EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Operační systémy

Správa paměti

Strategický projekt UTB ve Zlíně, reg. č. CZ.02.2.69/0.0/0.0/16_015/0002204

Martin Sysel
Fakulta aplikované informatiky
Univerzita Tomáše Bati ve Zlíně

Základní předpoklady

- Proces může běžet pouze v případě, že má přidělenu operační paměť.
  - V průběhu životního cyklu procesu musí proces dostat alespoň částečně na malou chvíli přidělenou část operační paměti.
- Vnitřní paměť uchovává data a programy pro právě běžící, resp. pro připravené procesy
  - Roli dlouhodobé paměti programů a dat plní vnější paměť
  - Požadavky, které se v počítačovém systému kladou na paměti se zatím nedají ekonomicky splnit jedinou pamětí.
    - Ideální paměť neexistuje (nekonečně velká, rychlá, perzistentní, levná)
    - Hierarchie pamětí

Hierarchie pamětí

Přístupová doba se snižuje
Přenosová rychlost se zvyšuje

Stoupá cena za bit
Kapacita se snižuje

CPU registry

Vyrovnávací paměť (Cache)

Operační paměť (RAM)

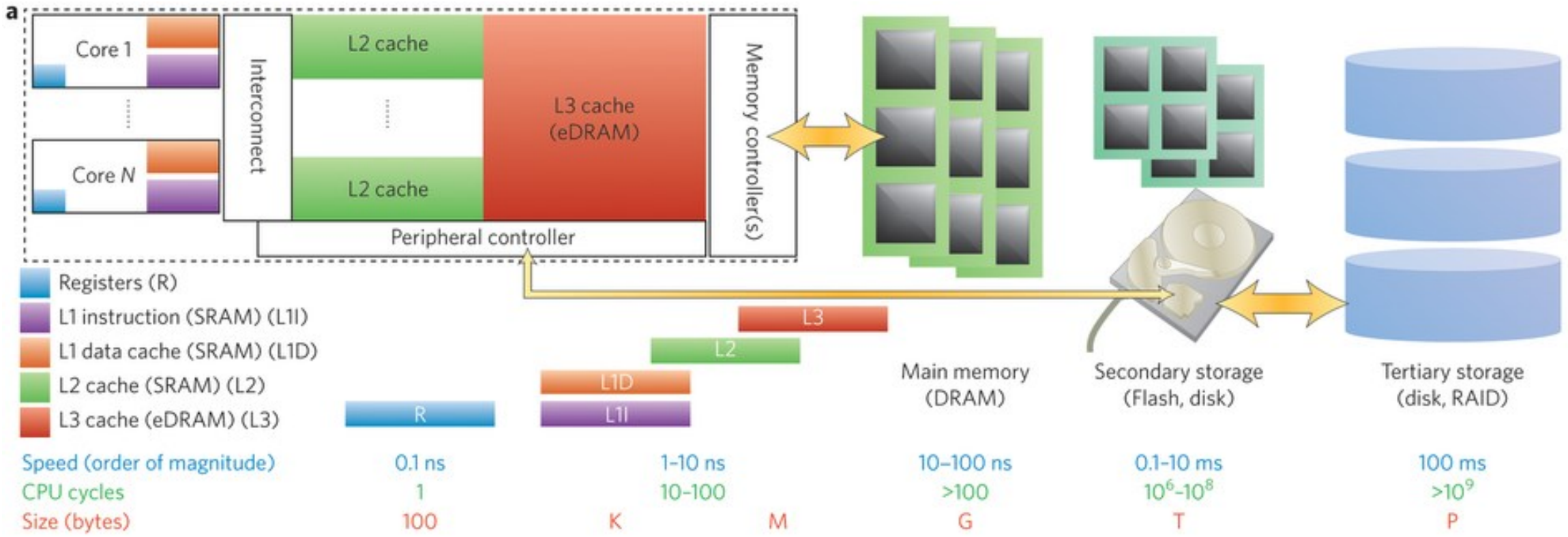
Sekundární úložiště (HDD, SSD)

Terciální úložiště (pásky, NAS, aj.)

CPU přistupuje napřímo

Data musí být nejprve přemístěna do hlavní (operační) paměti

Hierarchie pamětí



(zdroj: https://www.nature.com/articles/nnano.2015.29)

Základní předpoklady

- Správa paměti je nutně předmětem činnosti OS
  - Nelze ji svěřit aplikačnímu programování
    - Bylo by to velmi neefektivní a nebezpečné
  - Aplikační procesy nemají přístup k prostředkům pro řízení paměti
    - Privilegované akce

- Každý operační systém obsahuje správce paměti
  - Vyjma velmi malých systémů
    - Staré počítače, jednoduché řídící počítače (embedded), levné programovatelné mikrokontrolery

Strategie správy paměti

- Co je nutno vzít v úvahu
  - Kolik paměti bude mít každý proces přístupné?
  - Kde v paměti bude proces umístěn?
  - Který proces zůstane v paměti?

(Deitel, Deitel & Choffness, 2004)

Adresní prostor

- Fyzický adresní prostor (FAP)
  - Skutečná fyzická paměť počítače (RAM)
- Logický adresní prostor (LAP)
  - Logická adresa (případně virtuální) se kterou pracuje CPU
  - Její obsah je uložen ve FAP
- Rozsah je dán architekturou počítače a CPU

(Silberschatz, Galvin & Gagne, 2013)

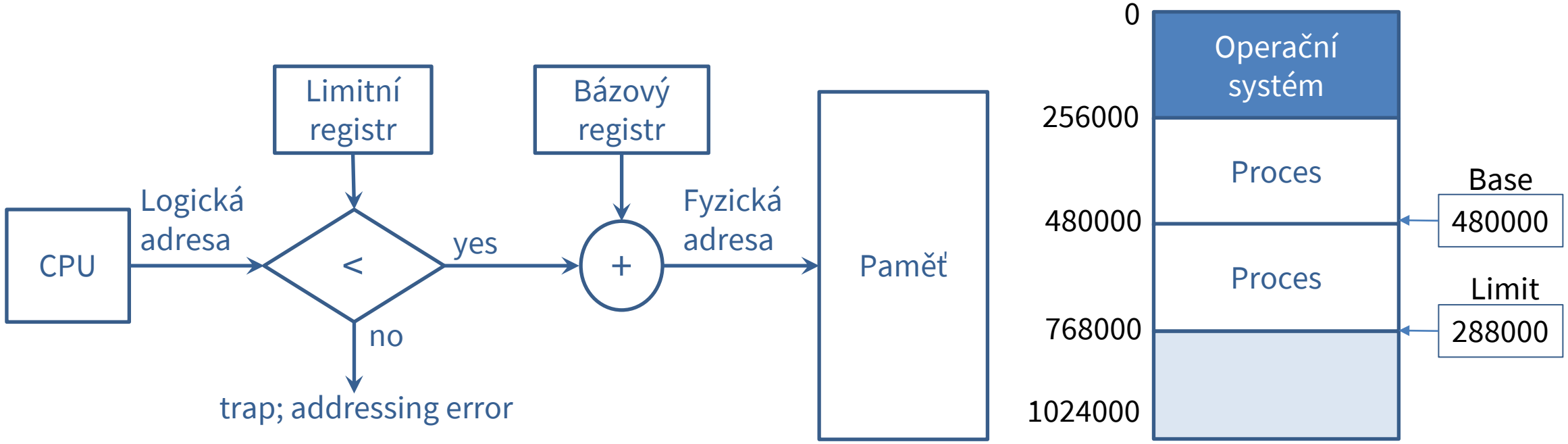
Vázání adres (*Address Binding*)

- Při překladu
  - Je-li umístění ve FAP známé před překladem, kompilátor může generovat absolutní kód
    - Jen v uzavřených a vestavěných systémech
    - Při přemístění je nutný nový překlad
- Při zavádění programu
  - Použitelné, je-li umístění ve FAP známé při sestavování nebo při zavádění programu
- Za běhu
  - proces může měnit své umístění v paměti
  - Koncept logického a fyzického adresního prostoru (LAP a FAP)
    - Cílovým prostorem je LAP. Program se zavádí do FAP ve tvaru pro LAP.
  - Potřeba hardwarové podpory pro mapy adres (MMU) (např. bázové a limitní registry)
  - Ochrana paměti, možnost sdílení paměti.
  - Dochází k dynamickému překladu adres DAT – *Dynamic Address Translation*

(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013; Tanenbaum, 2015)

Bázové a limitní registry

- CPU musí kontrolovat každý přístup do paměti generovaný v uživatelském režimu, aby zajistil ochranu paměti



(Silberschatz, Galvin & Gagne, 2013)

Memory-Management Unit (MMU)

- Hardwarové zařízení, které za běhu mapuje logickou adresu na fyzickou.
  - Obecně je součástí CPU.
  - Není vyžadována žádná zvláštní podpora z operačního systému

- Uživatelský proces pracuje s logickou adresou
  - Nikdy nevidí skutečné fyzické adresy
  - K vázání adresy dochází za běhu, když se odkazuje na umístění v paměti
    - Logická adresa je vázaná na fyzické adresy

(Silberschatz, Galvin & Gagne, 2013)

Dynamické načítání a linkování

- Při dynamickém načítání není rutina načtena, dokud není volána.
  - Všechny rutiny jsou uloženy na disku.
  - Rutina je načtena pouze v případě potřeby.
  - Dříve bylo nutné, aby celý program a všechna data procesu byla ve fyzické paměti, aby se proces mohl spustit.
- Dynamicky linkované knihovny se načítají za běhu.
  - Sdílené knihovny
- Staticky linkované knihovny – jsou zahrnuty přímo do spustitelného binárního souboru na pevně danou adresu
  - Např. systémové knihovny.

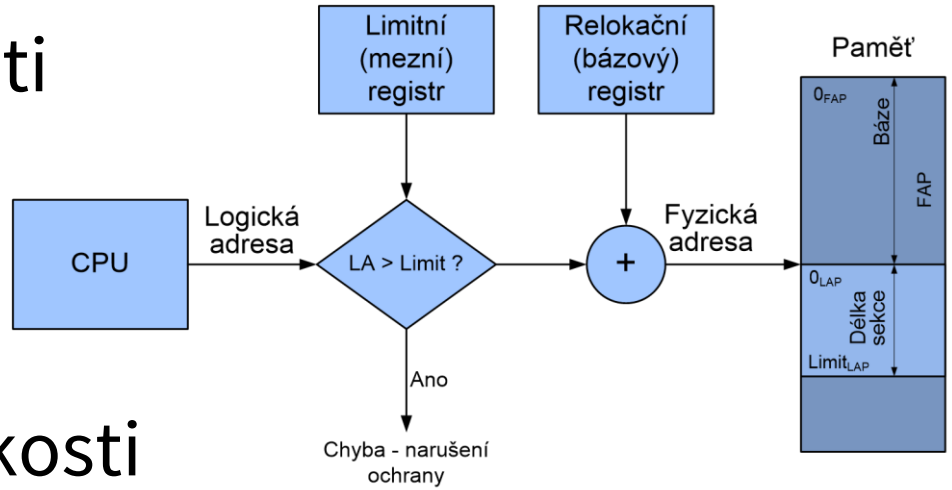
Spojité a nespojité adresní prostory

- Způsoby organizace programů v paměti (FAP)
  - Spojité přidělení
    - Proces musí existovat jako jeden blok souvislých adres
    - Někdy není možné najít dostatečně velký blok
    - Nízká režie
  - Nespojité přidělení
    - Program se dělí na části zvané segmenty
    - Každý segment lze umístit do jiné části paměti
    - Je snadnější najít „díry“, do kterých se segment vejde
    - Zvýšený počet procesů, které mohou současně existovat v paměti, kompenzuje režii
    - Může eliminovat externí fragmentaci a zlepšit využití úložiště
    - Rozděluje paměť procesu na různé bloky a přiděluje bloky v různých částech fyzické paměti.
      - Segmentace, stránkování

(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013)

Blokové přidělování paměti (Spojité)

- Nejjednodušší mechanismy správy paměti



(Lažanský, 2014)

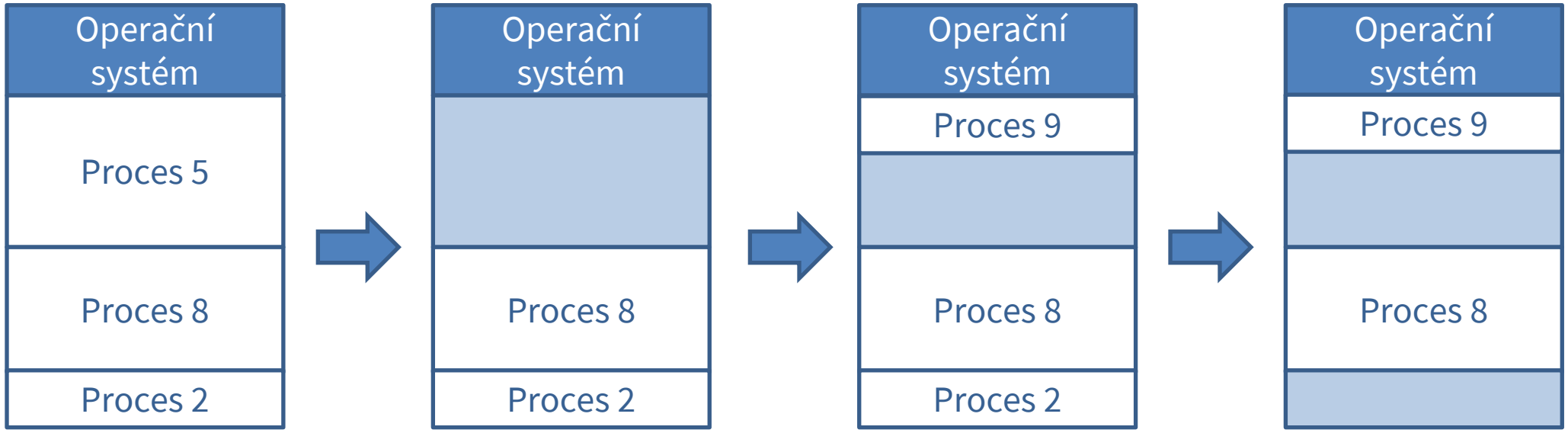
- Přidělování veškeré volné paměti
- Přidělování pevných bloků paměti
- Přidělování bloků paměti proměnné velikosti

- Operační paměť se dělí do dvou typů sekcí
  - sekce pro rezidentní OS, obvykle na počátku FAP
  - sekce pro uživatelský proces(y)
  - Přístup a ochrana paměti pomocí limitního (mezního) a relokačního (bázového) registru.

