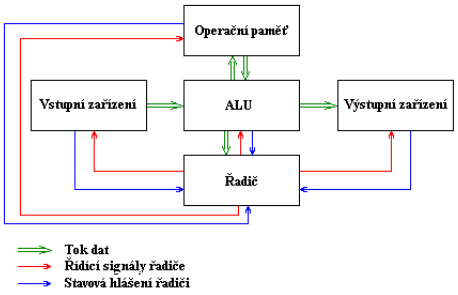
1. Klasifikace výpočetních systémů, přehled architektur.

Level 5 Problem-oriented language level
Translation - compiler
Level 4 Assembly language level
Translation - asambler
Level 3 Operation system machina level
Partial interpretetion - OS
Level 2 Instruction set architecture level
Interpretation (mikroprogram) or direkt execution
Level 1 Microarchitecture level
Hardware
Level 0 Digital logic level

HW / SW interface
- Jednouchá ISA
- CISC
- RISC
CISC versus RISC
Instrukce jsou přímo prováděny HW
Maximální průchodnost instrukcí (ILP)
Jednoduché instrukce (snadné dekódování)
Přístup do paměti jen instrukcemi Load/Store
Velké množství registrů
Pipelining

CISC (*Complete Instruction Set Computing*) - jsou procesory používané ve většině současných i dřívějších osobních počítačů. Hlavním rysem těchto procesorů je, že používají tzv. plnou instrukční sadu, nebo-li se snaží mít na každou úlohu jednu instrukci. Tyto instrukce jsou uloženy v mikrokódu, což je vlastně program vloženy o
paměti procesoru. Tento systém vytváření je z hlediska technologického jednodušší, ale instrukce se provádějí pomaleji než u obvodového řešení. Příklady: i8086, i80486, Pentium, M68040 ...
RISC (*Reduced Instruction Set Computing*) - procesory s redukovanou instrukční sadou. Obsahují jen několik základních instrukcí. Každá z nich by se měla vykonávat co nejkratší dobu, pokud možno během jediného strojového cyklu. Instrukce jsou vytvořeny obvodově a tudíž se většinou provádějí rychleji než u mikrokódového
řešení. Stejně jako je malý počet instrukcí i je malý počet způsobů adresování. Pro práci s pamětí se na rozdíl od CISC procesorů používají jen dvě instrukce (Load/Store). Všechny ostatní instrukce se vyhodnocují v registrech, kterých bývá většinou větší počet (obvykle 32). Příklady: PA-8000, Power PC, R 4200, UltraSparc II, ARM ...
V dnešní době se obě architektury přibližují. Mnohé procesory nesou rysy obou typů. Většinou mají jádro typu RISC, ale pro ostatní součástky se tváří jako procesor typu CISC. Příklady: Pentium III, Pentium 4, AMD Athlon...

Von-Neuman architekrura:



Podle tohoto schématu se počítač skládá z pěti hlavních modulů:

- **Operační paměť'** : slouží k uchování zpracovávaného programu, zpracovávaných dat a výsledků výpočtu
- **ALU - Arithmetic-logic Unit (aritmetickologická jednotka)**: jednotka provádějící veškeré aritmetické výpočty a logické operace. Obsahuje sčítačky, násobičky (pro aritmetické výpočty) a komparátory (pro porovnávání)
- **Řadič**: řídící jednotka, která řídí činnost všech částí počítače. Toto řízení je prováděno pomocí **řídících signálů**, které jsou zasílány jednotlivým modulům. Reakce na řídící signály, stavy jednotlivých modulů jsou naopak zasílany zpět řadiči pomocí **stavových hlášení**
- **Vstupní zařízení**: zařízení určená pro vstup programu a dat.
- **Výstupní zařízení**: zařízení určená pro výstup výsledků, které program zpracoval

Ve von Neumannově schématu je možné ještě vyznačit dva další moduly vzniklé spojením předcházejících modulů:

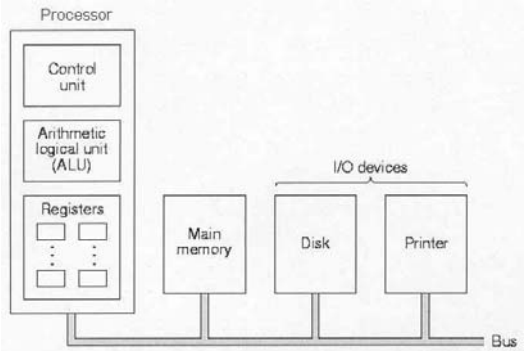
- **Procesor**: Řadič + ALU
- **CPU - Central Processor Unit (centrální procesorová jednotka)**: Procesor + Operační paměť

Princip činnosti počítače podle von Neumannova schématu

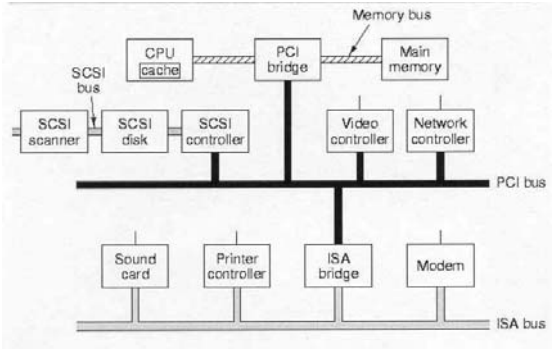
1. Do operační paměti se pomocí vstupních zařízení přes ALU umístí program, který bude provádět výpočet.
2. Stejným způsobem se do operační paměti umístí data, která bude program zpracovávat
3. Proběhne vlastní výpočet, jehož jednotlivé kroky provádí ALU. Tato jednotka je v průběhu výpočtu spolu s ostatními moduly řízena řadičem počítače. Mezivýsledky výpočtu jsou ukládány do operační paměti.
4. Po skončení výpočtu jsou výsledky poslány přes ALU na výstupní zařízení

- Podle von Neumannova schématu počítač pracuje vždy nad jedním programem. Toto vede k velmi špatnému využití strojového času. Je tedy obvyklé, že počítač zpracovává paralelně více programů zároveň - tzv. multitasking
- Počítač může disponovat i více než jedním procesorem
- Počítač podle von Neumannova schématu pracoval pouze v tzv. diskrétním režimu.
- Existují vstupní / výstupní zařízení I/O devices, která umožňují jak vstup, tak výstup dat (programu)
- Program se do paměti nemusí zavést celý, ale je možné zavést pouze jeho část a ostatní části zavádět až v případě potřeby

Počítač se sběrnicovou koncepcí:



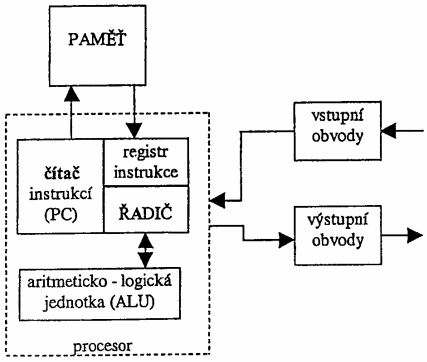
Anatomie moderního PC



Moderní počítače využívají tzv. *sběrnice*. Jejím úkolem je přenášet data a veškeré signály v rámci počítače mezi jeho jednotlivými částmi. Sběrnice podporuje modularitu systému (je možné relativně jednoduše přidávat či ubírat další moduly – části počítače).
Sběrnici si můžeme rozdělit na tři základní části:
- datová sběrnice
- řídící sběrnice
- adresová sběrnice

Architektura von Neumannova

Základní myšlenka této architektury spočívá v tom, že mikroprocesor má k dispozici jedinou paměť V ní se nachází program a data, přičemž není vymezeno v jaké části paměti se musí nacházet program a v jaké části data. Znamená to tedy, že jednotlivé instrukce se zpracovávají stejně jako data a je tak umožněno jednoduše modifikovat program.
Po inicializaci počítače ukazuje čítač instrukcí na adresu kódu první instrukce programu, který je uložen v hlavní paměti (obr. 3-1). Kód instrukce se přesune z paměti do registru instrukce. Řadič procesoru dekóduje instrukci a



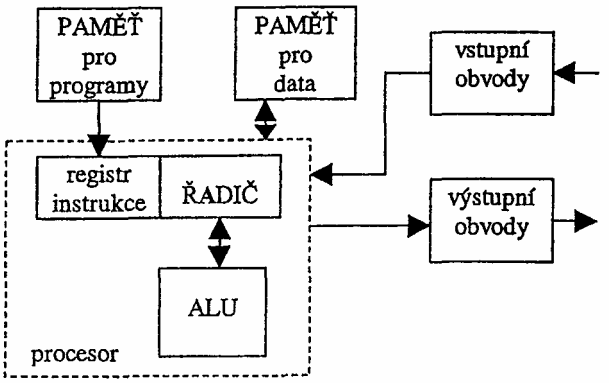
Obr. 3-1

na základě toho vygeneruje příslušné řídicí signály pro provedení operací s daty řídí provedení operací v aritmeticky-logické jednotce, přesuny dat mezi pamětí a registry ALU, případně přesuny dat mezi registry ALU a vstupními a výstupními obvody). Obsah čítače instrukcí se zvýší o příslušnou hodnotu tak, aby ukazoval na adresu začátku kódu následující instrukce programu v paměti. Dále se činnost opakuje již uvedeným způsobem.

Význačnou vlastností této architektury je univerzálnost - v paměti může být uložen dlouhý program a málo dat nebo krátký program a hodně dat. Základní vlastnosti von Neumannovy architektury lze shrnout takto:

· Struktura počítače vypadá podle obr. 3-1 a skládá se z paměti, řadiče, aritmeticko-logické jednotky a ze vstupních a výstupních obvodů.
· Instrukce, ze kterých se skládá program, a operandy (data) jsou uloženy v téže paměti.
· Činnost počítače lze změnit změnou programu v paměti.
· Počítač používá při zpracování informaci dvojkové číselné soustavy.
· Paměť je rozdělena do stejných paměťových buněk a každá buňka má svoji adresu.

Z této architektury vychází většina univerzálních mikropočítačů a minipočítačů. Nesmí nás přitom zmást, že většina mikropočítačů má část paměti realizovánu jako paměť typu ROM a část jako RAM. Takové rozdělení si však zvolil uživatel počítače z praktických důvodů - paměť typu ROM může být principielně nahrazena zase pamětí typu RAM.



Harvardská architektura

Tato architektura vychází z existence dvou oddělených pamětí, a sice paměti pro data a paměti pro programy (obr. 3-2). Důvodem pro rozdělení paměti na dvě samostatné části je nebezpečí zničení programu chybou při jeho běhu nebo programátorskou chybou. Použití pamětí (nikoliv obsah) nelze přitom měnit, je pevně dáno. Oproti p“redchozí architektuře se tato vyznačuje malou univerzálnosti.

Z toho vyplývá i použití pro jednoúčelové automaty, kapesní kalkulátory, některé monolitické mikropočítače, některé monolitické koprocesory. Myšlenky této architektury využívají také moderní procesory, které obsahují oddělené vyrovnávací paměti pro data a pro instrukce (kap. 6.2).

2. Instrukční cyklus, výběr instrukce, provedení instrukce, algoritmy základních operací.

Instrukční cyklus = čas potřebný pro výběr a provedení instrukce

Instrukční cyklus (IC - Instruction Cycle) tvoří několik strojních cyklů potřebných k provedení jedné instrukce.

Stav čekání (WAIT) je stav, který může nastat v průběhu strojního cyklu. Mikroprocesor ve stavu čekání vkládá do strojního cyklu tzv. čekací takty Tw a testuje přitom externí signál typu TEST nebo READY. Jestliže je uvedený signál aktivní, svědčí to o připravenosti pomalejších okolních obvodů zpracovávat další informaci nebo o tom, že okolní obvody dokončily požadovanou činnost a mají k dispozici její výsledek. Teprve pak mikroprocesor pokračuje v provádění strojního cyklu. Tento stav se využívá vždy při spolupráci mikroprocesoru s pomalejším periferním zařízením.

Stav HOLD je stavem odpojení mikroprocesoru od společné datové i adresové sběrnice. Odpojení se realizuje tak, že příslušné datové a adresové vývody mikroprocesoru jsou uvedeny do stavu vysoké impedance. Princip je stejný jako u třístavových logických členů. Stav HOLD se využívá při tzv. přímém přístupu do paměti (viz kap. 5.6), kdy se přenáší do paměti dlouhé bloky dat bez účasti mikroprocesoru (tzv. DMA přenos) nebo při sdílení sběrnice větším počtem mikroprocesorů.

Instrukce: přesuny dat, aritmetické operace, logické operace, posuvy a rotace, bitové oprace, nepodmíněné a podmíněné skoky, řídící instrukce

Fáze instrukčního cyklu

• Instrukční cykl = doba zpracování jedné instrukce.
• 4 základní fáze:
– **Fetch** – čtení kódu instrukce z paměti,
– **Decode** – dekódování instrukce,
– **Execute** – provedení instrukce,
– **Write Back** – zpětný zápis výsledku do registrů procesoru.
• Podle typu instrukce mohou být jednotlivé fáze různě dlouhé – vyžadují různý počet **strojových cyklů**.

Výběr instrukcí je řízen registrem:

- čítač instrukcí (adres)
- PC (Program Counter)
- IP (Instruction Pointer)

Po provedení instrukce se zvyšuje o délku instrukce. Plní se např. instrukcí skoku.

Provádění instrukcí:
Základní model:
-kód programu (instrukce) jsou uloženy ve vnější paměti
- Procesor je nejprve musí načíst

Proudové zpracování instrukcí
• Při sekvenčním zpracování je využita vždy jen část CPU.
• Proudové zpracování (pipeline) : v CPU se zpracovává několik instrukcí, každá v jiné fázi.
• Požadavky:
– stejná délka zakódovaných instrukcí,
– stejná délka provádění instrukcí.

3. Mikroarchitektura, mikroprogramové řízení, formáty mikroinstrukcí

Mikroarchitektura: souhrn vlastností implementace (které uživatel na instrukční úrovni nevidí)
Vlastnosti, které se mění (časování, výkon, technologie), patří do mikroarchitekrury.

Mikroinstrukce jsou elementární příkazy. Napsat aplikační program přímo z mikroinstrukcí by bylo nesmírně složité. Proto každý mikroprocesor je vybaven pro praktické programování „přívětivější“ sadou instrukcí. Převod těchto ***instrukcí*** (používá programátor) na ***mikroinstrukce*** (těm rozumí mikroprocesor) obstarává program napsaný v mikroinstrukcích - je další podstatnou částí mikroprocesoru (jeho řadiče).

Popis mikroprocesoru :

je potřeba udělat si představu o stavbě mikroprocesoru a významu jednotlivých komponent. Zde se ovšem nebudeme zabývat fyzikální stránkou (technologie na bázi křemíku) ani stránkou obvodovou, resp. elektrotechnickou (miliony tranzistorů), nýbrž se zaměříme na strukturu logickou:

Registry
každý mikroprocesor pracuje s daty a programovými instrukcemi uloženými v op. paměti - tedy mimo mikroprocesor. Momentálně zpracovávaná data i instrukce si musí ukládat do svých ***vnitřních pamětí - registrů***. Mikroproc. disponuje určitým počtem registrů, jejichž velikost, počet a přesné použití se u jednotlivých typů liší.

Adresování
jde o mechanismus, kterým mikroprocesor specifikuje adresy v paměti, na nichž leží zpracovávaná data. Existuje více způsobů adresace.

Instrukční sada
musí obsahovat instrukce pro přesuny dat mezi pamětí a registry, aritmetické a logické instrukce, posuny, instrukce pro řízení programu a několik systémových instrukcí.
S každým novým typem mikroprocesoru se počet instrukcí zvětšuje (Např. u Pentia MMX se instrukční sada rozšířila oproti Pentiu o 57 nových příkazů pro přehrávání videa, generování zvuku a grafiky).

