机对数值的编码
- 在编码下对数值的运算与外部的运算一致
  - 二进制的加、减、乘、除等与外界的相应运算一致

## • 典型的非数值编码

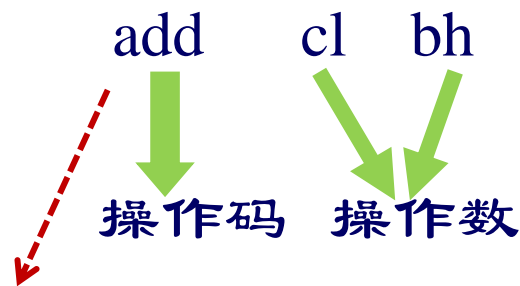
- 符号编码（文字与符号）
- 图像编码
- 声音编码
- 指令编码

视频编码



指令的功能

指令操作的对象



00000010	11001111 (add cl, bh)
00000010	11111100 (add bh, cl)

# 进制：进位制

根据不同的进位原则，可以得到不同的进位制

几种常用的进位制

- 10进制：0, 1, 2, ..., 9; 逢十进一
- 2进制：0, 1; 逢二进一
- 8进制：0, 1, 2, 3, 4, 5, 6, 7; 逢八进一
- 16进制：0, 1, 2, ..., 9, A, B, C, D, E, F; 逢十六进一

# 二进制 $\rightarrow$ 十进制

- 二进制数**B**所表示的值为:

$$B = b_{n-1} b_{n-2} \dots b_1 b_0 . b_{-1} \dots b_{-m}$$

- 按位权展开为:

$$B = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_1 \times 2^1 + b_0 \times 2^0 + b_{-1} \times 2^{-1} + \dots + b_{-m} \times 2^{-m}$$

其中, 2 为基数

- 例如:  $(10110.1)_2$   
 $= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1}$   
 $= (22.5)_{10}$

位权展开的结果为**10**进制数

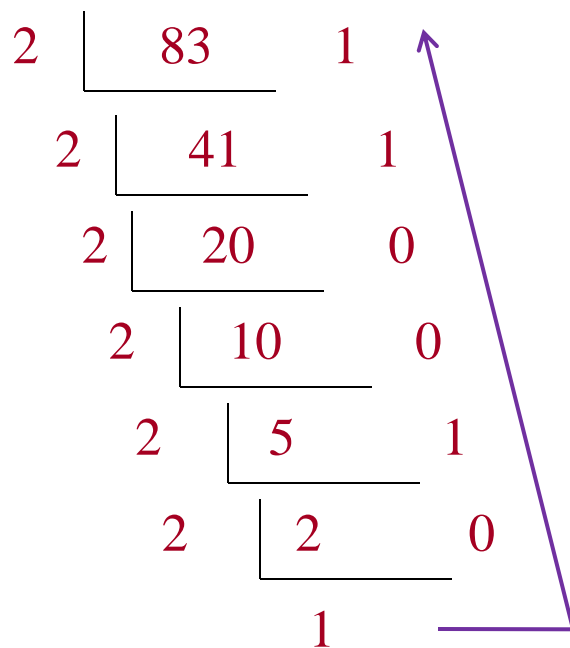


# 十进制 **整数** 转换为二进制

- **逐次除2取余法：**

用2逐次去整除待转换的十进制整数，得到商和余数；如果商大于1，再用2去整除商，得到新的商和余数；如此反复，直至商为0/1时停止。最后，**反向连接余数和商**，即得到对应的2进制值。

例如，将83转换成二进制数，逐次除2取余  
(辗转相除法)：



得到的余数从先至后依次为：

1 ← 1 ← 0 ← 0 ← 1 ← 0 ← 1

**反向连接**，可得到： $(83)_{10} = (1010011)_2$

# 10 — 2进制小数转换

## 乘2取整法：

逐次用2去乘待转换的十进制小数，将每次得到的整数部分（0或1）依次记为二进制小数 $b_{-1}$ ， $b_{-2}$ ， $\dots$ ， $b_{-m}$ 。

例如，将0.8125转换为二进制小数，逐次乘2取整（**顺次连接整数部分**）：

$$\begin{array}{r} 0.8125 \\ \times 2 \\ \hline 1 \quad .625 \\ \times 2 \\ \hline 1 \quad .25 \\ \times 2 \\ \hline 0 \quad .5 \\ \times 2 \\ \hline 1 \quad .0 \end{array}$$


可得：

$$\begin{aligned} & (0.8125)_{10} \\ & = (0.1101)_2 \end{aligned}$$

## 精确度问题：

并非每一个十进制小数都能转换为有限位的二进制小数，出现这种情况时，可以采用0舍1入的方法进行处理（类似于十进制中的四舍五入）。

例，将0.335转换为二进制小数，精确到0.001。


$$\begin{array}{r} 0.335 \\ \times 2 \\ \hline 0.67 \\ \times 2 \\ \hline 1.34 \\ \times 2 \\ \hline 0.68 \\ \times 2 \\ \hline 1.536 \end{array}$$

可得：  $(0.335)_{10} = (0.0101\dots)_2 \approx (0.011)_2$

# 任意进制转换

- 其它进制与十进制转换方式类似，只是基数变成相应进制；
- 任意进制之间的转换可以以十进制为中间进制转换，以  $p$ 进制 $\rightarrow$  $q$ 进制为例
  - ①  $p$ 进制 $\rightarrow$ 十进制
  - ② 十进制 $\rightarrow$  $q$ 进制

# 计算机内的数值表示

数值在计算机内的表示

十进制	二进制	八进制	十六进制
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	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



# 二进制数的算术运算

加法:  $0+0=0, 0+1=1, 1+1=10$ (进位)

$$\begin{array}{r} \text{例: } (1101)_2 \dots\dots(13)_{10} \\ + (1011)_2 \dots\dots(11)_{10} \\ \hline (11000)_2 \dots\dots(24)_{10} \end{array}$$

减法:  $0-0=0, 1-1=0, 0-1=1$ (借位)

$$\begin{array}{r} \text{例: } (1101)_2 \dots\dots(13)_{10} \\ - (1011)_2 \dots\dots(11)_{10} \\ \hline (0010)_2 \dots\dots(2)_{10} \end{array}$$

乘法:  $0 \times 0=0, 0 \times 1=0, 1 \times 1=1$

$$\begin{array}{r} \text{例: } (1101)_2 \dots\dots(13)_{10} \\ \times (1011)_2 \dots\dots(11)_{10} \\ \hline 1101 \\ 1101 \\ 0000 \\ 1101 \\ \hline (1000111)_2 \dots\dots(143)_{10} \end{array}$$

除法:  $0 \div 0=0, 1 \div 1=1, 0 \div 1=1$

$1 \div 0$ 无意义

$$\begin{array}{r} \text{例: } \quad \quad \quad 000110\dots \text{商} \\ \text{除数} \dots 110 \overline{) 100110\dots \text{被除数}} \\ \quad \quad \quad \underline{110} \\ \quad \quad \quad 0111 \\ \quad \quad \quad \underline{110} \\ \quad \quad \quad 10\dots \text{余数} \end{array}$$

