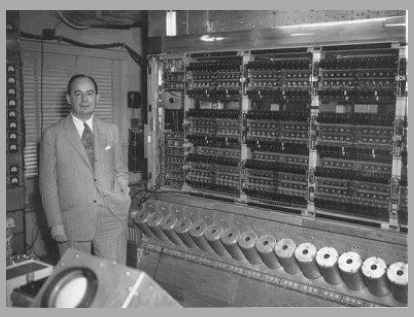
HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Lecture 1

Computer systems- overview

Ch 1 - Ch 8 [Sta06]
Some material from
Comp. Org I



John von Neumann
and EDVAC, 1949

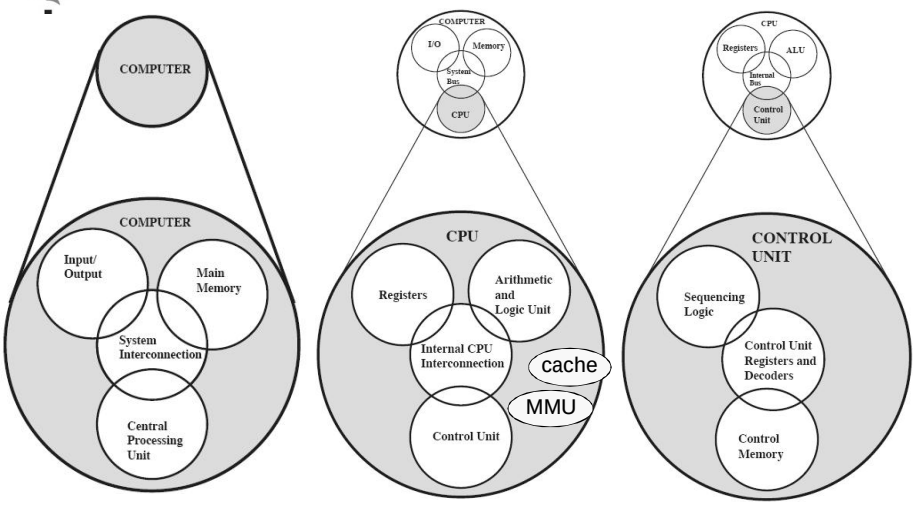
Content

- 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:
  - Remind what has already been covered on Comp. Org I

Structure of a computer (3)

Hardware vs Software



Control, Processing, Storage, Data movement

(Sta06 Fig 1.4, 1.5, 1.6)

Operating System's view point

User

Applications Shell System programs

System calls

Process management Resource management

protection

OS

File system Memory management

blocks

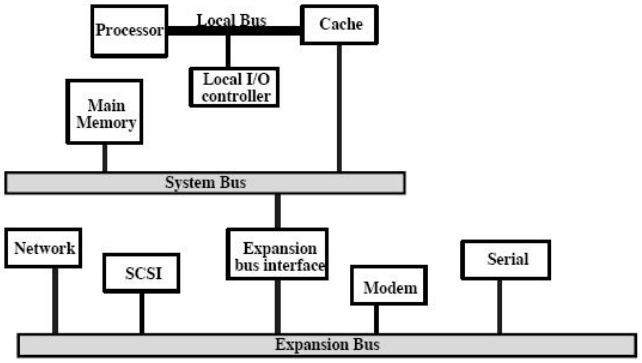
I/O management

I/O module (Device controller) Interrupt handling

Devices and device drivers

Buses

- Local (*Sisäinen*), System, I/O expansion
- Device controllers (*Laiteohjaimet*), NOTE: Sta06: I/O module

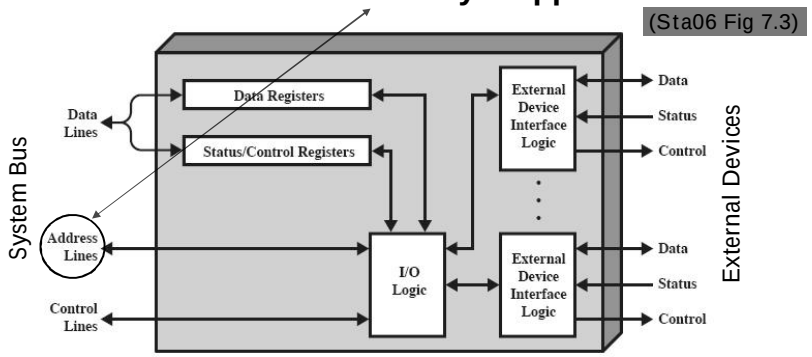


(a) Traditional Bus Architecture

(Sta06 Fig 3.18 a)

