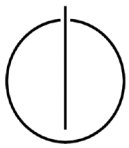
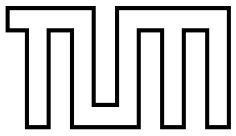


FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN



Lehrstuhl für Rechnertechnik und Rechnerorganisation
Prof. Dr. Martin Schulz

Einführung in die Rechnerarchitektur

Wintersemester 2017/2018

Tutorübung 3 06.11–10.11.2017

Aufgaben zu Speicherzugriffen

1. (a) Wie müsste man `MOV EBX, [EAX +ECX * 4 + 12]` nur mit reiner Benutzung der indirekten Adressierung schreiben (d.h. kein Index, kein Offset, keine Skalierung im Befehl). `MUL` soll auch nicht verwendet werden. Der „Nachbau“ darf nur dieselben Register wie das Original verändern!

   (b) Zwei an aufeinanderfolgenden Speicheradressen liegende 16-Bit-Worte sollen aufaddiert werden, das Ergebnis soll unmittelbar anschliessend abgespeichert werden. Haupspeicheradresse: `EAX` (soll nicht verändert werden!).

   (c) Es soll auf ein Feld mit 32-Bit-Worten zugegriffen werden. Der Feldbeginn stehe in `EBX`, die Nummer des Feldelements in `EAX`, das Ergebnis soll in EAX sein. Varianten mit und ohne „scaled index“-Adressierungsart.

   (d) Vertauschung zweier 64-Bit-Langworte, die hintereinander ab `EBX` im Speicher liegen.

2. Übersetzung von Hochsprachenstrukturen

   Gegeben seien folgende Strukturen/Klassen in Java-ähnlicher Notation

```
public class Struktur1 {
  public int wert;
  public int feld1[16];
}

public class Struktur2 {
  public Struktur1 feld2[8];
  public int info;
}
```

   Der Typ `int` besitzt einen Wert mit 32 Bit. Die Elemente vom Typ `Struktur1` in `feld2` sollen als Referenzen (Zeiger) gespeichert werden. Die Startadresse eines Objekts `obj` vom Typ `Struktur2` soll im Register `EAX` abgelegt sein. Die Feld-Indizes sollen mit 0 anfangen.

   (a) Wie sieht eine Realisierung der Datenstrukturen im Speicher aus?

   (b) Wie kann man auf den Inhalt von `obj.info` zugreifen (Ergebnis nach `EDX`)?

(c) Wie kann man auf den Inhalt von `obj.feld2[n]` zugreifen, wobei n im Register `EBX` abgelegt ist?

(d) Wie kann man auf den Inhalt von `obj.feld2[n].feld1[m]` zugreifen, wobei n und m in den Registern `EBX` bzw. `ECX` abgelegt sind?

(e) Wie müsste der Lesezugriff von 2d aussehen, wenn `feld2` nicht aus Referenzen/Zeigern, sondern direkt aus 8 hintereinanderliegenden `Struktur1`-Elementen bestehen würde?

3. Anhand der Intel Instruction Set Reference sollen die zu den Opcodes `MOV EAX, [EBX + 10]` und `MOV EBX, [EAX + ECX * 4 + 12]` gehörenden Bytefolgen (als Hexwerte) zusammengebaut werden.

Fließkommarechnung

4. Gegeben sei folgende Formel:

$$W = s \cdot \frac{m}{D} \cdot B^{e-b} \quad \text{mit}$$

s	Vorzeichen/Sign	$s \in \{-1, 1\}$
m	Mantisse	$m \in \mathbb{N}_0$
D	Mantissen-Divisor	$D \in \mathbb{N}$
B	Basis	$B \in \mathbb{N}, B > 1$
e	Exponent	$e \in \mathbb{N}_0$
b	Verschiebung/Bias	$b \in \mathbb{N}_0$

(a) Welchen Wertebereich kann man mit W darstellen, wenn $0 \leq m \leq 99$, $0 \leq e \leq 18$, $b = 9$, $D = B = 10$? Was ist die kleinste bzw. die größte Schrittweite?

(b) Der IEEE-Standard 754 legt den in vielen Programmiersprachen vorhandenen Datentyp *single precision* folgendermaßen fest:

$B = 2$, $m = 24$Bit, $D = 2^{23}$, $e = 8$Bit, $b = 127$. Die Werte 0 und 255 für e sind nicht benutzbar, da für die Signalisierung bestimmter Sonderfälle (unendlich, etc.) reserviert.

Welcher Wertebereich und welche Schrittweite können prinzipiell damit erreicht werden, wenn man das Vorzeichen nicht beachtet?

(c) Der IEEE-Standard legt fest, dass das höchstwertige Bit der Mantisse immer 1 sein muss (Man sagt „die Mantisse ist normalisiert“). Warum ist diese Festlegung überhaupt ohne Lücken in der Zahldarstellung möglich und welchen Vorteil hat sie?

(d) Wie kann man mit der Fließkommadarstellung addieren bzw. multiplizieren? Was fällt bei der Addition auf?

(e) Stellen Sie die Vor- und Nachteile von Festkomma und Fließkomma gegenüber.