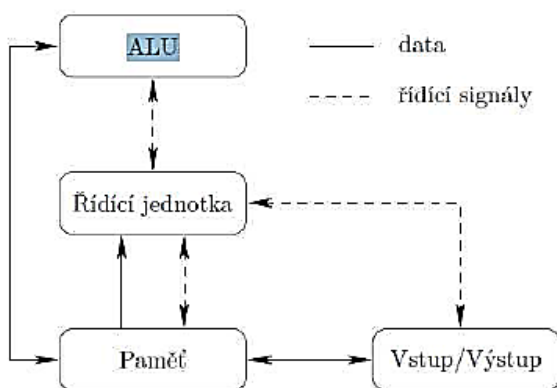


Von Neumannova architektura

- počítač se skládá z řadiče, ALU, paměti a V/V periférií
- struktura počítače je nezávislá na řešené úloze
- program je dán obsahem paměti
- společná paměť pro data a instrukce
- instrukce se vykonávají sekvenčně, pořadí lze měnit skoky
- krok procesoru závisí na předchozím kroku
- paměť je adresovatelná po blocích se stejnou velikostí, pořadové číslo určuje jejich adresu
- používá se výhradně dvojková soustava
- 1 zbernica

Výhody: jednotný přístup k datům a instrukcím, programátor určuje rozdělení paměti (neplývá se jí), jedna sběrnice

Nevýhody: jedna sběrnice, jedna technologie pro paměti, přepis kódu



Obrázek 1: Základní schéma počítače podle von Neumanna

Dnešní počítače

- superskalární¹ architektura, zřetězení → paralelismus
- řadič a ALU je na jednom čipu (společně s cache a někdy i grafickým čipem - APU), více procesorů na jednom čipu

Nevýhody Von Neumana oproti dnešním PC

- VN pracuje s programem sekvenčně a podle samotného principu zpracovává pouze jeden program.

Dnešní procesory

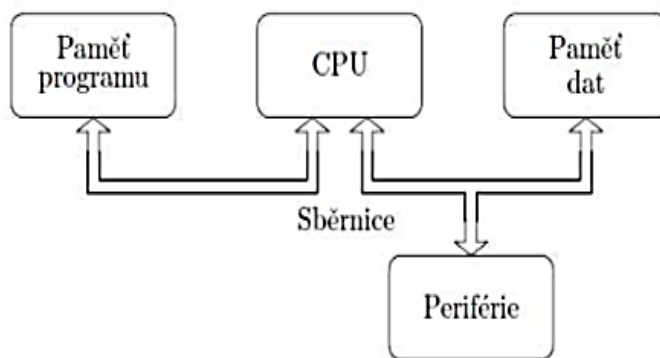
- pracují nad více programy současně, moderní CPU umožňují zavadení do paměti jen potřebných částí programu (on-demand) a tak jde optimalizovat paměť více než u VN (celý program v paměti)

Harvardská architektura

- instrukce a data jsou v oddělené paměti
- 2 zbernice

Výhody: program se nemůže přepsat, možnost různých technologií pro paměti dat a instrukcí, paralelismus dat a instrukcí

Nevýhody: nákladnější a složitější konstrukce (dvě sběrnice - složitější řídicí jednotka v CPU), nelze využít nevyužitou paměť dat pro instrukce a naopak



Obrázek 3: Harvardská architektura počítače

¹ **Superskalární arch.** znamená, že procesor načte více operací při jednom clocku a rozdělí je mezi jednotlivé části procesoru - například ALU a Shifter nebo Multiplier (atp.). Většinou to umí právě SIMD a MIMD procesory (superskalárnost zde vyjadřuje právě to MULTIPLE DATA)

CISC

(complex instruction set computer)

- označení pro skupiny procesorů vyznačující se sadou instrukcí, kterých je velké množství, mají nejednotnou délku a obsahující instrukce, které řeší několik operací naráz, obsahují relativně malý počet registrů
- toto pojmenování bylo zavedeno až zpětně po představení myšlenky RISC procesorů, těžko se hledá čistě CISC procesor, CISC procesory dnes mají prvky jak z CISC, tak z RISC

Výhody: specializované instrukce pro provedení složitých úkolů

Nevýhody: velká složitost provedení procesoru (instrukce musejí být naprogramovány mikroprogramově) i samotné instrukční sady (problémy s překladem z vyšších jazyků)

Typický zástupce: procesory řady Intel x86 (od Pentia Pro ve skutečnosti vnitřní implementace procesoru používá architekturu RISC, nicméně je zpětně kompatibilní s instrukční sadou x86, uvnitř procesoru probíhá překlad z x86 CISC na RISC instrukce)

RISC

(reduced instruction set computer)

- tato architektura byla vymyšlena a navržena v průběhu 70. let, cílem bylo redukovat složitost výroby procesorů a dosáhnout zpracování jedné instrukce během jednoho strojového cyklu

Vlastnosti, zásady a výhody platformy RISC

- Tato architektura se vyznačuje hlavně redukovanou instrukční sadou, která obsahuje pouze ty nejzákladnější instrukce (složitější operace se řeší jejich kombinací v kódu programu), v každém strojovém cyklu je dokončena právě jedna instrukce
- Instrukce mají pevnou délku a jednotný formát (spolu s jejich redukovaným počtem dovoluje jednodušší konstrukci procesoru, lze použít obvodový řadič místo mikroprogramového)
- Použití rychlé dočasné paměti - velký počet registrů a použití cache paměti
- Přístup do paměti provádí pouze specializované instrukce (LOAD - načíst / STORE - uložit)
- Používá se zřetězení instrukcí (pipeline) pro zrychlení procesoru
- Složitě instrukce nejsou implementovány procesorem, o optimalizaci se stará překladač

Nevýhody: jelikož se používají pouze základní instrukce, délka programů se mírně zvětšila, očekávané zpomalení se ale v praxi nepotvrdilo

Zástupce: ARM, MIPS



(RISC) Zřetězení

(pipelining)

Zpracování instrukce v procesoru je rozděleno do několika kroků, přičemž každý z těchto kroků lze provádět samostatnou jednotkou. Takto lze vykonávat několik instrukcí paralelně (např. první instrukce se vykonává, zatímco druhá se dekoduje a třetí načítá). Jednotlivé kroky by měly být stejně rychlé, jinak bude celé zpracování váznout na rychlosti nejpomalejší části. Jednotlivé kroky jsou odděleny vyrovnávacími registry, které udržují mezivýsledky a vyrovnávají případné časové rozdíly.

Aritmetické zřetězení

Např. ve FLU (floating-point unit) lze zřetěžit součet dvou čísel s plovoucí řádovou čárkou. Proces je rozdělen na porovnání exponentu, posuv, sečtení a normalizaci výsledku.

Instrukční zřetězení

Provedení instrukce lze rozdělit na několik kroků, např.:

- 1) Načtení instrukce z paměti
- 2) Dekódování instrukce
- 3) Výpočet adresy operandů
- 4) Výběr operandů
- 5) Provedení operace s operandy
- 6) Zapsání výsledku (retire)
- 7) Úprava IP (instruction pointeru)

Vzniká problém při větvení v kódu, pokud dojde ke skoku, musí se pipeline vymazat a načíst odznovu - zpomalení. Používá se proto např. predikce skoku, která se snaží předpovědět, jestli se skok provede nebo ne (buď staticky - predikce je součástí instrukce anebo dynamicky - např. 2bitová predikce favorizuje jednu akci až po dvou stejných operacích po sobě). Popřípadě v superskalární architektuře lze provádět obě dvě možnosti (kód při skoku i kód, pokud se skok neprovede) současně, a neplatná možnost se poté zahodí. Skoky lze také odkládat, pokud za nimi následují na skoku nezávislé instrukce.

Datové hazardy

Pokud instrukce potřebuje pracovat s daty, které závisí na výsledku jiné instrukce, ta ale ještě není dokončená. Řeší se vyhýbáním se této situaci překladačem anebo počkáním na výsledek druhé instrukce (vyplnění části pipeline bublinami - NOP - no operation). Vypočtené hodnoty lze také předávat dopředu v pipeline (forwarding).

Strukturální hazardy

Vznikají, když se více instrukcí pokusí ve stejný čas přistoupit ke sdílenému hardwarovému prostředku (typicky k paměti). Řeší se například vyplnění bublinami (NOP).

Řídící hazardy

Vznikají při použití instrukcí podmíněných a nepodmíněných skoků. Řeší se například predikcí skoků.

Zřetězení procesů

Zřetěžené zpracování toku dat, kde několik procesorů provádí různý kód (program) nad stejnými daty, které postupně zpracovávají.

- skalární řetězec (skalární data), vektorový řetězec (vektory)
- statický řetězec (má jednu pevně určenou konfiguraci), dynamický řetězec (za běhu si vybírá moduly, které bude používat)
- monofunkční řetězec (vykonává pouze jednu funkci), multifunkční řetězec (je schopen vykonávat různé funkce)

(RISC) Skok

Vzniká problém při větvení v kódu, pokud dojde ke skoku, musí se pipeline vymazat a načíst odznovu - zpomalení. Používá se proto např. predikce skoku, která se snaží předpovědět, jestli se skok provede nebo ne (buď staticky - predikce je součástí instrukce anebo dynamicky - např. 2bitová predikce favorizuje jednu akci až po dvou stejných operacích po sobě). Popřípadě v superskalární architektuře lze provádět obě dvě možnosti (kód při skoku i kód, pokud se skok neprovede) současně, a neplatná možnost se poté zahodí. Skoky lze také odkládat, pokud za nimi následují na skoku nezávislé instrukce.

Problémy plnění fronty instrukcí

- nepodmíněný skok (stejný problém jako Datový hazard), řešení: optimalizace už na úrovni překladače - pořadí instrukcí
- podmíněný skok: adresu sic většinou už známe, ale můžou nastat dvě situace, kdy se skok provede, či nikoliv. Řešení jsou dvě:
 - a. provádí se dál, a až podle vykonané instrukce skoku se buď rozpracované instrukce ignorují nebo využijí
 - b. implementují se dvě fronty, každá vykonává jednu možnost

Metody pro zlepšení plnění fronty instrukcí

- Bit predikce skoku
 - statická: již překladač tento bit umístí
 - dynamická: ukládá si jej procesor v registru
 - 1b: ukládá jeden bit, který rozhoduje zda byl předchozí skok proveden
 - 2b: až 4 stavy, které říkají zda se skok nejspíše provede či ne (4 stavy -> A/N)
- Zpoždění skokové instrukce: při podmíněném skoku se vykonávají ještě instrukce než dojde ke skoku, což může narušit stav programu, řešení jsou dvě:
 - místo následných instrukcí vložit NOP instrukci (která nic nedělá) -> neefektivní
 - najít v programu instrukce, které nejsou závislé na pořadí zpracování a ty (přeskupit a) zpracovat (místo NOP)
- Použití paměti skoku: prováděné skoky se ukládají do tabulky, aby se zlepšila efektivita skoků, navíc se k ní používá jedno či dvou bitová predikce

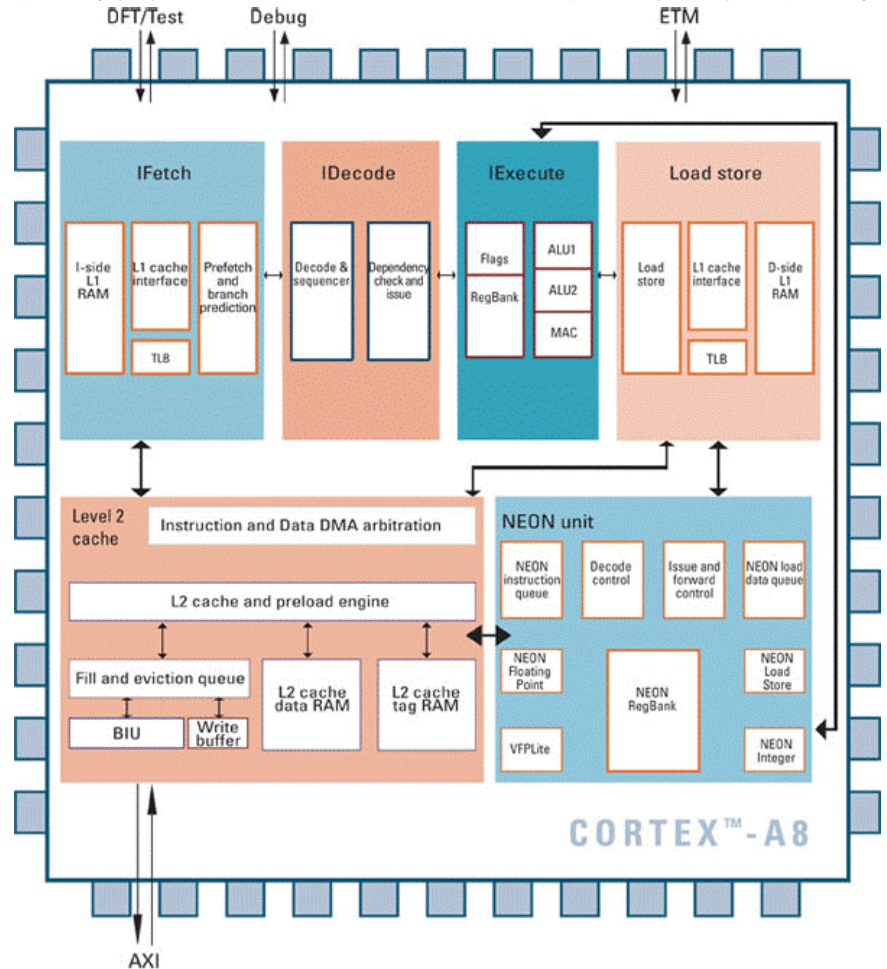
ARM

(Advanced RISC Machine, původně Acorn RISC Machine)

- plně 32-bitové, nejpoužívanější procesory pro mobilní zařízení
- procesory s nízkou spotřebou používané hlavně v mobilních zařízeních, řídí se filozofií RISC
- přístup do paměti pouze pomocí instrukcí LOAD/STORE (ostatní instrukce šahají do registrů)
- 25 částečně překrytých 32 bitových registrů, (dvě banky registrů, lze mezi nimi přepínat)
- možnost podmíněného vykonání instrukcí
- jednoduchý a výkonný instrukční soubor s pevně danou šířkou instrukcí (44 instrukcí, 32 bitů)
- dva typy přerušení - FIQ (fast interrupt request), který je pro neodkladné události, a IRQ (interrupt request), který se používá pro události nevyžadující extrémně krátkou dobu odezvy
- 3 stupňové zřetězení instrukcí
- 4 režimy (uživatelský, supervizor, IRQ a FIQ)
- aktuální režim je určen dvěma bity programového slova (PSW - program status word)

ARM Cortex-A8

- 32 bitový jednojádrový procesor, používá instrukční sadu ARMv7 (plus Thumb2, Jazelle, ...)
- Obsahuje 32 KiB L1 data cache a 32 KiB L1 instruction cache, dále také 512 KiB L2 cache. Frekvence je 600 MHz - 1 GHz. Používá třináctistupňový pipeline a výkonnou predikci skoků s úspěšností nad 95 %.
- The optional *NEON Media Processing Engine* (MPE) is the ARM Advanced Single Instruction Multiple Data (SIMD) media processing engine extension to the ARMv7-A architecture. It provides support for integer and floating-point vector operations.



MIPS

(microprocessor without interlocked pipeline stages)

- používá non-interlocked pipeline → programátor/překladač se musí starat o to, aby paralelně probíhající zřetěžené instrukce nevyužívaly stejných prostředků počítače
- jsou navrženy podle RISC filozofie
- přístup do paměti pouze pomocí instrukcí LOAD/STORE (ostatní instrukce šahají do registrů)
- obsahují 32 registrů, každá instrukce má délku 32 bitů
- používají se např. v herních konzolách, set-top boxech, mobilech a tiskárnách
- nejprve byl navržen kompilátor, až poté implementován obvod, s důrazem na rychlost

Rodina P6

(Pentium Pro, Pentium II, Celeron, Pentium III)

- spekulativní vykonávání instrukcí (out-of-order execution), instrukce jsou ukládány do instruction poolu (až 40 instrukcí v Pentiu Pro), 4 pipeliney
- FPU integrován na čipu
- 2 čipy - v jednom je jádro a L1 cache, v druhém L2 cache

BIU (bus interface unit)

- stará se o komunikaci procesoru se systémovou sběrnicí

Paměťový subsystém

- hlavní systémová paměť, L1 cache (datová a instrukční, interní sběrnice), L2 cache (64 bitová externí sběrnice)

Fetch/Decode unit

- načítá x86 CISC instrukce a dekóduje je na sekvenční RISC mikrooperace, který se ukládá do instrukčního poolu
- obsahuje tři dekodéry (dva pro jednoduché instrukce, jeden pro složité instrukce)
- obsahuje predikci skoků (buffer skoků)

Dispatch/execute unit

- dvě ALU, jedna FPU, dvě jednotky generování adres (AGU), dvě FPU-SIMD jednotky, memory interface unit (jednotka pro přístup do paměti - stará se o mikrooperace čtení a zápisu)
- vybírá si z instrukčního poolu mikrooperace v optimálním pořadí (podle datových závislostí, a dostupných prostředků - paměť)
- po spekulativním provedení je instrukce uložena zpět do instrukčního poolu

Retire unit

- vybírá provedené mikrooperace z instrukčního poolu

Pentium Pro

Přímo na procesoru je implementována paměť cache L2.

Pro lepší výkon bylo třeba RISC.

Kvůli kompatibilitě se staršími programy Fetch/Decode jednotka vybírá z paměti instrukce x86 a dekóduje je na 118 bitové RISC instrukce, které Intel ve své terminologii označuje jako mikrooperace. Za touto jednotkou už může následovat plnohodnotný RISC procesor.

Mikrooperace z dekódovací jednotky se však neřadí do fronty.

Instrukce jsou ukládány do banky dekódovaných instrukcí, označené na obrázku jako Instruction Pool. Z této banky instrukcí, až 40 mikrooperací, si může prováděcí jednotka

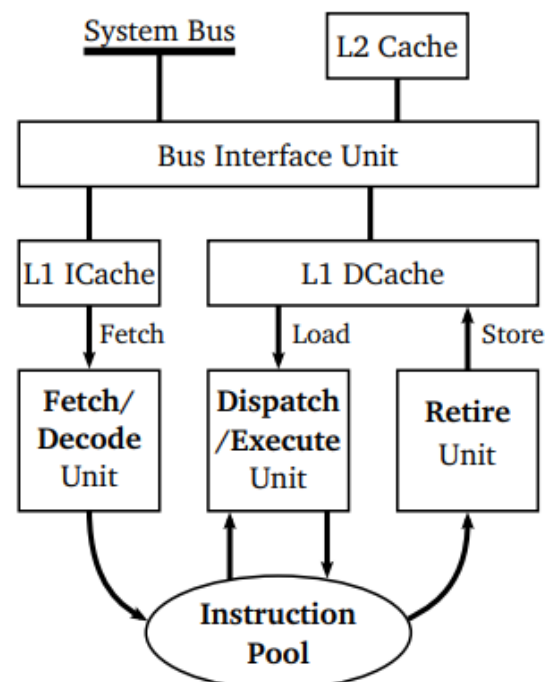
Dispatch/Execute vybírat instrukce mimo pořadí - out-of-order.

Provedené instrukce jsou z prováděcí jednotky uloženy zpět do banky instrukcí, odkud jsou vybírány ukončovací jednotkou Retire Unit. Odtud plynou data zpět do registrů a paměti cache L1.

Dále pak přes rozhraní sběrnice do cache L2 a do hlavní paměti.

Dekódovací jednotka může vyprodukovat až 4 mikrooperace v jediném cyklu. Jedině banka instrukcí zajistí, že si prováděcí jednotka bude umět sama rozhodovat o pořadí provádění instrukcí, aby maximálně využila paralelní činnost všech svých obvodů.

