CLASS-XI

Study Notes

Data Representation

Period-1

Introduction:

Learning Outcomes

- Ability to understand and apply basic computational thinking.
- Ability to understand the notion of data types and data structures and apply in different situations.
- Ability to appreciate the notion of an algorithm and apply its structure including how algorithms handle corner cases.
- Ability to develop a basic understanding of computer systems architecture, operating system, mobile and cloud computing.
- Ability to work in the cyber world with understanding of cyber ethics, cyber safety and cybercrime
- Ability to make use the value of technology in societies, gender and disability issues and the technology behind biometric ids.

General Concepts :

In order to work with data, the data must be represented inside the computer. Digital computers represent data by means of an easily identified symbol called a digit.

Digital technology have found their way into innumerable areas of technology and the far most reaching is digital computers.

In digital systems like computer ,the quantities are measured by symbol called digits. They occur in various forms like binary, octal, Decimal and hexadecimal.

DATA - The raw facts available before processing
INSTRUCTION – The operation which is going to be done over data
INFORMATION – Final Result

Digital Number System

Each number system has a base also called a Radix.

A decimal number system is a system of base 10; (0,1,2,3,. . . . . .. 9)

binary is a system of base 2; (0, 1)

octal is a system of base 8; (0,1,2, ...7)

and hexadecimal is a system of base 16. (0,1,2. . . .9 and 10, 11, 12,. . . . ,15

What are these varying bases? i,e. A, B, C, ... , F)

The answer lies in what happens when we count up to the maximum number that the numbering system allows. In base 10, we can count from 0 to 9, that is,10 digits.

Number System at a Glance

Number System	Base	Symbols used
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 where A = 10; B = 11; C = 12; D = 13; E = 14; F = 15

Example:

Binary – $(100010)_2$ Octal – $(136)_7$

Decimal – $(435)_{10}$ Hexadecimal – $(1D5)_{16}$

Least significant digit & Most Significant digit

Least Significant Digit:

Sometimes abbreviated as LSD, the least significant digit is the lowest digit in a number, located at the far right of a string.

For example, in the number 2006, the "6" is the least significant digit.

Most Significant Digit:

Sometimes abbreviated as MSD, the digit to the far left in a string of digits.

For example, in the number 2006, the "2" is the most significant digit.

Period-2

Number System Conversion

Decimal to Binary:

Method to convert a Decimal number into its Binary equivalent:

1. Divide the decimal number by 2.
2. Take the remainder and record it on the side.
3. Divide the quotient by 2.
4. REPEAT UNTIL the decimal number cannot be divided further.
5. Record the remainders in reverse order and you get the resultant binary number.

Example

Convert the Decimal number 125 into its Binary equivalent.

125 / 2 = 62	1
62 / 2 = 31	0
31 / 2 = 15	1
15 / 2 = 7	1
7 / 2 = 3	1
3 / 2 = 1	1
1 / 2 = 0	1

Answer: $(1111101)_2$

Converting Decimal to Octal

The method to convert a decimal number into its octal equivalent:

1. Divide the decimal number by 8.
2. Take the remainder and record it on the side.
3. Divide the quotient by 8.
4. REPEAT UNTIL the decimal number cannot be divided further.
5. Record the remainders in reverse order and you get the resultant binary

Example

Convert the Decimal number 125 into its Octal equivalent.

125 / 8 = 15 5
15/ 8 = 1 7
1/8 =0 1

Answer: $(175)_8$

Converting Decimal to Hexadecimal

Method to convert a Decimal number into its Hexadecimal equivalent:

1. Divide the decimal number by 16.
2. Take the remainder and record it on the side.
3. REPEAT UNTIL the decimal number cannot be divided further.
4. Record the remainders in reverse order and you get the equivalent hexadecimal number.

Example

Convert the Decimal number 300 into its hexadecimal equivalent.

300 / 16 = 18 12-(C)
18 / 16 = 1 2
1 / 16 = 0 1

Answer: $(12C)_{16}$

Practice session

Do as directed:

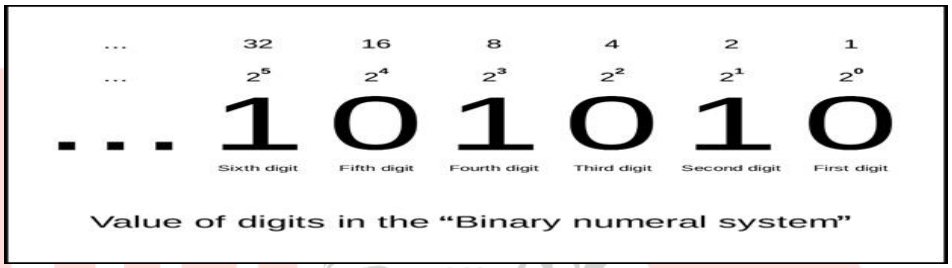
1. Convert the Decimal number 781 to its Binary equivalent.
2. Convert Decimal number 101101 to its Binary equivalent
3. Convert Decimal number 321 into its Binary equivalent
4. Convert the Decimal number 324 into its Binary equivalent
5. Convert the Decimal number 213 to its Hexadecimal equivalent
6. Convert the Decimal number 345 into Octal number.

7. Convert the Decimal number 736 into Hexadecimal number.
8. Convert the Decimal number 246 into Hexadecimal number.
9. Convert the Decimal number 29 into Octal number.
10. Convert the Decimal number 576 to octal.
11. Convert the Decimal number 59 to hexadecimal.

Period-3

Binary, Octal & Hexadecimal to Decimal

Binary to Decimal:



Converting a number from one Base to another:

Method to convert Binary to Decimal:

1. Start at the rightmost bit.
2. Take that bit and multiply by 2^n where n is the current position beginning at 0 and increasing by 1 each time. This represents a power of two.
3. Sum each terms of product until all bits have been used

Example: Convert the Binary number 101011 to its Decimal equivalent.

$$1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$$

$$32 + 0 + 8 + 0 + 2 + 1 = (43)_{10}$$

Converting Binary Fraction to Decimal

Example: Convert the Binary number 11011.1101 to its Decimal equivalent.

Position of digits

. . . . . 3 2 1 0 . -1 -2 -3 -4 . . . . . .

2^4 2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2} 2^{-3}

1 1 0 1 1 1 0 1

$= (1 \times 2^4)+ (1 \times 2^3)+ (0 \times 2^2)+ (1 \times 2^1)+ (1 \times 2^0)+ (1 \times 2^{-1})+ (0 \times 2^{-2})+ (1 \times 2^{-3})$

$= 16+8+0+2+1+0.5+0+0.125$

$= (27.625)_{10}$

Converting Octal to Decimal

Method to convert Octal to Decimal:

1. Start at the rightmost bit.
2. Take that bit and multiply by 8^n, where n is the current position beginning at 0 and increasing by 1 each time. This represents the power of 8.
3. Sum each of the product terms until all bits have been used.

Example

Convert the Octal number 321 to its Decimal equivalent.

$3 * 8^2 + 2 * 8^1 + 1 * 8^0$

$192+16+ 1 = (209)_{10}$

Converting Octal fraction to Decimal

Example

Convert the Octal number 23.25 to its Decimal equivalent.

8^1 8^0 . 8^{-1} 8^{-2}

2 3 2 5

$= (2 \times 8^1) + (3 \times 8^0) + (2 \times 8^{-1}) + (5 \times 8^{-2})$

$= 16+3+0.25+0.07812$

$= (19.32812)_{10}$

Converting Hexadecimal to Decimal

Method to convert Hexadecimal to Decimal:

1. Start at the rightmost bit.

2. Take that bit and multiply by 16^n ,where n is the current position beginning at 0 and increasing by 1 each time. This represents a power of 16.

3. Sum each terms of product until all bits have been used.

Example

Convert the Octal number AB to its Decimal equivalent.

$= A * 16^1 + B * 16^0$

$= 10 * 16^1 + 11 * 16^0$

$= 160+11 = (171)_{16}$

Converting Hexadecimal Fraction to Decimal

Example: Convert the Hexadecimal number IE.8C to its Decimal equivalent.

16^1	16^0	.	16^{-1}	16^{-2}
1	E		8	C

$= (1 \times 16^1) + (14 \times 16^0) + (8 \times 16^{-1}) + (12 \times 16^{-2})$

$= 16+14+0.5+0.04688$

$= (30.54688)_{10}$

Binary to Decimal Conversion using Dibble-Double method

A second technique for the conversion of a binary integer into its equivalent decimal is commonly called the double-dabble method. There are three basic steps to this technique:

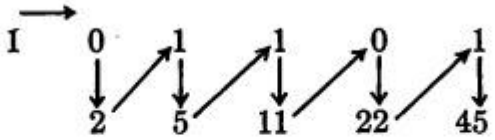
1. Multiply the higher-order binary bit by 2. Add this product to the next highest order bit. Record this total.
2. Multiply the total obtained (in step 1) by 2. Add the new product to the next highest order bit. Record this sum.
3. Continue this process until all the bits have been worked. The final total is the decimal equivalent of the original binary quantity.

Example 1. Convert the binary quantity 101101 to its decimal equivalent by use of the double-dabble method.

Solution. Following the procedure outlined above results in converting the binary quantity into its decimal equivalent in five steps. It should be noted that the number of steps involved in the conversion process is equal to one less than the number of bits in the original binary quantity.

Binary to Decimal Conversion using Dibble-Double method

Since the binary quantity 101101 has six bits, then its conversion by the double-dabble method requires five steps, as shown:



Taking it step by step,

Step 1. $(1 \times 2) + 0 = 2$
Step 2. $(2 \times 2) + 1 = 5$
Step 3. $(5 \times 2) + 1 = 11$
Step 4. $(11 \times 2) + 0 = 22$
Step 5. $(22 \times 2) + 1 = 45$

1 0 1 1 0 1
2 5 11 22 45

Therefore,

Binary 101101 = decimal 45

Practice Session :

- Y Convert the Decimal number 781 to its Binary equivalent.
- Y Convert Decimal number 101101 to its Binary equivalent
- Y Convert Decimal number 321 into its Binary equivalent
- Y Convert the Decimal number 324 into its Binary equivalent
- Y Convert the Decimal number 213 to its Hexadecimal equivalent
- Y Convert the Decimal number 345 into Octal number.
- Y Convert the Decimal number 736 into Hexadecimal number.
- Y Convert the Decimal number 246 into Hexadecimal number.
- Y Convert the Decimal number 29 into Octal number.
- Y Convert the Decimal number 576 to octal.
- Y Convert the Decimal number 59 to hexadecimal.

Period-4

Number System Conversion

Binary to Octal & Hexadecimal AND Octal & Hexadecimal to Binary:

Octal ➔ Binary Representation

Each octal digit can be represented by a 3-bit binary number as shown below:

Octal Digits	3-bit Binary number
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Octal ➔ Binary Conversion

- Conversion from octal to binary is very straightforward. Each octal digit is replaced by 3-bit binary number.

For example, (472)8 = (100 111 010)2.

```
 4     7     2          Octal number
 ↓     ↓     ↓
100   111   001         Binary number
```

- A binary number is converted into an octal number by taking groups of 3 bits, starting from LSB, and replacing them with an octal digit.

For example, (11 010 110)2 = (326)8.

```
011   010   110         Binary number
 ↓     ↓     ↓
 3     2     6          Octal number
```

Hexadecimal ➔ Binary Conversion

- Each hex digit can be represented by a 4-bit binary number as shown above. Conversion from hex to binary is very straightforward. Each hex digit is replaced by 4-bit binary number.

For example, $(1A7)_{16}$ - $(1\ 1010\ 0111)_2$

1	A	7
↓	↓	↓
1	1010	0111

- A binary number is converted into an octal number by taking groups of 4 bits, starting from LSB, and replacing them with a hex digit. For example, 110101102 = 3268.

For example, $(101111110101 10)_2$ = $(2FD6)_{16}$.

10	1111	1101	0110
↓	↓	↓	↓
2	F	D	6

Assignments

1. Convert decimal 234_{10} to

   a. binary

   b. octal

   b. hexadecimal

2. Convert binary 1001011101_2 to

   a. octal

   b. hexadecimal

3. Convert hexadecimal $ABF2_{16}$ to

   a. decimal

   b. binary

   c. octal

4. Convert 10010100100110001_2 to

   a. decimal

   b. octal

   c. hexadecimal

5. Convert number octal 526_8 to

   a. decimal

b. hexadecimal

c. binary.

6. Convert the following number to the indicated base/code.

a) 11101.11_2 to decimal.

b) 01101001_2 to octal.

c) 754_8 to Binary

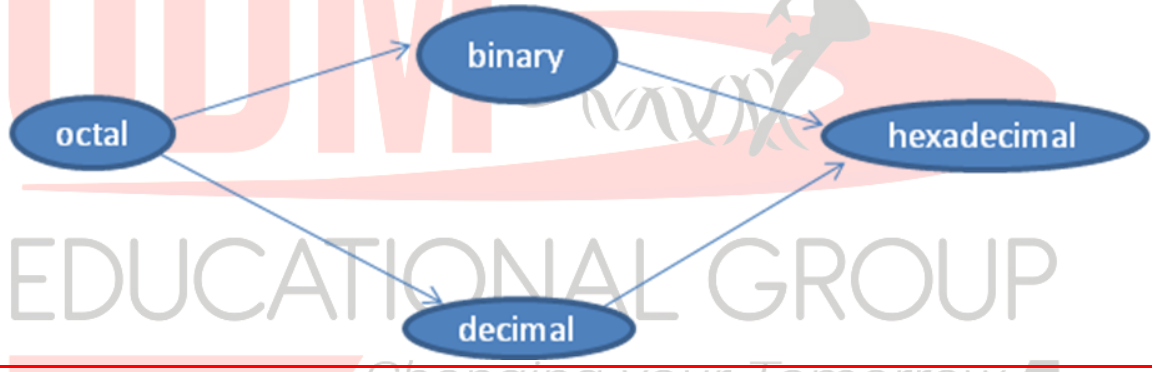
d) 152.25_{10} to hexadecimal.

Period-5

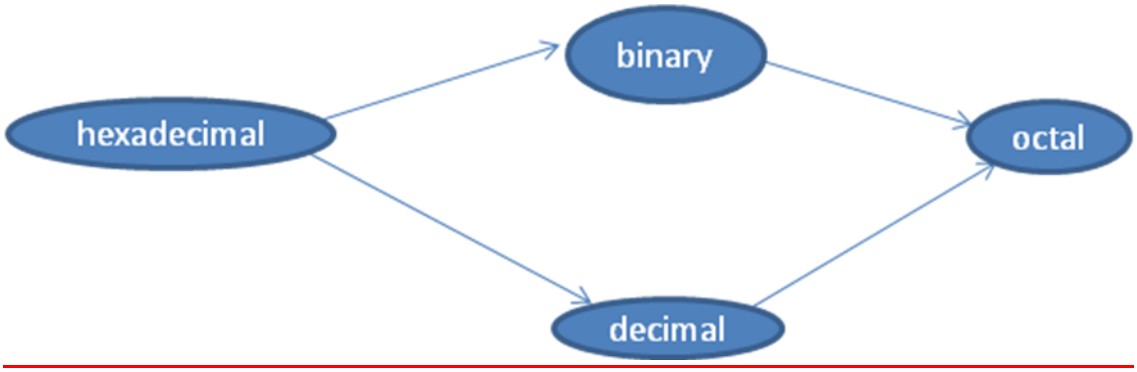
Number System Conversion

Octal to Hexadecimal AND Hexadecimal to Octal etc.:

Convert octal $(234)_8$ to Hexadecimal



Convert Hexadecimal $(AF4)_{16}$ to octal



1. Convert decimal 234_{10} to

a. binary

b. BCD

b. octal

c. hexadecimal

2. Convert binary 1001011101_2 to

a. decimal

b. octal

c. hexadecimal

d. BCD

3. Convert hexadecimal $ABF2_{16}$ to

a. decimal

b. binary

c. octal

d. BCD

4. Convert BCD 10010100100110001_{BCD} to

a. decimal

b. octal

c. hexadecimal

d. binary

5. Convert number octal 526_8 to

a. decimal

b. BCD

c. hexadecimal

d. binary

6. Convert the following number to the indicated base/code.

a) 11101.11_2 to decimal.

b) $FED.47_{16}$ to octal.

c) 01101001_2 to binary.

d) 754_8 to Binary

e) 152.25_{10} to hexadecimal.

