

---

# Boolean Algebra

Philipp Koehn

30 August 2019



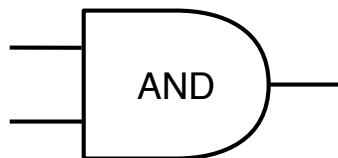
# Core Boolean Operators



**AND**

$\wedge$

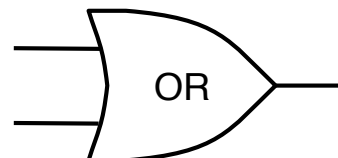
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1



**OR**

$\vee$

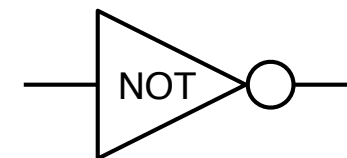
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



**NOT**

$\neg$

A	NOT A
0	1
1	0



# from Boolean expressions to circuits

# Truth Table $\rightarrow$ Boolean Expression



- Truth table

A	B	OUT
0	0	1
0	1	0
1	0	0
1	1	0

- Operation:

# Truth Table $\rightarrow$ Boolean Expression



- Truth table

A	B	OUT
0	0	1
0	1	0
1	0	0
1	1	0

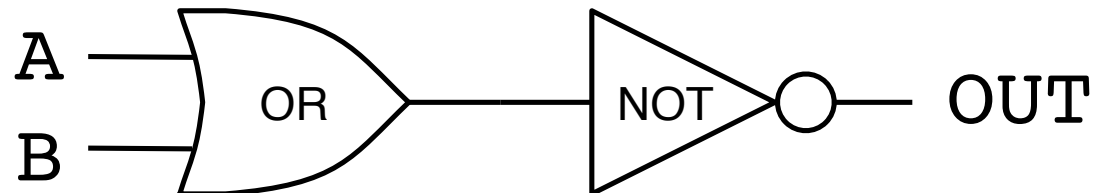
- Operation: `NOT ( A OR B )`  
(also called `NOR`)

# Boolean Expression $\rightarrow$ Circuit



- Operation: `NOT ( A OR B )`

- Circuit:



# 4-Bit AND



5

- 4 inputs (A, B, C, D), output 1 iff all inputs are 1
- Operation:

# 4-Bit AND



5

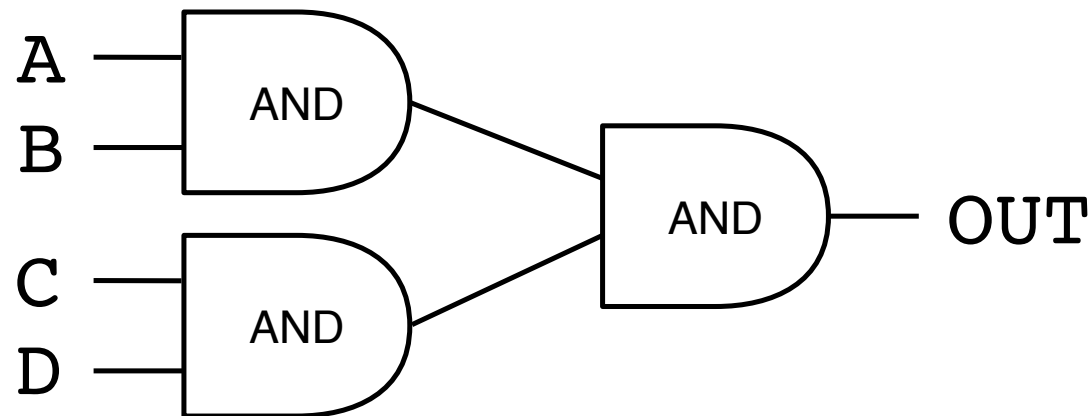
- 4 inputs (A, B, C, D), output 1 iff all inputs are 1
- Operation:  $(A \text{ AND } B) \text{ AND } (C \text{ AND } D)$
- Circuit:



# 4-Bit AND



- 4 inputs (A, B, C, D), output 1 iff all inputs are 1
- Operation:  $(A \text{ AND } B) \text{ AND } (C \text{ AND } D)$
- Circuit:



# 1-Bit Selector



- Truth table

A	OUT1	OUT2
0	1	0
1	0	1

- Operation:

# 1-Bit Selector



- Truth table

A	OUT1	OUT2
0	1	0
1	0	1

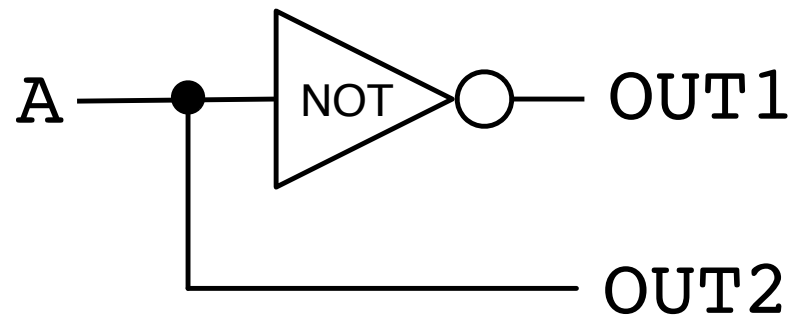
- Operation:  $\text{OUT1} = \text{NOT } A$   
 $\text{OUT2} = A$

# 1-Bit Selector



- Operation:  $\text{OUT1} = \text{NOT } A$   
 $\text{OUT2} = A$

- Circuit:



# A Complicated Example

- Truth table

A	B	C	OUT
0	0	0	0
0	1	0	1
1	0	0	0
1	1	0	0
0	0	1	1
0	1	1	1
1	0	1	0
1	1	1	0

- Operation:

# A Complicated Example

- Truth table

A	B	C	OUT
0	0	0	0
0	1	0	1
1	0	0	0
1	1	0	0
0	0	1	1
0	1	1	1
1	0	1	0
1	1	1	0

- Operation: Need a better way of doing this instead of relying on intuition

# disjunctive normal form

# DNF: Setup

A	B	C	OUT	Expression
0	0	0	0	
0	1	0	1	
1	0	0	0	
1	1	0	0	
0	0	1	1	
0	1	1	1	
1	0	1	0	
1	1	1	0	

Goal: find expression for each row that yields 1



## DNF: One Row

11



A	B	C	OUT	Expression
0	0	0	0	(NOT A) AND B AND (NOT C)
0	1	0	1	
1	0	0	0	
1	1	0	0	
0	0	1	1	
0	1	1	1	
1	0	1	0	
1	1	1	0	

Expression is 1 only for this row, 0 for all others

# DNF: All Rows

12



A	B	C	OUT	Expression
0	0	0	0	
0	1	0	1	(NOT A) AND B AND (NOT C)
1	0	0	0	
1	1	0	0	
0	0	1	1	(NOT A) AND (NOT B) AND C
0	1	1	1	(NOT A) AND B AND C
1	0	1	0	
1	1	1	0	

