Operacinės sistemos

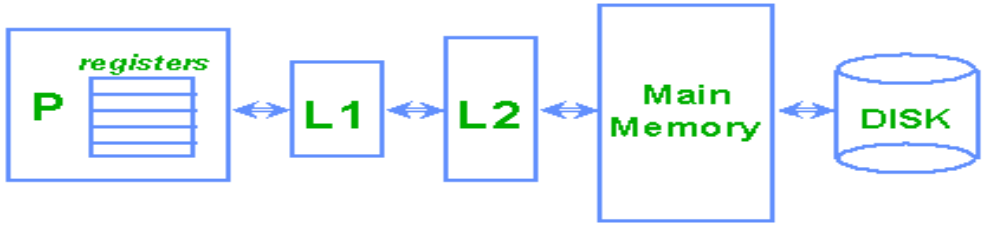
N. Sarafinienė, I. Lagzdinytė-Budnikė

2014m.

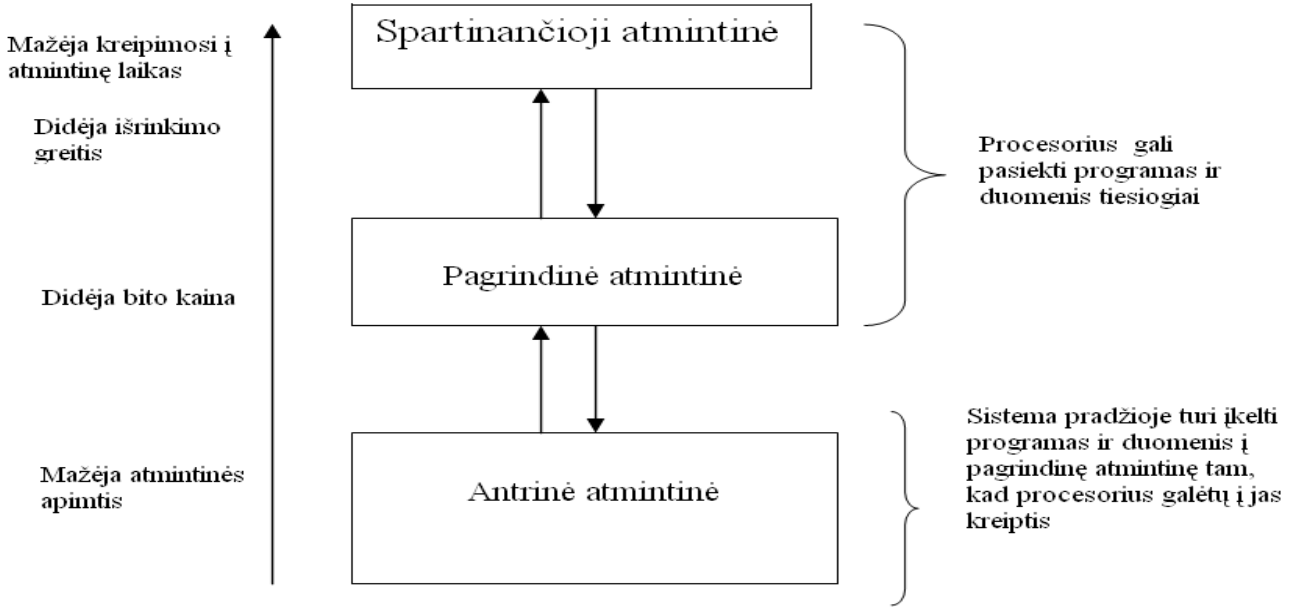
Kalbėsime

- Nagrinėsime atmintinės valdymo problemas
- Fiksuotų bei dinaminių skyrių sudarymą
- Paprastą segmentaciją
- Puslapių lenteles

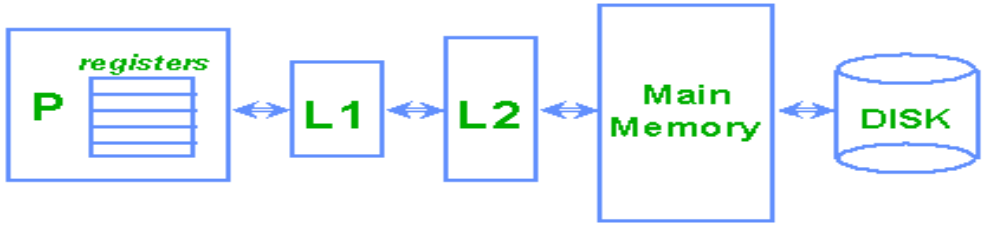
Atmintinės hierarchija ir charakteristikos



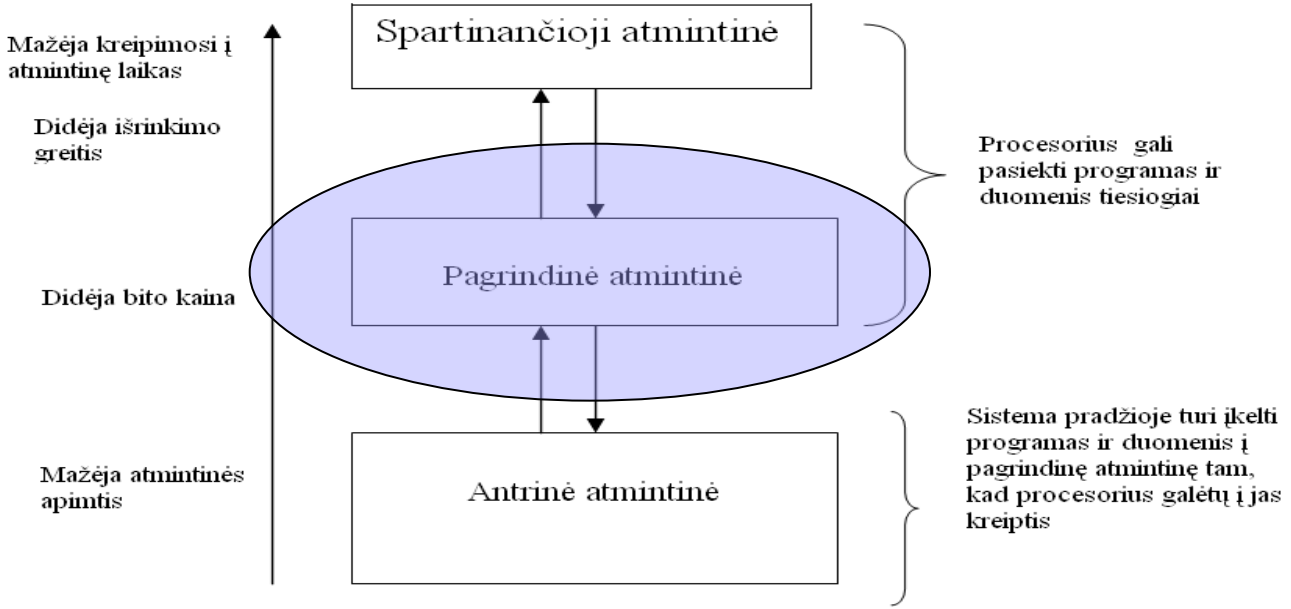
Access time	0.5 clk	1 clk	5 clks	10-50 clks	10^5 clks
Capacity	1KB	16KB	1MB	1GB	10GB
Block Size	8B	64B	128B	4-16 KB	



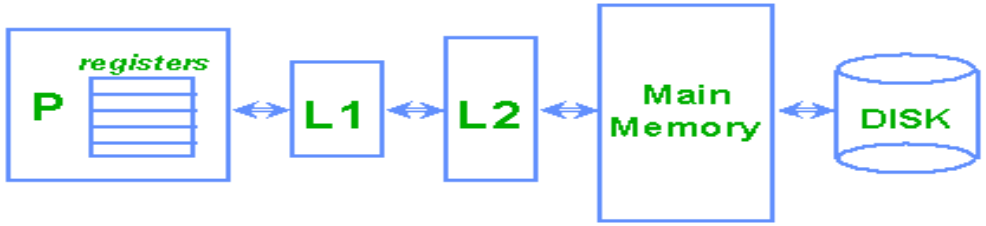
Atmintinės hierarchija ir charakteristikos



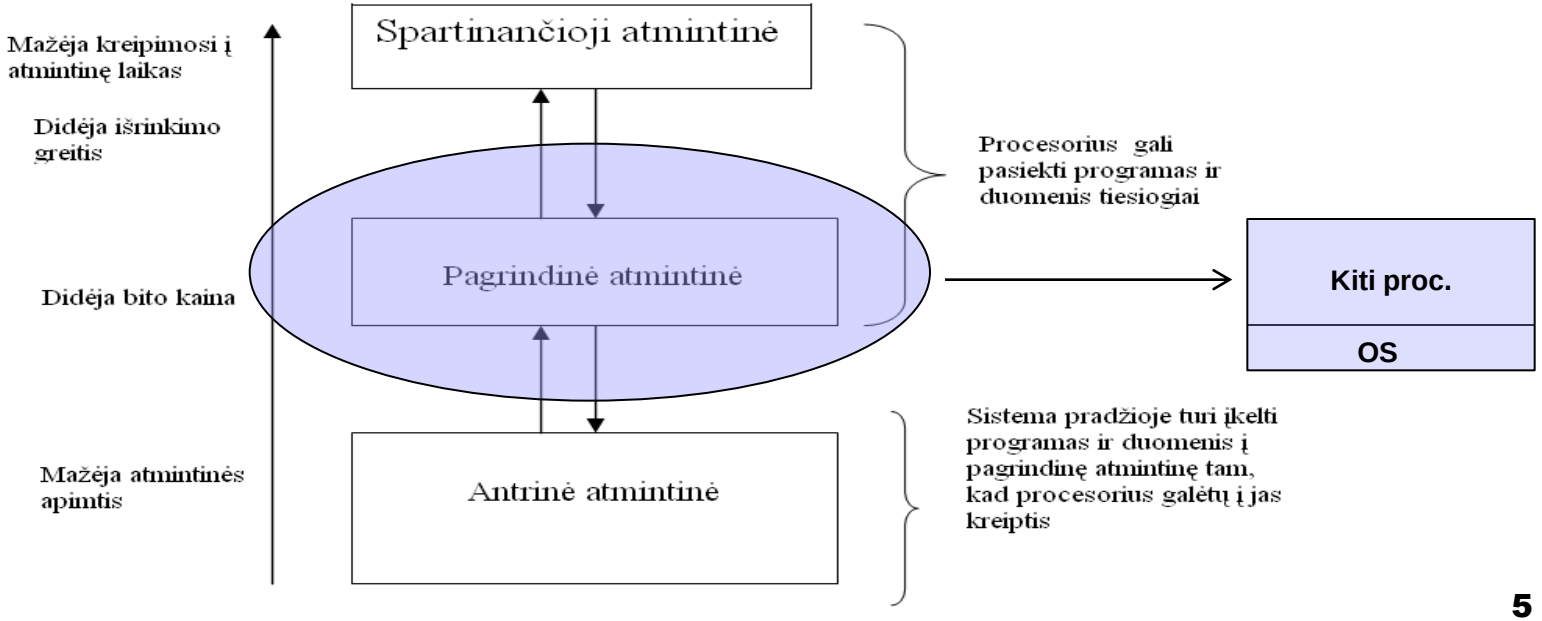
Access time	0.5 clk	1 clk	5 clks	10-50 clks	10^5 clks
Capacity	1KB	16KB	1MB	1GB	10GB
Block Size	8B	64B	128B	4-16 KB	



Atmintinės hierarchija ir charakteristikos



Access time	0.5 clk	1 clk	5 clks	10-50 clks	10^5 clks
Capacity	1KB	16KB	1MB	1GB	10GB
Block Size	8B	64B	128B	4-16 KB	



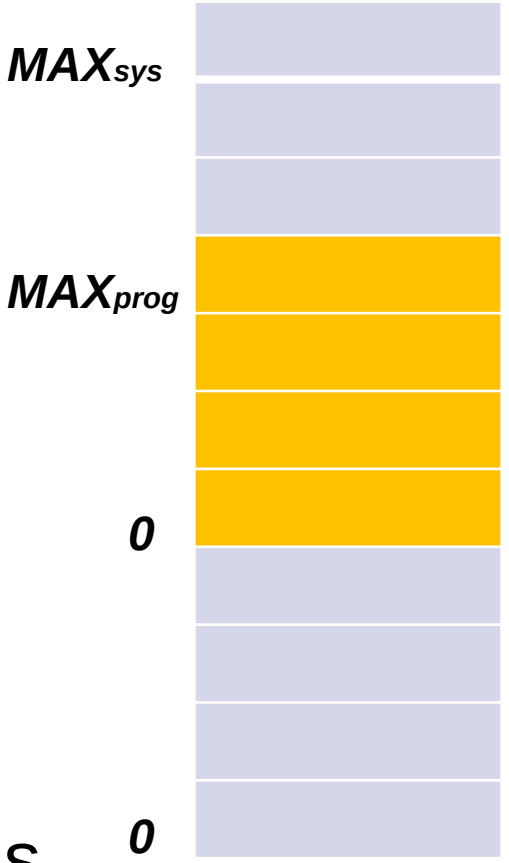
Reikalavimai keliami pagrindinės atminties valdymui

- Patalpinimo vietos pakeitimo galimybė
- Apsauga
- Dalinimasis
- Loginė organizacija
- Fizinė organizacija

Pagrindinės AV sąvokos (1)

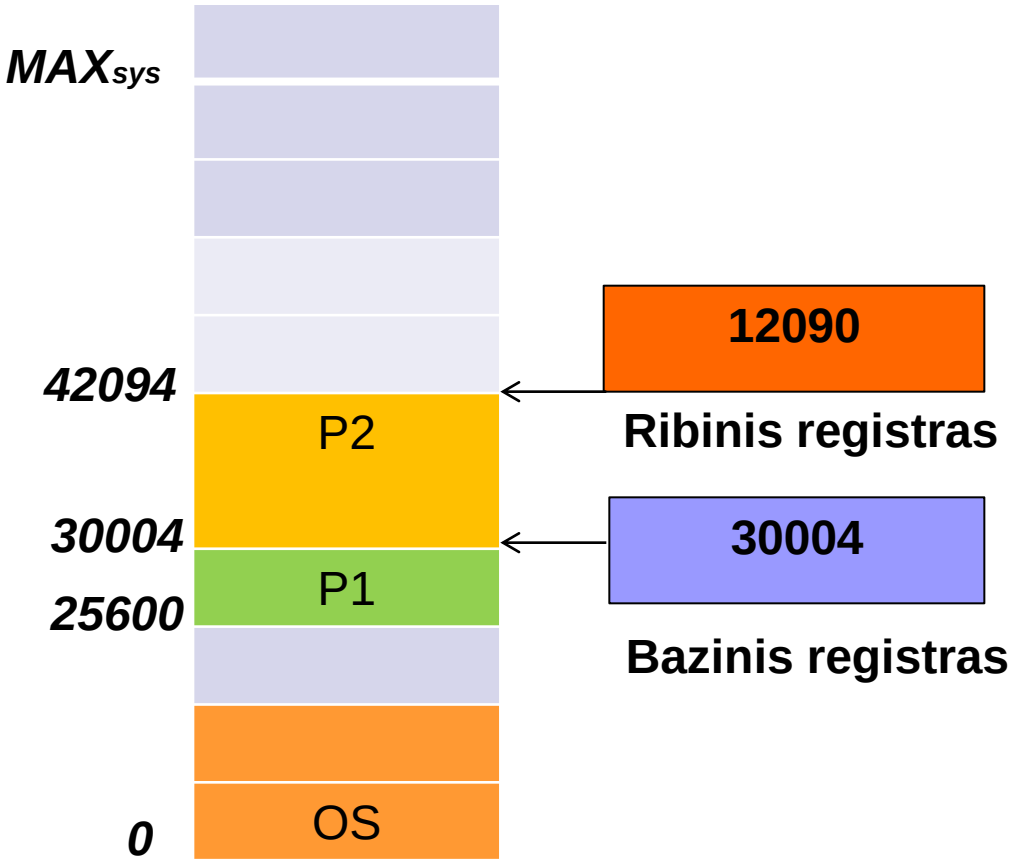
Adresų sritys

- Fizinė adresų sritis – techninės įrangos palaikoma adresų sritis
  - Nuo *0* iki MAX_{sys}
- Loginė/virtuali adresų sritis – procesui matoma atminties sritis
  - Nuo *0* iki MAX_{prog}
- Fizinis (absoliutusis) adresas, loginis (bazinis, reliatyvus) adresas



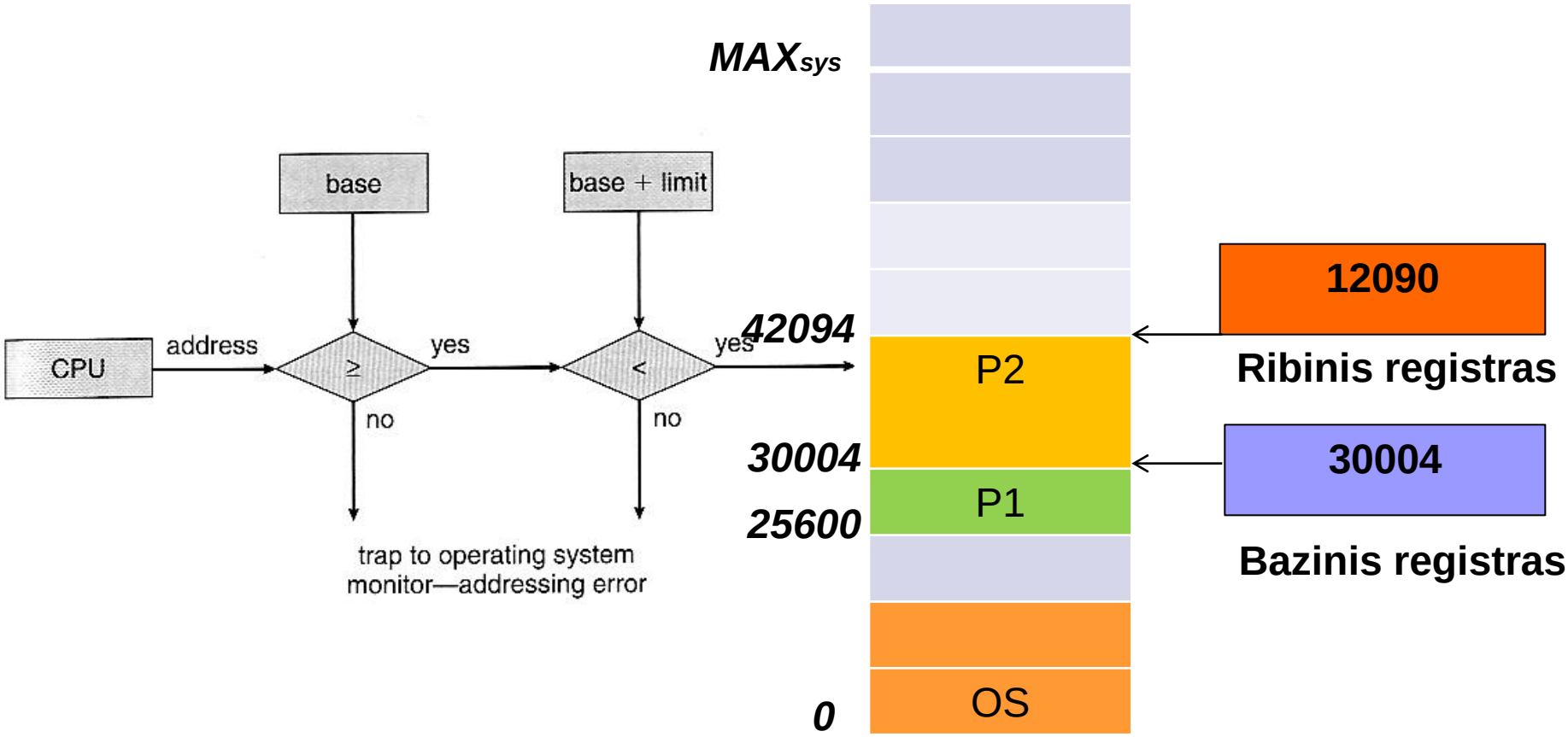
Pagrindinės AV sąvokos (2)

