Obligatorisk øving 4 i datateknikk

Dette er den fjerde obligatoriske øvingen i datateknikk.

Oppgave 1 Parallellitet på instruksjonsnivå

a) 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?
- Hvorfor er det ikke lurt på nyere prosessorer?

Svar:

På eldre prosessorer:

- Det er lettere å holde adekvate data i cpuens registre når flere påfølgende instruksjoner er dataavhengige av hverandre. Eldre cpu-design har svært få generelle registre, og da er dette spesielt viktig.
- Prinsippet om lokalitet gjør at cache-mekanismer har bedre effekt når avhengige instruksjoner ligger i kontinuerlige blokker.

På nye prosessorer:

- Superskalare arkitekturer gjør at flere instruksjoner utføres parallelt, men dette krever at disse instruksjonene er uavhengige av hverandre. For å optimalisere utføringen på superskalare prosessorer vil instruksjonsrekkefølgen ofte stokkes om, slik at pipene i størst mulig utstrekning får jobbe med hver sin instruksjon. Denne omstokkingen kan enten skje 'on-the-fly' (slik som på en Pentium), eller den kan utføres av kompilatoreren på forhånd (slik som på Itanium).

b) 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 e uheldig rekkefølge på en superskalar prosessor?

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

Svar:

Siden flere av instruksjonene er dataavhengige av hverandre vil denne utføringsrekkefølgen være uheldig dersom cpu er superskalar av en slik grad at flere dataavhengige instruksjoner potensielt kan utføres samtidig. Eks: i en cpu som er superskalar av grad 2 må to-og-to instruksjoner være datauavhengige dersom de skal kunne utføres samtidig.

I vårt tilfelle er prosessoren superskalar av grad to. Derfor kan ikke to påfølgende instruksjoner være dataavhengige. Da må vi lete etter instruksjons-par der en instruksjon lager et resultat som den neste instruksjon tar i bruk.

I dette tilfellet er gjelder dette instruksjon 3 og 4 (instruksjon 3 lager et resultat og legger idet i a, og instruksjon 4 tar i bruk a). Disse to instruksjonene kan ikke utføres parallelt. Det samme gjelder instruksjon 6 og 7.

Instruksjonene kan stokkes om på mange måter uten å endre sluttresultatene. En måte å stokke dem om på vil være å la en pipe ta seg av instruksjonene som bruker variablene a, b, c. Og la den andre pipen ta instruksjonene som bruker p, q og r:

`a = 1`	`// utføres i pipe 1`	`Utføres parallelt`
`p = 10`	`// utføres i pipe 2`	

`b = 1`	`// utføres i pipe 1`	`Utføres parallelt`
`q = 10`	`// utføres i pipe 2`	

`a = a + b`	`// utføres i pipe 1`	`Utføres parallelt`
`p = p + q`	`// utføres i pipe 2`	

`c = a + 2`	`// utføres i pipe 1`	`Utføres parallelt`
`r = q + 10`	`// utføres i pipe 2`	

I dette tilfellet kan instruksjonene utføres parvise.

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 minneaksesser.

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.

a) Prosessoren:

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

Svar:

Med en klokkefrekvens på 3GHz er det 0,333 ns mellom hver puls. Dette er det samme som 1/3 ns. Med én instruksjon pr klokkesyklus utføres det altså en instruksjon hver gang det har gått 0,333 ns. Eller for å si det på en annen måte: På ett nanosekund utføres 3 instruksjoner.

b) Minne

Svar:

Her er det viktig å forstå forskjellen mellom *bussfrekvens* og *overføringstakt* i en *burst*.

Med DDR3 1600 er bussfrekvensen 800 MHz (800 millioner sykluser pr sekund), men når vi først har kommet i gang med en *burst*, skjer nye overføringer dobbelt så ofte; det vil si 1600 millioner gang i sekundet.

- Hvor lang tid går det mellom hver klokkesyklus på minnebussen?

Svar:

Vi husker at syklustiden (periode, altså tiden mellom hver ny klokkepuls) er 1/frekvens.

Bussfrekvens er på 800 MHz. Da er syklustiden 1/800 mill, som er 1,25 ns.

- Hvor lang tid går det mellom hver overføring i en *burst* med denne minnetypen?

Svar:

I en *burst* skjer nye overføringer to ganger pr buss-syklus. Da må tiden mellom hver overføring være halvparten av 1,25 ns. Det vil si 0,62 ns.

- Hvor lang tid (i ns) tar det å hente ut nye data når rett rekke allerede er aktivisert?

Svar:

Dette er tiden fra CAS er satt og til den rette cellen i aktiv rekke er tilgjengelig. Denne ventetiden kalles *CAS-latency* (CL), og er det første tallet i timing-angivelsen. I dette tilfellet altså 8 (hele) klokkesykluser. Dette utgjør 8 x 1,25 ns, som er10 ns.

Enkelte fabrikanter 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, mens aksesstid er tiden for å lese en vilkårlig (altså hvilken som helst) lokasjon i minnet. Det skal vi se på i neste spørsmål:

- 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:

Fra leksjonen vet vi at totalt antall sykluser for en minneaksess i dette tilfellet er 8+8+8+24, altså 48 sykluser. Hver syklus er 1,25 ns. Total tid for minneaksessen blir dermed 60 ns (48*1,25).

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

Svar:

60 ns er i overensstemmelse med det som kompendiet angir som typisk aksesstid til DRAM.

c) Sammenligninger

- Hvor mange instruksjoner utføres mellom hver enkelt overføring i en *burst*?

Svar:

Siden hver instruksjon tar (ca) 0,3 ns og tiden mellom hver overføring i en burst er (ca) 0,6 ns blir antall instruksjoner mellom hver overføring (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:

I verste fall tar en minneaksess 60 ns (48*1,25).

Antall instruksjoner på 60 ns: 3*60 = 180 instruksjoner.