JavaControlStack

This page collects notes about the use by Java of the hardware control stack.

Interpreter Stack Frames

An interpreter stack frame consists of four main parts: a variable part containing locals is shared with the caller, a fixed part is addressed by the frame pointer FP, a variable below that contains zero or more locally locked monitors, and a variable part just above SP contains the JVM temporary stack. (On x86 this last part is directly addressed by SP (rsp). Some CPUs other than x86 make alignment requirements on SP, or require information to be stored there, such as register windows. On these CPUs, a separate register may be used to point to the Java VM stack top, inside the variable part, which varies more slowly.) The fixed part contains the basic state variables for the interpreter, plus information for sizing the variable part.

The machine-dependent code for frame layout is in files like frame_x86.hpp, which are included into type type frame.

Here is the layout (which is typical) for the x86 assembly-based interpreter. It is set up by InterpreterGenerator::generate_normal_entry. Items are presented in descending order of memory:

- arguments — any values pushed just before this frame's caller's `call` instruction
- non-argument locals — any extra locals, allocated contiguously with the arguments on frame entry (the return PC is recopied as necessary)
- return PC — pushed by this frame's caller's `call` instruction, or copied from that push
- link — value of FP (rbp) in the caller, pointed to by this frame's FP
- sender_sp — value of SP (rsp) as of this frame's caller's `call` instruction (may be deeper in stack than address of return PC); it is passed in the interpreter calling sequence, through any adapters, in a special register (rsi)
- method — methodOop of the current method
- mdx — method data pointer (or offset, during GC), or NULL if current activation is not being profiled
- cache — constantPoolCacheOop for the current method (linkage information associated with constant pool, usually shared by all methods of a given class)
- locals — base pointer of locals array (this is the highest address, where local #0 is placed; sender_sp can point near the arguments within this array, or it can be more recently allocated by an adapter)
- bcx — bytecode pointer (or offset, during GC)
- initial_sp — base of local JVM temporary stack (may be adjusted to make space for monitors)
- monitors — a series of zero or more pairs, of a pointer to a locked object plus a lock state word (expanded as needed by copying the temp stack)
- stack temporaries — any pushed temporary values; the shallowest of these become the outgoing arguments to the frame's next callee

last_sp

The value last_sp is stored by `InterpreterMacroAssembler::prepare_to_jump_from_interpreted` and read by `TemplateInterpreterGenerator::generate_return_entry_for`, which pops the arguments (very late) in the expression `lea(rsp, ...)`. So last_sp points (dangerously) to outgoing arguments owned by the callee. Note that the callee is free to modify those arguments, and even change oops to ints and vice versa, since they are the callee's locals. With invokedynamic, low-level adapters between the caller and callee may also rearrange the arguments. All these changes would look to the caller like scribbling. Therefore, even though the caller has a last_sp which appear to point at an outgoing argument list, it would be an error to read or write that memory.

In `frame::oops_interpreted_do` there is a special stanza (guarded by `map->include_argument_oops()`) to handle the outgoing arguments; it is used only in cases where a call has blocked in the linker and the callee has not been determined yet. Outgoing argument oops will never be processed twice; this would cause duplication in the root set which breaks the GC (throwing asserts, in debug builds). The logic that prevents this from happening is inside `OopMapCache::compute_one_oop_map`, which avoids pointing the `InterpreterFrameClosure` at the wrong parts of the expression stack which might have been scribbled. (Even though last_sp points at those wrong parts.)