Adresovanie informácií v počítači

MMU – (Memory Management Unit)
- správa pamäťového systému počítača

Jednoúrovňový pamäťový systém

Adresovanie hlavnej pamäte
-fyzická adresa
-logická 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)

Logická adresa sa používala napr. pri procesoroch I8086 (Ixx86 – reálny mód)
16- bitové registre sa používali na adresovanie cez 20- bitovú adresnú zbernicu (adresný priestor 1MB)

Bázové adresovanie (segmentovaná pamäť)
- **segment** - časti pamäte premenlivej dĺžky
- **adresa v segmente** je relatívne vztiahnutá k začiatku (báze) segmentu
- **logická adresa** sa skladá z **báze segmentu** a **offsetu** (posunutia v segmente)

Báza segmentu je v segmentových registroch CS, DS, ES a SS (16-bitové registre)

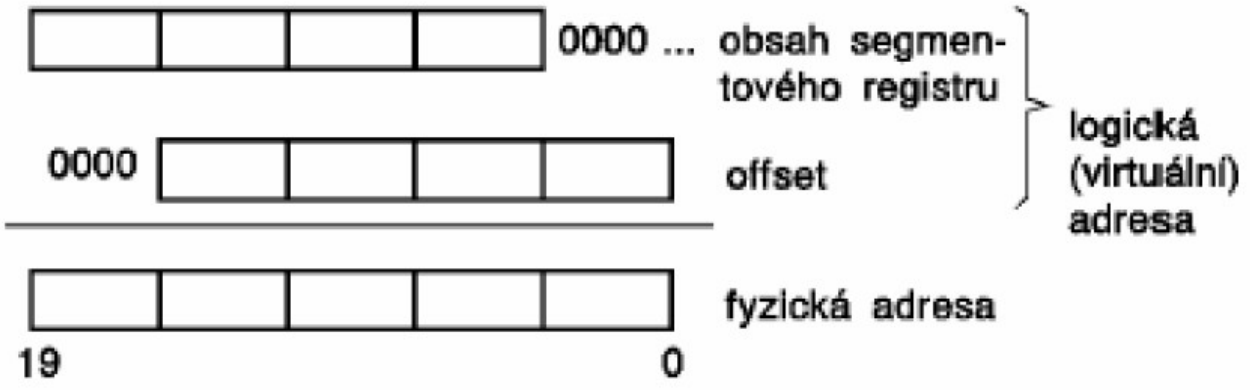
- **logickú adresu** tvoria dvojice registrov napr.

CS:IP (Code Segment : Instruction Pointer)

SS:SP (Stack Segment: Stack Pointer)

- **fyzická adresa** sa vypočítavala v časti procesora (BIU)

