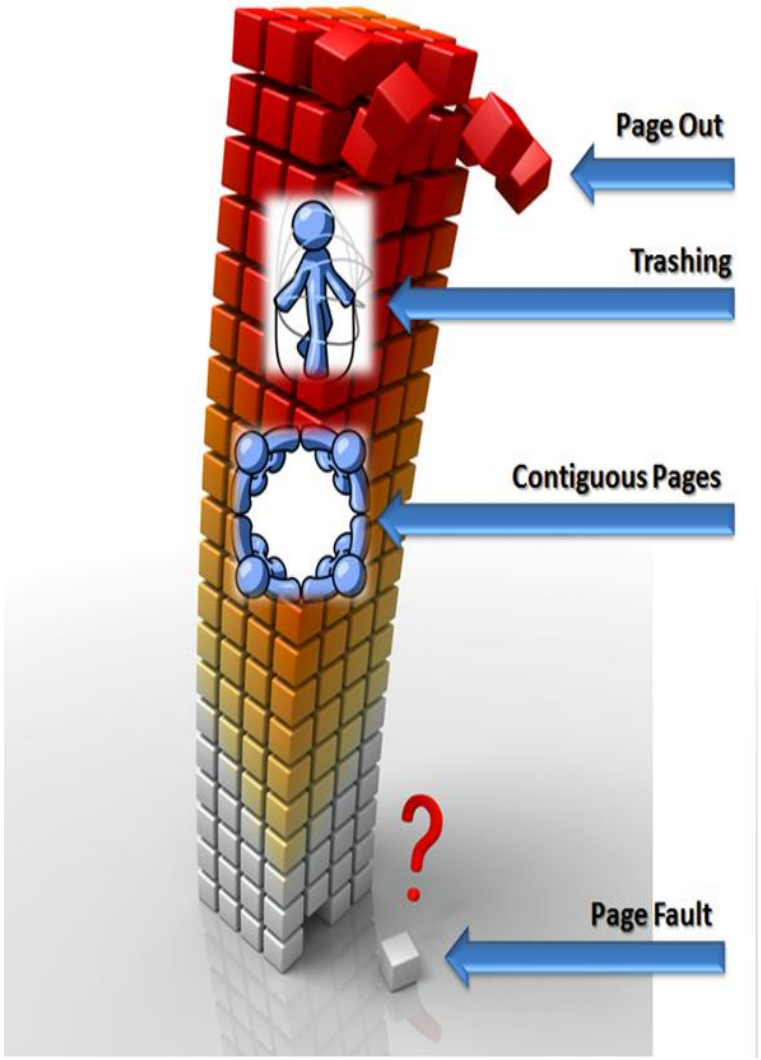
Operacinės sistemos

N. Sarafinienė

2013m.

Kalbėsime

- Puslapio trūkumo apdorojimas
- Puslapių mainai
- Rezidentinis puslapių kiekis
- Puslapių keitimo algoritmai
- Segmentavimas

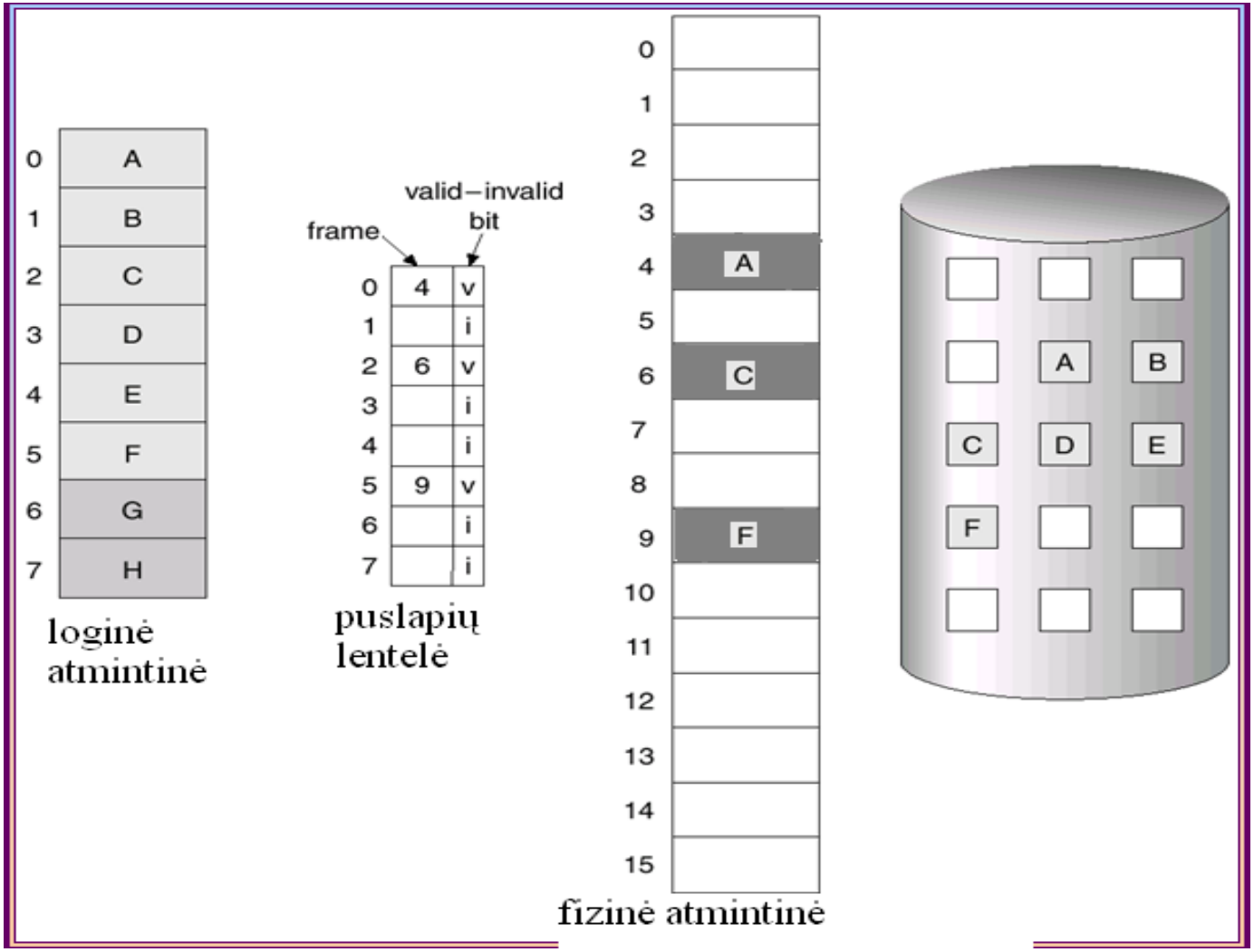


Operacinė sistema ir puslapių mainai

OS užsiima puslapiavimu 4 atvejais

1. Proceso sukūrimas
   - nustatomas programos dydis
   - Sukuriama puslapių lentelė
2. Proceso vykdymas
   - MMU (atmintinės valdymo įrenginys gauna informaciją apie naują procesą)
   - TLB (translation lookaside buffer) buferis išvalomas
3. Esant puslapio trūkumui
   - Nustatomas virtualus adresas, sukėlęs puslapio trūkumą
   - Iškeliamas kažkuris puslapis į diską ir reikiamas puslapis įkeliamas
4. Procesui baigus darbą
   - Atlaisvinama puslapių lentelė, puslapių rėmai

Puslapio trūkumas (page fault)



Puslapio trūkumo apdorojimas

1. Techninė įranga praneša branduoliui
2. Išsaugomi pagrindiniai registrai
3. OS nusprendžia, koks virtualus puslapis yra reikalingas
4. OS patikrina ar adresai yra teisėti, ieško tinkamo pagr. atmintinės rėmo.
5. Jei parinktas rėmas yra modifikuotas (dirty), jis perrašomas į diską.

Puslapio trūkumo apdorojimas

6. OS įkelia naują puslapį iš disko
7. Atnaujinamos puslapių lentelės

Grįžtama prie komandos sukėlusios puslapio trūkumą:

- Planuojamas proceso tęsimas
- Atstatomi registrai
- Programa tęsia savo veiksmus

Puslapių mainai

- Operacinė sistema, įkeldama puslapius į pagrindinę atmintinę gali elgtis įvairiai:
  - tai gali būti atliekama atsiradus tam tikro puslapio poreikiui (demand paging)
    - pradžioje gali būti įkeliamas tik pirmas reikalingas puslapis
    - po to keliamas tas puslapis, į kurį vyksta kreipinys
  - arba operacinė sistema gali bandyti nustatyti , kurių puslapių procesui reikės ir iš anksto įkelti keletą puslapių (prepaging).

Rezidentinis puslapių kiekis

- Rezidentinio kiekio minimumo bei maksimumo reikšmės.
  - Maksimalų kiekį riboja aktyvių procesų kiekis
  - Maksimalų kiekį riboja laisvų rėmų kiekis.
- Priskyrus mažai puslapių gali smarkiai išaugti puslapių mainai.
- Minimalų kiekį nusako architektūros naudojamos komandos:
  - Turi valdyt tiek skirtingų rėmų, į kiek skirtingų rėmų gali kreiptis bet kuri komanda.
  - Komanda taip pat gali būti susieta su kelias rėmais
- Kai laisvos pagrindinės atmintinės kiekis nukrinta žemiau leistinos ribos, operacinė sistema iškelia tuos proceso puslapius, kurie viršija minimumo reikšmę.
- Jei procesas turi mažiau puslapių nei min – procesas suspenduojamas

Fiksuotas priskyrimas

- Vienodas – pvz., jei yra 100 rėmų ir 5 procesai, tai kiekvienam skirti po 20 psl.
- Proporcionalus skyrimas – pagal proceso dydį.
  - $s_i = \text{size of process } p_i$
  - $S = \sum s_i$
  - $m = \text{total number of frames}$
  - $a_i = \text{allocation for } p_i = \frac{s_i}{S} \times m$

$$m = 64$$
$$s_i = 10$$
$$s_2 = 127$$
$$a_1 = \frac{10}{137} \times 64 \approx 5$$
$$a_2 = \frac{127}{137} \times 64 \approx 59$$

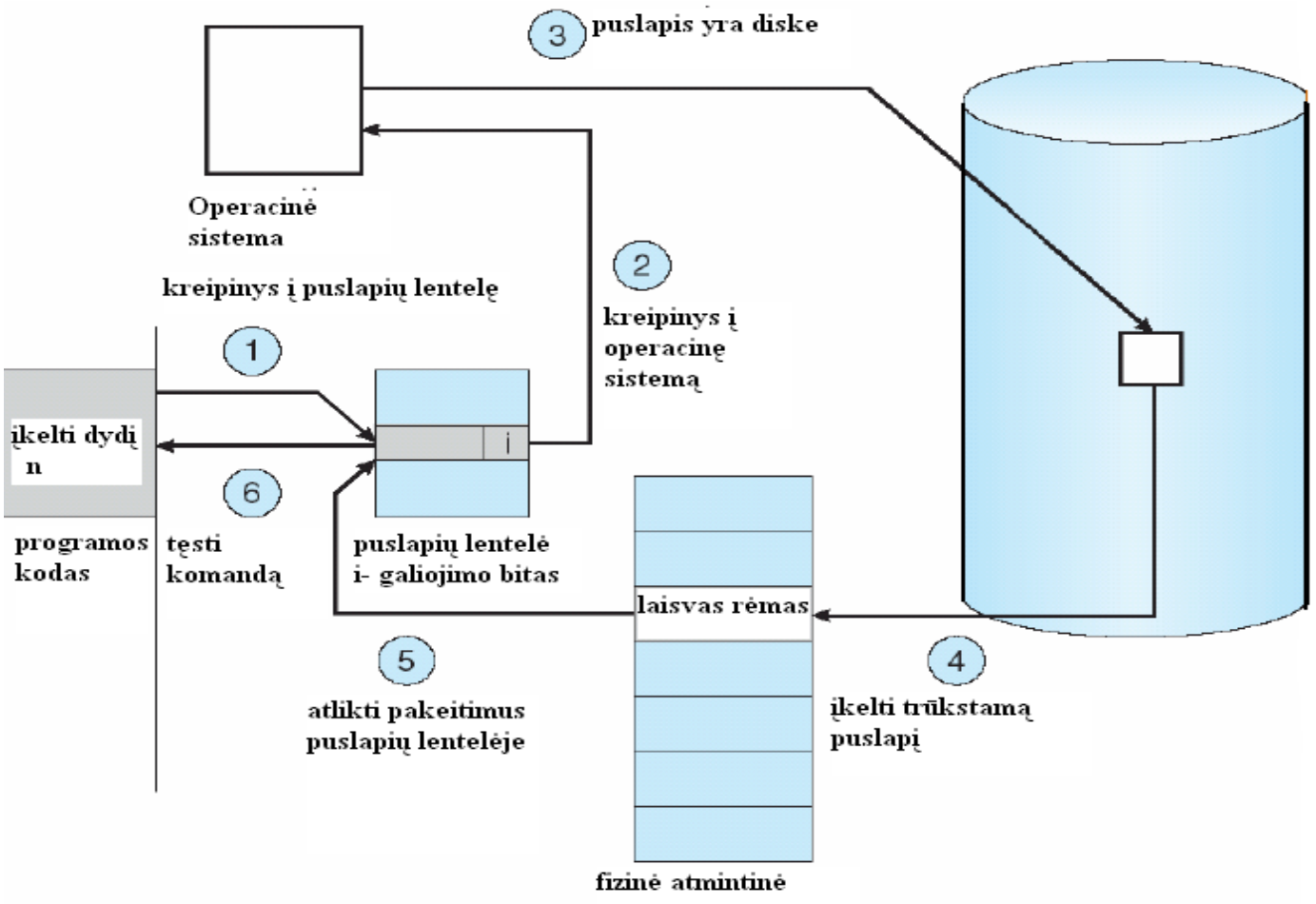
Lokali ir globali puslapių keitimo politika

- Keičiant puslapius galima:.
  - *Lokali politika* – keičiamas tam pačiam procesui priklausantis puslapis nauju puslapiu.
  - *Globali* – keičiamas bet kuriam procesui priklausantis puslapis.

Puslapių mainai

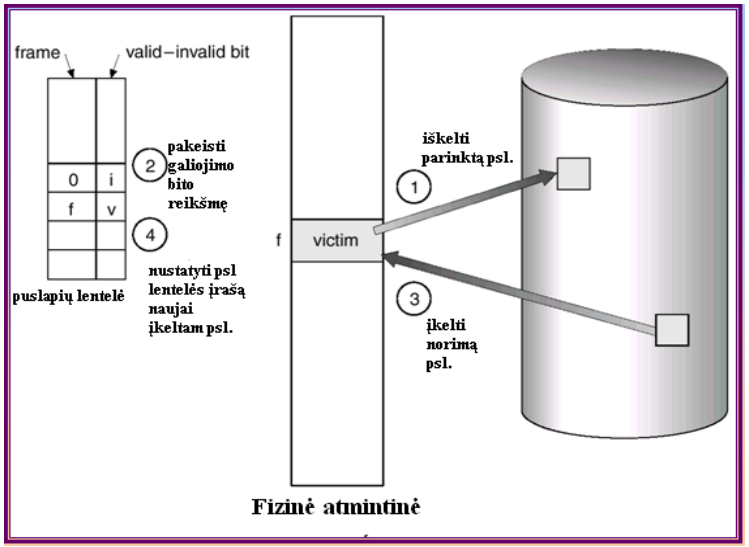
- Mainus sukelia tai, kad atmintinės kiekis yra ribotas.
- Operacinė sistema, norėdama įkelti trūkstamą puslapį turi surasti jam vietą:
  - Ieškomas laisvas rėmas.
  - Jei laisvo rėmo nėra, OS turi išlaisvinti vieną iš užimtų rėmų.
- Trūkstamas puslapis įkeliamas, suaktyvinus skaitymo iš disko užduotį.
  - Įkėlus puslapį daromi pakeitimai puslapių lentelėje,
  - įjungiamas įkelto puslapio galiojimo bitas
  - išjungiamas iškeliamo puslapio galiojimo bitas

Trūkstamo puslapio įkėlimo veiksmai



Puslapių keitimas

- Reikia tam tikro algoritmo
- Funkcionalumas – reikia tokio algoritmo, kuris garantuotų mažiausią puslapių trūkumų kiekį
- Kai kurie proceso puslapiai gali būti kelis kartus įkeliami į atmintinę



Puslapio trūkumo apdorojimas

- Puslapio trūkumo tikimybė $0 \leq p \leq 1.0$
  - jei $p = 0$ puslapio trūkumo situacijos nėra
  - if $p = 1$, kiekvienas kreipinys į atmintinę sukelia puslapio trūkumą

- Kreipinio į objektą laikas (EAT)

EAT = (1 – *p*) x kreipimosi į pagr_atm_laikas
+ *p* (psl. trūkumo iššauktas apdorojimas +
+ [pusl. iškėlimas] +
+ [pusl. įkėlimas] +
+ [proceso restarto veiksmai])

Puslapių keitimo algoritmai

- Įvykus pertraukimui dėl puslapio trūkumo reikia parinkti puslapį kurį reikia keisti.
- Optimalu būtų pakeisti tą puslapį, kurio nereikės ilgiausią laiką.
  - Realiai OS neturi kaip sužinoti kada ir į kokį puslapį bus kreipiamasi, tačiau gali būti fiksuojami buvę kreipiniai.

Puslapių keitimo strategijos

- Keičiamas puslapis, kurio prireiks vėliausiai.
- Keičiamas atsitiktinai parinktas puslapis.
- Keičiamas paskutiniu metu nenaudotas puslapis (NRU).
- Puslapiai keičiami laikantis FIFO disciplinos.
- Laikrodžio (clock) algoritmas.
- Keičiamas mažiausiai paskutiniu metu naudotas puslapis LRU (Least Recently Used)
- Nedažnai naudoto puslapio keitimas NFU (Not frequently used)

Keičiamas puslapis, kurio prireiks vėliausiai

- Reiktų keisti tą puslapį, kurio prireiks vėliausiai
  - yra panašus į procesų planavime naudojamą algoritmą- „trumpiausias procesas pirmas“,
  - **šis algoritmas yra optimalus.**

Jis nėra praktiškai taikomas:

- sunku nusakyti, kokie kreipiniai ir į kurį puslapį ir kokia tvarka vyks ateityje, operacinė sistema to nežino.

- Šis algoritmas tiesiog taikomas kitų algoritmų palyginimui.

Keitimui parinkti puslapį atsitiktinai

- Tokį algoritmą nesunku įgyvendinti, tačiau jis nepasižymi geru funkcionalumu.

