CMSC 11:
Introduction to Computer Science

Jaderick P. Pabico <jppabico@uplb.edu.ph>
Institute of Computer Science, CAS, UPLB

Pop QUIZ

- One fourth sheet of paper (your paper only)
- Write:
  - your name
  - laboratory section
  - today's date

Wrong information makes your answer wrong!

- Open notes (BUT you may only look at your notes)
- Mouth shut AND eyes on own paper
- Answer in **3 Minutes only**!

QUIZ PROPER

- Fill up the TRUTH TABLE for this LOGICAL EXPRESSION:

  (A AND NOT B) OR (NOT A AND B)

- Use this empty TRUTH TABLE:

Input (A)	Input (B)	OUTPUT
1	1	
1	0	
0	1	
0	0	

QUIZ ANSWER

- The TRUTH TABLE for this LOGICAL EXPRESSION:

  (A AND NOT B) OR (NOT A AND B)

- Is:

Input (A)	Input (B)	OUTPUT
1	1	0
1	0	1
0	1	1
0	0	0

- How? LOGICAL ANALYSIS!!!

QUIZ ANALYSIS

- **(A AND NOT B) OR (NOT A AND B)**

Input (A)	Input (B)	Not A	Not B	A & Not B C	Not A & B D	C or D
1	1	0	0	0	0	0
1	0	0	1	1	0	1
0	1	1	0	0	1	1
0	0	1	1	0	0	0

Intermediate Table Columns

Input (A)	Input (B)	OUTPUT
1	1	0
1	0	1
0	1	1
0	0	0

Review

- AND-gate
- OR-gate
- INVERTER
- Problem:Writing the I/O table given a logic diagram
- Real Problem: Designing a logic diagram given an I/O table

Real Problem

- Draw (design) the logic diagram for this input-output (I/O) table

Input (A)	Input (B)	Output (C)
1	1	0
1	0	1
0	1	1
0	0	0

- HOW?

Designing Logic Gates

- Begin by finding all rows where the output C=1.

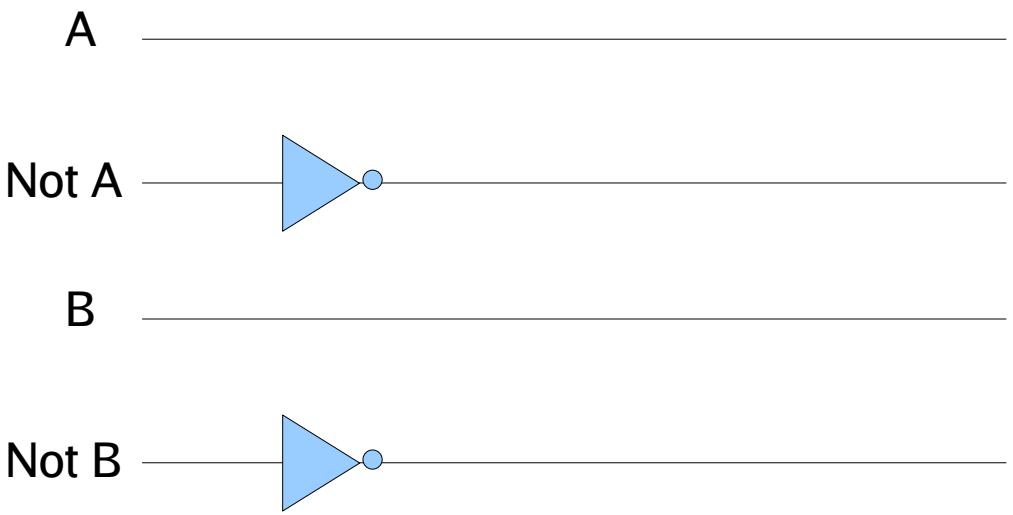
Input (A)	Input (B)	Output (C)
1	1	0
1	0	1
0	1	1
0	0	0

