CSE 4205
Digital Logic Design

Binary System

Course Teacher: Md. Hamjajul Ashmafee

Lecturer, CSE, IUT

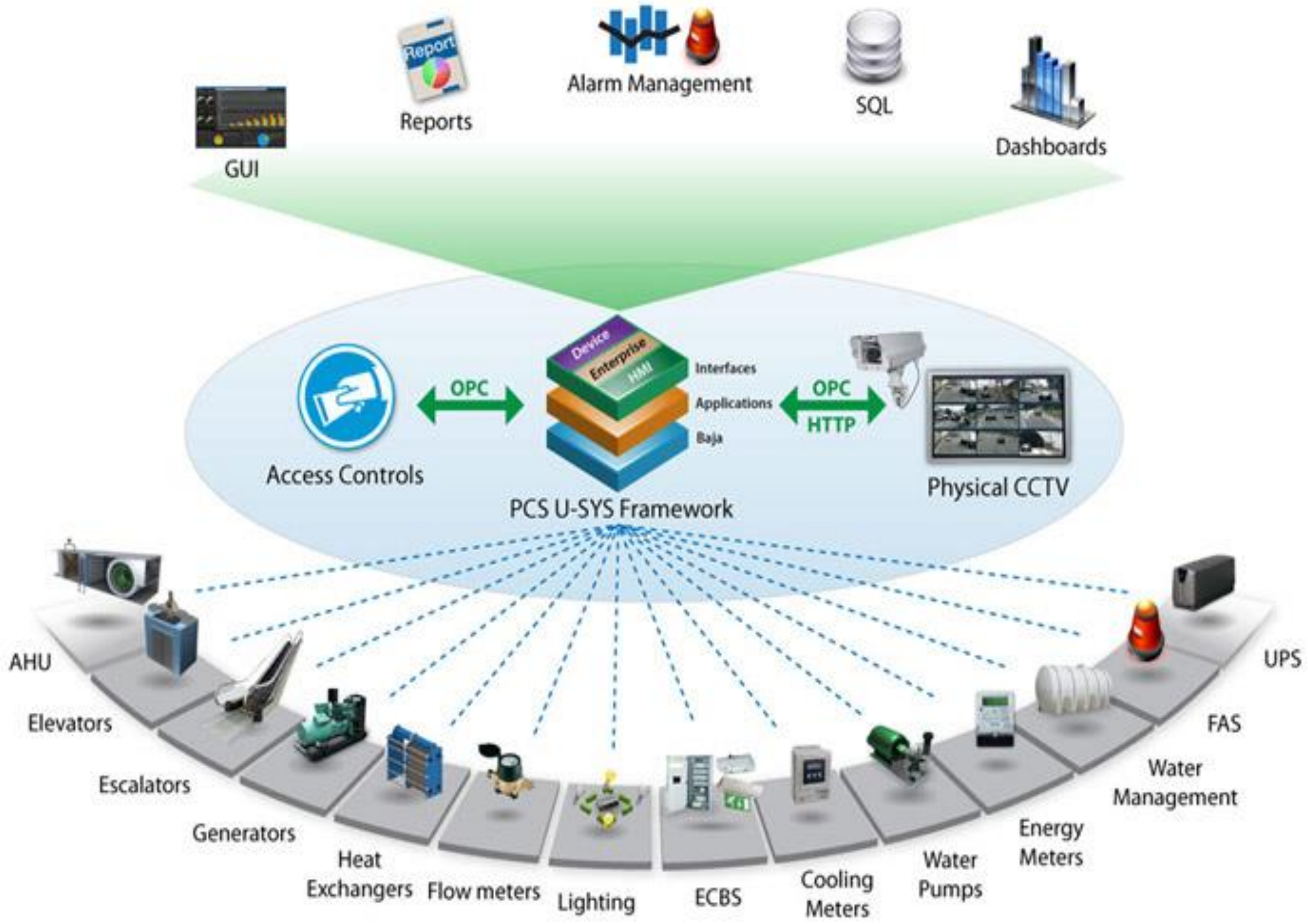
Email: ashmafee@iut-dhaka.edu

Digital Computer and Digital System

- Many scientific, industrial and commercial advances have been occurred through ***Digital Computer***.
- Examples:
  - Scientific Calculation
  - Commercial and business data processing
  - Air traffic control
  - Space guidance
  - Educational field
  - And many more…
- Generality of digital computer - involves it everywhere.

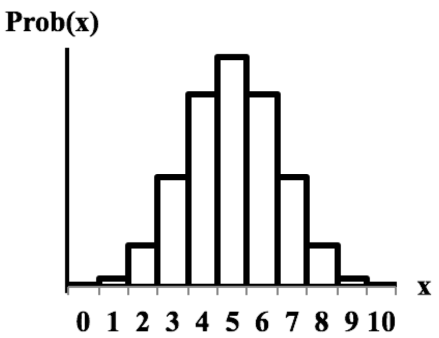
Digital Computer and Digital System …

- Generality of digital computer
  - ✓It follows a sequence of instructions (**program**)
  - ✓It operates on data (**user given data**)
  - ✓Based on those it generates user specified results
- For this reason, computer is now everywhere.

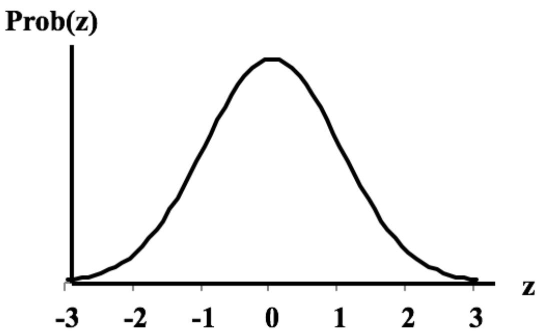


Digital Computer and Digital System …

- Digital system – process discrete element of information (not continuous)
- Early computers – used only for numerical computation and discrete elements were digits (*Digital computer*)
- Appropriate name for a digital computer – **Discrete Information Processing System**



Binomial Distribution
Discrete Data & Discrete Probability Curve



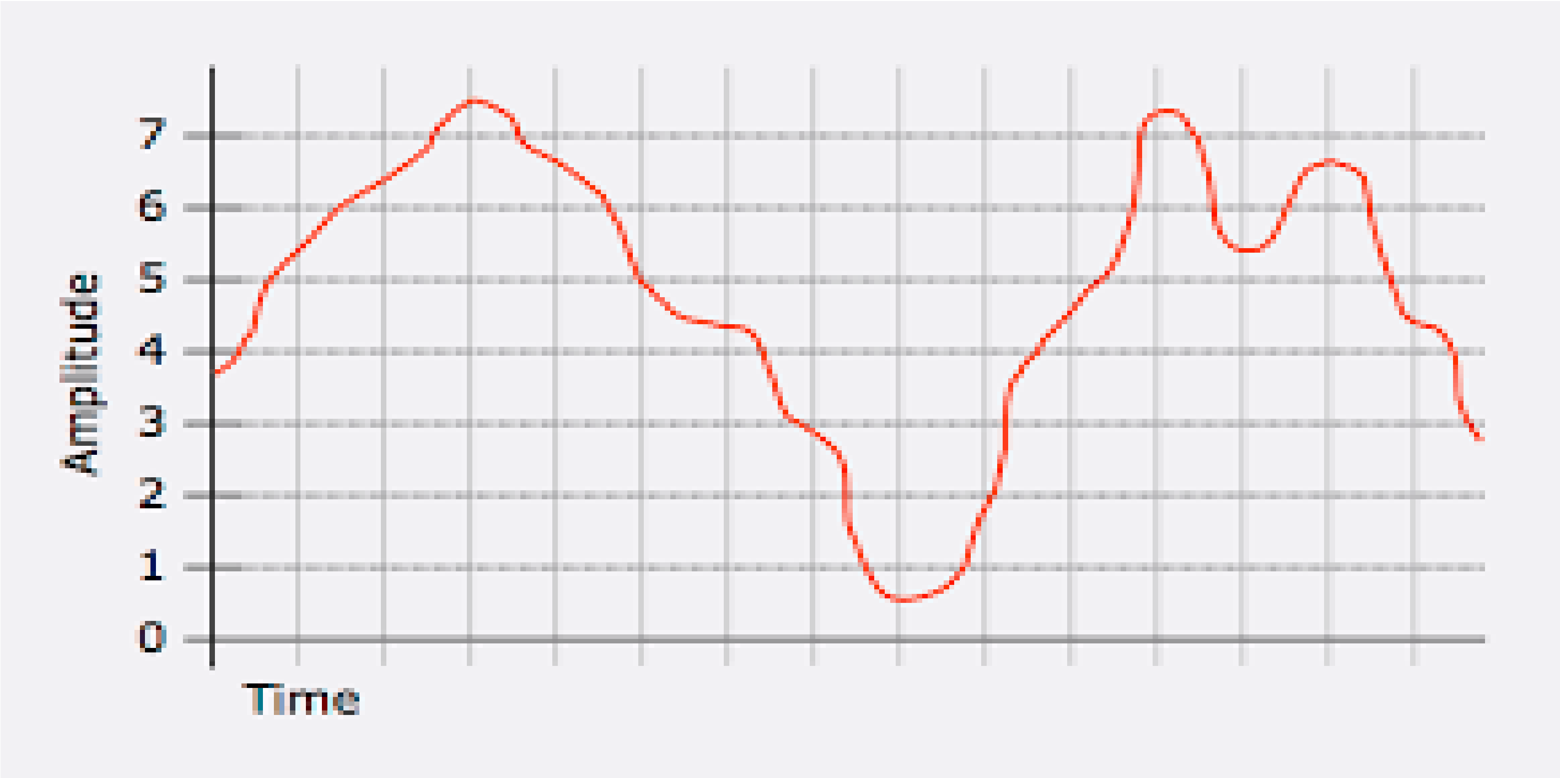
Standard Normal Distribution
Continuous Data and Continuous Probability Curve

Digital Computer and Digital System …

- Discrete element of **information** – signal
- Electric signal – voltage and current are most common
- Signal in all electronic digital system – two discrete values (Binary)
- Any system uses an alphabet to represent information
  - English language – an alphabet of 26 letters
  - Decimal number system – an alphabet of 10 digits
- Digital system uses an alphabet with two digits(bits) - (**0 and 1**)
- Example – switch (made of transistor) – on and off
- Example - Voltage values – high (1) and low (0)

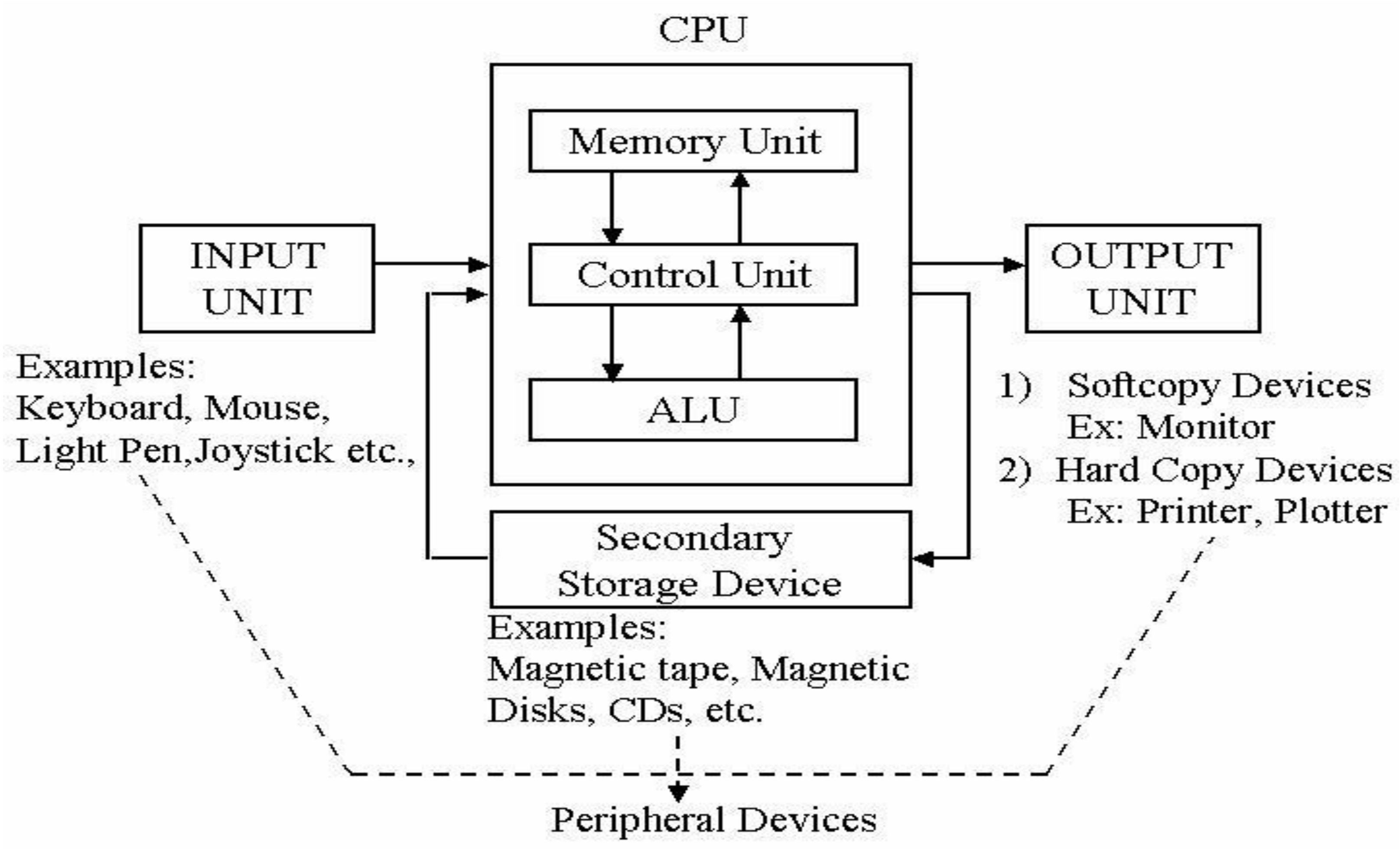
Digital Computer and Digital System …

- Analog to digital data conversion:



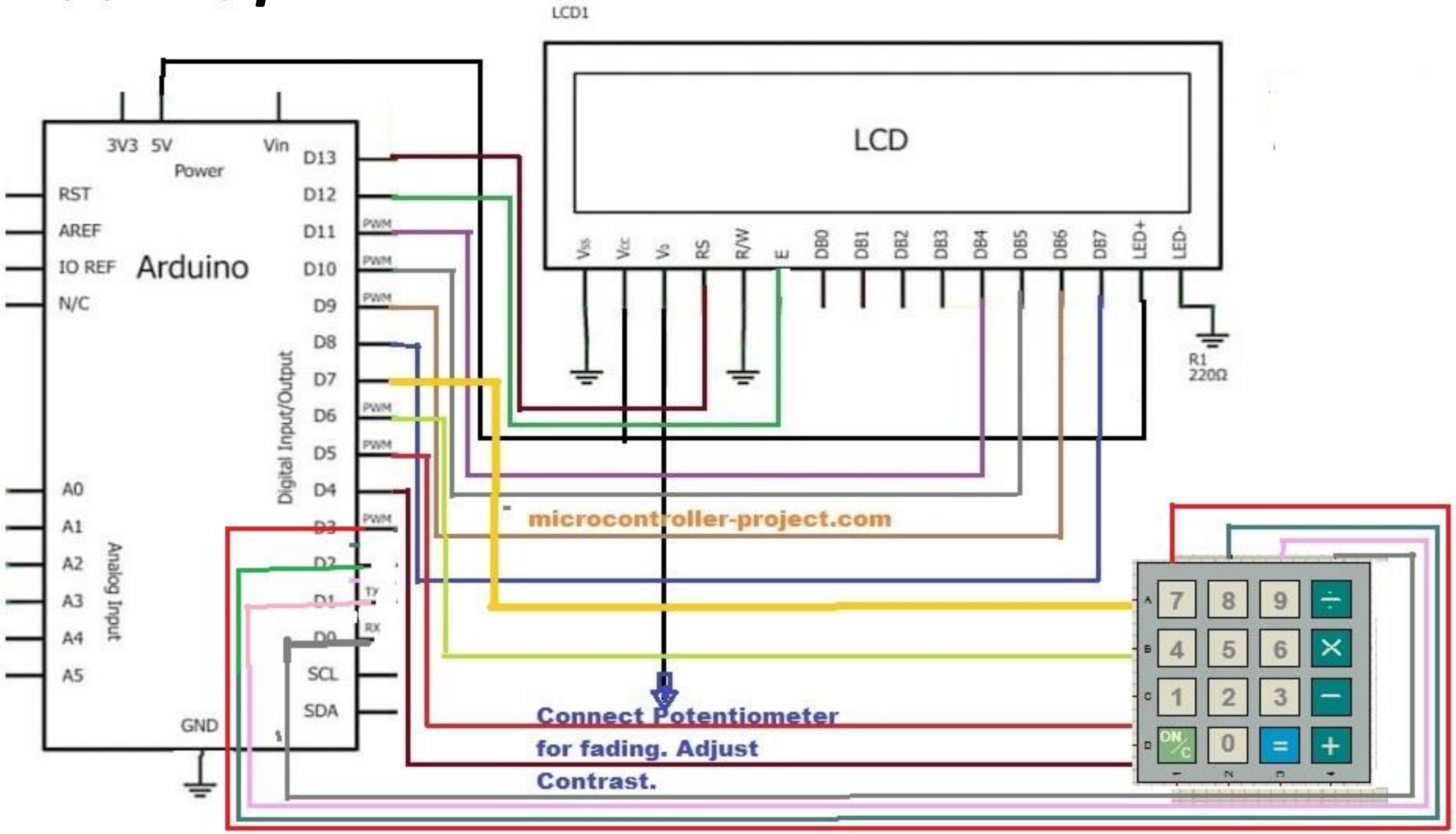
Digital Computer and Digital System …

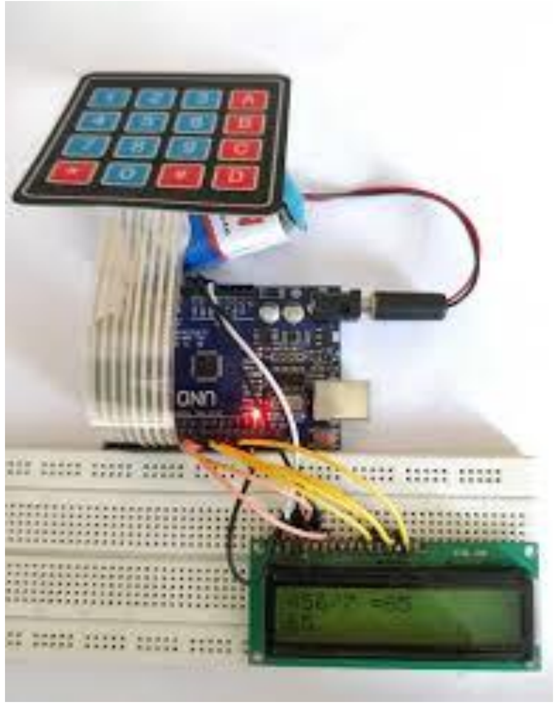
Block diagram of a digital computer:



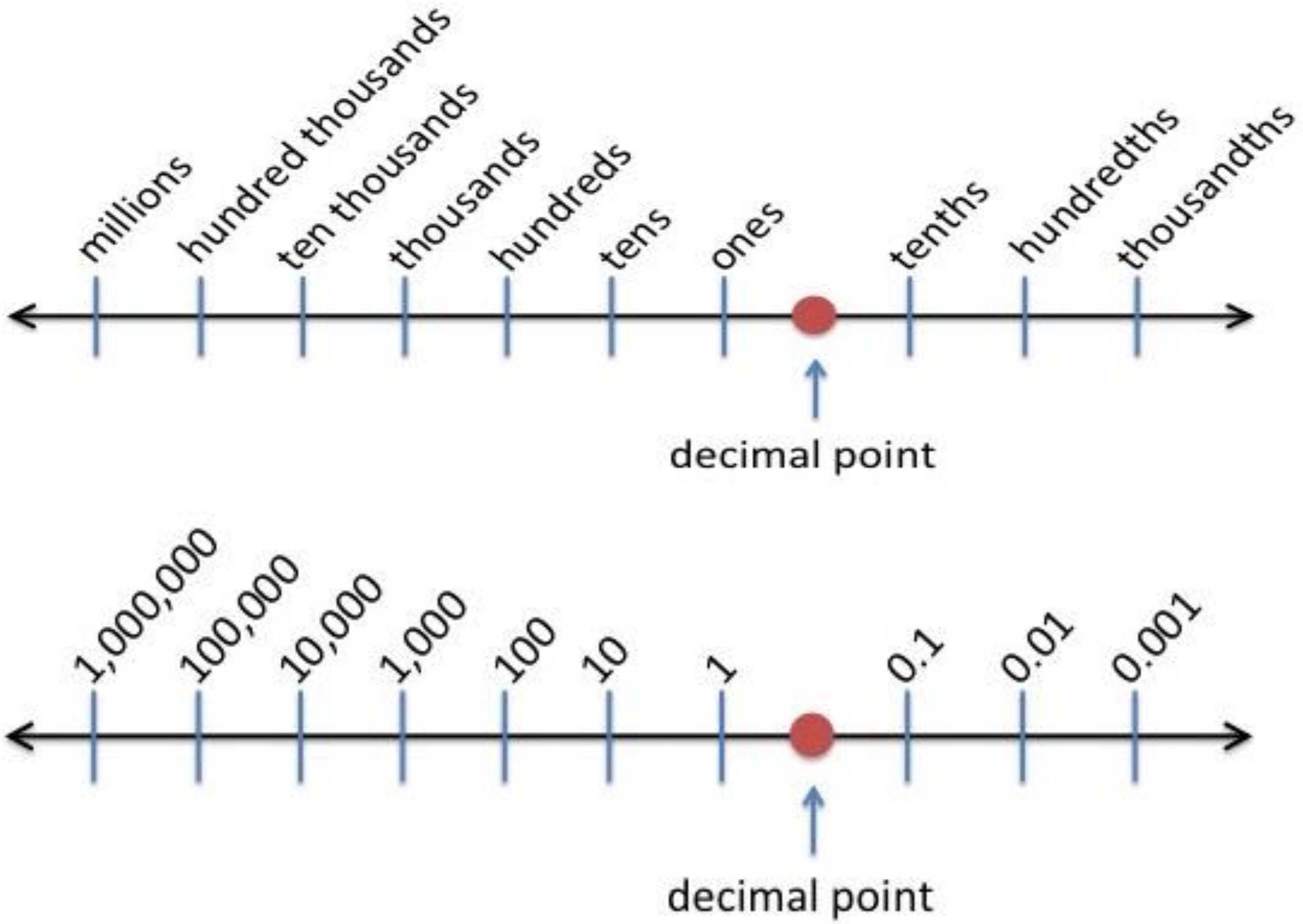
Digital Computer and Digital System …

Example: An Electronic Calculator





Decimal Numbers



Decimal Numbers

- Example:

Write 12,357 in expanded form.

12,357

We can leave our answer as it is or simplify some of the exponents. Any of the answers below are acceptable.

$$(1 \times 10^4)+(2 \times 10^3)+(3 \times 10^2)+(5 \times 10^1)+(7 \times 10^0)$$

$$(1 \times 10^4)+(2 \times 10^3)+(3 \times 10^2)+(5 \times 10^1)+ (7 \times 1)$$

$$(1 \times 10{,}000)+(2 \times 1{,}000)+(3 \times 100)+(5 \times 10)+(7 \times 1)$$

Number System

- Any number with a decimal point (**positional number system**) :

$$\pm (S_{k-1} \ldots S_2\, S_1\, S_0 .\, S_{-1}\, S_{-2} \ldots S_{-l})_b$$

$$n = \pm \; S_{k-1} \times b^{k-1} + \ldots + S_1 \times b^1 + S_0 \times b^0 \; + \; S_{-1} \times b^{-1} + S_{-2} \times b^{-2} + \ldots + S_{-l} \times b^{-l}$$

- $n = \pm \sum_{i=-l}^{k-1} S_i . b^i$
- Where **S** is the set symbols and **b** is the base or radix.
- **0 ≤ S < b**
- Example: Decimal, Binary, Octal

Number System

- Arithmetic Operation
  - **Addition** (augend, addend, sum)
  - **Subtraction** (minuend, subtrahend, difference)
  - **Multiplication** (multiplicand, multiplier, product)
  - **Division** (dividend, divisor, quotient, reminder)
  - **Negation**
- **Carry** and **borrow** in base b.

