UNIVERSITETET
I OSLO

Dagens temaer

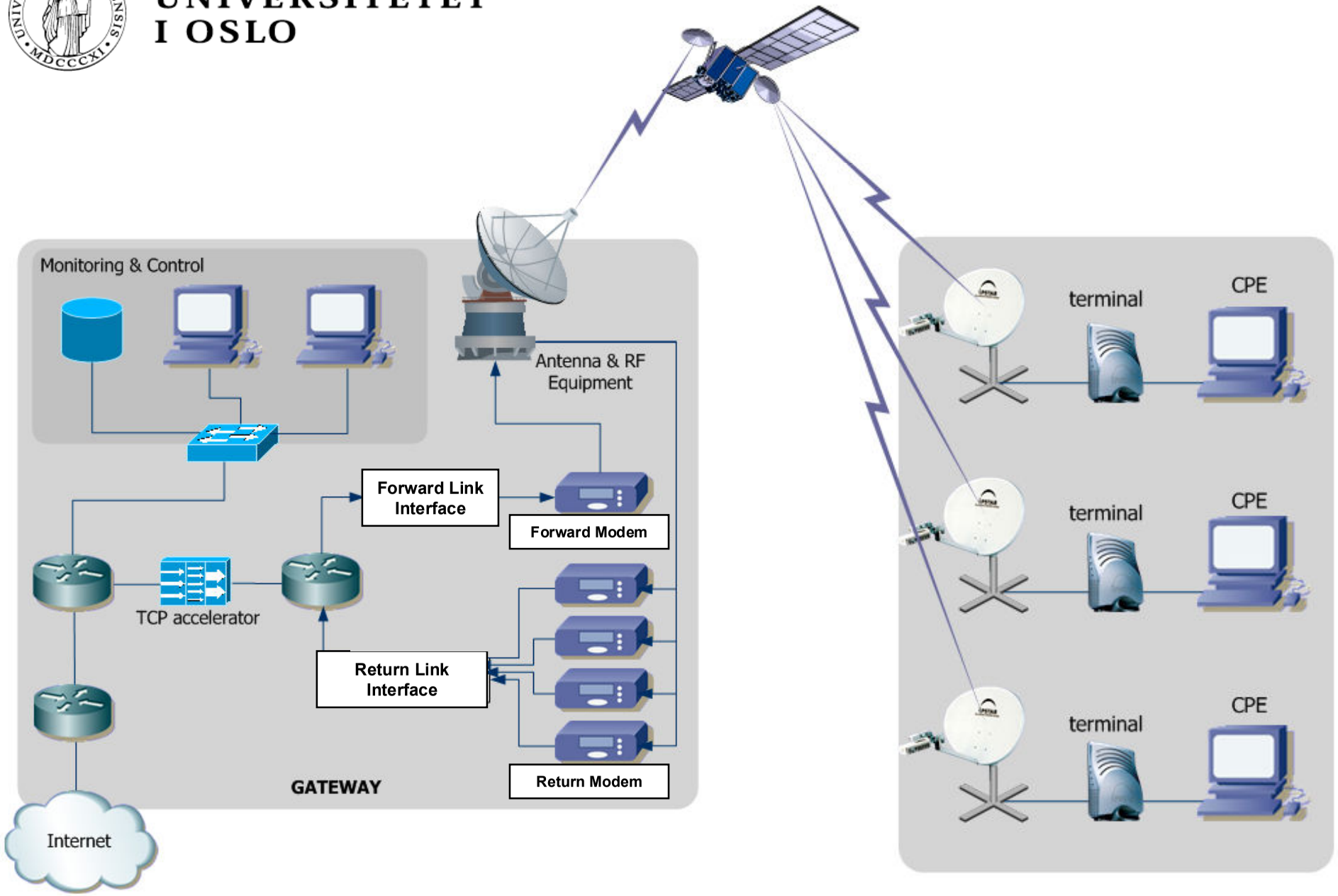
- Praktiske eksempler på bruk av assembler/C/arkitektur
- Flere teknikker for å øke hastigheten
- Cache-hukommelse del 1 (fra kapittel 6.5 i ”Computer Organisation and Architecture”)
  - Hvorfor cache
  - Grunnleggende virkemåte
  - Direkte-avbildet cache

