计算机信息表达

Information Representation

天津大学　计算机科学与技术学院 张鹏

- **Bit (位)**
  - the smallest unit of storage
  - Everything in a computer is 0's and 1's ➔ Bits
  - **why? ➔ Computer Hardware**
    - Chip uses electricity 0/1 states
    - Hard drive uses spots North/South magnetism 0/1 states
  - A bit is too small to be much use

Byte（字节）

- A larger unit of storage than Bit
- A group of 8 bits
  - e.g. 0 1 0 1 1 0 1 0
- One byte can store one letter, e.g. 'b' or 'x'

Number of bits	Distinct Patterns
1	0 1
2	00 01 10 11
3	000 001 010 011 100 101 110 111

• How much exactly can one byte hold?

- How many distinct patterns can be made with 1, 2, or 3 bits?

Think about 3 Bits

1) Consider just the leftmost bit, it can only be 0 or 1
2) Leftmost bit is 0, append 2-bit patterns
3) Leftmost bit is 1, append 2-bit patterns again

Result: 3-bits has twice as many patterns as 2-bits

Number of bits	Distinct Patterns
1	0 1
2	00 01 10 11
3	000 001 010 011 100 101 110 111

- **How much exactly can one byte hold?**
  - **In general: add 1 bit, double the number of patterns**

Bit number	Pattern number
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256

1 byte = 8 bits
One Byte - 256 Patterns

  - **Mathematically: n bits yields 2^n patterns (2 to the n-th power)**

- **How to use the 256 patterns?**
- **How to store a number in a byte?**

- Start with 0, go up, one pattern per number, until run out of 256 patterns
- One byte can hold a number between 0 and 255
  - i.e. with 256 distinct patterns, we can store a number in the range 0..255

- **Code:** pixel.setRed(n) took a number 0..255. Why?
  - The red/green/blue image numbers are each stored in **one byte**

- **Bytes**

  - "Byte" - **unit** of information storage
    - A document, an image, a movie .. how many bytes?

  - 1 byte is enough to hold 1 typed letter, e.g. 'b' or 'X’, **how?**

  - Later we‘ll look at **storage** in: RAM, hard drives, flash drives. All measured in bytes, despite being very different hardware.

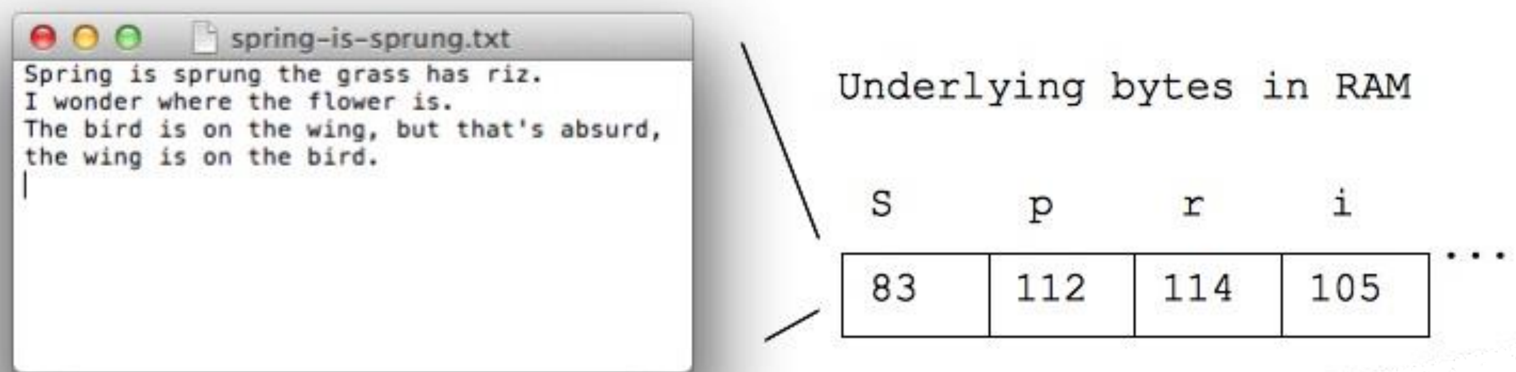
- **Bytes and Letters - ASCII Code**
  - ASCII: American Standard Code for Information Interchange
  - An encoding representing each typed letter by number, each number is stored in one byte of space in the computer (0..255)
    - A is 65
    - B is 66
    - a is 96
    - space is 32

- **Unicode Code**
  - An encoding for Mandarin, Greek, Arabic, etc. languages
  - Typically 2-bytes per "letter"

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

- **Typing, Bytes, and You**
  - An example of bytes in your daily life
    - When you type letters on your phone or computer, each letter is stored as a number in a byte. When you send a text message, the numbers are sent.
  - Text is quite compact, using few bytes, compared to images etc.



