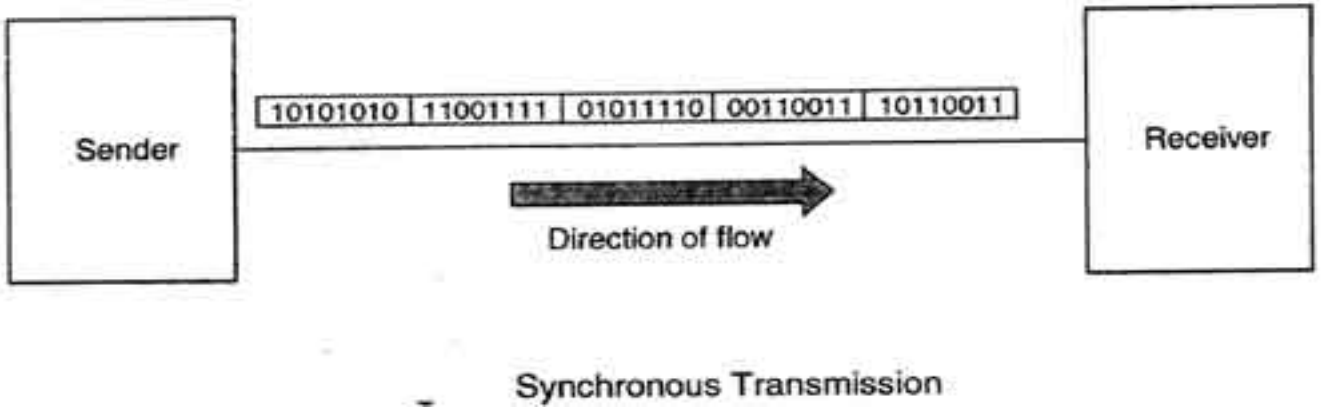
# WHY BINARY??



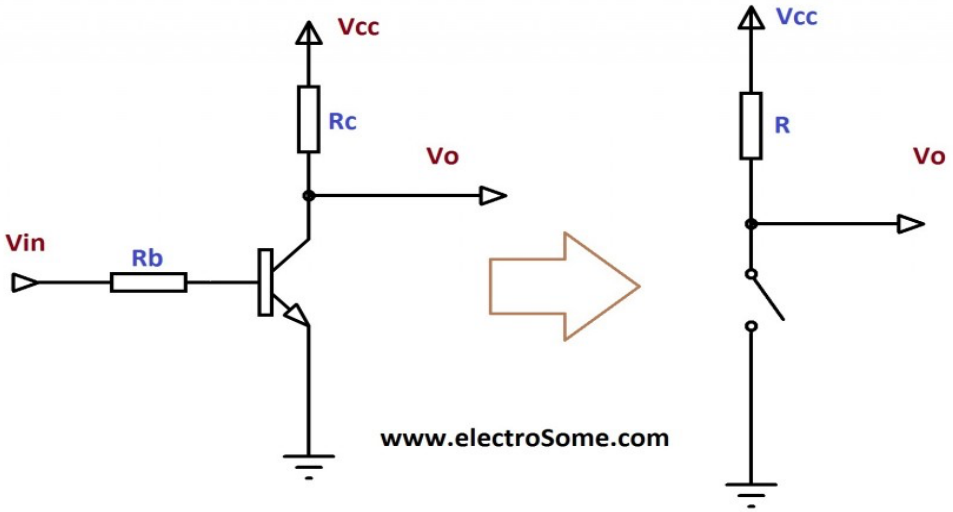

Synchronous Transmission

This is how digital data is sent over a wire (PHYSICAL layer), encoded as ‘0’ and ‘1’s!

# KEY POINTS

- Conversion between decimal, binary, and/or Hexadecimal
- Parity (Odd/Even)
- Endianess (Big/Small/Internet)
- Binary AND operations

# Why use digital?


Digital (using 0 and 1) is the basic building block (transistor) of all electronics. Just like a light switch, the transistor is, typically, on or off. From there, much more complex circuits are made!

# To talk about Binary, let's talk about normal numbers (Decimal)

- $904_{10} = 9 \times 10^2 + 0 \times 10^1 + 4 \times 10^0$
- $904_{10} = 9 \times 10^2 + 4 \times 10^0$

| | | | | | | |
|---|---|---|---|---|---|---|
| $10^n$ | 100000 | 10000 | 1000 | 100 | 10 | 1 |

## Positional number system

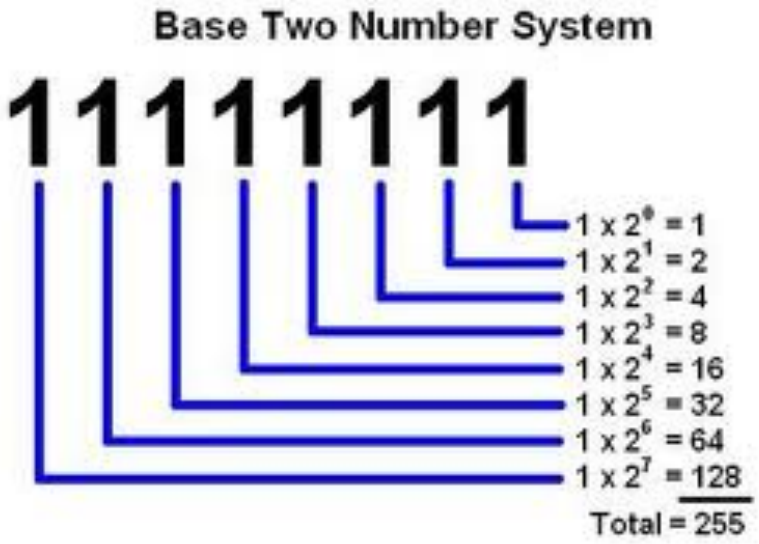
# Decimal Position

| thousand's place | hundred's place | ten's place | one's place |
|---|---|---|---|
| $10^3$ | $10^2$ | $10^1$ | $10^0$ |
| 0 | 4 | 3 | 2 |

$$4 \times 10^2 = 400$$
$$+ \ 3 \times 10^1 = 30$$
$$+ \ 2 \times 10^0 = 2$$
$$\overline{\qquad\qquad}$$
$$432$$

# Binary terminology

- Binary uses only 1's and 0's as symbols
- A binary digit is called a bit
- Eight binary digits together is a byte or an octet



# Number Base

- Base 2 is binary
- Base 10 is decimal
- Base 16 is hexadecimal

- Each is positional!

# Why Hexadecimal?

- Binary is too many digits to display.
- Need easier format for same values.
- Hexadecimal digit is represented by 4 binary digits exactly.
- Represented by 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F (16 values)
- There is no ‘G’, 16 decimal is 10 hexadecimal.

| | | |
|---|---|---|
| 0 | 0000 | 0x0 |
| 1 | 0001 | 0x1 |
| 2 | 0010 | 0x2 |
| 3 | 0011 | 0x3 |
| 4 | 0100 | 0x4 |
| 5 | 0101 | 0x5 |
| 6 | 0110 | 0x6 |
| 7 | 0111 | 0x7 |
| 8 | 1000 | 0x8 |
| 9 | 1001 | 0x9 |
| 10 | 1010 | 0xA |
| 11 | 1011 | 0xB |
| 12 | 1100 | 0xC |
| 13 | 1101 | 0xD |
| 14 | 1110 | 0xE |
| 15 | 1111 | 0xF |

Simple table of Decimal/Binary/Hex.

Hex numbers are traditional preceeded by ‘0x’ to denote them as hexadecimal.

The decimal value of 16 is represented in hex as ‘0x10’, NOT as ‘0xG’!

$$1000\ 1010_2 = 8A_{16}$$

8 A

# Binary, Decimal, and Hexadecimal is a positional number system

| | | | | | | |
|---|---|---|---|---|---|---|
| $10^n$ | 100000 | 10000 | 1000 | 100 | 10 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| $10^n$ | 100000 | 10000 | 1000 | 100 | 10 | 1 |
| $2^n$ | 32 | 16 | 8 | 4 | 2 | 1 |
| $16^n$ | | | 4096 | 256 | 16 | 1 |

# Aside, nice, pretty round numbers in number system aren't necessarily so in another:

| | | | | |
|---|---|---|---|---|
| $10^n$ (Decimal) | $1000_{10}$ | $100_{10}$ | $10_{10}$ | 1 |
| $2^n$ (Binary) | $1111101000_2$ | $1100100_2$ | $1010_2$ | 1 |
| $16^n$ (Hexadecimal) | $3E8_{16}$ | $3A_{16}$ | $A_{16}$ | 1 |

# How *I* calculate binary to decimal

Step 1  On a blank sheet of paper, on the far right, write the number ‘1’.  Then left of it, write ‘2’, then ‘4’, and keep doubling as necessary (at least as many digits in your binary number).

Step 2  Beneath this write out your binary number.  If the binary number has less digits, pad ‘0’ on the left.

Step 3  Add the corresponding decimal value for every ‘1’ in the binary number.

# Homework: Convert these numbers

Q1. What is the decimal value of the binary number:

a. 1001
b. 1111
c. 100011
d. 111100

From step 1, I will write out 6 decimal digits, as (Q1c) has the most binary digits.

Remember, we start with ‘1’ and double each step to the left. Sounds awkward, but decimal does the same thing, except in powers of 10 (1, 10, 100, 1000, …)

# Homework: Convert these numbers

Q1. What is the decimal value of the binary number:

a. 1001

b. 1111

c. 100011

At the rightmost side, write ‘1’


To the left, since this is binary, double the preceding value

To the left of the prior number, write ‘2’

# Homework: Convert these numbers, pt 2

Continue to double the digit on the left until done:

| 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|

Write the binary number under the corresponding blocks. Remember to left pad '0' as necessary. Lets try '1001'.

| 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 |

Add the decimal numbers above the binary '1' values. In this example, 1001 (binary) = 8 + 1 = 9 (in decimal)

# How ‘I’ convert Hex to Decimal

Just like in binary, but use 16.  leftmost position is ‘1’, next is ‘16’, then ‘256’.

| 256 | 16 | 1 |
|---|---|---|
| | | |

13A (hex) = 1*256 + 3*16 + 10*1 = 314 (decimal)

0BF (hex)= 0*256 + 11*16 + 15*16 =  191 (decimal)

# And how do you get the binary for a "normal" number?

- Subtracting Method: Subtract the largest power of two you can until you get to 0
- Dividing Method: Divide by 2 (writing down the remainder) until the number is gone.

# How ‘I’ convert decimal to binary

1. Start like you are converting binary to decimal, with your table

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|

2. Take a decimal number. Subtract it from the highest number. If your decimal number is less than the table number, put ‘0’.
3. If the decimal number is equal or less, write a ‘1’ on the table.
4. Continue steps 2,3 until you have zero left.

# How ‘I’ convert decimal to binary example

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|

For this example, take a decimal number (say 79).
Subtract it from the highest number. Since 128 is greater than 79, we write ‘0’.
Since 79 is greater than 64, we write ‘1’ under the 64 spot. This gives us 79-64=15.
15 is less than 32, so we write ‘0’ under 32 column.
15 is less than 16, again we write ‘0’ under 16 column.
15 is more than 8, so we write ‘1’ under 8 column. 15-8 gives us 7.
7 is more than 4, so we write a ‘1’ under the 4 column. 7-4=3
3 is more than 2, so we write a ‘1’ under the 2 column. 3-2=1
1 is equal to 1, so we write a ‘1’ under the 1 column. 1-1=0
We are now done

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

# Skills you should practice and master

1. Count in binary.
   0000, 0001, 0010, 0011, 0100, …
2. Count in Hex. Understand the symbols which represent decimal values 10-15. 0,1,2, … 9, A, B, C, D, E, F
3. Understand how 'AND' works
4. Convert freely between binary, decimal and hexadecimal

These are the skills you will use through out the course, many times over.

# So let's look at some Binary numbers

$$101101_2 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$101101_2 = 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0$$

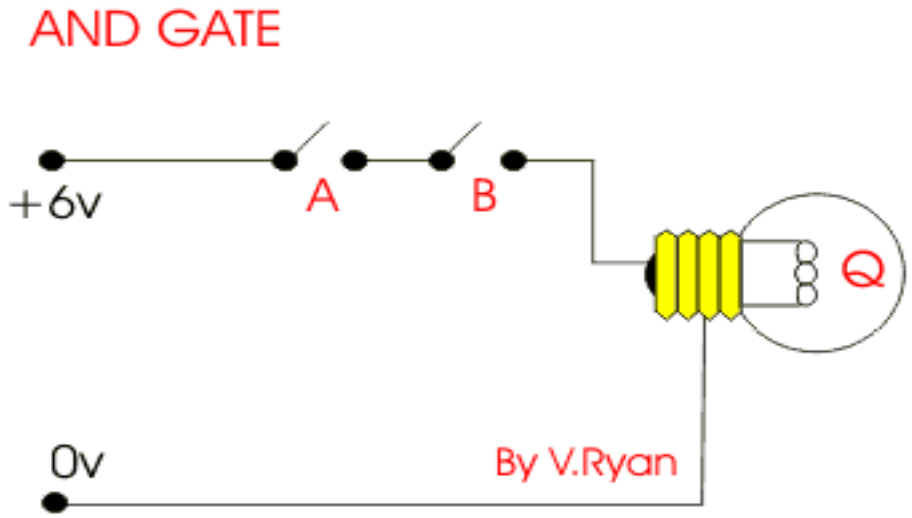
$$= 32 + 8 + 4 + 1 = 45$$

| | | | | | | |
|---|---|---|---|---|---|---|
| $\mathbf{2^n}$ (Binary) | $100000_2$ | $10000_2$ | $1000_2$ | $100_2$ | $10_2$ | 0 |
| $2^n$ (Decimal) | $32_{10}$ | $16_{10}$ | $8_{10}$ | $4_{10}$ | $2_{10}$ | 0 |

# Hexadecimal will help us

| | |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | A |
| 11 | B |
| 12 | C |
| 13 | D |
| 14 | E |
| 15 | F |

# Simple digital example




We wont get into digital gate, but use this as a simple example how two switches (transistors) can form a simple AND gate.

# Logic

| X | Y | X and Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| Bitwise **AND** | Bitwise **AND** | Bitwise **AND** | Bitwise **AND** |
|---|---|---|---|
| X 1100 | X 1101 | X 0101 | X 1111 |
| Y 1010 | Y 1010 | Y 1010 | Y 1010 |
| X•Y 1000 | X•Y 1000 | X•Y 0000 | X•Y 1010 |

- • means “and”

# What We Talked About Today

- The Doubles … 128 64 32 16 8 4 2 1
- The decimal value of a binary number
- The binary form of a decimal value
- The binary/hexadecimal table
- The binary form of a hex number
- The hex form of a binary number
- AND masking
- Studying with a blank piece of paper

# What is Parity Bit?

Wikipedia for Parity Bit:
A **parity bit**, or **check bit** is a bit added to the end of a string of binary code that indicates whether the number of bits in the string with the value one is even or odd. Parity bits are used as the simplest form of error detecting code.
There are two variants of parity bits: **even parity bit** and **odd parity bit**.

# What is Parity Bit?

https://biesnecker.com/2014/10/08/how-do-you-calculate-a-parity-bit/ for Parity Bit:

The parity of a string of binary data indicates whether the number of set bits in the string is odd or even. There are two kinds of -- *even parity* and *odd parity*. In *even parity* the parity bit is 1 if the number of 1s in the binary string is *odd*, thereby making the number of 1s in the full binary string *even*. In *odd parity* the parity bit is 1 if the number of 1s in the binary string is *even*, thereby making the number of 1s in the full binary string *odd*.

# Why use Parity Bits?

Parity bit is used to identify bit changes in data. By using a Parity Bit, you can identify data which has changed, due to errors

1. Random errors from COSMIC RAYS! You might get FANTASIC super powers!
2. Bad switch memory
3. Noise from the medium/Physical layer

# How to Calculate Parity Bit?

Step 1. Count the number of 1's in the binary word.

