1 `main` — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```
fn main() {
    let tup:(i32, i32) = (42, 99);

    assert!(tup == (42, 99));
}
```

1.2 Function Overview

- **Function:** `main`
- **Basic blocks:** 4
- **Return type:** `() (0 bytes, align 1)`
- **Notable properties:**
  - Contains panic path
  - Introduces borrows
  - Has conditional branches

1.3 Locals

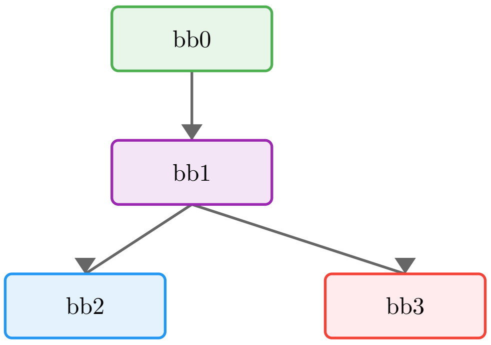
Local	Type	Notes
0	`() (0 bytes, align 1)`	Return place
1	`(i32, i32) (8 bytes, align 4)`	
2	`Bool`	
3	`&(i32, i32) (8 bytes, align 8)`	
4	`&(i32, i32) (8 bytes, align 8)`	
5	`()`	

1.4 Borrows

#	Borrow	Kind	Created At	Borrowed
0	`_3`	`&`	`bb0[1]`	`_1`

Borrows are tracked conservatively: active from creation until reassignment or scope end.

1.5 Control-Flow Overview



1.6 Basic Blocks

1.6.1 bb0 — entry

Entry point of the function.

MIR	Annotation
`_1 = Tuple(42, 99)`	Construct aggregate
`_3 = &_1`	Shared borrow
`_4 = 0`	Load constant
`→ _2 = eq(move _3, move _4) → bb1`	Call eq

1.6.2 bb1 — branch point

MIR	Annotation
`→ switch(move _2) [0→bb3; else→bb2]`	Branch on move _2

1.6.3 bb2 — return / success

Normal return path.

MIR	Annotation
`→ return`	Return from function

1.6.4 bb3 — panic path

Panic/diverging path.

MIR	Annotation
`→ _5 = panic([16 bytes])`	Call panic

1.7 Key Observations

TODO: Add bullet points summarizing what this MIR teaches

-
-

1.8 Takeaways

TODO: One or two sentences to generalize this example