1 `test_binop` — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```rust
fn test_binop(x:i32, y:i32) -> () {
// Arithmetic
    // Addition
    assert!(x + y == 52);
    assert!(52 == x + y);
    assert!(x + y == y + x);

    // Subtraction
    assert!(x - y == 32);
    assert!(y - x == -32);
    assert!(y - x != x - y);

    // Multiplication
    assert!(x * y == 420);
    assert!(x * -y == -420);
    assert!(-x * y == -420);
    assert!(-x * -y == 420);

    // Division
    // assert!(420 / 10 == 42); // FAILING SEE div.rs and div.mir

    // Modulo
    // assert!(x % 10 == 2); // FAILING SEE modulo.rs and modulo.mir

// Bitwise
    // Xor
    assert!(1 ^ 2 == 3);
    assert!(1 ^ 3 == 2);

    // Or
    assert!(1 | 2 == 3);
    assert!(1 | 3 == 3);

    // And
    assert!(1 & 2 == 0);
    assert!(1 & 3 == 1);

    // // Shl
    assert!(2 << 1 == 4);
    // assert!(-128_i8 << 1 == 0); FAILS SEE shl_min.rs and shl_min.mir
    // assert!(-32768_i16 << 1 == 0); FAILS SEE shl_min.rs and shl_min.mir
    // assert!(-2147483648_i32 << 1 == 0); FAILS SEE shl_min.rs and shl_min.mir
    // assert!(-9223372036854775808_i64 << 1 == 0); FAILS SEE shl_min.rs and shl_min.mir
    // assert!(-170141183460469231731687303715884105728_i128 << 1 == 0); FAILS SEE shl_min.rs and
shl_min.mir

    // // Shr
    assert!(2 >> 1 == 1);
    assert!(3 >> 1 == 1);
    assert!(1 >> 1 == 0);

// Comparisions
    // Less Then
    assert!(x < x + y);
```

```
    // Less Then or Equal
    assert!(x <= x + y);
    assert!(x <= x + y - y);

    // Greater Then
    assert!(x + y > x);

