Binary Numbers

CS207 Chapter 1

James YU
yujq3@sustech.edu.cn

Department of Computer Science and Engineering
Southern University of Science and Technology

Sept. 7, 2022

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Digital systems

- All things computing have a special-purpose digital computer embedded.
- *Digital system*s represent/manipulate discrete elements of information.
  - 10 decimal digits;
  - 26 letters of alphabet.
- Digit $\rightarrow$ Digital (computer, system, etc.)

~ ideas or things are separate and distinct from each other

where firmly & deeply it becomes ~ itself If an object ~ itself in a substance

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Digital systems

- In a digital system, input is given with the help of switches.
  - Usually have two distinct discrete levels or values: `HIGH` and `LOW`.

Discrete ideas or things are separate and distinct from each other.

- Such signals are called *digital signal*s and the circuit within the device is called a *digital circuit*.
- Digital circuits find applications in computers, telephony, radar navigation, data processing, and many other applications.
  - We first learn the general properties of number systems.

I Number systems 计数系统

- There are several number systems which we normally use:
  - *Decimal*: 0, 1, 2, ..., 9;
  - *Binary*: 0, 1;
  - *Octal*: 0, 1, 2, ..., 7;
  - *Hexadecimal*: 0, 1, 2, ..., 9, A, B, ..., F.

oct: things related to 8
hex: relate to 6
hexa: relate to 16

- With a decimal system, we have 10 different digits, but only 2 in a binary system.
  - Binary number system is easier to be dealt with.

NOT F ! 10: from 0 to 9

- In a digital world, we think in binary
  - A light is either *off* or *on*.
- ...and we use two digits to express everything: 0 and 1.
  - A decimal 25_{10} becomes 11001_2 in binary.

Number systems

- In general, we can express any number in any base or radix "X".
- Any number with base X, having n digits to the left and m digits to the right of the decimal point, can be expressed as

$$a_n X^{n-1} + a_{n-1} X^{n-2} + a_{n-2} X^{n-3} + \cdots + a_2 X^1 + a_1 X^0 + b_1 X^{-1} + b_2 X^{-2} + \cdots + b_m X^{-m}$$

for $(a_n a_{n-1} \ldots a_2 a_1 b_1 b_2 \ldots b_m)_X$.

- For example,

$$(4021.2)_5 = 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} = (511.4)_{10}.$$

Conversion between number systems

- Decimal numbers are the key
  - Base m to base $n \rightarrow$ base m to decimal to base n.
- Base m to decimal:

$$a_n X^{n-1} + a_{n-1} X^{n-2} + a_{n-2} X^{n-3} + \cdots + a_2 X^1 + a_1 X^0 + b_1 X^{-1} + b_2 X^{-2} + \cdots + b_m X^{-m}$$

- Decimal to base n?
  - We use multiplication for base m to decimal.
  - The inverse of multiplication is division.

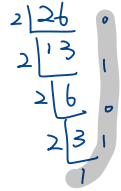
Conversion between number systems

- Example: convert 26_{10} into a binary number.

Way to convert decimal numbers to other number systems

the result of division

Division	Quotient	Remainder
$26/2$	13	0
$13/2$	6	1
$6/2$	3	0
$3/2$	1	1
$1/2$	0	1



from back to start

$\Rightarrow 11010$

- The converted binary number is 11010_2.

Conversion between number systems

- Example: convert 348_{10} into a hexadecimal number.

Division	Quotient	Remainder
$348/16$	21	$12 = C$
$21/16$	1	5
$1/16$	0	1

?? 12=B
10→A
11→B
12→C

- The converted binary number is $15C_{16}$.

Don't forget the index

Conversion between number systems

- For fraction, the computation is reversed again
- Example: convert 25.625_{10} into a binary number.

A fraction of something is a tiny amount or proportion of it.
A fraction is a number that can be expressed as a proportion of two whole numbers.

