



HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Käyttöjärjestelmät I

Osio 2: Luennot 5-8

Muistinhallinta

Tiina Niklander; kalvot © Auvo Häkkinen

Tietojenkäsittelytieteen laitos
Helsingin yliopisto

Käyttöjärjestelmät I

YKSINKERTAINEN MUISTINHALLINTA

Stallings, Luku 7

Sisältöä

Yleistä muistinhallinnasta (luku 7.1)

Yksinkertainen muistinhallinta

a) kiinteät partitiokoot (luku 7.2)

b) dynaamiset partitiokoot (luku 7.2)

c) Buddy System (luku 7.2)

d) yksinkertainen segmentointi (luku 7.4)

e) yksinkertainen sivutus (luku 7.3)

Yksinkertainen =

**prosessi aina kokonaan muistissa tai
kokonaan levyllä**

Käyttöjärjestelmät I

**YLEISTÄ
MUISTINHALLINNASTA**

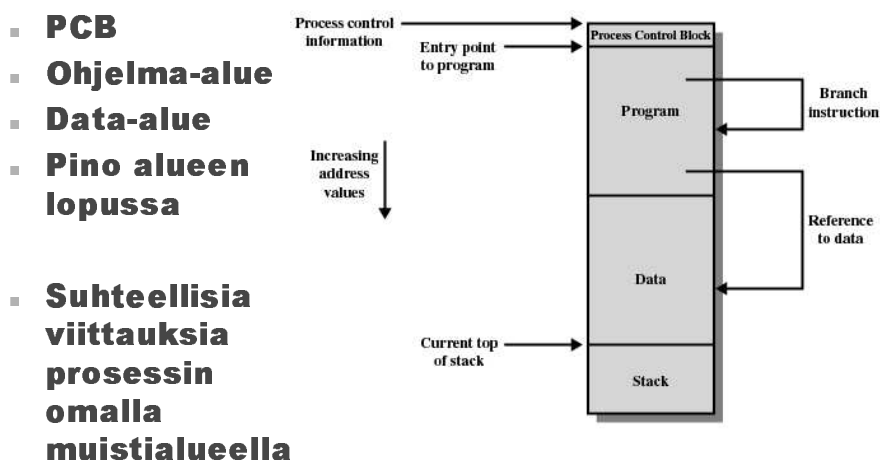
Fyysinen muisti

- **Fyysinen muisti, 'sitä voi potkaista'**
 - ◆ muistiavaruus
 - ◆ laitteisto (MMU, väylät) käyttää fyysisiä osoitteita
- **Keskusmuisti**
 - ◆ koodin + käsiteltävän datan suoritusaik. tallennus
 - ◆ joukko peräkkäisiä tavuja
- **Tukimuisti**
 - ◆ tiedon (ohjelmat, data) pysyvä tallennus
 - ◆ joukko peräkkäisiä lohkoja
- **Siirto näiden välillä muistinhallinnan ja tiedostojärjestelmän leipätyötä**
 - ◆ mahd. automaattisesti KJ:n toimesta

Looginen muisti

- **prosessin osoiteavaruus**
 - ◆ loogiset osoitteet eli virtuaaliosoitteet
- **voi olla suurempi kuin fyysinen muisti**
- **kullakin prosessilla oma osoiteavaruus**
 - ◆ osoitteet suhteellisia alun suhteen (0..MAX)
- **sovellus jakaantuu loogisesti moduuleihin**
 - ◆ ne voidaan tehdä eri aikoina, osoitteiden paikkaus linkityksessä
 - ◆ erilaisia käyttöoikeuksia
 - koodi R (vapaakäyntisyys)
 - data R / W / RW
 - ◆ osa moduuleista tarkoitettu yhteiskäyttöön

Prosessin rakenne



KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkk

Figure 7.1 Addressing Requirements for a Process

Yleistä muistinhallinnasta

- **Ytimelle voidaan sallia kiinteä paikka muistissa**
 - ◆ voi käyttää suoraan fyysisiä osoitteita
- **Loppu KJ:n muille osille ja sovelluksille**
- **KJ huolehtii siitä, että muistiin mahtuu mahd. monta prosessia**
 - ◆ vapaan / varatun tilan hallinta
- **Laitteisto huolehtii siitä, etteivät prosessit sotke toisiaan**
 - ◆ MMU

