Virtuálna pamäť

- Stránkovanie
  - Stránkové tabuľky
  - Asociatívna pamäť TLB
  - Algoritmy nahradzovania stránok
- Segmentovanie

Autor: Peter Tomcsányi

Použité obrázky z učebníc:
Andrew. S. Tanenbaum, Modern Operating Systems, 2nd edition
Andrew. S. Tanenbaum, Structured Computer Organization
http://www.cs.vu.nl/~ast/books/

Virtuálna pamäť

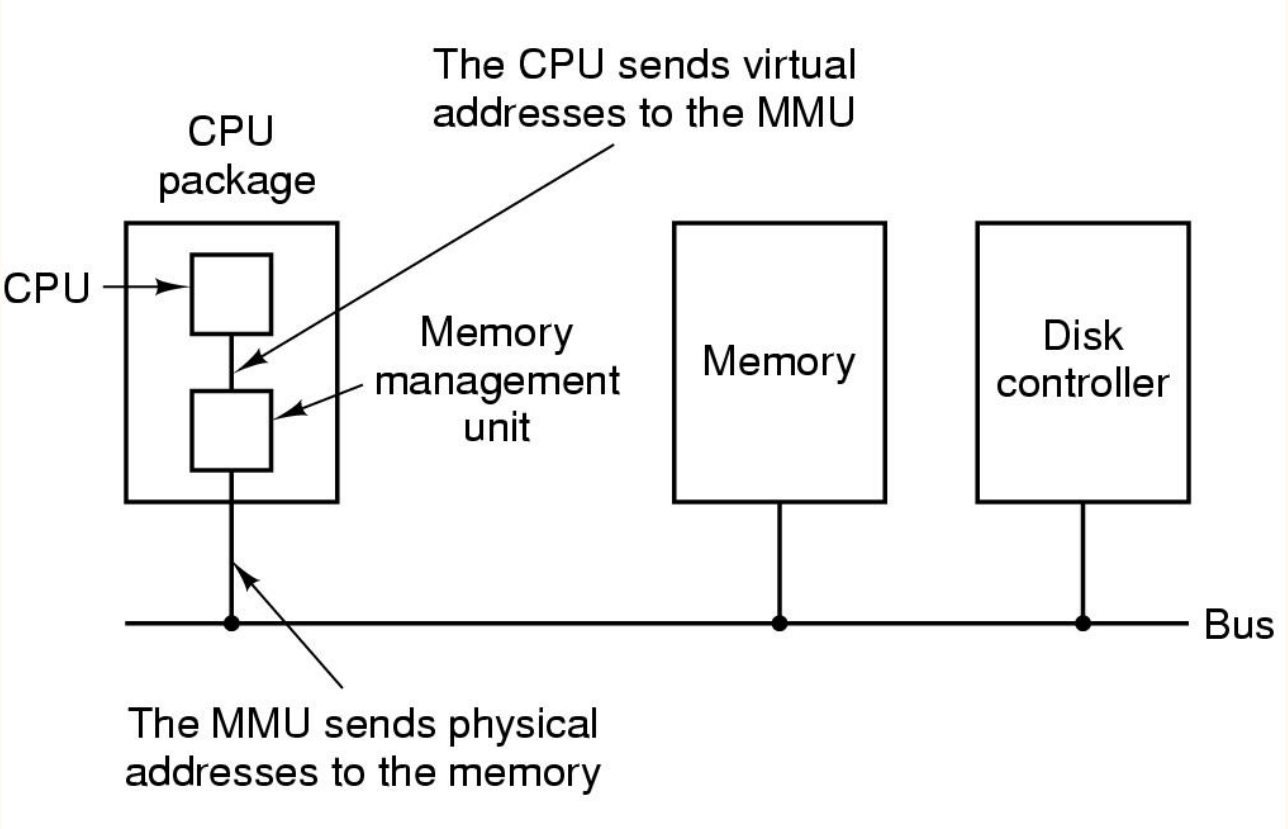
je taká správa pamäte, pri ktorej sa každému procesu dáva k dispozícii vlastná pamäť, ktorá má inú veľkosť alebo iný spôsob adresovania ako je fyzická pamäť

- Historicky vznikla hlavne z potreby vykonávať programy, ktoré sú väčšie než fyzická pamäť počítača

Stránkovanie

- Virtuálna pamäť je rozdelená na rovnako veľké úseky - **stránky (pages)**
- Fyzická pamäť je rozdelená na **rámce (page frames)** rovnakej veľkosti ako stránky
- Typická veľkosť stránky je 4KB
- Niektoré stránky virtuálnej pamäti sú vo fyzickej pamäti, iné sú odložené na disku
- Na prepočet adries sa používa **stránková tabuľka**
- Prepočet adries musí podporovať strojový kód daného procesora, inak by bola implementácia neefektívna

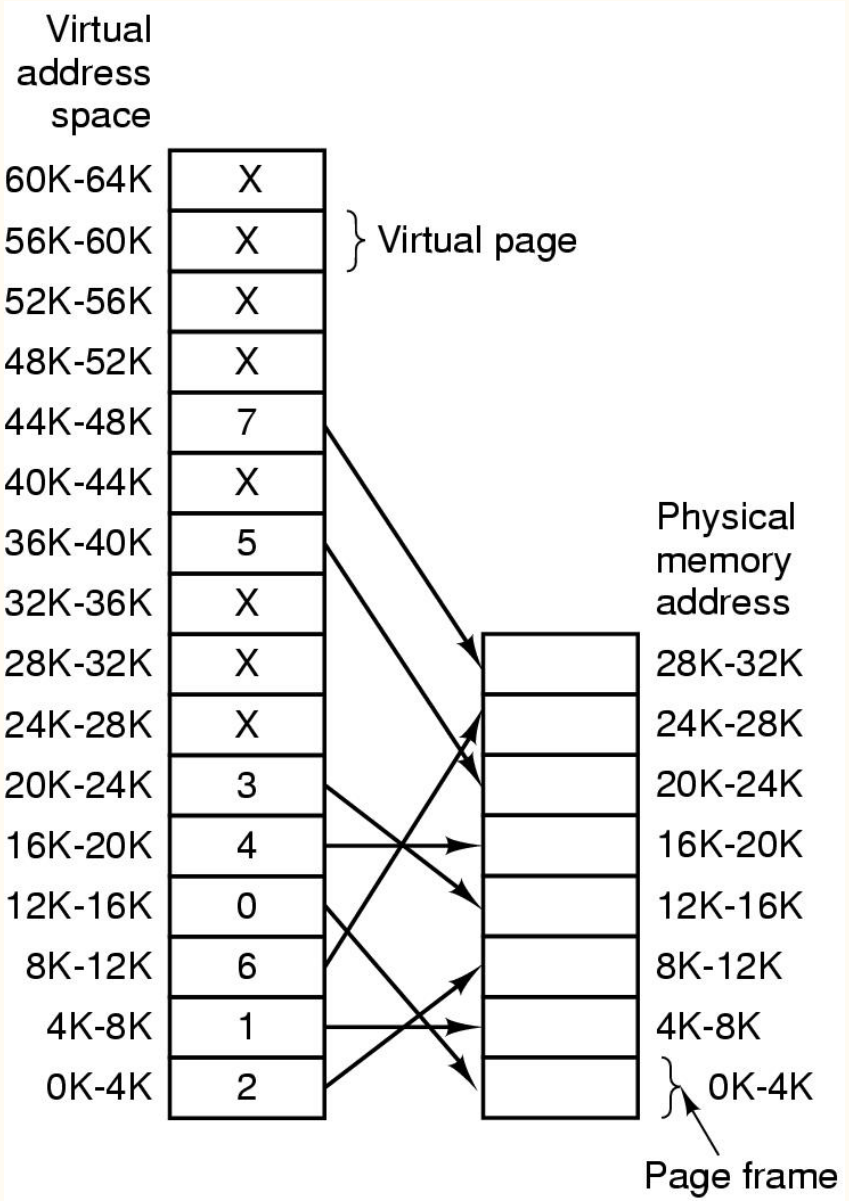
Stránkovanie (2)



