CS 130 : Computer Systems - III

Shankar Balachandran
Dept. of Computer Science & Engineering
IIT Madras

Recap : On Functions

a	f0
0	0
1	0

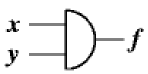
a	f1
0	0
1	1

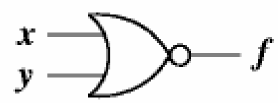
a	f2
0	1
1	0

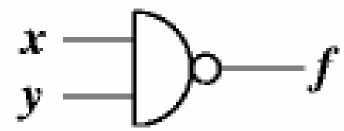
a	f3
0	1
1	1

a	b	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

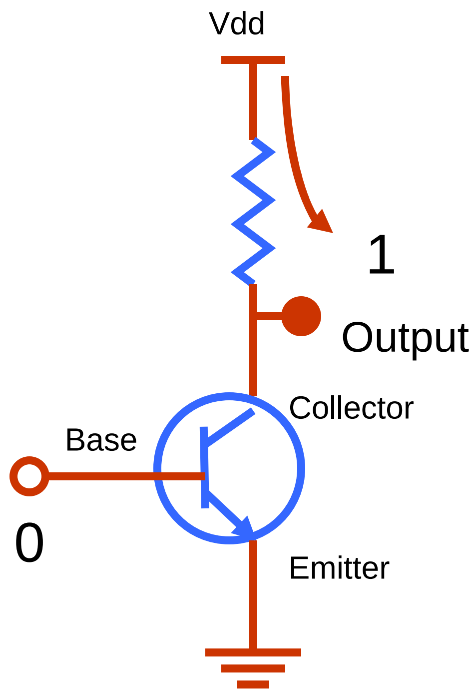
Recap : Abstraction Using Symbols, Equations and Tables

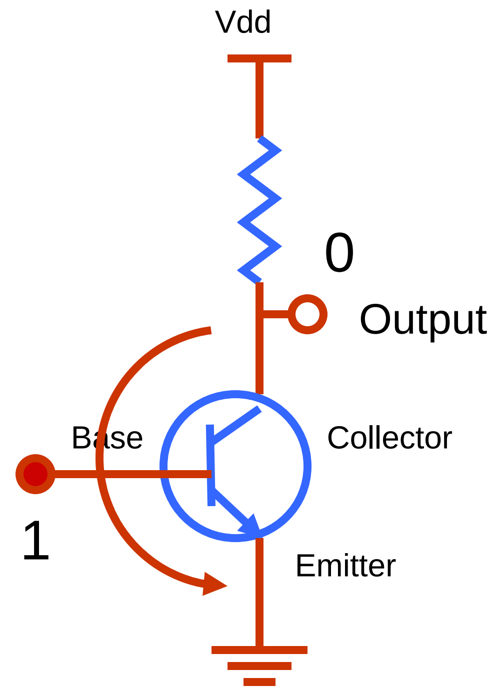
Gate	Schematic Symbol	Algebraic Function	Truth Table
BUFFER		$f = x$	x f 0 0 1 1
AND		$f = xy$	x y f 0 0 0 0 1 0 1 0 0 1 1 1
OR		$f = x + y$	x y f 0 0 0 0 1 1 1 0 1 1 1 1
XOR		$f = x \oplus y$	x y f 0 0 0 0 1 1 1 0 1 1 1 0



