CodeGen Overview and Focus on SelectionDAGs

Dan Gohman

What does CodeGen do?

* A very large analysis
* Input: LLVM IR
* Output: Machine Code
  * optimize for "efficient" generated code
  * run quickly

CodeGen Phases

* Preparation
* SelectionDAG
  * Instruction Selection and Scheduling
* Register Allocation
* Output

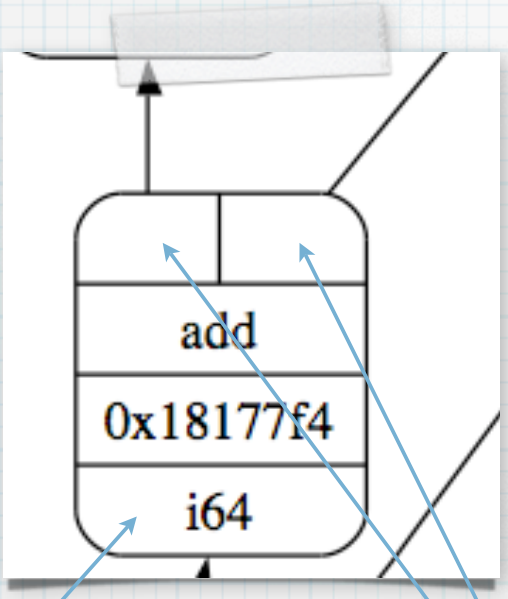
Intro

Input: LLVM IR

```
bb:
   %i = phi i32 [ 0, %entry ], [ %i.next, %bb ]
   %sum = phi double [ 0.0, %entry ], [ %sum.next, %bb ]

   %t0 = getelementptr i64* %x, i32 %i
   %t1 = load i64* %t0
   %t2 = mul i64 %t1, 101
   %t3 = add i64 %t2, 3
   store i64 %t3, i64* %t0

   %t4 = getelementptr double* %y, i32 %i
   %t5 = load double* %t4
   %sum.next = add double %sum, %t5

   %i.next = add i32 %i, 1
   %exitcond = icmp eq i32 %i.next, %n
   br i1 %exitcond, label %return, label %bb
```

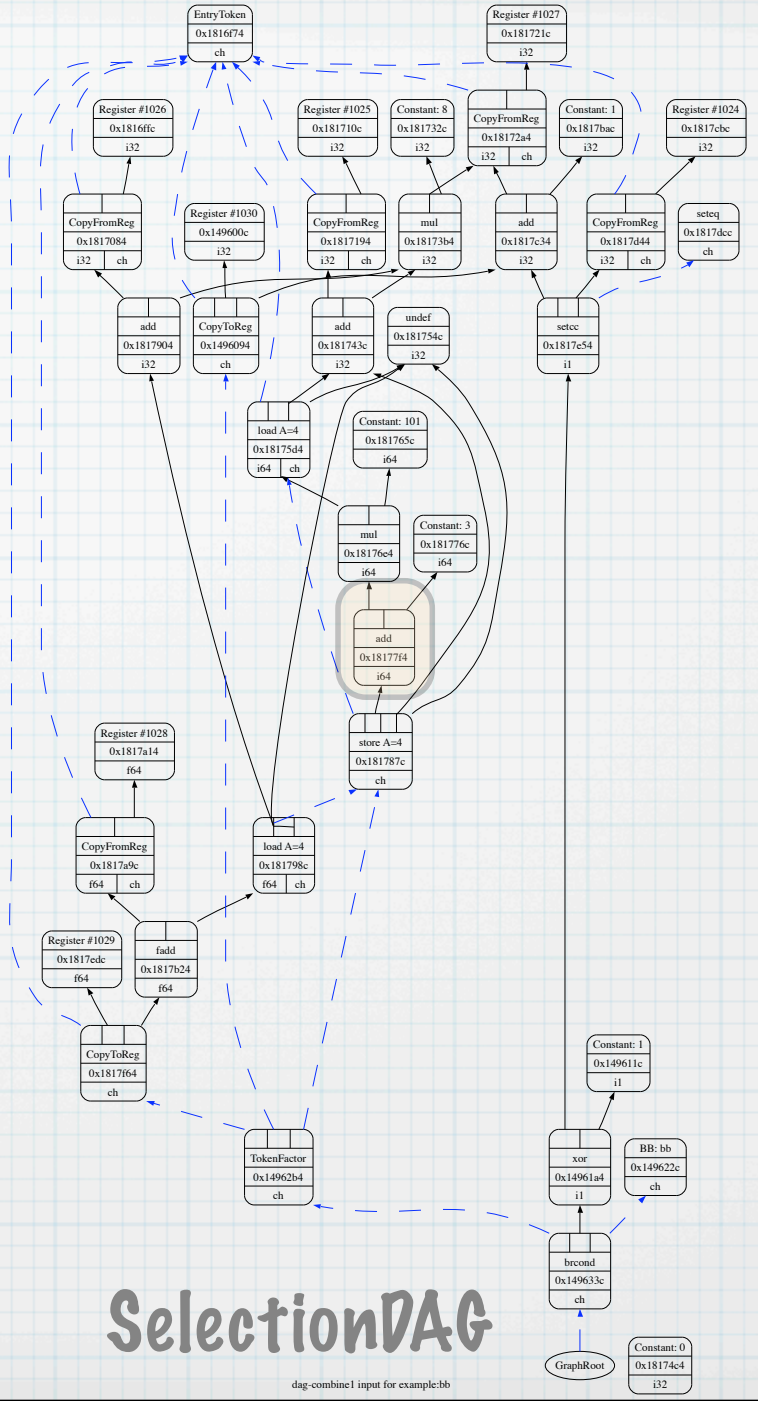
Intro

SDNode



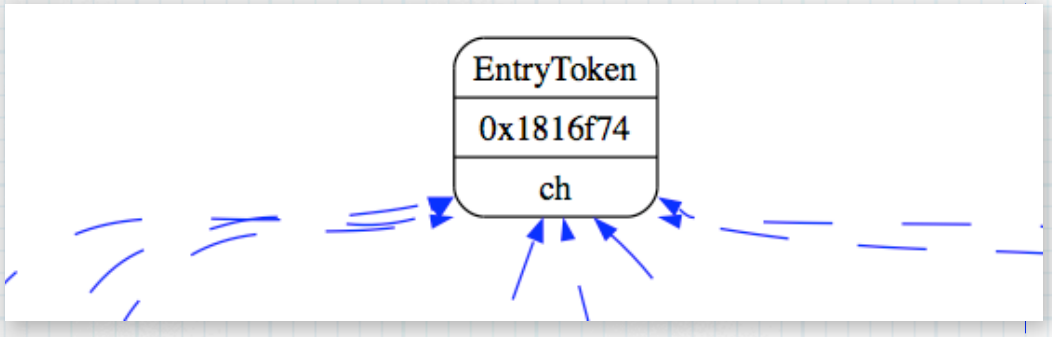
SDValue

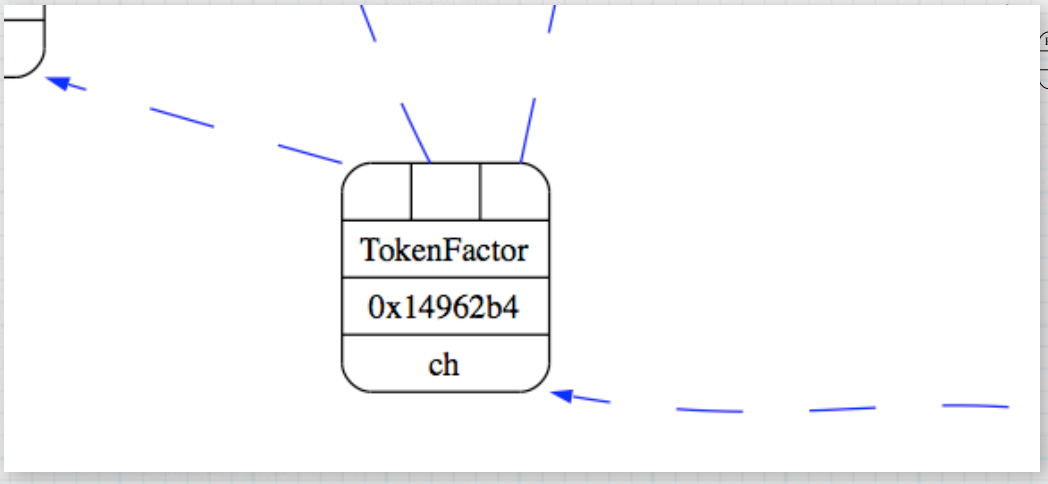
SDUse

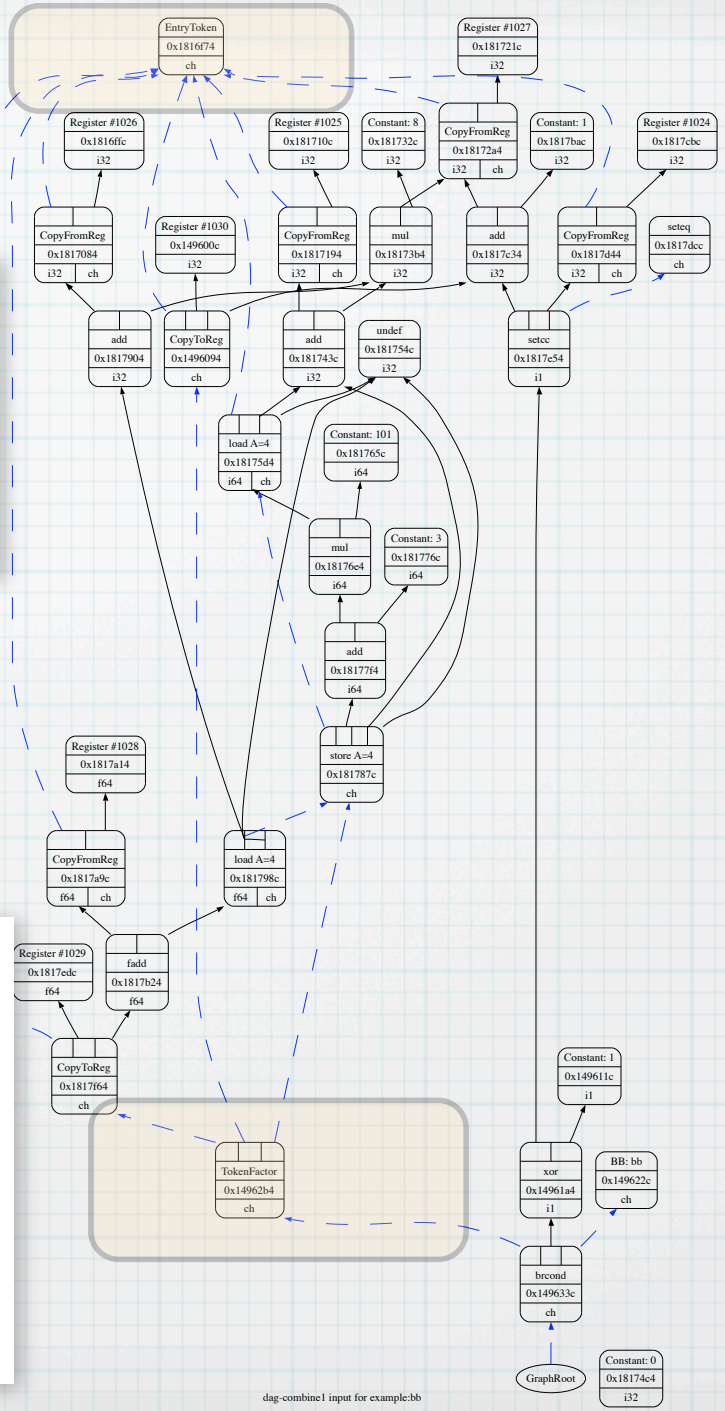


SelectionDAG

Intro

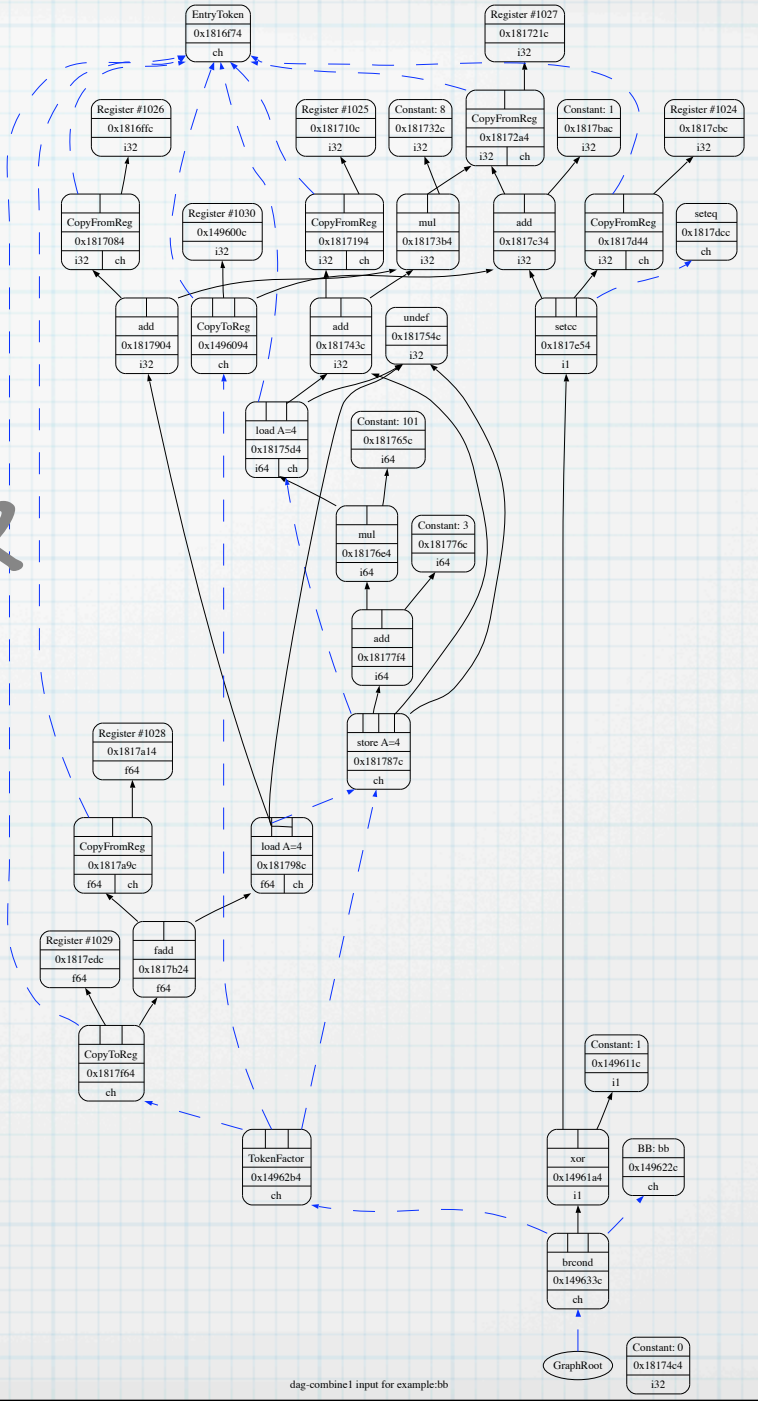

EntryToken
0x1816f74
ch


TokenFactor
0x14962b4
ch


EntryToken
0x1816f74
ch
Register #1027
0x181721c
i32
Register #1026
0x1816ffc
i32
Register #1025
0x181710c
i32
Constant: 8
0x181732c
i32
CopyFromReg
0x18172a4
i32 ch
Constant: 1
0x1817bac
i32
Register #1024
0x1817cbc
i32
CopyFromReg
0x1817084
i32 ch
Register #1030
0x149600c
i32
CopyFromReg
0x1817194
i32 ch
mul
0x18173b4
i32
add
0x1817c34
i32
CopyFromReg
0x1817d44
i32 ch
seteq
0x1817dcc
ch
add
0x1817904
i32
CopyToReg
0x1496094
ch
add
0x181743c
i32
undef
0x181754c
i32
setcc
0x1817e54
i1
load A=4
0x18175d4
i64 ch
Constant: 101
0x181765c
i64
mul
0x18176e4
i64
Constant: 3
0x181776c
i64
add
0x18177f4
i64
store A=4
0x181787c
ch
Register #1028
0x1817a14
f64
CopyFromReg
0x1817a9c
f64 ch
load A=4
0x181798c
f64 ch
Register #1029
0x1817edc
f64
fadd
0x1817b24
f64
CopyToReg
0x1817f64
ch
TokenFactor
0x14962b4
ch
Constant: 1
0x149611c
i1
xor
0x14961a4
i1
BB: bb
0x149622c
ch
brcond
0x149633c
ch
GraphRoot
Constant: 0
0x18174c4
i32
dag-combine1 input for example:bb

Intro

* Unique use of DAG-based IR
* Automatic CSE
* Lower Level than LLVM
* llc -view-*-dags

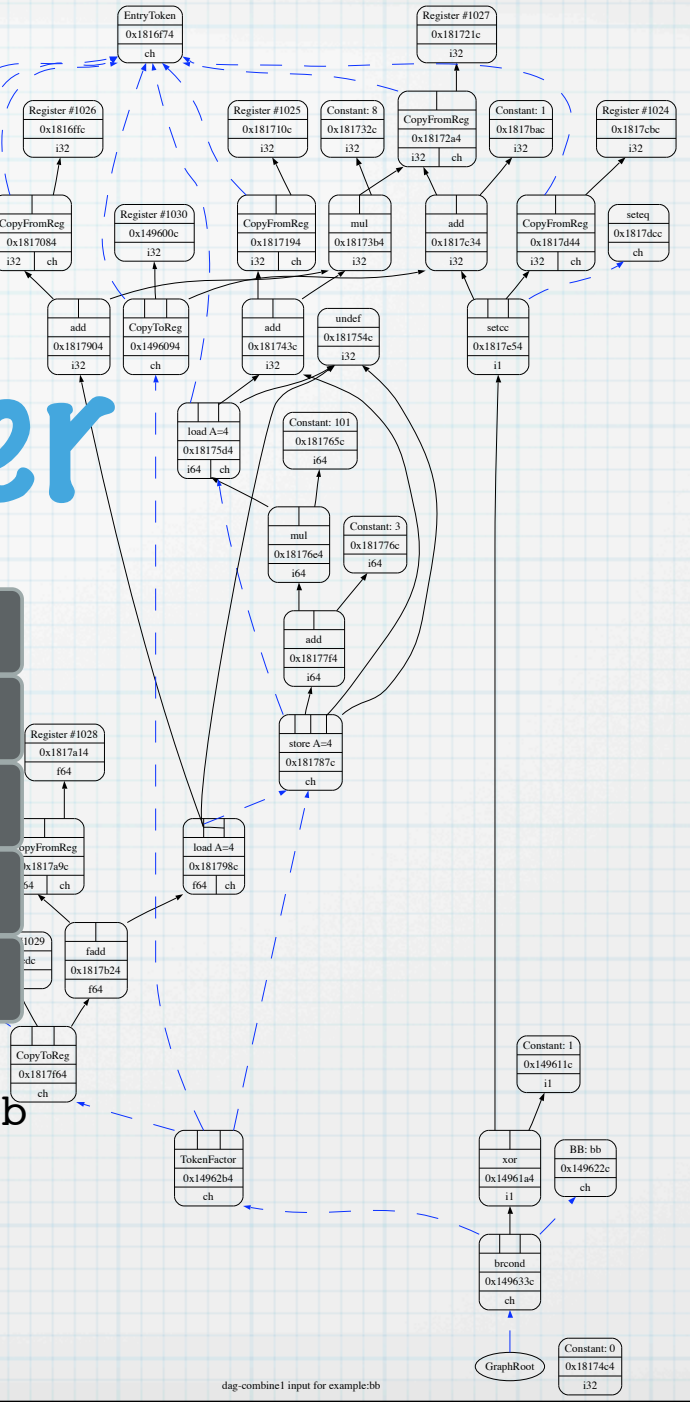


SelectionDAG Phases

```
bb:
 %i = phi i32 [ 0, %entry ],
              [ %i.next, %bb ]
 %sum = phi double [ 0.0, %entry ],
                   [ %sum.next, %bb ]

 %t0 = getelementptr i64* %x, i32 %i
 %t1 = load i64* %t0
 %t2 = mul i64 %t1, 101
 %t3 = add i64 %t2, 3
 store i64 %t3, i64* %t0

 %t4 = getelementptr double*
 %t5 = load double* %t4
 %sum.next = add double %sum,

 %i.next = add i32 %i, 1
 %exitcond = icmp eq i32 %i.next, %n
 br i1 %exitcond, label %return, label %bb
```

