McGill University
School of Computer Science
Sable Research Group

A Brief Description of Kaffe's Internals

Sable Technical Note No. 2

Raja Vallée-Rai

March 3, 1998

1 Relevant Files

▷ `kaffe/main.c`

This contains the starting point of the program. Note the call to `printProfileStats()`.

▷ `kaffevm/support.c`

Contains translation tables for the bytecode instructions (bytecode index to name), some custom global variables and the code for `printProfileStats()`.

▷ `kaffevm/kaffe.def`

Defines each Java bytecode instruction in terms of Kaffe's internal instruction representation. (uses the 1-argument form of `define_insn()` defined in `jit/machine.h`)

▷ `kaffevm/icode.h`

Defines the capabilities of the architecture, and the name of a method which implements the capability with native code.

▷ `kaffevm/jit/icode.c`

Contains the definitions for each internal instruction. Depending on the architecture (see `kaffevm/icode.h`), it may call one or several functions to produce the code for this instruction.

▷ `kaffevm/jit.def`

Contains functions which implement capabilities defined in `kaffevm/icode.h` by producing native code. The code is emitted with the macro OUT. (`define_insn(x, y)` is defined in `jit.funcs.c`)

▷ `kaffevm/jit/funcs.c`

Defines the two argument version of `define_insn`, as well as some macros such as `OUT = ...` and `RecordInstructionCost()`

▷ `kaffevm/jit/machine.c`

Contains the `translate(Method *)` function which translates methods into native code. (this is the starting point of the native code generation) the generation of native code)

2 Modification Hints

Emitting The Correct Opcodes

Determining the correct series of opcodes to emit by hand is pretty painful. Fortunately, there is an easier way:

1. Write a small C program which does roughly what you want (`sample.c`)
2. Compile it with `'gcc -S sample.c'` to produce an assembly version of the file (`sample.s`).
3. Modify `sample.s` to contain exactly the assembly instructions you wish to encode.
4. Assemble `sample.s` with `'as -a sample.s'` to produce an assembly listing containing the correct opcodes to use for each assembly instruction.

Use Grep

This advice is probably obvious to most readers. When studying Kaffe code and trying to locate the source file which contains a given symbol, simply use the unix tool `grep` (a common typical invocation is `grep symbol *.c *.h *.def`.)