Rychlá vyrovnávací paměť v architektuře PC

Cíl přednášky

- Prezentovat důvody, které vedly k zavedení rychlé vyrovnávací paměti (RVP) do architektury počítače.
- Vysvětlit principy činnosti RVP.
- Ukázat vývoj architektur RVP.
- Současný stav.

Důvody pro zařazení RVP (cache) do sestavy personálního počítače

- Procesory synchronizované vysokým kmitočtem potřebují spolupracovat s dostatečně rychlými pamětmi s krátkou vybavovací dobou.
- DRAM v době I80386 měly vybavovací dobu 60 - 100 ns, doby cyklu byly delší => do architektury počítače bylo nutno vložit paměťový prvek s kratší vybavovací dobou.
- Řešení tohoto problému představují prvky SRAM s dobou cyklu 10 - 20 ns (platilo v době I80386).

Vybavení personálních počítačů rychlou vyrovnávací pamětí – základní úvahy

- V době, kdy se objevil procesor I80386, existovala jistá technologie podpůrných komponent, paměťové prvky a jejich parametry byly důležité.
- Paměti: široká škála různě rychlých (z hlediska vybavovací doby) paměťových prvků DRAM.
- Začátek existence procesoru I80386 + odhad doby, po niž bude vyráběn (jednotky roků) + odhad rychlosti, na niž se procesor na konci „svého života“ dostane (MHz) + odhad vývoje technologie pamětí (stagnace ve zdokonalování parametrů prvků DRAM) => bylo nutno se zabývat úvahou, zda paměti budou rychlostně stačit.

Vybavovací doba prvků DRAM

- **Doba cyklu** - doba, za niž je možné generovat nový požadavek na paměťovou operaci.

  Sestává:

  z vybavovací doby,

  z doby potřebné na ustálení přechodových dějů na sběrnicích (data, adresa).
- Typické vybavovací doby [ns] paměťových prvků instalovaných v PC v době, kdy se na trhu objevil procesor I80386: 250, 150, 120, 85, 70, 60, 50 ns (byly používány ty nejrychlejší, tzn. 50 – 70 ns).
- PC na bázi I80286 - vybavovací doba paměťových prvků pod 100 ns (začátek existence I80286).
- Konec éry procesoru I80286 - vybavovací doba prvků DRAM 60 - 80 ns.

Stavy čekání (Wait State)

- **Byl požadavek, aby paměť byla schopna na požadavek od CPU reagovat během dvou taktů**: 8 MHz - takt 125 ns

  20 MHz - 50 ns

  33 MHz - 30 ns
- Pokud není požadavek realizován během 2 taktů, tak nastává další fáze komunikace – vkládání čekacích stavů.

  Stav čekání - jeden takt/více taktů synchronizačních pulsů přidaných navíc do komunikace s pamětí nebo řadičem periferního zařízení.

  Pozn.: pro tyto účely je vybavena i systémová sběrnice (viz ISA, PCI)
- Příklad:

  Uvažujeme o možnosti navrhnout PC s hodinami 10 MHz a paměťovými čipy 256 kB, 120 ns (toto byly úvahy na konci existence PC na bázi I80286).

  Synchronizace 10 MHz => bylo nutno udržet cyklus paměti pod 2 x 100 = 200ns.

  Doba pro ustálení přechodových dějů pro tyto čipy - 90 ns

  => doba cyklu = 120 + 90 = 210 ns - **paměti jsou o 10 ns pomalejší než potřebujeme**.

Řešení

- **Řešení:**

  zpomalit procesor (krok zpět),

  přidat stavy čekání (krok zpět),

  zrychlit paměť DRAM (může být drahé a nerealizovatelné – dáno úrovní technologie),

  doplnit architekturu o další paměťový prvek, který bude rychlejší než paměť DRAM.

Stav technologie v době I80386

Vybavovací doba [ns]	Doba cyklu [ns]
200	370
150	270
120	210
100	175
80	145
60	105
53	95

DC – doba cyklu

f [MHz]	takt [ns]	DC1 [ns], 0 čekacích stavů (2 takty)	DC2 [ns], 1 čekací stav (3 takty)	Prvky s vybav. dobou [ns]	Prvky s vybav. dobou [ns]
8	125	250	375	120	200
10	100	200	300	100	150
20	50	100	150	53	80
25	40	80	120	není	60
33	30	60	90	není	není

Obdobnou úvahu bylo nutno zrealizovat, když se rozhodovalo o zařazení další (rychlejší) RVP – tzn. L1.

Stav technologie v době I80386

- Kmitočet procesoru 10 MHz, požadujeme, aby komunikace s pamětí probíhala bez čekacích stavů, tzn. doba cyklu musí být kratší než 200 ns (vybavovací doba + doba na ustálení přechodových dějů)

  => musíme zvolit paměťové čipy s vybavovací dobou 100 ns .

  Tabulka napovídá, že pro PC386/25 MHz resp. 33 MHz byly některé situace neřešitelné klasickými postupy (využitím DRAM).

  => **pro vyšší kmitočty neexistovala dostatečně rychlá DRAM**.

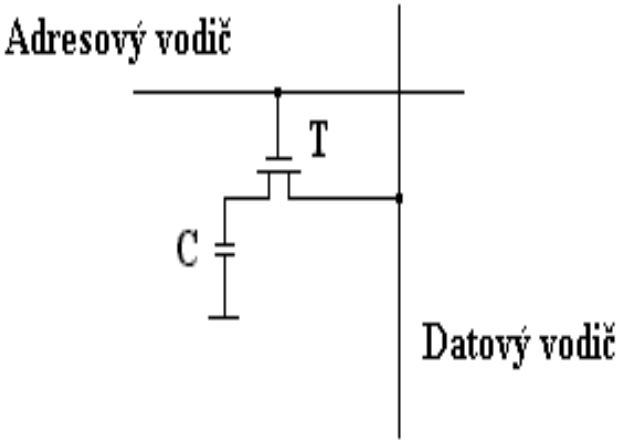
- **Řešení - paměť RVP komunikující s procesorem na vyšší frekvenci**.

