

# Architektura počítačových systémů

Róbert Lórencz

8. přednáška

## Paměťová hierarchie, návrh skryté paměti – cache 2

<http://service.felk.cvut.cz/courses/36APS>  
[lorencz@fel.cvut.cz](mailto:lorencz@fel.cvut.cz)

- Přímá mapovaná skrytá paměť (fully associative cache)
- Plně asociativní skrytá paměť (fully associative cache)
- Cache s omezeným stupněm asociativity
- Strategie výběru oběti
- Redukce Miss rate
- Redukce Miss penalty
- Shrnutí

## Přímý zápis – Write through

- současný zápis slova do paměťového bloku cache a na odpovídající místo v paměti
- vždy je prováděn současný zápis do cache a do hlavní paměti
- jednoduchý, ale pomalý způsob udržování shodného obsahu cache a paměti
- zatěžuje komunikaci s pamětí, vyžaduje **zapisovací buffer – write buffer**

## Odložený zápis – Write back (copy back)

- obnovit obsah slova jen v cache a na odpovídajícím místě v paměti ponechat původní slovo
  - ▶ přidat **dirty bit** každé řádce cache, který indikuje potřebu zápisu slova z cache na odpovídající místo v paměti v případě, že blok obsahující slovo bude z cache nahrazen jiným slovem
  - ▶ OS musí před operací I/O aktualizovat obsah paměti obsahem cache!!
- zápis dat jen do cache, zápis hodnoty slova z bloku cache do paměti se provádí jen když je daný blok, který je označený dirty bitem, z cache odstraňován
- rychlý, ale implementačně složitější, než write through
- problém konzistence obsahu cache a hlavní paměti

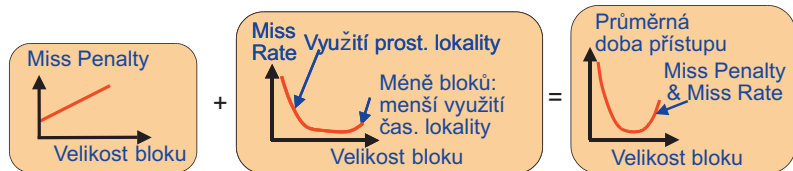
# Přímo mapovaná cache - velikost bloku 1

Výhody větších bloků  $\Rightarrow$  využití prostorové lokality

- větší blok obsahuje více slov v blízkosti požadovaného slova
- větší blok  $\Rightarrow$  více instrukcí po sobě jdoucích nebo více dat jednoho pole

Nevýhody větších bloků

- větší blok způsobuje méně výpadků, ale větší miss penalty
- k načtení většího bloku z nižší úrovně potřebujeme více času, než k načtení menšího bloku
- pro velké velikosti bloků vzhledem k velikosti cache existuje jen málo bloků v cache, a tak roste miss rate



# Přímo mapovaná cache - velikost bloku 2

Extrémní případ: jeden velký blok

Bit platnosti

Klíč

Cache data



- velikost cache = 4, slova = 16 B
- velikost bloku = 16 B
- jenom **jeden** vstup do cache!
- je pravděpodobné, že zpracovávaná položka bude znovu žádána
- je ale méně pravděpodobné, že bude žádána bezprostředně!
- potom je pravděpodobné, že další přístup do cache bude výpadek (miss)
- musí se načíst požadovaná data a nahradit původní blok novým
- nahrazená data mohou být požadována v dalších krocích: **noční múra návrhářů cache: ping pong efekt.**

# Přímo mapovaná cache - *AMAT*

## Průměrná doba přístupu do paměti

Average Memory Access Time – *AMAT*

$$AMAT = HT + MP \times MR$$

- ***HT* = Hit Time** čas potřebný pro nalezení a získání hledané položky v cache
- ***MP* = Miss Penalty** průměrný čas získání dat z nižší úrovně paměťové hierarchie při nenalezení hledané položky v cache (zahrnuje také detekci ***MR*** a předání dat procesoru)
- ***HR* = Hit Rate** poměrná úspěšnost nalezení dat v cache k celkovému počtu přístupů do paměti
- **$MR = 1 - HR = \text{Miss Rate}$**

# Přímo mapovaná cache - typy výpadků

## Studené výpadky – Compulsory misses

- vyskytují se po startu počítače
- cache po startu neobsahuje žádná data  $\Rightarrow$  výskyt studených výpadků až do naplnění cache