Loginių adresų sritį apibrėžiantys registrai



Pagrindinės AV sąvokos (2)

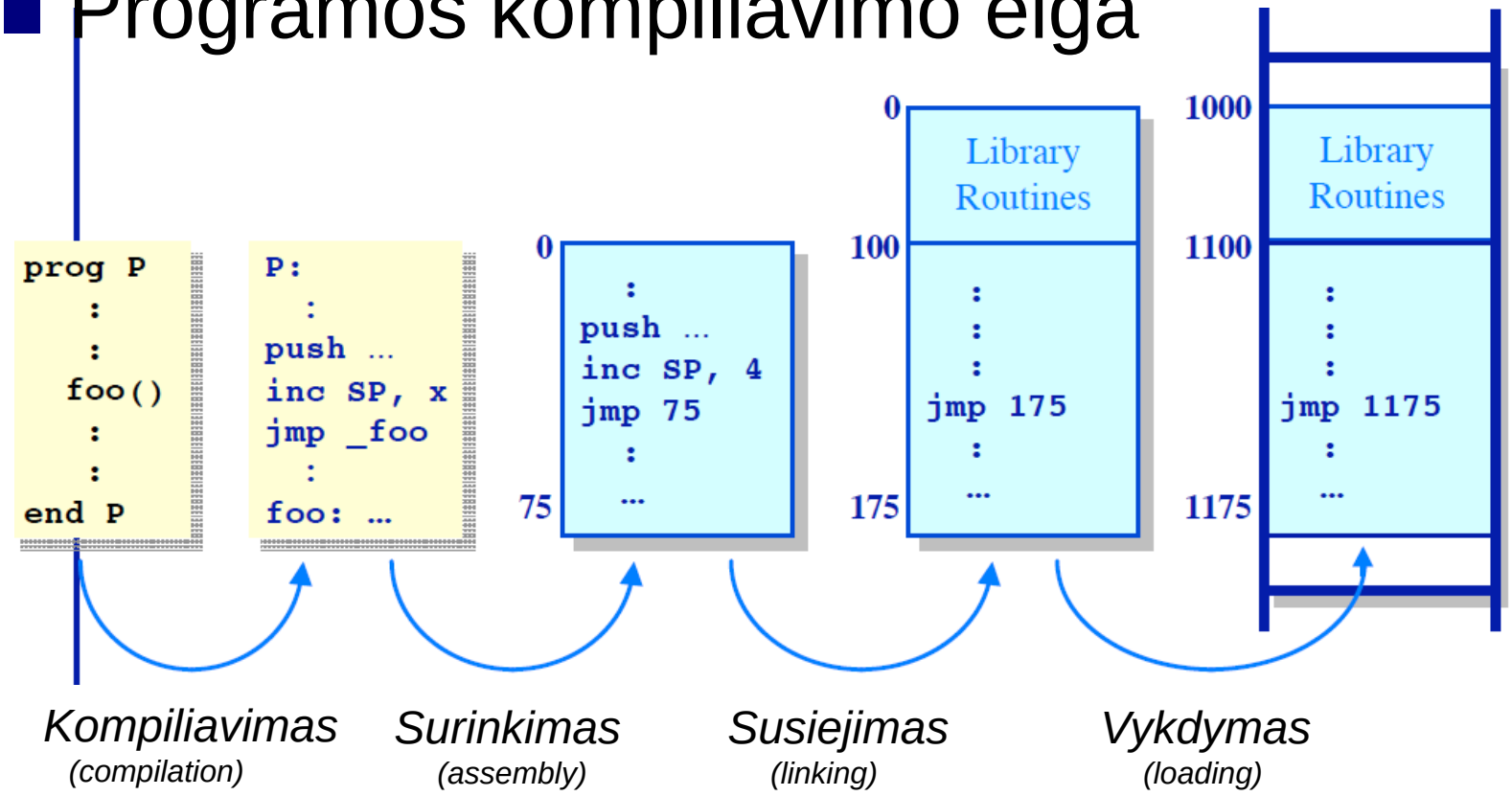
Loginių adresų sritį apibrėžiantys registrai



Pagrindinės AV sąvokos (3)

Loginių adresų formavimas

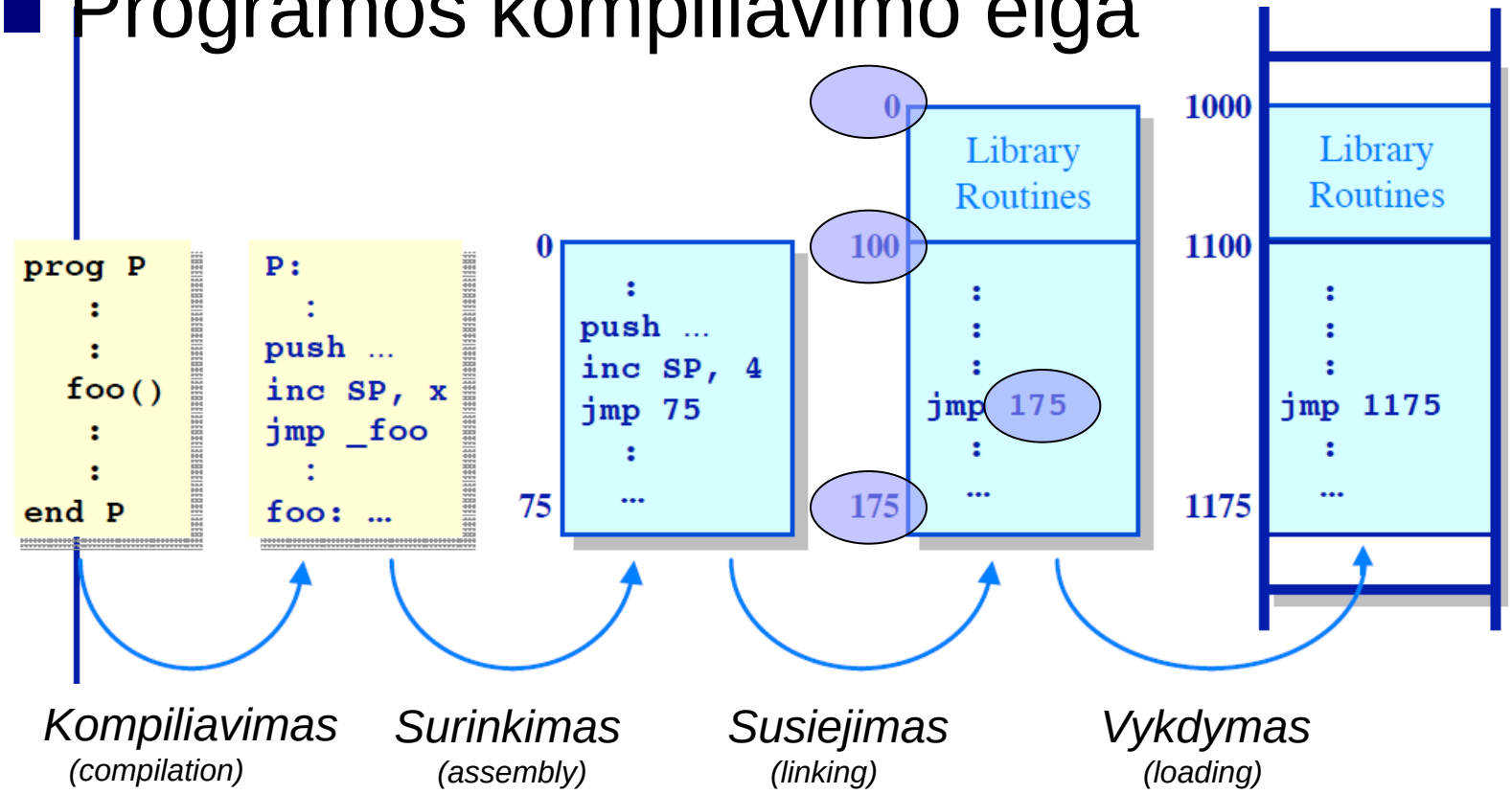
- Programos kompiliavimo eiga



Pagrindinės AV sąvokos (3)

Loginių adresų formavimas

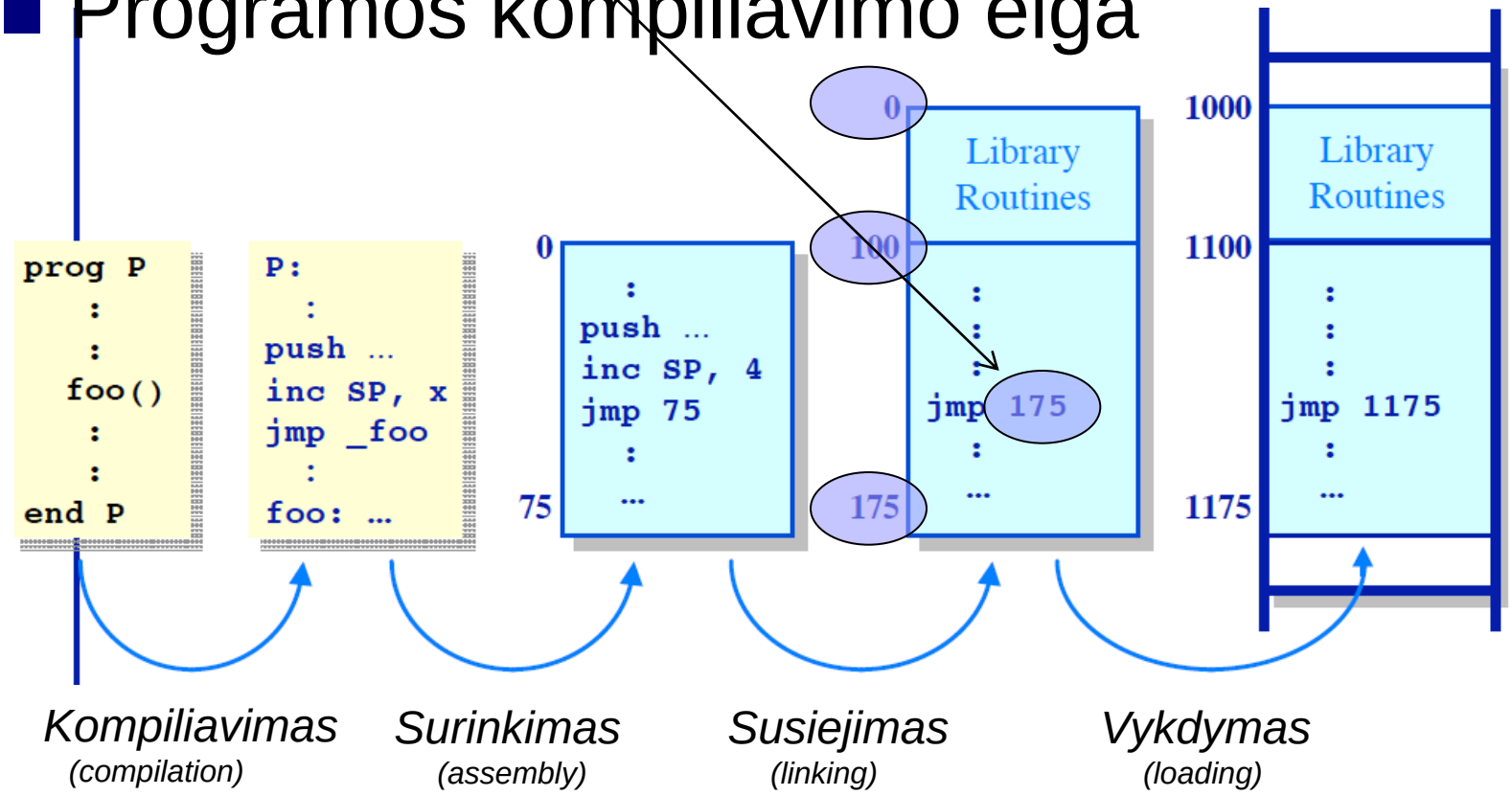
- Programos kompiliavimo eiga



Pagrindinės AV sąvokos (3)

Reliatyvus adresas – išreiškiamas bazinio adreso atžvilgiu

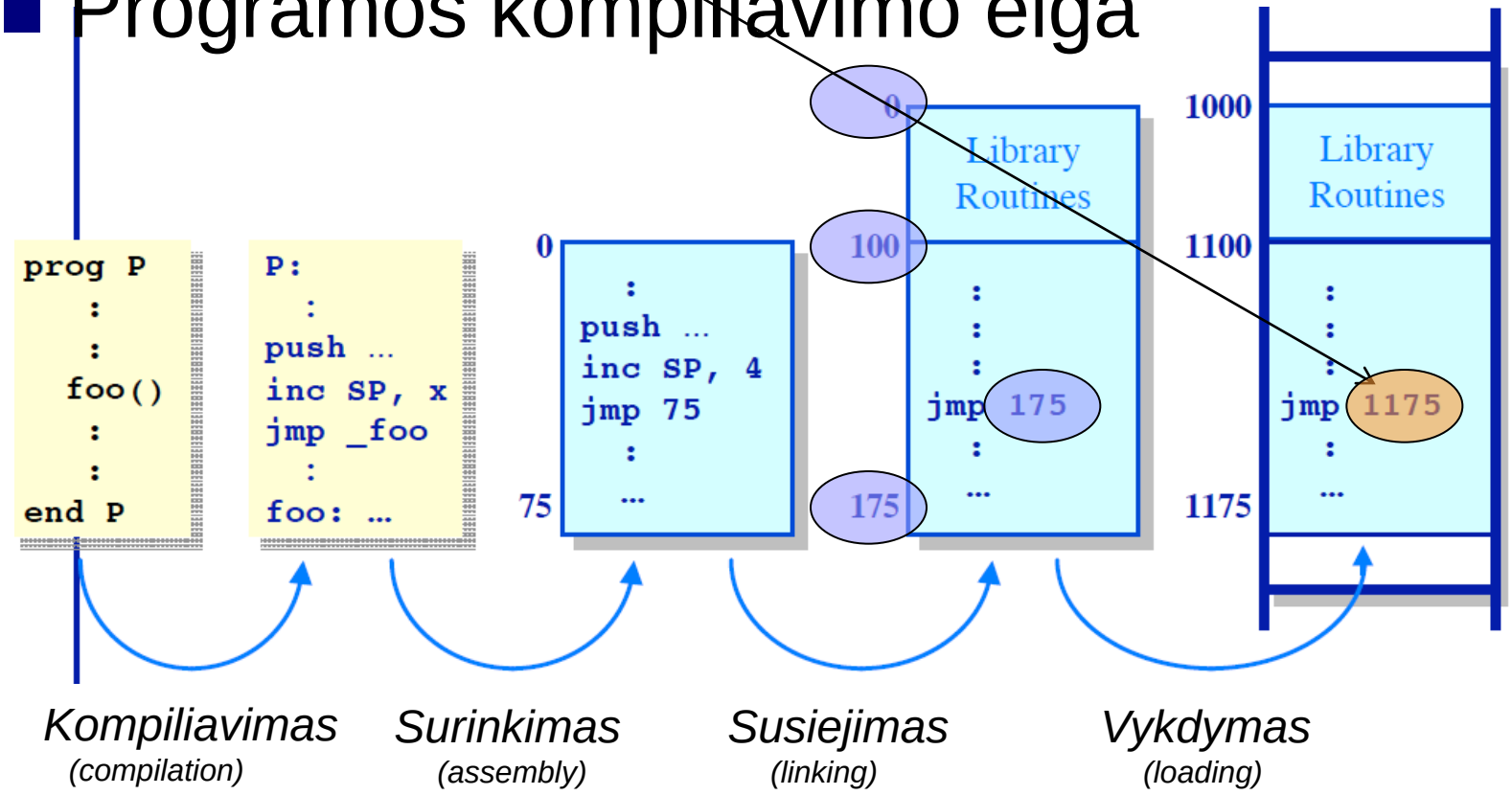
- Programos kompiliavimo eiga



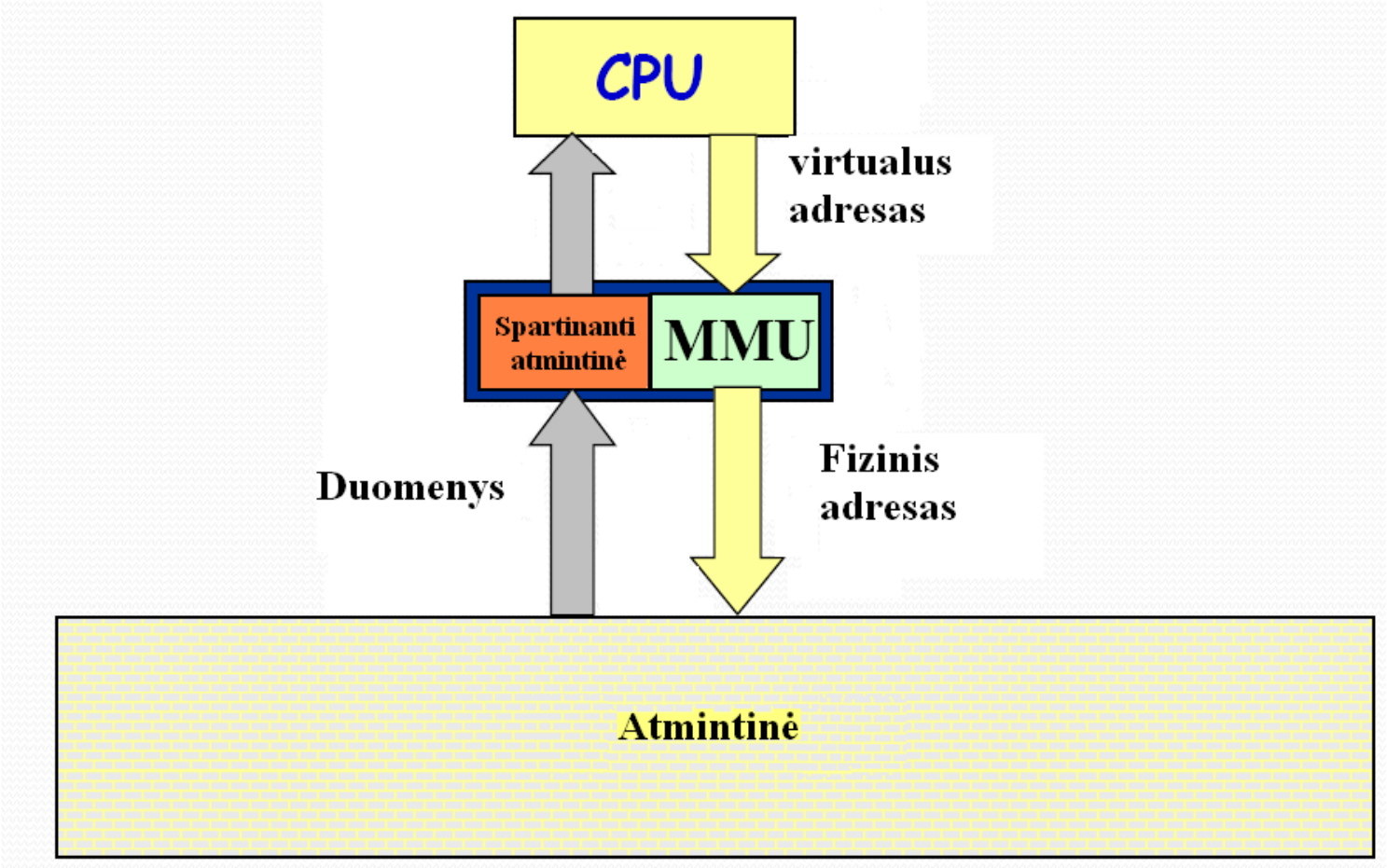
Pagrindinės AV sąvokos (3)