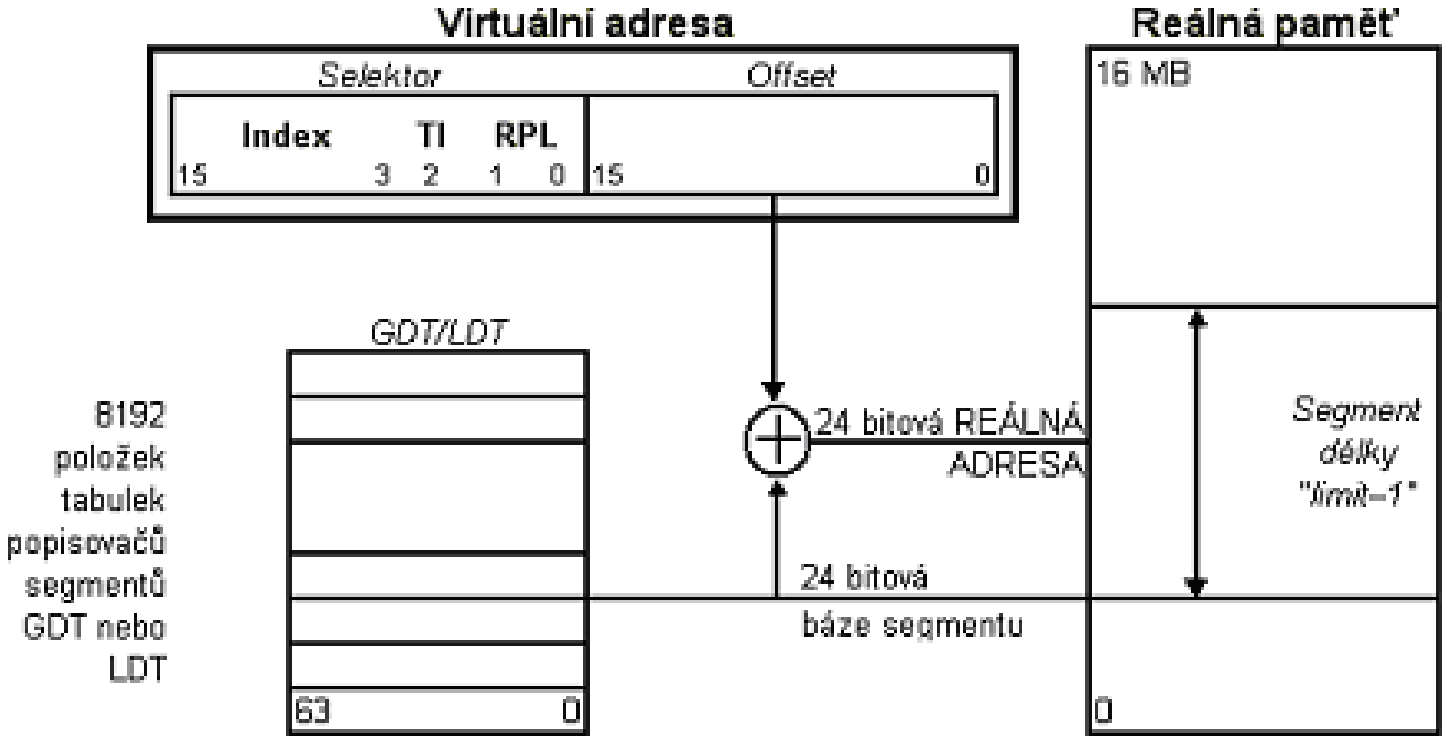
(binárny obsah smerníka bol zhodný s logickou adresou)



Začiatok segmentu na adresách 16*xS (xS – obsah segmentového registra), maximálna dĺžka segmentu 64kB)

Adresovanie pomocou tabuliek

I80286 - tabuľka deskriptorov (popisovačov) segmentu (protected mode (chránený režim))
logická adresa (pointer) **selektor:offset**



- deskriptory (8bytov) – GDT – globálne (pre systém), LDT lokálne pre aplikáciu

Štruktúra

2B – prázdne

1B - prístupové práva (R,W,X) (Read,Write,eXecute)

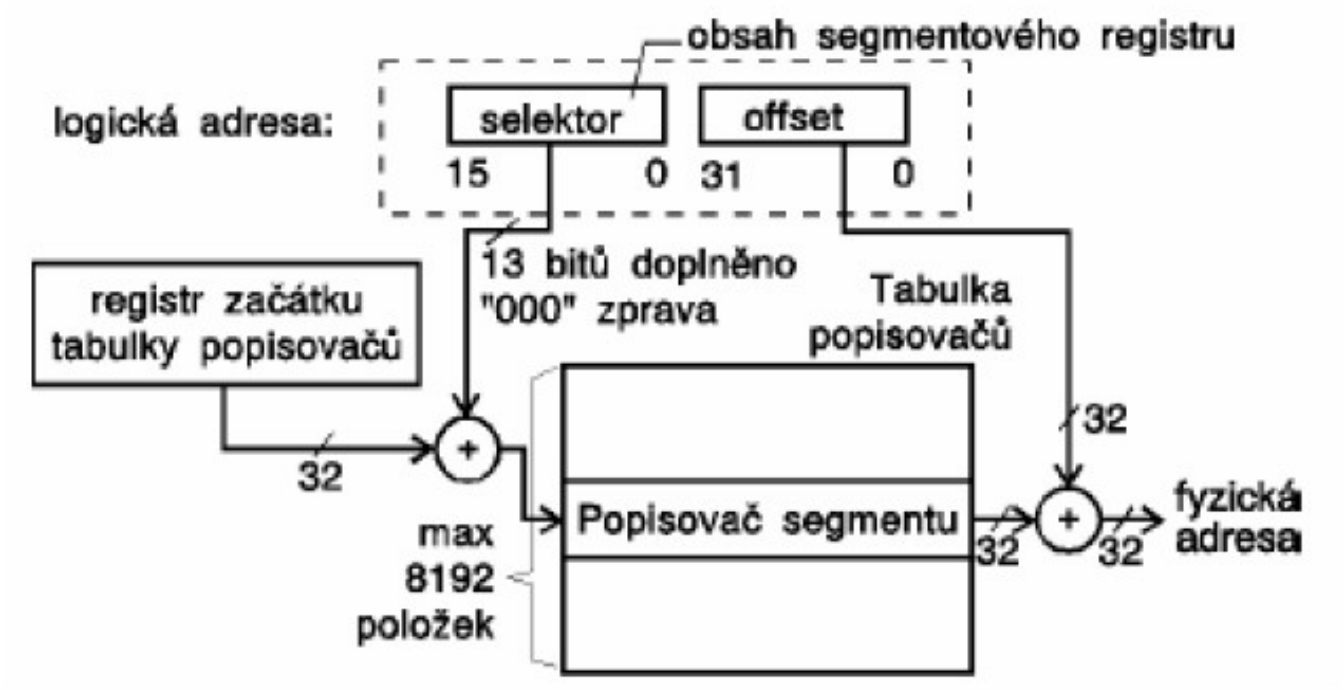
3B – báza segmentu (24-bitová adresa segmentu)⇒

