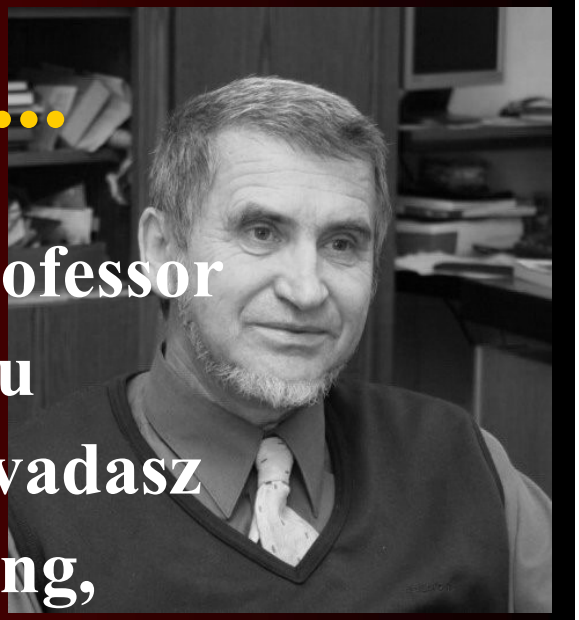
Computer architectures

Architectures

Általános
INFORMATIKAI
Tanszék

I introduce myself...



Dr. Dénes Vadász, associate professor

vadasz@iit.uni-miskolc.hu

http://www.iit.uni-miskolc.hu/vadasz

Informatics Institute building,
1st floor, room 111

University of Miskolc

Faculty of Mechanical Engineering and Informatics

IT and electrical engineering department group

Department of Information Technology

http://www.iit.uni-miskolc.hu

I introduce myself...

Dr. Szilveszter Kovács, professor

szkovacs @ iit.uni-miskolc.hu

http://www.iit.uni-miskolc.hu/~szkovacs

Phone: +36 46 565-111 / 21-07

Informatics Institute building,
1st floor, 10 7A . room

University of Miskolc

Faculty of Mechanical Engineering and Informatics

Institute of Informatics

Department of Information Technology

http://www.iit.uni-miskolc.hu

The purpose of the subject

- **Computer**
  - ***A tool*** **for performing** ***a computational task*** **described by a** ***programming language*** [1]
- **Architecture (structure, structure)** [2]
  - **If the goal is functional specification : specification,**
  - **If the goal is implementation: units and their connection.**
  - **Different levels of detail could happen in any orientation**
- **The purpose of the subject: to acquire general hardware knowledge, and to learn about user interfaces (command interpreter and graphical). And the development of the attitude.**

Computer story

- **Studying**
  https://en.wikipedia.org/wiki/Von_Neumann_architecture
- **The Neumann principle in brief:**
  - **The computer should be fully electronic with a separate control and execution unit.**
  - **Use a binary number system.**
  - **The data and the programs should be in the same internal storage, in the memory (*Stored-program principle*) .**
  - **The computer should be a universal Turing machine [1] (the principle of serial instruction execution should apply) (Turing 1937) The hypothetical machine had an infinite store that contained both instructions and data.**

[1] Operate without human intervention, automatically according to a program.

Important milestones

Year	Name	Maker	Comment
1834	Analytical Engine	Babbage	The first general purpose computer
1936	Z1	Zuse	First working, with relay technology
1943	COLOSSUS	British government	First electronic machine. Super secret
1944	Mark I	Aiken	First American general purpose (Harvard arch.)
1946	ENIAC	Ecker/Mauchley	The history of modern computers begins
1948		Neumann	The Neumann principle is born
1949	EDSAC	Wilkes	First stored-program principle
1952	IAS	Neumann	The basics of today's machine design
1961	1401	IBM	Business purpose, popular
1962	7094	IBM	For scientific calculations
1964	360	IBM	General purpose

Important milestones

Year	Name	Maker	Comment
1964	6600	CDC	First supercomputer for scientific purposes
1965	PDP-8	DEC	A widely used microcomputer
1970	PDP-11	DEC	A widely used microcomputer
1974	8080	Intel	General purpose machine on 8-bit chip
1974	CRAY-1	Cray	First vector supercomputer
1976	Z80	Zilog	For 8-bit embedded systems
1978	VAX	DEC	32-bit minicomputer, 1 MIPS
1981	IBM PC	IBM	The age of personal computers begins
1985	MIPS	MIPS	The RISC era begins
1987	SPARC	Sun	SPARC -based workstations
1990	RS6000	IBM	The first superscalar machine

