Rechnersysteme und -netze Winter 2017/2018

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

8. Übungsblatt

Aufgabe 1 Assemblersprache

a) Gegeben ist das nebenstehende Assembler-Programm in Hack-Assemblersprache, welches die Summe der Zahlen `first` $= 5$ bis `second` $= 7$ (inklusive) bildet und das Ergebnis an der Speicherstelle `res` ablegen soll. Es haben sich jedoch fünf Fehler eingeschlichen. Finden und korrigieren Sie diese Fehler!

b) Geben Sie die durch einen Assembler erzeugte Symboltabelle für das in (a) betrachtete Assembler-Programm an!

c) Nennen Sie zwei Anwendungsgebiete, in welchen Assemblersprache heutzutage noch verwendet wird! Warum wird sie in diesen Gebieten verwendet?

```
    @7
    D = A
    M = D
    @first
    @5
    D = A
    @second
    M = D
    @first
    D = M
    @res
    M = 0
(LOOP)
    @first
    D = M
    @second
    D = D+M
    @END   // if (D < 0)
    D;JLT  // goto END
    @first
    D = M
    @res
    M = D+M
    @first
    M = 1
    @LOOP
    0;JMP
(END)
    @END
    0;JMP
```

Aufgabe 2 Implementierung der Multiplikation in Software

Die Zusatzaufgabe von Aufgabenblatt 5 beschäftigte sich mit der Implementierung der Multiplikation. Dort sollte der Standard-Multiplikationsalgorithmus so angepasst werden, dass er ohne ein Rechtsschieben des Verbundregisters RA auskommt und nur die Funktionen der Hack-ALU nutzt, also im Prinzip nur Addition, Subtraktion, Einerkomplement sowie bitweise Konjunktion und Disjunktion.

Als Lösung wurde dort das folgende Java-Programm vorgestellt
(für eine Multiplikation 8 Bit $\times$ 8 Bit $\rightarrow$ 16 Bit):

```
static int multiply (int a, int b) {
    int i;                      // Laufvariable
    int r = 0;                  // Ergebnis
    int t = 1;                  // als nächstes zu testendes Bit
    for (i = 8; --i >= 0; ) {   // durchlaufe die Stellen des Faktors a
        if ((a & t) != 0)       // falls aktuelles Bit gesetzt (= 1),
            r = r + b;          // addiere (verschobenen) Faktor b zum Ergebnis
        t = t + t;              // verschiebe Testbit um eine Stelle nach links
        b = b + b;              // verschiebe Faktor b um eine Stelle nach links
    }
    return r;                   // gib das berechnete Ergebnis zurück
}
```

Übersetzen Sie nun dieses Java-Programm in die Hack-Assemblersprache, so dass es auch auf dem Hack-Prozessor ausgeführt werden kann! Prüfen Sie Ihre Implementierung mit Hilfe des Hack-CPU-Simulators!

Zusatzaufgabe Implementierung der Multiplikation in Hardware

Erstellen Sie ein Schaltnetz zur Multiplikation von zwei 4-Bit-Zahlen!
Gesucht ist also eine Gatterschaltung (ohne Takt!), die zwei Eingaben a und b mit jeweils 4 Binärstellen erhält und eine Ausgabe r mit 8 Binärstellen liefert, die das Produkt der beiden Eingaben ist!
Die 4-Bit-Zahlen a und b sollen als vorzeichenlos interpretiert werden. Eine Implementierung, die mit einer Interpretation im Zweierkomplement arbeitet, ist zwar auch möglich, aber aufwendiger, und daher hier nicht gefordert.