Pentium Pro Block Diagram



Intel Core Duo

- architektura vychází z Pentia Pro a Pentia M, vyrábí se od roku 2006
- datová sběrnice má 64 bitů, adresní sběrnice 36 bitů, technologie výroby je BiCMOS (65 nm)
- obsahuje tři celočíselné jednotky (ALU - arithmetic logic unit), které jsou 64bitové a plně podporují instrukční sady AMD64 (EM64T/x86-64), lze provést 64bit instrukci za jeden hodinový cyklus
- obsahuje dvě 128bitové FPU (floating point unit) s podporou SSE

(SSE je unita, pracující hlavně s obrázky)

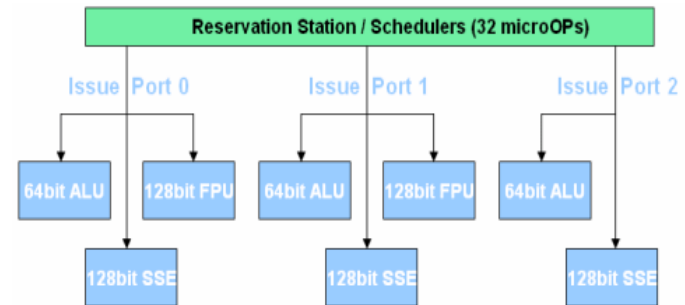
- obsahuje tři celočíselné 128bitové pro SSE (streaming single-instruction multiple-data extensions) operace
- tyto části jsou rozděleny do tří issue portů

0 - komplexní ALU, FPU/SSE a celočíselná SSE

1 - jednoduchá ALU, FPU/SSE a celočíselná SSE

2 - jednoduchá ALU a celočíselná SSE

- dále obsahuje tři issue porty pro práci s pamětí (načítání dat, ukládání dat a načítání adres), které pracují s TLB (translation look-aside bufferem) o kapacitě 256 záznamů



TLB - vyrovnávací paměť pro adresy převedené z virtuálního do fyzického adresního prostoru

- procesor obsahuje obvykle dvě jádra, každé jádro má k dispozici L1 cache (pro data 32 KiB, pro instrukce 32 KiB), dále procesor obsahuje L2 paměť pro data i instrukce, kterou sdílí všechna jádra (velikost několik MB)
- mezi L1 a L2 cachí platí non-inclusive (paměť v L1 může být v L2) a non-exclusive princip (paměť z L1 nemusí být celá v L2)
- stejně jako předchozí architektury se zde používá vykonávání instrukcí mimo pořadí (out-of-order execution), kdy procesor nejprve seřadí instrukce do optimálního pořadí (samozřejmě tak, aby nevznikaly hazardy souvislé se změnou pořadí), poté je vykoná a následně je opět seřadí do původního pořadí
- z bufferu mikrooperací jdou mikrooperace (až 4 za strojový cyklus) do tabulky přiřazení registrů (zde probíhá přejmenování registrů - po sobě jdoucí instrukce často vyžadují stejný registr, což zabraňuje paralelismu, proto se registry "přejmenují" a využije se nějaký jiný registr) a poté do reorder bufferu (kapacita 96 mikrooperací)

Dekodér

- převádí instrukce na microOPs (mikrooperace), obsahuje tři jednoduché dekodéry (pro základní operace - sčítání a podobné, 1 mikrooperace) a jeden komplexní (až 4 mikrooperace), ty jsou dohromady schopné předat do microOP bufferu až 7 mikrooperací za jeden takt

Macro-fusion

- sloučení dvou instrukcí v dekodéru na jednu, která kombinuje obě instrukce
- toto sloučení může provést libovolný ze čtyř dekodérů, v jednom taktu je tak možno zpracovat až 5 instrukcí
- např. často používaná kombinace porovnání (cmp) a skoku (např. jne) lze sloužit do jedné instrukce, cmpjne

Micro-fusion (sloučení mikrooperací)

- sloučení dvou mikrooperací do jedné

Loop detector

- speciální predikce skoku, která hledá podmínky ukončující cyklus
- např. do L2 si pak naháže věci, které by mohl potřebovat

Spekulativní předvykonávání (asymetrický multithreading, pomocná vlákna jsou napřed a vypočtou adresy a ládují do L2 data pro hlavní vlákno)

- asymetrický multithreading - kromě hlavního vlákna běží v procesoru i pomocná vlákna, která "pročítají" kód ještě před jeho vykonáním, vypočítávají adresy a načítají data z paměti RAM do L2 cache (prefetch)

Prefetch (do cache načítá okolní data, kdyby je náhodou needovala)

- pokud program narazí na požadavek o data, která nemá načtené v cache paměti, program se musí zastavit a načíst je z operační paměti, to program velmi zdržuje
- data se z paměti do cache načítají pomocí tzv. cache lines (obvykle 64 bytů), pokud tedy zažádáme o nějaká data, načtou se i okolní data z paměti, protože je velmi vysoká pravděpodobnost, že příští instrukce s nimi budou pracovat - zrychlení
- je zde několik hardwarových prefetcherů (RAM → L2, L2 → L1)

Paměti

(členění, typy)

Podle materiálu a fyzikálních principů

- magnetická paměť – založené na magnetických vlastnostech materiálu, informaci uchovává směr magnetizace.
- polovodičová paměť – využívá vlastností polovodičových tranzistorů, buď se realizují klopnými obvody (technologie TTL), nebo obnovováním elektrického náboje (CMOS)
- optická paměť – využívá optických vlastností materiálu, např. odraz světla.
- magnetooptická paměť – využívá změny orientace remanentní magnetické indukce po ohřevu materiálu
- feritová paměť – jako nosič jednoho bitu je používáno feritové jádro o rozměru cca 0,8 mm, magnetická orientace se překlápí proudovým impulsem (zastaralé)
- paměť se zpožďovací linkou – využívá pomalejšího průchodu vlny speciálním prostředím

Režim činnosti polovodičových pamětí

- dynamické – informace se musí periodicky obnovovat cyklem čtení, náročnější na řídicí logiku
- statické – informace zůstává uchována i bez obnovování, mají vyšší cenu za bit

Podle závislosti na napájení

- napěťově závislé (volatilní) – pro uchování a přístup k informacím potřebuje paměť napájecí napětí, při odpojení se informace ztrácí
- napěťově nezávislé (nevolatilní) – potřebuje napájení pro činnost (čtení, zápis), ale při odpojení napájení se informace uchová

Podle přístupu

- RAM (Random Access Memory) – s libovolným přístupem, doba přístupu k obsahu není závislá na umístění (adrese). Počítačové disky jsou považovány za paměti typu RAM, i když to není přesné.
- sekvenční – doba přístupu k obsahu je závislá na umístění, například páska
- asociativní – adresovaná obsahem, adresou je klíčová hodnota ukládaná s informací
- sériový – například fronta FIFO (to sem asi nepatří)

Podle schopnosti zápisu

- RWM (Read Write Memory) – Paměť pro zápis i čtení (Termín RAM obvykle označuje tento typ paměti - název RWM se neuchytil).
- ROM (Read Only Memory) – Paměť pouze pro čtení. Informace je do paměti uložena jednorázově při výrobním procesu.
- PROM (Programmable Read Only Memory) – Paměť se vyrobí bez informace a pomocí speciálního zařízení (programátor) si ji naprogramuje uživatel.
- EPROM (Eraseable Programmable Read Only Memory) – Paměť je možné vymazat speciálním způsobem (např. ultrafialovým zářením) a znovu přeprogramovat.
- WMM (Write Mostly Memory), někdy uváděna jako WOM (*Write Only Memory*) – Při provozu je používána jen pro zápis, informace je čtena jednorázově na konci provozního cyklu. Mívá speciální využití (černá skříňka).
- WOM (Write Only Memory) – Nerealizované nesmyslné zařízení, jež se stalo součástí inženýrského folklóru.
- EEPROM (E2PROM) (Electric Erasable PROM) – Obdoba EPROM, mazání však probíhá pomocí elektrického „impulsu“, maže se buňka po buňce. Počet zápisů je omezen – cca 1000 přepisů.
- Flash EPROM (Paměť EPROM s rychlým mazáním) – Obdoba EEPROM, mazání však probíhá po blocích buněk. Lze ji smazat pouze celou (1ms) nebo po částech – ne po jednotlivých buňkách. Počet zápisů je přibližně 100 000.

Všechny paměti xROM jsou statické a Non-Volatile – jednou zapsaná informace zůstává trvale uložena. Volatilita je schopnost paměťové buňky udržet si informaci i bez napájení.

Hierarchie

Podle toho, jestli je součástí přístroje anebo se k němu připojuje - kabelem, konektorem:


1. Vnitřní paměť (primární)

- **Akumulátor**
 - registr v procesoru o velikosti délky slova CPU (8, 16, 32, 64 bitů)
 - může být rychlejší než ostatní registry (kratší kód instrukcí)
 - s akumulátorem pracuje většina instrukcí (aritmetické a logické operace)
- **Registry procesoru**
 - několik (až desítky) registrů
 - součást procesoru
 - ukládání operandů a výsledků aritmetických a logických operací
 - nejrychlejší paměť připojená k procesoru (stejně rychlá, jako procesor)
- **Cache**
 - pro urychlení komunikace s pamětí
 - rychlá statická paměť
 - u novějších procesorů velikost stovky kB až MB
 - více úrovní, přičemž číslo určuje vzdálenost od procesoru
 - L1 – typicky přímo na procesoru
 - L2 – například na destičce s procesorem (tzv. boxované procesory)
 - L3 – na základní desce
 - write through – data se zapisují ihned (čeká se na dokončení zápisu)
 - write back – data se zapisují později (na dokončení zápisu se nečeká)
- **Operační paměť RAM**
 - pomalejší než procesor, rychlejší než vnější paměti
 - velikost desítky až stovky MB (až GB)
 - ve von Neumannově architektuře počítače použita pro program i pro data
 - typicky dynamická paměť

2. Vnější paměť

- **Sekundární paměti**
 - Pevný disk
 - je na nich systém souborů (struktura adresářů)
 - obsahuje obvykle statickou nebo dynamickou cache pro urychlení čtení/zápisu
- **Terciární paměti**
 - zařízení k zálohování dat
 - CD a DVD, Optické disky, ...

DRAM

(Dynamic Random Access Memory) **TO DO HISTORICKÝ VÝVOJ** 

- paměť je uložena ve formě náboje kondenzátoru (nabitý - 1, vybitý - 0)
- jelikož se kondenzátor samovolně vybíjí, je potřeba všechny buňky v pravidelném intervalu obnovovat (refresh), data jsou zapomenuta přibližně po 10 ms bez obnovení
- buňky jsou uloženy v matici, nejprve se zašle adresa řádku a poté adresa sloupce
- paměť může být organizována různě (např. 1 MiB = 1 milion po 1 bitu nebo 256k po 4 bitech)

Princip fungování

- 1) Adresní multiplexing - nejprve je zaslána adresa řádku (RAS - row address strobe, řídicí signál), potom sloupce (CAS - column address strobe), aktivovaný signál (RAS/CAS) poté předá adresu dekodéru řádků/sloupců
- 2) paměťová buňka poté předá data do I/O bufferu (data jsou zesílena čtecími zesilovači)
- 3) pokud mají být data zapsána, kontroler aktivuje WE (write enable), přesune data do I/O bufferu
- 4) přečtení jedné buňky vede k občerstvení celého řádku (řádek → I/O buffer → řádek), čtení je totiž destruktivní

Page Mode (1 řádek, 4 sloupce)

- do kontroleru se přivede adresa řádku a poté 4 adresy sloupců za sebou (předpokládá se stejný řádek, o 50 % menší přístupová doba)

Extended Data Out (Podržení sloupce dokud není nový)

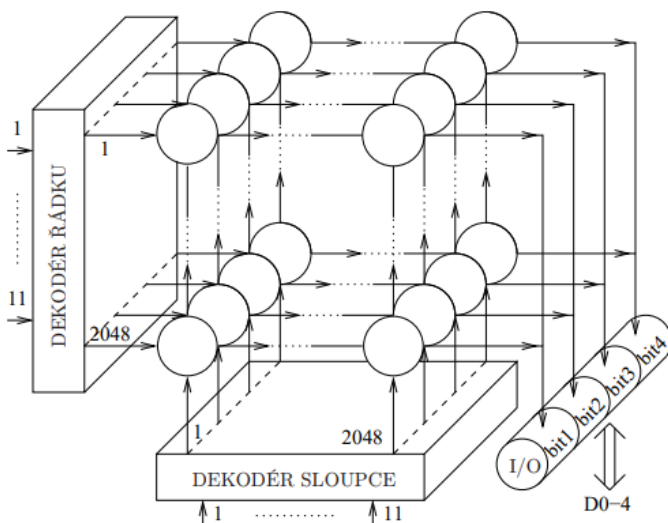
- časová vzdálenost mezi dvěma CAS aktivacemi je kratší (o 30 % rychlejší než PM), lze překrývat zadávání sloupce a výstup dat

SDRAM (synchronous DRAM)

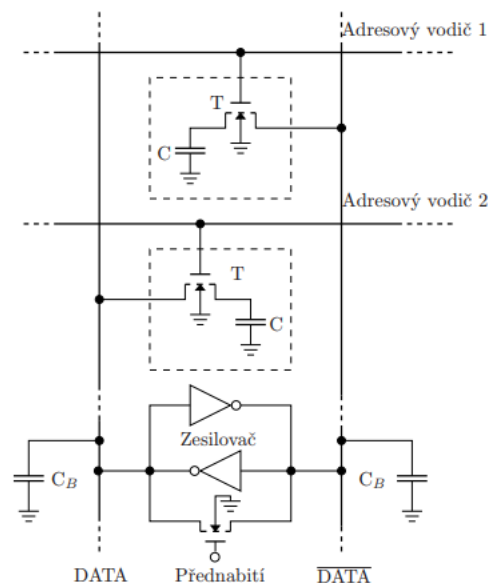
- malá přístupová doba, mohou fungovat synchronně se systémovou frekvencí
- pracuje v burst módu, zadá se adresa řádku, sloupce a poté se načítají data z přilehlých adres

DDR SDRAM (double data rate SDRAM)

- přenosová rychlost dat je zdvojnásobena, jelikož jsou data přenášena jak na sestupné, tak na náběžné hraně hodinového pulsu



Obrázek 1: Organizace paměti DRAM



Obrázek 2: Paměťová buňka DRAM s 1 tranzistorem

„Podle adresy, kterou poskytlo CPU přijme adresový buffer adresu paměti jako výstup externího paměťového kontroléru. Z tohoto důvodu je adresa rozdělena na dvě části, adresu řádku a adresu sloupce.

Tyto dvě adresy jsou čteny do adresového bufferu jedna za druhou. Tento proces se nazývá multiplexing. Důvod tohoto dělení je zřejmý: na adresování jedné buňky v 4 Mb čipu s 2048 řádky a 2048 sloupci by bylo třeba celkem 22 adresových bitů (11 pro řádek a 11 pro sloupec). Pokud by měly být přesunuty všechny adresové bity najednou, bylo by třeba 22 adresových vývodů. Potom by musel být pouzdro čipu velmi velký. Proto je lepší přesunout adresu paměti ve dvou částech.

Obvykle adresový buffer nejprve čte adresu řádku a potom adresu sloupce. Tento adresní multiplexing je kontrolován RAS (Row Address Strobe) a CAS (Column Address Strobe) řídicími signály. Pokud paměťový kontrolér pošle adresu řádku, 4 tak zároveň aktivuje RAS signál. RAS informuje čip DRAM, že dodaná adresa je adresa řádku. Nyní kontrolér DRAM aktivuje adresový buffer k získání adresy a přesune ji do dekodéru řádku, který ji dekoduje. Pokud později paměťový kontrolér poskytne adresu sloupce, potom aktivuje CAS signál. Tak kontrolér DRAM pozná, že tentokrát je přesunována adresa sloupce a aktivuje znovu adresový buffer. Adresový buffer přijme poskytnutou adresu a přesune ji do dekodéru sloupce.

Paměťová buňka adresovaná tímto způsobem předá na výstup uložená data, která jsou zesílena čtecími zesilovači a přesunuta do I/O bufferu. Buffer nakonec poskytne informace jako výstupní data Dout přes datové vývody paměťového čipu.

Pokud mají být data zapsána, paměťový kontrolér aktivuje WE (Write Enable) signál a přesune zapisovaná data Din do I/O bufferu. Pomocí čtecích zesilovačů je informace zesílena, přesunuta do adresované paměťové buňky a v ní uložena.“ - skripta

Paměťový kontrolér počítače tedy řeší 3 různé úkoly: rozdělení adresy získané z CPU na adresu řádku a sloupce, které jsou přesunuty do paměti jedna po druhé; správně aktivuje RAS, CAS, WE a READ signály; přesune uložená data a přijímá data k zápsání do paměti. Neupravené adresové a datové signály z CPU nejsou vhodné pro paměť, proto je paměťový kontrolér nezbytnou součástí počítačového paměťového subsystému.

SRAM

(Static Random Access Memory)

- informace je uložena stavem klopného obvodu, typicky 2 NMOS přístupové, 2 NMOS paměťové tranzistory a 2 zátěžové prvky (rezistory nebo tranzistory)
- dražší než RAM, pojme méně paměti, ale je rychlejší (není potřeba je obnovovat, snažší čtení)
- používá se hlavně u cache pamětí
- paměťové buňky jsou uspořádány do matice
- chybí adresní multiplexing (adresa řádku i sloupce je zadávána současně - potřeba více pinů)

Princip fungovania

- 1) Adresa je rozdělena na řádek a sloupec
 - 2) Je aktivován adresový vodič, v odpovídajícím sloupci se informace z vodiče DATA přesune do I/O bufferu
- při zápisu je proces opačný

Asynchronní SRAM

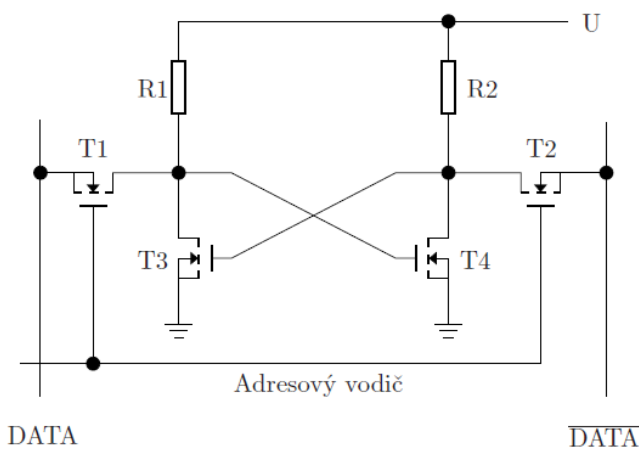
- není synchronizována se systémovými hodinami, procesor musí čekat na data

Synchronní burst SRAM

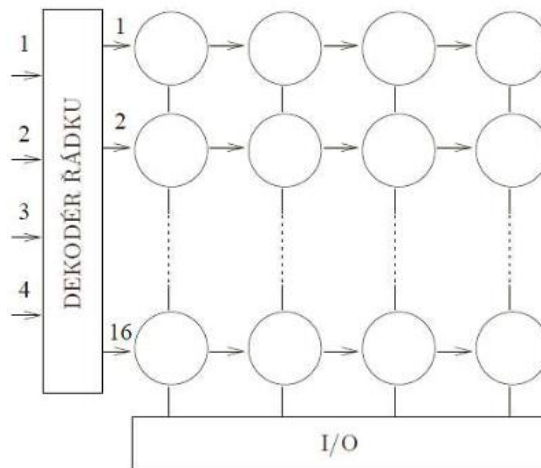
- je synchronizovaná se systémovými hodinami, maximálně ale s frekvencí 66 MHz

Pipeline burst SRAM

- požadavky jsou zřetězeny, je navržena pro frekvence 75 MHz a vyšší



Obrázek 10: Paměťová buňka SRAM se 4 tranzistory



Obrázek 9: Organizace paměti SRAM

„Při čtení dat z paměťové buňky SRAM aktivuje dekodér řádku odpovídající adresový vodič. Dva přístupové tranzistory T1 a T2 se zapnou a propojí klopný obvod paměti s datovými vodiči DATA, DATA. SRAM čip neprovádí multiplexing adres řádku a sloupce, signály adres řádku a sloupce jsou tedy poskytnuty současně. Dekodér adresy SRAM rozdělí adresu na řádek a sloupec. Po stabilizaci dat vybere dekodér sloupce odpovídající sloupec (to jest odpovídající datové vodiče DATA, DATA) a předá na výstupu data do I/O bufferu a tím do vnějších obvodů. Zápis dat probíhá opačným způsobem. Přes vstupní datový buffer a dekodér sloupce se zapisovaná data zavedou na odpovídající čtecí zesilovač. Ve stejnou dobu aktivuje dekodér řádku adresní vodič a zapne přístupový tranzistor T1. Stejně jako v procesu čtení dat se klopný obvod snaží předat uložená data na datové vodiče DATA, DATA. Nicméně čtecí zesilovač je silnější než paměťový tranzistor T3 a poskytne datovým vodičům DATA, DATA signál, který odpovídá zapisovaným datům. Proto se klopný obvod přepne podle nově zapisovaných dat nebo si udržuje již uloženou hodnotu v závislosti na tom, zda se zapisovaná data shodují s uloženými daty nebo ne.“ - skriptá

Nevolatilní paměti

ROM

- len raz naprogramovatelná paměť, dá sa len čítať

PROM

- užívateľ pomocí programátoru vypálí informácie do paměti (přepálení pojistky)

EPROM

- informace se uchovává pomocí elektrického náboje
- pro naprogramování je potřeba puls o délce asi 50 ms (5 V)
- je možné ji vymazať pomocí ultrafialového záření (proto obsahuje okénko na pouzdře)

EEPROM

- lze mazat stejně jako programovat, elektrickým pulsem

Flash

- doba pamatování informace je 10 - 100 let
- podobná struktura jako u EEPROMu, nevykonává se zde adresní multiplexing
- je možné zapsat 0, není možné zapsat 1 (data ze sektoru se musí načíst do paměti, tam se změní potřebné bity na 1 a poté se zapíše sektor zpět na flash paměť)
- data jsou rozdělená do sektorů, na rozdíl od EEPROM nemusí být vymazána celá paměť před přepsáním, stačí smazat blok (sektor).