## Konfliktní výpadky – Conflict misses

- Výpadky, které se vyskytují z důvodů, že 2 a více rozdílných adres paměti je mapováno na stejné místo v cache
- 2 a víc bloků je mapováno do téhož místa v cache (stejný index), přítomnost jednoho bloku v cache vylučuje přítomnost jiného bloku se stejným indexem

Problém u DM cache, řešení jak zmenšit konfliktní výpadky:

- 1 zvětšit velikost cache, zvětšení je ale limitováno
- 2 pro tentýž index mít vícenásobné umístění bloků

# Plně asociativní cache – popis

## Fully Associative Cache

### Paměťové adresní pole:

- **Klíč & Offset:** stejné jako u DM a FA cache
- **Index: neexistuje** ⇒
- neexistují řádky, každý blok může být umístěn kdekoliv v cache
- hledat se musí podle **Klíče** v celé cache, jestli se požadovaná data někde nenacházejí

**Výhoda:** neexistují konfliktní výpadky (s definice), protože data mohou být kdekoliv

**Nevýhoda:** potřeba množství HW komparátorů pro každý jednotlivý blok

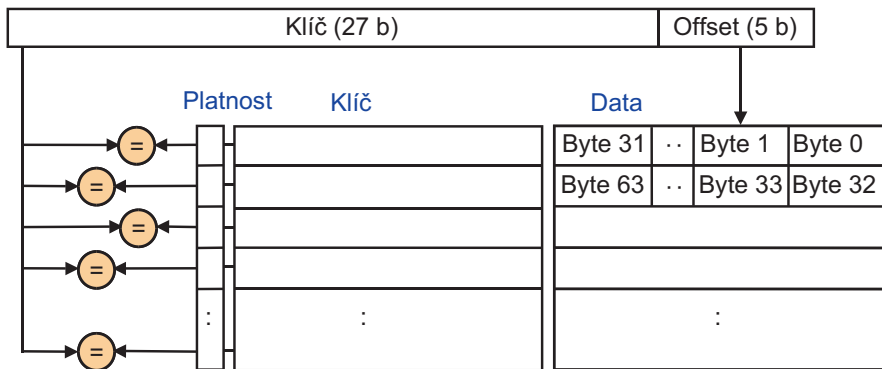
### Kapacitní výpadky – Capacity misses:

- **základní typ výpadků pro plně asociativní cache**
- výpadky způsobené omezenou kapacitou plně asociativní cache
- zmenšení kapacitních výpadků dosáhneme zvětšením velikosti cache (limitováno)

# Plně asociativní cache – příklad

**Příklad:** 64 KB cache, 32 bit adresa, velikost bloku = 32 B

Potřebujeme:  $2\text{ K} \times 27\text{-bit}$  komparátorů – **nereálné!**



# Cache s omezeným stupněm asociativity – popis 1

## N-Way Set Associative Cache

### Paměťové adresní pole:

- **Klíč & Offset:** stejné jako u DM cache
- **Index:** Ukazuje na řádek, který obsahuje tzv. **set**
- každý **set**: obsahuje několik bloků!
- když chceme najít hledané slovo v blocích jednoho **setu** (ukazuje na něj **index**), musíme porovnat všechny klíče příslušející blokům s klíčem adresy požadovaného slova

### Shrnutí:

- cache s omezeným stupněm asociativity je přímo mapovaná s ohledem na **sety**
- každý **set** je plně asociativní
- v podstatě N přímo mapovaných caches pracuje paralelně, tj. každý blok má svůj bit platnosti a data

# Cache s omezeným stupněm asociativity – popis 2

## Činnost cache

- je dána adresa požadovaného slova
- adresuje se set odpovídající indexu
- porovná se klíč žádaného slova s klíči bloků v setu
- výpadek, pokud není nalezena shoda ani s jedním klíčem
- pokud hit, potom použití offsetu k adresaci hledaného slova v daném bloku

## Výhoda cache s omezeným stupněm asociativity:

- již cache s  $N=2$  vyloučí množství konfliktních výpadků
- HW není o moc složitější, vyžaduje jen  $N$  komparátorů navíc atd.
- větší  $N \Rightarrow$  větší hit rate, protože více bloků paměti se stejným indexem může být pamatováno v cache

