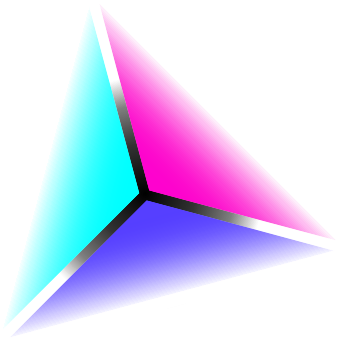
Struktura a architektura počítačů

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické

Paměťová hierarchie, princip cache pamětí

BI-SAP, květen 2021



Obsah

- Paměťová hierarchie
- Virtualizace
- Cache

Zdroje: Hennessy, J.L.: Patterson, D.A.: Computer Architecture A Quantitative Approach. Morgan Kaufmann Publishers, 3rd edition 2003

Přednášky BI-APS, *Roth*, A., *Martin*, *M*. *CIS 371 – Computer Organization* an. *Design*. *University* of *Pennsylvania*, *Dept*. of *Computer* and *Information Science*, *Spring 2009* (použito v přednáškách MFF UK)

Motivace

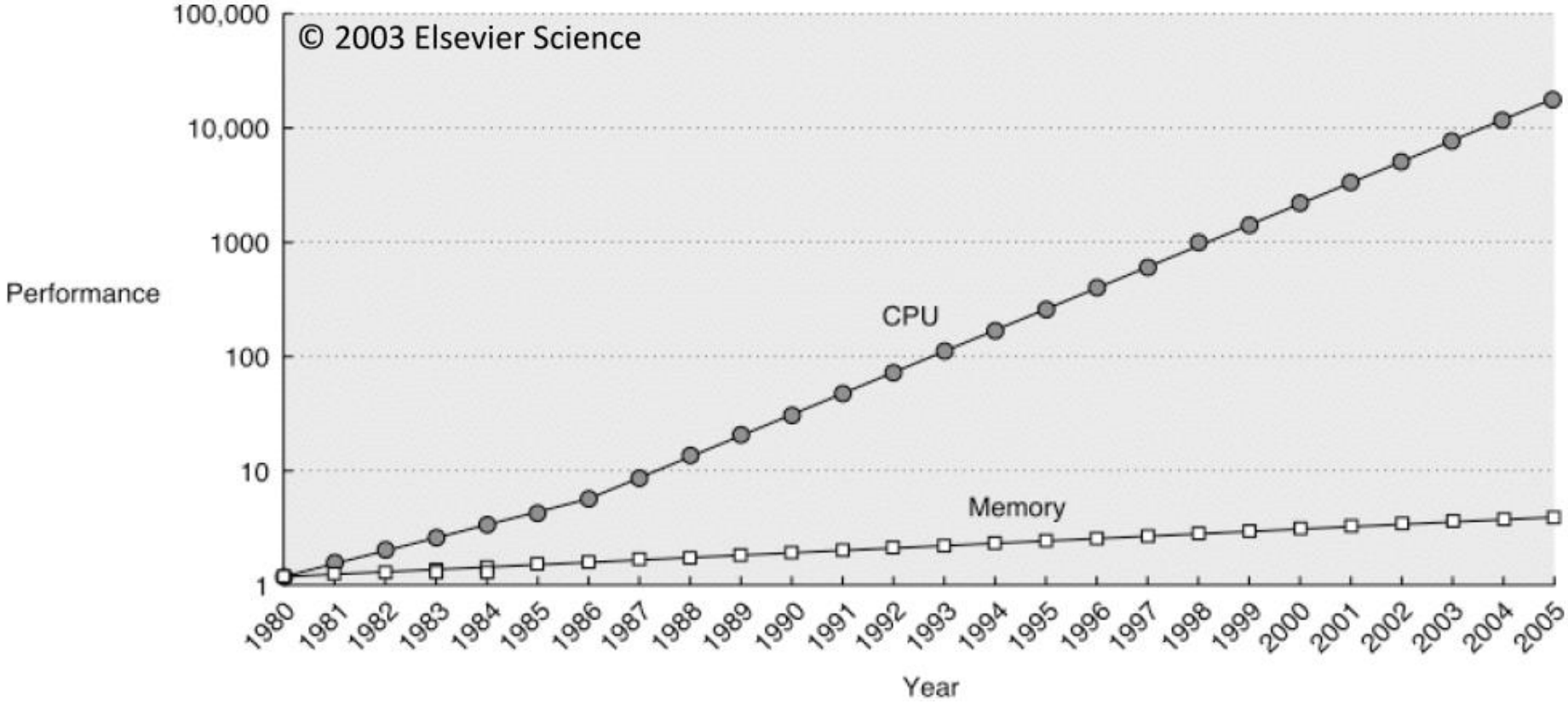
Výkon procesoru je dnes omezen „rychlostí“ paměti

- operace “add” trvá 0.33ns na 3GHz procesoru
- latence dnešních pamětí je více než 33ns

- ale přístup do paměti (čtení/zápis) nemůže být až 100x pomalejší než ostatní operace

Paměťová bariéra

- (the memory wall)



Výkon procesoru dominován výkonem pamětí

- 35-55% nárůst rychlosti vs. 7% snížení latence
- výkon procesorů roste rychleji než výkon pamětí

Řešení?

Aplikace v jednom okamžiku obvykle využívá jen malou část paměťového prostoru (cykly, pole):

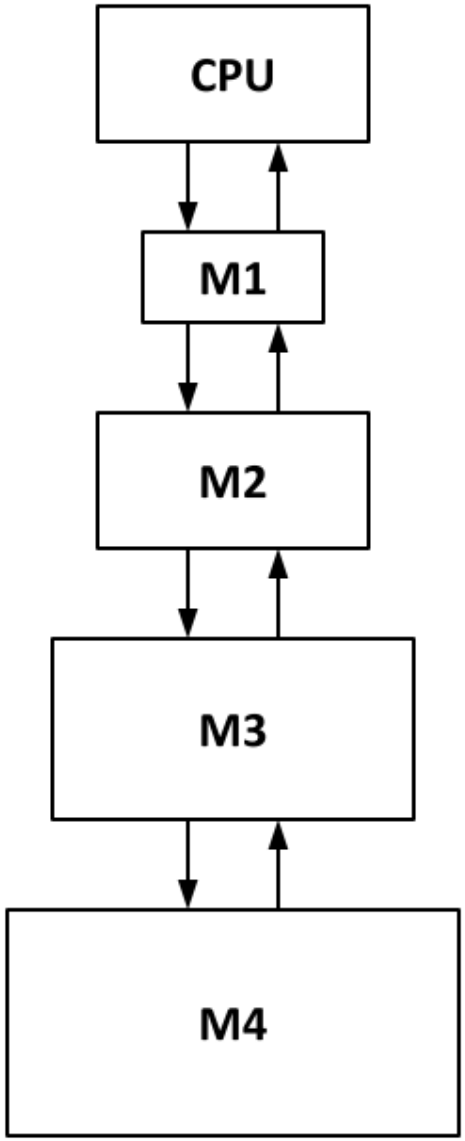
Časová (*temporal*) lokalita

- k nedávno použitým datům bude s velkou pravděpodobností přistupováno znovu
- $\longrightarrow$ nedávno použitá data si schováme v malé, velmi rychlé paměti

Prostorová (*spatial*) lokalita

- s velkou pravděpodobností se bude přistupovat k datům poblíž těch, se kterými se právě pracuje
- $\longrightarrow$ přistupovat k datům po větších blocích, které zahrnují i okolní data

Řešení: hierarchie pamětí



Hierarchie paměťových komponent

- vyšší vrstvy: rychlé ↔ malé ↔ drahé
- nižší vrstvy: pomalé ↔ velké ↔ levné

Vzájemně propojené "sběrnicemi"