Fiziniai adresai – priskiriami programos vykdymo metu

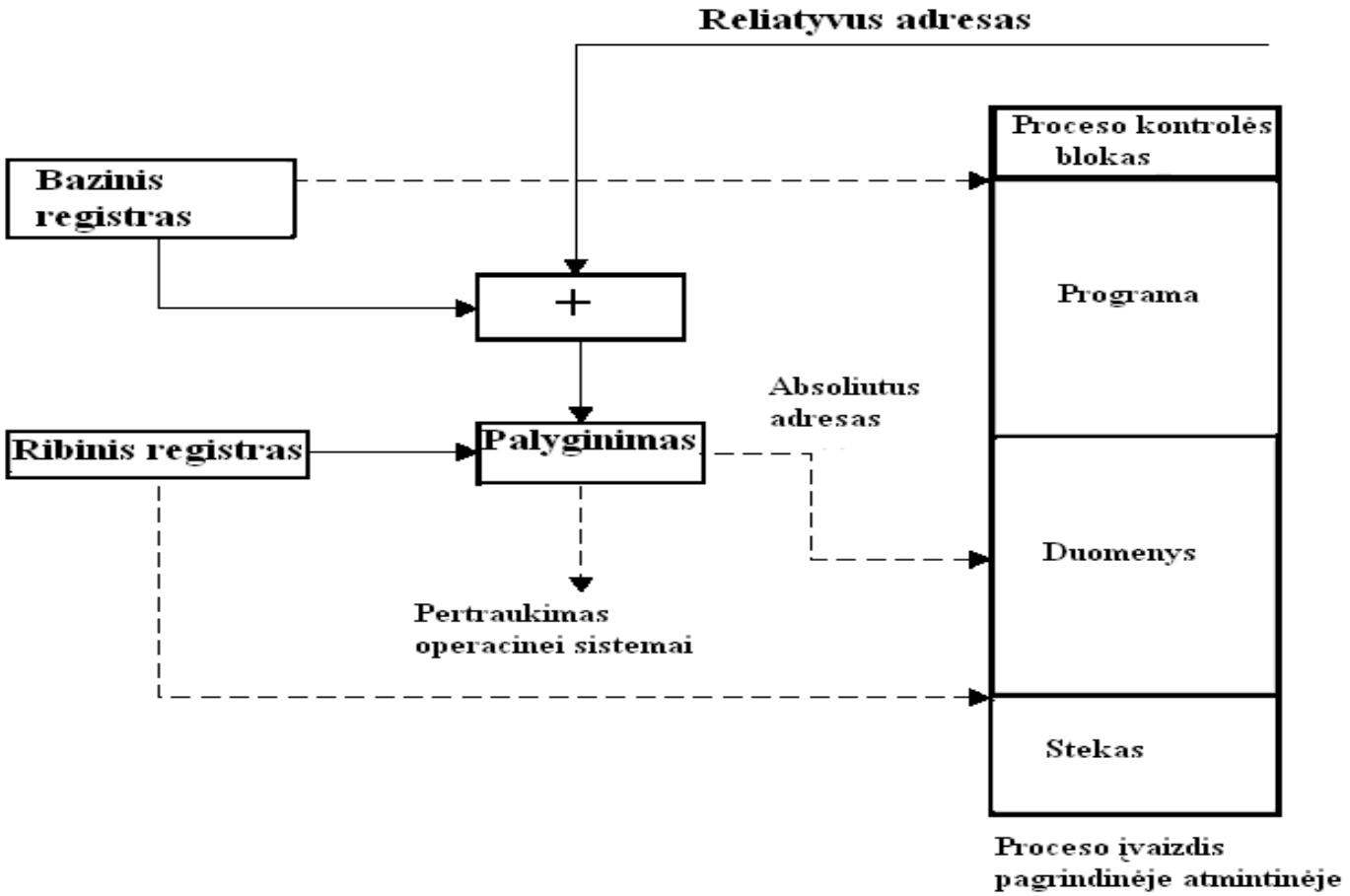
- Programos kompiliavimo eiga



Loginio adreso transliavimas į fizinį



Reliatyvaus adreso transliacija į fizinį adresą



Rezidentinė arba darbinė proceso dalis

- Į pagrindinę atmintinę įkelta proceso dalis dar yra vadinama **rezidentine** arba **darbine** dalimi.
- Kodėl gi proceso vykdymui pakanka mažesnės srities nei viso proceso dydis?
  - Tai surišta su *lokališkumo* principu, kuris pasireiškia tuo:
    - dažniausiai didelę proceso vykdymo laiko dalį procesas vykdo nedidelę komandų aibę (vyksta ciklas)
    - naudoja greta esančius duomenis,
    - dauguma skaičiavimų yra vykdomi nuosekliai.
  - Todėl bet kuriuo momentu procesui pakanka nedidelės rezidentinės srities pagrindinėje atmintinėje.

1-4 Užduotys

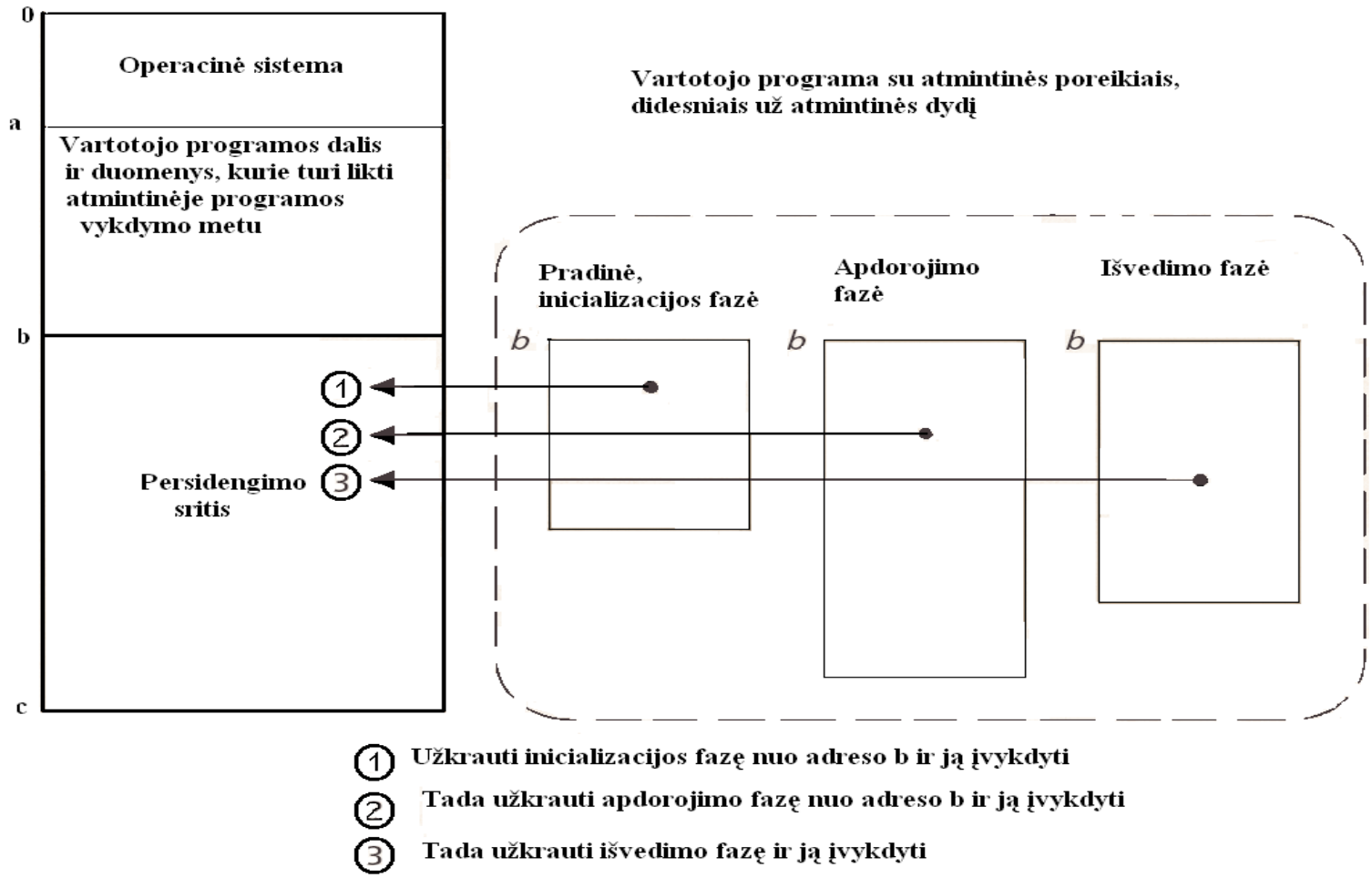
Atmintinės valdymo schemos

- Sprendžia problemas, kurių atsiranda keliant procesus iš antrinės atmintinės į pagrindinę:
  - kurią proceso dalį įkelti ir kada?
  - kur patalpinti įkeliamą procesą ar jo dalį?
  - kuriuos duomenis iškelti, kad būtų daugiau vietos kitiems, įkeliamiems procesams?

Vietos procesui skyrimas atmintinėje

- **Nuoseklių** (ištisinių adresų) zonos skyrimas
  - procesas egzistuotų kaip vientisas, nuoseklių adresų erdvėje esantis blokas.
  - tai labai paprastas atmintinės dalinimo būdas,
  - atsiranda problemos:
    - tinkamo dydžio laisvo bloko atradimas
    - netinkamas atmintinės panaudojimas.
- **Neištisinės** srities skyrimas
  - procesas, jo duomenys skaldomi tam tikro dydžio, atskirose atmintinės vietose talpinamais gabalais (puslapiais, segmentais).
  - lengviau atrasti tinkamas proceso patalpinimui vietas atmintinėje,
  - leidžia padidinti procesų, vienu metu esančių pagrindinėje atmintinėje kiekį,
  - realizacija yra sudėtingesnė.

Ištisinė sritis ir *perdengimo* mechanizmas



Daugiaprogramės, daugiavartotojiškos sistemos

- Prireikė algoritmų, kurie leistų paskirstyti atmintinę kelioms programoms (keliems procesams).
- Paprasti algoritmai, naudojami skirstant pagrindinę atmintinę yra šie:
  - Fiksuoto dydžio skyriai
  - Dinaminis skyrių formavimas
  - Paprasta segmentacija
  - Paprastas puslapiavimas

Daugiaprogramės, daugiavartotojiškos sistemos

- Prireikė algoritmų, kurie leistų paskirstyti atmintinę kelioms programoms (keliems procesams).
- Paprasti algoritmai, naudojami skirstant pagrindinę atmintinę yra šie:
  - Fiksuoto dydžio skyriai
  - Dinaminis skyrių formavimas
  - Paprasta segmentacija
  - Paprastas puslapiavimas

Fiksuoti skyriai

- Pagrindinė atmintis yra sudaloma į eilę nepersidengiančių skyrių (dalių). Šios dalys gali būti tiek vienodo tiek skirtingo dydžio.
- Procesas, kurio dydis yra mažesnis arba lygus skyriaus dydžiui, gali būti patalpinamas į šį skyrių.
- Procesorius gali greitai persijungti tarp procesų.
- Naudojami keli ribiniai registrai apsaugai nuo to, kad procesai negadintų vienas kito duomenų ar programos, kreipdamiesi į ne jam skirtą atmintinės bloką – tokie kreipiniai neleidžiami.
- Jei visi skyriai yra užimti, operacinė sistema gali iškelti (swap) procesą iš jo užimamo skyriaus.

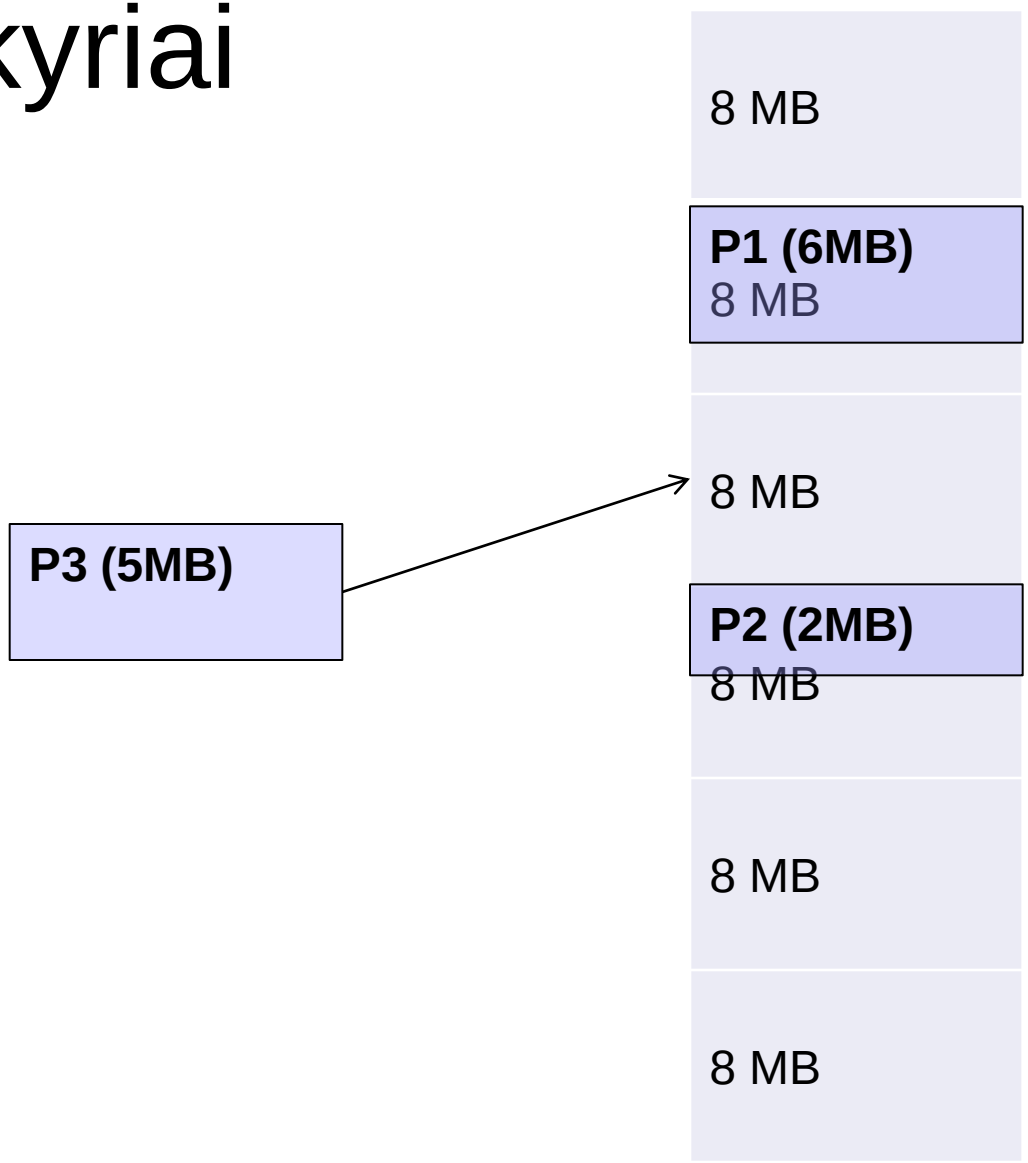
Vienodo dydžio skyriai

Opeacinė sistema 8M	8M	8M	8M	8M

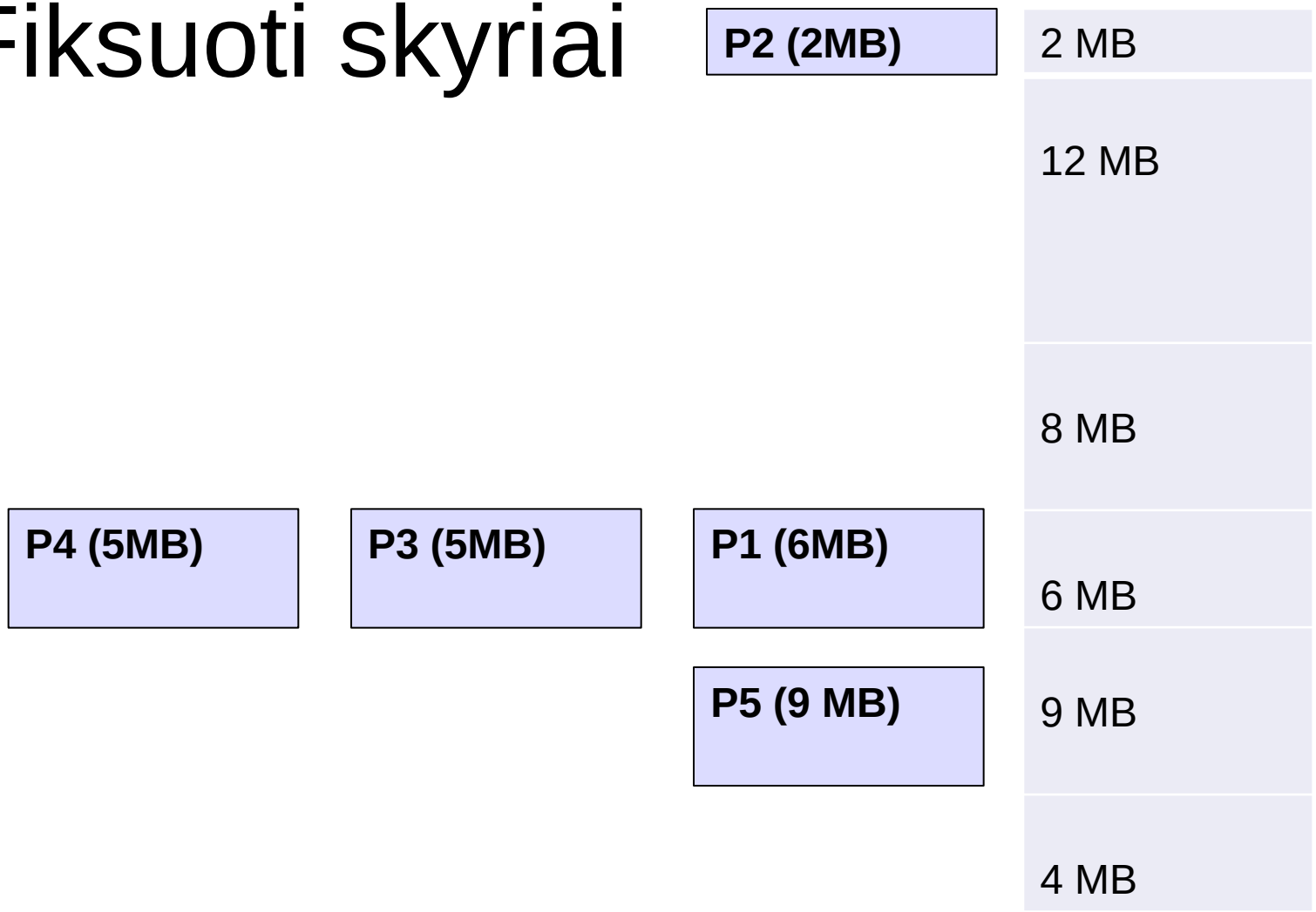
Skirtingo dydžio skyriai

Opeacinė sistema 8M	2M	4M	6M	8M	12M

Fiksuoti skyriai



Fiksuoti skyriai



Fragmentacija naudojant fiksuotus skyrius

- Atsiranda vidinės fragmentacijos problema:
  - nes nežiūrint kokia maža programa būtų – jai skiriamas visas skyrius ir jame gali būti daug nenaudojamos vietos.
  - Skirtingo fiksuoto ilgio skyriai kiek sumažina šią problemą, tačiau problema išlieka.

