

LUENTO 8

# VIRTUAALIMUISTI

## Stallings, Luku 8.1

1

## Sisältö

- Ohjelman suoritus virtuaalimuistissa
- Sivutus ja sivutaulut
- Osoitemuunnospuskuri TLB
  
- Segmentointi
- Segmentointi ja sivutus yhdistettynä
- Yhteiskäytöstä

2

## Suoritus virtuaalimuistissa

- Ohjelman loogiset osoitteet muutetaan fyysisiksi osoitteiksi vasta ajonaikana
  - prosessin paikka muistissa vaihtelee, sillä ei vaikutusta osoitemuunnokseen
  - MMU
- KJ käsittelee ohjelmaa sivuina tai kääntäjä jakaa ohjelman segmentteihin, jotka KJ voi sijoitella vapaasti muistiin
  - KJ:n kirjanpito osien sijainnista prosessin sivutaulussa tai segmenttitaulussa

3

## Suoritus virtuaalimuistissa

- Kaikkien sivujen / segmenttien ei tarvitse olla muistissa yhtäaikaan
  - riittää, että suoritettava osa ja sen data muistissa
  - paikallisuus, suoritus viihtyy samoilla alueilla
  - Kirjan kuva 8.1 näyttää muistivitteiden jakautumista
- Laitteisto (MMU) ja KJ huolehtivat, että tarvittavat osat oikeaan aikaan muistissa
  - MMU huomaa puuttumisen
  - KJ noutaa muistiin

= Virtuaalimuisti

4

## Suoritus virtuaalimuistissa

- Sivut / segmenttitaulun alkiossa läsnäolobitti, josta käy ilmi onko sivu / segmentti muistissa
- Lataaja tuo aluksi muistiin vain muutaman sivun / segmentin (ennaltanouto) tai ei yhtään (tarvenouto)
- Prosessi CPU:lle  $\bar{O}$ 
  - MMU:hun sivu- / segmenttitaulun fyysinen muistiosoite
  - TLB:n sisällön mitätöinti

Paikallisuuden vaikutus

5

## Puutoskeskeytys (memory / page fault)

- Jos viitattu osoite ei ole muistissa, MMU aiheuttaa keskeytyksen
- KJ siirtää keskeytyksen aiheuttaneen prosessin (A) Blocked-tilaan
- KJ etsii sivulle / segmentille vapaan paikan muistista
- KJ käynnistää ohjaimen siirtämään puuttuvaa sivua / segmenttia ko. paikkaan
- Siirron aikana CPU suorittaa muita prosesseja

6

## Puutoskeskeytykset

- Kun siirto valmis, ohjain keskeyttää suorituksessa olevan prosessin B
- KJ päivittää prosessin A sivu/segmenttitaulun ja siirtää prosessin A Ready-tilaan
- Suoritus palaa takaisin prosessiin B
- Kun prosessi A taas aikanaan suoritukseen, se viittaa uudestaan äskeiseen osoitteeseen
  - nyt viitattu mp:n sisältö muistissa

7

## Virtuaalimuistin etuja

- Muistia käytetään tehokkaammin hyväksi
  - kustakin prosessista vain tarvittava osa muistissa
  - montako sivua/prosessi pidetään muistissa yhtäaikaan?
    - lokaalit ja globaalit algoritmit
- Prosessoria käytetään (kenties) tehokkaammin
  - moniajoastetta voi nostaa
    - muistiin mahtuu paremmin, ei ruuhkautumista

8

## Virtuaalimuistin etuja

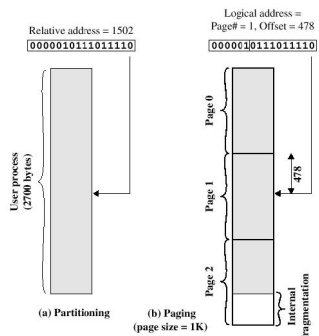
- Voi ajaa fyysistä muistia suurempia ohjelmia
  - ohjelmoijan ei tarvitse huolehtia kerrostuksesta
- Osoiteavaruus voi olla valtaisa verrattuna todelliseen muistin määrään
  - esim. 32 bittiä => 4GB:n osoiteavaruus
  - hyöty?
- Looginen osoiteavaruus saa sisältää 'reikiäkin'
  - vain tarvittavat osat kuvataan fyysiseen muistiin

