Chapter 1

CSCI-1510-003

What is a Number?

- An expression of a numerical quantity
- A mathematical quantity
- Many types:
  - Natural Numbers
  - Real Numbers
  - Rational Numbers
  - Irrational Numbers
  - Complex Numbers
  - Etc.

Numerical Representation

- A quantity can be expressed in several ways
  - XIV
  - 1110_2
  - E_{16}
  - 16_8
- All of these are symbols used to represent a quantity or value.

$$14_{10}$$

The base

What is the Base or Radix?

- The cardinality of the set of symbols in a number system
  - *Cardinality* – The number of elements in a given set.
- The value of the highest symbol is always **one less** than the base.
  - Base 10: S = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- Denoted with a subscript
  - 1110_2
  - E_{16}
  - 16_8

Examples

- The base determines the set of symbols
  - Base 10: S = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - Base 8: S = {0, 1, 2, 3, 4, 5, 6, 7}
  - Base 5: S = { 0, 1, 2, 3, 4}
  - Base 2: S = {0, 1}
  - Base 16: S = ?
    - Borrow the needed digits from the alphabet, so

      S = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Why do we care?

- Without knowing what base you are working in, there is no way to know what quantity is being enumerated.
  - 11 could mean 11_{10},
    $11_2 = 3_{10}$,
    $11_5 = 6_{10}$
  - 15 could mean 15_{10},
    $15_8 = 13_{10}$,
    $15_{16} = 21_{10}$

Elementary School Flashback

- Why is the 1’s column the 1’s column (base 10)?
- The 10’s the 10’s?
- Etc
- Everything to do with the base raised to a power.

Positional Number Representation
Base 10

Remember Expanded Notation?

$$a_4 \text{ x } 10^4 + a_3 \text{ x } 10^3 + a_2 \text{ x } 10^2 + a_1 \text{ x } 10^1 + a_0 \text{ x } 10^0$$

Also known as positional notation.

Example

$7{,}392_{10}$

$7 \times 1000 + 3 \times 100 + 9 \times 10 + 2 \times 1$

$7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$

$$a_n r^n + a_{n-1} r^{n-1} + \cdots + a_2 r^2 + a_1 r^1 + a_0 r^0 .\ a_{-1} r^{-1} + a_{-2} r^{-2} + \cdots + a_{-m} r^{-m}$$

Where:

r is the radix or base.

n is the number of digits to the left of the decimal point

m is the number of digits to the right of the decimal point

Why do we care?

- Nice way to convert from any base to decimal.
- Ex: 1011_2

  $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

  $1 \times 8 + 0 \text{ X } 4 + 1 \times 2 + 1 \times$ 1

  $8 + 2 + 1$

  11_{10}

Using Positional Notation

Ex : 75_8

Ex: $A2E_{16}$

Ex : 32.14_8

Some terminology

- Bits – digits in a binary number
- Byte – 8 bits
- Base 2 - Binary
- Base 8 – Octal
- Base 10 - Decimal
- Base 16 – Hexadecimal or Hex
- Base 32 – Duotrigesimal

Base conversion

- Decimal to any base.
- Use division
  - Whole numbers
  - Left of the decimal point
- Use multiplication
  - Fractional part of a number
  - Right of the decimal point

- Ex: Convert 13_{10} to base 2

$13_{10} = 1101_2$

```
      0 r 1
2 ) 1 r 1
2 ) 3 r 0
2 ) 6 r 1
2 ) 13
```

- How to check your answer

Checking your work

- Scientific notation can be how

Decimal to Binary Conversion

- Convert 857_{10} to base 2.

