NTNU
Kunnskap for en bedre verden

Fakultet for teknologi

Eksamensoppgave i TDAT1003 IT Datateknikk og operativsystem

Faglig kontakt under eksamen: Geir Ove Rosvold (95293039) og Geir Maribu (95807343)

Eksamensdato: 19. desember 2016
Eksamenstid (fra-til): 0900-1200

Hjelpemiddelkode/Tillatte hjelpemidler: Godkjent kalkulator emnegruppe 1. Kandidaten må selv bringe med seg kalkulatoren.

Annen informasjon:

Les gjennom oppgavesettet først. Disponer tiden fornuftig.

Målform/språk: Norsk

Antall sider (uten forside): 2

Antall sider vedlegg: 0

Oppgave 1. Generelt om opsys og om prosesser (vekt: 25%)

a) Vi sier at operativsystemet har to viktige roller. Den ene er å være ressursadministrator. Den andre er å være et abstraksjonslag slik at programvare kan kjøre på maskinen uten å måtte kjenne til maskinvaredetaljer. Forklar hva dette abstraksjonslaget er for noe.

Operativsystemet tildeler programvare en virtuell del av cpu og minne slik at programvaren I større grad vil være platformuavhengig. Programmer vil på den måten også være isolert fra hverandre slik at ikke data kan lekke over I et annet program.

b) I en enkel modell av prosessene kan en prosess være i 3 tilstander. Hvilke tilstander er det?

"Ready" – prosessen er klar og venter på tildeling til cpu for å kunne kjøre.
"Running" – prosessen kjører og kan utføre instruksjoner
"finished" – prosessen er avsluttet

c) Det er 4 mulige overganger mellom disse tilstandene nevnt i oppgaven ovenfor. Hvilke er det og hva er det som fører fra en tilstand til en annen?

Create() lager en ny prosess og prosessen er "ready"
Yield() suspender prosessen og den går tilbake til "ready"
Join() waits for thread to terminate, then goes to "ready"
Exit() suspends the process end exits to "finished" state

d) En prosess kan splitte kjøringen opp i flere tråder. Forklar hvorfor det å operere med flere tråder innenfor en prosess er hensiktsmessig.

Det er hensiktsmessig for å kunne opprettolde responsitivitet på flere områder samtidig. F.eks. kan det være en back-end og en front-end, f.eks. i et klientprogram der man navigerer i en gui, og det samtidig skjer behandling i bakgrunnen. Eller på en server der flere klienter behandles samtidig.

Oppgave 2. Minneadministrasjon (vekt: 25%)

a) Et viktig begrep i minneadministrasjon er adressebinding. I dagens operativsystemer skjer adressebindingen under kjøring. Slik har det ikke alltid vært.

- Hva er adressebinding og til hvilke tidspunkt skjer det?
- Forklar også hvordan adressebinding under kjøring foregår.

- adressebinding
-

b) På en 64-bits CPU har prosesser et gigantisk adresserom. Likevel kan prosessene kjøre på en maskin med relativt lite minne. Forklar.

d) To programmer innenfor et virtuelt minnesystem kan ha samme virtuelle adresse. Diskuter hold- barheten til denne påstanden.

Det er sant. Det er derimot ikke et problem fordi det blir omgjort til fysiske adresser der prosessene har egne adresseområder.

e) Hva menes med prosessens adresseområde? Kan prosessens adresseområde være større enn fysisk tilgjengelig minne? Forklar.

Prosessens adresseområde er et sett med adresser som er tildelt prosessen. Dette området kan ikke være større en fysisk tilgjengelig minne.

Oppgave 3. Tallsystemer (vekt: 15%)

a) Et vanlig tastatur har **104** taster. Når du trykker en tast sendes det et bitmønster til datamaskinen. Dette bitmønsteret identifiserer tasten som ble trykket, og må derfor være unikt for hver enkelt tast. Svar på følgende to spørsmål:

- Hvor mange bits må bitmønsteret minst være for at hver tast skal ha sitt unike bitmønster?
- Hvor mange flere taster kunne tastaturet ha hatt uten at vi trenger å bruke flere bits?

