Lecture 1: Computer architecture

- Numeral systems
- History
- Von-Neumann- architecture
- Boolean Algebra and logic gates

Computer architecture – numeral systems

It all begun with the number systems, which are a set of number representations together with simple operations like addition, subtraction, multiplication and division:

- There are systems with different signs to represent the rating (e.g. Egyptian or Roman)
- There are systems with positional values of the numbers for the rating (e.g. our decimal system which came from India and the nearer east to us)

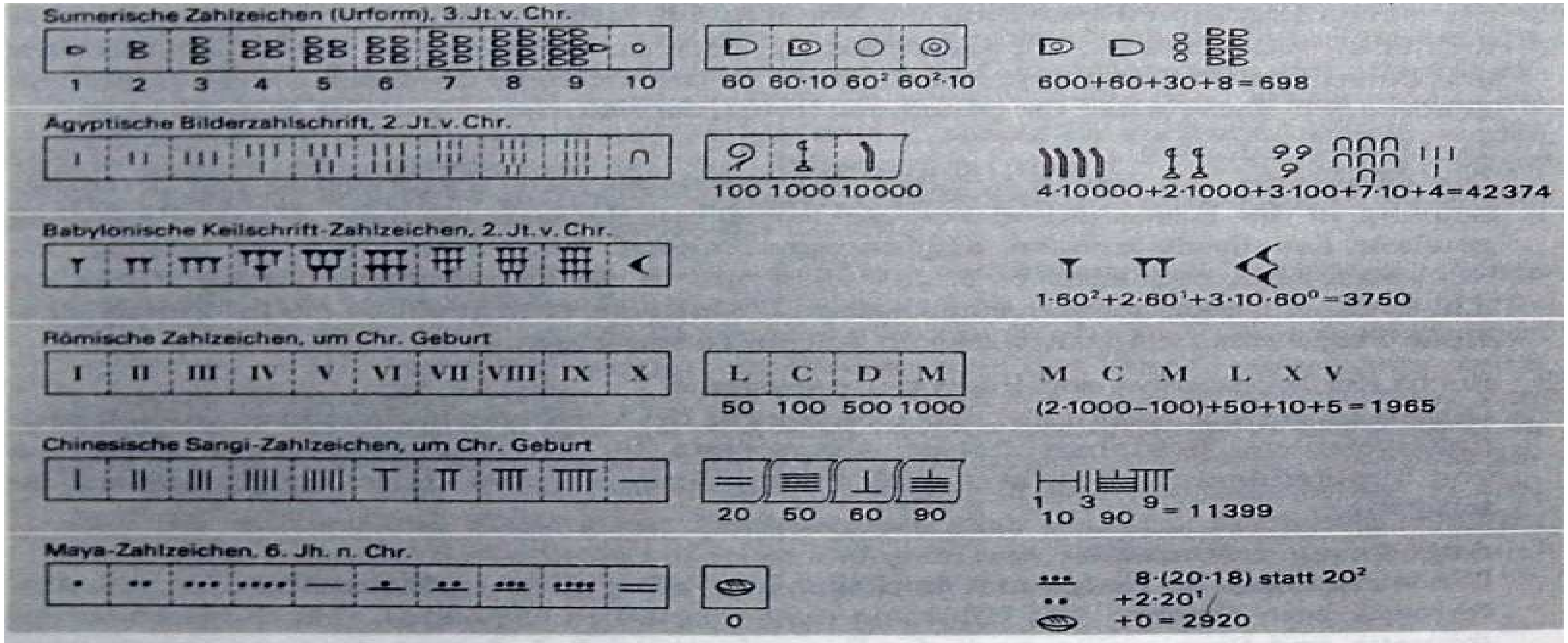


Bild 1.3 Entwicklung verschiedener Zahlensysteme [1.3]

See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991

Computer architecture – numeral system: decimal

- Based on the human experiences with the 10-finger-systems (Chinese and Hindu-Arabic, 3. century BC)
- Positional numeral system which represents each number just with 10 symbols, called digits (base-10 system)
- The simple numbers took a long journey until now (Arabic numerals/Indian numerals)
- It supports mathematical operations very easily
- Fractional parts can be represented by a decimal separator dot
- Positive and negative values can be represented

Bild 1.4 Entwicklung der Ziffern für das heutige Dezimalsystem [1.3]

See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991
See: http://en.wikipedia.org/wiki/Decimal, Download 2007-08-27

Computer architecture – numeral system: decimal

e.g.

2 8 7 7 7 3 9 9 0

$0 * 10^0$

$9 * 10^1$

$9 * 10^2$

$3 * 10^3$

$7 * 10^4$

$7 * 10^5$

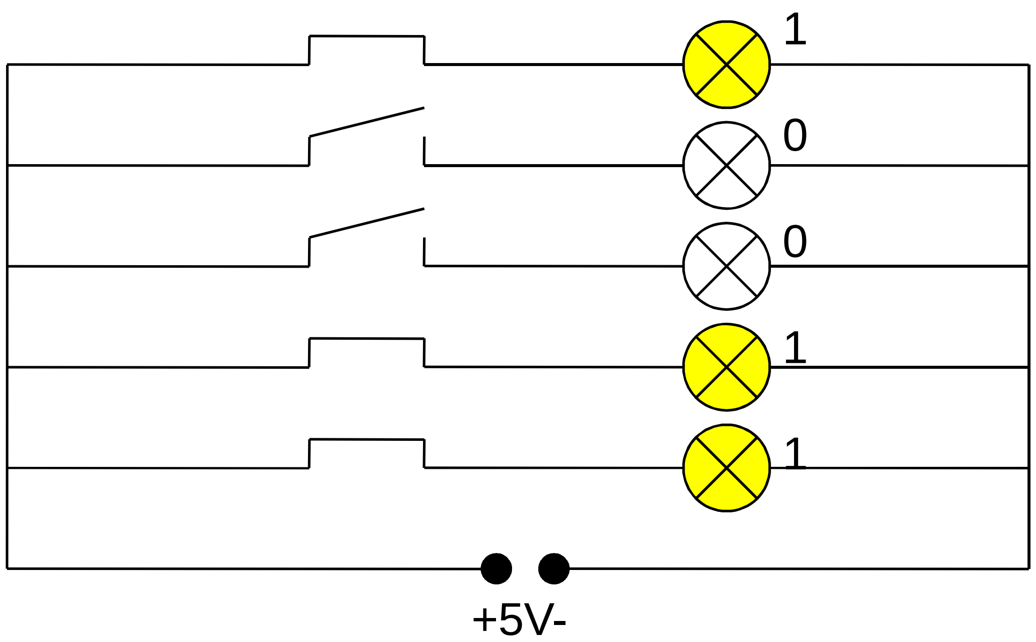
$7 * 10^6$

$8 * 10^7$

$2 * 10^8$

Computer architecture – numeral system: binary

- Origins at China and also developed by the mathematician Leibniz (17th century AD)
- Positional numeral system which represents each number just with 2 symbols, 0 and 1
- These values can be represented by voltage levels in electronic circuits
- For human use very inefficient but with electronic circuits it is possible to create very efficient arithmetic and logic units (ALUs) for the basic operations addition, subtraction, multiplication and division



See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991

Computer architecture – numeral system: binary

e.g.

