Eksamenen har 13 spørsmål og er delt inn i tre deler, ordnet etter spørsmålets størrelse. Du må svare på alle spørsmålene. Du kan bruke poengene per spørsmål som en veiledning for hvor mye tid du skal tildele. Poengene summerer seg til 100.

I denne eksamen kan du ikke bruke støttemateriell eller kalkulatorer.

Lykke til!

1 Mange OS har et "skall" (engelsk: "shell") som er en tekstorientert grensesnitt mellom OS og brukeren.

Beskriv kort hvilke systemkall et skall-program må utføre slik at det blir mulig å kjøre ulike andre programmer fra skallet, og hva disse systemkallene gjør. Det er nok med en minimal-skall her, der et program utføres om gangen, uten filhåndtering eller signaler.

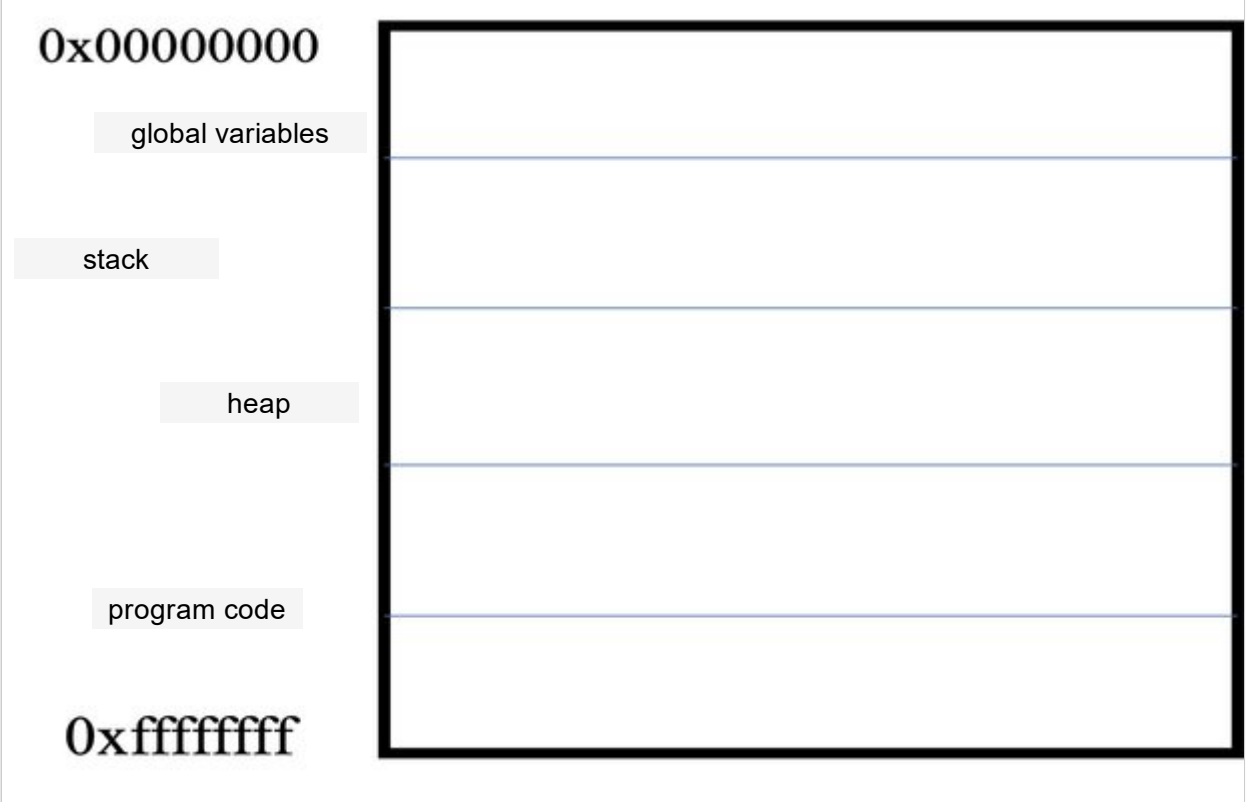
Skriv ditt svar her

Format

Words: 0

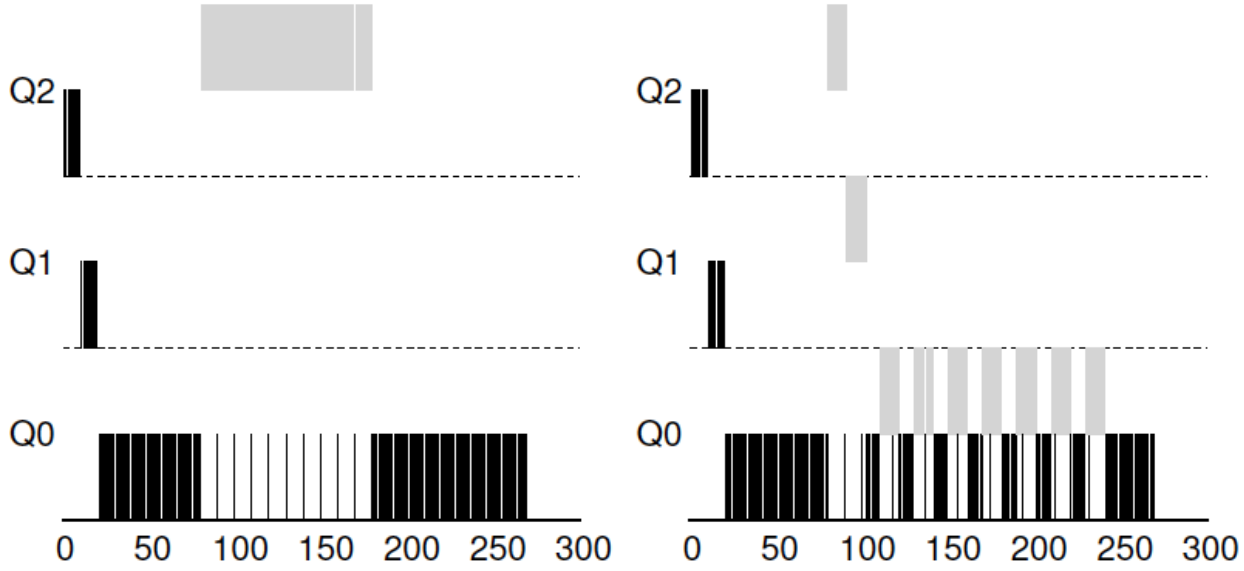
Maks poeng: 5

2 Dra og slipp hvert av de 4 elementene til minneblokken slik at rekkefølgen danner vanlig minneoppsett.



Maks poeng: 4

3 Forklar problemet med scheduling vist på venstre side og hvordan scheduling policy for Multi-level Feedback Queue løser problemet (vist på høyre side). De svarte og grå blokkene tilhører henholdsvis to prosesser.



Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

