CompArch - Assignment 2

Mads Ottendal - 202208533
Kristopher B. E. Märcher - 202205285
Mads K. T. Larsen - 202208599
Julian B. Christensen - 202108350

25th February 2024

Task 1

In this task we have to assume *n* (from the code provided in the assignment) stores a number between 0 and 9, so a single-digit number. We then have to modify the program such that it prints the value of n.

The ARM assembly code for how we do this can be seen below and our explanation of the code is further down.

```
.syntax unified
.data

digit: .ascii "x\n"
digitEnd:
n: .int 5

.text

.global _start
_start:
print:

mov r0,1                // standard output
mov r7,4                // Write System call
ldr r3,=n               // The address of int n
ldr r3, [r3]            // load the value at address r3 into r3
add r3,r3,48            // add the value at r3 with 48
                        //(since digit number n is n + 48 in ASCII)

ldr r1,=digit           // The address of string to write
strb r3, [r1]           // write single byte to memory location r1 from r3
mov r2,digitEnd-digit   // We write only two bytes, counting \n
swi 0

// Now we exit gracefully
mov r0,0
mov r7, 1
swi 0
```

What our program does is it loads the address of n into register 3. We then load into r3 the value from the address of r3, so the value of n. We then add 48 to n since we have to go from .int to .ascii. This is because the ASCII encoding of a single-digit number n is n + 48.

We then load the address of the string we want to write into r1. We then stores a byte, in this case the value of n+48, from a register to memory in r1 and that way replace the "x" with "5" in the ascii (5 is used in the code above but could be any positive single-digit number).

We then write the two bytes and exit the program.

This results in us printing a digit from 0-9.

Task 2

In this task we assume n can be a number between 0 and 99, however when a single-digit number is given (numbers 0-9) we print on, where $n \in [0,9]$. So we in some way always have a two-digit number.

We start by changing the .ascii string to be "xx\n" because of the change with 0n described above. This makes it so we can store two numbers in the .ascii instead of one.

Our solution can be found below, and the explanation to our solution below that:

```
.syntax unified
.data

digit: .ascii "xx\n"
digitEnd:
n: .int 30

.text

.global _start
_start:
print:

mov r0,1                    // standard output
mov r7,4                    // Write System call
ldr r3,=n                   // The address of int n
ldr r3, [r3]                // load the value at address r3 into r3

mov r9, 0                   // Counter for number of subtractions of 10

loop:
cmp r3 ,10                  // Compare n with 10
BPL subtract                // BPL checks if N == 0

B   continue

subtract:
SUB R3 ,R3 ,10              // Subtract 10
ADD r9, r9, 1               // Increment by 1
B loop

continue:                   // Needed for not doing subtract 1 too many times

ADD R3, R3, 48              // Gets R3 up to last digit
Add R9, R9, 48
```

```
    ldr r1,=digit              // The address of string to write
    strb r3, [r1,1]            // Adds the 1's digit
    strb r9, [r1,0]            // Adds the 10's digit
    mov r2,digitEnd-digit      // We write only two bytes, counting \n
    swi 0

    // Now we exit gracefully
    mov r0,0
    mov r7, 1
    swi 0
```

In this task, we used a loop. In the loop our conditional statement checks if the N flag is equal to 0. The N flag will be 1 if r3 - 10 is a negative number and 0 otherwise. For each iteration, we go to the subtract label, where we subtract 10 from r3 and store the result in r3. Furthermore we increment the r9 with one, to count how many times we have subtracted 10. Once the loop is done, r9 will store the number of times we have subtracted 10 from r3 and r3 will contain the last digit. Then we add 48 to both of these registers to have them contain the ascii value of the number. Lastly, we replace the 0 bit of the digit with the R9, the digit of the tens, and replace the 1 bit with the digit of the R3 which is the last digit.
We have tried this program with the following numbers and succeeded:

- 0 - to ensure it works when the 10's digit is not defined and there is no value of the last digit.

- 10 - to ensure it works when we only have a value for the 10's digit

- 05 - to ensure it works when 10's is 0 and the last digit have a value

- 99 - to ensure it works when both have a value