Otázka: Cache paměť

Předmět: Informační technologie

Přidal(a): Maturant

RWM paměť

- Dělí se na DRAM a SRAM

DRAM – dynamická RAM

- Levnější, ale za to pomalejší. Zapsaná hodnota je uchována jako náboj na **parazitní kapacitě tranzistoru** MOSFET. Je nutné provádět pravidelně refresh.
- Cenově vychází kolem **250Kč za GB**

SRAM – statická RAM

- Výrazně rychlejší, ale také dražší. Pro uchování hodnoty je použit **bistabilní klopný obvod** z minimálně 6 tranzistorů, z toho plyne výrazně vyšší cena.
- Cenově vychází kolem **10 000 až 100 000 Kč za GB**

Využití

- Statická se z ekonomických důvodů nemůže používat jako operační paměť počítače, používá se však jako cache paměť u procesorů.

Cache paměť

- Také označována jako vyrovnávací paměť je velmi rychlá **SRAM paměť** sloužící ke zrychlení toku dat mezi procesorem a operační pamětí.
- Poprvé se začínají objevovat u procesoru **Intel 80386**. Rychlost procesoru je větší než rychlost pamětí. Procesor by uměl vykonat mnoho instrukcí pro čtení a zápis do paměti, ale paměť nestíhá a trvá jí příliš dlouho operace s daty. Proto cache paměť řeší **nesoulad mezi rychlostí procesoru a operační pamětí**.
- Jejím účelem je urychlit přístup k **často používaným datům** v operační paměti překopírováním do rychlejší cache paměti. Kvůli malé velikosti cache paměti nelze uchovat celý obsah RAM. Nejlépe by měla být uložena data ke kterým se **přistupuje nejčastěji**. Musí tedy existovat mechanismus výběru adres které budou do cache zkopírovány. Zároveň musí existovat mechanismus, který umožní procesoru okamžitě zjistit zda adresa ke které se chce přistupovat je v cache dostupná, či nikoliv.

L1, L2, L3

- Externí cache paměť je paměť mezi operační pamětí a procesorem. První externí paměť se **objevuje u 80386. Z důvodu že rychlost procesoru začala být větší než operační paměti, dříve by neměla cenu**
- Později se cache paměť integruje spolu s mikroprocesorem **v jednom pouzdř**e. První interní paměť se **objevuje u 80486**.

- U moderních procesorů může být cache **dvoustupňová** nebo až **třístupňová.**
- **L1 (Level 1 cache)** s malou kapacitou je přímo součástí procesoru a je stejně rychlá jako vlastní procesor.
- **L2 (Level 2 cache)** je mezi procesorem a operační pamětí (dnes již v pouzdře procesoru) a je rychlostí přizpůsobená vnější sběrnici, avšak s větší kapacitou než L1.

Nejmodernější vícejádrové procesory používají **třístupňovou cache**.

- Každé jádro má svojí L1 cache a větší, ale pomalejší L2 cache. Všechna jádra mikroprocesoru pak sdílí dohromady velikou (obvykle v řádech MB) L3 cache (pomalejší než L1 a L2, ale levnější než L2)
- Moderní procesory mají oddělenou cache pro instrukce a pro data
- *Příklad cache u procesoru **i7-7600U** mobile verze*
    - ***L1** – 2×32 KB pro instrukce a 2×32 KB pro data*
    - ***L2** – 256 KB*
    - ***L3** – 4 MB*

Zápis do cache

- Když dojde k zaplnění paměti a je potřeba zavést další blok, je nutné aby některý z bloků pamět opustil a uvolnil tak místo.
- Tento problém řeší tyto strategie
    - LRU (Least Recently Used) – Používá se algoritmus který odstraní nejdéle nepoužívaný blok.
    - MFU (Most Frequently Used) – Kdy se ponechávají často používané položky a položka nejméně používná se ruší

potřeba si pamatovat rozdíl mezi nejméně používanou a nejdéle nepoužívanou

- RAND – náhodný výběr oběti
- FIFO (First In First Out) – odebrána je ta položka, která je v paměti nejdéle

Při zápisu do cache mohou vznikat problémy, nově zapsaný bajt by měl být zapsán jak v cache tak v RAM.

