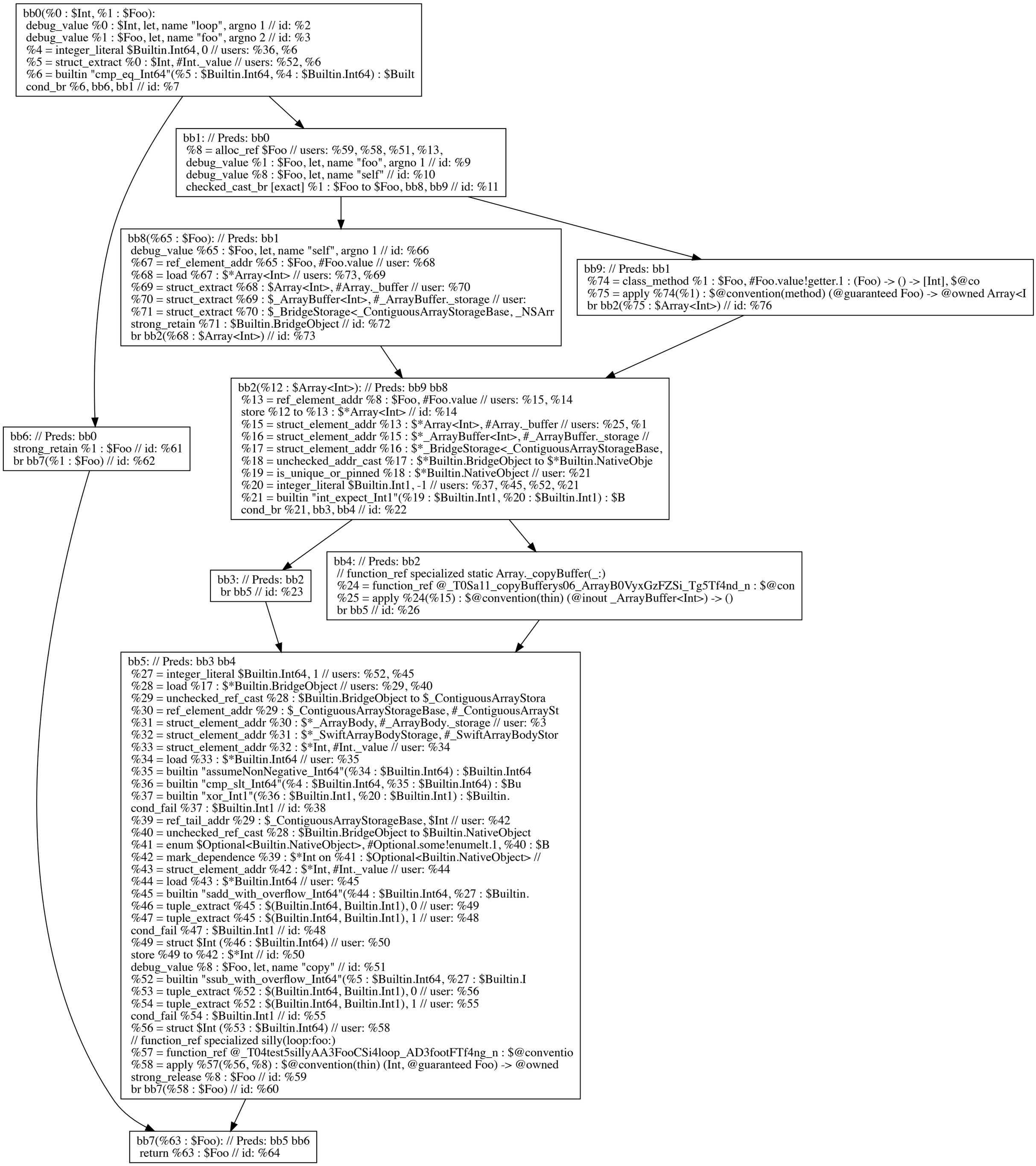
bb0(%0 : $Int, %1 : $Foo):
debug_value %0 : $Int, let, name "loop", argno 1 // id: %2
debug_value %1 : $Foo, let, name "foo", argno 2 // id: %3
%4 = integer_literal $Builtin.Int64, 0 // users: %36, %6
%5 = struct_extract %0 : $Int, #Int._value // users: %52, %6
%6 = builtin "cmp_eq_Int64"(%5 : $Builtin.Int64, %4 : $Builtin.Int64) : $Built
cond_br %6, bb6, bb1 // id: %7
bb1: // Preds: bb0
%8 = alloc_ref $Foo // users: %59, %58, %51, %13,
debug_value %1 : $Foo, let, name "foo", argno 1 // id: %9
debug_value %8 : $Foo, let, name "self" // id: %10
checked_cast_br [exact] %1 : $Foo to $Foo, bb8, bb9 // id: %11
bb8(%65 : $Foo): // Preds: bb1
debug_value %65 : $Foo, let, name "self", argno 1 // id: %66
%67 = ref_element_addr %65 : $Foo, #Foo.value // user: %68
%68 = load %67 : $*Array<Int> // users: %73, %69
%69 = struct_extract %68 : $Array<Int>, #Array._buffer // user: %70
%70 = struct_extract %69 : $_ArrayBuffer<Int>, #_ArrayBuffer._storage // user:
%71 = struct_extract %70 : $_BridgeStorage<_ContiguousArrayStorageBase, _NSArr
strong_retain %71 : $Builtin.BridgeObject // id: %72
br bb2(%68 : $Array<Int>) // id: %73
bb9: // Preds: bb1
%74 = class_method %1 : $Foo, #Foo.value!getter.1 : (Foo) -> () -> [Int], $@co
%75 = apply %74(%1) : $@convention(method) (@guaranteed Foo) -> @owned Array<I
br bb2(%75 : $Array<Int>) // id: %76
bb6: // Preds: bb0
strong_retain %1 : $Foo // id: %61
br bb7(%1 : $Foo) // id: %62
bb2(%12 : $Array<Int>): // Preds: bb9 bb8
%13 = ref_element_addr %8 : $Foo, #Foo.value // users: %15, %14
store %12 to %13 : $*Array<Int> // id: %14
%15 = struct_element_addr %13 : $*Array<Int>, #Array._buffer // users: %25, %1
%16 = struct_element_addr %15 : $*_ArrayBuffer<Int>, #_ArrayBuffer._storage //
%17 = struct_element_addr %16 : $*_BridgeStorage<_ContiguousArrayStorageBase,
%18 = unchecked_addr_cast %17 : $*Builtin.BridgeObject to $*Builtin.NativeObje
%19 = is_unique_or_pinned %18 : $*Builtin.NativeObject // user: %21
%20 = integer_literal $Builtin.Int1, -1 // users: %37, %45, %52, %21
%21 = builtin "int_expect_Int1"(%19 : $Builtin.Int1, %20 : $Builtin.Int1) : $B
cond_br %21, bb3, bb4 // id: %22
bb3: // Preds: bb2
br bb5 // id: %23
bb4: // Preds: bb2
// function_ref specialized static Array._copyBuffer(_:)
%24 = function_ref @_T0Sa11_copyBufferys06_ArrayB0VyxGzFZSi_Tg5Tf4nd_n : $@con
%25 = apply %24(%15) : $@convention(thin) (@inout _ArrayBuffer<Int>) -> ()
br bb5 // id: %26
bb5: // Preds: bb3 bb4
%27 = integer_literal $Builtin.Int64, 1 // users: %52, %45
%28 = load %17 : $*Builtin.BridgeObject // users: %29, %40
%29 = unchecked_ref_cast %28 : $Builtin.BridgeObject to $_ContiguousArrayStora
%30 = ref_element_addr %29 : $_ContiguousArrayStorageBase, #_ContiguousArraySt
%31 = struct_element_addr %30 : $*_ArrayBody, #_ArrayBody._storage // user: %3
%32 = struct_element_addr %31 : $*_SwiftArrayBodyStorage, #_SwiftArrayBodyStor
%33 = struct_element_addr %32 : $*Int, #Int._value // user: %34
%34 = load %33 : $*Builtin.Int64 // user: %35
%35 = builtin "assumeNonNegative_Int64"(%34 : $Builtin.Int64) : $Builtin.Int64
%36 = builtin "cmp_slt_Int64"(%4 : $Builtin.Int64, %35 : $Builtin.Int64) : $Bu
%37 = builtin "xor_Int1"(%36 : $Builtin.Int1, %20 : $Builtin.Int1) : $Builtin.
cond_fail %37 : $Builtin.Int1 // id: %38
%39 = ref_tail_addr %29 : $_ContiguousArrayStorageBase, $Int // user: %42
%40 = unchecked_ref_cast %28 : $Builtin.BridgeObject to $Builtin.NativeObject
%41 = enum $Optional<Builtin.NativeObject>, #Optional.some!enumelt.1, %40 : $B
%42 = mark_dependence %39 : $*Int on %41 : $Optional<Builtin.NativeObject> //
%43 = struct_element_addr %42 : $*Int, #Int._value // user: %44
%44 = load %43 : $*Builtin.Int64 // user: %45
%45 = builtin "sadd_with_overflow_Int64"(%44 : $Builtin.Int64, %27 : $Builtin.
%46 = tuple_extract %45 : $(Builtin.Int64, Builtin.Int1), 0 // user: %49
%47 = tuple_extract %45 : $(Builtin.Int64, Builtin.Int1), 1 // user: %48
cond_fail %47 : $Builtin.Int1 // id: %48
%49 = struct $Int (%46 : $Builtin.Int64) // user: %50
store %49 to %42 : $*Int // id: %50
debug_value %8 : $Foo, let, name "copy" // id: %51
%52 = builtin "ssub_with_overflow_Int64"(%5 : $Builtin.Int64, %27 : $Builtin.I
%53 = tuple_extract %52 : $(Builtin.Int64, Builtin.Int1), 0 // user: %56
%54 = tuple_extract %52 : $(Builtin.Int64, Builtin.Int1), 1 // user: %55
cond_fail %54 : $Builtin.Int1 // id: %55
%56 = struct $Int (%53 : $Builtin.Int64) // user: %58
// function_ref specialized silly(loop:foo:)
%57 = function_ref @_T04test5sillyAA3FooCSi4loop_AD3footFTf4ng_n : $@conventio
%58 = apply %57(%56, %8) : $@convention(thin) (Int, @guaranteed Foo) -> @owned
strong_release %8 : $Foo // id: %59
br bb7(%58 : $Foo) // id: %60
bb7(%63 : $Foo): // Preds: bb5 bb6
return %63 : $Foo // id: %64