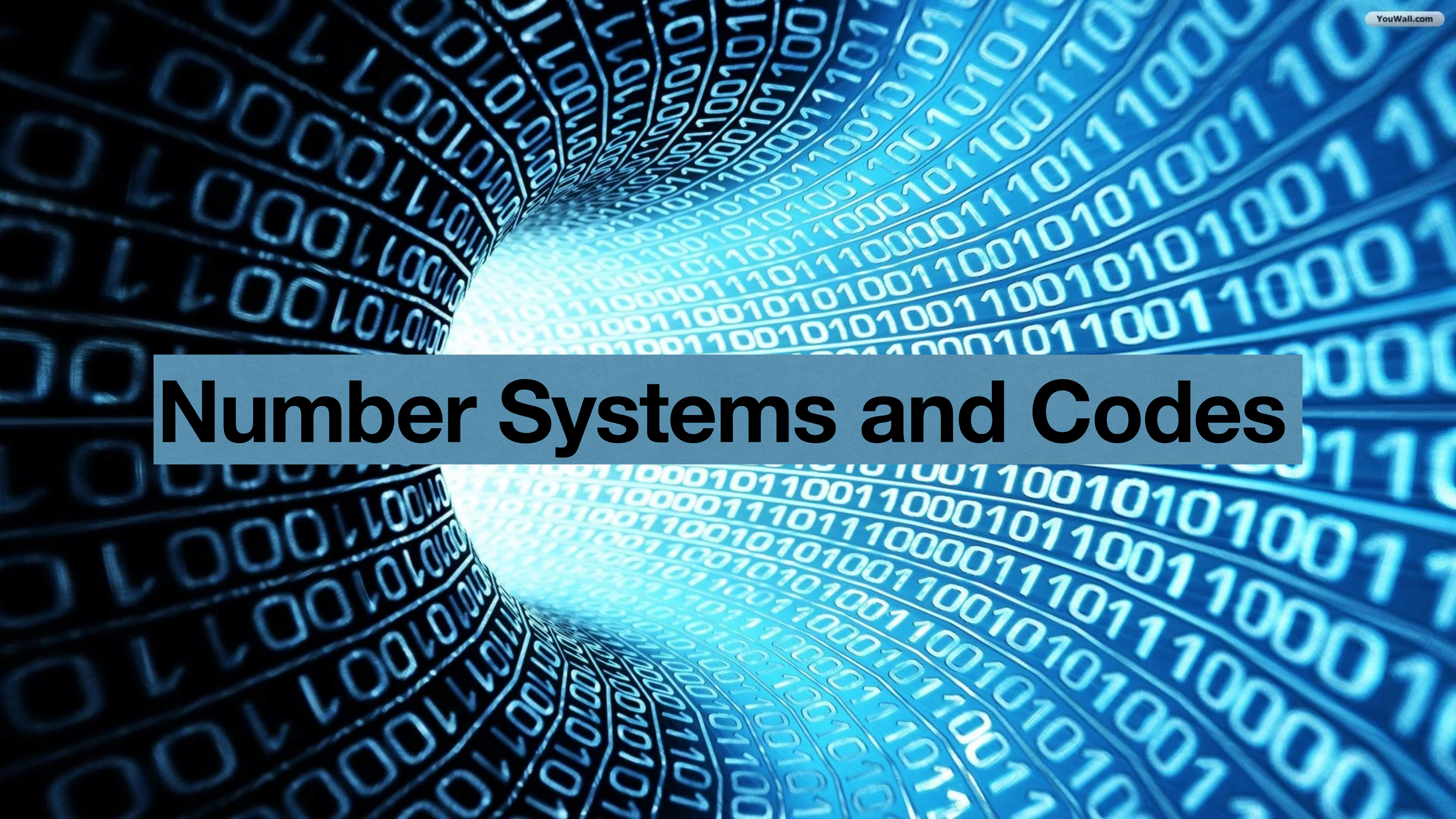
Digital Logic Design + Computer Architecture

