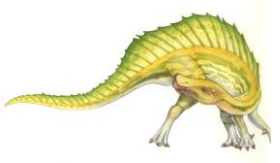
Operacinės sistemos

6. Atminties valdymas: pagrindinė ir virtualioji atmintis.

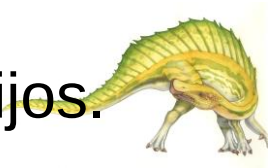
Pagrindinė atmintis potemės

- Pagrindai
- Apsikeitimas (angl. Swapping)
- Ištisinis atminties priskyrimas (angl. Contiguous Memory Allocation)
- Segmentavimas (angl. Segmentation)
- Puslapiavimas (angl. Paging)
- Puslapių lentelės stuktūra
- Pavyzdys: Intel 32 ir 64-bit architektūros
- Pavyzdys: ARM architektūra



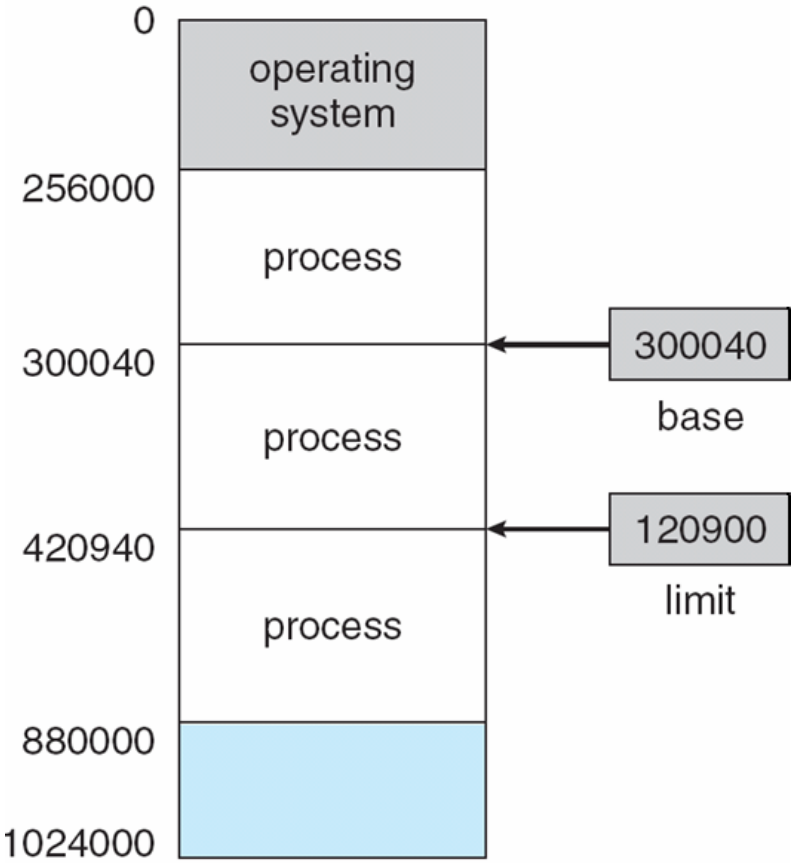
Pagrindai

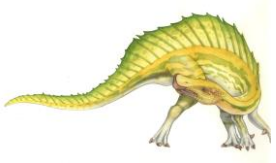
- Programa turi būti paimta iš disko į atmintį ir patalpinama procese vykdymui.
- Pagrindinė atmintis ir registrai yra vienintelės talpyklos, kurias CPU pasiekia tiesiogiai.
- Atminties įrenginys mato tik adresų srautą + skaitymo užklausas ar adresus + duomenų skaitymo ir rašymo užklausas.
- Registrai pasiekiami per vieną CPU taktą (ar mažiau).
- Pagrindinei atminčiai gali reikėti daugelio ciklų, taip sukeliant (angl. **stall**)
- **Kešas (angl. Cache)** yra tarp pagrindinės atminties ir CPU registrų.
- Atminties apsaugojimui reikalingos teisingos operacijos.



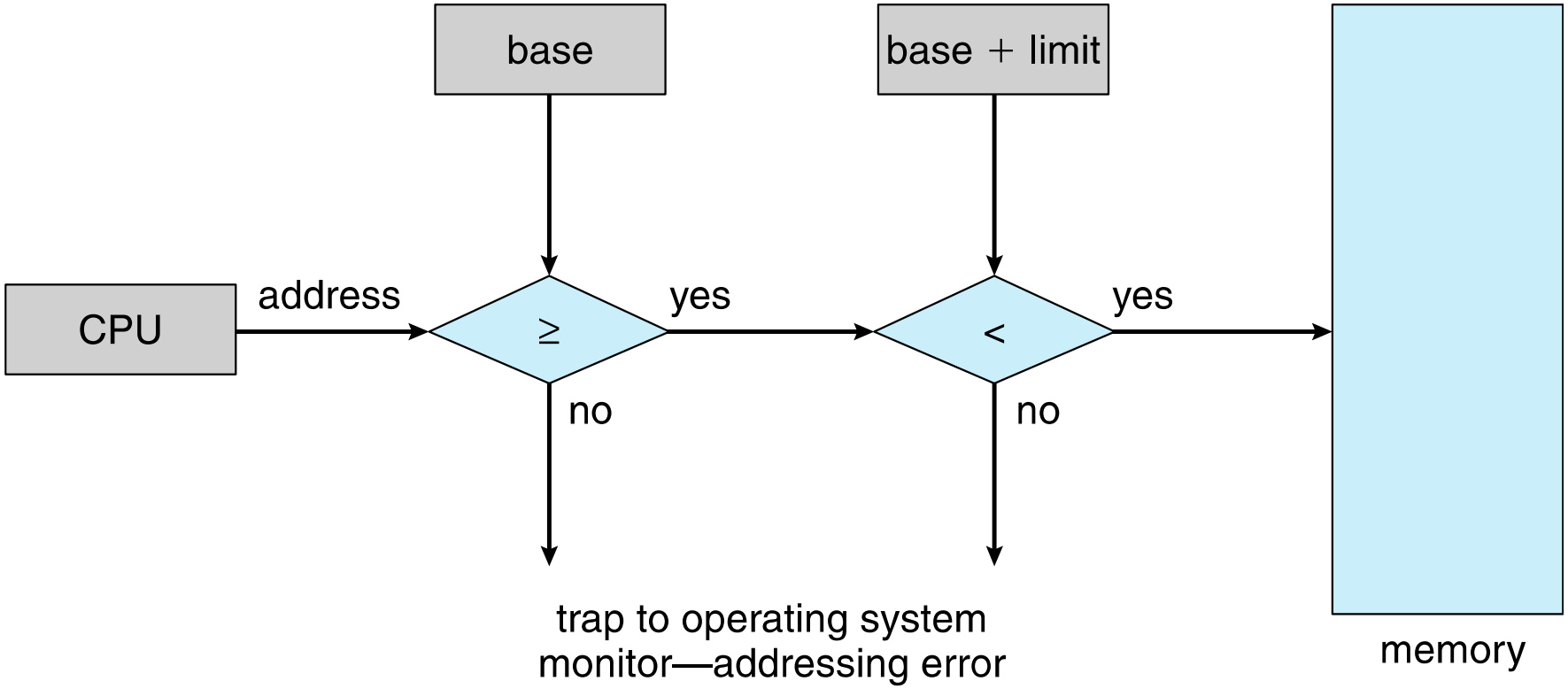
Bazinis ir ribinis registrai

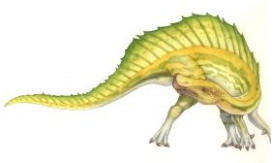
- Bazinio (angl. **base)** ir **ribinio** (angl. **limit)** registrų pora aprašo loginę adresų erdvę.
- CPU privalo patikrinti kiekvieną atminties prieigą, sugeneruotą vartotojo režime, kad įsitikintų, kad ji yra tarp bazės ir ribos tam vartotojui.





Techninis adresų apsaugojimas
(angl. Hardware Address Protection)





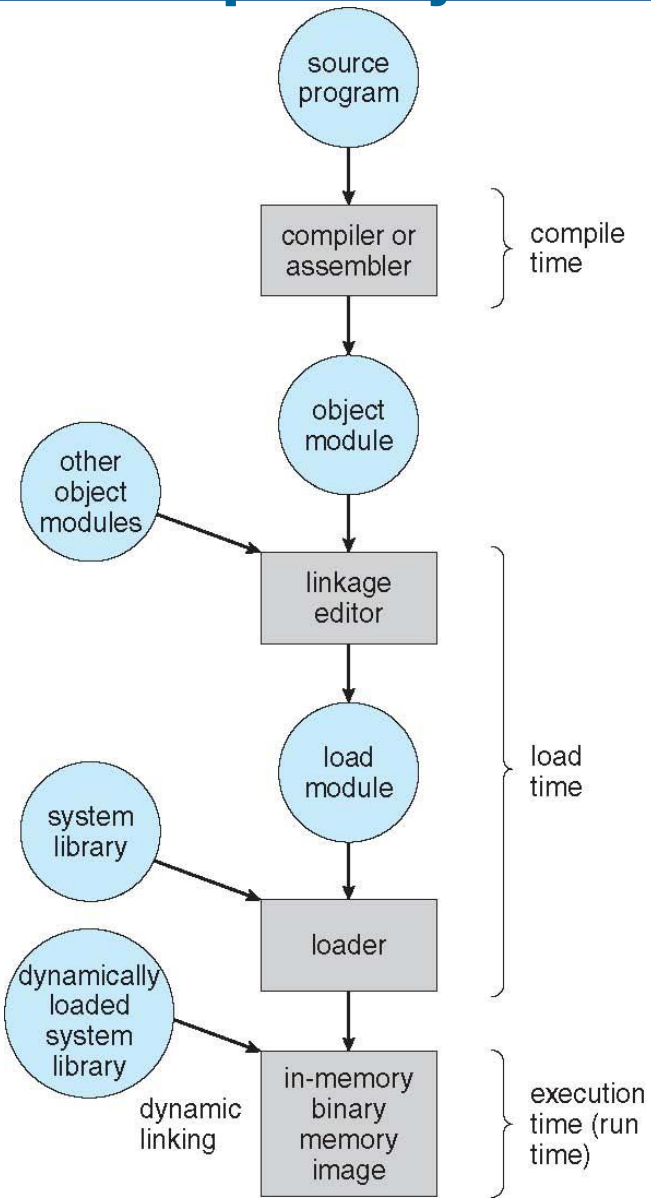
Adresų susiejimas
(angl. Address Binding)

- Programos diske, pasiruošusios būti įkeltos į atmintį vykdymui suformuoja **įvesties eilę** (angl. **input queue**)
  - Be palaikymo turi būti užkraunama adrese 0000.
- Nepatogu turėti pirmo vartotojo proceso fizinį adresą visada 0000.
  - Kaip galima to išvengti?
- Adresai atvaizduojami skirtingais būdais skirtingomis programos gyvavimo stadijomis.
  - Išeities kodo adresai, paprastai, simboliniai.
  - Sukompiliuoto kodo adresai susiejami (bind) su perkeliamais (relocatable) adresais.
    - pvz. “14 baitų nuo šio modulio pradžios”
  - Linker ar loader susieja perkeliamus adresus su absoliučiais adresais
    - pvz. 74014
  - Kiekvienas susiejimas susieja vieną adresų erdvę su kita.

Instrukcijų ir duomenų susiejimas su atmintimi

- Tai vykdoma trijose stadijose:
  - **Kompiliavimo metu**: Jei atminties vieta žinoma iš anksto, gali būti sugeneruojamas absoliutus kodas, būtina perkompiliuoti kodą, jei yra vietos pasikeitimų.
  - **Įkrovos metu**: būtina sugeneruoti perkeliamą kodą, jei atminties vieta nėra žinoma kompiliavimo metu.
  - **Vykdymo metu**: susiejimas uždelsiamas iki vykdymo, jei procesas gali būti perkeliamas vykdymo metu iš vieno atminties segmento į kitą.
    - Reikalingas techninės įrangos palaikymas adresų susiejimams (pvz., bazinis ir ribinis registrai).

Keleto žingsnių vartotojo programos apdorojimas

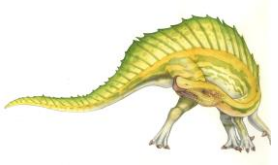


Loginė vs. Fizinė adresų erdvė

- Loginės adresų erdvės, apribotos atskiros fizinės adresų erdvės konceptas yra pagrindinis teisingame atminties valdyme.
  - **Loginis adresas** – sugeneruojamas CPU; taip pat, žymimas, kaip virtualus adresas (angl. **virtual address**)
  - **Fizinis adresas** – adresas matomas atminties įrenginio.
- Loginis ir fizinis adresai yra tokie patys kompiliavimo metu ir įkrovos metu, skiriasi vykdymo metu.
- **Loginė adreso erdvė** yra visų loginių adresų, sugeneruotų programos rinkinys.
- **Fizinė adresų erdvė** yra visų programos sugeneruotų fizinių adresų rinkinys.

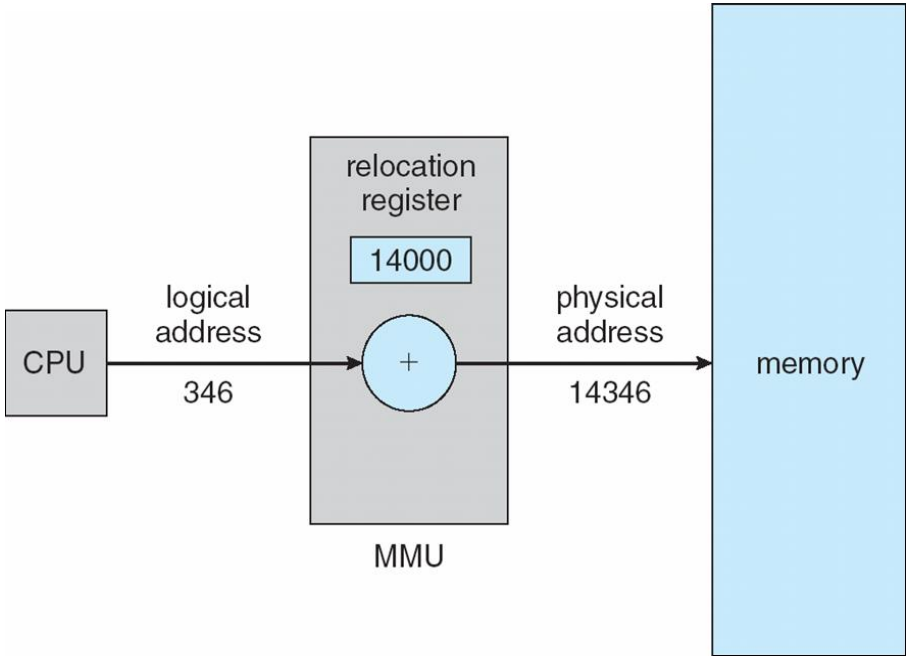
Atminties valdymo įtaisas
(angl. Memory-Management Unit (MMU))

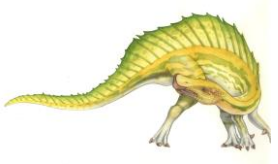
- Techninis įrenginys, vykdymo metu virtualius adresus priskiriantis fiziniams adresams.
- Galimi daug metodų.
- Paprastoje schemoje, kur reikšmė perkėlimo registre yra pridedama prie kiekvieno adreso, sugeneruoto vartotojo proceso, siuntimo į atmintį metu.
  - Bazinis registras čia vadinamas perkėlimo registru (angl. **relocation register**).
  - MS-DOS Intel 80x86 naudojo 4 perkėlimo registrus.
- Vartotojo programa dirba su loginiais adresas ir niekada nemato realių fizinių adresų.
  - Priskyrimas vykdymo laiku vyksta, kai yra padaroma nuoroda į vietą atmintyje.
  - Loginis adresas yra priskiriamas fiziniam adresui.



Dinaminis perkėlimas, naudojant perkėlimo registrą (angl. Dynamic relocation using a relocation register)

- Paprogramė nėra paleidžiama, kol nėra iškviečiama.
- Geresnis atminties išnaudojimas, nenaudojama paprogramė niekada nepaleidžiama.
- Visos paprogramės yra laikomos diske perkėlimo įkrovos (relocatable load) formatu.
- Naudinga, kai dideli kiekiai kodo reikalingi retai.
- Nereikalingas specialus OS palaikymas.
  - Įgyvendinamas per programų projektavimą.
  - OS gali padėti suteikdama bibliotekas dinaminei įkrovai



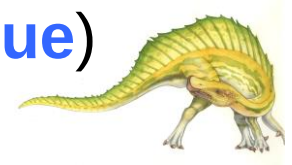


Dinaminis susiejimas
(angl. Dynamic Linking)

- **Statinis susiejimas (angl. Static linking)** – sistemos bibliotekos ir programos kodas yra apjungiami paleidėjo į binarinį programos atvaizdą.
- **Dinaminis susiejimas** – susiejimas atidedamas iki vykdymo.
- Mažos kodo dalys – **stub**, yra naudojamos, kad surasti reikiamą atmintyje esančią bibliotekos paprogramę.
- Stub pakeičia save paprogramės adresu ir vykdo paprogramę.
- OS patikrina, ar paprogramė yra procesų atminties adrese
  - Jei nėra adresų erdvėje, pridedama į adresų erdvę.
- Dinaminis susiejimas ypač naudingas bibliotekoms.
- Sistemoje dar žinoma, kaip **bendros bibliotekos** (angl. **shared libraries**)

Apkeitimas (angl. Swapping)

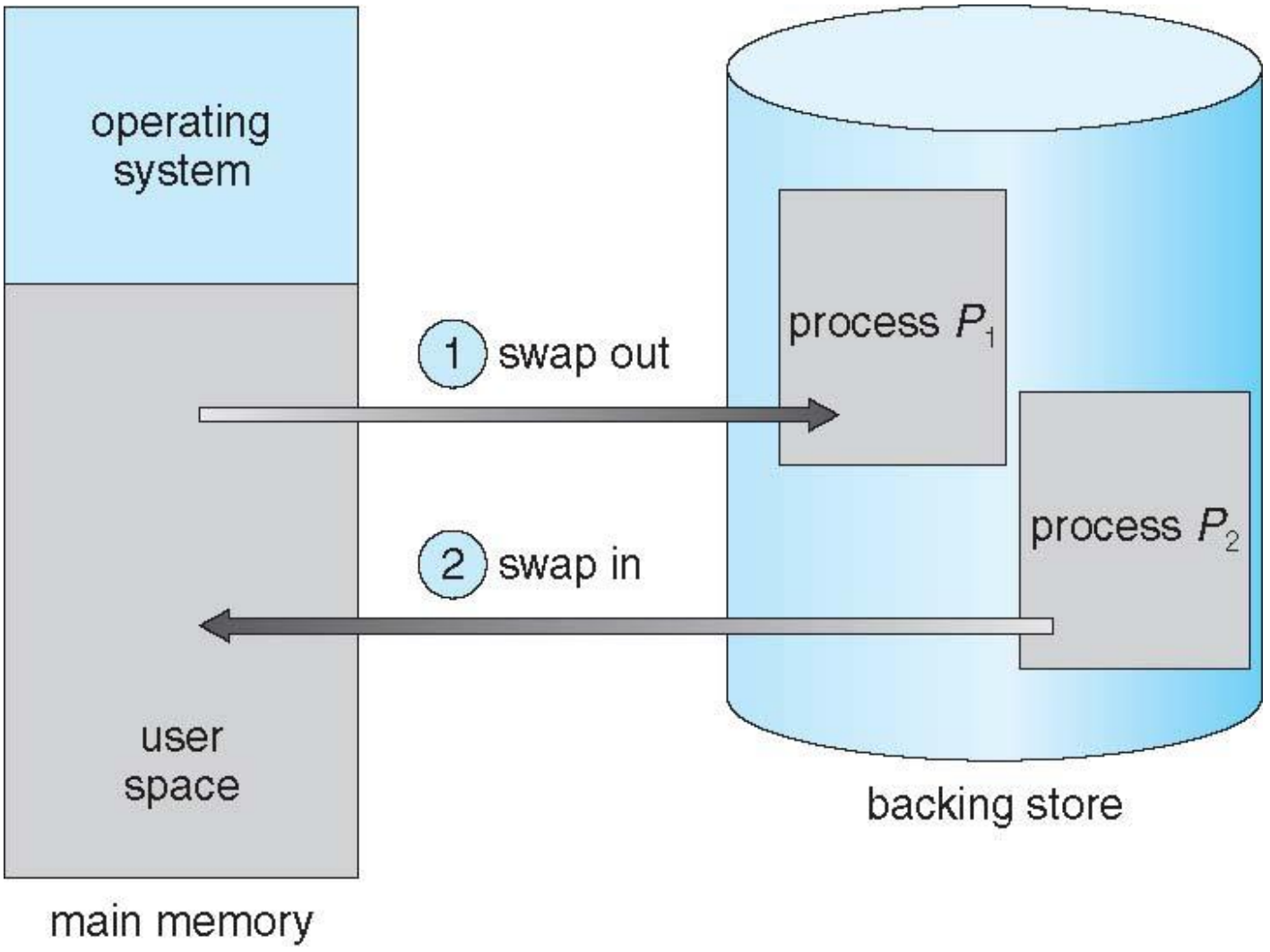
- Procesas gali būti išimtas iš atminties ir laikinai patalpintas į saugyklą, tuomet grąžintas į atmintį tolesniam vykdymui.
  - Bendra fizinė procesų atminties erdvė gali viršyti fizinę atmintį.
- **Atsarginė saugykla (angl. Backing store)** – pakankamai greitas diskas, leidžiantis saugoti visų vartotojų atminties atvaizdus, turi suteikti tiesioginę prieigą prie tų atvaizdų.
- **Roll out, roll in** – apkeitimo variantas, naudojamas prioritetais grindžiamuose tvarkaraščių sudarymo algoritmuose, žemesnio prioriteto procesai yra pakeičiami aukštesnio prioriteto procesais.
- Didžioji dalis laiko apkeičiant yra duomenų perdavimo laikas. Bendras perdavimo laikas yra tiesiogiai proporcingas apkeičiamos atminties dydžiui.
- Sistema saugoja pasiruošusių procesų (**ready queue**) eilę, kurių atminties atvaizdai yra diske.



Apkeitimas (tęs.)

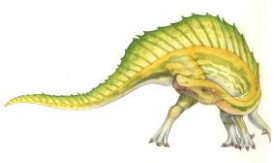
- Ar apkeistas procesas turi sugrįžti į tą pačią fizinio adreso erdvę?
- Tai priklauso nuo adresų susiejimo metodo.
  - Plius svarbu laukiantys I/O į/iš atminties erdvės.
- Modifikuotos apkeitimo versijos sutinkamos daugumoje sistemų (pvz. UNIX, Linux ir Windows)
  - Apkeitimas, paprastai, yra išjungtas.
  - Pradedamas, jei yra priskiriama daugiau nei numatyta atminties.
  - Išjungiama, kai atminties poreikis sumažėja

Schematinis apkeitimo vaizdas



Konteksto perjungimo laikas, įskaitant apkeitimą (angl. Context Switch Time including Swapping)

- Jei sekantis procesas, kuris turi būti paduotas CPU nėra atmintyje, reikia apkeisti procesą į paskirties.
- Tokiu atveju konteksto perjungimo laikas gali būti labai ilgas.
- 100MB proceso apkeitimas į kietąjį diską su duomenų perdavimo sparta 50MB/s.
  - Išėmimo laikas: 2000 ms
  - + kito proceso įkėlimo laikas
  - Bendras konteksto perjungimo laikas 4000ms (4 s)
- Galima sumažinti, jei sumažinama apkeičiamos atminties apimtis – žinant kiek atminties yra sunaudota
  - Sisteminiai iškvietimai informuoti OS apie atminties sunaudojimą: `request_memory()` ir `release_memory()`

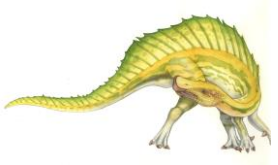


Konteksto perjungimo laikas ir apkeitimas (tęs.)

- Kiti apkeitimo apribojimai:
  - Laukiantys I/O – negalima apkeisti, nes I/O būtų atliekamas kitam procesui.
  - Arba visada perduoti I/O į branduolio erdvę, tuomet į I/O įrenginį.
    - Žinoma, kaip dvigubas buferizavimas (**double buffering**), atsiranda pridėtinės išlaidos.
- Standartinis apkeitimas nenaudojamas moderniose OS.
  - Tačiau paplitusios modifikuotos versijos.
    - Apkeičiama, kai laisvos atminties ypač mažai.

Apkeitimas mobiliose sistemose

- Paprastai nepalaikomas
  - Grindžiamas flash atmintimi
    - Mažas atminties kiekis
    - Ribotas rašymo ciklų skaičius
    - Prastas pralaidumas tarp flash atminties ir CPU mobilioje platformoje.
- Vietoje to naudojami kiti metodai atlaisvinti atminčiai.
  - iOS paprašo programų savanoriškai atlaisvinti priskirtą atmintį.
    - Skaityti tik išimtus duomenis ir perkrauti iš flash, jei reikia.
    - Nepavykęs atlaisvinimas gali lemti darbo nutraukimą.
  - Android nutraukia programų darbą, jei mažai atminties, tačiau iš pradžių įrašo programos būseną (**application state**) į flash greitam perkrovimui.
  - Abi OS palaiko puslapiavimą.

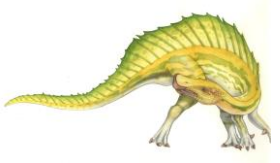


Ištisinis priskyrimas
(angl. Contiguous Allocation)

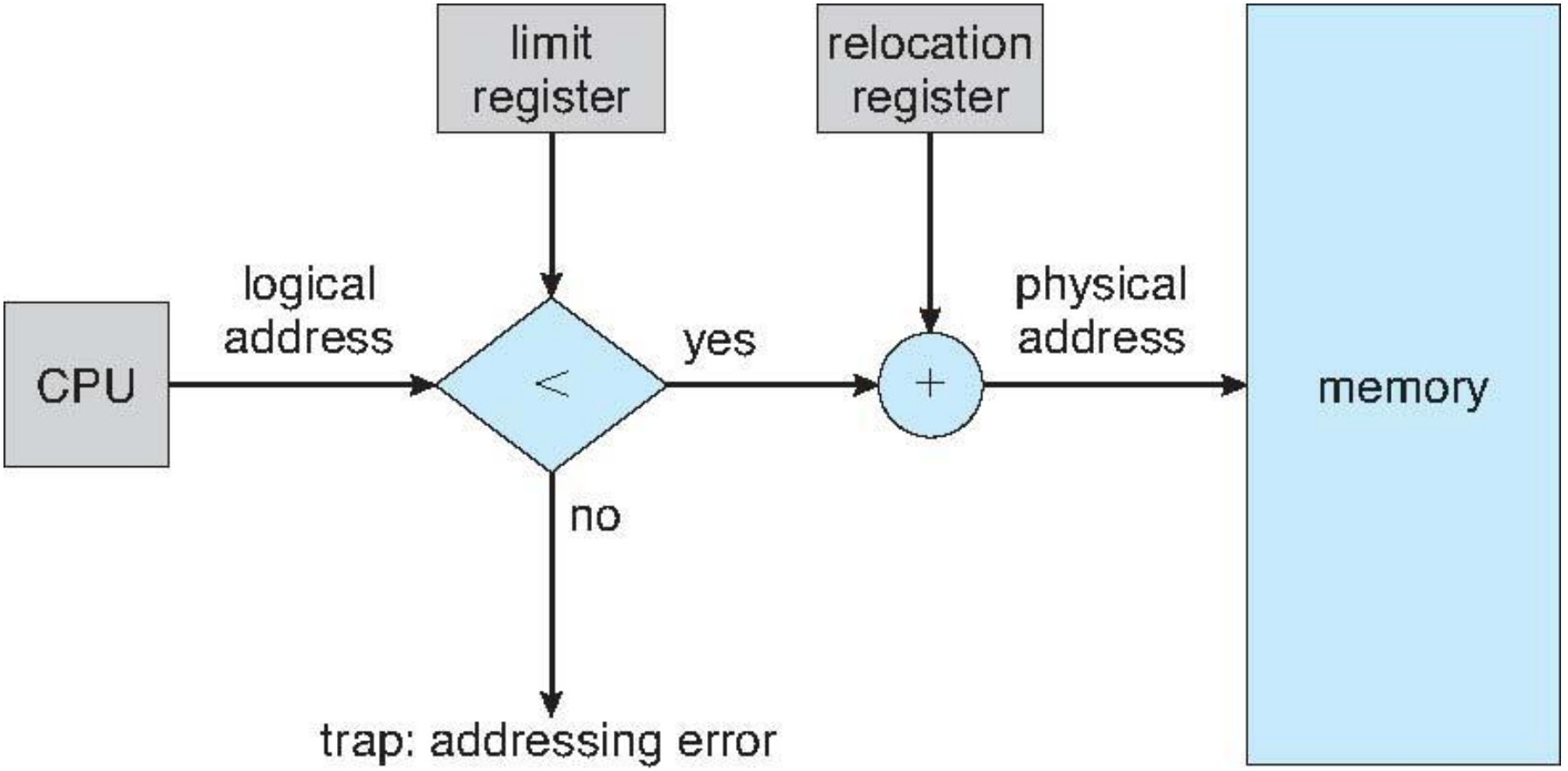
- Pagrindinė atmintis turi palaikyti tiek OS, tiek vartotojo procesus.
- Riboti resursai, reikia priskirti efektyviai.
- Gretimas priskyrimas yra ankstyvas metodas.
- Pagrindinė atmintis padalijama į dvi dalis (**partitions**):
  - OS, paprastai laikoma žemojoje atmintyje (low memory) su pertraukimo vektoriumi.
  - Vartotojo procesai laikomi aukštojoje atmintyje (high memory).
  - Kiekvienas procesas laikomas vienoje ištisinėje atminties sekcijoje.

