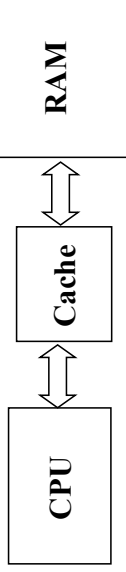


Dagens temaer

- Mer om cache-hukommelse (kapittel 6.5 i "Computer Organisation and Architecture")
- RAM
 - Typer, bruksområder og oppbygging
- ROM
 - Typer, bruksområder og oppbygging
- Hukommelsesbusser

Cache (repetisjon)

- Ønsker så mye og rask hukommelse som mulig tilgjengelig for et program under eksekvering, både for instruksjoner og data
- Cache-minnet er mye raskere enn konvensjonell RAM (ca 10-100 ganger), men dyrere slik at man ikke kan ha like mye av det som RAM
- Cache inneholder en kopi av områder i RAM
- Cache er integrert på CPU'en eller plassert rett ved siden av (eller begge deler)



Cache (repetisjon)

- Kapasiteten til cache ligger i mellom det interne i CPUen og RAM. Typiske verdier er fra 8 KB til 2 MB
- L1 cache ligger på samme brikke som CPU'en, mens L2 og L3 ligger ved siden av
- Siden cache'n er nesten like rask som CPUen, trenger ikke CPU'en å stoppe opp og vente i mange klokkesyklar på at RAM skal levere data.
- CPU'en vet ikke om data eller instruksjoner ligger i cache eller i RAM
- CPUen får beskjed utefra (buss- eller cache-kontroller) når data er klare. Om CPUen venter 1 eller 100 000 klokkesyklar spiller ingen rolle annet enn for hastigheten.

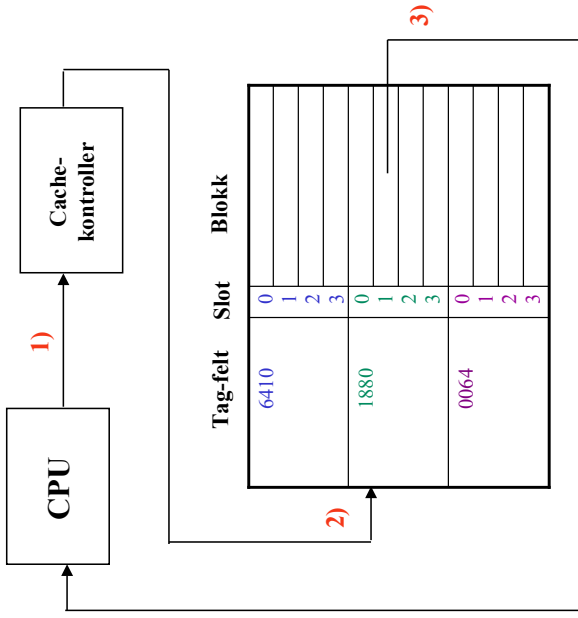
Cache (repetisjon)

- Siden cache'n er mindre en hurtigminnet (RAM) består mesteparten av administrasjonen av cache i å velge ut hvilken del av programmet og hvilke data som skal ligge i cache, og hvilke som må ligge i RAM.
- Cache benytter seg av **lokalitetsprinsippet**:
 - Instruksjoner/data aksepteres som regel sekvensielt, dvs fra samme område i hukommelsen
 - Som en konsekvens vil de samme instruksjonene/data også aksepteres nær hverandre i tid
- Cache kopierer derfor hele blokker (størrelse fra 4 byte opp til noen kilobyte)

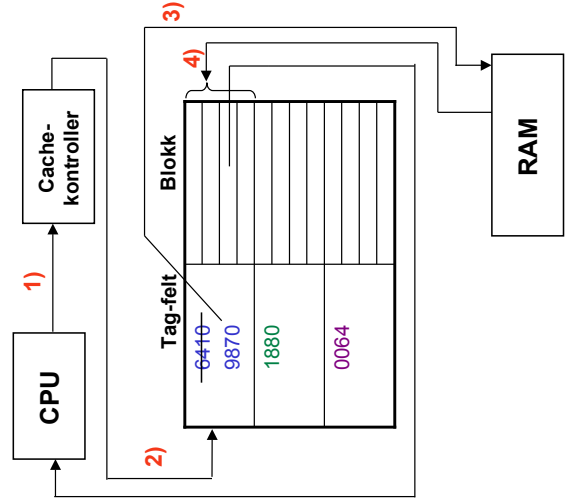
Tag-felt	Slot-nummer	Blokk	Tag+Slot = adresse
6410	0	Data/instruksjon	6410
	1	Data/instruksjon	6411
	2	Data/instruksjon	6412
	3	Data/instruksjon	6413
1880	0	Data/instruksjon	1880
	1	Data/instruksjon	1881
	2	Data/instruksjon	1882
	3	Data/instruksjon	1883
0064	0	Data/instruksjon	0064
	1	Data/instruksjon	0065
	2	Data/instruksjon	0066
	3	Data/instruksjon	0067