(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013; Lažanský, 2014)

Alokace adresního prostoru

- Proces předpokládá, že jeho paměťový prostor (LAP) je souvislý.
  - Díra - blok dostupné paměti; V paměti jsou díry různé velikosti.
  - Operační systém udržuje informace o přidělených blocích a dírách.

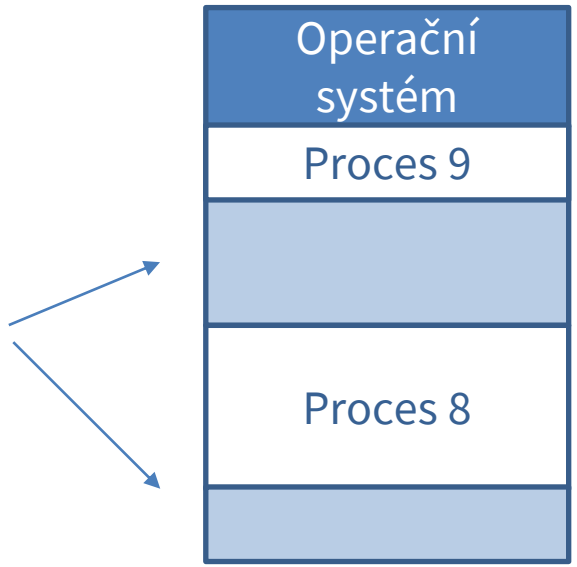


(Silberschatz, Galvin & Gagne, 2013)

Fragmentace

- Externí fragmentace
  - Celkový volný paměťový prostor není souvislý
  - Zhutnění
- Redukce externí fragmentace pomocí setřásání
  - Použitelné *pouze* při dynamické relokaci
  - Přesouvají se obsahy úseků paměti s cílem vytvořit velkou souvislou díru
  - Významná režie
  - Blitter

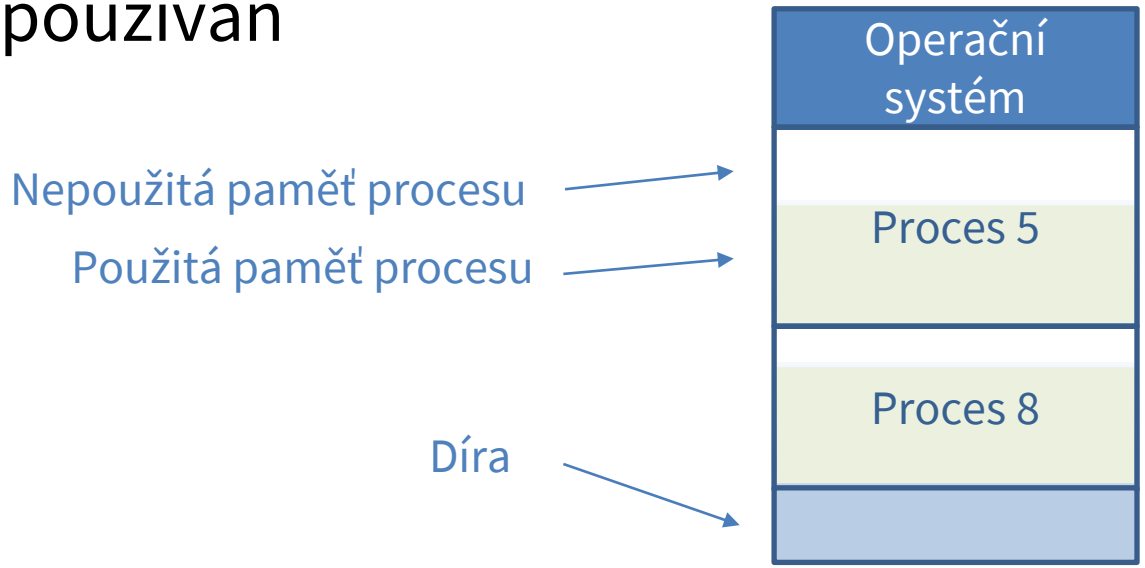
Fragmentovaná paměť



(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013)

Fragmentace

- Vnitřní fragmentace
  - Přidělená paměť může být o něco větší než požadovaná paměť
    - Mocniny při základu 2
  - Tento rozdíl velikosti není používán



(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013)

Strategie přidělování paměti

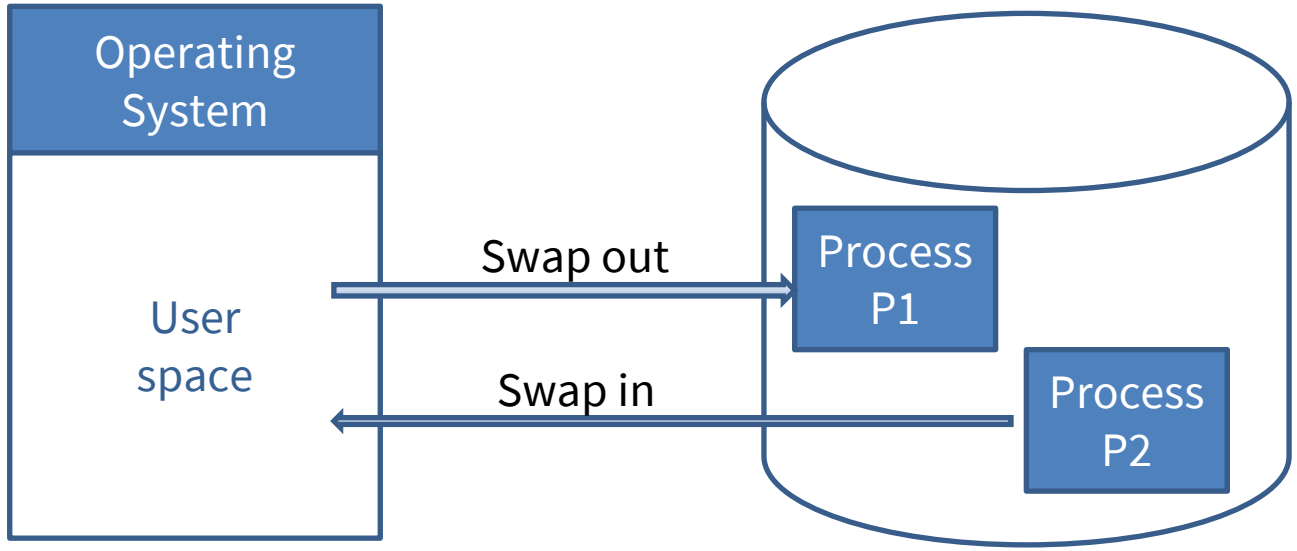
- Požadavek na určitou velikost paměti; vybírá a přiděluje se ze seznamu děr
- First-fit
  - Použije první dostatečně velkou díru
- Best-fit
  - Najde nejvhodnější díru, jejíž velikost nejvíce odpovídá požadavku.
  - Musí prohledat celý seznam
- Worst-fit
  - Použije největší díru
  - Musí také prohledat celý seznam
  - Předpoklad, že zbylá díra bude dostatečně velká pro jiný proces.

- First-fit a Best-fit jsou lepší než Worst-fit z hlediska rychlosti a využití úložiště

(Silberschatz, Galvin & Gagne, 2013)

Swapování

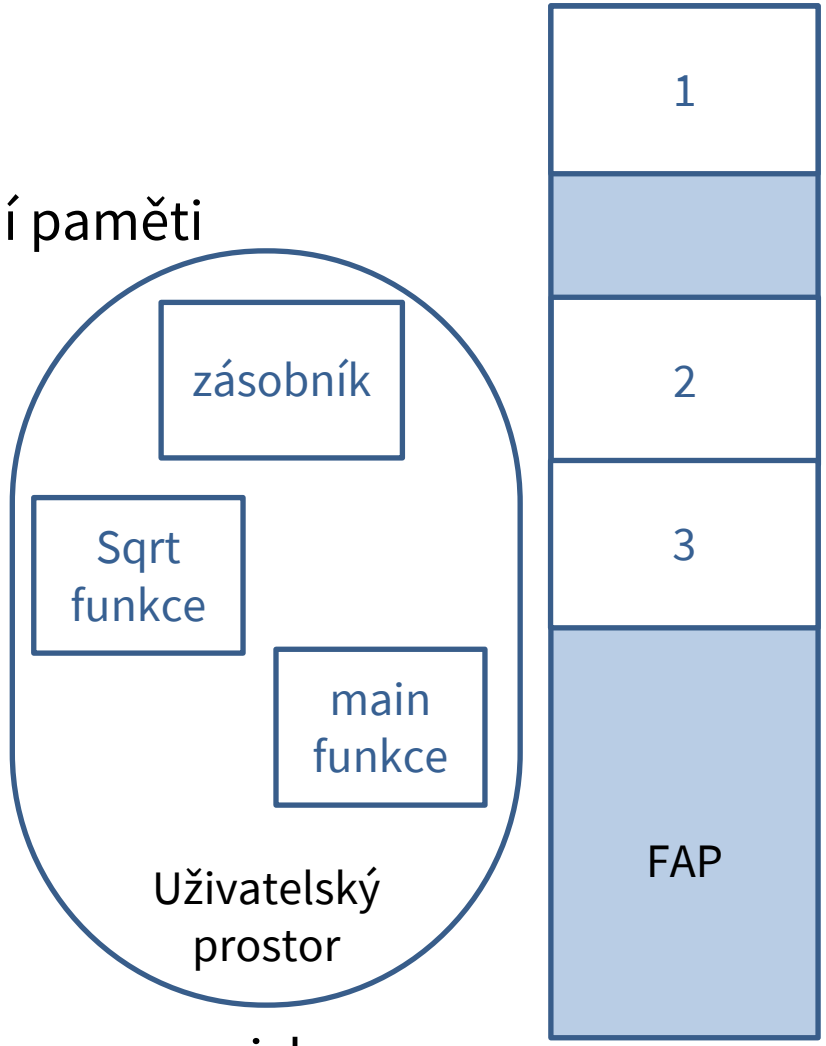
- Proces lze dočasně odložit z paměti do úložiště a poté přenést zpět do paměti.
  - Odkládá se celý proces
- Celkový prostor paměti procesů může přesáhnout fyzickou paměť.
- Různé modifikace myšlenky odkládání lze nalézt na mnoha systémech
  - např. Linux a Windows



(Silberschatz, Galvin & Gagne, 2013)

Segmentace

- Schéma správy paměti, která podporuje uživatelské zobrazení paměti
  - Dvourozměrný pohled. Používá logické adresy.
  - Řeší vnitřní fragmentaci. Externí fragmentace zůstává.
- Programy je rozdělen do různých částí
  - Segment je logická jednotka, jako je kód, data, halda, zásobník:
    - hlavní program
    - funkce
    - metody
    - objekty
    - lokální proměnné, globální proměnné
    - zásobník
    - tabulka symbolů
    - pole



- Oddělené segmenty, které mohou být ve fyzické paměti uloženy nesouvisle.

