Limitations

Developing programs on the Solana blockchain have some inherent limitation associated with them. Below is a list of common limitation that you may run into.

Rust libraries#

Since Rust based on-chain programs must run be deterministic while running in a resource-constrained, single-threaded environment, they have some limitations on various libraries.

SeeDeveloping with Rust - Restrictions for a detailed breakdown these restrictions and limitations.

Compute budget#

To prevent abuse of the blockchain's computational resources, each transaction is allocated acompute budget . Exceeding this compute budget will result in the transaction failing.

Seecomputational constraints in the Runtime for more specific details.

Call stack depth - CallDepthExceeded error#

Solana programs are constrained to run quickly, and to facilitate this, the program's call stack is limited to a max depth of64 frames .

When a program exceeds the allowed call stack depth limit, it will receive theCallDepthExceeded error.

CPI call depth - CallDepth error#

Cross-program invocations allow programs to invoke other programs directly, but the depth is constrained currently to4 .

When a program exceeds the allowedcross-program invocation call depth , it will receive aCallDepth error

Float Rust types support#

Programs support a limited subset of Rust's float operations. If a program attempts to use a float operation that is not supported, the runtime will report an unresolved symbol error.

Float operations are performed via software libraries, specifically LLVM's float built-ins. Due to the software emulated, they consume more compute units than integer operations. In general, fixed point operations are recommended where possible.

TheSolana Program Library math tests will report the performance of some math operations. To run the test, sync the repo and run:

cargo test-sbf -- --nocapture --test-threads=1 Recent results show the float operations take more instructions compared to integers equivalents. Fixed point implementations may vary but will also be less than the float equivalents:

u64 f32 Multiply 8 176 Divide 9 219

Static writable data#

Program shared objects do not support writable shared data. Programs are shared between multiple parallel executions using the same shared read-only code and data. This means that developers should not include any static writable or global variables in programs. In the future a copy-on-write mechanism could be added to support writable data.

Signed division#

The SBF instruction set does not supportsigned division . Adding a signed division instruction is a consideration.