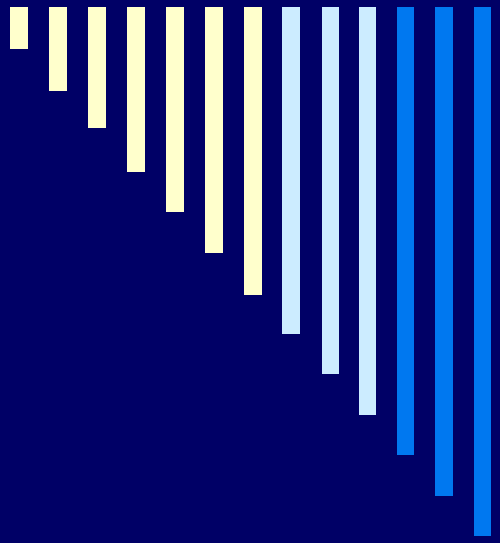
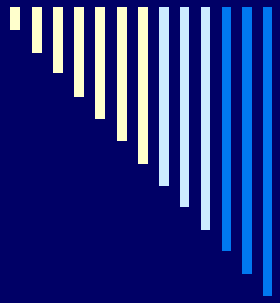


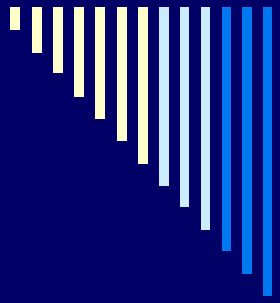
09. Memory management

ZOS 2006, L.Pešička



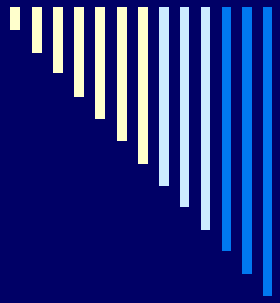
Správa paměti

- „paměťová pyramida“
- absolutní adresa
- relativní adresa
  - *počet bytů od absolutní adresy*
- fyzický prostor adres
  - fyzicky k dispozici výpočetnímu systému
- logický adresní prostor
  - využívají procesy



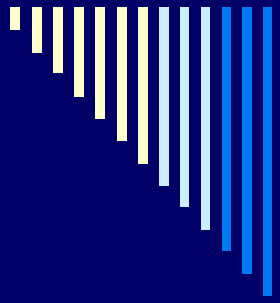
Modul pro správu paměti

- informace o přidělení paměti
  - která část je volná
  - přidělená (a kterému procesu)
- přidělování paměti na žádost
- uvolnění paměti, zařazení k volné paměti
- odebírá paměť procesům
- ochrana paměti
  - přístup k paměti jiného procesu
  - přístup k paměti OS

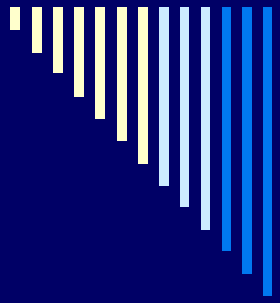


Memory management

- Základní mechanismy
  - Bez odkládání a stránkování
  - Jednoprogramové systémy
  - Multiprogramování s pevným přidělením paměti
  - Multiprogramování s proměnnou velikostí oblasti
  - Správa paměti
    - Bitové mapy
    - Seznamy
    - First fit, best fit, next fit
    - Buddy system



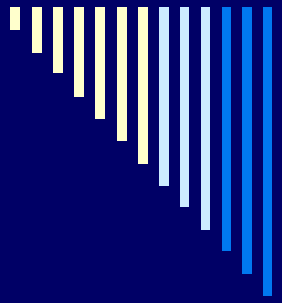
Statická a dynamická relokace



Relokace a ochrana

- Problémy při multiprogramování (víc programů současně v paměti):

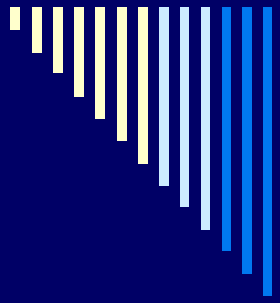
- Relokace
  - Programy běží na různých adresách
- Ochrana
  - Paměť musí být chráněna před zasahováním jiných programů