- přidávají latenci, omezují propustnost

Nejčastěji používaná data v M1

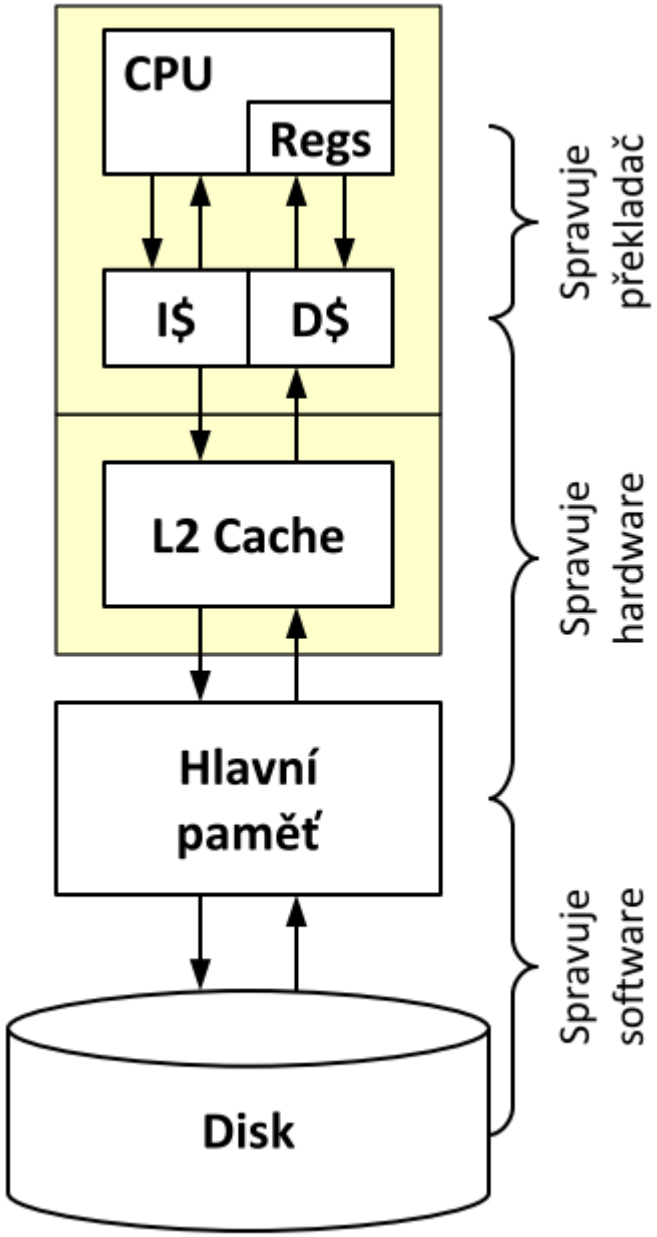
- M1 + druhá nejpoužívanější data v M2
- M2 + třetí ....

Cíl: víceúrovňové uspořádání pamětí různých typů (kapacit a rychlostí) s cílem dosáhnout optimálního poměru výkon x cena

Nutnost: optimalizace přesunu dat mezi vrstvami

Hierarchie

- M0: registry
- M1: primární cache
  - oddělená, instrukční a datová (I$, D$)
  - kapacita ~ KB
- M2: sekundární cache (L2)
  - nejlépe na čipu, určitě v pouzdře
  - SRAM
- M3: hlavní paměť
  - DRAM, ~ GB
- M4: záložní (odkládací) paměť
  - soubory, swap
  - magnetické disky, flash paměti



Paměťová hierarchie

Cena paměti je cca přímo úměrná kapacitě nepřímo době přístupu (rychlosti)

Typ paměti	Typická realizace	Doba přístupu	Kapacita	Propustnost
Registry	Klopné obvody	~ 1ns	Desítky-stovky bytů	
Cache (skrytá paměť)	Statická RAM	~ 10 ns	~ MB	300 GB/s
Hlavní paměť (operační paměť)	Dynamická RAM	~ 50ns	~ MB … GB	25 GB/s
Vnější paměť	Magnetický disk „hard“ disk	~ 10 ms	~ GB	100 MB/s
Záložní paměť	CD, DVD magnetická páska ZIP, flash disk	~ stovky ms až s	~ GB … TB	

Analogie s knihovnou

Registry ⇔ knihy na stole

- aktivně využívány, na stůl se jich moc nevejde

Cache ⇔ police na knihy (domácí knihovna)

- střední kapacita, poměrně rychlý přístup

Hlavní paměť ⇔ knihovna (městská, pro nás NTK)

- velká kapacita, je v ní skoro vše, pomalý přístup

Záložní paměť ⇔ meziknihovní výpůjčka

- velmi pomalé, ale možné

Virtualizace paměti

- systém několika pamětí s různými parametry (kapacita, rychlost), řízený tak, aby vytvářel paměťové prostory potřebné velikosti pro program a data
- umožňuje realizaci jednoho nebo několika logických (virtuálních) adresových prostorů, kde každý může být větší než skutečná kapacita hlavní paměti
- hlavní paměť ... **fyzický** paměťový prostor
- **logické** adresové prostory jsou ve skutečnosti ve vnější paměti

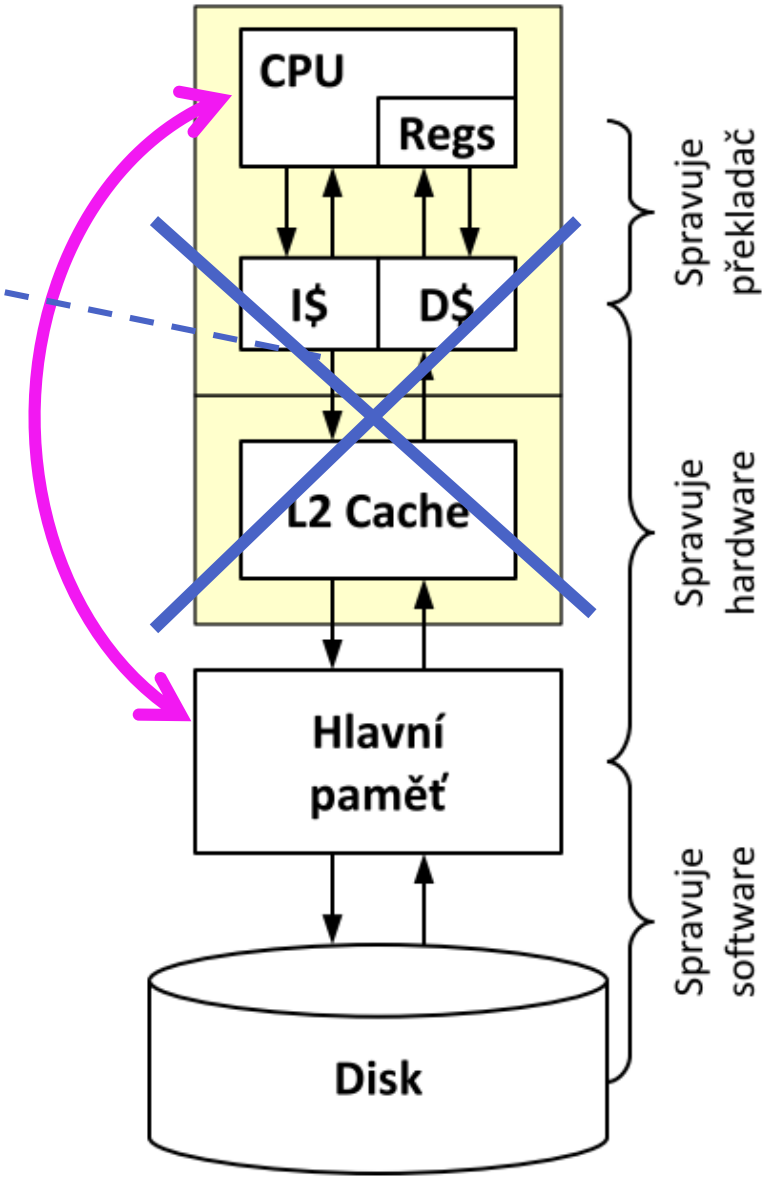
Virtuální paměť, princip

- části programů a data jsou přesouvány do hlavní paměti, požaduje-li k nim procesor přístup
- v hlavní paměti jsou jen ty programy a data, se kterými procesor právě pracuje
- lze přemístit části programu bez nutnosti je znovu překládat
- zajištění ochrany dat před neoprávněným přístupem a modifikací
- práce s ***logickými*** adresami ne absolutními
- hlavní paměť se adresuje ***fyzickými*** adresami
- překlad logických adres na fyzické zajišťuje mechanismus virtuální paměti … **stránkování, segmentace**

Přesun dat disk ↔ hlavní paměť

- Hlavní paměť ↔ CPU
- zatím neřešíme cache (je průhledná)
- HW/SW řešení (podrobněji BI-OSY)

jak vzbudit zdání, že procesor má k dispozici celý disk



Stránkování

Logický adresový prostor je rozdělen na úseky pevné délky
- **stránky** (logické stránky)

Fyzický adresový prostor je rozdělen na stejně velké úseky
- **stránkové rámce** (fyzické stránky)

Logický adresový prostor je realizován ve vnější paměti. Data (úseky programu) se přesouvají do hlavní paměti po jednotlivých stránkách, jsou-li v průběhu výpočtu požadována a pokud se příslušná stránka již v paměti nenachází.

Překlad (určení kam se stránka do hlavní paměti přesune) používá datovou strukturu ... **tabulku stránek**

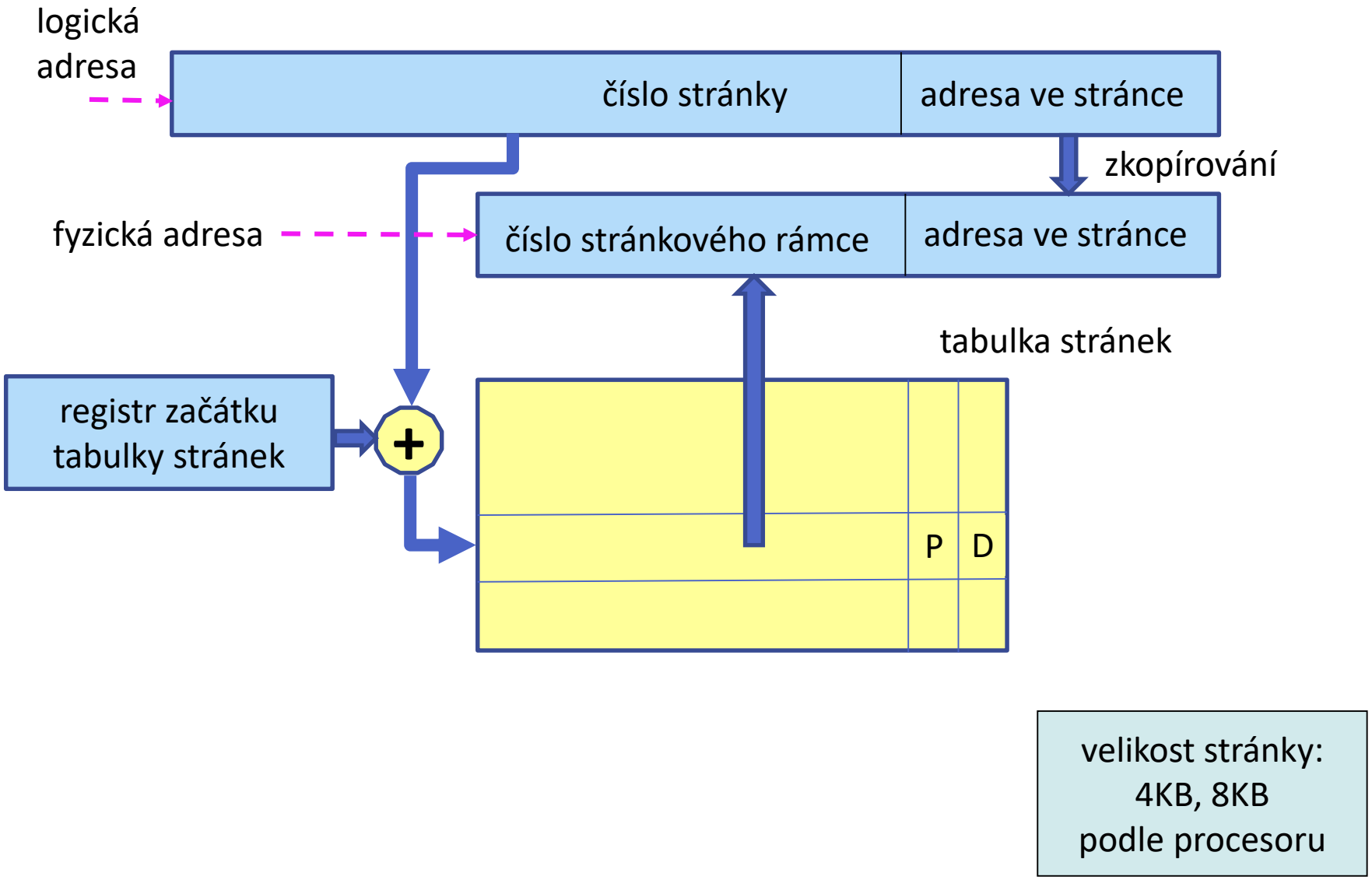
Tabulka stránek

- je uložena v hlavní paměti
- obsahuje pro každou logickou stránku jednu položku
- položka obsahuje informaci, zda se daná stránka nachází v hlavní paměti a pokud ano, tak kde (v kterém stránkovém rámci)

Poznámka 1: Mezi virtuálním a fyzickým adresním prostorem existuje **pevné mapování**. Virtuální adresa je pomocí tabulky stránek překládána na fyzickou adresu.

Poznámka 2: Pro urychlení překladu se využívá asociativní paměť (Translation Lookaside Buffer – TLB) s omezeným počtem posledních prováděných překladů.

Stránkovací mechanismus



velikost stránky:
4KB, 8KB
podle procesoru

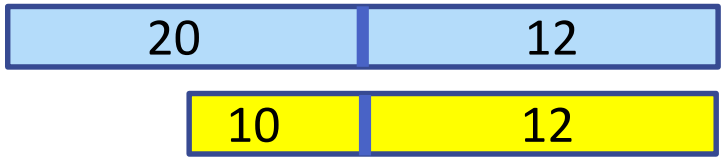
Realita, problémy

- tabulka stránek musí obsahovat 1 položku pro každou stránku v logickém adresovém prostoru, i když není použita
- při mnohem větším logickém prostoru proti velikosti hlavní paměti, může tabulka stránek zabírat velkou část hlavní paměti:

Příklad: virtuální paměť 4GB .... 32b logická adresa (**LA**)

Hlavní paměť 4MB .... 22b fyzická adresa (**FA**)

velikost stránky 4KB ..... 12b pro adresu ve stránce



Počet položek v tabulce stránek: 2^{20}

Položka musí obsahovat: 10 bitů pro horní část fyzické adresy

+ 1 bit platnosti

2B pro každou položku

→ Tabulka stránek zabírá 2MB .... tzn. polovinu hlavní paměti .... **Nepřípustné!!!**

Obsah tabulky stránek

- horní část fyzické adresy (číslo stránkového rámce)
- příznak **P**řítomnosti stránky v hlavní paměti (P)
- příznak změny dat ve stránce (zda byla po dobu přítomnosti v HP do stránky zapisováno) ... **D**irty bit (D)
- další bity, např. určující, zda je vhodné stránku přepsat (vyhodit z hlavní paměti, podle toho jak a kdy byla použita)

Stránkovací mechanismus

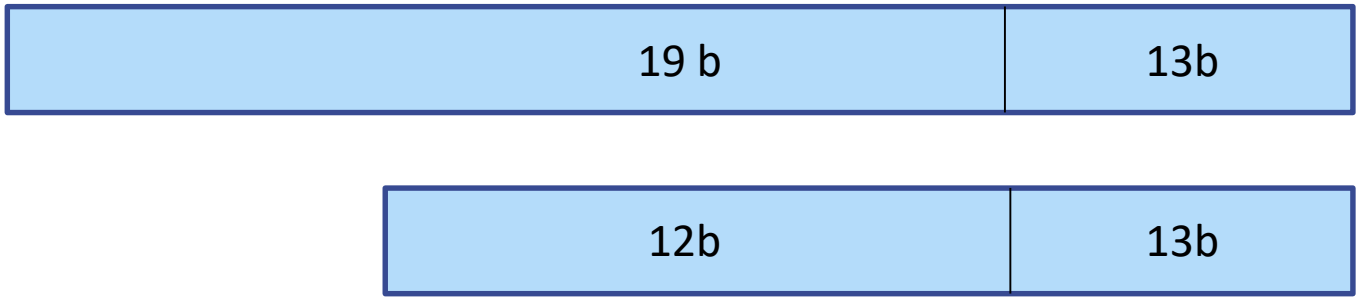
- je-li stránka přítomna v hlavní paměti, přeloží se logická adresa na fyzickou (příznak přítomnosti stránky v HP)
- není-li stránka přítomna, vyvolá se přerušení. Přerušovací mechanismus vyvolá načtení (celé) stránky z vnější paměti.
- pokud při načítání není volný žádný stránkový rámec v HP, je třeba nějaký uvolnit ... přesun vhodné stránky do vnější paměti (např. nejdéle nepoužité, viz další bity, slide 16 ...)
- nebylo-li do stránky zapisováno (*Dirty bit*), nemusí se přesouvat do vnější paměti, stačí ji jen přepsat.

Příklad

Hlavní paměť má velikost 32 MB (25b adresa)

Vnější paměť 4 GB (32b adresa)

- Jak velká bude tabulka stránek a kolik % hlavní paměti zabere (velikost stránky 8 KB)?



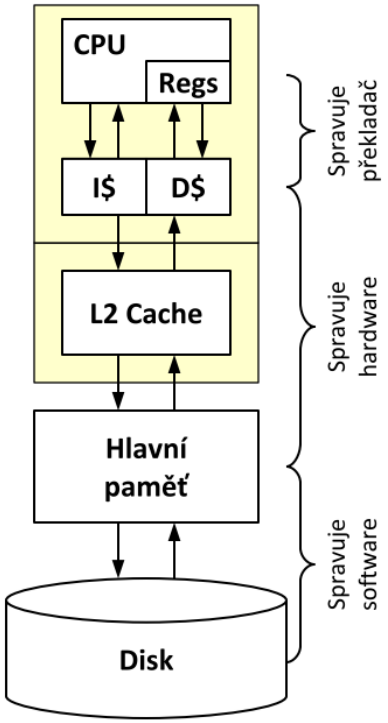
2^19 položek, položka 12++ bitů …. 2B …. Velikost tabulky stránek .. 2^20 = 1MB

2^20 : 2^25 = 1/32 …. cca 3%

Cache

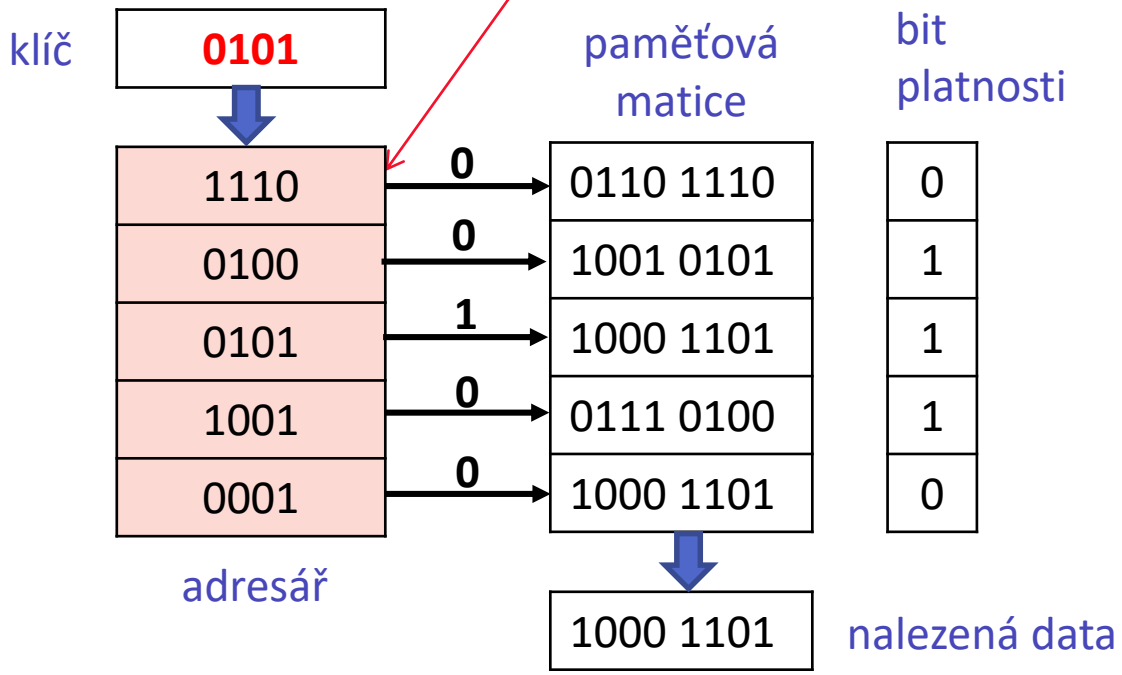
- "malá" rychlá paměť zařazená mezi procesor a hlavní paměť
- využití asociativního přístupu k položkám (CAM)
- obsahuje kopie nejčastěji používaných položek hlavní paměti (viz princip lokality)
- realizována SRAM
- synonyma .... *cache, buffer memory, skrytá paměť (SP),* *špatný termín:* ~~*rychlá vyrovnávací paměť*~~

Poznámka: zde se nebudeme zabývat různými typy cache pamětí (L1, L2, L3, datovou, ani instrukční). Zaměříme se možnou hardwarovou realizaci. Podrobněji viz BI-APS.