4 Anta at x = 20 og y = 30 i utgangspunktet. Etter den samtidige utførelsen av **tråd 1** og **tråd 2**, hva er de mulige verdiene for x og y etter at begge trådene har fullført utførelsen? Skriv ned de mulige parene av x og y - ett for hvert tilfelle. For eksempel er (x=4, y=7) og (x=6, y=5) to forskjellige tilfeller.

TIPS: Hvilken av variablene kan erstattes med startverdien, uavhengig av utførelsesrekkefølgen til de to trådene?

tråd 1	**tråd 2**
x = x + y + 100	y = x - y + 50

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

5 RAID-5 tåler 1 diskfeil og ikke mer. Gitt følgende harddisksett med RAID-5-konfigurasjon der paritetsblokkene er merket med gul farge, finn den manglende informasjonen på **Disk 1**. Hvilken metode brukte du for å beregne dem?

Disk 0	Disk 1	Disk 2	Disk 3
101		000	010
111		110	000
011		111	110
010		010	011

Skriv ditt svar her

Format

Words: 0

Maks poeng: 3

6 Anta at det tar én tidsenhet for CPU-en å gi en I/O-instruksjon, tre tidsenheter for I/O-enheten å utføre oppgaven, og deretter én tidsenhet til for CPU-en å fullføre I/O-instruksjonen. Samlete tiden fra å gi en I/O-instruksjon til å fullføre I/O-instruksjonen er da 1+3+1=5 tidsenheter, hvorav 2 tidsenheter brukes på CPU.

Vi skal kjøre to prosesser, der én prosess utfører 2 I/O-instruksjoner og den andre prosessen utfører 8 CPU-instruksjoner. Anta at prosessen som utfører I/O-instruksjoner starter først, og systemet vil bytte til en annen prosess når CPU-en blir inaktiv.