	0	r 1	MSB
	) 1	r 1	
	) 3	r 0	
	) 6	r 1	
	) 13	r 0	
	) 26	r 1	
	) 53	r 1	
	) 107	r 0	
	) 214	r 0	
	) 428	r 1	LSB
2	) 857_{10}		

1101011001_2

Check the result

- Use Positional Notation

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1
1	1	0	1	0	1	1	0	0	1

$$512 + 256 + 64 + 16 + 8 + 1 = 857_{10}$$

Octal

$$
\begin{array}{rrl}
 & & 0 \text{ r } 1 \\
 &) & 1 \text{ r } 5 \\
 &) & 13 \text{ r } 3 \\
 &) & 107 \text{ r } 1 \\
8 &) & 857
\end{array}
$$

$857_{10} = 1531_8$

8^3	8^2	8^1	8^0
512	64	8	1
1	5	3	1

$1 \times 512 + 5 \times 64 + 3 \times 8 + 1 \times 1 = 857_{10}$

Hexadecimal

```
              0 r 3
      )       3 r 5
      )      53 r 9
16    ) 857
```

$857_{10} = 359_{16}$

16^2	16^1	16^0
256	16	1
3	5	9

$3 \times 256 + 5 \times 16 + 1 \times 9 = 857_{10}$

Example

324_{10} to base 5

Ex 1.4

0.6875_{10} to binary

$.6875 \times 2 = 1.3750 = \boxed{1} + .3750$

$.3750 \times 2 = 0.7500 = \boxed{0} + .7500$

$.7500 \times 2 = 1.5000 = \boxed{1} + .5000$

$.5000 \times 2 = 1.0000 = \boxed{1} + .0000$

$0.6875_{10} = 0.1011_2$

Practice

- 0.513_{10} to octal

Powers of 2

- Conversion between binary, octal, and hexadecimal made easy.
- Just regroup and convert
- How to regroup
  - Consider the least number of bits it would take to encode the largest symbol of the new base.

Binary-coded Octal

- 3 bits to encode 7_8
- Why?
- Do the math.

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Binary to Octal

$1101\ 1010\ 1010_2$

$110\quad 110\quad 101\quad 010_2$ ← Binary Coded Octal

$6\quad 6\quad 5\quad 2$

6652_8

- To convert from Octal to Binary, do the same thing in reverse

Practice

- Convert 76_{10} to binary, then take the result and quickly convert to octal. Now use your octal result and convert back to decimal. You should get the original number.

- Do the same thing, with 57_{10}, 438_{10} and 311_{10}

Binary to Hexadecimal

How many bits are needed to represent the largest single cyphe in hexadecimal?

Hint: do the math.

110110101010_2

$1101 \quad 1010 \quad 1010_2$

D A A

DAA_{16}

Binary	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Practice

- Convert 76_{10} to binary, then take the result and quickly convert to hexadecimal. Now use your hexadecimal result and convert back to decimal. You should get the original number.
- Do the same thing, with 57_{10}, 438_{10} and 311_{10}

Octal to Hexadecimal and vice versa

- Almost the same thing.
- Convert to Binary
- Then regroup as necessary.
- Remember you always group from the decimal point to the right or left.

Example

237_8 to Hex

Convert to binary

$$\begin{array}{ccc} 2 & 3 & 7 \\ 010 & 011 & 111_2 \end{array}$$

Regroup

$$0\ 1001\ 1111_2$$

$$9 \quad F_{16}$$

Does the leading zero mean anything?

Why use Hexadecimal and Octal?

- Convenience
  - Ex. 1011011001010000_2
  - Better written as: $1011\ 0110\ 0101\ 0000_2$

    $= B65F_{16}$

Unsigned vs Signed Numbers

- Unsigned
  - All bits are used to show the magnitude of the number.
  - All numbers are considered to be positive
- Signed
  - Positive and Negative

Signed Numbers

- There are three basic ways to designate the sign of a number.
  - Sign and magnitude
  - Radix-1'scomplement
  - Radix complement

Why using complement ?

- simplifying the subtraction operation by adding a complement of that number instead of subtraction for that number

$\mathbf{15_{10} - 4_{10} = 15_{10} - (\text{complement of } 4_{10}) = 11_{10}}$

Sign and Magnitude

- What is taught in school.
- A value with a sign in front of it
- How does it work in Binary?
- Pretty much the same way as Decimal
- By convention a sign bit is used.
  - 0 → positive
  - 1 → negative

a = 1 111 ?

