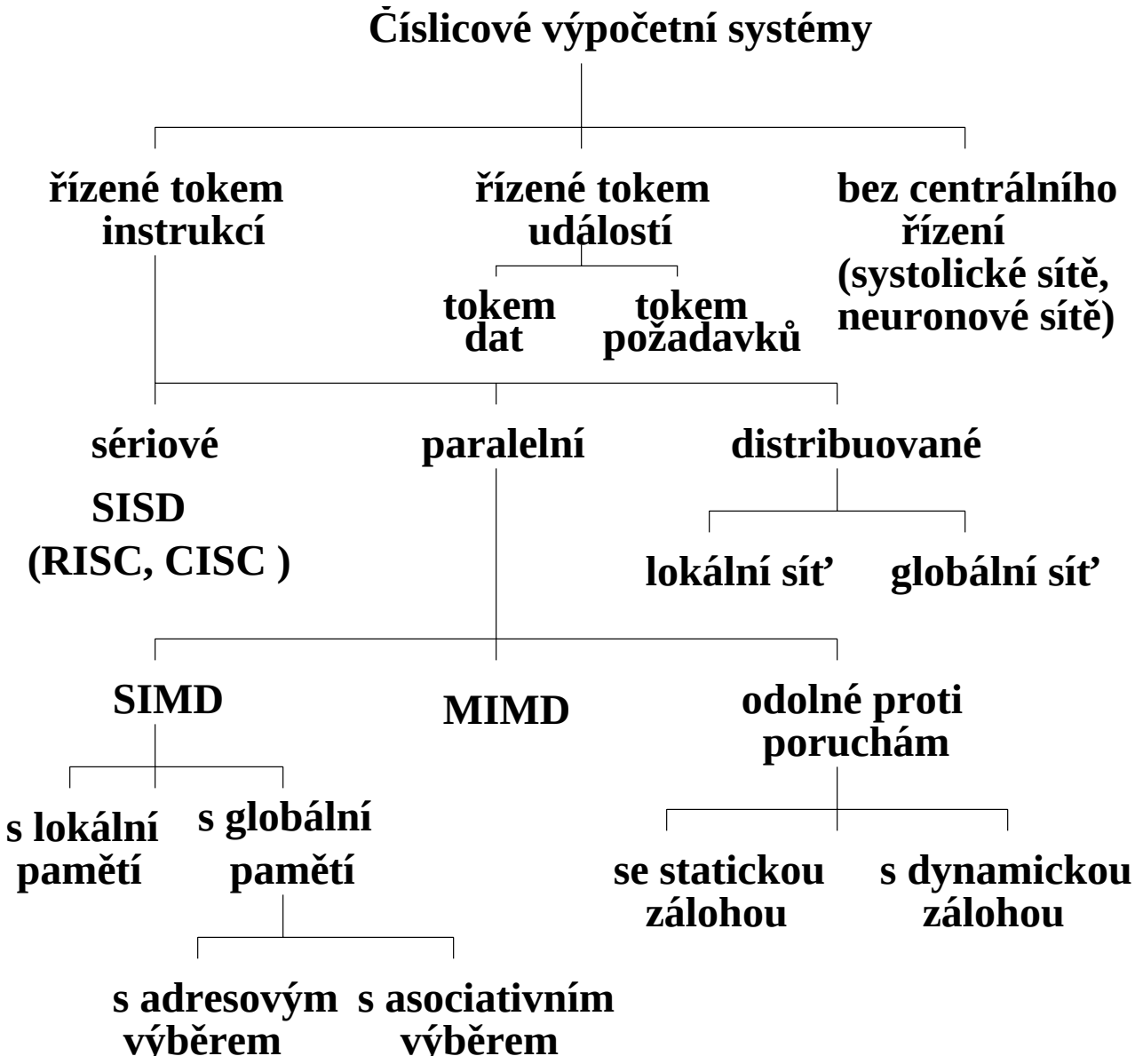
Rozdělení číslicových počítačů

– podle použité technologie (tzv. generace)

Generace Rok	Technologie příklad počítače	Kapacita oper. paměti	Rychlost [MIPS]	Oblast využití
1. 1949	elektronky (IAS, UNIVAC1, ENIAC)	1KB	10^4 op/s 0.01	vědecko-technické výpočty (VT) dávkové zpracování
2. 1957	tranzistory (DEC PDP-1, IBM 1401, IBM 7094)	10KB	10^5op/s 0.1	VT výpočty hromadné zpacování dat (HSD)
3. 1964	malá integrace (SSI) (IBM S/360)	1MB	10^6op/s 1	Systémy se sdílením času, interaktivní zpracování dat
3,5. 1971	střední integrace (MSI) (IBM S/370, CRAY 1)	1MB	10^6op/s 1	Systémy reálného času Sítě (terminál)
4. 1981	velká a velmi velká integrace (LSI, VLSI) (Intel 80286, 80386)	10MB 10GB	10^7op/s 10	Osobní výpočty Sítě LAN WAN
5. ? 1990	velmi velká integrace (Intel 80486, Pentium, Alfa 21064, IBM 6*860	>10GB	$>10^9$op/s 1001000	Porozumění přirozené řeči, umělá inteligence
6. ?	biologickké prvky neuroprocesory	?	?	?

