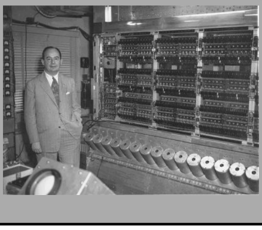
Lecture 1

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Computer Systems Overview
Digital Logic Combination Circuits

CO-I Ch 1-8 [Sta10]
Some material from
Comp. Org I

Digital Logic, Ch 20.1-3



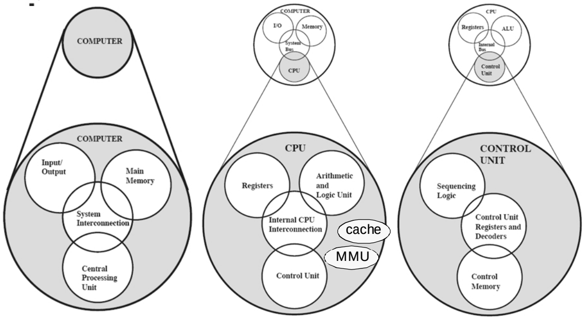
John von Neumann
and EDVAC, 1949

Overview Content

- Structure
- OS view point
- Buses
- I/O-controller and memory-mapped I/O
- Memory hierarchy
- I/O layers
- Privileged mode
- Instruction cycle
- Interrupt handling
- Goal:
  - Remember what has already been covered on Comp. Org I

Structure of a computer

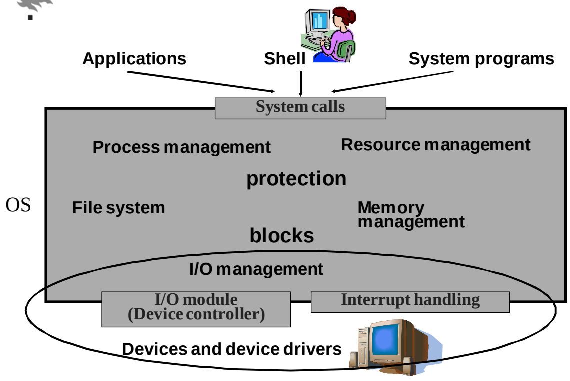
Hardware vs Software



Control, Processing, Storage, Data movement

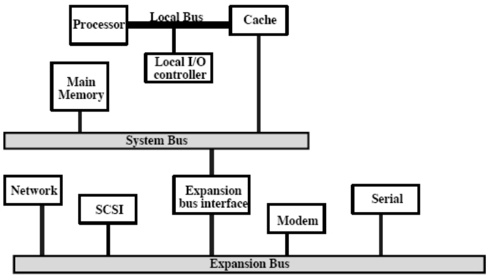
(Sta10 Fig 1.4)

Operating System's view point



Buses

- Local (*Sisäinen*), System, I/O expansion
- Device controllers (*Laiteohjaimet*), NOTE: Stal10: "I/O module"

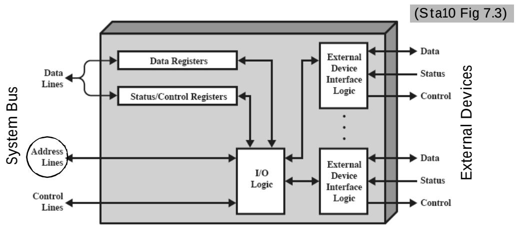


(Sta10 Fig 3.18 a)

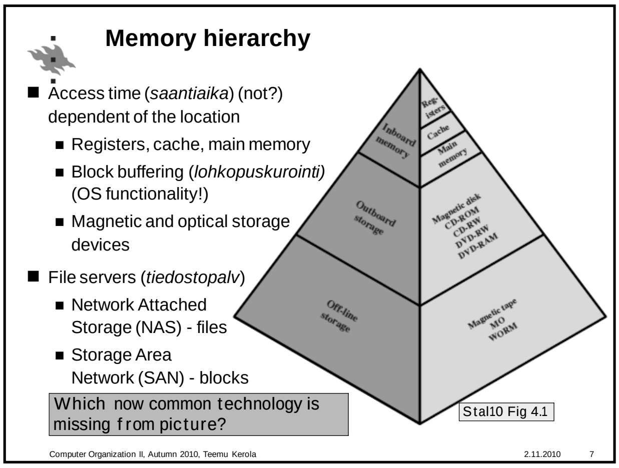
(a) Traditional Bus Architecture

I/O controller and memory-mapped I/O

(Sta10 Fig 7.3)