Lower

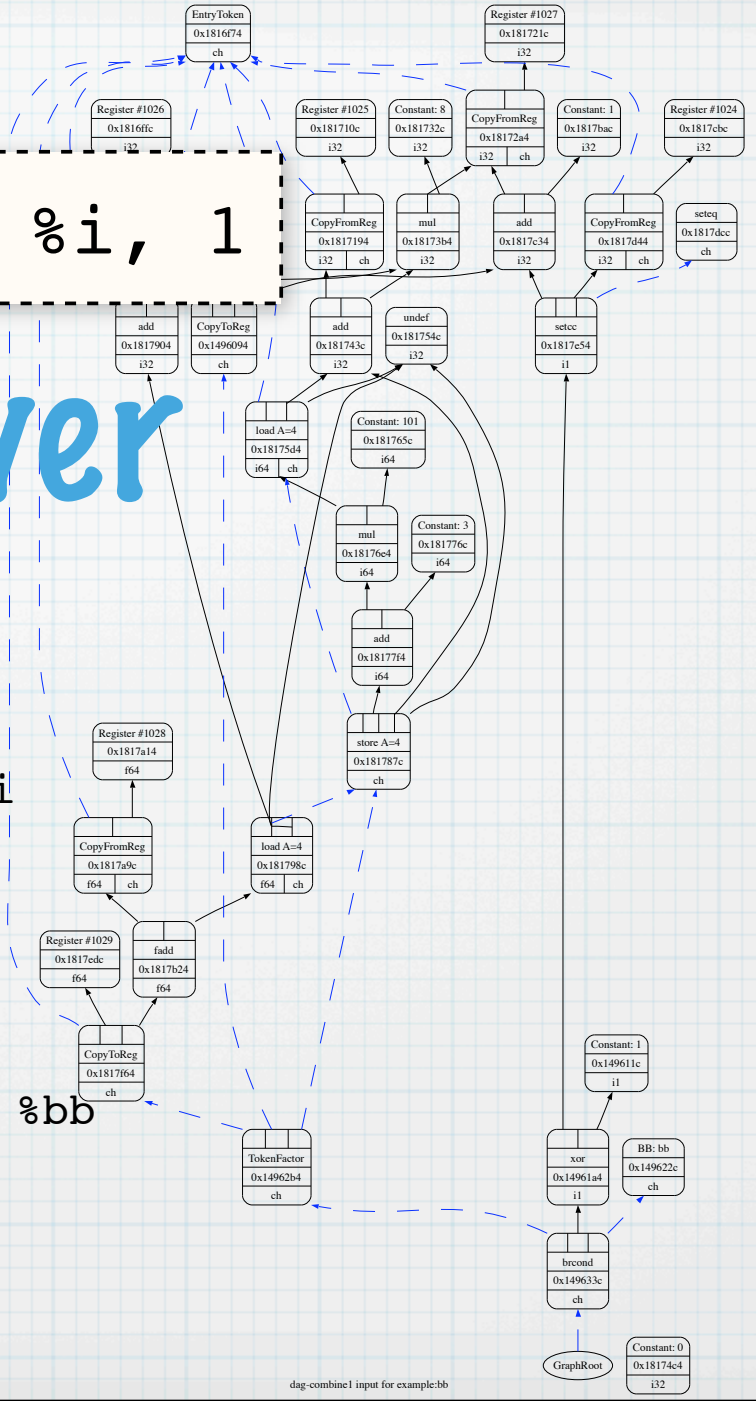
- Combine
- Legalize
- Combine
- Select
- Schedule



```
bb:
 %i = phi i32 [ 0, %entry
              [ %i.next,
 %sum = phi double [ 0.0, %entry ],
                   [ %sum.next, %bb ]

 %t0 = getelementptr i64* %x,
 %t1 = load i64* %t0
 %t2 = mul i64 %t1, 101
 %t3 = add i64 %t2, 3
 store i64 %t3, i64* %t0

 %t4 = getelementptr double* %y, i32 %i
 %t5 = load double* %t4
 %sum.next = add double %sum, %t5

 %i.next = add i32 %i, 1
 %exitcond = icmp eq i32 %i.next, %n
 br i1 %exitcond, label %return, label %bb
```

add i32 %i, 1

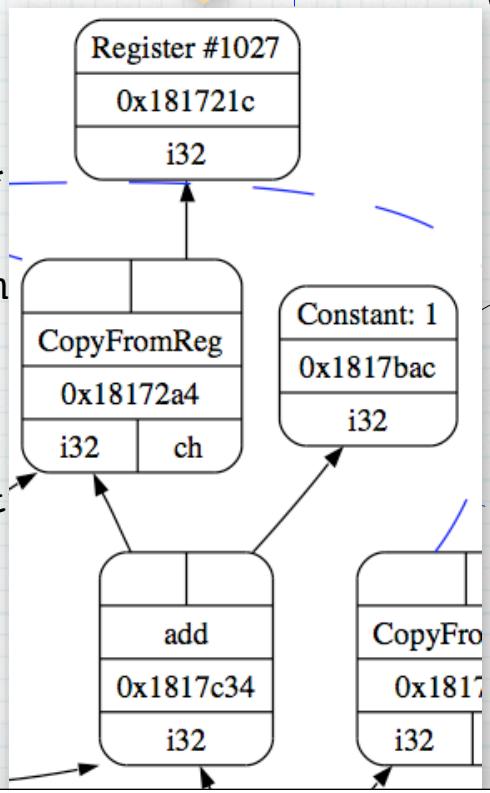
Lower

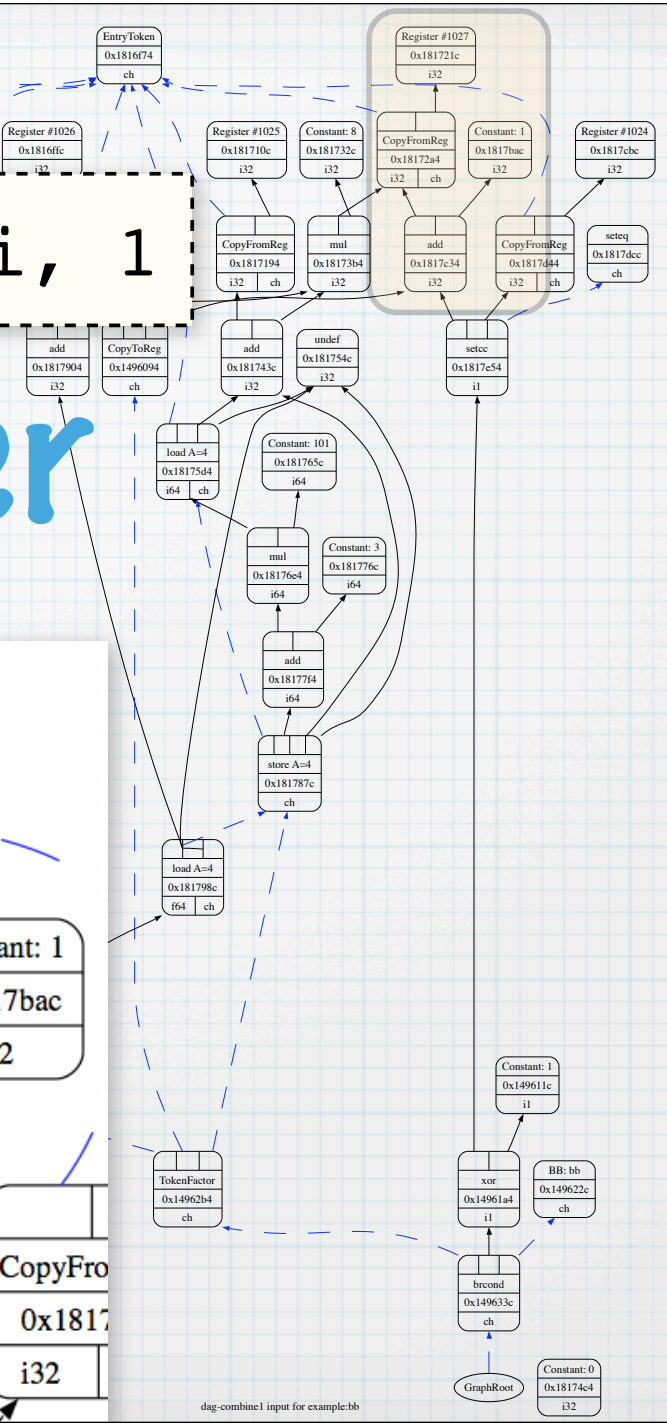


```
bb:
 %i = phi i32 [ 0, %entry
              [ %i.next,
 %sum = phi double [ 0.0, %entry ],
                   [ %sum.next, %bb ]

 %t0 = getelementptr i64* %x,
 %t1 = load i64* %t0
 %t2 = mul i64 %t1, 101
 %t3 = add i64 %t2, 3
 store i64 %t3, i64* %t0

 %t4 = getelementptr double*
 %t5 = load double* %t4
 %sum.next = add double %sum

 %i.next = add i32 %i, 1
 %exitcond = icmp eq i32 %i.
 br i1 %exitcond, label %ret
```

add i32 %i, 1

Lower

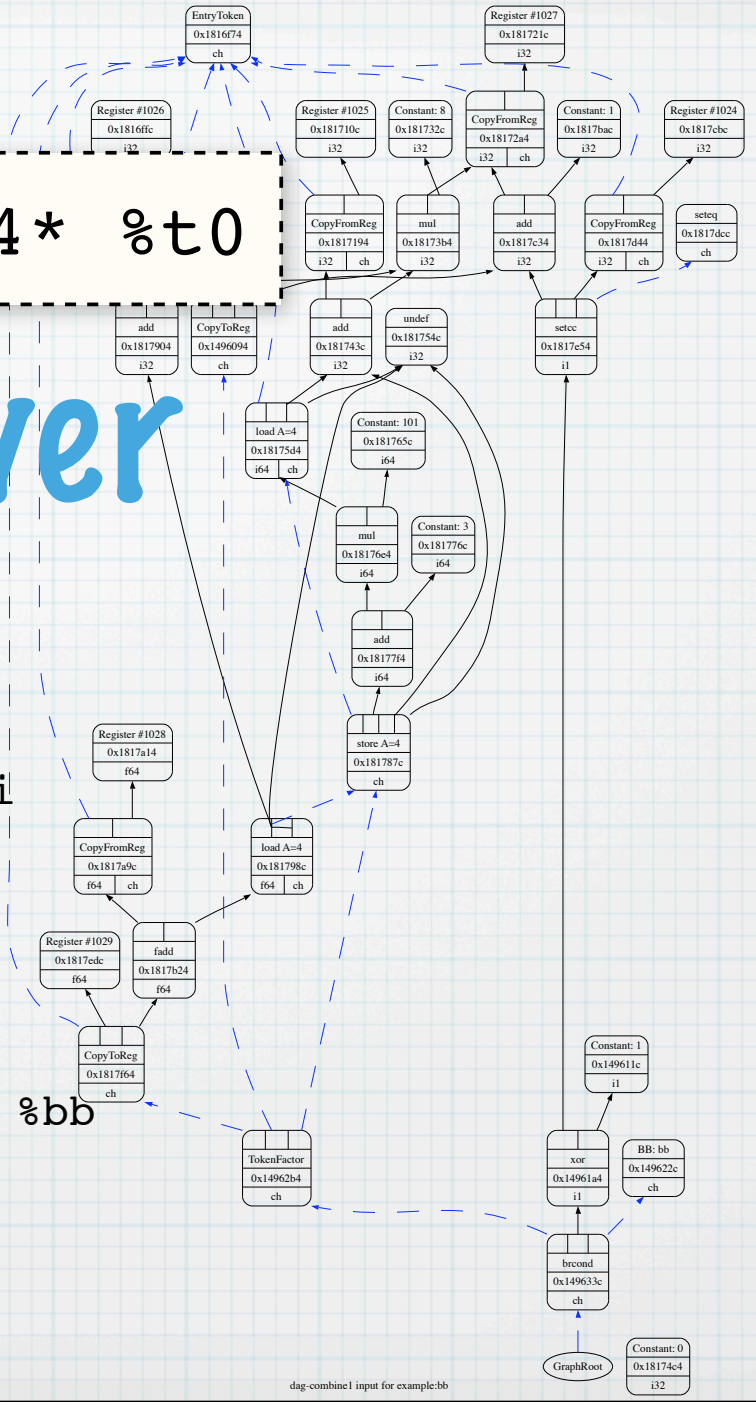




```
bb:
 %i = phi i32 [ 0, %entry
              [ %i.next,
 %sum = phi double [ 0.0, %entry ],
                   [ %sum.next, %bb ]

 %t0 = getelementptr i64* %x,
 %t1 = load i64* %t0
 %t2 = mul i64 %t1, 101
 %t3 = add i64 %t2, 3
 store i64 %t3, i64* %t0

 %t4 = getelementptr double* %y, i32 %i
 %t5 = load double* %t4
 %sum.next = add double %sum, %t5

 %i.next = add i32 %i, 1
 %exitcond = icmp eq i32 %i.next, %n
br i1 %exitcond, label %return, label %bb
```

load i64* %t0

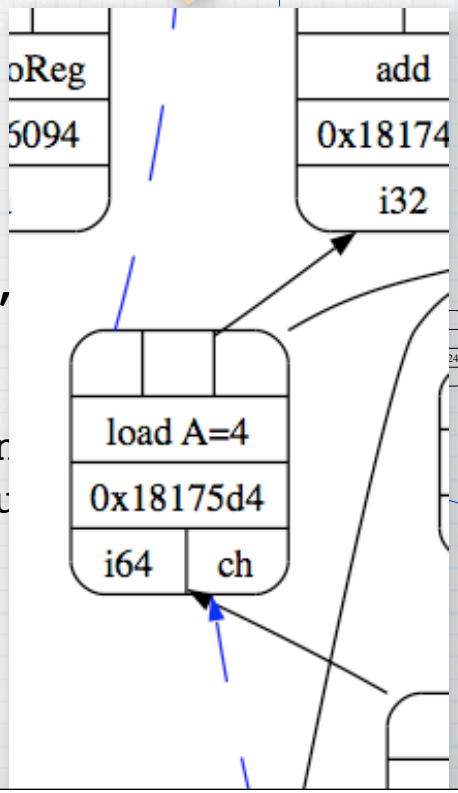
Lower

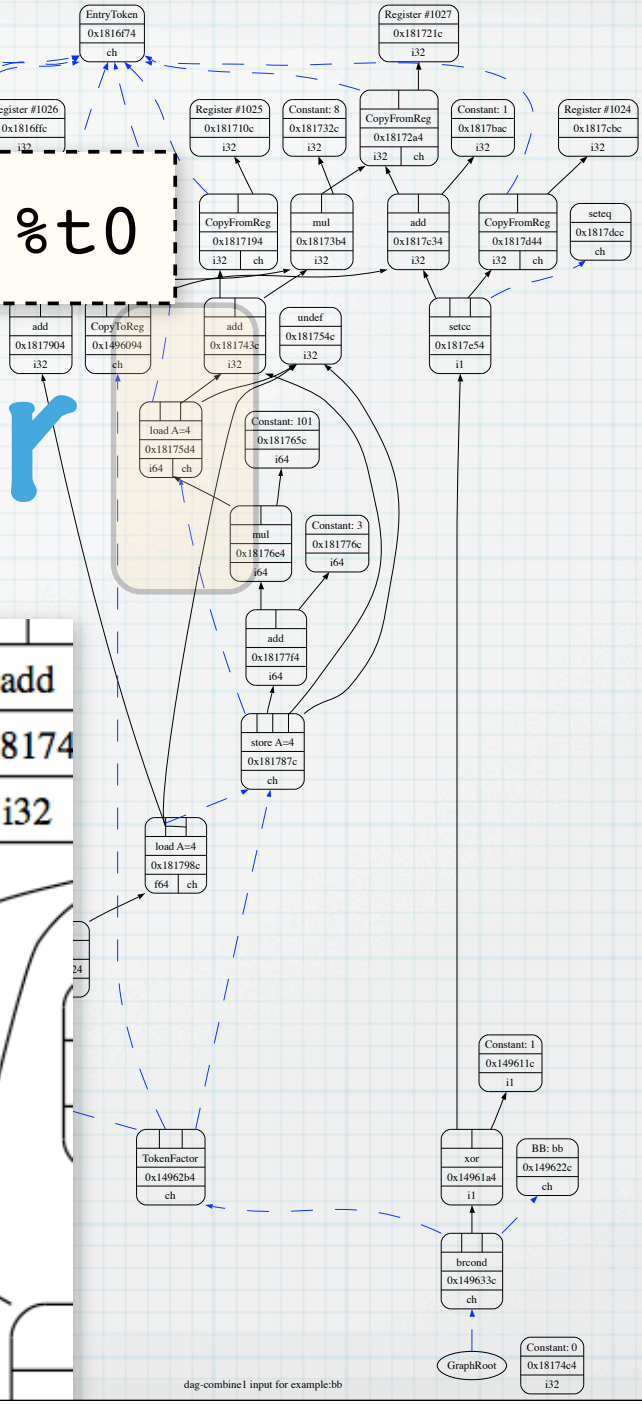


```
bb:
 %i = phi i32 [ 0, %entry
              [ %i.next,
 %sum = phi double [ 0.0, %entry ],
                   [ %sum.next, %bb ]

 %t0 = getelementptr i64* %x,
 %t1 = load i64* %t0
 %t2 = mul i64 %t1, 101
 %t3 = add i64 %t2, 3
 store i64 %t3, i64* %t0

 %t4 = getelementptr double*
 %t5 = load double* %t4
 %sum.next = add double %sum,

 %i.next = add i32 %i, 1
 %exitcond = icmp eq i32 %i.n
 br i1 %exitcond, label %retu
```

