# Obligatorisk øving 4 i datateknikk

Dette er den fjerde obligatoriske øvingen i datateknikk.

## Oppgave 1 Parallellitet på instruksjonsnivå

1. Tidligere var det «god skole» at kompilatorer samlet instruksjoner som er avhengige av hverandre i kontinuerlige blokker. På moderne prosessorer er ikke dette nødvendigvis særlig lurt lengre.

* Hvorfor var dette lurt på eldre prosessorer?

Svar:

Gamle prosessorer utførte bare en instruksjon om gangen og i et tregere tempo enn på nye versjoner, derfor var det mer effektivt å samle opp ting som var avhengige av hverandre slik at man kan alltid hente neste instruksjon fra cachen og spare tid på arbeidet.

* Hvorfor er det ikke lurt på nyere prosessorer?

Svar:

Nyere prosessorer utfører flere instruksjoner paralellt, derfor blir ofte rekkefølgen på instruksjonene stokket om slik at den ska få gjort mest mulig arbeid kontinuerlig og ikke ha noen stopp i arbeidet ved at den må vente på at en annen instruksjon skal bli ferdig.

1. Gode kompilatorer tar hensyn til egenskaper med prosessoren, og genererer kode som er optimalisert for den prosessoren. Anta at følgende instruksjoner skal utføres:

a = 1  
b = 1  
a = a + b  
c = a + 2  
p = 10  
q = 10  
p = p + q  
r = q + 10

Her er a, b, c, p, q, r variabler, og vi antar at utrykkene er så enkle at hver linje utgjør én instruksjon.

* Hvorfor kan dette være en uheldig rekkefølge på en superskalar prosessor?

Svar:

Siden flere av instruksjonene er avhengige av hverandre fører dette til at de paralelle prosessene i en superskalar prosessor får pauser i arbeidet sitt for å vente på andre skal bli ferdig, det er da mer ideelt at to og to separate instruksjonslinjer kan utføres uavhengig.

* Foreslå en instruksjonsrekkefølge som ikke endrer sluttresultatene av program­sekvensen, men som er optimalisert for en superskalar prosessor av grad 2. Vær nøye med å begrunne det du gjør.

Svar:

siden to påfølgende instruksjoner ikke kan være avhengige av hverandre for å få optimal ytelse i en superskalar grad 2, må vi endre rekkefølgen til f.eks.

A=1

P=10

B=1

Q=10

A=A+B

P=P+Q

C=A+2

R=Q+10

hvis det utføres i denne rekkefølgen(satt opp i par siden prosessoren jobber parvis), så får vi en konstant flyt av instruksjoner som kan utføres i hver pipeline.

## Oppgave 2 Minneaksess på moderne prosessorer

*Denne oppgaven krever at du husker stoffet om Systemarkitektur i tillegg til stoffet om moderne prosessor­arkitektur. Om nødvendig må du lese de aktuelle leksjonene på nytt.*

Anta en prosess med flere superskalare kjerner. En slik prosessor kan fint utføre én instruksjon pr klokkesyklus over relativt lang tid hvis man unngår minneaksess.

Anta at prosessoren har en (intern) klokkefrekvens på 3 GHz.

Anta at minnet er av typen DDR3 1600 med følgende timing 8-8-8-24.

Vi skal nå sammenligne prosessorytelse og minneytelse.

1. *Prosessoren:*

* Hvor lang tid tar hver instruksjon når prosessoren utfører en instruksjon pr klokkesyklus?

Svar:

0.333 ns for hver instruksjon

1. *Minne:*

* Hvor lang tid (i ns) går det mellom hver klokkesyklus på minnebussen?

Svar:

DDR3 1600 har 1600Mhz burstfrekvens, så den vanlige bussfrekvensen er på 800Mhz, som da tilsvarer 800million klokkesekvenser i sekundet.

Selve burstsekvensen tilsvarer da 1.6 milliarder klokkesekvenser i sekundet.

Det tar da 1.25ns for hver klokkesyklus på minnebussen i vanlig fart

* Hvor lang tid (i ns) går det mellom hver overføring (hvert ord) i en *burst* med denne minnetypen?

Svar:

i burst tar det da 0.625ns for hver overføring.

* Hvor lang tid (i ns) tar det å hente ut nye data når rett rekke allerede er aktivert?   
  (Enkelte artikler kaller denne tiden for aksesstiden til minnet. Det stemmer ikke med det vi kalles aksesstiden. Den tiden som beregnes i dette tilfellet krever jo at rett rekke allerede er aktivert)

Svar:

her må vi da se på timing-angivelsen til minnet, som er 8-8-8-24.

I dette tilfellet er i da på rett rekke fra før, så vi trenger kun å tenke på det første tallet for å finne rett lokasjon, og får da 8\*1.25ns = 10ns for å hente ut nye data.

* Hvor lang tid (i ns) tar en minne­aksess i verste fall?   
  (I verste fall betyr at minnet må begynne med å aksessere rett bank, deretter rett rad og så videre. Det er dette som er aksesstiden til minnet, altså tiden det tar å aksessere en vilkårlig plass i minnet. )

Svar:

I dette tilfellet må vi da gjennom hele timingsekvensen(8+8+8+24=48)

og får da 48\*1.25ns = 60ns som wrost-case for å finne noe i minnet.

* Sammenlign aksestiden du fant i forrige spørsmål med aksesstiden som kompendiet opererer med for DRAM. Er det overensstemmelse?

Svar:

ja, siden det i kompendiet står 35-70 så stemmer dette ganske bra.

1. *Sammenligninger:*

* Hvor mange instruksjoner utføres mellom hver enkelt overføring (hvert ord) i en *burst*?

Svar:

siden en burst tar 0.625ns og prosessoren utfører en isntruksjon hvert 0.33ns så blir det utført ca 2 instruksjoner for hver overføing i burst-modus.

* Hvor mange instruksjoner kan prosessoren utføre i løpet av den tiden en minne­aksess i verste fall tar (i verste fall betyr at minnet må begynne med å aksessere rett bank, deretter rett rad og så videre).

Svar:

i verste fall bruker minnet da 60ns, så det kan utføres ca 181 instruksjoner før minnet finner rett instruksjon.