CSE 4205
Digital Logic Design

Digital System

Course Teacher: Md. Hamjajul Ashmafee

Assistant Professor, CSE, IUT

Email: ashmafee@iut-dhaka.edu

Digital System and Digital Computer

- The present technological period is highly influenced by **digital systems**
  - Also known as ***digital age***
- **Digital systems** are used in, for example:
    - Communication
    - Business transections
    - Traffic controls
    - Spacecraft guidance
    - Medical treatment
    - Weather monitoring
    - Commercial, Industrial and scientific enterprises

  - Most of these devices have a specific ***digital computer*** embedded within them
  - A digital computer is ubiquitous because of its **generality/universality**

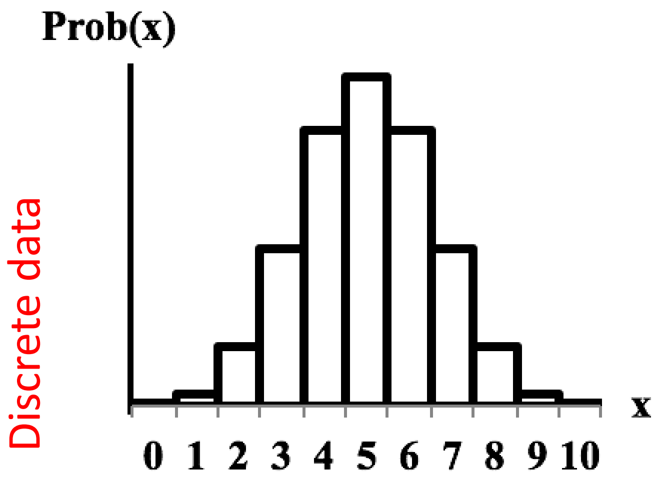
Generality of Digital Computer

- **Generality** of digital computer:
  1. It follows a sequence of instructions (**program**)
  2. It operates on data (**user given data**)
  3. Based on that program it generates user specific results
- For this reason, **computer** is now ***universal*** to perform a variety of information processing task.

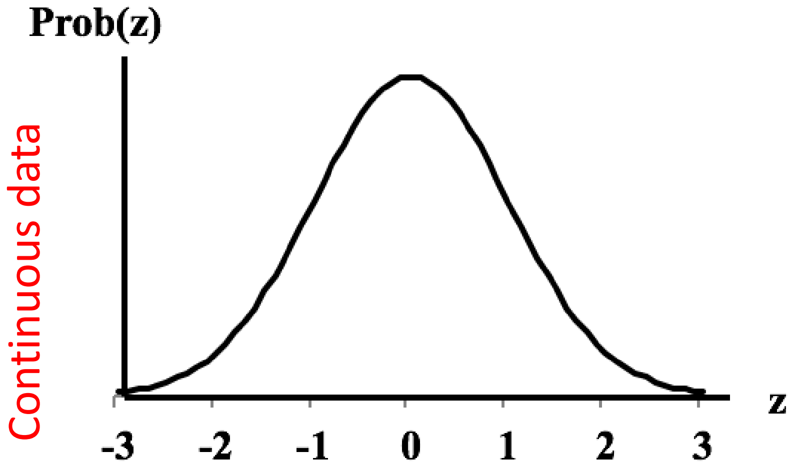
Digital System: Main Feature

:: Process Information

- Process and represent discrete elements of information (**not continuous**)
  - **Number of elements** of information is limited to a finite numbered set
    - These elements are required to represent the information
  - **Example:** decimal digits = {0,…,9}



Binomial Distribution
Discrete Data & Discrete Probability Curve



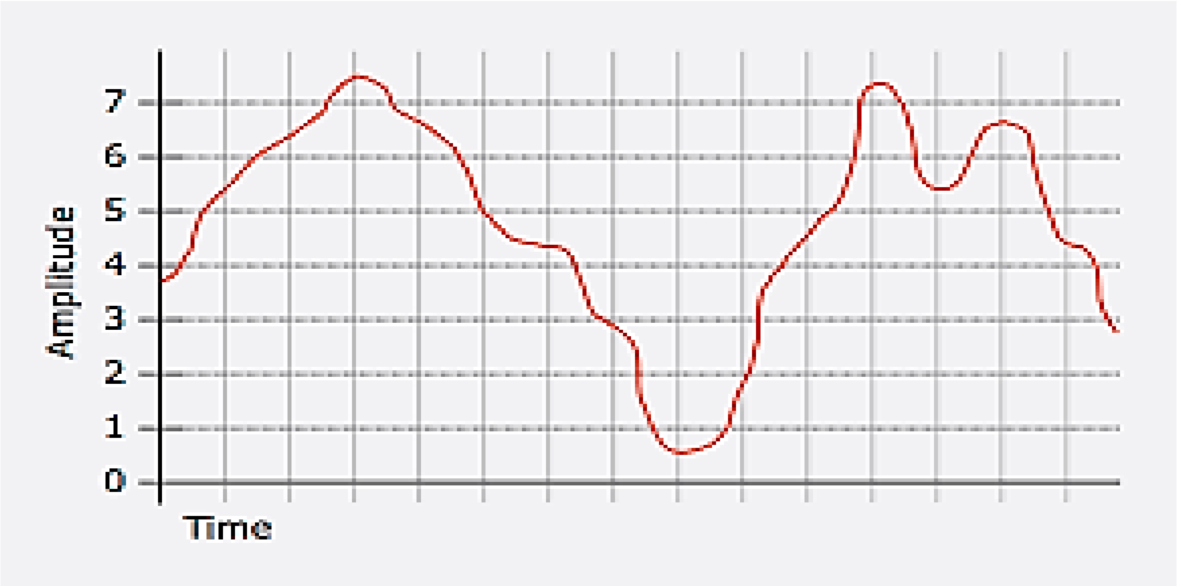
Standard Normal Distribution
Continuous Data and Continuous Probability Curve

Digital System: Discrete Information

- **Information** basically is represented as **signal**
- **Electric signal (continuous)** – **voltage** and **current** are most common
  - Generally used by **transistors** in circuitry of electronic devices
- Signals in most electronic digital systems are two discrete values (**Binary**)
- **Bit** – a binary digit
  - Two **discrete** values – 0 and 1
  - Sometimes discrete information may be represented as group of bits – **binary code**
- To interpret the **bit/digit pattern**, **code system** plays a very crucial role
  - **Example: $(0111)_2$ and $(0111)_{10}$ are not same in meaning**
- Binary system is so applicable to represent not only the **numbers** but also other **symbols/characters**
  - Largely used in different **electric devices**

Analog Information

- Analog data from any continuous process or nature needs to be converted to digital one
    - Sometimes an ADC is required to perform quantization on continuous system



Digital Information

- Digital data is the discrete and discontinuous representation of information
  - Example: payroll schedule of any company having names, ID, salary, income tax, and so on.