- **Other Storage Units**
  - 8 bits ➔ 1 byte
  - More Bytes !
    - **Kilobyte (KB):** about 1 thousand bytes **(~1000 Bytes)**
    - **Megabyte (MB):** about 1 million bytes **(~1000 KB)**
    - **Gigabyte, GB**: about 1 billion bytes **(~1000 MB)**
    - **Terabyte, TB:** about 1 trillion bytes **(~1000 GB)**

- **Kilobyte or KB**

  - A small email text is about 2 KB
  - A 5 page paper might be 100 KB
  - Text does not take a lot of bytes to store compared to images or video

  - Math: if you have N bytes, that's N/1000 KB
    - e.g. 23,000 bytes is about 23 KB

- **Megabyte or MB**

  - Megabyte (MB) - about 1 million bytes
  - aka about 1000 KB

  - MP3 audio is about 1 megabyte per minute

  - A high quality digital picture is about 2-5 megabytes

  - Math: if you have N KB, that's about N/1000 MB
    - e.g. 45,400 KB is 45.4 MB

- **Gigabyte or GB**
  - Gigabyte GB = about a billion bytes
  - aka about 1000 MB
  - Common sized unit modern hardware

  - An ordinary computer in 2012 might have 4 GB RAM
  - A DVD disk has a capacity 4.7GB (single layer)
  - A flash drive might hold 16 GB
  - A hard drive might hold 750 GB

- **Math - You Try It**
  - 2,000,000 bytes is about how many MB?
  - 23,000 KB is about how many MB?
  - 500 KB is about how many MB?
  - How many GB is 4,000,000,000 bytes?
  - Say you have many 5 MB .jpeg images. How many fit on a 16 GB flash drive?

- **Terabyte or TB**
  - One terabyte (TB) is about **1000 gigabytes**, or roughly **1 trillion bytes**.
  - You can buy 1 TB and 2 TB hard drives today, so we are just beginning the time when this term comes in to common use.
  - Gigabyte used to be an exotic term too, until Moore's law made it common.

- **Gigahertz (GHz) vs. Gigabyte (GB)**
  - Speed, not Bytes
  - One gigahertz is 1 billion cycles per second.
  - Higher gigahertz CPUs also tend to be more expensive to produce and they use more power.

- **Kilobyte / Megabyte / Gigabyte Word Problems**

Word Problems	Solution
Alice has 600 MB of data. Bob has 700 MB of data. Will it all fit on Alice's 2 GB thumb drive?	
Alice has 100 small images, each of which is 500 KB. How much space do they take up overall in MB?	
Your ghost hunting group is recording the sound inside a haunted Stanford classroom for 20 hours as MP3 audio files. About how much data will that be, expressed in GB?	

- **Alternate Terminology**
  - Kilobyte, KB, (~1000 Bytes) V.S Kibibyte (2^10 Bytes)
  - Megabyte, MB, (~1000 KB) V.S Mebibyte (2^20 Bytes)
  - Gigabyte, GB, (~1000 MB) V.S Gibibyte (2^30 Bytes)
  - Terabyte, TB, (~1000 GB) V.S Tebibyte (2^40 Bytes)
  - There are two schemes, the "1000" system, and the "1024" system .. these result in definitions of MB GB TB which differ by up to 10%.
  - For this class, we ignore that level of detail, and think of the factors as just "about a thousand".

- **Radix Number System (进制系统)**
  - Radix (进制)
    - In mathematical numeral systems, the radix is the number of unique digits, including zero, used to represent numbers in a positional numeral system (进位制).
    - In a system with radix b ($b > 1$), a string of digits $d_1 \ldots d_n$ denotes the number:

$$d_1 b^{n-1} + d_2 b^{n-2} + \ldots + d_n b^0, \qquad \text{where } 0 \le d_i < b$$