Period-6

Character Representation in Memory

As represented in introduction, all the input data given to the computer should be in understandable format. In general, 26 uppercase letters, 26 lowercase letters, 0 to 9 digits and special characters are used in a computer, which is called character set. All these character set are denoted through numbers only. All Characters in the character set needs a common encoding system. There are several encoding systems used for computer. They are

- ASCII – American Standard Code for Information Interchange
- Unicode
- ISCII - Indian Standard Code for Information Interchange

American Standard Code for Information Interchange (ASCII)

is the most popular encoding system recognized by United States. Most of the computers use this system. Remember this encoding system can handle English characters only. This can handle 2^7 bit which means 128 characters.

In this system, each character has individual number.

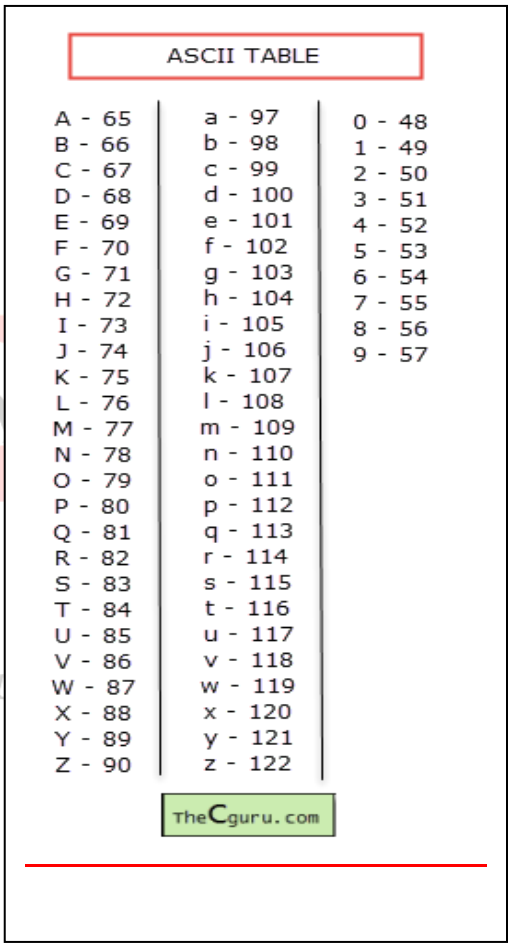
The new edition (version) ASCII -8, has 2^8 bits and can handle 256 characters are represented from 0 to 255 unique numbers.

The ASCII code equivalent to the uppercase letter 'A' is 65. The binary representation of ASCII (7 bit) value is 1000001. Also 01000001 in ASCII-8 bit.

ASCII. Pronounced ask-ee, ASCII is the acronym for the American Standard Code for Information Interchange. It is a code for representing 128 English characters as numbers, with each letter assigned a number from 0 to 127.

For example, the ASCII code for uppercase M is 77.

ASCII abbreviated from American Standard Code for Information Interchange is a character-encoding scheme. ASCII codes represent text in computers, communications equipment, and other devices that use text. Most modern character-encoding schemes are based on ASCII, though they support many additional characters



ASCII Table

AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE(ASCII)

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

www.simplifyccc.com

Character Storage

- Character sets
  - Standard ASCII: 7-bit character codes (0 – 127)
  - Extended ASCII: 8-bit character codes (0 – 255)
  - Unicode: 16-bit character codes (0 – 65,535)
  - Unicode standard represents a universal character set
    - Defines codes for characters used in all major languages
    - Used in Windows-XP: each character is encoded as 16 bits
  - UTF-8: variable-length encoding used in HTML
    - Encodes all Unicode characters
    - Uses 1 byte for ASCII, but multiple bytes for other characters
- Null-terminated String

Array of characters followed by a NULL character

ISCII (Indian Standard Code for Information Interchange)

ISCII is the system of handling the character of Indian local languages. This as a 8-bit coding system. Therefore it can handle 256 (2^8) characters. This system is formulated by the department of Electronics in India in the year 1986-88 and recognized by Bureau of Indian Standards (BIS). Now this coding system is integrated with Unicode.

ISCII (Indian Standard Code for Information Interchange)

ISCII was proposed in the eighties and a suitable standard was evolved by 1991. Here are the salient aspects of the ISCII representation.

- It is a single representation for all the Indian Scripts.
- codes have been assigned in the upper ASCII region (160 - 255) for the aksharas of the language.
- The scheme also assigns codes for the Matras (vowel extensions).

- Special characters have been included to specify how a consonant in a syllable should be rendered. Rendering of Devanagari has been kept in mind.
- A special Attribute character has been included to identify the script to be used in rendering specific sections of the text.

shown side is the basic assignment in the form of a Table. There is also a version of this table known as PC-ISCII, where there are no characters defined in the range 176-223. In PC-ISCII, The first three columns of the ISCII-91 table have been shifted to the starting location of 128. PC-ISCII has been used in many applications based on the GIST Card, a hardware adapter which supported Indian language applications on an IBM PC. In the table, some code values have not been assigned. Six columns of 16 assignments each start at the Hexadecimal value of A0 which is equivalent to decimal 160.

