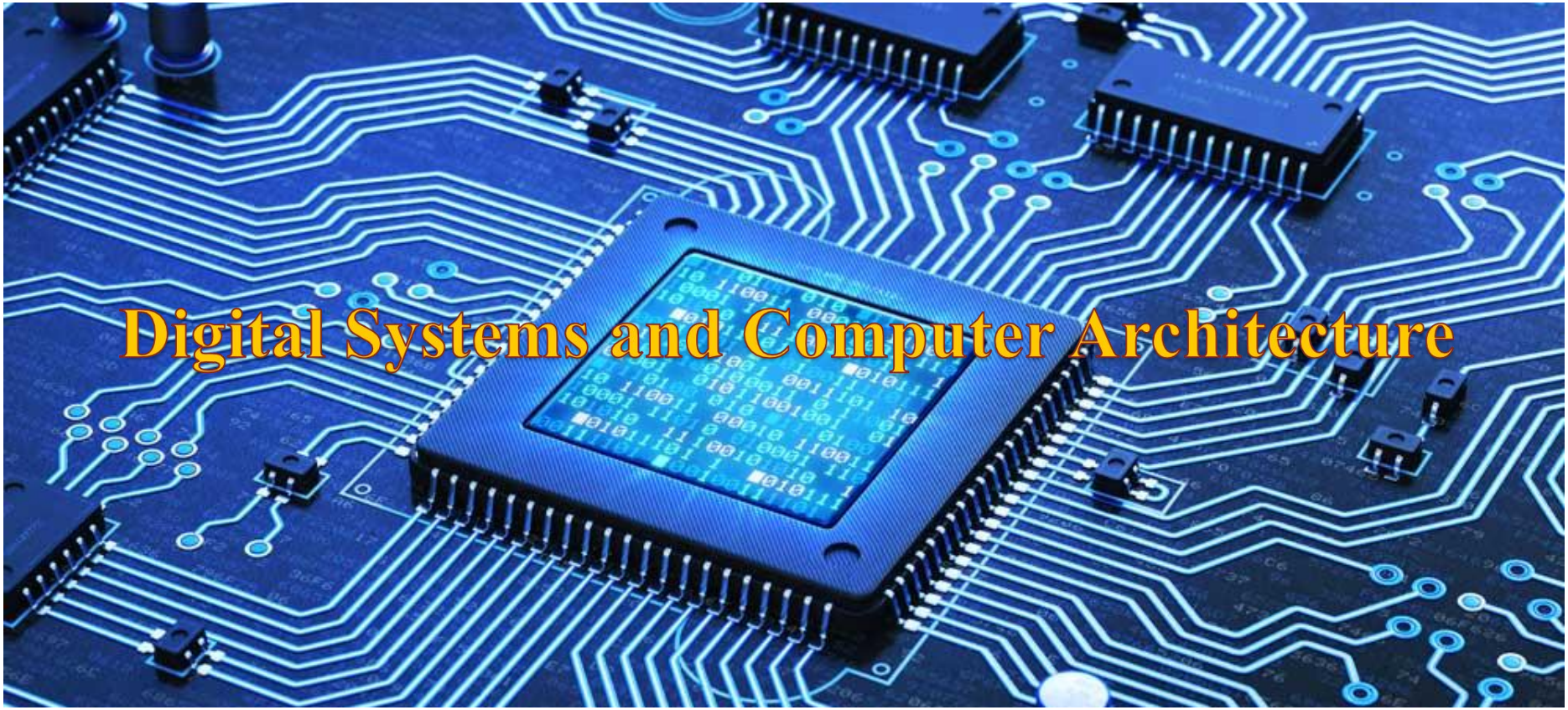


Digital Systems and Computer Architecture

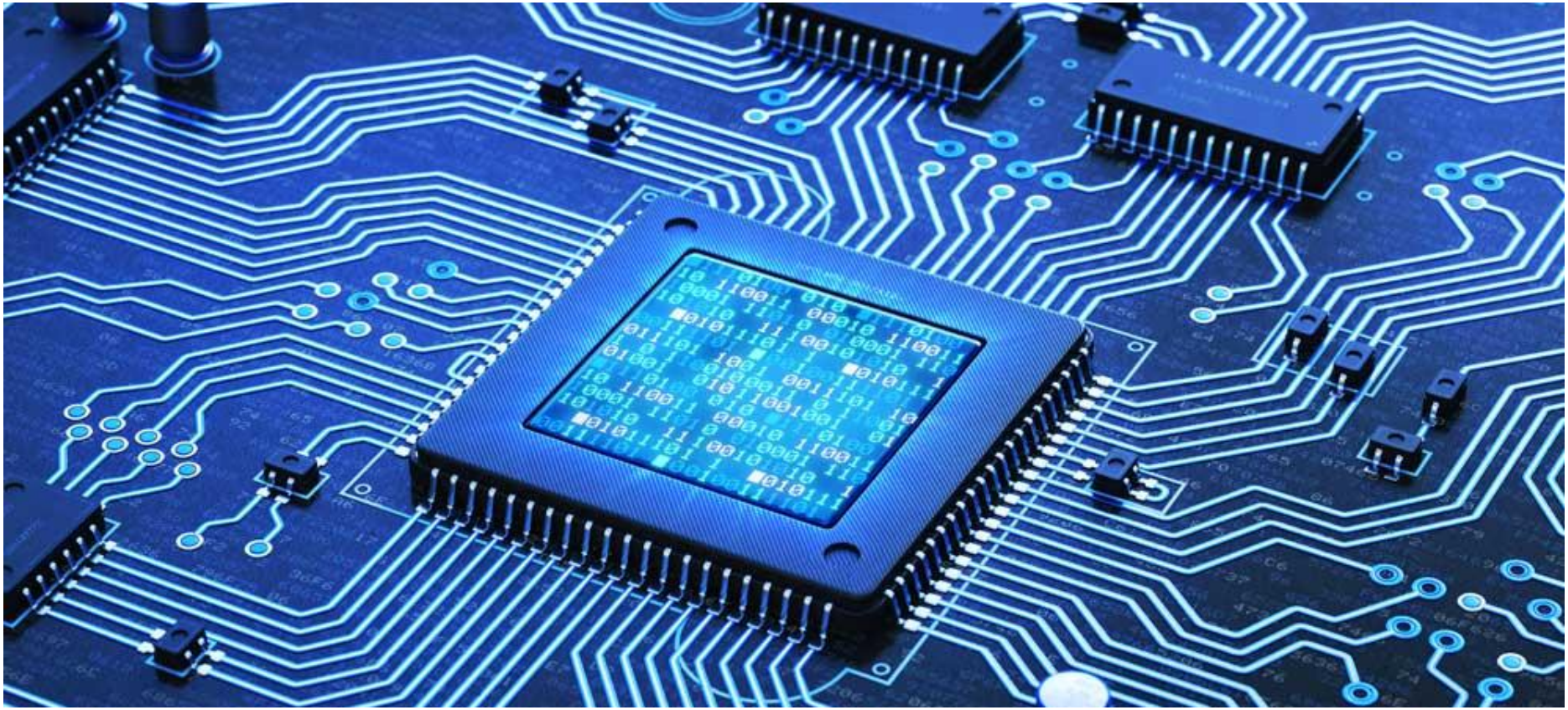
Session 2.1

Module 2

Number System and Codes

Session 2.1: Focus

- Analog Vs Digital
- Processing by Digital systems
- Number systems
- Binary, octal, hexadecimal
- Conversions from one to the other
- Binary codes and their classifications
- Weighted codes
  - Binary Coded Decimal (BCD)
- Non-weighted codes
  - Gray code
  - Optical Encoder Example using Gray Code
- ASCII codes



