Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH08-320101
Date: 2018-11-15
Due: 2018-11-22

ICS 2018 Problem Sheet #9

Problem 9.1: *simple cpu machine code* (2+2 = 4 points)

The following program has been written for the simple central processing unit introduced in class. The table below shows the initial content of the 16 memory cells. The first column denotes the memory address.

#	Machine Code	Assembly Code	Description
0	001 1 0001		
1	010 0 1111		
2	001 1 0000		
3	101 1 0100		
4	110 1 0110		
5	111 1 0000		
6	001 0 0011		
7	100 1 0001		
8	010 0 0011		
9	001 0 1111		
10	011 0 1111		
11	010 0 1111		
12	110 1 0010		
13	000 0 0000		no instruction / data, initialized to 0
14	000 0 0000		no instruction / data, initialized to 0
15	000 0 0000		no instruction / data, initialized to 0

a) Explain what the instructions are doing by filling in the assembly code column. Add meaningful text to the description column to describe the action performed by an instruction.

b) Explain how the program proceeds with its calculation. Describe the control flow of the program. Which cells change and what is the purpose of the changes? What is the result left in memory cell 15 when the program stops execution?

Problem 9.2: *fold function duality theorems* (2+2+2 = 6 points)

The fold functions compute a value form a list by applying an operator to the list elements and by using a neutral element. The foldl function assumes that the operator is left associative, the foldr function assumes that the operatore is right associative. For example, the function call

```
foldl (+) 0 [3,5,2,1]
```

results in the computation of ((((0+3)+5)+2)+1) and the function call

```
foldr (+) 0 [3,5,2,1]
```

results in the computation of (3+(5+(2+(1+0)))). The value computed by the fold functions may be more complex than a simple scalar. It is very well possible to construct a new list as part of the fold. For example:

```
map' :: (a -> b) -> [a] -> [b]
map' f xs = foldr (\x acc -> f x : acc) [] xs
```

The evaluation of `map' (+3) [1,2,3]` results in the list `[4,5,6]`. There are several duality theorems that can be stated for the fold functions. Prove the following three duality theorems:

a) Let `op` be an associative operation with `e` as the neutral element:

   `op` is associative: `(x op y) op z = x op (y op z)`
   `e` is neutral element: `e op x = x` and `x op e = x`

   Then the following holds for finite lists `xs`:

   ```
   foldr op e xs = foldl op e xs
   ```

b) Let `op1` and `op2` be two operations for which

   ```
   x `op1` (y `op2` z) = (x `op1` y) `op2` z
   x `op1` e           = e `op2` x
   ```

   holds. Then the following holds for finite lists `xs`:

   ```
   foldr op1 e xs = foldl op2 e xs
   ```

c) Let `op` be an associative operation and `xs` a finite list. Then

   ```
   foldr op a xs = foldl op' a (reverse xs)
   ```

   holds with

   ```
   x op' y = y op x
   ```