Jednoduché stránkování

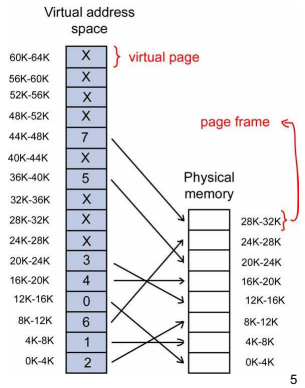
- **Hlavní paměť**
  - rozdělená na malé úseky stejné velikosti (např. 4kB) nazývané **rámce (frames).**
- **Program**
  - rozdělen na malé úseky stejné velikosti nazývané **stránky (pages)**
- **Velikost rámce a stránky je stejná.**
- **Celý program** je nahrán do **volných rámců** hlavní paměti.
- OS si musí pamatovat **rámce přidělené jednotlivým procesům** (např. pomocí tabulky stránek,…)
- OS si musí pamatovat **volné rámce** v hlavní paměti.

Virtuální paměť

- V 32 bitovém OS (např. Unix), **jeden proces může mít 4 části**, každou o velikosti (maximum) 1GB:
  - **text** (instrukce kódu),
  - **data** (statická a dynamická),
  - **shared text** (sdílené knihovny),
  - **shared data** (sdílená paměť).
- **Problém**
  - Pokud OS umožňuje, aby bylo současně spuštěno až 64k procesů, pak bychom potřebovali dohromady 256 TB paměti.
- **Řešení**
  - **Virtuální paměť** = proces je automaticky (pomocí OS) rozdělen na menší kousky.
  - Ve fyzické paměti jsou pouze kousky aktuálně používané, zbytek procesu je na disku.

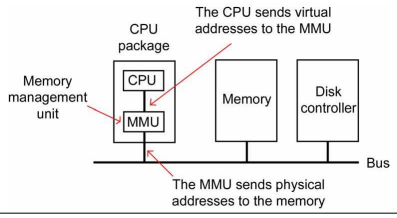
Virtuální paměť se stránkování

- Většinou je **virtuální paměť** kombinována se **stránkování**.
- **Princip**
  - Proces používá adresy, kterým se říká **virtuální adresy** a které tvoří **virtuální adresový prostor.**
  - Virtuální adresový prostor je rozdělen na stejně velké souvislé úseky nazývané **virtuální stránky (virtual pages)** (typicky 4KB).
  - Korespondující úseky ve **fyzické paměti** jsou nazývány **rámce stránek (page frames)**.
  - V hlavní paměti jsou pouze **stránky aktuálně používané**.



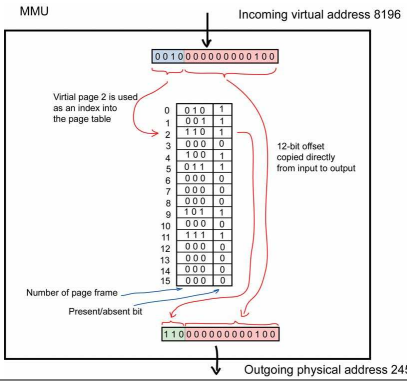
Memory Management Unit

- Proces adresuje paměť pomocí virtuálních adres (např. MOV reg, va).
- **Memory Management Unit** (MMU)
  - překládá virtuální adresu na fyzickou.
- **Výpadek stránky (Page fault)**
  - Pokud není virtuální stránka ve fyzické paměti, MMU způsobí, aby CPU požádalo OS o nahrání příslušné stránky do fyzické paměti.
  - OS nejdříve definuje, který rámec fyzické paměti je třeba uvolnit, a pak do něj nahraje obsah požadované virtuální stránky z disku.



Tabulka stránek

- MMU: **číslo_fyzického_rámce = f (číslo_virtuální_stránky)**
- Zobrazení *f()* může být implementováno pomocí **tabulky stránek**.



