Operační systémy

Paměť

Petr Krajča

Katedra informatiky
Univerzita Palackého v Olomouci

Operační pameť

- zásadní část počítače
- uložení kódu a dat běžících procesů i operačního systému
- přístup k zařízením (DMA)
- virtuální pamět 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/)

Požadavky na správu paměti (Memory Managment)

- 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 ($2\times$ spuštěný stejný program)
- logická organizace – paměť počítače (spojitý prostor, ,,sekvence bytů") vs. typický program skládájící se z modulů (navíc některých jen pro čtení nebo ke spoustění)
- fyzická organizace – paměť může mít více částí/úrovní (RAM, disk); program jako takový se nemusí vlézt do dostupné paměti RAM

Adresní prostory

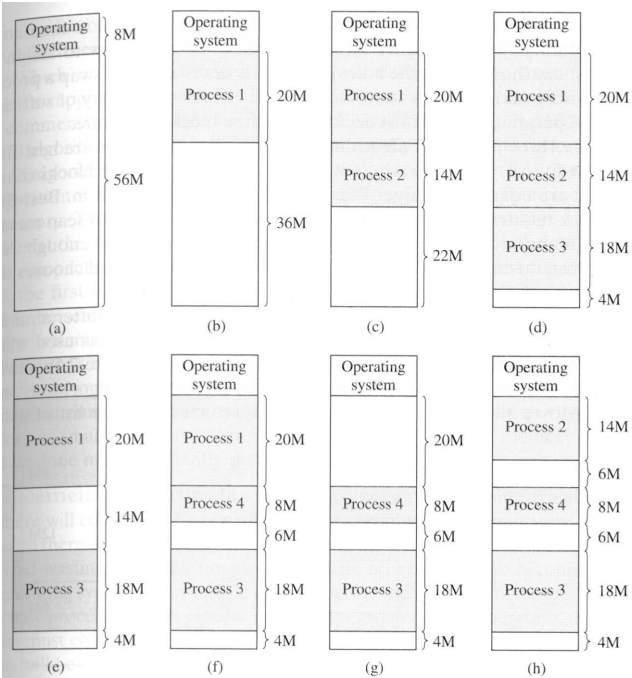
- v současných OS existuje několik různých nezávislých *adresních prostorů*, způsobů číslování paměťových buněk
- každý proces by měl mít vlastní paměťový prostor (izolace)
- různá zařízení mají odlišné adresní prostory
- ve spolupráci s hardwarem dochází k mapování fyzické paměti do adresního prostoru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

Rozdělení paměti na souvislé bloky pevné velikosti

- předpokládejme, že OS je umístěn v nějaké části paměti
- nejjednodušší přístup je rozdělit pamět na souvislé bloky stejné velikosit (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 $\rightarrow$ 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)

Přidělování paměti po blocích proměnlivé velikosti (1/2)

- 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, i.e., je problém přidělit větší blok)

