

# 1 main — MIR Walkthrough

**Purpose:** TODO: Describe why this walkthrough exists

## 1.1 Source Context

```
fn main() {  
    let s:St = St { a:1, b:2 };  
  
    assert!(s.a + 1 == s.b);  
}
```

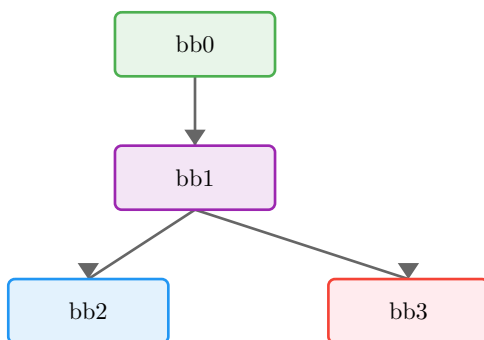
## 1.2 Function Overview

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

## 1.3 Locals

Local	Type	Notes
0	()	Return place
1	St	
2	bool	
3	u32	
4	u32	
5	(u32, bool)	
6	u32	
7	!	

## 1.4 Control-Flow Overview



## 1.5 Basic Blocks

### 1.5.1 bb0 — entry

*Entry point of the function.*

MIR	Annotation
-----	------------

<code>_1 = St(1, 2)</code>	Construct aggregate
<code>_4 = _1.0</code>	Copy value
<code>_5 = checked(_4 + 1)</code>	Checked Add (may panic)
<code>→ assert(move _5.1 == false) → bb1</code>	Panic if move _5.1 is true

### 1.5.2 bb1 — branch point

MIR	Annotation
<code>_3 = move _5.0</code>	Move value
<code>_6 = _1.1</code>	Copy value
<code>_2 = move _3 == move _6</code>	Equal operation
<code>→ switch(move _2) [0→bb3; else→bb2]</code>	Branch on move _2

### 1.5.3 bb2 — return / success

*Normal return path.*

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

### 1.5.4 bb3 — panic path

*Panic/diverging path.*

MIR	Annotation
<code>→ _7 = panic([16 bytes])</code>	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

