Logic and Computer Design Fundamentals

Chapter 1 – Digital Systems and Information

Asst.Prof.Dr. Preecha Tangworakitthaworn
Semester 2/2023

Overview

PART 1

- **Digital Systems, Computers, and Beyond**
- **Information Representation**

PART 2

- **Number Systems** [binary, octal and hexadecimal]

PART 3

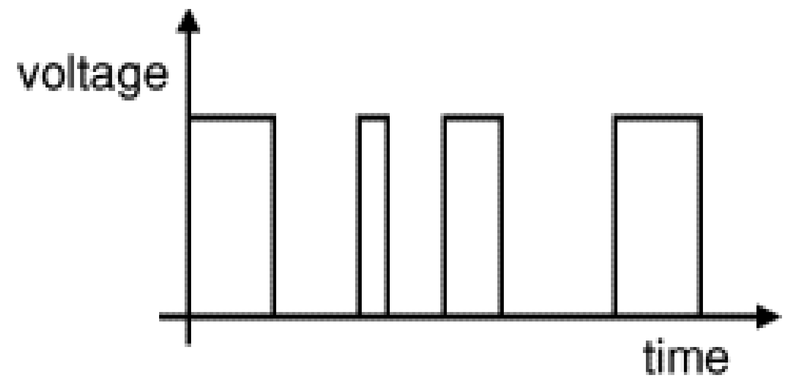
- **Arithmetic Operations**
- **Base Conversion**
- **Decimal Codes** [BCD (binary coded decimal)]

PART I

Introduction

What is digital system?

- Digital systems process digital signals which can take only a limited number of values (discrete steps), usually just two values are used: the positive supply voltage (1) and zero volts (0).



Digital (logic) signal

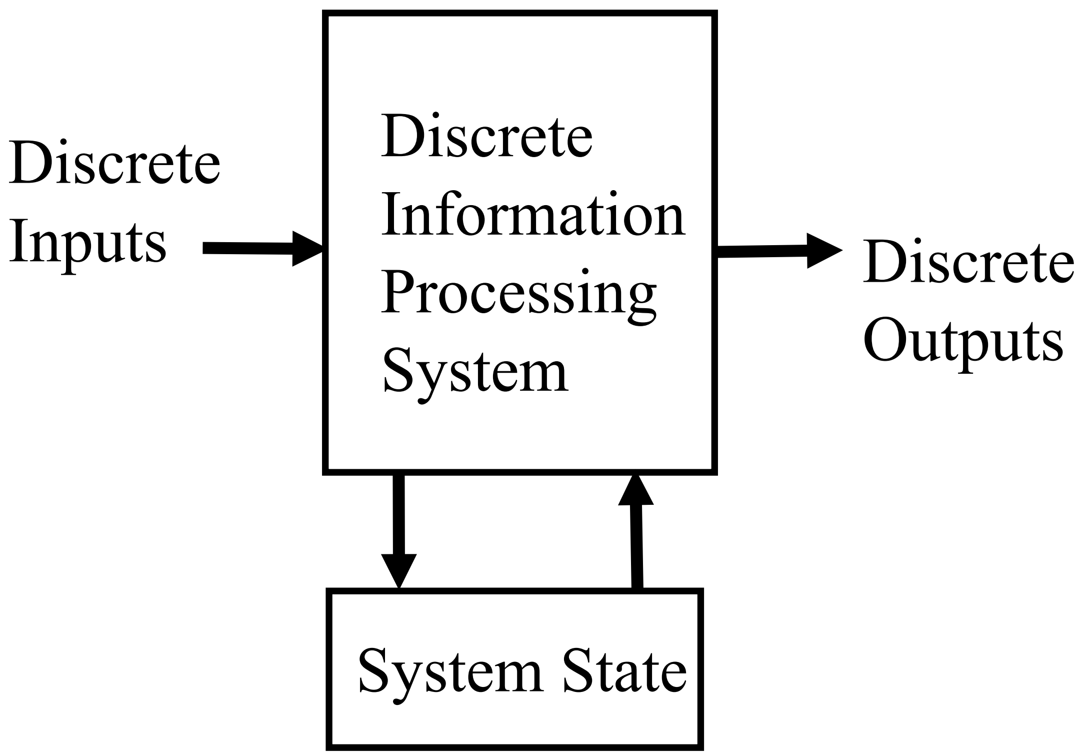
Source: https://electronicsclub.info/

What is digital system?

- Digital systems contain devices such as logic gates, flip-flops, shift registers and counters.
- A **computer** is an example of a digital system.

DIGITAL & COMPUTER SYSTEMS - Digital System

- **Takes a set of discrete information <u>inputs</u> and discrete internal information <u>(system state)</u> and generates a set of discrete information <u>outputs</u>.**

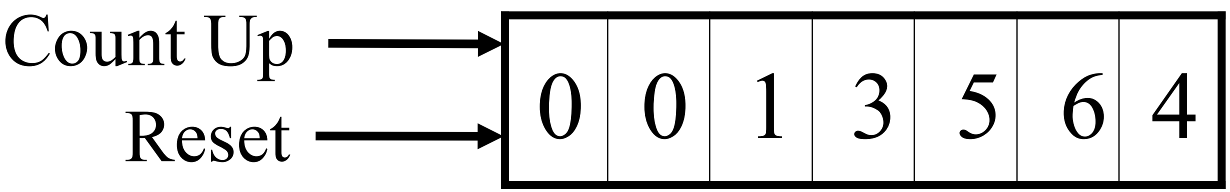


Useful clips

- Digital Computer:
  https://www.youtube.com/watch?v=AdF2uk-EscE
- Essence of Computer (Bits & Bytes, Logic)
  https://www.youtube.com/watch?v=6wU2NoAtWKM
- Inside your computer:
  https://www.youtube.com/watch?v=AkFi90lZmXA

Digital System Example:

