Základy počítačových systémů

Elektrické obvody

$\vec{F} = \vec{E}.q$ [N]
$\vec{E}$ je intenzita elektrického pole [V/m]
q je velikost náboje [C]

3Síla působící na náboj v ESP

$$F = \frac{1}{4\pi\varepsilon} \cdot \frac{Q_1 Q_2}{r^2}$$

2Coulombův zákon

$$C = \frac{Q}{U} \quad [\text{F, C, V}]$$

1Kapacita kondenzátoru

$$I = \frac{dQ}{dt} = \frac{U}{R}$$

5Ohmův zákon [A, V, Ω]

$$P = \frac{dA}{dt} = \frac{U.dQ}{dt} = U.I \quad [\text{W, V, A}]$$

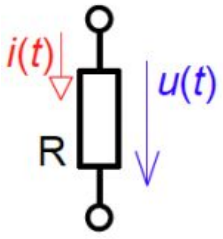
4Výkon - podíl práce a času

- **1. Kirchhoffův zákon:** Součet proudů v uzlu obvodu je roven nule.
- **2. Kirchhoffův zákon:** Součet napětí v uzavřené smyčce je roven nule.

- **obvodové prvky:** - **pasivní**: - nedodávají do obvodu novou energii
  - **dissipativní** - mění el. energii na jinou formu
    - rezistor (→ teplo)
  - **akumulační** - E akumulují ve formě E/M pole
    - kapacitor / kondenzátor (→ EP)
    - induktor / cívka (→ MP)

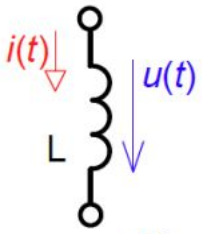
- **aktivní**: = zdroje; dodávají novou energii
  - zdroje napětí x proudu
  - závislé x nezávislé
  - ideální x reálné

Rezistor



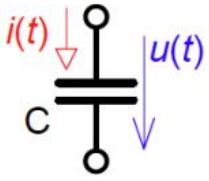
$u = R.i$

Induktor



$u = L.\frac{di}{dt}$

Kapacitor



$i = C.\frac{du}{dt}$

- **polovodičové prvky (součástky):**
    - **dioda:** - usměrňovací
        - vede proud ve směru anoda → katoda
        - v propustném směru úbytek napětí
        - v závěrném směru možný průraz, přesáhne-li max. napětí

    - **LED dioda**: - Light Emission Diode
        - při průchodu proudu v prop. směru vyzařuje světlo
        - nízké závěrné napětí
        - úbytek v propustném směru se liší dle barvy světla

Elektrický signál

- zpráva je přenášena veličinou elektrické povahy (napětí, proud, náboj,...)

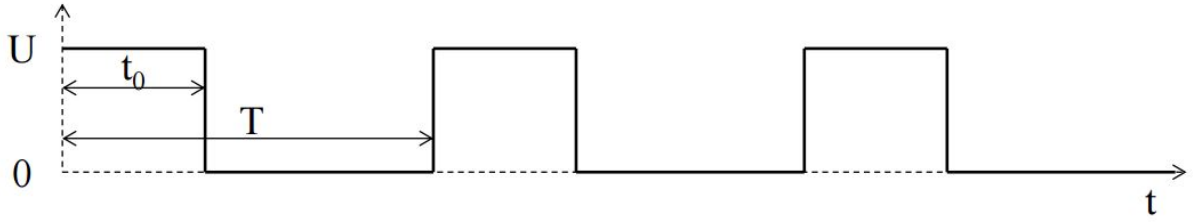
- **stacionární stejnosměrné signály**:
    - okamžitá hodnota je v čase konstantní $u(t) = U$

- **střídavé periodické signály**:
    - okamžitá hodnota je funkcí času

$$u(t) = u(t + k.T) \qquad f = 1/T)$$

- **střídavé neperiodické signály**:
    - pkamžitá hodnota je obecnou funkcí času (např. přechodové děje)

- obdélníkový průběh signálu:



- střída (S) = poměr mezi dobou trvání pulzu (t_0) a periodou (T)

Reprezentace čísel a textu

$$A = \sum_{i=-m}^{i=n} a_i \,.\, z^i \qquad X = (-1)^s \,.\, m \,.\, z^{(e)}$$

- **reprezentace textu:** - prostřednictvím číselných kódů
  - ASCII kód (tabulka) - 7bit, rozšířený 8bit
  - různé kódování CZ znaků (např. Win1250)

  - **UniCode**:
    - vícebajtová (původně 2B) reprezentace znaků
    - dříve 256 x více, ale…
      - nekompatibilita s ASCII, progr. jazyky, OS,...
      - větší texťáky
      - rozsáhlé grafické reprezentace (fonty)
    - dnes přes milion, s kódy 0x0 až 0x10FFFF
    - základní stránka BMP (basic multilingual plane)
    - flexibilní délka kódu, definuje i varianty kódování
    - vždy je třeba definovat endianitu – BE, LE, BOM

  - **UTF32:** (= UCS-4)
    - pouze 4 bajtové znaky

  - **USC-2:** - pouze 2 bajtové znaky, kódování jen BMP

  - **UTF-16:** - 2 bajtové znaky pro BMP
    - 4 bajtové znaky pro zbytek (surrogate pairs)

  - **UTF-8:** - pouze 1 až 4 (původně 6) bajtové znaky
    - poskytuje zpětnou kompatibilitu s ASCII

Paměti

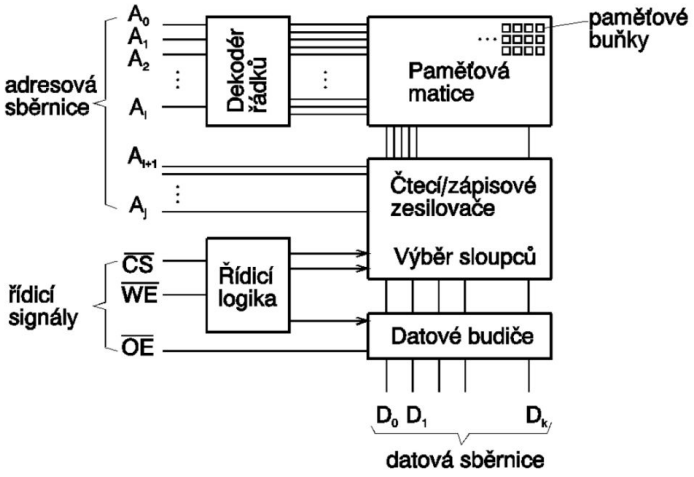
- **dle použití:**
  - hlavní (operační) paměť
  - skrytá paměť (cache)

- **dle volatility uložených informací:**
  - **volatilní** - informace se po vypnutí napájení ztratí
  - **nevolatilní** - informace zůstává

- **Dle způsobu výběru datových položek:**
    - **adresové**
    - **asociativní**
    - **s postupným výběrem**
        - **zásobník – LIFO**
        - **fronta – FIFO**

- **Dle možnosti a způsobu změny uložené informace**
    - **RWM (read/write memory, často označována jako RAM)**
    - **ROM (read only memory)**
    - **PROM (programmable ROM)**
    - **EPROM (Erasable PROM)**
    - **EEPROM (Electrically Erasable PROM)**
    - **FLASH (blokově mazatelné EEPROM)**

Blokové schéma typické paměti



Paměti RAM

- **SRAM – Static Random Access Memory:**
    - volatilní
    - rychlý náhodný přístup
    - kapacita až 32 MB

- **DRAM – Dynamic Random Access Memory**
    - 1 tranzistor na bit
    - Volatilní
    - blokový přístup u moderních DRAM pamětí, náhodný není nejrychlejší
    - Současné DRAM (DDR) - synchronní; využívají se obě hrany (D-Data-Rate)
    - uvnitř pole 2048x2048 1b buněk
    - Adresa ve dvou fázích
        - **RAS** – Row Address
        - **CAS** – Column Address
    - Velká latence mezi počátkem přístupu a prvními daty

- SDRAM, DDR1-4 → snížení napětí, vyšší frekvence, lepší propustnost

Paměti xxROM

- **ROM:**
  - programují se při výrobě → nelze změnit
  - nevolatilní
  - rychlé čtení

- **PROM:**
  - jednorázově elektricky programovatelné
  - nevolatilní

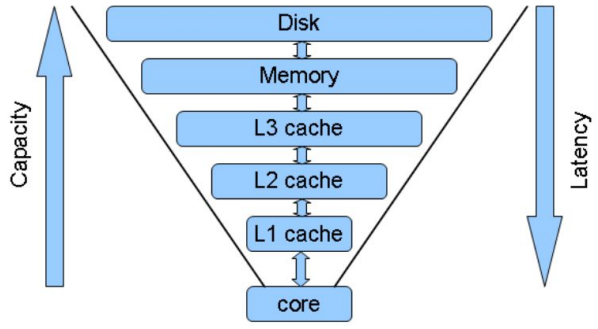
- **EPROM:**
  - elektricky programovatelné
  - nevolatilní
  - mazatelné UV
  - omezený počet cyklů a doby pamatování

- **E^{2}PROM:**
  - elektricky programovatelné i mazatelné (po B)
  - mazání je automatické před dlouhým zápisem
  - omezený počet cyklů
  - nevolatilní
  - až stovky kB
  - parametry ve vestavných systémech (nastavení, výrobní číslo,...)
  - sériové rozhraní

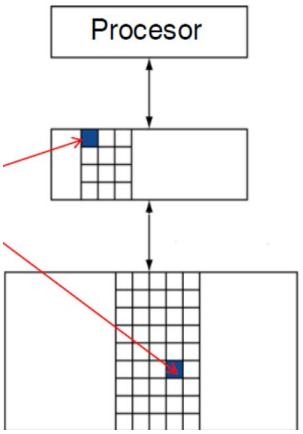
- **Flash EEPROM:**
  - elektricky programovatelné (po B) a mazatelné (celek nebo po blocích)
  - mazání je explicitní
    - **NOR:** - rychlý RA; program může běžet přímo z flash (X RAM)
    - **NAND:** - rychlý blokový přístup
      - paralelní čtení mnoha buněk, ukládání v RAM
      - SSD a paměťovky
      - postupná degradace, chyby v blocích