Important milestones

Year	Name	Maker	Comment
1991	R4000	MIPS	The processor is already 64-bit
1992	Alpha	DEC	Excellent RISC processor
1994	IA-64	Intel	He announces his plans, they plan to introduce them in 1998-1999
2001	Itanium	Intel	True 64-bit
2003	PowerPC	Apple	Mac OS operating system
2003	AMD64	AMD	Compatible with 32-bit (emulates) Opteron, Athlon

Simulation: make believe, pretending; Emulation: imitating, following an example

The last decades in computing

The years	60's	70's	80's	90's
The paradigm	Batch processing	Time allocation	Desktop machines	Networks
Where?	In a computer center	In the terminal room	On a desk	Mobile
The data	Numerical data	Texts + numbers	… + drawings	Multimedia
Main goal	Calculations	Access	Display	Communi-cation

Paradigm : An example or sample suitable for proof and comparison. Following it can help you make good decisions.
A collection of accepted items and concepts, which can of course be a model.

The last decades in computing 2

The years	60's	70's	80's	90's
The paradigm	Batch	Time sharing	Desktop	Network
The interface	Punch card	Keyboard + CRT	See and click	Ask and tell
Remote connection	There isn't	Terminal lines	LAN	Internet
Owner	Institute computer center	Classes	Employees of departments	Everyone

Attitudes...

- Computer
  - *A tool* for performing *a computational task* described by a *programming language* [1]
- Architecture (structure, structure) [2]
  - If the goal is functional specification : specification,
  - If the goal is implementation: units and their connection.
  - Different levels of detail could happen in any orientation
- The purpose of the subject: to acquire general hardware knowledge, and to learn about user interfaces (command interpreter and graphical). And the development of the attitude.
- There are also different perspectives of user roles ...

General user view

- Mostly upper level
- Graphic, or command language user interface (icons, windows, tools, files, directories, etc., command, open, start, click drag, etc.)
- Services
  - Office
  - Communication
  - Information collection
  - Defense, management
  - Specific purpose applications

The programmer's view

- We are getting closer to the specifics...
- What the user sees, the programmer also sees
- Development interface (editors, make , compiler and task builder, debugger , etc.)
  - We teach this approach in other subjects

The administrator's view

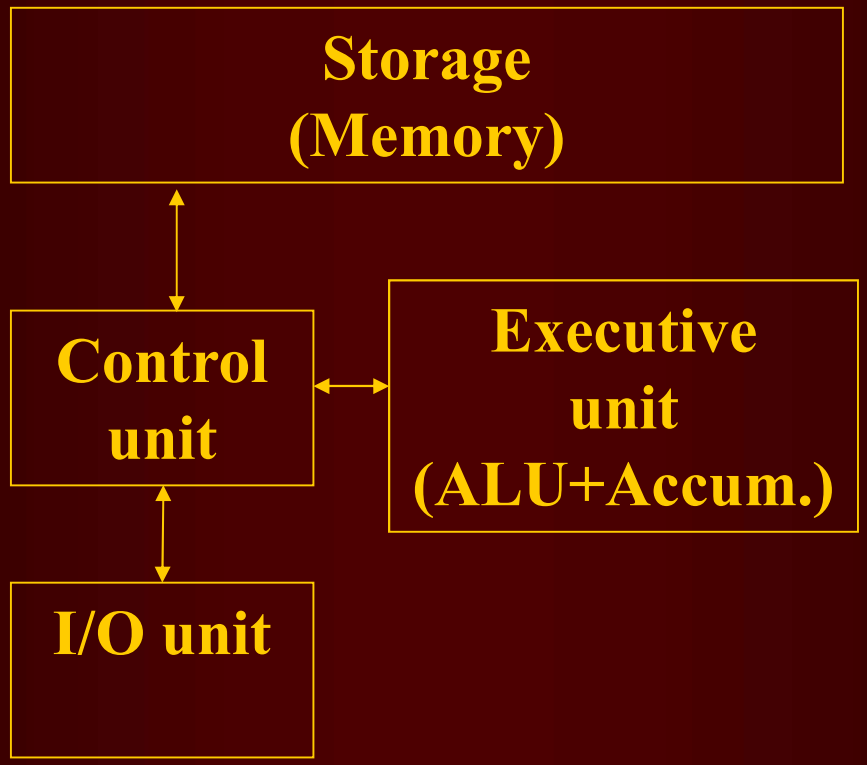
- Closer to the real machine
- Need to know the management of the operating system and their services.