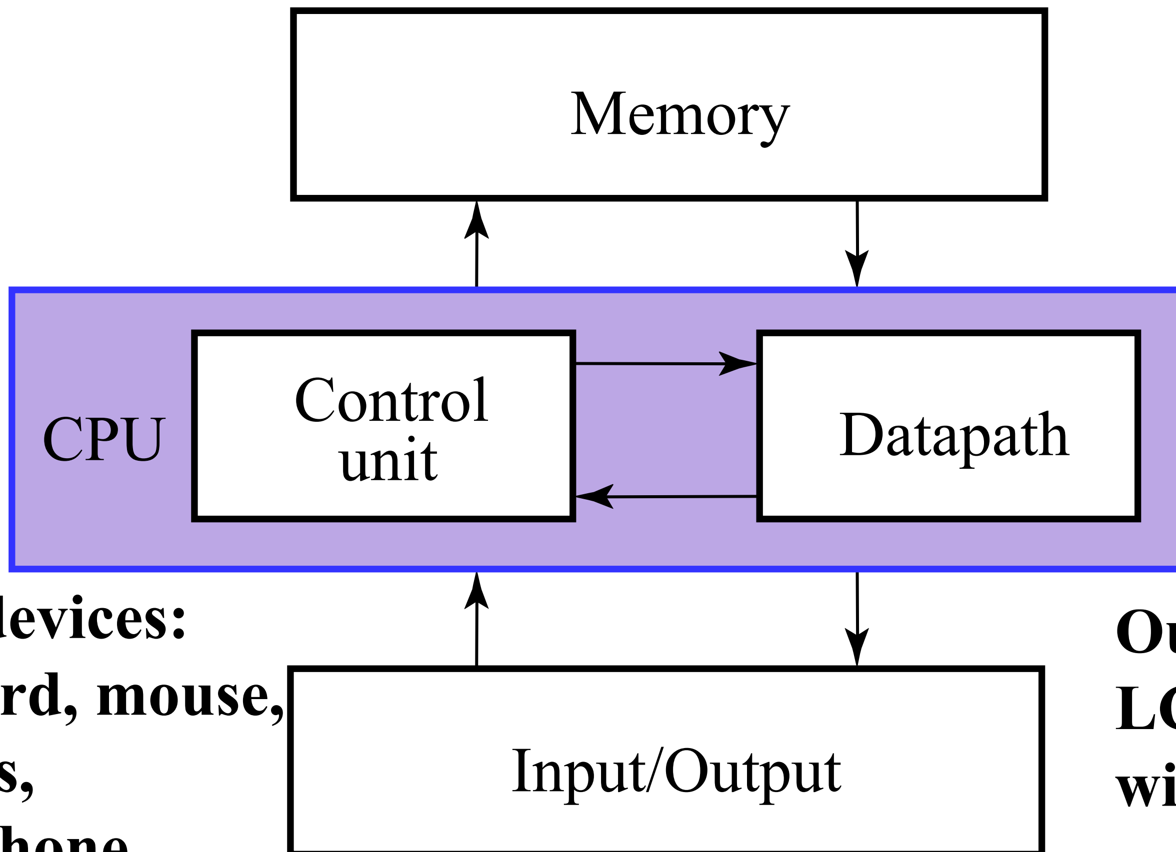
A Digital Counter



Inputs: Count Up, Reset
Outputs: Visual Display
State: "Value" of stored digits

Useful clip: LED => https://www.youtube.com/watch?v=Yo6JI_bzUzo

Digital System in Computer



Input devices:
keyboard, mouse, wireless, microphone

Output devices:
LCD screen, wireless, speakers

And Beyond – Embedded Systems

- Computers as integral parts of other products
- Useful clip:
  - What is Embedded system?

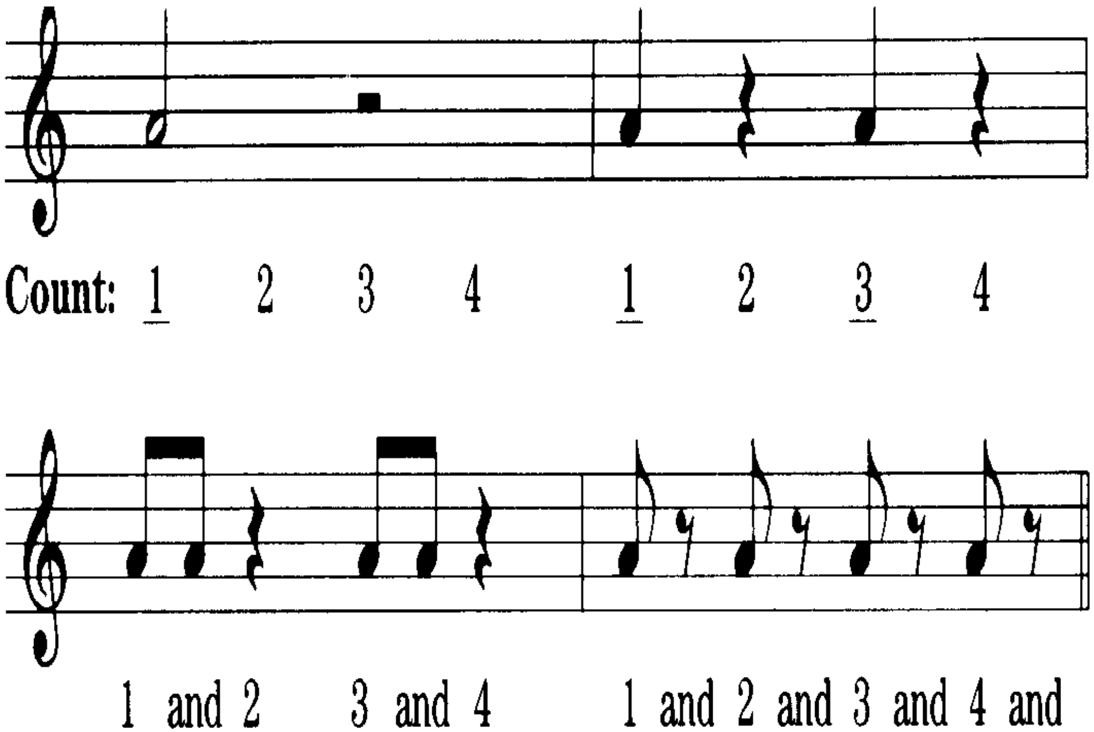
    https://www.youtube.com/watch?v=Qpc1M-BntaM
  - How it works?

    https://www.youtube.com/watch?v=y70V0qHAFNQ

Examples of Embedded Systems

INFORMATION REPRESENTATION - Signals

- Signals in Music Composition



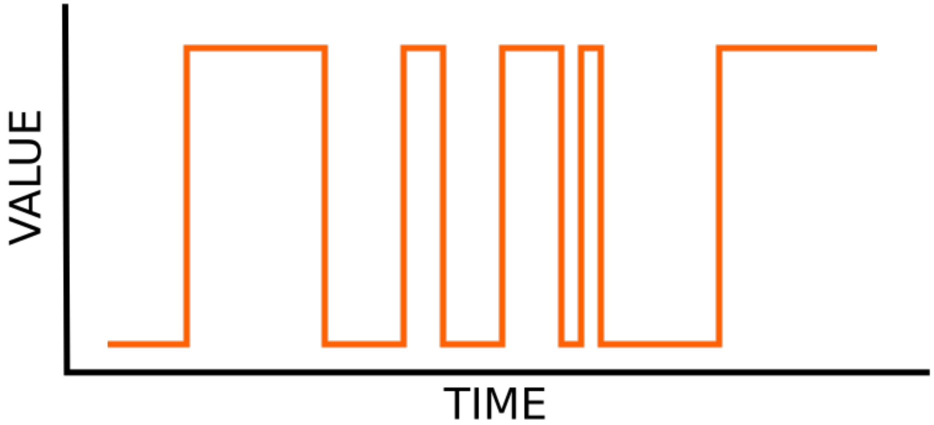
INFORMATION REPRESENTATION - Signals

- Signals in Computer

A **signal** is an electrical or electromagnetic current that is used for carrying data from one device or network to another.

