B35APO: Architektury počítačů

Lekce 13. Virtuální stroje a virtualizace

Pavel Píša
pisa@fel.cvut.cz

Petr Štěpán
stepan@fel.cvut.cz

11. června, 2023

Obsah

Cíl dnešní přednášky

- Mnohaúrovňová organizace počítače
- Virtualizace

Mnohaúrovňová organizace počítače

- Strojový jazyk počítače
  - množina jednotlivých instrukcí
  - úroveň L1 – abeceda 0,1
  - do ní je potřeba převést program pro vykonání
  - pro návrh aplikace člověkem i generátorem vhodnější jazyky vyšší úrovně L2 + další
- Vykonání programu v L2 na stroji jenž poskytuje L1
  - Kompilace – instrukce v L2 se nahradí posloupností instrukcí v L1
  - Interpretace – program v L1 zpracovává program v L2 jako data (pomalejší)
- Vývoj víceúrovňového stroje
  - první počítače – běžná strojová úroveň - 1.úr.
  - 50-tá léta – Wilkes – mikroprogram. - 2.úr.
  - 60-tá léta – operační systémy - 3.úr.
  - překladače, program. jazyky - 4.úr.
  - uživatelské aplikační programy - 5.úr.
  - problémově orientované jazyky a prostředí

Virtuální počítač

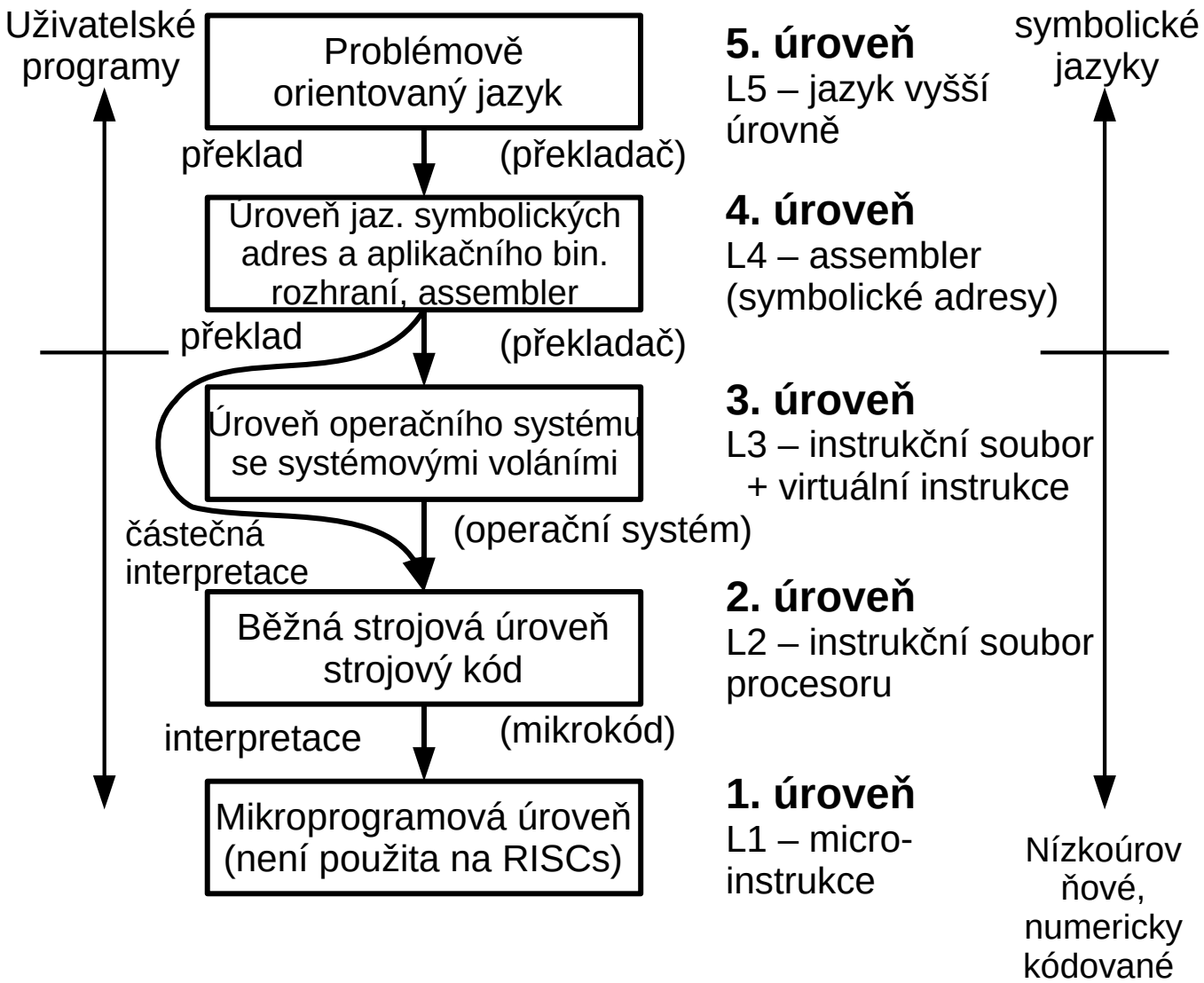
Technické prostředky (HW) a programoví vybavení (SW) jsou logicky ekvivalentní (lze je navzájem nahradit) souboj a splývání RISC a CISC procesorů Na úrovni i zavádíme virtuální počítač M_i s jazykem L_i. Program v L_i je překládán nebo interpretován počítačem M_{i-1} atd.