– podle způsobu řízení výpočetního systému



CISC (Complex Instruction Set Computers)

- mikroprogramové řízení
- bohatý instrukční soubor
- universální instrukce pro podporu vyšších programovacích jazyků
- proměnná délka instrukce
- k vykonání instrukce je potřeba několika strojových cyklů

RISC (Reduced Instruction Set Computers)

- jednoduchý soubor instrukcí
- stejná délka instrukcí
- instrukce se vykonávají během jediného strojového cyklu

Výhody:

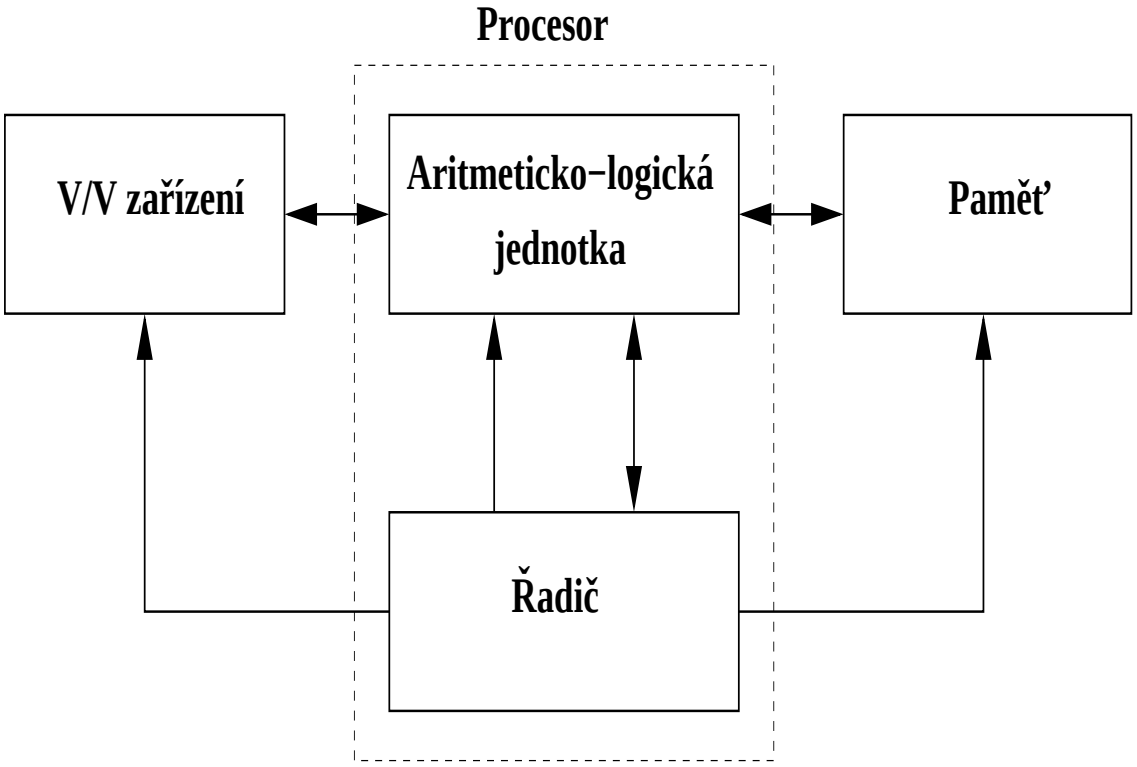
- **rychlost**
- **jednodušší hardware**
- **kratší doba návrhu**

Kategorie počítačů

- **mikrořadiče (microcontrolers) – malé specializované mikroprocesory instalované např. v automobilech, mob. telefonech apod.**
- **mikropočítače**
  - **osobní počítače (PC) – použití pro jednoduché aplikace (zpracování textů, tabulkové kalkulátory, úetní programy**
  - **pracovní stanice (workstation) – výkonné stroje použitelné v oblasti CAD/CAM, vývoj software, modelování, počítačová grafika**
- **minipočítače – obvykle vybavené řadou terminálů, použitelné obvykle pro potřeby středně velkých firem, CAD/CAM,**
- **sálové počítače (mainframes) – pracují v klimatizovaném prostředí, použitelné pro velké firmy zpracovávající data velkých objemů (banky, pojišťovací společnosti), charakteristická je velká vnější paměť (řádově 100 GB)**
- **superpočítače**
  - **vektorové procesory**
  - **paralelní počítače**

