Virtuální paměť

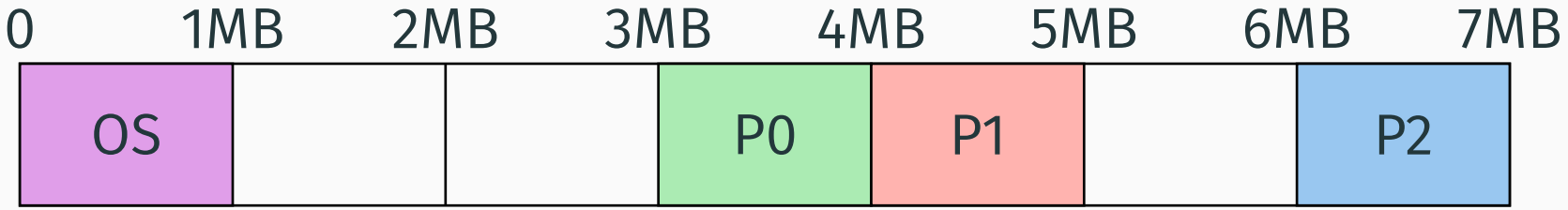
Stránkování, TLB, tabulky stránek

Milan Radojčić

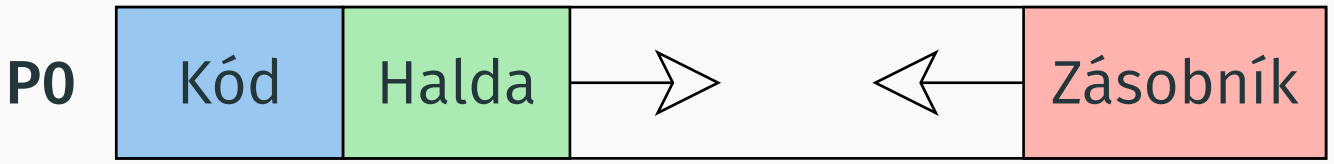
SSPŠaG – Operační systémy

4. 1. 2026

Opakování – base/bounds, adresový prostor



Rozdělení HP



Adresový prostor

- Co je smysl virtuální paměti?
- Co jsme si říkali minule?
- Kdo provádí překlad? (SW vs. HW)

Opakování – interní fragmentace

Význam virtuální paměti pro bezpečnost

- Exploiting the DRAM rowhammer bug to gain kernel privileges[1]
- Meltdown: Reading Kernel Memory from User Space[2]
- Spectre Attacks: Exploiting Speculative Execution[3]

---

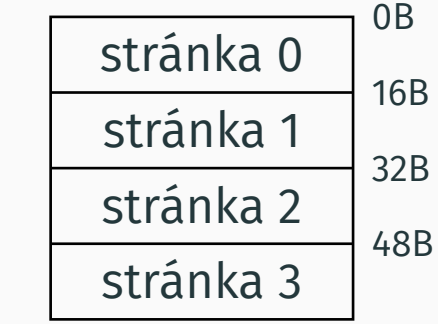
[1] https://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

[2] https://meltdownattack.com/meltdown.pdf

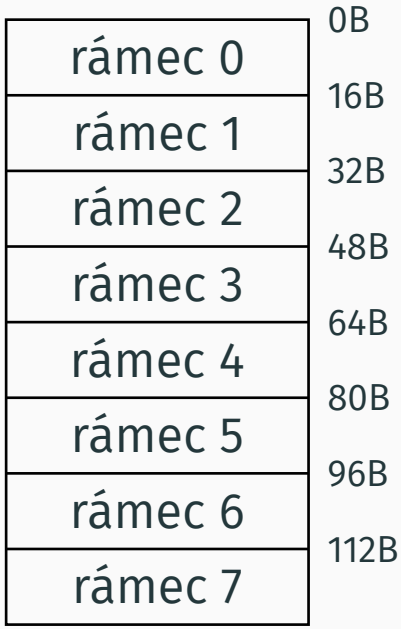
[3] https://spectreattack.com/spectre.pdf