2005

UNIVERSITETET
I OSLO

Praktisk anvendelse: Satellittkommunikasjon

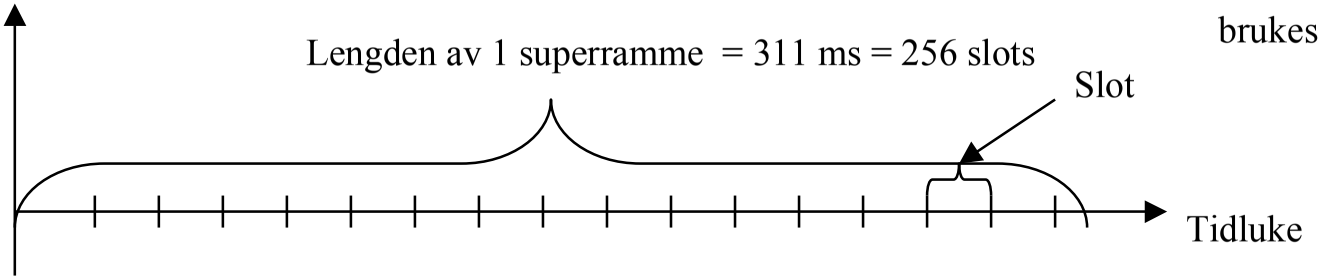
- Satellitt brukt til kommunikasjon gjør at man har trådløs adgang til telefoni og Internet hvor som helst i verden (nesten).
- Mye brukt av handelsflåte, forsvar, FN og etterhvert også som eneste bredbåndsalternativ i grisgrendte strøk
- Satellitter befinner seg i varierende høyde, fra ca 650 km til 36000 km over jorden
- Den største forskjellen mellom trådbasert og satellittbaserte systemer er den lange tidsforsinkelsen (ca 250 ms én vei for geostasjonære satellitter)
- En satellitt koster ofte flere milliarder kroner, og satellittkommunikasjon er som regel mye dyrere enn bakkebasert utstyr
- Et system for satellittkommunikasjon består av tre deler:


Monitoring & Control
Antenna & RF Equipment
Forward Link Interface
Forward Modem
TCP accelerator
Return Link Interface
Return Modem
GATEWAY
Internet
terminal
CPE
terminal
CPE
terminal
CPE

Forward Link Interface

- Radiolinken opp til satellitten har endel parametre som må settes
  - Modulasjon (frekvens)
  - Koding (antall ekstra bit som må brukes for å overføre et ”nytte-bit”)
  - Forsterking (sendeeffekt som skal brukes ved overføring av bit’ene)
- Før en IP-pakke kan sendes inn på modemet i forover-retningen, må innholdet re-formatteres, og del tilleggsinformasjon må beregnes for at pakken skal kunne sendes korrekt:
  - Plass i tidluken(e)
  - Modulasjon, koding og forsterkning (MKF)

1 slot = 4096 symboler
Mellom 2 og 5 symboler
brukes for å overføre 1 byte

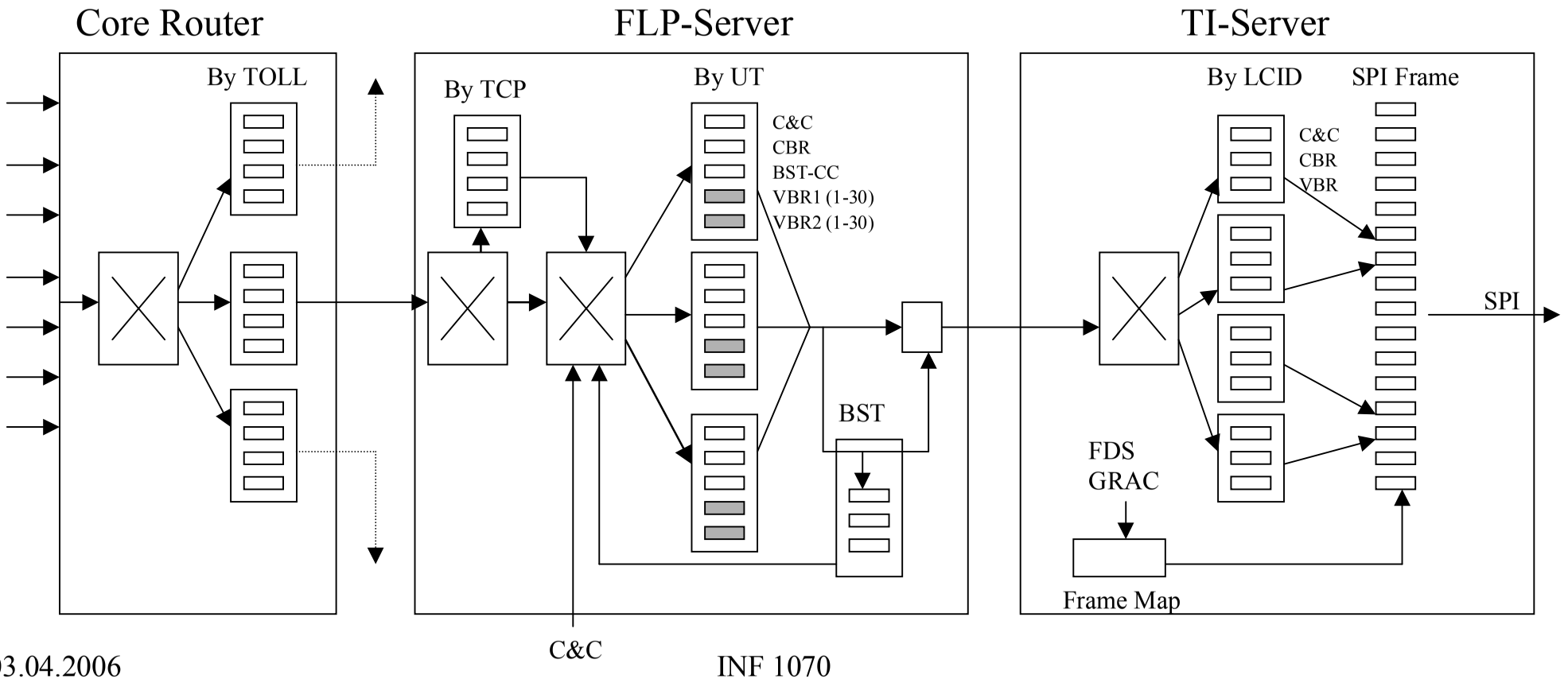


- En sluttbrukerhar typisk kjøpt en viss båndbredde (eks 1024 forover, 256 i retur)
- Man kjøper også en bestemt Service Level Agreement (SLA) for å få en bestemt QoS, f.eks for
  - VoIP (må ha lav forsinkelse, liten/ingen variasjon i forsinkelse og konstant båndbredde)
  - Streaming (tåler forsinkelse, men uten variasjon, og høy båndbredde)
  - Filoverføring (tåler både forsinkelse, variasjon i forsinkelse, og variasjon i båndbredde)
- Variasjon i atmosfæriske forhold gjør at man må bruke mer robust koding
- Hvis mange brukere er på systemet, konkurrerer man om satellittkapasiteten

- For eiere og brukere av systemet er det viktig at satellitten brukes så optimalt som mulig
  - Eierne tjener mer penger, og kan senke prisene til
  - Sluttbrukere som forventer å få en bestent tjeneste til en gitt pris
- Ved dårlig resursshåndtering vil satellitten utnyttes dårlig, samtidig som tjenesten blir dyr og kvaliteten mindre god.
- Hovedoppgave for 'Forward Link Interface'
  - For hver innkommende IP-pakke beregne hvilke(n) slot og superramme pakken skal plasseres i, og med hvilke MKF verdier
  - Plassere så mange IP-pakker som mulig innenfor en superramme slik at alle SLA'er overholdes
  - De to foregående jobbene må beregnes innen 311 millisekunder: Svært tidskritisk, og noen deler må skrives i assembler
  - Spesialdrivere må også skrives i assembler

Arkitektur på Forward Link Prosessor

- Består av 2 (kalt FLP og TI) stykker 19" industri-PC'er med Dual Pentium 2.4 GHz Xeon prosessorer.
- Kommunikasjon mellom FLP og TI er gigabit Ethernet, og mellom TI og modem er det SPI (standard for video-overføring)



UNIVERSITETET
I OSLO

Separate foiler som ikke deles ut

- Foilene som vises er fra Nera internt bruk og kan av kommersielle henyn ikke deles ut
- Foilene viser prosessen med å definere opp krav til ny hardware-plattform, og hvordan software-moduler fordeles på ulike hardware enheter

UNIVERSITETET
I OSLO

Cache

- Pipelining er viktig for å øke antall instruksjoner som utføres pr sekund.
- Andre teknikker som øker hastigheten ytterligere er
  - Flere CPUer i en maskin (multiprosessor)
  - Økt klokkehastighet
  - Bred ordlengde på instruksjoner og i datapath
  - Raskere disker
  - Hurtigere og større RAM
- Noen teknikker er ren forbedring i teknologi, mens andre er forbedringer av selve arkitekturen.