I/O controller and memory-mapped I/O

(Sta06 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

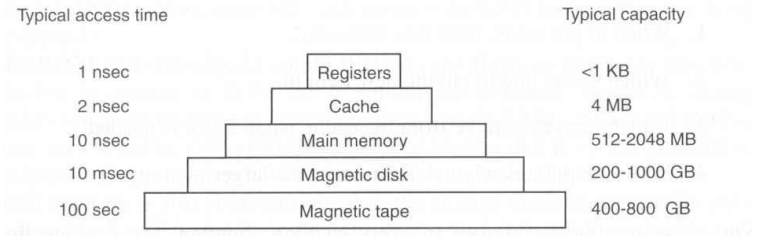


Figure 1-9. A typical memory hierarchy. The numbers are very rough approximations.

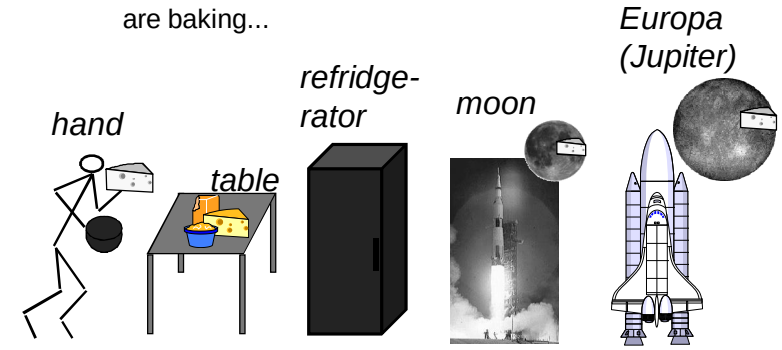
- Access time (*saantiaika*) (un?)dependent of the location
  - Registers, cache, main memory
  - Block buffering (*lohkopuskurointi)* (OS functionality!)
  - Magnetic and optical storage devices
- File servers (*tiedostopalvelimet*)
  - Network Attached Storage (NAS)
  - Storage Area Network (SAN)

Tan08 Fig 1.9

Computer Organization II, Spring 2010, Tiina Niklander 19.1.2010 7

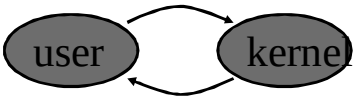
Teemu's cheese cake

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



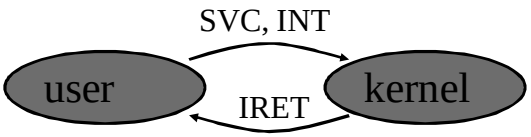
Computer Organization II, Spring 2010, Tiina Niklander 19.1.2010 8

CPU execution modes



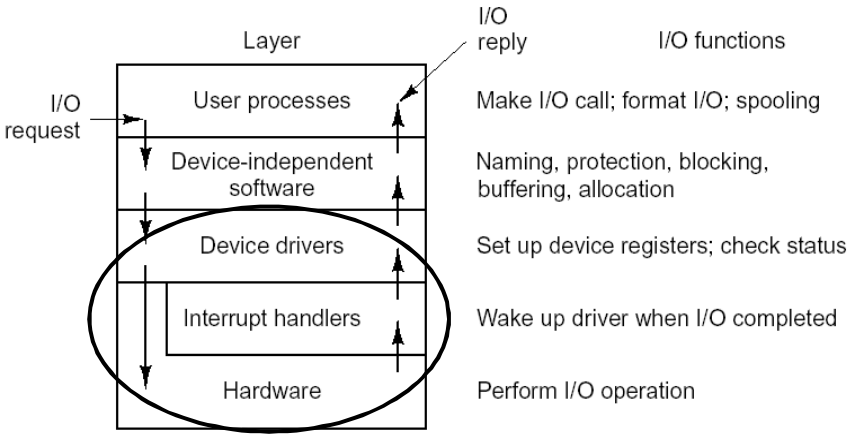
- Instruction privileges *privileged, kernel*
  - Privileged (*etuoikeutetut*) and normal *user, normal*
- Memory protection
  - Memory area marked for a user and controlled access
- User mode *(käyttäjätila*) *user mode, normal mode*
  - May use only normal instructions
  - Can refer only to its own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