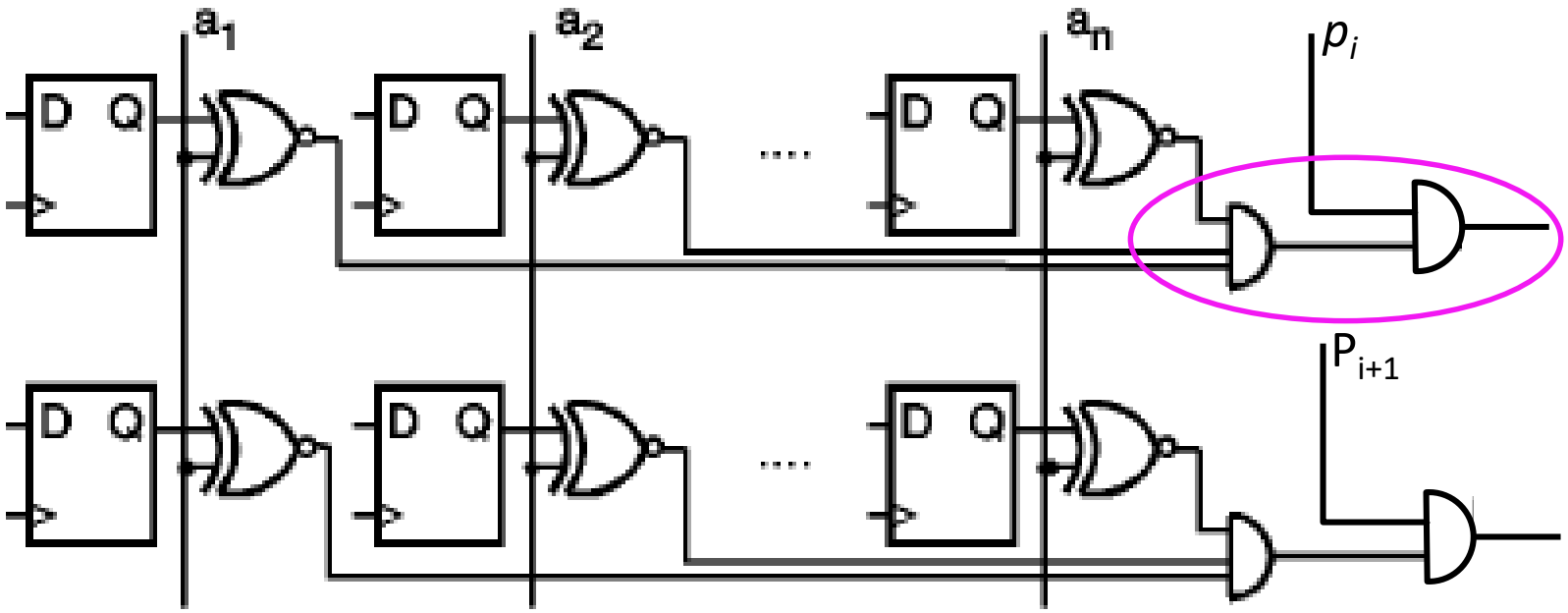
Princip plně asociativní paměti

- adresuje se částí datové položky, která se má vyhledat, tzv. **klíčem**
- na rozdíl od adresovatelné paměti (např. SRAM) neobsahuje adresový dekodér, ale adresář:



Adresář plně asociativní paměti

- Každá položka adresáře obsahuje logické obvody umožňující najednou prohledat všechny položky adresáře (srovnává adresu $a_1a_2…a_n$ a bit platnosti p_i)



Využití asociativních pamětí v počítači

- Cache: data jsou kopie často používaných položek dat z hlavní paměti, klíčem je adresa položky
- Translation Lookaside Buffer – TLB (urychlení překladu virtuální adresy na fyzickou při stránkování … viz BI-OSY)

Princip čtení ...

- Zahájení cyklu čtení současně z cache i z hlavní paměti.
- Pokud se položka v cache nalezne, cyklus hlavní paměti se nedokončí.
- V opačném případě se přečtou data z hlavní paměti (a obvykle zároveň uloží do cache)

Čtení dat z cache

- Cache Hit: data jsou v cache nalezena
- Četnost „úspěšného“ čtení z paměti cache (Hit rate) ... počet přístupů do cache s nalezením požadovaných dat dělená celkovým počtem přístupů do paměti
- Přístupová doba cache paměti (Hit time) ... nalezení dat v cache a jejich předání CPU
- Cache Miss: výpadek cache ... data v cache nejsou, je třeba je dodat z další úrovně paměti (např. HP)
- Četnost výpadků cache (Miss rate = 1 – Hit rate)
- Miss penalty: doba potřebná k získání dat z pomalejší paměti, tzn. k výměně bloku v cache a doručení dat do CPU

Zápis

- pokud položka v cache není, zapíše se zpravidla jen do hlavní paměti
- pokud je, postupuje se dvěma způsoby
  - průběžný zápis (write through) ... nová hodnota se zapíše se zároveň do cache i HP
  - odložený zápis (write back) ... nová hodnota se zapíše jen do cache a při uvolňování položky z cache se musí její obsah přepsat do HP pokud byla modifikována po dobu přítomnosti v cache
- kam zapisovat = kam uložit paměťový blok z HP?
  - Kamkoli (plně asociativní) x jen na určené místo (přímo mapovaná)/na více míst (n-cestně mapovaná)

Asociativní paměť s omezeným stupněm asociativity

- nevýhoda plně asociativní paměti:
  - adresář je tvořen speciálními obvody
  - při stejné kapacitě cca trojnásobná plocha čipu
- řešení ... omezený stupeň asociativity, tzn. každé položce je určeno místo (nebo několik míst podle stupně asociativity), kde se může nacházet. Toto místo je určené částí adresy položky.
- typická realizace cache pamětí

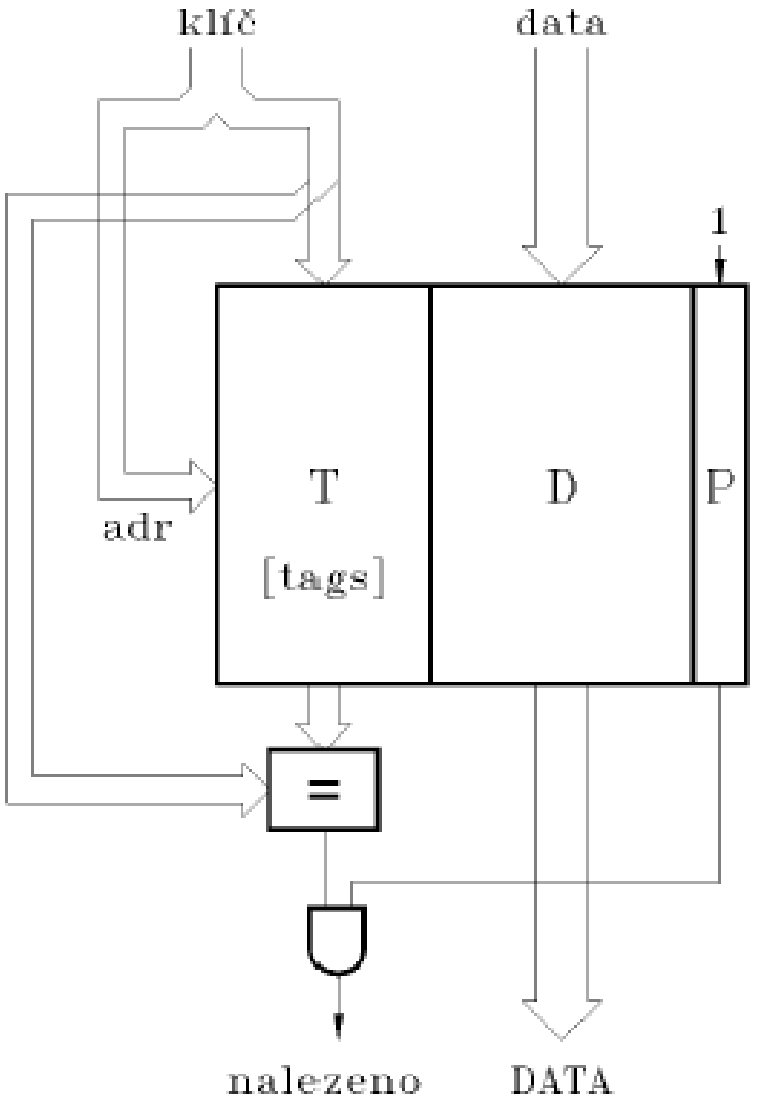
... cache

- adresář je možné realizovat běžnou pamětí RAM
- přítomnost položky se zjistí porovnání s klíčem (nebo několika klíči) uloženém v adresáři
- klíčem je část adresy
- stupeň asociativity je počet míst, na kterých se položka může nacházet
- pro zvýšení efektivity jsou data uložena po blocích (řádcích) např. 16 slabik