UNIVERSITETET
I OSLO

Multiprosessorer (parallellprosessorer)

- Ide: Istedenfor å la én CPU utføre instruksjoner, lar man flere CPUer samarbeide om den samme jobben.
- Gir en teoretisk hastighetsøkning direkte proporsjonal med antall ekstra CPUer: det går *n* ganger raskere med *n* CPUer enn med én CPU.
- I praksis ikke mulig av flere årsaker:
  - Ikke alltid mulig å dele opp et problem i like store deler som kan løses uavhengig av hverandre.
  - Det kreves administrasjon og koordinering av programeksekveringen før, under og etter at jobben er fordelt på de ulike CPUene, dvs ekstra overhead
- Endel applikasjoner egner seg for lastdeling, f.eks server-applikasjoner, dvs enkelt å parallellisere oppgavene

UNIVERSITETET
I OSLO

Økt klokkehastighet og større ordlengde

- Økt klokkehastighet reduserer tiden hver enkelt instruksjon tar (Resultatet av teknologisk utvikling.)
- Hastigheten til alle deler av en datamaskin øker ikke like raskt:
  - Interne data/adressebusser
  - Nettverk
- Klokkehastigheten til en CPU øker mye raskere enn lese/skrivehastigheten til hukommelsen, mao: Effektive hastighets-forbedring blir langt mindre.
- Bredere datapath og ordlengde gjør det mulig å behandle større tall i en operasjon og å lese/skrive mer data fra hukommelsen i én operasjon.
- Det er en øvre grense for hvor mange bit som kan behandles i en operasjon (typisk i dag er 64 eller 128 bit).

