REPUBLIC POLYTECHNIC

Lesson 01
Number System

SCHOOL OF ENGINEERING

Grading and Deliverables

Lessons 01-06: (deadline: lesson day 23:59)

- Complete all lessons LEO2.0 online-quiz
- Participate in **Lesson Checkpoints** and **Lesson Activities**
- Upload completed **Activity Sheet** for each lesson in individual submission
- Attempt to solve the Problem statement
- Self-evaluation
- Peer Evaluation

Lesson 07 CA week: (deadline: Lesson 07, 23:59)

- All the above, and
- Upload completed **Problem Statement Solution** in individual submission - ***AY2024 Sem1 E105_CA1_PS.docx (can be found in Lesson 1 Package)***
- Reflection Journal
- CA Quiz (will be conducted at 3:30pm)

***Exercises provided are for students to practice on their own. Submission is not required.**

IMPORTANT NOTE:

- **Absentees** with or without LOA, **must complete all submissions**. Failure to do so will lead to zero grade points for that particular CA.
- **Absentees with LOA** should **email** their deliverables to their class facilitator on LOA period + 1 (e.g. if your LOA ended on 11 April 2023, you should submit the deliverables by 12 April 2023, 2359.)
- There is a **make-up CA Quiz** arrangement for **absentees with LOA** only.

Lesson Plan E105 L01

Timing	Content
60 min 9.15 am – 10.15am	Session 1 ➲ Module Introduction ➲ Introduction to Digital Electronics ➲ Analog vs Digital
45 min	🍽 Study Break
90 min 11.00 am – 12.30 am	Session 2 ➲ Number Systems ➲ Activity 2
90 min	🍽 Study Break
75 min 2.00 pm – 3.45 pm	Session 3 ➲ Logic Gates ➲ Universal Gates ➲ Activity 3
15min 3.45 pm – 4.00 pm	➲ Consultation Session on CA1 PS After class: Complete Week 01 section of the PS (individual work)

**All Lesson Task (Submit by 2359)*

interactive seminar

Module Introduction

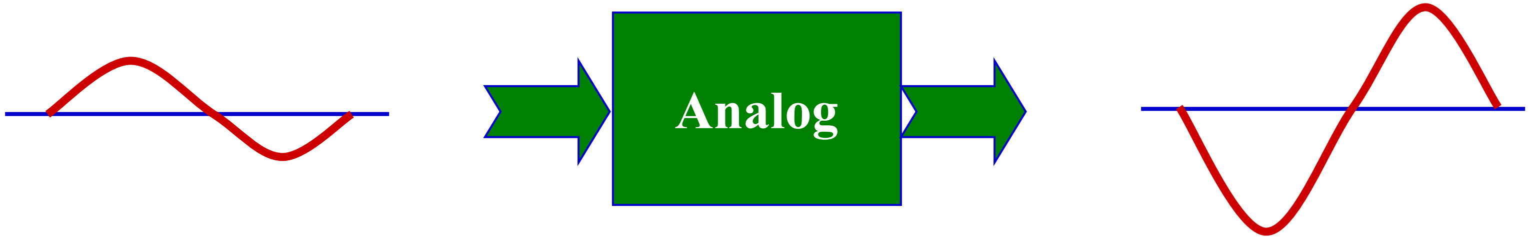
- Facilitator will go through the module introduction slides

Learning Outcomes

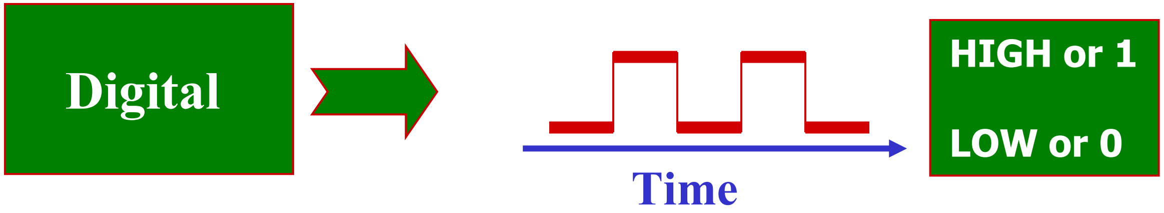
- Differentiate between digital and analog signals, and identify the HIGH and LOW portions of a digital waveform
- Describe the decimal, binary and hexadecimal number systems *(Activity 1)*
- Describe the operations of common logic gates *(Activity 2)*

ANALOG VS. DIGITAL

- **Analog signal**:
  - one whose output varies **continuously** in step with the input.
  - Example:



- **Digital signal:**
  - one whose output varies at **discrete voltage** levels commonly called **HIGH** or **LOW** (1 or 0).
  - Example:



Why Analog?

- Most “real-world” events are analog in nature.
- Analog processing is usually simpler.
- Analog processing is usually faster.
- Traditional electronic systems were mostly analog in nature.

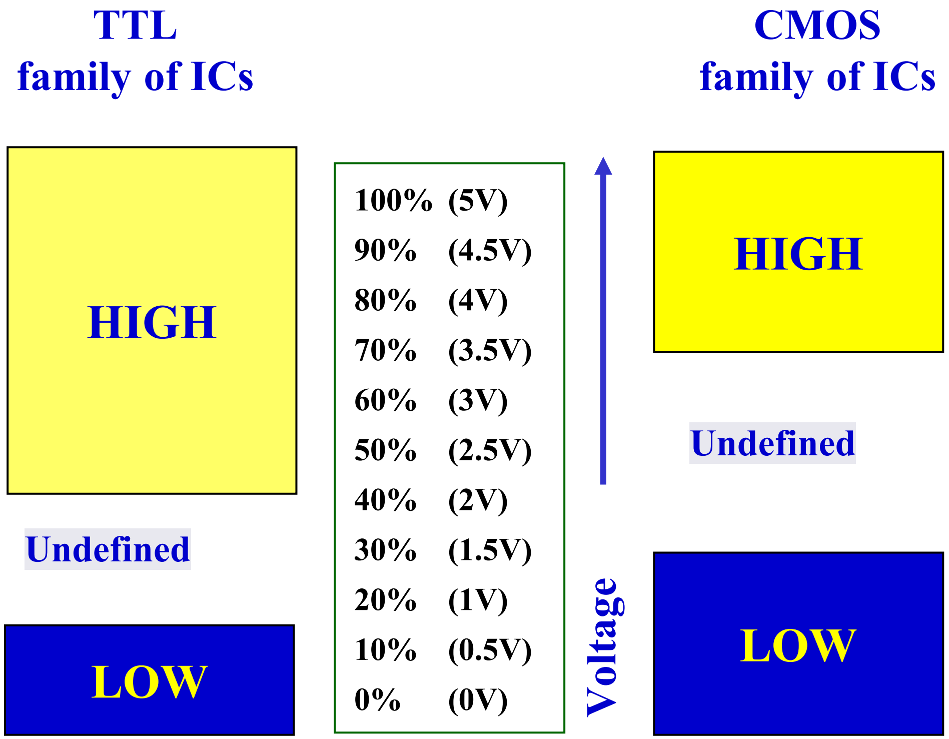
Why Digital?

- Data can be stored (memory characteristic of digital).
- Data can be used in calculations.
- Compatible with display technologies.
- Compatible with computer technologies.
- Systems can be programmed.
- Digital IC families make design easier.

Defining logic levels

- Logic devices interpret input voltages as either HIGH or LOW.
- TTL or CMOS IC families have their unique voltage profiles.
- Both TTL and CMOS IC input voltage profiles are shown below.

TTL: **T**ransistor-**T**ransistor **L**ogic
CMOS: **C**omplementary **M**etal **O**xide **S**emiconductor



TTL family of ICs

HIGH

Undefined

LOW

100% (5V)
90% (4.5V)
80% (4V)
70% (3.5V)
60% (3V)
50% (2.5V)
40% (2V)
30% (1.5V)
20% (1V)
10% (0.5V)
0% (0V)

Voltage

CMOS family of ICs

HIGH

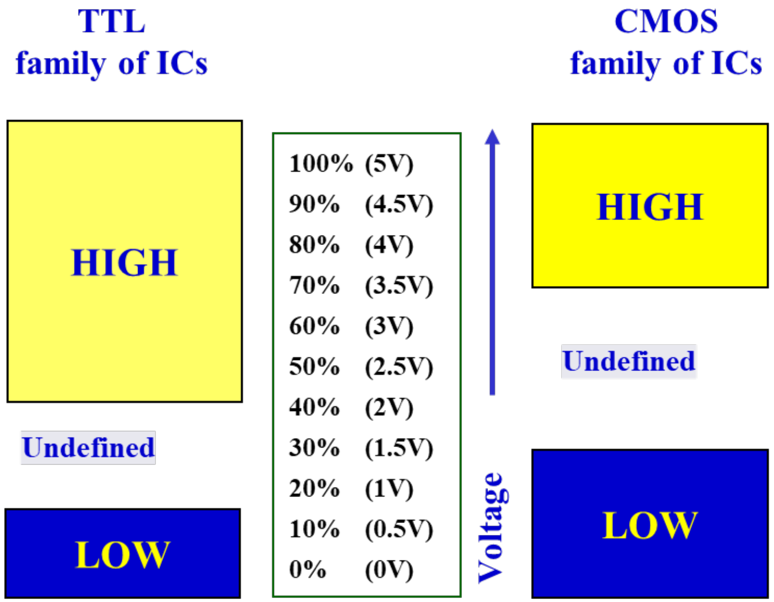
Undefined

LOW

CAUTION: Input voltages in the UNDEFINED region may yield unpredictable results.

Defining logic levels

- Example: Supply voltage range is from 0V to +5V
  - An input voltage of **+3V** to a **TTL IC** would be considered a **HIGH** logic level
  - An input voltage of **+3V** to a **CMOS IC** would be considered an **Undefined** logic level
  - An input voltage of **+1V** to a **TTL IC** would be considered an **Undefined** logic level
  - An input voltage of **+1V** to a **CMOS IC** would be considered a **LOW** logic level



Checkpoints

1. The +5V of the digital signal could also be called a logical 1 or a __________ (HIGH, LOW).

HIGH

2. A(n) ________ (analog, digital) device is one that has a signal which varies continuously in step with the input.

analog

3. The _______ (analog, digital) signal has only two voltage levels.

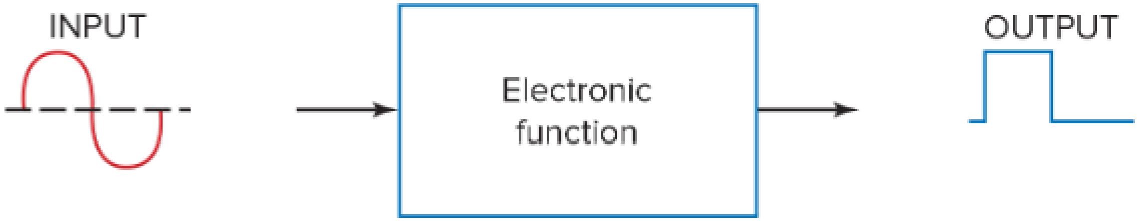
digital

4. An analog circuit is one that processes analog signals while digital circuit processes _________ (analog, digital) signals.

digital

Checkpoints

Refer to the below figure.



