Architektury Komputerów

dr inż. Adam Mrozek
amrozek@agh.edu.pl
p. 414/B5

Program przedmiotu

- **Wstęp**
  - historia…
  - podstawowe typy architektur
  - struktura klasycznego systemu mikroprocesorowego
- **Elementy techniki cyfrowej**
  - algebra Boole'a, systemy liczbowe, kodowanie liczb
  - podstawowe elementy składowe bloków funkcjonalnych procesora i komputera
- **Arytmetyka komputerowa**
  - + i -, algorytmy * i /
  - rozwiązania sprzętowe; jednostka arytmetyczno-logiczna
  - floating point
- **Typy, budowa i działanie procesorów**
  - cykl rozkazowy procesora, miary wydajności
  - lista rozkazów; tryby adresowania; podstawy assemblera; typy danych itp/
  - procesory współczesne: **przetwarzanie potokowe, superskalarność; instruction-level parallelism**

Program przedmiotu

- **Typy oraz hierarchia pamięci komputera**
  - ROM (UV,flash), RAM – styczna i dynamiczna
  - rejestry procesora
  - pamięć podręczna (cache)
  - pamięć operacyjna
  - pamięć wirtualna
- **Urządzenia wejścia-wyjścia**
  - sposoby wprowadzania i wyprowadzania danych (klawiatura, układ graficzny, układy interfejsów)
  - system przerwań
  - DMA (bezpośredni dostęp do pamięci)
- **Ochrona zasobów: I/O, przerwań, ochrona pamięci**
- **Budowa pliku wykonywalnego i bibliotek**
  - kompilacja i konsolidacja
  - biblioteki statyczne, dynamiczne, współdzielone

Na wykładzie

- omawiane zagadnienia będą ilustrowane przykładami pochodzącymi z różnych systemów mikroprocesorowych:
- RISC (MIPS, ARM,) i CISC (x86)
- główny nacisk położony będzie na sprzęt

Na laboratorium

- architektura x86-64 (AMD64/x64)
- system Linux (64 bitowy)
- kompilator GNU języka C(++) gcc (g++)
- GNU asembler (as), konsolidator LD (gcc)

Cel przedmiotu

- **Znajomość i zrozumienie zasady działania klasycznego i współczesnego systemu mikroprocesorowego**

- **Znajomość i zrozumienie podstaw programowania w języku niskiego poziomu (asembler)**

  - tworzenia pliku wykonywalnego („programu”)
  - sposobu wykonywania programu przez komputer (pętle, instrukcje warunkowe)
  - przechowywania i wymiany danych (np. z systemem operacyjnym, innym programem, biblioteką…)
  - bibliotek (statycznych, dynamicznych, współdzielonych)

Czyli… tego, co normalnie pozostaje „ukryte” przy programowaniu w języku wysokiego poziomu

Cel przedmiotu

- **Znajomość wpływu rozwiązań sprzętowych i programowych na wydajność komputera / obliczeń**

Nie da się tworzyć dobrych programów bez znajomości architektury sprzętu

Dobre programy:

- spełnianie wymagań funkcjonalnych
- niezawodność
- bezpieczeństwo
- wydajność

Znajomość sprzętu:

- jest istotna dla bezpieczeństwa i niezawodności programów
- jest konieczna do tworzenia wysokowydajnych programów

Tematyka kontynuowana będzie na przedmiotach dotyczących obliczeń wysokiej wydajności, programowania równoległego, systemów wbudowanych

Literatura

- **Patterson D.A., Hennessy J.L., Computer Organisation and Design, The Hardware/Software Interface, Fifth Edition, Morgan Kaufmann**
- Hennessy J.L., Patterson D.A., Computer Architecture, Morgan Kaufmann
- R. Blum, Professional Assembly Language, Wiley Publishing
- D. Kusswurm, Modern X86 Assembly language Programming, Apress

Podstawy systemów liczbowych, algebry Boole'a zawarte są w większości książek dotyczących techniki cyfrowej, języków programowania, np.:

- Pieńkos J., Turczyński J., Układy scalone TTL w systemach cyfrowych, WKŁ, 1980

Ponadto:

- materiały producentów elementów systemów komputerowych
- tysiące stron www (korzystać! – tylko z rozsądkiem…)

Regulamin

Wykład:

• kolokwium (ostatnie zajęcia w semestrze)

Laboratorium:

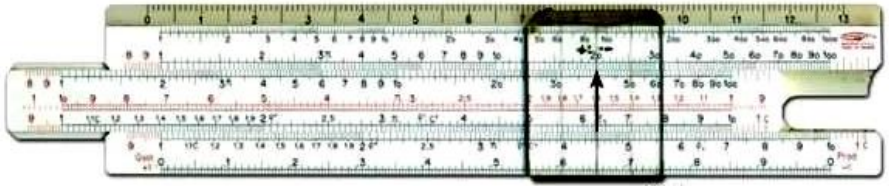
• regulamin na stronie przedmiotu: **galaxy.agh.edu.pl/~amrozek**

A brief history of computing

- **Abacus – Middle East ca. 3000-2000 BC**
- **300-200 BC mathematics, algorithms, numerical computations, logic (**Greeks, Egyptians, Babylonians, Indians, Chinese, and Persians…)
- **17th century**

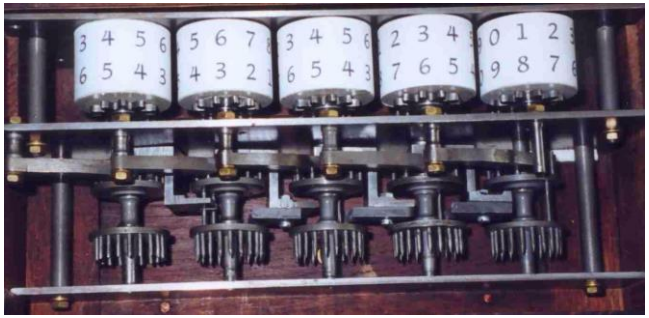
$$\log(a \cdot b) = \log(a) + \log(b)$$

- **1614 AD – logarithms (John Napier)**
- **Slide rule**



multiplication, division, rising to the power, logarithms…

- **Pascalino - Pascal (+,-) and Leibnitz(*,/)**
  mechanical arithmometer



zdjęcia „poglądowe” – z sieci…

A brief history of computing

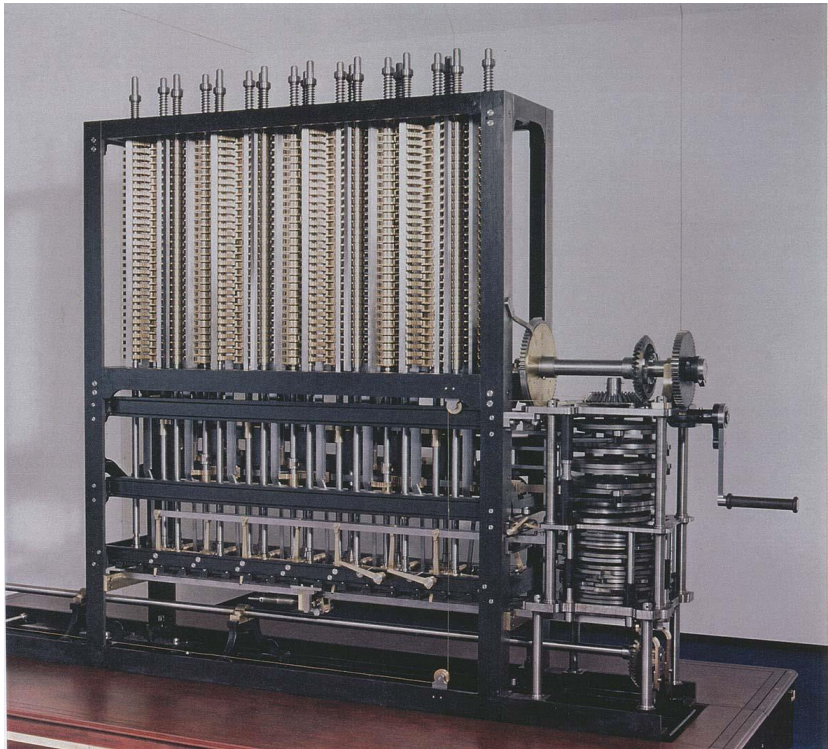
19th century
Charles Babbage's conceptual designs:

1822 AD Difference Engine

- Too complex!
- Working prototype was built after 150 years

1837 AD Analytical Engine

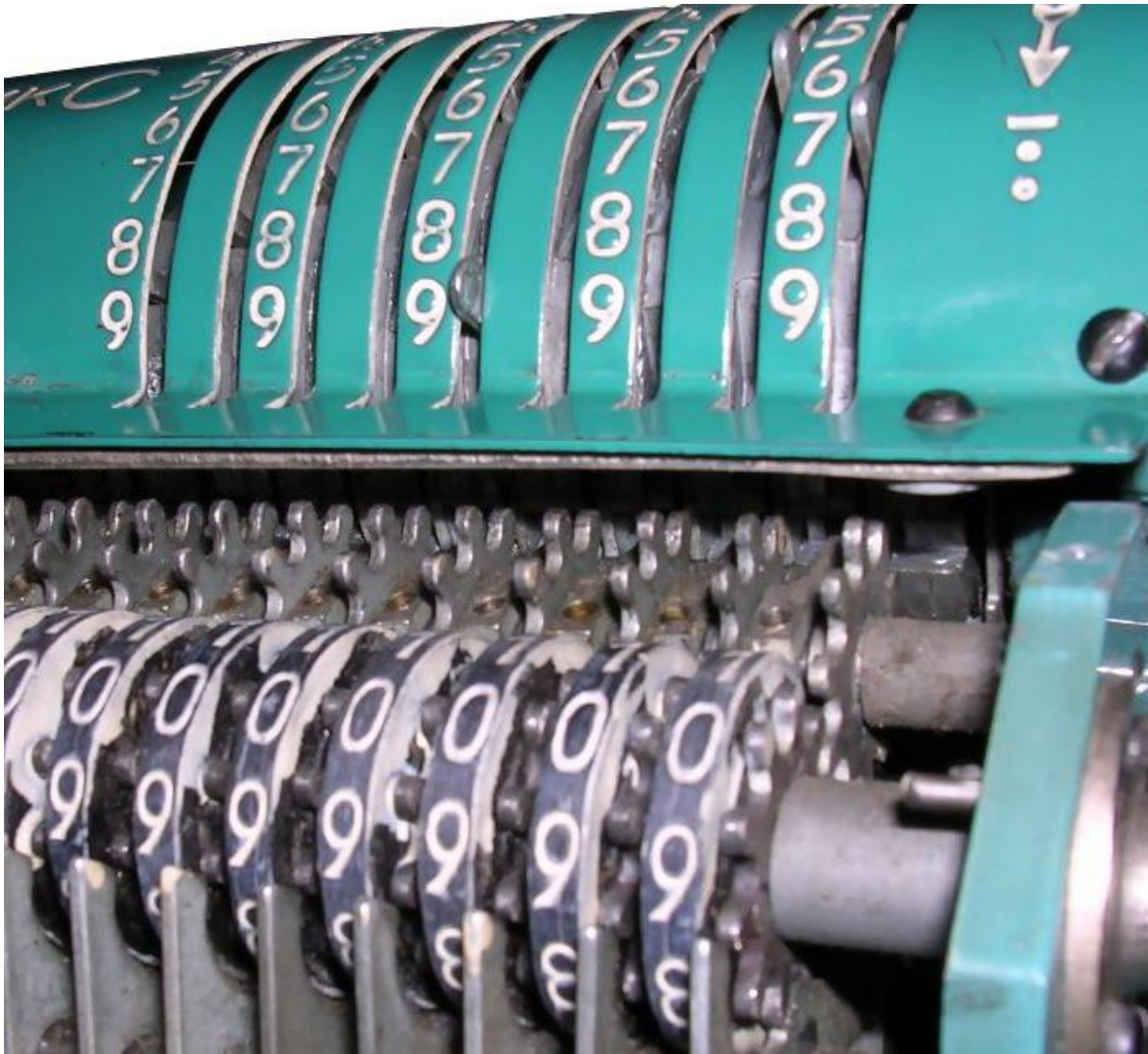
- **a general purpose mechanical computer**
  - Programmable arithmetic logical unit (loops & conditional branches)
  - Control unit
  - Input/output (punched cards)



zdjęcia „poglądowe" – z sieci…

All those machines used decimal numeric system, natural for humans, however, it dramatically complicates the design…

Other disadvantages:
- cost and complexity of precision mechanics
- dimensions
- low speed
- loudness



oldcomputer.info/calculator/pinwheel

A brief history of computing

- **17th century:** Leibniz's Dream

Can we find a universal language for mathematical algorithms that will let us describe and solve any problem?

- Reduce reasoning to a fixed set of basic rules
- Determine truth or falsity of sentences by fixed rules for manipulating sentences

1854 George Bool
Boolean algebra and logic

- **Just two possible values: true or false**

1930s Alan Turing
Turing Machine

- conceptual machine with a finite set of rules and an infinite tape memory for computation
- there is no physical computer that can do more than a Turing machine

1945 John von Neumann, (John Mauchly, Presper Eckert)
model of the computer with the main subsystems: processor, memory & I/O devices

A brief history of computing

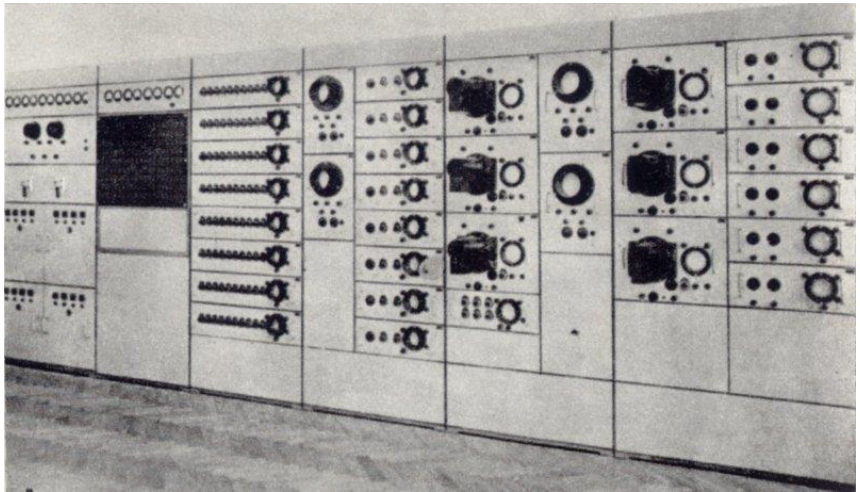
20th century: rapid development of electrical engineering

Analog computers

- input output signals represented by continuously changeable signals
- modular network of integrators, differentiators, logarithmic and summing amplifiers…

- Very convenient for solving particular class of problems
- And very cumbersome in usage and service…

- Faster and cheaper than early **digital** machines…



Rozwiązanie problemu: maszyny liczące oparte na **systemie dwójkowym.**

Podstawowy element funkcjonalny – **klucz przełączający**
wykorzystuje pracę dwustanową (otwarty-zamknięty, przewodzi-izoluje itp) elementów:

- **elektromechanicznych,**
  (łącznik, przekaźnik elektromagnetyczny)

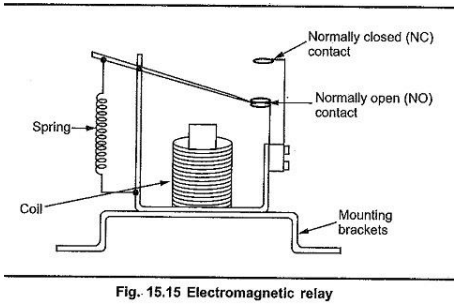
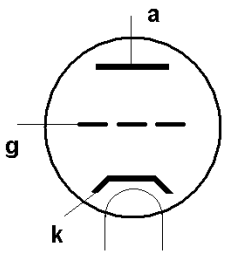


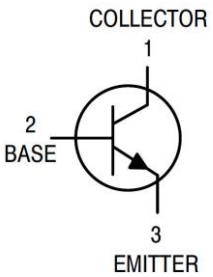
Fig. 15.15 Electromagnetic relay

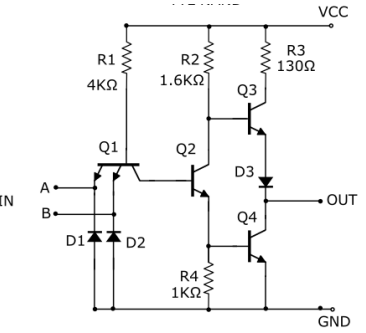
- **elektronowych (lampy elektronowe)**
  trioda, dekatron – licznik i impulsów, lamp pamięciowe, wyświetlacze jarzeniowe…



- **tranzystorów i układów scalonych**

  (kolejny wykład)





wygodna (szybkość, stabilność, niezawodność, możliwość integracji/miniaturyzacji) **realizacja sprzętowa podstawowych zarówno funkcji logicznych,**

jak i bardziej złożonych bloków funkcjonalnych.

wada: system dwójkowy jest nienaturalny i niewygodny dla człowieka

- konieczność stosowania konwersji
- pewnym rozwiązaniem jest zapis szesnastkowy i dwójkowo-dziesiętny (BCD)

Obecne odstępstwa od elementów dwustanowych:

pamięci Flash MLC, TLC i QLC

sposoby transmisji danych (modulacja) np. w telekomunikacji

Architektura komputera

Podstawowe bloki funkcjonalne typowego komputera (systemu mikroprocesorowego)

- **procesor (jednostka arytmetyczno-logiczna i sterująca)**
- **pamięć (RAM, ROM, danych, instrukcji programu…)**
- urządzenia wejścia-wyjścia
- magistrale (szyna adresowa, danych, struktura połączeń sygnałów kontrolnych)

Architektura komputera – sposób wzajemnej konfiguracji (połączenia ze sobą) powyższych elementów.

