Løsningsforslag Eksamen 2020 Operativsystemer

For oppgave 1, 2, 3, 15, 16, 17 og 18 fikk man tilfeldig trukket bare ett av to eller tre mulige spørsmål. Svar på alle alternativene er vist her.

**Datamaskinarkitektur**

## 1)

**Datamaskinarkitektur**

Hvilken type logisk port er dette sannhetstabell for? A og B er input og UT er resultatet.

| A | B | UT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Velg ett alternativ**

- NOT
- XOR
- OR ✓
- PMOS
- NMOS
- AND
- NOR

Riktig. 10 av 10 poeng. Prøv igjen

**Datamaskinarkitektur**

Hvilken type logisk port er dette sannhetstabell for? A og B er input og UT er resultatet.

| A | B | UT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Velg ett alternativ**

- OR
- XOR
- PMOS
- NOT
- AND ✓
- NMOS
- NOR

Riktig. 10 av 10 poeng. Prøv igjen

## Datamaskinarkitektur

Hvilken type logisk port er dette sannhetstabell for? A er input og UT er resultatet.

| A | UT |
|---|---|
| 0 | 1 |
| 1 | 0 |

**Velg ett alternativ**

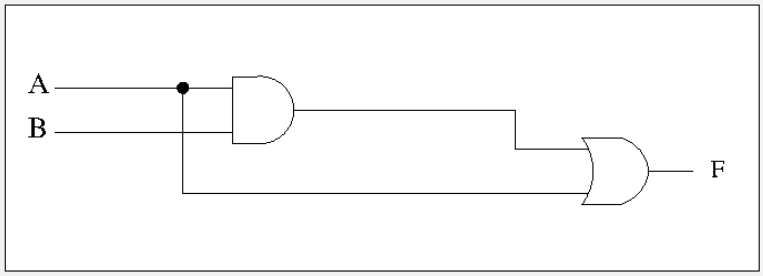
- PMOS
- NMOS
- NOT ✓
- NOR
- XOR
- OR
- AND

Riktig. 10 av 10 poeng. Prøv igjen

2)

## Datamaskinarkitektur

Hva blir det logiske uttrykket F(A,B) for denne kretsen?



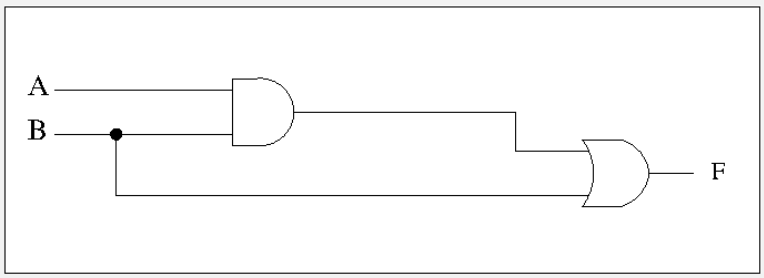

**Velg ett alternativ**

- $\overline{A+B}$
- $\overline{A} \cdot B + \overline{A}$
- $(A+B) \cdot A$
- $A \cdot B \cdot A$
- $A \cdot B + A$ ✓
- $A \cdot B + B$
- $(A+B) \cdot B$

Riktig. 10 av 10 poeng. Prøv igjen

## Datamaskinarkitektur

Hva blir det logiske uttrykket F(A,B) for denne kretsen?




**Velg ett alternativ**

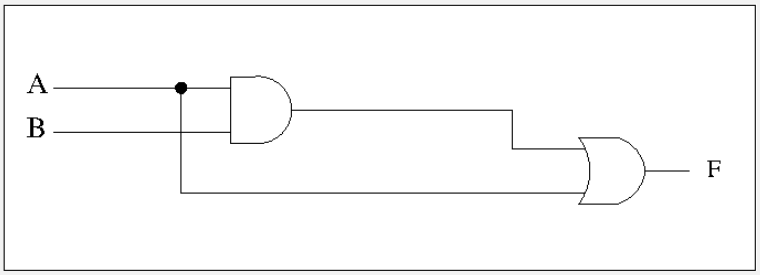
- $(A + B) \cdot B$
- $A \cdot B + B$ ✓
- $\overline{B} \cdot A + \overline{B}$
- $A \cdot B \cdot B$
- $(A + B) \cdot A$
- $\overline{A + B}$
- $A \cdot B + A$

Riktig. 10 av 10 poeng. Prøv igjen

**3)**

## Datamaskinarkitektur

Uttrykket for den kretsen kan forenkles, utifra sannhetstabellen eller ved å bruke boolsk algebra. Hva kan uttrykket for F(A,B) forenkles til?



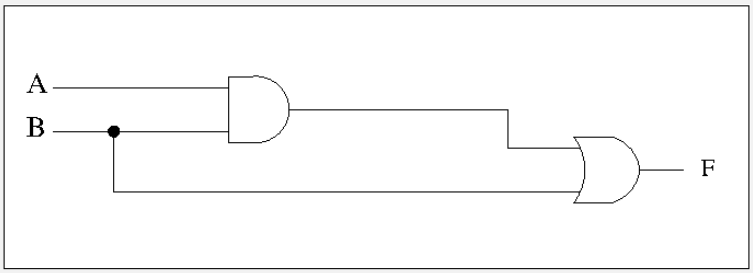

**Velg ett alternativ**

- $B$
- $1$
- $A + B$
- $0$
- $\overline{B}$
- $A \cdot B$
- $\overline{A}$
- $A$ ✓

Riktig. 10 av 10 poeng. Prøv igjen

## Datamaskinarkitektur

Uttrykket for den kretsen kan forenkles, utifra sannhetstabellen eller ved å bruke boolsk algebra. Hva kan uttrykket for F(A,B) forenkles til?




**Velg ett alternativ**

- $B$ ✓
- $A$
- $\overline{B}$
- $0$
- $A \cdot B$
- $1$
- $\overline{A}$
- $A + B$

Riktig. 10 av 10 poeng. Prøv igjen

**Threads**

**4)** En felles variabel y og en variabel x i hver thread

**Prosesser**

**5)** Hyperthreading

**6)** Den første kommandoen kompilerer C-programmet med gcc, opsjonen -O optimaliserer med tanke på at programmet skal utføres raskest mulig og dette resulterer i en kjørbar fil med navn a.out. I den andre kommandoen kjøres programmet med time og det resulterer i følgende:

- Real: Den relle tiden det tar å kjøre programmet
- User: Tid brukt av programmet i user-mode
- System: Tid brukt av programmet i kernel-mode
- Prosent: Hvor stor andel av CPU'en programmet bruker under kjøringen

**7)** Den første for-løkken kjører og tar tiden på a.out fire ganger, men hver av kjøringene fullføres før den neste starter. Den andre for-løkken gjør det samme, men i dette tilfellet startes a.out som en bakgrunnsprosess slik at fire instanser av programmet kjøres i parallell på systemet(får hver sin CPU, som kan sees av 100% CPU-bruk) . Dermed vil for den siste kjøringen alle de fire prosessene være ferdige etter ca 1.8 sekunder mens totaltiden for den første for-løkken er fire ganger så lang, ca 7.2 sekunder.

**8)**

1. I den første kjøringen ser vi at hver prosess får 100% av en CPU og at tiden er 1.8 sekunder.
2. Her ser vi at prosessene får ca 67% CPU-tid og det tyder på at de 6 prosessen blir fordelt på 4 CPUer. Totalt er det behov for 6x1.8 CPU-sekunder. Deles dette på 4 CPUer blir svaret $6x1.8/4 = 2.7$ sekunder som man ser er omtrentlig tiden de 6 prosessene bruker hver.
3. I dette tilfellet får hver prosess i snitt 50% CPU-tid og hver prosess bruker da dobbelt så lang tid, det vil si $2x1.8 = 3.6$ sekunder.

Det er tydelig fra den siste kjøringen at det er 8 prosesser som blir fordelt på 4 CPUer siden de får 50% CPU-tid. Det samme kan konkluderes fra andre kjøring, det må være 4 uavhngige CPUer(cores).

**9)** Her kjører programmet på en annen type CPU som tydligvis er raskere. Vi ser at i den siste kjøringen med 8 CPUer så er den reelle tiden for hver prosess ca 1.2 sekunder, omtrent dobbelt så lang tid som i den første kjøringe. Men time rapporterer også at de 8 prosessene hver får 100% CPU-tid. Dette må være et resultat av at systemet har 4 hyperthreading CPUer som operativsystemet behandler som om det var 8 uavhengige regneenheter.

