



Operační systémy

Paměť

Petr Krajča



Katedra informatiky
Univerzita Palackého v Olomouci



- zásadní část počítače
- uložení kódu a dat běžících procesů i operačního systému
- přístup k zařízením (DMA)
- virtuální paměť umožňuje např. přístup k souborovému systému
- z HW pohledu může být operační paměť realizovaná různými způsoby (DRAM, SRAM, (EEP)ROM, ale i HDD, SSD, flash paměti)
- přístup CPU k paměti nemusí být přímočarý (L1, L2 a L3 cache)
- pro jednoduchost budeme HW stránku věci zanedbávat
- Ulrich Drepper: What every programmer should know about memory (<http://lwn.net/Articles/250967/>)



- evidence prostoru volného a přiděleného procesům
- přidělování a uvolňování paměti procesů
- přesunutí (přiděleného prostoru) – program by neměl být závislý na místě, kde se v paměti nachází (nutné k umožnění swapování)
- ochrana (přiděleného prostoru) – jednotlivé procesy by měly být izolovány
- sdílení – pokud je to žádoucí, mělo by být možné sdílet některé části paměti mezi procesy (2× spuštěný stejný program)
- fyzická organizace – paměť může mít více částí/úrovní (RAM, disk); program jako takový se nemusí vlézt do dostupné paměti RAM
- logická organizace – paměť počítače (spojitý prostor, „sekvence bytů“) vs. typický program skládající se z modulů (navíc některých jen pro čtení nebo ke spouštění)



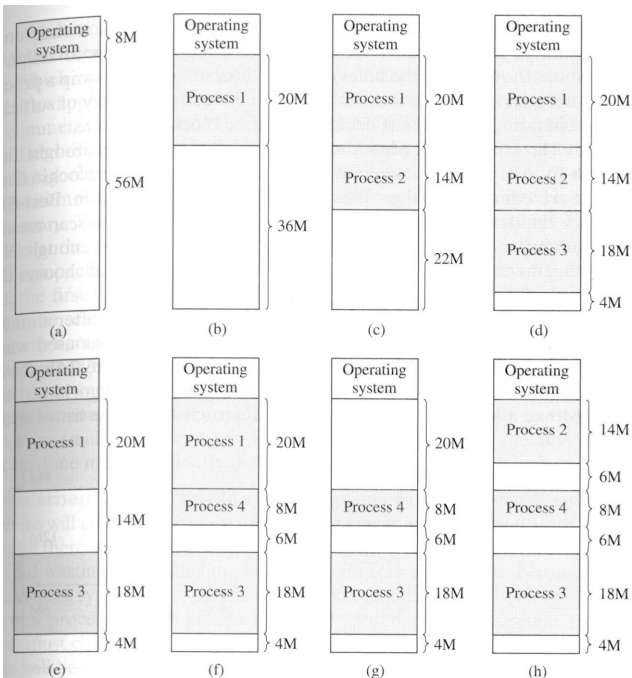
- v současných OS existuje několik různých nezávislých *adresních prostorů*, způsobů číslování paměťových buněk
- každý proces by měl mít vlastní paměťový prostor (izolace)
- různá zařízení mají vlastní (odlišné) adresní prostory
- ve spolupráci s hardwarem dochází k mapování fyzické paměti do adresního prostoru procesu
- např. grafická paměť může být namapována od adresy 0xD0000000



- předpokládejme, že OS je umístěn v nějaké části paměti
- nejjednodušší přístup je rozdělit paměť na souvislé bloky stejné velikosti (např. 8MB)
- pokud program potřebuje víc paměti, musí se sám postarat o odsun/načtení dat do sekundární paměti
- pokud program potřebuje míň paměti, dochází k neefektivnímu využití paměti – tzv. **vnitřní fragmentaci**
- výhodou je, že lze velice jednoduše vybrat umístění kam načíst proces
- problém přesunutí → relativní adresování
- v IBM OS/360 (historická záležitost)
- vylepšení: rozdělit paměť na bloky různých velikostí (např. 2, 4, 6, 8, 12 a 16 MB)
- jedna vs. více front (každá fronta pro procesy, které se vlezou do dané paměti)



- každý program dostane k dispozici tolik paměti, kolik potřebuje
- sníží se tak míra vnitřní fragmentace
- po čase dochází k **vnější fragmentaci** (paměť je volná, ale je rozkouskovaná, tj. je problém přidělit větší blok)





