

C++17 Interoperable Register Machines

Daniel Nikpayuk

April 7, 2021



This article is licensed under
Creative Commons Attribution-NonCommercial 4.0 International.

Each **atomic machine** has the following form:

```
template<>
struct machine<name>
{
    template<stack...>
    static constexpr auto result(heaps...) {...}
}
```

The *name* allows for dispatching (template resolution), while the *constexpr function* has a single *stack* made up of a variadic pack symbolically representing *registers*, along with a fixed number of *heaps* which are also made up of variadic packs, but which are *cached*, and thus more expensive in general.

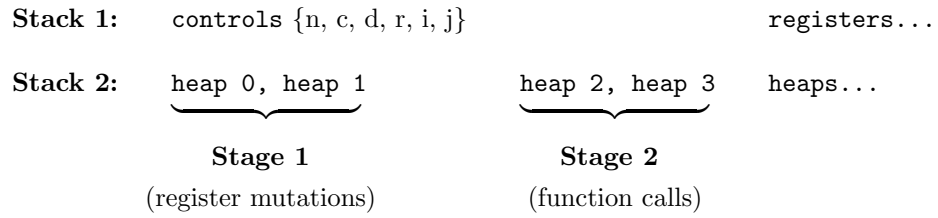
We then build **compound machines** by chaining them together with a **controller**. Such machines and controllers are organized into a **hierarchy** of machine orders using a *monadic* narrative design: The idea is that the atomics of a higher level are constructed from the compounds of lower levels which—assuming self-similarity propagates throughout—then allows this pattern to scale:

The idea is that with the chains of machines (at a given level) we can either end the chain with a *halting instruction* or a *passing instruction*. Halters effectively become standalone functions (with some interface hiding the chain), returning some standalone value. Passers on the other hand are intended to continuation pass to other machines at higher orders.

This then implies a few consequences for the design of each individual machine:

- Each machine is required to carry its own controller, which includes required indices (as well as a nesting depth counter), along with index iterators. Performance and modularization design suggests such info should generally be carried on the stack.
- Each machine that has a higher order is required to carry the controller, indices, iterators, of the machine it is eventually returning to. Abstraction-wise it makes the most sense to carry this info in a designated heap.

Anatomy of a Compile Time Register Machine (C++ Function Template)



A model for CTRM Trampolining which uses *internal* rather than tail function calls

external call machine

current call machine

internal call machine

filler.