Hardware knowledge

- **Should I know?**
  - **It is also easier for the simple user if he has certain knowledge**
  - **More for programmers...**
  - **A system administrator has many...**
  - **A lot for hardware, electrical engineering level**

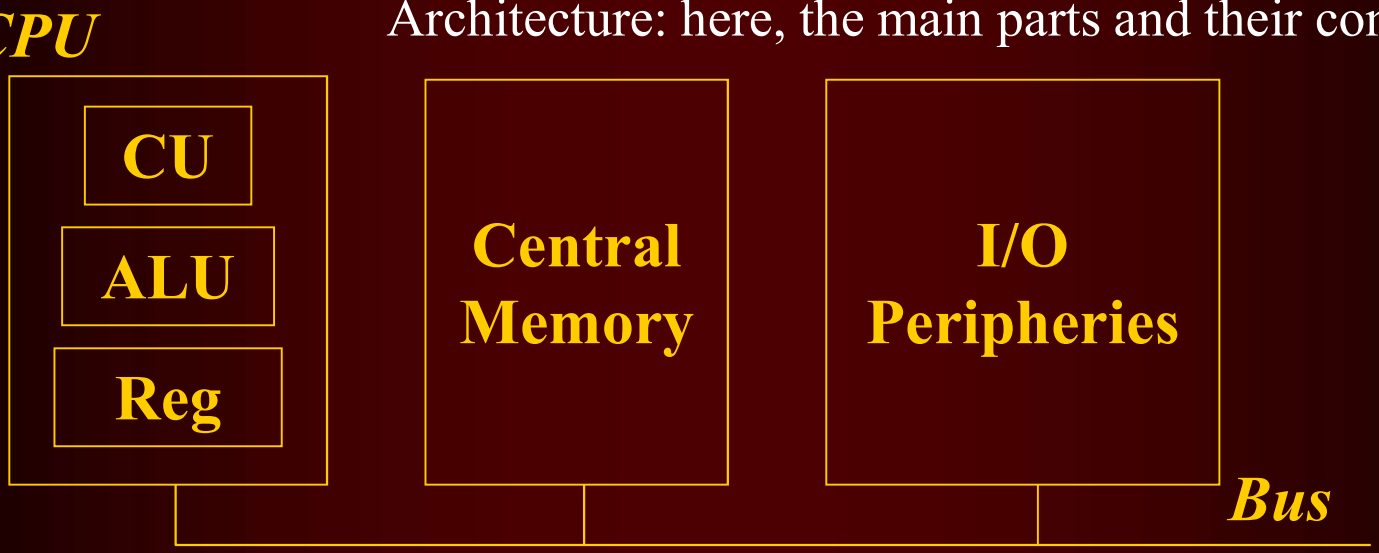
- **Note that a computer is defined by the hardware and software architecture together. We declare: this is an idea that can be derived from the Neumann principle!**

The original Neumann machine



The Neumann architecture (in today's terms)

Architecture: here, the main parts and their connections.



- **CU: Control unit**
- **ALU: Arithmetic and logical unit**
- **Regs: Registers**
- **Central Memory: Central memory, store ("Operative store")**
- **I/O Peripheries: I/O units, peripherals**
- **Bus: Rail, data transmission circuits**
- **Central Processing Unit: Central processing unit, processor**

Általános INFORMATIKAI Tanszék

Periphery : edge, outer part beyond the border.