Tabulka stránek - problémy

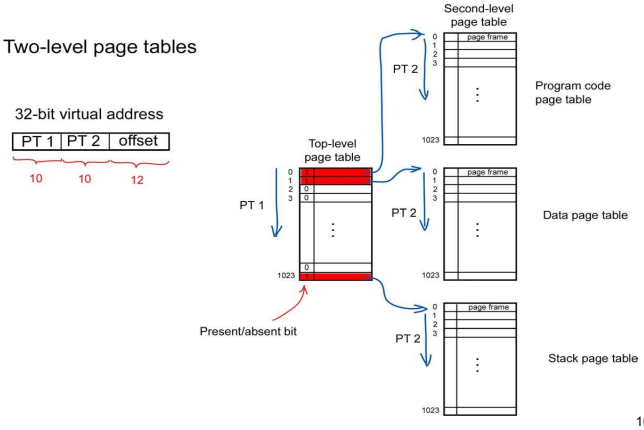
- Tabulka stránek může být **extrémně velká**.
  - 32-bitový virtuální adresový prostor bude mít při velikosti stránek 4-KB jeden milion stránek.
  - Tabulka stránek pak bude mít jeden milion položek.
  - Každý proces potřebuje svojí vlastní tabulku stránek (protože má svůj vlastní virtuální adresový prostor).
- Překlad adres by měl být **velmi rychlý.**
  - Překlad virtuální adresy na fyzickou musí být prováděn při každém přístupu do paměti.

Víceúrovňová tabulka stránek

- Proces **obvykle používá pouze podmnožinu** adres svého virtuálního procesu.
- Stačilo by mít v paměti pouze ty položky z tabulky stránek, které bude OS potřebovat při překladu.
- **Příklad**:
  - Mějme **32-bitový virtuální adresový prostor** s **4KB stránkami**.
  - Předpokládejme, že **proces bude skutečně používat pouze 12MB**:
    - dolní 4MB paměti pro kód programu,
    - následující 4MB pro data,
    - horní 4MB pro zásobník.
  - Ačkoli proces má virtuální adresový prostor veliký 1MB (tzn. 1M položek v tabulce stránek), stačí mít pouze **čtyři tabulky stránek**, každou mající 1K položek:
    - top-level page table,
    - program code page table,
    - data page table,
    - stack page table.

Víceúrovňová tabulka stránek (2)

Two-level page tables

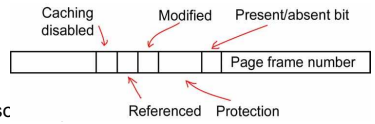


Víceúrovňová tabulka stránek (3)

- **Present/absent bity** 1021 položek v top-level page table jsou nastaveny na 0, protože virtuální stránky s nimi spojeny nebyly zatím používány.
- Při pokusu přístupu k těmto stránkám dojde k výpadku stránky a potřebné informace budou nahrány do paměti.
- Obecně lze tabulku stránek rozdělit do libovolného počtu úrovní.
- V praxi se z **důvodu rychlosti překladu adres používají pouze dvou a tříúrovňové tabulky**.
- Většina OS používá **demand paging**.
  - Když je proces spuštěn, nahrají se do RAM pouze první stránky kódu a první stránky dat.
  - Ostatní stránky budou nahrány do RAM až v okamžiku, kdy budou potřeba.
- **Výhody**: malá velikost tabulek v paměti.
- **Nevýhody**: pomalejší překlad.

Položka v tabulce stránek

- Její struktura je **závislá na architektuře CPU**, ale obvykle obsahuje:
- **Page frame number**
- **Present/absent bit**
  1 – stránka je v RAM,
  0 – stránka není v RAM,
  přístup na stránku způsc
- **Protection bits**
  - 3 bits - reading, writing, executing.
- **Modified bit**
  - Když je obsah stránky modifikován, HW automaticky nastaví bit na 1.
  - Když OS uvolňuje rámec stránky:
    - musí obsah stránky uložit na disk pokud je „Modified bit“ roven 1.
    - jinak může nahrát do rámce rovnou novou stránku.



Položka v tabulce stránek (2)

- **Referenced bit**
  - Kdykoliv je ke stránce přistupováno (pro čtení nebo zápis), je tento bit nastaven na 1.
  - Hodnota tohoto bitu je používána algoritmy pro náhradu stránek.
- **Caching disabled bit**
  - Je důležitý pro stránky, které jsou mapovány na registry periferních zařízení.
  - Pokud čekáme na V/V (např. v cyklu), musíme použít hodnoty z fyzických HW registrů, nikoliv (starý) obsah v paměti.