NRU algoritmas

- Keičiamas paskutiniu metu nenaudotas puslapis (Not Recently Used).
- Stengiamasi atmintyje išlaikyti tuos puslapius, į kuriuos nesenai buvo kreiptasi.
  - Yra nagrinėjami du bitai, kurie surišami su kiekvienu puslapiu, esančiu pagrindinėje atmintinėje:
    - R bitas, kurio vienetinė reikšmė rodo, kad į puslapį yra kreiptasi
    - bei M bitas, kuris rodo, kad puslapis yra pakeistas (modifikuotas).
- Esant laikrodžio mechanizmo pertraukimams, kurie vyksta kas tam tikrą laiko intervalą visi kreipimosi į puslapius R bitai yra verčiami į 0, taigi vienetinė R bito reikšmė rodo, kad į atitinkamą puslapį buvo kreiptasi šiame trumpame laiko intervale.

NRU

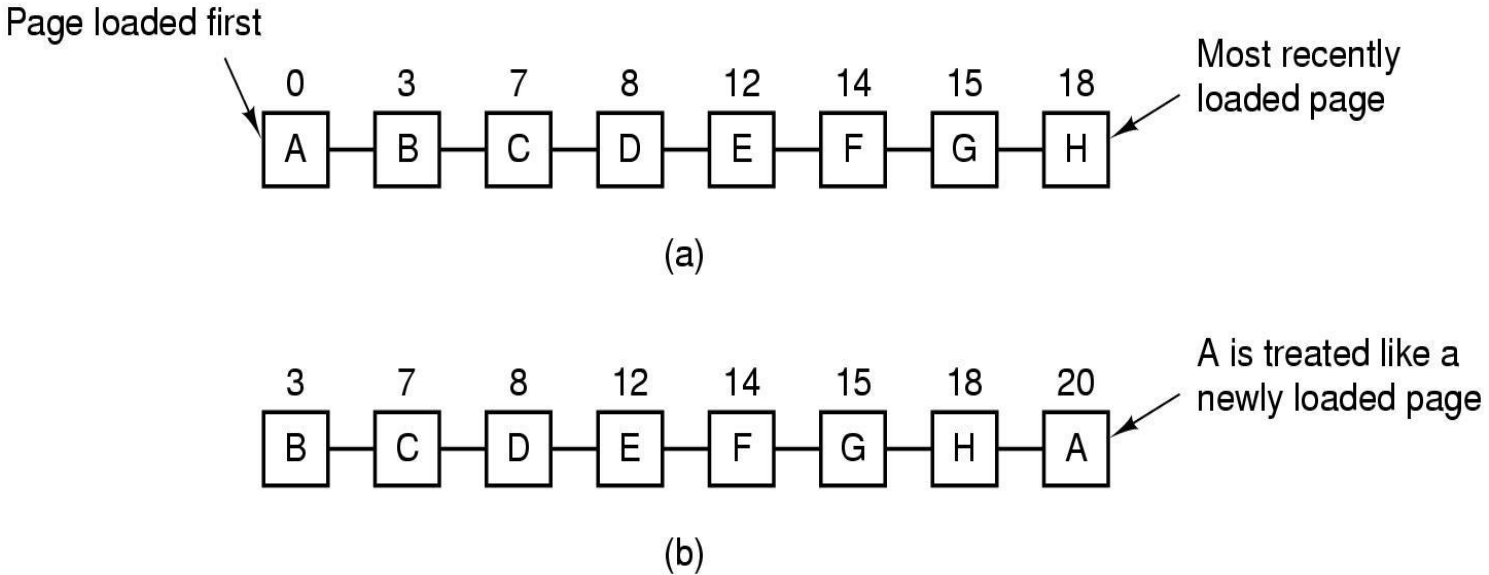
- Esant „puslapio trūkumo“ pertraukimui visi pagrindinėj atmintinėj esantys puslapiai suskirstomi į 4 klases pagal bitų R bei M reikšmes:
  - 1 klasė: R=0, M=0
  - 2 klasė: R=0, M=1
  - 3 klasė: R=1, M=0
  - 4 klasė: R=1, M=1
- Taikant NRU algoritmą yra pasirenkamas atsitiktinis puslapis iš žemiausios netuščios klasės. Pagal šį algoritmą yra skaitoma, kad puslapis, į kurį buvo nesenai kreiptasi yra svarbesnis, nei modifikuotas puslapis

FIFO

- Paprastas algoritmas, surikiuojant į eilę puslapius, įkeltus į pagrindinę atmintinę.
- Eilės priekyje yra seniausi puslapiai.
- Eilės gale – šviežiausi (naujausi puslapiai).
- Nors iš pažiūros tai ir teisinga taktika, tačiau taikant šį algoritmą nėra gaunami geri rezultatai, nes yra išmetami vienodu dažniu tiek tie puslapiai, kurie nėra dažnai naudojami, tiek tie, į kuriuos pastoviai vyksta kreipiniai.

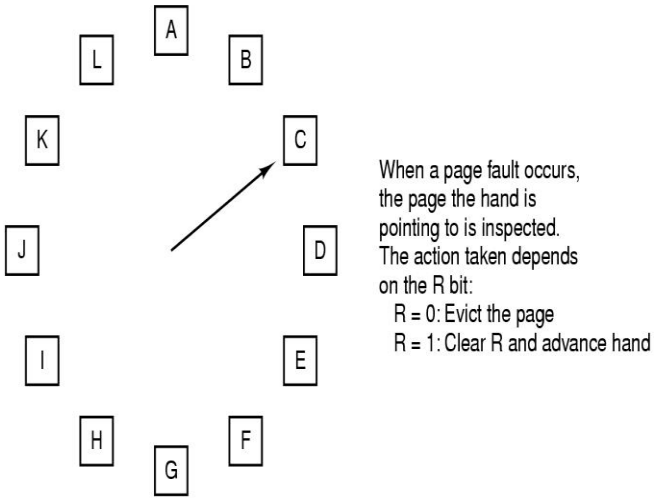
„antro šanso“ FIFO
FIFO with second chance

- Veikia kaip FIFO, bet tikrina R bitą.
- Jei R bitas eilės pradžioj yra 0, šis puslapis keičiamas, o jei 1, jo reikšmė išvaloma ir puslapis dedamas į eilės galą.
- Kas atsitinka, jei visų puslapių R yra 1?

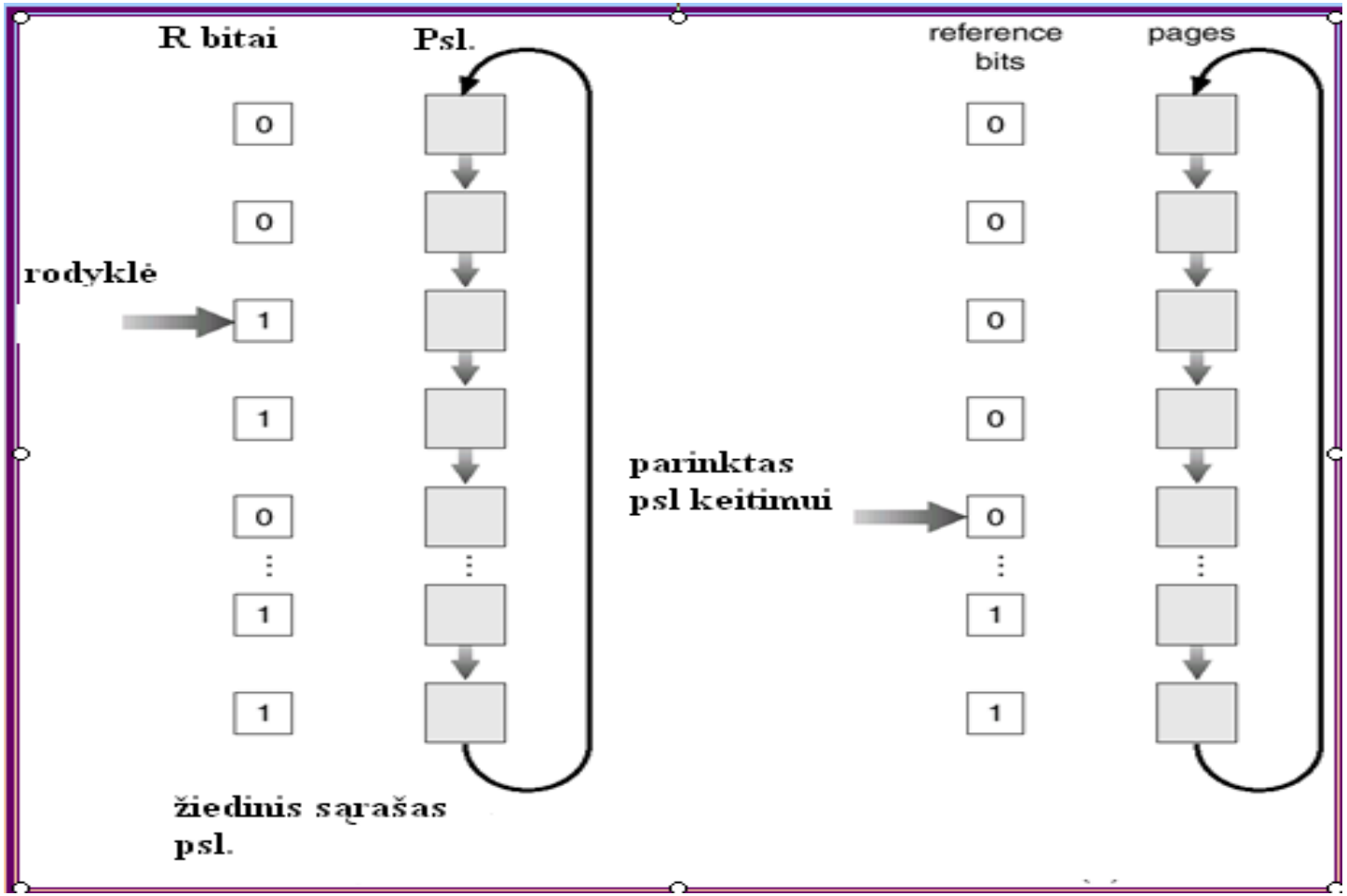


Laikrodžio (Clock) algoritmas

- Yra dar efektyvesnis algoritmas nei „antro šanso“ algoritmas.
- yra palaikomas žiedinis sąrašas, kuriame laikrodžio „rodyklė“ rodo į seniausią puslapį
- Jei esant puslapio mainams:
  - šiam puslapiui R bito reikšmė lygi 1, tai ji keičiama į nulį, o laikrodžio rodyklė pasislenka prie sekančio puslapio.
  - O jei R bito reikšmė lygi 0, tai naujas puslapis yra įkeliamas vietoje šio puslapio ir rodyklė taip pat pasislenka link sekančio puslapio.
  - Visų puslapių R bito reikšmė kas kažkiek laiko yra keičiama į 0.



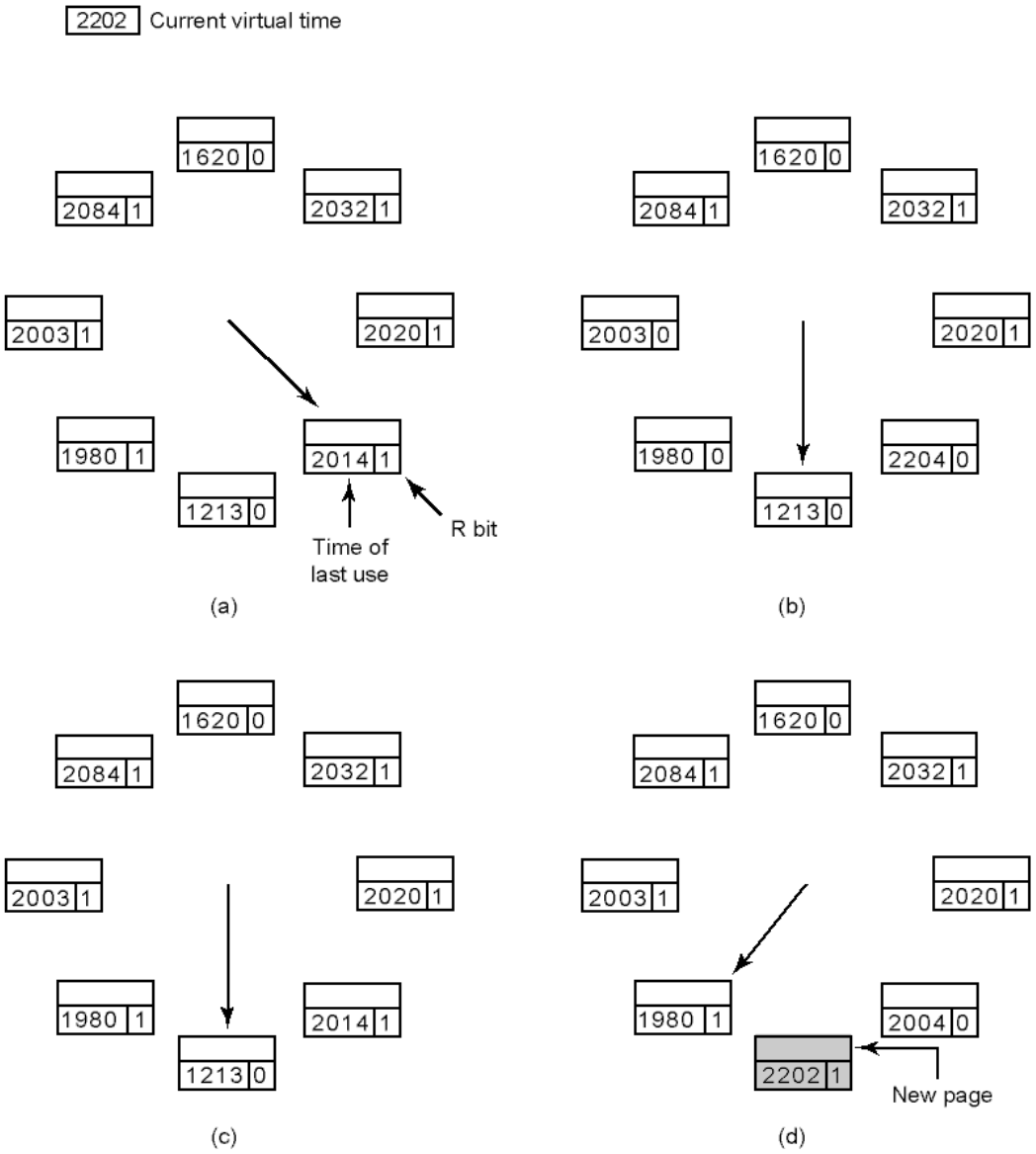
Laikrodžio (Clock) algoritmas



WSclock algoritmas

- Praktikoje yra naudojama laikrodžio algoritmo modifikacija WSclock algoritmas, kuris vertina kreipimosi į puslapį laiką.
- Kiekviename puslapyje yra saugomas įrašas su paskutinio kreipimosi laiku.
- Kai reikia rasti keičiamą puslapį, yra nagrinėjamas puslapis, į kurį rodo laikrodžio „rodyklė“.
  - Jei šiam puslapiui **R=1**, tai kreipimosi laikas yra nustatomas į virtualų laiką (jis gali būti sutapatinamas su tuo laiku, kiek laiko yra to puslapio procesas gavęs CPU aptarnavimo)
  - ir bitas R nustatomas į 0,
  - o laikrodžio „rodyklė“ pasukama tolyn.
  - Jei R=0, tai įvertinama, ar šio puslapio paskutinio kreipimosi laikas skiriasi nuo virtualaus laiko daugiau nei tam tikra konstantė, ir jei taip, tai šis puslapis gali būti keičiamas, jei ne – ieškomas sekantis puslapis, nes laikoma, kad šis puslapis dar reikalingas procesui.
- Šio algoritmo modifikacijose taip pat gali būti vertinamas ir M bito reikšmė. Kai laikrodžio rodyklė slenka per puslapį, kuriam yra įjungta M bito reikšmė, tai atitinkamas puslapis fiksuojamas kaip puslapis, kurį reikia perrašyti į diską, tuo pačiu išvalant M bito reikšmę.

WSClock algoritmo pavyzdys



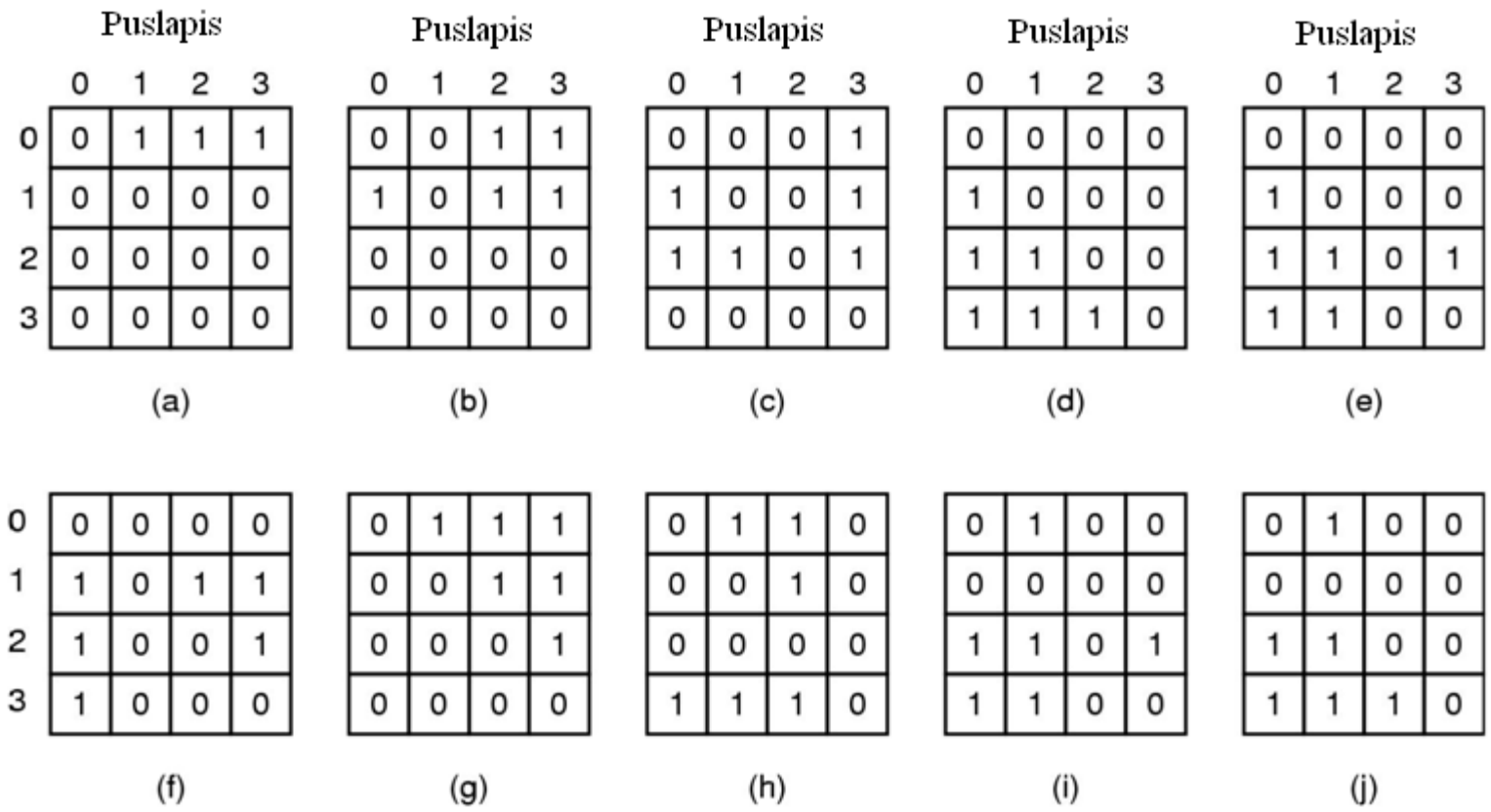
Mažiausiai paskutiniu metu naudoto puslapio keitimas
Least Recently Used -LRU

