Freie Universität Berlin

Übung 1

Rechnerarchitektur
Wintersemester 18/19
Prof. Dr.-Ing. Jochen Schiller

Ausgabe 19.10.2018
Abgabe 02.11.2018, 10:15 (s.t.)

Aufgabe 0: KVV & allgemeine Hinweise

1) Melden Sie sich bitte im KVV[1] unter *Section Info* zu einer Übungsgruppe an.

2) Lesen Sie bitte folgende Hinweise aufmerksam durch (gelten für alle Übungszettel):

Bearbeitung der Aufgaben:

- Beantworten Sie alle Aufgaben so verständlich und leserlich wie möglich, mit Ihren eigenen Worten.
- Abgaben sind in Englisch und Deutsch möglich.
- Bei Quellen jenseits der Vorlesungsfolien, sind diese anzugeben.
- Quellcode ist ausreichend zu kommentieren.
- Ihrem Programm sollten Testläufe (Beispielein-/ausgaben) beiliegen.
- Programmieraufgaben sollen im Linuxpool kompilierbar sein.

Abgabe des Zettels:

- Bitte geben Sie Ihre Lösung auf Papier im Fach Ihres Tutors oder als PDF im KVV ab.
- Quellcode reichen Sie bitte unkomprimiert im KVV ein.
- Abgaben sind bis zum Beginn der Vorlesung jeweils Freitags (10:15 Uhr) möglich.

Bewertung eines Zettels:

- Es gibt immer zwei (bewertete) Aufgaben.
- Jede der beiden Aufgaben wird mit 3 Punkten bewertet.
- Zum Bestehen des Zettels muss in jeder Aufgabe mindestens 1 Punkt erreicht werden, und
- mindestens 3 Punkte auf den gesamten Zettel.
- Die Punkte sind wie folgt pro Aufgabe definiert:

  3 Punkte:
  - Alles perfekt.
  - Klausur kann kommen.

  2 Punkte:
  - Es funktioniert / ist im Wesentlichen korrekt.
  - Kleinere Mängel sind mit Kommentaren versehen.

  1 Punkt:
  - Es funktioniert nicht, aber richtige Idee mit Fehlerbeschreibung (!) vorhanden.
  - Bzw. grobe Fehler enthalten oder unzureichend beschrieben.
  - Dennoch erkennbarer Aufwand.

  0 Punkte:
  - Aufgabe nicht bearbeitet.
  - Oder kein Ansatz für unabhängige Teilaufgaben vorhanden.
  - Oder sinnvoller Arbeitsaufwand nicht erkennbar.
  - Damit Zettel leider nicht bestanden.

[1] `https://kvv.imp.fu-berlin.de/portal/directtool/c6480563-9548-437d-aa6a-d478dc65144f/`

Aufgabe 1: Das Von-Neumann-Rechnermodell

1946 wurde das von-Neumann-Rechnermodell vorgestellt, das die Rechnerarchitektur bis heute maßgeblich beeinflusst. Arbeiten Sie die grundlegenden Organisationsprinzipien und Besonderheiten dieses Modells heraus, indem Sie folgende Fragen möglichst prägnant und in eigenen Worten beantworten.

- Mit dem von-Neumann-Rechnermodell wurde erstmalig das Konzept für einen echten „general-purpose Computer“ vorgeschlagen. Was ist darunter zu verstehen?
- „Programme sind auch nur Daten“ ist eine grundlegende und eng mit dem von-Neumann-Rechnermodell verbundene Sichtweise. Was ist darunter zu verstehen?
- Das von-Neumann-Rechnermodell setzt sich aus drei Hauptbestandteilen zusammen. Welche Bestandteile sind dies und welchem Zweck dienen sie?
- Im von-Neumann-Rechnermodell ist der Datenprozessor ein Bestandteil der CPU. Welche Aufgaben werden von welchen Komponenten dieses Prozessors erfüllt?
- Im von-Neumann-Rechnermodell ist der Befehlsprozessor ein Bestandteil der CPU. Welche Aufgaben werden von welchen Komponenten dieses Prozessors erfüllt?
- Das von-Neumann-Rechnermodell unterscheidet zwischen Daten- und Adressbus. Warum macht das Sinn? Es ergeben sich auch Zusammenhänge zwischen der Größe (in Bits) des MAR, des MBR, des Speichers, einer Speicherzelle sowie der Speicherzellenanzahl. Welche?
- Die Arbeitsweise eines von-Neumann-Rechners wird durch die Bezeichnung SISD allgemein charakterisiert. Welches Prinzip verbirgt sich hinter dieser Abkürzung?
- Bahnbrechend neu am von-Neumann-Rechnermodell war das Konzept einer quasi universellen Programmierbarkeit. Erörtern Sie in diesem Zusammenhang die Begriffe Maschinencode, Assemblersprachen sowie Ein- und Mehr-Adress-Befehle.
- Charakteristisch für das von-Neumann-Rechnermodell ist ein Zwei-Phasen-Konzept der Befehlsverarbeitung. Welches Problem wird damit auf welche Weise gelöst?
- Die Architektur eines klassischen von-Neumann-Rechners führte schon bald zu einem gewichtigen Problem, dem von-Neumannschen „Flaschenhals“. Was ist darunter zu verstehen und wie versuchte man später dieses Problem zunächst zu umgehen?

Aufgabe 2: Assembler

NASM

Die Programmieraufgaben in diesem Kurs sollen mit dem 64bit-"Netwide Assembler- kurz NASM – gelöst werden. NASM ist x86 Assembler der Intelsyntax verwendet. Wenn Ihnen also ein x86-PC mit einem 64bit Linux zur Verfügung steht, können Sie sich die NASM-Toolchain auf diesem installieren, ansonsten können Sie die Unirechner verwenden.

Um NASM entwickeln zu können, sollten Sie sich mit folgenden Begriffen / Tools bekannt machen. Beantworten Sie für sich folgende Fragen:

- Was ist eine Konsole/ein Terminal?
- Was ist eine Shell? Wie funktioniert eine Shell?

Beispiele für Shells wären:

- bash
- zsh
- fish

Folgende Programme benötigen Sie zum NASM-Programmieren. Machen Sie sich mit ihnen vertraut.

- nasm
- gcc
- ld
- gdb
- objdump

Weiterhin benötigen Sie noch einen Editor zum Schreiben der Programme. Es gibt Editoren, die in einer Konsole ausgeführt werden und auch welche, die als ”normales” graphisches Fenster ausgeführt werden. Beispiele die Ihnen auch auf den Poolrechner zur Verfügung stehen sind:

- nano (Konsole)
- gedit (graphisch)
- kate (graphisch)
- vim (Konsole)

Gausssumme

Schreiben Sie eine NASM-Funktion, welche die geschlossene Form der Gausssumme implementiert:

$$\frac{n(n+1)}{2} \quad \text{bzw.} \quad \frac{n^2+n}{2}$$

Machen Sie sich dazu mit Arithmetikbefehlen (ADD, SUB, MUL, DIV, IDIV, IMUL, NEG) in NASM vertraut. Wie genau funktionieren diese Befehle und warum gibt es zwei Multiplikations- und zwei Divisionsbefehle?
Die Funktion soll folgende Signatur haben:

```
uint64_t gauss(uint64_t n);
```

Machen Sie sich dafür mit Funktionsaufrufen auf Assemblerebene vertraut. Wo stehen die übergebenen Parameter? Wo muss der Rückgabewert hingeschrieben werden? (Stichwort: Calling Convention)
Ein geeigneter C-Wrapper zum Ausführen der Funktion wird Ihnen im KVV gestellt. Warum ist ein solcher Wrapper nötig?

Hinweis zu den C-Wrappern: Programmieren in C ist nicht Teil dieses Kurses, deswegen werden Ihnen die nötigen C-Programme/Programmteile für diesen Kurs gestellt. In darauf aufbauenden Veranstaltungen, bspw. Betriebs- und Kommunikationssysteme, müssen Sie selbständig in C programmieren. Es kann deswegen nicht schaden sich mit den C-Wrappern auseinander zu setzen bzw. auch mal einen selbst zu schreiben.