Translation Lookaside Buffer (TLB)

- Většina programů provádí velký počet přístupů k malému počtu stránek.
- **Translation Lookaside Buffer** (TLB)
  - Je organizovaný jako **asociativní paměť.**
  - Obsahuje **posledně používané položky tabulek stránek.**
  - TLB je obvykle uvnitř MMU a obsahuje desítky položek.

Valid	Virtual page	Modified	Protection	Page frame
1	140	0	RWX	31
1	12	0	R	12
1	25	1	R X	13
1	256	0	RWX	23
1	2	1	RWX	5
1	311	1	RWX	78

Translation Lookaside Buffer (2)

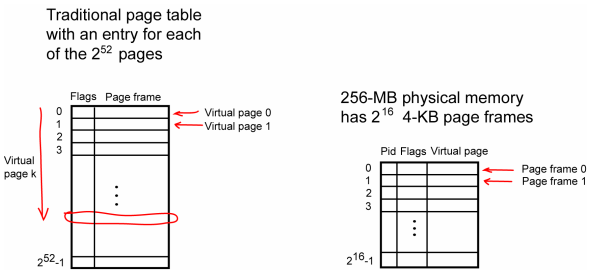
- Při překladu virtuální adresy(VA), MMU nejdříve **hledá informaci o VA v TLB**.
- Hledávání v TLB probíhá **paralelně.**
- Pokud informace o VA **existuje v TLB**, MMU použije tuto informaci pro překlad VA a nemusí hledat v tabulce stránek.
- Pokud informace **v TLB není**, MMU vyvolá **TLB fault**. OS pak musí **načíst informaci z tabulky stránek.**

Invertovaná tabulka stránek

- V klasické tabulce stránek **číslo virtuální stránky slouží jako index do tabulky**.
- V **32 bitových počítačích**, každý proces má 32 bitovou virtuální adresu. Při velikosti stránky 4kB, tabulka stránek každého procesu má 1M položek. Se 4B na každou položku, **tabulka stránek zabírá 4MB**.
- V **64 bitových počítačích** se 64 bitovou virtuální adresou je situace ještě více zřejmější. Při 4kB stránkách, tabulka stránek má 2^{52} položek.

Invertovaná tabulka stránek (2)

- Ačkoliv virtuální adresový prostor je obrovský, fyzický prostor RAM je stále malý.
- Tabulka stránek muže být organizována kolem **fyzické paměti**.
- V **invertované tabulce stránek**, *i*-th položka obsahuje informaci o virtuální stránce, která je nahrána v **rámci *i***.



Invertovaná tabulka stránek (3)

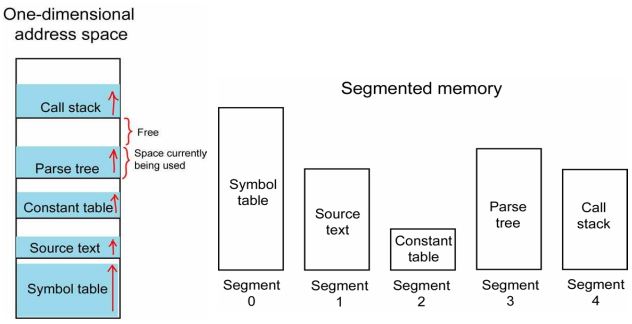
- Invertovaná tabulka stránek s obvykle **používá společně s TLB**.
  - Při nalezení v TLB, se invertovaná tabulka nepoužije.
  - Jinak musíme hledat v invertované tabulce stránek.
- Sekvenční hledání v tabulce může být urychleno pomocí **rozptylovací tabulky**.

