Binary Algebra & Arithmetic Operations

Arithmetic

- Arithmetic involves
  - Addition, subtraction, multiplication , division
- People use base 10
- Computers use base 2
- Base 2 is actually *easier* than base 10

- ***Computers do this*** Modulus Arithmetic
- ***32-bit registers***
- ***Each register can store an integer number***
  - ***between 0 and 2^n-1***
  - ***where n=32***
- ***if you have a value of 2^n-1 in a register, and you add 1 to this, the register will then hold a value of 0***

Example: a car odometer
if a car has 99999 miles on it,
and you drive another mile,
the odometer will then read 00000

Binary Terminology

- **Nibble – A four bit binary number**
- **Byte – A unit of storage for a single character, typically an eight bit (2 nibble) binary number (short for binary term)**
- **Word – Typically a sixteen bit (2 byte) binary number**
- **Double Word – A thirty-two bit (2 word) binary number**

Unsigned Binary to Decimal Conversion

Numbered bit position	7	6	5	4	3	2	1	0
Corresponding power of 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal equivalent of power of 2	128	64	32	16	8	4	2	1

Binary Number Addition

We are quite familiar with the decimal addition, in which the digits are added up and if a carry results then that carry is forwarded to next digit for addition. Binary addition is no different from the decimal addition. The basic binary operations are shown herewith.

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

The first three additions are self explanatory, only the last one needs some explanation, in normal addition 1+1 is 2 but in binary number system decimal 2 is represented by 10. The 1 is the carry which is forwarded to the next higher bit for addition.

Add the binary numbers 1101.101 and 0101.111

Let A_1 : *1101.101* ,A_2 : *0101.001*

Carry	:	1	**1**		**1**	**1**		**1**	**1**	
A_1	:		1	1	0	1	.	1	0	1
A_2	:		0	1	0	1	.	1	1	1
Sum	:	**1**	**0**	**0**	**1**	**1**	**.**	**1**	**0**	**0**

The carry which is transferred from fractional part to integer part is called *auxiliary carry* .

The Sum after adding 1101.101 and 0101.111 is 10011.100

Octal Number Addition

Octal Number Addition

Octal numbers can be added in two ways:

Method 1 : Add the numbers same way as the decimal addition is performed and if after addition the sum is greater than 7 then add 2 to it, to get the final sum in octal number.

Method 2 : Convert the octal digits into three bit binary number and then add them as we add the numbers in binary. Group the sum bits in to the groups of three bits. These three bit numbers are finally converted into octal digits.

Add the octal numbers $(23.14)_8+(17.71)_8$.

Let A_1 : *23.14* ,A_2 : *17.71*

Method1:

Carry	:	1	1			
A_1	:	2	3	.	1	4
A_2	:	1	7	.	7	1
Sum>7	:		11		8	5
Adding 2 to	:		+		+	
Sum	:		2		2	
		4	13	.	10	5
Final Sum	:	4	3	.	0	5

Thus the Octal Sum is $(43.05)_8$

Method2: $A_1 : (23.14)_8 : (010\ 011.001\ 100)_2$

$A_2 : (17.71)_8 : (001\ 111.111\ 001)_2$

Carry	:	1	1	1	1	1	1		1	1			
A_1	:	0	1	0	0	1	1	.	0	0	1	1	0	0
A_2	:	0	0	1	1	1	1	.	1	1	1	0	0	1
Binary Sum	:	1	0	0	0	1	1		0	0	0	1	0	1
Octal Sum	:		**4**			**3**		.		**0**			**5**	

Thus the Octal Sum is $(43.05)_8$

Hexadecimal Number Addition

Method 1 : **Add the numbers as the decimal addition and if after addition the SUM is greater than 15, subtract 6 to it to get the final SUM in hexadecimal number.**

Method 2: Convert the each hexadecimal digit into four bit binary number, add these binary numbers and then make the group of four bits of the SUM bits and convert them into hexadecimal digits

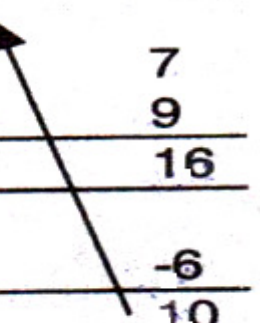
Cont……

Add the hexadecimal numbers $(37)_{16}+(49)_{16}$.

