Wiedereintrittsfestigkeit (Java)

Java-Algorithmus

```
10 public class Summe
11 {
12   public static void main(String[] args)
13   {
14     Summe MeineSumme = new Summe();
15   }

16   public Summe()
17   {
18     System.out.println(sum(20));     // Summe bis 20
19   }
```



```
20   public int sum(int n)
21   {
22     if (n>0) return n+sum(n-1);     // Summe bis n-1 berechnen
23     else return 0;                  // Rekursion abbrechen
24   }

25 }
```


Wiedereintrittsfestigkeit (Assembler)

Implementierung: Summe

```
10 _sum: MOVE.L  4(SP),D0    | n in Register
11       BEQ     ok          | n auf 0 prüfen
12       SUBQ.L  #1,D0       | D0 := n-1
13       MOVE.L  D0,-(SP)    | n-1 auf den Stack
14       JSR     _sum        | Rekursion
15       ADDQ.L  #4,SP       | Stack aufräumen
16       ADD.L   4(SP),D0    | letztes n addieren
17 ok:   RTS                 | fertig!
```


Beispiele zu Assemblerprogrammen (I)

Beispiel: Bytetausch

- 2 Computer (mit MC68000 bzw Intel-Pentium) über Netz verbunden
- MC68000: Berechnung von Daten, die Wort-weise hintereinander im Speicher abgelegt werden; letzter Eintrag: 0xFEDC
- byte-weise Übertragung der Daten zum Speicher des Intel-Pentium
- dann: Weiterbearbeitung der Daten durch Intel-Pentium
- Problem: MC68000 muss vor Übertragung die Daten in richtiges Format bringen: höherwertiges und niederwertiges Byte vertauschen
- soll durch Assemblerprogramm erfolgen, aufgerufen aus C mit `void biglittle(unsigned short *quelle, unsigned short *ziel);`
- `quelle`: Speicheradresse des ersten umzuwandelnden Datenwortes
- Ablage der umgewandelten Worte ab Adresse `ziel`

Assemblerbeispiele (III)

Beispiel: Hamming

- Fehlererkennung mit Hilfe des Hamming-Codes
- dazu: m zusätzliche Paritätsbits
- Nutzdatenbits und Paritätsbits ergeben Codewort
- m muss Gleichung $2^m \geq n + m + 1$ erfüllen
- n: Anzahl der Nutzdatenbits
- Assemblerprogramm zur Ermittlung des kleinsten m: beginnend mit $m = 1$ soll m solange inkrementiert werden, bis Gleichung erfüllt
- Aufruf aus C mit `unsigned int hamming(unsigned int *n);`
- Bereichsüberschreitung von 2^m soll durch $m = 0$ gemeldet werden