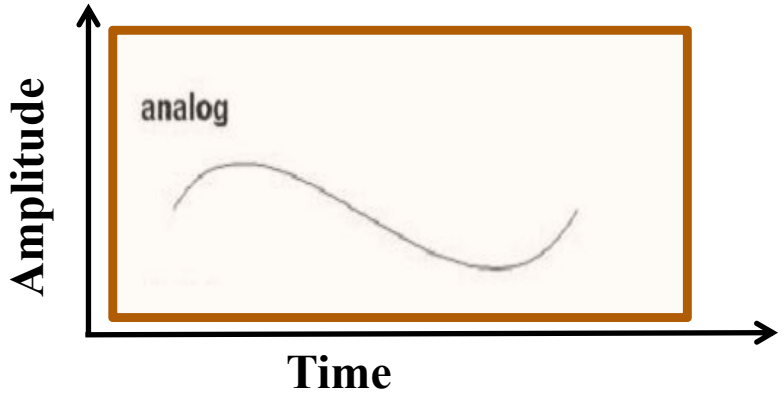
Analog Vs Digital

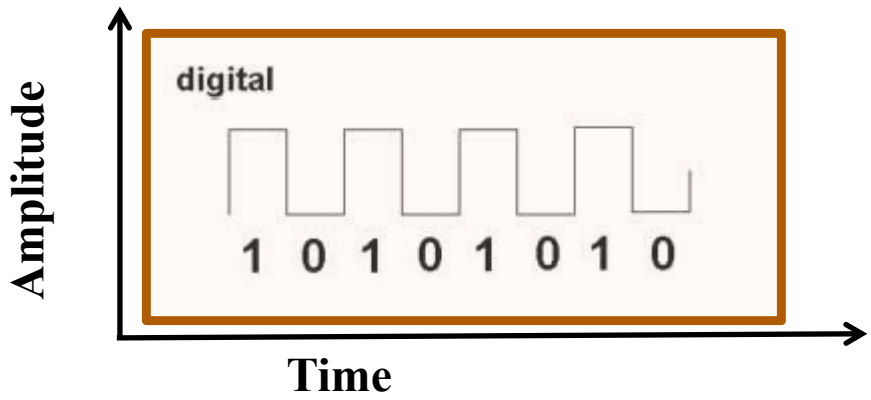
Real world Systems and Processes

- Real-world systems are mostly **continuous (Analog)**
  - Time, acceleration, chemical reactions, etc.
- Mathematics to represent physical systems is continuous
  - Calculus
- Sometimes **discrete (Digital)**
  - No. of students in a class, items in a box, etc
- Mathematics can be discrete for
  - Number theory
  - Counting
  - Approximating physical systems

Analog Vs Digital Signals

- An **analog signal** is a **continuous** wave
  - May **vary** in signal strength (amplitude) or frequency (time)
  - A sine wave
  - Any arbitrary signals can be represented using sine waves
- A **digital signal** is described using
  - Binary (**0s and 1s**)
  - Therefore, cannot take on other fractional values





Quiz 1: Analog or Digital Systems?

- Record players

Analog

- Compact disc (CD) players

Digital (stored data is in Digital)

- Cassette tape

Analog (stored voice is analog)

- Mercury thermometers

Analog

Quiz 2: Analog or Digital Systems?

- Car Speedometer

Analog

- Stethoscope

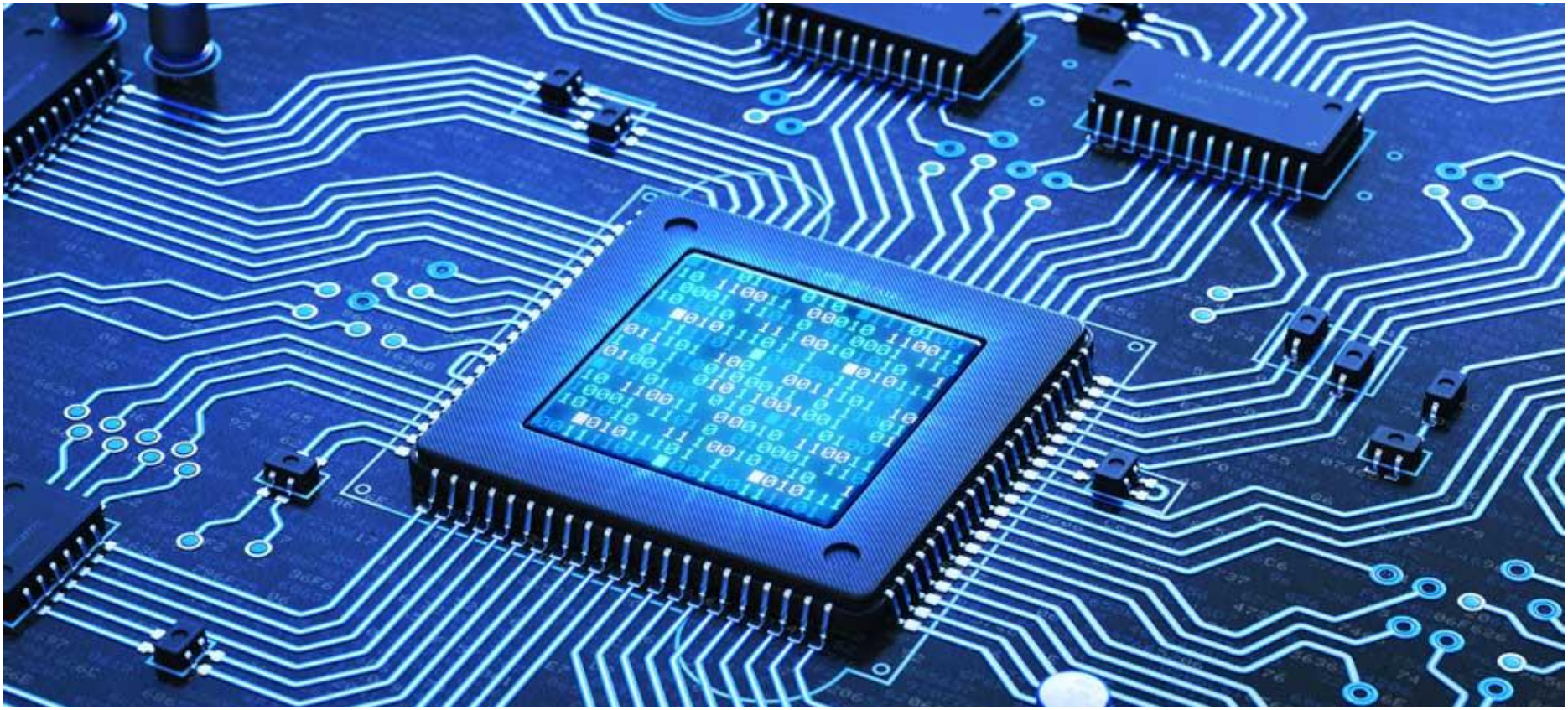
Analog

- Digital Video Disc (DVD) players

Digital

- Computers

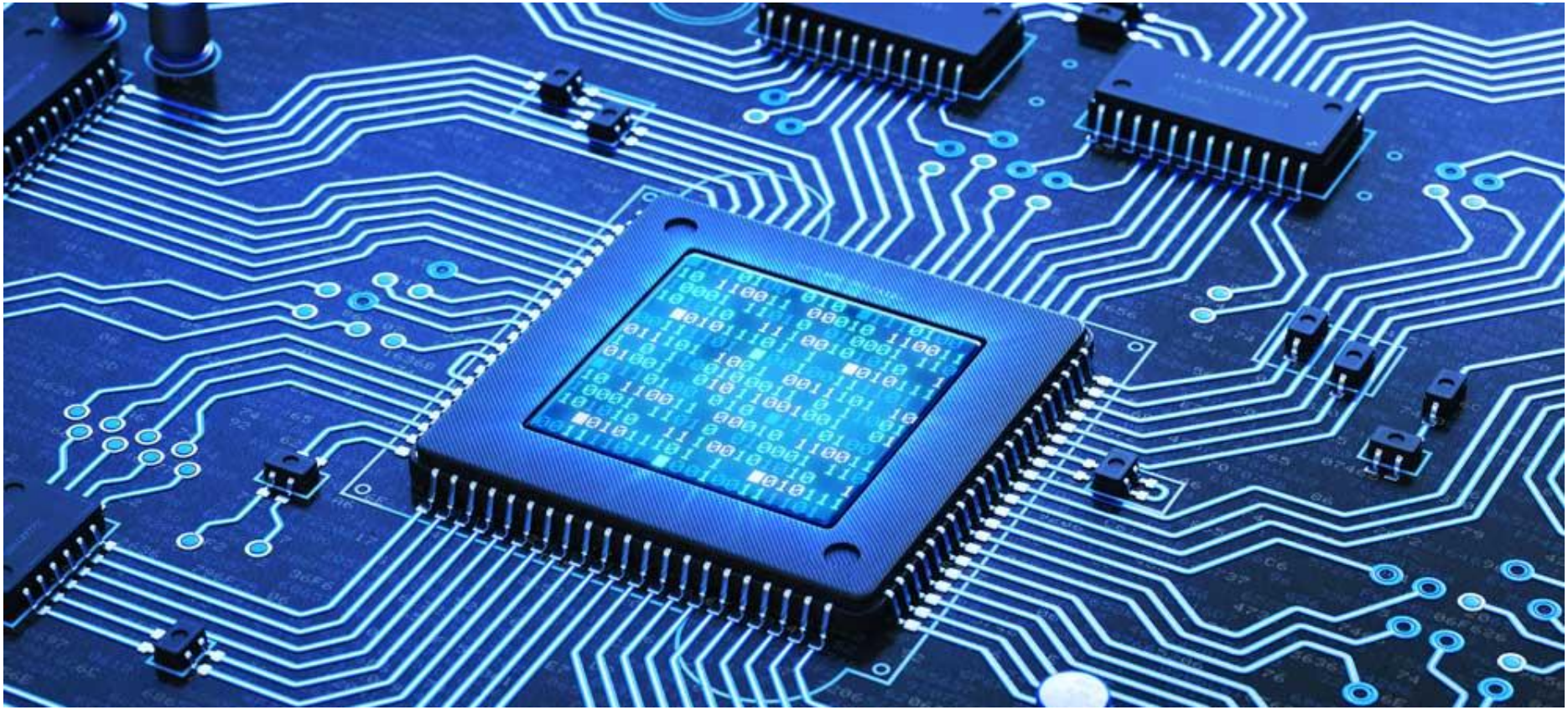
Digital



Why binary in computers?