Přímo mapovaná cache

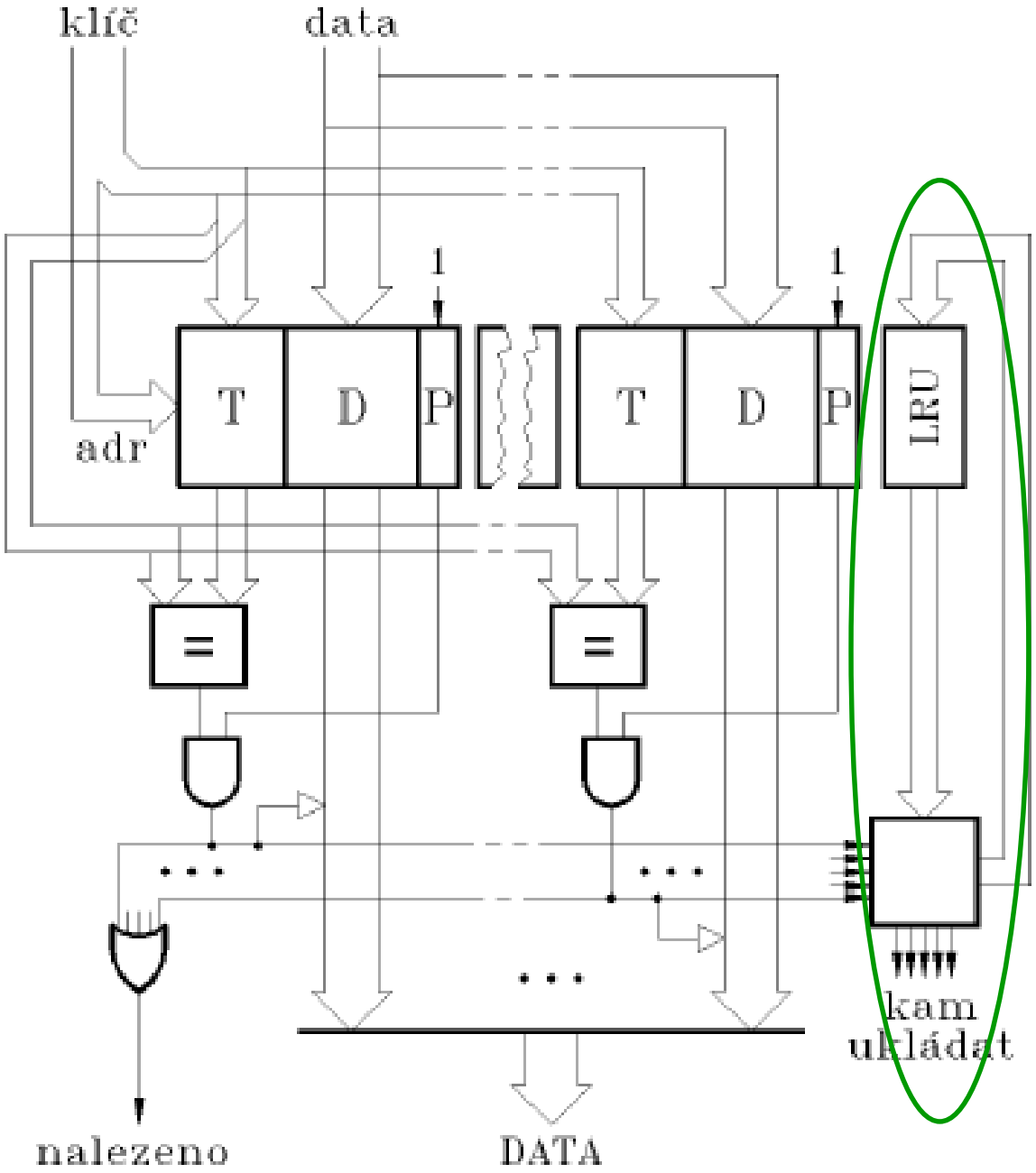
= stupeň asociativity **1**

pro každou položku (adresu, klíč) je **jedno** místo)



stupeň asociativity **2** (*nebo více*)

pro každou položku jsou určena dvě (*nebo více míst: 4, 8*)



Otázky

- Jaký stupeň asociativity má plně asociativní paměť?
- Z kolika možností se vybírá (např. LRU) při přepisování (vyhazování) bloku při stupni asociativity 1, 2, 4?

Jednoduchý příklad

- Kapacita hlavní paměti … 64 KB
- Kapacita cache … 32 B
- Blok … 4 B
- Stupeň asociativity … 2

- *Příklad: Hledám data na adrese 0FFE*

Jednoduchý příklad

- Kapacita hlavní paměti … 64 KB, *16bitová adresa*
- Kapacita cache … 32 B
- Blok … 4 B … *výběr slabiky 2 bitová adresa*
- Stupeň asociativity … 2 … *dva bloky po 16B v blocích po 4B*

- *Příklad: Hledám data na adrese 0FFE*

0FFE_{16} … 0000 1111 1111 **11**10_2

Hlavní paměť … kapacita 64KB = 2^{16} B

16 bitová adresa

adresa		položky		
….				
0FF0	F0 F1 F2 F3	F4 F5 F6 F7	00 01 02 03	04 05 **06** 07
1000	08 09 0A 0B	0C 0D 0E 0F	FF FE FD FC	EF EE ED ED
…..				

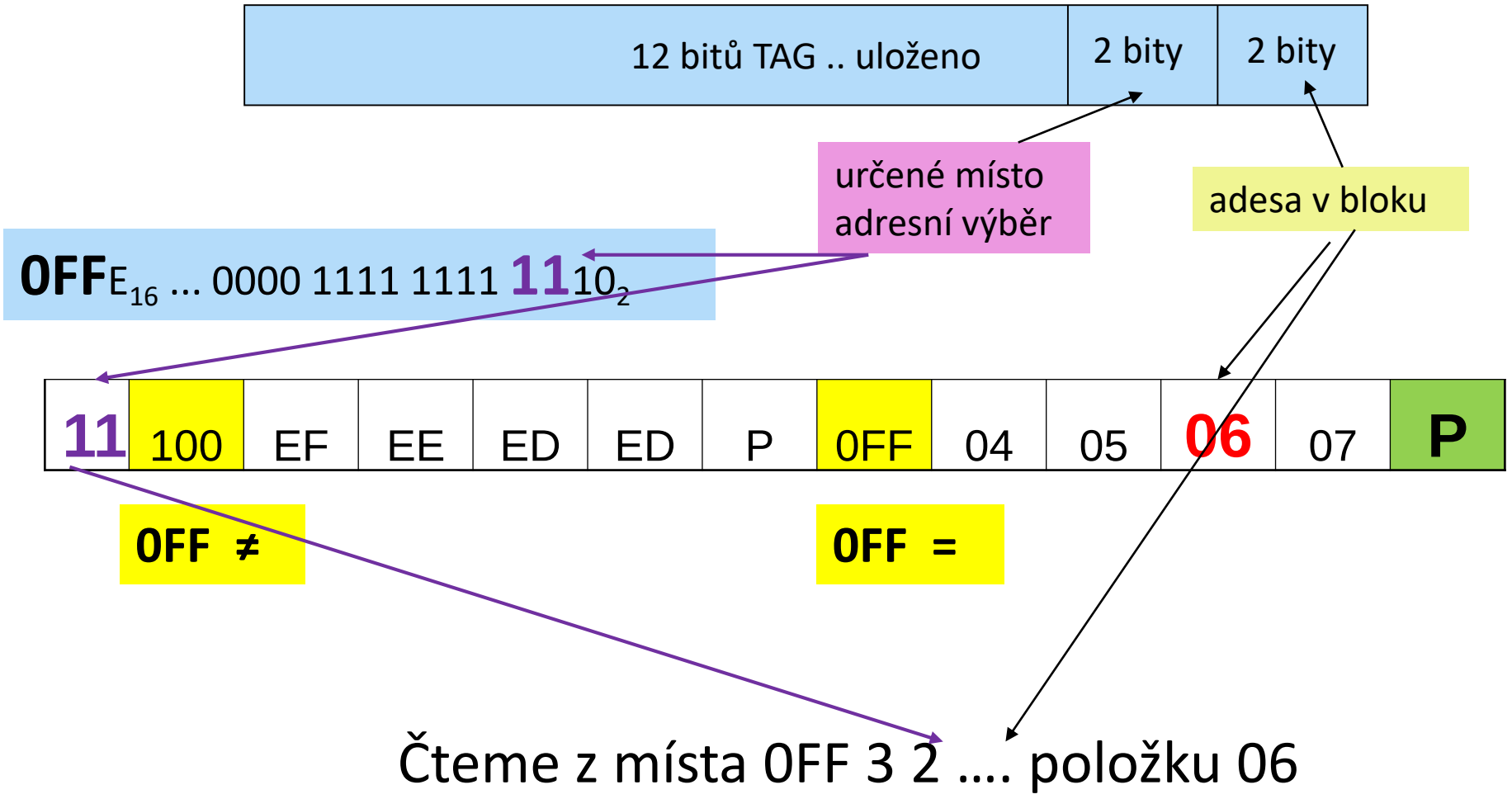
Cache .. 32B po 4B, s=2 … 4x4x2=32B … délka bloku 4B

