# 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:

Tidligere var det lurt å samle alle instruksjonene som var avhengig av hverandre, slik at mellomresultatet ble liggende i registret. Dette gjorde at man slapp å blande inn minnet i utregningen fordi prosessoren var klar over at mellomresultatet ville ligget i registrert.

* Hvorfor er det ikke lurt på nyere prosessorer?

Svar:

Man ønsker ikke samle instruksjonen på nye prosessorer pga superskalar prosessorer og måten de utnytter pipelines. De ønsker å kjøre flere uavhengige instruksjoner samtidig, men hvis de er avhengige, vil den ene vente på den andre (Data-hazard). Dette skaper forsinkelser, og utnytter ikke den fulle kapasiteten til superskalar pipelines.

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:

Grunnen er fordi instruksjonene som er avhengig av hverandre er samlet noe som gjør at man ikke bruker superskalar pipeline. Mellomresultatene blir lagret til minnet, uavhengig når de blir brukt. Derfor er det ikke nødvendig å utføre operasjonene i rekkefølge.

* 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:

A = 1

Q = 10

B = 1

P = 10

A = A + B

P = P + Q

C = A + 2

R = Q + 10

Kjører to og to instruksjoner som jeg passer på er uavhengig av hverandre. Dette gjør at jeg kan kjøre programmet på halvparten av tiden. Dette ser vi at første pipe, har ansvar for utregning med «A» operasjonen, mens den andre pipen har ansvar for «P»

## 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 1600MT/s 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:

Det tar 0.33 nano sekund for 1 instruksjon

1. *Minne:*

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

Svar:

2/1600\*10^6 = 0,00000000125 sekund = 1,25 nanosekund. Vi må halvere siden minnet har DDR (double digget rate)

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

Svar:

Siden i burst trenger man ikke tenke på aksesseringstiden, derfor er det kun tiden for hver klokkesyklus. Siden det er DDR vil tiden halveres og da tar det 0.625 nanosekund mellom 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:

1.25nanosekund\*8 = 10 nanosekund

* 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:

10ns + 10ns + 10ns + (1.25\*24 = 30 ns) = 60ns.

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

Svar:

I kompendiet er aksesstiden på 35-70, noe som passer veldig fint med aksesstiden over.

1. *Sammenligninger:*

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

Svar:

2 instruksjoner for hver (0.33+1.25 = 1.58ns)

* 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 kan den ta (60/0.33 = 181 instruksjoner) i den perioden.