    // Greater Then or Equal
    assert!(x + y >= x);
    assert!(x + y - y >= x);
}
```

1.2 Function Overview

- **Function:** `test_binop`
- **Basic blocks:** 81
- **Return type:** `()`
- **Notable properties:**
  - ‣ Contains panic path
  - ‣ Uses checked arithmetic
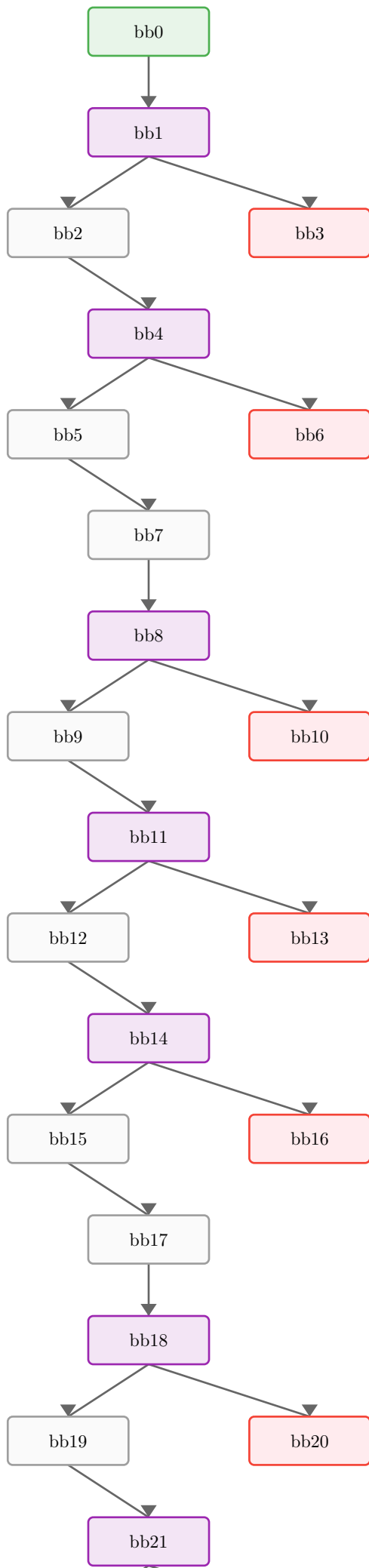 - ‣ Contains assertions
  - ‣ Has conditional branches

1.3 Locals

Local	Type	Notes
0	`()`	Return place
1	`i32`	
2	`i32`	
3	`i32`	
4	`(i32, bool)`	
5	`!`	
6	`i32`	
7	`(i32, bool)`	
8	`!`	
9	`bool`	
10	`i32`	
11	`(i32, bool)`	
12	`i32`	
13	`(i32, bool)`	
14	`!`	
15	`i32`	
16	`(i32, bool)`	
17	`!`	
18	`i32`	
19	`(i32, bool)`	
20	`!`	
21	`bool`	
22	`i32`	
23	`(i32, bool)`	
24	`i32`	

25	(i32, bool)	
26	!	
27	i32	
28	(i32, bool)	
29	!	
30	i32	
31	i32	
32	bool	
33	(i32, bool)	
34	!	
35	i32	
36	i32	
37	bool	
38	(i32, bool)	
39	!	
40	i32	
41	i32	
42	bool	
43	i32	
44	bool	
45	(i32, bool)	
46	!	
47	i32	
48	!	
49	i32	
50	!	
51	i32	
52	!	
53	i32	
54	!	
55	i32	
56	!	
57	i32	
58	!	
59	i32	
60	u32	
61	bool	
62	!	
63	i32	
64	u32	
65	bool	
66	!	
67	i32	
68	u32	
69	bool	

70	!	
71	i32	
72	u32	
73	bool	
74	!	
75	bool	
76	i32	
77	(i32, bool)	
78	!	
79	bool	
80	i32	
81	(i32, bool)	
82	!	
83	bool	
84	i32	
85	i32	
86	(i32, bool)	
87	(i32, bool)	
88	!	
89	bool	
90	i32	
91	(i32, bool)	
92	!	
93	bool	
94	i32	
95	(i32, bool)	
96	!	
97	bool	
98	i32	
99	i32	
100	(i32, bool)	
101	(i32, bool)	
102	!	

1.4 Control-Flow Overview

1.5 Basic Blocks

1.5.1 bb0 — entry

Entry point of the function.

MIR	Annotation
_4 = checked(_1 + _2)	Checked Add (may panic)
→ assert(move _4.1 == false) → bb1	Panic if move __4.1 is true

1.5.2 bb1 — branch point

MIR	Annotation
_3 = move _4.0	Move value
→ switch(move _3) \[52→bb2; else→bb3\]	Branch on move __3

1.5.3 bb2

MIR	Annotation
_7 = checked(_1 + _2)	Checked Add (may panic)
→ assert(move _7.1 == false) → bb4	Panic if move __7.1 is true

1.5.4 bb3 — panic path

Panic/diverging path.

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

1.5.5 bb4 — branch point

MIR	Annotation
_6 = move _7.0	Move value
→ switch(move _6) \[52→bb5; else→bb6\]	Branch on move __6

1.5.6 bb5

MIR	Annotation
_11 = checked(_1 + _2)	Checked Add (may panic)
→ assert(move _11.1 == false) → bb7	Panic if move __11.1 is true

1.5.7 bb6 — panic path

Panic/diverging path.

MIR	Annotation
→ _8 = panic(\[16 bytes\])	Call panic

1.5.8 bb7

MIR	Annotation
_10 = move _11.0	Move value
_13 = checked(_2 + _1)	Checked Add (may panic)
→ assert(move _13.1 == false) → bb8	Panic if move __13.1 is true

1.5.9 bb8 — branch point

MIR	Annotation

_12 = move _13.0	Move value
_9 = move _10 == move _12	Equal operation
→ switch(move _9) \[0→bb10; else→bb9\]	Branch on move _9

1.5.10 bb9

MIR	Annotation
_16 = checked(_1 - _2)	Checked Subtract (may panic)
→ assert(move _16.1 == false) → bb11	Panic if move _16.1 is true

1.5.11 bb10 — panic path

Panic/diverging path.

MIR	Annotation
→ _14 = panic(\[16 bytes\])	Call panic

1.5.12 bb11 — branch point

MIR	Annotation
_15 = move _16.0	Move value
→ switch(move _15) \[32→bb12; else→bb13\]	Branch on move _15

1.5.13 bb12

MIR	Annotation
_19 = checked(_2 - _1)	Checked Subtract (may panic)
→ assert(move _19.1 == false) → bb14	Panic if move _19.1 is true

1.5.14 bb13 — panic path

Panic/diverging path.

MIR	Annotation
