Hochschule Augsburg
Fakultät für Informatik

Wintersemester 2016/2017

Prüfung

Prüfungsfach:	Systemnahe Programmierung
Datum/Uhrzeit:	26. Januar 2017 / 12:30 Uhr
Raum:	M1.01, J2.18
Prüfer:	Dr. Hubert Högl
Dauer:	**60** Minuten
Hilfsmittel:	keine

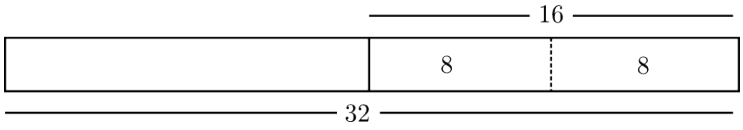
Hinweise:

1. Dieses Angabenblatt 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!

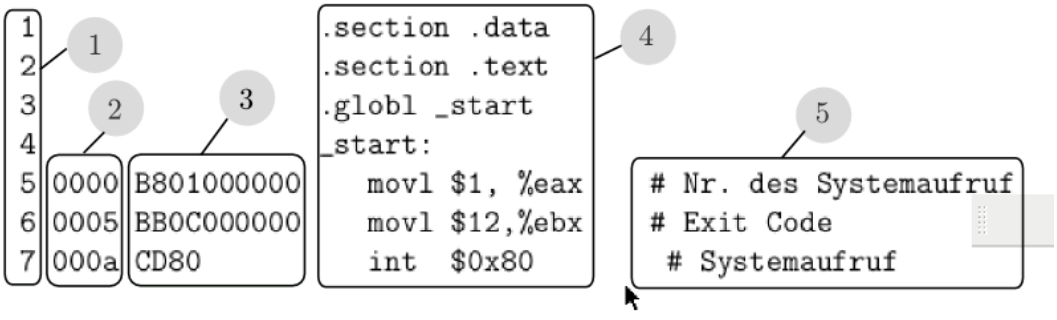
Aufgabe 1 (8 Punkte)

Schreiben Sie die wichtigsten **vier allgemeinen Register** und **vier speziellen Register** des x86 hin. Benennen Sie jeweils wie in der folgenden Abbildung das gesamte Register, den unteren 16-Bit Teil und die beiden unteren 8-Bit Teile (bitte auf den Antwortbogen übertragen).



Aufgabe 2 (10 Punkte)

Erläutern Sie kurz die **fünf Bestandteile** 1 bis 5 in der folgenden Abbildung eines **Assembler Listings**:



Aufgabe 3 (6 Punkte)

Welche **drei Möglichkeiten** gibt es, um in Assembler den **Array Datentyp** zu realisieren? Ein Array sei eine Folge einer bestimmten Anzahl von Elementen mit gleichem Typ. Verdeutlichen Sie jede Variante mit einer kleinen Skizze. Schreiben Sie für jede Variante den **Assembler-Code für die Iteration** über alle Elemente.

```
        # Code noch unvollstaendig!
array:  .long   3, 9, 4, 8
```

Aufgabe 4 (4 Punkte)

Fragen zur Endianness

(a) Was bedeutet die „Endianness“ eines Rechners?

(b) Wie kann man herausfinden, welche Endianness ein Rechner hat? Beschreiben Sie das verwendete Prinzip und geben Sie auch die nötigen Assembler-Befehle an.

Punkte: (a) 2, (b) 2

Aufgabe 5 (4 Punkte)

Obwohl Sie den Systemaufruf `poll()` wahrscheinlich nicht kennen, können Sie seinen Aufruf in Assembler sicher skizzieren. Dieser Aufruf hat die Nummer 168.

```
int poll(struct pollfd *fds, nfds_t nfds, int timeout);
```

In Ihrer Lösung nennen Sie die Parameter einfach `fds`, `nfds` und `timeout`. Wohin müssen diese Parameter übergeben werden?

