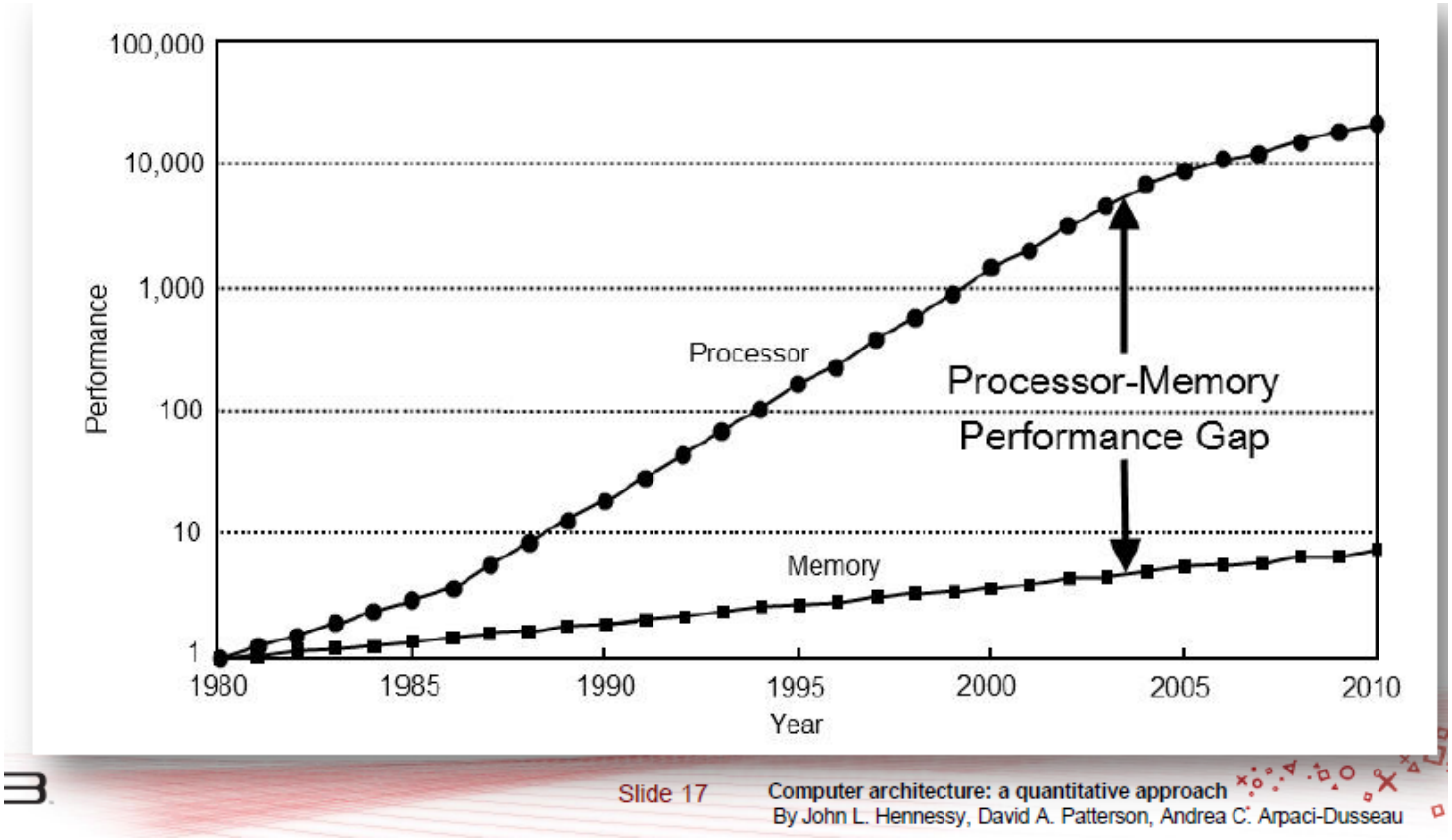
Paměťová hierarchie

INP 2019

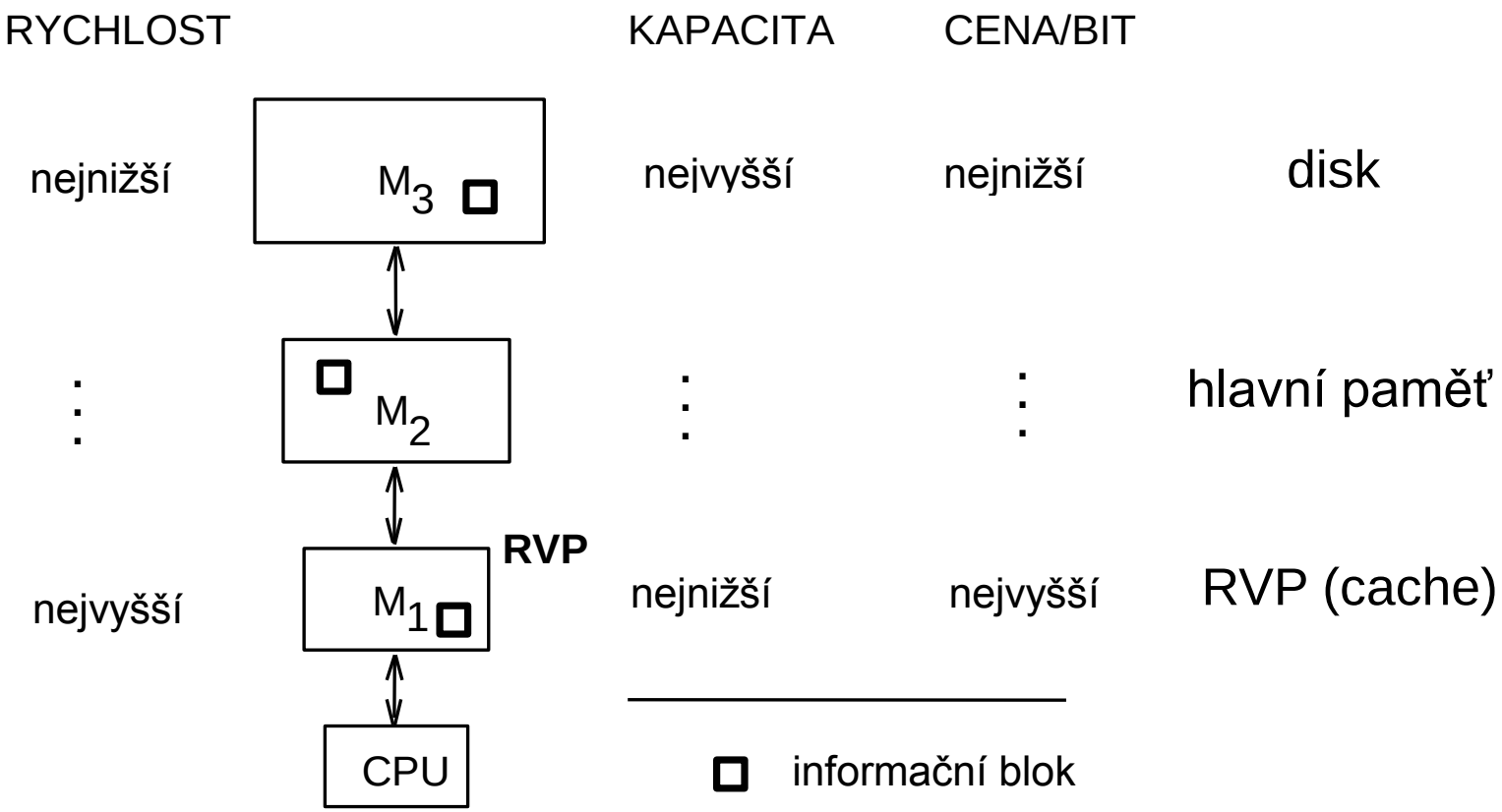
FIT VUT v Brně

Výkonová mezera mezi CPU a pamětí



Kvůli výkonové mezeře není užitečné, aby procesor přímo pracoval s hlavní pamětí, která je dnes levná, ale pomalá. Mezi CPU a hlavní paměť je proto umístěna jedna nebo několik rychlých (ale malokapacitních) vyrovnávacích pamětí (RVP), angl. cache. Vznikne hierarchický paměťový systém. Předpokladem