Block diagram of electronic circuit shaping a sine wave into a square wave.

5. The *input* to the electronic block is classified as a(n) ________ (analog, digital) signal.

analog

6. The *output* from the electronic block is classified as a(n) ________ (analog, digital) signal.

digital

Number Systems

- Common types of number systems used in Digital and Computer Technology:
  - Decimal Number System (Base **10** System)
  - Binary Number System (Base **2** System)
  - Hexadecimal Number System (Base **16** System)

Tip: For DE, you need to memorize the binary symbols used to **at least 15**

Decimal Numbering System

- Decimal is the universal system used to represent quantities outside a digital system.
- Has **10** symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Also called **Base 10** system
- Leftmost digit is the **MSD** (Most Significant Digit) and the right most digit is the **LSD** (Least Significant Digit).
- Place Values:

Thousands	Hundreds	Tens	Units
10^3	10^2	10^1	10^0
7	3	5	4

$$7000 + 300 + 50 + 4$$

(MSD) (LSD)

Example: $7354_{10} = (7 \times 10^3) + (3 \times 10^2) + (5 \times 10^1) + (4 \times 10^0)$

$= 7000 + 300 + 50 + 4$

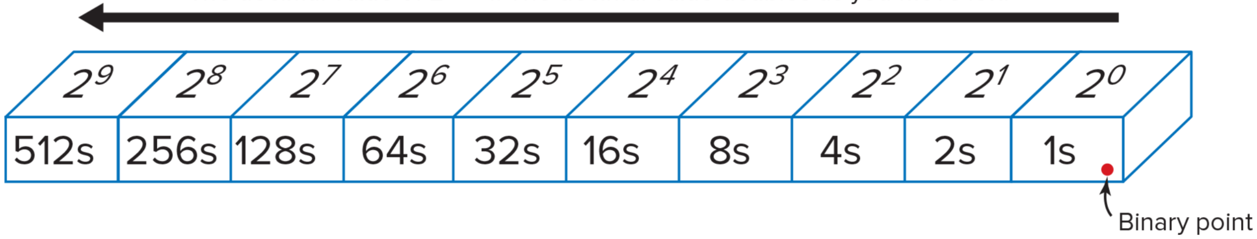
Binary Numbering System

- Most important system in Digital System.
- All values must be converted to binary before entering into the digital system.
- Has **2** symbols: 0 and 1
- Also called **Base 2** system
- Leftmost digit is the **MSB** (Most Significant Bit) and the right most digit is the **LSB** (Least Significant Bit).

E.g.	1	0	0	1_2
	(MSB)			(LSB)

Binary Numbering System

- Binary Place Values and its equivalent decimal values



The power of 2 starts at zero (0) and increases by one (1) as you move to the left of the binary point.
The decimal value of $2^0 = 1$. The decimal value doubles as you move left.

E.g. $1 \quad 0 \quad 0 \quad 1_2$

$$= (1\times2^3) + (0\times2^2) + (0\times2^1) + (1\times2^0)$$

$$= 8 + 0 + 0 + 1$$

$$= 9$$

Place Value	Decimal Value
2^9	512
2^8	256
2^7	128
2^6	64
2^5	32
2^4	16
2^3	8
2^2	4
2^1	2
2^0	1

Binary-to-Decimal Conversion

- Steps: Multiply with Binary Place Values

i. Write down the binary number

ii. Write down the place values for each binary digit (2^n: start with 2^0 and increases n by one (1) as you move to the left)

iii. Multiply the binary digit with its place value

iv. Add all the decimal values to find the equivalent

E.g. Convert 10101_2 to decimal:

ii Place values 2^4 2^3 2^2 2^1 2^0

i Binary 1 0 1 0 1

iii Decimal $16 + 0 + 4 + 0 + 1 = 21_{10}$ iv

Checkpoints

Convert the following binary number into decimal number:

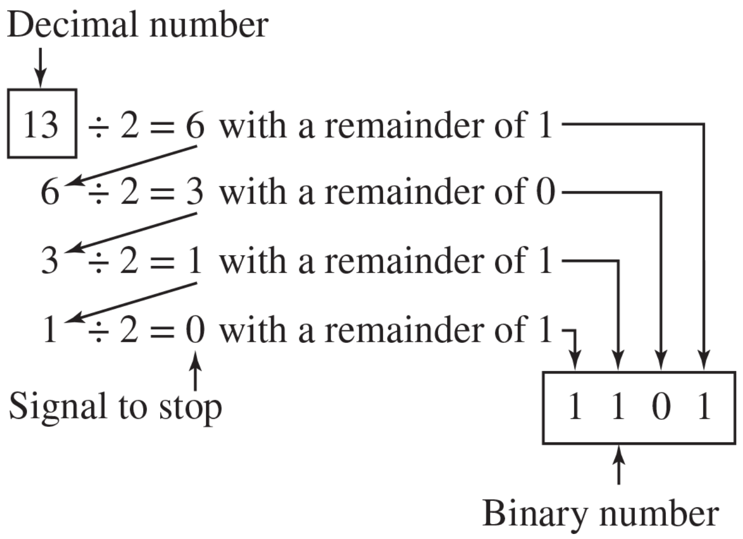
Binary $1001_2 = 9_{10}$

Place Value	Decimal Value
2^8	256
2^7	128
2^6	64
2^5	32
2^4	16
2^3	8
2^2	4
2^1	2
2^0	1

Decimal-to-Binary Conversion

- Steps: Repeated Divide-by-2 process
  1. Divide the decimal number by 2.
  2. Write the remainder after each division until a quotient of zero is obtained.
  3. Read the remainder from bottom to top
     (LAST remainder is the MSB, FIRST is the LSB)

Example 1: Convert 13_{10} to binary:



Decimal number

13 ÷ 2 = 6 with a remainder of 1

6 ÷ 2 = 3 with a remainder of 0

3 ÷ 2 = 1 with a remainder of 1

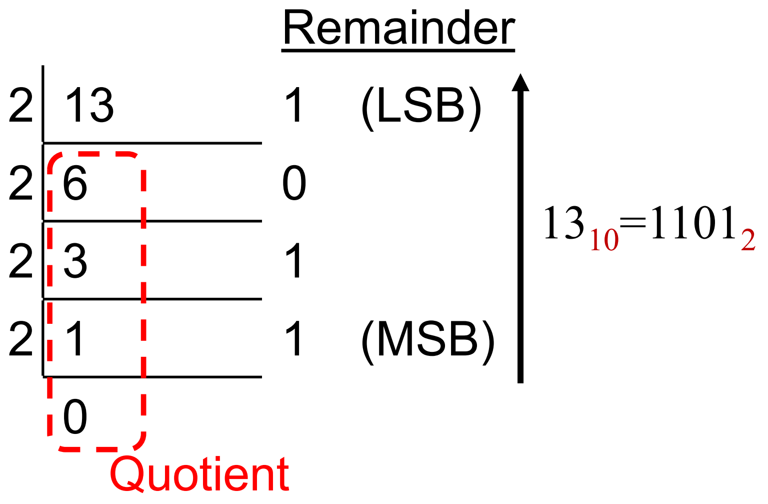
1 ÷ 2 = 0 with a remainder of 1

Signal to stop

1 1 0 1

Binary number

or



Remainder

2 | 13 1 (LSB)

2 | 6 0

2 | 3 1

2 | 1 1 (MSB)

0

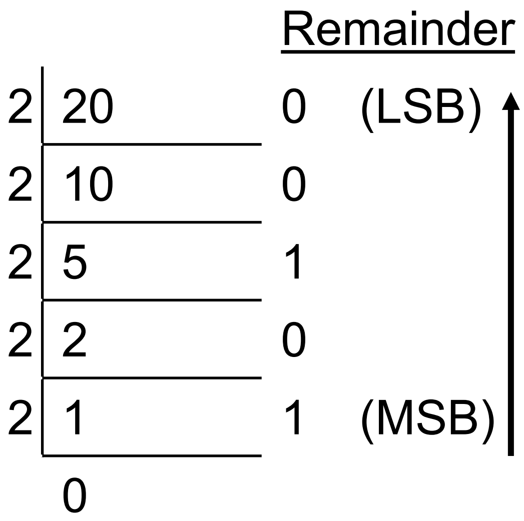
Quotient

$13_{10}=1101_2$

Decimal-to-Binary Conversion

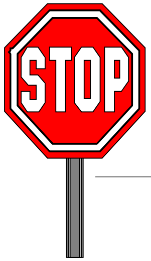
- Steps: Repeated Divide-by-2 process
  1. Divide the decimal number by 2.
  2. Write the remainder after each division until a quotient of zero is obtained.
  3. Read the remainder from bottom to top (LAST remainder is the MSB, FIRST is the LSB)

Example 2: Convert 20_{10} to binary:



		Remainder	
2	20	0	(LSB)
2	10	0	
2	5	1	
2	2	0	
2	1	1	(MSB)
	0		

$20_{10}=10100_2$



Checkpoints

Convert the following decimal number into binary:

Decimal $30_{10} = 11110_2$

Place Value	Decimal Value
2^8	256
2^7	128
2^6	64
2^5	32
2^4	16
2^3	8
2^2	4
2^1	2
2^0	1

Range of Binary Representation

- Using *N* bits:
  - – Total different numbers: 2^N
  - – Range: 0 to 2^N -1
- The more bits, the bigger the number

- Example: Number of bit *N* = 4
  - – We can count from 0000_2 to 1111_2, which is 0_{10} to 15_{10}, for a total of 16 different numbers
  - – Total different numbers = 2^4 = 16
  - – Largest decimal value = 2^4 – 1 = 15

Counting in Binary System

Decimal System	Binary System
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

- The least significant bit (LSB) of the binary representation of Even quantities is ‘0’ and Odd quantities is ‘1’.

