A tour of CPython's runtime

Brandt Bucher (October 20th, 2024)

A tour of CPython's runtime

...and how you can speed up every Python process on the planet!

Brandt Bucher (October 20th, 2024)

Brandt Bucher

Brandt Bucher

- 2017: Started using Python.
- 2018: Contributed code to CPython.
- 2019: Joined Python's Triage Team.
- 2020: Joined Python's Core Development Team.
- 2021: Joined Microsoft's CPython Performance Engineering Team.
- 2022: Made CPython 3.11 25% faster!
- 2023: Implemented CPython's new JIT compiler.
- 2024: Working on shipping the new JIT compiler in 3.14!

Microsoft's CPython Performance Engineering Team

The "Faster CPython" Project

The "Faster CPython" Project

- California
- Utah
- Washington
- Maryland
- United Kingdom
- Singapore

The "Faster CPython" Project

- California: Microsoft
- Utah: Microsoft
- Washington
- Maryland: Microsoft
- United Kingdom: Microsoft
- Singapore

The "Faster CPython" Project

- California: Microsoft
- Utah: Microsoft
- Washington: Snowflake
- Maryland: Microsoft
- United Kingdom: Microsoft, Arm
- Singapore: National University of Singapore

The "Faster CPython" Project

- California: Microsoft
- Utah: Microsoft
- Washington: Snowflake
- Maryland: Microsoft
- United Kingdom: Microsoft, Arm
- Singapore: National University of Singapore
- ...?

github.com/faster-cpython/ideas

Python

Python

- 33 years old!
- *Very* high-level
- *Very* widely used
- Dynamic
- Object-oriented
- Interpreted
- Automatic memory management
- Deep introspection and metaprogramming

Python

- 33 years old!
- *Very* high-level
- *Very* widely used
- Dynamic
- Object-oriented
- Interpreted
- Automatic memory management
- Deep introspection and metaprogramming

Python

- 33 years old!
- *Very* high-level
- *Very* widely used
- Dynamic
- Object-oriented
- Interpreted
- Automatic memory management
- Deep introspection and metaprogramming

Python

- Most objects have arbitrary mappings of attributes: `instance.__dict__`.
- Bytecode is a runtime object: `function.__code__`.
- *Frames* are runtime objects: `sys._getframe()`.
- Attribute/global name accesses and assignments can run arbitrary code.
- Even simple operators go through incredibly complex double-dispatching.
- A debugger can be entered *anywhere* and do *anything.*

Python

- Most objects have arbitrary mappings of attributes: `instance.__dict__`.
- Bytecode is a runtime object: `function.__code__`.
- *Frames* are runtime objects: `sys._getframe()`.
- Attribute/global name accesses and assignments can run arbitrary code.
- Even simple operators go through incredibly complex double-dispatching.
- A debugger can be entered *anywhere and do anything*.

Python

- Most objects have arbitrary mappings of attributes: `instance.__dict__`.
- Bytecode is a runtime object: `function.__code__`.
- *Frames* are runtime objects: `sys._getframe()`.
- Attribute/global name accesses and assignments can run arbitrary code.
- Even simple operators go through incredibly complex double-dispatching.
- A debugger can be entered *anywhere and do anything*.

Python

CPython

CPython

- Reference implementation of Python
- Used by the majority of Python programmers
- Reference-counted (augmented with cyclic stop-the-world GC)
- Has an incredibly rich ecosystem of third-party C extensions
- Maintained by a few dozen active "core developers"
- Free and open-source
- github.com/python/cpython

Optimizations

Optimizations

- Build IR
- Check types
- Optimize
- Compile

Optimizations

- Build IR
- Check types
- Optimize
- Compile

Optimizations

- Build IR
- Profile
- Optimize
- Compile

Optimizations

- Constant folding
- Dead code elimination
- Hot/cold splitting
- Jump threading
- Liveness analysis
- Peephole optimizations

- Common subexpression elimination
- Constant promotion
- Constant propagation
- Copy propagation
- Guard elimination
- Inlining
- Loop peeling
- Loop-invariant code motion
- Type propagation

Optimizations

- Constant folding
- Dead code elimination
- Hot/cold splitting
- Jump threading
- Liveness analysis
- Peephole optimizations

- Common subexpression elimination
- Constant promotion
- Constant propagation
- Copy propagation
- Guard elimination
- Inlining
- Loop peeling
- Loop-invariant code motion
- Type propagation

Optimizations

- Constant folding
- Dead code elimination
- Hot/cold splitting
- Jump threading
- Liveness analysis
- Peephole optimizations

- Common subexpression elimination
- Constant promotion
- Constant propagation
- Copy propagation
- Guard elimination
- Inlining
- Loop peeling
- Loop-invariant code motion
- Type propagation

Optimizations

Runtime Optimizations

Runtime Optimizations

```
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        a, b = b, a + b
    return a
```

Runtime Optimizations

```
for _ in range(n):
    a, b = b, a + b
```

Runtime Optimizations

CPython 3.10: Bytecode

```
for _ in range(n):    FOR_ITER
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP
                      STORE_FAST_STORE_FAST
                      JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.10: Bytecode