Division	Quotient	Remainder
$25/2$	12	1
$12/2$	6	0
$6/2$	3	0
$3/2$	1	1
$1/2$	0	1

When someone or something reverses a decision, policy, or trend, they change it to the opposite decision, policy, or trend.

- Therefore, $25_{10} = 11001_2$.
- fraction parts
  $0.625 \times 2 = 1.250, 0.250 \times 2 = 0.500, 0.500 \times 2 = 1.000$.
- Therefore, $(25.625)_{10} = (11001.101)_2$.

END when fraction part is 0 ⟷ NOT ZERO

$0.101101\ldots = 0.101$ $\quad 0.101111\ldots = 0.101$

在二进制中，0.1+0.1=1，即满2进1.
所以，在十进制中给小数部乘以2后有进位，则代表该数
在2进制表示中有 0.1_2，对于后面位乘了2次二，等似进位2次到这个位，即 $0.01_2 + 0.01_2 = 0.1_2$ $\quad 0.1_2 + 0.1_2 = 1_2$

Conversion between binary and octal

based on the number eight

consist of 0,1 use 3 number

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- The maximum digit in an octal number system is 7.
  - Represented as 111_2 in a binary system
- Starting from the LSB, we group three digits at a time and replace them by the octal equivalent of those groups.

the least significant bit ← odd / even

- Example: convert 101101010_2 into an octal number.

NOT use decimal as bridge

Starting with LSB and grouping 3 bits	101	101	010
Octal equivalent	5	5	2

least significant bite

- The octal number is 552_8.
- Example: convert 1011110_2 into an octal number.

add two zero without change the number

Starting with LSB and grouping 3 bits	001	011	110
Octal equivalent	1	3	6

Conversion between binary and hexadecimal

- Trivially.
- 显然。

Complements

- When human do subtraction, we use "borrow" to borrow a 1 from a higher significant position.
  - What if the position does not want to lend?
- It is hard for circuits to design "borrow". So we use complements to implement subtraction.
- For each number system of base r, two types of complements:
  - r's complement;
  - $r-1$'s complement.
- For a binary system: 2's complement and 1's complement.

If you implement something such as a plan, you ensure that what has been planned is done.

If one thing complements another, it goes well with the other thing and makes its good qualities more noticeable.

If people or things complement each other, they are different or do something different, which makes them a good combination.

补码在数字计算机中用于简化减法运算和逻辑操作，从而使电路更加简单、价格更加便宜。每个 r 进制系统都有两种类型的补码：补码（radix complement）和反码（diminished radix complement）。第一种可看作 r 的补码，第二种为（$r-1$）的补码。这里讨论二进制数的 2 的补码和 1 的补码以及十进制数的 10 的补码和 9 的补码。

Complements

When something diminishes, or when something diminishes it, it becomes reduced in size, importance, or intensity.

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

decimal : r=10

I
- $r-1$'s complement: *diminished radix complement*. Use $r-1$ minus each digit:
  - The 9's complement of 546700 is $999999 - 546700 = 453299$.
  - The 9's complement of 012398 is $999999 - 012398 = 987601$.

A digit is a written symbol for any of the ten numbers from 0 to 9.

+1

II
- r's complement: *radix complement*.
- Calculate the diminished raid complement, then plus one:
  - The 10's complement of 546700 is $999999 - 546700 + 1 = 453300$.
  - The 10's complement of 012398 is $999999 - 012398 + 1 = 987602$.

III
- Another way: use r minus the least significant non-zero digit, and $r-1$ minus digits on the left:
  - The least significant non-zero digit of 546700 is 7: $10 - 7 = 3$;
  - Digits on the left are 546: $999 - 546 = 453$;
  - The 10's complement of 546700 is $453\ 3\ 00$.