Přerušení

(Interrupt)

- odezva procesoru na požadavek o programové obslužení (nějaké události)
- zařízení žádá o obsluhu kanálem, který se přímo anebo pomocí specializovaného obvodu přivádí na vstup procesoru (INTn, INTR)

- 1) Mikroprocesor dokončí právě zpracovávanou instrukci
- 2) Dočasně vypne ostatní žádosti o přerušení (žádosti s vyšší prioritou toto můžou obejít)
- 3) Procesor převezme vektor přerušení, obsahující adresu podprogramu v paměti a spustí ho
- 4) Procesor se vrátí na původní místo vykonávání a pokračuje v programu

Obsluha priority přerušení a identifikace žadatelů o přerušení

Programová identifikace

- při přerušení se vyvolá program, který přejde do odpovídající větve podle toho, jaké zařízení žádá o přerušení (logický součet signálů přerušení)
- nenáročný způsob, ale pomalý a klade nároky na program

Sériová obvodová identifikace

- procesor očekává po přijetí přerušení identifikační znak, který je generován stykovými obvody v pořadí, které určuje prioritu
- procesor po přijetí instrukce vyše potvrzení po sběrnici, zařízení jej buď přepošle dál anebo vrátí vektor přerušení s adresou programu (tím potvrdí, že o přerušení žádalo, priorita je daná pořadím zařízení na sběrnici)

Řadič přerušení

- specializovaný pomocný obvod, který identifikuje zařízení žádající o přerušení a předává přerušení procesoru (pokud je jeho priorita vyšší než priorita požadavku, který procesor právě obsluhuje)

Programové čítače

- pomáhají řešit problémy časových intervalů a počítání událostí
- po uplynutí určitého počtu impulzů anebo intervalu se vyvolá přerušení procesoru
- čítače (registry) se obvykle dekrementují od zadané hodnoty do nuly (snažší na přepočet)
- čítač je inkrementován externí událostí, časovač vnitřním signálem procesoru
- lze nastavit předděličku před časovačem pro přizpůsobení chování časovače

DMA

(Direct Memory Access)

- způsob přímého přístupu dat periférií do operační paměti počítače (bez přímé účasti CPU)
- procesor programuje řadič DMA a uvolňuje sběrnici (vysokoimpedanční stav)

Způsoby řízení

- 1) Řadič DMA vyšle procesoru HOLD signál, čímž uvolní sběrnici, poté řadič přenese data a řízení se vrací procesoru - toto zpomaluje program běžící na procesoru
- 2) Po ukončení strojového cyklu se vypne hodinový signál procesoru a odpojí se jeho budiče sběrnice, obvykle na jeden dílčí přenos dat - zpomaluje program
- 3) Jednotlivé přenosy se uskutečňují v době, kdy procesor pracuje, ale nepoužívá sběrnici - nezpomaluje program, ale DMA a CPU musí pracovat synchronně

Registr dat - obsahuje slovo, které má být přesunuto z periferie do paměti nebo naopak

Registr adresy - obsahuje adresu paměti, do které se mají přesouvat data z periferie

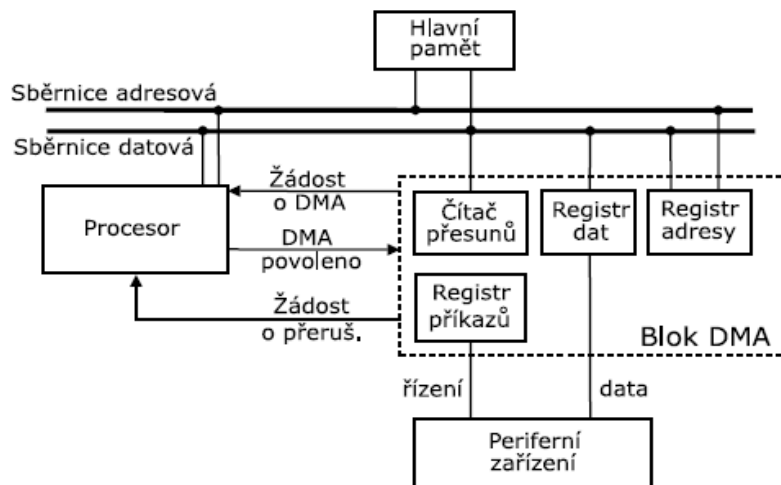
Čítač přesunů - uchovává informaci, kolik přesunů zbývá

Průběh

- 1) Procesor naprogramuje řadič DMA
- 2) Blok DMA spustí periférii a čeká, až bude připraveno vyslat/přijmout data
- 3) Procesor ve vhodnou chvíli (po dokončení strojového cyklu) zareaguje na DMA požadavek
- 4) a) Procesor pošle jednotce DMA informaci o povolení přenosu a uvolní sběrnici
 - b) DMA pošle adresu na adresovou sběrnici, data na datovou sběrnici a čeká na přenos
 - c) DMA inkrementuje registr adresy a dekrementuje čítač přesunů
 - d) Pokud není čítač přesunů nulový, DMA zkontroluje, zda-li má zařízení připravená další data, pokud ano, pokračuje znovu od b), pokud ne, předá řízení zpět procesoru

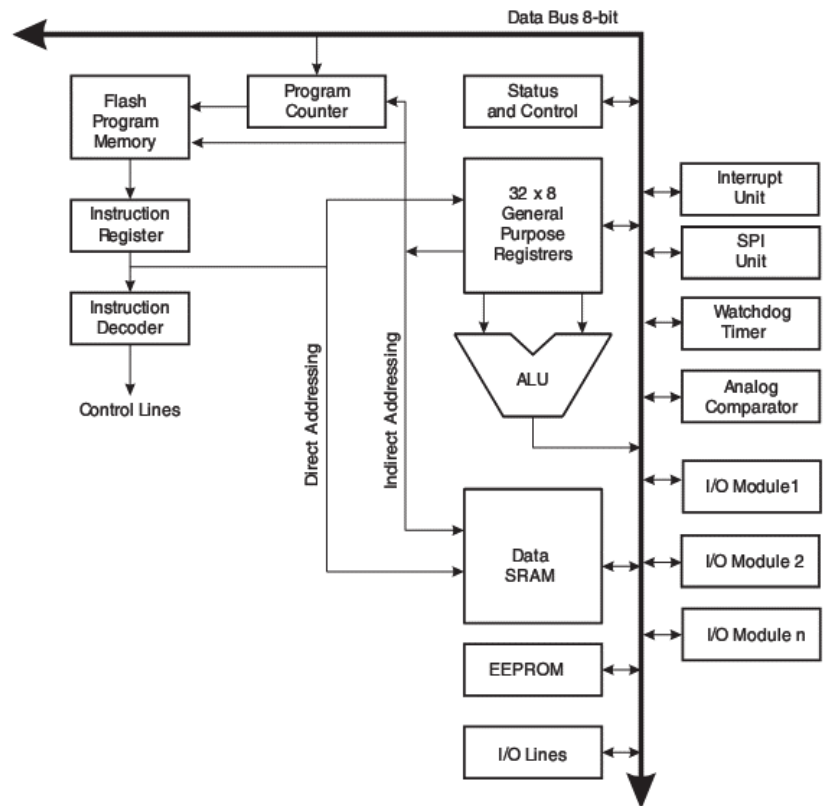
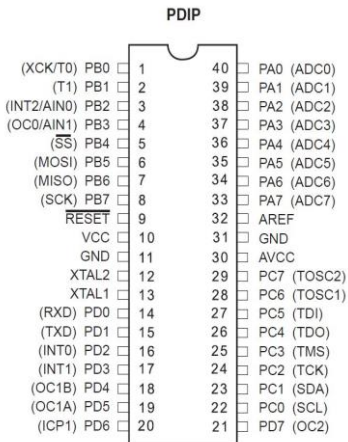
Kanál

- speciální přenosové procesory plně se starající o I/O operace mezi CPU a perifériemi
- řídí se kanálovým programem (multiplexní, selektivní)



ATmega32

- 8bitový nízkonapěťový procesor
- 40 pinů (dva napájecí, dva pro oscilátor, jeden analogově-digitálnímu převodníku a 32 pro libo
- piny umí fungovat v analogovém módu
- 3 vnitřní timery
- aproximační ADC (analog-to-digital converter)
- USART (universal synchronous-asynchronous)



- 32 KiB Flash programovatelné paměti, 1 KiB I
- frekvence 1 až 16 MHz
- lze programovat pomocí SPI, paralelně anebo pomocí JTAGu (Joint Test Action Group, slouží k testování plošných spojů a interních obvodů procesoru) - lze nastavit fuse bity

GPU

(Graphics Processing Unit)

- slouží k zobrazení dat dodaných procesorem, propočítává body obrazu a ukládá je do videopaměti, ulehčuje práci CPU, jsou specializovány na grafické výpočty (výpočty geometrie, zpracování rastrových dat, práce s texturami atd.)

Paměti

- **DRAM** (dynamic RAM) - jedna V/V brána pro čtení i zápis, jako u CPU
- **VRAM** (video ram) - jedna brána pro čtení, druhá pro zápis, urychlení přenosu
- **WRAM** (window ram) - stejné jako VRAM + podpora 3D transformací a zpracování videa

Obnovovací frekvence - počet snímků překreslených na monitoru za sekundu

Horizontální rozkladová frekvence - počet řádků vykreslených za sekundu

RAMDAC - musí být dostatečně rychlý, aby zvládal konvertovat data z digitální na analogovou reprezentaci, minimální rychlost se odvozuje od obnovovací frekvence, rozlišení a dodatečné kapacity pro řídicí signály (30 %)

- obsahuje malou SRAM paměť pro barevnou paletu
- každý barevný kanál má svůj vlastní převodník (R, G, B)

CUDA

(Compute Unified Device Architecture)

- unifikovaná architektura pro (obecné) výpočty na grafických kartách
- není potřeba znát architekturu konkrétní grafické karty, problémy se nemusí definovat v jazyce grafiky (texture), odpadá nutnost znalosti DirectX, OpenGL
- funguje pouze na kartách od Nvidie
- poskytuje programovací rozhraní pro často používané jazyky - C/C++, Fortran, Python
- optimalizováno na matematicky náročné výpočty (zpracování obrazu, problémy definované pomocí mřížek), lze použít větvení, ale výkonost programu je jím degradována
- grafické karty obsahují velké množství ALU a malé množství řídicích obvodů
- na kartě je několik multiprocesorů (SM), každý má sdílenou paměť (+ cache na konstanty a texture), několik procesorů (každý s vlastními registry) a instrukční jednotku, která rozděluje práci procesorům
- všechny multiprocesory mají přístup ke globální, konstantní a texturové paměti

Typická operace SIMD

- 1) Zkopírování dat z CPU na GPU
- 2) Spuštění vláken na GPU
- 3) Provedení výpočtu na GPU
- 4) Zkopírování výsledku z GPU na CPU

Vlákna

- vlákna musí na sobě být nezávislé, nelze zaručit pořadí jejich vykonávání
- vlákna jsou organizovány do bloku, bloky jsou dále také organizovány do mřížky (grid), lze rozšířit i do třetí dimenze
- skupiny vláken (warpy) jsou opakovaně spouštěny schedulerem a přepínají se mezi sebou (masivní paralelismus)

CUDA jádro

(Compute Unified Device Architecture)

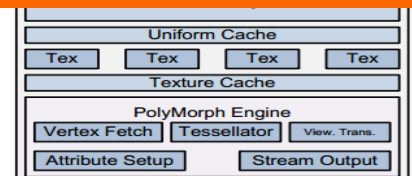
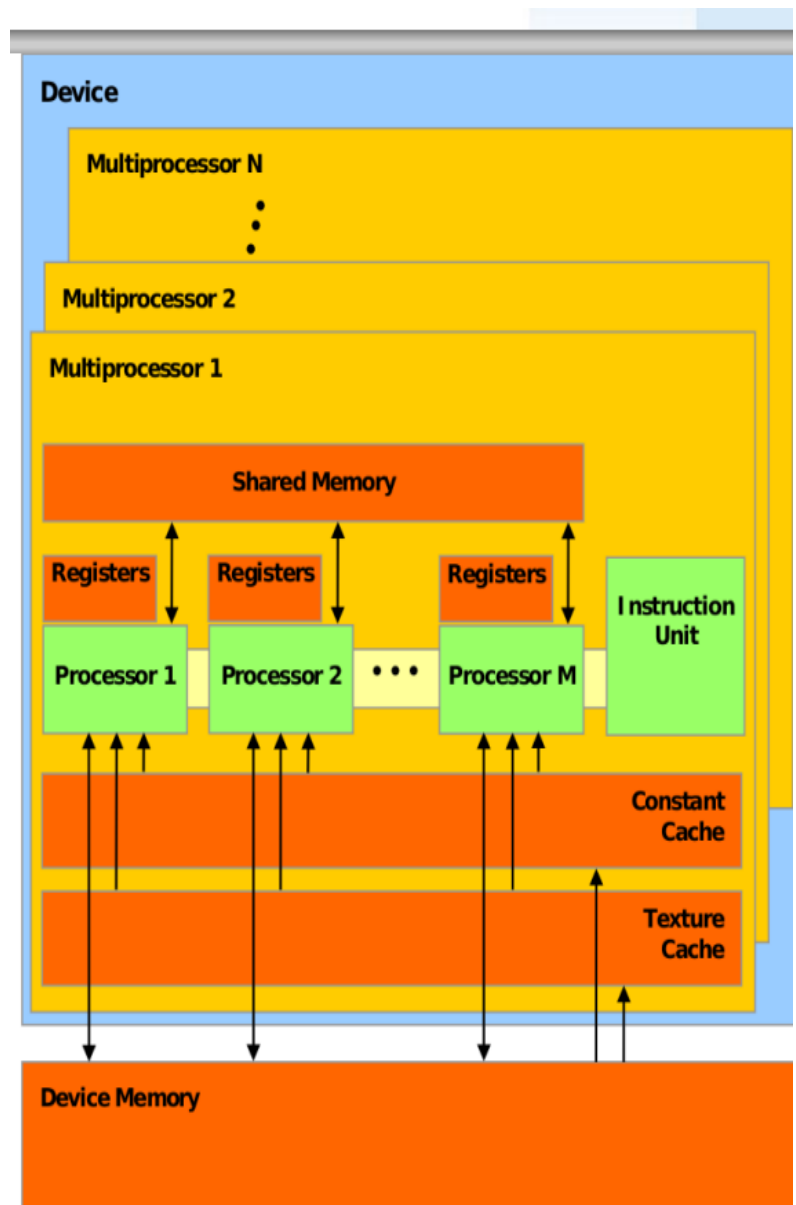
- obsahuje jednu FPU a jednu ALU jednotku (integer unit)

Rozšíření C/C++

- používá se nádstavba nad C/C++ kompilátorem NVCC
- kvantifikátory funkcí: `__global__`, `__device__`, `__host__`
- kvantifikátory proměnných: `__device__`, `__constant__`, `__shared__`
- spouštění kernelů:
`fn<<<pocet_bloku,pocet_vlaken>>>(params)`

Knihovny pro C/C++

- vektorové datové typy (struktury): `intx`, `charx`, `ucharx`, ... (kde x je číslo 1-4)
- metody pro komunikaci s grafickou kartou (`cuda***`)



1 kernel kernel funkcia je spustená na 1 gride. Na gride sú bloky, do ktorých sú sústredené jadrá.

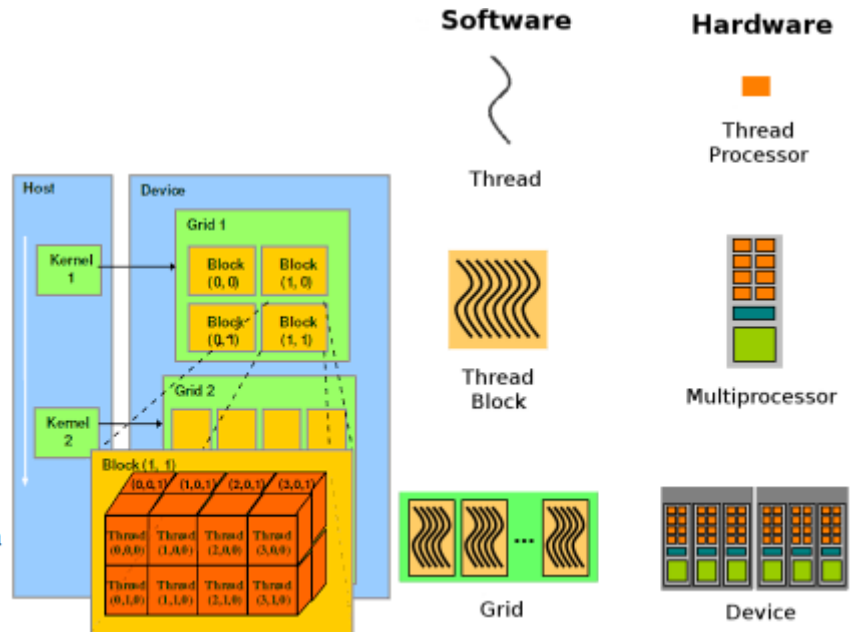
Bloky – multiprocesory používame pre jednoduchšiu synchronizáciu threadov. Pri spúšťaní kernelu zadávame počet blokov a počet threadov pre blok.

- ▶ **CUDA host** je CPU a operačná pamäť
- ▶ **CUDA device** je zariadenie pro paralelné spracovanie až stoviek tisíc nezávislých vlákien - threads
- ▶ **CUDA thread** je veľmi jednoduchá štruktúra - rýchle sa vytvára a rýchle sa prepína pri spracovaní
- ▶ komunikácia medzi výpočtovými jednotkami je hlavný problém v paralelnom spracovaní dat
- ▶ nemôžeme očakávať, že budeme schopní efektívne synchronizovať tisíce vlákien
- ▶ CUDA architektúra zavádza menšie skupiny vlákien zvané bloky - **blocks**
- ▶ programovanie v CUDA spočíva v písaní kernelů - **kernels**
 - ▶ kód spracovaný jedným vláknem
- ▶ kernely nepodporujú rekurziu
- ▶ podporujú vetvení kódu, ale to môže snižovať efektívnosť
- ▶ nemohou vrátiť žiadny výsledok
- ▶ jejich parametry nemohou byť reference
- ▶ podporujú šablony C++
- ▶ **od CUDA 2.0 podporujú funkciu `printf` !!!**
- ▶ jeden blok je spracovaný na jednom multiprocesore
- ▶ vlákna v jednom bloku sdieľajú veľmi rýchlu pamäť s krátkou latenciou
- ▶ vlákna v jednom bloku môžu byť **synchronizovaná**
- ▶ v jednom bloku môže byť až 1024 vlákien
 - ▶ multiprocesor prepína medzi jednotlivými vlákny
 - ▶ tým zakrýva latencie pomalé globálnej pamäte
 - ▶ spracováva vždy tá vlákna, ktorá majú načítané potrebné dáta, ostatní načítajú

Bloky vlákien jsou seskupeny do gridu - **grid**.

Pro získání efektivního kódu je nutné dodržet následující pravidla:

- ▶ redukovat přenos dat mezi CPU (CUDA host) a GPU (CUDA device)
- ▶ optimalizovat přístup do globální paměti
- ▶ omezit divergentní vlákna
- ▶ zvolit správnou velikost bloků



I²C

(Inter-Integrated Circuit)

- dvoudrátová, dvou vodičová sběrnice se sériovým přenosem
- obsahuje slave a master obvody
- lze propojit až 128 zařízení

Adresa zařízení

- skládá se ze 7 bitů (horní 4 určuje výrobce, dolní 3 jdou nastavit libovolně)

Signály - SCL (synchronous clock), SDA (synchronous data)

Komunikace

- v klidovém stavu je SCL

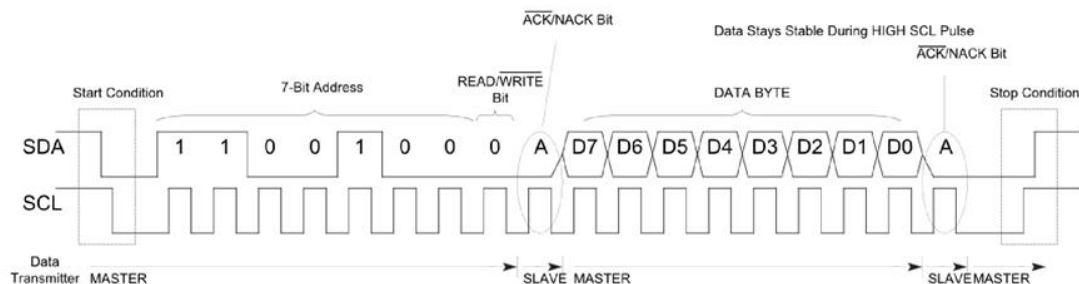
i SDA na 1

Start: SDA ↓, SCL ↓

Konec: SCL ↑, SDA ↑

Náběžná hrana

- "vezmi si data" (write)



Sestupná hrana

- "změň data" (read)

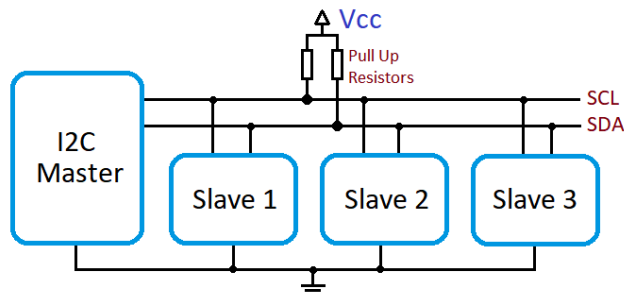
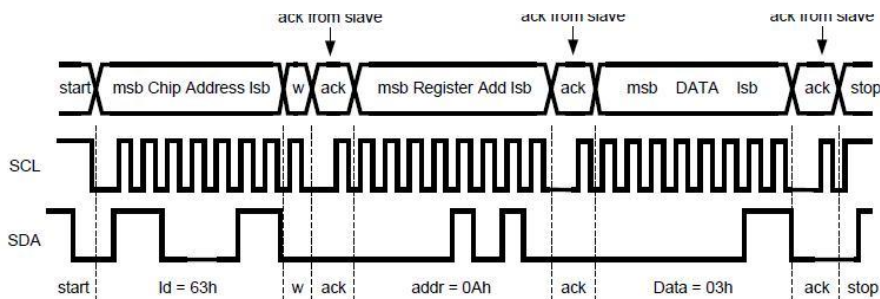
- po startu následuje adresa cílového zařízení (slave)

- 0bAAAAAAM
- A - adresa (7 bitů)
- M - Read (1), Write (0) - z pohledu mastera

- dále slave vygeneruje ACK

- pokud je ACK 0, je všechno v pořádku
- pokud je ACK 1, slave nereaguje (klidový stav)

- na konci komunikace při čtení se posílá NACK (1)



Každému přenosu předchází vyslání podmínky START. Potom je vysílána 7 bitová adresa příjemce a jeden bit R/W, který indikuje požadovanou operaci (čtení/zápis). Další bit ACK je vyslán s úrovní H a je určen k potvrzení přijímací stanice o připravenosti přijímat. Dále jsou přenášena data ve směru určeném předchozím bitem R/W. Každý byte je následován jedním bitem ACK. Po ukončení přenosu je vyslána podmínka STOP.