16M adresný priestor

2B – limit segmentu (max. veľkosť 64kB)

Pre I80386

- logická adresa rovnaká, ale
- deskriptor obsahuje okrem iného 32-bitovú bázu a 20- bitový limit



Nevýhoda – tabuľka deskriptorov – maximálna veľkosť tabuľky 64kB $\Rightarrow$ 8192 položiek (deskriptorov) v reálnom móde t.j. 64kB

Výhoda možnosť prideľovania práv na prístup do pamäte – multitasking, ochrana systému pred užívateľom

Dvojúrovňový pamäťový podsystém

Rozšírenie adresného priestoru využitím sekundárnej pamäte
Problém nerovnakého prístupu k údajom
- HP – čítanie s ľubovolným prístupom
- sekundárna pamäť (SP) – čítanie s prístupnením po blokoch

Hľadanie takého riešenia, aby sa programátor nemusel zaoberať technikou, ako sa dostať k údajom

Jedným z riešení je **virtuálne adresovanie**

- jednotný prístup k údajom (inštrukciám)
- rozdelenie pamäťového priestoru na nezávislé podpriestory
    - ochrana údajov
    - pamäťový priestor môže zdieľať viacero procesov (**multitasking)**

Virtuálna adresa

-adresovanie cez tabuľky, užívateľ nemusí vedieť, ako sa to robí

Tabuľka obyčajne obsahuje

- adresa bloku v HP
- prístupové práva
- ďalšie informácie, ako sú:
    - či blok, ktorého položka je adresovaná, je v HP alebo SP
    - či niektoré položky bloku sú zmenené
    - pomocné informácie pre rozhodovanie, či bude blok uvoľnený pre iný blok

Tabuľka
- slúži na „výpočet fyzickej adresy“ – preklad adresy
- nemôže obsahovať všetky možné adresy (bola by väčšia ako pamäťový priestor)

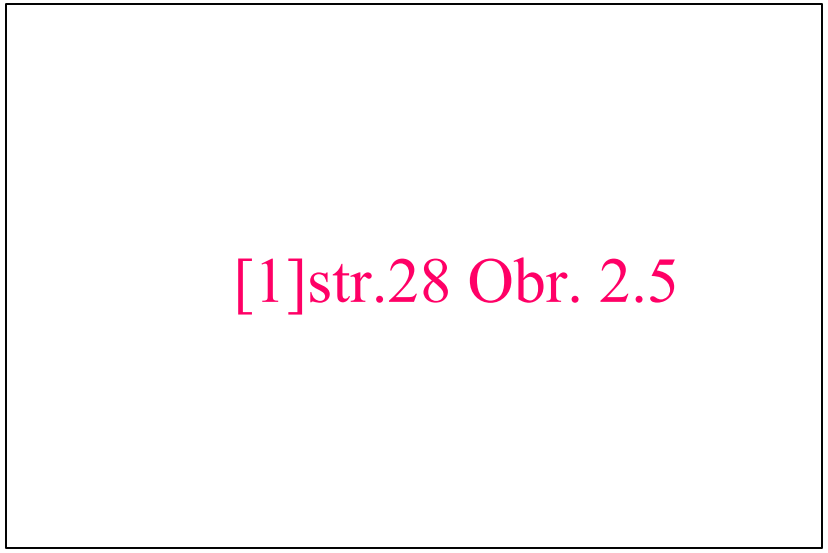
Virtuálna adresa sa delí na **prekladanú časť** a **neprekladanú časť** (offset) $\Rightarrow$ podstatne sa zmenší tabuľka