1 0 1 1 0 0 1 0

$0 * 2^0$

$1 * 2^1$

$0 * 2^2$

$0 * 2^3$

$1 * 2^4$

$1 * 2^5$

$0 * 2^6$

$1 * 2^7$

Computer architecture – numeral system: binary -> decimal

e.g.

$$\mathbf{1\ 0\ 1\ 1\ 0\ 0\ 1\ 0}_{\mathbf{bin}}$$

$0 * 2^0 = 0 * 1 = 0_{dec}$

$1 * 2^1 = 1 * 2 = 2_{dec}$

$0 * 2^2 = 0 * 4 = 0_{dec}$

$0 * 2^3 = 0 * 8 = 0_{dec}$

$1 * 2^4 = 0 * 16 = 16_{dec}$

$1 * 2^5 = 0 * 32 = 32_{dec}$

$0 * 2^6 = 0 * 64 = 0_{dec}$

$1 * 2^7 = 0 * 128 = 128_{dec}$

$\sum = 178_{dec}$

Computer architecture – numeral system: binary -> decimal

e.g. 10110010_{bin} = 1 Byte

1 Bit = 1

$0 * 2^0 = 0 * 1 = 0_{dec}$

$1 * 2^1 = 1 * 2 = 2_{dec}$

$0 * 2^2 = 0 * 4 = 0_{dec}$

$0 * 2^3 = 0 * 8 = 0_{dec}$

$1 * 2^4 = 0 * 16 = 16_{dec}$

$1 * 2^5 = 0 * 32 = 32_{dec}$

$0 * 2^6 = 0 * 64 = 0_{dec}$

$1 * 2^7 = 0 * 128 = 128_{dec}$

$\sum = 178_{dec}$

Computer architecture – numeral system: decimal -> binary

e.g. **1 7 8** $_{\mathbf{dec}}$

178 / 2	=	89_{dec}	Rest 0_{bin}
89 / 2	=	44_{dec}	Rest 1_{bin}
44 / 2	=	22_{dec}	Rest 0_{bin}
22 / 2	=	11_{dec}	Rest 0_{bin}
11 / 2	=	5_{dec}	Rest 1_{bin}
5 / 2	=	2_{dec}	Rest 1_{bin}
2 / 2	=	1_{dec}	Rest 0_{bin}
1 / 2	=	0_{dec}	Rest 1_{bin}

Read direction

1 0 1 1 0 0 1 0 $_{\mathbf{bin}}$

Computer architecture

How many bytes are needed to represent the decimal number 1024?

Which decimal number is given by the binary representation 11010011 (show calculation)?

Lecture 1: Computer architecture

- Numeral systems
- History
- Von-Neumann- architecture
- Boolean Algebra and logic gates

Computer architecture – history

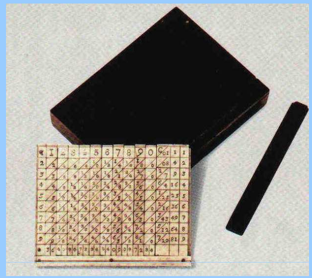
A short journey through the history of computer science:
calculation equipment

Calculation utilities

(2. cent. BC)

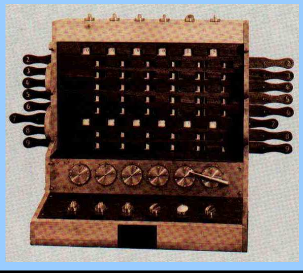
- **Calculation boards**
- **Abacus**
  (2. cent. BC)

(17th cent. AD)
John Napier of Merchistoun
(Napier's bones, Rabdology, 1623)
-> rods for mult./div.



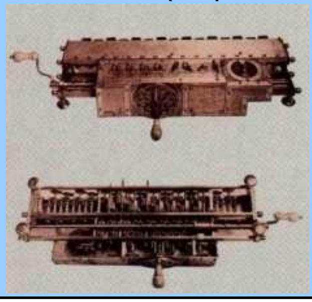
Mechanical calculation machines (18th cent. AD)

Wilhelm Schickard
(Speeding Clock, 1623)
-> slide rules for mult./div.
-> 6 digit add./sub.
-> carryover-gear
-> overflow detection



Blaise Pascal
(1641)
-> 8 digit add./sub.
-> similar to Schickards

Gottfried Wilhelm Freiherr von Leibnitz
(Stepped Reckoner, 1673)
-> add./sub./mult./div.
-> Staffelwalze (step reckoner)



Mathäus Hahn
(1674)
-> industrial produced calculation machines

See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991
See: http://privat.swol.de/SvenBandel/index.html, Download 01.09.2007
See: http://en.wikipedia.org/wiki/Image:Boulier1.JPG, 01.09.2007

Computer architecture – history

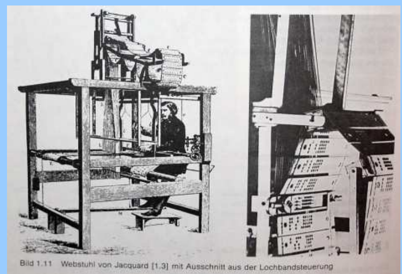
A short journey through the history of computer science:
calculation equipment

Programmable, mechanical calculation machines (19th/early 20th cent. AD)

Falcon
(1728)

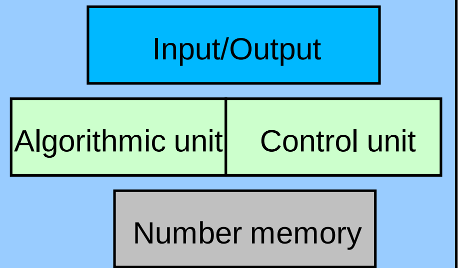
Joseph-Marie Jacquard
(1805)

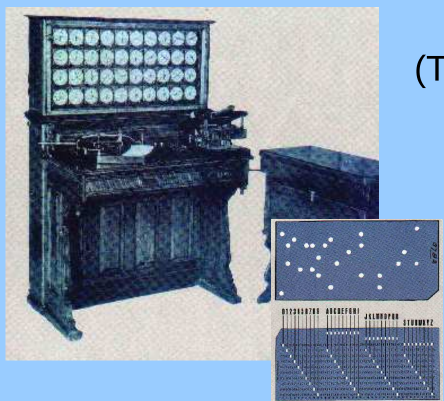
-> punchcard looms for work steps (program)



Charles Babage
(Difference Engine, 1822
Idea of a calculation machined)