jeho funkčnosti je existence časové a prostorové lokality odkazů k paměťovým místům („80 % času stráví program pouze ve 20 % kódu“).

Základní vlastnosti hierarchického paměťového systému



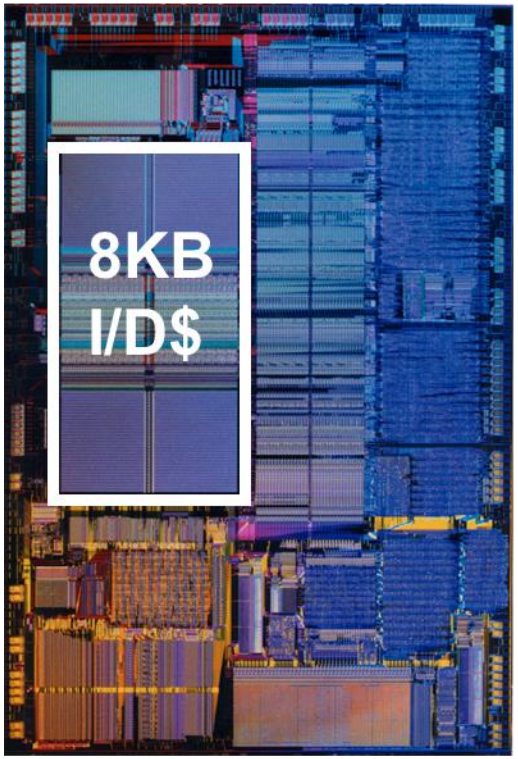
Jistý datový nebo instrukční blok (blok I/D) pak může v systému existovat až ve třech kopiích.
(na obrázku předpokládáme existenci jedné rychlé vyrovnávací paměti).
V moderních počítačích existuje více úrovní RVP: L1, L2 i L3.