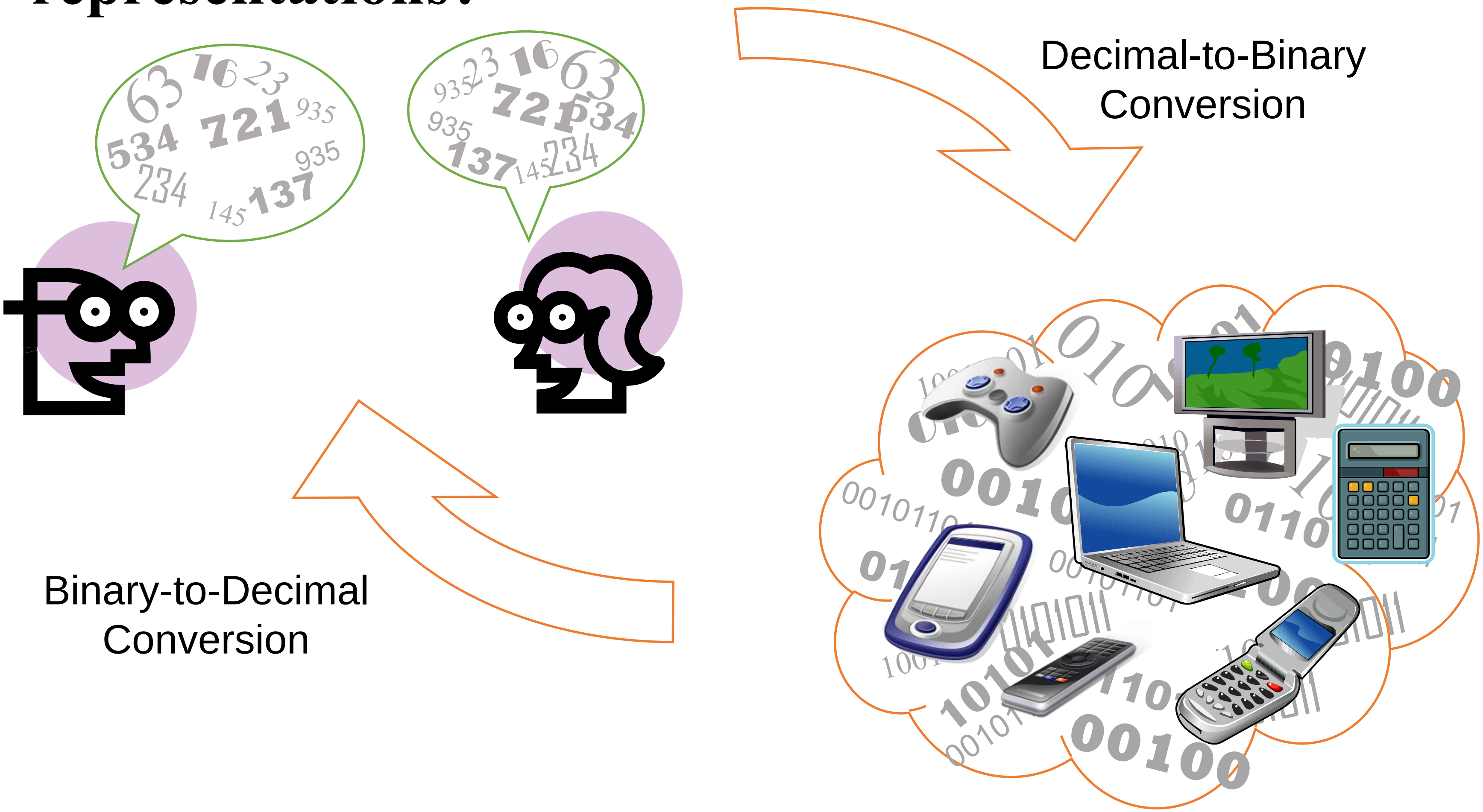
- **Radix Number System (进制系统): Radix=r**

$$a_{n-1}, a_{n-2}, \dots, a_0, a_{-1}, \dots, a_{-(m-1)}, a_{-m}$$

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

  - **Decimal System**: radix=10 ∈ {0,1,2,3,4,5,6,7,8,9}
    - $(4567.89)_{10}$
  - **Binary System:** radix= 2 ∈ {0,1}
    - $(11011.101)_2$=1*2^4+1*2^3+0*2^2+1*2^1+1*2^0+1*2^(-1)+0*2^(-2)+1*2^(-3)
  - Octave System: radix = 8 ∈ {0,1,2,3,4,5,6,7}
    - $(4334.56)_8$=4*8^3+3*8^2+3*8^1+4*8^0+5*8^(-1)+6*8^(-2)
  - **Hexadecimal System:** radix = 16 ∈ {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
    - $(23AB.4C)_{16}$=2*16^3+3*16^2+10*16^1+11*16^0+4*16^(-1)+12*16^(-2)

- **How to transform between decimal and binary representations?**



• Decimal –to– Binary Conversion

The Process : *Successive Division*

a) Divide the *Decimal Number* by 2; the remainder is the LSB of *Binary Number* .

b) If the quotation is zero, the conversion is complete; else repeat step (a) using the quotation as the Decimal Number. The new remainder is the **next most significant bit** of the *Binary Number*.