# 二进制数的逻辑运算

逻辑非	!	单操作数运算:	$!1 = 0$ ; $!0 = 1$
逻辑或		双操作数运算:	$0 0 = 0$ ; $0 1 = 1$ $1 0 = 1$ ; $1 1 = 1$
逻辑与	^	双操作数运算:	$0^0 = 0$ ; $0^1 = 0$ $1^0 = 0$ ; $1^1 = 1$

$$!11010 = 00101$$
$$\begin{array}{r} 11001 \\ | \underline{01101} \\ 11101 \end{array}$$
$$\begin{array}{r} 11001 \\ -^ \underline{01101} \\ 01001 \end{array}$$

# 计算机内的数值： 正数与负数

- 约定：

正负符号用最高位（最左位）表示：

0：表示正数的符号

1：表示负数的符号

例如：

$$(01011)_2 = (+11)_{10}$$

$$(11011)_2 = (-11)_{10}$$

# 原码、补码、反码

- **原码**：最高位为符号位，其余各位为数值本身的绝对值(前面例子)
- **反码**：
  - 正数：反码与原码相同
  - 负数：符号位为1，其余位对原码取反
- **补码**：
  - 正数：原码、反码、补码相同
  - 负数：最高位为1，其余位为原码取反，再对末位数加1

**例**：用一个字节表示 **-46** 的补码（注意计算步骤）：

绝对值的原码：        0 0 1 0    1 1 1 0

实际原码：            1 0 1 0    1 1 1 0

反码：                 1 1 0 1    0 0 0 1

补码：                 1 1 0 1    0 0 1 0

于是： $[-46]_{\text{补}} = [1101\ 0010]_2 = [D2]_{16}$

# 原码表示

- 在给定码长后，根据一个**整数**的正负填写符号位，再将这个整数之**绝对值**的二进制表示，按照数值位的长度在前面补足必要的0后，就得到这个整数的**原码表示**。

若码长为 8位，则  $123_{(10)}$  的原码表示是：

**0**1111011

$-123_{(10)}$  的原码表示是：

**1**1111011

若码长为 16位，则  $123_{(10)}$  的原码表示是：

**0**000000001111011

$-123_{(10)}$  的原码表示是：

**1**000000001111011

**0**有2种表示：  
**+0**和**-0**，其表示不同

# 反码表示

- 规定：
  - 一个正整数的反码与其原码的表示相同；
  - 一个负整数的反码表示：对其原码表示的数值位进行按位变反（按位将 1 换成 0、将 0 换成 1）的结果。

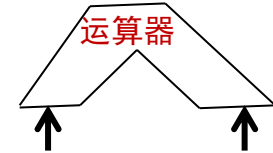
0也有2种表示：  
+0和-0，其表示不同

- 例如（若码长为 8）：

$$(26)_{(反)} = (26)_{(原)} = 0\ 0011010$$

$$(-26)_{(反)} = 11100101 \text{ (连符号位各位取反)}$$

# 算术运算与补码表示



- 二进制减法， $a - b = a + (-b)$
- 由于乘法可以用加法实现、除法可以用减法实现，如果将减法用加法实现，就可以用**统一的部件（加法器）**来进行二进制数的四则运算。
- 支持这种统一处理的基础，是计算机中数的**补码表示**。
- 减去一个数等于加上该数相反数的补码。

# 补码表示

- 码长的二进制表示:

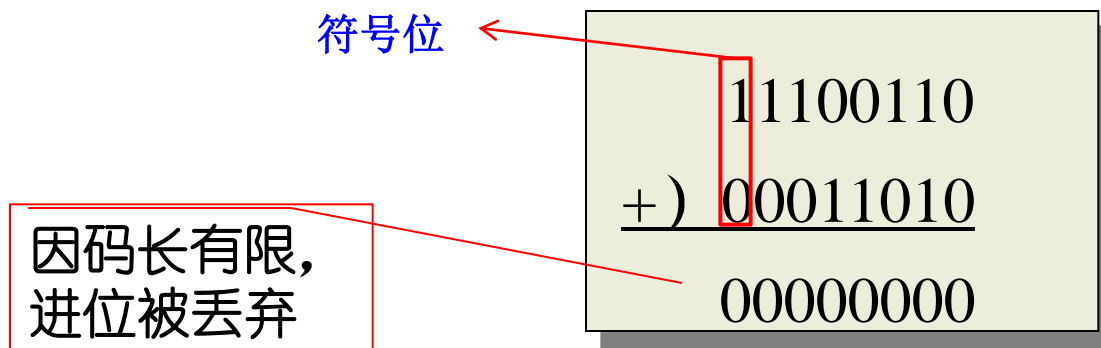
例如, 当码长为 8 (即数值位数为 7), 则

$$26_{(10)} = 0011010$$

那么, 要得到  $-26_{(10)}$ , 就是求一个二进制数  $c$ : 使得:

$$c + 0011010 = 0000000 \quad (\text{满足减法变加法的规律})$$

这样的  $c$  就是  $(-26_{(10)})$  的二进制表示: 1100110





# 补码表示

- 规定：
  - 一个正整数的补码与它的原码表示相同；
  - 一个负整数的补码：符号位为 1，数值位是其绝对值的求补结果（各位求反、末位加 1）。
- 对于一个负整数，怎样求它的补码表示？
  - 一条简单规则：对其原码表示的数值位按位变反后末位加 1。
  - 例：当码长为 8，求  $-26_{(10)}$  的补码表示 (11100110)：
    - 原码表示是：10011010
    - 按位变反后：11100101
    - 加 1 后得到：11100110，即得到其补码表示。

# 补码运算规则

- 可以证明，对于数  $X$  和  $Y$ ， $(X+Y)_{(补)} = X_{(补)} + Y_{(补)}$ ，  
 $(X-Y)_{(补)} = X_{(补)} + (-Y)_{(补)}$

- 两个数相加减，只需进行包括符号位在内的补码相加：

$$(-27)_{(补)} = 11100101 \quad (10011011 \rightarrow 11100100 \rightarrow 11100101)$$

$$(-1)_{(补)} = 11111111 \quad (10000001 \rightarrow 11111110 \rightarrow 11111111)$$

$$(-26)_{(补)} = 11100110 \quad (10011010 \rightarrow 11100101 \rightarrow 11100110)$$

$$(-25)_{(补)} = 11100111 \quad (10011001 \rightarrow 11100110 \rightarrow 11100111)$$

$26 - 27 = -1$ $\begin{array}{r} 00011010 \\ +) 11100101 \\ \hline 11111111 \end{array}$	$26 - 26 = 0$ $\begin{array}{r} 00011010 \\ +) 11100110 \\ \hline 00000000 \end{array}$	$26 - 25 = 1$ $\begin{array}{r} 00011010 \\ +) 11100111 \\ \hline 00000001 \end{array}$	减 法 变 加 法

# 几组特殊数值

(用一字节表示数)

	原码	反码	补码
+7	00000111	00000111	00000111
-7	10000111	11111000	11111001
+0	00000000	00000000	00000000
-0	10000000	11111111	00000000
数的范围	01111111~ 11111111 (-127~+127)	01111111~ 10000000 (-127~+127)	01111111~ 10000000 (-128~+127)