Mode change

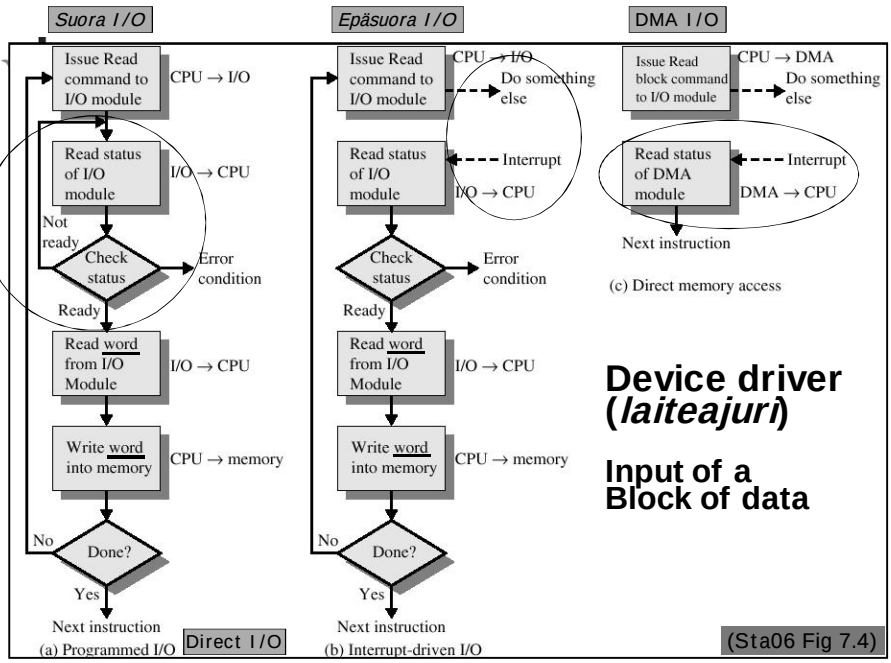


- User mode, normal mode → kernel mode, privileged mode
  - Interrupt or special SVC instructions (service request)
  - Interrupt handler checks the right for mode change
- Kernel mode → User mode
  - Privileged instuction, for example IRET (return from interrupt)
  - Returns the cotnrol and mode as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

Layers of the I/O system

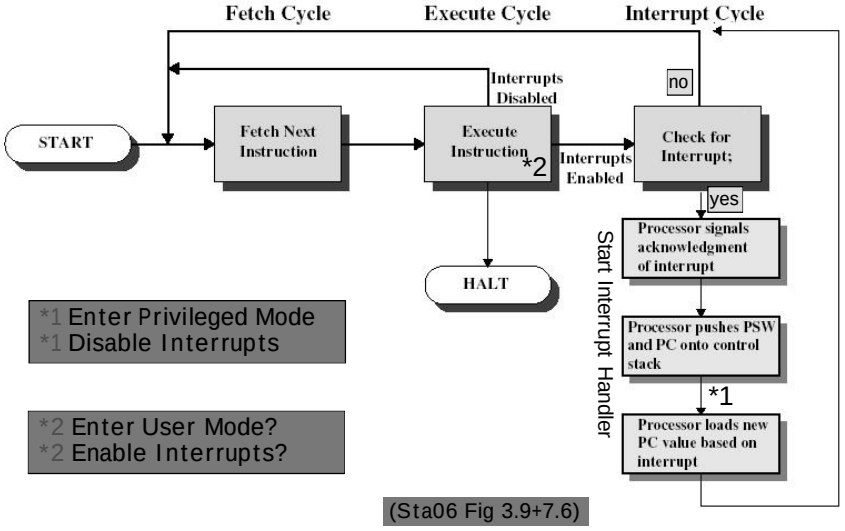


(Tan08, Modern Oper. Syst, Fig 5-17)



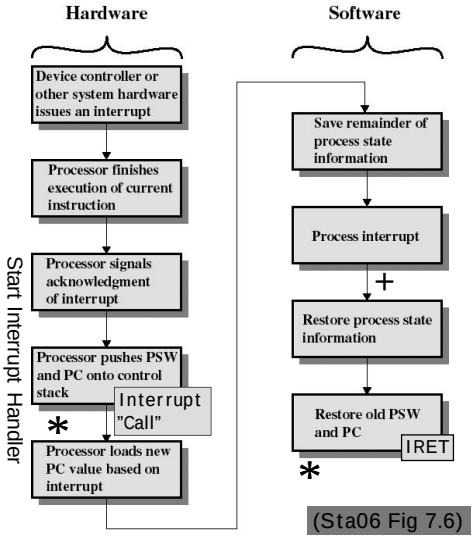
(Sta06 Fig 7.4)

CPU Instruction cycle (*käskysykli)*



(Sta06 Fig 3.9+7.6)

Interrupt handler (*keskeytyskäsittelijä*)



* Privileged mode vs. User mode

* Interrupt disabling vs. enabling

\+ Scheduling *(vuorotus)*

(Sta06 Fig 7.6)

Review Questions

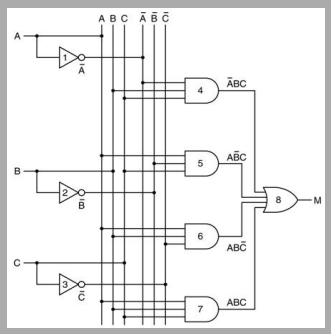
- Course book: at the end of each chapter
  - Answers in the chapter text
- From earlier courses: (see web)
  - Mainly in Finnish, created in project in earlier courses
- Create yourself:
  - List the most difficult and/or important issues
- Think at least about these:
  - Main parts of a computing system?
  - DMA: principles and functionalites?
  - Obligatory hardware and its features?
  - How to make CPU to execute normal user program? Operating system?

Self-study

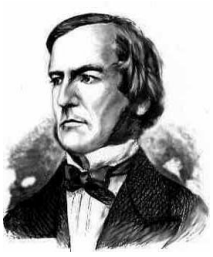
HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Digital logic

Stallings:
Online Chapter 20
Boolean Algebra
Combinational Circuits
Simplification
Sequential Circuits



Boolean Algebra



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

Topics:

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

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

$$D = A + \overline{B} \bullet C = A + ((\overline{B})C)$$

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

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

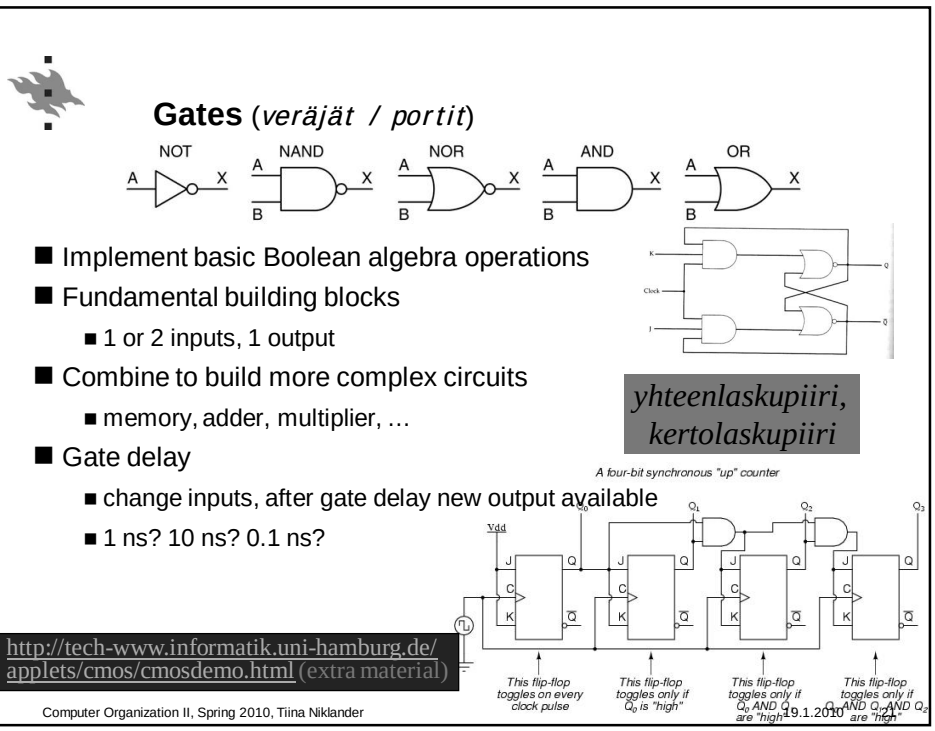
(Sta06 Table B.1)