- Device driver (*ajuri*) controls the device via controller's registers
- Driver refers to these registers as regular memory locations
  - Common memory references, like in load/store -instructions
  - Controller (*ohjain*) detects its own memory addresses on the bus
  - Device controller ~ 'intelligent' memory location

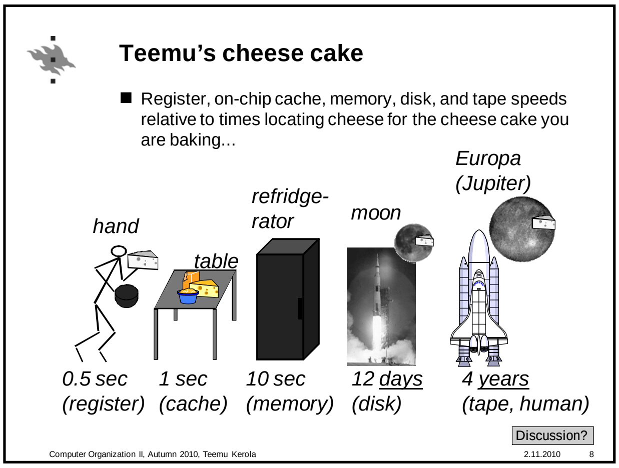


Memory hierarchy

- Access time (*saantiaika*) (not?) dependent of the location
  - Registers, cache, main memory
  - Block buffering (*lohkopuskurointi)* (OS functionality!)
  - Magnetic and optical storage devices
- File servers (*tiedostopalv*)
  - Network Attached Storage (NAS) - files
  - Storage Area Network (SAN) - blocks

Which now common technology is missing from picture?

Stal10 Fig 4.1

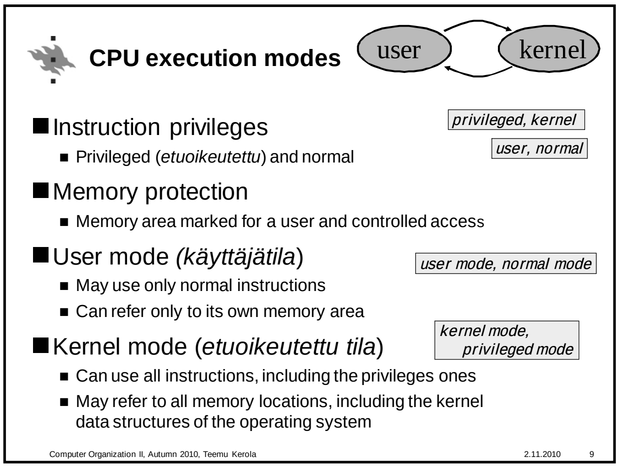


Teemu’s cheese cake

- Register, on-chip cache, memory, disk, and tape speeds relative to times locating cheese for the cheese cake you are baking...

0.5 sec	*1 sec*	*10 sec*	*12 days*	*4 years*
(register)	*(cache)*	*(memory)*	*(disk)*	*(tape, human)*

Discussion?



CPU execution modes

user
kernel

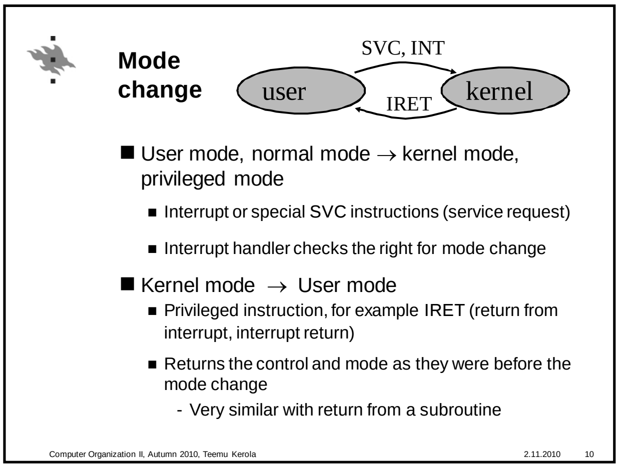
- Instruction privileges
  - Privileged (*etuoikeutettu*) and normal
- Memory protection
  - Memory area marked for a user and controlled access
- User mode (*käyttäjätila*)
  - May use only normal instructions
  - Can refer only to its own memory area
- Kernel mode (*etuoikeutettu tila*)
  - Can use all instructions, including the privileges ones
  - May refer to all memory locations, including the kernel data structures of the operating system