- Tikima, kad tie puslapiai, kurie yra naudojami bus naudojami vėl netrukus.
- Galimi du būdai:
  - Palaikomas surištas puslapių sąrašas.
    - Prieky- nesenai naudoti puslapiai
    - Sąrašas atnaujinamas po kiekvieno kreipinio į atmintinę!!
  - Naudojamas skaitliukas kiekviename puslapių lentelės įraše.
    - Renkamas puslapis su mažiausia skaitliuko reikšme
    - Periodiškai skaitliukai nustatomi į 0

LRU (tęsinys...)

- Alternatyva – n*n matrica , čia n – puslapio rėmų kiekis.
  - Visos pradinės reikšmės pradžioje 0
  - Kai kreipiamasi į k-tą rėmą, k-toje eilutėje visi bitai nustatomi į 1
  - Visi bitai k-tame stulpelyje nustatomi į 0.
  - Eilutė, kurios dvejetainė reikšmė yra mažiausia nusako nr puslapio kurį reikia keisti.

LRU pavyzdys



Į puslapius kreipiamasi taip: 0, 1, 2, 3, 2, 1, 0, 3, 2, 3

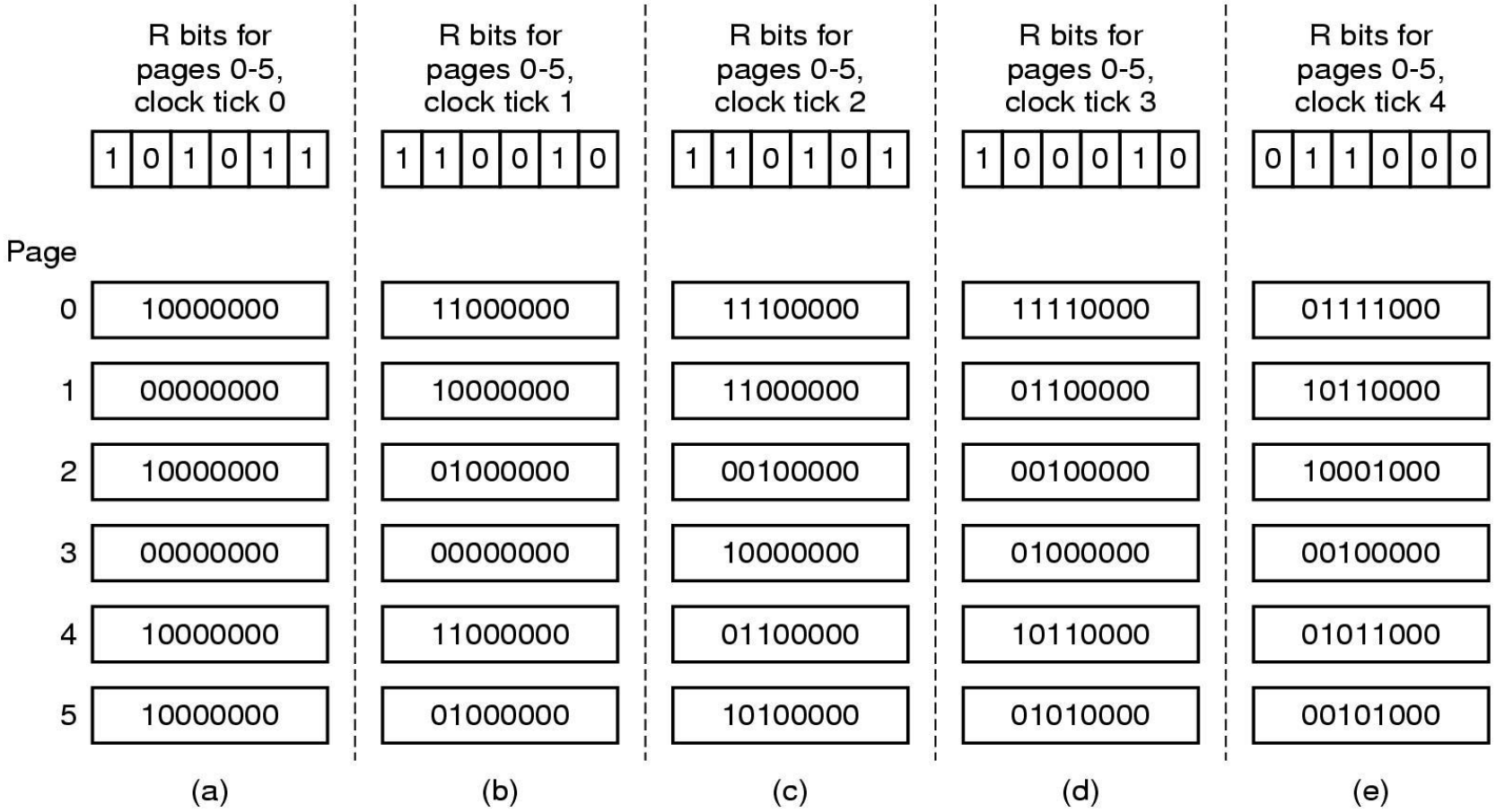
Imituotas LRU – Nedažnai naudoto puslapio keitimas (NFU)

- Dauguma kompiuterinių sistemų neturi techninio palaikymo tiksliai LRU algoritmo realizacijai.
- Šis algoritmas gali būti imituojamas.
- Galima naudoti skaitliukus, kurie registruotų visų puslapių R bitus sulig kiekvienu kompiuterio laikrodžio “tiksėjimu” .
- Puslapis su mažiausia R bitų reikšme yra parenkamas keitimui.

Modifikuotas NFU

- Visi skaitliukai yra perstumiami dešinėn per 1 bitą prieš vykdant R bito pridėjimą.
- R bitas yra pridedamas prie kraštinio kairiojo bito.
- Šis modifikuotas algoritmas žinomas kaip sendinimo (aging).

Modifikuotas NFU (Aging)

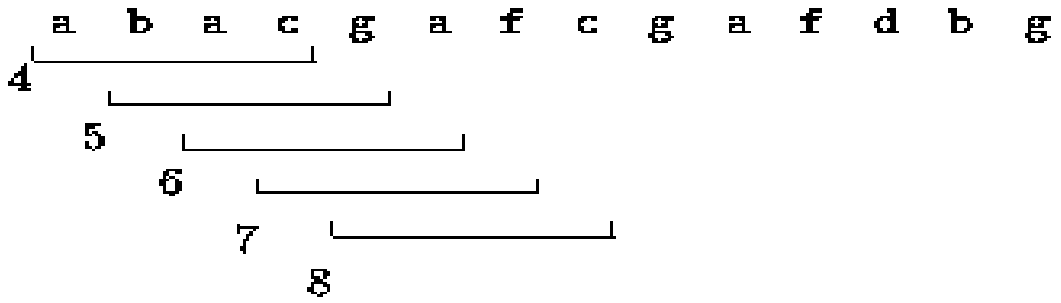


Rezidentine seka grindžiamas puslapių keitimas (Working Set Page Replacement)

- Kreipinių lokališkumo principas teigia, kad trumpame laiko intervale procesas kreipiasi tik į nedidelę seką savo puslapių.
- Puslapių seka, kuriuos procesas naudoja vadinama darbine seka (rezidentine) seka.
- *Rezidentine seka grindžiamas algoritmas keitimui siūlo puslapius, kurie daugiau nebepriklauso darbinei sekai.*
- *Darbinė seka fiksuojama T kreipinių (window size) į atmintinę metu.*
- *Puslapis gali būti pakeičiamas nebūtinai įvykus pertraukimui pagal puslapio trūkumą.*

Rezidentine seka grindžiamas puslapių keitimas (Working Set Page Replacement)

- Turim programą su 7 virtualiais puslapiais {a,b,...,g} į kuriuos kreipiamasi šia seka, vertinami paskutiniai 4 kreipiniai:
  - a b a c g a f c g a f d b g



lentelė rodo darbinę puslapių seką po kiekvieno kreipinio į atmintinę

1 a	8 acgf
2 ab	9 acgf
3 ab	10 acgf
4 abc	11 acgf
5 abcg	12 agfd
6 acg	13 afdb
7 acgf	14 fdbg

