# M4. Virtual memory

Explain the virtual memory system and its hardware support. Keywords: MMU, execution modes, memory protection, paging, TLB.

Physical and virtual memories

* Technology characteristics - density, cost, bandwidth, latency.
  + Desto højere densitet, desto mere hukommelse kan du få på en chip
  + Latency – hvor lang tid tager det at lave flere operationer i træk.
* SRAM vs. DRAM.
  + SRAM (static) har transistorer der holder deres status så længe der er strøm
  + DRAM (dynamic) skal refreshes, ellers går dataen tabt efter kort tid
* Memory hierarchy.
  + Lidt hurtig hukommelse, mere langsommere hukommelse.
* Temporal and spatial locality of programs.
* Importance of alignment of data.
  + Dårligt alignment giver unødvendige operationer hvis der skal hentes flere words, hvor dataen kunne have ligget i færre.
* Limits of addressing.
  + Byte addressing vs. word addressing
* Content addressable memory.
  + CAM har indbygget hardware til søgning
* Virtual memory
  + Benefits.
    - Removes the concrete implementation from the end user.
  + MMU
    - Mapping virtual to physical addresses.
      * Powers of two make it easy to translate
    - Memory protection.
  + Demand paging
    - Segmentation.
    - Resident vs. non-resident pages.
    - Pages.
  + Page tables
    - How address translation works.
    - Role of OS.
    - Necessity for hardware support.
    - What a TLB is.
      * Translate Lookaside Buffer

Fra PSS:

## Hukommelsesstyring

**Lagerhierarkiet og lokalitet**

CPU registre, L1, L2, L3, fysisk hukommelse, swap disk og virtuel hukommelse

Lokalitet betyder at størstedelen af et programs køretid kun bliver brugt i en lille del af koden.

**Paging (og segmentering)**

Paging:

- Virtuelle lager opdeles i sider af fast størrelse, fysiske lager opdeles i rammer på samme størrelse.

- Sider placeres i rammer, og OSet vedligeholder sidetabeller, der matcher sider med de korrekte - rammer. Der vedligeholdes en sidetabel for hver proces.

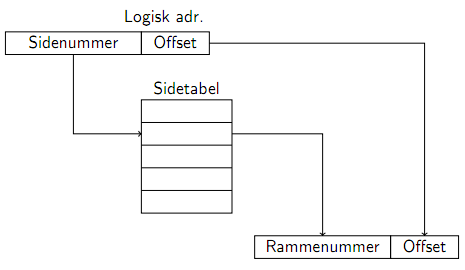
- Giver ingen ekstern fragmentering (ledig hukommelse bliver delt op i så små dele at de er ubrugelige) og kun lidt intern fragmentering (data bliver allokeret i hukommelsen, men aldrig brugt).

- Addresseoversættelse (32 bit):

- Logisk adresse (sidenummer, 20 bit, offset, 12 bit)

- Fysisk adresse på formen (rammenummer, offset)

- Simpel sidetabel (et niveau):



Segmentering:

- Hukommelsen deles ind i segmenter med egne adresserum, størrelse, rettigheder og så videre.

- De er defineret ved startadressen og længden, med logisk adresse på formen (segmentnr, offset), med segmenttabeller til at holde styr på dem.

- Fordele:

* Segmenter kan defineres af programmøren.
* De giver finere granularitet af rettigheder.
* Ingen intern fragmentering.

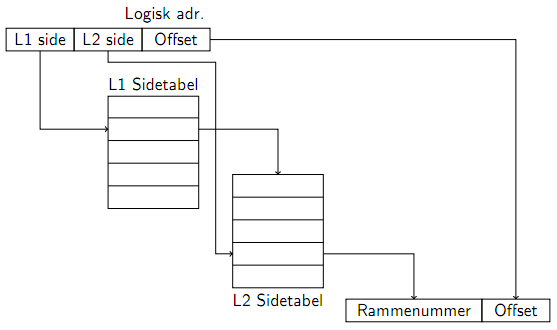
- Ulemper:

* Ekstern fragmentering
* Manglende understøttelse i Windows og Linux

Paging og segmentering sammen giver adresser på formen (segmentnr, sidenr, offset) og kræver en sidetabel per segment.

20 bit sidenumre giver for store sidetabeller (4 MB/2^20 indgange af 32 bit). Lav niveaudelt sidetabel med sidenummer delt i to dele af 10 bit. Hver sidetabel fylder kun 4 KB

To niveaus sidetabel:

Til 64-bit systemer bruges invertede sidetabeller. Der er kun en tabel for hele systemet, og den holder styr på rammerne, og fortæller hvilken side, der ligger i rammen. Der bruges en hash-tabel til at slå op i den, da tabellen er for lang til at få god performance i.

