Operační systémy

Virtální paměť, stránkování.

Jan Trdlička

České vysoké učení technické v Praze, Fakulta informačních technologií
Katedra počítačových systémů

https://courses.fit.cvut.cz/BI-OSY

Obsah přednášky

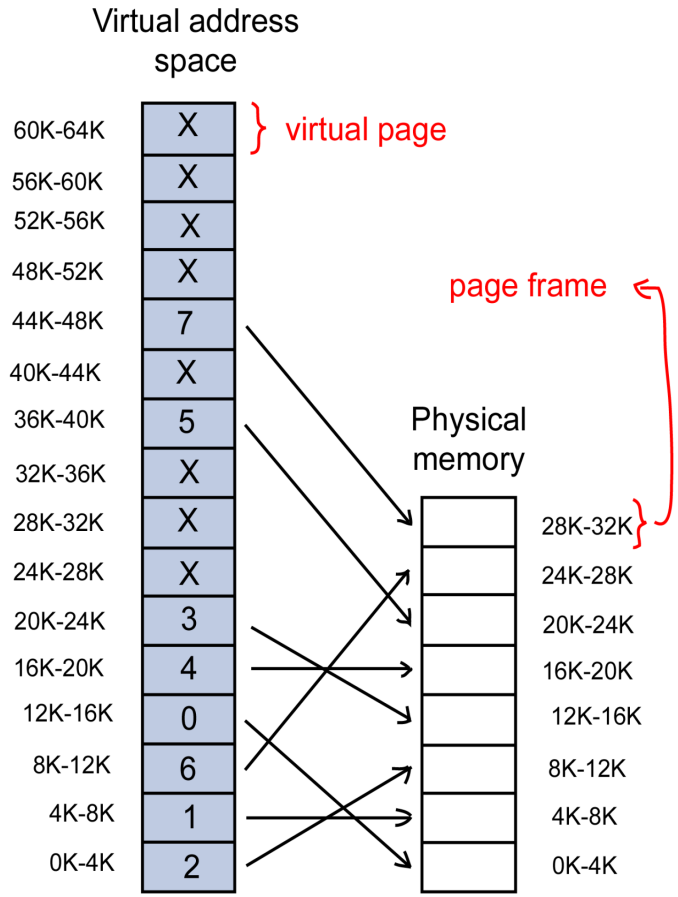
1. Virtuální paměť
2. Virtuální paměť se stránkováním
3. Překlad virtuálních adres na fyzické
   - Memory management unit (MMU)
   - Tabulka stránek
   - Víceúrovňová tabulka stránek
   - Invertovaná tabulka stránek
   - Translation lookaside buffer (TLB)

Virtuální paměť

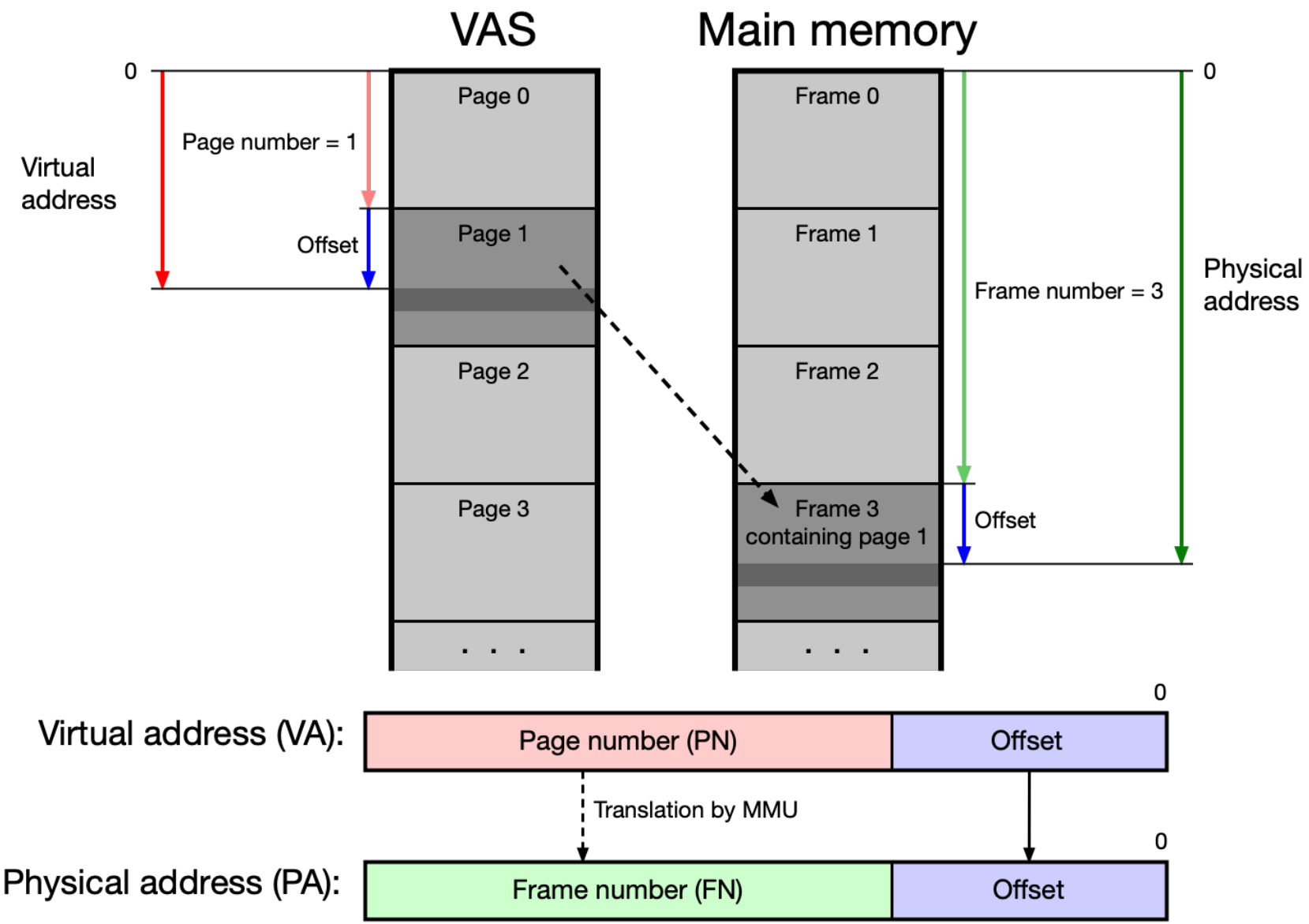
- **Příklad**
  - Ve 32-bitovém OS, jeden 32-bitový proces může adresovat až 2^{32} B $= 4$ GB.
  - V 64-bitovém OS, jeden 64-bitový proces může teoreticky adresovat až 2^{64} B $= 16$ EB (v praxi se zatím nevyužívá všech 64 bitů pro adresaci VAS).
  - Většina procesů ale reálně používá pouze zlomek prostoru ze svého VAS.
- **Problém**
  - VAS jednoho procesu může být větší než instalovaná fyzická paměť systému.
  - OS umožňuje současně spustit až tisíce procesů $\Rightarrow$ součet velikostí jednotlivých VAS je opět větší než instalovaná fyzická paměť systému.
- **Řešení: Virtuální paměť**
  - VAS každého procesu je automaticky (pomocí HW/OS) rozdělen na menší kousky a ve fyzické paměti musí být pouze kousky aktuálně používané, zbytek používaného VAS je na disku.