Typické parametry

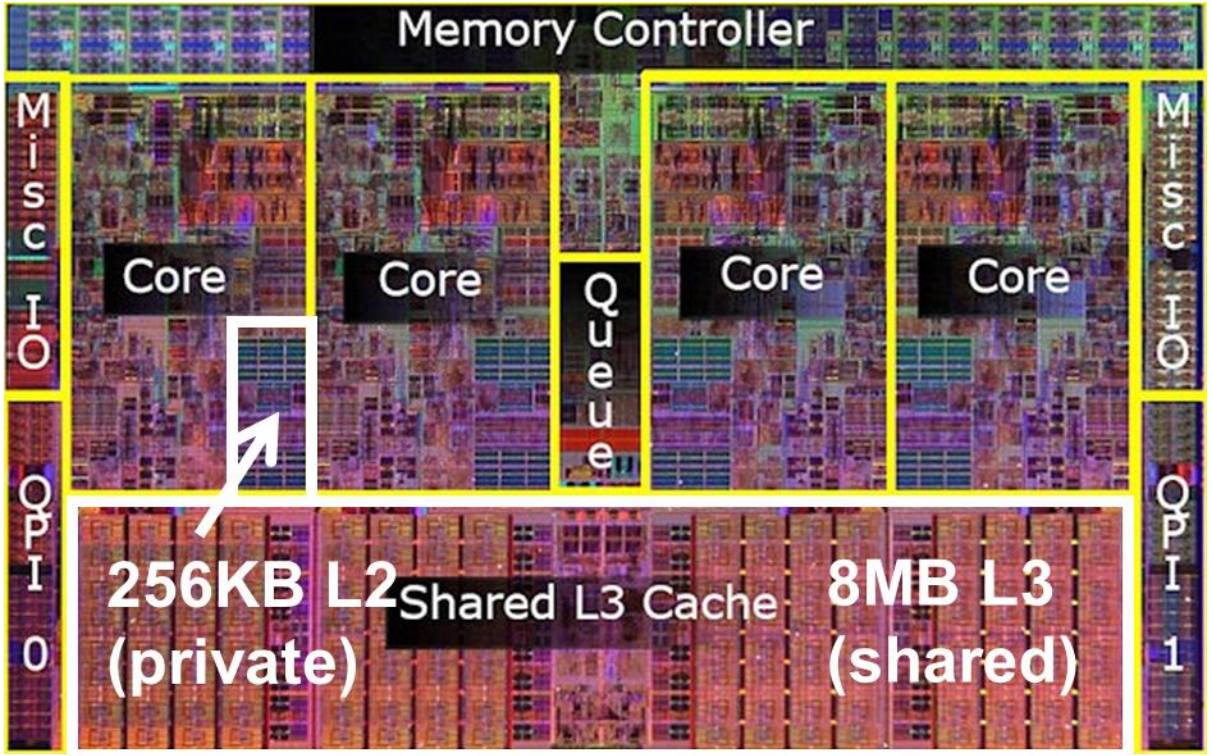
Procesor/ Parametr	80286 (1982)	Core i7-4770K (2013)
Jádra (vlákna)	1 (1)	4 (8)
CPU	8 MHz	3,5 GHz
Hlavní paměť, frekvence	8 MHz	1,6 GHz
Hlavní paměť, kapacita	128 KB	8 GB
RVP	Není	L1: 4 x 64 KB L2: 4 x 256 KB L3: 8 MB

Paměť	Latence
CPU Registr	<= 1
L1 Cache	4
L2 Cache	11
L3 Cache	40
RAM	~200

30%-70% plochy zabírají RVP



Intel 486



Intel Core i7 (quad core)

RVP

- RVP je rozdělena do bloků o konstantní velikosti (ideálně dle velikosti dat dodaných z hlavní paměti při blokovém přenosu).
  - Př. Vyrovnávací paměť o kapacitě 512 kB a velikosti bloku 32 bytů obsahuje celkem 16384 bloků.
- Hlavní paměť (např. DDR) je rozdělena na bloky o stejné velikosti. Těchto bloků je však mnohem více, než bloků ve vyrovnávací paměti => tudíž ne všechny bloky operační paměti mohou být v RVP.
- Musí existovat mechanismy umožňující potřebné bloky do RVP nahrávat a nepotřebné bloky odstraňovat.

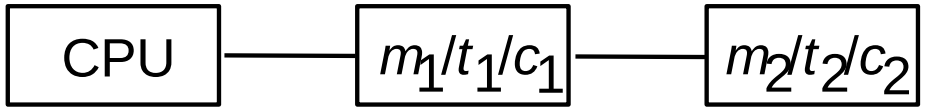
Účinnost RVP

- Základní údaj o účinnosti RVP je pravděpodobnost nalezení bloku p_{hit} (hit rate),
  - resp. pravděpodobnost neúspěchu (miss rate), neboli pravděpodobnost výpadku bloku ($1 – p_{hit}$).
  - Tyto parametry mohou být definovány zvlášť pro čtení a zápis, pro data i instrukce (data hit/miss rate, instruction hit/miss rate, atd.)
- Doba potřebná k nalezení bloku je přístupová doba RVP (ale jen v případě, kdy blok v RVP je).
- V případě neúspěchu (blok v RVP není) se přičítá ztrátová doba (miss penalty), což je doba potřebná na přesunutí bloku z hlavní paměti.
  - Je daná dobou potřebnou k uvolnění místa v RVP, přístupovou dobou k prvnímu slovu požadovaného bloku ve vzdálenější paměti plus doba přenosu celého bloku.
- Cílem je navrhnout organizaci a správu RVP tak, aby hodnota hit rate byla co nejvyšší (pozor, vždy závisí i na datech/programech)!!!
  - v praxi p_{hit} ~ 95-99%

