

# Digital Logic Design + Computer Architecture

Sayandeep Saha

Assistant Professor  
Department of Computer  
Science and Engineering  
Indian Institute of Technology  
Bombay





# Number Systems and Codes



# Baby Step

- **Let's go back to the days when we were 2 years old...**
  - Learning numbers again....but in a new way...



# Numbers in Computing

- We normally use the decimal number system.
- But computers understand only bits...
- How to compute on bits??

# Generalization of Number Representation

- Numbers are represented in a “base”.
- Decimal (Base 10):  $953.78_{10} = 9 \times 10^2 + 5 \times 10^1 + 3 \times 10^0 + 7 \times 10^{-1} + 8 \times 10^{-2}$
- Binary (Base 2):  $1011.11_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$   
 $= 8 + 0 + 2 + 1 + \frac{1}{2} + \frac{1}{4} = 11\frac{3}{4} = 11.75_{10}$
- The general case — base  $b$  number:  
 $(N)_b = a_{q-1}b^{(q-1)} + a_{q-2}b^{(q-2)} + \dots + a_2b^2 + a_1b^1 + a_0b^0 + a_{-1}b^{-1} + \dots a_{-p}b^{-p}, \quad 0 \leq a_i < b, b > 1$ 
  - $a_{(q-1)}$  is called the **Most Significant Digit (MSD)**
  - $a_{(-p)}$  is called the **Least Significant Digit (MSD)**
  - $a_{(-1)} — a_{(-p)}$  are digits in the **fractional part**.

# Generalization of Number Representation

	<i>Base</i>				
	2	4	8	10	12
0000	0	0	0	0	0
0001	1	1	1	1	1
0010	2	2	2	2	2
0011	3	3	3	3	3
0100	10	4	4	4	4
0101	11	5	5	5	5
0110	12	6	6	6	6
0111	13	7	7	7	7
1000	20	10	8	8	8
1001	21	11	9	9	9
1010	22	12	10	$\alpha$	
1011	23	13	11	$\beta$	
1100	30	14	12	10	
1101	31	15	13	11	
1110	32	16	14	12	
1111	33	17	15	13	

# Base Conversion

- Octal (b=8) to Decimal (b=10):  $(432.2)_8 = 4 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 + 2 \times 8^{-1} = (282.25)_{10}$
- Binary (base 2) to Decimal (b=10):  $(1010.011)_2 = 2^3 + 2^1 + 2^{-2} + 2^{-3} = (10.375)_{10}$

# Base Conversion: Decimal to...

## To Binary:

2 / 53

2 / 26

2 / 13

2 / 6

2 / 3

2 / 1

0

rem. = 1 = a<sub>0</sub>

rem. = 0 = a<sub>1</sub>

rem. = 1 = a<sub>2</sub>

rem. = 0 = a<sub>3</sub>

rem. = 1 = a<sub>4</sub>

rem. = 1 = a<sub>5</sub>

53<sub>10</sub> = 110101<sub>2</sub>

Convenient way of writing

<u>Integer</u>	<u>Remainder</u>
41	
20	1
10	0
5	0
2	1
1	0
0	1

↑

101001 = answer

## To Octal:

153

19

2

0

|

1

3

2

↑

= (231)<sub>8</sub>



# Base Conversion

- General Rule: for  $b_1 > b_2$

$$(N)_{b_1} = a_{q-1}b_2^{q-1} + a_{q-2}b_2^{q-2} + \cdots + a_1b_2^1 + a_0b_2^0$$

$$\frac{(N)_{b_1}}{b_2} = \underbrace{a_{q-1}b_2^{q-2} + a_{q-2}b_2^{q-3} + \cdots + a_1}_{Q_0} + \frac{a_0}{b_2}$$

$$\left(\frac{Q_0}{b_2}\right)_{b_1} = \underbrace{a_{q-1}b_2^{q-3} + a_{q-2}b_2^{q-4} + \cdots + a_1}_{Q_1} + \frac{a_0}{b_2^2}$$



# Base Conversion: Fractions

To Binary:

	<u>Integer</u>		<u>Fraction</u>	<u>Coefficient</u>
0.6875 × 2 =	1	+	0.3750	$a_{-1} = 1$
0.3750 × 2 =	0	+	0.7500	$a_{-2} = 0$
0.7500 × 2 =	1	+	0.5000	$a_{-3} = 1$
0.5000 × 2 =	1	+	0.0000	$a_{-4} = 1$
Answer: $(0.6875)_{10} = (0.a_{-1}a_{-2}a_{-3}a_{-4})_2 = \underline{(0.1011)_2}$				

- **Important!!!** If a number has both integer and fraction parts, then conversion of these two parts are done separately.
- Simplest way of converting from base  $b_1$  to  $b_2$ , is to convert from  $b_1$  to base 10 and then base 10 base  $b_2$ .