Paměťová hierarchie

- blíže k procesorovému jádru → rychlejší, nižší kapacita



Cache - skrytá paměť



- kopie části info z hlavní paměti
  - ty části, které procesor právě potřebuje
- data, instrukce...
- u hierarchické struktury:
  - plně inkluzivní - info v cache L_N je i v L_{N-1}
  - neinkluzivní - info jen v jedné úrovni
  - popř. může být a nemusí
- adresace - asociativní paměť; ukládá i adresu dat
- **cache line, cache block**
  - = základní kopírovatelný blok paměti (8 až 1024 B)
  - **Cache hit** - CPU přistupuje k info, která v cache je
  - **Cache miss** - není → musí se nakopírovat

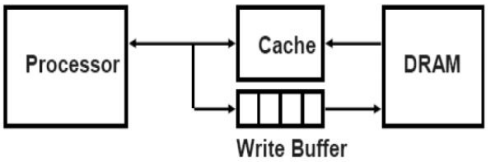
- **stupeň asociativity:**
  - **1 (přímo mapovaná)**
    - velikost cache line - 1 slovo (4 B) / 2 slova (8B)
  - **2 až 16 (praktický limit)**
    - N=2 → 2-cestná; mohou být 2 slova ze stejného setu
    - N=4 → …
  - **plně asociativní**
    - N = počet bloků
    - každá cache line může obsahovat data z jakékoliv adresy
    - pro každý blok (cestu) je vyžadován samostatný komparátor (tag) → HW náročné
    - jen pro malé kapacity

- **výkon cache:**
  - **Hit rate** - podíl úspěšných přístupů
  - **Miss rate** - podíl neúspěšných
  - **Miss penalty** - časová ztráta na načtení chybějících bloků

Průměrná rychlost přístupu do paměti:

– T_{Hit} + MissRate * MissPenalty

- **uvolňování bloků:**
  - je-li třeba načíst blok a cache je plná
  - u PM být plná nemusí
  - **RANDOM** - náhodně → neefektivní
  - **LRU** - Least Recently Used
  - **LFU** - Least Frequently Used)
  - **kombinované**



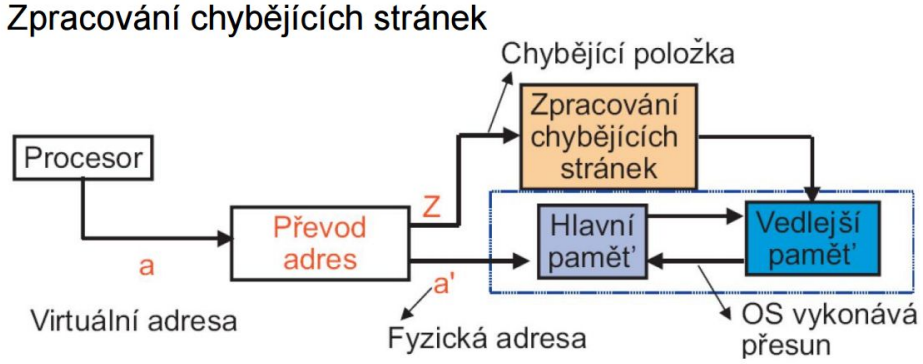
- **zápis dat procesorem do paměti:**
  - konzistence dat - požadavek na shodný obsah stejných adres
  - **write through** - společně s cache i do bufferu, pak asynchr. do paměti
  - **write back** - data do cache (dirty) → zápis do paměti až když hrozí ztráta

- **dirty bit -** pole v obsahu paměti; v cache je jiná hodnota než v paměti

Virtuální paměť

- na PC paralelní procesy → sdílejí fyzickou paměť
  - při překladu nevíme, kde budou → mohou škodit jiným → VP
- procesy pracují s virtuálními adresami (pro rozsahy lze omezit příst. práva)
- může být větší než fyzická operační paměť
- musí se zajistit překlad VA na FA v reálném čase
  - look-up table
  - pro 8B V/F adresní prostor potřebujeme 8 * 3 bity (na překladovou tabulku)

- → **stránkování:**
  - oba adresní prostory rozdělíme na souvislé stránky (4kB,...)
  - překlad probíhá na úrovni celých stránek
  - fyzické stránky mohou být uloženy i na pevném disku
  - souvislé stránky jsou výhodou pro cache
  - v tabulce stránek je i další info (platnost, přístupová práva,...)
  - **page fault** → stránka v paměti chybí
    - nahrána z disku
    - je-li plná paměť, uloží se na disk (LRU)



