Eksamensinformasjon

Fakultet: Teknologi, kunst og design
Utdanning: Teknologiske Fag
Emnenavn: Operativsystemer
Emnekode: DATA2500

Bokmål
Dato: 10 juni 2021
Tid: kl. 09.00-12.00 (3 timer)

Hjelpemidler:
Alle hjelpemidler er tillatt.

Andre opplysninger:
Les nøye gjennom oppgavene før du begynner og pass på å besvare alle spørsmålene.
Sett gjerne egne forutsetninger dersom du synes oppgaveteksten er uklar. Beskriv forutsetningene og løs oppgaven utifra dem.

Maks poeng på de fleste av deloppgavene er 10 og de bidrar 3.33 % hver til sluttpoengsummen som maksimalt er 300. Men legg merke til at det er noen unntak:

Oppgave 6, 7, 8 og 9 gir inntil 5 poeng
Oppgave 16, 17 og 20 gir inntil 30 poeng

Support:
Har du spørsmål til oppgaven under eksamen må du ta kontakt med Hårek Haugerud på denne Zoom-linken:
https://oslomet.zoom.us/j/68327977240?pwd=YTk4V3dsSmZJQ3pHVjUzR2wxbVdqZz09
Meeting ID: 683 2797 7240
Password: 968638

Har du spørsmål om eller problemer med Inspera ta kontakt med eksamen-tkd@oslomet.no eller på telefon 67 23 88 88. Vi oppfordrer at kun akutte henvendelser kommer inn på telefon da denne har begrenset bemanning. Epost henvendelser vil bli behandlet fortløpende, husk å skrive inn studentnummer, kandidatnummer og emnekode i mailen for raskere behandling.

Opplever du problemer med innleveringen må du ta kontakt med eksamenskontoret **før innleveringsfristen går ut.**

1 Systemkall

Et systemkall utføres når

Velg ett alternativ:

- en datamaskin skrus på.
- systemprogramvare startes opp.
- en prosess ber operativsystemkjernen om en tjeneste.
- operativsystemkjernen ber en prosess om en tjeneste.
- operativsystemkjernen kontaktes utenfra ved at det skjer et system interrupt.
- en ny enhet i systemet tilordnes et nytt navn.

Maks poeng: 10

2 Modusbit

Hva er en prosessors modusbit?
Velg ett alternativ:

- Et bit som sier om prosessoren er aktiv eller ikke
- Et bit som endrer CPU'ens klokkefrekvens
- Et bit som gir et program root-rettigheter selvom det kjøres av en vanlig bruker
- Et bit som kan begrense aksess til minne og instruksjoner
- Et bit som settes når det kommer et interrupt
- Et bit som avgjør om det skal utføres en context switch

Maks poeng: 10

3 CPU-migration

På en Linux-server med to CPUer (cores, regneenheter) har du et program **regn** som er 100% CPU-intensivt og hele tiden bruker så mye CPU-tid det kan. Du har også et script **run2.sh** som starter to instanser av programmet **regn**:

```
Linux$ cat run2.sh
#! /bin/bash
./regn &
./regn
```

Du bruker så programmet **perf** for å finne ut om **regn**-programmet noen gang i løpet av tiden det kjører flytter

seg mellom CPUene ved en såkalt cpu-migration:

```
Linux$ perf stat -B -e cpu-migrations ./run2.sh
 Performance counter stats for './run2.sh':
                 0      cpu-migrations
       9,996199705 seconds time elapsed
```

Når du bruker **perf** på denne måten vises det statistikk for den siste av **regn**-prosessene, siden den første legges i bakgrunnen som en selvstendig prosess. Du observerer at det ikke skjer noen cpu-migrations og at det tar ca 10 sekunder å fullføre kjøringen. Du har også et script **run3.sh** som starter tre instanser av programmet **regn**:

```
Linux$ cat run3.sh
#! /bin/bash
./regn &
./regn &
./regn
```

Så bruker du **perf** for å vise statistikk fra denne kjøringen og du får følgende resultat:

```
Linux$ perf stat -B -e cpu-migrations ./run3.sh
 Performance counter stats for './run3.sh':
               165      cpu-migrations
      15,146034568 seconds time elapsed
```

Forklar kort hvorfor den siste av **regn**-prosessene nå tar ca 15 sekunder på å fullføre og hvorfor den siste av **regn**-jobbene i scriptet denne gangen flytter seg en rekke ganger (165) fra CPU til CPU.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 10