INFORMATION REPRESENTATION - Signals

- Signals in Computer

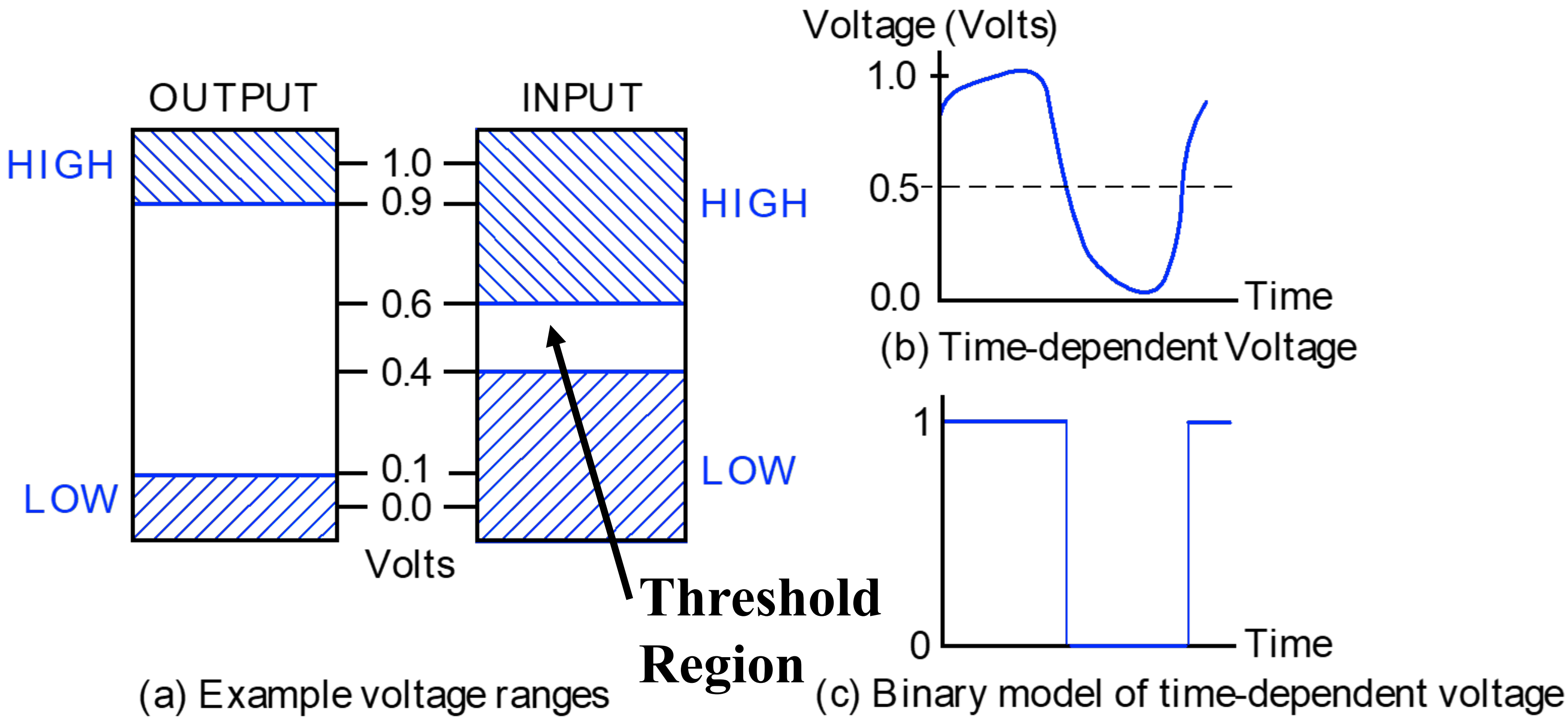


Two main factors: a Value at a Time

INFORMATION REPRESENTATION - Signals

- **Information variables represented by physical quantities.**
- **For digital systems, the variables take on discrete values.**
- **Two levels, or binary values are the most prevalent values in digital systems.**
- **Binary values are represented abstractly by:**
  - **digits 0 and 1**
  - **words (symbols) False (F) and True (T)**
  - **words (symbols) Low (L) and High (H)**
  - **and words On and Off.**
- **Binary values are represented by values or ranges of values of physical quantities**

Signal Example – Physical Quantity: Voltage

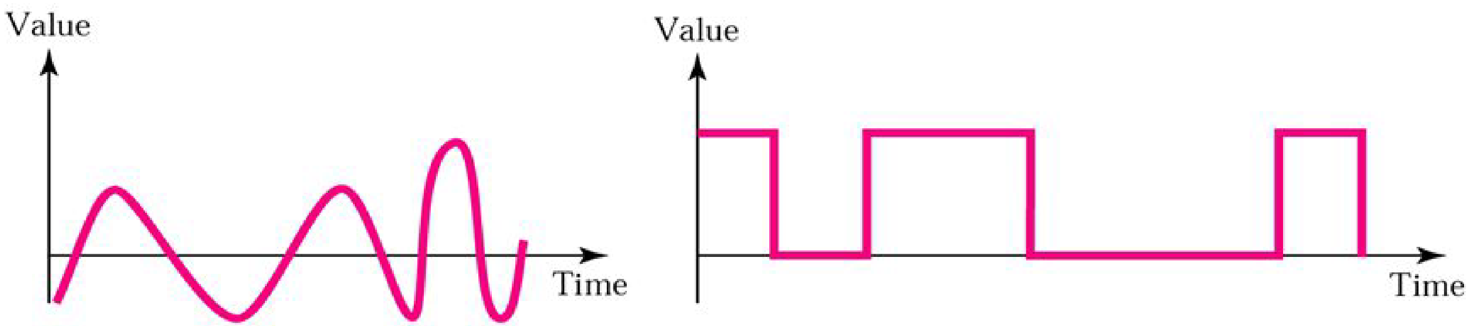


Binary Values: Other Physical Quantities

- **What are other physical quantities represent 0 and 1?**
  - **CPU** **Voltage**
  - **Disk** **Magnetic Field Direction**
  - **CD** **Surface Pits/Light**
  - **Dynamic RAM** **Electrical Charge**

Two Types of Signals

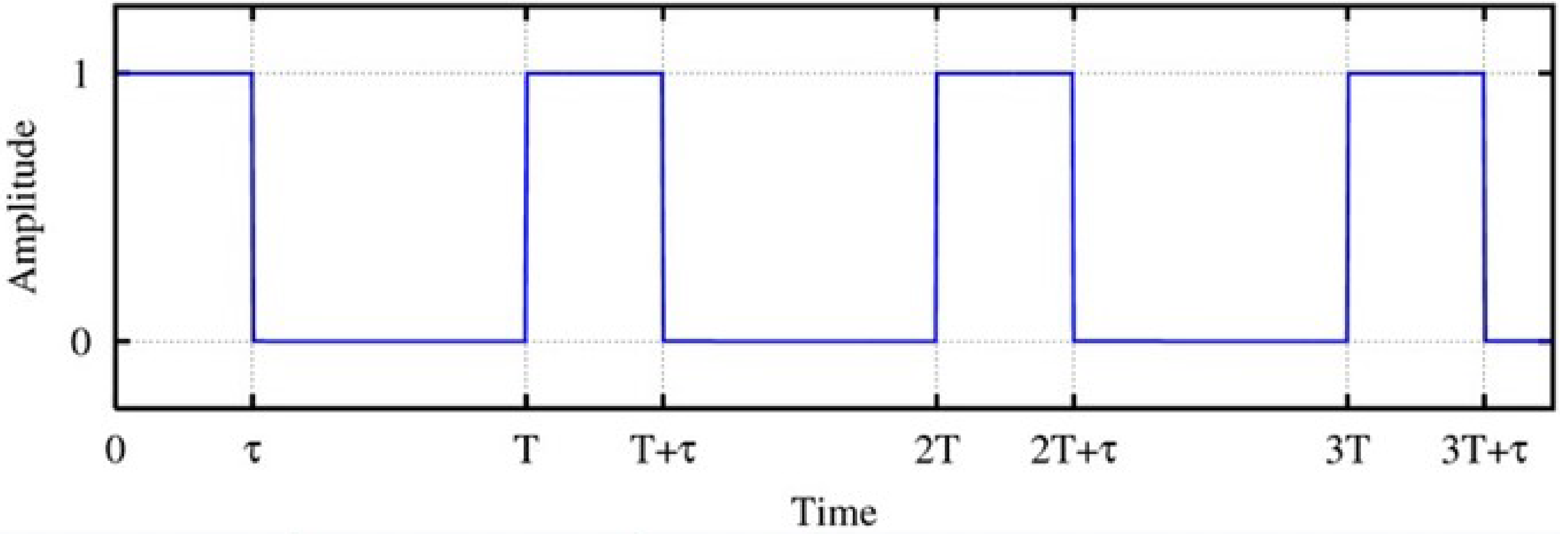
- Analog and Digital Signals
  - An analog is a continuous wave form that changes smoothly over time.
  - A digital signal is discrete. It can have only a limited number of defined values, often as simple as 1 and 0.



