INF1060: avsluttende eksamen

Kandidatnummer: _____

❑ **Oppgave 1 - operativsystemer** — *max 35 poeng*

- ❑ 1a **(6 poeng)** – formålet er å (1) skjule komplekse detaljer om underligende HW, være enkel å bruke, “inntrykk av en maskin for hver prosess”, ..., portabilitet av programmer –> være en “virtuell maskin”, OG
  (2) tillby forskjellige muligheter for å bruke HW på en fornuftig, effektiv, rettferdig og sikker måte –> “ressursmanager”.
  Fordelene er mer eller mindre det samme som punktene i formålet, men ulemper kan være ting som at OSet selv tar ressurser, ofte implementert for generelle applikasjoner og min applikasjon kunne kanskje hatt bedre ytelse med andre mekanismer, ...
- ❑ 1b **(6 poeng)** – memory is a volatile, limited resource: OS usually on disk. Most motherboards contain a basis input/output system (BIOS) chip (often flash RAM) stores instructions for basic HW initialization and management, and initiates the bootstrap: (1)loads the OS into memory, (2) reads the boot program from a known location on secondary storage first sector(s), often called master boot record (MBR), (3) run boot program which reads root file system and locate file with OS kernel, loads kernel into memory and finally starts the kernel (a figure can be found at http://www.ifi.uio.no/ inf1060/Forelesninger05/OS-intro.pdf - slide 30) (ops,)
- ❑ 1c **(6 poeng)** – et interrupt er et elektronisk signal som resulterer i "forced" overføring av kontroll til en interrupthandler slik at vi kan få avbrutt den kjørende prosessen og fortalt om at noe har skjedd, f.eks. en pakke har kommet fra nettveret, en I/O forespørsel er ferdig, en prosess har brukt opp tiden den har på CPU’en
  en “exception” er en lignende mekanisme for at prosessoren skal kunne avbryte et programm ved at synkrine hendelser genereres når forhåndsdefinerte betingelser oppdages under eksekvering av en instruksjon. Deler ofte opp i tre typer: (1) TRAPS hvor prosessoren ser en betingelse som kan håndteres (owerflow, system kall, ...), (2) FAULTS når man oppdager feil som kan rettes opp (dele med null, feil data format, ...) og (3) ABORTS som terminerer programmet pga en uopprettelig feil (som HW feil, ...)
  Både interrupts og exceptions håndteres likt av en interrupt handler (IA-32 har 256 mulige typer) og vi har kode assosiert med hver type. Overordnet har vi følgende steg: 1) avbryt kjørende prosess - lagre context, 2) se hva slags interrupt, slå opp i IDT for å finne riktig code segment, 3) kjør interrupt koden, 4) gjennoprett kjørende prosess (evt. annen), 5)

fortsett kjøringen (a figure can be found at http://www.ifi.uio.no/ inf1060/Forelesninger05/OS-intro.pdf - slide 35)

- 1d **(6 poeng)** – man definerer ofte tre basis tilstander: kjørende, blokkert og klar. I tilleg kan man ha anre tilstander som opprettet, terminert, etc.

```
terminated <--  running  <--->  ready <--- created
                   |              ^
                   |              |
                   +-> blocked ---+

  - creation --> ready  (a new prosess is created)
  - running --> termination  (prosess is finished)
  - running --> ready (used its timeslice,priority,interrupts)
  - running --> blocked (performes a I/O request)
  - blocked --> ready   (I/O request is finished)
  - ready   --> running (scheduler picks this process)
```

- 1e **(6 poeng)** – virtuelt minne er en mekanisme for å gi inntrykk av at man her mer minne tilgjengelig en det som er fysisk satt inn i maskinen. Skulle hjelpe på problemene med å partisjonere programmer, etc. fordi man tidligere hadde lite minne Observasjonen var da at man ikke trengte alle data hele tiden.
  Programmene blir her delt i mindre deler (like store rammer kaldt sider/pages) hvor da noen er fysisk i minnet mens andre er lagret på disk og hentes ved behov.
  En virtuell addresse oversettes til en fysisk addresse ved hjelp av en side/pagetabel i MMUen: Overste bittene i virtualle addressen brukes som offsett/indeks til pagetabellen. I denne raden i tabellen finnes informasjon om den fysiske siden. Man har minst en addresse og et "present bit": hvis bittet er 0 er siden på disk og addressen sier ofte noe om disk blok addressen, men er bittet 1 er siden minnet og addressen gir de addressen til siden i minnet. For så å finne byten vi leter etter brukes de nederste bittene i den virtuelle addressen (typisk de 12 nederste for 4KB sider) som offsett i siden
  (se http://www.ifi.uio.no/ inf1060/Forelesninger05/OS-memory.pdf - slides 18 - 22) for generell ide og (se http://www.ifi.uio.no/ inf1060/Forelesninger05/OS-memory.pdf - slides 32 - 40) for pagin på Intel

