Adresovanie informácií v počítači

MMU – (Memory Management Unit)

-správa pamäťového systému počítača

Adresovanie hlavnej pamäte pomocou

- **fyzickej** adresy
- **logickej** adresy

Pamäťový systém

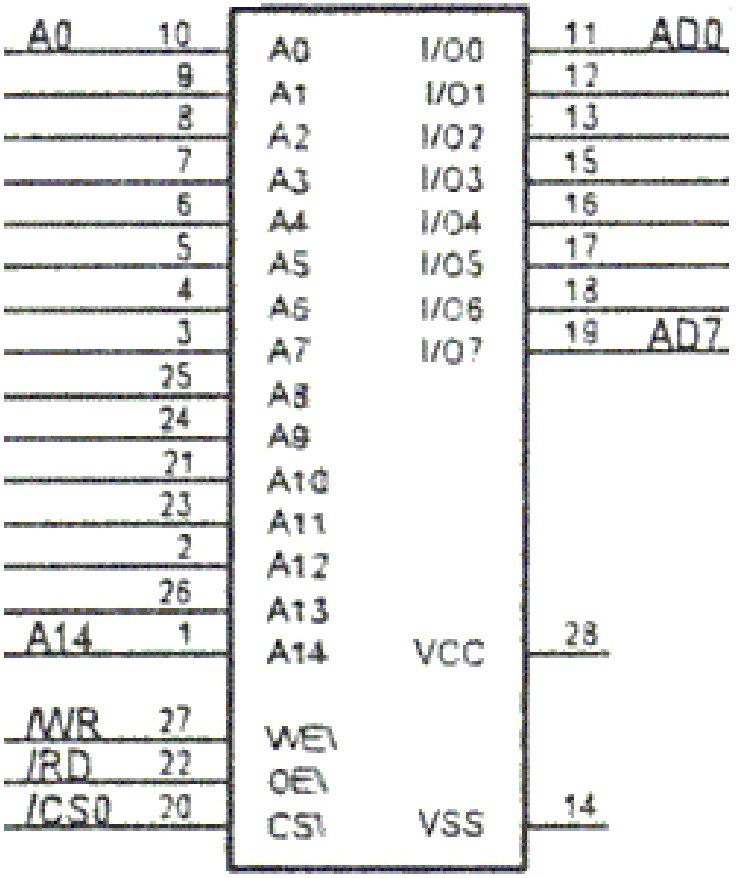
- **jednoúrovňový**
- **viacúrovňový**

Jednoúrovňový pamäťový systém

Fyzická adresa

V malých systémoch sa používa výlučne **fyzická adresa**
(binárne vyjadrenie adresy v inštrukciách je zhodné s binárnym vyjadrením signálov adresnej zbernice pripojenej k pamäťovému podsystému).

Realizácia pamäťového systému pomocou IC (príklad)



Pamäť RAM (RWM)

62C256

Kapacita 256 kibit, organizácia

8x32 ki (8x32kiB)

-adresové vývody A0-A14 (15bit)

-údajové vývody D0-D7 (8bit)

-VCC napájanie

-VSS zem

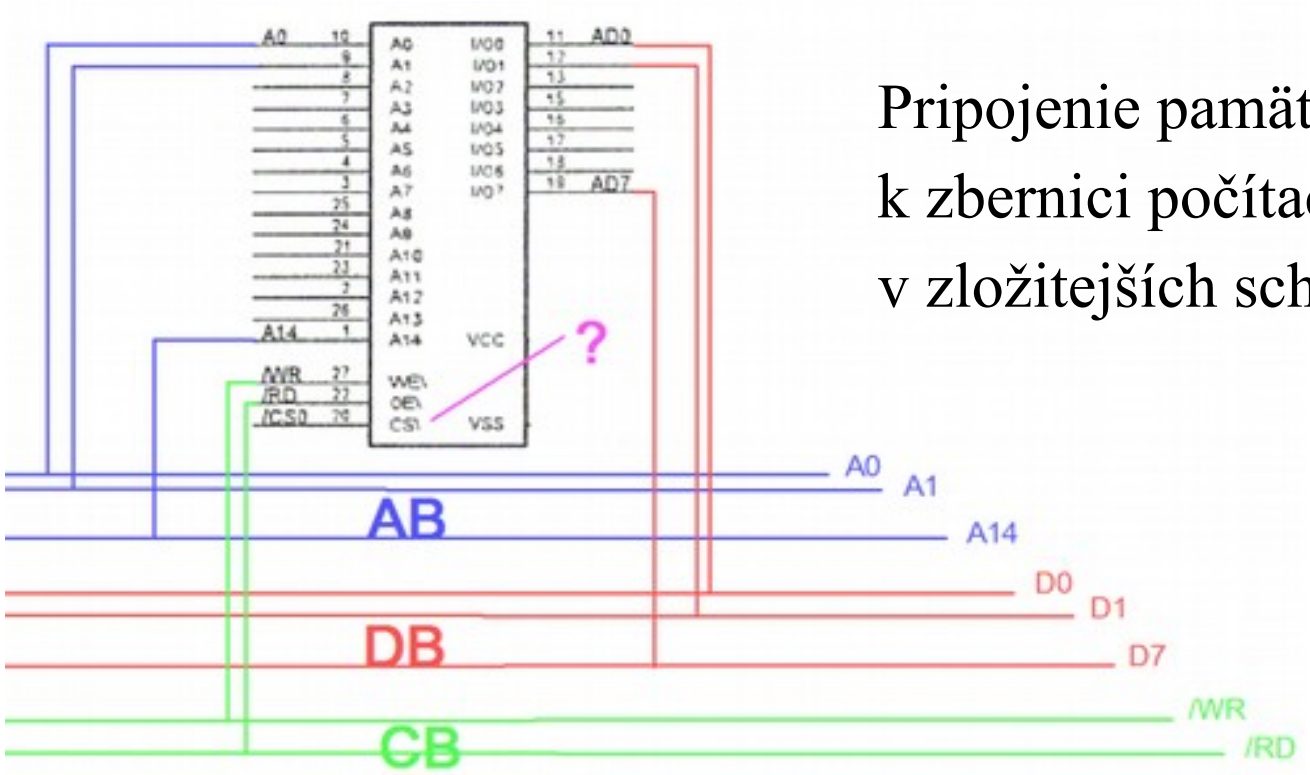
-ovládanie

WE\ - zápis do pamäte (log.0)

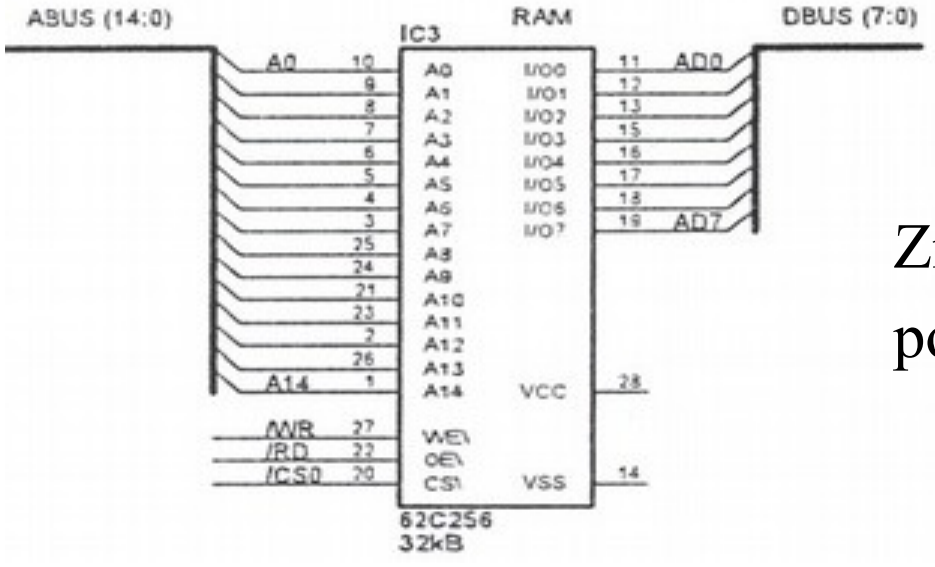
OE\ - čítanie z pamäte (log.0)

CS\ - výber obvodu (log.0)