load i64* %t0

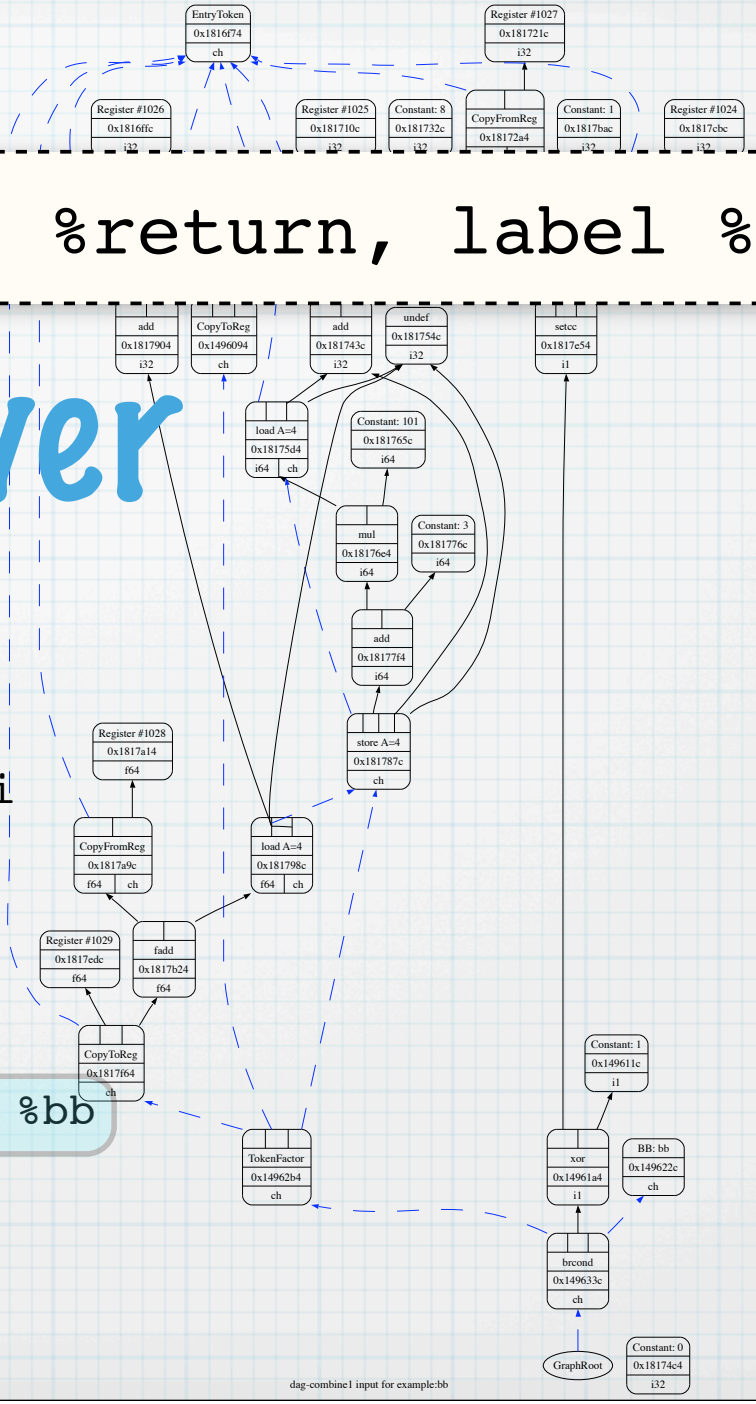
Lower





```
bb:
  %sum = phi double [ 0.0, %entry ],
                    [ %sum.next, %bb ]

  %t0 = getelementptr i64* %x, i32 %i
  %t1 = load i64* %t0
  %t2 = mul i64 %t1, 101
  %t3 = add i64 %t2, 3
  store i64 %t3, i64* %t0

  %t4 = getelementptr double* %y, i32 %i
  %t5 = load double* %t4
  %sum.next = add double %sum, %t5

  %i.next = add i32 %i, 1
  %exitcond = icmp eq i32 %i.next, %n
  br i1 %exitcond, label %return, label %bb
```

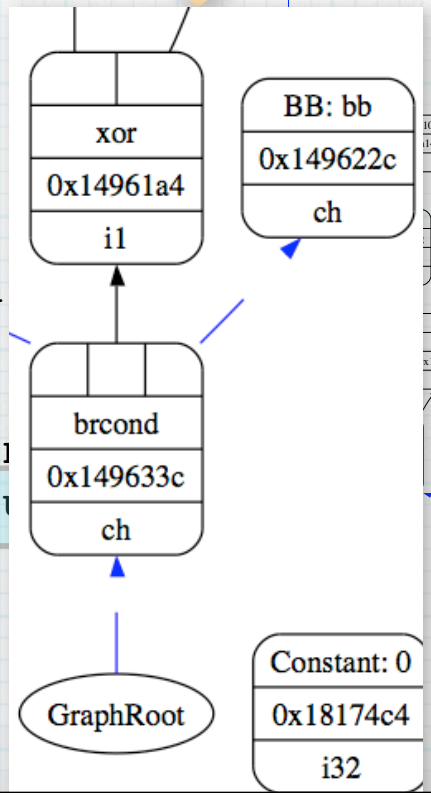
Lower

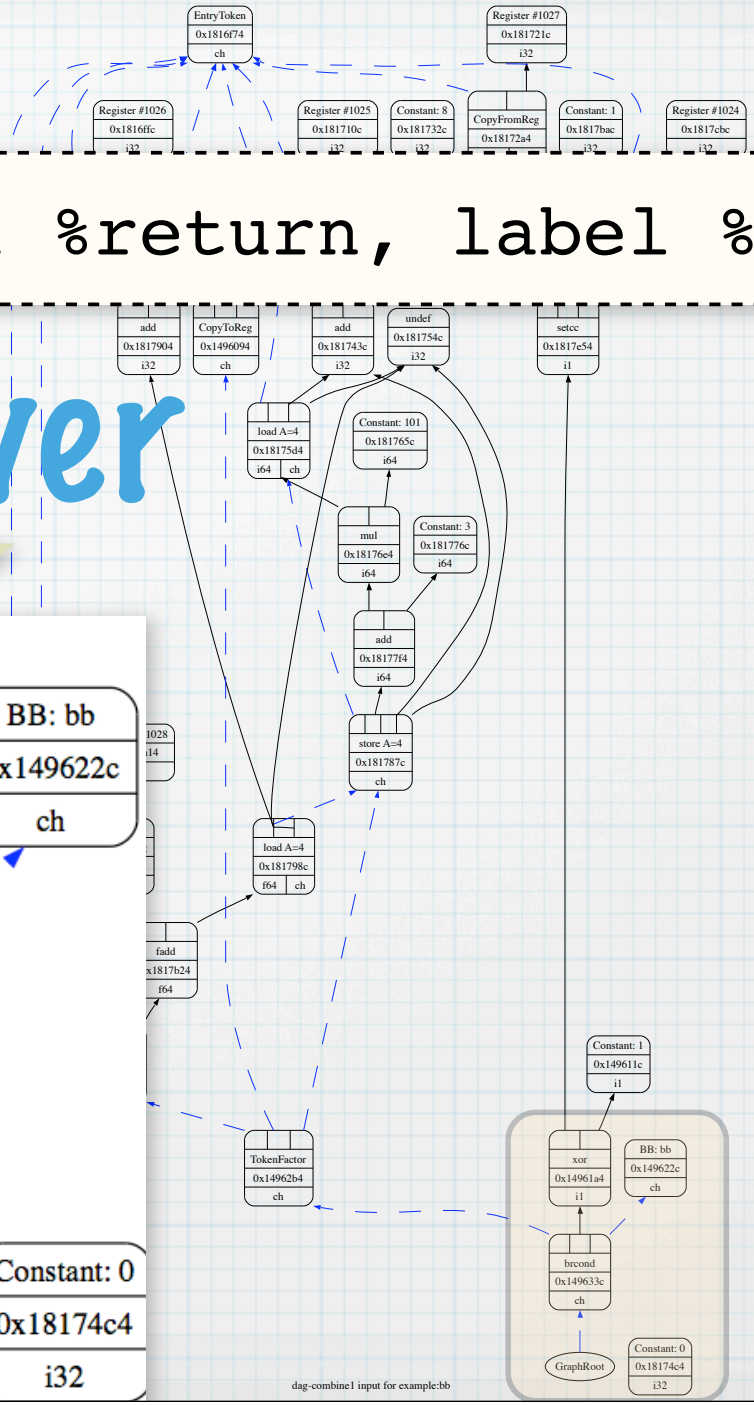


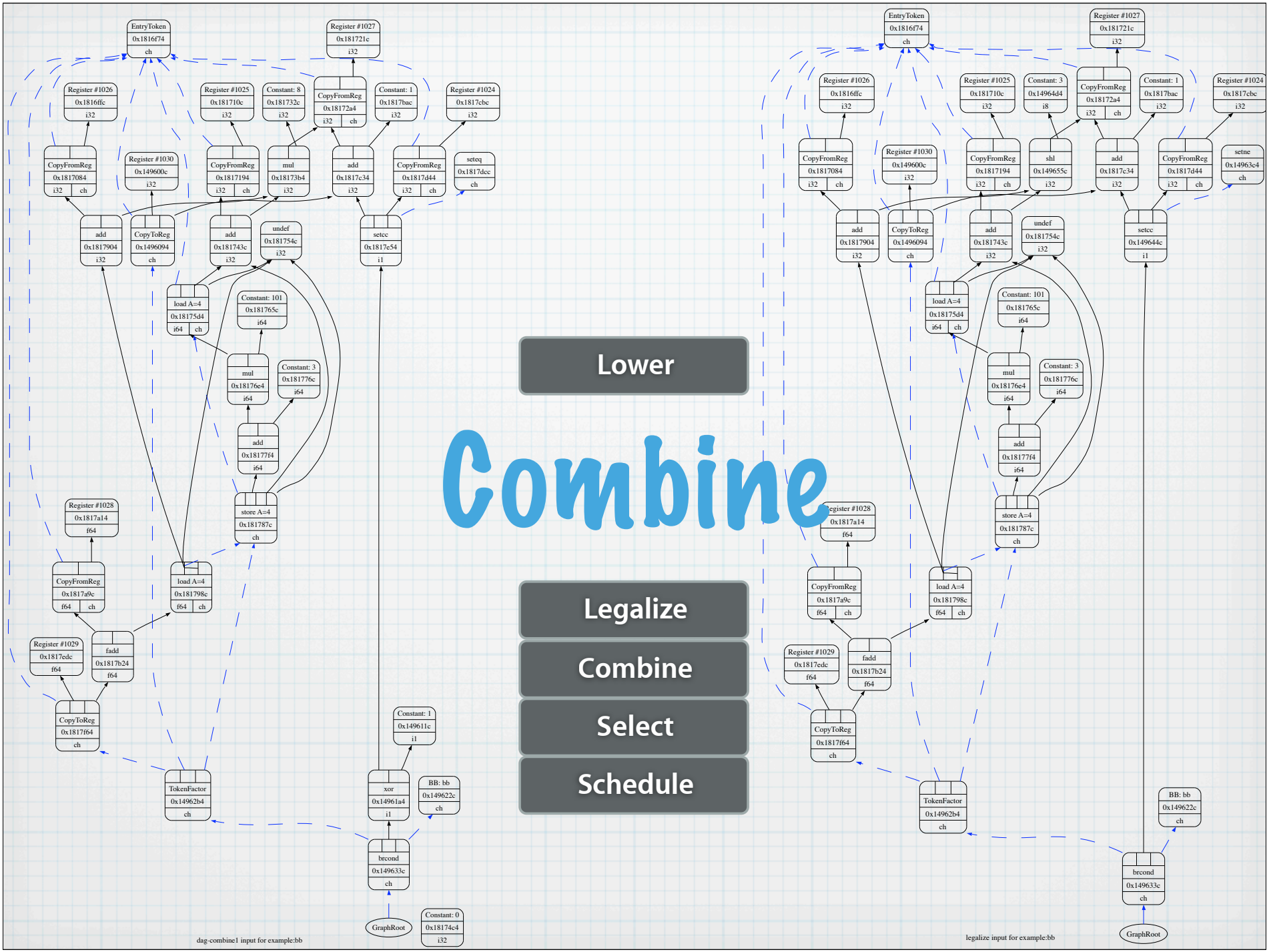
```
bb:
  %sum = phi double [ 0.0, %entry ],
                    [ %sum.next, %bb

  %t0 = getelementptr i64* %x, i3
  %t1 = load i64* %t0
  %t2 = mul i64 %t1, 101
  %t3 = add i64 %t2, 3
  store i64 %t3, i64* %t0

  %t4 = getelementptr double*
  %t5 = load double* %t4
  %sum.next = add double %sum

  %i.next = add i32 %i, 1
  %exitcond = icmp eq i32 %i.
  br i1 %exitcond, label %retu
```

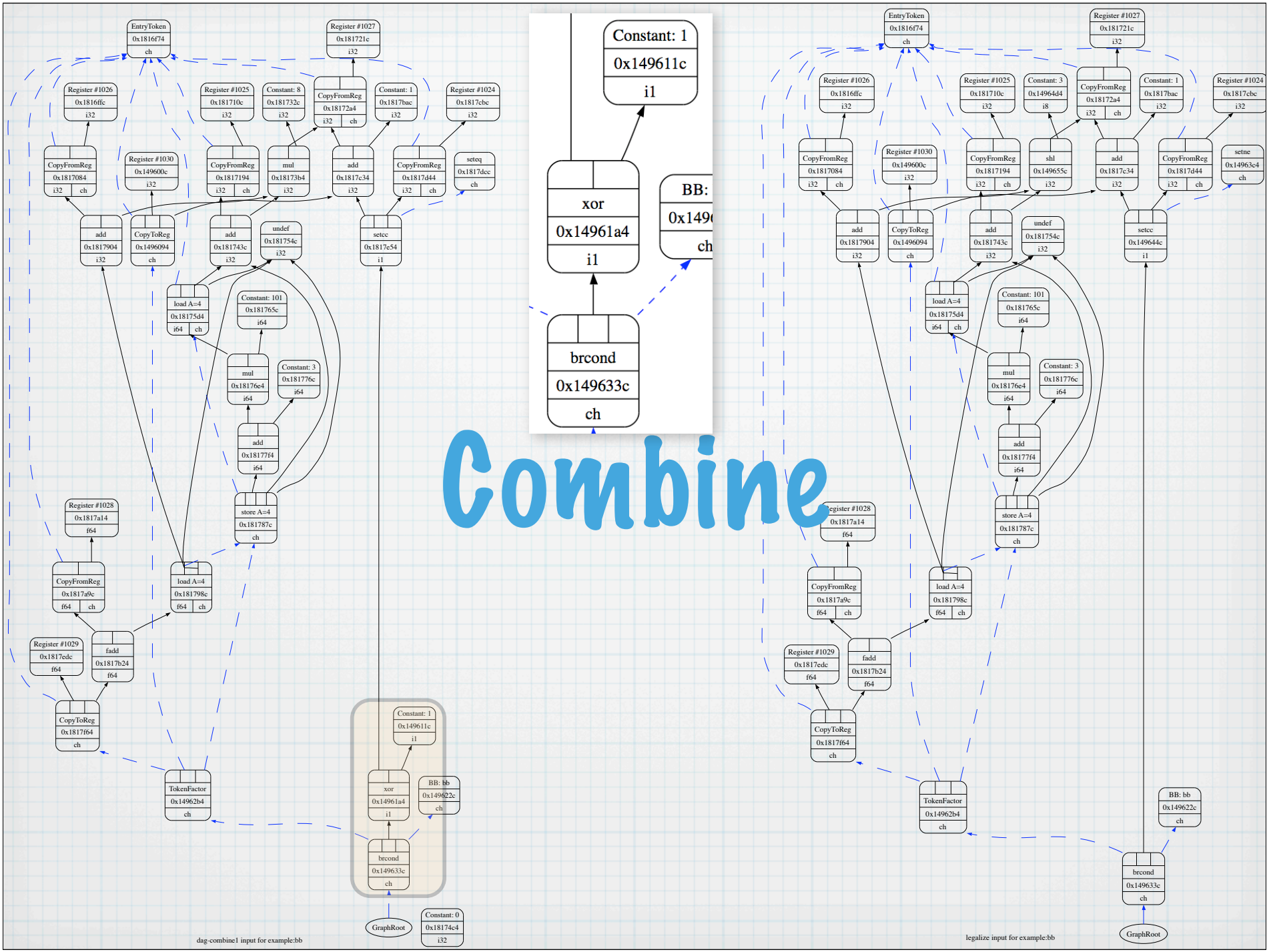
```
br i1 %exitcond, label %return, label %bb
```

