

# PB001: Úvod do informačních technologií

Luděk Matyska (Eva Hladká)

Fakulta informatiky Masarykovy univerzity

podzim 2015



EVROPSKÁ UNIE



MINISTERSTVO ŠKOLSTVÍ,  
MLÁDEŽE A TĚLOVÝCHOVY



OP Vzdělávání  
pro konkurenceschopnost



INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

- 1 Programové vybavení
- 2 Číselné soustavy
- 3 Operační systémy

- Nadstavba technických prostředků
- Vrstvy operačního systému:
  - Technické vybavení
    - Správa paměti
    - Správa procesů
    - Správa periferií (I/O)
    - Správa souborů dat
    - Uživatelský prostor (nepřesné)

# Programové vybavení – jiný pohled

- Operační systém
  - UNIX, Linux, OS/370, MS Windows, ...
- Programovací jazyky
  - C, Pascal, Ada, Occam, ML, Prolog, perl, python, Java, ...
- Podpůrné nástroje
  - debuggery, profilery, ...
- Aplikační programy

# Programovací jazyky

- Rozlišujeme
  - Styl
  - Míru abstrakce
  - „Dialekt“

# Programovací jazyky – styl

- Imperativní/Procedurální: C, Fortran
- Objektově orientované: Java, C++
- Deklarativní/Funkcionální: ML, Lisp, MIRANDA
- Deklarativní/Logické: Prolog, GHC
- S jediným přiřazením: SISAL
- Produkční systémy: OPS5
- Sémantické sítě: NETL
- Neuronové sítě: SAIC ANSpec

# Procedurální vs. deklarativní styl

```

fac := 1;
if n > 0 then
  for i:=1 to n do
    fac := i*fac;

```

```

|   fac(0)    := 1;
|   fac(n>0) := n*fac(n-1);
|
|-----
|
|   fac(0,1).
|   fac(N,F1*N) :- fac(N-1,F1).
|

```

# Programovací jazyky – míra abstrakce

- Strojový jazyk: přímo kódy jednotlivých instrukcí
- Assembler: jména instrukcí, operandy, pojmenované cílové adresy skoků
- Vyšší jazyky: obecné konstrukty, tvoří „kontinuum“
  - Agregované datové typy
  - Cykly namísto skoků
  - Procedury a funkce
  - Procesy a vlákna



# Programovací jazyky – implementace

- Překladače
  - Zdrojový kód–mezijazyk–strojový jazyk
  - Překlad a sestavení

# Programovací jazyky – implementace

- Překladače
  - Zdrojový kód–mezijazyk–strojový jazyk
  - Překlad a sestavení
- Interprety
  - Abstraktní počítač
  - Vhodné pro složité operace (např. práce s texty, s maticemi a algebraickými objekty)

# Programovací jazyky – implementace

- Překladače
  - Zdrojový kód–mezijazyk–strojový jazyk
  - Překlad a sestavení
- Interprety
  - Abstraktní počítač
  - Vhodné pro složité operace (např. práce s texty, s maticemi a algebraickými objekty)
- Just-in-time překladače (nejen Java)
  - Známý již od osmdesátých let (řešil se tak nedostatek paměti)

# Výpočetní model

- Souvislost mezi *architekturou* a *jazyky*
  - von Neumannova architektura a imperativní jazyky
  - objektová architektura a objektově orientované jazyky
  - redukční architektura a funkcionální programování

# Výpočetní model – varianty

- Datově orientovaný (nejpoužívanější)
- Objektový
- Funkcionální
- Logický

# Aplikace

Dávají počítačům smysl

- Vědecko-technické výpočty (vojenství: atomová bomba)

# Aplikace

Dávají počítačům smysl

- Vědecko-technické výpočty (vojenství: atomová bomba)
- Zpracování informací (Hollerith/IBM: census USA)

# Aplikace

## Dávají počítačům smysl

- Vědecko-technické výpočty (vojenství: atomová bomba)
- Zpracování informací (Hollerith/IBM: census USA)
- Zábava (počítačové hry, video-on-demand)



# Aplikace

## Dávají počítačům smysl

- Vědecko-technické výpočty (vojenství: atomová bomba)
- Zpracování informací (Hollerith/IBM: census USA)
- Zábava (počítačové hry, video-on-demand)
- Řízení (management strojů i lidí, real-time systémy)

# Číselné soustavy

- Definovány základem: desítková, dvojková, osmičková, šestnáctková
- Volně mezi sebou převoditelné (celá čísla bez ztráty přesnosti)
- Celá čísla a zlomky
- Reálná čísla
- Konečná reprezentace