Skirtingo dydžio skyriai

Opeacinė sistema 8M	2M	4M	6M	8M	12M

Procesai	Poreikiai atmintinei
A	64 KB
B	2,48 MB
C	8,12 MB

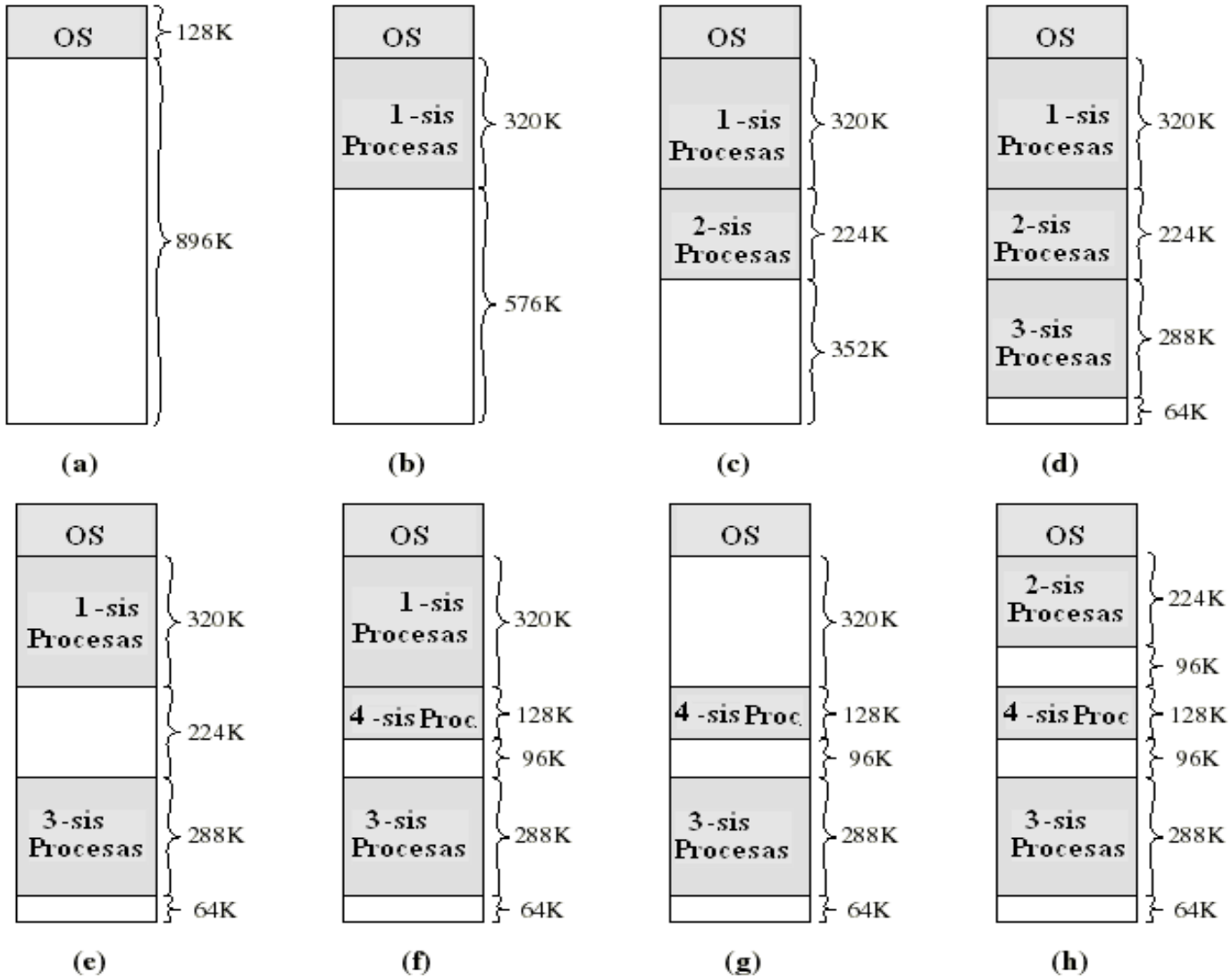
Pagrindinės atmintinės paskirstymo schemos

- Fiksuoto dydžio skyriai
- Dinaminis skyrių formavimas
- Paprasta segmentacija
- Paprastas puslapiavimas

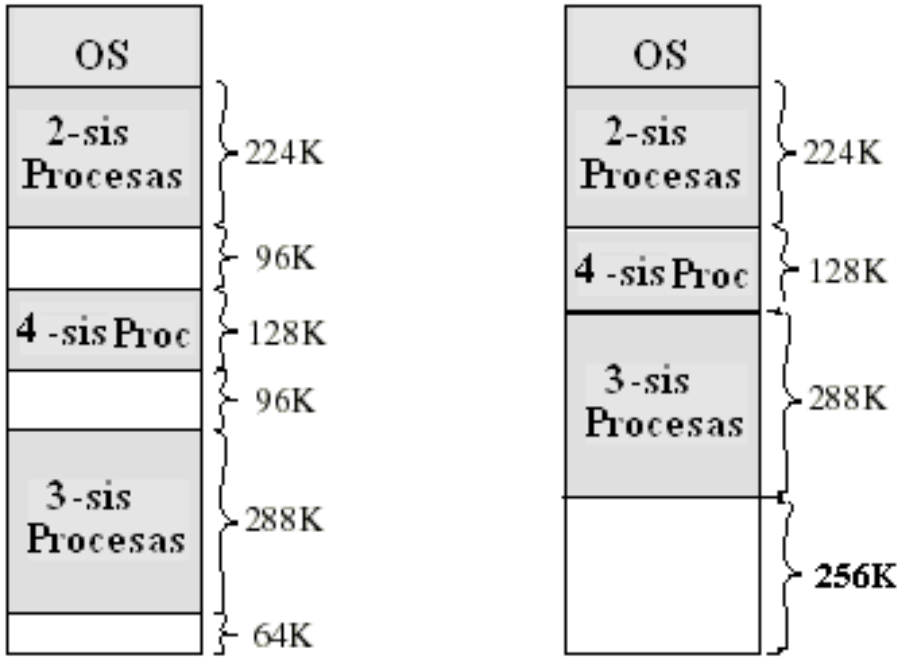
Dinaminis skyrių formavimas

- Taikant šį principą skyrių *kiekis*, jų *dydis* yra kintami.
- Kiekvienam procesui jį talpinant pagrindinėje atmintinėje yra išskiriamas *tokio dydžio skyrius, kokio jis prašo*.

Dinaminis skyrių formavimas



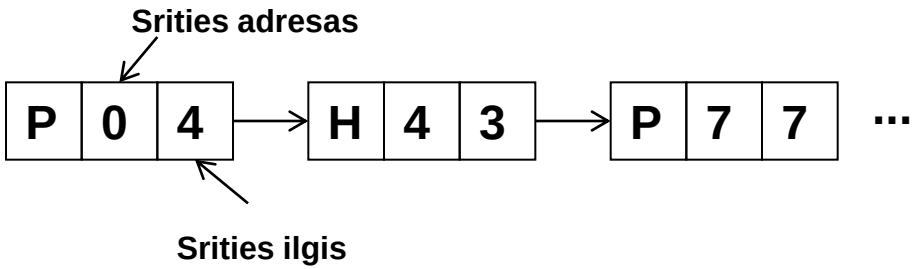
Suspaudimas



Užimtų ir laisvų sričių saugojimas

- Dvejetainiai žemėlapiai;
  - Didelę reikšmę turi srities dydis;
  - Patogu naudotis
- Surišti sąrašai

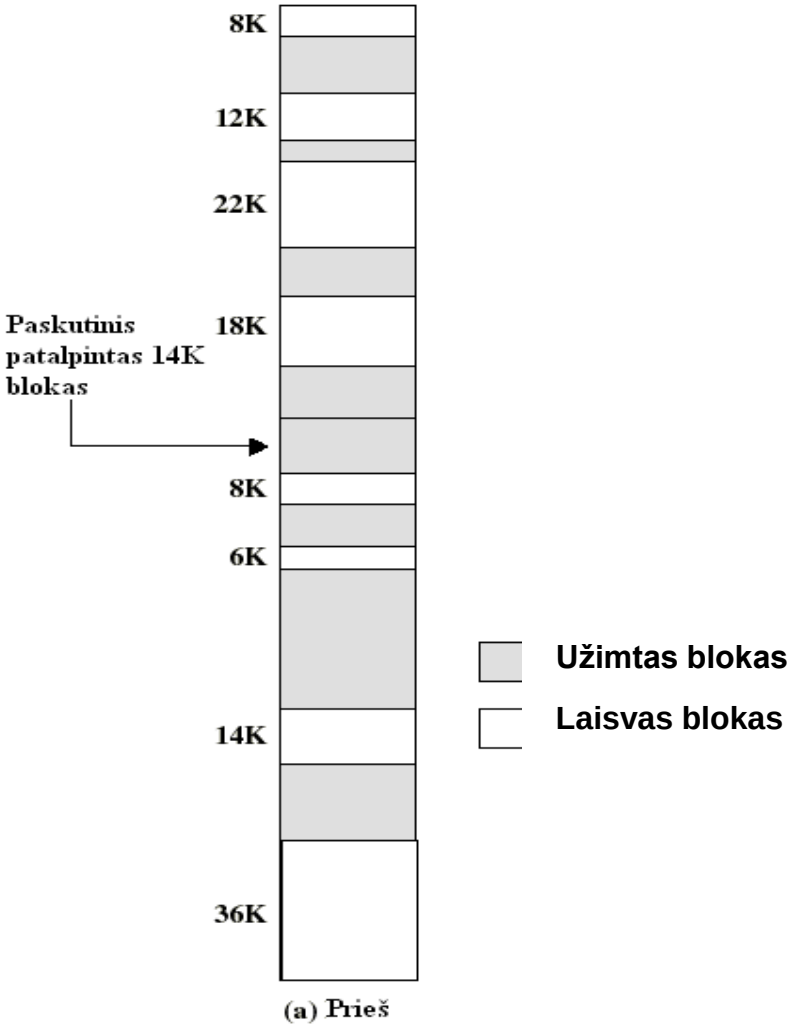
1	**1**	**1**	**1**	**0**	**0**	**0**	**1**
1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1
1	0	1	1	1	0	0	



Talpinimo algoritmai dinaminių skyrių atveju

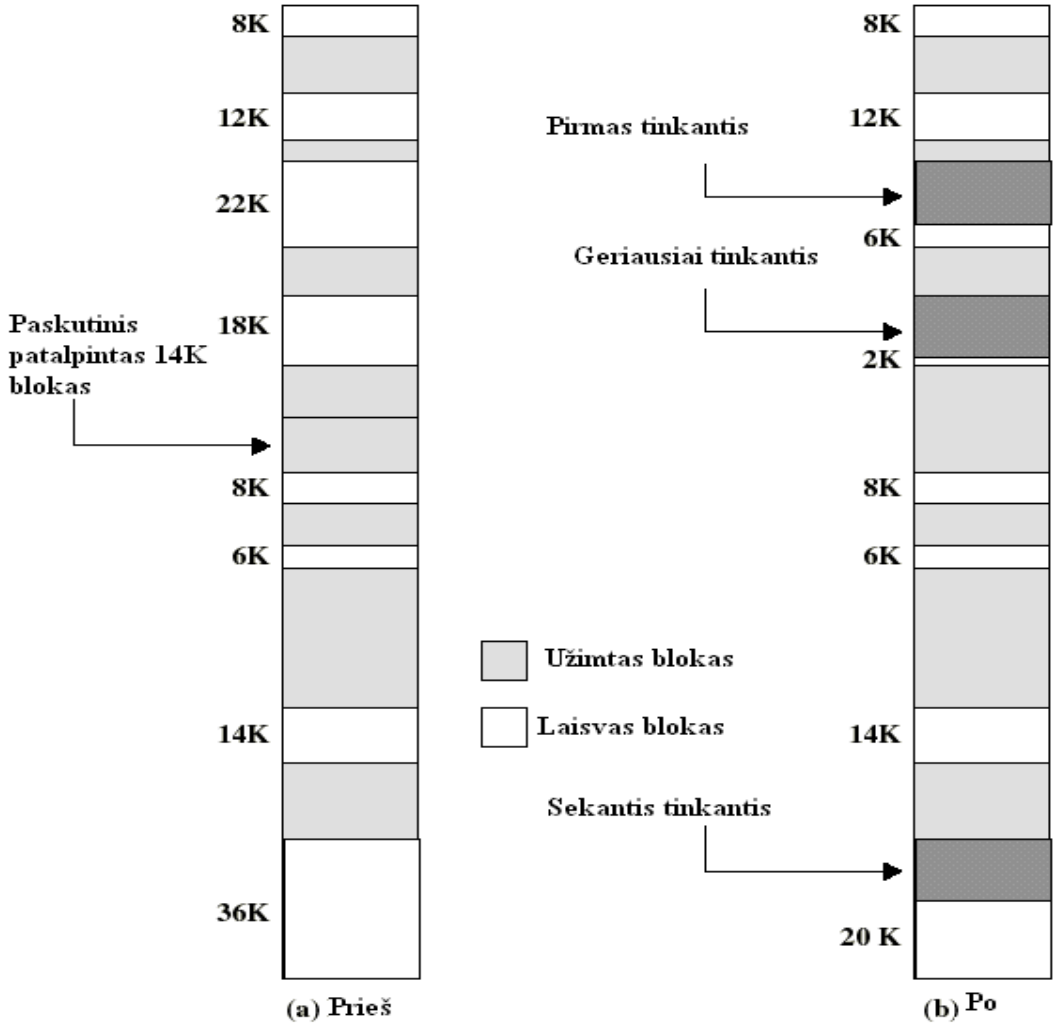
- Paskirtis - nuspręsti, kurį laisvą atmintinės skyrių („skylę“) paskirti procesui, jį talpinant pagrindinėje atmintinėje.
- Galimi ir naudojami šie algoritmai:
  - *Geriausiai tinkančio*
  - *Pirmo tinkamo*
  - *Sekančio tinkančio*.
  - *Blogiausiai tinkančio*

Talpinimo algoritmų pavyzdys, talpinant 16K procesą atmintinėje



(a) Prieš

Talpinimo algoritmų pavyzdys, talpinant 16K procesą atmintinėje



Talpinimo algoritmų savybės

- Taikant „*sekantis tinkamas*“ algoritmą dažnai naujai talpinamam procesui yra priskiriamas didžiausias blokas, esantis pagrindinės atmintinės pabaigoje.
- Taikant *pirmo tinkamo* paiešką, skylės radimas sukasi apie atmintinės pradžią, jį taikant gaunama mažesnė fragmentacija nei „sekančio tinkamo“ atveju.
- „*Geriausiai tinkančio*“ paieška yra susijusi su mažiausio neužimto bloko suradimu: likęs laisvas fragmentas bus gaunamas mažiausias.
- Kartais yra naudojamas algoritmas, kuris proceso patalpinimui ieško blogiausiai tinkančios savo dydžiu „skylės“. Randamas didžiausias savo apimtimi blokas, į kurį patalpinus procesą jame lieka didžiausia neišnaudota erdvė. Yra tikimasi, kad į šią neišnaudotą erdvę vėliau bus galima patalpinti kitą procesą.

Fragmentacija naudojant dinaminius skyrius

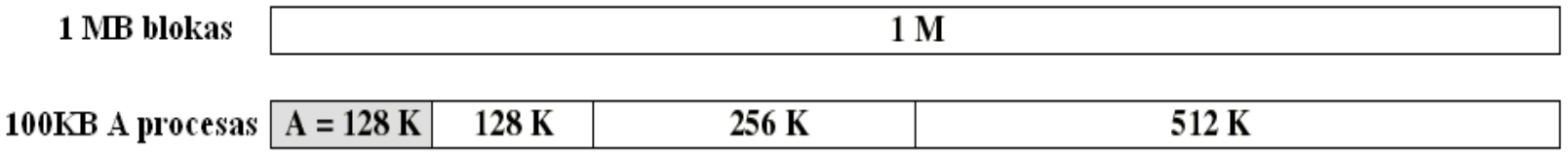
- Pagrindinėje atmintinėje labai greitai gaunamos skylės, kurios yra per mažos bet kokio proceso patalpinimui, ir reikia atlikti suspaudimus (išorinė fragmentacija).

