Übungen zur Vorlesung
GRUNDLAGEN DER INFORMATIK I
Studiengang Elektrotechnik
SS 03

AG Betriebssysteme FB3
Kirsten Berkenkötter

Ausgabe: 27.5.2003
Besprechung: 5.6.2003

Übungszettel 4

Aufgabe 1: Addition und Subtraktion im Zweierkomplement

Ihr sollt ein Programm schreiben, dass Zahlen als Short-Werte (16-Bit) einliest, in binäre Zahlen umwandelt und dann addiert und subtrahiert. Die Ergebniss sollen in Binär- und Dezimaldarstellung ausgegeben werden. Dabei soll das Ergebnis in der Dezimaldarstellung aus Kontrollgründen durch die Umwandlung des binären Ergebnisses in eine Dezimalzahl erfolgen und nicht durch die einfache Addition der eingegebenen Short-Werte!

Gegeben ist die Darstellung der Binärzahl als Bitfeld:

```
//Ein Bitfeld fuer die Darstellung einer 16-Bit großen Binaerzahl
//Das entspricht einem Wertebereich von -2^15..2^15-1
typedef struct
{
    unsigned b0 : 1;
    unsigned b1 : 1;
    unsigned b2 : 1;
    unsigned b3 : 1;
    unsigned b4 : 1;
    unsigned b5 : 1;
    unsigned b6 : 1;
    unsigned b7 : 1;

    unsigned b8 : 1;
    unsigned b9 : 1;
    unsigned b10 : 1;
    unsigned b11 : 1;
    unsigned b12 : 1;
    unsigned b13 : 1;
    unsigned b14 : 1;
    unsigned b15 : 1;
} binaerzahl;
```

a) Schreibt eine Funktion *addieren*, die zwei Zahlen in der Bitfeld-Darstellung addiert.

```
binaerzahl addieren(binaerzahl zahl1, binaerzahl zahl2);
```

Dies geschieht mit Hilfe der Funktion $+'$ aus dem Skript, d.h. die Bits und die Überträge werden einzeln Modulo addiert. Wenn der zulässige Wertebereich überschritten wird, ist das egal, die überflüssige Bit wird nicht mehr berechnet.

Für n>0 sollen die Überträge mit einer Funktion *uebertrag* berechnet werden, mit der die Funktion u_l aus dem Skript umgesetzt wird (etwas Schreib- bzw. Kopierarbeit, aber die Funktion aus dem Skript soll hier nachempfunden werden).

```
char uebertrag(char uebertrag, char zahl1, char zahl2);
```

b) Schreibt zwei Funktionen *einerkomplement* und *zweierkomplement*, mit deren Hilfe ihr das Einer- bzw. Zweierkomplement eines Bitfeldes bilden könnt.

```
binaerzahl einerkomplement(binaerzahl zahl);
binaerzahl zweierkomplement(binaerzahl zahl);
```

c) Schreibt eine Funktion *toBitfeld*, mit deren Hilfe ihr eine eingelesene Zahl in ein Bitfeld umwandeln könnt (s. auch Übungszettel 3). Beachtet, dass negative Zahlen im Zweierkomplement dargestellt werden müssen!

```
binaerzahl toBitfeld(short uebergabezahl);
```

Um etwas Schreibarbeit zu sparen, solltet ihr die Erzeugung der einzelnen Bits in eine Hilfsfunktion *toBit* auslagern, der ihr die aktuelle Umwandlungszahl für jedes Bit übergebt. Die Funktion gibt euch dann 0 oder 1 zurück.

```
char toBit(short zahl);
```

d) Schreibt eine Funktion *printBinaer*, mit deren Hilfe ihr das Bitfeld lesbar ausgeben könnt, d.h. ihr gebt alle Bits einzeln nacheinander aus.

```
void printBinaer(binaerzahl zahl);
```

Schreibt außerdem eine Funktion zum Subtrahieren von Bitfeldern.

```
binaerzahl subtrahieren(binaerzahl zahl1, binaerzahl zahl2);
```

e) Schreibt eine Funktion *toShort*, die ein Bitfeld wieder in eine Short-Zahl umwandelt. Um die Zweierpotenz 2^n zu bilden, solltet ihr eine Hilfsfunktion *zweierpotzenz* (s. Übungsblatt 3) verwenden.

```
short toShort(binaerzahl zahl);
short zweierpotenz(short n);
```

f) Schreibt ein Hauptprogramm, dass zwei Werte einliest und in Variablen speichert. Danach sollen diese Werte in Binärzahlen umgewandelt und addiert werden. Gebt diese binäre Addition samt Ergebnis auch aus (Format: zahl1 + zahl2 = ergebnis). Dann nehmt das binäre Ergebnis und wandelt es wieder in eine Dezimalzahl um. Gebt die Addition nochmal samt Ergebnis aus, diesmal dezimal.

Das gleiche (berechnen - ausgeben binär - umwandeln in dezimal - ausgeben dezimal) soll mit der Subtraktion zweier Zahlen gemacht werden.