Z hľadiska efektívnosti (presuny medzi HP a SP) sa vykonávajú po blokoch je voľba stratégie
- organizácia presunu údajov medzi HP a SP
- optimálna veľkosť bloku
- spôsob výberu blok
- stratégia voľby okamihu presunu bloku

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

Stránkovanie
- najčastejšie používaná metóda (**stránkovaná pamäť**), 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 **1kB** až **8kB**)

Preklad adresy (konverzia, dekódovanie) v stránkovom systéme



Vstupom do prekladu adresy
- **VAST** – virtuálna adresa stránky
- **POSUN –** (časť reálnej adresy)
- signály, z ktorých sa dá dekódovať, či ide o RWX

Výstupom je **reálna** adresa pamäťového miesta v HP

VAST je offset do tabuľky stránok, ktorá obsahuje okrem iného
- **P** – či je stránka v HP
- **RWX** – prístupové práva
- **RAST** – hornú časť reálnej adresy začiatku stránky v HP

Problém je v tabuľke stránok
obsahuje reálne adresy v HP všetkých stránok pamäťového priestoru, je **skoro prázdna**, lebo absolútna väčšina stránok nie je v HP (tabuľka musí obsahovať informácie o všetkých stránkach !!!!)

Príklad :

Virtuálna adresa 32-bitová,
stránka 4KB⇒ **posun** 12 bitov ⇒ 20-bitová **VAST**⇒ **tabuľka stránok** musí mať 2^{20} položiek
Pri **2B** položke je to 2MB pamäte pre tabuľku (musí byť celá v HP!!!, uloženie mimo nej by viedlo k **zacykleniu prekladu adresy**), ak by bolo v HP voľných 4 MB pamäte, môžeme mať v pamäti naraz maximálne **1024** rámov⇒potom z 1048576-1024=1047552 by bolo **neobsadených** položiek (cca **99,9**%)

Riešením je systém s **viacúrovňovým prekladom** (**spomalenie** preladu), treba čítať z HP viackrát, kým získame reálnu adresu (urýchlenie použitím špeciálnych pamätí)

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 uvolnený rám umiestni stránku zo SP
- aktualizuje príslušné položky tabuľky stránok

V multitaskovom 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)

Segmentovaný pamäťový systém

Nevýhoda stránkovania je, že vždy do HP sa presúva blok konštantnej veľkosti, ktorý obvykle nebýva obsadený (využitý celý) $\Rightarrow$ plýtvanie s pamäťou

Segment – skupina po sebe nasledujúcich pamäťových miest, ktorej dĺžka sa môže meniť

- výhoda – lepšie využitá HP
- nevýhoda - zložité algoritmy hľadania voľnej pamäte dostatočnej veľkosti