- Memory Management Unit prepočítava virtuálne adresy na fyzické
- Musí byť súčasťou architektúry procesora, inak by nebolo možné efektívne implementovať stránkovanie

Stránkovanie (3)

- Príklad stavu virtuálnej pamäti
- Vzťah medzi virtuálnymi a fyzickými adresami definuje stránková tabuľka

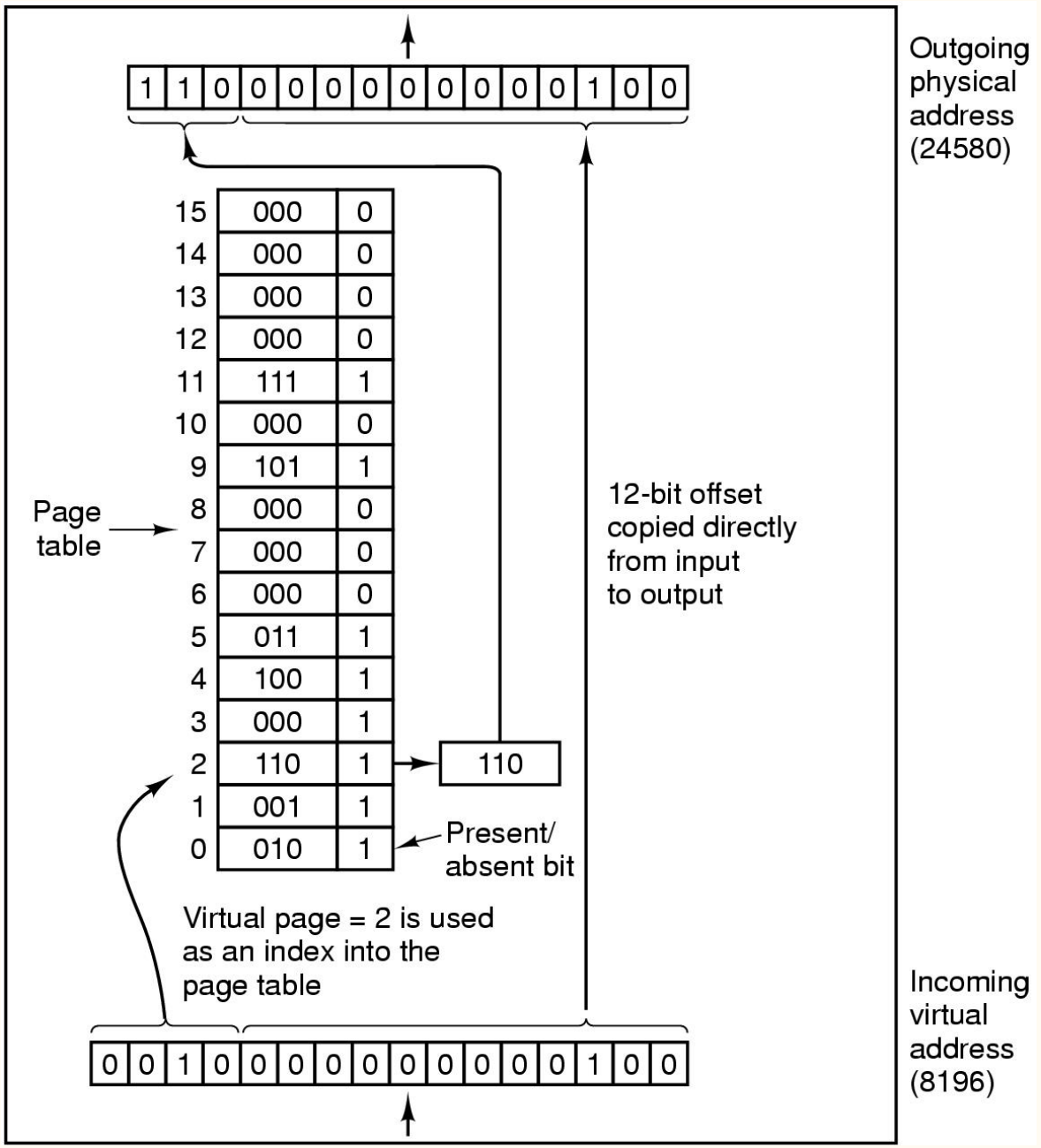


Stránkové tabuľky

Príklad stránkovej tabuľky priamo v procesore

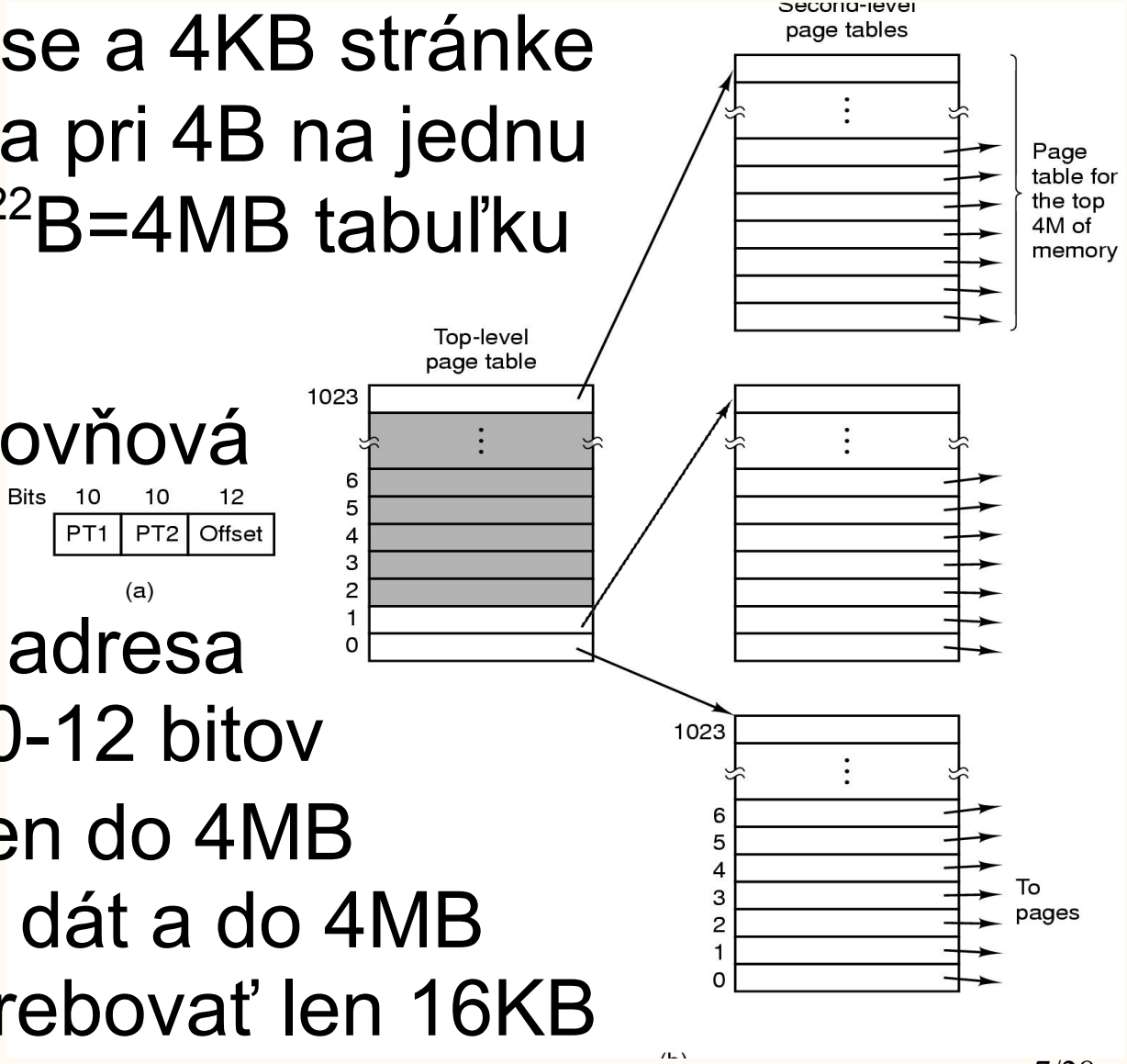
Použiteľné len pre malý počet stránok

Pri väčšom počte musí byť tabuľka v pamäti

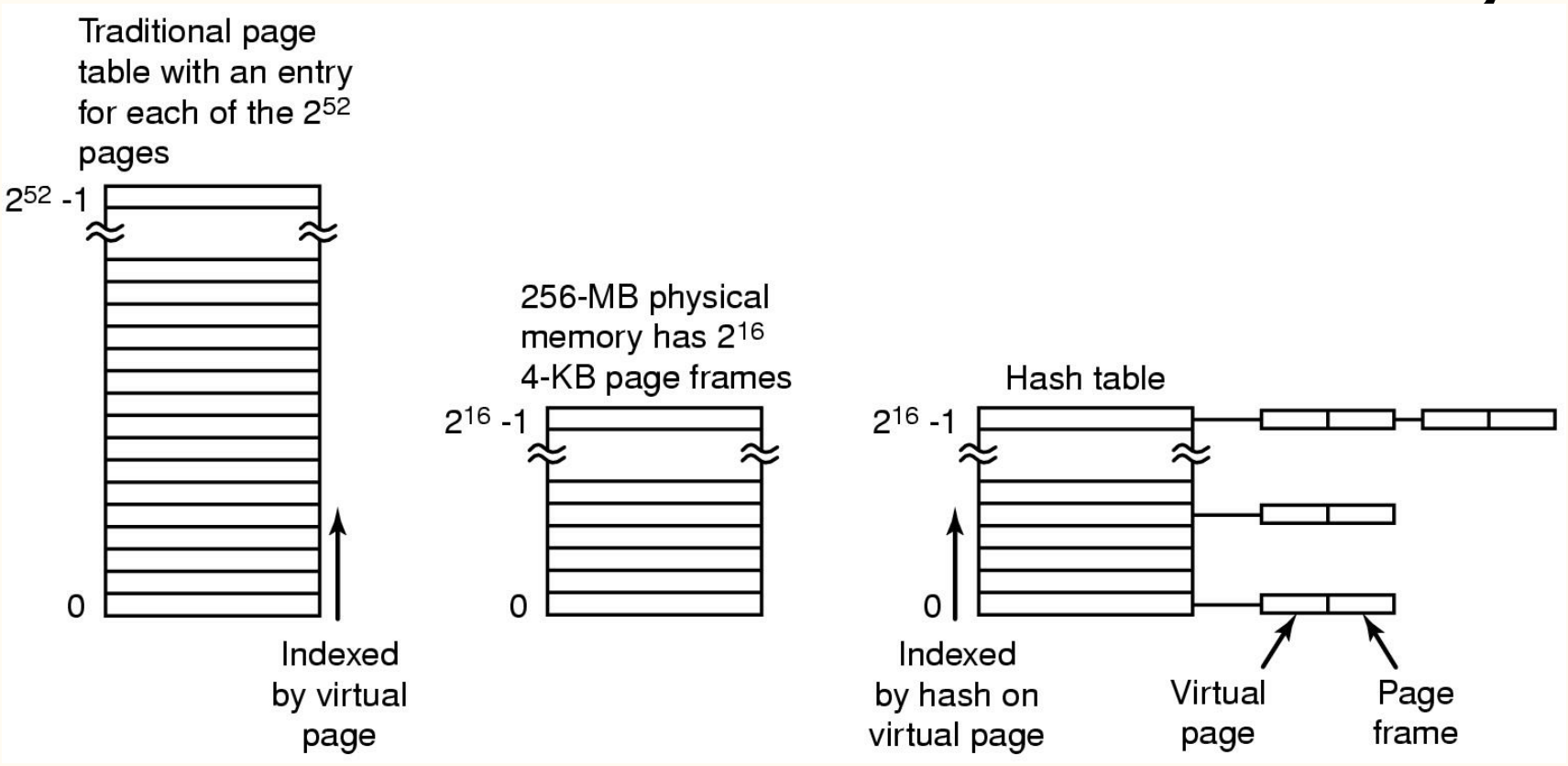


Stránkové tabuľky (2)

- Pri 32-bitovej adrese a 4KB stránke máme 2^{20} stránok a pri 4B na jednu položku je treba 2^{22}B=4MB tabuľku pre každý proces