Example:

Convert the decimal number 6_{10} into its binary equivalent.

$$2\,\overline{)\,6}^{\;3} \quad r = 0 \leftarrow \text{Least Significant Bit}$$

$$2\,\overline{)\,3}^{\;1} \quad r = 1$$

$$2\,\overline{)\,1}^{\;0} \quad r = 1 \leftarrow \text{Most Significant Bit}$$

$$\therefore\; 6_{10} = 110_2$$

• Decimal –to– Binary Conversion

Example:

Convert the decimal number 26_{10} into its binary equivalent.

Solution:

$$2\,\overline{)\,26}^{\;13} \quad r = 0 \quad \leftarrow \text{LSB (Less Significant Bit)}$$

$$2\,\overline{)\,13}^{\;6} \quad r = 1$$

$$2\,\overline{)\,6}^{\;3} \quad r = 0$$

$$2\,\overline{)\,3}^{\;1} \quad r = 1$$

$$2\,\overline{)\,1}^{\;0} \quad r = 1 \quad \leftarrow \text{MSB (Most Significant Bit)}$$

$$\boxed{\therefore \;\; 26_{10} = 11010_2}$$

• Decimal –to– Binary Conversion

Example:

Convert the decimal number 0.8125_{10} into its binary equivalent.

Solution:

$0.8125\times 2=1.625 \quad r=1 \leftarrow$ MSB (Most Significant Bit)

$0.625\times 2=1.25 \quad r=1$

$0.25\times 2=0.5 \quad r=0$

$0.5\times 2=1 \quad r = 1 \leftarrow$ LSB (Less Significant Bit)

$\therefore \; 0.8125_{10} = 0.1101_2$

• Binary –to– Decimal Process

The Process : *Weighted Multiplication*

a) Multiply each bit of the *Binary Number* by it corresponding bit-weighting factor (i.e. Bit-0→2^0=1; Bit-1→2^1=2; Bit-2→2^2=4; etc).

b) Sum up all the products in step (a) to get the *Decimal Number*.

Example:

Convert the decimal number 0110_2 into its decimal equivalent.

0	1	1	0	
2^3	2^2	2^1	2^0	Bit-Weighting Factors
8	4	2	1	

$$0 + 4 + 2 + 0 = 6_{10}$$

$$\therefore\ 0110_2 = 6_{10}$$

• Binary –to– Decimal Process

Example:

Convert the binary number 10010_2 into its decimal equivalent.

Solution:

1		0		0		1		0		
2^4		2^3		2^2		2^1		2^0		
16		8		4		2		1		
16	+	0	+	0	+	2	+	0	=	18_{10}

$$\therefore 10010_2 = 18_{10}$$

Transformation between Decimal and Binary Representations:

Base_{10} DECIMAL → **Successive Division** → **Base_{2} BINARY**

a) Divide the *Decimal Number* by 2; the remainder is the LSB of *Binary Number* .

b) If the Quotient Zero, the conversion is complete; else repeat step (a) using the Quotient as the Decimal Number. The new remainder is the next most significant bit of the *Binary Number.*

Base_{2} BINARY → **Weighted Multiplication** → **Base_{10} DECIMAL**

a) Multiply each bit of the *Binary Number* by it corresponding bit-weighting factor (i.e. Bit-0→2^0=1; Bit-1→2^1=2; Bit-2→2^2=4; etc).

b) Sum up all the products in step (a) to get the *Decimal Number*.

Binary Arithmetic

- Binary addition
- Binary subtraction
- Binary multiplication
- Binary division
- Complements of Binary Numbers

  1's complements

  2's complements

Addition (decimal)

```
      1 1 1
      3 7 5 8
+     4 6 5 7
-------------
      8 4 1 5
```

What just happened?

```
   1  1  1      (carry)
   3  7  5  8
 + 4  6  5  7
 ------------
   8 14 11 15  (sum)
 -   10 10 10  (subtract the base)
 ------------
   8  4  1  5
```

So when the **sum** of a column is **equal to** or **greater than** the **base**, we subtract the **base** from the **sum**, record the **difference**, and carry **one** to the next column to the left.