Why binary in Computers?

- Computers use **binary numbers** because they have **circuits** which can either be in **ON** or **OFF** states
  - That gives them **only two states** to work from
  - To make calculations,
  - To process data, etc.
- The two-digit, or **base 2**, number system is much **easier** for the computers **to process**
  - With the **circuits** they are built with



Processing by Digital Systems

How does Computer process data?

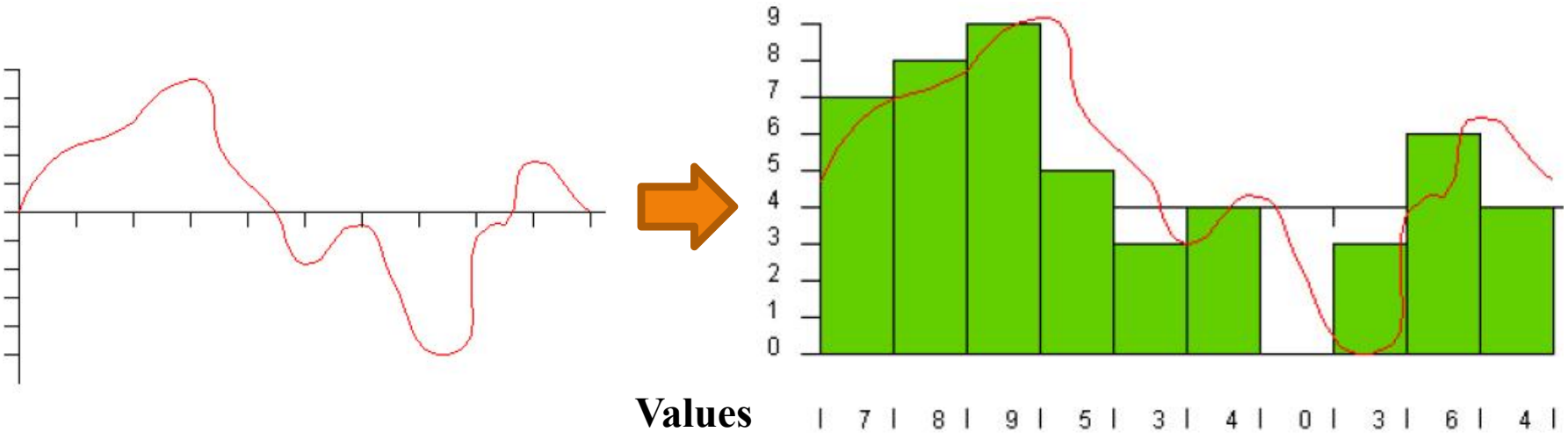
- Computers need digital data which they can understand and process.
- They output processed digital data out.

Input Digital Data

Output Digital Data

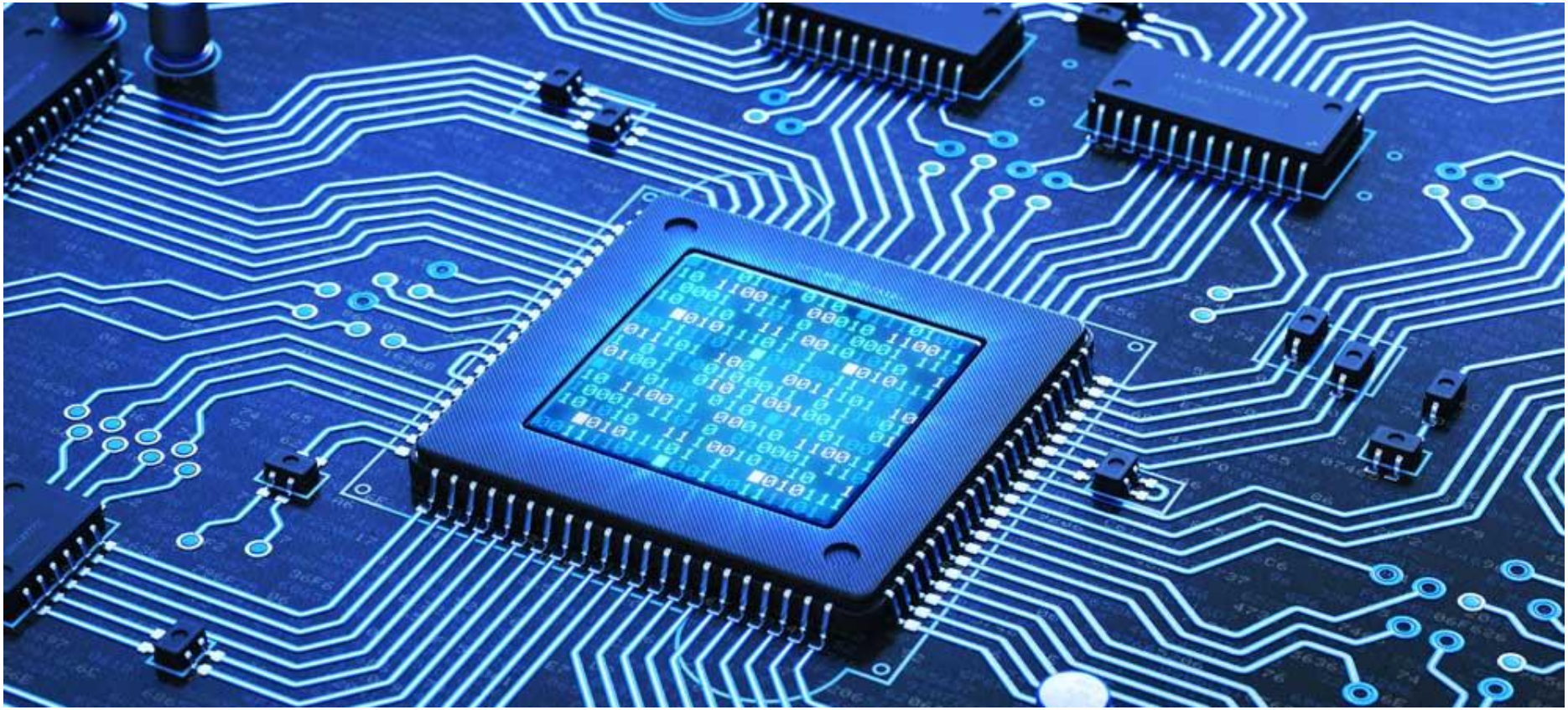
Analog to Digital Conversion

- In the **real world**, most **data** is characterized by **analog signals**
- To manipulate the data using a **microprocessor**
  - **Analog** signals **need** to be **converted to digital** signals, before feeding them into computers for further processing



Analog Signal

Digital Signal



Decimal Number Systems

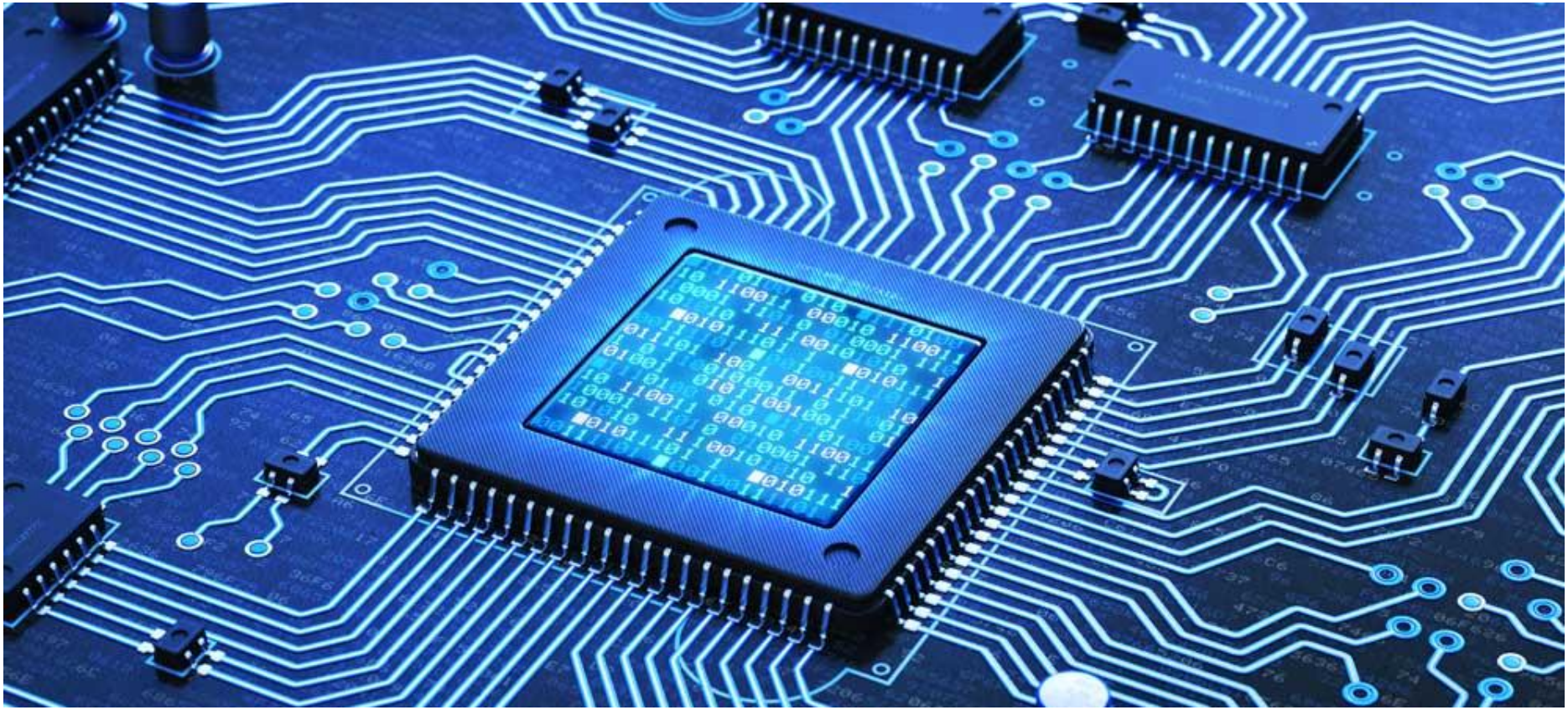
Decimal Number (base 10)