Lub bardziej formalnie:

Functional operation of the individual hardware units within a computer system, and the flow of information and control among them.

(from Richard C. Dorf Computers, Software Engineering, and Digital Devices)

von Neumann Architecture (machine)

1945 John von Neumann, (John Mauchly, Presper Eckert)

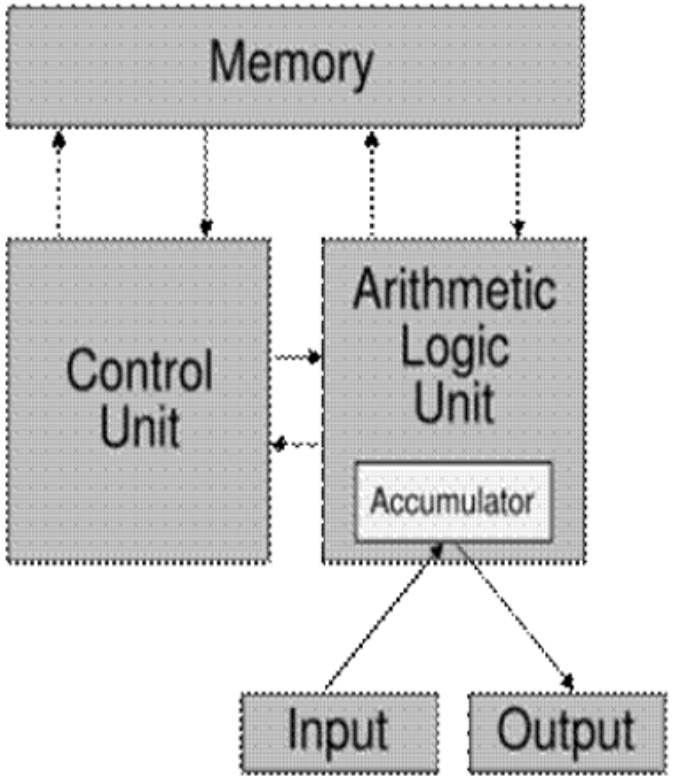
Model of the computer with the main subsystems:

- Arithmetic Logic Unit (ALU)
- Control Unit (CU)

} Processor (CPU - Central Processing Unit)

- one common memory (and bus) for both:
  program instructions and data

- Input/Output (I/O) system



- many modern computers and CPUs are more or less based on the Von Neumann Architecture:
IBM PC, Motorola MC6800…

von Neumann Architecture

- **program instructions and data are stored in one common memory** as numbers (in binary form)

- instructions and data are transferred between memory and CPU through **common shared bus**

- memory is divided into storage units of fixed size: **memory cells**.
- each memory cell has an **unique address**
- **memory cell is minimum accessible unit of memory**

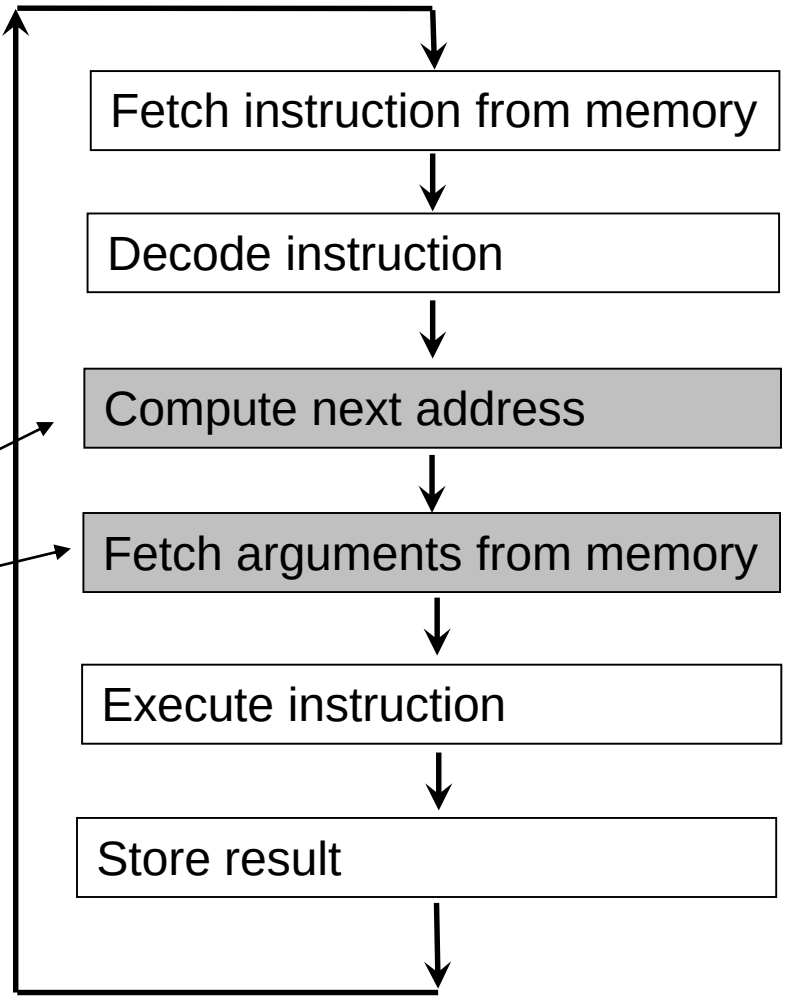
- **CPU executes program instructions sequentially**