RVP v personálních počítačích

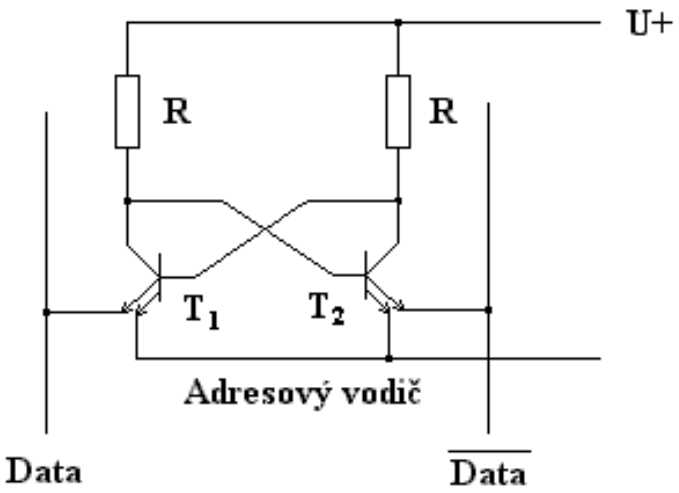
- Rychlá vyrovnávací paměť má výrazně menší kapacitu než paměť DRAM.
- Je realizována jako paměť SRAM (statická paměť RAM), je rychlejší => možnost komunikovat bez čekacích stavů. Cena/bit vyšší (obecně: čím menší kapacita, tím vyšší cena/bit), hustota menší (složitější paměťový prvek), vyšší příkon paměťového prvku.
- Snaha o integraci RVP do procesoru – poprvé u I80486.
- Dnes: L1, L2, L3

Opakování – realizace prvků DRAM a SRAM

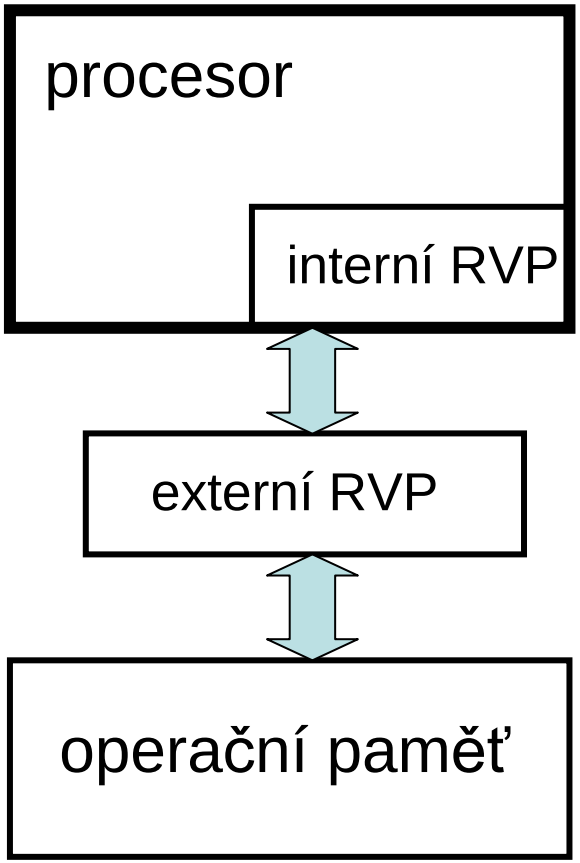
Prvek DRAM



Prvek SRAM



Hierarchie pamětí



Přístupy do paměti většinou probíhají tak, že jsou adresována paměťová místa, která následují bezprostředně za sebou => má smysl uchovávat bloky instrukcí/dat v RVP.

Závěr:

Kombinací paměti RVP typu SRAM a operační paměti typu DRAM se dosáhne výrazného snížení vybavovací doby a tím zvýšení rychlosti.

Principy spolupráce mezi procesorem a DRAM/SRAM

- Procesor žádá data z paměti - vloží na adresovou sběrnici adresu.
- Řadič paměti tuto adresu přijme a zahájí kroky k získání dat – zjistí se, zda jsou požadovaná data v RVP.
- Mohou nastat dvě situace: **cache hit** a **cache miss**.
- Cache hit - požadovaná data jsou v paměti RVP, řadič je přečte a předá je procesoru.
- To vše se odehraje bez čekacích stavů, tzn. maximální možnou rychlostí. Neproběhla žádná komunikace s operační pamětí.
- Cache miss - data nejsou v paměti RVP a musí být přečtena z RVP nižší úrovně/z operační paměti.

Principy spolupráce mezi procesorem a DRAM/SRAM

- Pokud nastane cache miss, pak musí řadič paměti RVP provést tyto činnosti:

  Přečíst z operační paměti celý řádek - cache line. Tato operace je označována jako **cache line fill**.

  Předtím je ale nutno nejprve **uvolnit v paměti RVP dostatečný prostor** – to je proces, jehož průběh musí být určen přesnými pravidly.

  Pokud se musí z paměti RVP odstranit data, která byla během předcházejících operací nějak modifikována, je nutno je nejprve přenést do operační paměti (předtím než se na toto místo v paměti RVP nahraje nový řádek).

- **Strategie, jimiž se řadič paměti RVP řídí**

  write-through

  write-back

  write-allocate

RVP typu Write-through

- Tato strategie je implementována nejčastěji.
- **Princip**:

  Všechny operace zápisu do paměti RVP se provedou také do operační paměti.

  => operace zápisu do RVP vede vždy k zápisu do operační paměti.

  Taková strategie by potenciálně mohla znamenat delší doby pro komunikaci s pamětí.

  Aby tomu tak nebylo, paměť RVP typu write-through využívá pro operaci zápis vyrovnávací paměť (buffer), operace zápisu do operační paměti se realizuje tak, aby nebyly zdržovány přenosy mezi procesorem a pamětí RVP.

Paměť RVP typu Write-back

- **Princip**:

  Při zápisu do paměti RVP se aktualizuje pouze obsah paměti RVP, obsah operační paměti se nemění.
- Pokud je nutné z paměti RVP odstranit řádky, přenesou se do operační paměti pouze ty řádky, které byly předtím nějak modifikovány (změněny).
- Nevýhoda: výměna dat ve srovnání s metodou Write-through trvá déle, protože před zápisem nových řádků se musí napřed modifikované řádky uložit do operační paměti.

Situace, které mohou nastat, když do operační paměti může zapisovat další prvek (např. řadič DMA)

- **Cache invalidation** - situace, když se data v operační paměti změní (při zápisu dat do operační paměti z periferního zařízení přes řadič DMA), takže data uložená v paměti RVP se stanou neplatná.

  Data z vnějšku mohou být důležitá např. v situacích, kdy je počítač v nějaké řídicí aplikaci a aktuální data přicházejí z vnějšku.

- **Cache flush** - situace, kdy má řadič DMA přenášet data z operační paměti do periferního zařízení, platná data jsou ale uložená v RVP.