# Číselné soustavy

- Definovány základem: desítková, dvojková, osmičková, šestnáctková
- Volně mezi sebou převoditelné (celá čísla bez ztráty přesnosti)
- Celá čísla a zlomky
- Reálná čísla
- Konečná reprezentace
- První počítače v desítkové soustavě

# Dvojková soustava

- Základ číslo dvě:
  - pouze dvě číslice/dva stavy
  - vhodná pro reprezentaci v elektrických systémech

# Dvojková soustava v počítači

- Konečná reprezentace: interval hodnot
- Pro reálná čísla:
  - Rozlišitelnost (nejmenší zobrazitelné číslo):  $X + \epsilon > X$  a  $X + \epsilon/2 = X$
  - Přesnost (rozsah)
  - Zobrazení: mantisa  $m$  a exponent  $e$   
 $0 \leq m \leq 1 \wedge x = m \cdot 2^e$
- Záporná čísla:
  - Přímý kód
  - Inverzní kód
  - Dvojkový doplňkový kód

# Záporná čísla – zobrazení

- Přímý kód:
  - Přidáme znaménko
  - Dvě nuly:  $+0$  a  $-0$  (**10...00**)
  - Rozsah:  $\langle -MAX, -0 \rangle$  a  $\langle +0, +MAX \rangle$
- Inverzní kód:
  - Přidáme znaménko
  - Dvě nuly:  $+0$  a  $-0$  (**11...11**)
  - Rozsah:  $\langle -MAX, -0 \rangle$  a  $\langle +0, +MAX \rangle$

# Záporná čísla – zobrazení

- Dvojkový doplňkový kód:
  - Inverze bitu a přičtení jedničky
  - Pouze jedna nula ( $\mathbf{11\dots 11}$  je  $-1$ )
  - Nesymetrický rozsah:  $\langle -MAX - 1, -1 \rangle$  a  $\langle +0, +MAX \rangle$
- Skutečně používán v počítačích

# Rozsahy čísel

- Podle počtu bitů:
  - Byte: 8 bitů, tj.  $\langle 0, 255 \rangle$  nebo  $\langle -128, 127 \rangle$
  - Půl slovo, 2 byte: 16 bitů, tj.  $\langle 0, 65535 \rangle$  nebo  $\langle -32768, 32767 \rangle$
  - Slovo, 4 byte: 32 bitů, tj. přibližně  $\langle -2 \cdot 10^9, 2 \cdot 10^9 \rangle$
  - Dvojslovo (nebo dlouhé slovo), 8 byte: 64 bitů, tj. přibližně  $\langle -9 \cdot 10^{18}, 9 \cdot 10^{18} \rangle$



# Racionální čísla

- Formát dle IEEE 754
- Součásti:
  - Znaménko
  - Mantisa (přímý kód, normalizace, **m** bitů)
  - Exponent (v kódu posunutě nuly, **n** bitů)

# Racionální čísla

- Normalizace mantisy (exponent má **n** bitů):
  - Nejvyšší bit vždy jedna: **1.aaaaaa**; **1** nezapisujeme
  - Nejmenší kladné číslo:  **$1.0 \times 2^{-2^{n-1}+1}$**  ( $2^{-127}$  pro **n = 8**)
  - Největší číslo:  **$1.0 \times 2^{2^{n-1}}$**  ( $2^{128}$ )
- Exponent (**n** bitů, dvojková soustava)
  - Přičteme  **$2^{n-1} - 1$**  (=127 pro **n = 8**), abychom získali správnou hodnotu pro uložení
  - **00000000** je **-127**
  - **11111111** je **128**
- Zvláštní a nenormalizovaná čísla

# Racionální čísla II

- Rozsah zobrazení: ⟨Největší záporné, Největší kladné⟩
- Přesnost zobrazení: počet bitů mantisy+1
- Rozlišitelnost: nejmenší nenulové číslo
  - Normalizované vs. nenormalizované ( $2^m$ krát menší,  $m$  počet bitů mantisy)

# Jiné soustavy

- Osmičková
  - $001\ 101\ 101\ 111_2 = 1557_8 = 879_{10}$
- Šestnáctková
  - $0011\ 0110\ 1111_2 = 36F_{16} = 879_{10}$
- Používány především pro „hutný“ zápis binárních čísel

# Operační systémy – trocha historie

- Bootstrap loader
- Spooling
  - Nezávislé zavádění programu a jeho vykonávání
  - Vyžaduje DMA (Direct Memory Access)
  - Zavedlo *multiprogramování*
  - Stále zpracování *dávek* (batch processing)
- Timesharing
  - Virtualizace počítače/CPU
  - Zpracování *interaktivních* úloh
  - Souvisí se zavedení *disků* (Direct Access Storage Device, DASD od IBM, 60tá léta)

