So then what is this course about?

- Understanding how computers are built, how they are designed, how they actually ***compute*** results, and ***how memory is managed, organized, and modified*** by programs in order to achieve anything

- Learning how CPUs work, and what you can do as programmers in order to obtain the maximum possible performance from the computer

- Becoming ***computer experts***, you have to understand computers better than the average (educated) person you meet on the street

- Learning the principles of CPU, memory, and peripheral design and operation that are needed to understand O/S, networking, file system design, and other later courses.

Overview of the course:

Learning Goals

- You will learn how information is represented, stored, and manipulated using the binary number system.
- You will understand how it is possible to use circuits to manipulate binary data and perform computation.
- You will expand your knowledge of logic, in particular, Boolean logic. You will use logic symbols and operators to represent and simplify logic functions and computations
- You will learn about memory, how it works, and how different types are used inside a computer
- You will study the components of a CPU, understand how they work together, and discover how code is executed. You will learn about modern CPUs
- You will learn assembly language programming, and through it, understand that all a program ever does is change information stored in memory

Overview of the course:

Skills to be developed

- Binary system manipulation and information representation

- Converting logic functions and computations to Boolean Algebra and/or circuits

- Construction and operation of simple circuits / memory banks

- Assembly programming – great preparation for C and C++

- Thinking in terms of pointers and memory addresses when programming

- Working with advanced programmable hardware (FPGA boards in the lab)

- Thinking about code optimization in terms of the hardware programs will run on

And now back to work.

What is a computer anyway?

“A computer is a programmable machine that receives input,
stores and manipulates data//information, and provides output in a useful format.”

Wikipedia (!!??)

Just a taste of what we can represent with ones and zeros:

Images

Music

Letters (and words)

Numbers, formulas, math!

People?

INFORMATION....

Counting in binary… is like counting in decimal, except we only have 0 and 1!

Now in binary:

$$000_2 = 0_{10}$$

Counting in binary… is like counting in decimal, except we only have 0 and 1!

Now in binary:

$$001_2 = 1_{10}$$

Counting in binary… is like counting in decimal, except we only have 0 and 1!

Now in binary:

$$010_2 = 2_{10}$$

Carry 1
to the left

Goes to 0

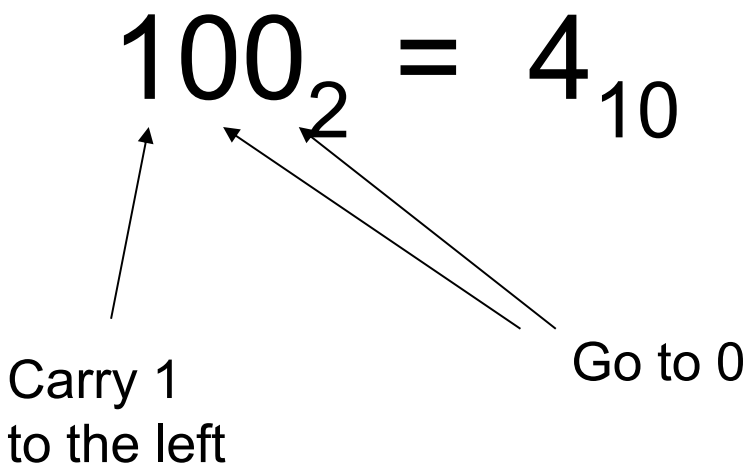
Counting in binary… is like counting in decimal, except we only have 0 and 1!

Now in binary:

$$011_2 = 3_{10}$$

Counting in binary… is like counting in decimal, except we only have 0 and 1!

Now in binary:



Binary numbers for 0-15

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

Make sure you know how to count in Binary!

Binary to decimal conversion (let's look at the board for a sec...)

Remember, each binary digit corresponds to a power of 2

For a 6 bit binary number:

32 16 8 4 2 1 - Value of each bit

0 1 1 0 1 0 = 16 + 8 + 2 = 26 in decimal

1 0 0 0 1 0 = 32 + 2 = 24 in decimal

0 0 1 0 0 1 = 8 + 1 = 9 in decimal

Decimal to binary conversion

You have to figure out what sum of powers of 2 equals the decimal number

Example with up to 6 binary digits;

		32	16	8	4	2	1	
21	→	0	-	-	-	-	-,	because 32 is too big
21	→	0	1	-	-	-	-,	21-16 = 5
5	→	0	1	0	-	-	-,	8 is too big
5	→	0	1	0	1	-	-,	5-4 = 1
1	→	0	1	0	1	0	-,	2 is too big
1	→	0	1	0	1	0	1,	1-1 = 0, done!