```
for _ in range(n):    FOR_ITER
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP
                      STORE_FAST_STORE_FAST
                      JUMP_BACKWARD               stack
```

Runtime Optimizations

CPython 3.10: Bytecode

```
for _ in range(n):    FOR_ITER
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP
                      STORE_FAST_STORE_FAST            stack
                      JUMP_BACKWARD                  iterator
```

Runtime Optimizations

CPython 3.10: Bytecode

```
for _ in range(n):     FOR_ITER
    a, b = b, a + b    STORE_FAST
                       LOAD_FAST_LOAD_FAST
                       LOAD_FAST
                       BINARY_OP                        stack
                       STORE_FAST_STORE_FAST     next(iterator)
                       JUMP_BACKWARD                iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):      FOR_ITER
    a, b = b, a + b     STORE_FAST
                        LOAD_FAST_LOAD_FAST
                        LOAD_FAST
                        BINARY_OP                            stack
                        STORE_FAST_STORE_FAST           next(iterator)
                        JUMP_BACKWARD                      iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):      FOR_ITER_RANGE
    a, b = b, a + b     STORE_FAST
                        LOAD_FAST_LOAD_FAST
                        LOAD_FAST
                        BINARY_OP                          stack
                        STORE_FAST_STORE_FAST       next(iterator)
                        JUMP_BACKWARD                  iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):     FOR_ITER_RANGE              _ = next(iterator)
    a, b = b, a + b    STORE_FAST
                       LOAD_FAST_LOAD_FAST
                       LOAD_FAST
                       BINARY_OP
                       STORE_FAST_STORE_FAST              stack
                       JUMP_BACKWARD                    iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE             _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP                          stack
                      STORE_FAST_STORE_FAST                b
                      JUMP_BACKWARD                     iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE            _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST                        stack
                      BINARY_OP                          a
                      STORE_FAST_STORE_FAST              b
                      JUMP_BACKWARD                   iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE            _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST             stack
                      LOAD_FAST                         b
                      BINARY_OP                         a
                      STORE_FAST_STORE_FAST             b
                      JUMP_BACKWARD                 iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE            _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST                           stack
                      BINARY_OP                           a + b
                      STORE_FAST_STORE_FAST                 b
                      JUMP_BACKWARD                     iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE               _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST                          stack
                      BINARY_OP                          a + b
                      STORE_FAST_STORE_FAST                b
                      JUMP_BACKWARD                     iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE              _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST                            stack
                      BINARY_OP_ADD_INT                    a + b
                      STORE_FAST_STORE_FAST                  b
                      JUMP_BACKWARD                       iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE              _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
>>> a = ""            LOAD_FAST                          stack
>>> b = "🐰🐰"         BINARY_OP_ADD_INT                  a + b
                      STORE_FAST_STORE_FAST                b
                      JUMP_BACKWARD                     iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):     FOR_ITER_RANGE               _ = next(iterator)
    a, b = b, a + b    STORE_FAST
                       LOAD_FAST_LOAD_FAST
>>> a = ""             LOAD_FAST                           stack
>>> b = "🐰🐰"          BINARY_OP_ADD_INT                   a + b
                       STORE_FAST_STORE_FAST                 b
                       JUMP_BACKWARD                     iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):       FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b      STORE_FAST
                         LOAD_FAST_LOAD_FAST
>>> a = ""               LOAD_FAST                      stack
>>> b = "🐰🐰"           BINARY_OP_ADD_UNICODE          a + b
                         STORE_FAST_STORE_FAST            b
                         JUMP_BACKWARD                 iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE             _ = next(iterator)
    a, b = b, a + b   STORE_FAST
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST                        stack
                      BINARY_OP_ADD_INT                a + b
                      STORE_FAST_STORE_FAST              b
                      JUMP_BACKWARD                   iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE             _ = next(iterator)
    a, b = b, a + b   STORE_FAST                 b = a + b
                      LOAD_FAST_LOAD_FAST
                      LOAD_FAST
                      BINARY_OP_ADD_INT                 stack
                      STORE_FAST_STORE_FAST               b
                      JUMP_BACKWARD                   iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE          _ = next(iterator)
    a, b = b, a + b   STORE_FAST              b = a + b
                      LOAD_FAST_LOAD_FAST     a = b
                      LOAD_FAST
                      BINARY_OP_ADD_INT
                      STORE_FAST_STORE_FAST         stack
                      JUMP_BACKWARD               iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
for _ in range(n):    FOR_ITER_RANGE           _ = next(iterator)
    a, b = b, a + b   STORE_FAST               b = a + b
                      LOAD_FAST_LOAD_FAST      a = b
                      LOAD_FAST
                      BINARY_OP_ADD_INT
                      STORE_FAST_STORE_FAST          stack
                      JUMP_BACKWARD                 iterator
```

Runtime Optimizations

CPython 3.11: Specialized Bytecode

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
BINARY_OP_ADD_INT
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
macro(BINARY_OP_ADD_INT) = _GUARD_TOS_INT + _GUARD_NOS_INT + _BINARY_OP_ADD_INT;
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
macro(BINARY_OP_ADD_INT) = _GUARD_TOS_INT + _GUARD_NOS_INT + _BINARY_OP_ADD_INT;

op(_GUARD_TOS_INT, (rhs -- rhs)) {
    EXIT_IF(!PyLong_CheckExact(rhs));
}
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
macro(BINARY_OP_ADD_INT) = _GUARD_TOS_INT + _GUARD_NOS_INT + _BINARY_OP_ADD_INT;

op(_GUARD_TOS_INT, (rhs -- rhs)) {
    EXIT_IF(!PyLong_CheckExact(rhs));
}

op(_GUARD_NOS_INT, (lhs, unused -- lhs, unused)) {
    EXIT_IF(!PyLong_CheckExact(lhs));
}
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
macro(BINARY_OP_ADD_INT) = _GUARD_TOS_INT + _GUARD_NOS_INT + _BINARY_OP_ADD_INT;

op(_GUARD_TOS_INT, (rhs -- rhs)) {
    EXIT_IF(!PyLong_CheckExact(rhs));
}

op(_GUARD_NOS_INT, (lhs, unused -- lhs, unused)) {
    EXIT_IF(!PyLong_CheckExact(lhs));
}

op(_BINARY_OP_ADD_INT, (lhs, rhs -- res)) {
    res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
}
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
macro(BINARY_OP_ADD_INT) = _GUARD_TOS_INT + _GUARD_NOS_INT + _BINARY_OP_ADD_INT;

op(_GUARD_TOS_INT, (rhs -- rhs)) {
    EXIT_IF(!PyLong_CheckExact(rhs));
}

op(_GUARD_NOS_INT, (lhs, unused -- lhs, unused)) {
    EXIT_IF(!PyLong_CheckExact(lhs));
}

op(_BINARY_OP_ADD_INT, (lhs, rhs -- res)) {
    res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
}
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
BINARY_OP_ADD_INT  = _GUARD_TOS_INT + _GUARD_NOS_INT + _BINARY_OP_ADD_INT
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
};
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {
                                                   }},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {           }},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {
                                             }},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {           }},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {

                                                         }},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]      = {3, {_GUARD_TOS_INT,
                                    _GUARD_NOS_INT,
                                    _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]         = {3, {_ITER_CHECK_RANGE,
                                    _GUARD_NOT_EXHAUSTED_RANGE,
                                    _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]          = {2, {_CHECK_PERIODIC,
                                    _JUMP_TO_TOP}},
    [LOAD_FAST]              = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]    = {2, {_LOAD_FAST,
                                    _LOAD_FAST}},
    [STORE_FAST]             = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                    _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]      = {3, {_GUARD_TOS_INT,
                                    _GUARD_NOS_INT,
                                    _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]         = {3, {_ITER_CHECK_RANGE,
                                    _GUARD_NOT_EXHAUSTED_RANGE,
                                    _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]          = {2, {_CHECK_PERIODIC,
                                    _JUMP_TO_TOP}},
    [LOAD_FAST]              = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]    = {2, {_LOAD_FAST,
                                    _LOAD_FAST}},
    [STORE_FAST]             = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {
                                              }},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {
                                               }},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