Virtuální paměť vs. Segmentace

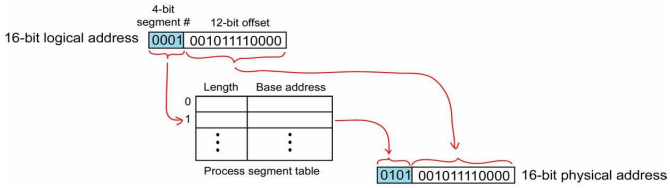
- **Virtuální paměť**
  - Proces má jednorozměrný virtuální adresový prostor.
  - Pro některé problémy, **dva nebo více oddělených adresových prostorů** (segmentů) je vhodnější.
- **Segmentace**
  - Virtuální adresový prostor procesu je rozdělen na několik **segmentů**.
  - Segment je lineární posloupnost adres, od 0 do nějaké maximální adresy.
  - Různé segmenty mohou mít různé délky, **délka segmentu se může měnit během výpočtu.**
  - Různé segmenty mohou mít **rozdílnou ochranu** a mohou být **sdílené**.

Jednoduchá segmentace

- **Příklad:** Překladač si udržuje několik tabulek a datových struktur, jejichž velikost se během překladu dynamicky mění.



Jednoduchá segmentace



- **Logická adresa** se skládá ze dvou částí: **číslo segmentu** a **offsetu**.
- Segmentace je obvykle **viditelná pro programátora**.

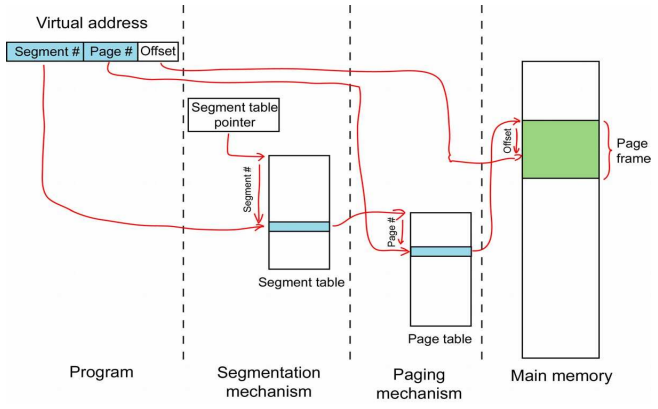
Segmentace se stránkováním

- **Stránkování**
  - Je transparentní pro programátora.
  - Eliminuje externí fragmentaci a poskytuje efektivní využití hlavní paměti.
- **Segmentace**
  - Je viditelná pro programátora.
  - Je vhodná pro dynamicky rostoucí datové struktury, modularitu, a podporuje sdílení a ochranu.
- **Segmentace se stránkováním**
  - Virtuální adresový prostor je rozdělen na **několik segmentů**.
  - Každý segment se skládá z **stejně velkých stránek**, které jsou stejně velké jak rámce v hlavní paměti.

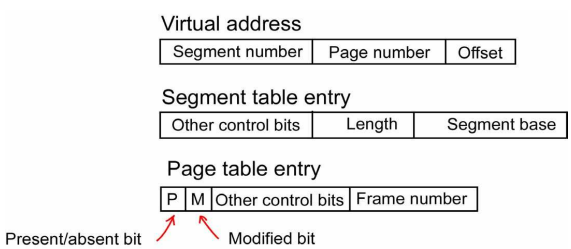
Segmentace se stránkováním (2)

- Z hlediska **programátora**
  - Virtuální adresa se skládá z **čísla segmentu** a **offsetu uvnitř segmentu**.
- Z hlediska **systému**
  - Offset segmentu se skládá z **čísla stránky** a **offsetu uvnitř stránky**.

Segmentace se stránkováním (3)

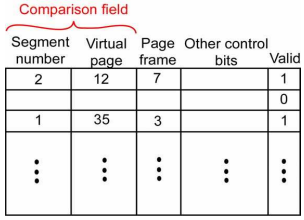


Segmentace se stránkováním (4)



- „Segment base“ ukazuje na začátek tabulky stránek pro daný segment.
- „Other control bits“ v tabulce segmentů slouží pro definici přístupových práv a sdílení mezi procesy.

Segmentace se stránkováním (5)



- Segmentace se stránkováním se může **používat společně s TLB**.
- Při překladu virtuální adresy:
  - MMU se nejdříve podívá zda není informace v TLB.
  - Pokud ano, použije pro překlad číslo rámce z TLB.
  - Jinak MMU hledá v tabulce segmentů,…