  Použití: rozsáhlé výpočty v matematice, fyzice, mechanice, simulace

Architektura číslicového počítače (John von Neumann)



- Aritmeticko-logická jednotka - slouží k provádění aritmetických a logických operací (součet, součin, AND, OR, XOR, atd.)
- Řadič - řídí činnost jednotlivých částí počítače

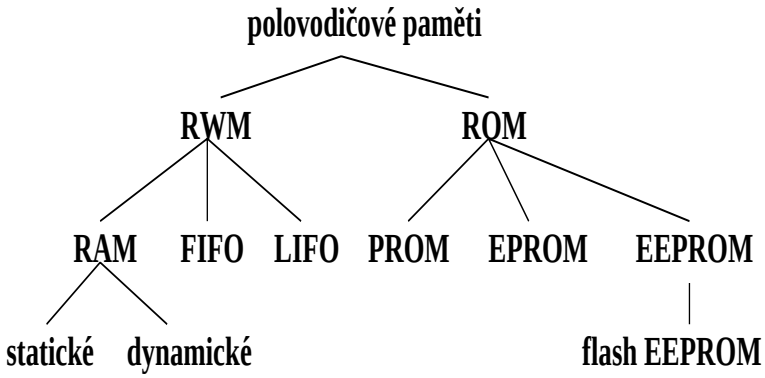
  ALU + Řadič = CPU (Central Processing Unit) - procesor
- Paměť (vnitřní, operační) - uchovává data a instrukce, elementární jednotkou paměti - 1 bit (pravda/nepravda)