```
_PyOpcode_macro_expansion[256] = {
    [BINARY_OP_ADD_INT]     = {3, {_GUARD_TOS_INT,
                                   _GUARD_NOS_INT,
                                   _BINARY_OP_ADD_INT}},
    [FOR_ITER_RANGE]        = {3, {_ITER_CHECK_RANGE,
                                   _GUARD_NOT_EXHAUSTED_RANGE,
                                   _ITER_NEXT_RANGE}},
    [JUMP_BACKWARD]         = {2, {_CHECK_PERIODIC,
                                   _JUMP_TO_TOP}},
    [LOAD_FAST]             = {1, {_LOAD_FAST}},
    [LOAD_FAST_LOAD_FAST]   = {2, {_LOAD_FAST,
                                   _LOAD_FAST}},
    [STORE_FAST]            = {1, {_STORE_FAST}},
    [STORE_FAST_STORE_FAST] = {2, {_STORE_FAST,
                                   _STORE_FAST}},
};
```

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.13: Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
```

```
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
```

```
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: object<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: int<?>
stack_0: range_iterator<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: int<?>
stack_0: range_iterator<?>

_: object<_>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
```

```
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: object<a>
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: object<a>
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: object<b>
stack_2: object<a>
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: object<b>
stack_2: object<a>
stack_1: object<b>
stack_0: range_iterator<?>

_: int<?>
b: object<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
```

```
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: int<b>
stack_2: object<a>
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<b>
a: object<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
```

```
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: int<b>
stack_2: int<a>
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: int<a+b>
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
```

```
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: int<a+b>
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: int<b>
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<a>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
```

```
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
```

```
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY_AND_SET_IP
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_LOAD_FAST
_CHECK_VALIDITY_AND_SET_IP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_CHECK_VALIDITY_AND_SET_IP
_STORE_FAST
_STORE_FAST
_CHECK_VALIDITY_AND_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_SET_IP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_SET_IP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_GUARD_TOS_INT
_GUARD_NOS_INT
_BINARY_OP_ADD_INT
_SET_IP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_MAKE_WARM
_CHECK_VALIDITY
_ITER_CHECK_RANGE
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_GUARD_FAST_INT
_GUARD_FAST_INT
_BINARY_OP_ADD_INT
_SET_IP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_ITER_CHECK_RANGE
_GUARD_FAST_INT
_GUARD_FAST_INT
_MAKE_WARM
_CHECK_VALIDITY
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
```

```
_BINARY_OP_ADD_INT
_NOP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