To Octal:

$$\begin{aligned} 0.513 \times 8 &= 4.104 \\ 0.104 \times 8 &= 0.832 \\ 0.832 \times 8 &= 6.656 \\ 0.656 \times 8 &= 5.248 \\ 0.248 \times 8 &= 1.984 \\ 0.984 \times 8 &= 7.872 \end{aligned}$$

$$(0.513)_{10} = (0.406517 \dots)_8$$

$$(N)_{b_1} = a_{-1}b_2^{-1} + a_{-2}b_2^{-2} + \dots + a_{-p}b_2^{-p}$$

$$b_2 \cdot (N)_{b_1} = a_{-1} + a_{-2}b_2^{-1} + \dots + a_{-p}b_2^{-p+1}$$



# Quiz Time

- Convert  $(231.3)_4$  to base 7



# Quiz Time

- Convert  $(231.3)_4$  to base 7

- Ans:

$$(231.3)_4 = 2 \times 4^2 + 3 \times 4^1 + 1 \times 4^0 + 3 \times 4^{-1} = (45.75)_{10}$$

$$\begin{array}{rcl} 45 & & \\ 6 & \text{rem} = 3 & \\ 0 & \text{rem} = 6 & \end{array} \left| \begin{array}{c} \uparrow \\ \longrightarrow \end{array} \right. (63)_7$$

$$\begin{array}{rcl} .75 \times 7 = 5.25 & \text{intpart} = 5 & \\ 0.25 \times 7 = 1.75 & \text{intpart} = 1 & \\ 0.75 \times 7 = 5.25 & \text{intpart} = 5 & \end{array} \longrightarrow (.5151\cdots)_7$$

$$(63.5151\cdots)_7$$




# Quiz Time

- Convert  $(0.625)_{10}$  to base 2



# Quiz Time

- Convert  $(0.625)_{10}$  to base 2

$.625 \times 2 = 1.25$	$intpart = 1$		$(.101)_2$
$0.25 \times 2 = 0.5$	$intpart = 0$		
$0.5 \times 2 = 1.00$	$intpart = 1$		



# Octal and Hexadecimal Numbers

- Base 8 and base 16 are very useful in computing.
- Hex: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Binary to Hex/Octal and vice versa:
  -

$$(673.124)_8 = ( \underbrace{110}_6 \underbrace{111}_7 \underbrace{011}_3 . \underbrace{001}_1 \underbrace{010}_2 \underbrace{100}_4 )_2$$

$$(306.D)_{16} = ( \underbrace{0011}_3 \underbrace{0000}_0 \underbrace{0110}_6 . \underbrace{1101}_D )_2$$

# Binary Arithmetic

<i>Bits</i>		<i>Sum</i>	<i>Carry</i>	<i>Difference</i>	<i>Borrow</i>	<i>Product</i>
<i>a</i>	<i>b</i>	<i>a + b</i>		<i>a - b</i>		<i>a · b</i>
0	0	0	0	0	0	0
0	1	1	0	1	1	0
1	0	1	0	1	0	0
1	1	0	1	0	0	1



# Binary Arithmetic

Binary addition:

1111 = carries of 1

1111.01 =  $(15.25)_{10}$

0111.10 =  $(7.50)_{10}$

---

10110.11 =  $(22.75)_{10}$

Binary subtraction:

1 = borrows of 1

10010.11 =  $(18.75)_{10}$

01100.10 =  $(12.50)_{10}$

---

00110.01 =  $(6.25)_{10}$

# Binary Arithmetic

Binary multiplication:

$$\begin{array}{r} 11001.1 = (25.5)_{10} \\ 110.1 = (6.5)_{10} \\ \hline 110011 \\ 000000 \\ 110011 \\ 110011 \\ \hline 10100101.11 = (165.75)_{10} \end{array}$$

Binary division:

$$\begin{array}{r} 10110 = \text{quotient} \\ 11001 \overline{) 1000100110} \\ \underline{11001} \phantom{0} \\ 00100101 \\ \underline{11001} \phantom{0} \\ 0011001 \\ \underline{11001} \\ 00000 = \text{remainder} \end{array}$$



# Complements

- Simplest way to represent negative numbers.
- The most intuitive way to represent a negative number is to use an extra bit for sign — **sign-magnitude representation**
- But the computation with this representation is little complex, we'll see this later.
- Complements are more handy
- (b-1)'s complement of  $(N)_b$ :  $(b^n - 1) - N$ , where n is the number of digits.
- b's complement of  $(N)_b$ :  $b^n - N$ , where n is the number of digits

# Complements

- Let's say we want to compute  $(1234)_{10} - (110)_{10}$
- First we take the 10's complement of 110:  $10^4 - 110 = 9890$  (Important!!! Number of digits has to be the same)
- Now add:  $1234 + 9890 = 11124$
- $11124 > 10000$ : so there will be an end carry in the addition — just discard the carry
- $11124 \rightarrow 1124$ , which is the answer.
- Now, what is the purpose????