These rows

- The table says C=1 if
  - **A=1 and B=0 or A=0 and B=1**
- C=0 otherwise

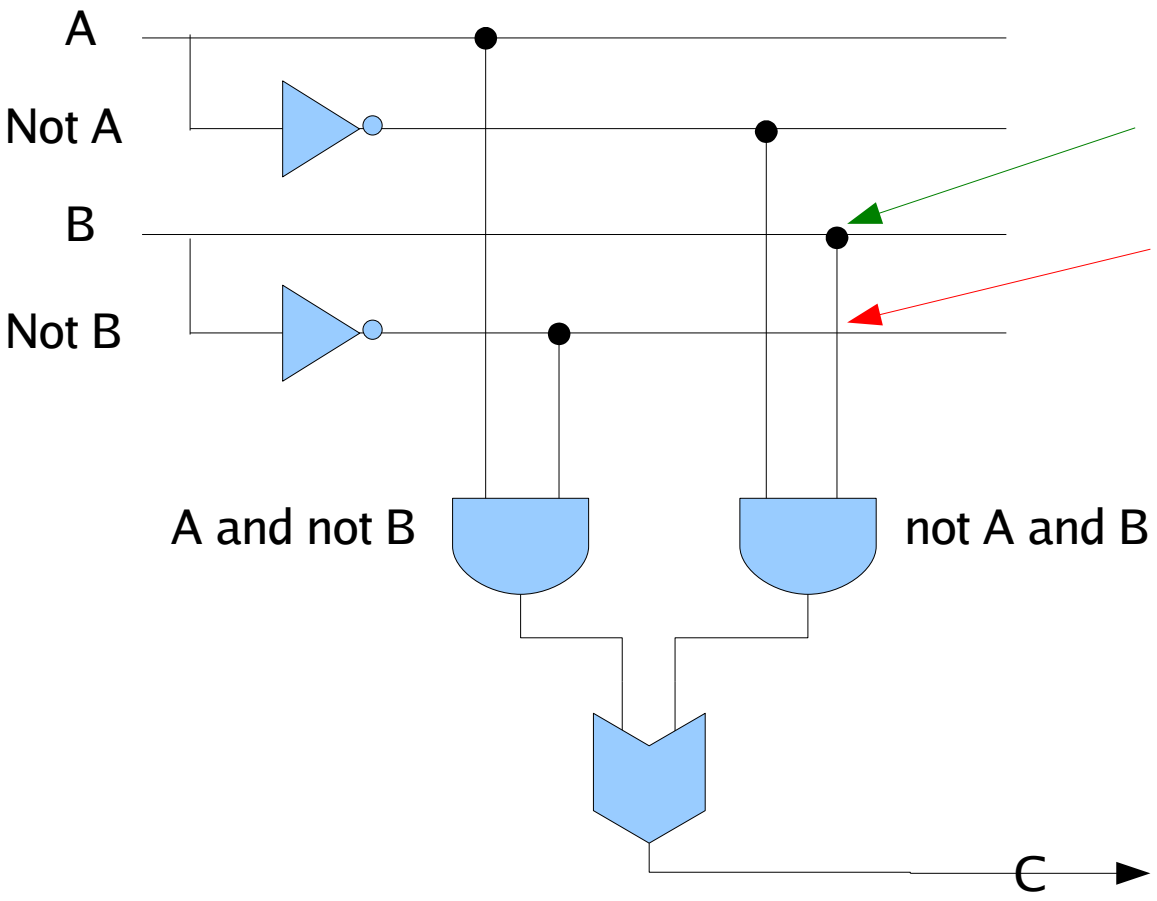
Designing Logic Gates

- In other words:
  - C = (A and not B) or (not A and B)
- To draw the circuit, run the input wires and their negatives in one direction (usually horizontal)



Designing Logic Gates

- Then attach the gates to the appropriate wires (usually vertical)

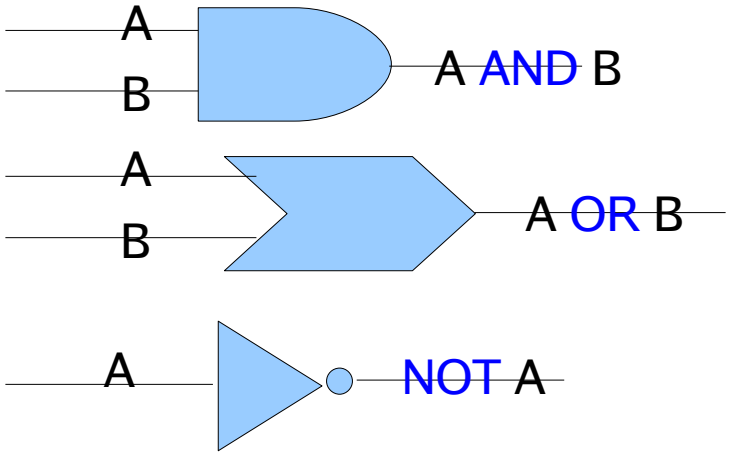


Intersecting lines with dots mean there is a connection

No connection here

Multiple Inputs

- Logic gates
  - have only one or two inputs
  - and a single output

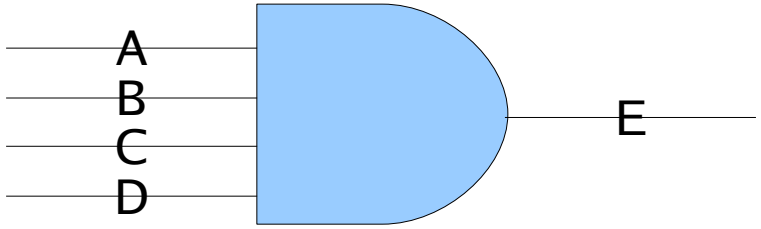


- Computer components have
  - many inputs
  - with complicated I/O behavior

- BUT any I/O table can be produced by a combination of logic gates.