4 RUN og CMD

Anta at du skal forklare for noen som relativt nylig har begynt å bruke Docker-containere hva som er forskjellen på instruksjonene **RUN** og **CMD** i en Dockerfile. For å forklare dette lager du to små script som du kaller **run.sh** og **cmd.sh** og en **Dockerfile** som bruker scriptene på følgende måte på en Ubuntu-server:

```
Linux# cat run.sh
#! /bin/bash
echo "ECHO FRA run.sh"
Linux# cat cmd.sh
```

```
#! /bin/bash
echo "ECHO FRA cmd.sh"
Linux# cat Dockerfile
FROM ubuntu
COPY run.sh run.sh
COPY cmd.sh cmd.sh
RUN ./run.sh
CMD ["./cmd.sh"]
```

Deretter bygger du og kjører containeren med følgende kommandoer (vist med boldface skrift) og output blir som vist under:

```
Linux# docker build -t runcmd .
Sending build context to Docker daemon  8.192kB
Step 1/5 : FROM ubuntu
 ---> 74435f89ab78
Step 2/5 : COPY run.sh run.sh
 ---> c0045ffc4e3e
Step 3/5 : COPY cmd.sh cmd.sh
 ---> c5b37da943ab
Step 4/5 : RUN ./run.sh
 ---> Running in 40b520b40c92
ECHO FRA run.sh
Removing intermediate container 40b520b40c92
 ---> 3b3b2bcf0850
Step 5/5 : CMD ["./cmd.sh"]
 ---> Running in 905ba5d97368
Removing intermediate container 905ba5d97368
 ---> 9e9d5648f376
Successfully built 9e9d5648f376
Successfully tagged runcmd:latest
Linux# docker container run runcmd
ECHO FRA cmd.sh
Linux#
```

Forklar kort for Docker-nybegynneren hva som skjer når du kjører de to kommandoene og utifra resultatene hva som er forskjellen på **RUN** og **CMD** i en Dockerfile. Ikke forklar hver minste detalj, ta bare med det viktigste for å forstå forskjellen på **RUN** og **CMD**.

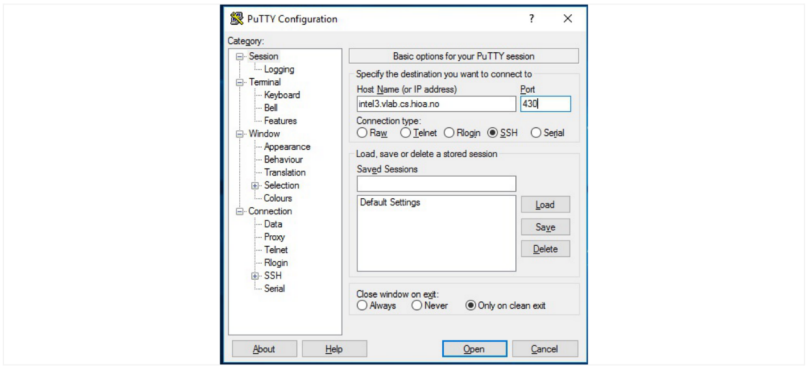
Skriv ditt svar her

Format

Words: 0

Maks poeng: 10

5 Innlogging på Linux-VM



På StudentWeb har du fått oppgitt et kandidatnummer, det kan for eksempel være et tall som 430 og vi vil bruke dette tallet som eksempel, men bytt ut 430 med ditt eget kandidatnummer når du gjør disse oppgavene. Det samme kandidatnummeret står øverst i venstre i vinduet når du starter eksamen i Inspera.

På serveren intel3.vlab.cs.hioa.no har hver kandidat sin egen Linux-VM som man som en del av eksamen skal logge inn på for å løse oppgaver (disse er helt like Linux-VMene som dere brukte under kurset, med navn som os13, og de er også egentlig Docker-containere). Alle disse Linux-VMene har forskjellig innhold, slik at svaret på alle oppgavene vil være forskjellig for hver kandidat. For eksempel vil svaret på et spørsmål som "Totalt hvor mange filer finnes det på din Linux-VM" variere for hver kandidat, fordi det ikke ligger samme antall filer på hver VM.

Anta at du er kandidat nummer 430. Da må du først laste ned riktig passord fra denne web-siden:

https://www.cs.hioa.no/~haugerud/os.php

Skriv inn kandidatnummeret (ditt eget og ikke 430) og trykk på 'Send inn nummeret'-knappen. Hvis du skriver inn et gyldig kandidatnummer får du da et passord som er av samme type som for os-VMene og kan være noe lignende som **bien6tuft**. Det er viktig at du bare henter passord for din egen VM, hvilken IP som henter hvilket passord blir loggført. Det er viktig at du ikke går inn på andre kandidaters servere.

