

1 main — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```
fn main() {  
    let ans = is_even(10);  
  
    assert!(ans == true);  
}
```

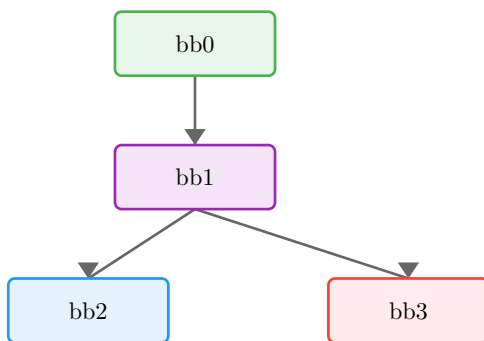
1.2 Function Overview

- **Function:** main
- **Basic blocks:** 4
- **Return type:** ()
- **Notable properties:**
 - Contains panic path
 - Has conditional branches

1.3 Locals

Local	Type	Notes
0	()	Return place
1	bool	
2	!	

1.4 Control-Flow Overview



1.5 Basic Blocks

1.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
→ <code>_1 = is_even(10) → bb1</code>	Call <code>is_even</code>

1.5.2 bb1 — branch point

MIR	Annotation
→ <code>switch(_1) \[0→bb3; else→bb2\]</code>	Branch on <code>_1</code>

1.5.3 bb2 — return / success

Normal return path.

MIR	Annotation
→ return	Return from function

1.5.4 bb3 — panic path

Panic/diverging path.

MIR	Annotation
→ _2 = 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

2 is_odd — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

2.1 Source Context

```
fn is_odd(n:u32) -> bool {  
    if n == 0 {  
        false  
    } else {  
        is_even(n - 1)  
    }  
}
```

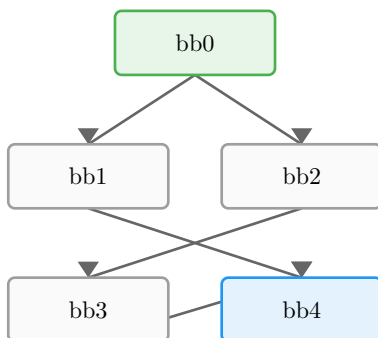
2.2 Function Overview

- **Function:** is_odd
- **Basic blocks:** 5
- **Return type:** bool
- **Notable properties:**
 - Contains panic path
 - Uses checked arithmetic
 - Contains assertions
 - Has conditional branches

2.3 Locals

Local	Type	Notes
0	bool	Return place
1	u32	
2	u32	
3	(u32, bool)	

2.4 Control-Flow Overview



2.5 Basic Blocks

2.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
→ switch(_1) \[0→bb1; else→bb2\]	Branch on _1

2.5.2 bb1

MIR	Annotation
<code>_0 = 0</code>	Load constant
<code>→ goto bb4</code>	Jump to bb4

2.5.3 bb2

MIR	Annotation
<code>_3 = checked(_1 - 1)</code>	Checked Subtract (may panic)
<code>→ assert(move _3.1 == false) → bb3</code>	Panic if move _3.1 is true

2.5.4 bb3

MIR	Annotation
<code>_2 = move _3.0</code>	Move value
<code>→ _0 = is_even(move _2) → bb4</code>	Call is_even

2.5.5 bb4 — return / success

Normal return path.

MIR	Annotation
<code>→ return</code>	Return from function

2.6 Key Observations

TODO: Add bullet points summarizing what this MIR teaches

-
-

2.7 Takeaways

TODO: One or two sentences to generalize this example

3 is_even — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

3.1 Source Context

```
fn is_even(n:u32) -> bool {  
    if n == 0 {  
        true  
    } else {  
        is_odd(n - 1)  
    }  
}
```

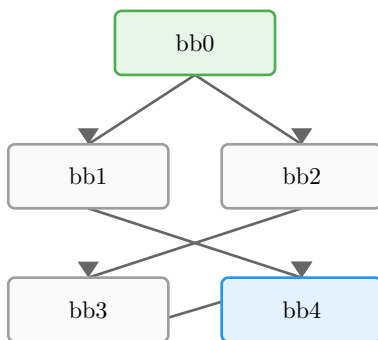
3.2 Function Overview

- **Function:** is_even
- **Basic blocks:** 5
- **Return type:** bool
- **Notable properties:**
 - Contains panic path
 - Uses checked arithmetic
 - Contains assertions
 - Has conditional branches

3.3 Locals

Local	Type	Notes
0	bool	Return place
1	u32	
2	u32	
3	(u32, bool)	

3.4 Control-Flow Overview



3.5 Basic Blocks

3.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
→ switch(_1) \[0→bb1; else→bb2\]	Branch on _1

3.5.2 bb1

MIR	Annotation
<code>_0 = 1</code>	Load constant
<code>→ goto bb4</code>	Jump to bb4

3.5.3 bb2

MIR	Annotation
<code>_3 = checked(_1 - 1)</code>	Checked Subtract (may panic)
<code>→ assert(move _3.1 == false) → bb3</code>	Panic if move _3.1 is true

3.5.4 bb3

MIR	Annotation
<code>_2 = move _3.0</code>	Move value
<code>→ _0 = is_odd(move _2) → bb4</code>	Call is_odd

3.5.5 bb4 — return / success

Normal return path.

MIR	Annotation
<code>→ return</code>	Return from function

3.6 Key Observations

TODO: Add bullet points summarizing what this MIR teaches

-
-

3.7 Takeaways

TODO: One or two sentences to generalize this example