(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013)

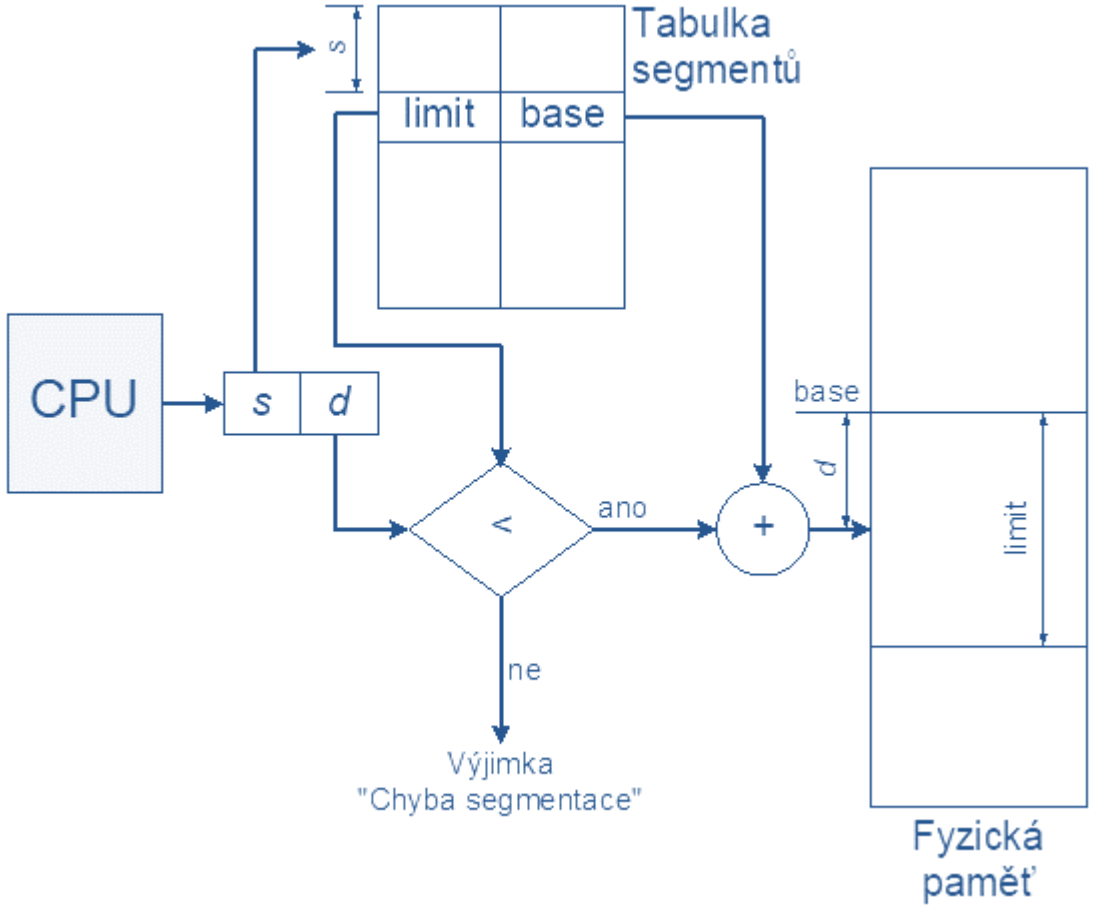
Segmentace

- Logická adresa se skládá ze dvou částí:
  <číslo segmentu, offset>
- Tabulka segmentů - mapuje dvourozměrné fyzické adresy do jednorozměrné fyzické adresy
  - každá položka tabulky má:
    - base - obsahuje počáteční fyzickou adresu, kde jsou segmenty uloženy v paměti
    - limit - určuje délku segmentu
- Segment-table base register (STBR)
  - ukazuje na umístění tabulky segmentů v paměti
- Segment-table length register (STLR)
  - označuje počet segmentů použitých programem
  - číslo segmentu je v pořádku, pokud s <STLR

Segmentace

- Ochrana
  - Každá položka v tabulce segmentů má:
    - validační bit = 0 ⇒ nelegální segment
    - read/write/execute oprávnění
- Ochranné bity spojené se segmenty
  - ke sdílení kódu dochází na úrovni segmentů
- Délka segmentů se liší
  - přidělování paměti je problematika přidělování dynamických úložišť
- Detekce přístupu mimo segment
  - výjimka typu „segmentation fault“

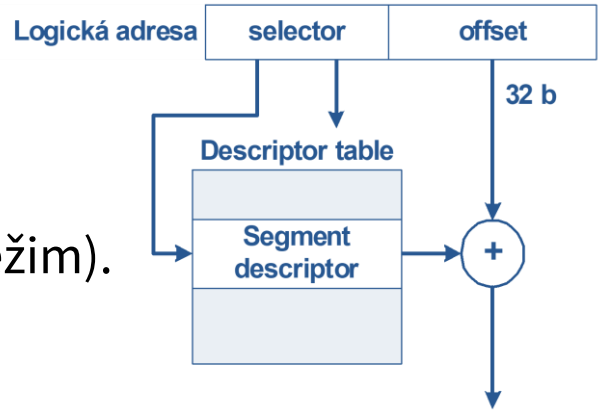
(Silberschatz, Galvin & Gagne, 2013)



Segmentace
(Silberschatz, Galvin & Gagne, 2013)

Segmentace

- 8086 má 16bit datovou sběrnici a registry, ale 20bit adresovou sběrnici.
  - Procesor 8086 má 4 tzv. segmentové registry, Adresa je tvořena adresou segmentu 16 bitů a adresou uvnitř segmentu (offset) 16 bitů. Výsledná fyzická adresa se tvoří podle pravidla (segment << 4)+ offset
- 80286 obsahuje MMU převádí adresu (16 bit selektor, 16 bit offset) na 24 bitů adresu ve fyzické paměti
- 80386 je plně 32 bit
  - 32 bity adresovaný paměťový prostor je 4 GiB
- x86-64 nepoužívá segmentaci v long mode (64bit režim).



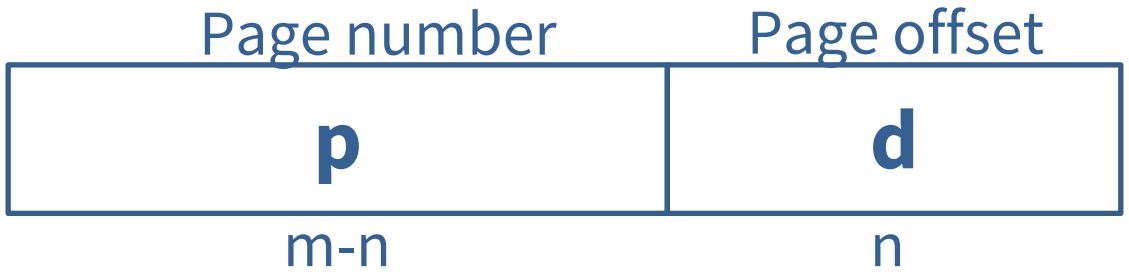
Stránkování (Paging)

- Fyzický adresní prostor procesu může být nesouvislý.
- FAP rozděluje na části pevné velikosti, nazývané rámce (frames).
- LAP rozděluje na části stejné velikosti, nazývané stránky (pages).
  - Velikost je mezi 512 B až 1 GiB, X86-64 podporuje: 4 kiB, 2MiB, 4MiB, 1GiB
- Mechanismus překladu z logické adresy na fyzickou pomocí tabulky stránek (Page Table)
- Řeší externí fragmentaci. Může ale vznikat vnitřní fragmentace.
  - průměrná vnitřní fragmentace = 1/2 velikosti rámce

(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013)

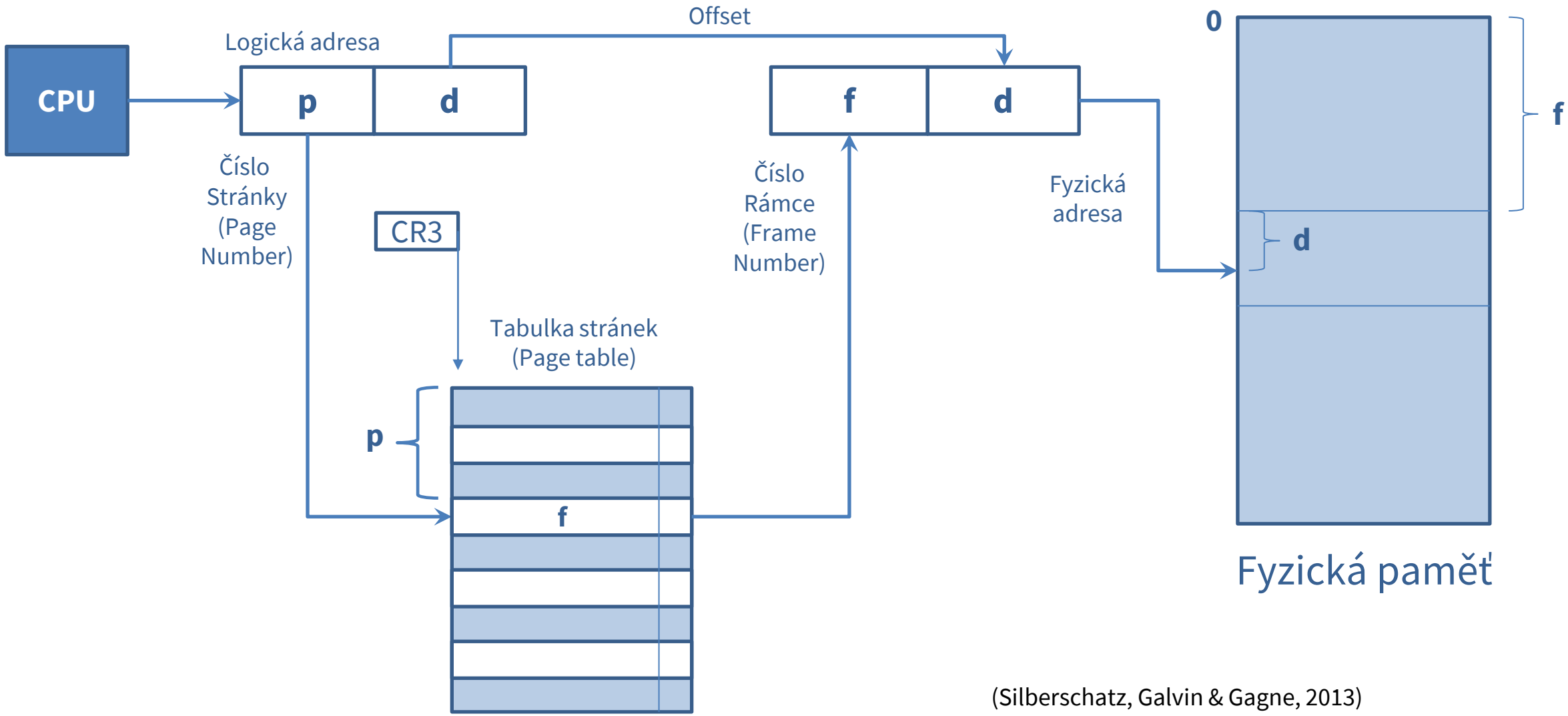
Stránkování

- Adresa generovaná CPU je rozdělena do dvou částí:
- **p** je číslo stránky v logické paměti
  - Používá se jako index do tabulky stránek, která obsahuje základní adresu každé stránky ve fyzické paměti (číslo rámce).
- **d** je posun od začátku stránky **p**, nazývaný posun stránky (***Page offset***)
- Pro daný logický adresní prostor 2^m a velikost stránky 2^n
  - předpokládejme 32bit adresy:
    - 4096 bajtů (12 bitů pro ofset)
    - 20 bitů pro číslo stránky