Princip stránkování

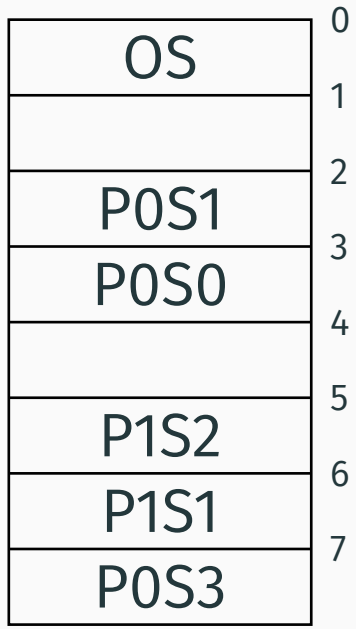
- Virtuální adresový prostor rozdělíme na **stránky**.
- Fyzickou paměť rozdělíme na **rámce**.
- Rámce i stránky budou mít **stejnou velikost**.
- **Virtuální** stránky budeme **mapovat** do **fyzických** rámců.



Adresový prostor



Hlavní paměť

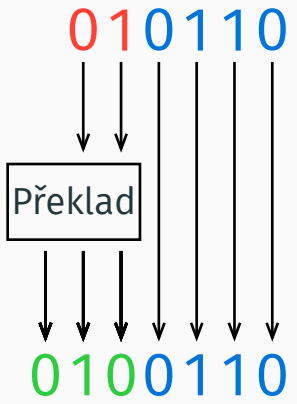


Mapování[4]

[4] P0S1 = proces 0, stránka 1, atd.

Jak překládat?

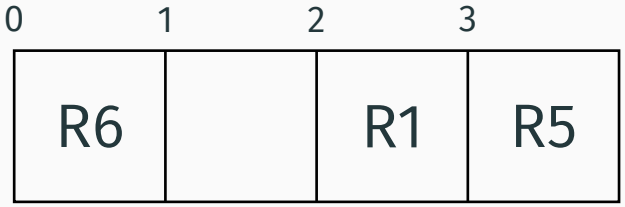
- Virtuální adresy rozdělíme na **číslo stránky a offset**.
- V našem příkladu máme **4 stránky a 16B v jedné stránce**.
- Rozdělení tedy bude **2b číslo stránky a 4b offset**: 010110.
- Stačí nám **jen překládat číslo stránky** na číslo rámce — offset necháme být!



Jak překládat?

Do jaké datové struktury překlady ukládat? A kde?

- Překlady uložíme do **jednorozměrného pole**.
- V *n*-té položce bude překlad *n*-té stránky.



- Pole uložíme do **hlavní paměti**.

Jak překládat?

- Překlad provádí **MMU** (memory management unit).
  - To je část CPU odpovědná za překlad adres.
- OS musí informovat MMU **kde** se v paměti nachází tabulka s překladem.

Průběh překladu

1. Je spuštěn nějaký proces, **do MMU je nahrána** adresa jeho **tabulky stránek**.
2. Proces chce přistoupit k nějaké adrese (nutně v jeho VAP!).
3. **MMU se podívá do tabulky stránek**, najde překlad.
   - Pokud překlad neexistuje/není validní, informuje o tom OS.
4. **Přepíše číslo stránky za číslo rámce** podle nalezeného překladu a k této adrese se přistupuje.

Problémy s jednoduchým stránkováním

Jaké vidíte problémy s tímto přístupem?

1. Je **pomalý** — každý přístup do paměti potřebuje ještě 1 extra přístup.
2. Nebudou **tabulky stránek moc velké**? Nechceme, aby překlady zabraly většinu hlavní paměti.

Velikost jednoduchých/lineárních tabulek

- Uvažme 32 bitový adresový prostor.
- Offset budeme brát 12b (velikost stránky bude 4KB).
- Na číslo stránky nám zbývá 20b.
- To znamená, že naše pole s překlady bude mít 2^{20} prvků!
- Řekněme, že jedna položka v tabulce bude mít 4B.[5]
- To nám celkem dává **4MB na jednu tabulku!**
- My ale **potřebujeme tabulku pro každý proces**.
- Pokud máme spuštěných 100 procesů, tak **400MB spotřebujeme jen na tabulky stránek!**

[5] Velikost závisí na HW. Obecně nemáme v tabulce jen překlady, ale ještě nějaké dodatečné informace. Více si řekneme dále.

Zrychlení překladů: TLB

Máte nápad jak zrychlit překlady?