a. Analog signal   b. Digital signal

INFORMATION REPRESENTATION - Signals

Digital Signals

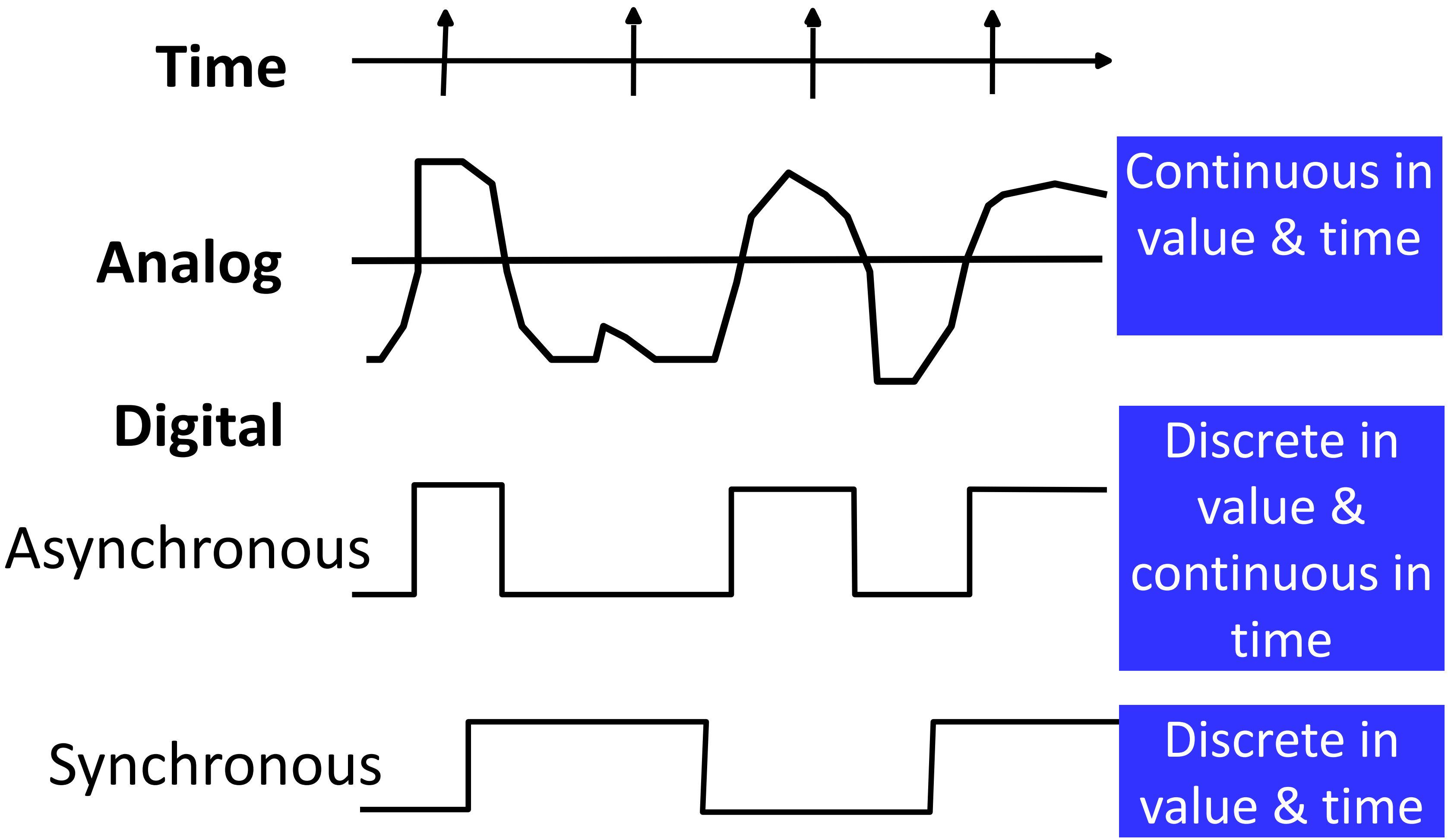


Binary code is a kind of digital signal. It is easy to be handled by digital circuit. So binary code is widely used. Because of "0" and "1" are represented by two kinds of physic status in digital signal.

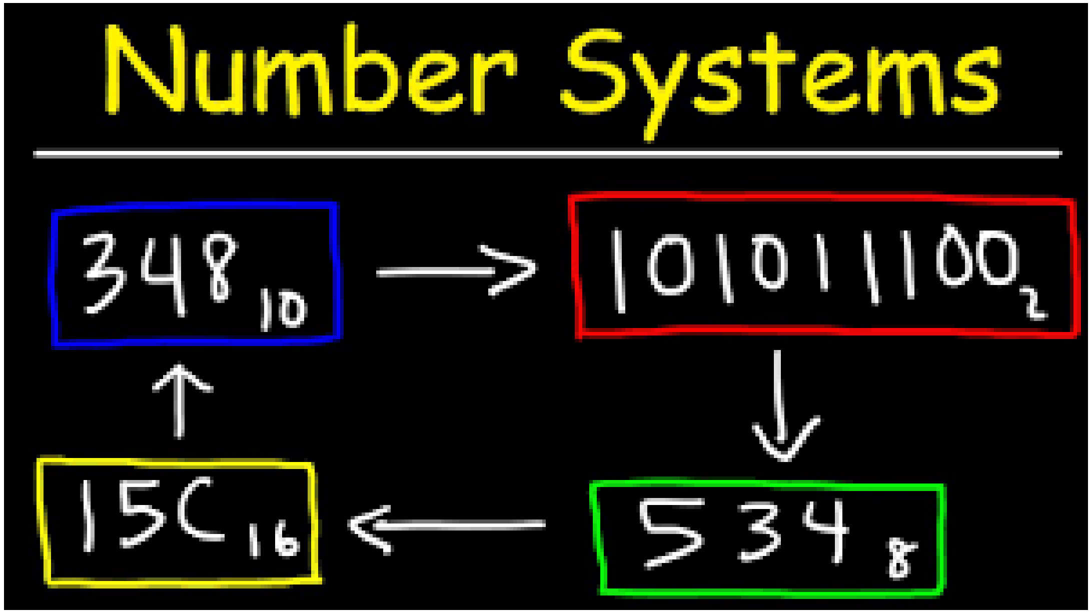
Analog and Digital Signals Transformation

Analog signal and digital signal can realize mutual transform. Analog signal usually use PCM (Pulse Code Modulation) method to quantify and transform to digital signal. PCM method is to make different range of analog signal correspond to different binary value.

Signal Examples Over Time



PART II



Number Systems

- The decimal number system is employed in everyday arithmetic to represent numbers by strings of digits.
- Depending on its position in the string, each digit has an associated value of an integer raised to the power of 10.

NUMBER SYSTEMS – Representation

- Positive radix, positional number systems
- A number with *radix* ***r*** is represented by a string of digits:

$$\underbrace{A_{n-1}A_{n-2}\cdots A_1A_0}_{\text{Integer Portion}}\,.\,\underbrace{A_{-1}A_{-2}\cdots A_{-m+1}A_{-m}}_{\text{Fraction Portion}}$$

in which $\mathbf{0} \leq A_i < r$ and **.** is the *radix point*.

NUMBER SYSTEMS – Representation

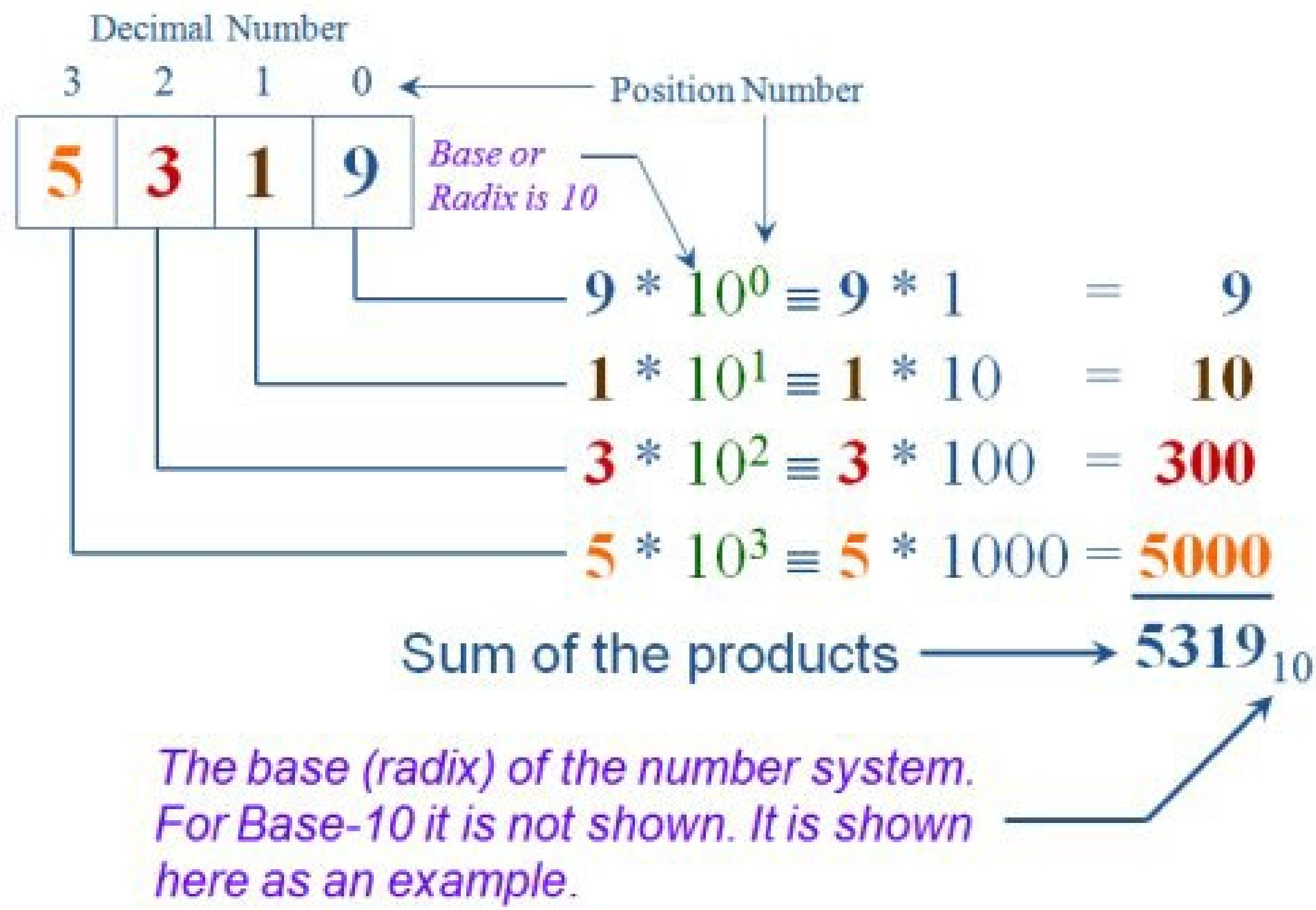
Integer Portion

$$5246 = 5 \times 10^3 + 2 \times 10^2 + 4 \times 10^1 + 6 \times 10^0$$
$$= 5 \times 1000 + 2 \times 100 + 4 \times 10 + 6 \times 1$$

Integer and Fraction Portion

$$254.68 = 2 \times 10^2 + 5 \times 10^1 + 4 \times 10^0 + 6 \times 10^{-1} + 8 \times 10^{-2}$$
$$= 200 + 50 + 4 + \frac{6}{10} + \frac{8}{100}$$

NUMBER SYSTEMS – Representation



Number Systems in other bases

- Four main bases:

Number Systems		
System	Base	Digits
Binary	2	0 1
Octal	8	0 1 2 3 4 5 6 7
Decimal	10	0 1 2 3 4 5 6 7 8 9
Hexadecimal	16	01 2 3 4 5 6 7 8 9 A B C D E F

Representation of Number systems

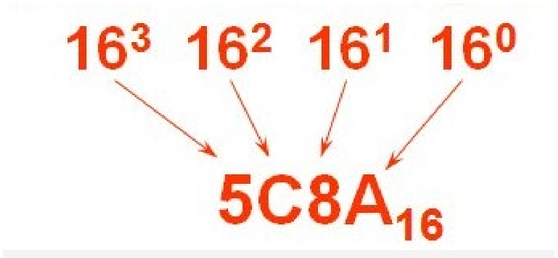
Binary (digits 0-1)

Power of radix	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal value	256	128	64	32	16	8	4	2	1
Binary digit value	1	1	0	1	0	0	1	1	0

Octal (digits 0-7)

Power of radix	8^8	8^7	8^6	8^5	8^4	8^3	8^2	8^1	8^0
Decimal value	16,777,216	2,097,152	262,144	32,768	4,096	512	64	8	1
Octal digit value	6	2	4	6	5	1	2	5	0

Hexadecimal (digits 0-9, A-F)



Power of Two

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,476
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

Number Systems in other bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

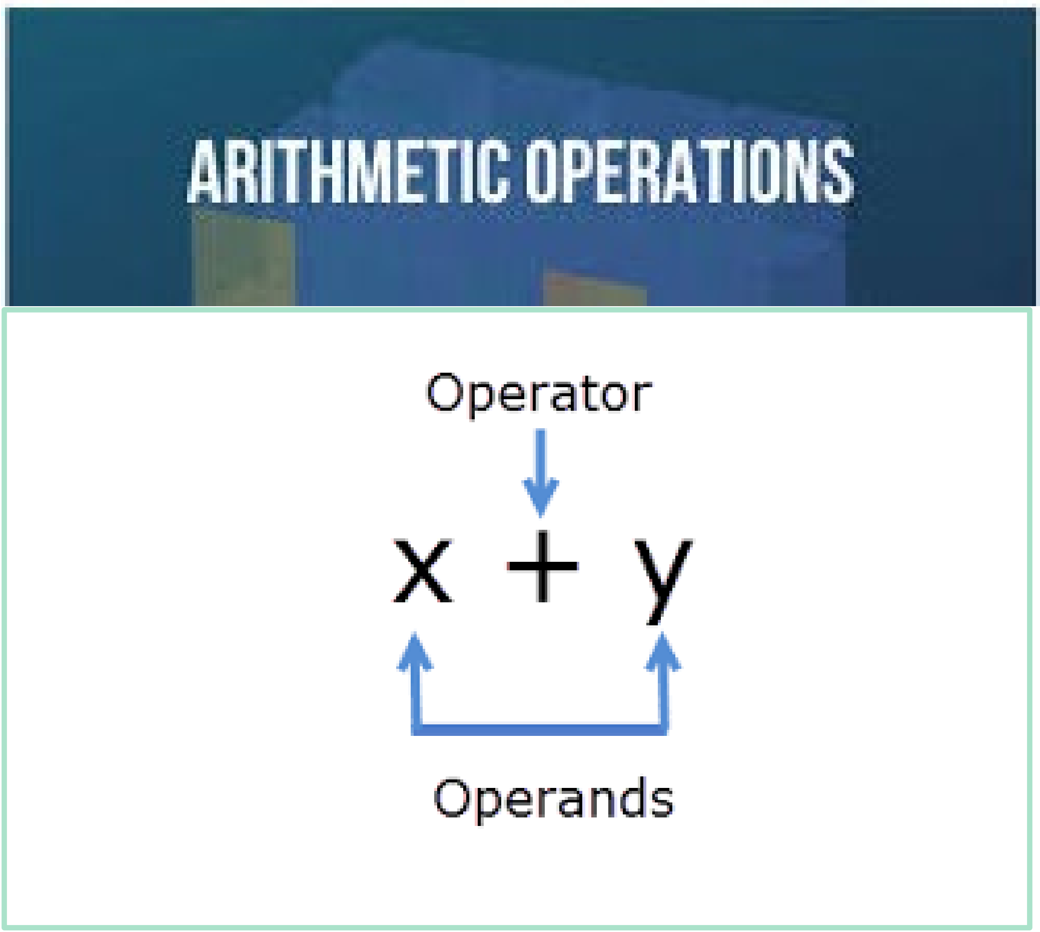
Number Systems – Examples

		General	Decimal	Binary
Radix (Base)		r	10	2
Digits		0 => r - 1	0 => 9	0 => 1
Powers of Radix	0	r^0	1	1
	1	r^1	10	2
	2	r^2	100	4
	3	r^3	1000	8
	4	r^4	10,000	16
	5	r^5	100,000	32
	-1	r^{-1}	0.1	0.5
	-2	r^{-2}	0.01	0.25
	-3	r^{-3}	0.001	0.125
	-4	r^{-4}	0.0001	0.0625
	-5	r^{-5}	0.00001	0.03125

Special Powers of 2

- **2^{10} (1024) is Kilo, denoted "K"**
- **2^{20} (1,048,576) is Mega, denoted "M"**
- **2^{30} (1,073, 741,824)is Giga, denoted "G"**
- **2^{40} (1,099,511,627,776) is Tera, denoted “T"**

PART III.I



ARITHMETIC OPERATIONS - Binary Arithmetic

- **Single Bit Addition with Carry**
- **Multiple Bit Addition**
- **Single Bit Subtraction with Borrow**
- **Multiple Bit Subtraction**
- **Multiplication**
- **Division**