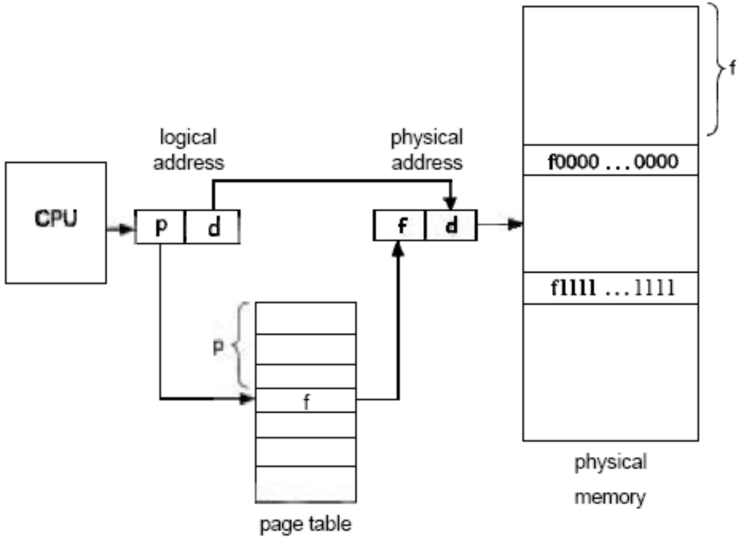

Operating system
8M
56M
(a)
Operating system
Process 1
20M
36M
(b)
Operating system
Process 1
20M
Process 2
14M
22M
(c)
Operating system
Process 1
20M
Process 2
14M
Process 3
18M
4M
(d)
Operating system
Process 1
20M
14M
Process 3
18M
4M
(e)
Operating system
Process 1
20M
Process 4
8M
6M
Process 3
18M
4M
(f)
Operating system
20M
Process 4
8M
6M
Process 3
18M
4M
(g)
Operating system
Process 2
14M
6M
Process 4
8M
6M
Process 3
18M
4M
(h)

Přidělování paměti po blocích proměnlivé velikosti (2/2)

- 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 nevyužité bloky, ale jsou co nejmenší)
  - **worst fit** – vezme největší blok (v paměti nejsou malé nevyužité bloky, ale nelze spustit větší program)
- informace o volném místu lze ukládat do *bitmap* nebo *seznamů volného* místa
- v MS-DOS (historická záležitost)
- 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)

Stránkování

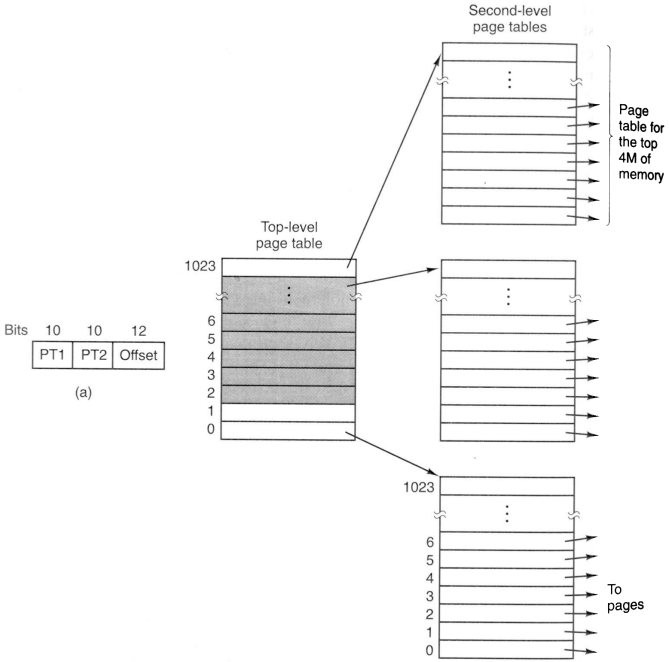
- adresní (logický) prostor procesu je rozdělen na menší úseky – stránky (pages)
- fyzická paměť je rozdělena na úseky stejné délky – rámce (frames, page frames)
- provádí se mapování logické paměti $\rightarrow$ na fyzickou
- procesy už nemusí být umístěny v souvislých blocích paměti
- výpočet fyzické adresy musí být implementovaný hardwarově (efektivita)
- CPU si udržuje *stránkovací tabulku*
- logická adresa má dvě části: pd, kde p je číslo stránky a d je offset
- fyzicka adresa vznikne jako fd, kde f je číslo rámce v tabulce stránek příslušného stránce p
- p a f horní (nejvýznamější) bity adresy, d spodní bity adresy
- velikost stránek je většinou mocnina 2 (typicky 4 KB)
- jednoduchý přesun na disk



Obrázek 14: Základní model stránkování. [SGG05]

Adresář stránek

- uvažujme velikost stránky 4 KB a velikost adresního prostoru 32 bitů
- pak velikost adresní 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 4KB 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