Step 2 A. If using Even Parity, and the number of 1's is even, parity is '0'. Otherwise it is '1'.

Step 2 B. If using Odd Parity, and the number of 1's is even, parity is '1'. Otherwise it is '0'.

Parity is used to keep the total number of 1's in a binary word to be even/odd (depending on the parity type)

# Parity Bit Examples

| 7 bits of data | Count of ‘1’ bits | Even Parity bit | Odd Parity bit |
|---|---|---|---|
| 0000000 | 0 | 0 | 1 |
| 1010001 | 3 | 1 | 0 |
| 1101001 | 4 | 0 | 1 |
| 1111111 | 7 | 1 | 0 |

# Parity Bit for YOU to Calculate

| 7 bits of data | Count of bits | Even Parity bit | Odd Parity bit |
|---|---|---|---|
| 0000001 | | | |
| 1110001 | | | |
| 001101001 | | | |
| 111111111 | | | |
| | | | |

# Parity Bit Answers

| 7 bits of data | Count of ‘1’ bits | Even Parity bit | Odd Parity bit |
|---|---|---|---|
| 0000001 | 1 | 1 | 0 |
| 1110001 | 4 | 0 | 1 |
| 001101001 | 4 | 0 | 1 |
| 111111111 | 9 | 1 | 0 |
| | | | |

Silly Raptor, Even and Odd Parity bits will ALWAYS be opposite of each other!

# Limitations of Parity Bits?

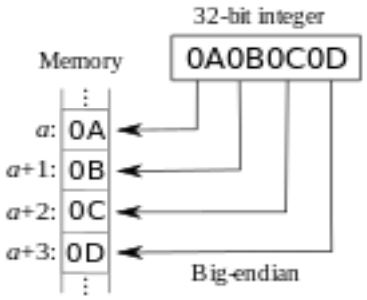
Parity bits can only detect errors for an odd number of changes. Even number of changes cant be detected!!

| Original Data | Error'd Data | Original Even Parity | Error'd Even Parity | # bits of error |
|---|---|---|---|---|
| 1001 | 1100 | 0 | 0 | 2 |
| 1001 | 1110 | 0 | 1 | 3 |
| 1001 | 1101 | 0 | 1 | 1 |

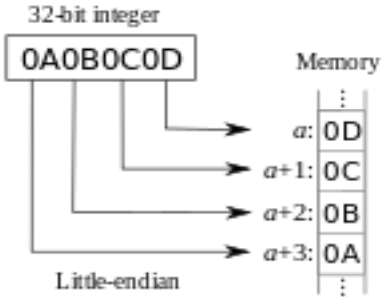
# ENDIANNESS

Endianness refers to the order of the bytes, comprising a digital word, in computer memory. It also describes the order of byte transmission over a digital link. Words may be represented in big-endian or little-endian format. With big-endian the most-significant byte of a word is stored at a particular memory address and the subsequent bytes are stored in the following higher memory addresses, the least significant byte thus being stored at the highest memory address. Little-endian format reverses the order and stores the least-significant byte at the lower memory address with the most significant byte being stored at the highest memory address. -- wikipedia

# ENDIANNESS




Big Endian : Most significant (byte representing largest value) is first, or at a lower address in memory.




Small Endian : Least significant (byte representing the smallest value) is first, or at a lower address in memory.

# ENDIANNESS

How does Endianess fit into the internet?

Data is sent in what order, byte wise?

Each bit of the byte is sent in what order?

# ENDIANNESS

Networks use ‘Network Byte Order’. Technically, this is large Endian format for Bytes. Thus, with a number like 0xABCD, 0xA is sent first, and 0xD is sent last.

Likewise, bit wise, the most significant bit is sent first. Thus, $170_{10}$ or $10101010_2$, we send a ‘1’ first, and the ‘0’ last.

# ENDIANNESS

From our last example, if we sent $170_{10}$ or $10101010_{2}$: