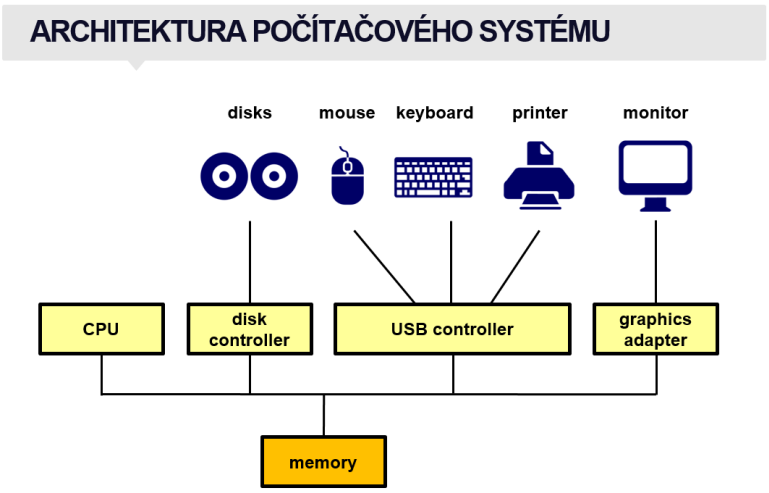
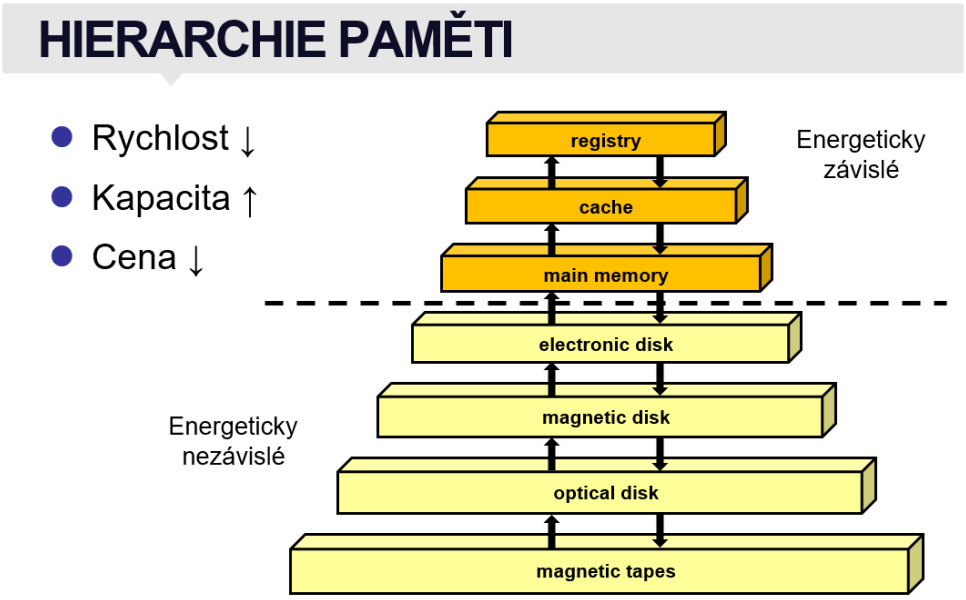
Práca s pamäťou

- Práca s pamäťou
- Pamäťová hierarchia
- Správa pamäte
- Logický a Fyzický adresový priestor
- Virtualizácia Pamäte
- Segmentácia
- Stránkovanie



Pamäťová Hierarchia



Primárna pamäť

- Tiež operačná, alebo hlavná pamäť
- Je jedinou väčšou pamäťou ktorú môže adresovať priamo CPU
- Zvykne byť energeticky závislá
- V súčasnosti sa kapacita pohybuje v stovkách MB až desiatkach GB
- Najrýchlejšia - V súčasnosti sa rýchlosť meria v nanosekundách
- Aby proces mohol bežať, je nutné aby bol v tejto pamäti
- Delí sa na rezidentný OS obvykle na počiatku FAP a užívateľské procesy

Sekundárna pamäť (on-line storage)

- Rozširuje pamäťovú kapacitu systému
- Je energeticky nezávislá
- Chce byť vysokokapacitná – od stoviek GB po niekoľko Terabajtov
- Relatívne pomalá doba prístupu – od mikrosekúnd po jednotky sekúnd
- Dnes najbežnejšie magnetický disk

Terciálna pamäť (off-line storage)

- Lacné a vymeniteľné médiá

- Cena je daná tým, že sa pracuje s veľa lacnými zväzkami v malom počte drahých mechanizmov
- Vymeniteľná knižnica sa najlepšie použije pre ukladanie riedko používaných dát
- Pomalá
- Energeticky nezávislá
- Floppy disky, magnetické pásky, optické disky
- Pevný disk je *asi* spoľahlivejší ako vymeniteľný a optický než magnetická páska
- Padnutie hlavy v pevnom disku obvykle znamená zničenie dát
- Porucha pásky alebo optického disku obvykle neznamená zničenie všetkých dát
- Aplikácie na páskach nemajú k dispozícii súborový systém

RAID

- Redundant Arrays of independet (Inexpensive)Disks
- Veľa diskov radených tak, aby poskytovali objem jedného veľkého
- Dáta sa uchovávajú redundantne pre zamedzenie straty
- Vďaka paralelnému spracovanou veľká rýchlosť
- Vyššia pravdepodobnosť zlyhania niektorého s diskov
- Zrkadlenie (Tieňovanie), Mirroring (Shadowing)
    - Každý logický disk "obsahuje" 2 fyzické, ktoré uchovávajú rovnaké dáta
    - číta sa len z jedného, druhý je zálohou
- Výkon sa zvyšuje tým, že sa súćasne zapisuje do viacerých diskov tak, že sa buď na bitovej úrovni dáta delia do jednotlivých diskov; dnes už nepoužívané, alebo tak, že sa súbory delia do blokov a tie sa zapisujú na rôzne disky.