Virtuální paměť se stránkováním

- **Většinou je virtuální paměť kombinována se stránkování.**
- **Princip**
  - Proces používá adresy, kterým se říká virtuální adresy a které adresují virtuální adresový prostor.
  - VAS je rozdělen na stejně velké souvislé oblasti nazývané virtuální stránky (virtual pages). V závislosti na architektuře CPU je minimální velikost 4 KB (Intel), 8KB (Sparc).
  - Korespondující úseky ve fyzické paměti stejné velikosti jsou nazývány rámce stránek (page frames).
  - V hlavní paměti musí být aspoň stránky aktuálně používané.

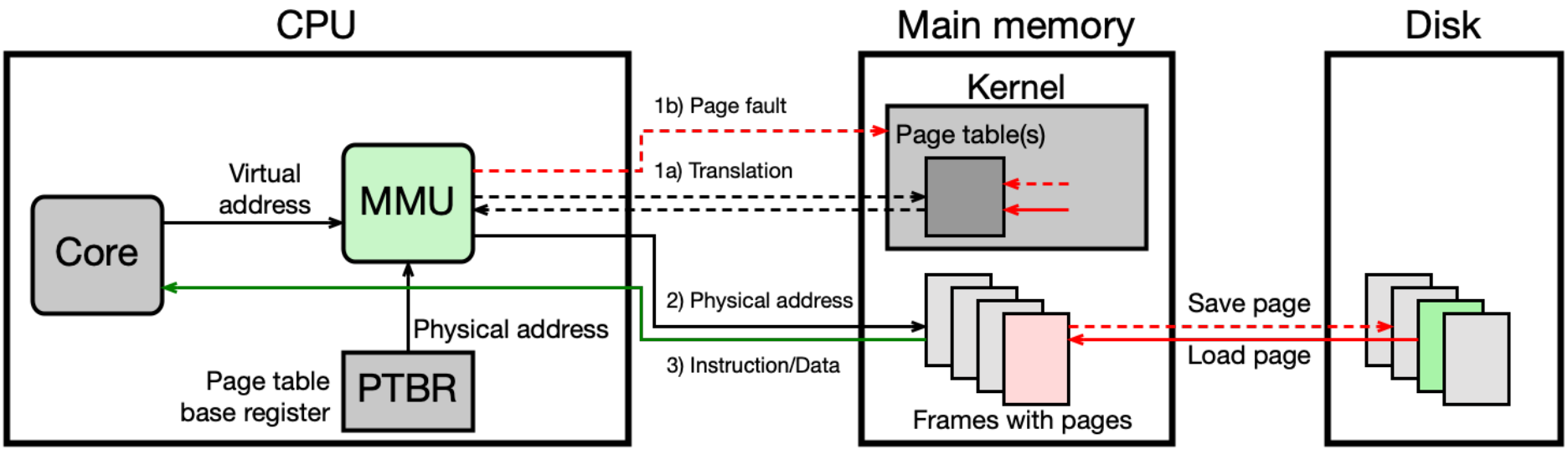


Struktura virtální a fyzické adresy



Memory management unit (MMU)

- Proces má iluzi, že celý jeho VAS je v hlavní paměti a pro adresaci instrukcí/dat používá virtuální adresy (vztažené k začátku VAS).
- Tuto iluzi a překlad adres zajišťuje hardware, který se nazývá **"Memory management unit" (MMU)**, ve spolupráci s OS.
- **Výpadek stránky (Page fault)**
  - Pokud není virtuální stránka v hlavní paměti, MMU způsobí přerušení a "požádá" OS o nahrání příslušné stránky do fyzické paměti.
  - OS nejdříve najde vhodný rámec fyzické paměti (pokud je nutné, uloží jeho obsah na disk), a pak do něj nahraje požadovanou virtuální stránku z disku nebo v něm "vytvoří" novou stránku (zásobník, heap).



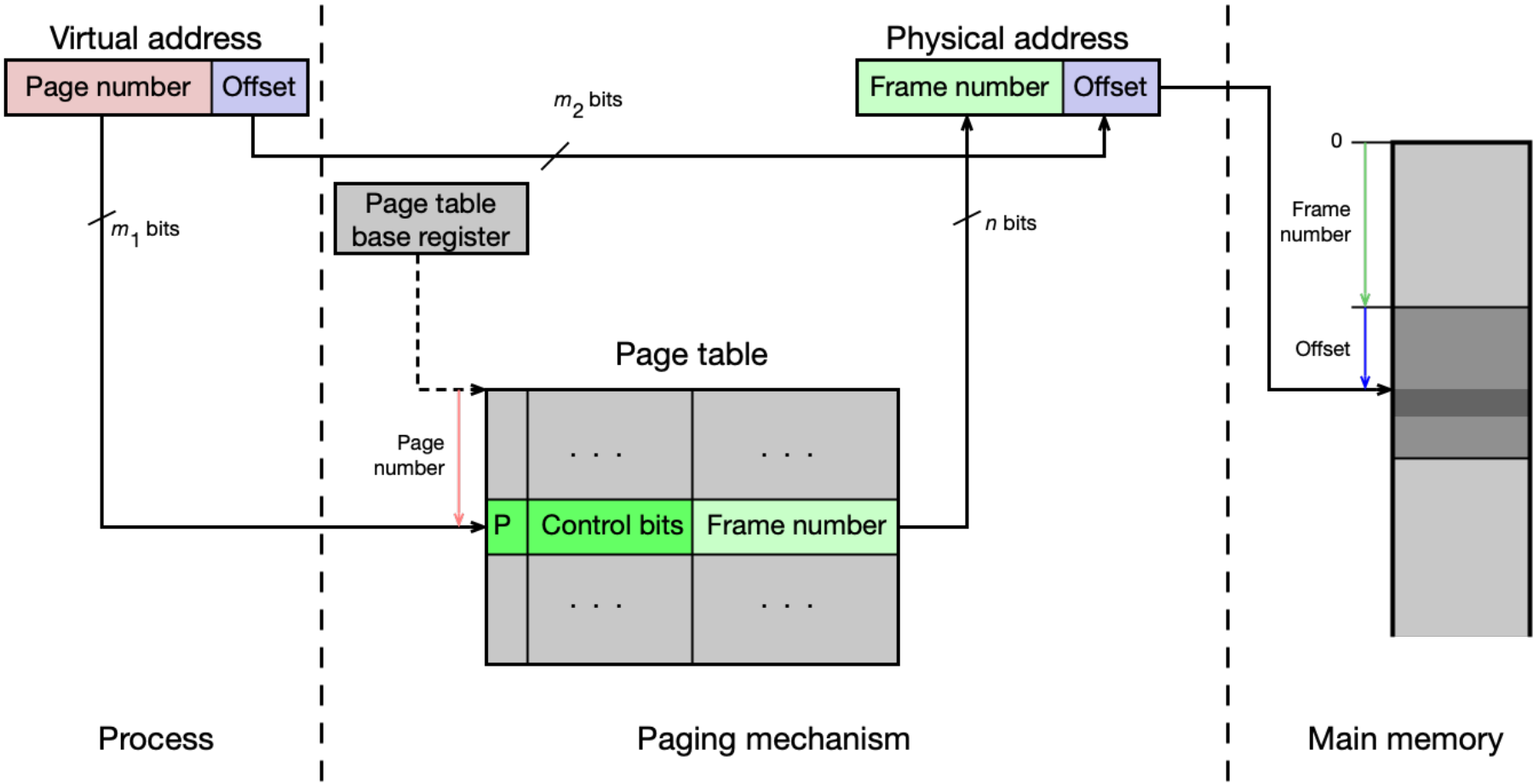
Překlad virtálních adres na fyzické

- OS si musí udržovat informaci o tom, do kterých rámců fyzické paměti se namapovaly jednotlivé stránky VAS jednotlivých procesů, a které rámce fyzické paměti jsou zatím volné.
- Tuto informaci si OS udržuje v následujících strukturách, které jsou závislé na ISA použitého procesoru,
  - tabulka stránek (page table),
  - víceúrovňová tabulka stránek (multilevel page table),
  - invertovaná tabulka stránek (inverted page table).
- Pro urychlení překladu pak MMU využívá "Translation lookaside buffer" (TLB), ve kterém jsou informace o naposledy překládaných virtuálních adresách.

Tabulka stránek

- Tabulka obsahuje pro každou stránku VAS daného procesu jednu řádku, ve které jsou uložené následující informace
  - **číslo rámce**, do kterého se tato stránka namapovala,
  - **kontrolní bity**
    - Present bit (P): informace, zda stránka je v hlavní paměti,
    - Reference bit (R): informace, zda se ke stránce přistupovalo,
    - Modify bit (M): informace, zda byl obsah stránky modifikován,
    - Přístupová práva
    - Cache disable/enabled: zda jsou registry periferií mapovány přímo do paměti,
    - Read/write (R/W): informace, zda lze stránku modifikovat,
    - User/supervisor (U/S): informace, zda lze na stránku přistupovat v uživatelském modu, ...
- Číslo stránky (page number) funguje jako index do této tabulky.
- OS si musí udržovat pro každý proces jednu tabulku.

Tabulka stránek: Překlad virtuální adresy



Tabulka stránek

- **Příklad**
  - Na 32-bitovém CPU, které podporuje pouze 4KB stránky/rámce, je nainstalovaný 32-bitový OS, ve kterém běží 32-bitový proces.
  - Předpokládejme, že proces bude alokovat ve svém VAS pouze následující datové struktury
    - TEXT a DATA: 4 MB na nejnižších virtuálních adresách,
    - halda: 4 MB na následujících virtuálních adresách,
    - zásobník: 4 MB na nejvyšších virtuálních adresách.

Tabulka stránek

- **Příklad**
  - Na 32-bitovém CPU, které podporuje pouze 4KB stránky/rámce, je nainstalovaný 32-bitový OS, ve kterém běží 32-bitový proces.
  - Předpokládejme, že proces bude alokovat ve svém VAS pouze následující datové struktury
    - ⋆ TEXT a DATA: 4 MB na nejnižších virtuálních adresách,
    - ⋆ halda: 4 MB na následujících virtuálních adresách,
    - ⋆ zásobník: 4 MB na nejvyšších virtuálních adresách.
- **Jaká bude struktura virtuální a fyzické adresy?**
  - Virtuální adresa (32 bitů): číslo stránky (20 bitů) + offset (12 bitů).
  - Fyzická adresa (32 bitů): číslo rámce (20 bitů) + offset (12 bitů).
- **Kolik řádek bude mít tabulka stránek?**
  - Pro každou stránku musí být jedna řádka $\Rightarrow 2^{20}$ řádek.
- **Kolik místa zabírá jedna řádka?**
  - Číslo rámce (20 bitů) + kontrolní bity (P+R+M+...)$\Rightarrow$ zaokrouhlíme na celé bytes/slova ~32 bitů = 4 B.
- **Kolik místa zabírají všechny tabulky pokud na systému běží 2^7 podobných procesů?**
  - $2^7 \times [2^{20} \times 4]$ B = $2^7 \times 4$ MB = 512 MB.

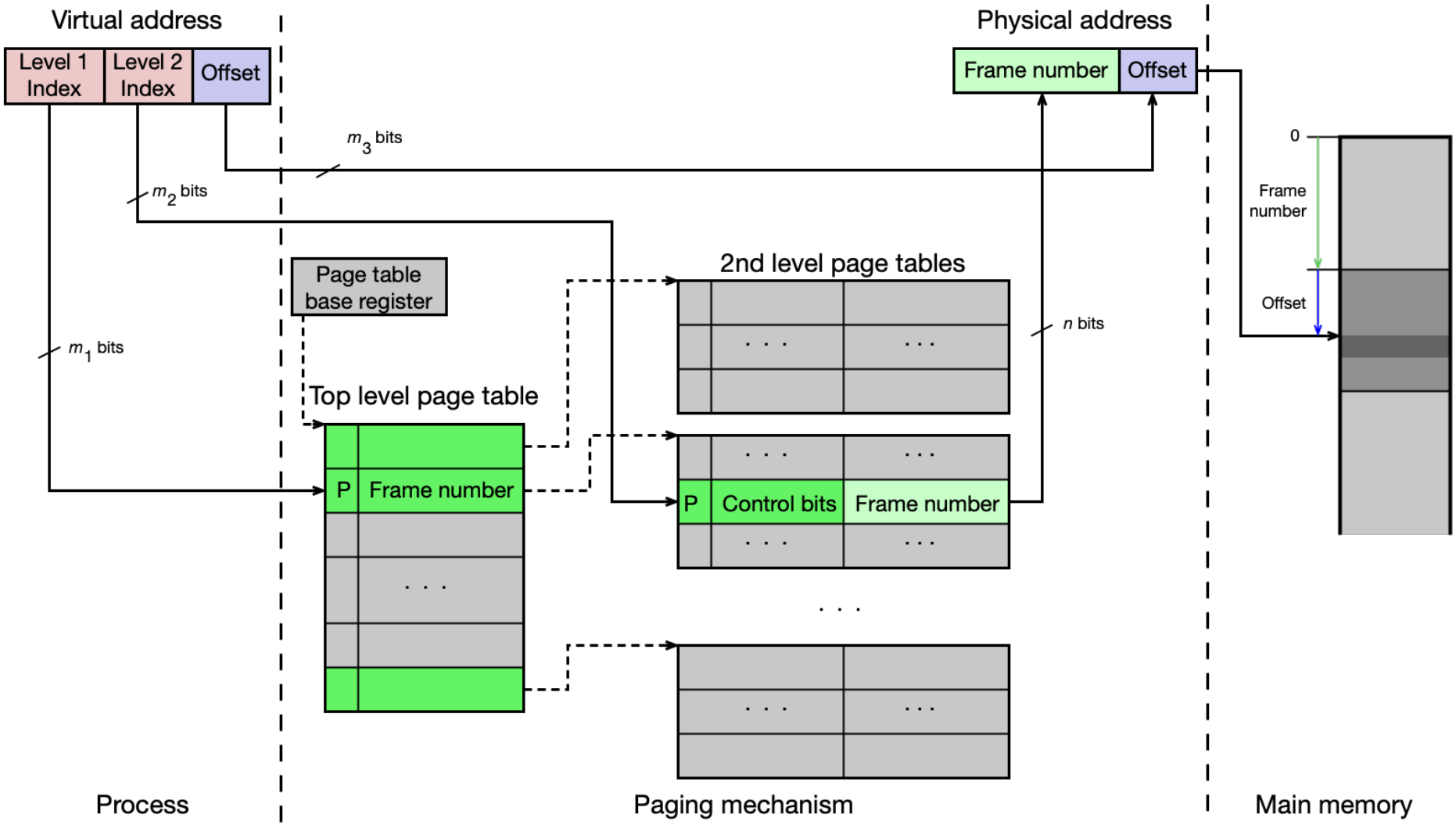
Tabulka stránek: Problémy

- **Problémy**
  - Přestože proces používá pouze zlomek místa ze svého VAS, tak tabulka stránek obsahuje informaci i o nepoužitých stránkách.
  - Pro každý proces musí OS držet jednu tabulku stránek.
    $\Rightarrow$ plýtvání pamětí.
- **Řešení**
  - Víceúrovňová tabulka stránek, která umožňuje se držet v paměti pouze informace o používaných stránkách.
    $\Rightarrow$ u procesů, které alokují málo paměti, můžeme výrazně ušetřit.

Víceúrovňová tabulka stránek

- **Pro n úrovňovou tabulku stránek platí**
  - virtuální adresa se skládá z n indexů, které ukazují do tabulek jednotlivých úrovní, a offsetu,
  - fyzická adresa se skládá z čísla rámce a offsetu,
  - tabulky stránek úrovní $1, \ldots, n-1$ obsahují "present bit" a číslo rámce, ve kterém je uložena/začíná tabulka následující úrovně,
  - tabulka stránek úrovně n obsahuje "present bit" a číslo rámce, ve kterém je uložena samotná virtuální stránka.
- V hlavní paměti je vždy "Top level tabulka" (tabulka úrovně 1).
- Tabulky ostatních úrovní v paměti být nemusí , pokud proces nepoužívá stránky z oblastí VAS, která tyto tabulky pomáhají adresovat $\Rightarrow$ šetří se fyzická paměť.
- Za ušetřené místo však platíme pomalejším překladem $\Rightarrow$ pro urychlení překladu se používá společně s TLB.

Dvouúrovňová tabulka stránek



Víceúrovňová tabulka stránek

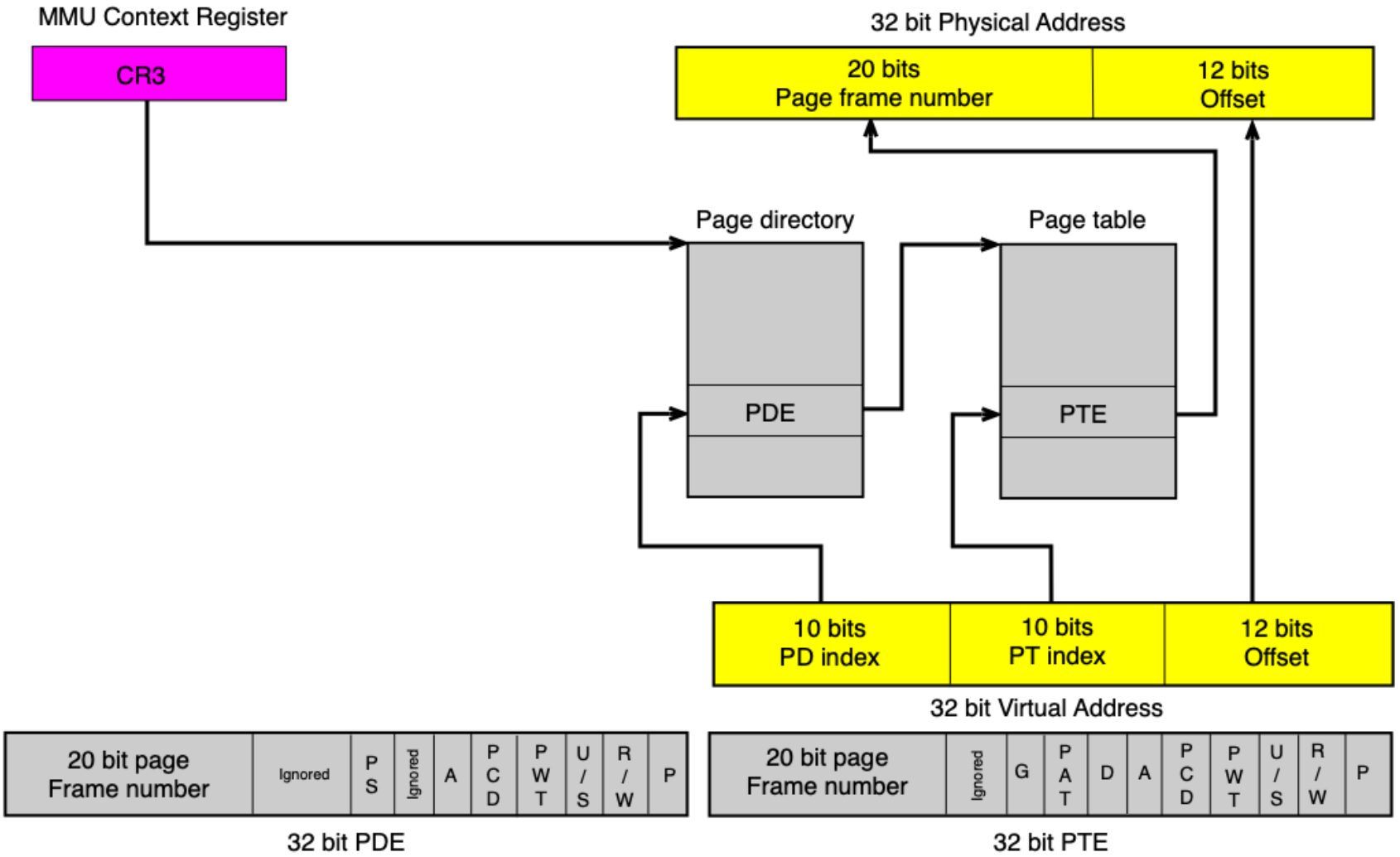
- **Příklad**
  - Na 32-bitovém CPU, které podporuje pouze 4KB stránky/rámce, je nainstalovaný 32-bitový OS, ve kterém běží 32-bitový proces. Systém používá dvouúrovňovou tabulku stránek a indexy do jednotlivých tabulek mají stejnou velikost.
  - Předpokládejme, že proces bude alokovat ve svém VAS pouze následující datové struktury
    - TEXT a DATA: 4 MB na nejnižších virtuálních adresách,
    - halda: 4 MB na následujících virtuálních adresách,
    - zásobník: 4 MB na nejvyšších virtuálních adresách.

Víceúrovňová tabulka stránek

- **Příklad**
  - Na 32-bitovém CPU, které podporuje pouze 4KB stránky/rámce, je nainstalovaný 32-bitový OS, ve kterém běží 32-bitový proces. Systém používá dvouúrovňovou tabulku stránek a indexy do jednotlivých tabulek mají stejnou velikost.
  - Předpokládejme, že proces bude alokovat ve svém VAS pouze následující datové struktury
    - ★ TEXT a DATA: 4 MB na nejnižších virtuálních adresách,
    - ★ halda: 4 MB na následujících virtuálních adresách,
    - ★ zásobník: 4 MB na nejvyšších virtuálních adresách.
- **Jaká bude struktura virtuální a fyzické adresy?**
  - Virtuální adresa (32 bitů): level 1 index (10 bitů) + level 2 index (10 bitů) + offset (12 bitů).
  - Fyzická adresa (32 bitů): číslo rámce (20 bitů) + offset (12 bitů).
- **Kolik řádek bude mít top level tabulka?** $\Rightarrow 2^{10}$ řádek.
- **Kolik místa zabírá jedna řádka?**
  - Číslo rámce (20 bitů) + kontrolní bit (P)$\Rightarrow$ zaokrouhlíme ~32 bitů = 4 B.
- **Kolik řádek bude mít tabulka druhé úrovně?** $\Rightarrow 2^{10}$ řádek.
- **Kolik místa zabírá jedna řádka?**
  - Číslo rámce (20 bitů) + kontrolní bity (P,...)$\Rightarrow$ zaokrouhlíme ~32 bitů = 4 B.
- **Kolik místa zabírají všechny tabulky pokud na systému běží 2^7 podobných procesů?**
  - $2^7 \times [1 \times (2^{10} \times 4) + 3 \times (2^{10} \times 4)]$ B = $2^7 \times 16$ KB = 2 MB.

