Task 1:

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

We use these lines to define constants:

“cr equ 13”: Defines a constant “cr” with the value 13 in ASCII, which represents a carriage return. This constant can be used throughout the program instead of using the actual numeric value 13. It's used for formatting and controlling output, such as when you want to move the cursor to the beginning of a line or start a new line of text.

“If equ 10”: defines a constant if with the value 10 in ASCII, which represents a line feed character. This constant can be used throughout the program instead of using the actual numeric value 10. LF is used to denote the end of a line of text and the beginning of a new one.

The rest of lines from 8 to 13 are lines that define different types of system calls, where, for example, we give a name “SYS_EXIT equ 1” where 1 is the command to run system call exit. So, we can use this constant by name instead of using the actual numeric value 1. In this way, we define all system calls we need, and we use what when we need it, by using the name of system call not the numeric value. As a result, we get a program which is readable and use to understand.

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

We see that “resb” is in the section BSS and The BSS segment, also known as uninitialized data, is usually adjacent to the data segment. The BSS segment contains all global variables and static variables that are initialized to zero or do not have explicit initialization in source code.

“siffer” is the name of the variable.

“resb” declare uninitialized storage space. Each takes a single operand, which is the number of bytes,

Since there are “b and 4” here refers to 4 bytes.

So, in particular, we initialized variable “siffer” without any value right now and we can use in the program later.

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

_start:

```
35        mov  edx,meldlen
36        mov  ecx,meld
37        mov  ebx,STDOUT
38        mov  eax,SYS_WRITE
39        int  80h
```

Name	Eax	Ebx	Ecx	edx
print	SYS_WRITE	STDOUT	meld	meldlen

35 move or copy the length of variable “meldlen” to edx register to define how many bytes we need to write.

36 move or copy const char of variable meld to the register ecx (self-message).

37 mov, copy or execute the system call “stander out”.

38 mov, copy or execute the system call “system write”.

39 tell the OS to execute.

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

Program will jump to line 86: lessiffer.

Which lines are included in the lessiffer block?

```
89        push  eax
90        push  ebx
```

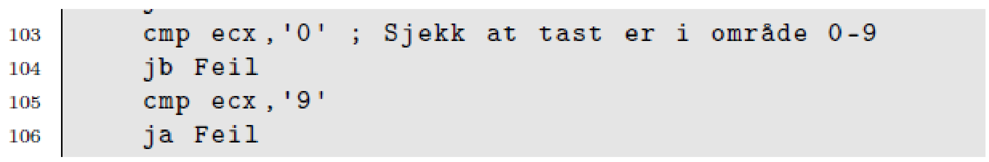
89 push eax

90 push ebx.

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

1- The ret instruction on Line 122 return to line 104 or 106 if the case is right which means the input number between [0-9] in this case the input is correct, and the program will continue from 104 or 106.

Those are lines:



```
103     cmp ecx,'0' ; Sjekk at tast er i område 0-9
104     jb Feil
105     cmp ecx,'9'
106     ja Feil
```

«cmp» is comparing if ecx is less than 48 in ASCII or greater than 57 in ASCII.

2- The ret instruction on Line 122 will not return to line 104 or 106 in this case if the input number is not between [0,9] and will continue to run in the same block and put edx to value 1 and exit