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)
Tlf.: i parentes etter navnet

Eksamensdato: 26. mai 2016
Eksamenstid (fra-til): 0900-1200
Hjelpemiddelkode/Tillatte hjelpemidler: Godkjent kalkulator emnegruppe 1. Kandidaten må selv bringe med seg kalkulatoren.

Annen informasjon:
OBS! Disponer tiden fornuftig. Les gjennom oppgavesettet. Skriv korte og konsise svar. Ikke dvel for lenge ved hver oppgave. Alle deloppgavene teller likt.

Målform/språk: Bokmål
Antall sider (uten forside): 2
Antall sider vedlegg: 0

Oppgave 1. Operativsystemer: Diverse. Vekt: 30%

a) Sidedelte systemer (paging) for minneadministrasjon løste flere problemer. Hvilke problemer var det?
Paging solved the problem of memory fragmentation where a running program would have data in memory that is dynamically changing, leading to pieces of data not being able to fit together contiguously because of other programs also running in memory. With paging, the problem is solved by having the data organized virtually in pages and memory can be accessed quicker as a result.
Another problem it solved is that parts (pages) of memory can now be stored on disk to save it for later.

b) Påstand: To programmer innenfor et virtuelt minnesystem kan ha samme virtuelle (logiske) adresse. Diskuter holdbarheten til denne påstanden.
Ja, fordi den fysiske adressen er ikke den samme.
Under transformering blir det lagt på en «offset»
for å hindre overlapping av minne-adresser.

c) Hva brukes et baseregister til i forbindelse med minneadministrasjon? En sidetabell kan sees på som en tabell med mange base-registre. Forklar.
Baseregister er en oversikt over «base» og «bound» for hver «adress space» til hvert program.

d) Skriv et kort C-program (pseudo-kode) for Linux som oppretter 2 barneprosesser. Den første barneprosessen skal skifte ut koden sin med koden til **ls**-kommandoen, den andre prosessen skal skifte ut koden med koden til **ps**-kommandoen.

```
pid child1, child2
if (child1 = fork() == 0) {
   child1.exec("ls")
}
if (child2 = fork() == 0) {
   child2.exec("ps")
}
else {
   return -1
}
wait()
return 0
```

e) I operativsystemer opererer en med *kernel modus* og *user modus*. Forklar hva dette er og hva hen- sikten med slike modus er.
Kernel mode is used to execute very low level trusted code as it has direct access to the computer's hardware. However, a crash in kernel mode will hault the entire computer. Most of the code is run in user mode. This is ideal for running less trusted software and allows for relatively safe code execution.

f) En datamaskin bruker 22-bits virtuelle adresser og 16-bits fysiske adresser. Sidestørrelsen er 8K. Hvor mange innslag trengs i sidetabellen?

Oppgave 2. Operativsystemer: Prosesser og kommunikasjon mellom prosesser. Vekt: 30%

a) Gi en definisjon av følgende begreper:

i. Program

En sekvens med instruksjoner som ligger på disk klar til å sekvenseres.

ii. Prosess

Et program som kjører i minnet.

iii. Tråd

Rekkefølgen prosessoren utfører programmet.

b) Forklar hvordan semaforer virker.

A semaphore is a type of lock variable that allows or disallows a process to access a resource based on its variable. It is either binary or an integer and is used in multiprocessor environments where data can be shared and potentially corrupted upon simultaneous access.

c) Anta at prosessene P0 og P1 deler en felles variabel V2, prosessene P1 og P2 deler variabelen V0 og prosessene P2 og P3 deler variabelen V1

Vis hvordan disse prosessene kan bruke *enableInterrupt* og *disableInterrupt* (dvs slå av og på avbruddssystemet) for å styre adgangen til V0, V1 og V2 slik at *kritisk region*-problemer ikke oppstår.

d) Vis også hvordan prosessene kan bruke semaforer for å styre adgangen til V0, V1 og V2 slik at *kritisk region*-problemer ikke oppstår.

f) To prosesser P1 og P2 er laget slik at P2 skriver ut data som er produsert av P1. Lag et enkelt pro- gram (skjelett, pseudo-kode) for P1 og P2 for å illustrere hvordan de synkroniserer hverandre.

f) I utgangspunktet ønsker vi ikke å bruke *enableInterrupt* og *disableInterrupt* for å sikre gjensidig utelukkelse. Hvorfor? Men i semaforer bruker vi det likevel. Forklar hvorfor det er akseptabelt å bruke *enableInterrupt* og *disableInterrupt* i semaforer.