Každá stanice připojená na I²C má přidělenou 7 bitovou adresu. Po zachycení podmínky START, porovnávají všechny obvody svou adresu s adresou, která je vysílána na sběrnici. Zjistí-li některý z obvodů shodu, je vysílání určeno právě jemu a musí přijetí adresy potvrdit bitem ACK. Potom přijímá nebo vysílá další data. Několik adres je na I²C vyhrazeno pro speciální účely. Například adresa 0000000 je určena pro vysílání broadcast adresy 0000011, 00001XX a 11111XX jsou rezervovány pro další účely. Každý vysílaný byte a vyslaná adresa je následována vysláním jednoho bitu ACK. Vysílající stanice jej vysílá v úrovni H. Přijímající stanice potvrzuje přijetí tím, že v době vysílání ACK připojí SDA na úroveň L. Pokud vysílající stanice nedostane potvrzení příjmu, ukončí vysílání podmínkou STOP.

CRT

(Cathode Ray Tube)

- vzduchoprázdňá skleněná buňka, její přední část tvoří luminiscenční látka (luminofor)

Princip

- elektronové dělo "vystřeluje" (zahřátím katody) proudy elektronů (pro každou barvu existuje právě jedno dělo)
- elektronové svazky dále poté prochází Wheneltovým válcem (řízením napětí na válci se kontroluje intenzita elektronů - některé se odkloní k anodě v horní části trubice)
- elektrony dále procházejí přes mřížky, které je zaostří
- k setkání jednotlivých svazků dojde u masky, která propustí jen část svazků a dopadne na luminofor
- elektronové svazky jsou vychylovány pomocí cívek horizontálně i vertikálně
- proces začíná v levém horním rohu, směrem k pravému hornímu rohu – pak se sníží řádek (rastrování/řádkování)
- obnovovací frekvence je asi 60 Hz (komfortní 72Hz, ideální 80 - 100Hz)
- maska je zakulacená kvůli lepšímu zaostření paprsků

Vlastnosti

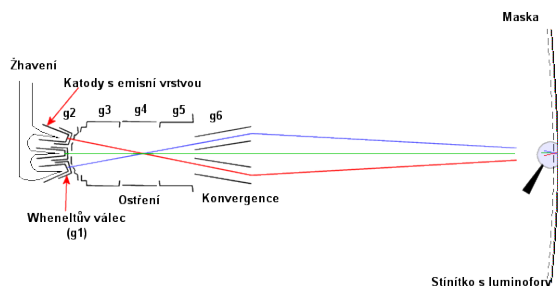
- uhlopříčka, rozlišení, horizontální a vertikální frekvence

Výhody

- ostrost, věrné barvy, doba odezvy, pozorovací úhly

Nevýhody

- velikost, vyzařování, spotřeba



Plazmové monitory

Plazma - skupenství iontů a elementárních částic (čtvrté skupenství)

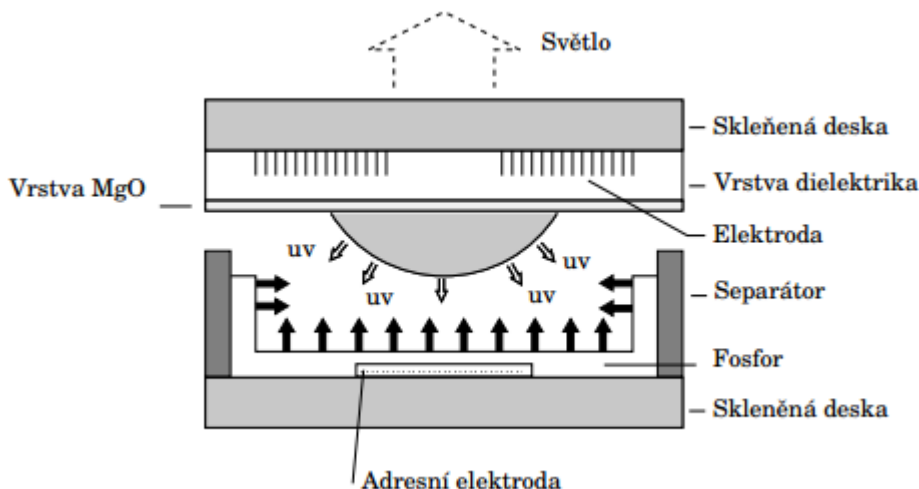
- v klidovém stavu se nachází v displeji směs vzácných plynů (*argon, neon, xenon*)
- přivedením elektrického proudu do plynu vzniknou volné elektrony, srážky mezi nimi a částicemi plynu vyvolají vznik kladně nabitých iontů, vznikne tedy plazma
- nabité částice se začnou pohybovat k opačným pólům (ionty k zápornému, elektrony ke kladnému), při nárazech elektronů dochází k uvolňování fotonů
- displej je tvořen maticí miniaturních buněk ovládaných sítí elektrod, horizontální řádky tvoří adresovací elektrody, vertikální sloupce zobrazovací (výbojové) elektrody
- buňky jsou uzavřeny do dvou skleněných tabulek, každá obsahuje kondenzátor a tři elektrody
- pixely se skládají ze tří sub-pixelů (červený, zelený, modrý)
- ovládání intenzity světla se používá PWM, snímky jsou rozděleny na pod-snímky, během nich je vytvářeno světlo podle přijatých pulzů

Výhody

- kvalitní a kontrastní obraz, není nutné podsvícení, velké pozorovací úhly, minimální hloubka a hmotnost

Nevýhody

- paměťový efekt (obraz se může na displeji vypálit - nevhodné pro počítače), cena, energetická náročnost, levnější displeje mají problémy s kontrastem



LCD

(liquid crystal display)

- jádrem je TN (twisted nematic) struktura, která je ze dvou stran ohraničena skleněnými deskami s polarizačními filtry

Princip činnosti

- 1) nepolarizované světlo prochází polarizačním filtrem, který ho otočí o 90°
 - 2) polarizované světlo prochází strukturou tekutých krystalů, které světlo otočí o 90°
 - 3) světlo projde druhým polarizačním filtrem (pootočený o 90° oproti prvnímu)
- ve výchozím stavu tedy displej svítí (propouští světlo)
 - elektrickým proudem můžeme kontrolovat orientaci krystalů, což následně způsobí, že část světla bude polarizována odlišně a neprojde druhým filtrem (proud určuje intenzitu světla)
 - vrstva krystalů je rozdělena na malé buňky stejné velikosti tvořící jednotlivé pixely monitoru
 - je nutné displej podsvětlovat bílým světlem (elektroluminiscenční výbojky, nověji LEDky)



Barevné LSD

- každý bod je rozdělen na tři části (subpixely), červenou, zelenou a modrou, sloužící jako filtry, které skládají výslednou barvu

Pasivní displeje

- obsahují mřížku vodičů, body se nacházejí na průsečících mřížky
- při vyšším počtu bodů narůstá potřebné napětí → rozostřený obraz, velká doba odezvy (3 FPS) - nevhodné pro hry, filmy, televizi atd.
- používá se v zařízeních s malým displejem (hodinky)

Aktivní displeje (TFT - thin film transistor)

- každý průsečík v matici obsahuje tranzistor nebo diodu, která řídí činnost daného bodu, je možné velmi přesně a rychle ovládat svítivost každého bodu
- šířka subpixelů je zhruba 0.2 - 0.3 mm

IPS (In-plane switching)

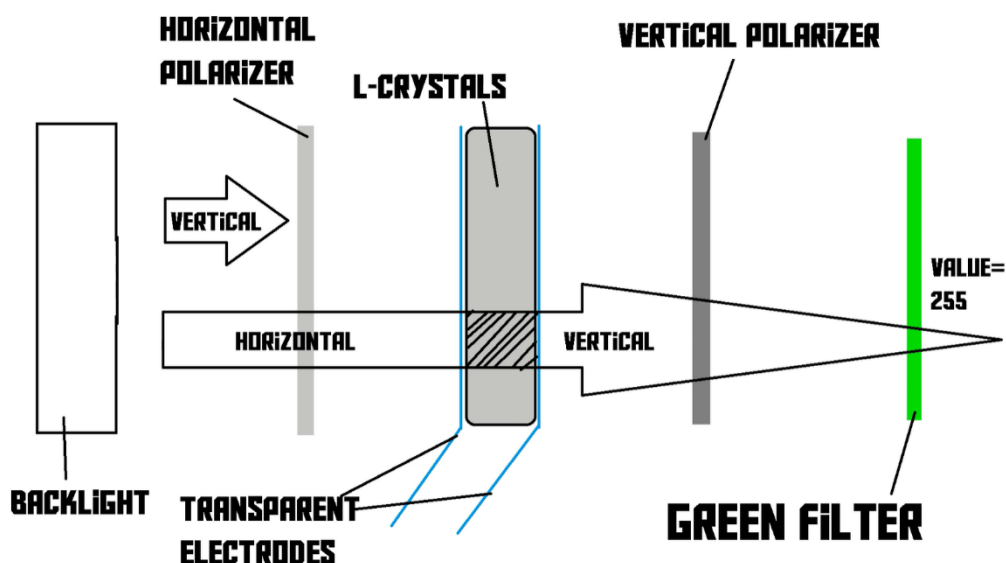
- krystaly jsou ve stejné rovině jako první polarizační filtr, displej nesvítí v klidovém stavu
- opačná logika oproti TN

Výhody

- kvalita obrazu, životnost, spotřeba energie, odrazivost a oslnivost, bez emisí

Nevýhody

- citlivost na teplotu, pevné rozlišení, vadné pixely, doba odezvy



OLED

(organic light-emitting diode/display)

- hlavním prvkem displeje je organická dioda emitující světlo
- v buňce je kovová katoda, vrstva pro přenos elektronů, organická vrstva emitující světlo, vrstva pro přenos děr a průhledná anoda
- po přivedení napětí na obě elektrody se začnou hromadit elektrony na straně anody (vrstva pro elektrony), díry (kladné částice) se nahromadí u katody (vrstva pro přenos děr)
- v organické vrstvě začne docházet ke srážkám mezi elektrony a dírami
- srážky vyvolají jejich vzájemnou eliminaci (rekombinace), přičemž dojde k vyzáření fotonu

PMOLED (Passive Matrix OLED)

- pasivní OLED, viz pasivní LCD

AMOLED (Active Matrix OLED)

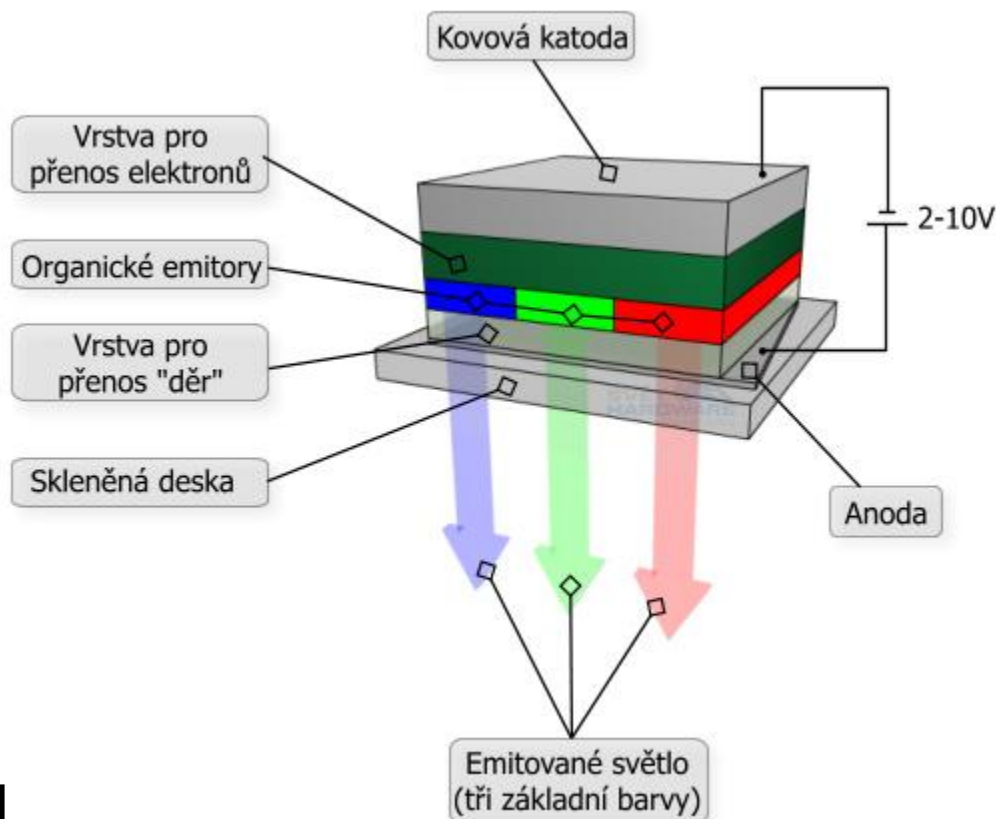
- aktivní OLED, viz aktivní LCD

Výhody

- nepotřebují podsvícení, vysoký kontrast, velmi tenké, nízká spotřeba, dobrý pozorovací úhel, minimální zpoždění, snadná výroba, možnost použití v pružných displejích

Nevýhody

- životnost, náchylnost na zničení vodou



E-ink / EPD

(elektronický inkoust, electrophoretic display)

- o Inkoust je tvořen mikrokapslemi o velikosti desítky až stovky μm
- o Kapsle obsahuje elektricky separovatelný roztok
- o V roztoku jsou částice v jednotkách mikrometrů
 - o černé částice – záporné +
 - o bílé částice – kladné -
- o Mezi kapslemi jsou elektrody
- o Po přivedení elektrického napětí na obě elektrody se začnou částice v kapslích přitahovat k elektrodě s opačnou polaritou
- o Jako roztok se používá hydrokarbonový olej (částice vydrží na místě i po odpojení napájení)
- o Černé částice jsou z uhlíku
- o Bílé jsou z oxidu titaničitého
- o K pohybu částic potřebujeme proud o velikost několika desítek nA při napětí 5 - 15 V
- o Pokud se display nepřekresluje nespotřebovává energii

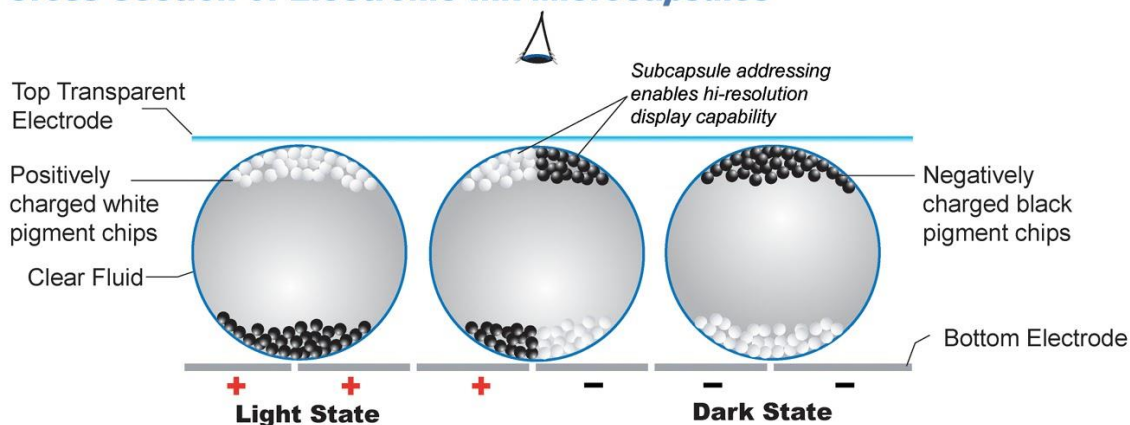
Výhody

- o Vysoké rozlišení
- o Dobrý kontrast a pozorovací úhly
- o Čitelnost na přímém slunci
- o Není nutné podsvícení
- o Možnost používat na pružném podkladu
- o Minimální spotřeba při překreslení, nulová spotřeba při statickém zobrazení

Nevýhody

- o Pouze 16 odstínů šedi (existují barevné verze po 4096 barev)
- o Špatné barevné rozlišení
- o Zpoždění při překreslování

Cross-Section of Electronic-Ink Microcapsules



NOTE: Copyright E Ink Corporation, 2002. Image not drawn to scale - for illustration purposes only.

Princip multiplexu na zobrazovacích zařízeních

- multiplexované displeje nejsou ovládány naráz, ale po částech (řádky, znaky, pixely)
- data nejsou zasílána naráz, ale za sebou (např. řádek, sloupec, znak)
- používá se časový multiplex, na displeji se zobrazuje naráz pouze část dat, ale vzhledem k rychlému blikání (alespoň 30 - 50 za sekundu) se to lidskému oku jeví jako souvislý obraz

Znakově orientované displeje

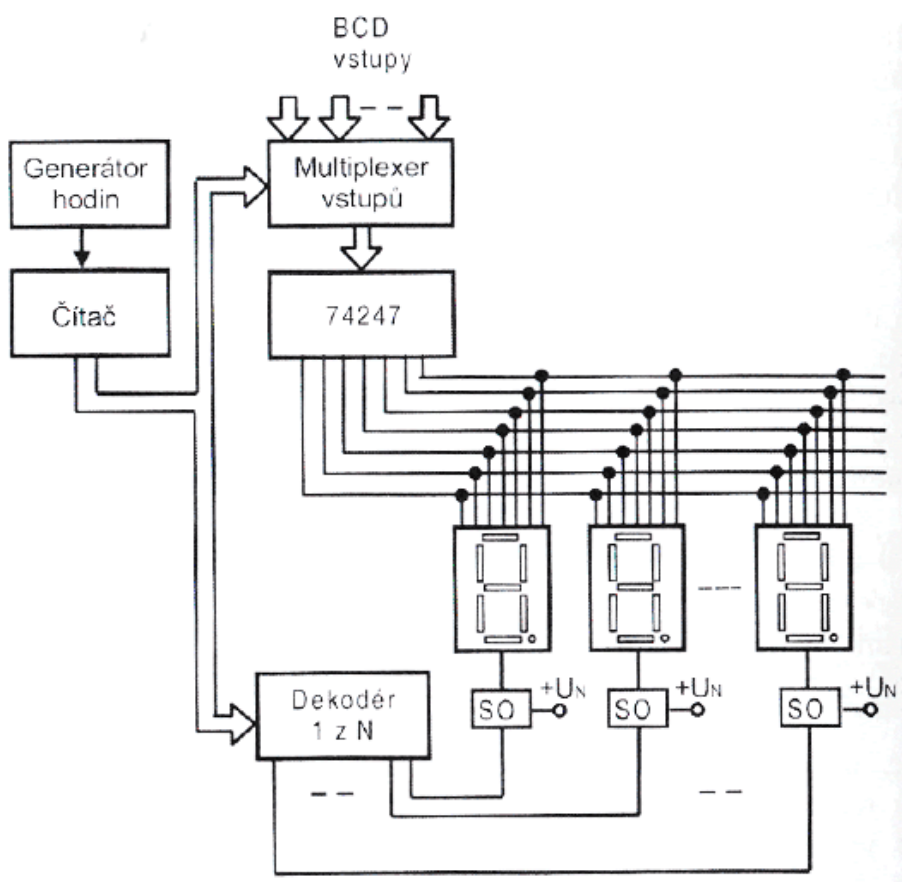
- např. sedmisegmentové displeje, vykreslují se po jednom znaku
- na displeji je několik pozic pro znaky, což tvoří matici
- při vykreslení musíme udát, na které pozici (řádek, sloupec) a jaký znak chceme vykreslit
- znaky jsou uloženy v paměti displeje a stačí tedy vybrat znak podle jeho adresy v paměti
- u 7 segmentového displeje s 4 znaky nám stačí 11 vodičů (7 pro segmenty a 4 pro zvolení, který znak se má změnit) - stačí $m + n$ místo $m * n$ vodičů

