XIDIAN UNIVERSITY

2020版

微机原理与系统设计

第一章 微机中的信息表示

董明皓，博士

西安电子科技大学

dminghao@xidian.edu.cn

键盘输入55，计算机怎么识别此数？

基础知识回顾

- 位：bit，1个二进制位，0、1，计算机存储信息的最小单元
- 字节:byte，8个二进制位，存储8个二进制数，0~255，-128~127
- 字：word，16个二进制位，2个字节，存储16个二进制数
- 双字:double word，32个二进制位，4个字节，存储32个二进制数
- 字长：基本数据单元所包含的二进制位数，8086-8/16。

存储器是按字节组织，即存储器存放的数位字节数

目录（2课时）

1.1数制表示-常用表示法

表征-Representation

常用数制：二、十、十六进制

- **二级制表示：**

符号 0、1 集合，尾符 B(Binary)， 110111B

- **十进制表示：**

符号 0～9 集合，尾符 D(Decimal) 或缺省，55D或55

- **十六进制表示：**

符号 0～9、A～F 集合，尾符H(Hexadecimal)， 37H

1.1数制表示-常用表示法

- 计算机中的数均用二进制表示（高低电平）；
- 人们最熟悉、最常用的是十进制数；
- 为书写方便，微机中的二进制数用十六进制数表示；

1.1数制表示-常用表示法

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

1.1数制表示-权值表示法

任意一个数X可以表示成*b*进制数：

$$x = \sum_{i=-m}^{n} k_i b^i = k_{-m} b^{-m} + \dots + k_{-1} b^{-1} + k_0 + k_1 b + k_n b^n$$

b-基数；m-小数位的位数；n-整数位的位数-1

$$255 = 2 \times 10^2 + 5 \times 10^1 + 5 \times 10^0$$

$$369.2D = 3 \times 10^2 + 6 \times 10^1 + 9 \times 10^0 + 2 \times 10^{-1}$$

$$11010B = 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$11010B = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$3C.4H = 3 \times 16^1 + 12 \times 16^0 + 4 \times 16^{-1}$$

1.2数制转换-其他→十进制

权值表示法

$$110111B = 1\times 2^5 + 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^1$$

$$=32 + 16 + 0 + 4 + 2 + 1 = 55$$

$$3C.4H = 3\times 16^1 + 12\times 16^0 + 4\times 16^{-1}$$

$$= 48 + 12 + 0.25 = 60.25/60.25D$$

1.2数制转换–十进制→其他（整数）

（1）整数部分：除基取余法

例. 十进制数55转换为十六进制数

转换过程如下：

$$
\begin{array}{l|l}
16 & 55 \\
\hline
16 & 3 \;\ldots\ldots\text{余数为：}7 = K_0 \\
\hline
16 & 0 \;\ldots\ldots\text{余数为：}3 = K_1
\end{array}
$$

低位

↑

高位

则：55 = 37H

十进制转其他进制，整数除基取余法，直至商为0，下高上低

1.2数制转换-十进制→其他(整数)

（1）整数部分：除基取余法

例. 十进制数55转换为二进制数

转换过程如下：

2	55		
2	27	……余数为：	$1 = K_0$
2	13	……余数为：	$1 = K_1$
2	6	……余数为：	$1 = K_2$
2	3	……余数为：	$0 = K_3$
2	1	……余数为：	$1 = K_4$
2	0	……余数为：	$1 = K_5$

低位

高位

则：55 = 110111B

十进制转其他进制，整数除基取余法，直至商为0，下高上低

1.2数制转换-十进制→二进制(整数)

（1）整数部分：权逼近法

例2. 十进制数55转换为二进制数

实际应用中：权逼近法变体

	512	256	128	64	32	16	8	4	2	1
55	0	0	0	0	1	1	0	1	1	1

则：55 = 110111B

十进制转其他进制，权逼近法

1.2数制转换-十进制→其他(小数)

（2）小数部分：乘基取余法

例. 十进制小数0.6875转化为二进制小数

转换过程如下：

0.6875×2 =1.375	K_{-1} =1	高位
0.375 ×2 =0.75	K_{-2} =0	↓
0.75×2 =1.5	K_{-3} =1	
0.5×2 =1.0	K_{-4} =1	低位