Přerušovací systém
Přerušení je signál, který k mikroprocesoru vyšle některé hardwarové zařízení nebo program a tím se snaží zabrat činnost mikroprocesoru na nějaký čas pro sebe. Klasickým příkladem je stisk klávesy na klávesnici. Mikroprocesor musí přerušit svoji činnost a povel daný klávesou zpracovat. Všechny moderní mikroprocesory mají ***vektorový systém přerušení***.To znamená, že každé ***přerušení je identifikováno svým číslem***.
Na určitém místě v paměti je uložena ***tabulka vektorů přerušení***. Každý vektor z této tabulky ukazuje na určitou adresu v paměti, kde je uložen obslužný program pro dané přerušení.
N-té přerušení tedy spustí (přes **N**-tý vektor přerušení) **N**-tý program, který zpracuje požadavek zdroje přerušení.
Mikroprocesor musí obsahovat i mechanismus, kterým se přerušení dočasně zakáže.

Správa paměti
Paměťové adresy, které jsou zapsány v programu nejsou definitivní - skutečné. Jednotka správy paměti mění tyto zadané adresy tak, jak je to momentálně výhodné pro oper. systém. Důvodem tohoto „překladu“ adres je lepší využití operační paměti.
Druhým, neméně důležitým úkolem jednotky správy paměti je zabezpečení ochrany paměti. V moderních operačních systémech pracuje současně několik programů (mezi nimi i samotný operační systém). Jednotka správy paměti musí zabránit každému programu v narušení činnosti ostatních programů. Nesmí se např. stát, že by dva programy současně používaly stejnou adresu paměti. Pro splnění tohoto úkolu nabízejí mikroprocesory nejméně dva režimy práce: - systémový, v němž je povoleno vše
- uživatelský, kdy je povoleno jen to, co povolil program běžící v systémovém

režimu . Jednotka správy paměti tohoto mechanismu umí využívat a může tak uživatelským programům zabránit v provádění destrukčních prací.

Architektura mikroprocesoru

Tím rozumíme schopnost mikroprocesoru zpracovávat posloupnost instrukcí.
Starším řešením (až po mikroprocesory typu ***486***) je sekvenční zpracování instrukcí, tzn. instrukce se zpracovávají jedna po druhé.
Moderní řešení (Pentia, PowerPC) se vyznačují architekturou ***superskalární*** - ta dokáže zpracovávat i několik instrukcí najednou.Této vlastnosti lze dosáhnout různě:
- buď zdvojením některých funkčních celků (Pentium)
- nebo promyšleným návrhem mikroprocesoru, díky kterému mohou jednotlivé celky mikroprocesoru pracovat nezávisle na sobě (PowerPC).

Je však zřejmé, že ani superskalární mikroprocesor nemůže souběžně zpracovávat instrukce vždy. Pokud v programu následují dvě instrukce, z nichž první připravuje data pro druhou, musí se zpracovávat sekvenčně.
Dalším zrychlujícím prvkem je ***pipelining*** . Díky tomuto mechanismu mohou i nesuperskalární mikroprocesory ve skutečnosti zpracovávat více instrukcí najednou. Zpracování každé instrukce je rozloženo do více fází (dekódování,vyhledání parametrů, atd.). Jakmile je jedna fáze instrukce hotova, postoupí tato instrukce do další fáze. Uvolněnou fázi začne hned využívat následující instrukce.Celý proces připomíná pásovou výrobu, kdy do jednotlivých výrobních fází vjíždí jeden výrobek za druhým.

Vnitřní šířka dat
určuje schopnost mikroprocesoru najednou zpracovat určité množství bitů. Čím více bitů mikroprocesor zpracuje, tím je rychlejší.

Řadič – konečný automat
- stav: generování řídících signálů
- hrany: podmínky přechodu

Řízení:
Problémy spojené s řízením: - reálné procesory mají stovky stavů, tisíce interakcí mezi stavy, grafický návrh automatu je por reálné architektury nemožný
Řešení: Mikroprogramování – Wilkesův automat – návrh založený na software – řídící signály-> pole mikroinstrukcí, posloupnost mikroinstukcí -> mikroprogram
Mikrprogramování – návrh mikroinstrukcí a jejich časování + ošetření vyjímek

Formát mikroisntrukcí MIPS:
Mikroinstrukce – abstrakce řízení datových cest

Pole	Specifikuje
ALU control	operace ALU v daném cyklu
SRC1	zdroj 1. operandu ALU
SRC2	zdroj 2. operandu ALU
Registr Control	registr read/write, zapisovaná data
Memory	read/write pro paměť, zapisovaná data, cílový registr
PCWrite Control	zdroj pro PC
Sequencing	výběr příští mikroinstukce

Režim výběru:
- inkrementace – je-li aktuální adresa mikroinstrukce A, potom příští 32-bitová adresa mikroinstrukce leží na A+4
- větvení – skok na instrukci, která načítá příští mikroinstrukci
- řízený výběr – příští mikrointrukce je vybrána podle vstupu řadiče

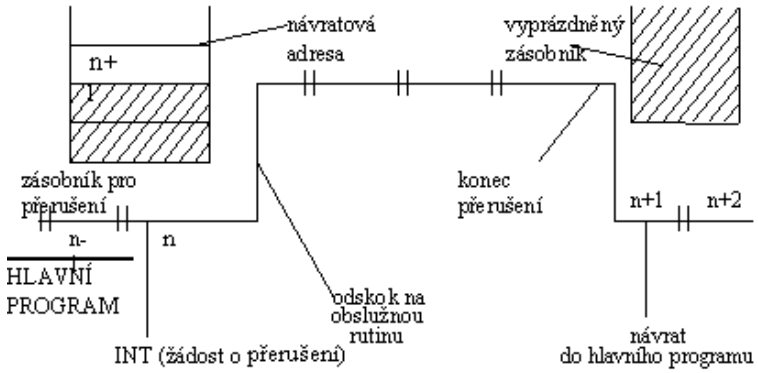
Problémy mikroprogramování:
HW implementace – podobné řízení s pevným řadičem, stav je uložen v registru, přechodová fce v ROM nebo PLA, možnosti: mikroprogram používá čítač

Výjimky (inetrní události) vs. Interupty (externí)
- detekce výjimek – vyžaduje speciální hw
- oštření výjimek – vyžaduje více hw - cena

4. Přerušovací systém.

Přerušovací systém

Princip přerušení:



n -1, ***n*** atd. jsou adresy instrukcí. Pokud při běhu programu dojde k žádosti o přerušení, dokončí se právě probíhající instrukce a systém zajistí úklid kontextu do zásobníku pro přerušení. Kontext obsahuje obsahy střadače registrů v době přerušení a dále je uložena návratová adresa n +1 hlavního programu. Je proveden odskok na adresu instrukce pro obslužnou rutinu (podprogram), přerušení je obslouženo. Instrukce ***konec přerušení*** zajistí přepsání kontextu zpět do registru (dojde k vyprázdnění zásobníku pro přerušení a přes návratovou adresu hlavního programu je tento spuštěn a dále běží.

Přerušení může být:

- **vnitřní** (chyba při běhu programu)
- **vnější** (žádost některé periferie o obsluhu dat)

Dále se dělí:

- vektorové
- maskovatelné

Maskovatelné je druh přerušení, které závisí na nastavení klopného obvodu tzv.masky, který podle stavu na výstupu povolí nebo nepovolí přerušení. K tomu je určen v příznakovém registru bit IF (Interupt flag), který při nastavení do jedničky zajistí maskovatelné přerušení.

Jelikož periferních zařízení je více než dvě, přidává se k procesorům řadič přerušení, který je řadí podle důležitosti (priority) a sděluje procesoru o jaký druh přerušení jde. Při každém přerušení se volá obslužná rutina, která je k tomu určena.

