1 `main` — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```rust
fn main() {
    let a = 42;
    let b = 3 + 39;

    assert_eq!(b, a);
}
```

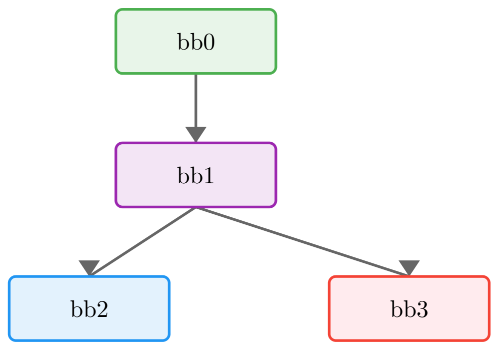
1.2 Function Overview

- **Function:** `main`
- **Basic blocks:** 4
- **Return type:** `()`
- **Notable properties:**
  - Contains panic path
  - Uses checked arithmetic
  - Introduces borrows
  - Contains assertions
  - Has conditional branches

1.3 Locals

Local	Type	Notes
0	`()`	Return place
1	`i32`	
2	`i32`	
3	`(i32, bool)`	
4	`(&i32, &i32)`	
5	`&i32`	
6	`&i32`	
7	`&i32`	
8	`&i32`	
9	`bool`	
10	`i32`	
11	`i32`	
12	`core::panicking::AssertKind`	
13	`!`	
14	`std::option::Option<std::fmt::Arguments<'_>>`	

1.4 Control-Flow Overview



$$\begin{tikzcd} & bb0 \arrow[d] & \\ & bb1 \arrow[dl] \arrow[dr] & \\ bb2 & & bb3 \end{tikzcd}$$

1.5 Basic Blocks

1.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
`_1 = 42`	Load constant
`_3 = checked(3 + 39)`	Checked Add (may panic)
`→ assert(move _3.1 == false) → bb1`	Panic if move _3.1 is true

1.5.2 bb1 — branch point

MIR	Annotation
`_2 = move _3.0`	Move value
`_5 = &_2`	Shared borrow
`_6 = &_1`	Shared borrow
`_4 = Tuple(move _5, move _6)`	Construct aggregate
`_7 = _4.0`	Copy value
`_8 = _4.1`	Copy value
`_10 = (*_7)`	Copy value
`_11 = (*_8)`	Copy value
`_9 = move _10 == move _11`	Equal operation
`→ switch(move _9) [0→bb3; else→bb2]`	Branch on move _9

1.5.3 bb2 — return / success

Normal return path.

MIR	Annotation
`→ return`	Return from function

1.5.4 bb3 — panic path

Panic/diverging path.

MIR	Annotation
`_12 = AssertKind::Eq()`	Construct aggregate
`_14 = Option::None()`	Construct aggregate
`→ _13 = assert_failed(move _12, _7, _8, move _14)`	Call assert_failed

1.6 Key Observations

TODO: Add bullet points summarizing what this MIR teaches

-
-

1.7 Takeaways

TODO: One or two sentences to generalize this example