则：0.6875=0.1011B

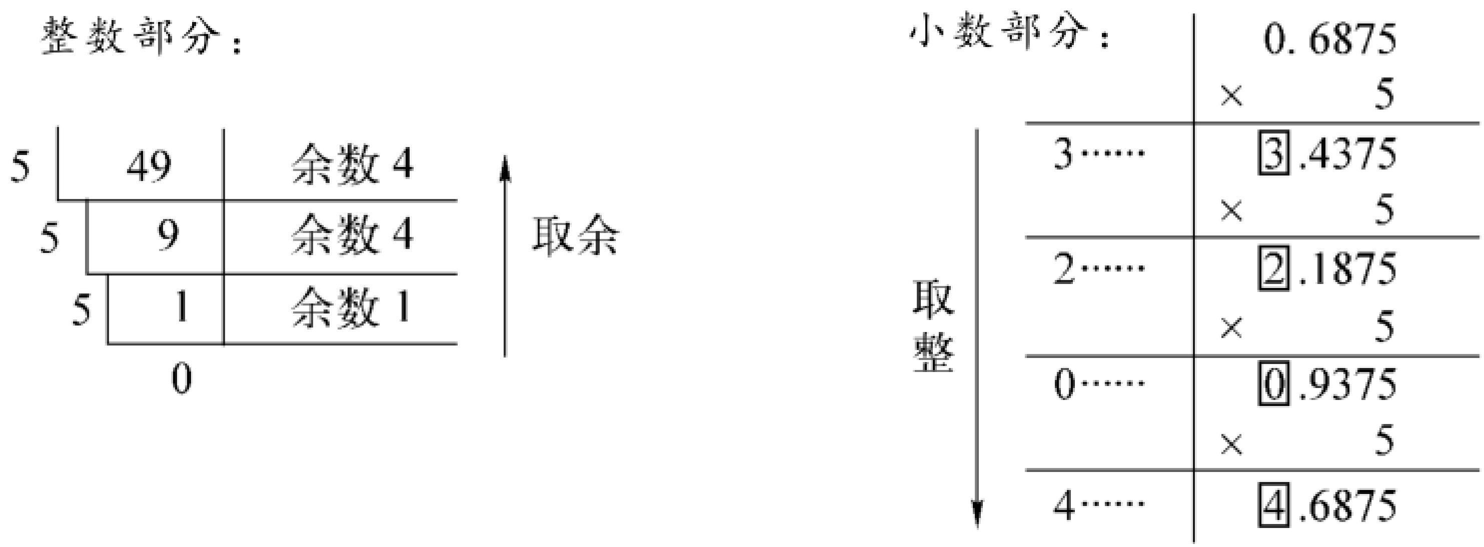
十进制转其他进制，小数乘基取整法，直至积为0，上高下低

小结：数制转换

其他-->十进制	权值表示法
十进制-->其他	权逼近法
	整数除基取余法，直至商为0，下高上低
	小数乘基取整法，直至积为0，上高下低

（3）整数小数都存在

例. 将49.6875D 转化为5进制数，取4位小数



即$(49.6875)_{10}=(144.3204)_5$（取 4 位小数）。

整数、小数分别转换，结果拼接

目录（2课时）

2二进制数运算-算数运算

以下皆为无符号数

加（+）、减（-）、乘（×）、除（÷）

所有运算前应搞清：字长（字运算？、字节运算？）

例：

$$\begin{array}{r} 1\,0\,1\,1\,0\,1\,0\,1\,B \\ +\,0\,0\,0\,0\,1\,1\,1\,1\,B \\ \hline 1\,1\,0\,0\,0\,1\,0\,0\,B \end{array} \qquad \begin{array}{r} 1\,0\,1\,1\,0\,1\,0\,1\,B \\ -\,0\,0\,0\,0\,1\,1\,1\,1\,B \\ \hline 1\,0\,1\,0\,0\,1\,1\,0\,B \end{array}$$

例：96+72=?

XIDIAN UNIVERSITY 西安电子科技大学

2二进制数运算-逻辑运算

以下皆为无符号数

与（AND）、或（OR）、异或（XOR）、非四种运算（NOT）

```
  1 0 1 1 0 1 0 1 B
∧ 0 0 0 1 1 1 1 B
-------------------
  0 0 0 0 0 1 0 1 B
```

定位清0，剩余位不变

```
  1 0 1 1 0 1 0 1 B
∨ 0 0 0 0 1 1 1 1 B
-------------------
  1 0 1 1 1 1 1 1 B
```

定位置1,剩余位不变

```
  1 0 1 1 0 1 0 1 B
∀ 0 0 0 1 1 1 1 B
-------------------
  1 0 1 1 1 0 1 0 B
```

是否不同？某些位取反/不变

X= 1 0 1 1 0 1 0 1 B

$\overline{X}$ = 0 1 0 0 1 0 1 0 B

定位取反

逻辑运算不越位

XIDIAN UNIVERSITY

目录（2课时）

-55，计算机如何表示？

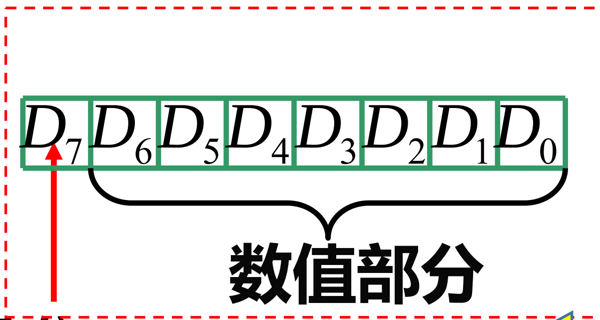
3有符号数表示-机器数表示

机器数的符号表示法：

- 为了表示一个有符号数，除了数值位以外，还应制定**符号位**，通常以这个数的最高位表示符号位，这种表示方法称为**机器数表示法**。
- 我们假定数为整数，对8位有符号二进制整数，表示：