	A0	B0	C0	D0	E0	F0
0		ओ	ढ	ॠ	◌	EXT
1	ँ	औ	ण	ल	◌	०
2	ं	ऑ	त	ळ	◌	१
3	ः	क	थ	ॡ	◌	२
4	अ	ख	द	व	◌ ◌	३
5	आ	ग	ध	श	◌	४
6	इ	घ	न	ष	◌	५
7	ई	ङ	ऩ	स	◌	६
8	उ	च	प	ह	◌	७
9	ऊ	छ	फ	ऺ	◌	८
A	ऋ	ज	ब	◌	।	९
B	ऌ	झ	भ	◌		
C	ए	ञ	म	◌		
D	ऐ	ट	य	◌		
E	ऍ	ठ	य़	◌		
F	ऑ	ड	र	◌	ATR	

Differentiate between ASCII and ISCII

ASCII is American standard code for information interchange. **ASCII** uses a 7-bit encoding is **used** to represent character. It has 3-bits zone portion to distinguish character from numbers, but in **ASCII**-8, it has 8-bits code for characters and its zone portion contains 4-bits. ASCII is 7 bit code and comprises 128 characters to represent standard keyboard characters and various control characters

ISCII (1991): It stands for Indian Script Code for Information Interchange for Indian language.

-ISCII is 8 bit code with 256 characters , which 128 characters of ASCII and rest 128 for Indian scripts ex :- Bengali, Gujarati etc. ISCII retains all ASCII characters and offers coding for Indian Scripts only. Thus, ISCII is generally called as Indian standard code for information interchange.

Period-7

Unicode Encoding System 1 Octet & 2 Octet Representation

Unicode is a new universal coding standard adopted by all new platforms. It is promoted by Unicode Consortium which is a non profit organization. Unicode provides a unique number for every character irrespective of the platform, program and the language. It is a character coding system designed to support the worldwide interchange, processing, and display of the written texts of the diverse languages.

This coding system is used in most of the modern computers. The popular coding scheme after ASCII is Unicode. ASCII can represent only 256 characters. Therefore English and European Languages alone can be handled by ASCII. Particularly there was a situation, when the languages like Tamil, Malayalam, Kannada and Telugu could not be represented by ASCII. Hence, the Unicode was generated to handle all the coding system of Universal languages. This is 16 bit code and can handle 65536 characters.

Unicode is developed as a Universal Character Set with an aim:

- to define all the characters needed for writing the majority of known languages in use of computers in one place.
- To be superset of all other character sets that have been encoded

Unicode is a computing industry standard for the consistent encoding, representation and handling of text expressed in most of the world's writing systems.

- Unicode provides a unique number for every character,
    - no matter what the platform,
    - no matter what the program,

- no matter what the language.

Where is Unicode Used ?

- The Unicode standards has been adopted by many software and hardware vendors.
- Most OSs support Unicode.
- Unicode is required for international document and data interchange.
- Several modern standards use Unicode, such as, Programming languages, such as Java, C#, Perl, Python.
- Markup Languages such as XML, HTML, XHTML, JavaScript, LDAP, Cobra etc.

You have learnt about two different encoding schemes ASCII and ISCII in previous sections. These encoding schemes represent different sets of characters belonging to different languages by assigning a number to each of the character. Likewise, there are many encoding schemes available that represent different characters of different languages.

As world is becoming a global village thanks to modern technology, a need was being felt for an encoding scheme that could represent all the known languages characters through one encoding scheme. **Unicode is the answer.**

Unicode is developed as a universal character set with an aim:

- ❑ to define all the characters needed for writing the majority of known languages in use on computers in one place.
- ❑ to be a superset of all other character sets that have been encoded.

Encoding Terminology

Before we proceed further, it is important to discuss basic terms related to encoding.

Encoding Scheme It is a predefined way for converting information, character by character in a machine intelligible code. Same character set may be represented through different encoding schemes.

Code Space It refers to all the codes that an encoding scheme uses to represent characters.

e.g., ASCII encoding scheme has a code space from O to 127 (OxO to Ox7F)

Code Point The code point refers to a code (from a code space) that represents a single character from the character set represented by an encoding scheme, e.g., Ox41 is one code point of ASCII that represents character 'A'. A code point value represents the position of a character in the coded character set. For example, the code point for the letter '**a**' in the Unicode coded character set is 225 in decimal, or E1 in hexadecimal notation (OxE1). ASCII has 128 code points while unicode has 1,112,064 code points.

Code Unit It refers to unit of storage (number of bits used) used to represent one encoded code point. It is important to understand this e.g., UTF-8 encoding scheme uses 8 bits units to represent characters, but it is a variable length scheme. For some characters it uses just 8 bits (1 byte), for some it may use more number of 8 bit units, e.g., to represent snowman character [ ], it requires 24 bits or 3 UTF-8 units. In this case we will say that snowman code point uses 3 UTF-8 code units.

Unicode Encoding Schemes

Unicode defines multiple encoding systems to represent characters. These are UTF-8, UTF-16 and UTF-32. Let us discuss about these Unicode encoding schemes.

The character encoding scheme reflects the way the coded character set is actually mapped to bytes in form of binary code (machine intelligible code) for manipulation in a computer.

UTF-8 (Unicode Transformation Format)-8

UTF-8 is a variable-width encoding that can represent every character in Unicode character set. In other words, it can encode each of the 1,114,112 code points in the Unicode character set.

The code unit of UTF-8 is 8 bits, called an octet. **UTF-8** can use 1 to maximum 6 octets to represent code points depending on their size, although till now it has used up to 4 octets to represent any character.