- strategie přiřazování paměti
 - **first fit** – začne prohledávat paměť od začátku a vezme první vhodný blok
 - **next fit** – začne prohledávat paměť od místa kam se podařilo umístit blok naposledy
 - **best fit** – přiřadí nejmenší vyhovující blok (v paměti jsou nevyužitá místa, ale jsou co nejmenší)
 - **worst fit** – vezme největší blok (v paměti nejsou malé nevyužitá místa, ale nelze spustit větší program)
- informace o volném místě lze ukládat do *bitmap* nebo *seznamů volného místa*
- v jiných obměnách používána i dnes (dynamická alokace paměti)
- vylepšení: zahuštění (compaction) – procesy jsou v paměti přesunuty tak, že vznikne souvislý blok volného místa (časově náročná operace)

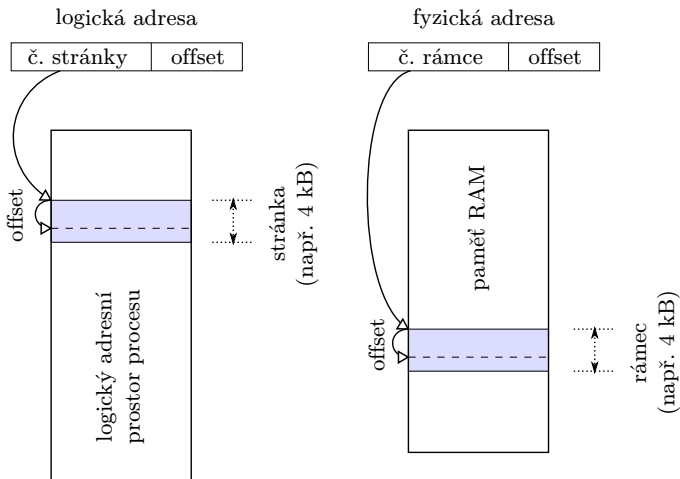


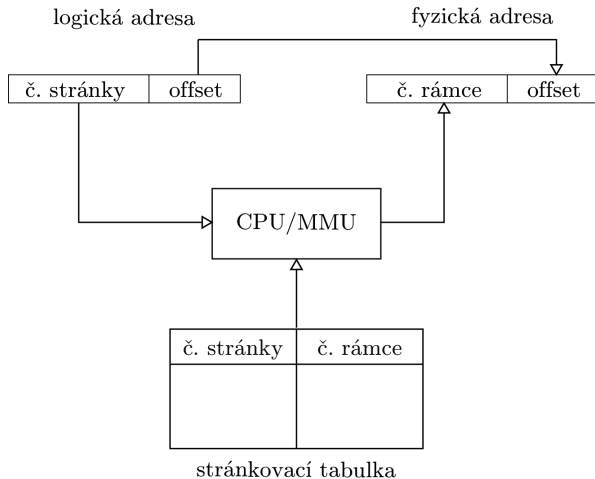
- adresní (logický) prostor procesu je rozdělen na menší úseky – **stránky** (pages)
- fyzická paměť je rozdělena na úseky stejné délky – **rámce** (frames, page frames)
- provádí se mapování úseků logické paměti (stránek) → na úseky fyzické paměti (rámce)
- procesy už nemusí být umístěny v souvislých blocích paměti
- výpočet fyzické adresy musí být implementovaný hardwarově (efektivita)
- OS a CPU si udržuje **stránkovací tabulku** (tabulka mapující č. stránky na č. rámce)
- logická adresa pd má dvě části, kde p je číslo stránky a d je offset
- fyzická adresa fd má dvě části, kde f je číslo rámce v tabulce stránek příslušného stránce p
- p a f horní (nejvýznamější) bity adresy, d spodní bity adresy
- při překladu adresy dojde k nahrazení čísla stránky číslem rámce
- velikost stránek je mocnina 2 (typicky 4 kB)

Vztah adres a rozdělení paměti



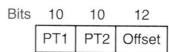
```
mov eax, [0x12345678] ;; logicka adresa  
;; 0x12345 ;; c. stranky  
;; 0x678 ;; offset
```