Multiple Input Logic Gates

- Example: 4-Input AND-gate

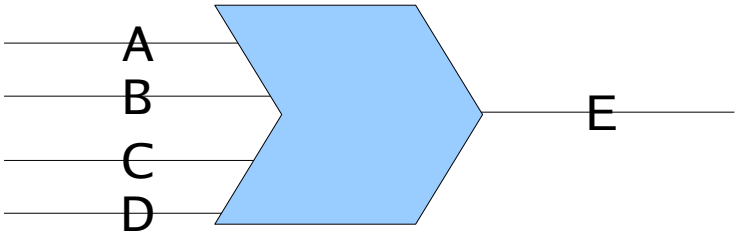


A	B	C	D	E
1	1	1	1	1
1	1	1	0	0
1	1	0	1	0
…	…	…	…	…
0	0	0	0	0

- This means that E=1 if A=B=C=D=1 and
- E=0 otherwise

Multiple Input Logic Gates

- Similarly, there is a multiple-input OR-gate
- Example: 4-input OR-gate

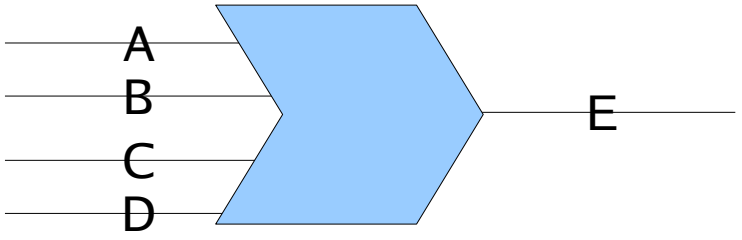


A	B	C	D	E
1	1	1	1	1
1	1	1	0	1
1	1	0	1	1
…	…	…	…	…
0	0	0	0	0

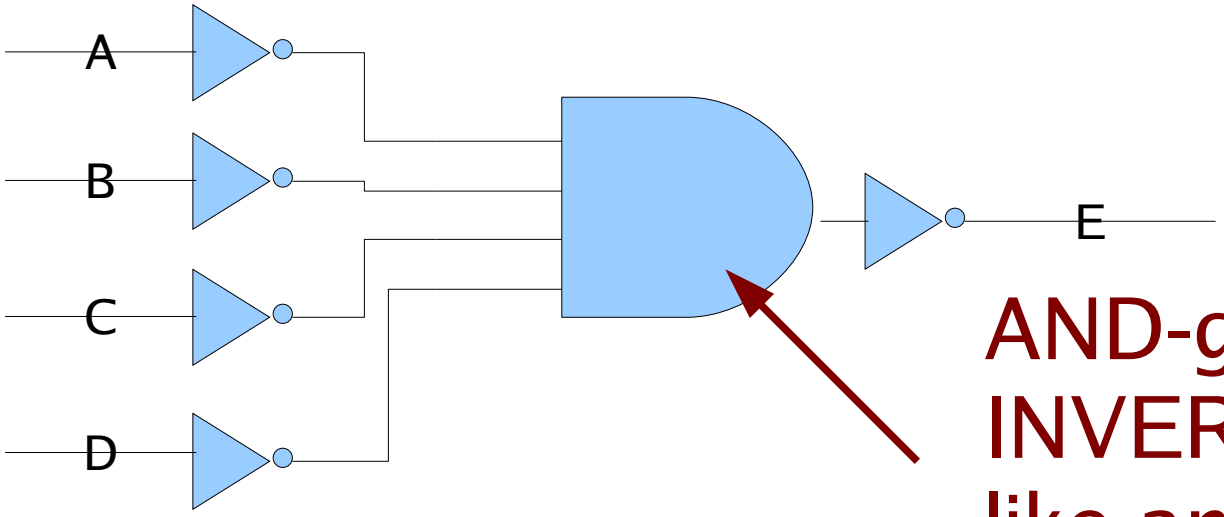
- An OR-gate can actually be made from an AND-gate and some inverters.