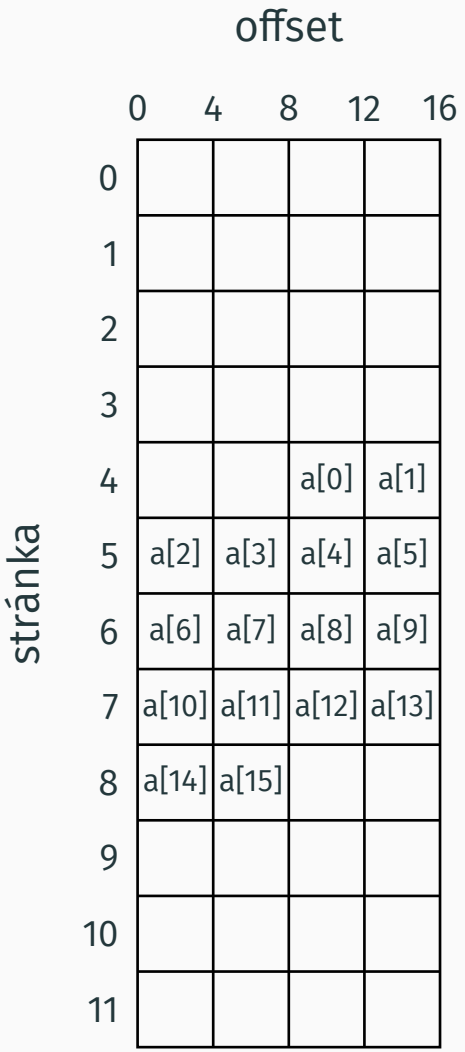
- Do MMU přidáme část, které se říká **TLB — translation lookaside buffer**.
- TLB bude **cachovat překlady**.
- MMU se při překladu nejprve podívá do TLB.
  - Pokud v TLB **je validní překlad, tak se použije**.
  - Pokud v TLB **není validní překlad**, tak se MMU **podívá** pro překlad **do hlavní paměti**. Poté ho **uloží do TLB**.

Proč TLB pomáhá?

- Při cachování využíváme obecně něčeho, čemu se říká **časová a prostorová lokalita**.
  - Často budeme znovu přistupovat k **datům, které jsme před chvílí potřebovali**.
  - Často budeme znovu přistupovat k **datům, co jsou blízko těch co jsme před chvílí potřebovali**.

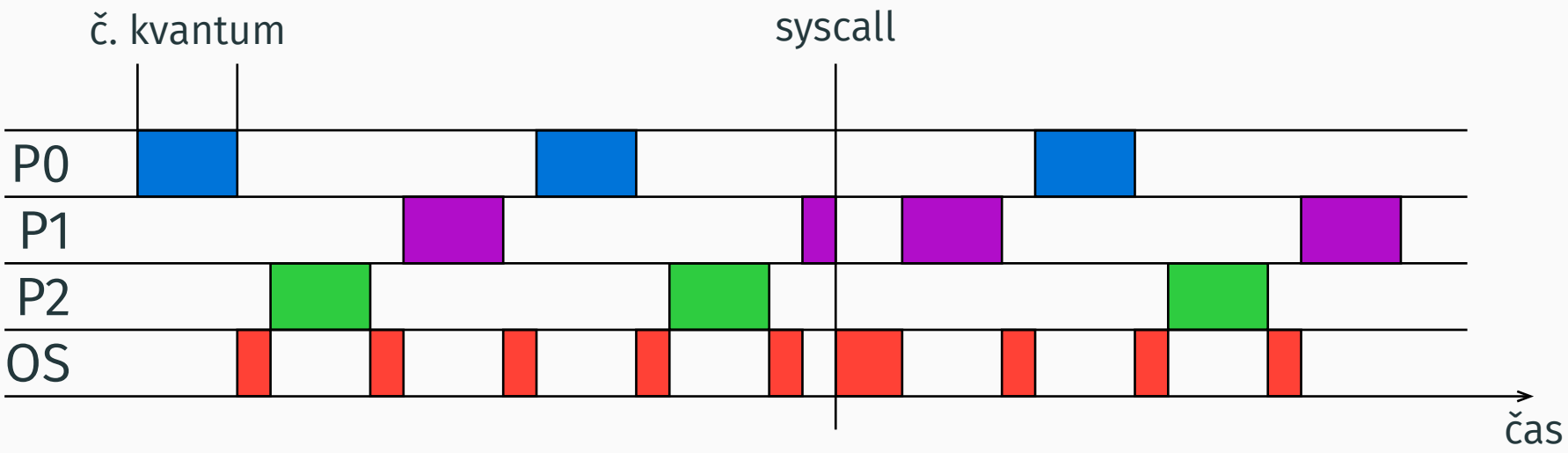
Proč TLB pomáhá?

- Představme si situaci na diagramu vpravo.
- Sekvenčně procházíme pole.
- Kolikrát se musí kvůli překladu přistoupit k hlavní paměti bez TLB?
- A kolikrát s TLB?



Pole v paměti

Problém TLB: změny kontextu



Preemptivní plánování

Máme jen jednu TLB na jádro CPU. Na co si musíme dát pozor, když se děje změna kontextu? (Vyměňuje se proces, který běží.)

Musíme si dávat pozor, abychom **nepletli překlady pro různé procesy**. Každý záznam v TLB má položku **ASID** (address space identifier), která slouží jako ID procesu, kterému patří tenhle překlad.

Ukázka záznamů TLB

č. stránky	č. rámce	valid	práva	ASID
1	3	ano	rwx	1
1	2	ne	r--	1
1	8	ano	r--	2
2	5	ano	r--	1

Menší tabulky stránek

Pomocí TLB jsme urychlili překlady, ale **jak zmenšíme tabulky?**
Napadá vás **jiná struktura?**

Menší tabulky stránek

Máme více možností jak na to:

1. **větší stránky,**[6]
2. **kombinace stránkování a segmentace,**[7]
3. **invertované tabulky stránek,**[8]
4. **víceúrovňové tabulky stránek.**

My si popíšeme jen víceúrovňové tabulky. Pro vysvětlení jiných principů viz např. *Operating Systems: Three Easy Pieces* nebo *Understanding the Linux Kernel*[9].

---

[6] Má přidaný bonus toho, že zlepší efektivitu TLB.

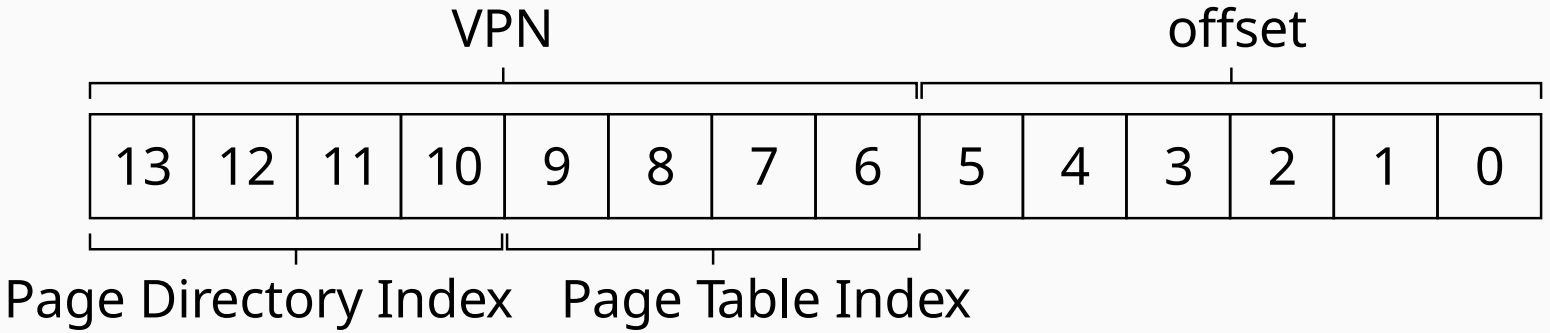
[7] Používal např. Multics.

[8] Používá např. SPARC.