Ištisinis priskyrimas (tęs.)

- Perkėlimo registrai naudojami apsaugoti vartotojo procesus vieni nuo kitų ir nuo OS kodo ir duomenų pakeitimo.
  - Baziniai registrai saugo mažiausio fizinio adreso reikšmę.
  - Ribiniai registrai saugo loginių adresų ruožą – kiekvienas loginis adresas turi būti mažesnis, nei ribinis registras.
  - MMU loginius adresus priskiria dinamiškai.

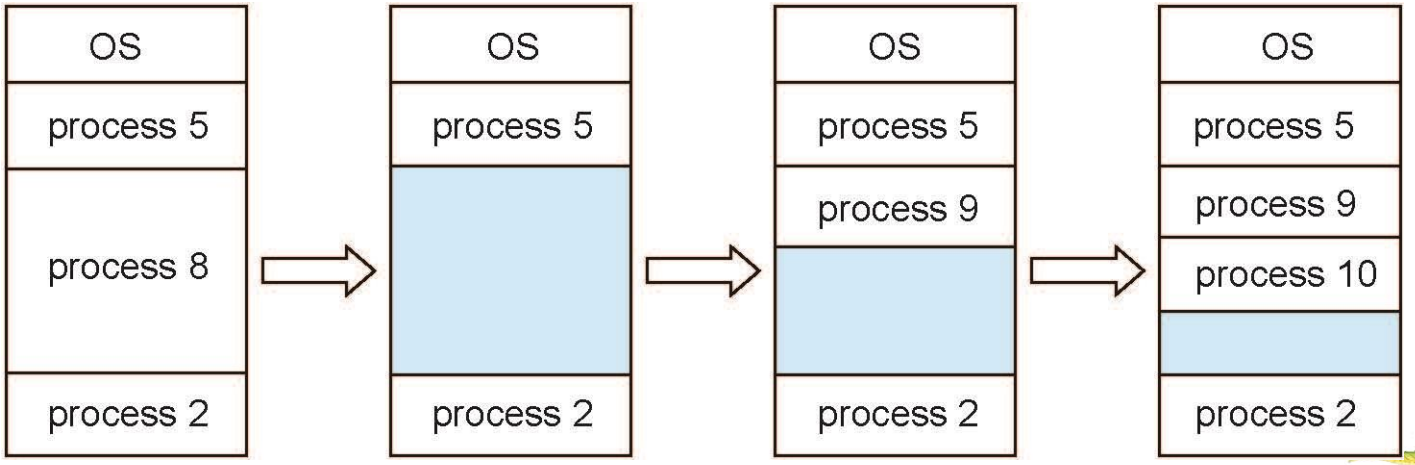


Techninis perkėlimo ir ribinių registrų palaikymas



Keleto dalių priskyrimas
(angl. Multiple-partition allocation)

- Multiprogramavimo laipsnis apribotas dalių skaičiaus.
- **Skirtingi dalių** dydžiai efektyvumui (dydis, pagal proceso poreikius)
- **Plyšys (Hole)** prieinamos atminties kiekis, įvairaus dydžio plyšiai išmėtyti atmintyje.
- Procesui atkeliavus, atmintis jam priskiriama iš pakankamai didelio plyšio.
- Procesui užbaigus darbą, jis atlaisvina atmintį, gretimos laisvos dalys apjungiamos.
- OS laiko informaciją apie
- a) priskirtas dalis b) laisvas dalis (plyšius)

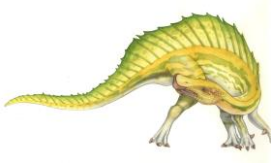


Dinaminė saugyklos priskyrimo problema
(angl. Dynamic Storage-Allocation Problem)

Kaip patenkinti ***n*** dydžio užklausą iš trijų plyšių sąrašo?

- **Pirmas tinkamas (First-fit)**: priskirti pirmą pakankamai didelį plyšį
- **Geriausias tinkamas (Best-fit)**: priskirti mažiausią pakankamai didelį plyšį; reikia peržiūrėti visą sąrašą, nebent išrikiuoti pagal dydį.
  - Gaunamas mažiausias likęs plyšys.
- **Blogiausias tinkamas (Worst-fit)**: priskirti didžiausią plyšį; taip pat reikia patikrinti visą sąrašą.
  - Gaunamas didžiausias likęs plyšys

Pirmas tinkamas ir geriausias tinkamas yra geresni, nei blogiausias tinkamas dėl greičio ir vietos išnaudojimo.



Fragmentavimas (angl. Fragmentation)

- **Išorinis fragmentavimas (angl. External Fragmentation)** – bendra atminties erdvė, tenkinanti užklausą, tačiau ne ištisinė.
- **Vidinis fragmentavimas (angl. Internal Fragmentation)** – priskirta atmintis gali būti šiek tiek didesnė nei užklausos; šis skirtumas yra atminties dalyje, tačiau nėra naudojama.
- Pirmas tinka analizė parodo, kad turint priskirtus N blokų, 0,5N blokų yra prarandami dėl fragmentavimo.
  - 1/3 gali būti nenaudojama -> **50-procentų taisyklė**

Fragmentavimas (tęs.)

- Sumažinamas išorinis fragmentavimas, naudojant glaudinimą.
  - Sumaišo atminties turinį, kad sutalpinti visą laisvą atmintį kartu viename dideliame bloke.
  - Glaudinimas įmanomas tik tuo atveju, jei perskirstymas yra dinamiškas ir yra atliekamas vykdymo metu.
  - I/O problemos:
    - Darbas užrakinamas atmintyje, kol jis susijęs su I/O.
    - Atlikti I/O tik OS buferiuose.
- Apsvarstykite, jei atsarginė saugykla turi tokių pačių fragmentavimo problemų.

Segmentavimas (angl. Segmentation)

- Atminties valdymo schema, kuri palaiko atminties vaizdą iš vartotojo pusės (user view of memory).
- Programa yra segmentų rinkinys.
  - Segmentas yra loginis vienetas, kaip:

    pagrindinė programa

    procedūra

    funkcija

    metodas

    objektas

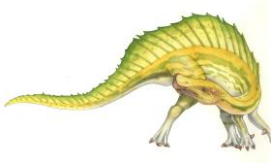
    lokalūs kintamieji, globalūs kintamieji

    bendri blokai

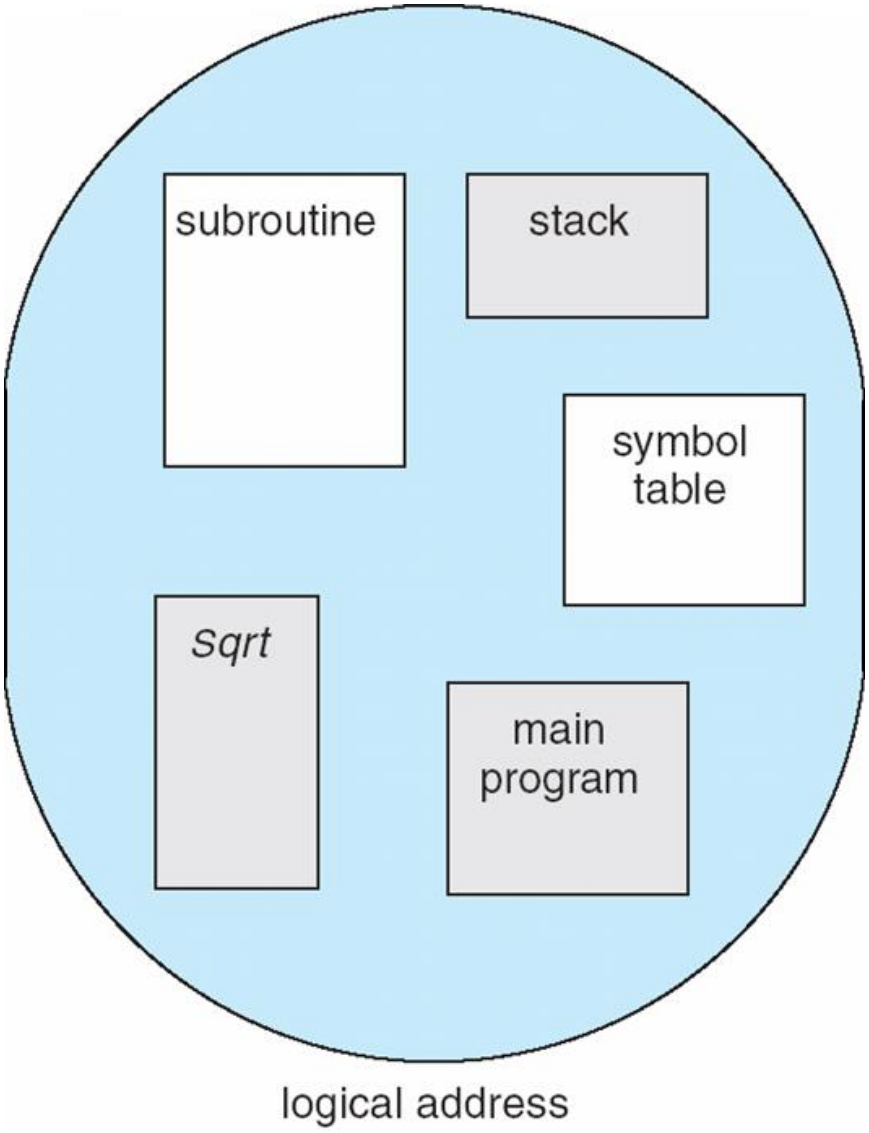
    stekas

    simbolių lentelė

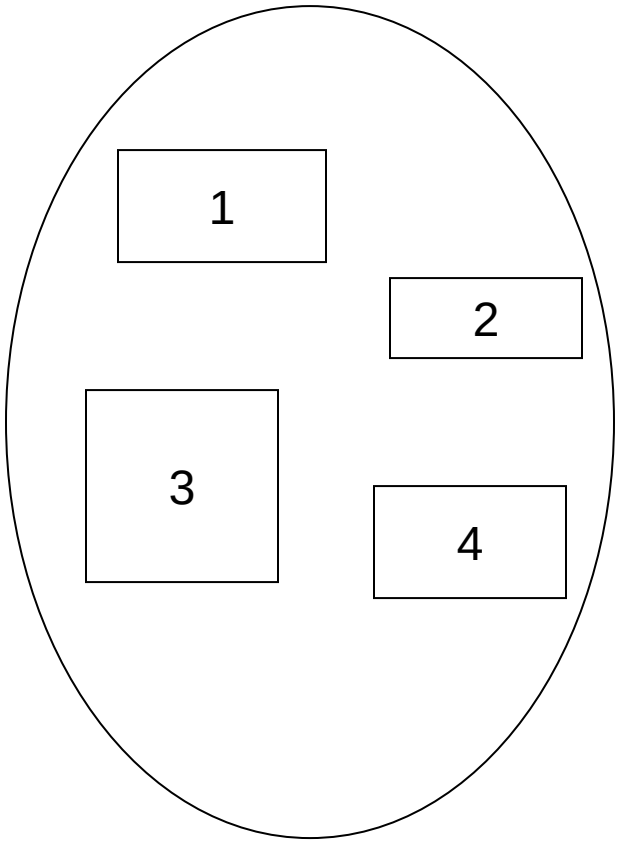
    masyvai



Atminties vaizdą iš vartotojo pusės



Loginis segmentavimo vaizdas



vartotojo erdvė

Fizinė atminties erdvė

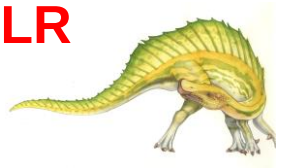
Segmentavimo architektūra

- Loginis adresas susideda iš dviejų elementų:

  <segment-number, offset>,

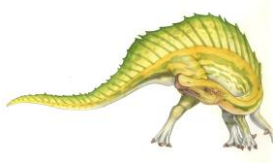
- **Segmentų lentelė (Segment table)** – susieja dviejų dimensijų fizinius adresus; kiekviena lentelė turi:
  - **bazę** – saugo pradinį fizinį adresą, kur segmentas yra atmintyje.
  - **ribą** – nurodo segmento ilgį.
- **Segmentų-lentelės bazinis registras (STBR)** nurodo segmento lentelės vietą atmintyje.
- **Segmentų lentelės ilgio registras (Segment-table length register (STLR))** nurodo segmentų kiekį, naudojamą programos

  segmento numberis ***s*** yra teisingas, jei ***s*** < **STLR**

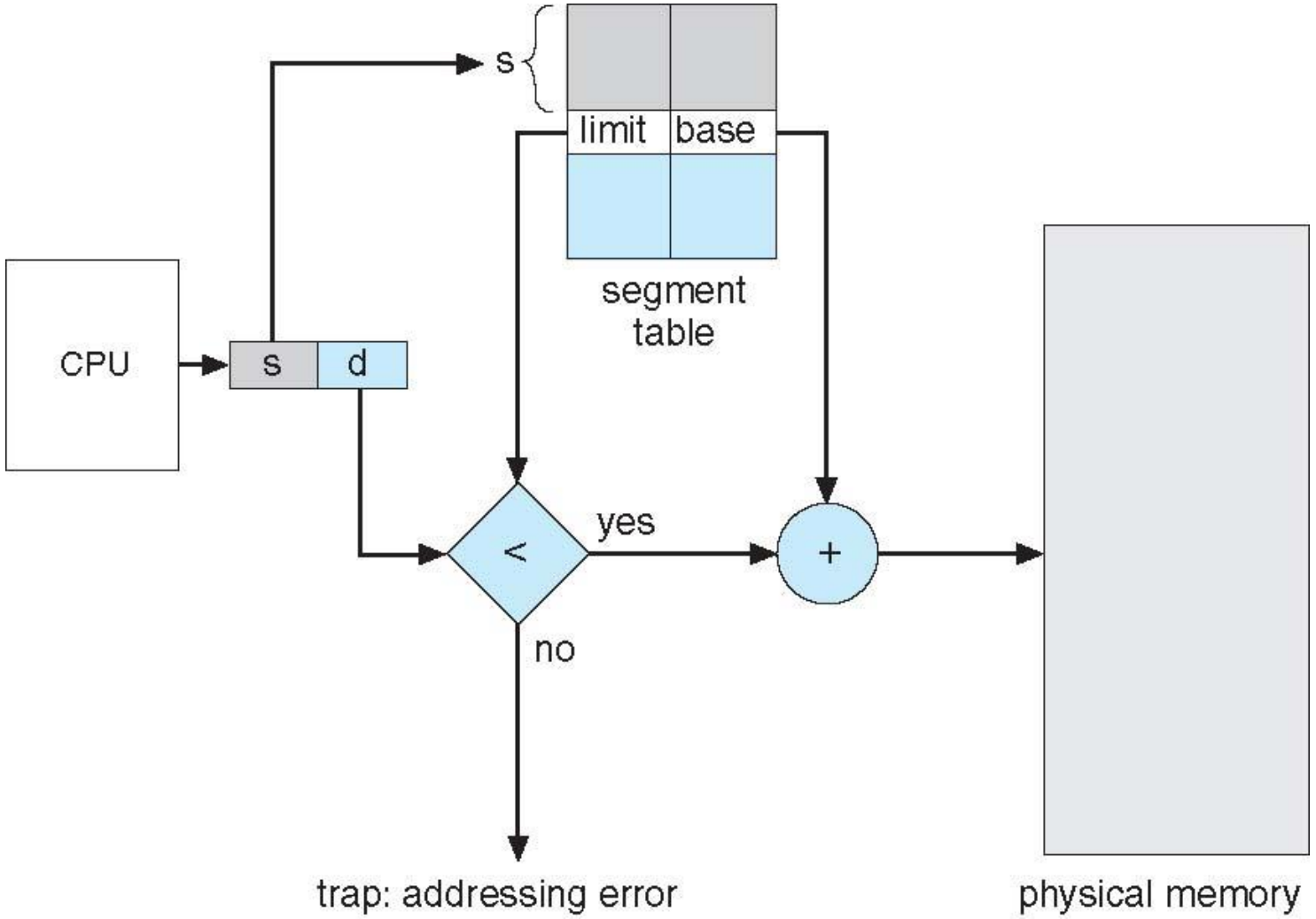


Segmentavimo architektūra (tęs.)