# 补码的进一步说明

任意负数  $X$  的补码表示满足如下规律：

$[X]_{\text{补}} = 2^n - |X|$ ，于是，对于1个字节（ $n=8$ ），有，

$$[-1]_{\text{补}} = 2^8 - |-1| = 11111111$$

$$[-127]_{\text{补}} = 2^8 - |-127| = 100000000 - 01111111 = 10000001$$

注意：  $[0]_{\text{补}} = [-0]_{\text{补}} = 00000000$

对于10000000，在补码中定义为-128：

$$[-128]_{\text{补}} = 2^8 - |-128| = 2^8 - 2^7 = 100000000 - 10000000 = 10000000, \\ \text{高位为1。}$$

# 内容

➤ 编码与数值

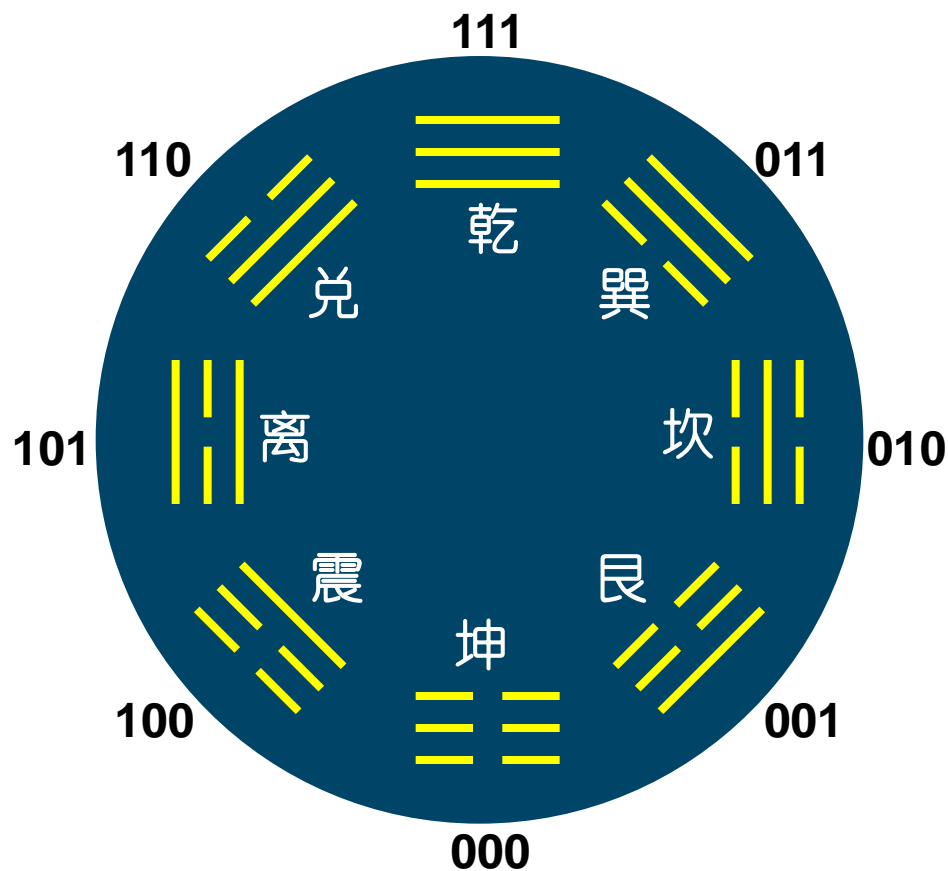
➤ 符号与文字表示

➤ 多媒体（模态）表示

# 文字符号编码的特点

- 已知集合大小
  - 需要编码表示的符号集相对固定
  - 根据符号集大小确定编码的位数

# 符号的二进制编码



8种不同符号，需要  
3位2进制符号编码：

$$2^3=8$$

问题：128种不同符号，用2进制编码至少需要多少位？

# 英文字符等编码：ASCII码

- ASCII码
  - 美国信息交换标准码 (American Standard Code for Information Interchange)
  - 最初由美国制订，后来由国际标准组织 (ISO) 确定为国际标准字符编码，被广泛使用
- ASCII码的编码规则
  - 每个字符用7位二进制值( $d_6d_5d_4d_3d_2d_1d_0$ )表示；可表示128个字符 (7位二进制共有128种状态/符号,  $2^7 = 128$ )
  - 每个ASCII码用一个字节表示，其最高位 $d_7$ 用0填充；后7位为实际编码值
  - 在计算机中，每个ASCII码用一个字节存储




# 编码的区域划分

- 33个控制码(第0~31号)及第127号;如: LF (换行)、CR (回车)、DEL (删除)、BEL (振铃)等;
- 1个空格码;
- 94个字符码,如, 阿拉伯数字, 英文大小写, 运算符, 标点符号等;

# ASCII中的控制符（不可显示）

ASCII 碼					ASCII 碼				
十進位	十六進位	字元	控制字元	意義	十進位	十六進位	字元	控制字元	意義
000	00		NULL	空字元	016	10	▶	DLE	
001	01	☺	SOH		017	11	◀	DC1	
002	02	☺	STX		018	12	↕	DC2	
003	03	♥	ETX		019	13	❗	DC3	
004	04	♦	EOT		020	14	⏏	DC4	
005	05	♣	ENQ		021	15	§	NAK	
006	06	♠	ACK		022	16	—	SYN	
007	07	•	BELL	鈴聲	023	17	↕	ETB	
008	08	◻	BS	倒退鍵	024	18	↑	CAN	
009	09		HT	定位鍵	025	19	↓	EM	
010	0A		LF	line feed	026	1A	→	SUB	檔案結束
011	0B	♂	VT	home	027	1B	←	ESC	escape
012	0C	♀	FF	form feed	028	1C	L	FS	向右游標
013	0D		CR	carriage return	029	1D	↔	GS	向左游標
014	0E	♪	SO		030	1E	▲	RS	向上游標
015	0F	☀	SI		031	1F	▼	US	向下游標

# ASCII中的可显示符

ASCII 码		字符	ASCII 码		字符	ASCII 码		字符	ASCII 码		字符
十进位	十六进位		十进位	十六进位		十进位	十六进位		十进位	十六进位	
032	20		056	38	8	080	50	P	104	68	h
033	21	!	057	39	9	081	51	Q	105	69	i
034	22	"	058	3A	:	082	52	R	106	6A	j
035	23	#	059	3B	;	083	53	S	107	6B	k
036	24	\$	060	3C	<	084	54	T	108	6C	l
037	25	%	061	3D	=	085	55	U	109	6D	m
038	26	&	062	3E	>	086	56	V	110	6E	n
039	27	'	063	3F	?	087	57	W	111	6F	o
040	28	(	064	40	@	088	58	X	112	70	p
041	29	)	065	41	A	089	59	Y	113	71	q
042	2A	*	066	42	B	090	5A	Z	114	72	r
043	2B	+	067	43	C	091	5B	[	115	73	s
044	2C	,	068	44	D	092	5C	\	116	74	t
045	2D	-	069	45	E	093	5D	]	117	75	u
046	2E	.	070	46	F	094	5E	^	118	76	v
047	2F	/	071	47	G	095	5F	_	119	77	w
048	30	0	072	48	H	096	60	`	120	78	x
049	31	1	073	49	I	097	61	a	121	79	y
050	32	2	074	4A	J	098	62	b	122	7A	z
051	33	3	075	4B	K	099	63	c	123	7B	{
052	34	4	076	4C	L	100	64	d	124	7C	
053	35	5	077	4D	M	101	65	e	125	7D	}
054	36	6	078	4E	N	102	66	f	126	7E	~
055	37	7	079	4F	O	103	67	g	127	7F	 DEL