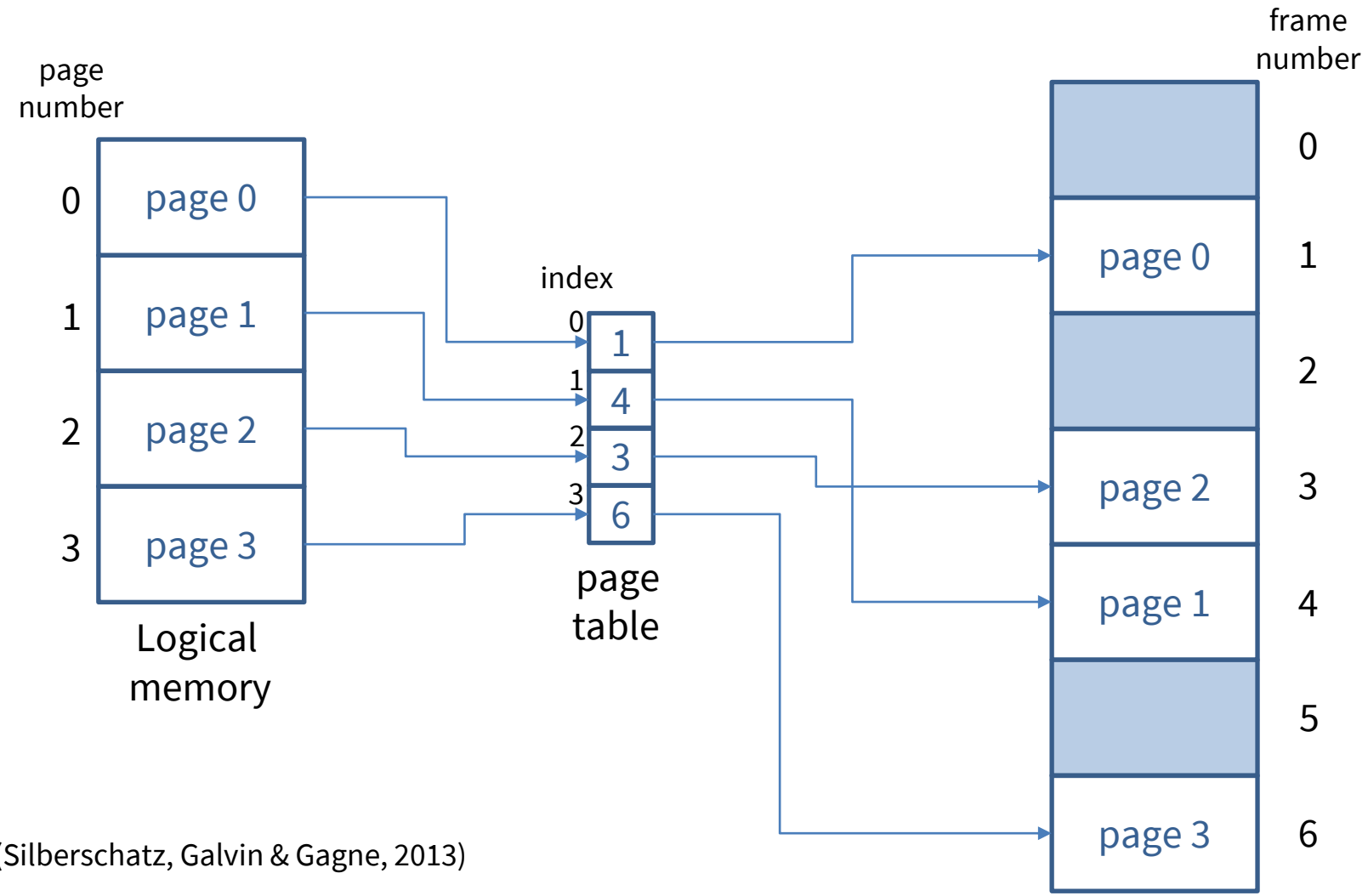
(Silberschatz, Galvin & Gagne, 2013; Tanenbaum, 2015)

Stránkování



(Silberschatz, Galvin & Gagne, 2013)

Příklad stránkování



(Silberschatz, Galvin & Gagne, 2013)

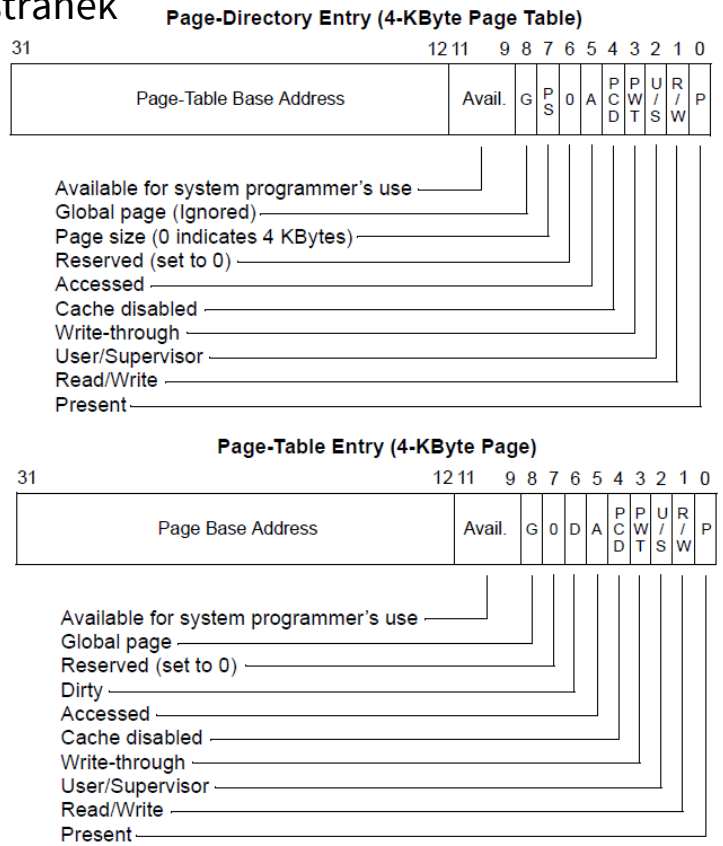
Hardwarová podpora stránkování

- Každý operační systém má své vlastní metody ukládání tabulek stránek.
  - *Obecně existuje tabulka stránek pro každý proces*.
- Tabulka stránek (PT) je uložena v paměti.
  - Page-table base register (PTBR) ukazuje na tabulku stránek
    - Změna tabulek stránek vyžaduje změnu pouze tohoto jednoho registru.
      - rychlejší přepínání kontextu
  - Page-table length register (PTLR) obsahuje velikost tabulky stránek
  - Instrukce k načtení nebo úpravě těchto registrů jsou privilegované.

(Silberschatz, Galvin & Gagne, 2013)

Obsah tabulky stránek

- Položky tabulky stránek:
  - Číslo rámce
    - umístení stránky v reálné pameti pocítace
  - Atributy stránek



(AMD, 2018)

- Atributy stránek
  - Základní príznaky
    - p present – stránka je v pameti, císlo rámce je platné
    - ps page size – velikost stránky
    - g global – stránka je globální, nepatrí jednomu procesu
  - Řízení přístupu
    - r/w read/write – povolení zápisu do stránky
    - u/s user/supervisor – povolení prístupu pro uživatele
  - Optimalizace
    - pwt page-level write through – nastavení cache
    - pcd page-level cache disable – zákaz použití cache
  - Statistika
    - a accessed – stránka použita pro ctení
    - d dirty – stránka použita pro zápis
  - Virtualizacní príznaky
    - v/i = valid/invalid indikuje prítomnost stránky ve FAP
    - a = accessed znací, že stránka byla použita
    - d dirty indikuje, že obsah stánky byl modikován

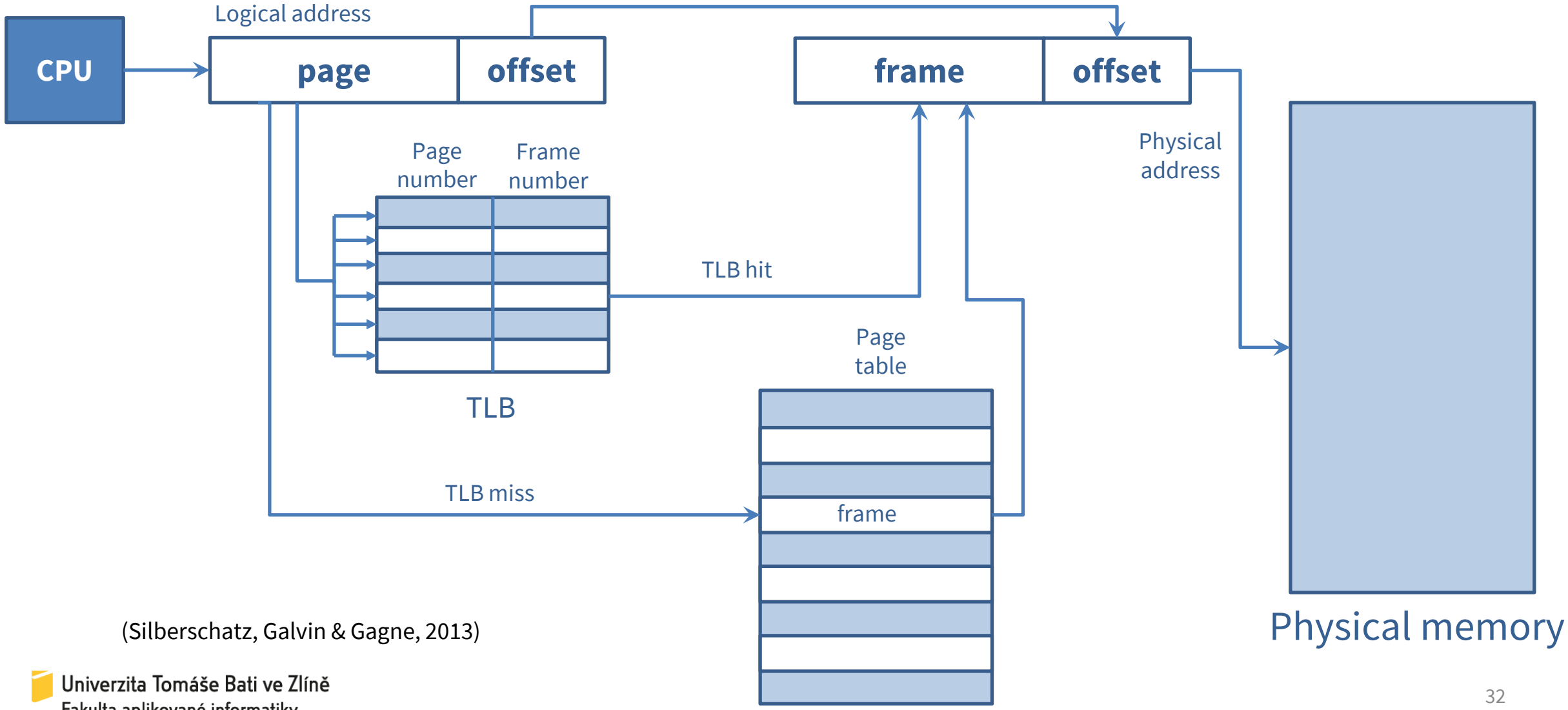
(Štěpán, 2014)

Implementace tabulky stránek

- Tabulka stránek je uložena v hlavní paměti.
  - V tomto schématu vyžaduje pro každý přístup k datům/instrukcím dva přístupy do paměti.
    - Jeden pro tabulku stránek a jeden pro data / instrukce.
- Problém s dvojnásobným přístupem do paměti se řeší hardwarovou mezipamětí
  - Vyrovnávací paměť pro překlad (TLB - Translation Lookaside Buffer)
    - asociativní paměť - speciální rychlá, vyhledává podle obsahu
    - obvykle malá kapacita (8 až 4096 záznamů), více úrovní TLB (existuje i pro cache)
    - Pokud adresa není v paměti TLB (TLB miss) je hodnota načtena do TLB pro příští použití.
      - Pokud překlad adresy nalezen v TLB (TLB hit), tak výrazné zrychlení (Přístup během 1 taktu CPU), vysoká úspěšnost (>99 %)
  - TLB ukládá identifikátory adresního prostoru (ASID) pro každou položku v TLB.
    - ASID jednoznačně identifikuje každý proces a používá se k zajištění ochrany adresového prostoru pro tento proces.
    - Bez podpory ASID musí být TLB vyprázdněna s každým přepnutím kontextu.