Pripojenie pamäťového IC k zbernici počítača

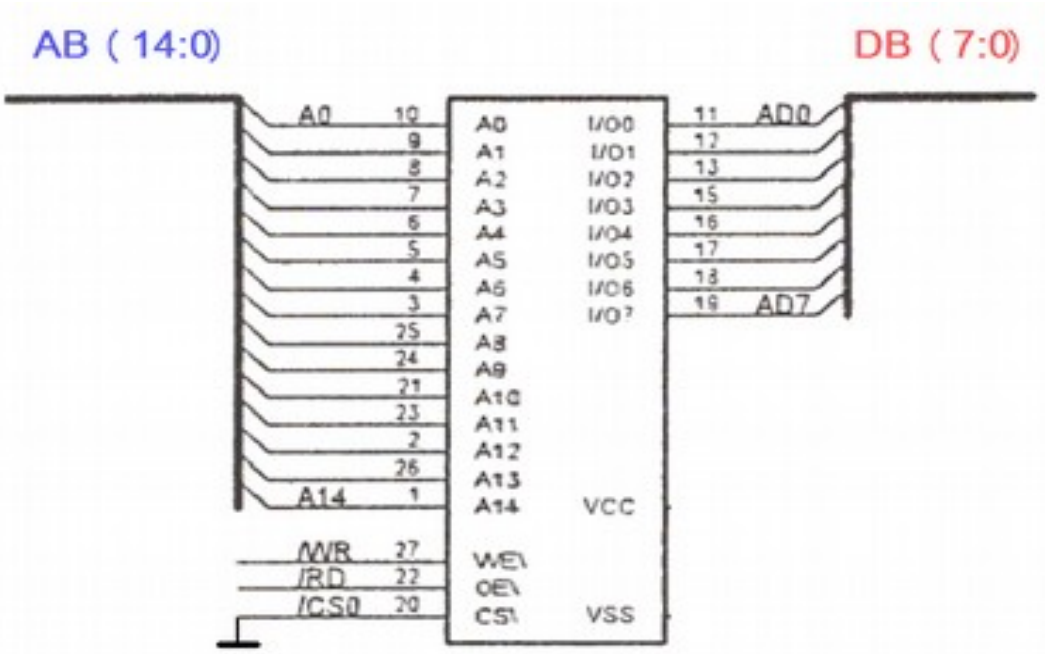


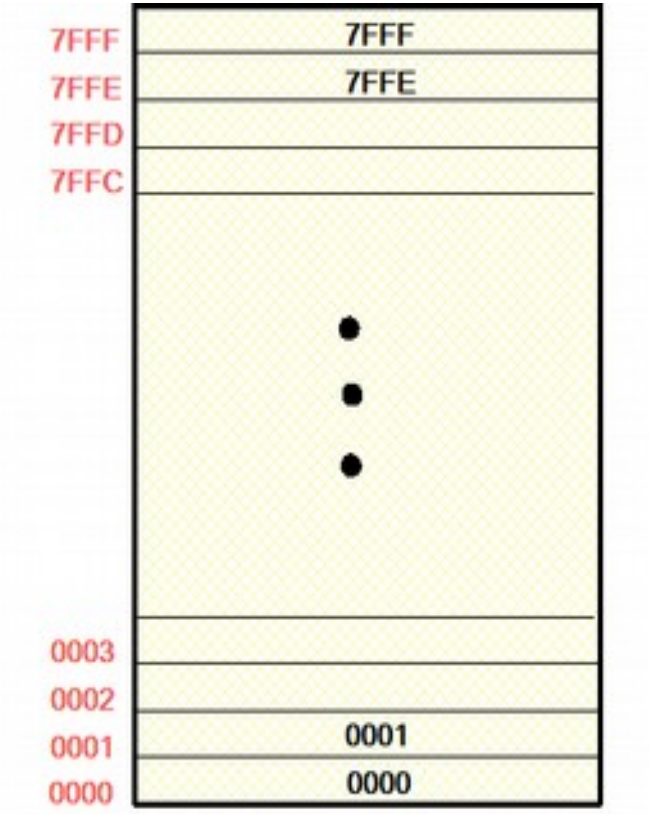
Pripojenie pamäťového obvodu
k zbernici počítača
v zložitejších schémach neprehľadné



Znázornenie pripojenia k zbernici
počítača v schémach zapojenia

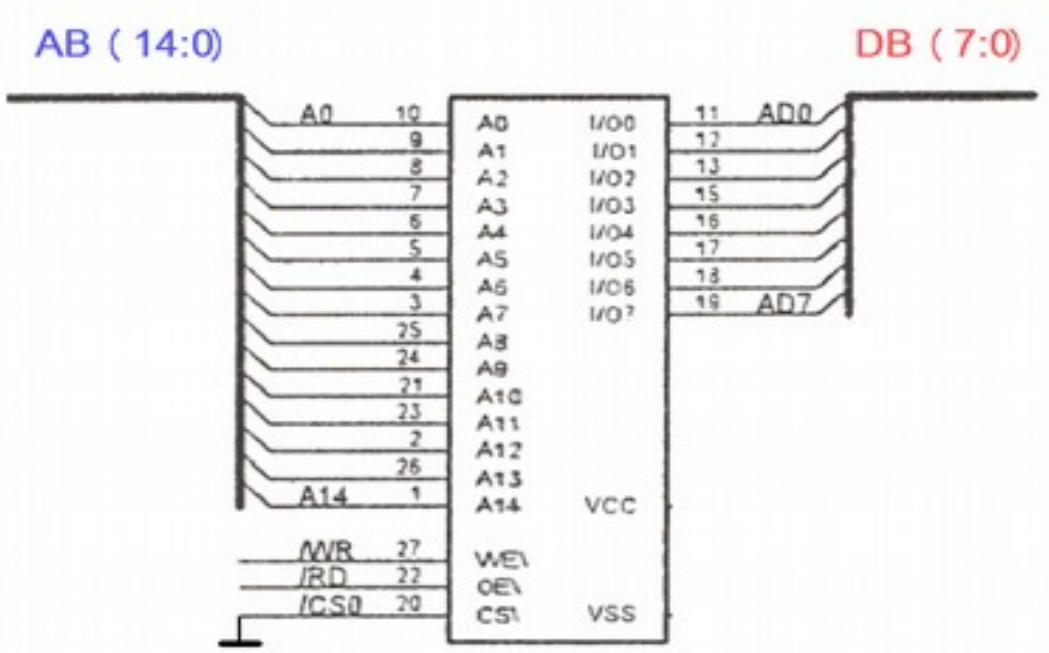
Prípad - šírka AB a počet adresných vstupov je rovnaký

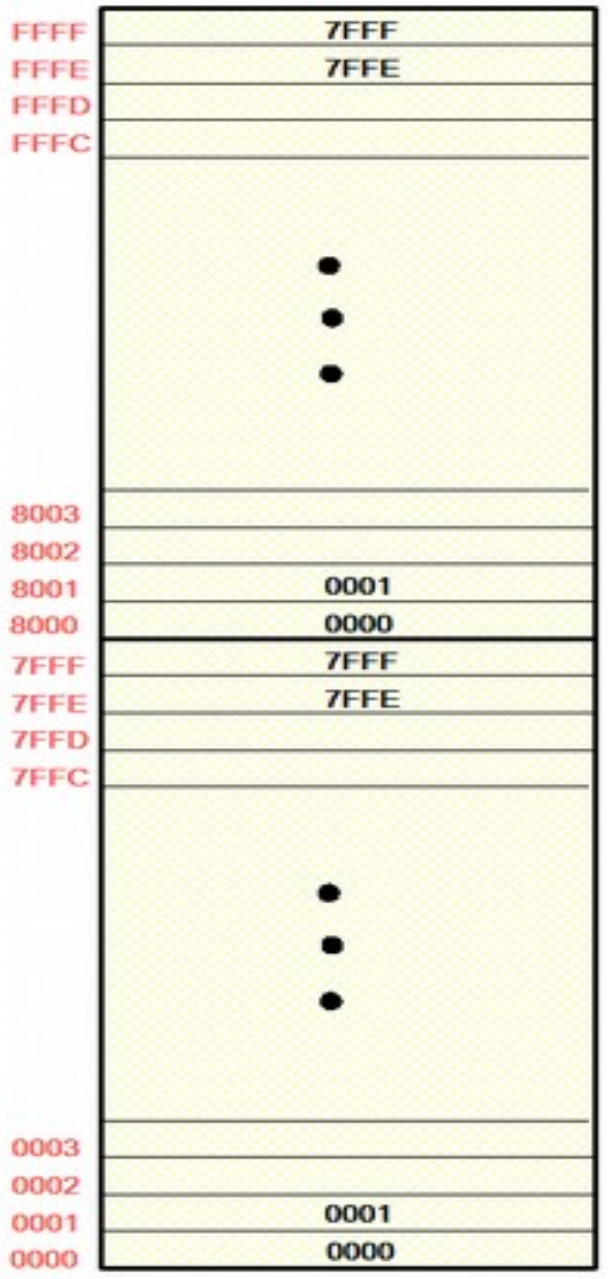




V tomto prípade je možné pripojiť CS trvalo na log. nulu,
iný obvod môže byť zapojený použitím špeciálneho zapojenia.
Vzťah medzi poradím bytu v pamäti a kódom na adresnej zbernici

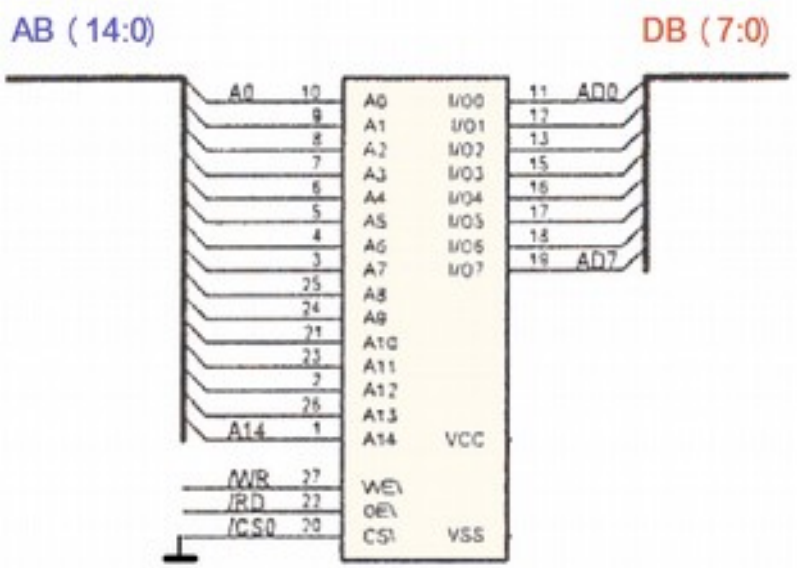
Prípad šírka AB je väčšia ako počet adresných vstupov