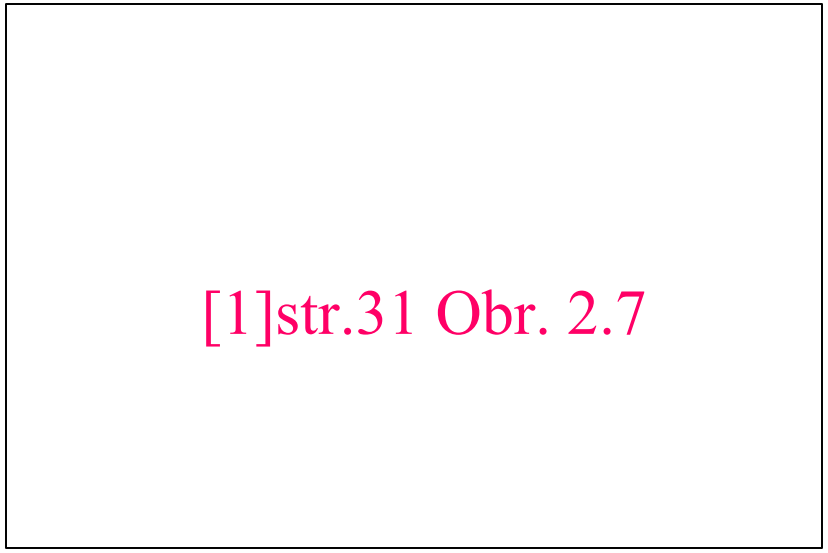
Preklad virtuálnej adresy v segmentovanom pamäťovom systéme

preklad je podobný ako u **VAST**, len v tabuľke je navyše informácia o dĺžke segmentu

Stránkovaná segmentovaná pamäť

Kombinácia predchádzajúcich metód
Filozofia – aplikovať princíp segmentovanej pamäte na tabuľku stránok, čo umožňuje mať v HP len tabuľky stránok **bežiacich úloh !!!**

Preklad virtuálnej adresy v stránkovom segmentovanom pamäťovom systéme



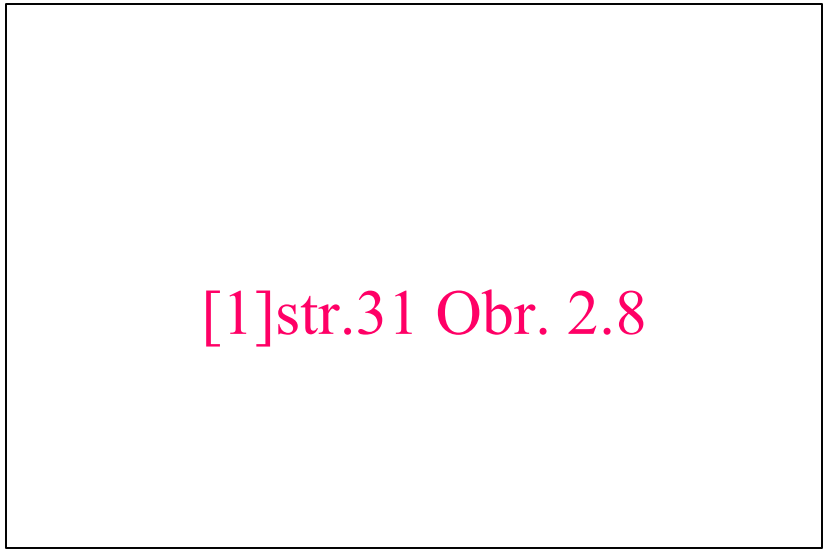
Virtuálna adresa sa skladá z :

VASE – posun v tabuľke segmentov (tabuľka obsahuje adresy začiatkov tabuliek stránok)

VAST – posun v tabuľke stránok

POSUN – posun v stránke (ráme)

Je možné zvoliť aj preklad s viacerými úrovňami napr. viacúrovňový preklad virtuálnej adresy v procesore SPARC



Kontext – (úloha) má k dispozícii 256*64*64 stránok dĺžky 4kB
Súčasne môže bežať **4096** úloh