Decimal to binary conversion

You have to figure out what sum of powers of 2 equals the decimal number

Example with up to 6 binary digits;

		32	16	8	4	2	1	
57	→	1	-	-	-	-	-,	57-32 = 25
25	→	1	1	-	-	-	-,	25-16 = 9
9	→	1	1	1	-	-	-,	9-8 =1
1	→	1	1	1	0	-	-,	4 is too big
1	→	1	1	1	0	0	-,	2 is too big
1	→	1	1	1	0	0	1,	1-1 = 0, done!

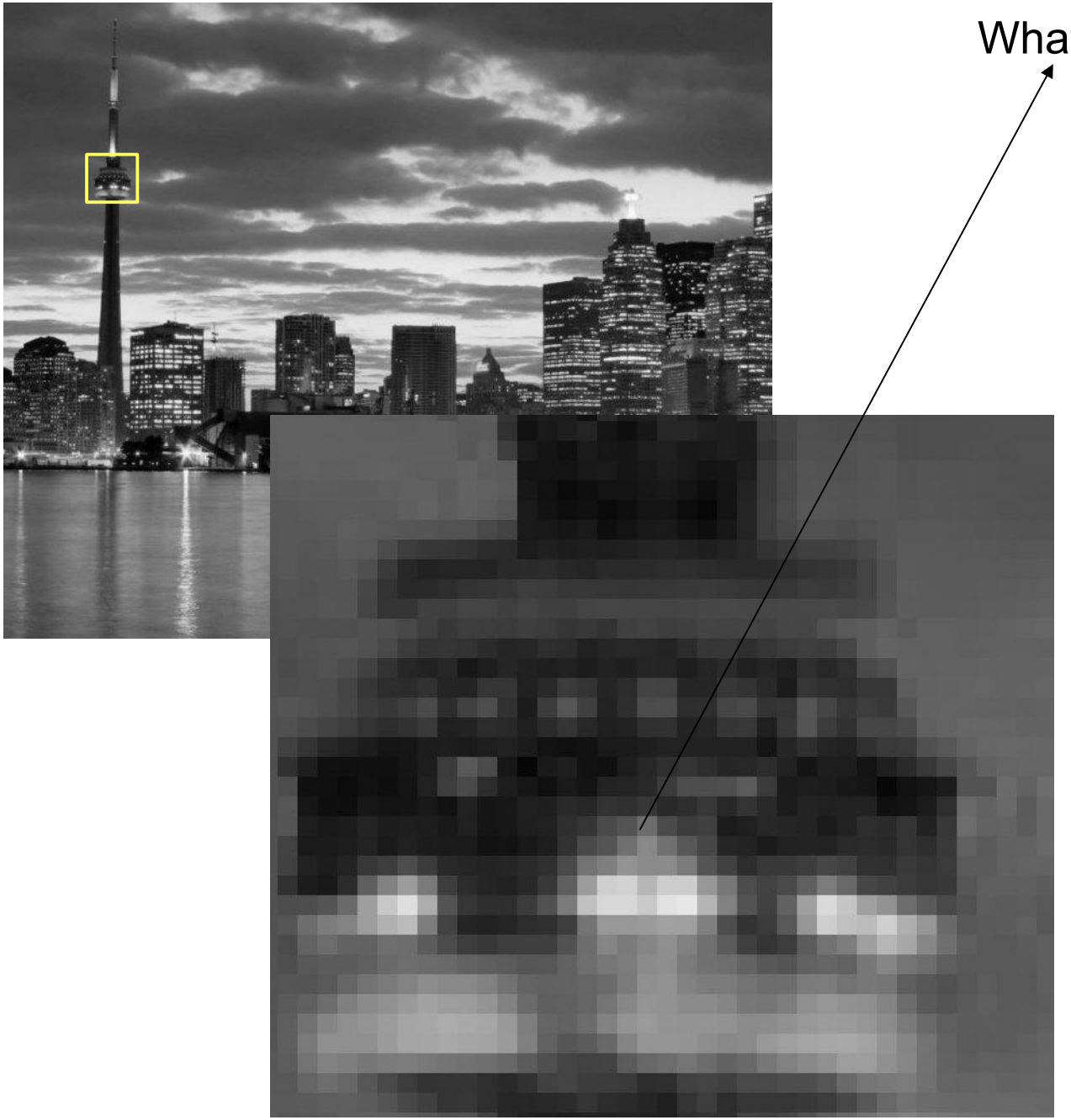
How do we represent letters? (text)

Assign a numeric value to each character... then convert to binary!