Instruction Processing

• CPU is a kind of sequential circuit:

a finite state machine:

i.e. executes instructions
sequentially and repeatedly,
clocked by the pulses of clock signal



• not all phases are needed
by every instruction

• each phase of instruction cycle is called
machine cycle.

• machine cycles may take variable
number of clock ticks
(sometimes 1 clock pulse per 1 machine cycle)

von Neumann Architecture

von Neumann bottleneck:

- only one bus, used for both data transfers and instruction fetches:

- data transfers and instruction fetches must be scheduled
  (they can not be performed at the same time)

- throughput (data transfer rate) between the CPU and memory is limited

program code stored in Random Access Memory can be modified during runtime!

- selfmodification
- accidental modification, due to errors
- malicious modification

Memory protection is needed…

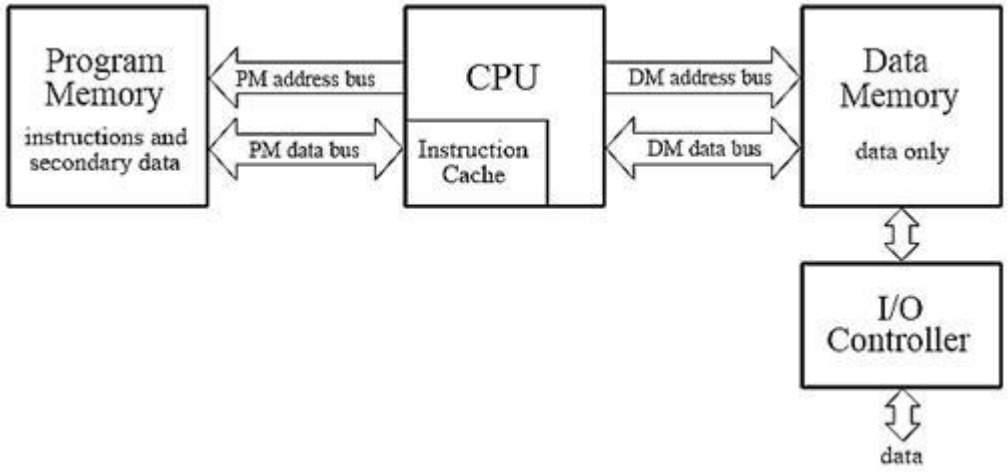
Harvard Architecture

• (at least…) **two separated memory systems for instructions and data**
• connected to the CPU through **separated buses**

• instructions and data can be fetched and transferred simultaneously on both busses

- greater throughput (data transfer rate)

• address spaces, data widths, implementation technologies of these memories can differ

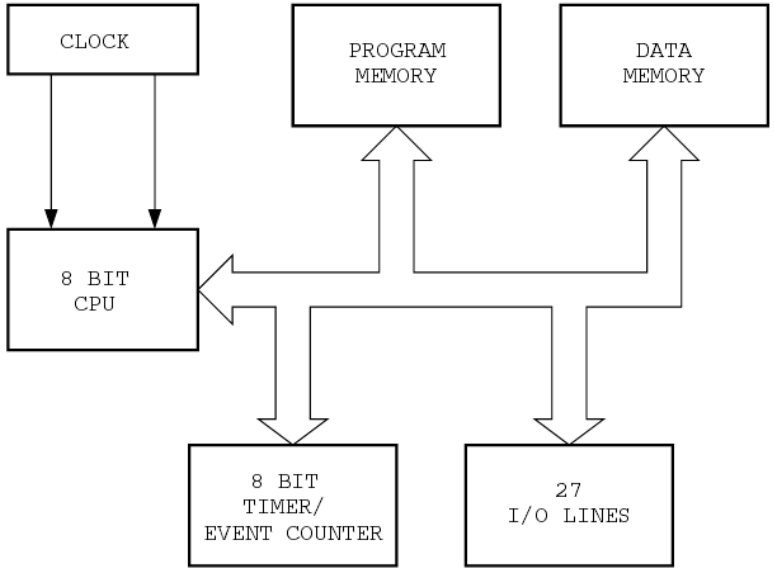


Modified Harvard Architecture (1)

