2. Paměťová konzistence a předbíhání operací čtení a zápisu, podpora virtuálního adresového prostoru.

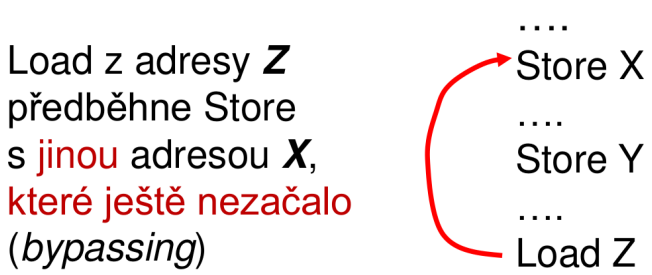
Paměťová konzistence a předbíhání operací čtení a zápisu

Striktné dodržiavanie programového poradia čítaní a zápisov má negatívny dopad na výkon CPU. Moderné CPU používa cache na vyrovnanie priepastných rozdielov medzi rýchlosťou vykonávania operácií a rýchlosťou pamäte. Ak by sme teda striktne dodržiavali poradie čítaní a zápisov, tak by výpadok D-cache pri vykonávaní load znamenal zastavenie vykonávania všetkých následných inštrukcií manipulujúcich pamäť a pridružených závislých inštrukcií. Medzi load a store inštrukciami na rovnakú adresu vznikajú obdobné závislosti (RAW, WAR, WAW) ako pri Out-of-Order (OoO) vykonávaní inštrukcií. Tieto závislosti je nutné rešpektovať, aby bola sémantika programu zachovaná, no ich detekcia je možná až v dobe, kedy sú spočítané adresy pamäťových operandov. Ak medzi load/store inštrukciami nie sú závislosti (tj. nemanipulujú s rovnakou adresou), je možné ich vykonávať mimo programového poradia. Rozlišujeme:

- RPR - Read can pass Read
- **RPW - Read can pass Write**
- WPR - Write can pass Read
- WPW - Write can pass Write

Read can pass write

- HW v podstate vykonáva rozbalenie smyčky, pri ktorom sa môžu iterácie čiastočne prekrývať – load (načítanie dát pre nasledovnú iteráciu) môže predbehnúť store aktuálnej iterácie
- Hlavný zdroj výkonnosti
- 2 spôsoby akým môže load predbehnúť store:

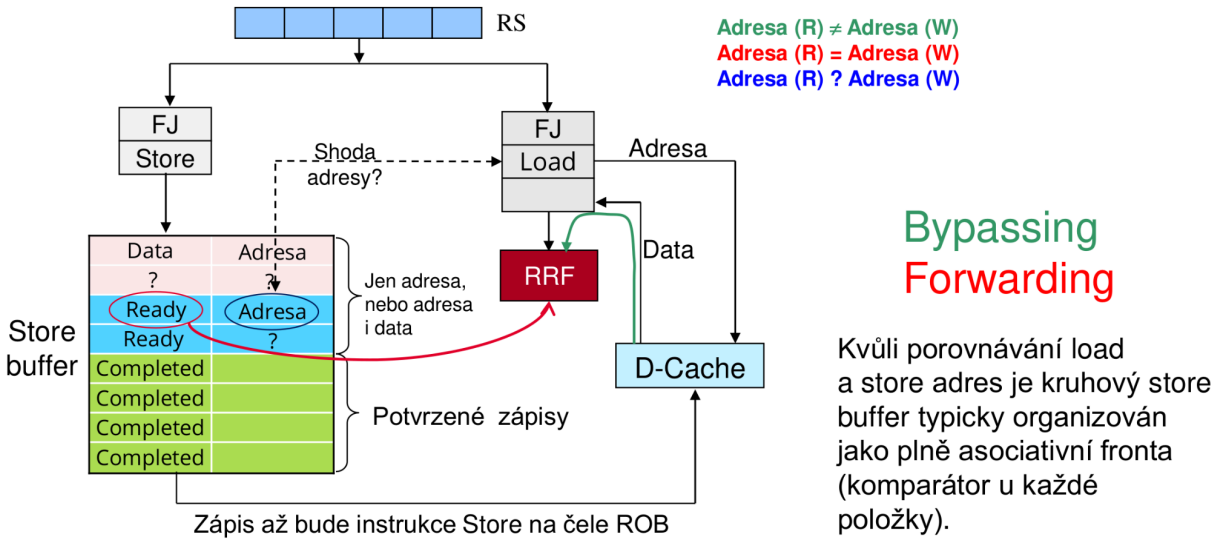


Load načte data z ještě nedokončené instrukce Store se stejnou adresou ***X*** (load from store *forwarding*, předávání).

- aby mohol load predbehnúť store, je potreba vedieť, či je medzi load (R) a store (W) inštrukciami závislosť, tj. vedieť či `address(R) == address(W)`, čo vedie na dynamické rozlišovanie adries (memory disabiguation):
  - ak `address(R) != address(W)`, tak sa nejedná o konflikt RAW, a je možné aby load predbehol write
  - ak `address(R) == address(W)`, tak load môže načítať dáta z nedokončeného write (uloženého v store buffer)
  - ak je odpoveď neznáma, čakaj, alebo špekulatívne predpokladaj, že `address(R) != address(W)`

Store jednotka

- ukladá rozpracované zápisy do store buffer, ich adresy sú v store bufferu v programovom poradí
- položky môžu byť v nasledovných stavoch:
  - *available* - položka je voľná
  - *adr only* - zatiaľ je známa len adresa
  - *ready* - dáta aj adresa sú známe
  - *completed* - zápis je potvrdený, čaká sa na prepustenie z čela ROB. Keď sa príslušná store inštrukcia dostane na čelo ROB, tak sa zápis vykoná do D-cache a položka prejde do stavu *available*.
- pri spracovaní nových load inštrukcií sa ich adresy konzultujú s adresami v store bufferu na zhodu:



Load jednotka

- ak nie sú známe niektoré store adresy, tak je nutné špekulovať, a teda následne špekuláciu overiť
- adresy dokončených špekulatívnych load inštrukcií sú uložené v load jednotke
- každá dokončená store inštrukcia skontroluje adresy v load jednotke na zhodu:
    - špekulovali sme, že load načíta z adresy, na ktorú iná inštrukcia nezapisuje. Ak dôjde ke zhode, je nutné takto invalidovaný load a všetky závislé inštrukcie zrušiť a výpočet opakovať
    - ak k žiadnej zhode nedošlo až dokiaľ sa korešpondujúca load inštrukcia nedostane na čelo ROB, tak sa vykoná prepis z RRF (register rename file) do ARF (architectural rename file)

Pamäťová konzistencia

Zmeny poradí zápisov a čítaní s cielom zvýšiť výkon má nepríjemné následky v multicore prostredí, pretože poradie, v akom jednotlivé vlákna/procesy vidia zmeny hodnôt pamäťových lokácií je základ synchronizačných prostriedkov. Hypotetický príklad, nijak nesúvisí s realitou:

> *Posledný termín skúšky z predmetu IOS bude podľa plánu v D105. Po katastrofálnej havárii osvetlenia 24h pred skúškou, bolo nutné zmeniť učebňu na E102. Garant predmetu preto vykoná zmenu miestnosti v informačnom systéme, a pošle email všetkým registrovaným. Chvíľu však trvá, než zmena učebne prebublá ISom, a teda aktívny študenti, ktorý si po prečítaní email-u skontrolujú termín uvidia, že skúška je stále v D105. Všimnite si, že garant (producent) vykonal zápis v správnom poradí (update shared resource then notify consument) ale študenti (consuments) videli email (resource is available) no v skutočnosti shared resource nebol aktualizovaný.*

Pamäťová konzistencia pojednáva o **poradí** v akom procesory pozorujú čítania a zápisy **na rôzne adresy.**

Typy pamäťovej konzistencie:

- **Sekvenčná** - čítania a zápisy sa vykonávajú v programovom poradí pozorovateľným všetkými procesormi rovnako
- **Relaxovaná** - čítania a zápisy sa môžu predbiehať, dokiaľ to nemení správnosť programu

Dôsledky relaxovanej pamäťovej konzistencie

- Programátor musí explicitne stanoviť kedy je poradie operácií dôležité – použitím např. OpenMP direktívou `flush`, premennými typu `volatile` apod.; ostatné inštrukcie sa môžu vykonávať mimo poradia
- Výhodou je teda väčší výkon, jednoduchší hardware; nevýhodou väčšia záťaž pre programátora a možnosť záludných chýb

Pamäťové zábrany (memory barriers, fences)

- fences = špeciálne inštrukcie, ktoré zabraňujú presunom load a store inštrukcií tam, kde je to nežiadúce
- typy bariér:
    - *plná zábrana* - všetky loads a stores (L/S) sa pred fence musia dokončiť, až potom môžu začať L/S nachádzajúce sa za bariérov
    - čiastočné zábrany - obmedzujú len čítanie alebo len písanie
    - jednosmerné zábrany - dvojica acquire (bráni presunu L/S nahor) a release (bráni presunu L/S nadol). Sémantika odpovedá implementácii binárneho zámku.

Podpora virtuálního adresového prostoru

Benefity virtualizácie:

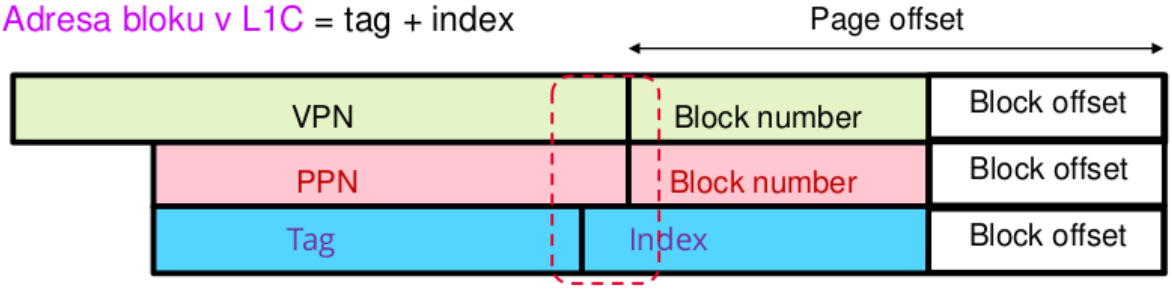
- Aplikácie nepotrebujú spravovať adresový priestor (AP)) zdieľaný viacerými procesmi – virtualizácia vytvára ilúziu, že celý AP patrí procesu
- Zvýšenie bezpečnosti vďaka izolácii AP aplikácií
- Možnosť zdieľať zdieľané knižnice v rámci procesov
- Možnosť zdanlivého použitia väčšieho množstva pamäte ako je na systéme nainštalované

Pri virtualizácii rozlišujeme:

- Virtuálnu adresu (VA) s ktorou pracuje program
    - VA = Virtual page number (VPN) + page offset
- Fyzickú adresu (PA) ktorá je vystavená na zbernicu
    - PA = Physical page number (PPN) + page offset
- Page offset = block number (číslo bloku na stránke) + block offset (číslo slova v bloku)
- Naviac cache sú adresované pomocou tag + offset

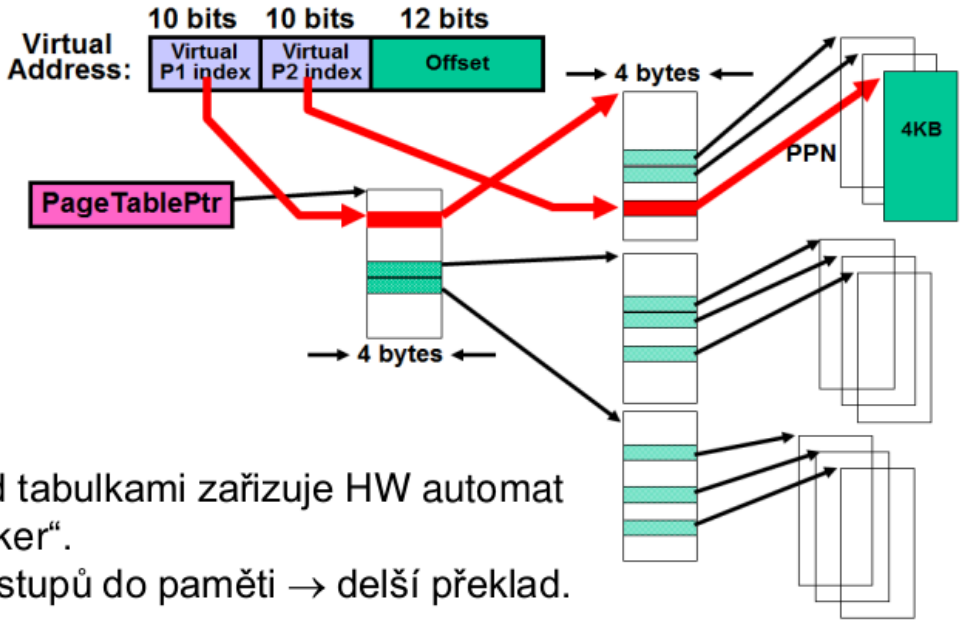
Adresa bloku v paměti = PPN + Block number

Adresa bloku v L1C = tag + index



Tabulky stránek (Page Table, PT)

- položky (Page Table Entry, PTE) ukladajú mapovanie VPN na PPN
- PT je uložená v hlavnej pamäti
- Ak máme 32b systém s veľkosťou stránky 4kB, tak veľkosť tabuľky stránok je
  - (počet adries) / (koľko bytov je adresovateľných v jednej stránke) = 2^32 / (4*2^10) = 2^32 / (2^12) = 2^20 PTEs. Ak uvážime veľkosť jednej PTE 4B dostávame 2^20 *4 = 2^22 = 4MB
- Pri 64b adresovom priestore je veľkosť tabuľky neúnosná
- Pre veľké pamäťové nároky plochých (flat) tabuliek sa používajú viacúrovňové (x86_64 používa 4 úrovne) tabuľky stránok, ktoré využívajú to, že obsadenie virtuálneho AP je riedke. Príklad 2-úrovňovej tabulky:



- Dnes je možné použiť aj väčšie ako štandardne používané 4kiB stránky a redukovať tak množstvo TLB miss (popísaný ďalej) ak má aplikácia veľký working set – napr. Linux podporuje tzv. huge pages s veľkosťou 2048 kB). Použitie väčších stránok však nemá iba výhody v podobe menšieho počtu TLB miss:

- Zvyšuje sa interná fragmentácia
- Pri výpadku stránky je nutné načítať viac dát z disku (dlhšia doba čakania pri výpadku)

Translation Lookaside Buffer

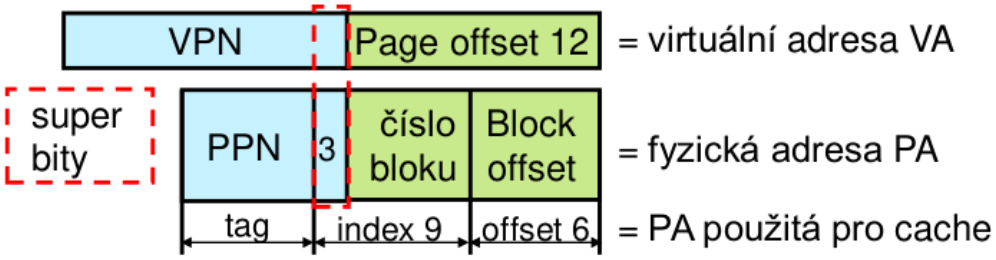
- prechod (viacúrovňovou) tabuľkou stránok je časovo náročný, preto sa používa malá rýchla pamäť cache Translation Lookaside Buffer (TLB) ukladajúca posledné preklady VPN na PPN
- TLB je teda plne asociatívna tabuľka dvojíc (VPN, PPN), typicky o veľkosti 32-128 položiek
- môže byť rozdelená na inštrukčnú a dátovú časť
- TLB spravuje buť HW, alebo SW (operačný systém)
- cena prístupu do pamäte:
    - zásah v TLB aj L1C (Level 1 Cache) - 1-3 takty
    - zásah v TLB, výpadok v L1C, zásah v L2C - 8-12 taktov
    - zásah v TLB, výpadok v L1C, výpadok v L2C, zásah v Main memory - 75-250 taktov
    - výpadok v TLB, zásah v PT - 2000 taktov
    - výpadok v TLB, výpadok v PT (page fault) - až 10^8 taktov

Virtuálna pamäť a Level 1 Cache

- Spolupráca CPU s Level 1 Cache (L1C) vyžaduje čo najrýchlejší prístup do cache
- Je potrebné dospieť od VA k adrese bloku (index, tag) v L1C:
    - na adresáciu je možné použiť:
        - VA - je k dispozícii ihneď
        - PA - je k dispozícii až po preklade
- Tri možnosti realizácie, na základe toho, či sa použie VA, PA alebo hybridný prístup:
    - P/P cache: fyzický index, fyzický tag
    - V/V cache: virtuální index, virtuální tag
    - V/P cache: virtuální index, fyzický tag - virtuálny index se dá použiť hneď, paralelne s prekladom VA