Startadresse i blokken

Tilsvarende adresse i RAM



- Når en prosessor skal hente en instruksjon eller lese/skrive data, vet den ikke om den skal hente fra RAM eller cache. Hvis det prosessoren ber om ligger i cache, kalles det for **cache hit**.
- Figur til høyre: CPUen ber om å få innholdet i lokasjon 1881



- Hvis data ikke ligger i cache, får man en **cache miss**
- Figur til høyre: CPU'en ber om lokasjon 9872
- Cachekontrolleren finner ut at denne ikke finnes i cachen
- Blokken som starter på 9870 leses inn fra RAM
- Blokken som starter på 6410 overskrives for å gi plass til 9870

Read/write hit/miss

- Cache-kontrolleren må håndtere 4 tilstander av aksess av cache
- Read hit**
 - Det skal leses fra hukommelsen og blokken med ordet befinner seg i cache
 - Data leveres med en gang fra cache'n til CPU'en
- Read miss**
 - Det skal leses fra hukommelsen, men blokken det skal leses fra er ikke i cache
 - Cache-kontrolleren må bestemme
 - Hvilken blokk i cache som kan/skal overskrives
 - Kopiere inn riktig blokk fra RAM til cache
 - Levere ordet til CPU'en
- Ved begge **read-operasjoner** vil innholdet i RAM og innholdet i kopien av blokkene i cache være identiske -> **OK**



Read/write hit/miss (forts)

- **Write hit**
 - Det skal skrives til hukommelsen og blokken med lokasjonen det skal skrives til ligger i cache
 - Data skrives til riktig lokasjon i blokken i cache
- **Write miss**
 - Det skal skrives til hukommelsen og blokken med lokasjonen det skal skrives til ligger ikke i cache
 - Cache-kontrolleren må bestemme
 - Hvilken blokk i cache som kan/skal overskrives
 - Kopiere inn riktig blokk fra RAM til cache
 - Besørge skrivning til riktig lokasjon i cache
- Ved begge **write-operasjoner** vil innholdet ikke innholdet i RAM og innholdet i kopien av blokkene i cache være identiske -> **ikke OK**

24.04.2006

INF 1070

9

2005



Mer om write hit/miss

- Hvis det bare skrives til cache og ikke RAM, vil ikke RAM inneholde gyldige data
- Hvis en blokk det er skrevet til kastes ut fordi cache er full, mister man data
- To strategier benyttes for å håndtere write-operasjonen korrekt:
 - **Write-through:**
 - Etter hver gang det skrives til en blokk, skrives innholdet i blokken også tilbake til RAM
 - **Fordel:**
 - Enkelt å implementere i cache-kontrolleren
 - **Ulempe:**
 - Hvis det skal skrives flere ganger rett etterhverandre til samme blokk må prosessoren vente mellom hver gang

24.04.2006

INF 1070

10

2005



Mer om write hit/miss (forts.)

- **Write-back**
 - Innholdet i en blokk i cache kopieres kun tilbake til RAM hvis blokken skal overskrives i cache og det har blitt skrevet til den i cache
 - For raskt å detektere om en blokk har blitt skrevet til benyttes et kontroll-bit i cache (kalles **dirty-bit**)
- **Fordel:**
 - Gir ingen hastighetsreduksjon ved flere påfølgende write-operasjoner til samme blokk siden CPU'en ikke trenger vente mellom hver skrivning
- **Ulempe:**
 - Hvis det skal skrives flere ganger rett etterhverandre til samme blokk må prosessoren vente mellom hver gang.
 - Hvis arkitekturen støtter direkte lesing/skriving mellom RAM og Input/Output-enheter uten å gå via CPU'en (kalt Direct memory Access eller DMA), risikerer man inkonsistente data hvis ikke DMA-kontrolleren kommuniserer med cache-kontrolleren