Pixelově orientované displeje

- displej je tvořen maticí pixelů, pixely můžeme ovládat buď jednotlivě anebo po řádcích/sloupcích
- LCD displeje s aktivní maticí si pamatují poslední stav na každém pixelu, takže ten zůstává svítit i když zrovna není aktivně ovládán

Výhody

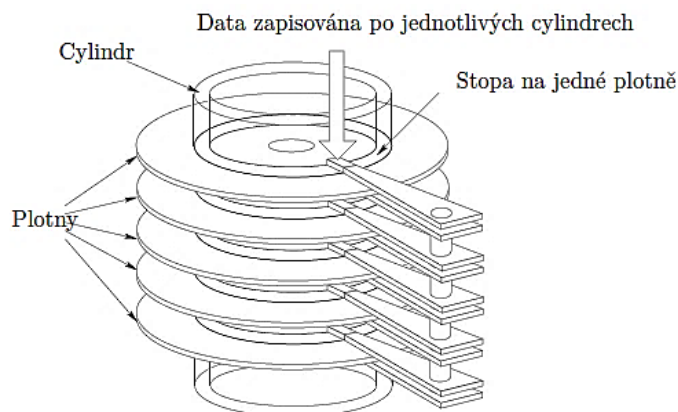
- méně vodičů, jednodušší elektronika, snížená spotřeba



HDD

(Hard Disk Drive)

- záznamové medium - disk, páska s magnetickou vrstvou
- zápis a čtení pomocí hlavičky, která je těsně nad médiem
- magnetický obvod je tvořen jádrem a cívkou navinutou na jádře
- pokud magnetickým obvodem prochází proud vzniká magnetický tok, který se díky štěrbině dostává do okolí a ovlivňuje magnetickou vrstvu média.
- pokud se mění směr proudu mění se také směr magnetického toku a tím orientace zmagetizování média
- nejprve byly záznamy naležato, nyní jsou svisle (princip gigantické magnetorezistivity)



Obrázek 2: Cylindry

Magnetické reverzace

- místo kde se mění orientace magnetizace média
- při čtení se hlavička pohybuje stále jedním směrem
- pokud hlavička narazí na místo magnetické reverzace dojde k vyvolání magnetického toku a vzniku napěťového impulsu
- data jsou kódovány (neodpovídají 1:1 reverzacím na disku, např. Run-length limited 2.7)

Části

- Plotny disku – média, oboustranné
- Hlavy pro čtení a zápis
- Pohon hlav
- Vzduchové filtry
- Pohon otáčení ploten disku
- Řídící deska



Podélný zápis

Jednotlivé bity, interpretované jako malá opačně orientovaná magnetická pole, se uchovávají jako vektory rovnoběžné s plotnou disku. Tímto způsobem lze však dosáhnout hustoty zápisu jen kolem 150 gigabitů na čtverečný palec. Při vyšších hustotách dochází vlivem paramagnetismu k samovolné ztrátě uložených dat. Superparamagnetismus je přirozený fyzikální jev, nejedná se tedy o konstrukční nedokonalost disku či technickou bariéru, nýbrž o fyzikální mez této metody.

Kolmý zápis

Vektory magnetické indukce jednotlivých bitů zde nejsou orientovány rovnoběžně s plotnou, nýbrž kolmo na ni. Tím je možné zvýšit kapacitu pevných disků až desetinásobně a přiblížit se hranici hustoty 1 terabit na čtverečný palec. S novou technologií však přichází i větší technologická náročnost řešení. Pro potřeby kolmého zápisu bylo nutné vyvinout novou diskovou hlavu pro zápis a přidat pod datovou vrstvu ještě vrstvu z magneticky měkkého materiálu.

Stopa

- rozdělení plotny na kružnice, u moderních disků statisíce

Cylindr

- stopy nad sebou tvoří pomyslný válec – cylindr

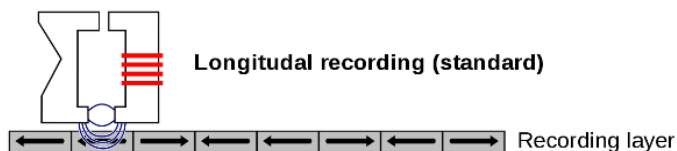
Sektor

- nejmenší adresovatelná jednotka - velikost je dána formátováním (kdysi 512 B, dnes 4096 B)
- číslování od 1, na začátku je hlavička s číslem sektoru, na konci je zakončení sektoru, obsahující kontrolní součet
- před a za sektorem jsou mezisektorové mezery, do kterých nelze uložit data

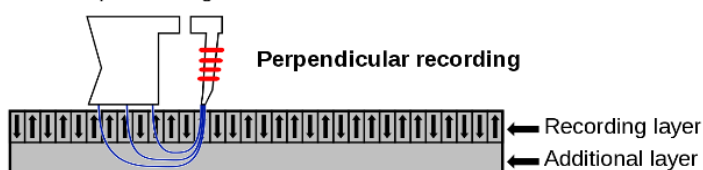
Čtení

- 1) Hlava se přesune nad požadovanou stopu (nejdelší)
- 2) Disk se natočí do požadované pozice

"Ring" writing element



"Monopole" writing element



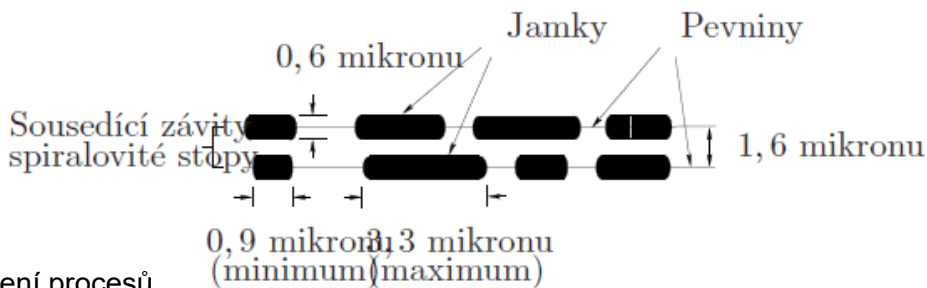
Optická média

CD-ROM (Compact Disc - Read Only Memory)

- kapacita **650 - 700 MiB**, šířka pitu – 0,6 um, hloubka – 0,125 um, červený laser
- tvořen polykarbonátem, do kterého jsou zapsána data lisováním matrice
- jedničky jsou zapsány pomocí přechodů mezi pití (prohlubně) a landy (ostrůvky), nuly se nezapisují (nulu označuje úsek bez přechodů mezi pitím a landem či naopak)
- při čtení prochází laserový paprsek tělem disku, pokud se v místě odrazu od horního povrchu nachází pit, dojde k rozptýlení paprsku, což vyvolá na čtecím senzoru jinou odezvu než když se paprsek odrazí od landu
- data jsou organizovány do stopy (spirály), která se vine směrem k okraji disku
- stopa se dělí na sektory (1x - 75 sektorů za vteřinu - 540 ot./m), kapacita sektoru je asi 3 KiB
- přehrávač musí měnit rychlost čtení v závislosti na místě čtení (CLV - constant linear velocity)

Části

- laserová (optická) hlava
- fotodetektor
- servomechanismus
- ovládací čip
- vyrovnávací paměť
- elektronika pro dekodování signálů a řízení procesů



Obrázek 6: Jamky a pevniny u CD médií

CD-R (CD - Recordable)

- paměť typu WORM (write once, read many times), uživatel může jednou vypálit data
- silným laserovým paprskem (výrazně vyšší výkon než při čtení) dojde k zahřátí a změně vlastností materiálu (vznik pitu)

CD-RW (CD - ReWritable)

- data lze zapisovat a opět vymazat pomocí změny krystalické struktury nebo přeměny amorfni látky na krystalickou

DVD (Digital Versatile Disc)

- vysokokapacitní CD
- větší kapacita je dána zmenšením rozměrů pitů, odstraněním zbytečného kódování a změnou vlnové délky paprsku (780 nm u CD, 650nm u DVD)

DVD-RAM (DVD - Random Access Memory)

- dá se s ním pracovat jako s pevným diskem, je uložen v pouzdře

DVD-RW, DVD+RW

- konkurenční formáty pro opakovatelně přepisovatelné DVD disky

DVD-ROM

- záznam může být oboustranný a ve dvou vrstvách (kapacita **4,7 - 17 GiB**)
- šířka pitu – 0,4 um
- hloubka – 0,105 um



Flynnova klasifikace, paralelní systémy

SIMD (single instruction, multiple data)

- MMX, SSE, vektorové instrukce a procesory

SISD (single instruction, single data)

- klasický přístup, dle von Neumanna

MISD (multiple instruction, single data)

- moc se nepoužívá, např. ovládání Space Shuttle, VLIW

MIMD (multiple instruction, multiple data)

- vícejádrové systémy, nejčastější (přešlo se tímto směrem), např. počítačové clustery

SIMT (single instruction, multiple thread)

- CUDA

MSIMD (multiple single instruction stream, multiple data stream)

- několik SIMD systémů pracuje nezávisle na sobě

SPMD (same program, multiple data stream)

- modifikace SIMD, všechny procesory provádějí stejný program nezávisle na sobě (bez synchronizace)

Amdahlův zákon

- pravidlo používané v informatice k vyjádření maximálního předpokládaného zlepšení systému poté, co je vylepšena pouze některá z jeho částí

- využívá se např. u víceprocesorových systémů k předpovězení teoretického maximálního zrychlení při přidávání dalších procesorů

skládá se z $f_s + f_p = 1$

f_s = sériový čas

f_p = paralelní čas

p = počet procesorů

$$Z_p = \frac{t * f_r}{p}$$

Z_p = zrychlení paralelní části

$$S_z = \frac{t}{t * f_s + Z_p} = \frac{1}{1 - f_p + \frac{f_p}{p}}$$

u S_z je důležité, abychom odbourali sériový čas \rightarrow musí platit $f_p > f_s$

Paměť

Sdílený prostor - procesy mají přístup k jedné paměti

Distribuovaná paměť - každý proces má vlastní adresní paměť

Paměť	Ne (sdílené)	Ano (sdílené)
Ne (distribuované)	klasické 1x CPU	UMA, SMP (desítky CPU)
Ano (distribuované)	Clustery, multipočítače	NUMA

UMA - uniform memory access

SMP - symetrický multiprocessing, všechny procesory jsou stejně výkonné

NUMA - non-uniform memory access

Systémy se sdílenou pamětí (SMP - shared memory processing)

- 1) Paměť a procesor jsou na sdílené sběrnici
- 2) Paměť je rozdělena na menší kousky, do kterých je přistupováno paralelně
 - a) Sběrnice je nahrazena přepínačem (drahý)
 - b) Systém lineárních přepínačů (výhybek) se 2 vstupy a 2 výstupy (levnější)

Systémy s distribuovanou pamětí (DMP - distributed memory processing)

- k přístupu do vzdálené paměti se posílá zasílání zpráv (message passing)
- nutné při větším počtu procesorů

- 1) Mřížka sběrnic - každý procesor je připojen na 2 sběrnice (vodorovnou a svislou), procesory si přeposílávají zprávy
- 2) Křížový přepínač - drahé, komunikaci lze provádět po ethernetu
- 3) Vícevrstvý přepínač (nepřímá síť) - není potřeba přeposílat zprávy, každý procesor má přístup všude
- 4) Přímá síť - obdoba mřížky sběrnic, ale procesory jsou propojeny přímo

Statické sítě

- přímé spojení (point-to-point) mezi jednotlivými uzly
- uzly jsou pevně propojeny, zprávy musí přejít přes několik uzlů
- propojení všech (obtěžně realizovatelné v praxi), hvězda, mřížka, kružnice, cesta, strom, krychle, hyperkrychle

Dynamické sítě

- spojení je realizováno přepínači a komunikačními linkami
- uzly jsou propojeny až za běhu aplikace, pokud je to potřeba (přímé propojení dvou uzlů)
- sběrnice, křížový přepínač, promíchání s výměnou (omega síť)

VLIW (very long instruction word)

- počítače s dlouhým instrukčním slovem, každá instrukce se používá k řízení většího počtu jednotek v jednom procesoru, instrukční úroveň paralelismu (např. Intel Itanium)

Zálohované systémy

- výpočet probíhá v několika procesorech zároveň, jedná se ale o stejná data i instrukce, takže jde o SISD (parallelismus se zde nepoužívá ke zlepšení výkonu, ale k zajištění bezpečnosti a spolehlivosti)
- Statická záloha - jsou zálohovány neustále
- Dynamická záloha - jsou zálohovány pouze v případě potřeby
- duplexní (2 procesory), TMR (třímodulově redundantní, 3 procesory), biduplexní (4 procesory)

Systémy řízené tokem událostí

- k určité operaci dochází v okamžiku, kdy to okolnosti dovolí, operace neprobíhají na základě instrukcí (řízeno tokem dat nebo požadavků - redukční počítače)

Systémy bez centrálního řízení

- neexistuje v nich program ani řadič

Systolické systémy

- obvykle jednoúčelové, provedená operace je dána vnitřní strukturou systému

Model neuronové sítě

- modelování umělé inteligence, vzdalují se Von Neumannově koncepci (přestávají se používat věci jako řadič, procesor, paměť)

Úrovně granularity

- určuje, jak velké úseky zpracování informace probíhají současně
- nejjemnější je granulován proces na úrovni 1, nejhruběji proces na úrovni 5
- vývoj rostl od nižších po vyšší úrovně

5 - nezávislé úlohy a programy

4 - části úloh a programů

3 - podprogramy

2 - cykly, iterace

1 - příkazy, instrukce

Stupně paralelismu

1) Pracovní stupeň paralelismu

- poskytování činností jako souboru nezávislých úloh (např. během čekání na I/O data je proces pozastaven a přednost dostane jiný proces)

2) Programový stupeň paralelismu

- řeší se v kódu programu, např. rozdělení matice na 4 submatice, každou vypočte jeden procesor (4x zrychlení) nebo třídění částí seznamu více procesory

3) Instrukční stupeň paralelismu

- stará se o něj procesor nebo překladač, pro programátora je plně transparentní

4) Aritmetický a bitový stupeň paralelismu

- paralelismus na velmi nízké úrovni, doména návrhářů ALU a FPU jednotek

HPC

(high performance computing)

- superpočítače k vědeckotechnickým výpočtům, obvykle za použití výpočetních clusterů
- clustery se nejčastěji skládají z multiprocessorových SIMD sestav, používají se heterogenní součásti, které ale jsou na úrovni běžných uživatelských strojů, jen je jich hodně
- clustery jsou propojeny vysokorychlostní sítí (Ethernet, Infiniband), sdílejí úlohy pomocí technik SMP (symmetric multiprocessing) a NUMA (non-uniform memory access)

Migrace procesů

- přesunutí procesu z jednoho procesoru na jiný v rámci clusteru
- Preemptivní - proces lze přesunout za jeho běhu, složitější, ale často se používá
- Nepreemptivní - proces lze přesunout pouze než začne, jednodušší na implementaci
- zvyšuje výpočetní výkon tím, že přesouvá procesy na nevyužité procesory (load balancing)
 - může přesunout proces blíže ke zdroji dat
 - lze přesunout dlouhodobě trvající proces ze selhávajícího uzlu na jiný uzel
 - lze restartovat uzly (jeho procesy se přesunou jinam)
 - implementace může být na úrovni uživatele (API) nebo kernelu (signály, systémová volání)

Checkpoint-restart

- systém pro pozastavení procesu a jeho následné obnovení

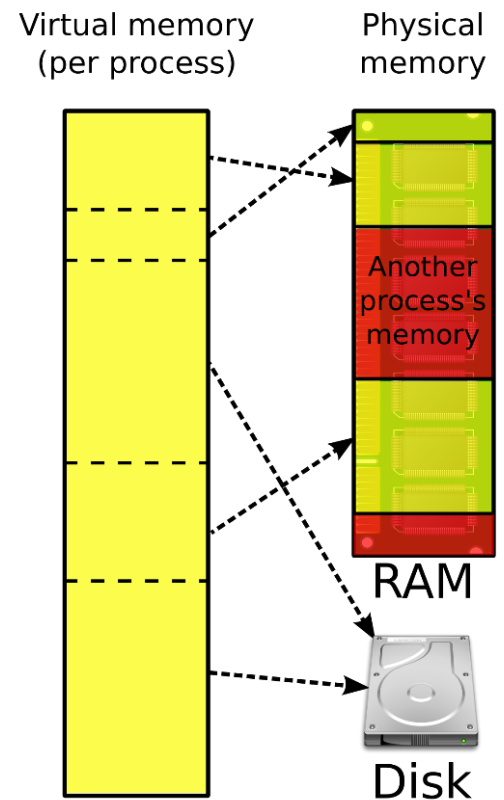
Virtuální paměť

Virtuální paměť (též virtualizace paměti) je v informatice způsob správy operační paměti počítače, který umožňuje předložit běžícímu procesu adresní prostor paměti, který je uspořádán jinak nebo je dokonce větší, než je fyzicky připojená operační paměť RAM. Z tohoto důvodu procesor rozlišuje mezi virtuálními adresami (pracují s nimi strojové instrukce, resp. běžící proces) a fyzickými adresami paměti (odkazují na konkrétní adresové buňky paměti RAM). Převod mezi virtuální a fyzickou adresou je zajišťován samotným procesorem (je nutná hardwarová podpora) nebo samostatným obvodem.

V současných běžných operačních systémech je virtuální paměť implementována pomocí stránkování paměti spolu se stránkováním na disk, které rozšiřuje operační paměť o prostor na pevném disku (stránkování na disk je nesprávně označováno jako swapování).

- je řešena jednotkou nazývanou MMU (memory management unit)

Všechny adresy, které proces používá, jsou spravovány pouze jako virtuální – transformaci na fyzické adresy provádí správa virtuální paměti.



Existují dvě základní metody implementace virtuální paměti – stránkování a segmentace paměti.

Při **stránkování** je paměť rozdělena na větší úseky stejné velikosti, které se nazývají stránky. Správa virtuální paměti rozhoduje samostatně o tom, která paměťová stránka bude zavedena do vnitřní paměti a která bude odložena do odkládacího prostoru (swapu).

Při **segmentaci** je paměť rozdělena na úseky různé velikosti nazývané segmenty.

Stránkování

- paměť je rozdělena na stránky, udržované v tabulce stránek (mají příznak, jestli jsou v operační paměti nebo ne)
- při pokusu o načtení stránky, která není v paměti, se vyvolá page fault (transparentní pro programátora, řeší OS) a daná stránka se načte z pevného disku do paměti
- velikost stránky je 4 KiB (nebo 16, 64), musí být dostatečně velká, aby se kompenzovaly dlouhé přístupové doby k disku
- pokud je operační paměť plná, používají se algoritmy pro zjištění ideální stránky na odstranění

Virtuální adresa

- je rozdělena na 10, 10, 12 bitů (řádek v adresáři stránek, řádek v stránkovací tabulce, posuv v rámci - fyzické stránce)
- u 64 bitové architektury je to 9, 9, 9, 9, 12 bitů

Segmentace

- paměť je rozdělena na úseky různé délky (narozdíl od stránek se stejnou délkou)
- každý segment má svůj začátek, limit, směr, práva a atributy
- adresa se skládá ze segmentu a posunu (výsledkem je jejich součet - lineární adresa)
- probíhá kontrola oprávnění (no execute - nelze vykonávat, execute only - nelze číst)



- bazová adresa segmentu se nastavuje predem (nastavuje ho OS, môže ho premiestit), všetky adresy jsou vyjádřeny relativně k němu (offset - posuv)

- procesy mají k dispozici typicky několik segmentů (kód, knihovny, data, zásobník, halda

DMA

(direct memory access)

-Princíp spočíva v prostom presune informácii z periférneho zariadenia do hl. pamäte a naopak

-Princíp je realizovaný formou priameho prístupu do pamäte DMA, na kt. nie je prítomné CPU, jediná podmienka je uvolnenie sbornice, kde CPU prepne všetky zbornice do 3, vysokoimpedačného, ~~meeđu~~ módu. Počas doby presunu nemôže sbornica ostať bez riadenia preto tu máme **block DMA**

BLOCK DMA

Skladá sa z:

Register dát - obsahuje slovo, ktoré má byť presunuté

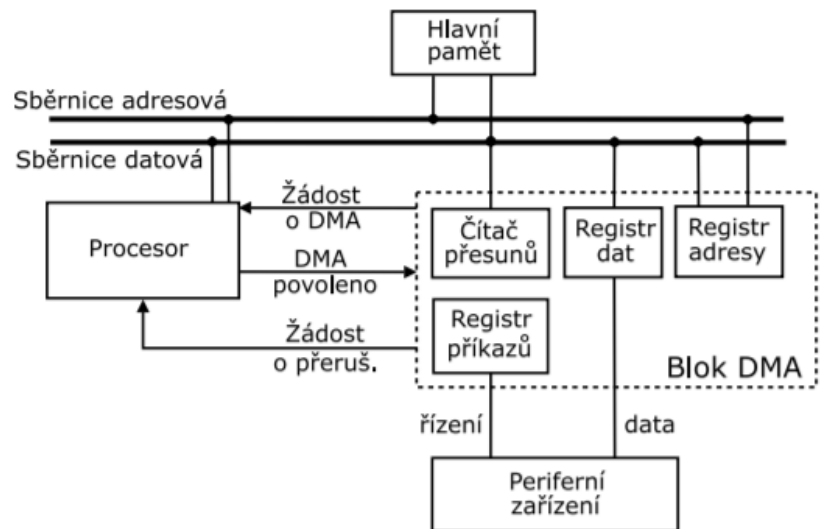
Register adresy - obsahuje adresu hl.