- Riešením je dvojúrovňová stránková tabuľka
- 32 bitová virtuálna adresa rozdelená na 10-10-12 bitov
- Proces, ktorý má len do 4MB programu, do 4MB dát a do 4MB zásobník bude potrebovať len 16KB tabuliek.

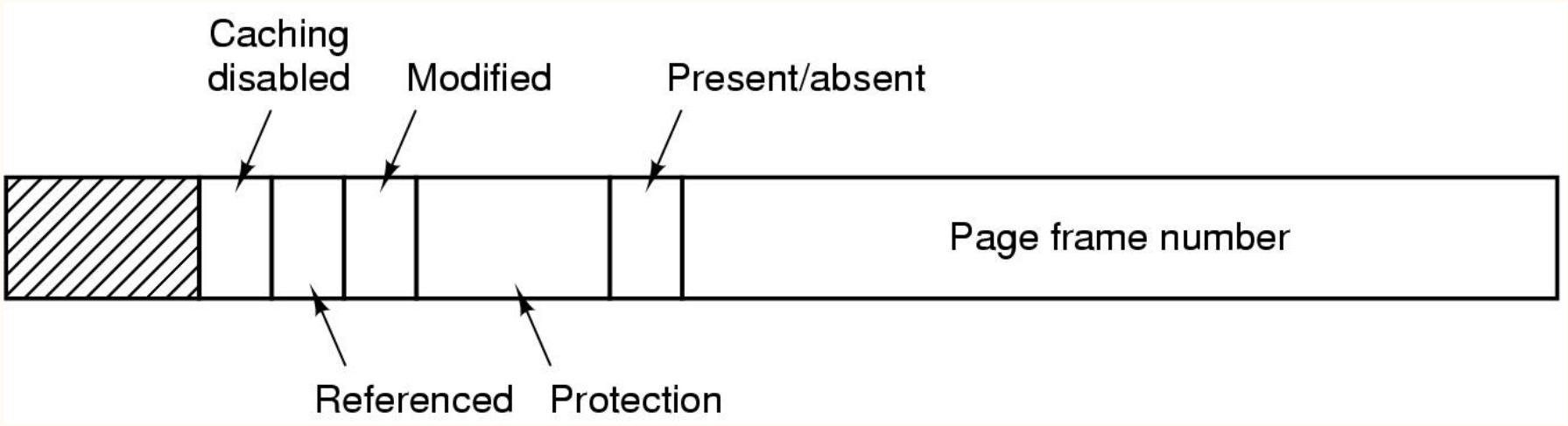


Stránkové tabuľky (3)
Invertované stránkové tabuľky



- Pre 64-bitovú adresu a 4KB stránku máme 2^{52} stránok a pri 8B na položku potrebujeme 2^{55}-bajtovú tabuľku (32PB)
- Jedna z možností, ako to riešiť je invertovaná tabuľka: Každý rámec má informáciu o uloženej stránke
- TLB a pomocná hašovaná tabuľka zrýchľuje prepočet adries na rozumnú mieru