Another functional model

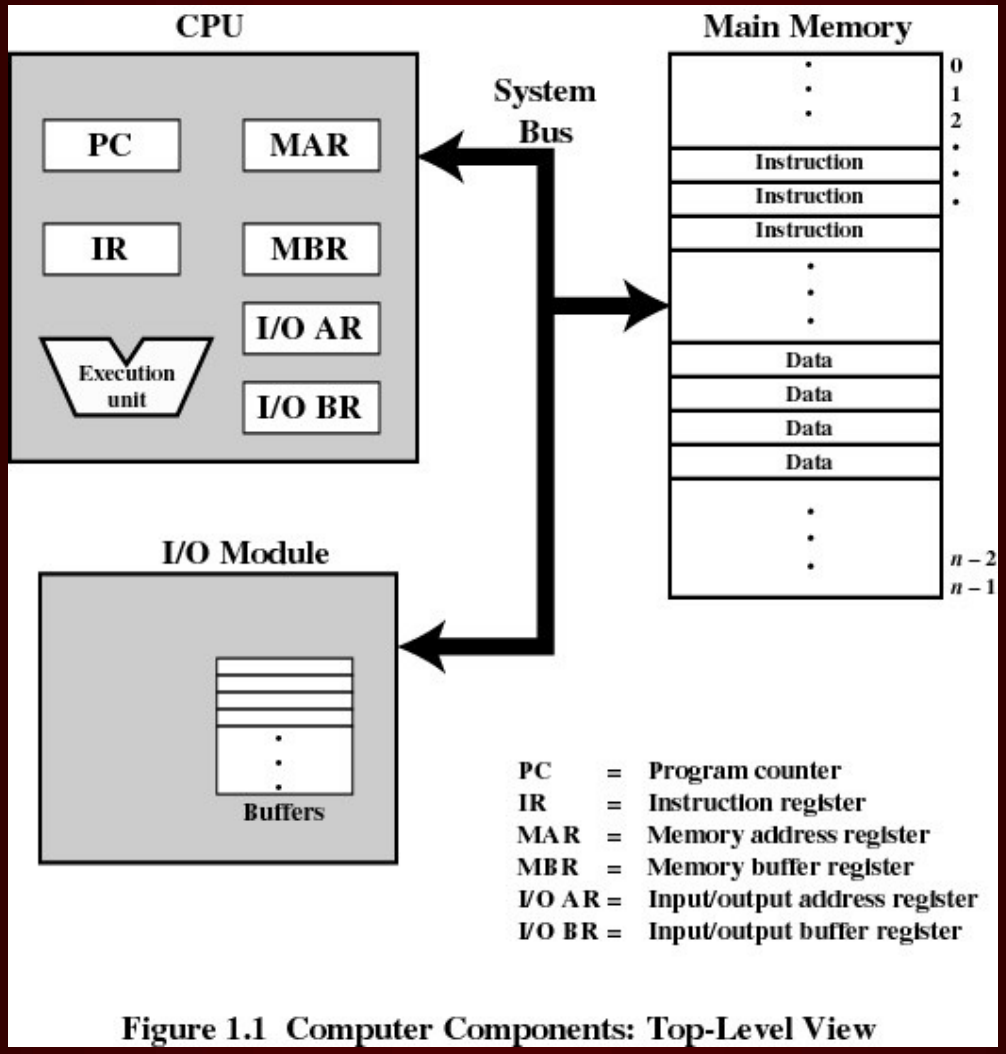


Figure 1.1 Computer Components: Top-Level View

The central storage and peripherals

- **The memory**
  - Data (bit, byte, word, block, record of fields, file, etc.) and
  - containing machine instructions,
  - set of addressable cells (compartments).
  - A state or state change due to some physical effect (magnetization, charge, voltage level, refraction of light, etc.)
- **Peripherals (I/O units)**
  - Devices connected with peripheral control circuits (controller, adapter) .
- **The CPU**
  - The processor is the unit that processes machine instructions.
    It consists of several functional elements (CU , ALU, Regs , etc.)

The operation of the Neumann machine

- The machine instructions (the code, the program) and the data are in the memory.
- The CPU fetches the next machine instruction from memory
- The CU analyzes the instruction. It interprets.
- If necessary, it fetches the operand of the instruction from memory
- The ALU executes the instruction
- The result of the execution is written back to the registers, or possibly to the corresponding compartment of the memory
- Continues with the next instruction ...

The state spaces

- Note the following abstractions
  - The "next instruction" concept is an *instruction stream*
  - A pointer can point to the next element on this stream.
    This pointer is the program counter register (PC: Program Counter , IP: Instruction Pointer)
  - The set of instructions in the instruction flow determines the *control state space*. One of these states is determined by which instruction is executed by the processor in the i-th step.
  - There is also a *data stream*: the row of memory compartments and registers that appear as operands in the successive instructions.
  - The elements of the data stream determines the *data state space*.
  - The execution of an instruction causes *a state change*.

Questions

- Only one IS and DS can be imagined? No way!
  - SIMD: can we imagine the same instruction stream on variable (different, multiple) data streams?
  - Where can it be? (e.g. vector processor)
- What is the control state space?
  - What is the control state (an element of the state space)?
- Data state space? An element of this?
- State change? What causes it?

Execution of the instruction stream

- **Running the program (the instruction stream is the code) is a chain of**

  state transitions.

