DM534

Introduction to Computer Science

Joan Boyar

September 4, 2013

Format

- Lectures (most in English)
  - Joan Boyar + other CS faculty
  - My office hours: Mondays 13:00–13:45, Fridays 8:30–9:15
  - Questions in English or Danish
- Labs and discussion sections
  - Uffe Thorsen S17
  - Magnus Gausdal Find S7
- Study groups (with and without advisors)

Studiestartsopgave

- Study start project
  - available from course homepage
  - due September 22, 23:59
  - turn in paper and through Blackboard — 1 PDF file
  - start early
  - read questions carefully
  - write clear, complete answers
  - explain your answers
  - do not write too much

Assignments

- assignments to be approved
  (6 – at most 2 retries total)
  - no working together
    (talk with me or instruktor)
  - no late assignments
  - turn in paper to me and via Blackboard – 1 PDF file
  - if sick, use a retry
  - must be nearly correct
  - grading – pass/fail

Discussion sections and labs

- Read notes/textbook sections
- Think about problems
- Prepare at least one problem to present
- There will be support for installing LaTeX, Java, etc.
  See DM536 schedule (also available through this course).

Computer Science

Computer Science is Not:

- Learning applications
- Programming

The course gives a broad overview.

Course Topics:

- Algorithms
- Computer architecture
- Operating systems
- Networks
- Database systems
- Theoretical limits
- Artificial intelligence
- Cryptology
- Software tools — LaTeX, Subversion (version control)
- Computers and society – study group topics

Computer Science

Computer science = Science of algorithms?????

Computer Science

Computer science = Science of algorithms?????

Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Computer science = Science of algorithms?????

Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Computer Science

Computer science = Science of algorithms?????

Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Computer Science

Computer science = Science of algorithms?????

Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Computer Science

Computer science = Science of algorithms?????

Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Computer science = Science of algorithms?????

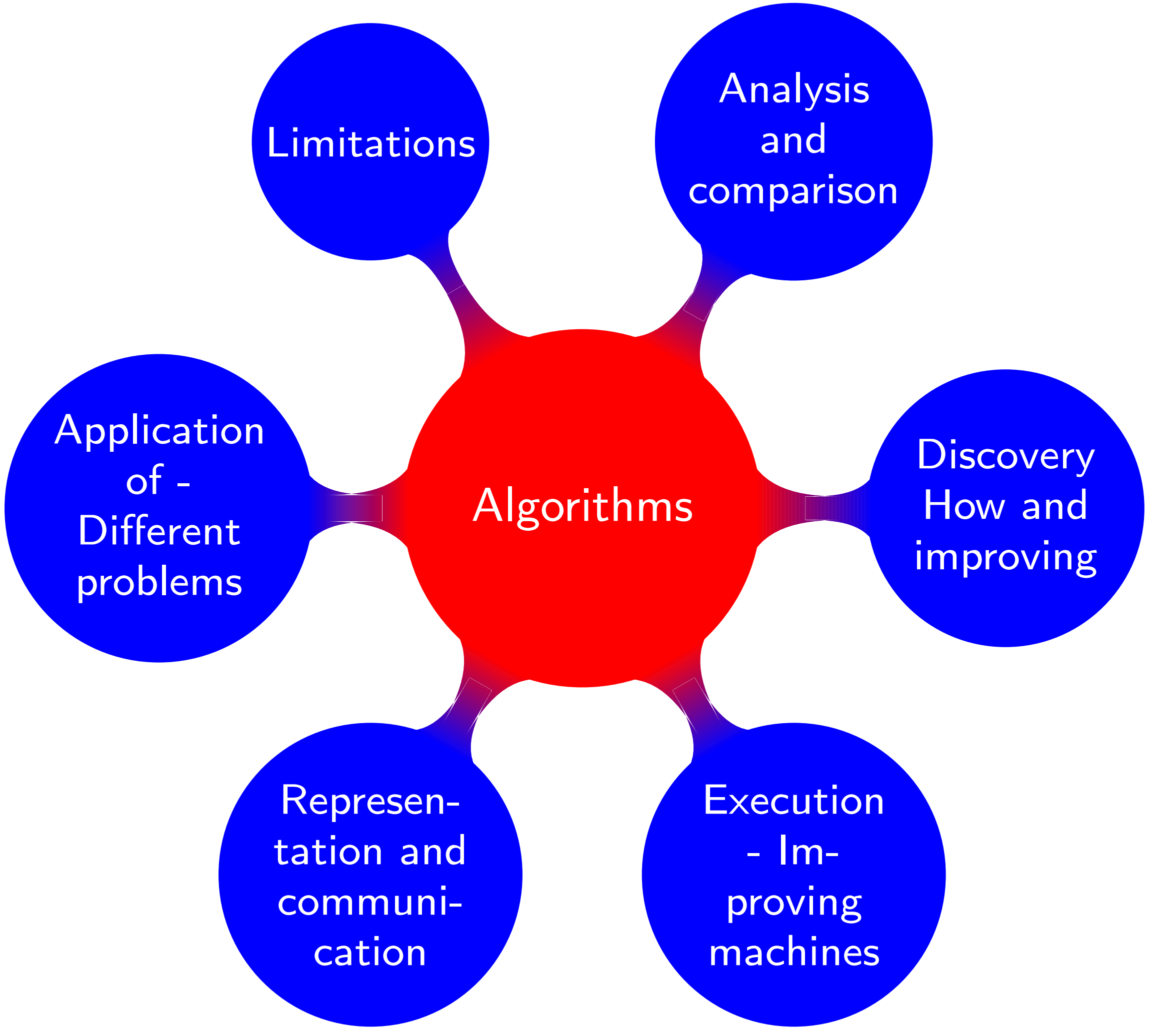
Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Computer Science