Postulates and Identities

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

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

(Sta06 Table B.2)



Gates (*veräjät / portit*)

NOT NAND NOR AND OR

- Implement basic Boolean algebra operations
- Fundamental building blocks
  - 1 or 2 inputs, 1 output
- Combine to build more complex circuits
  - memory, adder, multiplier, …
- Gate delay
  - change inputs, after gate delay new output available
  - 1 ns? 10 ns? 0.1 ns?

yhteenlaskupiiri, kertolaskupiiri

A four-bit synchronous "up" counter

This flip-flop toggles on every clock pulse

This flip-flop toggles only if Q_0 is "high"

This flip-flop toggles only if Q_0 AND Q_1 are "high"

This flip-flop toggles only if Q_0 AND Q_1 AND Q_2 are "high"

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

Describing the Circuit

Boolean equations

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

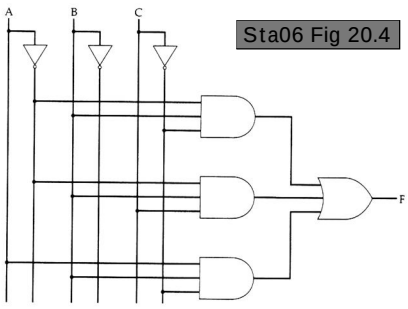
Truth table

<---------- 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

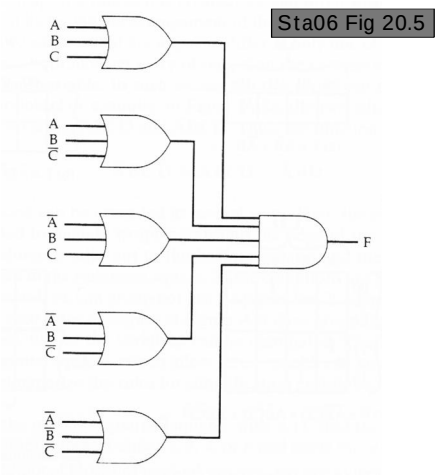
(Sta06 Table 20.3)

Graphical symbols
(next slide)

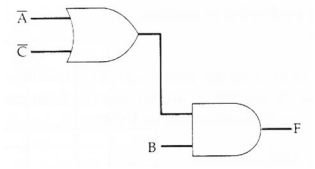
Sum-of-Products, Product-of-sums



Sta06 Fig 20.4

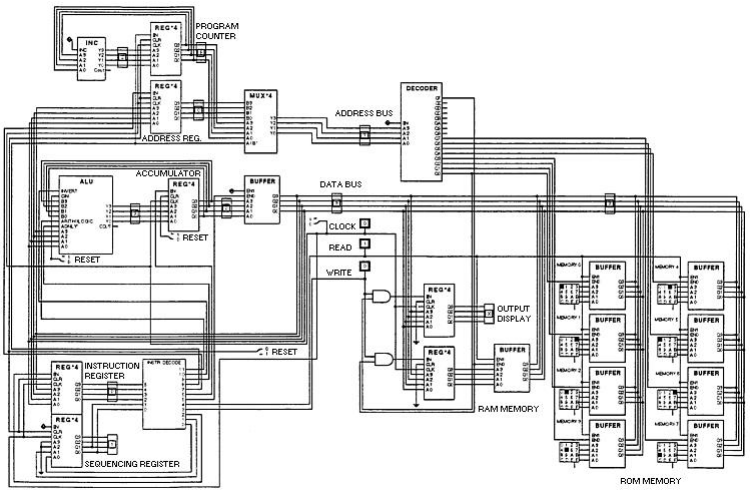


Sta06 Fig 20.5



Sta06 Fig 20.6

Simple processor



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