- **The control state-transitions chain described by the successive values of the program counter register:**

  control flow

  (this is the concept of process or thread)

- **The Neumann machines has single *control flow* (Single Instructions Stream) *on a single data stream* (Single Data Stream): SISD**

Flinn's 1966 classification (SISD, SIMD, MIMD)

State dependence

- The state to which an instruction "tilts" depends on the previous state ...
- The Neumann machine is state-sensitive

- Can we imagine a state-independent machine? (Function)

The Neumann machine and imperative programming

- The imperative **languages** are munipulating the **course of control**
  - Do this with this, then this, etc.
  - FORTRAN , C, Pascal, Basic
- Therefore, the **imperative programming paradigm is well suited to the Neumann machine**

Imperative paradigm : fully specified steps on fully specified data

Error and event handling on the Neumann machine

- **Events** have a **handler instruction** flow
- When the event occurs, **the course of the control process changes**, "jump to the handler" (after "saving" the CPU state, the context)
- After the **event handler** (if possible), the control flow should return to the "normal" instruction flow, **the process should continue running** (after restoring the status and context back, of course).
- In summary: **error and event handling is done by manipulating the control flow**.

A machine with a different principle: Dataflow Machine

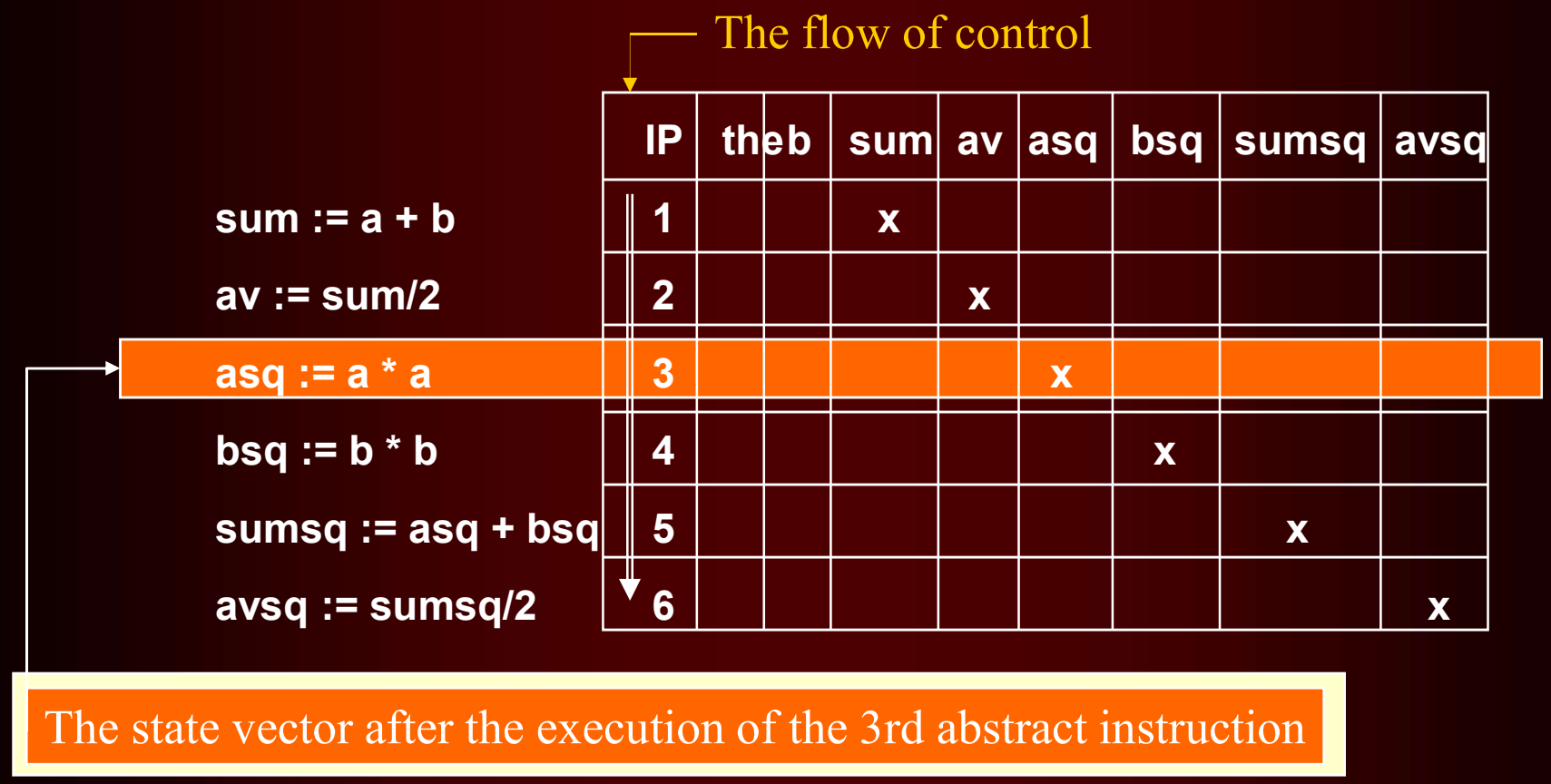
The **Dataflow Machine** (as a contrast) idea:

- **separate processors for each operation** (operation can be: arithmetic, logical, function call, etc.)
- Operations (processors) wait until the value of their operand is generated, and then produce their result.
- The processors (operations) are independent. They give their results at the earliest possible moment.
- **The order in which the operations are executed is determined by the data stream.**

programming **with declarative languages**

Example: given a and b, let us calculate their mean and their mean squares

A virtual Neumann machine (sequential execution)



Frequently used concepts

- Virtuality, virtual (apparent)
  - Something that doesn't really exist, but we can still use it as if it did
  - E.g. virtual drive, emulated terminal, virtual machine, etc.
- Transparency, transparent
  - Something that is there, but we don't see it, we don't notice it, we don't have to care about it, because it is transparent. (E.g. the aforementioned virtual disk drive is provided by a file server over the network, then the network is transparent, you don't have to worry about it.)
  - Clear, clean, undisguised ...

Example: given a and b, let us calculate their mean and their mean squares

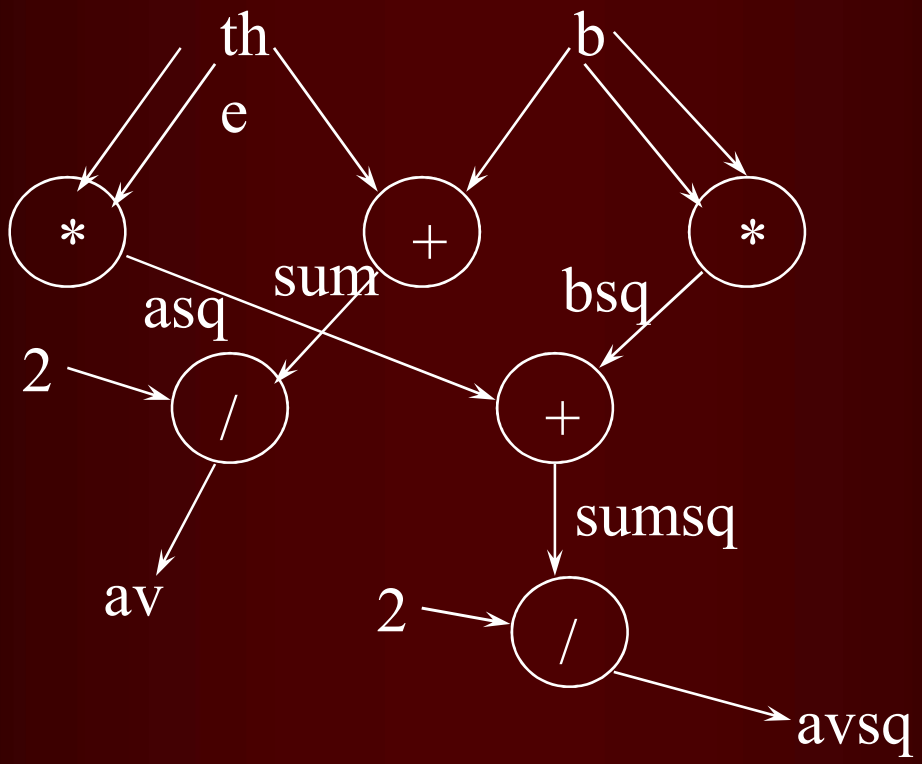
Neumann machine (sequential execution)

IP		800	804	808	80C	810	814
600	a						
604	b						
608	sum	x					
60C	av		x				
610	asq			x			
614	bsq				x		
618	sumsq					x	
61C	avsq						x
800	ADD a,b,sum						
804	DIV sum, 2, av						
808	MUL a, a, asq						
80C	MUL b, b, bsq						
810	ADD asq, bsq, sumsq						
814	DIV sumsq, 2, avsq						

Example: given a and b, let us calculate their mean and their mean squares

Dataflow Machine

```
sum := a + b
av := sum/2
asq := a * a
bsq := b * b
sumsq := asq + bsq
avsq := sumsq/2
```



Error handling: with explicit error values

The arrows: named or unnamed values.

The circles: the processors assigned to the operations.

