Aufgabe 7: Funktionsaufruf

Gegeben sei folgende Methodendeklaration:

```
int max (int a, int b) {
   if (a > b) return a; else return b;
}
```

a) Geben Sie den MicroJava-Bytecode für diese Methode an.

b) Geben Sie den MicroJava-Bytecode für den Methodenaufruf `x = max(y, 10)` an, wobei `x` und `y` lokale `int`-Variablen auf den Adressen 0 und 1 sind.

Lösung

a) Bytecode für die Methodendeklaration

```
100  enter 2, 2
103  load0                    if (a > b)
104  load1
105  jle 9 (false-jump => 114)
108  load0                        return a;
109  exit
110  return
111  jmp 6 (=> 117)
114  load1                    else return b;
115  exit
116  return
117  trap 1
```

Die Methode `max` hat 2 Parameter und sonst keine lokalen Variablen. Daher ist auch die Anzahl der lokalen Variablen 2. Die `enter`-Instruktion legt einen Stack Frame mit 2 Worten an und kopiert die beiden Parameter vom EStack an den Anfang des Frames.

Die beiden `return`-Anweisungen laden jeweils den Rückgabewert auf den EStack und verlassen dann die Methode mittels `exit` und `return`.

Die `jmp`-Instruktion auf Adresse 111 ist eigentlich redundant, da sie nie erreicht wird. Man könnte sie bei einer Optimierung des Codes entfernen, was aber im MicroJava-Compiler nicht gemacht wird.

Am Ende der Funktion steht eine `trap`-Instruktion, die einen Laufzeitfehler melden soll, wenn die Methode ohne `return`-Anweisung verlassen wird. Auch das könnte man wegoptimieren, da hier in beiden Zweigen der `if`-Anweisung ein `return` erfolgt. Der MicroJava-Compiler optimiert das aber nicht.

b) Bytecode für den Funktionsaufruf

```
200  load1
201  const 10
206  call -106 (=> 100)
209  store0
```

Die beiden aktuellen Parameter `y` und 10 werden auf den EStack geladen. Anschließend wird die Methode `max` aufgerufen. Unter der Annahme, dass `max` auf Adresse 100 liegt, beträgt die Call-Distanz `100 - 206 = -106`. Die Funktion `max` hinterlässt ihr Ergebnis auf dem EStack, so dass es mit `store0` in der Variablen `x` abgelegt werden kann.