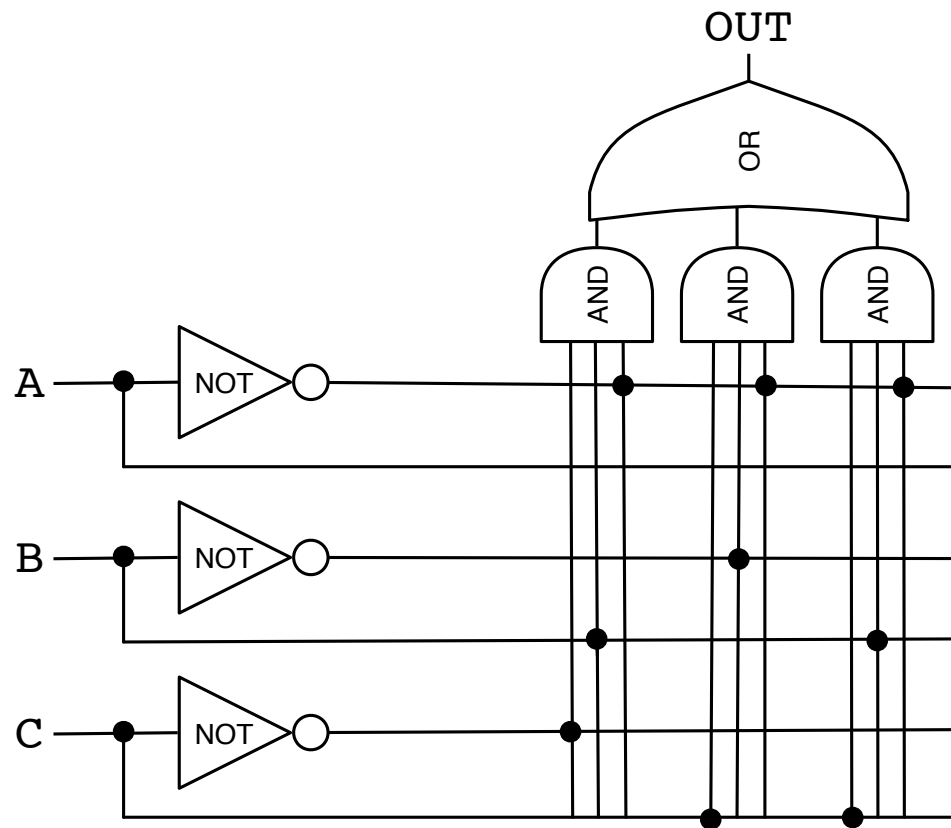
# DNF: Complete Operation

A	B	C	OUT	Expression
0	0	0	0	
0	1	0	1	(NOT A) AND B AND (NOT C)
1	0	0	0	
1	1	0	0	
0	0	1	1	(NOT A) AND (NOT B) AND C
0	1	1	1	(NOT A) AND B AND C
1	0	1	0	
1	1	1	0	

Putting it all together:  $((\text{NOT } A) \text{ AND } B \text{ AND } (\text{NOT } C)) \text{ OR}$   
 $((\text{NOT } A) \text{ AND } (\text{NOT } B) \text{ AND } C) \text{ OR}$   
 $((\text{NOT } A) \text{ AND } B \text{ AND } C)$

# DNF: Circuit

- Operation:  
 $((\text{NOT } A) \text{ AND } B \text{ AND } (\text{NOT } C)) \text{ OR}$   
 $((\text{NOT } A) \text{ AND } (\text{NOT } B) \text{ AND } C) \text{ OR}$   
 $((\text{NOT } A) \text{ AND } B \text{ AND } C)$
- Circuit:





# conjunctive normal form

# DNF

16



A	B	C	OUT	Expression
0	0	0	0	
0	1	0	1	(NOT A) AND B AND (NOT C)
1	0	0	0	
1	1	0	0	
0	0	1	1	(NOT A) AND (NOT B) AND C
0	1	1	1	(NOT A) AND B AND C
1	0	1	0	
1	1	1	0	

Putting it all together: ((NOT A) AND B AND (NOT C)) OR  
((NOT A) AND (NOT B) AND C) OR  
((NOT A) AND B AND C)

## CNF: One Row

17



A	B	C	OUT	Expression
0	0	0	0	NOT ((NOT A) AND (NOT B) AND (NOT C))
0	1	0	1	
1	0	0	0	
1	1	0	0	
0	0	1	1	
0	1	1	1	
1	0	1	0	
1	1	1	0	

Expression is 0 only for this row, 1 for all others

# CNF: All Rows

18



A	B	C	OUT	Expression
0	0	0	0	NOT ((NOT A) AND (NOT B) AND (NOT C))
0	1	0	1	
1	0	0	0	NOT (A AND (NOT B) AND (NOT C))
1	1	0	0	NOT (A AND B AND (NOT C))
0	0	1	1	
0	1	1	1	
1	0	1	0	NOT (A AND (NOT B) AND C)
1	1	1	0	NOT (A AND B AND C)