9

## SIVUTAULU

10

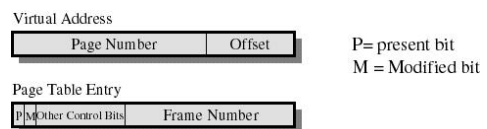
## Sivutus



11

## Sivutaulu

Kuva 8.2a



- Jokaisella prosessilla oma sivutaulu
  - sivutaulun fyysinen alkuosoite PCB:ssä, taulu muistissa
  - Alkioissa: missä sivutiloissa tämän prosessin sivut sijaitsevat
- Jokaisessa alkiossa läsnäolobitti P
  - P=1: sivu muistissa, alkiossa sivutilan numero
  - P=0: sivu ei muistissa, alkiossa esim. tieto missä sivu sijaitsee tukimuistissa (suoraan/epäsuorasti)

12

## Sivutaulu

- Jokaisessa sivutaulun alkiossa muutettu-bitti M (modified)
  - M=1: sivun sisältö muuttunut muistissa, sivu kirjoitettava levyille, jos varaus vapautetaan
  - M=0: sivua ei muutettu, ei tarvitse kirjoittaa levyille sivutilaa vapautettaessa
- Sivutaulun alkiossa mahd. myös muuta tietoa
  - käyttötapabitti: R / RW
  - suojautasobitti/bitit: KJ:n sivu / tav. prosessin sivu
  - milloin sivuun viitattu viimeksi tai viitelaskuri
    - poistoalgoritmit tarvitsevat näitä

13

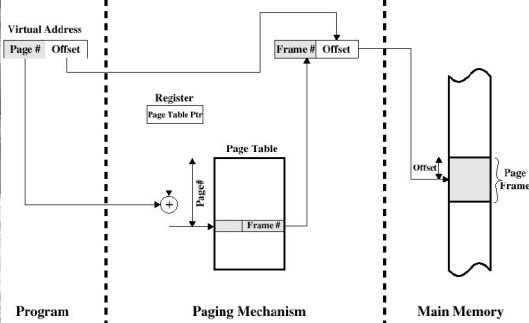
## Osoitemuunnos

- MMU:ssa sivutaulurekisteri PTR, jossa vain suoritettavan prosessin sivutaulun fyysinen alkuosoite
- MMU jakaa loogisen osoitteen pariaksi (sivunro,siirtymä)
  - Esim. kun sivukoko 1024 B ( $= 2^{10}$ )
    - 10 viimeistä bittiä siirtymä
    - alkuosa sivunumero
- MMU korvaa sivunumero-bitit sivutaulusta löytyvillä sivutilannumero-biteillä
  - ts. MMU katenoit sivutilanumeron ja siirtymän bitit
  - Helppo laiteointo

14

## Osoitemuunnos

Kuva 8.3



15

## Lisää sivutauluista

16

## Lisää sivutauluista

- Monet järjestelmät sallivat suuren virtuaaliosoitevaruuden
  - looginen osoite esim. 32 tai 64 bittiä
- Jokaisella prosessilla suuri sivutaulu
  - jos 32-bittinen osoite ja sivukoko 4KB (12 bittiä), niin sivuja  $2^{20} = 1\text{M}$  kappaletta
  - jokainen alkio useita tavuja, esim. 4 B, joten sivutaulu 4 MB
- Myös sivutaulu jaetaan sivuihin ja myös sivutaulun osia voidaan pitää levyllä
  - riittää, kun suorituksessa olevaan osaan liittyvät sivut muistissa

17

## 2-tasoinen sivutaulu

- Ylin hakemisto mahtuu yhteen sivuun, aina muistissa

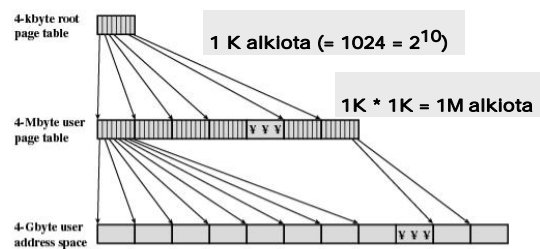
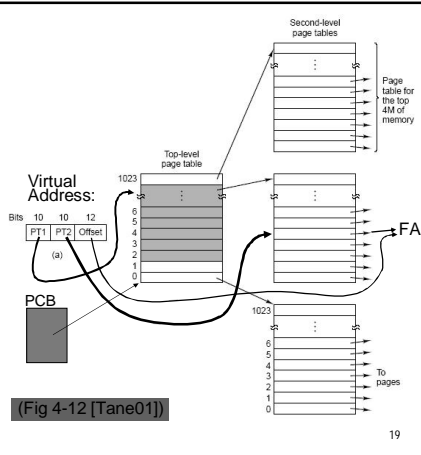