Procesor pracuje s **vektorovým systémem přerušení**. Tabulka vektorů je uložena v operační paměti, většinou od adresy 0 a každý vektor má velikost 4B. Je to celá adresa (segment i offset) a v operační paměti zabírá asi 1kB. Procesor rozezná 256 přerušení (0-255). Některé vektory jsou pevně určeny a nelze je měnit.

Tabulka vektoru:

00000	Dělení nulou	vektor 0
00004	Krokování	vektor 1
00008	Nemaskovatelné přerušení	vektor 2
00012	INT 3	vektor 3
	Aritmetické přetečení	vektor 4
	27 rezervovaných vektorů	
	224 volných vektorů	

U maskovatelného přerušení dodá řadič přerušení i číslo vektoru, které specifikuje některou z periferií. 27 vektorů si rezervovali tvůrci systému a většinou nejsou využity.

Druh zařízení, které žádalo o přerušení a tím i obslužná rutina, která má být provedena, je možné zjistit několika způsoby:

- **metoda dotazování** (pooling) - všechna zařízení generují stejný požadavek o přerušení a tedy spouštějí i stejnou obslužnou rutinu. Ta nejprve zjistí, které zařízení žádalo o přerušení, a pak provede příslušnou obslužnou rutinu. Tato metoda neumožňuje rekonfigurovat snadno systém.
- **metoda pomocí kódu typu přerušení** - zařízení žádající o přerušení vyšle jako odezvu na akceptování požadavku o přerušení procesorem svůj speciální kód tzv. kód typu přerušení. Podle něj procesor přejde k provedení obslužné rutiny přerušení. Metoda je nevýhodná v tom, že při ní může dojít ke kolizi kódů typu přerušení jednotlivých zařízení.
- **řadič přerušení** - je v současné době nejpoužívanější. Řadič přerušení má určitý počet vstupů pomocí nichž žádájí jednotlivá zařízení o přerušení. Aktivuje-li některé zařízení žádost o přerušení, pak řadič vyšle procesoru souhrný požadavek o přerušení. Je-li tento požadavek procesorem akceptován, potom řadič předá procesoru kód typu zařízení žádajícího o přerušení. Protože je řadič přerušení programovatelným obvodem, musí být před zahájením činnosti vhodně naprogramován. Při jeho programování se přiřadí vstupům, na nichž jsou napojena zařízení žádající o přerušení, určité kódy typů přerušení. V průběhu činnosti můžeme řadič přerušení dále modifikovat např. nastavovat prioritu jednotlivých vstupů, maskovat (zablokovat) některé vstupy apod.

5. Sběrnice

Pod pojmem sběrnice obecně rozumíme soustavu vodičů, která umožňuje přenos signálů mezi jednotlivými částmi počítače. Pomocí těchto vodičů mezi sebou jednotlivé části počítače komunikují a přenášejí data.
Zařízení jako jsou procesor, koprocesor, cache paměť, operační paměť, řadič cache paměti a operační paměti a některá další zařízení jsou propojena tzv. **systémovou sběrnicí (CPU bus)**.
Osobní počítače musí být navrženy tak, aby bylo možné jejich snadné rozšiřování o další zařízení (zvukové karty, síťové karty, řadiče disků apod.). Takovéto rozšiřování je velmi často uskutečňováno pomocí tzv. **rozšiřující sběrnice** počítače (častěji označované pouze jako sběrnice), na kterou se jednotlivá zřízení zapojují. Tato rozšiřující sběrnice a zapojovaná zařízení musí tedy splňovat určitá pravidla. Takže ve výpočetní technice je pojem sběrnice také chápán jako standard, dohoda o tom, jak vyrobit zařízení (rozšiřující karty), která mohou pracovat ve standardním počítači.
Podle způsobu práce a zapojení rozlišujeme několik základních typů sběrnic:
- synchronní sběrnice: sběrnice pracující synchronně s procesorem počítače. Platnost údajů na sběrnici jednoznačně určuje hodinový signál. Tímto způsobem dnes pracuje převážná většina všech sběrnic.
- pseudosynchronní sběrnice: dovoluje zpozdit přenos údajů o určitý počet hodinových period.
- multimaster sběrnice: dovoluje tzv. **busmastering**, jedná se o sběrnici, která může být řízena několika zařízeními, nejen procesorem.
- lokální sběrnice: spočívá ve vytvoření technické podpory toho, že se náročné operace s daty realizují rychlou systémovou sběrnicí. Tato systémová sběrnice se prodlouží a umožní se tak přístup na ni i ze zásuvných modulů dalších zařízení. O rozvoj lokálních sběrnic se nejvýrazněji zasloužili výrobci videokaret, pro něž byly dosavadní sběrnice pomalé. Nevýhodou lokálních sběrnic je o něco vyšší cena samotné základní desky s lokální sběrnicí a také zařízení pro ni určených.

Mezi základní parametery každé sběrnice patří:

Parametr	Význam	Jednotka
Šířka přenosu	Počet bitů, které lze zároveň po sběrnici přenést	bit
Frekvence	Maximální frekvence, se kterou může sběrnice pracovat	Hz
Rychlost (propustnost)	Počet bytes přenesených za jednotku času	B/s

Fyzické sběrnice – instalováno v počítačích – limitována velikost a spotřeba el. Výkonu, výkon je omezen šířkou sběrnice.
Logické sběrnice: - I/O rekonfigurovatelné sběrnice – využívají fyzické sběrnice, vhodné pro maximální využití sběrnice, CPU se chová, jako by měl variabilní sběrnice, vyžaduje komplexní řadič a plánovací sw

Fyzické sběrnice:

- paralelní – N paraleleních vodičů, rychlé, složité
- sériové – jedna přenosová cesta, obvykle kroucená dvoulinka (diferenciální vstupy), simplexní/duplxní komunikace, probélm kolize (duplex), chyby, výpadky, výhoda: jednoduchost, nevýhoda: pomalé

šířka pásma – propustnost: kolik dat lze přenést za jednotku času
četnost poruch a cena – četnost – ppst poruchy sběrnice, cena – kolik stojí restart

Technologie sběrnic:
- Měděné vodiče, cesty na PCB (2-8 GHz)
- koaxiální – Fiber optic – 100 Mbps až 2Gbps
- optický přenos volným prostorem – 20+ Gbps – omezení šířkou pásma vysílače/přijímače, problém s atmosférickými vlivy rozptyl a absorpce

6. Paměťový systém – hierarchie, mapování, ochrany paměti

Hierarchie:

1. **Registry** – rychlé interní paměti v CPU – rychlost = 1 cykl CPU, perzistence = několik cyklů, kapacita = až několik KB
2. **Cache** – rychlé paměť – interní nebo externí k procesoru – rychlost = několik cyklů hodin CPU, perzistence = desítky až stovky cyklů, pipeline, velikost až několik MB
3. **Hlavní paměť** – obvykle externí vzhledem k CPU – velikost i desítky GB, rychlost = 1-10 hodin CPU, perzistence = milisekundy až dny
4. **Disková paměť** – velmi pomalá – doba přístupu = 1-15 milisekund, pro odkládání dat z hlavní paměti

Fyzická / virtuální paměť:
Fyzická – instalována, limitována velikostí a spotřebou, potenciální rozšíření limitováno velikostí adresního prostoru
Virtuální paměť – stránky – přesouvané z/na disk – tabulka stránek spolupracuje s jednotkou Memory Management Unit při řízení přesunu stránek.

Přidělování bloků paměti pevné velikosti
OS MFT (Multitasking with Fixed nuber of Tasks). Paměť je při startu OS rozdělena na bloky. OS přidělí programu blok, jehož délka je větší nebo rovna nárokům tohoto programu. Je jednoduchý a v paměti může být několik procesů současně. Oproti tomu dochází k fragmentaci (špatné využití paměti), je potřeba znát nároky programu předem a u procesu, který má větší nároky než je velikost největšího bloku paměti nedojde k odstartování.
Adresování
Nelze předem stanovit, kde (na jaké adrese) bude program uložen -> program musí být relokabilní.
- použití relokační tabulky
- použití bázování a relativních skoků

Ochrana paměti
- **Metoda mezních registrů** – Užití dvou mezních registrů (nastavuje OS), které uvádějí nejnižší a nejvyšší dostupnou adresu.
- **Mechanismus zámků a klíčů** – Rozdělení paměti na stránky o pevné velikosti. Každé je přidělen zámek (celé číslo). Ve speciálním registru procesoru je klíč. Ty stránky, které mají stejnou hodnotu zámku jako je hodnota klíče, může proces používat.