```
stack_3: NULL
stack_2: NULL
stack_1: NULL
stack_0: range_iterator<?>

_: int<?>
b: int<a+b>
a: int<b>
n: object<n>
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
_START_EXECUTOR
_ITER_CHECK_RANGE
_GUARD_FAST_INT
_GUARD_FAST_INT
_MAKE_WARM
_CHECK_VALIDITY
_GUARD_NOT_EXHAUSTED_RANGE
_ITER_NEXT_RANGE
_SET_IP
_STORE_FAST
_CHECK_VALIDITY
_LOAD_FAST
_LOAD_FAST
_NOP
_LOAD_FAST
_NOP
_BINARY_OP_ADD_INT
_NOP
_STORE_FAST
_STORE_FAST
_SET_IP
_CHECK_PERIODIC
_JUMP_TO_TOP
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
JUMP_BACKWARD
```

Runtime Optimizations

CPython 3.14: Optimized Micro-Op Traces

```
FOR_ITER_RANGE
STORE_FAST
LOAD_FAST_LOAD_FAST
LOAD_FAST
BINARY_OP_ADD_INT
STORE_FAST_STORE_FAST
ENTER_EXECUTOR
```

Just-In-Time Compilation

Just-In-Time Compilation

- Technical goals:
  - Remove interpretive overhead
  - Statically compile optimized traces
  - Reduce indirection:
    - "Burn in" constants, caches, and arguments
    - Move data off of frames and into registers
    - Bring hot code paths in-line
- Deployment goals:
  - Broad platform support
  - Few runtime dependencies
  - Low implementation complexity

Just-In-Time Compilation

- Technical goals:
  - Remove interpretive overhead
  - Statically compile optimized traces
  - Reduce indirection:
    - "Burn in" constants, caches, and arguments
    - Move data off of frames and into registers
    - Bring hot code paths in-line

- Deployment goals:
  - Broad platform support
  - Few runtime dependencies
  - Low implementation complexity

Just-In-Time Compilation

Copy-And-Patch Compilation

Copy-And-Patch Compilation

- Haoran Xu and Fredrik Kjolstad. 2021. Copy-and-Patch Compilation: A Fast Compilation Algorithm for High- Level Languages and Bytecode. Proc. ACM Program. Lang. 5, OOPSLA, Article 136 (October 2021), 30 pages. https://doi.org/10.1145/3485513
- Haoran Xu. 2023. Building a baseline JIT for Lua automatically. (12 March 2023). Retrieved from https://sillycross.github.io/2023/05/12/2023-05-12/.
- A way of automatically turning a C interpreter into a fast template JIT compiler

Copy-And-Patch Compilation

- Compared to WebAssembly baseline compiler (`Liftoff`):
  - 5x faster code generation
  - 50% faster code

- Compared to traditional JIT toolchain (`LLVM -O0`):
  - 100x faster code generation
  - 15% faster code

- Compared to an optimizing JIT with hand-written assembly (`LuaJIT`):
  - Faster on 13/44 benchmarks
  - Only 35% slower overall

Copy-And-Patch Compilation

- At runtime, walk over a sequence of bytecode instructions.
- For each:
    - Copy some static, pre-compiled machine code into executable memory
    - Patch up instructions that need to have runtime data encoded into them

Copy-And-Patch Compilation

- At runtime, walk over a sequence of bytecode instructions.
- For each:
  - Copy some static, pre-compiled machine code into executable memory
  - Patch up instructions that need to have runtime data encoded into them

Copy-And-Patch Compilation

- Copy some static, pre-compiled machine code into executable memory
- Patch up instructions that need to have runtime data encoded into them

Copy-And-Patch Compilation

- When linking or loading a relocatable object file (ELF, COFF, Mach-O, etc.):
  - Copy some static, pre-compiled machine code into executable memory
  - Patch up instructions that need to have runtime data encoded into them

Copy-And-Patch Compilation

```
case _LOAD_FAST:
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
    break;
```

Copy-And-Patch Compilation

```
PyObject *value = frame->localsplus[oparg];
Py_INCREF(value);
*stack_pointer++ = value;
```

Copy-And-Patch Compilation

```
int
_load_fast(void)
{
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```
int
_load_fast(void)
{
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```
int
_load_fast(_PyInterpreterFrame *frame)
{
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```
int
_load_fast(_PyInterpreterFrame *frame)
{
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```
int
_load_fast(_PyInterpreterFrame *frame)
{
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```
int
_load_fast(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```
int
_load_fast(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```
int
_load_fast(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[oparg];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```
int
_load_fast(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[MAGICALLY_INSERT_OPARG];
    Py_INCREF(value);
    *stack_pointer++ = value;
}
```

Copy-And-Patch Compilation

```c
int
_load_fast(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[MAGICALLY_INSERT_OPARG];
    Py_INCREF(value);
    *stack_pointer++ = value;
    return MAGICALLY_RUN_NEXT_MICRO_OP(frame, stack_pointer);
}
```

Copy-And-Patch Compilation