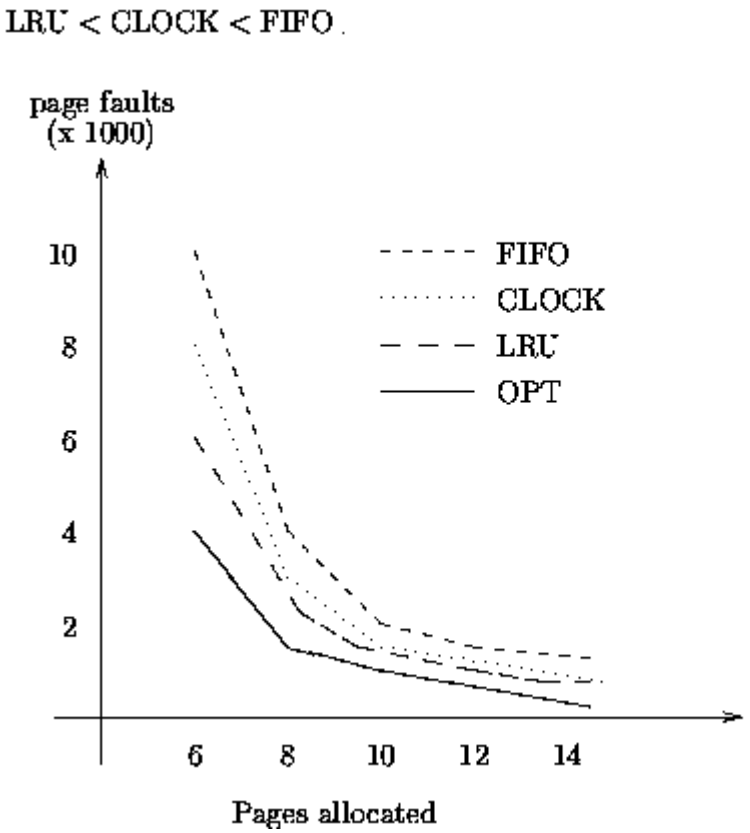
Išankstinis puslapių įkėlimas

- *Paprastai puslapis įkeliamas į atmintinę, iškilus puslapio trūkumo situacijai (demand paging).*
- *Iškilus puslapio trūkumui gali būti įkeliamas ne vienas, o keli puslapiai (demand prepaging).*
  - Tikimasi, kad jų greitai prireiks.
  - Sumažinamas pertraukimų pagal puslapių trūkumą kiekis.
  - Padidėja CPU efektyvumas
- Išankstinis puslapių įkėlimas taip pat sutinkamas proceso "restart" situacijoje:
  - Įsimenami puslapiai sudarę proceso rezidentinių puslapių kiekį.
  - Prieš proceso "restartą" proceso puslapiai yra įkeliami. (prepaging).

Puslapių keitimo algoritmų suminė lentelė

Algoritmas	Komentaras
Optimalus	Nėra diegiamas, bet naudojamas palyginimui
NRU (Paskutiniu metu nenaudotas)	Primityvus
FIFO	Gali išmesti svarbius puslapius
Sekančio šanso	Ryškiai patobulintas FIFO
Laikrodžio	Realistinis
LRU (Mažiausiai paskutiniu metu naudotas)	Puikus, bet tikslus taikymas sunkus
NFU (Nedažnai naudotas)	Nebloga LRU aproksimacija
Sendinimo (aging)	Efektyvus algoritmas, kuris gerai aproksimuoja LRU
Darbo seka grindžiamas	Išlaidus diegimo kaštams algoritmas
WSClock	Geras, efektyvus algoritmas

Puslapių trūkumų kiekio priklausomybė nuo priskirto psl. kiekio



Pavyzdyje parodytas algoritmų funkcionavimas, esant nedidelei programai su nedideliu priskirtų puslapių kiekiu.

- Thrashing – puslapių kilojimas – kai programai pritrūksta puslapių labai dažnai, vyksta tik puslapių mainai, ir jokio naudingo darbo.

Belady anomalija

All pages frames initially empty

	0	1	2	3	0	1	4	0	1	2	3	4
Youngest page	0	1	2	3	0	1	4	4	4	2	3	3
		0	1	2	3	0	1	1	1	4	2	2
Oldest page			0	1	2	3	0	0	0	1	4	4
	P	P	P	P	P	P	P			P	P	

9 Page faults

(a)

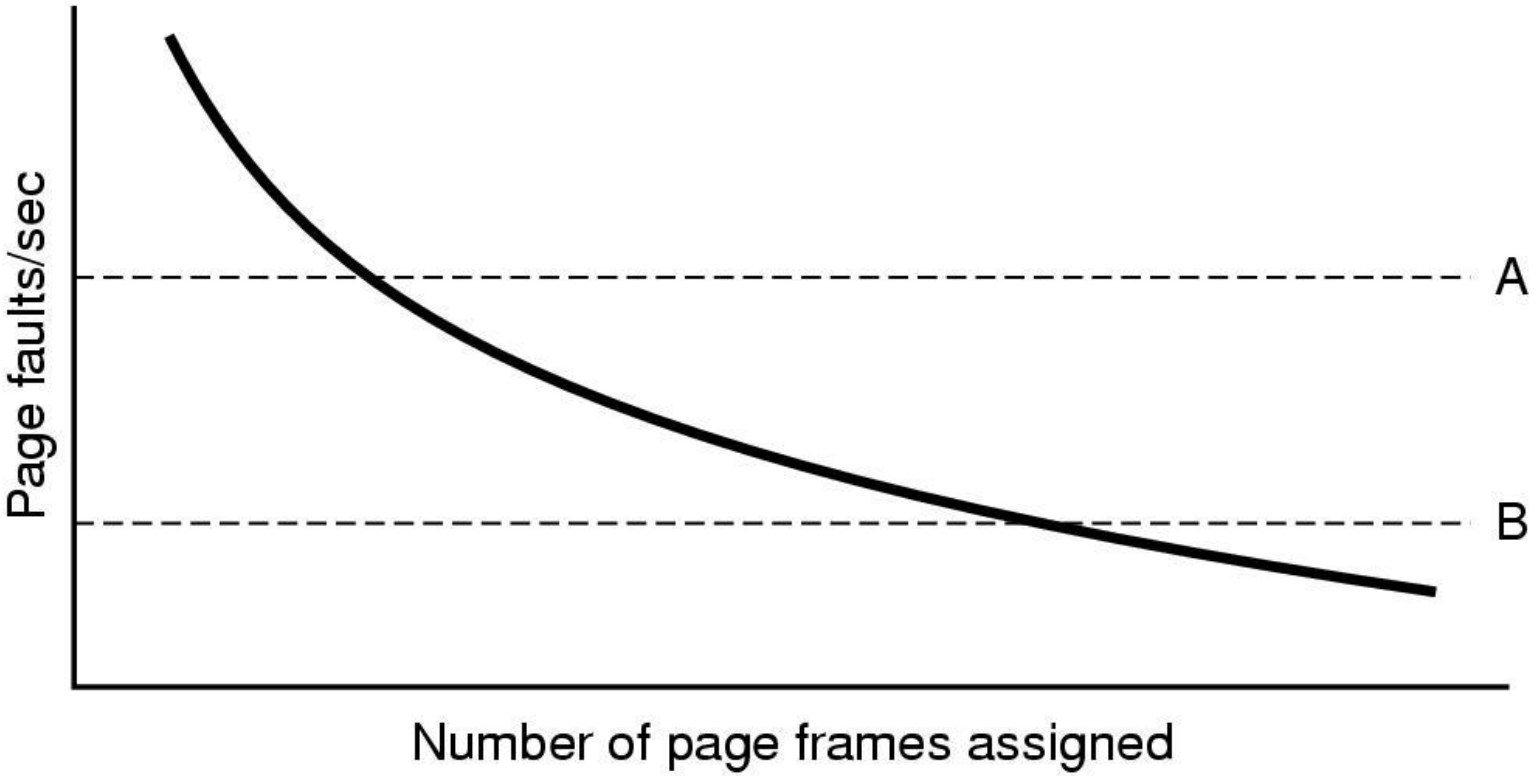
	0	1	2	3	0	1	4	0	1	2	3	4
Youngest page	0	1	2	3	3	3	4	0	1	2	3	4
		0	1	2	2	2	3	4	0	1	2	3
Oldest page			0	1	1	1	2	3	4	0	1	2
				0	0	0	1	2	3	4	0	1
	P	P	P	P			P	P	P	P	P	P

10 Page faults

(b)

- a) FIFO skiriant procesui 3 pagrindinės atmintinės rėmus
- b) FIFO skiriant 4 rėmus
- Puslapių trūkumų kiekis priklauso nuo to kokia seka vyks kreipiniai į puslapius

Puslapių mainai



Kai puslapių mainai smarkiai išauga, tenka mažinti procesų kiekį, kurie varžosi dėl atmintinės, dalis procesų iškeliama į swap sritį.

Kai proceso turimas psl kiekis artimas min – gali to nepakakti ir suaktyvėja psl mainai

“trashing” priežastys

- OS stebi CPU panaudojimą:
  - Žemas panaudojimas ->didinamas multiprogramavimo lygis
  - Naudojant globalią psl. mainų strategiją.
  - Procesai laukiantys psl. įkėlimo yra blokuojami – mažėja CPU panaudojimas-didinamas procesų kiekis.
  - Norint sumažinti psl. mainus gali tekti naudot lokalią psl. mainų strategiją arba sumažinti multiprogramavimo lygį.

Valymo politika

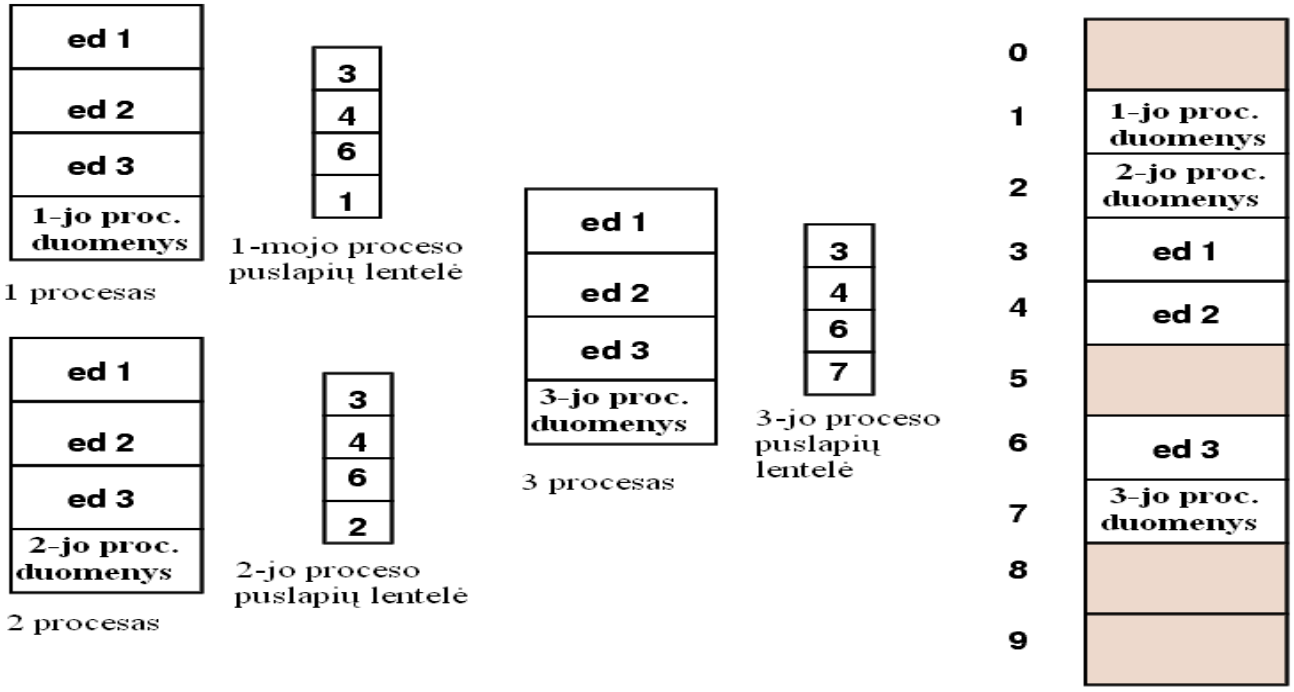
- Sistemoje veikia puslapiavimo “demono” procesas. (dažniausiai foniname režime)
- Kai lieka labai mažai laisvų rėmų, kviečiamas puslapių keitimo algoritmas.

Bendrai naudojamos atmintinės sritys

- Procesai dažnai naudojasi tom pačiom programom, funkcijom, ...
- Procesams reikia leisti šiuo kodu naudotis bendrai. Privalumai:
  - Nereikėtų laikyti daug vienodų kopijų.
  - Tai sumažintų programų naudojamos atmintinės kiekį.
- Reikalavimai:
  - Šio bendrai naudojamo kodo procesai neturėtų keisti.
  - Jis turėtų būti patalpintas srityje, kuri būtų žinoma visiems procesams.
  - Operacinė sistema tokiu atveju turi identifikuoti puslapius kaip bendrai naudojamus arba tokius, kuriais nėra bendrai naudojamasi.
  - Tokiu atveju tik kiekvieno proceso privatus kodas bei duomenys turėtų būti laikomi skirtingose atmintinės vietose.

Bendrai naudojamos atmintinės sritys

- Pavyzdys, kai trys procesai naudojasi tais pačiais puslapiais ed1, ed2, ed3.



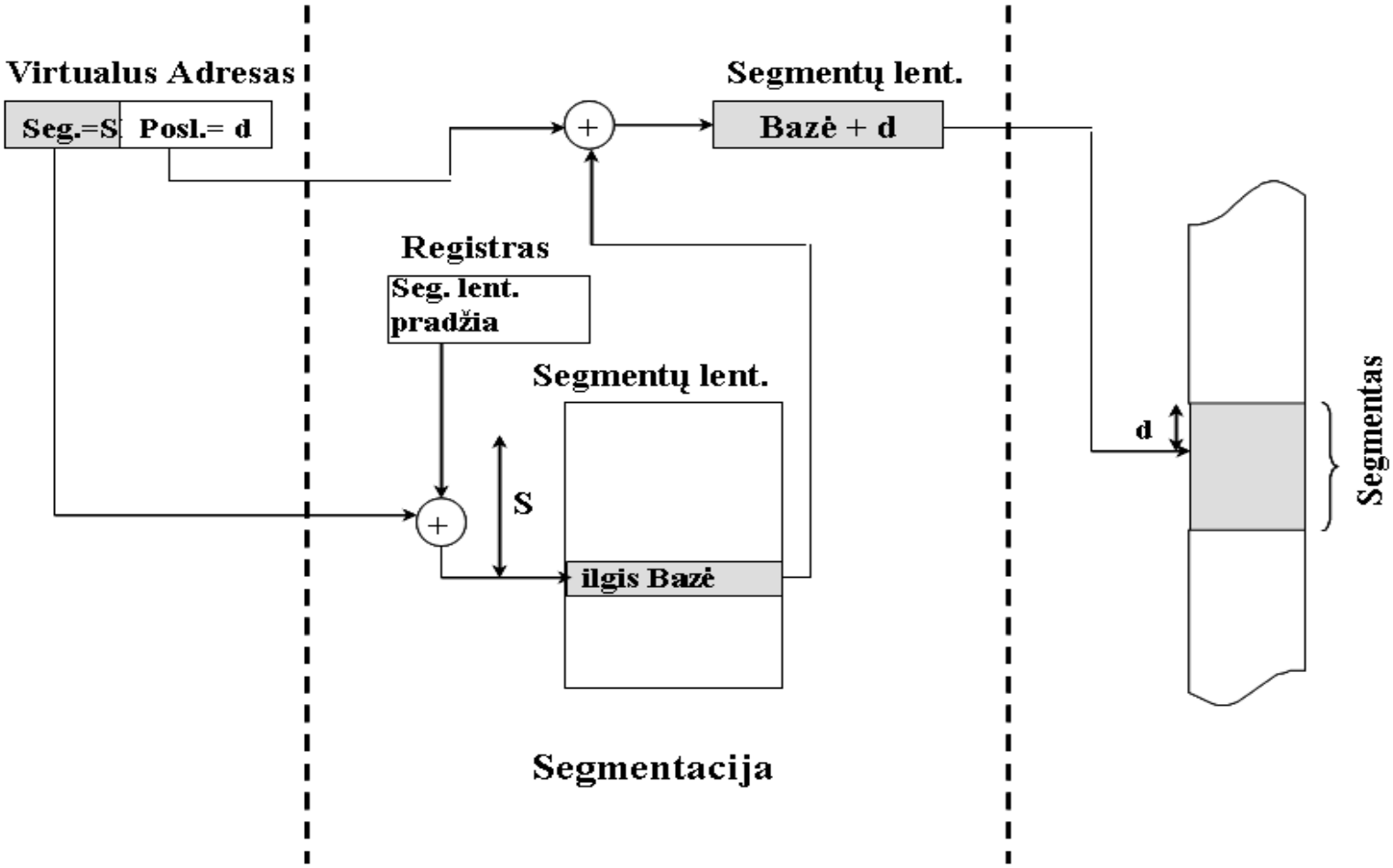
Segmentavimas

- Bazinės atmintinės skirstymo segmentais idėja yra ta, kad yra naudojama ne viena virtualios atminties sąvoka, o daug virtualių adresų erdvių sričių:
  - viena virtuali adresų erdvė programos kodui,
  - kita duomenims,
  - kita stekui ir t.t.
- Skirstant segmentais išplečiama bazinio ir ribinio adreso idėja – naudojama visa lentelė, sauganti bazinio/ribinio adreso poras kiekvienam segmentui.
- Segmentai susiejami su proceso sudėtinėm dalimis bei programos kodo moduliais:
  - segmentų kiekis yra žymiai mažesnis nei puslapių kiekis,
  - segmentų lentelės informaciją galima saugoti registruose.

Skirstymas segmentais

- Kiekvienam iš segmentų fizinėje atmintinėje yra skiriama nuoseklių adresų erdvė.
- Kiekvieno elemento adresas segmente yra nusakomas segmento numeriu bei poslinkiu segmente.
- Transliuojant adresą į fizinį adresą atmintinėje yra imamas bazinis segmento adresas, dedama poslinkio reikšmė ir gautas dydis lyginamas su ribiniu to segmento adresu. Jei gauta reikšmė yra didesnė už ribinę reikšmę gaunama segmentavimo klaida.
- Segmentų lentelės įrašas- bazinis segmento adresas ir segmento riba.