真实的数制：-55

真值



机器数

0：表示正数
1：表示负数
符号位

有符号数有大小，无符号数有高低。

3有符号数表示-机器数表示

机器数的表示方法：

- 机器数是为让机器能准确表示并运算数据的表示方法
- 机器数的表达方式不止一种
- 在将数的符号用数码（0或1）表示后，数值部分究竟是保留原来的形式，还是按照一定的规则做某些变化，这取决于运算方法的需要。

我们常用的机器数有三种形式

1. 原码表示：机器数最直观，最基础的表示
2. 反码表示：原码除符号位外，所有位取反
3. 补码表示：反码+1（含符号位）

3.1有符号数表示-原码表示法

原码表示法：

约定数码序列中的最高位为符号位；

正数的符号位用0表示，负数的符号位用1表示；

其余有效数值部分则用二进制的绝对值表示。

一个数X的原码记为：$[X]_{原}$

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	1	1

$[+3]_{原} = 0000\ 0011B$

数值部分=绝对值（用无符号二进制数表示）

符号位

3.2有符号数表示-补码表示法

补码表示法：有符号二进制数的表示方法

（1）补码的定义

一个数X的补码记为 $[X]_{补}$，补码可定义为：

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

从定义可见，正数的补码=原码，即 $[X]_{补} = [X]_{补}$，

所以，只有负数求补的问题。

计算机中有符号数，均用补码表示。

3.2有符号数表示-补码表示法

补码表示法：有符号二进制数的表示方法

（2）补码的求法：定义法

$$[x]_{补} = 2^n + x = 2^n - |x| ,\quad x<0$$

即负数x的补码等于模 2加上其真值（或减去其真值的绝对值）。

例：字长为8位，原码x=-15，则求其补码

$$[X]_{补} = 2^8 - (15)$$

$$= 256D-15D$$

$$= 1111\ 0001B \qquad (\mathrm{mod}\ 2^n)$$

这种方法因要做一次减法，很不方便。

正数求补：原码即补码

负数求补：负数变原码，原码按位取反(除符号位) → +1(含符号位)

补码表示法：有符号二进制数的表示方法

（3）补码→原码

一个用补码表示的负数，如将$[X]_{补}$再求一次补，即将$[X]_{补}$除符号位外取反加1，就可得到$[X]_{原}$，用下式表示：

$$\boxed{[[X]_{补}]_{补}=[X]_{原}}$$

如：

$[X]_{原}$= 11010111B

$[X]_{补}$= 10101001B

$[[X]_{补}]_{补}$= 11010111B=$[X]_{原}$

3.3有符号数表示-练习

例：某内存单元的内容是D2H，为8位二进制数，其所表示的十进制数是多少？

$[X]_{补} = D2H = \underline{1}\ 1010010B$

$X = [[X]_{补}]_{补} \qquad = [11010010]_{补}$

$= -\ 0101110B$

所以：$X = -\ 46$

- 特殊数10000000B所表示的十进制数是多少？

8位二进制补码所能表示的数的范围：-128~+127
16位二进制补码所能表示的数的范围：-32768 ~+32767

目录（2课时）

补码运算规则1：

两个n位二进制数之和的补码等于这两数补码之和，即：

$$[X+Y]_{补}= [X]_{补} + [Y]_{补}$$

例：[（+33）+（-15）]补

	0 0 1 0 0 0 0 1 B	$[+33]_{补}$
	+ 1 1 1 1 0 0 0 1 B	$[-15]_{补}$
进位，丢掉 →	[1] 0 0 0 1 0 0 1 0 B	$[+18]_{补}$

补码表示的数运算后仍然是补码。

4有符号数运算

如果计算机的字长为n位，n位二进制数的最高位为符号位，其余n-1位为数值位，采用补码表示法时，可表示的数X的范围为：$-2^{n-1} \le X \le 2^{n-1}-1$