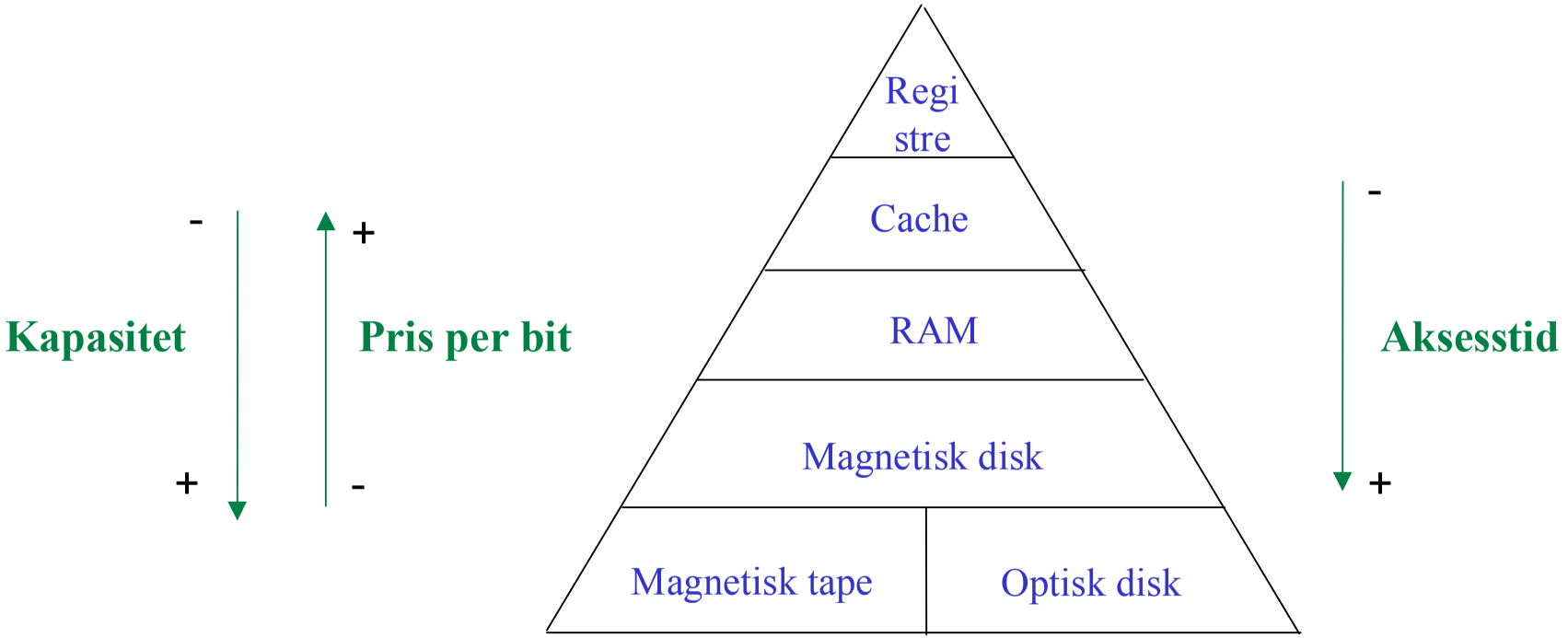
UNIVERSITETET
I OSLO

Raskere disker

- Programmer trenger å aksessere disker av og til ved f.eks innlesning og skriving av data, ved oppstart og virtuell hukommelse
- En harddisk ca 100 000 ganger langsommere enn en CPU, dvs at en CPU kan gjøre 100 000 instruksjoner mens en harddsik gjør én lese/skrive-operasjon.
- Teknikker finnes for å lese store blokker av data ad gangen, prøve å gjette hvilke data som skal leses neste gang og lagre disse i hurtigminne, etc
- Harddisker blir både raskere og får mer kapasitet, men allikevel vil hyppig diskaksess være et betydelig problem hvis hastighet er viktig.

Større og raskere hurtigminne

- Det optimale ville være ***ubegrenset tilgang*** på ***billig*** hurtigminne ***like raskt*** som prosessoren selv. Henting av instruksjoner, og lesing/skriving av data ville gå like raskt som registeraksess.
- I praksis må man velge hva som er viktigst av pris, hastighet og kapasitet



UNIVERSITETET
I OSLO

Bruksområder for de ulike hukommelsestyper

- **Registre:** Integrert på CPU'en, relativt få (32-64 stykker) med like mange bit i hver som maskinens ordbredde
- **Cache:** Mellomlager som ligger inne på (L1) eller i nærheten av (L2, L3) CPU'en, typisk kapasitet fra 8 KiloByte (L1) til 512 (L2) KiloByte, og noen MegaByte (L3).
- **RAM:** Internt på hovedkortet i nærheten av CPUen, størrelse opptil mange GigaByte.
- **Magnetisk disk:** Ekstern eller intern lagringsenhet i maskinen, med kapasitet opptil TeraByte.
- **Magnetisk tape**: Sekvensielt medium med opptil 10-talls sekunders aksesstid.
- **Optisk disk:** (CD-ROM, CD-RW og DVD). Billig eksternt lagringsmedium med kapasitet opptil flere GigaByte (DVD).

UNIVERSITETET
I OSLO

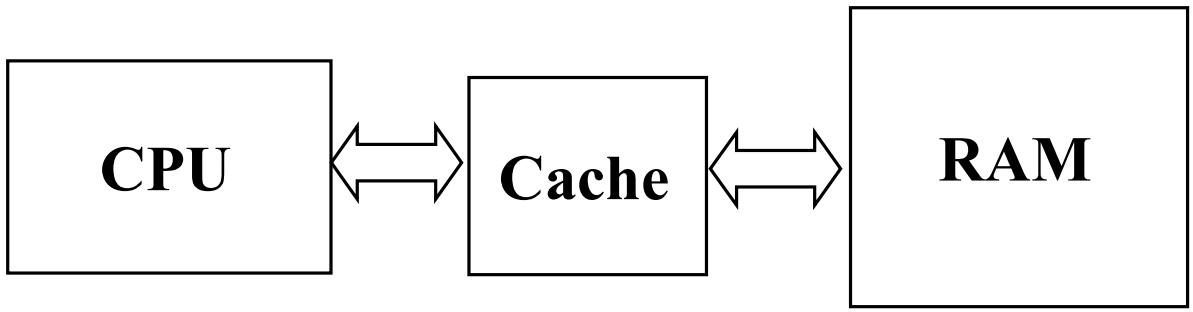
Aksesstider

- **Registre:** En klokkesykel, dvs like raskt som resten av prosessoren (nano-sekunder).
- **Cache:** Samme størrelsesorden som interne registre, men allikevel noe langsommere (avhengig av L1 eller L2/L3).
- **Hovedminne**: Flere titalls nano-sekunder.
- **Magnetisk disk:** 10 000 til 100 000 ganger langsommere aksesstid, dvs flere millisekunder. For å finne data må disken rotere til riktig posisjon
- **Magnetisk tape:** Sekvensielt medium med opptil flere titalls sekunders aksesstid (man må i verste fall spole gjennom hele tapen før man finner det man leter etter.
- **Optisk disk:** (CD-ROM, CD-RW og DVD): Opptil flere sekunders aksesstid. For å finne data må platen rotere til riktig posisjon.