Beregn prosentandelen av tiden som CPUen er i bruk fra starten av den første prosessen til begge prosessene er fullført. Gjør dette i de følgende to systemene:

1.) Når en I/O-instruksjon fullføres, kjøres IKKE prosessen som utstedte den med en gang; snarere kjører den andre prosessen videre.

2.) Når en I/O-instruksjon fullføres, kjøres prosessen som utstedte den umiddelbart.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 7

7 Beregn gjennomsnittlig behandlingstid og responstid når du kjører 6 jobber med **Shortest Time-to-Completion First (STCF)**-planleggeren. Anta at de første 3 jobbene med lengde 1, 2 og 3 ankommer systemet omtrent samtidig på tidspunkt 0, de neste 2 jobbene med lengde 1 og 2 ankommer systemet omtrent samtidig på tidspunkt 3, og siste jobb med lengde 1 kommer inn i systemet på tid 7.

- Hver gang en ny jobb kommer inn i systemet, bestemmer STCF-planleggeren hvilken av de gjenværende jobbene (inkludert den nye jobben) som har minst tid igjen, og planlegger deretter den.

Hva er den totale behandlingstiden og responstiden ved bruk av henholdsvis STCF?

Skriv ditt svar her

Format

Words: 0

Maks poeng: 7

8 Anta at størrelsen på header er 2 byte for hvert **malloc()**-kall og heapen starter på adresse 100 med maksimal størrelse 10 byte totalt. Observer resultatene av hver **malloc()** og **free()**-forespørsel vist nedenfor.

Innledende haug:
free list : [adresse:100 størrelse:10]

ptr0 = malloc(2); //returned 102
free list : [adresse:104 størrelse:6]

free(ptr0);
free list : [adresse:104 størrelse:6][adresse:100 størrelse:4]

ptr1 = malloc(1); //returned 102
free list : [adresse:104 størrelse:6][adresse:103 størrelse:1]

free(ptr1);
free list : [adresse:104 størrelse:6][adresse:103 størrelse:1][adresse:100 størrelse:3]

ptr2 = malloc(5);

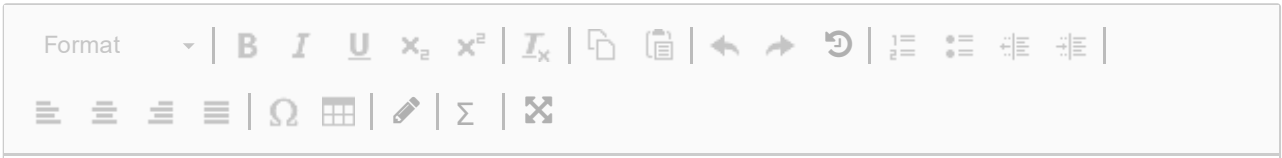
Oppgaver:

Hvordan er free-listen sortert?

Hvilken policy for administrasjon av ledig plass brukes (first-fit, best-fit eller worst-fit)?

Hva kommer til å skje i den siste linjen (malloc 5)? Hvordan løser vi problemet?

Skriv ditt svar her



Words: 0

Maks poeng: 5

9 Når fysisk minne ikke er stor nok, vil operativsystemet bytte sider ut av minnet til swap og bytte sider til minnet fra swap. Anta en fysisk minne som kan inneholde maksimalt 3 sider og at minnet er tomt i utgangspunktet. Spor minnetilstandsendringene ved tilgang til sider med sidenummer 1,0,1,2,0,1,3,0,2,4,1,4 én etter én i en sekvens ved hjelp av **Least-Frequently-Used (LFU)** byttepolicyen.

- LFU-byttepolicyen holder styr på **antall ganger en side refereres** til i minnet. Når minnet er fullt, velger den siden med den laveste referansefrekvensen så langt for å bytte ut.

Hva er treffprosenten i dette tilfellet? Skriv ned beregningstrinnene dine.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 7

10 Når dette C-programmet kompileres,

```c
int add(int a, int b) {
    return a+b;
}
```

```c
int main() {
    int i = 11;
    int j = 22;
    return add(i, j);
}
```

får vi følgende assembly-kode:

```
1   add:
2           pushq   %rbp
3           movq    %rsp, %rbp
4           movl    %edi, -4(%rbp)
5           movl    %esi, -8(%rbp)
6           movl    -4(%rbp), %edx
7           movl    -8(%rbp), %eax
8           addl    %edx, %eax
9           popq    %rbp
10          ret
11  main:
12          pushq   %rbp
13          movq    %rsp, %rbp
14          subq    $16, %rsp
15          movl    $11, -4(%rbp)
16          movl    $22, -8(%rbp)
17          movl    -8(%rbp), %edx
18          movl    -4(%rbp), %eax
19          movl    %edx, %esi
20          movl    %eax, %edi
21          call    add
22          leave
23          ret
```