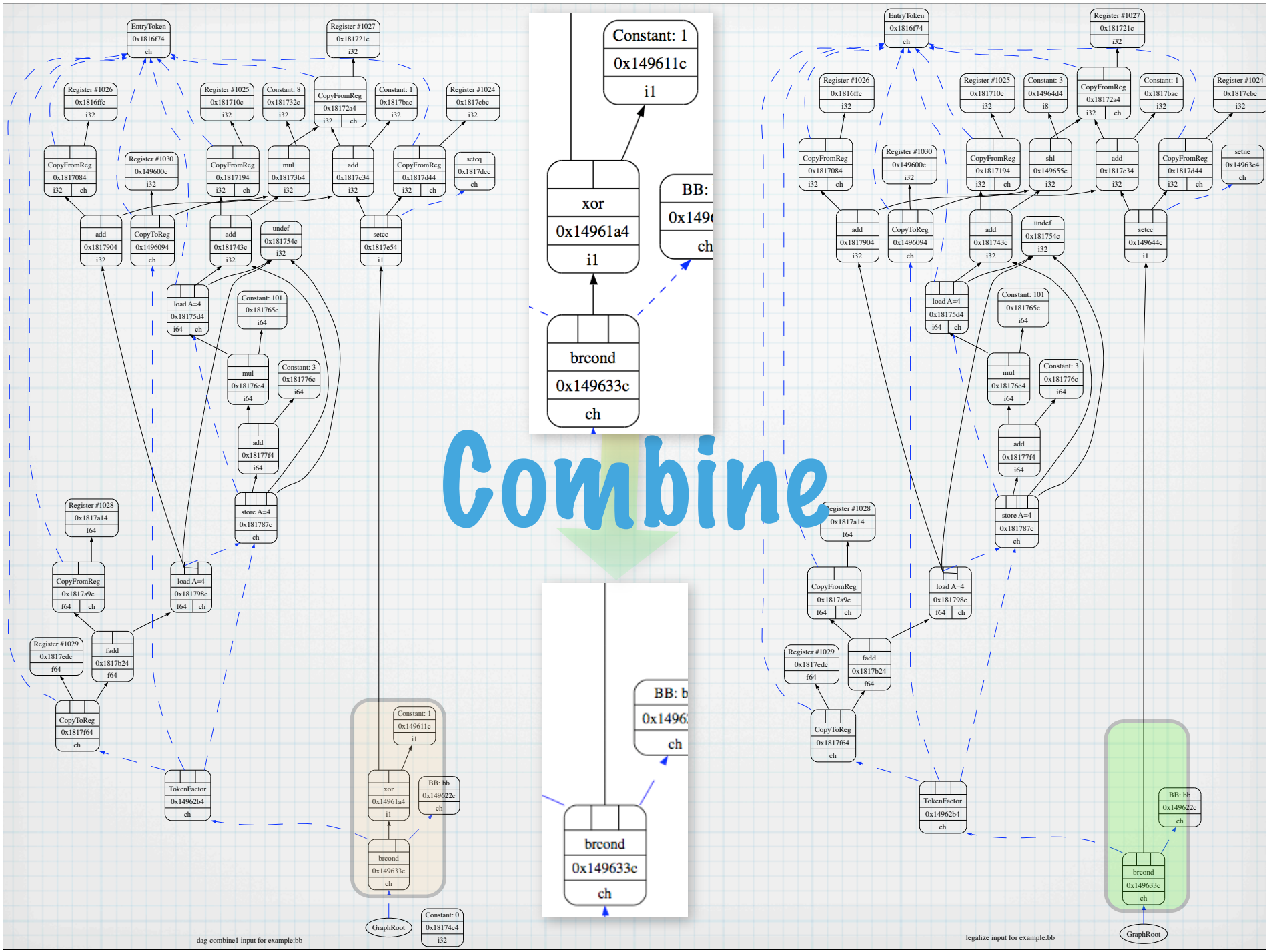
Lower

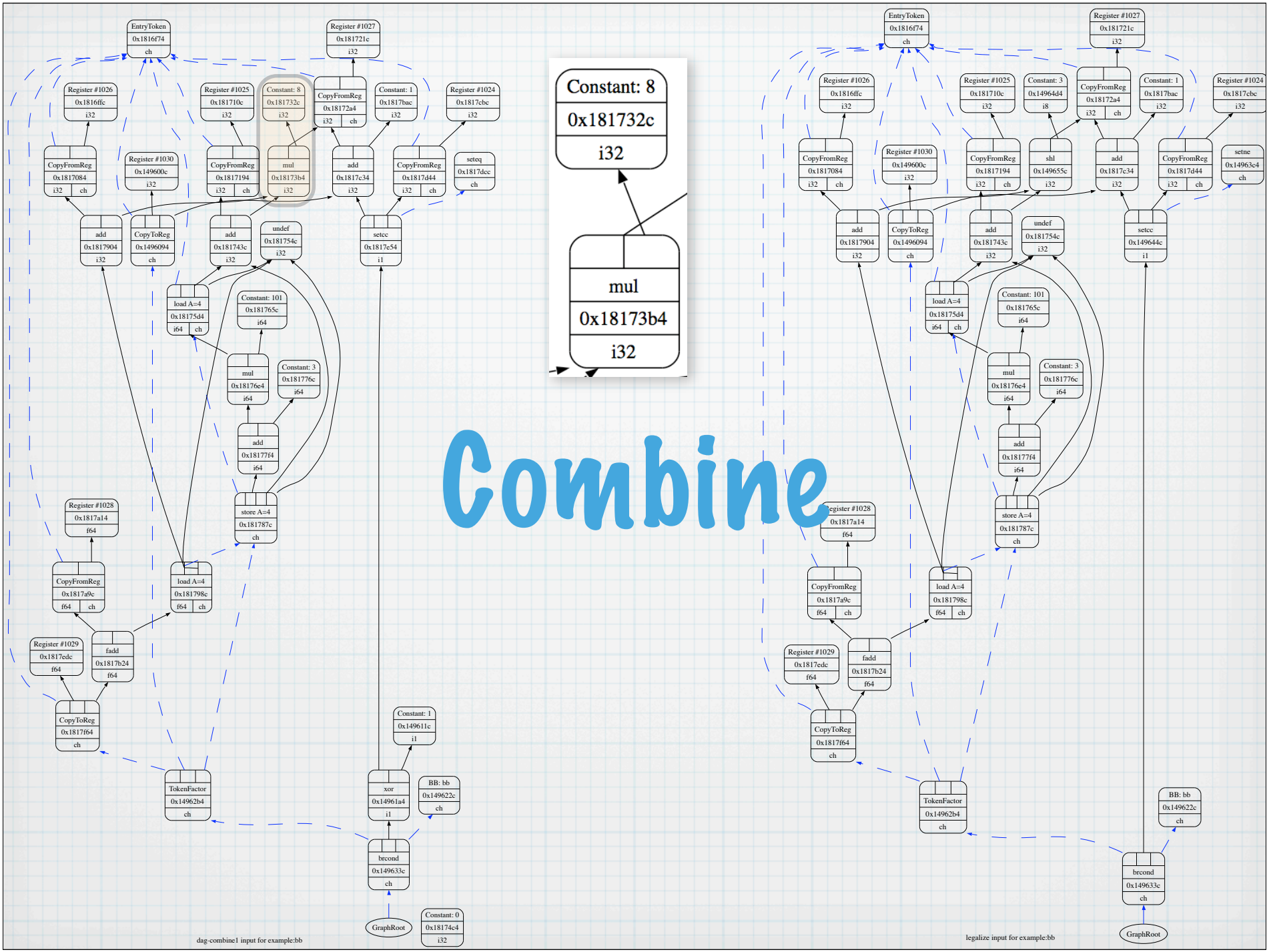


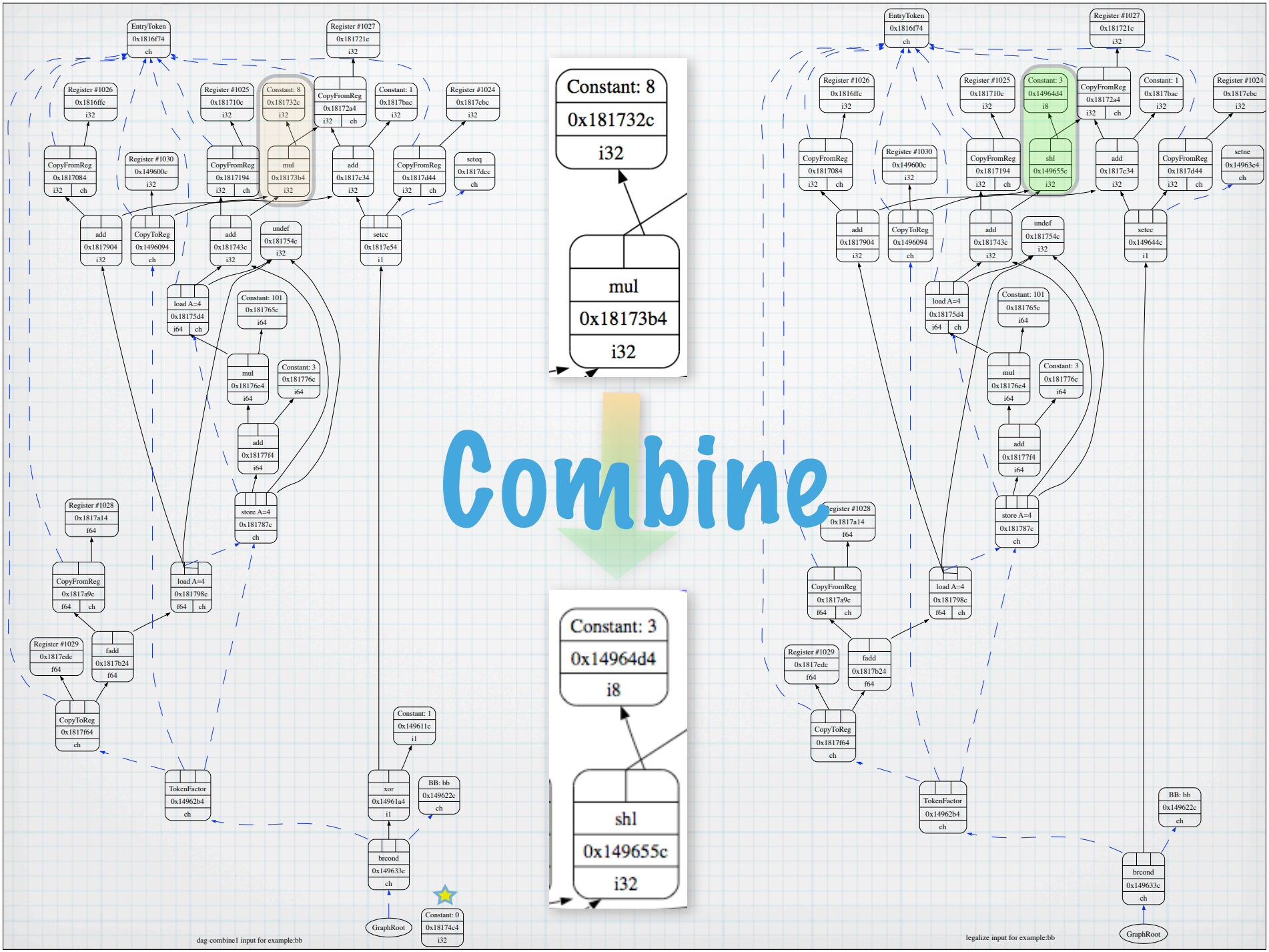


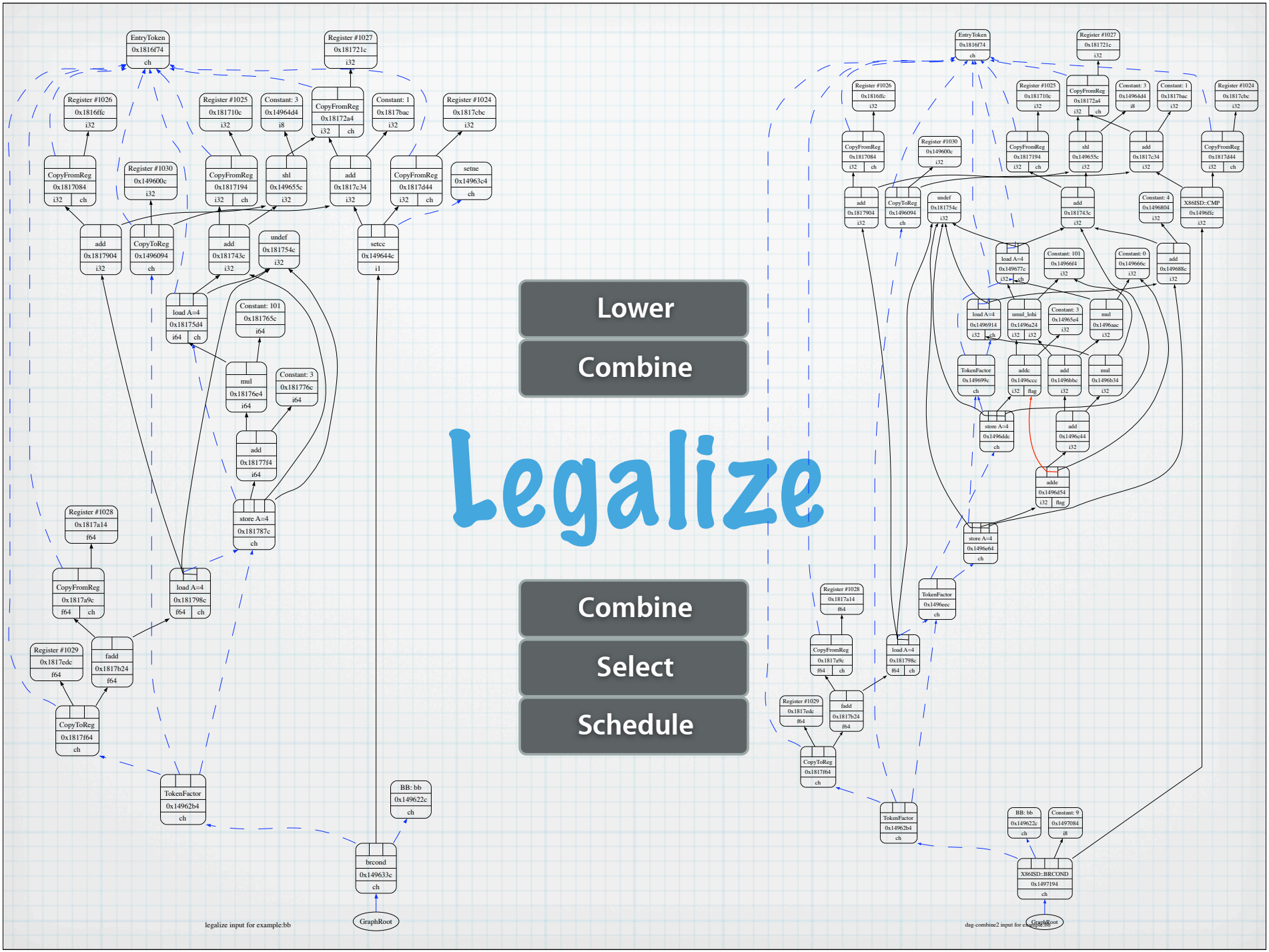

Lower
Combine
Legalize
Combine
Select
Schedule
dag-combine1 input for example:bb
legalize input for example:bb

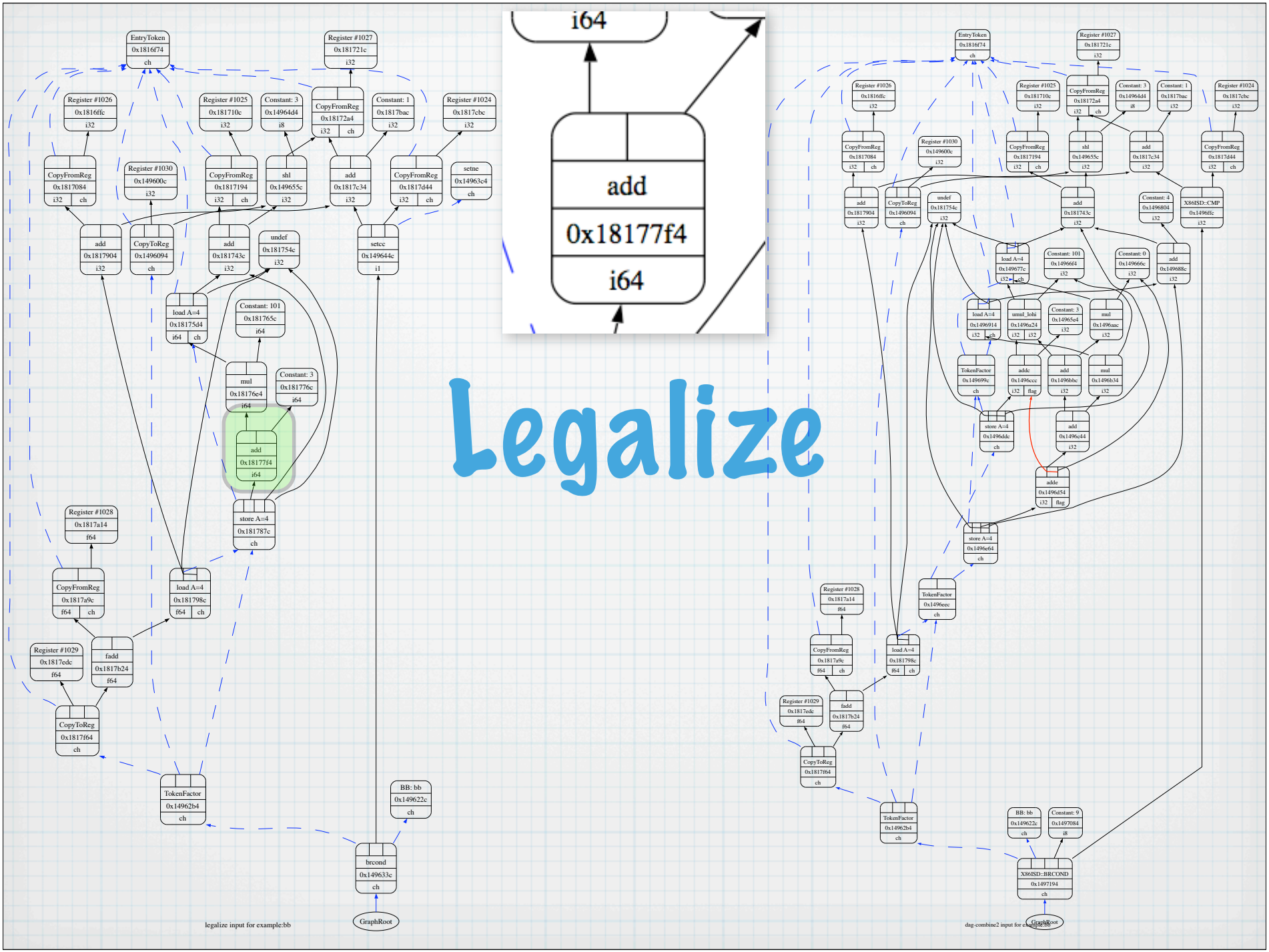

Combine
Constant: 1
0x149611c
i1
xor
0x14961a4
i1
BB:
0x149
ch
brcond
0x149633c
ch
dag-combine1 input for example:bb
legalize input for example:bb

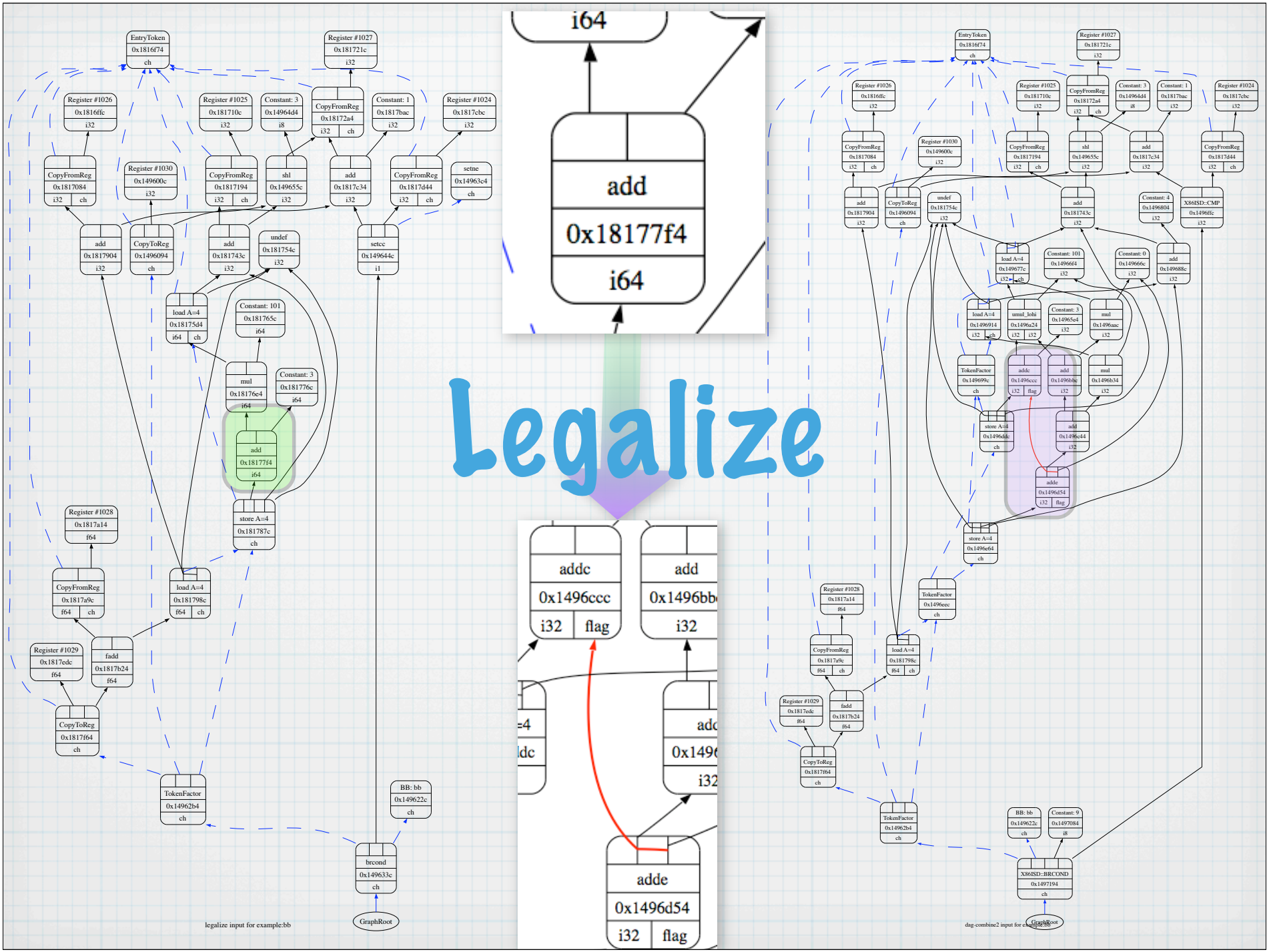

Combine
dag-combine1 input for example:bb
legalize input for example:bb
EntryToken
0x1816f74
ch
Register #1027
0x181721c
i32
Register #1026
0x1816ffc
i32
Register #1025
0x181710c
i32
Constant: 8
0x181732c
i32
CopyFromReg
0x18172a4
i32 ch
Constant: 1
0x1817bac
i32
Register #1024
0x1817cbc
i32
CopyFromReg
0x1817084
i32 ch
Register #1030
0x149600c
i32
CopyFromReg
0x1817194
i32 ch
mul
0x18173b4
i32
add
0x1817c34
i32
CopyFromReg
0x1817d44
i32 ch
seteq
0x1817dcc
ch
add
0x1817904
i32
CopyToReg
0x1496094
ch
add
0x181743c
i32
undef
0x181754c
i32
setcc
0x1817e54
i1
load A=4
0x18175d4
i64 ch
Constant: 101
0x181765c
i64
mul
0x18176e4
i64
Constant: 3
0x181776c
i64
add
0x18177f4
i64
store A=4
0x181787c
ch
Register #1028
0x1817a14
f64
CopyFromReg
0x1817a9c
f64 ch
load A=4
0x181798c
f64 ch
Register #1029
0x1817edc
f64
fadd
0x1817b24
f64
CopyToReg
0x1817f64
ch
TokenFactor
0x14962b4
ch
Constant: 1
0x149611c
i1
xor
0x14961a4
i1
BB: bb
0x149622c
ch
brcond
0x149633c
ch
GraphRoot
Constant: 0
0x18174c4
i32
BB:
0x1496
ch
setne
0x14963c4
ch
shl
0x149655c
i32
Constant: 3
0x14964d4
i8
setcc
0x149644c
i1