(Silberschatz, Galvin & Gagne, 2013)

Hardwarová podpora stránkování s TLB

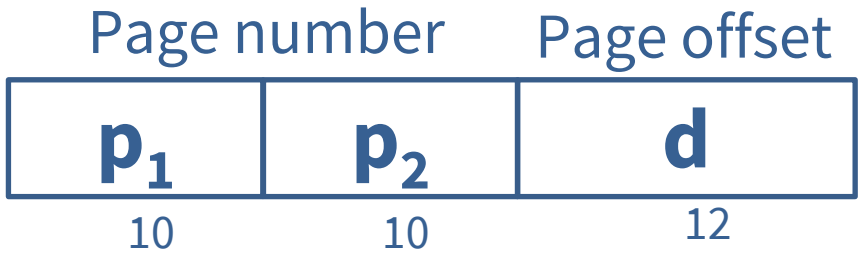


(Silberschatz, Galvin & Gagne, 2013)

Problematika velikosti tabulky stránek

- Jedna tabulka stránek může být velmi velká
  - Předpokládejme 32bitové logické adresy (2^{32} - 4GiB) a 4KiB (2^{12}) stránky
  - Tabulka stránek by měla 1 milion záznamů (2^{32-12})
- Řešení
  - hierarchické stránkování, tabulky s hashovanými stránkami, inverzní tabulka stránek
- Hierarchické tabulky stránek
  - více tabulek stránek, přístupné hierarchicky
    - Každá úroveň obsahuje tabulku, která ukládá ukazatele na tabulky v úrovni níže.
    - Nejspodnější úroveň se sestává z tabulek obsahujících překlady adres.
  - Např. dvouúrovňová tabulka stránek
    - Vnější tabulka se nazývá adresář stránek, který ukazuje na jednotlivé tabulky stránek. (např. IA-32)
    - Logická adresa je rozdělena do tří částí

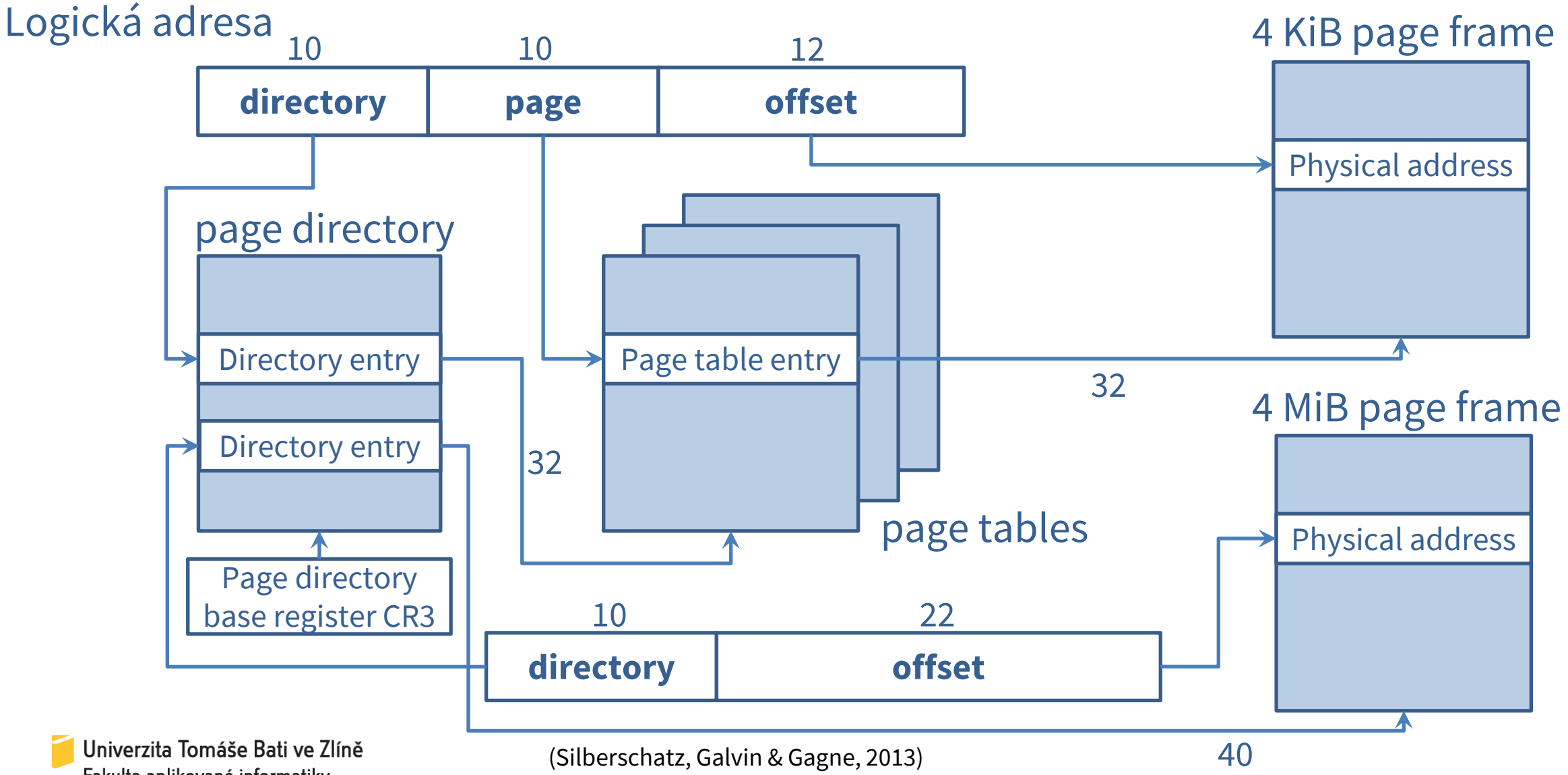
(Silberschatz, Galvin & Gagne, 2013)



IA-32 stránkování (32 bitů)

- Stránkování používá 32bitový logický adresní prostor
  - vytvoří buď 52bitovou nebo 36bitovou fyzickou adresu
  - adresování 4 GiB nebo až 4 PiB paměti.
- 52bitové adresování je povoleno mechanismem fyzického rozšíření adres (PAE - Physical Address Extension).
  - Proces však může kdykoli získat přístup pouze k 4 GB adresnímu prostoru.
- 36bitový adresní prostor je povolen prostřednictvím rozšíření velikosti stránky (PSE-36 - Page Size Extension).
- Architektura stránkování podporuje 4 KiB stránky nebo 4 MiB.

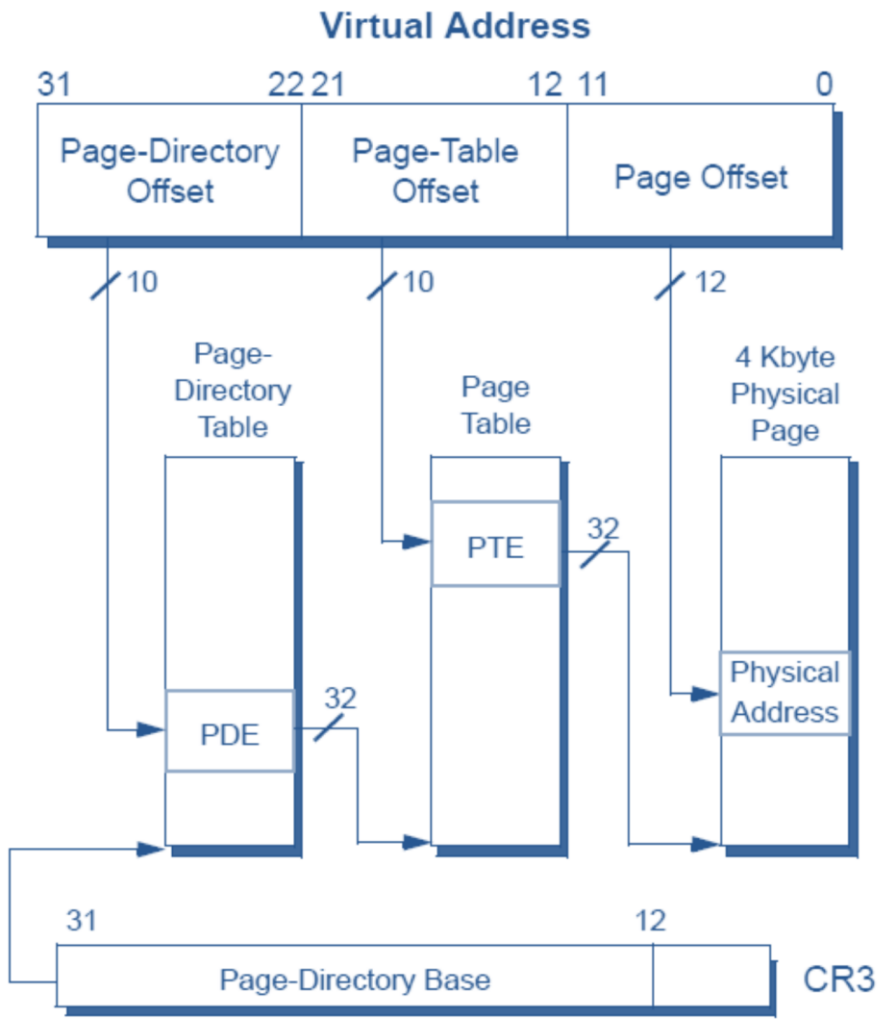
IA-32 stránkování (32 bitů)



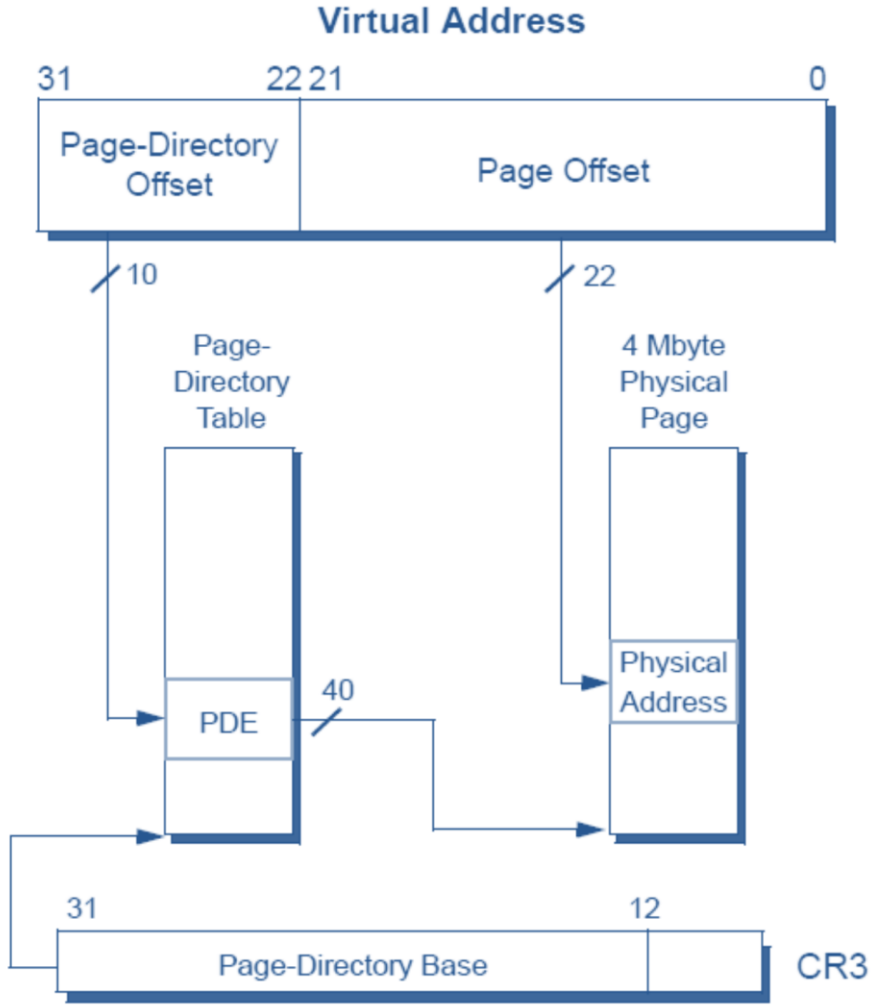
(Silberschatz, Galvin & Gagne, 2013)

IA-32 stránkování (32 bitů) (Non PAE)

pouze rozdělený předchozí slide



4-Kbyte Non-PAE Page Translation
Legacy Mode (AMD, 2018)