MORE commonly used

look from left to right ① ②

r's complement

Binary subtraction

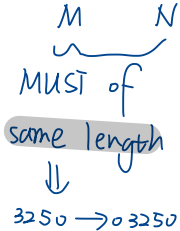
- Three ways:
  - The direct "borrow" method;
  - The r's complement method;
  - The $r-1$'s complement method
- We discuss the r's complement method in this course.

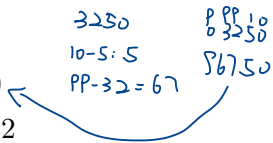
Binary subtraction

- Subtraction $M - N$, if $M \geq N$:
  - Add M to r's complement of N then discard the end carry.
- Example: $72532 - 3250$

If you discard something, you get rid of it because you no longer want it or need it.

$$
\begin{aligned}
M &= 72532 \\
10\text{'s complement of } N &= +96750 \\
\text{Sum} &= 169282 \\
\text{Discard end carry} &= -100000 \\
&= 69282
\end{aligned}
$$





Binary subtraction

- Subtraction $M - N$ if $M < N$:
  1. Add M to r's complement of N,
  2. then take an r's complement,
  3. then add a negative sign.
- Example: $3250 - 72532$

$M - N = -\left[M + (10^5 - N) - 10^5\right]$

$M > N$ / $M < N$ of N, NOT depends on always calculate complement

	M =	03250
1	10's complement of N =	+ 27468
2	$M + (10^5 - N)$ Sum =	30718
3	10's complement =	69282
4	Add a negative sign =	− 69282

computers do NOT know sign ⟹ 1/0

Signed binary numbers

- In real life one may have to face a situation where both positive and negative numbers may arise.
  - We have $+$ and $-$.
  - Digital systems represent everything with binary digits.
- Three types of representations of signed binary numbers:
  - Sign-magnitude representation;
  - 1's complement representation;
  - 2's complement representation.

NOT use "3 kinds"

signed number

1000 → 0001

1000 → 0001 $\xrightarrow{+1}$ 0010

used for subtraction

1000 $\xrightarrow{8\text{'s}-}$ 8'sc

1000 $\xrightarrow{1\text{'s}-}$ 8'sc $\xrightarrow{+1}$ 8'sc

= 1000 $\xrightarrow{10000-}$ 8'sc

Sign-magnitude representation

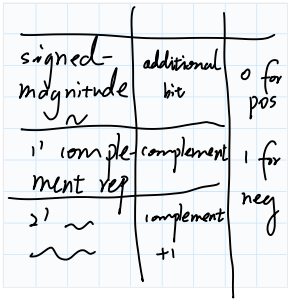
the bit that has the largest value in a multi-bit binary number

most significant byte

- An additional bit is used as the *sign bit*, usually placed as the MSB.
  - Generally a 0 is reserved for a positive number and a 1 is reserved for a negative number.
  - Example: an 8-bit signed binary number 01101001 represents a **positive** number whose magnitude is $1101001_2 = 105_{10}$.
  - Example: an 8-bit signed binary number 11101001 represents a **negative** number whose magnitude is $1101001_2 = 105_{10}$, i.e., -105.

1's and 2's complement representation

- In 1's complement representation, both numbers are a complement of each other.
  - Example: 0111_2 represents $+7_{10}$ and 1000_2 represents -7_{10}.
  - Also, MSB 0 for positive numbers and 1 for negative numbers.
- In 2's complement representation, 1 is added to 1's complement representation.
  - Example: 0110_2 represents $+6_{10}$ and 1010_2 represents -6_{10}.
  - Also, MSB 0 for positive numbers and 1 for negative numbers.



Signed binary numbers

most commonly used

Decimal	2's complement representation	1's complement representation	Sign-magnitude representation
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	−	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011

should be same

⇓

advantage of 2's ..