ÚROVNĚ RAID, PŘEHLED

RAID Level 0:
Žádná redundance, jen souběžnost

RAID Level 1:
Spolehlivost dosažená zrcadlením disků

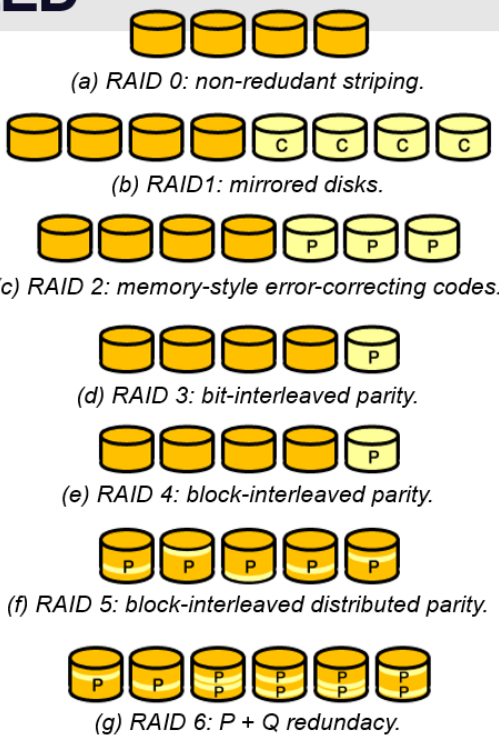
RAID Level 2:
Hamming code error correction

RAID Level 3:
1 kontrolní disk na skupinu, dělení bitů

RAID Level 4:
Nezávislé operace read/write, dělení bloků

RAID Level 5:
Data/parity přes všechny disky
(více souběžný přístup)

RAID Level 6:
Odolnost při více než jedné poruše disku



(a) RAID 0: non-redudant striping.

(b) RAID1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

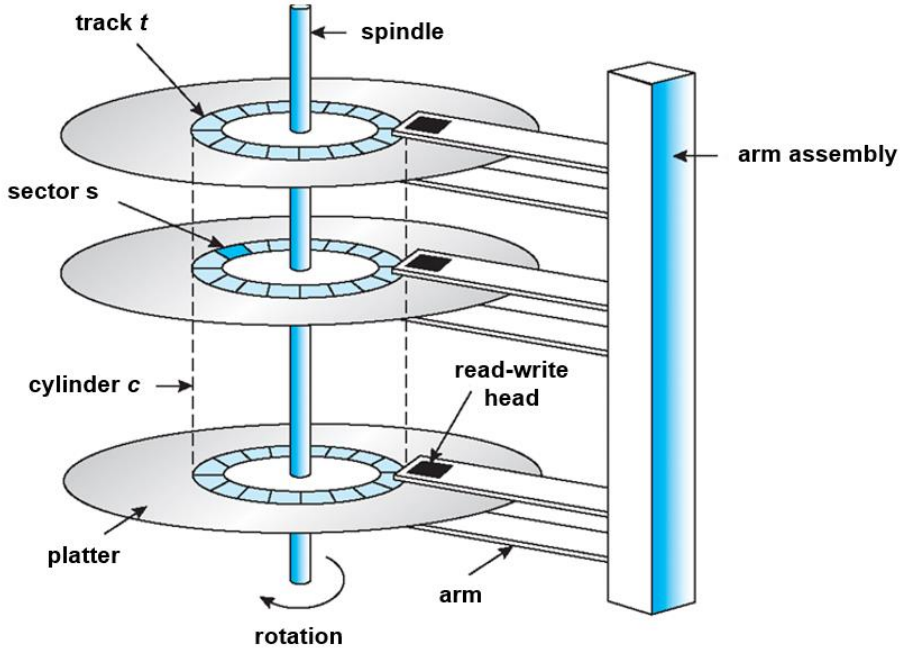
(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundacy.

Disky

MAGNETICKÉ DISKY



- Diskové mechanizmy sa adresujú ako veľké 1-dimenzionálne pole logických blokov ktoré sú najmenšou jednotkou prenosu dát a sú zobrazované do sektorov disku sekvenčne
    - 0 sektor je prvým sektorom na prvej stope vonkajšieho cylindru (cylinder - všetky stopy v rovnakej vzdialenosti od stredu)
    - zobrazovanie pokračuje od 0 po tejto stope. Potom po ostatných stopách tohto cylindru a potom po cylindroch smerom ku stredu
    - dáta sa zapisujú najskôr v jednom cylindri, aby sa zefektívnila práca disku

Plánovanie Disku

- OS je zodpovedný za čo najefektívnejšie využitie hardwaru čo pre disky znamená rýchly prístup a čo najväčšia šírka pásma
- Doba prístupu je daná :
    - dobou presunutia hlavy na cylinder so stopu s adresovaným sektorom
        - Optimalizujeme plánovaním činnosti disku tak, aby vzdialenosť čítaných sektorov bola čo najmenšia
    - dobou rotačného oneskorenia – dodatočný čas do priechodu adresovaného sektoru popod hlavu
        - Z hora obmedzená konštantou
            - Na disku < 35 ms
            - Na páske desiatky až stovky s – typicky 1000x pomalšie
- Šírka pásma je počet prenesených bytov / doba od zadania skupiny požiadavkou po ich ukončenie
    - Podporovaná - priemerná rýchlosť behom veľkého prenosu
    - efektívna – priemerná za celú dobu I/O operácie

Algoritmy pre plánovanie disku:

- First-Come First-Served (FCFS)

    - Sektory sa vyhľadávajú v poradí vakom o nich bolo žiadané
- Shortest seek time first (SSTF)
    - Z fronty požiadavkou vyberá ten, ktorý vyžaduje minimálnu dobu prístupu
    - môže spôsobiť starnutie požiadavkou
- SCAN
    - Hlavička chodí od kraju disku ku kraju a cestou plní požiadavky – „algoritmus typu výťah“
- C-SCAN
    - Funguje ako SCAN, ale hlavička číta dáta iba pri pohybe jedným smerom
    - jednotnejšia čakacia doba ako C-SCAN
    - Cylindre považuje za kruhový zoznam – za posledným cylindrom nasleduje zase prvý
- C-LOOK
    - C-SCAN s tým, že hlavička ide po kraj len kým existujú požiadavky tým smerom, potom sa vracia späť

Výber algoritmu

- SSTF je prirodzený no (C-)SCAN je vhodnejší pre ťažké zaťaženie disku
- Výkon algoritmu závisí na počte a typoch požiadavkou a tie zase metódami organizácie súborov v súborovom systéme
- Algoritmus by mal byť napísaný ako samostatný modul OS, aby bol o možné ho zamieňať
- Častou implicitnou voľbou je SSTF, alebo LOOK
- Pri modernom HW je možné, že disk si optimalizáciu rieši sám a my mu predávame len sadu požiadaviek. V takomto prípade ale OS stále môže mať záujem o uprednostnenie niektorých požiadavkou – I/O operácie z dôvodu výpadku stránky z operačnej pamäte, alebo zápis metadát súborového systému

Algoritmy v Linuxe

- NOOP
    - nemení poradie požiadavkou, len v prípade, že nový priamo nadväzuje na predošlí tieto sú zlúčené
- DEADLINE
    - Požiadavkám sa pridáva deadline (do kedy)
    - Algoritmus SCAN, ktorý po prechode hlavy ku kraju uprednostní (ak je treba) v ďalšom prechode požiadavky s exspirovaným deadlineom
    - uprednostňuje čítanie pred zápisom
- Anticipatory Scheduler (AS)
    - Rovnaký ako DEADLINE, ale očakáva, že po jednom požiadavku príde čoskoro rovnaký takže chvíľu počká (naozaj :D...)
    - Umožňuje aj krátke presuny späť
    - Neodlišuje zápis a čítanie
    - Odlišuje asynchrónne a synchrónne požiadavky
    - Neobjavuje sa v linuxovom jadre od verzie 2.6.33
- Completely Fair Queuing
    - snaží sa byť spravodlivý k procesom – každý proces dostáva časový diel (slice) kedy má exkluzívny prístup pre synchrónne požiadavky
    - Parametre:
        - slice_sync – dĺžka slice-u v ms
        - quantm – počet požiadavkou

- 17 front (pre každú prioritu 1) pre asynchrónne požiadavky – každá získava určitý slice

Správa pamäte

Správa Operačnej Pamäte

- Pri spúšťaní procesu, musíme proces nahrať do pamäte
- Program je zložený z častí, ktoré majú odlišné vlastnosti :
  - moduly s inštrukciami – execute-only
  - dátové moduly – read only alebo read/write
  - moduly môžu byť privat aj public
- Proces môže vyžadovať dodatočnú pamäť alebo túto OS vracať
- Procesy môžu časť pamäte zdieľať, ak si takto cielene predávajú informácii
- Po ukončení procesu musí OS všetku pamäť používanú procesom uvolniť
- OS teda musí:
  - Vedieť ktorá pamäť sa používa a kým
  - Alokovať a dealokovať pamäť podľa požiadavkou procesov
  - Rozhodovať ktorý proces kedy zavedie do pamäte
- Prekryvy, Overlays
  - v operačnej pamäti sa uchovávajú len inštrukcie, ktoré sú potrebné po célu dobu behu
  - táto technika je nutná v prípade, že proces dostal pridelený menší priestor ako potrebuje
  - Prekryvy implementuje programátor, od OS sa nepožaduje podpora
  - Návrh takejto štruktúry je značne zložitý

Správa Súborov

- Súbor je kolekcia súvisiacich informácii definovaná svojim tvorcom, môže ale nemusí mať štruktúru a z hľadiska OS je väčšinou len postupnosťou bajtov.
- Existuje mnoho typov I/O zariadení na ktorých sa súbory ukladajú :
  - Magnetické disky, Optické disky, mag. Pásky,...
- OS zavádza abstraktný koncept súboru
- OS typicky súbory a adresáre :
  - Vytvára a ruší
  - poskytuje pre nich základné operácie – čítanie, zápis, zoznam
  - Zálohuje

Správa I/O

- K I/O zariadeniam sa môže pristupovať cez súbory
- Je snaha o skrytí ich špecifikácie

Správa Sekundárnej Pamäte

- Typické sekundárne pamäte sú disky
- Spravuje sa obvykle formou súborov – súborového systému
- OS typicky:
  - spravuje voľné miesto
  - prideľuje miesto
  - plánuje činnosť disku (kedy čo)