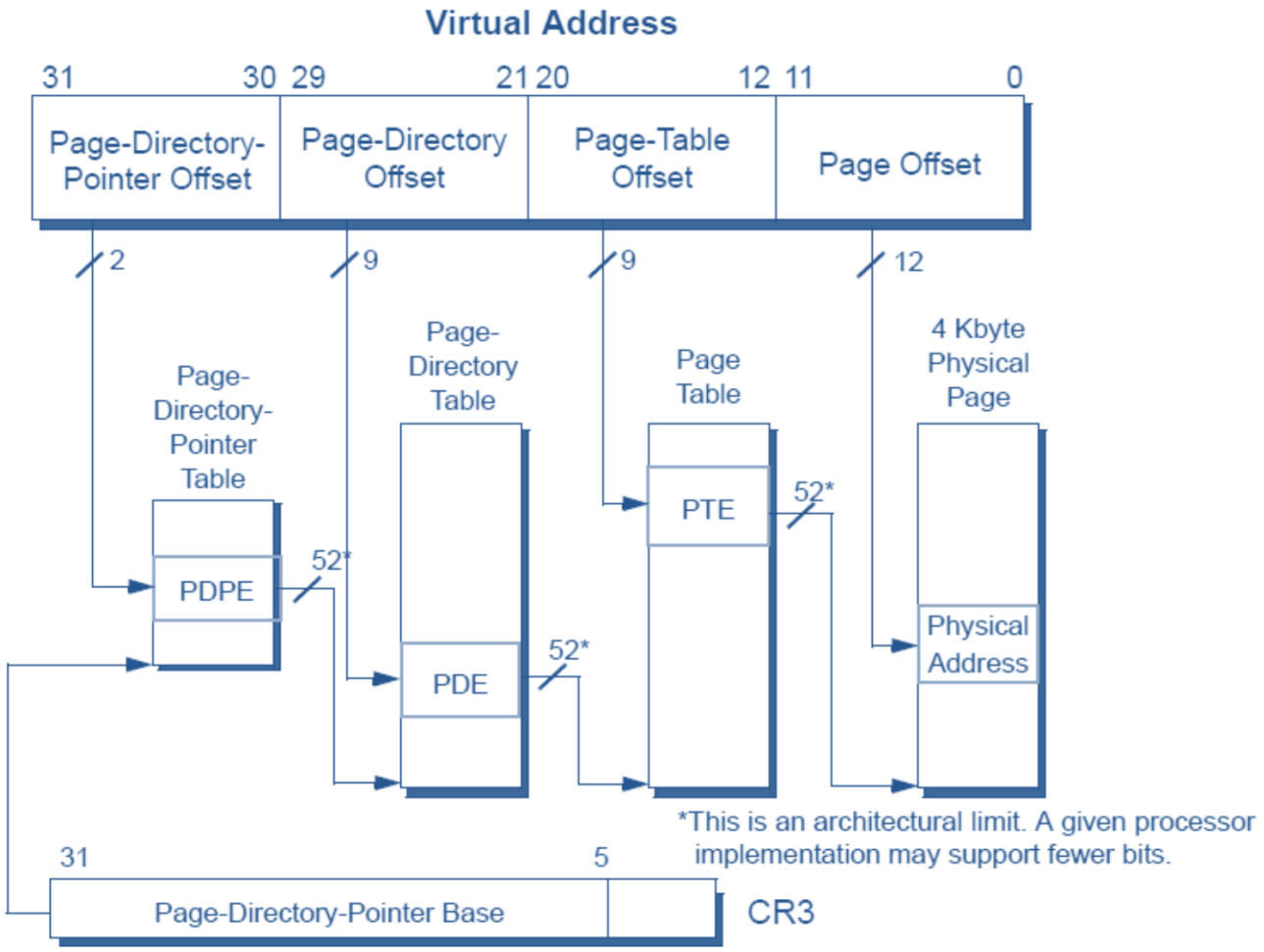


4-Mbyte Page Translation
Non-PAE Paging Legacy-Mode
(AMD, 2018)

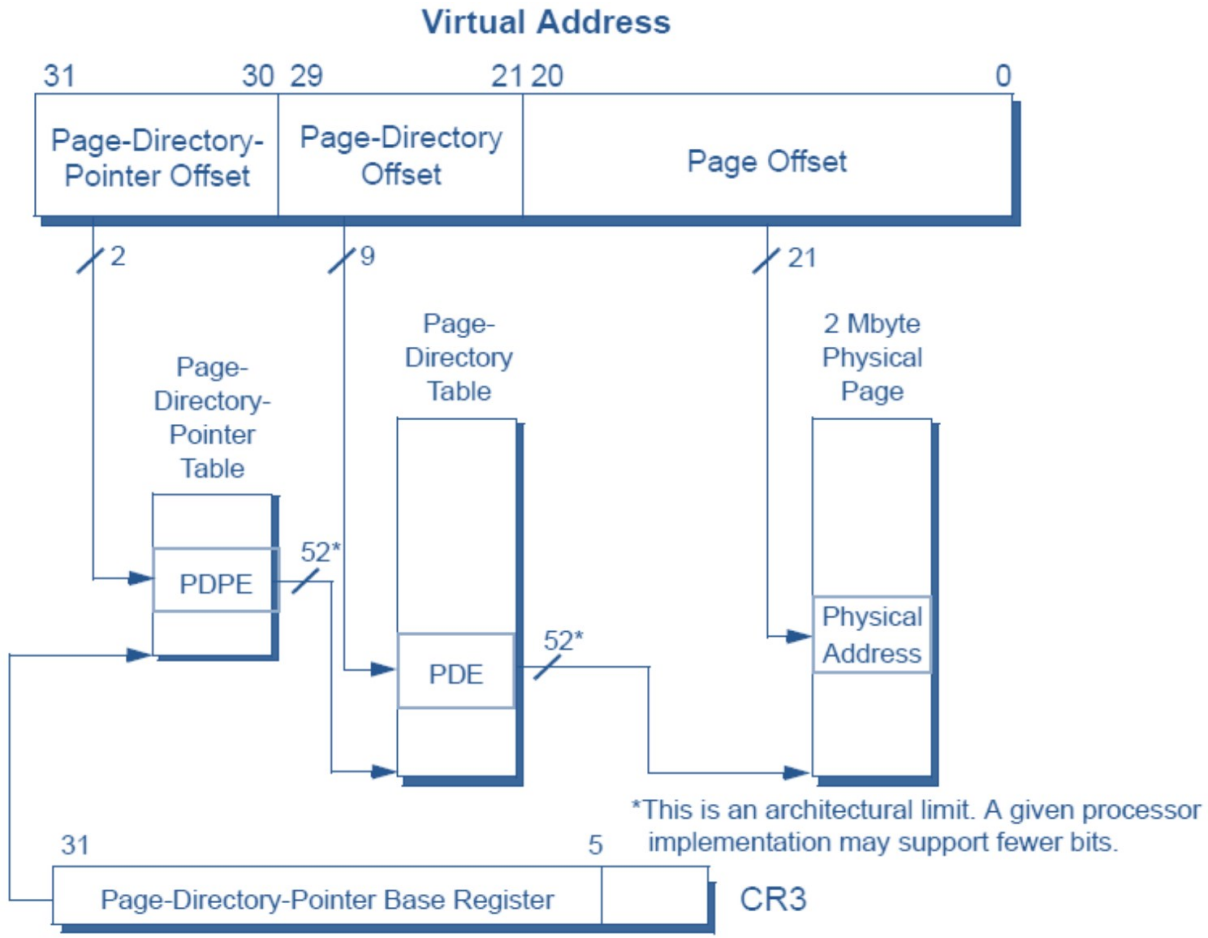
IA-32 stránkování s PAE

- Využívá 32bitové virtuální adresy a generuje až 52bitové fyzické adresy pomocí stránek 4 KiB nebo 2 MiB.
- U 4 KiB stránek MMU používá tříúrovňovou hierarchii tabulky stránek, která se používá k mapování 32bitové virtuální adresy na 52bitovou fyzickou adresu.
  - Horní tabulka je tabulka ukazatelů adresáře stránky se čtyřmi vstupy (PDPT), která je indexována podle horních dvou bitů virtuální adresy.
  - Záznam v PDPT ukazuje na adresář dílčí stránky, který je indexován 9 bitů (512 záznamů na adresář dílčí stránky).
  - Adresář stránky odkazuje na částečnou tabulku stránek, která je indexována podle dalších 9 bitů (také 512 záznamů na dílčí tabulku stránek).
    - PTE (Page Table Entry) obsahuje prvních 40 bitů fyzické adresy.
  - Dolních 12 bitů je získáno přímo z lineární adresy a jsou použity pro 4 KiB stránku.

IA-32 stránkování s PAE



4-Kbyte PAE Page Translation—Legacy Mode (AMD, 2018)

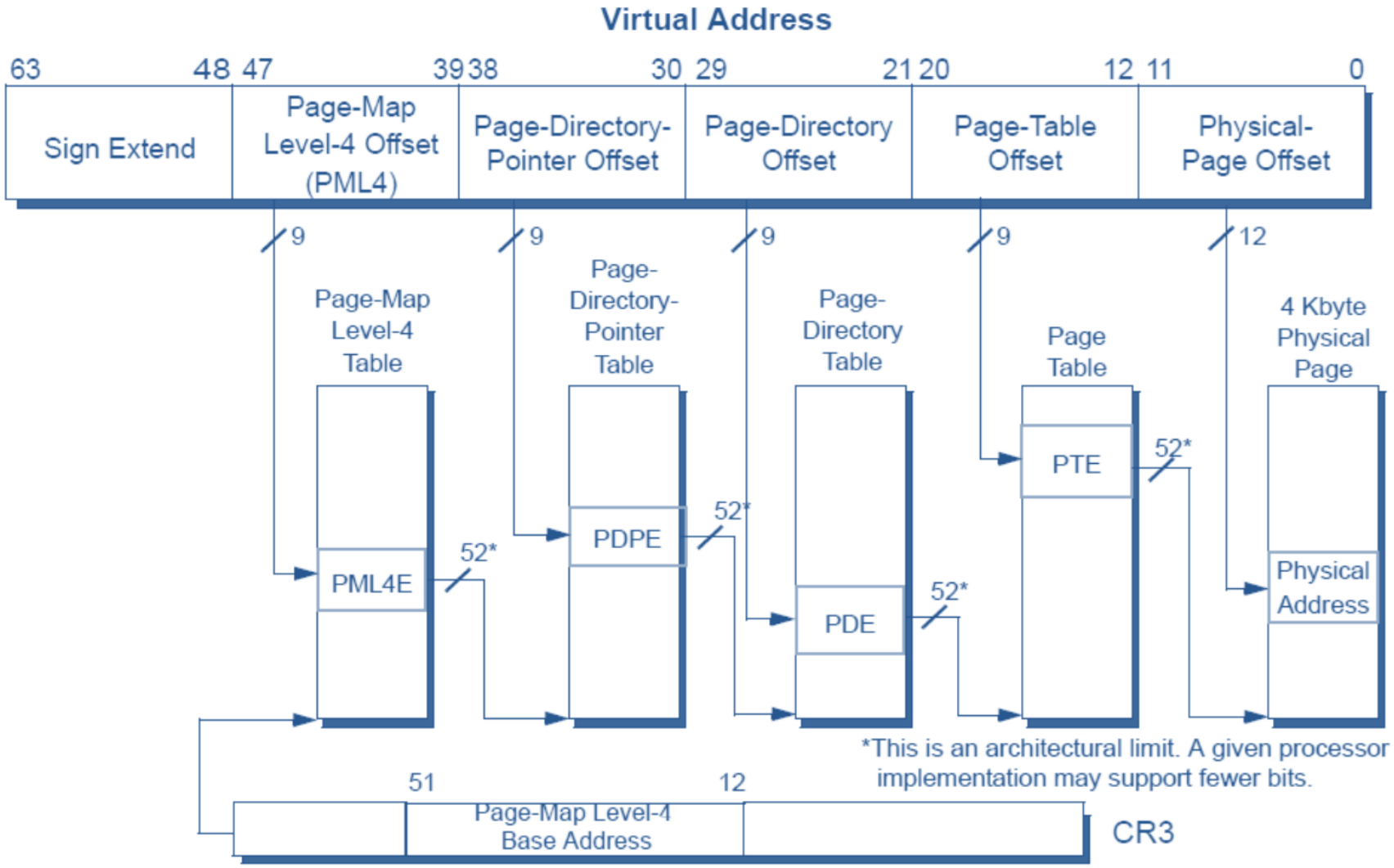


2-Mbyte PAE Page Translation—Legacy Mode (AMD, 2018)

Stránkování v Long mode (64bit)