# CNF: Complete Operation

A	B	C	OUT	Expression
0	0	0	0	NOT ((NOT A) AND (NOT B) AND (NOT C))
0	1	0	1	
1	0	0	0	NOT (A AND (NOT B) AND (NOT C))
1	1	0	0	NOT (A AND B AND (NOT C))
0	0	1	1	
0	1	1	1	
1	0	1	0	NOT (A AND (NOT B) AND C)
1	1	1	0	NOT (A AND B AND C)

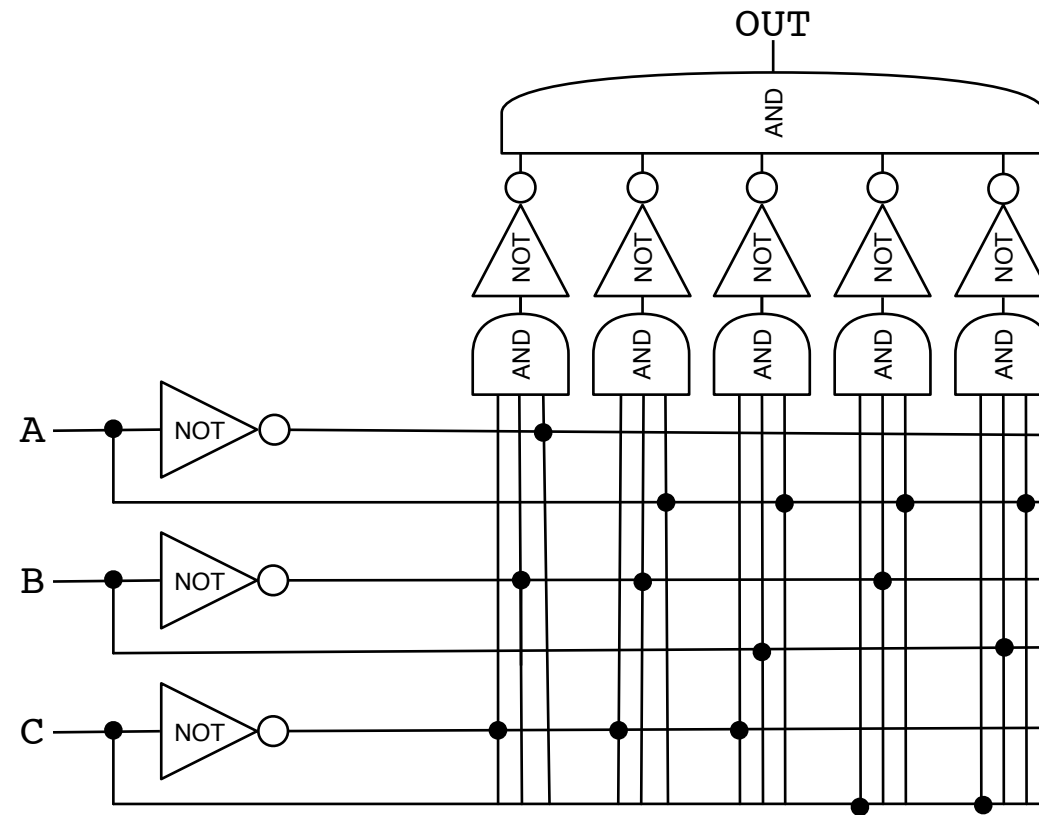
Putting it all together:  $(\text{NOT } ((\text{NOT } A) \text{ AND } (\text{NOT } B) \text{ AND } (\text{NOT } C))) \text{ AND}$   
 $(\text{NOT } (A \text{ AND } (\text{NOT } B) \text{ AND } (\text{NOT } C))) \text{ AND}$   
 $(\text{NOT } (A \text{ AND } B \text{ AND } (\text{NOT } C))) \text{ AND}$   
 $(\text{NOT } (A \text{ AND } (\text{NOT } B) \text{ AND } C)) \text{ AND}$   
 $(\text{NOT } (A \text{ AND } B \text{ AND } C))$

# CNF: Circuit

- Operation:

$(\text{NOT } ((\text{NOT } A) \text{ AND } (\text{NOT } B) \text{ AND } (\text{NOT } C))) \text{ AND}$   
 $(\text{NOT } (A \text{ AND } (\text{NOT } B) \text{ AND } (\text{NOT } C))) \text{ AND}$   
 $(\text{NOT } (A \text{ AND } B \text{ AND } (\text{NOT } C))) \text{ AND}$   
 $(\text{NOT } (A \text{ AND } (\text{NOT } B) \text{ AND } C)) \text{ AND}$   
 $(\text{NOT } (A \text{ AND } B \text{ AND } C))$

- Circuit:





# universal gates

# Universality of NAND

- Truth table:

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

- NOT:

# Universality of NAND

- Truth table:

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

- NOT:  $A \text{ NAND } A$
- AND:

# Universality of NAND

- Truth table:

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

- NOT:  $A \text{ NAND } A$
- AND:  $(A \text{ NAND } B) \text{ NAND } (A \text{ NAND } B)$
- OR:

# Universality of NAND

- Truth table:

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

- NOT:  $A \text{ NAND } A$
- AND:  $(A \text{ NAND } B) \text{ NAND } (A \text{ NAND } B)$
- OR:  $(A \text{ NAND } A) \text{ NAND } (B \text{ NAND } B)$

# Universality of NOR

- Truth table:

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

- NOT:



# Universality of NOR

- Truth table:

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

- NOT:  $A \text{ NOR } A$
- AND:

# Universality of NOR

- Truth table:

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

- NOT:  $A \text{ NOR } A$
- AND:  $(A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B)$
- OR:

# Universality of NOR

- Truth table:

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

- NOT:  $A \text{ NOR } A$
- AND:  $(A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B)$
- OR:  $(A \text{ NOR } B) \text{ NOR } (A \text{ NOR } B)$



There are only 10 kinds of people.  
Those who understand binary  
and those who don't.

# Roman Numerals

25



- Basic units

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

# Roman Numerals

- Basic units

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

- Additive combination of units

II   III   VI   XVI   XXXIII   MDCLXVI   MMXVI

# Roman Numerals

- Basic units

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

- Additive combination of units

II	III	VI	XVI	XXXIII	MDCLXVI	MMXVI
2	3	6	16	33	1666	2016

# Roman Numerals

- Basic units

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

- Additive combination of units

II	III	VI	XVI	XXXIII	MDCLXVI	MMXVI
2	3	6	16	33	1666	2016

- Subtractive combination of units

IV	IX	XL	XC	CD	CM	MCMLXXI
----	----	----	----	----	----	---------



# Roman Numerals

- Basic units

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

- Additive combination of units

II	III	VI	XVI	XXXIII	MDCLXVI	MMXVI
2	3	6	16	33	1666	2016

- Subtractive combination of units

IV	IX	XL	XC	CD	CM	MCMLXXI
4	9	40	90	400	900	1971

# Roman Numerals

- Basic units

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

- Additive combination of units

II	III	VI	XVI	XXXIII	MDCLXVI	MMXVI
2	3	6	16	33	1666	2016

- Subtractive combination of units

IV	IX	XL	XC	CD	CM	MCMLXXI
4	9	40	90	400	900	1971

# Arabic Numerals

- Developed in India and Arabic world during the European Dark Age
- Decisive step: invention of zero by Brahmagupta in AD 628
- Basic units

