Informatik II

Übungsstunde 4

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Stacks, Ackermann, Bytecode
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Java-Bytecode

- Bytecode ist die Maschinensprache der Java-VM

- Bytecode ist ziemlich kompakt: Die meisten Instruktionen („Operationen") sind nur 1 Byte (= 8 Bit) lang
  - Kennzeichnung durch einen 8-Bit-Operationscode
  - Haben zusätzlich auch eine „symbolische" Bezeichnung, z.B.:
    - **add** mit Operationscode 01100000 (dezimal 96, hexadezimal 0x60)
    - **iconst_3** mit Operationscode 00000100
    - **pop** mit Operationscode 01010111

Übersetzung in Java-Bytecode (2)

```
static void q() {
  int i=0, j=0;
  boolean b = false;
  b = (4*i + (3+j)*2) > j+7*i;
  b = b & (33 > 2+j);
}
```

Mit dem Konsolen-Kommando
javap -c xx
kann man sich den Java-Bytecode der Klasse xx in symbolischer Form anzeigen lassen, den der Java-Compiler (z.B. in der Datei xx.class) generiert hat.

Tipp: Dieses ausprobieren, man macht interessante Entdeckungen!

```
 0 iconst_0
 1 istore_0   i
 2 iconst_0
 3 istore_1   j
 4 iconst_0
 5 istore_2   b
 6 iconst_4
 7 iload_0
 8 imul
 9 iconst_3
10 iload_1
11 iadd
12 iconst_2
13 imul
14 iadd
15 iload_1
16 bipush 7
18 iload_0
19 imul
20 iadd
21 if_icmpgt 28
24 iconst_0
25 goto 29
28 iconst_1
29 istore_2
30 iload_2
31 iconst_2
32 iload_1
33 iadd
34 bipush 33
36 if_icmplt 43
39 iconst_0
40 goto 44
43 iconst_1
44 iand
45 istore_2
```

Eine Auswahl von VM-Instruktionen (2)

iconst_m1
Push the integer -1 onto the stack.

iconst_0, ... iconst_5
Push the integer 0,..., 5 onto the stack.

iload_vindex
The value of the local variable at vindex in the current Java frame is pushed onto the operand stack.

istore_vindex
Local variable vindex in the current Java frame is set to value.

nop
Do nothing

bipush byte
Push one-byte signed integer.

pop
Pop top stack word from the stack.

iadd
Value1 and value2 must be integers. The values are added and replaced on the stack by their integer sum.

imul
...replaced on the stack by their integer product.

vindex: die Variablen eines „frames" sind durchnummeriert; vindex ist dabei die Nummer („index") einer Variablen

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Stacks, Ackermann, Bytecode
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Übung 3

- 1. Stack
  - Wir kennen alles -> Wir koennen alles...
- 2. Stackimplementierung Ackermannfunktion
- 3. Java Bytecode
  - Disassembling & Analyse

Übung 3 – Aufgabe 1

- Stack
  - `pop(), push(), empty(), size, capacity` kennen wir schon
  - `peek(), toString()` sind neu...
- Baut einen Stack
- Soll automatisch wachsen (genau wie in C++)

Übung 3 – Aufgabe 2

- Iterativer Ackermann...
- Keine Rekursion! Verwendet einen Stack, um einen iterativen Algorithmus zu bauen...
- Idee ist, zu zeigen, was ein Stack alles kann! ☺
- Einfache Aufgabe.....

  ...wenn man begriffen hat, wie der Stack zu verwenden ist!

Übung 3 – Aufgabe 2

$$
\begin{aligned}
A(0,m) &= m+1 \\
A(n+1,0) &= A(n,1) \\
A(n+1,m+1) &= A(n,A(n+1,m))
\end{aligned}
$$

`Beispiel:  A(1,1) = A(0, A(1,0)) = A(0, A(0,1))`

`Kennt ihr (postfix...):  *(4, +(2,3)`

...wie koennen wir das nun mit einem Stack verwenden?

- Idee: Gleich wie bei Postfix-Rechnungen (`4 2 3 + *`)
  - Argumente pop-en
  - Argumente anschauen → Fallunterscheidung
  - Ergebnis push-en

- Funktioniert das? Warum? Fuehlt sich nach Problem an, oder...

Übung 3 – Aufgabe 3

- Java Bytecode und Disassembler `javap`
  - `javap` liegt in eurem jdk-Verzeichnis, im /bin folder (wie `java`)

- Falls beim disassemblen nicht alle Methoden auftauchen, verwendet noch die -private Option

```
javap -c -private MyClass
```

a) RecursiveAckermann.A disassemblen
b) Einzelne Zeilen analysieren
c) Vergleich mit Bytecode aus 2c
   - Iterativ vs. rekursiv...

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Stacks, Ackermann, Bytecode
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Projekt: Keine Ahnung

- Schauen wir mal auf die Zeit...

Informatik II

Übungsstunde 4

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich