Topic: 1.1.1 Binary systems

Theory of Computer Science

The expansion of the computer systems has been witnessed by all ages today from the large room-sized systems seen in old sci-fi movies to the recent versions of tablets and Laptop computers. It must be understood that a long pathway has been established in reaching this age of technology and discreteness in size and memory.

1.1 DATA REPRESENTATION

Data Representation is basically the methods used within the computer systems to represent information to be stored. Computers store different types of information:

- Numbers
- Text
- Graphics (including photo stills, videos, animations)
- Audio

They may seem very different to us. On the contrary, all types of information stored in a computer are stored internally in the same simple format: a sequence of 0's and 1's. You may be confused at this point as to how can a sequence of 0's and 1's represent things as distinct as your selfie on the webcam, your favorite track, your favorite movie, and your boring exam paper?

It all comes back to how we *interpret* the information. Computers use numeric codes to correspond to all the information they store. These codes are related to those you may have used as a child to encrypt secret notes: like a single finger represents a 1 and 7 fingers held together represent a 7 in decimal numbering. Any written message can be represented numerically with this code. The codes used by computers are a tad more complicated, and they are based on the binary number system (base two) instead of the more recognizable decimal system (base ten). Computers use a variety of different codes used for numbers, text, and still others for sound and graphics.

1.1.1 BINARY SYSTEMS

RECOGNISE THE USE OF BINARY NUMBERS IN COMPUTER SYSTEMS

FUN FACT: The base 10 (decimal) systems are sometimes called denary, which is more related with the name binary for the base 2 system. The word "denary" also refers to the Roman denarius coin, which was worth ten asses (an "ass" was a copper or bronze coin).

WHAT IS BINARY NUMBER SYSTEM?

Normally we write numbers using digits 0 to 9. This is called base 10. However, any positive integer (whole number) can be easily represented in binary by a sequence of 0's and 1's. Numbers in this form are said to be in base 2 and they are called "binary numbers". Base 10 numbers use a positional system based on powers of 10 to indicate their value. The number 123 is really 1 hundred + 2 tens + 3 ones. The value of each position is determined by ever-higher powers of 10, read from left to right. Base 2 works the same way, but instead of base ten, binary uses base two.

Example:

The number 101 in base 2 is really:

$$1 \text{ four} + 0 \text{ twos} + 1 \text{ one} = (5)_{10}$$

This representation was created so as to help the computers in identifying 1 as the state of 'on' and 0 as state of 'off'.

WHY USE BINARY NUMBER SYSTEMS:

The microprocessor makes use of transistors that basically identify voltage levels rather than any affirmative value in decimal number system. Therefore a voltage level 'high' or 'on' will be identified as '1' and a voltage level of 'low' or 'off' will be identified as '0'. Normally high voltage is classified as 5V or 3.3V whereas low voltage is treated as 0V. Some hardware may use different mode of identification for binary number levels like in the case of CD-ROMs, certain microscopic black spot will be considered as binary number '0' while a shiny spot reflecting light will be considered as on. Hard disks basically apply the law of magnetism whereas static memory utilizes electric charges on passive devices like capacitors for recognizing number systems.

The numerous patterns of 0's and 1's are obtained from the devices or the computer's internal hardware corresponds to various representations of numbers in decimal and other formats.

Topic: 1.1.1 Binary systems

CONVERT DENARY NUMBERS INTO BINARY AND BINARY NUMBERS INTO DENARY

UNDERSTANDING "BITS":

For the sake of understanding, binary numbers can be broken down into their smallest representation called **bits**

The easiest way to understand bits is to think of them as digits like we learned in first grade. A digit is a single place that can hold numerical values between 0 and 9. Digits are normally combined together in groups to create larger numbers. For example, "6,357" has four digits. It is understood that in the number 6,357;

- -The 7 is filling the "1s place"
- -The 5 is filling the "10s place"
- The 3 is filling the "100s place"
- The 6 is filling the "1,000s place"