- The decimal number **3586.265** has two parts
  - **Integer** part : **3586**
  - **Fractional** part : **265**
- Represented in the **base 10** format
- **Integer Part:**

$$3586 = 6 \times 10^0 + 8 \times 10^1 + 5 \times 10^2 + 3 \times 10^3$$
$$= 6 + 80 + 500 + 3000 = 3586$$

- **Fractional Part:**

$$265 = 2 \times 10^{-1} + 6 \times 10^{-2} + 5 \times 10^{-3}$$
$$= 0.2 + 0.06 + 0.005 = 0.265$$



Decimal
To
Binary, Octal and Hexadecimal

Binary Weights

POSITIVE POWERS OF TWO (WHOLE NUMBERS)									NEGATIVE POWERS OF TWO (FRACTIONAL NUMBER)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2	1/4	1/8	1/16	1/32	1/64
									0.5	0.25	0.125	0.0625	0.03125	0.015625

16

Decimal to Binary (base 2) Conversion

- **Convert** the **decimal** number $\mathbf{9.3125_{10}}$ to **Binary**
- **Binary** is represented in the **base 2** format
  - It **means** that there can only be **2 literals** : **0** and **1**
- **Integer Part:**

$$9 = 1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3$$
$$= 1 + 0 + 0 + 8 = 9$$

- **Fractional Part:**

$$3125 = 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$$
$$= 0 + 0.25 + 0 + 0.0625 = 3125$$

$$\mathbf{9.3125_{10} = 1001.0101_2}$$

Decimal to Binary Conversion
Repeated Division

- **Convert** the **decimal** number **13.375_{10}** to **Binary**
- **Integer Part: 13**

Divisor	Dividend	Remainder
2	13	—
2	6	1
2	3	0
2	1	1
—	0	1

- **Fractional Part: 37**

$0.375 \times 2 = 0.75$ with a carry of 0
$0.75 \times 2 = 0.5$ with a carry of 1
$0.5 \times 2 = 0$ with a carry of 1
The binary equivalent of $(0.375)_{10} = (.011)_2$

$13.375_{10} = 1101.011_2$

Decimal to Octal: Procedure

- **Convert** the **decimal** number $\mathbf{73.75_{10}}$ to **Octal**
- **Integer Part: 73**

Divisor	Dividend	Remainder
8	73	—
8	9	1
8	1	1
—	0	1

- **Fractional Part: 75**

$0.75 \times 8 = 0$ with a carry of 6

The octal equivalent of $(0.75)_{10} = (.6)_8$

$$\mathbf{73.75_{10} = 111.6_8}$$

Decimal to Hexadecimal: Procedure

- **Convert** the **decimal** number **82.25_{10}** to **Hexadecimal**
- **Integer Part: 82**

Divisor	Dividend	Remainder
16	82	—
16	5	2
—	0	5

- **Fractional Part: 25**

$0.25 \times 16 = 0$ with a carry of 4

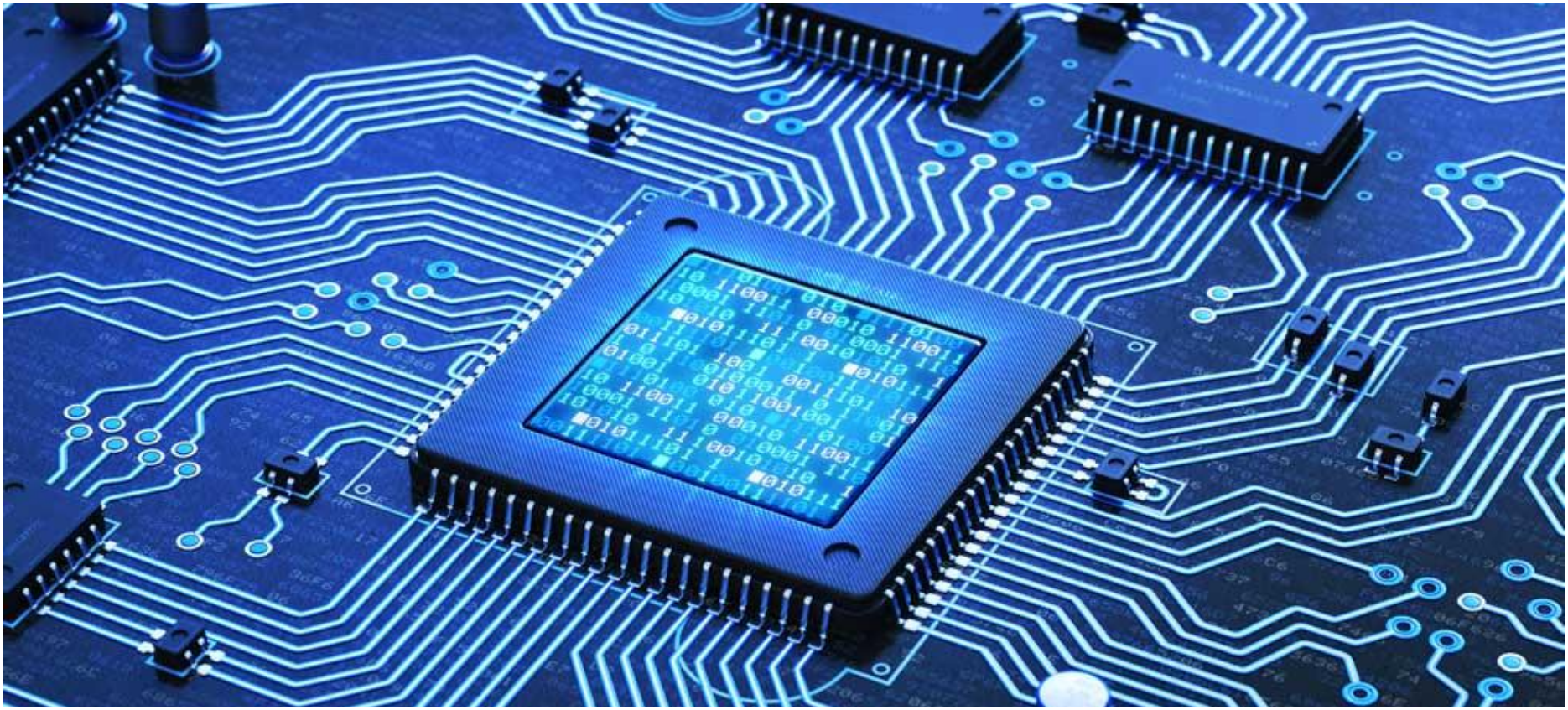
$82.25_{10} = 52.4_{16}$

Binary, Octal and Hex Table

Decimal	Binary	Octal	Hexadecimal
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

Binary, Octal and Hex … the same slide

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	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