  Kapacita paměti - ve slabikách tzv. bytech 1B = 8 bit, popř. násobcích

  K (kilo) = 2^{10} = 1024 bytů

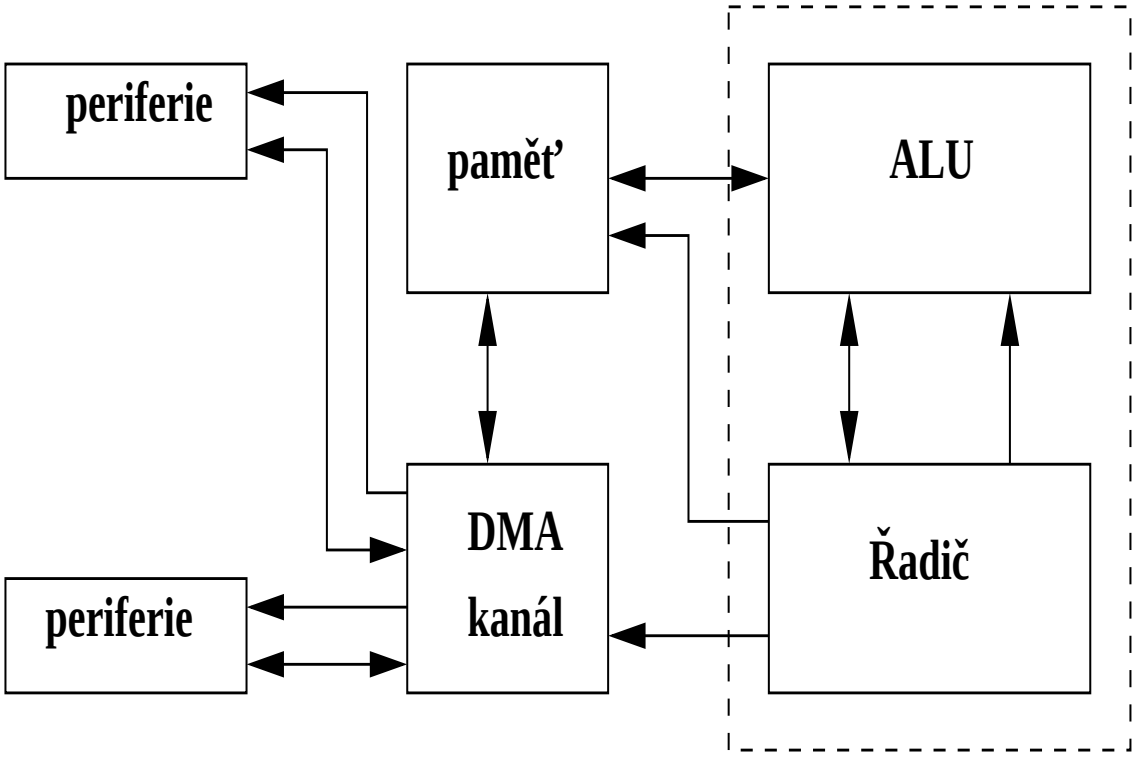
  M (mega) = 2^{20} = 1048576 bytů

  G (giga) = 2^{30} = 1073741824 bytů

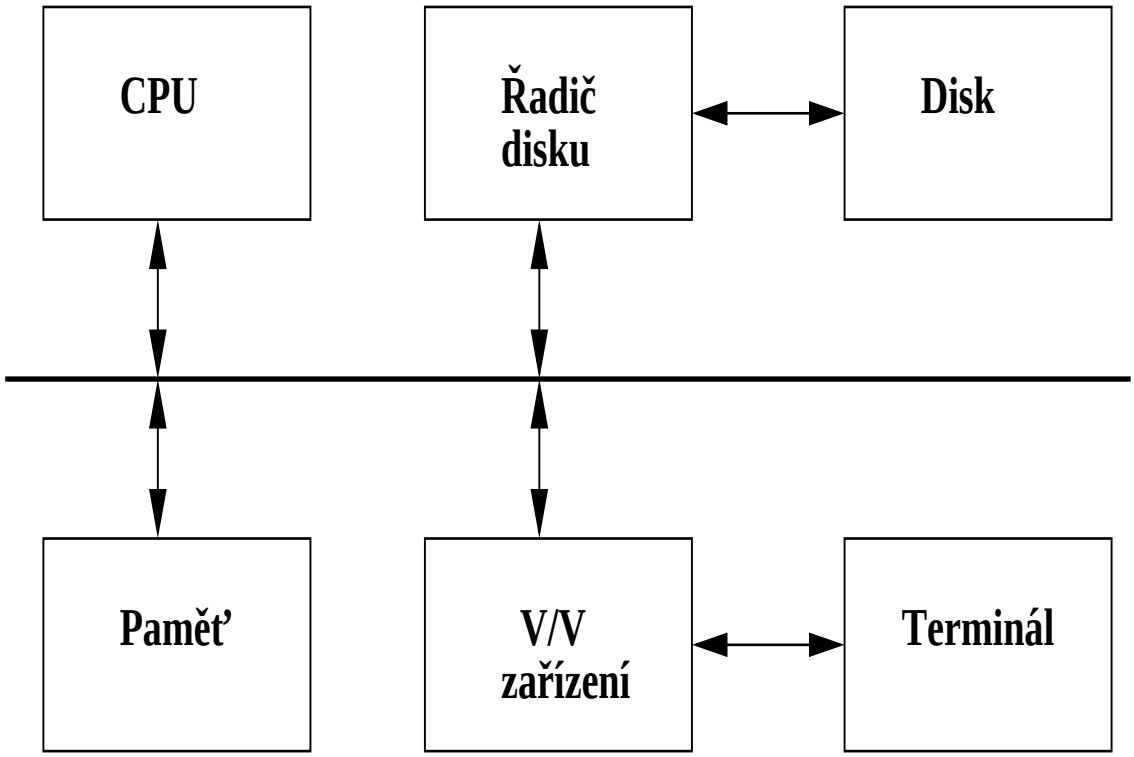


– Vstupní/výstupní zařízení (periferní zařízení) – komunikace počítače s okolím
  – obrazovka
  – klávesnice
  – tiskárny
  – vnější paměti (disky, pásky,CDROM)

Řízení přenosu dat mezi vstupně výstupními zařízeními - *kanál* -řídí na pokyn procesoru periferní řenosy jednotek, které jsou k němu připojeny. Komunikuje s pamětí přímo bez účasti procesoru (Direct Memory Access).



Sběrnicová architektura počítače



Sběrnice - propojuje jednotlivé funkční bloky počítače, je to norma určující elektrické, mechanické, časové propojení jednotlivých prvků.

- **datová sběrnice - přenos dat mezi zařízeními**
- **adresová sběrnice -přenos adresy pro jednotlivá zařízení**
- **řídicí sběrnice - signály pro řízení jednotlivých komponent**