Logg så inn på din VM med kommandoen

ssh -p 430 kan430@intel3.vlab.cs.hioa.no

hvis du logger inn fra et bash-shell (Mac eller Linux) på din hjemme-PC og bruk så passordet du fikk fra websiden os.php på den samme hjemme-PCen (se instruksjoner for putty fra Windows lenger ned). Opsjonen -p 430 betyr at du bruker port-nummer 430 og ikke standard ssh-portnummer som er 22. Dette er laget slik for at hver kandidat skal nå sin egen VM. kan430 er brukernavnet du logger inn med. For både port-nummeret og brukernavnet må du bytte ut 430 med ditt eget kandidatnummer.

Hvis du logger deg på direkte fra putty (se figur) må du bruke Host Name intel3.vlab.cs.hioa.no, skrive inn dtt kandidatnummer som port number (430 i eksempelet), bruke brukernavn kan430 og passordet du lastet ned (igjen, bytt ut 430 med ditt kandidatnummer).

For å svare på denne oppgaven, skriv inn følgende:

Kandidatnummer
Passordet du hentet fra websiden os.php
"Din IP-adresse" som du fikk fra websiden os.php
Innholdet av filen **file1** som ligger øverst i hjemme-mappen til din bruker (kan430)

Skriv ditt svar her

Format

Words: 0

Maks poeng: 10

6 Fil på Linux-VM

I filesystemet til brukeren du logget på med på Linux-VM ligger det en tekst-fil med navn **file2**. Finn denne filen, kopier og paste ut innholdet av filen her:

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

7

Filer på Linux-VM

I filesystemet til brukeren du logget på med på Linux-VM er det en mappe som heter **files**. Hvor mange filer inneholder denne mappen?

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

8

xfile på Linux-VM

I hjemme-mappen til brukeren root ligger det en fil med navn **xfile**. Kopier og paste ut innholdet av denne tekstfilen her:

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

9

Filrettigheter

I filesystemet til brukeren du logget på med på Linux-VM ligger det en tekst-fil med navn **rett.txt**. Hvilke filrettigheter har denne filen? (skriv svaret på formen **rwxrw-r--** hvis dette er filrettighetene filen har)

Skriv ditt svar her

Format

Words: 0

Maks poeng: 5

10

Skjulte filer

Når en Linux-server blir hacket skjer det ofte at hackerene med enkle midler prøver å skjule mapper og filer. I filesystemet til brukeren du logget på med på Linux-VM finnes det en kjørbar binær fil med et navn og en plassering som gjør at den er litt vanskligere å finne enn normalt. Finn denne filen, kjør den, kopier output og paste ut her:

Skriv ditt svar her

Format

Words: 0

Maks poeng: 10

11 Linking av filer

Øverst i filesystemet til brukeren du logget på med på Linux-VM ligger det to binære filer med navn **main** og **rand**. For å kunne kjøre disse filene må de først linkes sammen med **gcc**. Gjør det fra kommandolinjen, kjør programmet og skriv inn den output programmet gir her (husk at du kan installere det du måtte trenge for å gjøre denne oppgaven siden du har sudo-rettigheter):

Skriv ditt svar her

Format

Words: 0

Maks poeng: 10

12 Internminne

Hva er betydningen av begrepet **Random Access Memory**?

Velg ett alternativ

- At alle enheter har tilgang til minnet
- At det er tilfeldig hvilken byte som hentes først om to byte leses samtidig
- At adressene til hver byte i minnet allokeres tilfeldig av operativsystemet
- At det går like fort å skrive til minnet som å lese fra minnet
- At data som lagres kan bli lagt hvor som helst i minnet
- At det tar like lang tid å hente en byte fra hvor som helst i minnet

Maks poeng: 10

13

Internminne

Hva deles det virtuelle adresserommet opp i?
Velg ett alternativ

- Pages
- Partitions
- Sectors
- Stacks
- Heaps
- Tracks
- Sections
- Frames
- Swaps

Maks poeng: 10

14

Internminne

I et C-program er det et integer (int) array som har 2*1024*1024 elementer. Hvor mange byte utgjør dette arrayet?

Velg ett alternativ:

- 512 KiB
- 1 GiB
- 16 GiB
- 4 Gib
- 2 MiB
- 4 MiB
- 1 MiB
- 32 GiB
- 256 KiB
- 8 GiB
- 16 MiB
- 8 MiB
- 32 MiB

Maks poeng: 10

15 RAID 3

Anta at du har et RAID 3 oppsett med 4 disker og en paritets-disk. Alle dataene på disk 3 har gått tapt etter en disk-crash. Bruk informasjonen fra paritets-disken til å gjennopprette det som var på disk 3 og skriv inn 0 eller 1 for hver rad på disk 3.