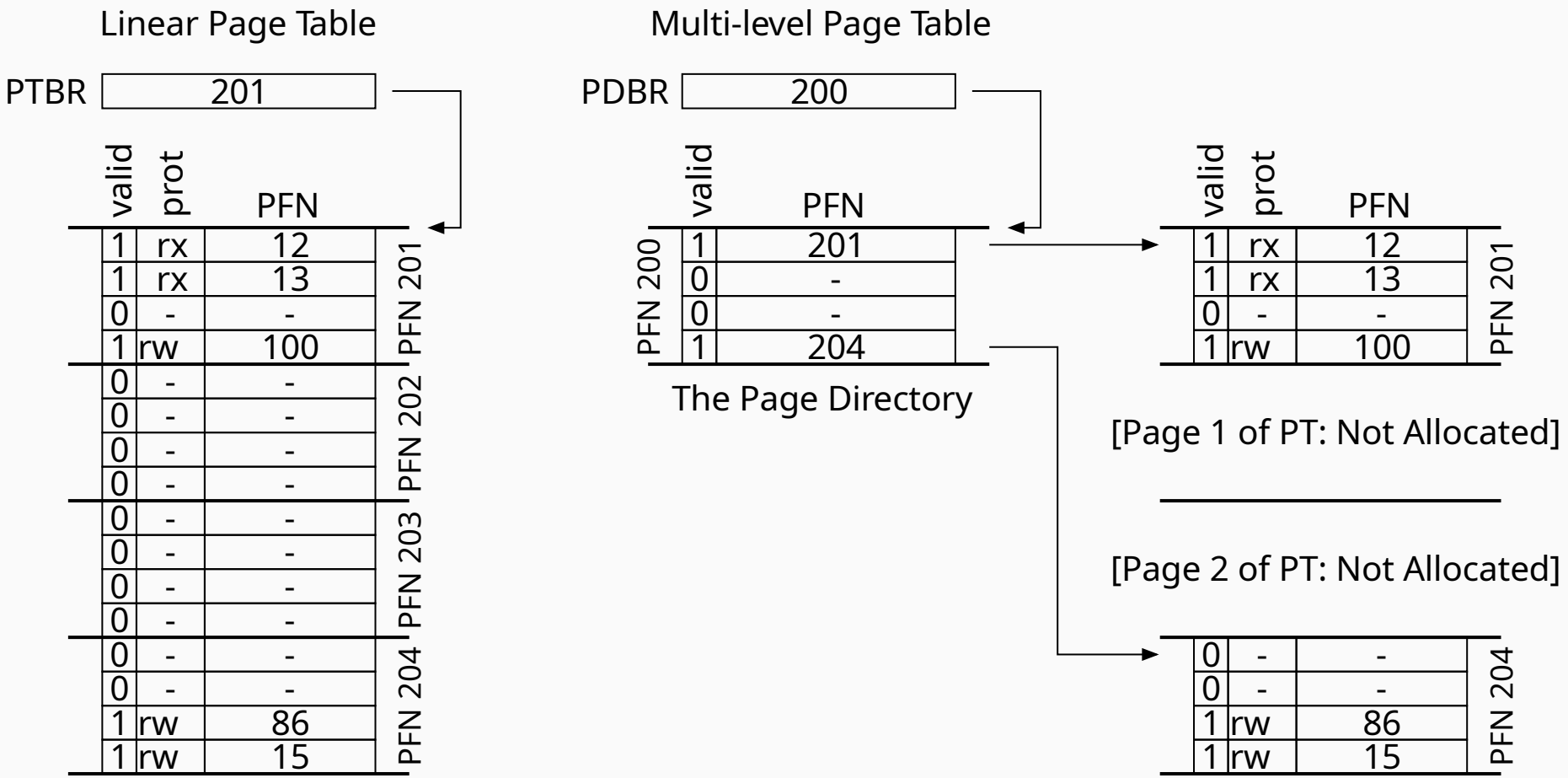
[9] Tohle dobře popisuje architekturu x86.