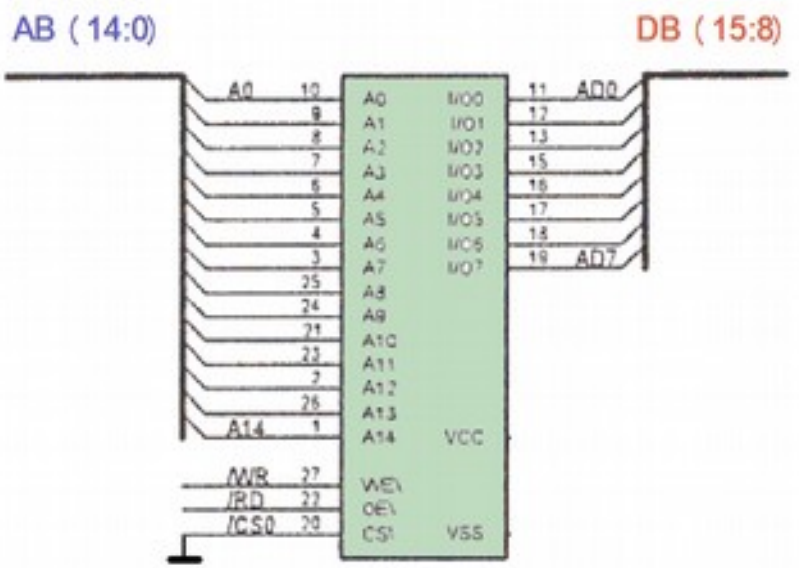


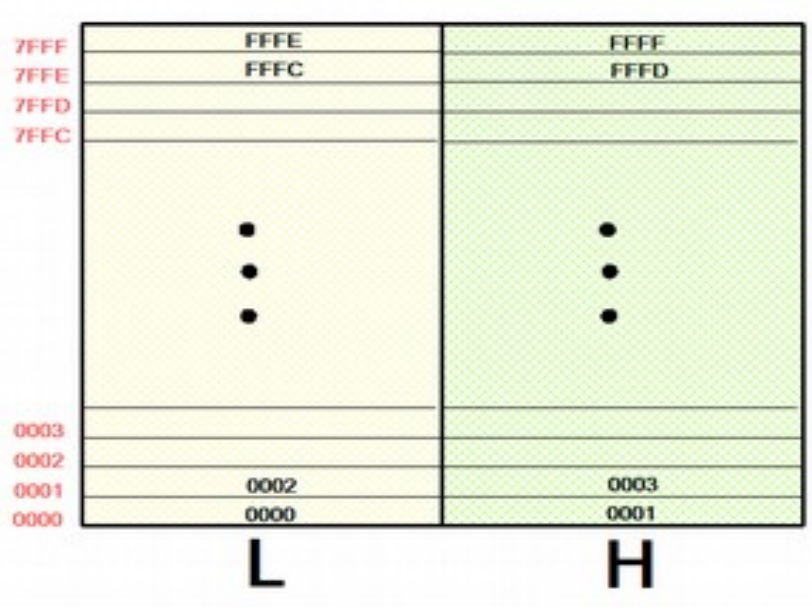


„Zrkadlenie pamäte“ prístup k bytom nezávisí od logickej hodnoty A15

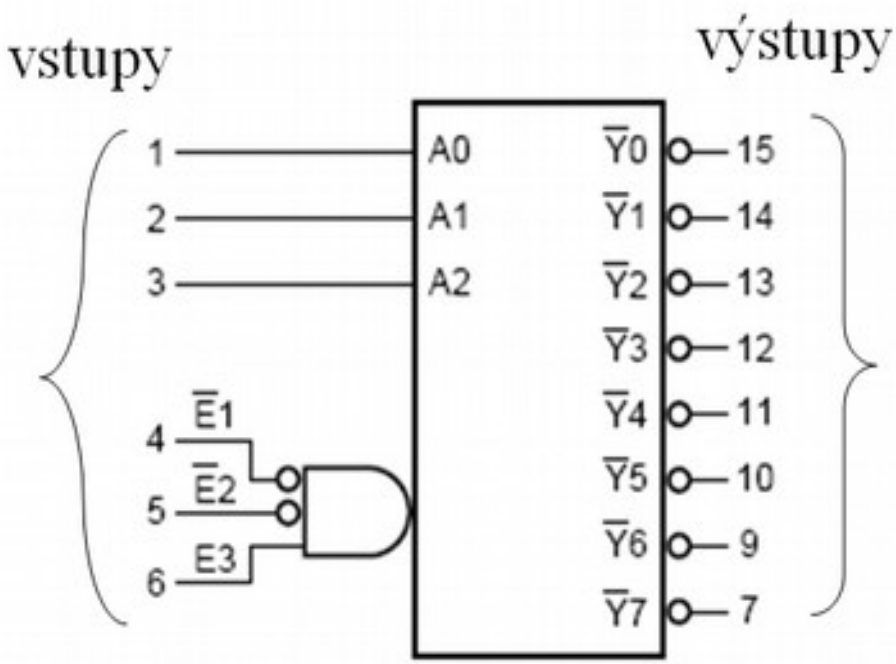
Prípad šírka AB a počet adresných vstupov je rovnaký
Šírka DB je násobkom počtu údajových vývodov pamäte



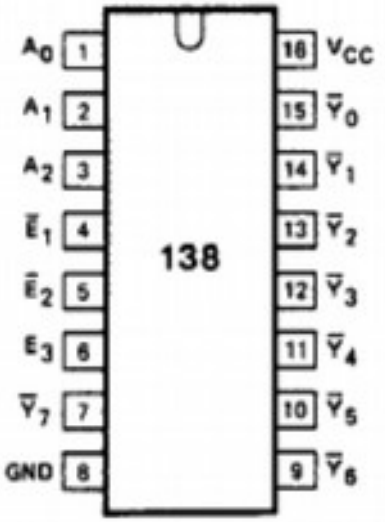




Adresový dekóder - použitie



MH 3205, 74HC138

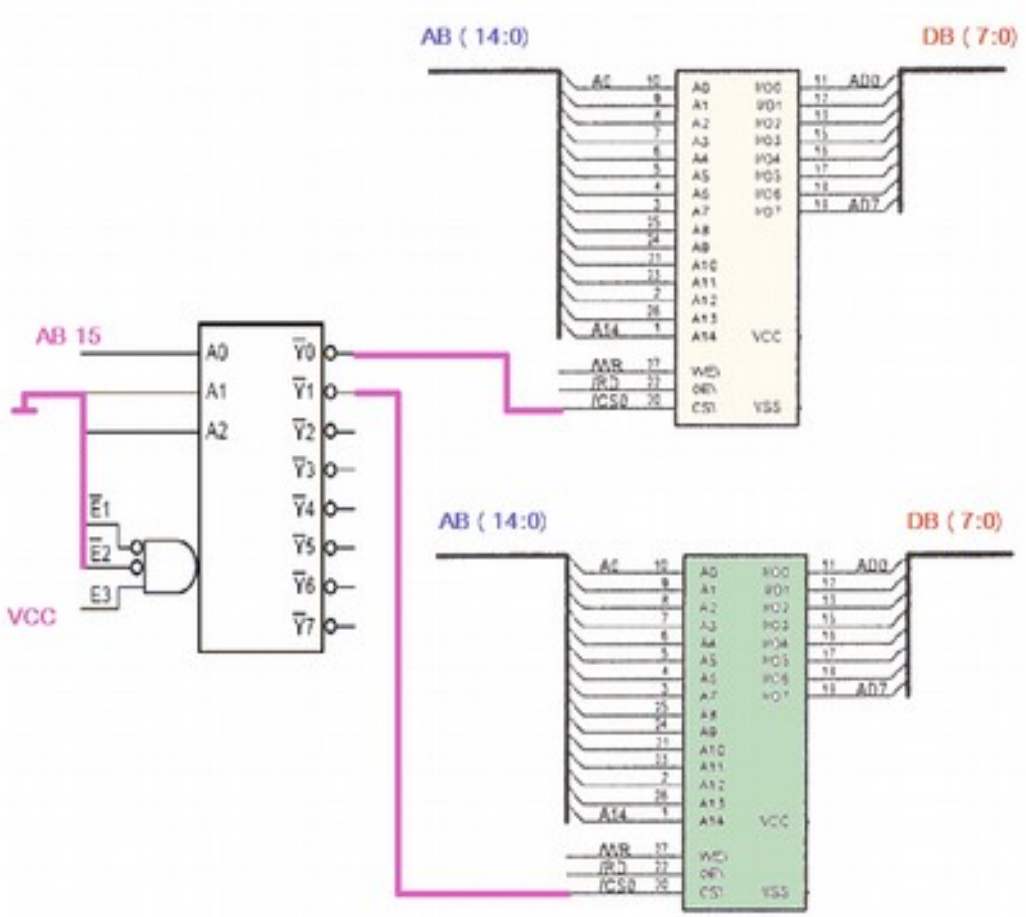


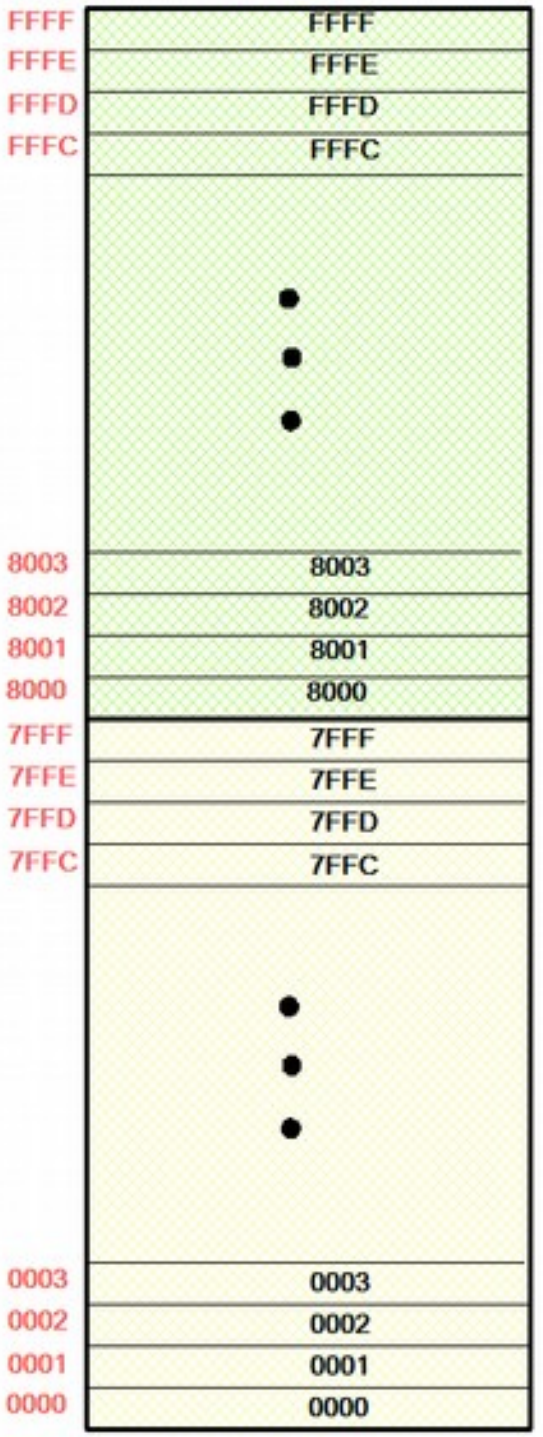
INPUTS						OUTPUTS							
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