Relokace při zavedení do paměti

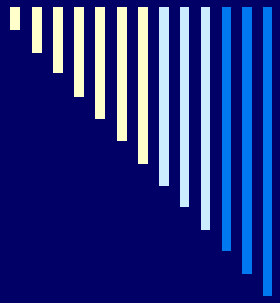
- Nejprve – proces, jak je program vytvořen a spuštěn

- Překlad a sestavení programu
  - Aplikace ve vysokoúrovňovém jazyce
  - Větší SW – rozděleny do modulů – musejí být přeloženy a sestaveny do spustitelného programu
  - Objektové moduly
    - Výsledkem překladu
    - Příkazy ve zdrojovém textu – přeloženy do stroj. instrukcí
    - Zůstávají symbolické odkazy – adresy prom., procedur,fcí



Relokace při zavedení do paměti 2

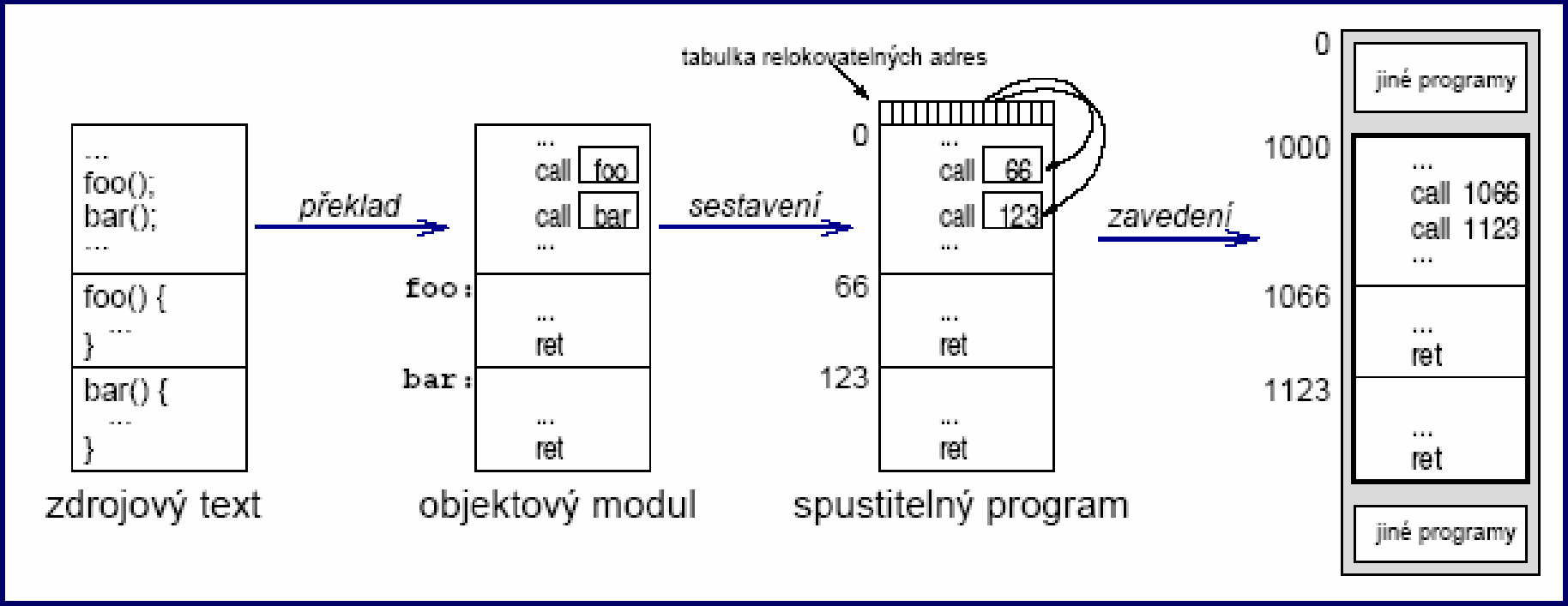
- Výsledný spustitelný program
  - Sestavení (linking) modulů a knihoven
- Při sestavení se řeší hlavně externí reference
  - Všechna místa výskytů referencí – seznam
  - Když je adresa známa – vloží se všude, kde se používá
  - Symbolické odkazy se převedou na číselné hodnoty
  - Výsledek – spustitelný program

