Hochschule Augsburg
Prof. Dr. Hubert Högl
E-mail: Hubert.Hoegl@hs-augsburg.de
WWW: http://hhoegl.informatik.hs-augsburg.de/hhwiki/SysProg

Übungen in Systemnaher Programmierung

Übungsblatt 4

Aufgabe 1

(aus einer Klausur)

Wozu braucht man einen Stack? Über welche Befehle spricht man den Stack an?

Aufgabe 2

(aus einer Klausur)

Sehen Sie sich folgenden Assembler-Quelltext an. Die drei Punkte ... stehen für beliebigen Code, der uns nicht interessiert. Beantworten Sie bitte folgende Fragen:

- Beschreiben Sie, was an den Stellen (1) bis (7) gemacht wird.
- Zeichnen Sie den Stack direkt nach der Ausführung von Zeile 5. Zeichnen Sie auch Framepointer und Stackpointer ein.
- Wie greift man innerhalb der Funktion `tuwas()` auf die lokalen Daten zu? Nehmen Sie an, dass die 8 Byte aus zwei Integer Werten bestehen. Schreiben Sie die Framepointer-relative Adressierung für den Integer mit der kleineren Adresse hin.

```
...`
pushl $2                # (1)
pushl $4                # (1)
call tuwas              # (2)
addl $8, %esp           # (3)
...

tuwas:
  pushl %ebp            # (4)
  movl %esp, %ebp       # (4)
  subl $8, %esp         # (5)
  ...
  movl %ebp, %esp       # (6)
  popl %ebp             # (6)
  ret                   # (7)
```

Aufgabe 3

(aus einer Klausur)

Hier sind einige Fragen zur C Aufrufkonvention:

- In welcher Reihenfolge werden die Argumente der Funktion `cfun(int a, int b, int c)` auf dem Stack abgelegt?

- Wie wird der Rückgabewert einer Funktion an den Aufrufer übergeben? Unterscheiden Sie: (a) der Wert ist 32-Bit gross, (b) der Wert ist grösser als 32-Bit.
- Wer kümmert sich um die Sicherung der Register -- der Aufrufer oder der Aufgerufene?
- Wer korrigiert den Stack, der Aufrufer oder der Aufgerufene?

Aufgabe 4

Analysieren Sie die Funktionsweise des Beispiels "power" aus dem Kapitel 4 ("All About Functions") mit dem GNU Debugger, so wie wir das in der Vorlesung gemacht haben.

Aufgabe 5

(aus "Use the Concepts" am Ende von Kapitel 4)

1. Schreiben Sie eine Funktion `quadrat(x)`, die aus dem Argument `x` das Quadrat `x * x` berechnet. Rufen Sie diese Funktion zum Test auch ein paar Mal mit unterschiedlichem Argument auf.
2. Schreiben Sie die Maximumsuche aus dem Kapitel 3 nun als Funktion `maximum(ptr)`. Der Zeiger `ptr` zeigt auf eine Liste von Elementen, deren grösster Wert zurückgegeben wird. Rufen Sie diese Funktion zum Test ein paar Mal mit unterschiedlichen Listen auf.