- Apsauga
  - Su kiekvienu įrašu segmentų lentelėje susieta:
    - validavimo bitas = 0 $\Rightarrow$ netinkamas segmentas
    - read/write/execute privilegijos
- Apsaugos bitai susieti su segmentais; kodo bendrinimas vyksta segmento lygmenyje.
- Kadangi segmentai skiriasi ilgiu, atminties priskyrimas yra dinaminis saugojimo-priskyrimo uždavinys.
- Segmentavimo pavyzdys pateiktas sekančioje diagramoje.



Segmentavimo techninė įranga

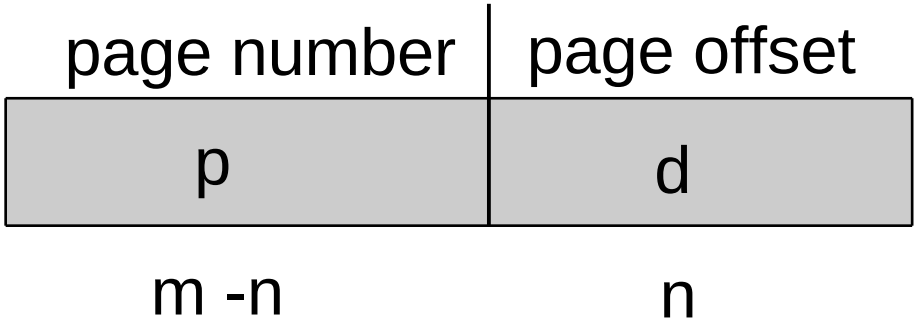


Puslapiavimas (angl. Paging)

- Fizinė proceso adresų erdvė gali būti neištisinė; procesui priskiriama ta fizinė atmintis, kuri yra prieinama:
  - Išvengiama išorinio fragmentavimo
  - Išvengiama skirtingo dydžio atminties dalių problemos.
- Fizinė atmintis padalijama į fiksuoto dydžio blokus, vadinamas kadrais (angl. **frames**).
  - Dydis yra 2 laipsnyje, tarp 512 bytes ir 16 Mbytes.
- Loginė atmintis padalijama į tokio paties dydžio blokus, vadinamus puslapiais (angl. **pages**).
- Sekami visi laisvi kadrai.
- Kad paleisti N puslapių dydžio programą, reikia surasti N laisvų kadrų ir paleisti programą.
- Sudaroma puslapių lentelė (angl. **page table**), kuri išverčia loginius adresus į fizinius.
- Backing store padalijama į puslapius.
- Išlieka vidinis fragmentavimas.

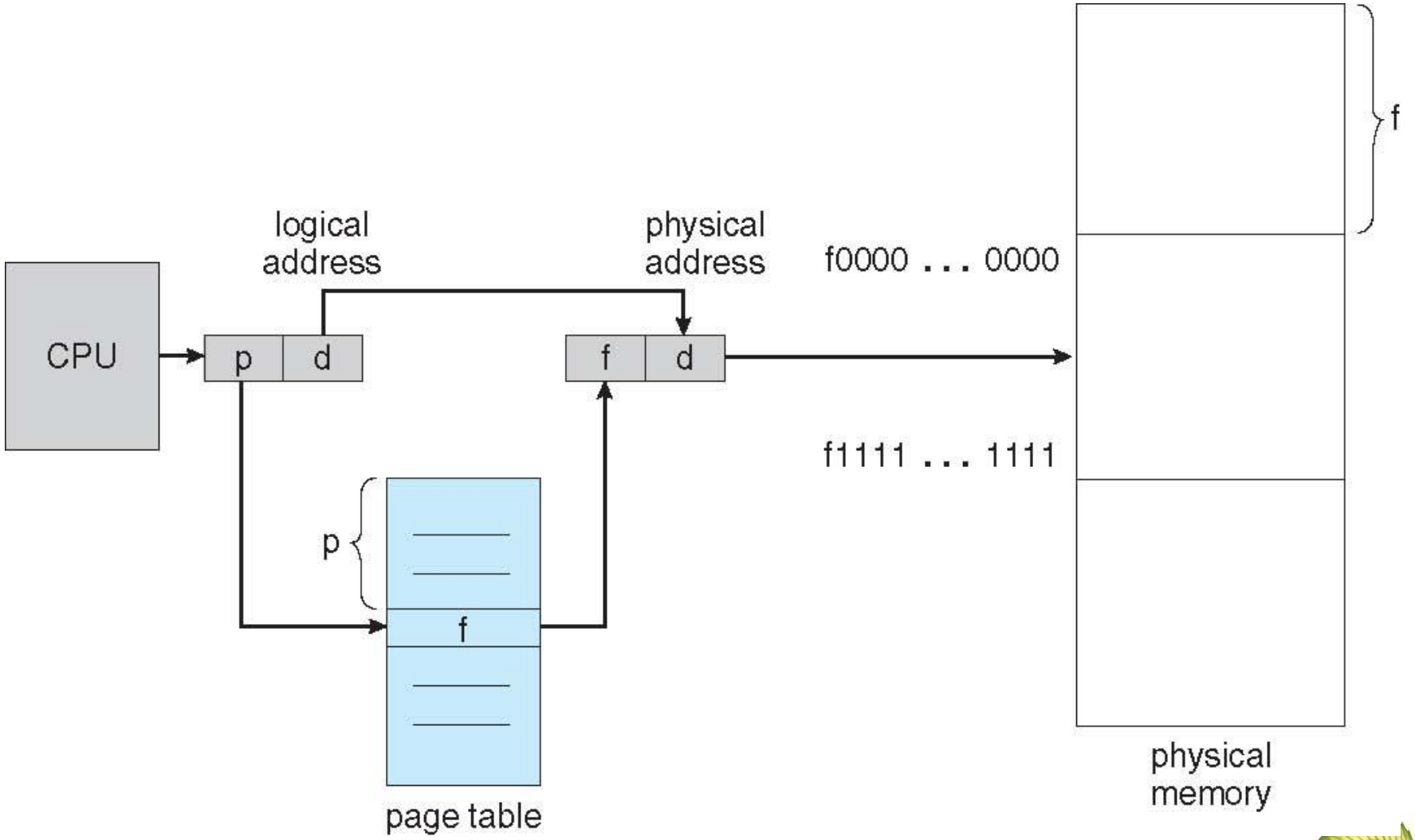
Adresų transliavimo schema
(angl. Address Translation Scheme)

- Adresai sugeneruoti CPU yra padalijami į:
  - **Puslapio numeris** (***p***) – naudojamas, kaip indeksas puslapių lentelėje (angl. **page table**), kuri turi bazinį kiekvieno puslapio adresą fizinėje atmintyje.
  - **Puslapio postūmis** (***d***) – apjungtas su baziniu adresu, kad aprašyti fizinį atminties adresą, kuris yra siunčiamas į atminties vienetą.

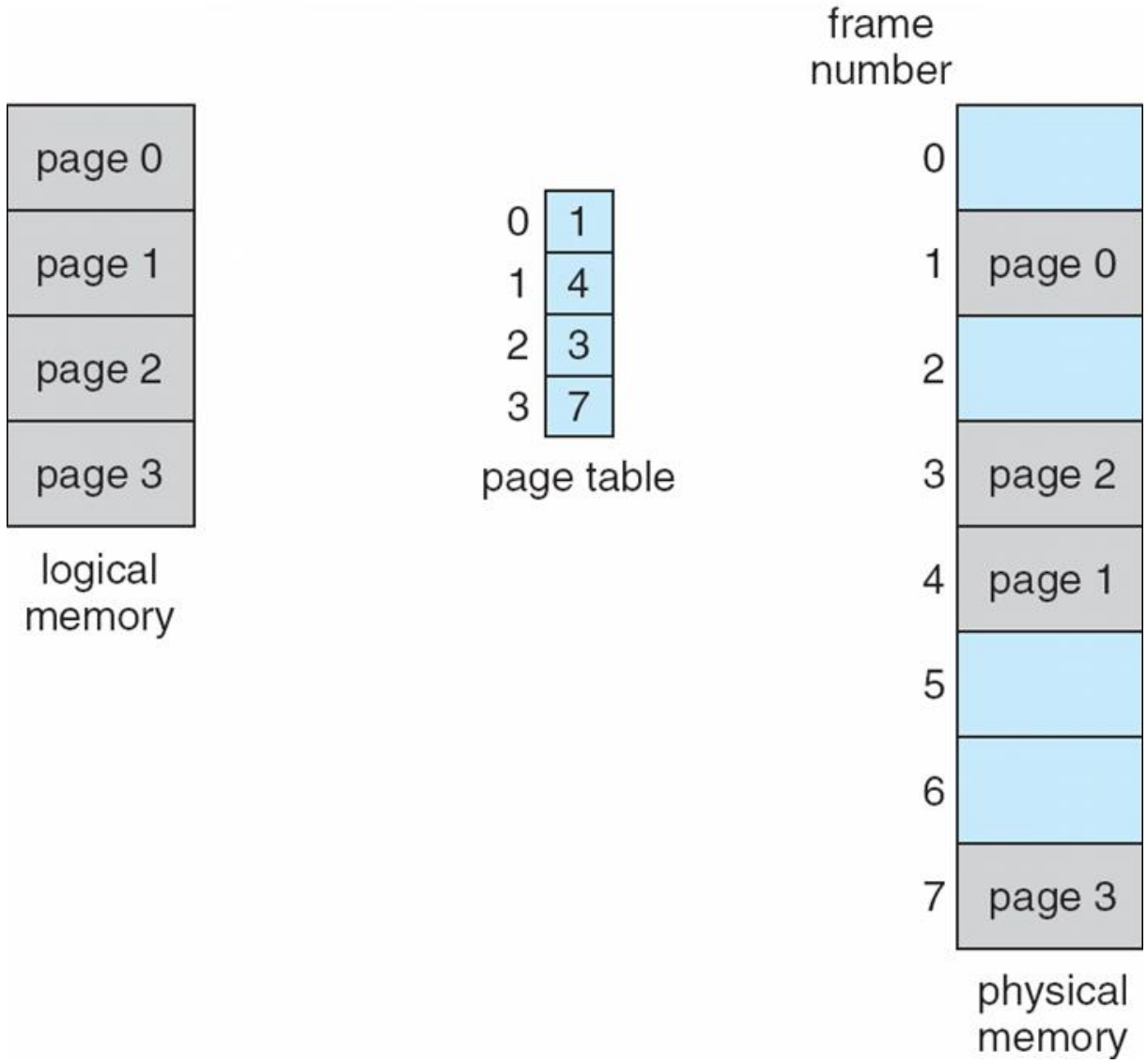


- Duotai loginių adresų erdvei 2^m ir puslapio dydžiui 2^n

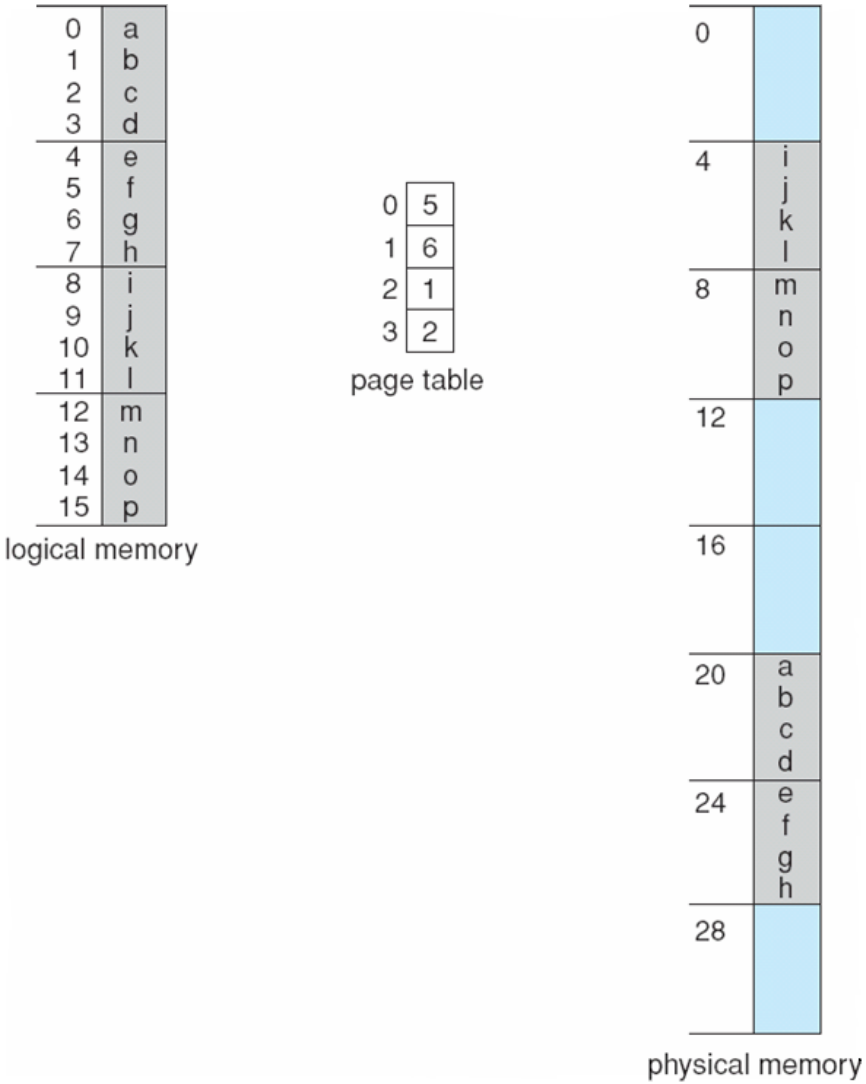
Puslapiavimo techninė įranga



Loginės ir fizinės atminties puslapiavimo modelis



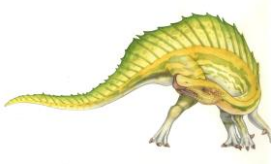
Puslapiavimo pavyzdys



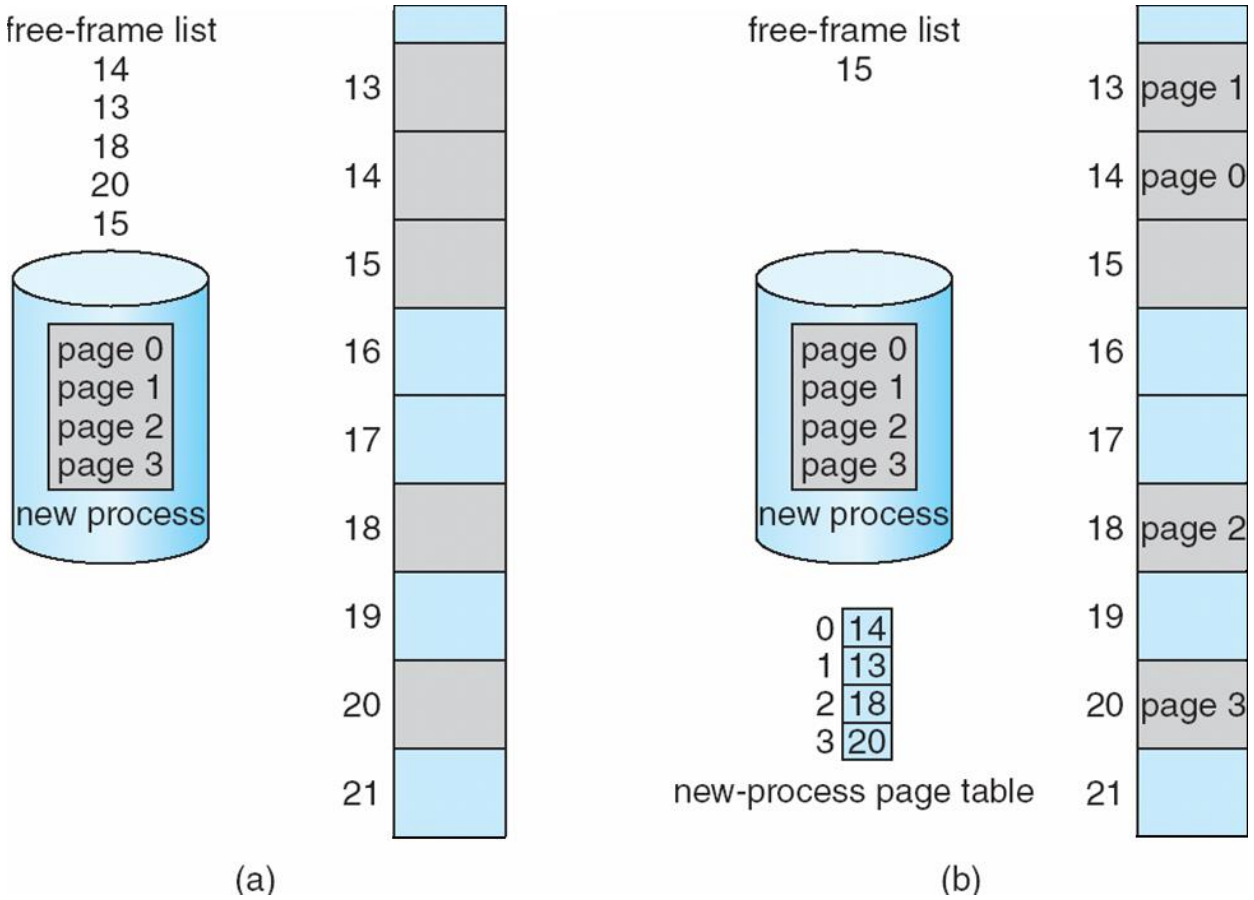
n=2 ir *m*=4 32-baitų atmintis ir 4-baitų puslapiai

Puslapiavimas (tęs.)

- Apskaičiuojamas vidinis fragmentavimas
  - Puslapio dydis = 2,048 bytes
  - Proceso dydis = 72,766 bytes
  - 35 puslapiai + 1,086 bytes
  - Vidinis fragmentavimas 2,048 - 1,086 = 962 bytes
  - Blogiausio atvejo fragmentavimas = 1 kadras – 1 byte
  - Vidutinis fragmentavimas = 1 / 2 kadro dydžio
  - Taigi mažas kadro dydis pageidaujamas?
  - Kiekvienas puslapio įrašas užima atmintį.
  - Puslapių dydis auga su laiku.
    - Solaris palaiko du puslapių dydžius – 8 KB ir 4 MB.
- Proceso vaizdas ir fizinė atmintis gali būti labai skirtingi.
- Pagal įgyvendinimą, procesas gali pasiekti tik savo atmintį.



Laisvi kadrai



Prieš priskyrimą

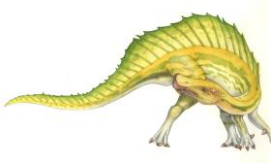
Po priskyrimo

Puslapių lentelės įgyvendinimas

- Puslapių lentelė yra saugoma pagrindinėje atmintyje.
- **Puslapių lentelės bazės registras (Page-table base register** (**PTBR**)) nurodo į puslapių lentelę.
- **Puslapių lentelės ilgio registras (angl. Page-table length register** (**PTLR**)) nurodo puslapių lentelės dydį.
- Šioje schemoje kiekvienas duomenų / instrukcijos pasiekimas reikalauja dviejų atminties prieigų.
  - Vienos puslapių lentelei ir vienos duomenims / instrukcijoms.
- Dviejų prieigų prie atminties problema gali būti išspręsta, naudojant specialių greitos peržiūros techninės įrangos kešą, vadinamą asociatyvia atmintimi (angl. **associative memory**) arba vertimo peržiūros iš šalies buferiais (angl. **translation look-aside buffers** (**TLBs**)).

Puslapių lentelės įgyvendinimas (tęs.)

- Kai kurie TLB saugo adresų erdvės identifikatorius (**address-space identifiers** (**ASIDs**)) kiekviename TLB įraše – unikaliai identifikuoja kiekvieną procesą, kad tam procesui suteiktų adresų erdvės apsaugą.
  - Kitu atveju, reikia išvalyti per kiekvieną konteksto perjungimą.
- TLB, paprastai, yra maži (64 - 1,024 įrašai)
- Jei TLB praleidžiamas, į TLB yra užkraunama vertė greitesnei prieigai sekantį kartą.
  - Turi būti nustatyta pakeitimo tvarka.
  - Kai kurie įrašai gali būti išsaugomi (**wired down**) nuolatinei greitesnei prieigai.

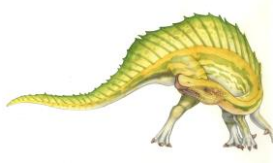


Asociatyvi atmintis
(angl. Associative Memory)

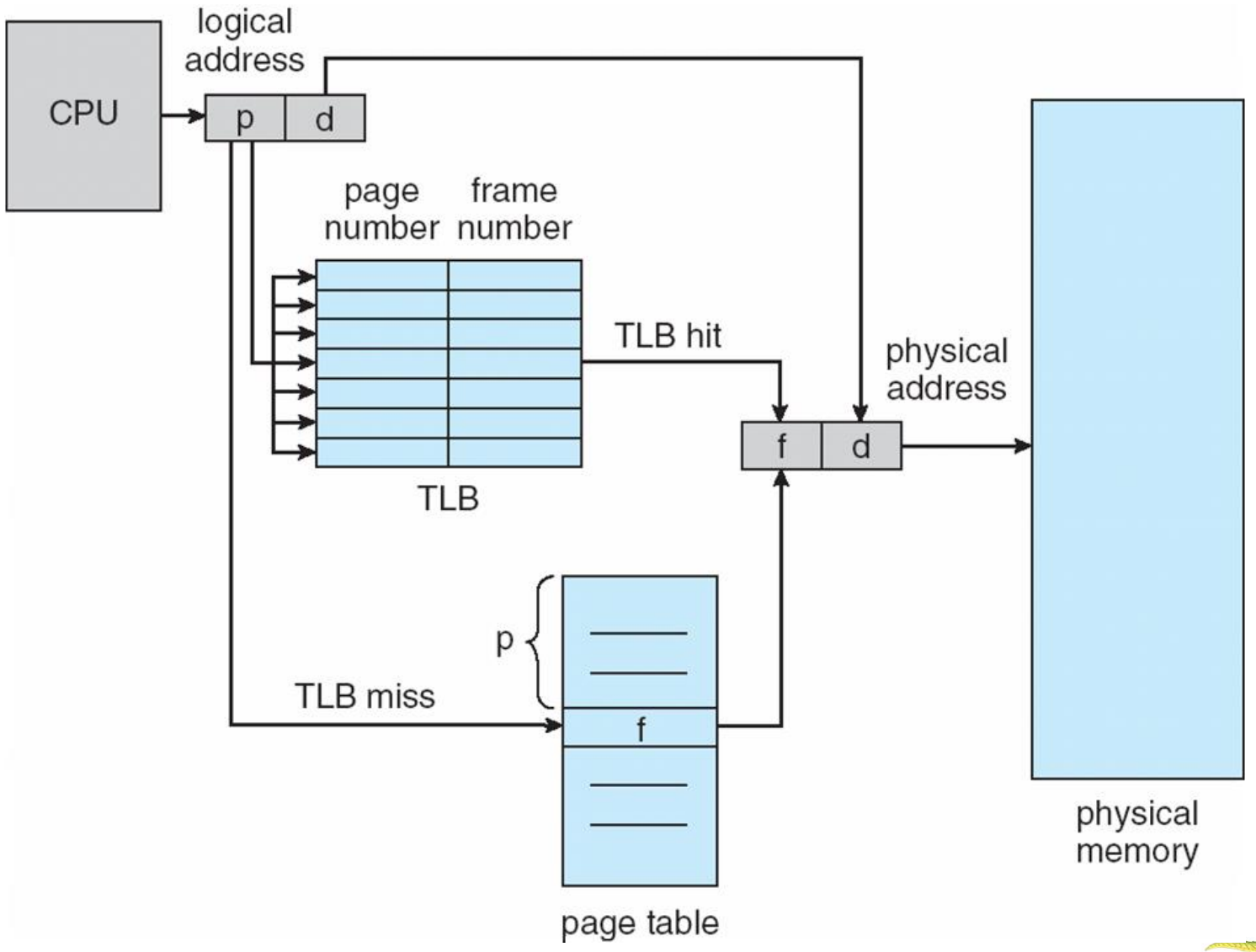
- Asociatyvi atmintis – lygiagreti paieška

Page #	Frame #

- Adresų transliavimas (p, d)
  - Jei p yra asociatyviame registre, gauti kadro #.
  - Kitu atveju, gauti kadro # iš puslapių lentelės atmintyje.



Puslapiavimo techninė įranga su TLB



Efektyvus prieigos laikas
(angl. Effective Access Time)

- Asociatyvi paieška = ε laiko vienetų
  - Gali būti < 10% atminties prieigos laiko
- Hit santykis (Hit ratio) = α
  - Hit ratio – procentas, kiek kartų puslapio numeris yra rasas asociatyviuose registruose; santykis susijęs su asociatyvių registrų skaičiumi.
- Pvz. α = 80%, ε = 20ns TLB paieškai, 100ns atminties prieigai
- **Efektyvus prieigos laikas (Effective Access Time (EAT))**

$$EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$$

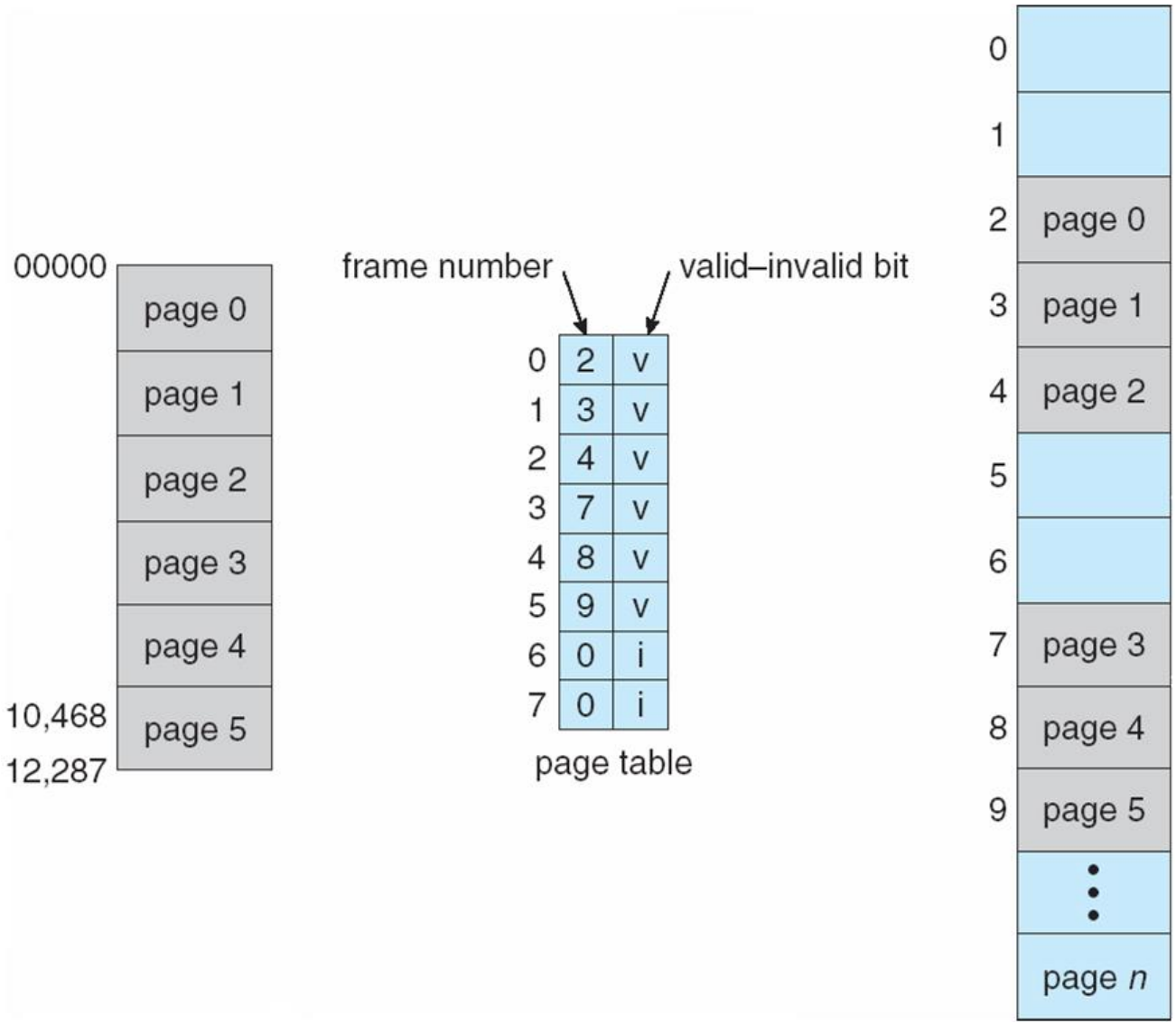
$$= 2 + \varepsilon - \alpha$$

- Pvz. α = 80%, ε = 20ns TLB paieškai, 100ns atminties prieigai
  - EAT = 0.80 x 100 + 0.20 x 200 = 120ns
- Pvz., realistiškesnis hit santykis -> α = 99%, ε = 20ns TLB paieškai, 100ns atminties prieigai
  - EAT = 0.99 x 100 + 0.01 x 200 = 101ns

Atminties apsauga
(angl. Memory Protection)

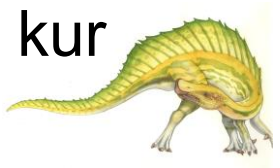
- Atminties apsauga įgyvendinama susiejant apsaugos bitą su kiekvienu kadru, kad indikuoti, jei leidžiama tik skaitymo ar skaitymo-rašymo prieiga.
  - Galima pridėti daugiau bitų, kad indikuoti tik vykdymas ir pan.
- **Galioja-negalioja (Valid-invalid)** bitas pridedamas prie kiekvieno puslapio lentelėje:
  - “valid” nurodo, kad susietas puslapis yra proceso loginėje adresų erdvėje ir yra galiojantis.
  - “invalid” nurodo, kad puslapis nėra proceso loginėje adresų erdvėje.
  - Arba naudojamas puslapio-lentelės ilgio registras (**page-table length register** (**PTLR**)).

Valid (v) ar Invalid (i) bitas puslapio lentelėje

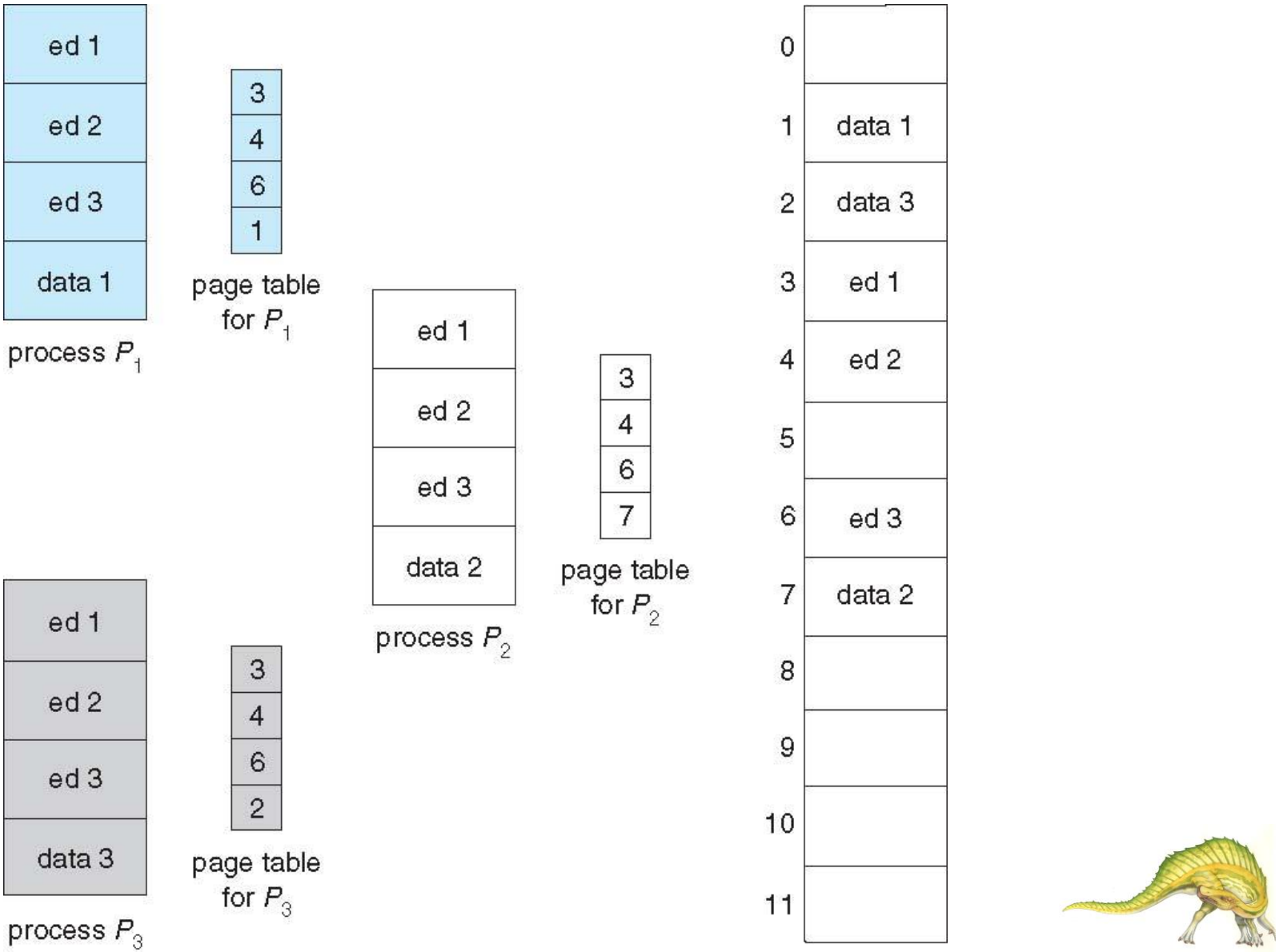


Bendrieji puslapiai (angl. Shared Pages)

- **Bendrasis kodas (Shared code)**
  - Viena tik skaitomo (**reentrant**) kodo kopija bendrinama tarp procesų (pvz. teksto redaktorių, kompiliatorių, langų sistemos).
  - Panašu į keletą gijų besidalijančių ta pačia proceso erdve.
  - Taip pat, naudinga tarpprocesinėje komunikacijoje, jei leidžiami skaitymo-rašymo puslapiai.
- **Privatus kodas ir duomenys (Private code and data)**
  - Kiekvienas procesas saugo atskirą kodo ir duomenų kopiją.
  - Privataus kodo ir duomenų puslapiai gali būti bet kur loginėje adresų erdvėje.

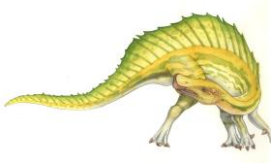


Bendrųjų puslapių pavyzdys



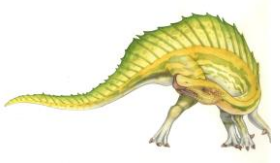
Puslapių lentelės struktūra

- Atminties puslapiavimo struktūros gali būti didelės, naudojant paprasčiausius metodus.
  - Pvz. 32-bitų loginių adresų erdvė kompiuteryje.
  - Puslapio dydis 4 KB (2^{12})
  - Puslapių lentelė turėtų 1 milijoną įrašų ($2^{32} / 2^{12}$).
  - Jei kiekvienas įrašas yra 4 baitų -> 4MB fizinės adresų erdvės / atminties vien lentelei.
    - Nenorima to priskirti ištisai pagrindinėje atmintyje

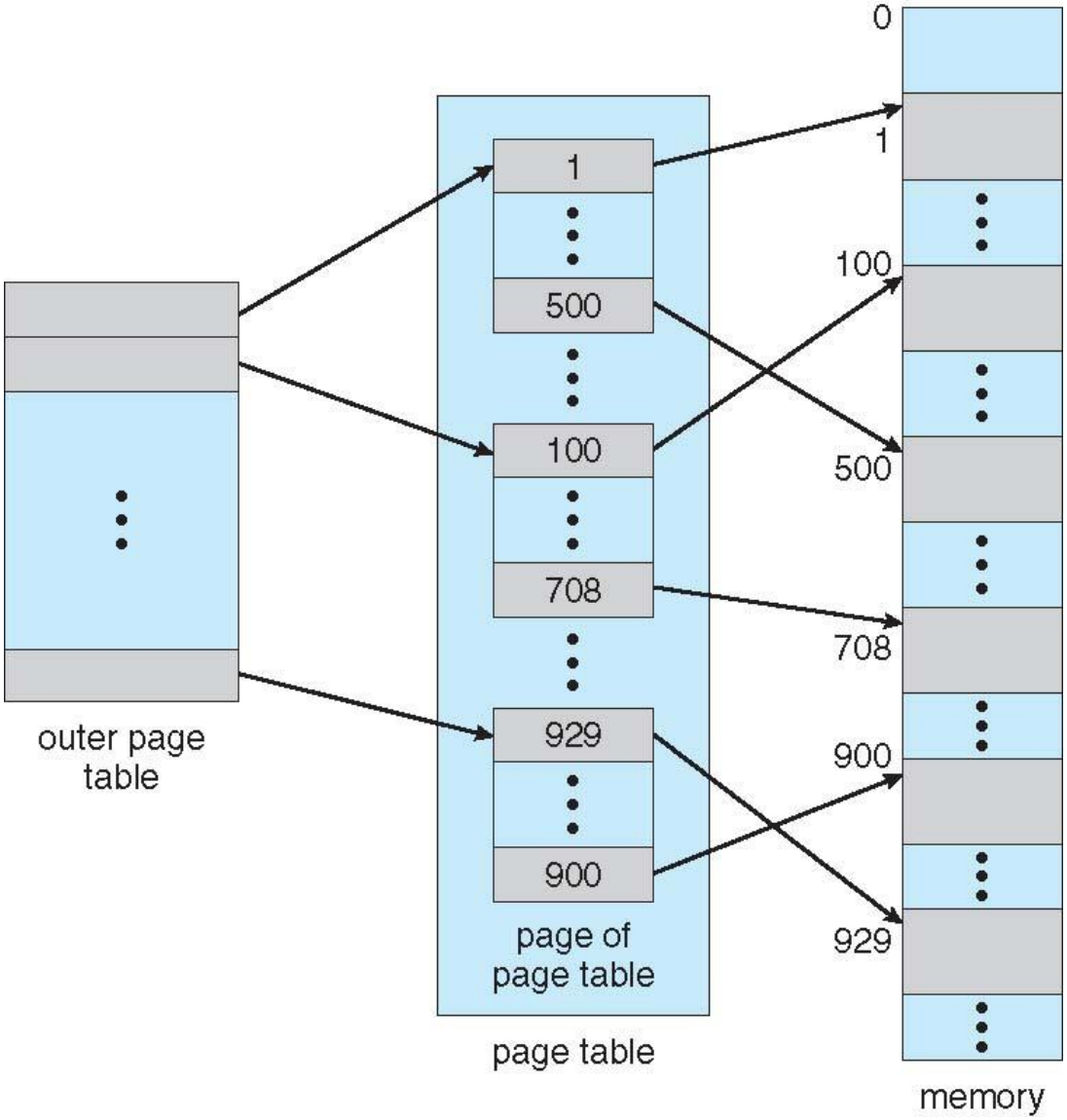


Hierarchinės puslapių lentelės (angl. Hierarchical Page Tables)

- Loginė adresų erdvė sudalijama į keletą puslapių lentelių.
- Paprastas būdas yra dviejų lygių puslapių lentelė.



Dviejų lygių puslapių lentelės schema



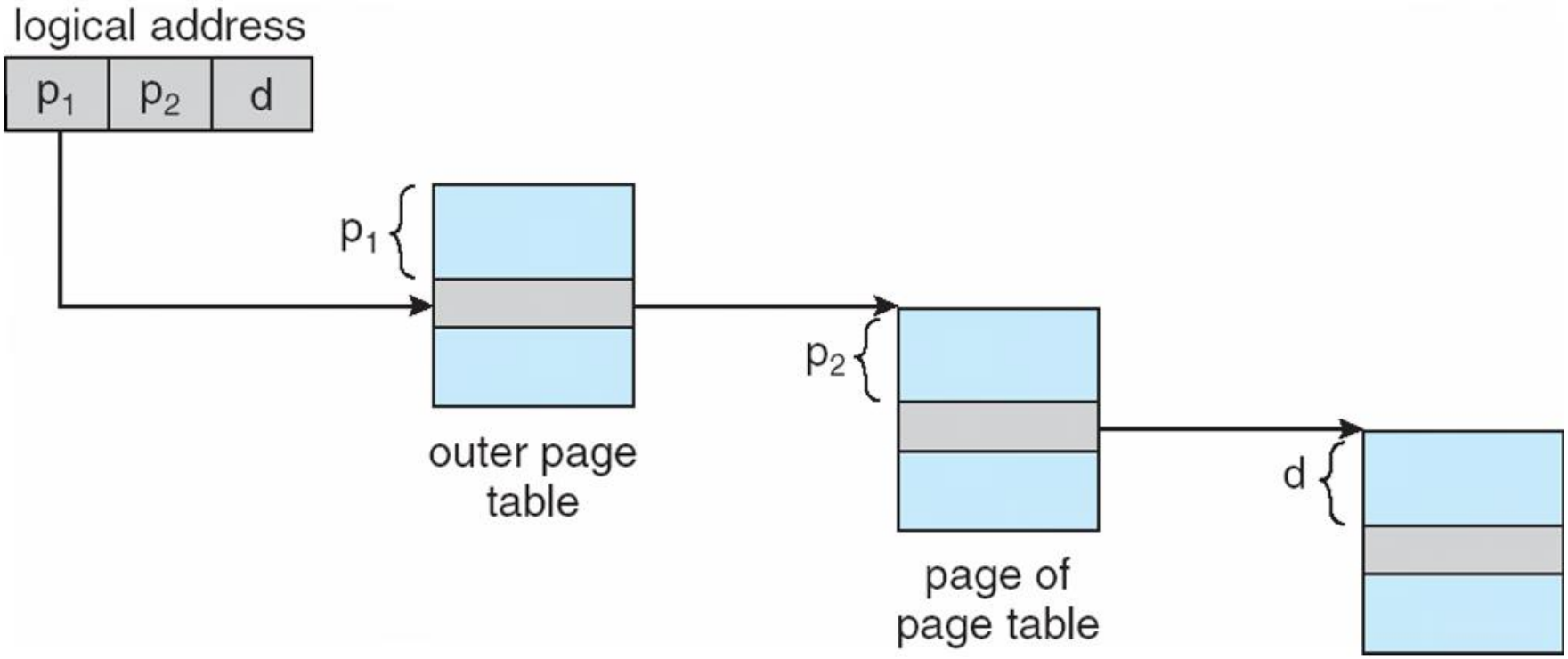
Dviejų lygių puslapiavimo pavyzdys

- Loginis adresas (32-bitų mašinoje su 1K puslapio dydžiu) yra padalintas į:
  - Puslapio numerį susidedantį iš 22 bitų.
  - Puslapio postūmį, susidedantį iš 10 bitų.
- Kadangi puslapių lentelė yra puslapiuota, puslapio numeris yra padalintas į:
  - 12-bitų puslapio numerį
  - 10-bitų puslapio postūmį
- Taigi loginis adresas:

page number		page offset
p_1	p_2	d
12	10	10

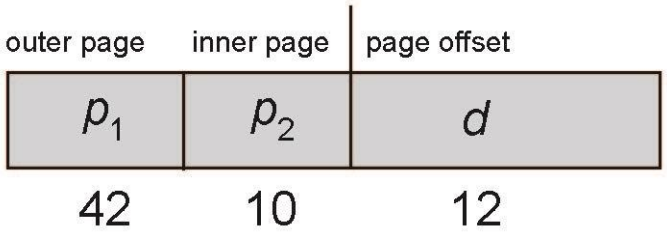
- kur p_1 yra nuoroda į išorinį puslapio numerį, o p_2 yra patalpinimas puslapyje vidinėje puslapių lentelėje.
- Dar žinoma, kaip tiesiogiai atvaizduojama puslapių lentelė (**forward-mapped page table)**

Adreso-translaivimo schema



64-bitų loginių adresų erdvė

- Net dviejų lygių puslapiavimo schemos nepakanka.
- Jei puslapio dydis yra 4 KB (2^{12})
  - Tuomet puslapio lentelė turi 2^{52} įrašus.
  - Jei dviejų lygių schema, vidinės puslapių lentelės galėtų būti 2^{10} 4-baitų įrašai.
  - Adresas atrodytų:

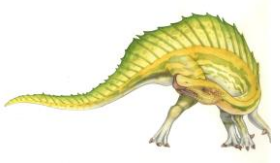


  - Išorinė puslapių lentelė turi 2^{42} įrašus arba 2^{44} baitus.
  - Vienas sprendimas - pridėti 2^{nd} išorinę puslapių lentelę.
  - Tačiau 2^{nd} išorinė puslapių lentelė vis tiek yra 2^{34} baitų dydžio.
    - Ir galimai 4 atminties prieigos, kad gauti vieną fizinę atminties vietą.

Trijų lygių puslapiavimo schema

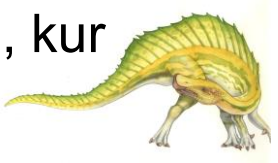
outer page	inner page	offset
p_1	p_2	d
42	10	12

2nd outer page	outer page	inner page	offset
p_1	p_2	p_3	d
32	10	10	12

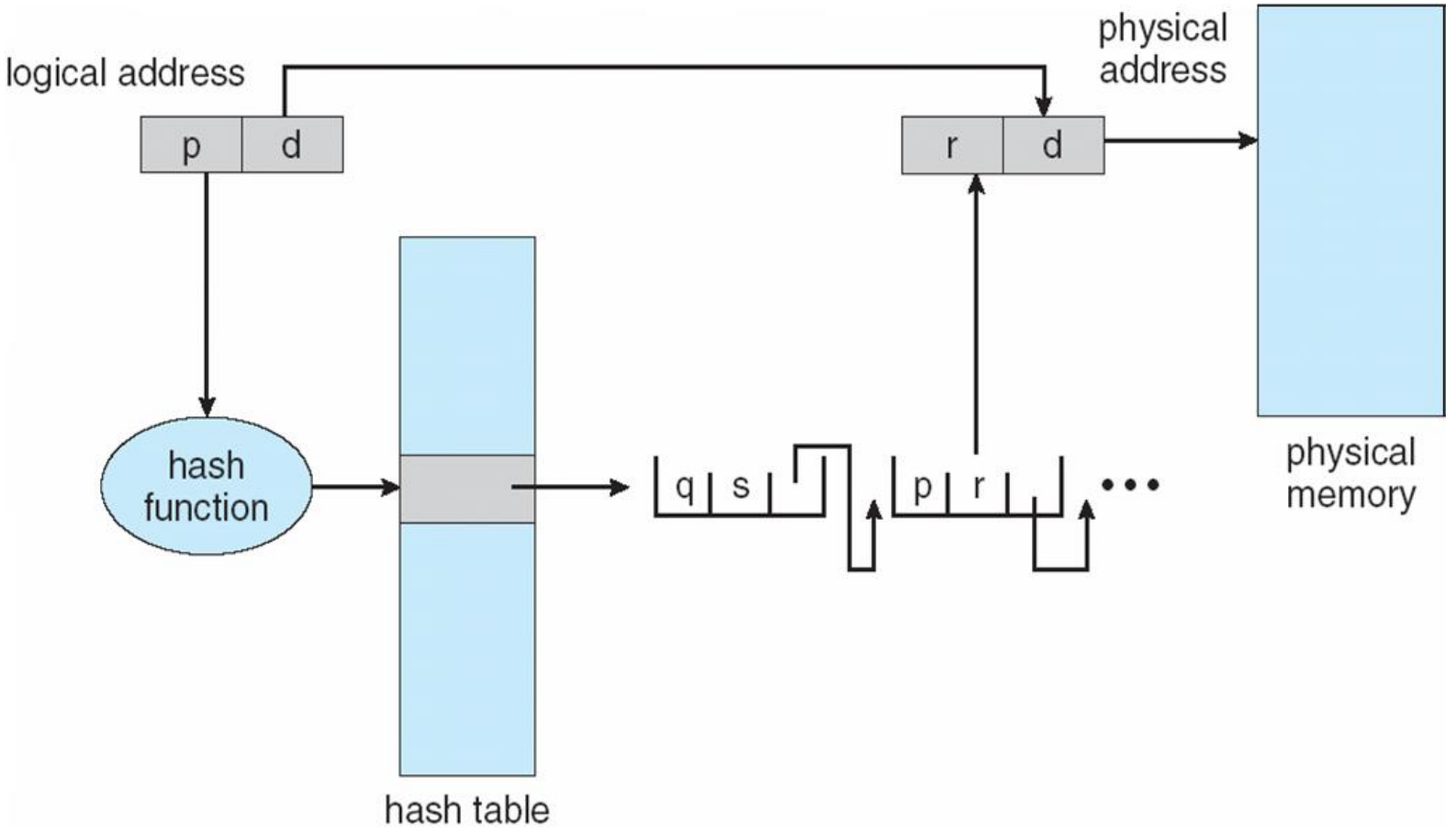


Sumaišytos puslapių lentelės
(angl. Hashed Page Tables)

- Paplitusios adresų erdvėse > 32 bits
- Virtualus puslapio numeris yra įmaišytas į puslapio lentelę.
  - Ši puslapio lentelė turi grandinę elementų sumaišytų toje pačioje vietoje.
- Kiekvienas elementas turi (1) virtualų puslapio numerį, (2) atvaizduoto puslapio kadro reikšmę, (3) rodyklę į sekantį elementą.
- Virtualūs puslapių numeriai yra palyginami šioje grandinėje, ieškant atitikmens.
  - Jei atitikmuo yra rastas, atitinkamas fizinis kadras yra „ištraukiamas“.
- 64 bitų adresų variacija yra klasterizuotos puslapių lentelės (angl. **clustered page tables**)
  - Panašios į sumaišytas, tačiau kiekvienas įrašas nurodo į keletą puslapių (pvz. 16 vietoje 1).
  - Ypač naudingas išsklaidytoje (angl. sparse) adresų erdvėje, kur atminties nuorodos yra nevientisos ir išbarstytos



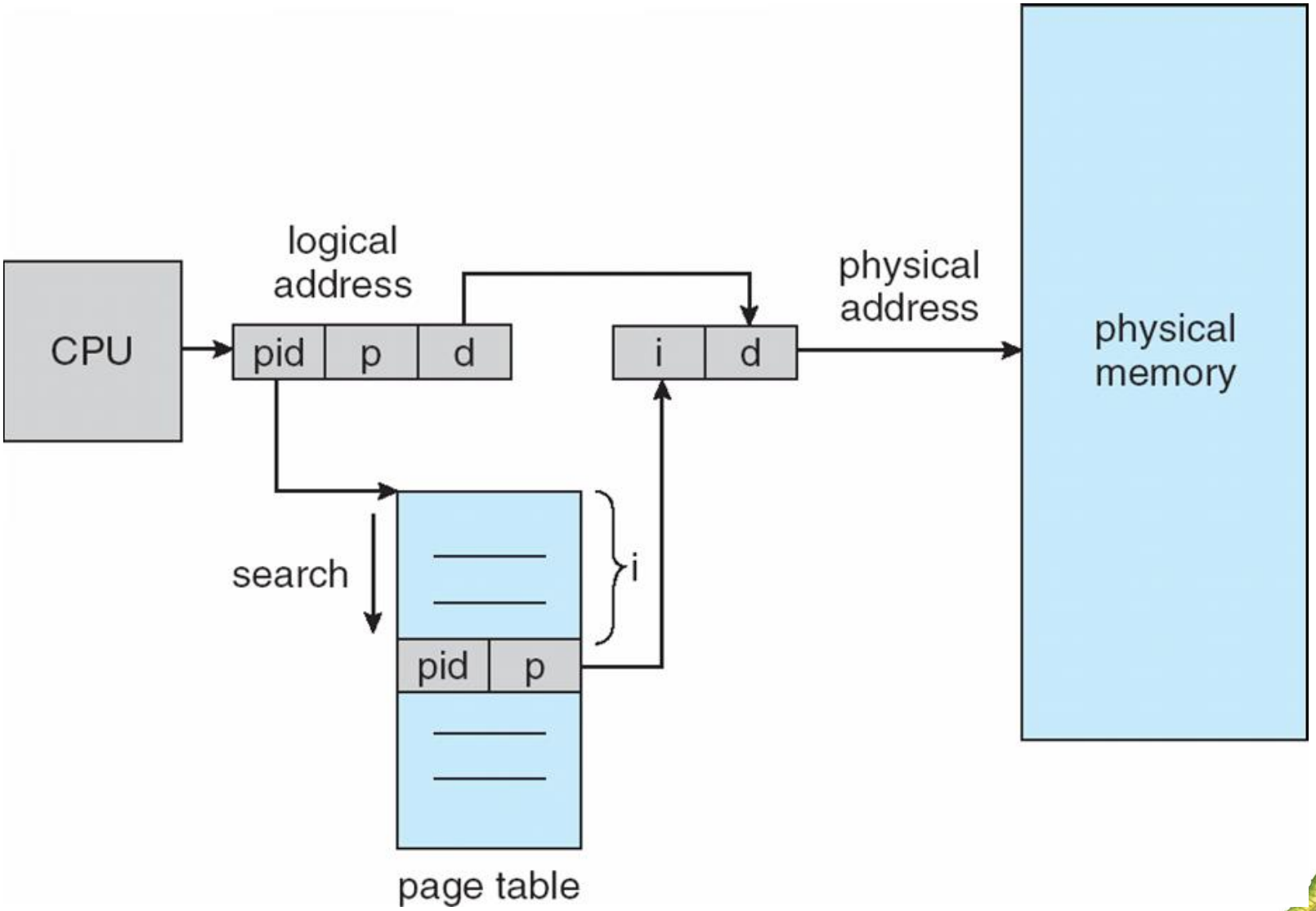
Sumaišytos puslapių lentelės



Invertuota puslapio lentelė

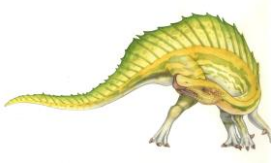
- Vietoje to, kad kiekvienas procesas turėtų puslapių lentelę ir sektų visus galimus loginius puslapius, sekami visi fiziniai puslapiai.
- Vienas įrašas vienam realiam atminties puslapiui.
- Įrašas susideda iš virtualaus puslapio adreso, saugomo toje atminties vietoje su informacija apie procesą, kuriam priklauso tas puslapis.
- Sumažinamas reikalingos atminties kiekis saugoti kiekvienai puslapio lentelei, tačiau padidėja laikas, reikalingas paieškai, kai gaunama puslapio nuoroda.
- Naudojamos maišos lentelės, kad apriboti paiešką iki vieno ar daugiausiai kelių galimų puslapio-lentelės įrašų.
  - TLB gali pagreitinti prieigą.
- Tačiau kaip įgyvendinti su bendrąja atmintimi?
  - Vienas virtualaus adreso atvaizdavimas į bendrą fizinį adresą.

Invertuotos puslapio lentelės architektūra



Pavyzdžiai: Intel 32 ir 64-bit architektūros

- Dominuojančios schemos
- Pentium 32-bitų vadinama IA-32 architektūra
- 64-bitų vadinama IA-64 architektūra
- Daug variacijų tarp čipų, apžvelgiamos pagrindinės idėjos.



Pavyzdžiai: Intel 32 ir 64-bit architektūros

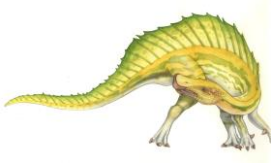
- Palaiko tiek segmentavimą, tiek segmentavimą su puslapiavimu.
  - Kiekvienas segmentas gali būti 4 GB.
  - Iki 16 K segmentų procesui
  - Paskirstyti į dvi dalis:
    - Pirma dalis iki 8 K segmentų yra privatūs procesui (laikomi **local descriptor table** (**LDT**))
    - Antra dalis iki 8 K segmentų bendrų tarp visų procesų (laikomi **global descriptor table** (**GDT**))

Pavyzdžiai: Intel 32 ir 64-bit architektūros (tęs.)