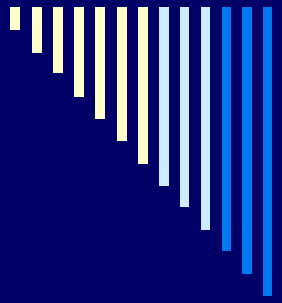


Relokace při zavedení do paměti 3

- Komplikace při více programech v paměti
  - Příklad
  - 1. instrukcí programu volání podprogramu *call 66*
  - Program v paměti od adresy *1000*, ve skutečnosti provede call *1066*
- Jedno z řešení – modifikovat instrukce programu při zavedení do paměti
  - Linker – do spustitelného programu přidá seznam nebo bitmapu označující místa v kodu obsahující adresu
  - Při zavádění programu do paměti se každé adrese přičte adresa začátku oblasti

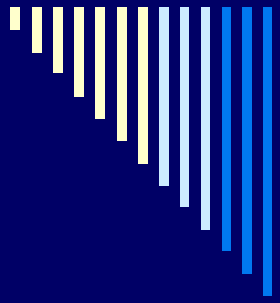
Relokace při zavedení do paměti





Statická relokace

- Výše uvedený způsob
- “Adresy se natvrdo přepíší správnými”
- Např. OS/MFT od IBM

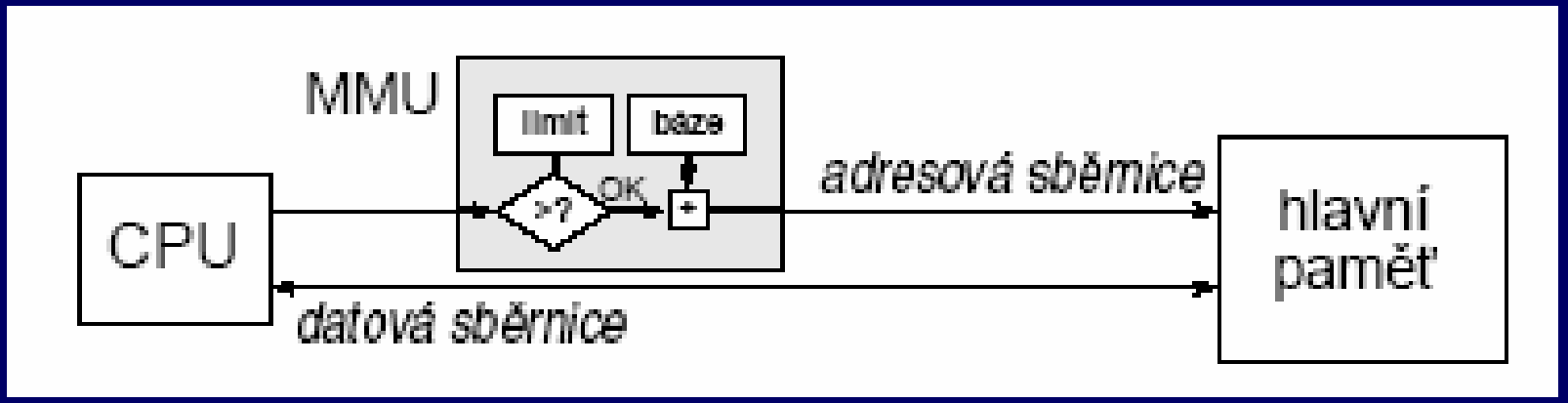


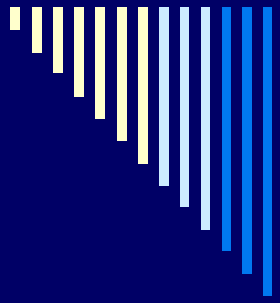
Ochrana paměti při statické relokaci

- Proces mohl zasahovat do paměti jiných procesů
- IBM 360 – přístupový klíč
  - Pamět rozdělena do bloků 2KB
  - Každý blok – sdružený hw 4 bitový kód ochrany
  - PSW procesoru obsahuje 4 bitový klíč
  - Při pokusu o přístup k paměti jejíž kód ochrany se liší od klíče PSW – výjimka
  - Kód ochrany a klíč může měnit jen OS (privilegované instrukce)
  - Výsledek – ochrana paměti