由 20H 到 7FH 共 96 个，这 95 个字符是用来表示阿拉伯数字、英文字母大小写和底线、括号等符号，都可显示

# ASCII码所有符号

L \ H	0000	0001	0010	0011	0100	0101	0110	0111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	—	—	·	·
0011	ETX	DC3	#	3				
0100	EOT	DC4	\$	4	D	I	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	,	7	G	W	g	w
1000	BS	CAN	)	8	H	X	h	x
1001	HT	EM	(	9	I	Y	:	—
1010	LF	SUB	*	:	J	Z	x: ?	—
1011	VT	ESC	+	;	K		^	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

A:  $(01000001)_2 / (65)_{10}$

ASCII码

# 汉字编码

- 典型的汉字编码
  - **GB2312/GBK**
    - GBK是对GB2312的扩展，同时包含了BIG5以及日文韩文的汉字，从Win95后全面支持GBK
  - BIG5（港台使用繁体字）
  - ISO/IEC 10646
  - **Unicode**

# 国标：GB2312-80

- 国标2312：中华人民共和国国家标准信息交换用汉字编码；1980年发布，在大陆及海外使用简体中文的地区（如新加坡等）是强制使用的唯一中文编码
- 含6763个简体汉字和682个非汉字图形符号（如，外文字母、数字和符号）
- 2312是我国的标准序号

# 汉字的分级规则

- GB2312-80编码标准收简体汉字**6763**个，分为二级：
  - 第一级3755个，位于16至55区，55区的最后5个字符没有定义
  - 第二级3008个，位于56至87区
- 第一级汉字为常用字，按照汉语拼音字母顺序排列，同音字以笔形顺序横（一）、直（丨）、撇（丿）、点（丶）、折（乙）为序。起笔相同按第二笔，依次类推
- 第二级汉字按部首排序，本标准采用的部首与一般字典用的部首基本相同。部首次序及同部首字按笔划数排列

# 汉字偏旁部首小知识

- 偏旁与部首：偏旁表“形”，部首表“义”
  - 偏旁：构成汉字字形的基本构建，如，“晴”，由“日”和“青”两个偏旁构成
  - 部首：是能够表义的偏旁，如，“晴”由“日”表义，表示“晴”与“日”（太阳）相关
  - 部首是偏旁的一个子集，汉字中，偏旁有500多个，部首200多个
- 部首对汉字的释义
  - 在自然语言处理（NLP）中，可以用200多个部首对汉字进行基本释义，作为字（词）义分析的基础



# GB2312-80的编码特点

- 二字节编码，每字节用低七位表示，高位为 1；
- 第一个字节为“高字节”，表示区，共94个区；
- 第二个字节为“低字节”，表示位，每个区94个位；
- 两个字节的分别称为**区号和位号**，各加上32 ( $= 2^5$ ) (20H)，称为**国标码**。
- 94个区的划分如下：
  - 1—09区：表示西文字母、数字、图形符号
  - 10-15区：预留未使用
  - 16-87区：编排汉字**
  - 88—94区：预留使用，用户可以自定义

# 区位举例（第1级汉字）

54区	1	2	3	4	5	6	7	8	9
0	幀	症	郑	证	芝	枝	支	吱	蜘
1	知	肢	脂	汁	之	织	职	直	植
2	执	值	侄	址	指	止	趾	只	旨
3	志	摯	擲	至	致	置	帜	峙	制
4	秩	稚	质	炙	痔	滞	治	室	中
5	忠	钟	衷	终	种	肿	重	仲	众
6	周	州	洲	诒	粥	轴	肘	帚	咒
7	宙	昼	骤	珠	株	蛛	朱	猪	诸
8	逐	竹	烛	煮	拄	瞩	嘱	主	著
9	助	蛀	贮	铸	筑				

55区	1	2	3	4	5	6	7	8	9
0	住	注	祝	驻	抓	爪	拽	专	砖
1	转	撰	赚	篆	桩	庄	装	妆	撞
2	状	椎	锥	追	赘	坠	缀	諄	准
3	拙	卓	桌	琢	茁	酌	啄	着	灼
4	兹	咨	资	姿	滋	淄	孜	紫	仔
5	滓	子	自	渍	字	鬚	棕	踪	宗
6	总	纵	邹	走	奏	揍	租	足	卒
7	祖	诅	阻	组	钻	纂	嘴	醉	最
8	尊	遵	昨	左	佐	柞	做	作	坐
9									

“中”：位于54区，48位

每区最多94位有汉字，第0位为空，95~99位为空

# 区位举例（第2级汉字）

56区	1	2	3	4	5	6	7	8	9
0	亍	丌	兀	丐	廿	卅	丕	亘	丞
1	鬲	孛	罍	丨	禺	丿	匕	乇	夭
2	卮	氏	凶	胤	馗	毓	辇	叢	、
3	鼎	乚	乚	亍	聿	嗇	嘏	仄	厓
4	厝	厝	厥	廝	厝	厝	匚	叵	匱
5	匾	贖	卦	卣	刂	刂	刂	刂	刂
6	剗	刺	剗	剗	剗	剗	剗	剗	剗
7	劓	冂	冂	冂	冂	冂	冂	冂	冂
8	仞	仞	仞	仞	仞	仞	仞	仞	仞
9	佞	佞	佞	佞	佞				

57区	1	2	3	4	5	6	7	8	9
0	佟	佗	佗	伽	佶	佻	侑	侑	侃
1	侏	侑	佻	侑	侑	侑	侑	侑	侑
2	侏	侑	佻	侑	侑	侑	侑	侑	侑
3	倬	倬	倬	倬	倬	倬	倬	倬	倬
4	偃	偃	偃	偃	偃	偃	偃	偃	偃
5	僖	僖	僖	僖	僖	僖	僖	僖	僖
6	余	余	余	余	余	余	余	余	余
7	輾	輾	輾	輾	輾	輾	輾	輾	輾
8	兗	毫	袞	袞	袞	袞	袞	袞	袞
9	羸	彳	互	洌	洗				

2级汉字很少使用

# 区位码、国标码及内码的关系

- 国标码：区号（高字节）与位号（低字节）各自加上32（=  $2^5$ ）变成**国标码**
- 内码：国标码两个字节的最高位变为1，内码是汉字在计算机中的存储形式
  - 避免和ASCII码（高位为0）发生冲突