- Doubling a decimal value will have the equivalent effect of shifting the binary value to the left by one position

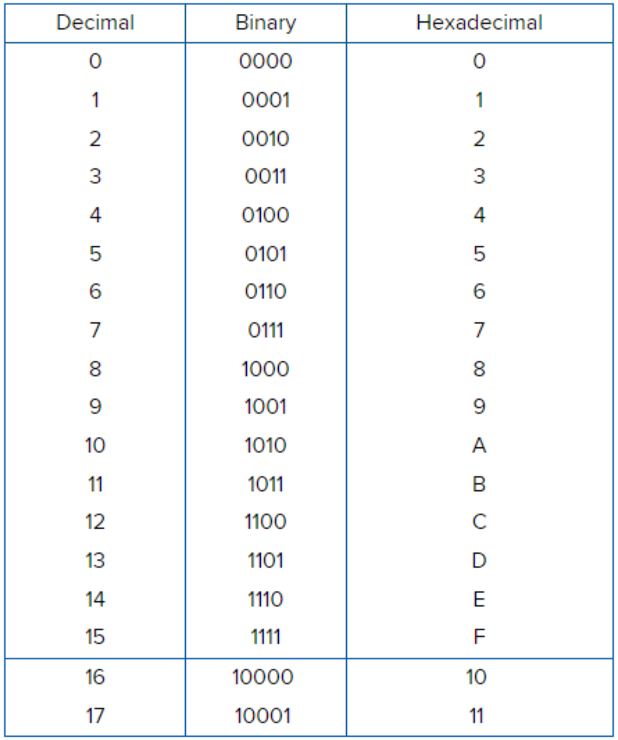
E.g.:

$2_{10} = 0010_2$,
$4_{10} = 0100_2$,
$8_{10} = 1000_2$

Note: Memorize the binary system

Hexadecimal Numbering System

- Primarily used a “shorthand” method for representing binary.
- Has **16** symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Also called **Base 16** system

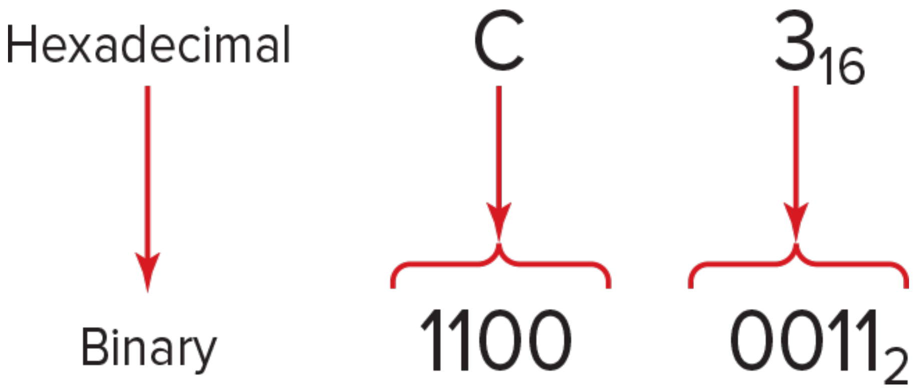


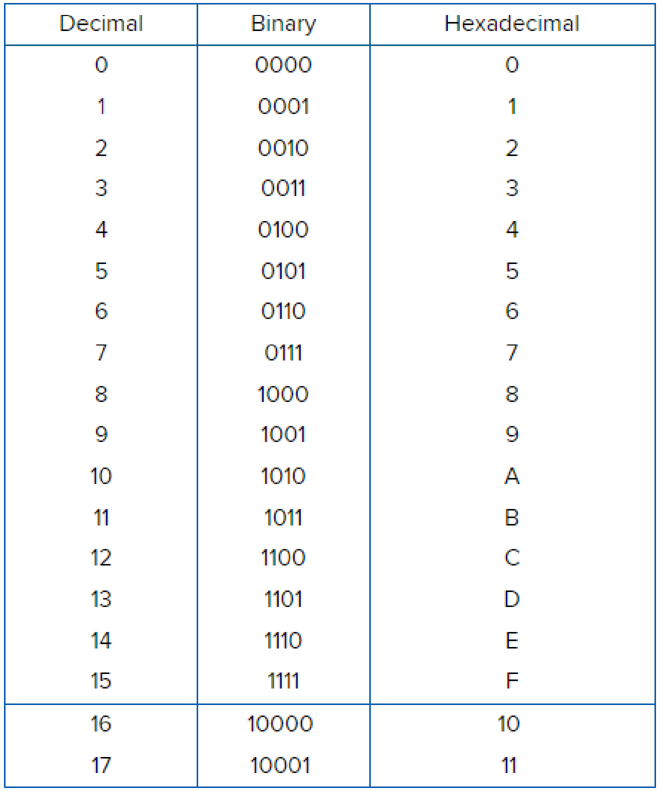
Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11

Fig. 2-15 Binary and hexadecimal equivalents to decimal numbers.

Hexadecimal-to-Binary Conversion

- Steps: Convert each Hexadecimal digit to its **4-bit Binary equivalent**
- Example: $C3_{16}$ to binary





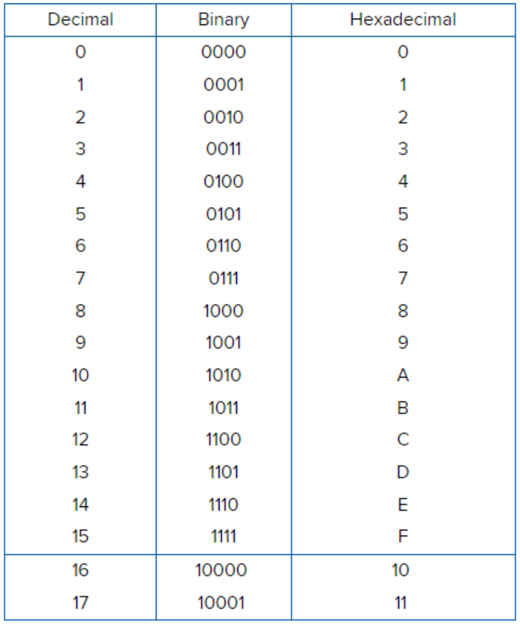
Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11

Fig. 2-15 Binary and hexadecimal equivalents to decimal numbers.

Checkpoints

Convert the following Hexadecimal number into binary number:

$$1F6_{16} = 1\ 1111\ 0110_2$$



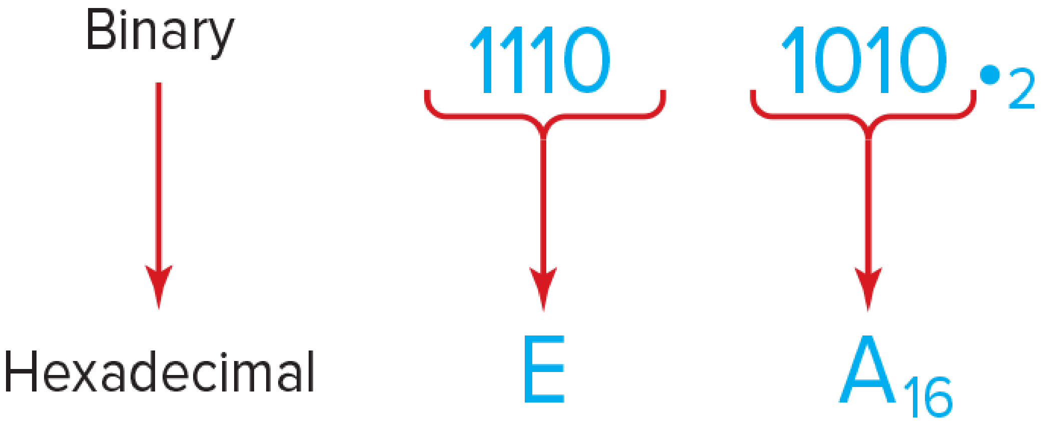
Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11

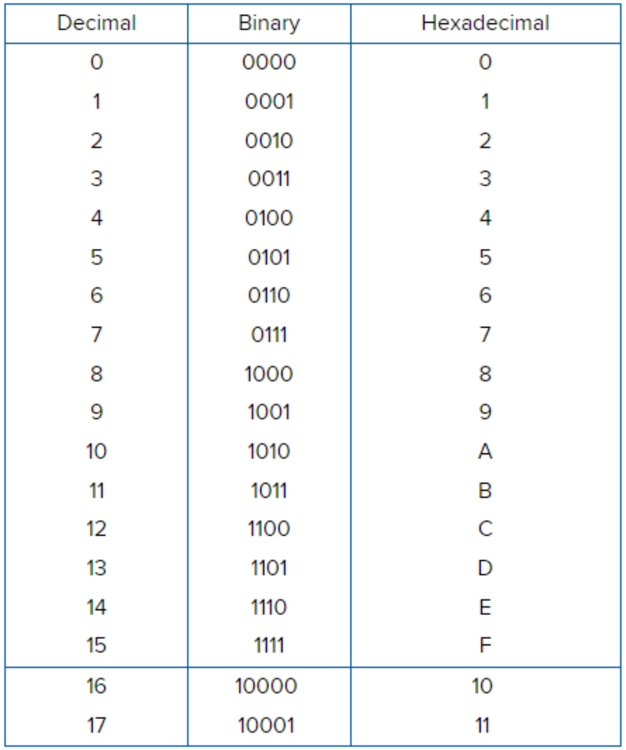
Fig. 2-15 Binary and hexadecimal equivalents to decimal numbers.

Binary-to-Hexadecimal Conversion

- Steps: Convert each group to its hexadecimal equivalent
  - i. Split the binary number into **4-bit groups** (from right to left)
  - **ii.** **Convert** each 4-bit group into its equivalent **hexadecimal** number

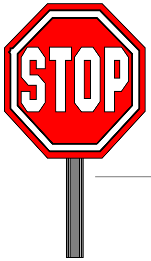
- Example: 11101010_2 to Hexadecimal





Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11

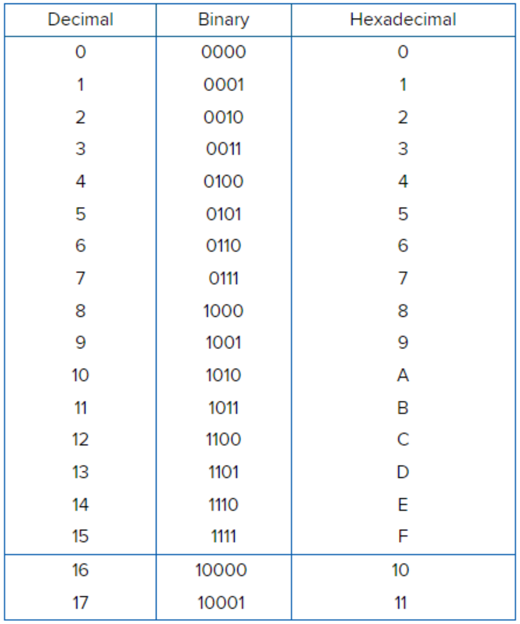
Fig. 2-15 Binary and hexadecimal equivalents to decimal numbers.



