

1 main — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```
let z:bool = test(x, y);
assert!(z);
return ();
}
```

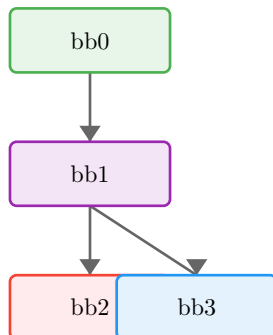
1.2 Function Overview

- **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	usize	
3	usize	
4	!	

1.4 Control-Flow Overview



1.5 Basic Blocks

1.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
_2 = 42	Load constant
_3 = 0	Load constant
→ _1 = test(move _2, move _3) → bb1	Call test

1.5.2 bb1 — branch point

MIR	Annotation
→ switch(_1) [0→bb2; else→bb3]	Branch on _1

1.5.3 bb2 — panic path

Panic/diverging path.

MIR	Annotation
→ _4 = panic([16 bytes])	Call panic

1.5.4 bb3 — return / success

Normal return path.

MIR	Annotation
→ return	Return from function

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 test — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

2.1 Source Context

```
fn test(x: usize, y:usize) -> bool {  
    return x > y;  
}
```

2.2 Function Overview

- **Function:** test
- **Basic blocks:** 1
- **Return type:** bool

2.3 Locals

Local	Type	Notes
0	bool	Return place
1	usize	
2	usize	

2.4 Control-Flow Overview

bb0

2.5 Basic Blocks

2.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
<code>_0 = _1 > _2</code>	Greater than operation
<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