1. I den første kjøringen ser vi at hver prosess får 100% av en CPU og at tiden er 0.6 sekunder.
2. Her ser vi at hver prosess får 100% CPU-tid, men det er fordi OS 'ser' 8 uavhengige CPUer. Først kjører jobbene på 6 av de 8 CPUene som OS "ser" og de to som da kjører på helt egen kjerne blir ferdig etter 0.6 sekunder. De 4 andre som deler kjerne med hyperthreading har da bare kommet halveis og kjører så ferdig de 0.3 sek de har igjen på hver sin selvstendige kjerne.
3. I dette tilfellet vil det være to prosesser på hver hyperthreading kjerne. Dermed må de to dele på ALUen når de regner og dette gjør at det tar dobbelt så lang tid, 1.2 sekunder, som når det var en prosess per kjerne som i den første kjøringen.

## 10)

1. I den første kjøringen ser vi at hver prosess får 100% av en CPU og at tiden er ca 2.3 sekunder.
2. Prosessene får fortsatt hver sin kjerne, 100% CPU og hver jobb tar 2.3 sekunder.
3. Prosessene får fortsatt hver sin kjerne, 100% CPU og hver jobb tar 2.3 sekunder.

Det betyr at det på denne serveren er 8 (eller flere) uavhengige regne-enheter (ALUer/CPUer/kjerner). (I dette konkrete tilfellet hadde serveren 48 kjerner, men det kan man ikke 'se' før man tester med flere enn 48 prosesser). Man kan ikke avgjøre om disse 8 (eller flere) kjernene er hyperthreading eller ikke, siden det ikke scheduleres mer enn en prosess per kjerne.

### Docker

## 11)

```
FROM ubuntu
RUN apt-get -y update
RUN apt-get -y install apache2
COPY index.html /var/www/html
CMD ["/usr/sbin/apachectl","-D","FOREGROUND"]
```

## 12)

```
docker build -t ubuntuA .
```

eller

```
docker image build -t ubuntuA .
```

Det siste argumentet er PATH til der hvor Dockerfile ligger. Så hvis man bruker '.' i kommandoen må man stå i samme mappe som Dockerfile ligger. Men man kan også stå i en annen mappe om man angir PATH til der hvor Dockerfile ligger.

## 13)

```
docker container run -p 5555:80 -d ubuntuA
```

Filen blir inkludert i image'et når det bygges. Om man endrer filen i ettertid, vil den ikke endres i containere som er startet med dette image'et.

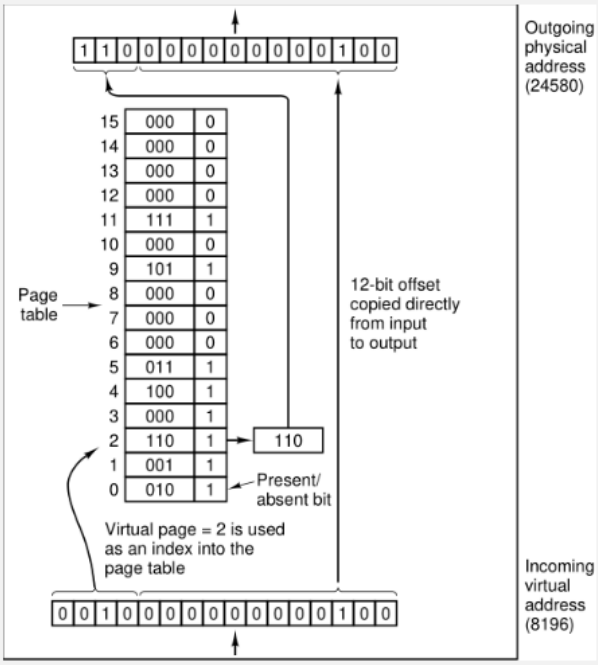
### Internminne

**14)** L1 cache

## 15)

## MMU

Anta at en innkommende virtuell adresse til MMU er 8. Hva blir da den utgående fysiske adressen?