Write-Through

- **Zápis skrze cache**, nejstarší a **nejpomalejší** způsob. Data se zapisují současně do cache i do RAM.

Write-Back

- **Opožděný zápis**, data jsou zapisována do operační paměti až ve chvíli, **kdy je to třeba**, a ne okamžitě při jejich změně.
- Data jsou zapsána pouze do cache a teprve při odstranění z cache (např. LRU strategií) jsou provedené změny zapsány do operační paměti. Než se data dostanou do operační paměti, mohou několikrát změnit svoji hodnotu – při tom se tedy mění pouze hodnota cache a stav původního bajtu v operační paměti je neaktuální (to nás netrápí, aktuální se hledají nejdříve v cache). Tento způsob práce cache paměti vykazuje oproti předešlému způsobu **vyšší výkon**.

Princip fungování cache

- Cache paměti jsou organizované jako **tzv. asociativní paměti**, vyhledává se podle obsahu a ne podle adresy.
- K datům v cache paměti se nepřistupuje přes adresu jako u běžné paměti.

- Asociativní paměti jsou tvořeny **tabulkou**, která obsahuje vždy sloupec, v němž jsou **umístěny tzv. klíče**, podle kterých se v asociativní paměti vyhledává.
- Každý záznam v paměti cache se tedy skládá **z klíče**, **vlastních dat** a dalších několika řídících bitů.
- Při přístupu do paměti je nutné zadat adresu, kterou hledáme. Tato adresa buď celá nebo několik jejich bitů je **tzv. vstupní klíč.** Pokud je v paměti nalezena položka jejíž klíč se shoduje se vstupním klíčem, jsou v paměti požadovaná data.
- Představit se to dá tak že v paměti se vyhledává na **základě shody klíčů**. Klíčem je část nebo celá adresa na které mají data ležet v operační paměti. Dochází tedy k hledání zda v cache je obsah této adresy.
- Hledání není sekvenční, ale paralelní, proběhne okamžitě naráz. Každý „řádek“ paměti obsahuje **komparátory** pro okamžité porovnání všech bitů klíče.

Plně asociativní cache

- Jako **klíč** je brána celá adresa, ze které se bude číst. Každý řádek paměti má vlastní komparátor.

Příklad primitivní plně asociativní cache s kapacitou 8 bajtů

- Jako **klíč** tady slouží celá adresa.
- Tento způsob je neefektivní protože je tady moc dlouhý klíč, který zabírá místo v paměti. Data jsou moc krátká, takže jich nelze tolik uchovat a je potřeba hodně komparátorů, který na chipu zabírají místo pro paměť (pro každej řádek jeden).

Klíč	Data
12345h	A7h
2A4D1h	FFh

FF2C5h	14h
145ADh	BCh
75683h	11h
A1122h	22h
71243h	5Ch

Příklad plně asociativní cache se 4 bajtovými bloky a 7bitovým klíčem

- Data se v cache ukládají po blocích protože je velká pravděpodobnost že budou také potřeba. Zároveň to také zkracuje délku vstupního klíče.
- Pokud bychom hledali data z adresa **101110000**. Jako vstupní klíč se použije horních 7 bitů a vyberou se adresy začínající těmito sedmi bity, které končí 00 až 11, tedy 1011100**00** až 1011100**11**.

Příklad plně asociativní cache se 4 bajtovými bloky a 8bitovým klíčem

- V tomto případě funguje cache podobně, ale má 6bitový klíč. To znamená že okolní vybrané adresy budou končit v rozmezí 000 až 111. V příkladu to znamená že když hledáme podle klíče **101110101.** Jako vstupní klíč se použije horních 6 bitů, tedy 101110 a vyberou se adresy 101110**000** až 101110**111**.