Víceúrovňové tabulky stránek

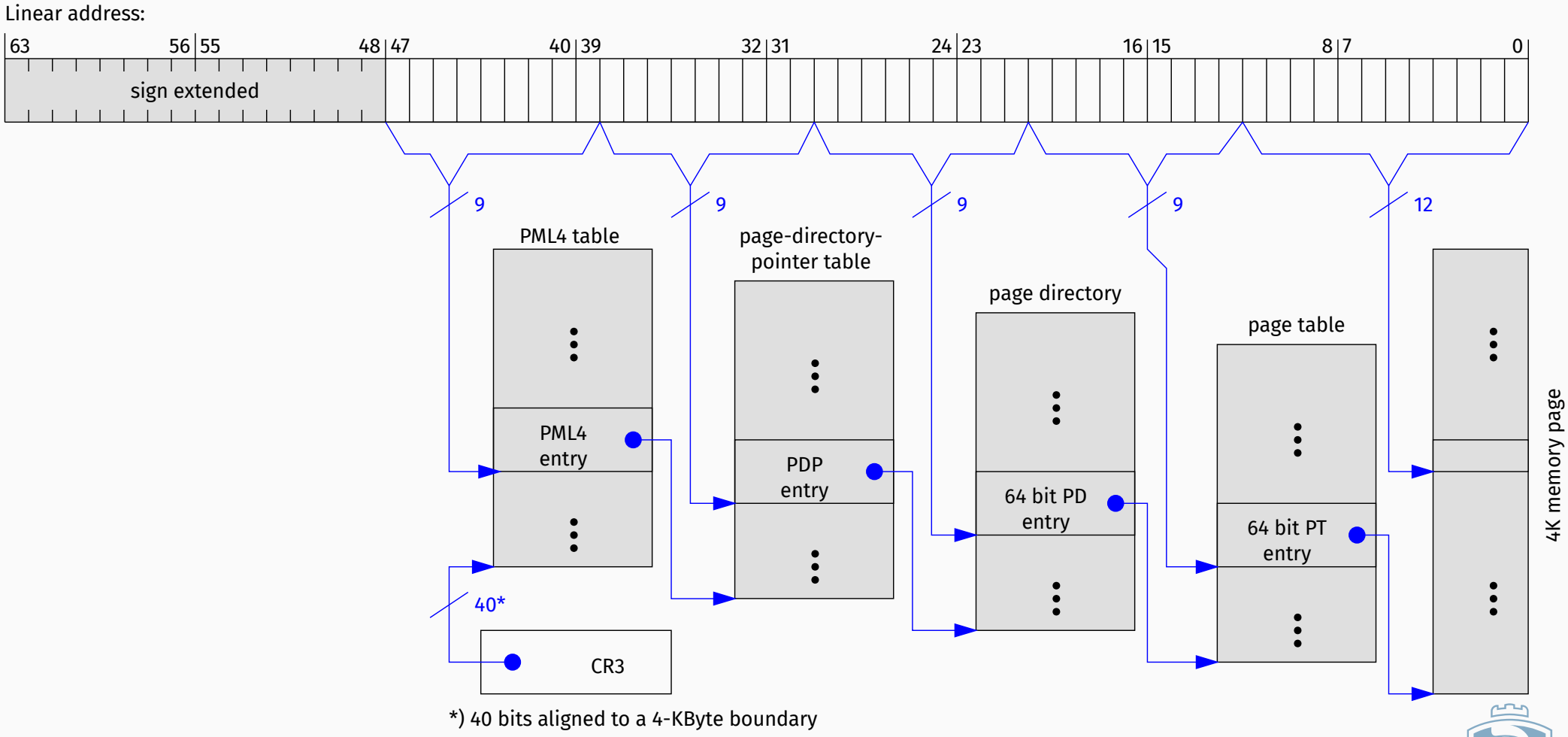
- Číslo stránky rozdělíme na více částí.
- Každá část funguje jako **offset do jiné úrovně překladu**.