privileged, kernel

user, normal

user mode, normal mode

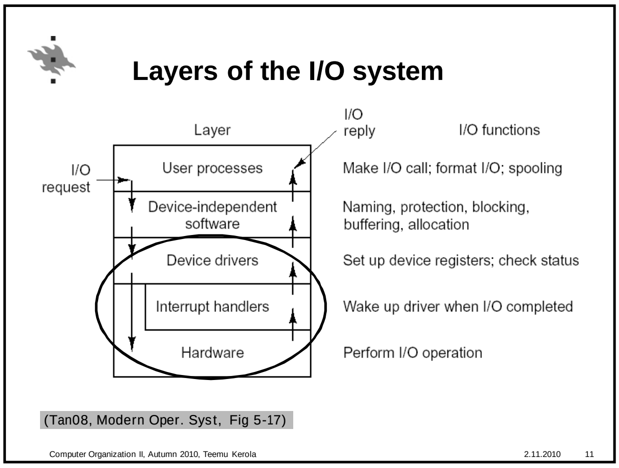
kernel mode, privileged mode



Mode change

SVC, INT
user
kernel
IRET

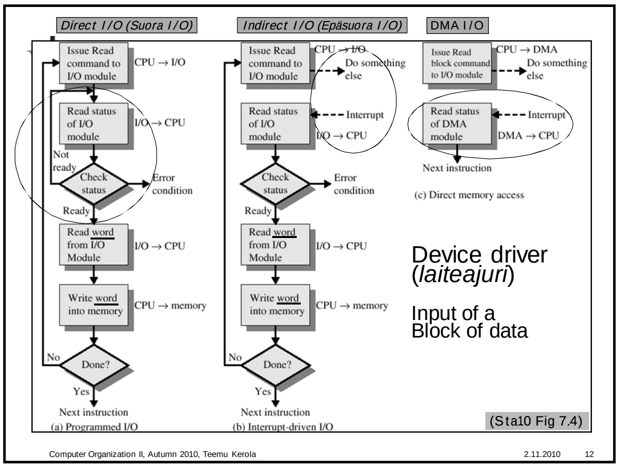
- User mode, normal mode → kernel mode, privileged mode
  - Interrupt or special SVC instructions (service request)
  - Interrupt handler checks the right for mode change
- Kernel mode → User mode
  - Privileged instruction, for example IRET (return from interrupt, interrupt return)
  - Returns the control and mode as they were before the mode change
    - Very similar with return from a subroutine



Layers of the I/O system

Layer	I/O functions
User processes	Make I/O call; format I/O; spooling
Device-independent software	Naming, protection, blocking, buffering, allocation
Device drivers	Set up device registers; check status
Interrupt handlers	Wake up driver when I/O completed
Hardware	Perform I/O operation

(Tan08, Modern Oper. Syst, Fig 5-17)

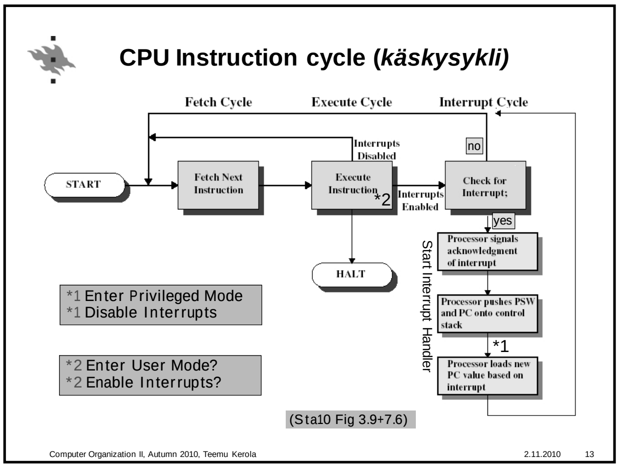


Direct I/O (Suora I/O) *Indirect I/O (Epäsuora I/O)* DMA I/O

Device driver (*laiteajuri*)

Input of a Block of data

(Sta10 Fig 7.4)



