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?

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

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

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.

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

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

   ii. Prosess

   iii. Tråd

b) Forklar hvordan semaforer virker.

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.

e) To prosesser P1 og P2 er laget slik at P2 skriver ut data som er produsert av P1. Lag et enkelt program (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.

b) 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.

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.

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.

f) 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.