adresa	TAG	Blok dat				platnost	TAG	Blok dat				Platnost
0	987	10	20	30	40	P	100	08	09	0A	0B	P
1	100	0C	0D	0E	0F	P	0FF	0C	0D	0E	0F	N
2	000	11	22	33	44	N	321	F1	F2	F3	F4	P
3	100	EF	EE	ED	ED	P	0FF	04	05	**06**	07	P

adresa	TAG	Blok dat				platnost	TAG	Blok dat				Platnost
0	987	10	20	30	40	P	100	08	09	0A	0B	P
1	100	0C	0D	0E	0F	P	0FF	0C	0D	0E	0F	N
2	000	11	22	33	44	N	321	F1	F2	F3	F4	P
3	100	EF	EE	ED	ED	P	0FF	04	05	**06**	07	P

určené místo
adresní výběr

adesa v bloku


12 bitů TAG .. uloženo
2 bity
2 bity
určené místo
adresní výběr
adesa v bloku
0FFE16 ... 0000 1111 1111 1110 2
11 100 EF EE ED ED P 0FF 04 05 06 07 P
0FF ≠
0FF =
Čteme z místa 0FF 3 2 …. položku 06

Příklady ze slidu 32

- Jsou v cache uloženy položky s adresou:

  0FF4, 0FFF, 1000 a co obsahují?

- Jak se změní obsah cache při uložení dat AABBCCDD na adresu DCBA?

data AABBCCDD na adresu DCBA

DCB_{16} 1010_2

adresa	TAG	Blok dat				platnost	TAG	Blok dat				Platnost
0	987	10	20	30	40	P	100	08	09	0A	0B	P
1	100	0C	0D	0E	0F	P	0FF	0C	0D	0E	0F	N
2	000	11	22	33	44	**N**	321	F1	F2	F3	F4	**P**
3	100	EF	EE	ED	ED	P	0FF	04	05	06	07	P

data AABBCCDD na adresu DCBA

DCB_{16} $10xx_2$

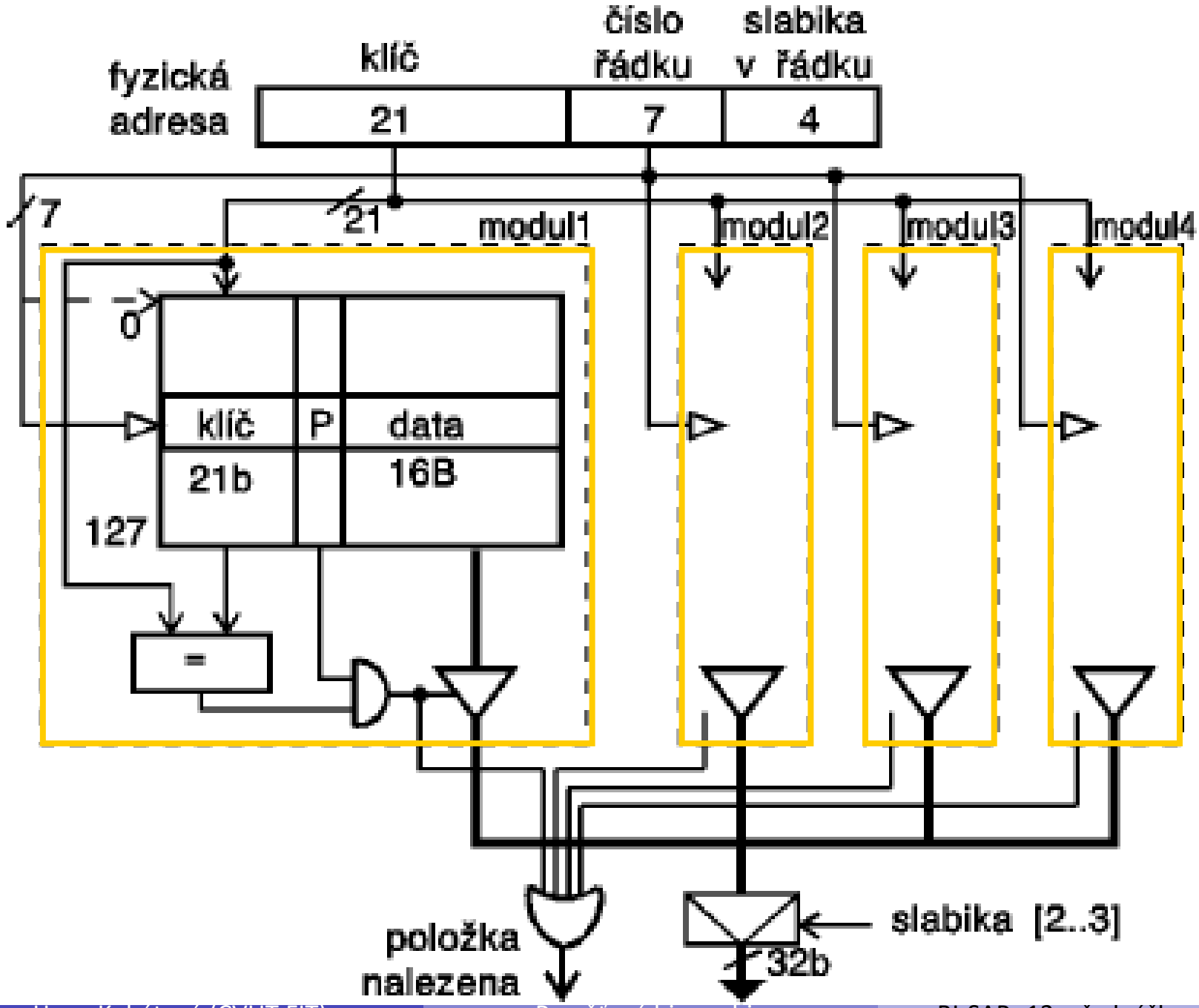
adresa	TAG	Blok dat				platnost	TAG	Blok dat				Platnost
0	987	10	20	30	40	P	100	08	09	0A	0B	P
1	100	0C	0D	0E	0F	P	0FF	0C	0D	0E	0F	N
2	DCB	DD	CC	BB	AA	P	321	F1	F2	F3	F4	**P**
3	100	EF	EE	ED	ED	P	0FF	04	05	06	07	P

Příklad - cache v procesoru I80487

- 32 bitová fyzická adresa
- 8 KB cache
- stupeň asociativity 4
- bloky dat 16 B (slabik)

(4 x 2 KB, 2KB → adresa 11b, ale blok 16B → adresa v bloku 4b,
Adresní výběr: 11 – 4 = 7b
Tag: 32 – 11 = 21)

- uvolňování položek - různé strategie, např. LRU, náhodný výběr z počtu položek v řádku (stupeň asociativity 4 ... tedy výběr ze 4 pozic)


fyzická adresa
klíč
číslo řádku
slabika v řádku
21
7
4
7
21
modul1
modul2
modul3
modul4
0
klíč
P
data
21b
16B
127
=
položka nalezena
slabika [2..3]
32b

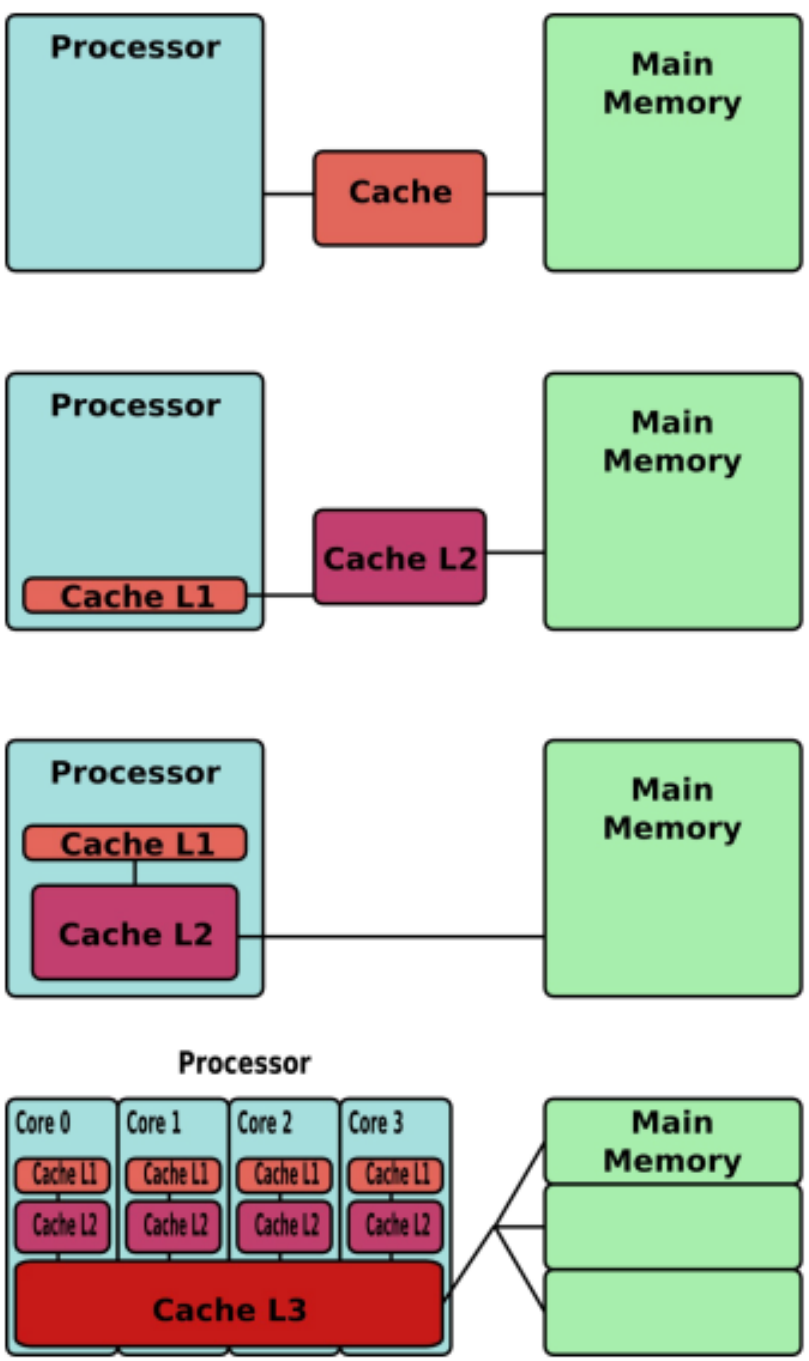
Otázky ke slidu 40

- Co a jak se změní při stejných kapacitách HP i cache, ale stupni asociativity 2?
- Kolik je třeba srovnávacích obvodů (XNOR) pro s=4 a pro s=2?
- Jak se změní rozložení adresy a struktura při zdvojnásobení velikosti cache?

Evoluce v hierarchii pamětí

vyrovnání rychlostí

- externí cache
- cache je na čipu
- vícejádrové procesory

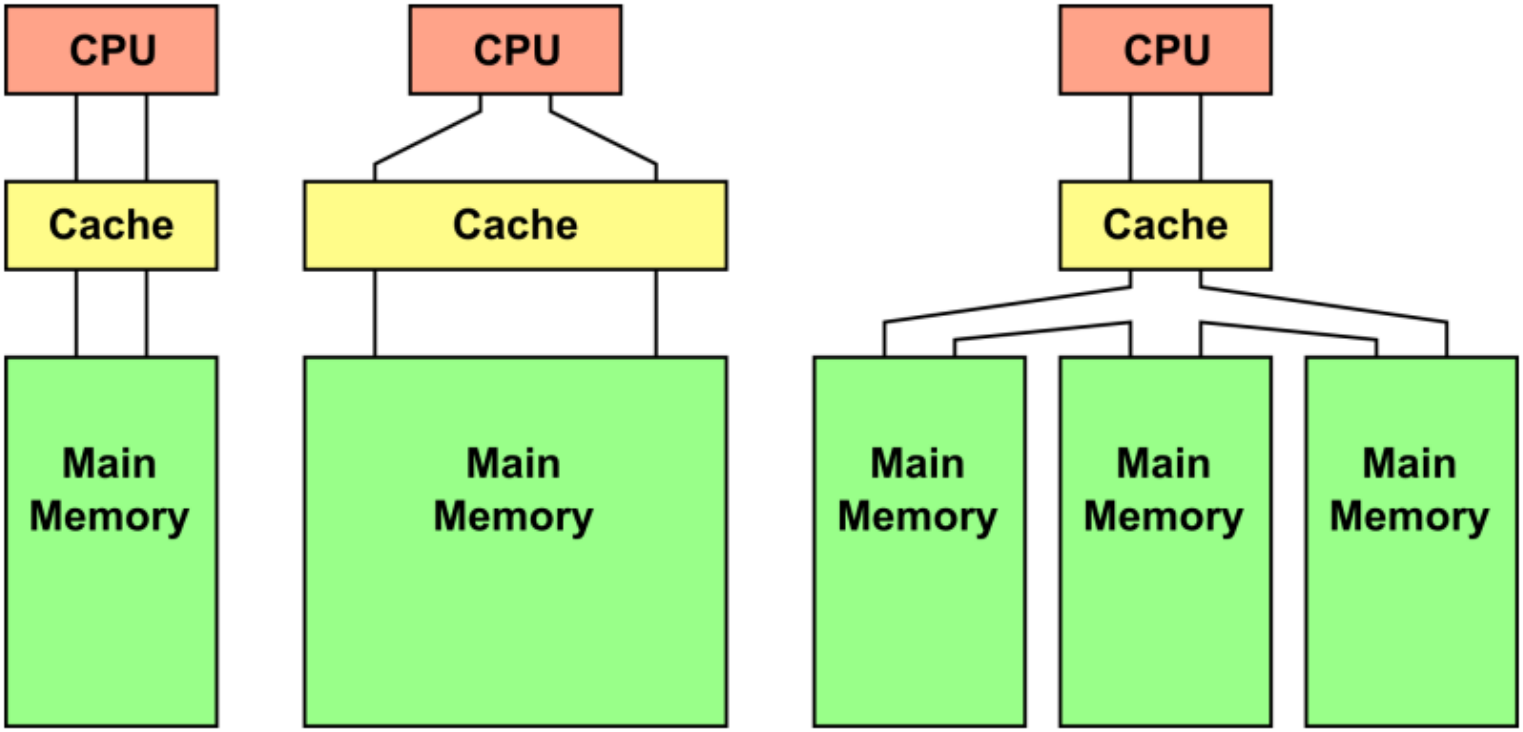


Organizace paměťového subsystému

Stejná šířka sběrnice mezi CPU a Cache, Cache a Hlavní pamětí

Sběrnice mezi CPU a Cache je rozšířena o multiplexor (zúžení k CPU). Cache a Hlavní pamětí propojena N širokou sběrnicí

Paměť organizována do samostatných banků. Datová sběrnice má stejnou šířku na všech úrovních.



Popis organizace paměťového subsystému

1. Jednoduchá organizace hlavní paměti
   - Šířka datového slova je zachována
   - Jednoduchost modelu, snadná implementace, velká latence
2. Paměť s širokým datovým slovem
   - Datová sběrnice mezi HP a cache je rozšířena na N násobek velikosti datového slova procesoru.
   - Dodatečný HW k realizaci výběru konkrétního slova z N slov dlouhého řádku v cache.
   - Významné snížení latence paměti při sekvenčním čtení v rámci N slov
3. Prokládaná hlavní paměť
   - Hlavní paměť je rozdělena na nezávislé moduly. Na každý z modulů jsou požadavky zasílány samostatně. Dochází ke zřetězení přístupů do HP.
   - Snížení latence, odpadá potřeba široké sběrnice, komplexnější řadič paměti