Cache (HW)

Hvad er et memory hierarki?

- **Et hierarki** bestående af **hukommelsesenheder,** sorteret ifht. **størrelse**, **pris** og **access time**.
    - Interne **processor registre** -> Meget hurtig, meget dyr
    - **Lille memory**: processor cache
    - **Main memory**: RAM
    - **Device:** Sekundær hukommelse.
- Jo længere man kommer ned, bliver hukommelsen **større** og **langsommere**

Hvad er en cache? - og hvad bruger vi den til?

- **Meget lille** og **hurtig** hukommelse
- En **bro** mellem **hurtig og langsom** enheder -> En bro mellem processorkerne og hukommelsen.
- Da hukommelsen er langsommere, bruges cache til **hurtig tilgang til data der oftest bliver brugt** af processoren.
- **Adgang** til mest brugte data i cachens hastighed.

Hvordan er en cache opbygget?

- Cache kan være i en **Harvard arkitektur** hvor der en til **instruction** og en til **data.**
- Der er **2 dele** i en cache:
    - **Cache memory:**
        - Selve **indholdet** af cache ligger her
        - Hver linje består af:
            - **Cache-tag:** Adressen på hukommelsen
            - **Status:** 2 bits
                - **Valid:** Angiver om cache linjen er **aktiv**, original data fra hukommelse
                - **Dirty:** Angiver om data der er i cachen, også er i hukommelsen
            - **Data:** Selve data fra hukommelse
    - **Cache Controller:**
        - **Kontrollerer** den ønskede adresse **fra processoren**
        - **Deler** adressen op i 3:
            - Bit 4:11(**set index**): Bruges til at **lokalisere** cache line i **cache memory**
            - Bit 12:31 (**tag index**): Først tjekkes **status** bitsene. **Sammenligner** denne tag (**adressen i main memory**) med den pågældende linjes tag
                - **Cache-hit** -> data bliver **leveret** til processoren vha. data index. Hvis ikke
                - **Cache-miss** -> Data bliver **kopieret** i cachen fra hukommelsen og sendes derefter til processor
            - Bit 0:3 (**data index**): Bruges til at give data tilbage til processoren når den er klar

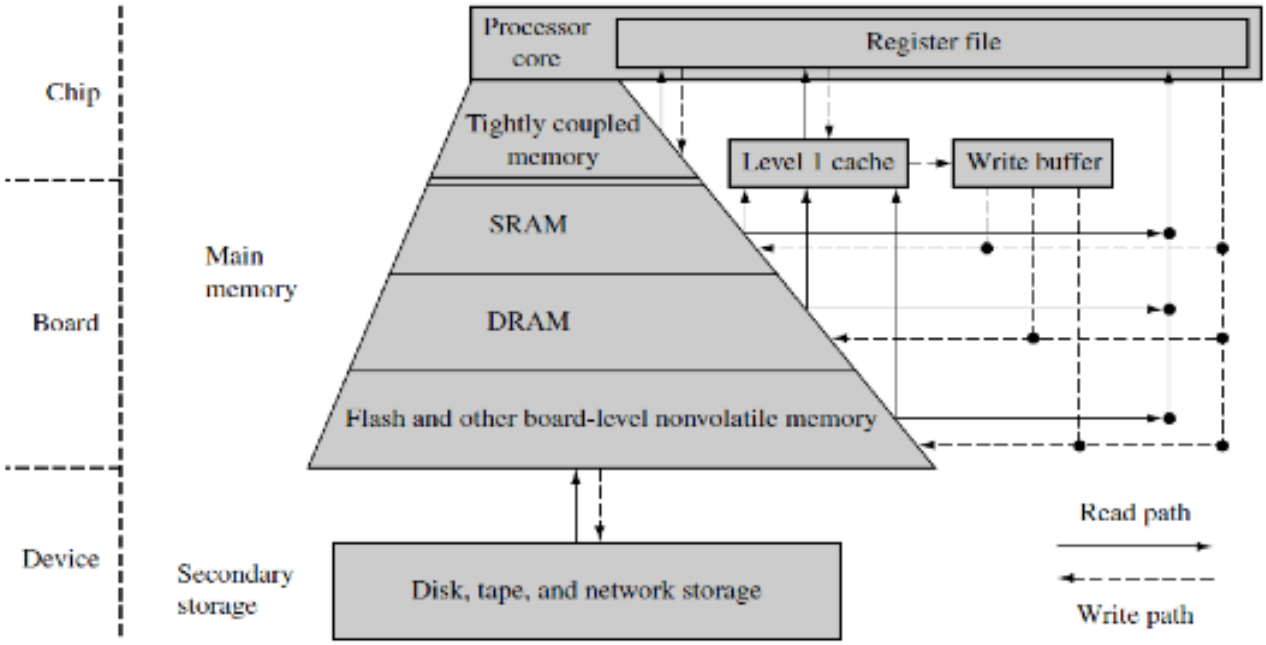
Hvilke begrænsninger har en cache og hvordan kan vi minimere disse?