-> ideas how they are used in moden computers





Herrmann Hollerith
(Tabulating machine, 1886)

-> punch cards with personal data (information storage)
-> counting clocks
-> sorter machines
-> punch card writer
=> US-population census 1890

Konrad Zuse
(Z1, 1934)

-> calculator similar to Babbage‘s
-> binary floating point numbers
-> logic operations

See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991
See: http://privat.swol.de/SvenBandel/index.html, Download 01.09.2007
See: http://en.wikipedia.org/wiki/Image:Boulier1.JPG, 01.09.2007

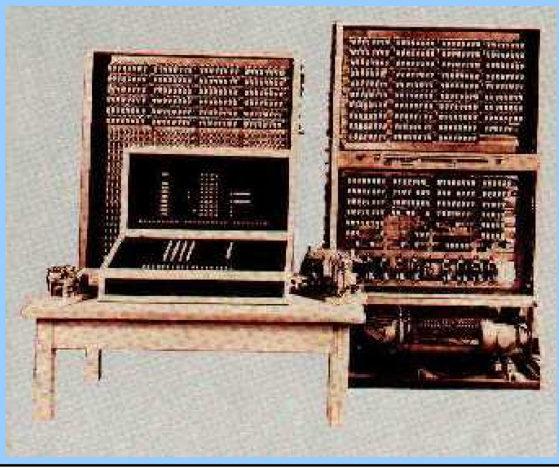
Computer architecture – history

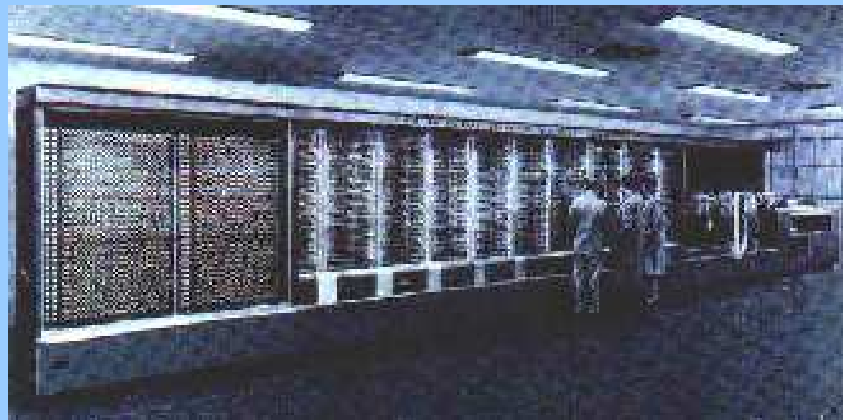
A short journey through the history of computer science:
calculation equipment

Modern electrical calculation utilities (middle 20th cent. BC)

Konrad Zuse
(Z3, 1941)

-> relaisbased algorithmic and logic unit (binary)
-> program on punch tapes
-> instructions consist of address- and operationpart
-> structure of modern computers





Howard H. Aiken
(Mark 1, 1944)

-> electromechanical
-> decimal system based
-> instructions consist of address-, control unit- and order-part

See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991
See: http://privat.swol.de/SvenBandel/index.html, Download 01.09.2007; See: http://en.wikipedia.org/wiki/Eniac, Download 02.09.2007

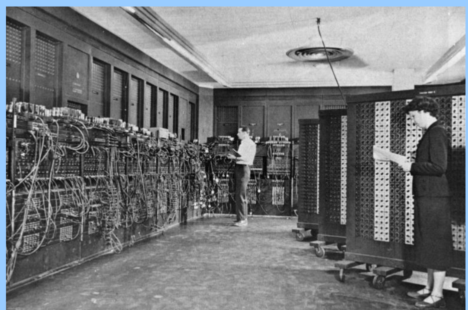
Computer architecture – history

A short journey through the history of computer science:
calculation equipment

Modern electronic (digital) calculation utilities (Computer) (late 20th cent. BC)

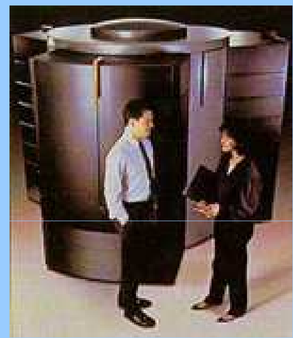
John P. Eckert
John W. Mauchly
(Electronic Numerical Integrator and Computer (ENIAC), 1946)

-> valvebased algorithmic and logic unit
-> digital computer (binary)



1953 - 1958

-> valves, core memory, magnetic memory, mainframe



1958 - 1966

-> transistors, operating systems, prog. languages, computer science

1966 - 1974

-> integrated circuits, structured prog., realtime processing

1974- 1982

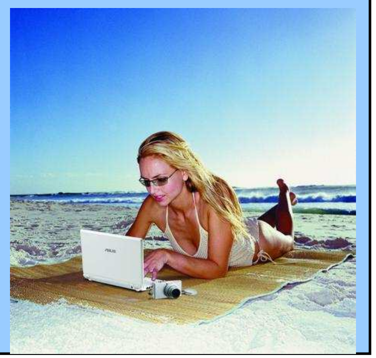
-> micro chips, networking

1982- 1990

-> high integrated circuits, parallel computing, home computer

1990 - now

-> personal computer, internet, middleware, object oriented and higher languages



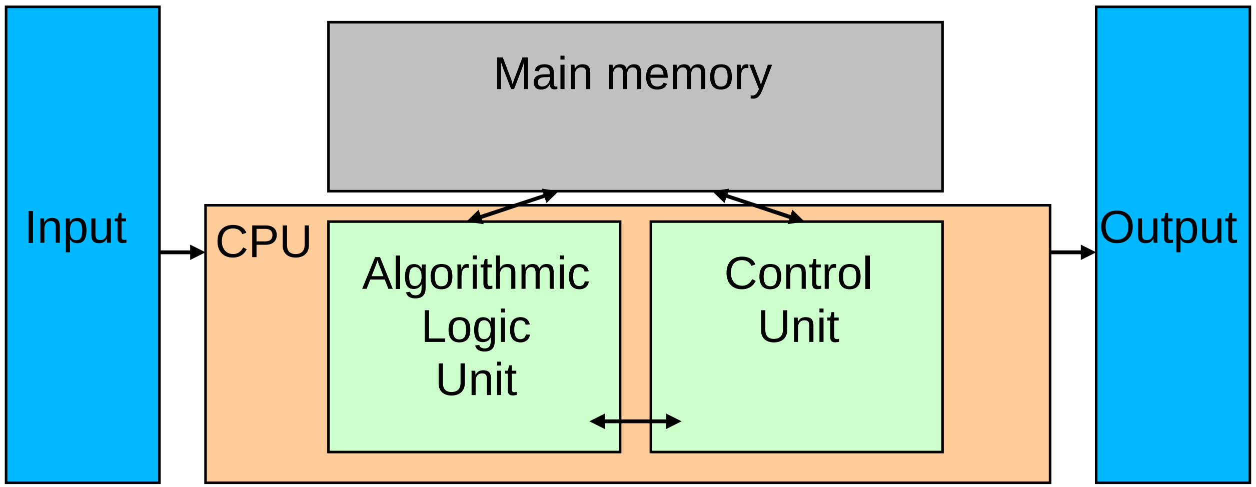
See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991; See: http://privat.swol.de/SvenBandel/index.html, Download 01.09.2007;
See: http://en.wikipedia.org/wiki/Eniac, Download 02.09.2007; See: http://renebrossard.com/NetworkingStudy.htm, Download 04.10.2009;
See: http://www.luxist.de/luxist/2008/01/16/it-gadgets-asus-eeepc-4g-ein-weltweit-einmaliges-pc-konzept/, Download 04.10.2009