→ _17 = panic(\[16 bytes\])	Call panic

1.5.15 bb14 — branch point

MIR	Annotation
_18 = move _19.0	Move value
→ switch(move _18) \[4294967264→bb15; else→bb16\]	Branch on move _18

1.5.16 bb15

MIR	Annotation
_23 = checked(_2 - _1)	Checked Subtract (may panic)
→ assert(move _23.1 == false) → bb17	Panic if move _23.1 is true

1.5.17 bb16 — panic path

Panic/diverging path.

MIR	Annotation
→ _20 = panic(\[16 bytes\])	Call panic

1.5.18 bb17

MIR	Annotation
_22 = move _23.0	Move value
_25 = checked(_1 - _2)	Checked Subtract (may panic)
→ assert(move _25.1 == false) → bb18	Panic if move __25.1 is true

1.5.19 bb18 — branch point

MIR	Annotation
_24 = move _25.0	Move value
_21 = move _22 != move _24	Not equal operation
→ switch(move _21) \[0→bb20; else→bb19\]	Branch on move __21

1.5.20 bb19

MIR	Annotation
_28 = checked(_1 * _2)	Checked Multiply (may panic)
→ assert(move _28.1 == false) → bb21	Panic if move __28.1 is true

1.5.21 bb20 — panic path
Panic/diverging path.

MIR	Annotation
→ _26 = panic(\[16 bytes\])	Call panic

1.5.22 bb21 — branch point

MIR	Annotation
_27 = move _28.0	Move value
→ switch(move _27) \[420→bb22; else→bb23\]	Branch on move __27

1.5.23 bb22

MIR	Annotation
_32 = _2 == -2147483648	Equal operation
→ assert(move _32 == false) → bb24	Panic if move __32 is true

1.5.24 bb23 — panic path
Panic/diverging path.

MIR	Annotation
→ _29 = panic(\[16 bytes\])	Call panic

1.5.25 bb24

MIR	Annotation
_31 = -_2	Negation
_33 = checked(_1 * _31)	Checked Multiply (may panic)
→ assert(move _33.1 == false) → bb25	Panic if move __33.1 is true

1.5.26 bb25 — branch point

MIR	Annotation

MIR	Annotation
_30 = move _33.0	Move value
→ switch(move _30) \[4294966876→bb26; else→bb27\]	Branch on move __30

1.5.27 bb26

MIR	Annotation
_37 = _1 == -2147483648	Equal operation
→ assert(move _37 == false) → bb28	Panic if move __37 is true

1.5.28 bb27 — panic path

Panic/diverging path.

MIR	Annotation
→ _34 = panic(\[16 bytes\])	Call panic

1.5.29 bb28

MIR	Annotation
_36 = -_1	Negation
_38 = checked(_36 * _2)	Checked Multiply (may panic)
→ assert(move _38.1 == false) → bb29	Panic if move __38.1 is true

1.5.30 bb29 — branch point

MIR	Annotation
_35 = move _38.0	Move value
→ switch(move _35) \[4294966876→bb30; else→bb31\]	Branch on move __35

1.5.31 bb30

MIR	Annotation
_42 = _1 == -2147483648	Equal operation
→ assert(move _42 == false) → bb32	Panic if move __42 is true

1.5.32 bb31 — panic path

Panic/diverging path.

MIR	Annotation
→ _39 = panic(\[16 bytes\])	Call panic

1.5.33 bb32

MIR	Annotation
_41 = -_1	Negation
_44 = _2 == -2147483648	Equal operation
→ assert(move _44 == false) → bb33	Panic if move __44 is true

1.5.34 bb33

MIR	Annotation
_43 = -_2	Negation
_45 = checked(_41 * _43)	Checked Multiply (may panic)

MIR	Annotation
→ assert(move _45.1 == false) → bb34	Panic if move __45.1 is true

1.5.35 bb34 — branch point

MIR	Annotation
_40 = move _45.0	Move value
→ switch(move _40) \[420→bb35; else→bb36\]	Branch on move __40

1.5.36 bb35 — branch point

MIR	Annotation
_47 = 1 ^ 2	XOR operation
→ switch(move _47) \[3→bb37; else→bb38\]	Branch on move __47

1.5.37 bb36 — panic path
Panic/diverging path.

MIR	Annotation
→ _46 = panic(\[16 bytes\])	Call panic

1.5.38 bb37 — branch point

MIR	Annotation
_49 = 1 ^ 3	XOR operation