Blue: sign(not change)
Red: complement/magnitude(change)

Binary codes

- Computers and other digital circuits process data in binary format.
- The interpretation of the data is only possible if the code in which the data is being represented is known.
  - Example: 1000010 represents 66 (decimal) in straight binary, 42 (decimal) in BCD, and letter B in ASCII code.

different way to translate

Binary-Coded Decimal

Binary-coded decimal (8421)

- The full form of BCD is "Binary-Coded Decimal".
- Four bits are required to code each decimal number.
  - Example: 35_{10} is represented as $0011\ 0101$ using BCD code, rather than 100011_2.
  - It is convenient to use BCD for input and output in digital systems.
- Also known as 8-4-2-1 code, as 8, 4, 2, and 1 are the weights of the four bids of BCD.
- Example: Give the BCD equivalent for the decimal number 69.27.

The decimal number	6	9	.	2	7
BCD code is	0110	1001	.	0010	0111

- Therefore, $(69.27)_{10} = (01101001.00100111)_{\text{BCD}}$.

BCD addition

- There are certain rules to be followed in BCD addition as given below.
  - First add the two numbers using normal rules for binary addition.
  - If the 4-bit sum is equal to or less than 9, it becomes a valid BCD number.
  - If the 4-bit sum is greater than 9, or if a carry-out of the group is generated, it is an invalid result.
    - In such a case, add 0110_2 or 6_{10} to the 4-bit sum in order to skip the six invalid states and return the code to BCD. If a carry results when 6 is added, add the carry to the next 4-bit group.
- Example: $0111_{\text{BCD}} + 1001_{\text{BCD}}$:

$$
\begin{array}{rl}
 & 0111 \\
+ & 1001 \\
\hline
 & 10000 \rightarrow \text{Invalid BCD number} \\
+ & 0110 \rightarrow \text{Add 6} \\
\hline
 & 0001\ \ 0110 \rightarrow \text{Valid BCD number}
\end{array}
$$

Four binary digits count up to 15 (1111) but in BCD we only use the representations up to 9 (1001). The difference between 15 and 9 is 6. If you want 9+1 to produce 10, which is 1 0000, you have to add 6 to make 1010 wrap to 1 0000.

BCD addition

- Example: $10010010_{\text{BCD}} + 01011000_{\text{BCD}}$:

$$
\begin{array}{rrl}
1001 & 0010 & \\
+0101 & 1000 & \\
1110 & 1010 & \rightarrow \text{Both groups are invalid} \\
+0110 & +0110 & \rightarrow \text{Add 6} \\
0001 & 0101 & 0000 \rightarrow \text{Valid BCD number}
\end{array}
$$

BCD subtraction

- Example: $768_{10} - 274_{10}$:

$$
\begin{array}{rrrl}
0111 & 0110 & 1000 & \\
+0111 & 0010 & 0110 & \\
\hline
1110 & 1000 & 1110 & \rightarrow \text{Left and right groups are invalid} \\
+0110 & 0000 & 0110 & \rightarrow \text{Add } 6 = 0110 \\
\hline
0100 & 1001 & 0100 & \rightarrow \text{Ignore carry}
\end{array}
$$

- The final result is $010010010100_{\text{BCD}}$ or 494_{10}.

When using complement to calculate the subtraction, if a carry appeared, it indicates that m-n>0, so we can just ignore the carry, then other part of the result is the final result

Gray code

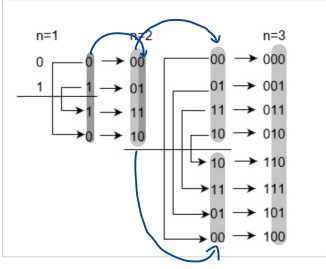
- Gray code belongs to a class of code known as minimum change code.
  - A number changes by only one bit as it proceeds from one number to the next.

Gray Code	Decimal
000	0
001	1
011	2
010	3
110	4
111	5
101	6
100	7

Gray code

- *n*-bit Gray code can be generated recursively using reflect and prefix method which is explained as following below.
  - Generate code for n=1: 0 and 1 code.
  - Take previous code in sequence: 0 and 1.
  - Add reversed codes in the following list: 0, 1, 1 and 0.
  - Now add prefix 0 for original previous code and prefix 1 for new generated code: 00, 01, 11, and 10.