24.04.2006

INF 1070

11

2005



Cache arkitektur

- Cache kan plasseres på (minst) to måter i forhold til CPU og RAM
- De to organiseringene påvirker hvordan lesing håndteres ("read architecture")
 - "Read architecture" og "write policy" er det viktigste karakteristika til cache
- Cache koherens: Hvorvidt innholdet i cache og RAM er identisk
- For å bevare koherens kan cache gjøre
 - "Snoop": Cache overvåker adresselinjer for å se etter transaksjoner som angår data den sitter på
 - "Snarf": Cache tar data fra datalinjer og kopierer dem inn
- Inkohrens kan være
 - "Dirty data": Data i cache er modifisert, men ikke tilsvarende data i RAM
 - "State data": Data i RAM er modifisert, men ikke tilsvarende data i cache.

24.04.2006

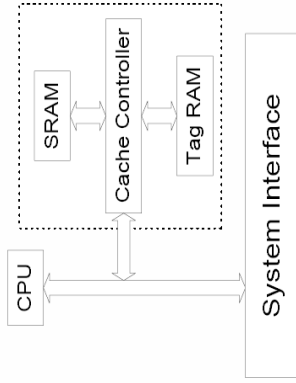
INF 1070

12

2005

Look-aside arkitektur (1)

- Cache er plassert parallelt med RAM (system-interface)
- Både RAM og cache leser adresselinjene samtidig

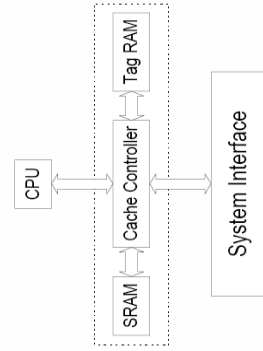


Look-aside arkitektur (2)

- Når prosessorer starter en lesing, sjekker cache adressen (snooping) for å se om det er en "read hit"
- Hvis "read hit":
 - Cache svarer prosessoren og stopper videre buss-syklus for å hindre lesing fra hovedminnet (RAM)
 - Hvis "read miss":
 - Hovedminnet (RAM) vil besvare henvendelsen fra prosessoren. Cache "snarfer" data slik at de er i cache neste gang prosessoren ber om dem.
- Fordeler:
 - Mindre komplisert enn look-through
 - Bedre responstid (RAM og CPU ser samme buss-syklus)
- Ulempe
 - Prosessoren kan ikke aksessere cache hvis en annen enhet aksesserer RAM

Look-through arkitektur (1)

- Cache sitter fysisk mellom CPU og internbussen
- Cache-kontrolleren vil avgjøre om buss-syklus skal slippe videre til RAM



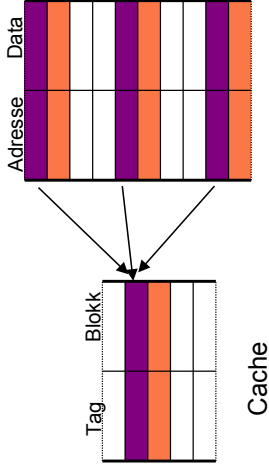
Look-through arkitektur (2)

- Når prosessoren starter en minne-aksess vil cache-kontrolleren sjekke om det er en "read-hit"
- Hvis "read hit"
 - Cache leverer data til prosessoren uten aksess til hovedminnet
- Hvis "read miss"
 - Buss-syklusen sendes videre til hovedminnet
 - Hovedminnet (RAM) vil besvare henvendelsen fra prosessoren. Cache "snarfer" data slik at de er i cache neste gang prosessoren ber om dem
- Fordeler:
 - Prosessoren er isolert fra resten av systemet og kan jobbe uavhengig (viktig i fler-prosessor systemer)
- Ulemper
 - Mer komplisert siden cache-kontrolleren må håndtere all minneaksess
 - Trengere ved cache-miss siden cache og RAM sjekkes i sekvens

Indeksering av cache og blokkoverskriving

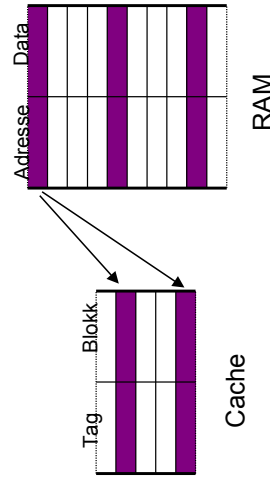
- To gjenstående utfordringer ved design av cache:
 - **Hvordan** cache-kontrolleren indekserer (dvs leter fram) riktig blokk ved lesing/skriving
 - **Hvilken** blokk som skal overskrives/kastes ut når cache er full
- Det finnes tre ulike indekseringsteknikker for blokker i cache:
 - Direkte-avbildet,
 - Set-assosiativ
 - Full assosiativ.
- Metodene skiller seg fra hverandre ved
 - Hvor enkle de er å implementere,
 - Hvor raskt det går å finne en blokken/tag
 - Hvor god utnyttelse man får av cache
 - Hvor ofte man får cache miss.