Single Bit Binary Addition with Carry

Given two binary digits (X,Y), a carry in (Z) we get the following sum (S) and carry (C):

Carry in (Z) of 0:

Z	0	0	0	0
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0

Carry in (Z) of 1:

Z	1	1	1	1
X	0	0	1	1
+ Y	+ 0	+ 1	+ 0	+ 1
C S	0 1	1 0	1 0	1 1

Multiple Bit Binary Addition

- Extending this to two multiple bit examples:

Carries	0	0
Augend	01100	10110
Addend	+10001	+10111
Sum	11101	101101

Single Bit Binary Subtraction with Borrow

- **Given two binary digits (X,Y), a borrow in (Z) we get the following difference (S) and borrow (B):**
- **Borrow in (Z) of 0:**

Z	**0**	**0**	**0**
X	**0**	**1**	**1**
- Y	**-0**	**-0**	**-1**
BS	**0 0**	**0 1**	**0 0**

Multiple Bit Binary Subtraction

- **Extending this to two multiple bit examples:**

Borrows	**0**	**0**
Minuend	**10110**	**10110**
Subtrahend	**- 10010**	**- 10011**
Difference	**00100**	**00011**

Binary Multiplication

The binary multiplication table is simple:

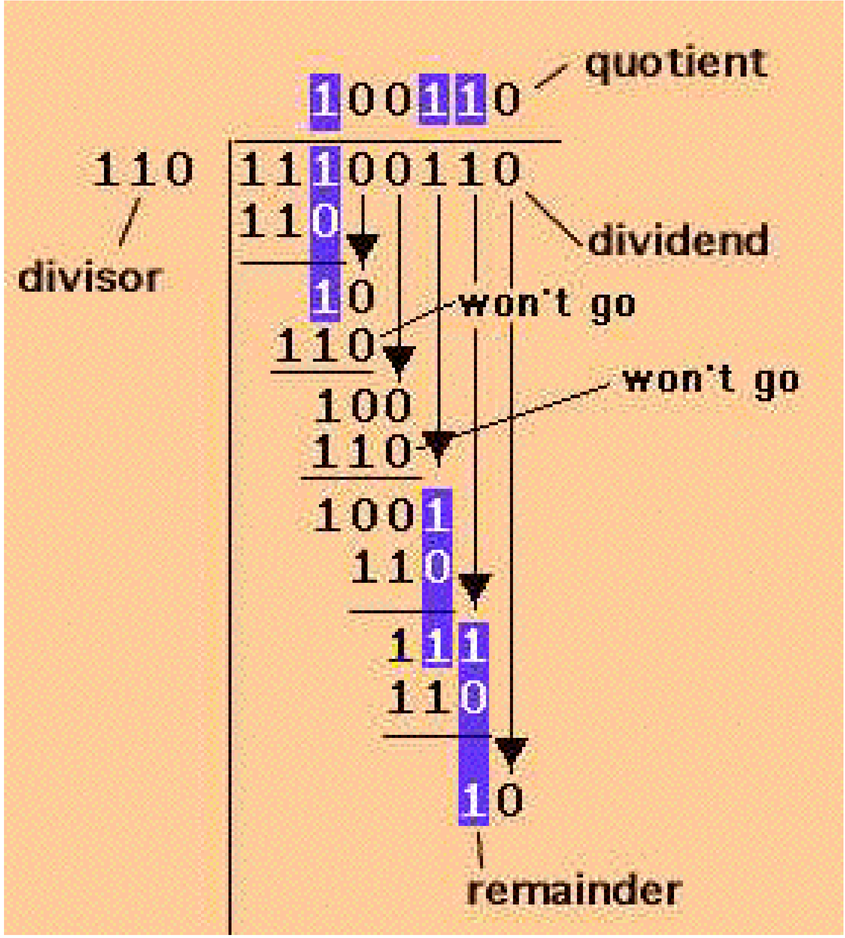
0 * 0 = 0 | 1 * 0 = 0 | 0 * 1 = 0 | 1 * 1 = 1

Extending multiplication to <u>multiple digits</u>:

Multiplicand	**1011**
Multiplier	**<u>x 101</u>**
Partial Products	**1011**
	0000 -
	<u>1011 - -</u>
Product	**110111**

Binary Division

Step 1: First, look at the first three numbers in the dividend and compare with the divisor.
Step 2: Add the number 1 in the quotient place. Then subtract the value, you get 1 as remainder.
Step 3: Repeat the process until the remainder becomes zero by comparing the dividend and the divisor value.



Commonly Occurring Bases

Name	Radix	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

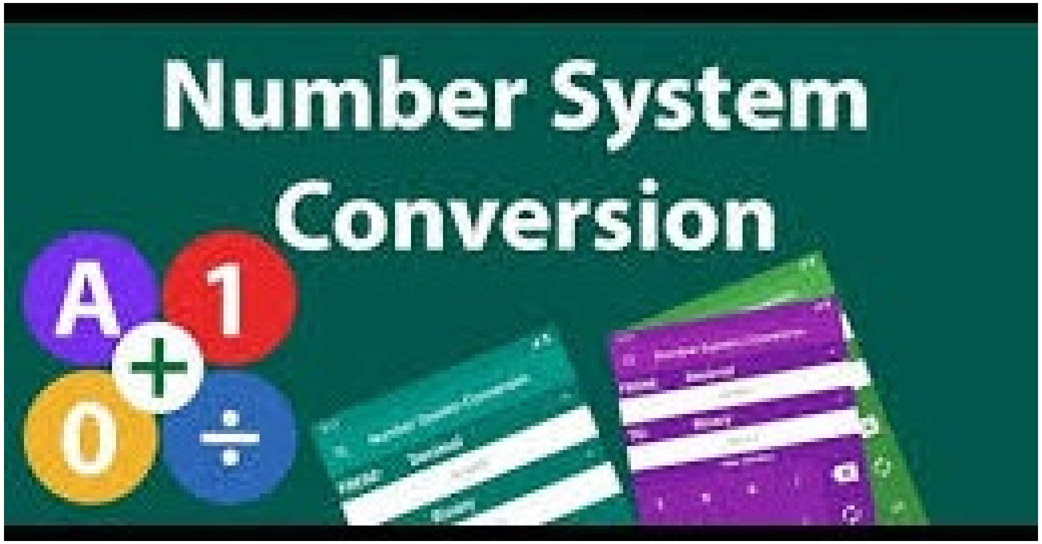
- The six letters (in addition to the 10 integers) in hexadecimal represent: A=10, B=11, C=12, D=13, E=14, F=15

Numbers in Different Bases

- **Good idea to memorize!**

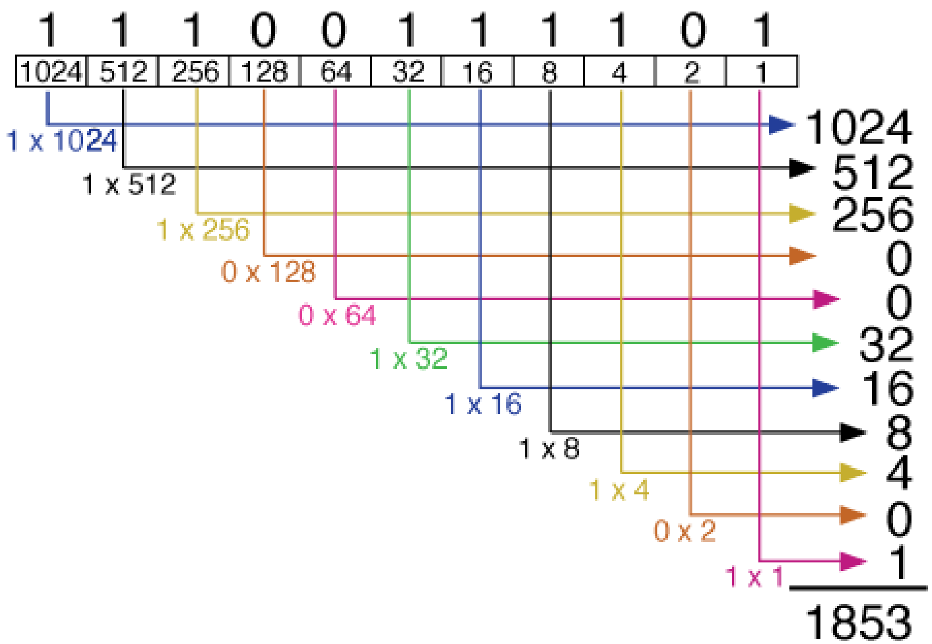
Decimal (Base 10)	Binary (Base 2)	Octal (Base 8)	Hexadecimal (Base 16)
00	00000	00	00
01	00001	01	01
02	00010	02	02
03	00011	03	03
04	00100	04	04
05	00101	05	05
06	00110	06	06
07	00111	07	07
08	01000	10	08
09	01001	11	09
10	01010	12	0A
11	01011	13	0B
12	01100	14	0C
13	01101	15	0D
14	01110	16	0E
15	01111	17	0F
16	10000	20	10