Lecture 1: Computer architecture

- Numeral systems
- History
  - Von-Neumann- architecture
  - Boolean Algebra and logic gates

Computer architecture – John von Neumann

The heart of a computer:
the central processing unit (CPU) or processor

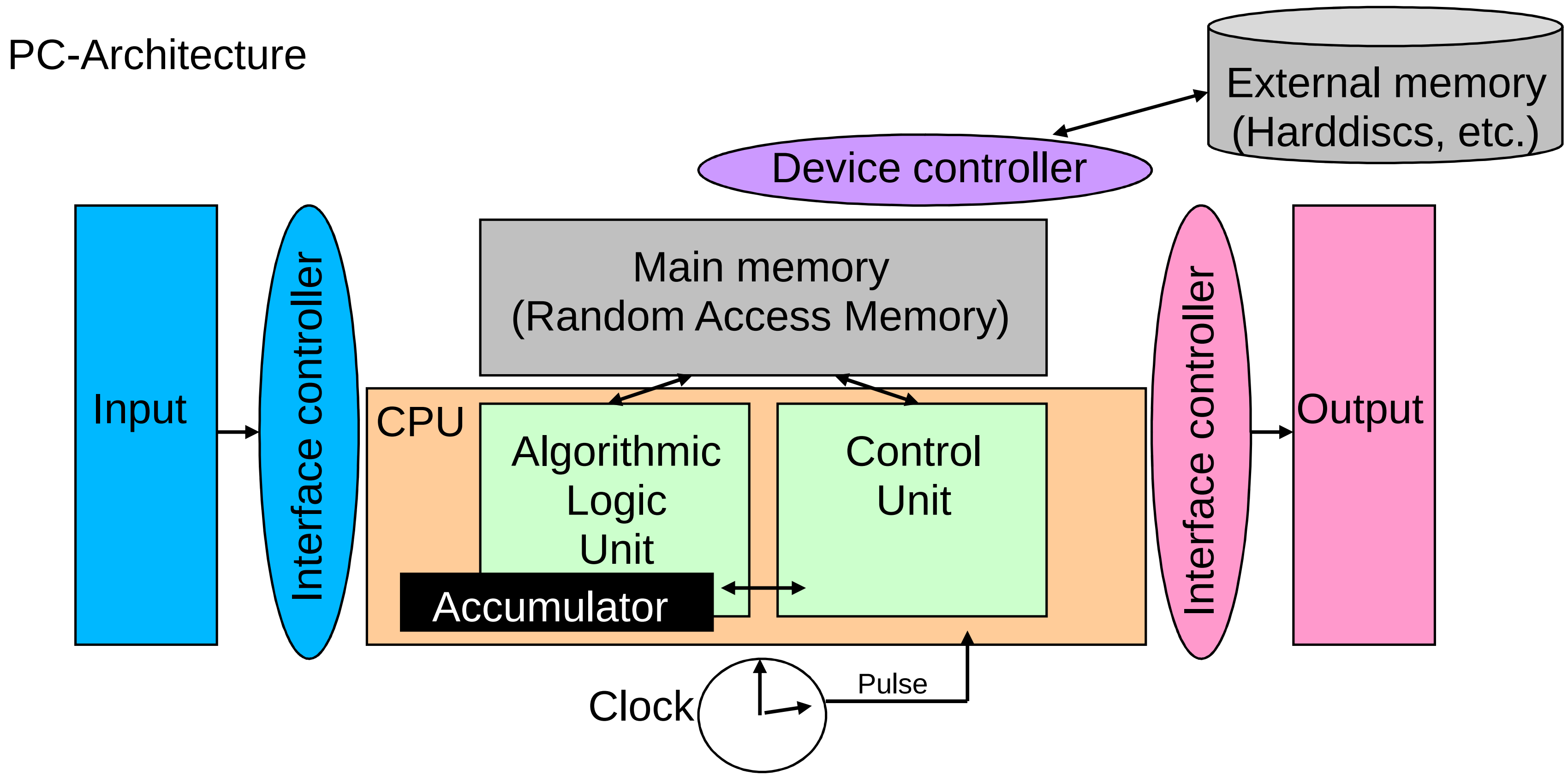


John von Neumann –Architecture (1945)