Viazanie Adries

- Ide o napojenie odkazov na dáta na ich skutočné umiestnenie v pamäti

Pri kompilácii

- Umiestnenie v pamäti je známe predom
- Ide generovať absolútny kód
- Pri zmene umiestnenia, je nutné program znovu preložiť

Pri zavádzaní

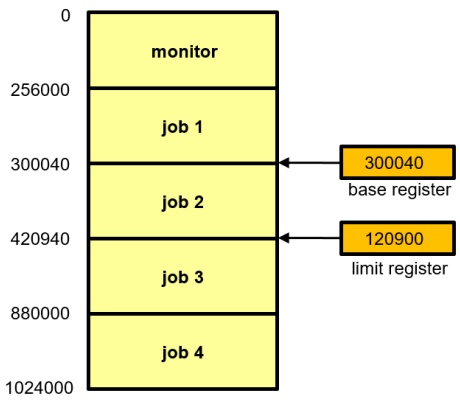
- Umiestnenie v pamäti nie je známe v dobe kompilácie – generuje sa relocatable code

Za behu (Dynamické)

- V tomto prípade sa viazanie adries prebieha v dobe behu programu
- Je nutná hardwarová podpora
- Pre umiestnenie knižníc rezidentných v operačnej pamäti sa používa kód malého rozsahu – stud, ktorý sám seba nahradí pri volaní adresou skutočnej funkcie a predá jej riadenie
- OS kontroluje, čie je funkcia mapovaná do pamäte procesu

Ochrana pamäte

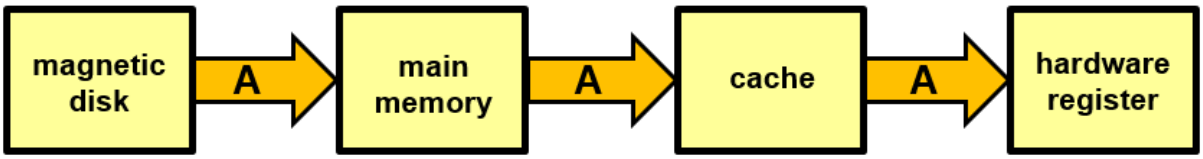
- V prípade, že sa v operačnej pamäti nachádza niekoľko procesov, je nutné zabrániť im, aby si navzájom prepisovali dáta a pod.
- OS a CPU sa tomuto snažia zabrániť a to pomocou:
    - Režimov procesoru
        - **Užívateľský**
        - **Privilegovaný** – má právo k viacerým inštrukciám
        - Z privilegovaného režimu sa CPU dostane sériou inštrukcií z Užívateľského do privilegovaného pri spracovaní prerušenia
    - Vyhradením pamäte jednotlivým procesom
        - CPU na základe registrov alebo princípov nastavených OS kontroluje prístupy procesu.
        - Príklad: báza a limit určuje rozsah adries ktoré má proces právo čítať a zapisovať (báza + 0 až báza + limit)
        - Prístup k nepovolenej adrese spôsobí prerušenie ktoré spracováva OS
        - Jednoduchá implementácia, nastavenie obmedzení definujeme ako privilegovanú operáciu



    - Časovač zaisťuje, že vládu nad CPU má OS

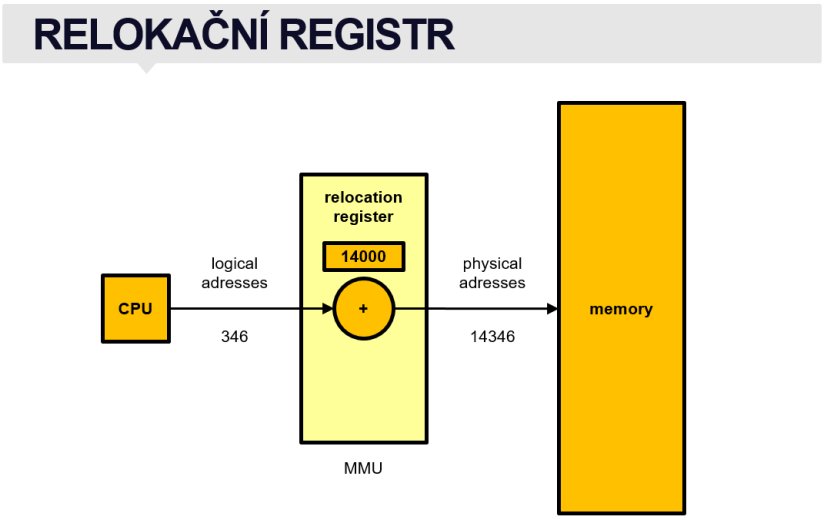
Kešovanie, cache, medzipamäť

- Ide o použitie rýchlejšej pamäte pre uchovávanie naposledy použitých dát z pamäte pomalšej
- Ak sa dáta v cache nachádzajú, použijú sa, ak nie, je nutné och natiahnuť z pomalšej pamäte a zároveň sa prenesú aj okolité dáta :
    - princíp časovej lokálnosti (Temporal Locality)
        - Môžeme predpokladať, že práve použité dáta budú čoskoro použité znova
    - princíp priestorovej lokálnosti (Spatial Locality)
        - Môžeme predpokladať, že dáta v okolí práve použitých dát budú čoskoro použité
- Veľkosť cache je obmedzená
- Dáta sa zároveň udržujú v niekoľkých úrovniach pamäte, je nutné udržiavať konzistenciu!



Logický (LAP) a fyzický (FAP) adresový priestor

- o dresový priestor sa v prípade OS s mikrojadrom stará práve mikrojadro
- hardwarový modul prevádza logické adresy na fyzické
- užívateľský program pracuje len s logickými adresami, na fyzické nevidí
- pripočítava sa obsah „relokačného registru“ (**MMU**) k adresám generovaným užívateľským procesom práve v okamihu keď je predávaná ako ukazateľ do operačnej pamäte
- FAP a LAP sa zhodujú v dobe kompilácie a v dobe zavedenia
- FAP a LAP môžu byť rozdielne pri viazaní za behu



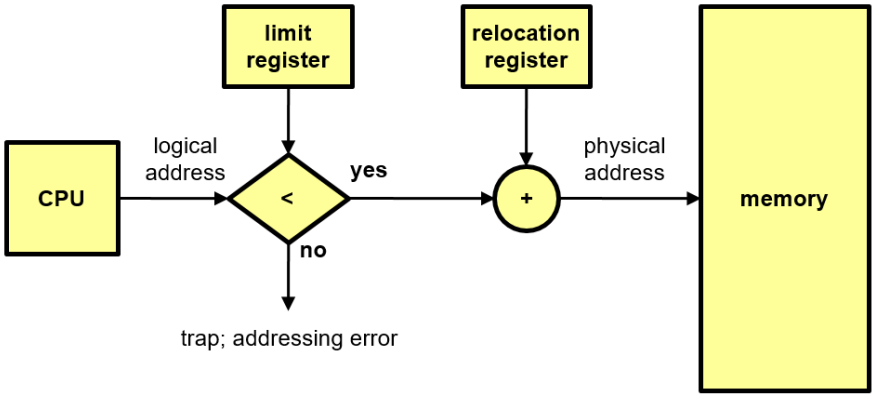
Logický adresový priestor

- logická adresa je daná adresou v strojovom jazyku a generuje ju CPU
- (opisne) ide o akísi pseudonym pre kus fyzických dát, ktorých úložisko sa môže meniť
- logická adresa sa delí na :
    - číslo stránky (p) – index do tabuľky stránok a bázovej adresy rámca
    - Offset v stránke (d)

Fyzický adresový priestor

- fyzická adresa v pamäti s ktorou samotná pamäť pracuje

HW PODPORA



Súvislé oblasti

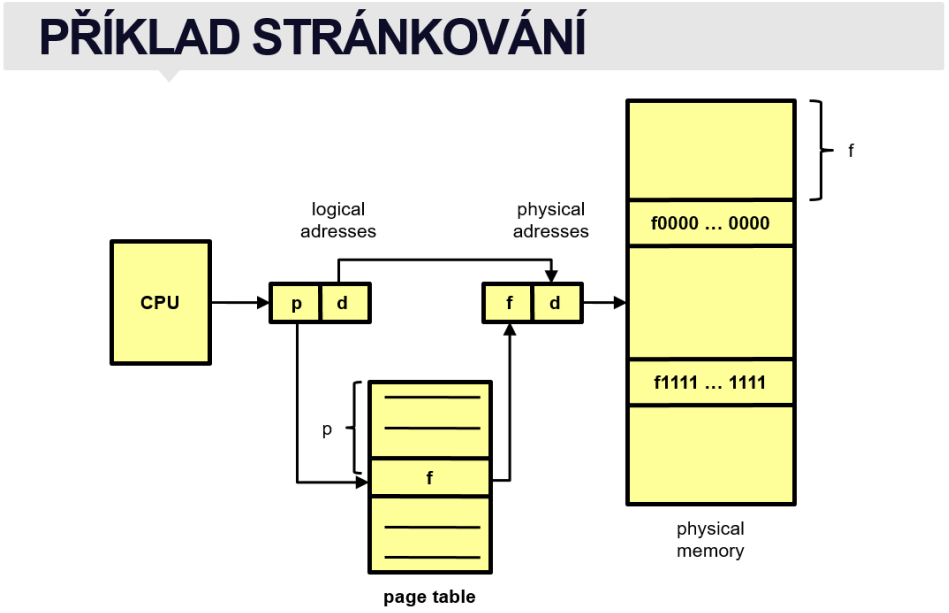
- pre ochranu procesou medzi sebou sa používa relokačný – hodnota najmenšej fyzickej adresy procesu, a mezdný register – ktorý udáva aj rozopätie logických adries procesu
- Pri prideľovaní niekoľkých častí pamäte a ukončovaní procesov môžu vznikať diery blok voľnej pamäte, ktoré sú roztrúsené po FAP – tzv. súvislé oblasti
    - evidenciu rieši OS
- Pre určenie ktorú súvislú oblasť procesu prideliť sa používajú napr. nasledujúce algoritmy:
    - First-fit – prvá dosť dlhá voľná oblasť
    - Best-fit – prideľuje sa najmenšia dosť dlhá oblasť (jej počiatok)
    - Worst-fit - prideľuje sa najväčšia dosť dlhá oblasť, aby sa generovali čo najväčšie diery
- Z hľadiska rýchlosti a efektivity sú lepšie First-fit a Best-fit
- Z tohto vzniká problém fragmentácie :
    - Vonkajšia – voľnej pamäte je dosť, ale nie v súvislej oblasti
        - Rieši sa presúvaním, s cieľom vytvoriť jednu veľkú voľnú oblasť
            - Len v prípade, že je možná dynamická realokácia
            - V dobre behu
            - Nejde hýbať vyrovnávacími pamäťami ktoré sa plnia z periférií

