ERA-Tutorium 3

Thomas Kilian

Organisatorisches

- Gibt es Fragen zum letzten Tutorium?

- Wünsche und Anregungen?

Fragen zur Vorlesung

Fragen zur Vorlesung

Arbeitsablauf C-Compiler?

Fragen zur Vorlesung

Arbeitsablauf C-Compiler?

➡ compile, assemble, link, load, (execute)

Fragen zur Vorlesung

Arbeitsablauf C-Compiler?

➡ compile, assemble, link, load, (execute)

Wie ist Speicher beim x86 organisiert?

Fragen zur Vorlesung

Arbeitsablauf C-Compiler?

➡ compile, assemble, link, load, (execute)

Wie ist Speicher beim x86 organisiert?

➡ Größe Speicherzelle: 8 Bit (1 Byte)

Fragen zur Vorlesung

Arbeitsablauf C-Compiler?

➡ compile, assemble, link, load, (execute)

Wie ist Speicher beim x86 organisiert?

➡ Größe Speicherzelle: 8 Bit (1 Byte)

➡ Jede einzelne Zelle adressierbar (in 8-Bit-Schritten)

Fragen zur Vorlesung

Fragen zur Vorlesung

Welche Adressierungsarten gibt es?

Fragen zur Vorlesung

Welche Adressierungsarten gibt es?

➡ Register Indirect Mode

Fragen zur Vorlesung

Welche Adressierungsarten gibt es?

- ➡ Register Indirect Mode
- ➡ Based Mode

Fragen zur Vorlesung

Welche Adressierungsarten gibt es?

- ➡ Register Indirect Mode
- ➡ Based Mode
- ➡ Based Indexed Mode

Fragen zur Vorlesung

Welche Adressierungsarten gibt es?

- ➡ Register Indirect Mode
- ➡ Based Mode
- ➡ Based Indexed Mode
- ➡ Based Indexed Mode with Displacement

Fragen zur Vorlesung

Welche Adressierungsarten gibt es?

- ➡ Register Indirect Mode
- ➡ Based Mode
- ➡ Based Indexed Mode
- ➡ Based Indexed Mode with Displacement
- ➡ Based and Scaled Index with Displacement

Fragen zur Vorlesung

Fragen zur Vorlesung

Sind folgende Assembler-Befehle legal?

Fragen zur Vorlesung

Sind folgende Assembler-Befehle legal?

- MOV [EAX], [EBX]
- MOV EAX, [AX]
- MOV [ESI], EDX

Fragen zur Vorlesung

Sind folgende Assembler-Befehle legal?

- ~~MOV [EAX], [EBX]~~
- MOV EAX, [AX]
- MOV [ESI], EDX

Fragen zur Vorlesung

Sind folgende Assembler-Befehle legal?

- ~~MOV [EAX], [EBX]~~
- ~~MOV EAX, [AX]~~
- MOV [ESI], EDX

Aufgabe 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.

Aufgabe 4

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

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	$b \in \mathbb{N}_0$

Aufgabe 4a

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?

Aufgabe 4b

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?

Aufgabe 4c

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?

Aufgabe 4d

Wie kann man mit der Fließkommadarstellung addieren bzw. multiplizieren?

Was fällt bei der Addition auf?

Aufgabe 4e

Vergleich Fest- und Fließkomma

Aufgabe 4e

Vergleich Fest- und Fließkomma

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	Aufwendige Addition

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	Aufwendige Addition
Konstante Auflösung	

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	Aufwendige Addition
Konstante Auflösung	Dynamische Auflösung

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	Aufwendige Addition
Konstante Auflösung	Dynamische Auflösung
Kein Rundungsfehler	

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	Aufwendige Addition
Konstante Auflösung	Dynamische Auflösung
Kein Rundungsfehler	Auslöschung möglich

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	Aufwendige Addition
Konstante Auflösung	Dynamische Auflösung
Kein Rundungsfehler	Auslöschung möglich
Geringe Dynamik	

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	Aufwendige Addition
Konstante Auflösung	Dynamische Auflösung
Kein Rundungsfehler	Auslöschung möglich
Geringe Dynamik	Großer Wertebereich

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	Aufwendige Addition
Konstante Auflösung	Dynamische Auflösung
Kein Rundungsfehler	Auslöschung möglich
Geringe Dynamik	Großer Wertebereich
Kein Universalformat	

Aufgabe 4e

Vergleich Fest- und Fließkomma

Festkomma	Fließkomma
Einfache Grundrechenarten	Aufwendige Addition
Konstante Auflösung	Dynamische Auflösung
Kein Rundungsfehler	Auslöschung möglich
Geringe Dynamik	Großer Wertebereich
Kein Universalformat	Relativ universell einsetzbar

Aufgabe 1a

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!

Aufgabe 1b

Zwei an aufeinanderfolgenden Speicheradressen liegende 16-Bit-Worte sollen aufaddiert werden, das Ergebnis soll unmittelbar anschliessend abgespeichert werden.

Haupspeicheradresse: EAX (soll nicht verändert werden!).

Aufgabe 1c

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.

Aufgabe 1d

Vertauschung zweier 64-Bit-Langworte, die hintereinander ab EBX im Speicher liegen.

Aufgabe 2: Datenstruktur

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

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

Aufgabe 2b

Wie kann man auf *obj.info* zugreifen (->EDX)?

Aufgabe 2c

Wie kann man auf *obj.feld2*[n] zugreifen (n<-EBX)?

Aufgabe 2d

Wie kann man auf *obj.feld2*[n].*feld1*[m] zugreifen (n<-EBX, m<-ECX)?

Aufgabe 2e

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