# DAT103 – Oblig 2

## Task 1

### 1. Explain Lines 5–13 and what is the purpose of these lines?

```
5  ; Konstanter
6      cr equ 13 ; Vognretur
7      lf equ 10 ; Linjeskift
8      SYS_EXIT  equ 1
9      SYS_READ  equ 3
10     SYS_WRITE equ 4
11     STDIN     equ 0
12     STDOUT    equ 1
13     STDERR    equ 2
```

Alle linjer som er skrevet med semikolon og ut linjen er en kommentar. Første linje (linje 5) er bare en kommentar som sier at det som kommer etter kommentaren er definering av konstanter.
Linje 6 definerer "cr" (carriage return eller vognretur) og gir den et desimaltall med verdi 13.

I linje 7 tildeles "lf" (line feed eller linjeskift) verdien 10. Både "cr" og "lf" er varianter av kontrolltegn, hvor vognreturen brukes til å flytte markøren til begynnelsen av en linje, mens linjeskift flytter markøren til neste linje.

Linje 8 - SYS_EXIT gis verdien 1. SYS_EXIT er en konstant som blir brukt i systemkall til å avslutte et program.

Linje 9 – Her tildeles SYS_READ desimalverdien 3. SYS_READ er et systemkall for lesing av inndata.

Linje 10 – SYS_WRITE tildeles verdien 4, og er et systemkall for skriving av utdata.

Linje 11 – STDIN er kort for «standard in» og står for standard inndatastrøm. I linje 11 blir den tildelt verdien 4.

Linje 12 – Her defineres STDOUT, som motsatt av STDIN er standard utdatastrøm. På denne linjen får konstanten STDOUT desimalverdien 2.

Linje 13 – Her gis konstanten STDERR verdien 2. STDERR er kort for «standard error», og står for standardfeil.

Som nevnt over er alle disse linjene med på å definere konstanter som brukes i assembly-programmet. Her er linje 6 og 7 kontrolltegn, linje 8, 9 og 10 er systemkall, og linje 11, 12 og 13 er standard inn-, ut- og feilstrømmer.

### 2. What does Line 17 do? In particular, explain `resb` in this line.

```
17     siffer resb 4
```

I denne linjen er "siffer" navnet vi har satt på en variabel. "resb" er forkortelse for "reserve byte(s)". Denne brukes for å reservere et gitt antall byte i minnet for den oppgitte variabelen.

Tallet 4 angir størrelsen på minnet som skal reserveres. Kort oppsummert sier linje 17 at variabelen ‘siffer’ reserverer 4 bytes i minnet.

### 3. Which lines in the program are supposed to print the messages below? Explain your answer.

```
Skriv to ensifrede tall skilt med mellomrom.
Summen av tallene maa vaere mindre enn 10.
```

For å skrive ut en melding i assembly for Linux må man først lagre det som skal printes ut i en melding, for så å gjøre et systemkall med ‘SYS_WRITE’. Selve meldingen blir definert på linjene 21 og 22. Linjen som faktisk printer ut meldingen er linje 39 ‘int 80h’ under ‘.text’. Det er denne linjen som utløser systemkallet ‘SYS_WRITE’. ‘int 80h’ er et programvareavbrudd (eller systemkall) som svarer til avbruddsnummer 80. Når denne linjer blir utført, utløser det kjernen for å utføre et systemkall som baserer seg på verdien som er satt i ‘eax’-registeret, og det er her det blir spesifisert hvilket systemkall som skal utføres. Det er på linje 38 at programmet sier at register ‘eax’ skal utføre operasjonen ‘SYS_WRITE’, men det er ‘int 80h’ som setter i gang hele prosessen.

### 4. After executing the instruction on Line 43, which line the program counter will jump to?

```
43 |     call lessiffer
```

Linje 43 er et kall på "metoden" ‘lessiffer’, som starter på linje 86. Etter linje 43 hopper altså koden til linje 86.

### 5. Which lines are included in the `lessiffer` block?

‘lessiffer’-blokken starter på linje 86. ‘lessiffer’ er en metode som utfører det som er beskrevet under i "Lokke" (den delen som faktisk sjekker om det som skrives inn er et siffer, konverterer og returnerer det i ‘ecx’) og eventuelt "Feil" dersom brukeren oppgir noe ugyldig. Både "Lokke" og "Feil" bruker ‘ret’ på slutten av blokken sin, for å returnere til hoveddelen av programmet etter henholdsvis vellykket eller mislykket inndatahåndtering. Linjene som er inkludert i ‘lessiffer’ strekker seg altså fra starten (linje 86) til slutten av "Feil", altså linje 122.

### 6. The block `Feil` on Line 113 is called conditionally from Lines 104 & 106. Where does the `ret` instruction on Line 122 return to?

Både linje 111 og linje 122 bruker ‘ret’, altså return, og begge returneres til etter det opprinnelige kallet på ‘lessiffer’, altså til linje 44. Her blir det så sjekket om det som returneres er gyldig eller ugyldig ved å sammenligne returverdien med 0. Dersom dette stemmer, er verdien gyldig, og koden vil hoppe over linje 45. Dersom det er ‘ret’ fra linje 122 som returnerer verdien, vil denne være ugyldig, og koden vil da kjøre linje 45 som sier ‘jne Slutt’, med forklaringen om at koden vil hoppe til avslutning ved feil i innlesning. ‘jne’ står for ‘Jump if Not Equal’, noe som stemmer dersom returverdien er ugyldig. Da hopper koden til "Slutt" på linje 60, og avslutter programmet.