4. Aufgabenblatt

Ausgabe	Abgabe
17.05.11	27.05.11

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 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!

1. Rechenaufgaben (bewertete Aufgabe)

Führen Sie die folgenden Berechnungen mittels B+V-Darstellung, im Einer- und Zweierkomplement durch (nehmen Sie 8 bit breite Register an). Falls Sie eine Berechnung in einer Darstellungsform nicht durchführen können, begründen Sie, warum! Was würde im Prozessor geschehen?

- (a) 21 + 22
- (b) 120 + 10
- (c) 250 + 1
- (d) 70 - 30
- (e) 72 - 87
- (f) 3 * 10

2. Assembler-Befehle

Machen Sie sich mit den grundlegenden arithmetischen Befehlen (ADD, SUB, SHL, SHR, INC, DEC, IMUL, IDIV) und Anweisungen zur Steuerung des Kontrollflusses vertraut (JMP, LOOP, CMP, JE, JNE, JZ, JNZ, JG, JL, JGE, JLE). Welche Register und welche FLAGS-Bits werden von welchen Befehlen beeinflusst bzw. berücksichtigt?

Beachten Sie besonders:

- Wie kann man ganzzahlige Divisionen in x86 vornehmen?
- Wie erhält man das ganzzahlige Ergebnis und den Rest?

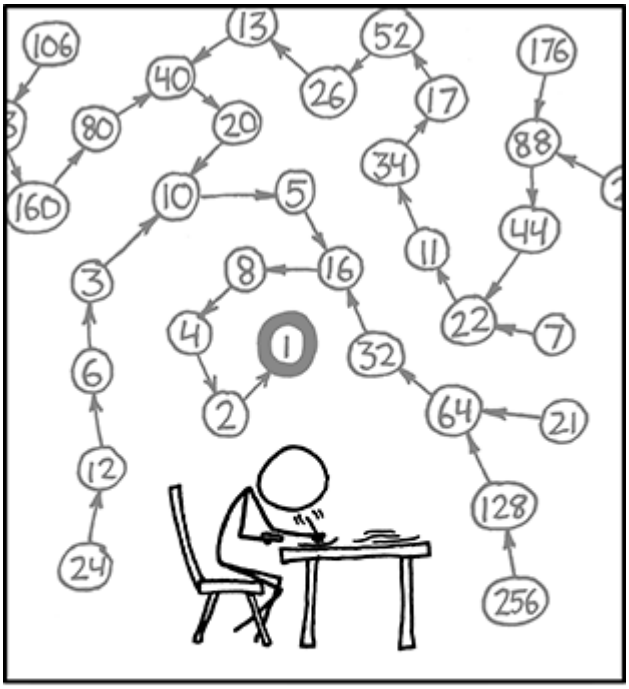
Recherchieren Sie außerdem die Anweisungen PUSH, POP, ENTER, LEAVE, RET, CALL.

3. Über 32-bit hinaus

Schauen Sie sich den Befehl ADC (add with carry) und die Befehle zum Umgang mit dem Carry-Flag (CLC, STC) an. Wie könnten Sie unter zu Hilfe Nahme dieser Funktionalität ein 128-bit breites Integer implementieren (berücksichtigen Sie nur die Addition)?

4. Collatz Conjecture (bewertete Aufgabe)

Gegeben sei das folgende Stück Pseudo-Code:



THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

```
function collatz(k)
    i <- 0
    while k > 1
        i <- i + 1
        if is-even(k)
            k <- k * 2
        else
            k <- k * 3
            k <- k + 1
    return i
```

Übersetzen Sie es mit Hilfe des gegebenen Frameworks in ein 32-bit x86-Assembler-Programm!

Hinweise:
- is-even(x) ist äquivalent zu “x modulo 2 ist gleich 0”

Anhang: collatz.S

```
.text

.globl collatz
.globl _collatz

collatz:
_collatz:
    push %ebp
    movl %esp, %ebp

    # Implementieren Sie Ihre Funktion hier

    pop %ebp
    ret
```

Anhang: collatz.c

```
#include "stdio.h"
```

```
__attribute__((fastcall)) extern int collatz(int a); // fastcall is needed for similar calling conventions between Windows and Unix

int collatz_in_c(int acc) {
	int i = 0;
	while (acc > 1) {
		i++;
		if (acc % 2 == 0) {
			acc /= 2;
		} else {
			acc *= 3;
			acc++;
		}
	}
	return i;
}

int main () {
	int i, c;
	for (i = 1; i <= 15; i++) {
		c = collatz(i);
		printf("%i\t%i\t%s\n", i, c, collatz_in_c(i) == c ? " - korrekt!" : " - falsch.");
	}


	return 0;
}
```