Přidělování bloků paměti proměnné velikosti
Paměť není rozdělena na pevné bloky, ale programu je při jeho startu přidělena paměť podle nároků. Neboli se přidělí celý volný blok a to, co nepořebuje, program vrátí.
Metody výběru bloku
- First fit - vybere se první blok kde je délka větší nebo rovna chtěné velikosti
- Last fit - přidělí se poslední vyhovující
- Worst fit - přidělí největší volný blok
- Best fit - přidělí nejmenší, který vyhovuje

Ochrana paměti
- Metoda mezních registrů
- Mechanismus zámků a klíčů
    - paměť je rozdělena na bloky stejné velikosti (4kB)
    - každému programu přidělíme zámek (celé číslo)
    - bloky patřící danému programu označujeme stejným číslem (zámek)
    - v CPU je speciální registr s číslem běžícího procesu programu (klíč)
    - CPU povolí jen typ operace s pamětí kde zámek=klíč
    - při porušení ochrany -> vnitřní přerušení (zařídí OS)
    - speciální klíč 0 slouží OS (bez ochrany)
    - nastavení registru s klíčem je privilegovanou instrukcí
    - zámky jsou uloženy v RAM jako pole
    - metoda zámků a klíčů se používá dodnes

7. Vyrovnávací paměti procesoru
Cache paměť: obsahuje kopie úseků paměti -> dočasná paměť
3 mapovací schémata:
- přímo mapované: cache jedna k jedné – paměťová mapa (B může být obsažen jen v jediném bloku cache)
- částečně asociativní – jeden z n bloků cache obsahuje B
- plně asociativní – kterýkoliv blok cache může obsahovat B

Různá mapovací schémata pro různé způsoby přístupu na data.

Blok: skupina slov
Set: skupina bloků
Hit: je-li hledán blok B v cache a je nalezen
Hit-time – čas potřebný k nalezení bloku B
Hit Rate- poměrná doba, kdy B je nalezen v cache

Miss – je-li hledná blok a není v cache
Miss rate – poměrná doba, kdy B je hledán a není nalezen

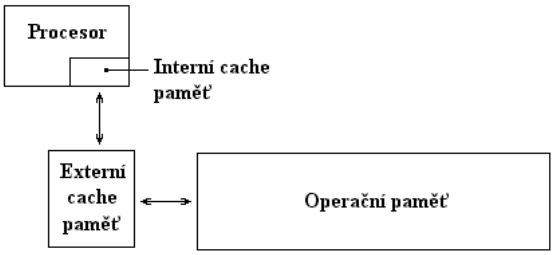
Příčiny: špatná strategie výměny bloků, špatná lokalita prováděných programů

Cache paměť je rychlá vyrovnávací paměť mezi rychlým zařízením (např. procesor) a pomalejším zařízením (např. operační paměť). V dnešních počítačích se běžně používají dva druhy cache pamětí:
- externí (sekundární, L2) cache:
Externí cache paměť je paměť, která je umístěna mezi pomalejší operační pamětí a rychlým procesorem. Tato paměť je vyrobena jako rychlá paměť SRAM a slouží jako vyrovnávací paměť u počítačů s výkonným procesorem, které by byly bez ní operační pamětí

velmi zpomalovány. První externí cache paměti se objevují u počítačů s procesorem 80386. Jejich kapacita je 32 kB popř. 64 kB. S výkonnějšími procesory se postupně zvyšuje i kapacita externích cache pamětí na 128 kB, 256 kB, 512 kB. Externí cache paměť je osazena na základní desce počítače (výjimku tvoří procesory Pentium Pro a Pentium II, které mají externí cache paměť integrovánu v pouzdře procesoru). Její činnost je řízena řadičem cache paměti.

- interní (primární, L1) cache:

Interní cache paměť je paměť, která slouží k vyrovnání rychlosti velmi výkonných procesorů a pomalejších pamětí. Tento typ cache paměti je integrován přímo na čipu procesoru a je také realizován pomocí paměti SRAM. Interní cache paměť se objevuje poprvé u procesoru 80486 s kapacitou 8 kB. Takovýto procesor musí mít v sobě integrován také řadič interní cache paměti pro řízení její činnosti.



Práce cache paměti vychází ze skutečnosti, že program má tendenci se při své práci určitou dobu zdržovat na určitém místě paměti, a to jak při zpracování instrukcí, tak při načítání (zapisování) dat z (do) paměti. Je-li požadována nějaká informace z paměti, je nejdříve hledána v cache paměti (interní, pokud existuje, a následně v externí). Pokud požadovaná informace není přítomna v žádné z cache pamětí, je zavedena přímo z operační paměti. Kromě momentálně požadované informace se však do cache paměti zavede celý blok paměti, takže je velká pravděpodobnost, že následně požadované informace již budou v cache paměti přítomny. Pokud dojde k zaplnění cache paměti a je potřeba zavést další blok, je nutné, aby některý z bloků cache paměť opustil.

Nejčastěji se k tomuto používá **LRU** (Least Recently Used) algoritmu, tj. algoritmu, který vyřadí nejdéle nepoužívaný blok.

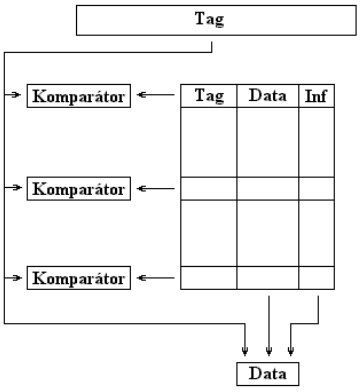
Cache paměti bývají organizovány jako tzv. **asociativní paměti**. Asociativní paměti jsou tvořeny tabulkou (tabulkami), která obsahuje vždy sloupec, v němž jsou umístěny tzv. **tagy** (klíče), podle kterých se v asociativní paměti vyhledává. Dále jsou v tabulce umístěna data, která paměť uchovává, a popř. další informace nutné k zajištění správné funkce paměti. Např.:

- informace o platnosti (neplatnosti) uložených dat
- informace pro realizaci LRU algoritmu
- informace protokolu **MESI** (Modified Exclusive Shared Invalid), který zajišťuje synchronizaci dat v cache pamětech v případě, že cache pamětí je v počítači více (u interních cache pamětí v okamžiku, kdy počítač obsahuje více procesorů).

Při přístupu do cache paměti je nutné zadat adresu, z níž data požadujeme. Tato adresa je buď celá, nebo její část považovaná za tag, který se porovnává s tagy v cache paměti.

Cache paměti jsou konstruovány jedním ze tří způsobů.