Mechanismus báze a limitu

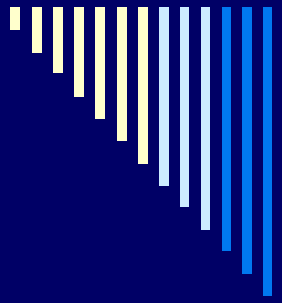
- Jednotka správy paměti MMU mezi CPU a paměť
- Dva registry – báze a limit
- Báze – počáteční adresa oblasti
- Limit – velikost oblasti





Mechanismus báze a limitu

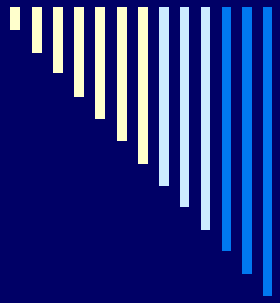
- Funkce MMU
  - Dostává adresu od CPU, převádí na adresu do fyzické paměti
  - Nejprve zkontroluje, zda adresa není větší než limit
    - Ano – výjimka, Ne – k adrese přičte bázi
- Pokud báze 1000, limit 60
  - Adresa 55 – ok, výsledek 1055
  - Adresa 66 – není ok, výjimka



Dynamická relokace

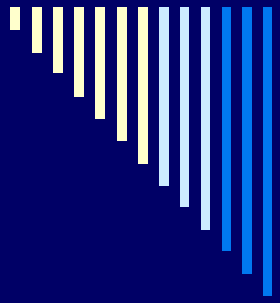
- Výše uvedený a podobné mechanismy
- Provádí se dynamicky za běhu
- Nastavení báze a limitu může měnit pouze OS (privilegované instrukce)

- Např. 8086 – slabší varianta (nemá limit)
- Bázové registry = segmentové registry DS,SS,CS,ES



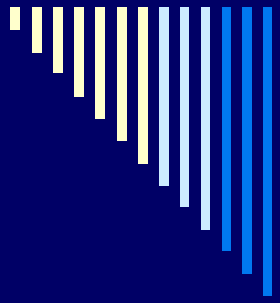
Správa paměti s odkládáním celých procesů

(Proces se vejde do fyzické paměti)



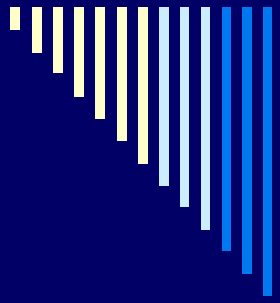
Správa paměti s odkládáním celých procesů

- Pro dávkové systémy – dosud uvedené mechanismy - přiměřené (jednoduchost, efektivita)
- Systémy se sdílením času – víc procesů, než se jich vejde do paměti současně
- 2 strategie
  - Odkládání celých procesů (swapping)
    - Nadbytečný proces se odloží na disk
    - Např. UNIX Version 7; co platí pro velikost procesu?
  - Virtuální paměť – v paměti nemusí být procesy celé
    - Překrývání (overlays), virtuální paměť



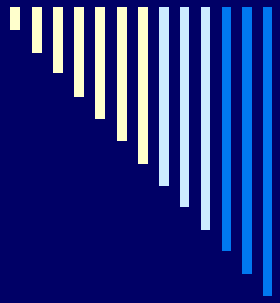
Odkládání celých procesů

- co víme o velikosti procesu?
- data procesu mohou růst
- pro proces alokováno o něco více paměti, než je třeba
- potřeba více paměti, než je alokováno
  - přesunout proces do větší oblasti (díry)
  - překážející proces odložit – prostor pro růst procesu
  - odložit žadatele o paměť, dokud nebude prostor
  - proces zrušit (odkládací paměť je plná)



Odkládání celých procesů

- proces – dva rostoucí segmenty
  - data, zásobník (co se kde alokuje?)
  - možnost rozrůstání proti sobě
  - překročení velikosti – přesun, odložit, zrušit