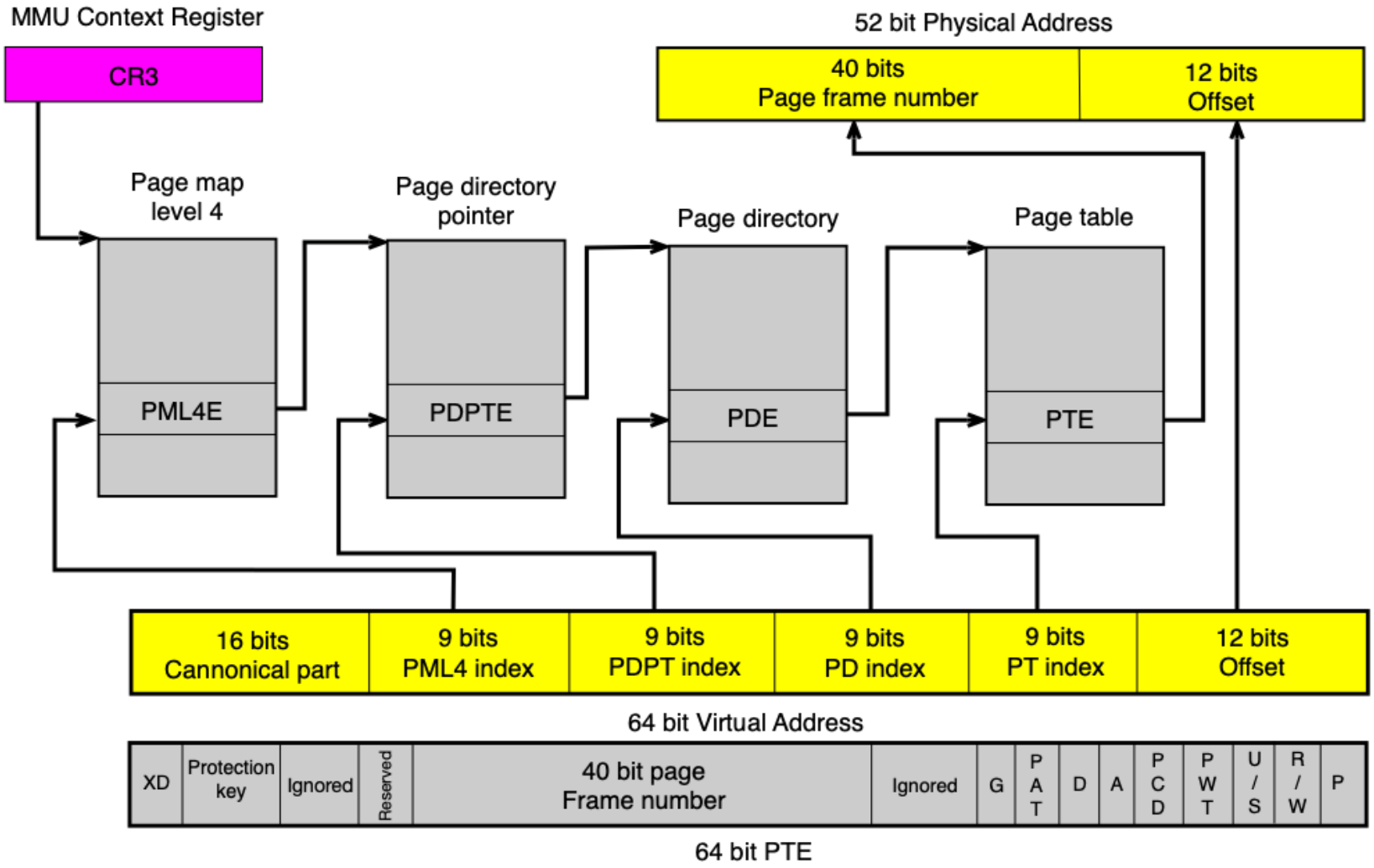
Víceúrovňová tabulka stránek

- Příklad dvouúrovňové tabulky stránek procesoru x86.



Víceúrovňová tabulka stránek

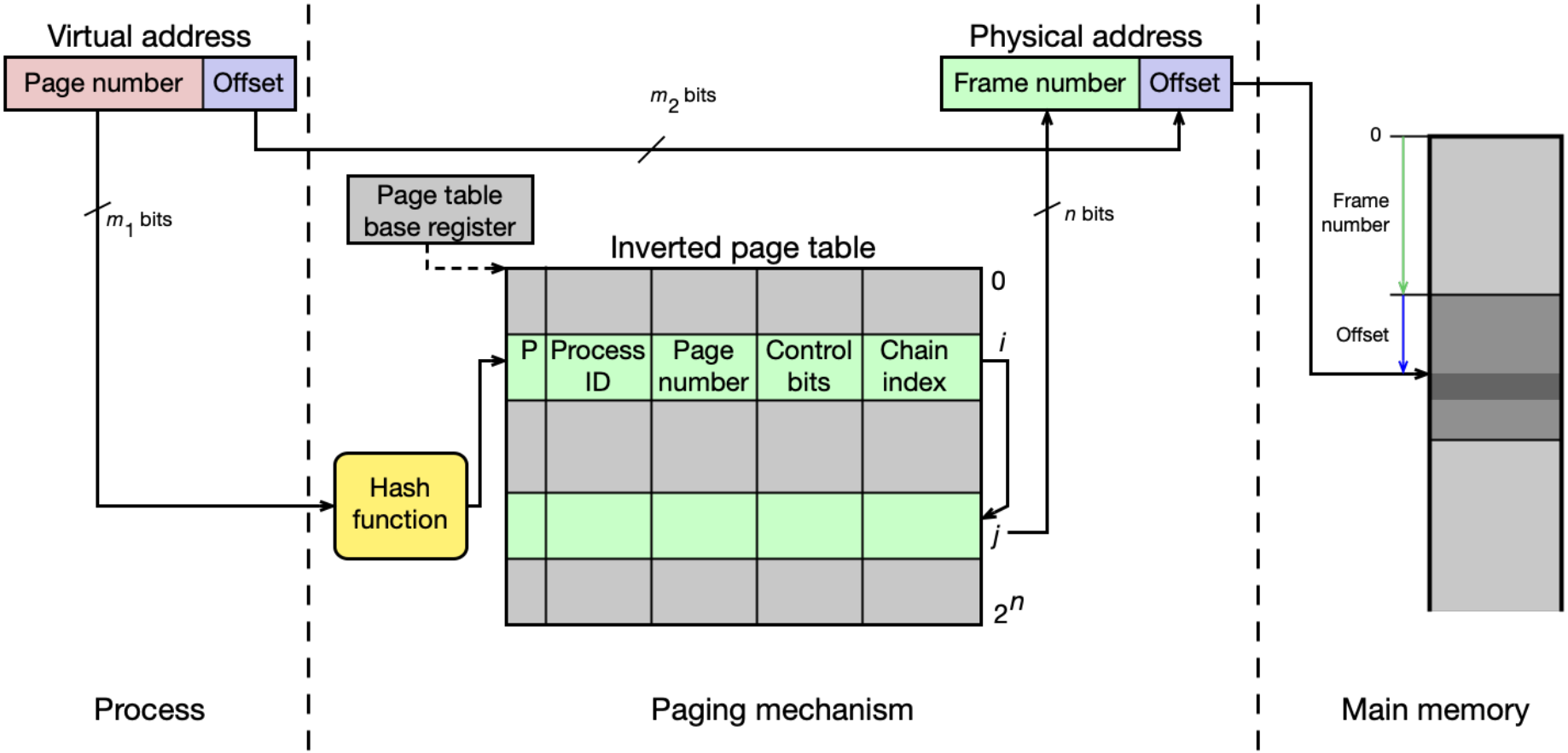
- Příklad čtyřúrovňové tabulky stránek procesoru x86-64.