当n=8时，可表示的有符号数的补码表示范围为：

-128 ～+127

当n=16时，可表示的有符号数的补码表示范围为：

-32768 ～ +32767

两个有符号数进行加减运算时，如果运算结果超出可表示的有符号数的范围时，就会发生溢出，使计算结果出错。

很显然，溢出只能出现在两个同号数相加或两个异号数相减的情况下。

4有符号数运算

例： （+72）+（+98）=+170>+127 **溢出**

	0 1 0 0 1 0 0 0 B	[+72]	补
+	0 1 1 0 0 0 1 0 B	[+98]	补
	1 0 1 0 1 0 1 0 B	[-86]	补

有进位 C_p=1

无进位 C_S=0

溢出，结果出错（正溢出）

$$C_s \forall C_p = 1$$

异或

设次高位（数值部分最高位）向最高位（符号位）的进位标志为 C_p；最高位（符号位）和次高位的进位相加的进位标志为 C_S 。

4有符号数运算

例： （-83）+（-80）=-163<-128 **溢出**

	1 0 1 0 1 1 0 1 B	$[-83]_{补}$
+	1 0 1 1 0 0 0 0 B	$[-80]_{补}$
[1]	0 1 0 1 1 1 0 1 B	$[+93]_{补}$

无进位 C_p=0

有进位 C_S=1

溢出，结果出错（负溢出）

$$\boxed{C_s \forall C_p = 1}$$

4有符号数运算

结论：对于加法运算

（1）如果次高位有进位而最高位无进位，则结果溢出；

（2）如果次高位无进位而最高位有进位，则结果溢出。

对于减法运算

（1）如果次高位有借位而最高位无借位，则结果溢出；

（2）如果次高位无借位而最高位有借位，则结果溢出。

两个8位带符号二进制数相加或相减时，若$C_7 \oplus C_6 = 1$，则结果产生溢出。（C_7为最高位的进(借)位；C_6为次高位的进(借)位。）

4有符号数运算

补码运算规则2：

两个n位二进制数之差的补码等于这两数补码之差，即：

$$[X-Y]_{补} = [X]_{补} - [Y]_{补}$$

例：[（+33）-（-15）]补

	0 0 1 0 0 0 0 1 B	[+33]补
	- 1 1 1 1 0 0 0 1 B	[-15]补
借位，丢掉 →	[1]0 0 1 1 0 0 0 0 B	[+48]补

补码表示的数运算后仍然是补码。

4有符号数运算

补码运算规则3：

补码减法运算时，也可以利用加法基本公式，即：

$$[X-Y]_{补} = [X]_{补} + [-Y]_{补}$$

因为：$X-Y = X+(-Y)$

所以：$[X-Y]_{补} = [X+(-Y)]_{补} = [X]_{补} \quad [-Y]_{补}$

一般称已知 $[Y]_{补}$求得 补的过程叫变补或求负，实现减法变加法。

补码表示的数运算后仍然是补码。

目录（2课时）

8241BCD码(Binary Coded Decimal):

- 计算机只认识0、1二进制代码，但人们最习惯的是十进制。为了解决这一矛盾，提出了一个比较适合于十进制系统的二进制代码的特殊形式—BCD码。
- BCD码是用四位二进制数表示1位0 —9的十进制数，而4位二进制数码有16种组合，原则上可任选10种作为代码，但为便于记忆和比较直观，最常用的是8421BCD码，8、4、2、1分别是4位二进制数的位权值。

5BCD编码

十进制数和8421BCD编码的对应关系。

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

十进制数和BCD码相互转换

75.4 → BCD码

$$75.4 = (0111\,0101.0100)_{BCD}$$

BCD码10000101.0101 → 十进制数

$$(1000\,0101.0101)_{BCD} = 85.5$$

同一个8位二进制代码表示的数，当认为它表示的是二进制数和认为它表示的是二进制编码的十进制数，数值是不相同的。如：

$$(00011000)_2 = 24$$

$$(0001\,1000)_{BCD} = 18$$

5BCD编码

在计算机中，BCD码有两种基本格式

- **a.组合式BCD码**
- **b.分离式BCD码**

a.组合式BCD码

两位十进制存放在一个字节中。

如：数24的存放格式：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	0	0	1	0	0
2				4			

b.分离式BCD码

每位数存放在8位字节的低4位，高4位的内容与数值无关。

如：数24的存放格式：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
x	x	x	x	0	0	1	0	2
x	x	x	x	0	1	0	0	4

BCD码的加减法运算

由于BCD编码是将每个十进制数用一组4位二进制数表示，若将这种BCD编码直接交计算机运算，计算机总是把它按二进制数处理，所以结果可能出错。 如：38+49=87

0 0 1 1 1 0 0 0	BCD	38
0 1 0 0 1 0 0 1	BCD	49
1 0 0 0 0 0 0 1		87

81

显然，结果出错。

出错原因：十进制相加应逢十进一，但计算机按二进制运算，每四位一组，低四位向高四位进位相当十六进制运算，“逢十六进一”。所以当结果超过9时将比正确值少6。

解决办法：加六修正

加六修正规则：

（1）如果两个BCD码位相加没有进位，并且结果≤9，则该位不需修正。

（2）如果两个BCD码位相加有进位，或者其结果>9，该位进行加六修正。

（3）低位修正结果使高位>9时，高位进行加六修正。

5BCD编码

例：94+7=101

	1 0 0 1	0 1 0 0	[BCD]94
+	0 0 0 0	0 1 1 1	[BCD] 7
	1 0 0 1	1 0 1 1	[BCD]低4位满足法则1
+	0 0 0 0	0 1 1 0	[BCD]加六修正
	1 0 1 0	0 0 0 1	[BCD]高4位满足法则3
+	0 1 1 0	0 0 0 0	[BCD]加六修正
[1]	0 0 0 0	0 0 0 1	[BCD]101结果正确