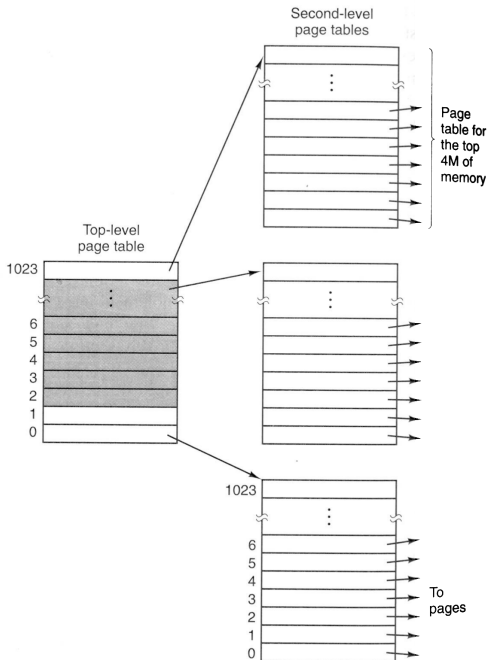




- uvažujme velikost stránky 4 kB a velikost adresy 32 bitů
- pak velikost stránkovací tabulky je 1 milion záznamů
- je nepraktické udržovat tak velkou tabulku (obzvlášť pro každý proces)
- používá se víceúrovňová tabulka – systém obsahuje více tabulek
- část logické adresy udává tabulku, další část index v tabulce a další část offset
- např. pro 4 kB stránky může být toto rozložení 10-10-12 bitů
- tzn. systém k adresaci používá 1024 tabulek po 1024 záznamech
- nevyužité tabulky mohou být prázdné
- v praxi se používají i tří a čtyřúrovňové tabulky



(a)





- cache procesoru obsahující hodně používané části stránkových tabulek
- pro danou stránku uchovává adresu rámce
- pokud je adresa v cache (cache hit) je hodnota vrácena velice rychle (cca 1 hodinový cyklus)
- pokud hodnota není v cache (cache miss) načtení adresy trvá delší dobu (10-100 cyklů), typický miss rate $< 1\%$ \rightarrow průměrný přístup k zjištění rámce je ~ 1.5 hodinového cyklu
- princip lokality – je vhodné programovat tak, aby data, ke kterým se přistupuje, byla na jednom místě (lokální proměnné na zásobníku)



- může být užitečné sdílet paměť (komunikace, úspora místa)
- dva stejné programy/knihovny v paměti
- **stránky více procesů** jsou navázány na **jeden rámec**

Copy-on-Write (CoW)

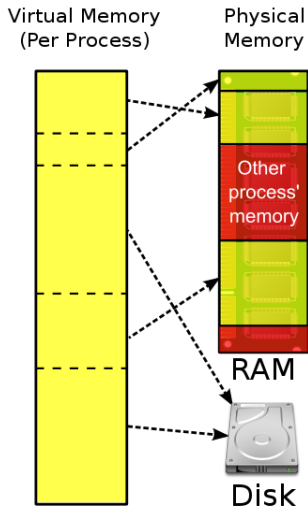
- speciální případ
- daná stránka sdílena a má nastavený příznak CoW
- dojde-li k pokusu o zápis, vznikne výjimka a procesu je vytvořena kopie rámce
- bude-li na rámec s příznakem CoW odkazovat jenom jedna stránka, příznak se odstraní
- `fork()` – vytvoří identickou kopii procesu; data jsou sice izolovaná, ale je možné sdílet kód
- úspora místa; úspora strojového času (není potřeba vytvářet kopie stránek/rámců); nízká penalizace, pokud stránky přepíšeme



- paměť je rozdělena na několik segmentů (např. kód, data, zásobník)
- speciální případ přidělování souvislých bloků paměti
- stránkování je obvykle transparentní (pro proces nerozeznatelné)
- naproti tomu segmentace umožňuje rozdělit program do logických celků (kód, data)
- při použití segmentace a stránkování programy nepracují přímo s lineární adresou
- používají adresu ve tvaru segment + offset a ta se až převádí na fyzickou adresu pomocí stránkování
- ochrana paměti přes začátek a délku segmentu + oprávnění



- paměť RAM je relativně drahá \implies nemusí vždy dostačovat
- aktuálně používaná data (např. instrukce) musí být v RAM, nepoužívaná data nemusí (velké programy \implies nepoužívané funkce a data)
- je vhodné rozšířit primární paměť (RAM) o sekundární (např. HDD)
- zvětšením dostupné paměti je možné zjednodušit vývoj aplikací (není potřeba se omezovat v množství použité paměti)
- sekundární paměť bývá řádově pomalejší
- z pohledu procesu musí být přístup k paměti transparentní
- k efektivní implementaci je potřeba spolupráce HW (MMU) a OS
- virtuální paměť (VM) je součástí soudobých OS (swapování)
 - Windows NT – stránkový soubor (pagefile.sys)
 - Linux – swap partition (ale může být i soubor)
- bezpečnost dat v sekundární paměti? (např. po vypnutí počítače)



Zdroj: Wikipedia.org

Předpoklad

- systém si eviduje, které stránka má přiřazený rámec v primární paměti a které ne
- příznak ve stránkovací tabulce (MMU)

Postup

- přístup na stránku, která **má** přiřazený rámec v primární paměti \implies bez změny
- přístup na stránku, která **nemá** přiřazený v primární paměti
 - 1 přerušení – **výpadek stránky** (page fault)
 - 2 obsluha přerušení (operační systém) načte stránku do rámce v primární paměti, aktualizuje stránkovací tabulku (je-li primární paměť plná, je potřeba nějakou jinou stránku přesunout do sekundární paměti, „odswapovat“)
 - 3 zopakuje se instrukce, která vyvolala výpadek stránky



Rezervovaná stránka

- existuje v adresním prostoru, ale nezapisovalo se do ní
- každá stránka je nejdříve rezervovaná
- vhodné pro velká pole, ke kterým se přistupuje postupně
- zásobník

Komitovaná stránka (Committed)

- stránka má rámec v primární nebo sekundární paměti
- musí řešit jádro
- paměť je často současně komitovaná i rezervovaná

Další vlastnosti

- present/absent bit – přítomnost stránky v primární paměti (HW, nutné k detekci výpadků stránek)
- reference (přístupový) bit – indikuje, že se ke stránce přistupovalo
- dirty bit – 0, pokud má stránka přesnou kopii v sekundární paměti; 1 nastaveno při změně (nutná podpora HW)
- mohou mít přístupová práva (NX bit)

- není-li volný rámec v primární paměti, je potřeba jej nějak získat

Varianta 1

- najde se stránka, která má přesnou kopii v primární i sekundární paměti (dirty bit = 0),
- nastaví se stránce dirty bit 1 a daný rámec primární paměti se použije

Varianta 2

- vybere se „oběť“ – stránka v primární paměti, která bude přesunuta do sekundární
- pokud má stránka nastavený dirty bit 1, překopíruje se obsah rámce do sekundární paměti

Poznámka

- některé stránky je možné zamknout, aby nebyly odswapovány (nutné pro jádro, rámce sdílené s HW)



- hledáme stránku, která nebude v budoucnu použita (případně bude použita v co nejbližší budoucnosti)

FIFO

- velice jednoduchý algoritmus
- stačí udržovat frontu stránek
- při načtení nové stránky je stránka zařazena na konec fronty
- pokud je potřeba uvolnit stránku, bere se první z fronty
- nevýhoda – odstraní i často užívané stránky
- Beladyho anomálie – za určitých okolností může zvětšení paměti znamenat více výpadků stránek

Least Frequently Used (LFU)

- málo používané stránky nebudou potřeba
- problém se stránkami, které byly nějaký čas intenzivně využívány (např. inicializace)

Most Frequently Used (MFU)

- právě načtené stránky mají malý počet přístupů

Least Recently Used (LRU)

- jako oběť je zvolena nejdéle nepoužívaná stránka
- je potřeba evidovat, kdy bylo ke stránce naposledy přistoupeno
- řešení:
 - 1 počítadlo v procesoru, inkrementované při každém přístupu a ukládané do tabulky stránek
 - 2 „zásobník“ stránek – naposledy použitá stránka se přesune na vrchol
- nutná podpora hardwaru

LRU (přibližná varianta)

- každá stránka má přístupový bit (*reference bit*) nastavený na 1, pokud se ke stránce přistupovalo
- na počátku se nastaví reference bit na 0
- v případě hledání oběti je možné určit, které stránky se nepoužívaly
- varianta
 - možné mít historii přístupových bitů pro danou stránku
 - jednou za čas se bity posunou doprava
 - nastavuje se nejvyšší bit
 - přehled o používání stránky \implies bity jako neznaménkové číslo \implies nejmenší = oběť



Algoritmus druhé šance

- založen na FIFO
- pokud má stránka ve frontě nastavený přístupový bit, je nastaven na nula a stránka zařazena nakonec fronty
- pokud nemá, je vybrána jako oběť
- lze vylepšit uvážením ještě dirty bitu

Buffer volných rámců (optimalizace)

- proces si udržuje seznam volných rámců
- přesun oběti je možné udělat se zpožděním
- případně, pokud je počítač nevytížený, je možné ukládat stránky s dirty bitem na disk a připravit se na výpadek (nemusí být vždy dobré)

Minimální počet rámců

- každý proces potřebuje určité množství rámců v primární paměti (např. movsd potřebuje v extrémním případě 6 rámců)
- stránkovací tabulka(y) musí být opět v rámci
- přidělování rámců procesům
 - rovnoměrně
 - podle velikosti adresního prostoru
 - podle priority
 - v případě výpadků stránek podle priority (globální alokace rámců)
- pokud počet rámců klesne pod nutnou mez, je potřeba celý proces odsounout z paměti
- hrozí thrashing (proces začne odsouvat stránky z paměti, které bude potřebovat)



- stránky mají velikost 2^n , typicky v intervalu $2^{12} - 2^{22}$, i.e., 4 kB – 4 MB
- závisí na HW architektuře (může být i víc nebo míň)
- z pohledu fragmentace je vhodnější mít stránky menší
- více menších stránek zabírá místo v TLB \implies časté cache miss
- při přesunu do swapu může být velká stránka výhodnější (přístupová doba)
- některé systémy umožňují používat různé velikosti
- Windows NT \leq 5.1 & Solaris velké stránky pro jádro malé pro uživatelský prostor
- Windows Vista a novější (podporuje, ale je nutné explicitně povolit)
- Linux – velké stránky 2.6.38



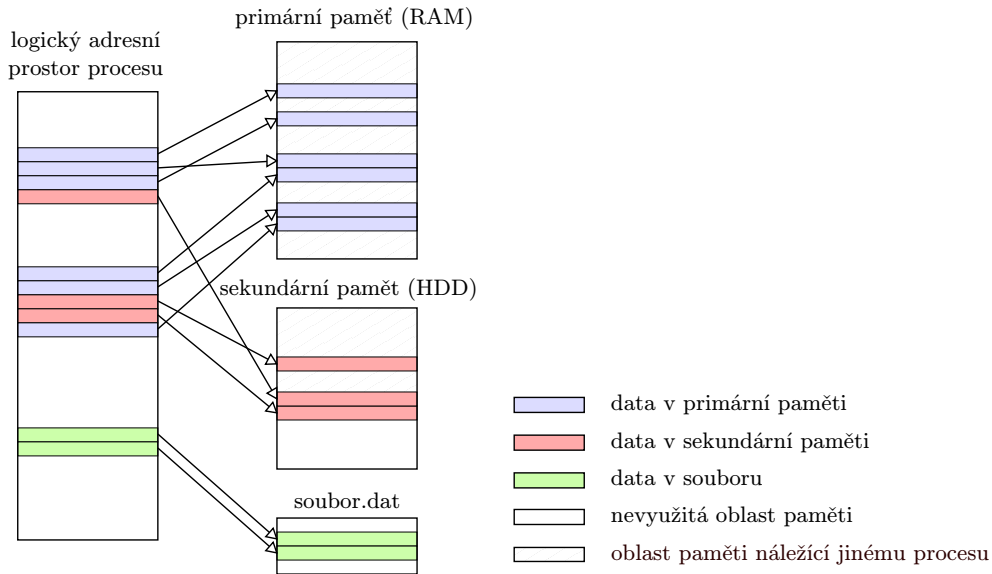
Soubory

- operace open, read, write mohou být pomalé (systémové volání)
- mechanismus, který je použitý pro práci se sekundární paměti, lze použít pro práci se soubory
- soubor se načítá do paměti po blocích velikosti stránky podle jednotlivých přístupů (demand paging)
- k souboru se přistupuje pomocí operací s pamětí (přiřazení, memcpy, ...)
- data se nemusí zapisovat okamžitě (ale až s odmapováním stránky/souboru)
- více procesů může sdílet jeden soubor \implies sdílená paměť (WinNT)
- možnost použít copy-on-write

I/O

- lze namapovat zařízení do paměti (specifické oblasti) a přistupovat k němu jako k paměti
- pohodlný přístup, rychlý přístup
- např. grafické karty