Example: given a and b, let us calculate their mean and their mean squares

- Stream machine
  - There are 6 processors for 6 operations,
  - there are no "variables" (name, value, type, address),
  - there are named values (a,b,asq,bsq,sum etc.).
  - Named values cannot be redefined! In case of redefinition, the processors would not know which value to wait for!
  - Named values have a type and *an explicit error value*! P.s. a processor must definitely produce a value, which can be an “error”! It may also get “error” as input.

Computer - languages - computational model

- **Components of computational models**
  - basic elements of calculation
  - model of problem description
    - the nature of the description and
    - the method of the description;
  - model of execution
    - the implementation semantics
    - the control of execution

The Neumann model

- The basic elements : (tipified) data assigned to identifiable entities. (Variables, multiple value giving)
- Problem description
  - procedural/imperative (given step by step …)
- The model of the execution
  - the semantics: state-transition semantics
  - the control: direct control (… the course of the control…)

Semantics (Greek): semantics, the branch of linguistics dealing with the meaning and change of meaning of linguistic forms
Entity : Something that exists in substance. Element.

The Dataflow model

- The basic elements : (tipified) data assigned to identifiable entities . (One-time value delivery)
- Problem description
  - declarative (listing operations, e.g. by using functions ...)
- Model of model of the execution
  - the semantics : applicative
  - the control : dataflow controlled

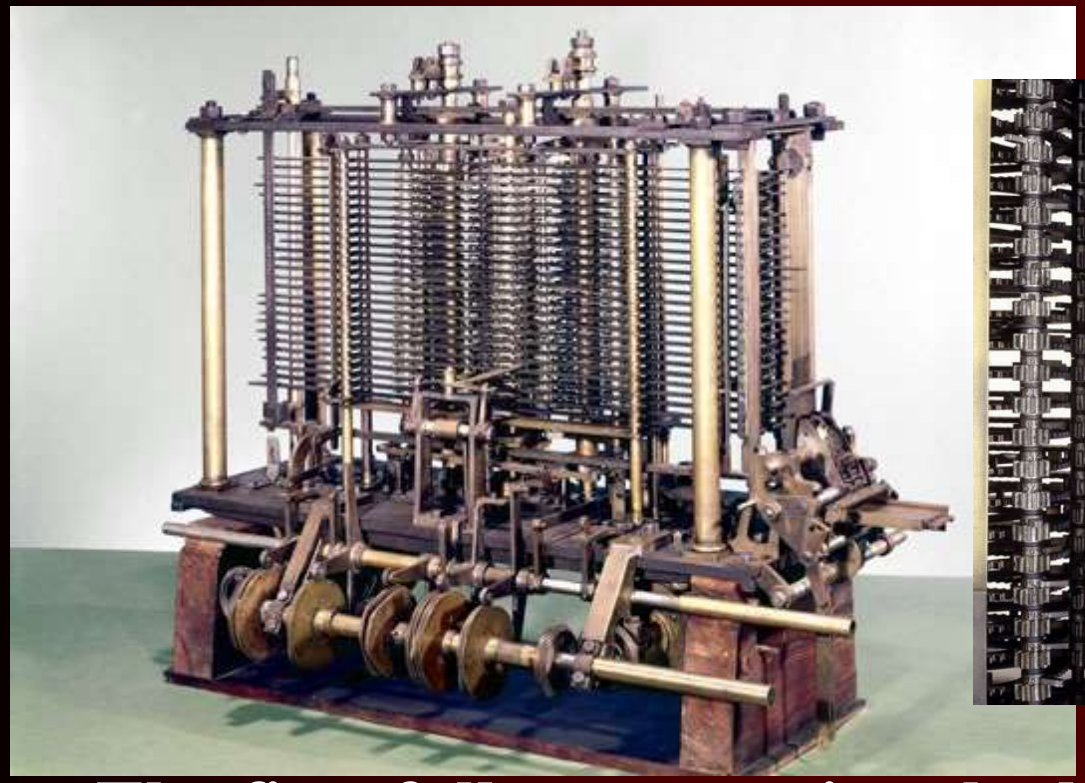
Declarative paradigm : A declarative program just enumerates.
The order of the "instructions" of such a program does not matter.
A logical programming language can also be declarative
(where the inference engine "travels" through the entire rule base).