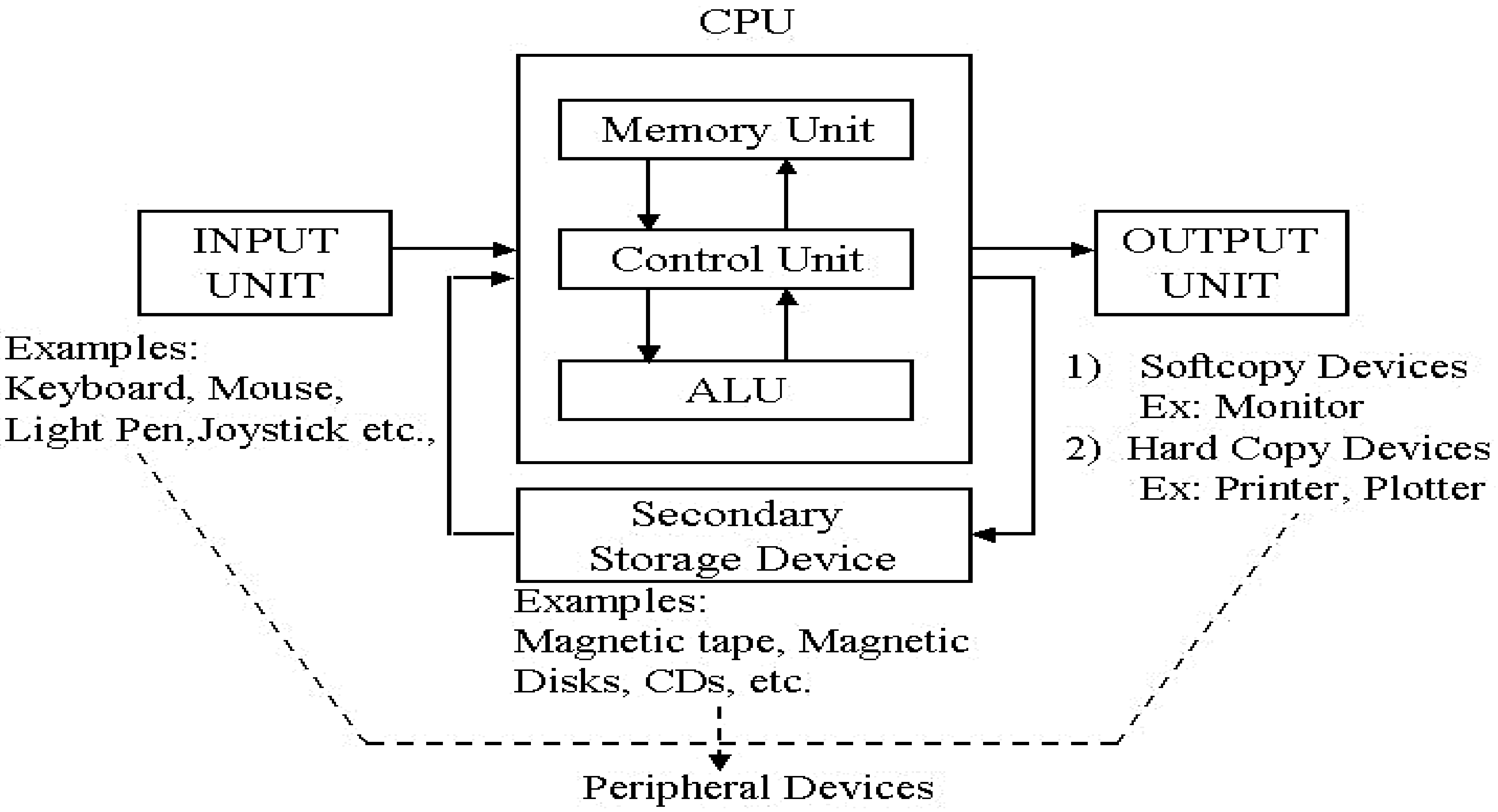
063013 — JD Edwards World — Page - 2
Summary Payroll Register — Date - 7/22/17
Period - 08/15/17

Payroll ID - 692 Company - Home . . 00100 Model Finan/Distrib Co (Mktg)
Home Bus. Unit . . 90 Administrative Department

. . . . EMPLOYEE Number	. . . . . Name	Hours	Wages	Benefits	Gross Pay	Deductions	Taxes	Net Pay	Check Control	I C	Err Msg
7503	Kraton, Ralph	80.00	2,333.33	102.92	2,333.33	239.89	587.38	1,506.06	173930	N	
7504	Meade, Jane	80.00	1,458.33	120.90	1,458.33	390.49	253.82	814.02	173948	N	
7505	Mastro, Robert	80.00	1,572.92	86.22	1,572.92	34.11	369.52	1,169.29	173956	N	
7506	Mayeda, Donald	80.00	616.00		616.00	230.19	125.39	260.42	173964	N	
7510	Moralez, Jesus	80.00	520.00	11.44	520.00	13.80	103.73	402.47	173972	N	
Total.		400.00		321.48		908.48		4,152.26			
			6,500.58		6,500.58		1,439.84				

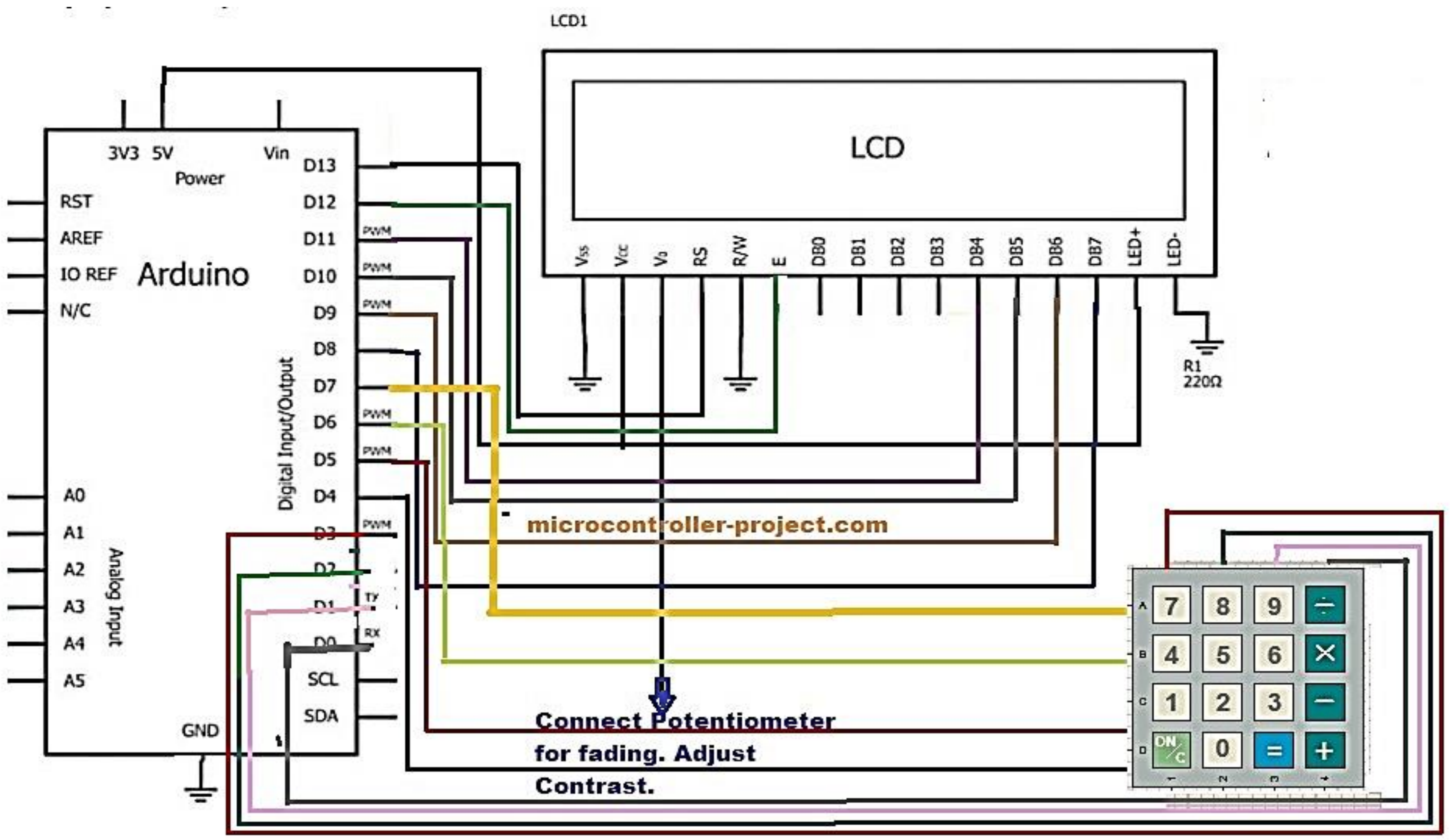
Digital System: An Example

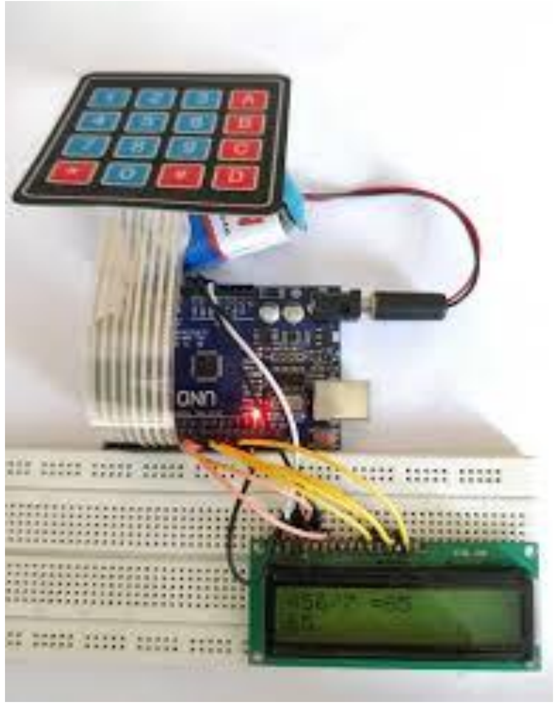
Block diagram of a general-purpose digital computer



Digital System: An Example

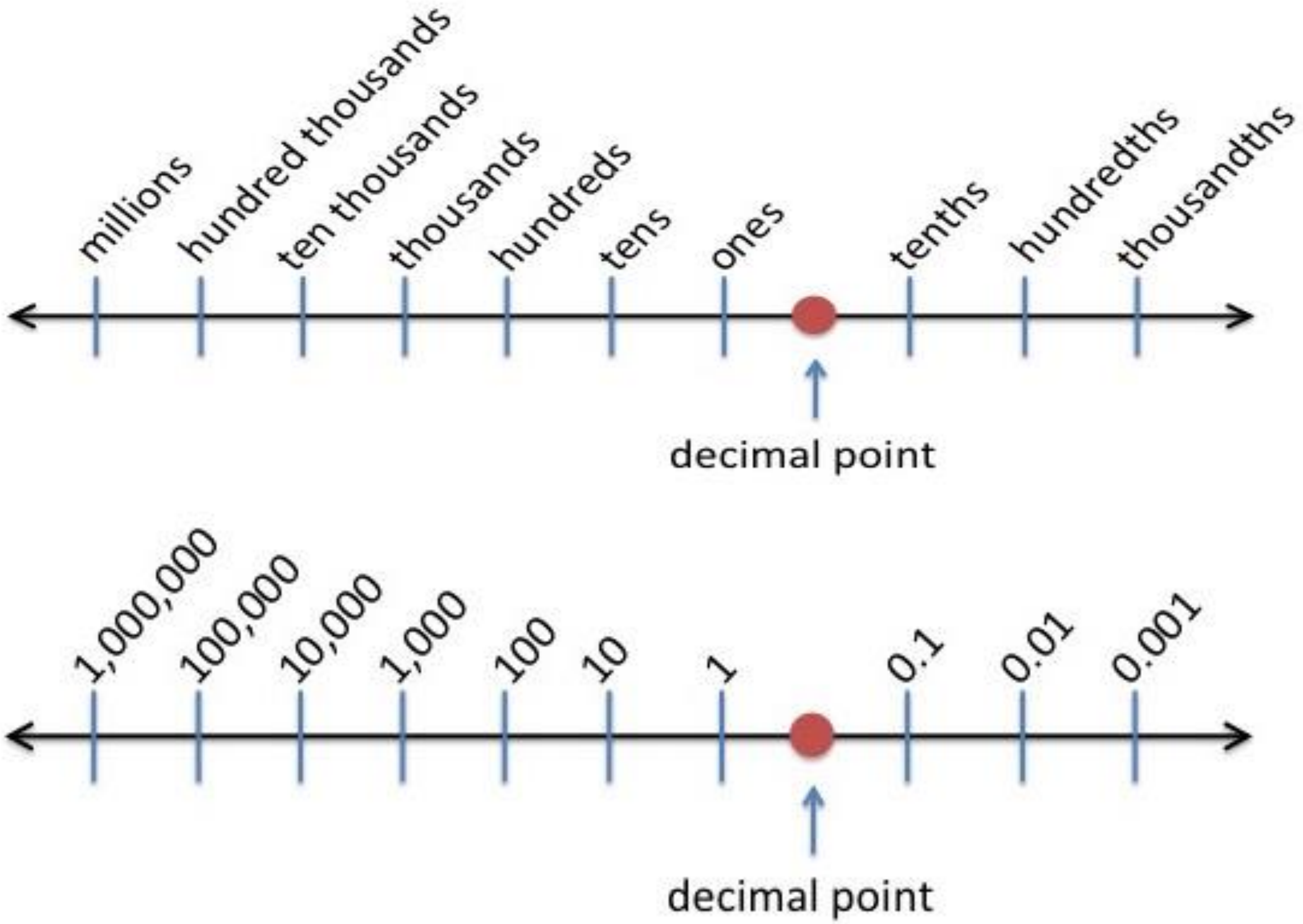
Example: An Electronic Calculator





Number system: Decimal Numbers

Power of 10



Decimal Numbers: Example

Write 12,357 in expanded form.