Ilustrace uspořádání virtuální paměti a mapování stránek



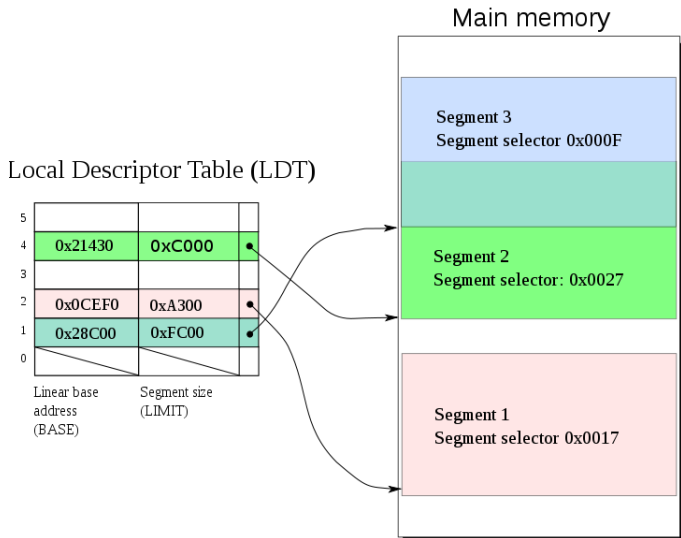
- **logická adresa** – pracuje s ní proces; 48 bitů (16 selektor segmentu; 32 offset); segment je často implicitní
 - instrukce implicitně brány z kódového segmentu (registr CS)
 - přístup do paměti přes ESP nebo EBP implicitně používá zásobníkový segment (registr SS)
 - ostatní instrukce využívají převážně datový segment (registr DS)
 - další segmenty např. pro lokální data vláken (thread-local storage)

```
jmp 0xdeadbeef          ;; -->  jmp cs:0xdeadbeaf
mov eax, [esp + 4]      ;; -->  mov [ss:esp + 4]
mov eax, 0x12345678     ;; -->  mov [ds:0x12345678]
mov eax, [gs:ebx + 10]  ;; explicitni pouziti segmentu
```

- **lineární (virtuální) adresa** – v adresním prostoru procesu (32 bitů)
- **fyzická adresa** – „číslo bytu“ přímo v primární paměti (32 bitů, s PAE 36); sdílená dalšími HW zařízeními; (nevyužité adresy mohou mít další použití, např. SWAP)

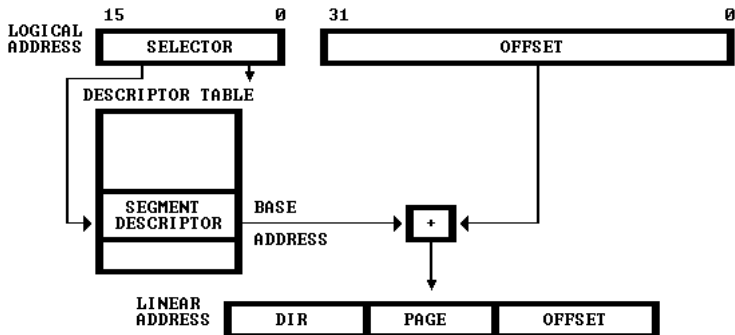


- paměť je možné rozdělit na segmenty (kód, data, zásobník, etc.)
- pro každý segment lze nastavit oprávnění (ochrana paměti)
- segmenty jsou popsány pomocí deskriptorů 8 B záznam
 - báze
 - limit (velikost segmentu)
 - požadovaná úroveň oprávnění (ring 0-3)
- deskriptory segmentů uloženy v
 - Global Descriptor Table (GDT) – sdílená všemi procesy
 - Local Descriptor Table (LDT) – každý proces má vlastní
- každá může mít až 8192 záznamů
- přístupné přes registry GDTR, LDTR
- první záznam v GDT „null“ deskriptor



- logická adresa \implies linární adresa (segmentace)
- ověří se oprávnění a limit (přístup za hranici segmentu) \implies neoprávněný přístup
- báze segmentu je sečtena s offsetem \implies lineární