Second-level
page tables
Page
table for
the top
4M of
memory
Top-level
page table
1023
6
5
4
3
2
1
0
Bits 10 10 12
PT1
PT2
Offset
(a)
1023
6
5
4
3
2
1
0
To
pages

TLB: Translation Lookaside Buffer

- cache procesoru obsahující hodně používané části stránkovacích 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, byly na jednom místě (lokální proměnné na zásobníku)

Segmentace

- paměť je rozdělena na několik segmentů (např. kód, data, zásobník)
- speciální případ přídělování souvislých bloků paměti
- stránkování je obvykle pro programátora 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 záčatek a délku segmentu + oprávnění

Sdílená paměť

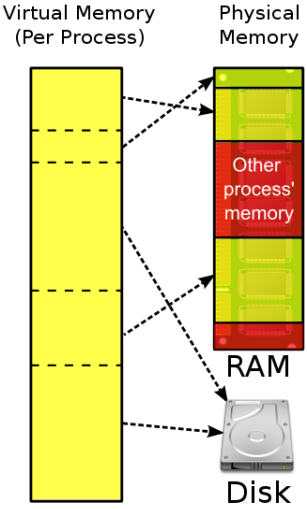
- může být užitečné sdílet paměť (komunikace, úspora místa)
- dva stejné programy/knihovny v paměti
- **stránky více procesů** jsou navázány na **jeden rámec**

Copy-on-Write (COW)

- speciální případ
- daná stránka 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
- virtuální paměť umožňuje další optimalizace (mapování souborů do paměti, etc.)

Virtuální paměť: motivace

- pameť RAM je relativně drahá $\Longrightarrow$ nemusí vždy dostačovat
- aktuálně používaná data (např. instrukce) musí být v RAM, nepoužívaná data nemusí (velké programy $\Longrightarrow$ nepoužívané funkce)
- 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
- k efektivní implementaci je potřeba spolupráce HW (MMU) a OS
- z pohledu aplikace musí být přístup k paměti transparentní
- virtuální paměť (VM) je součástí soudobých OS (swapování)
  - Windows NT – stránkovací soubor (pagefile.sys)
  - Linux – swap partition (ale může být i soubor)
- bezpečnost dat v sekundární paměti? (např. po vypnutí počítače)



Zdroj: Wikipedia.org

Přístup do paměti (1/2)

Předpoklad

- systém si eviduje, které stránky jsou v primární paměti, a které ne
- stránkovací tabulka (MMU)

Postup

- přístup na stránku, která **má** přiřazený rámec v primární paměti $\Longrightarrow$ 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

Přístup do paměti (2/2)

Dílčí komplikace

- pokud je odsouvaná stránka sdílená (např. CoW), je potřeba aktualizovat všechny tabulky, kde se vyskytuje
- potřeba efektivně převádět rámce na stránky $\Longrightarrow$ invertovaná tabulka stránek

Optimalizace inicializace

- můžeme načíst celý proces do primární paměti (může být neefektivní)
- demand paging (stránkování na žádost)
- do paměti se načtou jen data (stránky), která jsou potřeba (případně související $\Longrightarrow$ sekvenční čtení; prefetch)

Vlastnosti stránek

Rezervovaná stránka

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

Komitovaná stránka (Commited)

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

Další vlastnosti

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

Výměna stránek

- 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 možne zamknout, aby nebyly odswapovány (nutné pro jádro, rámce sdílené s HW)

Výběr oběti (1/3)

- hledáme stránku, která nebude v budoucnu použitá (případně v co nejvzdálenější 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. inicilizace)

Most Frequently Used (MFU)

- právě načtené stránky mají malý počet přístupů

Least Recently Used (LRU)

- jako obět 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 navrchol
- 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 několik přístupových bitů
  - nastavuje se nejvyšší bit
  - jednou za čas se bity posunou doprava
  - přehled o používání stránky $\Longrightarrow$ bity jako neznaménkové číslo $\Longrightarrow$ nejmenší = oběť

Výběr oběti (3/3)

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)

Problémy stránkování

Minimální počet rámců

- každý proces potřebuje určité množství rámců (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)

Velikost stránek

- 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 $\Longrightarrow$ časté cache miss
- při přesunu do swapu může být velká stránka výhodnější (přístupová doba)
- některé systémy umožňují používat různé velikosti
- Windows NT ≤ 5.1 & Solaris velké stránky pro jádro malé pro uživatelský prostor
- Windows Vista a novejší (podporuje, ale je nutné explicitně povolit)
- Linux – velké stránky 2.6.38

Mapování souborů a I/O do paměti

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 $\Longrightarrow$ sdílená pamět (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

Typy adres na procesorech i386

- kombinují segmentaci i stránkování
- logická adresa – používá ji aplikace; 48 bitů (16 selektor segmentu; 32 offset); segment je často implicitní
- 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. pro SWAP)

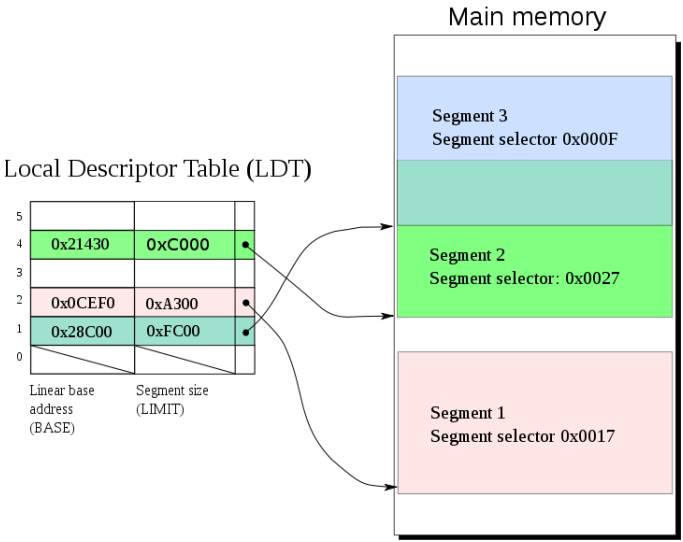
PAE: Physical Address Extenstion

- umožňuje rozšířit využitelnou paměť RAM z 4GB na 64GB (Pentium Pro a novější)
- přesměruje část adresního prostoru do jiné části fyzické paměti
- změna formátu segmentových deskriptorů
- změna na úrovni OS (případně ovladačů);
- bez úprav (AWE) jednotlivé procesy stále omezeny na 4GB
- přidává podporu pro NX bit

I386: Segmentace (1/2)

- 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

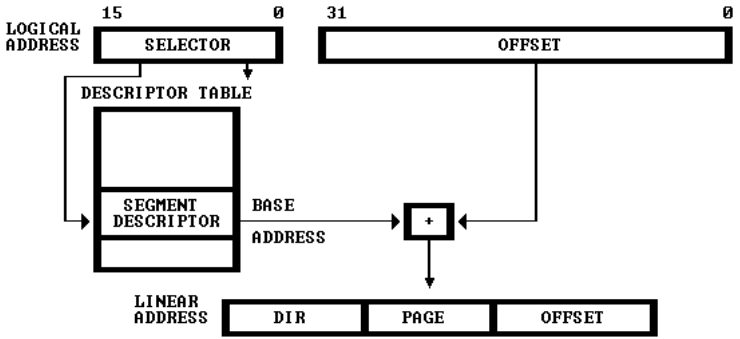
I386: Segmentace (2/2)



I386: Překlad adres I. (segmentace)