## Cache s omezeným stupněm asociativity $N$ a s $M$ bloky v setu je:

- DM cache  $\Leftrightarrow N = 1$
- plně asociativní cache  $\Leftrightarrow N = M$

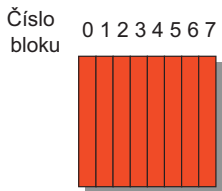
# Cache s omezeným stupněm asociativity – popis 3

Blok 12 je umístěn v 8 blokové cache:

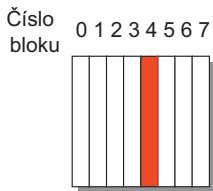
- plně asociativní
- přímo mapované
- s omezeným stupněm asociativity  $N=2$

$$\text{číslo setu} = \text{číslo bloku} \% \# \text{ setů}$$

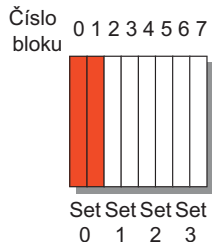
Plně asociativní:  
blok 12 umístěn  
kdekoliv



Přímo mapovaná:  
blok 12 umístěn jen  
do bloku 4 (12 mod 8)



Omezený stupeň:  
blok 12 umístěn  
do setu 0 (12 mod 4)



# Cache s omezeným stupněm asociativity – popis 4

## Příklad organizace cache

8 bloková cache

One-way set associative  
(direct mapped)

Block	Tag	Data
0	█	□
1	█	□
2	█	□
3	█	□
4	█	□
5	█	□
6	█	□
7	█	□

Two-way set associative

Set	Tag	Data	Tag	Data
0	█	□	█	□
1	█	□	█	□
2	█	□	█	□
3	█	□	█	□

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0	█	□	█	□	█	□	█	□
1	█	□	█	□	█	□	█	□

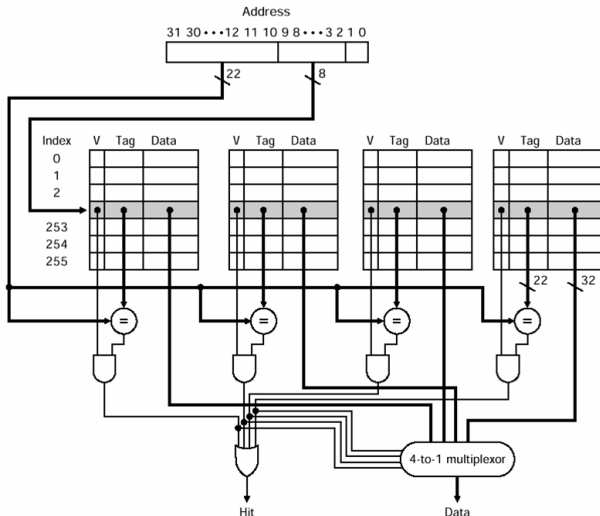
# bloků = N

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data
█	□	█	□	█	□	█	□	█	□	█	□	█	□	█	□

# Cache s omezeným stupněm asociativity – popis 5

**Příklad:** 4 KB N=4 cache,  $4 \times 1024$  B, velikost bloku = 4 B (1 slovo)



# Strategie výběru oběti – principy

## Přímo mapovaná cache:

- úplně specifikuje blok, který má být vyměněn.

## Cache s omezeným stupněm asociativity:

- index specifikuje set, ale blok může obsadit kteroukoliv pozici uvnitř setu.

## Plně asociativní cache:

- blok může obsadit kterýkoliv blok v cache.

## Pokud máme na výběr, kam zapsat nový blok, pak jak vybrat místo?

### Řešení:

- když je bit platnosti nula potom nový blok na dané místo
- když dané místo obsahuje blok s platným bitem platnosti, potom se musí určit pravidlo, které určí blok, který má být nahrazen novým blokem

# Strategie výběru oběti – LRU

## Nejméně používaná položka – LRU (Least Recently Used):

- vyměnit blok v setu, kterého slova byla nejméně čtena/zapisována

### Výhoda:

- využívá časovou lokalitu  $\Rightarrow$  zvyšuje hit rate
- při  $N=2$  je velmi jednoduché udržovat informaci o nejméně používané položce v setu (1 LRU bit)

### Nevýhoda:

- při  $N>2$  je HW komplikovanější, časová složitost pro udržení LRU informace také roste

**Příklad:** Máme cache s  $N=2$ , která má kapacitu 4 slova a bloky velikosti jednoho slova. Budeme vykonávat čtení slov na adresách: 0, 2, 0, 1, 4, 0, 2, 3, 5, 4. Kolik hitů a kolik výpadků bude připadat pro strategii výběru oběti pomocí LRU?