Character	Decimal	Binary
A	0	0000
B	1	0001
C	2	0010
.		
.		
.		
K	10	1010
L	11	1011
etc.		

Several standard ***encodings*** exist, e.g. ***ASCII – A****merican* ***S****tandard* ***C****ode* ***F****or* ***I****nformation* ***I****nterchange* – represents characters, numbers, symbols with numbers from 0 to 255 (how many bits per character is that?)

How about pictures?



What is each 'dot' called?

Encode brightness
at each pixel

0 = black
255 = white

Convert to binary... done!

Transistors as binary switches

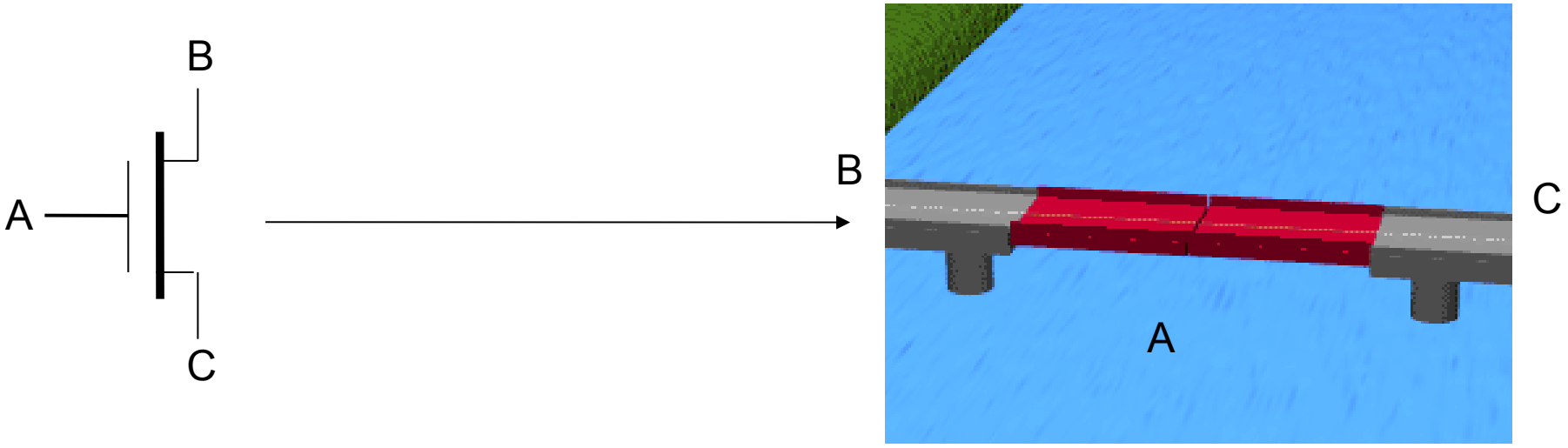


Illustration from Wikipedia, CC by ***Y_tambe***

- Similar to a lifting bridge. Open, cars can not cross, closed, cars can move along.

- Control of the state of the bridge corresponds to terminal ***A*** in the transistor

- ***Here's the key for understanding digital circuits:*** Forget about voltages and currents. All that matters is which circuit paths are connected, and which are not.

Simple binary computation : Logic AND function

Input: *Two bits (x,y) whose values are set by us (or some other circuit component)*

Output: *1 bit (z) whose value is the result of the* ***AND*** *operation on the two input bits*

z = x ***AND*** y is ***true*** if and only if both x and y are also ***true***

x	y	z = x ***AND*** y
0	0	0
0	1	0
1	0	0
1	1	1

Programming language example:

if (today is monday) ***and*** (time of day is 11am) then (go to B58 class)
x y only occurs if both x and y are true!

