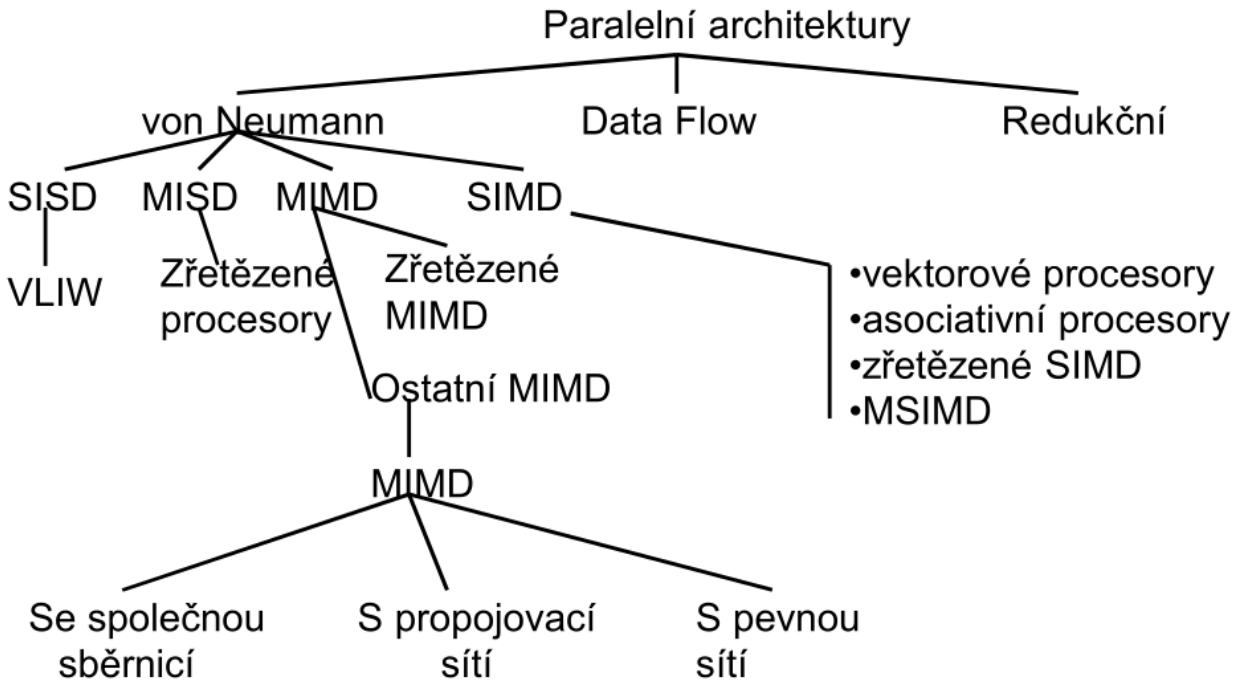
1) Architektury

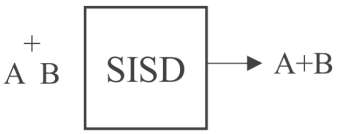
Klasifikace paralelních systémů

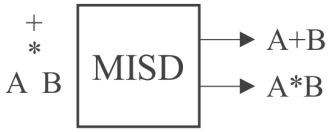


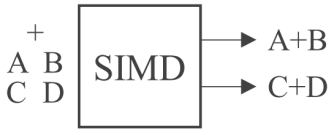
Von Neumannova / Harvadská architektura

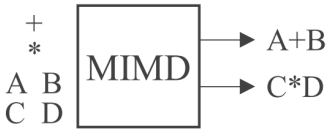
- Má instrukční čítač, je známo, kde se nacházíme v posloupnosti instrukcí

Flynnova klasifikace







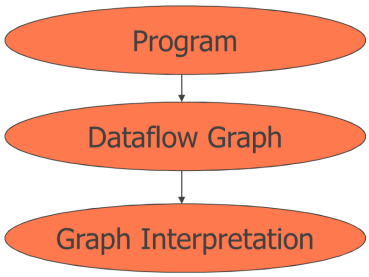


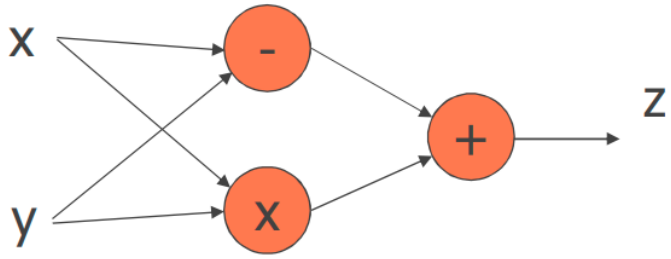
- **SISD**

    - V paralelním prostředí: **VLIW** (Very long instruction word)
    - Instrukce až tisíce bitů, je rozdělena na pole, které provádějí samostatné operace (paralelně).
- **SIMD**
    - Vektorové / maticové procesory
- **MISD**
    - Zřetězené procesory (překrývání instrukcí)Zřetě
- **MIMD**
    - Multiprocesory (sdílená paměť)
    - Multicomputery (předávání zpráv) – Distribuované systémy

Data Flow architektura

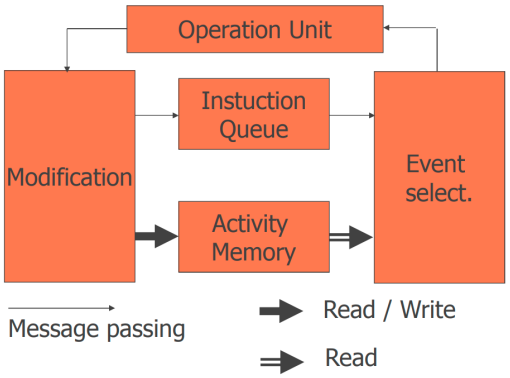
- Nemá instrukční čítač
- Vyhodnocuje výrazy, které jsou v paměti reprezentovány pomocí grafu.
- Výpočet provádí tehdy, je-li pro operaci dostatečný počet vstupních dat.





- Typicky funkcionální jazyky
- Při psaní programu nemusíme přemýšlet paralelně, architektura se sama snaží najít všechno co lze vypočítat paralelně
- Realizovatelná architektura, používá se pro konkrétní věci (např. simulace)

Dataflow procesor



Redukční architektura

- Program je napsán jako řetězec, který se počítač snaží postupně redukovat, až dojde k jednomu výsledku.
- Mám pole ve kterém jsou řetězce, toto pole prohledávám a snažím se najít takové podřetězce, které se dají zredukovat na něco jednoduššího.
- Redukce se provádí paralelně.
- Zatím nemá využití – hledá se obtížně hardware realizace.

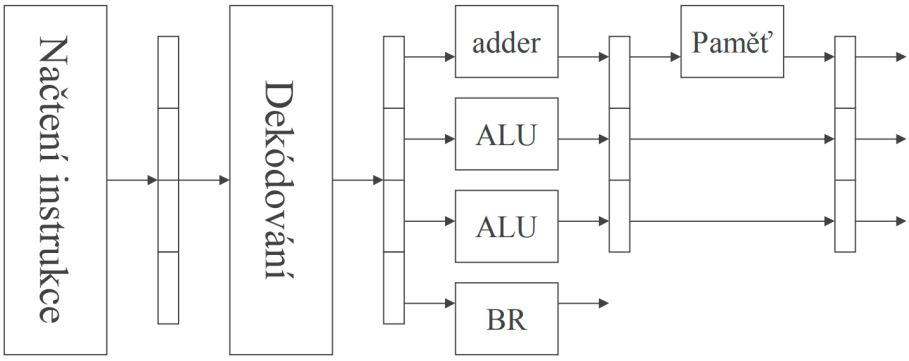
Redukce pro x=3 y=2

- +<(*<3 2>)(-<3 2>)>
- +< 6 1 >
- 7

Konkrétní Von Neumannovy (Harvardské) architektury

VLIW – Very Long Instruction Word

- Jediný tok řízení, který řídí všechny procesory (SISD)
- Přesto lze paralelismus
- Operační kód je velmi dlouhý a je rozdělen na části, které provádí různé operace (ale jsou řízeny jedním čítačem).



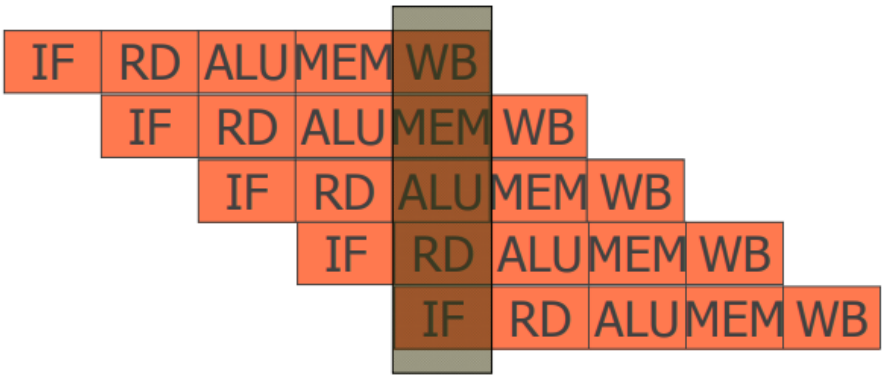
- Překladač musí přeložit dlouhou instrukci na menší a poskládat je tak, aby šly řešit paralelně.
- Oproti tomu Super Skalární procesory rozložení instrukcí řeší až na úrovni hardware
- Výhody
    - Jednoduchý hardware (instrukce překládá překladač)

    - Dobře škálovatelný
- Nevýhody
    - Podmíněné skoky jsou problém
    - Problém toku dat – instrukce zpracované v jednom kroku, nemohou navzájem používat své výsledky
    - Velikost programu – synchronizační NOPy

Zřetězené procesory

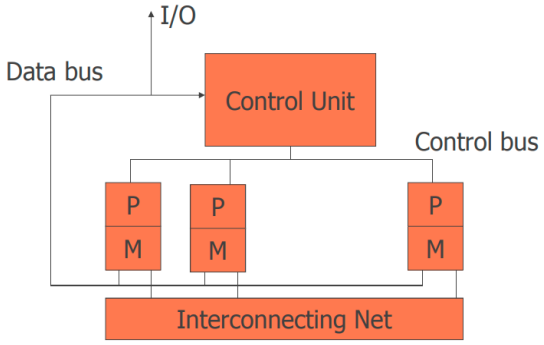
- MISD
- Lineárně propojené procesory
- Řešení úloh s proudovým charakterem
- Data procházejí postupně jednotlivými procesory

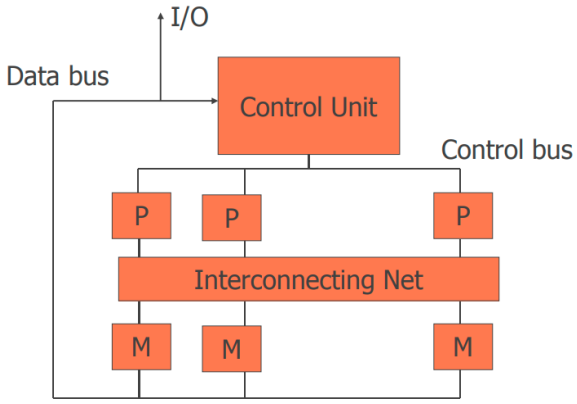
- “Fáze” instrukcí



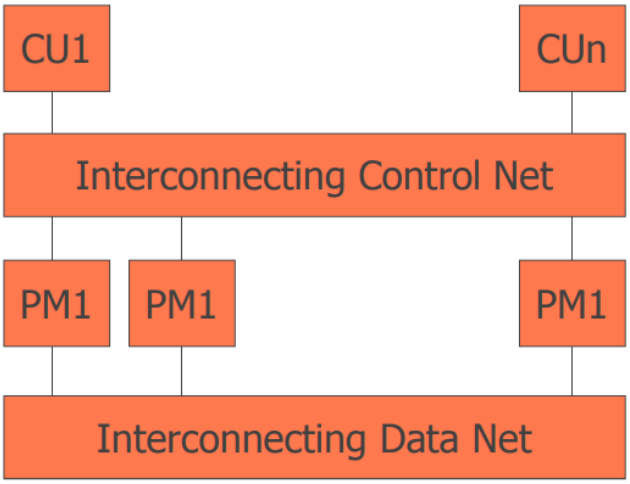
Vektorové procesory

- SIMD
- Jedna řídící jednotka
- Propojení více výpočetních (datových) jednotek, které mají každá segment paměti.
- Všechny procesory provádějí stejnou instrukci, ale s různými daty.
- Propojovací síť propujuje jednotlivé paměťové moduly.
- Rozdílný čas přístupu do paměti.
- Dva modelu (obrázky)





- Reálně se používá Multiple SIMD (MSIMD)
    - Několik řídících jednotek, každá si může zabrat několik procesorů
    - Z důvodu běhu OS apod.



- Výhody
    - Jednodušší než MIMD (menší počet tranzistorů na procesor)
    - Menší nároky na paměť.
    - Žádná synchronizace mezi procesy
- Nevýhody
    - Ne všechny problémy jsou datově paralelizovatelné
    - Pokles výkonnosti u programů s mnoha podmíněnými skoky
    - Nejsou vhodné při malém počtu procesorůGranularGranularGranularGranular

Granularita paralelismu

Uvnitř instrukcí

- Uvnitř jedné instrukce několik jednotek, které paralelně provádí činnost té instrukce
- Nejjemnější paralelismus
- Pro programátora neviditelný
- Např.:
    - Načíst operační kód, paralelně se spočítá jeho hodnota a zároveň se inkrementuje programový čítač

Mezi instrukcemi

- Některé instrukce se provádějí paralelně
- Není vidět na úrovní programovacího jazyka (na úrovni assembleru může, záleží na implementaci)

Mezi příkazy

- Vektorové počítače
- Specializované koprocesory (FPU)
- C, pragma překladačům, knihovny, apod.
- Např.:
    - Vektorizace smyček

