# 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:

Eldre prosessorer var mest effektive hvis instruksjonene var dataavhengig av hverandre. Grunnen til dette er at eldre prosessorer hadde få registre som gjorde det vanskeligere å beholde nyttig data i registrene over lengre tid. Derfor er det mest effektivt hvis instruksjonene som er dataavhengig av hverandre kjøres sekvensielt.

* Hvorfor er det ikke lurt på nyere prosessorer?

Svar:

Nyere prosessorer har en annen måte å prosessere instruksjoner. De kjører ulike deler av instruksjoner (mikrooperasjoner) i parallell (pipeline). Moderne prosessorer er superskalare som betyr at de kan kjøre flere pipelines parallell. Ulempen med instruksjoner som er dataavhengige av hverandre som kjøres sekvensielt, er at mikrooperasjoner vil «krasje» med hverandre. Man kan f.eks. ikke lese og skrive til minnet samtidig, derfor må man vente, og da utsettes alle instruksjonene. Derfor er rekkefølgen på instruksjoner viktig i moderne prosessorer.

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:

Superskalare prosessorer utfører mikrooperasjoner parallelt i en såkalt «pipeline», som dermed gjør at instruksjonene kan bli behandlet og utført mye fortere. Derimot hvis man har en uheldig rekkefølge av instruksjoner så vil flere mikrooperasjoner «krasje» med hverandre. Altså det blir ventetid imellom. Slike situasjoner kalles «Hasard».

Problemet med rekkefølgen av instruksjoner ovenfor er når prosessoren skal lese fra minnet på instruksjon nr. 3 (a=a+b), vil ikke prosessoren ha lagt data fra AC registeret i minnet. Derfor må man vente til at dette er utført før man leser den lokasjonen.

* 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:

En måte å optimalisere måten instruksjonene kjøres på er ved å endre rekkefølgen på dem. Problemet ligger i at en instruksjon trenger data før de er lagret i minnet. Man kan vente, eller så kan man legge inn en annen instruksjon imellom som ikke er avhengig av data fra minnet.

a = 1

p = 10

b = 1

q = 10

a = a+b

p = p + q

c = a+2

r = q+10

## 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:

Frekvens = 1 / Periode

F = 1 / 3\*109 = 3,33\*10-10 = 0,333ns

1. *Minne:*

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

Svar: Frekvensen på minnebussen er 1600 Mhz

F = 1 / 1,6\*109 = 6,25\*10-10 = 0.625ns

Legg merke til at minnet er av typen DDR (Double Data Rate), derfor vil den aktuelle klokkefrekvensen være 0.625ns\*2 = 1.25ns.

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

Svar:

Med DDR overføres to ganger pr. klokkepuls. Derfor er det forskjell på klokkefrekvensen på minnebussen og klokkefrekvensen på en overføring i en burst med dette minnet. Klokkefrekvensen ved burst ble utregnet i forrige oppgave 0.625ns.

* 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:

Tiden det tar å hente ut ny data gitt at rett rekke allerede er aktivert er CL (Column Access Strobe Latency), denne er oppgitt i timingen til minnet. Det første tallet i timingen er CL, altså 8. Det betyr at det går 8 klokksykluser før minnet har hentet ut dataen. Tiden blir derfor 8\*1,25ns (aktuelle klokkefrekvensen) = 10ns

* 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:

Hvis man skal beregne tiden en vilkårlig aksess i minnet tar, må man beregne alle forsinkelsene til de ulike operasjonene. Disse er oppgitt i timingen til minnet som følger CL-tRCD-tRP-tRAS (8-8-8-24).

Tiden blir derfor 8\*1,25ns\*3+24\*1,25ns = 60ns

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

Svar:

Kompendiet sier at typisk aksesstid for DRAM er 35-70ns som stemmer godt med aksesstiden utregnet i forrige oppgave.

1. *Sammenligninger:*

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

Svar:

Tiden mellom hver overføring er 0.625ns (DDR klokkefrekvens i burst).

Tiden hver instruksjon tar i prosessoren 0.333ns

Antall instruksjoner = 0.625ns / 0,333ns = 1,9 = ca. 2 instruksjoner

* 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:

Tiden i verste fall er 60ns

Tiden hver instruksjon tar i prosessoren 0.333ns

Antall instruksjoner = 60ns / 0,333ns = ca. 180 instruksjoner