Atmiņas hierarhija un keši

Atmiņas struktūra

Iepriekš:

- Ideāla atmiņa
- Kāpēc atmiņa ir tik nozīmīga?
- Atmiņas hierarhija
- Atmiņas pamatprincipi
- Kešs
- Četri pamata jautājumi
- Kešu asociativitātes veidi
- Keša tega izmērs un saturs

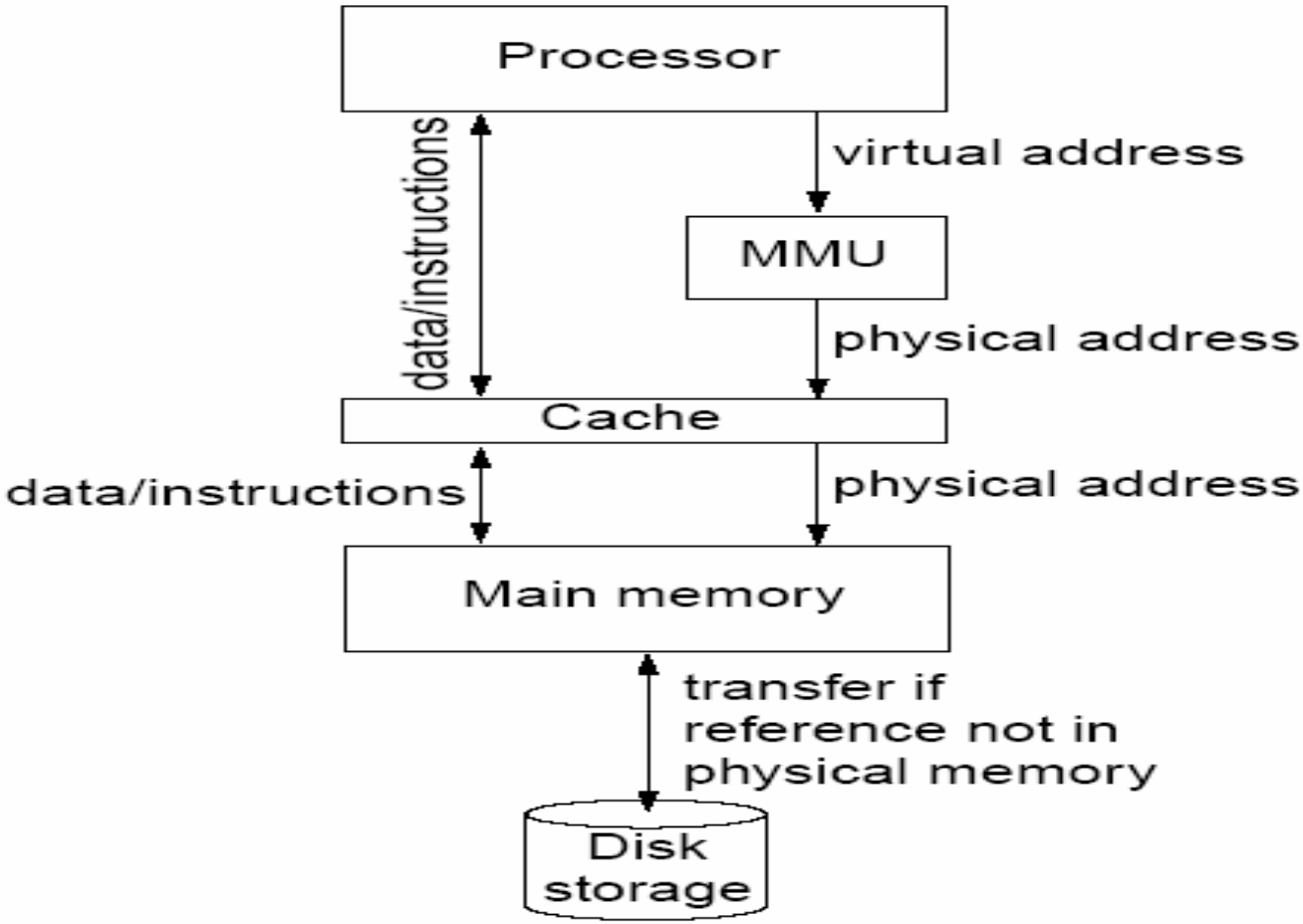
Kas notiek ieraksta laikā?