Multiple Input Logic Gates

- How?



A	B	C	D	E
1	1	1	1	1
1	1	1	0	1
1	1	0	1	1
…	…	…	…	…
0	0	0	0	0



AND-gate with INVERTERS acting like an OR-gate

Designing Logic Gates

- How do you now design logic gates with multiple inputs?
- Exactly the same method works:
- For example:

A	B	C	D
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

Again, find all rows with output=1

Note: All possible input combinations!

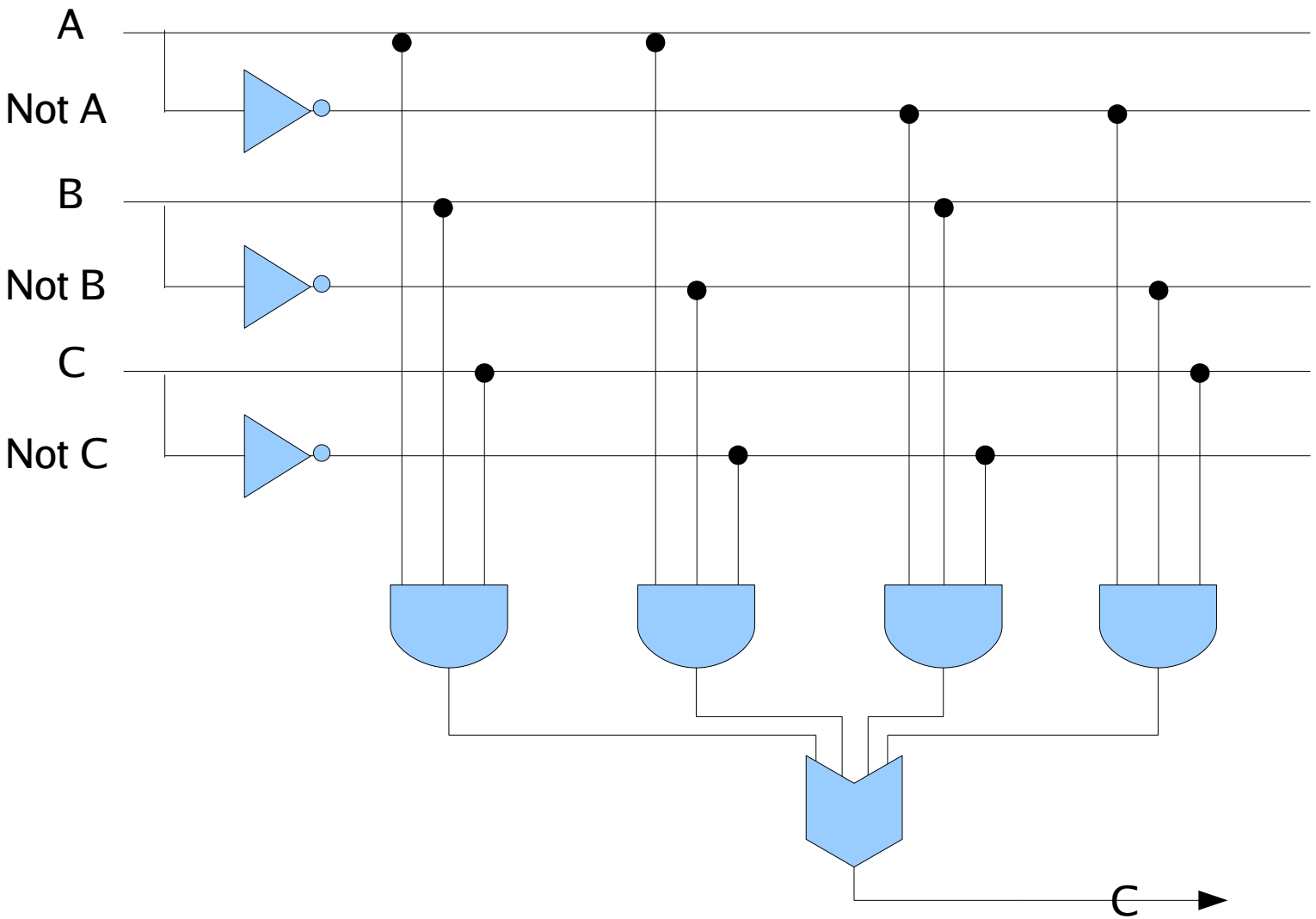
Designing Logic Gates

- Based on D=1 from the I/O table:
- D = (A and B and C) or
  (A and not B and not C) or
  (not A and B and not C) or
  (not A and not B and C)

A	B	C	D
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

- Can you draw/design the logic diagram?

