Computer Organization and Architecture Tutorial Session – 3 (Date – 28/01/2020)

1. **Why do we need to use different number representation systems?**

Binary is needed in computers because computer systems can recognize only 2 states, on (1) or off (0), so all instructions need to be coded in 2 states, which is the binary system. By comparison, octal was developed because a few traditional systems (like UNIX) works on clumps of four groups of three boolean values, and 3 binary digits at a time helps shorten the whole process of representation of a binary number. In contrast, Hexadecimal numbers help shorten the whole representation even further, and take up less digits as opposed to decimal for the same number (decimal byte values can take 50% more digits than hex). Also, newer addressing modes like IPV6 directly incorporates HEX numbers in the addressing mode for ease of use.

2. **Give the generalized steps for conversion from Hexadecimal, Binary or Octal number to decimal.**

- Get the last digit of the number, call this digit the **currentDigit**.
- Make a variable, let's call it **power**. Set the value to 0.
- Multiply the **current digit** with (n^**power**), store the result, where n is 2 for binary, 16 for hex and 8 for octal Increment **power** by 1.
- Set the the **currentDigit** to the previous digit of the number**.**
- Repeat from step 3 until all digits have been multiplied. Sum the result of step 3 to get the answer number.

3. **Give the generalized code for conversion from Decimal to Hex, Binary or Octal**

   a. Divide the decimal number by n. Treat the division as an integer division, where n is 2 for binary, 8 for octal and 16 for hex
   b. Write down the remainder (in hexadecimal, octal or binary).
   c. Divide the result again by n. Treat the division as an integer division.
   d. Repeat step 2 and 3 until result is 0.
   e. The hex, binary or octal value is the digit sequence of the remainders from the last to first.

4. **Represent 1011011.01101_2 into Hexadecimal**

0101 1011 .0110 1000

Decimal Number	4-bit Binary Number	Hexadecimal Number
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8

9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10 (1+0)
17	0001 0001	11 (1+1)

So, $1011011.01101_2 = 5B.68_{16}$

Alternatively, we can go from binary to any intermediate number format (like decimal) by multiplication and reconvert back to hexadecimal (by division by 16)

5. Explain various types of complements.

Complements are used to simplify the subtraction operation and for logical manipulation.

Diminished Radix or (r-1)'s Complement

- Given a number N in base r having n digits, the (r – 1)'s complement of N is defined as **(r – 1) – N**.
- The 9's complement of 546700 is
  999999 – 546700 = 453299
- The 1's complement of a binary number is formed by changing 1's to 0's and 0's to 1's.

Radix or r's Complement

- The r's complement of n-digit number N in base r is defined as $r^n - N$ for $N \neq 0$ and 0 for N = 0.
- Comparing with the (r – 1)'s complement, the r's complement is obtained by adding 1 to the (r – 1)'s complement since $r^n - N = [(r^n - 1) - N] + 1$.
- The 10's complement of decimal 2389 is $(10^4 - 1) - 2389 + 1 = 7611$.
- The 2's complement of binary 1101100 is 0010100

6. Assume numbers are represented in 8-bit twos complement representation. Show the calculation of the following:

a. 6 + 13
b. -6 + 13
c. 6 – 13
d. -6 – 13

+6 00000110	−6 11111010	+6 00000110	−6 11111010
+13 00001101	+13 00001101	−13 11110011	−13 11110011
+19 00010011	+7 00000111	−7 11111001	−19 11101101

7. Give the value of the following number in 8-bit 1's complement representation, 8-bit 2's complement signed representation and 8-bit sign-magnitude representation.

A) +88 **B.) -88**

Assignment question

C.) -127 **D.) +127**

Solution.

In 1's complement representation:

Again, the most significant bit (msb) is the sign bit, with value of 0 representing positive integers and 1 representing negative integers.

The remaining n-1 bits represents the magnitude of the integer, as follows:

- For positive integers, the absolute value of the integer is equal to "the magnitude of the (n-1)-bit binary pattern".
- For negative integers, the absolute value of the integer is equal to "the magnitude of the complement (inverse) of the (n-1)-bit binary pattern" (hence called 1's complement).

In 2's complement representation:

The most significant bit (msb) is the sign bit, with value of 0 representing positive integers and 1 representing negative integers.

The remaining n-1 bits represents the magnitude of the integer, as follows:

- For positive integers, the absolute value of the integer is equal to "the magnitude of the (n-1)-bit binary pattern".
- For negative integers, the absolute value of the integer is equal to "the magnitude of the complement of the (n-1)-bit binary pattern plus one" (hence called 2's complement).

In sign-magnitude representation:

- The most-significant bit (msb) is the sign bit, with value of 0 representing positive integer and 1 representing negative integer.
- The remaining n-1 bits represents the magnitude (absolute value) of the integer. The absolute value of the integer is interpreted as "the magnitude of the (n-1)-bit binary pattern".

A.)
8-bit 1's complement representation of +88 = (01011000)
8-bit 2's complement signed representation of +88 = (01011000)
8-bit sign-magnitude representation of +88 = (01011000)

B.)
8-bit 1's complement representation of -88 = (1010 0111)
8-bit 2's complement signed representation of -88 = (1010 1000)
8-bit sign-magnitude representation of -88 = (1101 1000)

Assignment solution =>

C.)
8-bit 1's complement representation of -127 = (10000000)
8-bit 2's complement signed representation of -127 = (10000001)
8-bit sign-magnitude representation of -127 = (11111111)

D.)
8-bit 1's complement representation of +127 = (01111111)
8-bit 2's complement signed representation of +127 = (01111111)
8-bit sign-magnitude representation of +127 = (01111111)

8. **Add the following using 2's complement representation in 8-bit register. Also check overflow/underflow.**

   i. $+45+(-65)$

   ii. $-27+(-101)$

Assignment question

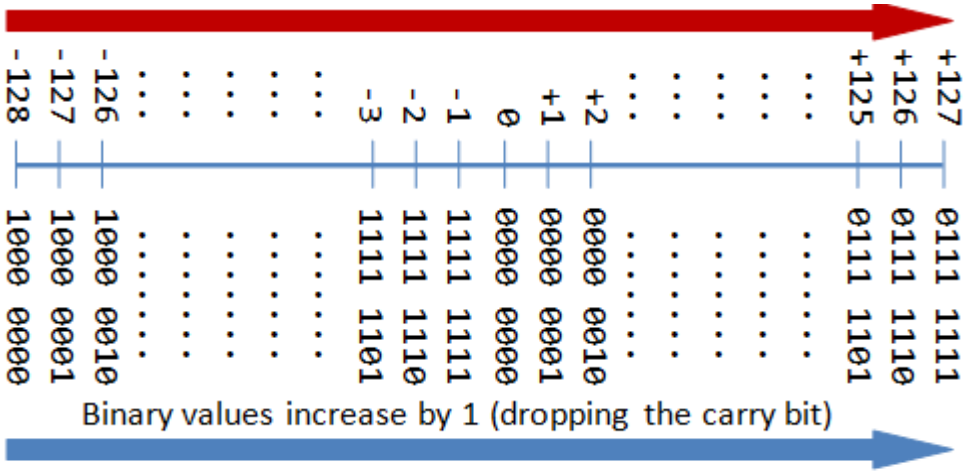
   iii. $+27+101$

   iv. $-103+(-69$

Solution:

Detecting Overflow/Underflow

Carry Bit	Sign Bit	Status
0	0	No Overflow/Underflow
1	1	
0	1	Overflow
1	0	Underflow



2's Complement Representation

i. $45 + (-65)$

Represent 45 in binary → 00101101

Represent -65 by 2's complement → 11101100

0	0	1	1	1	1	1	1		
	0	0	1	0	1	1	0	1	45
	1	0	1	1	1	1	1	1	-65
	1	**1**	**1**	**0**	**1**	**1**	**0**	**0**	-20