Stránkové tabuľky (4)
Typická položka stránkovej tabuľky



- Okrem čísla rámca a bitu indikujúceho prítomnosť obsahuje aj ďalšie informácie:
  - možnosť vypnúť cache pre danú stránku
  - príznak, či bola stránka použitá
  - príznak, či bol obsah stránky zmenený
  - bity ochrany

TLB

- Translation Look-Aside Buffer
- Je to špeciálna asociatívna pamäť, ktorá uchováva niekoľko posledne použitých riadkov stránkových tabuliek
- Podstatne sa tým zrýchľuje prepočet adries lebo sa minimalizuje počet prístupov do pamäti

TLB (2)

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

- Príklad obsahu TLB

Prepočet adries - spolupráca procesora a operačného systému

- Ak je stránka v pamäti, tak MMU prepočíta virtuálnu adresu na fyzickú podľa stránkovej tabuľky
- Keď stránka nie je v pamäti (je to poznačené v stránkovej tabuľke), tak MMU spôsobí výnimku typu výpadok stránky
- Výnimku musí obslúžiť operačný systém
- Ak je ešte voľný rámec, tak do neho prečíta z disku požadovanú stránku a upraví stránkovú tabuľku
- Ak nie je voľný rámec, tak musí niektorú stránku v pamäti obetovať, ak bola modifikovaná, tak ju zapíše na disk a do uvoľneného rámca prečíta z disku požadovanú stránku a upraví stránkovú tabuľku
- Po návrate z výnimky procesor znova vykoná inštrukciu, ktorá spôsobila výnimku, teda prepočíta tú istú adresu podľa upravenej stránkovej tabuľky

Algoritmy výmeny stránok

- Určia, ktorú stránku obetovať
- Cieľ: Minimalizovať počet výmien stránok
- Optimálny algoritmus je známy ale nedá sa implementovať

Optimálny algoritmus

- Vyber tú stránku, ktorá sa v budúcnosti najneskôr použije
- Nedá sa implementovať, lebo treba predpovedať budúcnosť
- Má význam ako dolný odhad pri teoretických úvahách a pokusoch
- Reálne algoritmy sa snažia používať minulosť na odhad budúcnosti

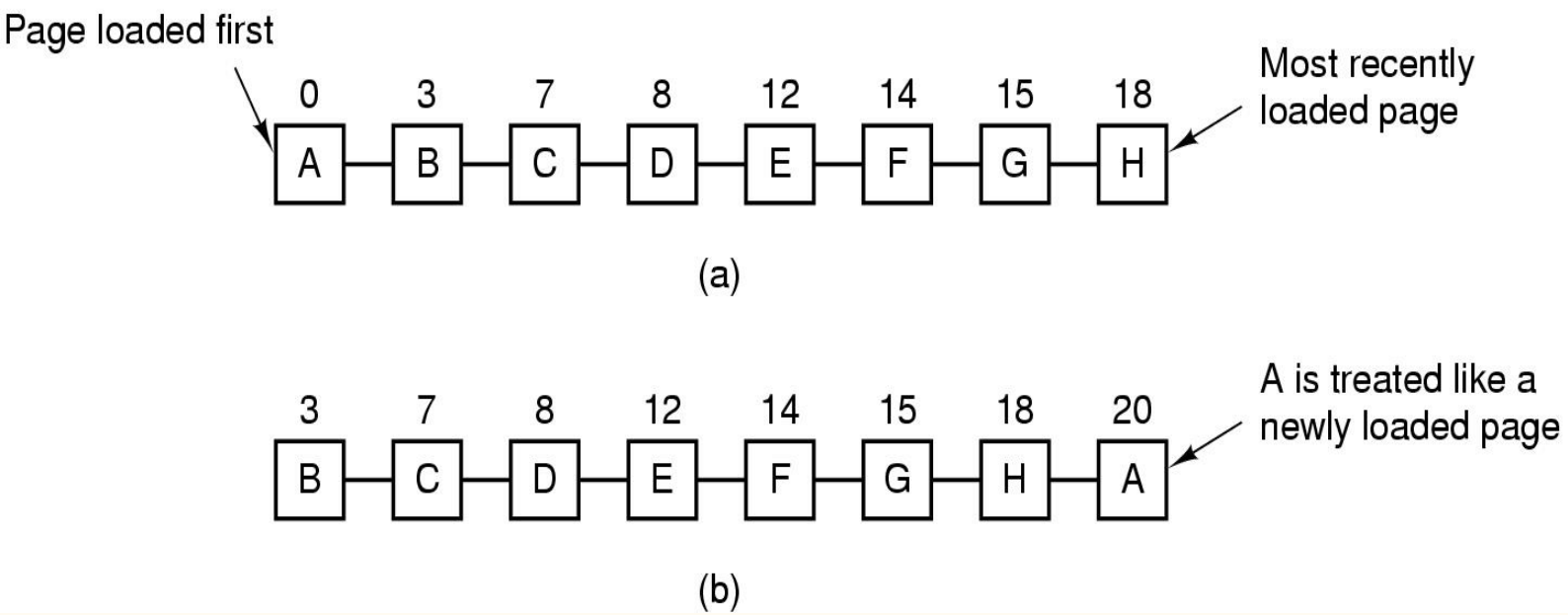
NRU - Not Recently Used
Nepoužitá v blízkej minulosti

- Stránka má bit R (referenced, použitá) a M (modified, zmenená). Oba bity nastavuje MMU na hodnotu 1 keď sa do stránky adresuje (R) resp. zapisuje (M)
- V istých časových intervaloch OS nuluje R, teda R je 1 ak sa stránka použila počas posledného intervalu.
- Bit M sa nuluje pri čítaní stránky do pamäti. Bit M indikuje, či sa stránka zmenila oproti obrazu na disku
- Na základe bitov R a M delíme stránky na 4 skupiny:
  1. nepoužitá a nezmenená
  2. nepoužitá a zmenená
  3. použitá a nezmenená
  4. použitá a zmenená
- Algoritmus NRU: Obeťou je stránka náhodne vybraná z neprázdnej skupiny s najnižším číslom