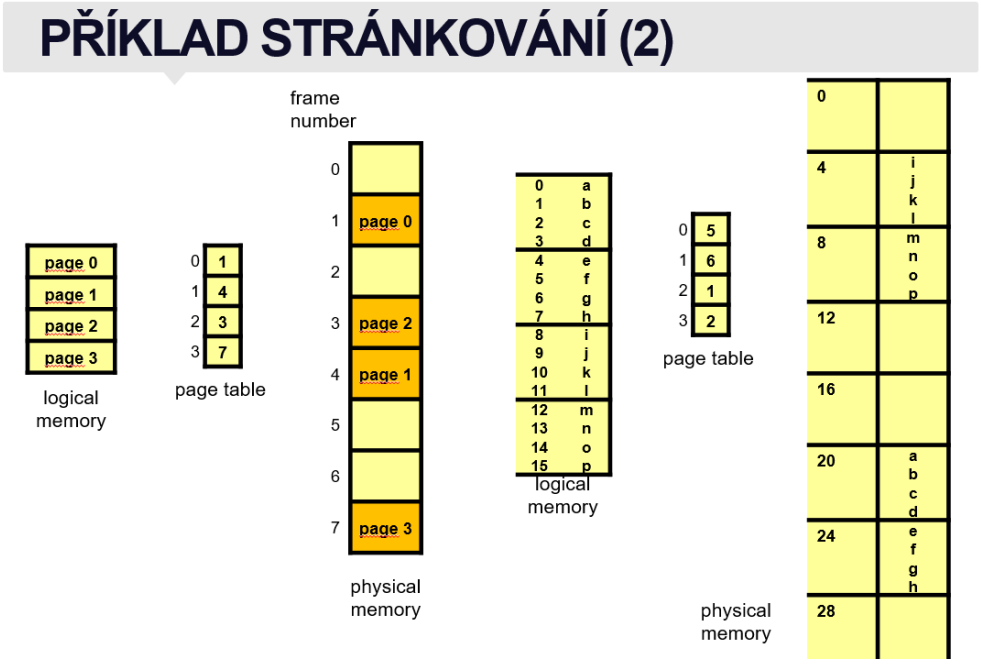
- Vnútorná – pridelená oblasť je väčšia ako požadovaná veľkosť

Stránkovanie

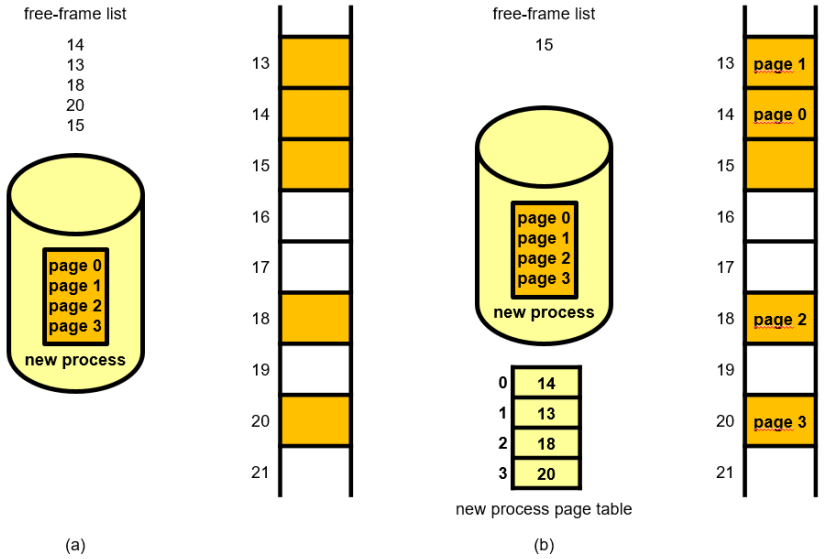
LAP procesu nemusí byť súvislým blokom v FAP

- FAP sa delí na sekcie zvané rámce (frames)
  - Pevná dĺžka v násobkoch mocniny 2 (zväčša 512 až 8192 bajtov)
- LAP sa delí na sekcie zvané stránky (pages)
  - dĺžka zhodná s dĺžkou rámcu
- Udržuje sa zoznam voľných rámcov
- Program dĺžky n rámcov sa umiestni do n rámcov
- Logická adresa sa prevádza na fyzickú pomocou prekladovej tabuľky nastavené OS a interpretovanej MMU
- Vzniká vnútorná fragmentácia, lebo pamäť je prideľovaná v násobkoch veľkosti rámca





PŘÍKLAD STRÁNKOVÁNÍ (3)



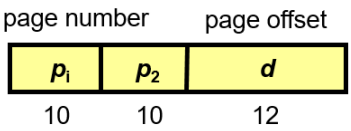
Tabuľka stránok

- Uložená v operačnej pamäti
- Počiatok aj koniec je odkazovaný registrom (PTBR a PTLR)
- Pre získanie inštrukcie/údaju je nutné 2x vstúpiť do operačnej pamäte, toto je možné riešiť špeciálne rýchlou cache pamäťou

Dvojúrovňová tabuľka stránok

- 32-bitový procesor s 4KB stránkou

- Logická adresa
  - číslo stránky: 20 bitů
  - adresa ve stránce: 12 bitů
- Číslo stránky se dále dělí
  - číslo stránky 10-bitů
  - adresa v tabulce stránek 10-bitů