- **Direkte-avbildet:** En bestemt blokk fra RAM kan bare plasseres i en bestemt blokk i cache. Flere RAM-lokasjoner må "konkurrere" om samme blokk i cache.



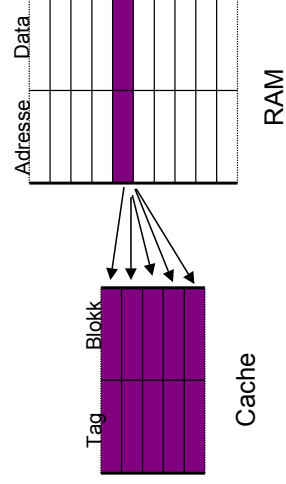
- **Fordel:** Lett å sjekke om riktig blokk finnes i cache: Sjekk kun ett tag-felt og se om denne inneholder blokken man leter etter.
- **Ulempe:** Høyere miss-rate (selv om det er ledig plass andre steder, kan en blokk kun plasseres ett bestemt sted)

- **Set-assosiativ:** En bestemt blokk fra RAM kan plasseres i et begrenset antall blokk-lokasjoner i cache. En tilgjengelig lokasjon kallen en "way"



- **Fordel:** Lett å sjekke om riktig blokk finnes i cache: Søk gjennom et begrenset antall tag-felt og se om blokken man leter finnes i cache. Bedre utnyttelse av cache fordi en blokk kan plasseres flere steder
- **Ulempe:** Høyere miss-rate og lenger søketid enn ved direkte-avbildet (med mindre man søker i parallell gjennom tag-feltene).

- **Full assosiativ:** En bestemt blokk fra RAM kan plasseres hvorsomhelst i cache



- **Fordel:** Cachen utnyttes meget godt, og har minst sjanse for cache miss av de tre metodene
- **Ulempe:** Søket for å finne en blokk kan bli tidkrevende, og man lager mekanismer for å forenkle søk, f.eks hashing. Dette kompliserer cache-kontrolleren og krever ekstra hardware. Brukes ved cache < 4KB

- Ved cache miss må noen ganger en eksisterende blokk kastes ut for å gi plass til en ny blokk. Hvilken blokk som fjernes kan ha stor betydning for hastigheten til programmet som eksekveres.

- **LRU (Least Recently Used):**

- Blokken som har ligget lengst i cache *uten å ha blitt skrevet til eller lest* skrives over, pga. lokalitetsprinsippet
- Ren LRU benyttes sjelden fordi det medfører mye administrasjon som i seg selv er tidkrevende (bl.a må et tidsstempel oppdateres hver gang en blokk aksesseres)

- **Random:**

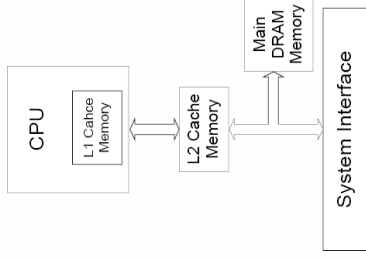
- Kaster man ut en tilfeldig blokk når man trenger å frigj plass til en blokk.

- **Hybrid:**

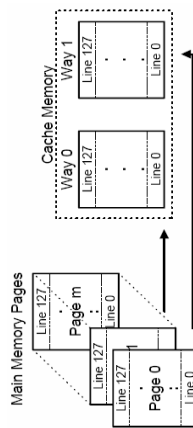
- Deler inn blokker i tidsgrupper, og kaster så ut en tilfeldig valgt fra den gruppen som har ligget lengst i cache uten å bli brukt.