PART III.II



Converting Binary to Decimal

- **Converting Binary to Decimal, We use decimal arithmetic to form Σ (digit × respective power of 2).**
- **Example: Convert 11100111101_2 to N_{10}:**



Converting Binary to Decimal

Convert 110100110_2 to N_{10}

Fill in the Binary Pattern (digits 0-1)

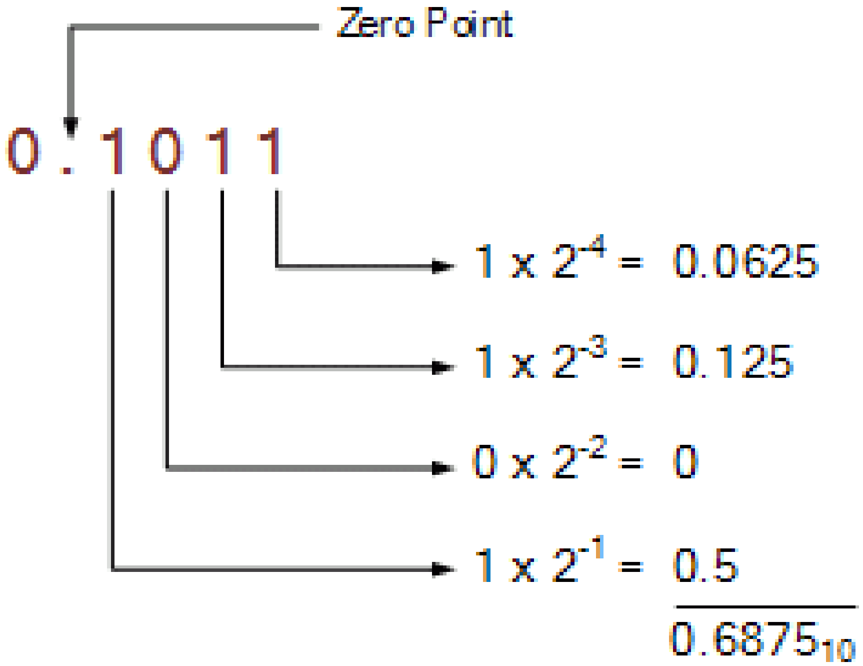
Power of radix	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal value	256	128	64	32	16	8	4	2	1
Binary digit value	1	1	0	1	0	0	1	1	0

$$= 256+128+32+4+2$$

$$= 422$$

Converting Binary to Decimal

- **Fractional portion**
- **Example: Convert .1011 to N_{10}:**

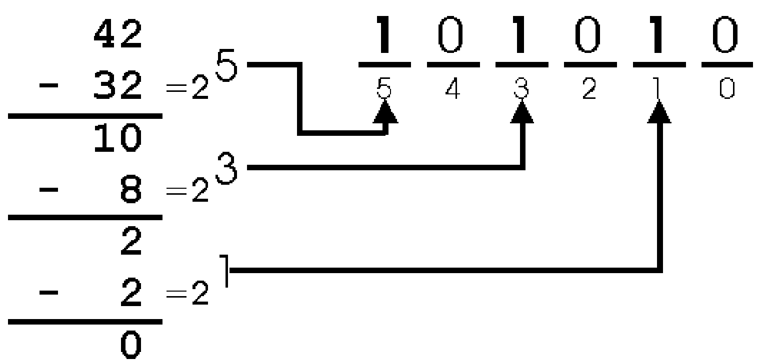


Converting Decimal to Binary

- There are 2 methods:
  - Subtracting method
  - Dividing method
- Dividing method
  - Integral part
  - Fractional part

Converting Decimal to Binary

- **Method 1: Subtracting**
  - **Subtract the largest power of 2 that gives a positive remainder and record the power.**
  - **Repeat, subtracting from the prior remainder and recording the power, until the remainder is zero.**
  - **Place 1's in the positions in the binary result corresponding to the powers recorded; in all other positions place 0's.**
- **Example: Convert 42_{10} to N_2**



Method 2: Dividing method

- **To Convert the Integral Part:**

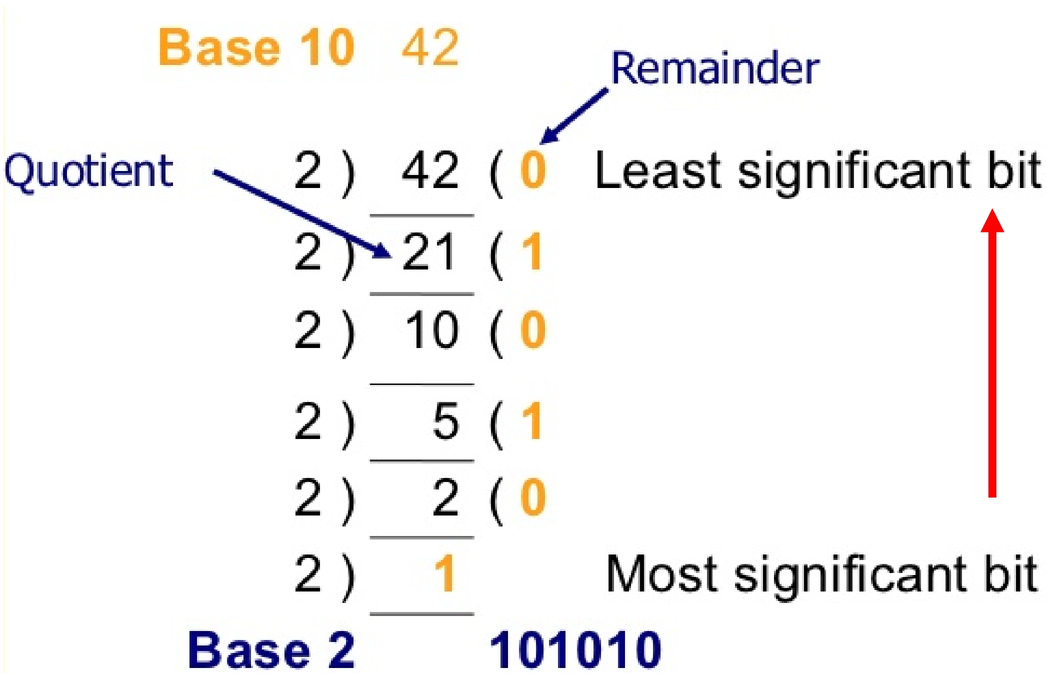
  Repeatedly divide the number by the new radix and save the remainders. The digits for the new radix are the remainders in *reverse order* of their computation. If the new radix is > 10, then convert all remainders > 10 to digits A, B, …

- **To Convert the Fractional Part:**

  Repeatedly multiply the fraction by the new radix and save the integer digits that result. The digits for the new radix are the integer digits in *order* of their computation. If the new radix is > 10, then convert all integers > 10 to digits A, B, …