Mikroprocesory

Architektura počítače - hardware

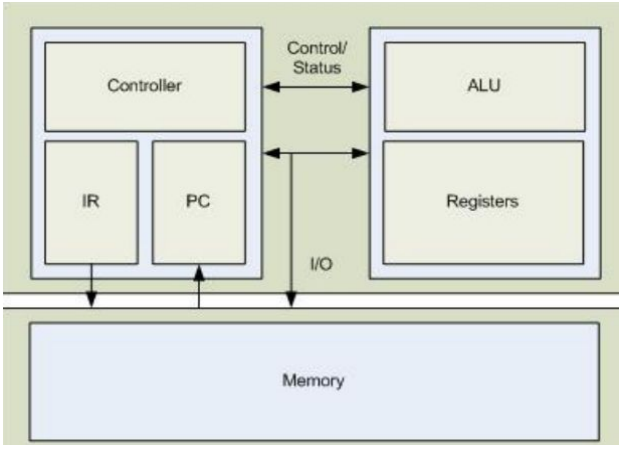
- Harvardská - oddělená paměť kódu a dat
- Von Neumannova - sdílená
- I/O
- programovatelné řadiče

Architektura počítače - software

- **Operační systém**
  - správa zdrojů
  - správa procesů

- API

Architektura procesoru



- **Řadič:** (controller)
  - řídí průběh instrukčního cyklu

- **IR:** - registr instrukce
  - instrukční kód

- **PC:** - čítač instrukcí
  - adresa aktuálně prováděné instrukce

- **ALU:** - aritmeticko-logická jednotka

- **Registry:**
  - dle viditelnosti pro programátora:
    - **viditelné:**
      - **PC – program counter** - adresa prováděné (a násl.) instrukce
      - **ACC – accumulator** - registr pro obecné použití
        - střadač → ukládá výsledky ALO
      - **SP – stack pointer** - ukazatel zásobníku

    - **neviditelné:**
      - **IR – instruction register** - I-kód aktuálně prováděné instrukce
      - **MAR - Memory Adress Register**
      -

Instrukční cyklus procesoru

- cyklicky probíhá: načtení → dekódování → vykonání instrukce
- **ISA** (Instruction Set Architecture) definuje:
  - kódování instrukcí (délka, struktura)
  - počet a umístění operandů
  - datové typy operandů
  - operace (činnost instrukcí)
  - uložení výsledku instrukcí
  - výběr následující instrukce

Střadačově orientovaná ISA

- akumulátor je zdrojem implicitního operandu a impl. místem uložení výsledku
- jediná explicitní adresa operandu
- do 70. let (drahý HW, paměti rychlejší než procesor)

+ krátké I-kódy
+ jednodušší implementace
+ minimální vnitřní stav procesoru – rychlé přepnutí kontextu

- **-** častý přístup do paměti
- **-** limitovaný paralelismus na úrovni instrukcí

Zásobníkově orientovaná ISA (např. JVM)

- implicitní umístění operandů a výsledku instrukce je zásobník
- vrchol zásobníku v procesoru, zbytek simulován v hlavní paměti
    - „neomezená velikost“
    - přesuny řešeny automaticky na úrovni HW při přetečení/podtečení

- **+** jednoduché a rychlé instrukce
- **+** krátké instrukční kódy
- **+** rychlá interpretace nebo emulace – virtuální stroje
- **-** není náhodný přístup k lokálním proměnným
- **-** obtížná paralelizace (sekvenční zásobník)

S univerzálními registry (GPR)

- operandy obvykle explicitní
- uložení výsledku někdy implicitně, někdy explicitně
- prakticky vše od 80. let

- **+** nejrychlejší přístup do registrů
- **+** registry mohou uchovávat lokální proměnné
- **+** méně přístupů do paměti
- **+** náhodný přístup k registrům je možný – paralelizace
- **-** složitější překladač – více možností – optimalizace
- **-** v registrech nelze uchovávat složené datové typy (pole, záznamy)
- **-** nelze adresovat prostřednictvím ukazatelů

RISC architektura

- pevná délka instrukcí
    - delší kód, vyšší nároky na paměť
    - snazší paralelizace
    - adresní mód je součástí IK
    - ARM procesory → instrukce 2 či 4 bajty
- ALU instrukce pracují s registry
- instrukce pro přesun dat mezi registry a pamětí
- jednodušší instrukce i adresní módy
    - rychlejší provádění instrukcí
    - doba vykonání instrukce záleží jen na prováděné operaci

- **+** jednodušší implementace, překladač a paralelizace
- **-** delší programy

CISC architektura

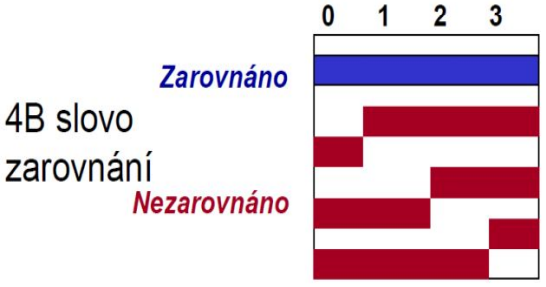
- proměnná délka instrukcí
    - kratší kód
    - každý operand může mít vlastní specifikaci adresního módu

    - složitá implementace
    - x86 architertura – IK 1 až 13 bajtů
- ALU instrukce pracují s registry a/nebo s pamětí
- komplexní instrukce i adresní módy (doba záleží na OP i aresních módech)

+ kratší programy
+ pro načtení operandu/uložení výsledku není vždy třeba samostatná instrukce
- složitější implementace
- obtížná paralelizace na úrovni instrukcí