Prípad šírka AB a počet adresných vstupov nie je rovnaký

AB (15:0)

Šírka DB je zhodná s počtom údajových vývodov pamäte





Logická adresa (virtuálna adresa)

reprezentácia adresy, ktorá jednoznačne určuje fyzickú adresu

Získanie fyzickej adresy

-výpočtom (príklad Ixx86 reálny mód)

-použitím tabuľky

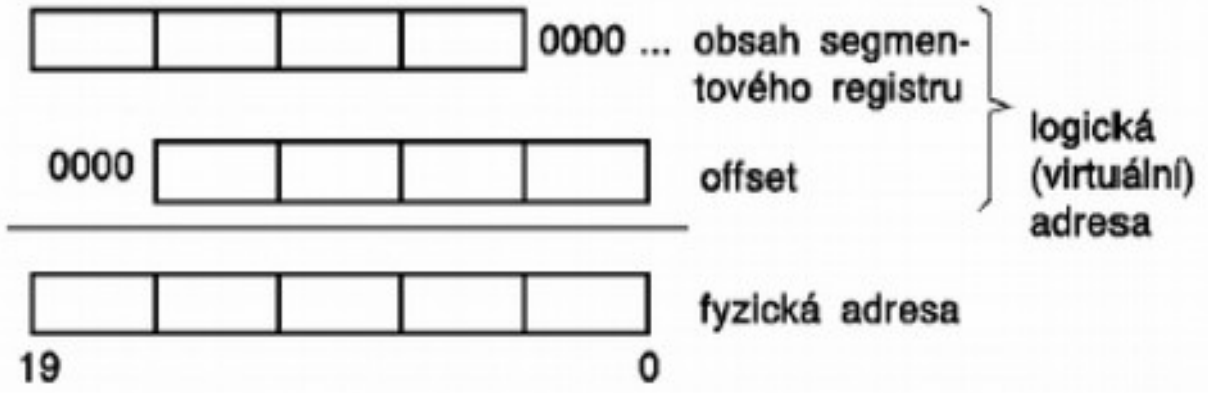
Výpočet fyzickej adresy (súvisí so smerníkmi typov near, far a huge)
Near= len offset. Far= max. 1 segment ale hocikde, Huge= a viac segmentov

logická adresa dvojica 16-bitových registrov (segment:offset)

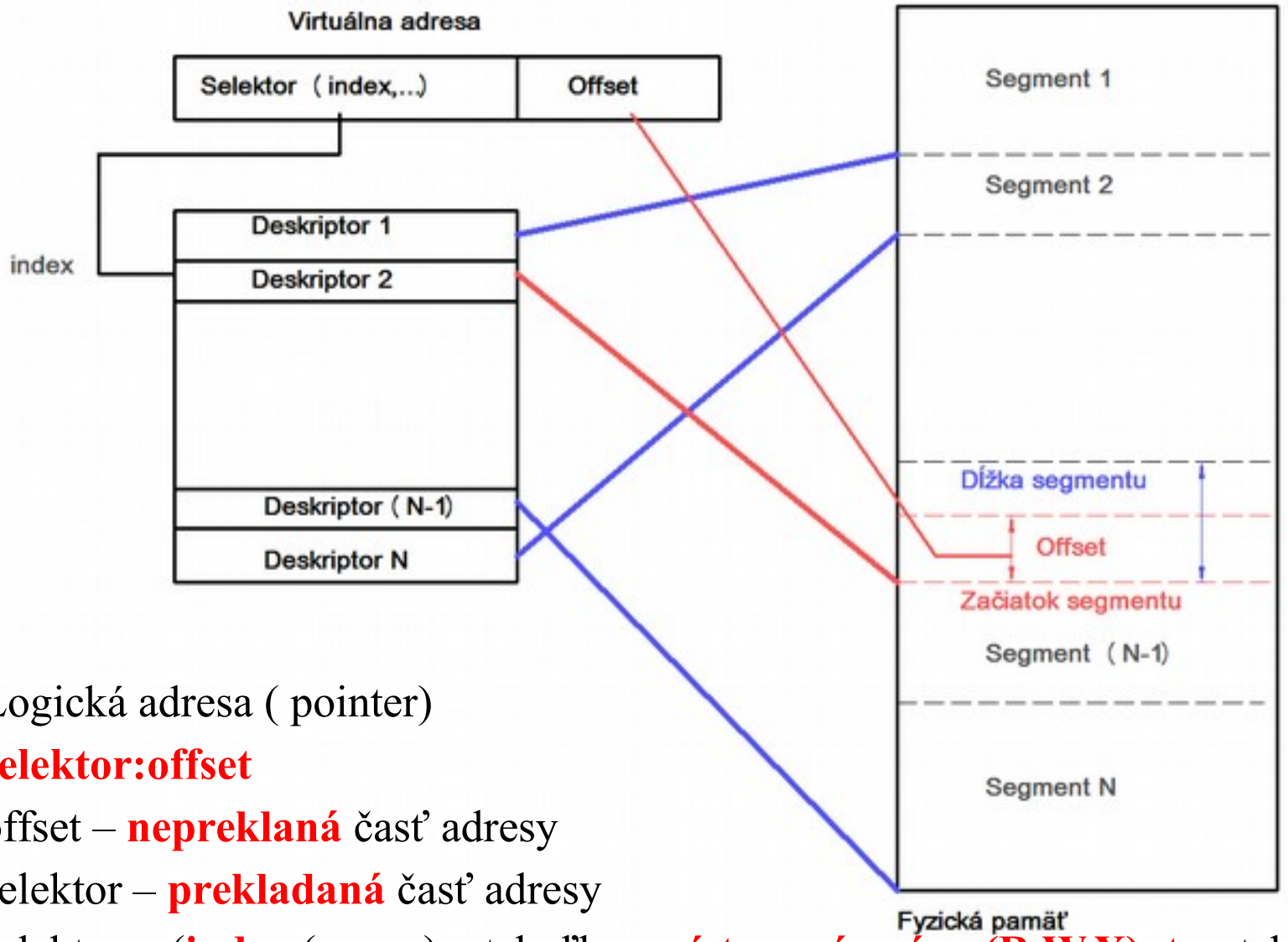
CS:IP (Code Segment : Instruction Pointer)

SS:SP (Stack Segment: Stack Pointer)

fyzická adresa – 20-bitov (adresový priestor 1 MB)



Adresovanie pomocou tabuliek



Logická adresa (pointer)

selektor:offset

offset – **neprekladaná** časť adresy

selektor – **prekladaná** časť adresy

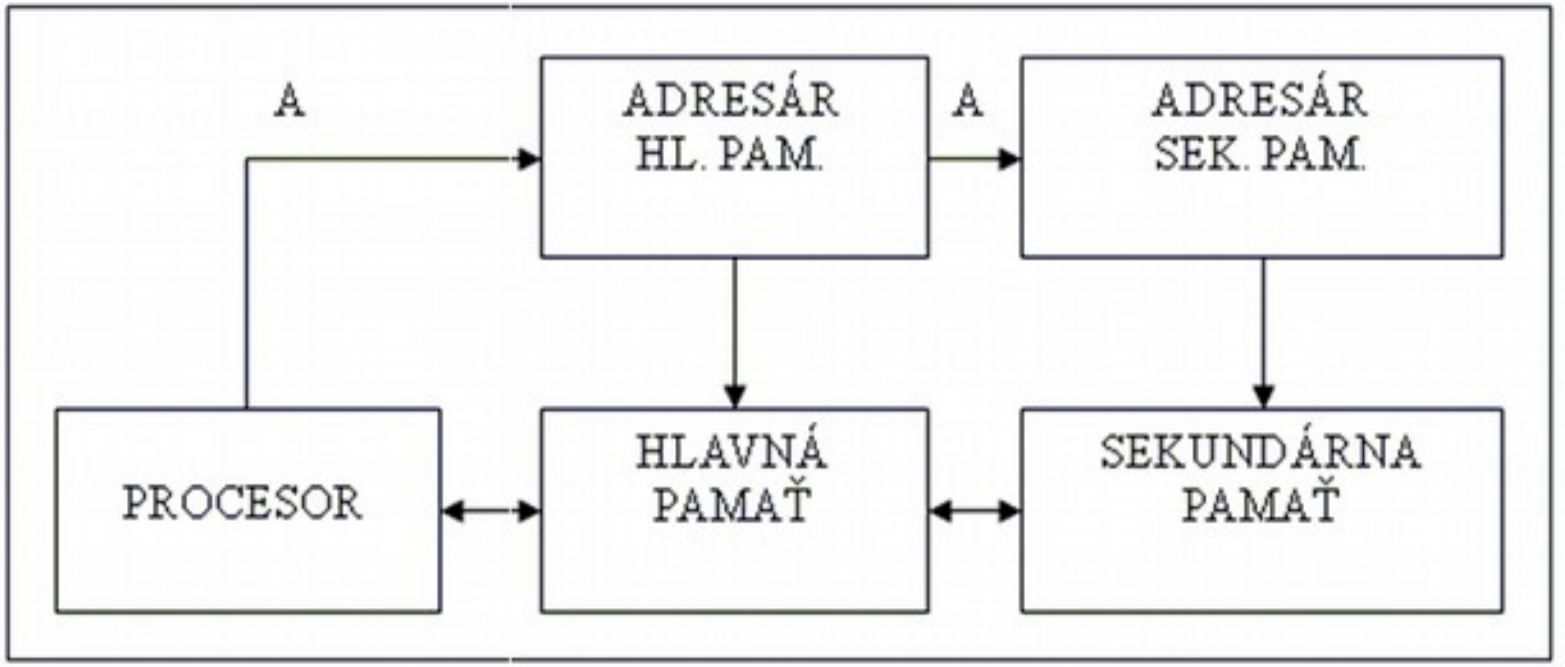
selektor = (**index** (posun) v tabuľke, **prístupové práva (R,W,X)**, **typ** tabuľky)

deskriptor = (fyzická adresa začiatku segmentu, dĺžka (limit) segmentu)

fyzická adresa = začiatok segmentu+offset

Nevýhody – malý počet tabuliek, pevná a obmedzná dĺžka tabuliek

Dvojúrovňový pamäťový podsystém



Rozšírenie adresného priestoru využitím sekundárnej pamäte

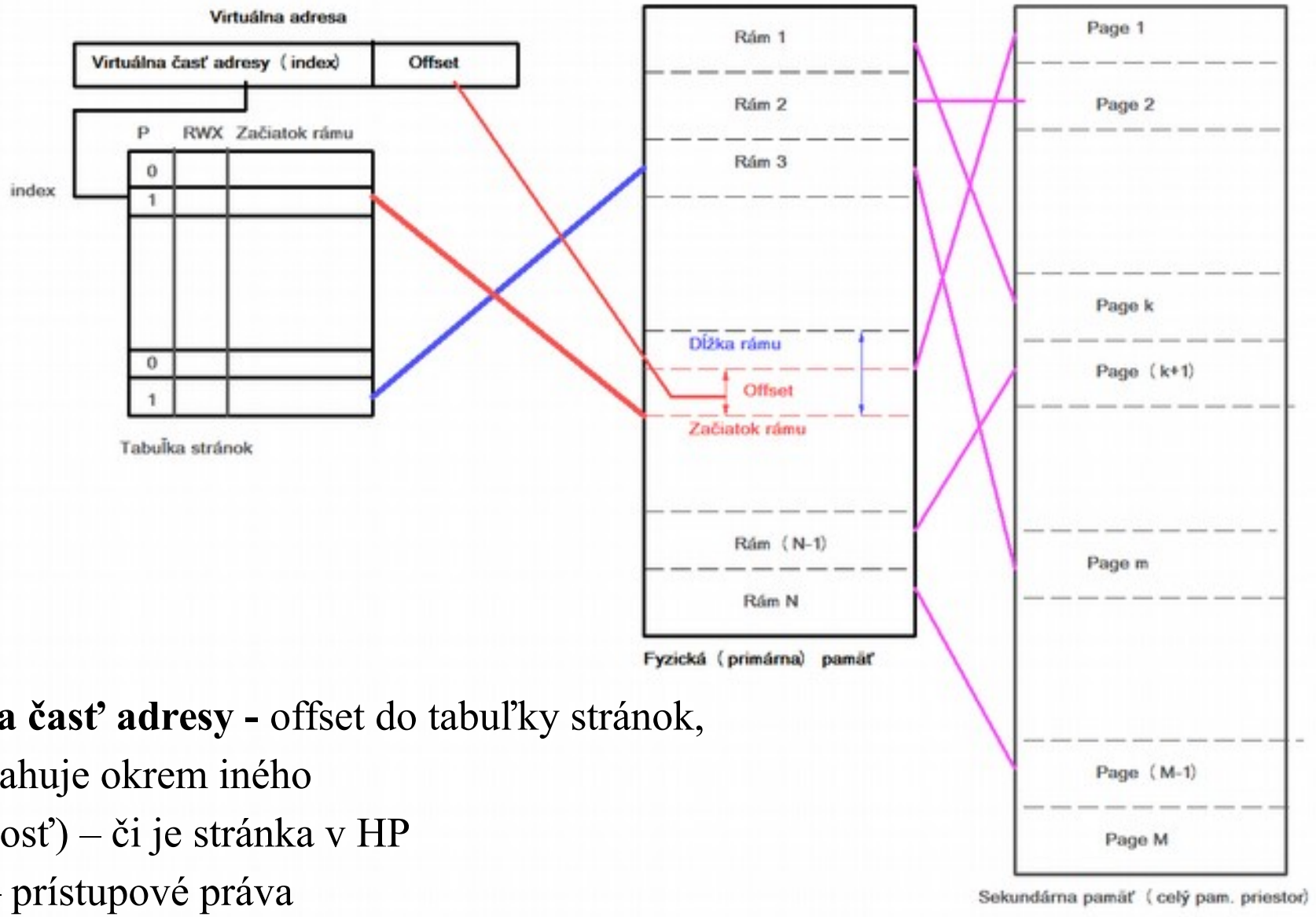
Problém nerovnakého prístupu k údajom

- HP – čítanie s ľubovolným prístupom
- sekundárna pamäť (SP) – čítanie so sprístupnením po blokoch

Tri hlavné mechanizmy prekladu virtuálnej adresy na reálnu adresu
- stránkovanie
- segmentovanie
- stránkované segmentovanie

Stránkovanie

- najmenší element – **stránka** (údajový blok **konštantnej dĺžky**), ktorý sa **presúva** medzi HP a SP
- HP je rozdelená na bloky konštantnej veľkosti = veľkosti stránky hovoríme im **rámy stránok** (veľkosť závisí od architektúry systému zvyčajne **1kiB** až **8kiB**)



Virtuálna časť adresy - offset do tabuľky stránok,
ktorá obsahuje okrem iného

- **P** (platnosť) – či je stránka v HP
- **RWX** – prístupové práva
- reálnu adresu začiatku rámu v HP, ktorá obsahuje odpovedajúcu stránku celého pamäťového priestoru (ten je celý iba v SP)

Tabuľka tiež obsahuje
- či položky stránky v bloku sú zmenené
- pomocné informácie pre rozhodovanie, či bude rám uvoľnený pre inú stránku

Výpadok stránky (page fault) – ak **P** signalizuje, že stránka nie je v HP, nasleduje výpadok stránky, čo znamená
- prerušenie vykonávania programu
- obslužný program uvolní jeden rám stránky a
- na uvoľnený rám umiestni stránku zo SP
- aktualizuje príslušné položky tabuľky stránok

V multitaskingovom prostredí úloha beží pokiaľ má údaje v HP, pri výpadku stránky sa preruší a spustí iná úloha (aktualizácia HP sa deje na pozadí inej úlohy)

Stratégia presunu stránok

Vyradenie stránky = veľká strata času, preto je dôležitá stratégia vyraďovania stránok, pred načítaním chýbajúcej stránky

Presun stránky, ak sa do nej nepísalo, prepíše sa novou

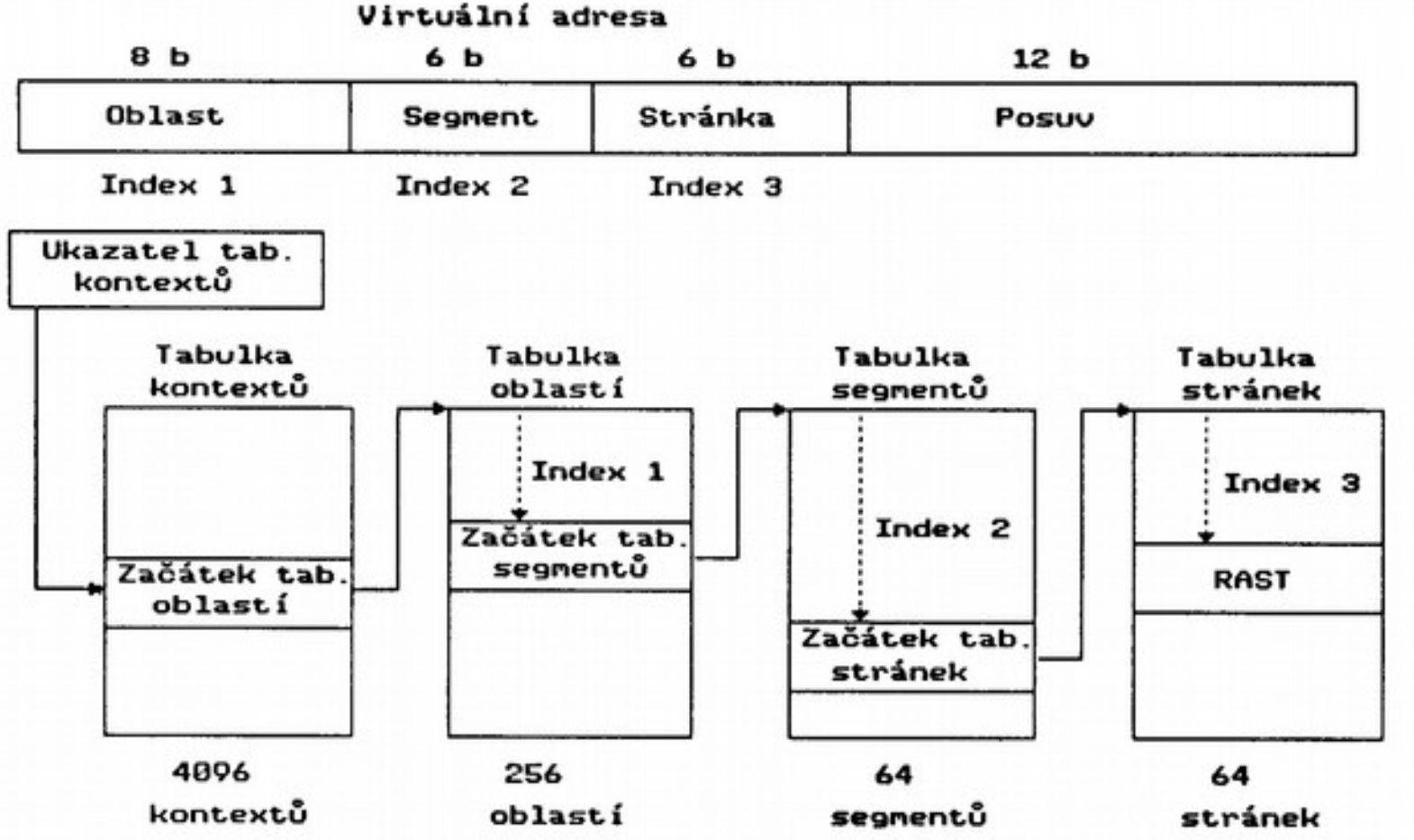
inak sa musí obsah stránky najprv uložiť do SP - ak sa požíva inclusive metóda- čo je v HP je aj v SP, exclusive -Ak je v HP nie je v SP

Najčastejšie stratégie vyraďovania stránok sú **LRU**, **FIFO**, **náhodný výber**, **LFU**

Urýchlenie prekladu adresy – použitie rýchlych CAM pamätí (**TLB** – Translation Lookaside Buffer) – obsahuje malý počet najčastejšie prekladaných adries

Možnosť odkladania tabuliek do SP rieši **viacúrovňový preklad** virtuálnej adresy

Viacúrovňový preklad virtuálnej adresy v procesore architektúry SPARC
(od Stanford University Network Microsystems =SUN microsystems, dnesOracle,
pôvodne na Univerzite Leelanda Stanforda mladšieho v mestách Stanford,CA a Palo Alto,CA)



Kontext – (úloha) má k dispozícii 256*64*64 stránok dĺžky 4kiB

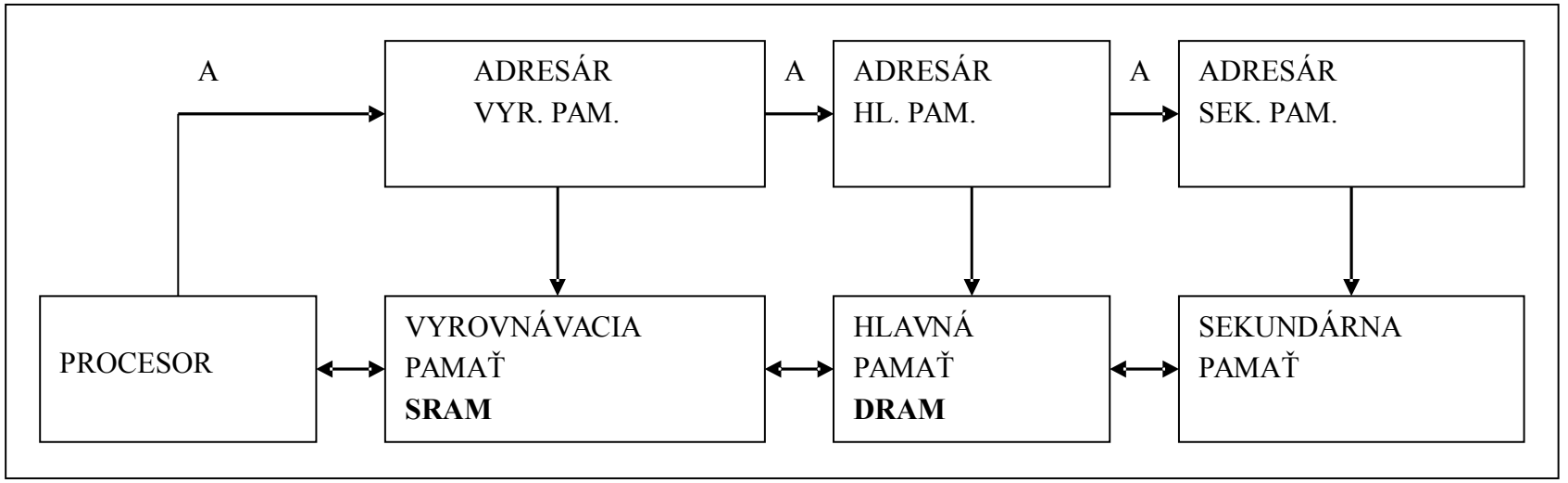
Súčasne môže bežať **4096** úloh

Vyrovnávacia pamäť (Cache memory)

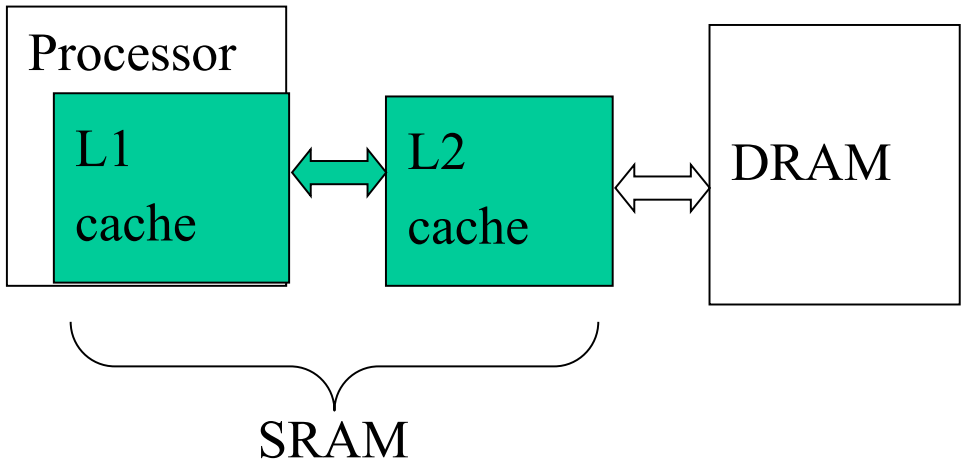
Hlavná pamäť veľmi pomalá (relatívne k rýchlosti procesorov), preto sa vkladá ďalšia úroveň pamäte:

-vyrovnávacia pamäť (VP) (musí byť podstatne rýchlejšia ako HP cca. 10 krát)

- slúži na urýchlenie najmä čítanie (aj zápis) do HP
- preto trojúrovňový pamäťový systém



Prvýkrát (IBM S/360 model 85 r.v. 1968) – označenie „cache“(skrýša),
iní výrobcovia označujú ako „buffer memory“
- pre programátora je „ **úplne priehľadná**“ (nedá sa adresovať
ani na úrovni strojového kódu) – činnosť VP riadi riadiaca jednotka (RJ)
- kapacita VP je **zlomkom kapacity** HP
- sú v nej uložené **kópie častí HP** (v trojúrovňovom systéme teoreticky
môže byť zapísaná tá istá informácia 3-krát (inclusive architektúra)
- dnešné počítače majú dve-tri cache pamäte. L1 cache je menšia
a je súčasťou porcesora a L2 a L3 cache je väčšia a je zapojená medzi procesor
a HP. (môže byť aj na to istom chipe)
Cache je statická pamäť.



Stratégia VP vychádza zo štatisticky overených vlastností programov pri práci s pamäťou. Sú to princípy:

-časovej lokality (vysoká pravdepodobnosť potreby informácie z tej istej adresy v krátkej budúcnosti

- a **miestnej lokality** (vysoká pravdepodobnosť potreby informácie z pamäťového miesta s adresou blízkou danej adrese), preto sa presúva blok so susednými bajtami („smerník subnear“)

Zásadný problém – ako realizovať predvýber blokov do VP, aby **CPU čítal len z VP** – neexistuje úplne uspokojivá stratégia:

- prvý presun až pri **požiadavke čítania** (on demand)
- ak je vyžiadané slovo, ktoré nie je vo VP, najskôr sa číta žiadané slovo do procesora, až potom sa číta zbytok bloku do VP. Ešte raz – číta sa celý blok.

Špekulatívne dopredné čitanie

Procesor odhaduje, aké dáata budú potrebené a tie načíta do L1 cache

Nutnosť pri exclusive cache(AMD), pri inclusive a semi-exclusive cache (Intel)

nekritické, vhodné len pre typické úlohy, úspešnosť okolo 87%(uspokojivo 96+%)

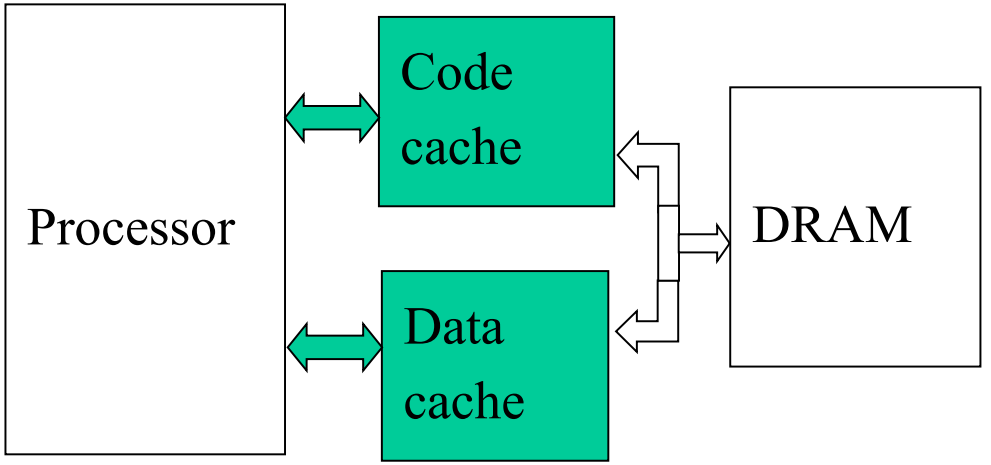
Princíp VP :

- obsahuje **adresár miest**, ktoré sa v danom čase nachádzajú vo VP
- podľa **obsahu adresára** sa dá rýchlo zistiť, či je daná informácia vo VP
- ak je tam, potom sa informácia prenáša do procesora z VP namiesto HP
- ak RJ zistí, že vo VP nie je požadovaná informácia, prečíta ju do CPU a VP
- očakáva sa vysoká pravdepodobnosť opakovaného čítania požadovanej informácie v krátkom časovom intervale
- okrem požadovanej informácie sa do VP presúva niekoľko susedných adries (**blok**) (dĺžka presúvaných blokov **8** až **32** bytov, extrémna veľkosť **128** bytov)

Definovanie bloku: ak je adresa HP 16-bitová a posledné 4-bity „nahradíme“ x-kami, získame blok veľkosti 16 bytov, (slov).