RAID 3

disk 1	disk 2	disk 3	disk 4	paritets-disk
1	1		1	1
1	0		0	0
1	1		1	0
0	1		1	1
1	1		0	1
0	0		0	0

Maks poeng: 10

16 $RANDOM

Du ønsker å bruke tilfeldige (random) tall i et bash-script og du finner følgende på nettet:

```
# $RANDOM returns a different random integer at each invocation.
# Nominal range: 0 - 32767 (signed 16-bit integer).
# If you need a random int within a certain range, use the 'modulo'
# operator %.
# This returns the remainder of a division operation.
```

Deretter eksperimenterer du litt med dette og gjør følgende i et bash-shell (kommandoene du skriver er vist i boldface):

```
Linux$ echo $RANDOM
570
Linux$ echo $RANDOM
13755
Linux$ echo $(( $RANDOM % 10 ))
7
Linux$ echo $(( $RANDOM % 10 ))
8
Linux$ echo $(( $RANDOM % 10 ))
9
Linux$ echo $(( $RANDOM % 10 ))
6
Linux$ echo $(( $RANDOM % 2 ))
0
Linux$ echo $(( $RANDOM % 2 ))
1
Linux$ echo $(( $RANDOM % 2 ))
1
Linux$ echo $(( $RANDOM % 2 ))
0
```

Bruk det du har lært om $RANDOM til å skrive et bash-script **4bit.sh** som skriver ut 10 linjer med tilfeldige (random) 4-bits tall, slik som vist under:

```
Linux$ ./4bit.sh
1111
0011
0101
0110
1011
0111
0100
```

```
0011
0011
1111
```

Skriv ditt svar her

```
1
```

Maks poeng: 30

17 $RANDOM

Lag et bash-script som bruker $RANDOM til å lage 10 mapper med tilfeldige student-brukernavn av typen **s435302**, etter oppskriften: "bokstaven s fulgt av 6 siffer mellom 0 og 9". I hver av disse mappene skal scriptet deretter lage tomme filer av typen **fil1.txt** (se eksempelet under). Antallet slike nummerert filer skal for hver mappe være et tilfeldig tall mellom 10 og 20.

(Du trenger ikke sjekke for den lille muligheten at to like student-navn genereres og tallet kan begynne på 0 men skal ha 6 siffer.)

Etter at scriptet har blitt kjørt, kan det se slik ut når man lister mappene; utskriften under har blitt kuttet etter de to første mappene, det skal være 8 slike mapper til.

```
Linux$ ls -l s*
s435302:
total 0
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil10.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil11.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil12.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil1.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil2.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil3.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil4.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil5.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil6.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil7.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil8.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil9.txt
s460176:
total 0
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil10.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil11.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil12.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil13.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil14.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil15.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil16.txt
```

```
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil1.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil2.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil3.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil4.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil5.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil6.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil7.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil8.txt
-rw-r--r-- 1 haugerud haugerud 0 mai   28 13:36 fil9.txt
```

Skriv ditt svar her

```
1
```

Maks poeng: 30

18

PowerShell pipe

I PowerShell kan man, akkurat som i et bash-shell, sende output fra en kommando til en annen kommando ved hjelp av en pipe. For eksempel er

ls | sort

en gyldig kommando i begge tilfeller. Hva er den viktigste forskjellen på det som sendes igjennom en pipe når man sammenligner PowerShell og bash?

Velg ett alternativ

- I PowerShell krypteres alt som sendes slik at overføringen er sikrere
- I PowerShell sendes hele objekter med metoder og egenskaper
- I PowerShell sendes også ikoner, noe som egner seg bedre i et grafisk brukergrensesnitt
- I bash er denne teknologien mer avansert fordi den har blitt utviklet gjennom 40 år
- I bash kan informasjonen sendes til andre kanaler, som stderr og /dev/null
- I bash komprimeres informasjonen som sendes slik at det går raskere

Maks poeng: 10

19

Get-Random

Du leser litt om PowerShell cmdlet'en **Get-Random** og prøver den ut:

```
The Get-Random cmdlet gets a randomly selected number. If you submit
a collection of objects to Get-Random, it gets one or more randomly
selected objects from the collection.
PS C:\Users\os> Get-Random
49903642
PS C:\Users\os> Get-Random
570616567
PS C:\Users\os> ps | Get-Random
Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    129      10     3720       7800              3656   0 svchost


PS C:\Users\os> ps | Get-Random

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    312      14     3012      10312       0,08   1220   1 MSASCuiL
```