# 汉字输入码

## 汉字的输入码（外码）

是指从键盘上输入汉字时使用的编码，它与汉字的内码相对应。主要有三类：数字编码、拼音编码和字型编码。

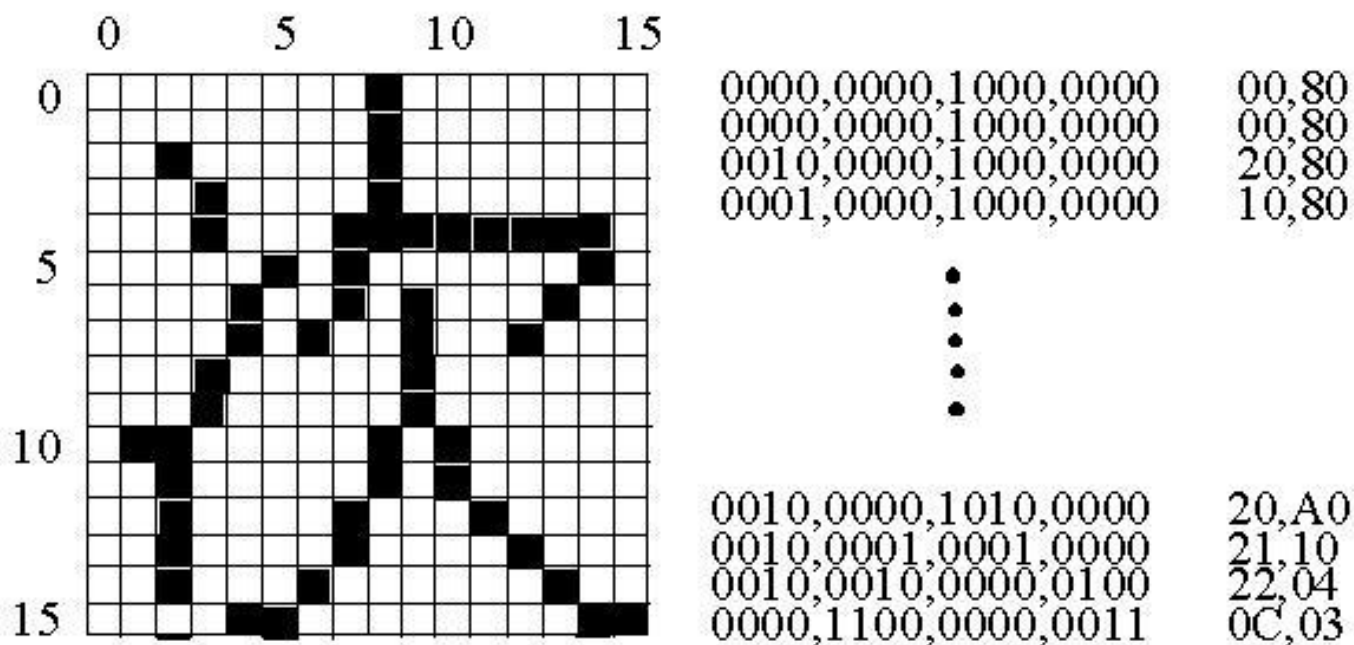
**数字编码**是用一串数字代表一个汉字。最常用的是区位码。它是把国标码的每一个字节减去00100000(20H)得到的。高字节为区码，低字节为位码。无重码，难记忆。

**音码**是以汉字读音为基础的输入方法。简单，重码率高，输入速度较慢。微软拼音输入法、搜狗拼音输入法等。

**形码**是根据汉字形状确定的编码。因为构成汉字的部件是一定的，所以对这些部件进行编码，按书写顺序依次输入，就能表示一个汉字。如，五笔字形码。

# 汉字输出：字形码

- 字形码用在汉字输出时产生汉字字形。有两种显示字形的方法：矢量字符和位图(Bitmap)字符（0-1点阵）。



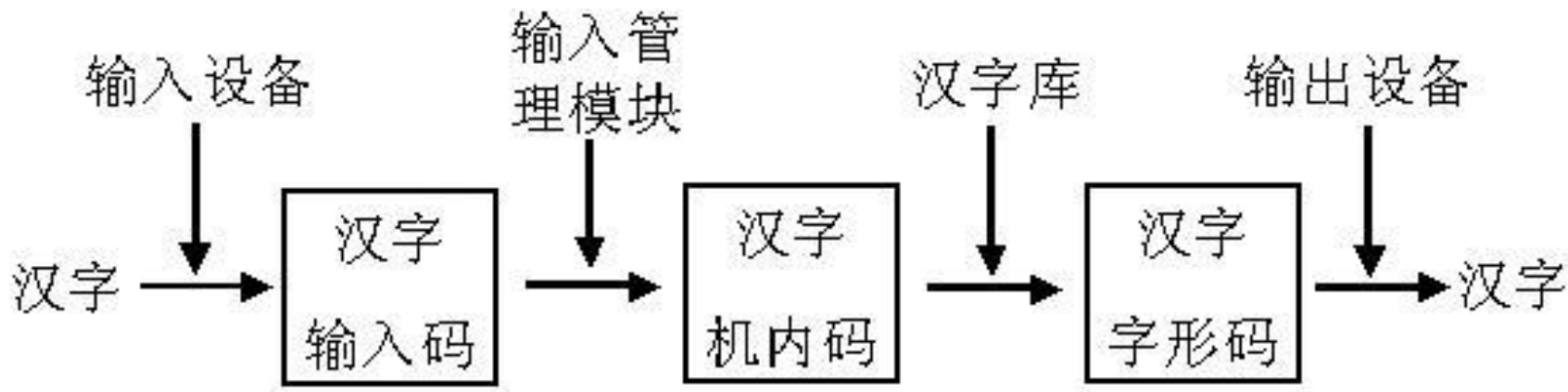
# 矢量字符与位图字符

- 矢量字符：可以缩放，它们由指令表示。优点：
  - 存储空间小：
  - 可以方便修改：可以将某个部分作为一个单独对象，进行编辑。
- 位图字符：计算机指定每个独立的点表示是否有笔画经过，也称 0-1 点阵。如前面表示的字符。

# 汉字输入、存储、输出过程

## 汉字库

字库中存储汉字字型（字形码），供显示和打印输出时使用。字库可以是固化在只读存储器芯片上的汉卡，也可以是以文件形式存储在外存上的软字库。





# 汉字举例——“啊 大学生!”

1. 键盘输入码 a da xue sheng ! (拼音输入码)
2. 机器内部存储码, 十六进制表示  
B0A1**20**B4F3D1A7C9FA**21**
3. 机器内部存储码, 二进制表示  
1011000010100001 (B0A1)  
0010000010110100 (**20**B4)  
1111001111010001 (F3D1)  
1010011111001001 (A7C9)  
1111101000100001 (FA21)
4. 输出显示码: 矢量格式 或 0-1点阵格式

汉字错位/半个汉字

# Unicode编码简介

- 符号编码面临的问题
  - 不同的符号编码系统可能有相同的2进制表示形式
  - 当输入端与接收端编码方案不一样时，会出现奇怪符号
- Unicode用于解决上述问题
  - 统一编码表示不同符号：不同语言在统一体系下编码
  - Unicode 是一个大集合，现在的规模有100多万个符号
  - 每个符号的编码都不一样
- Unicode 由Xerox、Apple等于1988年组成的联盟提出并开发的**万国语言**编码标准，1994年正式公布

