Rechnersysteme und -netze Winter 2017/2018

Christian Borgelt, Christoph Doell,
Heiko Dreyer, Alice Hildebrand, Josua Sattler, Simon Suckut

7. Übungsblatt

Aufgabe 1 Hack-Assembler- und -Maschinensprache

a) Übersetzen Sie den rechts stehenden Quelltext aus der Hack-Assemblersprache in die Hack-Maschinensprache!

b) Wie kann man den rechts stehenden Quelltext auf vier Anweisungen kürzen, die das gleiche Ergebnis liefern?

```
@i
M=M-1
@i
D=M
@k
M=D+M
```

Aufgabe 2 Hack-Assemblersprache

a) Was berechnet das rechts gezeigte Programm 1? Wie sieht die Symboltabelle für dieses Programm aus?
Hinweis: Hack-CPU-Emulator

b) Was ändert sich am Programmablauf und an der Symboltabelle, wenn die ersten vier Anweisungen geändert werden zu:

```
@sum
M=0
@i
M=0
```

Wie verändert sich hierdurch das Ergebnis?

c) Betrachten Sie das rechts gezeigte Programm 2. Was berechnet es? Wie unterscheidet es sich in seinen Berechnungen von Programm 1?

d) Wie könnte man das Ergebnis von Programm 1 mit weniger Anweisungen berechnen? Schaffen Sie es, mit weniger als 13 Anweisungen (Markierungsdefinitionen wie `(LOOP)` nicht gerechnet) auszukommen?
Hinweis: Berechnungsreihenfolge!
Testen Sie Ihr Programm im Hack-CPU-Emulator!

Programm 1

```
    @i
    M=1
    @sum
    M=0
(LOOP)
    @i
    D=M
    @R0
    D=D-M
    @WRITE
    D;JGT
    @i
    D=M
    @sum
    M=D+M
    @i
    M=M+1
    @LOOP
    0;JMP
(WRITE)
    @sum
    D=M
    @R1
    M=D
(END)
    @END
    0;JMP
```

Programm 2

```
    @sum
    M=0
    @i
    M=0
    @TEST
    0;JMP
(LOOP)
    @i
    D=M
    @sum
    M=D+M
    @i
    M=M+1
(TEST)
    @i
    D=M
    @R0
    D=D-M
    @LOOP
    D;JLE
    @sum
    D=M
    @R1
    M=D
(END)
    @END
    0;JMP
```

Aufgabe 3 Hardware-Beschreibungssprache HDL

Bearbeiten Sie diese Aufgabe mit dem Hardware-Simulator von der Webseite

`http://nand2tetris.org/software.php`

a) Implementieren Sie einen Halb- und einen Volladdierer in HDL!
Zur Implementierung können Sie vorgebene logische Gatter wie z.B. Nand, Nor, And, Or, Not, Xor etc. verwenden (siehe `nand2tetris/tools/builtInChips`).

b) Implementieren Sie einen 16-Bit-Addierer in HDL!

c) Implementieren Sie einen 16-Bit-Inkrementierer in HDL!

d) **Zusatzaufgabe:** Implementieren Sie die arithmetisch-logische Einheit (ALU) des Hack-Systems in HDL!
Zur Lösung dieser Zusatzaufgabe können Sie die vorgegebenen einfacheren Chips (siehe `nand2tetris/tools/builtInChips`) implizit oder explizit verwenden (z.B. `BUILTIN Mux16(...)`).