Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH-232-A
Date: 2020-11-13
Due: 2020-11-20

ICS 2020 Problem Sheet #10

Problem 10.1: *assembly programming* (1+2+3+1 = 7 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 and the second column shows the memory content in hexadecimal notation.

#	Hex	Binary	Assembly Code	Description
0	2e			
1	b0			
2	d4			
3	e0			
4	2f			
5	6f			
6	4f			
7	2e			
8	91			
9	4e			
10	cb			
11	00			
12	00			
13	00			
14	06			
15	01			

a) Convert the machine code given in hexadecimal notation into binary notation.

b) Write down the assembly code for the machine code. Add meaningful descriptions.

c) The program leaves a result in memory cell 15 when it halts. What is the value? Explain how the program works, either in words or by providing an equivalent program code in a higher level imperative language.

d) What happens if the value stored in memory cell 14 is changed to 10 before execution starts? Explain.

Problem 10.2: *type classes (haskell)* (1+1+1 = 3 points)

You work at a university and your task is to write a program that will sends proper emails to students, employees, or certain roles. (A role is the position or purpose that someone has in a organization, like being the president or the provost or a dean.) You start with the following definitions:

```haskell
module Email where

data Person = Employee { eFirstName :: String, eLastName :: String }
            | Student  { sFirstName :: String, sLastName :: String }

data Role = Role { rName :: String }
```

Since you want to address the different persons and roles properly, you define the following type class:

```haskell
class Email a where
    address :: a -> String                  -- <something@example.com>
    greeting :: a -> String                 -- proper greeting phrase
    message :: a -> String -> String        -- proper Internet message
```

The `address` function returns a well formed email address, consisting of the first name followed by the last name of a person (separated by a dot). The `greeting` function returns a suitable greeting phrase. Finally, the `message` function takes a message and returns a well formed Internet message with a `To:` header line and a body that starts with a greeting followed by the message.

a) Make `Role` an instance of the type class `Email`.

b) Make `Person` an instance of the type class `Email`.

c) Define `message` as a default function in the `Email` type class.

You program should produce emails like shown in the following unit test cases. (Note that a backslash at the end of the line followed by a backslash at the beginning of the next line removes the line break.)

```haskell
{- |
   Module: emailtest.hs
-}

module Main where

import Email
import Test.HUnit

s = Student { sFirstName = "Kim", sLastName = "Smart" }
e = Employee { eFirstName = "Michael", eLastName = "Faraday" }
r = Role { rName = "President" }

nw = "Have a nice weekend"
gw = "You are doing a great job, expect to receive a bonus payment."

es = "To: <kim.smart@example.com>\n\n\
\Dear Kim,\n\n\
\Have a nice weekend"

ee = "To: <michael.faraday@example.com>\n\n\
\Dear Michael Faraday,\n\n\
\Have a nice weekend"

er = "To: <president@example.com>\n\n\
\Dear President,\n\n\
\You are doing a great job, expect to receive a bonus payment."

tl = TestList [ TestCase (message s nw @=? es)
              , TestCase (message e nw @=? ee)
              , TestCase (message r gw @=? er)
              ]

main = do
    runTestTT tl
```