CPU Instruction cycle (*käskysykli)*

Fetch Cycle

Execute Cycle

Interrupt Cycle

START

Fetch Next Instruction

Execute Instruction *2

Interrupts Disabled

Interrupts Enabled

Check for Interrupt;

no

yes

HALT

Start Interrupt Handler

Processor signals acknowledgment of interrupt

Processor pushes PSW and PC onto control stack

*1

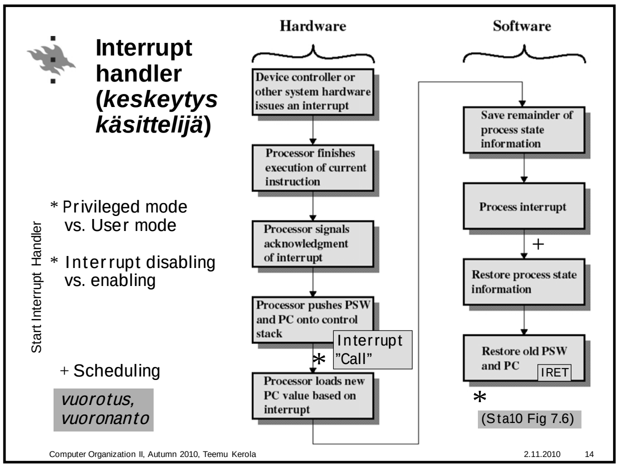
Processor loads new PC value based on interrupt

*1 Enter Privileged Mode
*1 Disable Interrupts

*2 Enter User Mode?
*2 Enable Interrupts?

(Sta10 Fig 3.9+7.6)

Computer Organization II, Autumn 2010, Teemu Kerola 2.11.2010 13



Interrupt handler (*keskeytys käsittelijä*)

Start Interrupt Handler

* Privileged mode
vs. User mode

* Interrupt disabling
vs. enabling

+ Scheduling

vuorotus, vuoronanto

Hardware

Device controller or other system hardware issues an interrupt

Processor finishes execution of current instruction

Processor signals acknowledgment of interrupt

Processor pushes PSW and PC onto control stack

Interrupt "Call"

*

Processor loads new PC value based on interrupt

Software

Save remainder of process state information

Process interrupt

+

Restore process state information

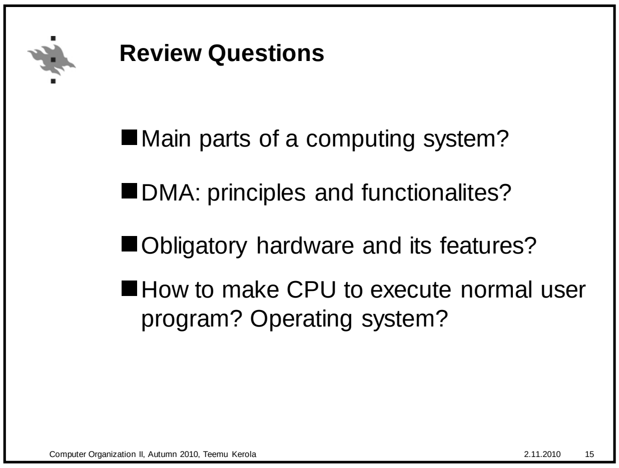
Restore old PSW and PC

IRET

*

(Sta10 Fig 7.6)

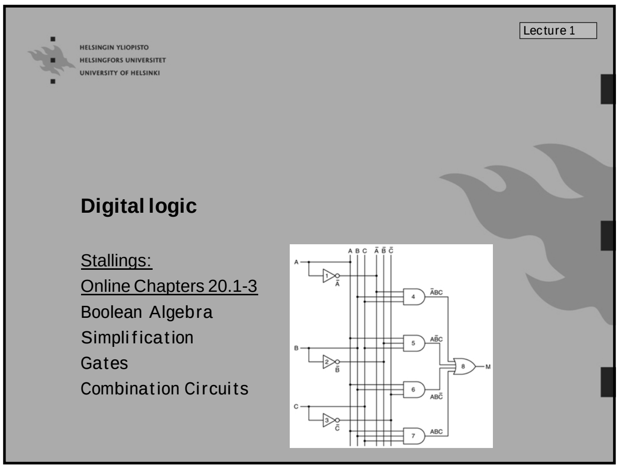
Computer Organization II, Autumn 2010, Teemu Kerola 2.11.2010 14



Review Questions

- Main parts of a computing system?
- DMA: principles and functionalites?
- Obligatory hardware and its features?
- How to make CPU to execute normal user program? Operating system?