?=101

减六修正规则：

（1）如果两个BCD码位相减没有借位，则该位不需修正。

（2）如果两个BCD码位相减有借位，则该位进行减六修正。

5BCD编码

例：50 - 29=21

	0 1 0 1 ¦ 0 0 0 0	50
—	0 0 1 0 ¦ 1 0 0 1	29
	0 0 1 0 ¦ 0 1 1 1	低码位有借位
—	0 0 0 0 ¦ 0 1 1 0	减六修正
	0 0 1 0 0 0 0 1	21结果正确

目录（2课时）

6ASCII码

使用场合：汇编语言

ASCII：用7位二进制代码对任一字符编码，

包括：显示字符、非显示字符

共128个

- ‘0’- ‘9’ 10个字符 30H – 39H
- ‘A’- ‘F’41H -46H
- ‘a’- ‘f’ 61H –66H

要求掌握常用字符的ASCII码：

0～9， A~Z，a~z， 空格，回车，换行，Esc

行	列 位 654→ ↓ 3210	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	“	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	’	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	—	*	M	]	m	}
E	1110	SO	RS	.	>	N	Λ	n	~
F	1111	SI	US	/	?	O	_	o	DEL

- 0~9： 30H~39H　　回车CR： 0DH
- A~Z: 41H~5AH　　换行LF： 0AH
- a~z: 61H~7AH　　退格BS： 08H
- 　　Tab： 09H

数制的表示（常用表示法）

1. 数制表示与转换
2. 二进制数的运算规则
3. 有符号数的表示
4. 有符号数的运算及其溢出规则
5. BCD编码方法及其运算
6. ASCII编码方法

总结

- 1.数制：十进制、二进制、十六进制的表示、关系、相互转换
- 2. 二进制运算：数学运算，逻辑运算
- 3. 有符号数的表示：原码、补码
- 4. 有符号数的运算及溢出问题：
- 5. BCD数的表示及计算：加6修正，减6修正
- 6. ASCII码

课后作业

【1.1】（1）（2）
【1.2】（2）（4）（6）
【1.3】（1）（2）
【1.4】（1）（2）
【1.5】（2）
【1.6】（1）（6）
【1.7】（1）（2）
【1.8】（1）（2）（3）（4）
【1.9】（1）（2）（5）（6）
【1.10】（2）（3）（4）（6）（7）
【1.11】（1）
【1.13】
【1.15】（1）
【1.16】
【1.17】
【1.20】
【1.22】（2）