FIFO
Prvá dnu, prvá von

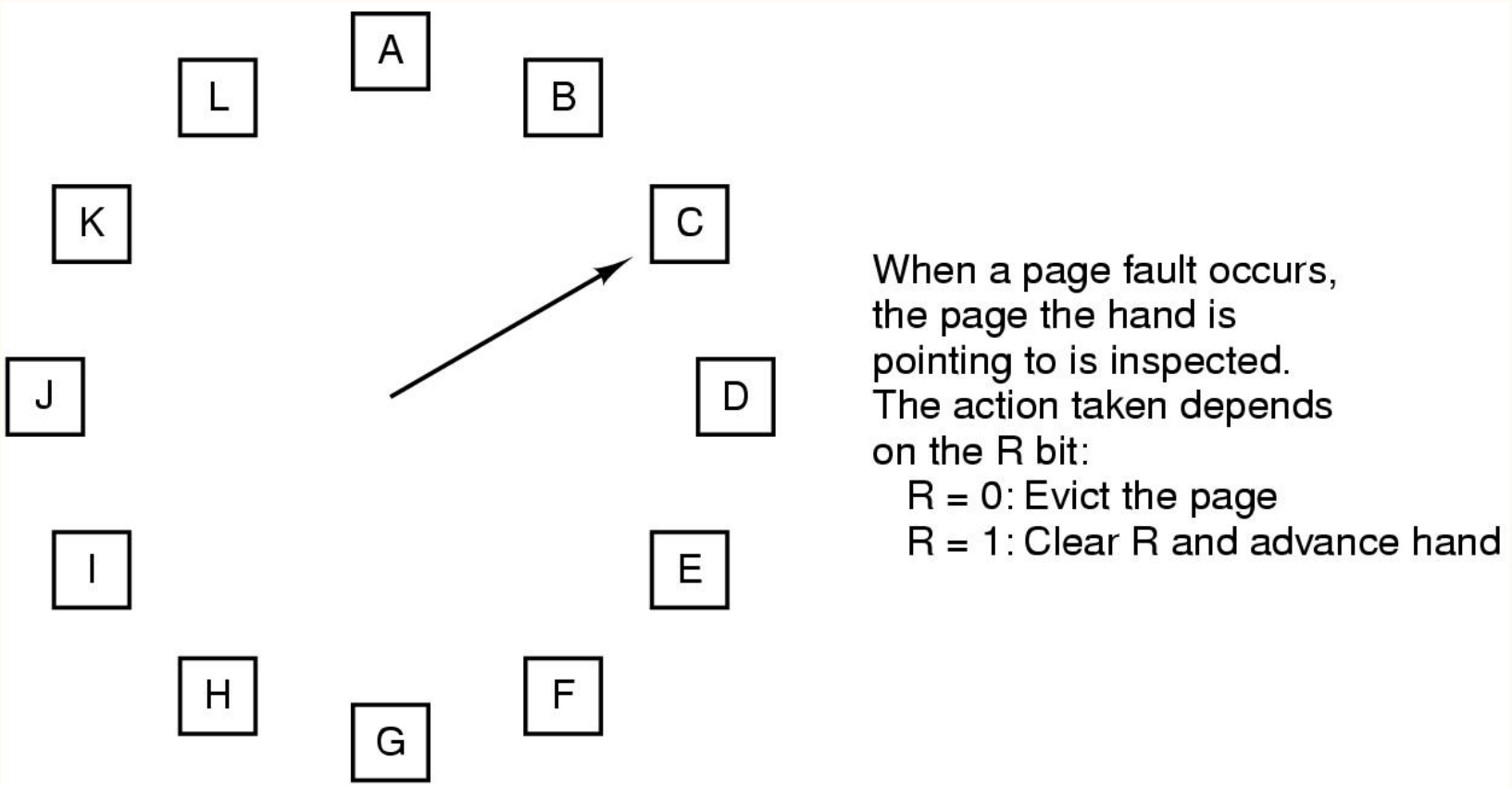
- Udržiava sa spájaný zoznam stránok v poradí, v akom sa prečítali do pamäte
- Vyber stránku, ktorá prišla do pamäte najdávnejšie, teda je už najdlhšie v pamäti
- Nevýhoda: Stránka, ktorá je najdlhšie v pamäti môže byť stále silne používaná

Druhá šanca



- Ako FIFO, ale keď má najstaršia stránka nastavený bit R, tak sa prehodí na koniec zoznamu a R sa jej vynuluje a hľadá sa ďalej
- Nájde stránku, ktorá je najdlhšie v pamäti a nebola použitá v poslednom časovom intervale
- Ak taká nie je, tak sa dostane späť na prvú stránku (už má vynulovaný R) a vyberie tú

Hodiny



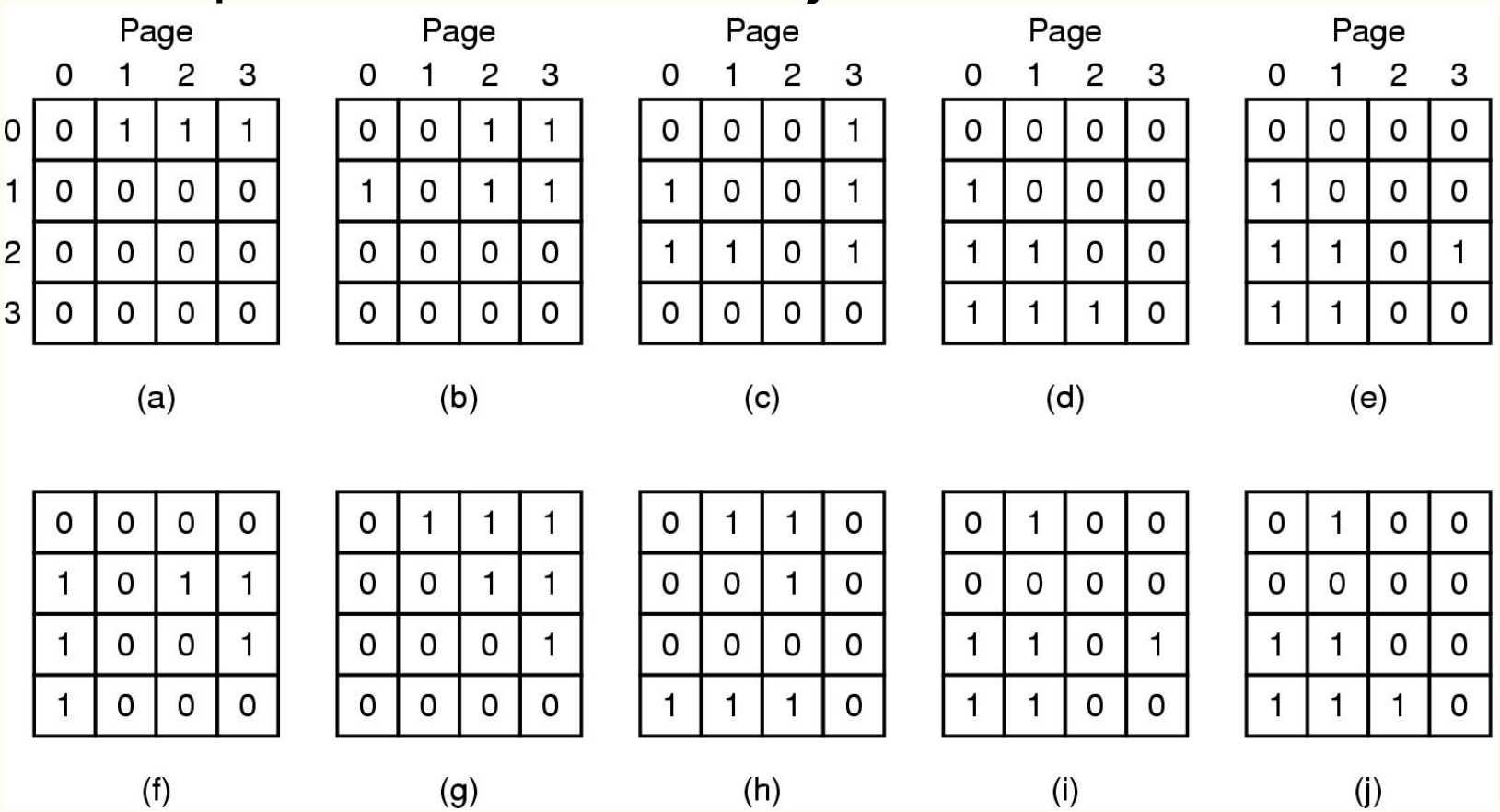
- Iná implementácia Druhej šance
- Používa kruhový zoznam namiesto lineárneho