UNIVERSITETET
I OSLO

Cache

- Ønsker så mye og rask hukommelse som mulig tilgjengelig for et program under eksekvering, både for instruksjoner og data
- Cache-minnet er logisk sett plassert mellom CPUen og RAM slik det er vist i figuren under:
- Innholdet i cache vil alltid være et subsett av innholdet i RAM
- Fysisk er cache enten integrert på CPU'en eller plassert rett ved siden av (eller begge deler)



UNIVERSITETET
I OSLO

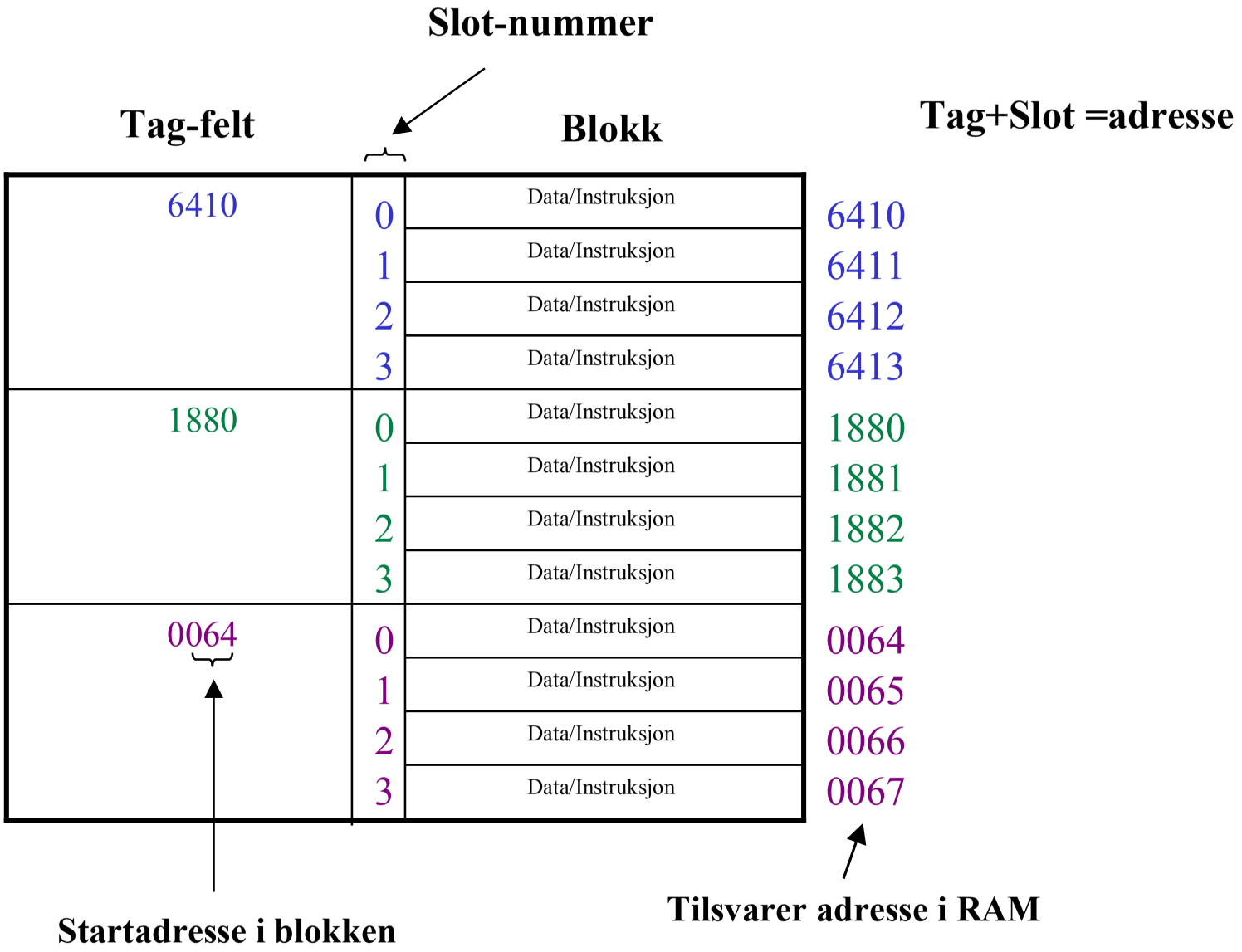
Cache (forts.)

- Kapasiteten til cache ligger i mellom det interne i CPUen og RAM. Typiske verdier er fra 8 KB til 2 MB (Finnes i flere nivåer kalt L1, L2 og L3)
- Siden cache'n er nesten like rask som CPUen, trenger ikke CPU'en å stoppe opp og vente i mange klokkesykler på at RAM skal levere data.
- CPU'en har ingen "bevissthet" om at det finnes cache og ser ikke forskjell på 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 klokkesykler spiller ingen rolle annet enn for hastigheten.

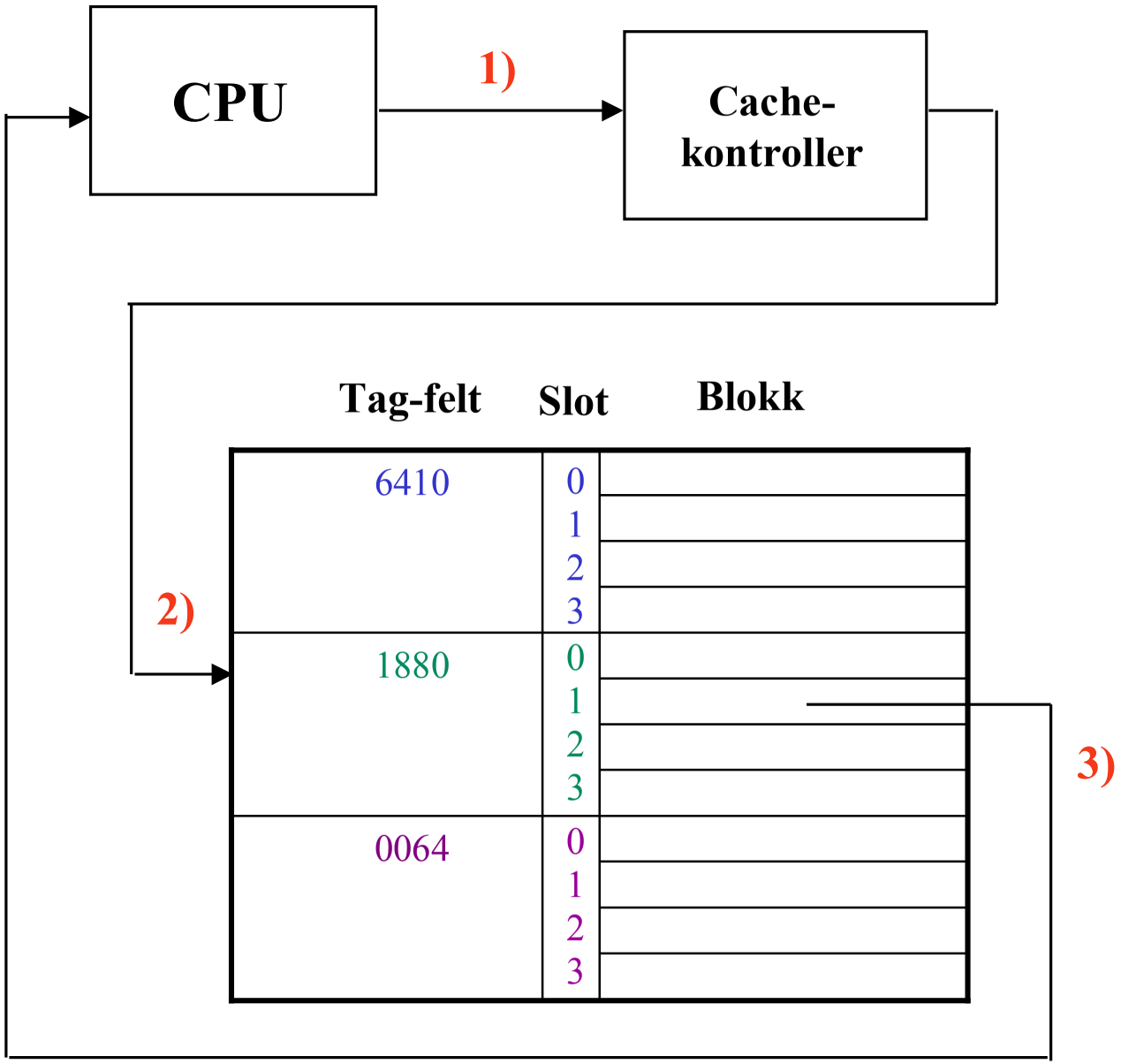
UNIVERSITETET
I OSLO

Cache (forts.)