Invertovaná tabulka stránek

- Tabulka obsahuje pro každý rámec fyzické paměti jednu řádku, ve které jsou uložené následující informace
  - **číslo stránky**, která je nahrána do tohoto rámce,
  - **číslo procesu**, do jehož VAS tato stránka patří,
  - **kontrolní bity**
    - ★ Present bit (P): informace, zda stránka je v hlavní paměti,
    - ★ Reference bit (R): informace, zda se ke stránce přistupovalo,
    - ★ Modify bit (M): informace, zda byl obsah stránky modifikován,
    - ★ Přístupová práva
    - ★ Cache disable/enabled: zda jsou registry periferií mapovány přímo do paměti, ...
  - **index zřetězení (chain)** .
- OS si musí udržovat pouze jednu tabulku pro celý systém.
- Číslo stránky (page number) se pomocí hašovací funkce přepočte na index do tabulky. Protože počet stránek je větší než počet rámců fyzické paměti, několik různých stránek se může namapovat na stejnou řádku v tabulce $\Rightarrow$ proto ze používá technika zřetězení.

Invertovaná tabulka stránek



Invertovaná tabulka stránek

- **Příklad**
  - Na 32-bitovém CPU, které podporuje pouze 4KB stránky/rámce, je nainstalovaný 32-bitový OS, ve kterém běží 32-bitový proces. Systém používá invertovanou tabulku stránek a lze na něm vytvořit maximálně 2^{10} prosesů.
  - Předpokládejme, že proces bude alokovat ve svém VAS pouze následující datové struktury
    - ★ TEXT a DATA: 4 MB na nejnižších virtuálních adresách,
    - ★ halda: 4 MB na následujících virtuálních adresách,
    - ★ zásobník: 4 MB na nejvyšších virtuálních adresách.

Invertovaná tabulka stránek

- **Příklad**
  - Na 32-bitovém CPU, které podporuje pouze 4KB stránky/rámce, je nainstalovaný 32-bitový OS, ve kterém běží 32-bitový proces. Systém používá invertovanou tabulku stránek a lze na něm vytvořit maximálně 2^{10} prosesů.
  - Předpokládejme, že proces bude alokovat ve svém VAS pouze následující datové struktury
    - ★ TEXT a DATA: 4 MB na nejnižších virtuálních adresách,
    - ★ halda: 4 MB na následujících virtuálních adresách,
    - ★ zásobník: 4 MB na nejvyšších virtuálních adresách.
- **Jaká bude struktura virtuální a fyzické adresy?**
  - Virtuální adresa (32 bitů): číslo stránky (20 bitů) + offset (12 bitů).
  - Fyzická adresa (32 bitů): číslo rámce (20 bitů) + offset (12 bitů).
- **Kolik řádek bude mít tabulka stránek?**
  - Pro každý rámec musí být jedna řádka $\Rightarrow 2^{20}$ řádek.
- **Kolik místa zabírá jedna řádka?**
  - Číslo procesu (10) + číslo stránky (20 bitů) + kontrolní bity (P+R+M+...) + chain index (20) $\Rightarrow$ zaokrouhlíme na celé bytes/slova ~64 bitů = 8 B.
- **Kolik místa budou všechny tabulky zabírat pokud běží v systému 2^7 podobných procesů?**
  - $1 \times 2^{20} \times 8$ B = 8 MB.

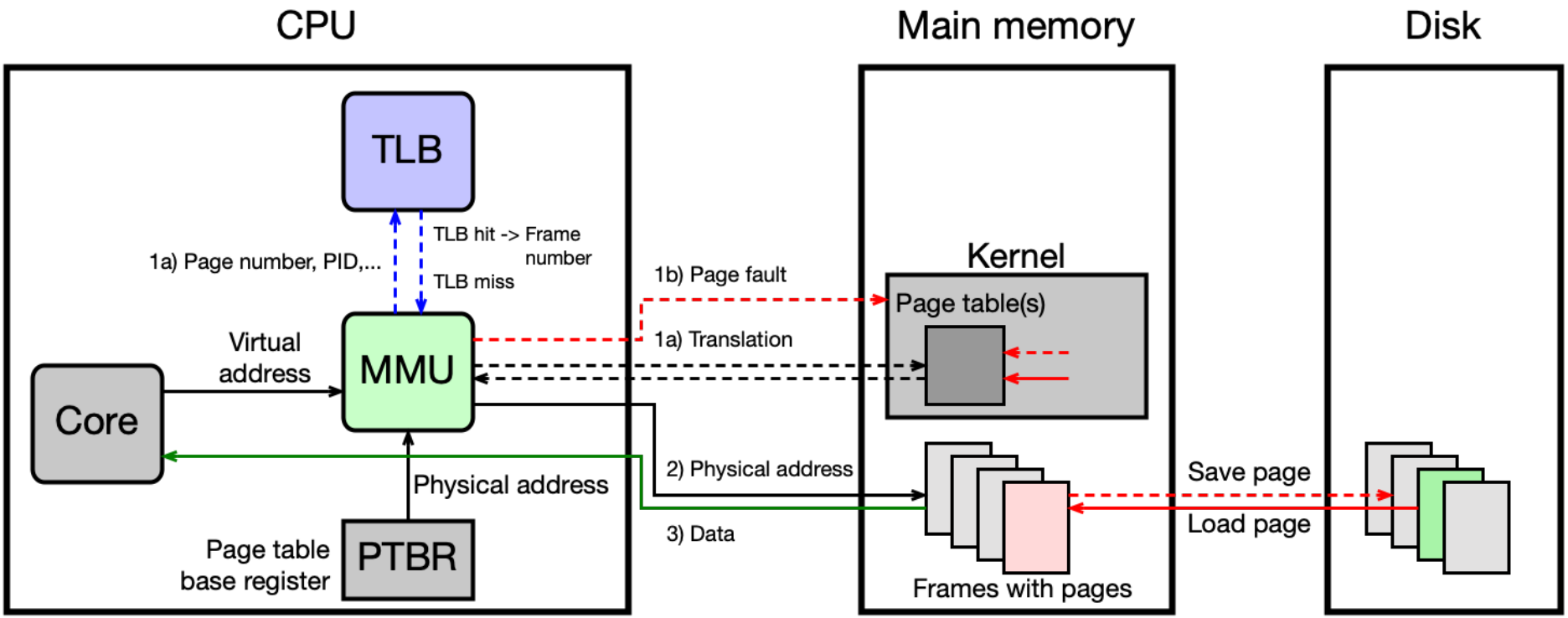
Invertovaná tabulka stránek

- Invertovaná tabulka stránek se používala/používá v procesorech PowerPC a UltraSPARC.
- Zabírá méně místa než tabulka stránek, ale hledání v této tabulce je pomalé. $\Rightarrow$ pro urychlení překladu se používá společně s TLB.