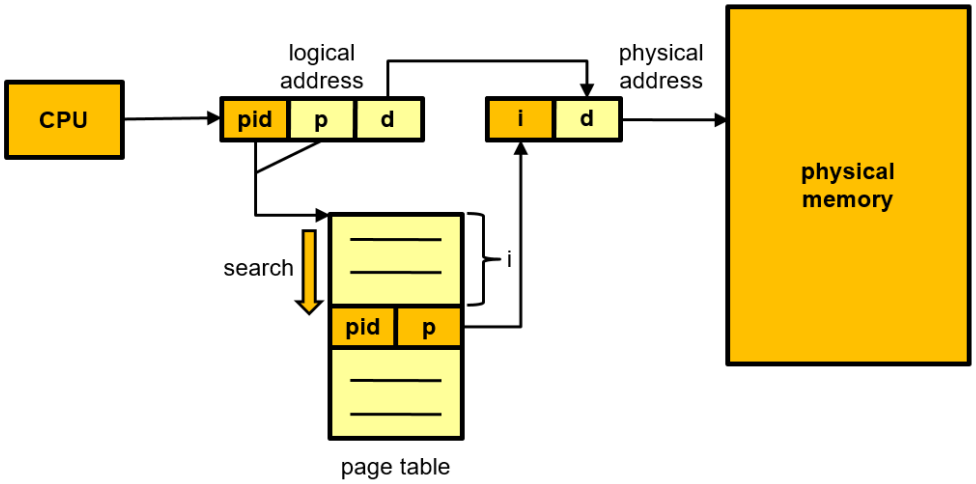


Inverotovaná tabuľka stránok

- Tabuľka obsahuje záznam o všetkých rámcoch
- Obsahuje logickú (virtuálnu) adresu stránky a informáciu o procese ktorý ju vlastní
- Znižuje sa veľkosť pamäte potrebnej pre uchovanie tabuľky
- Zvyšuje sa doba prístupu do tabuľky, je nutné prehľadávať – vylepšenie hašovaním

INVERTOVANÁ TABULKA: PŘÍKLAD

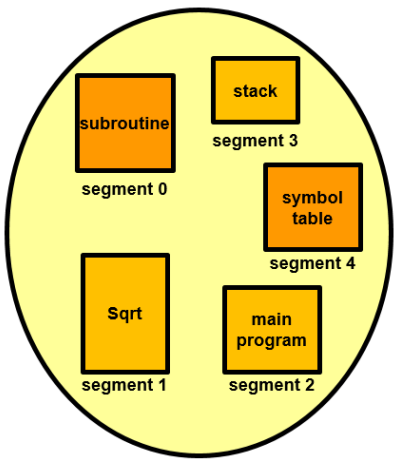
- Např. AS400 (IBM), UltraSPARC, PowerPC



Segmentácia

- Pamäť sa nedelí na stránky a rámce, ale na segmenty
- Logickou adresou je číslo segmentu a offset
- V tabuľke sa uchováva začiatok segmentu v FAP – base, a jeho dĺžka – limit
- STBR – odkaz na tabuľku v pameti
- STLR – dĺžka tabuľky
- Počet segmentov je legálny ak je < STLR
- Relokácia je dynamická pomocov ST

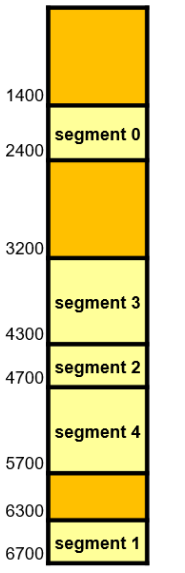
PŘÍKLAD SEGMENTACE



logical adresses space

	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table



physical memory

Virtualizácia pamäte

- Separácia LAP a FAP (doteraz LAP ukazoval na FAP)
- Vo FAP sa môžu nachádzať len časti programov, ktoré sú nutné pre bezprostredné riadenie procesov
- LAP môže byť väčší než FAP
- Adresové priestory je možné zdieľať medzi jednotlivými procesmi
- Je možné efektívnejšie vytvárať procesy

Techniky implementácie

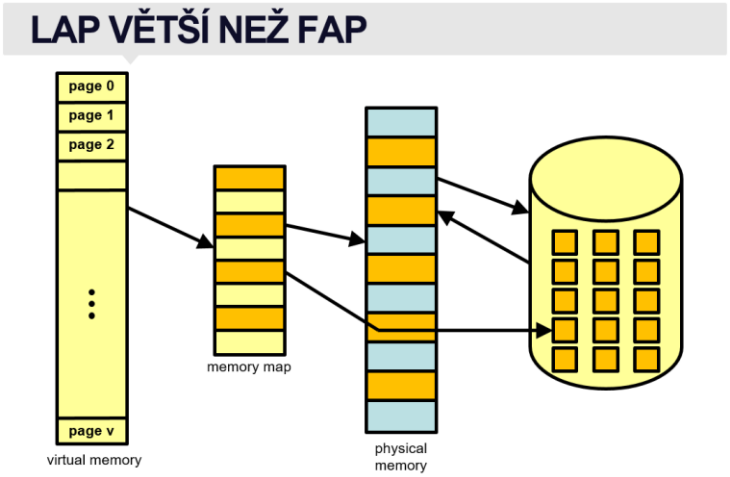
- Stránkovanie na žiadosť – Demand Paging
- Segmentovanie na žiadosť – Demand Segmentation

Beh procesov vo virtuálnej pamäti

- Časť programu uloženej vo FAP nazývame Rezidenčná množina
- Odkaz mimo rezidenčnú množinu spôsobí prerušenie a proces je označený ako čakajúci
- OS spustí I/O operácie ktoré zabezpečia nutnú správu pamäte pre zavedenie odkazovanej časti do FAP, zatiaľ beží iný proces. Po skončení zavedenia je generované I/O prerušenie a proces je označený ako pripravený
- Preklad LAP adresy na FAP adresu prebieha pomocou indexovania tabuľky PT/ST pomocou CPU