Sayandeep Saha

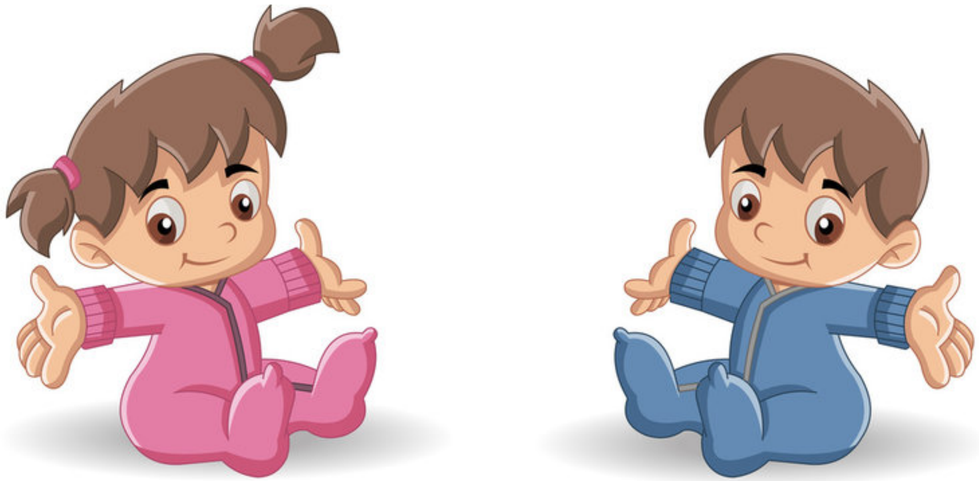
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Bombay

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):

$$1011.11_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$
$$= 8 + 0 + 2 + 1 + \tfrac{1}{2} + \tfrac{1}{4} = 11\tfrac{3}{4} = 11.75_{10}$$

- The general case — base b number:
$(N)_b = a_{q-1}b^{(q-1)} + a_{q-2}b^{(q-2)} + \cdots + a_2b^2 + a_1b^1 + a_0b^0 + a_{-1}b^{-1} + \cdots a_{-p}b^{-p}, \quad 0 \le a_i < b, b > 1$
  - $a_{(q-1)}$ 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

	Base			
2	4	8	10	12
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	10	4	4	4
0101	11	5	5	5
0110	12	6	6	6
0111	13	7	7	7
1000	20	10	8	8
1001	21	11	9	9
1010	22	12	10	α
1011	23	13	11	β
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): $(1010.011)_2 = 2^3 + 2^1 + 2^{-2} + 2^{-3} = (10.375)_{10}$
- General Rule: for $b_1 > b_2$

$$(N)_{b_1} = a_{q-1}b_2^{q-1} + a_{q-2}b_2^{q-2} + \cdots + a_1 b_2^1 + a_0 b_2^0$$

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

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

Base Conversion: Decimal to…

To Binary:

$$
\begin{array}{lll}
2\,\big/\,53 & & \\
2\,\big/\,26 & \text{rem.} = 1 = a_0 & \\
2\,\big/\,13 & \text{rem.} = 0 = a_1 & \\
2\,\big/\,6 & \text{rem.} = 1 = a_2 & 53_{10} = 110101_2 \\
2\,\big/\,3 & \text{rem.} = 0 = a_3 & \\
2\,\big/\,1 & \text{rem.} = 1 = a_4 & \\
0 & \text{rem.} = 1 = a_5 &
\end{array}
$$

Convenient way of writing

Integer	Remainder
41	
20	1
10	0
5	0
2	1
1	0
0	1

101001 = answer

To Octal:

$$
\begin{array}{r|l}
153 & \\
19 & 1 \\
2 & 3 \\
0 & 2
\end{array}
\quad = (231)_8
$$

Base Conversion: Fractions

To Binary:

	Integer		Fraction	Coefficient
$0.6875 \times 2 =$	1	+	0.3750	$a_{-1} = 1$
$0.3750 \times 2 =$	0	+	0.7500	$a_{-2} = 0$
$0.7500 \times 2 =$	1	+	0.5000	$a_{-3} = 1$
$0.5000 \times 2 =$	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:

$0.513 \times 8 = 4.104$

$0.104 \times 8 = 0.832$

$0.832 \times 8 = 6.656$

$0.656 \times 8 = 5.248$

$0.248 \times 8 = 1.984$

$0.984 \times 8 = 7.872$

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

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

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

Quiz Time

- Convert $(231.3)_4$ to base 7

Quiz Time

- Convert $(231.4)_4$ to base 7
- <u>Ans:</u>

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

$$\begin{array}{ll} 45 & \\ 6 & rem = 3 \\ 0 & rem = 6 \end{array} \longrightarrow (63)_7$$

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

$$\boxed{(63.5151\cdots)_7}$$

Quiz Time

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

Quiz Time

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

$$
\begin{aligned}
.625 \times 2 &= 1.25 \quad & intpart &= 1 \\
0.25 \times 2 &= 0.5 \quad & intpart &= 0 \\
0.5 \times 2 &= 1.00 \quad & intpart &= 1
\end{aligned}
\quad \longrightarrow \quad (.101)_2
$$

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:
-

$$(673.124)_8 = (\underbrace{110}_{6}\ \underbrace{111}_{7}\ \underbrace{011}_{3}\ .\ \underbrace{001}_{1}\ \underbrace{010}_{2}\ \underbrace{100}_{4})_2$$

$$(306.D)_{16} = (\underbrace{0011}_{3}\ \underbrace{0000}_{0}\ \underbrace{0110}_{6}\ .\ \underbrace{1101}_{D})_2$$

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

Bits		*Sum*	*Carry*	*Difference*	*Borrow*	*Product*
a	b	$a+b$		$a-b$		$a \cdot b$
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:

$$
\begin{array}{rl}
1111 & = \text{carries of } 1 \\
1111.01 & = (15.25)_{10} \\
0111.10 & = (\ \ 7.50)_{10} \\
\hline
10110.11 & = (22.75)_{10}
\end{array}
$$

Binary subtraction:

$$
\begin{array}{rl}
1 & = \text{borrows of } 1 \\
10010.11 & = (18.75)_{10} \\
01100.10 & = (12.50)_{10} \\
\hline
00110.01 & = (\ \ 6.25)_{10}
\end{array}
$$

Binary Arithmetic

Binary multiplication:

```
      11001.1 = (25.5)10
         110.1 = (  6.5)10
      ---------------------
      110011
     000000
    110011
   110011
   ----------------------
  10100101.11 = (165.75)10
```

Binary division:

```
                 10110 = quotient
11001 | 1000100110
        11001
        ----------
        00100101
           11001
        ----------
          0011001
             11001
        ----------
             00000   = remainder
```

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'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 —> 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: $2^n - N = comp(N) + 1$
- 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.

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

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

There is no end carry.

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

- 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

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

$X - Y = 1010100 - 1000011$

$$
\begin{array}{rr}
X = & 1010100 \\
\text{1's complement of } Y = & +\ \underline{0111100} \\
\text{Sum} = & 10010000 \\
\text{End-around carry} & \longrightarrow\ \underline{+\ 1} \\
\text{Answer: } X - Y = & 0010001
\end{array}
$$

$Y - X = 1000011 - 1010100$

$$
\begin{array}{rr}
Y = & 1000011 \\
\text{1's complement of } X = & +\ \underline{0101011} \\
\text{Sum} = & 1101110
\end{array}
$$

There is no end carry.

Answer: $Y - X = -(\text{1's complement of } 1101110) = -0010001$

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	Negative Integers: 2's Complement N^*	Negative Integers: 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	Negative Integers: 2's Complement N^*	Negative Integers: 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

Signed Binary Numbers

- Leftmost bit is the sign bit — 0 implies positive number and 1 implies negative number
- **Signed magnitude** representation of -9: 10001001
- **Signed 2's complement representation**: 11110111 — take 2's complement of the positive number including the signed bit.
- Signed 2's complement is generally used for computer arithmetic

Signed Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Signed Binary Numbers

- The addition of two signed binary numbers with negative numbers represented in signed- 2's-complement form is obtained from the addition of the two numbers, including their sign bits. **A carry out of the sign-bit position is discarded**

- **Subtraction:** Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including the sign bit). A carry out of the sign-bit position is discarded.