Práce s daty

- zarovnaná (čte RISC) a nezarovnaná (čte CISC) data
- datový element je zarovnán, pokud je uložen na adrese beze zbytku dělitelné jeho velikostí
- nezarovnaná data šetří paměť, ale složitější přístup CPU



- **skupiny instrukcí:**
    - **přesuny dat:**
        - mezi pamětí a registrem
        - mezi registry
        - mezi paměťovými místy
    - **aritmetické a logické operace:**
        - +, -, x, /
        - and, or, xor...
        - posuny (aritmetické, cyklické)
        - mohou pracovat s registry či pamětí
    - **řízení běhu programu:**
        - skoky (podmíněné, nepodmíněné)
        - volání a návrat z podprogramu
            - šetří paměť
            - call – uložení násl. adresy pro návrat, skok na podprogram
            - ret – skok na návratovou adresu
            - návratová adresa na zásobníku či do registrů

Přerušovací systém procesoru

- přerušení (interrupt) vykonávané sekvence instrukcí a skok na jinou sekvenci
    - obslužný podprogram – ISR
- asynchronně vnější události / synchronně speciální instrukcí
- při vyvolání ISR musí být zapamatována návratová adresa do přerušené sekvence
- při ukončení ISR je uložená adresa nahrána do PC
    - instrukce návratu z přerušení – reti

- **ISR nesmí** modifikovat data přerušené sekv.; využití zdrojů → záloha a obnova

Řadič přerušení

- procesor (jádro) má obvykle jeden přerušovací vstup
- zdrojů přerušení je obvykle více (každý má svůj ISR)
- vyhodnocuje prioritu žádosti o přerušení
- umožňuje maskovat vybrané žádosti a vnoření ISR

Přímý přístup do paměti - DMA

- přenos dat z/do paměti počítače bez účasti procesoru
    - ten může pracovat z cache
    - cílem/zdrojem dat je nějaká periferie procesoru (nebo i paměť)
- řadič DMA soupeří o přístup k paměti s procesorem
    - různé priority aplikací, pozastavování procesoru, kradení cyklů…
- parametry přenosu naplánovány procesorem
    - ODKUD / KAM / JAK / KOLIK

- **aktivace přenosu**:
    - **asynchronní** (hardwarová) - periferie
    - **synchronní** (softwarová) – procesor

- **ukončení přenosu**: signalizováno přerušením

Optimalizace výkonu

Pipeling

- zřetězené zpracování instrukcí
    1. **Instruction fetch** - vyzvednutí instrukce
    2. **Decode** - dekódování instrukce, zároveň se načítají registry
    3. **Execute** - provedení instrukce
    4. **Access** - čtení z paměti
    5. **Writeback** - zápis výsledku do paměti

- **Strukturální hazard:**
    - dvě paralelní-I soupeří o přístup k témuž zdroji

- **Datový hazard:**
    - instrukce čeká na výsledek předchozí instrukce

- **Řídicí hazard:**
    - vyvolán instrukcí skoku

Superskalární procesory

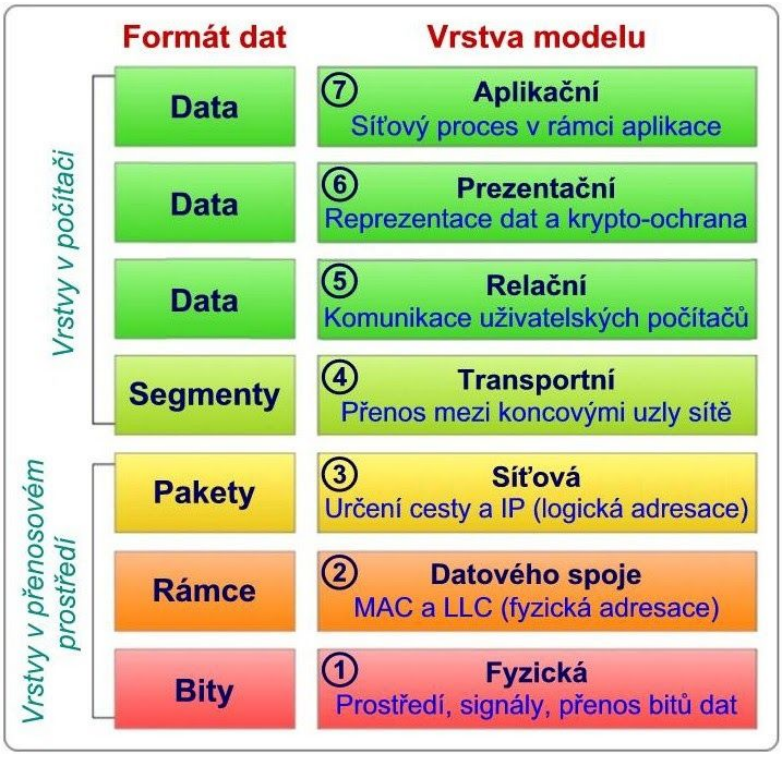
- Paralelní zpracování některých kombinací typů instrukcí
    - např. 1 celočíselná, jedna FP
    - teoreticky 2x rychlejší, ale poměr je jiný

VLIW procesory

- **VLIW – Very Long Instruction Word**
- přesun zodpovědnosti za detekci a řešení hazardů na překladač
- pokud nejde paralelizovat provádění užitečných instrukcí, vkládá se NOP
- nárůst délky programu

Počítačové sítě

- obecný rámec pro návrh protokolů distribuovaných systémů
- sedmivrstvý protokolový zásobník
    - jednotlivé vrstvy poskytují daný typ služeb
    - je definován způsob interakce mezi vrstvami



Fyzická vrstva

- jen přenos bytů
    - přijmi bit, odešli bit
- kódování, konektory,...
- nijak data neinterpretuje
- přenos (a)synchronní, par. / sér.