Mezi bloky procesu

- Vlákna (threads)
- Obvykle je mezi vlákny sdílen adresový prostor, ale každé má oddělený programový čítač
- Vyšší programovací jazyky

Mezi procesy

- Žádná sdílená paměť
- Nutné zajistit komunikaci a synchronizaci
- Vyšší programovací jazyky

2) Komunikace

Synchronizace

- Zajištění požadovaných vztahů mezi událostmi.
- Nepřenáší se data

Prostředky

- Zasílání zpráv
- Semafor
- Monitor
- Bariéra

Typické synchronizační úlohy

- Soupeření – vzájemné vyloučení (čtenáři – písaři)
- Kooperace – dohoda (producent - konzument)

Komunikace

- Přenáší data
    - Předávání informací mezi procesy (vlákny)
- Prostředky pro komunikaci
    - Sdílená paměť
    - Zasílání zpráv

Sdílená paměť

- Skutečná – HW
- Simulovaná – na úrovni OS
- Všechny procesy mají přístup do společného paměťového prostoru
- Je třeba řešit současný přístup k jedné buňce paměti

EREW – Exclusive read exclusive write

- Snadný pro HW, složité pro algoritmy

ERCW – Exclusive read concurrent write

- Nelogická varianta, souběžné čtení je daleko snažší zajistit než souběžný zápis, v reálu neexistuje

CREW – Concurrent read exclusive write

- Něco mezi

CRCW – Concurrent read concurrent write

- Složitý pro HW, snadný pro algoritmy
- Několik způsobů řešení zápisových konfliktů
    - **COMMON** – všechny zapisované hodnoty musí být shodné (logický OR)
    - **ARBITRATY** – zapisované hodnoty můžou být různé a zapíše se libovolná z nich (vyhledávání, index)
    - **PRIORITY** – zapisované hodnoty mohou být různé a zapíše se ta s nejvyšší prioritou (vyhledávání, nejlevější index)

Zasílání zpráv

- Každý procesor má svůj adresový prostor
- Komunikace probíhá pomocí zpráv

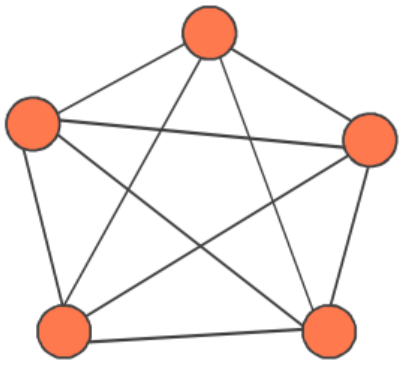
Propojovací sítě

- Propojit procesory se sdílenou pamětí
- Propojit procesory spolu

Statická

- Během výpočtu se propojení nemění
- Používají se pro architektury bez sdílené paměti
- Různé architektury

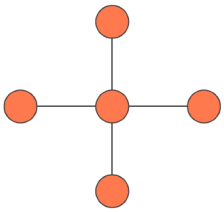
Úplné propojení



- Diameter = 1
- Arc connectivity = p-1
- Bisection width = $p^2/4$

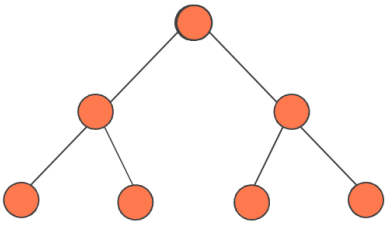
Hvězda

- pro architektury slave – master



- Diameter = 2
- Arc connectivity = 1
- Bisection width = $(p-1)/2$

Stromy

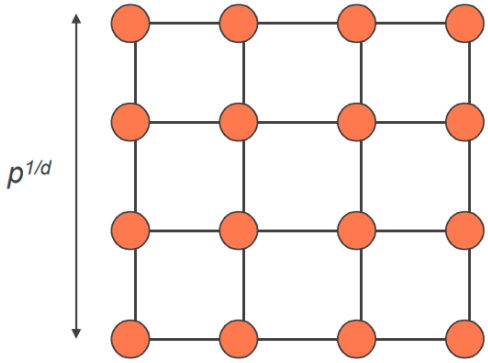


- Diameter = $2 \log_d ((p+1)/2)$
- Arc connectivity = 1
- Bisection width = 1
- Obvykle binární

Lineární pole

- Diameter = $p-1$
- Arc connectivity = 1
- Bisection width = 1

Mřížka



- Kartézský součin d lineárních polí, z nichž každé má $p^{1/d}$ uzlů
- Diameter = $dp^{1/d}$
- Arc connectivity = d
- Bisection width = $2p^{(1-1/d)}$
- Obvykle d=2

Dynamická

- Během výpočtu se propojení mění (uvnitř běhu algoritmu)
- Více se do toho pouštět nebudeme – složité