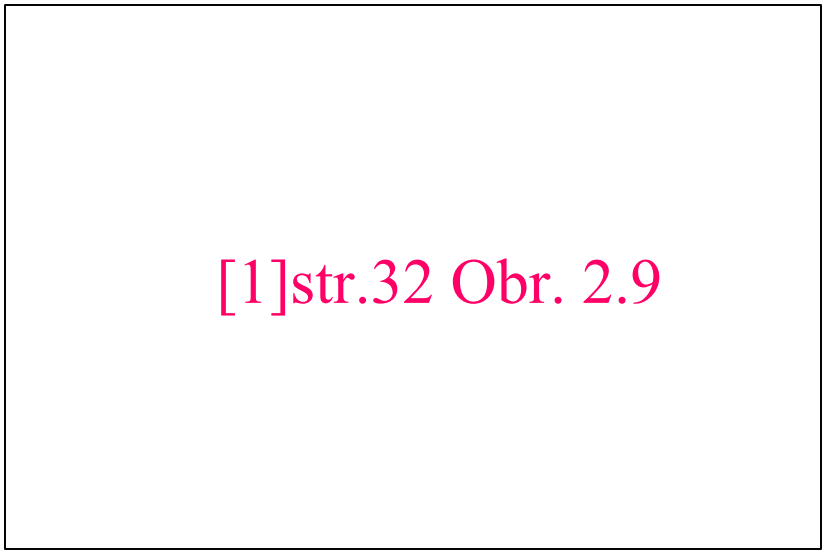
Inverzná tabuľka stránok

Rieši úsporu HP minimalizáciou tabuľky stránok algoritmicky pri zachovaní štatisticky minimálneho počtu čítaní z HP pri preklade adries.

Na minimalizáciu počtu položiek v tabuľke sa používa **pseudonáhodná transformácia** VAST na kratšiu binárnu postupnosť
(redukcia dĺžky VAST adresy $\Rightarrow$menšia pamäť na tabuľku),
Vychádza z filozofie, že treba v tabuľke držať adresy len využívaných stránok

Pseudonáhodná transfromácia

- transformácia dlhšieho binárneho reťazca na kratší
- realizuje sa XOR vybraných bitov slova napr. IBM 370 (obvodová realizácia kombinačným obvodom)
- na rovnaký vstup dáva rovnaký výstup (je deterministická)
- znáhodňuje len postupnosť adries tak, aby susedným (blízkym) hodnotám VAST nepriradila rovnaký kód (adresu do tabuľky)
- môže sa stať (pravdepodobnosť je nízka), že pre dve rôzne VAST vyjde rovnaký kód, preto je pridaná položka do tabuľky ZREŤAZENIE a pôvodná hodnota VAST, pri zhode VAST v tabuľke s VAST na vstupe kódera sa vytvorí zreťazená položka
- zvýšenie času čítania z tabuľky hrozí len pre zreťazené VAST



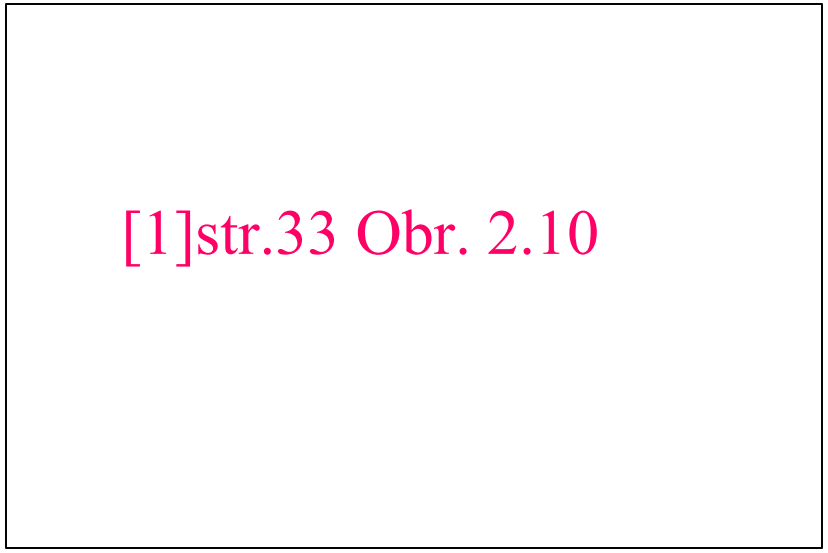
Postup prehľadávania

- prečítanie položky z inverznej tabuľky stránok
- ak VAST sa zhoduje s VAST adresy použije sa RAST
- ak nie číta sa položka definovaná posunom v časti ZREŤAZENIE a postup pokračuje, kým nebude prečítaná položka so zhodným VAST
- priemerné zreťazenie býva 1 až 2

Úplne asociatívny adresár hlavnej pamäte

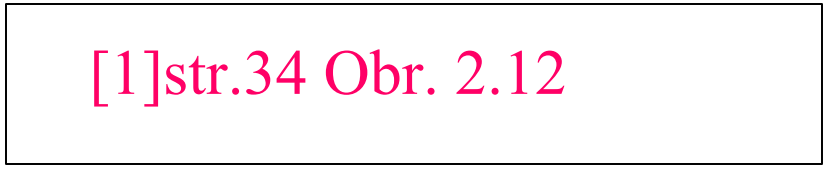
Používa sa na ukladanie najčastejšie používaných stránok
- pamäť pre VAST , klúč do asociatívnej (obsahom adresovateľnej pamäte CAM)
- pamäť pre RAST – reálne adresy začiatkov rámov v HP)
- nezáleží na poradí uložených VAST
- pri čítaní sa naraz porovnajú všetky VAST v CAM s VAST na vstupe, ak je v tabuľke výstupom je RAST
- pamäte nebývajú veľké, plnia v podstate úlohu vyrovnávacej (cache) pamäte pre MMU
- ak sa nenájde položka v CAM, prebieha preklad adresy štandartným spôsobom, prípadne aktualizácia CAM

Plne asociatívna pamäť pre preklad adries
(TLB – Translation Lookaside Buffer)



Štruktúra plne asociatívnej pamäte (časti jedného riadku)

Slovo plne asociatívneho adresára HP



-v položke zmena sa zapisuje informácia, či do danej stránky bol vykonaný zápis (stránka sa musí pri výmene zapísať do SP)
- LRU je informácia o využívaní stránky, slúži MMU pre rozhodovanie, ktorú stránku vyradí z HP

Stratégia presunu stránok

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

Presun stránky ak sa do nej nepísalo, prepíše sa novou inak sa musí obsah stránky najprv uložiť do SP

Najčastejšie stratégie vyraďovania stránok
- LRU
- FIFO
- náhodný výber
- LFU

LRU (Least RecentlyUsed – **najdlhšie nepoužívaný**)

Vyraďuje najdlhšie nepoužívanú (nevolanú) stránku z HP – zložitá realizácia – vyžaduje veľa zápisov pri každom použití stránky, preto sa používajú zjednodušené varianty stratégie LRU
Napr. inkrementácia počítadla pre danú stránku v tabuľke, a periodická dekrementácia všetkých počítadiel hodinami

FIFO (First In First Out) – v podstate **fronta** – vyraďuje sa stránka, ktorá je v HP **najdlhšie**, realizácia jedným čítačom, ktorého dĺžka (počet bitov) je rovná binárnemu vyjadreniu počtu rámov v HP), alebo index vyraďovaného rámu sa rovná zvyšku po delení obsahu počítadla vyraďovania počtom rámov v HP)

Náhodný výber – vygeneruje sa číslo generátorom **pseudonáhodných** čísel, ktorého zvyšok po delení počtom rámov v HP je indexom vyraďovaného rámu

LFU (Least Frequently Used) – vyradí sa **najmenej volaná stránka**

Problém vyraďovania pri multitaskingu

-najjednoduchšie riešenie, každá úloha dostane svoj úsek pamäte, pričom správa vyraďovania je pre každú úlohu autonómna, nie je ekonomické,

- optimálne je aby jedna úloha mohla vyraďovať stránky inej úlohy, ktoré sú podľa nejakého kritéria najmenej využívané

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), iný 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 trojúrovňvom systéme teoreticky môže byť zapísaná tá istá informácia 3-krát

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)

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

Zásadný problém – ako realizovať predvýber blokov do VP, aby **CPU čítal len z VP** – neexistuje uspokojivá stratégia
-prvý presun až pri **požiadavke čítania** (on demand)
- ak je vyžiadané slovo, ktoré nie je vo VP, najprv sa číta žiadané slovo do procesora, až potom sa číta zbytok bloku do VP

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**“)

Ideálna cache obsahuje toľko položiek, že sa do nej zmestí celý 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.

Vyhľadávanie použitím princípu asociativity
-úplne asociatívne – nepoužiteľné, lebo by bol vo VP veľmi malý počet položiek
- používajú sa princípy s obmedzeným stupňom asociativity

Adresár VP s obmedzeným stupňom asociativity

-najčastejší spôsob realizácie VP

Stupeň asociativity udáva **počet slov adresára**, ktoré treba prehľadať pri hľadaní bloku vo VP.