0 1 2 3 4 5 6 7 8 9

- Positional system

1 10 100 1000 10000 100000 1000000

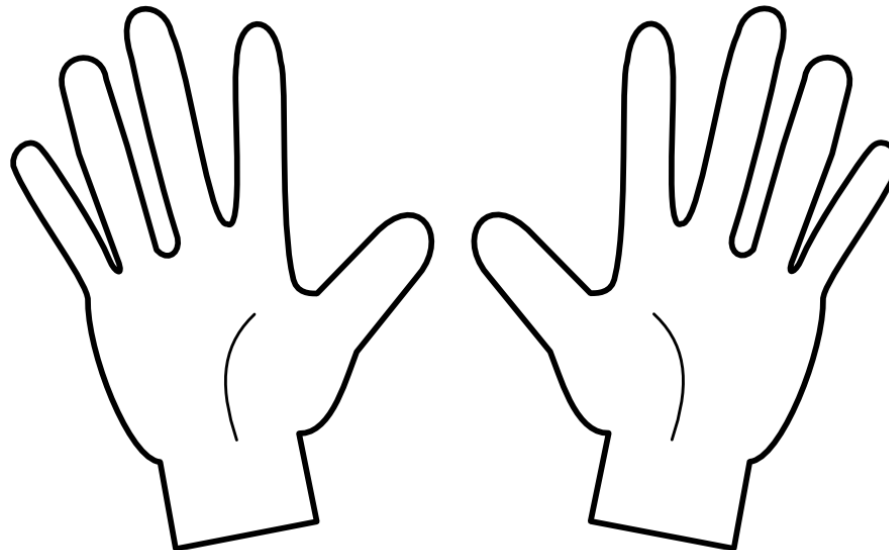
# Why Base 10?

dig·it

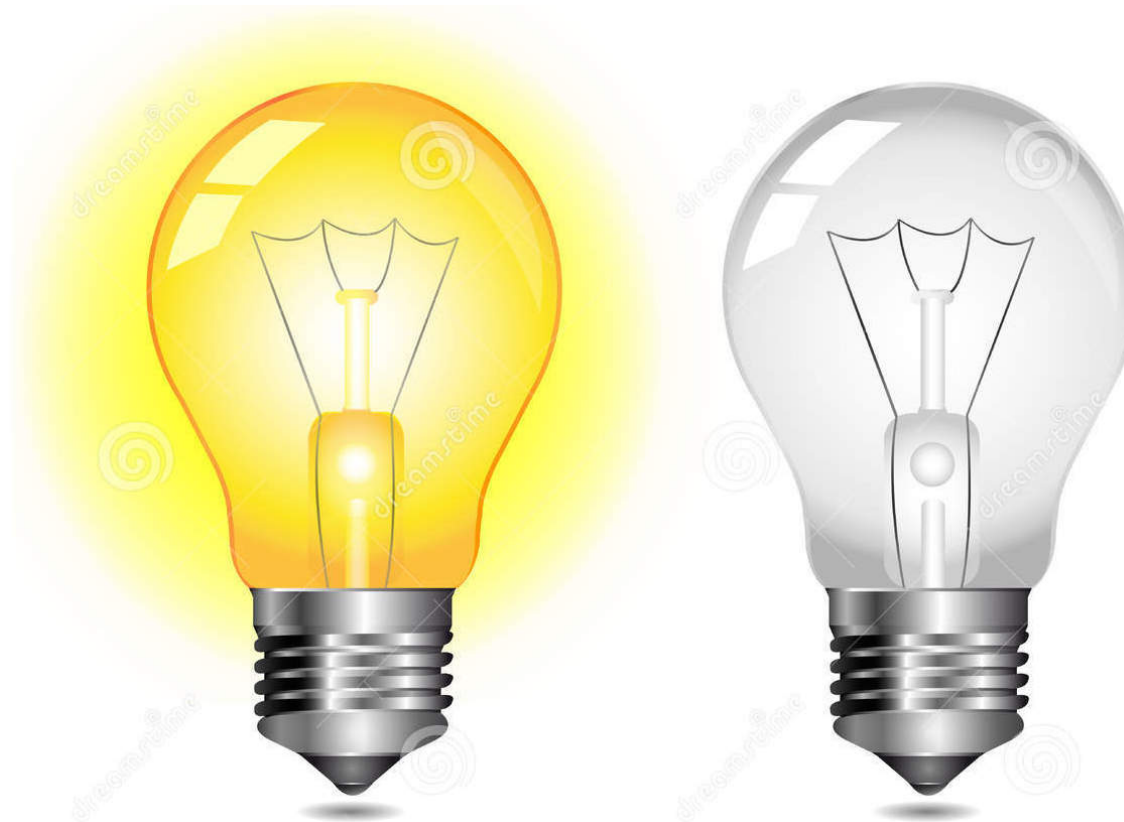
/ˈdɪjɪt/ 

*noun*

1. any of the numerals from 0 to 9, especially when forming part of a number.  
*synonyms:* numeral, number, figure, integer  
"the door code has ten digits"
2. a finger (including the thumb) or toe.  
*synonyms:* finger, thumb, toe; extremity  
"we wanted to warm our frozen digits"