Pamäte, na ktorú bude dané slovo zapísane, alebo z ktorej bude prečítané (to slovo ofc)

Čítač presunov - obsahuje počet slov, ktoré ešte budú presunuté v rámci spojenia

Block DMA pracuje v 2 ~~meeđu~~ módoch:

1) presúvanie dát jednotlivo 2) presúvanie v blokoch



Priebeh operácie v krokoch:

1) naprogramovania **bloku DMA** procesorom

2) Blok spustí periferne zariadenie a čaká kým je ready, keď periferne zariadenie povie bloku že je ready, blok pošle žiadosť o prístupu do pamäte na CPU

3) procesor dokončí cyklus a potom reaguje na žiadosť, prístup do pamäte funguje tak, že blok DMA vysiela synchronne data s fázou f1 a procesor používa pamäť s fázou f2 (moja špekulácia: striedajú sa jeden za druhou xd)

4) prístup do pamäte môže prebehnúť až vtedy keď procesor vyšle signál ACK a uvoľní sbornicu, blok potom pošle na adresovú sbornicu obsah svojho **registru adresy** a na dátovú sbornicu obsah **registru dát**, ďalej zvýši obsah **registru adresy +1** a zníži obsah **čítača presunu -1**

Ak obsah **čítača presunu nie je nulový** checkuje či periferne zariadenie nepresunulo nové dáta na **register dát**, ak nebolo, dočasne sa ukončí a prenechá sbornicu procesoru

5) procesor pokračuje vo svojej činnosti, dokiaľ znova z bloku DMA nepríde žiadosť o prístupu do pamäte, teda periferne zariadenie má pripravené nové dáta

6) ak je obsah **čítača presunov 0**, blok ukončí presun a uvoľní zbornicu, požiada o nové naprogramovanie **bloku DMA**

VOLTA



Figure 5. Volta GV100 Streaming Multiprocessor

Table 2. Compute Capabilities: GK180 vs GM200 vs GP100 vs GV100

GPU	Kepler GK180	Maxwell GM200	Pascal GP100
Compute Capability	3.5	5.2	6.0
Threads / Warp	32	32	32
Max Warps / SM	64	64	64
Max Threads / SM	2048	2048	2048
Max Thread Blocks / SM	16	32	32
Max 32-bit Registers / SM	65536	65536	65536
Max Registers / Block	65536	32768	65536
Max Registers / Thread	255	255	255
Max Thread Block Size	1024	1024	1024
FP32 Cores / SM	192	128	64
Ratio of SM Registers to FP32 Cores	341	512	1024
Shared Memory Size / SM	16 KB/32 KB/48 KB	96 KB	64 KB

¹ The per-thread program counter (PC) that forms part of the improved SIMT model typically requires 16 register slots per thread.

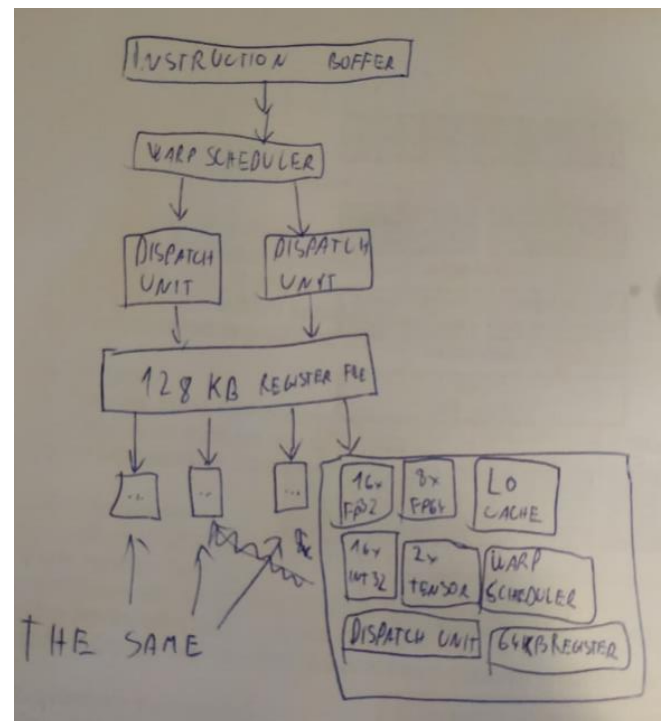
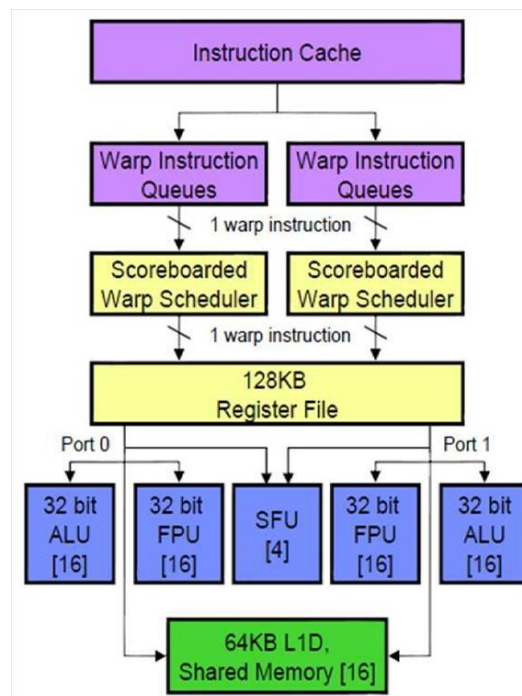
7.8 TFLOPS na 64 double precision-floating point

15.7 TFLOPS na 32 floating point

125 Tensor TFLOP

Legenda: FP(floating point), INT (Integer), SM(streaming multiprocessor)

-Streaming processor: 32xFP32bit Cores, 16 x FP64 Cores, an instruction buffer, one warp scheduler, two dispatch units, and a 128 KB Register File. The GV100 SM is partitioned into four processing blocks, each with 16 x FP32 Cores, 8x FP64 Cores, 16 x INT32 Cores, two of the new mixed-precision Tensor Cores for deep learning matrix arithmetic, a new L0 instruction cache, one warp scheduler, one dispatch unit, and a 64 KB Register File



- **L0 instruction cache** is now used in each partition to provide higher efficiency than the instruction buffers used in prior NVIDIA GPUs

-**Tensor Core**: špeciálne cores pre násobenie matíc, podpora pre machine learning, 12x TFLOPS oproti Pascal GP100 na pri rovnakej spotrebe

-50% energy efficiency na general compute workloads

-Data cache a shared memory implementuje do jedného bloku (30% nárast výkonnosti oproti Pascalu)

Intel Core i series

- nástupce řady Intel Core, zaměřený na modularitu a škálovatelnost
- Nehalem, Sandy Bridge, Ivy Bridge, Haswell, Broadwell, Skylake, Kaby Lake, Coffee Lake
- i3, i5, i7, i9
- procesory mají integrovaný paměťový řadič - každý procesor může mít přiřazen vlastní část paměti, přístup k částem paměti od jiného procesoru je pomalejší (NUMA - non-uniform memory access)

QPI (Quick Path Interface)

- náhrada za zastaralou systémovou sběrnici FSB (Front Side Bus)
- sériová linka, dva 20 bitové spoje (16 pro data, 4 pro opravu chyb a řízení)
- propustnost 12.8 GB/s v každém směru
- lze je kombinovat a použít až čtyři pro komunikaci s periferiemi

Turbo Boost

- technologie, která umí dočasně přetaktovat jádra procesoru (každé zvlášť)
- monitorováním vytížení lze např. převést výpočet pouze na dvě jádra ze čtyř a obě přetaktovat
- teplota procesorů je monitorována, pokud přesáhne bezpečnou mez, je jejich takt opět snížen

Hyper-Threading

- technologie umožňující lepší využití všech prostředků procesoru (aby např. FPU neleželo, když zrovna probíhá výpočet na ALU)
- je zdvojeno vše, co udržuje stav procesoru (registry, čítač instrukcí), lze mezi těmito dvěma sadami přepnout v jednom hodinovém taktu - nárůst výkonu o cca. 30 %
- každé jádro tak tvoří dva logické procesory (jádra)

Virtualization Technology

- režim VMX (Virtual Machine eXtension) operation – sandbox instrukcí, mají jen svou paměť
- v režimu root-operation běží VMM (Virtual Machine Monitor), v non-root běží VM systém
- VM OS může běžet pouze v protected režimu, reálný mód je emulován pomocí VMM
- VM OS není emulovaný, ale běží přímo na hardwaru, takže přímo využívá ovladače zařízení

Digital Media Boost - lepší výkon s plovoucí desetinnou čárkou a v multimediálních výpočtech

Trusted Execution Technology - hardwarová výjimka poskytující větší bezpečnost programů

AES New Instructions

- hardwarová podpora (6 instrukcí) pro šifrování AES (Advanced Encryption Standard)
- podporuje klíče velikosti 128/192/256 bitů

SpeedStep Technology

- opak Turbo Boostu, snižuje takt procesorů, pokud nejsou využívány
- snižuje hlučnost větráků a šetří výdrž baterie u notebooků

Execute Disable Bit

- ochrana před útoky typu buffer overflow a škodlivým softwarem
- lze označit paměťové oblasti, ve kterých smí nebo nesmí být spouštěn kód

SSE 4.2

- 7 nových instrukcí
- spočítání nenulových bitů v registru (rozpoznávání hlasu, sekvencování DNA)
- instrukce pro výpočet CRC32 (Cyclic Redundancy Check - kontrola integrity dat)

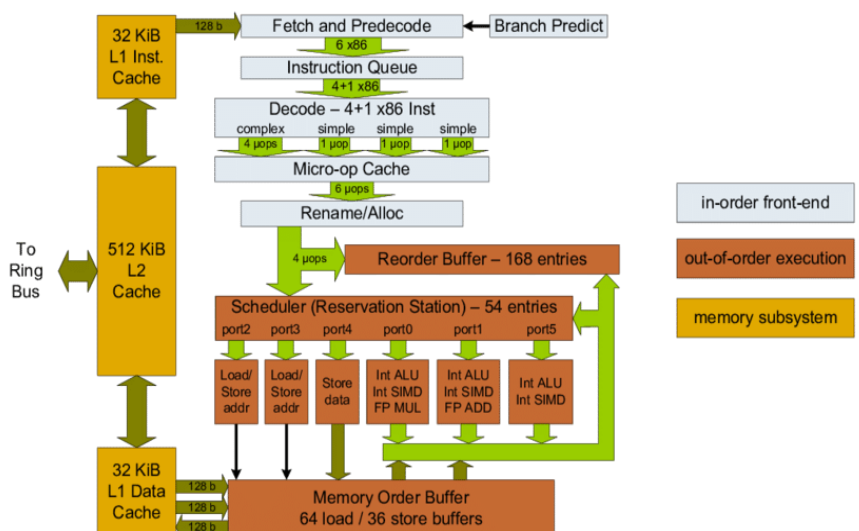


Figure 1 Sandy Bridge core microarchitecture

- řetězcové operace

Threads

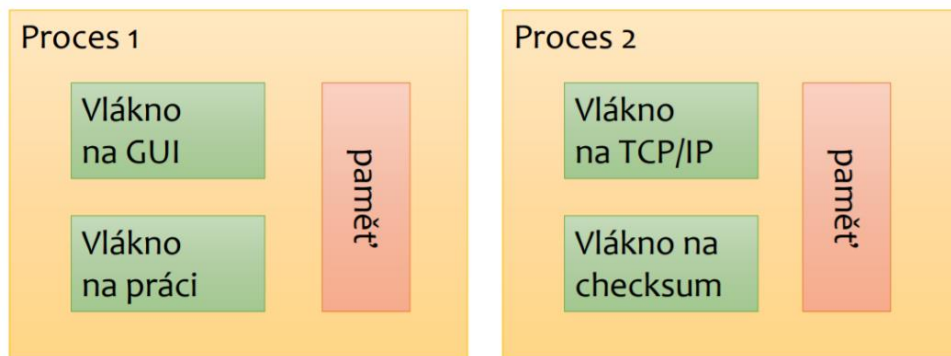
Běh programu se nazývá proces. Proces obsahuje jedno nebo více vláken. Vlákno je nejhrubší jednotka paralelismu. Procesy žijí v navzájem oddělených adresních prostorech. Vlákna jednoho procesu naopak paměť sdílejí. Vzhledem k tomu, že vlákna sdílejí paměť, je potřeba je synchronizovat (mutexy, zamky, atomické proměnné...).

Modely paralelismu

Boss-Worker - Hlavní vlákno rozděljuje úkoly ostatním vláknům. Vlákna sdílejí kód, ale nesdílejí data.

Pipeline - Každé vlákno provede nějakou práci nad daty a pošle výsledek dalšímu.

Work crew - Jedno vlákno je privilegované a rozdává práci. Ostatní vlákna zpracovávají stejnou práci nad různými daty a vrací výsledky.



Použití

Obsluha periférií

- Periodické pollovanie hardware.
- Jedno vlákno pro komunikaci s uživatelem a druhé obsluhuje hardware

Síťová komunikace

- Jedno vlákno akceptuje příchozí komunikace.
- Jedno vlákno odesílá data.
- Jedno vlákno zpracovává data.

Vyvolání dojmu rychlé odezvy programu

- Práce s velkým objemem data uložených v databázi.
- Hlavní vlákno pouze obsluhuje uživatelské rozhraní, další pracuje s databází

Urychlení výpočtu

- Lze-li spustit na víceprocesorovém stroji kooperující vlákna na několika procesorech.

Efektivita

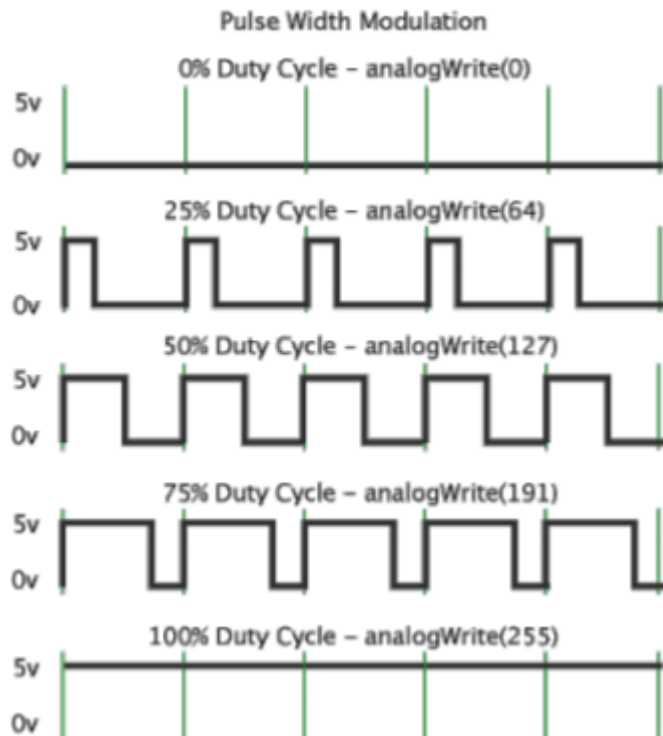
- Některé aplikace jsou ze své podstaty nevhodná pro jednovláknovou architekturu.
- Použití vláken může vést k výraznému zpřehlednění programového kódu

PWM

(Pulse-width Modulation)

Pulzně šířková modulácia je modulácia periodického signálu zmenou šírky impulzu v závislosti od vstupnej veličiny za účelom prenosu informácie. PWM má pevnou frekvenciu pulzu s ruznou šírkou pulzu a mezery; perioda a šírka pulzu se zadáva v mikrosekundách nebo v milisekundách. Když je šírka pulzu rovná době cyklu, strída (duty cycle) je 100 % a výstup je trvale zapnutý. Jestliže je šírka pulzu nulová, tvorí pulz 0 % periody a výstup je stále vypnutý.

Vysoká účinnosť pri regulácii výkonu je daná tým, že regulátor je (v ideálnom prípade) vždy buď úplne uzavretý, alebo úplne otvorený. Nevznikajú v ňom preto tepelné straty v dôsledku úbytku napätia na regulačnom prvku s odporovým charakterom (rezistor, polovodičový priechod), ako je tomu pri spojitých regulátoroch. Je to však vykúpené zložitejším zapojením nespojitých regulátorov, vysokými nárokmi na použité spínacie súčiastky a vysokofrekvenčným rušením, vznikajúcim rýchlym prerušovaním výkonového obvodu, ktoré je potrebné odstraňovať filtermi a elektromagnetickým tienením nespojitého regulátora.



Riadenie jasu LED pomocou PWM:

Logická 0 značí zhasnutie LED diody, logická 1 zapnutú. Zmenou striedy sa mení hodnota jasu diódy, ktorá teda v podstate bliká z rozsvietenia do zhasnutia. Pokiaľ teda LED dióda svieti polovicu času, je jeho strieda a teda aj intenzita svietenia 50%.

Rozdelenie paralelných systémov podľa Flynna:

Flynn rozdeľuje systémy podľa dvoch kritérií a to:

1. Podľa počtu programov riešených paralelne:-

SI (single instruction stream)- Jeden program

MI (Multiple Instruction Stream) - Niekoľko programov

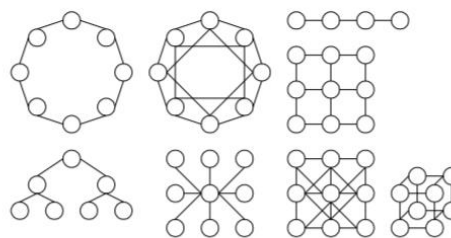
2. Podľa počtu údajových súborov spravovaných paralelne:

SD(Single Data Stream) - jeden tok údajov

MD(Multiple Data Stream) - Niekoľko tokov údajov

Statické – v nichž spojovací cesty zůstávají neměnné.

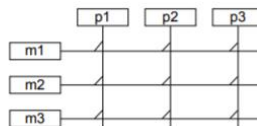
Typ	průměr	stupeň
Cesta	$N-1$	2
Kružnice	$N/2$	2
2xkruž	$N/4$	4
bin.strom	$2\lceil \log(N+1) \rceil$	3
Krychle	$\log N$	$\log N$



Dynamické - spoje volně vznikají a zanikají (prvky které umožňují spojovat).

• *křížový přepínač nebo jen trojúhelníkový*

- hlavní výhodou je vzájemná nezávislost propojovacích cest, která umožňuje současně realizovat tolik spojení, kolik dvojic In/Out lze vytvořit.
- vysoká cena.



Propojovací sítě, statické / dynamické

Problém komunikace mezi jednotkami. jednostranná složitější, každý s každým. U dvojstranné vyžadujeme pouze spoje p do m. neumožňuje spoje mezi. p x p. je to v podstatě přepínač. Důležitá i strategie řízení.

Paralelní architektura II.

- **Globální paměť** - zprostředkovává komunikaci mezi výpočetními jednotkami

Virtuální paměť

Virtuální paměť

- způsob přidělování paměti procesům tak, aby byla lineární a zdánlivě využívala celý adresní prostor, tímto způsobem lze efektivně sdílet paměť mnoha programy
- je řešena jednotkou nazývanou MMU (memory management unit)

Stránkování

- paměť je rozdělena na stránky, udržované v tabulce stránek (mají příznak, jestli jsou v operační paměti nebo ne)
- při pokusu o načtení stránky, která není v paměti, se vyvolá page fault (transparentní pro programátora, řeší OS) a daná stránka se načte z pevného disku do paměti
- velikost stránky je 4 KiB (nebo 16, 64), musí být dostatečně velká, aby se kompenzovaly dlouhé přístupové doby k disku
- pokud je operační paměť plná, používají se algoritmy pro zjištění ideální stránky na odstranění

LRU (least recently used) - odstraňuje z paměti stránky, které byly v poslední době nejméně používány, teoreticky poskytuje téměř optimální výkon, ale v praxi se složitě implementuje

Virtuální adresa

- je rozdělena na 10, 10, 12 bitů (řádek v adresáři stránek, řádek v stránkovací tabulce, posuv v rámci - fyzické stránce)
- u 64 bitové architektury je to 9, 9, 9, 9, 12 bitů

TLB (translation lookaside buffer)

- používá se pro urychlení výpočtu reálných adres, pamatuje si naposledy vypočtené reálné adresy a při požadavku na překlad virtuální adresy je rovnou vrátí, bez nutnosti výpočtu

Segmentace

- paměť je rozdělena na úseky různé délky (narozdíl od stránek se stejnou délkou)
- každý segment má svůj začátek, limit, směr, práva a atributy
- adresa se skládá ze segmentu a posunu (výsledkem je jejich součet - lineární adresa)
- probíhá kontrola oprávnění (no execute - nelze vykonávat, execute only - nelze číst)
- báze adresa segmentu se nastavuje předem (nastavuje ho OS, může ho přemístit), všechny adresy jsou vyjádřeny relativně k němu (offset - posuv)
- procesy mají k dispozici typicky několik segmentů (kód, knihovny, data, zásobník, halda)

Selektor

- horních 13 bitů je index do tabulky deskriptorů
- další bit určuje, zda-li jde o GDT (global descriptor table) nebo LDT (local descriptor table)
- dva nejnižší bity určují oprávnění

Deskriptor

- 32-bitová báze (počáteční adresa)
- 20-bitový limit segmentu (nelze použít vyšší offset, ochrana)
- bit granularity (0 - limit se počítá po bytech, 1 - limit se počítá po stránkách - 4 KiB)
- typ segmentu (kódový, datový, systémový)
- přístupová práva (READ/WRITE/EXECUTE)

- v segmentovém registru je uložena báze (začátek segmentu), k tomu se připočte offset (posuv), což vytvoří lineární (virtuální adresu), která je dále přeložena jednotkou MMU na fyzickou adresu s použitím TLB

Monolitické PC

- obsahují procesor, paměť a vstupní/výstupní periferie integrované v jednom pouzdře
- obvykle nesou znaky RISC, ale ve zjednodušené podobě, často se pro ně používá Harvardská architektura (lze použít různé paměti pro data a program - ten je často ROM - E(E)PROM nebo Flash)
- jsou zalité v jednom pouzdře - monolit.
- převážně se používá Harvardská architektura
- nejsou zobecnitelné: jiné využití, jiný návrh, jiná velikost slova, jiné instrukce.