Simple binary computation : Logic AND function

Input: *Two bits (x,y) whose values are set by us (or some other circuit component)*

Output: *1 bit (z) whose value is the result of the* ***AND*** *operation on the two input bits*

z = x ***AND*** y is ***true*** if and only if both x and y are also ***true***

x	y	z = x ***AND*** y
0	0	0
0	1	0
1	0	0
1	1	1

*Input(s) – **provided** to the circuit (by the user or some other component in our system) – equivalent to input values in a program*

*Output(s) – **computed** by the circuit – equivalent to result variables in a program*

Programming language example:

if (today is monday) ***and*** (time of day is 1pm) then (go to B58 class)

x y only occurs if both x and y are true!

Simple binary computation : Logic AND function

Input: *Two bits (x,y) whose values are set by us (or some other circuit component)*

Output: *1 bit (z) whose value is the result of the* ***AND*** *operation on the two input bits*

z = x ***AND*** y is ***true*** if and only if both x and y are also ***true***

x	y	z = x ***AND*** y
0	0	0
0	1	0
1	0	0
1	1	1

Truth table – lists for all possible combinations of input value(s) what the expected output value(s) should be. It completely defines the function being computed

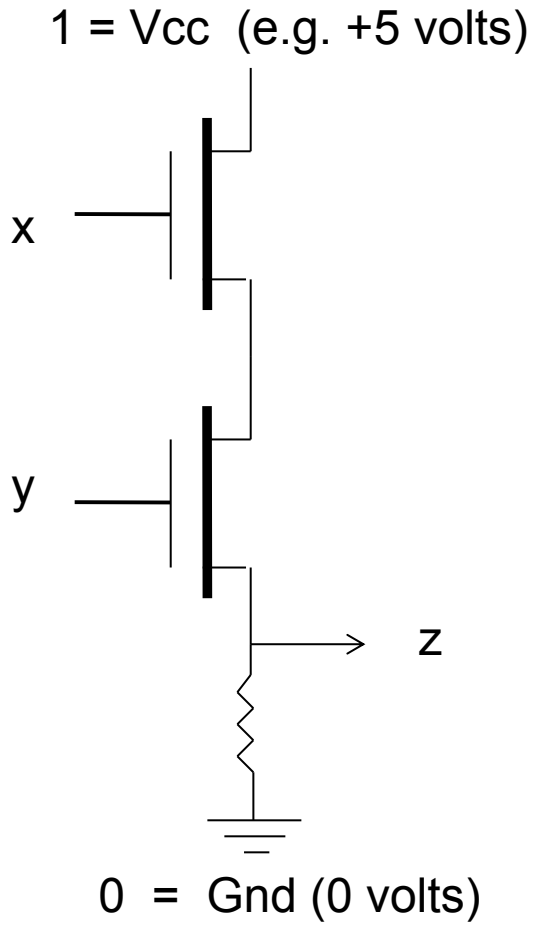
Programming language example:

if (today is monday) ***and*** (time of day is 1pm) then (go to B58 class)

x y only occurs if both x and y are true!

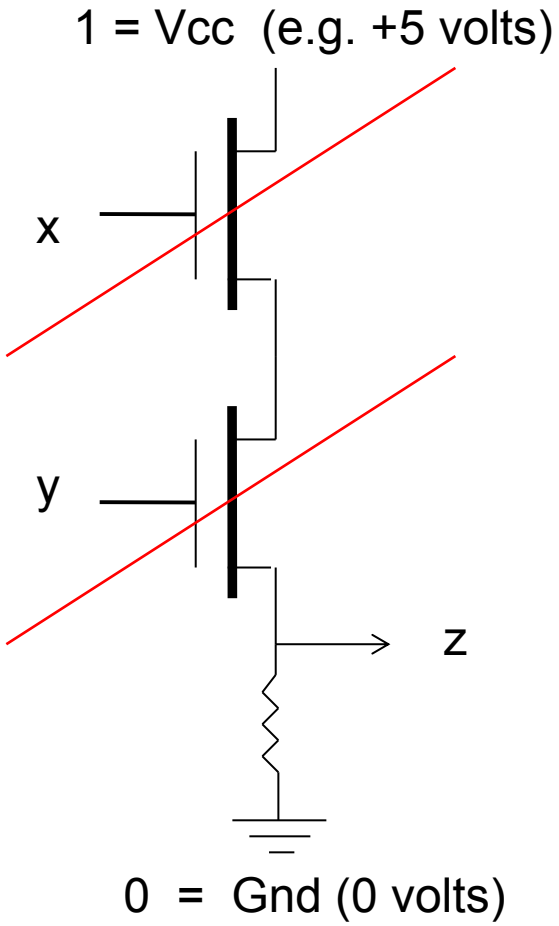
Simple binary computation : Logic AND function

*Implementing **AND*** with switches... how?