几个关键问题

4. 怎么学习这门课？-了解课程特点

60%靠记忆：内容多，知识庞大（慢即是快、冰山原则）

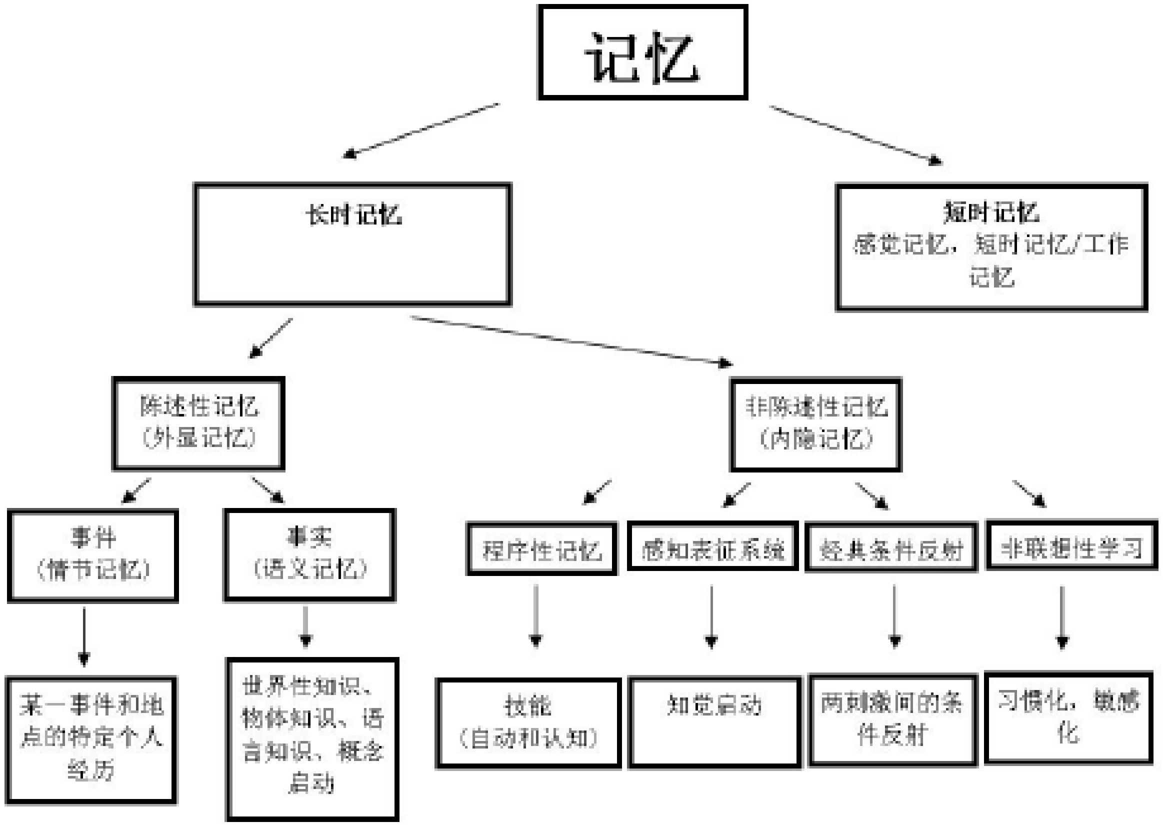
名词重复+刻意训练（讲课当复习）

40%靠理解：摸清事物的合理性

善用脑图、软件硬件齐头并进

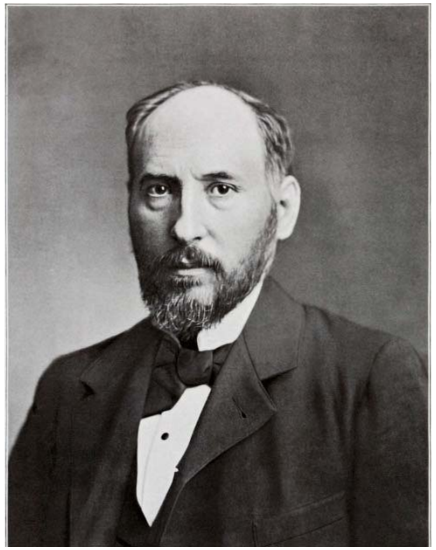
几个关键问题

4. 怎么学习这门课？-了解大脑记忆的原理



几个关键问题

4. 怎么学习这门课？ -专家化的神经生理基础

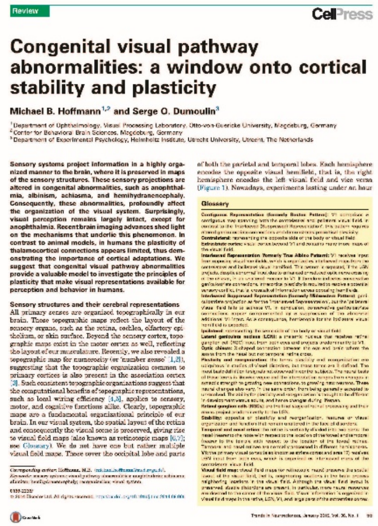


卡哈尔，1906诺奖

articles

Long-term *in vivo* imaging of experience-dependent synaptic plasticity in adult cortex

2002, 420:788-794



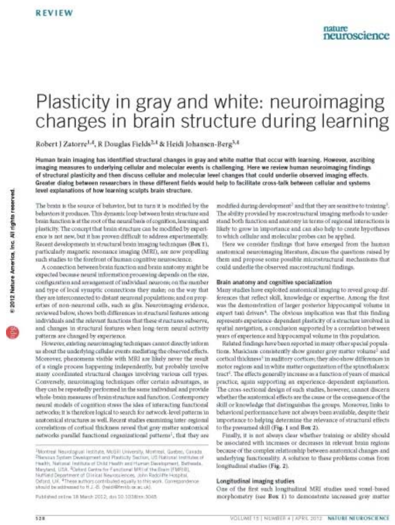
Review

CellPress

Congenital visual pathway abnormalities: a window onto cortical stability and plasticity

Michael B. Hoffmann and Serge O. Dumoulin

2015, 38:55-65



REVIEW

nature neuroscience

Plasticity in gray and white: neuroimaging changes in brain structure during learning

2012, 15:528-532

脑时时可变，现时之脑是此前所有时刻变化的集中体现	神经元存在可观测的功能和结构可塑性变化	脑可塑性是学习和记忆、脑疾病及康复的生物基础	神经影像无侵、宽视场、高时空分辨地观测脑可塑性变化

几个关键问题

4. 怎么学习这门课？-转化到日常学习中

- 区分外显学习(explicit learning)和过程学习(procedural learning)的特点
- 刻意训练(deliberate practice)，形成共同激活
- 建立专家模板
- 构建知识矩阵