Converting Decimal to Binary

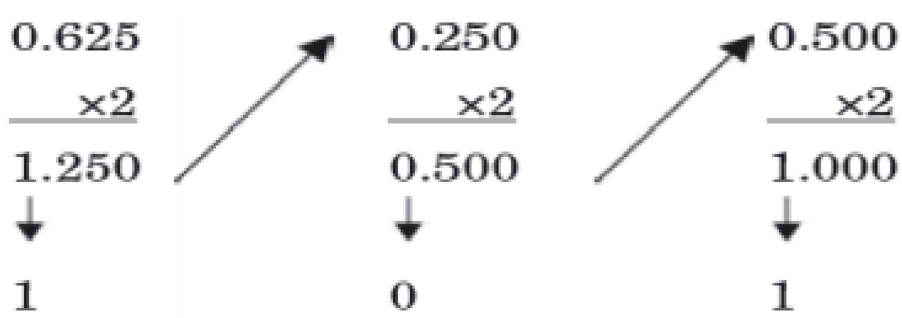
- **Method 2 (Integral Part)**
  - **Divide the decimal number by 2, keep remaining digit as the final results.**
  - **Repeat, dividing the remaining results until its value is zero or one.**
  - **Example, convert 42 to N_2**



Read the Binary results from bottom up

Converting Decimal to Binary

- **Method 2 (Fractional Part)**
  - **Multiply the decimal number by 2, keep remaining digit as the final results.**
  - **Repeat, multiplying the floating value with 2 until its fractional value is zero.**
  - **Example, convert 0.625 to N_2**



Read the Binary results from left to right

Final result is 0.101

Additional Issue - Fractional Part

- **Note that in this conversion, the fractional part can become 0 as a result of the repeated multiplications.**
- **In general, it may take many bits to get this to happen or it may never happen.**
- **Example Problem: Convert 0.65_{10} to N_2**
  - **0.65 = 0.1010011001001 …**
  - **The fractional part begins repeating every 4 steps yielding repeating 1001 forever!**
- **Solution: Specify number of bits to right of radix point and round or truncate to this number.**

Checking the Conversion

- **To convert back, sum the digits times their respective powers of r.**
- **From the prior conversion of 46.6875_{10}**

$$101110_2 = 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1$$
$$= 32 + 8 + 4 + 2$$
$$= 46$$

$$0.1011_2 = 1/2 + 1/8 + 1/16$$
$$= 0.5000 + 0.1250 + 0.0625$$
$$= 0.6875$$

Octal (Hexadecimal) to Binary and Back

- **Octal (Hexadecimal) to Binary:**
  - **Restate the octal (hexadecimal) as three (four) binary digits starting at the radix point and going both ways.**
- **Binary to Octal (Hexadecimal):**
  - **Group the binary digits into three (four) bit groups starting at the radix point and going both ways, padding with zeros as needed in the fractional part.**
  - **Convert each group of three bits to an octal (hexadecimal) digit.**

Hexadecimal to Octal via Binary

Useful Trick

Step 1: Convert hexadecimal (or octal) to binary.

- **Use groups of four bits (hexa) or three bits (octal) to form binary.**

$5F_{(16)}$ → $N_{(2)}$

Hexadecimal	5 F

Apply 4 Bits pattern: 8 4 2 1

	5				F			
	8	4	2	1	8	4	2	1
Binary	0	1	0	1	1	1	1	1

Hexadecimal to Octal via Binary

Useful Trick

Step 2: Convert binary to octal (or hexadecimal).

- **Use groups of four bits (hexa) or three bits (octal) of binary to convert to another base.**

$01011111_{(2)}$

Binary	0101 1111

Apply 3 Bits pattern

4	**2**	**1**

	4	**2**	**1**	**4**	**2**	**1**	**4**	**2**	**1**
Binary	0	0	1	0	1	1	1	1	1
		1			**3**			**7**	

$N_{(8)}$

Binary Numbers and Binary Coding

- **Flexibility of representation**
  - **Within constraints below, can assign any binary combination (called a code word) to any data as long as data is uniquely encoded.**
- **Information Types**
  - **Numeric**
    - **Must represent range of data needed**
    - **Very desirable to represent data such that simple, straightforward computation for common arithmetic operations permitted**
    - **Tight relation to binary numbers**
  - **Non-numeric**
    - **Greater flexibility since arithmetic operations not applied.**
    - **Not tied to binary numbers**

Non-numeric Binary Codes

- **Given *n* binary digits (called <u>bits</u>), a <u>binary code</u> is a mapping from a set of <u>represented elements</u> to a subset of the 2^n binary numbers.**
- **Example: A binary code for the seven colors of the rainbow**
- **Code 100 is not used**

Color	Binary Number
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111

DECIMAL CODES - Binary Codes for Decimal Digits

- There are over 8,000 ways that you can choose 10 elements from the 16 binary numbers of 4 bits. A few are useful:

Decimal	8,4,2,1	Excess3	8,4,-2,-1
0	0000	0011	0000
1	0001	0100	0111
2	0010	0101	0110
3	0011	0110	0101
4	0100	0111	0100
5	0101	1000	1011
6	0110	1001	1010
7	0111	1010	1001
8	1000	1011	1000
9	1001	1100	1111

Binary Coded Decimal (BCD)

- **The BCD code is the 8,4,2,1 code.**
- **8, 4, 2, and 1 are weights**
- **BCD is a *weighted* code**
- **This code is the simplest, most intuitive binary code for decimal digits and uses the same powers of 2 as a binary number, but only encodes the first ten values from 0 to 9.**
- **Example:** $11_{10} = 1011_2 = 00010001_{BCD}$

Warning: Conversion or Coding?

- **Do NOT mix up conversion of a decimal number to a binary number with coding a decimal number with a BINARY CODE.**
- **$13_{10} = 1101_2$ (This is conversion)**
- **13 $\Leftrightarrow$ 0001|0011 (This is coding)**

BCD Arithmetic

- Given a BCD code, we use binary arithmetic to add the digits:

8	1000	Eight
+5	+0101	Plus 5
13	1101	is 13 (> 9)

- **Note that the result is MORE THAN 9, so must be represented by two digits!**
- To correct the digit, subtract 10 by adding 6 modulo 16.

8	1000	Eight
+5	+0101	Plus 5
13	1101	is 13 (> 9)
	+0110	so add 6
	carry = 1 0011	leaving 3 + cy
	0001 | 0011	Final answer (two digits)

- If the digit sum is > 9, add one to the next significant digit

BCD Addition Example

- **Add 2905_{BCD} to 1897_{BCD} showing carries and digit corrections.**

	1	1	1	0
	0001	1000	1001	0111
+	0010	1001	0000	0101
	0100	10010	1010	1100
		+ 0110	+ 0110	+ 0110
	0100	1000	0000	0010
=	4	8	0	$2_{(bcd)}$

Error-Detection Codes using PARITY BIT

- **Redundancy** (e.g. extra information), in the form of extra bits, can be incorporated into binary code words to detect and correct errors.
- A simple form of redundancy is **parity**, an extra bit appended onto the code word to make the number of 1's odd or even. Parity can detect all single-bit errors and some multiple-bit errors.
- A code word has **even parity** if the number of 1's in the code word is even.
- A code word has **odd parity** if the number of 1's in the code word is odd.

4-Bit Parity

- **Fill in the even and odd parity bits:**

Even Parity Message - Parity	Odd Parity Message Parity
000 0	000 1
001 1	001 0
010 1	010 0
011 0	011 1
100 1	100 0
101 0	101 1
110 0	110 1
111 1	111 0

- **The codeword "1111" has <u>even parity</u> and the codeword "1110" has <u>odd parity</u>. Both can be used to represent 3-bit data.**

ALPHANUMERIC CODES - ASCII Character Codes

- American Standard Code for Information Interchange
- This code is a popular code used to represent information sent as character-based data. It uses 7-bits to represent:
  - 94 Graphic printing characters.
  - 34 Non-printing characters
- Some non-printing characters are used for text format (e.g. BS = Backspace, CR = carriage return)
- Other non-printing characters are used for record marking and flow control (e.g. STX and ETX start and end text areas).

ASCII Characters Codes

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

UNICODE

- **UNICODE extends ASCII to 65,536 universal characters codes**
  - **For encoding characters in world languages**
  - **Available in many modern applications**
  - **2 byte (16-bit) code words**