Hochschule Augsburg
Fakultät für Informatik

Sommersemester 2009

Klausur

Prüfungsfach:	Maschinennahe Programmierung (Diplom)
Datum/Uhrzeit:	8. Juli 2009 / 13:30 Uhr
Raum:	wie angekündigt
Prüfer:	Dr. Hubert Högl
Dauer:	**90** Minuten
Hilfsmittel:	keine

Hinweise:

1. Dieses Klausurangabenblatt hat auch eine **Rückseite**! Bitte sofort überprüfen.
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.

Viel Glück!

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)

Welche elementaren Datentypen kennt man in der Assemblerprogrammierung? Auf welche Weise könnte man die folgenden Daten interpretieren?

```
2B332C30
```

Aufgabe 3 (4 Punkte)

Wie funktioniert die folgende Adressierungsart? Welche Adresse wird mit welcher Breite angesprochen, wenn man `data_items` mit `0x1000` und `edi` mit 5 annimmt.

```
movl    data_items(, %edi, 4), %eax
```

Um welchen weiteren Freiheitsgrad könnte man diese Adressierungsart erweitern?

Aufgabe 4 (4 Punkte)

Beschreiben Sie die Register der x86 CPU.

Aufgabe 5 (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 (mit Argumenten und Umgebungsvariablen).

Aufgabe 6 (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 7 (4 Punkte)

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

Aufgabe 8 (2 Punkte)

Was passiert bei einem Linux Systemaufruf? Schreiben Sie einen Systemaufruf Ihrer Wahl in Assembler hin, der mindestens einen Parameter hat.

Aufgabe 9 (4 Punkte)

Schreiben Sie die folgende C Funktion in Assembler. Geben Sie auch an, wie man die Funktion in Assembler aufruft und was nach ihrer Rückkehr passiert.

```
int addiere(int p1, int p2)
{
    return p1 + p2;
}
```

Aufgabe 10 (4 Punkte)

Was versteht man unter robusten Programmen?

Aufgabe 11 (4 Punkte)

Sie haben zwei C Module main.c und lib.c.

a) Wie übersetzt man das Programm so dass alles statisch gelinkt wird?

b) Wie übersetzt man das Programm so dass f() erst zur Laufzeit gelinkt wird?

```
main.c:
        main() { f(); ... }

lib.c:
        f() { ... }
```

Aufgabe 12 (4 Punkte)

Übertragen Sie die folgende if/else Kontrollstruktur in Assembler:

```
if (a == b) {
   /* wahr */
}
else {
   /* falsch */
}
/* weiter */
```

Aufgabe 13 (4 Punkte)

Erläutern Sie die Funktionsweise der dynamischen Speicherverwaltung an Hand der Funktionen `allocate()` und `deallocate()`, die im Kapitel 9 im Bartlett vorgestellt werden.

Aufgabe 14 (8 Punkte)

Das folgende Programm schreibt in die Speichertabelle am Ende des Angabenblattes. Nehmen Sie dazu an, dass das Array `tabelle` mit der Speichertabelle identisch ist.

```
    .section .bss
    .lcomm tabelle, 32

    .section .text
    .globl _start
_start:
    movl $tabelle, %eax
    movb $0xfe, (%eax)

    movw $0x1111, 2(%eax)
    rolw $1, 2(%eax)

    xor  %ebx, %ebx
    mov  $4, %ecx
    mov  $-1, %edx
l1: movl %edx, 4(%eax, %ebx, 8)
    incl %ebx
    subl $1, %edx
    loop l1

    # exit() with return value 0
exit:
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

(a) Welche Werte stehen nach dem Ablauf in der Speichertabelle?

(b) Schreiben Sie die gdb Kommandos hin für: