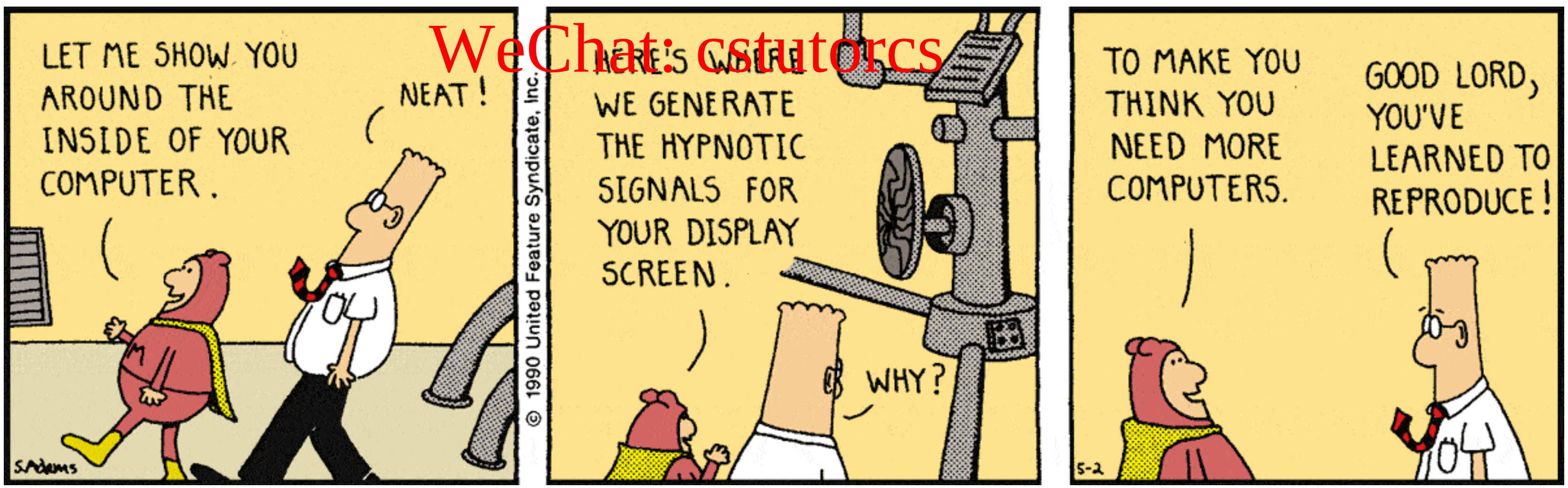
Lecture 2

Digital Devices and : Binary and Hexadecimal Numbers



Last Time: Computers

We have distinguished between

- General-purpose computers – e.g., desktops, laptops
- Embedded computers – e.g., microcontrollers

Two things both classes have in common is: They are **digital electronic computers**

Digital computers process information in discrete form, in particular **binary** two values: **0** and **1**

Electronic computers as opposed to mechanical systems

Vacuum tubes ➡ Transistors ➡ MOSFET ➡ **Integrated Chips (IC)**

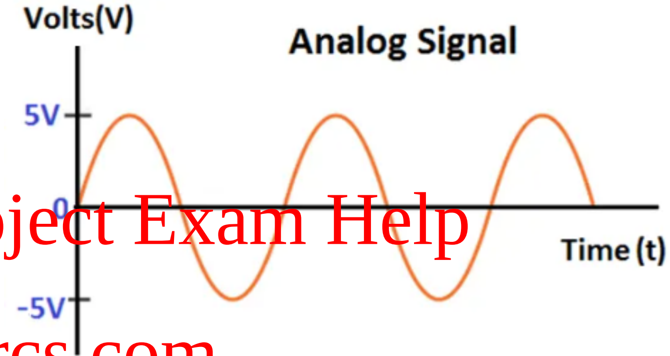
➡ Quantum computers

➡ ???

Analog vs Digital

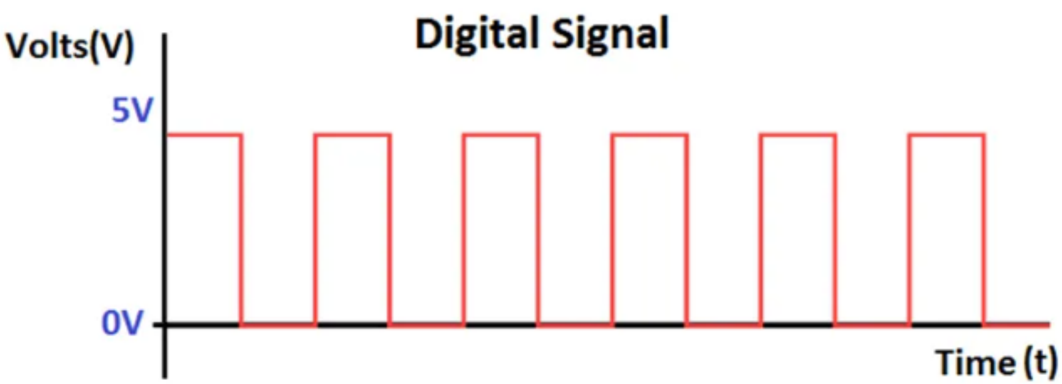
An **analog signal** takes a continuous range of values

$\Rightarrow$ infinite set of possible values



A **digital signal** takes values from **a finite set**

Often this finite set has only **two possible values** **0** and **1**



aka **logic signal**
or **binary signal**

The Digital Revolution

The **digital revolution** changed human civilization

Agricultural Rev. ~10,000 BCE	Industrial Rev. ~1750-1850	**Digital Rev. ~1950**
Agricultural Age	**Industrial Age**	**Information Age**

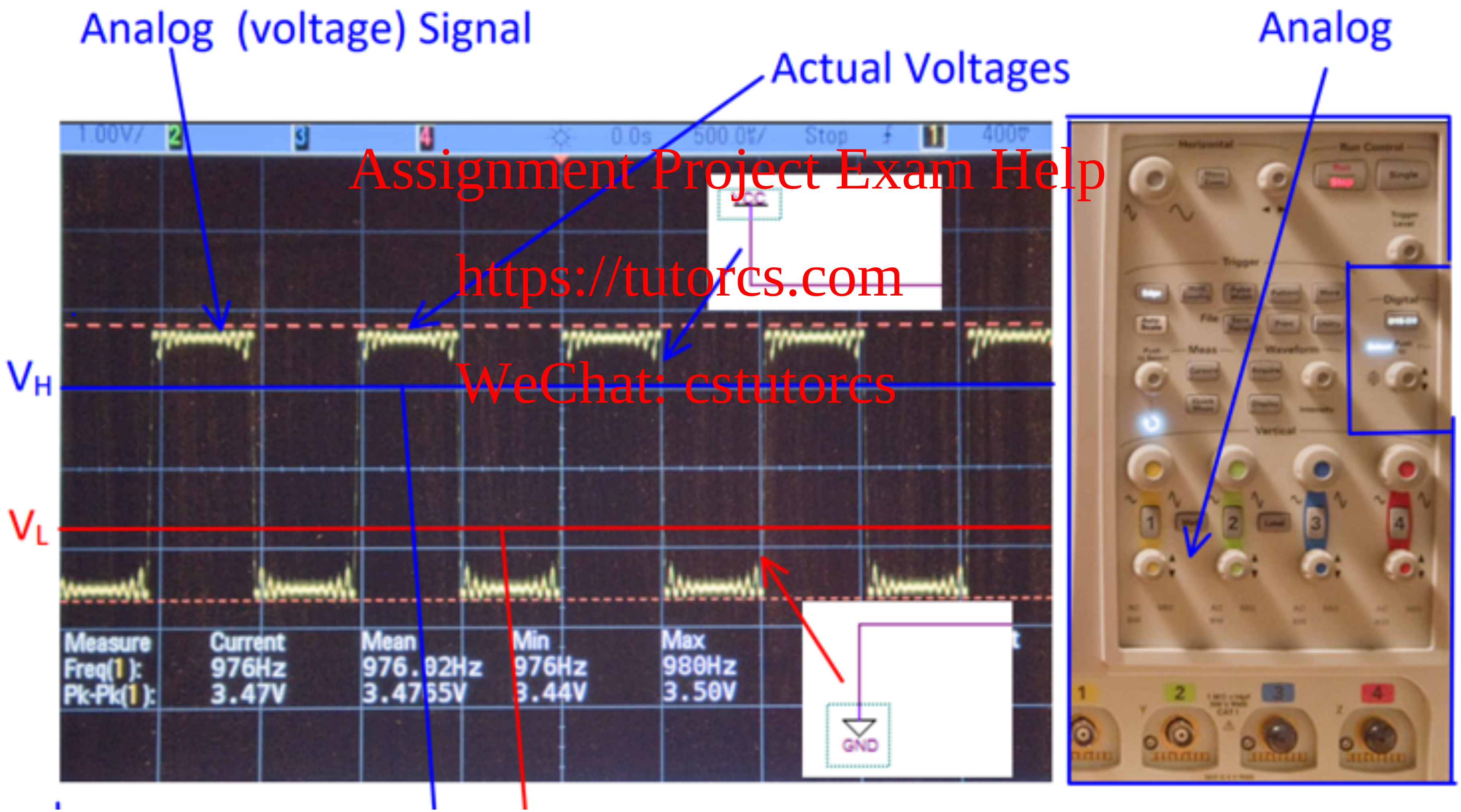
What is the digital revolution?

- Transition from analog to digital technology
- Transistor (1947) ⇒ Integrated Circuits (IC) ⇒ Computers
- Digital signals and digital logic
- Digital communication ⇒ The Internet

YET digital signals are ~~a big fat lie~~ only an abstraction

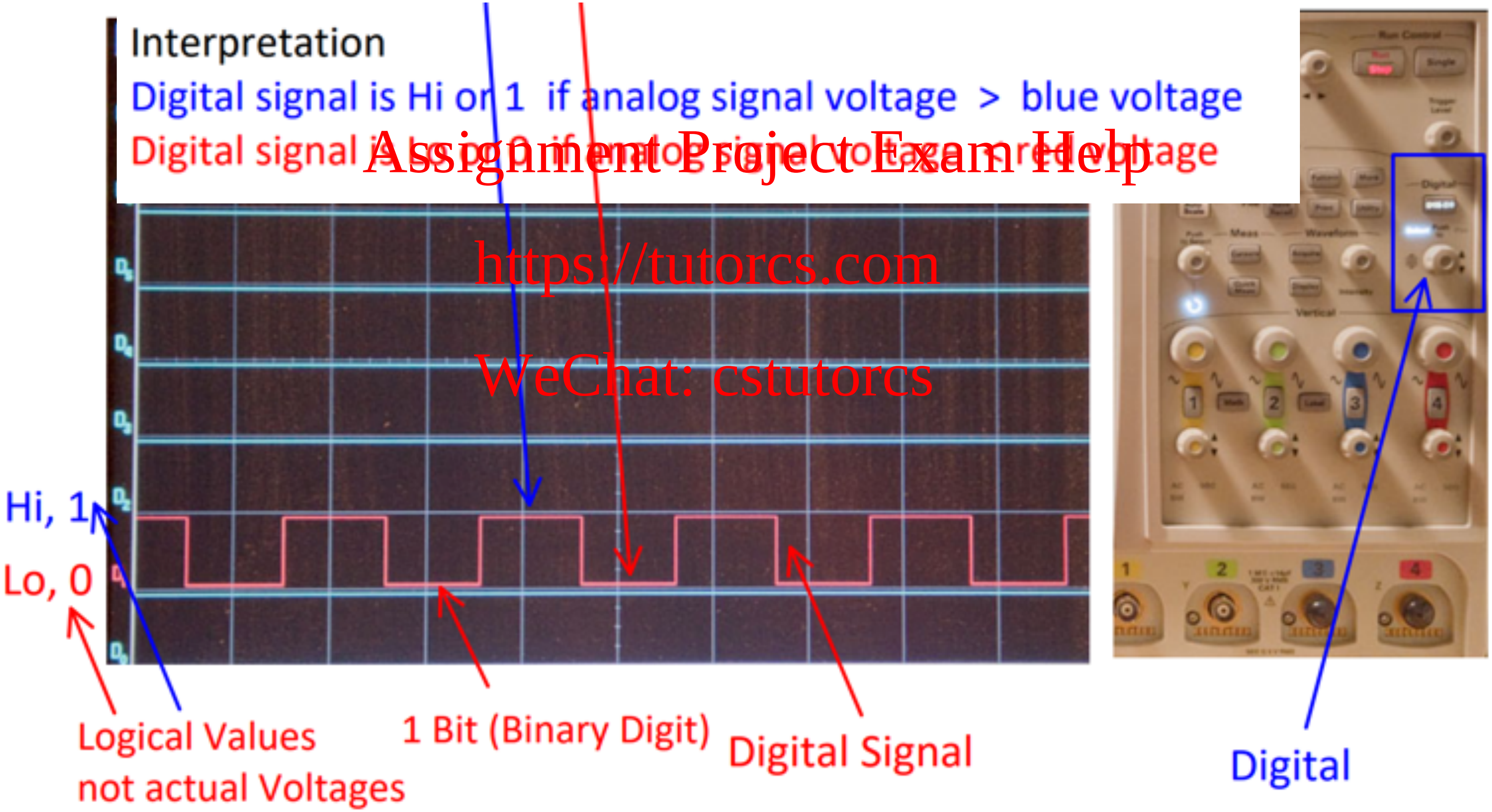
Digital Signals

Physical digital signals do not exist – a digital signal is an abstraction



Digital Signals

When a mixed signal oscilloscope displays a digital signal it is an abstraction



Digital Signals

This makes **digital signals very resilient to noise**

Digital signal is interpreted as

1 if (analog signal + noise) > blue voltage

0 if (analog signal + noise) < red voltage

Noise will have **no effect** as long as it is not too big

Listen to AM radio (analog) and HD radio and compare the sound quality

(Slightly) scratch a vinyl recording and a CD and see what happens

Binary Numerals

Digital signals take two values: **0** and **1**

Hardware can distinguish between **two stable states**: **0** and **1**

- Transistor and capacitor (dynamic RAM)
- Flip-flops (static RAM)
- Magnetic material (HDD)
- Cells in solid-state drives (SSD)

$\Rightarrow$ We have to learn to do everything with **0** and **1** only

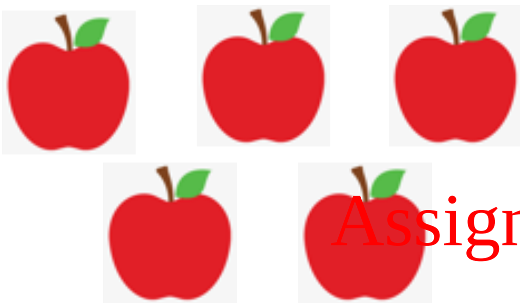
What do computers do? Duh, they **compute**!

- **Math – Numbers and operations**
- Logic – True/False

$\Rightarrow$ **We have to re-learn to do math using 0 and 1 only**

Numbers and Numeral Systems

A **number** is a mathematical object (abstraction) used to count and measure



Physical objects

"five"

Mathematical abstraction

There are multiple symbolic ways to express the number **"five"**

Tally marks

V

Roman numerals

5

Numerals we use in everyday life

101

Numerals used by computers

Numeral Systems

A **numeral system** is a mathematical notation for representing numbers

We focus on **positional** numeral systems where the **value** of the expressed number depends on numeral symbols (**digits**) and their **position**

	tens ones		
decimal base-10	11	= 1x10 + 1x1	"eleven"

	twos ones		
binary base-2	11	= 1x2 + 1x1	"three"

	16s ones		
hexa-decimal base-16	11	= 1x16 + 1x1	"seventeen"

Decimal Numerals

Base 10 ("ten")

- Each position is a power of 10
- **Digits** 0, 1, 2, …, 9

Why base 10?

Example:

100 10 1 0.1 0.01

721 . 53

But we all know this!

Binary Numerals

Base 2 ("two")

Why base 2?

- Each position is a power of 2
- **Digits 0 and 1**

Binary dig**it** = **bit**

Example:

8 4 2 1 ½ ¼

1101.01

represents value

1x8 + 1x4 +0x2 + 1x1 + 0x ½ + 1x ¼ = 13.25 in decimal

Notation: $(1011)_2$ 0b1011 1011b

Binary to Decimal Conversion

32 16 8 4 2 1
101011

= 32 + 8 + 2 + 1 = 43 in decimal

Decimal to Binary Conversion

e.g. $41 = 32 + 8 + 1$

32	16	8	4	2	1
1	0	1	0	0	1

More systematically:

$41 = 2 \times 20 + 1$

$20 = 2 \times 10 + 0$

$10 = 2 \times 5 + 0$

$5 = 2 \times 2 + 1$

$2 = 2 \times 1 + 0$

$1 = 2 \times 0 + 1$

What about 52579? Bookmark your favorite decimal-to-binary converter!!

Hexadecimal Numerals

Base 16 ("sixteen")

- Each position is a power of 16
- **Digits** 0, 1, 2, …, 9, A, B, C, D, E, F

Why base 16?

Example:

256 16 1

0x 3AD

represents value

$$3 \times 256 + 10 \times 16 + 13 \times 1 = (941)_{10}$$

in binary

0011 1010 1101

0x3 0xA 0xD

Hexadecimal to Binary

One hexadecimal digit encodes four **bits** (i.e., four binary digits)

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

Hexadecimal ↔ Binary

6FB

0110 1111 1011

001011000111

2 C 7

Binary	Hex
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

Bookmark your favorite number converter!!

Operations with Binary Numerals

We can easily add two binary numbers

```
carry    1    1
          01101011
       +  11010010
       ------------
         100111101
```


$$\begin{array}{r} 107 \\ +\ 210 \\ \hline 317 \end{array}$$

Note that the result is 9 bits

This would be an overflow in a 8-bit register

Operations with Binary Numerals

We can easily multiply two binary numbers

```
      1011
  x   1101
  --------
      1011
     0000
    1011
 + 1011
 ---------
  10001111
```

```
    11
  x 13
  ----
   143
```

Operations with Binary Numerals

We can subtract two binary numbers – as long as the result is not negative

At this point we do not know how to represent negative numbers with 0 and 1

Next time: 1’s complement, 2’s complement, signed numbers …