Cache i Pentium III-arkitekturen (1)

- Cache i Pentium har to nivåer, L1 (on-chip) og L2 (off-chip)
- Nivået angir rekkefølgen de aksesserer, ikke hvor de fysisk er plassert
- Bussbredden til L1 er 256 bit, mens den er 64 bit til L2 (endret i senere versjoner)
- Separat data- og instruksjons-cache er hver på 8 KB

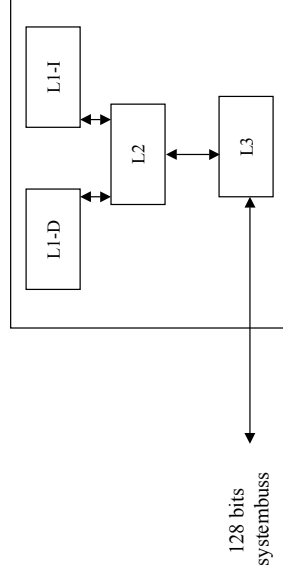


Cache i Pentium III-arkitekturen (2)



- Både L1 og L2 er 2-veis assosiativ
- Cache linjebredden er 256 bits, og fylles ved 4 påfølgende lese-operasjoner
- Siderstørrelsen er 4K eller 128 linjer (128 ways = sider i set-assosiativ cache)
- Mulig å styre i software om cache skal være "write-back" eller "write-through"
- Mulig å disable L1 cache i software
- L1 er look-through, mens L2 er look-aside

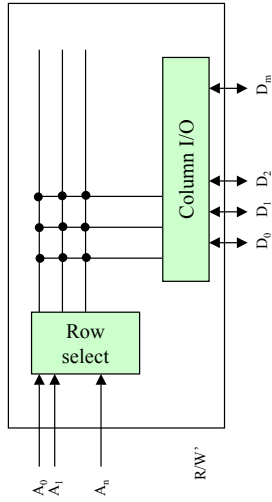
Cache i Itanium-arkitekturen (1)



- L1-D og L1-I er 4-veis set-assosiativ og hver på 16 KB
- Kan håndtere 2 load og 2 store samtidig
- L1-D håndterer ikke flyttall, dette skjer i L2 somer på 256 KB og 8-veis set-assosiativ
- L3 er 1.5 eller 3 MB, og pipelinet for å takle opptil 8 load/store operasjoner

RAM-typer

- RAM kan implementeres på ulike måter, avhengig av bruksområde
- De to vanligste hovedtypene er **Statisk RAM** og **Dynamisk RAM**
- Statisk RAM:** Hver enkelt lagercelle er laget av en RS-flipflop med transistorer
- Fordeler:**
 - . Kort aksesstid
 - . Holder lagret verdi så lenge det er spenning på chip'en
- Ulemper:**
 - . Hver celle trenger 5-10 ganger mer plass enn dynamisk RAM
 - . Dyrere per bit



24.04.2006

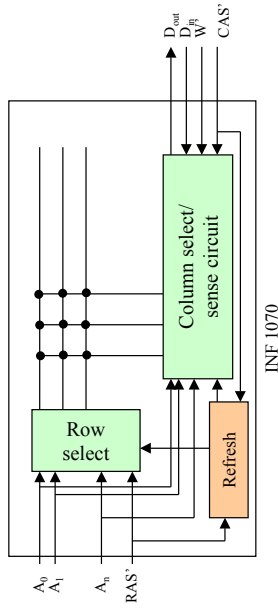
INF 1070

25

2005

RAM-typer (forts)

- Dynamisk RAM:** Hver enkelt lagercelle er laget av en kondensator
- Fordeler:**
 - . Mer kompakt enn SRAM
 - . Billigere per bit enn SRAM
- Ulemper:**
 - . Innholdet blir borte etter kort tid (15ns)
 - . Trenger ekstra logikk for å friske opp innholdet slik at lagringen blir permanent



24.04.2006

INF 1070

26

2005

ROM

- I motsetning til RAM kan Read-Only Memory (ROM) kun skrives til et begrenset antall ganger
- Den vanligste formen er ROM som enten
 - . prefabrikeres (hard-wired) med et fast innhold i hver celle
 - . programmeres én gang av brukeren (PROM)
- EPROM er en type som kan nullstilles og programmeres igjen
 - . Ultrafiolett lys brukes til å nullstille koblingspunkter
- EEPROM bruker elektrisk spenning istedenfor UV-lys for å nullstille
- NVRAM er svært utbredt i dag
 - . Kan re-programmeres ca 100 000 ganger
 - . Skrivning er flere størrelsesordener langsommere enn lesing
 - . Brukes i bl.a. MP3-spillere og mobiltelefoner

24.04.2006

INF 1070

27

2005