Checkpoints

Convert the following binary numbers into hexadecimal numbers:

$$100100100_2 = 124_{16}$$



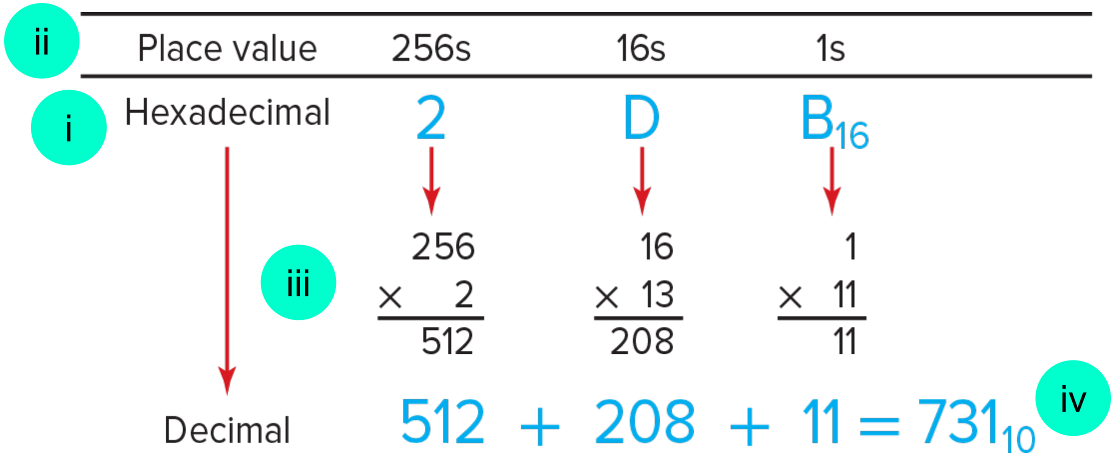
Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11

Fig. 2-15 Binary and hexadecimal equivalents to decimal numbers.

Hexadecimal-to-Decimal Conversion

- Steps: Multiply with Hexadecimal Place Values
  - i. Write down the hexadecimal number
  - ii. Write down the place values for each hexadecimal digit (16^n: start with 16^0 and increases n by one (1) as you move to the left)
  - iii. Multiply the hexadecimal digit with its place value
  - iv. Add all the decimal values to find the equivalent

E.g. Convert $2DB_{16}$ to decimal:



ii

Place value	256s	16s	1s
Hexadecimal	2	D	B_{16}

i

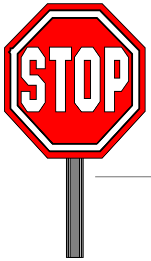
iii

$256 \times 2 = 512$ $16 \times 13 = 208$ $1 \times 11 = 11$

Decimal $512 + 208 + 11 = 731_{10}$

iv

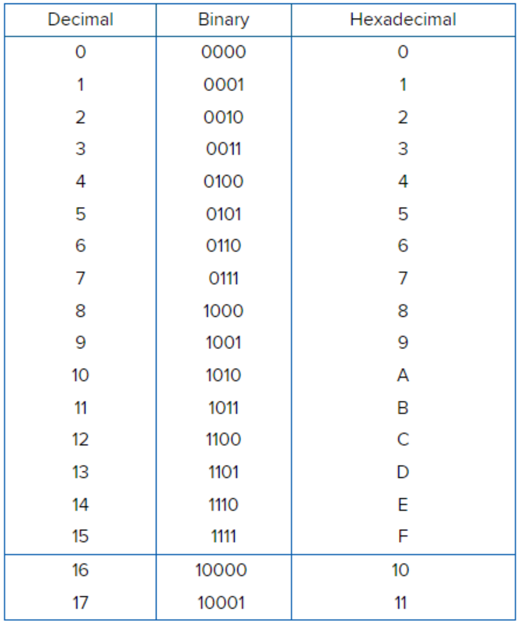
Place Value	Decimal Value
16^5	1,048,576
16^4	65,536
16^3	4096
16^2	256
16^1	16
16^0	1



Checkpoints

Convert the following hexadecimal numbers into decimal numbers:

$$A6_{16} = 166_{10}$$



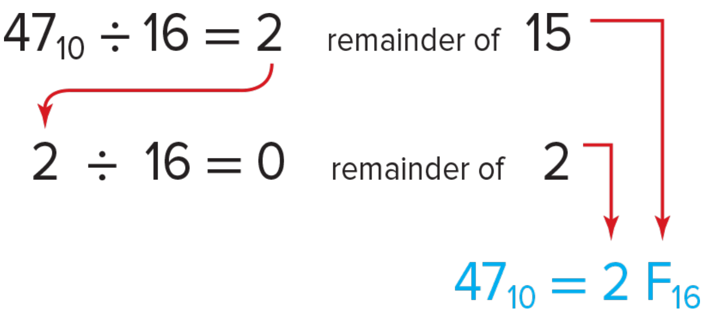
Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11

Fig. 2-15 Binary and hexadecimal equivalents to decimal numbers.

Decimal-to-Hexadecimal Conversion

- Steps: Repeated Divide-by-16 process
  - i. Divide the decimal number by 16.
  - ii. Write the remainder after each division until a quotient of zero is obtained.
  - iii. Read the remainder from bottom to top (LAST remainder is the MSD, FIRST is the LSD)

Example 1: Convert 47_{10} to hexadecimal:

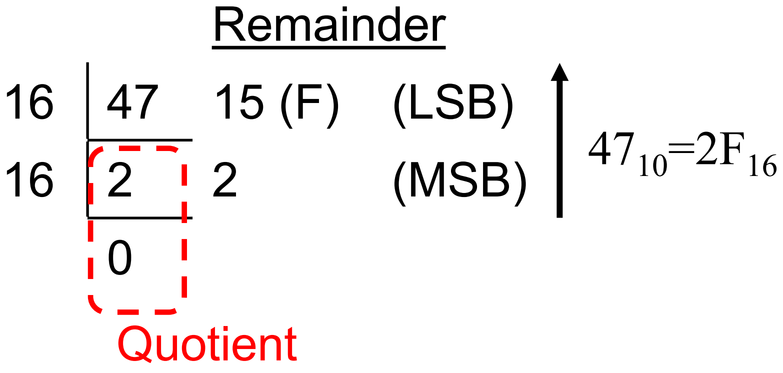


$47_{10} \div 16 = 2$ remainder of 15

$2 \div 16 = 0$ remainder of 2

$47_{10} = 2\,F_{16}$

or



Remainder

16	47	15 (F)	(LSB)
16	2	2	(MSB)
	0		

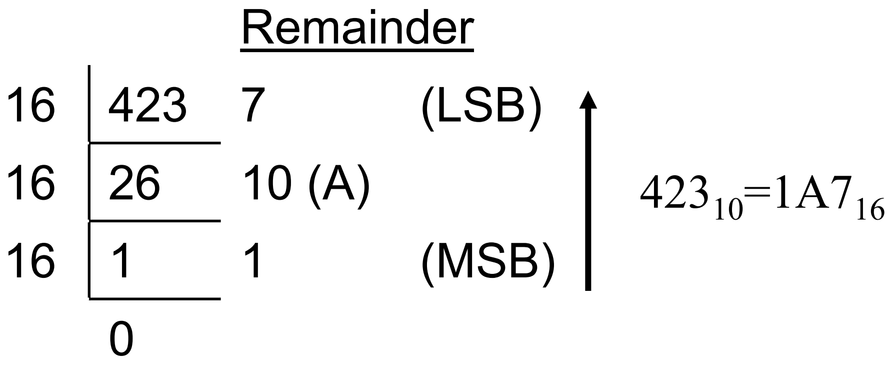
Quotient

47_{10}=$2F_{16}$

Decimal-to-Hexadecimal Conversion

- Steps: Repeated Divide-by-16 process
  i. Divide the decimal number by 16.
  ii. Write the remainder after each division until a quotient of zero is obtained.
  iii. Read the remainder from bottom to top
  (LAST remainder is the MSD, FIRST is the LSD)

Example 2: Convert 423_{10} to hexadecimal:



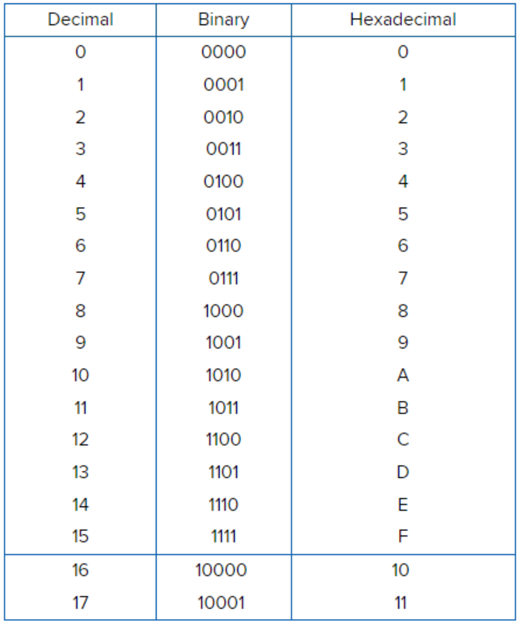
		Remainder	
16	423	7	(LSB)
16	26	10 (A)	
16	1	1	(MSB)
	0		

$423_{10}=1A7_{16}$

Checkpoints

Convert the following decimal number into hexadecimal number:

$$292_{10} = 124_{16}$$



Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11

Fig. 2-15 Binary and hexadecimal equivalents to decimal numbers.

Bits, Bytes, Nibble and Word Size

- A single binary number (either 0 or 1) is called a **bit** (**b**inary dig**it**)
- Bit is the **smallest unit** of data in a digital system
- Binary values are often grouped into a common bit-length:

Name	Bit Length	Example
Bit	1	0_2 or 1_2
Nibble	4	1001_2
Byte	8	$1100\ 1101_2$
Word	16	$1101\ 0111\ 0101\ 1000_2$
Double-word	32	$1101\ 1011\ 0100\ 0101\ 1101\ 0101\ 0100\ 1100_2$
Quad-word	64	$1010\ 1011\ 0011\ 1001\ 1100\ 1101\ 0100\ 1100\ 0110\ 1001\ 1101\ 0101\ 0111\ 0101\ 0011\ 1101_2$

Padding with leading zeros

- Binary digits are often **grouped in sets of four** to aid conversion to hexadecimal.
- If a binary grouping has fewer than four digits, **leading zeros** can be added.
- Leading zeros **don't alter the binary number's** value; they simply extend the grouping to four digits.
- This process of adding leading zeros is commonly known as **padding with zeros**.

- Example:

$1\ 0110_2$ → $0001\ 0110_2$
$10\ 1010_2$ → $0010\ 1010_2$
$110\ 1011_2$ → $0110\ 1011_2$

In a nutshell

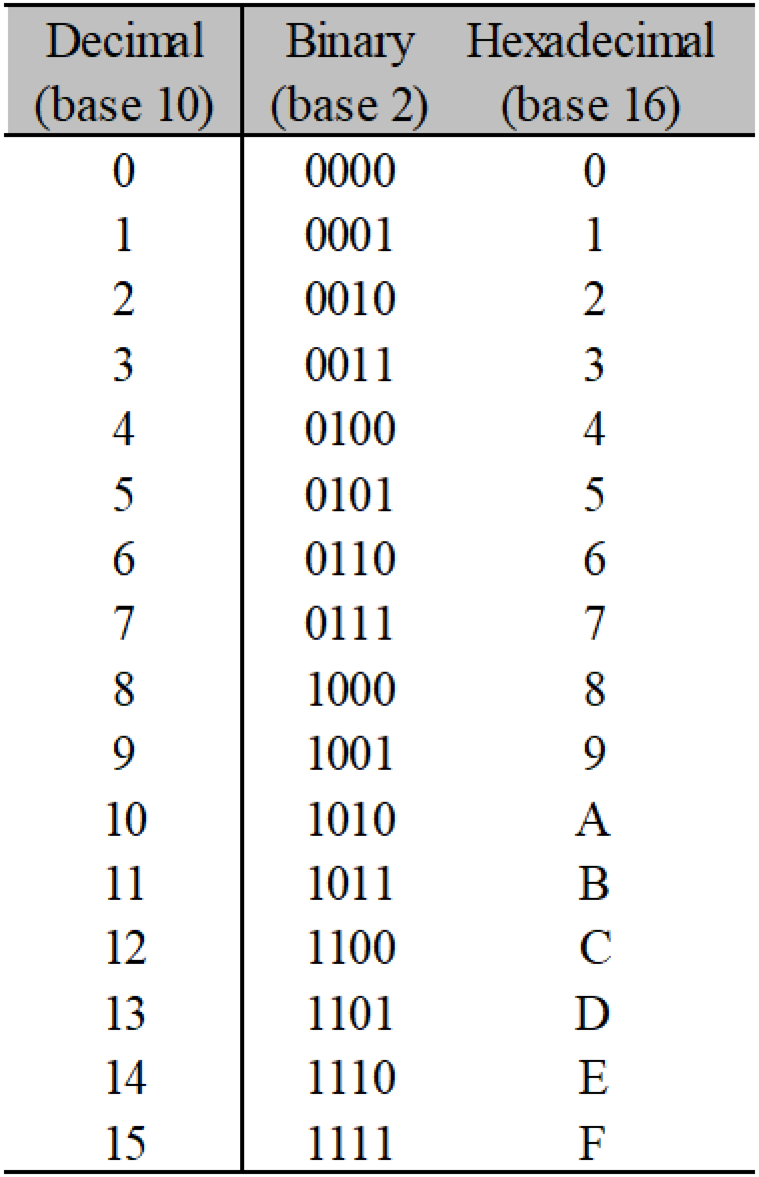
- Decimal vs Hexadecimal vs Binary

	Decimal	Hexadecimal	Binary
Number of symbols	10	16	2
Base	10	16	2
Symbols	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	0, 1
Highest Symbol Value	9	F	1

In a nutshell

Decimal	Binary	Hexadecimal
• Easy Readable • Used by humans • Easy to manipulate	• Used by computers • Easy to differentiate and switch thresholds (Eg 0V and V_{cc}) • Easy to understand and build logic gates. • Binary data is robust in transmission and reject noise.	• Very compact • Lesser digits to represent numbers in binary and decimal • Used in memory addresses • Easy conversion from binary
• Wastage of space • Wastage of Time	• Difficult to read and write for human	• Difficult to read and write for human • Difficult to perform operations like multiplication and division.

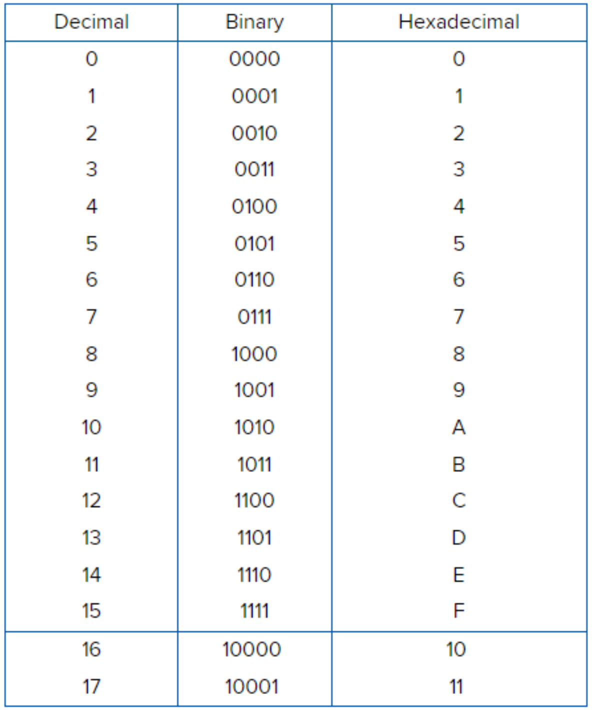
In a nutshell



Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Number System Conversion:

To From	Dec	Bin	Hex
Dec	[redacted]	Divide by 2	Divide by 16
Bin	Multiply by Base 2	[redacted]	Refer to table
Hex	Multiply by Base 16	Refer to table	[redacted]



Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11

Activity 1: Complete the questions in the Activity Sheet

- Complete Activity 1 questions in the activity sheet

Individual Work followed by Class Discussion

Logic Gates

- The basic building block of any digital circuit is a **logic gate**.
- Logic gate can be constructed using **TTL** ICs or **CMOS** ICs.

- The task performed by a logic gate is called its **logic function**.
- Logic functions can be implemented by hardware (logic gates) or by programming devices such as microcontrollers or computers.

- NOT, AND and OR gates are the basic logic gates which can be used to implement any combinational logic circuit.

Truth Table

- A truth table describes the **relationship** between the **input and output** of a **logic circuit**.
- The number of entries corresponds to the number of inputs.
  For example:
  A 2 input table would have 2^2 or 4 entries.
  A 3 input table would have 2^3 or 8 entries.

N-input truth table: 2^N input combinations

Truth Table

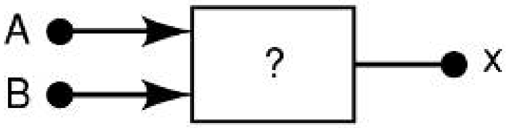
- Examples of truth tables with 2, 3, and 4 inputs:

Output

Inputs

$2^2 = 4$
input combinations

A	B	x
0	0	1
0	1	0
1	0	1
1	1	0



(a)

A	B	C	x
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b) $2^3 = 8$

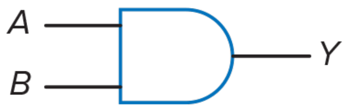
Binary counting sequence

A	B	C	D	x
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

(c) $2^4 = 16$

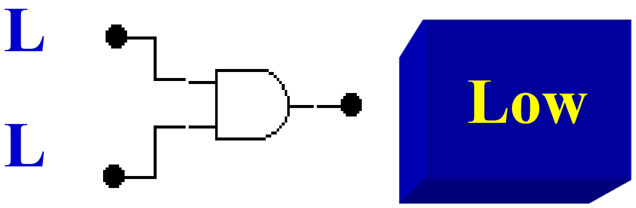
AND Gate

- “**All or Nothing**” Gate: Output is HIGH only when all its inputs are HIGH; else the output is LOW
- Four ways to express the logical ANDing of A and B:

<table>
<tr><td>In the English language</td><td>Input <i>A</i> is ANDed with input <i>B</i> to get output <i>Y</i>.</td></tr>
<tr><td>As a Boolean expression</td><td>$A \cdot B = Y$
AND symbol</td></tr>
<tr><td>As a logic symbol</td><td>
</td></tr>
<tr><td>As a truth table</td><td><table>
<tr><th><i>A</i></th><th><i>B</i></th><th><i>Y</i></th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table></td></tr>
</table>

Checkpoints

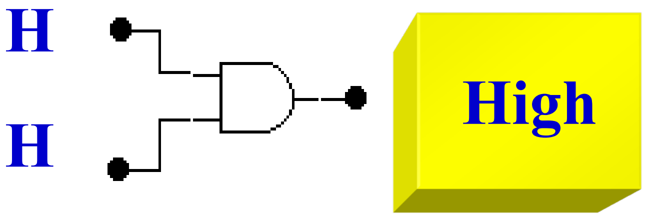
What is the output of the AND gate?



L

L

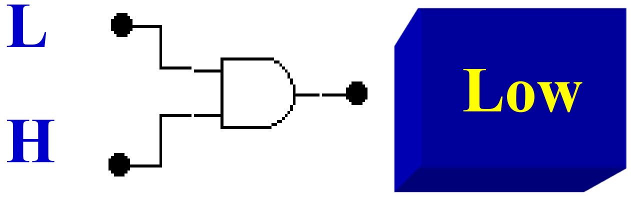
Low



H

H

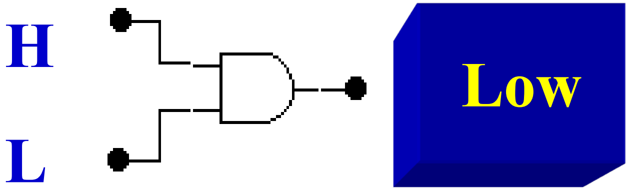
High



L

H

Low



H

L

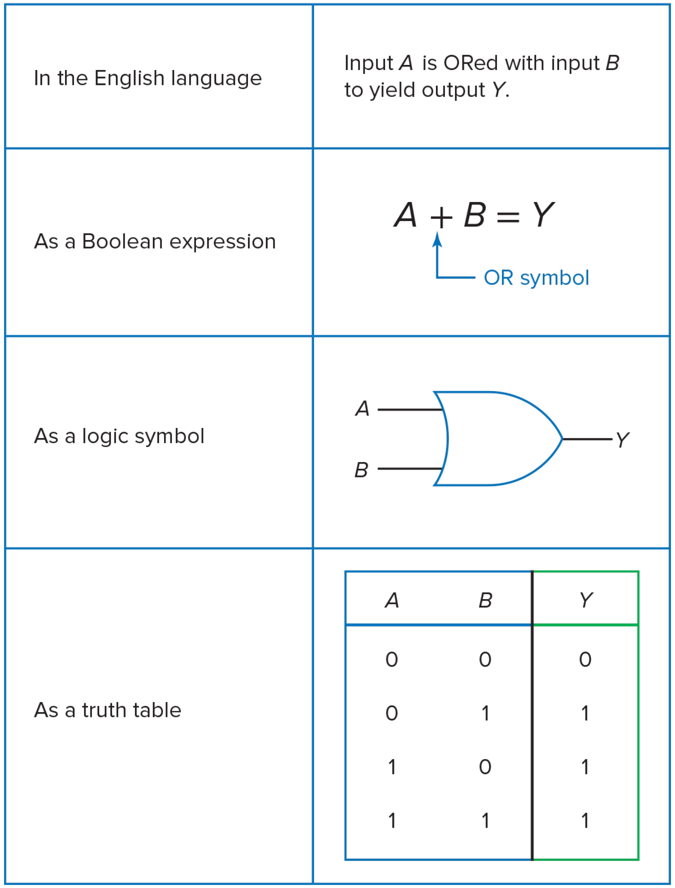
Low

Unique Output: Output HIGH only when all inputs are HIGH.