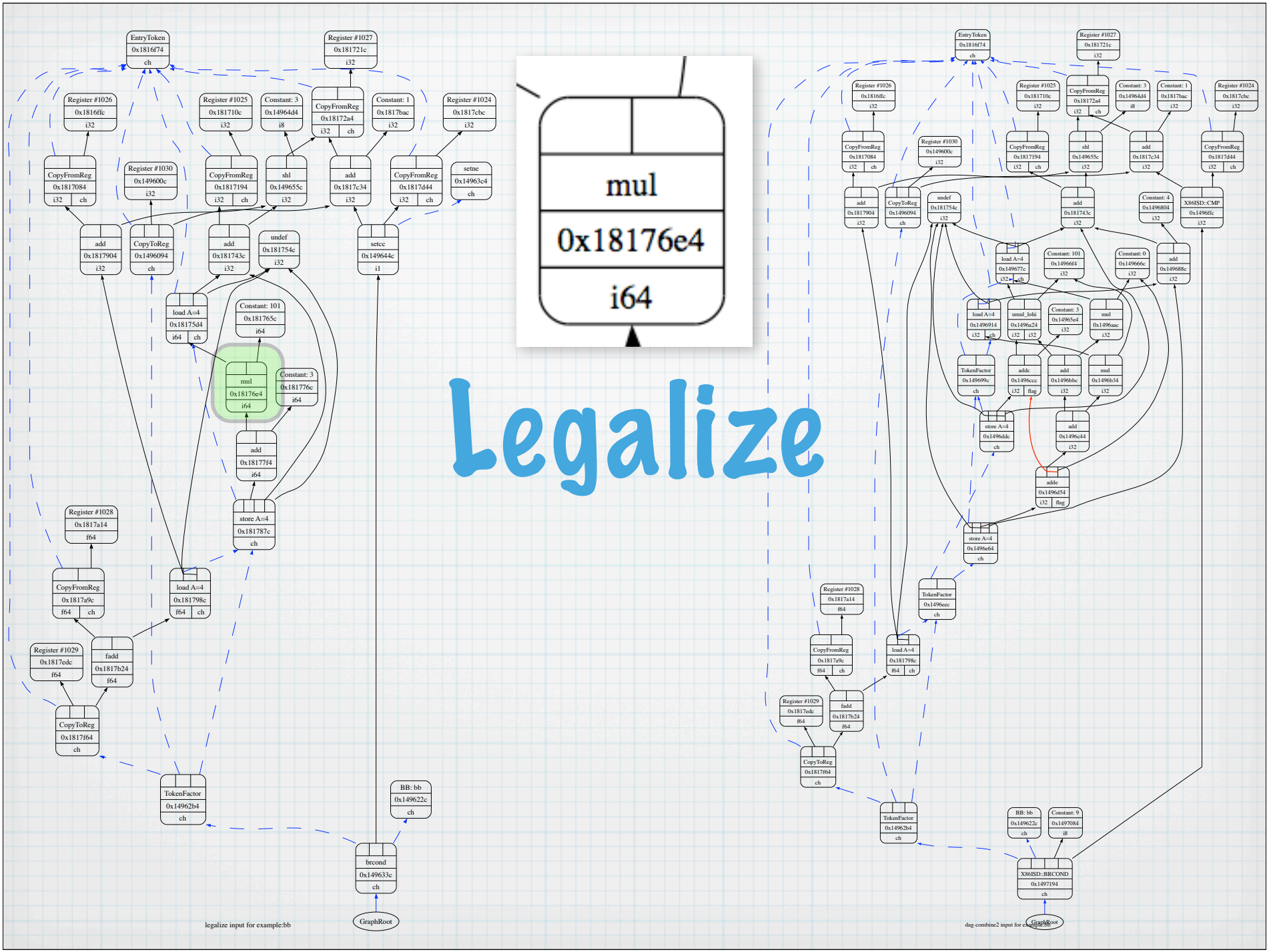

Constant: 8
0x181732c
i32
mul
0x18173b4
i32
Combine
dag-combine1 input for example:bb
legalize input for example:bb

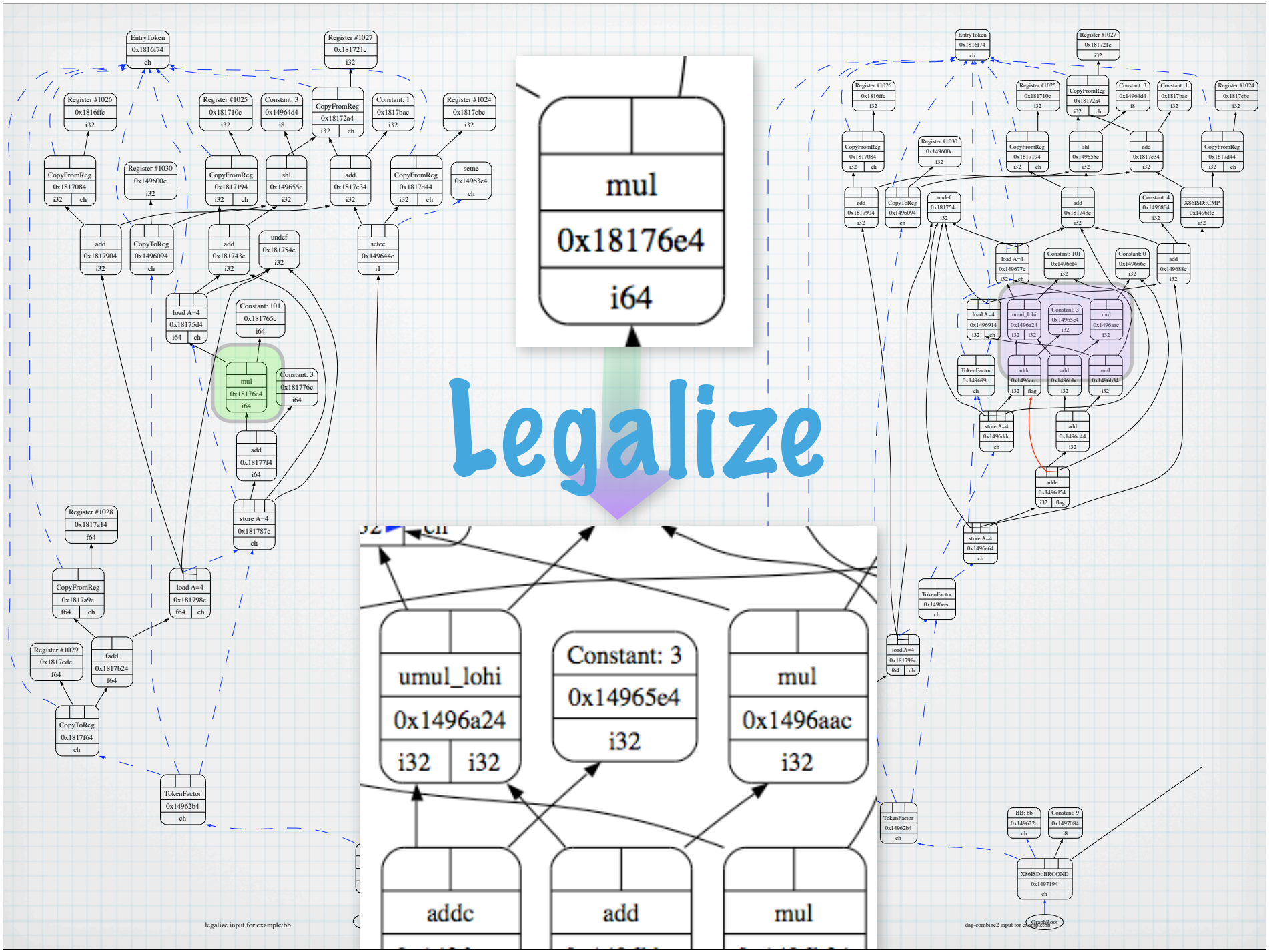

Constant: 8
0x181732c
i32
mul
0x18173b4
i32
Constant: 3
0x14964d4
i8
shl
0x149655c
i32
Combine
dag-combine1 input for example:bb
legalize input for example:bb

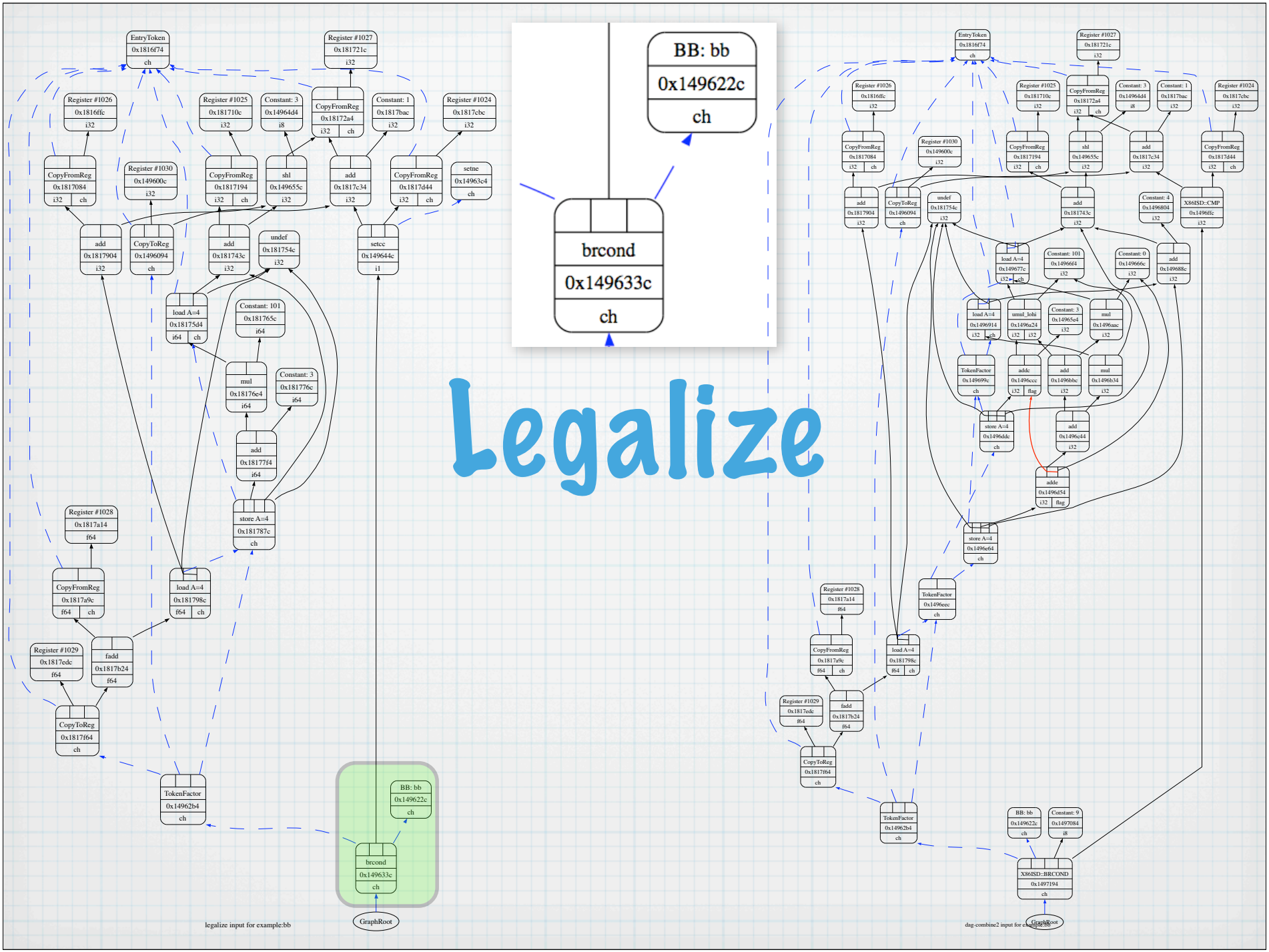

Lower
Combine
Legalize
Combine
Select
Schedule
legalize input for example:bb
dag-combine2 input for example:bb

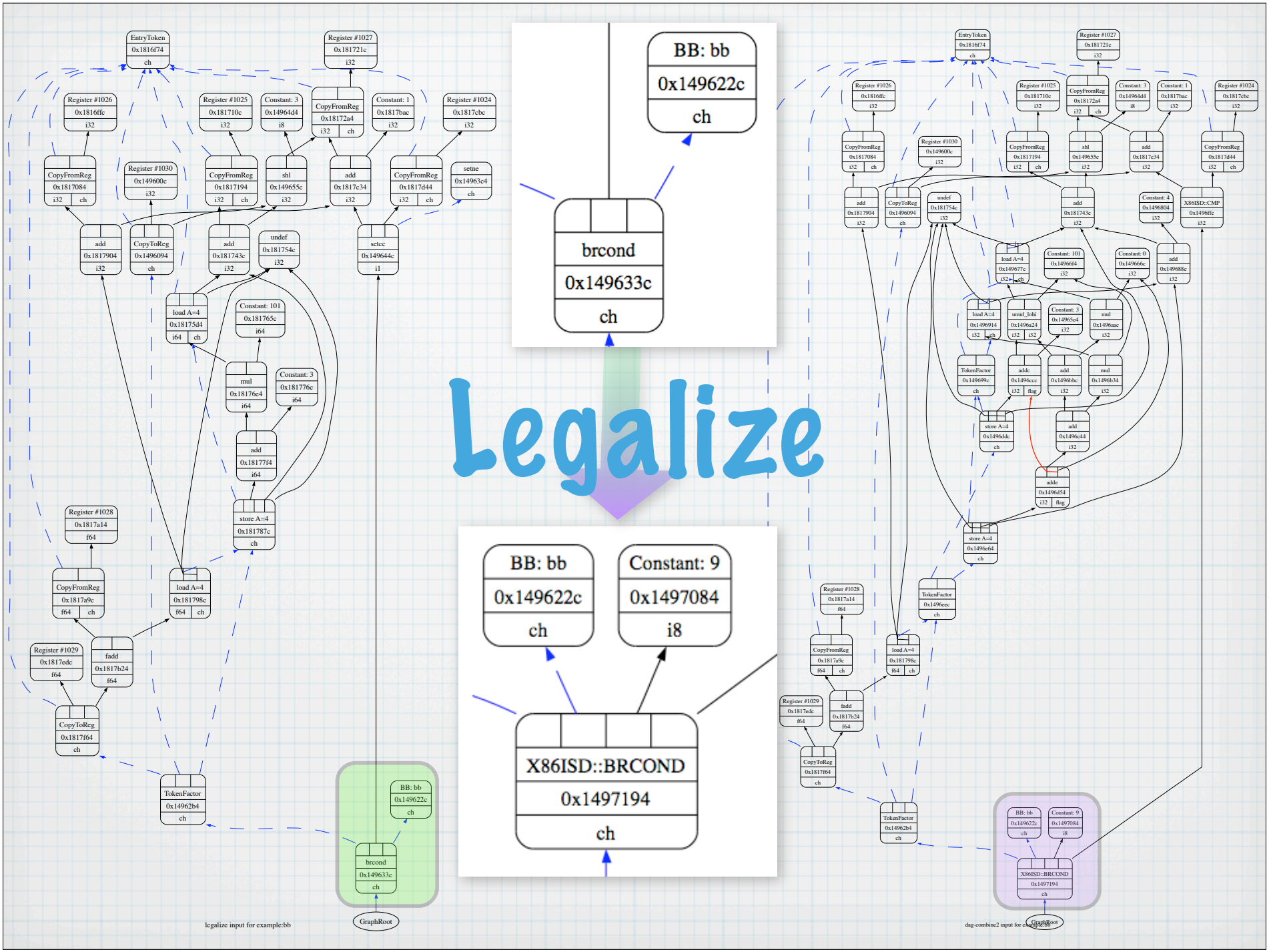

i64
add
0x18177f4
i64
Legalize
legalize input for example:bb
dag-combine2 input for example:bb

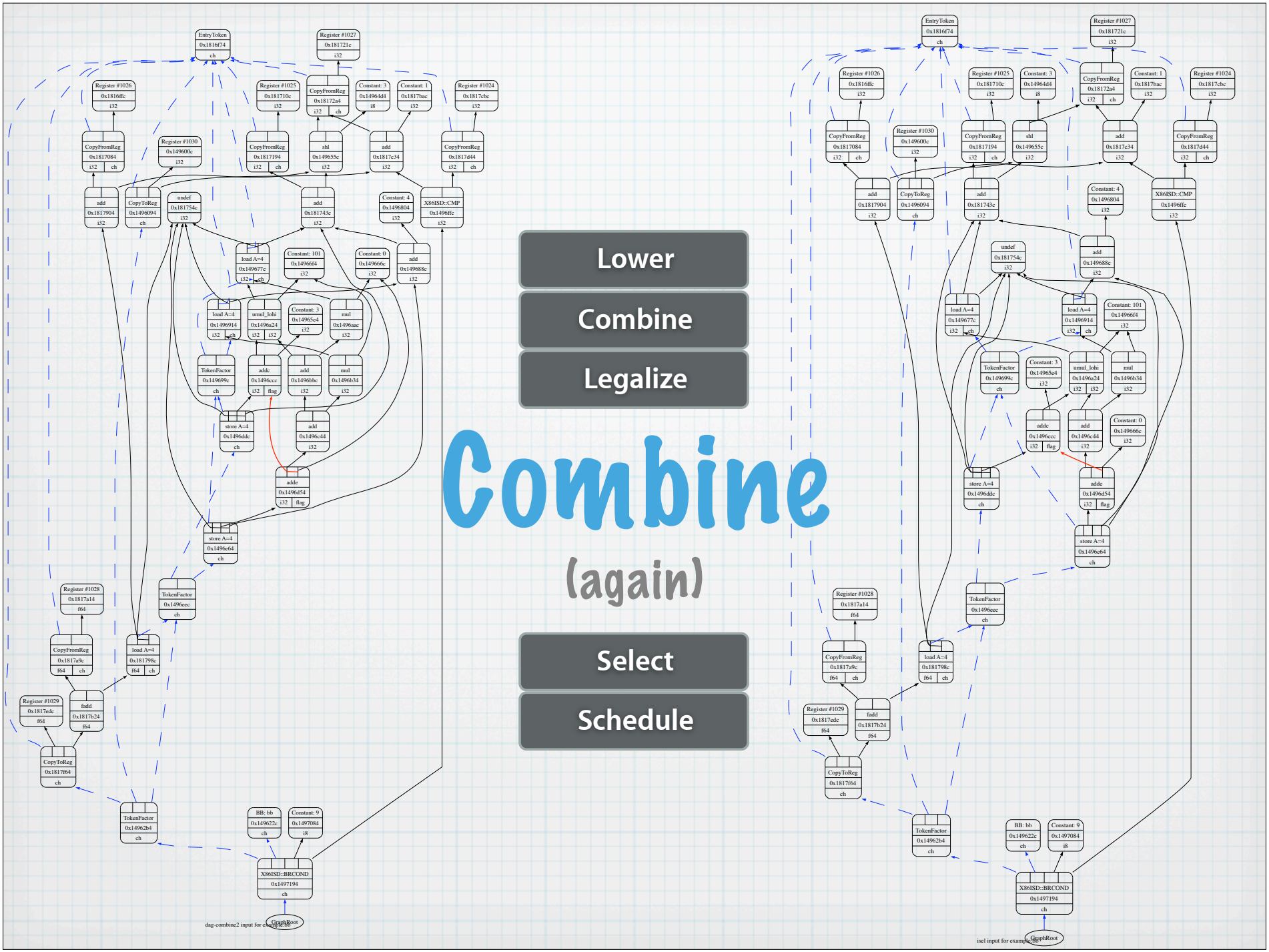

Legalize
legalize input for example:bb
dag-combine2 input for example:bb
i64
add
0x18177f4
i64
addc
0x1496ccc
i32
flag
add
0x1496bb
i32
adde
0x1496d54
i32
flag

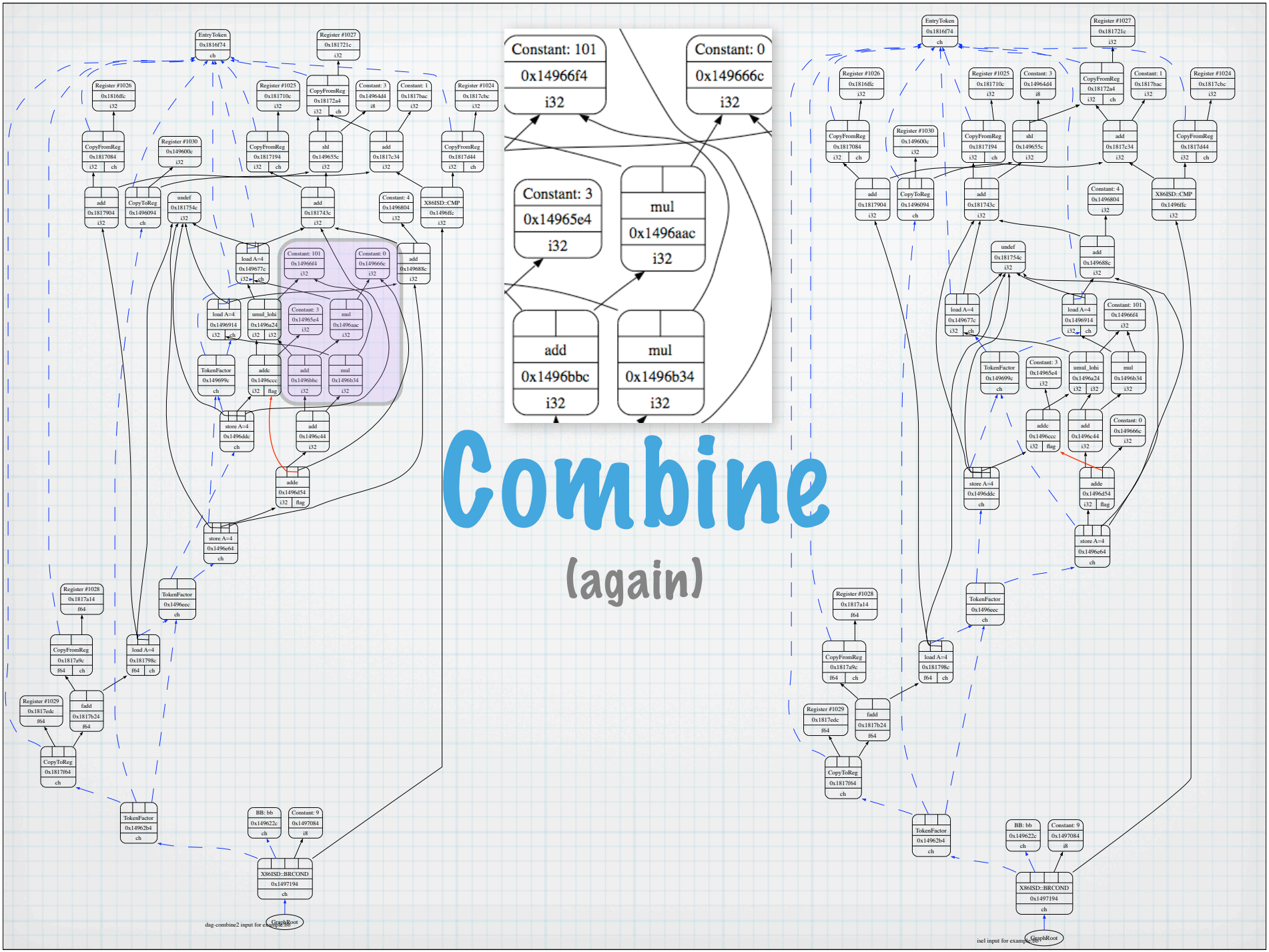

mul
0x18176e4
i64
Legalize
legalize input for example:bb
dag-combine2 input for example:bb

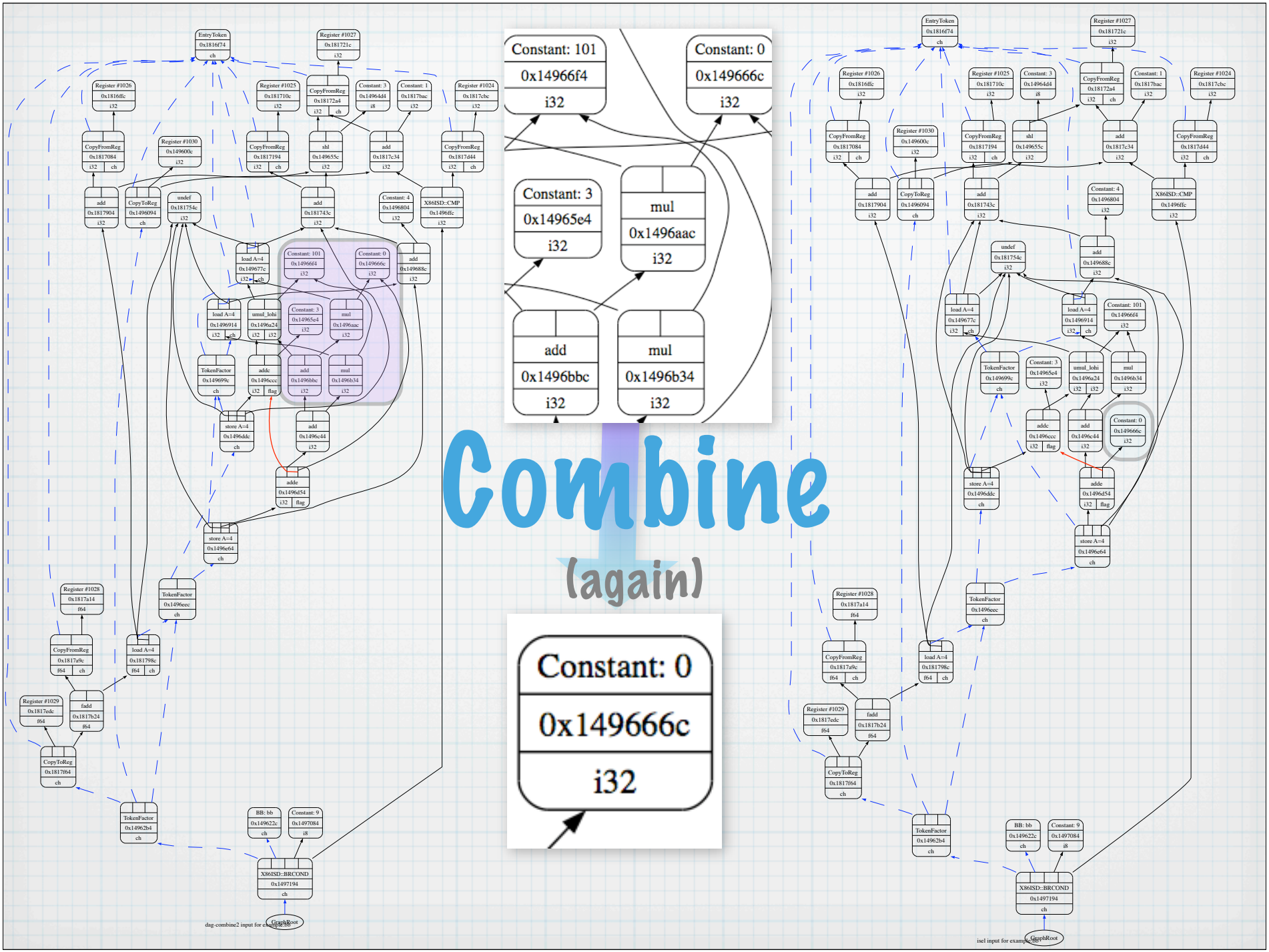

mul
0x18176e4
i64
Legalize
umul_lohi
0x1496a24
i32 i32
Constant: 3
0x14965e4
i32
mul
0x1496aac
i32
addc
add
mul
legalize input for example:bb
dag-combine2 input for example:bb

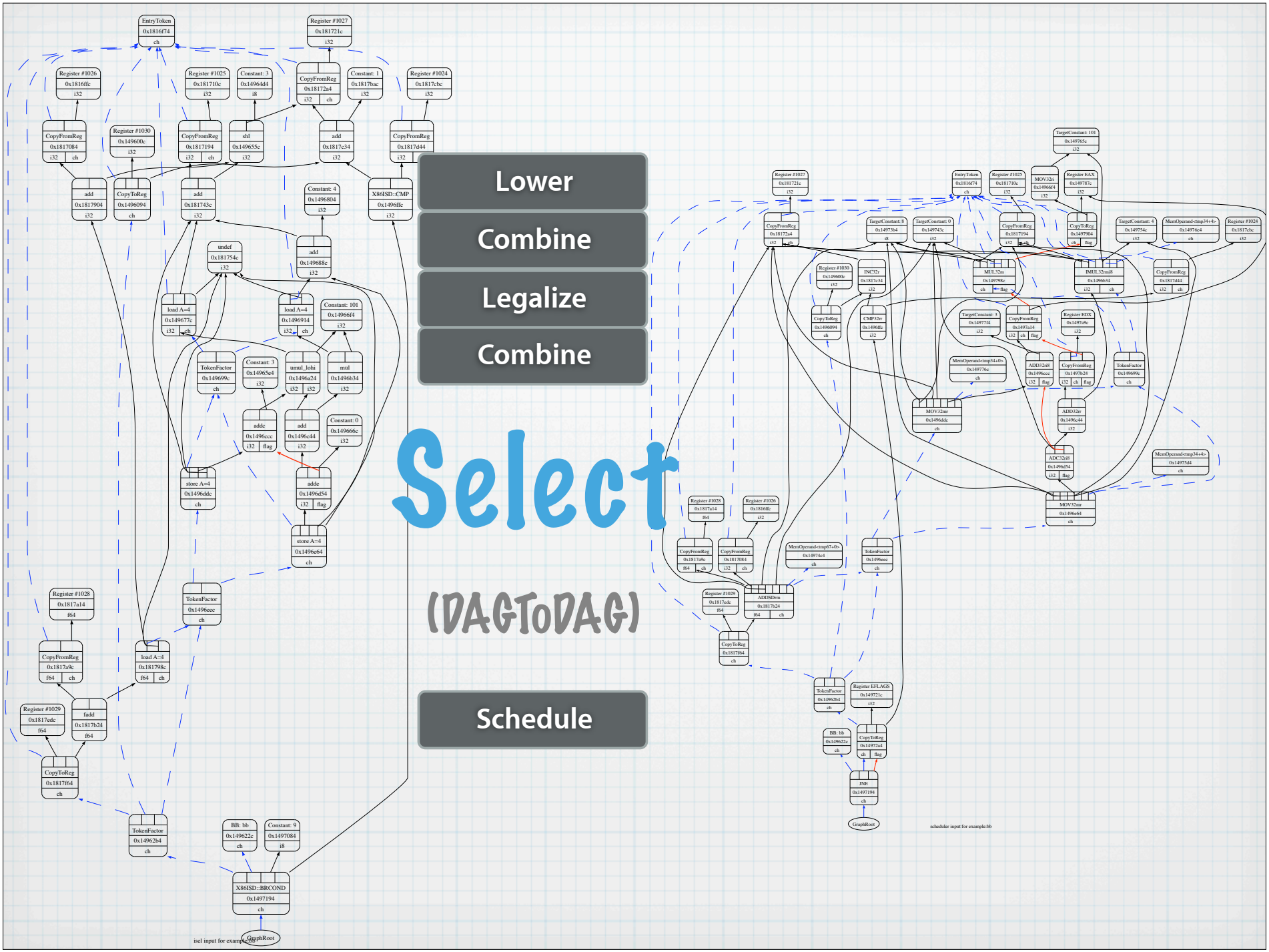

BB: bb
0x149622c
ch
brcond
0x149633c
ch
Legalize
legalize input for example:bb
dag-combine2 input for example:bb

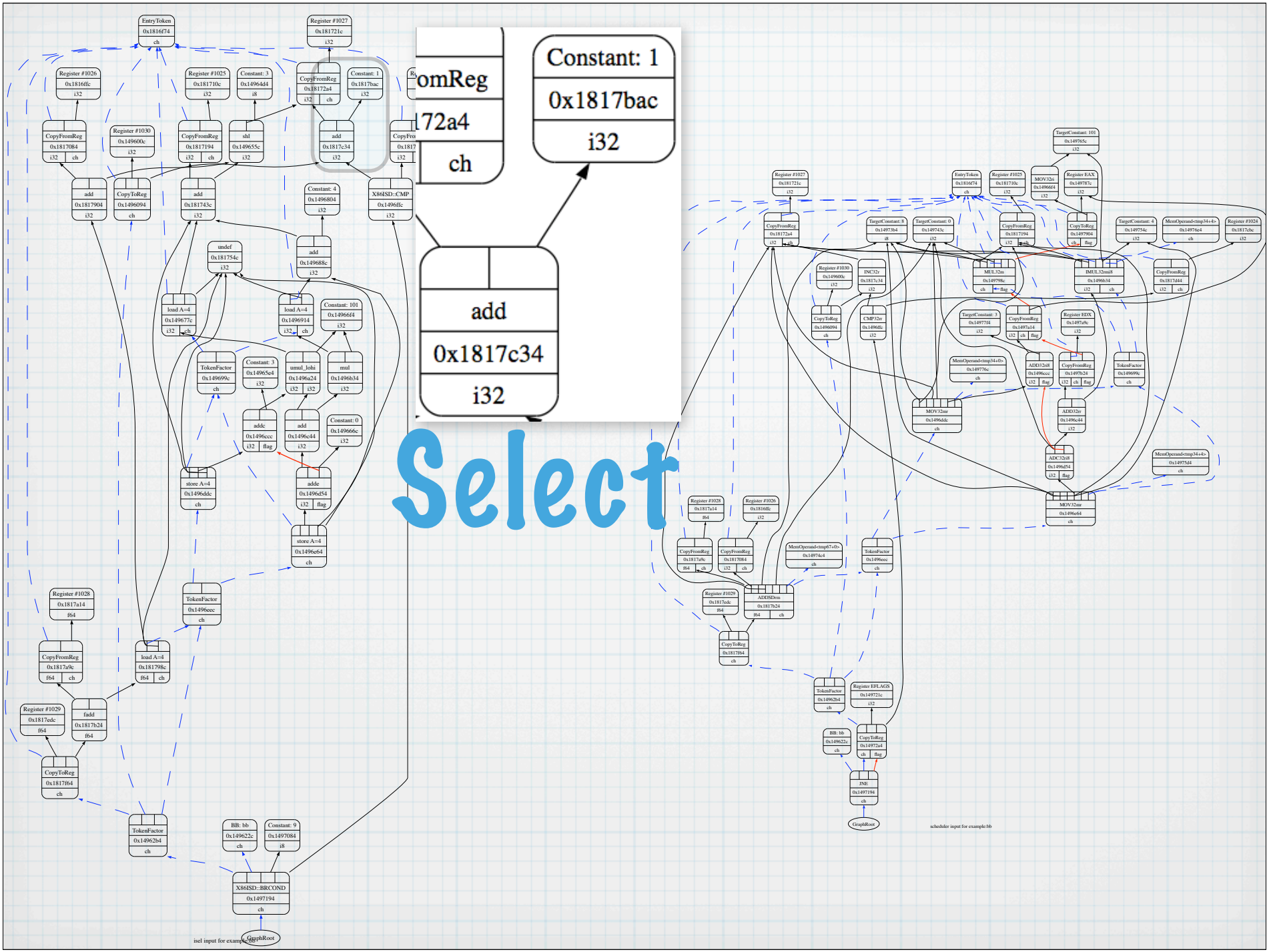

BB: bb
0x149622c
ch
brcond
0x149633c
ch
Legalize
BB: bb
0x149622c
ch
Constant: 9
0x1497084
i8
X86ISD::BRCOND
0x1497194
ch
legalize input for example:bb
dag-combine2 input for example:bb

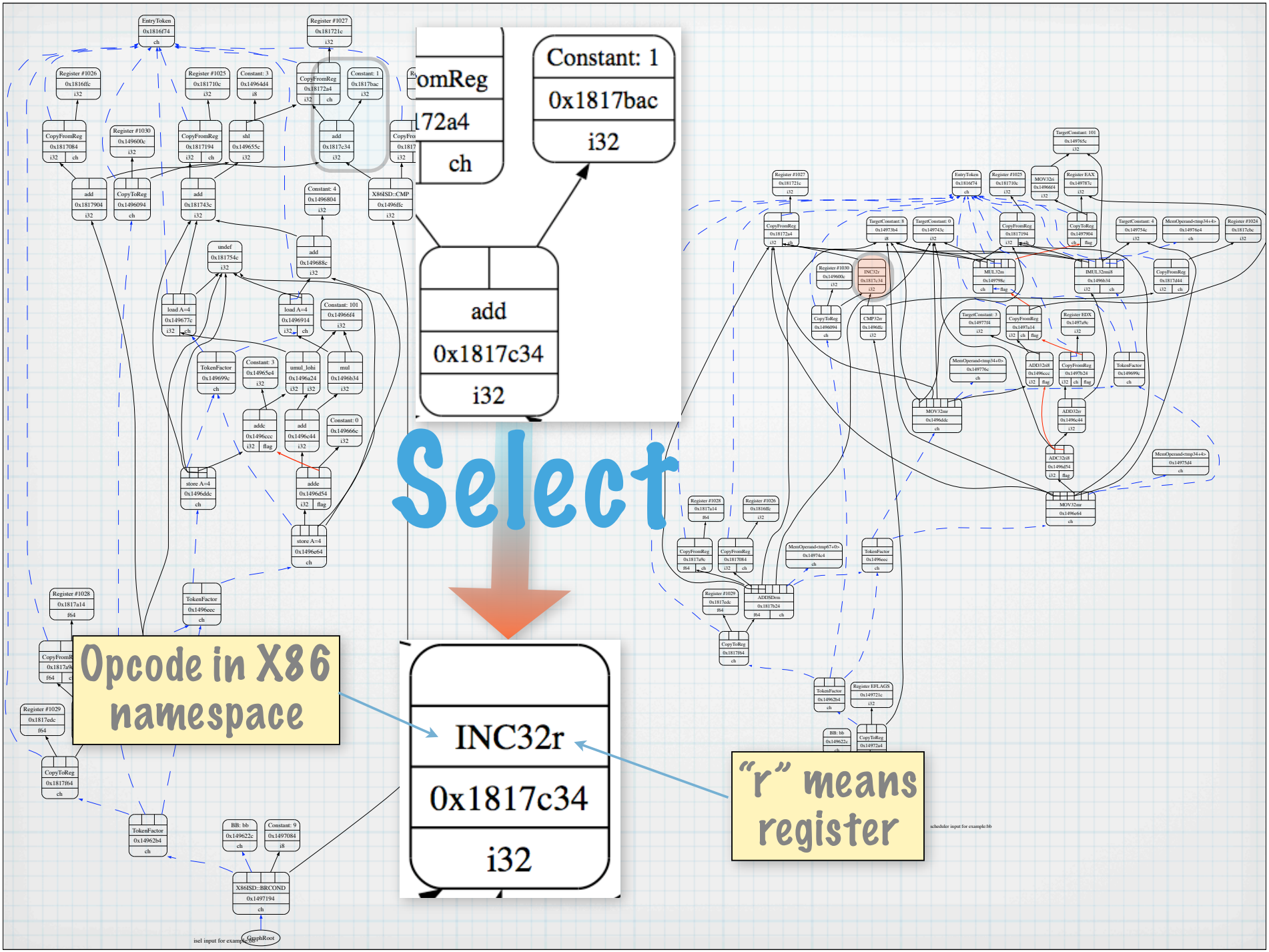

Lower
Combine
Legalize
Combine
(again)
Select
Schedule

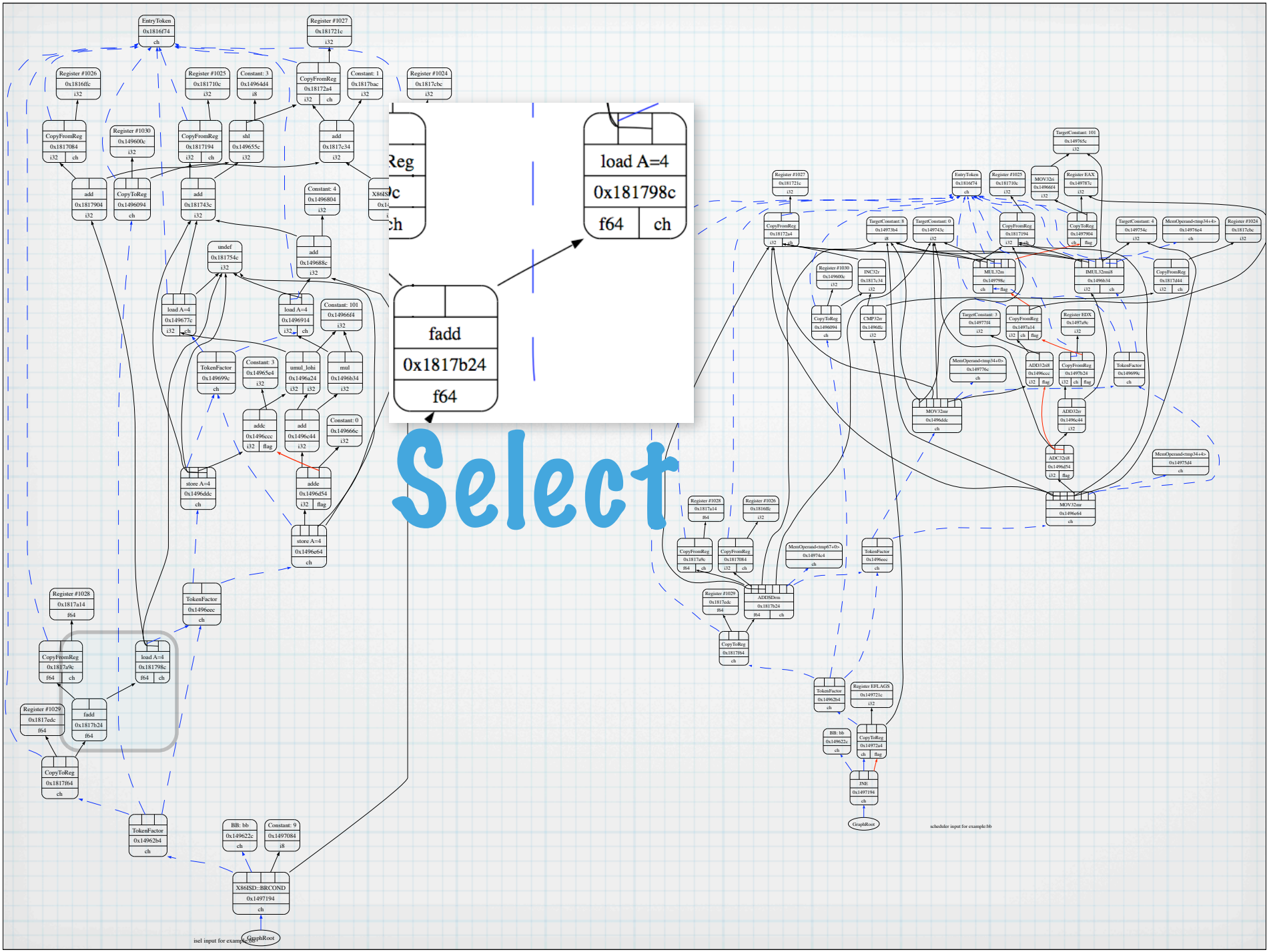

Constant: 101
0x14966f4
i32
Constant: 0
0x149666c
i32
Constant: 3
0x14965e4
i32
mul
0x1496aac
i32
add
0x1496bbc
i32
mul
0x1496b34
i32
Combine
(again)

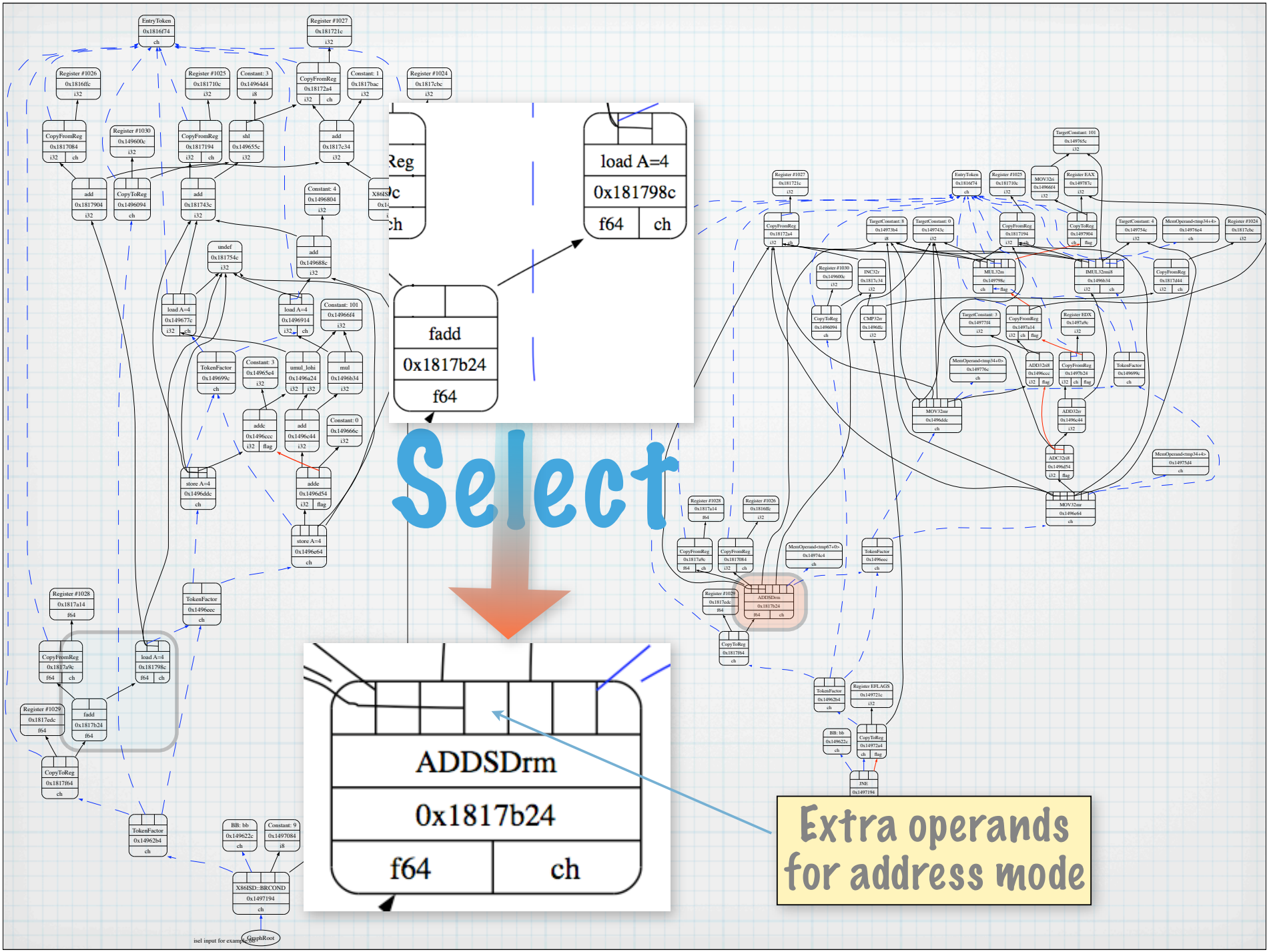

Constant: 101
0x14966f4
i32
Constant: 0
0x149666c
i32
Constant: 3
0x14965e4
i32
mul
0x1496aac
i32
add
0x1496bbc
i32
mul
0x1496b34
i32
Combine
(again)
Constant: 0
0x149666c
i32


Lower
Combine
Legalize
Combine
Select
(DAGtoDAG)
Schedule
isel input for example:bb
scheduler input for example:bb


Constant: 1
0x1817bac
i32
add
0x1817c34
i32
Select
isel input for example:bb
scheduler input for example:bb


Constant: 1
0x1817bac
i32
add
0x1817c34
i32
Select
Opcode in X86 namespace
INC32r
0x1817c34
i32
"r" means register


load A=4
0x181798c
f64
ch
fadd
0x1817b24
f64
Select


load A=4
0x181798c
f64
ch
fadd
0x1817b24
f64
Select
ADDSDrm
0x1817b24
f64
ch
Extra operands
for address mode

Instruction Patterns

```
def ADDI :
  DForm_2<14,
          (outs GPRC:$rD),
          (ins GPRC:$rA, s16imm:$imm),
          "addi $rD, $rA, $imm",
          IntGeneral,
          [(set GPRC:$rD,
                (add GPRC:$rA,
                     immSExt16:$imm))]>;
```

Instruction Patterns

```
def ADDI :
  DForm_2<14,
          (outs GPRC:$rD),
          (ins GPRC:$rA, s16imm:$imm),
          "addi $rD, $rA, $imm",
          IntGeneral,
          [(set GPRC:$rD,
                (add GPRC:$rA,
                     immSExt16:$imm))]>;
```

Instruction Patterns

```
def ADDI :
  DForm_2<14,
          (outs GPRC:$rD),
          (ins GPRC:$rA, s16imm:$imm),
          "addi $rD, $rA, $imm",
          IntGeneral,
          [(set GPRC:$rD,
                (add GPRC:$rA,
                     immSExt16:$imm))]>;
```

Instruction Patterns

```
def ADDI :
  DForm_2<14,
          (outs GPRC:$rD),
          (ins GPRC:$rA, s16imm:$imm),
          "addi $rD, $rA, $imm",
          IntGeneral,
          [(set GPRC:$rD,
                (add GPRC:$rA,
                     immSExt16:$imm))]>;
```

Instruction Patterns

```
def ADDI :
  DForm_2<14,
          (outs GPRC:$rD),
          (ins GPRC:$rA, s16imm:$imm),
          "addi $rD, $rA, $imm",
          IntGeneral,
          [(set GPRC:$rD,
                (add GPRC:$rA,
                     immSExt16:$imm))]>;
```

Instruction Patterns

```
def ADDI :
  DForm_2<14,
          (outs GPRC:$rD),
          (ins GPRC:$rA, s16imm:$imm),
          "addi $rD, $rA, $imm",
          IntGeneral,
          [(set GPRC:$rD,
                (add GPRC:$rA,
                     immSExt16:$imm))]>;
```

Instruction Patterns

```
def ADDI :
  DForm_2<14,
          (outs GPRC:$rD),
          (ins GPRC:$rA, s16imm:$imm),
          "addi $rD, $rA, $imm",
          IntGeneral,
          [(set GPRC:$rD,
                (add GPRC:$rA,
                     immSExt16:$imm))]>;
```

Instruction Patterns

```
def ADDI :
  DForm_2<14,
          (outs GPRC:$rD),
          (ins GPRC:$rA, s16imm:$imm),
          "addi $rD, $rA, $imm",
          IntGeneral,
          [(set GPRC:$rD,
                (add GPRC:$rA,
                     immSExt16:$imm))]>;
```

Lower

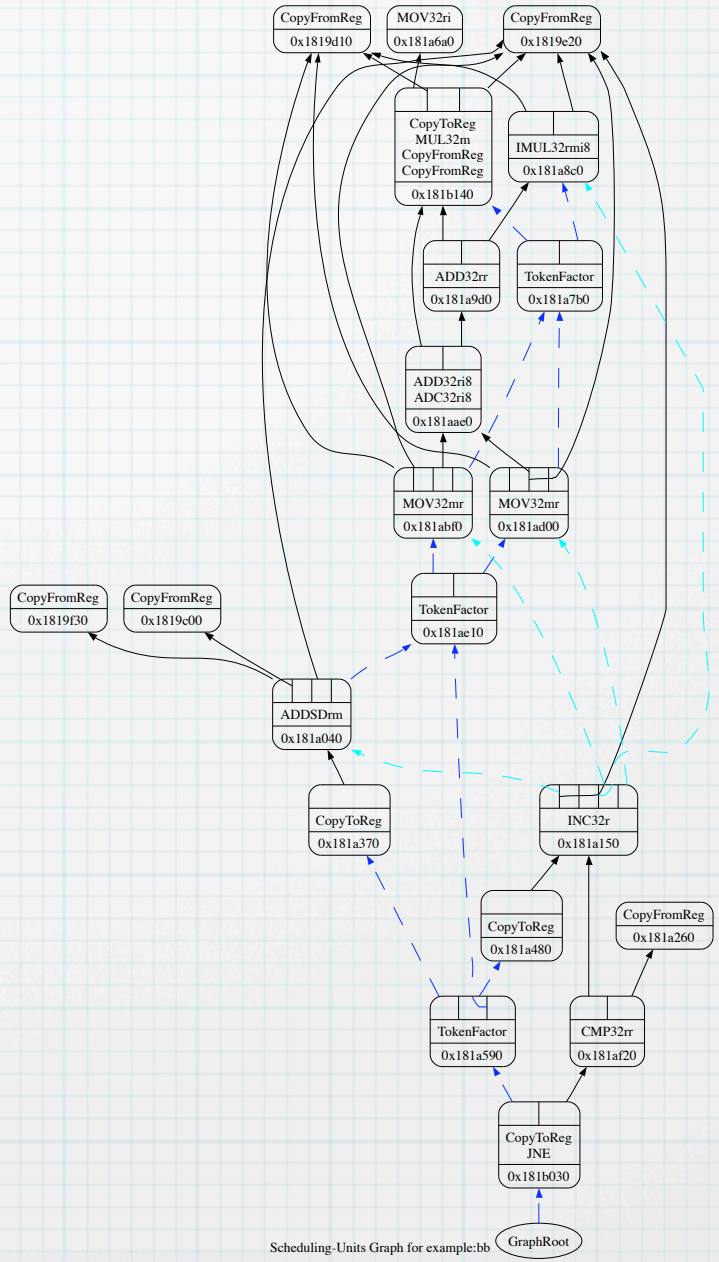
Combine

Legalize

Combine

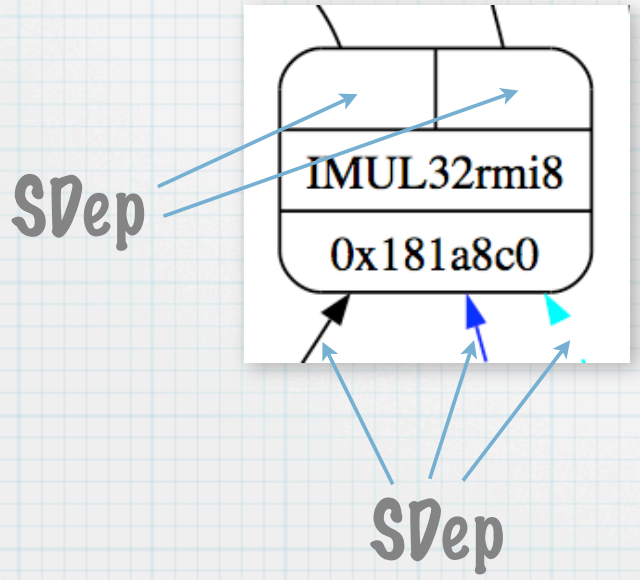
Select

Schedule

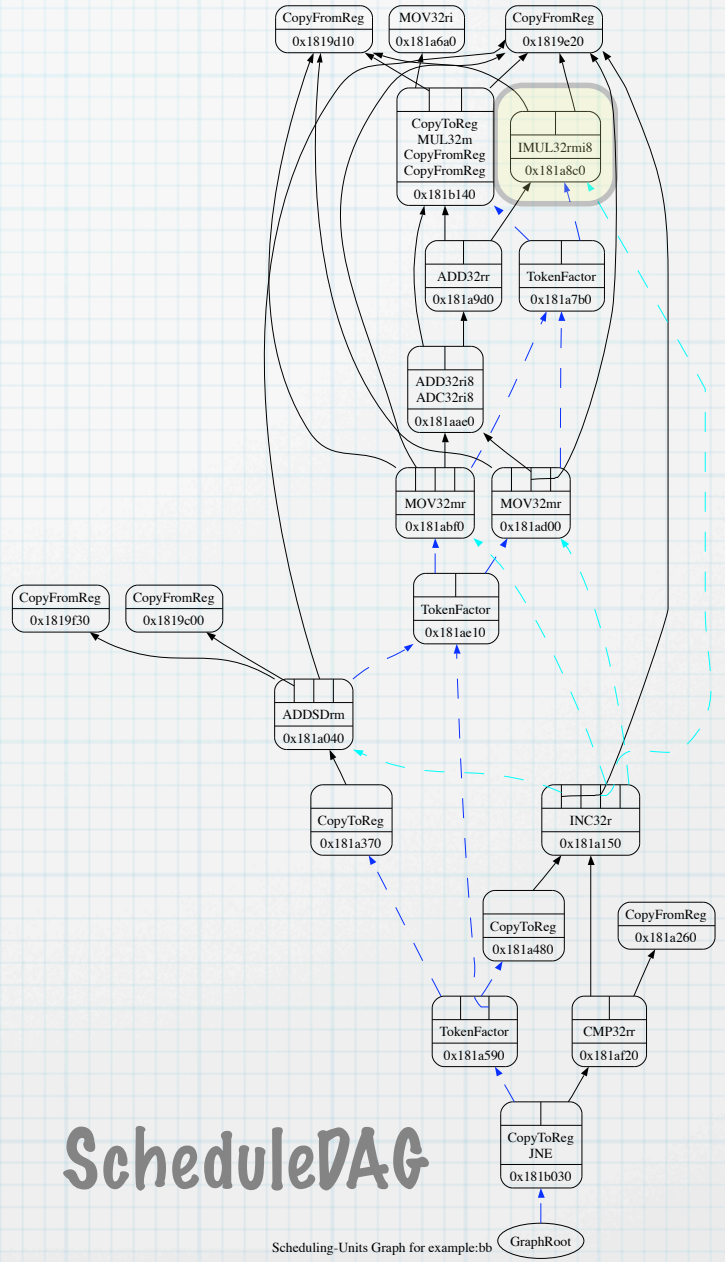


Schedule

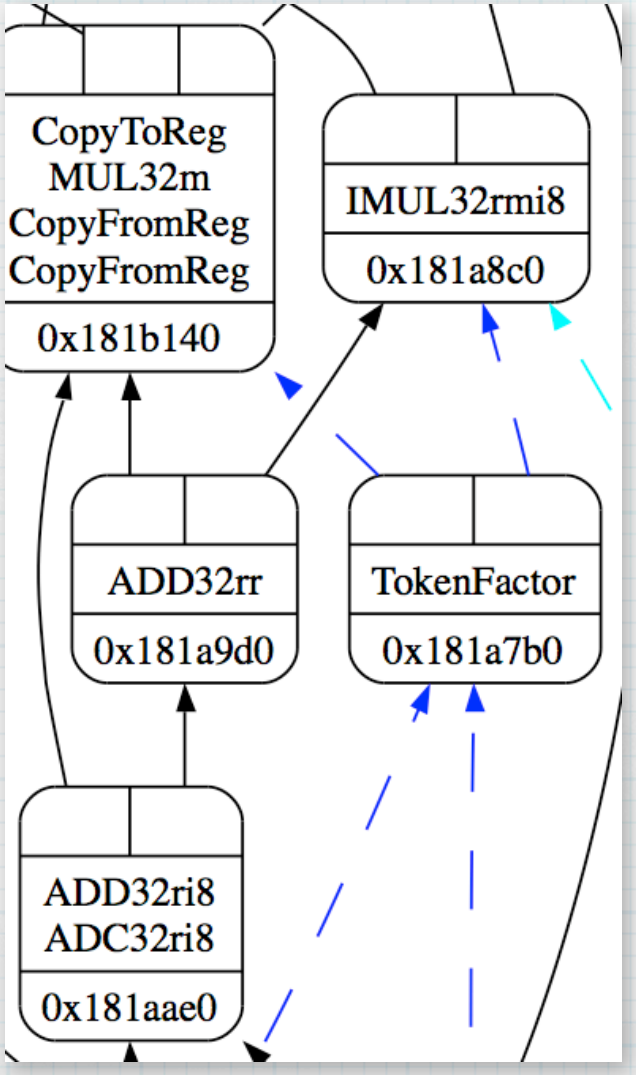
SUnit

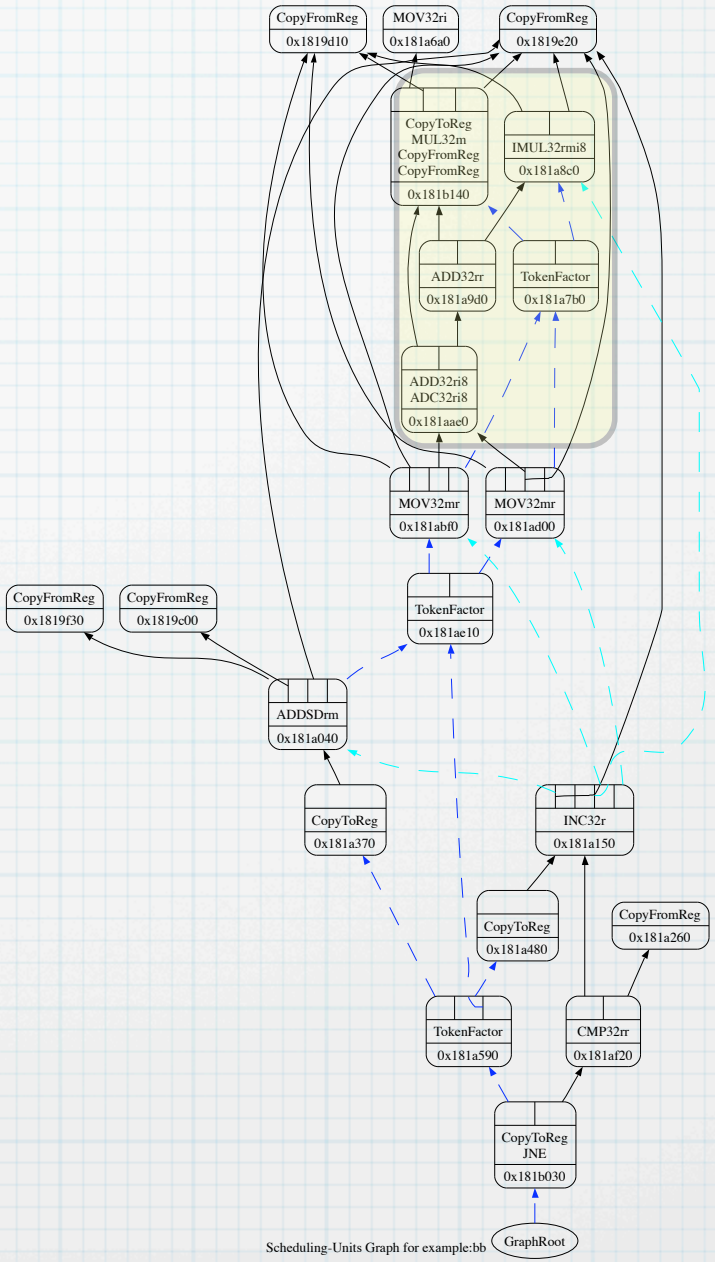


ScheduleDAG

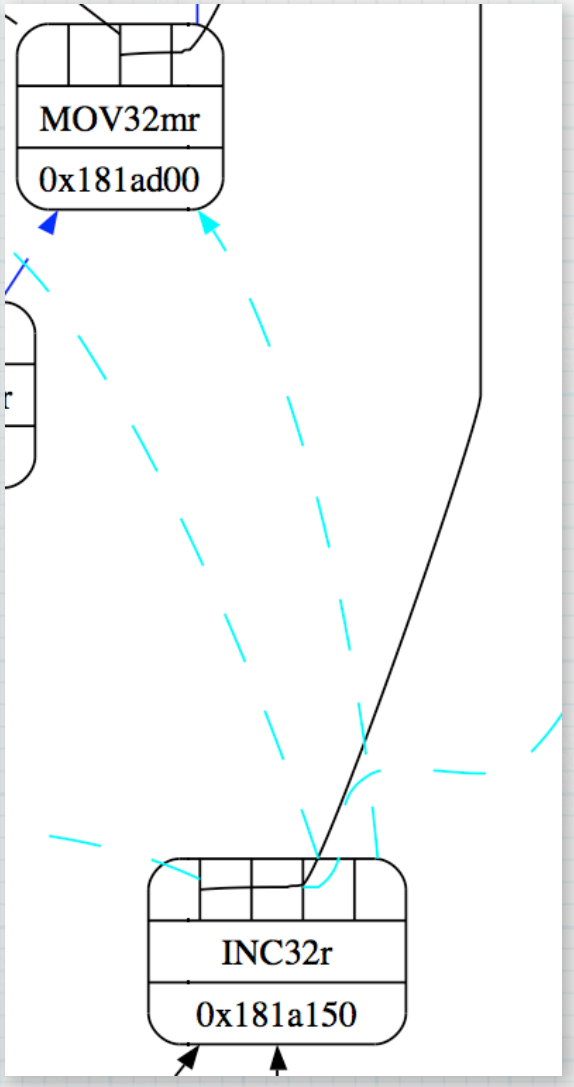


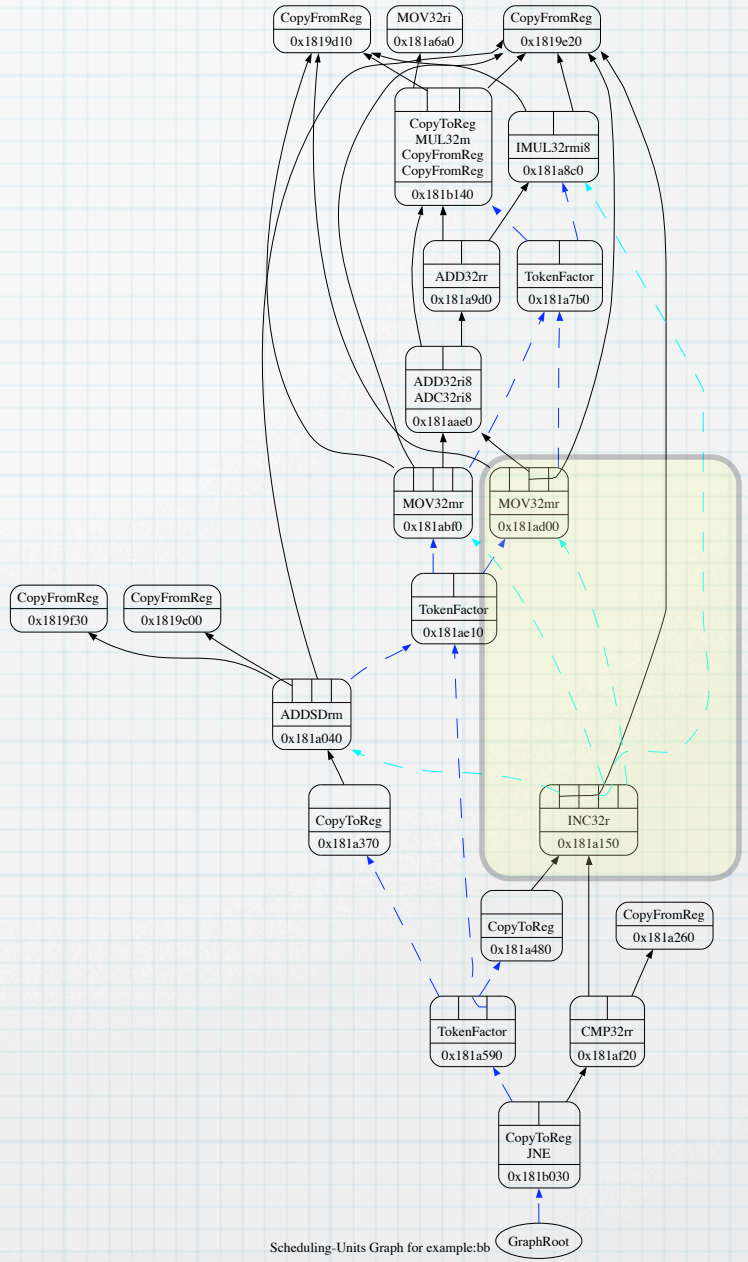
Schedule


CopyToReg
MUL32m
CopyFromReg
CopyFromReg
0x181b140
IMUL32rmi8
0x181a8c0
ADD32rr
0x181a9d0
TokenFactor
0x181a7b0
ADD32ri8
ADC32ri8
0x181aae0


CopyFromReg
0x1819d10
MOV32ri
0x181a6a0
CopyFromReg
0x1819e20
CopyToReg
MUL32m
CopyFromReg
CopyFromReg
0x181b140
IMUL32rmi8
0x181a8c0
ADD32rr
0x181a9d0
TokenFactor
0x181a7b0
ADD32ri8
ADC32ri8
0x181aae0
MOV32mr
0x181abf0
MOV32mr
0x181ad00
CopyFromReg
0x1819f30
CopyFromReg
0x1819c00
TokenFactor
0x181ae10
ADDSDrm
0x181a040
CopyToReg
0x181a370
INC32r
0x181a150
CopyToReg
0x181a480
CopyFromReg
0x181a260
TokenFactor
0x181a590
CMP32rr
0x181af20
CopyToReg
JNE
0x181b030
GraphRoot
Scheduling-Units Graph for example:bb

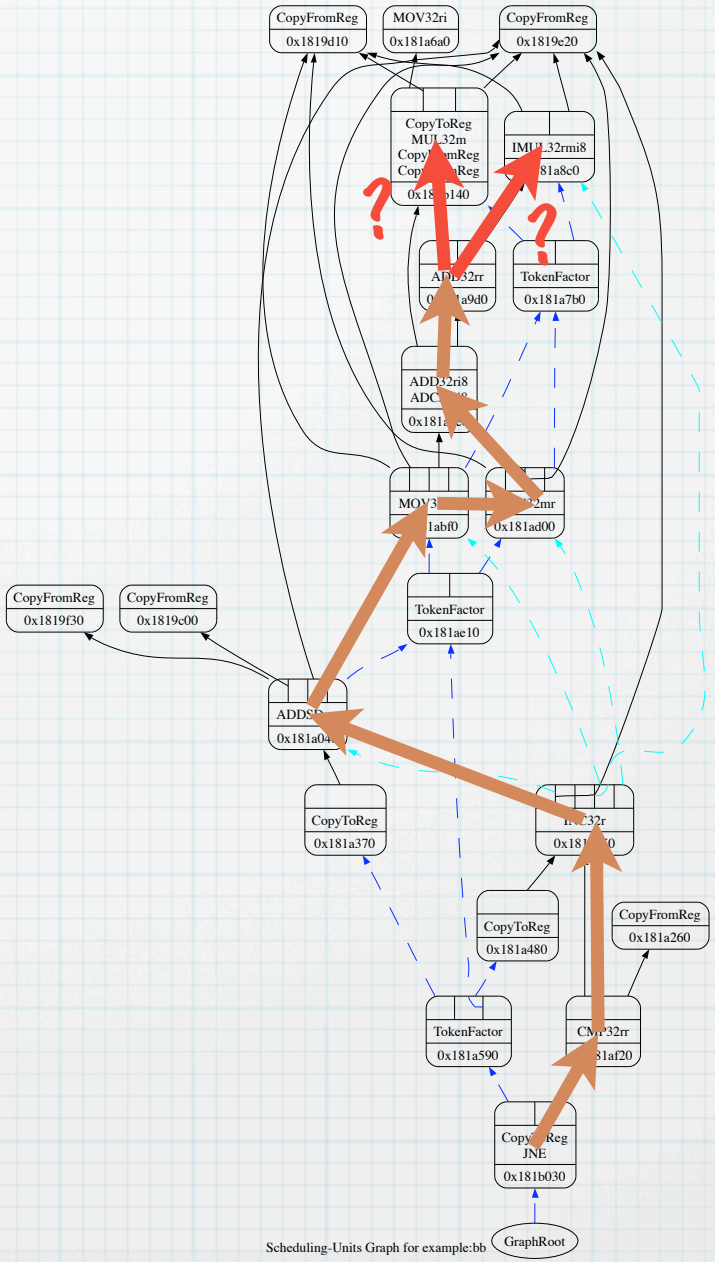
Schedule


MOV32mr
0x181ad00
INC32r
0x181a150


CopyFromReg
0x1819d10
MOV32ri
0x181a6a0
CopyFromReg
0x1819e20
CopyToReg
MUL32m
CopyFromReg
CopyFromReg
0x181b140
IMUL32rmi8
0x181a8c0
ADD32rr
0x181a9d0
TokenFactor
0x181a7b0
ADD32ri8
ADC32ri8
0x181aae0
MOV32mr
0x181abf0
MOV32mr
0x181ad00
CopyFromReg
0x1819f30
CopyFromReg
0x1819c00
TokenFactor
0x181ae10
ADDSDrm
0x181a040
CopyToReg
0x181a370
INC32r
0x181a150
CopyToReg
0x181a480
CopyFromReg
0x181a260
TokenFactor
0x181a590
CMP32rr
0x181af20
CopyToReg
JNE
0x181b030
GraphRoot
Scheduling-Units Graph for example:bb

Schedule

- Topological Sort
- Schedule for register pressure
- Schedule for latency
- Preserve physical register dependencies



Schedule

Output: A sequence of MachineInstrs

```
%reg1027 = PHI %reg1033, mbb<entry,0x1811988>, %reg1030, mbb<bb,0x18119f8>
%reg1028 = PHI %reg1034, mbb<entry,0x1811988>, %reg1029, mbb<bb,0x18119f8>
%reg1035 = MOV32ri 101
%EAX = MOV32rr %reg1035
MUL32m %reg1025, 8, %reg1027, 0, %EAX, %EDX, %EFLAGS, %EAX
%reg1036 = MOV32rr %EAX
%reg1037 = MOV32rr %EDX
%reg1038 = IMUL32rmi8 %reg1025, 8, %reg1027, 4, 101, %EFLAGS, Mem:LD(4,4)[t7+4]
%reg1039 = ADD32rr %reg1038, %reg1037, %EFLAGS
. . .
```

SelectionDAG Future

* LegalizeTypes
* SEME regions, whole-functions
* More precise dependencies
* Fast -O0 isel?
* BURG-style isel?

CodeGen continues...

* Late Code Motion
* Register Allocation
* Output

Questions?