Octal to Binary
and
Binary to Octal

Octal to Binary: Procedure

- **Convert** the **Octal** number **374.26_8** to **Binary**
- **Binary** equivalent, in a **group** of **three bits**

$$(011\ 111\ 100.010\ 110)_2:$$

- **Can be expressed as:** $(011111100.010110)_2$
- Omit the leftmost zeros of the integer part and the rightmost zeros of fractional part
- **The result:**

$$\mathbf{374.26_8 = 11111100.01011_2}$$

Binary to Octal: Procedure

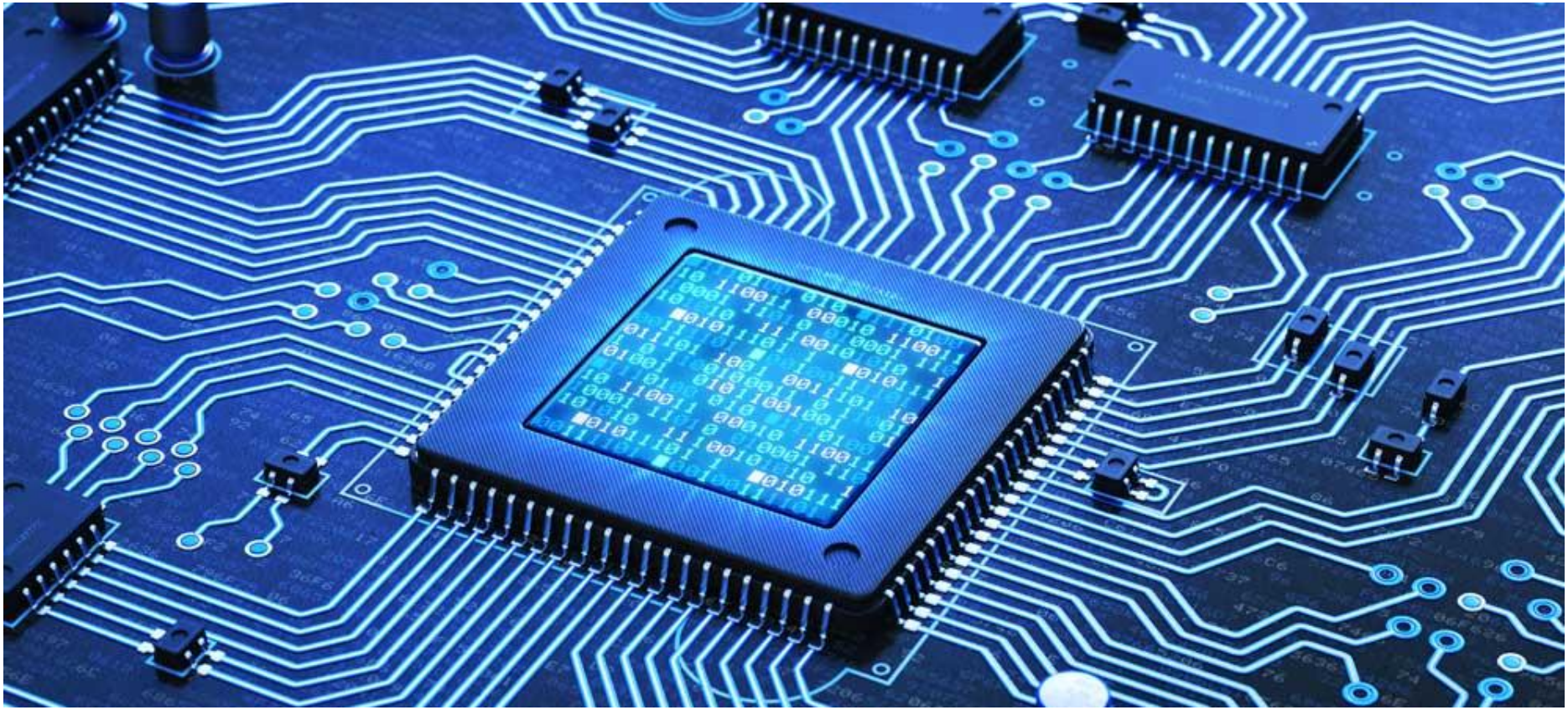
- **Convert** the **binary** 1110100.0100111_2 to **Octal**
- Group the bits into three starting from the decimal point on both directions

$$(1\ 110\ 100.010\ 011\ 1)_2$$

- **Express it in a group of three binary digits:**
  - Add both leading and trailing zeros if needed

$$(001\ 110\ 100.010\ 011\ 100)_2$$

$$1110100.0100111_2 = 164.234_8$$



Hex to Binary
and
Binary to Hex

Hex to Binary: Procedure

- **Convert** the **Hex** number **$2F.C4_{16}$** to **Binary**
- **Binary** equivalent, in a **group** of **four bits**

$$(0010\ 1111.1100\ 0100)_2$$

- **Can be expressed as:** $(00101111.11000100)_2$
- Omit the leftmost zeros of the integer part and the rightmost zeros of fractional part
- **The result:**

$$\mathbf{2F.C4_{16} = 101111.110001_2}$$

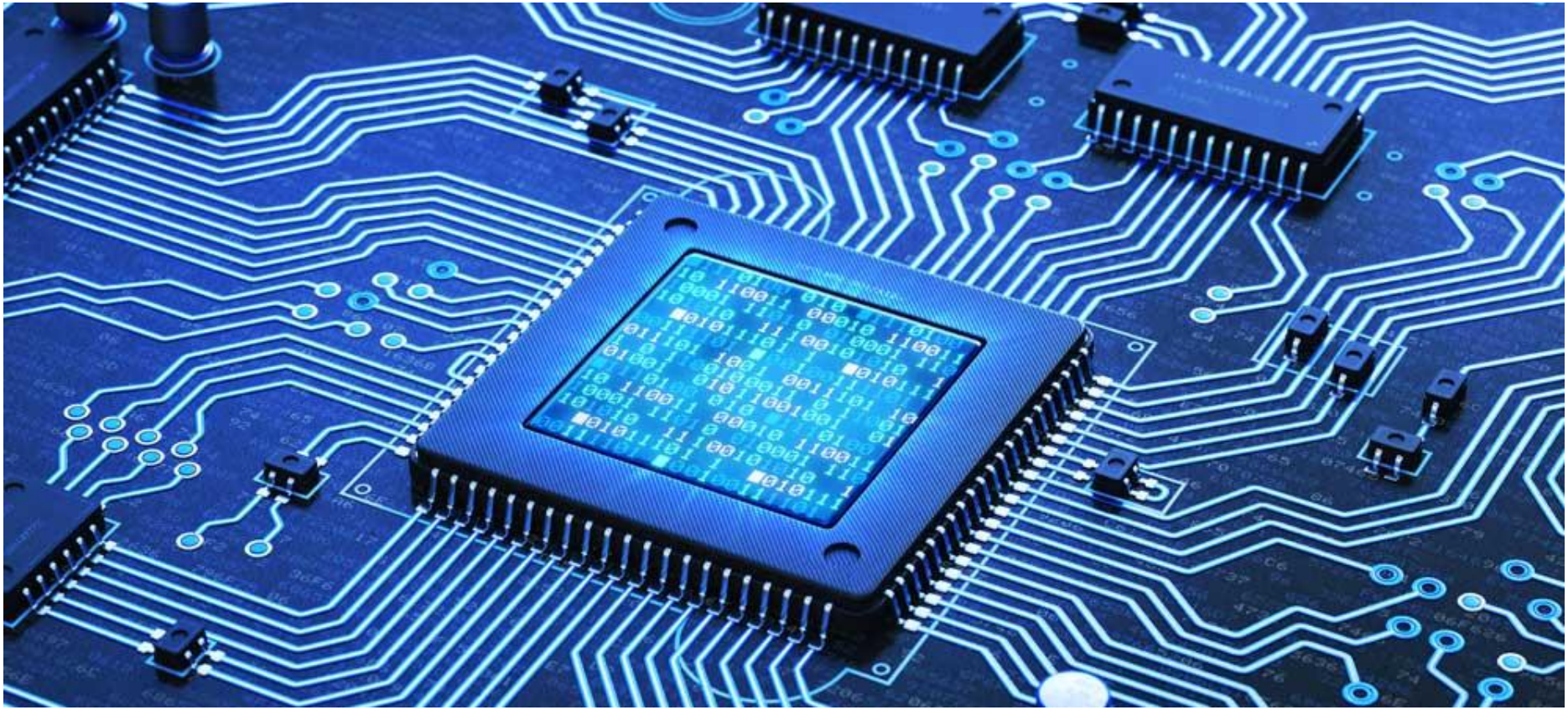
Binary to Hex: Procedure

- **Convert** the **binary** $\mathbf{1011001110.011011101_2}$ to **Hex**
- Group the bits into four starting from the decimal point on both directions

$$(10\ 1100\ 1110.0110\ 1110\ 1)_2$$

$$(0010\ 1100\ 1110.0110\ 1110\ 1000)_2$$

$$\mathbf{1011001110.011011101_2 = 2CE.6E8_{16}}$$



Hex to Octal
and
Octal to Hex

Hex to Octal: Procedure

- **Convert** the **Hex** number **$2F.C4_{16}$** to **Octal**
- Write the **Binary** equivalent, in a **group** of **four bits**

$$(0010\ 1111.1100\ 0100)_2$$

- Group them into three bits starting from the decimal point on both directions:

$$(101\ 111.110\ 001)_2$$

- **The result:**

$$\mathbf{2F.C4_{16} = 57.61_8}$$

Octal to Hex: Procedure

- Convert the Octal 762.013_8 to Hex
- Write the binary equivalent in a group of three bits, starting from the decimal point on both directions