# Strategie výběru oběti – LRU 2

Adresesa: 0, 2, 0, 1, 4, 0, ...

- 0: miss, zápis do set 0 (loc 0)
- 2: miss, zápis do set 0 (loc 1)
- 0: **hit**
- 1: miss, zápis do set 1 (loc 0)
- 4: miss, zápis do set 0 (loc 1, replace 2)
- 0: **hit**

	loc 0	loc 1
set 0	0	<i>lru</i>
set 1		
set 0	<i>lru</i> 0	2
set 1		
set 0	0	<i>lru</i> 2
set 1		
set 0	0	<i>lru</i> 2
set 1	1	<i>lru</i>
set 0	<i>lru</i> 0	4
set 1	1	<i>lru</i>
set 0	0	<i>lru</i> 4
set 1	1	<i>lru</i>

# Strategie výběru oběti – LRU vs. Random

Miss rates v porovnání LRU × Random strategie výběru oběti

## Asociativita

Velikost	N = 2		N = 4		N = 8	
	LRU	Random	LRU	Random	LRU	Random
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

Je jen malý rozdíl Miss rate pro LRU a Random v případě velkých paměť cache

# Redukce Miss rate 1

Doposud známe redukci Miss rate:

- zvětšením velikosti bloku
- zvětšením  $N$  (stupně asociativity)

Větší cache:

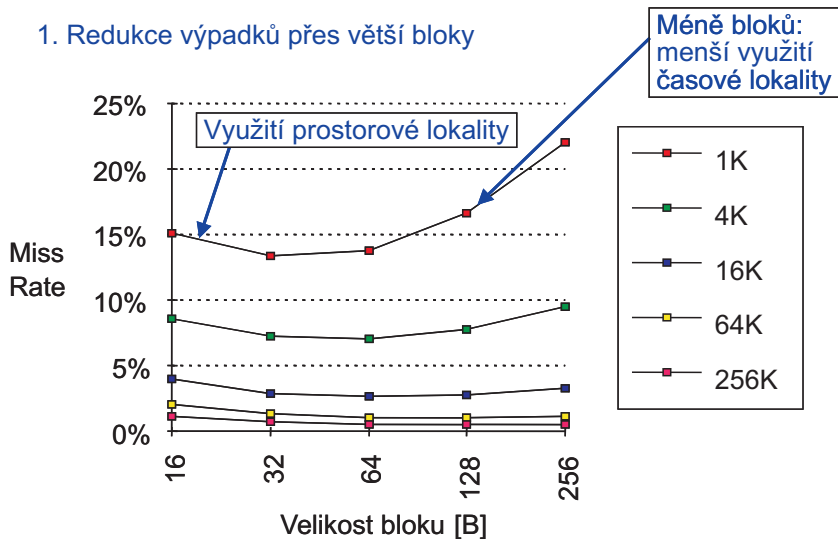
- limitována cenou a technologií
- Hit time L1 cache < doba taktu

Více místa pro bloky paměti v cache:

- plně asociativní cache  $\Rightarrow$  blok kdekoliv v cache
- $N > 1 \Rightarrow N$  možností pro umístění bloku paměti v cache
- pro DM cache  $N=1$