Number System

- Number with different bases:

Decimal	Binary	Octal	Hexadecimal
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	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

Special Powers of 2

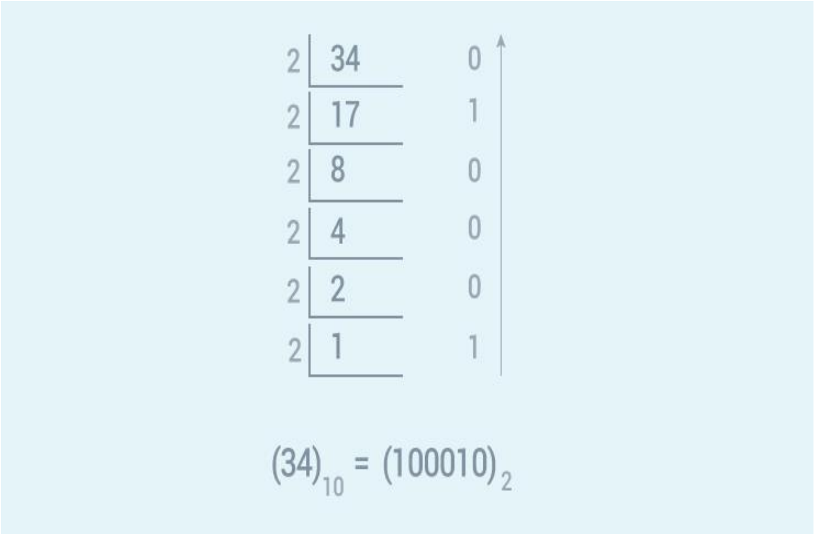
2^{10} (1024) is Kilo, denoted "K"

2^{20} (1,048,576) is Mega, denoted "M"

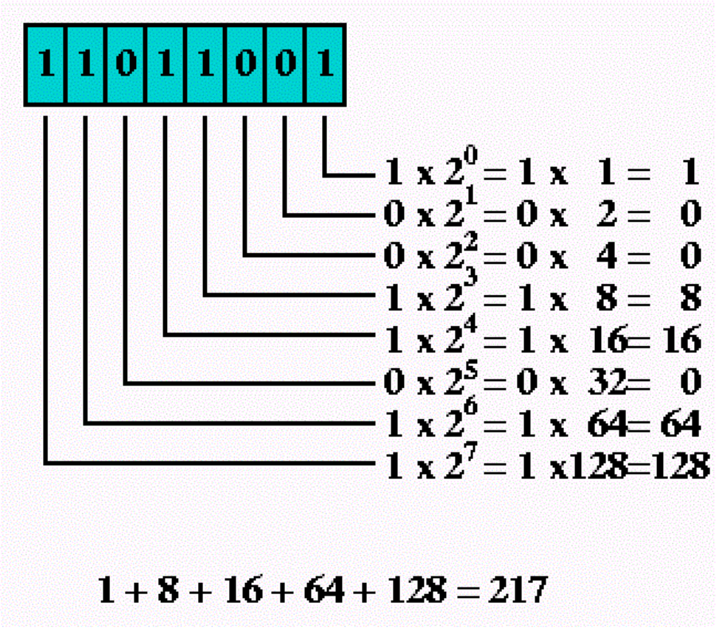
2^{30} (1,073, 741,824)is Giga, denoted "G"

Number conversions for r=2

Decimal to Binary



Binary to Decimal



Number conversions for r=2 with fractions

Decimal to Binary

			Result Digit	
$.3125 \times 2$	=	0.625	0	(MSB)
$.625 \times 2$	=	1.25	1	
$.25 \times 2$	=	0.50	0	
$.5 \times 2$	=	1.0	1	(LSB)

$0.3125_{10} = .0101_2$

Binary to Decimal

Zero Point

0 . 1 0 1 1

$1 \times 2^{-4} = 0.0625$

$1 \times 2^{-3} = 0.125$

$0 \times 2^{-2} = 0$

$1 \times 2^{-1} = 0.5$

0.6875_{10}

Octal & Hexadecimal Numbers

- For inspection by human user
- Example:

$$1\,0\,1\,1\,0\,1\,0_2 = 132_8 = 5A_H$$

$$\underbrace{0\,0\,1}_{1}\,\underbrace{0\,1\,1}_{3}\,\underbrace{0\,1\,0}_{2}{}_2 = \underbrace{1}_{0\,0\,1}\quad\underbrace{3}_{0\,1\,1}\quad\underbrace{2}_{0\,1\,0}{}_8$$

$$\underbrace{0\,1\,0\,1}_{5}\,\underbrace{1\,0\,1\,0}_{A}{}_2 = \underbrace{5}_{0\,1\,0\,1}\quad\underbrace{A}_{1\,0\,1\,0}{}_{16}$$

- Same for fractions

Complements

- For simplifying the subtraction operation (making the subtrahend negative and add with minuend)
- Two types of complements for any base r:
  - r’s complement
  - (r-1)’s complement

r's Complement

- N=a positive number in base r
- n=number of digits at *integer part* of N
- (r's complement of N) $= r^n - N$
- Example:
  - 10's complement of $(52520)_{10}$ = 10^5 - 52520 = 47480
    - Here, the number of digits in integer part is 5.
  - 10's complement of $(0.3267)_{10}$ = 10^0 – 0.3267 = 1-0.3267 = 0.6733
    - Here, the number of digits in integer part is 0.
  - 10's complement of $(25.639)_{10}$ = 10^2 – 25.639 = 100-25.639 = 74.361
    - Here, the number of digits in integer part is 2.

r's Complement…

- Example:
  - 2's complement of $(101100)_2 = 2^6 - (101100)_2 = 1000000_2 – 101100_2 = 010100_2$
    - Here, the number of digits in integer part is 6
  - 2's complement of $(0.0110)_2 = 2^0 - (0.0110)_2 = 1_2 – 0.0110_2 = 0.1010_2$
    - Here, the number of digits in integer part is 0
- Second Rule:
  - Leaving all least significant zeros unchanged and next least significant digit will be subtracted from r and others will be subtracted from (r-1).
- Third Rule:
  - After learning (r-1)'s complement.

(r-1)'s Complement…

- N=a positive number in base r
- n=number of digits at *integer part* of N
- m= number of digits at *fractional part* of N
- ((r-1)'s complement of N) $= r^n - r^{-m} - N$
- Example:
  - 9's complement of $(52520)_{10}$ = 10^5 - 10^0 - 52520 = 10^5 - 1 - 52520 = 47479
    - Here, n=5 and m=0.
  - 9's complement of $(0.3267)_{10}$ = 10^0 – 10^{-4} - 0.3267 = 0.9999-0.3267 = 0.6732
    - Here, n=0 and m=4.
  - 9's complement of $(25.639)_{10}$ = 10^2 – 10^{-3} - 25.639 = 99.999-25.639 = 74.360
    - Here, n=2 and m=3.

(r-1)'s Complement…

- Example:
  - 1's complement of $(101100)_2 = 2^6 - 2^0 - (101100)_2 = 1000000_2 - 1 - 101100_2 = 010011_2$
    - Here, n=6 and m=0
  - 1's complement of $(0.0110)_2 = 2^0 - 2^{-4} - (0.0110)_2 = 0.1111_2 - 0.0110_2 = 0.1001_2$
    - Here, n=0 and m=4
- Second Rule:
  - Leaving all the digits subtracted from (r-1).
- Third Rule of r's complement
  - Addition of r^{-m} with the (r-1)'s complement
- Note: Complement of the complement restore the number to its original value.
  - r's complement of N is $(r^n - N)$ and complement of $r^n - (r^n - N) = N$

Subtraction with r's Complement

- Subtraction method ,taught earlier, uses the **borrow** concept.
- Easiest way for people but less efficient for digital system
- Subtraction using complement is more efficient
- Subtraction of two positive numbers (*M*-*N*) both of base r done as follows:
  - Add the minuend M to the r's complement of the subtrahend N
  - Inspect the result obtained in step 1 for an end carry:
    - If an end carry occurs, discard it
    - If an end carry doesn't occur, take the r's complement of the number/result obtained in step 1 and place a negative sign in front