OR Gate

- "**Any or All Gate**" Gate: Output is HIGH when one or more inputs are HIGH; output will be LOW when all inputs are LOW
- Four ways to express the logical ORing of A and B:

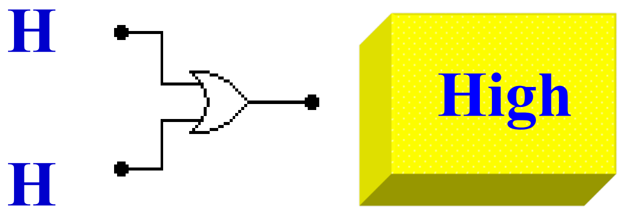
Describing the OR Function

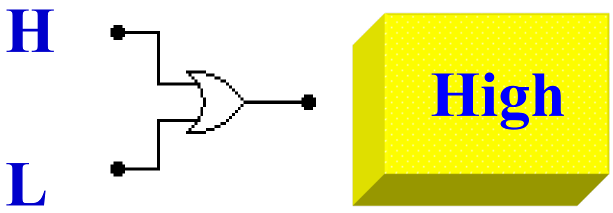


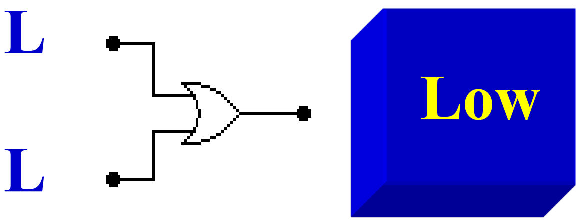
In the English language	Input *A* is ORed with input *B* to yield output *Y*.
As a Boolean expression	$A + B = Y$ OR symbol
As a logic symbol	*A* *B* *Y*
As a truth table	*A* *B* *Y* 0 0 0 0 1 1 1 0 1 1 1 1

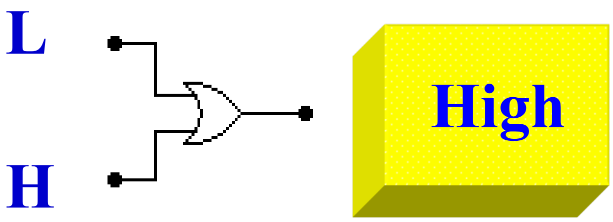
Checkpoints

What is the output of the OR gate?





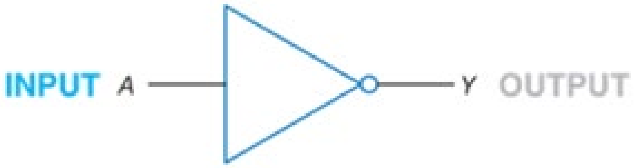




Unique Output: Output LOW only when all inputs are LOW.

NOT Gate (Inverter) and Buffer

- NOT gate operates in such a way that its output is always an **inversion** of its input
- Only one input and one output

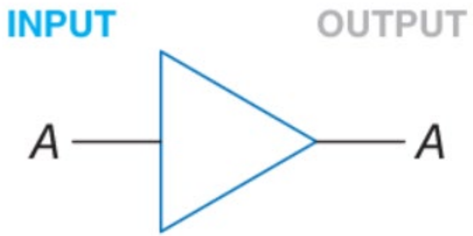


$Y = \overline{A}$ NOT symbol

$Y = A'$ Alternate NOT symbol

A	Y
0	1
1	0

- Buffer gate operates in such a way that its output is **same** as its input



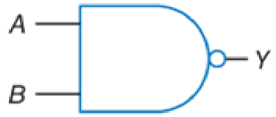
$Y = A$ buffer symbol

A	Y
0	0
1	1

NAND Gate

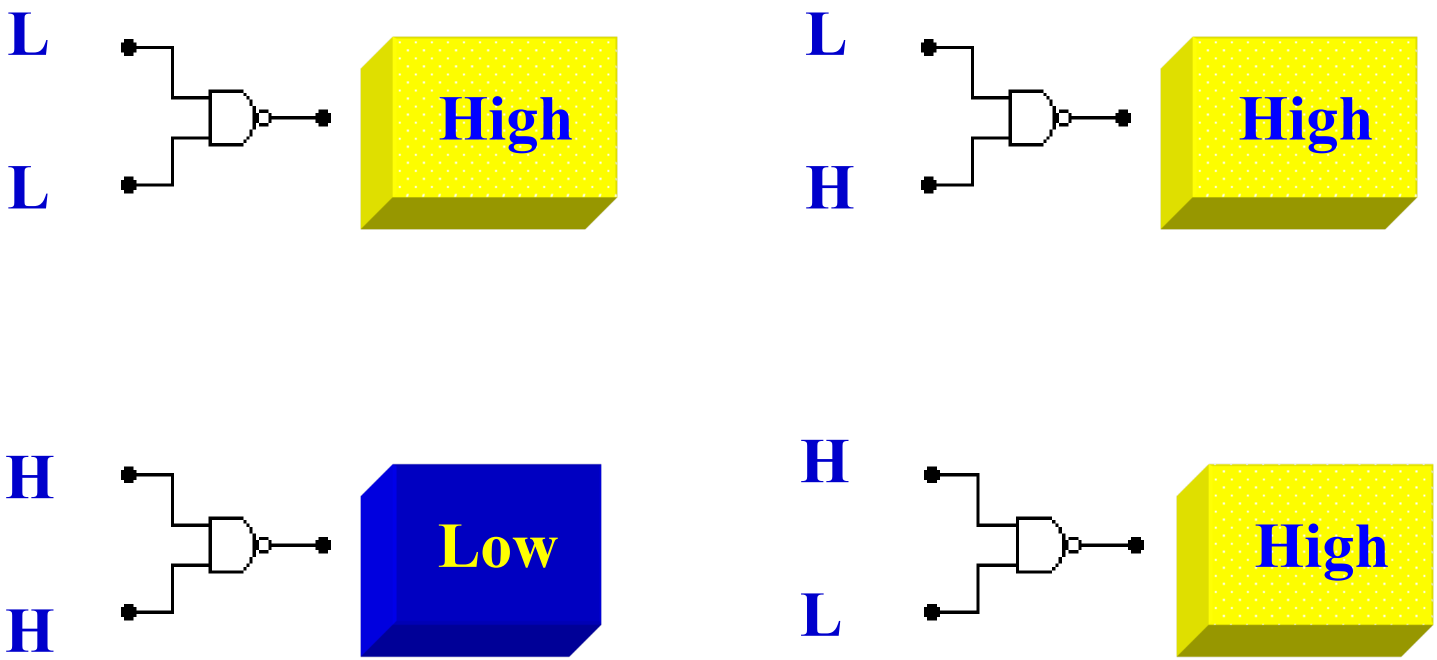
- NAND Gate is a **NOT AND** or an inverted AND function.
- NAND gate operates in such a way that its output is LOW only when all its inputs are HIGH; else the output is HIGH

4 ways to express the logical NANDing of A and B.

<table>
<tr><td>In the English language</td><td>Input <i>A</i> is NANDed with input <i>B</i> yielding output <i>Y</i>.</td></tr>
<tr><td>As a Boolean expression</td><td>NOT symbol
$\overline{A \cdot B} = Y$ or
AND symbol
$\overline{AB} = Y$ or $(AB)' = Y$</td></tr>
<tr><td>As a logic symbol</td><td>
</td></tr>
<tr><td>As a truth table</td><td><table>
<tr><th>A</th><th>B</th><th>Y</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table></td></tr>
</table>

TEST

What is the output of the NAND gate?



L

L

High

L

H

High

H

H

Low

H

L

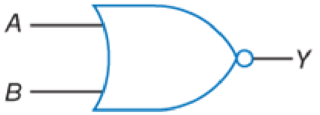
High

Unique Output: Output LOW only when all inputs are HIGH.

NOR Gate

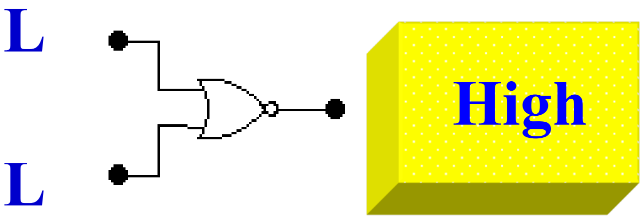
- NOR Gate is a **NOT OR** or an inverted OR function.
- NOR gate operates in such a the way that its output is LOW if when one or more inputs are HIGH; output will be HIGH when all inputs are LOW

4 ways to express the logical NORing of A and B.

<table>
<tr><td>In the English language</td><td>Input <i>A</i> is NORed with input <i>B</i> yielding output <i>Y</i>.</td></tr>
<tr><td>As a Boolean expression</td><td>NOT symbol
$\overline{A+B}=Y$ or
OR symbol
$(A+B)' = Y$</td></tr>
<tr><td>As a logic symbol</td><td> </td></tr>
<tr><td>As a truth table</td><td><table>
<tr><th><i>A</i></th><th><i>B</i></th><th><i>Y</i></th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table></td></tr>
</table>

TEST

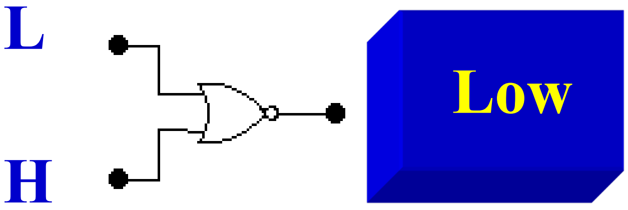
What is the output of the NOR gate?