Spojová (linková) vrstva

- celé bloky dat - rámce (frames)
- přenos pouze v dosahu přímého spojení
- (ne)spolehlivá, (ne)spojovaná
- synchronizace na úrovni rámců
    - ident. začátek a konec
- adresace (příjemce / od. dat)
- přístup ke sdílenému médiu
- zajištění spolehlivosti
- řízení datového toku (x zahlcení)

Síťová vrstva

- přenáší bloky dat → pakety (packets)
- doručení paketů až ke konečnému adresátovi (hledá cestu, zajišťuje směrování)
- Musí si uvědomovat skutečnou topologii celé sítě (obecně)
- poslední vrstva, kterou musí mít přenosová infrastruktura
- nejrozšířenější je protokol IP

Transportní vrstva

- vyšší vrstvy mohou mít jiné pož. na charakter komunik., než jaký nabízejí nižší vrstvy
- obvykle nelze měnit vlastnosti a funkce nižších vrstev
- úkolem transportní vrstvy zajistit potřebné přizpůsobení
- protokoly TV jsou implementovány pouze v koncových účastnících (jinak by poskytovala stejnou službu všem)
- může měnit nespolehlivý charakter přenosu, spojovanost přenosu,...

Relační vrstva

- sestavení, řízení a zrušení relací (hlavně pro spojovanou komunikaci)
- může zajišťovat synchronizaci (více datových toků), šifrování,...

Presentační vrstva

- nižší vrstvy se snaží doručit každý bit přesně tak, jak byl odeslán
- stejná posloupnost bitů může mít pro příjemce odlišný význam
    - kódování znaků, formát čísel, formát struktur,...

  → dělá potřebné konverze

Aplikační vrstva

- obsahuje jádro aplikací, které má smysl standardizovat
    - přenos elektronické pošty, služby pro přístup k objektům na síti,...
- ostatní části aplikací (GUI,...) byly vysunuty nad AV

Datové přenosy:

- **jednosměrný (simplex)**
- **obousměrný střídavý (half-duplex)**
- **obousměrný současný (full-duplex)**

❖ **sériový** - bity (symboly) jsou přenášeny v čase postupně
❖ **paralelní** - přenášeny po skupinách

➢ **spolehlivá služba:**

- nemusí doručit všechny pakety
- může provádět kontrolu správnosti obsahu
- při zjištění chyb se data zahodí (neopakuje se přenos)
- např. Ethernet

➢ **nespolehlivá služba:**

- zaručuje doručení všech paketů bez chyb
- provádí kontrolu správnosti obsahu
- zjištění chyby → opakování přenosu / oprava chyb
- např. USB

NAT (Network Address Translation)

- selektivní překlad vnitřních (např. privátních na veřejné) adres jen pro potřebné
- šetří IP adresy, vyšší bezpečnost
- předpokladem je nízký počet uživatelů z privátní do veřejné sítě

ICMP – Internet Control Message Protocol

- type, code, checksum
- **echo, echo reply**: - např. aplikace ping (dostupnost cílové sítě / uzlu)
- **Destination Unreachable** (network, host, protocol, port,...)
- **Time Exceeded**

ARP - Address Resolution Protocol

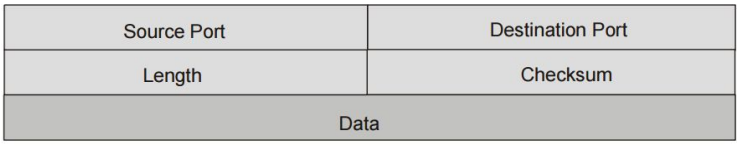
- konverzi IP na MAC adresu (spojová vrstva neumí s IP pracovat)
- uzel sítě si udržuje tabulku s relacemi IP→MAC (ARP cache)
- pokud MAC pro požadovanou IP není nalezena, ARP ji vyžádá (broadcast)
- ARP odpověď obsahuje požadovanou MAC adresu

Porty a Sokety

- port se využívá jako lokální multiplexor pro současně běžící aplikace (procesy)
- data (protokoly) jsou app doručeny: protokol → IP adresa → číslo portu (16bit)
- socket = struktura definující komunikační kanál (viz výše)

UDP - User Datagram Protocol

- neposkytuje spolehlivost, emulaci spojení, řízení datového toku...



– velikost 8 bajtů

TCP - Transmission Control Protocol

- spolehlivá komunikace
- generování potvrzení
- opakování při chybě
- emulace spojení - garantuje pořadí přenášených dat
- řízení datového toku - mechanismus okének
    - **ACK flag** - potvrzovací číslo
    - **PSH flag** - push function – vynucené odeslání dat
    - **RST flag** - reset spojení
    - **SYN flag** - synchronizace pořadových čísel při otevírání spojení
    - **FIN flag** - ukončení spojení
    - **checksum**
    - **...**

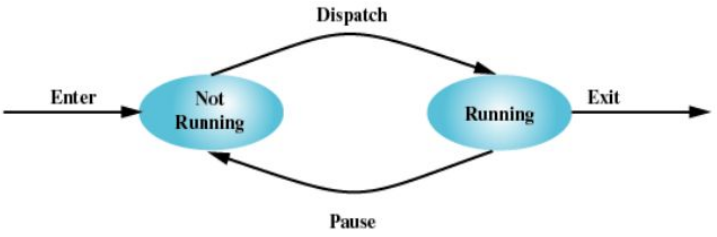
DNS (Domain Name System)

- hierarchický jmenný systém
- distribuovaný, se správou
- často kopíruje organizační struktury

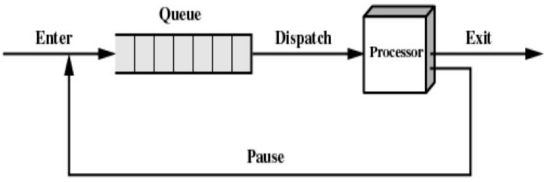
- hostname.sub-level-domain……top-level-domain

Operační systémy - procesy a vlákna

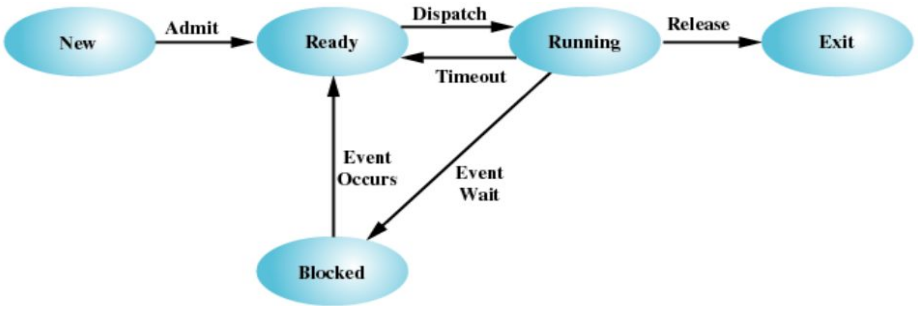
- **proces** = instance programu / entita přidělena procesoru
    - kód programu + data
    - privátní adresový prostor
    - systémové prostředky
    - nejméně jedno vlákno



- **PCB** =(„tabulka popisu procesů“)
    - PID / Stav / Priorita / info o I/O…
    - v multitasking OS se používá při **přepínání kontextu (context switch)**
    - CS řídí **plánovač (scheduller)**



- **2-stavový model: running / not running**
- **5-stavový model: např. blocked (spící)**



Vlákno (thread)

- = abstrakce „toku programu“
- jeden proces obsahuje jedno vlákno = **jednovláknový proces** (Unix / MS-DOS)

- **vícevláknový proces** - spuštění více současných úloh v rámci jednoho procesu
    - po startu procesu běží jediné tzv. primární vlákno
        - PV může vytvářet sekundární
    - sdílejí privátní adresový prostor procesu (glob. prom.)
    - sdílejí systémové prostředky procesu
    - každé vlákno má vlastní kontext procesoru a zásobník

    + není blokován při blokaci jednoho vlákna
    + paměť a systémové prostředky sdíleny

+ context switch vláken je výrazně rychlejší
+ efektivní I/O operace (mohou se překrývat s výpočty)
+ výrazné zrychlení u multiprocesorů → skutečně paralelní

- **implementace vláken (přepínání kontextu vláken):**
  - **na uživatelské úrovni (ULT User-level Thread)**
    - Na úrovni aplikačního procesu (User Mode) musí být implementována speciální knihovna pro správu vláken (thread library).
    + Nezávislost na podpoře jádra
    + Rychlejší přepínání kontextu a vytváření nových vláken
    + Plná kontrola procesu nad správou vláken
    - nutnost dodatečného programování
    - Vlákna jednoho procesu nemohou běžet na více procesorech

  - **na úrovni jádra (KLT Kernel-level Thread):**
    - MS XP, Vista, Linux,...

    + Systémová volání neblokují ostatní vlákna téhož procesu
    + Vlákna jednoho procesu mohou běžet na více procesorech
    + Programy jádra mohou být vícevláknová
    - Správa vláken je nákladnější (časově náročnější) než u ULT

  - **kombinace ULT+KLT:**
    - WAX, MS XP s nadstavbou ThreadFiber
    - Uživatelská vlákna se k systémovým přiřazují automaticky nebo programátor specifikuje uživatelská vlákna jako ULT nebo KLT.

Plánování procesů a vláken (Schedulling)
- rozhodnutí, který proces / vlákno poběží
- typicky optimalizuje vybraný **parametr OS**
  - **doba odezvy**
  - **doba zpracování**
  - **doba čekání**
  - **propustnost (dokončené P/V za jednotku času)**
  - **využití procesoru**
  - **spravedlnost**

- **typy plánování:**
  - **Dlouhodobé (long-term):** při vzniku nového procesu
  - **Střednědobé**: přesunutí procesu ve stavu READY nebo BLOCKED na disk (swapping) nebo použití virtuální paměti
  - **Krátkodobé**: přepínání READY-RUNNING, BLOCKEDRUNNING, reakce na signály, přerušení, systémová volání

- **Off-line plánování:**

- **předpoklady**: všechny procesy od začátku, počet se nemění
- **výsledek**: dávkové zpracování, běh optimální (bez přerušení)
- **problém**: nereálné předpoklady

- **FCFS – First Come First Served**: po pořadě, běží dokud neskončí
- **SJF - Shortest Job First**: kratší úlohy přednostně → menší doba odezvy

- **On-line plánování:**
  - **předpoklady:** nejsou k dispozici od začátku, mění se počet, objevují se neočekávaně, doba běhu není známá
  - **Kritéria plánování:** vázanost na CPU / I/O; charakter pr.; chování v min.; prior.

K plánování procesoru dochází v následujících situacích:

1. pokud některý běžící proces přejde do stavu blokovaný
2. pokud některý proces skončí
3. pokud je běžící proces převeden do stavu připravený
4. pokud je některý proces převeden ze stavu blokovaný do RTOS)

Pokud 1. a 2. => nepreemptivní plánování

Pokud 1. až 4. => preemptivní plánování:

- potřebuje podporu HW (časovač)
- možnost měnit plán na základě nových informací
- context switch - přepnutí kontextu = přepnutí na jiný proces/vlákno
- běžící proces je přepnut do stavu READY

Preemptivní plánování umožňuje přerušení procesu/vlákna ve stavu RUNNING, přesun do stavu READY a přidělení procesoru novému procesu/vláknu!

- **Kooperativní plánování (= nepreemptivní):**

Přepnutí kontextu při jedné z událostí:

1. běžící proces/vlákno přejde do stavu blokovaný
2. běžící proces/vlákno skončí
3. běžící proces/vlákno vytvoří nový proces/vlákno (nevynucuje přeplánování)
4. běžící proces/vlákno se dobrovolně vzdá procesoru

Kooperativní plánování neumožňuje přerušení běžícího procesu/vlákna a přidělení procesoru novému procesu/vláknu!

On-line plánování– základní algoritmy

- **FCFS – First Come First Served: NEPREE**
- **RR – Round Robin: cyklické, stejná priorita vláken, PREE**
- **PP – prioritní plánování - (NE)PREE**
    - **priorita: integer přidělení procesu**
        - **statická / dynamická / kombinovaná**
        - **problémy: Stárnutí, též vyhladovění (starvation)**
                Inverze priorit

Plánování pomocí prioritních tříd:

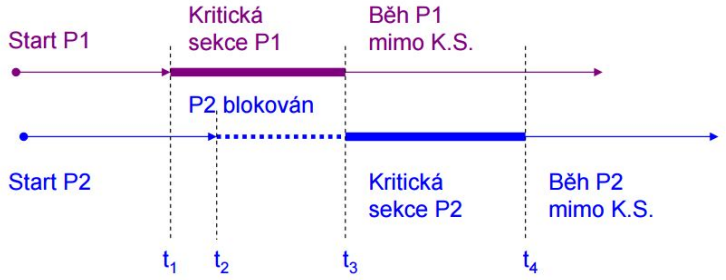
- procesy se stejnou prioritou seskupeny do tříd
- plánovač používá prioritní plánování mezi prioritními třídami
- v rámci každé třídy se používá RR plánování.

Synchronizace

- cílem zabránit konfliktu mezi vlákny (sdílení prostředků)
- pozastavení běhu vláken, dokud nejsou splněny podmínky
- zajistit vykonání progr. sekvence v určitém pořadí

- vlákno / proces se hodí do BLOCKED → dokud se nesplní podmínky → pak se probudí a přesune se do aktivní fronty

- **Kritická sekce:**
    - část programu, kde dochází k přístupu P/V ke sdílenému prostředku
    - P/V vstupující do kritické sekce musí být synchronizovány
    - **podmínky přístupu do KS:**
        - výhradní přístup
        - žádné požadavky na rychlost / počet CPU
        - procesy mimo KS nesmí blokovat ostatní
        - omezené čekání



Semafor: - synchronizační objekt obsahující čítač a frontu čekajících P/V
- hodnota semaforu se pohybuje od 0 do specifikované max. hodnoty
- 3 atomicky prováděné operace:
    - **Init()** → čítač se nastaví na číslo; fronta vyprázdní
    - **Down()** → čítač > 0 → čítač--; jinak se P/V zablokuje
    - **Up()** → jsou-li P/V ve frontě, první se probudí; jinak čítač++

Monitor: - datová struktura + operace pro čtení/změnu stavu
- synchronizační objekt; konstrukce VÚ programovacího jazyka

- **wait()** → zablokuje volající proces v monitoru, monitor uvolní
- **signal()** → odblokuje zabl.; neuvolňuje monitor (zabl. → monitor)