Pogram v L3 je interpretován programem v M2 nebo M1, nebo je přeložen do L2 nebo L1

Program v L2 je interpretován programem v M1, nebo je přeložen do L1

Program v L1 je skutečně vykonáván technickými prostředky

Současná mnohaúrovňová architektura počítače



Procesy a jejich stavy

proces probíhající program (program – pasivní, proces – aktivní)

stav procesu informace, které při zastavení procesu umožní jeho znovuspuštění

1. program
2. následující instrukce
3. hodnoty proměnných a data
4. stavy a polohy všech V/V (pozice v otevřených souborech atd..)
5. rozložení/obsazení adresního prostoru (paměťový kontext)

předpoklad proces P sám nemění svůj program, pokud ano, tak jsou změněné části součástí stavu, dat

stavový vektor proměnné složky stavu procesu – mění je program, OS nebo HW

$$PROCES = PROGRAM + STAVOVVEKTOR$$

Konvenční strojová úroveň (ISA) (M2)

Definovaná Instrukční sadou (často označovaná architektura procesoru) ISA – Instruction Set Architecture × Mikroarchitektura (Implementace v M1 nebo přímo v logickém návrhu procesoru)

- struktura počítače, procesoru, V/V kanálů, sběrnic, organice a přístupy k paměti, organizace registrů a další
- instrukční soubor, formát dat a uložení v paměti, adresování, zápisníková paměť (registry)

Instrukční formát

- skládá se z polí uložených v jednom nebo více po sobě jdoucí slovech
- operační znak (opcode)
- přímé operandy (immediate), konstanty vložené do instrukce
- adresové části mohou být adresami do paměti nebo odkazy, čísla registrů a ty použity buď přímo nebo jako adresy do paměti (nepřímá adresace)

Obsah

Virtualizace

- Virtualizace skrývá implementaci/vlastnosti nižších vrstev (reality) a předkládá prostředí s požadovanými vlastnostmi
- V počítačové technice rozdělujeme virtualizaci na
  - Čistě aplikační/na úrovni jazyků a kompilovaného kódu (byte-kód) – virtuální stroje, např. JVM, dotNET
  - Emulace a simulace typicky jiné počítačové architektury (také zvaná křížová virtualizace)
  - Nativní virtualizace – izolované prostředí poskytující shodný typ architektury pro nemodifikovaný OS
  - Virtualizace s plnou podporou přímo v HW
  - Částečná virtualizace – typicky jen adresní prostory
  - Paravirtualizace – systém musí být pro běh v nabízeném prostředí upraven
  - Virtualizace na úrovni OS – pouze oddělená uživ. prostředí (kontejnery)

Virtualizace na úrovni operačního systému a aplikací

- I vlákna lze brát jako virtualizaci stavu procesoru a procesy jako virtualizaci celého prostředí včetně oddělení paměti
- Virtuální stroje a interpretery v rámci aplikací (prohlížeč, Python v LibreOffice, atd.)
  - přímá interpretace v textové podobě málo častá, většinou překlad do binární reprezentace (bytecode) a z té často i do nativního kódu (JVM)
- Kontejnery
  - chroot – samostatný pohled na adresářovou strukturu (Mount – mnt)
  - LXC (Linux Containers) – oddělitelnost různých jmenných prostorů jádra (namespace) a prioritizace prostředků (cgroups)
    - Mount (mnt), Process ID (pid), Network (net), Inter-process Communication (ipc), UTS (host+domain), User ID (user), Control group (cgroup) Namespace, Time Namespace
  - Docker – většinou nad LXC
  - systemd-nspawn
  - Solaris Containers (Zones)
  - FreeBSD jail

Virtualizace na úrovni celého stroje

- Hostitelský počítač (Host, Domain DOM 0)
- Hostovaný systém (Guest)
- Procesor pro hostovaný systém
  - pro nativní případ běžný kód/neprivilegované instrukce zpracovány přímo CPU
  - pro křížový případ interpretace instrukcí emulátorem (program v DOM 0), případně akcelerace
- Privilegované instrukce v hostovaném systému
  - způsobí výjimky, které obslouží monitor/hypervizor tak, že je odemuluje
  - CPU má podporu pro HW virtualizaci (AMD-V, Intel VT-x), stínové stránkovací tabulky
- Periferie/zařízení
  - přístupu na IO a paměťově mapované periferie způsobují výjimky a hypervizor odsimuluje jejich činnost
  - hostovaný systém se přizpůsobí tak, aby předal požadavky přímo ve formátu, kterému hypervizor rozumí (ovladače na míru atd.)

Hypervizor

- zajišťuje spouštění a zastavování domén
- monitoruje jejich činnost a ošetřuje výjimky – především emuluje činnost privilegovaných instrukcí
- zajišťuje rozdělení paměti a výpočetního výkonu do hostovaných systémů
- emuluje činnost periferií a předává data do ovladačů fyzických zařízení a sítí na úrovni hostitelského systému
- může být implementovaný
  - v uživatelském prostoru hostitelského systému (QEMU)
  - s využitím podpory v HW a jádře OS (KVM)
  - jako samostatný systém/mikrojádro, který využívá systém v jedné doméně (DOM 0) pro komunikaci s fyzickými zařízeními a z tohoto systému data přeposílá do ostatních (XEN)