12,357

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

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

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

Note: Convention is to write only the **digits** in any number. From their positions, powers of 10 is deduced increasing from right to left

Number system: Formal Explanation

- A number with a ***radix point*** in any base ***b*** (**positional number system**) :

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

k = # of digits in integers
l = # of digits in fractions

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

- $n = \pm \left(\sum_{i=-l}^{i=(k-1)} \boldsymbol{S}_i . \boldsymbol{b}^i \right)_{\boldsymbol{b}}$
  - Where ***S*** is the set of symbols (alphabet) and ***b*** is the base or radix.
  - $\mathbf{0} \leq \boldsymbol{S_i} \leq \boldsymbol{b} - \mathbf{1}$
- **Example:** Decimal, Binary, Octal, Hexadecimal number system

Number system: Its Operations

- Arithmetic Operations (**Unary and Binary**)
  - **Addition** (augend, addend, sum)
  - **Subtraction** (minuend, subtrahend, difference)
  - **Multiplication** (multiplicand, multiplier, product)
  - **Division** (dividend, divisor, quotient, reminder)
  - **Negation, Absolute, Increment, Decrement**
- **Carry** and **borrow** is also used in any base ***b***.
- How to make the computer understand about decimal point and sign of any number? (mantissa and exponent)

Number System: 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

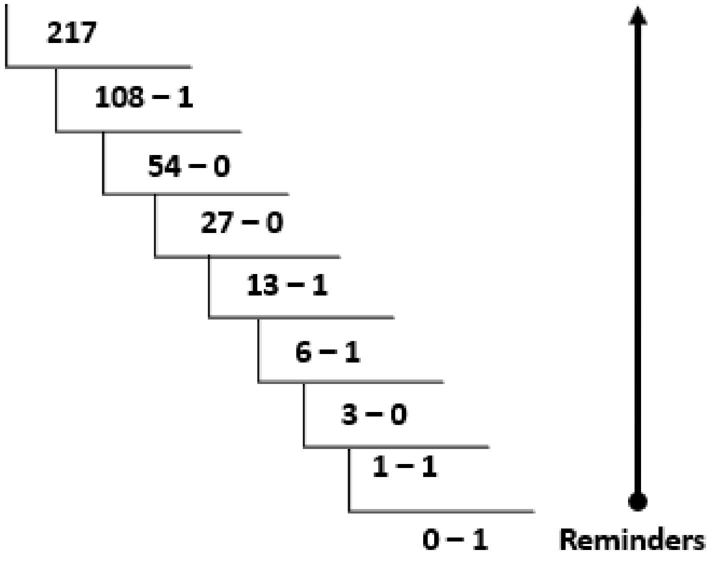
10

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} (1.0995116e+12)is Tera, denoted “T"
- 2^{50} (1.1258999e+15)is Peta, denoted “P“
- 2^{60} (1.1529215e+18)is Exa, denoted “E“
- 2^{70} (1.1805916e+21)is Zetta, denoted “Z“
- 2^{80} (1.2089258e+24)is Yotta, denoted “Y"

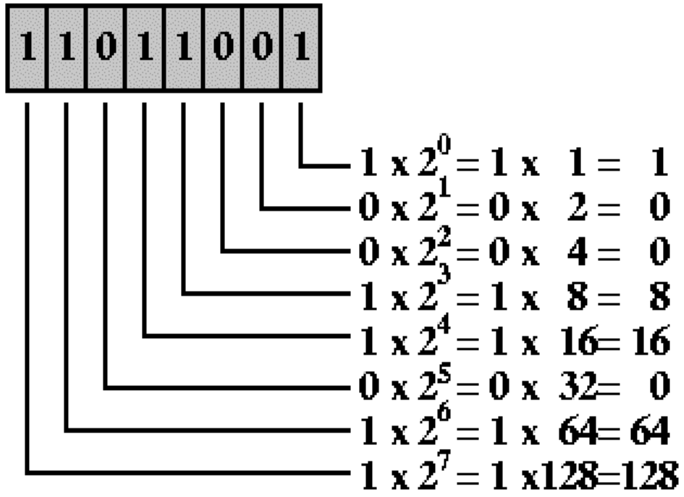
Number conversions for r = 2 for Integers

Decimal to Binary



$(217)_{10} = (11011001)_2$

Binary to Decimal



$1 + 8 + 16 + 64 + 128 = 217$

Number conversions for r=2 with Fractions

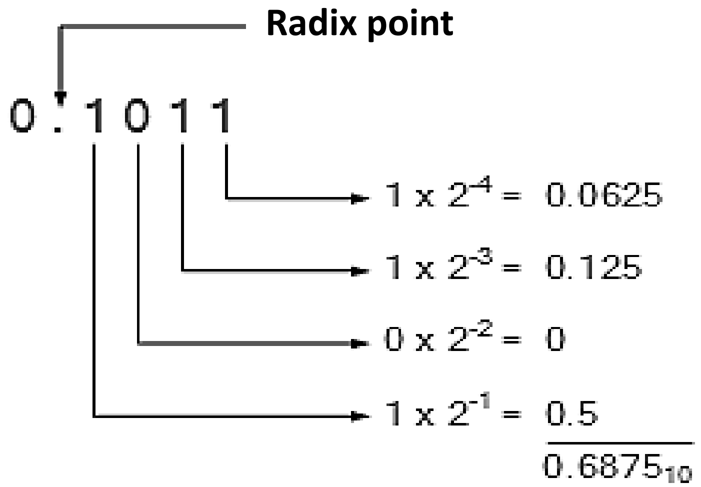
Decimal to Binary

			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)

$0.3125_{10} = .0101_2$

- Multiplication is used instead of division
- Integers instead of reminders are considered

Binary to Decimal



Octal & Hexadecimal Numbers: Examples…

Reverse way…

$$
\begin{array}{lcccccccc}
(673.124)_8 = & (110 & 111 & 011 & \cdot & 001 & 010 & 100)_2 \\
 & 6 & 7 & 3 & & 1 & 2 & 4
\end{array}
$$

$$
\begin{array}{lccccc}
(306.D)_{16} = & (0011 & 0000 & 0110 & \cdot & 1101)_2 \\
 & 3 & 0 & 6 & & D
\end{array}
$$

Complements of Numbers

- Complements are used to **simplify** the subtraction operation
    - **Concept:** Make the **subtrahend** negative and add with minuend
- Also used in different logical operation like **negation**
- **Two types** of complements for any base ***r*:**
    1. *r*'s complement **or** radix complement
    2. (***r*-1**)'s complement **or** diminished radix complement
  - **Example:** 2's and 1's complement in binary, 10's and 9's complement in decimal

(r-1)'s Complement

- Also known as **Diminished Radix Complement**
- **If:**

  N = a number in base r
  n = total number of digits at ***integer part*** of N
  m = total number of digits at ***fractional part*** of N

- **(r-1)'s complement of** $N = r^n - r^{-m} - N$ [**Original Rule**]
- Examples in decimal:
  - 9's complement of $(52520)_{10}$ = $10^5 - 10^0$ - 52520 = 99999-52520 = 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 in binary:
  - 1's complement of $(101100)_2 = 2^6 – 2^0 - (101100)_2 = 111111_2 – 101100_2 = 010011_2$
    - 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
- Example in Octal or Hexa - **???** (subtracting each digit from 7 and 15 respectively)
- **Alternative method # 1:**
  - Leaving all the digits subtracted from $(r - 1)$. (instead of using any formula)

r's Complement

- Also known as **Radix Complement**
- **If:**

  N = a positive number in base ***r***

  n = number of digits at ***integer part*** of N
- **r's complement of $\boldsymbol{N} = \boldsymbol{r^n} - \boldsymbol{N}$ [Original Rule]**
  - Where $N \neq 0$; if $N = 0$, r's complement $= 0$
- Example in decimal:
  - 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 in binary:
  - 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
- **Alternative method # 1**
  - Leaving all least significant zeros unchanged, next first non-zero least significant digit will be subtracted from ***r*** and next all higher significant digits will be subtracted from ***(r-1)***.
- **Alternative method # 2**
  - **Addition** of r^{-m} with the ***(r-1)*'s complement**

Complement of Complement

- Complement of the complement restores the number to its original value.
  - **Proof:** r's complement of N is $(\boldsymbol{r^n} - \boldsymbol{N})$ and complement of $\boldsymbol{r^n}$ – $(\boldsymbol{r^n} - \boldsymbol{N}) = \boldsymbol{N}$
  - **Expression:** $(A')' = A$