So you could express things this way if you wanted to be clear:
(6 * 1000) + (3 * 100) + (5 * 10) + (7 * 1) = 6000 + 300 + 50 + 7 = 6357

Another way to express it would be to use powers of 10. Assuming that we are going to represent the concept of "raised to the power of" with the "^" symbol (so "10 squared" is written as "10^2"), another way to express it is like this:

(6 * 10^3) + (3 * 10^2) + (5 * 10^1) + (7 * 10^0) = 6000 + 300 + 50 + 7 = 6357

What you can see from this expression is that each digit is a placeholder for the next higher power of 10, starting in the first digit with 10 raised to the power of zero. The power of ten decreases from the highest weighted digit (on the L.H.S.) to the least weighted digit (on the R.H.S.).

The binary number system works exactly the same way as the decimal system, except that it contains only two digits, 0 and 1. Like this "1011". How do you figure out what the value of the binary number "1011" is? You do it in the same way we did it above for 6357, but you use a base of 2 instead of a base of 10. So:

(1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (1 * 2^0) = 8 + 0 + 2 + 1 = 11

You can see that in binary numbers, each bit holds the value of increasing powers of 2.

Topic: 1.1.1 Binary systems

CONVERSION TECHNIQUES:

Let us now understand the basic conversion techniques that basically run in the background of every computer system that is conversion from denary to binary and binary to denary number systems.

DENARY TO BINARY:

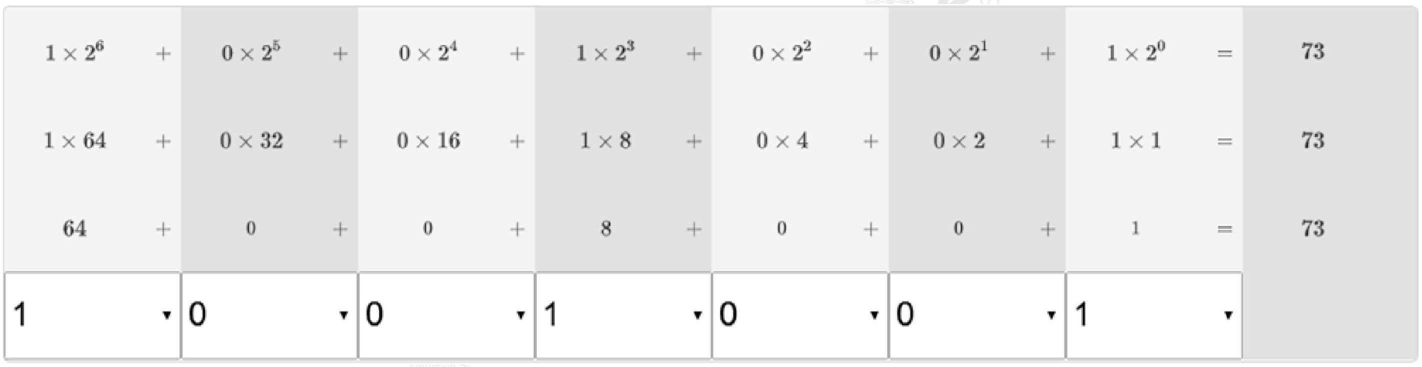
Binary number system makes use of bits and 8 bits comprise of a larger unit called byte.
In the decimal system the number 34567 could be represented as:

10^4	10^3	10^2	10^1	10^0
3	4	5	6	7

In binary the number 11001 could be written as:

2^4	2^3	2^2	2^1	2^0
1	1	0	0	1

EXAMPLE OF A SIMPLE CONVERSION OF $(73)_{10}$:



1×2^6	+	0×2^5	+	0×2^4	+	1×2^3	+	0×2^2	+	0×2^1	+	1×2^0	=	73
1×64	+	0×32	+	0×16	+	1×8	+	0×4	+	0×2	+	1×1	=	73
64	+	0	+	0	+	8	+	0	+	0	+	1	=	73
1 ▾		0 ▾		0 ▾		1 ▾		0 ▾		0 ▾		1 ▾		