# Unicode的编码长度

- Unicode面临的问题
  - 如用统一的多字节（4字节）表示，会造成大量的浪费
    - 英语符号（ASCII）只需1个字节足够了
    - 用多个字节扩展ASCII将使高位扩展为0，导致大量浪费
  - 长度问题导致Unicode编码很长时间无法推广使用
- Unicode的两种典型实现方案
  - **UTF-8**（变长字节：1~4字节）
  - **UTF-16**（介于变长与定长之间）
  - 也有**UTF-32**（定长4字节）

**UTF**: Unicode Transformation Format的缩写，意为Unicode转换格式

# UTF-8编码

- 特点：直接  $n$  变长的编码（**1~4**个字节）
  - 如果是1个字节编码，则首位为0，后7位为 Unicode码
    - 与 ASCII保持一致
  - 当用 $n$  ( $>1$ ) 个字节编码时：
    - 第1个字节的前 $n$ 位为1，第 $(n+1)$ 位为0；
    - 后面字节的前**2位**均为**10**
    - 剩下的没有提及的二进制位全为这个符号的 Unicode 码

1字节： 0xxxxxxx

2字节： **110**xxxxx **10**xxxxxx

3字节： **1110**xxxx **10**xxxxxx **10**xxxxxx

4字节： **11110**xxx **10**xxxxxx **10**xxxxxx **10**xxxxxx

**UTF-16**同学们自己在网上查阅资料

# 内容

➤ 编码与数值

➤ 符号与文字表示

➤ **多媒体（模态）表示**

# 多媒体

**定义：**利用计算机技术，把多种媒体的信息结合在一起，建立起其间逻辑上的联系，并对它们进行各种处理。典型的媒体包括：

- ❑ 文本
- ❑ 图形：用几何形状表述的物体表象
- ❑ 图像：一个区域内带有属性的象素点集合
- ❑ 声音
- ❑ 影像：带有声音的连续图像信号的统称
- ❑ 动画：数字化了的动态图像，关键是定义轨迹、关键帧或动画语言

# 多媒体的特点

- **多样性**

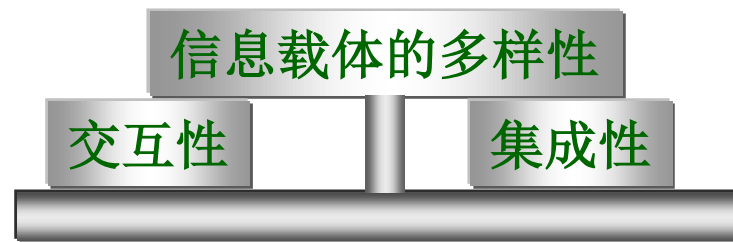
文字，数字，图像、音频等

- **交互性**

用户可以主动控制计算机，增加对信息的注意力

- **集成性**

信息的集成，操作/开发环境的集成

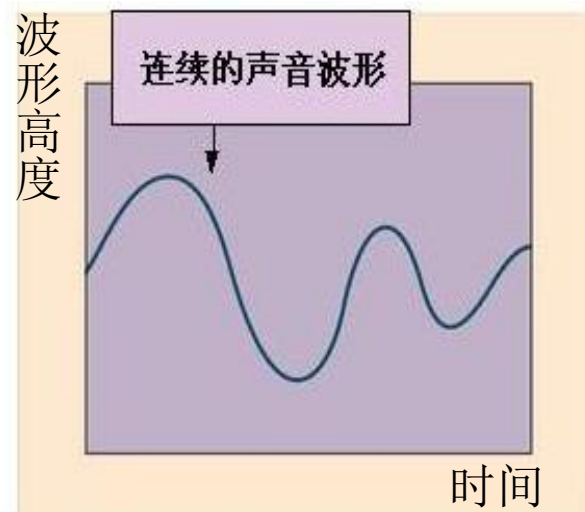


# 声音编码：连续现象的离散化

## 声音编码：一种连续现象的编码

计算机内部表示信息只能用二进制位表示，这就需要将连续信号**离散化采样**，最终用有限位数的二进制数来描述它们。

所谓离散化采样就是把连续信号划分为离散的区段，把每一个区段内这些相近的现象看作一种现象，并用一个特定的二进制数来表示。

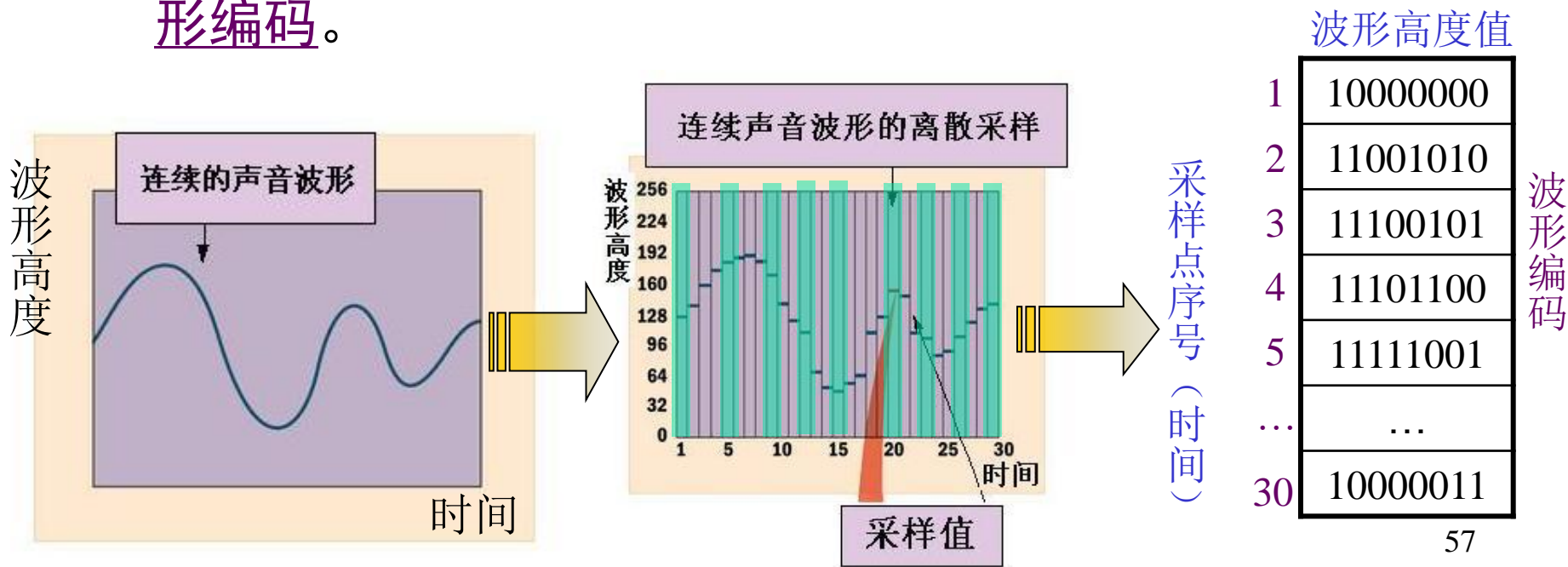


随时间连续变化的波  
(波峰/波谷)



# 声音编码：声音离散采样与编码

在时间和波形高度两个维度上独立采样。在时间维上进行时间的离散化（按一定的均匀时间间隔采样）；同时，每一个采样点的波形高度值（声波的振幅）也要离散化，记录为若干个二进制位的整数编码。两者合在一起形成了声音波形的编码。这种记录声音的方式称为声音的波形编码。



# 声音编码（声音质量与存储空间）

采样点的时间间隔越小（越密集），与原始波形函数越吻合。声音编码中的“**采样频率**”，就是单位时间对声音波形的采样次数。采样频率用Hz（赫兹）为单位，每秒钟采样500次即为500Hz。显然，采样频率越高，声音质量会更好，所用的存储空间也越大。

一般音乐CD的采样频率为44**KHz**。为了获得较好的声音效果，往往采用**双声道**，每个声道用2字节（16 bit）来记录波形高度值。

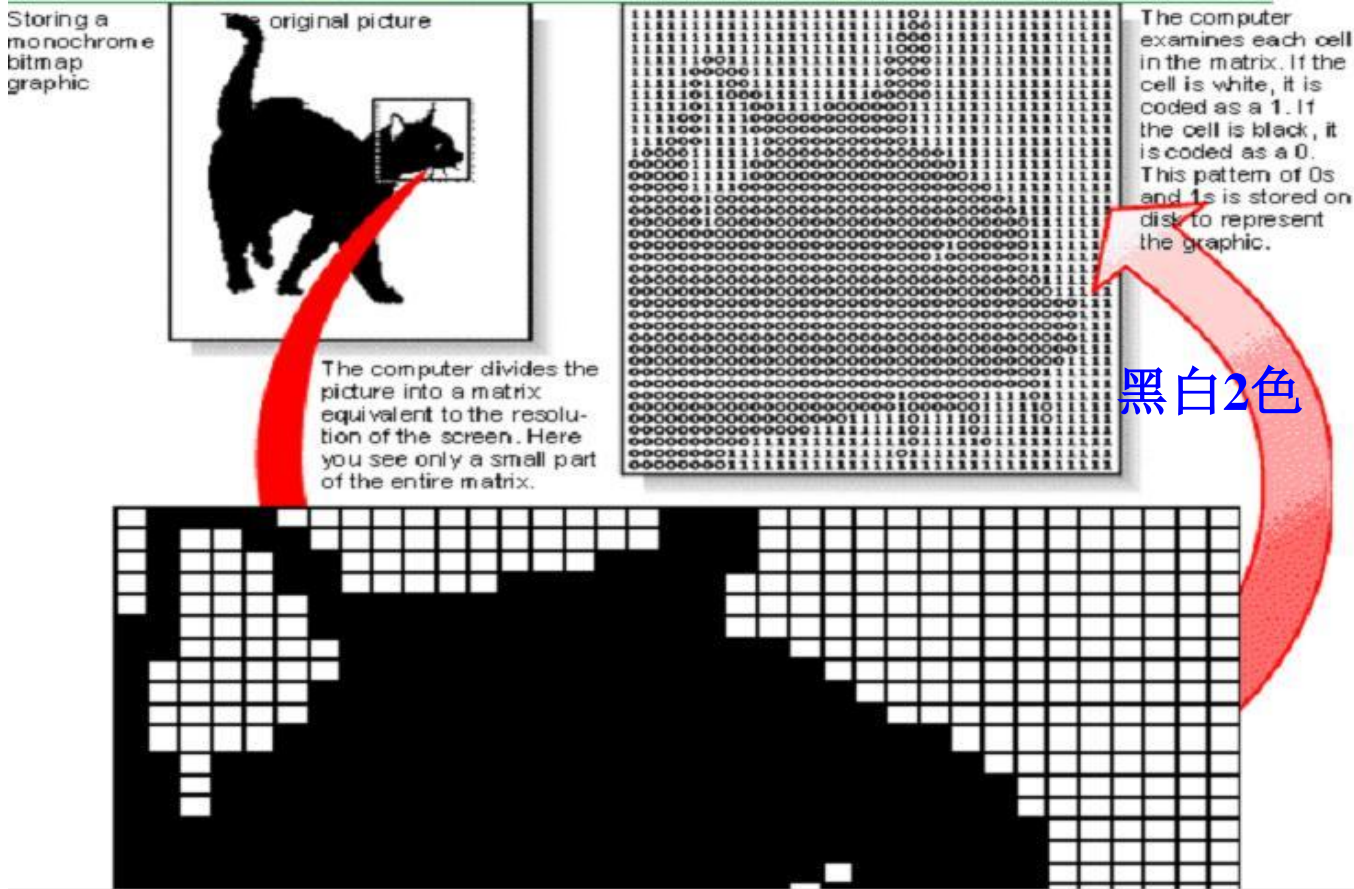
# 声音编码*de*压缩存储

波形编码，存储所需的空间非常大。如果要存储1个小时的音乐，通常需要>600M的存储空间。

现在流行的MP3音乐，实际上对声音的波形编码进行了压缩处理，它使得一首4分钟的歌曲（原本需要40M左右的存储空间）只需约4M左右的存储空间了。

$$44\text{KHz} * 4\text{分钟} * 60\text{秒} * 2\text{声道} * 2\text{字节} = 42\text{M}$$

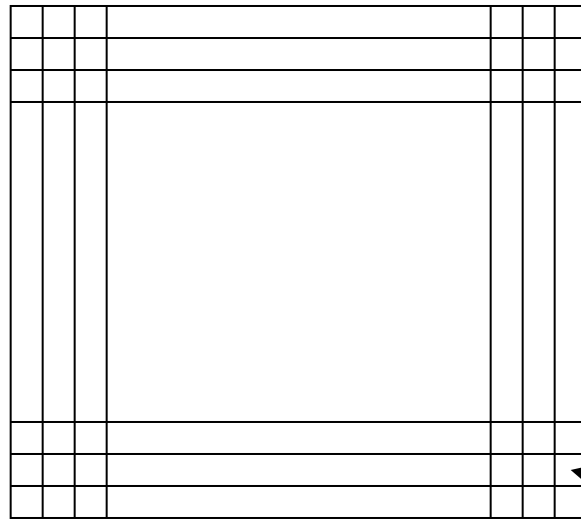

# 图像表示：点阵表示



以颜色编码为基础，将二维平面（空间）离散化为网格点，记录每个网格点上的代表性颜色值。

# 图像表示

640\*480  
个像素



像素的多少代表了分辨率的高低，同样大小的显示器上，像素点越多，分辨率越高

480

像素

640

分辨率：网格点的数目，如1024 × 768

# 颜色编码

颜色是连续的，因此，也需要将颜色的连续光谱以及其它和视觉有关的连续特性离散化，将近似的颜色划分为同一种颜色，用一个特定的二进制数表示它。

位数	颜色数	图像名称
1	2	单色图像
4	16	16色图像
8	256	256色图像
16	65536	HI—Color 图像
24	16672216	True Color 图像 (真彩)