Bruk det du har lært om Get-Random til å skrive en PowerShell-kommando som skriver ut en tilfeldig trukket fil eller mappe fra mappen du står i og alle dens undermapper.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 10

20

Get-Random

Du leser litt mer om **Get-Random** og tester ut følgende kommando:

```
-Max number
     A maximum value for the random number.
     Get-Random returns a value that is less than the maximum (not equal).
PS C:\Users\os>  for($i=1;$i -lt 4;$i++){ (Get-Random -Min 0 -Max 10)}
5
9
0
```

Bruk dette til å skrive et PowerShell-script **rand.ps1** som skriver ut 10 tilfeldige 4bits tall og som gir noe slikt som dette når det kjøres:

```
PS C:\Users\os> .\rand.ps1
0111
1111
0101
1101
1010
0100
1001
1011
1100
1010
```

Skriv ditt svar her

```
1
```

Maks poeng: 30

21 4-bits register

Anta at du har et 4-bits register og du definerer 0000 som tallet 0, 0001 som 1, 0010 som desimaltallet 2 og så videre.

Skriv med fire binære siffer inn tallet 12 her:

Hva er med et desimaltall det største tallet du kan lagre i dette registeret?

Skriv inn ditt svar her: .

Maks poeng: 10

22 Simulerings-CPU

Ta utgangspunkt i den lille 4-bits maskinen som det ble kjørt en simulering av på forelesning og som ble brukt i ukeoppgavene. Tabellen nedenfor viser hva maskinkode for denne CPU'en betyr.

Instruksjonssett

binært Nr	operand1	operand2	Nr	Navn
0010	DR	tall	2	MOVI
0100	DR	SR	4	ADD
1100	DR	SR	12	CMP
1111	nr	nr	15	JNE

Hva betyr maskinkoden 00101011? Forklar det med ord eller med assemblykode for denne maskinen.

Skriv inn ditt svar her:

Format

Words: 0

Maks poeng: 10

23

Simulerings-CPU

Forklar kort hvorfor ikke følgende assembly-instruksjon for simulerings-CPUen vil kunne skrives i maskinkode og utføres på simulerings-CPUen diskutert i forrige spørsmål:

```
MOVI R1 <- 5
```

Skriv ditt svar her

Format

Words: 0

Maks poeng: 10

24

Simulerings-CPU

Betrakt følgende assemblykode for maskinen:

```
0 MOVI R0 <- 0 (tallet 0 legges i R0)
1 MOVI R1 <- 1
2 MOVI R2 <- 3
3 MOVI R3 <- 1
4 ADD R3 <- R3 + R3
5 ADD R0 <- R0 + R1
6 CMP R0 R2
7 JNE 4 (Jump Not Equal 4, hopp til linje 4 hvis R0 != R2)
```

Forklar kort hva som skjer når koden kjøres og hva de fire registerne R0, R1, R2 og R3 inneholder når koden har kjørt ferdig og skal igang med kodelinje 8.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 10

25

Simulerings-CPU

For å kunne kjøre koden må CPU'en ha maskinkode. Bruk tabellen nedenfor og oversett linje 5-7 i assemblykoden vist under til maskinkode.

Instruksjonssett

binært Nr	operand1	operand2	Nr	Navn
0010	DR	tall	2	MOVI
0100	DR	SR	4	ADD
1100	DR	SR	12	CMP
1111	nr	nr	15	JNE

```
5 ADD R0 <- R0 + R1
6 CMP R0 R2
7 JNE 4 (Jump Not Equal 4, hopp til linje 4 hvis R0 != R2)
```

Skriv inn binærkoden for line 5-7 med nuller og enere her:

5 ☐☐☐☐☐☐☐☐

6 ☐☐☐☐☐☐☐☐

7 ☐☐☐☐☐☐☐☐

Maks poeng: 10

26

Simulerings-CPU

Skriv ned høynivåkode med C-syntaks som kunne gitt tilsvarende assemblykode som vist under om den ble kompilert for denne CPU-arkitekturen. Anta gjerne at registeret R0 tilsvarer variabelen **i** og at R3 tilsvarer variabelen **j**.

(du trenger bare å skrive selve algoritmen, ikke main() eller funskjons-definisjon)

```
0 MOVI R0 <- 0 (tallet 0 legges i R0)
1 MOVI R1 <- 1
2 MOVI R2 <- 3
3 MOVI R3 <- 1
4 ADD R3 <- R3 + R3
5 ADD R0 <- R0 + R1
6 CMP R0 R2
7 JNE 4 (Jump Not Equal 4, hopp til linje 4 hvis R0 != R2)
```

Skriv ditt svar her

```
1
```

Maks poeng: 10