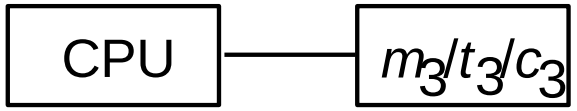
Analýza hierarchické paměti z hlediska ceny a výkonu

- Máme dvě konfigurace počítače podle obrázku s parametry podle tabulky, kde m_i je kapacita paměti, t_i je doba přístupu a c_i je cena v centech/Kbit.

a)



b)



i	m_i [B]	t_i [ns]	c_i [centů/Kbit]
1	5K	100	35
2	2M	500	14
3	2M	300	20

Výpočet

Máme vypočítat průměrnou cenu na KB v obou konfiguracích a máme určit, kdy bude výhodnější varianta a).

Střední ceny paměti v obou konfiguracích jsou (1B = 9b = 8b + 1b paritní, 100 centů = 1 $)

$$C_a = 9(m_1c_1 + m_2c_2)/100(m_1 + m_2) \quad \$/KB$$

$$C_b = 9c_3/100 \quad \$/KB$$

V konfiguraci a) máme pravděpodobnost úspěchu při čtení z M_1 danou p_h. Zjednodušeně můžeme napsat výraz pro střední dobu přístupu

$$t_a = t_1p_h + t_2(1 - p_h)$$

Má-li být konfigurace a) výhodnější než b) z hlediska výkonu, musí platit

$$t_a \leq t_3, \text{ tedy}$$

$$t_3 \geq t_1p_h + t_2 - t_2p_h$$

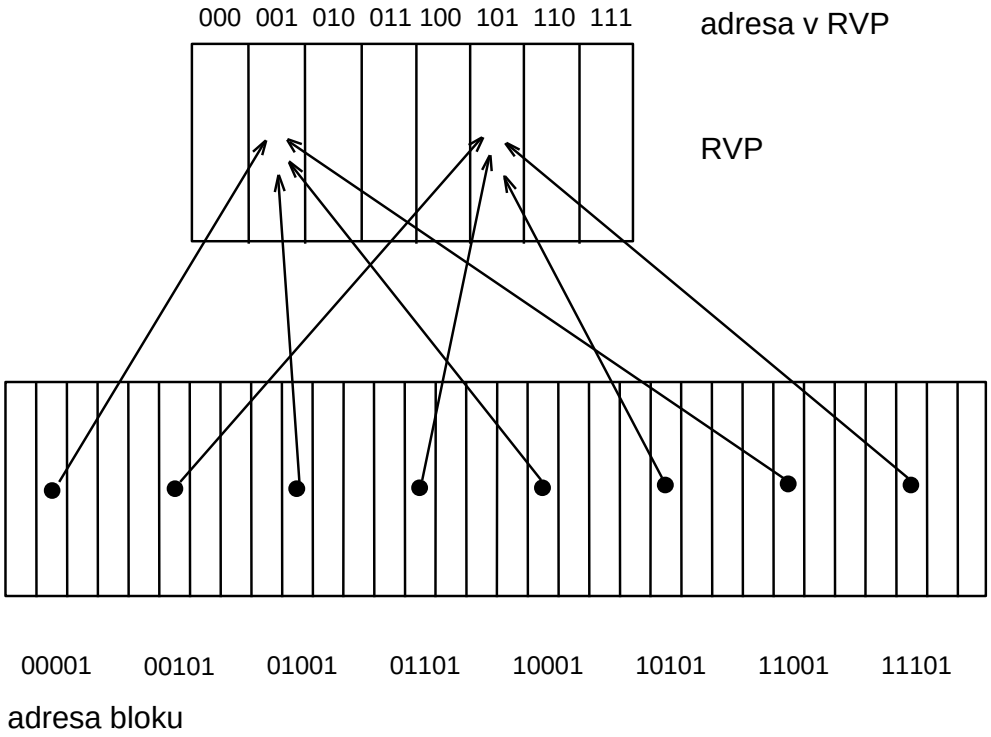
Odtud dostáváme podmínku pro pravděpodobnost úspěchu

$$p_h \geq (t_3 - t_2)/(t_1 - t_2)$$

Po dosazení hodnot podle tabulky dostáváme

$$p_h \geq 0{,}5, \text{ což je velmi mírný požadavek.}$$

RVP s přímým mapováním



Př. RVP má 8 blokových rámců s adresami 000 až 111, adresa bloku je pětibitová. Adresa polohy bloku v RVP se určí podle nejnižších tří bitů.

