

**BCA- III**  
**Computer System Architecture**

**Unit I- Data Representation**

- Q. 1**      What do you mean by Data Representation? Explain the all data types with example.      2005  
Explain the difference between internal and external representation of data using suitable examples.      2006

**Ans.- Data Representation**

Data representation refers to the method used internally to represent information stored in a computer. Computer store lot of different types of information- numbers, text, graphics of many varieties (still, video, animation) Character and numbers are understandable and usable by people therefore people feed them to the computer and they expect the result of computation or interaction from computer as output many time in the same form i.e, characters of text of character after processing must appear for them as in English like natural language and processed numbers or results as represented in decimal and in accordance with mathematical notation.

**External Data Representation:**

Natural (English) language characters and decimal numbers are usual be and understandable by the people and termed as External Data Representation. But as such the computer being electronic machine cannot understand and use natural language symbols and decimal numbers directly. External data Representation does not hold good for computations inside the machine.

**Internal Data Representation:**

The internal representation of data inside the computer machine must be acceptable by machine and suit the electronic concepts. This led to the development of separate machine language which is in binary form, consequently the data to be processed must also be in binary form. Different methods of representing natural language symbols and decimal numbers in binary form inside the machine constitute internal data representation.

The characters and numbers are fed to the computer machine and the results produced from the machine, must be in a form that is usable and understandable to the external world; irrespective of internal data representation. The external input to the computer and external output from the computer will normally be natural language symbols and characters, numbers in decimal form.

- Q. 2**      Represent decimal number 8620 in the following:      2005  
(i) BCD  
(ii) Excess-3 code  
(iii) Gray codes  
(iv) Binary number  
(v) Hexadecimal

**Ans.-** Decimal number 8620 is represented as-

- (i)      **BCD-** In this representation, each digit of decimal is expressed in 4 bit value so above number is expressed as (1000 0110 0010 0000 )<sub>BCD</sub>

- (ii) **Excess-3 code-** To obtain excess-3 code, add three (0011) into each nibble of BCD code, so excess-3 of 8620 is ( 1011 1001 0101 0011 )<sub>2</sub>
- (iii) **Gray codes-** It is an unweighted code, at successive steps only single bit is changed. It is obtain from binary number. Write the MSB as it is, and write sum of the adjacent bit and ignore the carry till getting LSB. So, its code is (11000101111010)<sub>Gray</sub>
- (iv) **Binary number-** Decimal number 8620 is divided by 2, write remainder at right; this is repeated till quotient is less than base. Write remainder from bottom to top (10000110101100)<sub>2</sub>
- (v) **Hexadecimal-** It is obtained from making a group of 4 bit binary value from LSB to MSB, represent each nibble in its corresponding digit, so its value is (21AC)<sub>16</sub>.

**Q. 3**      **Derive the circuits for a 3 bit parity generator and 4-bit parity checker using an even-parity bit.**      **2005**

**Ans.-** A parity bit is used for the purpose of detecting errors during transmission of binary information. It is an extra bit included with a binary message to make the number of 1's either odd or even. The message including the parity bit is transmitted and then checked at the receiving end for errors. The circuit that generates the parity bit in the transmitted is called parity generator and the circuit that checks the parity in the receiver is called a parity checker. Parity generator circuit generates even/odd parity for 3-bits. Truth table for 3-bit parity generator is given as follows-

S. No.	Input (Three bit message)			Output (Even parity bit)
	X	Y	Z	P
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

From the truth table the expression for the output parity bit is,

$$P(X,Y,Z) = \sum(1,2,4,7)$$

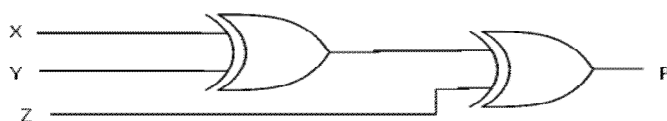
Also written as,

$$P = X'Y'Z + X'YZ' + XY'Z' + XYZ$$

$$P = (X'Y' + XY)Z + (X'Y + XY')Z'$$

$$P = (X \oplus Y)'Z + (X \oplus Y)Z'$$

$$P = X \oplus Y \oplus Z$$





So, for subtracting  $+10 - 5$  by 2's complement, first of all change the decimal value into binary form, then convert negative value into its 2's complement form. 2's complement of (-5) is 1011, so

$$\begin{array}{rcl} +10 & = & (1010)_2 \\ -5 & = & (0101)_2 \rightarrow 2's \text{ complement} \rightarrow 1011 \end{array}$$

+10	Write as it is	(1010) <sub>2</sub>
-5	2's complement	(1011) <sub>2</sub>
		10101 <sub>2</sub>

After ignoring carry, final answer is 0101 i.e. +5.

Subtraction of  $111001 - 101010$  by 2's complement is done by converting negative binary number 101010 into its 2's complement form and then add it to positive binary 111001. After addition ignore carry if carry is generated, otherwise again complement it into 2's complement form and change the sign-

(111001) <sub>2</sub>	Write as it is	(111001) <sub>2</sub>
- (101010) <sub>2</sub>	2's complement	(010110) <sub>2</sub>
		1001111 <sub>2</sub>

Finally, we get  $001111_2$  after ignoring the carry.

Similarly, we can subtract  $110110_2$  from the  $101101_2$ -

(101101) <sub>2</sub>	Write as it is	(111001) <sub>2</sub>
- (110110) <sub>2</sub>	2's complement	(001010) <sub>2</sub>
		1000011 <sub>2</sub>

After ignoring the carry, we obtain  $000011_2$ .

- Q. 6** Perform the subtraction with the following unsigned binary number by taking the 2's complement of the subtrahend: 2005
- (i)  $11010 - 10000$
  - (ii)  $11010 - 1101$
  - (iii)  $100 - 110000$
  - (iv)  $1010100 - 1010100$
  - (v)  $1011101 - 111011$

**Ans.-** Here, subtraction is performed by using 2's complement method, in which only negative value is complemented. And if carry is obtained after adding with positive, ignore the carry, otherwise again complement the obtained result to get final result.

- (i)  $11010 - 10000$

(11010) <sub>2</sub>	Write as it is	(11010) <sub>2</sub>
- (10000) <sub>2</sub>	2's complement	(10000) <sub>2</sub>
		101010 <sub>2</sub>

Finally, we get  $01010_2$  after ignoring the carry.

- (ii)  $11010 - 1101$

Here, the negative binary bits are less than the positive value, so make it to 5 bits value, so it may be 01101, and then convert it into 2's complement form- 10011

$(11010)_2$	Write as it is	$(11010)_2$
$-(01101)_2$	2's complement	$(10011)_2$
		$10110_2$

After ignoring carry, we get final answer-  $0110_2$ .

(iii)  $100 - 110000$

$(100)_2$	Write as it is	$(000100)_2$
$-(110000)_2$	2's complement	$(010000)_2$
		$010100_2$

Here, there is no carry, so we have to again convert the obtained result into 2's complement form and then change the sign. So, we get  $-101100_2$ .

(iv)  $1010100 - 1010100$

$(1010100)_2$	Write as it is	$(1010100)_2$
$-(1010100)_2$	2's complement	$(0101100)_2$
		$10000000_2$

Finally, we get  $0000000_2$  output.

(v)  $1011101 - 111011$

$(1011101)_2$	Write as it is	$(1011101)_2$
$-(111011)_2$	2's complement	$(1000101)_2$
		$10100010_2$

Final answer is  $010010_2$ .

**Q. 7** Explain with examples:

(i) Gray code

(ii) BCD code

(iii) Overflow and underflow

(iv) Error detection codes

(v) Excess 3 code

2007,

2009,

2011,

2012

2012,

2010,

2006

2007,

2009,

2012

**Ans.-**

(i) **Gray code-** It is an unweighted code. At successive steps only single bit is changed. It is used to design servo related application, Analog to Digital converter, addressing of K-map. The binary number is easily converted into gray code by following steps-

(a) Write the MSB as it is to get MSB of gray.

(b) Add adjacent bit, write sum and ignore carry.

(c) Repeat step (b) till LSB.

(ii) **BCD (Binary Coded Decimal)**- It is also called 8421 code. It is a weighted code. It is used to represent decimal number into binary form easily. In this, each digit of decimal number is represented by 4-bit binary form. 0 to 9 is represented as 0000 to 1001,  $(10)_{10}$  is expressed as  $(0001\ 0000)_{BCD}$ . The main advantage of BCD is that any decimal value can be expressed into binary form very easily. But it is very difficult to perform arithmetic operation.

(iii) **Overflow and underflow**- In computer system, registers inside the CPU are used to store data; the size of the register is fixed, it may be 8-bit, 16-bit etc. Because of the limitation it stores only limited range of data. If the represented data is out of range then it may be called underflow or overflow. If the data is more than the upper limit, it is called overflow otherwise called underflow. Overflow and underflow are detected by carry bits. If carry comes from previous bits of MSB, then it is called Carry in, finally obtained carry is called carry out. Overflow and underflow are occurred if both the carries are unequal.

In the case of overflow, carry in =1 and carry out=0. In underflow it is reverse i.e. carry in=0 and carry out=1. In the normal operation, both the carries are equal.

(iv) **Error detection codes**- Any internal or external noise like heat, magnetism and other form of electricity alter binary data. In a single-bit error, a 0 is changed to 1 and a 1 to 0. In burst error, multiple bits are changed. Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination. There are four types of redundancy checks are used in data communications; vertical redundancy check (VRC) (also called parity check), longitudinal redundancy check (LRC), cyclical redundancy check (CRC), and checksum. In vertical redundancy check, a parity bit is added to every data unit so that the total number of 1s becomes even. It can detect all single-bit errors. It can detect burst errors only if the total number of errors in each data unit is odd. In longitudinal redundancy check, a block of bits is divided into rows and a redundant row of bits is added to the whole block. The most powerful of the redundancy checking techniques is the cyclic redundancy check. Unlike VRC and LRC, which are based on addition, CRC is based on binary division. In CRC, instead of adding bits together to achieve a desired parity, a sequence of redundant bits, called CRC or the CRC remainder, is appended to the end of a data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number. At its destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be intact and is therefore accepted.

(v) **Excess-3 code**- It is an unweighted, self-complementing code. It can be obtained by adding three (0011) into each nibble of BCD code, for example excess-3 code of  $(78)_{10}$  is  $(1010\ 1011)_{X-3}$ . Basically, this code is used to solve the arithmetic problem of BCD. In BCD addition operation, obtained result may be in BCD or may not. In addition process of Excess- 3, if carry occur, then add 3 into generated nibble group and carry also, otherwise subtract 3 from nibble to get final excess-3 result.

**Q. 8** Explain the following using proper illustrations:  
(i) ASCII codes  
(iv) Error correction codes

2011

**Ans.-**

**(i) ASCII codes-** It stands for American Standard Code for Information Interchange. ASCII of two types: ASCII-7 and ASCII-8. ASCII-7 is 7 bits code, where first 3 bits are used as zone bits and last 4-bits are digit bits. It can represent  $2^7 = 128$  characters. ASCII-8 is extended version of ASCII-7. It is an 8-bits code with 4 bits as zone bit. It can represent  $2^8 = 256$  characters. ASCII also uses hexadecimal as its four to one short-cut notation for memory dump.

**(ii) Error correction codes-** Error detection code detects errors only does not correct them. Error correction can be handled in two ways. In one, when an error is discovered, sender retransmits the entire data unit. In the other, a receiver can use an error- correcting code which automatically corrects certain errors. To correct the error, the receiver simply reverses the value of the altered bit. The secret of error correction, therefore, is to locate the invalid bit or bits. Hamming code is used to correct single bit error. In this, transmitter calculates the redundancy code which is sent with data bits. At receiver end, recalculate the redundancy code and it is compared with transmitted redundancy code, if zero then there is no error, otherwise it shows the error position.

To calculate the number of redundancy bits (r) required to correct a given number of data bits (m),  $2^r$  must be equal to or greater than  $m+r+1$  ( $2^r \geq m+r+1$ ). The length of the resulting code is  $m+r$ . Redundancy bits are placed in positions 1, 2, 4 and 8. For clarity, we refer to these bits as  $r_1, r_2, r_4$  etc. In Hamming code, each r bits is the VRC bit for one combination of data bits:  $r_1$  is the VRC bit for one combination of data bits;  $r_2$  is the VRC bit for another combination of data bits, and so on. The combination used to calculate each of the four r values for a seven-bit data sequence are as follows-

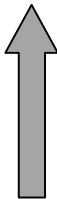
- $r_1$  : bits 1, 3, 5, 7, 9, 11
- $r_2$  : bits 2, 3, 6, 7, 10, 11
- $r_4$  : bits 4, 5, 6, 7
- $r_8$  : bits 8, 9, 10, 11

In the subsequent steps, we calculate the even /odd parities for the various bit combinations. The parity value for each combination is the value of the corresponding r bit. The receiver takes the transmission and recalculates four new VRCs using the same sets of bits used by the sender plus the relevant parity (r) bit for each set. Then it assembles the new parity values into a binary number in order of r position. Once the bit is identified, the receiver can reverse its value and correct the error.

**Q. 9** Convert the following:  
 $(255)_{10} = ( )_2$   
 $(175)_8 = ( )_{10}$   
 $(3FF)_{16} = ( )_{10}$   
 $(11101010)_2 = ( )_{16}$

2008,  
2012

**Ans.- (i)  $(255)_{10} = ( ? )_2$**

Decimal number = 255			
Division expression	Quotient	Remainder	Direction
255 / 2	127	1	
127 / 2	63	1	
63 / 2	31	1	
31 / 2	15	1	
15 / 2	7	1	
7 / 2	3	1	
3 / 2	1	1	
1 / 2	0	1	
Binary number = (11111111) <sub>2</sub>			

Hence  $(255)_2 = ( 11111111 )_2$

**(ii)  $(175)_8 = ( ? )_{10}$**

Number	1	7	5
ON/OFF	ON	ON	ON
Place value	$8^2$	$8^1$	$8^0$
Exponential Expression	$1 * 8^2$	$7 * 8^1$	$5 * 8^0$
Solved Multiplication	64	56	5
Add to calculate decimal value	$64 + 56 + 5 = 125$		

Hence,  $(175)_8 = (125)_{10}$

**(iii)  $(3FF)_{16} = ( )_{10}$**

Number	3	F	F
ON/OFF	ON	ON	ON
Place value	$16^2$	$16^1$	$16^0$
Exponential Expression	$3 * 16^2$	$15 * 16^1$	$15 * 16^0$
Solved Multiplication	768	240	15
Add to calculate decimal value	$768 + 240 + 15 = 1023$		

Hence,  $(3FF)_{16} = (1023)_{10}$

**(iv)  $(11101010)_2 = ( )_{16}$**

Make a group of 4 bits from LSB i.e. 1110 1010



Binary group	1110	1010
Hexa equivalents	E	A
Hexa number	EA	

Hence,  $(11101010)_2 = (EA)_{16}$

- Q. 10** Convert the following:
- (i)  $(101110110)_2 = ( )_{10}$
- (ii)  $(110110110111)_2 = ( )_{16}$
- (iii)  $(15)_8 = ( )_{10}$
- (iv)  $(FCB)_{16} = ( )_{10}$

2007,  
2010

**Ans.-** (i)  $(101110110)_2 = ( )_{10}$

Number	1	0	1	1	1	0	1	1	0
ON/OFF	ON	OFF	ON	ON	ON	OFF	ON	ON	OFF
Place value	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
Exponential Expression	$1*2^8$	$0*2^7$	$1*2^6$	$1*2^5$	$1*2^4$	$0*2^3$	$1*2^2$	$1*2^1$	$0*2^0$
Solved Multiplication	256	0	64	32	16	0	4	2	0
Add to calculate decimal value	$256+0+64+32+16+0+4+2+0 = 374$								

Hence,  $(101110110)_2 = (374)_{10}$

(ii)  $(110110110111)_2 = ( )_{16}$

Make a group of 4 bits from LSB i.e. 1101 1011 0111

Binary group	1101	1011	0111
Hexa equivalents	D	B	7
Hexa number	DB7		

(iii)  $(15)_8 = ( )_{10}$

Number	1	5
ON/OFF	ON	ON
Place value	$8^1$	$8^0$
Exponential Expression	$1 * 8^1$	$5 * 8^0$
Solved Multiplication	8	5
Add to calculate	$8+5 = 13$	

decimal value	
---------------	--

Hence,  $(15)_8 = (13)_{10}$

(iv)  $(FCB)_{16} = ( )_{10}$

Number	F	C	B
ON/OFF	ON	ON	ON
Place value	$16^2$	$16^1$	$16^0$
Exponential Expression	$15 * 16^2$	$12 * 16^1$	$11 * 16^0$
Solved Multiplication	3840	192	11
Add to calculate decimal value	$3840 + 192 + 11 = 4043$		


Hence,  $(FCB)_{16} = (4043)_{10}$

**Q. 11** Express the number  $(43.52)_8$  in decimal and  $(43.52)_{10}$  in hexadecimal form. 2006

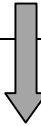
**Ans.-**

Number	4	3	5	2
ON/OFF	ON	ON	ON	ON
Place value	$8^1$	$8^0$	$8^{-1}$	$8^{-2}$
Exponential Expression	$4 * 8^1$	$3 * 8^0$	$5 * 8^{-1}$	$2 * 8^{-2}$
Solved Multiplication	32	3	0.625	0.03125
Add to calculate decimal value	$32+3+0.625+0.03125 = 35.65625$			

