Grundlagen der Informatik III

Wintersemester 2010/2011

Wolfgang Heenes, Patrik Schmittat

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Praktikum

Ausgabe: 03.11.2010; Abgabe: 17.11.2010, 23:59 Uhr

Aufgabe 1: Einrichtung und Test der Entwicklungsumgebung, Abgabe der Aufgaben

Die Programmierung der Assemblerprogramme kann auf den Rechnern der RBG durchgeführt werden. Folgender Rahmen steht zur Verfügung.

```
.data
intout:
    .string  "Wert␣%d\n"

.text
.globl  main

main:

#  ...  Hier Ihren Code einfuegen ...

# Wert im %eax ausgeben
pushl %eax
pushl $intout
call  printf

# Exit
movl $1, %eax
int  $0x80
```

Wenn Sie ihr Programm in der Datei test.asm abspeichern, können Sie mittels folgender Befehle und dem entsprechenden Betriebsystem ihr Programm assemblieren, linken und ausführen. Vorausgesetzt wird eine funktionierende Installation von Yasm (`http://www.tortall.net/projects/yasm/`).

Unix / RBG-Pool	Mac OS X
yasm −f elf −p gas **test**.asm gcc −melf_i386 −o **test** **test**.o ./**test**	yasm −f macho32 −p gas **test**.asm gcc −arch i386 −o **test** **test**.o ./**test**

Hinweise zu einem Shell-Skript finden Sie unter `http://d120.de/forum/viewtopic.php?f=179&t=20777`.

Unter Windows kann das Assemblieren und Linken der Assemblerprogramm mit Visual Studio und Yasm erfolgen. Eine Anleitung steht z. B. im Moodle-Forum unter `https://moodle.informatik.tu-darmstadt.de/mod/forum/discuss.php?d=5493`.

Der Zugriff auf die RBG-Rechner kann von Zuhause erfolgen: Folgende Schritte sind bei Windows-Rechnern dazu notwendig (sofern nicht schon geschehen):

- WinSCP installieren um Dateien an die RBG-Rechnern zu übermitteln (Quelle: `http://winscp.net/eng/index.php`)
- Zugangsdaten eingeben Host: clientssh1.rbg.informatik.tu-darmstadt.de, Port: 22, Benutzername: RBG-Account (ohne Zusatz)

- Nun können Sie Assemblerprogrammen in ihr Home-Verzeichnis hochladen.

Um noch die Assemblierung, das Linken und die Ausführung vorzunehmen kann man für Windows-Rechner z. B. PuTTY installieren.

- PuTTY runterladen, Quelle: `http://www.putty.org/`
- PuTTY konfigurieren
  Host: clientssh1.rbg.informatik.tu-darmstadt.de, Port: 22
- Einloggen
  Die Konsole startet, indem man auf Open klickt. Nach Eingabe von Benutzername (RBG-Account ohne Zusatz) und Passwort kann man die Assemblierung, das Linken und die Ausführung vornehmen.

Bei Unix-Rechnern können die Konsolenbefehle `scp` und `ssh` verwendet werden. Hier gelten wieder die gleichen Hosts und Ports, sowie Benutzernamen.

```
ssh <RBG−Account>@clientssh1.rbg.informatik.tu−darmstadt.de
scp <lokalerPfad> <RBG−Account>@clientssh1.rbg.informatik.tu−darmstadt.de:<entfernterPfad>
```

Abgabe der Programme: Die Abgabe der Programme erfolgt über den SVN-Server `https://ics.ra.informatik.tu-darmstadt.de/svn/`. Die Beantwortung der Theoriefragen erfolgt in einem eigenen Dokument (PDF).

Achtung: Aufgrund der verlängerten Anmelde- und Ummeldefristen für die Übungen ist der Upload im SVN erst ab dem 15.11.2010, 12:00 Uhr möglich.

Aufgabe 2: Ägyptisches Multiplizieren

Das Produkt **m** zweier ganzer, vorzeichenbehafteter Zahlen **a** und **b** lässt sich leicht durch das sogenannte *ägyptische Multiplikationsverfahren* berechnen.

Der Multiplikand wird ständig verdoppelt, der Multiplikator (unter Wegwerfen des Restes) ständig halbiert; aufaddiert werden sogleich oder schlussendlich diejenigen Vielfachen, bei denen in der Multiplikatorhalbierung ein Rest weggeworfen wurde. Bedenken Sie zusätzlich Sonderfälle wie $b < 0$ und $a == 0$.

Beispiel: $\mathbf{a} = 11$, $\mathbf{b} = 5$

11	5
(22)	2
44	1
55	

Die mit „(..)“gekennzeichnete Zahl wird für die Produktbildung nicht aufaddiert.

1. Implementieren Sie den Algorithmus in Java. Die Benutzung der Multiplikation und Division (*, / und %) ist dabei nicht zulässig. Benutzen Sie stattdessen die Shiftoperationen.
2. Implementieren Sie den Algorithmus in IA32-Assembler. Dabei sollen 32-Bit Integer, also vorzeichenbehaftete Zahlen in 2K-Darstellung, verwendet werden. Sie können ATT- oder Intel-Syntax verwenden. Beachten Sie dabei:
   - Die Multiplikationsbefehle mul(l) und imul(l) sowie die Divisionsbefehle div(l) und idiv(l) dürfen nicht verwendet werden.
   - Die Werte a und b können Sie fest im **.data**-Bereich unterbringen sowie andere Werte die gegebenenfalls benötigt werden.
   - Kommentieren Sie Ihre Lösung.
3. Testen Sie Ihr Programm mit den Werten

a	b
11	5
-99	99
13	-50
-72	-32
0	56

Aufgabe 3: Variablentausch

Speziell bei Prozessoren und Mikrokontrollern mit einer geringen Anzahl von Registern stellt sich häufig das Problem, den Inhalt zweier Register miteinander vertauschen zu müssen, ohne dass ein weiteres, unbenutztes Register zur Verfügung steht. Eine erste Lösung des Problems liegt in der Verwendung eines Zwischenspeicherbereiches im Hauptspeicher.

```
  ...
movl  %eax, temp
movl  %ebx, %eax
movl  temp, %ebx
  ...
```

Eine andere Möglichkeit, die ausschließlich unter Verwendung der Register %eax und %ebx funktioniert, besteht in der ausschließlichen Verwendung des XOR-Befehls.

1. Welchen Nachteil hat die Lösung mit Verwendung des Hauptspeichers?
2. Schreiben Sie ein IA32-Assemblerprogramm (ATT- oder Intel-Syntax), welches unter Verwendung des XOR den Variablentausch durchführt. Erklärung zu XOR: **a** und **b** sind die Eingangsvariablen, **c** ist die Ausgangsvariable.

a	**b**	**c**
0	0	0
0	1	1
1	0	1
1	1	0

3. Es existiert eine weitere Möglichkeit Werte zwischen zwei Registern mittels eines einzigen Befehls auszutauschen. Um was für einen Befehl handelt es sich? Geben Sie als Beispiel einen Tausch zwischen %**eax** und %**ebx** mit diesem Befehl an.

Aufgabe 4: Arithmetische Ausdrücke

Gegeben ist der folgende arithmetische Ausdruck:

c - a * b + d / ((e % f) - g) / h

1. Stellen Sie diesen Ausdruck als binären Baum dar. Ergänzen Sie zunächst im Ausdruck Klammern, die einzelne Terme zusammenfassen. Beachten Sie dabei die Berechnungsfolge. **Hinweis:** Operatoren gleicher Priorität sollen von links nach rechts ausgewertet werden.
2. Geben Sie die LR- und RL-Postorder-Reihenfolge der Knoten (mit eingefügten push-↓ / pop-↑-Operationen) an.
3. Welche Reihenfolge ist hinsichtlich der Anzahl der Speicherzugriffe (Stack) günstiger?
4. Schreiben Sie ein IA32-Assemblerprogramm (ATT- oder Intel-Syntax) welches die LR-Postorder-Darstellung in Maschinenbefehle abbildet. Dabei sollen 32-Bit Integer, also vorzeichenbehaftete Zahlen in 2K-Darstellung, verwendet werden. Halten Sie die Zwischenergebnisse auf dem Stack. Fehlersituationen wie Überlauf und Division durch Null sollen **nicht** berücksichtigt werden. Das Ergebnis soll am Ende in Register %**eax** stehen. Es dürfen nur die Register %**eax**, %**ebx**, %**edx** verwendet werden, die Variablen dürfen nicht verändert werden und es dürfen, außer die Eingabewerte a, b, c, d, e, f, g, h keine zusätzlichen Variablen im **.DATA**-Bereich vereinbart werden. Testen Sie Ihr Programm mit folgenden Werten.

   a = 3, b = 11, c = 67, d = 16, e = 15, f = 2, g = 5, h = -2
5. Was passiert, falls bei einer Division der Nenner Null ist?