adresa v RVP = adresa bloku mod *počet bloků v RVP*

Výhody: jednoduchý koncept

Nevýhody: dva bloky, které mají stejnou adresu v RVP, nemohou být současně v RVP

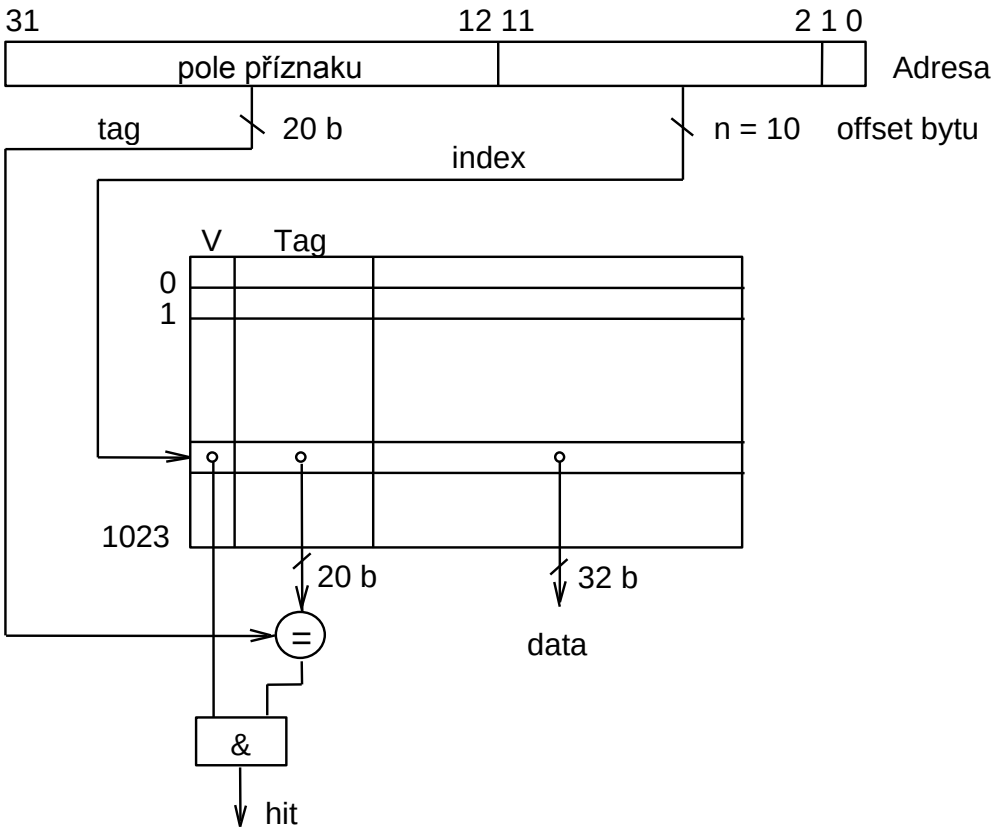
Činnost RVP s přímým mapováním

- U RVP se dále musí poskytnout informace, zda je požadovaný blok přítomen, příp. zda je informace v bloku platná. K tomu slouží adresový příznak (tag), což jsou zbývající horní bity adresy, a příznak platnosti (valid bit - V).
- Činnost RVP je pro zadanou posloupnost adres bloků ilustrována v tabulce. Předpokládáme, že je RVP na počátku činnosti prázdná.

čas

Adresa	Úspěch/neúspěch	Přidělený blok RVP	Tag	Validity
10110	Miss	110	10	0→1
11010	Miss	010	11	0→1
10110	Hit	110	10	1
11010	Hit	010	11	1
10000	Miss	000	10	0→1
00100	Miss	100	00	0→1
10000	Hit	000	10	1
10010	miss	010	11→10	1

RVP s přímým mapováním s 32-bitovou adresou



- Levá část RVP je adresová, pravá je datová.
- Celkový počet bloků o velikostí jednoho slova je 2^{30}.
- V RVP je umístěno 2^{10} bloků.
- Dolní odhad pravděpodobnosti úspěchu je $p_{hit} = 2^{10}/2^{30} = 2^{-20}$
- Díky lokalitě odkazů se v praxi dosahuje hodnot p_{hit} 0,9 až 0,98.

Proč je důležité programovat s ohledem na RVP?

Př.
Datový typ int je na 4B.
Blok dat RVP má velikost 4 x 4B
Mějme 2D pole `int a [4][5];`
Úloha: Sečtěte hodnoty všech položek pole **a**.

```
int sum_array_rows(int a[M][N])
{
  int i, j, sum = 0;
  for (i = 0; i < M; i++)
    for (j = 0; j < N; j++)
      sum += a[i][j];
  return sum;
}
```

Miss rate = 25%

```
RVP:
a[0][0], a[0][1], a[0][2], a[0][3]
a[0][4], a[1][0], a[1][1], a[1][2]
a[1][3], a[1][4], a[2][0], a[2][1]
a[2][2], a[2][3], a[2][4], a[3][0]
atd.
```

```
int sum_array_cols(int a[M][N])
{
  int i, j, sum = 0;
  for (j = 0; j < N; j++)
    for (i = 0; i < M; i++)
      sum += a[i][j];
  return sum;
}
```

Miss rate = 100%

```
RVP:
a[0][0], a[0][1], a[0][2], a[0][3]
a[1][0], a[1][1], a[1][2], a[1][3]
a[2][0], a[2][1], a[2][2], a[2][3]
a[3][0], a[3][1], a[3][2], a[3][3]
atd.
```