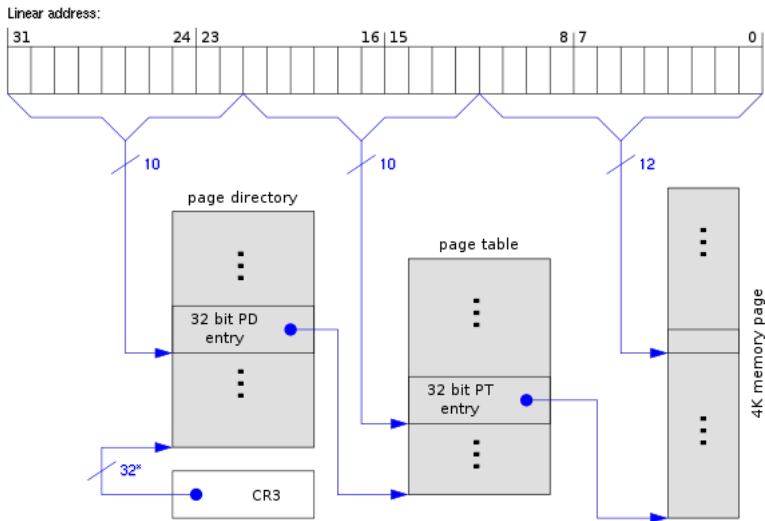
Figure 5-2. Segment Translation





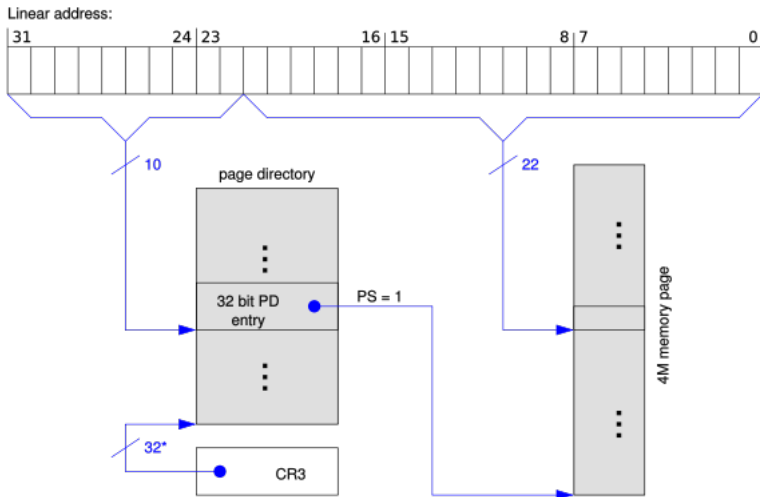
- linární adresa \implies fyzická adresa
- standardní stránka/rámec: 4 kB
- hierarchická struktura
 - adresář stránkových tabulek (Page Directory)
 - adresář stránek (Page Tables)
 - offset
- adresáře mají velikost jedné stránky, každá položka 4 B \implies 1024 záznamů
- lineární adresa rozdělena na 10 + 10 + 12 bitů (PDI + PTI + offset)
- maximální kapacita 4 GB
- adresa PDT v CR3 (zarovnané na celé stránky)
- nastavením příznaku v PDT (pro adresu rámce se používá jen 20 b), lze obejít přepočítání přes PT a používat stránky velikosti 4 MB (zbytek adresy je offset)
- velikost stránek lze kombinovat

I386: Překlad adres III. (stránkování – 4 kB stránka)



*) 32 bits aligned to a 4-KByte boundary

I386: Překlad adres IV. (stránkování – 4 MB stránka)