See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991

Computer architecture – Personal Computer

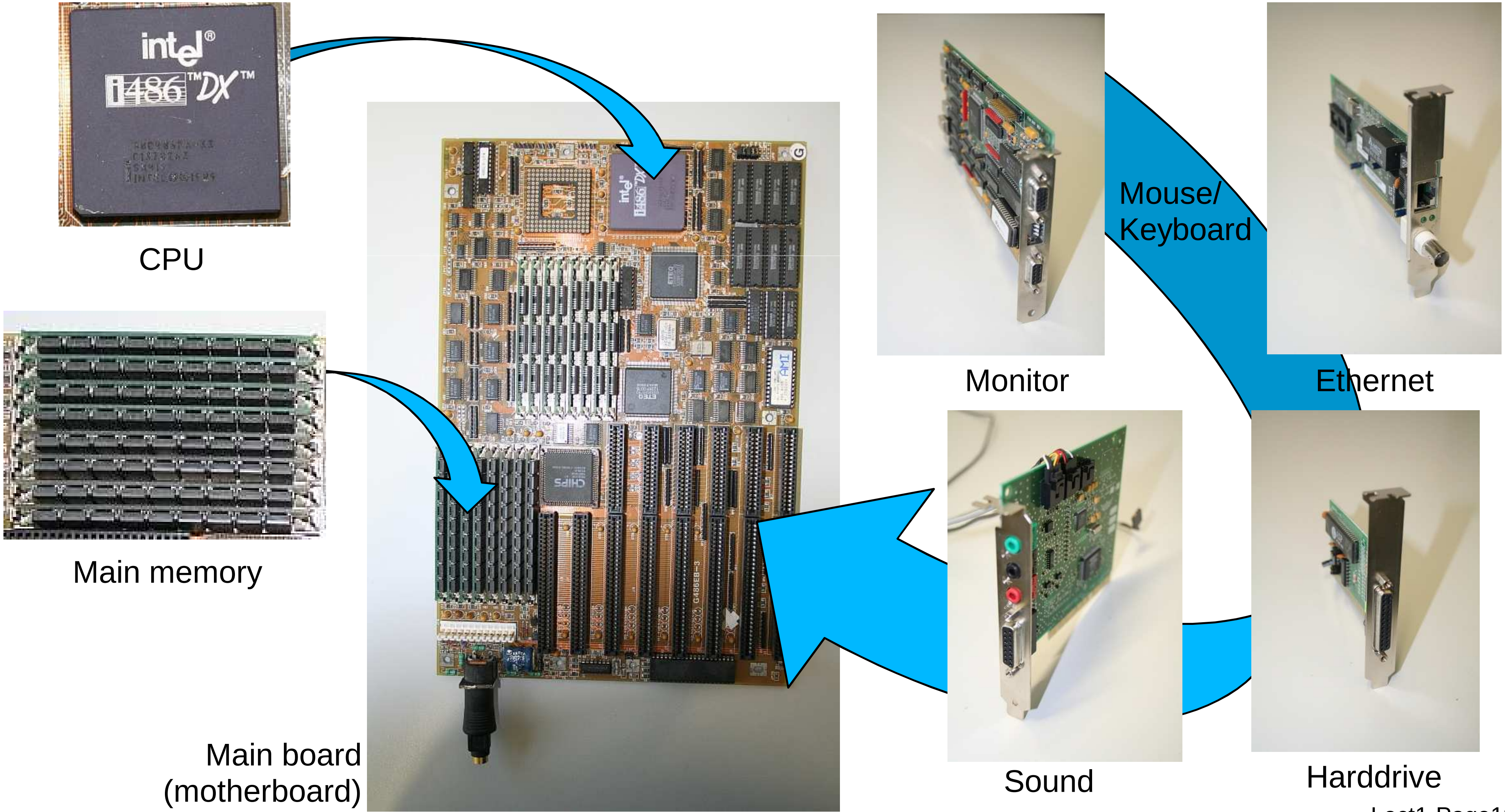
Scheme of a computer



See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991

Computer architecture – Personal Computer

PC-architecture, e.g. old main board



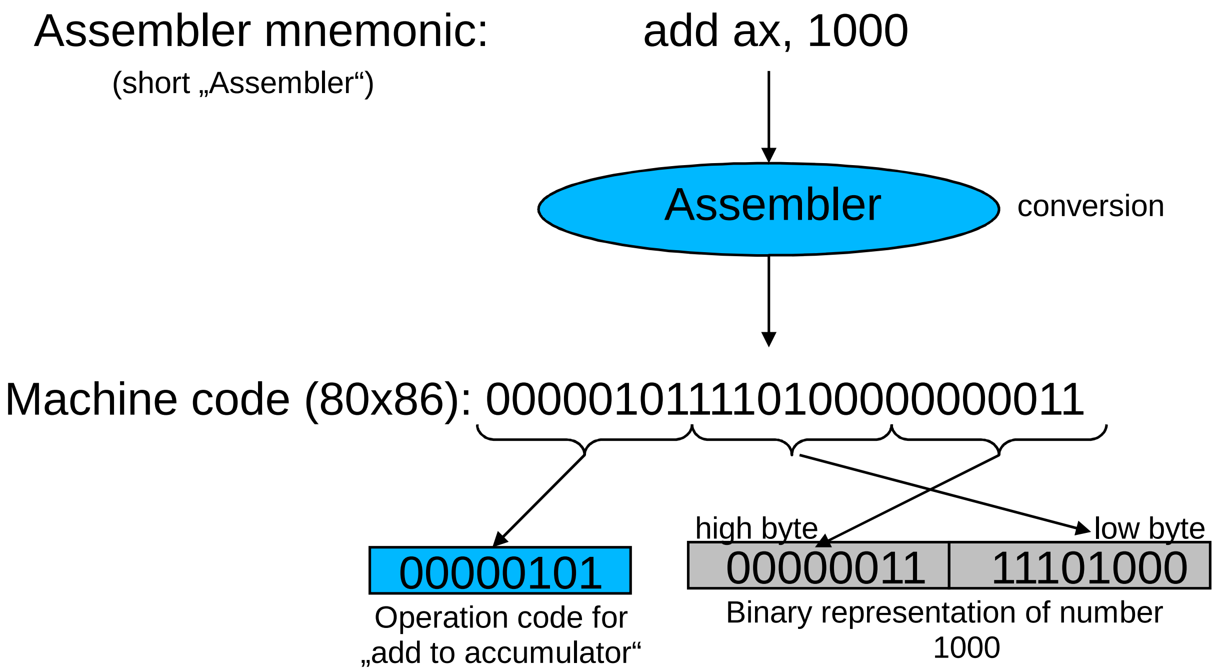
Computer architecture – Personal Computer

CPU operations work flow

FETCH:	Fetch the binary, numerical representation of an instruction from memory
DECODE:	Break the instruction into ist parts, which are used at the different areas of the CPU
EXECUTE:	Performe the operation at the different areas of the CPU
WRITEBACK:	Write the result back to memory

See: http://en.wikipedia.org/wiki/Central_processing_unit, Download 30.08.2007

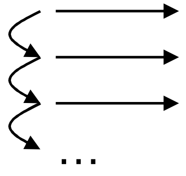
Computer architecture – Machine instructions



See: http://ivs.cs.uni-magdeburg.de/bs/lehre/sose99/bs1/seminare/assembler.shtml, Download 30.08.2007

Computer architecture – Machine instructions sequence

Instruction counter




```
segment code

 start:
 mov ax, data
 mov ds, ax

 mov dx, hello
 mov ah, 09h
 int 21h

 mov al, 0
 mov ah, 4Ch
 int 21h

 segment data
 hello: db 'Hello World!', 13, 10, '$'
```

→ Prints „Hello World!“ to output (monitor)

See: http://de.wikibooks.org/wiki/Assembler_(80x86_Prozessor)-Programmierung:_Das_erste_Assemblerprogramm, Download 02.09.2007

Lecture 1: Computer architecture

- Numeral systems
- History
- Von-Neumann- architecture
  - Boolean Algebra and logic gates

Computer architecture – Boolean algebra

A **Boolean algebra** is a set *A*, supplied with two elements 0 (called zero) and 1 (called one) and at least one binary operation (AND or OR) and an unary operation (NOT), such that, for all elements *a*, *b* and *c* of set *A*, the following axioms hold:

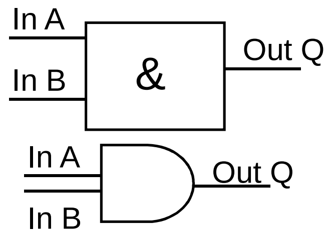
$a \vee (b \vee c) = (a \vee b) \vee c$	$a \wedge (b \wedge c) = (a \wedge b) \wedge c$	associativity
$a \vee b = b \vee a$	$a \wedge b = b \wedge a$	commutativity
$a \vee (a \wedge b) = a$	$a \wedge (a \vee b) = a$	absorption
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	distributivity
$a \vee \neg a = 1$	$a \wedge \neg a = 0$	complements
$a \vee a = a$	$a \wedge a = a$	idempotence
$a \vee 0 = a$ $a \vee 1 = 1$	$a \wedge 1 = a$ $a \wedge 0 = 0$	boundedness
$\neg 0 = 1$	$\neg 1 = 0$	0 and 1 are complements
$\neg(a \vee b) = \neg a \wedge \neg b$	$\neg(a \wedge b) = \neg a \vee \neg b$	De Morgan's laws
$\neg\neg a = a$		involution

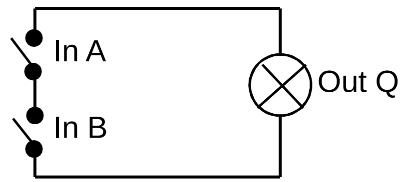
SEE: http://en.wikipedia.org/wiki/Boolean_algebra_%28structure%29, Download 01.09.2007

Computer architecture – Boolean algebra and logic gates

AND $\wedge$, $\circ$

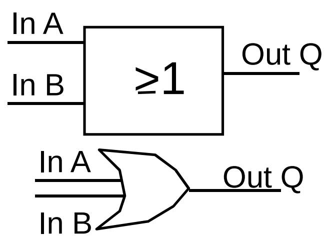
In A	In B	Out Q
0	0	0
0	1	0
1	0	0
1	1	1

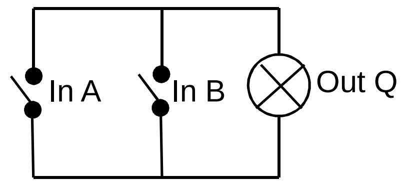




OR $\vee$, +

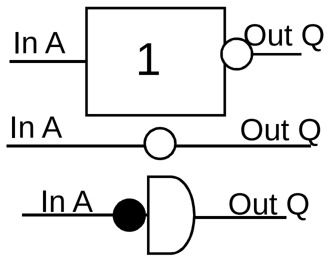
In A	In B	Out Q
0	0	0
0	1	1
1	0	1
1	1	1

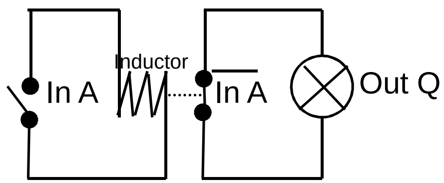




NOT $\overline{}$, $\neg$

In A	Out Q
0	1
1	0





Computer architecture – Design logic gate combinations

Complex gate structures are possible => ALU

e.g.

In A	In B	Out Q	
0	0	0	
0	1	1	$\overline{A}$ AND B
1	0	1	A AND $\overline{B}$
1	1	0	

$$Q = (\overline{A} \text{ AND } B) \text{ OR } (A \text{ AND } \overline{B})$$

Disjunctive normal form

SEE: http://edascript.ims.uni-hannover.de/230b_Logikoptimierung/fla/Logikoptimierung.swf, Download 02.09.2007

Computer architecture – Design logic gate combinations

e.g.

In A	In B	Out Q	
0	0	1	$\overline{A}$ AND $\overline{B}$
0	1	1	$\overline{A}$ AND B
1	0	0	
1	1	1	A AND B

$$Q = (\overline{A} \text{ AND } \overline{B}) \text{ OR } (\overline{A} \text{ AND } B) \text{ OR } (A \text{ AND } B)$$

Disjunctive normal form

Simplify (reduce gates)

Mathematical formal solution

$Q = (\overline{A} \text{ AND } \overline{B}) \text{ OR } (\overline{A} \text{ AND } B) \text{ OR } (A \text{ AND } B)$

distributivity

$= (\overline{A} \text{ AND } \overline{B}) \text{ OR } (\overline{A} \text{ OR } A) \text{ AND } B$

complements

$= (\overline{A} \text{ AND } \overline{B}) \text{ OR } 1 \text{ AND } B$

boundedness

$= (\overline{A} \text{ AND } \overline{B}) \text{ OR } B$

distributivity

$= (B \text{ OR } \overline{A}) \text{ AND } (B \text{ OR } \overline{B})$

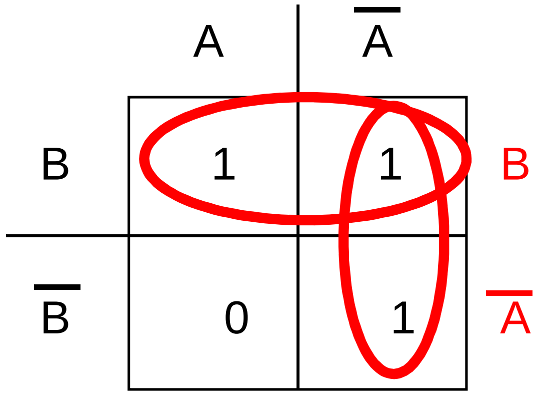
complements

$= (B \text{ OR } \overline{A}) \text{ AND } 1$

boundedness

$= (B \text{ OR } \overline{A})$

Graphical solution



Karnaugh-Veitch-diagram

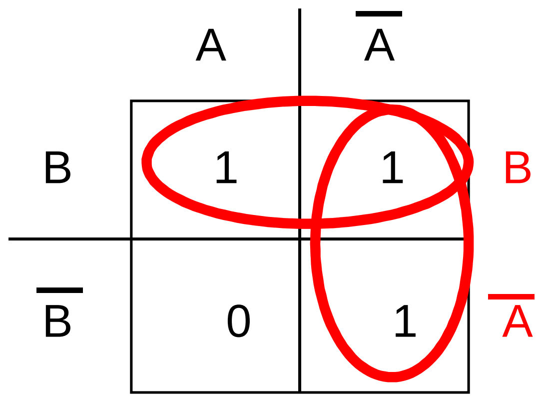
$$Q = (\overline{A} \text{ OR } B)$$

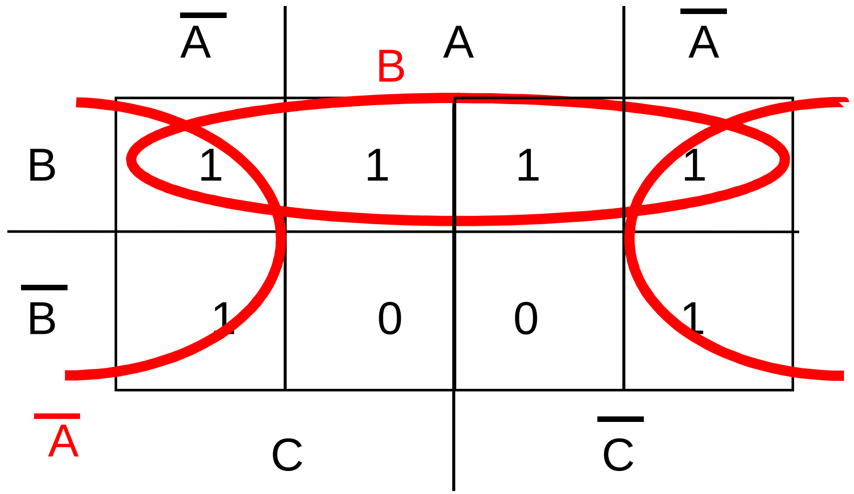
SEE: http://edascript.ims.uni-hannover.de/230b_Logikoptimierung/fla/Logikoptimierung.swf, Download 02.09.2007