→ switch(move _49) \[2→bb39; else→bb40\]	Branch on move __49

1.5.39 bb38 — panic path
Panic/diverging path.

MIR	Annotation
→ _48 = panic(\[16 bytes\])	Call panic

1.5.40 bb39 — branch point

MIR	Annotation
_51 = 1	2	OR operation
→ switch(move _51) \[3→bb41; else→bb42\]	Branch on move __51

1.5.41 bb40 — panic path
Panic/diverging path.

MIR	Annotation
→ _50 = panic(\[16 bytes\])	Call panic

1.5.42 bb41 — branch point

MIR	Annotation
_53 = 1	3	OR operation
→ switch(move _53) \[3→bb43; else→bb44\]	Branch on move __53

1.5.43 bb42 — panic path
Panic/diverging path.

MIR	Annotation

MIR	
→ _52 = panic(\[16 bytes\])	Call panic

1.5.44 bb43 — branch point

MIR	Annotation
_55 = 1 & 2	AND operation
→ switch(move _55) \[0→bb45; else→bb46\]	Branch on move __55

1.5.45 bb44 — panic path
Panic/diverging path.

MIR	Annotation
→ _54 = panic(\[16 bytes\])	Call panic

1.5.46 bb45 — branch point

MIR	Annotation
_57 = 1 & 3	AND operation
→ switch(move _57) \[1→bb47; else→bb48\]	Branch on move __57

1.5.47 bb46 — panic path
Panic/diverging path.

MIR	Annotation
→ _56 = panic(\[16 bytes\])	Call panic

1.5.48 bb47

MIR	Annotation
_60 = 1 as RigidTy(Uint(U32))	Integer conversion
_61 = move _60 \< 32	Less than operation
→ assert(move _61 == true) → bb49	Panic if move __61 is false

1.5.49 bb48 — panic path
Panic/diverging path.

MIR	Annotation
→ _58 = panic(\[16 bytes\])	Call panic

1.5.50 bb49 — branch point

MIR	Annotation
_59 = 2 \<\< 1	Shift left operation
→ switch(move _59) \[4→bb50; else→bb51\]	Branch on move __59

1.5.51 bb50

MIR	Annotation
_64 = 1 as RigidTy(Uint(U32))	Integer conversion
_65 = move _64 \< 32	Less than operation
→ assert(move _65 == true) → bb52	Panic if move __65 is false

1.5.52 bb51 — panic path
Panic/diverging path.

MIR	Annotation
→ _62 = panic(\[16 bytes\])	Call panic

1.5.53 bb52 — branch point

MIR	Annotation
_63 = 2 \>\> 1	Shift right operation
→ switch(move _63) \[1→bb53; else→bb54\]	Branch on move _63

1.5.54 bb53

MIR	Annotation
_68 = 1 as RigidTy(Uint(U32))	Integer conversion
_69 = move _68 \< 32	Less than operation
→ assert(move _69 == true) → bb55	Panic if move _69 is false

1.5.55 bb54 — panic path
Panic/diverging path.

MIR	Annotation
→ _66 = panic(\[16 bytes\])	Call panic

1.5.56 bb55 — branch point

MIR	Annotation
_67 = 3 \>\> 1	Shift right operation
→ switch(move _67) \[1→bb56; else→bb57\]	Branch on move _67

1.5.57 bb56

MIR	Annotation
_72 = 1 as RigidTy(Uint(U32))	Integer conversion
_73 = move _72 \< 32	Less than operation
→ assert(move _73 == true) → bb58	Panic if move _73 is false

1.5.58 bb57 — panic path
Panic/diverging path.

MIR	Annotation
→ _70 = panic(\[16 bytes\])	Call panic

1.5.59 bb58 — branch point

MIR	Annotation
_71 = 1 \>\> 1	Shift right operation
→ switch(move _71) \[0→bb59; else→bb60\]	Branch on move _71

1.5.60 bb59

MIR	Annotation
_77 = checked(_1 + _2)	Checked Add (may panic)
→ assert(move _77.1 == false) → bb61	Panic if move _77.1 is true

1.5.61 bb60 — panic path

Panic/diverging path.

MIR	Annotation
→ _74 = panic(\[16 bytes\])	Call panic

1.5.62 bb61 — branch point

MIR	Annotation
_76 = move _77.0	Move value
_75 = _1 \< move _76	Less than operation
→ switch(move _75) \[0→bb63; else→bb62\]	Branch on move __75

1.5.63 bb62

MIR	Annotation
_81 = checked(_1 + _2)	Checked Add (may panic)