$$(111\ 110\ 010.000\ 001\ 011)_2$$

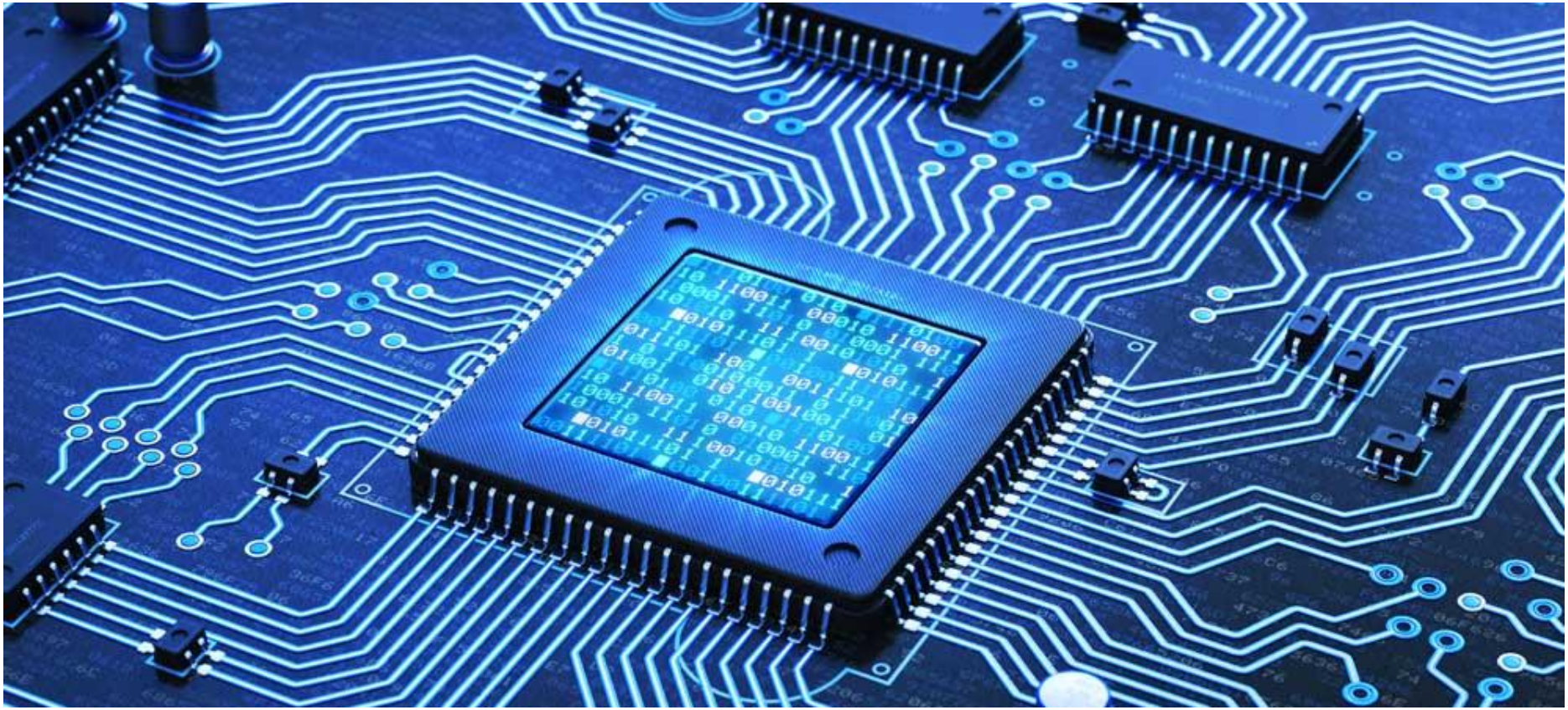
- After combining the bits:

$$(111110010.000001011)_2$$

- Re-group them into four bits:

$$(0001\ 1111\ 0010.0000\ 0101\ 1000)_2$$

$762.013_8 = 1F2.058_{16}$



Binary Codes

What are Binary Codes and Why are they needed?

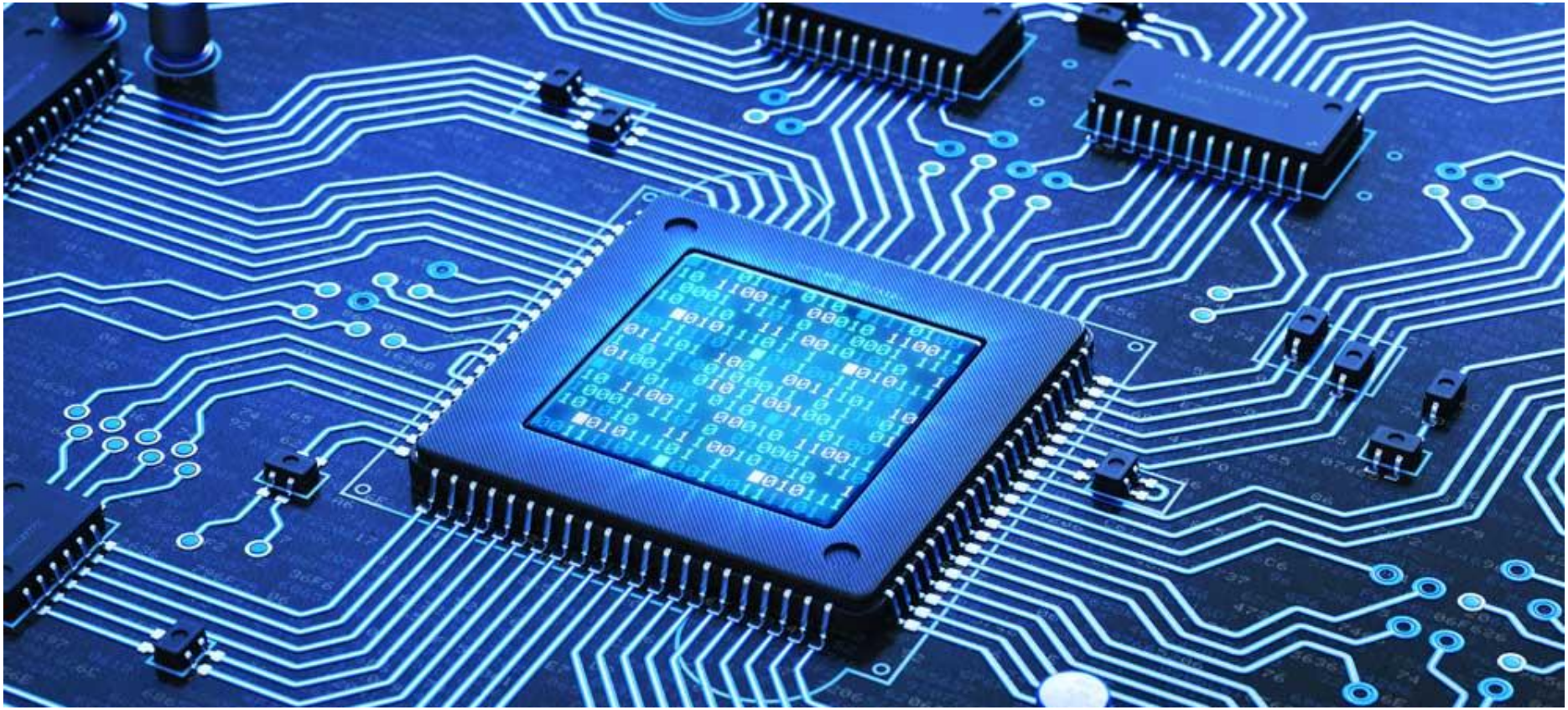
- Digital systems **represent** and **manipulate** not only binary numbers, but **also**
  - Many other **discrete** elements of **information**.
  - **Example**: Digital speech signal, Character symbols (a, b, etc.)
- Any discrete element of information that is distinct among a group of quantities can be represented
  - With binary code, i.e., a pattern of 0's and 1's
- The digital data is represented, stored and transmitted as group of binary bits
- A set of eight elements requires a three-bit code and a set of 16 elements requires a four-bit code

Binary Codes

- Both letters, numbers, symbols are represented by binary codes
  - Various *fonts* used in **word processors** are examples of **binary codes**
- Binary codes are used in **computer applications** and digital data **communication**
- Binary codes ease implementation of digital circuits to process encoded digital data

