1 `main` — MIR Walkthrough

> **Purpose:** TODO: Describe why this walkthrough exists

1.1 Source Context

```rust
fn main() {
    let a = [1, 2, 3, 4];

    let b = &a[1..3];

    assert!(b == [2, 3]);
}
```
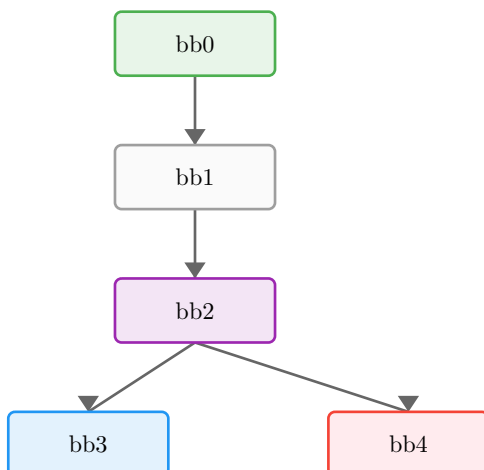
1.2 Function Overview

- **Function:** `main`
- **Basic blocks:** 5
- **Return type:** `()`
- **Notable properties:**
  ‣ Contains panic path
  ‣ Introduces borrows
  ‣ Has conditional branches

1.3 Locals

Local	Type	Notes
0	`()`	Return place
1	`\[i32; 4\]`	
2	`&\[i32\]`	
3	`&\[i32\]`	
4	`&\[i32; 4\]`	
5	`std::ops::Range\<usize\>`	
6	`bool`	
7	`&&\[i32\]`	
8	`&\[i32; 2\]`	
9	`!`	

1.4 Control-Flow Overview

1.5 Basic Blocks

1.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
_1 = Array(1, 2, 3, 4)	Construct aggregate
_4 = &_1	Shared borrow
_5 = Range(1, 3)	Construct aggregate
→ _3 = index(move _4, move _5) → bb1	Call index

1.5.2 bb1

MIR	Annotation
_2 = _3	Copy value
_7 = &_2	Shared borrow
_8 = 0	Load constant
→ _6 = eq(move _7, move _8) → bb2	Call eq

1.5.3 bb2 — branch point

MIR	Annotation
→ switch(move _6) \[0→bb4; else→bb3\]	Branch on move _6

1.5.4 bb3 — return / success

Normal return path.

MIR	Annotation
→ return	Return from function

1.5.5 bb4 — panic path

Panic/diverging path.

MIR	Annotation
→ _9 = panic(\[16 bytes\])	Call panic

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