Computer science = Science of algorithms?????

Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Algorithms


Limitations
Analysis and comparison
Application of - Different problems
Algorithms
Discovery How and improving
Represen- tation and communi- cation
Execution - Im- proving machines

Greatest Common Divisor

gcd$(a, b) = \max\{g \mid g$ divides a and $b\}$

Examples:

gcd(15,9) = gcd(9,15) = 3

gcd(15,8) = gcd(8,15) = 1

Greatest Common Divisor

GCD(M, N):
{ Input: two positive integers M, N }
{ Output: gcd(M, N) }

$A \leftarrow \max(M, N)$
$B \leftarrow \min(M, N)$

$Q \leftarrow A$ div B
$R \leftarrow A - (Q \cdot B)$
while $R \neq 0$ **do**
  $A \leftarrow B$
  $B \leftarrow R$
  $Q \leftarrow A$ div B
  $R \leftarrow A - (Q \cdot B)$
return(B)

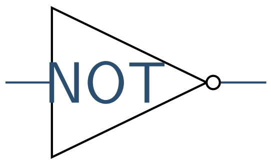
Types of data

The basic unit of data is a bit — 0 or 1.

Bit string — 01101000

- chars
- numbers
- images
- sounds
- truth values
  - 0 – false
  - 1 – true

Gates



$$\neg 0 = 1; \quad \neg 1 = 0;$$

x_1	NOT(x_1)
0	1
1	0

Gates

$$0 \wedge 0 = 0;\ 0 \wedge 1 = 0;\ 1 \wedge 0 = 0;\ 1 \wedge 1 = 1;$$

x_1	x_2	AND(x_1, x_2)
0	0	0
0	1	0
1	0	0
1	1	1

Gates

$$0 \lor 0 = 0;\ 0 \lor 1 = 1;\ 1 \lor 0 = 1;\ 1 \lor 1 = 1;$$

x_1	x_2	OR(x_1, x_2)
0	0	0
0	1	1
1	0	1
1	1	1

Gates

$$0 \oplus 0 = 0;\ \ 0 \oplus 1 = 1;\ \ 1 \oplus 0 = 1;\ \ 1 \oplus 1 = 0;$$

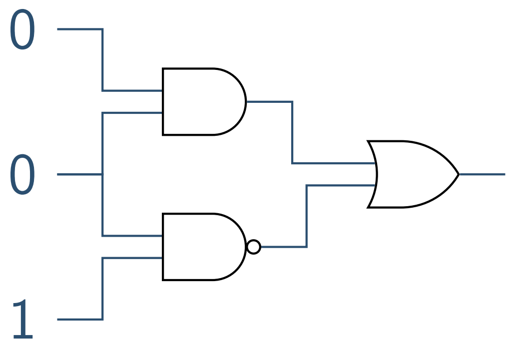
x_1	x_2	XOR(x_1, x_2)
0	0	0
0	1	1
1	0	1
1	1	0

Gates

0 nand $0 = 1$; 0 nand $1 = 1$; 1 nand $0 = 1$; 1 nand $1 = 0$;

x_1	x_2	NAND(x_1, x_2)
0	0	1
0	1	1
1	0	1
1	1	0

Example circuit



What are the top, bottom and rightmost gates?

A. AND, NAND, XOR

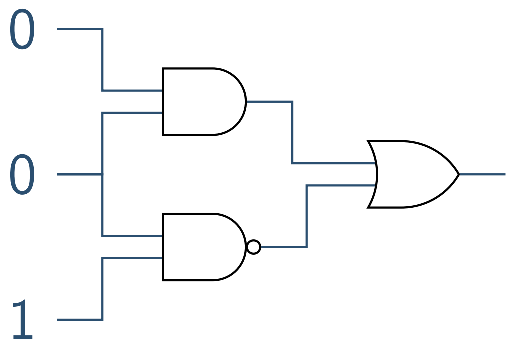
B. OR, NAND, XOR

C. AND, NAND, OR

D. OR, NAND, OR

Vote at m.socrative.com. Room number 415439.

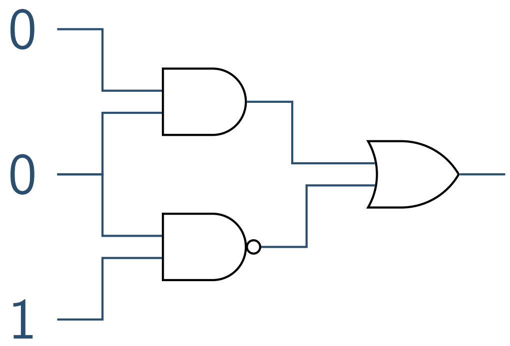
Example circuit



What are the top, bottom and rightmost gates?

C. AND, NAND, OR

Example circuit



What is the output of this circuit?

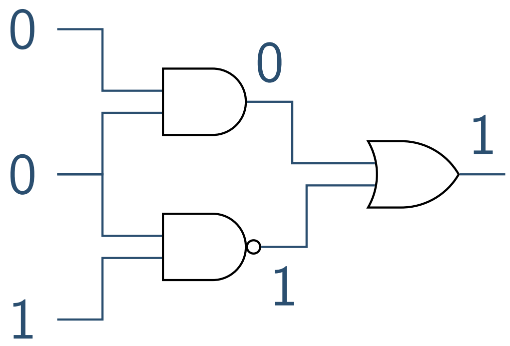
A. 0

B. 1

C. not defined

Vote at m.socrative.com. Room number 415439.

Example circuit

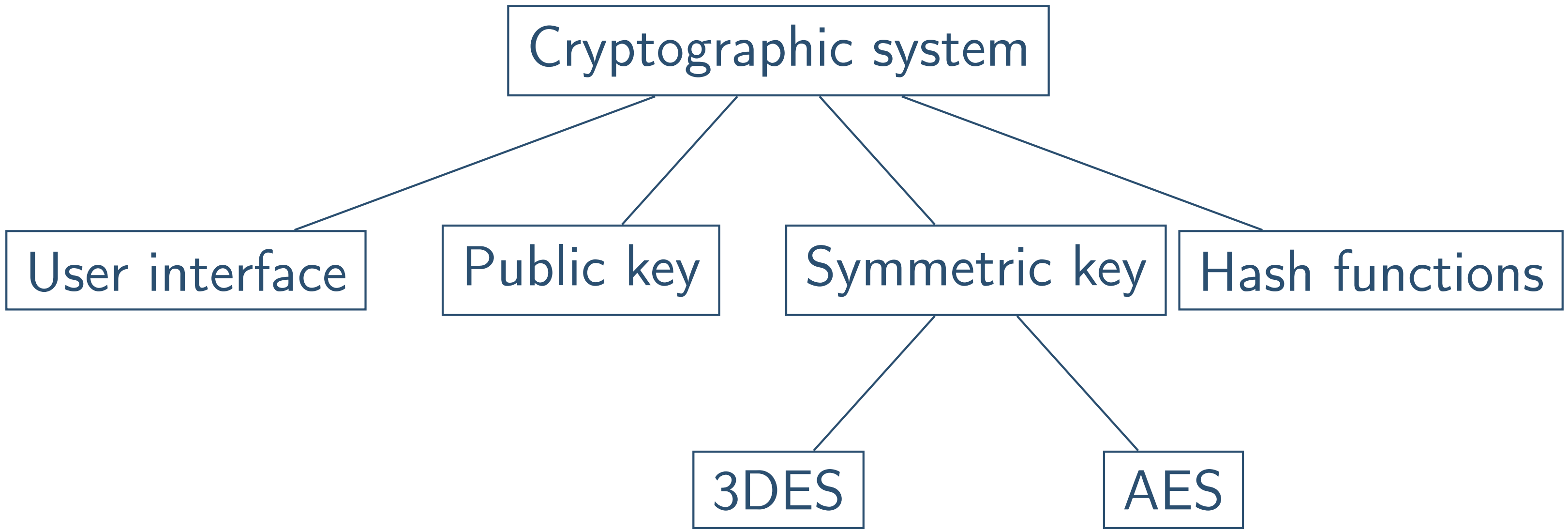


What is the output of this circuit?