L

L

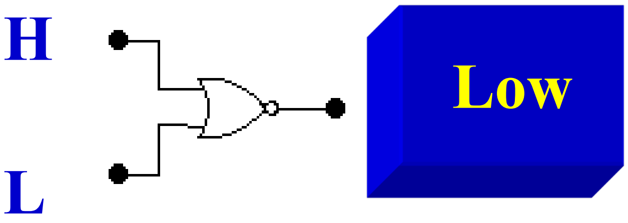
High



L

H

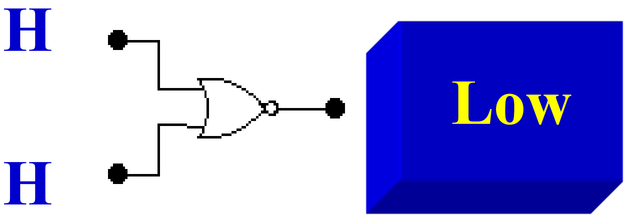
Low



H

L

Low



H

H

Low

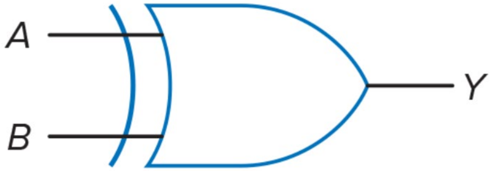
Unique Output: Output HIGH when all inputs are LOW.

XOR Gate

- Exclusive OR Gate or “Anything but not all”: Output is HIGH if only when an **odd number** of inputs are HIGH.

- 4 ways to express the logical XORing of A and B:

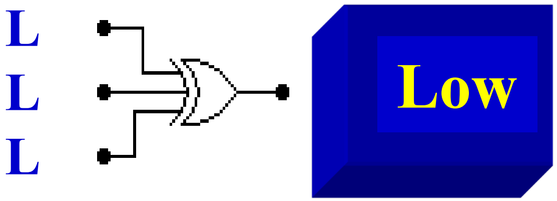
Describing the XOR Function

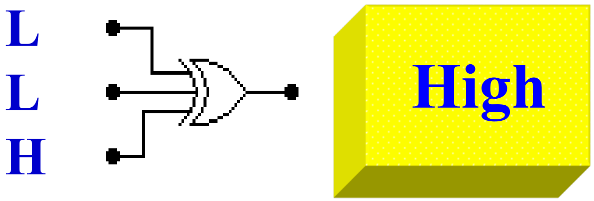
In the English language	Inputs A and B are XORed yielding output Y	As a truth table
As a Boolean expression	$A \oplus B = Y$ XOR symbol	
As a logic symbol		

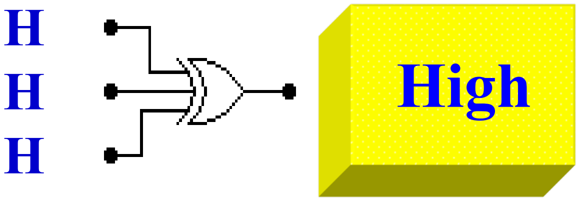
INPUTS		OUTPUT	
A	B	OR	XOR
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

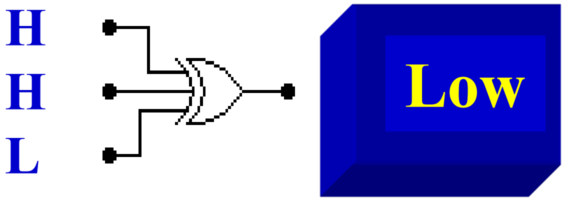
Checkpoints

What is the output from the XOR gate?







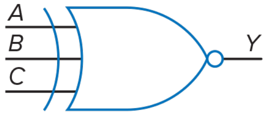


XOR output is HIGH only when odd number of inputs are HIGH

XNOR Gate

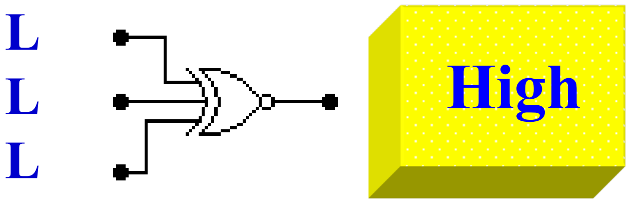
- Exclusive NOR Gate or Inverted XOR:
  Output is **LOW** if only when an **odd number of inputs** are **HIGH**, which is the opposite of XOR gate

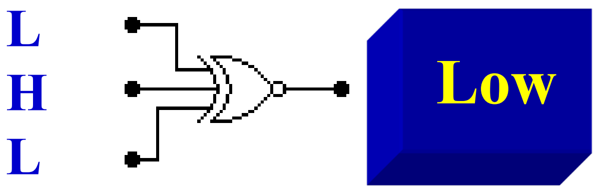
- 4 ways to express the logical XNORing of A, B and C:

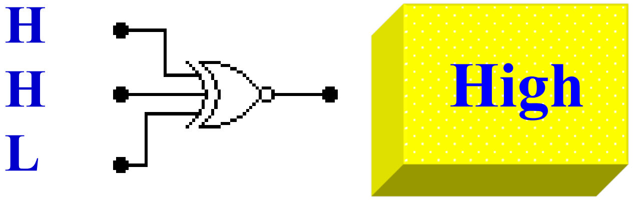
<table>
<tr><td>In the English language</td><td>Inputs <i>A</i>, <i>B</i>, and <i>C</i> are XNORed yielding output <i>Y</i>.</td></tr>
<tr><td>As a Boolean expression</td><td>NOT symbol
$\overline{A \oplus B \oplus C} = Y$
XOR symbol</td></tr>
<tr><td>As a logic symbol</td><td>
</td></tr>
<tr><td>As a truth table</td><td>
<table>
<tr><th colspan="3">INPUTS</th><th colspan="2">OUTPUT</th></tr>
<tr><th>A</th><th>B</th><th>C</th><th>XOR</th><th>XNOR</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

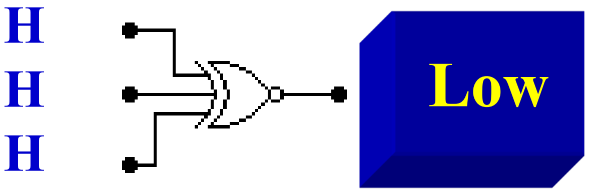
TEST

What is the output from this XNOR gate?





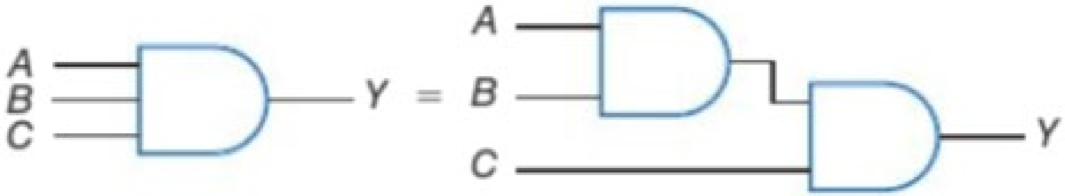




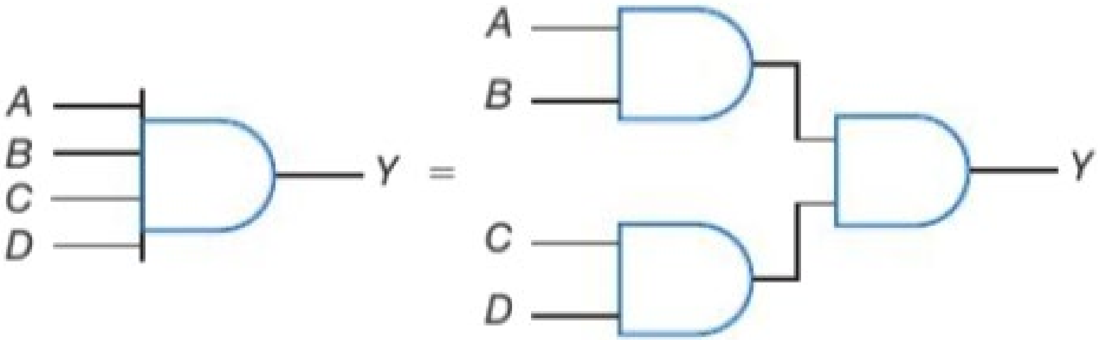
XNOR output is HIGH only when odd number of inputs are LOW

AND Gates with more than 2 inputs

- The Boolean expression A . B . C = Y is illustrated below
- A 3-input AND gate can be re-wired using two 2-input AND gate

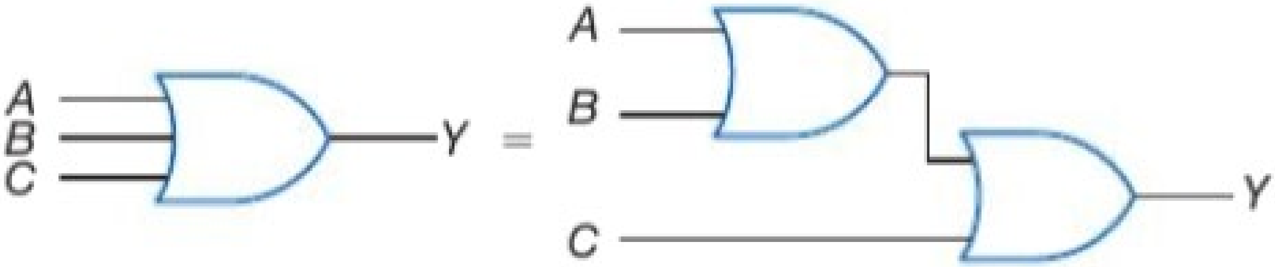


- The Boolean expression A . B . C . D = Y is illustrated below
- A 4-input AND gate can be re-wired using three 2-input AND gates.