Subtraction with r's Complement

- Subtraction method uses the **borrow** concept – earlier intuitive method.
  - Easiest way for human being **but** less efficient for digital system
  - A 1 is borrowed from higher significant minuend position if the minuend digit is smaller than the subtrahend digit
- Subtraction using complement is more efficient for computer system
- Subtraction of two positive numbers **(*M-N*); both of them being base *r*,** done as follows (*informal*):
  1. Add the **minuend M** to the **r's complement** of the **subtrahend N**
  2. **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 result obtained in step 1 and place a negative sign in front of it

Subtraction with r's Complement…

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

Subtraction with (r-1)'s Complement…

- **Similar** with subtraction 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:
  1. Add the minuend M to the (***r-1***)'s complement of the subtrahend N
  2. 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 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 $N=72532_{10}$ (Answer: -69282_{10})
  - $M=1010100_2$ and $N=1000100_2$ (Answer: 0010000_2)
  - $M=1000100_2$ and $N=1010100_2$ (Answer: -10000_2)

Overflow during Addition

- Important during operation with **signed binary numbers**
- **Problem:** the magnitude **exceeds limit** which can't be represented with the allotted number of bits.
- **Approach:**
  - If 2 **Two's Complement** numbers are added, and they both have the same sign (both positive or both negative), then overflow occurs **if and only if** the result has the opposite sign.
  - **Overflow occurs if:**
    - (+A) + (+B) = −C
    - (−A) + (−B) = +C

(−7) **1**001
\+ (−6) **1**010
\------------
(−13) 1 **0**011 = 3
= Overflow; as the **result** is positive.
(where it exceeds the largest (−)ve number is −8)

Signed Binary Number

- In decimal number system and ordinary arithmetic, it is easier to represent a **positive** and a **negative** number
  - A negative number is indicated by a **minus sign** and a positive number is indicated by a **plus sign**

1. Because of hardware limitation, it is customary to use **MSB** as **sign bit**
   - **Signed-magnitude system**
     - The convention is to make **sign bit 0** for positive and **sign bit 1** for negative number
     - **Point to be noted**: Both signed and unsigned binary numbers consist of 0s and 1s. User himself has to determine if a number is **signed** or **unsigned (user specified)**
       - **Example:** $(11001)_2$ can be equivalent of 25 or -9 as per consideration.

Signed Binary Number…

2. **Alternatively for arithmetic operations, signed-complement system** is more convenient to represent a negative number in **binary system**
    - In this system, a negative number is indicated by its complement (either ***r***'s or ***(r-1)***'s complement)
    - Usually, a positive number starts with **0**, whereas a complement starts with **1** (negative)
    - Between **1's** and **2's** complement, **2's** complement is more preferable
    - Practically, for any negative number in 1's or 2's complement, MSB should be 1.

- **Example:**
    - A number +9 represented in 1 byte storage as 00001001
    - But negative number -9 can be represented as:
        1. Signed-magnitude representation = 10001001
        2. Signed-1's-complement representation = 11110110
        3. Signed-2's-complement representation = 11110111

    Note: In all cases in negative numbers, we get 1 in MSB position

Signed Binary Number: Comparison Table

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Arithmetic Addition

- If we perform addition following the rules of **ordinary arithmetic**, we have to consider the **sign** and the **magnitude separately**
  - To calculate the final sign of the result we have to compare those magnitudes
  - Same rules will be applied in binary if **signed-magnitude system** is followed
  - **Example:** addition of +25 and -37
- In contrast, if **signed-complement system** is followed, there is **no such difficulties** to compare the magnitudes or handle the signs separately
  - Rather **simple addition** is applied here
  - **Concept:** In this type of addition, negative value must be represented in 2's complement form and carry out of the sign bit position (MSB) is discarded
  - **Advantage: If the result is negative, the result obtained is automatically in 2's complement form**
    - You might perform the 2's complement of the result again and put a minus sign in front of it.

Arithmetic Addition: Example

$$\begin{array}{rr}
+\ 6 & 00000110 \\
\underline{+13} & \underline{00001101} \\
+19 & 00010011
\end{array}
\qquad
\begin{array}{rr}
-\ 6 & 11111010 \\
\underline{+13} & \underline{00001101} \\
+\ 7 & 00000111
\end{array}$$

$$\begin{array}{rr}
+\ 6 & 00000110 \\
\underline{-13} & \underline{11110011} \\
-\ 7 & 11111001
\end{array}
\qquad
\begin{array}{rr}
-\ 6 & 11111010 \\
\underline{-13} & \underline{11110011} \\
-19 & 11101101
\end{array}$$

Note: Overflow problem should be maintained as the storage in computer system is limited

Arithmetic Subtraction

- Subtraction in signed-2's-complement format:
  - **Concept:** Take the 2's complement of the subtrahend (including sign bit) and add it to the minuend (including sign bit)
    - A **carry out** of the **MSB** position is discarded
- In signed-2's-complement, all subtraction operations are **changed to addition** operations as follows:

$$(\pm A) - (+B) = (\pm A) + (-B)$$
$$(\pm A) - (-B) = (\pm A) + (+B)$$

Binary Code

- Electronic Digital system
    - **Generates signal** – two distinct values – 0 and 1
    - **Has circuit element** – two stable states – on and off (***e.g.,*** LED, Switches)
- **Bit** – basic unit of information – a binary digit (0 or 1)
- Moreover, a **group of discrete elements** of information that is distinct among themselves can be also separated with a **binary code**
    - ***n*** **distinct bits of any code**– can be represented within a group of 2^n distinct elements/codes
    - A **group** of 2^n **distinct elements'** representation– **counts** in binary number **from 0 to (2^n-1)**
    - Mostly the **binary codes** are used for coded information rather than **binary numbers**
- **Note:** If the total number of distinct elements is not equal to 2^n (***i.e.,*** $\boldsymbol{total\ number} < \boldsymbol{2^n}$), some bit combinations will be unassigned (**e.g.,** BCD with 4 bits)

BCD Addition: Example # 1

4	0100	4	0100	8	1000
+5	+0101	+8	+1000	+9	1001
9	1001	12	1100	17	10001
			+0110		+0110
			10010		10111

BCD Addition: Example # 2

- Addition of two ***n***-digit BCD numbers follows same procedure significant position wise:

BCD	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

Subtraction:

579
- 237

234
- 179

Decimal in Other Binary Codes

- Other binary codes also requires 4 bits per decimal digit
  - But these 4 bits can be combined in different ways into 10 distinct combinations
    - Other 6 out of 16 combinations have no meaning and should be avoided
- Other binary codes as follows:

i. **2421** – weighting factors 2,4,2,1

ii. **84-2-1** – weighting factors 8,4,-2,-1
   - Has both **positive** and **negative** weight

iii. **5043210 (Bi-quinary)** – weighting factors 5,0,4,3,2,1,0
   - Each code has **two** 1s

iv. **Excess-3** – **unweighted code**
   - Obtained from the corresponding **BCD+3**