# Complements — in the binary world

- 2's complement of  $N$ :  $2^n - N$
- 1's complement of  $N$ :  $(2^n - 1) - N$
- We have a super easy way to compute these: can you guess why?

# Complements in the binary world

- 2's complement of  $N$ :  $2^n - N$
- 1's complement of  $N$ :  $(2^n - 1) - N$
- We have a super easy way to compute these: can you guess why?
  - Observe that for any  $n$ 
    - $(2^n - 1)$  is basically 1111...  $n$  times.
    - Now if you subtract  $N$ , all the bits of  $N$  are flipped.
    - **Example:** 1111 - 1011 = 0100
  - So, we do not need to do any subtraction really, —just flip the bits.
  - 2's complement = 1's complement + 1



# Complements in the binary world

- Compute  $M - N$
- Step 1: compute 2's complement of  $N$ :  $2^n - N = \text{comp}(N) + 1$
- Step 2:  $M + (2^n - N) = 2^n + (M - N)$
- Now if  $M \geq N$ : the sum will be  $> 2^n$  — so there will be a carry. **Just discard it and output the result.**
- If  $M < N$  the result is  $2^n + (N - M) < 2^n$ . This is basically the 2's complement of  $(N - M)$ . No carry will be produced. **The output will be 2's complement of the result, with a negative sign in the front.**

# Complements in the binary world

- Let  $X = 1010100$ ,  $Y = 1000011$ ; compute  $X - Y$  and  $Y - X$ .



# Complements in the binary world

- Let  $X = 1010100$ ,  $Y = 1000011$ ; compute  $X - Y$  and  $Y - X$ .

$$\begin{array}{r} X = 1010100 \\ 2\text{'s complement of } Y = + \underline{0111101} \\ \text{Sum} = 10010001 \\ \text{Discard end carry } 2^7 = - \underline{10000000} \\ \text{Answer: } X - Y = 0010001 \end{array}$$

$$\begin{array}{r} Y = 1000011 \\ 2\text{'s complement of } X = + \underline{0101100} \\ \text{Sum} = 1101111 \end{array}$$

There is no end carry.

$$\text{Answer: } Y - X = -(2\text{'s complement of } 1101111) = -0010001$$

- Important!!!** We are doing unsigned subtraction!!
- Food of thought:** we can do the same with 1's complement too!!, then why 2's complement???



# Complements in the binary world

- We can do the same with 1's complement too!!, then why 2's complement???

$+N$	Positive Integers (all systems)	$-N$	Negative Integers		
			Sign and Magnitude	2's Complement $N^*$	1's Complement $\bar{N}$
+0	0000	-0	1000	—	1111
+1	0001	-1	1001	1111	1110
+2	0010	-2	1010	1110	1101
+3	0011	-3	1011	1101	1100
+4	0100	-4	1100	1100	1011
+5	0101	-5	1101	1011	1010
+6	0110	-6	1110	1010	1001
+7	0111	-7	1111	1001	1000
		-8	—	1000	—

- For signed magnitude, we use 4 bits to represent  $[-7,7]$  with the 4th bit being the sign bit



# Complements in the binary world

- We can do the same with 1's complement too!!, then why 2's complement???

$+N$	Positive Integers (all systems)	$-N$	Negative Integers		
			Sign and Magnitude	2's Complement $N^*$	1's Complement $\bar{N}$
+0	0000	-0	1000	—	1111
+1	0001	-1	1001	1111	1110
+2	0010	-2	1010	1110	1101
+3	0011	-3	1011	1101	1100
+4	0100	-4	1100	1100	1011
+5	0101	-5	1101	1011	1010
+6	0110	-6	1110	1010	1001
+7	0111	-7	1111	1001	1000
		-8	—	1000	—

- For 2's complement, we still use 4 bits to represent  $[-7,7]$ ; but here the encoding is different.
- 1's complement has a negative 0

# Complements in the binary world

System	Range
Unsigned	$[0, 2^N - 1]$
Sign/Magnitude	$[-2^{N-1} + 1, 2^{N-1} - 1]$
Two's Complement	$[-2^{N-1}, 2^{N-1} - 1]$



**Thank you**