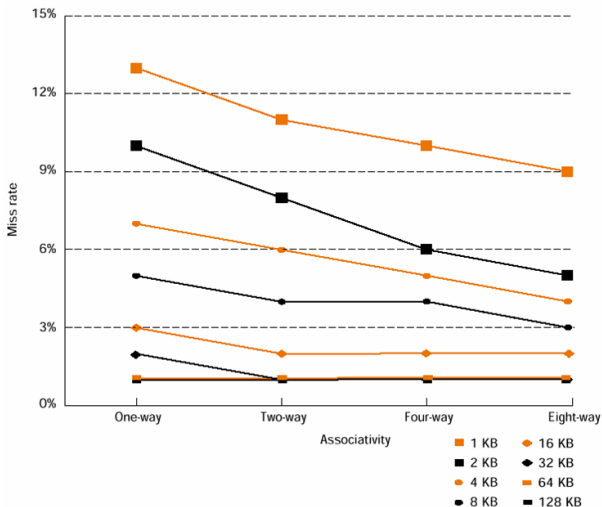
# Redukce Miss rate 2

## 1. Redukce výpadků přes větší bloky



# Redukce Miss rate 3

## 2. Reduce konfliktních výpadků prostřednictvím N



## 3. Reduce konfliktních výpadků prostřednictvím L2 cache

### Příklad 1:

$HT = 1$  takt

$MR = 5 \%$

$MP = 20$  taktů

$$AMAT = HT + MP \times MR = 1 + 0.05 \times 20 = 2 \text{ takty}$$

Při prvním použití cache: Miss Penalty  $\sim 10$  taktům procesoru.

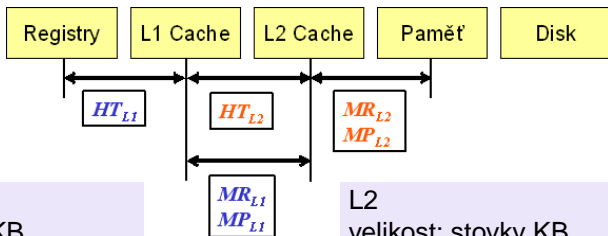
V současnosti: "1 GHz procesor" (1 ns takt) a 100 ns latence DRAM  
 $\Rightarrow 100$  taktů rozdíl!

**Řešení:** další cache mezi pamětí a procesorem: **L2 cache**.

## 3. Reduce konfliktních výpadků prostřednictvím L2 cache – 2

Otázka správného výběru mezi stupněm asociativity, velikosti bloku, výběru oběti atd., je nejlépe zodpovězena na základě návrhu výkonnostního modelu.

- Minimalizace:  $AMAT = HT + MP \times MR$
- Zahrnout technologii a chování aplikace



L1  
velikost: desítky KB  
 $HT$ : provedeno v 1 taktu  
 $MR$ : 1-5%

L2  
velikost: stovky KB  
 $HT$ : několik taktů  
 $MR$ : 10-20%

## 3. Reduce konfliktních výpadků prostřednictvím L2 cache – 3

$$AMAT_{L1} = HT_{L1} + MP_{L1} \times MR_{L1}$$

$$MP_{L1} = AMAT_{L2} = HT_{L2} + MP_{L2} \times MR_{L2}$$

$$AMAT_{L1\&L2} = HT_{L1} + MR_{L1} \times (HT_{L2} + MP_{L2} \times MR_{L2})$$

Definice:

**Local miss rate** – počet výpadků v dané cache dělený počtem přístupů do paměti pro danou cache (Miss rate L2 cache -  $MR_{L2}$ ).

**Global miss rate** – počet výpadků v dané cache dělený celkovým počtem přístupů do paměťového systému – generováno CPU ( $MR_{L1} \times MR_{L2}$ ). Jde nám hlavně o tyto výpadky.

# Redukce Miss rate 7

## 3. Reduce konfliktních výpadků prostřednictvím L2 cache – 4

### Příklad: 2

$$HT_{L1} = 1 \text{ takt}$$

$$MR_{L1} = 5 \%$$

$$HT_{L2} = 5 \text{ takt}$$

$$MR_{L2} = 15 \%$$
 (% L1 výpadků)

$$MP_{L2} = 100 \text{ taktů}$$

$$MP_{L1} = HT_{L2} + MP_{L2} \times MR_{L2} = 5 + 100 \times 0.15 = 20 \text{ taktů}$$

$$AMAT_{L1\&L2} = HT_{L1} + MP_{L1} \times MR_{L1} = 1 + 20 \times 0.05 = 2 \text{ takty}$$

$$AMAT_{L1} = HT_{L1} + MP_{L1} \times MR_{L1} = 1 + 100 \times 0.05 = 6 \text{ taktů}$$

↑  
 $MP_{L2}$

S L2 cache je systém 3× rychlejší!