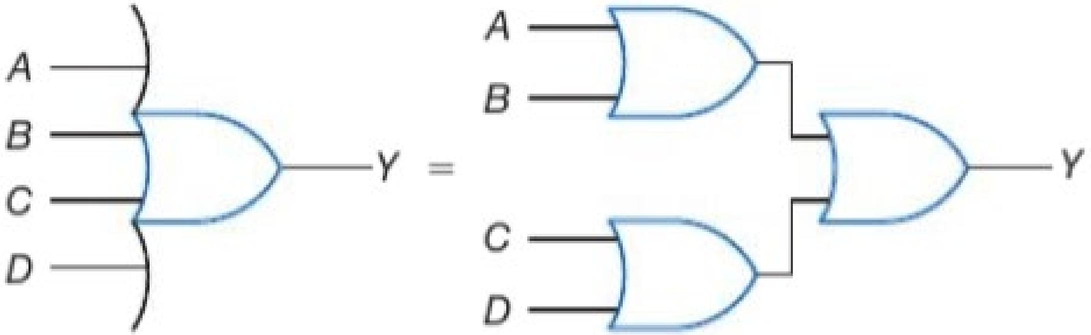


OR Gates with more than 2 inputs

- The Boolean expression A + B + C = Y is illustrated below
- A 3-input OR gate can be re-wired using two 2-input OR gate

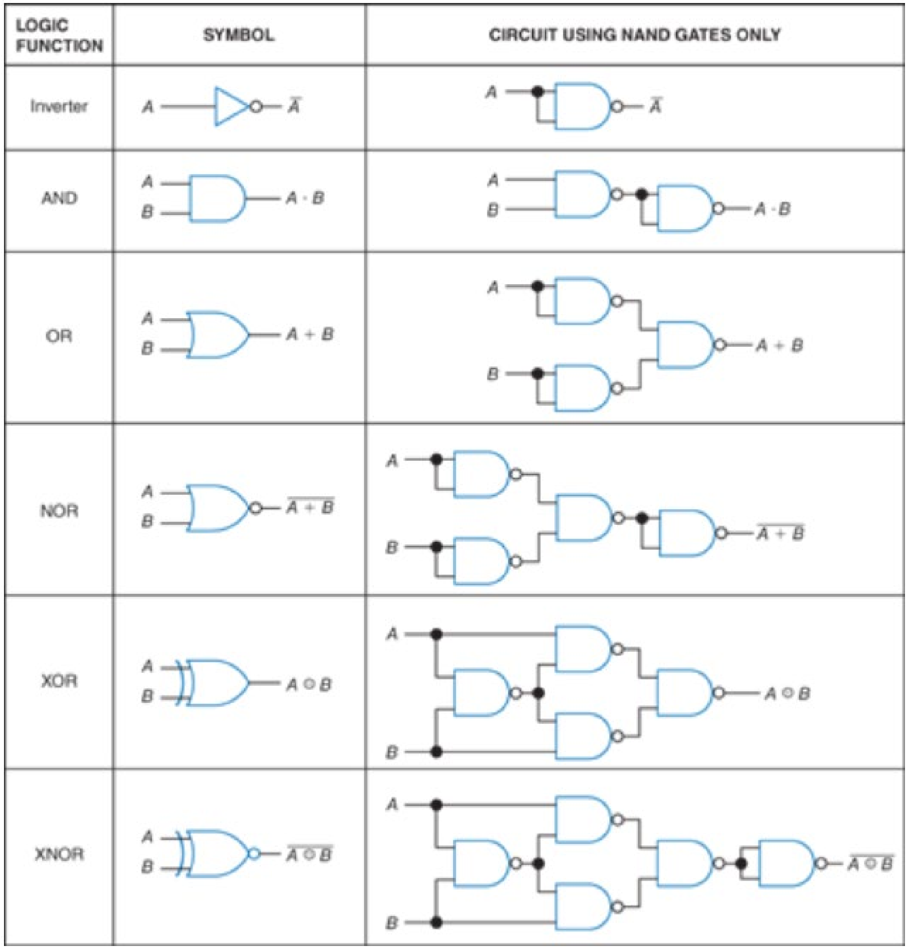


- The Boolean expression A + B + C + D = Y is illustrated below
- A 4-input OR gate can be re-wired using three 2-input OR gates.



Universal Gate – NAND Gate

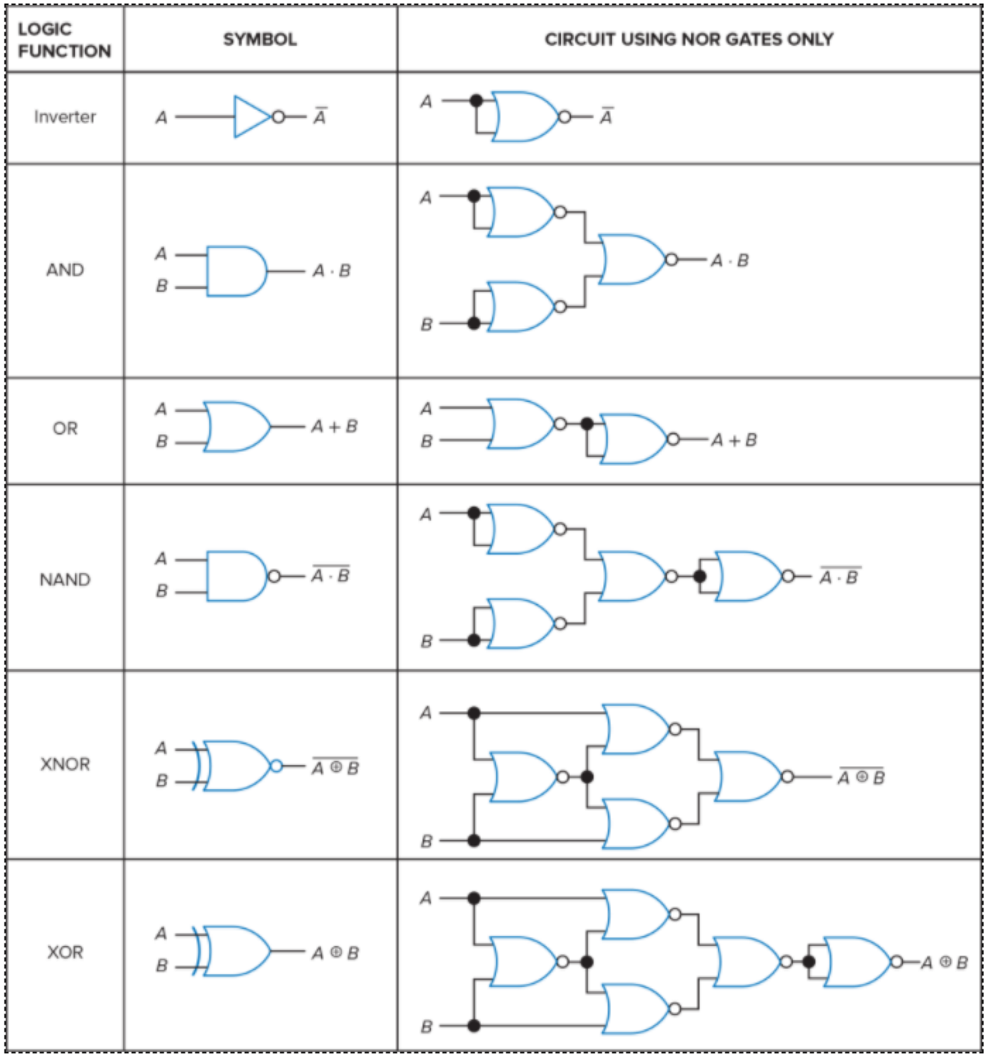
- There are 7 types of gating circuits: AND, OR, NAND, NOR, XOR and XNOR gates and the inverter.
- NAND gate is more widely available than other gates as it can duplicate the output of the other logic gates and provide the same logic function.



LOGIC FUNCTION	SYMBOL	CIRCUIT USING NAND GATES ONLY
Inverter	A $\overline{A}$	A $\overline{A}$
AND	A B $A \cdot B$	A B $A \cdot B$
OR	A B $A + B$	A B $A + B$
NOR	A B $\overline{A + B}$	A B $\overline{A + B}$
XOR	A B $A \oplus B$	A B $A \oplus B$
XNOR	A B $\overline{A \oplus B}$	A B $\overline{A \oplus B}$

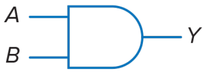
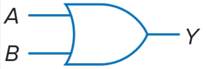
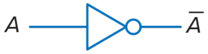
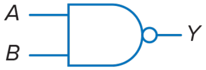
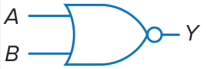
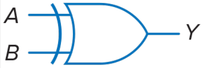
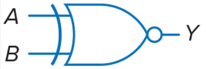
Universal Gate – NOR Gate

- Like the NAND gate, NOR gate can produce a duplicate output of other logic gates.



LOGIC FUNCTION	SYMBOL	CIRCUIT USING NOR GATES ONLY
Inverter	A $\overline{A}$	A $\overline{A}$
AND	A B $A \cdot B$	A B $A \cdot B$
OR	A B $A + B$	A B $A + B$
NAND	A B $\overline{A \cdot B}$	A B $\overline{A \cdot B}$
XNOR	A B $\overline{A \oplus B}$	A B $\overline{A \oplus B}$
XOR	A B $A \oplus B$	A B $A \oplus B$

Logic Gate Summary

LOGIC FUNCTION	LOGIC SYMBOL	BOOLEAN EXPRESSION	TRUTH TABLE		
			INPUTS		OUTPUT
			A	*B*	*Y*
AND		$A \cdot B = Y$	0	0	0
			0	1	0
			1	0	0
			1	1	1
OR		$A + B = Y$	0	0	0
			0	1	1
			1	0	1
			1	1	1
Inverter		$A = \overline{A}$	0		1
			1		0
NAND		$\overline{A \cdot B} = Y$	0	0	1
			0	1	1
			1	0	1
			1	1	0
NOR		$\overline{A + B} = Y$	0	0	1
			0	1	0
			1	0	0
			1	1	0
XOR		$A \oplus B = Y$	0	0	0
			0	1	1
			1	0	1
			1	1	0
XNOR		$\overline{A \oplus B} = Y$	0	0	1
			0	1	0
			1	0	0
			1	1	1

Activity 2: Google Jamboard +

Team Work followed by Class Discussion

Let a student in the team to login to https://jamboard.google.com/

Click + to create a new "Jam" and invite all teammates to join.

Fill up the below table for a 2-input and 3-input gate assigned to your team. Discuss the differences between each gate.

Team	Gate
1	AND, NOR, XOR
2	AND, NOR, XOR
3	AND, NOR, XOR
4	OR, NAND, XNOR
5	OR, NAND, XNOR

As a Boolean expression	
As a logic symbol	
As a truth table	

(*Label the inputs: A, B and C and the output X and the expression)

Consultation Session:
CA1 Problem Statement

Consultation with lecturer

Complete the **Week 01** section of the Problem Statement (individual work)
AY2024 Sem1 E105_CA1_PS.docx

You can consult your lecturer if you have any questions.