Subtraction with r's Complement…

- Example:
  - $M=72532_{10}$ and $N=03250_{10}$ (Answer: 69282_{10})
  - $M=3250_{10}$ and $N=73532_{10}$ (Answer: -69282_{10})
  - $M=1010100_2$ and $N=1000100_2$ (Answer: 0010000_2)
  - $M=1000100_2$ and $N=1010100_2$ (Answer: -10000_2)
- Proof: Homework

Subtraction with (r-1)'s Complement…

- Similar with r's complement except for one variation called "end around carry"
- Subtraction of two positive numbers (*M*-*N*) both of base r done as follows:
  - Add the minuend M to the (r-1)'s complement of the subtrahend N
  - Inspect the result obtained in step 1 for an end carry:
    - If an end carry occurs, add 1 to the least significant digit (end around carry)
    - If an end carry doesn't occur, take the (r-1)'s complement of the number/result obtained in step 1 and place a negative sign in front

Subtraction with (r-1)'s Complement…

- Example:
  - $M=72532_{10}$ and $N=03250_{10}$ (Answer: 69282_{10})
  - $M=3250_{10}$ and $N=73532_{10}$ (Answer: -69282_{10})
  - $M=1010100_{2}$ and $N=1000100_{2}$ (Answer: 0010000_{2})
  - $M=1000100_{2}$ and $N=1010100_{2}$ (Answer: -10000_{2})
- Proof: Homework

Comparison between 1's and 2's Complements

- Both of them have the advantages and disadvantages
- Implementation:
  - 1's complement:
    - Easier to implement (changing of 0s and 1s)
  - 2's complement:
    - Implemented in two ways:
    - Adding 1 at the least significant digit of the 1's complement
    - Leaving all ending 0s in the least significant positions and the first 1. Then changing all the 0s and 1s

Comparison between 1’s and 2’s Complements

- Subtraction:
  - 1’s complement:
    - Requires two arithmetic addition operations when an end around carry occurs
  - 2’s complement:
    - Only one arithmetic addition operation is required
- Another disadvantage of 1’s complement:
  - Two arithmetic zero: one with all 0s (positive) and another with all ones (negative)
  - Example: 1100-1100=0 (using 1’s and 2’s complement)

Comparison between 1's and 2's Complements

- 1's complement = logical inversion
- So 2's complement is only used in conjunction with arithmetic operation.
- So when only word *complement* is occurred, it will be 1's complement

Binary Code

- Electronic Digital system
  - Signal – two distinct values – 0 and 1
  - Circuit element – two stable states – on and off (LED, Switches)
- Bit – A binary digit
- n distinct bits – can be represented in a group of 2^n distinct elements
- A group of 2^n distinct elements – count in binary number from 0 to (2^n-1)
- If the total number of distinct elements is not equal to 2^n, some bit combinations will be unassigned (Example: BCD)

Decimal in Binary Code

- Numbers are represented in **binary** or in **decimal through binary code**
  - Storage – binary coding
  - Arithmetic operation – binary form
- Binary codes for decimal digits require minimum 4 bits (?)
- **Binary Coded Decimal (BCD)** – binary equivalent
  - Weight – 8,4,2,1
- **84-2-1** – weight 8,4,-2,-1
- **2421**– weight 2,4,2,1
- **5043210**– weight 5,0,4,3,2,1,0 (each code has two 1s)
- **Excess-3** – unweighted code – (obtained from the corresponding BCD +3)

Decimal in Binary Code

Decimal digit	(BCD) 8421	Excess-3	84-2-1	2421	(Biquinary) 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

Decimal in Binary Code

- **BCD** – most natural
- **Others** – self complementary – 9's complement can be obtained by flipping 1s and 0s.
- Example: 395 $(001111111011)_{2421}$ – 604 $(110000000100)_{2421}$
- **Biquinary** – error detection property – two 1s and five 0s

Error Detection Code

- **Binary information** – *transmitted* through communication medium such as wires or radio waves
- **External Noise** – in physical communication medium changes bit values from 0 to 1 or vice versa
- **Error detection code** – to detect error during transmission
  - Detected error can't be corrected but indicated
- Usual procedure is to observe the frequency of errors
  - Error –randomly occurred – nothing done/ retransmission of that message
    - Not that much effective error
  - Error – often occurred - system is checked for malfunction
    - Distort meaning of the received message

Error Detection Code…

- Parity bit – an extra bit included with the message to make the total number of 1s either odd or even
- Handle of parity bit during information transfer –
  - Sending end –
    - **Parity Generation** – from the message, generate the parity bit, P
    - The message including its parity bit, P sent to the destination
  - Receiving End
    - **Parity Check** – all incoming bits are checked whether proper parity is adopted or not. If the checked parity does not correspond to the adopted one, error detected.
    - If matched with adopted parity, discard the parity bit, P and sent to the original service
- The parity method detects the presence of odd number of errors. Even number of errors are undetectable. (?)

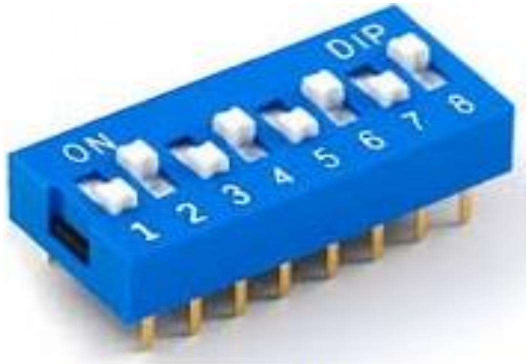
Error Detection Code…

D_3	D_2	D_1	D_0	Even-parity P	Odd-parity P
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	1

Data Block	Parity bit	Code word
0000	0	00000
0001	1	00011
0010	1	00101
0011	0	00110
0100	1	01001
0101	0	01010
0110	0	01100
0111	1	01111
1000	1	10001
1001	0	10010
1010	0	10100
1011	1	10111
1100	0	11000
1101	1	11011
1110	1	11101
1111	0	11110

The Reflected Code

- The benefit of the reflected code over binary numbers is that *a number in the reflected code changes by only one bit as it proceeds from one number to the next.*
- ***Motivation:*** *To indicate position/state , closing or opening switches are in several devices.*
- *If they use natural binary number, state 3 (011) and 4 (100) are next to each other but all three bits are different.*
- *Physical switches are not ideal, unlikely to change states exactly in synchrony*



8-way DIP switches

The Reflected Code…

- *In the brief period of changing, the switches will read some spurious position.*
- *Transition might look like 011-001-101-100*
- *The observer can not tell if that is reading a real position or a transitional state between two states.*
- *The reflected binary code solves this problem- changing only one switch at a time – cyclic property.*
- *Also known as Gray code, Single distance code (Hamming distance=1)*

The Reflected Code…

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

Alphanumeric Code

- A **binary code** of a group of elements consisting of ten decimal digits, 26 letters of the alphabet, and certain number of special symbols like {$,#,.,/…}
- Also known as *alphameric*
- More than 36 characters.
- At least required: $\log_2 (36) = 5.1699 \approx 6$ bits
- Upper and lower case letters and other characters increase the number of bits
- Internal code (6 bits), ASCII code (7 bits), EBCDIC code (8 bits)

Alphanumeric Code…

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

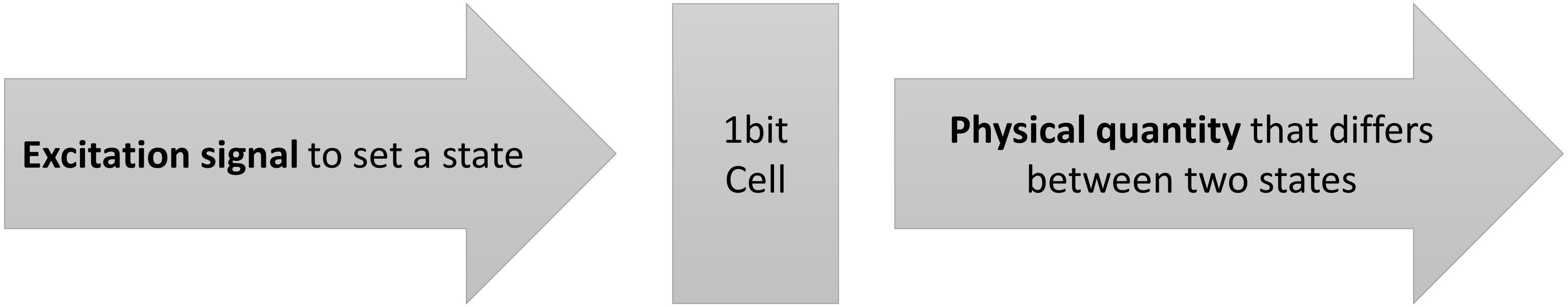
Converting the text "hope" into binary

Characters:	h	o	p	e
ASCII Values:	104	111	112	101
Binary Values:	01101000	01101111	01110000	01100101
Bits:	8	8	8	8