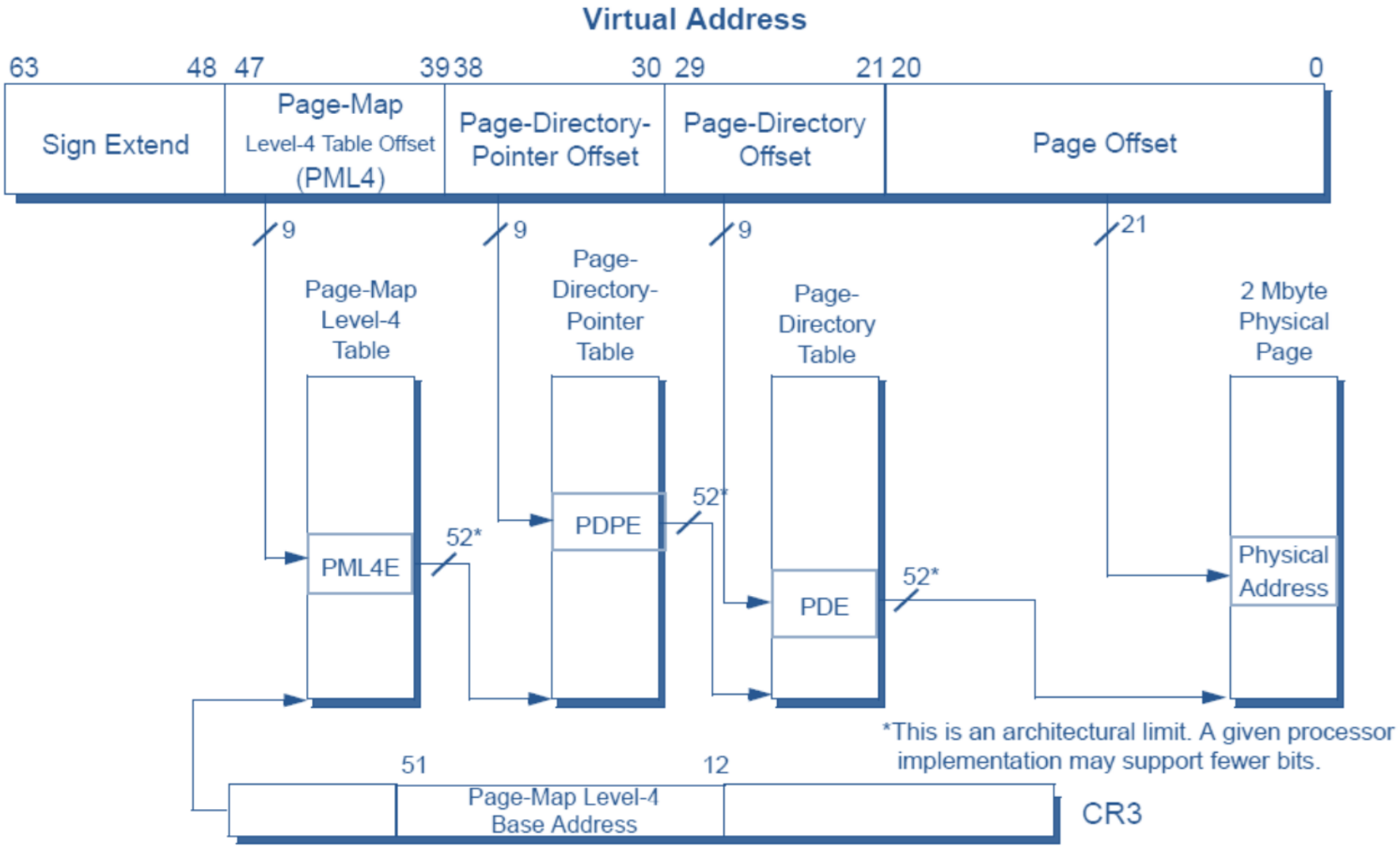
- Tento režim používá 48bitové virtuální adresy a generuje až 52bitové fyzické adresy pomocí stránek 4 KiB, 2 MiB nebo 1 GiB.
  - 52bitový adresový prostor může adresovat 4096 TiB fyzické paměti
    - pravděpodobně mnohem více, než jaké pravděpodobně budou mít systémy v blízké budoucnosti.
- Nejsložitější je stránkování s 4 KiB stránkami.
  - Bity 63:48 jsou znakové rozšíření bitu 47,
    - vyžadováno pro formáty kanonických adres (doplnění jedničkama nebo nulama).
  - Bity 47:39 se indexují do tabulky page-map level-4 (512 záznamů).
  - Bity 38:30 se indexují do tabulky ukazatelů adresářů stránek (512 záznamů).
  - Bity 29:21 se indexují do tabulky adresářů stránek (512 záznamů).
  - Bity 20:12 index do tabulky stránek (512 záznamů).
  - Bity 11: 0 obsahují offset stránky.

Stránkování v Long mode (64bit)



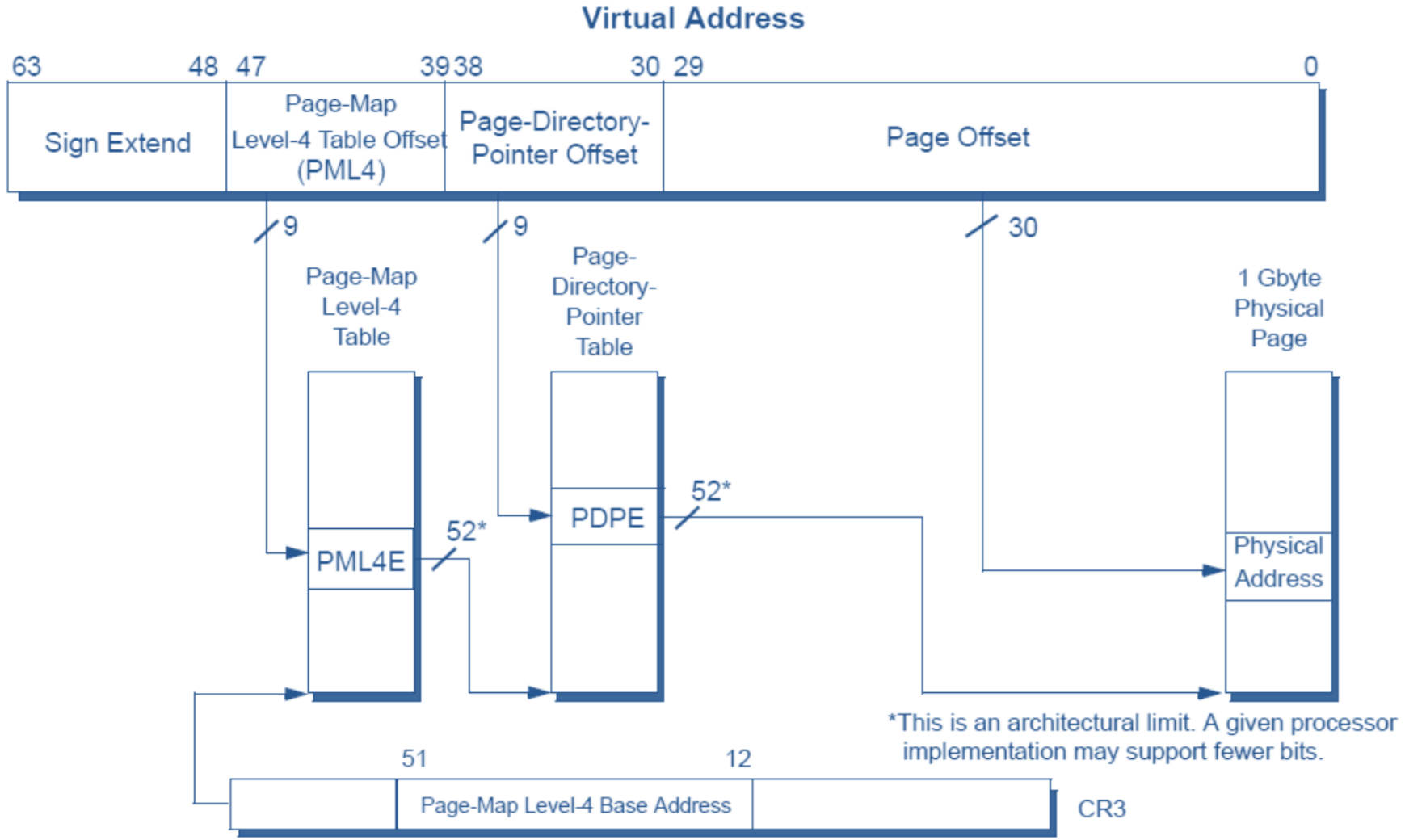
4-Kbyte Page Translation—Long Mode (AMD, 2018)

Stránkování v Long mode (64bit)



2-Mbyte Page Translation—Long Mode (AMD, 2018)

Stránkování v Long mode (64bit)



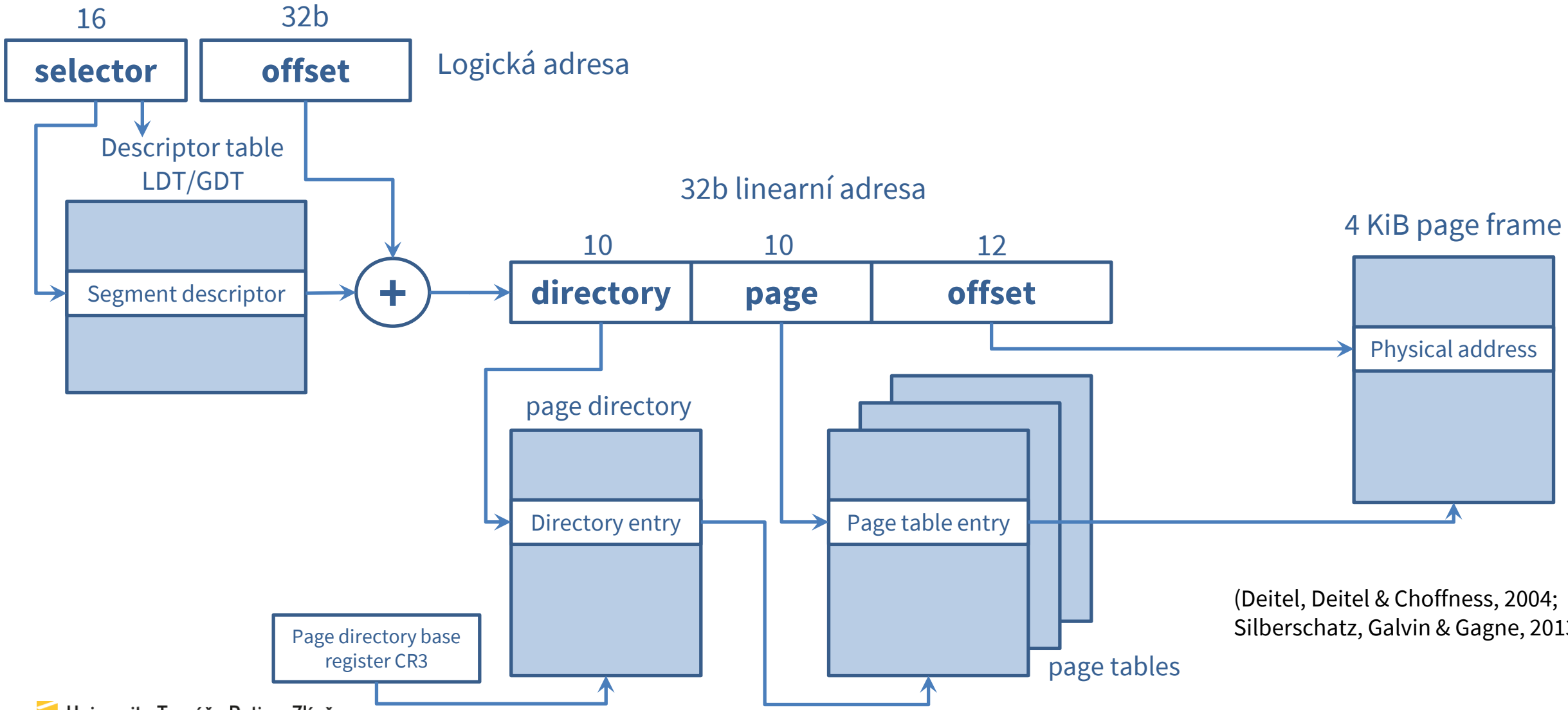
1-Gbyte Page Translation—Long Mode (AMD, 2018)

Segmentace se stránkováním

- V 32 bitovém módu nelze zrušit používání segmentu
  - x86-64 nepoužívá segmentaci v long mode (64bit režim).
- LAP: 2x8192 segmentů (GDT/LDT) na proces, s každý délkou až 4 GiB (32bitová adresa v segmentu)
- Lineární adresní prostor všech segmentů se stránkuje
- Moderní OS to řeší použitím 4GiB segmentu
  - Prakticky to ruší segmenty
  - Potom pouze 4 segmenty
    - Kernel Code Segment, Kernel Data Segment
    - User Code Segment and User Data Segment

(AMD, 2018; Štěpán, 2014, Lažanský, 2018)

Segmentace se stránkováním IA-32



(Deitel, Deitel & Choffness, 2004;
Silberschatz, Galvin & Gagne, 2013)