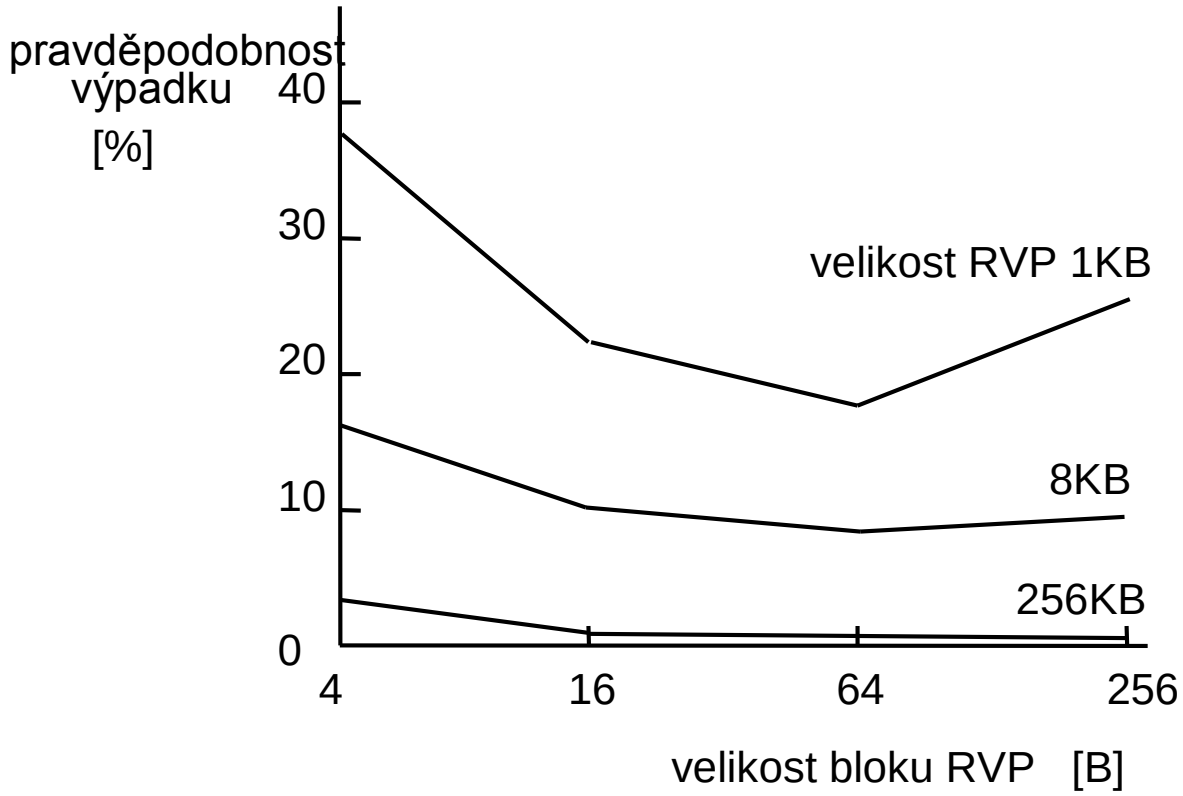
```
Hlavní paměť
adr: data
100: a[0][0]
104: a[0][1]
108: a[0][2]
10c: a[0][3]
110: a[0][4]
114: a[1][0]
118: a[1][1]
11c: a[1][2]
120: a[1][3]
124: a[1][4]
128: a[2][0]
12c: a[2][1]
130: a[2][2]
134: a[2][3]
138: a[2][4]
13c: a[3][0]
140: a[3][1]
144: a[3][2]
148: a[3][3]
14c: a[3][4]
150: ...
```

Koherence dat

- Změní-li se data v bloku zápisem, ztratí bloky na vzdálenějších úrovních platnost a nesmí se již použít. Vzniká tak datová nekonzistence, neboli nekoherence.
- Pro udržování koherence dat ve všech kopiích bloků je možno použít tyto strategie:
- přímý zápis (write-through, průpis)
  - Při přímém zápisu do RVP se zapisuje okamžitě i do bloku v M. Má výhodu ve snadné realizaci a rovněž ztráty při R-neúspěchu jsou nízké, protože nový blok se přisune bez dalšího zdržování. Je-li však větší rozdíl mezi rychlostí primární a sekundární (vzdálené) paměti, pak přímý zápis častými opravnými zápisy příliš zdržuje a je tedy nevýhodný.
- zápis s mezipamětí (write buffer)
  - Umožňuje odložit opravné zápisy až do okamžiku uvolnění přístupu k vzdálenější paměti, takže nedochází ke zdržování procesoru.
- zpětný zápis vždy
  - Při zpětném zápisu se opraví celý blok v sekundární paměti až při jeho odsouvání. Strategie zpětného zápisu vždy je nepraktická, protože blok se zapisuje do sekundární paměti i v případě, že k žádné změně nedošlo.
- zpětný zápis podle příznaku změny (write-back, copy-back, store on flag)
  - Prakticky použitelný je proto zpětný zápis podle příznaku změny (dirty bit). To je výhodné, protože střední počet zápisů do sekundární paměti je nižší než u přímého zápisu. Navíc blokový přesun lze zrychlit zvětšením šířky sběrnice mezi vyrovnávací a hlavní pamětí.

Typická pravděpodobnost výpadku vzhledem k velikosti bloku

Růst velikosti bloku má příznivý vliv jen do určité míry. Je-li velikost bloku vzhledem k velikosti RVP příliš značná, je v RVP málo bloků a příliš často se musí vyměňovat, protože požadovaná data nejsou často k dispozici.
Pozn. Pravděpodobnost výpadku se měří pro danou aplikaci nebo na sadě testovacích úloh.