Přímo mapovaná cache

- V přímo mapované cache je možno každý obraz z určité adresy v RAM uložit pouze na jedno konkrétní místo. Kapacita cache určuje tedy počet **tříd**, např. kapacita 1024 položek bude znamenat 1024 tříd. Požadovaná data se tedy vždy hledají na

konkrétním místě (řádku), stačí tedy jeden komparátor.
- Vstupní hledaná adresa se rozdělí na klíč a třídu. Podle třídy se vybere konkrétní řádek a poté se porovnávají klíče, pokud se klíč shoduje, žádaná položka se v cache nachází.
- **Třída** jsou tedy bity adresy, které říkají na jakém řádku by mohla informace v cache ležet.
- **Klíčem** jsou zbývající bity adresy, které je potřeba otestovat na shodu.
- Čím větší je třída a data, tím se zmenšuje klíč.

Vícecestná cache

- Je organizována podobně jako přímo mapovaná cache, s tím rozdílem jakoby zde bylo několik takových pamětí paralelně.
- Stupeň asociativity se značí **n** nebo **a**. Adresa třídy je přivedena na **n** dekodérů které v každé tabulce vyberou stejný řádek.
- **N-cestně** asociativní paměti částečně eliminují nevýhody plně asociativních cache pamětí a v současnosti jsou **nejpoužívanějším typem cache pamětí**.

Příklady

- Cache s kapacitou 1MB a blokem dat 8B
- Velikost klíče, třídy, dat a počet komparátorů?

<table>
<tr>
<td>Přímo mapovaná
1MB = 2^{20}
8B = 2^{3}</td>
<td>4 cestná
4 tabulky po 256kB
256kB = 2^{18}
8B = 2^{3}
Třída = 2^{18}-2^{3} = 2^{15}
Data = 2^{3}
4 komparátory (každá tabulka jeden)
<table>
<tr><td>14b</td><td>15b</td><td>3b</td></tr>
<tr><td>klíč</td><td>třida</td><td>data</td></tr>
</table>
</td>
<td>Plně asociativní
Data = 8B = 2^{3}
2^{17} komparátorů (2^{20}-2^{3})
<table>
<tr><td>29b</td><td>0b</td><td>3b</td></tr>
<tr><td>klíč</td><td>třída není</td><td>data</td></tr>
</table>
</td>
</tr>
</table>

Určit kapacitu cache podle parametrů

- 14b klíč, 16B blok, 4 cestná
  - Data jsou 16B = 2^{4}, takže 4b. Klíč je 14b, pro třídu tedy zbývá 14b. Velikost jedné tabulky je 2^{14}x2^{4}=2^{18} podle třídy a dat. Tabulky jsou 4 = 2^{2}. Velikost cache je tedy 2^{18}x2^{2} = ^{20} = 1MB
- 16b klíč, 8B blok, 2 cestná
  - Data jsou 8B = 2^{3}, takže 3b. Klíč je 16b, pro třídu tedy zbývá 13b. Velikost jedné tabulky je 2^{13}x2^{3}=2^{16} podle třídy a dat. Tabulky jsou 2 = 2^{1}. Velikost cache je tedy 2^{16}x2^{1} = 2^{17} = 128kB
- 15b klíč, 16B blok, 4 cestná
  - Data jsou 16B = 2^{4}, takže 4b. Klíč je 15b, pro třídu tedy zbývá 13b. Velikost jedné tabulky je 2^{13}x2^{4}=2^{17} podle třídy a dat. Tabulky jsou 4 = 2^{2}. Velikost cache je tedy 2^{17}x2^{2} = 2^{19} = 512kB
- **Při vylosování této otázky dostanete na papírku vytištěný nějaký příklad ne vícecestnou cache** – dle zadání nakreslíte několik tabulek (podle toho kolikacestná bude) a vypočítáte počet řádků, délku bloku, šířku klíče atd. – vše nakreslíte během přípravy