Hochschule Augsburg Sommersemester 2011
Fakultät für Informatik

Klausur

Prüfungsfach:	Systemnahe Programmierung
Datum/Uhrzeit:	11. Juli 2011 / 12:30 Uhr
Raum:	wie angekündigt
Prüfer:	Dr. Hubert Högl
Dauer:	**60** Minuten
Hilfsmittel:	keine

Hinweise:

1. Dieses Angabenblatt umfasst 5 Seiten.
2. Schreiben Sie bitte nicht auf das Angabenblatt. Verwenden Sie für Ihre Antworten die separat ausgeteilten Bögen. Die Angaben dürfen Sie behalten.
3. Schreiben Sie nicht mit Bleistift.

Aufgabe 1 (4 Punkte)

Worin unterscheiden sich Programmiersprachen die sich zur maschinennahen (= systemnahen) Programmierung eignen von Sprachen, die man „high-level“ Sprachen nennt? Nennen Sie jeweils zwei Sprachen aus jeder Gattung.

Aufgabe 2 (4 Punkte)

Beschreiben Sie die Register der x86 CPU.

Aufgabe 3 (3 Punkte)

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

Aufgabe 4 (13 Punkte)

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

1. Beschreiben Sie, was an den Stellen (1) bis (7) gemacht wird. Bitte keine trivialen Bemerkungen, sondern die übergeordnete Absicht des Codes deutlich machen.
2. Zeichnen Sie den Stack direkt nach der Ausführung von Zeile 5. Zeichnen Sie auch Framepointer und Stackpointer ein.
3. Wie greift man innerhalb der Funktion `tuwas()` auf die lokalen Daten zu? Nehmen Sie an, dass die 8 Byte aus zwei Integer (long) 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 %esp, 8            # (5)
...
movl %ebp, %esp         # (6)
popl %ebp               # (6)
ret                     # (7)
```

Aufgabe 5 (4 Punkte)

Hier sind einige Frage zur C Aufrufkonvention:

1. In welcher Reihenfolge werden die Argumente der Funktion `cfun(int a, int b, int c)` auf dem Stack abgelegt?
2. 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.
3. Wer kümmert sich um die Sicherung der Register – der Aufrufer oder der Aufgerufene?
4. Wer korrigiert den Stack, der Aufrufer oder der Aufgerufene?

Aufgabe 6 (4 Punkte)

Wenn man ein Programm aufruft, dann wird es zunächst in den Speicher geladen. In welche Abschnitte ist das Programm im Speicher gegliedert und wozu dienen diese Abschnitte? Zeichnen Sie auch ein Bild des gesamten Linux Programmes beim Start (inklusive Programmname, Argumenten, Umgebungsvariablen und Heap).

Aufgabe 7 (4 Punkte)

Beschreiben Sie den Zusammenhang zwischen virtuellem und physikalischem Speicher auf Ihrem Rechner. Betrachten Sie zwei Programme, die zur gleichen Zeit ausgeführt werden.

Aufgabe 8 (4 Punkte)

Beantworten Sie bitte folgende Fragen zu den Systemaufrufen:

1. Ist ein Systemaufruf ein Funktionsaufruf oder ein Interrupt?
2. Wie wird ein bestimmter Systemaufruf ausgewählt?
3. Wie werden die Parameter übergeben?
4. Schreiben Sie einen Systemaufruf Ihrer Wahl hin (mit Parameterübergabe).

Aufgabe 9 (4 Punkte)

Im Buch von Bartlett wird im Kapitel 9 die Funktionsweise des Heap erklärt. Beantworten Sie dazu folgende Fragen:

1. Wie heissen die wesentlichen Funktionsaufrufe zur Verwendung des Heap?
2. Was verstehen Sie unter *Unmapped Memory*?
3. Wie kann der Heap-Speicherbereich vergrössert werden?

4. In welchen Datenstrukturen werden die vom Heap angeforderten Speicherblöcke verwaltet? Verwenden Sie ein Diagramm zur Erläuterung.

Aufgabe 10 (4 Punkte)

Übertragen Sie die folgende C Funktion nach Assembler.

```
int vergleich(int a, int b)
{
    if (a == b) {
        return 1;
    }
    else {
        return 0;
    }
}
```

Aufgabe 11 (4 Punkte)

Sehen Sie sich die folgende GDB Sitzung an:

```
GNU gdb 6.8-debian

(gdb) file main
Reading symbols from /home/hhoegl/prog-3-1/main...done.
(gdb) list
1   .section .text
2   .globl _start
3
4   _start:
5       movl $1, %eax
6       movl $0, %ebx
7   b1:
8       int $0x80

(gdb) br b1
Breakpoint 1 at 0x804805e: file main.s, line 8.

(gdb) run ene mene miste
Starting program: /home/hhoegl/prog-3-1/main ene mene miste

Breakpoint 1, b1 () at main.s:8
8       int $0x80
Current language:  auto; currently asm

(gdb) x/5xw $esp
0xbfacd0c0: 0x00000004  0xbfacd748  0xbfacd763  0xbfacd767 0xbfacd76c

(gdb) x/s 0xbfacd748
0xbfacd748:  "/home/hhoegl/prog-3-1/main"
...
```

1. Kommentieren Sie die einzelnen gdb Ein- und Ausgaben.

2. Was sehen Sie, wenn Sie die weiteren Werte 0xbfacd763, 0xbfacd767 und 0xbfacd76c als Argumente an `x/s` übergeben?

Ende der Klausur