Figure 8.4 A Two-Level Hierarchical Page Table

8

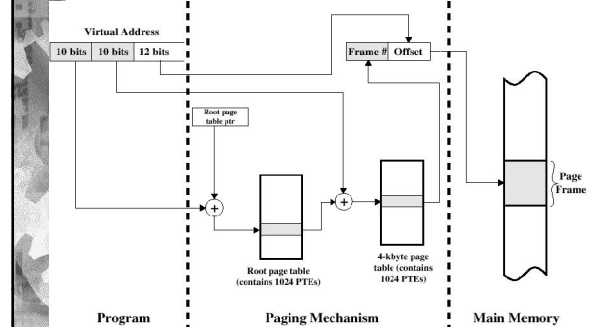
## Moni-tasoinen sivutaulu



19

## 2-tasoinen sivutaulu

Kuva 8.5



20

## KÄÄNTEINEN SIVUTAULU

21

## Käänteinen sivutaulu

- Fyys.muisti pienempi kuin virtuaaliavaruus
- Kirjaa sivutilakohtaisesti mikä sivu sijaitsee ko. sivutilassa
  - vain yksi globaali käänteinen sivutaulu
  - yksi alkio per fyysinen sivutila, jossa ko. sivutilassa sijaitsevan sivun numero
- Jokaisella prosessilla sivu 0, 1, 2, ...
  - miten tiedetään minkä prosessin sivu kyseessä?
  - alkiossa myös prosessin numero (pid)
  - MMU:hun rekisteri, jossa suoritettavan prosessin pid

22

## Käänteinen sivutaulu

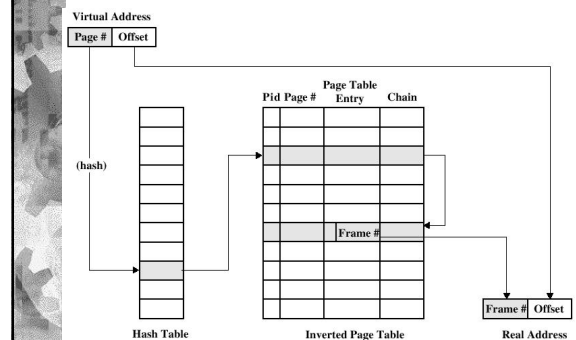
- Käänteisen ST:n indeksit sivutilan numeroita, mutta prosessin osoitteesta käy ilmi sivunumero ja siirtymä
  - etsittävä sisällön perusteella
- Etsintä peräkkäishakuna tehontonta
- Käytetään apuna hajautustaulua
  - käytä hajautinta indeksin laskentaan, etsi kotisolusta
  - samaan hajautusosoitteeseen kuvautuvat alkiot linkitetty toisiinsa
- Jos sivun tiedot ei listassa, aiheuta sivunpuutoskeskeytys
- Käyttökelpoinen vain jos TLB riittävän suuri

23

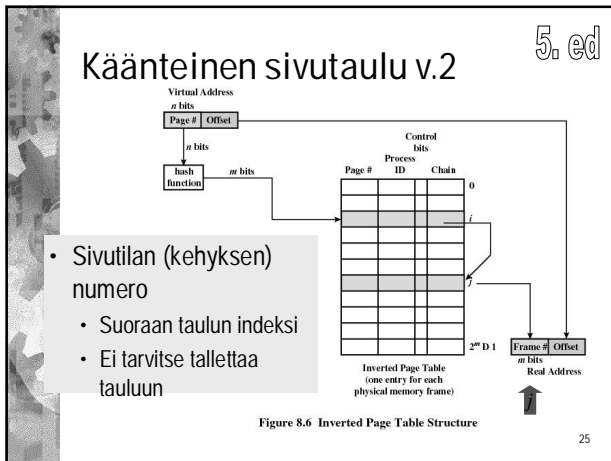
## Käänteinen sivutaulu

Kuva 8.6

4. ed



24



## Osoitemuunnospuskuri TLB

26

## Osoitemuunnospuskuri TLB

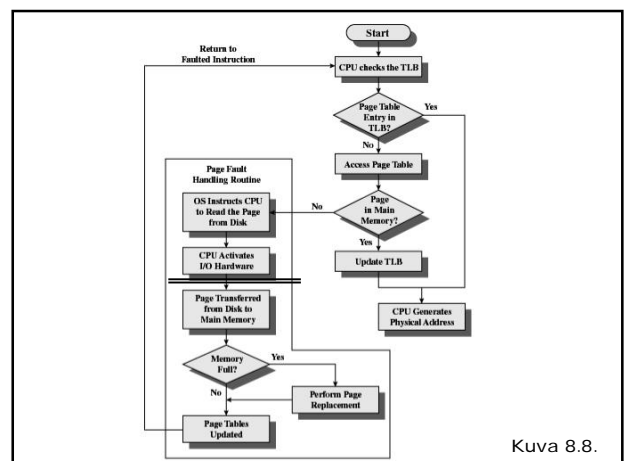
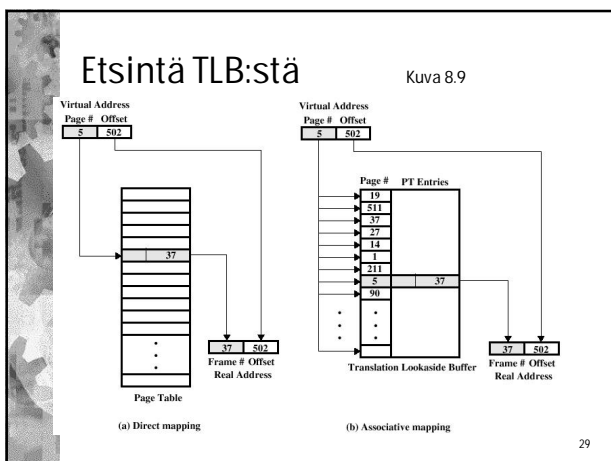
- Koska sivutaulu muistissa, jokaiseen muunnokseen näyttäisi liittyvän yksi ylimääräinen muistinuoto
  - hae sivutaulun alkio MMU:hun
  - tee osoitemuunnos
  - nouda / talleta ko. fyysiseen osoitteeseen
- Ratkaisu: osoitemuunnospuskuri TLB
  - Translation Lookaside Buffer
- MMU pitää tallessa edellisissä muunnoksissa tarvittavat sivutaulun tiedot
  - paikallisuus: sitä tarvitaan het'kohta uudestaan
  - sivutaulun alkioita ei tarvitse noutaa joka kerta

27

## Etsintä TLB:stä

- TLB nopeiden rekistereiden joukko
  - esim. 32 rekisteriä
- Käyttää assosiativista hakua
  - vertailu kohdistuu kaikkiin alkioihin yhtäaikaan
  - etsintä nopeaa
  - laittoiminto!
- Kaikilla prosesseilla sivut 0, 1, 2, ..., mutta eri sivutiloissa
  - TLB tyhjennettävä prosessin vaihdossa
- TLB:n alkiossa (ainakin)
  - sivunro, sivutaulun alkio, validibitti V

28



## TLB ja osoitemuunnos

sivunro = loog. osoitteen alkubitit  
siirtymä = loog. osoitteen loppubitit

Jos sivun tiedot ei TLB:ssä tai  $V=0$ ,  
nouda TLB:hen sivutaulun alkio  
**osoitteesta PTR + sivunro**

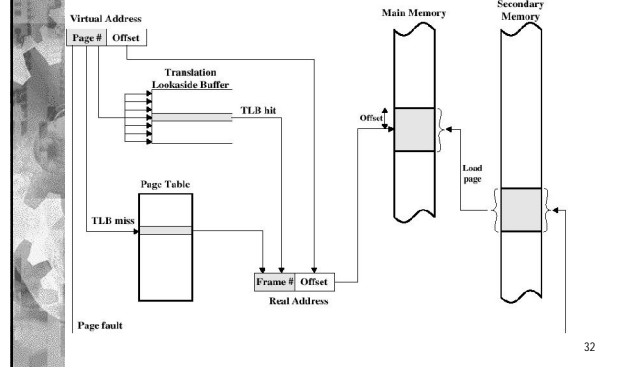
Jos  $P=0$ , aiheuta sivunpuutoskeskeytys

Fyys.os = Katenoi(Sivutilanro, Siirtymä)

- Kun keskeytys käsitelty, sama osoite tulee viitattavaksi uudelleen
  - Esim. PC:n kasvatus vasta osoitemuunnoksen jälkeen

31

## TLB ja osoitemuunnos Kuva 8.7



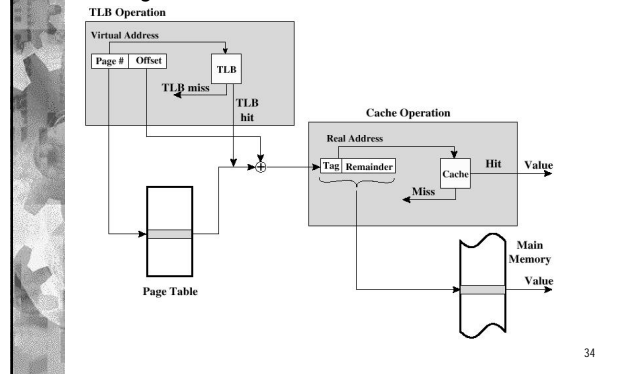
32

## TLB:n alustus

- Kun suoritettava prosessi vaihtuu, TLB:n vanha sisältö mitätöitävä
  - PTR osoittamaan uuden prosessin sivutauluun
  - nollattava TLB:n alkioiden validiibitit  $V=0$
- Koska TLB suhteellisen pieni, tarvitaan sopiva laitetason algoritmi, jonka perusteella valitaan korvattava alkio
  - TLB:n alkiossa mahd. myös laitetason viitelaskureita: poista se, jota ei ole aikoihin käytetty

33

## TLB ja välimuisti Kuva 8.10



34

## TLB tarkemmin? → Tietokoneen rakenne -kurssi

Fig. 5.47 from Hennessy-Patterson, Computer Architecture

DEC Alpha AXP 21064 memory hierarchy

Fully assoc, 12 entry instruction TLB

8 KB, direct mapped, 256 line (each 32B) instruction cache

2 MB, 64K line (each 32B) direct mapped, unified, write-back L2 cache

Fully assoc, 32 entry data TLB

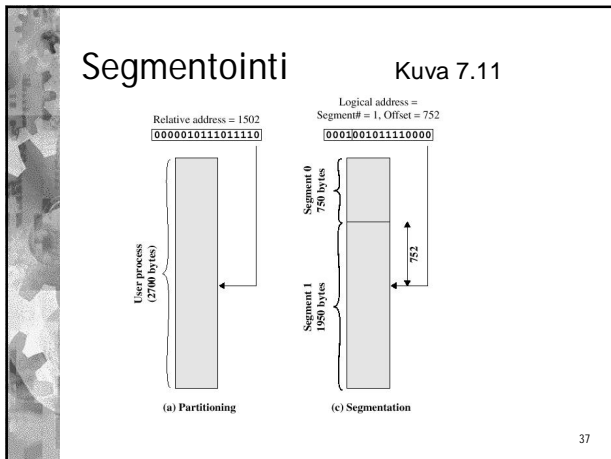
8 KB, direct mapped, 256 line (each 32B) data cache

main memory

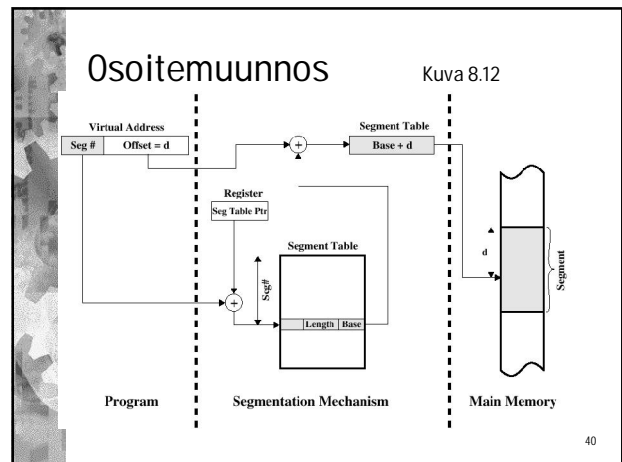
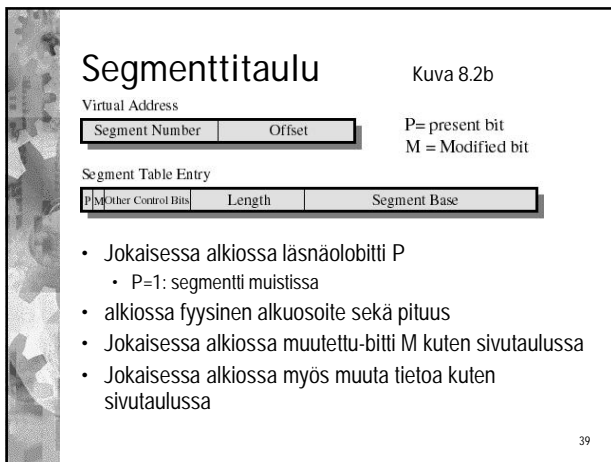
paging disk (dma)

## SEGMENTOINTI

36



- ### Segmentointi
- Perusideoiltaan samanlainen kuin sivutus, 'yksiköt' vain keskenään erikokoisia segmenttejä
  - Jokaisella prosessilla oma segmenttitaulu
    - Kertoo missä tämän prosessin segmentit sijaitsevat
  - Osoitemuunnos segmenttitaulun avulla
  - TLB:n käyttö kuten edellä
  - MMU:n rekisterissä nyt segmenttitaulun fyysinen osoite
- 38



- ### Huomioita
- Segmenttitaulun alkiossa alkuosoite ja pituus
    - segmentin kokoa helppo kasvattaa/pienentää dynaamisesti
      - saattaa vaatia segmentin uudelleensijoittamista
    - osoitteen oikeellisuus tarkistettavissa MMU:ssa
  - Segmentit erikokoisia, syntyy ulkoista pirstoutumista
    - Varaus/vapaus ei niin tehokasta kuin sivuilla
    - Muistin tiivistämistarvetta
  - Segmentti sopiva suojausyksikkö
    - ohjelmoija määrittelee segmentit ja käyttöoikeudet
    - käytötapa kopioitu segmenttitaulun alkioon
- 41

### Segmentointi ja sivutus yhdistettynä

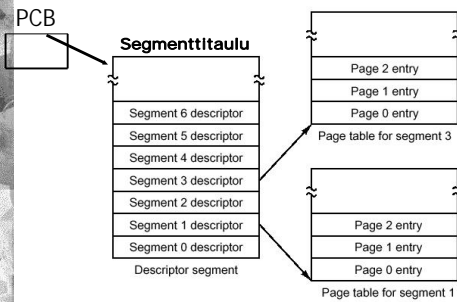
42

## Sivuttava segmentointi

- Eräät laitteistot ja KJ:t jakavat myös segmenttejä sivuiksi, ohessa eräs tapa
  - muistia helpompi hallita samankokoisina sivuina
  - ei ulkoista pirstoutumista
  - ei tiivistämistarvetta
- Jokaisella prosessilla
  - oma segmenttitaulu ja
  - yksi sivutaulu per segmentti
- Vrt. 2-tasoinen sivutaulu

43

## Sivuttava segmentointi Tan01 4-39



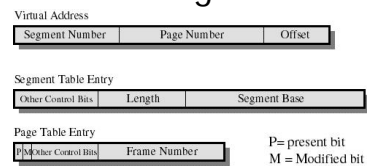
44

## Sivuttava segmentointi

- Looginen osoite jakautuu nyt kolmeen osaan
  - segmenttinumero
    - segmentin sivutaulun fyysinen osoite löytyy segmenttitaulun tästä alkiossa
  - sivunumero
    - sivunumeroa vastaava sivutilan numero löytyy sivutaulun tästä alkiossa
  - siirtymä
    - sivulla viitattu sana näin kaukana sivun alusta
- Myös segmenttitaulua / sivutaulua voidaan sivuttaa

45

## Sivuttava segmentointi Kuva 8.2c

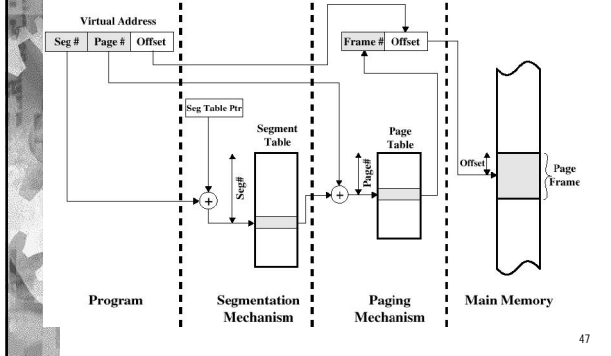


- Segmentin alkuosoitteen (segment base) paikalla ko. segmentin sivutaulun fyysinen osoite
- P-bitti ja M-bitti vain sivutaulun alkiossa
- Yhteiskäytön ja suojauksen data luonnollisimmin segmenttitaulun alkiossa
  - oikeudet annettu koko segmenttiin (sen kaikkiin sivuihin)

46

## Osoitemuunnos

Kuva 8.13



47

## Hyötyjä

- Ratkaisee dynaamisen linkittämisen ongelmakohdat
  - uuden segmentin (dynaaminen) linkittäminen tarkoittaa uuden alkion lisäämistä segmenttitauluun
- Segmentin koko voi kasvaa sivu kerrallaan, eikä segmentille tarvitse etsiä uutta paikkaa fyysisessä muistissa
- Yhteiskäyttö ja käyttöoikeudet voi määrittää segmenttiikohtaisiksi
  - kauniisti loogisten kokonaisuuksien mukaan
  - useita erilaisia suojaustasoja

48

# Yhteiskäyttö ja suojaus

49

## Yhteiskäyttö

- Jos usea prosessi ajaa samaa koodia, riittää siitä yksi kopio muistissa
- jokaisen prosessin sivu / segmenttitaulussa viitteet yhteisen koodin ja datan sisältäviin sivutiloihin / segmentteihin
- mutta kullakin prosessilla omat tilat yksityiselle datalle
- Koodin oltava vapaakäyntistä
  - ei saa muuttua
- Sivutus ei paras tapa yhteiskäyttöä ajatellen:
  - sivu 'kokoyksikkö', ei looginen yksikkö
  - käyttöoikeudet vaikea rajata funktiotasolla

50

## Yhteiskäyttö: editori

51

## Segmentointi ja yhteiskäyttö

- Segmentointi kaunis tapa yhteiskäyttöä ajatellen
  - ohjelman jakaminen eripituisiin segmentteihin loogisempaa kuin jakaminen tasapitkiksi sivuiksi
  - esim. yhteiskäyttöön tarkoitettu data omaksi segmentiksi, yksityinen data omaksi segmentiksi
- Ohjelmoija kertoo kääntäjälle
  - haluamastaan segmenttijaosta
  - haluamistaan käyttöoikeuksista
- Kääntäjä muodostaa tällä perusteella ohjelman loogiset osoitteet
  - segmentti ja siirtymä sen sisällä

52

## Erilaisia suojausryhmiä kuva 8.14

- Mahdollinen luokittelu:
- Ei saa käyttää lainkaan
- Saa käyttää vain dataa
- Saa käyttää vain koodia (???)
- Saa käyttää sekä koodia että dataa

53

## Yhteenvedoa ks. Taulukko 8.1

Sivutus	Segmentointi
Keskusmuisti jaettu vakiokokoisin sivutiloihin	
KJ jakaa prosessin vakiokokoisin sivuihin	Ohjelmoija/kääntäjä jakaa prosessin vaihtelevankok. segmentteihin
Prosessikoht. sivutaulut: missä sivutilassa sivu sijaitsee	Prosessikoht. segmenttitaulut: segmentin alkuos. ja pituus
Virt.os: (sivu, siirtymä)	Virt.os: (segmentti, siirtymä)
Sisäistä pirstoutumista	Ulkoista pirstoutumista muistin tiivistämistarve
Vapaiden sivutilojen lista	Vapaiden muistialueiden lista
Kaikki sivut ei muistissa: lasnaolobitti Sivut:n alkiossa	Kaikki segmentit ei muistissa: lasnaolobitti SegT:n alkiossa

54