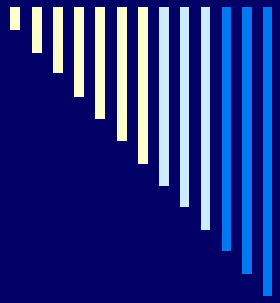


Alokace odkládací oblasti

- tj. jak vyhradit prostor pro proces na disku

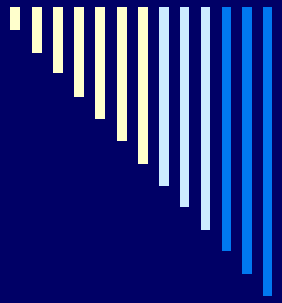
- na celou dobu běhu programu
- alokace při každém odložení

- stejné algoritmy jako pro přidělení paměti
- velikost oblasti na disku
  - násobek alokační jednotky disku



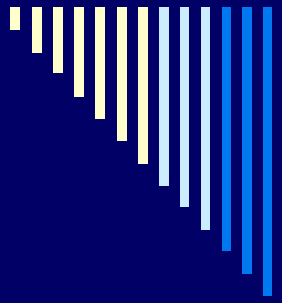
Virtuální paměť

Proces > dostupná fyzická paměť



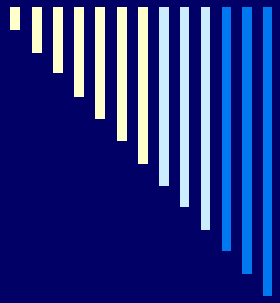
Virtuální paměť

- program větší než dostupná fyzická paměť
- mechanismus překrývání (overlays)
- virtuální paměť



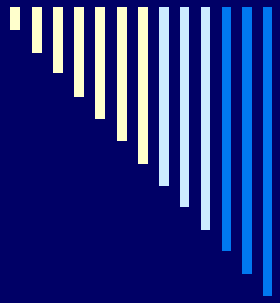
Překrývání (overlays)

- program – rozdělen na moduly
- start – spuštěna část 0, při skončení zavede část 1 …
- časté zavádění některých modulů
  - více překryvných modulů + data v paměti současně
  - moduly zaváděny dle potřeby (nejen 0,1,2,..)
  - mechanismus odkládání (jako odkládání procesů)
- kdo zařizuje zavádění modulů?
- kdo navrhuje rozdělení dat na moduly?



Překrývání

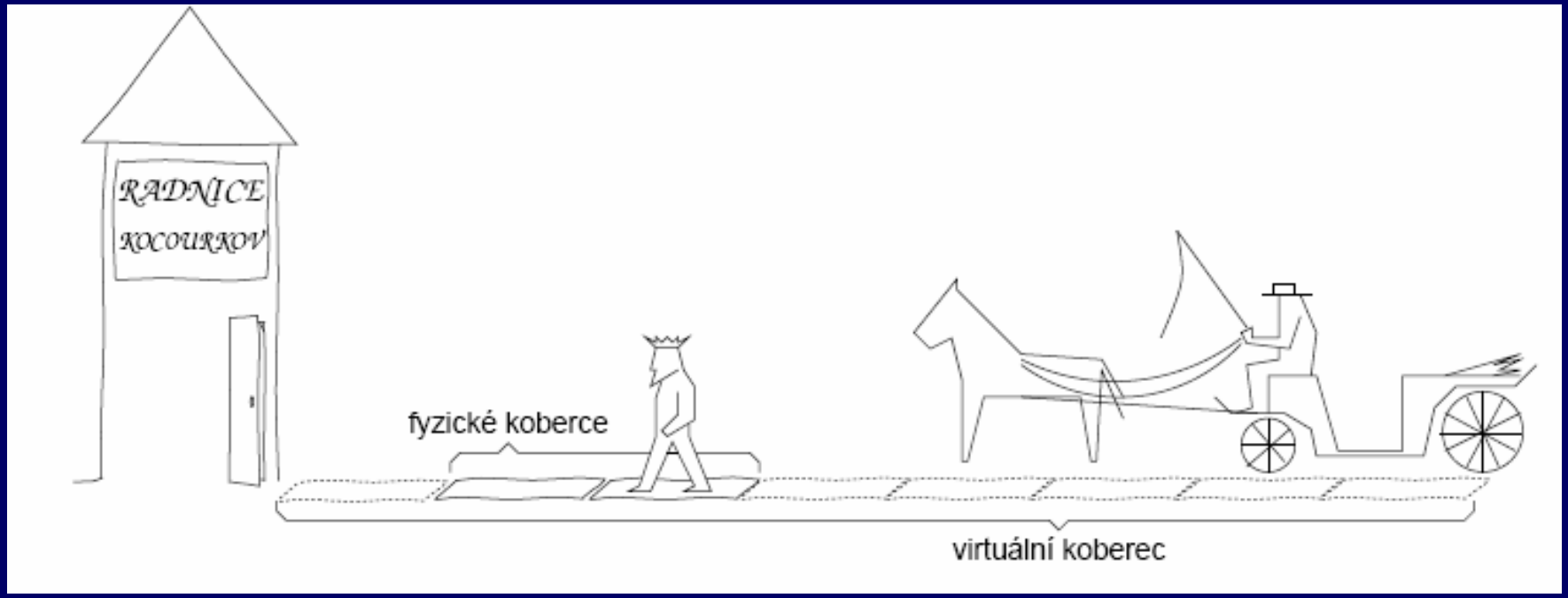
- zavádění modulů zařizuje OS
- rozdělení programů i dat na části – navrhuje programátor
  - vliv rozdělení na výkonnost, komplikované
  - pro každou úlohu nové rozdělení
- příklad – overlay.pas
- snaha, aby se o vše postaral OS



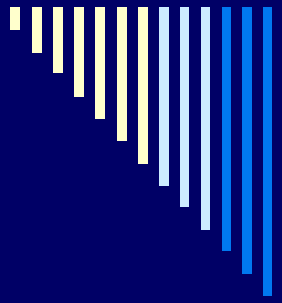
Virtuální paměť

- potřebujeme rozsáhlý adresový prostor
- ve skutečné paměti je pouze část adresového prostoru
  - jinak by to bylo příliš drahé
- zbytek může být odložen na disku
- kterou část v paměti?
  - tu co právě potřebujeme ☺

Historie – královský koberec



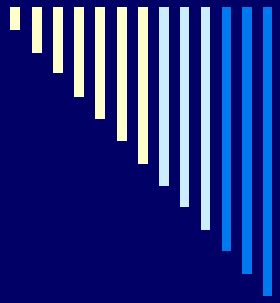
Na pokrytí celé cesty stačí pouze dva fyzické koberce



Virtuální adresy

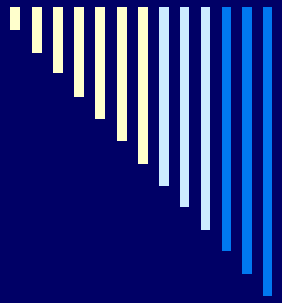
- fyzická paměť slouží jako cache virtuálního adresního prostoru procesů

- procesor – používá virtuální adresy
- požadovaná část VAProstoru JE ve fyzické paměti
  - MMU převede VA=>FA, přístup k paměti
- požadovaná část NENÍ ve fyzické paměti
  - OS ji musí přečíst z disku
  - I/O operace – přidělení CPU jinému procesu
- většina systémů virtuální paměti používá stránkování



Mechanismus stránkování (paging)

- program používá virtuální adresy
- rychle zjistit, zda je požadovaná adresa v paměti
  - ANO – převod VA => FA
- co nejrychlejší – při každém přístupu do paměti

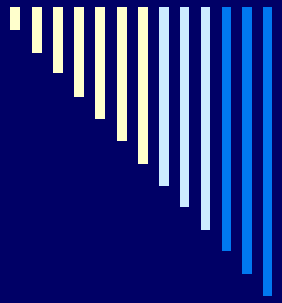


Pojmy – důležité !!

- VAP – stránky (pages) pevné délky
  - délka mocnina 2, nejčastěji 4KB, běžně 512B - 8KB
- fyzická paměť – rámce (page frames) stejné délky