Let A_1 : *37* ,A_2 : *49*

Method1:

Carry	:	**1**	
A_1	:	**3**	**7**
A_2	:	**4**	**9**
Sum	:		**16**
Sum>15			
subtracting 6	:		**-6**
			10
Final Sum	:	**8**	**0**



Thus the hexadecimal Sum is $(80)_{16}$

Method 2: $(37)_{16}$: $(0011\ \ 0111)_2$

$(49)_{16}$: $(0100\ \ 1001)_2$

Carry	:	**1**	**1**	**1**	**1**	**1**	**1**	**1**	
A_1	:	0	0	1	1	0	1	1	1
A_2	:	0	1	0	0	1	0	0	1
Binary Sum	:	1	0	0	0	0	0	0	0
Hexadecimal Sum	:		**8**				**0**		

Thus the hexadecimal Sum is $(80)_{16}$

Cont.....

Add the hexadecimal numbers $(57.82)_{16}+(A6.43)_{16}$.

Let A_1 : *57.82* ,A_2 : *A6.43*

Method1:

Carry								
A_1	:	5	7	.	8	2		
A_2	:	10	6	.	4	3		
Sum 1	:	15	13	.	12	5		
		↓	↓		↓	↓		
Final Sum	:	**F**	**D**	.	**C**	**5**		

Thus the hexadecimal Sum is $(FD.C5)_{16}$

Method2 : *$(57.82)_{16}$: $(0101\ 0111\ .\ 1000\ 0010)_2$*

$(A6.43)_{16}$: $(1010\ 0110\ .\ 0100\ 0011)_2$

Carry	:				1	1								1		
A_1	:	0	1	0	1	0	1	1	1	.	1	0	0	0	0	1	0
A_2	:	1	0	1	0	0	1	1	0	.	0	1	0	0	0	1	1
Binary Sum	:	1	1	1	1	1	1	0	1	.	1	1	0	0	0	1	0	1
Hexadecimal Sum	:		**F**			**D**				.		**C**				**5**	

Thus the hexadecimal Sum is $(FD.C5)_{16}$

Binary Number Substation

Binary Number Subtraction

Binary subtraction is done in similar manner as the decimal subtraction. The four basic cases of binary subtraction are:

$0 - 0 = 0$
$0 - 1 = 1$ (Borrow 1 from next higher bit)
$1 - 0 = 1$
$1 - 1 = 0$

In the second case bigger number(1), is subtracted from the smaller one $(0)_2$ for this borrow has to be taken from the next higher bit this makes it $(10)_2$, now $(10)_2 - (1)_2 = (1)_2$ (i.e. 2 -1= 1 in decimal).

Cont……

Subtract 100111 from 110010

Let A = *110010* and B = *100111*

Let us onece again understand the binary subtraction in steps :

Borrow						
A	1	1	0	0	0~~1~~	10
B	1	0	0	1	1	1
Difference (A-B)						1

Now the subtraction numbers look as :

Borrow						
A	1	1	0	0	0	10
B	1	0	0	1	1	1
Difference (A-B)						1

Taking borrow from next 1 will change the numbers as:

A	1	~~1~~0	10	0	0	10
B	1	0	0	1	1	1
Difference (A-B)						1

A	1	0	~~1~~0 1	10	0	10
B	1	0	0	1	1	1
Difference (A-B)						1

A	1	0	1	~~1~~0 1	10	10
B	1	0	0	1	1	1
Difference (A-B)						1

Cont.....

After taking the borrow the final subtraction is

A	1	0	1	1	10	10
B	1	0	0	1	1	1
Difference (A-B)	0	0	1	0	1	1

The complete borrowing process can be summarized as

Borrow						
A	1	~~1~~ 0	~~0~~ 1	~~0~~ 1	10	10
B	1	0	0	1	1	1
Difference (A-B)	0	0	1	0	1	1

Thus the binary difference is 1011

Cont……

Subtract 100111.111 from 111011.101

Let A = *111011.101* and B = *100111.111*

Final Subtraction becomes

A	1	1	**0**	1	**10✗**	**10✗**	.	**10✗**	10	1
B	1	0	0	1	1	1	.	1	1	1
Difference (A-B)	**0**	1	**0**	**0**	1	1	.	1	1	**0**

Thus the binary difference is 10011.110

Octal Number Substation

Octal number subtraction follow the same procedure as the decimal, but there is a slight difference, When a subtraction performed with the help of borrow in Octal, then 2 is subtracted from the difference to get the octal number.

Subtract $(25)_8$ *from* $(71)_8$

Let A = 71 and B = *25*

A	:	7	1
B	:	2	5

5 cannot be subtracted from 1, taking carry from the next higher number

A	:	~~7~~6	11
B	:	2	5
Difference (A-B)	:		6

Since subtraction is done with the help of borrow therefore subtracting 2 from the number .

A	:	~~7~~6	11
B	:	2	5
Subtracting 2			6
from 6	:		-2
Difference (A-B)	:	4	4

Thus the difference is $(44)_8$

Hexadecimal Number Substation

If borrow is taken for the subtraction then after subtraction add 6 to the number

Subtract $(35)_{16}$ *from* $(A4)_{16}$

Let A = 35 and B = *A4 : 104*

A	:	~~10~~ **9**	14
B	:	3	5
$(A-B)_{10}$	:		9
Borrow has been			+6
taken , adding 6	:		15
			↓
$(A-B)_{16}$	:	6	F

Thus the difference is $(6F)_{16}$

Binary Multiplications

Binary Multiplication

Multiplication in binary is similar to decimal multiplication. Four basic cases of binary multiplication are:

0 X 0 = 0

0 X 1 = 0

1 X 0 = 0

1 X 1 = 1

Multiply $(110)_2$ *and* $(101)_2$

Let A:110 and B: 101

A: Multiplicand				1	1	0
B: Multiplier			x	1	0	1
First Multiplier bit is 1 copy multiplicand				1	1	0
Next multiplier bit is 0 copy 0's			0	0	0	x
Last Multiplier bit is 1 copy multiplicand		1	1	0	x	x
		1	1	1	1	0

Thus $(110)_2$ *X* $(101)_2 = (11110)_2$

Binary Division

It is no different to the decimal division and require no explanation. The binary division is illustrated through the examples.

Divide $(1010)_2$ *by* $(10)_2$

Divider: $(10)_2$ Dividend: $(1010)_2$

$$10\overline{)1010}\,(101$$
$$\underline{10}$$
$$XX10$$
$$\underline{10}$$
$$XX$$

The Quotient is $(101)_2$

1's & 2's Compliment

1's Compliment

1's compliment of any binary number can be found by converting 1's into 0's and 0's into 1's

Example Find the 1's compliment of the number 110111011101.

Solution:

1	1	0	1	1	1	0	1	1	1	0	1
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0	0	1	0	0	0	1	0	0	0	1	0

2's Compliment

The 2's compliment of any number can be calculated by , first converting the binary number into its 1's compliment and then adding 1 to the least significant bit (LSB) of 1's compliment number.

Cont……

Example Find the 2's compliment of the number a) 11011100 b) 00101111

Solution: **(a)**

Number	1	1	0	1	1	1	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
1's Compliment	0	0	1	0	0	0	1	1
Adding 1 to LSB							+	1
2's Compliment	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**

(b)

Number	0	0	1	0	1	1	1	1
	↓	↓	↓	↓	↓	↓	↓	↓
1's Compliment	1	1	0	1	0	0	0	0
Adding 1 to LSB							+	1
2's Compliment	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**

Subtraction Using 2's Compliment

Subtraction in computer is done through addition, and not only subtraction the multiplication, division is also done using addition. The number which has to be subtracted its 2's compliment is found out and added to the number from which the number has to be subtracted. After the addition if carry results discard the carry as this is positive difference but if no carry results after the addition then the difference is negative then we find its 2's compliment to get this negative number.