$\Rightarrow$ a = 15_{10} (if unsigned)
$\Rightarrow$ a = -7 (if signed).

Complements of Numbers

- Two basic types
  - Diminished radix complement
    - Defined as: $(r^n - 1) - N$; given a number N in base r having n digits
  - Radix complement
    - Defined as $r^n - N$; given a number $N \neq 0$ in base r having n digits and 0 if $N = 0$.
- **In base 10, we have : 9'complement and 10's complement**
- **In base 2, we have : 1'complement and 2's complement.**

Diminished Radix Complement

- AKA R-1's complement

$$(r^n - 1) - N$$

In base 10: Finding 5 digits the 9's complement of 1357.

We have n = 5; r = 10, N = 1357.

Result = $(10^5 - 1) - 1357$ = 98642

Radix Complement

- Radix complement
  - Defined as $r^n - N$;
- **In base 10: Finding 5 digits the 10's complement of 1357.**
- **Result = $10^5 - 1357$ = 98643**

Diminished Radix Complement

- AKA R-1's complement

  $(r^n - 1) - N$

In base 2: Finding 8 digits the 1's complement of 0110 1100.

We have n = 8; r = 2, N = 0110 1100 (108_{10})

Result = $(2^8 - 1) - 108 = 147_{10}$ = 1001 0011.

(do directly in base 2):

1111 1111 (8 digits 1)

-0110 1100

=1001 0011

- TRICK by "flipping the bits" : 0 -> 1 ; 1->0

Radix Complement

AKA R-1's complement

$(r^n) - N$

In base 2: Finding 8 digits the 2's complement of 0110 1100.

We have n = 8; r = 2, N = 0110 1100 (108_{10})

Result = $(2^8) - 108 = 148_{10}$ = 1001 0100.

(do directly in base 2):

1 0000 0000 (8 digits 1)

-0110 1100

= 0 1001 0100

- TRICK by "flipping the bits" : 0 -> 1 ; 1->0. Then Add 1.

Practice

- **Finding 8 digits the 1's complement of 0111 1000.**

- **Finding 8 digits the 2's complement of 0111 1000.**

Practice

- **Finding 8 digits the 1's complement of 0111 1000.**
  - **Answer: 1000 01111**
- **Finding 8 digits the 2's complement of 0111 1000.**
  - **Answer: 1000 1000**

(R-1)'s complement

- Octal: 8's complement and 7's complement
- Hexadecimal: 16's complement and 15'complements.

R-1's complement : $(r^n - 1) - N$.

Radix complement: $r^n - N$;

Why is this wrong?

- Find the (R-1)'s complement of 56_8

$8^2 = 64$
$64 - 1 = 63$
$63 - 56 = 7$

WRONG
This is base 10 math not base 8.

The Right Way

$$(8_{10})^2 = 64_{10} = 100_8$$

$$100_8 – 1_8 = 77_8$$

$$77_8 - 56_8 = 21_8$$

21_8 is the complement of 56_8

The Right Way

- TRICK: CONVERT TO BINARY. DO TRICK IN BINARY. THEN CONVERT BACK.
- 21_8 = 010 001 => 101 110 = 56_8

Complements Summary

- The complement of the complement returns the original number
- If there is a radix point
  – Calculate the complement as if the radix point was not there.
- **Used in computers to perform subtraction.**

Complements Summary

- 1's complement
  - Is the interim step towards 2's complement
  - Problem: Two values of 0.

+ 0: 0000 0000, let's flip all bits
- 0: 1111 1111.

- 2's complement
  - Most CPU's today use 2's complement.
  - Only one value 0:

0000 000

Rules of Addition

- Binary addition

$0_2 + 0_2 = 0_2$

$0_2 + 1_2 = 1_2 + 0_2 = 1_2$

$1_2 + 1_2 = 10_2$

Notice the carry into the next significant bit.

Sign and Magnitude

- Example

$$\begin{array}{r} 0100\ 1111_2 \\ \underline{+\ 0010\ 0011}_2 \end{array}$$

Sign bit

Carry								
	0	1	0	0	1	1	1	1
	0	0	1	0	0	0	1	1
Result								

Sign and Magnitude

- Example

$$\begin{array}{r} 0100\ 1111_2 \\ +\ \underline{0010\ 0011_2} \end{array} \qquad \begin{array}{r} 79 \\ \underline{+\ 35} \end{array}$$

Carry	0	0	0	1	1	1	1	
	0	1	0	0	1	1	1	1
	0	0	1	0	0	0	1	1
Result	0	1	1	1	0	0	1	0

$= 0111\ 0010_2$ 114

Sign and Magnitude

- Another example

$$\begin{array}{r} 1\ 1111_2 \\ +\ \underline{1\ 1011}_2 \end{array}$$

Carry					
	1	1	1	1	1
	1	1	0	1	1
Result					

= ?

Sign and Magnitude

- Another example

$$
\begin{array}{r}
1\ 1111_2 \\
+\ \underline{1\ 1011}_2
\end{array}
\qquad\qquad
\begin{array}{r}
-15 \\
\underline{+\ (-11)}
\end{array}
$$

Carry	1	1	1	1	
	1	1	1	1	1
	1	1	0	1	1
Result	1	1	0	1	0

$= 1\ 1010_2$ $\qquad$ $-\ 26$

Sign and Magnitude

- And another example ....

$$
\begin{array}{r}
0100\ 1111_2 \\
+\ \underline{0110\ 0011}_2
\end{array}
$$

Carry								
	0	1	0	0	1	1	1	1
	0	1	1	0	0	0	1	1
Result								

= ?

Sign and Magnitude

- Another example ....

$$
\begin{array}{r}
0100\ 1111_2 \\
+\ \underline{0110\ 0011}_2
\end{array}
\qquad
\begin{array}{r}
79 \\
\underline{+\ 99} \\
-50
\end{array}
$$

Carry	1	0	0	1	1	1	1	
	0	1	0	0	1	1	1	1
	0	1	1	0	0	0	1	1
Result	1	0	1	1	0	0	1	0

$= \quad 1011\ 0010_2$

What Happened ???

Should be 178

Sign and Magnitude

- The error is due to overflow
- Caused by a carry out of the MSB of the number to the sign bit.
- Signed magnitude works but….
  - Suffers from limitations
    - Adding positive and negative numbers doesn't always work. (overflow)

Keep in mind

- If you add 2 number of the same sign AND the result is a different sign, you have

OVERFLOW

Rules of Subraction

- Use the fact you can add a negative number to get the same result.

Ex. $15_{10} - 4_{10} = 15_{10} + (-4_{10}) = 11_{10}$

Unsigned math

- Procedure for subtraction of 2 ***unsigned*** numbers.
- Ex: $M - N = x$
  1. Add M to the r's complement of N
  2. If $M \geq N$, an end carry is produced which can be discarded.
     - Then x is positive as indicated by the carry-out
  3. If $M \leq N$
     - Then x is negative and in r's complement form

Complement Arithmetic

- What is a complement?
  - Webster's

    a : something that fills up, completes, or makes perfect.

    b : the quantity, number, or assortment required to make a thing complete
- A way to represent negative numbers.
- Two types
  - R-1's Complement (diminished radix complement)
  - R's Complement (radix complement)

R-1's Complement

- A reminder:
  - $\overline{N} = (r^n - 1) - N$
    - $\overline{N}$ is –N in 1's complement notation
    - where n is the number of bits per word
    - N is a positive integer
    - r is the base

R-1's Complement

- Word Range:
  - The range of numbers that can be represented in a given number of bits.

$$-(r^{n-1}-1) \text{ to } r^{n-1}-1$$

- Ex: What is the range of signed binary numbers that can be represented in 3 bits?
- $-\ 3_{10}$ to 3_{10}

Some Terminology

- Augend, addend, sum

15	Augend	(AKA top number)
+7	Addend	(AKA bottom number)
22	Sum	

- Minuend, subtrahend, difference

15	Minuend	(AKA top number)
- 7	Subtrahend	(AKA bottom number)
8	Difference	

1’s Complement (Subtraction)

- Never complement the top number in a problem.
- Add the 1’s complement of the bottom number to the top number.
  - This will have the same effect as subtracting the original number.
- *If there is a carry-out, <u>end around carry</u> and add back in*.

How to Subtract Numbers

- Step 1: 1’s complement the bottom number.
- Step 2: Do the math.
- Step 3: If there is a carry out, end around carry and add it back in.
- Note: If there is no carry the answer is in 1’s complement form.
  - What does this mean?

Examples

X = 0111   Y = 0101

X – Y

```
  0111        0111
- 0101   →  + 1010
            10001
          +     1
             0010
```

Y - X

```
  0101        0101
- 0111   →  + 1000
              1101
```

Things to notice about 1's complement

- Any negative number will have a leading 1.
- There are 2 representations for 0, 00000 and 11111.
  - Not really a problem, but still have to check for it.
- There is a solution.

R’s Complement Word Range

- A reminder
- $\overline{N} = (r^n) - N$
  - $\overline{N}$ is –N in 2’s complement notation
  - where n is the number of bits per word
  - N is a positive integer
  - r is the base

R's Complement

- Word Range is:

  $-(r^{n-1})$ to $r^{n-1}-1$

Ex: What is the range of signed binary numbers that can be represented in 3 bits?

$(-4_{10}$ to $3_{10})$

2’s Complement Subtraction

- Step 1: 2’s complement the bottom number.
  - Never the top number
- Step 2: Perform the addition
- Step 3: if there is a carry out, ignore it.
  - If a number leads with a 1, it is negative and in 2’s complement form.

Examples

X = 0110 Y = 0101

X – Y

$$\begin{array}{r} 0110 \\ -\ \underline{0101} \end{array} \longrightarrow \begin{array}{r} 0110 \\ +\ \underline{1011} \\ \cancel{1}0001 \end{array}$$

Y - X

$$\begin{array}{r} 0101 \\ -\ \underline{0110} \end{array} \longrightarrow \begin{array}{r} 0101 \\ +\ \underline{1010} \\ 1111 \end{array}$$

Intro to Binary Logic

Overview

- Two discrete values
  - True or false
  - Yes or no
  - High or low
  - 1 or 0

Overview

- Consists of binary variables and a set of logical operations
- 3 basic logical operations
  - AND
  - OR Each of which produces a result
  - NOT

AND

- Denoted by a dot (·) or the absence of a symbol.

$$x \cdot y = z = xy$$

- Interpreted to mean that
  - $z = 1$ if and only if (iff) $x = 1$ and $y = 1$
  - otherwise the result is $z = 0$.

AND

- The results of the operation can be shown by a truth table.

Inputs		Result
x	y	$x \cdot y$
0	0	
0	1	
1	0	
1	1	

$x \cdot y$ = 1 if and only if
x = 1 and y = 1;
otherwise = 0

- **All** inputs must be true for the result to be true.

OR

- Denoted by a plus (+)

$$x + y = z$$

- Interpreted to mean that
  - $z = 1$ if $x = 1$ or $y = 1$
  - otherwise the result is $z = 0$.

OR

- The truth table

x	y	$x + y$
0	0	
0	1	
1	0	
1	1	

$x + y$ = 1 if x = 1 or y = 1;
otherwise the result is z = 0

- **At least 1** input must be true for the result to be true.

NOT

- This operation is represented by a prime (‘)

x‘

- Referred to as the complement operation.

x	x'
0	
1	

Pitfall

- Binary logic should not be confused with binary arithmetic.