B. 1

Abstraction

Example: Top-down design - cryptographic system



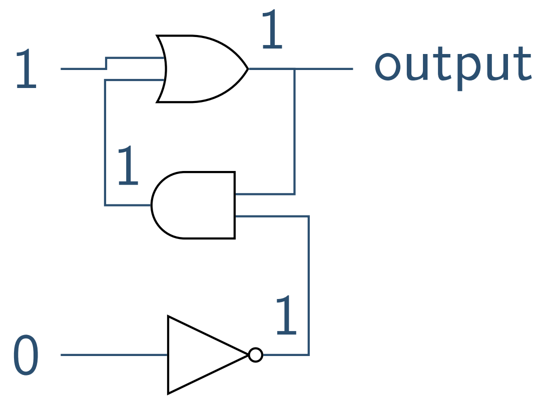
Abstraction

Things at higher levels need not know how things at lower levels function, only how to use them.

Interface, modularity, and modelling give:

- Structure — divide up work
- Independence between modules
  (can re-implement without changing the rest)
- Ability to analyze
- Increased innovation, productivity
  (don't need to re-invent the wheel)

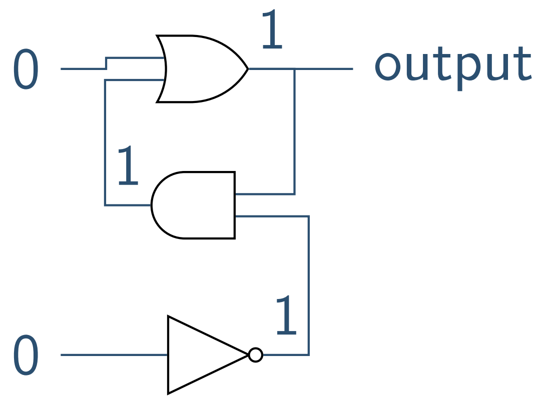
Flip flop



Note that this is stable.
Keeps same output until temporary outside pulse.
Can store a bit.

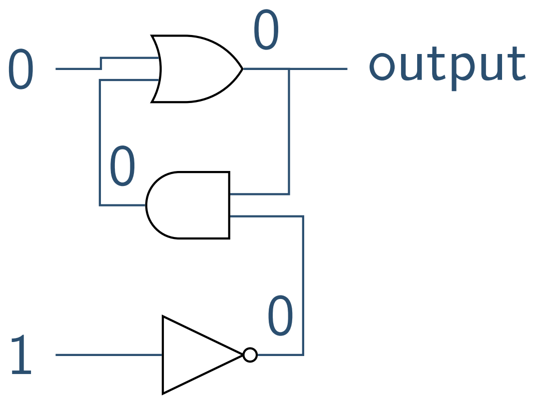
University of Southern Denmark

Flip flop



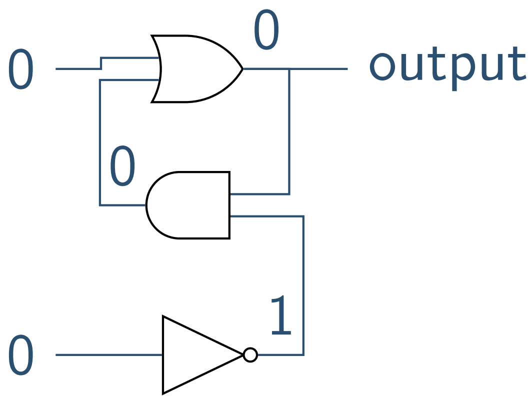
Note that this is stable.

Flip flop



Note that this is stable.

Flip flop



Note that this is stable.
But two different stable outputs are possible with input (0,0).

Flip flops can be implemented differently. Fig. 1.5, p. 38.
Abstraction: know input/output effect —
don't care about implementation.