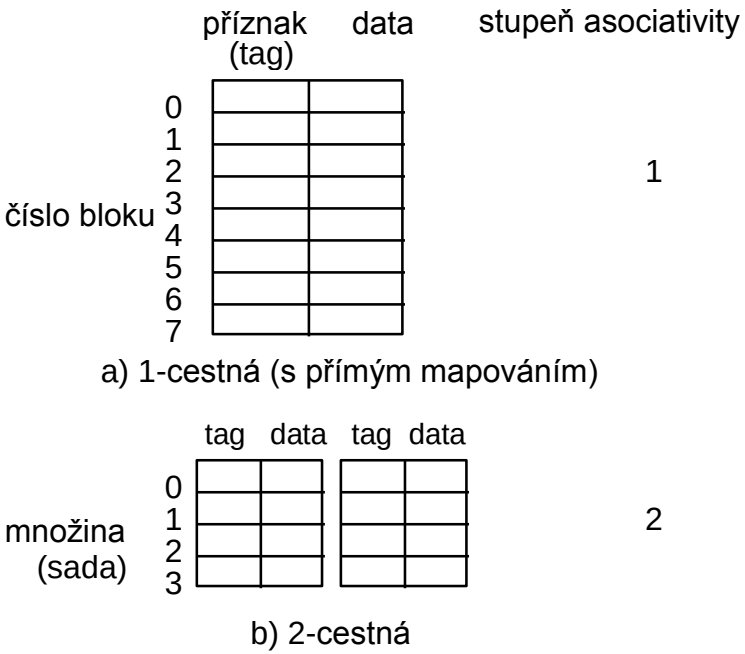
Jak umístit do RVP více položek se stejnou adresou bloku?

Problém vzájemného vytlačování položek se stejným ukazatelem se řeší zvýšením stupně asociativity.

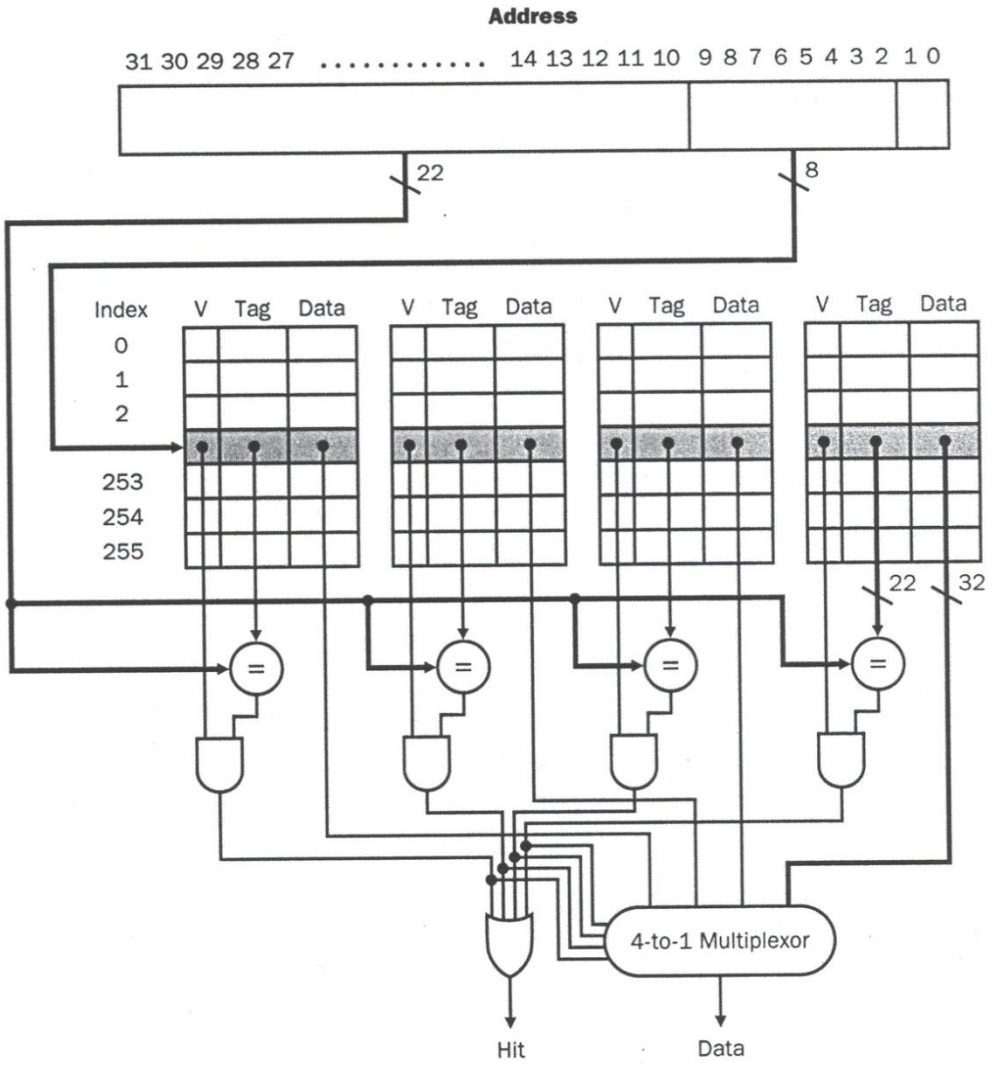
U dvoucestné paměti mohou být v paměti uloženy současně dvě položky se stejným ukazatelem.

