Exercise 7: Function Call

Consider the following declaration of a function method:

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

a) Specify the MicroJava bytecode for this method.

b) Specify the MicroJava bytecode for the function call sum = max(x, 10);, where sum and x are local int variables at addresses 0 and 1.

Solution

a) Bytecode for the method declaration

```
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
```

The method max() has 2 parameters and no other local variables. Therefore, the number of local variables is also 2. The enter instruction creates a stack frame with 2 words and copies the two parameters from the EStack to the start of the frame.

The two return statements each load the return value onto the EStack and then exit the method using exit and return.

The jmp instruction at address 111 is actually redundant, as it is never reached. It could be removed when optimizing the code, but this is not done in the MicroJava compiler.

At the end of the function method there is a trap instruction that should report a run-time error if the method is exited without a return statement. This could also be optimized away, as a return occurs in both branches of the if statement. However, the MicroJava compiler does not optimize this.

b) Bytecode for the function call

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

The two actual parameters y and 10 are loaded onto the EStack. The function max() is then called. Assuming that max() is at address 100, the call distance is 100 - 206 = -106. The function max() leaves its result on the EStack so that it can be stored in the variable x with store0.