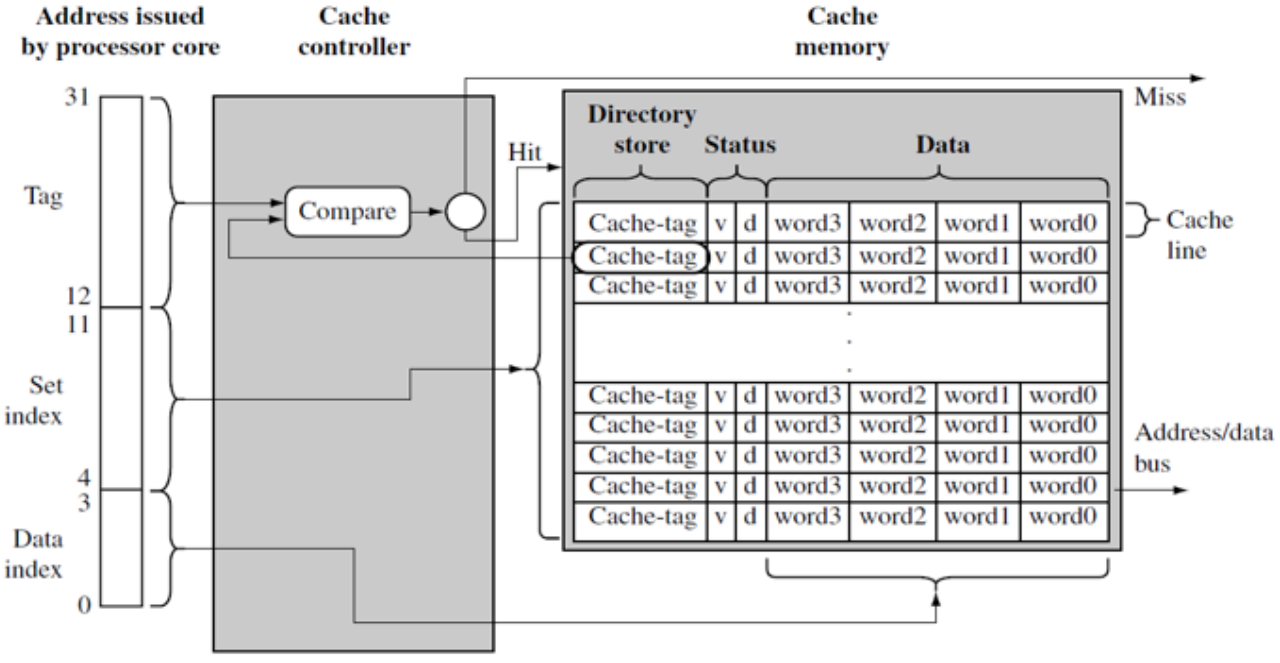
- En begrænsning for cache er, at der kan opstå **trashing:**
    - Gentagende **loading** og **evictions** (overskrivning af valid line)
    - **Løsning: Set Associative Cache:**
        - Opdele cache memory i mindre dele (**ways**).
        - **Flere** kan skrive til samme cache line uden at data bliver overskrevet.

Hvordan kan man som programmør optimere brugen af cache?

- **Write buffers:** Brug en **FIFO buffer**, så cache kan skrive til i stedet for direkte til hukommelse -> **Frigør** ressourser hos cache.
- **Write-back:** Opdatér kun cachen -> **Dirty bit set.** Når linjen skal fjernes, opdateres hukommelsen.
- Andre optimeringer:
    - Brug af **lokal variabel** -> write-back
    - **Undgå brug af cache når:** Streaming, engangsdata.

Cache (HW)


Chip
Board
Device
Main memory
Secondary storage
Processor core
Register file
Tightly coupled memory
Level 1 cache
Write buffer
SRAM
DRAM
Flash and other board-level nonvolatile memory
Disk, tape, and network storage
Read path
Write path


Address issued by processor core
Cache controller
Cache memory
31
Tag
12
11
Set index
4
3
Data index
0
Compare
Hit
Miss
Directory store
Status
Data
Cache-tag
v
d
word3
word2
word1
word0
Cache line
Address/data bus