Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH08-320101
Date: 2017-11-14
Due: 2017-11-21

**ICS Problem Sheet #9**

**Problem 9.1:** *simple cpu machine code* (3+2+1 = 6 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?

c) What is the result left in memory cell 15 when the program stops execution?

**Problem 9.2:** *x64 assembly language* (1+1+2 = 4 points)

In this problem, you are expected to familiarize yourself with the basic concepts of the x64 assembly language (i.e., the x64 CPU architecture). A good overview can be found online at `https://software.intel.com/en-us/articles/introduction-to-x64-assembly`.

a) What are the 64-bit registers of the x64 CPU? What is the difference between RAX, EAX, AX, AL and AH?

b) What are the common addressing modes supported by x64 CPUs? Where is the stack pointer maintained?

c) What is the result produced by the following x64 assembly program? Comment the assembler source code. (The syntax used is the GNU assembler syntax.) How is the result returned from the `main` function?

```
        .global main
        .text
main:
        xor     %rax, %rax
        mov     %rax, %rbx
.L1:
        add     $1, %rbx
        add     %rbx, %rax
        cmp     $10, %rbx
        jne     .L1
        ret
```