Translation lookaside buffer (TLB)

- Za účelem urychlení překladu se v procesorech používá "Translation lookaside buffer" (TLB).
- TLB je v praxi implementovan jako *n*-way cache (paměť s omezeným stupně asociativity).
- Obsahuje informace o posledně překládaných virtuálních adresách (číslo stránky–číslo rámec).
- Počet položek TLB je výrazně menší než počet virtuálních stránek/počet fyzických rámců.
- Položka TLB obsahuje
  - valid bit: zda je platná položka,
  - číslo stránky,
  - číslo rámce,
  - ASID: address space ID (identifikátor adresního prostoru),
  - control bits,...

Translation lookaside buffer (TLB)



Translation lookaside buffer (TLB)

- Moderní CPU cca od roku 2000 podporují současné používání různě velkých virtuální stránek/fyzický rámců.
- V některých OS (např. Solaris, AIX) je tato funkcionalita podporována implicitně, v jiných OS je nutné tuto vlastnost povolit.
- **Příklad**
  - Pomocí příkazu `pagesize` můžeme v Solarisu zjistit jaké velikosti stránek CPU podporuje (velikost je v B).

```
user@solaris:~ > pagesize -a
8192
65536
4194304
268435456
```

Translation lookaside buffer (TLB)

- **Příklad**
  - Pomocí příkazu `pmap` můžeme v Solarisu zjistit jaké velikosti stránek jsou požívané ve VAS procesu s PID=7291.

```
user@solaris:~ > pmap -s 7291
7291:   -bash
         Address       Bytes Pgsz Mode    Mapped File
0000000100000000        960K  64K r-x----  /usr/bin/bash
00000001000F0000         40K   8K r-x----  /usr/bin/bash
00000001001FA000         48K   8K rwx----  /usr/bin/bash
0000000100300000         16K   8K rw-----  /usr/bin/bash
000000D1A2D6A000         24K   8K rw-----    [ heap ]
000000D1A2D70000        256K  64K rw-----    [ heap ]
00007FA191B00000        192K  64K r-x----  /lib/sparcv9/libcurses.so.1
. . .
00007FA191F00000         64K  64K rwx----    [ anon ]
00007FA192000000         64K  64K rwx----    [ anon ]
00007FA19204C000          8K   8K rwxs---    [ anon ]
00007FA192100000         24K   8K rwx----    [ anon ]
00007FA192200000        384K  64K r-x----  /lib/sparcv9/libc.so.1
00007FA192380000          8K   8K r-x----  /lib/sparcv9/libc.so.1
00007FA192486000         64K   8K rwx----  /lib/sparcv9/libc.so.1
00007FA192496000          8K   8K rwx----  /lib/sparcv9/libc.so.1
00007FA192500000         64K  64K rw-----    [ anon ]
00007FA192530000          8K   8K r--s---    [ anon ]
00007FA192600000        256K  64K r-x----  /lib/sparcv9/ld.so.1
00007FA192640000         16K   8K r-x----  /lib/sparcv9/ld.so.1
00007FA192744000         24K   8K rwx----  /lib/sparcv9/ld.so.1
FFFFFDC097960000         24K   8K rw-----    [ stack ]
         total         3848K
```

Translation lookaside buffer (TLB)

- Pokud aplikace používá/alokuje velkou část ze svého VAS, pak může docházet k velké frekvenci TLB miss (časté přepisování položek v TLB).
- CPU i OS nám umožňují sledovat jak využití TLB, tak i ostatních parametry "Memory managementu".
  $\Rightarrow$ na základě těchto informací můžeme změnit nastavení OS/chování aplikací (např. pro některé oblasti VAS (třeba pro "Heap") můžeme nestavit používání stránek větších velikostí).

Translation lookaside buffer (TLB)

- **Příklad**
  - Pomocí příkazu `trapstat` můžeme v Solarisu zjistit využití TLB.
    - u = user mode/k = kernel mode
    - itlb miss= instruction TLB miss / dtlb miss = data TLB miss
    - %tim = percentage of CPU time in miss handler
    - TSB = Translation Storage Buffer (varianta invertované tabulky stránek)

```
root@solaris:~ > trapstat -t
        cpu m| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
        -----+-------------------------------+-------------------------------+----
          0 u|      2571  0.3         0  0.0 |     10802  1.3         0  0.0 | 1.6
          0 k|         0  0.0         0  0.0 |    106420 13.4       184  0.1 |13.6
        -----+-------------------------------+-------------------------------+----
          1 u|      3069  0.3         0  0.0 |     10983  1.2       100  0.0 | 1.6
          1 k|        27  0.0         0  0.0 |    106974 12.6        19  0.0 |12.7
        -----+-------------------------------+-------------------------------+----
          2 u|      3033  0.3         0  0.0 |     11045  1.2       105  0.0 | 1.6
          2 k|        43  0.0         0  0.0 |    107842 12.7       108  0.0 |12.8
        -----+-------------------------------+-------------------------------+----
          3 u|      2924  0.3         0  0.0 |     10380  1.2       121  0.0 | 1.6
          3 k|        54  0.0         0  0.0 |    102682 12.2        16  0.0 |12.2
        -----+-------------------------------+-------------------------------+----
          4 u|      3064  0.3         0  0.0 |     10832  1.2       120  0.0 | 1.6
          4 k|        31  0.0         0  0.0 |    107977 13.0       236  0.1 |13.1
        =====+===============================+===============================+====
         ttl |     14816  0.3         0  0.0 |    585937 14.1      1009  0.0 |14.5
```

Použité zdroje

1. A. S. Tanenbaum, H. Bos: *Modern Operating Systems (4th edition)*, Pearson, 2014.
2. W. Stallings: *Operating Systems: Internals and Design Principles (9th edition)*, Pearson, 2017.
3. A. Silberschatz, P. B. Galvin, G. Gagne: *Operating System Concepts (9th edition)*, Wiley, 2012.
4. R. McDougall, J. Mauro: *Solaris Internals: Solaris 10 and OpenSolaris Kernel Architecture (2nd edition)*, Prentice Hall, 2006.
5. *Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4*, 2021.