Víceúrovňové tabulky stránek

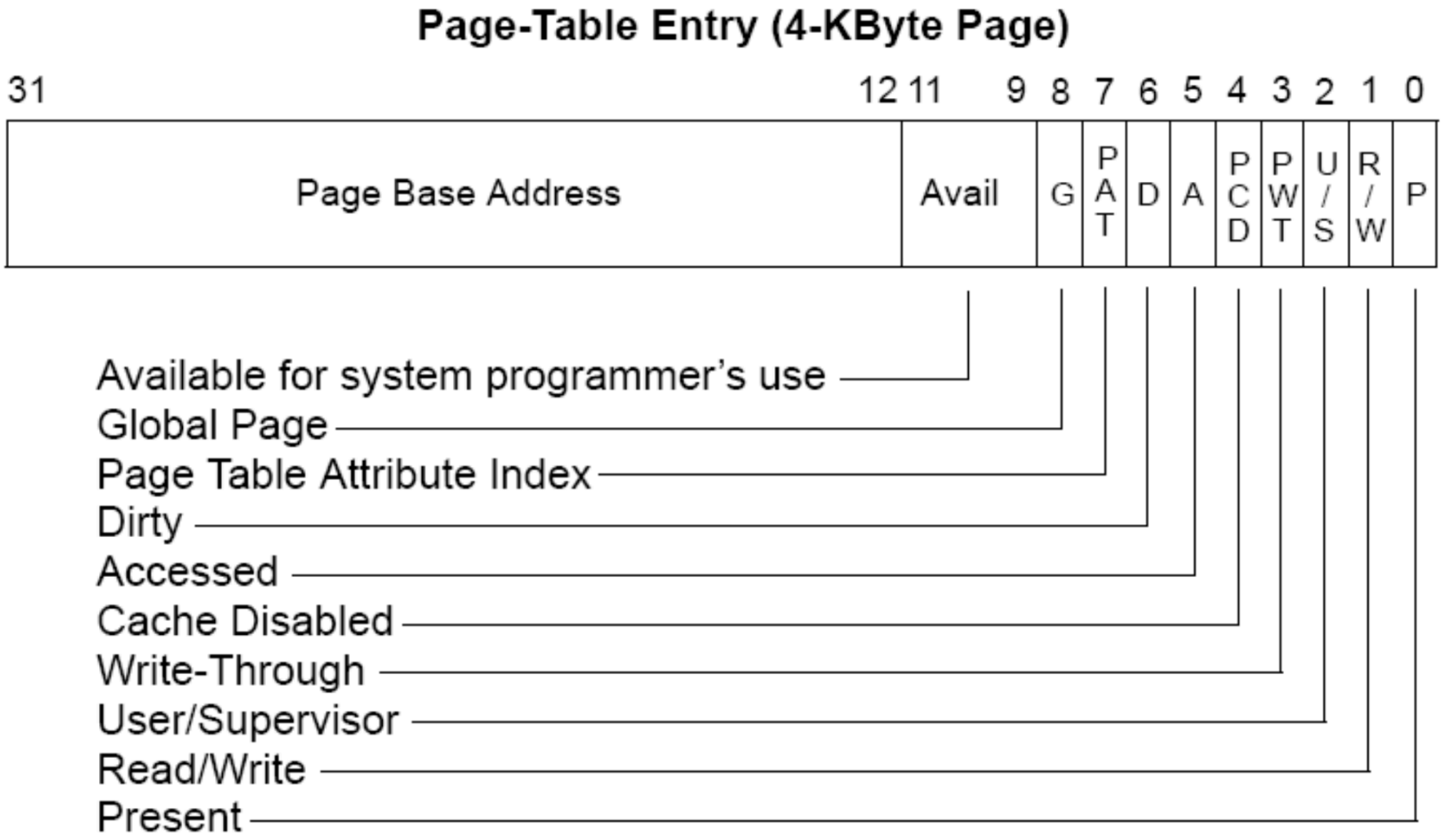


Struktura tabulky stránek architektury x86-64



Záznam v tabulce stránek x86-64

- **PTE of x86 architecture**



Shrnutí

- Stránkování, princip překladu, jednoúrovňové tabulky
- Zrychlení TLB, zmenšení tabulek pomocí víceúrovňových tabulek
- Vynechali jsme problematiku **nahrazování stránek** a **nahrazování záznamů v TLB**.[10]

[10] Viz např. *Operating Systems: Three Easy Pieces* – https://ostep.org.