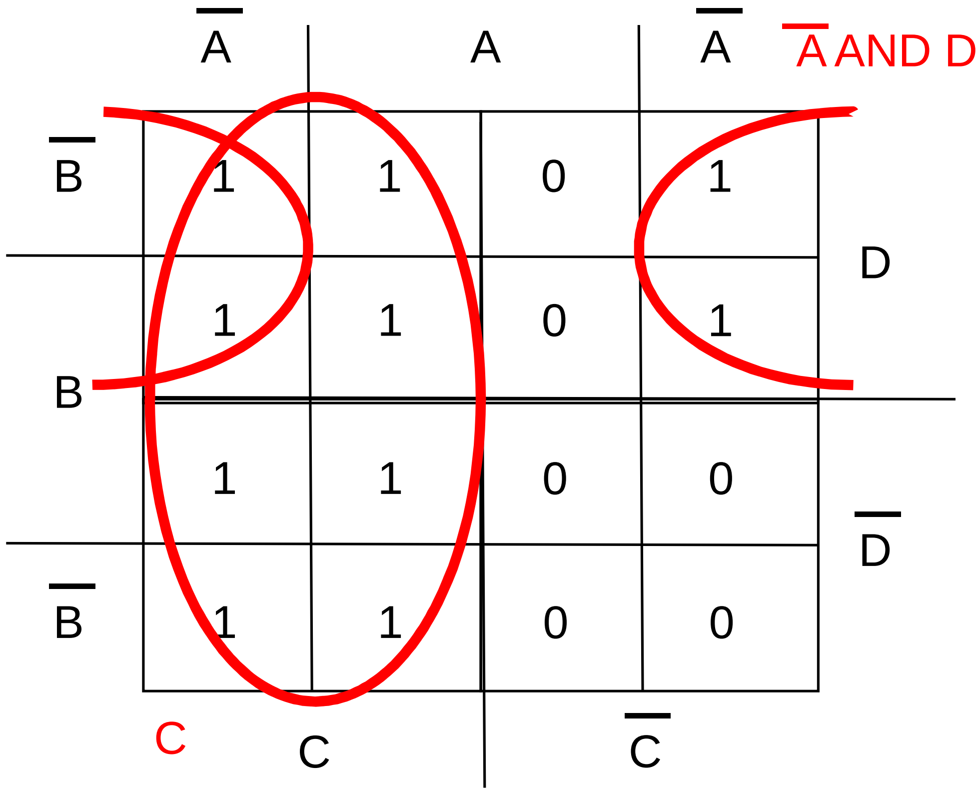
Computer architecture – Design logic gate combinations

Karnaugh-Veitch-diagram

- Graphical solution
- Reduce number of gates to optimal minimum
- Combine groups with 1, 2, 4 or 8 members
- Combine always max. number of elements
- Border crossing is alowed
- Multiple usage is allowed







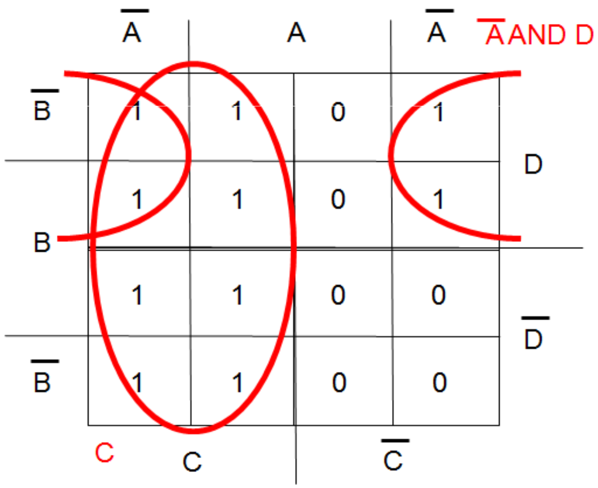
Computer architecture – Design logic gate combinations

Cross-checking:

Source table:

A	B	C	D	OUT
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Karnaugh-Veitch-diagram:



Check simplification:

A	B	C	D	NOT A	D	((NOT A) AND D)	C	OUT = C OR ((NOT A) AND D)
0	0	0	0	1	0	0	0	0
0	0	0	1	1	1	1	0	1
0	0	1	0	1	0	0	1	1
0	0	1	1	1	1	1	1	1
0	1	0	0	1	0	0	0	0
0	1	0	1	1	1	1	0	1
0	1	1	0	1	0	0	1	1
0	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0
1	0	1	0	0	0	0	1	1
1	0	1	1	0	1	0	1	1
1	1	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	0	0	0	0	1	1
1	1	1	1	0	1	0	1	1

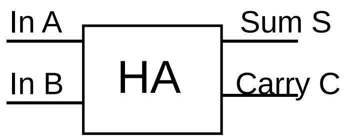
equal?

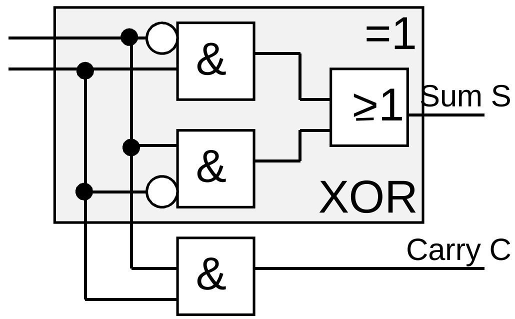
Computer architecture – Design logic gate combinations

Addition as basis of all mathematical operations: Half- and Full-Adder

HA

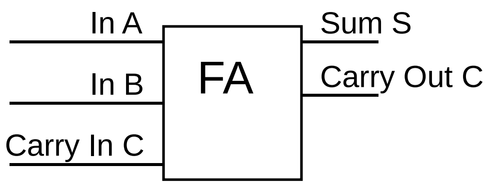
In A	In B	Sum S	Carry C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