Počet blokov pre náš príklad.: $2^{(16-4)} = 4096$

Ďalšia otázka, čo presúvať do VP (pôvodne sa presúvalo všetko, čo bolo požadované [údaje aj inštrukcie]), v súčasnosti sa presadzuje koncepcia oddeleného čítania údajov a inštrukcií, čo umožňuje súčasné čítanie údajov a inštrukcií.



Adresovanie vyrovnávacej pamäte

-Informácie uložené vo VP sa volajú **reálnou adresou** t.j. ich adresou v HP

-VP je podstatne menšia, adresa nemôže priamo adresovať miesto vo VP, ale musí byť **prekladací (konverzný algoritmus)** na hľadanie informácie vo VP

-Stačí vymyslieť **algoritmus hľadania bloku** vo VP (hľadaná informácia je súčasťou bloku vo VP)

- Na vyhľadanie v bloku stačia najnižšie bity adresy (označujú sa ako „**číslo slova v bloku**“)

-Výhodou VP je jednoduchší adresný dekóder =>
rýchlejšie dekódovanie.

Ideálna cache obsahuje toľko položiek, že sa do nej zmestí celá HP, adresovanie položky cache a HP by bolo zhodné, len by bola VP obrovská a skoro prázdna

Pri redukcii počtu položiek VP - opäť problém prehľadávania obsahu VP, sekvenčné prehľadávanie je pomalé.

VP - Cache pamäte bývajú organizované ako tzv. **asociatívne pamäte**.
Asociatívne pamäte sú tvorené tabuľkou (-ami), ktorá obsahuje vždy:
- stĺpec s názvom: **tagy** (kľúče), slúžia na vyhľadávanie v asociatívnej pamäti. Tag je časťou pôvodnej adresy.
- stĺpec **data** – uchovávané údaje
- stĺpec s názvom **info**rmácie o stave správnej funkcie pamäte. (platnosť/neplatnosť uložených údajov, „LRU“)

Riadok tabuľky = „trieda“ = „rám“ =„riadok“=slot (tag, data, info)

Pr.:

Hlavná pamäť:

- 64kiB (16 bitov adresy)
- 8ki blokov, každý po 8 B

Cache:

- 8B blok (3 bity adresy)
-1kiB (1024B), blok = 8B => 128 slot-ov (rámov)

Priamo mapovaná cache pamäť (Direct mapping)

C = 128 (počet slotov v cache)

M = 8k (počet blokov v HP) C << M

S – poradové číslo slotu

Priradenie blok HP do slotu Cache.

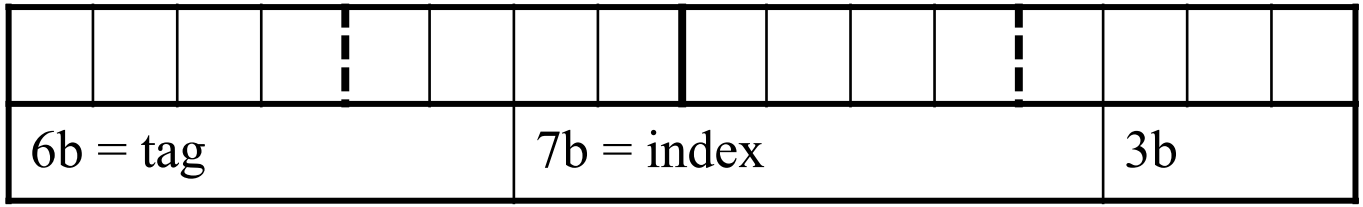
S = (adresa bloku HP) modulo (C)

T.j. Slotu 0 patria bloky: 0, 128, 256, … 8 064

a slotu 1 patria bloky: 1, 129, 257, … 8 065,

….

slotu 127 patria bloky: 127, 255, 383, . . . 8191.



Hlavná pamäť

Blok 0
Blok 1
Blok 127
Blok 128
Blok 255
Blok 256
Blok 8190
Blok 8191

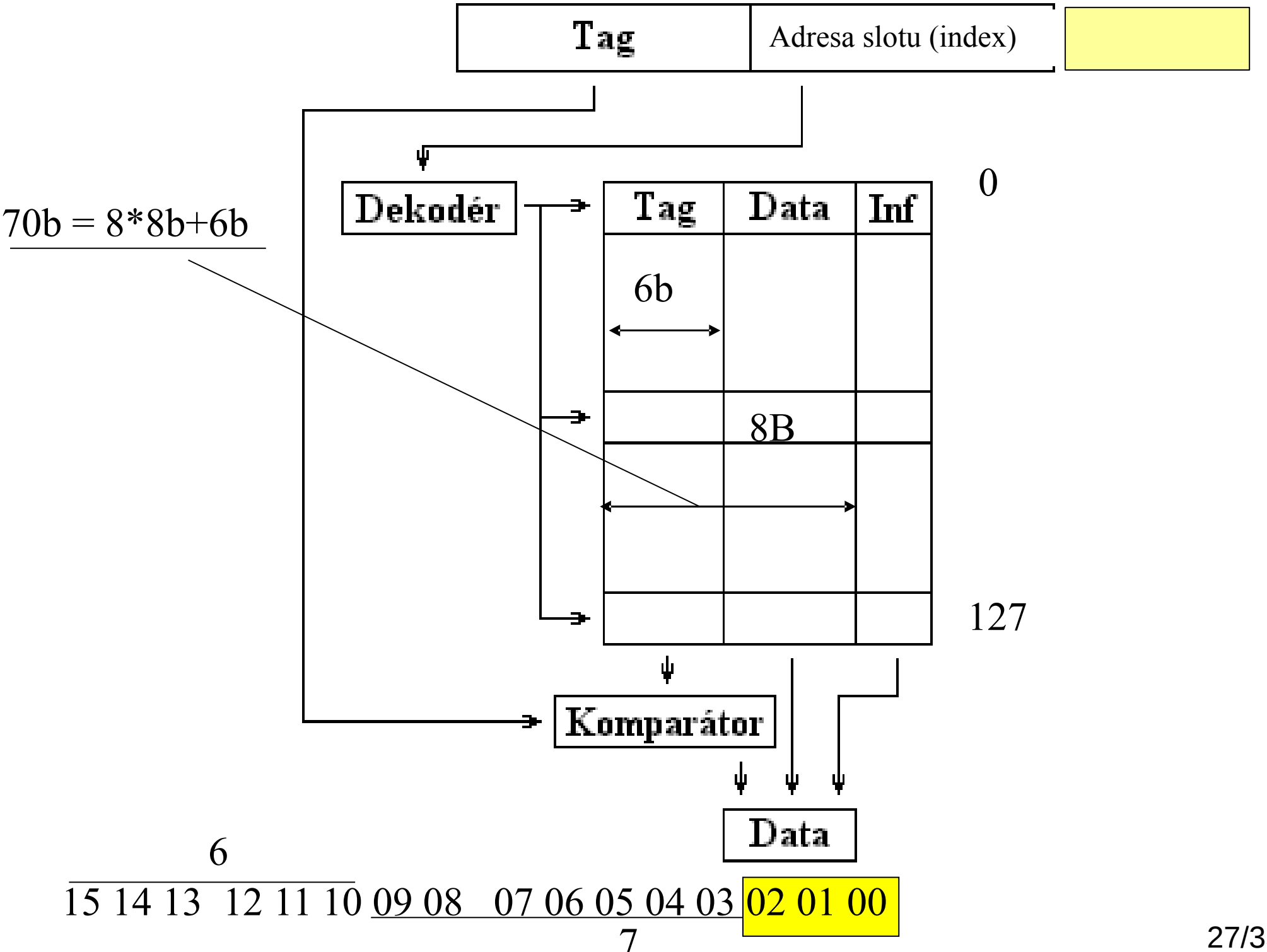
000000 0000000 xxx

000001 0000000 xxx

000010 0000000 xxx

Cache

Blok 0
Blok 1
Blok 2
Blok 3
Blok 126
Blok 127


Tag
Adresa slotu (index)
Dekodér
Tag
Data
Inf
0
70b = 8*8b+6b
6b
8B
127
Komparátor
Data
6
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
7