Zvláštna stratégia zápisu blokov do VP
-pre každý **blok** je vyhradená určitá skupina **rámov** VP (**sada rámov**)
- do určitej sady rámov VP môžeme zapisovať len **určitú podmnožinu blokov** z HP (podmnožinu nazývame **triedou**)

Trieda obsahuje oveľa viacej blokov ako sada rámov vo VP.

Bloky presúvané do VP **súperiť** o miesto v sade VP.

Zmenšenie konkurencie – bloky nasledujúce v HP patria do iných tried (miestna lokalita).

Príklad VP so stupňom asociativity 2

Kapacita HP – **1MB**, strojové slovo **4B** ⇒ celá pamäť **256k slov** ⇒na adresovanie stačí **18 bitov** (namiesto 20) **A19 až A2**.
Dĺžka bloku **16 slov** (**64 B**), počet blokov **16k**.
VP **8kB** rozdelená do **128** rámov po **16** slov, každá **sada** má **2** rámy ⇒ **64 sád** ⇒HP musí byť rozdelená do **64 tried** (kódovanie triedy priamo **A11 až A6** adresy).
A5-A2 je offset v bloku.

Vyhľadanie obsahu podľa adresy vo VP
A11 až A6 priamo adresuje zoznam častí adries A19 až A12, na jednej adrese sú 2 položky, tie sa porovnajú (**komparátorom**) s požadovanou časťou adresy A19 až A12, ak je jedna z nich v adresári aktivuje sa čítanie z bloku údajov VP
Porovnanie A19 až A12 vyhodnotí MPX (**multiplexor**), pri zhode aktivuje výstup z pamäte dát alebo generuje **výpadok VP** (údaje z danej adresy nie sú vo VP)

Výhody

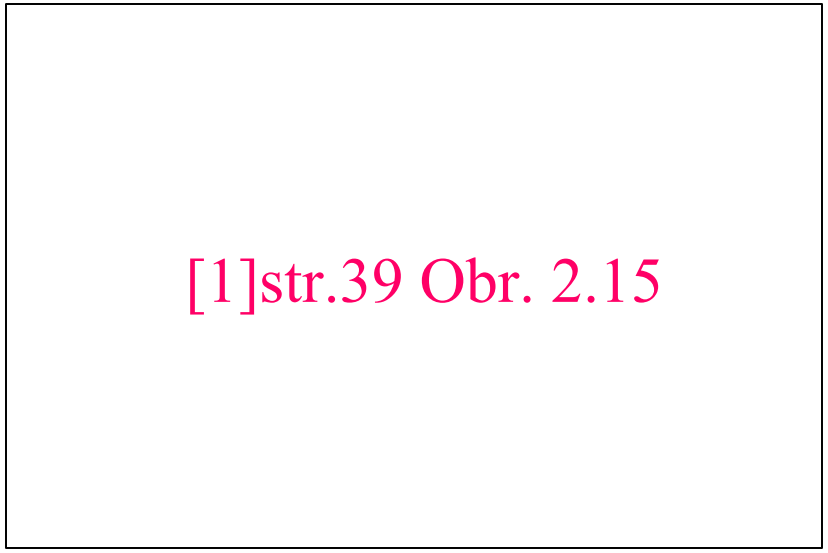
-obvodová podpora sa dá realizovať mimo pamäť,

- dá sa realizovať štandardnými prostriedkami

- umožňuje súčasné čítanie s adresára aj z pamäte dát (rýchlejšie ako pri asociatívnej pamäti)

- stupeň asociativity môže byť až **16**

Adresár VP s priamym zobrazením



Špeciálny prípad predchádzajúceho riešenia (direct mapping)
stupeň asociativity=1

Každá trieda má len jeden rám vo VP
-výhoda – jednoduchšie obvodové riešenie (netreba porovnávať adresy)
- veľkosť VP zostáva rovnaká, lebo pri rovnakej veľkosti blokov treba 128 rámov
- nevýhoda pri nezhode adries sa vždy generuje výpadok VP, väčšia pravdepodobnosť výpadku⇒pomalší systém

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 uvolňovania blokov** a **stupňa asociativity** (pomer býva blízky **0,05**)

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

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:

2. **Hlavička, J.:Architektura počítačú. ČVUT 1998**