→ assert(move _81.1 == false) → bb64	Panic if move __81.1 is true

1.5.64 bb63 — panic path

Panic/diverging path.

MIR	Annotation
→ _78 = panic(\[16 bytes\])	Call panic

1.5.65 bb64 — branch point

MIR	Annotation
_80 = move _81.0	Move value
_79 = _1 \<= move _80	Less or equal operation
→ switch(move _79) \[0→bb66; else→bb65\]	Branch on move __79

1.5.66 bb65

MIR	Annotation
_86 = checked(_1 + _2)	Checked Add (may panic)
→ assert(move _86.1 == false) → bb67	Panic if move __86.1 is true

1.5.67 bb66 — panic path

Panic/diverging path.

MIR	Annotation
→ _82 = panic(\[16 bytes\])	Call panic

1.5.68 bb67

MIR	Annotation
_85 = move _86.0	Move value
_87 = checked(_85 - _2)	Checked Subtract (may panic)
→ assert(move _87.1 == false) → bb68	Panic if move __87.1 is true

1.5.69 bb68 — branch point

MIR	Annotation

MIR	Annotation
_84 = move _87.0	Move value
_83 = _1 \<= move _84	Less or equal operation
→ switch(move _83) \[0→bb70; else→bb69\]	Branch on move __83

1.5.70 bb69

MIR	Annotation
_91 = checked(_1 + _2)	Checked Add (may panic)
→ assert(move _91.1 == false) → bb71	Panic if move __91.1 is true

1.5.71 bb70 — panic path
Panic/diverging path.

MIR	Annotation
→ _88 = panic(\[16 bytes\])	Call panic

1.5.72 bb71 — branch point

MIR	Annotation
_90 = move _91.0	Move value
_89 = move _90 \> _1	Greater than operation
→ switch(move _89) \[0→bb73; else→bb72\]	Branch on move __89

1.5.73 bb72

MIR	Annotation
_95 = checked(_1 + _2)	Checked Add (may panic)
→ assert(move _95.1 == false) → bb74	Panic if move __95.1 is true

1.5.74 bb73 — panic path
Panic/diverging path.

MIR	Annotation
→ _92 = panic(\[16 bytes\])	Call panic

1.5.75 bb74 — branch point

MIR	Annotation
_94 = move _95.0	Move value
_93 = move _94 \>= _1	Greater or equal operation
→ switch(move _93) \[0→bb76; else→bb75\]	Branch on move __93

1.5.76 bb75

MIR	Annotation
_100 = checked(_1 + _2)	Checked Add (may panic)
→ assert(move _100.1 == false) → bb77	Panic if move __100.1 is true

1.5.77 bb76 — panic path
Panic/diverging path.

MIR	Annotation
→ _96 = panic(\[16 bytes\])	Call panic

1.5.78 bb77

MIR	Annotation
_99 = move _100.0	Move value
_101 = checked(_99 - _2)	Checked Subtract (may panic)
→ assert(move _101.1 == false) → bb78	Panic if move _101.1 is true

1.5.79 bb78 — branch point

MIR	Annotation
_98 = move _101.0	Move value
_97 = move _98 \>= _1	Greater or equal operation
→ switch(move _97) \[0→bb80; else→bb79\]	Branch on move _97

1.5.80 bb79 — return / success

Normal return path.

MIR	Annotation
→ return	Return from function

1.5.81 bb80 — panic path

Panic/diverging path.

MIR	Annotation
→ _102 = 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 `main` — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

2.1 Source Context
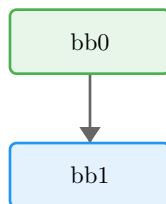
```
    test_binop(x, y);
    return ();
}
```

2.2 Function Overview
- **Function:** `main`
- **Basic blocks:** 2
- **Return type:** `()`

2.3 Locals

Local	Type	Notes
0	()	Return place
1	()	
2	i32	
3	i32	

2.4 Control-Flow Overview

2.5 Basic Blocks

2.5.1 bb0 — entry
Entry point of the function.

MIR	Annotation
_2 = 42	Load constant
_3 = 10	Load constant
→ _1 = test_binop(move _2, move _3) → bb1	Call test_binop

2.5.2 bb1 — return / success
Normal return path.

MIR	Annotation
→ 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