DM534/DM558

Introduction to Computer Science

Joan Boyar

September 2, 2015

Format

- Lectures (most in English)
  - Joan Boyar + other CS faculty
  - Joan's office hours:
    Mondays 9:00–9:45, Fridays 9:00–9:45
  - Questions in English or Danish
- Labs and discussion sections
  - Kristine Vitting Klinkby Knudsen (D1)
  - Mathias W. Svendsen (D2)
  - Jesper With Mikkelsen (D3)
- Study groups (with and without advisors)

Studiestartsopgave

- Study start project
  - available from course homepage — with rules
  - due September 22, 8:15
  - turn in through Blackboard — 1 PDF file
  - start early
  - read questions carefully
  - write clear, complete answers
  - explain your answers, but do not write too much
  - no working together
  - must be essentially correct for pass
  - pick up from me after graded

Course requirements

- Pass/Fail
- 80% attendance at lectures, labs, and discussion sections
- All assignments approved
- Note: there is no formal exam

Assignments

- assignments to be approved
  (6 – at most 2 retries total)
  - no working together
    (talk with me or instruktor)
  - no late assignments
  - turn in via Blackboard – 1 PDF file
  - if sick, use a retry
  - must be nearly correct
  - grading – pass/fail (approved/not approved)

Assignments

- Begin early
- Ask if you do not understand
- Short, clear answers, but explain
- Do not reinvent the wheel
  it is fine to make minor modifications to something from the textbook or slides, just give a reference

Discussion sections and labs

- Read notes/textbook sections
- Think about problems
- Prepare at least one problem to present
- There will be an IMADA install “party” for LaTeX, Java, etc.

Computer Science

Computer Science is Not:

- Learning applications
- Programming

The course gives a broad overview.

Course Topics:

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

Computer Science

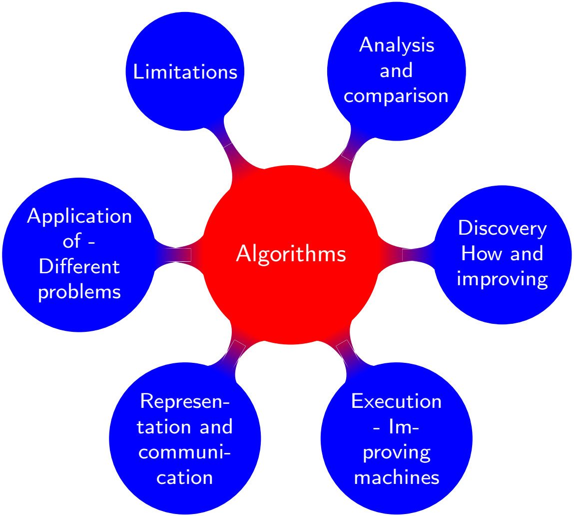
Computer science = Science of algorithms?????

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.

Algorithms



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

Greatest Common Divisor

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

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 := max(M, N)
B := min(M, N)

Q := A div B
R := A − (Q · B)
while R ≠ 0
begin
    A := B
    B := R
    Q := A div B
    R := A − (Q · B)
end
return(B)
```

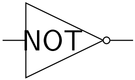
Types of data

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

Bit string — 01101000

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

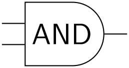
Gates



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

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	$\mathrm{AND}(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

Gates

$$0 \vee 0 = 0;\ \ 0 \vee 1 = 1;\ \ 1 \vee 0 = 1;\ \ 1 \vee 1 = 1;$$

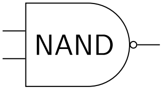
x_1	x_2	$\mathrm{OR}(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Gates

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

x_1	x_2	$\text{XOR}(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

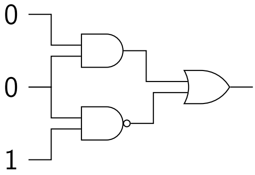
Gates



$$0 \text{ nand } 0 = 1;\ \ 0 \text{ nand } 1 = 1;\ \ 1 \text{ nand } 0 = 1;\ \ 1 \text{ nand } 1 = 0;$$

x_1	x_2	$\text{NAND}(x_1, x_2)$
0	0	1
0	1	1
1	0	1
1	1	0

Example circuit

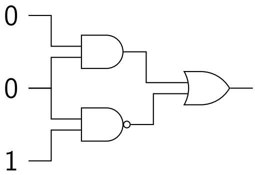


What are the top, bottom and rightmost gates?

A. AND, NAND, XOR
B. OR, NAND, XOR
C. AND, NAND, OR
D. OR, NAND, OR

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

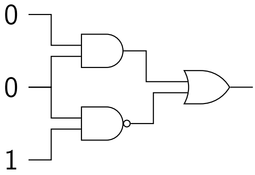
Example circuit



What are the top, bottom and rightmost gates?

C. AND, NAND, OR

Example circuit

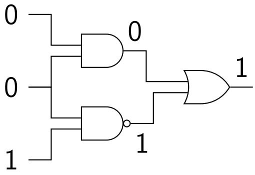


What is the output of this circuit?

A. 0
B. 1
C. not defined

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

Example circuit

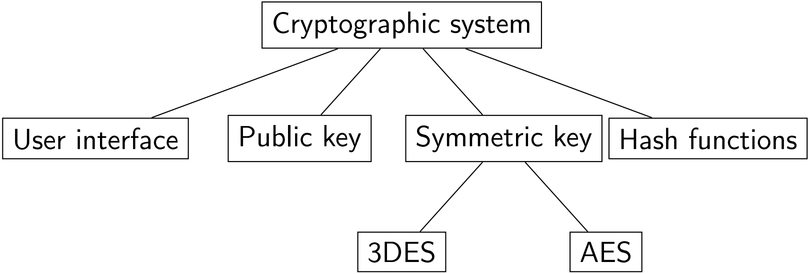


What is the output of this circuit?

B. 1

Abstraction

Example: Top-down design - cryptographic system



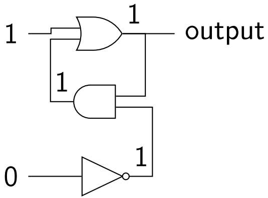
Abstraction

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

Interface, modularity, and modelling give:

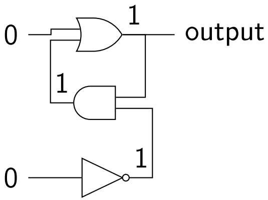
- Structure — divide up work
- Independence between modules
  (can re-implement without changing the rest)
- Ability to analyze
- Increased innovation, productivity
  (don't need to re-invent the wheel)

Flip flop



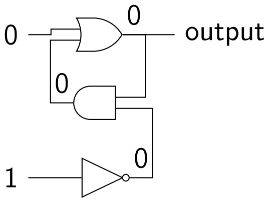
Note that this is stable.
Keeps same output until temporary outside pulse.
Can store a bit.

Flip flop



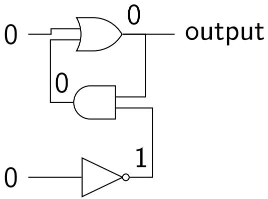
Note that this is stable.

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. 36.
Abstraction: know input/output effect —
don't care about implementation.