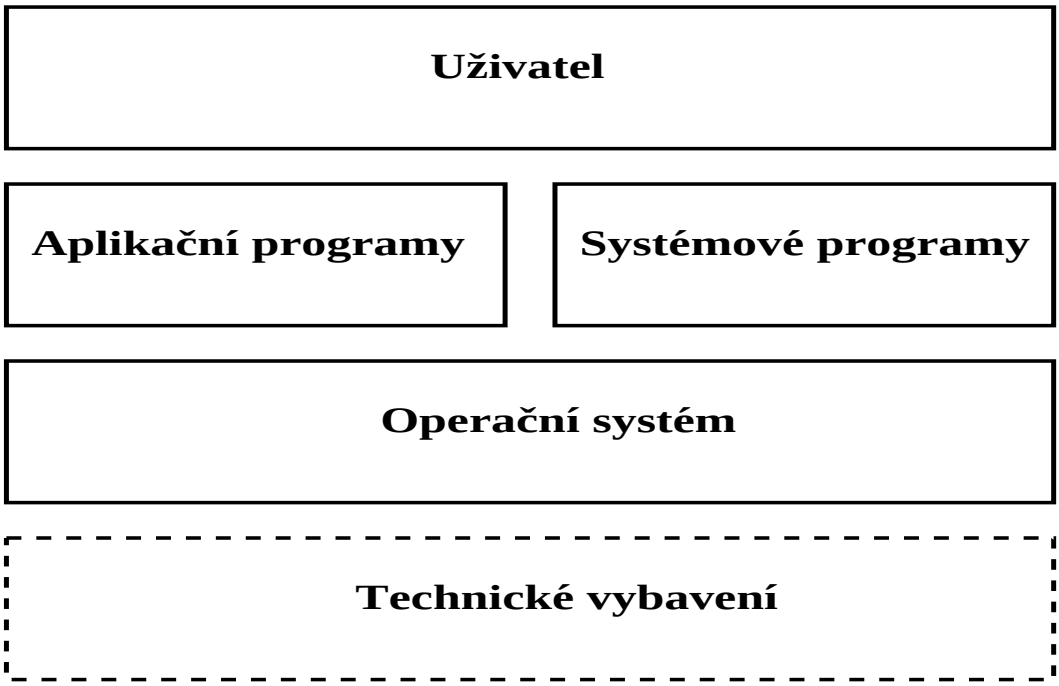
Systémové programové vybavení

Řešení úloh na "holém" počítači je obtížné, programátor se musí zabývat obsluhou základních prostředků

$$\Downarrow$$

systémové programové vybavení

Systémové programové vybavení transformuje "holý počítač" na tzv. *virtuální počítač*, který je z hlediska ovládání výhodnější.



Operační systém – základní programové vybavení, které má uživatel k dispozici. Je to soubor programů, které zajišťují základní komunikaci mezi jednotlivými komponentami počítače.

- **monouživatelské – umožňují práci pouze jedinému uživateli**
- **multiuživatelské – současná práce více uživatelů**
- **monoprogramní – běh povolen pouze jedinému programu**
- **multiprogramní – současný běh více úloh (programů)**

Příklady operačních systémů:

- **MSDOS – monoprogramní, monouživatelský**
- **OS2 – multiprogramní, monouživatelský**
- **UNIX – multiprogramní, multiuživatelský**

Systémové programy – prostředky pro údržbu systému, prostředky pro vytváření aplikačních programů

- ladící prostředky
- editory
- překladače
- assembler
- zaváděč
- programy pro údržbu

Aplikační programy :

- vlastní
- firemní
  - textové procesory
  - tabulkové kalkulátory
  - CAD systémy
  - databázové programy atd.

Řešení úloh na počítači

Řešení úloh na počítači nespočívá jen v psaní programů. Je to činnost, která v praxi obsahuje celou řadu etap.

Etapy:

- **definice problému**
- **nástin řešení**
- **sestavení algoritmu**
- **kódování programu**
- **ladění programu**
- **zpracování dokumentace a údržba programu**

Definice problému

V této fázi řešení úlohy se definuje vlastní problém a určuje se :

- **jaká budou vstupní data**
- **jaká budou výstupní data**
- **jak bude program reagovat na nesprávné, popř. neúplné vstupy**

Nástin řešení

- **rozmyšlení a návrh celkové strategie řešení problému (např. výběr vhodných numerických metod atd.)**
- **částečný rozklad na podproblémy, které mohou být řešeny samostatně jednotlivými členy týmu**
- **definice rozhraní mezi jednotlivými podproblémy**

Sestavení algoritmu – algoritmizace

Algoritmus = návod, jak provést určitou činnost, který splňuje následující vlastnosti :

1. **Je *elementární*, tj. skládá se z konečného počtu jednoduchých, snadno realizovatelných činností (kroků)**
2. **Je *determinovaný,* tj. po každém kroku lze určit, zda proces skončil, popř. jakým krokem se bude pokračovat.**
3. **Je *konečný*, tj. počet opakování jednotlivých kroků algoritmu je vždy konečný.**

4. Je *rezultativní*, tj. vede ke správnému výsledku.
5. Je *hromadný,* tj. lze jej použít k řešení velké skupiny podobných úloh

Algoritmus je prováděn procesorem ⇒ při formulaci algoritmu bychom měli znát procesor.

Metody návrhu algoritmu:

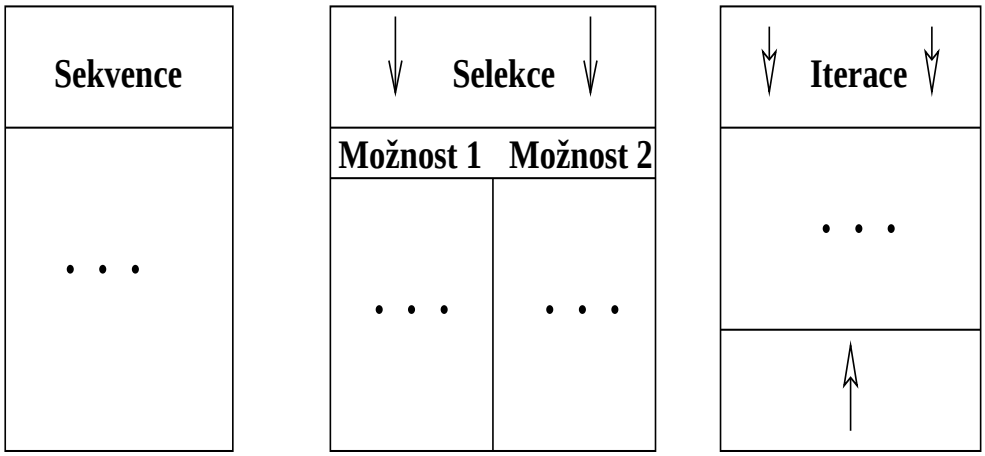
- shora dolů = opakovaný rozklad složitějších problémů na jednodušší
- zdola nahoru = z elementárních prostředků postupně vytváříme celek

Základní složky algoritmu:

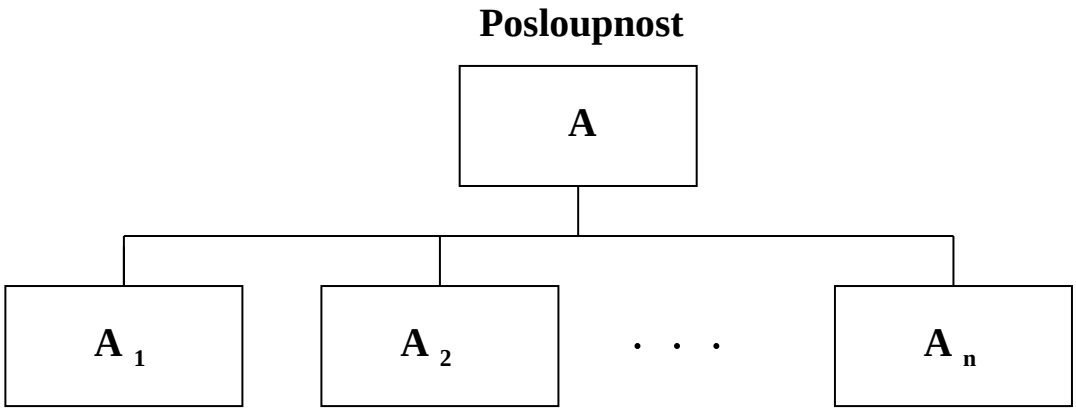
- *posloupnost* (sekvence) - je tvořena jedním nebo několika kroky, které se provedou právě jednou v daném pořadí
- *cyklus* (iterace) - část algoritmu, která se opakuje na základě splnění určité podmínky. Skládá se z podmínky opakování a z těla cyklu.
- *podmíněná operace* (selekce) - představuje větvení algoritmu. Je tvořena podmínkou a jednou, dvěma, nebo více výběrovými složkami

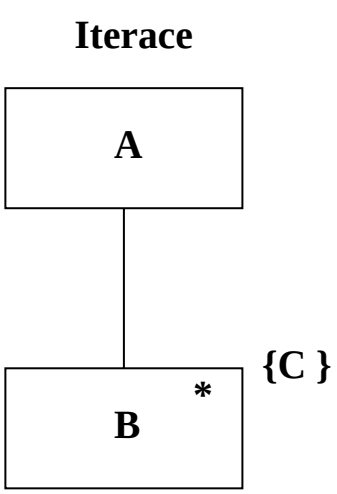
Popis algoritmů

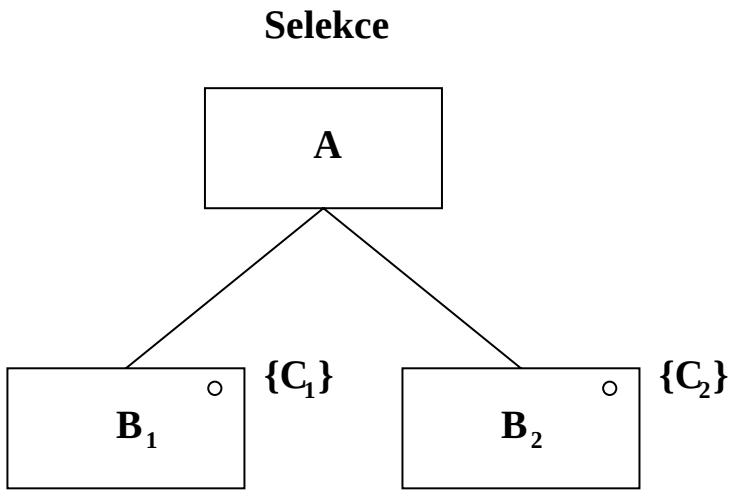
- jazyk pro popis algoritmů (Program Description Language) - vychází ze slovního popisu algoritmu, k popisu jsou často použita klíčová slova programovacího jazyka (tzv. pseudojazyk)
- strukturogramy - graficky znázorňují strukturu algoritmu