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 8

Muistinhallinta: vaatimuksia

Vapaa sijoitettavuus

- **Heittovaihto**

- ◆ KJ voi siirtää prosesseja välillä levyllä
- ◆ KJ voi ottaa suoritettavaksi useampia prosesseja
- ◆ prosessin paikka voi vaihdella suorituksen aikana

- **Ohjelmoija ei voi tietää minnepäin muistia sovellus sijoittuu suoritusajana**

- ◆ suhteelliset osoitteet

- **Viittaukset fyysiksi osoitteiksi viimeistään ennen muistinoutoa/talletusta**

→ Ajonaikainen osoitemuunnos MMU:ssa

Muistinhallinta: vaatimuksia

Suojaus

- **Luvatta ei saa käyttää toisen muistialueita**

- **Osoitetarkistus käännoajana mahdotonta**

- ◆ sovelluksen moduulit voidaan kääntää eri aikoina
- ◆ prosessien sijainti voi vaihtua suoritusajana
- ◆ käskykanta voi sallia osoitustapoja, joissa osoite lasketaan suoritusajana

- **KJ ei voi sitä tehdä!**

→ Ajonaikainen laillisuustarkistus osittain MMU:ssa ja osittain KJ:ssa

Muistinhallinta: vaatimuksia

Yhteiskäyttö

- **Sallittava yhteisen koodin / datan käyttö**
 - ◆ suojauksista tinkimättä!
 - **Koodi ei muutu suorituksen kuluessa**
 - ◆ vapaakäyntisyys (reentrancy)
 - ◆ järkevämpää sallia koodin yhteiskäyttö kuin pitää muistissa useita kopioita
 - **Monet prosessit tekevät yhteistyötä muiden kanssa, joten niillä yhteisiä tietorakenteita**
 - ◆ Esim. tuottajalla ja kuluttaja yhteinen puskuri
- käytä säikeitä, palvelupyynnöjä

Käyttöjärjestelmät I

YKSINKERTAINEN MUISTINHALLINTA

Yksink. muistinhallinta

= **Prosessi kokonaan muistiin / levyille**

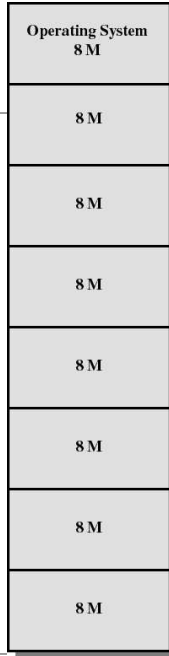
- **Jos ohjelma > fyysisen muistin koko, sitä ei välttämättä pystytä suorittamaan**
- **Menetelmät:**
 - ◆ a) kiinteä partitiointi
 - ◆ b) dynaaminen partitiointi
 - ◆ c) Buddy System
 - ◆ d) yksinkertainen segmentointi
 - ◆ e) yksinkertainen sivutus
- **Katso ns. katoava kansanperinne**
 - ◆ ei paljon käyttöä nykyisissä järjestelmissä
 - ◆ kuuluu kuitenkin KJ:n peruskäsitteistöön

Käyttöjärjestelmät I

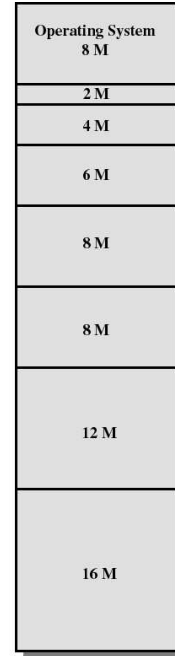
a) Kiinteät partitiot

Kiinteät partitiot

- **KJ (operaattori) jakoi muistin kiinteänkokoisiin partitioihin**
- **Varausalueet olivat kaikki yhtäsuuria tai niiden koot saattoivat vaihdella**
- **Ohjelma, joka oli pienempi tai yhtäsuuri kuin partitio, voitiin ladata ja ajaa ko. partitiossa**



Equal-size partitions



Unequal-size partitions

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

Kiinteät partitiot

- **Jos ei riittävän suurta vapaata partitiota, KJ teki tilaa heittovaihdolla**
 - ◆ joku prosessi levyille
 - ◆ PCB jäi muistiin
- **Jos ohjelma niin iso, ettei sopinut mihinkään partitioon, piti ohjelmoijan ratkaista tilanne**
 - ◆ kerrostus (overlying)
 - ☞ “ohjelmoijan hoitama segmentointi”
 - ◆ vain osa ohjelmasta muistissa
 - ◆ piti koodata mitä osia (aliohjelmia, ‘segmenttejä’) kullakin hetkellä muistissa ja missä kohdassa

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 16

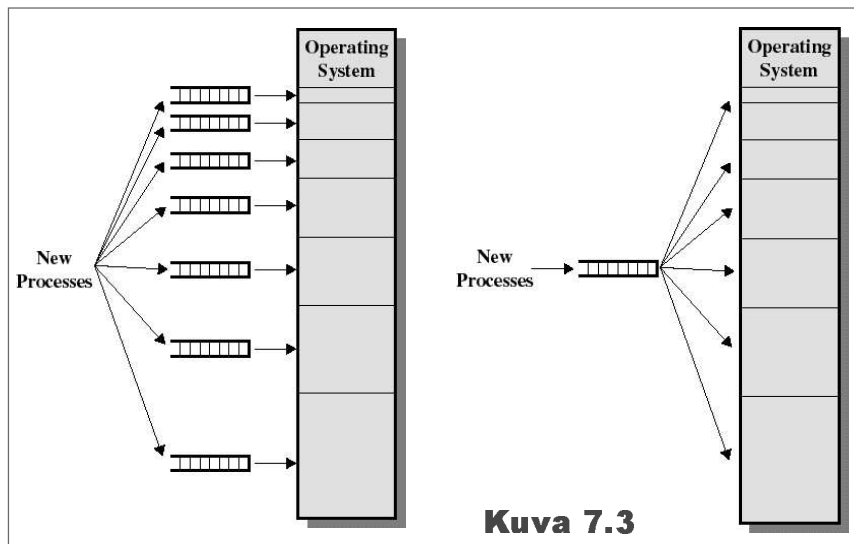
Kiinteät partitiot

- **Muistin käyttö melko tehotonta**
 - ◆ varattiin aina kokonainen partitio, vaikka vähempikin olisi riittänyt
- **Sisäinen pirstoutuminen (internal fragmentation)**
 - ◆ partitioiden sisälle jäi tyhjää tilaa
 - ◆ vapaa tila yhdessä olisi saattanut riittää uudelle prosessille, mutta se ei ollut yhtenäisellä alueella

Kiinteät partitiot: Sijoitus

- **Jos kaikki partitiot samankokoisia**
 - ◆ jos vapaita, valitse joku niistä
 - ◆ jos kaikki varattuja, heittovaihtaa joku Blocked-prosessi levyille
 - tilaa vapautuu aina saman verran
- **Jos partitiot eri kokoisia**
 - ◆ valitse pienin partitio, johon prosessi sopii
 - yritti minimoida sisäistä pirstoutumista
 - ◆ a) erikokoisille partitioille omat prosessijononsa
 - partitiokoko yksi työn parametreista
 - ◆ b) yksi jono, josta valittiin johonkin vapaaseen partitioon

