Covering Trivial Loop Evaluator optimization

# Idea of optimization

This optimization is useful for reducing loops that calculate invariant results. Basically what it does is it first detects whether a loop has constant inputs. Then determines if the operations on those constants from loop also yield constants. If that is the case, the compiler iterates through the loop and computes its output results. Those output results are then set via const bytecodes at the loop exit and the entire loop is then eliminated.

# What we need to cover

We need to cover all arithmetic operation types for all value types. Also, we need to test different cycle lengths, complex loop conditions. Final elements?

Negative tests: Possible exceptions in the loops, non-constant input, non-constant cycle lengths, loop breaks.

# Test matrix

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Type\op | add | sub | mul | div | rem | or | and | xor | shl | shr | ushr |
| Byte | X | X | X | X | X | X | X | X | X | X | X |
| Short | X | X | X | X | X | X | X | X | X | X | X |
| Int | X | X | X | X | X | X | X | X | X | X | X |
| Long | X | X | X | X | X | X | X | X | X | X | X |
| Float | X | X | X | X | X |  |  |  |  |  |  |
| Double | X | X | X | X | X |  |  |  |  |  |  |

Total 54 cases \*2 bytecode types (regular and 2addr) \* 4 cycle types (very short, short, very long, complex conditions) = 432 positive tests