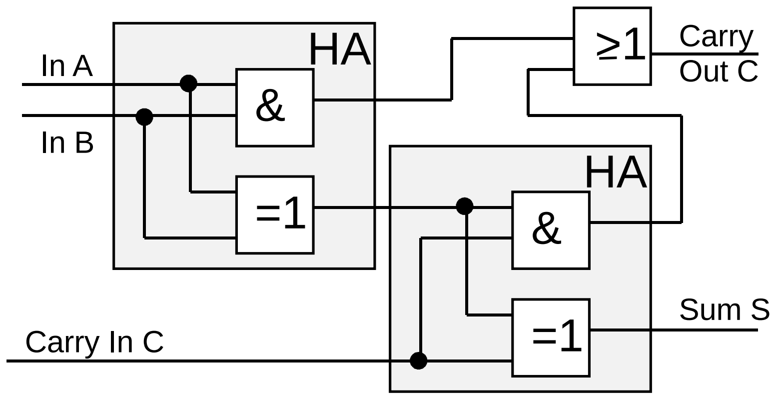




FA

Carry In C	In A	In B	Sum S	Carry Out C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



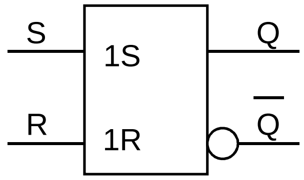


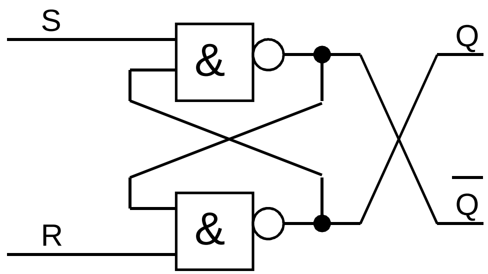
Computer architecture – Design logic gate combinations

Memory gates: Flip-flops

RS-Flipflop

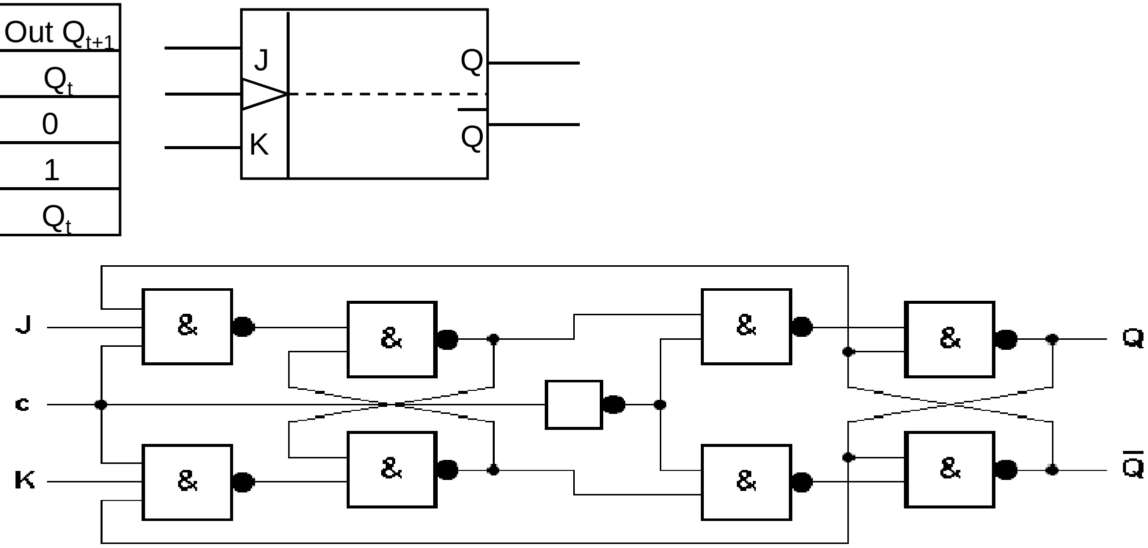
Set	Reset	Out Q_{t+1}
0	0	-
0	1	0
1	0	1
1	1	Q_t





Triggered JK-Master-Slave-Flipflop

J	K	Out Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	Q_t

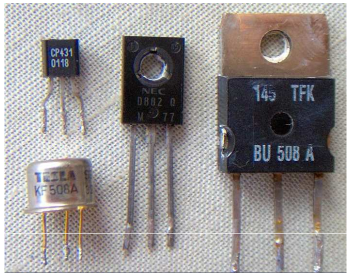


See: http://www.iris.uni-stuttgart.de/lehre/eggenberger/ksn/9_Flipflops/JK-FF.htm, Download 01.09.2007

Computer architecture – Modern logic gate design

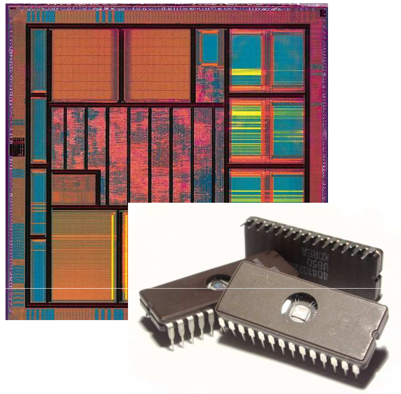
The modern logic gates: from valves to wafer circuits

„In electronics, a vacuum tube, electron tube (inside North America), thermionic valve, or just **valve** (elsewhere); is a device used to amplify, switch, otherwise modify, or create an electrical signal by controlling the movement of electrons in a low-pressure space.“



„A **transistor** is a semiconductor device, commonly used as an amplifier or an electrically controlled switch. The transistor is the fundamental building block of the circuitry (intecrated circuits) that governs the operation of computers and all other modern electronics.“

„In microelectronics, a wafer is a thin slice of semiconducting material, such as a silicon crystal, upon which microcircuits are constructed by doping (for example, diffusion or ion implantation), chemical edging, and deposition of various materials. Wafers are thus of key importance in the fabrication of semiconductor devices such as integrated circuits.“



„In electronics, an **integrated circuit** (also known as IC, microcircuit, microchip, silicon chip, or chip) is a miniaturized electronic circuits (consisting mainly of semiconductor device, as well as passive components) that has been manufactured in the surface of a thin substrate of semiconductor material.“

See: http://en.wikipedia.org/wiki/Vacuum_tube, Download 02.09.2007 ; See: http://en.wikipedia.org/wiki/Transistor, Download 02.09.2007
See: http://en.wikipedia.org/wiki/Integrated_circuits, Download 02.09.2007; See: http://en.wikipedia.org/wiki/Wafer_%28electronics%29, Download 02.09.2007

Computer architecture

Simplify the following truth table (define your used signs/ pictograms for AND, OR, NOT and use disjunctive normal form; advice for final test: task type could be similar with changed truth table values)

A	B	C	D	OUT
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Use a Karnaugh-Veitch-diagram

Use the mathematical method given by the axioms of Boolean algebra

Draw the logic gate map for simplified table (define the used graphical symbols in a short caption)

Thank you!