ComputerHope.com

Binary Storage and Registers

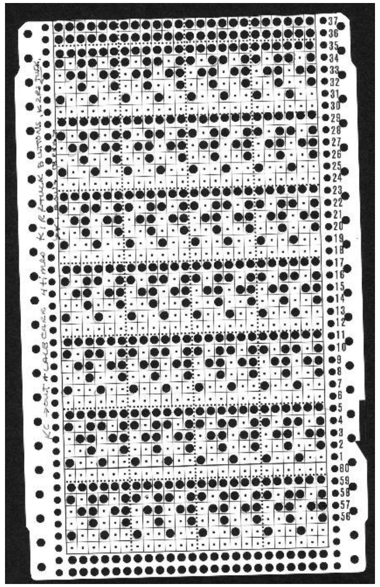
- Information in binary form – stored in **binary storage elements** for individual bits
- Binary Cell – having two stable states – store one bit of information

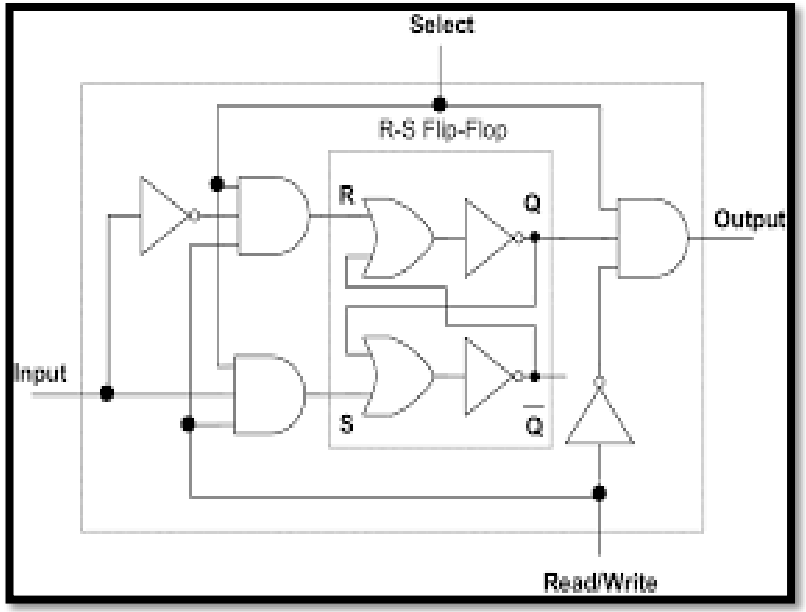


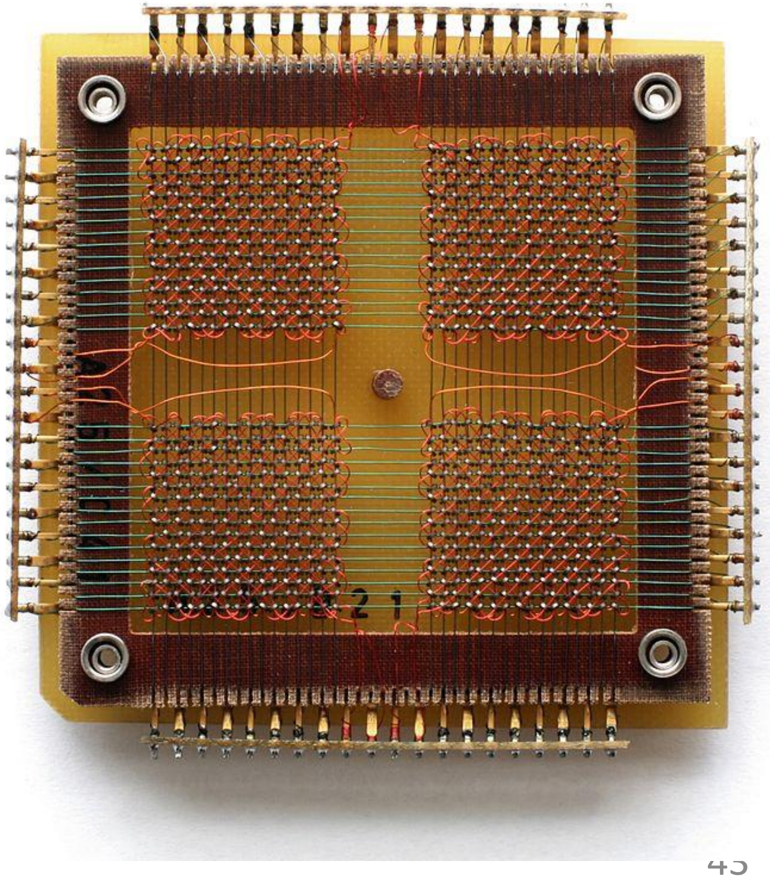
- Examples: Binary cells as flip-flop, ferrite cores, punch card with holes

Binary Storage and Registers

- Binary Cell Examples:







Registers

- **Register** – a group of binary cell
- **N-cell register** – store n bit discrete information
- **State of register** – n-tuple number of 1s and 0s
  - Each bit designating the state of one cell in the register
- **Content of a register** – a function of the interpretation of stored information in it.
- Example of 16 bits register:

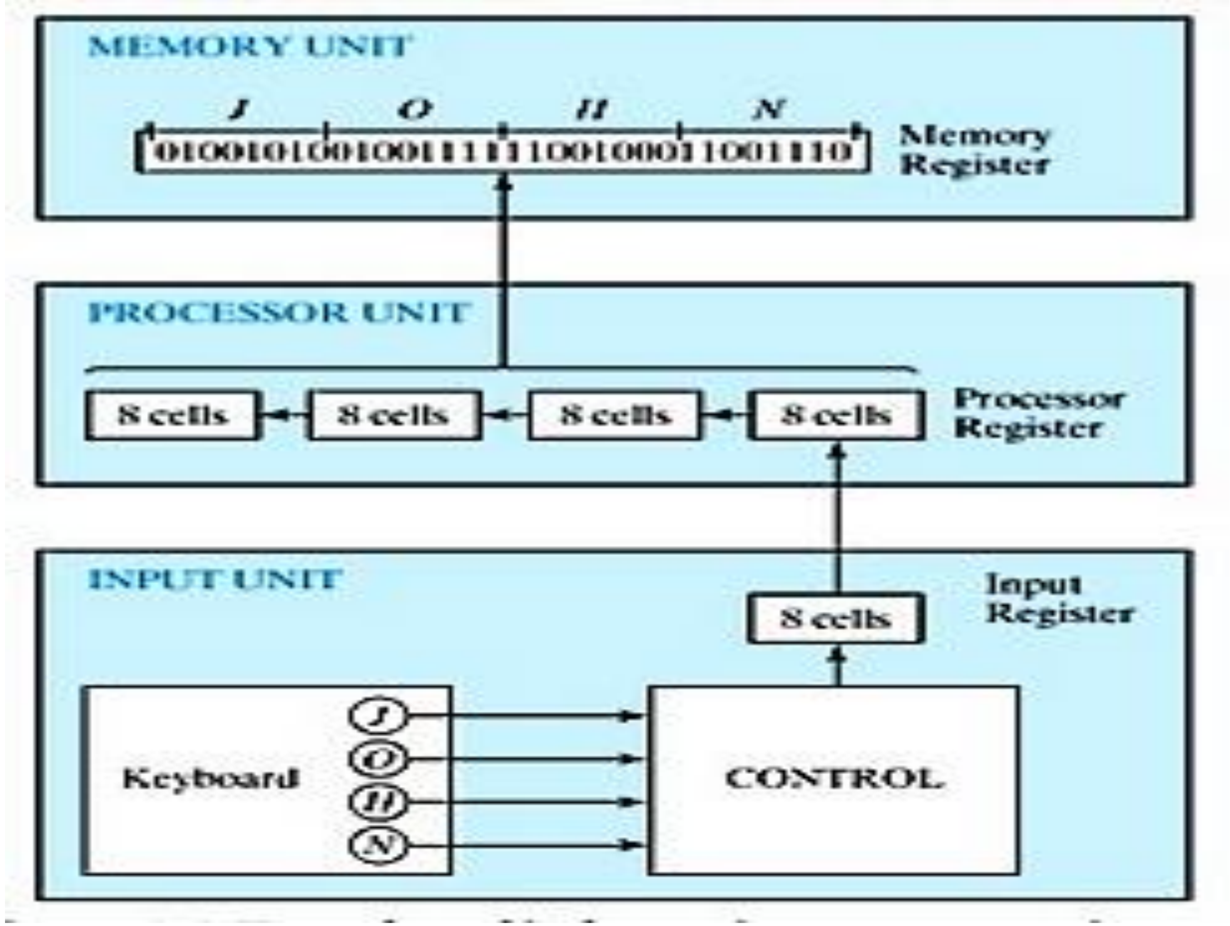
1	1	0	0	0	0	1	1	1	1	0	0	1	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