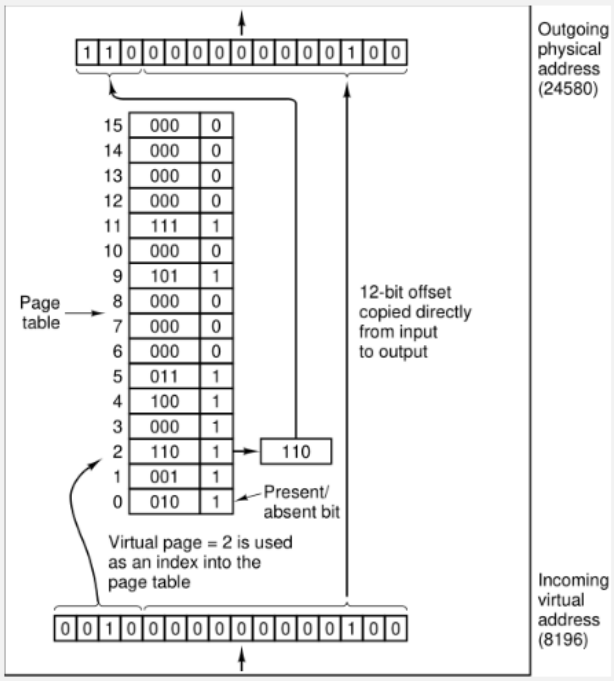



**Velg ett alternativ:**

- Gir page fault
- 8
- 4096
- 24580
- 8204
- 8200 ✓
- 4104
- 8196
- 24588
- 24584

**MMU**

Anta at en innkommende virtuell adresse til MMU er 12. Hva blir da den utgående fysiske adressen?




**Velg ett alternativ:**

- 4108
- 24588
- 8196
- 8200
- 12
- 24592
- 24580
- 4096
- Gir page fault
- 8204 ✓

**Linus kommandolinje**

## 16)

- Hvilken mappe referer .. (to punktum) til i et bash-shell? - Mappen over den du står i
- Hvilken mappe referer . (ett punktum) til i et bash-shell? - Mappen du står i
- Hvilken mappe referer / til i et bash-shell? - Roten av filsystemet

## 17)

- Hvilken Linux-kommando lager en tom fil med navn fil.txt? - touch fil.txt
- Hvilken Linux-kommando lager en mappe med navn ny? - mkdir ny

## 18)

- Hva slags Linux-filer pleier å ha filendelse .lin ? -Dette er ikke en vanlig filendelse i Linux
- Hva slags Linux-filer pleier å ha filendelse .com ? -Dette er ikke en vanlig filendelse i Linux

### Bash-scripting

## 19)

```
#! /bin/bash

fra=$(date -d $1 +%s)
til=$(date -d $2 +%s)
diff=$(( $til - $fra ))

echo "$diff"
```

## 20)

```
#! /bin/bash

fra=$(date -d $1 +%s)
til=$(date -d $2 +%s)
diff=$(( $til - $fra ))
diffTMS=$(date -d@$diff -u +%H:%M:%S )

echo "$diffTMS"
```

## 21)

```
#! /bin/bash

video=$1
vid=$(basename -s .mp4 $1)
shift

part=0

while [ $# -gt 0 ]  # Så lenge det er noe i $*
do
   start=$1
   shift
   stop=$1
   toSec=$(date -d "$stop" "+%s")
   fromSec=$(date -d "$start" "+%s")
   diffSec=$(( $toSec - $fromSec ))
   diff=$(date -d@$diffSec -u +%H:%M:%S)
   ((part++))
   ./fromTo.sh $video $vid$part.mp4 $start $diff
   shift          # skyver ut $1 og legger neste argument i $1
done
```

### PowerShell

**22)** Den fundamentale forskjellen er at en Linux-kommando kun returnerer tekst mens en PowerShell-kommandoen returnerer et helt objekt med metoder og properties. Default så ser man bare tekst-resultatet, men om man sender output til en pipe eller trekker ut properties, vil man i PowerShell-tilfellet se forskjellen.

**23)**

```
$start = date $args[0]
$stop  = date $args[1]

$diff = $stop - $start

$h = '{0:d2}' -f $diff.Hours
$m = '{0:d2}' -f $diff.Minutes
$s = '{0:d2}' -f $diff.Seconds

echo "${h}:${m}:$s"
```