3. Aufgabenblatt

Ausgabe	Abgabe
06.05.11	20.05.11

Hinweis zu Programmieraufgaben: Bitte machen Sie zu jeder Programmieraufgabe Testläufe, die die Funktionialität ihrer Programme ausreichend dokumentieren. Die Testläufe und den Quellcode drucken Sie bitte aus, und geben beides mit der jeweiligen Übung auf Papier ab. Der Quellcode muss zusätzlich in einer kompilierbaren .mms Datei per Mail an den jeweiligen Tutor geschickt werden. Als Betreff der Email geben Sie bitte Ihre Matrikelnummer und die Nummer des Aufgabenblattes an.
Die bewertete Aufgabe ist als solche gekennzeichnet!

Aufgabe 1: Zahlensysteme

a) Rechnen Sie die Zahlen 1234, 23.77, 256 vom Dezimal ins Hexadezimalsystem um.
b) Rechnen Sie die Zahlen 0x77, 0x80, 0x66.01 ins Dezimalsystem um

Ausgabe 2: GNU Debugger

Setzen Sie sich mit dem gdb (http://www.gnu.org/software/gdb/) auseinander. Lassen Sie sich dabei während der Ausführung eines Programms die Register und den Stack anzeigen. Dazu müssen sie ihr Programm mit dem "-g" Flag übersetzen: "gcc -g -m32 ...". Starten sie dann den gdb mit "gdb a.out", wobei a.out ihr übersetzets Programm ist. Geben sie "start" ein. Jetzt können Sie mit dem "step" Kommando Ihr Programm Zeile für Zeile ausführen. Probieren sie die Kommandos "info register" und "x/16xw $sp" aus. Recherchieren sie die Bedeutung dieser Kommandos.

Aufgabe 3: Einfacher arithmetischer Ausdruck

Übersetzen Sie die folgenden Anweisungen von Pseudo-Code in 32-bit x86-Assembler. Es ist ihnen freigestellt, ob sie Intel- oder AT&T-Syntax verwenden, aber bitte mischen Sie nicht beide!

```
function f(a, b)
    return <- 7 + a * 2 + b * 30
```

Beachten Sie Punkt vor Strich!

Hinweis: Verwenden Sie “add” und “imul”.

Aufgabe 4: Maximum

Übersetzen Sie die folgenden Anweisungen von Pseudo-Code in 32-bit x86-Assembler. Es ist ihnen freigestellt, ob sie Intel- oder AT&T-Syntax verwenden, aber bitte mischen Sie nicht beide!

```
function max(a, b)
    if a > b
        return a
    else
        return b
```

Aufgabe 5: Fibonacci-Zahlen (bewertete Aufgabe)

Die Fibonacci-Zahlen seien wie folgt definiert:
fib(0) = 0, fib(1) = 1, fib(n > 1) = fib(n-2) + fib(n-1).

Eine iterative Möglichkeit die n-te Fibonacci-Zahl zu berechnen ist, dass man sich drei Laufvariablen x1, x2 und k wählt, die man mit 0, 1 und 0 initialisiert. Am Ende jedes Schleifendurchlaufes soll gelten,
* dass der neue Wert von x1 gleich dem alten Wert von x2 ist,
* der neue Wert von x2 gleich dem alten Wert von k ist.
* und k = x1 + x2 ist,
Die Schleife soll n mal wiederholt werden (für n = 0 also kein mal, für n = 1 einmal, usw. ausgeführt werden).

(i) Überlegen sie sich zunächst eine Implementierung in Pseudo-Code.
(ii) Implementieren sie ihre Lösung in C.
(iii) Übersetzen sie ihre Lösung in 32-bit x86-Assembler. Es ist ihnen freigestellt, ob sie Intel- oder AT&T-Syntax verwenden, aber bitte mischen Sie nicht beide!
(iv) Freiwillige Zusatzaufgabe: Vergleichen sie mit Hilfe des Benchmarks ihre Lösung in C und ihre handgeschriebene Assembler-Lösung!