Simple binary computation : Logic AND function

*Implementing **AND*** with switches... how?



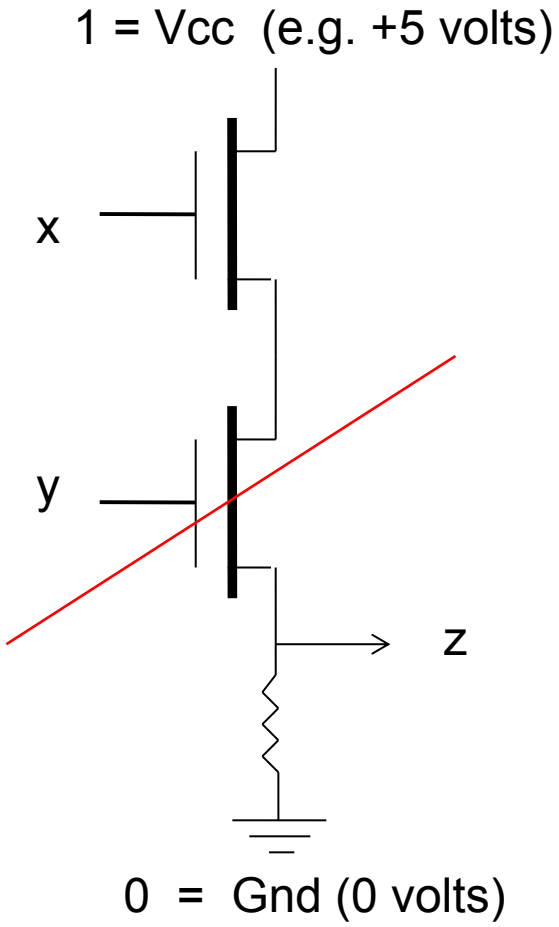
x = false (0)
y = false (0)

Both transistors ***open***

z = false (0)

Simple binary computation : Logic AND function

Implementing ***AND*** with switches... how?



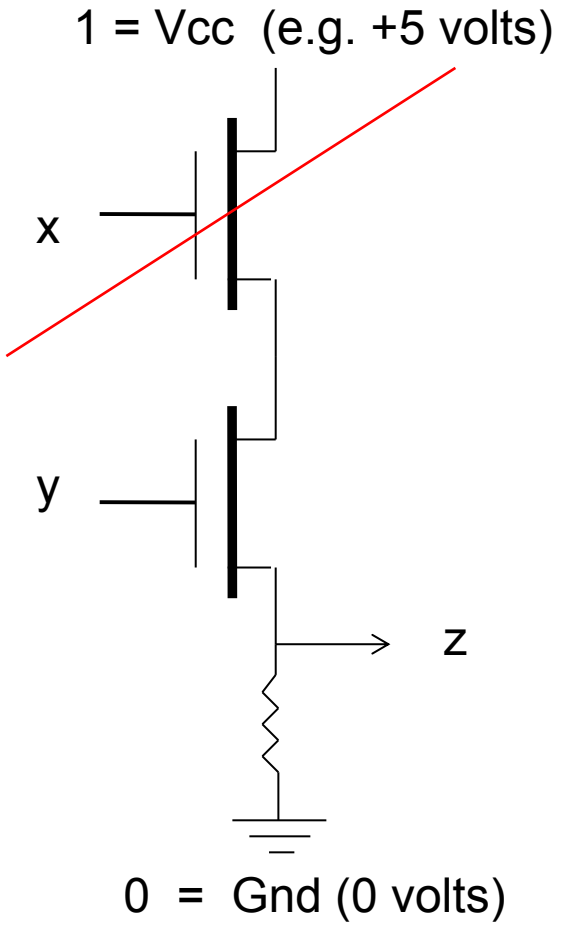
x = true (1)
y = false (0)

First transistor ***closed***, second is ***open***

z = false (0)

Simple binary computation : Logic AND function

*Implementing **AND*** with switches... how?



