1 `main` — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```rust
fn main() {
    let bytes: [u8; 8] = [0x15, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00];
    let opt: Option<[u8; 8]> = Some(bytes);
    let result = opt.map(u64::from_le_bytes);
    assert_eq!(result, Some(21u64));
}
```

1.2 Function Overview

- **Function:** `main`
- **Basic blocks:** 5
- **Return type:** `() (0 bytes, align 1)`
- **Notable properties:**
  - Contains panic path
  - Introduces borrows
  - Has conditional branches

1.3 Locals

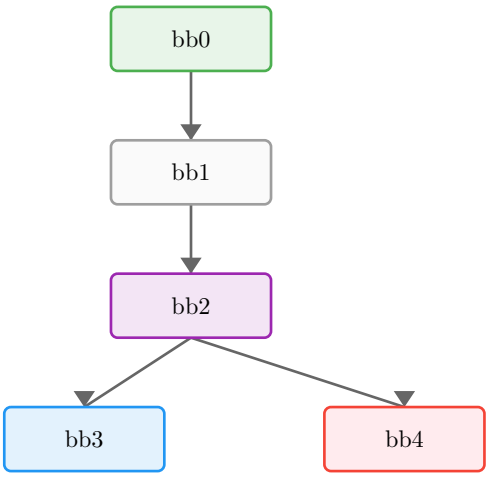
Local	Type	Notes
0	`() (0 bytes, align 1)`	Return place
1	`[u8; 8] (8 bytes, align 1)`	
2	`std::option::Option<[u8; 8]> (9 bytes, align 1)`	
3	`std::option::Option<u64> (16 bytes, align 8)`	
4	`(&std::option::Option<u64>, &std::option::Option<u64>) (16 bytes, align 8)`	
5	`&std::option::Option<u64> (8 bytes, align 8)`	
6	`&std::option::Option<u64> (8 bytes, align 8)`	
7	`&std::option::Option<u64> (8 bytes, align 8)`	
8	`&std::option::Option<u64> (8 bytes, align 8)`	
9	`Bool`	
10	`core::panicking::AssertKind (1 bytes, align 1)`	
11	`()`	
12	`std::option::Option<std::fmt::Arguments<'_>> (48 bytes, align 8)`	

1.4 Borrows

#	Borrow	Kind	Created At	Borrowed
0	_5	&	bb1[0]	_3

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 = Array(21, 0, 0, 0, 0, 0, 0, 0)`	Construct aggregate
`_2 = Option::Some(_1)`	Construct aggregate
`→ _3 = map(_2, ()) → bb1`	Call map

1.6.2 bb1

MIR	Annotation
`_5 = &_3`	Shared borrow
`_6 = 0`	Load constant
`_4 = Tuple(move _5, move _6)`	Construct aggregate
`_7 = _4.0`	Copy value
`_8 = _4.1`	Copy value
`→ _9 = eq(_7, _8) → bb2`	Call eq

1.6.3 bb2 — branch point

MIR	Annotation
`→ switch(move _9) [0→bb4; else→bb3]`	Branch on move _9

1.6.4 bb3 — return / success

Normal return path.

MIR	Annotation
`→ return`	Return from function

1.6.5 bb4 — panic path

Panic/diverging path.

MIR	Annotation

`_10 = AssertKind::Eq()`	Construct aggregate
`_12 = Option::None()`	Construct aggregate
`→ _11 = assert_failed(move _10, _7, _8, move _12)`	Call assert_failed

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