UTF-8 is a type of multi-byte encoding. Sometimes you only use 8 bits to store the character, other times, 16 or 24 or more bits. The challenge with a multi-byte encoding is how do you know, for example, that these 16 bits represent a single two-byte character and not two one-byte characters. UTF-8 solves this character boundary problem

Unicode code points are often written as **U+<code point number>** e.g., U+0041 represents letter 'A'. We shall represent following Unicode code points with UTF-8 encoding for understanding purposes:

Unicode CodePoint	Symbol/character
U + 0024	'$'
U + 0041	'A'
U + 00A2	¢
U + 00F1	ñ
U + 2122	™

Before we proceed, let us recall that **UTF-8** is a variable length encoding scheme. That is, it uses different number of bytes or octets (set of 8 bits) to represent different characters. Following table gives an idea about how many octets will be used to represent a Unicode code point.

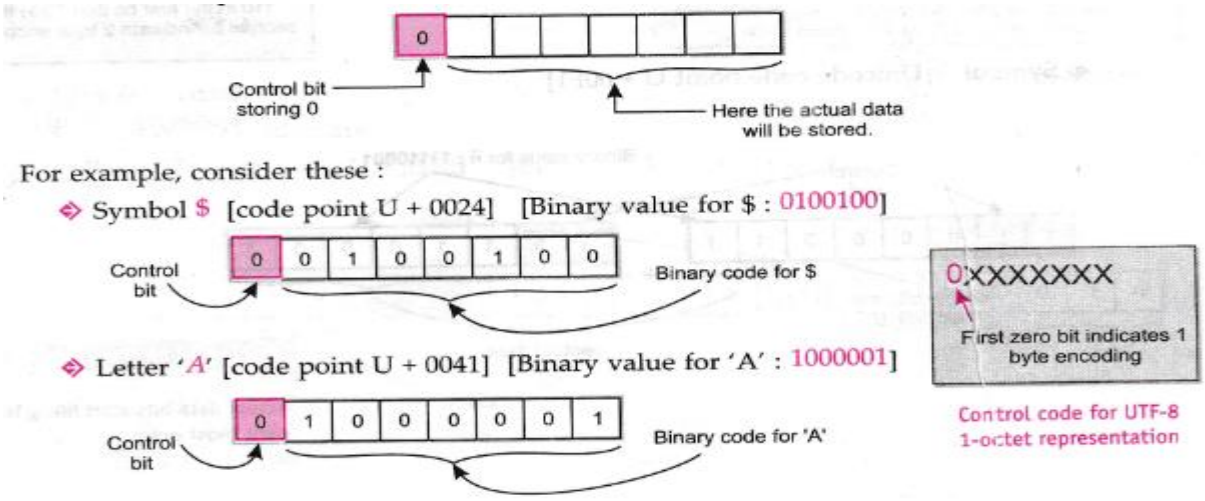
Unicode Code Points and No. of Octets used in UTF-8

Unicode Code Points (in decimal)	Unicode Code Points (in Hexadecimal)	Number of Octets used
U-0 – U-127	(U+00 to U+07F)	1 octet (8 bits)
U-128 – U-2047	(U+80 to U+7FF)	2 octets (16 bits)
U-2048 – U-65535	(U+800 to U+FFFF)	3 octets (24 bits)
U-65536 – U-2097151	(U+10000 to U+1FFFFF)	4 octets (32 bits)

UTF-8 1 Octet (8-bits) Representation

For 1 byte or 1 octet representation, UTF-8 uses the left-most bit as control bit which stores control code as 0. Control bits are special bits that store the control code and not the actual data. The rest of the bits store the actual data's binary code.

That is, for code points U + 0 to U + 7F (0 to 127 decimal), UTF-8 will store them in octets like

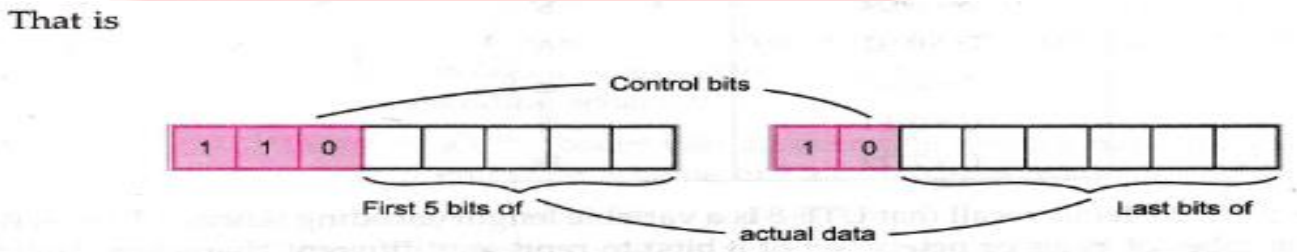


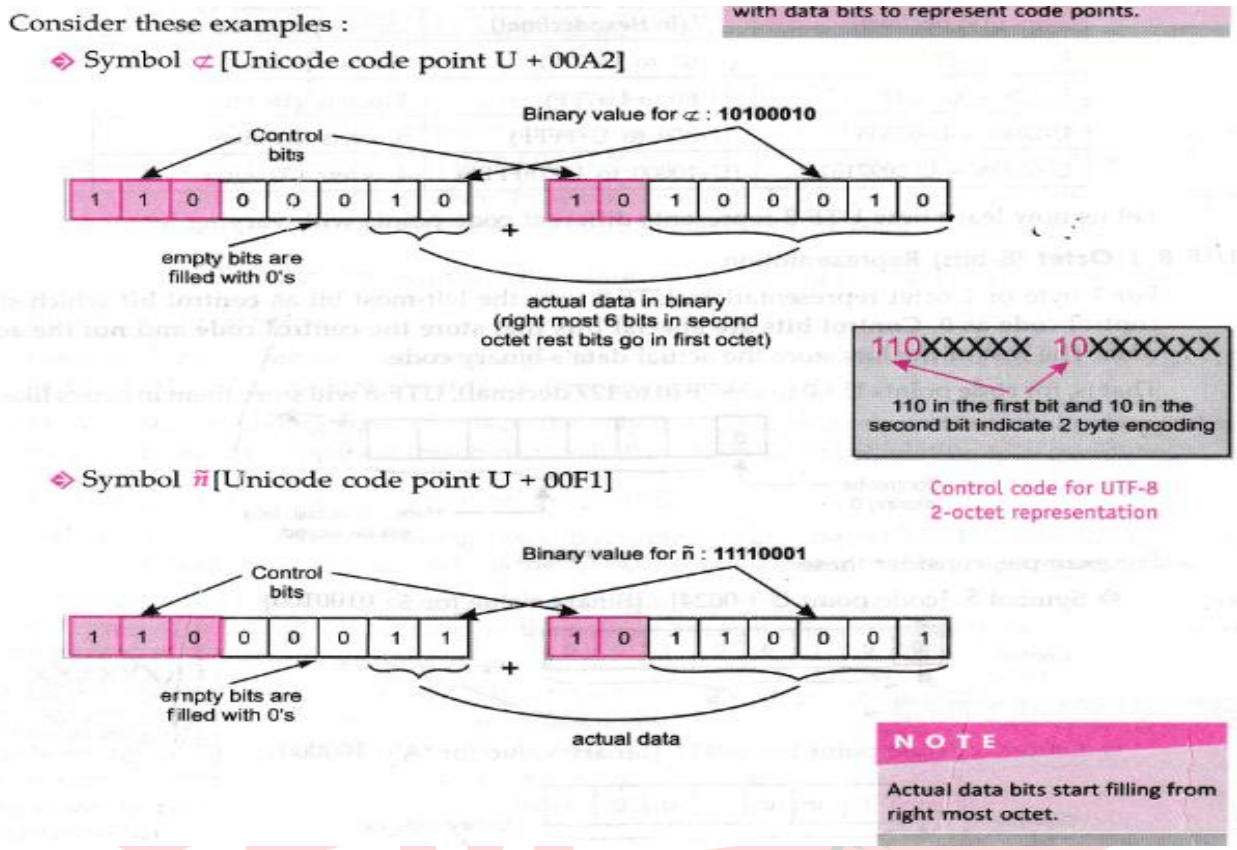
UTF-8 2 Octet (8-bits) Representation

For 2 byte or 2 octet representation, UTF-8 uses these bits for storing control code 110, 10 :

-> Left most 3 bits of first octet that store control code 110.

-> Leftmost 2 bits of next octet that store control code 10.





Period-8

Unicode Encoding System 3 Octet & 4 Octet Representations

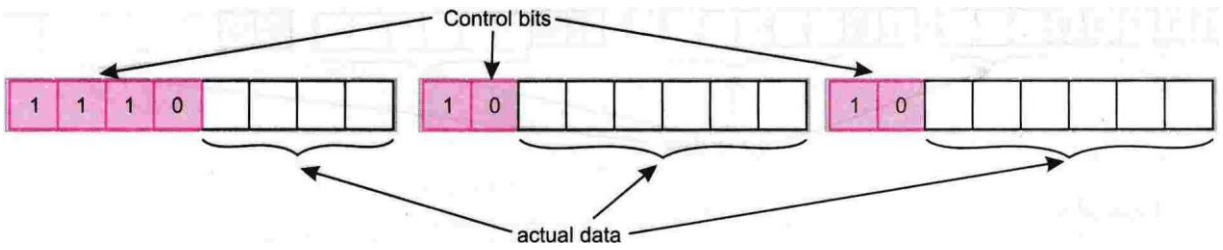
UTF-8 3 Octet (24-bits) Representation

For 3 byte or 3 octet representation, UTF-8 uses following bits for storing control code :

¢) Left most 4 bits of first octet to store control code 1110.

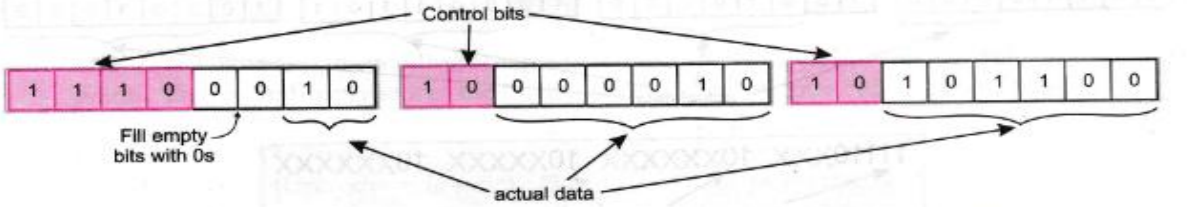
¢)Leftmost 2 bits of middle octet to store control code 10.

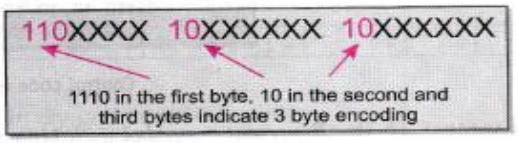
¢)Leftmost 2 bits of third octet to store control code 10.



Consider these examples :

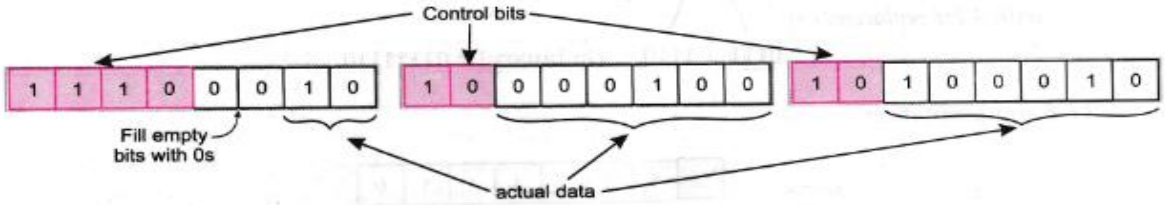
- Symbol € [Unicode code point U + 20AC] [Binary value : 10000010101100]





Control code for UTF-8 3-octet representation

- Symbol ™ (Trademark)
  [Unicode code point U + 2122] [Binary code : 10000100100010]



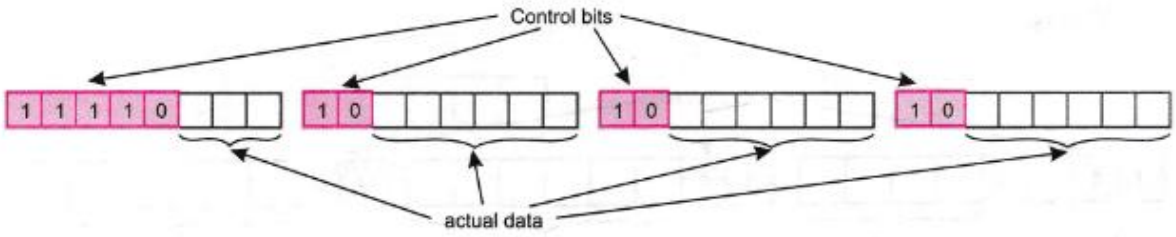
UTF-8 4 Octet (32-bits) Representation

For 3 byte or 3 octet representation, UTF-8 uses following bits for storing control code :

¢) Left most 4 bits of first octet to store control code 1110.

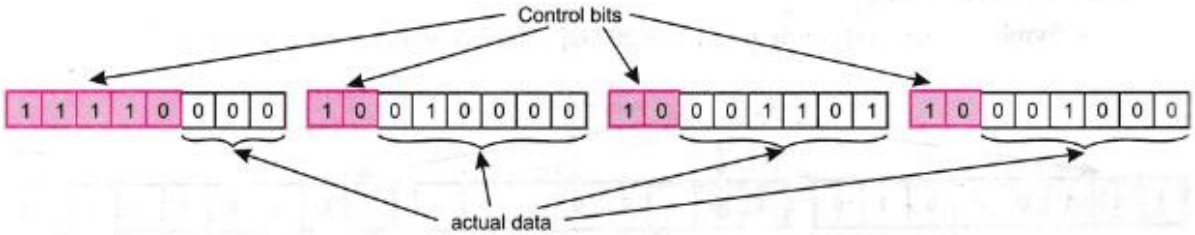
¢)Leftmost 2 bits of middle octet to store control code 10.

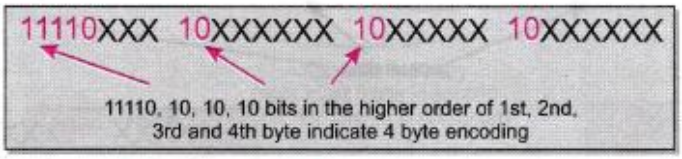
¢)Leftmost 2 bits of third octet to store control code 10.



Examples :

- **Symbol** . [Unicode codepoint U + 10348] [Binary value : 00001000001101001000]





Control code for UTF-8 4-octet representation

Examples related to this.

EXAMPLE 13.16 *Represent ~ symbol with Unicode code point as U+007E in hex and 126 in decimal, in UTF-8.*

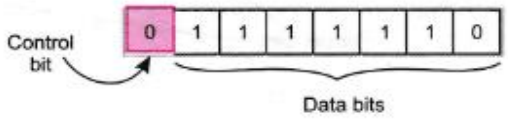
Solution. For Unicode code points in the decimal range **0-127**, **UTF-8 1-octet representation** is used. (Refer Table 13.8)

Given code point : U + 00 7E (in hex)

(Converting hex to binary with 4 bit replacement)

0111 1110 (in binary) = **01111110**

UTF-8 representation will be



EXAMPLE 13.17 *Represent © copyright symbol with Unicode code point U+00A9 in hex* **in UTF-8 encoding.**

Solution. Given code point 00A9 in hex is equivalent to $A \times 16^1 + 9 \times 16^0 = 169$ in decimal. 169 falls in range **128-2047**.

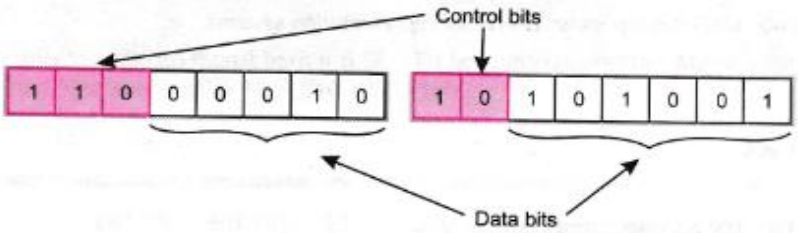
For Unicode code points in decimal range 128...2047, **UTF-8 2-octet** representation is used. *(Refer Table 13.8)*

Given code point is U + 0 0 A 9 (in hex)

(Converting hex to binary with 4-bit replacement)

1010 1001 = **10101001**

UTF-8 representation will be :



****