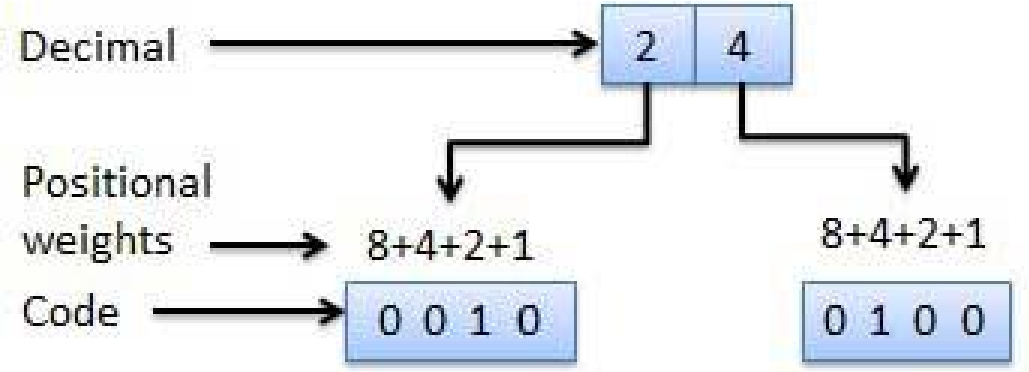
Classification of Binary Codes

- Weighted Codes
- Non-Weighted Codes
- Binary Coded Decimal Code
- Alphanumeric Codes
- Error Detecting Codes – Not covered in this course
- Error Correcting Codes – Not covered in this course



Weighted Codes
Binary Coded Decimal (BCD)

Weighted Codes



- Weighted binary codes are those binary codes which obey the **positional weight principle**.
- Each position of the number represents a specific weight.
- Several systems of the **codes** are used to **express** the **decimal digits 0 through 9**.
- In these codes each **decimal digit** is **represented** by a group of **four bits**

Courtesy: Tutorial Point

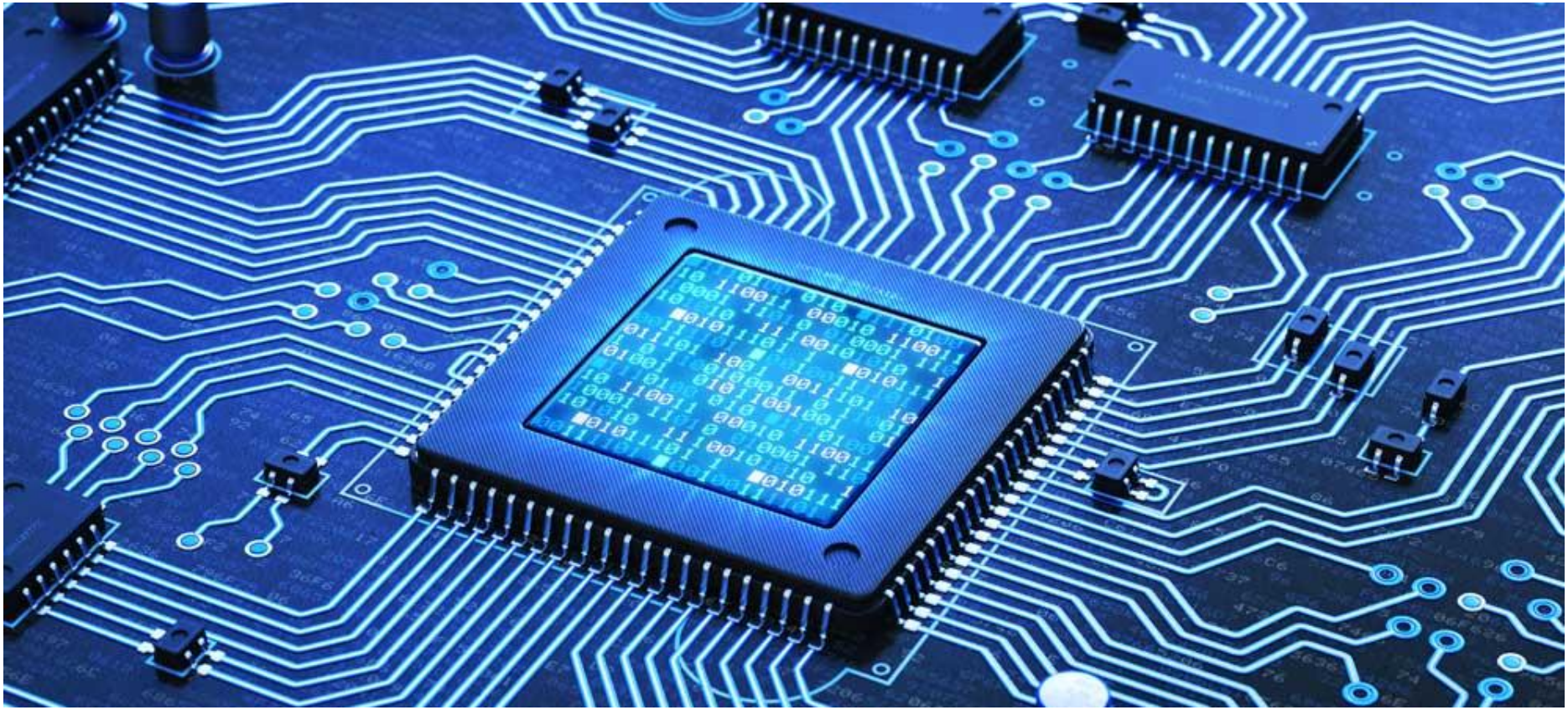
Binary Coded Decimal (BCD)

- Only the **first 9** combinations of four bits are **valid**
- The **remaining 6** combinations are **unused** in **BCD**
- **Easier** to represent decimal digits in **BCD** which is **straightforward**
- **Example: $(12.75)_{10}$**
- **BCD (8421)**:
- **$(0001\ 0010.0111\ 0101)_{8421}$**
- **$(0001\ 0010.1101\ 1011)_{2421}$**

Decimal Digit	BCD 8421	2421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	1011
6	0110	1100
7	0111	1101
8	1000	1110
9	1001	1111
Unused bit combi-nations	1010	0101
	1011	0110
	1100	0111
	1101	1000
	1110	1001
	1111	1010

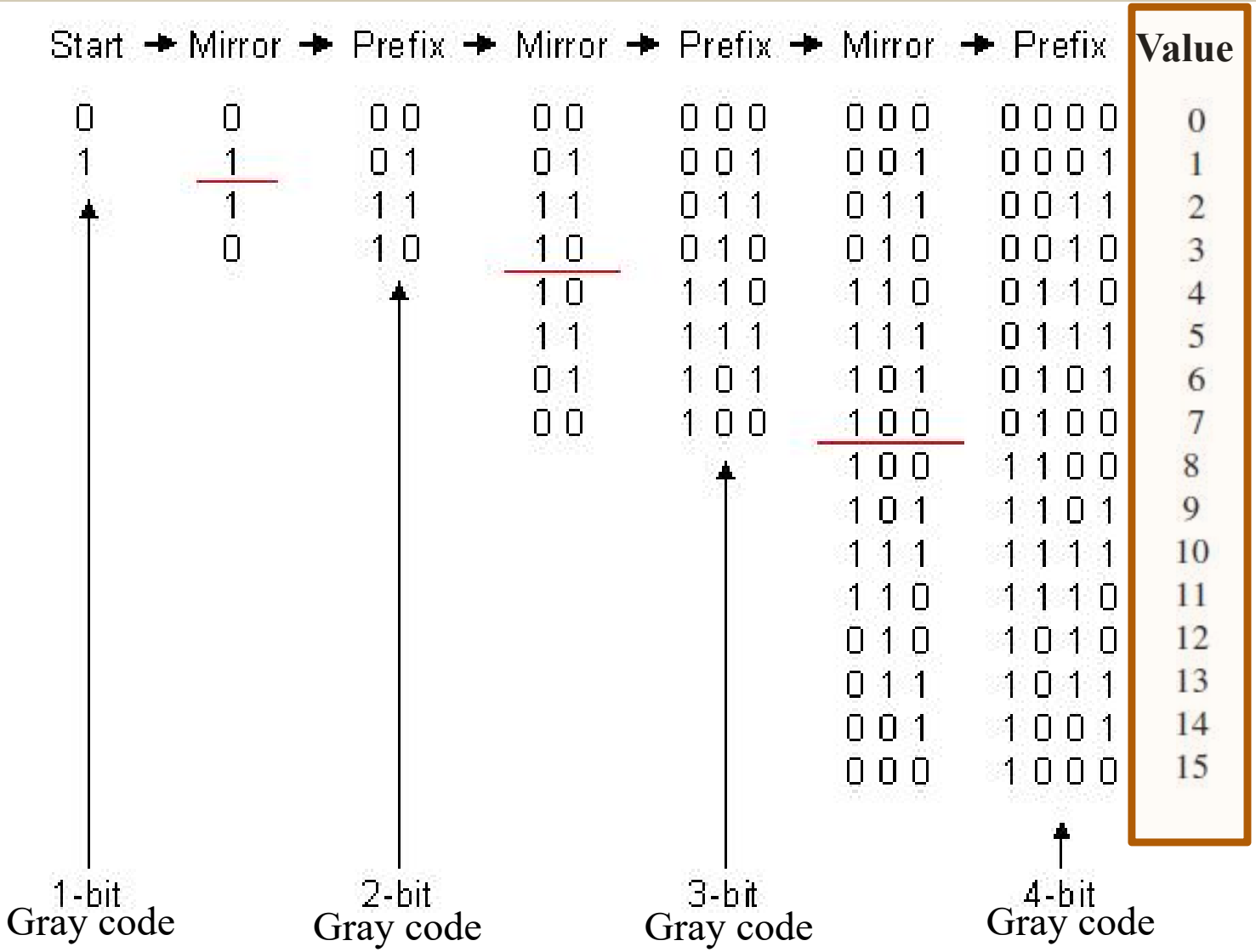
Other BCDs

Decimal	8421 BCD code	4221 BCD code	5421 BCD code
0	0000	0000	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	1000	0100
5	0101	0111	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100