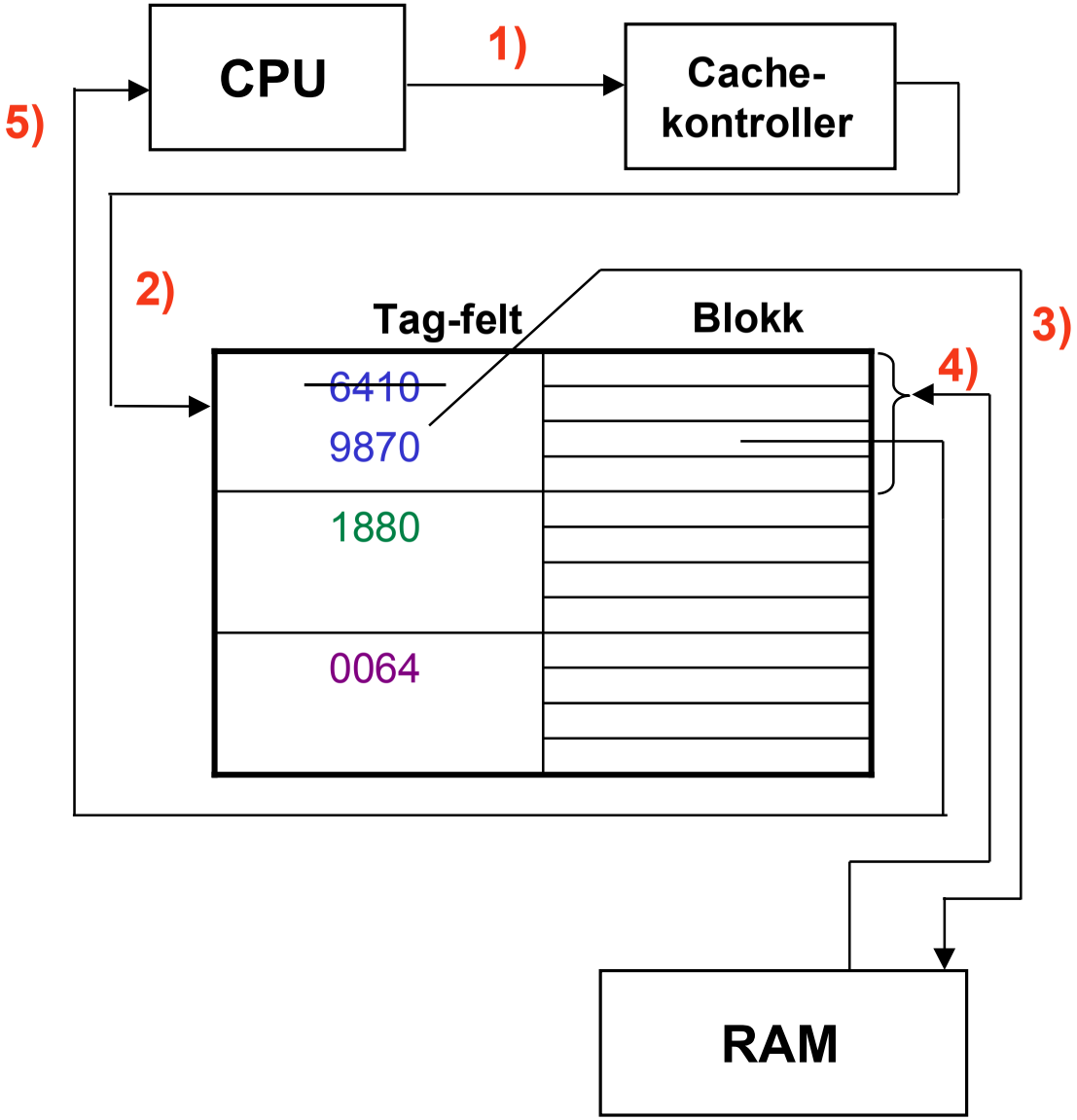
- Siden cache'n er mindre enn hurtigminnet (RAM) består mesteparten av administrasjonen av cache i å velge ut hvilken del av programmet og hvilke data som skal ligge i cache, og hvilke som må ligge i RAM.
- Cache benytter seg av lokalitetsprinsippet:
  - Instruksjoner/data aksesseres som regel sekvensielt,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


Slot-nummer
Tag-felt
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
Tilsvarer 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



UNIVERSITETET
I OSLO

Read/write hit/miss

- Cache-kontrolleren må håndetere 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**

UNIVERSITETET
I OSLO

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 skriving 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**

UNIVERSITETET
I OSLO

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

UNIVERSITETET
I OSLO

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'et)
  - **Fordel:**
    - Gir ingen hastighetsreduksjon ved flere påfølgende write-operasjoner til samme blokk siden CPU'en ikke trenger vente mellom hver skriving
  - **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