Organizace paměti RVP

- Data a programy vytvářejí bloky => je pravděpodobné, že v dalším přístupu do paměti budou vyžadována data, která jsou součástí stejného řádku - zvyšuje se tak podíl hit cache.
- Řadiče RVP pracují v tzv. **nárazovém režimu** (burst mode) - čtou se celé bloky dat.
- Řádky paměti RVP (cache lines) mohou mít různou velikost, např. 16 byte, 32 byte, dnes výrazně více (čím větší kapacita RVP, tím větší cache line).
- **Cache line** – nejmenší jednotka, která je přenášena mezi pamětí a RVP.

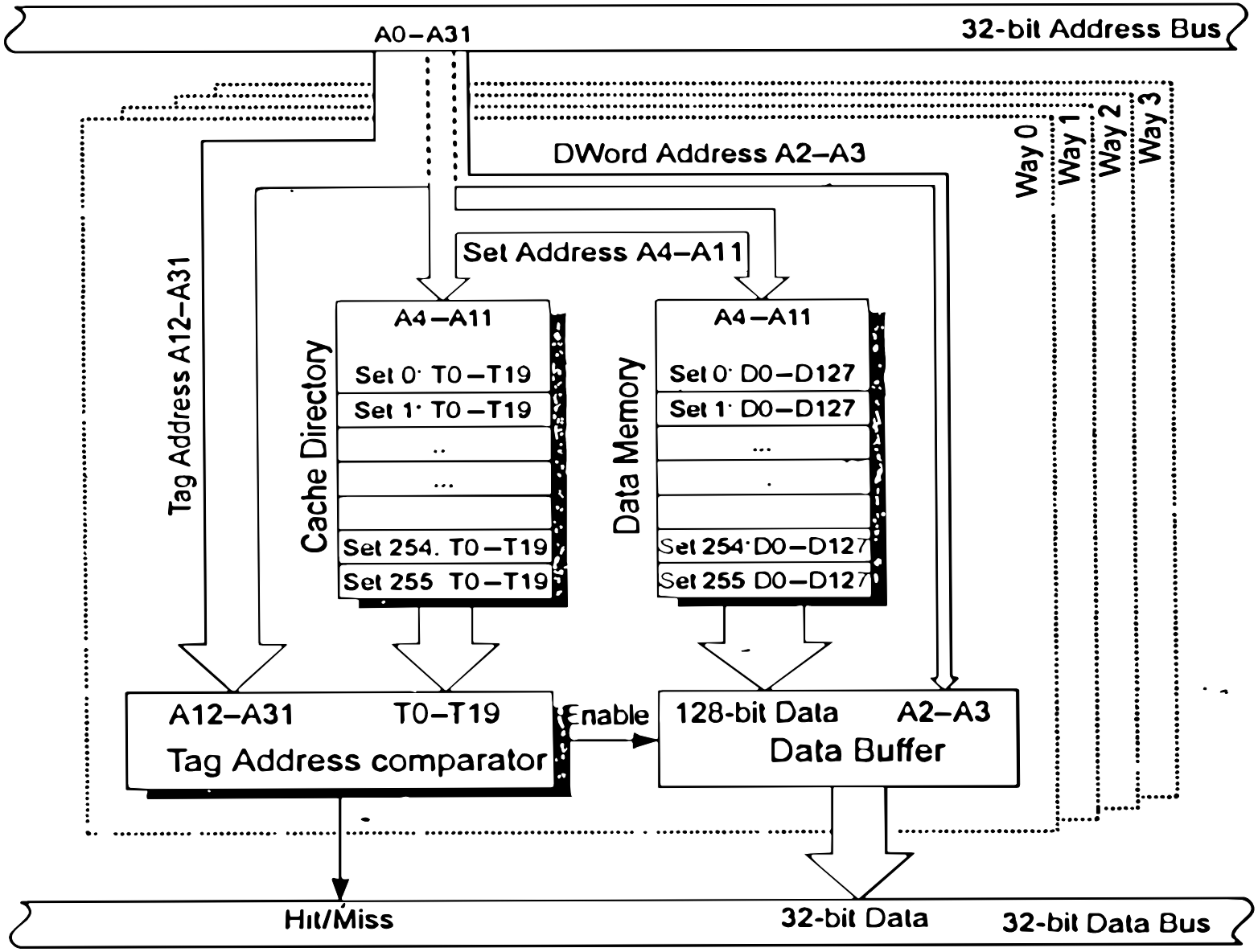
Organizace paměti RVP v PC na bázi procesoru I80386

- Řadič RVP rozděluje 32 bitovou adresu na 20 bitovou adresu tagu A12 - A31, 8 bitovou adresu sady (set) A4 - A11 a 4 bitovou adresu bytu A0 - A3.
- Bity A12 – A31 můžeme považovat za ukazatel na začátek bloku:

  při zápise informace do RVP zapíšeme tyto bity do Cache Directory na pozici danou obsahem bitů A4 – A11, do Data Memory zapíšeme na stejnou pozici data (pozice je dána také bity A4 – A11).
- Čtení dat:

  bity A12 – A31 z adresové části systémové sběrnice jsou vloženy na vstup komparátoru, na další vstup je přivedena informace z Cache Directory, současně jsou čtena data z Data Memory.

  Cache Hit – nastane shoda => **signálem Enable se otevře výstup Data Buffer na datovou sběrnici**.


A0–A31
32-bit Address Bus
DWord Address A2–A3
Way 0
Way 1
Way 2
Way 3
Set Address A4–A11
Tag Address A12–A31
Cache Directory
A4–A11
Set 0: T0–T19
Set 1: T0–T19
Set 254. T0–T19
Set 255 T0–T19
Data Memory
A4–A11
Set 0: D0–D127
Set 1: D0–D127
Set 254: D0–D127
Set 255 D0–D127
A12–A31
T0–T19
Tag Address comparator
Enable
128-bit Data
A2–A3
Data Buffer
Hit/Miss
32-bit Data
32-bit Data Bus

Organizace prvních typů RVP v PC na bázi procesorů Intel

- Adresa paměti RVP sestává z těchto částí:

  adresy položky v adresáři (cache directory entry) – obsah položky v adresáři reflektuje, zda jsou aktuálně adresovaná data k dispozici v RVP

  adresy položky v paměti RVP (cache memory entry)

- **Adresa položky v adresáři (Tag)**:

  Je to informace o tom, která data jsou v RVP uložená.

  První typy RVP (např. I80386) – tato informace byla uložena v externí paměti.

  V takovém případě bylo pro paměť RVP potřeba více čipů než odpovídá její kapacitě.

  Důvod je v tom, že část paměťových prvků musí uchovávat položky adresáře (tzv. Tag SRAM), v ostatních jsou uložena vlastní data (data SRAM).

  Tag SRAM by měla být rychlejší (tzn. kratší vybavovací doba).