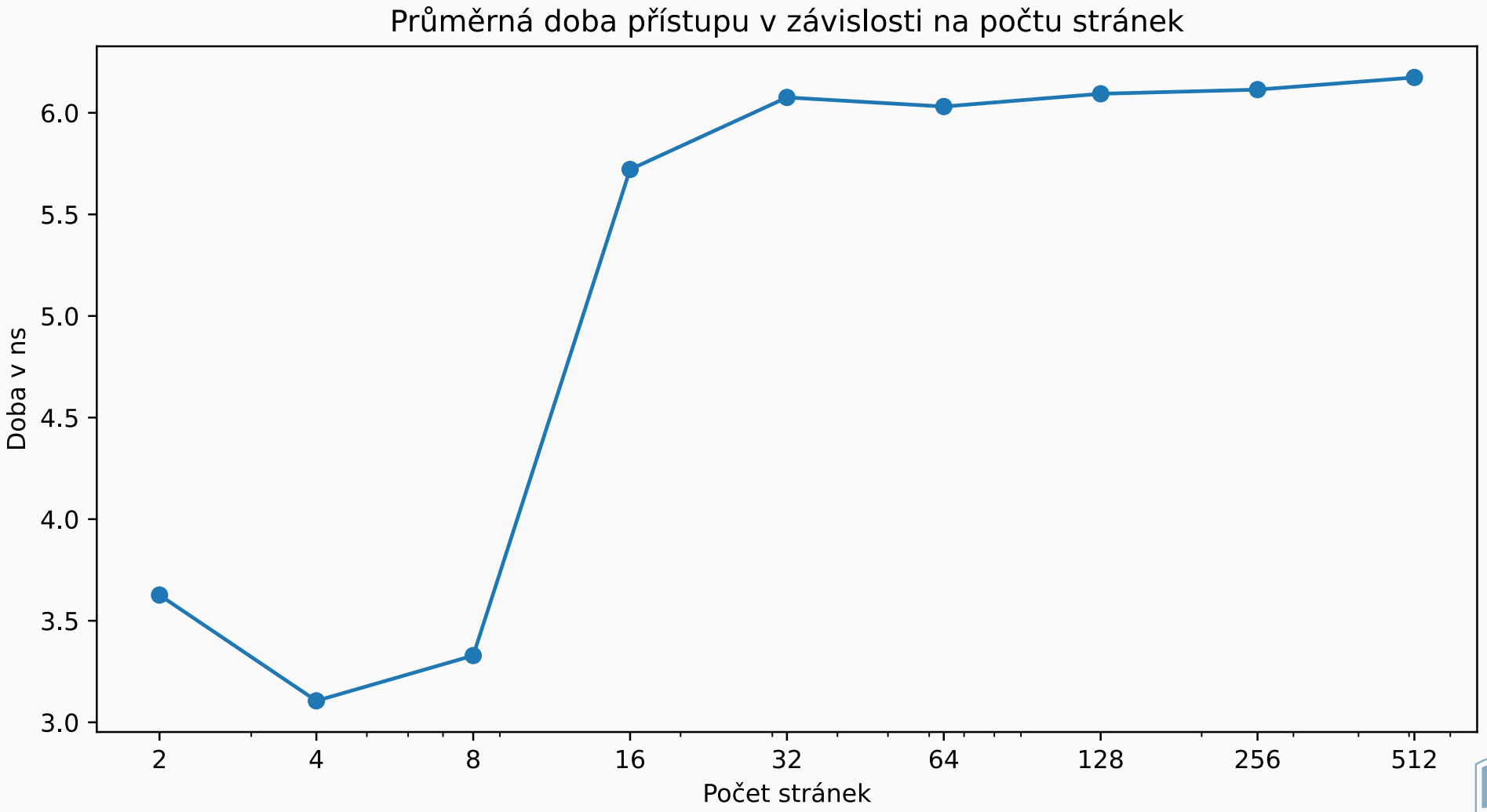
Bonusové cvičení

- Jako bonus za 3 body si můžete zkusit **změřit efekt TLB na rychlost přístupu do paměti**.
- Základní kód pro měření:

```c
int jump = PAGESIZE / sizeof(int);
for (i = 0; i  < NUMPAGES * jump; i += jump)
  a[i] += 1;
```

- Toto chcete spouštět ve smyčce (např. 1000 krát) a měřit průměrnou dobu jednoho přístupu do paměti.
- Okolo toho všeho chcete mít další smyčku, která bude měnit počet stránek.

Bonusové cvičení: ukázka



Bonusové cvičení: tipy

- Pomocí `prctl(PR_SET_THP_DISABLE, 1, 0, 0, 0)` vypněte Transparent Huge Pages.
- K měření můžete použít `clock_gettime( ... )` (přes `clock_getres( ... )` jde zkontrolovat přesnost).
- Pomocí `sched_setaffinity( ... )` můžete operačnímu systému říct, aby plánoval jen na některé konkrétní jádro.
- Měli by jste vidět 2 hezké úrovně, pokud tomu tak není zkontrolujte měření.
  - Jak počítáte průměrný čas? Nedochází k overflow někde?
  - Neměřte časy ve vnitřní smyčce, zkazí vám to průměry.
  - Zkuste zvětšit počet spuštění pro jednu velikost, které se průměrují.[11]

[11] Hlavně menší počty stránek se můžou chovat divně, když máte moc nízký počet spuštění.

Bonusové cvičení: odevzdávání

- Odevzdejte mailem do uzavření pololetí klasifikace.
- Měli by jste odevzdat:
  1. Váš kód pro měření.
  2. Nějaký graf z vašeho měření.
  3. Krátký popis, co jste zjistili (jak velká je cca L1, případně L2 TLB).