LRU - Least Recently Used
Najdávnejšie použitá

- Predpokladáme, že nedávno použité stránky bude treba aj v blízkej budúcnosti
- Za obeť vyberme tú stránku, ktorá bola posledne použitá najdávnejšie, teda bola najdlhšie nepoužitá
- Možné implementácie:
  - Udržiavať zoznam stránok, v poradí, v akom sa použili a pri každom použití stránky (teda adresovaní do nej) presunúť stránku na koniec zoznamu. Obeťou je prvá stránka v zozname.
  - V každom riadku tabuľky uchovávať čas posledného použitia a pri každom použití stránky tam zapísať momentálny čas. Obeťou je stránka s najnižšou hodnotou času.
- Obe implementácie sú časovo náročné a vyžadujú spoluprácu od architektúry strojového kódu, ktorá v bežných procesoroch nebýva implementovaná

Softvérové simulovanie LRU

- Pre N stránok pole NxN bitov, riadok je N-bitové číslo
- Pri použití i-tej stránky sa vyjedničkuje i-ty riadok a vynuluje i-ty stĺpec
- Obeťou je stránka, ktorej riadok má najnižšiu hodnotu
- Príklad: poradie adresovania je 0,1,2,3,2,1,0,3,2,3

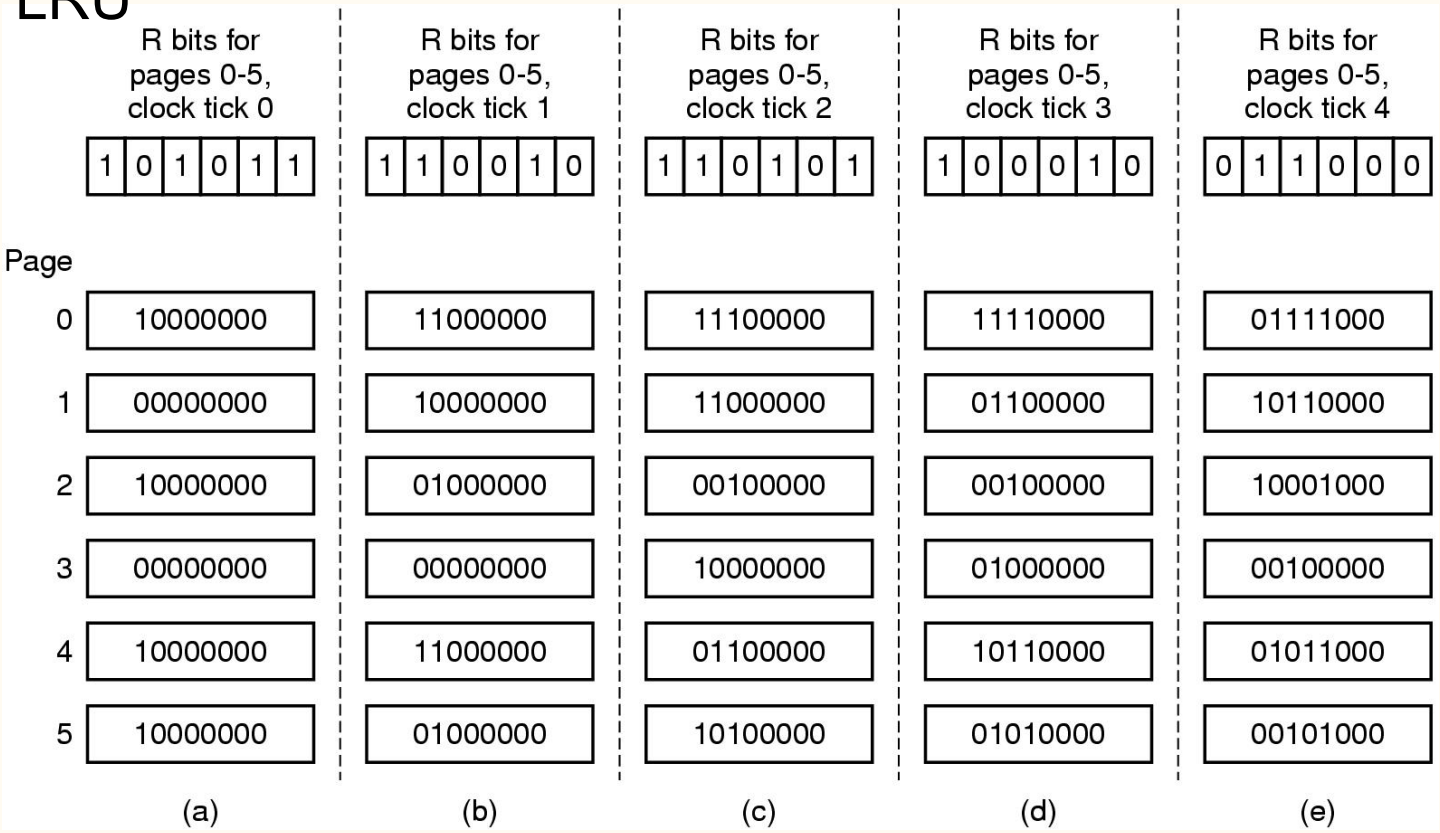


NFU - Not frequently used
Málo používaná

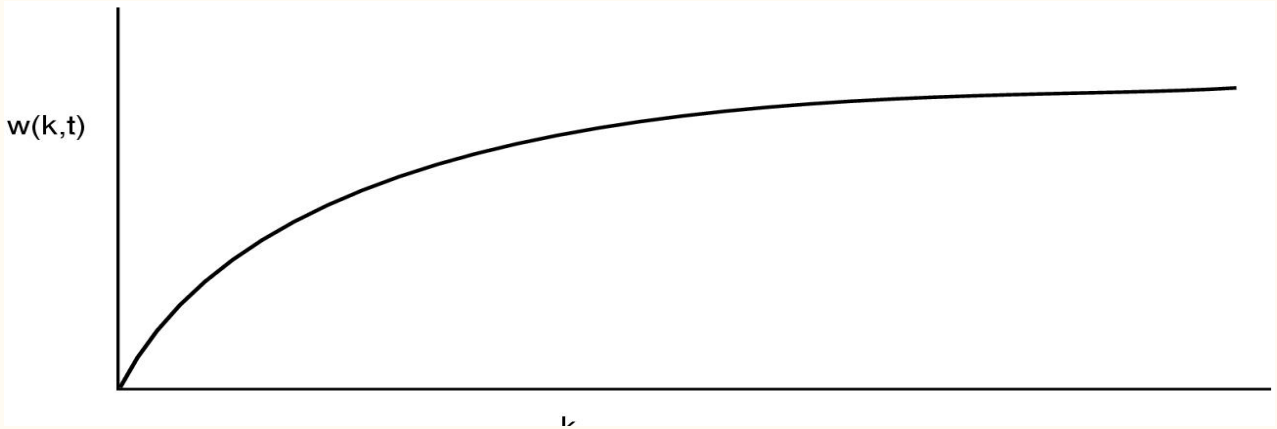
- Snaha o jednoduché priblíženie sa k LRU
- Bity R sa nulujú ako pri NRU
- Každá stránka má počítadlo použití a k nemu sa pred nulovaním pripočíta hodnota bitu R
- číslo v počítadle hovorí, v koľkých časových intervaloch sa stránka použila
- Obeťou je stránka s najnižším počítadlom
- Problémom je, že sa nič nezabúda, aj veľmi dávne použitia stránky majú stále váhu, preto to nie je dobré priblíženie k LRU

Aging - Starnutie

- Vylepšenie NFU
- Počítadlo sa pred pripočítaním posunie o bit doprava a bit R sa “pripočíta” do najvyššieho bitu počítadla
- Tým sa staršie použitia stránok stávajú menej významné, postupne sa “zabúdajú”
- Obeťou je stránka s najnižším počítadlom
- Je to dobré a lacné priblíženie k LRU, nie je to však úplne rovnaké ako LRU

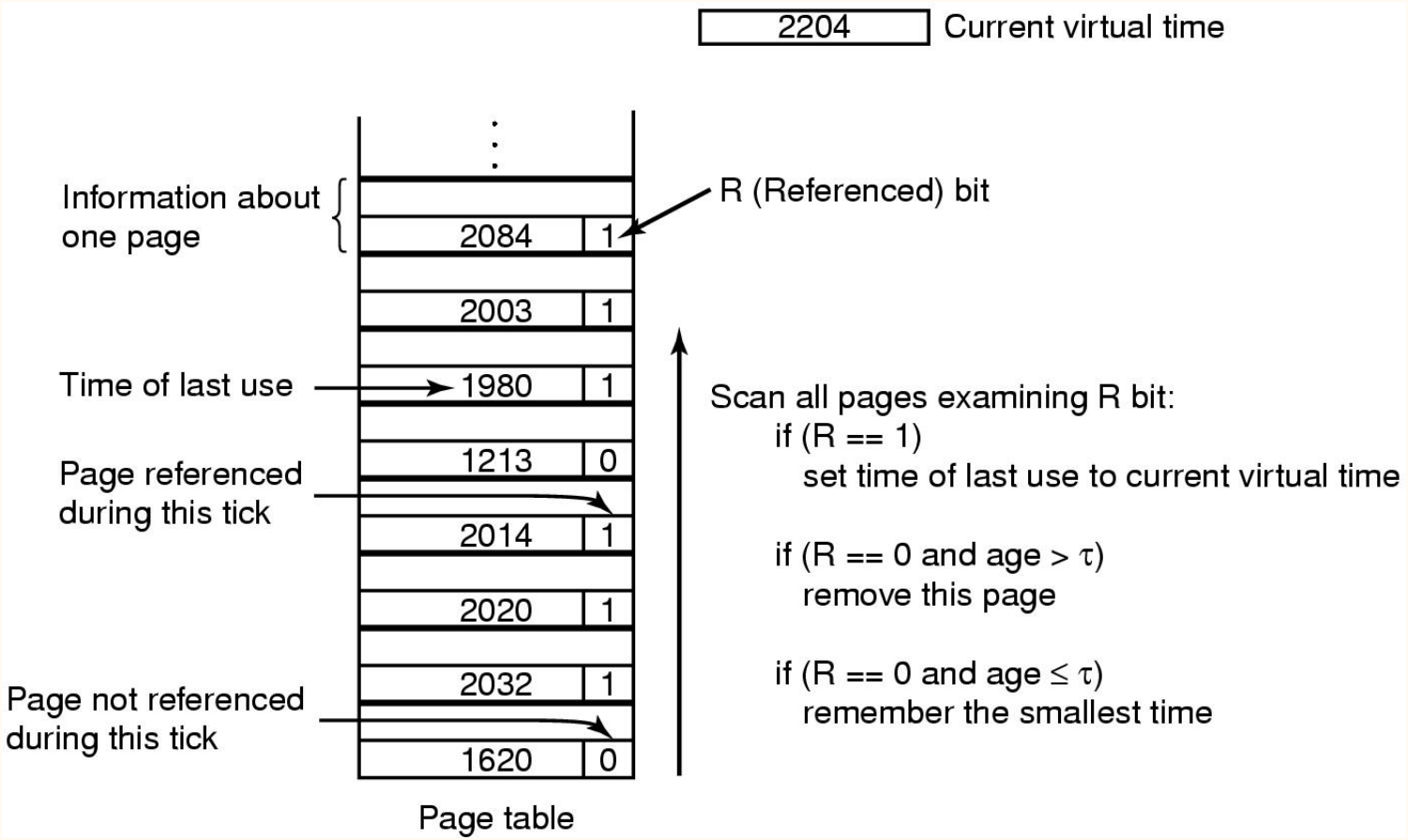


Pracovná množina



- Pracovná množina v čase t je množina stránok použitá v k posledných adresovaniach
- w(k,t) je veľkosť pracovnej množiny
- Graf ukazuje zmenu w(k,t) v závislosti od k v pevnom čase t - od istej hodnoty k je už zmena malá - pracovná množina sa mení “pomaly”
- Keď je celá pracovná množina v pamäti, tak má proces málo výpadkov stránok, keď je ale menej pamäti než je pracovná množina, tak proces neustále generuje výpadky stránok - thrashing.
- Preto je dobré pri výmenách procesov dostať celú pracovnú množinu do pamäte - prepaging
- Idea algoritmu: Obeťou je stránka nepatriaca do pracovnej množiny
- Ako ale zistiť pracovnú množinu?

Pracovná množina (2)



- možná implementácia myšlienky pracovnej množiny do algoritmu
- k posledných adresovaní nahradíme adresovaniami v čase τ dozadu (čas sa ráta pre každý proces zvlášť)
- Hardware a software udržiava bit R ako pri NRU, ale pri každom nulovaní bitu R sa pri R=1 zaznačí čas posledného použitia
- Pri každom výpadku stránky sa pre stránky s R=1 upravia časy posledného použitia
- Obeťou je stránka, ktorej posledné použitie je staršie než τ, ak taká nie je, tak je to najstaršia nepoužitá stránka. Ak neexistuje nepoužitá stránka, tak sa vyberie náhodná obeť

WSClock

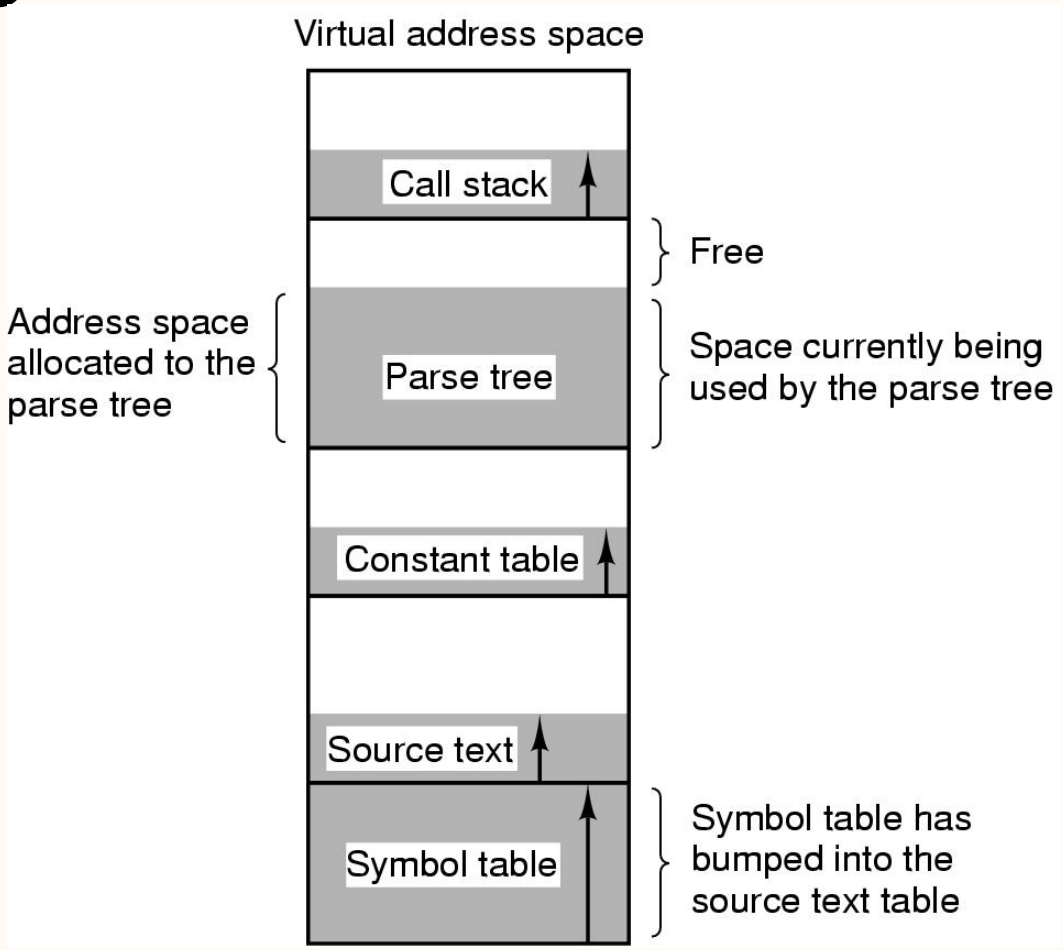
iná implementácia myšlienky pracovnej množiny

