14.2 BYTECODE OPTIMIZATION TECHNIQUES

Sometimes you can compute the results of an operation even before it is executed and replace the code with a simple push of the results. Consider this Java code:

```java
type seconds_in_a_day = 60*60*24;
```

A naïve compiler might compile this to

- `bipush 60`
- `bipush 60`
- `imul`
- `bipush 24`
- `imul`

A better compiler would generate

```java
ldc 86400
```

This single instruction is likely to execute significantly faster than the five instructions. Although the difference may seem trivial, when it is executed a million times the differences add up.

14.2.4 Loop Unrolling

In some loops, the time it takes to execute the body of the loop may not be all that much larger than the loop tests. Consider, for example, this code to count the number of 1 bits in the `int m`:

```java
for(int i = 0; i < 32; i++)
    n += m >> i & 1;
```

A naïve Java compiler might generate this code:

```java
loop: iload_1 ; Compare i to 32
        bipush 32
        if_icmpge break ; Break when i >= 32
        iload_3 ; Push n
        iload_2 ; Push m
        iload_1 ; Push i
        ishr ; Compute m >> i
        iconst_1
        iand ; Compute m >> i & 1
        iadd ; Add that value to n
        istore_3 ; Store n
        iinc 1 1 ; Increment i
        goto loop ; Loop again
```