- **separate memory blocks** of different type with **one shared bus**

  simple 8bit microcontrollers (single-chip microcomputers like 8051, MC6805)
  Program: Read-Only-Memory (Non-volatile, Flash, EPROM)
  Data: Random-Access-Memory (Read-Write)

- program memory can store constant data (fonts, sounds, text strings)
- special instructions allow to load them directly to the CPU core



Modified Harvard Architecture (2)

Memory hierarchy in modern personal computer

- main operational memory – von Neumann architecture

- level 1 cache – Harvard architecture

  - concurrent access to the instructions and data may reduce stalls and structural hazards in pipelined processors

Computer Architecture

Any digital computer (with CPU, memory and I/O) can (given enough time) complete any algorithm*

like Infinite Monkey Theorem…

*taken from:
Using the Intel MCS-51 Boolean processing capabilities, Intel Corporation, 1980

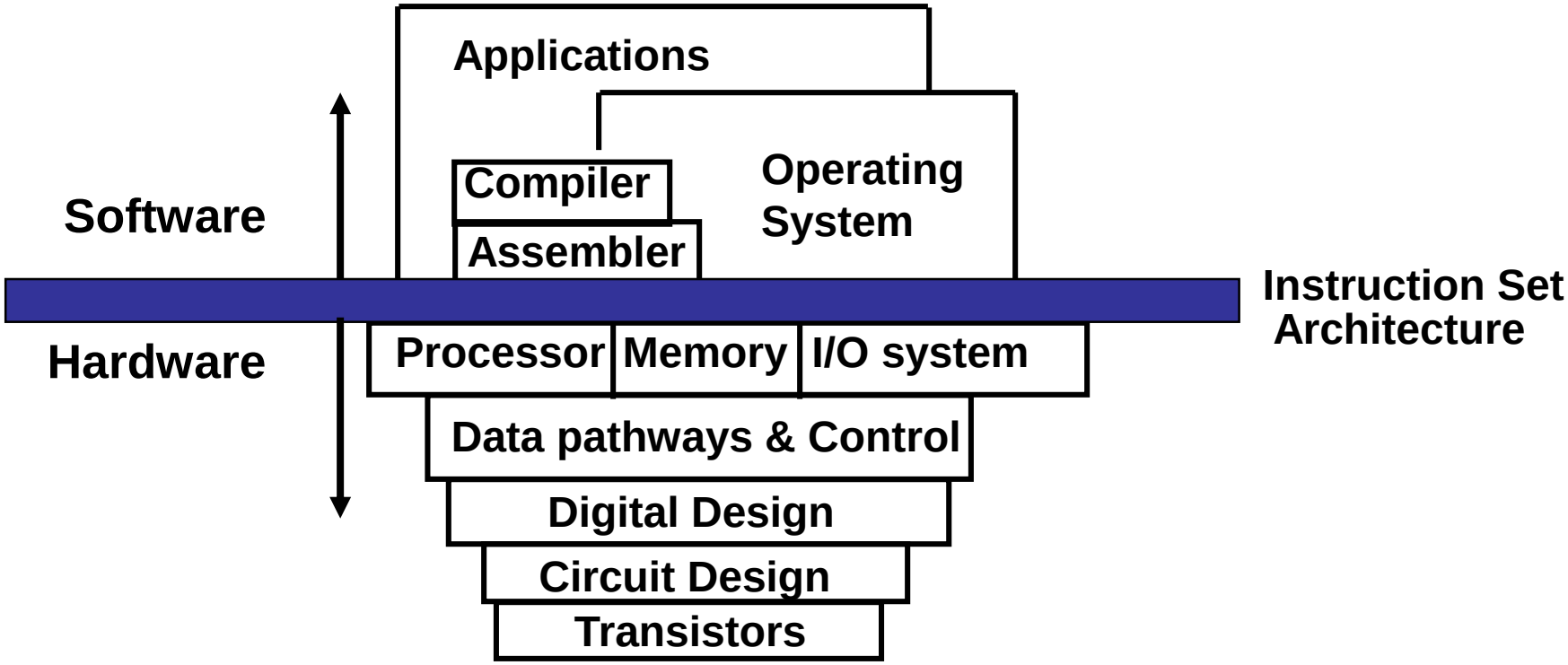
Computer Architecture

ISA – Instruction Set Architecture – programmer's model

i.e. how computer system looks to programmer:

- number and type of registers
- instruction formats, operation code set
- specifies data types and data structures and storage elements
- addressing modes and accessing data items
- I/O handling

Computer Architecture = ISA + Hardware Organization



Architektura komputera

To **cechy systemu komputerowego widziane przez programistę**:

• typ i lista rozkazów procesora - dawniej, obecnie określana jako:

Instruction Set Architecture – model programowy komputera

• liczba i rodzaj rejestrów
• typy danych i tryby adresowania pamięci
• sposób komunikacji z urządzeniami I/O (oraz ich rodzaj i liczba)

Czyli: co komputer może wykonać sprzętowo, co i jak trzeba oprogramować…?
I jak optymalnie przygotować dane i stworzyć algorytm?

Współczesny komputer PC:

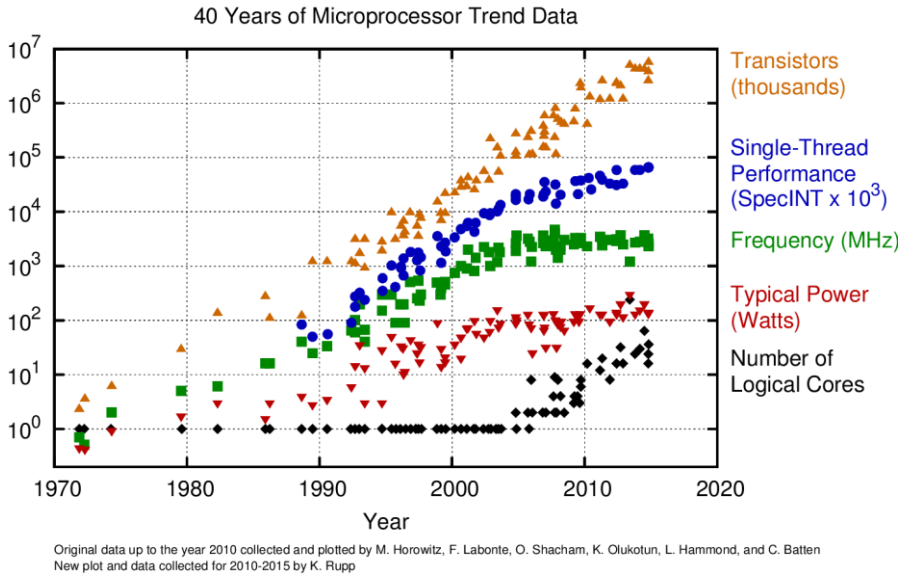
Zasada działania* i pamieć RAM – architektura von Neumanna

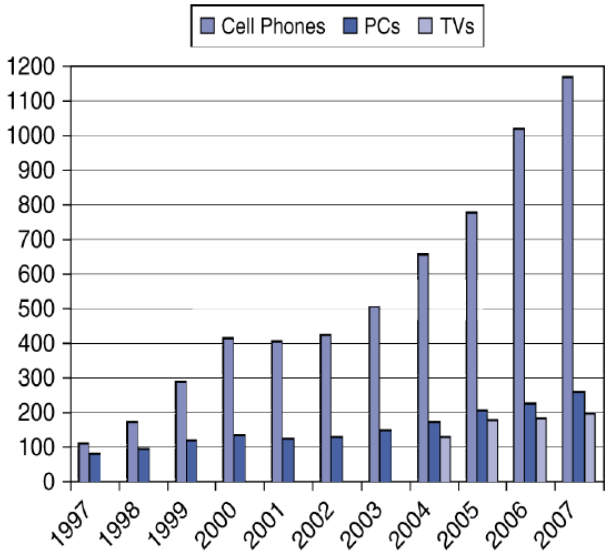
Pamieć cache – architektura harwardzka

* mechanizmy przetwarzania we współczesnych procesorach odbiegają od modelu von Neumanna

Podsumowanie

• procesory/mikrokontrolery są elementem powszechnie spotykanym w różnego typu urządzeniach





• niezależnie od typu, generacji i stopnia zaawansowania współcześnie spotykanych systemów mikroprocesorowych, posiadają one pewne cechy wspólne

• **zrozumienie zasady działania podstawowych układów** logicznych, bloków funkcjonalnych **prostego komputera** i jego modelu programowego **ułatwi poznanie złożonych mechanizmów** (np. przetwarzanie potokowe), wpływających (**razem z oprogramowaniem**) na wydajność systemu komputerowego

• jeśli używa się jakiegoś skomplikowanego narzędzia – warto poznać zasadę jego działania…