Bičiuliška sistema (Buddy System)

- Bičiuliška sistema, tai algoritmas, kuriuo bandoma apeiti tiek fiksuotų, tiek dinaminių skyrių problemas.
- Modifikuota šio algoritmo versija yra naudojama UNIX SVR4 .
- Atmintinės blokai, kurie yra išskiriami procesams yra **2^{K}** dydžio

Algoritmo žingsniai

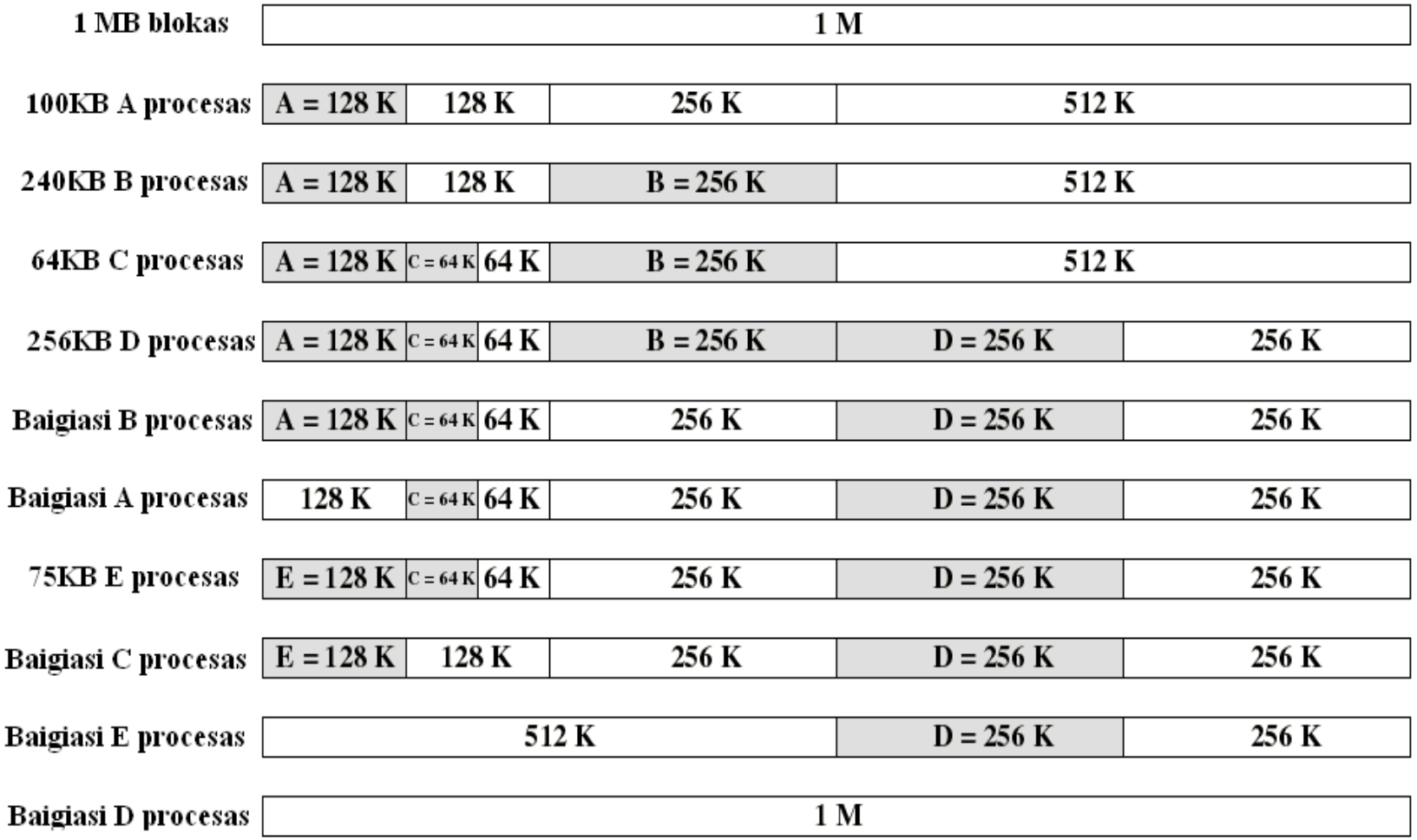
- Pradžioje visa atmintinė yra laisva, taigi pradedama turint **2^{U}** dydžio bloką. Tarkime, atsiranda pareikalavimas patalpinti **S** dydžio procesą.
  - Jei **2^{U-1} < S <= 2^{U}**, tai yra išskiriamas visas blokas **2^{U}**.
  - Priešingu atveju blokas sudalomas į dvi vienodo dydžio **2^{U-1}** dalis (bičiulius).
    - Jei **2^{U-2} < S <= 2^{U-1}**, tai procesui išskiriama viena iš dalių (vienas bičiulis), o jei ne, tai viena iš dalių vėl yra daloma į dvi dalis.
    - Šis procesas yra kartojamas tol, kol gaunamas mažiausias blokas, kuris yra lygus arba didesnis nei **S**.
    - Pavyzdys: Turim 1MB= 2^{10} KB=1024KB . Procesas prašo 100 KB
    - 2^9=512<100< 2^{10} =1024 – netinka
    - 2^8=256<100<2^9=512 –netinka
    - 2^7=128<100<2^8=256 –netinka
    - 2^6= 64<100<2^7=128- tinka, skiriam 128 KB



Bičiuliška sistema

- Jei du bičiuliai tampa laisvais , bičiuliai yra apjungiami.
- Operacinė sistema palaiko keletą sąrašų apie esančias skyles.
  - **i**-tas sąrašas apima skyles, kurių dydis yra **2^{i}**.
  - Kai tik pora bičiulių atsiranda **i**-tame sąraše, jie yra išmetami iš šio sąrašo ir apjungiami į vieną bendrą skylę **(i+1)** sąraše.
- Atsiradus naujai užklausai, t.y. norint patalpinti **k** dydžio procesą, tokį:
  - kuriam galioja: **2^{i-1} < k <= 2^{i}**
  - yra patikrinamas **i**-tas sąrašas. Jei šis sąrašas yra tuščias, tikrinamas **(i+1)** sąrašas.
  - Jame radus skylę ji bus sudaloma į du bičiulius, viena iš dalių bus priskirta procesui, o kita įtraukta į **i**-tą sąrašą.

Bičiulių sistemos taikymo pavyzdys



Fragmentacija naudojant Bičiulių sistemą

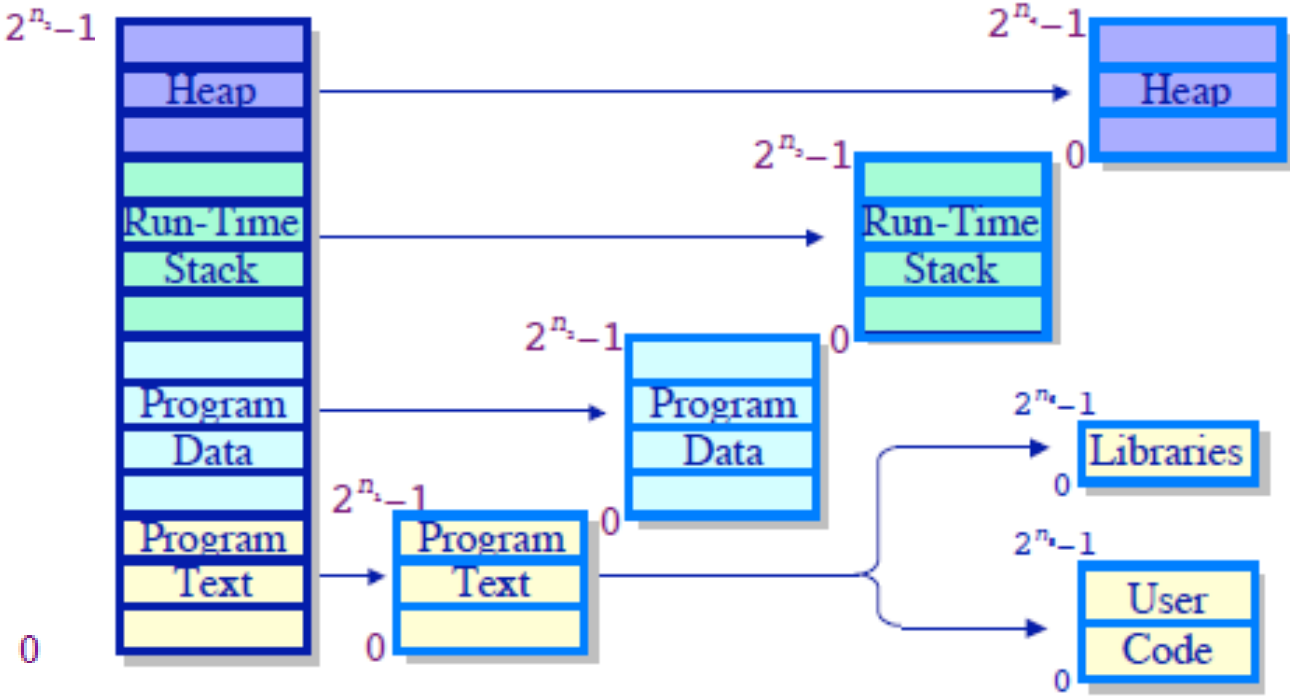
- ??? (klausimas jums)???

Pagrindinės atmintinės paskirstymo strategijos

- Fiksuoto dydžio skyriai
- Dinaminis skyrių formavimas
- Paprasta segmentacija
- Paprastas puslapiavimas

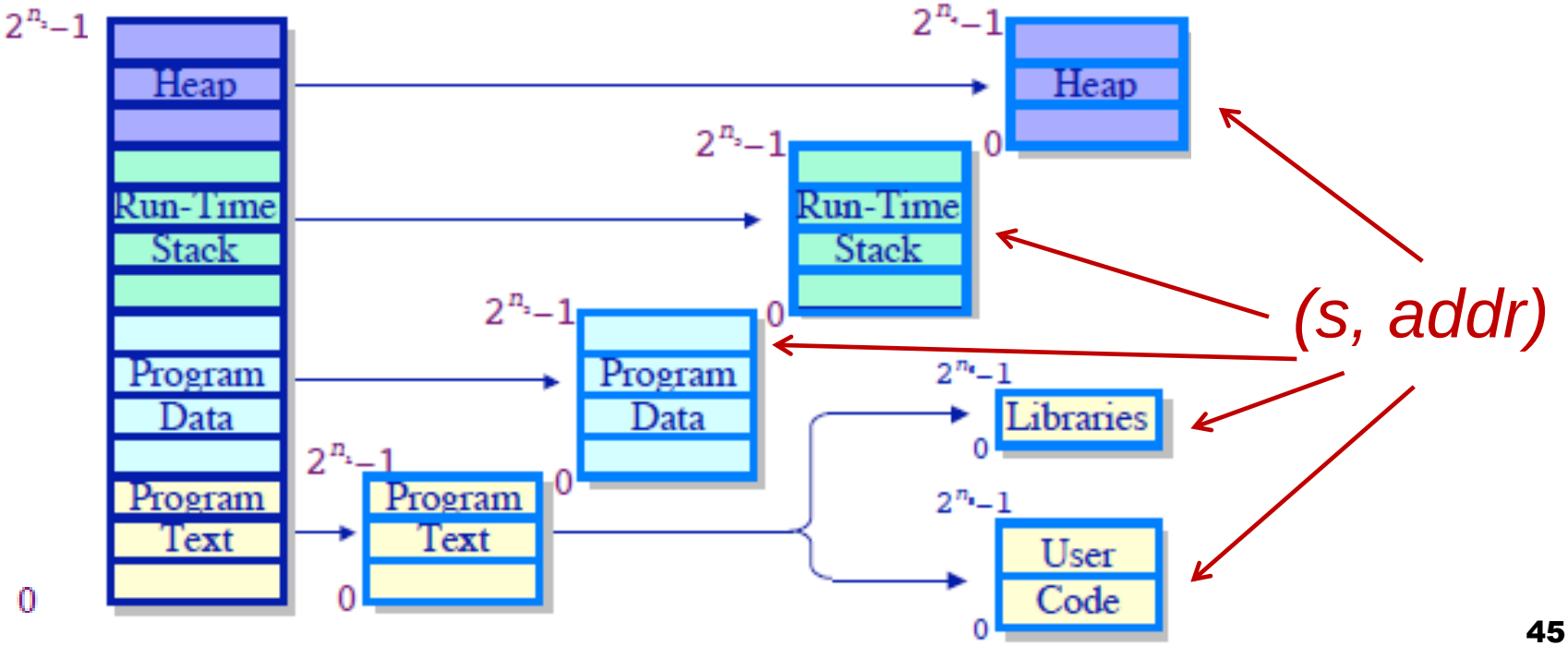
Segmentacija

- Procesas išskaidomas loginiais segmentais
- Kiekvienas segmentas apibrėžiamas adresu, nurodančiu segmento pradžią bei ribiniu poslinkiu.



Segmentacija

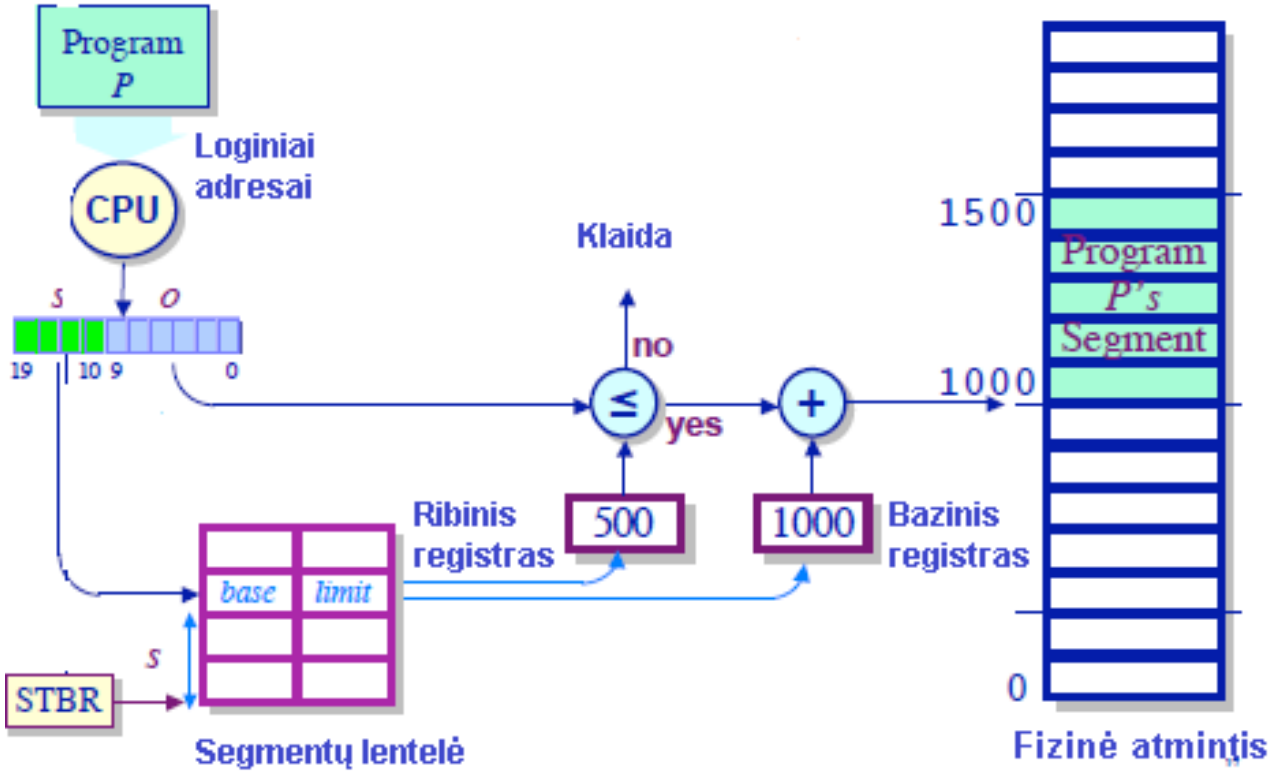
- Procesas išskaidomas loginiais segmentais
- Kiekvienas segmentas apibrėžiamas adresu, nurodančiu segmento pradžią bei ribiniu poslinkiu.



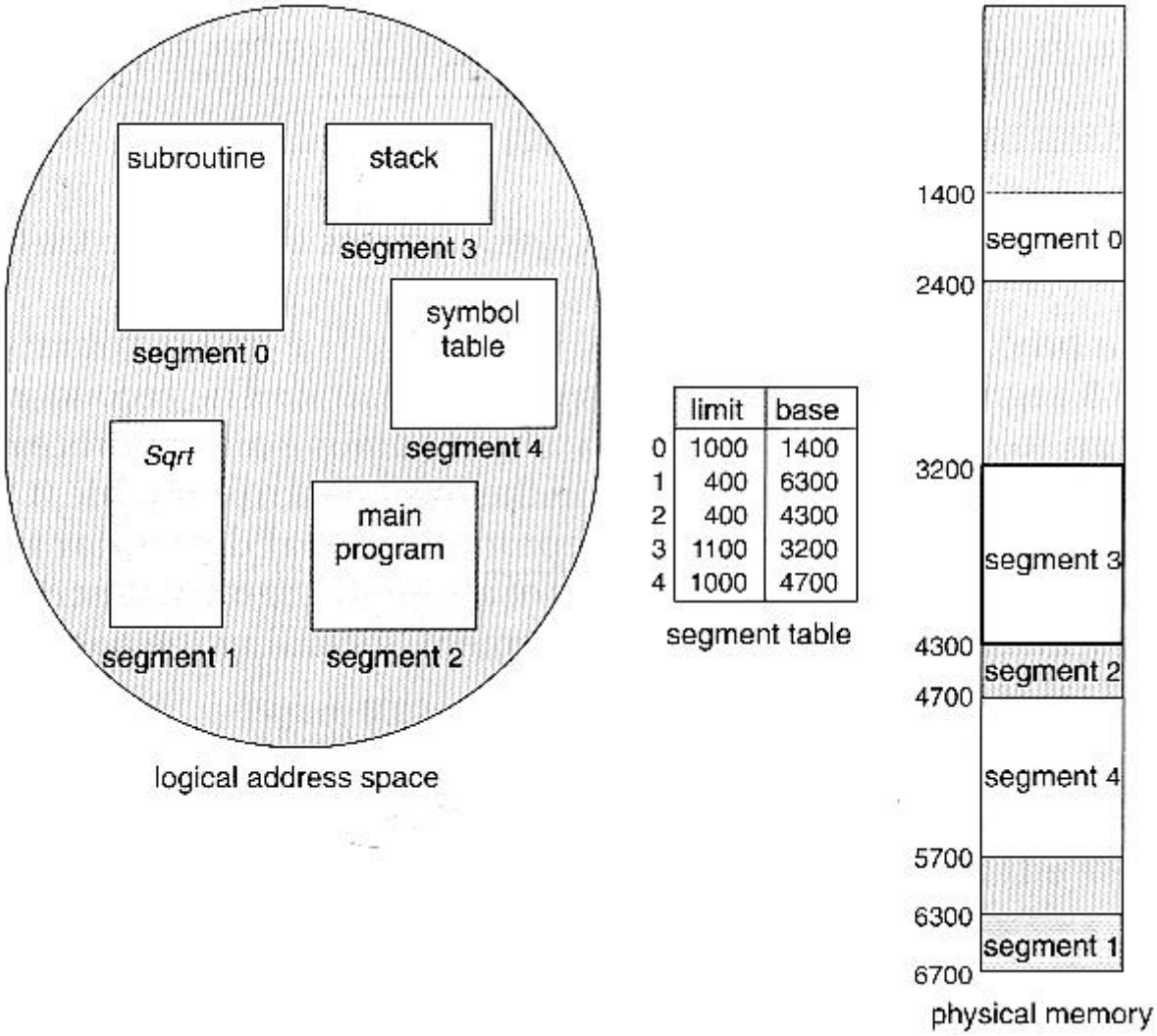
Segmentacijos realizaciniai aspektai

Virtualaus adreso transliacija į fizinį

- Segmentų lentelė su segmentų pradžiomis bei poslinkiais



Segmentacijos pavyzdys



Fragmentacija naudojant segmentus

- ???

Fragmentacija naudojant segmentus

- Išorinė fragmentacija egzistuoja.
- Gali būti sprendžiama skaidant segmentą jo viduje (puslapiuoti segmentai)

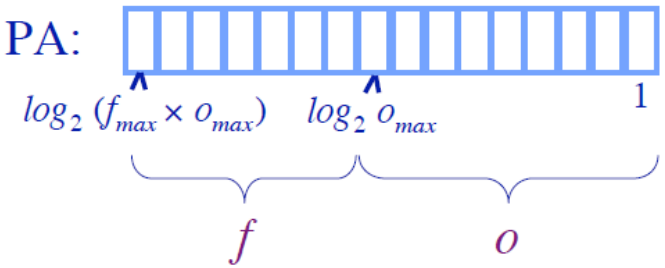
Pagrindinės atmintinės paskirstymo strategijos

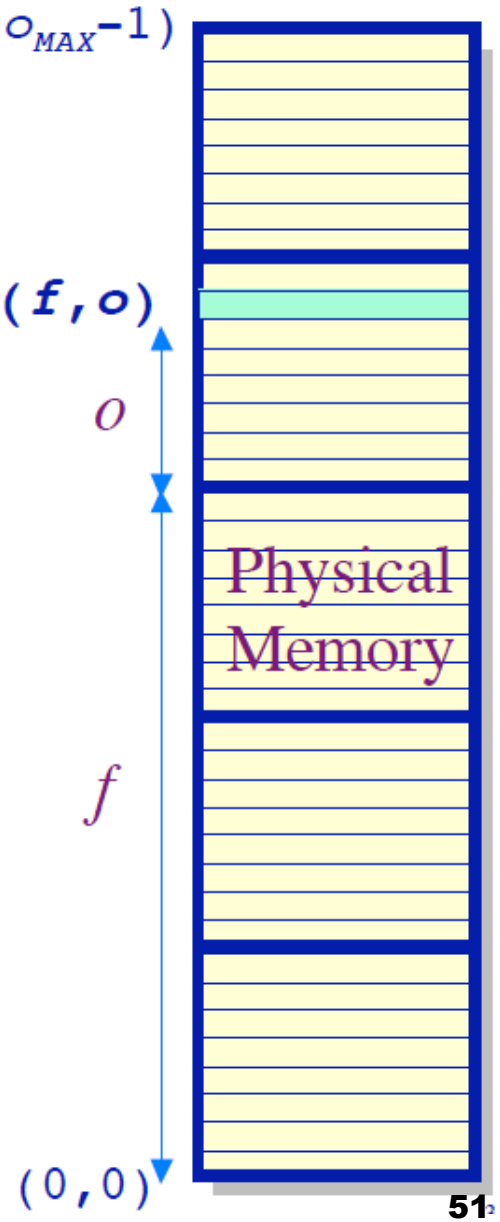
- Fiksuoto dydžio skyriai
- Dinaminis skyrių formavimas
- Paprasta segmentacija
- Paprastas puslapiavimas

Puslapiavimas (1)

- Fizinė atmintis yra sudaloma į fiksuoto, puslapio dydžio blokus, vadinamus rėmais (frames).

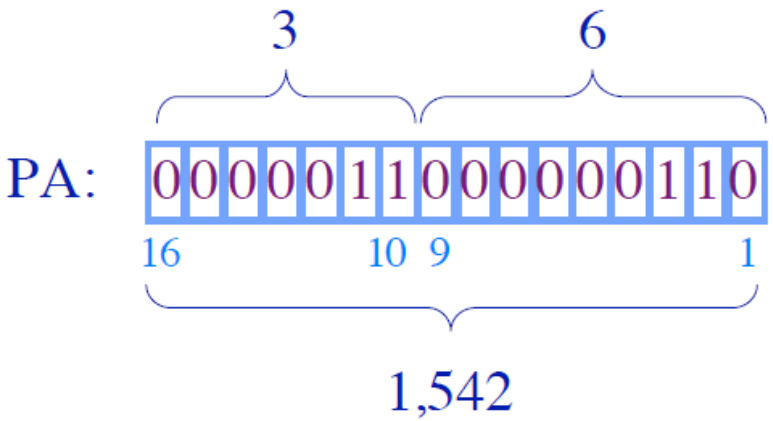
- Atminties adresas yra rinkinys (*f*, *o*):
  - *f* – rėmo numeris (f_{max} rėmų)
  - *o* –poslinkis nuo rėmo pradžios (o_{max} baitai/rėmai)
  - Fizinis adresas= $o_{\max} \times f + o$

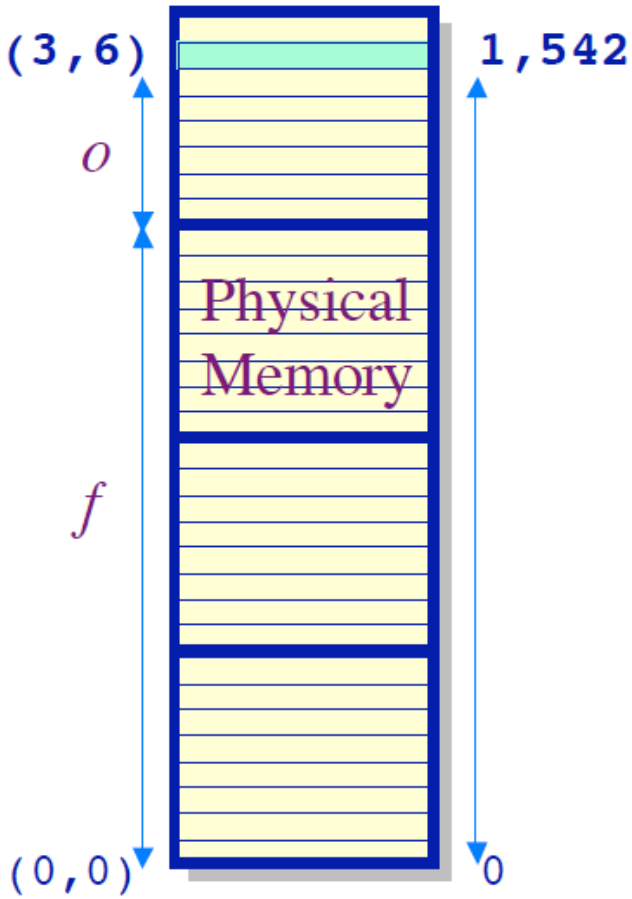




Puslapiavimas (1)

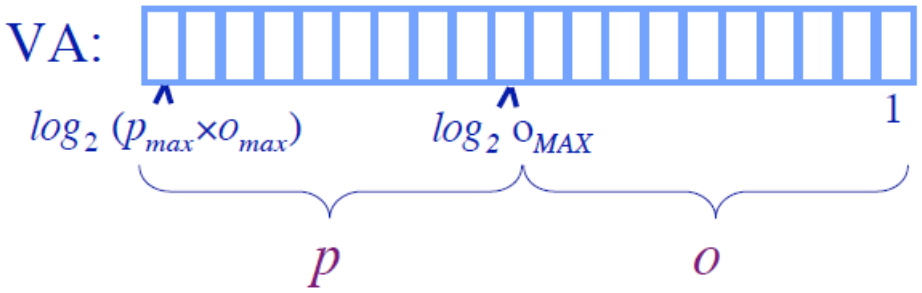
- Pavyzdys: 16 bitų adresų erdvė padalinta į 512 B rėmus.
  - Adresuojama vieta (3,6) = 1542

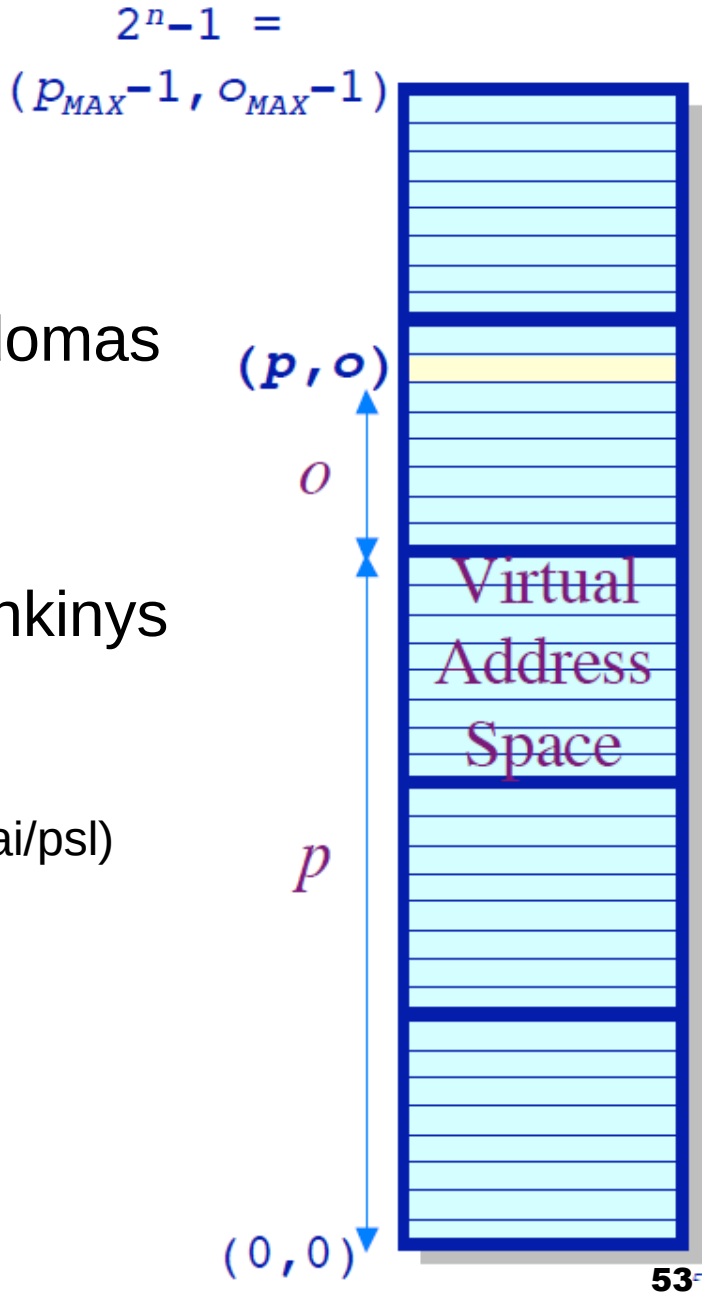




Puslapiavimas (2)

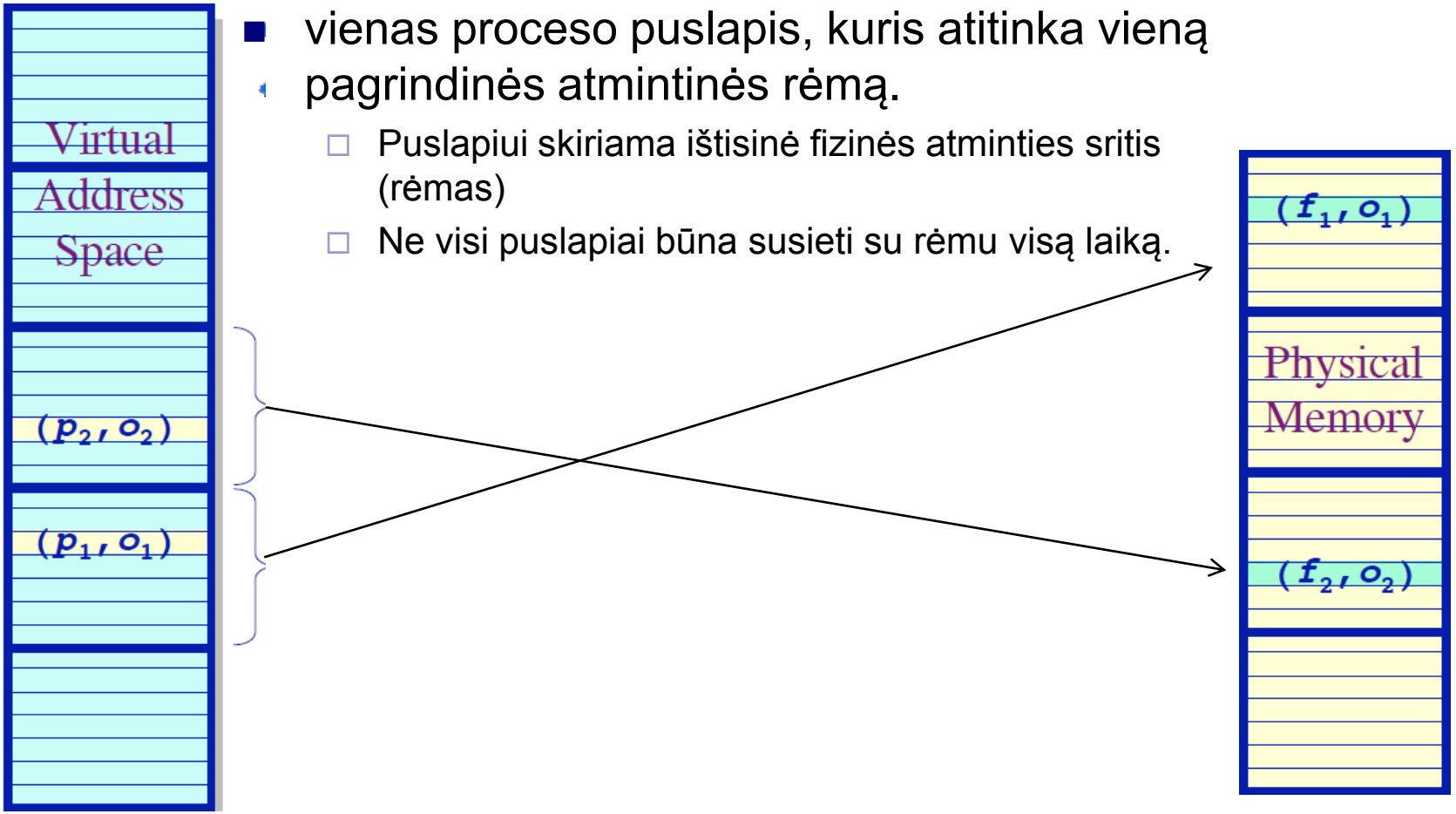
- Kiekvienas procesas yra suskaidomas į vienodo dydžio puslapius.
  - |puslapis|=|puslapio rėmas|
- Virtualus puslapio adresas yra rinkinys (*p*, *o*):
  - *p* – psl numeris (p_{max} psl)
  - *o* – poslinkis nuo psl pradžios (o_{max} baitai/psl)
  - Virtualus psl adresas= $o_{\max} \times p + o$



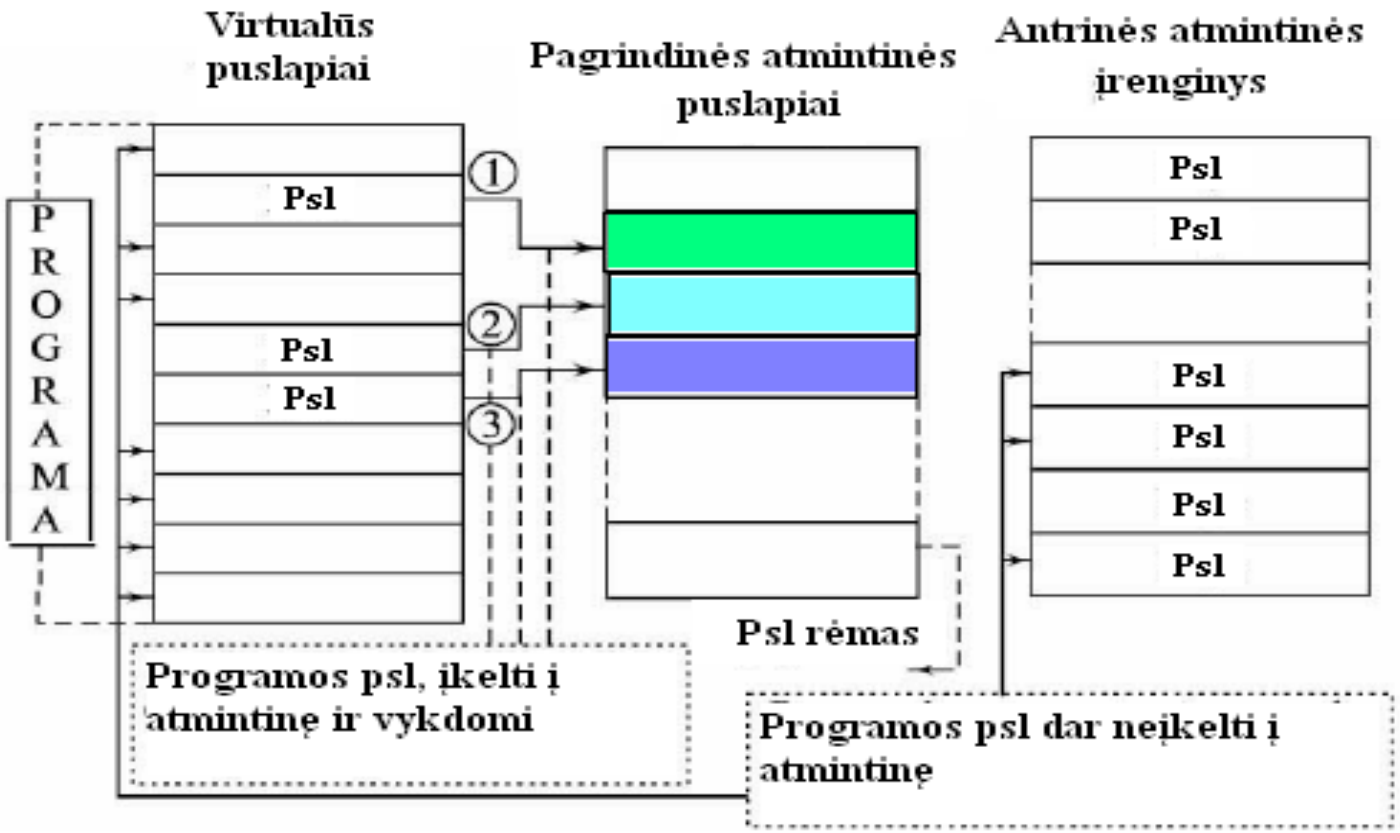


Puslapiavimas (3)

- vienas proceso puslapis, kuris atitinka vieną pagrindinės atmintinės rėmą.
  - Puslapiui skiriama ištisinė fizinės atminties sritis (rėmas)
  - Ne visi puslapiai būna susieti su rėmu visą laiką.



Programos įkėlimas

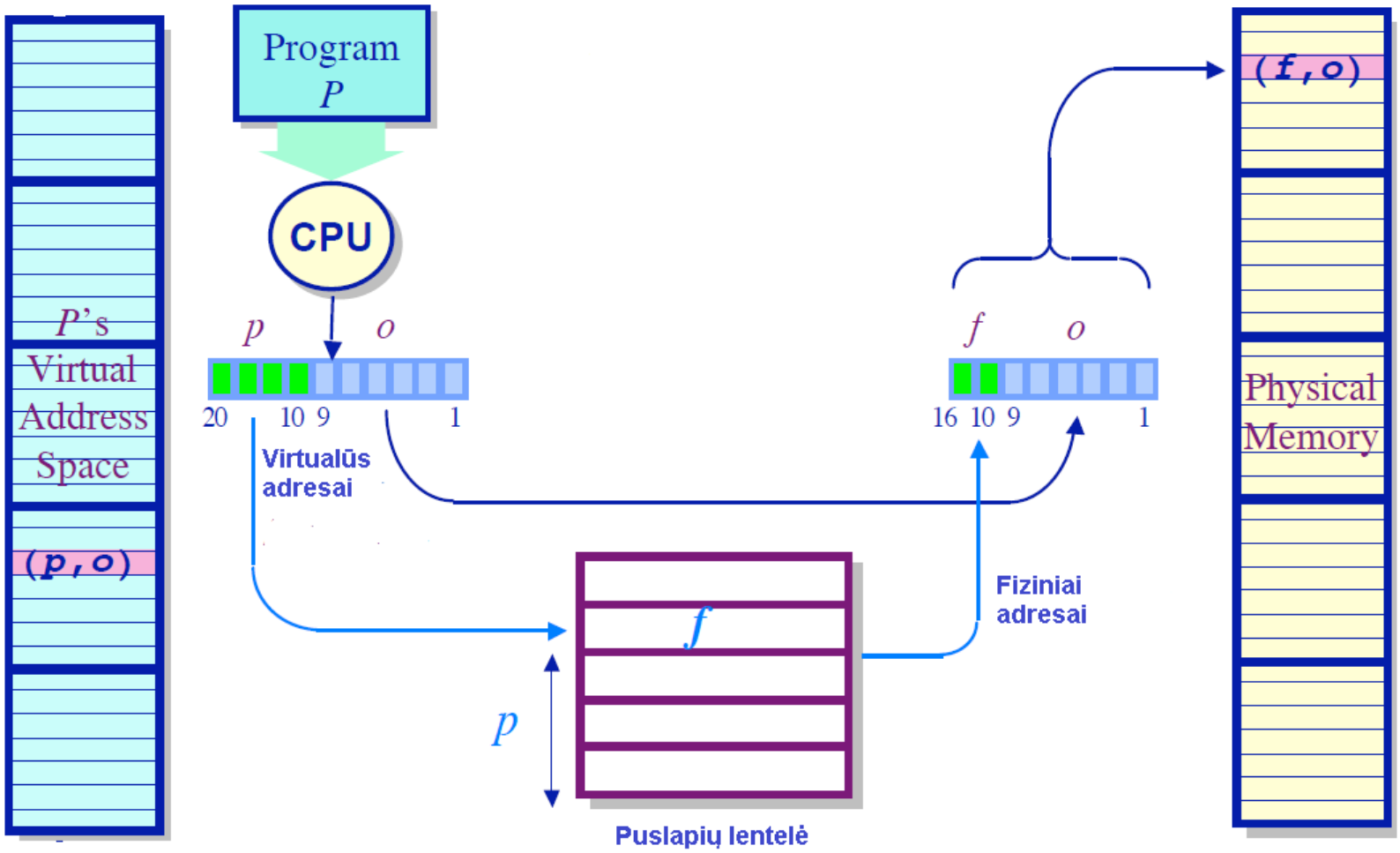


Fragmentacija naudojant puslapiavimą

- Nebelieka išorinės fragmentacijos :
  - Proceso puslapiai proceso talpinimo į pagrindinę atmintinę metu gali užimti laisvus rėmus (page frames).
  - Nauda:
    - procesui nebūtina užimti ištisinės adresų erdvės pagrindinėje atmintinėje
    - proceso puslapiai gali būti išbarstomi į egzistuojančius laisvus rėmus.
  - Procesui pasibaigus, tiesiog padaugėja laisvų rėmų.
- Kadangi puslapio (rėmo) dydis yra pakankamai nedidelis, tai sumažėja ir vidinė fragmentacija.