Carry into Sign-Bit = 0

Carry out of Sign-Bit = 0

Therefore, no overflow

ii. $-27 + (-101)$

Represent -27 by 2's complement → 11100101
Represent -101 by 2's complement → 10011011

1	1	1	1	1	1	1	1		
	1	1	1	0	0	1	0	1	-27
	1	0	0	1	1	0	1	1	-101
	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	-128

Carry into Sign-Bit = 1
Carry out of Sign-Bit = 1
Therefore, no overflow

iii. $+27 + (+101)$

Represent 27 in binary → 00011011
Represent 101 in binary → 01100101

0	1	1	1	1	1	1	1	1	
	0	0	0	1	1	0	1	1	27
	0	1	1	0	0	1	0	1	101
	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	~~128~~

Carry into Sign-Bit = 1
Carry out of Sign-Bit = 0
Therefore, overflow

iv. $-103 + (-69)$

Represent -103 by 2's complement → 10011001
Represent -69 by 2's complement → 10111011

1	0	1	1	1		1	1		
	1	0	0	1	1	0	0	1	-103
	1	0	1	1	1	0	1	1	-69
	0	**1**	**0**	**1**	**0**	**1**	**0**	**0**	~~-172~~

Carry into Sign-Bit = 0
Carry out of Sign-Bit = 1
Therefore, underflow

9. Explain different types of alphanumeric codes.

American Standard-Code for Information Interchange (**ASCII**)

The American Standard-Code for Information Interchange (ASCII) pronounced "as-kee" is a 7-bit code based on the ordering of the English alphabets. The ASCII codes are used to represent alphanumeric data in computer input/output. it is a seven-bit code, it can almost represent 128 characters. These include 95 printable characters including 26 upper-case letters (A to Z), 26 lowercase letters (a to z), 10 numerals (0 to 9) and 33 special characters such as mathematical symbols, space character etc.

ASCII 7 Code table containing 95 printable characters

(Digits) $X_3X_2X_1X_0$	$X_6X_5X_4$(Zoned bits) 010	011	100	101	110	111
0000	SP	0	@	P	a	p
0001	!	1	A	Q	b	q
0010	"	2	B	R	c	r
0011	#	3	C	S	d	s
0100	$	4	D	T	e	t
0101	%	5	E	U	f	u
0110	&	6	F	V	g	v
0111	(	7	G	W	h	w
1000	)	8	H	X	i	x
1001	*	9	I	Y	j	y
1010	*	:	J	Z	k	z
1011	+	;	K	[	l	{
1100	'	<	L	\	m	;
1101	-	+	M	]	n	y
1110	.	>	N	^	o	~
1111	/	?	O	-		DEL

An eight-bit version of the ASCII code, known as US ASCII-8 or ASCII-8, has also been developed. Since it uses 8-bits, so this version of ASCII can represent a maximum of 256 characters.

Digits $X_3X_2X_1X_0$	$X_7X_6X_5X_4$ (Zoned bits) 0101	1010	1011
0000	0		P
0001	I	A	Q
0010	1	B	R
0011	2	C	S
0100	3	D	T
0101	4	E	U
0110	S	F	V
0111	6	G	X
1000	7	H	Y
1001	8	I	Z
1010	9	J	
1011		K	
1100		L	
1101		M	
1110		N	
1111		O	

ASCII-8 Code table containing code For some characters

When the ASCII -7 codes was introduced, many computers dealt with eight -bit groups (or bytes) as the smallest unit of information. This eight bit code was commonly used as parity bit for detection of error on communication lines.

Extended Binary Coded Decimal Interchange Code (EBCDIC)

The Extended Binary Coded Decimal Interchange Code (EBCDIC) pronounced as "ebi-si disk" is another frequently used code by computers for transferring alphanumeric data. It is 8-bit code in which the numerals (0-9) are represented by the 8421 BCD code preceded by 1111. Since it is a 8-bit code, it can almost represent 23 (= 256) different characters which include both lowercase and uppercase letters in addition to various other symbols and commands.

Digits $X_3X_2X_1X_0$	$X_7X_6X_5X_4$(Zoned bits) 0100	0101	0110	0111	1000	1001	1100	1101	1110	1111
0000	SP	&		a	j	-	{		/	0
0001				b	k	s	A	J		1
0010				c	l	t	B	K	S	2
0011				d	m	u	C	L	T	3
0100				e	n	v	D	M	U	4
0101				f	o	w	E	N	V	5
0110				g	p	x	F	O	W	6
0111				h	q	y	G	P	X	7
1000				i	r	z	H	Q	Y	8
1001						l	R	Z	9	
1010	#	!	:	;						
1011	-	$	,	#						
1100	<	*	%	@						
1101	(	)								
1110	+	;	>	=						
1111	[	]	?	"						

EBCDIC was designed by IBM corp. so it is basically used by several IBM· models. In this code, we do not use a straight binary sequence for representing characters, as was in the case of ASCII code. Since it is a 8-bit code, so it can be easily grouped into groups of 4 so as to represent in arm of hexadecimal digits. By using the hexadecimal number system notation, the amount of digits used to represent various characters and special characters using EBCDIC code is reduced in volume of one is to four. Thus 8-bit binary code could be reduced to 2 hexadecimal digits which are easier to decode if we want to view the internal representation in memory. The above table lists the EBCDIC code for certain characters.

Read the above table as you read the graph. Suppose you want to search for EBCDIC code for letter *'A'*. To that case, the value of X_3 X_2 X_1 X_0 bits is 0001 and value X_7 X_6 X_5 X_4 bits is 1100.

Therefore, EBCDIC code for letter 'A' is 11000001(A). Similarly, the EBCDIC code for 'B' is 11000010(B).

The EBCDIC code '=' is 01111110

The EBCDIC code for '$' is 0101 1011

UNICODE

The ASCII and EBCDIC encodings and their variants that we have studied suffer from some limitations.

- These encodings do not have a sufficient number of characters to be able to encode alphanumeric data of all forms, scripts and languages. As a result, they do not permit multilingual computer processing.
- These encoding suffer from incompatibility. For example: code 7A (in hex) represents the lowercase letter 'Z' in ASCII code and the semicolon sign ';' in EBCDIC code.

To overcome these limitations, UNICODE also known as universal code was developed jointly by the Unicode Consortium and the International Organization for Standardization (ISO). The Unicode is a 16-bit code so it can represent 65536 different characters. It is the most complete character encoding scheme that allows text of all forms and languages to be encoded for use by computers. In addition to multilingual support, it also supports a comprehensive set of mathematical and technical symbols, greatly simplifying any scientific information interchange.

10. With an ASCII-7 keyboard, each keystroke produces the ASCII equivalent of the designated character. Suppose that you type PRINT X. What is the output of an ASCII-7 keyboard?

Solution: The sequence is as follows:

The ASCII-7 equivalent of P = 101 0000

The ASCII-7 equivalent of R = 101 0010

The ASCII-7 equivalent of I = 1001010

The ASCII-7 equivalent of N = 100 1110

The ASCII-7 equivalent of T = 1-010100

The ASCII-7 equivalent of space = 010 0000

The ASCII-7 equivalent of X = 101 1000

So the output produced is 10100001010010100101010011101010100010000001011000. The output in hexadecimal equivalent is 50 52 49 4E 54 30 58

11. A computer sends a message to another computer using an odd-parity bit. Here is the message in ASCII-8 Code.

1011 0001

1011 0101

1010 0101

1010 0101

1010 1110

What do these numbers mean?

Solution

On translating, the 8-bit numbers into their equivalent ASCII-8 code we get the word 1011 0001 (Q), 10110101 (U), 10100101 (E), 1010 0101 (E), 1010 1110 (N)

So, on translation, we get QUEEN as the output.

12. Find the excess-3 code of 6. Find the compliment of the excess-3 code of 6. Find the resultant number in decimal. What is the relation between 6 and the resultant decimal number?

Solution- Excess 3 code pf 6 is 1001.
Complement is 0110.
0110 is the excess 3 code of 3.
Hence the resultant number Is 3.

Finally, 6 and 3 are 9's complement of each other. Hence Excess-3 code is called self complementing code.

13. Repeat question 1 using the 2421 code.

Solution- 2421 code of 6 is 0110.
Complement is 1001.
1001 is the 2421 code of 3.
Hence the resultant number Is 3.

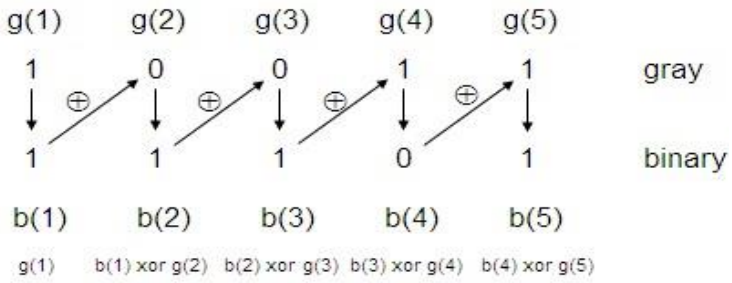
Finally, 6 and 3 are 9's complement of each other. Hence Excess-3 code is called self complementing code.

14. Find the 2421 code for 6. Is there more than one possibility for a 2421 code of 6? If so please mention all the 2421 codes of 6.

Solution: 2421 code for 6 are 1100 and 0110.

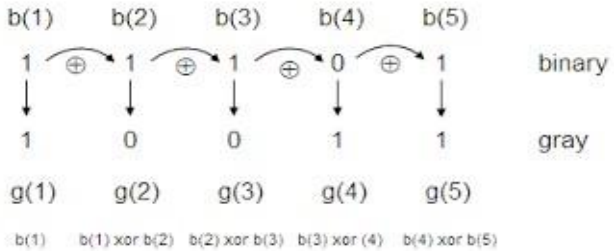
15. Find the gray code for the binary number 10011 using diagonal addition.

Solution :



16. Find the binary code from the gray code 11101.

Sol. -



17. Generate the truth table for the half and full adder. Design the circuit diagram based on the truth table.

Solution:

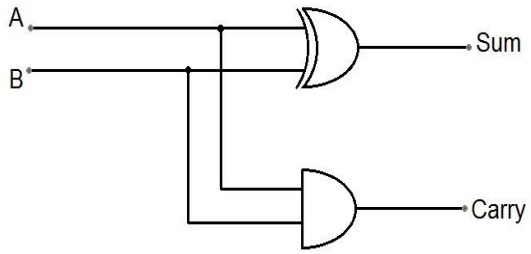
The truth table for half adder

Truth Table		
A	B	Carry
0	0	0
0	1	0
1	0	0
1	1	1

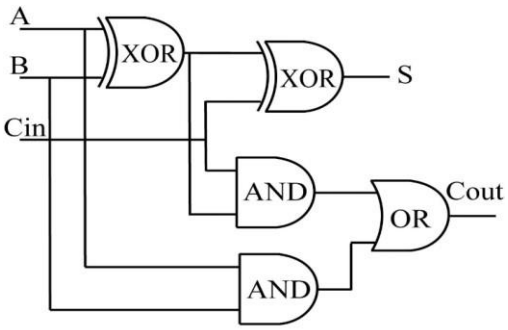
The truth table for full adder

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The circuit design of half adder