Illustration:

A
-B

C

- Find the 2's compliment of B
- Add the 2's compliment of B to A
- After Sum, if Carry Results: Discard carry and the difference is positive number
- After Sum, if No Carry Results: Find the 2's compliment of the difference, the result is negative number.

Cont.....

Example : Subtract :

a) 10 from 14 in decimal.

b) 10 from 14 in binary using 2's compliment method.

Solution :

a) Subtracting 10 from 14 (i.e 14 – 10)

Decimal Subtraction

A		1	4
B	-	1	0
Difference		0	4

2's Compliment Subtraction

A		1	1	1	0
B		1	0	1	0
2's compliment of B		0	1	1	0
A		1	1	1	0
Adding	1	0	1	0	0
Discarding Carry, positive difference			1	0	0

The difference is $(4)_{10} = (100)_2$

b) Subtracting 14 from 10 (i.e 10 – 14)

Decimal Subtraction

A		1	0
B	-	1	4
Difference	-	0	4

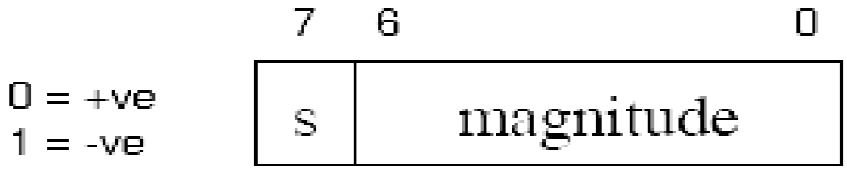
2's Compliment Subtraction

A	1	0	1	0
B	1	1	1	0
2's compliment of B	0	0	1	0
A	1	0	1	0
Adding	1	1	0	0
No carry, finding 2's compliment	0	1	0	0
Negative Difference	–	1	0	0

The difference is $(-4)_{10} = (1100)_2$, here 1 at MSB represents that the number is negative . To obtain the binary number equivalent to decimal, we find its 2's compliment and placing negative sign before it , the number is $(-100)_2$.

Signed Numbers Basics

- So far, numbers are assumed to be unsigned (i.e. positive)
- How to represent signed numbers?
- Solution 1: **Sign-magnitude -** Use one bit to represent the **sign**, the remain bits to represent **magnitude**



+27 = 0001 1011$_b$
-27 = 1001 1011$_b$

  - Problem: need to handle sign and magnitude separately.
- Solution 2: **One's complement** - If the number is negative, invert each bits in the magnitude

+27 = 0001 1011$_b$
-27 = 1110 0100$_b$

  - Not convenient for arithmetic - add 27 to -27 results in 1111 1111$_b$
  - Two zero values

Sign and magnitude bits should be differently treated in arithmetic operations.

"Zero" has two representations:

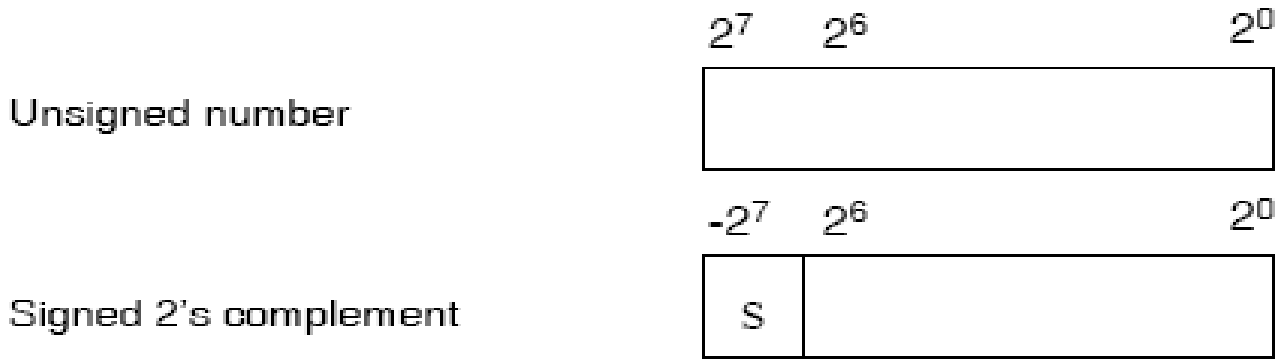
$+ 0_{ten} = 00000000_{two}$

$- 0_{ten} = 10000000_{two}$

2's Complement

- Solution 3: **Two's complement -** represent negative numbers by taking its magnitude, invert all bits and add one:

Positive number	**+27 = 0001 1011$_b$**
Invert all bits	**1110 0100$_b$**
Add 1	**-27 = 1110 0101$_b$**



$$x = -b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + \bullet\bullet\bullet + b_1 2^1 + b_0 2^0$$

Sign Extension

- Sometimes we need to extend a number into more bits
- Decimal
  - converting 12 into a 4 digit number gives 0012
  - we add 0's to the left-hand side
- Unsigned binary
  - converting 0011 into an 8 bit number gives 00000011
  - we add 0's to the left-hand side
- For signed numbers we duplicate the sign bit (MSB)
- Signed binary
  - converting 0011 into 8 bits gives 00000011 (duplicate the 0 MSB)
  - converting 1011 into 8 bits gives 11111011 (duplicate the 1 MSB)
  - Called "**Sign Extension**"

Radix Complement Representation

The radix complement of an n digit number D is obtained by subtracting it from r^n.

Number	10's Complement	9s' Complement
1849	8151	8150
2067	7933	7932
100	9900	9899
7	9993	9992
8151	1849	1848
0	10000 (= 0)	9999

Cont.....

Signed and Unsigned Numbers

Example : *(i) Subtract $(2)_{10}$ from $(5)_{10}$ using 2's compliment method.*

(ii) Subtract $(5)_{10}$ from $(2)_{10}$ using 2's compliment method.

Solution: (i) Let: A = 5 , B = 2

(i) ***Un Signed Number***

		Number		
A	5 =	1	0	1
B	2=	0	1	0
2'scompliment of 2	-2=	1	1	0
Adding A	5=	1	0	1
Sum		10	1	1
Discarding Carry	+3=	0	1	1

The difference of unsigned binary number 010 from 101 is 011

Signed Number

		Sign Bit	Number		
A	5=	0	1	0	1
B	2=	0	0	1	0
2'scompliment of 2	-2=	1	1	1	0
Adding A	5=	0	1	0	1
Sum		10	0	1	1
Discarding Carry	+3=	0	0	1	1

The difference of signed binary number 0010 from 0101 is 0011

Cont.....

(ii) Let: A = 2 , B = 5

Here we have to subtract a bigger number (5) from a smaller number (2) resulting into negative difference (- 3)

Un Signed Number

		Number		
A	2=	0	1	0
B	5=	1	0	1
2'scompliment of 5	-5=	0	1	1
Adding A	2=	0	1	0
Difference, no carry find its 2's compliment		1	0	1
Difference	**-3=**	-0	1	1

The difference of unsigned binary number 101 from 010 is -011

Signed Number

		Sign Bit	Number		
A	2	0	0	1	0
B	5	0	1	0	1
2'scompliment of 5	-5	1	0	1	1
Adding A	2	0	0	1	0
Difference	**-3**	1	1	0	1

The difference of signed binary number 0 101 from 0 010 is 1 101,

The number 1 101 represents -3 , the number can be calculated as:

1	1	0	1	
-8	+4	+0	+1	- 8+5= -3