Designing Logic Gates



Designing Logic Gates

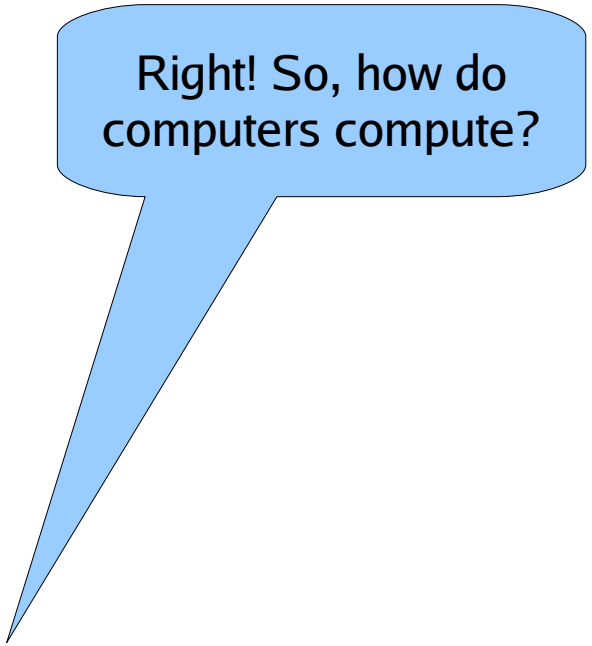
- Where does this leave us?

With one foot in the CMSC 11 grave?

- By now you may be getting the idea that
  - information is encoded inside computers as strings of 1's and 0's
  - which can be transformed in any way we like by the right combination of logic gates

Computer's job

- BUT we haven't really seen how logic gates can do the job computers were designed for



The questions:

- IS there some natural way to represent numbers using only 0's and 1's?
- CAN the operations of arithmetic be built out of logic?

The answer:

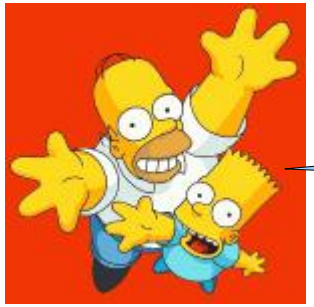
- (which goes back to Leibniz)

Here is the answer.
This answer is as sure
as the sun is shining

- IS a system called BINARY NUMBERS

Binary Numbers

- Are based on TWO
- Our decimal system
  - based on TEN
  - was a result of our having ten fingers
    - an accident of nature!
- Binary system are what would have evolved if we'd been born with TWO fingers.



So, what number system do we use at Springfield?

10

- Look at the symbol (one-zero)
- Forget that it usually means ten!
- From now on, think of it as a one followed by a zero.
  - In and of itself, it has nothing to do with ten!

10

10

- The symbol only makes “ten” flash through your mind because you’ve always called it that
- It’s like a ritual:
  - perform it over and over and over and it becomes automatic

10

- In actuality, “10” means

1 **One handful and**

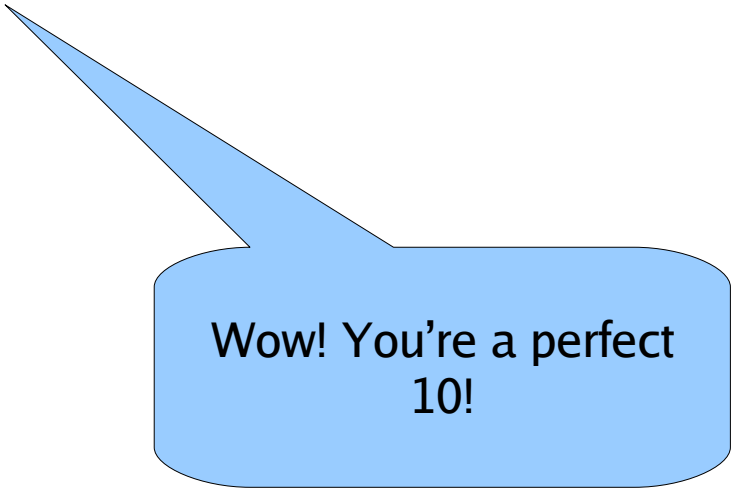
0 **Zero fingers left over**

- Since we humans have ten fingers, our “10” is ten
- But to an organism with, say, eight fingers, “10” would mean eight!

10

- In the case at hand, with just two fingers in a handful

10 means TWO



So ...