Definisjoner

Når programmet starter, antar vi at både **rsp**- og **rbp**-registeret holder tallet 1000. Det er bunnen av *stack*'en.

pushq X - trekker 8 fra rsp, og lagrer X på stedet den nye rsp peker på
popq X - lagrer verdien som rsp peker på i X, og øker rsp med 8
leave - gjør det samme som **movq %rbp, %rsp** og så **popq %rbp**

Svar på følgende delspørsmål:

(a) Hvilke registere er det som holder funksjonsparametere a og b når add-funksjonen kalles fra main?

(b) Hvilket register er det som holder returverdien til add-funksjonen når den avslutter på linje 10?

(c) Hva er rollen til rsp og rbp?

(d) Hvordan ser stack'en ut når programmkjøringen havner på assembly-linje 6. Du kan fomatere det som vist under (erstatt tallene!)

adresse: verdi (kort forklaring)
...
...
...
920: 22 (tallet j)
932: 998 (verdien til rbp når main starter)
1000: -/- (bunnen)

Skriv ditt svar her

Format

Words: 0

Maks poeng: 14

11 Vi skal jobbe med et enkelt filsystem som holder 2 bitmaps til inodes og data, selve inodes og så datablokker, Brukeren vil legge til mange flere linjer til en allerede eksisterende fil.

Da finnes det minst 3 skrive-operasjoner som må utføres:

- de nye datalinjene må skrives til nye datablokker
- inode til filen må oppdateres for å peke til de nye blokkene
- data-bitmap må markeres med de nye blokkene som er opptatt.

Oppgave

PCen kræsjer og noen av disse 3 operasjonene ikke blir utført. I hver situasjon nevnt under, skal du beskrive

- (1) symptomene som er synlige for brukeren når maskinen starter opp igjen, både de som er synlige med en gang og de som bare viser seg etter en stund.
- (2) om det er mulig for et automatisk system til å finne feilen ved oppstart av PCen

a) når ny data ikke blir lagt inn i datablokkene, mens inode og bitmap oppdateres riktig.

