1 `main` — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```
    assert!(sum(20, 22) == 42);
}
```

1.2 Function Overview

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

1.3 Locals

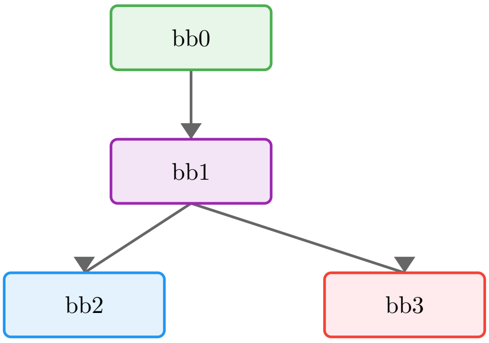
Local	Type	Notes
0	`() (0 bytes, align 1)`	Return place
1	`{closure@tests/integration/programs/closure-args.rs:2:15: 2:28}`	
2	`Int(I32)`	
3	`&{closure@tests/integration/programs/closure-args.rs:2:15: 2:28} (8 bytes, align 8)`	
4	`(i32, i32) (8 bytes, align 4)`	
5	`()`	

1.4 Borrows

#	Borrow	Kind	Created At	Borrowed
0	`_3`	`&`	`bb0[0]`	`_1`

Borrows are tracked conservatively: active from creation until reassignment or scope end.

1.5 Control-Flow Overview



1.6 Basic Blocks

1.6.1 bb0 — entry

Entry point of the function.

MIR	Annotation
`_3 = &_1`	Shared borrow
`_4 = Tuple(20, 22)`	Construct aggregate
`→ _2 = call(move _3, move _4) → bb1`	Call call

1.6.2 bb1 — branch point

MIR	Annotation
`→ switch(move _2) [42→bb2; else→bb3]`	Branch on move _2

1.6.3 bb2 — return / success

Normal return path.

MIR	Annotation
`→ return`	Return from function

1.6.4 bb3 — panic path

Panic/diverging path.

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

1.7 Key Observations

TODO: Add bullet points summarizing what this MIR teaches

-
-

1.8 Takeaways

TODO: One or two sentences to generalize this example

2 {closure#0} — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

2.1 Source Context

```
fn main() {
    let sum = |x, y| -> i32 { x + y };
```

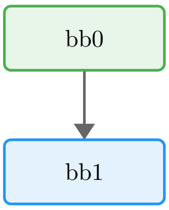
2.2 Function Overview

- **Function:** `{closure#0}`
- **Basic blocks:** 2
- **Return type:** `Int(I32)`
- **Notable properties:**
  - Contains panic path
  - Uses checked arithmetic
  - Contains assertions

2.3 Locals

Local	Type	Notes
0	`Int(I32)`	Return place
1	`&{closure@tests/integration/programs/closure-args.rs:2:15: 2:28} (8 bytes, align 8)`	
2	`Int(I32)`	
3	`Int(I32)`	
4	`(i32, bool) (8 bytes, align 4)`	

2.4 Control-Flow Overview



2.5 Basic Blocks

2.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
`_4 = checked(_2 + _3)`	Checked Add (may panic)
`→ assert(move _4.1 == false) → bb1`	Panic if move _4.1 is true

2.5.2 bb1 — return / success

Normal return path.

MIR	Annotation
`_0 = move _4.0`	Move value
`→ return`	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