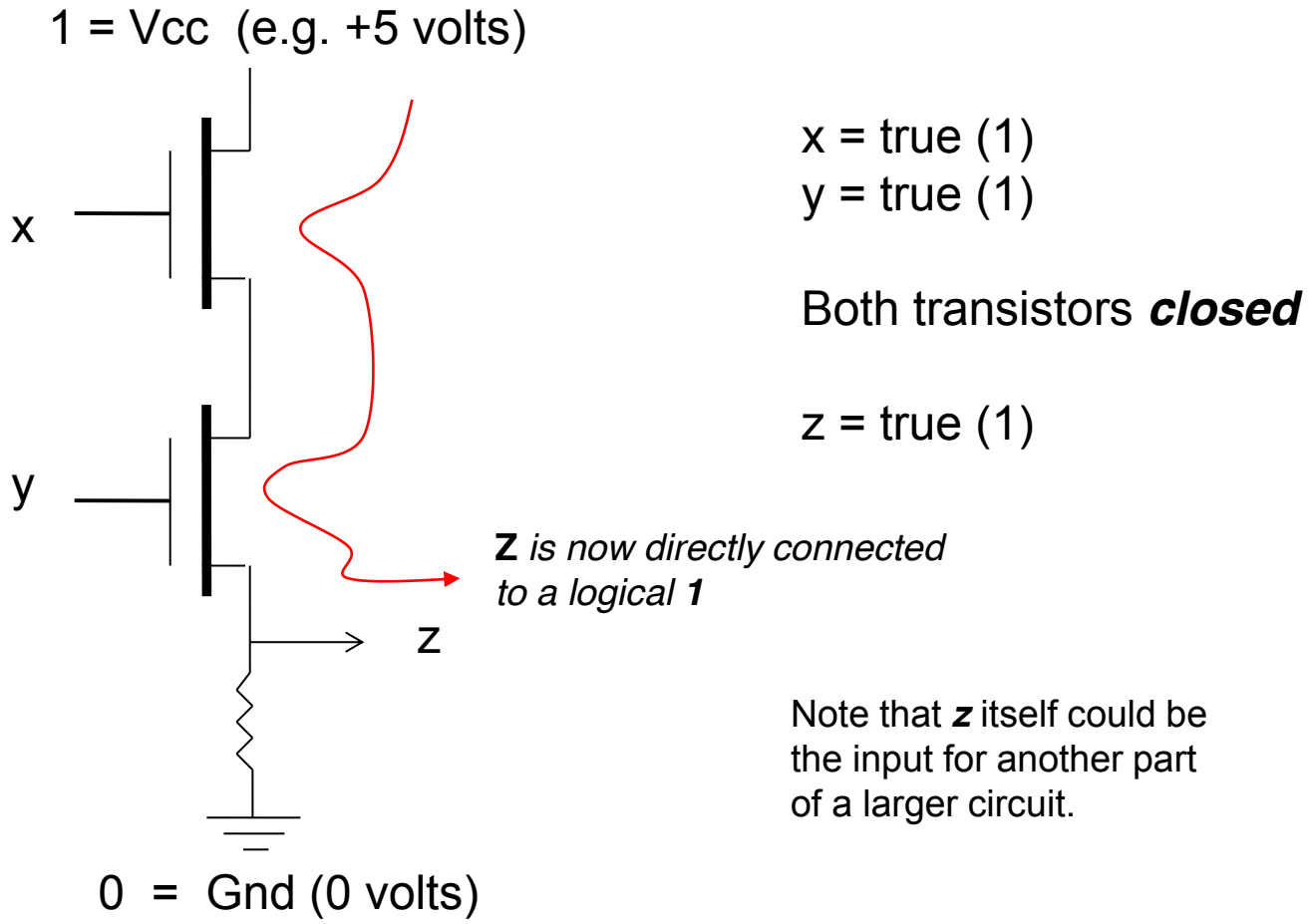
x = false (0)
y = true (1)

First transistor ***open***, second is ***closed***

z = false (0)

Simple binary computation : Logic AND function

*Implementing **AND*** with switches... how?



x = true (1)
y = true (1)

Both transistors ***closed***

z = true (1)

Z** is now directly connected to a logical **1

Note that ***z*** itself could be the input for another part of a larger circuit.

At this point you should be able to

- Explain the general definition of a computer

- List the components of a general computer

- Explain why we use electricity to build digital computers

- Represent decimal numbers in binary – convert from decimal to binary and back

- Explain what an ***encoding*** is

- Determine in a simple circuit whether transistors are open or closed