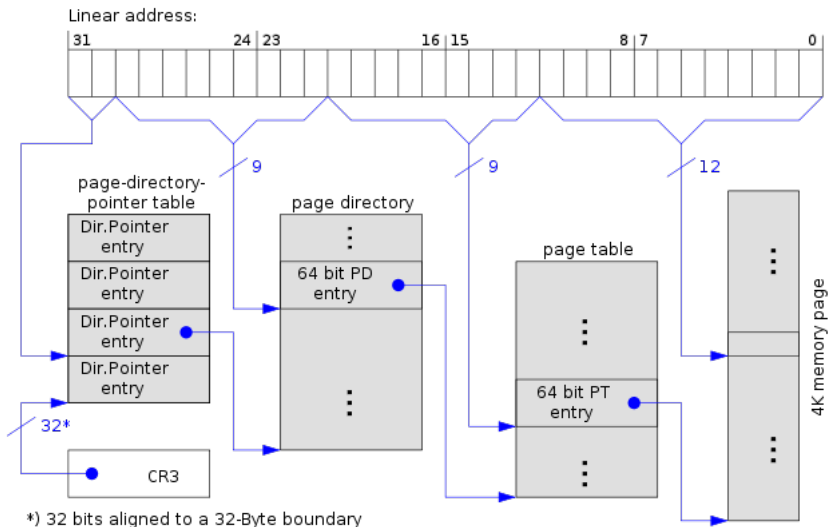


*) 32 bits aligned to a 4-KByte boundy



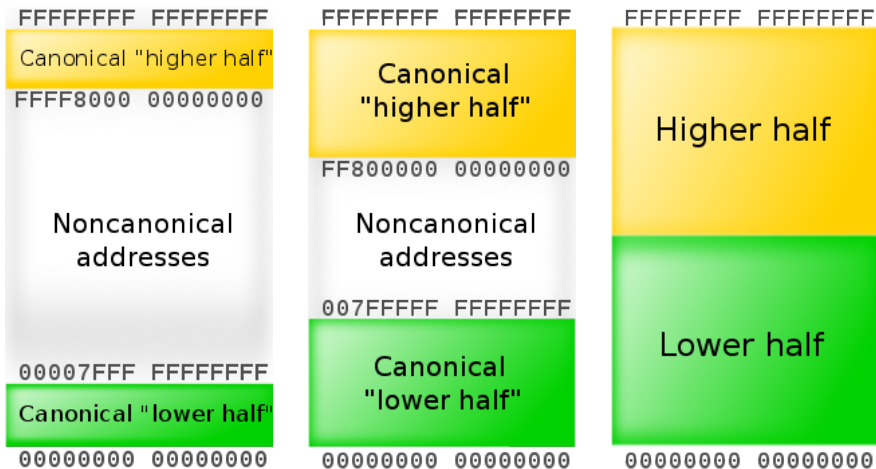
- od Pentium Pro
- každá tabulka 4 kB, ale velikost záznamu 8 B \implies 512 záznamů
- stránkování trojúrovňové
- adresa rozdělena na 2 + 9 + 9 + 12 bitů
 - 2b – ukazatel na adresář tabulek stránek (Page Directory Pointer Index)
 - 9b – ukazatel v adresáři tabulek stránek
 - 9b – ukazatel v tabulce stránek
 - 12b – offset
- velké stránky 2MB \implies offset 21 bitů
- potenciální rozšíření

I386: Překlad adres VI. (PAE – 4 KB stránka)





- segmenty existují, ale nepoužívají se k adresaci, pouze ke kontrole oprávnění
- deskriptor kódového segmentu se používá k přechodu mezi 32- a 64bitovým režimem
- současné procesory AMD64:
 - 36-46bitové fyzické adresy (max. 52; způsob stránkování)
 - 48bitové logické adresy (max. 64; velikost registrů)
- možnost rozšíření \implies kanonické adresy
- nejvyšší platný bit je okopírován do vyšších bitů
- dělí paměť na tři bloky



Zdroj: Wikipedia.org



- zavedena čtyřúrovňová hierarchie
- záznamy v tabulkách stránek mají 8 B
- při 4 kB stránkách $\implies 4 \times 9 + 12 = 48$ adresovatelných bitů
- pro 2 MB stránky vynechaná jedna úroveň, možnost použít 4 rezervované bity $\implies 52$ bitů



- ochrana paměti na úrovni segmentů je zapnutá a nejde vypnout (ale jde nastavit, aby nebyla účinná)
- prováděné kontroly
 - kontrola typu segmentu (některé segmenty nebo segmentové registry mohou být použité jenom určitým způsobem)
 - kontrola velikosti segmentu (limitu), i.e., jestli program našahá za hranice segmentu
 - kontrola oprávnění
 - omezení adresovatelné domény (omezení přístupu jen k žádoucím segmentům)
 - omezení vstupních bodů procedur (brány)
 - omezení instrukční sady
- AMD64 v long mode neprovádí některé kontroly (báze a limit jsou ignorovány)



- stránkování funguje souběžně se segmentací
- bit pro systémové stránky (zákaz přístupu z ring 3) \implies volání funkce OS (přes bránu)
- bit pro zákaz zápisu
- AMD64 + PAE mají NX bit (zákaz spouštění) \implies viry
- možnost nastavit bity na jednotlivých stránkách i adresářích \implies efektivnější



- Keprt A. Operační systémy.
- Kapitoly 12–14, tj. strany 129–167.