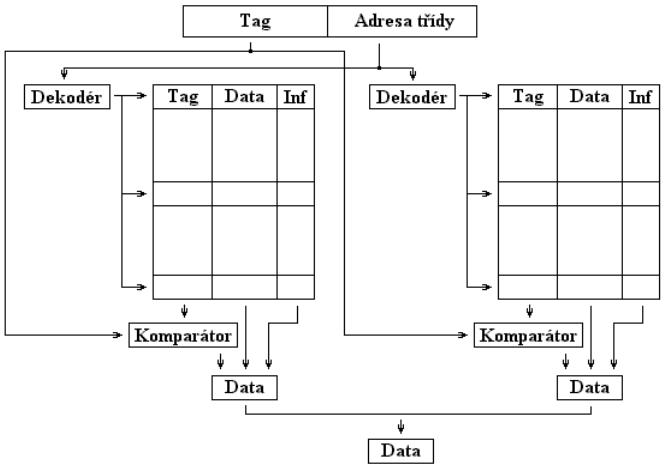
- **plně asociativní**:



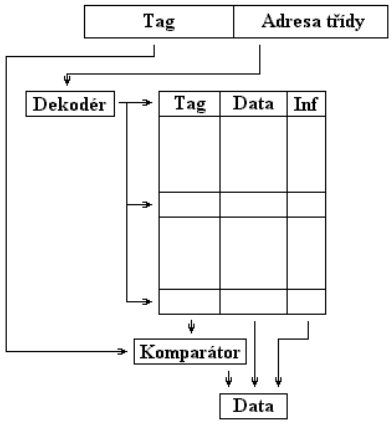
- U plně asociativní cache paměti je celá adresa, ze které se budou číst data (popř. na kterou se budou data zapisovat), brána jako tag. Tento tag je přiveden na vstup komparátorů (zařízení realizující porovnání dvou hodnot) společně s tagem v daném řádku tabulky. Pokud některý z tagů v tabulce je shodný se zadaným tagem na vstupu, ohlásí odpovídající komparátor shodu a znamená to, že požadovaná informace je v cahe paměti přítomna a je možné ji použít. Pokud všechny komparátory signalizují neshodu, je to známka toho, že požadovaná informace v cache paměti není a je nutné ji zavést odjinud (externí cache paměť, operační paměť).
- Tento způsob cache paměti má své nevýhody:
  1. Je nutné velké množství komparátorů
  2. Vzhledem k tomu, že se musí v každém řádku tabulky uchovávat celý tag, musí mít cache paměť velkou kapacitu, do které se tyto tagy ukládají a kterou není možné využít k uchovávání dat.

  Z těchto důvodů se plně asociativní paměti prakticky nepoužívají.

- **n-cestně asociativní**:



- N-cestně asociativní paměti pracují tak, že zadaná adresa se rozdělí na dvě části: tag, adresa třídy
  - Adresa třídy je přivedena na n dekodérů (zařízení, které na základě vstupní hodnoty vybere jeden ze svých výstupů, na který umístí hodnotu log. 1, a na ostatní výstupy umístí hodnotu log. 0), které v každé tabulce vyberou jeden řádek. Z těchto řádků se potom vezmou příslušné tagy a komparátorem se porovnají se zadaným tagem. Podobně jako u plně asociativních cache pamětí pokud jeden z komparátorů signalizuje shodu, je informace v cache paměti přítomna. V opačném případě je nezbytné informaci hledat jinde.
  N-cestně asociativní paměti částečně eliminují nevýhody plně asociativních cache pamětí a v současnosti jsou nejpoužívanějším typem cache pamětí.
- **přímo mapovaná**:



- Přímo mapovaná cache paměť je speciální případ n-cestně asociativní cache paměti pro n=1. Zadaná adresa je opět rozdělena na tag a adresu třídy. Adresa třídy je přivedena na vstup dekodéru, který podle ní vybere jeden řádek v tabulce. Tag na tomto řádku je následně porovnán se zadaným tagem, čímž se rozhodne o přítomnosti resp. nepřítomnosti informace v cache paměti.
- Přímo mapovaná cache ve srovnání s n-cestně asociativní cache pamětí vykazuje nižší výkon, a proto její použití není dnes příliš časté.

Adresa je rozdělena na tři části:

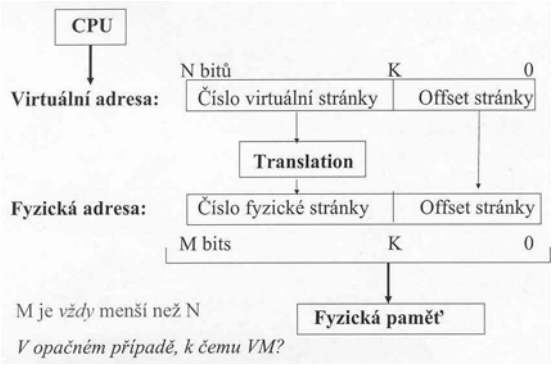
- **Tag** - horních 21 bitů
- **Adresa třídy** - 7 bitů => 128 řádků tabulky
- **Slabika** - dolní 4 bity

Adresa třídy je přivedena na dekodér, který vybere jeden řádek. Zadaný tag je dále komparátorem porovnán proti 4 tagům ve vybraném řádku. Pokud jeden z komparátorů ohlásí shodu, provede se výběr dat v datové části paměti. Datová část obsahuje v každém sloupci 16B, ze kterých je pomocí dolních 4 bitů zadané adresy vybrán jeden požadovaný byte.

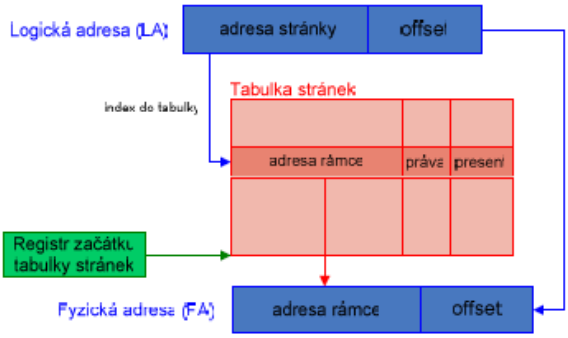
Každý řádek cache paměti ještě obsahuje jeden bit, který říká, zda informace v daném sloupci jsou platné, a 3 bity pro realizaci pseudo-LRU algoritmu. Pomocí tří bitů nelze vždy určit nejdéle nepoužívaný blok cache paměti. Tento algoritmus je však jednoduchý a rychlý a díky tomu poskytuje dostatečný výkon.
Podle způsobu práce při zapisování dat lze cache paměti ještě rozdělit do dvou skupin:
- write-through: cache paměti, u kterých v případě zápisu procesoru do cache paměti dochází okamžitě i k zápisu do operační paměti. Procesor tak obsluhuje jen zápis a o další osud dat se stará cache paměť.
- write-back: cache paměti, u nichž jsou data zapisována do operační paměti až ve chvíli, kdy je to třeba, a nikoliv okamžitě při jejich změně. K zápisu dat do operační paměti tedy dochází např. v okamžiku, kdy je cache zcela zaplněna a je třeba do ní umístit nová data. Tento způsob práce cache paměti vykazuje oproti předešlému způsobu vyšší výkon.

8. Organizace virtuální paměti

Stránka: blok ve virtuální paměti
Page fault = výpadek stránky – neúspěch při operaci MemRead
Virtuální adresa – určena CPU, odpovídá velkému adresnímu prostoru, je transformována pomocí HW a SW na fyzickou adresu
Memory Mapping nebo Address Trabnslation = proices transformace virtuální adresy na fyzickou
Transaltion Lookaside buffer – zvyšuje efektivitu procesu transformace adresy
Činnost virtuální paměti:



Cena virtuální paměti: načtení dat z disku – latence – milisekundy
Redukce ceny VM – velikost stránek, aby se amortizovala dlouhá doba přístupu na disk, plně asociativní mechanismus umisťování stránek, aby se snížila četnost výpadků
Obsluha výpadku stránek v sw – dost času v porovnání s cache
Překlad stránek:



Algoritmy náhrady stránek:
- **optimální** = vyhoď / přesun stránky, které budou použity nejpozději od tohoto momentu. pouze teoretické, lze uskutečnit pouze pro programy jejichž průběh je znám.
- **NRU** (not recently used) = nahrazuje se nejdéle nepoužívaná stránka. 2 bity M - modified (nastaven, když byl obsah stránky změněn. R - referenced (nastaven, když bylo ze stránky čteno, nebo do ní zapisováno). bity jsou nulovány čítačem. Priorita výměny = min (R,M) - 0 0 -> 1 0 -> 0 1 -> 1 1 -> max (R M)
- **FIFO** = nahrazuje se stránka která je v paměti nejdéle bez ohledu na používání - je nutné mít seznam stránek seřazený podle času vstupu do paměti, modifikace second chance kdy přidám rozhodování podle R bitu a pokud je 1 tak premístím stránku na konec fronty a resetnu R bit
- **clock** = modifikace second chance, jen fronta je nahrazena kruhovým bufferem a tedy nemusím přesouvat na konec, ale jen pohnu ukazatel (ručičku hodin) na další prvek.
- **LRU** (Last Recently used) = nahrazuje se stránka, na kterou nebylo odkazováno nejdelší dobu (nejstarší). Dobrá náhrada optimálního algoritmu, náročná realizace - každá stránka by měla mít něco na způsob čítače přístupů. vyhazují se stránky s nejmenším počtem přístupů, až se najde oběť musí se vynulovat - Pro n stránek potřebuji n x n matici inicializovanou n 0. Pokud je stránka k referencovaná je nejprve nastaven řádek k na 1 a pak sloupec k na 0.řádek kde je nejmenší binární hodnota je nejdéle nepoužíván.
- **NFU** (not frekventlu used) = přiblížení softwarové k LRU. software counter asociaovaný s každou stránkou, inicializován na 0, s každým klokem jsou projety všechny stránky, pokud je nastaven bit R tak je count ++, vyhazuje se ta s nejmenší count.
- **Random** = nahrazuje se náhodně vybraná stránka

Virtuální pamět stránkovaná a segmentovaná
Dvojúrovňový paměťový systém. Jeden typ vnitrní a jeden typ vnější. Problém adresace -> virtuální adresa. (vytvoření nezávislých paměťových prostorů, které jsou navzájem spojeny) Pro každou část samostatný adresový prostor. Překlad VA (virtuální adr.) na RA (reálnou adr.) se provádí vždy pomocí tabulky. Adresa rozdělena na 2 části. Jedna se překládá, druhá zůstává při překladu nezměněna.
Stránková paměť
paměť rozdělena na bloky konstantní délky (stránkt 1,2, 4, 8 KB).

- VSAT Virtualní adresa stránek
- P platnost stránky (jestli je v HP)
- RWX autorizace
- RAST reálná adresa stránek

Velmi neefektivní využití místa v tabulce stránek, protože počet stránek skutečně umístěných v HP bývá jen zlomkem celkového počtu stránek, pro ně je vytvořeno místo ve virtuálním adresovém prostoru. (adresa 32b a stránka 4KB-> 12b na Offset. = 20b pro VSAT, 2B na jednu položku = 2MB pro tabulku stránek. V HP je např. jen 1024 stránek = 99.9% neobsazených). Použijeme-li víceúrovňový překlad ztrácíme rychlost. Vložíme paměť (adresář, který obsahuje část tabulky stránek), která je rychlejší než HP. Pokud hledaná stránka není nalezena v adresáři následuje výpadek. Program, který ji požadoval je přerušen. Obslužný program uvolní jeden rám stránky v HP a na uvedené místo zapíše stránku, v níž adresa zádaná procesorem leží. Při čtení se provádějí kontroly: P příznaku RWX

Segmentová paměť

Segment je skupina po sobě následujících paměťových míst, která může měnit svoji velikost až po určitou maximální hodnotu. Délku lze nastavit podle potřeb programu. Zajímá v paměti vždy souvislé místo. Složitá práce se segmenty. VA se skládá ze 2 částí:

- VASE virtuální adresa segmentu
- Offset posuv v segmentu

Stránkovaná segmentová paměť

Využití výhod obou předcházejících. Segment je rozčleněn na stránky konstatní délky. VA má 3 části. Pro překlad 2 tabulky. Nejprve pomocí VASE zjistím začátek tabulky stránek a pak zjistím celou adresu. Výhoda spočívá v tom, že v HP mohou být přítomny pouze stránky aktivních procesů.

- VASE virtuální adresa segmentu
- VAST virtuální adresa stránek
- Offset posuv v segmentu

Inverzní tabulka stránek - Problémy s příliš velké rozměry tabulky stránek -> dlouhá doba potřebná pro jejich čtení. Vyhledávání pomocí hashing. To je celá podstata.

9. Systém pro vstup a výstup

Typy I/O:

- **Polled** – programová kontrola stavu I/O zařízení a následné plánování činnosti I/O zařízení , dotaz na stav (status) (busy, wait, idle, dead, ...)
- **Interupt-Driven –** obsluha na I/O požadavek, vyjádřený interuptem, vyžaduje frontu na uložení čakajících I/O žádostí
- **Direct Memory Access (DMA) –** přímý přenos mezi I/O zařízením a pamětí, velmi rychlé, používá se pro velká množství dat, blokové přenosy

Koncepce připojení:

Pomocí řadičů:

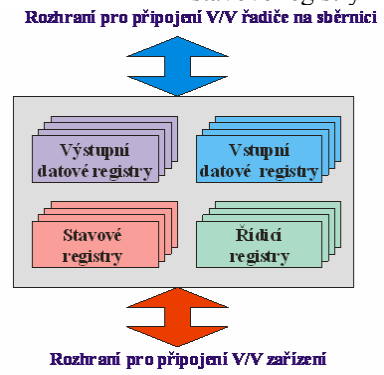
- specializované (řadič disku) – zařízení jen určitého tyu
- univerzální – lze k nim připojit různá zařízení různých typů

Rozhraní řadiče:

- rozhraní pro připojení na sběrnici – obvykle navrženo univerzálně pro snadné připojení na různé sběrnice nebo naopak jen pro určitou sběrnici, zahrnuje datové vodiče, několik adresních vodičů, řídící signály
- rozhraní pro připojení I/O zařízení – u specializovaných řadičů přizpůsobeno připojenému zařízení u univerzálních je buď programovatelné nebo podle standardu (USB, SCSI, ...)

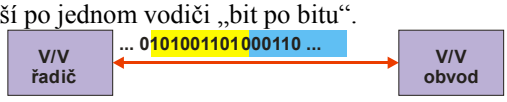
Řadič z pohledu programátora:

- datové registry – pro I/O dat
- řídící registry – nastavení parametrů řadiče, resp. Zařízení
- stavové registry – zjištění stavu řadiče (zařízení)

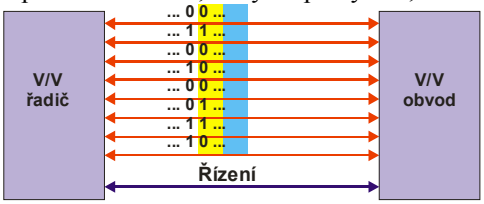


Základní typy přenosu dat z/do V/V zařízení

- Sériový přenos dat – data se přenáší po jednom vodiči „bit po bitu“.

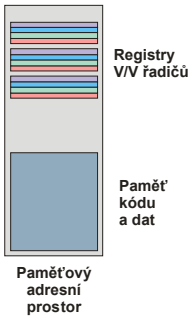


- Paralelní přenos dat – data se přenáší po více vodičích, obvykle po bytech, slovech (16 bitů) apod.

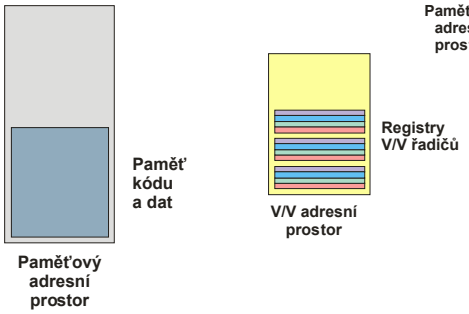


Jednotlivé registry V/V řadiče mohou být mapovány v paměťovém nebo ve V/V adresním prostoru.

- **V/V řadič v paměťovém adresním prostoru (například procesor H8S):**
  - pro přístup k registrům V/V řadiče se používají stejné instrukce jako pro práci s pamětí (MOV, ...).
  - lze používat i další instrukce (pokud jsou v instrukčním souboru),
  - často se používá neúplné dekódování adres.