- Ieraksts cauri (Write through): Informācija tiek ierakstīta abās vietās (keša blokā un zemāka līmeņa blokā).
- Ieraksts atpakaļ (Write back): Informācija tiek ierakstīta tikai keša blokā. Modificētais keša bloks tiek ierakstīts atmiņā tikai tad kad tas tiek aizvietots.
  - Tīrs / netīrs? (jāpievieno ”dirty bit” katram blokam)
- Ieraksts cauri
  - Vieglāk izveidot
  - Pamatatmiņa ir vienmēr konsistenta
  - Jālieto ieraksta buferi jo citādi rodas ieraksta aizkaves (write stalls)
- Ieraksts atpakaļ
  - Mazāka atmiņas plūsma
  - Ieraksti notiek ar keša ātrumu
  - Pamatatmiņa ***ne vienmēr ir konsistenta*** ar kešu saturu
  - Aizvietošanas (Evictions) darbības ir ilgākas jo tad ir jāveic ieraksts atmiņā pirms var aizvietot bloku

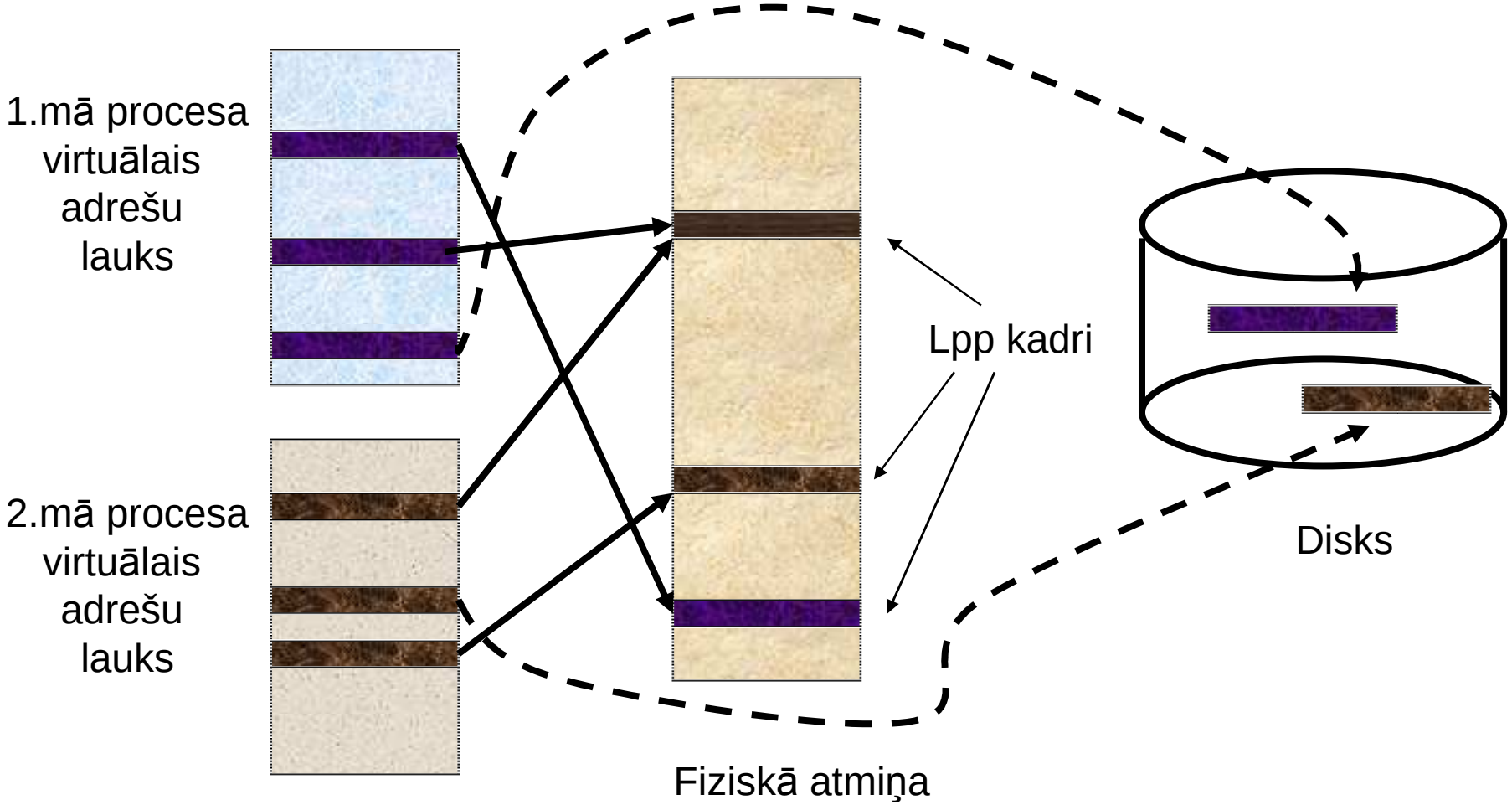
Virtuālā atmiņa

- Adrešu lauks kas ir nepieciešams programmas darbam parasti ir daudzkārt lielāks nekā pieejamais pamat atmiņas apjoms.
- Tikai neliela programmas daļa ietilpst pamatatmiņā bet pārējais tiek glabāts sekundārajā atmiņā (diskos)
- Lai programmu varētu izpildīt tai ir jāatrodas pamatatmiņā. Tapēc kādam tās segmentam vispirms ir jātop ielādētam pamatatmiņā (potenciāli aizvietojot kādu citu tur jau esošu segmentu)
- Datu un programmu pārvietošanu no un uz pamatatmiņu notiek automātiski (OS)
- Tas ***kā to dara*** tiek saukts par VM tehnoloģiju
- CPU binārā adrese ir virtuāla (loģiskā) adrese kas ir daudzkārt lielāka nekā RAM apjoms un kura ir attiecināta uz ***neesošu atmiņu***.
- Ja virtuālā adrese attiecas uz to programmas daļu kas jau atrodas RAM (kešā) tad piekļuve tai notiek tieši. Ja tā nav tad tā vispirms ir jāielādē pamatatmiņā.
- Virtuālo adrešu translāciju uz fiziskām veic speciāls mezgls - Memory Management Unit (MMU).

