Tinker Architecture Overview

Tinker is a 64-bit processor whose architecture includes 32 general-purpose registers (R0 through R31), a program counter (PC), and implements an instruction set that features 32 instructions. By convention, R31 functions as a stack pointer. The instruction set while limited, can express any program.

The registers R0 through R30 can contain any value. The interpretation of the value depends on the instruction that manipulates it. For example, a value R0 can hold a 64-bit signed integer, a 64-bit unsigned integer, a 64-bit Boolean array, a 64-bit double-precision number, or any other interpretation. There are no state flags (carry, overflow or zero). This is meant to simplify the implementation of the project.

When Tinker starts, the PC is set to address 0 and instructions are fetched from there sequentially until the halt instruction executes, which stops the processor (and the simulation run). Instructions are 4-byte wide and follow a fixed format. There is an opcode field (5 bits), followed by 3 registers specifiers, each taking up (5 bits), followed by a placeholder for a literal (12 bits). Not all fields are used for every instruction. Instructions fall into the following categories:

- Integer arithmetic
- Logical
- Control (branching and subroutines)
- Data movement
- Floating point arithmetic
- Input output

All arithmetic integer operations are done using signed 64-bit arithmetic. All floating-point arithmetic use double precision numbers. Tinker does not support single-precision floating point arithmetic.

Tinker uses integer and double precision arithmetic (it does not support single-precision arithmetic). For integers, the most significant bit is used as a sign bit according to the two's complement convention. The machine is a little-endian machine in which the smallest byte in a number is stored first (bits 56-63), and the most significant byte is stored last (bits 0-7). For double precision, it uses the IEEE 754-2000 standard format.