Aufgabe 6 (18 Punkte)

Punkte: (a) 4, (b) 2, (c) 2, (d) 2, (e) 2, (f) 4, (g) 2

Das folgende Programm `main.s` kann mit einem Kommandozeilenargument aufgerufen werden, z.B. so: `main 5`.

```
 1      .section .data
 2      .section .text
 3      .globl _start
 4
 5  _start:
 6      movl    8(%esp), %edx
 7      movb    (%edx), %al
 8      subb    $0x30, %al
 9
10      movl    $0, %ecx
11      movl    $1, %ebx
12  L1: decb    %al
13      addl    %ebx, %ecx
14      addl    $1, %ebx
15      cmpb    $0, %al
16      jne     L1
17
18      movl    %ecx, %ebx
19      movl    $1, %eax
20      int     $0x80
```

(a) Was macht das Programm? Kommentieren Sie die drei Abschnitte 6 - 8, 10 - 16 und 18 - 20. Bitte keine trivialen Kommentare verwenden!

(b) Was wird ziemlich sicher passieren, wenn man das Programm ohne Kommandozeilenargument aufruft?

(c) Wie kann man den Fall (b) abfangen, d.h. wie kann man das Programm robuster machen? Beschreiben Sie knapp die Lösung, es ist kein Quelltext erforderlich.

(d) Was wird passieren, wenn man das Programm mit einer mehrstelligen Zahl aufruft, z.B. `main 12`?

(e) Was muss man tun, damit der Fall (d) funktioniert (kurze Beschreibung ohne Quelltext).

(f) Machen Sie aus dem Abschnitt 4 – 10 eine Funktion `f1()` mit Prolog und Epilog, die man so aufruft:

```
s = f1(int n);
```

Das Argument `n` ist die Zahl von der Kommandozeile, der Rückgabewert ist eine ganze Zahl in `eax`.

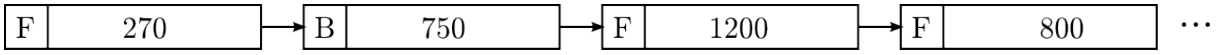
(g) Statt das Ergebnis über den Exitcode zu erhalten, möchten Sie die Funktion `printf()` aus der C-Bibliothek aufrufen. Schreiben Sie den nötigen Assembler-Quelltext hin.

Aufgabe 7 (8 Punkte)

Geben Sie die Vor- und Nachteile von statischem und dynamischem Linken wieder.

Aufgabe 8 (10 Punkte)

Die folgende Abbildung zeigt eine **Kette aus freien (F) und belegten (B) Blöcken auf dem Heap**. Die Zahl gibt die Grösse des Blockes an.



a) **Wo liegt der Heap-Speicher** im Speicher eines Prozesses?

b) **Welcher Block** wird bei einem `allocate(600)` Aufruf nach dem Algorithmus aus dem Buch von Bartlett belegt?

c) **Welche Nachteile** hat dieser einfache Algorithmus?

d) Wie kann der einfache Algorithmus **verbessert** werden?

e) Wie kann der gesamte, dem Heap zur Verfügung stehende Speicher **vergrössert** werden?

Aufgabe 9 (13 Punkte)

Hier geht es um Exploits

Punkte: (a) 4, (b) 4, (c) 2, (d) 1, (e) 2

(a) Was bedeuten die Begriffe **Opfer**, **Einbrecher**, **Injektionscode** und **Shellcode**?

(b) Erläutern Sie die einzelnen nötigen Schritte, damit ein Einbrecher in dem Beispiel aus der Vorlesung/Übung zu einer Shell mit Admin-Rechten kommen kann.

(c) Warum ist der folgende naive Shellcode falsch?

```
naive_shellcode:
        mov  ebx, <filename_addr> ; filename
        mov  ecx, <argv_addr>     ; 0x804971c   ; argv
        mov  edx, 0x0             ; envp
        mov  eax, 0xb             ; code for execve system call
        int  0x80
```

(d) Wie welchem GDB-Kommando kann man eine Folge von Maschinencode-Bytes wieder in Assembleranweisungen verwandeln?

(e) Nennen Sie **zwei** Massnahmen, die diese Art Einbrüche heutzutage verhindern.