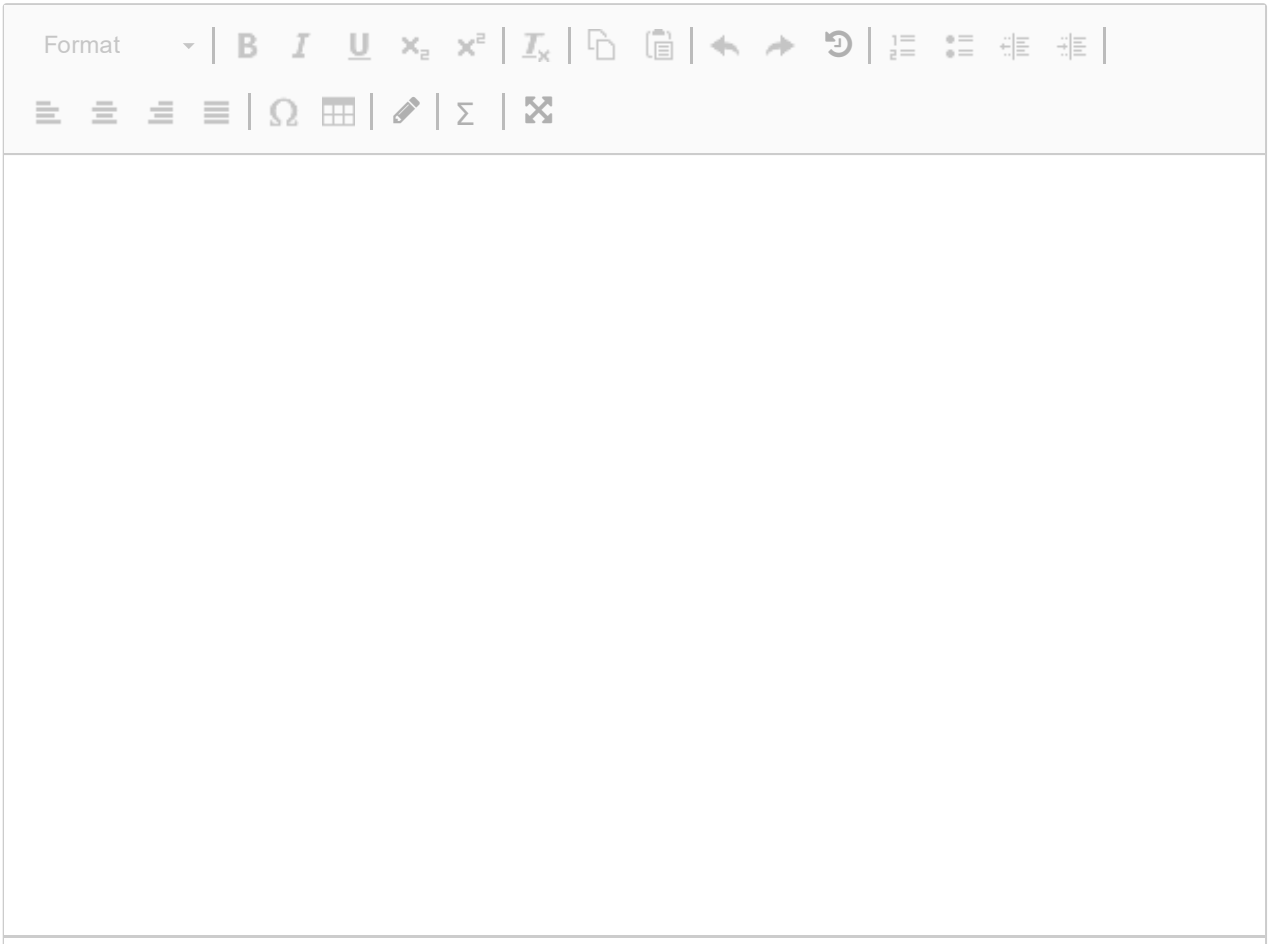
b) når inoden ikke blir oppdatert, mens bitmap og datablokkene oppdateres riktig.

c) når bitmap ikke blir oppdatert, mens inoden og datablokkene oppdateres riktig.

En mulig løsning som gjør filsystemer mer robust mot tilfeldige kræsj kalles for "Journalling".

d) Beskriv hvordan en enkel journalling-løsning kan se ut, og hvordan det kan hjelpe i en kræsj-situasjon.

Skriv ditt svar her



Words: 0

Maks poeng: 10

12 Vi ser på et enkelt filsystem. Root-mappen i filsystemet (dvs. "/") tilsvarer inode 2.

Filsystemet på disken inneholder:

Bitmap for data

block	occupied?
1-5	0 0 0 0 1
6-10	0 1 1 0 1
11-15	1 1 0 0 0

Bitmap for inodes

inode	occupied?
1-5	1 1 1 1 1
6-10	1 0 0 0 0

Inodes

Nr.	Type	reference count	data blocks
2	dir	3	5
3	file	1	7, 8
4	file	1	10
5	dir	2	11
6	file	1	12

Data blocks:

Data block 5
2 . // 2 .. // 3 worlds.txt // 5 my_data // 4 alice.txt //

...

Data block 7
No one would have believed in the last years of the nineteenth century that
this world was being watched keenly and closely by intelligences greater tha

Data block 8
n man's and yet as mortal as his own; that as men busied themselves about
their various concerns they were scrutinised and studied.

...

Data block 10
Alice was beginning to get very tired of sitting by her sister on the bank.

Data block 11
5 . // 2 .. // 6 bergen_rain.csv //

Data block 12
jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec\n255,210,200,140,110,135,1
55,210,245,270,270,290\n

Oppgaver

OBS!
Hver deloppgave tar utgangspunktet i situasjonen ovenfor.
Effektene skal ikke kombineres!
OBS!

a) Terminalen er i mappen "/". Brukeren skriver
cat worlds.txt
Skriv kort om de ulike ting som nå må utføres i filsystemet.

b) Terminalen er i mappen "/". Brukeren skriver
ln my_data/bergen_rain.csv rain.csv
Hvilke steder i filsystemet blir endret? Hva er endringene?