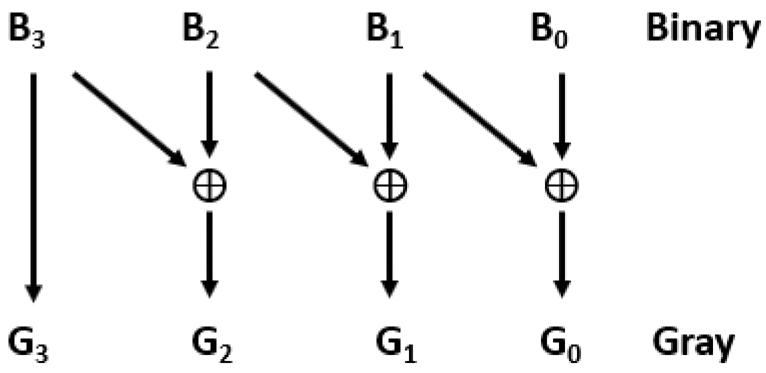
The Reflected Code or Gray Code

- To convert **continuous (analog) data** to discrete one, sometimes **reflected code** is more beneficial over binary code
- Primary benefit is that a number in the reflected code **changes by only one bit** as it proceeds from one number to the next.
  - **Example:** In natural binary number, state 3 (011) and state 4 (100) are in sequence **but** all three bits are different whereas reflected code for 3 (010) and 4 (110) overcome this problem
- Reflected code is basically used **where generating binary numbers** by the hardware **in normal sequence** which may produce **ambiguity** during transition **from** one state **to** the next

The Reflected Code: Comparison

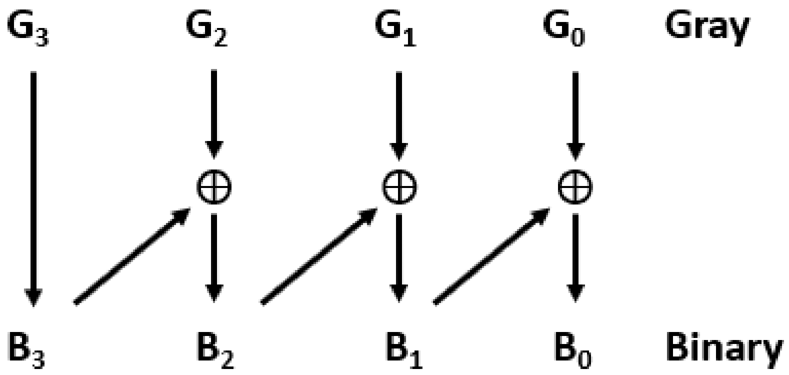
Decimal	Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

The Reflected Code: Approach



i.e.

$$\begin{aligned} G_3 &= B_3 \\ G_2 &= B_3 \oplus B_2 \\ G_1 &= B_2 \oplus B_1 \\ G_0 &= B_1 \oplus B_0 \end{aligned}$$



i.e.

$$\begin{aligned} B_3 &= G_3 \\ B_2 &= B_3 \oplus G_2 \\ B_1 &= B_2 \oplus G_1 \\ B_0 &= B_1 \oplus G_0 \end{aligned}$$

Alphanumeric Code

- Most of the computer application manipulate not only **numbers** but also other **characters or symbols**
  - A **binary code** represents a group of elements consisting of ten **decimal digits**, 26 **letters of the alphabet** in both cases, and certain number of **special symbols** like {$,#,.,/…}
  - This code is also known as ***alphanumeric code***
- More than 36 distinct characters/symbols.
  - **Considering Upper- and lower-case** letters and other **special characters**
- Minimum required number of bits: $\log_2(36) = 5.1699 \approx 6\, bits$
  - If both cases and other special characters are considered: $\log_2(36+26+32) = \log_2(94) = 6.56$
    - At least 7 bits are required
- **Example:** **Internal code** (6 bits), **ASCII code** (7 bits), **EBCDIC code** (8 bits)

Alphanumeric Code: ASCII – Basic Characters

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	∧	n	~
1111	SI	US	/	?	O	–	o	DEL

Alphanumeric Code: ASCII – Control Characters

Control Characters			
NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

Alphanumeric Code: Example

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

Error Detection Code

- **Binary information is usually:**
  - ***Transmitted*** through communication medium such as wires or radio waves
  - ***Processed*** in different electric circuitry
- **Problem: External Noise**
  - During those above-mentioned operations, medium accidentally changes bit values from 0 to 1 or vice versa (**alteration**)
- **Solution: Error detection code**
  - **To detect error** during operations
    - Detected error can't be corrected but identified
    - **Error Correction Code** (**?**) - **Alternative**

Error Detection Code: Approach

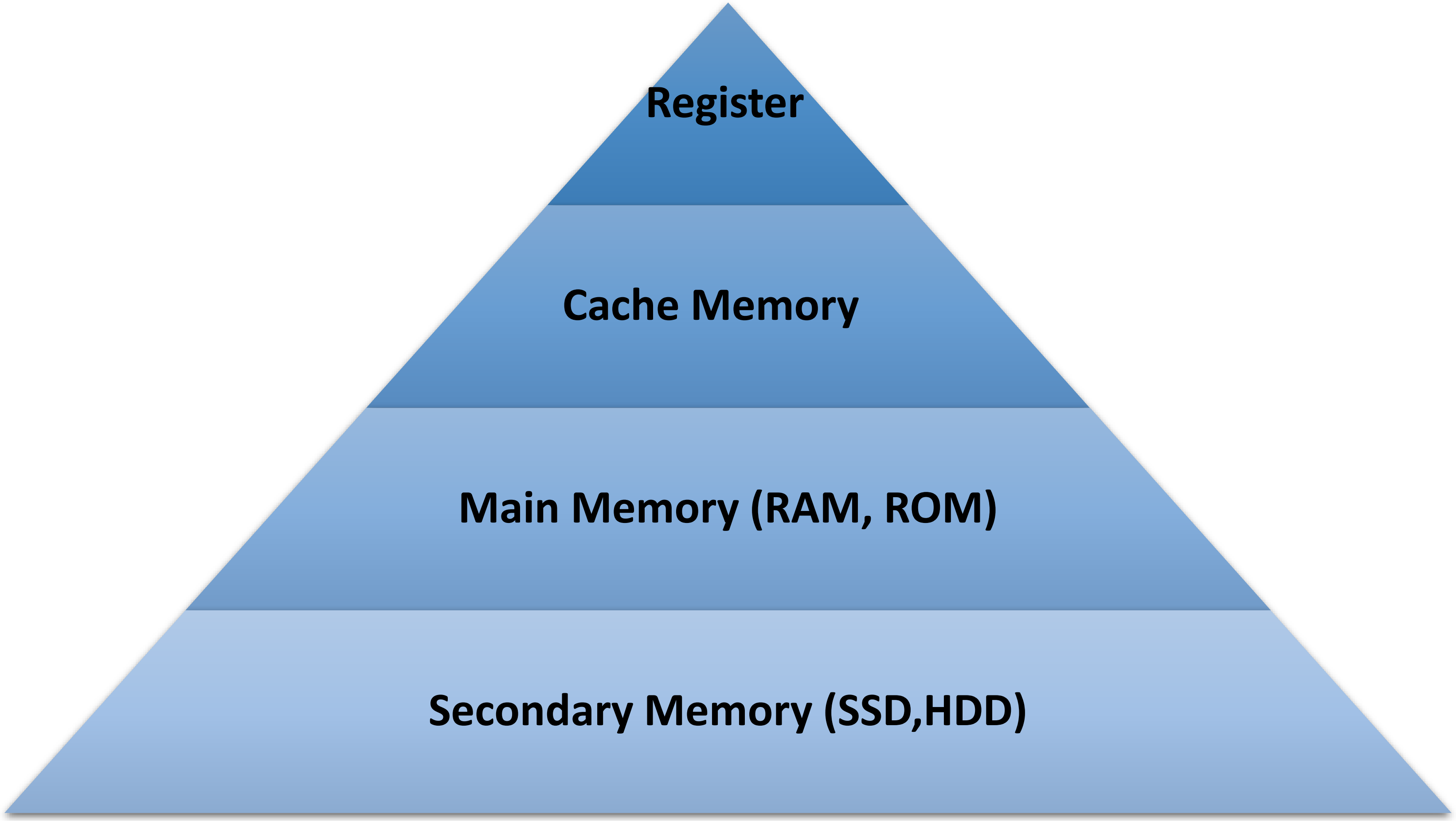
- **Parity bit** – an **extra bit** included with the message to make the total number of 1s either odd or even
- Handle of parity bit during information transfer –
  - **Sending end (Transmitter)**
    - **Parity Generation** – from the message, generate the parity bit, ***P***
    - The message including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