Kiinteät partitiot: Sijoitus



KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 19

Käyttöjärjestelmät I

b) Dynaaminen partitiointi

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 20

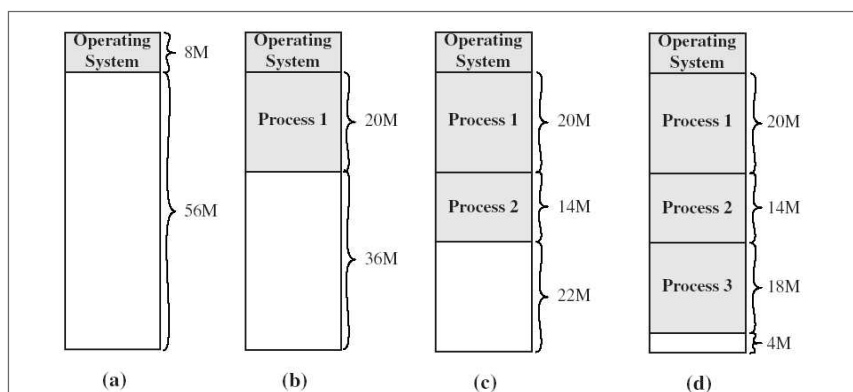
Dynaaminen partitiointi

- **Ei etukäteen partitointia**
- **Varausten koot ja lukumäärä vaihtelivat dynaamisesti prosessien tarpeiden mukaan**
- **Prosessille muistia vain sen verran kuin tarvitsi**
- **Ulkoinen pirstoutuminen** (external fragmentation)
 - ◆ varausten / vapautusten tuloksena väleihin jäi pieniä vapaita alueita
- **KJ tiivistä muistia välillä** (compaction)
 - ◆ prosesseja siirrettiin, jotta vapaa tila yhteen kohtaan
 - ◆ yleisrasite

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 21

Dynaaminen partitiointi Kuva 7.4

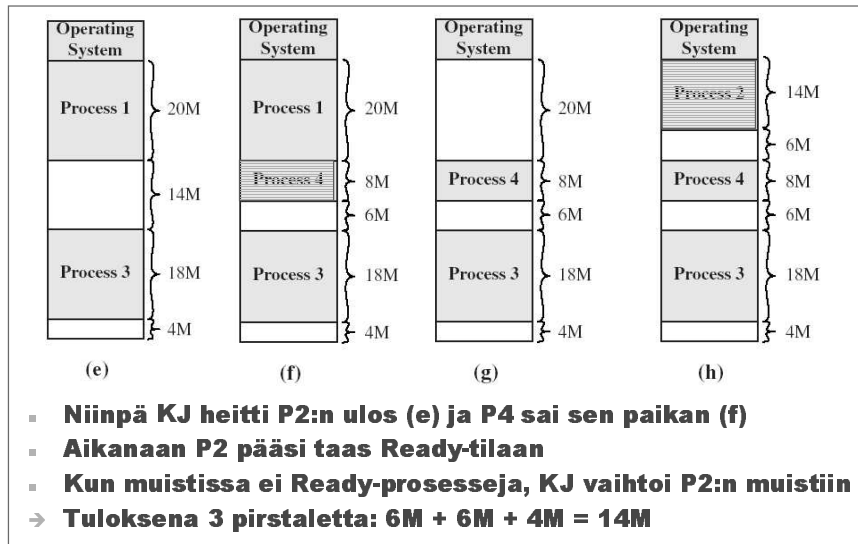


- **Vaiheessa (d) ei enää tilaa prosessille 4, koko 8 M**
- **Jos kaikki 3 prosessia joutuivat Blocked-tilaan, KJ otti uuden P4:n suoritettavaksi**

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 22

Dynaaminen partitiointi



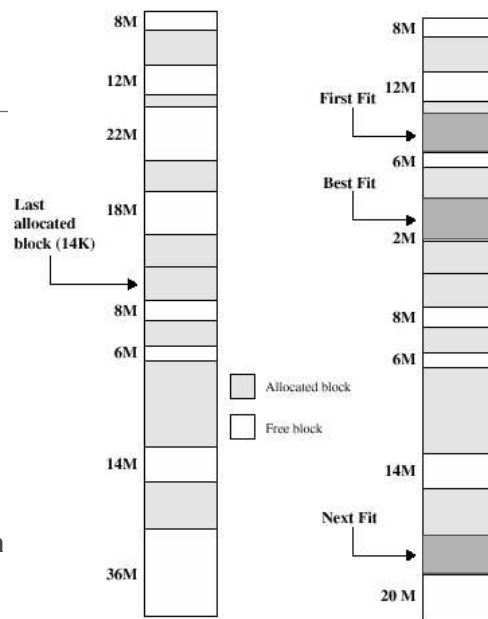
KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 23

Sijoitus Kuva 7.5

Mistä kohtaa varataan?

- Tavoitteena vähäinen tiivistämistarve
- Best-fit kooltaan sopivin
- First-fit ens. kooltaan riittävän suuri
- Next-fit jatka etsintää edellisestä kohdasta



Example Memory Configuration Before and After Allocation of 16 Mbyte Block

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

Paras sijoitusalgoritmi?

- **First fit**
 - ◆ yksinkertainen
 - ◆ helpoin toteuttaa tehokkaasti (paras)
 - ◆ isosta aukosta jää jäljelle isohko aukko
- **Best fit**
 - ◆ hyvä nimi, mutta tuloksena mahd. pieniä aukkoja
 - ◆ tuloksena nopeimmin pirstoutuminen
- **Next fit**
 - ◆ varaukset / aukot muistin loppuosaan
- **Toteutus: linkitetty lista, jossa vapaat alueet koon tai osoitteen mukaisessa järjestyksessä**
 - ◆ kumpi parempi?
 - ◆ (osoite,pituus)→... →(osoite,pituus)→

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 25

Poistoalgoritmi

- **Kun muistissa vain Blocked-prosesseja, kannatti ottaa uusi prosessi suoritukseen**
 - ◆ ettei CPU jouten
 - ◆ heittovaihda joku levyllä
 - ◆ poistettava prosessi Blocked-Suspend-tilaan
 - ◆ uusi prosessi tilaan Ready-Suspend tai Ready
- **Mikä pois, jotta saatiin sopivasti vapaata?**
 - ◆ iso vs. pieni
 - ◆ sellainen, jonka vieressä iso tyhjä alue
 - ◆ pieni vs. suuri prioriteetti

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 26

Käyttöjärjestelmät I

c) Buddy System

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 27

Buddy System

- **Kompromissi kahdesta em. menetelmästä**
 - ◆ kiinteät partitiokoot, mutta dynaaminen jako partitioihin
 - ◆ ei haluta jättää pieniä tyhjiä tiloja
 - ◆ varaa pikkuisen enemmänkin kuin tarve vaatii
- **Varausyksikön koko ilmaistaavissa 2:n potenssissa**
 - ◆ suurin mahdollinen yleensä koko muisti
 - ◆ pienimmälle varausyksikölle joku minimikoko
 - ☞ yksinkertaistaa vapaiden alueiden kirjanpitoa
- **Varaukset ja vapaiden alueiden yhdistely toteutettavissa tehokkaasti**

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 28

Buddy System

- **KJ ylläpitää kustakin koosta omaa listaa**
 - ◆ lista osoitteiden mukaisessa järjestyksessä
 - ◆ aluksi vain yksi suuri varausyksikkö
- **Varaus**
 - ◆ jos oikean kokoluokan listassa ei vapaata alkiota, jaa luokkaa suurempi alue kahdeksi pienemmäksi
 - ◆ toista tarvittaessa
- **Vapautus**
 - ◆ kun listassa kaksi fyysisesti vierekkäistä aluetta, yhdistä ne kokoluokkaa suuremmaksi alueeksi
 - ◆ toista tarvittaessa

Buddy System: esimerkki

1 Mbyte block	1 M					
Request 100 K	A = 128 K	128 K	256 K	512 K		
Request 240 K	A = 128 K	128 K	B = 256 K	512 K		
Request 64 K	A = 128 K	C = 64 K	64 K	B = 256 K	512 K	
Request 256 K	A = 128 K	C = 64 K	64 K	B = 256 K	D = 256 K	256 K
Release B	A = 128 K	C = 64 K	64 K	256 K	D = 256 K	256 K
Release A	128 K	C = 64 K	64 K	256 K	D = 256 K	256 K
Request 75 K	E = 128 K	C = 64 K	64 K	256 K	D = 256 K	256 K
Release C	E = 128 K	128 K	256 K	D = 256 K	256 K	
Release E	512 K			D = 256 K	256 K	
Release D	1 M					

Kuva 7.6

Buddy: varaukset puuna

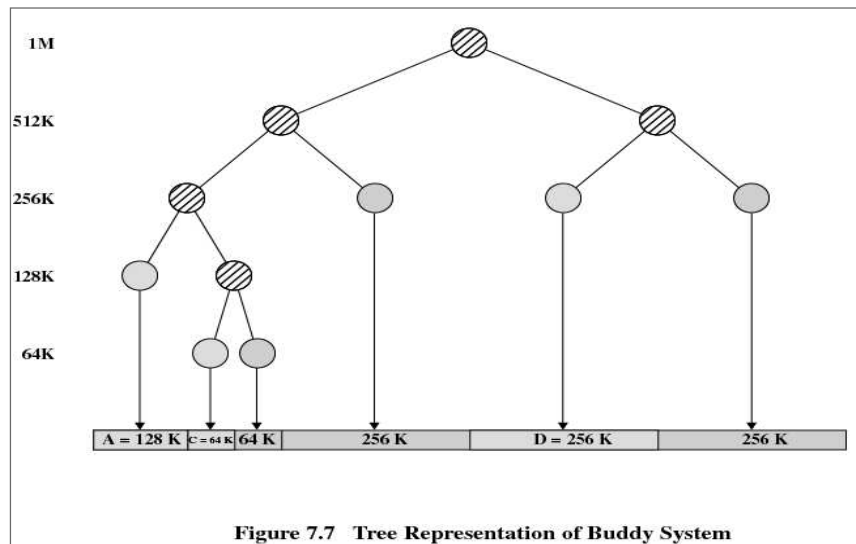


Figure 7.7 Tree Representation of Buddy System

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 31

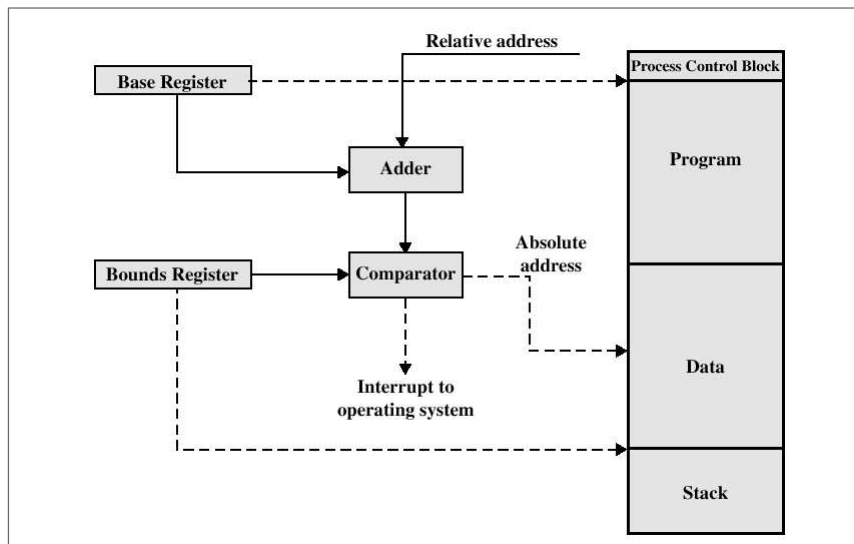
a) b) c) Osoitemuunnos

- **Ohjelmassa loogiset osoitteet**
 - ◆ kaikki suhteellisia sen alun suhteen
- **Historia**
Lataaja (KJ:n osa) muutti loogiset osoitteet fyysisiksi osoitteiksi samalla, kun latasi ohjelmakoodin muistiin
- **Nyky aika**
osoitemuunnos vasta käskyjä suoritettaessa
- **KJ:n ydin voi käyttää fyysisiä osoitteita, joten kun CPU suorittaa KJ:tä, osoitemuunnosta ei tarvitse tehdä**

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 32

a) b) c) Osoitemuunnos Kuva 7.8



KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 33

Laitteistotuki

- **MMU:hun muunnosta ja suojaustarkistusta varten kaksi rekisteriä**
 - ◆ ns. kanta- ja rajarekisteri
 - ◆ Base prosessin fyysinen alkuosoite
 - ◆ Limit prosessin loppuosoite (tai pituus)
- **Kun prosessi suoritukseen, kopioidaan näille arvot PCB:stä**
- $\text{Fyys. osoite} = \text{loog. osoite} + \text{Base}$
jos fyysinen osoite > Limit
aiheuta poikkeus 'virh. muistiosoite'
muuten
 $\text{MAR} \leftarrow \text{fyysinen osoite}$

KJ-I S2004 / Tiina Niklander; kalvot Auvo Häkkinen

5 - 34