Využití informace uložená v Tag SRAM

- Příklad: pokud v T0 – T19 jsou ve všech set uložena stejná čísla, pak v D0 – D127 (16 slabik) jsou ve všech set (4 kB) uložena data, která tvoří celek.
- Na obsah T0 – T19 se můžeme dívat jako na adresu bloku (stránky) o velikosti 4 kB, vlastní data jsou uložena v Data Memory (cache SRAM).
- Adresa položky v paměti - na této adrese jsou uložena vlastní data.

Tag

- Pomocí této položky řadič paměti zjišťuje, zda nastal **cache hit** nebo **cache miss** – obsah T0 – T19 je srovnáván s obsahem adresové sběrnice A12 – A31.
- V každé položce Tag je uloženo 20 bitů.
- Tag je platný pouze tehdy, když je nastaven **valid bit** (bit platnost). Pokud tomu tak není, pak jsou data uložená v patřičném řádku neplatná.

  Je-li nastaven bit **write protection**, pak řádek s takto nastavenou ochranou nemůže být přepsán.

Pojmy Set a Way

- **Set**

  Set je tvořen tagem uloženým v paměti tagů a odpovídajícím řádkem RVP (cache line) uloženým v paměti dat.

  Adresa setu sestává z 8 bitů A4 - A11 (256 možností).

- **Way**

  Paměť RVP sestává ze 4 ways. Pojem way vlastně představuje asociativnost paměti (čte se z té adresy, jejíž obsah v paměti tagů souhlasí s adresou zadanou z procesoru).

Další pojmy a principy

- Princip :

  Tagy uložené v paměti jsou srovnávány s tagy generovanými procesorem (vložené na adresovou sběrnici), výsledkem je cache hit nebo cache miss.

  => skupina dat odpovídající jednomu řádku může být v RVP uložena na jedné ze 4 různých posic (4 ways).

  Každá way obsahuje 256 sets, každá set obsahuje tag uložený v paměti tagů a řádek velikosti 16 slabik v paměti dat (hledaná data).
- Výpočet kapacity RVP:

  kapacita = počet way x počet sets x velikost řádku

  Pro i386 jsme tak dostali 16 kbyte kapacity RVP (way - 4, set - 256, řádek - 16 byte) .

Činnost řadiče RVP, když nastane cache hit

- Procesor vloží na adresovou sběrnici 32 bitovou adresu paměťového místa, z něhož mají být data čtena na adresovou sběrnici.
- Řadič paměti tuto kombinaci rozdělí na tyto části: tag, adresu setu a bytu.
- 20 bitová adresa tagu dělí 4 Gbytový adresový prostor procesoru i386 na 2^{20} stránek paměti RVP, každá kapacity 4 kbyte

  Paměť RVP sestává ze 4 way, v každé z nich se zjišťuje, zda adresa tagu souhlasí.
- Na této adrese přečte tag, ten se pak pošle do komparátoru tagu. Na jeho druhý vstup se přivádí tag z adresové sběrnice.
- Pokud nastane cache hit, je řádek přečtený z RVP řádkem hledaným.
- Z vyrovnávací paměti (buffer) se vybere podle hodnoty A2 - A3 double word (32 bitů).

  Tagy musí být k dispozici dříve než data, poněvadž přečtený tag musí být ještě srovnáván v komparátoru s tagem dodaným z procesoru

  v architektuře RVP by měly být prvky pro uložení Tag s kratší vybavovací dobou, ostatní čipy s vybavovací dobou delší.

Komparátor adresy tagu

- Komparátor srovnává adresu tagu z procesoru s adresou tagu přečtenou z RVP.
- Jestliže jsou tyto dvě adresy shodné, je aktivována vyrovnávací paměť (buffer) a data jsou k dispozici.
- Jestliže jsou tyto dvě adresy rozdílné, je výstup z bufferu zablokován, došlo ke cache miss => řadič RVP pak musí:

  adresovat operační paměť a z ní přenést data do procesoru,

  zahájit plnění řádku těmito daty v RVP.
- Všechny tyto operace, tzn.výběr a srovnání tagu se provádí stejným způsobem ve všech 4 way => jsou zapotřebí 4 komparátory, čtyři vyrovnávací paměti => jestliže RVP sestává z více way, počet těchto prvků se zvyšuje.
- Paměť RVP typu direct mapped cache (přímo mapovaná - paměť RVP s jednou way)) je nejjednodušší z hlediska konstrukce, ale má nejnižší počet úspěšných hledání (hit rate).
- Častější sestava, paměť RVP se 2 way je jistým kompromisem. Ve srovnání s pamětí RVP se 4 way má nižší úspěšnost hledání, ale logika je jednodušší.
- RVP cache s 8 way je zákonitě složitější, ale úspěšnost vyhledávání je vyšší.
- **Obecně: čím vyšší počet way, tím vyšší úspěšnost ale vyšší náklady**.

Příklad organizace RVP

- Příklad:

  RVP L2 kapacity 512 kbyte (I80486):

  organizovaná jako asociativní RVP se 2 way,

  s velikostí řádku 64 byte,

  měla celkem 8192 set (512k/64)

  s rozdělením adresy takto:

- adresa bytu A0-A5 (2^6 = 64), adresa setu A6-A18 (8192 sets), tag A19-A31.

Princip uvolňování obsahu RVP

- Bit LRU - Last Recently Used (ve smyslu „nejdéle nepoužívaný“).
- Strategie LRU:

  LRU bit B0 = 1, poslední přístup byl do way 0 nebo way 1:

  B1 = 1, poslední přístup byl do way 0

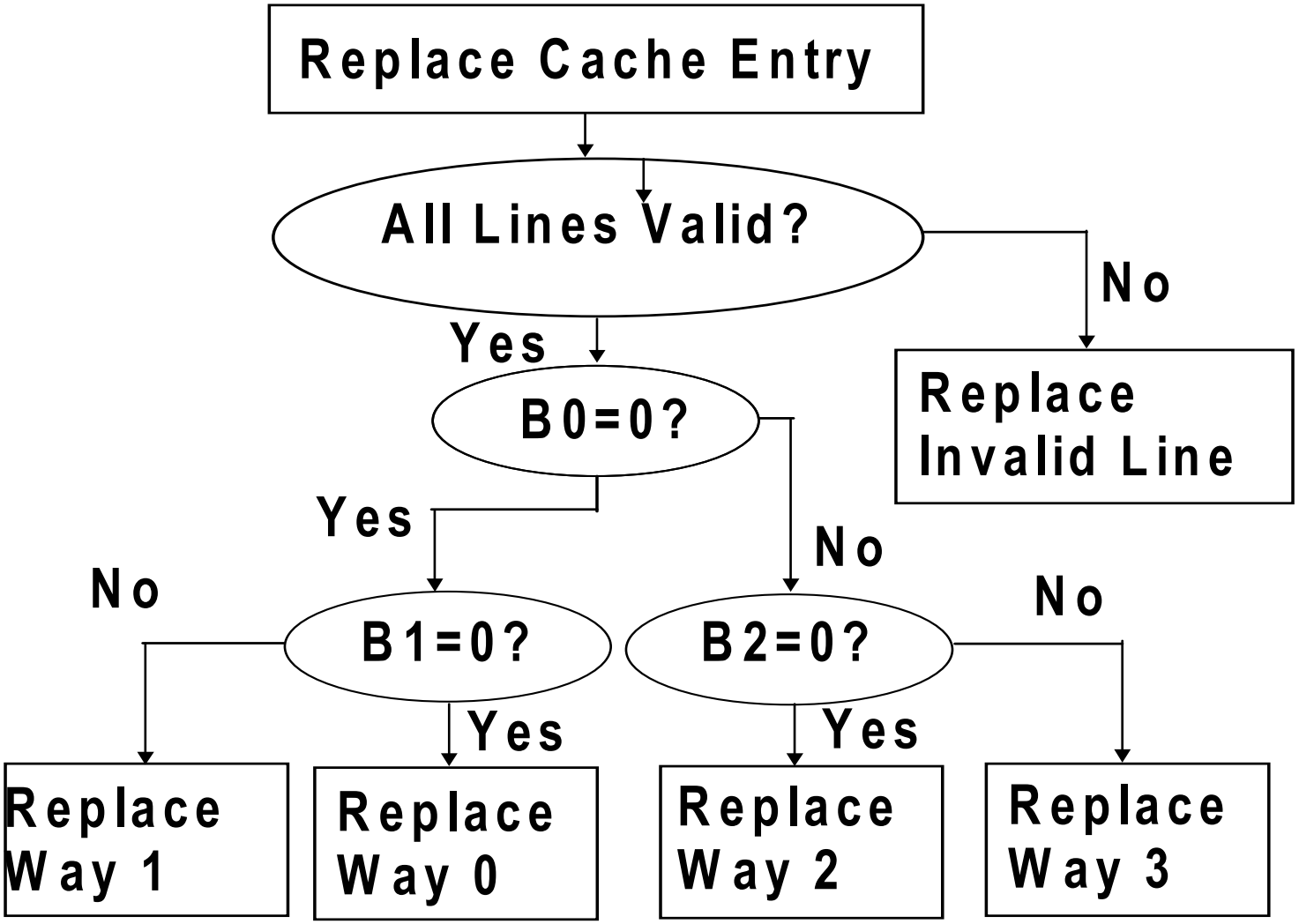
  B1 = 0, poslední přístup byl to way 1

  LRU bit B0 = 0, poslední přístup byl do way 2 nebo way 3:

  B2 = 1, poslední přístup byl do way 2

  B2 = 0, poslední přístup byl do way 3

Strategie výměny informace



Další vývoj

- i486 byl vybaven on-chip RVP pamětí (first level cache, L1 cache) kapacity 8 kB.
- Byla k dispozici také externí RVP (second-level cache, or L2-cache).
- Na první pohled se instalace RVP typu L1 kapacity 8 Kb mohla zdát nevýznamná.
- Přesto tato malá RVP L1 malé kapacity dovolila významným způsobem zvýšit výkon např. u procesorů DX2, kdy je synchronizována stejným kmitočtem jako procesor, tudíž 2x vyšším než systémová deska a RVP typu L2.

=> **vložení RVP na stejný čip jako procesor – významný krok vpřed.**

Mechanismus vyhledávání dat, když je nainstalována RVP L1 i L2

1. Jestliže procesor zjistí, že hledaná data nejsou v interní RVP paměti L1,
2. musí řadič paměti zjistit, zda tato data nejsou v externí RVP typu L2,
3. v případě, že nastane cache miss, přepne se požadavek na čtení na operační paměť.

- Rozdíl mezi RVP typu L1, resp. L2:

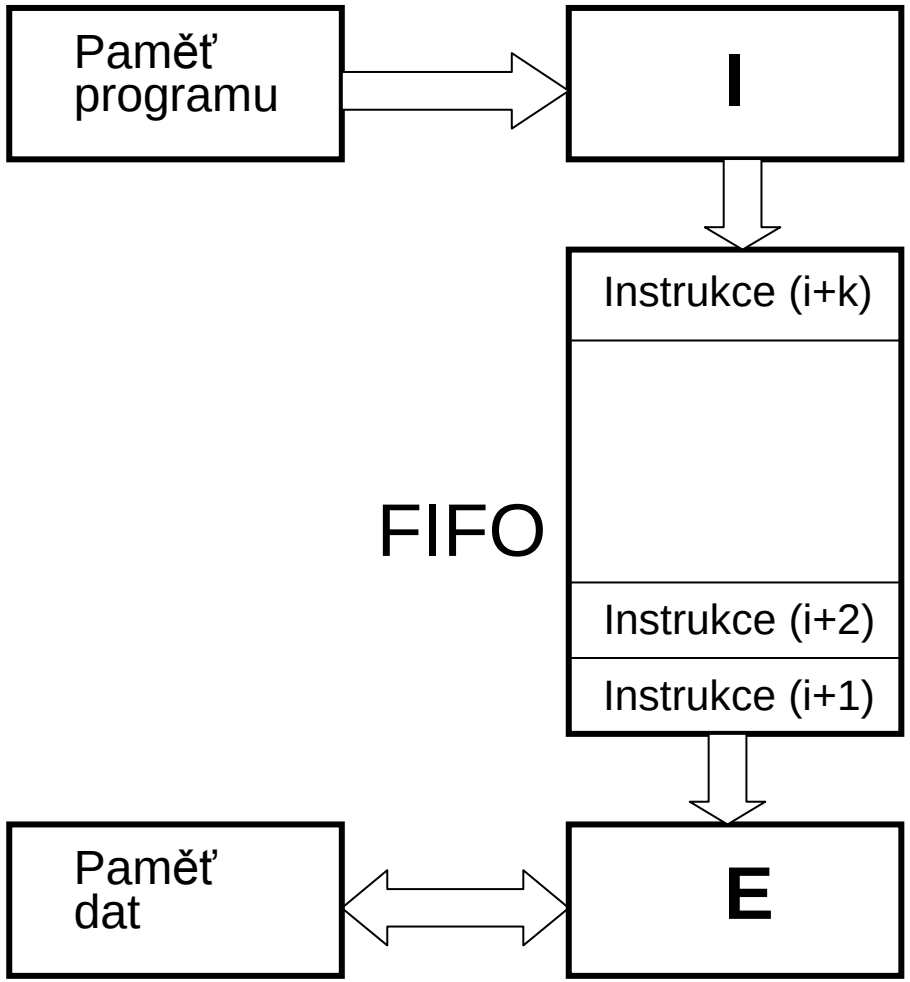
  V případě RVP L1 se data četla z RVP v cyklech sestávajících pouze z 1 synchronizačního pulsu, zatímco

  při přístupu do paměti L2 je nutné provést cyklus, sestávajícího ze 2 synchronizačních pulsů,

  => přístup do RVP L1 je rychlejší (platí to pochopitelně za předpokladu, že data jsou sekvenčně uložena v paměti).

Další vývoj v architekturách PC s RVP

Snahy o oddělení paměti programu od paměti operandů (**Harvardská architektura**).



Způsoby mapování

- 2 principy:

  přímé mapování (direct mapping)
  asociativní mapování (associative mapping)

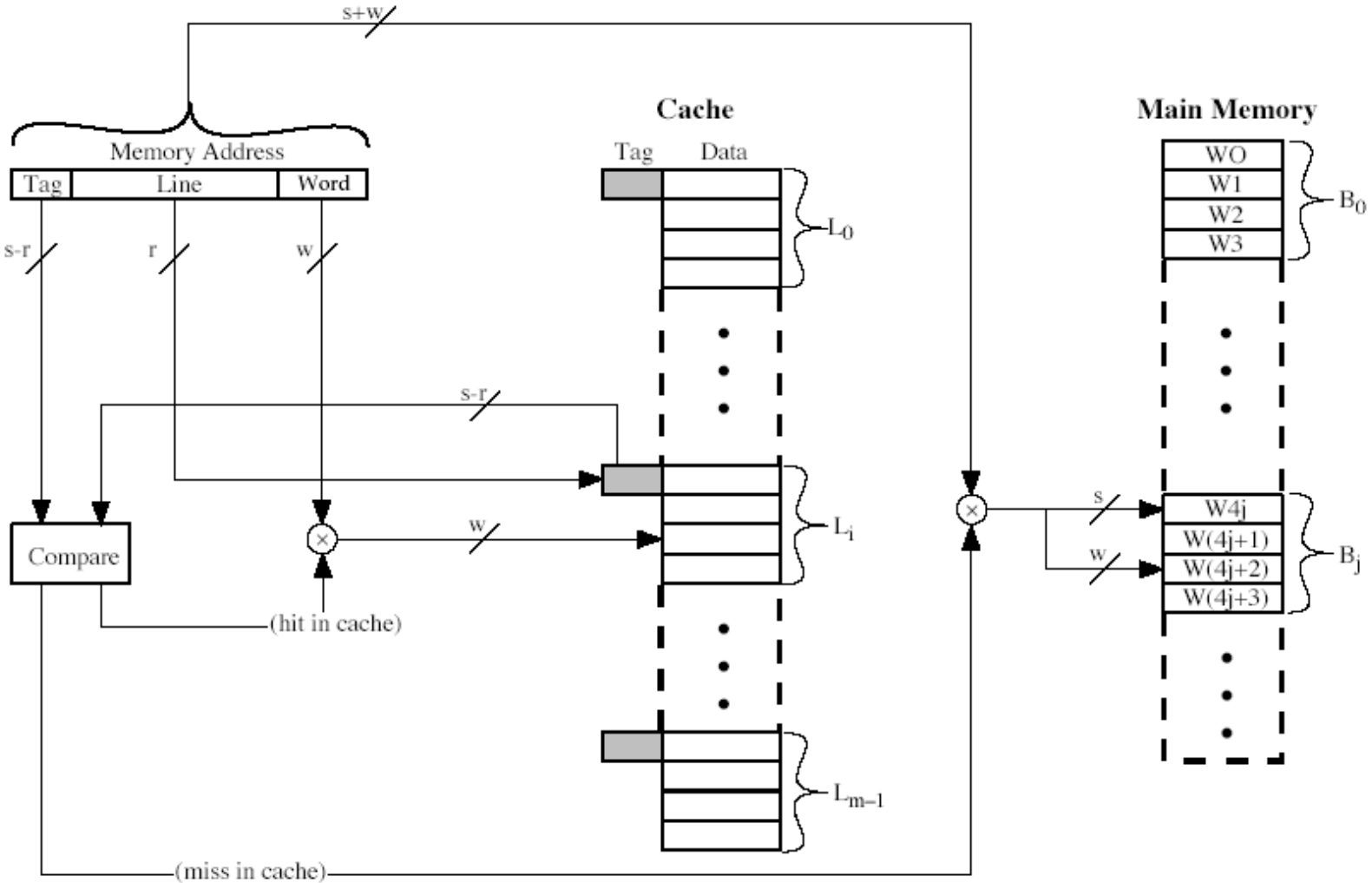
Přímé mapování

- Přímé mapování: každý řádek operační paměti se vkládá do stejného řádku RVP.
- Adresa sestává ze dvou částí:

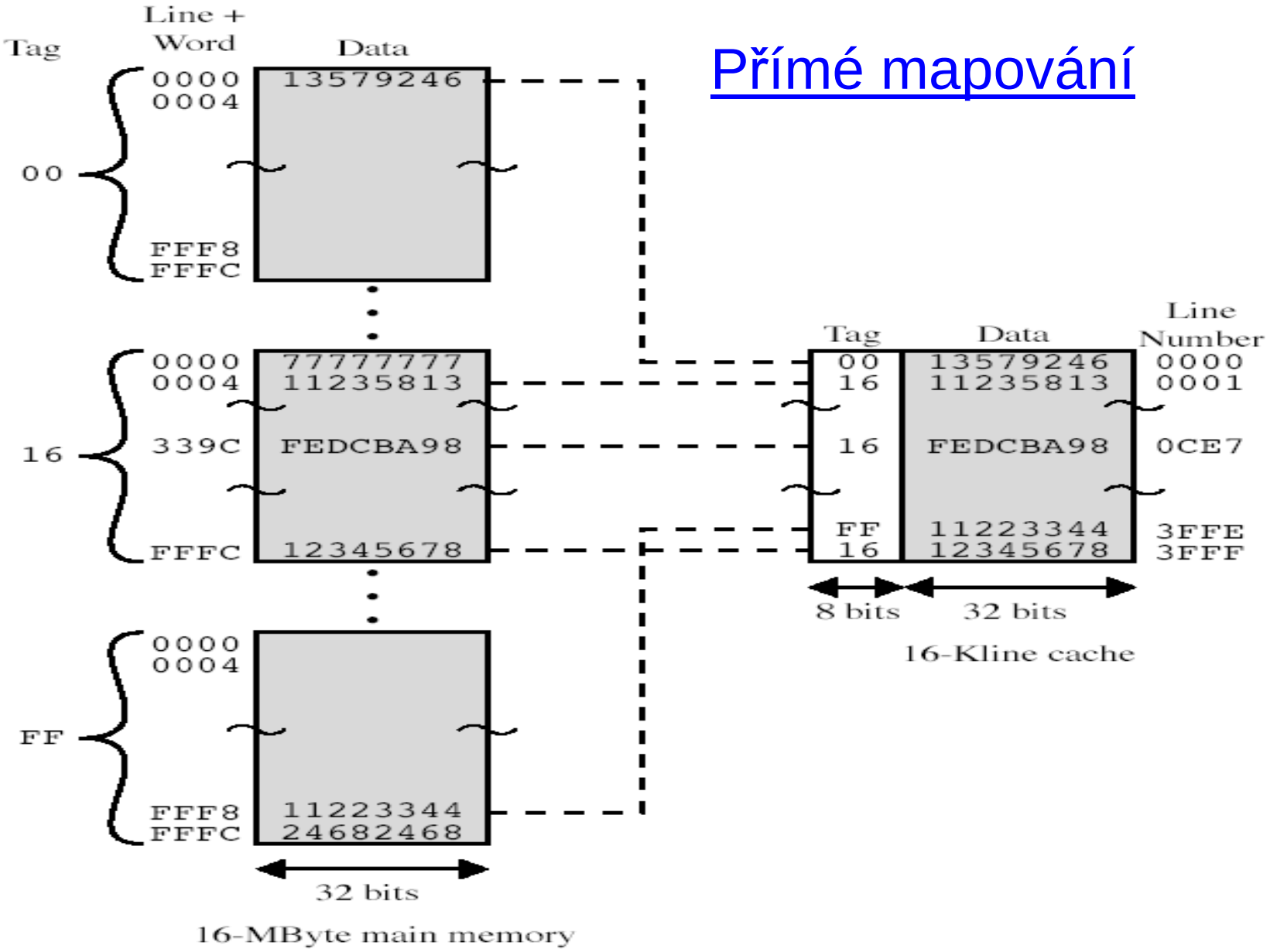
  nižší bity identifikují patřičné slovo

  vyšší bity specifikují blok (řádek)

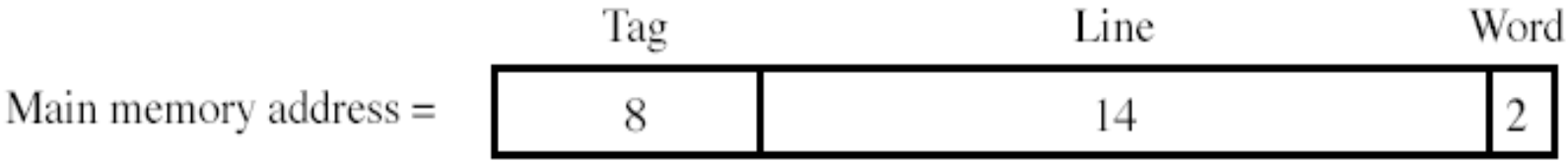
Přímé mapování



Přímé mapování



Přímé mapování složení adresy (příklad)

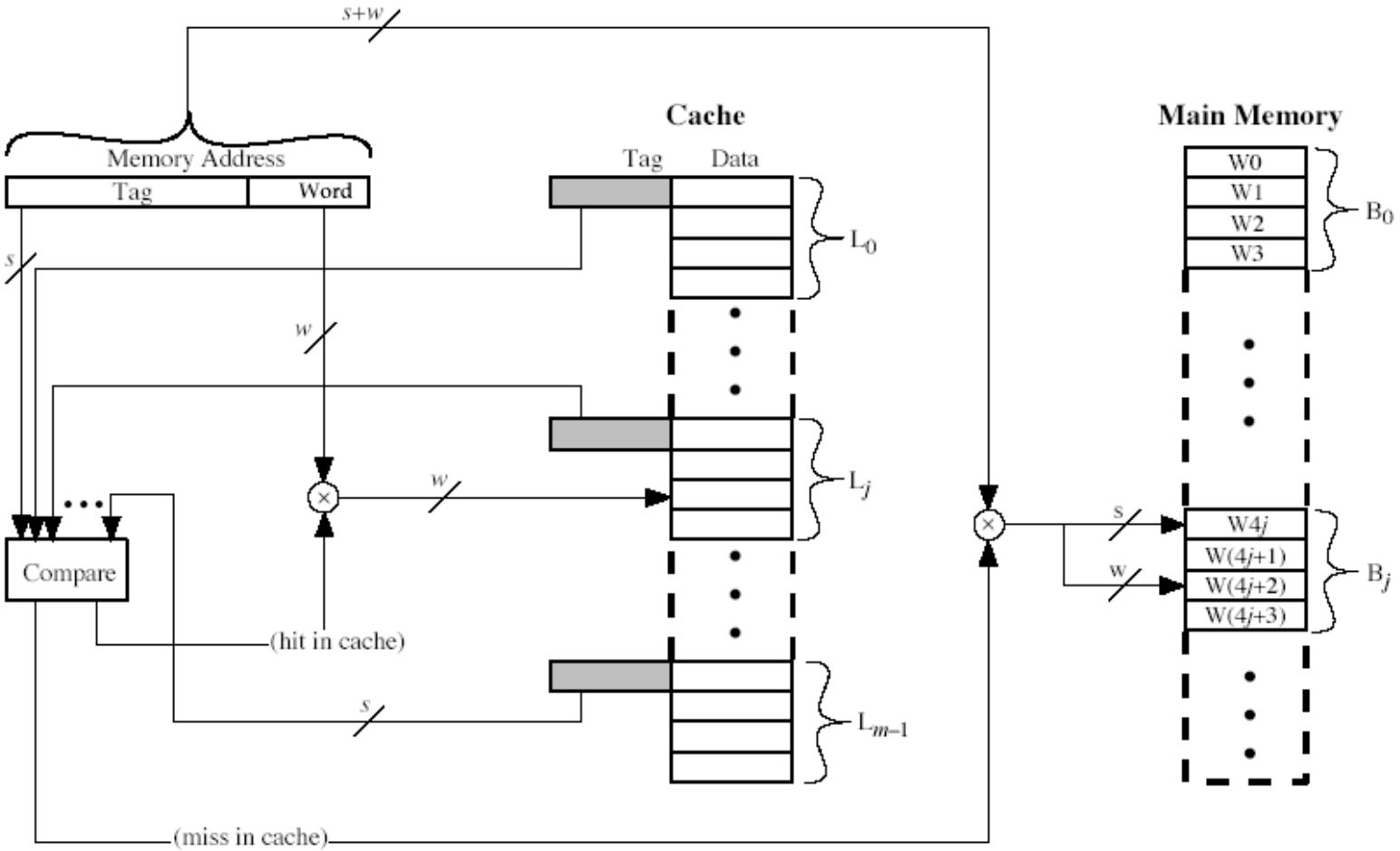


Výhody přímého mapování:
jednoduchý princip
levný princip
pevné místo pro každý řádek – pokud program potřebuje opakovaně dva řádky, které jsou mapovány do stejného řádku RVP, pak situace „cache miss“ jsou časté.