Paměť

Můžou mít vnitřní paměť pro program

- přepisovatelná - EPROM, EEPROM
- jednou programovatelné (OTP)
- bez vnitřní programové paměti (ROM-Less)

Organizace Paměťě

- **Střadačové (pracovní) registry** - jeden či dva, obsahují právě zpracovávaná data a výsledky
- **Univerzální zápisníkové registry** - nejčastěji používaná data, instrukce lze provádět přímo nad těmito registry, lze mít několik přepínatelných skupin registrů - registrové banky
- **Data** - RWM (RAM), rozsáhlejší, méně používanější data
- **Čítač instrukcí** (program counter, instruction pointer) - ukazuje na právě vykonávanou instrukci
- **Zásobník** - slouží pro ukládání návratových adres, potřebuje další registr (ukazatel vrcholu zásobníku - stack pointer)

RESET - výchozí stav procesoru, je detailně popsán v dokumentaci, bývá realizován tlačítkem s RC obvodem

Ochrana proti rušení - mechanická, elektromagnetická, watchdog (hlídá zacyklení programu, postupně se dekrementuje/inkrementuje, při podtečení/přetečení provede RESET procesoru - procesor ho musí v určitých intervalech nulovat, pokud to neudělá, tak je nejspíše zacyklen a watchdog ho resetuje), obvod hlídající pokles napětí - brownout (při poklesu pod dovolenou úroveň vyvolá RESET)

Přerušovací podsystém - - Dost často bývá mikropočítač propojen s periferiemi, ty potřebují být obslouženy procesorem. Programátor musí povolit/zakázat přerušení, detekovat ho, určit jeho zdroj a co musí při obsluze provést.

Zdroj synchronizace - vytváří časový základ pro synchronní provádění operací

Krystal (křemen), keramický rezonátor - nejpřesnější, nejdražší

RC obvod, LC obvod - levné, oscilace se mění se změnou teploty a napětí

Periferie - obvody zajišťující komunikaci procesoru s okolím

V/V brány - nejjednodušší způsob komunikace - paralelní brána (port, realizován registrem - 4 nebo 8

jednobitových vývodů, u kterých lze nastavit, jestli slouží k zápisu nebo čtení), vstupy mohou používat pull-down nebo pull-up rezistory pro udržení logické 0 nebo 1 při nezapojeném vstupu

Čítač/časovač - registry, které jsou inkrementovány (čítač vnějšími událostmi, časovač vnitřním signálem procesoru), při přetečení vyvolají přerušení

A/D převodníky

Komparační - porovnává napětí s referenční hodnotou, rychlé, ale jejich potřebný počet narůstá velmi rychle, pokud chceme vyšší rozlišení

D/A převod - obsahuje komparátor

Sledovací - který zkouší porovnat napětí se vstupem, pokud neodpovídá, tak jej inkrementuje nebo dekrementuje (nehodí se pro skokově se měnící vstupy)

Aproximační - používá metodu půlení intervalu (maximálně n kroků, n je počet bitů rozlišení převodníku)

Integrační - používá metodu dvojité integrace (potřebuje externí kondenzátor)

S RC článkem - doba nabití/vybití kondenzátoru závisí na vstupním odporu

D/A převodníky

PWM (vid'. hore)

- výstupní napětí se mění pomocí měnění poměru doby logické 1 a 0 na výstupu

- převod šířkově modulovaných pulzů na analogovou veličinu zajišťuje RC článek (musí platit $R \cdot C \gg T$)

Paralelní převodníky

- přímý převod číselné hodnoty na proud (používá se odporová síť $R-2R$ - pouze odpory hodnoty R a dvojnásobku R)

Obvody RTC

- potřebují záložní zdroj (baterii), aby mohly "počítání" času provádět i při odpojení napájení

- je nutno zajistit atomicitu čtení času (hodnoty se na pozadí mění v čase), řeší se vhodným programovým řízením anebo pomocným registrem (do toho se uloží čas před čtením a během čtení se nemůže změnit)

POWER PC, IU, FPU, BPU, ZŘETĚZENÍ INSTRUKCÍ //???

- vnější adresová sběrnice šířky 32 b
- vnější datová sběrnice šířky 64 b

Struktura RISC procesoru PowerPC601 má tři výkonové jednotky, které jsou schopny pracovat současně.

- jednotka pro řízení skoků **BPU**
- **celočíslná jednotka IU**
- jednotka pro operace s pohyblivou čárkou **FPU**

Kromě toho jsou na čipu umístěny též:

- instrukční jednotka obsahující frontu instrukcí a BPU
- paměťová jednotka
- vyrovnávací paměť
- jednotka řízení paměti MMU
- systémové rozhraní.

Instrukční jednotka má za úkol zajistit, aby všechny tři výkonné jednotky (BPU, IU, FPU)

byly stále zaměstnány, protože procesor je schopen **začít zpracovávat až tři instrukce v jednom**

taktu. Není-li požadovaná instrukce přítomna ve vyrovnávací paměti, čeká instrukční jednotka s vysláním požadavku na čtení z hlavní paměti až do chvíle, kdy se fronta instrukcí vyprázdní nejméně do poloviny

Celočíslná aritmetika

provádí všechny celočíselné operace, operace s bitovými poli a zápisy do paměti a čtení z ní. Moduly zřetězeného zpracování instrukce v celočíselné jednotce jsou: dekódování, provedení a zápis výsledku.

Jednotka FPU

obsahuje sčítačku s dvounásobnou přesností, násobičku s jednoduchou přesností,

děličku a pole registrů. Toto pole registrů je dvoubránové, avšak není vybaveno schopností přímého předávání operandu. Jednotka FPU je tvořena **pěti moduly** zřetězeného zpracování. **První** je čekací registr, pak následuje **dekódování, dva prováděcí moduly a zápis výsledku**.

Jednotka BPU

je vybavena vlastní sčítačkou, používanou při výpočtu adres skoku.

História procesorov Intel

Intel 8080

- 8 bitový počítač
- základ pre prvé počítače intelu
- na úrovni assembleru kompatibilný s intelom 8086

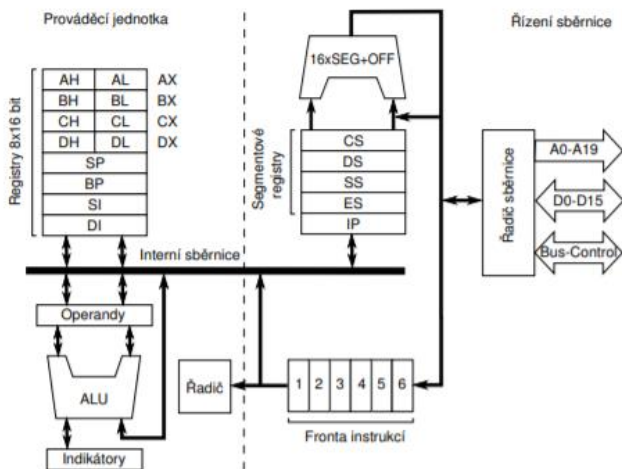
Prvá generácia:

- prvý 16b počítač
- vďaka segmentácii na 64kB bloky dokázal adresovať až 1MB (wau)

2.2 Intel 8086

Výroba	1976 ÷ 1990
Technológia	HMOS, 3,2 μ m
Tranzistory	29000
Frekvencie	4.77 MHz ÷ 10 MHz
Datová sb.	16 bit
Adresní sb.	20 bit

Tento procesor je prvním 16 bitovým procesorem. Dovoľoval adresovať až 1MB pamäti a to díky segmentaci paměti na 64kB bloky.



Obrázek 1: Architektura procesoru i8086

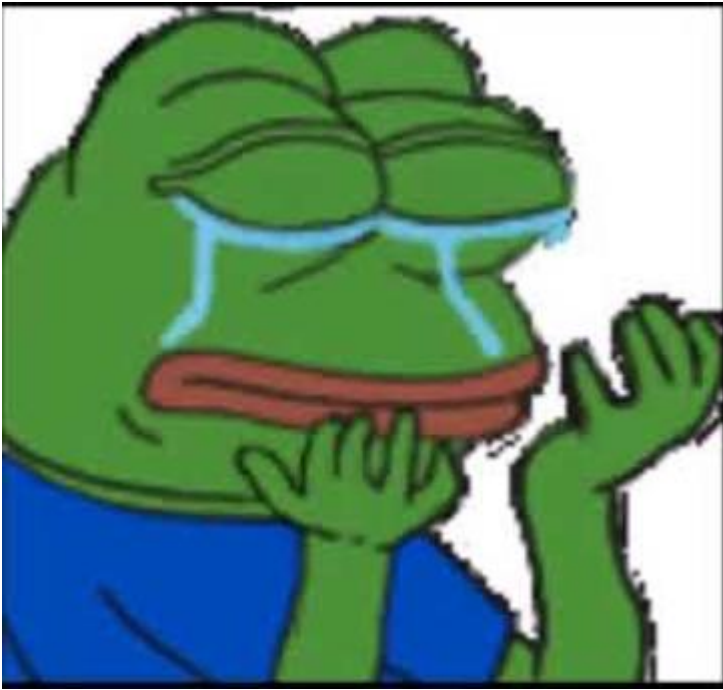
Vnitřní schéma je na obrázku 1. Z obrázku je patrná jednoduchá vnitřní architektura s oddělením výkonné a adresovací části procesoru. Osm 16 bitových registrů tvoří základ, který je implementován ve všech následujících generacích x86.

Na obrázku:

- rozdělená výkonná a adresovacia časť procesor
- základ tvorí 8x16b registrov FeelsOldMan

Druhá generácia:

- označenie 80286
- zvýšenie výkonu: možnosť taktovať procesor na vyššie frekvencie, skrátenie času vykonávania väčšiny inštrukcii
- umožňuje *protected mód (PM)*, možnosť adresovať "až" 16MB pamäte, 4 rôzne úrovne opravy programu
- real mód, (RM)* režim kompatibilný so staršími verziami, programi napísane v RM nepracujú v PM -> nebola možnosť navrhnuť program, ktorý by využíval všetky možnosti procesoru a zároveň používal existujúce programy



(reakcia developerov)

- prepnutie z rm do pm jedine resetom
- vybavený jednotkou MMU (stránkovanie), možnosť používať virtuálnu pamäť s 30bit adresov (adresovanie až 1GB virtualne)

Tretia generácia:

- prvý 32bit x86 procesor s označením 80386
- plná spätná kompatibilita s predchádzajúcimi verziami, ale predchádzajúcich 8 x16bit registrov bolo nahradených za 32bit
- nové inštrukcie
- ďalší mód: virtual mod: dovoľoval po prepnutí do PM vykonávať programy napísané pre RM (spätná kompatibilita pre nové OS so staršími programmi)



(znova reakcia developerov)

- pridanie radiča pre vyrovnávaciu pamäť
- implementácie L 1 Cache (veľkosť jednotky až desiatky kB)
- modernizovaná virtuálna pamäť (až 64TB), zachované až po 64bit procesory

-doteraz procesory pracovali len s INT, ak chceli FP, museli počítač rozšíriť o matematický koprocessor špeciálny pre práve jeden typ procesoru. Ant

Štvrtá generácia:

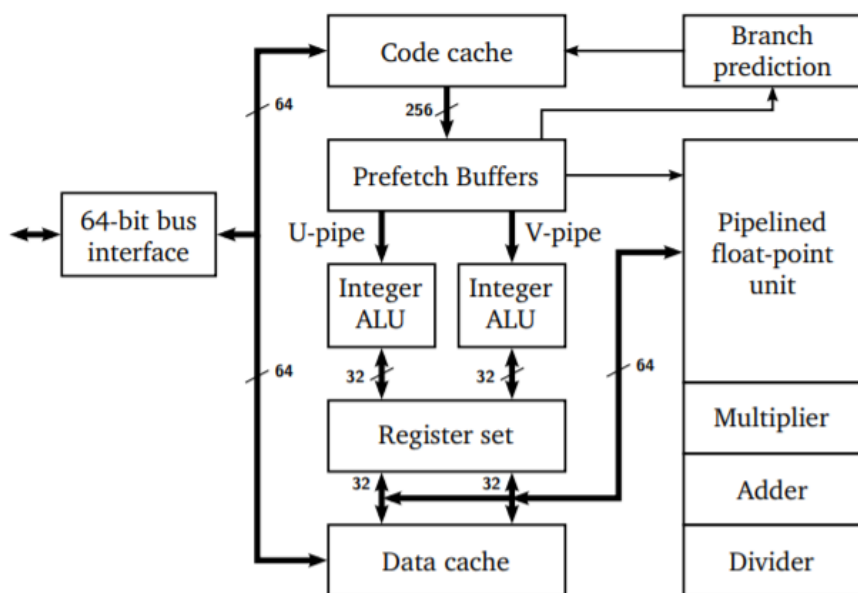
- zostáva 32bitová, 5x viac tranzistorov 1.2M-275k
- dvojnásobný výkon oproti 80386 pri rovnakej frekvencii, vylepšenie ALU, dlhšia fronta instrukcií a lepšia priepustnosť medzi jednotlivými jednotkami procesora
- vyrovnavacia pamäť L1 priamo v procesore, 8kB, spoločná pre kód aj data
- vylepšenie MMU -> urychlená činnosť v PM
- integrácia matematického koprocessoru (pripojený na vnútornú zbernicu)

Piata generácia:

Intel Pentium

- superskalárna architektúra, 2 paralelné ALU: procesor mohol spracovávať 2 jednoduché inštrukcie súčasne (ideálny prípad), pri zložitejších inštrukciách pracovali naraz
- tu už prišlo rozdelenie L1 pamäte pre data a pre program
- po prvýkrát sa objavuje jednotka na predikciu skokov
- zostáva oddelená jednotka na INT a na FPU
- o pár rokov neskôr bola pridaná MMX(MultiMedia eXtension) jednotka pre podporu multimedialných inštrukcií

Pentium Block Diagram



Šiesta generácia:

Intel Pentium Pro

Viz Rodina P6- už spracované

Intel Pentium II

- pokračovateľ PPro
- kvôli vyššej frekvencii bolo nutné frekvenciu L2 cache znížiť na polovicu frekvencie procesoru
- najvykonnejšia verzia predstavená v 1998 512kB L2, nazývané Xeon

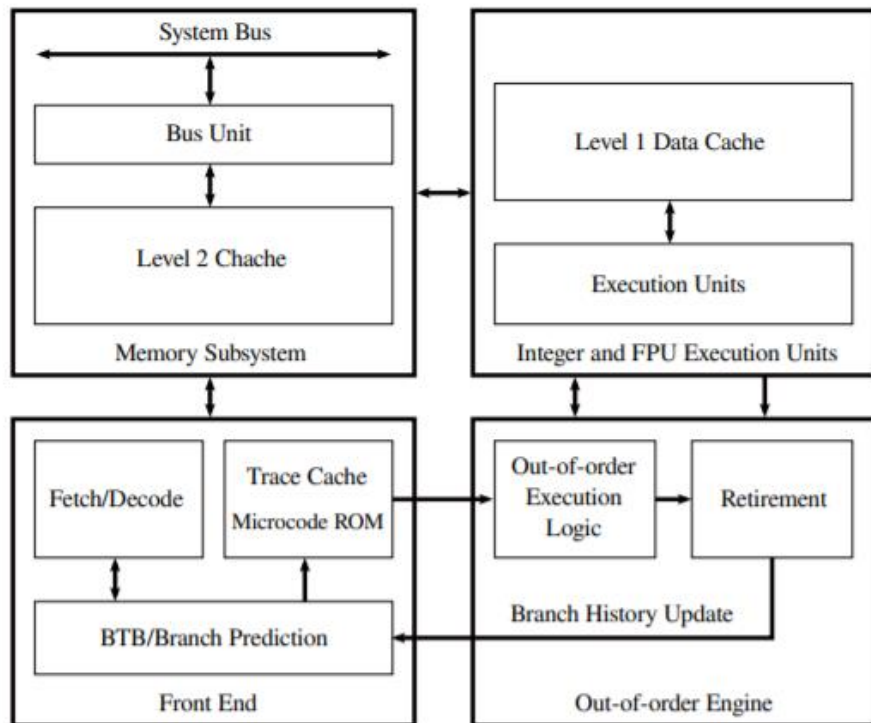
Intel Pentium III

- nástupca 2ky
- naskôr mala samostatnú L2, neskôr integrovanú

-rozšírené o provádející jednotky, například SSE (streaming SIMD extensions), zlepšuje predikciu skokov, minimalizacia spotreby : vhodné pre prenosné počítače

Intel Pentium 4

- nová mikroarchitektúra NetBurst, sklamanie, pri rovnakej frekvencii jako PP3 mal podobný výkon + sa výrazne viac prehrieval
- procesor rozdelený na 4 logické jednotky, kde aj tu dekodovacia jednotka predava mikro ops do banky instrukcii, provádející jednotka vykonáva instrukcie out of order
- L1 instruction cache schovana pod Trace Cache, obsahuje dekodovane mikro-ops, čím šetrí prácu dekodovacej jednotke (nemusi opakujúce sa sekvencie kodu znova dekodovat)
- celočíselné jednotky pre jednoduché instrukcie maju oznacenie 2x(pracuju na dvojnásobnej frekvencii procesoru)
- dvojurovňové dekodovanie: jedna predikcia pracuje pred dekodovaním, druhá pomaha pri predávaní dekodovaných inštrukcii do banky inštrukcii



Obrázek 4: Zjednodušené blokové schéma procesoru Pentium 4

- pri P4 prešiel intel na 64bit architekturu
- V procesoru byly registry rozšířeny na 64 bitů, zdvojnásoben jejich počet a datová sběrnice byla 40 bitová.
- pre dosiahnutie potrebného výkonu bola implementovana 30urovňove zretazenie
- museli byt taktovane na 3GHz, prehrievali sa

Intel Pentium M

- určený pre prenosné počítače, nízka spotreba energie
- nebol postavený na P4 ale na PIII (lebo bol energeticky nenáročný)
- priamo na čip umiestnili 1MB L2 cache => procesor při 1.5GHz mal lepší výkon jako P4 při 2.5GHz při tretinovej spotrebe energie

Intel Core, Core Duo, Core Solo

- pokračovanie rady PM
- rozšírenie adresnej sběrnice na 36bit
- prechod na 65nm technológu => bolo možné implementovať na jeden čip dve jadrá procesoru
- pamat L2 bola rozšírená na 2-4 MB

Intel Atom

- nízkopříkonový pro ntb a také pičoviny, architektúra Bonnell
- implementovaný HyperThreading (fronta instrukcí, registre se zdvojnásobí => 1 jádro obsahuje 2 logické jádra)
- Instrukce vybrané z paměti jsou dekodovány dvěma jednotkami XLAT/FL a ukládány do fronty. Z této fronty mikro-operací si je vybírají prováděcí jednotky, a to buď FP/SIMD nebo celočíselná. Výpočet adres (AGU) a řízení datové vyrovnávací paměti zajišťuje jednotka správy paměti. Komunikace s okolím je pak řízena jednotkou BIU
- ide o minimalizaci spotřeby

Intel Itanium

- RISC procesor pro výkonné servery
- 64bit, nová instrukční sada různá od x86
- využívá paralelismus na úrovni vykonávaných instrukcí a potřebné pořadí instrukcí musí být připraveno překladačem
- problém při kompatibilitě s 32bit x86

Jaká architektura je vyobrazena na obrázku? Popište jednotlivé části W, X a Z. Popište, co by se stalo, kdyby část X chyběla? V čem spočívá univerzálnost počítače

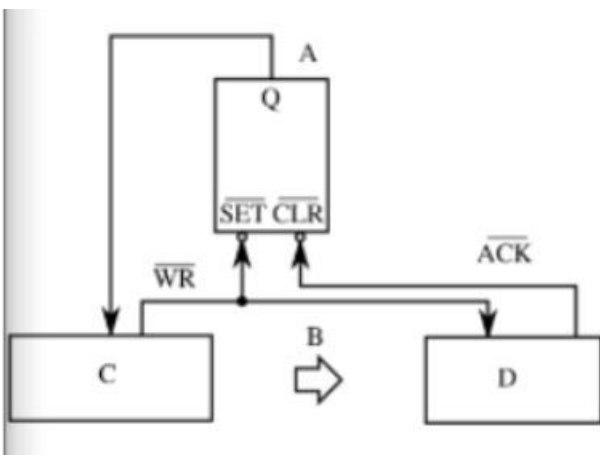
1) von Neumannova

Y – CPU X – paměť W – periférie Z – zbernice

Ak by chýbalo X tak by nemal odkiaľ brať instrukcie, nebol by program podľa ktorého sa má spracovávať, a nemal by kde zapisovať dáta z výsledkov alebo dáta z vstupných zariadení.