Error Detection Code: Example # 2

	With even parity	With odd parity
ASCII A = 1000001	01000001	11000001
ASCII T = 1010100	11010100	01010100

*** Parity bit is included in MSB position**

Binary Storage: Memory Hierarchy

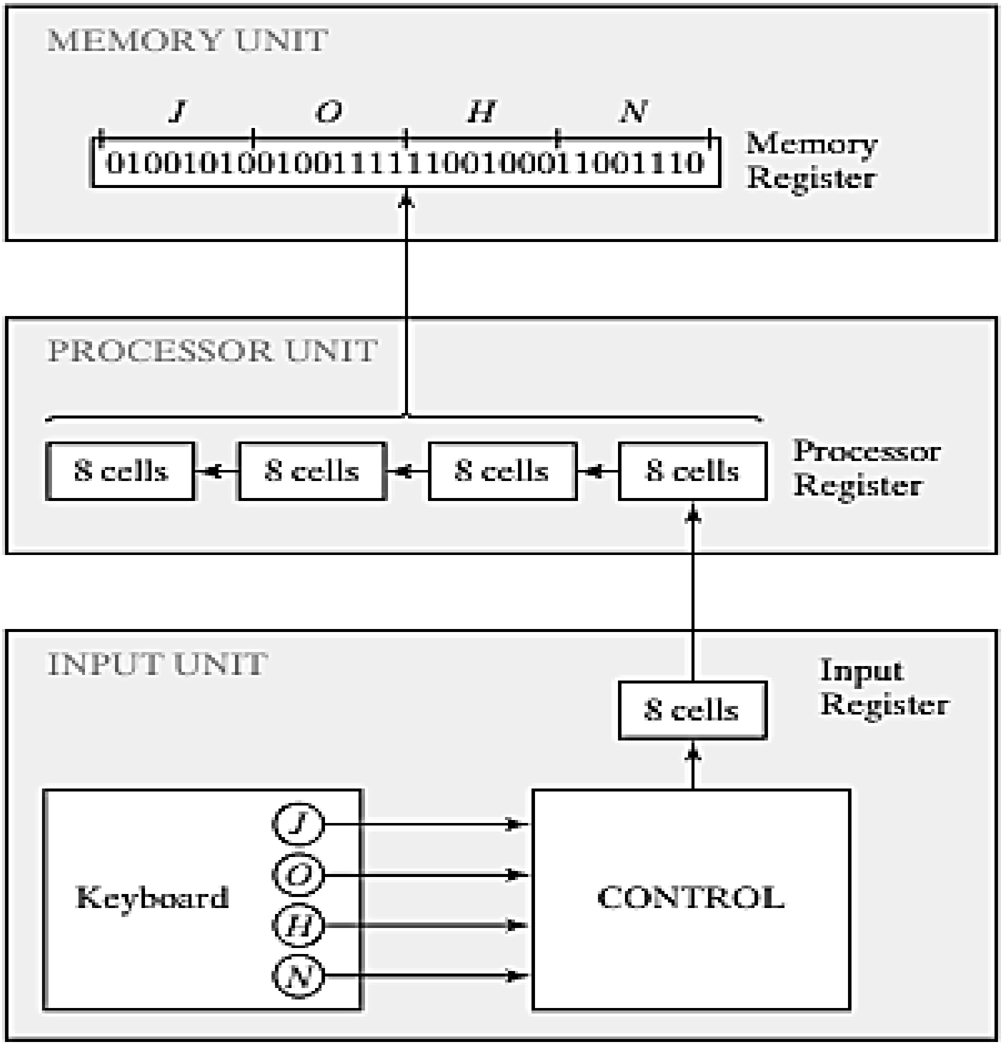


Registers

- **Register** – a **group** of binary cells
  - ***n*-cell register** – store ***n*** bit discrete information of any combination of 0s and 1s
- **A "state" of a register** – an **n-tuple number (???)** of 1s and 0s (***i.e.,*** in binary)
  - Each bit designating the **state** of **a corresponding cell** in the register
- **Content of a register** – can be defined as a **function** of the **interpretation of stored information** in it.

Transfer of Information among Registers

Register Level Operation

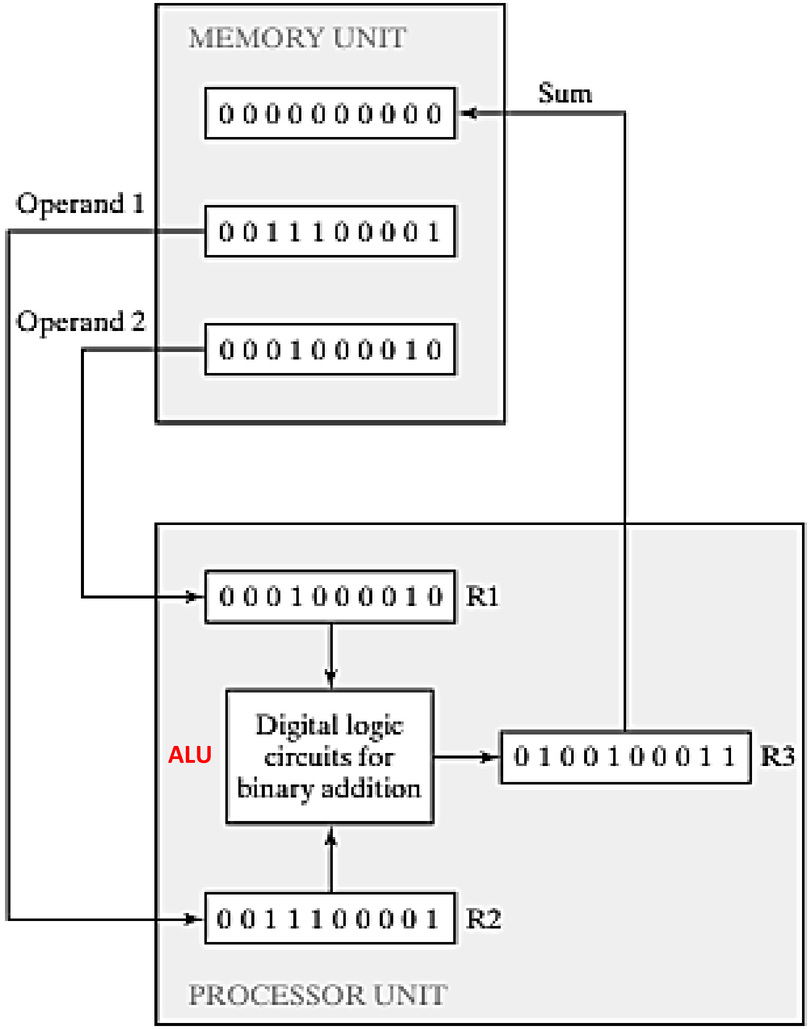


Memory unit stores data at desired location

Processor store data and perform shift operation

Input unit accepts data from user

Example of Binary Information Processing



Register Level Operation

Binary Logic

- **Binary logic** consists of binary **variables (operands)** and a set of **logical operations**
  - Basically, used to represent (or a unit of) **data processing**
  - Variables can be designated by letters from English alphabet
  - Deals with mostly **two discrete values (binary)**
    - Values can be defined as either **true or false**, **yes or no** and so on
- **Important:** **Binary Logic** and **Binary Arithmetic** are not same!
  - **Binary Logic** deals with only logic (1+1=1)
    - All variables represent any of the **two discrete values**
  - **Binary Arithmetic** deals with binary number (1+1=10)
    - All variables represent a **binary number** of one or more digits considering positional weights