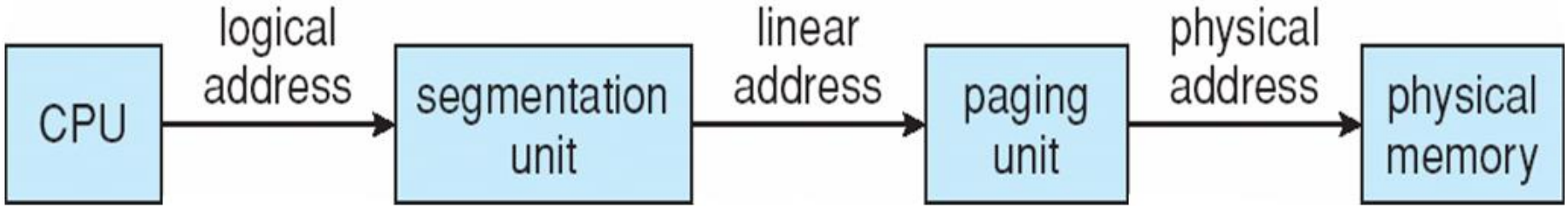
- CPU generuoja loginius adresus
  - Pasirinkimas duodamas segmentavimo įrenginiui
    - Kuris sudaro tiesinius adresus

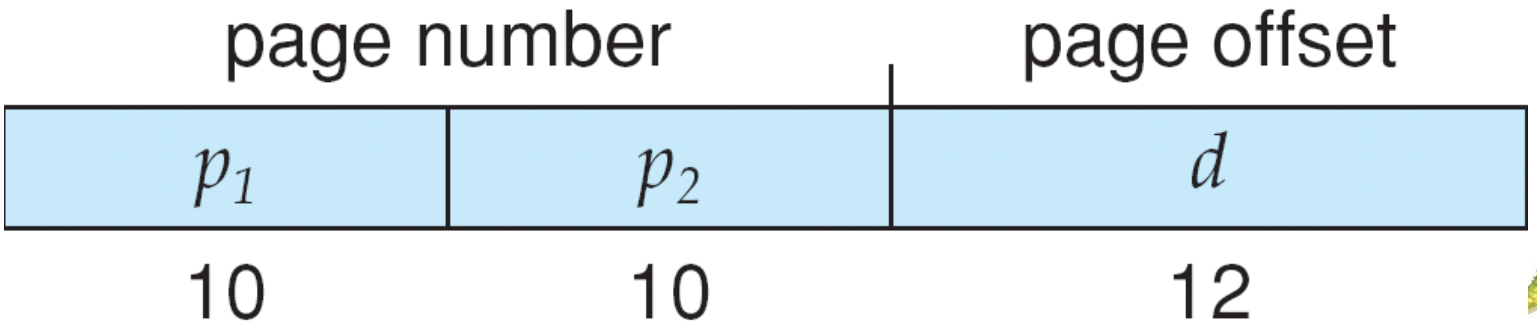
s	g	p
13	1	2

  - Tiesiniai adresai duodami puslapiavimo įrenginiui
    - Kuris sugeneruoja fizinius adresus į pagrindinę atmintį
    - Puslapiavimo įrenginiai yra MMU atitikmuo
    - Puslapių dydžiai gali būti 4 KB or 4 MB

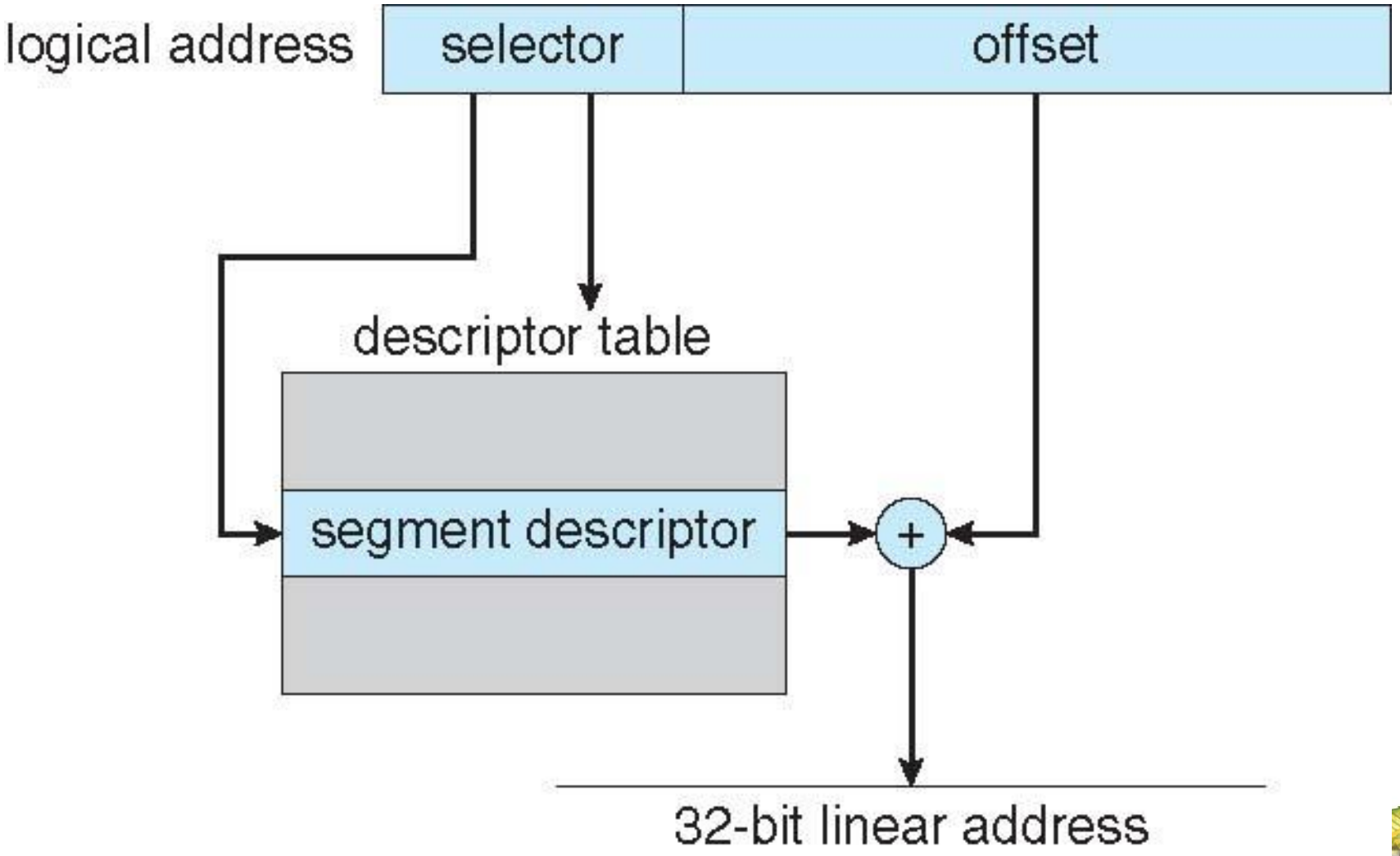


Loginių adresų išvertimas į fizinius IA-32

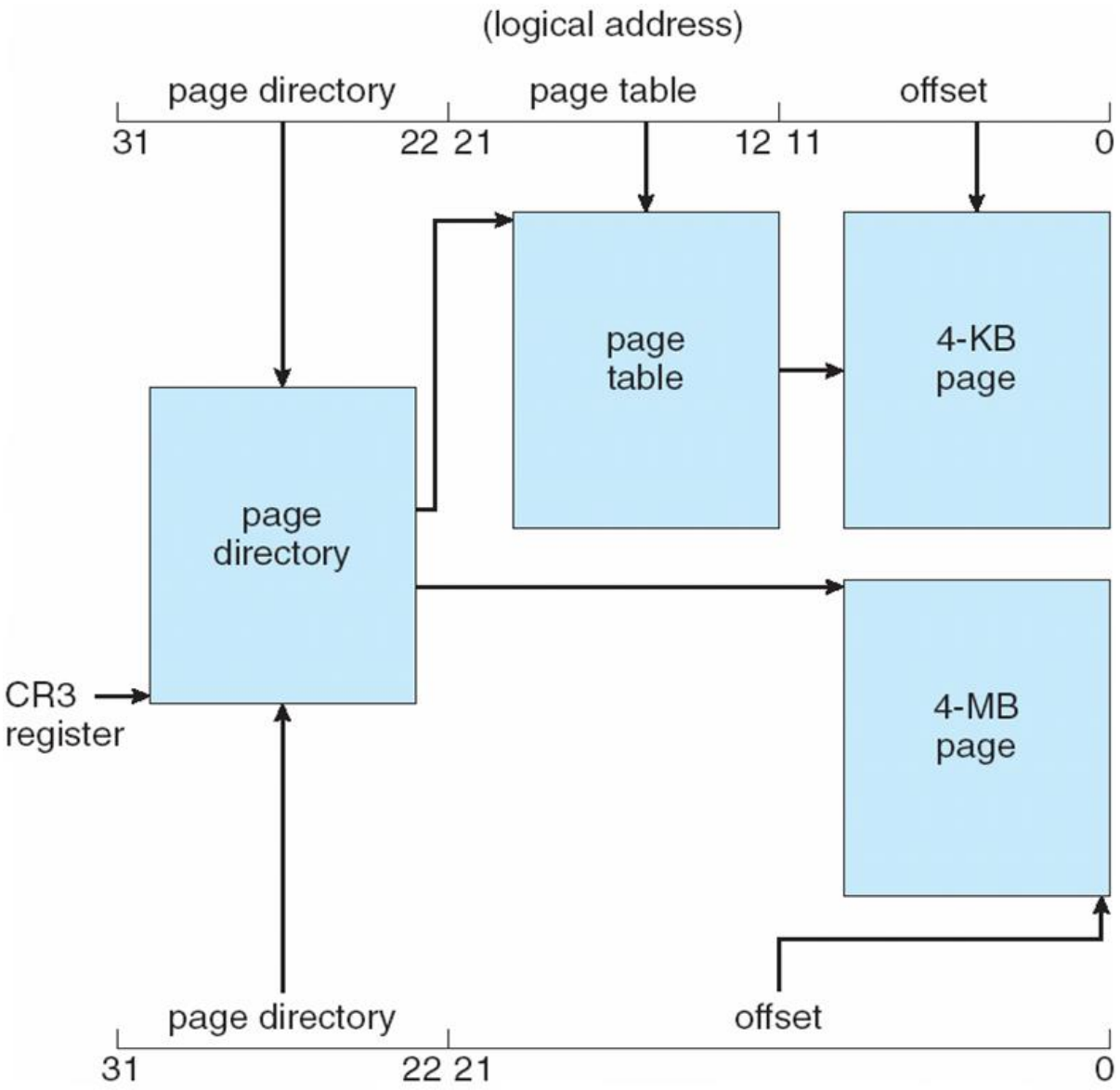




Intel IA-32 segmentavimas

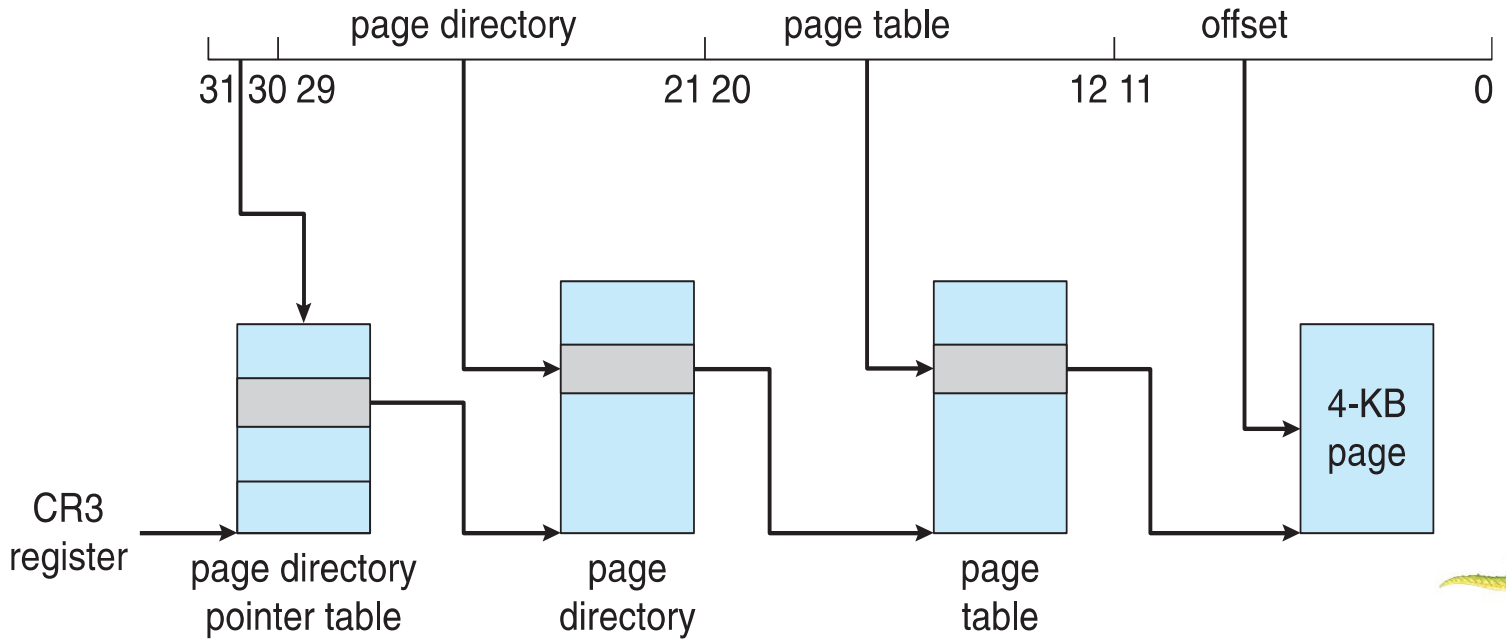


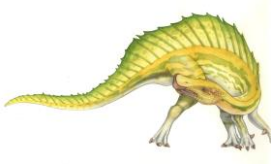
Intel IA-32 puslapiavimo architektūra



Intel IA-32 puslapių adresų praplėtimai (Page Address Extensions)

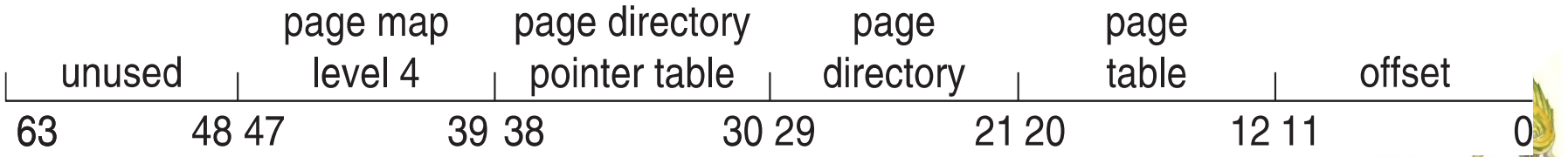
- 32-bitų adresų ribos lėmė, kad Intel sukūrė **page address extension** (**PAE**), leidžiančius 32-bitų programoms pasiekti daugiau nei 4GB atminties.
  - Puslapiavimas remiasi 3-lygių schema
  - Viršutiniai du bitai nurodo į puslapių katalogo rodyklės lentelę (**page directory pointer table)**
  - Puslapių katalogas ir puslapių lentelės įrašai perkelti į 64 bitų dydį.
  - Adresų erdvė išplečiama iki 36 bitų – 64GB fizinės atminties





Intel x86-64

- Dabartinė Intel x86 architektūros karta
- > 16 exabytes
- Praktiškai reikalingas tik 48 bitų adresavimas
  - Puslapių dydžiai 4 KB, 2 MB, 1 GB
  - Keturių lygių puslapiavimo hierarchija
- Gali naudoti PAE, taigi virtualūs adresai yra 48 bitų, o fiziniai 52 bitų.



Pavyzdys: ARM architektūra

- Dominuojanti mobilios platformos architektūra (Apple iOS ir Google Android)
- Moderni, energetiškai efektyvi, 32-bitų ir 64-bitų CPU.
- 4 KB ir 16 KB puslapiai
- 1 MB ir 16 MB puslapiai (vadinami **sections**)
- Vieno lygmens puslapiavimas sekcijoms, dviejų lygių – mažesniems puslapiams.
- Dviejų lygių TLB
  - Išorinis lygis turi du mikro TLB (duomenims ir instrukcijoms)
  - Vidinis yra pagrindinis TLB
  - Pirmiausia tikrinamas vidinis, jei nerandama, tikrinamas išorinis