Addition (binary)

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} {}^{1}1 \\ +\,1 \\ \hline 10 \end{array}$$

Addition (binary)

Rules:

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1 (just like in decimal)

- 1 + 1 $= 2_{10}$
  $= 10_2$ = 0 with 1 to carry

- 1 + 1 + 1 $= 3_{10}$
  $= 11_2$ = 1 with 1 to carry

Addition (binary)

$$
\begin{array}{r}
1111 \\
01101 \\
+01011 \\
\hline
11000
\end{array}
$$

Addition (binary)

Example 1: Add binary **110111** to **11100**

		1	1	1	1		
		1	1	0	1	1	1
+		0	1	1	1	0	0
	1	0	1	0	0	1	1

Col 1) Add **1 + 0 = 1**
Write **1**

Col 2) Add **1 + 0 = 1**
Write **1**

Col 3) Add **1 + 1 = 2** (**10** in binary)
Write **0**, carry **1**

Col 4) Add **1+ 0 + 1 = 2**
Write **0**, carry **1**

Col 5) Add **1 + 1 + 1 = 3** (**11** in binary)
Write **1**, carry **1**

Col 6) Add **1 + 1 + 0 = 2**
Write **0**, carry **1**

Col 7) Bring down the carried 1
Write **1**

Subtraction (Decimal)

Subtract
4657 from **8025**:

- ❑ Minuend
- ❑ Subtrahand
- ❑ Difference

$$\begin{array}{ccccc} & 7 & 9 & 11 & \\ & \cancel{8} & {}^{1}\cancel{0} & \cancel{2} & {}^{1}5 \\ - & 4 & 6 & 5 & 7 \\ \hline & 3 & 3 & 6 & 8 \end{array}$$

1) Try to subtract 5 – 7 → can't.
Must borrow 10 from next column.
Add the borrowed 10 to the original 5.
Then subtract 15 – 7 = 8**.**
2) Try to subtract 1 – 5 → can't.
Must borrow 10 from next column.
But next column is 0, so must go to column after next to borrow.

Add the borrowed 10 to the original 0.
Now you can borrow 10 from this column.

Add the borrowed 10 to the original 1..
Then subtract 11 – 5 = 6

3) Subtract 9 – 6 = 3

4) Subtract 7 – 4 = 3

Subtraction (Decimal Rules)

```
   8  0  2  5
 - 4  6  5  7
 ------------
   3  3  6  8
```

- So when you cannot subtract, you borrow from the column to the left.
  - The amount borrowed is **1 base unit**, which in decimal is **10**.
  - The 10 is added to the original column value, so you will be able to subtract.

Subtraction (Decimal Rules)

- In binary, the **base unit** is **2**
- So when you cannot subtract, you borrow from the column to the left.
  - The amount borrowed is **2**.
  - The 2 is added to the original column value, so you will be able to subtract.

Subtraction (Binary Example 1)

Example 1: Subtract binary **11100** from **110011**

$$
\begin{array}{cccccccc}
 & & 2 & 1 & & & \\
 & 0 & \cancel{0} & \cancel{2} & 2 & & \\
 & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{0} & 1 & 1 \\
- & & 1 & 1 & 1 & 0 & 0 \\
\hline
 & & 1 & 0 & 1 & 1 & 1
\end{array}
$$

Col 1) Subtract 1 – 0 = 1

Col 2) Subtract 1 – 0 = 1

Col 3) Try to subtract 0 – 1 → can't.
Must borrow 2 from next column.
But next column is 0, so must go to column after next to borrow.
Add the borrowed 2 to the 0 on the right.
Now you can borrow from this column (leaving 1 remaining).
Add the borrowed 2 to the original 0.
Then subtract 2 – 1 = 1

Col 4) Subtract 1 – 1 = 0

Col 5) Try to subtract 0 – 1 → can't.
Must borrow from next column.
Add the borrowed 2 to the remaining 0.
Then subtract 2 – 1 = 1

Col 6) Remaining leading 0 can be ignored.

