Hochschule Augsburg
Fakultät für Informatik

Wintersemester 2008/2009

Klausur

Prüfungsfach:	Systemnahe Programmierung (Bachelor)
Datum/Uhrzeit:	30. Januar 2009 / 10:30 Uhr
Raum:	wie angekündigt
Prüfer:	Dr. Hubert Högl
Dauer:	**60** Minuten Minuten
Hilfsmittel:	keine

Hinweise:

1. Schreiben Sie bitte nicht auf das Angabenblatt. Verwenden Sie für Ihre Antworten die separat ausgeteilten Bögen. Die Angaben dürfen Sie behalten.
2. 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)

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)

Was machen folgende GDB Kommandos?

```
a)  (gdb) list main
b)  (gdb) br 15
c)  (gdb) p/x $edi
d)  (gdb) x/4xw $esp+8
```

Ende der Klausur