- Therefore, computers need only one common hardware circuit to handle both signed and unsigned arithmetic.

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

Binary Codes

- Solves our problems of interpretation.
- **How to represent a decimal digit with bits?**
- Several encoding techniques can be used
- Note that we have 10 digits — what is the minimum number of bits we need?

Binary Codes

- Solves our problems of interpretation.
- **How to represent a decimal digit with bits?**
- Several encoding techniques can be used
- Note that we have 10 digits — what is the minimum number of bits we need? — 4
- But we can represent total 16 digits with 4 bits, so many different encodings are possible.
- **Weighted code**: If x_1, x_2, x_3, x_4 are the binary digits, with weights w_1, w_2, w_3, w_4, then the decimal digit is:

  $$N=w_4x_4+w_3x_3+w_2x_2+w_1x_1$$

  We say, the sequence (x_1, x_2, x_3, x_4) denotes the code word for N.

Binary Codes

Decimal digit	$w_4w_3w_2w_1$											
	8	4	2	1	2	4	2	1	6	4	2	−3
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	1	0	1
2	0	0	1	0	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1	1	0	0	1
4	0	1	0	0	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1	1	0	1	1
6	0	1	1	0	1	1	0	0	0	1	1	0
7	0	1	1	1	1	1	0	1	1	1	0	1
8	1	0	0	0	1	1	1	0	1	0	1	0
9	1	0	0	1	1	1	1	1	1	1	1	1

BCD

Self-complementing Codes

Self-complementing code: Code word of 9's complement of N obtained by interchanging 1's and 0's in the code word of N

Binary Codes

Decimal digit	*Excess-3*				*Cyclic*			
0	0	0	1	1	0	0	0	0
1	0	1	0	0	0	0	0	1
2	0	1	0	1	0	0	1	1
3	0	1	1	0	0	0	1	0
4	0	1	1	1	0	1	1	0
5	1	0	0	0	1	1	1	0
6	1	0	0	1	1	0	1	0
7	1	0	1	0	1	0	0	0
8	1	0	1	1	1	1	0	0
9	1	1	0	0	0	1	0	0

Add 3 to BCD

Successive code words differ in only one digit

Binary Codes

Decimal digit	*Excess-3*				*Cyclic*			
0	0	0	1	1	0	0	0	0
1	0	1	0	0	0	0	0	1
2	0	1	0	1	0	0	1	1
3	0	1	1	0	0	0	1	0
4	0	1	1	1	0	1	1	0
5	1	0	0	0	1	1	1	0
6	1	0	0	1	1	0	1	0
7	1	0	1	0	1	0	0	0
8	1	0	1	1	1	1	0	0
9	1	1	0	0	0	1	0	0

Add 3 to BCD

Successive code words differ in only one digit

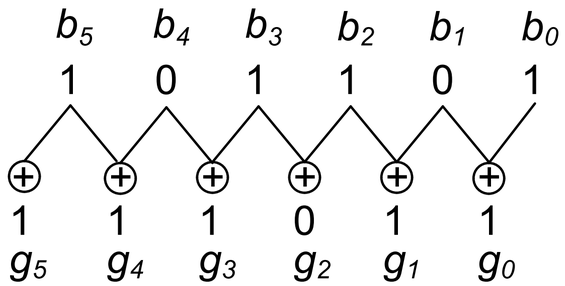
Binary Codes

Decimal number	*Gray* g_3	g_2	g_1	g_0	*Binary* b_3	b_2	b_1	b_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

Example:

Binary:

Gray:



Gray-to-binary:

- $b_i = g_i$ if no. of 1's preceding g_i is even
- $b_i = g_i'$ if no. of 1's preceding g_i is odd

Thank you