Therefore, Gray code 0 and 1 are for Binary number 0 and 1 respectively. Gray codes: 00. 01, 11, and 10 are for Binary numbers: 00, 01, 10, and 11 respectively. Similarly you can construct Gray code for 3 bit binary numbers:



① reflect: numbers below & above horizontal lines

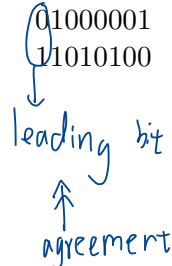
② prefix: MSD is fixed. MSD of 1st half of 2^n is 0.

Basically, binary code is changed to gray equivalent in order to lessen the switching operations. As only a single bit is changed at a particular time duration this leads to a reduction in switching from one bit to another.

Error-detection codes

- Binary information may be transmitted through some form of communication medium such as wires or radio waves or fiber optic cables, etc.
  - Any external noise introduced into a physical communication medium changes bit values from 0 to 1 or vice versa.
- An *error detection code* can be used to detect errors during transmission.
  - The detected error cannot be corrected, but its presence is indicated.
  - *Parity bit*

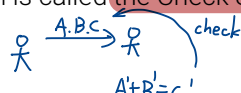
	With even parity	With odd parity
1000001	01000001	11000001
1010100	11010100	01010100



Checksum

eg. 10110010 → 10100010
2 bits are changed

- Parity bit technique fails for double errors.
- *Checksum* adds all transmitted bytes and transmit the result as an error-detection code.
- Example: initially any word A 10010011 is transmitted; next another word B 01110110 is transmitted.
  1. The binary digits in the two words are added and the sum obtained is retained in the transmitter.
  2. Then any other word C is transmitted and added to the previous sum retained in the transmitter and the new sum is now retained.
  3. After transmitting all the words, the final sum, which is called the Check Sum, is also transmitted.
  4. The same operation is done at the receiving end.
  5. There is no error if the two sums are equal.



ASCII

- Many applications of the computer require not only handling of numbers, but also of letters.
- To represent letters it is necessary to have a binary code for the alphabet.
- American Standard Code for Information Interchange (ASCII)
  - Seven bits to code 128 characters.

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	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	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	–	o	DEL

Binary storage and registers

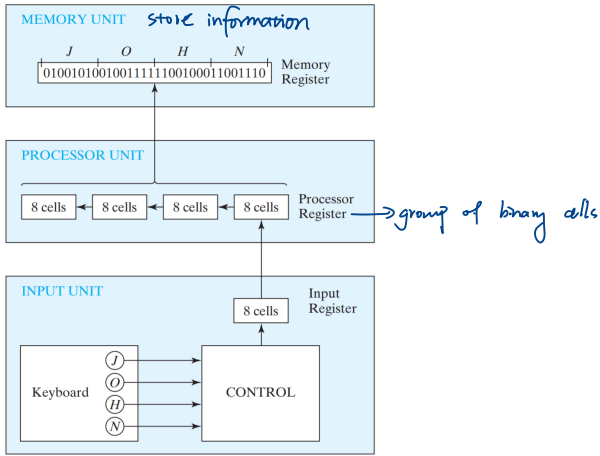
- Binary information must have a physical existence.
- Binary *cell* (0 or 1): two stable states; a light bulb switch
- *Register*: a group of binary cells. a register as 16 binary cells
  - A 16-bit register: `1100 0011 1100 1001`.
  - Assume it is a binary integer value: 50121.
  - Assume it is two ASCII characters with even parity: `CI`.
  - The same binary storage means different interpretation, depending on the application.

⇒ first tell computer how to interpreting

⇒ in Java why indicate the type of a variable first

Binary storage and registers

- Register transfer to move binary storages in between.



