Čísla INTEGER se znaménkem

- Přímý kód:
  - Běžné vyjádření čísel v matematice - znaménko a hodnota (+1234, -5678)
  - V počítači pouze 0 a 1, a proto se používá 0 = + a 1 = -
  - Nevýhody:
    - Při aritmetických operacích se jinak pracuje s bitem znaménka a jinak s bity hodnoty
    - Jsou dvě možné vyjádření nuly
  - V jazyce C jsou celá čísla bez znaménka jako `unsigned int` a se znaménkem `signed int`

- Inverzní kód:
  - Málo výhodná varianta, kdy negujeme číslo, např.: 1 = 0001 a -1 = 1110
  - Liší se od dvojkového kódu jen o jedničku, tzv. horkou jedničku (Hot-One), kterou přičítáme k nejnižšímu řádu
  - Výhodnější je doplňkový kód

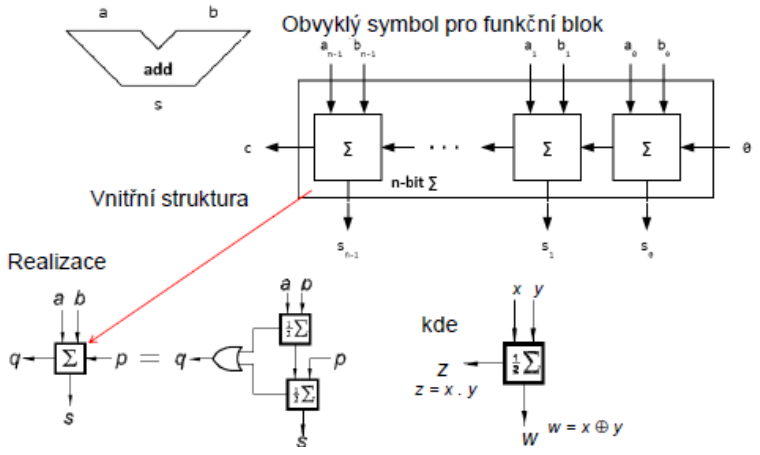
- Aditivní kód (s posunutou nulou):
  - K číslům přičítáme nějakou známou konstantu
  - Toto se používá pro reprezentaci exponentu reálných čísel

- Doplňkový kód (Dvojkový doplněk):
  - Výhodnější proto, že je to úspora HW
  - Postup:
    - Číslo převedeme do inverzního kódu a přičteme jedničku k nejmenšímu bitu
    - Např.: 1 = 0001 → inverzní kód -1 = 1110 → dvojkový kód -1 = 1111
    - Pokud tyto dvě čísla sečteme, dostaneme opět nulu 0000
  - Stejně se může postupovat ve všech soustavách, kde se tomuto říká *doplněk do modulo*

- Sčítání ve dvojkové soustavě:
  - 1 + 0 = 1
  - 0 + 0 = 0
  - 1 + 1 = 0 a 1 se přenáší do vyššího řádu

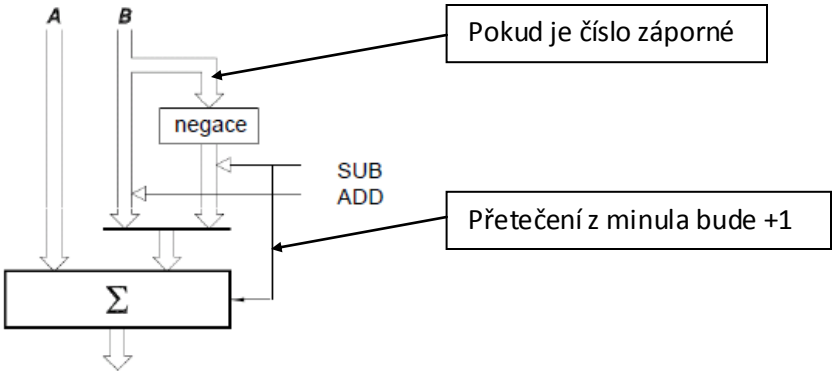
- Odčítání ve dvojkové soustavě:
  - Pokud chceme od nějakého čísla odečíst nějaké číslo, převedeme odečítané číslo pomocí dvojkového doplňku na číslo záporné a poté normálně sčítáme

- Sčítací HW blokově:



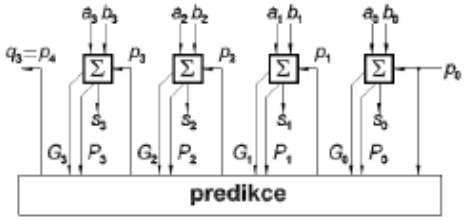
  - Paralelní sčítačka s minimálním zpožděním je prakticky nerealizovatelná – pro 64 bitovou verzi bychom potřebovali 10^{20} hradel!
  - Proto se staví tak, jak vidíme výše s postupným přenosem a ze dvou půlsčítaček

- Sčítací/odčítací HW:



- Přetečení:
  - Situace, kdy výsledek operace není správný, protože se nevešel do dané řádové mřížky
  - Nastává v situaci, kdy znaménko výsledku je jiné, než znaménka operandů, byla li stejná
    - Např.: záporné a záporné, nemůže být ve výsledku kladné

- Paralelní sčítačka je kombinační obvod a je dosti pomalá
  - Sčítání zrychlíme tak, že použijeme sčítačku s predikcí přenosů – CLA



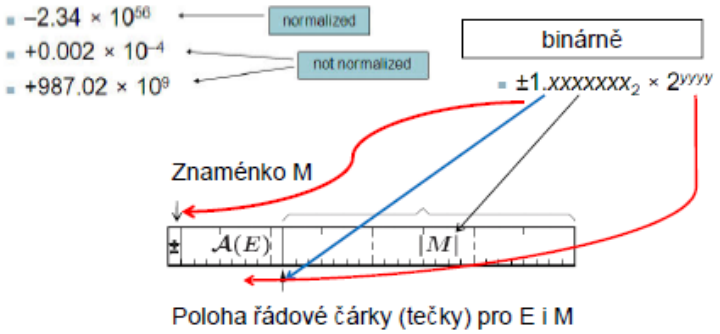
- CLA – Carry Look-Ahead:
  - Dostatečné zrychlení při přijatelném nárůstu ceny
  - 64-bitová verze zvýší cenu HW o necelých 50%, ale rychlost se zvýší 9x

- Aritmetické posuny:
  - Vlevo:
    - Pokud posuneme celou řádovou mřížku o jeden bit vlevo – násobíme dvěma
    - Např.: 2 = 0010 → posun 4 = 0100
    - Zde hrozí přetečení
  - Vpravo:
    - Pokud posuneme celou řádovou mřížku o jeden bit vpravo – dělíme dvěma
    - Např.: 4 = 0100 → posun 2 = 0010
    - Zde hrozí ztráta přesnosti

- Násobení binárních čísel bez znaménka:
  - Příklad pro pochopení:
    - 5 * 3 = 15
    - 0101 * 0011
      0000
      0101
      01010
      01111 = 15

Čísla REAL s řádovou čárkou

- Vědecká, neboli semilogaritmická notace – dvojice EXPONENT(E) a MANTISA(M)
- Notace je normalizovaná – Mantisa vždy začíná nenulovou číslicí
- Normalizováno jako IEEE754 ve verzích s jednoduchou a dvojnásobnou přesností
- V jazyce C se deklaruje jako `float` a `double`
- Provedení:



- Kód mantisy:
  - Přímý kód – znaménko a hodnota

- Kód exponentu:
  - Aditivní kód (s posunutou nulou)

- Příklady některých důležitých hodnot:

Nula

Kladná nula	0 00000000 00000000000000000000000	+0.0
Záporná nula	1 00000000 00000000000000000000000	-0.0

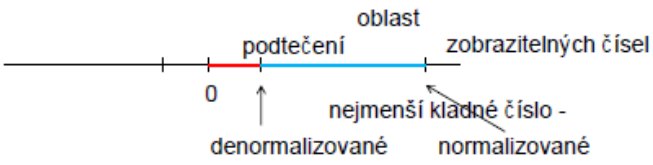
Nekonečno

Kladné nekonečno	0 11111111 00000000000000000000000
Záporné nekonečno	1 11111111 00000000000000000000000

Některé krajní hodnoty

Nejmenší normalizované číslo	* 00000001 00000000000000000000000	$\pm 2^{(1-127)}$
Největší denormalizované číslo	* 00000000 11111111111111111111111	$\pm(1-2^{-23})*2^{-126}$
Nejmenší denormalizované číslo	* 00000000 00000000000000000000001	$\pm 2^{-23}*2^{-126}$

- Denormalizované číslo:
  - Smyslem zavedení denormalizovaných čísel je rozšíření reprezentovatelnosti čísel, která se nacházejí blíže k nule – tedy velmi malých čísel
  - Denormalizovaná čísla mají nulový exponent a i skrytý bit před řádovou čárkou je implicitně nulový
  - Nutnost speciálního ošetření, nulový exponent, nenulová mantisa
    - Jde o situaci, kdy zobrazované číslo není rovno nule, ale neobsahuje hodnoty nejmenšího zobrazitelného normalizovaného čísla



- NaN
  - Samé jedničky v exponentu
  - Mantisa je nenulová
  - V tomto případě není bitový řetězec obrazem žádného čísla (Not-a-Number)

- Zobrazitelná čísla a výjimky

s-bit	Obraz exponentu	m	význam
0	0<e<255	>0	normalizované kladné číslo
1	0<e<255	>0	normalizované záporné číslo
0	0	>0	denormalizované kladné číslo
1	0	>0	denormalizované záporné číslo
0	0	0	kladná nula
1	0	0	záporná nula
0	255	0	kladné nekonečno (overflow)
1	255	0	záporné nekonečno (overflow)
0	255	≠0	NaN – not a number – nečíselná hodnota
1	255	≠0	NaN – not a number – nečíselná hodnota

- Jestliže, je obraz exponentu nenulový je skrytý bit 1, mluvíme o normalizovaných číslech
- Jestliže, je obraz exponentu nulový je skrytý bit 0, mluvíme o denormalizovaných číslech

- Algoritmus pro sčítání v pohyblivé řádové čárce
  - Odečteme exponenty
  - Mantisu čísla s menším exponentem posuneme doprava o počet rozdílu exponentů
  - Sečteme mantisy obou čísel
  - Určíme počet nul mezi řádovou čárkou a první platnou číslicí součtu mantis
  - Posuneme součet doleva o tolik míst, kolik nul bylo nalezeno za řádovou čárkou
  - Zmenšíme původní exponent o počet nalezených nul
  - Zaokrouhlíme

- Algoritmus pro násobení čísel v pohyblivé řádové čárce
  - Exponenty sečteme
  - Mantisy vynásobíme
  - Normalizujeme
  - Zaokrouhlíme

Procesor
- Počítač podle von Neumanna
  - Řadič + ALU – Procesor/mikroprocesor
  - Paměť – Harvardská architektura je variantou s oddělenou pamětí programu a dat
  - Vstup/Výstup – O/I podsystém

- Řadič
  - Součást procesoru, která řídí jeho činnost
  - Sestává ze dvou částí:
    - Datové – což jsou registry a další potřebné obvody
    - Vlastní řídící části
  - Důležité registry řadiče:
    - PC (Program Counter) – programový čítač
    - IR (Instruction Registr) – registr instrukce
    - Další – Pracovní registry, Stack Pointer, Program Status Word, Interrupt Mask
  - Funkce řadiče: V příslušný časový okamžik generovat řídící signály a přijímat signály stavové
  - Řadič – jednotka/sekvenční obvod
    - Výstupy: řídící signály
    - Vstupy: stavové signály
  - Možné realizace řadiče:
    - Obvodový:
      - Řadič s řídícími řetězci
      - Řadič na bázi čítače
    - Mikroprogramovaný:
      - Horizontální – Vertikální – Diagonální

    - Mikroprogramový řadič je vlastně počítačem v počítači
        - Raμl odpovídá PC
        - Řídící paměť odpovídá paměti programu
        - μOZ odpovídá obsahu IR
    - Mikroprogramovaný vs. Klasický řadič
        - Rychlost – klasický je rychlejší
        - Cena – levnější jeve složitější variantě mikroprogramovatelný
        - Flexibilita – mikroprogramovatelný

- RISC (Reduced Instruction Set Computers)
    - Jde o procesory s redukovanou instrukční sadou optimalizovanou na jejich rychlé vykonávání
    - Často se tato architektura označuje jako Load-Store
    - Instrukce jsou stejně dlouhé a vykonávají se v jednom cyklu

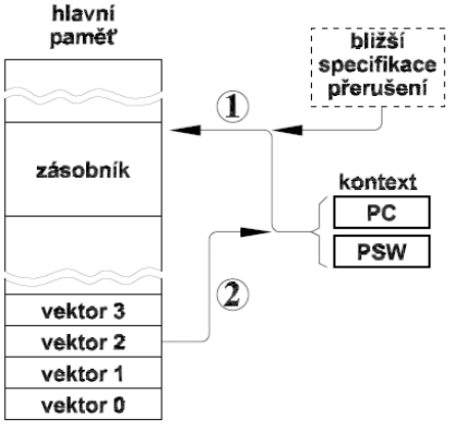
- CISC (Complex Instruction Set Computers)
    - Různě dlouhé, různě dlouho trvající instrukce
    - Motorola, Intel x86

- Základní cyklus počítače
    1) Počáteční nastavení
    2) Čtení instrukce
    3) Dekódování operačního znaku
    4) Provedení operace (včetně vyhodnocení)
    5) Dotaz na možné přerušení – pokud ano, obsluha
    6) Pokud není přerušení zpět na bod 2.

- Instrukce

Instrukce	Syntax	Operace	Význam
Add	add $d, $s, $t	$d = $s + $t;	Add: Sečte dva registry $s + $t a výsledek uloží do registru $d
Addi	addi $t, $s, C	$t = $s + C;	Add immediate: Sečte hodnotu v $s a znaménkově rozšířenou přímou hodnotu, a výsledek uloží do $t
Sub	sub $d,$s,$t	$d = $s - $t	Subtract: Odečte znaménkově obsah registru $t od $s a výsledek uloží do $d
Bne	bne $s, $t, offset	if $s != $t go to PC+4+4*offset; else go to PC+4	Branch on not equal: Skáče pokud si registry $s a $t nejsou rovny
Beq	beq $s, $t, offset	if $s == $t go to PC+4+4*offset; else go to PC+4	Branch on equal: Skáče pokud si registry $s a $t jsou rovny
slt	slt $d,$s,$t	$d = ($s < $t)	Set on less than: Nastavi registr $d, pokud plati podminka $s < $t
jump	j C	PC = (PC ∧ 0xf0000000) ∨ 4*C	Jump: Skáče bezpodmíněčně na návěstí C
lw	lw $t,C($s)	$t = Memory[$s + C]	Load word: Načte slovo z paměti a uloží jej do registru $t
sw	sw $t,C($s)	Memory[$s + C] = $t	Store word: Uloží obsah registru $t do paměti
lui	lui $t,C	$t = C « 16	Load upper immediate: Uloží předanou přímou hodnotu C do horní části registru. Registr je 32-bitový, C je 16-bitová.
la	la $at, LabelAddr	lui $at, LabelAddr[31:16]; ori $at,$at, LabelAddr[15:0]	Load Address: 32-bitové návěstí uloží do registru $at. Jedná se o pseudoinstrukci - tzn. při překladu se rozloží na dílčí instrukce.

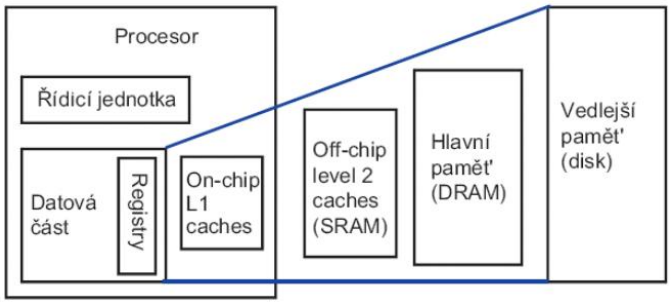
- Přerušení
    - Vnější přerušení
        - Je asynchronní obsluha vnější události
        - Procesor přeruší sekvenční sémantiku vykonávání instrukcí, přejde na obsluhu. Po jejím ukončení se vrátí na místo, kde přerušení detekoval, a pokračuje v činnosti předchozí.
    - Výjimka
        - Je přerušením (obsluhou neplánované události) vyvoláním události uvnitř (v procesoru)
        - Dělení nulou, výpadek stránkování…
    - Obsluha přerušení
        - Uložení adresy místa přerušení a dalších informací na zásobník a nastavení nového kontextu z vektorů přerušení



Paměť

- Základní princip
  - Programy/procesy přistupují v daném okamžiku jen k malé části svého adresového prostoru
  - Časová lokalita
    - Položky, ke kterým se přistupovalo nedávno, budou zapotřebí za chvilku znovu
  - Prostorová lokalita
    - Položky poblíž nedávno používaným budou brzy zapotřebí také
  - Z tohoto plyne, že:
    - Je výhodné uspořádat paměť hierarchicky
    - Všechny potřebné informace uchovávat v sekundární paměti
    - Položky nedávno používané a blízké zkopírujte do (menší) paměti implementované DRAM
    - Položky ještě častěji používané i ty jím blízké zkopírujte do menší a rychlejší SRAM

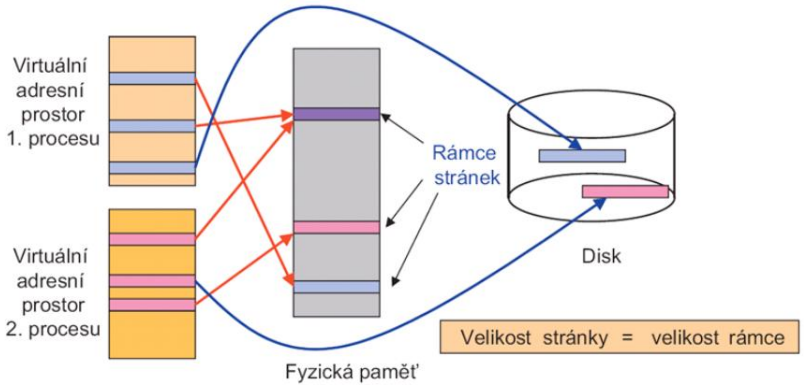
- Paměťová hierarchie



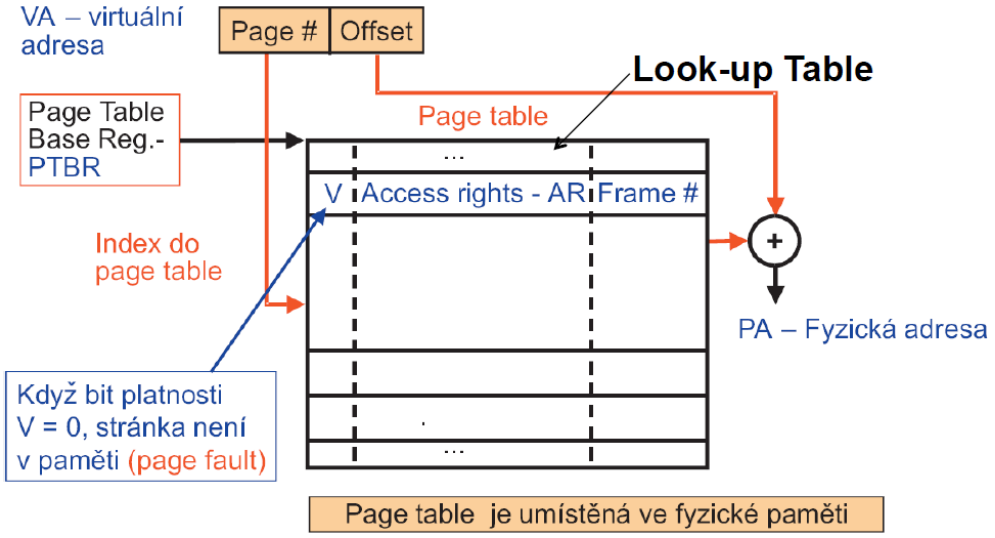
- Jak poté hledanou informaci najdeme:
  - Podle adresy a informace o platnosti
  - Hledat začneme v paměti nejblíže procesoru
  - Požadavky: paměťová konzistence
  - Prostředky:
    - Virtualizace paměti
    - Mechanizmy uvolňování místa a migrace informace mezi paměťovými úrovněmi
    - Hit, miss

- Virtualizace paměti
  - Způsob správy operační paměti umožňující běžícímu procesu zpřístupnění paměťového prostoru, který je uspořádán jinak, nebo je dokonce větší, než je fyzicky připojená operační ať
  - Převod mezi virtuální VA a fyzickou PA adresou může podporovat procesor
  - V současných operačních systémech je virtuální paměť implementována pomocí stránkování paměti spolu se stránkováním na disk, které rozšiřuje OP o prostor na disku

- Stránkování
  - Virtuální prostor tvoří stejně velké stránky (pages), které se přiřazují jednotlivým běžícím procesům
  - Fyzickou paměť tvoří stejně velké rámce (frames)



- Realizace převodu adres
  - Tabulka stránek – page table
  - Jednotka mapování – pages
  - Stránka je také jednotkou přenosu mezi vnější a hlavní pamětí
  - Mapovací funkce se nejčastěji implementuje Look-up Table (vyhledávací tabulkou)



  - Každý proces má svoji tabulku stránek, tedy i svou PTBR, což zajišťuje paměťovou bezpečnost
  - V – Validity Bit, V = 0 Stránka není platná
  - AR – Access Rights, Přístupová práva (Read Only, Read/Write, Executable…)
  - Frame# - číslo rámce

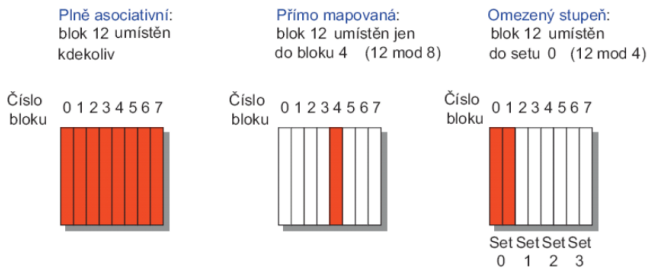
- Cache (Skrytá paměť)
  - Označení pro vyrovnávací paměť používanou ve výpočetní technice
  - Zařazována mezi dva subsystémy s různou rychlostí
  - Účelem je urychlit přístup k často používaným datům

- Terminologie okolo Cache paměti
  - Cache hit – požadovaná hodnota je v cachy
  - Cache miss – opak, není tam
  - Tag – index odpovídající bloku v OP
  - Dirty bit – indikuje, že v cachy je jiná hodnota, než v paměti hlavní
  - Write-back – obsah hlavní paměti se aktualizuje až při uvolňování bloku v SP
  - Write-through – při zápisu do SP se data propisují i do hlavní paměti
  - LRU (Least Recently Used) – informace o posledním použití tohoto bloku
  - LFU (Least Frequently Used) – informace o tom, jak často byl blok používán
  - ARC (Adaptive Repleacement Cache) – výhodným způsobem kombinuje LRU a LFU

- Příklad ke cache

Mějme 8-mi blokovou skrytou paměť. Kam se do ní umístí blok 12 v případě

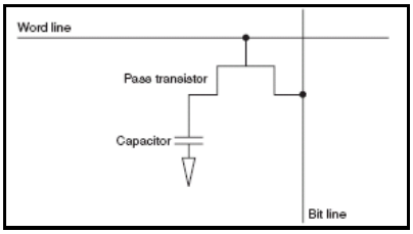
- Plně asociativní.
- Přímo mapované, nebo
- S omezeným stupněm asociativity N=2 (2-cestná, 2-way cache).



