Eksamensprosjekt vår 2019

Torstein H Sundfær

**Oppgave 1. Operativsystemer: Diverse.**

1. **Sidedelte systemer (paging) for minneadministrasjon løste flere problemer. Hvilke problemer var det?**

Kjøring av programmer som er større enn tilgjengelig minne og bedre innhenting av informasjon som trengs for kjøring av programmer.

1. **Påstand: To programmer innenfor et virtuelt minnesystem kan ha samme virtuelle (logiske) adresse. Diskuter holdbarheten til denne påstanden.**

Dette er sant da alle programmer har en virtuell adresse som starter på 0 og går så langt som den trenger eller til så langt som det er plass i det virtuelle minnet.

1. **Hva brukes et baseregister til i forbindelse med minneadministrasjon? En sidetabell kan sees på som en tabell med mange base-registre. Forklar.**

Baseregisteret angir adressen til programmer i minnet, en sidetabell brukes i forbindelse med sidedelt minne administrasjon, den henter inn adressen til de delene det er bruk for og viser hvor de aktuelle delene ligger i fysisk minne, likt at baseregisteret viser hvor hele programmet ligger.

1. **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.**

Main()

Firstchild = fork();

If (firstchild == 0)

Exec(«ls»);

Secondchild = fork();

If (secondchild == 0)

Exec(«ps»);

1. **I operativsystemer opererer en med *kernel modus* og *user modus*. Forklar hva dette er og hva hensikten med slike modus er.**

User modus brukes til vanlig kjøring av programmer som ikke trenger tilgang til operativsystemets fulle ressurser, mens kernel mode brukes gjerne når noe trenger f.eks. tilgang til å lese/skrive til harddisken eller få input fra tastatur. Det byttes mellom disse to med systemkall fra prosessen som kjører. Kernel modus brukes også til administrasjon av datamaskinens ressurser.

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

22 bits virtuelle adresser blir 22^2 = 4Mbyte

4Mbyte / 8k = 512, så det er 512 innslag i sidetabellen.

**Oppgave 2. Operativsystemer: Prosesser og kommunikasjon mellom prosesser.**

1. **Gi en definisjon av følgende begreper:**
   1. **Program**
   2. **Prosess**
   3. **Tråd**

Et program er et sett instruksjoner som utfører en gitt oppgave.

En prosess er et program som kjører

En Tråd er minste mulige programinstruksjon som kan kjøres av cpu

1. **Forklar hvordan semaforer virker.**

En semafor er en variabel og en kø, som har to operasjoner, wait og signal som kjører på semaforen.

Hvis noe trenger tilgang så kaller det først wait, hvis den får svar 1 så er den ledig, hvis den får svar 0 så er den opptatt og den må vente.

1. **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.**

P0 gjør noe, disableInterrupt, bruk V2, enableInterrupt, gjør noe annet.

P1 gjør noe, disableInterrupt, bruk V2, enableInterrupt, gjør noe annet, disableInterrupt, bruk V0, enable Interrupt.

P2 gjør noe, disableInterrupt, bruk v0, enableInterrupt, gjør noe annet, diasbleInterrupt, bruk V1, enableInterrupt.

P3 gjør noe, disableInterrupt, bruk v1, enable interrupt, gjør noe annet.

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

P0 gjør noe, wait(sem), bruk V2, signal(sem), gjør noe annet.

P1 gjør noe, wait(sem), bruk V2, signal(sem), gjør noe annet, wait(sem), bruk V0, signal(sem).

P2 gjør noe, wait(sem), bruk v0, signal(sem), gjør noe annet, wait(sem), bruk V1, signal(sem).

P3 gjør noe, wait(sem), bruk v1, signal(sem), gjør noe annet.

1. **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.**

P1 produser data, signal(sem), repeat.

P2 wait(sem), bruk data, repeat.

1. **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.**

De kan brukes i semaforer fordi der er instruksjonene små slik at det er lav sjanse for at det skjer noe galt når disse brukes.

**Oppgave 3. Datateknikk: Moderne prosessorarkitektur**

1. **En instruksjon sørger for at følgende 5 hendelser ( i., ii.,…,v. ) skjer:**
   1. **Dekod instruksjonen.**
   2. **Hent operand fra minnet.**
   3. **Hent instruksjonen fra minnet og legg den i instruksjonsregisteret.**
   4. **Lagre resultatet i minnet.**
   5. **Utfør instruksjonen.**

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

3, 1, 2, 5, 4.

1. **For hvert av trinnene i..v:**

**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.**

3, alle instruksjoner må alltid hentes, må derfor alltid brukes.

1, alle instruksjoner må dekodes, må derfor alltid brukes.

2, instruksjoner som ikke trenger inndata må ikke bruke denne, må derfor ikke alltid brukes.

5, alle instruksjoner må utføres.

4, ikke alle instruksjoner har et resultat som må lagres, må derfor ikke alltid brukes.

1. **En prosessor bruker en 5-trinns pipeline med trinnene som er angitt i deloppgave a). Det skal ut-føres 1000 instruksjoner på denne prosessoren. Anta at hvert trinn utføres i løpet av en klokkesyk-lus. 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.**

Dette tar 5 + 999 = 1004 klokkesykluser ut ifra formelen k+(n-1) der k er antall trinn i pipeline og n er instruksjoner.

1. **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:**
   1. **Strukturell hasard**

Strukturell hasard betyr at to operasjoner trenger samme ressurs samtidig, den siste instruksjonen må da vente, denne ventingen forplanter seg videre slik at det blir en forsinkelse.

* 1. **Datahasard**

Datahasard er når en instruksjon lagrer data som en senere instruksjon trenger, hvis resultatet ikke er ferdig enda må den siste instruksjonen vente. Dette skaper også en forsinkelse som forplanter seg videre.

* 1. **Kontrollhasard**

Kontrollhasard kommer fra hopp i programmet, dette gjør at de andre instruksjonene som starter å utføres før den vet at den skal hoppe ikke lenger er nødvendige, denne tiden er da bortkastet.

1. **En av de følgende har ansvaret for å løse hasarder på en slik måte at programutføringen blir kor-rekt:**
   1. **Brukeren**
   2. **Programmereren**
   3. **Kompilatoren**
   4. **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.**

Prosessoren er den som har ansvaret for å løse hasarder og at ting utføres korrekt. En god kompilator kan selvfølgelig lage koden slik at det blir lettere for prosessoren, men i bunn er det opp til prosessoren at alt går som det skal.

1. **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.**

Dette er for at de to delene av cachen har hver sin buss, som gjør at vi kan lese fra instruksjons-cachen og skrive til data-cachen samtidig.

Pipeline har dette behovet slik at flere instruksjoner kan kjøres samtidig med redusert sjanse for hasarder å oppstå.