# 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);
}
```

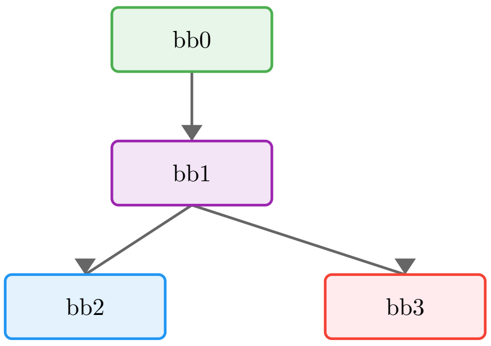
## 1.2 Function Overview

- **Function:** `main`
- **Basic blocks:** 4
- **Return type:** `() (0 bytes, align 1)`
- **Notable properties:**
  - Contains panic path
  - Uses checked arithmetic
  - Contains assertions
  - Has conditional branches

## 1.3 Locals

| Local | Type | Notes |
|---|---|---|
| 0 | `() (0 bytes, align 1)` | Return place |
| 1 | `St (8 bytes, align 4)` | |
| 2 | `Bool` | |
| 3 | `Uint(U32)` | |
| 4 | `Uint(U32)` | |
| 5 | `(u32, bool) (8 bytes, align 4)` | |
| 6 | `Uint(U32)` | |
| 7 | `()` | |

## 1.4 Control-Flow Overview



## 1.5 Basic Blocks

### 1.5.1 bb0 — entry

*Entry point of the function.*

| MIR | Annotation |
|---|---|

| | |
|---|---|
| `_1 = St(1, 2)` | Construct aggregate |
| `_4 = _1.0` | Copy value |
| `_5 = checked(_4 + 1)` | Checked Add (may panic) |
| `→ assert(move _5.1 == false) → bb1` | Panic if move __5.1 is true |

### 1.5.2 bb1 — branch point

| MIR | Annotation |
|---|---|
| `_3 = move _5.0` | Move value |
| `_6 = _1.1` | Copy value |
| `_2 = move _3 == move _6` | Equal operation |
| `→ switch(move _2) [0→bb3; else→bb2]` | Branch on move __2 |

### 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 |
|---|---|
| `→ _7 = 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