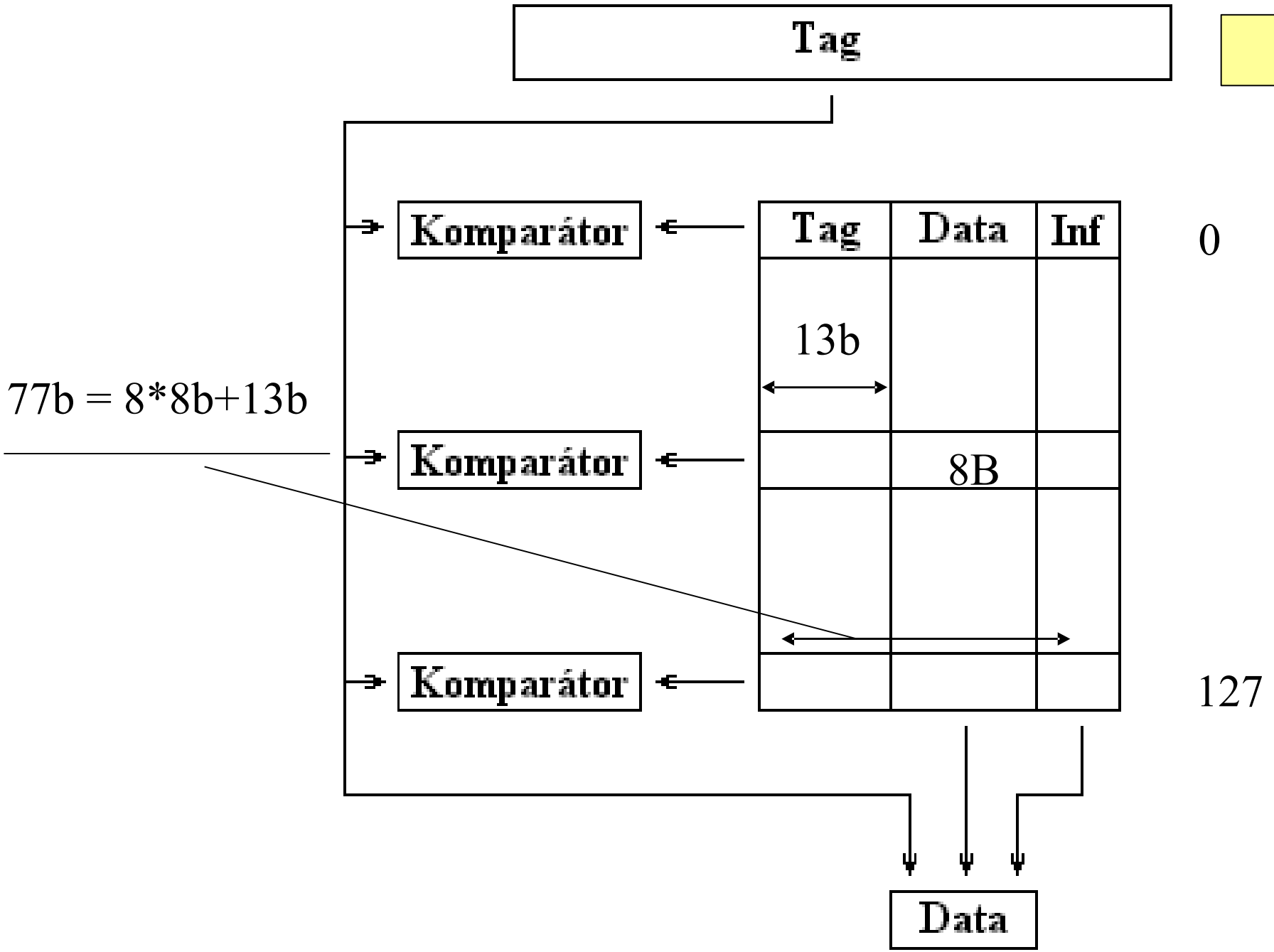
Priamo mapovaná cache pamäť:

Nevýhody: blok HP má pevnú polohu v cache
T.j. ak program pendluje medzi dvoma blokmi v tom istom slote, tak je nám cache „nepotrebná“.
Výhody: jednoduché a lacné

Plne asociatívna cache pamäť :

Tag – 13b
Blok – 3b
Blok môže byť presunutý do ľubovolného slotu.

Plne asociatívna cache pamäť



Plne asociatívna cache pamäť :

Nevýhody:

– zložitý blok riadenia cache

– požadujem veľa komparátorov

– veľká pamäť pre pamätanie si tagov

– nejasný mechanizmus výmeny blokov

Výhody:

– odstraňuje nevýhody priamo mapovanej cache

!!! Prakticky sa plne asociatívna pamäť nepoužíva. !!!

Asociatívna cache pamäť s obmedzením (stupňa n)

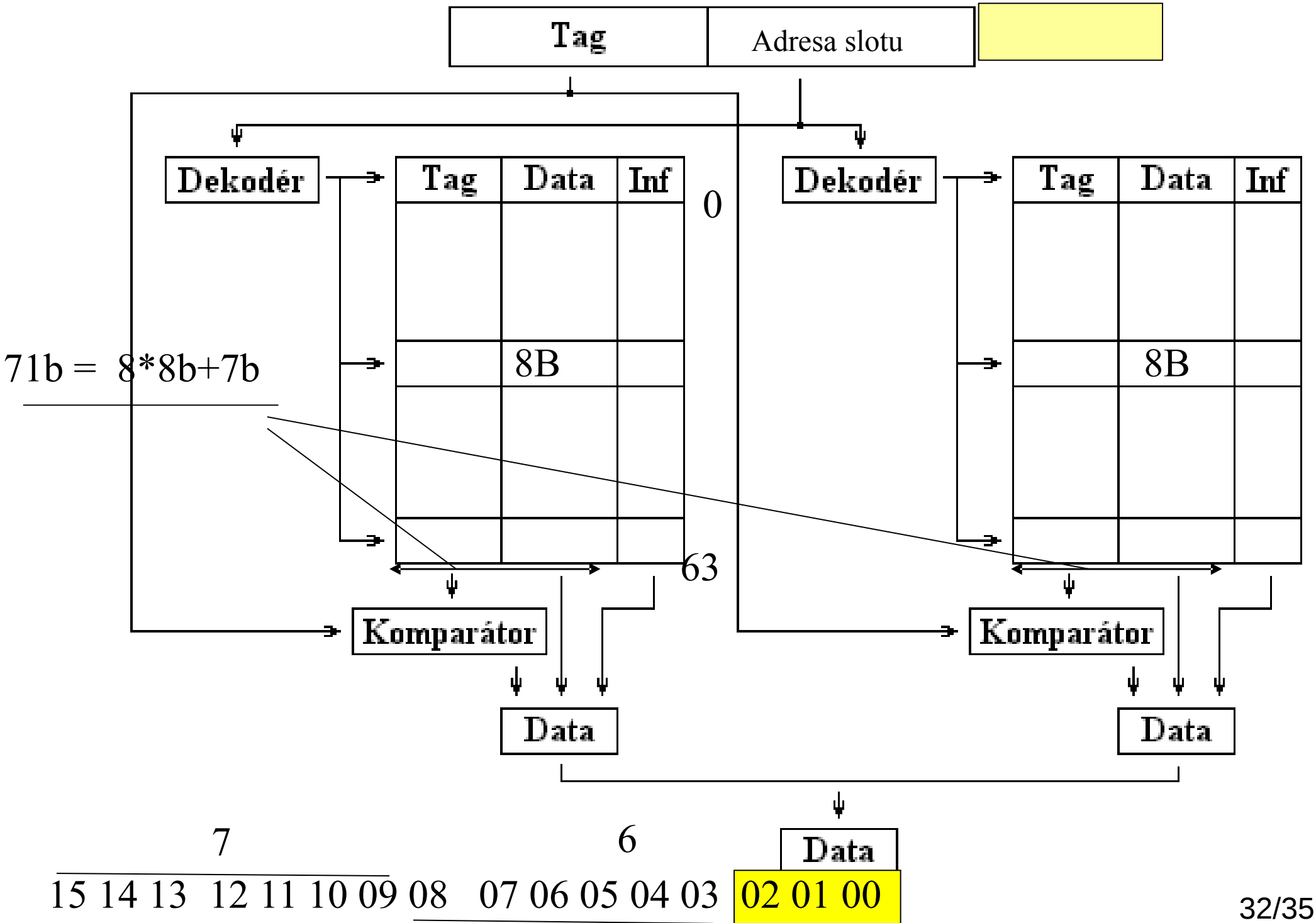
Odstraňuje nevýhody oboch predchádzajúcich.
Cache je rozdelená do **I** skupín a každá má **J** slotov.

C = I * J

Tento algoritmus umožňuje mapovať blok s adresou A
do ľubovolného slotu v skupine.

Pre „krajné“ hodnoty:
I = 1, J = C – plne asociatívna pamäť
I = C, J = 1 – priamo mapovaná cache pamäť
Najčastejšie – najjednoduchšie sa realizuje takáto cache pre n = 2

Asociatívna cache pamäť s obmedzením (stupňa n = 2)



Stratégia výmeny údajov medzi VP a HP

Podobné problémy ako virtuálnom adresovaní

Efektívnosť VP sa vyjadruje ako pomer
(**počet výpadkov VP**)/(**počet prístupov k VP**)

Po spustení programu (úlohy) na začiatku je toto číslo veľké, po cca **100 000** prístupoch sa ustáli na hodnote, ktorá závisí od **stratégie uvoľňovania blokov** a **stupňa asociativity** (pomer býva blízky **0,05**)

Používajú sa stratégie uvoľňovania **LRU**, **FIFO, LFU** a **náhodná**, preferuje sa obvodová jednoduchosť a najmä rýchlosť

Stratégia výmeny údajov medzi VP a HP

LRU (Least Recently Used – **najdlhšie nepoužívaný**)

Vyraďuje najdlhšie nepoužívaný blok dát
– zložitá realizácia
– používajú sa zjednodušené varianty stratégie LRU

FIFO (First In First Out) – v podstate **fronta** – vyraďuje sa blok dát, ktorý je v cache **najdlhšie**,

LFU (Least Frequently Used – **najmenší počet úspešných zásahov**)

Ak použijeme asociatívnu cache pamäť s obmedzením stupňa dva je najjednoduchší spôsob výmeny údajov: “posledne použitý“
Postačuje pridať jeden bit k dvom riadkom.

Zabezpečenie zhody údajov vo VP a HP

- **write through** – zapisovanie z procesora do VP a súčasne aj do HP (pomalé)
- **write back** – zapisuje len pri vyraďovaní bloku (3 spôsoby)
  - **zapisuje vždy**
  - **zapisuje len pri zmene obsahu**
  - **zapisuje len pri zmene** cez **pomocnú pamäť bloku** (zápis do pamäte sa uskutoční z pomocnej pamäte po prečítaní nového bloku) (**najrýchlejšie**)

Efektívnosť VP – odhad, ak je rýchlosť **VP 10x** väčšia ako HP potom v strednej hodnote je rýchlosť (čítanie/zápis) **cca 6x** vyššia pri použití VP oproti systému bez VP.

Literatúra:

1. **Hlavička, J.:Architektura počítačů. ČVUT 1998**