Subtraction (Binary Example 2)

Example 2: Subtract binary **10100** from **101001**

$$
\begin{array}{rcccccc}
 & 0 & 2 & 0 & 2 & & \\
 & \cancel{1} & \cancel{0} & \cancel{1} & \cancel{0} & 0 & 1 \\
- & & 1 & 0 & 1 & 0 & 0 \\
\hline
 & & 1 & 0 & 1 & 0 & 1
\end{array}
$$

Verification

$$
\begin{array}{rlrl}
 & 101001_2 \rightarrow & & 41_{10} \\
- & \underline{\;10100_2} & - & \underline{20_{10}} \\
 & & & 21_{10}
\end{array}
$$

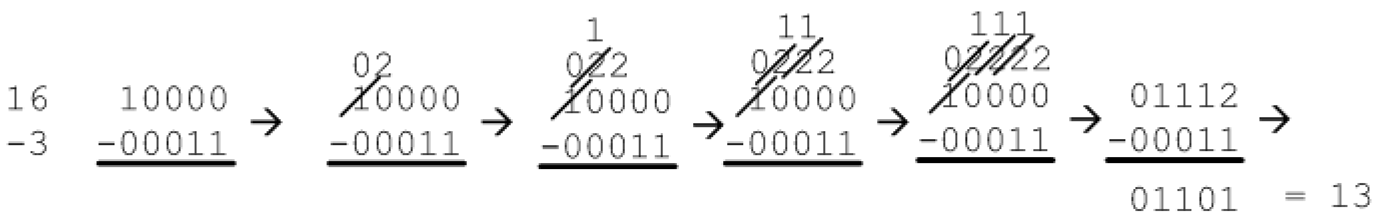
64	32	16	8	4	2	1
		1	0	1	0	1

$= 16 + 4 + 1$

$= 21_{10}$

Subtraction (Binary In General)

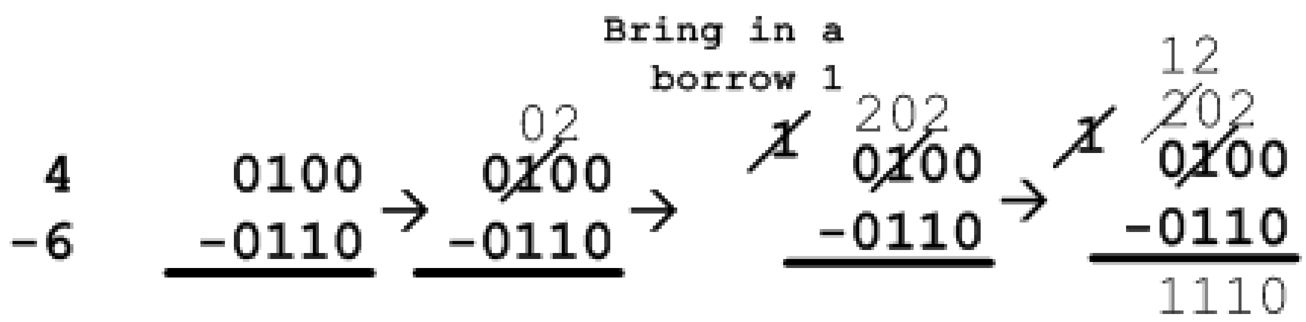
- When there is no borrow into the MSB position, then the subtrahend in not larger than the minuend and the result is positive and correct.



- If a borrow into the MSB does occur, then the subtrahend is larger than the minuend. Then the result is negative.

Subtraction (Consider another situation)

- Now do the operation 4 – 6



How to handle this?

- Correct difference is -2 or -0010
- Different because 2^n was brought in and made the operation M-N+2^n

Complements of Binary Numbers

- How do you represent a minus sign electronically in a computer?
- How can you represent it such that arithmetic operations are manageable?
- There are two types of complement for each number base system.
  - Have the **r's complement**
  - Have the **(r-1)'s complement**
- For base 2 have **2's complement** and **1's complement**

1's Complement （反码）

- 1's complement of N is defined as **(2^n-1)-N**.
  - If n=4 have (2^n-1) being 1 0000 - 1 = 1111
- So for n=4 would subtract any 4-bit binary number from 1111.
- This is just inverting each bit.
- Example: 1's compliment of 1011001
- is 0100110

2’s complement （补码）

- The 2’s complement is defined as 2^n-N
- Can be done by subtraction of N from 2^n or adding 1 to the 1’s complement of a number.
- For 6 = 0110
  - The 1’s complement of (-6) is 1 1001
  - The 2’s complement of (-6) is 1 1010
- For 2 = 0010
  - The 1’s complement of (-2) is 1 1101
  - The 2’s complement of (-2) is 1 1110

	Sign	2’s complement
4	0	0100
-6	1	1010
-2	1	1110

Thank You!

Q&A