- we can write...

$$10_{binary} = 2_{decimal}$$

Note: Do NOT read this as "ten equals two"

10, 10, 10, 10,
10, 10, 10
10, 10, 10, 10

Two, two,
two, two,
two, two,
two, two,
two!

Likewise ...

- One-zero-zero

$$100_{binary} = ?_{decimal}$$

In Decimal, that's 10x10 or a hundred!

Well, in Binary, it's 10x10 also

BUT that only amounts to FOUR

Similarly ...

- One-zero-zero-zero

$$1000_{\text{binary}} = ?_{\text{decimal}}$$

It's 10x10x10

Which is the same as 2x2x2 = 8

In General ...

- One followed by **n** zeroes is

$$\underbrace{2x \ldots x2}_{n \text{ times}} = 2^n$$

That's two to the nth power

In the future ...

- everyone will be required by law to memorize the powers of two, up to 2^{10}.

$$
\begin{aligned}
1 &= 2^0 = 1 \\
10 &= 2^1 = 2 \\
100 &= 2^2 = 4 \\
1000 &= 2^3 = 8 \\
10000 &= 2^4 = 16 \\
100000 &= 2^5 = 32 \\
1000000 &= 2^6 = 32 \\
10000000 &= 2^7 = 128 \\
100000000 &= 2^8 = 256 \\
1000000000 &= 2^9 = 512 \\
10000000000 &= 2^{10} = 1024
\end{aligned}
$$

Homework Assignment

- How do you translate/convert binary numbers to decimal?
- How do we calculate with binary numbers?
  - addition
  - subtraction
  - multiplication
  - division

Binary = Decimal ?

- All other binary numbers
  - example, 101, 1111, 11000
  - and every other pattern of 0s and 1s
  - are sums of such powers of two

Including 1010101010101010?

- Completely analogous to decimal
- HOW?

Numbers as addition:

In Decimal:	In Binary:	
497	111110001	= 497
---	---	---
400	100000000	256
90	10000000	128
7	1000000	64
	100000	32
	10000	16
	1	1

Number translation

- To translate a binary number into the decimal system:
  - list the powers of two over the corresponding places
  - add those lying over a 1
  - Example: 100011010

2^{10} 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2 1

---

1 0 0 0 1 1 0 1 0

256 + 16+8 + 2 = 282

Counting in Binary

- To make this a bit more concrete, let's count from one in binary.
  - It's just like counting in decimal (only easier)
- In decimal, to count past nine,
  - you write 0 and carry 1
- In binary,
  - you have to carry 1 every other number

Binary	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
10000	16
...	...
etc.	etc.

Counting in Binary: observation

- as you may have noticed, binary numbers get LONNNNNG very fast!
- This makes them hard for us humans to use without making mistakes
- But for computers they are ideal!

Binary	Decimal
10001	17
10010	18
10011	19
10100	20
10101	21
10110	22
10111	23
11000	24
11001	25
11010	26
11011	27
11100	28
11101	29
11110	30
11111	31
100000	32
...	...
etc.	etc.

Binary calculation

- Binary calculation is simple
- There are only five rules to remember:

  (1) 0 + 0 = 0

  (2) 0 + 1 = 1

  (3) 1 + 0 = 1

  (4) 1 + 1 = 10

  (5) 1 + 1 + 1 = 11

As opposed to one hundred sums in decimal like: 0+0, 0+1, 0+2, 0+3, 0+4, etc., etc., etc., 9+5, 9+6, 9+7, 9+8, 9+9

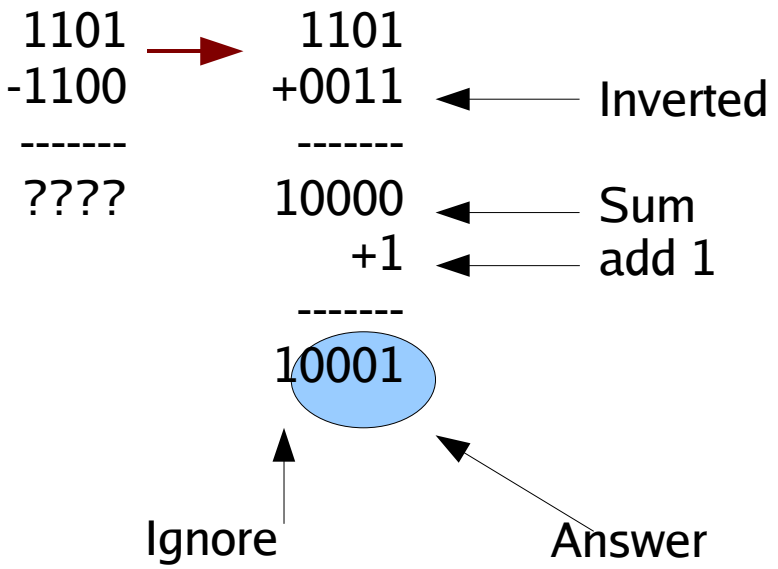
Binary addition

- To add two binary numbers:
  - proceed place by place from right to left
  - carry a 1 when necessary

	1	11	11	← carries
1110	1110	1110	1110	
111	111	111	111	
---	---	---	---	
1	01	101	10101	

Binary subtraction

- SUBTRACTION IS DONE BY ADDITION
- Method is called “two’s complement”
  - invert the number to be subtracted
    - all 1s become 0s
    - all 0s become 1s
  - add the two numbers
  - add 1 to the sum
  - ignore the final carry
- Example: 1101 - 1100



Binary multiplication/division

- Binary multiplication (actually, any multiplication) may be done by repeated addition
- To multiply A by B,
  - add A to itself B times
- Similarly, division can be done by repeated subtraction
- Example: 110 x 11 = ?

```
  110
+110      } 11 times
+110
------
10010
```

Binary arithmetic: summary

- The computer can do all arithmetic by ADDING!
- Addition: 5 rules
- Subtraction: addition with two's complement
- Multiplication: repreated addition
- Division: repeated subtraction

- This brings us to a component of the processing unit: THE ADDER

The ADDER

- We will combine logic gates to create a binary adder.

- But before we do that, we need a bit of terminology.

Terminology

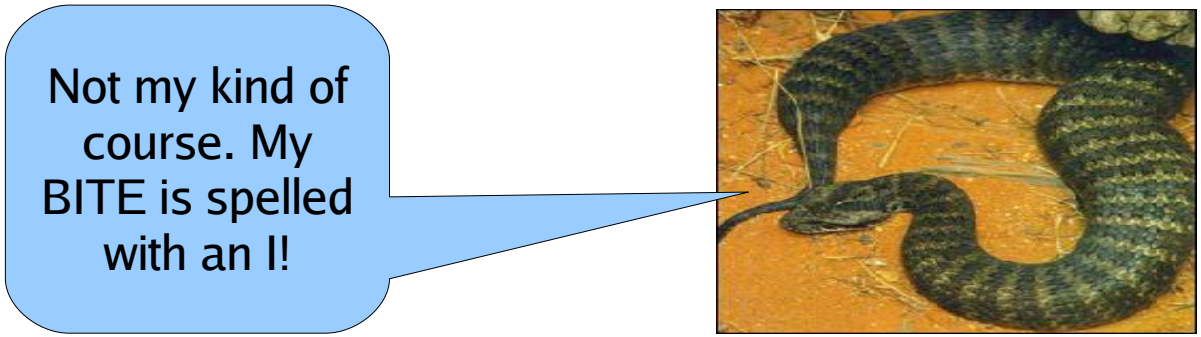
- BIT
  - an abbreviation of Binary Digit
  - it refers to a single 0 or 1

Is it
BInary digi**T**
or
Binary dig**IT**?

- BYTE
  - a group of eight bits
  - there are 2^8 or 256 possible bytes
  - from 00000000 to 11111111.

The ADDER

- Now, let's see what an adder look like.



- Example: 4-bit adder
  - capable of adding two four-bit numbers
  - Trivia: NIBBLES = four-bit number

The 4-bit adder

- The input of our 4-bit adder
  - must consist of eight bits
    - four for each nibble
    - two nibbles
- The output must be five bits
  - a nibble
  - plus one bit for a possible carryber

The 4-bit adder

- How to proceed?
  - One way is to make a giant truth table
  - match every possible combination of inputs with the correct output
  - construct a huge stew of AND-gates and NOT-gates to force a solution
  - This procedure is doable
    - BUT complexity of the task might make you throw up your hands.

Or just throw up, if you have no hands!

The 4-bit adder

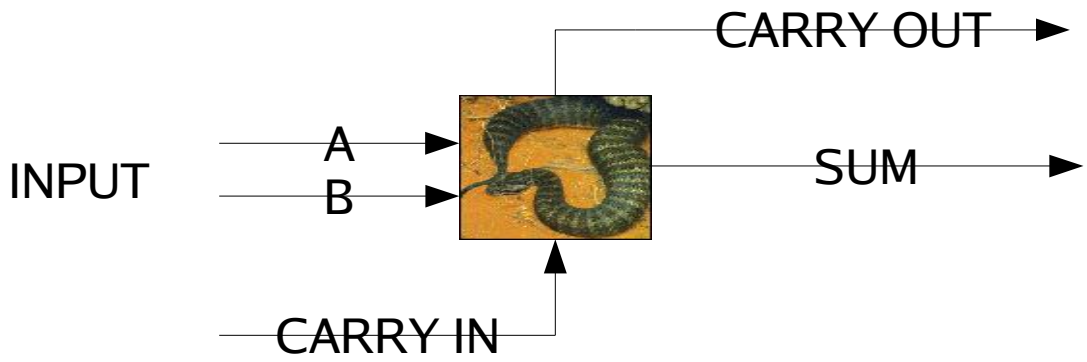
- INSTEAD:
  - recall how addition works in practice:
    - column by column
    - with a CARRY BIT carrying out of one column and into the next

```
1 1 1 0      ◄carry
  1 1 1 0
  1 0 1 1
---------------
1 1 0 0 1
```

  - So, it should be possible to make a 4-bit adder out of four 1-bit adders!

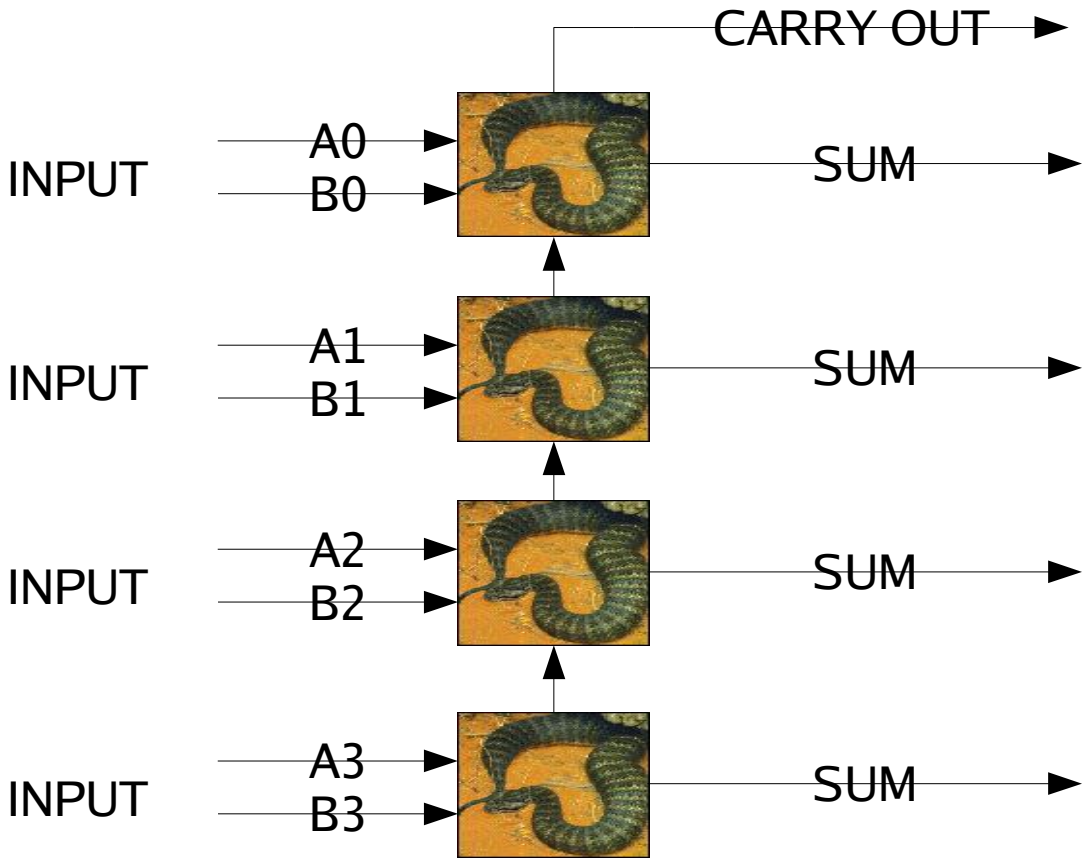
The 1-bit adder

- The 1-bit adder must have three inputs:
  - one for each of the two summand bits
  - and one for the bit carried in
- It must have two outputs:
  - one sum bit
  - and one carry out bit



The 4 1-bit adders

- Four 1-bit adders can be hooked up to produce a 4-bit adder:



NOTE: 8 inputs and
5 outputs, as promised!

The I/O Table of the 1-bit adder

A	B	Carry In	Carry Out	SUM BIT
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0