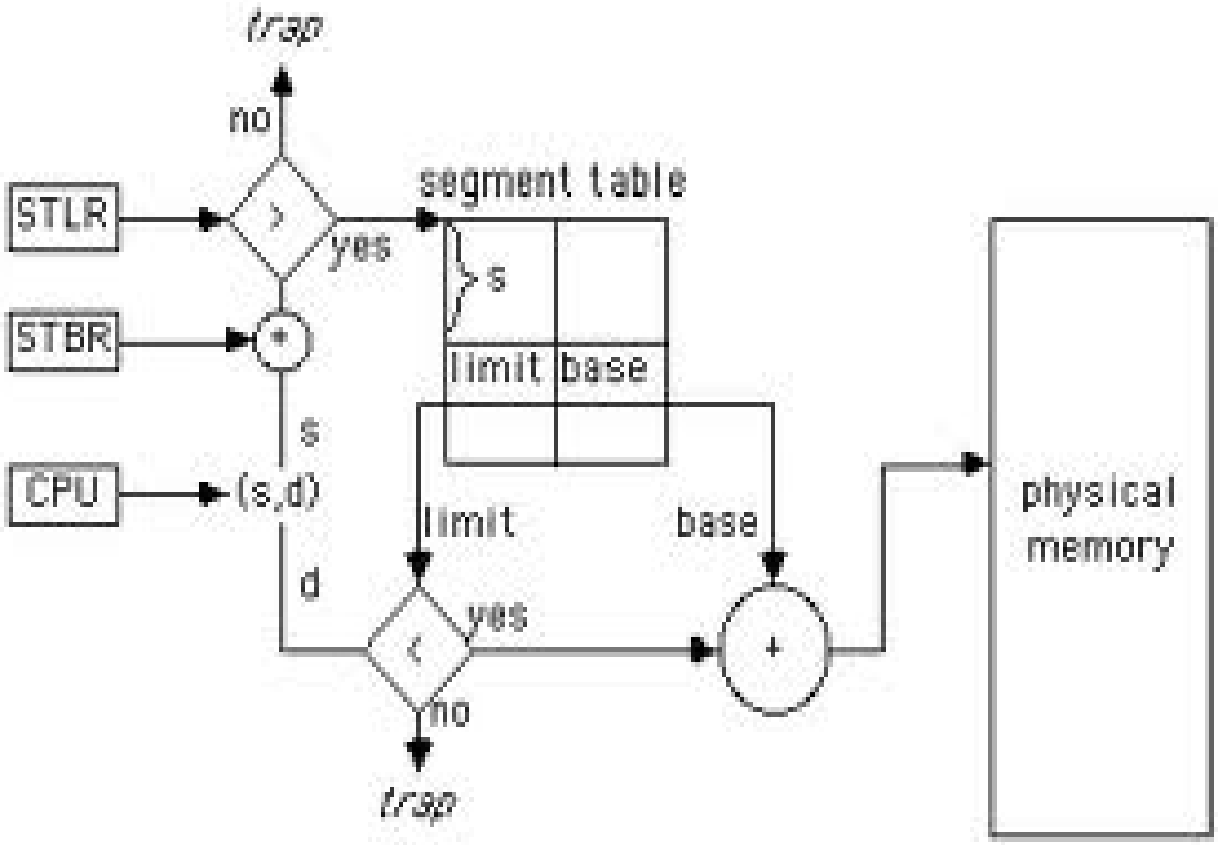
Adreso transliavimas



Segmentavimas

- Šis skirstymo būdas yra paprastesnis kompiliatoriams. Sukompiliuota programa pateikiama kaip segmentų rinkinys.
- Problema yra tai, kad :
  - segmentą reikia visą patalpinti į nuosekliai einančius pagrindinės atmintinės adresus,
    - tokios tinkamos srities suradimas yra sunki problema
    - taikant neišvengiamai susiduriama su išorine fragmentacija (atmintinėje paliekamos „skylės“).

Segmentavimas



Segmentų apsauga ir prieigos kontrolė

- segmentų apsauga ir prieigos kontrolė nusakoma apsaugos bitais.
- Jie nurodo, ar procesas gali atlikti :
  - skaitymo, rašymo, kodo vykdymo arba pridėjimo (append) veiksmus su šiuo segmentu.

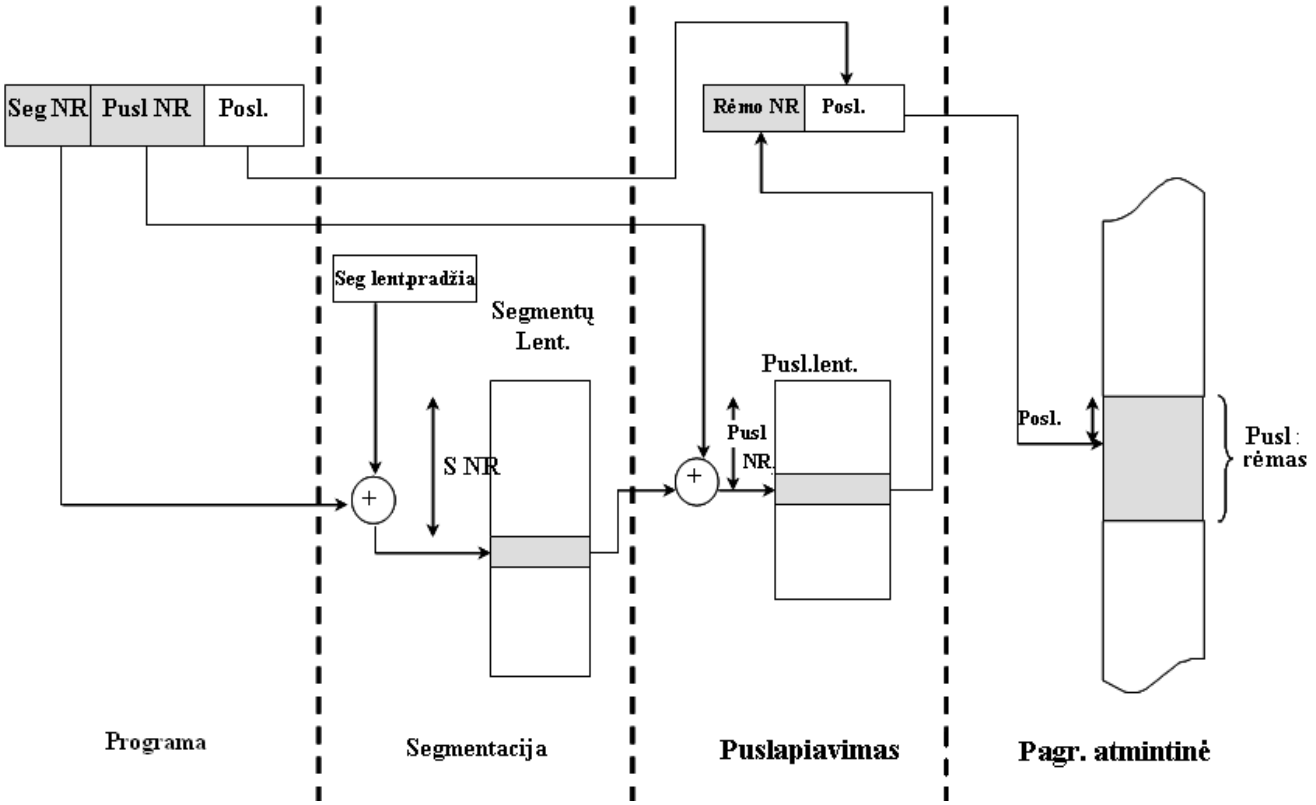
Segmentų apsaugos modos

Moda	Skaity-mo bitas	Rašymo bitas	Vykdy-mo bitas	Aprašas	Taikymas
0-nė moda	0	0	0	Jokia prieiga neleistina	Saugus
1-nė moda	0	0	1	Leistinas tik vykdymas	Segmento prieiga leistina procesams, kurie negali keisti ar kopijuoti šio segmento, bet gali jį vykdyti.
2-ji moda	0	1	0	Leistinas tik rašymas	Ši galimybė nėra naudinga, nes rašymo prieigos galimybė be skaitymo prieigos galimybės yra nepraktiška.
3-ji moda	0	1	1	Leistinas rašymas bei vykdymas	
4 –ji moda	1	0	0	Leistinas tik skaitymas	Informacijos skaitymas
5-ji moda	1	0	1	Leistinas skaitymas, vykdymas	Programa gali būti kopijuojama ar vykdoma, bet negali būti modifikuota
6-ji moda	1	1	0	Leistinas skaitymas bei rašymas	Apsaugo duomenis nuo klaidingo bandymo juos vykdyti
7-ji moda	1	1	1	Neribota prieiga	Ši prieiga garantuojama patikimiems vartotojams

Segmentų skirstymas puslapiais

- Objekto adresas yra nusakomas trimis dedamosiomis: segmento numeriu, puslapio numeriu ir poslinkio reikšme.
- Segmentų talpinimas fizinėje atmintinėje nebesukelia išorinės fragmentacijos ir yra lengvai realizuojamas
- Naudojama:
  - Segmentų lentelės
    - Įėjimas- bazinis puslapių lentelės adresas
  - Kiekvienam segmentui – atskira puslapių lentelė

Adreso transliavimas skirstant segmentus puslapiais



I LOVE My Computer Because My Friends Live In It

"I forgot to make a back-up copy of my brain, so everything I learned last semester was lost."