  + implies OR

  NOT addition

  · implies AND

  NOT multiplication

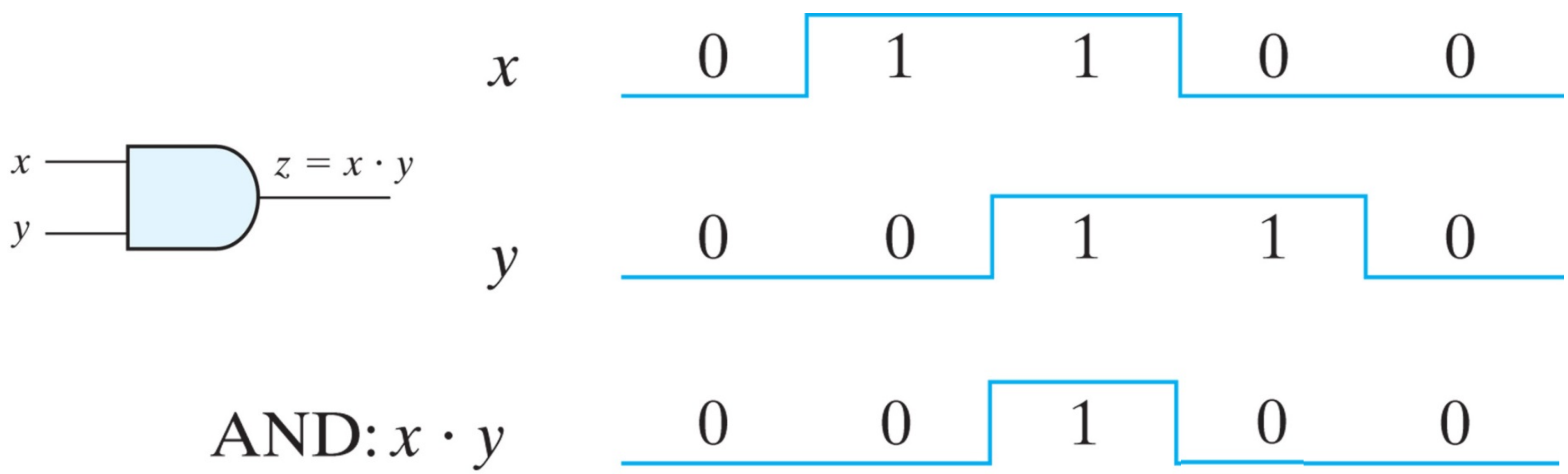
- Ex: a(b + cd) = a and (b or (c and d))

Another way to describe logic operations

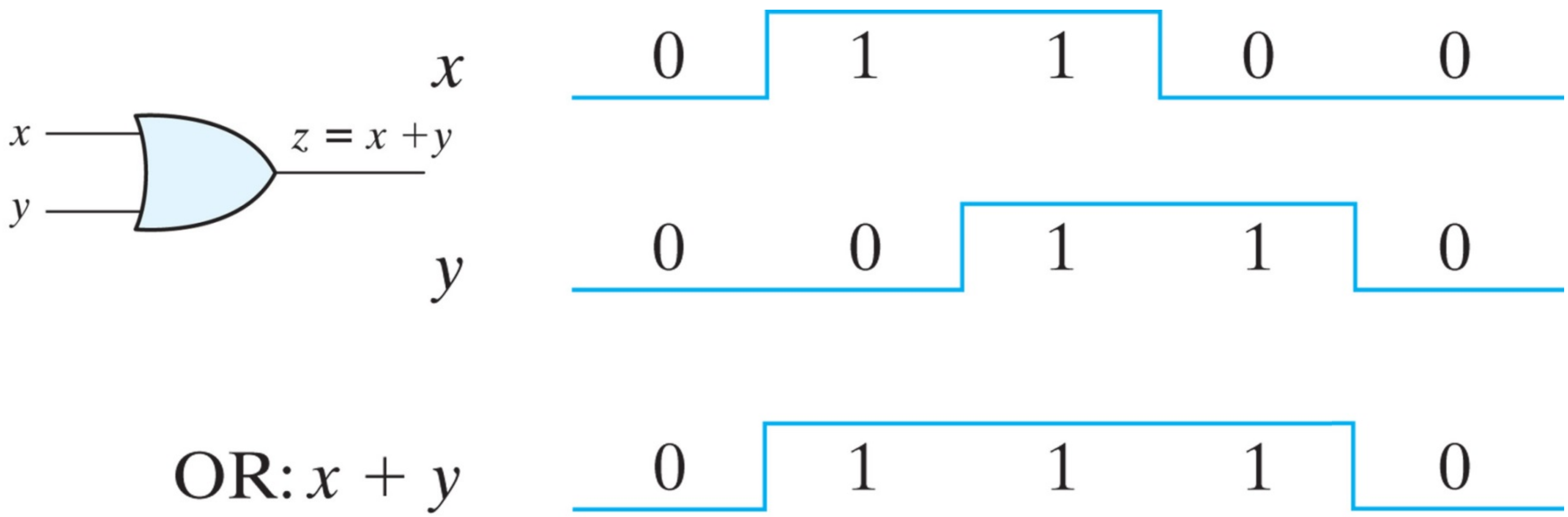
Logic Gates

- Electronic circuits that operate on one or more input signals to produce an output.

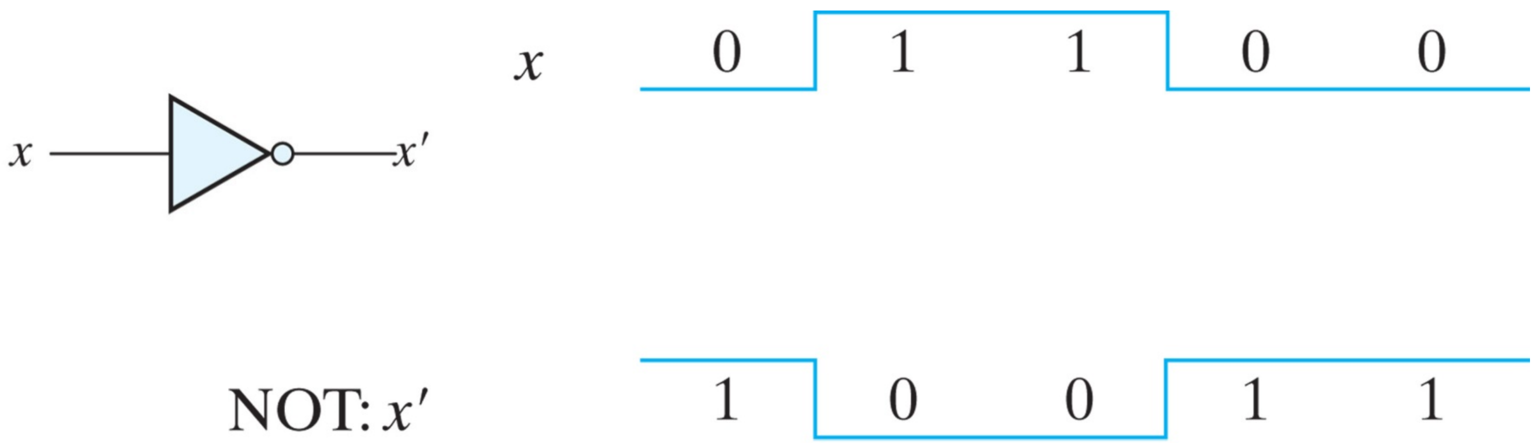
Timing Diagram



Timing Diagram



Timing Diagram



Summary

- Binary logic is comprised of 3 basic operations.
  - AND, NOT, OR
- Be wary of the pitfall
  - Binary logic is not binary arithmetic
- Each operation has a matching logic gate