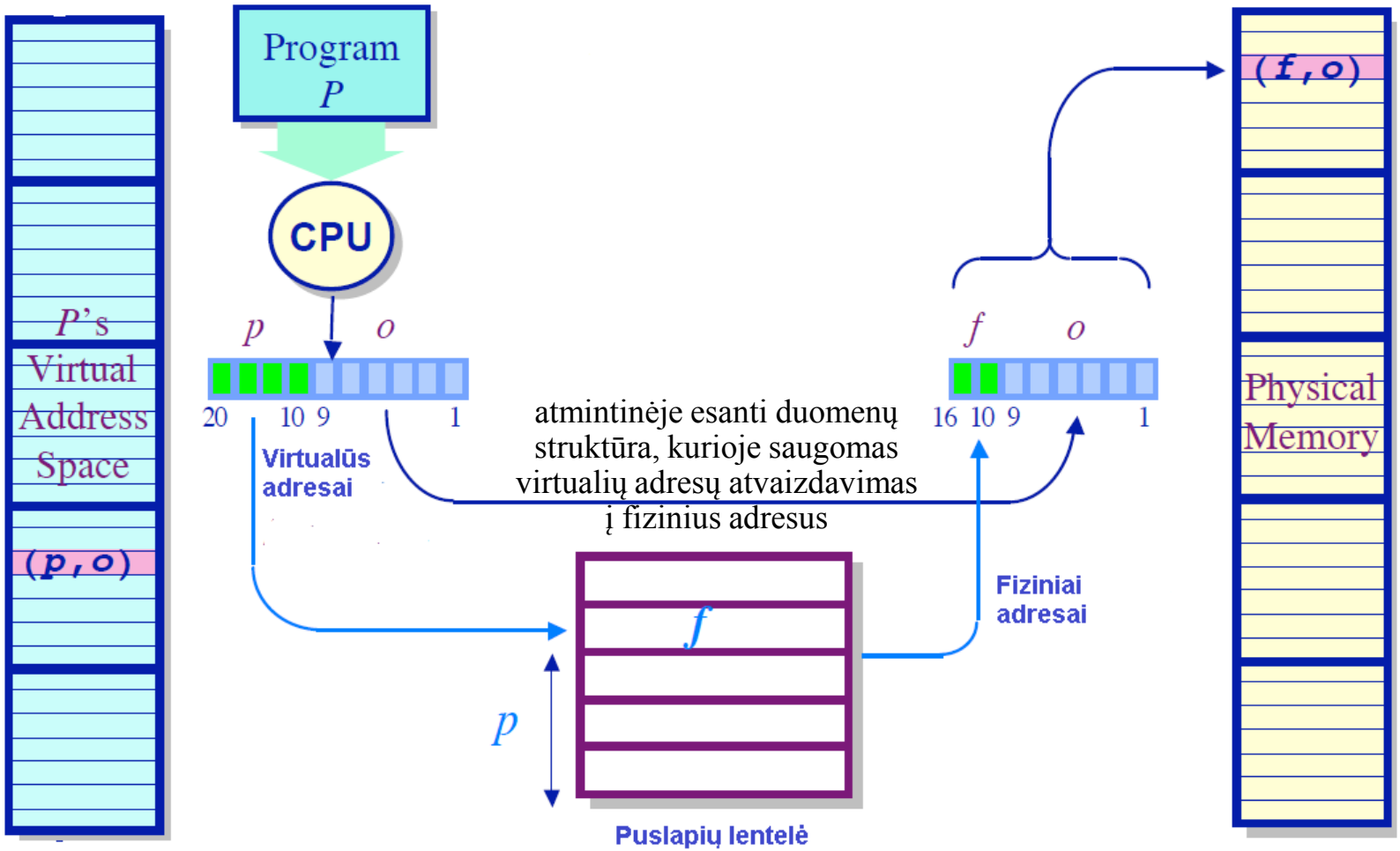
Puslapiavimas (4)

Virtualaus adreso transliavimas į fizinį



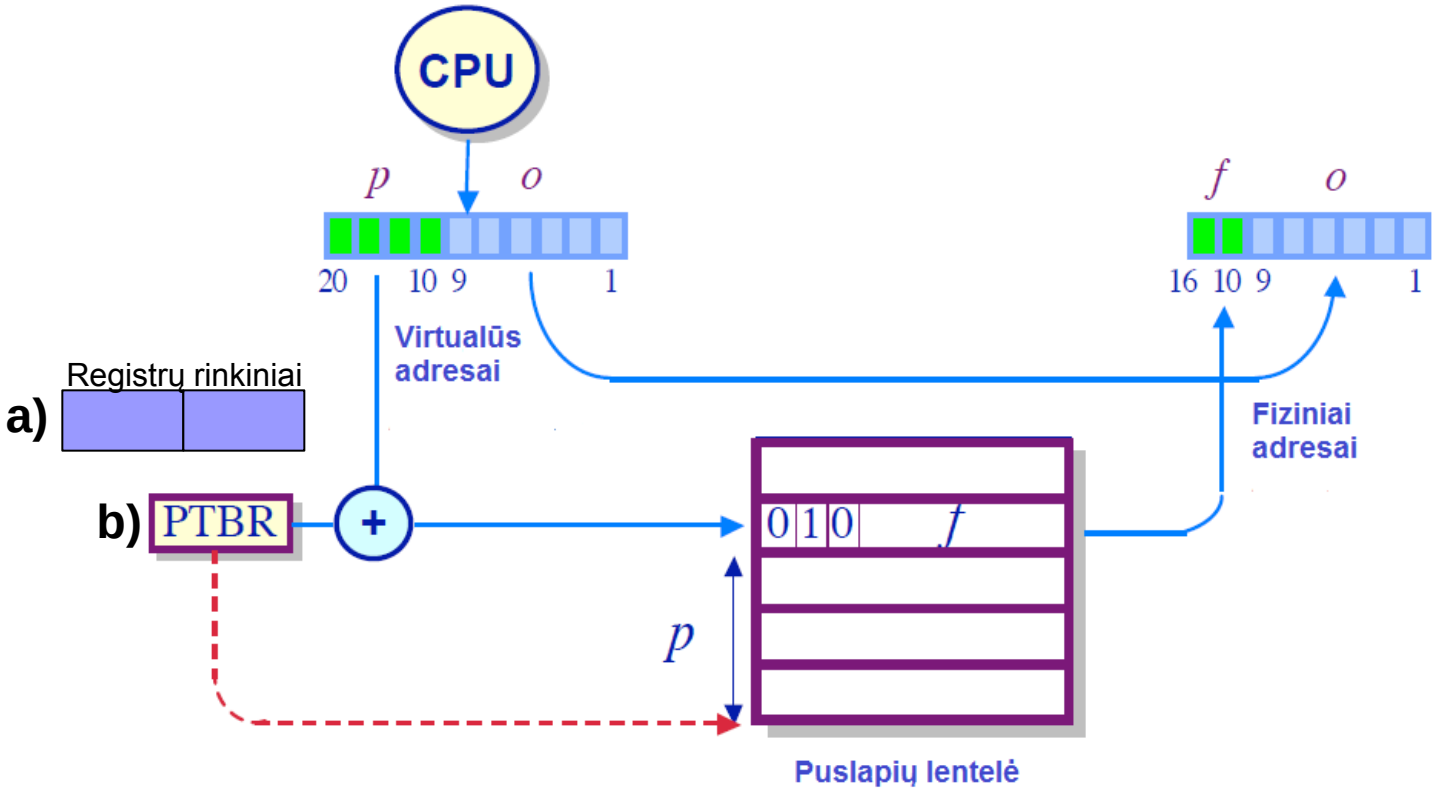
Puslapiavimas (4)

Virtualaus adreso transliavimas į fizinį



Puslapiavimas (4)

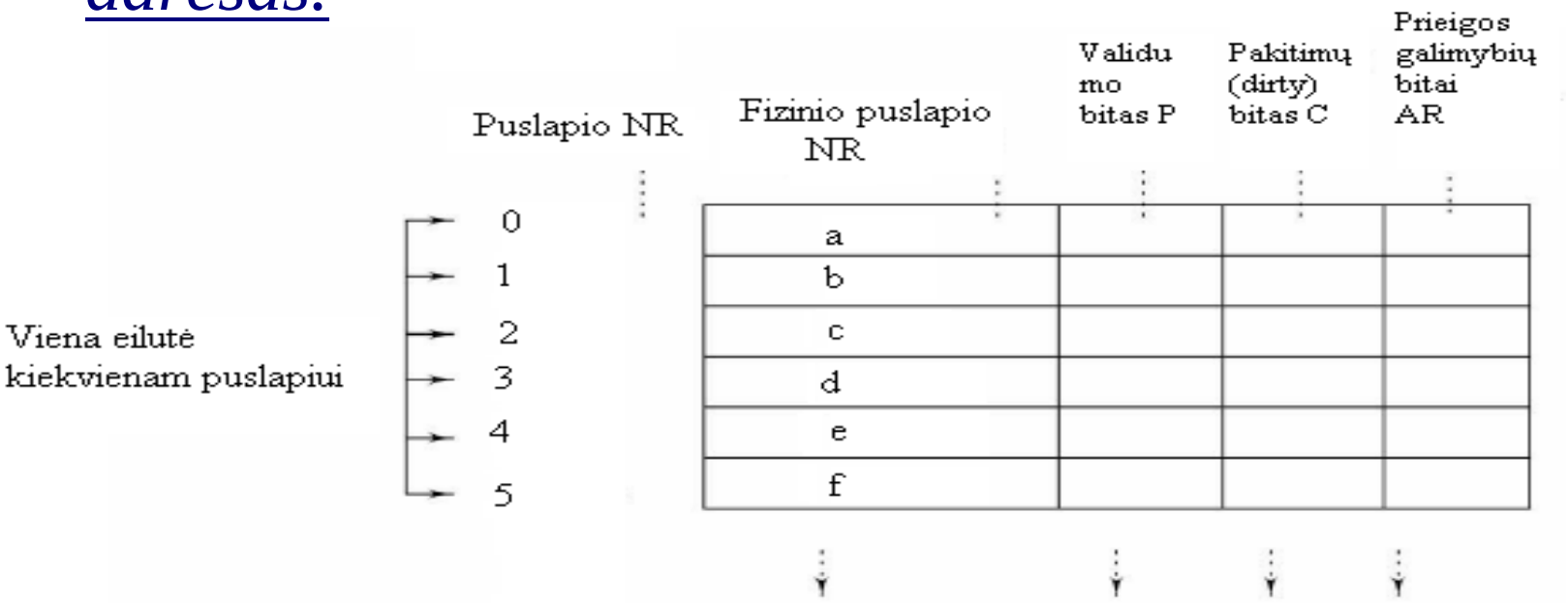
Virtualaus adreso transliavimas į fizinį



Puslapiavimas (5)

Puslapių lentelės struktūra

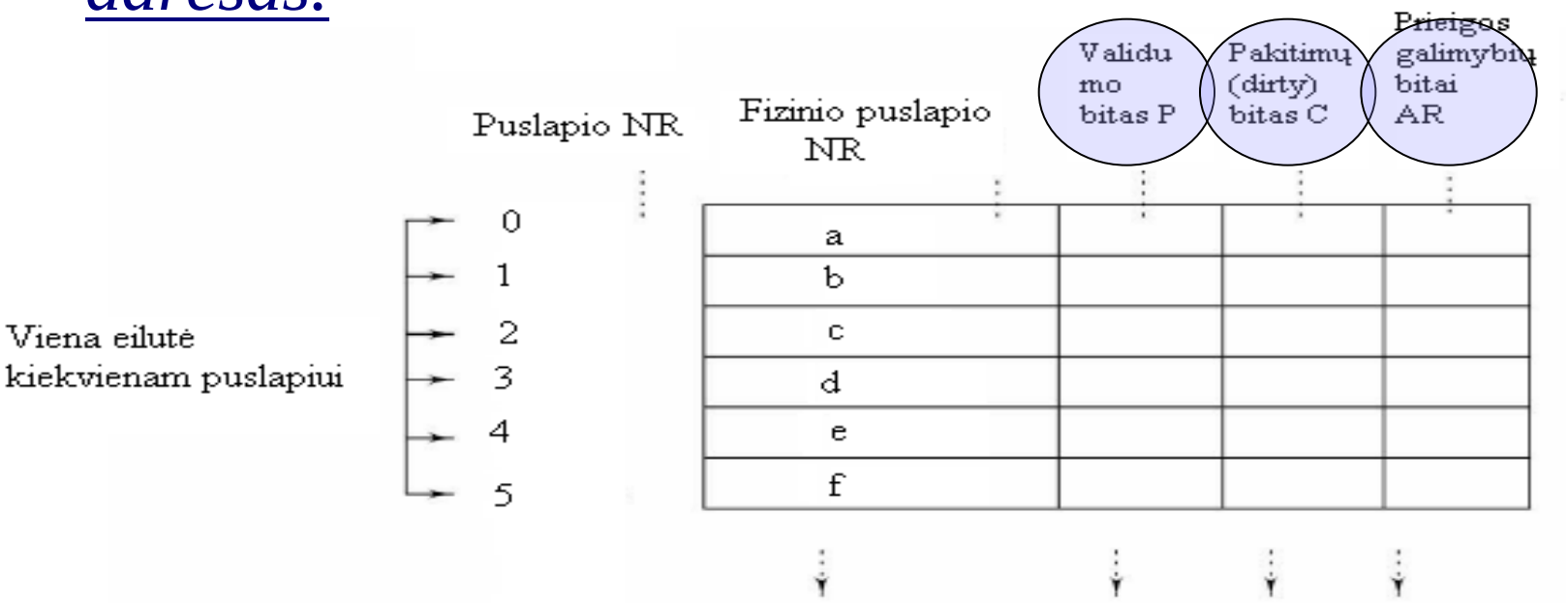
- Viename iš CPU registrų visada yra tuo metu vykdomo proceso puslapių lentelės *pradžios fizinis adresas.*



Puslapiavimas (5)

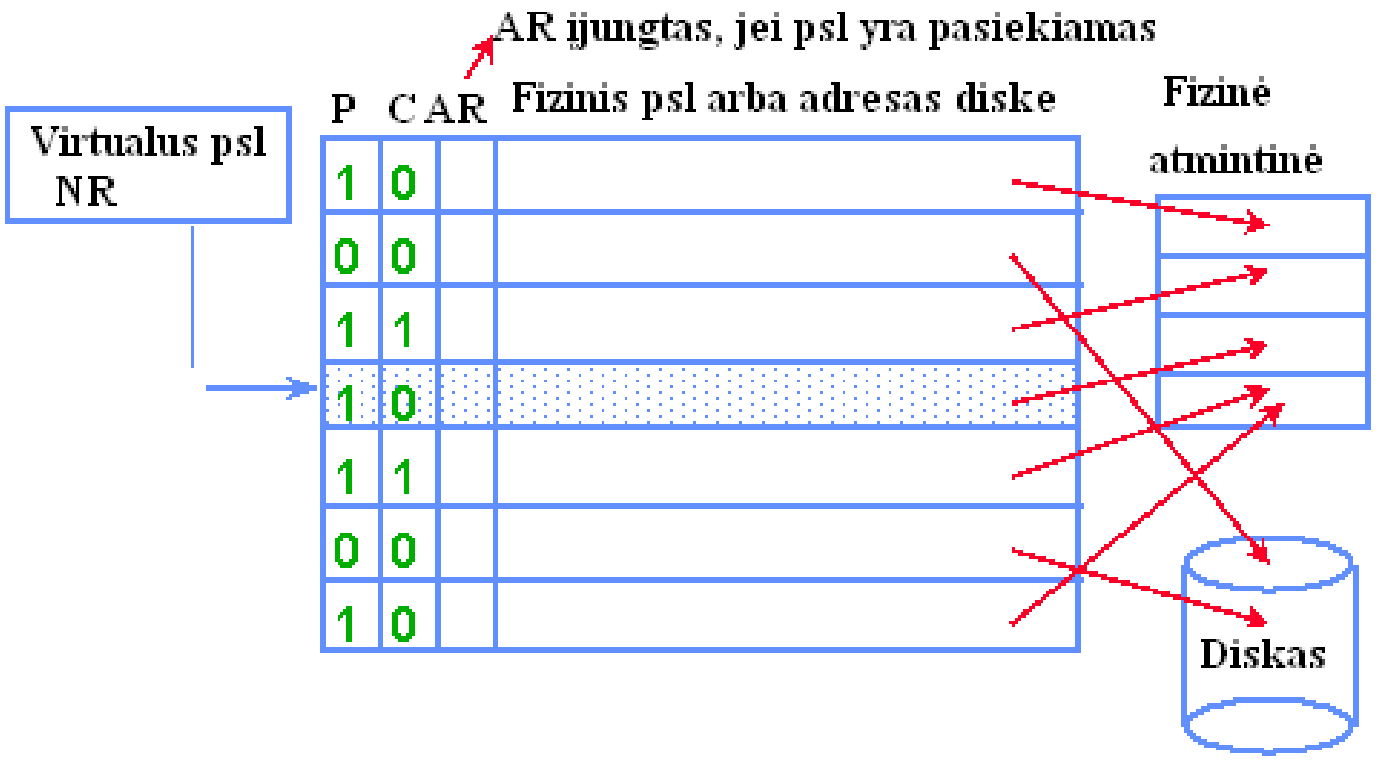
Puslapių lentelės struktūra

- Viename iš CPU registrų visada yra tuo metu vykdomo proceso puslapių lentelės *pradžios fizinis adresas.*



Puslapiavimas (6)