Computer Organization II, Autumn 2010, Teemu Kerola 2.11.2010 15

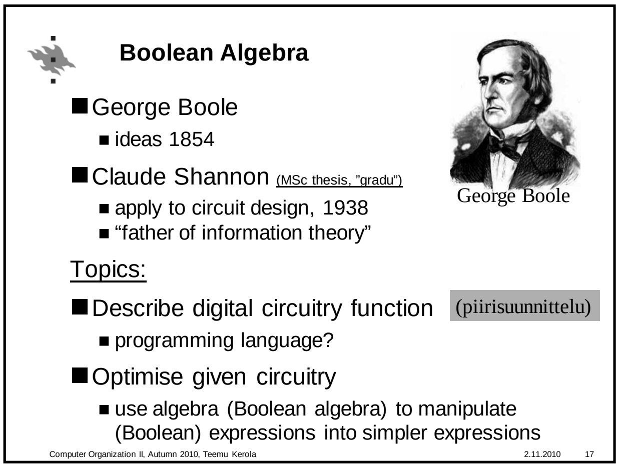


Lecture 1

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Digital logic

Stallings:
Online Chapters 20.1-3
Boolean Algebra
Simplification
Gates
Combination Circuits



Boolean Algebra

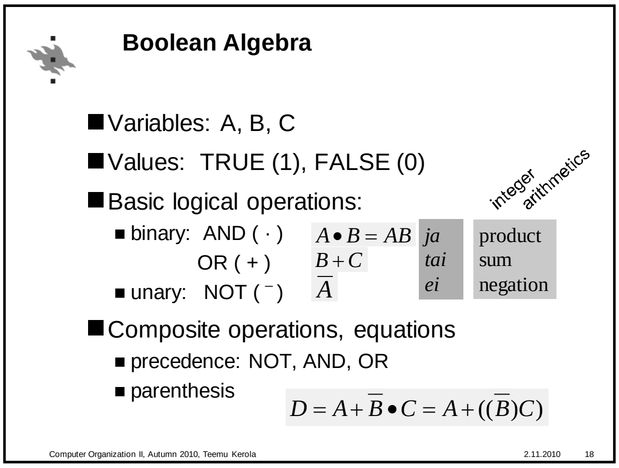
- George Boole
  - ideas 1854
- Claude Shannon (MSc thesis, "gradu")
  - apply to circuit design, 1938
  - "father of information theory"

George Boole

Topics:

- Describe digital circuitry function (piirisuunnittelu)
  - programming language?
- Optimise given circuitry
  - use algebra (Boolean algebra) to manipulate (Boolean) expressions into simpler expressions

Computer Organization II, Autumn 2010, Teemu Kerola 2.11.2010 17



Boolean Algebra

- Variables: A, B, C
- Values: TRUE (1), FALSE (0)
- Basic logical operations:
  - binary: AND (·) $A \bullet B = AB$ *ja* product
    OR (+) $B + C$ *tai* sum
  - unary: NOT (¯) $\overline{A}$ *ei* negation

integer arithmetics

- Composite operations, equations
  - precedence: NOT, AND, OR
  - parenthesis $D = A + \overline{B} \bullet C = A + ((\overline{B})C)$

Computer Organization II, Autumn 2010, Teemu Kerola 2.11.2010 18

Boolean Algebra

- Other operations
  - XOR (exclusive-or)
  - NAND
  - NOR

$$A \text{ NAND } B = \text{NOT}\,(A \text{ AND } B) = \overline{AB}$$

$$A \text{ NOR } B = \text{NOT}\,(A \text{ OR } B) = \overline{A+B}$$

- Truth tables
  - What is the result of the operation?

function

input

Boolean Operators

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P NAND Q	P NOR Q
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	1	0
1	1	0	1	1	0	0	0

(Sta20 Table 20.1)

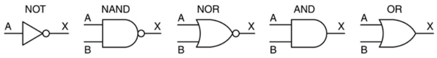
Postulates and Identities

- How can I manipulate expressions?
  - Simple set of rules?

Basic Postulates			
$A \bullet B = B \bullet A$	$A + B = B + A$	Commutative Laws	*vaihdantalaki*
$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$	$A + (B \bullet C) = (A + B) \bullet (A + C)$	Distributive Laws	*osittelulaki*
$1 \bullet A = A$	$0 + A = A$	Identity Elements	*neutraalialkiot*
$A \bullet \overline{A} = 0$	$A + \overline{A} = 1$	Inverse Elements	*alkion ja komplementin tulo ja summa*
Other Identities			
$0 \bullet A = 0$	$1 + A = 1$		*tulo 0'n kanssa, summa 1'n kanssa*
$A \bullet A = A$	$A + A = A$		*tulo ja summa itsensä kanssa*
$A \bullet (B \bullet C) = (A \bullet B) \bullet C$	$A + (B + C) = (A + B) + C$	Associative Laws	*liitäntälait*
$\overline{A \bullet B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \bullet \overline{B}$	DeMorgan's Theorem	

(Sta10 Table 20.2)

Gates, circuits, combination circuits



- Implement basic Boolean algebra operations
- Gates - fundamental building blocks
  - 1 or 2 inputs, 1 output

veräjät, portit ja piirit

- Combine to build more complex circuits
  - memory, adder, multiplier, …
- Gate delay in **combination circuits**

yhdistelmäpiirit

  - change inputs, after (combined) gate delay new output available
  - 1 ns? 10 ns? 0.1 ns?

http://tech-www.informatik.uni-hamburg.de/applets/cmos/cmosdemo.html (extra material)

Describing the Circuit

a) Boolean equations

$$F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$$

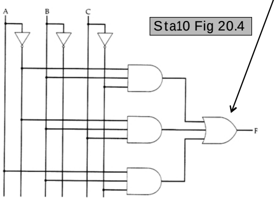
b) Truth table

<------------- inputs	----------->		<- output ->
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

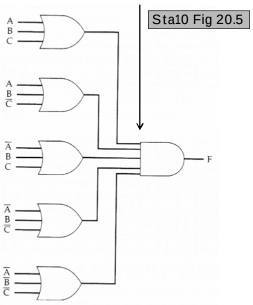
(Sta10 Table 20.3)

c) Graphical symbols
(next slide)

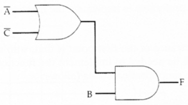
Graphical Symbols, $F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$ Sum-of-Products, Product-of-sums



Sta10 Fig 20.4


A B C
A B $\overline{C}$
$\overline{A}$ B C
$\overline{A}$ B $\overline{C}$
$\overline{A}$ $\overline{B}$ $\overline{C}$
F

Sta10 Fig 20.5



Sta10 Fig 20.6

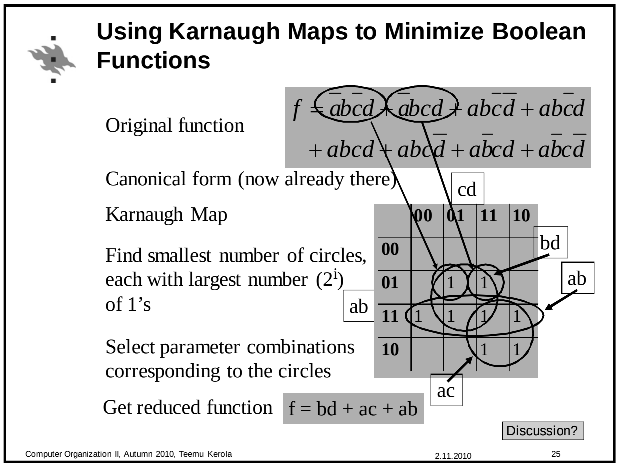
Discussion?

Simplification of Circuits

- Algebraic Simplification

$$F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$$
$$= \overline{A}B + B\overline{C} = B(\overline{A} + \overline{C})$$

- Simplification with Karnaugh Maps
  - E.g., see Kerola slides 2003/appa
    (http://www.cs.helsinki.fi/group/nodes/kurssit/tikra/2003s/luennot/appa_v.pdf)



Using Karnaugh Maps to Minimize Boolean Functions

Original function

$$f = \bar{a}b\bar{c}d + \bar{a}bcd + ab\bar{c}\bar{d} + abc\bar{d}$$
$$+ abcd + ab\bar{c}d + a\bar{b}cd + a\bar{b}c\bar{d}$$

Canonical form (now already there)

Karnaugh Map

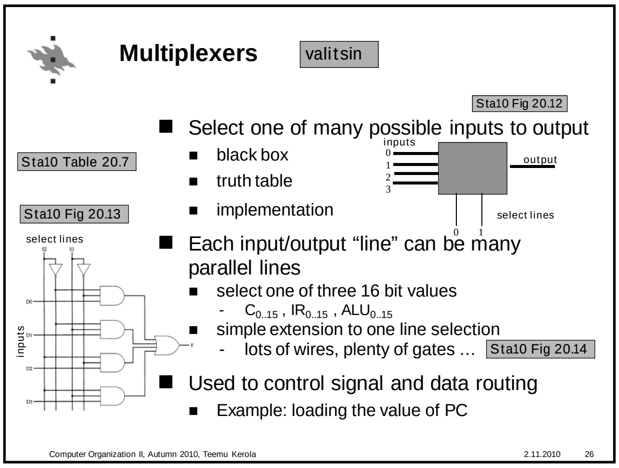
Find smallest number of circles, each with largest number (2^i) of 1’s

Select parameter combinations corresponding to the circles

	00	01	11	10
00				
01		1	1	
11	1	1	1	1
10			1	1

Get reduced function f = bd + ac + ab

Discussion?



Multiplexers

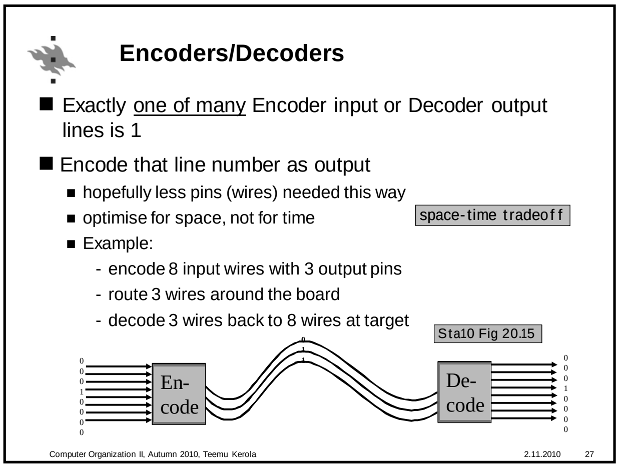
valitsin

Sta10 Fig 20.12

Sta10 Table 20.7

Sta10 Fig 20.13

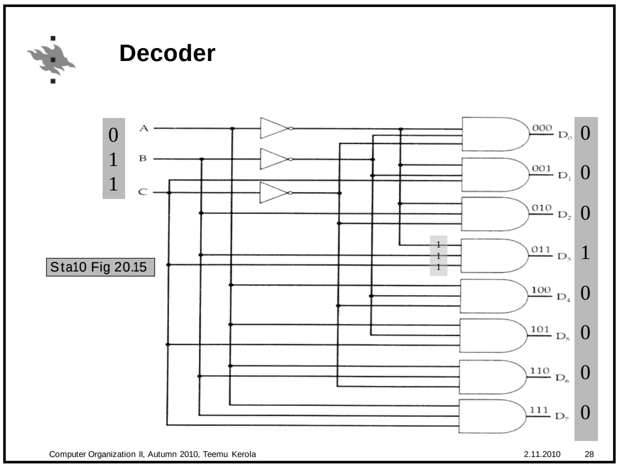
- Select one of many possible inputs to output
  - black box
  - truth table
  - implementation
- Each input/output “line” can be many parallel lines
  - select one of three 16 bit values
    - $C_{0..15}$, $IR_{0..15}$, $ALU_{0..15}$
  - simple extension to one line selection
    - lots of wires, plenty of gates … Sta10 Fig 20.14
- Used to control signal and data routing
  - Example: loading the value of PC



Encoders/Decoders

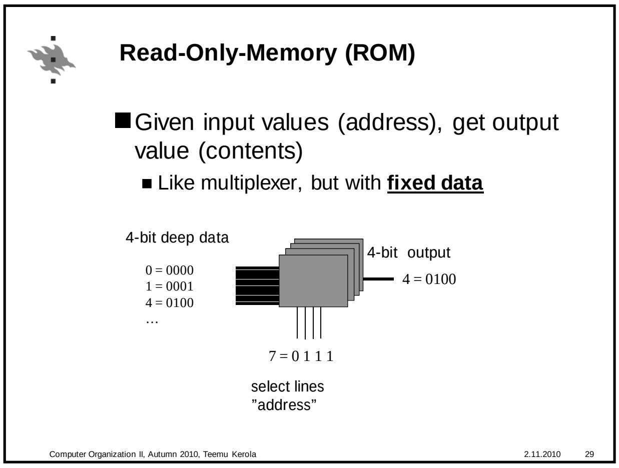
- Exactly one of many Encoder input or Decoder output lines is 1
- Encode that line number as output
  - hopefully less pins (wires) needed this way
  - optimise for space, not for time — space-time tradeoff
  - Example:
    - encode 8 input wires with 3 output pins
    - route 3 wires around the board
    - decode 3 wires back to 8 wires at target

Sta10 Fig 20.15



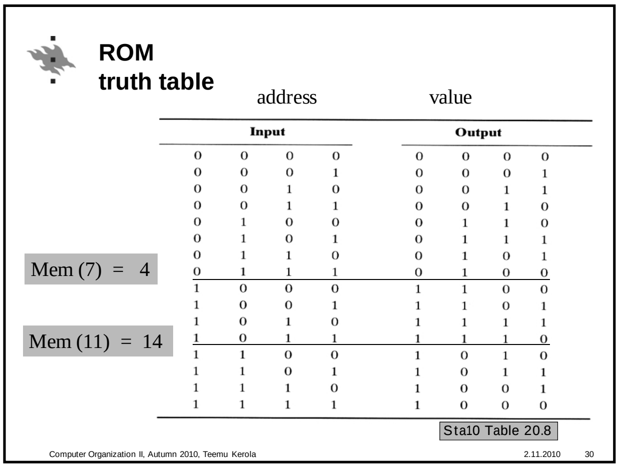
Decoder

Sta10 Fig 20.15



Read-Only-Memory (ROM)

- Given input values (address), get output value (contents)
  - Like multiplexer, but with **fixed data**



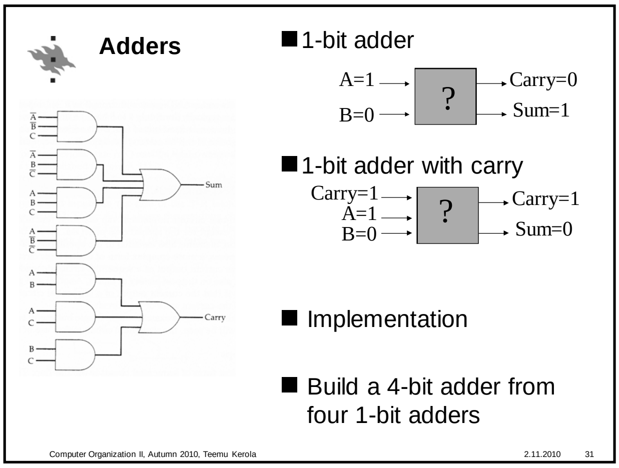
ROM truth table

Mem (7) = 4

Mem (11) = 14

address				value			
Input				Output			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Sta10 Table 20.8



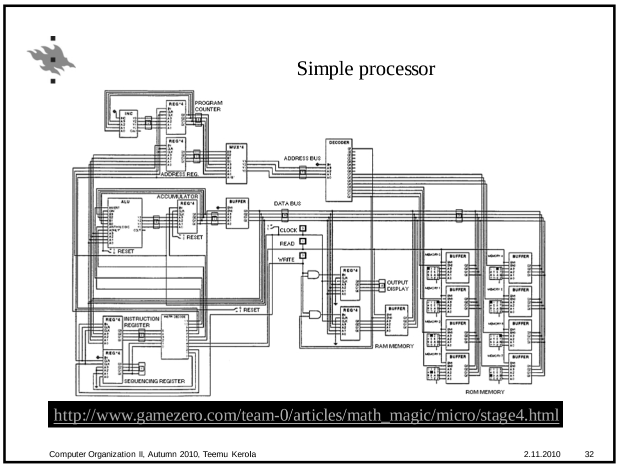
Adders

- 1-bit adder

- 1-bit adder with carry

- Implementation
- Build a 4-bit adder from four 1-bit adders

Computer Organization II, Autumn 2010, Teemu Kerola 2.11.2010 31



Simple processor

http://www.gamezero.com/team-0/articles/math_magic/micro/stage4.html

Computer Organization II, Autumn 2010, Teemu Kerola 2.11.2010 32