- 7 bits gir 129 adresser (0-128)
- Det betyr at vi kunne hatt (129-104) 25 flere taster uten å bruke flere bits.

b) Skriv følgende bitmønster på *heksadesimal* og *desimal* form:

```
1001011101
```

Vis fremgangsmåten nøye, og forklar hva du gjør.

```
1001011101
0010 0101 1101
2    5    D
```

25D(16)
(16^2)*2 + (16^1)*5 + (16^0)*13
605

Oppgave 4. Prinsippet om lokalitet (vekt: 15%)

Man skulle kanskje tro at når CPU aksesserer primærminnet, så hadde alle minnelokasjoner den samme sannsynligheten for å bli brukt til enhver tid. I praksis viser det seg imidlertid at dersom man vet hvilken lokasjon som ble aksessert forrige gang, så kan man med stor sannsynlighet forutsi (omtrent) hvor neste aksess blir.

a) Hvor ligger sannsynligvis neste aksess?

Enten i nærheten fysisk eller den kan ligge i tidligere brukte aksess. Dette kalles prinsippet om lokalitet, og utnyttes med cache.

b) Denne egenskapen ved programmene har gitt opphav til et viktig prinsipp i datateknikk, nemlig prinsippet om lokalitet. Hva sier prinsippet om lokalitet?

Prinsippet om lokalitet sier at ting er nær hverandre. I tilfellet for minneaksess, er det sannsynlig at dersom en lokasjon har blitt aksessert, vil nærliggende lokasjoner også bli aksessert.

c) Gjelder prinsippet både for instruksjoner og for data, eller gjelder det bare for en av dem? Begrunn svaret nøye, og gi eksempler på at det gjelder / ikke gjelder.

Det gjelder både for instruksjoner og for data. Grunnen er at instruksjoner utføres sekvensielt (dvs. etter hverandre), noe som er enkelt å forutse. Minne med data bruker også dette prinsippet. Grunnen er at data kan bli brukt om igjen f.eks. dersom det kjøres en løkke i et program.

Oppgave 5. Moderne primærminne (vekt: 20%)

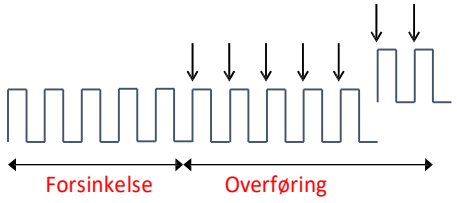
Moderne RAM-typer er synkrone. Det betyr at alle hendelser synkroniseres med hjelp at et klokke-signal (pulser med fast frekvens). Når vi skal lese fra minnet må vi alltid vente på at minnet skal gjøre informasjonen tilgjengelig. Denne forsinkelsen (engelsk: *latency*) oppgis i antall pulser, og fremgår av spesifikasjonene til minnet. I beste fall er forsinkelsen gitt av parameteren CL (CAS-*latency*). Tabellen nedenfor viser et eksempel på at et minne kan brukes med ulike frekvenser, hvor hver frekvens har en tilhørende forsinkelse:

Frekvens:	266 MHz	333 MHz	400 MHz
CL:	4	5	6

a) Du trenger ikke forklare hva CL er, men svar på følgende spørsmål: Vi ser at en og samme minnebrikke kan brukes på ulike frekvenser. Men forsinkelsen (latency) blir større ved høyere frekvenser. Betyr det at minnet blir tregere for høyere frekvenser? Begrunn svaret nøye.

Nei, det betyr bare at det går flere minneklokkeslag over den samme tiden.

b) Ta utgangspunkt i figuren nedenfor og forklar i hvilken av de to fasene (*forsinkelse* og *over-føring*) vi har nytte av høyest mulig frekvens. Begrunn svaret ditt nøye.



I «forsinkelse»-fasen vil det ikke ha noe nytte med høyere frekvens fordi minnet gjør ikke noe annet enn å gjøre klar til neste fase. I «overføring»-fasen, derimot, vil høyere minneklokkeslag gi mer høyfrekvent overføring.

c) I figuren ovenfor viser de loddrette pilene tidspunkt der data overføres på databussen. Synkrone minneteknologier kan være såkalt DDR-teknologi. DDR betyr *Double Data Rate*.

- Forklar hva vi mener med DDR.
- Viser dataoverføringene i figuren et eksempel på DDR? Begrunn svaret.

- DDR en teknologi i minnet som gjør at minnet kan overføre data i begge delene av klokkesyklusen, altså både på høy og lav spenning.
- nei, fordi pilene peker kun på der spenningen er høy, altså en gang per klokkesyklus.