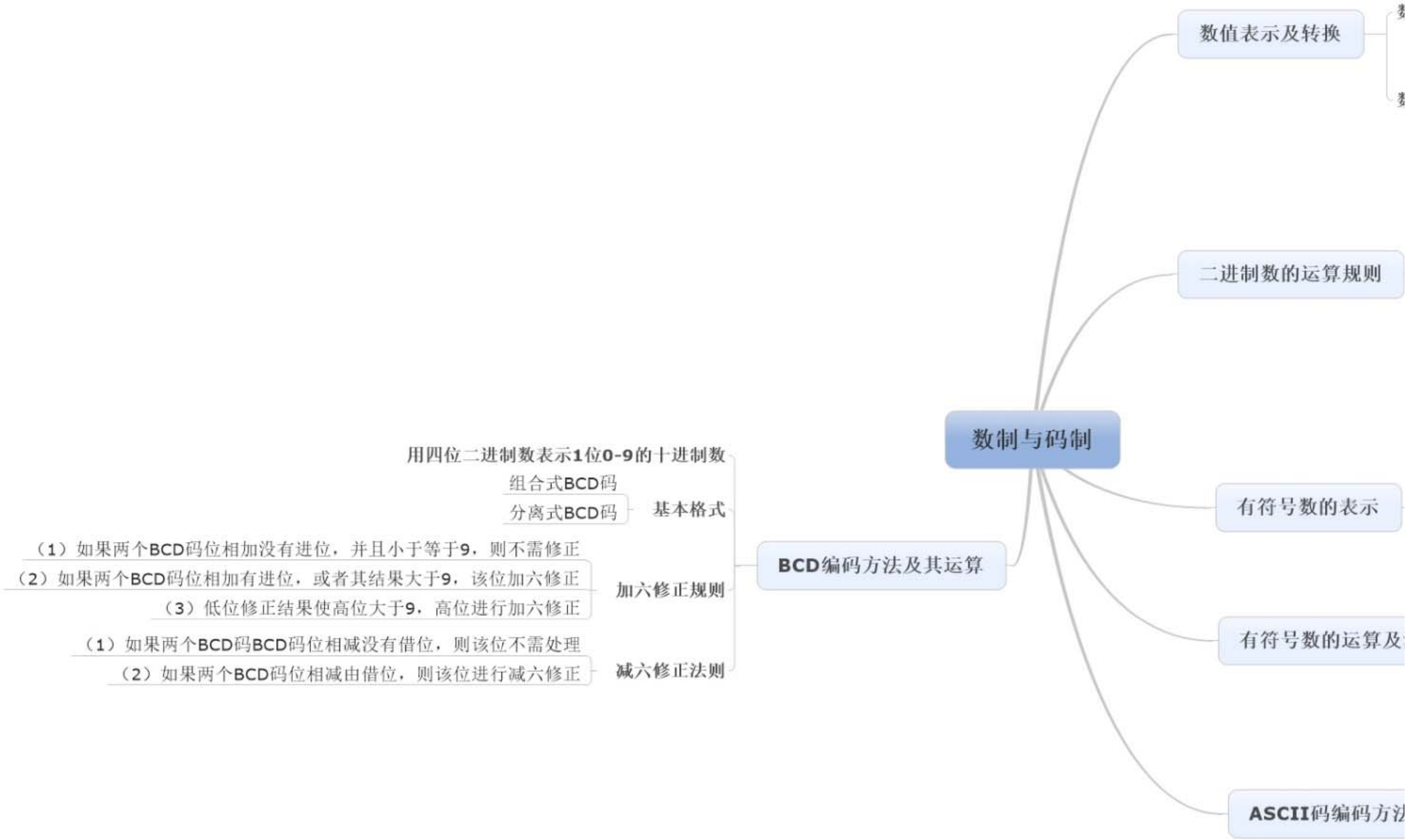
课后阅读

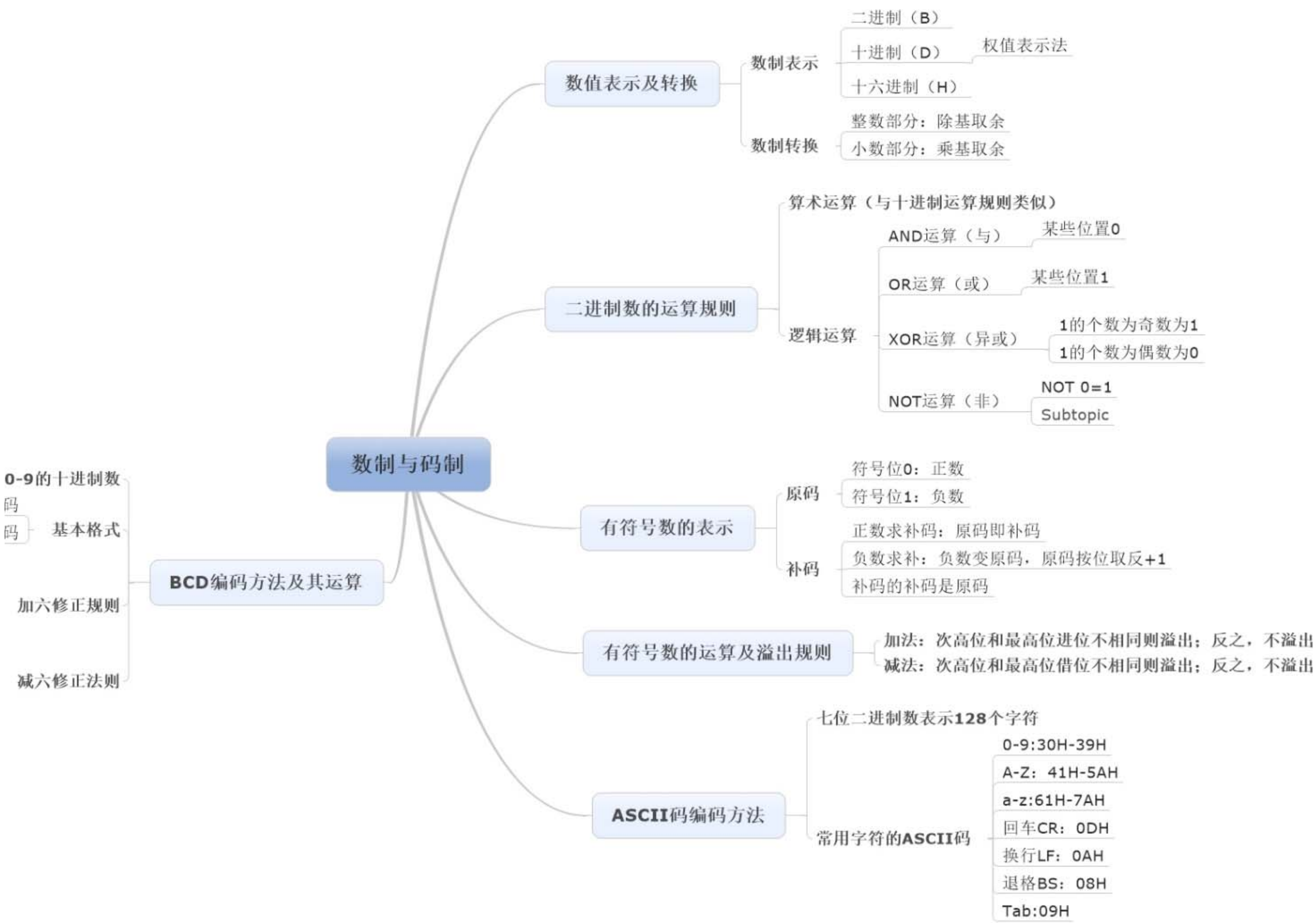
1. 个人主页阅读材料

   类脑计算芯片与类脑智能机器人发展现状与思考.pdf

   类脑智能研究现状与发展思考.pdf

2. 学习使用软件：Mind manager/ Mind Jet


数制与码制
数值表示及转换
二进制数的运算规则
有符号数的表示
有符号数的运算及
ASCII码编码方
BCD编码方法及其运算
基本格式
用四位二进制数表示1位0-9的十进制数
组合式BCD码
分离式BCD码
加六修正规则
（1）如果两个BCD码位相加没有进位，并且小于等于9，则不需修正
（2）如果两个BCD码位相加有进位，或者其结果大于9，该位加六修正
（3）低位修正结果使高位大于9，高位进行加六修正
减六修正法则
（1）如果两个BCD码BCD码位相减没有借位，则该位不需处理
（2）如果两个BCD码位相减由借位，则该位进行减六修正


数制与码制
数值表示及转换
数制表示
二进制（B）
十进制（D）
权值表示法
十六进制（H）
数制转换
整数部分：除基取余
小数部分：乘基取余
二进制数的运算规则
算术运算（与十进制运算规则类似）
逻辑运算
AND运算（与）
某些位置0
OR运算（或）
某些位置1
XOR运算（异或）
1的个数为奇数为1
1的个数为偶数为0
NOT运算（非）
NOT 0=1
Subtopic
有符号数的表示
原码
符号位0：正数
符号位1：负数
补码
正数求补码：原码即补码
负数求补：负数变原码，原码按位取反+1
补码的补码是原码
有符号数的运算及溢出规则
加法：次高位和最高位进位不相同则溢出；反之，不溢出
减法：次高位和最高位借位不相同则溢出；反之，不溢出
ASCII码编码方法
七位二进制数表示128个字符
常用字符的ASCII码
0-9:30H-39H
A-Z：41H-5AH
a-z:61H-7AH
回车CR：0DH
换行LF：0AH
退格BS：08H
Tab:09H
BCD编码方法及其运算
基本格式
0-9的十进制数
码
码
加六修正规则
减六修正法则