# Base 2



# Base 2

- Decoding binary numbers

Binary number	1	1	0	1	0	1	0	1
---------------	---	---	---	---	---	---	---	---

# Base 2

- Decoding binary numbers

Binary number	1	1	0	1	0	1	0	1
Position	7	6	5	4	3	2	1	0

# Base 2

- Decoding binary numbers

Binary number	1	1	0	1	0	1	0	1
Position	7	6	5	4	3	2	1	0
Value	$2^7$	$2^6$	0	$2^4$	0	$2^2$	0	$2^0$



- Decoding binary numbers

Binary number	1	1	0	1	0	1	0	1	
Position	7	6	5	4	3	2	1	0	
Value	$2^7$	$2^6$	0	$2^4$	0	$2^2$	0	$2^0$	
	128	64	0	16	0	4	0	1	= 213

# Base 8

- Numbers like 11010101 are very hard to read

⇒ Octal numbers

Binary number	1	1	0	1	0	1	0	1
	-----		-----			-----		
Octal number	3		2			5		

# Base 8

- Numbers like 11010101 are very hard to read

⇒ Octal numbers

Binary number	1	1	0	1	0	1	0	1
	-----		-----			-----		
Octal number	3		2			5		
Position	2		1			0		

# Base 8

- Numbers like 11010101 are very hard to read

⇒ Octal numbers

Binary number	1	1	0	1	0	1	0	1
	-----		-----			-----		
Octal number	3		2			5		
Position	2		1			0		
Value	$3 \times 8^2$		$2 \times 8^1$			$5 \times 8^0$		

# Base 8

- Numbers like 11010101 are very hard to read

⇒ Octal numbers

Binary number	1	1	0	1	0	1	0	1	
	-----		-----			-----			
Octal number	3		2			5			
Position	2		1			0			
Value	$3 \times 8^2$		$2 \times 8^1$			$5 \times 8^0$			
	192		16			5			= 213

- ... but grouping **three** binary digits is a bit odd

# Base 16

31



- Grouping 4 binary digits  $\rightarrow$  base  $2^4 = 16$
- "Hexadecimal" (hex = Greek for six, decimus = Latin for tenth)

# Base 16



- Grouping 4 binary digits  $\rightarrow$  base  $2^4 = 16$
- "Hexadecimal" (hex = Greek for six, decimus = Latin for tenth)
- Need characters for 10-15:

# Base 16

31



- Grouping 4 binary digits  $\rightarrow$  base  $2^4 = 16$
- "Hexadecimal" (hex = Greek for six, decimus = Latin for tenth)
- Need characters for 10-15: use letters a-f

Binary number	1	1	0	1	0	1	0	1
	-----				-----			
Hexadecimal number	d				5			



# Base 16

31



- Grouping 4 binary digits  $\rightarrow$  base  $2^4 = 16$
- "Hexadecimal" (hex = Greek for six, decimus = Latin for tenth)
- Need characters for 10-15: use letters a-f

Binary number	1	1	0	1	0	1	0	1
	-----				-----			
Hexadecimal number	d				5			
Position	1				0			

# Base 16

- Grouping 4 binary digits  $\rightarrow$  base  $2^4 = 16$
- "Hexadecimal" (hex = Greek for six, decimus = Latin for tenth)
- Need characters for 10-15: use letters a-f

Binary number	1	1	0	1	0	1	0	1
	-----				-----			
Hexadecimal number	d				5			
Position	1				0			
Value	$13 \times 16^1$				$5 \times 16^0$			
	208				5			

# Examples

32



Decimal	Binary	Octal	Hexademical
0			
1			
2			
3			
8			
15			
16			
20			
23			
24			
30			
50			
100			
255			
256			