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

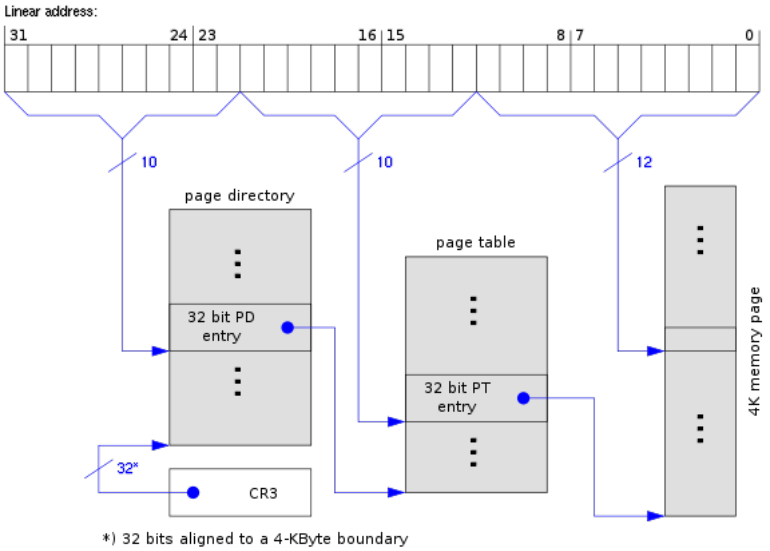
Figure 5-2. Segment Translation



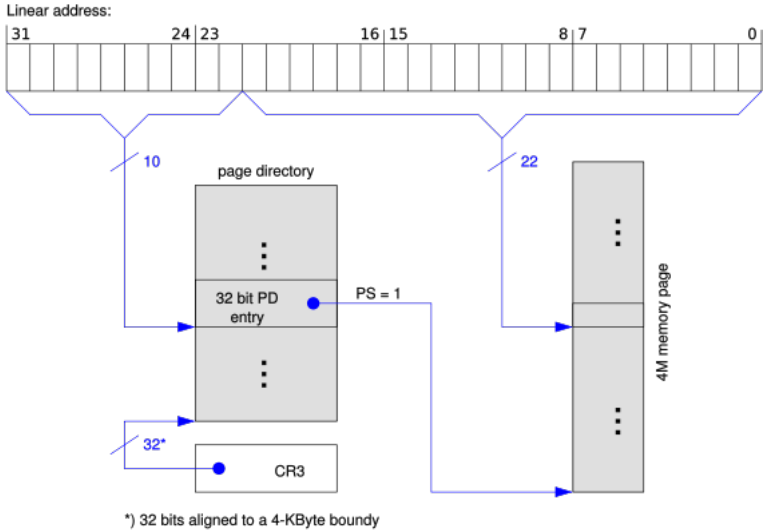
I386: Překlad adres II. (stránkování)

- linární adresa $\Longrightarrow$ fyzická adresa
- standardní stránka/rámec: 4 KB
- hierarchická struktura
  - adresář stránkovacích tabulek (Page Directory)
  - adresář stránek (Page Tables)
  - offset
- adresáře mají velikost jedné stránky, každá položka 4B $\Longrightarrow$ 1024 záznamů
- lineární adresa rozdělana 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 20b), lze obejít přepočet přes PT a používat stránky velikosti 4MB (zbytek adresy je offset)
- velikost stránek lze kombinovat