Puslapių išrinkimas



Su puslapiavimu susijusios problemos ir sprendimai

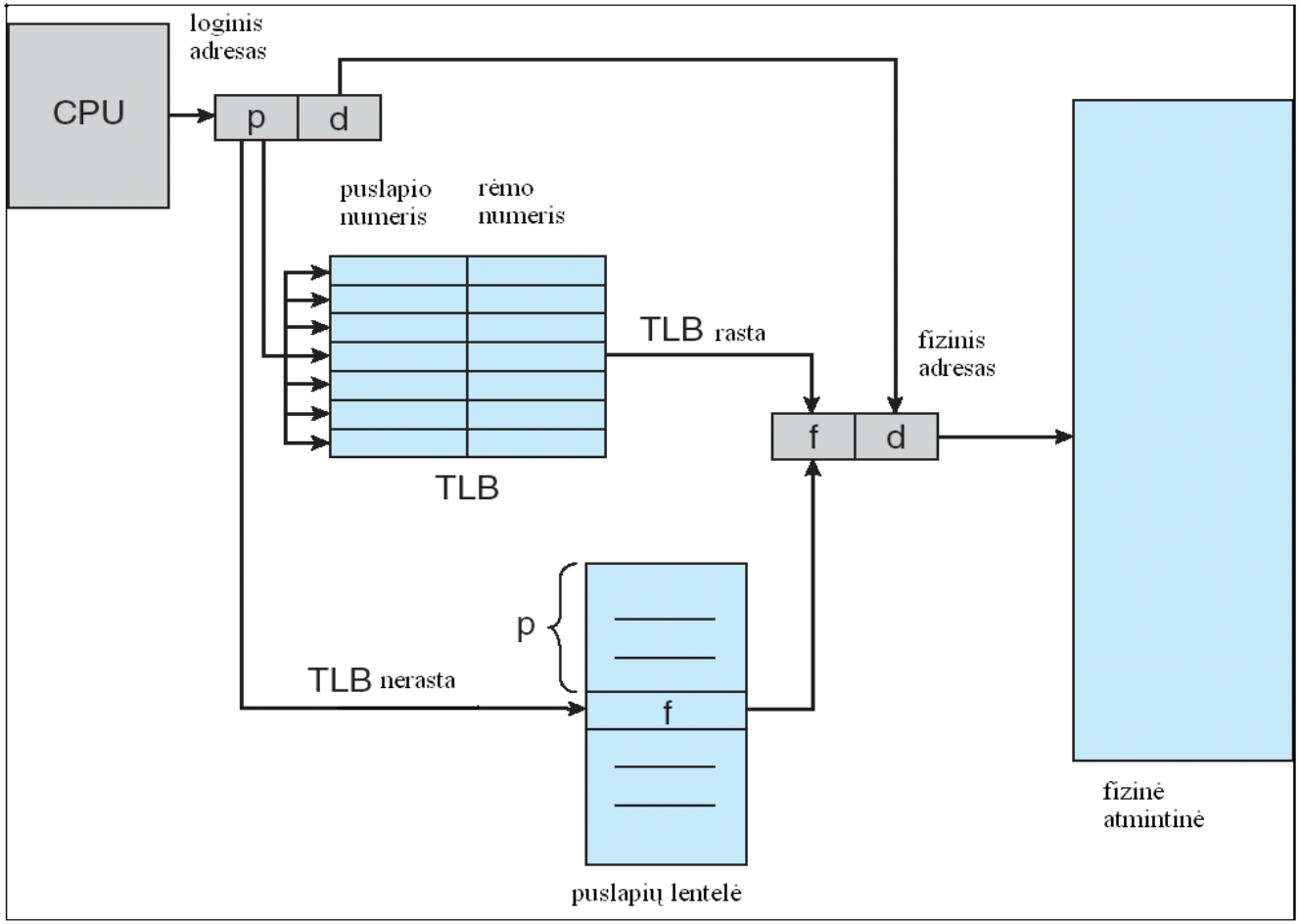
- Du kreipiniai į atmintį;
- Puslapio dydis
- Rezidentinis proceso dydis

Su puslapiavimu susijusios problemos ir sprendimai

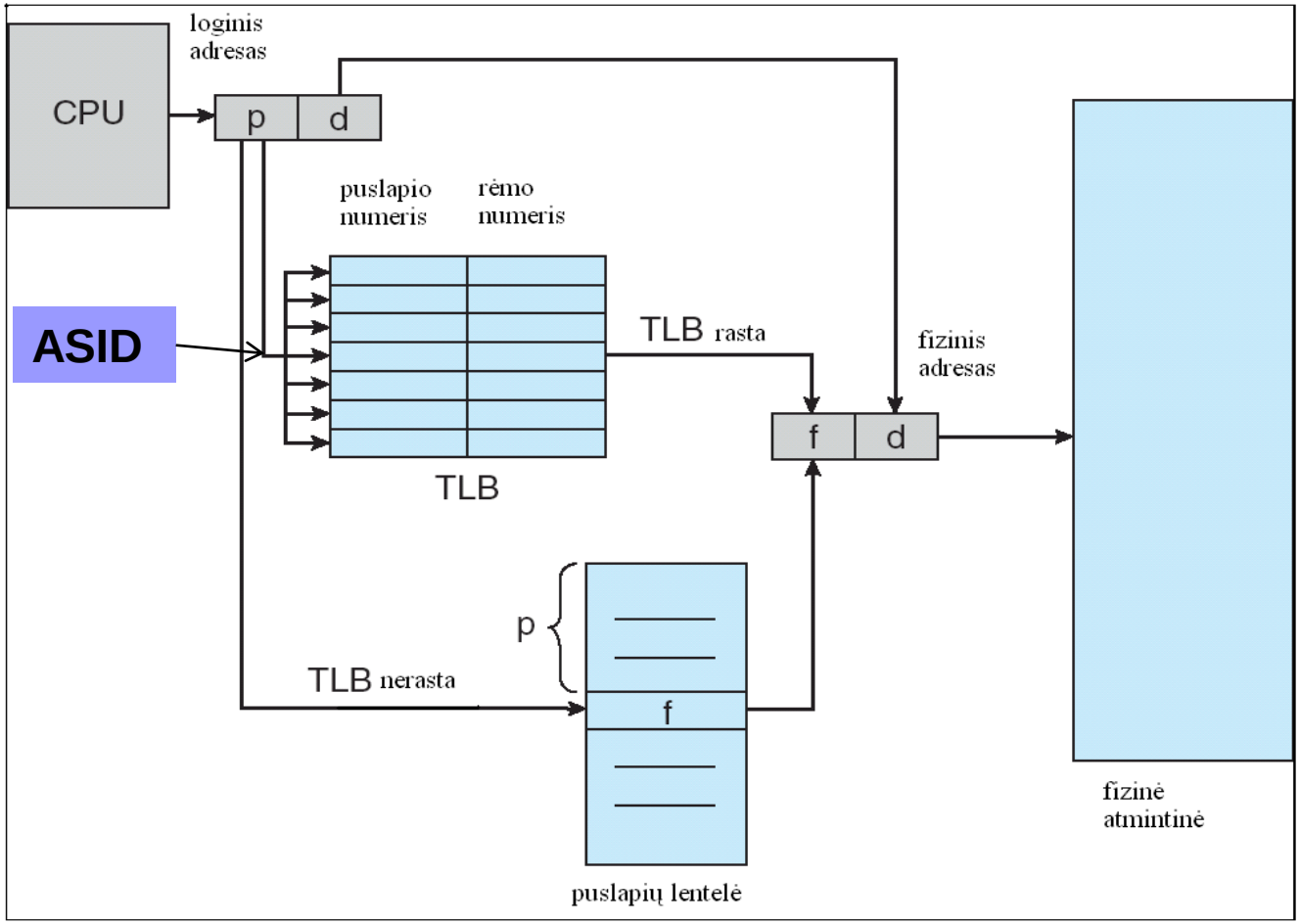
- Du kreipiniai į atmintį;
- Puslapio dydis
- Rezidentinis proceso dydis

- TLB lentelės
- 4KB, 8KB arba 16 KB
- Vienodi rėmų kiekiai
- Rėmų kiekis proporcingas procesų dydžiui

Adreso transliavimas naudojant TLB (Translation Look-Aside Buffer) įrašus



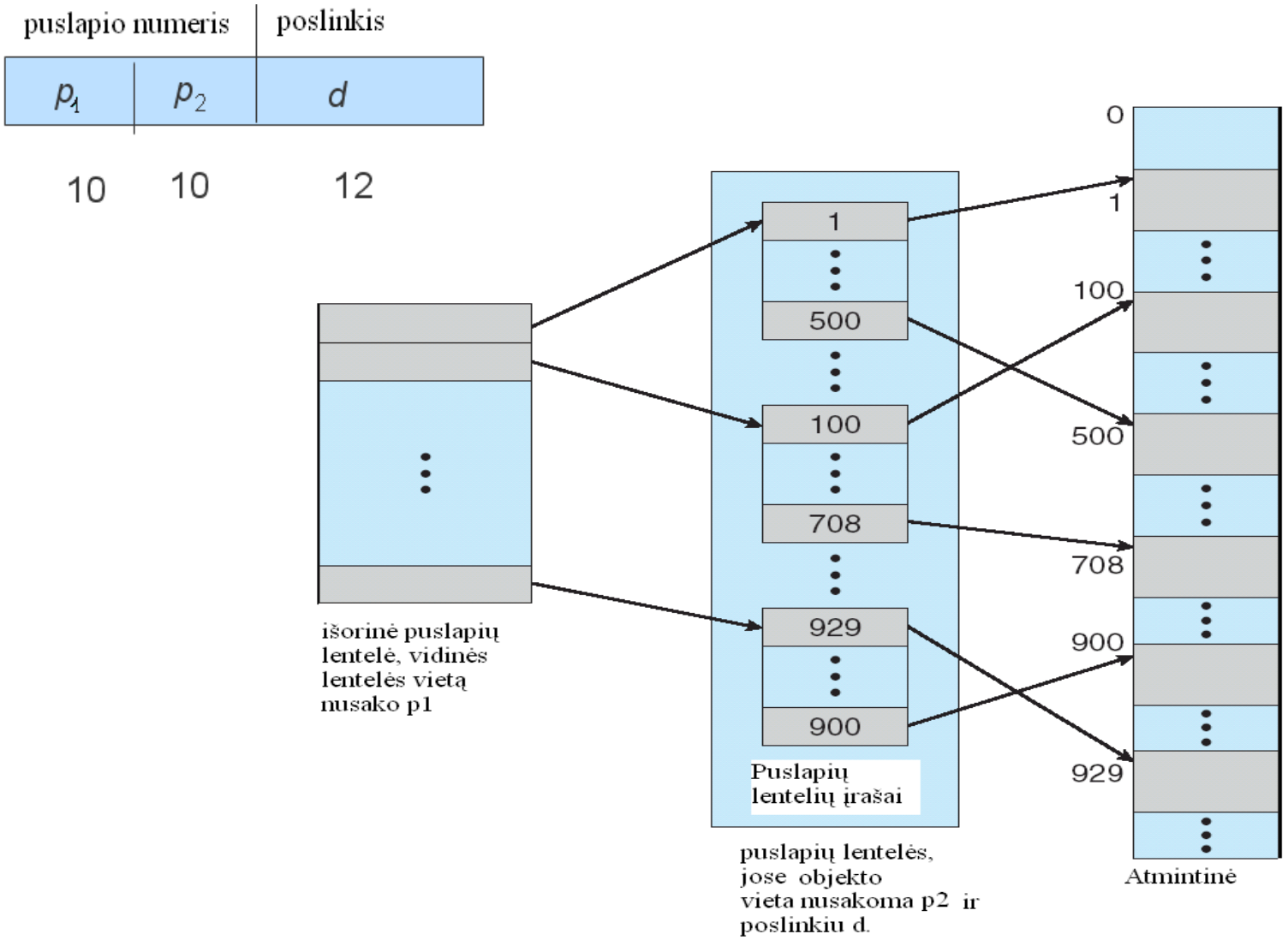
Adreso transliavimas naudojant TLB įrašus



Puslapių lentelių tipai

- Vieno lygio,
- hierarchinės,
- invertuotos santraukos (hash) tipo, puslapių lentelės

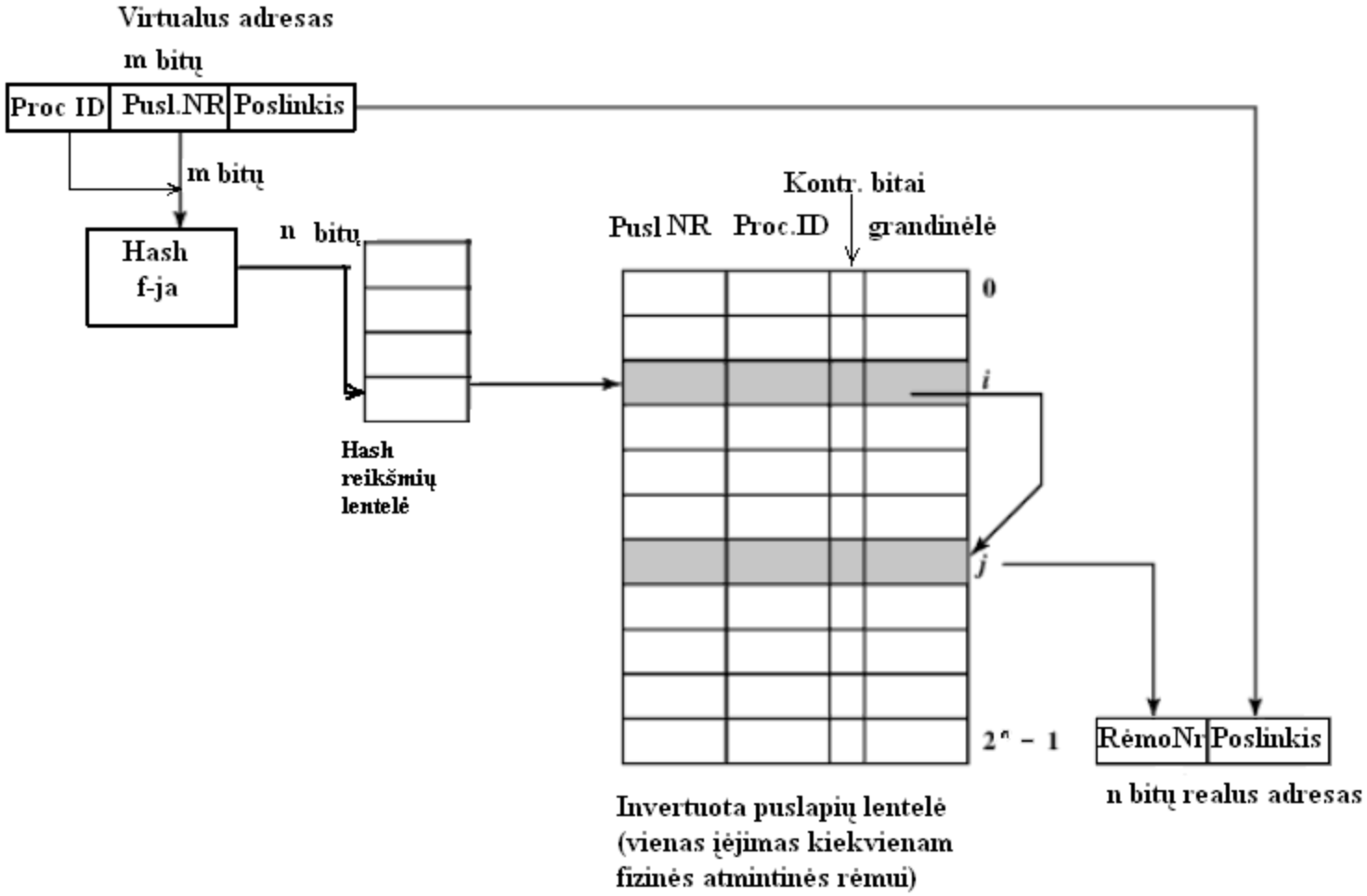
Dviejų lygių hierarchinė puslapių lentelė



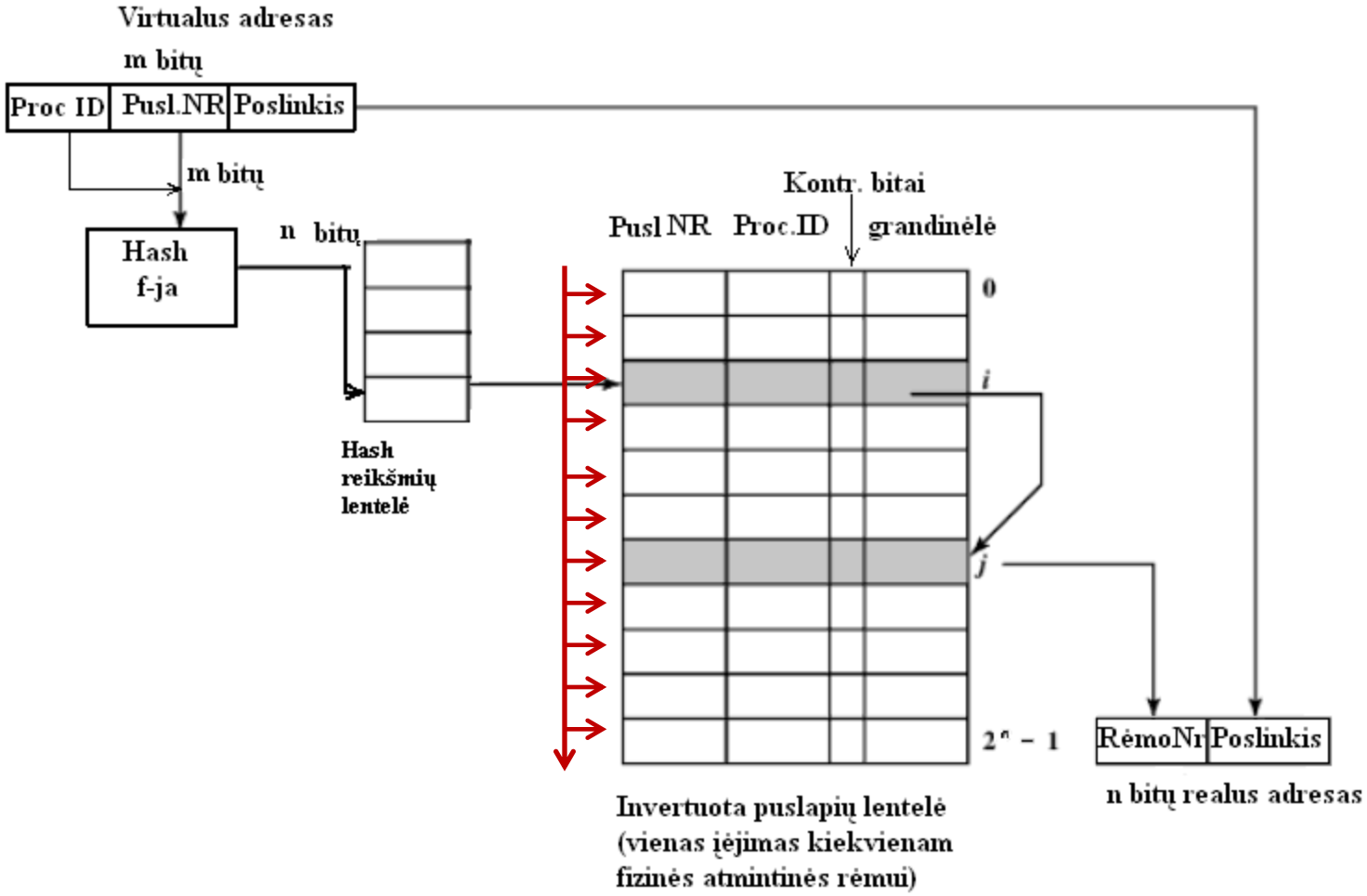
Puslapių lentelių tipai

- Vieno lygio,
- hierarchinės,
- Invertuoto tipo puslapių lentelės

Invertuoto (inverted) tipo puslapių lentelės



Invertuoto (inverted) tipo puslapių lentelės



Invertuoto (inverted) tipo puslapių lentelės