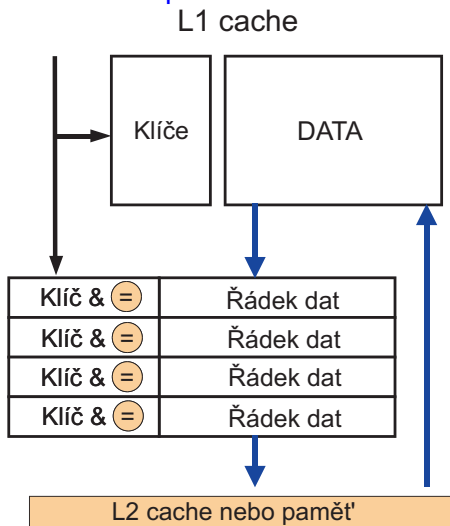
# Redukce Miss rate 8

## 4. Reduce prostřednictvím "Victim Cache" – paměť oběti

Využití malého HT DM cache a nějaké malé paměti odstraněných bloků.

Tím se značně eliminují konfliktní výpadky charakteristické pro DM cache.

Jouppi [1990]: 4-úrovňová "victim cache" odstraní 20% až 95% konfliktů pro 4 KB DM cache. (Alpha a HP)



## 5. Reduce prostřednictvím "HW prefetching" – HW přednačtení

### Přednačtení instrukcí a bloků dat

- Alpha 21064 načítá 2 bloky v případě výpadku
- blok je navíc umístěn do paměti – **stream buffer**
- pokud je výpadek, potom je kontrolován také **stream buffer**
- přednačtení vyžaduje rozšířenou šířku přenosu z nižších úrovní paměti

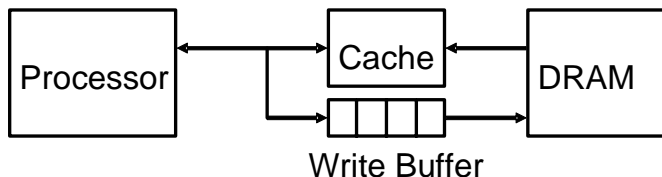
Existuje také **SW prefetching** – přednačtení dat.

## 6. Reduce prostřednictvím pseudoasociativity

- využití HT DM cache s nízkou hodnotou konfliktních výpadků cache s omezeným stupněm asoc.  $N = 2$
- nejdřív se prohlédne 1. polovina cache, když je miss, až pak se prohlíží 2. polovina cache

# Redukce Miss penalty 1

## 1. Reduce prostřednictvím zapisovací paměti – write buffer 1



### Write buffer

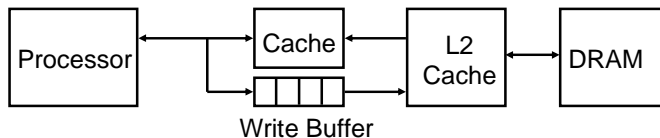
- je umístěn mezi cache a paměť (vedle cache)
- procesor zapisuje data do cache a současně do write bufferu
- řadič zapisuje obsah bufferu do nižší úrovně paměti (DRAM)
- write buffer je FIFO, typický počet položek je 4
- musí být ošetřeno přetečení zápisů
- pracuje optimálně pro frekvenci zápisu  $\ll 1/\text{cyklus zápisu DRAM}$

# Redukce Miss penalty 2

## 1. Reduce prostřednictvím zapisovací paměti – write buffer 2

### Write buffer problém

- frekvence zápisu  $\approx$  cyklus zápisu DRAM  $\Rightarrow$
- Write buffer saturace
- řešení:
  - ▶ použít **Write back cache**
  - ▶ přidat L2 cache





## Redukce Hit time

- zmenšení velikosti cache, ale zvětšení Miss rate
- použití přímo mapované cache zvětší Miss rate

## Redukce Miss rate

- zvětšení velikosti cache může zvětšit také Hit time
- $N > 1$ , ale může se zvětšit Hit time
- zvětšení velikosti bloku může zvětšit Miss penalty

## Redukce Miss penalty

- redukovat čas přenosu komponent Miss penalty
- přidat další cache (L2)

## Paměťová hierarchie

- optimalizuje cenu a výkon paměti
- zahrnuje kompromis mezi velikostí, rychlostí a cenou
- poskytuje iluzi o paměti, která má dobu přístupu paměti vyšší úrovně a velikost a cenu paměti na nižších úrovních
- to vše zásluhou platnosti **principů časové a prostorové lokality**

## Cache je součástí paměťové hierarchie

- využívá **princip časové a prostorové lokality**
- přímo mapovaná cache je jednoduchá a rychlá, má ale větší Miss rate
- cache s omezeným stupněm asociativity má nižší Miss rate, ale je složitější a pomalejší
- více úrovně cache systémy mají značnou popularitu, redukuje Miss rate, a také Miss penalty