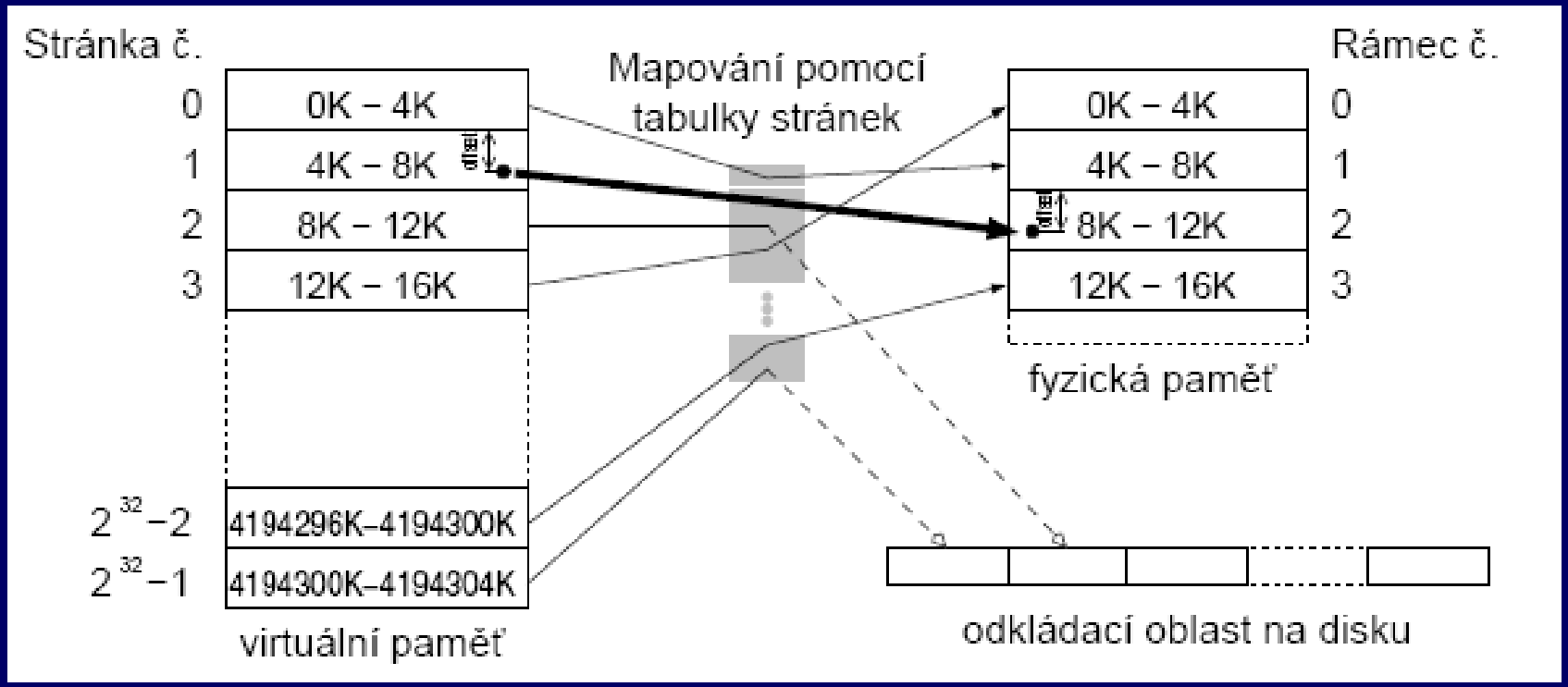
- rámec může obsahovat PRÁVĚ JEDNU stránku
- na známém místě v paměti – tabulka stránek

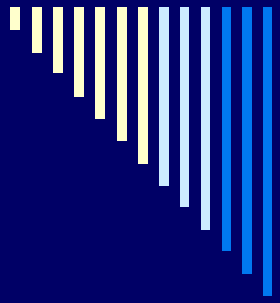
- tabulka stránek poskytuje mapování virtuálních stránek na rámce



Opakování

- virtuální adresní prostor
- fyzický adresní prostor
- procesy používají VA nebo FA?
- co dělá MMU?
- k čemu slouží tabulka stránek?
- stránka
- rámec


Stránka č.
Mapování pomocí tabulky stránek
Rámec č.
0
1
2
3
0K – 4K
4K – 8K
8K – 12K
12K – 16K
offset
0K – 4K
4K – 8K
8K – 12K
12K – 16K
fyzická paměť
$2^{32}-2$
$2^{32}-1$
4194296K–4194300K
4194300K–4194304K
virtuální paměť
odkládací oblast na disku



Pokračování

- Pokračování je v souboru p9vm.pdf