Univerzálnosť počítača spočíva v tom, že jeho funkčnosť alebo spracovanie závisí od obsahu jeho pamäte, teda jeho štruktúra nezávisí od vykonávanej úlohy.

Jaký spôsob komunikácie s perifériami je na obrázku? Co jsou bloky označené A, B, C a D? Jak tato komunikace funguje a v čem jsou její výhody a v čem nedostatky.



Technika podmieneného výstupu dát (neuplný režim)

A – indikátor B – Data C – Počítač D – výstupné zariadenie (záleží podľa šípky ak bude naopak tak vstupné)

Pre vstup -> ak sú poskytované platné dáta zo vstupného zariadenia STB sa nastaví na Q = 1 a potom pomocou impulzov na RD sa prenešu samotné dáta. Po prenose sa indikátor STD nastaví späť na 0

Pre vystup -> pocitac vysle impulz po WR na prepis dato do vystupneho zariadenia potom nastavi indikator ACK na 1. Vystupne zariadenie po prevzati dat nastavi Ack na 0. Takto je pc zdeleno ze moze poslat dalsie data Je to len jednosmerny prenos. Jednoduchsi na konstrukciu, menej casti

Jaké jsou obecné vlastnosti vstupních a výstupních bran u mikropočítačů?

K čemu se tyto základní periférie využívají?

Bývá obvykle organizována jako skupina 4 nebo 8 jednobitových vývodů, kde lze současně zapisovat i číst logické informace 0 a 1. U většiny bran lze jednotlivě nastavit, které bitové vývody budou sloužit jako vstupní a které jako výstupní. „pull-up“ nebo „pull-down“ odpory, které zajišťují po přepnutí do vstupního režimu definovanou hodnotu logické hodnoty 0 nebo 1. Taktiez analog vystup a vstup pomocou adprevodnikov alebo PWM

Které základní konstrukční vlastnosti procesorů RISC přinesly zvýšení výpočetního výkonu a jak?

data jsou z hlavní paměti vybírána a následně ukládána výhradně jen pomocí dvou instrukcí LOAD a STORE, v každém strojovém cyklu by měla být dokončena jedna instrukce, instrukce mají pevnou délku a jednotný formát, je použit vyšší počet registrů, složitost se z technického vybavení přesouvá částečně do optimalizujícího kompilátoru, používat zřetěžené zpracování instrukcí, mikroprogramový řadič může být nahrazen rychlejším obvodovým řadičem

Co představují v počítači jednotlivé úrovně hierarchické uspořádání paměti? Proč je takové uspořádání paměti potřebné či nutné?

1)reg 2)L1 cache 2)L2 cache 3)L3 cache 4)L4 cache 5)main operating memory 6)SSD/HDD 7)disk array 8)magnetickepasky

Cim su vysie tym su rychlejsie ale aj drahsie a mensie

U procesorů Intel/AMD jsou u strojových instrukcí jaké obvyklé kombinace operandů? Ukažte příkladech. Lze někdy využít i dva paměťové operandy?

operandy su v kombinacii vacsinou register a register, register a konstanta, pamat a register. Napr. add <register>, <konstanta>, add <register>, <register>, mov <register>, <pamat>, s pouzitim dvoch pamatovych operandov som sa nestretol, alebo si na to nespominam, kedze RISC architektura je v presile v dnesnych pocitacoch a tam sa pristupuje do pamate len presunovymi instrukciami, tak si myslim ze mozno nejaka CISC instrukcia by bola schopna pouzit dva pamatove operandy

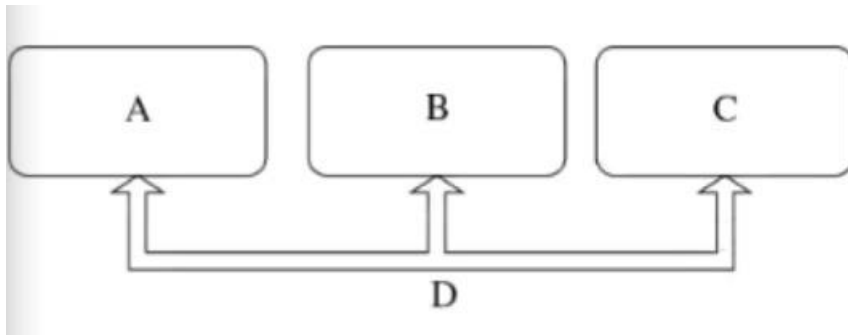
měli bychom znát tyto zásady - nepoužívat větvení v jednotlivých kernel vláknech - nedělá to dobře schedulerům, který organizuje a přiděluje práci jednotlivým vláknům ve warpu - minimálně přenášet data mezi pamětí device a pamětí host - pokud už přenášíme data mezi device a hostem, měli bychom využít pipelining - správně inicializovat počet bloků, dimenze mřížky - všechny vlákna musí vykonávat v podstatě stejnou činnost, aby docházelo k masivnímu paralelizmu - používat cudu na specifické problémy - trénování modelů v ai - nepoužívat cudu na některé problémy - např. práce s databází

Jaké zásady programování by měl programátor znát o technologie CUDA?

Jaké zobrazovací technologie je vyobrazená na obrázku? Popište jednotlivé části 1, 2, 3 až 7. Vysvětlete princip fungování. Kde vzniká barva jednoho bodu?

je zobrazena technologie LCD 1 = backlight - bílé podsvícení 2. polarizační filtr 3. začátek tekutého krystalu 4. prostředek tekutého krystalu, zde dochází k ovlivnění směru světla 5. konec tekutého krystalu 6. polarizační filtr, který propustí jenom správné světlo 7. svítící bod - subpixel backlight vygeneruje světlo, které projde polarizačním filtrem, který ho otočí, poté putuje do tekutého krystalu, kde se změní jeho směr - tekutý krystal je ovládán napětím předem definovaným způsobem, projde krystalem, poté narazí na další polarizační filtr a pokud má správný směr, tak pr barva jednoho bodu vzniká v tekutém krystalu - TN twisted nematic struktura

Jaká architektura počítače je vyobrazená na obrázku? Popište jednotlivé bloky A, B, C a D. Co to jsou strojové instrukce, kde je v počítači najdeme? Kdo nebo co je vytváří a jak je počítač využívá nebo k čemu slouží?

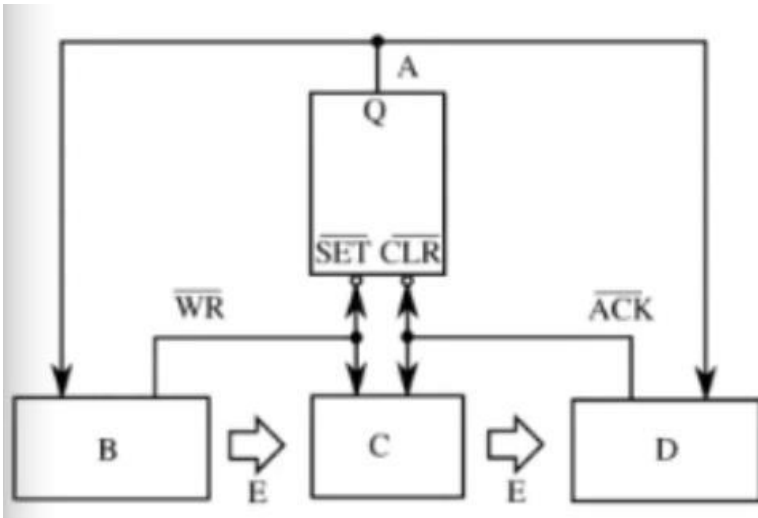


1) von Neumannova

A – CPU C – paměť B – periférie D – sběrnice

Jaký způsob komunikace s perifériemi je na obrázku? Co jsou bloky označené A, B, C, D a E? Jak tato komunikace funguje a v čem jsou její výhody a v čem nedostatky?

Technika výstupu dat s vyrovnávací pamětí



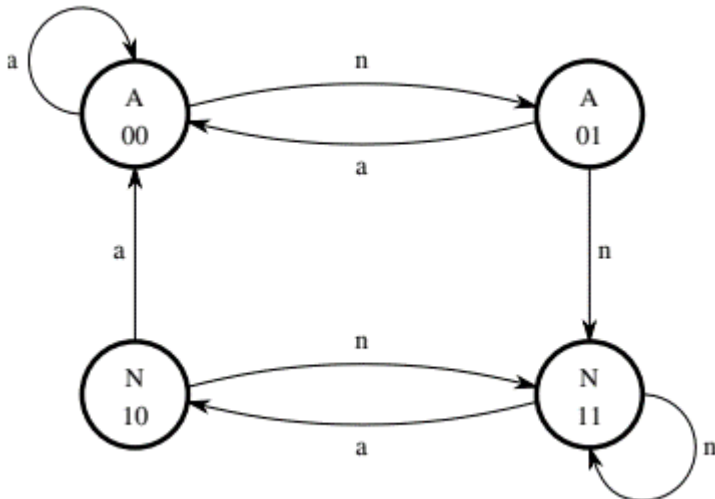
A – indikato B – pocitac C – Register D – vystupne zariadenie E – Data

Semafor informuje procesor o připravenosti dat ve vyrovnávací paměti. Pro periférii tento semafor představuje informaci, zda je možno do vyrovnávací paměti zaslat další data, či zda obsah paměti nebyl ještě procesorem přečten.

Proč se u procesoru RISC využívá více registrů? Jak funguje zřetěžené zpracování instrukcí? Jaké zrychlení teoreticky přinese N úrovně zřetěžení? Jaké problémy může zřetěžení způsobovat?

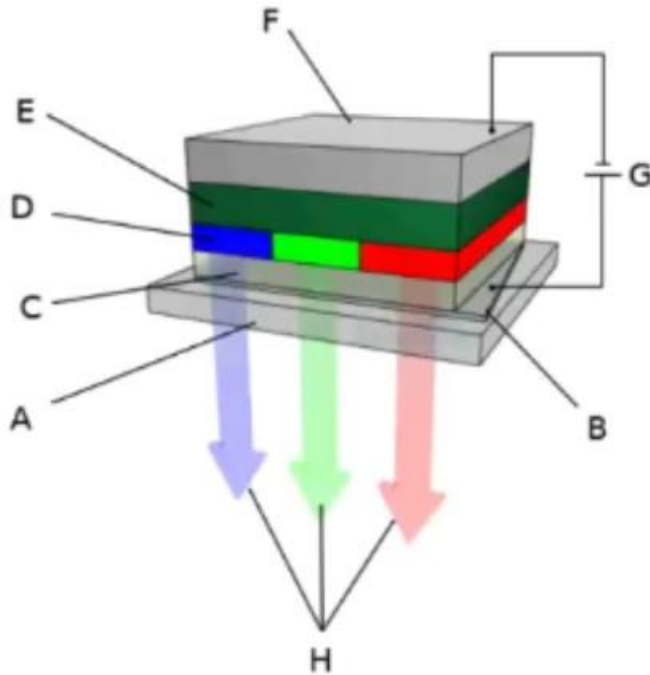
Zvýšení výkonu zřetěžení - paralelismus = umožňuje až n krát zvětšit rychlost procesoru, kde n je řád paralelní fronty, prakticky je to degradováno nutností používat registry pro uložení jednotlivých kroků, tím, že ne vždy trvají operace stejně dlouho - řešíme nopenem více dočasné paměti - vyplynulo z zmenšení počtu instrukcí, které šahají do paměti - nyní pouze load/store (více cache, registrů) používáme obvodový řadič místo pomalejší mikroprogramového, který jsme museli použít u cisc architektury pro implementování většího počtu instrukcí

Nakreslete schéma fungování dvoubitové predikce skoků. Jednotlivé části označte 1, 2,3. atd. a do textu je popište. K čemu se v procesoru predikce skoků využívá a proč?



Jde o čtyřstavový automat, kde stav A predikuje provedení skoku, zatím co stav N říká, že skok se provádět nebude. Přechody a a n označují, zda se naposledy skok prováděl, či nikoliv. Je tedy patrné, že pouze pokud se skok dvakrát po sobě provede, může se změnit predikce z N na A a obráceně. V ustáleném stavu se tedy predikce nastaví na $A - 00$ nebo $N - 11$.

Jaká zobrazovací technologie je na obrázku? Popište jednotlivé části A, B, C až H. Vysvětlete princip fungování. Čím je určena barva jednoho bodu?



OLED

A) skleněná deska B) Transparentní anoda
C) Přenos díry D) organické emitory E) přenos elektronů F) kovová anoda G) 2-10V

Po přivedení napětí na obě elektrody se začnou elektrony hromadit na straně organické vrstvy

blíže k anodě. Díry, představující kladné částice, se hromadí na opačné straně blíže ke katodě. V organické vrstvě začne docházet ke „srážkám“ mezi elektrony a dírami a jejich vzájemné eliminaci, doprovázené vyzářením energie ve formě fotonu. Tomuto jevu říkáme rekombinace

Jak jsou na pevném disku organizována data? Jaká je nejmenší paměťová jednotka na pevném disku, kterou lze přečíst či zapsat?

Data jsou na disk ukládána v bajtech. Bajty jsou uspořádány do skupin po 512, nebo-li sektorů. Sektor je nejmenší jednotka dat. Sektory jsou seskupeny do stop. Stopy jsou uspořádány do skupin zvaných cylindry nebo válce. Předpokladem je, že disk má nejméně dva povrchy. Systém adresuje sektory na pevném disku pomocí prostorové matice cylindrů, hlav a sektorů

Výhody von Neumannovy oproti Harvardské

9

Von Neumann je jednodušší na výrobu, lebo má len jednu zbernicu, jednu pamäť, jeden typ prístupu k pamäti pre inštrukcie a data, programátor si určuje kde v pamäti budú data a kde program/inštrukcie.

Harvardská používa 2 pamäte čo je výhodou pri programovaní, nedokáže teda prepísať svoje inštrukcie inštrukci ako von Neumann, keďže v jednej pamäti sú data a v druhej program, má dve sbernice, teda je možné pristupovať po data a inštrukcie paralelne.

...

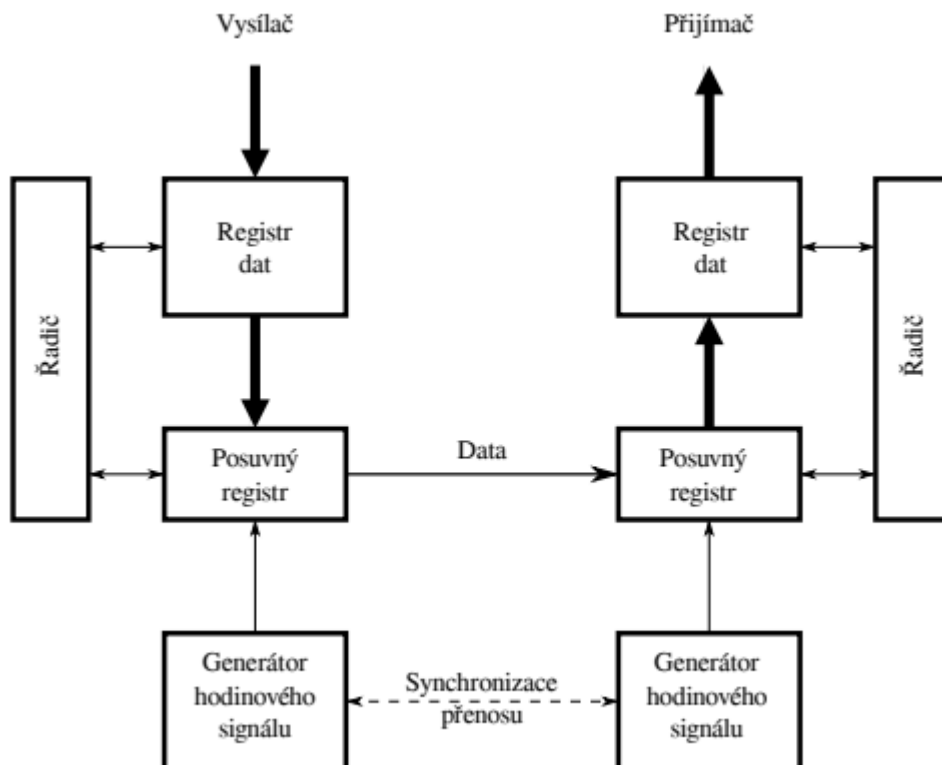
ROZDIEL MEDZI KOLMIM a podielnym zapisom

Jednotlivé bity, interpretované jako malá opačně orientovaná magnetická pole, se uchovávají jako vektory rovnoběžné s plotnou disku. Vektory magnetické indukce jednotlivých bitů zde nejsou orientovány rovnoběžně s plotnou, nýbrž kolmo na ni. Tím je možné zvýšit kapacitu pevných disků až desetinásobně a přiblížit se hranici hustoty 1 terabit na čtverečný palec. S novou technologií však přichází i větší technologická náročnost řešení. Pro potřeby kolmého zápisu bylo nutné vyvinout novou diskovou hlavu pro zápis a přidat pod datovou vrstvu ještě vrstvu z magneticky měkkého materiálu.

JAKY JE PRÍNOS VACŠIEHO POČTU REGISTROV PRE RISC

navýšení počtu registrů souvisí s omezením komunikace s pamětí na dvě instrukce LOAD a STORE. Pokud ostatní instrukce nemohou používat paměťové operandy, je třeba mít v procesoru uloženo více dat. Jednotná délka instrukcí dovozuje rychlejší výběr instrukcí z paměti a tím zajišťuje lepší plnění fronty instrukcí (viz další kapitola). Jednotný formát pak zjednodušuje dekódování instrukcí. Aby uvedený princip zřetězení měl co největší přínos, musí být všechny fáze zpracování stejně časově náročné. A při RISC architektuře platí, že každá jedna instrukcia sa dokonci v jednom cycle

RS232



Bitová predikce

Bitová predikce - statická = při příkladu dynamická - 1 bitová - pamatuje si předchozí stav, podle toho predikuje dynamická - 2 bitová - pamatuje si 2 předchozí stavy, podle toho predikuje Superskalární architektura - provádí se obě větve i skok i bez skoku; na základě toho, který případ dále nastane se jedna nebo druhá část zahodí - na modernějších Také můžeme se skokem počkat, pokud instrukce, které jsou za skokem na něm nezávisí Pokud bychom tyto metody používali, museli bychom zahazovat celou frontu instrukcí

5

kernel je funkcia ktora je vykonavana nad jednym vlaknom, grid je usporiadanie vlakien do blokov a celkovej mriezky, teda prerozdelenie dat pre jednotlivé vlakna nad ktorým bude vykonavany kernel, kernel je jadro celeho fungovania cudy, parametre sa urcuju kernel < <velkost mriezky, velkost bloku> >(parametre), mriezka je rozdelená na stĺpce a riadky, a nasledne

6

7

V/V brany - komunikácia s perifériami, ochrany proti ruseniu - vplyv z mena, Citace a casovace - na sledovanie casovych udalosti, Zdroje synchronizacie - na synchronizovanie chodu pocitaca, operacii, Seriove rozhrania - prenos dat na vacsie vzdialenosti, , D/A alebo A/D prevodniky - na premeny signalov medzi analogovymi a digitalnymi

8

Staticka pamat udrzuje informáciu pomocou klopneho obvodu, nie je nutné obcerstvovat tieto pamate, su rychlejsie, ale mensie, pri pristupe k jednotlivým pamatovým bunkám nie je nutný adresový multiplexing. Dynamicke pamate udrzuju informáciu v naboji kondenzatoru, je nutné obcerstvovat tieto pamatove bunky, pretoze priblizne kazdych 10ns stracaju svoj naboj uplne, pri pristupe je pouzivany adresovy multiplexing, su lacnejšie na výrobu

9

Von neumann je jednoduchsi na výrobu, lebo ma len jednu zbernicu, jednu pamat, jeden typ pristupu k pamati pre instrukcie a data, programator si urcuje kde v pamati budu data a kde program/instrukcie. Harvardska pouziva 2 pamate co je vyhoda pri programovani, nedokazu teda prepisat svoje instrukcie instrukci ako von neumann, kedze v jednej pamati su data a v druhej program, ma dve sbernice, teda je mozne pristupovat po data a instrukcie paralelne.