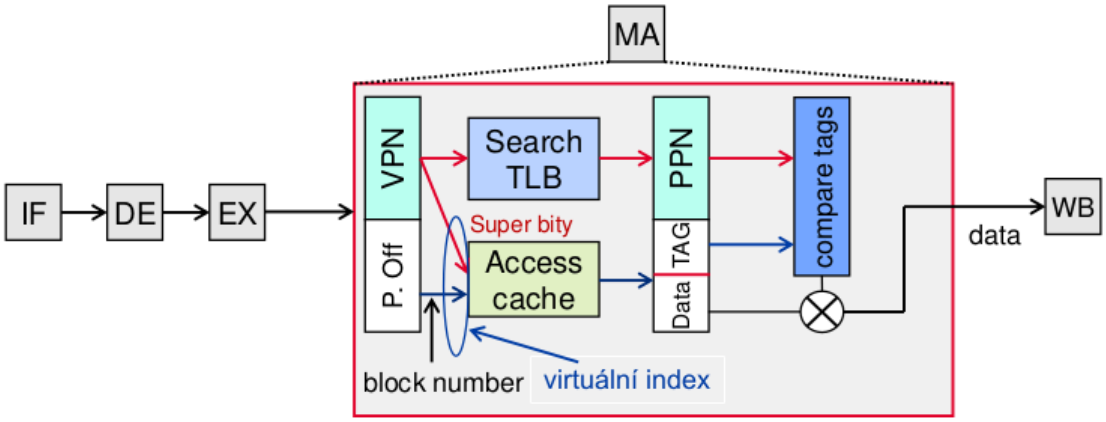
Fyzicky adresované P/P cache

- Sériový prístup = najprv preklad TLB (při výpadku až PT), potom L1C lookup je pomalý
- Paralelný prístup je možný:
    - Ak sa index kryje s číslom bloku - nie je potreba index prekladať (page offset, ktorého je index súčasťou sa neprekladá), avšak kapacita cache je obmedzená.
    - Ak sa spodné bity (tzv. superbity) VPN a PPN použité v indexu zhodujú (tj. pri preklade sa nemenia). Takúto zhodu môže zariadiť OS.



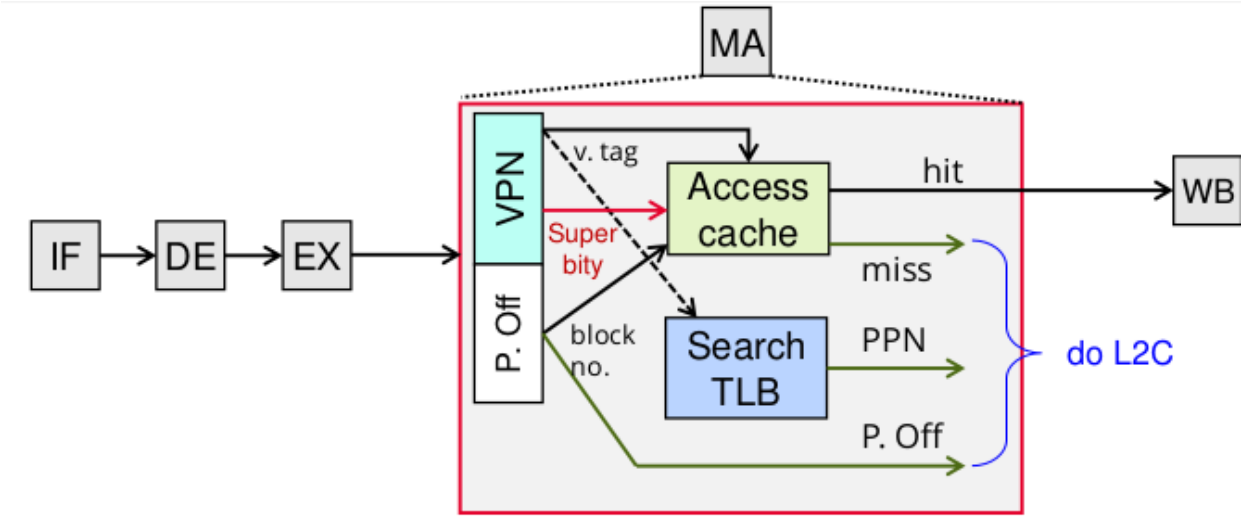
V/P cache – cache adresovaná virtuálnym indexom a fyzickým príznakom (tagom)

- Indexovanie L1C môže začať hneď, spolu s prekladom TLB. Pre následné porovnanie sa použije porovnanie tagov z L1C s tagom získaným prekladom z TLB
- Vniká problém synoným - 2 alebo viac VA z TLB sa preloží na rovnaký fyzický index



V/V cache (virtuálny index aj tag)

- Na indexáciu sa použije tzv virtuálny index = block number + spodná časť VPN (tzv. virtuálne superbity). Pri zásahu v L1C preklad nie je potreba - TLB sa neuplatní.
- Výstup TLB sa použije len v prípade výpadku L1C, teda málo kedy. Na lookup v L2C sa použije výsledok TLB, teda získané PPN.
- Problémy:
  - TLB sa pri zásahoch v L1C neuplatní - je potreba kopírovať do L1C réžijné bity stránky
  - Okrem synonym vznikajú homonymá - rovnaká VA sa mapuje na rôzne PA (prepínanie kontextov)


MA
IF
DE
EX
VPN
P. Off
v. tag
Super
bity
block
no.
Access
cache
hit
WB
miss
Search
TLB
PPN
P. Off
do L2C