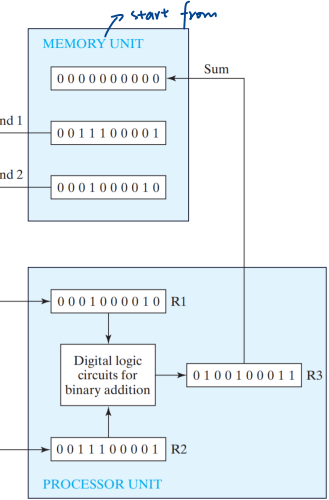
Binary storage and registers

- Register transfer to move binary storages in between.

a quantity or function upon which a mathematical or logical operation is performed

operator : 算子

operand : 运算数(被)
操作数(计)



Binary logic

- Binary logic deals with variables that take on two discrete values and with logical operations.
- Three basic logical operations: ⟷ 4
  - **AND**: $x \cdot y = z$ or $xy = z$. like multiple
  - **OR**: $x + y = z$.
  - **NOT**: $x' = z$

1: true
0: false

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

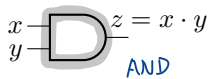
var
||
prime notation

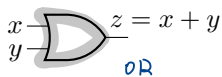
NOT

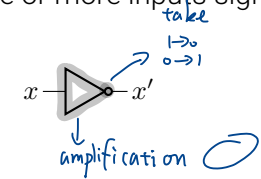
x	x'
0	1
1	0

Binary logic

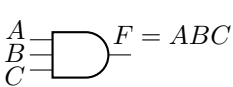
- Logic gates are electronic circuits that operates on one or more inputs signals to produce an output.

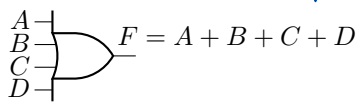






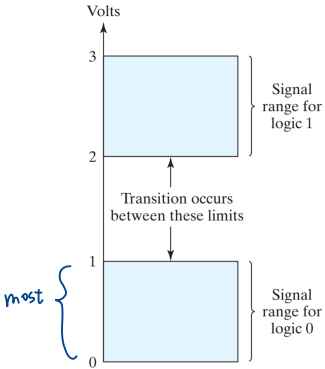
- It is fine to have more than two inputs for AND/OR.

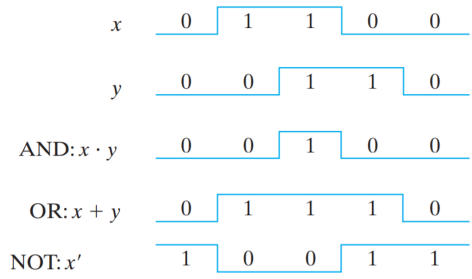




Binary logic

- Voltage-operated, though on a range, interpreted to be either of the two values.





Notices

- The lab session will start from today. Attendance is required.
- Sakai site:
  `https://sakai.sustech.edu.cn/portal/directtool/39d1a621-2c43-46f7-90af-fcf9d1360b64/`

Quiz: Sept. 14, 2022

1. Convert the following numbers with the indicated bases to decimal:
   - $(4310)_5$
   - $(198)_{12}$

   2为0 ↑ second value

2. Perform subtraction on the given unsigned numbers using the 10's complement of the subtrahend. Where the result should be negative, find its 10's complement and affix a minus sign. Verify your answers.
   - $4637 - 2579$
   - $125 - 1800$ ⇒ minus number 0125 + 8200 = 8325 → 1675 → -1675

3. Formulate a weighted binary code for the decimal digits, using the following weights:
   - 6,3,2,1
   - 6,4,2,1

	6 4 2 1	6 3 2 1
	× × × ×	~~2 2 2 2~~
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
3	0 0 1 1	0 0 1 1 / 0 1 0 0
4	0 1 0 0	0 1 0 1
6		1 0 0 0 / 0 1 1 1

both is ok