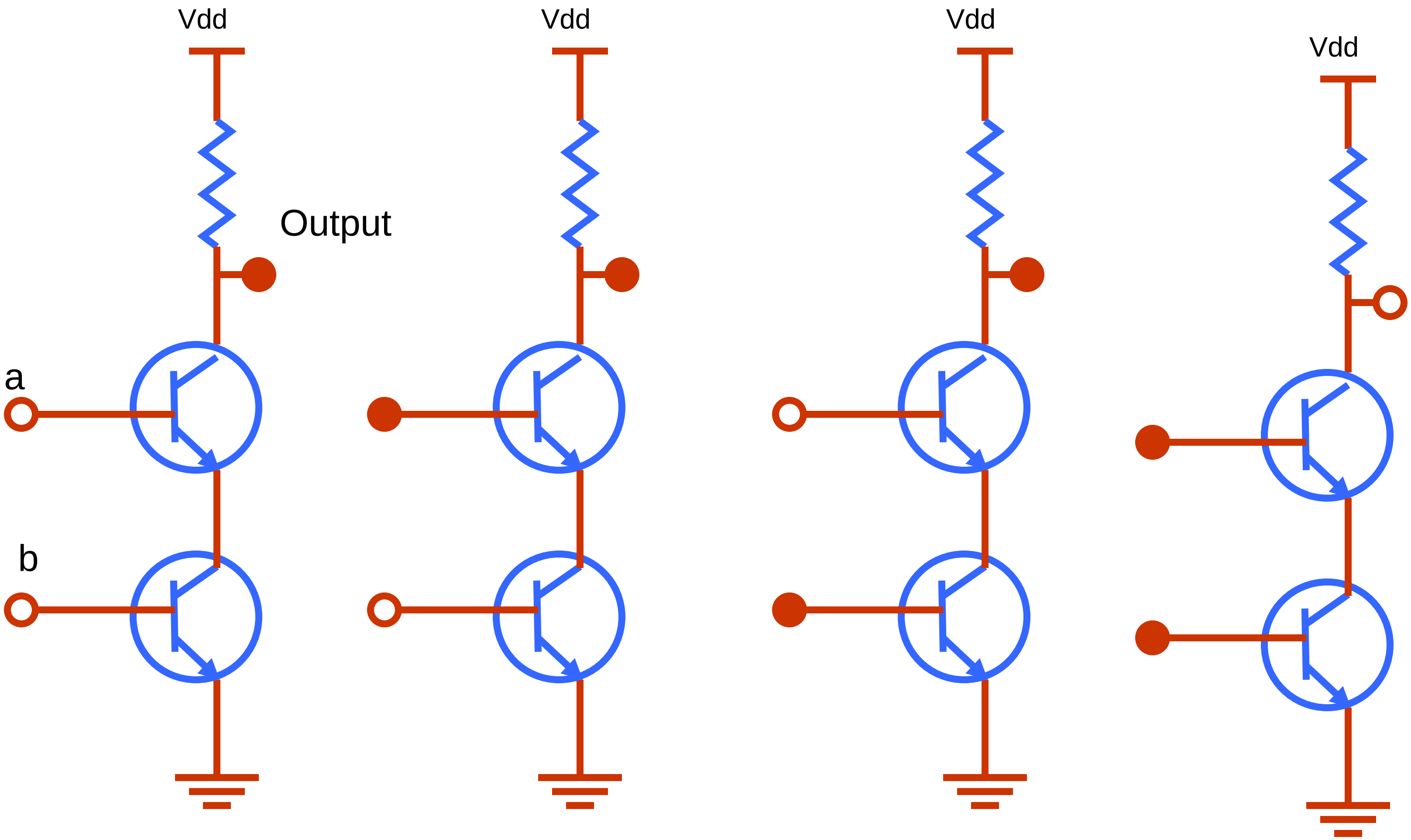


Electrical Equivalence NOT Gate (Inverter)

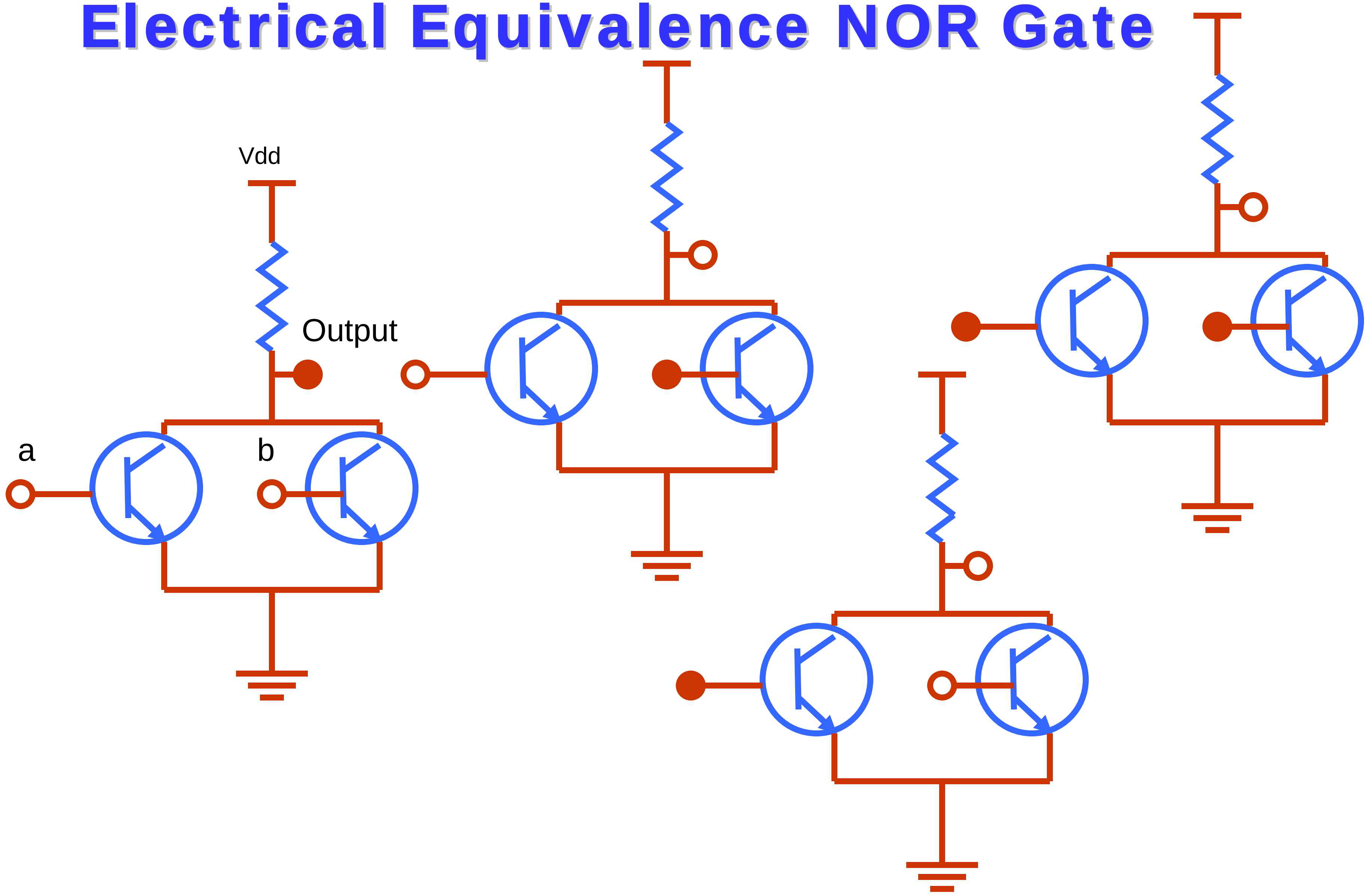




Electrical Equivalence NAND Gate

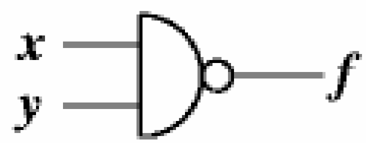


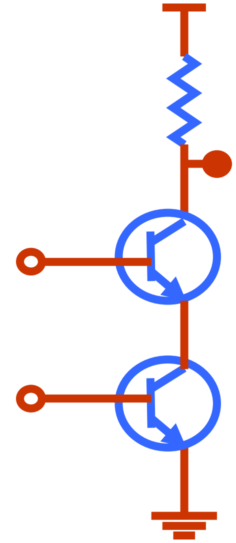
Electrical Equivalence NOR Gate



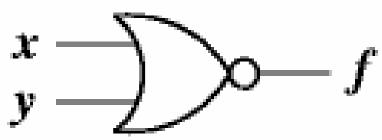
Equivalence

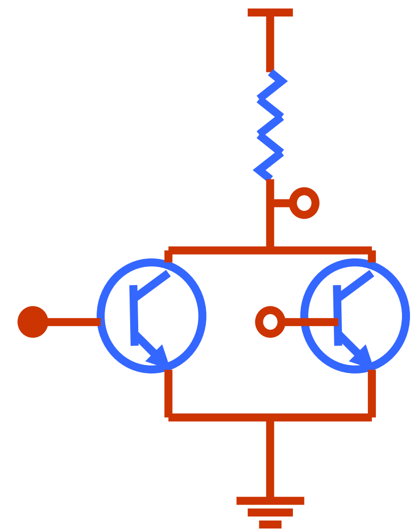
x	y	NAND
0	0	1
0	1	1
1	0	1
1	1	0





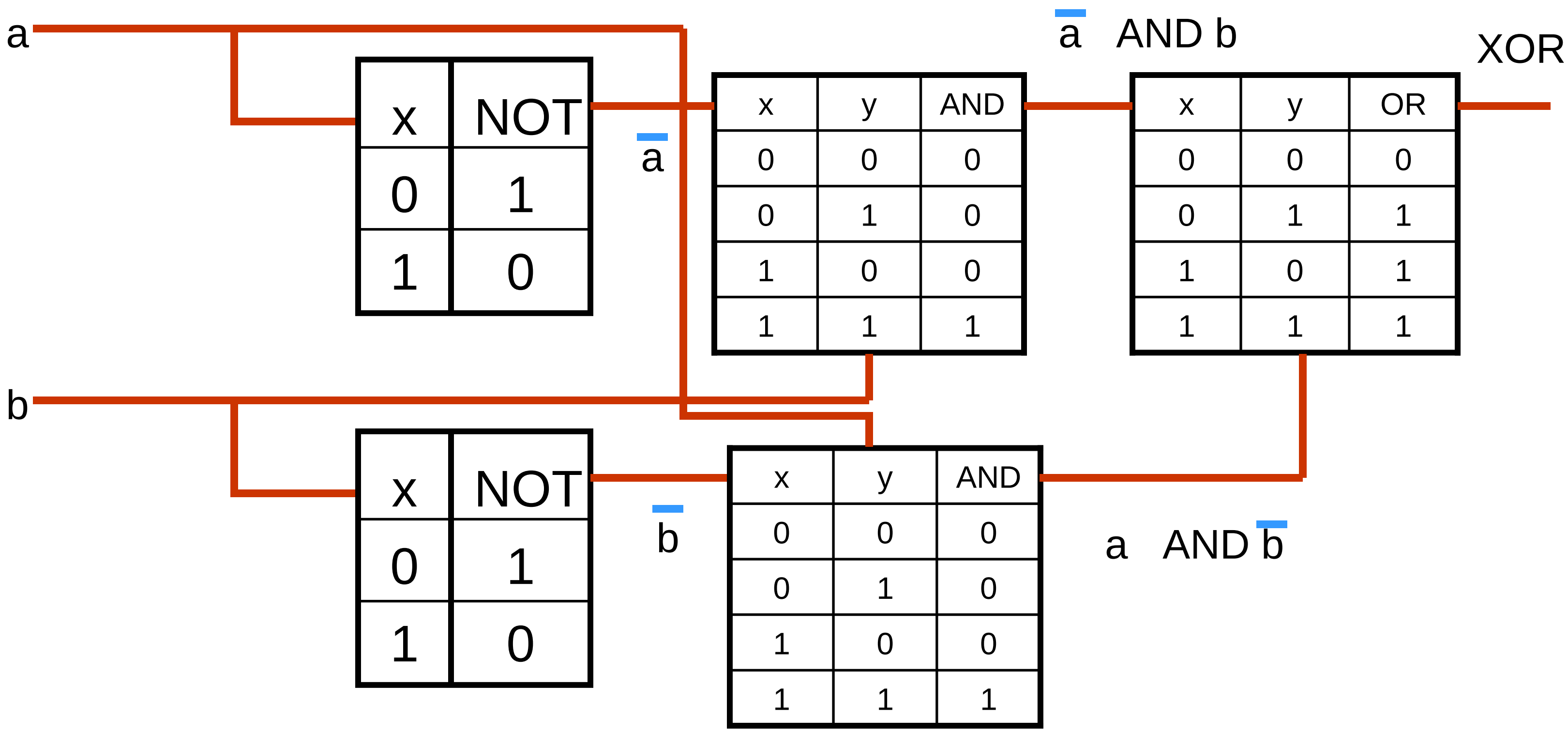
x	y	NOR
0	0	1
0	1	0
1	0	0
1	1	0



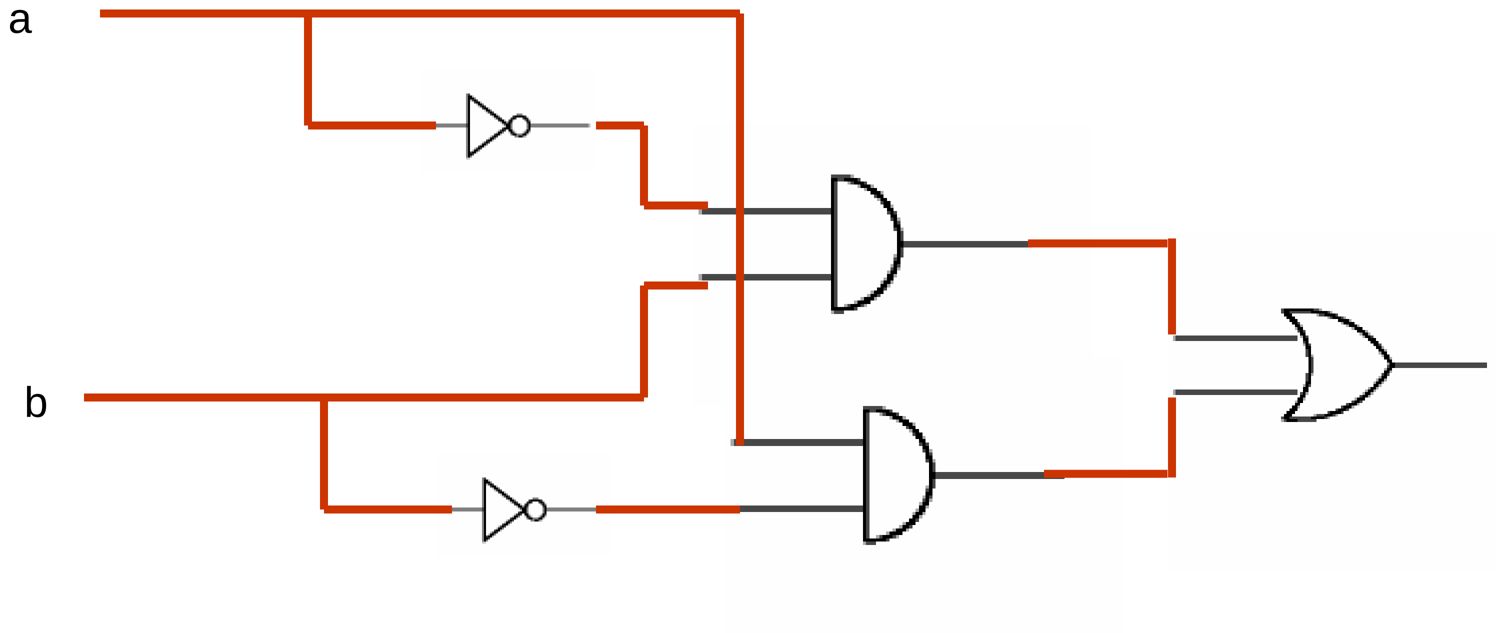


Let’s Implement XOR

- XOR(a,b) = ($\bar{a}$ AND b) OR (a AND $\bar{b}$)



XOR Using Gates



Electrical Circuits

- Voltages, Currents
- Resistors, transistors etc.
- Well known device behaviors
  - Equations for current, voltage etc.
- Lots of details !!
  - Work out XOR with transistors
- Evaluation of a function
  - Simulate and see
  - Apply rules of the devices

Truth Tables

- Logical function
  - Abstraction
- Do not have to worry about implementation
  - All nasty transistor diagrams are gone
- Machine understandable
  - You can look up along the row and column for values
- Enumeration
  - Can get clumsy very easily

Gates

- Abstraction of Functions
  - Different from truth tables
- Gates and Truth tables are equivalent
  - One can be transformed to another
- The abstraction using gates is also useful
  - Easy to draw schematics than write truth tables
    - Or writing functions directly
  - A nice visual representation

To Ponder

- Why need different levels of abstraction?
- Hints:
  - Human vs Computer understandability
  - Scalability
    - Larger functions
    - More variables
  - Automation
  - Correctness of results
    - Can you verify?
    - How can you automate verifying?

Let’s Do Some Arithmetic

- 5 + 5 =
- 34 + 65 =
- 139 + 217 =
- 558 + 404 =
- 2279 + 656 =
- 53412 + 21387 =
- 883092 + 642190 =

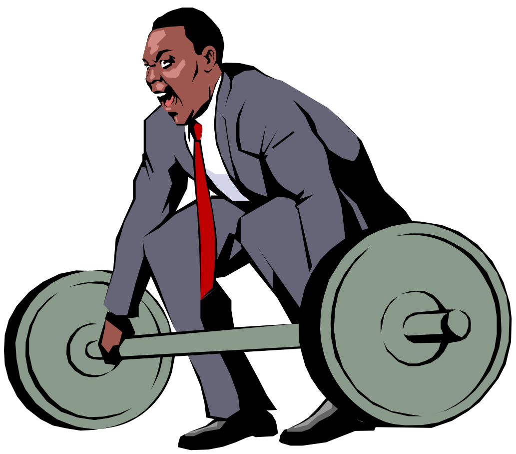
How Did You Perform the Arithmetic?

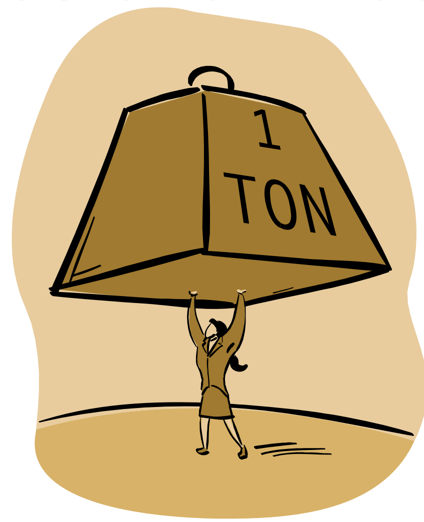
- Not how well, just how ☺
- For small numbers
  - Lookup sums in memory
    - 5 + 5 =
    - 34 + 65 =
- For medium sized numbers
  - Partly lookup, partly add
    - 139 + 217 =
    - 558 + 404 =
- For large numbers
  - Add starting from right; use carries etc.
    - 2279 + 656 =
    - 53412 + 21387 =
    - 883092 + 642190 =
  - For every digit, you looked up the value in your memory though

Let’s Think About Our Process

- Why lookup for small number arithmetic?
  - Have been doing it for a long time
  - Very few to remember
- What happened in medium numbers?
  - Do rough summations starting from left side
  - Adjust with values from the rightmost digits
  - Still manageable partly in your mind
- For large numbers :
  - Probably did not even consider starting from left
  - We know there is a methodical procedure
    - Play it safe??
  - Probably less error if you start from right
    - Easy to verify your answer also

Many Design Issues are Those of Scale





- The operation itself was simple : addition
  - Complexity at different stages are dealt with different strategies
  - Larger the numbers, a different approach was required
- The same strategy is not useful everywhere
- As problems get larger, more resources are required
- A methodical rather than ad-hoc technique must be employed

Let's Do Another Task

- Try and remember the results of the following operations and the sequence of the results
  - 3 + 5 =
  - 4 + 9 =
  - 14 + 27 =
  - 23 + 6 =
- Now, repeat the results to me :

Similar Task

- Remember the results
  - 4 + 3 =
  - 2 + 7 =
  - 1 + 5 =
  - 9 + 12 =
  - 3 + 8 =
  - 11 + 6 =
  - 8 + 4 =
  - 10 + 1 =
- Now, repeat the values to me :

What Happened Now?

- Operator is the same everywhere
- Operands are all small
  - Why unable to remember values in the second case?
- Computations were all small
- Remembering values were hard though
- Again an issue of scale
  - Few numbers, easy to remember
  - More of them, gets harder to remember
- Scale is not only an issue with operations
  - It is also an issue with storage