- Same bit configuration may be interpreted differently for different types of elements of information (types should be synched with the computer)

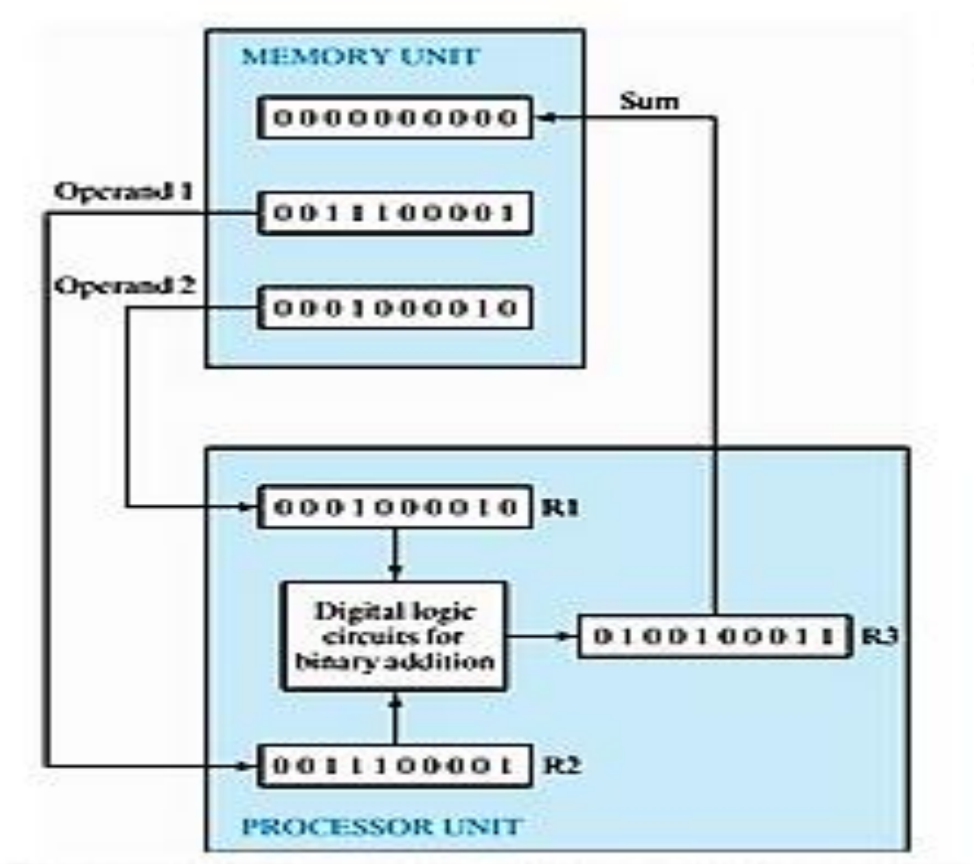
Register Transfer

- Registers in different components:
  - **Processor Unit:** store operands upon which operations are performed
  - **Control Unit:** keep track of various computer sequences
  - **I/O devices:** store information transferred to or from the device
- To process binary information, a computer must have:
  - **Devices** which hold the data to be processed (Register)
  - **Circuit elements** which manipulate data (Logic circuit)

Transfer of Information among Registers



Example of Binary Information Processing



Binary Logic

- Binary Logic – deals with two discrete values and a logic operation
- Basic operations in **Binary Logic**:
  - **AND:** denoted as *xy* or *x.y* and output will be true when both of the inputs are true. (Multiplication)
  - **OR:** denoted as *x+y* and output will be true when any of the inputs is true. (ADD)
  - **NOT:** denoted as prime *(x')* and output will not be equal to x.
- Binary Logic and Binary Arithmetic are not same!
  - **Binary Logic** deals with only logic (1+1=1)
  - **Binary Arithmetic** deals with binary number (1+1=10)

Binary Logic

- **Truth Table:** A table of all possible combination of the input variables with the output based on the definition of the logical operation.

NOT

x	x'
0	1
1	0

AND

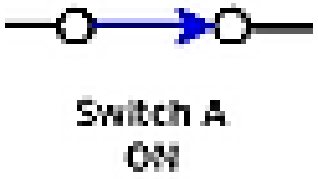
x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

OR

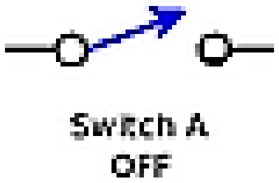
x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

Switching Circuits and Binary Signal

- **Application of Binary logic** – simple switching circuits (made of switches)
- Binary logic variable ***A*** can be represented as a switch ***A*** as following :

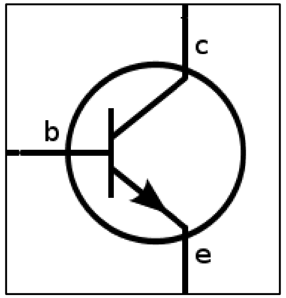


Logic 1

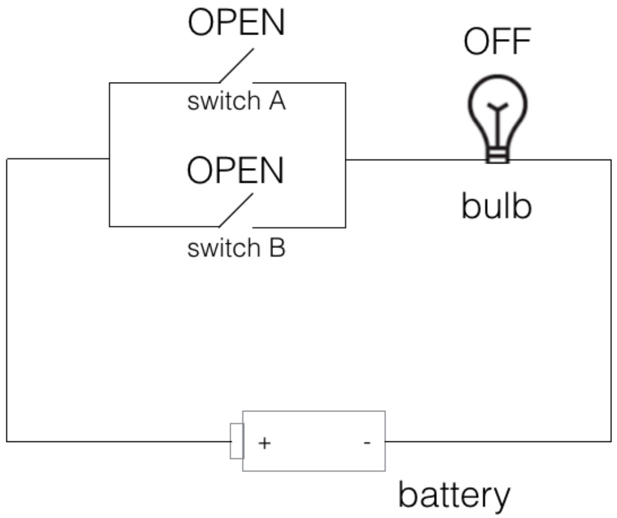


Logic 0

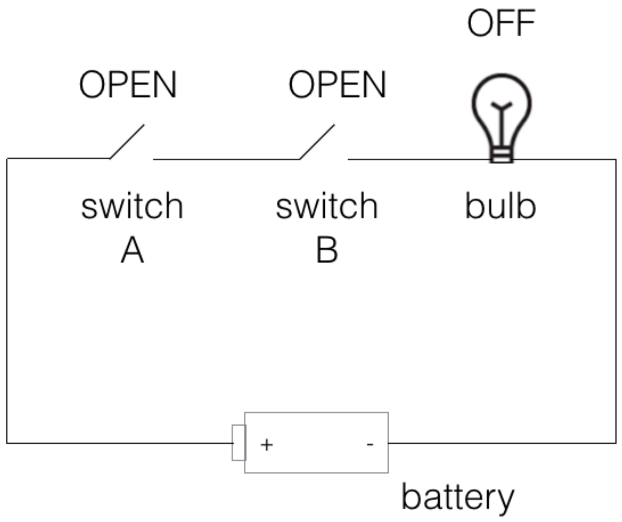
- Electronic digital circuit uses **transistor** as switches – named as switching circuit.
  - Conduct current – switch on
  - Not conducting current – switch off



