Freie Universität Berlin

Übung 3

Rechnerarchitektur
Wintersemester 17/18
Prof. Dr.-Ing. Jochen Schiller

Ausgabe 03.11.2017
Abgabe 17.11.2017, 10:15 (s.t.)

Bitte beachten Sie die allgemeinen Hinweise auf Übungszettel 1

Aufgabe 1: Zahlenbasen

(a) Rechnen Sie die folgenden Zahlen vom Dezimalsystem ins Binär- und Hexadezimalsystem um:

- 113_{10}
- $257,23_{10}$

(b) Rechnen Sie die folgenden Zahlen vom Binärsystem ins Dezimalsystem um:

- 101011010_2
- $10010,1001_2$

(c) Rechnen Sie die folgenden Zahlen vom Hexadezimalsystem ins Dezimalsystem um:

- $0xE37A$
- $0x39B,2D8$

(d) Rechnen Sie folgende Binärzahl ohne Umweg über das Dezimalsystem direkt ins Hexadezimalsystem um.

- 0101101010110010111_2

Der Lösungsweg soll stets erkennbar sein.

Aufgabe 2: Fibonacci Zahlen

Auf dem letzten Zettel haben Sie die Wiederholung von Instruktionen durch Sprünge kennengelernt. Diese Art der Wiederholung wird *Iteration* genannt. Die andere Art der Wiederholung ist *Rekursion*. Machen Sie sich mit Rekursion auf Assemblerebene und dafür mit dem Callstack vertraut (Befehle: PUSH, POP, CALL, RET).

Die Fibonacci-Zahlen seien wie folgt definiert:

```
f(1) = 1;
f(2) = 1;
f(n) = f(n-1) + f(n-2); // für n>2
```

Im KVV wird Ihnen ein C-Framework gestellt, welches unterschiedlich implementierte Fibonacci-Funktionen gegeneinander vergleicht. Schreiben Sie in Assembler eine iterative und eine rekursive Fibonacci-Funktion (siehe Pseudocode unten). Linken Sie die Funktionen mit dem C-Framework. Ihre Funktionen müssen dafür folgende Signaturen haben:

```
uint64_t asm_fib_it(uint64_t n);
uint64_t asm_fib_rek(uint64_t n);
```

Erklären Sie die Zeitunterschiede sowohl zwischen den rekursiven und iterativen Funktionen als auch zwischen den C und Assembler Funktionen. Warum wird Rekursion überhaupt benötigt?

- Pseudocode iterativ:

```
fib_it(n) {
    x = 0;
    y = 1;
    k = 0;
    while(n > 0) {
        x = y;
        y = k;
        k = x + y;
        n--;
    }
    return k;
}
```

- Pseudocode rekursiv:

```
fib_rek(n) {
    if(n < 3) {
        return 1;
    } else {
        return fib_rek(n-1) + fib_rek(n-2);
    }
}
```

Hinweis: Für das Erklären der Zeitunterschiede sollten Sie sich die Kompilate der C-Funktionen ansehen. Möglichkeiten um sich diese anzusehen sind "gdb <prog>", "objdump -d <prog>" oder "gcc -S fib.c".