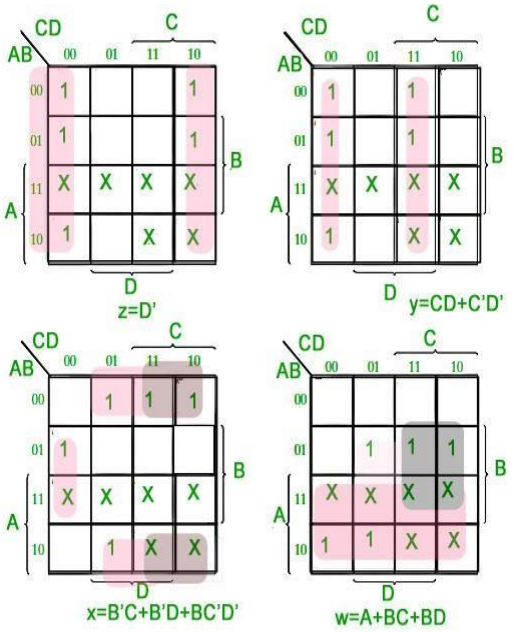
The circuit design of full adder.



18. Convert BCD to excess 3 Code? Draw circuit diagram?

Solution: Truth Table **:**

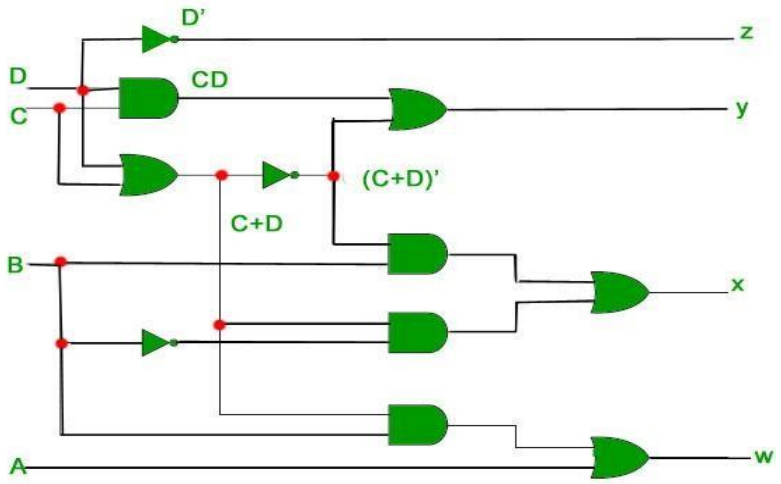
BCD(8421)				Excess-3			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



K-Map technique for each of the Excess-3 code bits as output with all of the bits of the BCD number as input.

Corresponding minimized Boolean expressions for Excess-3 code bits –

$$w = A + BC + BD$$
$$x = B'C + B'D + BC'D'$$
$$y = CD + C'D'$$
$$z = D'$$



Assignments:

Draw the circuit Diagram for Full Adder(Using Universal gates)?

Convert Excess-3 to BCD(8421)?

19. Normalize the binary number 1011.12

Solution.

A floating-point number is typically expressed in the scientific notation, with a fraction (F), and an exponent (E) of a certain radix (r), in the form of F×r^E. Decimal numbers use radix of 10 (F×10^E); while binary numbers use radix of 2 (F×2^E).

Representation of floating point number is not unique.

Eg.

the number 55.66 can be represented as 5.566×10^1

1.0111 × 2^3

20. Convert the number 13.25 in binary and normalized the representation of the number.

Solution.

Binary of 13.25 is 1101.01

Normalized floating-point representation is: 1.10101*2^3

Assignment question 1. .

Calculate normalized representation of 0.0010110 * 2^9

21. What are the range of 8-bit, 16-bit, 32-bit and 64-bit integer, in "unsigned" and "signed" representation?

Solution.

The range of the unsigned n-bit integers is => [0 to (2^n) - 1].

The range of the signed n-bit integers is => [-2^ (n -1) to +2^ (n-1)-1].