**$(43.52)_{10}$  in hexadecimal form-**

Integer number = 43			
Division expression	Quotient	Remainder	Direction
43 / 16	2	11=B	
2 / 16	0	2	
Hexa number = (2B) <sub>16</sub>			

**Fractional part = .52**

Direction	Hexa digit	$.52 * 16$
	8	$.32 * 16$
	5	$.12 * 16$

	1	.92 * 16
	14 = E	.72 * 16
	11 = B	.52
Hexa number = .851EB..... recurring		

Hence  $(43.52)_{10} = (2B.851EB...)_{16}$

**Q. 12** Convert the following:

2009

(i)  $(624)_8 = ( )_{10}$

(ii)  $(6EA)_{16} = ( )_2$

(iii)  $(654)_{10} = ( )_8$

**Ans.-**

(i)  $(624)_8 = ( )_{10}$

Number	6	2	4
ON/OFF	ON	ON	ON
Place value	$8^2$	$8^1$	$8^0$
Exponential Expression	$6 * 8^2$	$2 * 8^1$	$4 * 8^0$
Solved Multiplication	384	16	4
Add to calculate decimal value	$384 + 16 + 4 = 404$		


Hence,  $(624)_8 = (404)_{10}$

(ii)  $(6EA)_{16} = ( )_2$

Hexadecimal number	6	E	A
Binary equivalents to Hexadecimal digit	0110	1110	1010
Binary number	011011101010		

Hence,  $(6EA)_{16} = (011011101010)_2$

(iii)  $(654)_{10} = ( )_8$

Decimal number = 654			
Division expression	Quotient	Remainder	Direction
654 / 8	81	6	
81 / 8	10	1	
10 / 8	1	2	
1 / 8	0	1	
Octal number = (1216) <sub>8</sub>			

Hence,  $(654)_{10} = (1216)_8$

**Q. 13** Convert the following:

2011

(i)  $(1101.1001)_2 = ( )_{10}$

(ii)  $(37.65)_{10} = ( )_2$

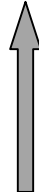
(iii)  $(10001111101001)_2 = ( )_8$

(iv)  $(43.52)_{10} = ( )_{16}$

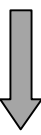
**(i)  $(1101.1001)_2 = ( )_{10}$**

Number	1	1	0	1	.	1	0	0	1
ON/OFF	ON	OFF	OFF	ON		ON	OFF	OFF	ON
Place value	$2^3$	$2^2$	$2^1$	$2^0$		$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$
Exponential Expression	$1*2^3$	$1*2^2$	$0*2^1$	$1*2^0$		$1*2^{-1}$	$0*2^{-2}$	$0*2^{-3}$	$1*2^{-4}$
Solved Multiplication	8	4	0	1		.5	0	0	.0625
Add to calculate decimal value	$8+4+1+0.5+0.0625=13.5625$								

**(ii)  $(37.65)_{10} = ( )_2$**

Integer number = 37			
Division expression	Quotient	Remainder	Direction
37 / 2	18	1	
18 / 2	9	0	
9/2	4	1	
4/2	2	0	
2/2	1	0	
1/2	0	1	
Binary number = (100101) <sub>2</sub>			

**Fractional part = .65**

Direction	Binary number	$.65 * 2$
	1	$.30 * 2$
	0	$.60 * 2$
	1	$.20 * 2$
	0	$.40 * 2$
	0	$.80 * 2$
	1	$.60 * 2$
Binary number = .101001..... recurring		

Hence  $(37.65)_{10} = (100101.101001\dots)_2$


(iii)  $(10001111101001)_2 = ( )_8$

Make a group of 3 bits from LSB i.e. 10 001 111 101 001

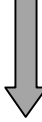
Binary group	010	001	111	101	001
Octal equivalents	2	1	7	5	1
Octal number	$(21751)_8$				

Hence,  $(10001111101001)_2 = (21751)_8$

(iv)  $(43.52)_{10} = ( )_{16}$

Integer number = 43			
Division expression	Quotient	Remainder	Direction
43 / 16	2	11=B	
2 /16	0	2	
Hexa number = (2B) <sub>16</sub>			

**Fractional part = .52**

Direction	Hexa digit	$.52 * 16$
	8	$.32 * 16$
	5	$.12 * 16$
	1	$.92 * 16$
	14 = E	$.72 * 16$
	11 = B	$.52$
Hexa number = .851EB..... recurring		

Hence  $(43.52)_{10} = (2B.851EB\dots)_{16}$