Plan 10

April 10 – April 17

• Read Section 12.6 in the textbook.

• Next course day at 1 o'clock Mads Kjærgaard from the network group of Campus-IT will give a talk about RUC´s network organization.

• **Exercise 17** Solve the exercise on the next pages.

Exercise 17 A *palindrome* is a word or sentence that can be read the same forward as it does backward. For example, ignoring the difference between upper and lower case letters, special characters as well as spaces between words, the sentence “Madam, I´m Adam” is a palindrome.

The C program on the next pages decides whether or not some sentences are palindromes.

(1) Generate x86 assembly code for the recursive function `palindrome` and comment the generated code.

(2) Generate optimized assembly code (using option –O2) for the function and indicate the differences between the non-optimized and the optimized code.

(3) Write a function in C, `my_palindrome`, which is as equivalent as possible to the generated assembly code. Use a goto statement for each jump instruction in the assembly code.

```c
#include <stdio.h>
#include <string.h>

int palindrome(char *first, char *last) {
    if (last <= first)
        return 1;
    if (isalpha(*first)) {
        if (isalpha(*last)) {
            if (toupper(*first) != toupper(*last))
                return 0;
            first++;
        }
        last--;
    } else
        first++;
    return palindrome(first, last);
}

int isPalindrome(char *s) {
    return palindrome(s, s + strlen(s) - 1);
}
```

cont’d on next page

```
int main() {
    char *sentence[4] =
        { "Madam, I'm Adam. ",
          "Nope! ",
          "Go hang a salami, I'm a lasagna hog.",
          "Was it a car or a cat I saw?" };
    int i;
    for (i = 0; i < 4; i++)
        printf("\"%s\" is%s a palindrome.\n",
               sentence[i],
               isPalindrome(sentence[i]) ? "" : " NOT");
}
```