# Examples

Decimal	Binary	Octal	Hexademical
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
8	1000	10	8
15	1111	17	f
16	10000	20	10
20	10100	24	14
23	10111	27	17
24	11000	30	18
30	11110	36	1e
50	110010	62	32
100	1100100	144	64
255	11111111	377	ff
256	100000000	400	100

# adding binary numbers

# Binary Addition

- Adding binary numbers - just like decimal numbers

A	0	1	0	1	0	1	0	1
B	0	0	0	1	1	1	0	0
<hr/>								
Carry								
<hr/>								
A+B								

- Problem setup

# Binary Addition

- Adding binary numbers - just like decimal numbers

A	0	1	0	1	0	1	0	1
B	0	0	0	1	1	1	0	0
<hr/>								
Carry							-	
<hr/>								
A+B								1

- Adding the last two digits:  $1 + 0 = 1$

# Binary Addition

- Adding binary numbers - just like decimal numbers

A	0	1	0	1	0	1	0	1
B	0	0	0	1	1	1	0	0
Carry						-	-	
A+B							0	1

- Adding the next two digits:  $0 + 0 = 0$



# Binary Addition

- Adding binary numbers - just like decimal numbers

A	0	1	0	1	0	1	0	1
B	0	0	0	1	1	1	0	0
Carry					1	-	-	
A+B						0	0	1

- Adding the next two digits:  $1 + 1 = 0$ , carry 1

# Binary Addition

- Adding binary numbers - just like decimal numbers

A	0	1	0	1	0	1	0	1
B	0	0	0	1	1	1	0	0
Carry				1	1	-	-	
A+B					0	0	0	1

- Adding the next two digits, plus carry :  $0 + 1 + 1 = 0$ , carry 1

# Binary Addition

- Adding binary numbers - just like decimal numbers

A	0	1	0	1	0	1	0	1
B	0	0	0	1	1	1	0	0
<hr/>								
Carry			1	1	1	-	-	
<hr/>								
A+B				1	0	0	0	1

- Adding the next two digits, plus carry :  $1 + 1 + 1 = 0$ , carry 1

# Binary Addition

- Adding binary numbers - just like decimal numbers

A	0	1	0	1	0	1	0	1
B	0	0	0	1	1	1	0	0
Carry	-	-	1	1	1	-	-	
A+B	0	1	1	1	0	0	0	1

- And so on...



# negative numbers

# Positive Numbers

Bits			Unsigned			
0	0	0	0			
0	0	1	1			
0	1	0	2			
0	1	1	3			
1	0	0	4			
1	0	1	5			
1	1	0	6			
1	1	1	7			

- Encoding for unsigned binary numbers

# One Bit for Sign

Bits			Unsigned	Sign + Magnitude		
0	0	0	0	+0		
0	0	1	1	+1		
0	1	0	2	+2		
0	1	1	3	+3		
1	0	0	4	-0		
1	0	1	5	-1		
1	1	0	6	-2		
1	1	1	7	-3		

- Use the first bit to encode sign: 0 = positive, 1 = negative
- How can we do addition with this?

# One's Complement

Bits			Unsigned	Sign + Magnitude	One's Complement
0	0	0	0	+0	+0
0	0	1	1	+1	+1
0	1	0	2	+2	+2
0	1	1	3	+3	+3
1	0	0	4	-0	-3
1	0	1	5	-1	-2
1	1	0	6	-2	-1
1	1	1	7	-3	-0

- Negative number: flip all bits
- Some waste: two zeros (+0=000 and -0=111)



# Two's Complement

Bits	Unsigned	Sign + Magnitude	One's Complement	Two's Complement
0 0 0	0	+0	+0	+0
0 0 1	1	+1	+1	+1
0 1 0	2	+2	+2	+2
0 1 1	3	+3	+3	+3
1 0 0	4	-0	-3	-4
1 0 1	5	-1	-2	-3
1 1 0	6	-2	-1	-2
1 1 1	7	-3	-0	-1

- Negative number: flip all bits, add 001
- Addition works as before:  
 $-1 + -1 = 111 + 111 = 1110 = -2$   
 $+2 + -1 = 010 + 111 = 1001 = +1$