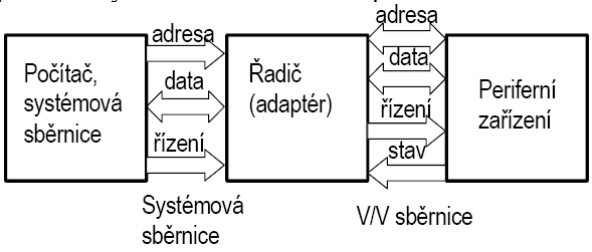


- **V/V řadič ve V/V adresním prostoru (např. IA-32):**
  - procesor rozlišuje adresní prostor paměti a adresní prostor V/V zařízení,
  - pro práci se zařízeními ve V/V adresním prostoru slouží zvláštní instrukce (IN, OUT),
  - na sběrnici se vyskytují kromě cyklů „*čtení z paměti*“ a „*zápis do paměti*“ také cykly „*čtení z V/V zařízení*“ a „*zápis do V/V zařízení*“,
  - adresní prostor V/V zařízení má obvykle menší velikost než adresní prostor paměti.



10. Technické prostředky pro provádění periferních operací a jejich řízení

Periferní operace – základní principy
• Na periferní operaci se podílejí:
počítač – systémová sběrnice – adaptér – V/V sběrnice – periferní zařízení (PZ).



Při úvahách o způsobu realizace periferní operaci je třeba řešit:
- způsob zahájení periferní operace,
- komunikace mezi zařízeními na systémové sběrnici podílejícími se na periferní operaci (procesor → řadič PZ a naopak, nebo komunikace mezi zařízeními přímo bez účasti procesoru – např. sběrnice UNIBUS),
- komunikace mezi zařízeními na V/V sběrnici podílejícími se na periferní operaci,
- jakým způsobem bude realizován přenos dat na systémové sběrnici,
- jakým způsobem bude realizován přenos dat na V/V sběrnici (mezi řadičem a PZ a naopak), jakým způsobem bude procesor informován o stavu PZ (velmi důležité, protože V/V operaci PZ nemá smysl začínat, pokud je PZ n připraveno nebo v poruše, stejně tak validita dat získaných v průběhu periferní operace, během níž došlo k poruše, je diskutabilní) => stav PZ se zjišťuje vždy před zahájením a po skončení PZ,
- jakým způsobem bude procesor informován o ukončení PO, pokud byla periferní operace realizována autonomně,
- způsob ukončení periferní operace (stavová informace, reakce na chybové a poruchové stavy).

Principy komunikace mezi procesorem a řadičem (nebo jiným zařízeníním v systémové sběrnici)
• Je založena na komunikaci s registry řadiče (čtení/zápis), tento způsob komunikace je realizován např. před zahájením periferní operace a po jejím skončení.
• Odehraje se na základě realizace *instrukce vstupu/výstupu*, příp. jiné skupiny instrukcí (viz vstupy/výstupy mapované do adresového prostoru operační paměti).
• Typy přenosů:
- mezi univerzálním registrem procesoru (dříve střadačem) a registrem řadiče (adresován na sběrnici je registr řadiče), instrukce vstupu/výstupu – pouze omezený repertoár
- není možné realizovat přenos mezi registrem řadiče a pamětí – adresy obou by musely být ve stejném okamžiku vystaveny na sběrnici,
- přenosy dat mezi řadičem (jeho datovým registrem) a pamětí se odehrávají vždy ve 2 fázích: registr řadiče → universální registr procesoru, universální registr procesoru → paměť. Stejný problém vzniká při přenosech DMA (Direct Memory Address) - přímý přístup do paměti – tam je způsob současného adresování dvou komponent, mezi nimiž je realizován přenos, řešen podobně. - Pozor: při DMA přenosech je procesor mimo hru – data se přenášejí z řadiče PZ přímo do operační paměti.

Dva způsoby komunikace s registry řadiče:
- isolované vstupy/výstupy (instrukce typu IN/OUT),

- vstupy/výstupy mapované do paměťového prostoru operační paměti (instrukce pro práci s pamětí).

Řadič PZ (adaptér)
Při návrhu řadiče je nutno se zabývat: - principy komunikace s procesorem přes systémovou sběrnici,
- principy komunikace s PZ přes V/V sběrnici, - z toho vycházející konstrukcí řadiče.
Z hlediska provádění periferních operací => v počítači musí být prvek, který řídí PZ, zjišťuje jeho stav, přebírá data a následně je přenáší do procesoru (a v opačném směru) – to jsou funkce řadiče.
• Různá úroveň řízení PO:
Průběh PO je řízen řadičem, pak signály mezi řadičem a PZ jsou typické pro dané PZ – např. rozhraní ST 506, v němž najdeme signál, jímž se řídí vystavení hlav na požadovanou stopu – jeden puls na tomto vodiči – vystavení o jednu stopu => ze struktury rozhraní jednoznačně poznáme, pro jaký typ PZ bylo toto rozhraní zkonstruováno.
Autonomní provádění PO – řadič nejprve zjistí stav PZ, pokud je PZ v pořádku, zahájí PO na tomto PZ (vložením parametrů PO do registrů PZ), tu pak PZ realizuje autonomně bez pozornosti procesoru – po jejím skončení jsou pak ve hře mechanismy, jimiž PZ uvědomí řadič a řadič následně procesor, že PO byla skončena.
To je okamžik, kdy řadič musí zjistit, jak PO dopadla => musí se zjistit stav PZ, pokud je hlášena nějaká chyba (včetně třeba i poruchy hardware PZ), pak bude PO považována za neplatnou a musí se zopakovat.

Konstrukce řadiče
Komunikuje s procesorem přes registry řadiče dostupné z procesoru pomocí instrukcí procesoru – do registrů je možné zapisovat, resp. jejich obsah číst.
• Informace uložená v registrech – parametry PO.
Pojem *vyrovnávací paměti*
Vyrovnávací paměť "vyrovnává" rozdíl v rychlosti zařízení komunikujících mezi sebou.
- V komunikaci řadič – PZ může být fyzicky umístěna buď v řadiči nebo v PZ.
- Tiskárna – vyrovnávací paměť s kapacitou řádku/stránky je součástí tiskárny.
- Disková paměť – vyrovnávací paměť je součástí řadiče (kapacita jeden sektor/více sektorů).

V/V sběrnice
Příklady: Centronics – rozhraní pro připojení tiskárny. ST506/IDE/EIDE – rozhraní pro připojení disků. SCSI – sběrnice pro připojení různých typů PZ (tiskárna, CD ROM, disky, skenery, …)

***Blokový (nárazový)/slabikový* režim.**
Blokový režim - přenášejí se souvislé bloky dat.
Slabikový režim - přenos po slabikách.

Periferní zařízení
Výstupní periferní operace pak sestává z těchto fází:
- přenos dat z operační paměti do vyrovnávací paměti řadiče,
- přenos dat z řadiče do PZ,
- realizace periferní operace (tisk řádku, zápis sektoru/více sektorů),
- hlášení z PZ do řadiče počítače o skončení operace (V/V sběrnice musí být pro tyto účely vybavena),
- analýza stavové informace řadičem PZ o výsledku PO,
- hlášení z řadiče do počítače o skončení operace (možnost využití přerušení)
- analýza stavové informace o výsledku PO procesorem.
• Při vstupní operaci bude průběh PO analogický.

PŘERUŠENÍ
• PO probíhají autonomně bez pozornosti procesoru – o ukončení PO je třeba informovat procesor –
vznikla potřeba zařadit do mechanismu obsluhy PO pojem „požadavek na přerušení“.
• Pro tyto účely musí být vybavena systémová sběrnice tak, aby každý řadič mohl takový požadavek generovat.
• Obdobně musí být vybavena i V/V sběrnice (mezi řadičem a PZ), pokud je na řadič připojeno více PZ – tam to bylo spíše posloupností signálů, která měla význam žádosti o přerušení.
• Po vzniku přerušení musí počítač zajistit přenos stavové slabiky z PZ a její analýzu, tzn. zjistit, jak proběhla periferní operace. Přerušení může sloužit také (kromě vstupu/výstupu dat) k synchronizaci programu a vnějších událostí a k okamžité reakci procesoru na důležitou stavovou změnu mimo procesor.