Vlastnosti virtuálnej pamäte

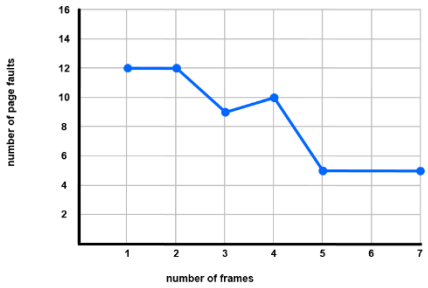
- Vo FAP je možné udržovať viacero procesov – čím viac tým väčšia šanca, že nejaký bude pripravený
- Je možné realizovať procesy, ktoré požadujú viac pamäte ako je kapacita FAP
    - nie je nutné, aby tento problém riešil programátor alebo kompilátor
- Obraz LAP sa ukladá v externej pamäti
    - Nepoužíva sa štandardný systém súborov OS, ale špeciálne metódy optimalizované pre tento účel – špeciálna partícia disku
- Stránkovanie/Segmentáciu musí podporovať hardware správy pamäte
    - stránkovanie na žiadosť
    - segmentácia na žiadosť
    - žiadosť – dynamicky, kontextovo generovaná indikácia nedostatku
- OS musí byť schopný organizovať tok stránok/segmentov medzi vnútornou a vonkajšou pamäťou



Zavádzanie stránky

- Kedy?
  - na žiadosť (Demand paging)
    - Stránka sa zavádza do FAP pri odkazu na ňu, ak sa tam už nenachádza
  - Predstránkovanie (Prepaging)
    - Princíp lokálnosti
    - Zavádza sa viac stránok než sa požaduje
    - Vhodné pri inicializácii procesu
- Kam?
  - pri segmentácii – Bestfit , Worstfit, Firstfit
  - pri stránkovaní a kombinácii oboch nie je nutné riešiť
- Pri zaplnení ktorú stránku nahradiť:
  - Niektore stránky nie je možné prepísať – napr. I/o buffery, riadiacu štruktúru OS
  - 2 prístupy:
    - Intra (local) množina „obetí“ – stránky procesu, ktorý vyvolal výpadok
    - Extra (global) množina „obetí“ – aj stránky ostatných procesov (napr. podľa ich priority)
  - od algoritmu sa požaduje čo najmenej výpadkov stránok – optimálny algoritmus je nereálny, lebo nepoznáme budúcnosť, používa sa pre porovnanie
  - Čím viac rámcov v FAP máme, tým menšia je pravdepodobnosť výpadk
  - Príklady algoritmov pre výber obete :
    - LRU (Least Recently Used)
      - Najdlhšie neodkázaná stránka
      - Výkon blízko optimálneho algoritmu
      - Implementuje sa:
        - Udržovaním času posledného prístupu v PT
        - Zásobníkom ktorý udržuje poradie prístupu – na vrchu naposledy pristupovaná stránka
        - aproximácia – bit, ktorý len hovorí, že k stránke bolo pristupované, nevieme poradie prístupu k stránkam iba ktoré boli použité
      - Vysoká réžia implementácie
    - FIFO
      - Najdlhšie zobrazená stránka vo FAP
      - Beladyho anomália

BELADYHO ANOMÁLIE



- Vo veľa prípadoch sú často odkazované stránky práve tie najstaršie čo nie je brané do úvahy týmto algoritmom

        - Jednoduchá implementácia
    - Poslednej šance
        - FIFO s vynechaním stránok na ktoré sa od posledného výberu odkazovalo
        - Pri výbere obete cyklicky prechádzame – podobne ako FIFO
        - Odkazom na stránku sa jej nastaví príznak
        - Obeť, ktorá nemá príznak a je na rade pri cyklickom prechode je nahradená novou stránkou, po každom nahradení sa príznak „use bit“ nastavuje na 0 všetkým stránkam
        - Experimenty ukazujú optimálnosť blížiacu sa LRU
        - Pri pridaní modified bitu šetrí výpis nemodifikovanej stránky
    - ak je uvoľňujúca stránka pozmenená jej kópia na disku sa rovnako upraví (rozpoznáva sa pomocou bitu modify (dirty) ktorý nastavuje HW automaticky)
- Funny fact zo zdrojákov:

STRUKTURA PROGRAMU

- `int data[128][128];`
- Jeden řádek odpovídá jedné stránce
- Program 1

```
for (j = 0; j <128; j++)
    for (i = 0; i < 128; i++)
        data[i,j] = 0;
```

    - 128 x 128 = 16,384 výpadků stránky
- Program 2

```
for (i = 0; i < 128; i++)
    for (j = 0; j < 128; j++)
        data[i,j] = 0;
```

    - 128 výpadků stránky

Stránkovanie na žiadosť

- Prínosy:
    - menej I/O operácií
    - menšie požiadavky na pamäť
    - rýchlejšie reakcie
    - možnosť práce viacerých užívateľov
- Zavádzanie do FAP sprostredkováva OS

Bit Valid-Invalid

- Pri virtualizácii sa do PT (Page Table) prikladá valid-invalid bit
- Na začiatku sú všetky bity nastavené na 0
- V/I bit hovorí o tom, či sa stránka s danou LAP adresou nachádza v FAP pamäti, alebo nie

- V prípade, že sa volá LAP adresa s V/I bitom nastaveným na 0 generuje sa prepušenie typu page fault – k je to legálna referencia, OS stránku zavedie, nastaví bit na 1 a opakuje inštrukciu

PŘÍKLAD: TABULKA STRÁNEK