- 1f **(5 poeng)** – Akkess tiden er tiden fra vi ber om å få lest en disk blokk til den er på plass i minnet. Faktorer som inngår er typisk *søking, rotasjonsforsinkelse og overføringstid* - disse må man ha nevnt for det er disse som ofte dominerer den totale tiden. I tillegg kan man nevne tid for å prosessere forespørselen, venting i forbindelse med kontrolleren, bussen og minnet, verifisering av lese/skrive operasjonen, venting i scheduleringskø, ...
  (se http://www.ifi.uio.no/ inf1060/Forelesninger05/OS-storage.pdf - slides 8 - 13)
  Optimeringer kan være scheduling algoritmer, data plassering strategier, parallelle disker, batching av forespørsler, preloading/prefetching som read-ahead, minne/buffer caching, blokkstørrelser, ...

- **Oppgave 2 - kommunikasjon** — *max 35 poeng*
  Løsningsforslagene er mer eller mindre hentet fra foilene. Merk at andre forklaringer kan også være fornuftige.

- 2a **(3 poeng)** – Internet er et nettverk av nettverk, dvs. mange proprietære, regionale og offentlige nett knyttes sammen via rutere i "kjernenettverket". Internet er delvis hierarkisk.

- 2b **(3 poeng)** – Protokoller definerer formater, rekkefølgen for sending og mottak av meldinger, samt hvilke aksjoner som mottak av meldinger initierer.

- 2c **(5 poeng)** – De omtalte referansemodellene er lagdelte (hierarkiske). Hovedgrunnen til at de har blitt spesifisert er at kommunikasjonssystemer er meget komplekse, og det viste seg å være nødvendig å modularisere. Dette forenkler bl.a. design, vedlikehold og oppdatering av et system. Lagdelingen gjør det mulig å samle og identifisere ulike funksjonelle egenskaper i ulike lag og å spesifisere relasjoner mellom dem. Intensjonen er også at ulike implementasjoner av et lag skal kunne fungere sammen med alle implementasjoner av andre lag (dvs. det er funksjonelle modeller). > En god besvarelse bør inneholde en beskrivelse av hovedfunksjonene til hvert lag. Hovedforskjellen mellom TCP/IP og OSI modellen er at førstnevnte mangler Sesjons- og Presentasjonslag. Dette innebærer at Applikasjonene i Internet selv må ivareta de funksjonene som disse lagene utfører (må være innebakt i applikasjonsprotokollene).

- 2d **(4 poeng)** – Her bør tvunnet parkabel, koaksialkabel, fiberoptisk kabel og radiolinker tas med. Følgende kan sies: - parkabel: Bra: billig; finnes allerede lagt ut mange steder; kan håndtere forholdsvis høye hastigheter; lett å legge ut. Ulempe: dårlig skjerming; mulig el.magnetisk interferense ==> kan oppleve høy feilrate; lett å avlytte. - Koaks: Bra: God skjerming mot el.magnetisk støy; kan handtere høye hatigheter; lett tilkopling av nye stasjoner (on the fly - uten driftsavbrudd). Ulempe: stiv - kan være vanskelig å legge. - Fiber: Meget høye hastigheter; lang rekkevidde; ikke påvirket av el.magnetisk interferens. Ulempe: koplisert å "skjøte"; dyrt tilkoplingsutstyr til maskiner; brukes primært mellom rutere og servere. - Radio: (Mange ulike typer) Bra: fleksibelt - kan f.eks. brukes i ulendt terreng; satelitt-kommunikasjon åpner for langdistanse overføring; forholdvis høye hastigheter; Ulempe: "Åpent" nettverk - alle kan lytte på linkene. Stor feilsannsynlighet pga. refleksjoner, blokkering (bygninger) og vær.

- 2e **(3 poeng)** – Linjesvitsjing: Tradisjonelt: Svitsjene sørger for å etablere en "galvanisk" forbindelse mellom endesystemene. I dag gjøres dette elektronisk slik at ressurser kan deles, men brukeren vil bli garantert tilgjengelige ressurser. Fordel: Etter at en forbindelse er etablert har brukeren en garantert båndbredde så lenge forbindelsen er oppe. Ulempe: Siden brukeren er garantert en gitt overføringskapasitet, vil ressurser kunne stå ubrukt når det ikke sendes noe. Pakkesvitsjing: Overføringen deles opp i pakker som rutes gjennom nettet på en mest mulig optimal måte. Hver pakke benytter den totale kapasiteten på linken. Det er konkuranse om bruk av nettressursene. Dette innebærer at pakkene kan ta ulike veier gjennom nettet alt etter hvordan de enkelte linkene/nodene er belastet. Denne formen for ressursdeling er den viktigste fordelen (men det kan også være en ulempe).Den viktigste ulempen er at det ikke gis garantier for at pakker kommer fram og det kan ikke gis en garantert overføringshastighet (throughput).

- 2f **(3 poeng)** – Forbindelsesløs overføring: Kalles også datagramtjeneste. Hver pakke bærer nødvendig adresseinfo til å finne fram til mottaker samt sin egen adresse. Rutes individuelt gjennom nettet og ruterne forsøker å finne en mest mulig optimal vei. Dette er hovedfordelen. Ulempen er at det ikke gis noen garantier ("best effort") for at pakker kommer fram, forsinkelser og tap av pakker kan oppstå. Forbindelsesorientert overføring: Følgende skjer: Oppkopling, dataoverføring, nedkopling. Opp- og nedkoplingen skal sikre at mottatte data ikke misstolkes (f.eks. at stasjonen mener at de hører til en annen forbindelse). I datafasen vil dataene komme

i samme sekvens som de ble sendt og feilfritt. Mottaker kan signalisere at bufferne er fulle (flytkontroll).

- ❏ 2g **(4 poeng)** – Hovedteknikkene er: Tidsmultipleksing, frekvensmultipleksing og pakkemultipleksing. Kort beskrivelse: - Tidsmultipleksing: hver kanal får tilgang til mediet i en viss tid (tidsluke). Hver kanal avtastes i sekvens om igjen og om igjen. Ledig kapasitet i en kanal kan ikke utnyttes av en annen kanal. - Frekvensmultipleksing: mediet deles inn i frekvensbånd og hver kanal tilordnes hvert sitt bånd. Ledig kapasitet i en kanal kan ikke utnyttes av en annen kanal. - Pakkemultipleksing: Pakker plukkes fra innlinjene og legges i en FIFO utkø i ruteren. Pakkene i denne køen legges etter hverandre ut på utlinja. I mottager-ruteren foretas demultipleksing basert på adresser i pakkene.

- ❏ 2h **(4 poeng)** – Adresse-notasjonen er enten en tekstlig eller en digital (dot-)notasjon. Eks. niu.ifi.uio.no og 129.240.002.35. I begge tilfelle er adresse-strukturen hierarkisk for å forenkle framsending av pakker på Internet (uten struktur måtte man hatt gigant-rutere som inneholdt alle tildelte adresser). Hierarkiet er vanligvis som følger: vert.subnett.domene.land for begge representasjonene, men antall bit som brukes til hva kan variere. "land"-identifikasjonen kan også være "organisasjon" og andre ting, men dette er ikke nevnt på forelesningene.

- ❏ 2i **(3 poeng)** – Hovedkomponentene i en ruter er inn og utkøer, rutetabell samt preprosseserings-, ruting- og framsendings-prosesser. Preprosseserings-prosessen plukker pakker fra inn-køen og leverer adresse-info (samt linjenummer) til ruting-prosessen. Denne oppdaterer rutetabellen vha. den mottatte informasjonen, og overlater pakka til framsendings-prosessen som legger pakka i riktig utkø.

- ❏ 2j **(3 poeng)** – Kommunikasjonen over nettet mellom distribuerte applikasjons-prosesser må foregå i en (standardisert) syntaks som begge sider oppfatter på samme måte. En slik syntaks kaller vi "overføringssyntaks". Denne er nødvendig fordi kommunikasjonen foregår mellom inhomogene ende-systemer (eks. ulik maskinvare, ulike OS, ulike programmeringsspråk).

- ❏ **Oppgave 3 - flervalgsoppgaver** — *max 30 poeng*
  Hvert riktige gir 2 poeng. Har de listet opp noen andre gir hvert av disse -1 poeng. Totalt sett skal man ikke kunne få minus poeng på oppgave 3. Dvs. at man i så fall gir 0 poeng.

  - ❏ 3a – **C programmering (16 poeng)**: 3, 6, 11, 12, 15, 17, 21 og 23 er riktige. **NB! 6 er ment å være riktig, men det står feil linjenummer i oppgaven - 13 skal være 14**. Output'en fra programmet er:

```
linje 9/10: 100 1001 1001
linje 18:   500  400    1
linje 20:   100 1001    1
```

  - ❏ 3b – **OS (6 poeng)**: 2, 7 og 8 er riktige
  - ❏ 3c – **Datacom (8 poeng)**: 2, 6, 7 og 8

➥ **Poeng totalt: _____** (*max 100 poeng*)