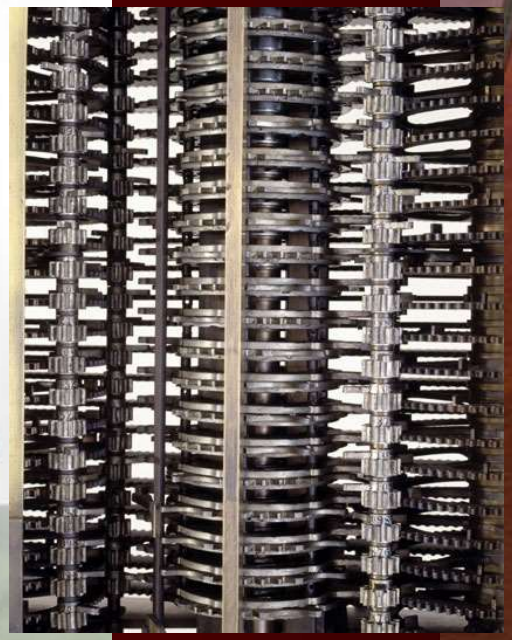
The Neumann principle machine

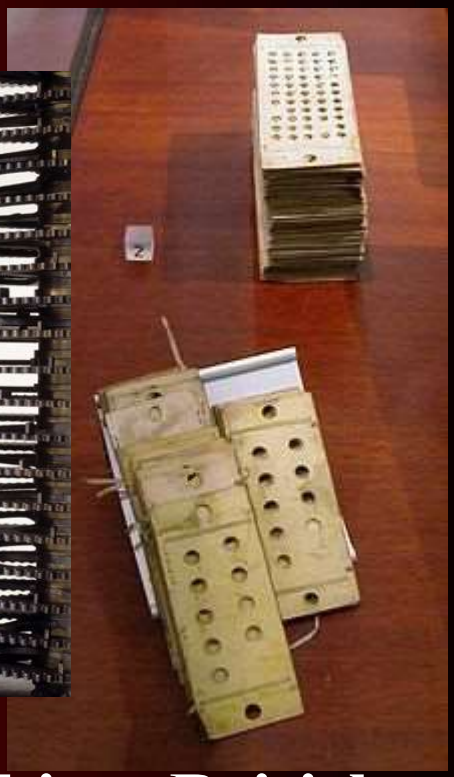
- Main parts of the machine, requirements:
  - ALU, control unit, memory, peripherals,
  - Number system 2, electronic. Before?
- Stored-program principle:
  - The store contains both the data and the program. Before?
  - Consequences: good and bad.
- Automatic operation (direct control):
  - according to program, states, state-transitions, control process, role of PC/IP.
- Babbage's Analytical Engine: Does it fit?

The Neumann principle machine

- Babbage's Analytical Engine: Does it fit?



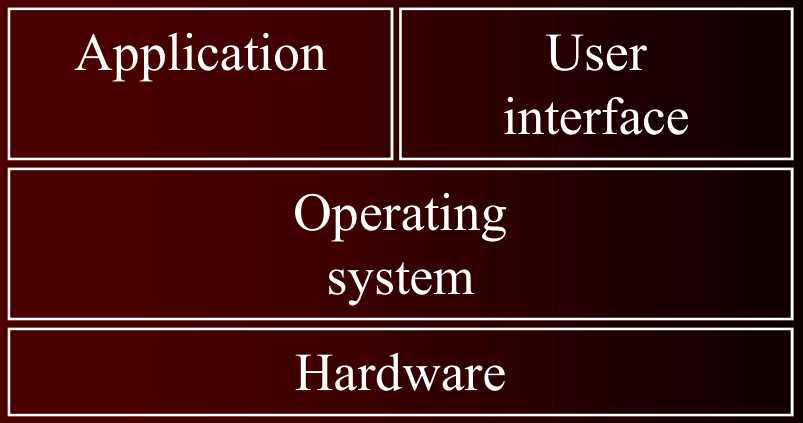




- The first fully automatic calculating machine. British computing pioneer Charles Babbage (1791-1871)
- Foreground: "operational cards", for instructions ; background: "variable cards", for inputting data .

Computer: hardware and software architecture

- The most common SW architecture
- Direct run, monitor, operating system
- Concept of OS
  - Extended machine
  - Resource manager



The layering
(layered architecture)
A layer hides the details of the layers below it.
It is enough to know only the interface of the layer below you

Operating system classification

- **By purpose : general, purpose**
- **According to HW "size" : PC, small, large, super**
- **By number of processors, processes, and users**
- **According to time sharing : sequential,**
  time sharing: cooperative, preemtive
- **According to memory management : real, virtual**
- **According to the implementation of the File system**

Summary

- Introduction
- A little history ...
- The Neumann principle machine and the
- Dataflow machine.
- Computational models

Computer architectures

Architectures

End

Általános
INFORMATIKAI
Tanszék