```
int
_load_fast(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[MAGICALLY_INSERT_OPARG];
    Py_INCREF(value);
    *stack_pointer++ = value;
    return MAGICALLY_RUN_NEXT_MICRO_OP(frame, stack_pointer);
}
```

Copy-And-Patch Compilation

```
int
_load_fast(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[MAGICALLY_INSERT_OPARG];
    Py_INCREF(value);
    *stack_pointer++ = value;
    return MAGICALLY_RUN_NEXT_MICRO_OP(frame, stack_pointer);
}
```

Copy-And-Patch Compilation

```
extern int MAGICALLY_INSERT_OPARG;
extern int MAGICALLY_RUN_NEXT_MICRO_OP(_PyInterpreterFrame *frame,
                                       PyObject **stack_pointer);
int
_load_fast(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[&MAGICALLY_INSERT_OPARG];
    Py_INCREF(value);
    *stack_pointer++ = value;
    __attribute__((musttail))
    return MAGICALLY_RUN_NEXT_MICRO_OP(frame, stack_pointer);
}
```

Copy-And-Patch Compilation

```
extern int MAGICALLY_INSERT_OPARG;
extern int MAGICALLY_RUN_NEXT_MICRO_OP(_PyInterpreterFrame *frame,
                                       PyObject **stack_pointer);
int
_load_fast(_PyInterpreterFrame *frame, PyObject **stack_pointer)
{
    PyObject *value = frame->localsplus[&MAGICALLY_INSERT_OPARG];
    Py_INCREF(value);
    *stack_pointer++ = value;
    __attribute__((musttail))
    return MAGICALLY_RUN_NEXT_MICRO_OP(frame, stack_pointer);
}
```

Copy-And-Patch Compilation

```
0f b7 05 00 00 00 00 movzwl (%rip), %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
ff 25 00 00 00 00    jmpq *(%rip)

03: R_X86_64_GOTPCREL  MAGICALLY_INSERT_THE_OPARG - 0x4
1d: R_X86_64_GOTPCRELX MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
0f b7 05 00 00 00 00 movzwl (%rip), %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
ff 25 00 00 00 00    jmpq *(%rip)

03: R_X86_64_GOTPCREL  MAGICALLY_INSERT_THE_OPARG - 0x4
1d: R_X86_64_GOTPCRELX MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
0f b7 05 00 00 00 00 movzwl (%rip), %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
ff 25 00 00 00 00    jmpq *(%rip)

03: R_X86_64_GOTPCREL  MAGICALLY_INSERT_THE_OPARG - 0x4
1d: R_X86_64_GOTPCRELX MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
0f b7 05 00 00 00 00 movzwl (%rip), %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
ff 25 00 00 00 00    jmpq *(%rip)

03: R_X86_64_GOTPCREL  MAGICALLY_INSERT_THE_OPARG - 0x4
1d: R_X86_64_GOTPCRELX MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
0f b7 05 00 00 00 00 movzwl (%rip), %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
ff 25 00 00 00 00    jmpq *(%rip)

03: R_X86_64_GOTPCREL  MAGICALLY_INSERT_THE_OPARG - 0x4
1d: R_X86_64_GOTPCRELX MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
66 90 b8 00 00 00 00 nop; mov $0x0, %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
ff 25 00 00 00 00    jmpq *(%rip)

03: R_X86_64_32         MAGICALLY_INSERT_THE_OPARG
1d: R_X86_64_GOTPCRELX MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
66 90 b8 00 00 00 00 nop; mov $0x0, %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
ff 25 00 00 00 00    jmpq *(%rip)

03: R_X86_64_32        MAGICALLY_INSERT_THE_OPARG
1d: R_X86_64_GOTPCRELX MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
66 90 b8 00 00 00 00 nop; mov $0x0, %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
ff 25 00 00 00 00    jmpq *(%rip)

03: R_X86_64_32          MAGICALLY_INSERT_THE_OPARG
1d: R_X86_64_GOTPCRELX MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
66 90 b8 00 00 00 00 nop; mov $0x0, %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
e9 00 00 00 00 90    jmp 0x0; nop

03: R_X86_64_32        MAGICALLY_INSERT_THE_OPARG
1c: R_X86_64_PC32      MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
66 90 b8 00 00 00 00 nop; mov $0x0, %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
e9 00 00 00 00 90    jmp 0x0; nop

03: R_X86_64_32        MAGICALLY_INSERT_THE_OPARG
1c: R_X86_64_PC32      MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
66 90 b8 00 00 00 00 nop; mov $0x0, %eax
48 8b 44 c7 48       movq 0x48(%rdi,%rax,8), %rax
8b 08                movl (%rax), %ecx
ff c1                incl %ecx
74 02                je 0x14
89 08                movl %ecx, (%rax)
48 89 06             movq %rax, (%rsi)
48 83 c6 08          addq $0x8, %rsi
e9 00 00 00 00 90    jmp 0x0; nop

03: R_X86_64_32        MAGICALLY_INSERT_THE_OPARG
1c: R_X86_64_PC32      MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
b8 00 00 00 00          mov $0x0, %eax
48 8b 44 c7 48          movq 0x48(%rdi,%rax,8), %rax
8b 08                   movl (%rax), %ecx
ff c1                   incl %ecx
74 02                   je 0x12
89 08                   movl %ecx, (%rax)
48 89 06                movq %rax, (%rsi)
48 83 c6 08             addq $0x8, %rsi
e9 00 00 00 00          jmp 0x0

01: R_X86_64_32           MAGICALLY_INSERT_THE_OPARG
1a: R_X86_64_PC32         MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
b8 00 00 00 00        mov $0x0, %eax
48 8b 44 c7 48        movq 0x48(%rdi,%rax,8), %rax
8b 08                 movl (%rax), %ecx
ff c1                 incl %ecx
74 02                 je 0x12
89 08                 movl %ecx, (%rax)
48 89 06              movq %rax, (%rsi)
48 83 c6 08           addq $0x8, %rsi
e9 00 00 00 00        jmp 0x0

01: R_X86_64_32          MAGICALLY_INSERT_THE_OPARG
1a: R_X86_64_PC32        MAGICALLY_CONTINUE_EXECUTION - 0x4
```

Copy-And-Patch Compilation

```
b8 00 00 00 00        mov $0x0, %eax
48 8b 44 c7 48        movq 0x48(%rdi,%rax,8), %rax
8b 08                 movl (%rax), %ecx
ff c1                 incl %ecx
74 02                 je 0x12
89 08                 movl %ecx, (%rax)
48 89 06              movq %rax, (%rsi)
48 83 c6 08           addq $0x8, %rsi


01: R_X86_64_32         MAGICALLY_INSERT_THE_OPARG
```

Copy-And-Patch Compilation

```
void
emit__LOAD_FAST(unsigned char *code, _PyUOpInstruction *uop)
{
    const unsigned char code_body[25] = {
          b8,   00,   00,   00,   00,   48,   8b,   44,
          c7,   48,   8b,   08,   ff,   c1,   74,   02,
          89,   08,   48,   89,   06,   48,   83,   c6,
          08,
    };
    memcpy(code, code_body, sizeof(code_body));
    patch_32(code + 0x1, uop->oparg);
}
```

Copy-And-Patch Compilation

```
void
emit__LOAD_FAST(unsigned char *code, _PyUOpInstruction *uop)
{
    const unsigned char code_body[25] = {
        0xb8, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8b, 0x44,
        0xc7, 0x48, 0x8b, 0x08, 0xff, 0xc1, 0x74, 0x02,
        0x89, 0x08, 0x48, 0x89, 0x06, 0x48, 0x83, 0xc6,
        0x08,
    };
    memcpy(code, code_body, sizeof(code_body));
    patch_32(code + 0x1, uop->oparg);
}
```

Platform Support

Platform Support

Platform Support

x86-64

- `x86_64-apple-darwin/clang`
- `x86_64-pc-windows-msvc/msvc`
- `x86_64-unknown-linux-gnu/clang`
- `x86_64-unknown-linux-gnu/gcc`

Platform Support

x86 and x86-64

- `i686-pc-windows-msvc/msvc`
- `x86_64-apple-darwin/clang`
- `x86_64-pc-windows-msvc/msvc`
- `x86_64-unknown-linux-gnu/clang`
- `x86_64-unknown-linux-gnu/gcc`

Platform Support

AArch64, x86, and x86-64

- `aarch64-apple-darwin/clang`
- `aarch64-pc-windows-msvc/msvc`
- `aarch64-unknown-linux-gnu/clang`
- `aarch64-unknown-linux-gnu/gcc`
- `i686-pc-windows-msvc/msvc`
- `x86_64-apple-darwin/clang`
- `x86_64-pc-windows-msvc/msvc`
- `x86_64-unknown-linux-gnu/clang`
- `x86_64-unknown-linux-gnu/gcc`

Platform Support

AArch64, x86, and x86-64

- `aarch64-apple-darwin/clang`
- `aarch64-pc-windows-msvc/msvc`
- `aarch64-unknown-linux-gnu/clang`
- `aarch64-unknown-linux-gnu/gcc`
- `i686-pc-windows-msvc/msvc`
- `x86_64-apple-darwin/clang`
- `x86_64-pc-windows-msvc/msvc`
- `x86_64-unknown-linux-gnu/clang`
- `x86_64-unknown-linux-gnu/gcc`

Results

Results (so far...)

Results
(so far...)

- Build time:
  - ~1100 lines of complex Python
  - ~100 lines of complex C
  - LLVM dependency

- Run time:
  - ~400 lines of simple C (hand-written)
  - ~4000 lines of simple C (generated)
  - No dependencies

Results

(so far...)

- Build time:
  - ~1100 lines of complex Python
  - ~100 lines of complex C
  - LLVM dependency

- Run time:
  - ~400 lines of simple-*ish* C (hand-written)
  - ~4000 lines of simple C (generated)
  - No dependencies

Results

(so far...)

- Build time:
  - ~1100 lines of complex Python
  - ~100 lines of complex C
  - LLVM dependency

- Run time:
  - ~400 lines of simple-*ish* C (hand-written)
  - ~4000 lines of simple C (generated)
  - No dependencies

Results

(so far...)

- ~20% slowdown with the micro-op interpreter enabled

Results

(so far...)

- ~~~20%~~ ~0% slowdown with the JIT enabled

Results

(so far...)

- ~~~20%~~ ~0% slowdown with the JIT enabled
- ~6% of the benchmark code is run in the JIT

Results

(so far...)

- ~~~20%~~ ~0% slowdown with the JIT enabled
- ~~~6%~~ ~65% of the benchmark code is run in the JIT

Results

(so far...)

- ~~~20%~~ ~0% slowdown with the JIT enabled
- ~~~6%~~ ~~~65%~~ ~91% of the benchmark code is run in the JIT

Results
(so far...)

- ~~~20%~~ ~0% slowdown with the JIT enabled
- ~~~6%~~ ~~~65%~~ ~91% of the benchmark code is run in the JIT
- ~10% increase in total memory with the JIT enabled

Results

(so far...)

- ~~~20%~~ ~0% slowdown with the JIT enabled
- ~~~6%~~ ~~~65%~~ ~91% of the benchmark code is run in the JIT
- ~~~10%~~ ~6% increase in total memory with the JIT enabled

Future Work

Future Work

Stack Caching

- M. Anton Ertl. 1995. Stack caching for interpreters. In Proceedings of the ACM SIGPLAN 1995 conference on Programming language design and implementation (PLDI '95). Association for Computing Machinery, New York, NY, USA, 315–327. https://doi.org/10.1145/207110.207165

Future Work

Stack Caching

```
int
_binary_op_add_int(PyThreadState *tstate, _PyInterpreterFrame *frame,
                   PyObject **stack_pointer)
{
    PyObject *lhs = stack_pointer[-2];
    PyObject *rhs = stack_pointer[-1];
    PyObject *res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
    stack_pointer[-2] = res;
    stack_pointer -= 1;
    __attribute__((musttail))
    return MAGIC_CONTINUATION(tstate, frame, stack_pointer);
}
```

Future Work

Stack Caching

```
int
_binary_op_add_int(PyThreadState *tstate, _PyInterpreterFrame *frame,
                   PyObject **stack_pointer)
{
    PyObject *lhs = stack_pointer[-2];
    PyObject *rhs = stack_pointer[-1];
    PyObject *res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
    stack_pointer[-2] = res;
    stack_pointer -= 1;
    __attribute__((musttail))
    return MAGIC_CONTINUATION(tstate, frame, stack_pointer);
}
```

Future Work

Stack Caching

```
int
_binary_op_add_int(PyThreadState *tstate, _PyInterpreterFrame *frame,
                   PyObject **stack_pointer, PyObject *tos)
{
    PyObject *lhs = stack_pointer[-2];
    PyObject *rhs = stack_pointer[-1];
    PyObject *res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
    stack_pointer[-2] = res;
    stack_pointer -= 1;
    __attribute__((musttail))
    return MAGIC_CONTINUATION(tstate, frame, stack_pointer, tos);
}
```

Future Work

Stack Caching

```
int
_binary_op_add_int(PyThreadState *tstate, _PyInterpreterFrame *frame,
                   PyObject **stack_pointer, PyObject *empty)
{
    PyObject *lhs = stack_pointer[-2];
    PyObject *rhs = stack_pointer[-1];
    PyObject *res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
    stack_pointer -= 2;
    __attribute__((musttail))
    return MAGIC_CONTINUATION(tstate, frame, stack_pointer, res);
}
```

Future Work

Stack Caching

```
int
_binary_op_add_int(PyThreadState *tstate, _PyInterpreterFrame *frame,
                   PyObject **stack_pointer, PyObject *rhs)
{
    PyObject *lhs = stack_pointer[-1];
    PyObject *res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
    stack_pointer -= 1;
    __attribute__((musttail))
    return MAGIC_CONTINUATION(tstate, frame, stack_pointer, res);
}
```

Future Work

Stack Caching

```
int
_binary_op_add_int(PyThreadState *tstate, _PyInterpreterFrame *frame,
                   PyObject **stack_pointer, PyObject *lhs, PyObject *rhs)
{
    PyObject *res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
    __attribute__((musttail))
    return MAGIC_CONTINUATION(tstate, frame, stack_pointer, res, JUNK);
}
```

Future Work

Stack Caching

```
__attribute__((preserve_none)) int
_binary_op_add_int(PyThreadState *tstate, _PyInterpreterFrame *frame,
                   PyObject **stack_pointer, PyObject *lhs, PyObject *rhs)
{
    PyObject *res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
    __attribute__((musttail))
    return MAGIC_CONTINUATION(tstate, frame, stack_pointer, res, JUNK);
}
```

Future Work

Stack Caching

```
__attribute__((preserve_none)) int
_binary_op_add_int(PyThreadState *tstate, _PyInterpreterFrame *frame,
                   PyObject **stack_pointer, PyObject *lhs, PyObject *rhs,
                   PyObject *_g, PyObject *_f, PyObject *_e, PyObject *_d,
                   PyObject *_c, PyObject *_b, PyObject *_a)
{
    PyObject *res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
    __attribute__((musttail))
    return MAGIC_CONTINUATION(tstate, frame, stack_pointer, JUNK, res, _g,
                              _f, _e, _d, _c, _b, _a);
}
```

Future Work

Stack Caching

```
__attribute__((preserve_none)) int
_binary_op_add_int(PyThreadState *tstate, _PyInterpreterFrame *frame,
                   PyObject **stack_pointer, PyObject *rhs, PyObject *lhs, PyObject *_s,
                   PyObject *_r, PyObject *_q, PyObject *_p, PyObject *_o, PyObject *_n,
                   PyObject *_m, PyObject *_l, PyObject *_k, PyObject *_j, PyObject *_i,
                   PyObject *_h, PyObject *_g, PyObject *_f, PyObject *_e, PyObject *_d,
                   PyObject *_c, PyObject *_b, PyObject *_a)
{
    PyObject *res = _PyLong_Add(lhs, rhs);
    ERROR_IF(res == NULL);
    Py_DECREF(lhs);
    Py_DECREF(rhs);
     __attribute__((musttail))
    return MAGIC_CONTINUATION(tstate, frame, stack_pointer, JUNK, res, _s, _r, _q, _p,
                              _o, _n, _m, _l, _k, _j, _i, _h, _g, _f, _e, _d, _c, _b,
                              _a);
}
```

Future Work

Trace Stitching

- Andreas Gal, Brendan Eich, Mike Shaver, David Anderson, David Mandelin, Mohammad R. Haghighat, Blake Kaplan, Graydon Hoare, Boris Zbarsky, Jason Orendorff, Jesse Ruderman, Edwin W. Smith, Rick Reitmaier, Michael Bebenita, Mason Chang, and Michael Franz. 2009. Trace-based just-in-time type specialization for dynamic languages. In Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '09). Association for Computing Machinery, New York, NY, USA, 465–478. https://doi.org/10.1145/1542476.1542528

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    s = ""
    if not i % 3: bail()
    if not i % 5: bail()
    if s: bail()
    print(i)
    goto T0
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    s = ""
    if not i % 3: bail()
    if not i % 5: bail()
    print(i)
    goto T0
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    s = ""
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: bail()
T2: bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: s = ""
    bail()
T2: s = ""
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: s = ""
    bail()
T2: s = ""
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: s = ""
    s += "fizz"
    if not i % 5: bail()
    if s: print(s)
    else: bail()
    goto T0
T2: s = ""
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: s = "fizz"
    if not i % 5: bail()
    if s: print(s)
    else: bail()
    goto T0
T2: s = ""
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: s = "fizz"
    if not i % 5: bail()
    print("fizz")
    goto T0
T2: s = ""
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: s = "fizz"
    if not i % 5: goto T3
    print("fizz")
    goto T0
T2: s = ""
    bail()
T3: bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: s = ""
    bail()
T3: s = "fizz"
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: s = ""
    bail()
T3: s = "fizz"
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: s = ""
    s += "buzz"
    if s: print(s)
    else: print(i)
    goto T0
T3: s = "fizz"
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: s = "buzz"
    if s: print(s)
    else: print(i)
    goto T0
T3: s = "fizz"
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: s = "buzz"
    print("buzz")
    goto T0
T3: s = "fizz"
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: print("buzz")
    goto T0
T3: s = "fizz"
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: print("buzz")
    goto T0
T3: s = "fizz"
    bail()
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: print("buzz")
    goto T0
T3: s = "fizz"
    s += "buzz"
    if s: print(s)
    else: print(i)
    goto T0
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: print("buzz")
    goto T0
T3: s = "fizzbuzz"
    if s: print(s)
    else: print(i)
    goto T0
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: print("buzz")
    goto T0
T3: s = "fizzbuzz"
    print("fizzbuzz")
    goto T0
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: print("buzz")
    goto T0
T3: print("fizzbuzz")
    goto T0
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: print("buzz")
    goto T0
T3: print("fizzbuzz")
    goto T0
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
...
```

```
T0: if exhausted(iterator): bail()
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: print("buzz")
    goto T0
T3: print("fizzbuzz")
    goto T0
```

Future Work

Trace Stitching

```
for i in range(n):
    s = ""
    if not i % 3: s += "fizz"
    if not i % 5: s += "buzz"
    if s: print(s)
    else: print(i)
...
```

```
T0: if exhausted(iterator): goto T4
    i = next(iterator)
    if not i % 3: goto T1
    if not i % 5: goto T2
    print(i)
    goto T0
T1: if not i % 5: goto T3
    print("fizz")
    goto T0
T2: print("buzz")
    goto T0
T3: print("fizzbuzz")
    goto T0
T4: ...
```

Future Work

Trace Stitching

- Works for:
  - ...explicit control flow.
  - ...polymorphic code.
  - ...pretty much any reason we might branch in JIT code.

Other Projects

Other Projects

- Better benchmarks, with more emphasis on modern idioms.
- Reduced reference counting overhead.
- Improving the object model.
- True function inlining.
- Integer unboxing.
- Incremental GC.
- Subinterpreters.
- Free-threading.
- ...?

Thank you!

@brandtbucher

Thank you!

@brandtbucher | brandt@python.org

Thank you!

@brandtbucher | brandt@python.org | https://xkcd.com/451