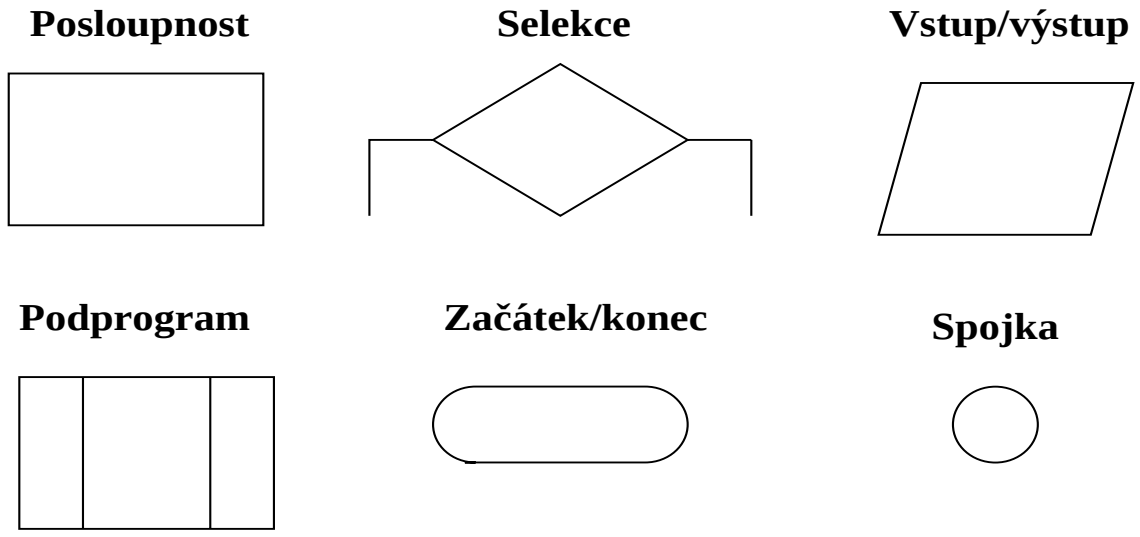
– Jacksonovy diagramy







– Vývojové diagramy - znázorňují tok řízení v algoritmu. Značky jsou upravené normou.



Vývojové diagramy jsou graficky náročné a prakticky se dnes už nepoužívají.

Kódování programu

Etapa, ve které se navrený algoritmus zapisuje algoritmu ve zvoleném programovacím jazyce.

Programovací jazyky :

– *nižší úrovně*

  – strojový kód
  – jazyky symbolických instrukcí

  Výhody : univerzální – lze přistupovat k libovolným komponentám počítače.

  Nevýhody : závislost na použitém procesoru, nepřehledný zápis algoritmu

– *vyšší programovací jazyky*

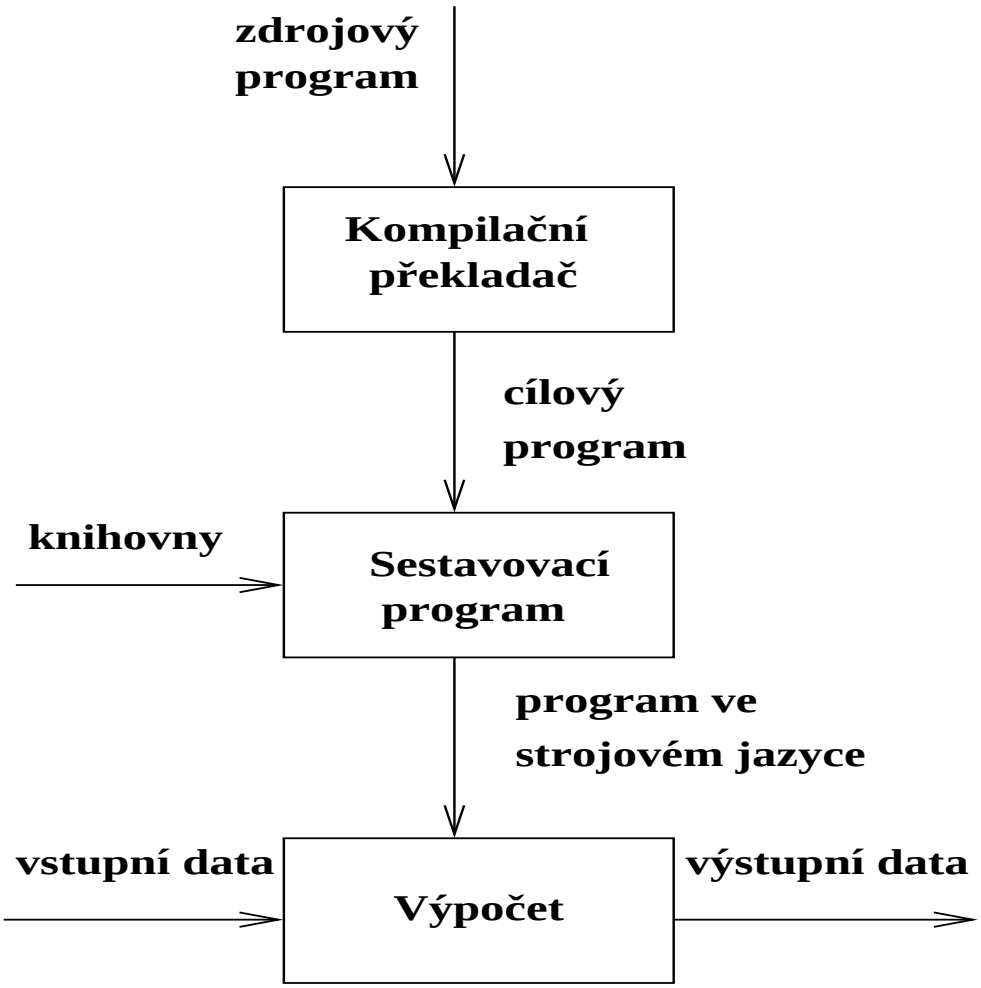
  Výhody: umožňují přehledný zápis, jsou obvykle nezávislé na použitém procesoru

  Nevýhody: některé jazyky neumožňují využívat všechny možnosti technického vybavení (hardware).

Překladače programovacích jazyků

– provádí překlad z jednoho programovacího jazyka do strojového kódu, popř. do jiného jazyka

- kompilační překladače



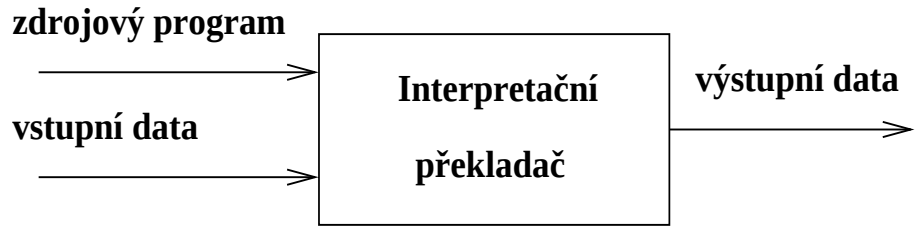
Části překladače:

– lexikální analyzátor

– syntaktický analyzátor

– generátor kódu

- interpretační překladač



Ladění programu

Cílem ladění je odstranit chyby v programu.

Chyby:

- *syntaktické* - prohřešky proti pravidlům syntaxe programovacího jazyka. Na tyto chyby upozorní překladač
- *logické*
  - chybný algoritmus
  - chybné kódování algoritmu
  - chyby při přepisu programu

Pro zjištění logických chyb - nutná volba vhodných testovacích dat (data volit tak, abychom prošli všechny větve programu).

Zpracování dokumentace a udržování programu

Dokumentace:

- uživatelská dokumentace (návod k použití) – obsahuje stručný popis funkce programu, popis tvaru vstupních a výstupních dat, způsob spouštění a popis významu signalizovaných chyb.
- technická (programátorská) – je podkladem pro opravy, popř. úpravy programu, a to i jinými osobami, než je autor programu. Hlavní částí musí být detailní a srozumitelný popis algoritmu, použité datové struktury, organizace dat ve vstupních, popř. výstupních souborech atd.

Příklad: Algoritmus řešení kvadratické rovnice

1. **Definice problému**

$$a \cdot x^2 + b \cdot x + c = 0$$

Vstup : **koeficienty *a, b, c*** – **libovolná reálná čísla**
Výstup : **kořeny $\mathbf{x}_1$, $\mathbf{x}_2$** – **komplexní čísla**

2. **Nástin řešení**

a=0 ∧ b=0 ∧ c≠0 → **nemá řešení**

a=0 ∧ b=0 ∧ c=0 → **nekonečně mnoho řešení**

a=0 ∧ b≠ 0 → **lineární rovnice** $x = -\frac{c}{b}$

a≠ 0 → $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$

$b^2 - 4 \cdot a \cdot c \geq 0$ → $x_1 = \frac{-b}{2 \cdot a} + \frac{\sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$

$x_2 = \frac{-b}{2 \cdot a} - \frac{\sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$

$b^2 - 4 \cdot a \cdot c < 0$ → $x_1 = \frac{-b}{2 \cdot a} + j\frac{\sqrt{|b^2 - 4 \cdot a \cdot c|}}{2 \cdot a}$

$x_2 = \frac{-b}{2 \cdot a} - j\frac{\sqrt{|b^2 - 4 \cdot a \cdot c|}}{2 \cdot a}$

3. Algoritmus - vývojový diagram