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



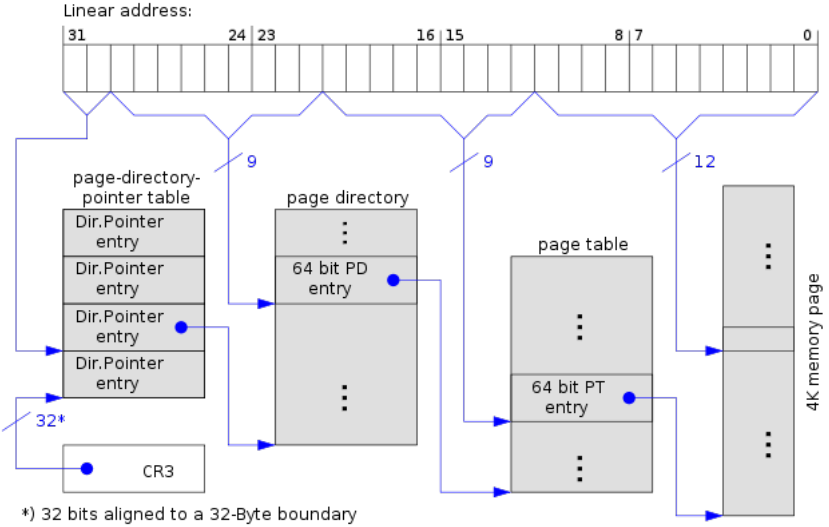
I386: Překlad adres IV. (stránkování – 4 MB stránka)



I386: Překlad adres IV. (PAE)

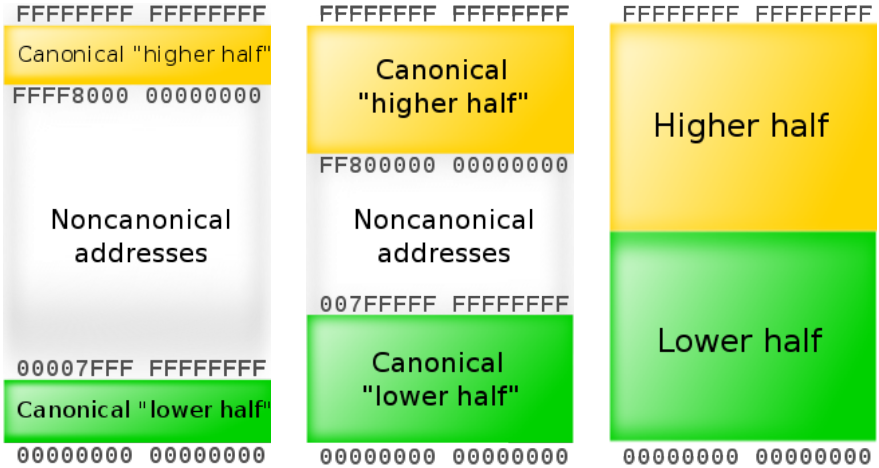
- od Pentium Pro
- každá tabulka 4KB, ale velikost záznamu 8B $\Longrightarrow$ 512 záznamů
- stránkování trojúrovňové
- adresa rozdělena na $2 + 9 + 9 + 12$ bitů
  - 2b – ukazatel na adresář tabulek stránek (Page Directory Pointer Index)
  - 9b – ukazatel v adresáři tabulek stránek
  - 9b – ukazatel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

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



AMD64: Typy adres

- 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 $\Longrightarrow$ kanonické adresy
- nejvyšší platný bit je okopírován do vyšších bitů
- dělí paměť na tři bloky



Zdroj: Wikipedia.org

AMD64: Stránkování

- zavedena čtyřúrovňová hierarchie
- záznamy v tabulkách stránek mají 8 B
- při 4KB stránkách $\Longrightarrow 4 \times 9 + 12 = 48$ adresovatelných bitů
- pro 2MB stránky vynechaná jednoúroveň, možnost použít 4 rezervované bity $\Longrightarrow$ 52 bitů
- nepoužité bity: nejvyšší – NX bit, ostatní k dispozici OS

x86/AMD64: Ochrana paměti I. (segmenty)

- ochrana paměti na úrovni segmentů je zaplá a nejde vypnout (ale jde nastavit, aby nebyla účinná)
- prováděné kontroly
  - kontrola typu segmentu (některé segmenty nebo segmentové registry můžou 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)

x86/AMD64: Ochrana paměti II. (stránkování)

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

Samostudium

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