Example:
Let us consider a number 98 and convert it into binary system and take it step by step:
To do this, we should check the following:

- Does the number go into 128? No. so this becomes a 0.
- Does it go into 64? Yes, so this becomes a 1.
- Does the remaining 34 go into 32? Yes so this is also a 1
- Does the remaining 2 go in 16, 8, 4, or 1? No so these are all 0's.
- Does the remaining 2 go into 2? Yes, so this is a 1.
- Since there is nothing left to compare with 1, it becomes a 0 by default.

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

98 in binary = 01100010
The largest number we can make with 1 byte is 255. So if we wanted to make a number which is larger, we would need to add another byte.

BINARY TO DENARY CONVERSION:

The term bit is short for **B**inary dig**it**.
So 8 bits combined together make 1 **byte.**

Given "10010111" as an example.

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

To work this out, you multiply the numbers which are a 1 by 1 and add them together to give you the decimal number
= (128×1) + (16×1) + (4×1) + (2×1) +(1×1)
= 128 + 16 + 4 + 2 + 1
= 151

BYTE AND ITS USE IN MEMORY SIZE

WHAT IS A BYTE?

A byte is a series of 8 bits (enough to represent one alphanumeric character) processed as a single unit of information. A single letter or character would use one byte of memory (8 bits); two characters would use two bytes (16 bits).
In other ways, a bit is either an 'on' or an 'off' which is processed by a computer processor, we represent 'on' as '1' and 'off' as '0'. 8 bits are known as a byte, and it is bytes which are used to pass our information in its basic form "characters".
An alphanumeric character (e.g. a letter or number such as 'A', 'B' or '7') is stored as 1 byte. For example, the letter 'R' uses 1 byte, which is stored by the computer as 8 bits, '01010010'.

A document containing 1000 characters would use 1000 bytes (8000 bits) Note: many non-alphanumeric characters such as symbols (!, @,#,$,etc..) and foreign language characters (Arabic,Japanese,etc..) use multiple bytes.

A kilobyte (KB) is 1024 bytes and a megabyte (MB) is 1024 kilobytes and so on…

Topic: 1.1.1 Binary systems

UNIT	ABBREVIATION	STORAGE
Bit	b	Binary Digit, Single 1 or 0
Nibble	-	4 bits
Byte/Octet	B	8 bits
Kilobyte	KB	1024 bytes
Megabyte	MB	1024 KB
Gigabyte	GB	1024 MB
Terabyte	TB	1024 GB
Petabyte	PB	1024 TB
Exabyte	EB	1024 PB
Zettabyte	ZB	1024 EB
Yottabyte	YB	1024 ZB

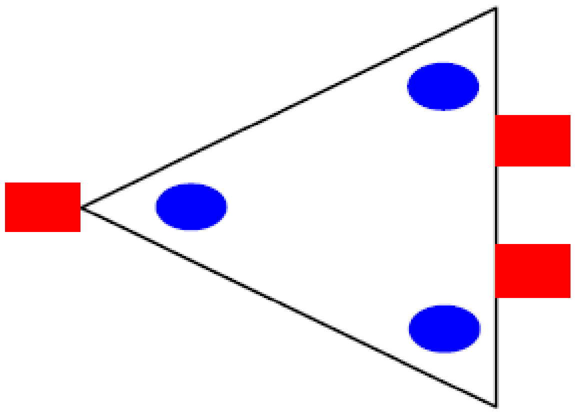
USE OF BINARY SYSTEMS FOR SPECIFIC APPLICATIONS

It is useful to understand that computer systems are capable to perform specific and user controlled and oriented applications like robotics, power systems, and digital instruments and so on. Controlled devices usually contain registers which are made up of binary digits (bits). The following example shows how these registers can be used to control a device.

ROBOTIC SYSTEMS:

Robotic systems rely on processors to access user provided values and data. It is essential to note here that all the data is sent via the processor in a binary format so conversions and storage has to be provided by the processor.

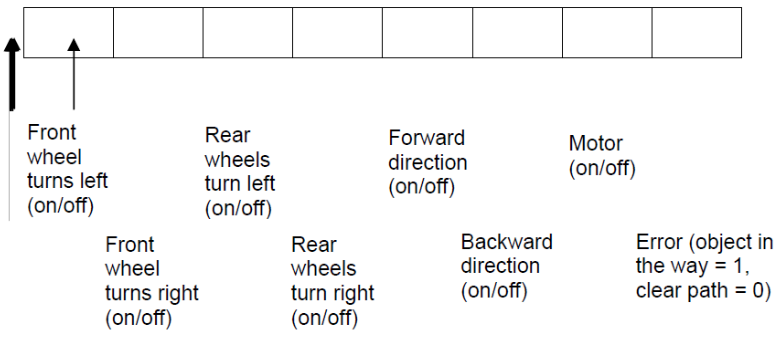
Example



The device on the left is a mobile trolley with 3 wheels. All three wheels can turn left or right and each wheel has its own electric driving motor. Sensors at the front and rear of the trolley detect an object in its path which would subsequently stop all movement. An 8-bit register is used to control the device.

●	these represent the three wheels
🟥	these represent the three sensors

Topic: 1.1.1 Binary systems


Front wheel turns left (on/off)
Front wheel turns right (on/off)
Rear wheels turn left (on/off)
Rear wheels turn right (on/off)
Forward direction (on/off)
Backward direction (on/off)
Motor (on/off)
Error (object in the way = 1, clear path = 0)

(1 = ON and 0 = OFF)

Therefore if the input is for example: 1 0 1 0 1 0 1 0

1	**0**	**1**	**0**	**1**	**0**	**1**	**0**
front wheel turns left	-	Back wheels turn left	-	The direction is forward	-	The motors are on	-

Hence, the trolley is moving forward and turning left.

(i) What does this register mean?

"0 0 0 1 0 1 1 1"

(ii) How would the following be represented using the above register?

- front wheel turning right
- back wheels turning left
- moving in a forward direction
- motors on
- no object in its path

Answers

(i) - front wheel **not** turning left or right

- Rear wheels turning right
- going in backward direction
- motors on
- Error – object in path

Topic: 1.1.1 Binary systems

So the vehicle is going nowhere.

(ii) **0 1 1 0 1 0 1 0**

EXAMPLE#2:
SENSORS IN PRINTING DEVICES:
Three sensors are attached to a printing device, with three alarms attached to the sensors. The first sensor, "A," detects if the device needs ink. The second sensor, "B," detects if the device needs repair. The third sensor, "C," detects if the device has jammed paper. If the device jams or needs repair, alarm 1 sound. If the device jams or is short on ink, alarm 2 sounds. If two or more problems occur at once, alarm 3 sounds.
It can now be implied that:

- A=1 refers to "low ink"
- B=1 refers to "Device needs repair"
- C=1 refers to "device should jam"

The outputs to the system are as follows:

- A1: alarm 1 sounds if B=1 or C=1(either of B=1 or C=1 will result in A1)
- A2: Alarm 2 sounds if C=1 or A=1 (either of A or C being true will result in A2)
- A3= any two or more of A, B, and C being '1' will result in A3.

Let us now look at the cases that will form:

A	B	C	A1	A2	A3
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Topic: 1.1.1 Binary systems

Questions:

1) What binary values can the register hold to alert the user of a "low ink" situation?
2) The register is holding "010". Which alarm will sound?
3) Is the system accurate in telling the user of exactly which problem is being occurred? Point out any one situation where user is misguided?

Answers:

1) 010, 111, 111, 111
2) A2 will sound.
3) No. System has the same alarm for one problem or more than one problem. E.g. 111 could mean that all three problems have occurred but 111 can be received even if printer is "full on ink" or not having "jammed paper."