V8: Hooking up the Ignition to the Turbofan

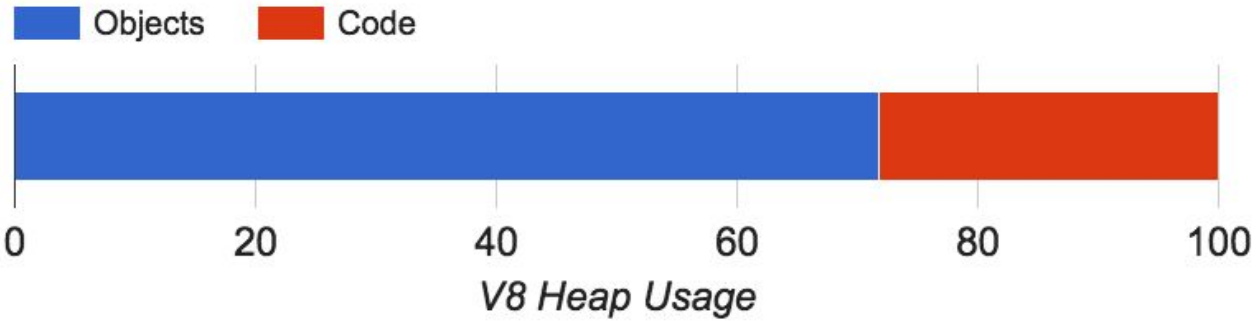
Leszek Swirski & Ross McIlroy

Google London

Ignition + Turbofan pipeline

Why a new pipeline?

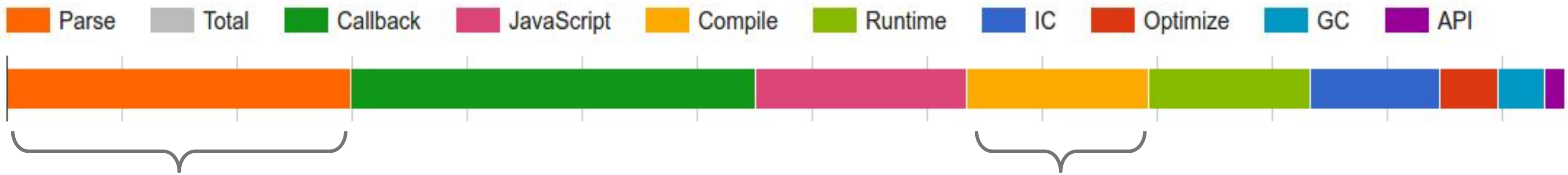
- Reduce memory usage



Why a new pipeline?

- Reduce memory usage

- Reduce startup time

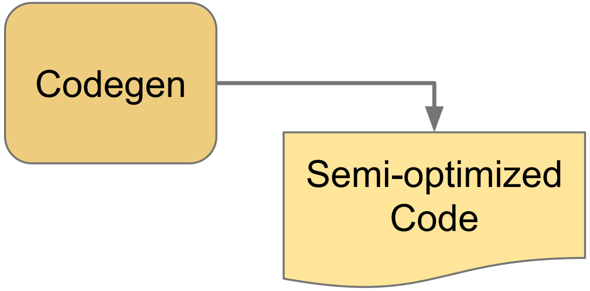


33% of time spent parsing + compiling

Why a new pipeline?

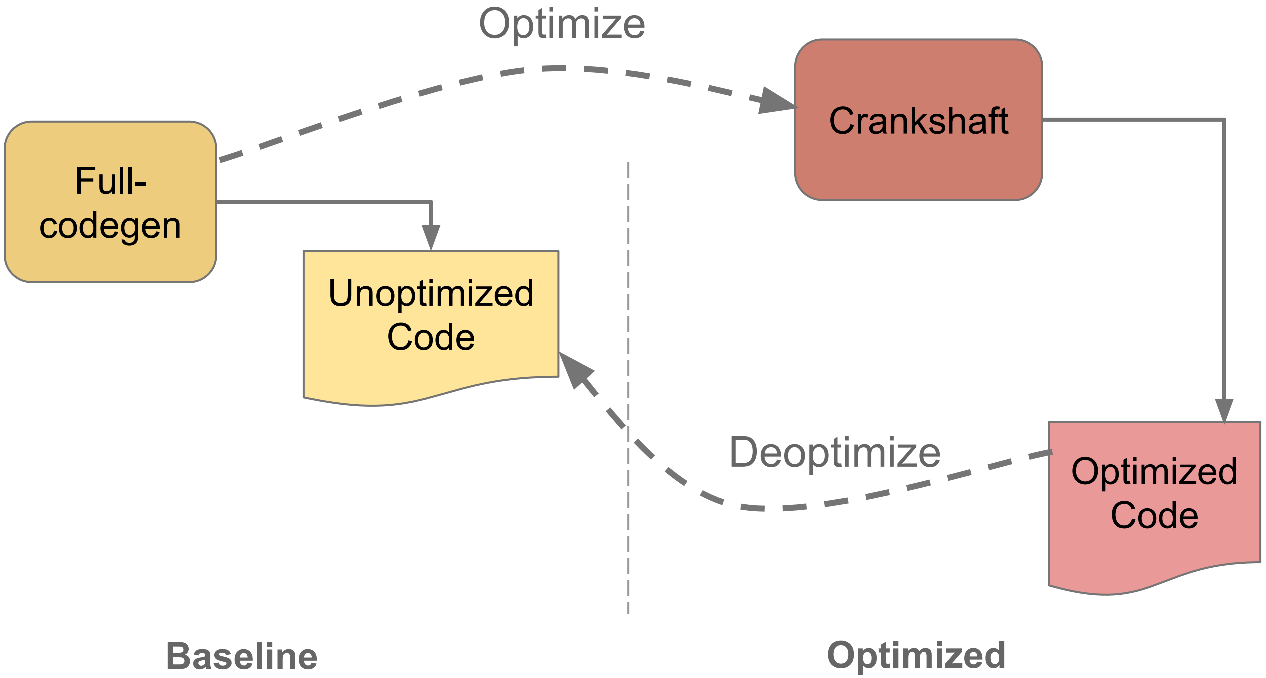
- Reduce memory usage
- Reduce startup time
- Reduce complexity

Compiler Pipeline (2008)

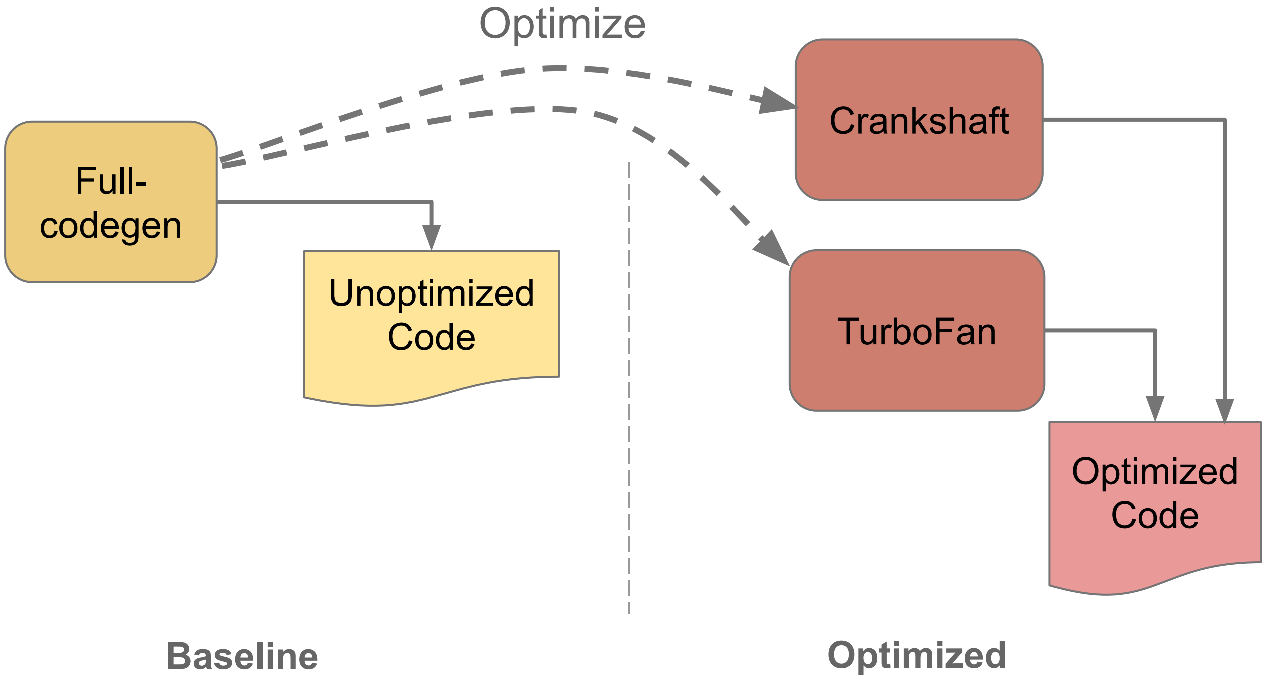


Baseline

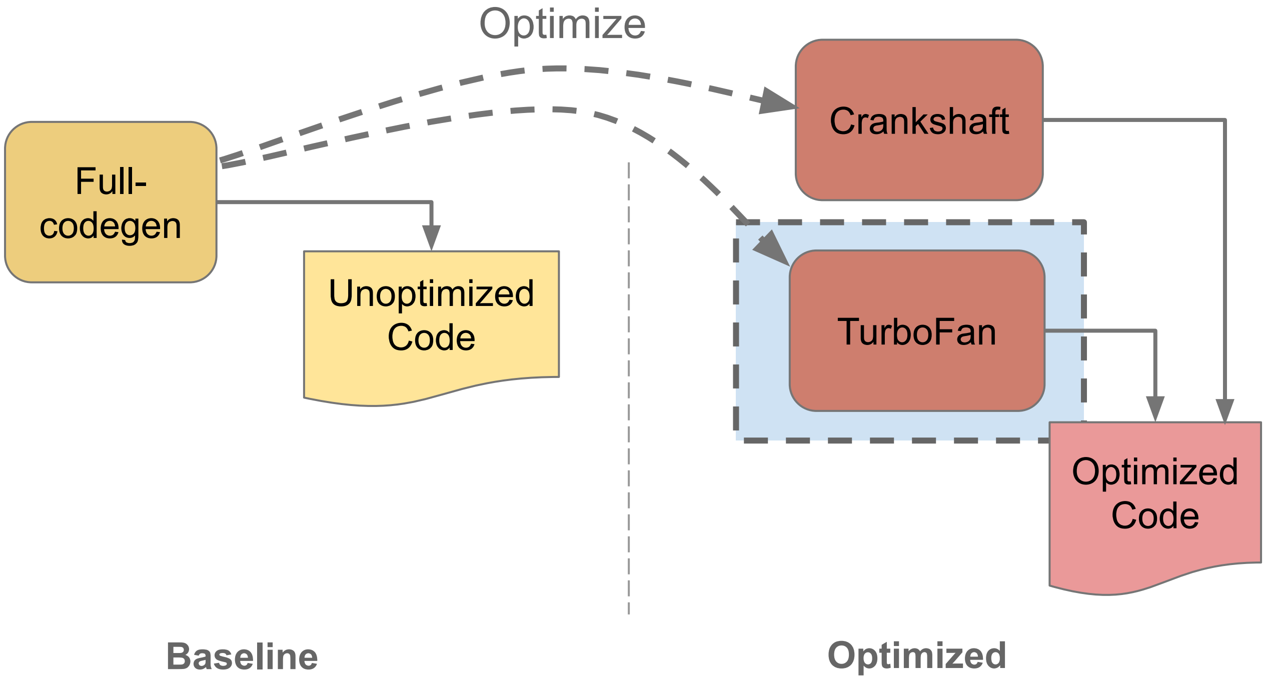
Compiler Pipeline (2010)



Compiler Pipeline (2015)

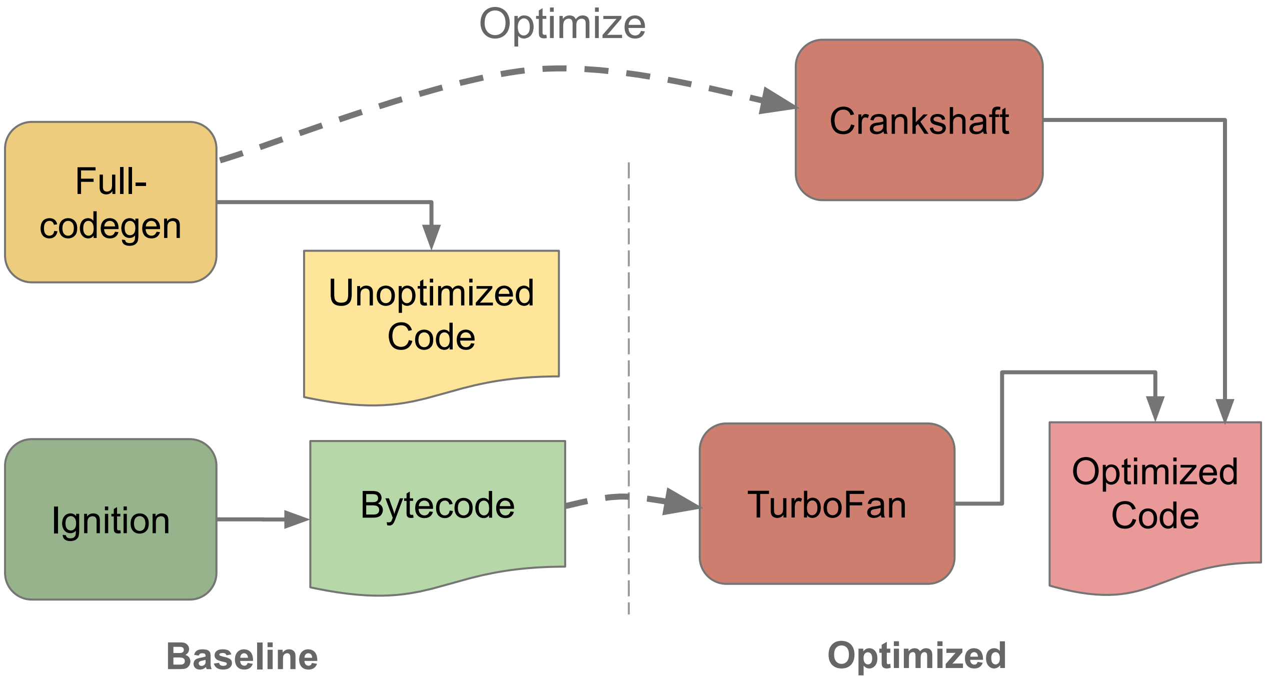


Compiler Pipeline (2015)

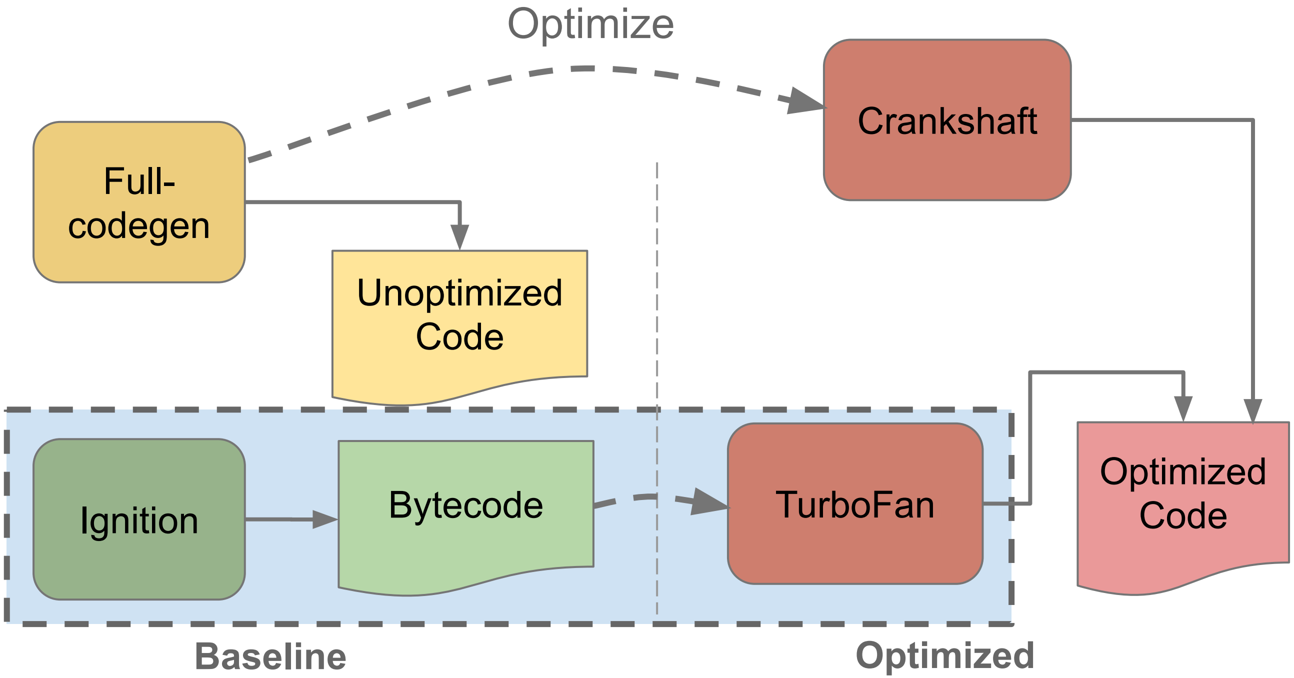


Google

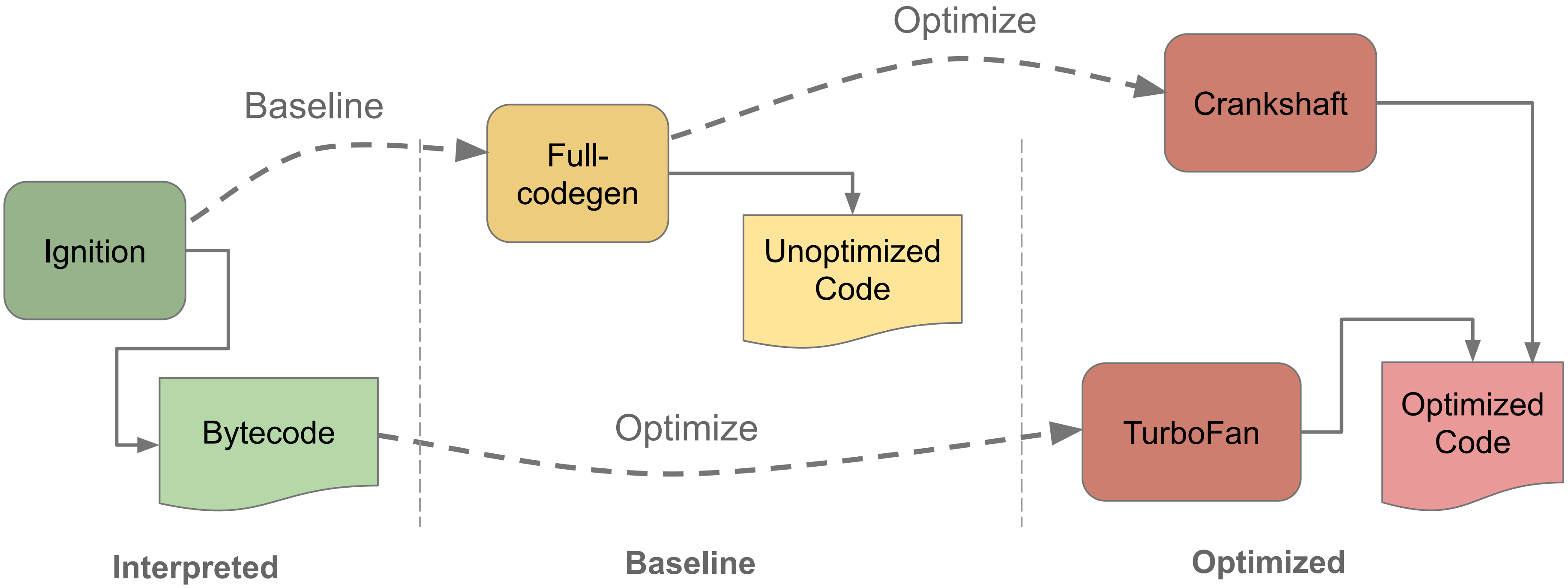
Compiler Pipeline (2016)



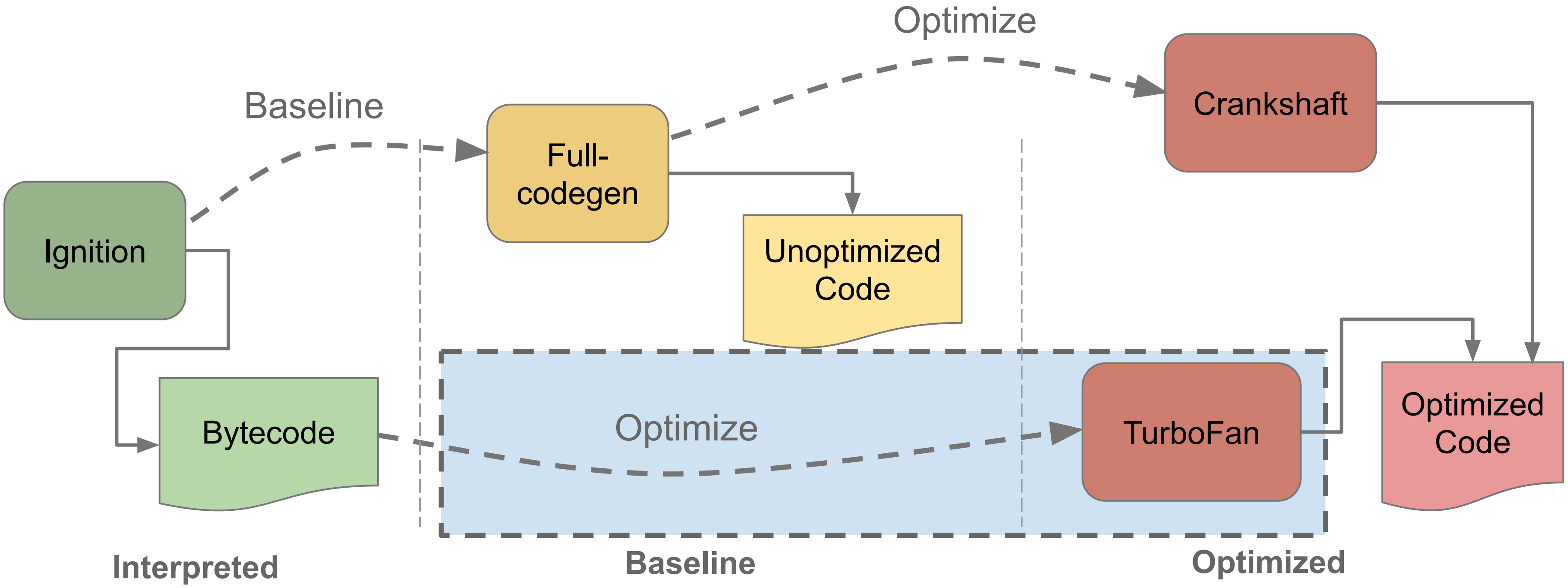
Compiler Pipeline (2016)



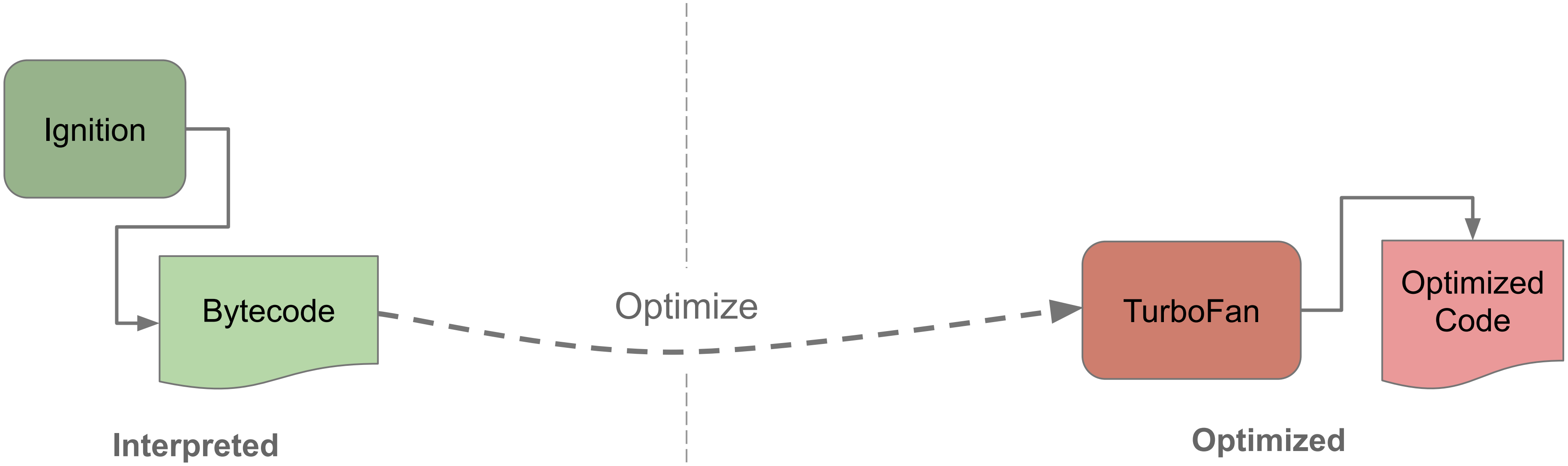
Compiler Pipeline (Svelte Devices)



Compiler Pipeline (Svelte Devices)



Compiler Pipeline (2017)



Bytecode to graph

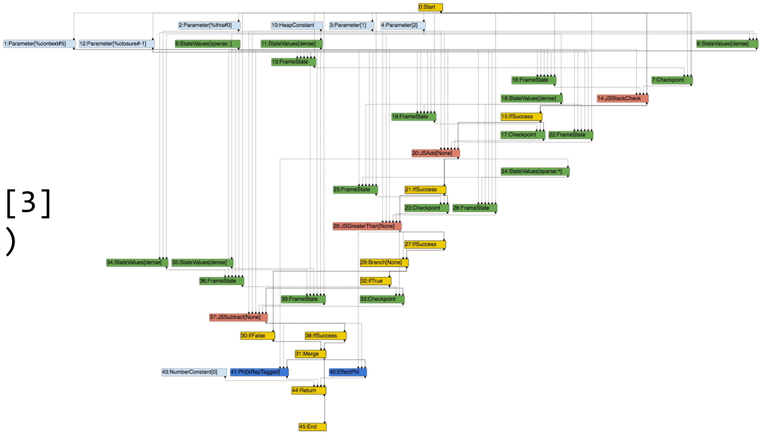
Ignition to Turbofan

```
function f(a, b) {
    var result = a + b;
    if (a > b) {
        result = a - b;
    }
    return result;
}
```

Text

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

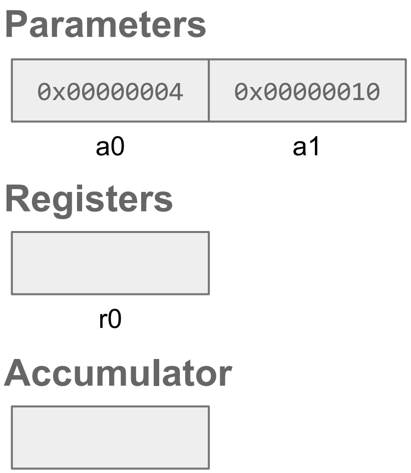
Bytecode
(Ignition)



Sea-of-nodes graph
(Turbofan)

Executing bytecode

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```



Parameters

0x00000004	0x00000010
a0	a1

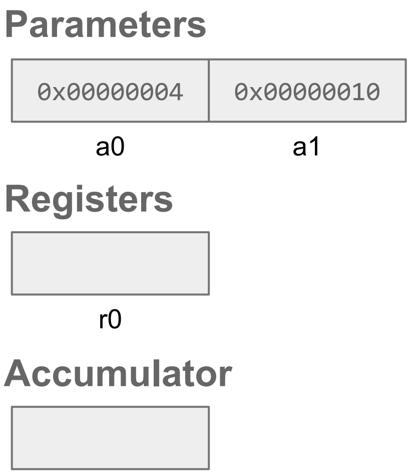
Registers

r0

Accumulator

Executing bytecode

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```



Parameters

0x00000004 0x00000010

a0 a1

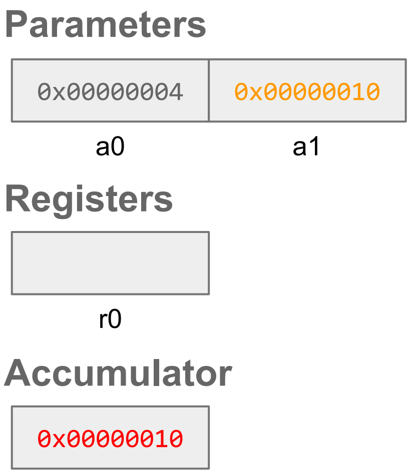
Registers

r0

Accumulator

Executing bytecode

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```



Parameters

a0	a1
0x00000004	0x00000010

Registers

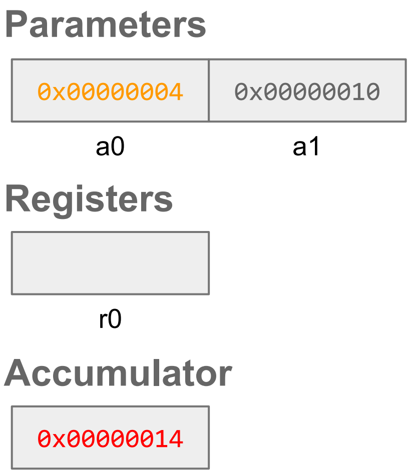
r0

Accumulator

0x00000010

Executing bytecode

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```



Parameters

a0	a1
0x00000004	0x00000010

Registers

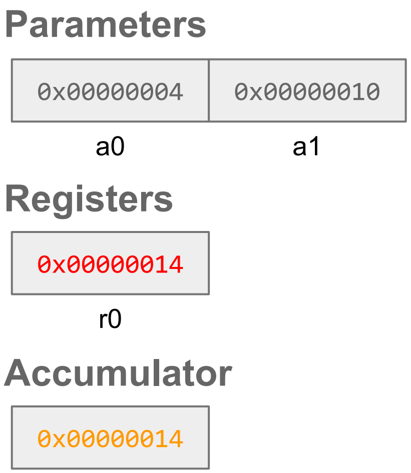
r0

Accumulator

0x00000014

Executing bytecode

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```



Parameters

0x00000004	0x00000010
a0	a1

Registers

0x00000014

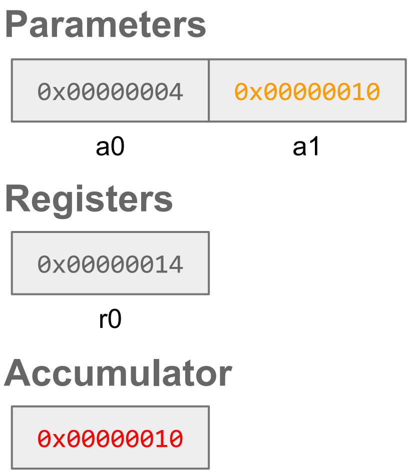
r0

Accumulator

0x00000014

Executing bytecode

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```



Parameters

0x00000004	0x00000010
a0	a1

Registers

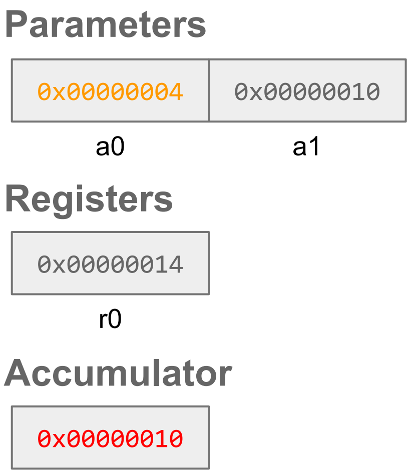
0x00000014
r0

Accumulator

0x00000010

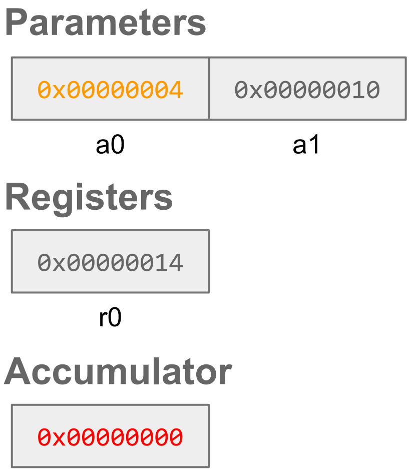
Executing bytecode

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```



Executing bytecode

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```



Parameters

0x00000004 — a0

0x00000010 — a1

Registers

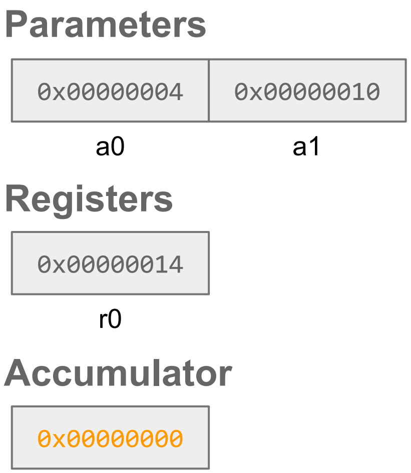
0x00000014 — r0

Accumulator

0x00000000

Executing bytecode

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```



Parameters

0x00000004	0x00000010
a0	a1

Registers

0x00000014

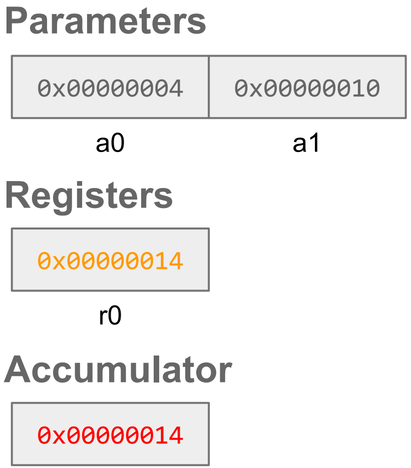
r0

Accumulator

0x00000000

Executing bytecode

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```



Parameters

0x00000004	0x00000010
a0	a1

Registers

0x00000014

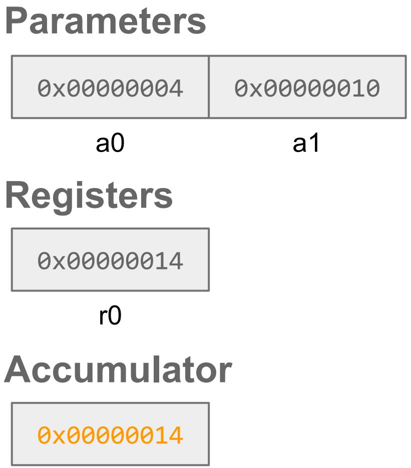
r0

Accumulator

0x00000014

Executing bytecode

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```



Parameters

0x00000004	0x00000010
a0	a1

Registers

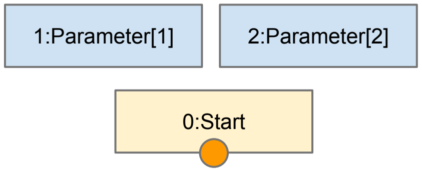
0x00000014

r0

Accumulator

0x00000014

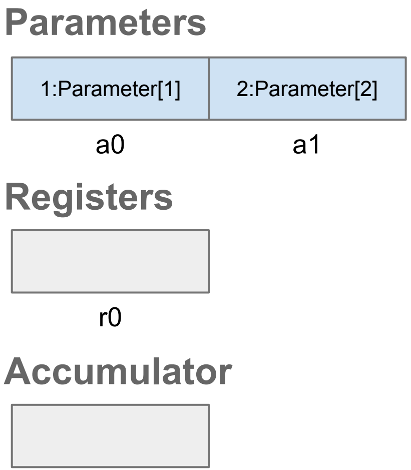
Building a graph


1:Parameter[1]

2:Parameter[2]

0:Start

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```


Parameters

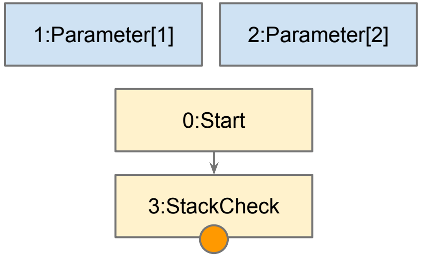
1:Parameter[1] — a0

2:Parameter[2] — a1

Registers

r0

Accumulator



Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

Parameters

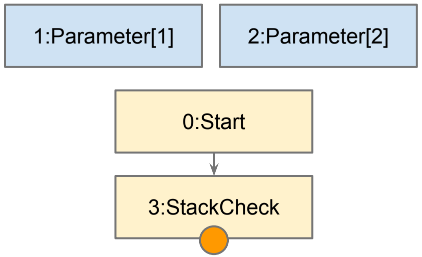
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

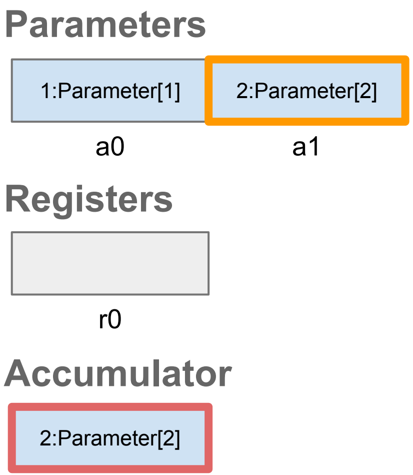
r0

Accumulator

Building a graph



```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```



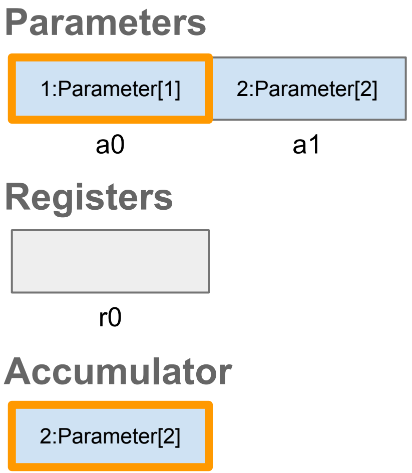
Parameters

Registers

Accumulator

Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```



Parameters

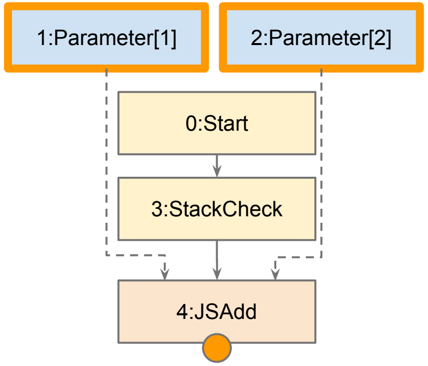
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

r0

Accumulator

2:Parameter[2]



Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

Parameters

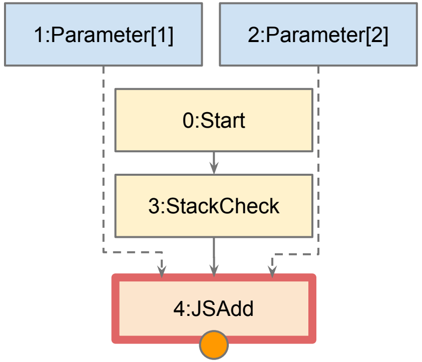
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

r0

Accumulator

4:JSAdd



Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

Parameters

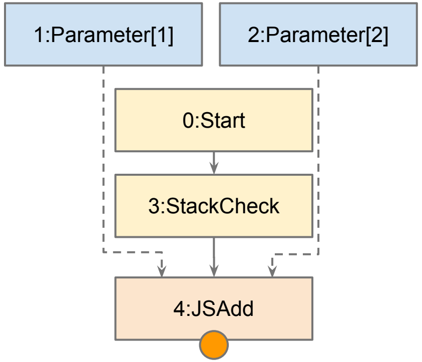
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

Accumulator

4:JSAdd



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

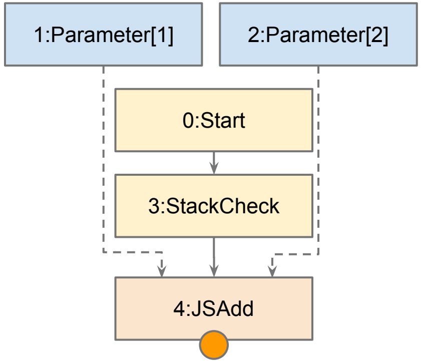
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

Accumulator

2:Parameter[2]



Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

Parameters

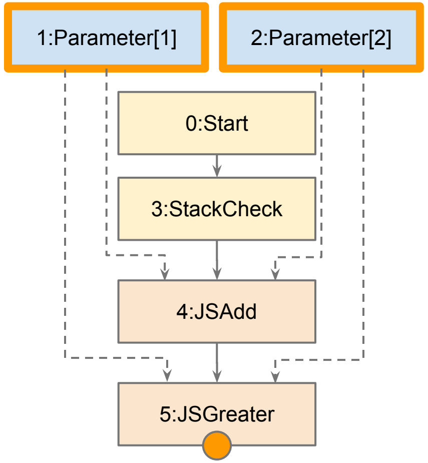
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

Accumulator

2:Parameter[2]



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

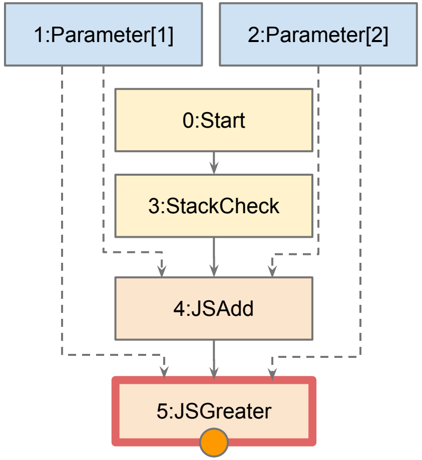
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

Accumulator

5:JSGreater



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

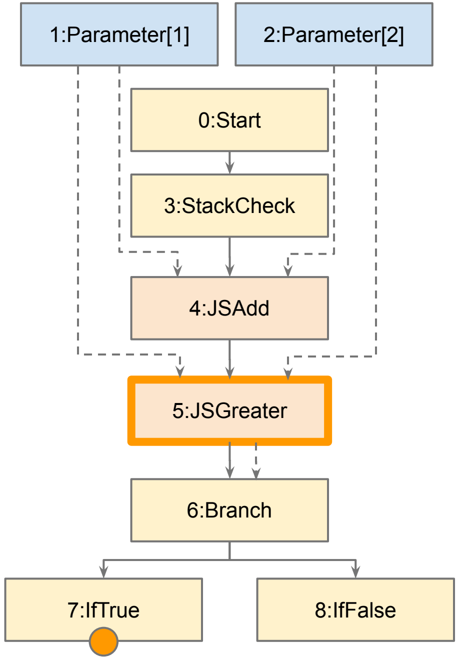
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

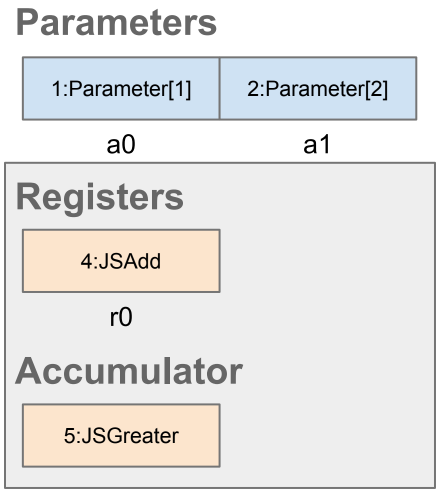
Accumulator

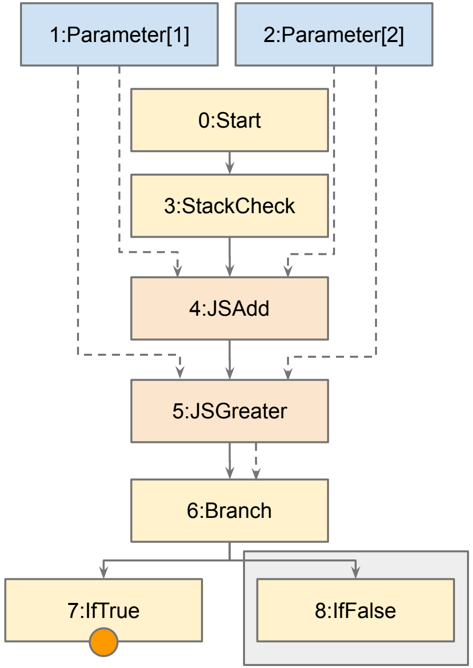
5:JSGreater



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```





Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

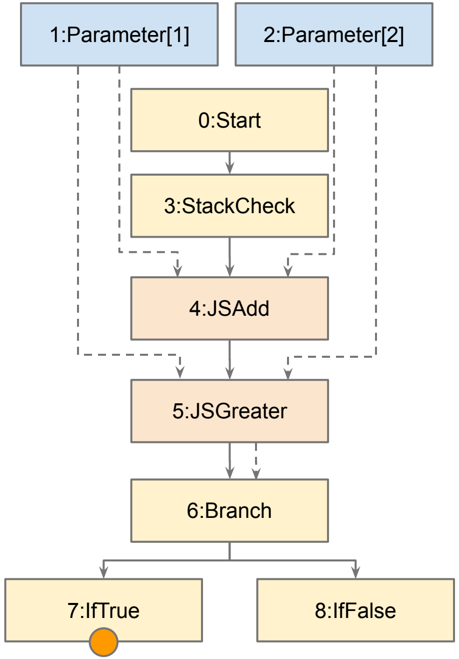
4:JSAdd
r0

Accumulator

5:JSGreater

22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse



Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse

Parameters

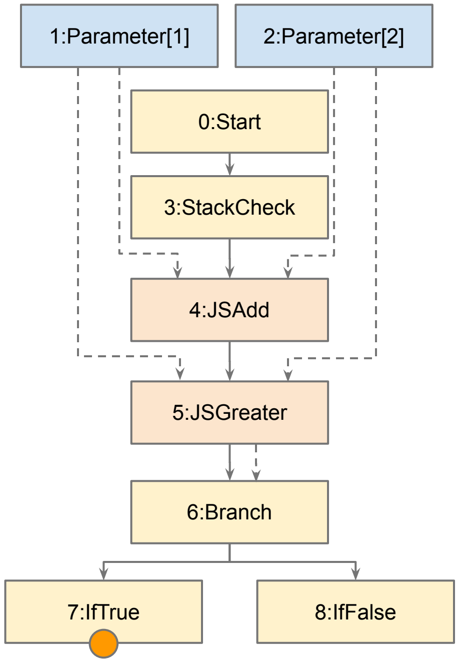
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

Accumulator

2:Parameter[2]



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

22:

Registers	Accumulator	Parent
4:JSAdd	5:JSGreater	8:IfFalse
r0		

Parameters

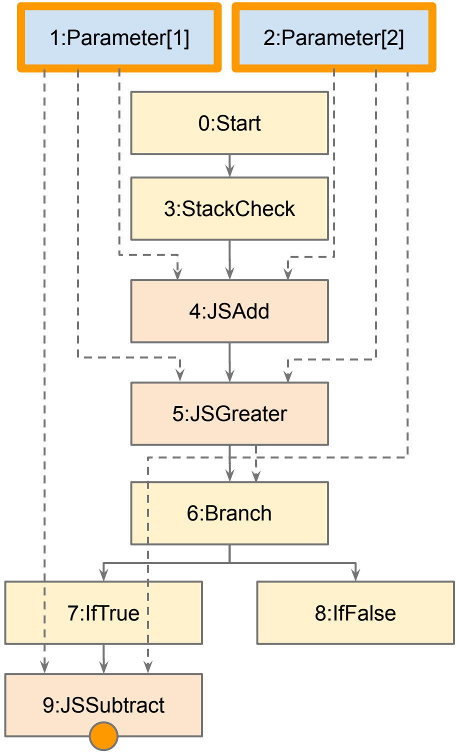
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

Accumulator

2:Parameter[2]



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

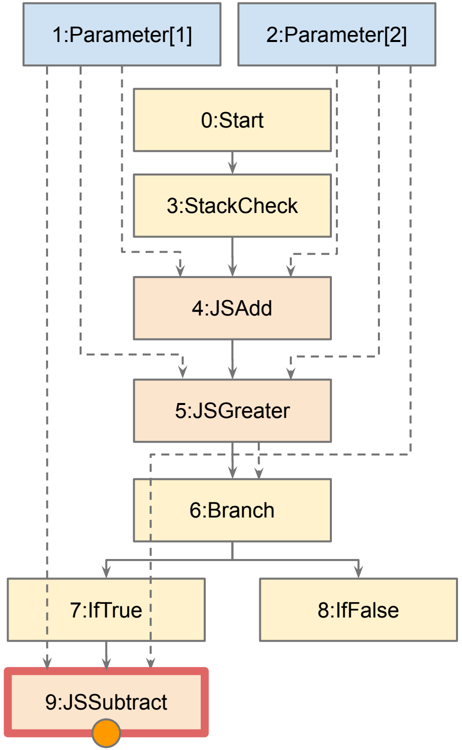
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

Accumulator

9:JSSubtract



22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse

Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

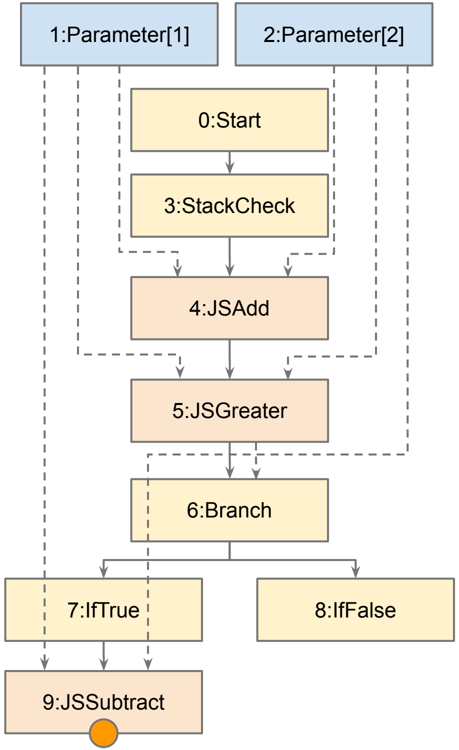
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

9:JSSubtract
r0

Accumulator

9:JSSubtract



22:

Registers	Accumulator	Parent
4:JSAdd	5:JSGreater	8:IfFalse
r0		

Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

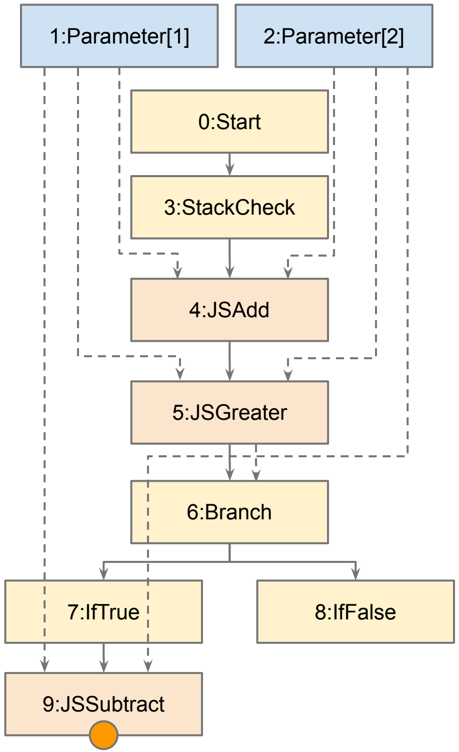
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

9:JSSubtract
r0

Accumulator

9:JSSubtract



22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse

Building a graph

```
 0 : 7d        StackCheck
 1 : 1c 02     Ldar a1
 3 : 28 03 02  Add a0, [2]
 6 : 1d fa     Star r0
 8 : 1c 02     Ldar a1
10 : 4f 03 03  TestGreaterThan a0, [3]
13 : 74 09     JumpIfFalse [9] (@22)
15 : 1c 02     Ldar a1
17 : 29 03 04  Sub a0, [4]
20 : 1d fa     Star r0
22 : 1c fa     Ldar r0
24 : 81        Return
```

22:

Registers	Accumulator	Parent
4:JSAdd	5:JSGreater	8:IfFalse
r0		

Parameters

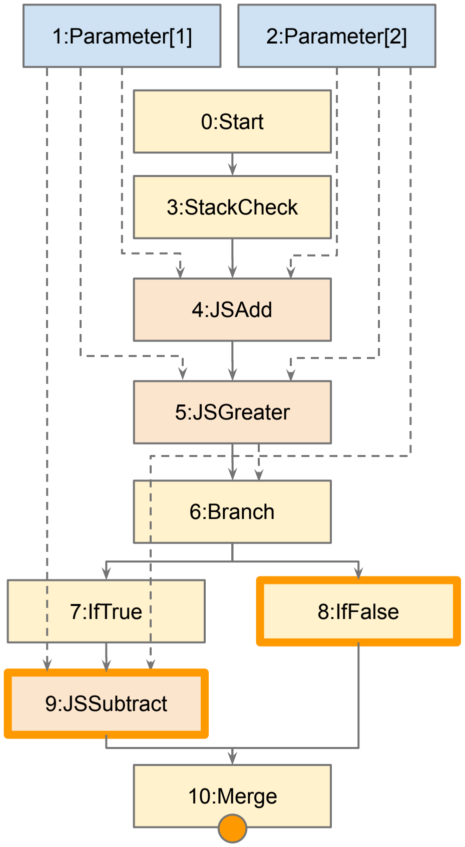
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

9:JSSubtract
r0

Accumulator

9:JSSubtract



Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse

Parameters

1:Parameter[1]	2:Parameter[2]
a0	a1

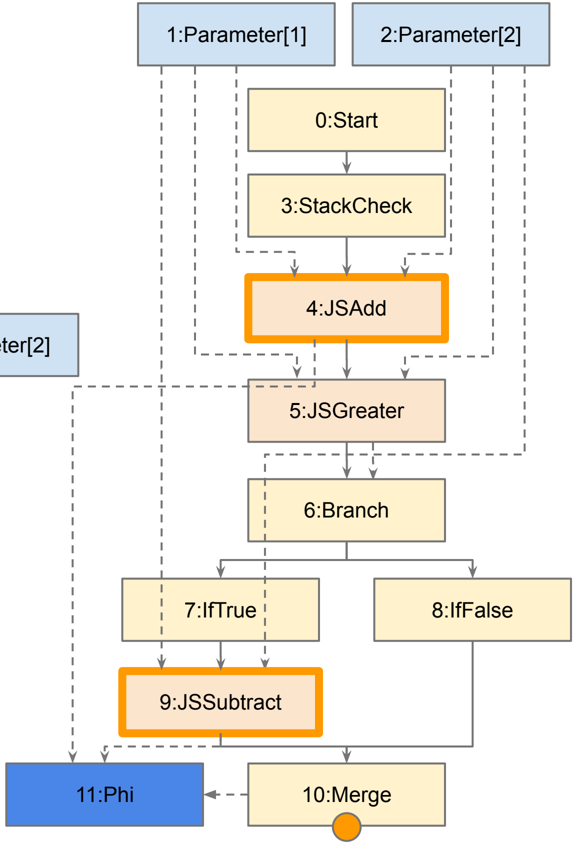
Registers

9:JSSubtract

r0

Accumulator

9:JSSubtract



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

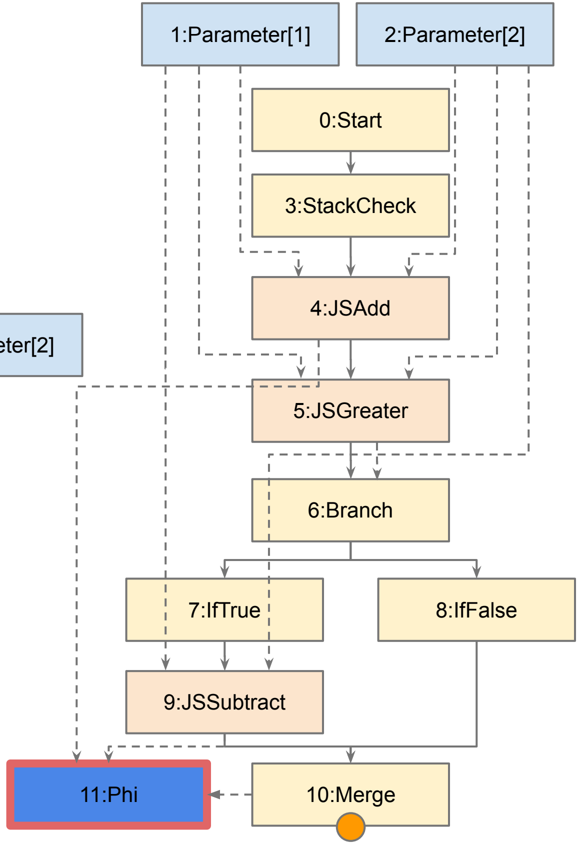
11:Phi
r0

Accumulator

9:JSSubtract

22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse



Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

Parameters

1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

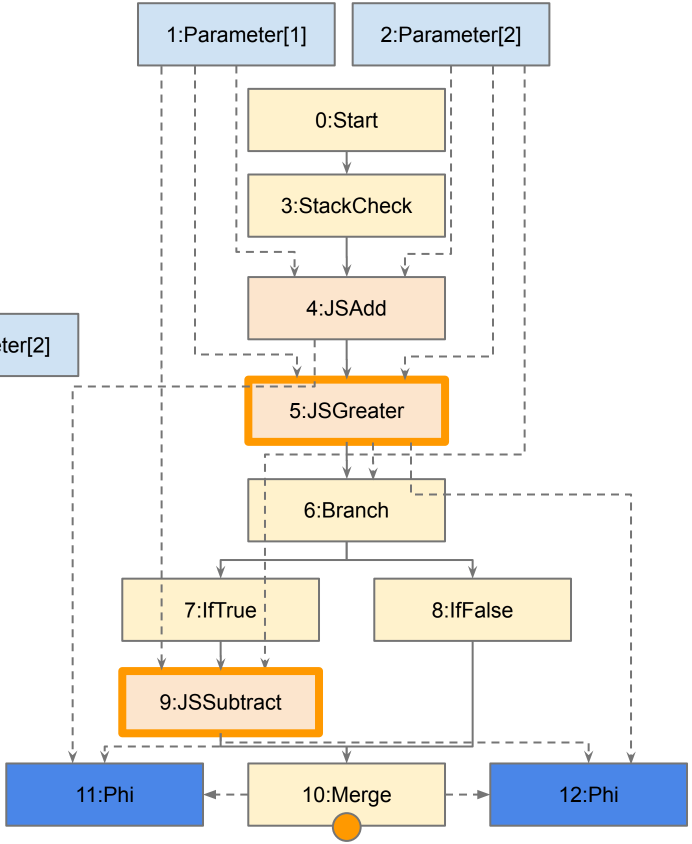
11:Phi
r0

Accumulator

9:JSSubtract

22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

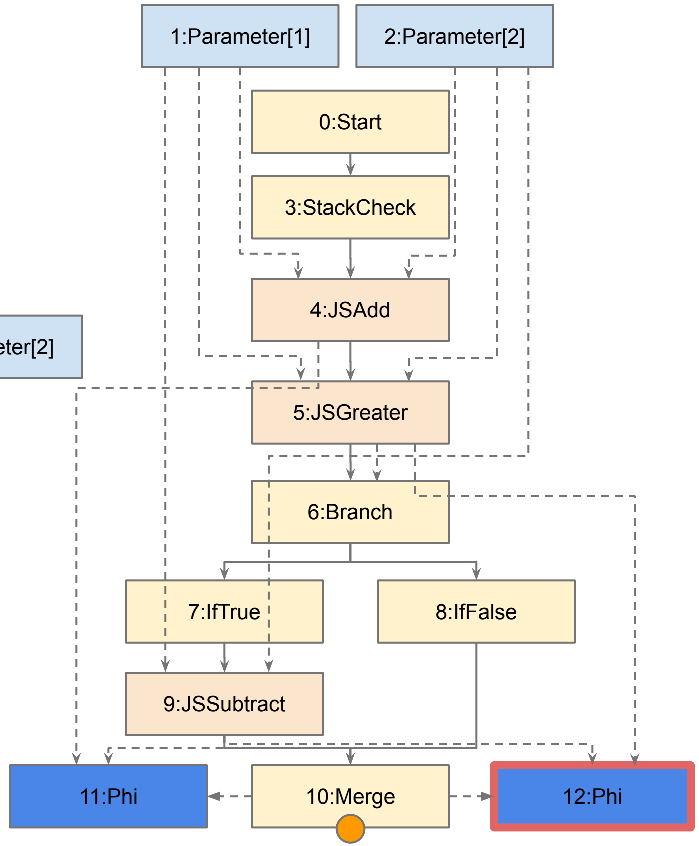
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

11:Phi
r0

Accumulator

12:Phi



22:

Registers	Accumulator	Parent
4:JSAdd	5:JSGreater	8:IfFalse
r0		

Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

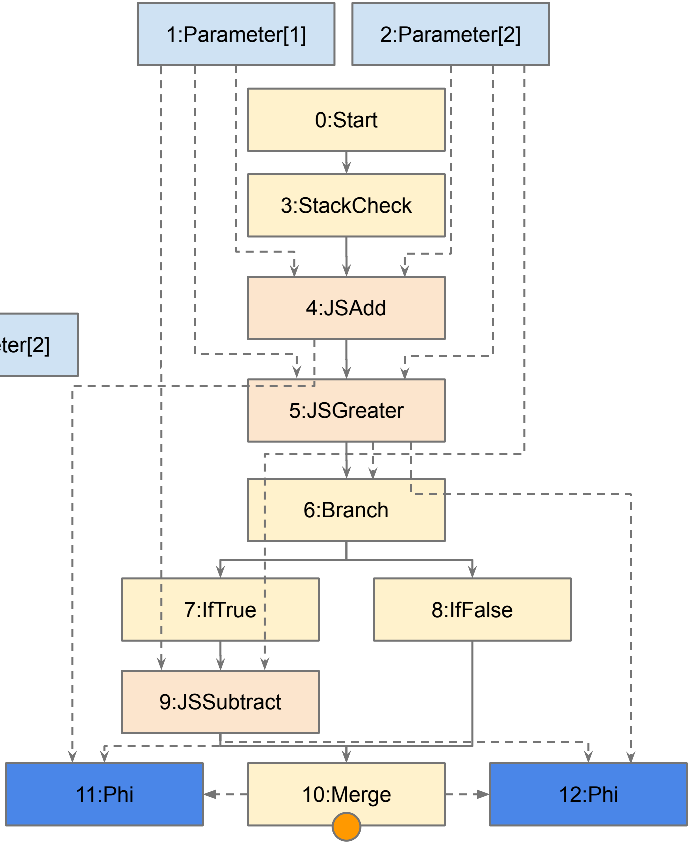
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

11:Phi
r0

Accumulator

11:Phi



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

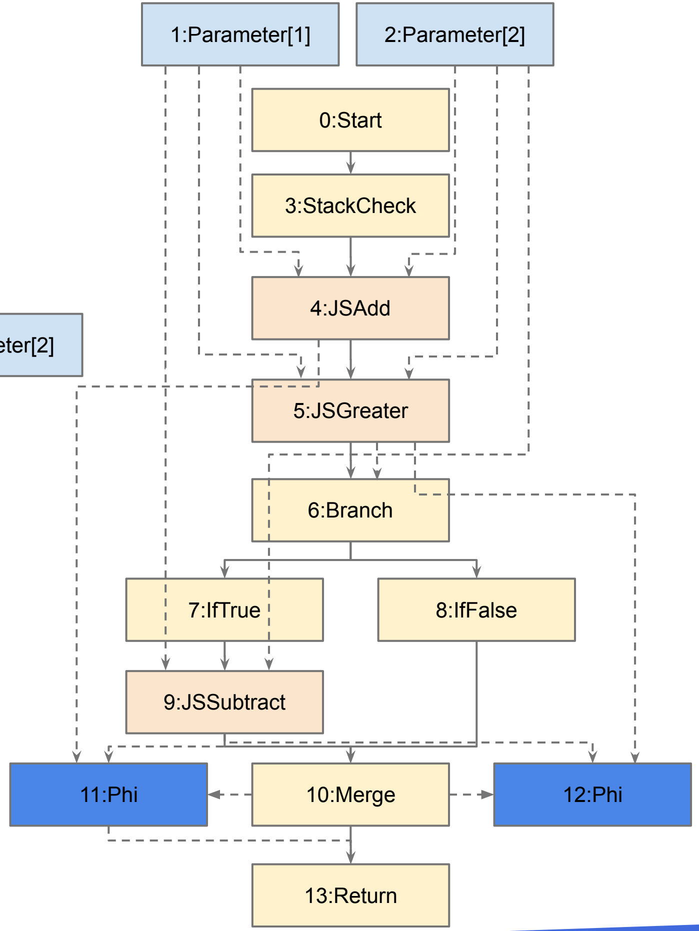
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

11:Phi
r0

Accumulator

11:Phi



Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

22:

Registers	Accumulator	Parent
4:JSAdd	5:JSGreater	8:IfFalse
r0		

Parameters

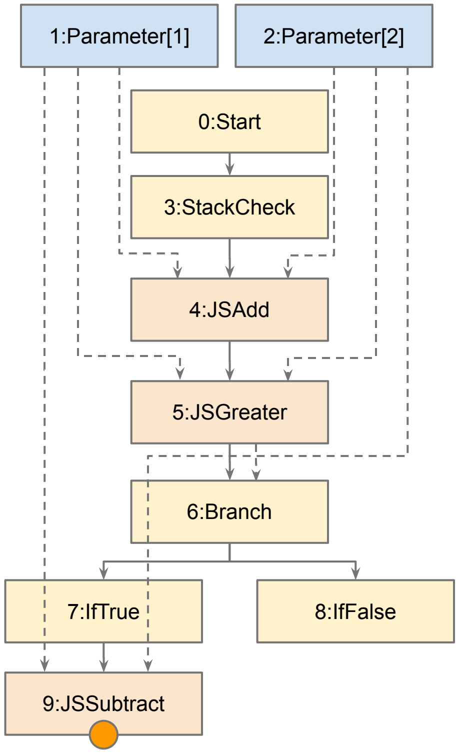
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

Accumulator

2:Parameter[2]



Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```

22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse

Parameters

1:Parameter[1]	2:Parameter[2]
a0	a1

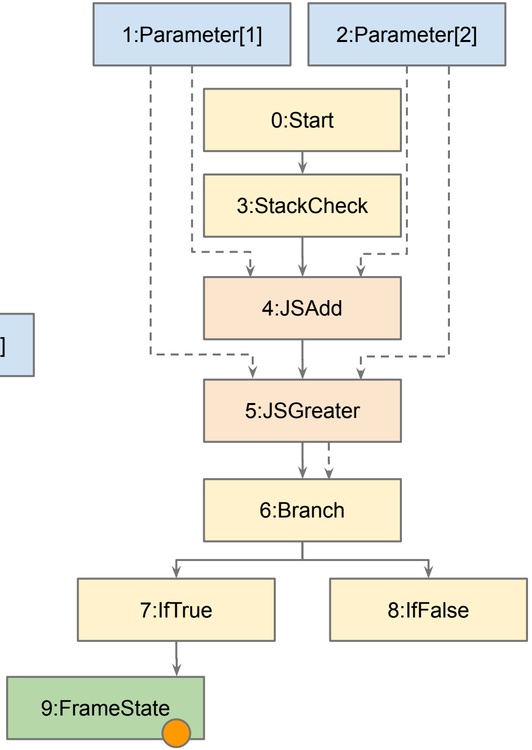
Registers

4:JSAdd

r0

Accumulator

2:Parameter[2]



1:Parameter[1]

2:Parameter[2]

0:Start

3:StackCheck

4:JSAdd

5:JSGreater

6:Branch

7:IfTrue

8:IfFalse

9:FrameState

Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse

Parameters

1:Parameter[1]	2:Parameter[2]
a0	a1

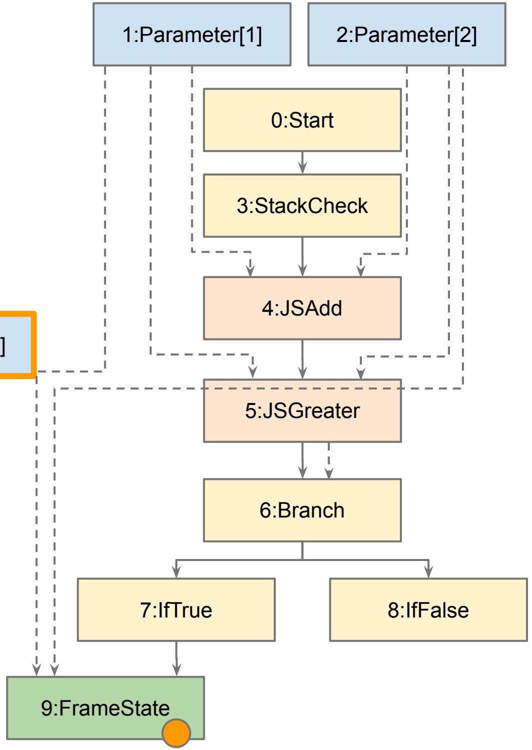
Registers

4:JSAdd

r0

Accumulator

2:Parameter[2]



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

22:

Registers	Accumulator	Parent
4:JSAdd	5:JSGreater	8:IfFalse
r0		

Parameters

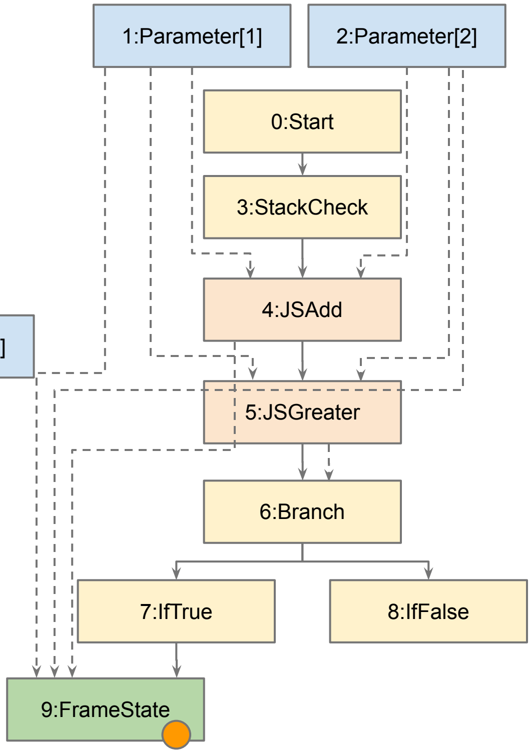
1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

4:JSAdd
r0

Accumulator

2:Parameter[2]



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

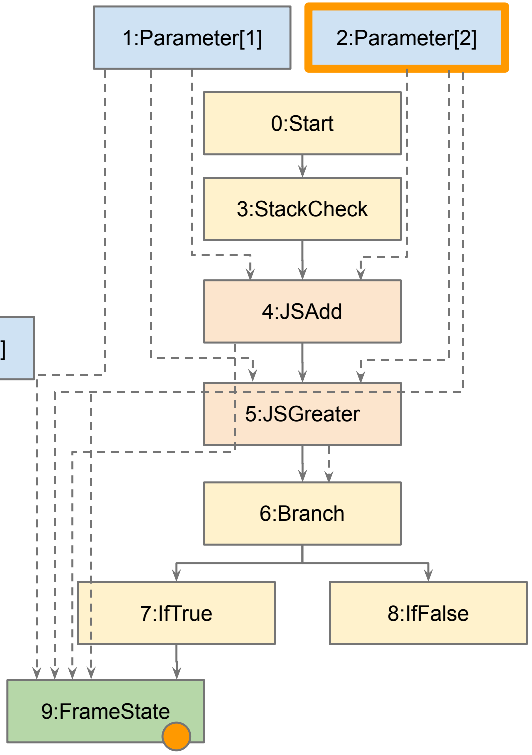
4:JSAdd
r0

Accumulator

2:Parameter[2]

22:

Registers	Accumulator	Parent
4:JSAdd	5:JSGreater	8:IfFalse
r0		



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

Parameters

1:Parameter[1]	2:Parameter[2]
a0	a1

Registers

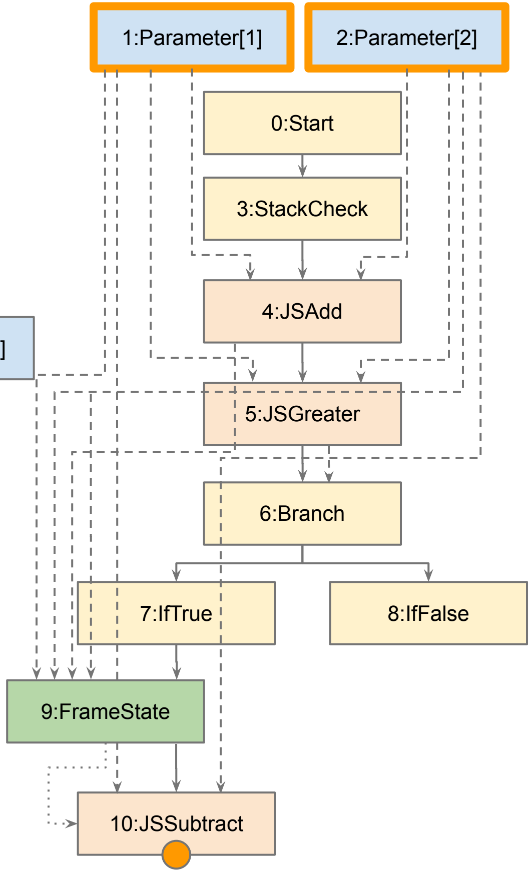
4:JSAdd
r0

Accumulator

2:Parameter[2]

22:

Registers	Accumulator	Parent
4:JSAdd r0	5:JSGreater	8:IfFalse



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```

22:

Registers	Accumulator	Parent
4:JSAdd	5:JSGreater	8:IfFalse
r0		

Parameters

1:Parameter[1]	2:Parameter[2]
a0	a1

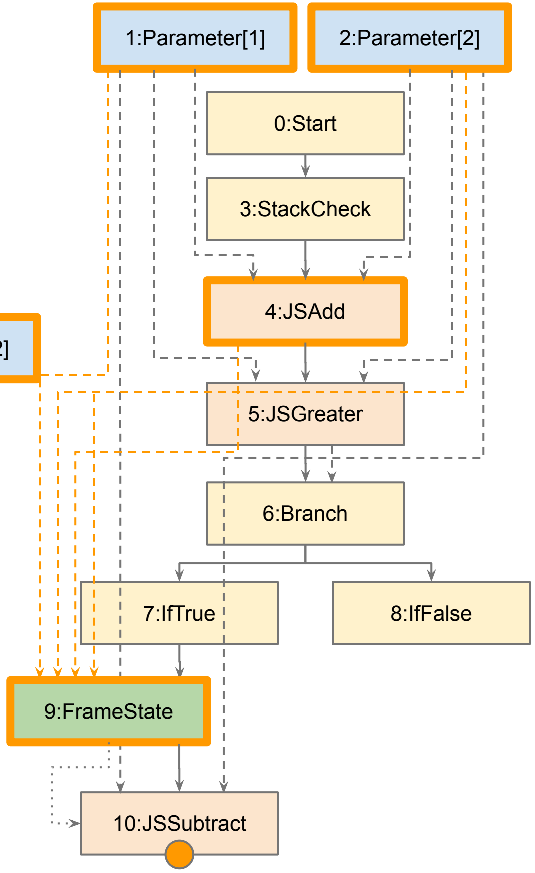
Registers

4:JSAdd

r0

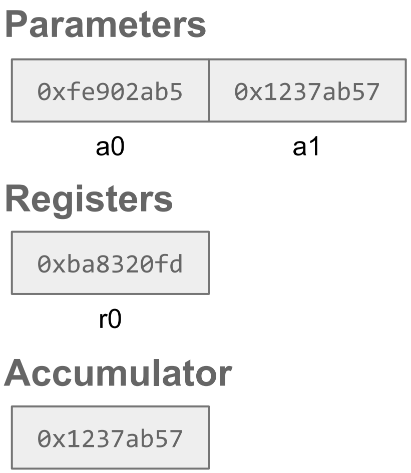
Accumulator

2:Parameter[2]



Building a graph

```
 0 : 7d          StackCheck
 1 : 1c 02       Ldar a1
 3 : 28 03 02    Add a0, [2]
 6 : 1d fa       Star r0
 8 : 1c 02       Ldar a1
10 : 4f 03 03    TestGreaterThan a0, [3]
13 : 74 09       JumpIfFalse [9] (@22)
15 : 1c 02       Ldar a1
17 : 29 03 04    Sub a0, [4]
20 : 1d fa       Star r0
22 : 1c fa       Ldar r0
24 : 81          Return
```



Parameters

0xfe902ab5	0x1237ab57
a0	a1

Registers

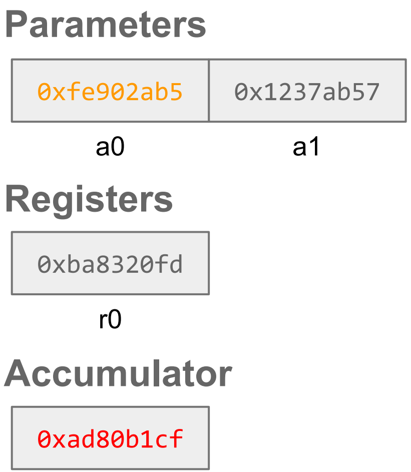
0xba8320fd
r0

Accumulator

0x1237ab57

Building a graph

```
 0 : 7d         StackCheck
 1 : 1c 02      Ldar a1
 3 : 28 03 02   Add a0, [2]
 6 : 1d fa      Star r0
 8 : 1c 02      Ldar a1
10 : 4f 03 03   TestGreaterThan a0, [3]
13 : 74 09      JumpIfFalse [9] (@22)
15 : 1c 02      Ldar a1
17 : 29 03 04   Sub a0, [4]
20 : 1d fa      Star r0
22 : 1c fa      Ldar r0
24 : 81         Return
```



Parameters

0xfe902ab5	0x1237ab57
a0	a1

Registers

0xba8320fd
r0

Accumulator

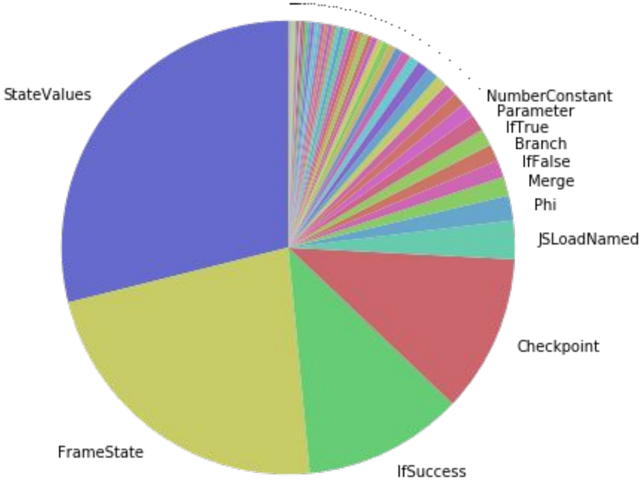
0xad80b1cf

Bytecode Restrictions to Simplifying Graph Creation

- Always deopt to a bytecode
- Well-scoped basic blocks
  - Exception handlers cover a single linear range of bytecode
- No irreducible control flow
- Single backwards-branch to loop header
- Registers in loop-closed form

Static analysis of bytecode

- Pre-analyze bytecode before building graph
  - Liveness analysis (for deoptimisation frame states)
  - Loop assignment analysis (for loop phis)
- Don’t generate unnecessary nodes
  - Avoid memory overhead (40+ bytes per node)
  - Avoid graph traversals

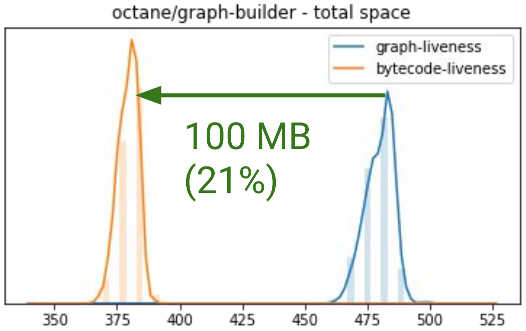


Liveness analysis

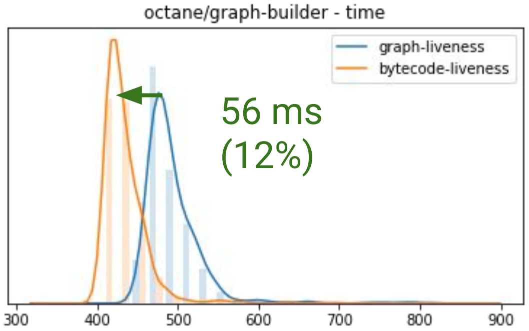
- Previously iterated traversal over basic blocks
  - Create liveness maps and state value nodes during graph building
  - Re-create state value nodes based on liveness afterward
- Now iterated passes over bytecode array
  - State value nodes created only once

Liveness analysis

- Previously iterated traversal over basic blocks
  - Create liveness maps and state value nodes during graph building
  - Re-create state value nodes based on liveness afterward
- Now iterated passes over bytecode array
  - State value nodes created only once


octane/graph-builder - total space

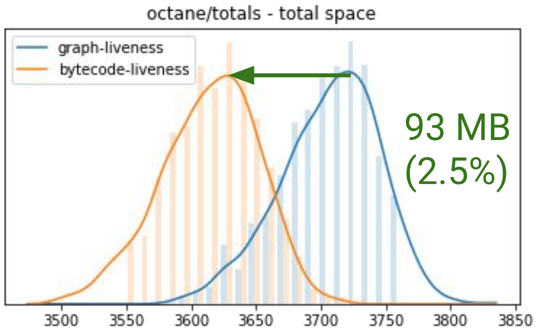
100 MB
(21%)

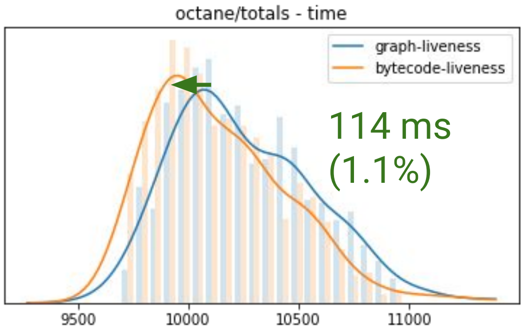

octane/graph-builder - time

56 ms
(12%)

Liveness analysis

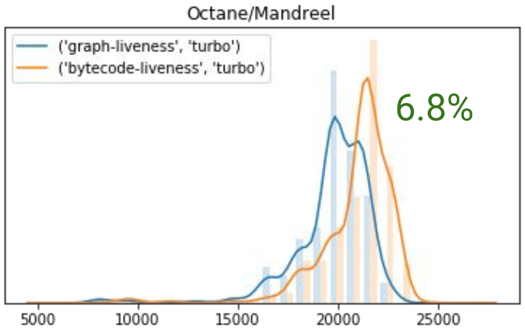
- Previously iterated traversal over basic blocks
  - Create liveness maps and state value nodes during graph building
  - Re-create state value nodes based on liveness afterward
- Now iterated passes over bytecode array
  - State value nodes created only once

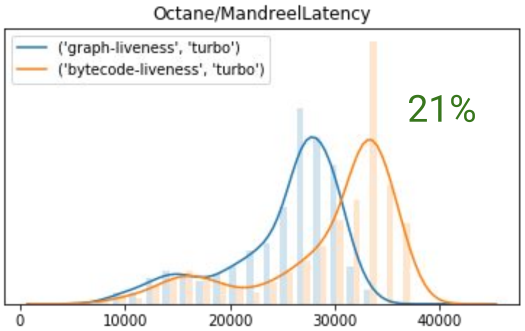




Liveness analysis

- Previously iterated traversal over basic blocks
  - Create liveness maps and state value nodes during graph building
  - Re-create state value nodes based on liveness afterward
- Now iterated passes over bytecode array
  - State value nodes created only once





Burning away complex control (Generators)

- Javascript Generators can `yield` expressions at arbitrary points
    - Can introduce irreducible control flow
    - Solution: transform into switch statements on hidden token in prologue and loop headers
    - Result: Turbofan doesn't need to know anything about generator control flow

Burning away complex control (Generators)

```
function* f() {
    var i = 0;
    while (true) {
        yield i++;
    }
}
```

```
function f(hidden_token) {
    switch(hidden_token)
        case %normal%: break;
        case %resume%: goto loop;
    var i = 0;
    loop:
        switch(hidden_token)
            case %normal%: break;
            case %resume%: goto resume;
        hidden_token = %resume%;
        return i++;
    resume:
        hidden_token = %normal%;
        goto loop;
}
```

Burning away complex control (`try-finally`)

- `try-finally` exception handlers
    - Exit differently depending on what triggered the finally block (fall-through, return or throw)
    - Solution: switch statement at end of finally block
    - Result: Turbofan doesn't need to know anything about `try-finally` control flow

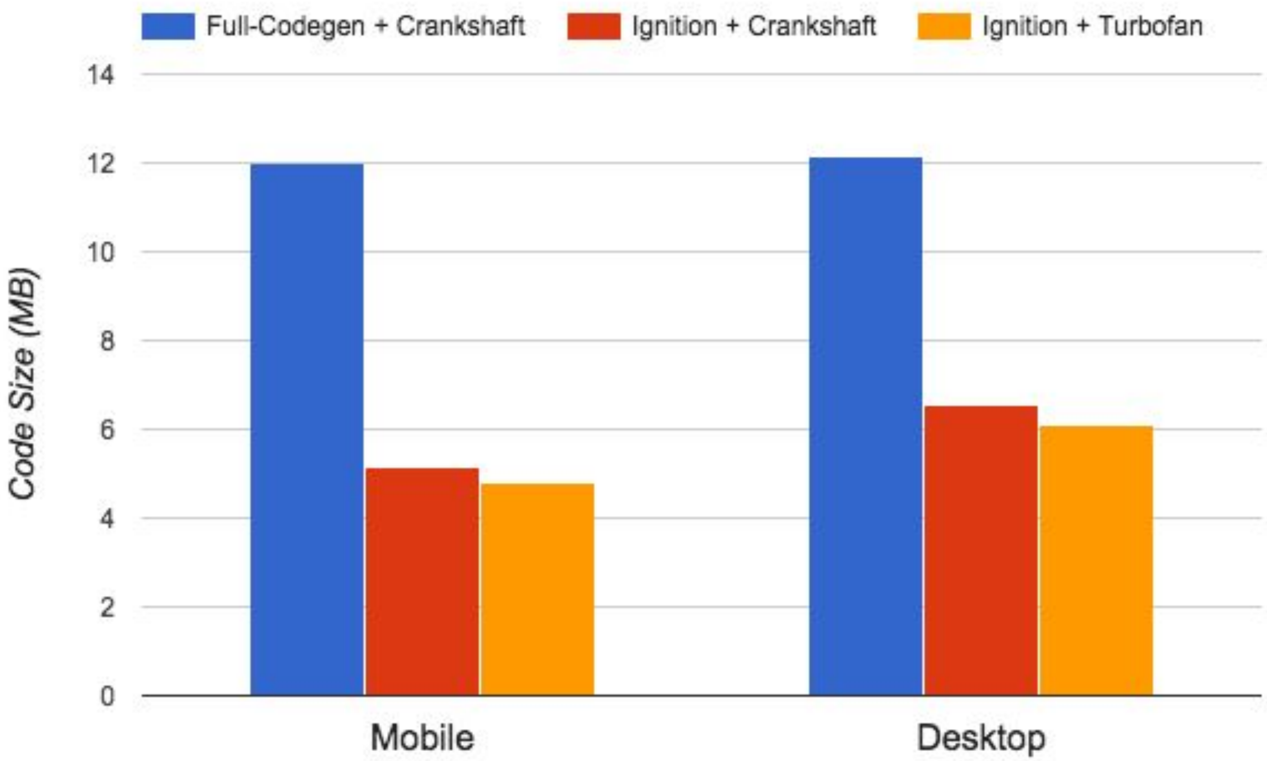
Burning away complex control (try-finally)

```
try {
    if a() return;
} finally {
    b();
}
```

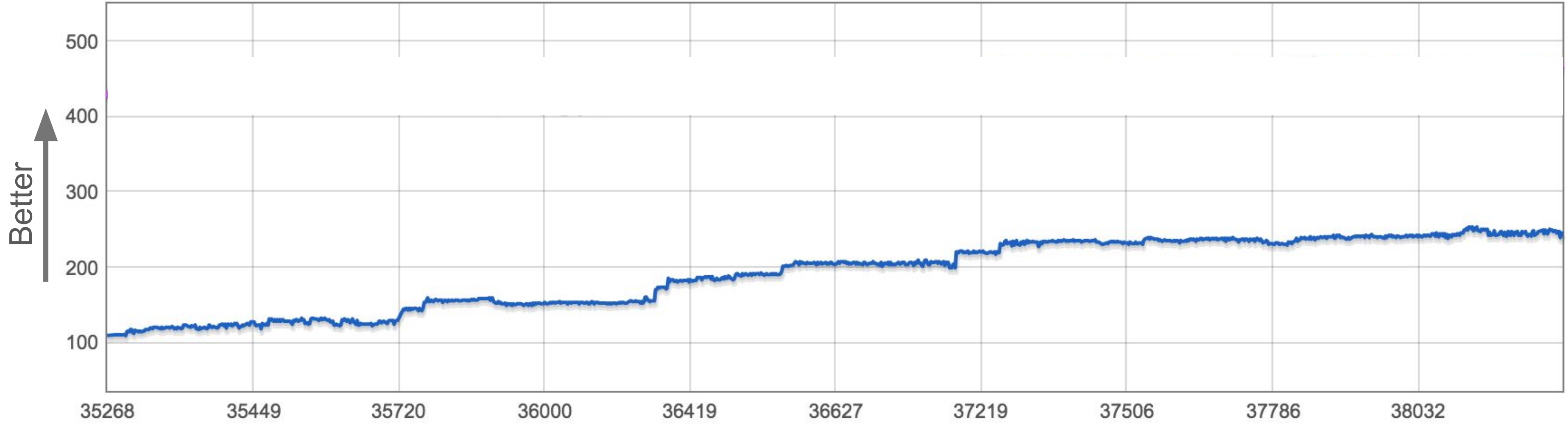
```
 try:
    if a() {
        finally_token = %return%;
        goto finally;
    }
    finally_token = %fallthrough%;
    goto finally;

throw_handler:
    finally_token = %throw%;