Oppgave 3. Datateknikk: Moderne prosessorarkitektur Vekt: 40%

a) En instruksjon sørger for at følgende 5 hendelser (i., ii.,…,v.) skjer:

i. Dekod instruksjonen.
ii. Hent operand fra minnet.
iii. Hent instruksjonen fra minnet og legg den i instruksjonsregisteret.
iv. Lagre resultatet i minnet.
v. Utfør instruksjonen.

Ordne disse (i. - v.) i riktig rekkefølge.

For hvert av trinnene i..v:

iii. Hent instruksjonen fra minnet og

legg den i instruksjonsregisteret.

i. Dekod instruksjonen.

ii. Hent operand fra minnet.

v. Utfør instruksjonen.

iv. Lagre resultatet i minnet.

b)

Angi om trinnet trengs for alle instruksjoner, eller om trinnet bare trengs for enkelte instruksjoner. Vær nøye med å begrunne dine påstander. Gi gjerne eksempler på instruksjoner som kan illustrere dine påstander.

«hent instruksjonen fra minnet og legg den i instruksjonsregisteret» trengs for alle instruksjoner fordi vi må vite hva som skal gjøres.
«dekod instruksjonen» trengs også for å vite hva som skal gjøres.
«Hent operand fra minnet» trengs ikke hvis operanden er gitt i selve instruksjonen.
«utfør instruksjonen» trengs for alle instruksjoner.
«lagre resultatet i minnet» trengs ikke f.eks. hvis noe slettes.

c) En prosessor bruker en 5-trinns pipeline med trinnene som er angitt i deloppgave a). Det skal utføres 1000 instruksjoner på denne prosessoren. Anta at hvert trinn utføres i løpet av en klokkesyklus. Hvor mange klokkesykluser tar dette under ideelle forhold? (Ideelle forhold vil si at vi får maksimal uttelling ved bruk av pipeline). Vis tydelig hvordan du finner svaret.

It will take 1004 cycles because it will start the first instruction on the first clock cycle, so the 1000th cycle will begin the 1000th instruction. The 1001st cycle executes the 2nd step on the last

instruction. The 1002nd cycle executes the 3rd step on the last instruction, same for the 4th ant the 5th step of the instruction which counts up to 1004.

d) Vanligvis vil en prosessor bruke flere klokkesykluser enn det antall du beregnet i oppgave i forrige oppgave Dette skyldes at det oppstår såkalte hasarder. Beskriv følgende hasarder og forklar hvordan de løses:

i. Strukturell hasard

ii. Datahasard

iii. Kontrollhasard

e) En av de følgende har ansvaret for å løse hasarder på en slik måte at programutføringen blir korrekt:

i. Brukeren

ii. Programmereren

iii. Kompilatoren

iv. Prosessoren

Hvem av disse (i. – iv.) har ansvaret? Vær nøye med å begrunne svaret ditt. Kommenter gjerne på om de andre også kan spille en rolle.
Kompilatoren har hovedsakelig ansvaret

g) En *splittet cache* er en cache som består av to del-cacher. Den ene del-cachen brukes **bare** til instruksjoner, og den andre brukes **bare** til data. Er pipeline en årsak til at det kan være fornuftig å bruke en slik *splittet cache*? Begrunn svaret nøye.

Pipeline gjør at det kan være nyttig dersom det trengs å aksessere både instruksjoner og data samtidig. I en pipeline vil instruksjonene bli delt opp i sub-instruksjoner som kan trenge kun instruksjoner eller kun data. Dette gjør at aksess kan gjøres i begge del-cachene uten at problemer oppstår. Resultatet er raskere utførelse pga. parallell aksess fremfor seriell i noen tilfeller.