Organizace b) však přináší oproti uspořádání a) jen malé zlepšení, protože kapacita paměti se nezměnila a do jednoho řádku se mapuje dvojnásobné množství adres.

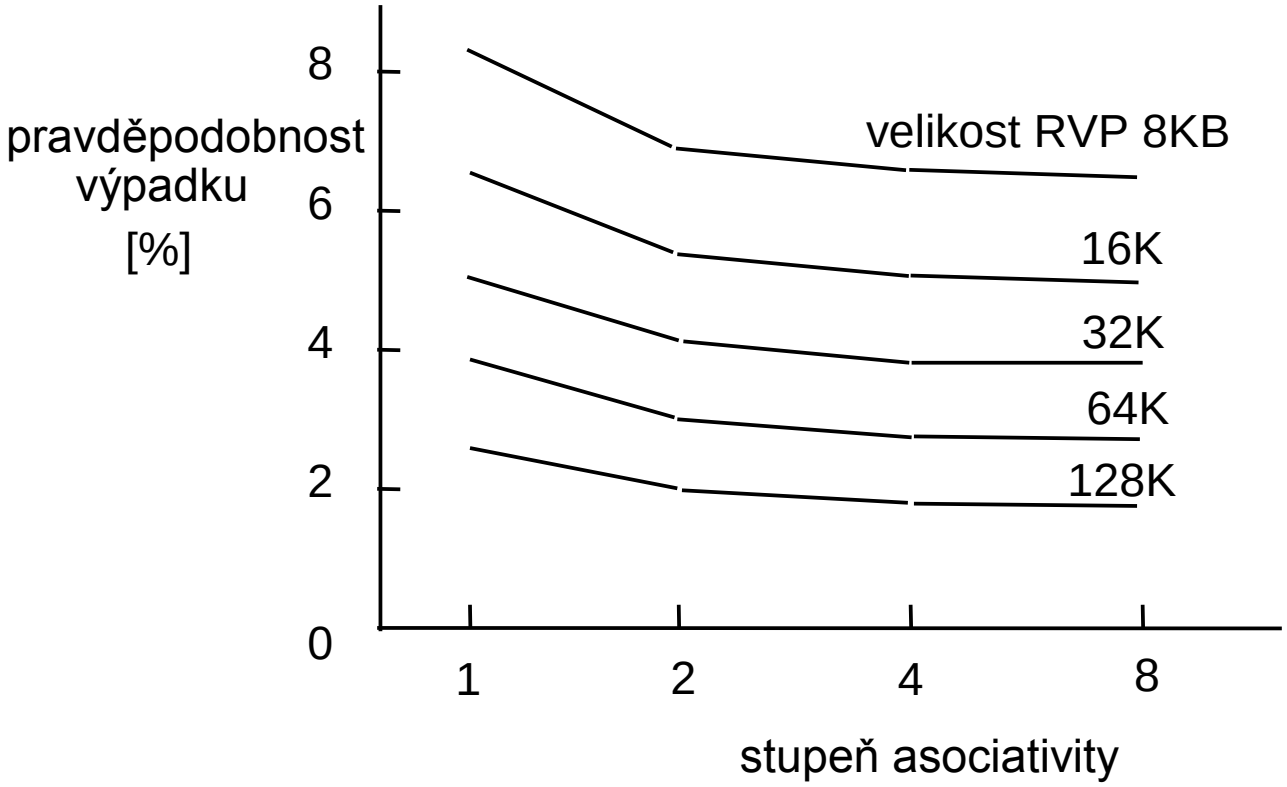
Stupeň asociativity lze dále zvyšovat, až dospějeme k plně asociativní paměti, kdy již je příznak celá adresa, která může být umístěna v kterékoliv pozici (v praxi nepoužitelné, příliš drahé).



Implementace 4-cestné RVP



Typická závislost pravděpodobnosti výpadku bloku na stupni asociativity



Výběr oběti

- Jsou-li všechny položky pro daný ukazatel obsazeny, je třeba rozhodnout, kterou položku zrušíme, a uvolníme tak místo pro novou položku. U dvoucestné RVP a obecně pro stupeň asociativity >1 tak vzniká problém výběru oběti.
- Tento problém řeší některá ze strategií náhrady:
  - Least Recently Used (LRU)- ponechávají se položky používané v poslední době a ruší se nejdéle nepoužitá položka
  - Most Frequently Used (MFU) - ruší nejčastěji použitou položku
  - RAND - náhodný výběr oběti
  - FIFO - oběť je položka, která je v RVP nejdéle
- Strategie LRU, MFU i FIFO vyžadují další obvodové doplňky, jako registry pro udržování času použití, resp. jejich úsporné modifikace, nebo čítače četnosti použití. Zde se musí řešit např. otázka přeplnění čítačů, a jsou navrženy algoritmy, které zavčas zahájí dekrementaci vybraných čítačů.

Související problematika – viz IOS

- virtuální paměť
- překlad adresy
- fyzický adresový prostor
- logický adresový prostor
- stránka
- segment
- rámec
- Translation Look-Aside Buffer (TLB)