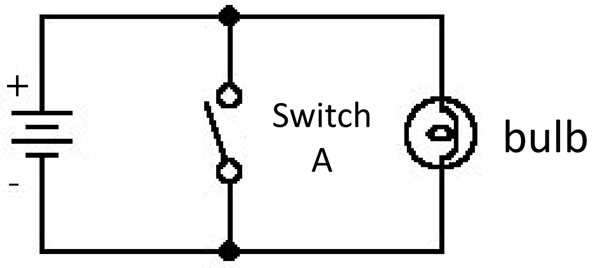
Switching Circuits



L = A + B



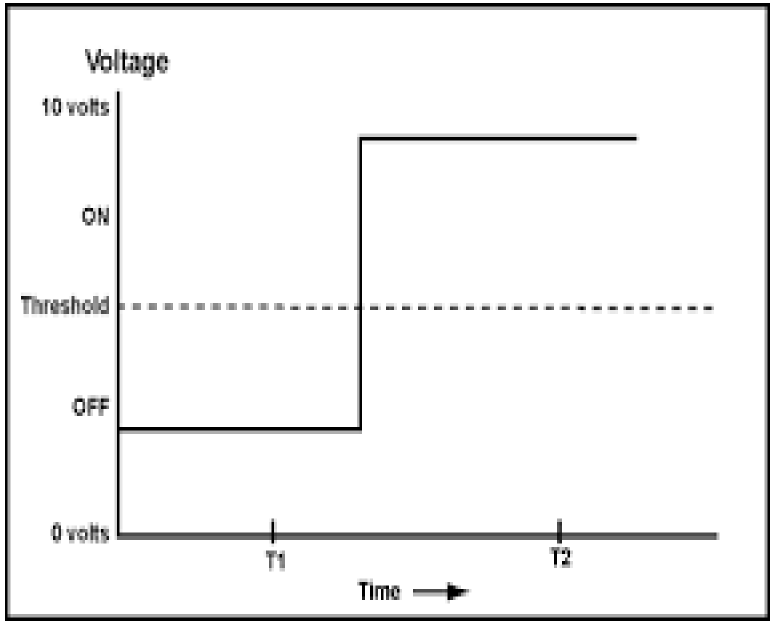
L = A . B

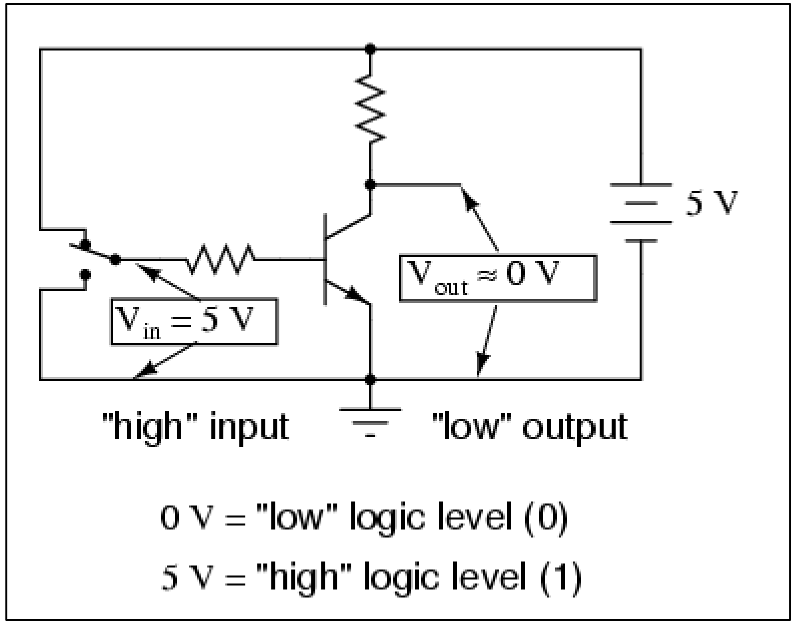


L = A'

Switching Circuits

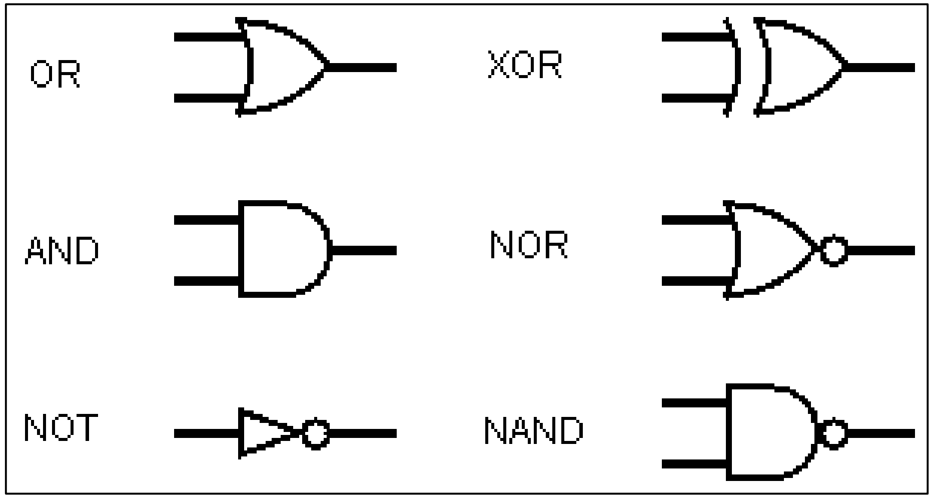
- Switches – controlled electrical signal – current or voltage
- Voltage operated circuit – two separate voltage level
- Current operated circuit – (in transistors) cut off or saturation states

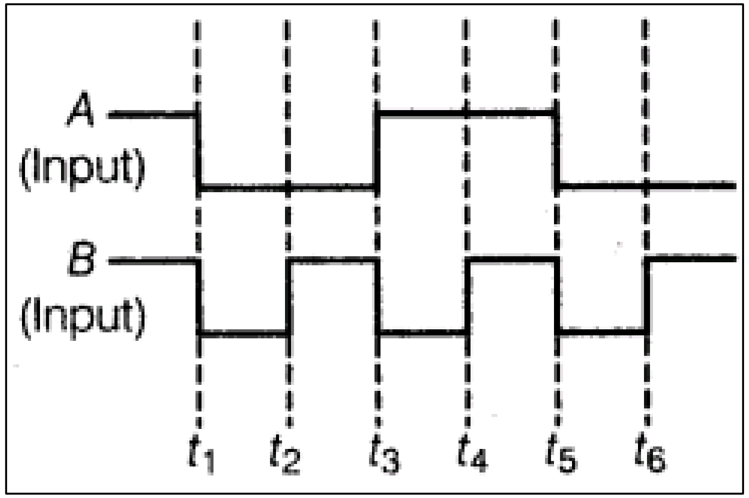




Logic Gate

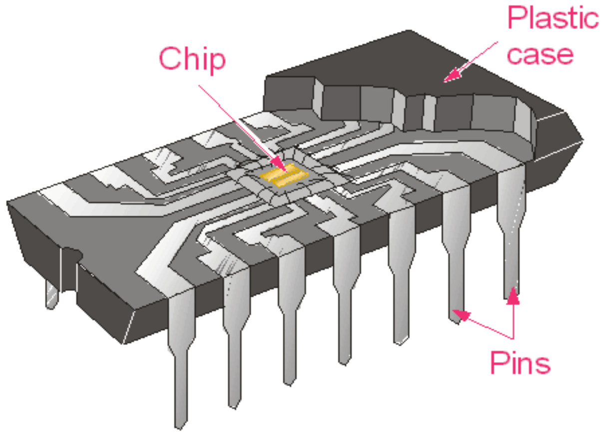
- Logic gate – establish logical manipulation path carrying one bit of information
- Also named as digital circuit, logic circuit or switching circuit
- Mathematical representation – Boolean algebra





Integrated Circuits

- Digital circuit constructed with **integrated circuit**
- IC – small semiconductor crystal named as **chip**
  - Components of *Chip* – transistors, diodes, resistors, capacitors and so on
  - These components are interconnected inside
  - This chip is mounted on a metal/plastic package and connections are made of external pin
  - Differ from other detachable electronic circuit
- Two types of packages
  - Flat
  - Dual in Line



Integrated Circuits

- **Advantages:**
  - Small in size
  - Cost effective
  - Reduced power consumption
  - High reliability against failure
- **Linear and Digital IC** – continuous and discrete data
- Based number of gates inside – SSI, MSI, LSI, VLSI IC