finally:
    b();
    switch(finally_token)
        case %return%: return;
        case %throw%: rethrow();
        case %fallthrough%: break;
```

Performance Results

Code Memory Usage (Real Websites)



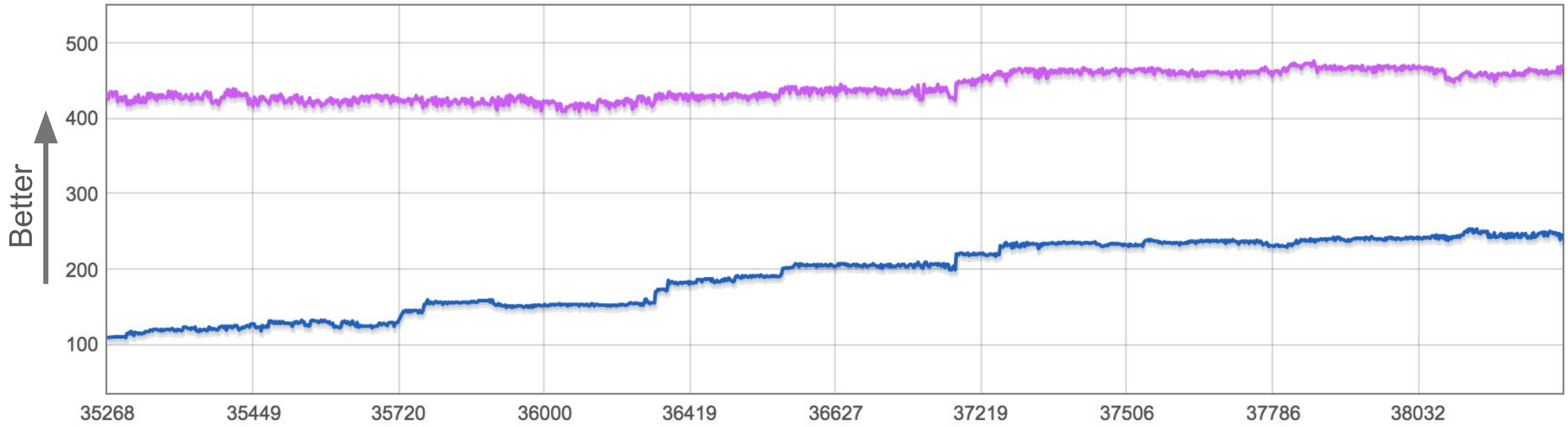
Ignition vs Full-Codegen



Octane (Nexus 5)

Crankshaft and TurboFan disabled

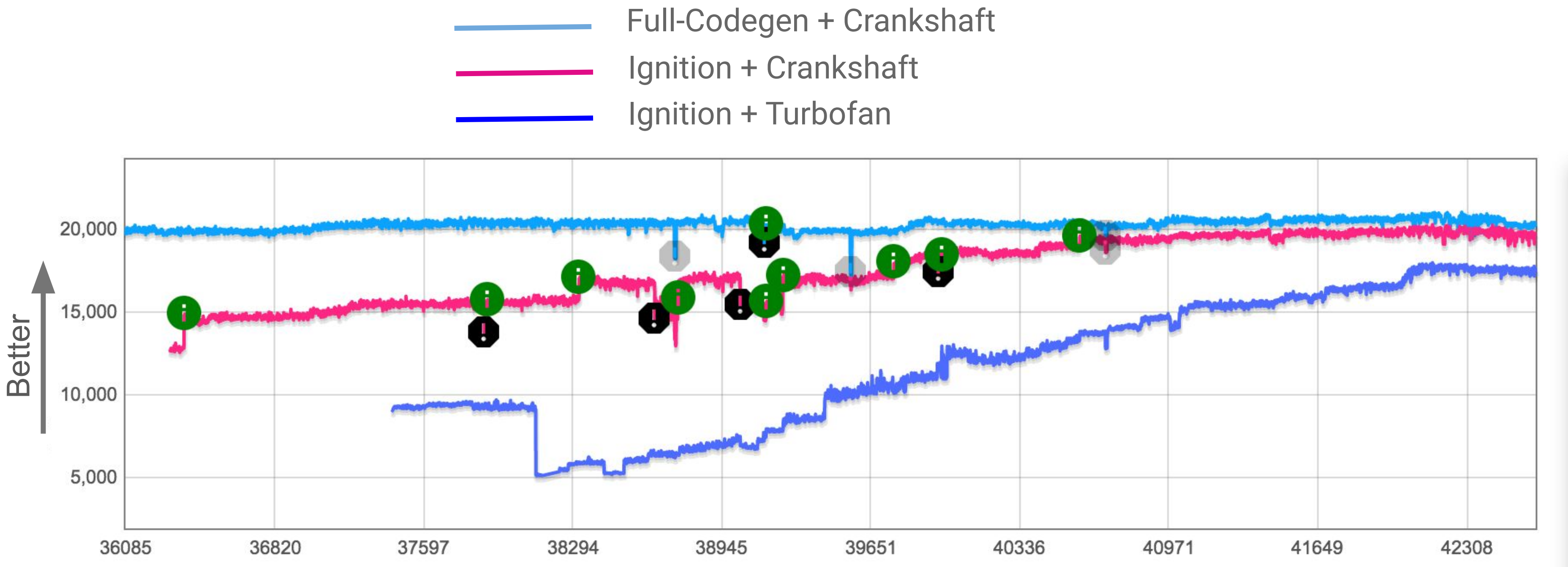
Ignition vs Full-Codegen



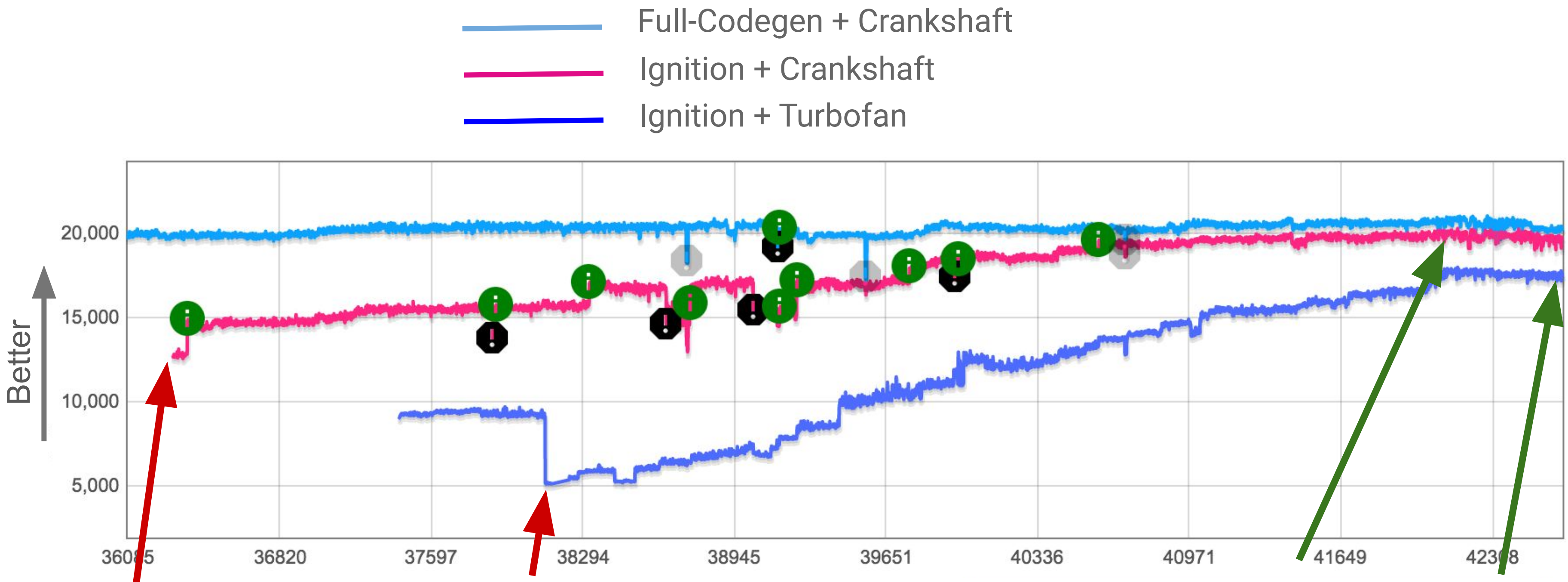
Octane (Nexus 5)

Crankshaft and TurboFan disabled

Octane Performance



Octane Performance



40% slower

75% slower

<5% slower

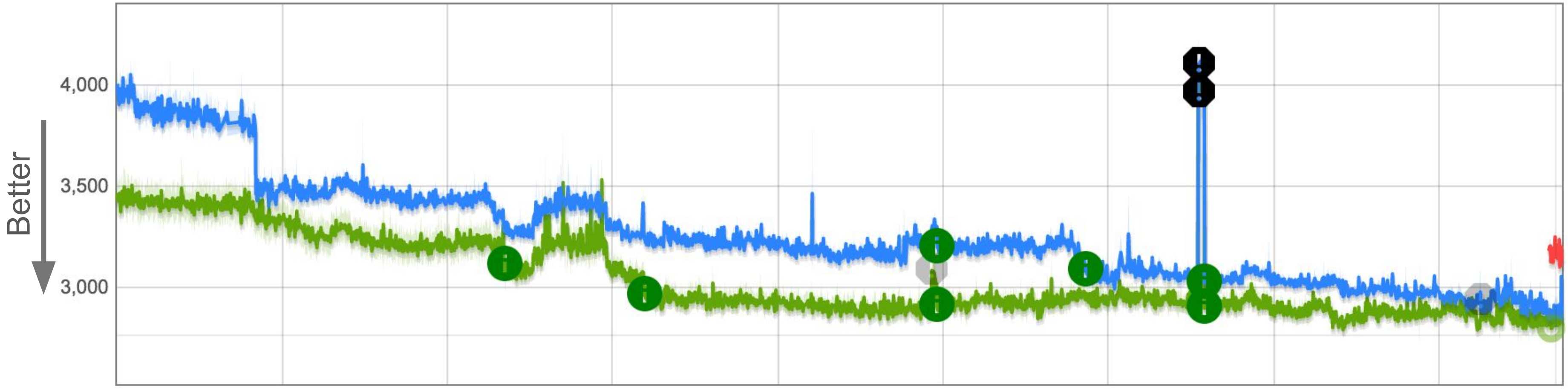
15% slower

Speedometer Performance

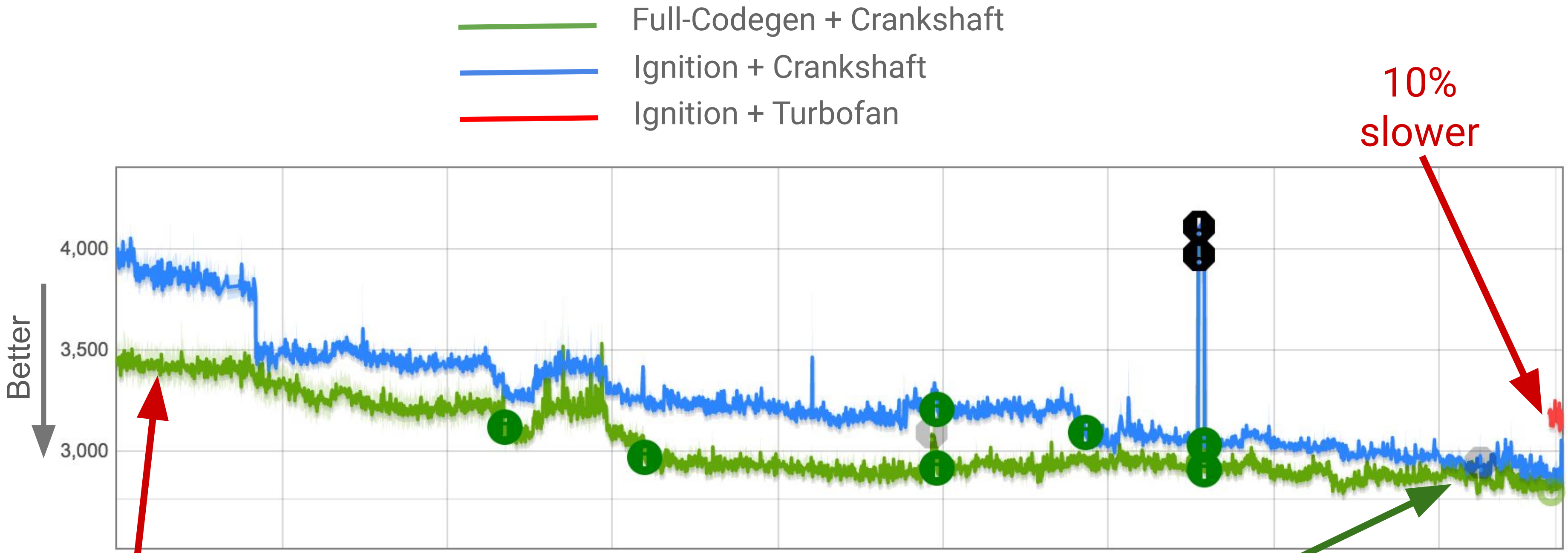
🟩 Full-Codegen + Crankshaft

🟦 Ignition + Crankshaft

🟥 Ignition + Turbofan



Speedometer Performance

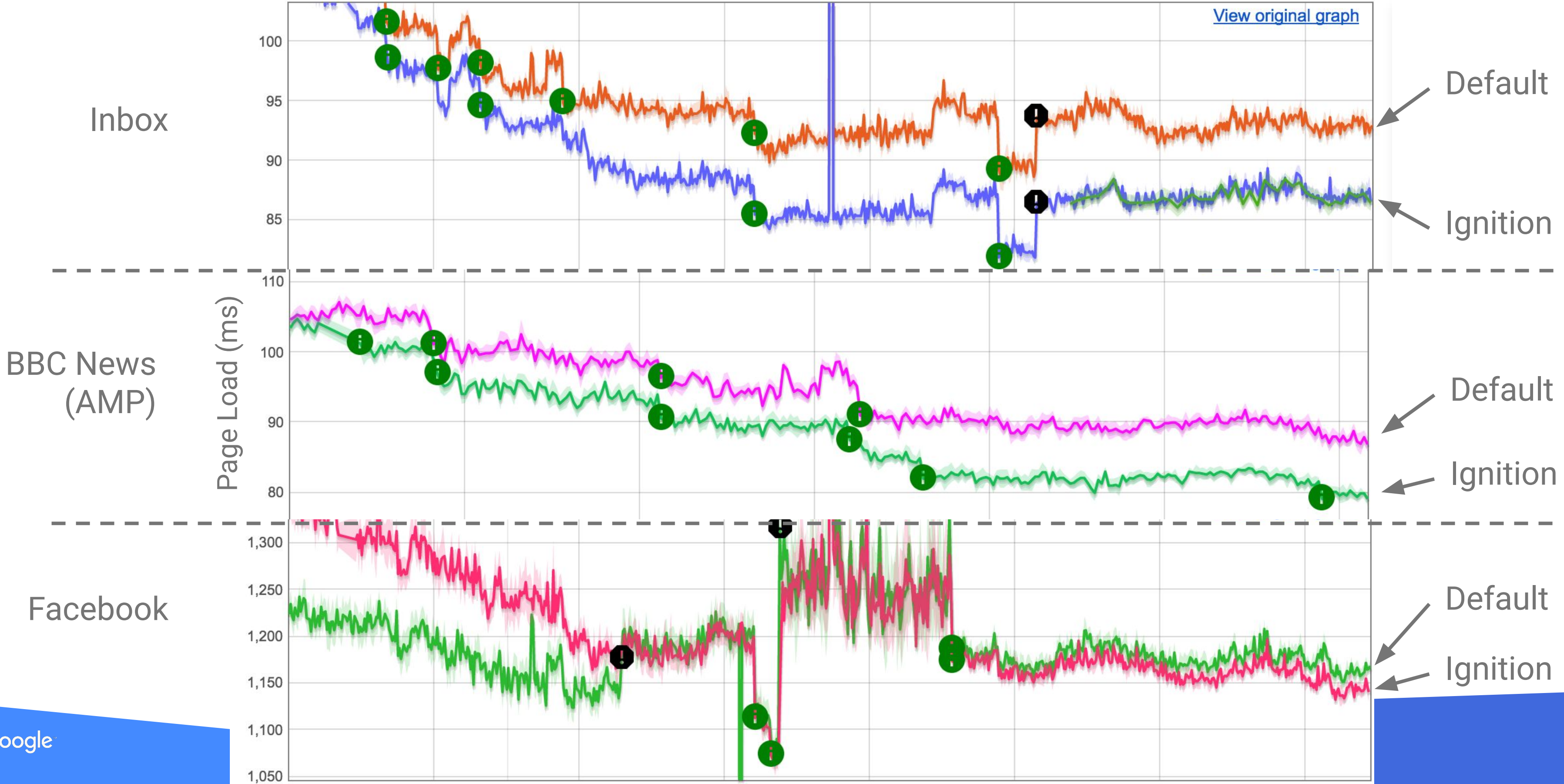


10% slower

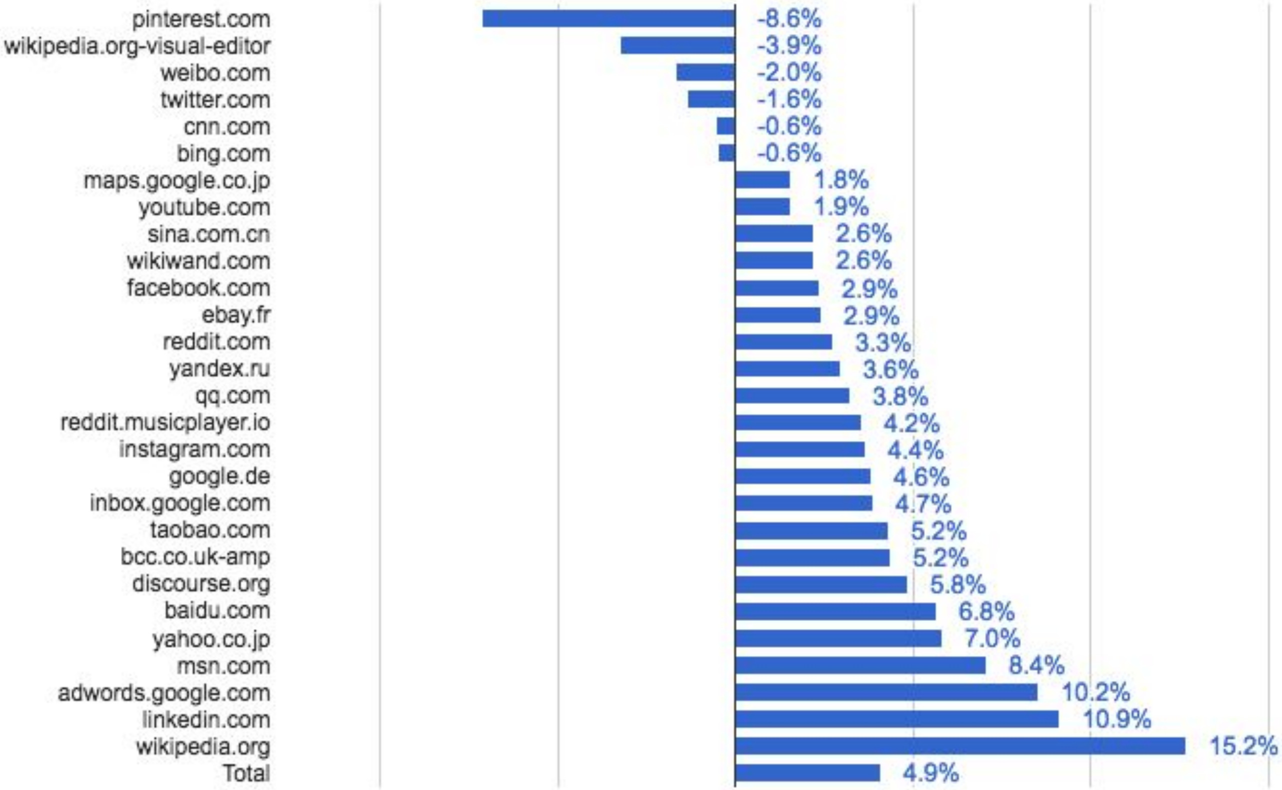
14% slower

<1% slower

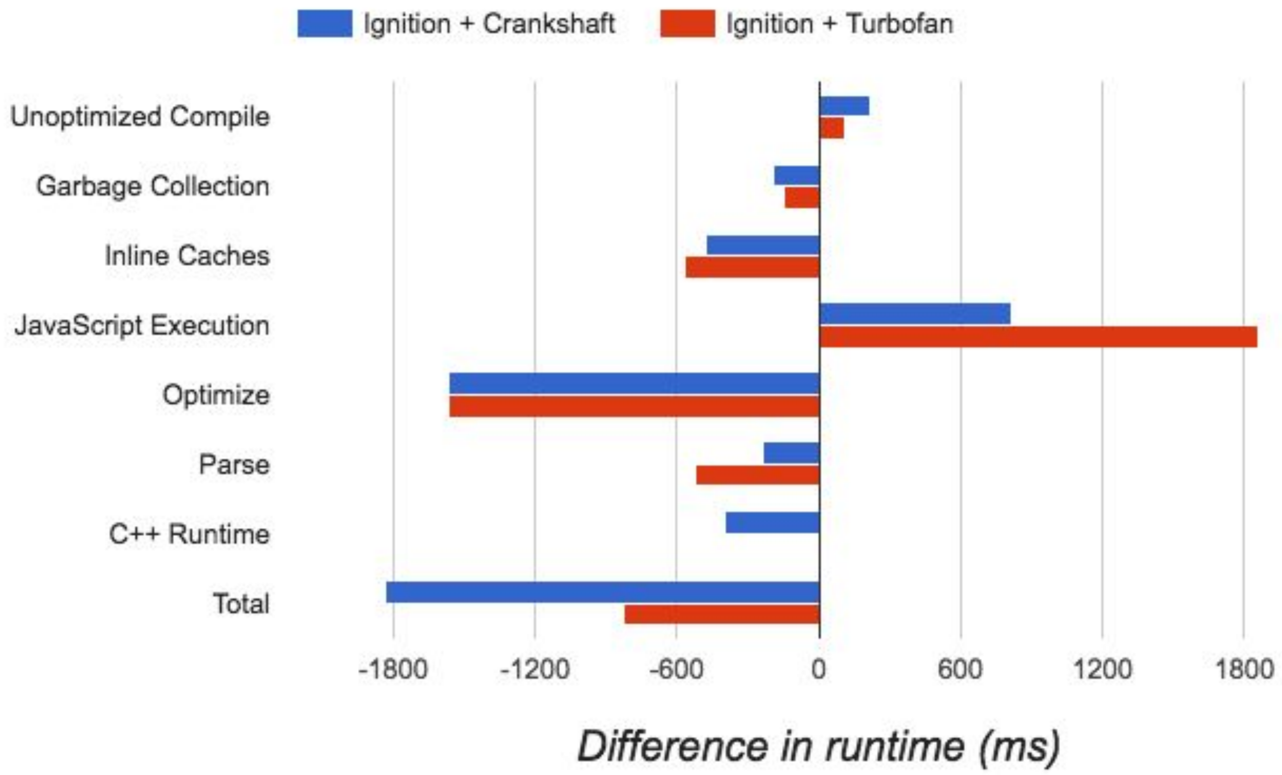
Startup Time (Real Websites)



Startup Time (Real Websites)



Where time is spent (Real Websites)



Summary

- Ignition + Turbofan is the future of V8
- Restrictions on bytecode can simplify optimized graph creation
- Optimizing for real-world exposes different trade-offs