# Operační systémy: účel

- *Zkrásnění:*
  - Zjednodušení práce s počítačem
    - Práce s pamětí
    - Práce se soubory
    - Přístup k perifériím

# Operační systémy: účel

## *Sdílení:*

- Zajistit sdílení vzácných zdrojů
- Musí zajistit:
  - Aby to vůbec fungovalo
  - Aby to fungovalo účinně (využití, propustnost, rychlost odezvy)
  - Aby to fungovalo správně
    - Omezení následků chyb (avšak pozor na chyby v samotném operačním systému)
    - Oprávnění k přístupu (autentizace a autorizace)

# OS: problém časování

- Periferie výrazně pomalejší než procesor
- Příklad
  - 1 GHz Pentium IV:  **$1 \cdot 10^9$**  operací za sekundu
  - Běžný disk: 10 ms pro přečtení 1 byte
  - Poměr **1 : 10 000 000**
  - Stejně zpomalení člověka: 1 úhoz na klávesnici cca 20 dní.
- Možné řešení: prokládání I/O a výpočtu
  - Spust' diskovou operaci  
Prováděj instrukce nad jinými daty (alespoň 1 M instrukcí)  
Počkej na dokončení
  - Příliš těžkopádné a složité



# OS časování: jiné řešení

```
Proces 1 {  
    Spust' diskovou operaci  
    Počkej na dokončení  
    Zpracuj získaná data  
}  
Proces 2 {  
    Nějaká jiná aplikace  
}
```

- Přehlednější
- OS musí „přepínat“ mezi procesy (*priorita*)

# OS: paměť

- Většina paměti nevyužita
  - Zpracování cyklu (zbytek programu)
  - Zpracování konkrétních dat (ostatní neaktivní)
  - Čekání na I/O
- *Virtualizace* paměti
  - Data a programy na disku
  - Do paměti *na žádost*
  - Umožňuje
    - Každý program má „celou“ paměť
    - Program může adresovat více jak rozsah fyzické paměti
- Ochrana paměti

# OS: základní složky

- Procesy a jejich správa
- Paměť a její správa
- Periferie a jejich správa
- Systém souborů
- Ochrana a bezpečnost

# Procesy

- Proces je abstrakce průchodu programem
  - Sekvenční model: program = 1 proces
  - Paralelní model: program > 1 proces
- Proces má *interní stav*, charakterizovaný
  - programovým čítačem (program counter)
  - zásobníkem (volání funkcí a procedur)
  - vlastní paměť pro data

# Typy procesů

- Klasické (heavy-weight) procesy (např. UNIX)
  - Všechna data privátní
  - Sdílen pouze program (read-only)
- *Lehké* (light-weight) procesy či Vlákna (threads)
  - Minimum vlastní paměti
  - Většina dat sdílena

# Procesy detailněji

- Vytvoření procesu
  - `fork()` a jeho varianty
  - *Přesná* kopie původního procesu
  - *Rodič* a *potomek*
  - První proces v OS vytvářen jinak (`init` v Unixu)
- Stav
  - Start/vytvoření, připraven (`ready`), běží (`running`), je blokován (čeká), skončil

# Synchronizace – problém

- Race condition: soupeření v čase
  - Proces P {  
    Load RegistrA, X  
    Load RegistrB, Y  
    Add RegistrA, RegistrB  
    Store RegistrA, X   # X+=Y  
}
- Dvě instance procesu P, používají stejná X a Y
- Nedefinovatelné výsledky
  - Je-li na začátku  $X=Y=1$ , pak na konci může být  $X=2$  nebo  $X=3$

# Synchronizace – řešení

- Kritická sekce
  - Semaforey: celočíselné proměnné (čítače)
  - Monitory: vyšší konstrukty programovacího jazyka  
Je možné semafor implementovat pomocí monitoru a naopak
- Smrtné objekty (deadlock)
- Odstranění sdílených zdrojů: zasílání zpráv
  - Synchronizace na úrovni zasílání a přijímání zpráv
  - Buffery



# Procesy – plánování

- Sdílení (timesharing)
  - časové kvantum
  - přerušení
- Prioritní
  - Statistické
  - Real-time
- Plánovač (scheduler)

# Správa paměti

- Dvě základní operace:
  - alokuj/přiděl paměť (velikost, vrací počáteční adresu)
  - dealokuj/uvolni paměť (velikost a počáteční adresa)
  - Většinou závislé (lze uvolnit jen přesně totéž, co jsme alokovali dříve)
  - Doplnková operace: změň rozsah alokované paměti (reallocate)
- Organizace paměti
- Čištění paměti (garbage collection)

# Správa paměti OS

- Virtualizace paměti – nutno uvolnit fyzickou paměť
- Swapping
  - Celých procesů
  - „Děř“ v paměti
- Stránkování
- Segmentace