1 `main` — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```
    assert!(a + b == 4.7);

    let c:f64 = 3.5;
    let d:f64 = 1.2;

    assert!(c + d == 4.7);
}
```

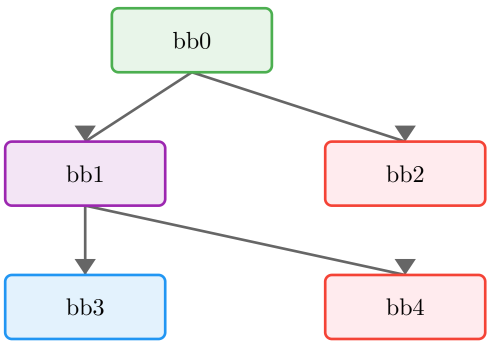
1.2 Function Overview

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

1.3 Locals

Local	Type	Notes
0	`()`	Return place
1	`bool`	
2	`f32`	
3	`f32`	
4	`f32`	
5	`!`	
6	`bool`	
7	`f64`	
8	`f64`	
9	`f64`	
10	`!`	

1.4 Control-Flow Overview



1.5 Basic Blocks

1.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
`\_3 = 1080033280`	Load constant
`\_4 = 1067030938`	Load constant
`\_2 = move \_3 + move \_4`	Add operation
`\_1 = move \_2 == 1083598438`	Equal operation
`→ switch(move \_1) \[0→bb2; else→bb1\]`	Branch on move _1

1.5.2 bb1 — branch point

MIR	Annotation
`\_8 = 4615063718147915776`	Load constant
`\_9 = 4608083138725491507`	Load constant
`\_7 = move \_8 + move \_9`	Add operation
`\_6 = move \_7 == 4616977747989548237`	Equal operation
`→ switch(move \_6) \[0→bb4; else→bb3\]`	Branch on move _6

1.5.3 bb2 — panic path

Panic/diverging path.

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

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
`→ \_10 = 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