- Hardware a software udržiava bity R a M ako pri NRU, ale pri každom nulovaní bitu R sa pri R=1 zaznačí momentálny čas daného procesu, teda dostaneme dobrý odhad času posledného použitia.
- Stránky v pamäti sú usporiadané do kruhového zoznamu.
- V ďalšom budeme nazývať stránky, ktoré nie sú v pracovnej množine svojho procesu, teda boli posledne použité dávnejšie než τ, **staré stránky**. Stránky, ktoré nie sú staré nazveme **nové stránky**. Zároveň o každej stránke vieme či je **zmenená** alebo **nezmenená** podľa jej bitu M.
- Pri výpadku stránky hľadáme starú nezmenenú stránku. Keď nájdeme prvú takú stránku, tak je to obeť a prestaneme ďalej prehliadať.
- Vždy keď narazíme na starú ale zmenenú, tak iniciujeme jej zápis na disk. Pokračujeme v hľadaní starej nezmenenej stránky. Pritom predpokladáme, že časom sa stránka zapíše na disk a jej M sa nastaví na 0, teda sa z nej stane stará nezmenená stránka.
- Ak sa pri prezretí celého zoznamu nenašla stará nezmenená stránka, ale iniciovali sme zápis niektorých starých zmenených stránok, tak proces, ktorý spôsobil výpadok stránky, zostane čakať na ukončenie zápisu niektorej z nich.
- Ak sa pri prezretí celého zoznamu nenašla žiadna stará stránka (bez ohľadu na jej M), tak použijeme ako obeť najdávnejšie použitú novú stránku.

Prehľad algoritmov výmeny stránok

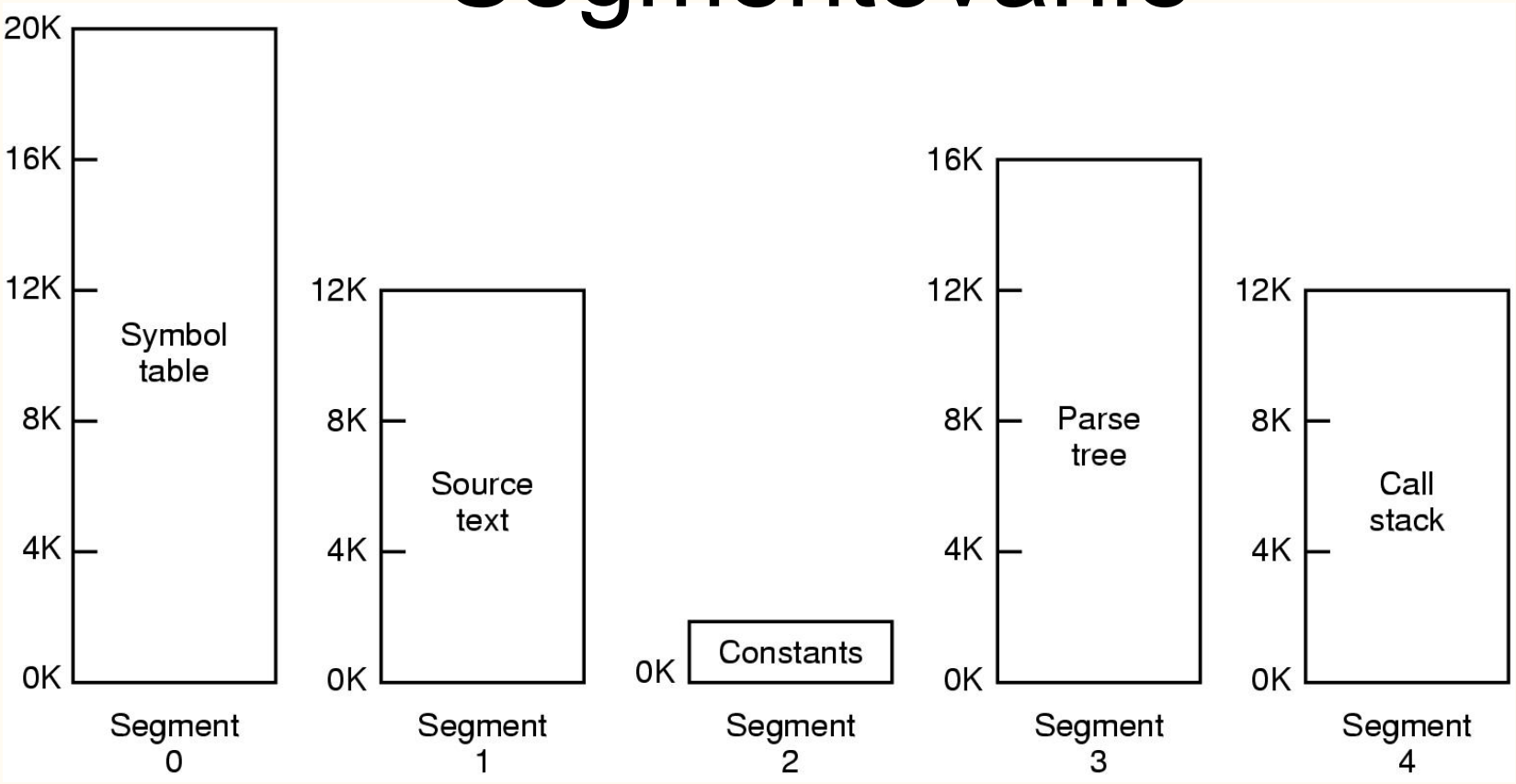
	Algoritmus	Komentár
	Optimálny	Nedá sa implementovať
Jednoduché algoritmy	NRU	Veľmi hrubý
	FIFO	Môže obetovaťpoužívané stránky
	Druhá šanca	Podstatné vylepšenie FIFO
	Hodiny	Použiteľný algoritmus
LRU	LRU	Výborný ale ťažko implementovateľný
	NFU	Veľmi hrubé priblíženie k LRU
	Starnutie	Dobré a efektívne priblíženie k LRU
Prac. množina	Pracovná množina	Drahá implementácia
	WSClock	Dobrý a efektívny

Segmentovanie - motivácia



- Jednorozmerný adresný priestor s niekoľkými dynamickými dátovými štruktúrami
- Programátor ich musí sám umiestniť na konkrétne adresy a starať sa o situácie keď niektorej štruktúre nestačí vymedzený interval

Segmentovanie



- Virtuálny adresný priestor pozostáva zo segmentov - úsekov pamäti nerovnakej veľkosti
- Dvojrozmerný adresný priestor - adresuje sa dvojicou - (číslo segmentu, adresa v segmente)
- Programátor má k dispozícii služby na vytváranie, rušenie a zmenu veľkosti segmentov
- Ukladanie do jednorozmernej fyzickej pamäte má na starosti operačný systém