1 `main` — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```rust
fn main() {
    let a: WithParam<u32> = WithParam{the_t: 42, another: 42};

    let b: WithParam<u64> = WithParam{the_t: 42, another: 42};

    let c: Result<u8, usize> = Err(a.another);
    let d : Result<u64, u8> = Ok(b.the_t);

    let x = c.err().unwrap();

    assert!(x as u64 == d.unwrap());
}
```

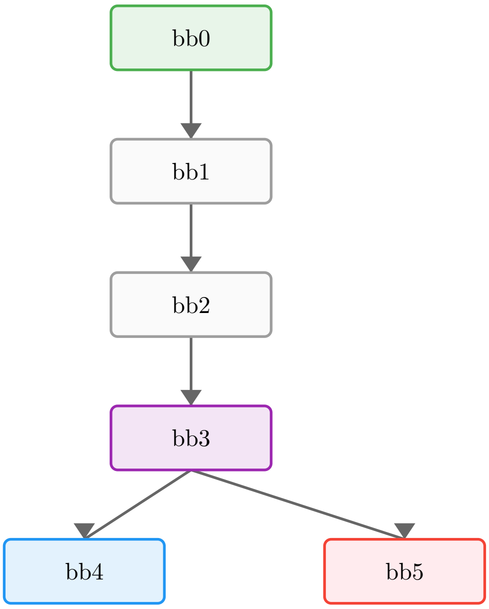
1.2 Function Overview

- **Function:** `main`
- **Basic blocks:** 6
- **Return type:** `()`
- **Notable properties:**
  - Contains panic path
  - Has conditional branches

1.3 Locals

Local	Type	Notes
0	`()`	Return place
1	`WithParam\<u32\>`	
2	`WithParam\<u64\>`	
3	`std::result::Result\<u8, usize\>`	
4	`usize`	
5	`std::result::Result\<u64, u8\>`	
6	`u64`	
7	`usize`	
8	`std::option::Option\<usize\>`	
9	`bool`	
10	`u64`	
11	`u64`	
12	`!`	

1.4 Control-Flow Overview



1.5 Basic Blocks

1.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
`\_1 = WithParam(42, 42)`	Construct aggregate
`\_2 = WithParam(42, 42)`	Construct aggregate
`\_4 = \_1.1`	Copy value
`\_3 = Result::Err(move \_4)`	Construct aggregate
`\_6 = \_2.0`	Copy value
`\_5 = Result::Ok(move \_6)`	Construct aggregate
`→ \_8 = err(\_3) → bb1`	Call err

1.5.2 bb1

MIR	Annotation
`→ \_7 = unwrap(move \_8) → bb2`	Call unwrap

1.5.3 bb2

MIR	Annotation
`\_10 = \_7 as RigidTy(Uint(U64))`	Integer conversion
`→ \_11 = unwrap(\_5) → bb3`	Call unwrap

1.5.4 bb3 — branch point

MIR	Annotation
`\_9 = move \_10 == move \_11`	Equal operation
`→ switch(move \_9) \[0→bb5; else→bb4\]`	Branch on move _9

1.5.5 bb4 — return / success

Normal return path.

MIR	Annotation
`→ return`	Return from function

1.5.6 bb5 — panic path

Panic/diverging path.

MIR	Annotation
`→ \_12 = 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