Virtuālā atmiņa



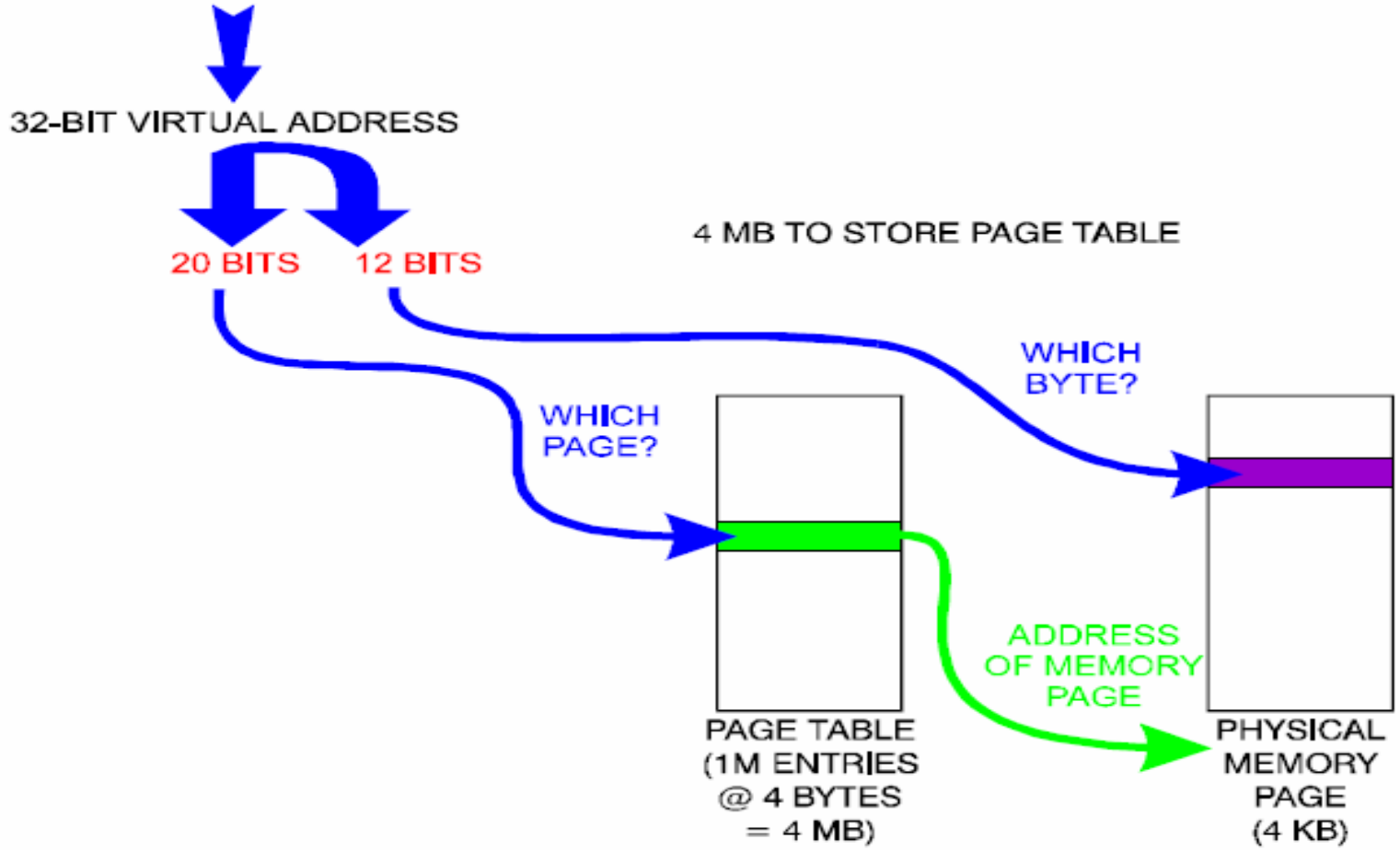
Virtuālā atmiņa



Virtuālā atmiņa un pieprasījumlapošana

- Virtuālā atmiņa nodrošina arī uzdevumu aizsardzību vienam no otra.
- Pamatā šodien virtuālais atmiņas lauks (kods un dati) ir sadalīts vienāda izmēra lapaspusēs (no 512B līdz 16MB tipiski 4KB) “pages”
- Fiziskā atmiņa ir sadalīta kadros “frames” kas pēc izmēra atbilst lapaspusei
- Kādas vēl organizācijas var būt +/- ?
- LPP ir pamata informācijas elements kas ar VM sistēmas palīdzību var tikt pārvietots starp pamatatmiņu un disku
- OS izlemj kuras dotās lietotnes lpp ievietot RAM un kuras aizvietot tā lai minimizētu lpp kļūdas “page faults”
- Lpp kļūda ir situācija kurā CPU atsaucas uz adresi kas atrodas lapā kas savukārt neatrodas RAM.

Adrešu translācija



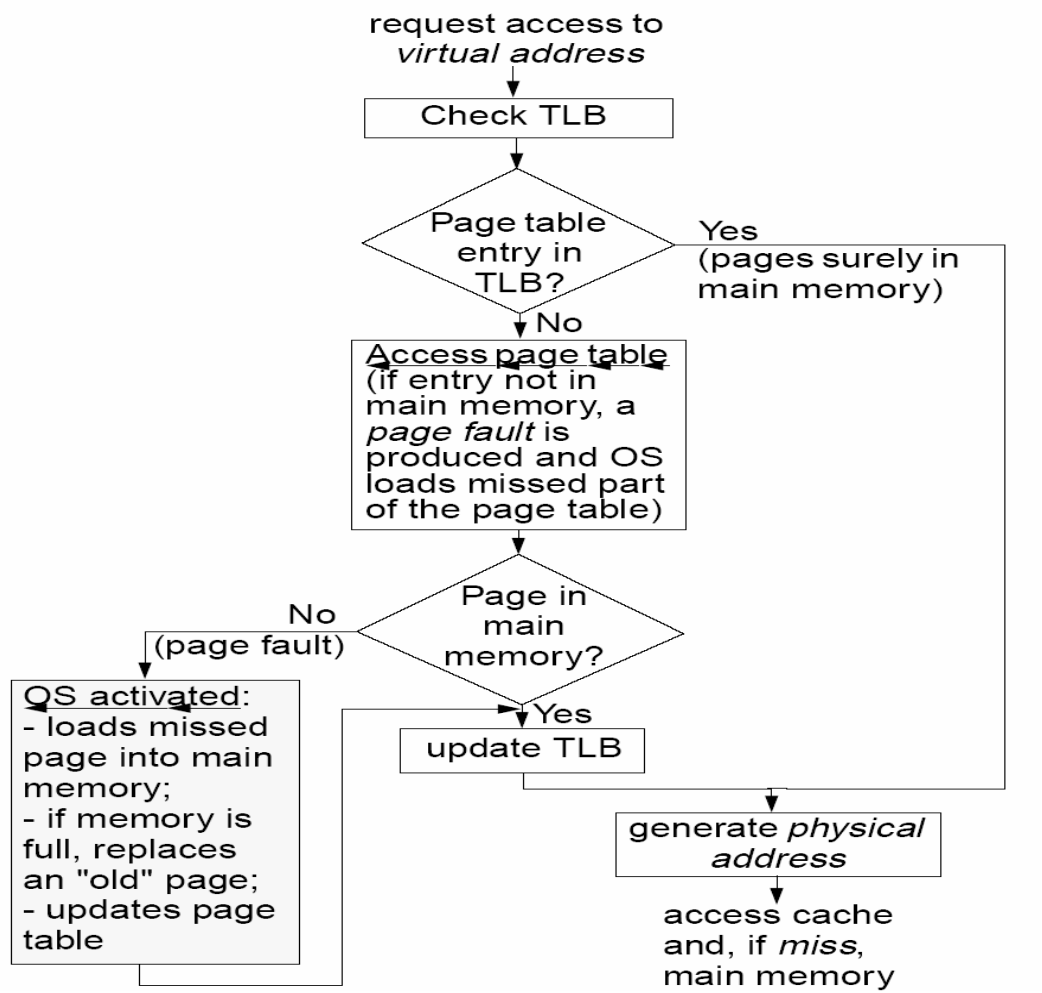
Adrešu translācija LPP tabula

- Katra piekļuve atmiņā izvietotam vārdam izsauc virtuālās adreses translāciju fiziskajās adresēs:
  - Virtuālā adrese = lpp. Nr. + nobīde
  - Fiziskā adrese = kadra nr. + nobīde
- Translāciju veic MMU ar lpp tabulas palīdzību
- Lpp tabula satur ***vienu ierakstu katrai*** virtuālās atmiņas lpp
- Katrs Lpp tabulas ieraksts satur atmiņas kadra adresi kurā atrodas vajadzīgā lpp (ja tā ir ievietot pamatatmiņā)
- Katrs Lpp tabulas ieraksts satur arī papildus informāciju:
  - Ir/nav ielādēta RAM
  - Ir/nav izmainīta
  - Piekļuves tiesību informācija
- Diemžēl lpp tabula ir ***ļoti liela*** un piekļuvei tai ir jānotiek ***ļoti ātri***.

LPP tabula/TLB

- Lai atrisinātu šo problēmu lpp tabulas ierakstiem lieto speciālu kešu “translation lookaside buffer (TLB)”
- Tā darbība ir līdzīga atmiņas kešu darbībai un tā satur tikai nesen lietotos ierakstus
- Lpp tabula ir pat pārāk liela lai to glabātu pamatatmiņā tāpēc to ar VM palīdzību sadala pa atmiņas hierarhiju:
  - TLB kešu
  - Pamatatmiņu
  - Disku