Virtualní paměť

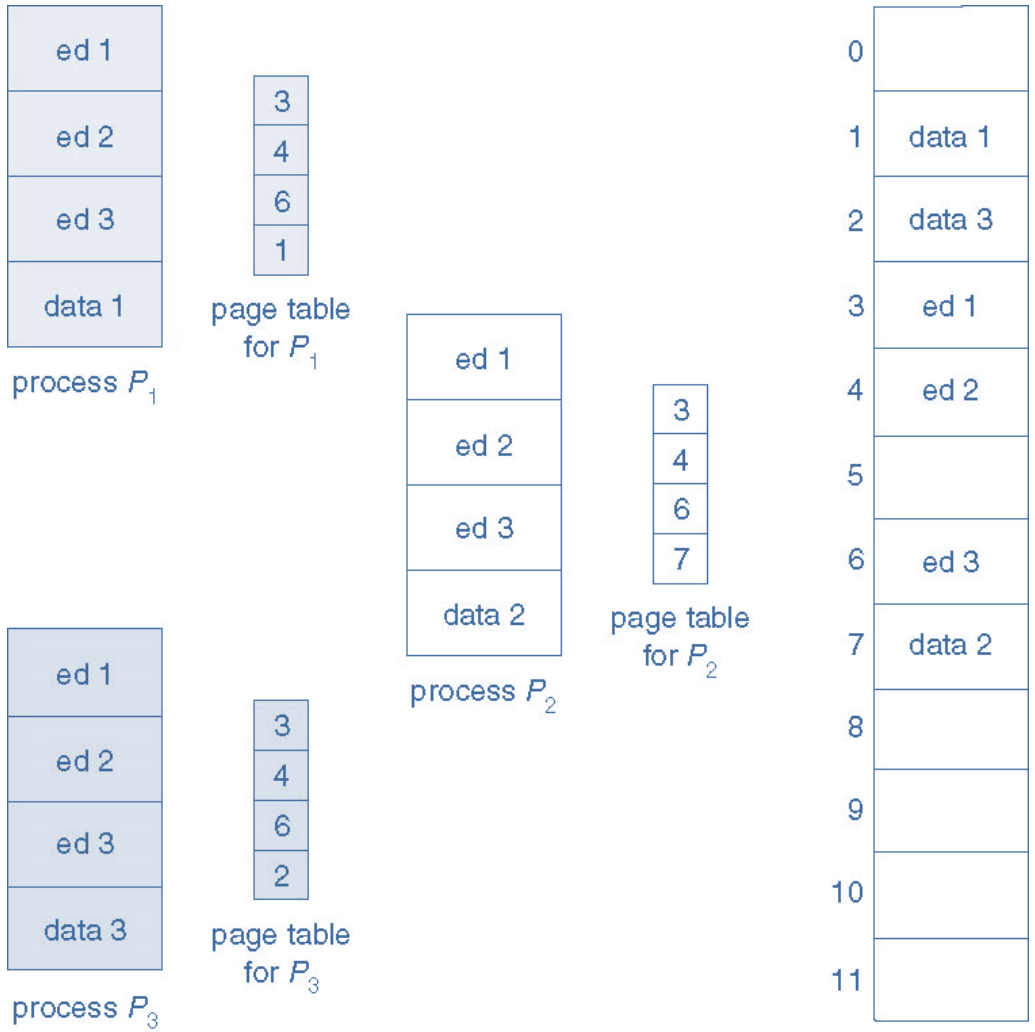
- Oddělení LAP od FAP. LAP může být mnohokrát větší než FAP.
  - Představuje iluzi větší fyzické paměti.
  - OS musí potom ukládat části virtuální paměti pro každý proces mimo hlavní paměť.
    - OS přepíná části virtuální paměti mezi hlavní pamětí a sekundárním úložištěm.
- Virtuální adresní prostor - logické zobrazení, jak je proces uložen v paměti. Možnost sdíleného kódu a dat.
- Jednotka správy paměti (MMU) musí logicky mapovat na fyzické adresy.
- Virtuální paměť je implementována pomocí dříve popsaného
  - Segmentace, stránkování

Sdílené stránky

- Sdílený kód a data
  - Jedna kopie read-only (reentrant) kódu sdílená mezi procesy
    - např. Textové editory, kompilátory, okenní systémy
  - Podobně jako více vláken sdílejících stejný procesní prostor
  - Užitečné pro meziprocesovou komunikaci, pokud je povoleno sdílení stránek pro čtení a zápis
- Soukromý kód a data
  - Každý proces uchovává samostatnou kopii kódu a dat
  - Stránky soukromého kódu a dat se mohou objevit kdekoli v prostoru logických adres

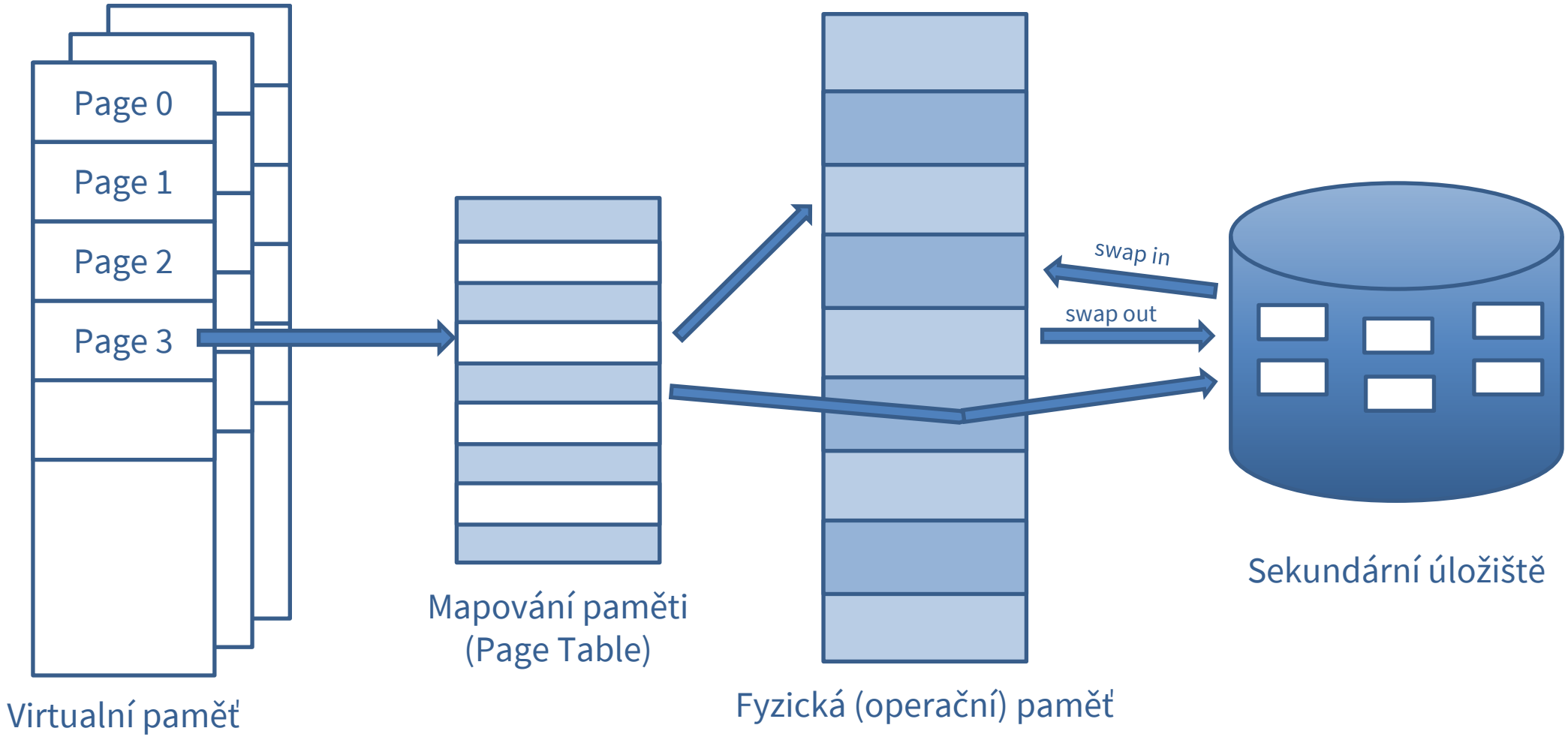
(Silberschatz, Galvin & Gagne, 2013)

Shared Pages Example



(Silberschatz, Galvin & Gagne, 2013)

Virtuální paměť



Zavádění stránky do FAP

Výpadek stránky (Page Fault)
odkaz mimo rezidentní množinu stránek. Takový proces je pozastaven a OS zavede chybějící stránku do FAP do volného rámce (případně uvolní rámec – *výběr oběti*)

- Stránkování při spuštení
  - Program je celý vložen do paměti při spuštění
    - velmi nákladné a zbytečné, předem nejsou známy nároky na paměť
- Stránkování (segmentace) na žádost (Demand Paging/Segmentation)
  - Načítá až v případě *výpadku stránky*
- Předstránkování (Prepaging)
  - Nahrává stránku, která bude pravděpodobně brzy použita. Princip lokality.
- Čištení (Pre-cleaning)
  - změněné rámce jsou ukládány na disk v době, kdy systém není vytížen
- Kopírování při zápisu (COW copy-on-write)
  - Při vytvoření nového procesu (fork) se nekopírují žádné stránky
    - ani kódové ani datové. U datových stránek se zruší povolení pro zápis.
  - Při pokusu o modifikaci datové stránky nastane chyba, která vytvoří kopii stránky a umožní modifikaci.

(Štěpán, 2014, Lažanský, 2018)

Princip lokality

- Odkazy na instrukce programu a data tvořívají shluky
- Vzniká časová lokalita a prostorová lokalita
  - Provádění programu je s výjimkou skoků a volání podprogramů sekvenční
  - Programy mají tendenci zůstávat po jistou dobu v rámci nejvýše několika procedur
  - Většina iterativních výpočtů představuje malý počet často opakovaných instrukcí prováděných v cyklech
  - Často zpracovávanou strukturou je pole dat nebo posloupnost záznamů, které se nacházejí v „sousedních“ paměťových lokacích
- Lze pouze dělat odhady o částech programu/dat, která budou potřebná v nejbližší budoucnosti.

(Štěpán, 2014, Lažanský, 2018)

Princip funkce

- **HW** – CPU (MMU)
  - MMU automaticky převádí logickou adresu programu na fyzickou podle tabulek stránek.
  - když MMU nemůže převést logickou adresu na fyzickou vyvolá výjimku (přerušení).
- **SW** – operační systém
  - Při svém zavádění nastaví CPU, aby používalo stránkování (tj. zapne MMU)
    - Kvůli zpětné kompatibilitě podporují moderní procesory více typu stránkování (32-bit, 64-bit, PAE, ...). OS si rozhodne, jaký typ se použije (většinou ten nejnovější, podporovaný jak HW, tak OS).
  - Plní obsahy tabulek stránek, aby logické adresy odpovídaly určeným fyzickým rámcům
    - Každý proces má vlastní tabulku stránek.
  - Řeší výjimky (výpadky stránek) – přístup k virtuálním adresám, které:
    - Nejsou mapovány do fyzické paměti
      - buď stránka na HDD (swap), nebo typicky chyby v programu, napr. dereference NULL ukazatele
    - Jsou mapovány s jinými příznaky
      - např. zápis do read-only stránky)buďto chyba nebo aktivace copy-on-write)

(Štěpán, 2014)

Použitá a doporučená literatura

- DUARTE, Gustavo. Anatomy of a Program in Memory. *Many But Finite: Tech and science for curious people.* [online]. 2009, Jan 27th, 2009 [cit. 2018-02-28]. Dostupné z: https://manybutfinite.com/post/anatomy-of-a-program-in-memory/
- DEITEL H. M., DEITEL P. J. & CHOFFNES D. R.: *Operating systems*. 3rd ed., Pearson/Prentice Hall, 2004. ISBN 0131246968.
- TANENBAUM A. S.: *Modern operating systems*. 4th ed. Boston: Pearson, 2015. ISBN 0-13-359162-x.
- SILBERSCHATZ A., GALVIN P. B. & GAGNE G.: *Operating system concepts*. 9th ed. Hoboken, NJ: Wiley, 2013. ISBN 978-1-118-06333-0.
- STALLINGS W.: *Operating Systems: Internals and Design Principles.* 8th ed., Pearson Education Limited, 2014.

Použitá a doporučená literatura

- YOSIFOVICH, P., IONESCU, A., RUSSINOVICH, M.E., SOLOMON, D. A.: Windows Internals, Part 1: System architecture, processes, threads, memory management, and more (7th Edition). Microsoft Press, 2017.
- Yu-Hsin Hung. *Linux Kernel: Process Scheduling* [online]. Mar 25, 2016 [cit. 2019-02-07]. Dostupné z: https://medium.com/hungys-blog/linux-kernel-process-scheduling-8ce05939fabd
- HOFFMAN, Chris. What Is a "Zombie Process" on Linux?. *How-To Geek* [online]. September 28, 2016 [cit. 2019-02-07]. Dostupné z: https://www.howtogeek.com/119815/htg-explains-what-is-a-zombie-process-on-linux/

Použitá a doporučená literatura

- AMD. *AMD64 Architecture Programmer's Manual: Volume 1: Application Programming* [online]. AMD64 Technology, [cit. 2019-03-02], 2018. Dostupné z: https://www.amd.com/system/files/TechDocs/24592.pdf
- AMD. *AMD64 Architecture Programmer's Manual: Volume 2: System Programming* [online]. AMD64 Technology, [cit. 2019-03-02], 2018. Dostupné z: https://www.amd.com/system/files/TechDocs/24593.pdf
- Krzyzanowski Paul. Memory Management: Paging [online]. 2012. Dostupné z: https://www.cs.rutgers.edu/~pxk/416/notes/10-paging.html