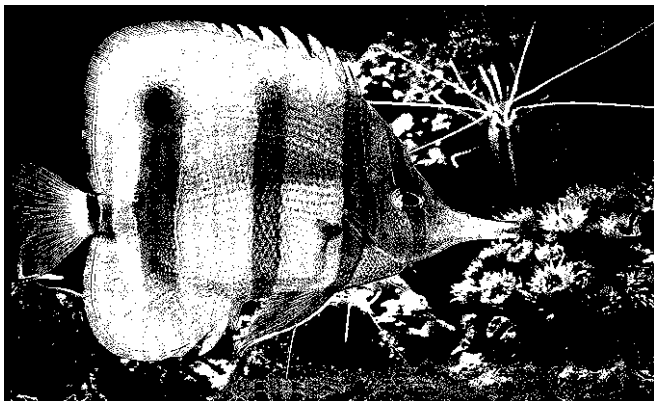
# 颜色混合模型

- **理论上**，任何一种颜色都可用**红 (Red)**、**绿 (Green)**、**蓝 (Blue)** 三种基本颜色按不同的比例混合得到，称为**相加混色**。
- **在计算机中**，将红、绿、蓝三种颜色分别按光强度（深浅）的不同分为256个级别，0级实际上是**黑色**，255级是**纯色**（红、绿或蓝），分别用**8位二进制数**表示，每个像素占**24位**。

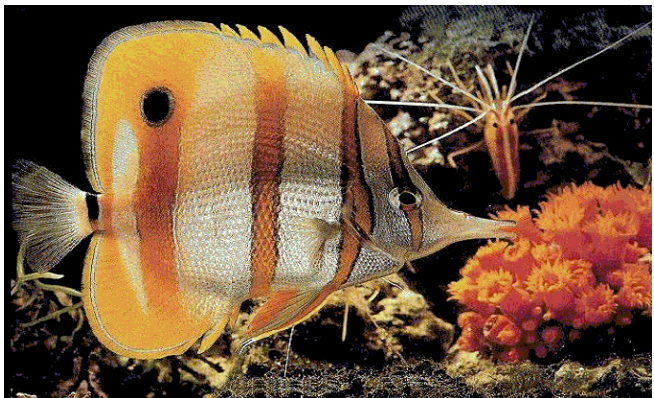
图像的质量由分辨率和颜色确定：分辨率越高以及颜色越丰富，图像的质量就越好

相同分辨率（像素）、不同的颜色编码

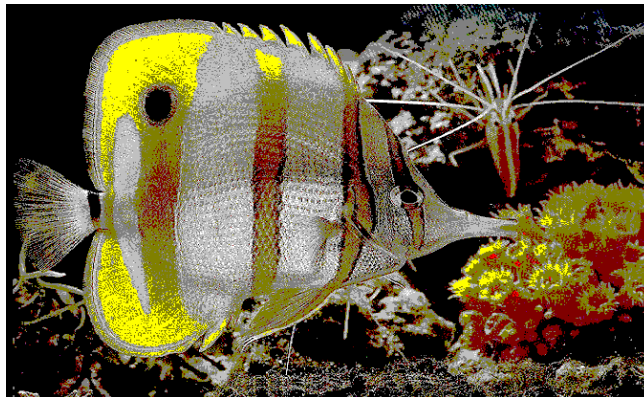
黑白色  
(1位)



256色  
(8位)



16色  
(4位)



真彩色  
(24位)



相同分辨率、不同颜色编码的图像质量比较



# 图像编码的质量—— 由分辨率和颜色决定

不同分辨率、不同颜色编码的图像，其图像质量差别非常大。  
对于同样一幅原始图像：

- 如果对其离散化后的网格点越多，即分辨率越高，则图像越精细，质量越好；
- 如果颜色编码所采用的二进制位数越多，即所能表示的颜色数越多，则图像质量越好。

当然，图像质量越高，其所需的存储空间也越大

# 点阵图的大小计算

用**字节**表示图像文件大小时，一幅未经压缩的数字图像的数据量大小计算如下：

**图像数据量大小 = 像素总数 × 颜色位数 ÷ 8**

例如：一幅  $640 \times 480$  的 256 色图像为  
 $640 \times 480 \times 8 / 8 = 307200$  字节

# 数字视频（影像）

- 视频是由一幅幅单独的画面序列（帧 frame）组成，这些画面以一定的速率（fps）连续地投射在屏幕上，使观察者具有图像连续运动的感觉。
- 视频文件的存储格式有AVI、MPG、MOV等。

# 数字视频表示

- 每个帧由点阵表示。对于分辨率 $640*480$ 的256色图像，有307200个像素，即307200字节（ $640*480=307200$ ）。
- 为了达到视觉效果，通常每秒需要30帧画面。即， $30*307200=9\text{M Byte}$ 来表示。2小时的电影需要66G 字节。
- 上述视频未考虑声音

# 视频数据有大量冗余

- 图像数据表示中的冗余：
  - 空间相关冗余：单张画面（如静态画面）中的很多部分往往有相同的颜色和图像，这种相关性称为空间相关，可用少量数据表示这些空间相关数据
  - 时间相关冗余：在动画或影视图像（动态画面）中，相邻的两帧图像之间产生的变化往往很小（连续节目中活动目标的瞬间变化不大），存在大量重复数据。
  - 视觉冗余（一种数据冗余）：人类的视觉系统对图像场的变化并不是同样敏感（如，对亮度敏感，对色度不敏感；对直线敏感，对斜线不敏感…），但是记录的原始图像数据通常假定系统近似均匀，对视觉敏感和不敏感的部分同等对待，从而产生比理想编码（即把视觉敏感和不敏感的部分区分开来的编码）更多的数据，这就是视觉冗余

# 数据压缩

- 压缩的必要性：待处理的数字化视频、音频信号数据量极大（如前），不压缩，几乎是不可能的。
- 压缩的可行性：原始信息源数据存在着大量冗余

# 压缩的几个基本概念

- 压缩定义：对数据重新编码，压减冗余，从而减少所需的存储空间。
- 解压：压缩数据的还原，称为解压缩或展开
- 压缩比：压缩后的字节数量与原来字节数量比
- 压缩方式：
  - 软硬件结合（特别是视频与图像——解压芯片）
  - 纯软件：文件压缩

# 压缩的类型

- 无损压缩：不丢失信息
- 有损压缩：部分（不重要信息）可能丢失

压缩过程是一种重编码的过程



# 小结

- 信息表示是计算机对信息加工的基础
- 信息表示的核心是编码（计算机以2进制形式编码）
  - 离散化的2进制位编码表示
- 计算机内的信息表示分为两种类型
  - 数值型信息表示：需要考虑与外界一致的运算并产生一致的结果
  - 非数值型信息表示：
    - 符号文字表示：有限集合的离散表示
    - 声音表示：无限连续值，需要离散化
    - 图像视频表示：无限连续值，需要离散化
    - **指令表示：有限集合的离散表示**

## 信息与表示 随堂测试

- 答题二维码