Non-weighted Codes: Gray Codes

Constructing Gray Codes



Gray Codes

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

Gray Codes

- It is **non-weighted code** and it is not arithmetic code.
  - i.e., Gray code **cannot** be **used** for **arithmetic operations**
- There are no specific weights assigned to the bit positions in Gray codes
- It has a very special feature that, **only one bit** will **change** each time the **decimal number** is **incremented**
- As only one bit changes at a time, the gray code is called as a **unit distance code**
- The gray code is a **cyclic code**.
  - **Circular** shifts of each codeword gives another word that belongs to the **code**

Use of Gray Codes

- It is a **non-weighted code** which belongs to a class of codes called **minimum change codes**
- Here, **two adjacent** code **numbers differs** from each other by **only one bit**
- Gray code is popularly used in the **shaft position encoders**.
  - A shaft position encoder produces a code word which represents the **angular position** of the **shaft**
- It is also used in the transmission of digital signals
- The Gray code is used for **labelling** the **axes** of **Karnaugh maps**

Optical Encoder

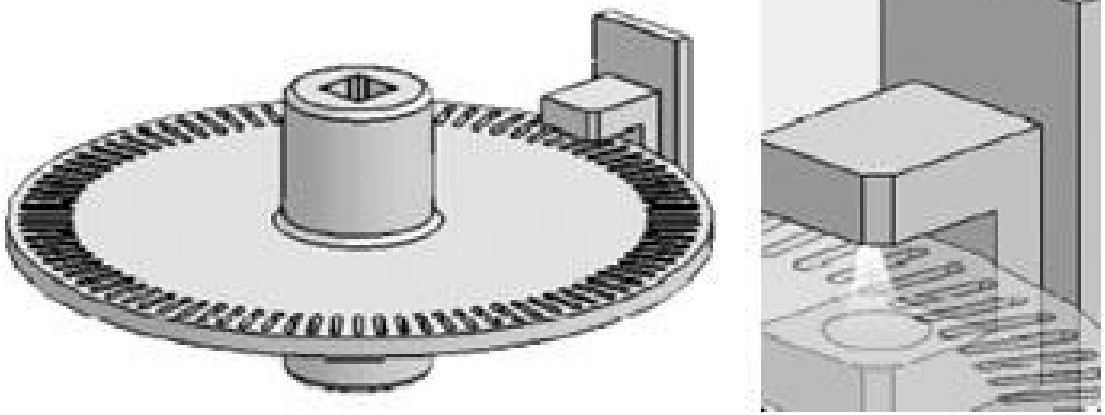
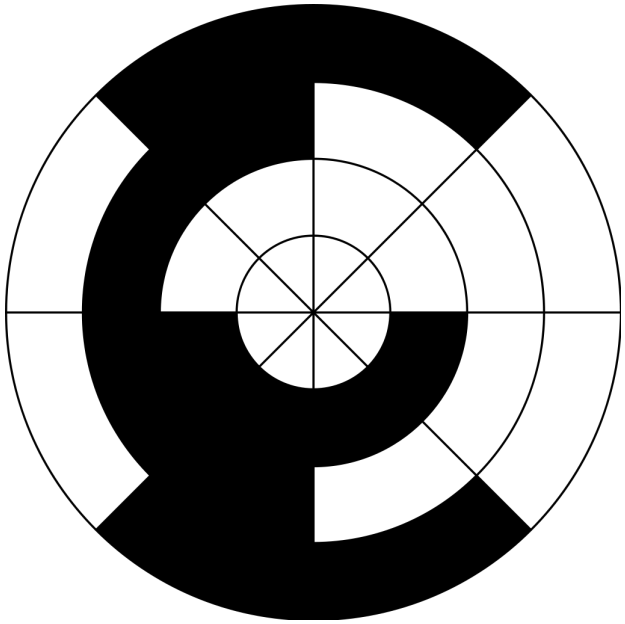
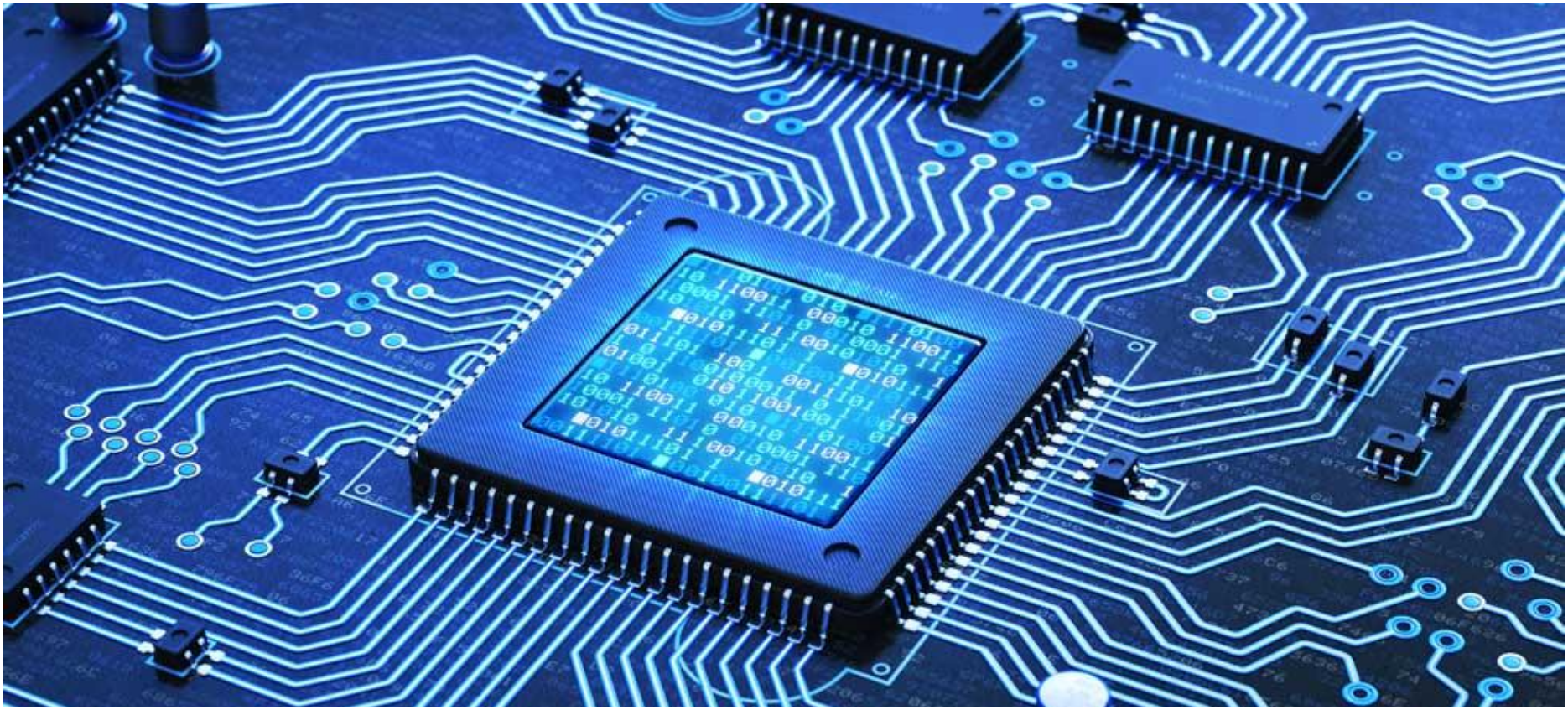


Figure 2. Optical shaft encoder disk



Coded Disc Connected to a Rotating Shaft



Alphanumeric Codes (ASCII)

ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

- **ASCII:** American Standard Code for Information Interchange

ASCII Codes

- The alphanumeric codes are the codes that represent **numbers** and **alphabetic characters**
- **ASCII** is a **7-bit** code
- Extended Binary Coded Decimal Interchange Code (**EBCDIC**), is an **8-bit c**ode
- With the limited support that an 8 bit code can provide to all the languages in the world, Unicode is defined in 1987
- **Unicode (UTF-16 and UTF-32)** are **16 bit and 32 bits** later versions, used for supporting various languages

Session 2.1: Summary

- Analog Vs Digital
- Processing by Digital systems
- Number systems
- Binary, octal, hexadecimal
- Conversions from one to the other
- Binary codes and their classifications
- Weighted codes
  - Binary Coded Decimal (BCD)
- Non-weighted codes
  - Gray code
  - Optical Encoder Example using Gray Code
- ASCII codes