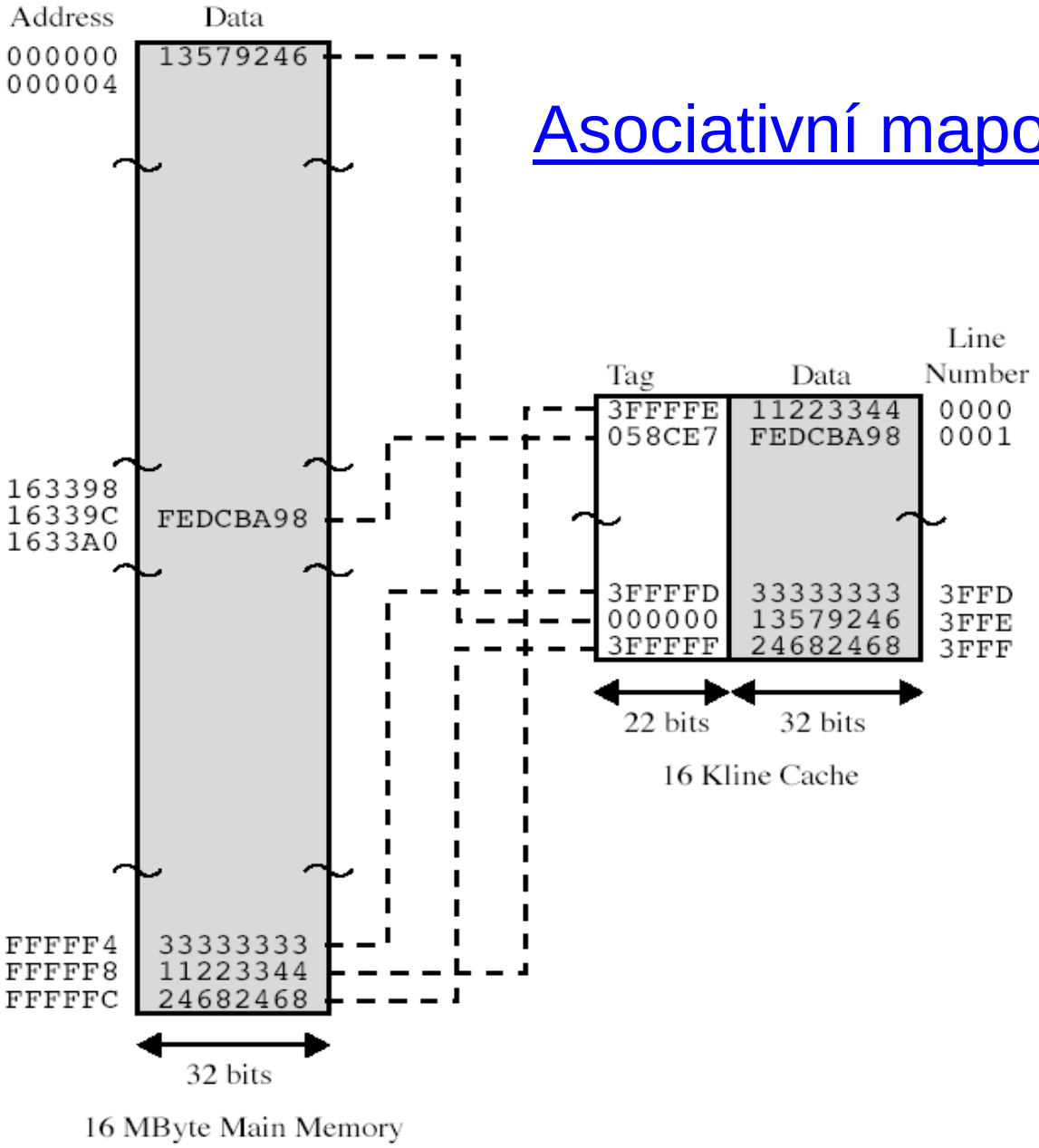
Asociativní mapování

- Řádek operační paměti může být vložen do libovolného řádku RVP.
- Adresa je interpretována jako adresa řádku a tag.

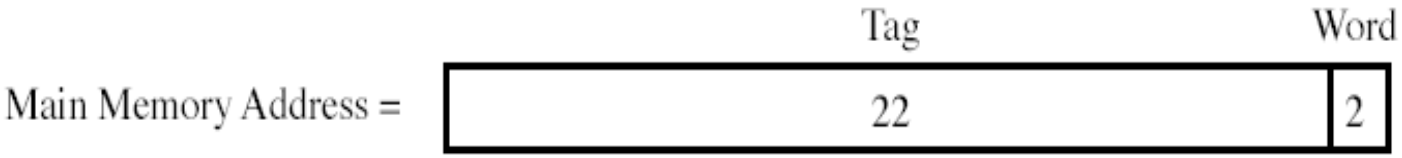
Asociativní mapování



Asociativní mapování



Dekódování adresy



- Dekódování adres:
  IIII IIII IIII IIII IIII I000
  F F F F F 8

- 2 nejnižší bity představují adresu slova, zbytek je Tag, zůstane 22 bitů:
  II IIII IIII IIII IIII III0
  3 F F F F E

- Obdobně FFFFFC, po oddělení dvou nejnižších bitů 00 zůstane 3FFFFF

Stav technologie

- L1: 8kB – 64kB
- L2: 64 kB – 2 MB
- L3: 2 MB – 256 MB

Stav technologie před asi 10 lety – L1: DRAM – 60 ns, SRAM 10 ns, dnes

L2: 10 – 20 ns

Dnes: L1: 4 ns, L2: 10 – 20 ns

Uvědomit si pojem: CAM (Content Addressable Memory) – paměť adresovaná obsahem

Další pojmy

- Inclusive cache – data existují na více úrovních RVP – např. L1 i L2
- Strictly inclusive cache – data existují povinně na více úrovních RVP
- Exclusive cache – data existují pouze na jedné úrovni RVP