Binary Numbers

CS207 Chapter 1

James YU
yujq3@sustech.edu.cn

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

Jun. 21, 2021

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Digital systems

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- 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.)

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`.
- Such signals are called *digital signals* 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.

Number systems

- 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.
- 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.
- 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

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

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

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

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:

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

- 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.

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

- 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
$21/16$	1	5
$1/16$	0	1

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

Conversion between number systems

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

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

- Therefore, $25_{10} = 11001_2$.
- $0.625 \times 2 = \mathbf{1}.250$, $0.250 \times 2 = \mathbf{0}.500$, $0.500 \times 2 = \mathbf{1}.000$.
- Therefore, $(25.625)_{10} = (11001.101)_2$.

Conversion between binary and octal

- 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.
- Example: convert 101101010_2 into an octal number.

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

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

Complements

- $r-1$'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 $999999-012398=987601$.
- 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 $999999-012398+1=987602$.
- 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 $453\ 3\ 00$.

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: $72532 - 3250$

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

Binary subtraction

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

$$\begin{aligned} M &= \quad 03250 \\ \text{10's complement of } N &= +\,27468 \\ \text{Sum} &= \quad 30718 \\ \text{10's complement} &= \quad 69282 \\ \text{Add a negative sign} &= -\,69282 \end{aligned}$$

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.

Sign-magnitude representation

- 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

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- 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

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

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.

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, $(69.27)_{10} = (01101001.00100111)_{\mathrm{BCD}}$.

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: $0111_{\text{BCD}} + 1001_{\text{BCD}}$:

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

BCD addition

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

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

BCD subtraction

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

$$
\begin{array}{rrrrl}
 & 0111 & 0110 & 1000 & \\
 & +0111 & 0010 & 0110 & \\
 & 1110 & 1000 & 1110 & \rightarrow \text{Left and right groups are invalid} \\
 & +0110 & 0000 & 0110 & \rightarrow \text{Add } 6 \\
1 & 0100 & 1001 & 0100 & \rightarrow \text{Ignore carry}
\end{array}
$$

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

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

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

- 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.

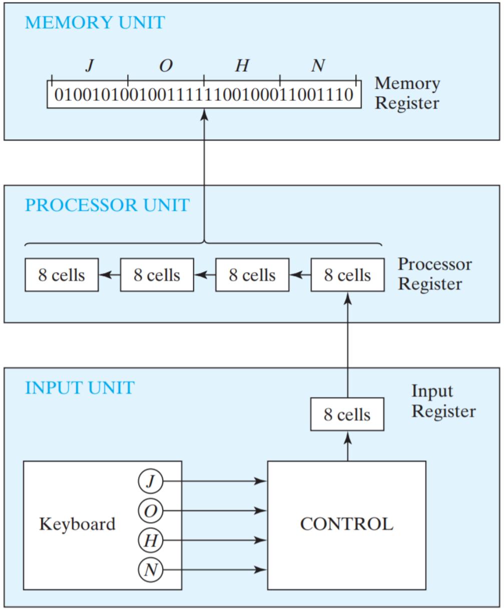
	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	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;
- *Register*: a group of 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.

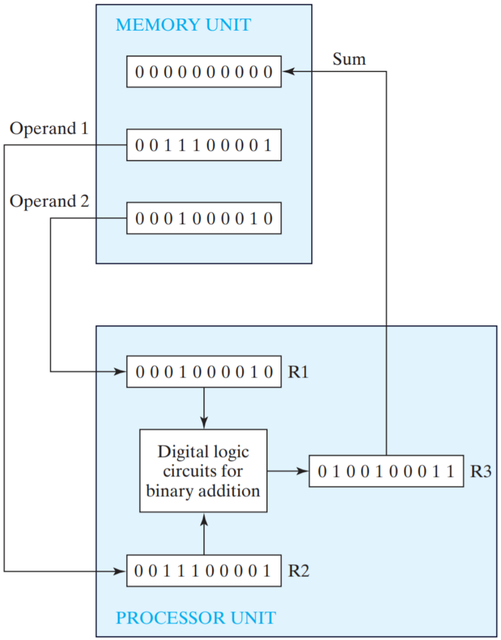
Binary storage and registers

- Register transfer to move binary storages in between.



Binary storage and registers

- Register transfer to move binary storages in between.



Binary logic

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

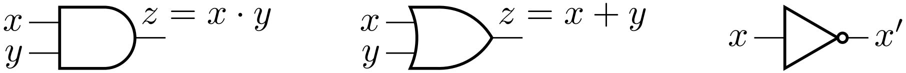
AND		
x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

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

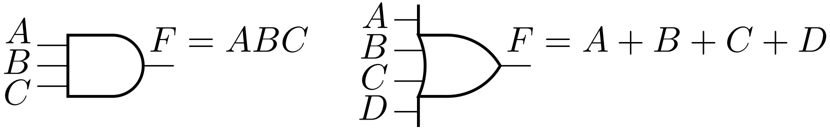
NOT	
x	x'
0	1
1	0

Binary logic

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

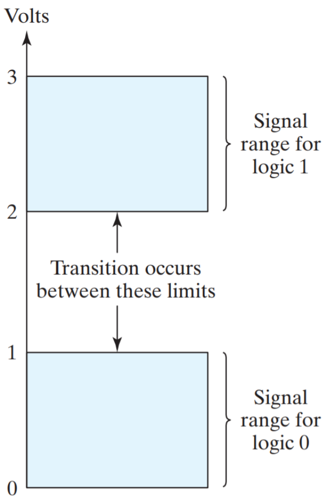


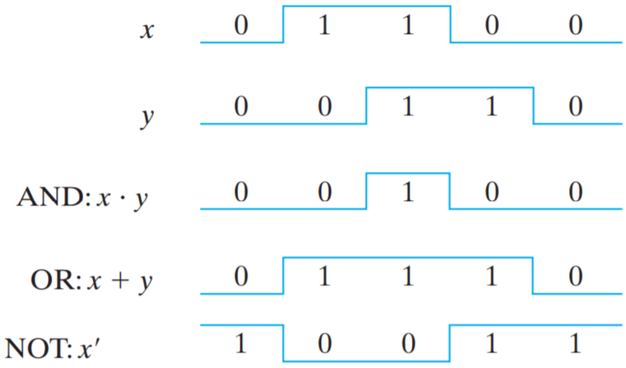
- It is fine to have more than two inputs for AND/OR.



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/`