# ERA-Tutorium 5

Thomas Kilian

# Organisatorisches

- Gibt es Fragen zum letzten Tutorium?

- Wünsche und Anregungen?

# Flags

- beim x86

  ➡ CF: Übertrag (Carry)

  ➡ OF: Überlauf (Overflow)

  ➡ ZF: Zero-Flag

  ➡ SF: Sign-Flag

- das Setzen der Flags erfolgt immer bezogen auf die Operandengröße (Bitgröße)

# Flags

- werden hauptsächlich durch arithmetische Operationen gesetzt
- insbesondere nicht verändert durch folgende Befehle:

  ➡ MOV, PUSH, POP, JMP, CALL, RET

➡ werden für bedingte Sprünge verwendet

# Beispiel: Zahlenvergleich

- **SUB: SUB EAX, EBX**
  - ➡ ZF = 1 => EAX = EBX
  - ➡ ZF = 0 => EAX ≠ EBX
  - ➡ SF = 1 => EAX < EBX (unsigned)
  - ➡ CF = 1 => EAX < EBX (signed)
- **CMP: CMP EAX, EBX**
  - ➡ wie SUB, allerdings wird Ergebnis verworfen

# Bedingte Sprünge

**Jcc <marke/adresse>**

| cc | Bemerkung | Flags |
|---|---|---|
| C / NC | carry / not carry | CF = 1 / CF = 0 |
| E, Z / NE, NZ | equal, zero / not equal, not zero | ZF = 1 / ZF = 0 |
| O / NO | overflow / not overflow | OF = 1 / OF = 0 |
| G / NLE | greater, not less or equal | SF = OF ∧ ZF = 0 |
| GE, NL | greater or equal, not less | SF = OF |
| L, NGE | less, not greater or equal | SF ≠ OF |
| LE, NG | less or equal, not greater | SF ≠ OF ∨ ZF = 1 |

# Beispiele

**CMP EAX, EBX**

JE <Ziel EAX = EBX>

JLE <Ziel EAX <= EBX>

**ADD EAX, EBX**

JO <Ziel bei Überlauf>

# Beispiele

- unbedingt, statisches Ziel

  ➡ JMP/CALL <marke/adresse>

- unbedingt, dynamisches Ziel

  ➡ JMP EAX, CALL [EAX], RET

- bedingt

  ➡ JE <marke/adresse>

  ➡ nicht dynamisch!

# Aufgabe 1a

Welche Möglichkeiten an Befehlskombinationen gibt es, um an die Marke “*marke1*” zu springen, wenn der Inhalt von AL größer oder gleich als 5 ist?

# Aufgabe 1b

Bestimmen Sie die Bestandteile folgender einfacher while-Schleife in C/Java und übersetzen Sie sie in 80386-Assembler:

```
int ebx;
ebx = 50;
while (ebx <= 60)
{
   fkt (ebx);
   ebx = ebx + 1;
}
```

Die Funktion *fkt* sei bereits definiert und erwarte ihre Argumente in *EAX*.

# Aufgabe 1c

Erstellen einer Zählschleife von 100 bis einschließlich 0; Aufruf einer Funktion *fkt* (per Call) mit aktuellem Zählerstand in *EAX*.

# Aufgabe 1d

Zwei verschränkte Schleifen: Die äußere (über EAX) von 0 bis 100, die innere (über EBX) von 0 bis (einschließlich) EAX.

# Aufgabe 2

Es sollen mit einem Unterprogramm **Bytes ab Adresse *ESI* nach Adresse *EDI*** (und folgende) kopiert werden.

Die Anzahl liegt dabei nicht fest, stattdessen soll solange kopiert werden, bis ein **Byte mit dem Wert 0** kopiert wurde (Funktionsweise von *strcpy*).

# Aufgabe 3a

Verschiebung der Bits 3 bis 6 aus *EAX* an die Position 0 bis 3 in *EBX*, alle anderen Bits von *EBX* sollen 0 sein.

# Aufgabe 3b

Einfügen der drei niederwertigsten Bits von *EAX* in die Bits 7 bis 5 von *EBX*, ohne Veränderung von *EAX*/*ECX*/*EDX* und der übrigen Bitstellen von *EBX*.

# Aufgabe 3c

Überprüfung, ob Bit 14 bis 11 in *EAX* den Wert *1001* haben, bei Gleichheit Sprung irgendwohin.

# Aufgabe 3d

Schnelle Division von *EAX* (signed) durch 16, keinen *div*-Befehl nutzen!

An zwei Beispielwerten (positiv und negativ) nachrechnen.

# Aufgabe 3e

Rechnung *EAX* Modulo 64.

An einem Beispielwert nachrechnen.

# Aufgabe 3f

Nullsetzen von Bit *n* (<- *EBX*) in *EAX*.