- Asociativita je počet bloků, z kolika je cache složena

- Co je to Page Fault
  - Jedná se o výpadek stránky a nastává ve chvílích kdy
    - Fyzická paměť je volná, ale
      - Rámec je prázdný, data jsou ve vedlejší paměti (disku)
      - Požadovaná stránka se nějak načítá do prázdného rámce
    - Paměti je nedostatek
      - Pomocí LRU najdeme rámec, který můžeme uvolnit
      - Máme-li nastaven Dirty bit, zapíšeme stránku do vedlejší paměti (disk)
      - Aktualizuje se tabulka stránek procesu

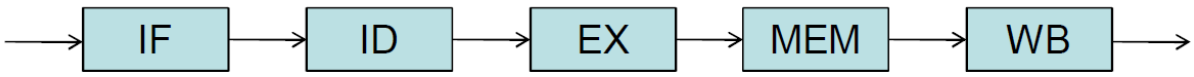
- Realizace paměti – paměťové čipy
  - Klopné obvody
    - RS, D latch – úrovňový klopný obvod, D flip-flop – hradlový klopný obvod
    - Dynamická paměťová buňka



- FLASH paměti
  - Kombinace vlastností EPROM, DRAM, ROM
  - Data jsou ukládána v poli tranzistorů
  - Lze programovat každý blok samostatně
  - Typ RAM (s náhodným přístupem)
  - Přístupová doba 50 – 110ns
  - Řádově stovky tisíc přemazání
  - Uchová informaci několik let
  - Využití: USB flash, SSD disky, paměťové karty
  - Paměťová buňka
    - Modifikovaný MOSFET rozšířený o elektricky izolované plovoucí hradlo

Zřetězené vykonávání instrukcí

- Předpokládejme, že vykonání instrukce můžeme rozdělit do 5 stupňů:



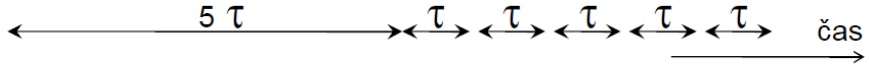
IF – Instruction Fetch, ID – Instruction decode (and Operand Fetch),
EX – Execute, MEM – Memory Access, WB – Write Back

a dále $\tau = \max \{ \tau_i \}_{i=1}^{k}$, kde τ_i je čas šíření (*propagation delay*) *v i*-tém stupni

  - IF – posílání PC do paměti a vybrání aktuální instrukce. Aktualizace PC = PC + 4
  - ID – dekódování instrukce a načtení registrů specifikovaných v instrukci, provedení testu na rovnost registrů (kvůli možnému větvení), znaménkové rozšíření offsetu, výpočet cílové adresy pro případné větvení
  - EX – operace ALU
  - MEM – v případě instrukce load/store – čtení/zápis do paměti
  - WB – v případě instrukcí typu register-register nebo instrukce load – zápis výsledku do RF

- Paralelizmus na úrovni instrukcí – zřetězení

IF	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
ID		I1	I2	I3	I4	I5	I6	I7	I8	I9
EX			I1	I2	I3	I4	I5	I6	I7	I8
MEM				I1	I2	I3	I4	I5	I6	I7
ST					I1	I2	I3	I4	I5	I6
	1	2	3	4	5	6	7	8	9	10

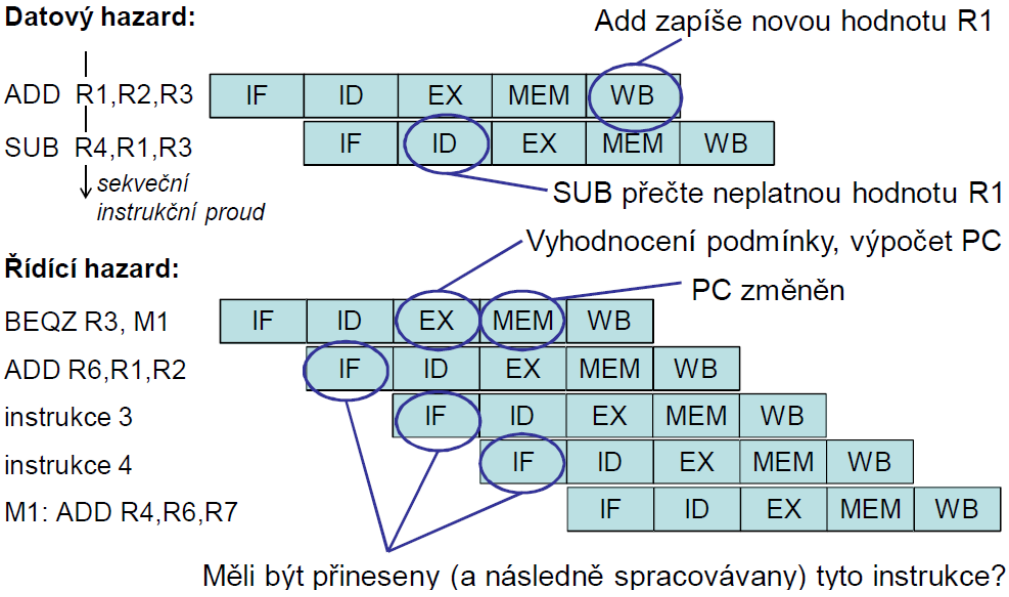


Čas vykonání *n* instrukcí *k*-stupňové pipeline:

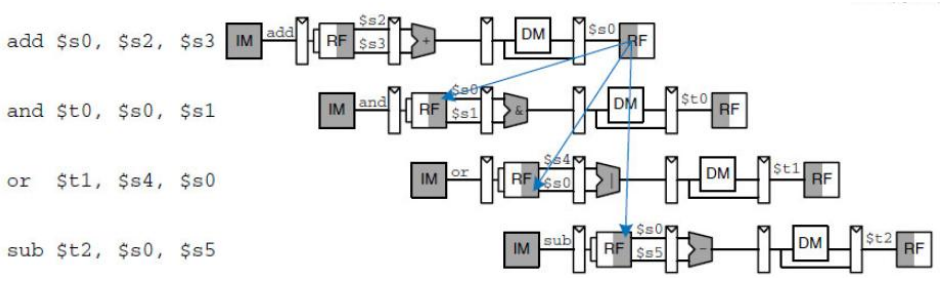
$$T_k = k.\tau + (n-1)\,\tau$$

Zrychlení: $S_k = \frac{T_1}{T_k} = \frac{nk\tau}{k\tau + (n-1)\tau}$ $\lim_{n\to\infty} S_k = k$

- Neredukuje čas vykonání instrukce, právě naopak…
- Hazardy:
  - Strukturální (řešeny duplikací)
  - Datové (důsledek datových závislostí)
  - Řídící (instrukce měnící PC)
- Hazardy způsobují pozastavení vykonávání (stall) nebo vyprázdnění pipeline

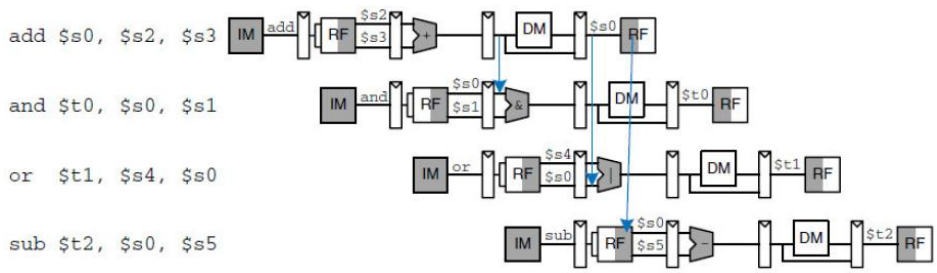


- Vznik datových hazardů



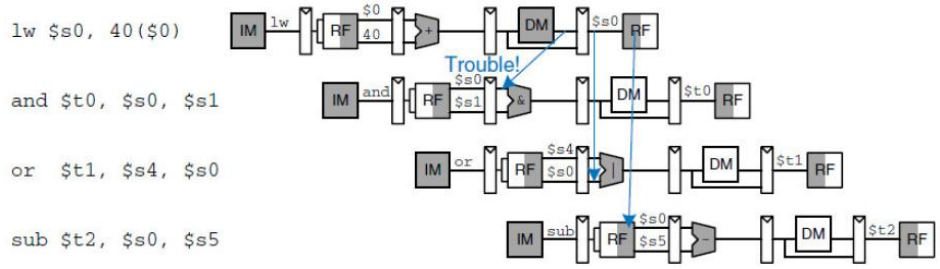
- Register File – přístup v dvou fázích (Decode, WritBack) – zápis v první polovině cyklu, zápis ve druhé..

- Řešení datových hazardů přeposíláním (forwarding)

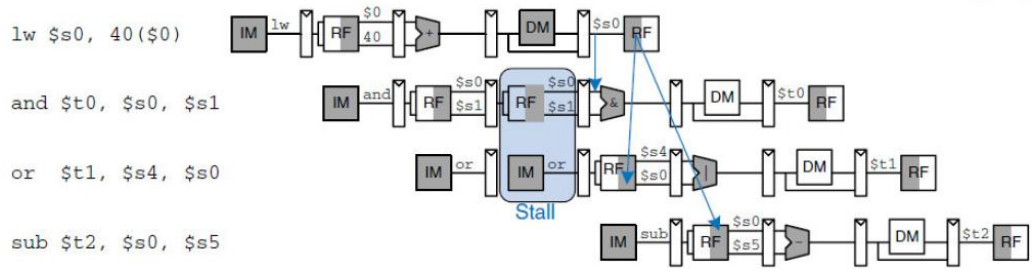


  - Pokud výsledek vzniká dřív, než jej následující instrukce skutečně potřebují, je možné tento hazard řešit přeposílání
  - Nastává, když se použité zdrojové registry instrukce ve stupni EX shodují s cílovým registrem ve stupni MEM nebo WB
  - Proto musejí být čísla těchto registrů z těchto stupňů posílána do Hazard Unit
  - Taktéž, zda cílový registr bude skutečně použit k zápisu – RegWrite z MEM a WB

- Řešení datových hazardů pozastavením (stall)



  - Pokud následující instrukce potřebují výsledek dříve, než skutečně vzniká, je možné tento hazard řešit pozastavením (stall)
  - Pozastavení pipeline je prostředkem řešení hazardu, nezvyšuje však propustnost systému
  - Stupně pipeline předcházejí stupni, kde hazard vzniká jsou pozastaveny do doby, než jsou k dispozici výsledky požadované následujícími instrukcemi – ty jsou pak přeposílány (forwarding)



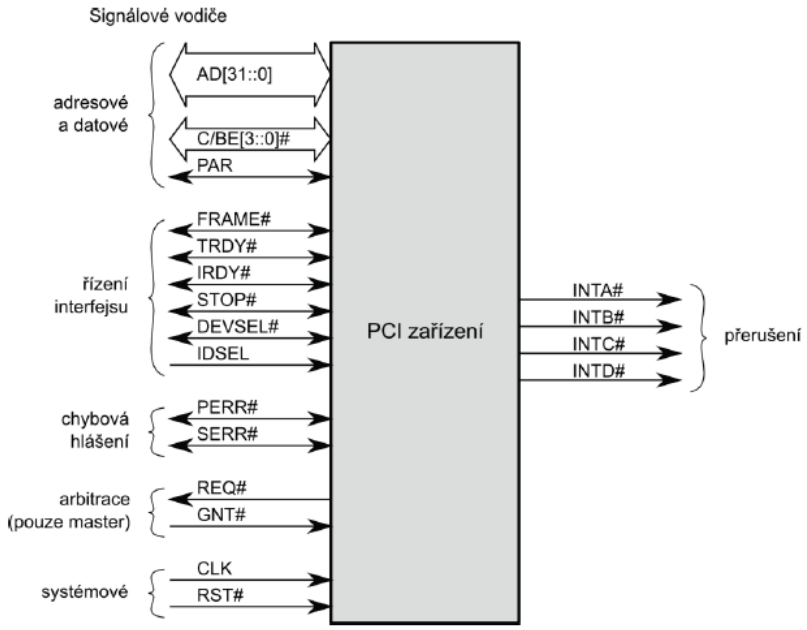
  - Pozastavení se dosáhne podržení hodnoty mezistupňových registrů
  - Výsledky z kolizního stupně se musejí „ztratit“ - řídící signály umožňující změnit stav procesoru se nulují

- Zřetězení – Výkon
  - $IPS = IC / T = IPC_{str} * f_{CLK}$
  - Příklad:
    - Tc = 300 ns → 3 333 kHz
    - Zanedbejme plnění pipeline, všechna pozastavení a vyprázdnění pipeline → IPC = 1
    - IPS = 1 * 3 333e3 = 3 333 000 instrukcí za sekundu

O/I Podsystém

- Propojení jednotlivých částí počítače mezi sebou → sběrnice (BUS)
- Interface
  - Společná komunikační část sdílená dvěma systémy, zařízeními nebo programy
  - Synchronizace přenosu údajů interfacem, možnosti:
    - Asynchronní
    - Synchronní
      - Pseudosynchronní
    - Izochronní
  - Izochronní systém
    - Údaje se přenášejí s konstantní průměrnou rychlostí
    - Typické pro multimediální zařízení
    - Řešení: používá se několik souběžných přenosových kanálů, jedním kanálem se obsluhuje několik periférií
  - Synchronní
    - Přenos konstantní okamžitou rychlostí
  - Asynchronní
    - Zařízení jsou z hlediska časování nezávislá

- PCI
  - Každé sběrnicové transakce se zúčastní dvě zařízení
    - Initiator (master) X Tagret (slave)
  - Multimaster – když je více masteru
  - Arbitr sběrnice – rozhoduje o tom, kdo bude aktuálním masterem
  - Pro načasování sběrnicového cyklu je vztažným okamžikem náběžná hrana hodin
  - Sběrnicový cyklus má:
    - Adresovou fázi
    - Datovou fázi
  - Synchronizace přenosu = pseudosynchronní
  - PCI signály:

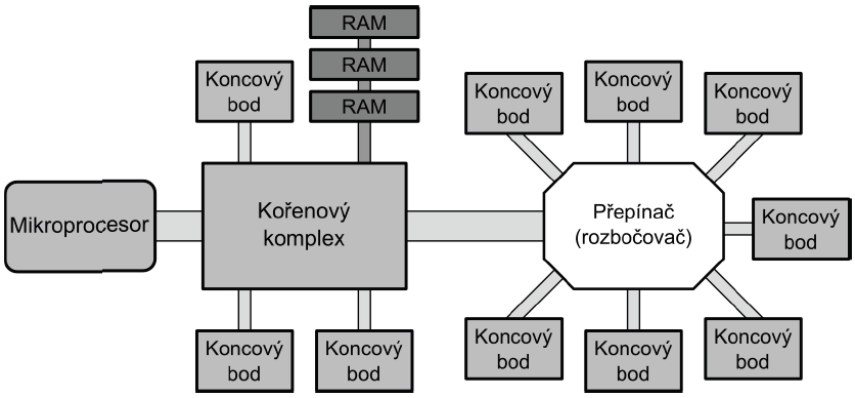


- Význam bitů C/BE[3..0]#

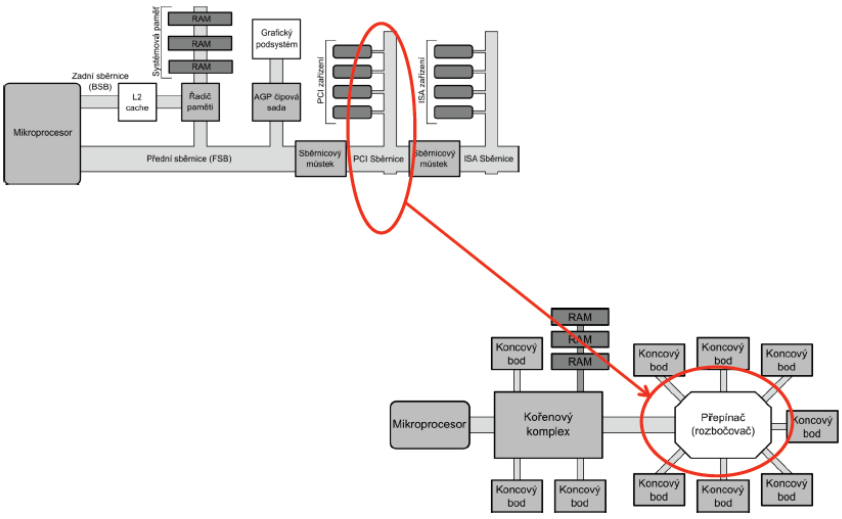
C/BE[3::0]#	Typ prikazu
0000	Potvrzení prerušení (Interrupt Acknowledge)
0001	Speciální cyklus (Special Cycle)
0010	Ctení z portu (I/O Read)
0011	Zápis na port (I/O Write)
0100	Rezervováno (Reserved)
0101	Rezervováno (Reserved)
0110	Ctení z pameti (Memory Read)
0111	Zápis do pameti (Memory Write)
1000	Rezervováno (Reserved)
1001	Rezervováno (Reserved)
1010	Konfiguracní ctení (Configuration Read)
1011	Konfiguracní zápis (Configuration Write)
1100	Memory Read Multiple
1101	Dual Address Cycle
1110	Memory Read Line
1111	Memory Write and Invalidate

- Problém
  - Procesor musí uložit informace o právě probíhajícím programu a to stojí nějaký čas
  - Řadič přerušení vnutí procesoru vektor přerušení, ze kterého se odvodí startovací adresa obsluhy a i to něco stojí

- PCIe
  - Architektura dnešního PC

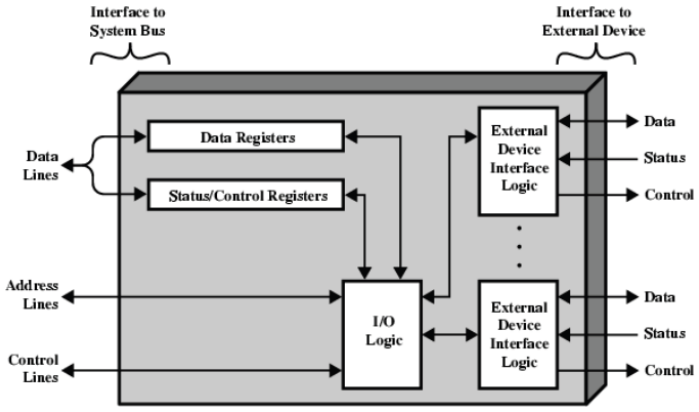


  - U PCIe nahradil sdílenou sběrnici sdílený přepínač

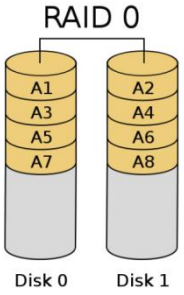


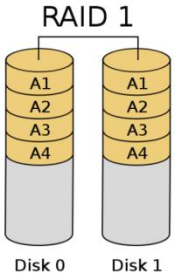
- Přenos údajů, PCIe linka, PCIe proud
  - Spojení mezi PCIe zařízením a přepínačem se nazývá linka (link) s nesymetrickým způsobem přenosu signálu (rozdílové vodiče)
  - Každá linka se skládá z jednoho nebo více proudů (lane)
  - Proud tvoří 2 páry vodičů – 1 přijímací a 1 vysílací
  - Každý přenese jeden bit za tak v obou směrech (plně duplexní přenos)

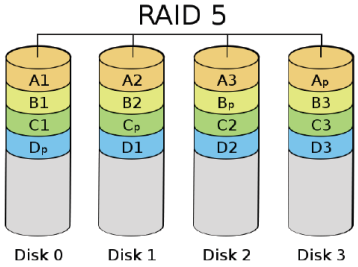
- Adaptér
  - Řadič V/V zařízení
  - Skládá se ze:
    - Sběrnice (v našem případě PCI, nebo PCIe) rozhraní
    - Vlastního řadiče
  - Příklad – řadič disku



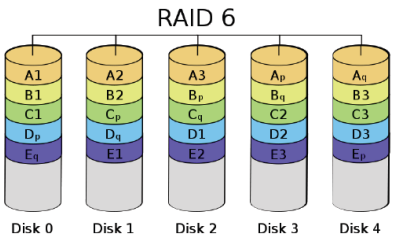
  - Nejdůležitější periférií je disk, tak ho musíme zrychlit
    - RAID 0
      - Pro zvýšení výkonu systému pevných disků tzv. „stripping“ (proužkování)
    - RAID 1
      - Pro zvýšení spolehlivosti uložených dat tzv. „mirroring“
      - Nezrychluje, ale zvyšuje spolehlivost
    - RAID 10
      - Kombinace dvou výše popsaných
      - Vytvoří se RAID 0 a ten se pak zrcadlí na RAID 1. Výsledkem jsou dva RAID 0 obsahující identická data
      - RAID 10 zvyšuje jak výkon, tak spolehlivost, ale musíme použít nejméně čtyři disky, nejlépe se stejnými parametry
    - RAID 5
      - Ukládá paritní informace, nikoli však jen na jeden vyhrazený disk
      - V degradovaném režimu se musejí data uložená na vadném disku odvodit z dat zbývajících disků a parity
      - Zrychluje čtení, zpomaluje zápis







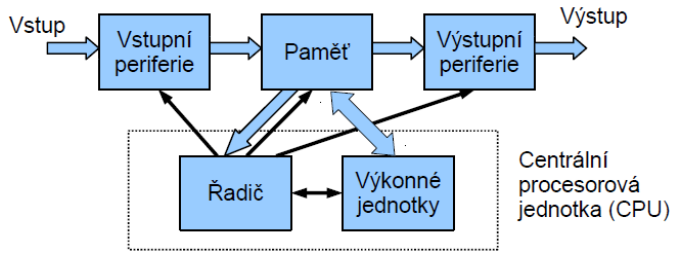
- RAID 6
  - Obdoba RAID 5, používá dva paritní disky s různě vypočtenou paritou
  - Odolný proti výpadkům dvou disků
  - Rychlost čtení jako RAID 5, zápis ještě pomalejší



- Brána (port)
  - Buňka v adresovatelném prostoru V/V zařízení nebo paměti
  - Stavební prvek interface
  - Obecně je to 8/16/32 b buňka, registr, ale někdy se bez klopných obvodů obejdeme

Technické a organizační prostředky (vnější události, výjimky, reálný čas)

- Model průchodu dat počítačem

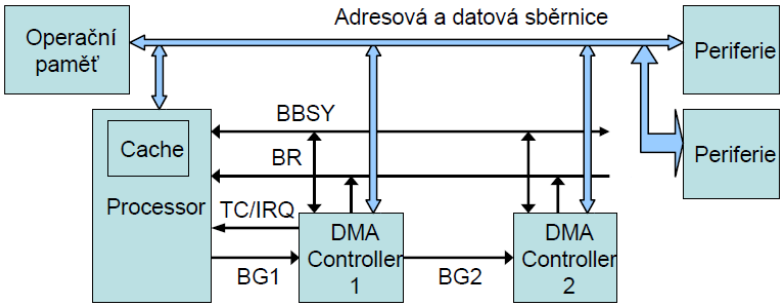


- Různé způsoby řízení zpracování dat
  - Dávkové zpracování – řídí si přístup dat tak, jak je zpracovává
  - Interaktivní – řízené událostmi ať od uživatele, nebo podle příchodu vnějších požadavků nebo událostí)
  - Řízení v reálném čase, kdy po příliš pomalém vyhodnocení/dosažení i správných výsledků nejsou výsledky použitelné

- Vstupně výstupní (O/I) podsystém
  - Pouze vstupní periférie
    - Klávesnice, myš, kamera…
    - Logické vstupy, převodníky fyzických veličin většinou nejdřív na analogový elektrický signál a pak A/D převodníkem na číslo na vstupní bráně
  - Pouze výstupní periférie
    - Výstup obrazu, zvuku…
    - Výstup s hmatatelnou fyzikální podobou, přes převodník D/A
  - Obousměrné
    - Disk, komunikační rozhraní
    - I většina výše uvedených jednosměrných periférii potřebuje obousměrnou komunikaci

- Metody přenosu dat z/na periférii
  - Programový kanál (polling)
    - Procesor ve smyčce čeká na informace o dostupnosti dat, nebo na volné místo ve výstupním registru
    - Nejméně vhodné řešení, procesor čeká na data ve smyčce
    - I pokud opravdu není možné nalézt, pro CPU jinou činnost, tak je toto řešení minimálně neekologické
  - Programový kanál s přerušením (interrupt driven)
    - Na vstupu je inicializační kód programu/OS pouze konfiguruje periférii
    - Při příchodu dat dojde k asynchronní události (přerušení). V její obsluze je blok dat z periférie vyčten
    - Procesor zapíše blok do výstupní periférie a provedením přenosu je potvrzeno přerušením
    - Periférie se sama stará o oznámení přítomnosti nových dat – vyvolá výjimku/přerušení
    - Není o moc lepší než předchozí řešení

- Přímý přístup do paměti (DMA)
  - Vlastní přenos dat realizuje speciální jednotka



  - Počítačový systém je doplněn o specializované jednotky určené pro přenos dat
  - Velké datové přenosy zbytečně nevytěsňují data z CACHE
  - Program/OS naplánuje parametry přenosu
  - Procesor nastaví adresy do DMA řadiče, ten po ukončení operace vyvolá přerušení
- Autonomní kanál (inteligentní periférie, bus master DMA)
  - Inteligence se přesouvá do periférie
  - Periférie je doplněna vlastním řadičem
    - Konečný automat
    - Vstupně/výstupní procesor (IOP)
  - Průběh přenosu:
    - Nadřazený/centrální procesor vloží sekvenci datových bloků do paměti
    - Nakonfiguruje nebo přímo naprogramuje řadič periférie a ta provede sekvenci přenosů z/do hlavní paměti
    - Po úplném nebo částečném dokončení činnosti, informuje CPU vyžádáním přerušení
  - Procesor/OS zpracuje přerušení a přeplánuje na úlohu, která data čeká

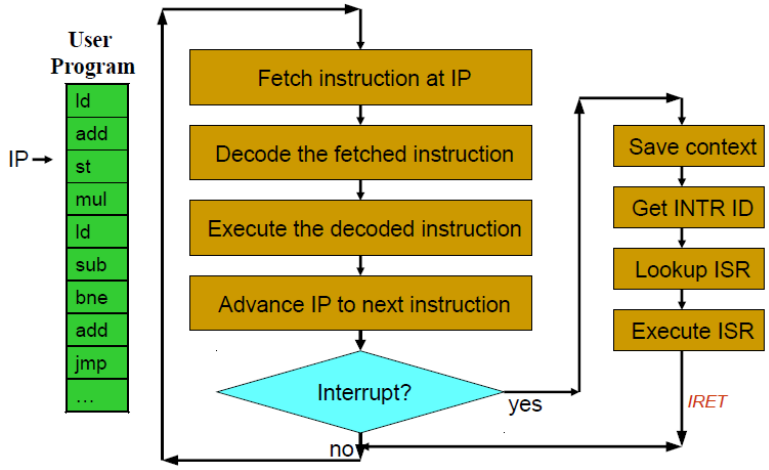
- Paměťově mapované periférie a konzistence dat
  - V/V operace a CPU
    - Pro oblasti adresního prostoru, kam jsou mapované periférie, musí být zakázané ukládání a čtení dat z cache
    - Zřetězené zpracování instrukcí samotné nevadí
    - Přeposílání, vynechávání zbytečných čtení a zápisů a případné vykonávání instrukcí mimo pořadí vadí
    - Nutnost zavést specializované instrukce, které zajistí dokončení všech přenosů před další operací
      - MIPS IV – sync (lx a sx se dokončí před dalším lx)
      - PwerPC – eieio a sync
    - Stejně tak je nutné informovat optimalizaci v kompilátoru (volatile,…)

- DMA a konzistence dat
  - Při přenosu dat do periférie je nutné počkat, až se data dostanou do hlavní paměti (zpožděný zápis)
  - Před načtením dat periferie je nutné vyprázdnit část CACHE
  - CPU/jednotku správy paměti doplnit o instrukce/prostředky pro správu řádek cache

- Výjimky a přerušení
  - Výjimky – ošetření zvláštních situací, které brání dalšímu vykonávání instrukcí (exceptions)
    - Pro případ procesoru MIPS se jedná především o
      - Matematické přetečení
      - Načtena nedefinovaná instrukce
      - Systémové volání (syscall)
    - Nedostupnost dat, nebo selhání zápisu
      - Chybná adresa, nebo stránka označená za nevalidní (invalid)
      - Chyba hlášená sběrnicí (parite, ECC, vypršení limitu na potvrzení)
  - Asynchronní vnější události – přerušení (interrupts)
    - Maskovatelná – lze je zakázat v PSW, případně zadat priorotu

- Nemaskovatelná – většinou ošetření HW chyb (WatchDog)

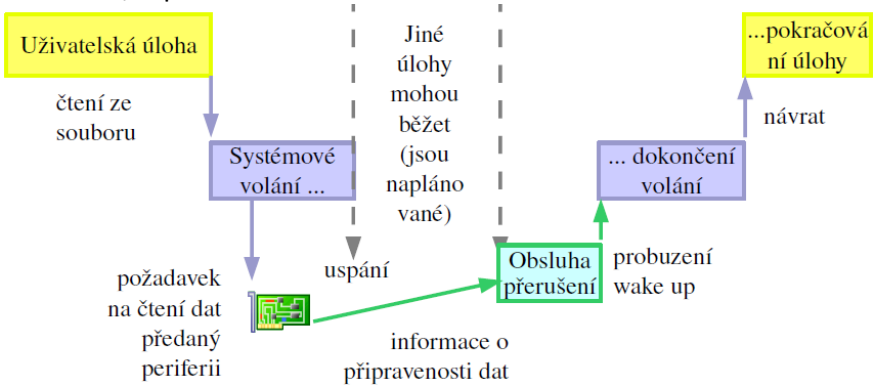
- Průběh zpracování výjimky, nebo přerušení
  - Výjimka je přijata vždy, přerušení jen pokud je povolené
  - Dojde k uložení PSW včetně PC
  - Čítač instrukcí je nastavený na adresu odpovídající typu výjimky případně i číslu zdroje přerušení
  - Z této adresy je vykonávaný kód obslužné rutiny
  - Ta uvolní/uloží další registry na zásobník a obslouží periferii/ provede nahrání stránky, vyhlásí neopravitelnou chybu úlohy nebo celého systému, atd…
  - Obnoví registry
  - Příslušnou instrukcí, provede návrat CPU do odchozího stavu



- Určení zdroje výjimky/přerušení
  - Softwarové vyhledávání (polled exception handing)
    - Veškerá přerušení a výjimky spouštějí rutinu od stejné adresy, např.: MIPS 0x00000004
    - Rutina zjistí důvod ze stavového registru (MIPS – cause registr)
  - Vektorová obsluha přerušení
    - Již hardware CPU zjistí příčinu/číslo zdroje
    - V paměti se nachází na pevné/řídicím registrem specifikované VBR adrese tabulka vektorů přerušení
    - Procesor převede číslo zdroje na index do tabulky
    - Z daného indexu načte slovo a vloží ho do PC
  - Nevektorová obsluha více pevně určených adres podle priorit
  - Často jsou přístupy kombinované

- Asynchronní a synchronní výjimky/přerušení
  - Vnější přerušení jsou obecně asynchronní, nejsou vázané na instrukci
    - RESET – základní nulování a nastavení
    - NMI – nemaskované přerušení
    - INT – maskované přerušení
  - Naopak synchronní výjimka a přerušení jsou svázané s instrukcí
    - Aritmetické přetečení, dělení nulou
    - TRAP – krokovací režim, přerušení na konci každé instrukce

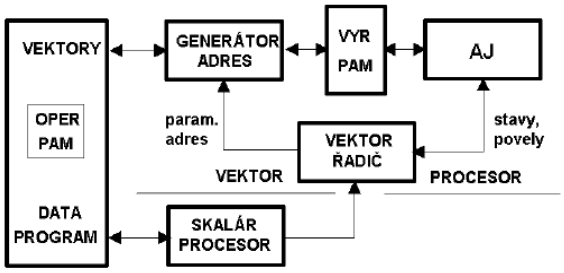
- Obsluha O/I operací na úrovni OS



Sítě procesorů

- Struktury procesorů
  - Klasický – SISD – jednoduchý tok dat + instrukcí
  - Procesorové pole – SIMD – jednoduchý tok instrukcí a vícenásobný tok dat
  - Pipeline – MISD – vícenásobný tok instrukcí, jednoduchý tok dat
  - Multiprocesorové – MIMD – vícenásobný tok instrukcí a dat

- Vektorové procesy
  - Mnohokrát tatáž operace s různými operandy (pole – vektory)
  - Odstraňují řízení cyklu, výpočet indexu, výběr dat podle indexu
  - Realizují se jako paralelní nebo sériové



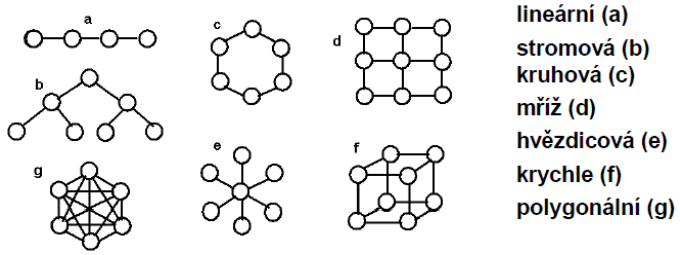
  - Skalární procesor – klasický procesor – všechny nevektorové operace, příprava práce pro vektorový
  - Vektorový procesor – provádí vektorové instrukce (proudově pracující AJ)

- RISC a CISC procesory
  - RISC procesory
    - Jednoduché instrukce
    - Pevný formát dat a délka
    - Stejná doba vykonání
    - Obvodový řadič
    - Velký počet registrů
  - Procesory podporující pipeline
    - Čtení instrukce do fronty přednahr.
    - Překrývání výkonu instrukcí (proudové zpracování po sekcích)
  - Problémy
    - Nepodmíněný skok ve frontě
    - Podm. Skok má bit predikce
    - Operand pro následující instuk. – zpoždění
  - Superskalární režim
    - Řešení několika instrukcí najednou
    - Dáno počtem výkonných jednotek procesoru (pracují paralelně)

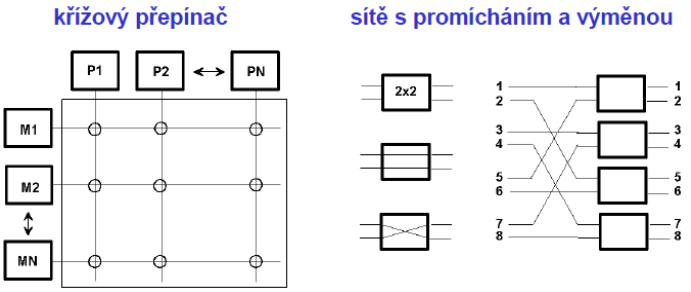
- Paralelní systémy
  - Požadavek
    - Zvyšování rychlosti a výkonnosti počítačů
    - Zvýšení spolehlivosti a bezpečnosti systému
    - Víceuživatelské prostředí
  - Vlastnosti
    - Probíhá několik procesů současně
    - Procesorová část nahrazena sítí spec/univ procesorů
  - Dělíme je na
    - Počítače řízené tokem instrukcí
    - Počítače s netradičním řízením

- Propojovací sítě a jejich topologie
  - Zajišťují propojení a komunikaci mezi procesory
  - Jsou
    - Statické (spojovací cesty zůstávají neměnné)

        - Dynamické (spojovací cesty vznikají a zanikají)
            - Spínače řízeny - lokálně a centrálně
    - Statické propojovací sítě



    - Dynamické propojovací sítě



    - Sběrnice
        - Dovoluje obousměrnou komunikaci „každý s každým“
        - Vždy jen jeden přenos v daném okamžiku
        - Omezená prostupnost přenosových cest

- SISD a paralelismus
    - Paralelní systémy SISD
        - Systémy VLIW
        - Zálohové systémy
        - Systémy používající pipelining
    - Systémy VLIW
        - Množství propoj. jednotek
        - Velmi dlouhé instrukce
        - V OZ inf. o řízení přenosových cest
        - Všechny oper. v instr. paralelně
        - Procesory jsou specializované
    - Zálohové systémy
        - Zvýšení spolehlivosti a bezpečnosti
            - Duplexní systém
            - Systém s majoritou
            - Biduplexní systém
        - Porovnávají se výskl. v komparátoru

- SIMD – maticové procesory
    - Větší počet prvků se zpracovává současně (řádek, sloupec)
    - Pole procesorů, synchronně provádějí tutéž operaci
    - Řízení společným řadičem
    - Aplikace: matice, lin. program, zprac. obrazu

- SIMD s lokální pamětí
    - Procesorové pole řídí universální počítač
        - Řeší nadřízený program
        - Rozhoduje o maticových úlohách
        - Zabezpečuje přesun dat
    - Řadič (procesor)

    - Každý procesor má paměť operandů
    - Procesory si posílají data

- SIMD se sdílenou pamětí
    - Procesory od pamětí odděleny (síť)
    - Počet paměťových modulů jiný, než počet procesorů
    - Základní distribuci dat do paměti modulů zajišťuje řadič

- MIMD – SMP
    - Paralelní systémy MIMD – multiprocesorové systémy
        - Každý procesor zpracovává data svého vlastního programu
            - Zvýšení výkonnosti
            - Zvýšení spolehlivosti a bezpečnosti (zálohování)
        - Dělení
            - Těsně vázané (společná paměť)
            - Volně vázané (vlastní paměť)
    - Těsně vázané multiprocesorové systémy
        - Malá vyrovnávací paměť
        - Procesory sdílejí společnou OP
        - Periférie mají malou autonomii
        - Propojená síť umožňuje libovolné propojení

- Volně vázané MIMD a NUMA
    - Volně vázané multiprocesorové systémy
        - Procesory s velkou lokální pamětí a vlastními periferiemi
        - Znační stupeň autonomie
        - Lokální paměť obsahuje program i data
        - Odpovídá počítačové síti, která není distribuována
    - NUMA (Non-Uniform Memory Access)
        - Těsnější vazba více CPU v uzly, ty propojené do větších celků, přitom paměťový subsystém nabízí plnou adresovatelnost

Sítě počítačů

- Vícepočítačové systémy
    - Množina vzájemně propojených autonomních počítačů (různé typy), které jsou vybaveny vlastní periférií a vlastním programovým vybavením

- Dovolují zajistit
    - Komunikaci lib. uživatele s programem na lib. počítači
    - Komunikaci lib. uživatelů mezi sebou
    - Komunikaci lib. programů mezi sebou

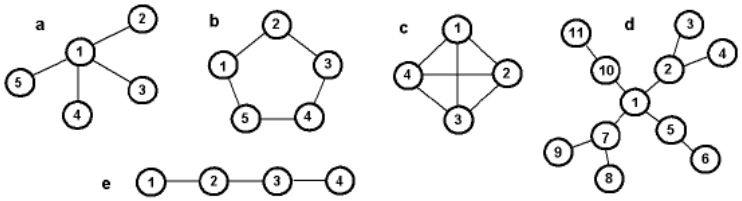
- Způsob spolupráce v síti
    - Služby mezi klientem a serverem. Spojení klient-server dynamické
    - Podobné volně vázaným multiprocesorovým systémům MIMD

- Přenos počítačových sítí
    - Komunikace mezi účastníky sítě
    - Sdílení nákladných prostředků
    - Bezpečnost a spolehlivost systému
    - Zvýšení výkonu
    - Řízení diatrib. technolog. Procesů

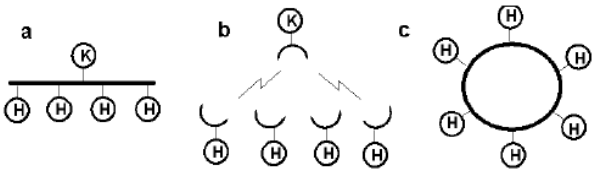
- Typy počítačových sítí
    - LAN (Local Area Networks)
        - propojit velký počet uživ.počítačů v lokál. měřítku (do 20 km)

        - sdílí společná pam. media a perif., realizace distrib. syst. (paralel.pr.)
        - pracují na sdíleném mediu (koaxiál, optika, dvoudrát), protokoly.
        - rychlosti přenosu 4Mb/s až 2Gb/s, zpoždění v řádech ms
    - MAN (Metropolitan Area Networks)
        - propojit počítače nebo LANy v oblasti velkých měst (do 200km)
        - sdílení databází, přenos souborů, vzdálené spouštění úloh
        - sdíl. medium (koax., radio), nebo bod-bod (telefon, opto), protokoly
        - rychlosti přenosu 100kb/s až 100Mb/s, zpoždění desítky ms
    - WAN (Wide Area Networks)
        - propojení poč. nebo sítí (LAN, MAN) v mezinár. měřítku (tisíce km)
        - sdílení databází, přenosy souborů, vzdálené spouštění úloh, info.sl.
        - dvoubod. spoje, spec.přenos.cesty (opto, radio, družice), protokoly
        - rychlosti přenosu 9.6 kb/s až 40Mb/s, zpoždění stovky ms.
    - Průmyslové a řídicí sítě - „CAN“ (Control Area Networks)
        - různé topologie a technologie spíše na kratší vzdálenosti
        - specifické požadavky na spolehlivost a zaručené časy doručení

- Topologie sítí
    - Propojovací struktury - topologie sítí
        - používají se statické propojovací struktury (dvoubod., vícebod.)
    - Dvoubodové propojovací struktury (spojení bod-bod)
        - Přímé propojení uživ. poč. s uzlovým poč. sítě nebo s jiným uživ. poč.



    - Topologie se volí s ohledem na:
        - Průchodnost sítě - závislá na počtu seriových spojů (min)
        - Spolehlivost sítě - možnost vybrat alternativní cestu při poruše (max)
        - Přizpůsobení procesu (výpočetnímu nebo řízení technologie)
    - Hierarchické řízení:- stromová struktura.
        - jeden počítač jako uzlový - střed sítě (kom.problém)
    - Vícebodové propojovací struktury (broadcast)
        - Sdílené medium (arbitr přístupu - centrální/distribuovaný)
        - Dovolují vzájemné propojení celé skupiny počítačů
        - Zprávu přijímají všichni uživatelé v síti
        - Přebírá zprávu uživatel, kterému je určena

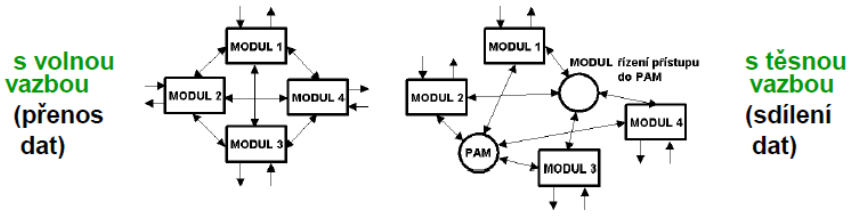


    - Topologie
        - Sběrnice (a)
        - Radiové nebo satelitní komunikace (b)
        - Topologie kruhu (c)

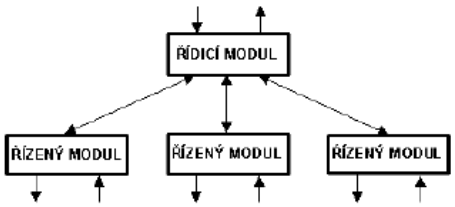
- Přenos dat v síti
    - Přepojování kanálů - fyzický dat. spoj mezi konc.účast. po dobu spojení
        - Malé a přesně definované zpoždění
        - Statická alokace spoje (ostatní čekají)

- Přepojování zpráv - komunikace po zprávách (adr. příj. + služ. údaje)
  - Sousedovi ve směru opt.cesty k adresátovi (store-and-forward).
  - Dynamičnost v alokaci segm.sítě, max využití sdíl. Kanálů
  - Složitá práce se zprávami a přiděl. dočasné paměti
- Přepojování paketů - komunikace po paketech (obdoba přepoj.zpráv)
  - Paket - část zprávy - pevná max. délka (data, adr.příjemce, služ. údaj)
  - Zjedn. práce průchozích uzlů, větší propustnost sítě
  - Složitější práce koncových stanic (rozklad/složení)
- Směrování - paměť uzlů o nejkratší cestě mezi uživateli (obnova kom.)
  - Virtuální kanály – dyn. se aktivují (pakety v pořadí jako při vysílání)
  - Datagramová služba - pakety opt. cestou bez ohledu na předchozí

- Počítačové sítě typu LAN
  - LAN - požadavky:
    - Rozložit výpočetní kapacitu, minimalizovat datové přenosy
    - Zajistit spolehlivost struktury a sdílení dat
    - Přímé propojení komunikujících účastníků
  - Topologie sítě
    - Fyzická - jak je síť propojena
    - Logická - jak stanice mezi sebou spolupracují
  - Způsob koordinace
    - Komunikace výpočet. modulů při řešení společné úlohy
    - Dělíme na:
      - Sítě s identickými moduly – multipočítačové
      - Sítě se spec. moduly - s rozloženou inteligencí
  - Logická topologie
    - Uspořádání systému na základě vzájemné spolupráce:
      - S rovnocennými moduly – modul rozdělení úloh je distribuován

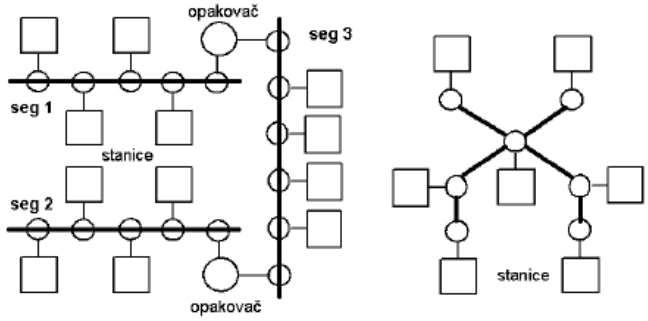


      - S jedním řídícím a několika podřízenými
        - Hierarchická struktura

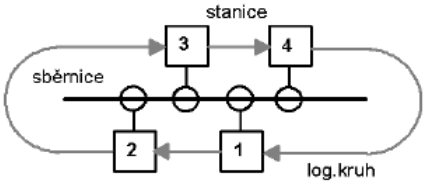


  - Fyziologická topologie
    - Volba má vliv na:
      - Rozšiřitelnost (snadnost doplňování počtu stanic)
      - Rekonfigurovatelnost (změna strukturu při závadě)
      - Spolehlivost (odolnost vůči poruchám)
      - Výkonnost (slož.obsluhy, přenos.rychl, zpoždění,…)
  - Topologie využívá:
    - Dvoubodové spoje - hvězda, strom, kruh
    - Vícebodové spoje - sběrnice, kruh (sběrnice)
- Signály – kódovány a modulovány
- Přenosové medium
  - Kroucený dvoudrát (max 300Mb/s), symetr., RS-422, RS-485

- Koax. kabel (max100Mb/s), nesymetr.
- Světlovodná vlákna (max 1Gb/s)

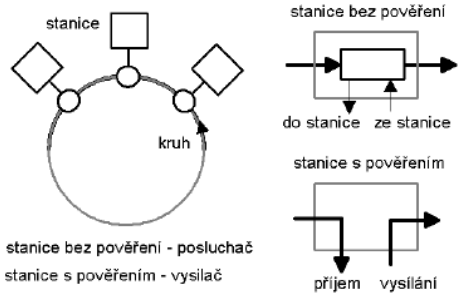


- Přidělování média a zařízení přenosu
  - Závisí na topologii sítě
  - Dvoubodová propojení - medium se nepřiděluje, strategií je optimalizace předávání paketů
  - Vícebodová propojení - základním problémem přidělování sdíleného media
  - Přidělování média
    - Centrální řízení
      - Jedna ze stanic sítě vyhrazena jako řídicí
      - Jednoduchý algoritmus (přiřazování priorit stanicím)
      - Zvláštní podkanál žádostí o přidělení (spolehlivost řídicí ?)
      - Způsoby přidělování - obdobné jako u sběrnic počítače
    - Distribuované řízení
      - Řídicí stanice distribuována k uživatelům
      - Stanice se samy hlásí (požadavky mají náhodný charakter)
      - Komunikační řadiče/procesory stanic rozhodují na základě:
        - Deterministických metod přidělování
        - Nederministických metod přidělování
  - Deterministické metody
    - Předávání t.zv. pověření (token) mezi komunikačními řadiči
      - Výhody: - bezkolizní přístup, začlenění a vyjmutí stanice ze sítě
      - Nevýhody: - determ. sled stanic, režijní časy (token, rekonf.)
  - Token bus - metoda přidělování pro:
    - Fyzické topologie typu sběrnice, hvězda (centr. bod je hub)
    - Logickou topologii typu logický kruh (řazeny svými adresami)



Token ring

pro fyz. i log. topologii dvoubodově zapojené kruhové sítě (lib. medium)
- dvoubod. spojení jsou jednosměrná, inform. se vrací zpět
- komunikační řadič/procesor obsahuje posuvný registr
- doba průchodu inf. sítí je dána: - délkou registrů
- počtem komunikačních stanic
- rychlostí přenosu mediem



- v klid.stavu obíhá obecné pověření
- vysílající získá toto pověření
- změní obecné pov. na rámec s daty
- po proběhn. sítí vysilač rámec likv.
- kontrola spolehlivosti přenosu
- snadné potvrzování příjmu
- po odvysílání vyšle obecné pov.
- není co vyslat - přeposílá obecné p.

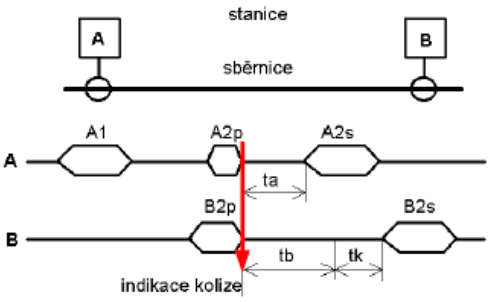
Výhody: - ohraničená doba zpoždění přenosu paketu v síti
- vysoké využití kapacity kanálu

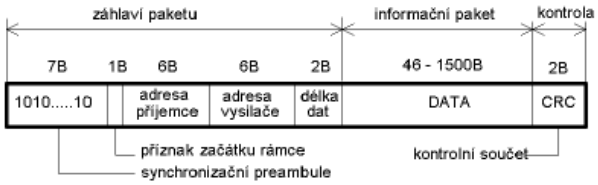
Nedeterministické metody

přístup na základě pozorování provozu pro topologie broadcast - sběrnice

CSMA (přístup s odposlechem nosné)
- stanice před odesláním paketu testuje stav kanálu (media)
- kanál volný/obsazen (není/je identifikována nosná), lze/nelze vysílat
  - přístup za náhod.stanov. dobu (nenaléhající CSMA)
  - čeká se na dokončení relace (naléhající CSMA).

CSMA/CD - řešení kolize vysilačů (pokles úr. nosné), používá Ethernet
- všechny přestanou vysílat, (jam)
- pokus o nové vysílání za interval náhodné délky (prodlužuje se)



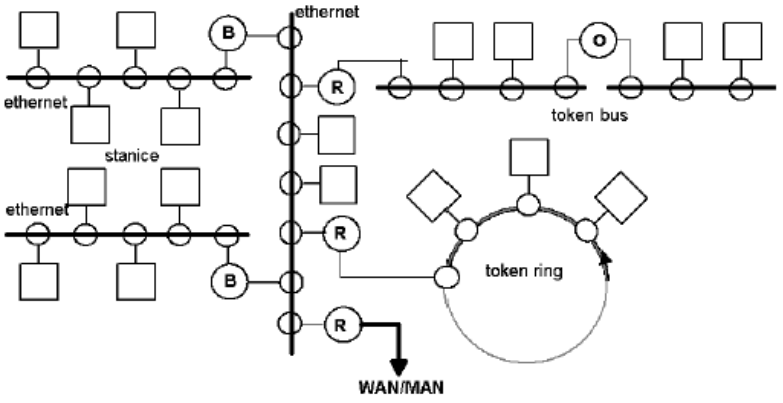


Propojování v sítích LAN (max. délka segmentu < 500m)

O - opakovač (repeater) - kopie přenosu dat mezi segmenty sítě LAN

B - most (bridge) - propojuje sítě se shodnou strukturou rámců
- rozhoduje o směrování paketů do správné sítě
- umožňuje vzájem. izolaci segmentů i sítí navzájem

R - brána (router), - transformace přenášené informace a její směrování mezi LAN sítěmi odlišného typu
- konstruuje novou obálku pro připoj.síť
- transformace a směrování rámců mezi sítěmi LAN (Ethernet - token …)



G – gateway - router + nadřaz.síťové služby (LAN – WAN)

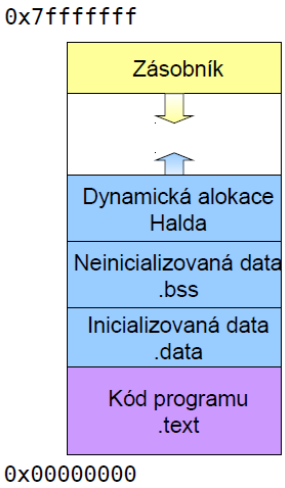
Předávání parametrů funkcím a virtuálním instrukcím OS

- Volání běžných funkcí (podprogramů)
  - Možnosti předávání parametrů – v registrech, přes zásobník, s využitím registrových oken
  - Způsob definuje volací konvence – musí odpovídat možnostem daného CPU a je potřeba, aby se na ní shodil tvůrci kompilátorů, uživatelských a systémových knihoven
  - Zásobníkové rámce pro uložení lokálních proměnných a alloca()

- Volání systémových služeb
  - Přechod mezi uživatelským a systémovým režimem

- Vzdálená volání funkcí a metod (volání nemůže číst paměť)
  - Přes síť (RPC, CORBA, SOAP, XML-RPC)
  - Lokální jako síť + další metody: OLE, UNO, D-bus,…

- Rozložením programu ve virtuálním adresním prostoru
  - Do adresního prostoru procesu je nahraný – mapovaný soubor obsahující kód a inicializovaná data programu – sekce .data a .text
  - Oblast pro neinicializovaná data (.bss) je pro C programy nulová
  - Nastaví se ukazatel zásobníku a předá řízení na startovací adresu programu (_start)
  - Dynamická paměť je alokovaná do symbolu _end nastaveného na konec .bss



- Postup volání podprogramu
  - Volající program vypočítá hodnoty parametrů
  - Uloží data, která využívala registry, které mohou být volaným programem změněné
  - Parametry jsou uloženy do registrů, a nebo na zásobník tak jak definuje použitá volající konvence
  - Řízení se přesune na první instrukci podprogramu, přitom je zajištěné, že návratová adresa je uložena na zásobník, nebo registr
  - Podprogram ukládá hodnoty registrů, které musí být zachovány a sám je chce využít
  - Alokuje oblast pro lokální proměnné
  - Provede vlastní tělo podprogramu
  - Uloží výsledek do konvencí daného registru
  - Obnovení uložené registry a provede návrat na následující instrukci
  - Instrukce pro návrat může v některých případech uvolnit parametry ze zásobníku, většinou je to ale starost volajícího

- Registry a volající konvence MIPS

  a0 – a3: argumenty (registry $4 – $7)

  v0, v1: výsledná hodnota funkce ($2 a $3)

  t0 – t9: prostor pro dočasné hodnoty ($8-$15,$24,$25)
  - volaná funkce je může používat a přepsat

  at: pomocný registr pro assembler ($1)

  k0, k1: rezervované pro účely jádra OS ($26, $27)

  s0 – s7: volanou funkcí ukládané/zachované registry ($16-$23)
  - pokud jsou využity volaným, musí být nejdříve uloženy

  gp: ukazatel na globální statická data ($28)

  sp: ukazatel zásobníku ($29)

  fp: ukazatel na začátek rámce na zásobníku ($30)

  ra: registr návratové adresy ($31) – implicitně používaný instrukcí
  jal – jump and link

- MIPS instrukce pro volání a návrat
  - Volání podprogramu: jump and link
    - Adresa instrukce následující za instrukcí jal je uložena do registru ra ($31)
    - Cílová adresa je uložena do čítače instrukcí PC
  - Návrat z podprogramu: jump register
    - Obsah registru ra je přesunutý do PC
    - Instrukce je také použitelná pro skoky na vypočítanou adresu nebo adresu z tabulky

- Rozdíl mezi voláním a skokem
  - Skok neuloží návratovou hodnotu, kód tedy nejde využít z více míst
  - Volání s využitím registru ra umožňuje volat podprogramem tehdy, kdy je potřeba

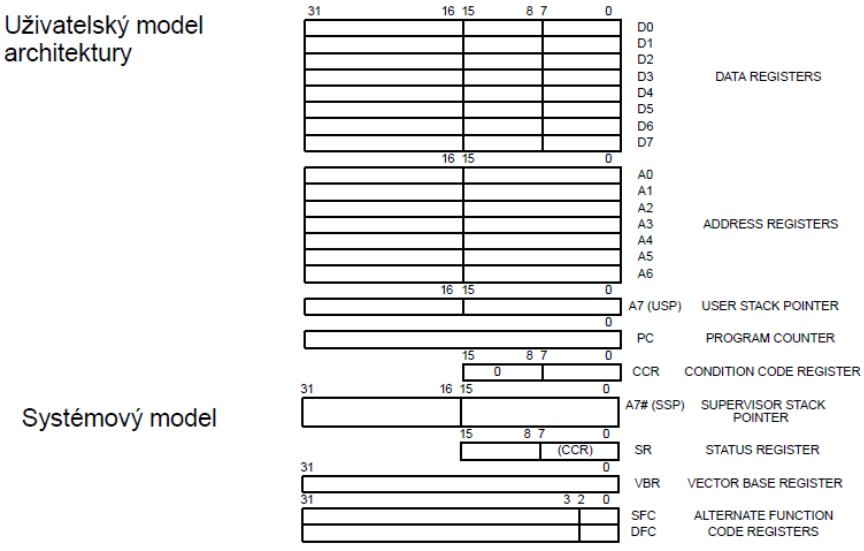
- Systémové volání – kroky
  - Systémové služby (např.: open, close, read, write, octl, map) jsou většinou běžným programům zpřístupněným přes běžné funkce C knihovny (GLIBC, CRT atd…), kterým se předávají parametry běžným způsobem
  - Knihovní funkce pak přesune parametry nejčastěji do smluvených registrů, kde je očekává jádro OS
  - Do zvoleného registru vloží číslo systémové služby (EAX na x86)
  - Vyvolá výjimku (x86 Linux např.: 0x80 nebo sysenter)
  - Obslužná rutina jádra podle čísla dekóduje parametry služby a zavolá již obvyklým způsobem výkonnou fci
  - Jádro uloží do registru návratový kód a provede přepnutí zpět do uživatelského režimu
  - Zde je vyhodnoceno hlášení chyb (errno) a běžný návrat do volajícího programu

%eax	Name	Source	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	kernel/exit.c	int				
3	sys_read	fs/read_write.c	unsigned int	char	size_t		
4	sys_write	fs/read_write.c	unsigned int	const char	size_t		
15	sys_chmod	fs/open.c	const char	mode_t			
20	sys_getpid	kernel/timer.c	void				
21	sys_mount	fs/namespace.c	char __user *	char __user *	char __user *	unsigned long	void __user *
88	sys_reboot	kernel/sys.c	int	int	unsigned int	void __user *	

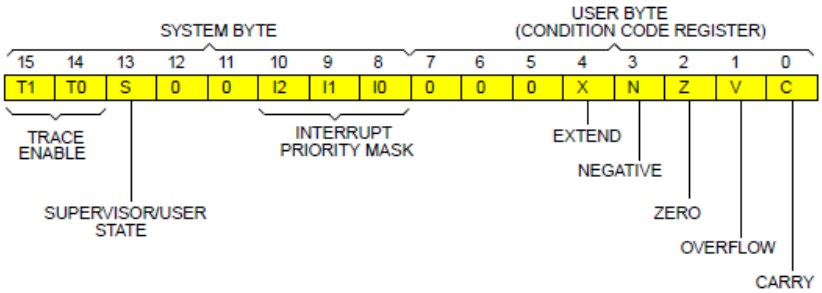
System Call Number
System Call Name
First parameter
Second parameter
Third parameter

Procesory Motorola 68000, 683xx a ColdFire

- Základní registr procesorů



- Stavový registr



- N (Negative) = 1 pokud je nejvíce významný byte operandu nebo výsledku roven 1 (negativní výsledek v doplňkovém kódu)
- Z (Zero) = 1 pokud jsou veškeré bity operandu nebo výsledku nulové
- V (Overflow) = 1 pokud je výsledek mimo zobrazitelný rozsah (např.: přetečení při znaménkových operacích)
- C (Carry) = 1 pokud dochází k přenosu (carry) z nejvíce významného bitu během aritmetické operace nebo je potřeba výpůjčka (barrow) při odčítání
- X (Extend) – použitý při rozšíření operací na vícenásobnou přesnost (odpovídá hodnotě C)
- T1, T0 (trace) – pokud je některý z bitů nastavený, dojde ke generování výjimky pro provedení každé instrukce nebo po instrukcích změny toku programu (skok, návrat, volání)
- S (Supervisor) – pokud je nastaven na 1, procesor se nachází v systémovém režimu a SP odpovídá SSP. Jinak se procesor nachází v uživatelském režimu, SP odpovídá USP a manipulace se systémovou částí není možná a na přístupy k paměti (MMU) jsou aplikovaná omezení příslušná uživatelskému režimu
- I2, I1, I0 (Interrupt mask) – definuje úroveň přerušení po kterou je přijetí žádosti blokované – odložené na později. Výjimkou jsou žádosti úrovně 7, která nejsou maskovatelná

- Instrukční soubor 68000

Mnemonic	Descritption
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical AND
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B<*cc*>	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Reigster Against Bounds
CLR	Clear Operand
CMP	Compare
DB<*cc*>	Decrement and Branch Conditionally
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive OR
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right

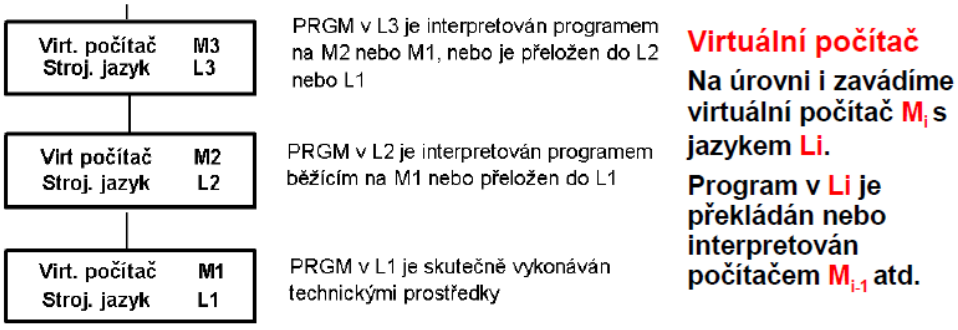
Mnemonic	Descritption
MOVE	Move Source to Destination
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extened
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical OR
PEA	Push effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right wiht Extend
RTD	Return and Delocate
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal wiht Extend
S<*cc*>	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap data register halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink Stack Frame

Víceúrovňový model počítače

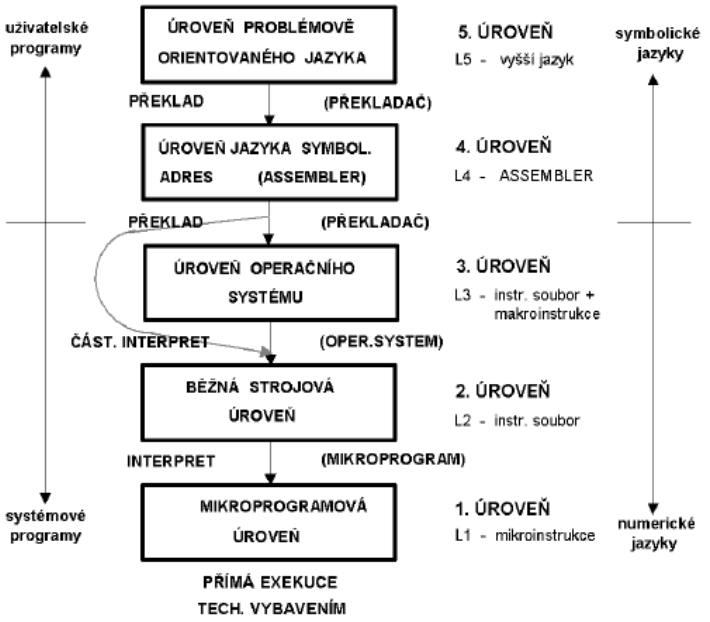
- Strojový jazyk počítače
  - Monož. jedn. instruk – do ní převést prog. pro výkon
  - Úroveň L1 – abeceda {0,1}, obtížná komunikace

- Jazyky vyšší úrovně
  - Vhodnější pro lidskou komunikaci L2 + další

- Vykonání programu v L2 na stroji jenž má L1
  - Kompilace – instrukce v L2 se nahradí posloupností instrukcí v L1
  - Interpretace – program v L1 zpracovává program v L2 jako data (pomalejší)



- Současný, mnoha úrovňový počítač



Vývoj víceúr. 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.
- HW a SW jsou logicky ekvivalentní (lze je navzájem nahradit)
- souboj a splývání RISC a CISC procesorů

- Procesy a jejich stavy

PROCES - probíhající program (program - pasivní, proces - aktivní)

STAV PROCESU - info, které při zast. procesu umožní jeho znovuspuštění

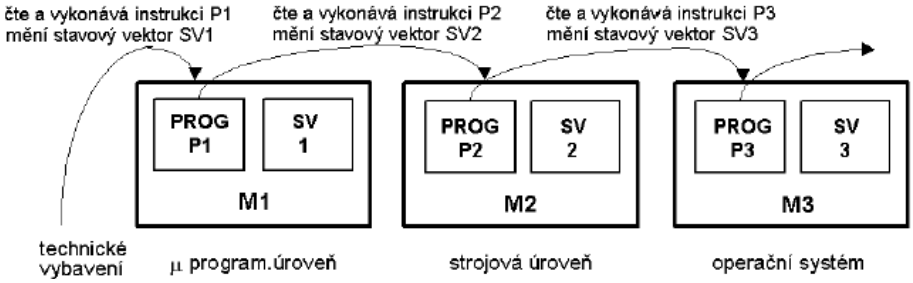
1. program
2. násled. instrukce
3. hodnoty proměnných a data
4. stavy a polohy všech I/O

PŘEDPOKLAD: proces P sám nemění svůj program!

STAVOVÝ VEKTOR - proměnné složky stavu procesu – mění HW nebo prgm.

PROCES = PROG + STAV. VEKT.

ZMĚNA STAV. V. – stav. vektor proc. P2 mění P1 -> P1 interpret programu P2

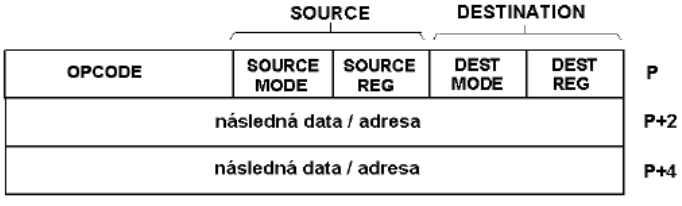


- Konvenční strojová úroveň (ISA) (M2)
  - Definovaná instrukční sadou (často označovaná architektura procesoru)
  - ISA (Industry Set Architectury) x Mikroarchitektura (Inmpelmentace v M1)

Instrukční formát

- skládá se z polí, (jedno nebo po sobě jdoucí slova)
- adr. části mohou být adresami nebo odkazy na reg. (nepřímo)

Nejpoužívanější formát je u CISC dvouadresový, u RISC tříoperandový jen mezi registry



- Požadavky na dobrého programátora
    - Každý programátor by měl umět vytvořit takový kód, aby splnil následující požadavky:
        - Efektivní využití procesoru, tj. rychlý kód
        - Efektivní využití paměti, tj. používat jen takové datové typy, aby stačily k účelům programátora a zároveň nezabíraly příliš místa v paměti
        - Následování stanovených pravidel konvence, tj. pravidla zápisu pro lepší čtení zdrojového kódu (mezery, odsazení..)
        - Možnost jednoduché úpravy a zlepšování kódu (používání funkcí nebo OOP podle možností programátora)
        - Dobře otestovaný a pevný kód, tj. zjištění a oprava možných chyb (většinou se vyskytují takové chyby, které neočekáváme, že by se mohly objevit!)
        - Dokumentace (návod pro používání programu)

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

- 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)