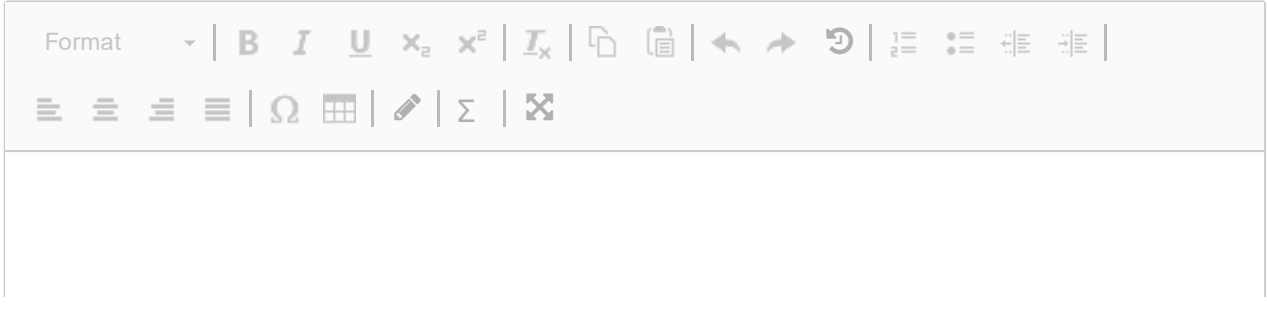
c) Terminalen er i mappen "/". Brukeren skriver
rm alice.txt
Hvilke steder i filsystemet blir endret? Hva er endringene?

d) Terminalen er i mappen "/". Brukeren skriver
shred alice.txt
Hvilke steder i filsystemet blir endret? Hva er endringene?

e) Terminalen er i mappen "/my_data". Brukeren skriver
echo HelloWorld > greeting.txt
Hvilke steder i filsystemet blir endret? Hva er endringene?

f) Terminalen er i mappen "/my_data". Brukeren skriver
mkdir more_data
Hvilke steder i filsystemet blir endret? Hva er endringene?

Skriv ditt svar her



Words: 0

Maks poeng: 14

13

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Anta at den fysiske minnestørrelsen er 256KB, adresseplassstørrelsen for prosessen vi ser på er 16KB, og sidestørrelsen er 2KB. Formatet til sidetabellen er beskrevet som følgende: Den høye ordensbiten (lengst til venstre) i sidetabelloppføringen er GYLDIG bit. Hvis biten er 1, er resten av oppføringen siderammenummeret (PFN). Hvis biten er 0, er siden ikke gyldig. En tabell fra heksadesimal til binær kan finnes i den vedlagte filen.

Sidetabell (fra oppføring 0 ned til maks størrelse)
[0] 0x861
[1] 0x000
[2] 0x000
[3] 0x833
[4] 0x826
[5] 0x000
[6] 0x874
[7] 0x824

Gitt sidetabellen ovenfor, svar på hvert av de følgende spørsmålene for å avgjøre hvilke av disse tre virtuelle adressene som er gyldige og hvilke ikke?

Virtuelle adresser :
VA 0x2793 --- PA eller ugyldig ?
VA 0x1008 --- PA eller ugyldig ?
VA 0x3ee5 --- PA eller ugyldig ?

(1) Hvor mange sider kan adresseområdet inneholde? (1 poeng)
(2) Hvor mange biter kreves det i en virtuell adresse (VA) for å adressere hele prosessens adresserom? (1 poeng)
(3) Hvor mange toppbiter av en virtuell adresse (VA) brukes i dette tilfellet for å indikere sidenummeret til siden som VA lokaliserer? (1 poeng)
(4) Hvor mange biter i en virtuell adresse (VA) indikerer offset av VA på siden? (1 poeng)
(5) Hvor mange bits kreves i en fysisk adresse (PA) for å adressere hele det fysiske minnet? (1 poeng)
(6) For hver virtuelle adresse ovenfor, svar på følgende spørsmål:
(6-1) Beregn sidetallet. (1 poeng hver)
(6-2) Er den virtuelle adressen gyldig eller ikke?
(6-2-a) Hvis den virtuelle adressen er ugyldig, forklar hvorfor den er ugyldig. (2 poeng hver)
(6-2-b) Hvis den virtuelle adressen er gyldig, hva er det tilsvarende siderammenummeret? (1 poeng hver)
(6-3) Skriv ned den tilsvarende fysiske adressen (PA) i heksadesimalt format.
Tips: PA dannes av sammenkoblingen av siderammenummeret og offset i binært format. (2 poeng hver)

Skriv ditt svar her

Format

Words: 0

Maks poeng: 14