Sidetabeller på Pentium har 32 bit per indgang med 12 af bitsne til ekstra information. Giver god performance på trods af ”spild” på 12 bit, da størrelsen passer med sider.

**Swapping, demand paging**

Demand paging er når en side indlæses ved første tilgang. Sider i sidetabellen kan have status ubrugt (giver fejl ved tilgang), reserveret (hentes ved næste tilgang) eller committed (er placeret i en ramme).

Små sider giver bedre lokalitet og få sidefejl, men store sidetabeller.

Større sider knap så god lokalitet og flere sidefejl, men mindre sidetabeller.

Meget store sider giver få sidefejl, men meget I/O og meget intern fragmentering.

Fordele:

* Sider der ikke bruges bliver aldrig allokeret.
* Processor kan være større end fysisk lager.
* Intet behov for at kende processors lagerbehov.
* Det er nemt at indlæse programmer

Swapping er at flytte data ud på sekundært lager, hvilket er nyttigt hvis det primære lager fyldes op. Lokalitetsprincippet siger at vi kan fjerne de sider der bruges mindst uden at det har en stor effekt på systemets performance.

Optimering:

* Uændrede sider kan bare slettes fra primært lager i stedet for at blive skrevet tilbage til det sekundære lager.
* Page buffering, optimering til swap out. Et minimum af rammer holdes altid fri, og periodisk udvælges et antal rammer der kan skrives til sekundært lager. Siderne slettes ikke, men placeres i en sidebuffer, hvorfra de bliver fjernet hvis siden tilgås igen. Hermed kan dårlige valg fortrydes.
* Pre-paging, optimering af swap in. Sider indlæses før de tilgås, hvilket kan give markant bedre ydelse.

**Sidefejl og sideerstatningsalgoritmer**

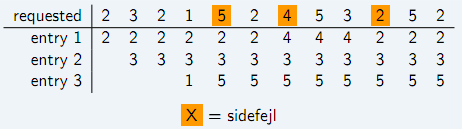
Sidefejl fås hvis processen tilgår en ubrugt eller reserveret side. Herfra overtager styresystemet så.

Hvis en sidefejl opstår, men der ingen ledige rammer er skal en side erstattes. Til dette bruges sideerstatningsalgoritmer. Disse kan have global (sider vælges på tværs af processor) eller lokal (en proces for et antal rammer allokeret, og sider i disse erstattes efter processens behov, se working set) politik.

**OPT, LRU, FIFO, Clock**

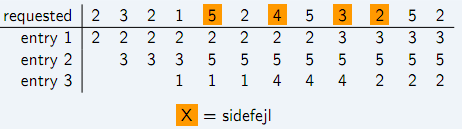
Optimal sideerstatning (OPT):

* Siden der forbliver ubrugt længst skal erstattes.
* Ikke en rigtig algoritme, mere et princip.



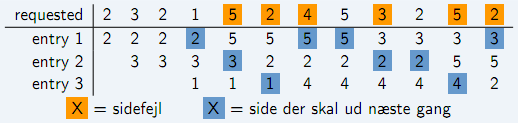
Least Recently Used (LRU):

* Via lokalitetsprincippet, fjern siden der har været ubrugt i længst tid.
* Processor med god lokalitet får næsten OPT-lignende performance.
* Bruges ikke i praksis.



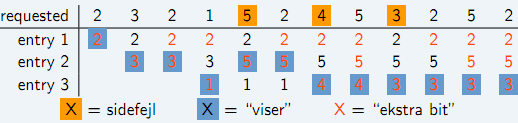
First in, first out (FIFO):

* Erstat de sider der har været i lageret længst.
* Implementeres med en linked list.
* Tager ikke højde for lokalitet.
* Kan optimeres med page buffering.



Clock:

* Ekstra bit i sidetabellen sættes ved tilgang til siden, hvilket holder øje med lokaliteten.
* Sider erstattes kun hvis de ikke har været brugt siden sidst ”viseren” på ”uret” var forbi.
* Næsten lige så god som LRU.



* Modified clock går efter umodificerede sider. Kræver en ekstra bit i sidetabellen.

**Rammeallokering**

Fast rammeallokering

* Lige allokering giver alle processor det samme antal rammer.
* Proportionel allokering giver rammer i forhold til hvor mange sider processen fylder.

**Working set modellen**

En proces' working set er mængden af sider processen har tilgået inden en bestemt tidsramme.

Tidsrammen fastsættes eksperimentelt for finde der hvor den bedst rammer processornes lokalitet.

**Thrashing**

Når CPUen bruger det meste af tiden på at håndtere sidefejl i stedet for ”almindelige” instruktioner.