Virtuālā atmiņa

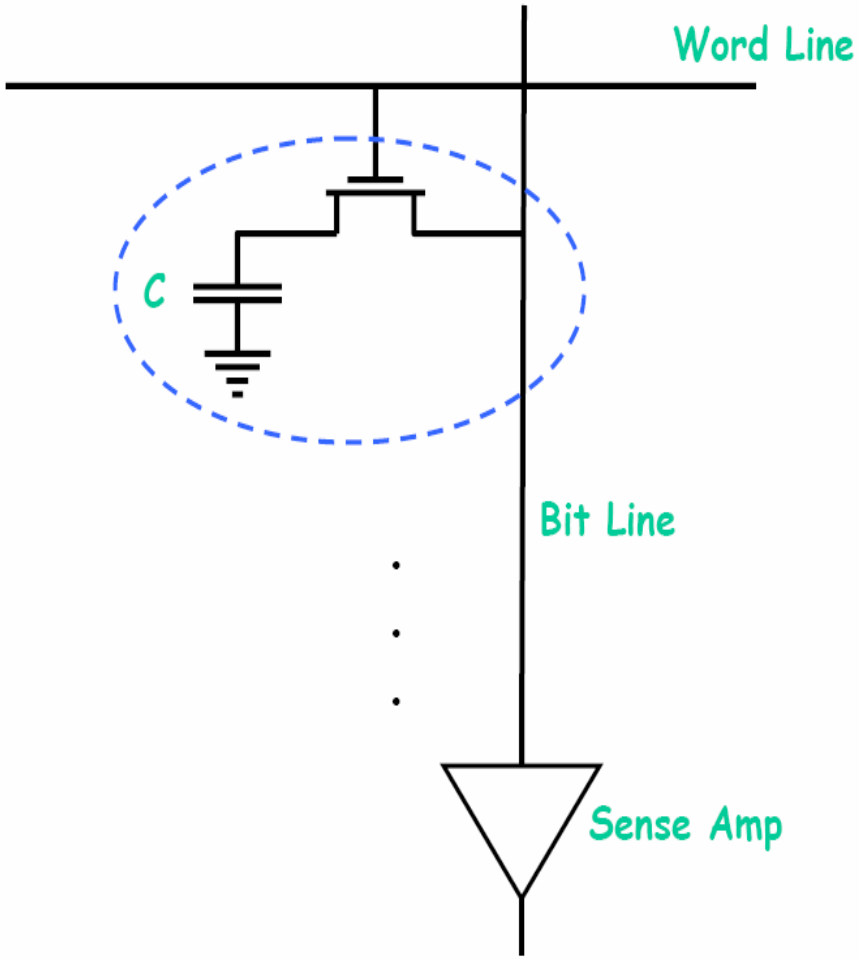


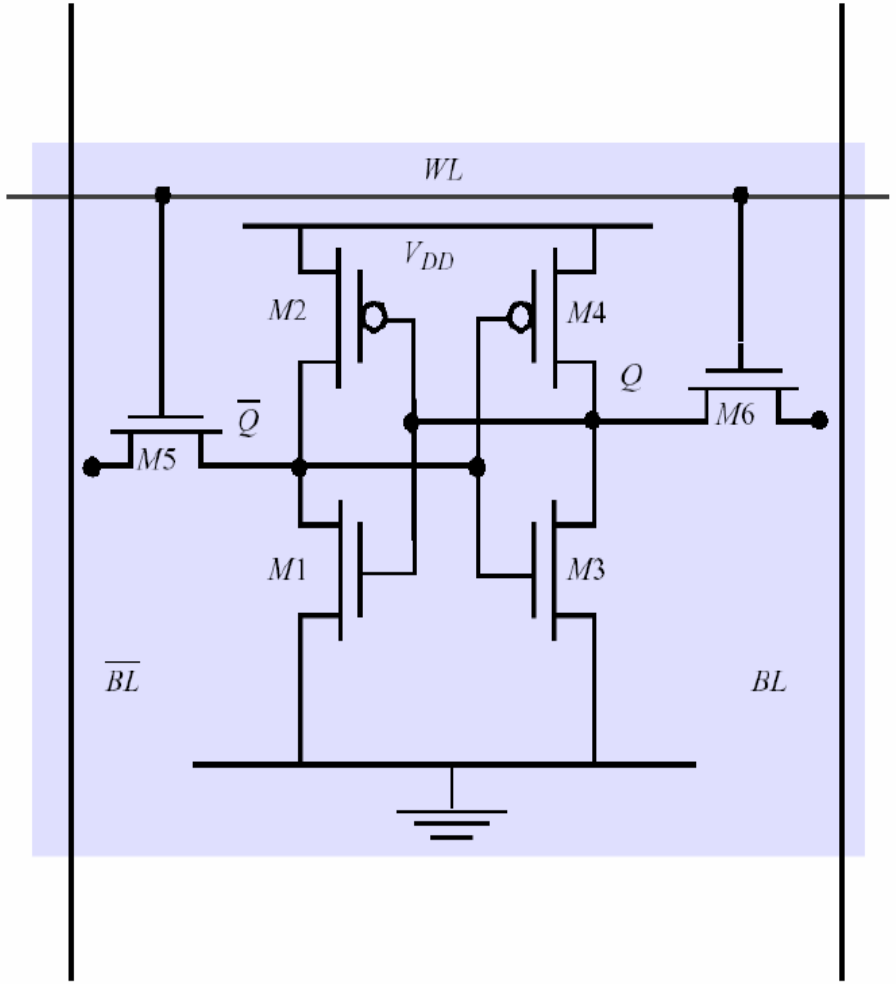
Atmiņas aizsardzība

- Daudzuzdevumu vidēs procesi nedrīkstētu iespaidot viens otru
- Vienkāršākā realizācija ir divi reģistri - bāze un ierobežojums:
  - Bāze + Nobīde < Ierobežojumu
- Bet kas var modificēt šādus reģistrus?
  - Lietotne to nedrīkst darīt (jo tad zūd jēga)
- OS kodola režīms ir apveltīts ar šādām iespējām:
  - Var piekļūt atmiņai izmantojot fiziskās adreses
  - Var izmainīt šos reģistrus
- Sistēmas izsaukumi “lēkā” starp lietotāja un kodola režīmu
- Lietotāja process pasaka ko tas vēlas veikt un kodols to izdara
- Vēl labāk ja var uzturēt katram procesam savu Lpp tabulu

Fiziskā realizācija

- Lūdzu izlasīt lekcijas online!





Fiziskā realizācija

- Pēc savas būtības ***D***RAM nav radīta ātrdarbības sasniegšanai
  - Atbildes laiks ir atkarīgs no ***kapacitātes*** shēmas
  - Šīs shēmas ***parametri pasliktinās samazinot izmēru*** vai palielinot blīvumu
  - Kapacitātes lādiņš zūd tāpēc vajadzīga periodiska informācijas atjaunošana
  - DRAM grūti izveidot CMOS procesā

- ***S***RAM (static nevis synchronous) :

Fiziskā realizācija

- Tikai lasīšanai paredzētās atmiņas (ROM)
  - “Pastāvīga” informācijas glabātuve
  - Datus ieraksta ražošanas procesā
  - Parasti lieto liela pasūtījuma skaita gadījumos
  - » PROMs
- Programmējamās ROM atmiņas (PROM)
  - Lietotājs pats var ***vienu reizi*** ierakstīt savu informāciju izmantojot PROM programmatoru
  - Izmanto sīksēriju ražošanā
- Dzēšamās PROM atmiņas
  - Programmēšana līdzīgi kā PROM
  - Var izdzēst paturot zem UV gaismas avota

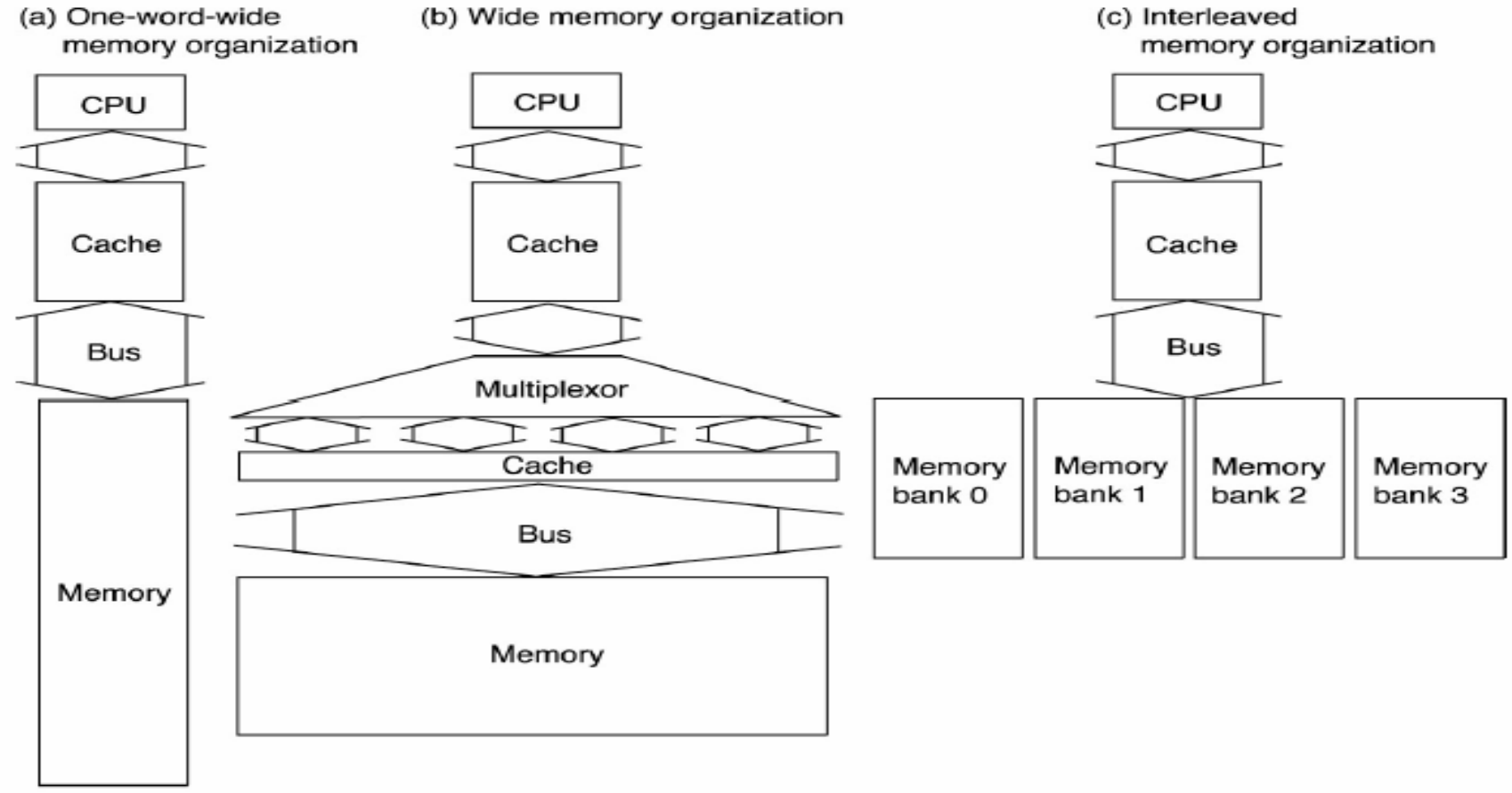
Fiziskā realizācija

- Elektriski dzēšamās PROM atmiņas (EEPROMS)
  - Var dzēst elektriski
  - Var rakstīt vairakkārt ***neizņemot no sistēmas***
  - Nav obligāti jādzēš pirms ieraksta
  - Var ierakstīt pa baitam
  - Ieraksta liks ir mērāms mikrosekundēs
  - Lieto izstrādes laikā vai tur kur vajadzīga ***personalizētas*** informācijas pastāvīga glabāšana
- Flash atmiņa
  - Līdzīga EEPROM jo lieto elektrisku dzēšanu
  - Ātrāka dzēšana un iespēja dzēst blokus nevis tikai baitus
  - Lielāks blīvums nekā EEPROM

Atmiņas kļūdas

- RAM kā jau pusvadītāju ierīce ir pakļauta kļūdām:
  - Pastāvīgās kļūdas “Hard (permanent) errors”
    - Apkārtējās vides iespaids
    - Ražošanas defekti
    - Nolietošanās
  - Pārejošās kļūdas “Soft (transient) errors”
    - Barošanas problēmas
    - Radiācija
- Jo mazāki elementi jo lielāka kļūdu rašanās varbūtība
- Atmiņas sistēmas satur loģiku un citas tehnoloģijas kļūdu noteikšanai ***un/vai*** labošanai
- Par to gan ir jāmaksā ar atmiņas vārda palielinājumu, papildus paritātes bitiem....

Fiziskā realizācija



Fiziskā realizācija

- *Vienāda platuma*
  - Vienkārši realizēt
  - CPU, Keši, Kopnes, Atmiņa visi vienādā platumā (32 / 64 biti)
- *Platāka atmiņa (Wide)*
  - CPU/Multipleksors ***1*** vārds
  - Multipleksors/Kešs,Kopne, Atmiņa ***N*** vārdi
  - Alpha, UtraSPARC
- *Pārklāta atmiņa (Interleaved)*
  - CPU, Kešs, Kopne ***1*** vārds
  - Atmiņa satur ***N*** moduļus
  - Moduļi satur *vārda daļu vai veselus vārdus*

Kopumā

- CPU – atmiņa veiktspējas atšķirība ir pamata ierobežojošais faktors kas neļauj kāpināt kopējo datora veiktspēju
- Atmiņas hierarhija:
  - Izmanto lokalitātes principus
  - Tuvāk CPU => mazāk, ātrāk, dārgāk
  - Tālāk no CPU => lielāks, lēnāks, lētāks
- 4 pamata jautājumi
- Programmas kas nepielieto lokalitāti neiegūst uzlabojumu no atmiņas hierarhijas
- Keši ir aparatūra bet VM vairāk tomēr ir programmatūras pārziņā.
- Kešu parametri
  - Kopējais izmērs, bloka izmērs, asociativitāte
  - Adresācija ar virtuālo vai fizisko adresi?
- Nākotnē - Intelligent RAM (“IRAM”)?

Mājās

- http://en.wikipedia.org/wiki/Computer_memory
- http://dt.cs.rtu.lv/viewfile.php/18/file/27/477/9.clekcija.pdf