Binary Logic: Basic Logic Operations

- **AND:**
  - **Denoted** as xy or $x.y$
  - **Output** will be true when both of the inputs are true.
  - Similar as multiplication in ordinary arithmetic
- **OR:**
  - **Denoted** as $x + y$
  - **Output** will be true when any of the inputs is true.
  - Similar as addition in ordinary arithmetic
- **NOT:**
  - **Denoted** as prime (x') or overbar $(\overline{\boldsymbol{x}})$
  - **Output** will not be equal to x.
  - Similar as negation in ordinary arithmetic

Binary Logic: Truth Table

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

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

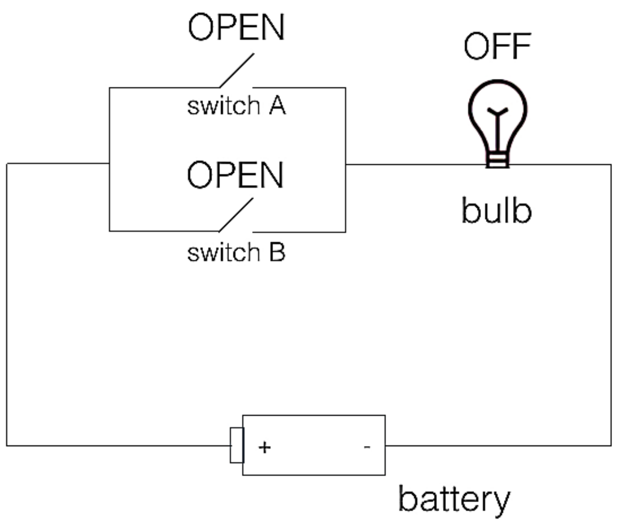
OR

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

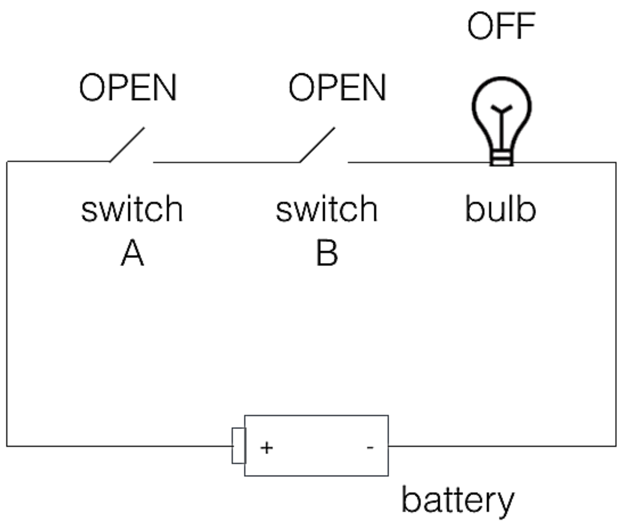
NOT

x	x'
0	1
1	0

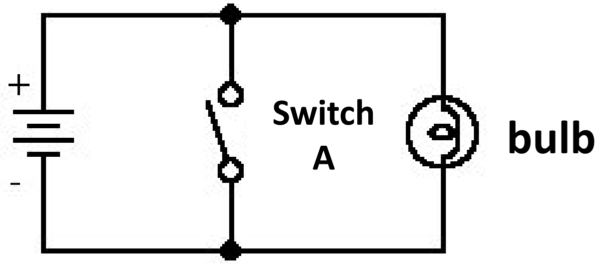
Switching Circuits: Basic Logic Operations



L = A + B (OR)



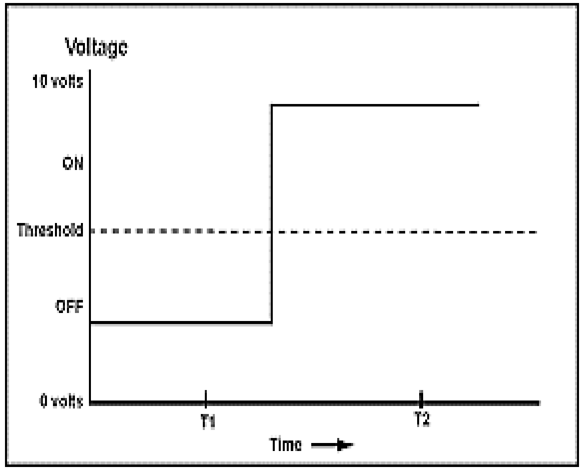
L = A . B (AND)

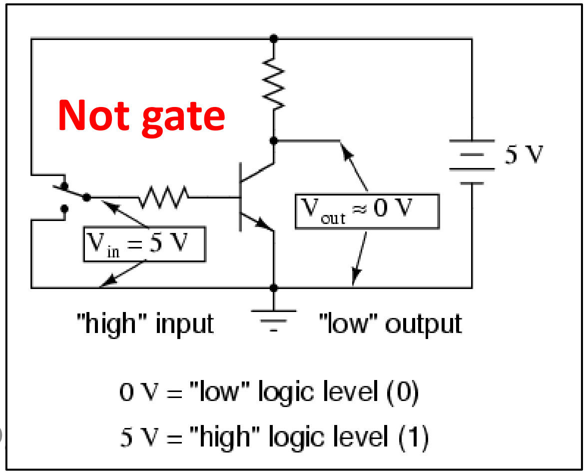


L = A' (NOT)

Switching Circuits: Basic of Switch

- **Switches (transistors)** control electrical **signal** based on current or voltage
  - **Voltage** operated/controlled circuit – two separate voltage level
    - Output depends on input voltage
    - Example of Switch: **FET (Field Effect Transistor)**
  - **Current** operated/controlled circuit – (in transistors) cut off or saturation states
    - Output depends on input voltage
    - Example of Switch: **BJT (Bipolar Junction Transistor)**





Logic Gates: Basic Idea

- **Logic gate:** An **electric circuit** that operate on one or more input signals to generate an output signal based on specific requirement(s)
    - Establish a logical manipulation **path** <u>producing one bit of information as output</u>
    - Unit of a **digital circuit**, **logic circuit** or **switching circuit**
- Electrical signal (***e.g.,*** voltage and current) exists as **analog signal** which has **continuous values** over a range (***e.g.,*** 0v to 5v)
    - These analog data must be **interpreted** to either of two discrete values, 0 and 1
    - In **ADC**, input signal in the specified range is responded with binary signal in output terminal
        - Region between two allowed regions are crossed during the **transition period**

Logic Gates: Symbols