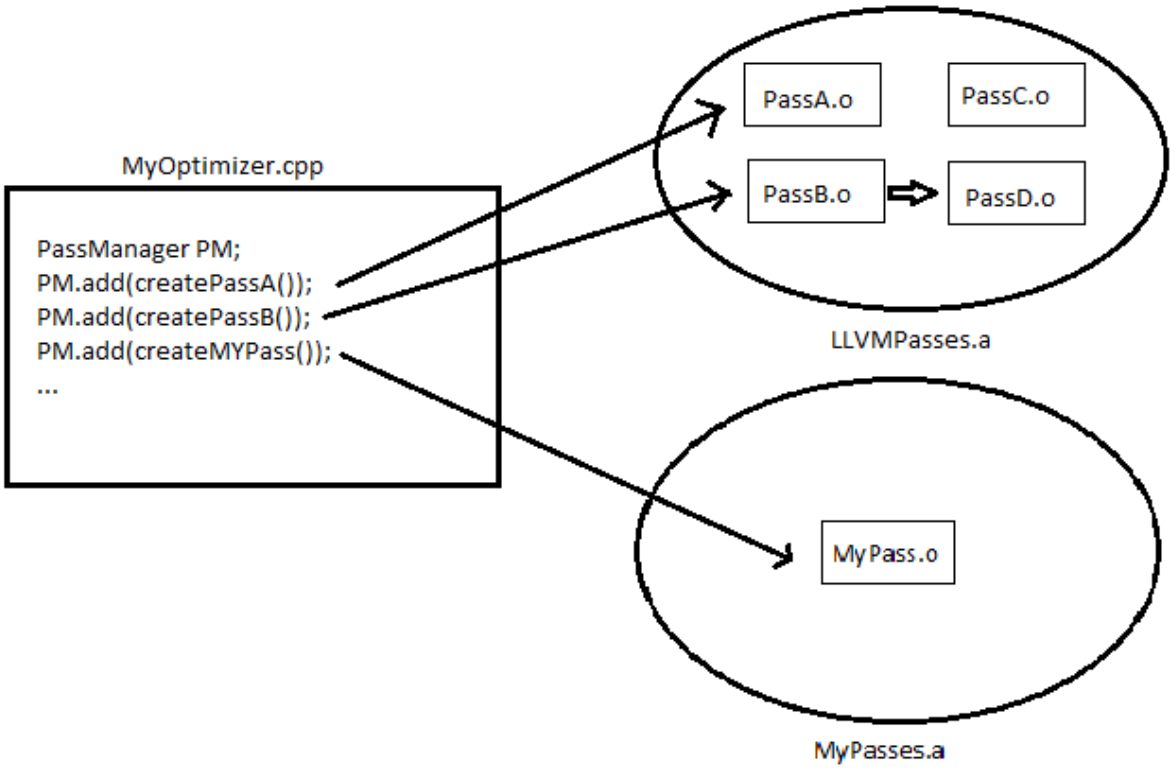
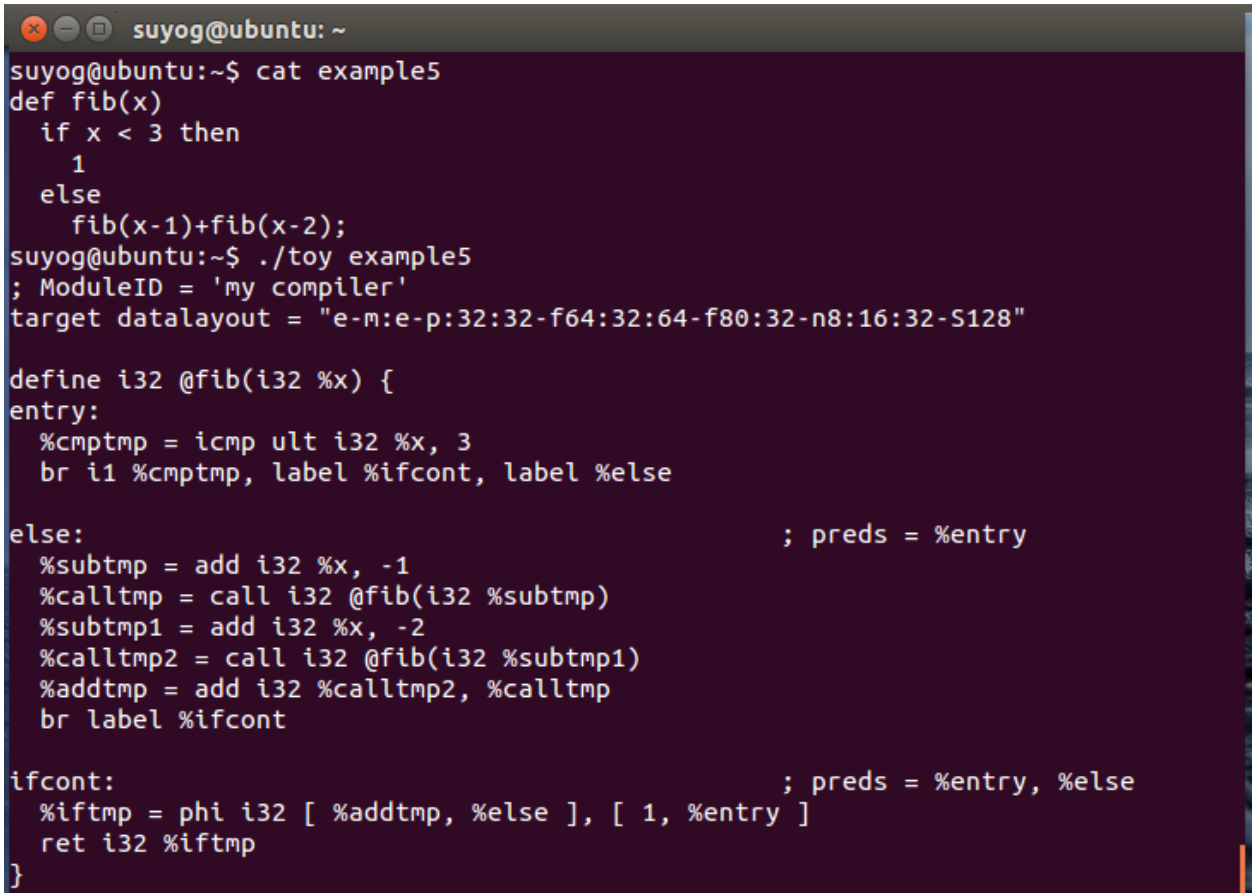
LLVM Cookbook

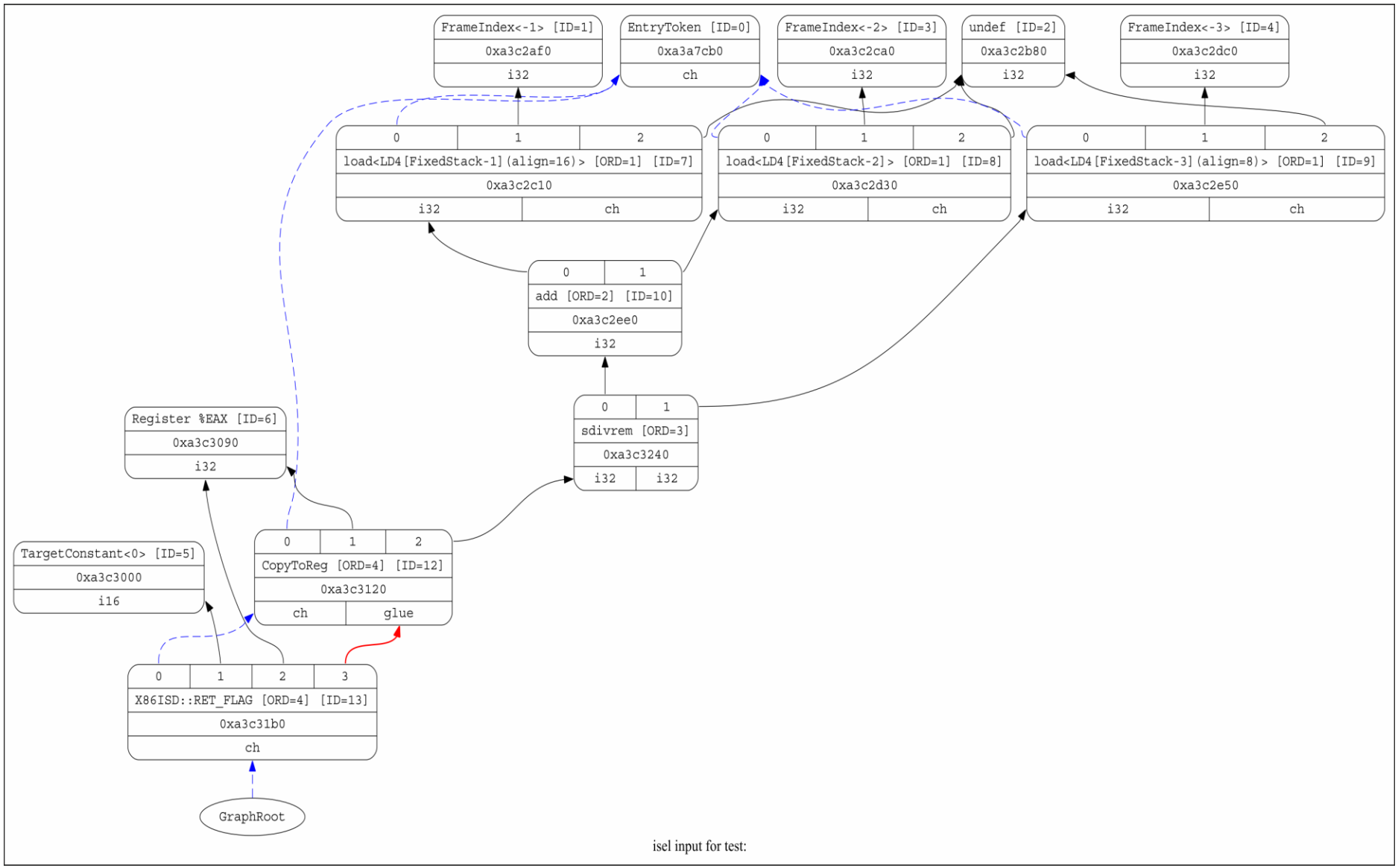
Chapter 1


$ hexdump -C test.bc
00000000 42 43 c0 de 21 0c 00 00 68 00 00 00 0b 82 20 00 |BC..!...h..... .|
00000010 02 00 00 00 13 00 00 00 07 81 23 91 41 c8 04 49 |..........#.A..I|
00000020 06 10 32 39 92 01 84 0c 25 05 08 19 1e 04 8b 62 |..29....%......b|
00000030 80 0c 45 02 42 92 0b 42 64 10 32 14 38 08 18 4b |..E.B..Bd.2.8..K|
00000040 0a 32 32 88 48 90 14 20 43 46 88 a5 00 19 32 42 |.22.H.. CF....2B|
00000050 e4 48 0e 90 91 21 c4 50 41 51 81 8c e1 83 e5 8a |.H...!.PAQ......|
00000060 04 19 46 06 89 20 00 00 0b 00 00 00 32 22 c8 08 |..F.. ......2"..|
00000070 20 64 85 04 93 21 a4 84 04 93 21 e3 84 a1 90 14 | d...!....!.....|
00000080 12 4c 86 8c 0b 84 64 4c 10 14 73 04 60 50 06 00 |.L....dL..s.`P..|
00000090 94 81 80 11 00 00 00 00 43 1c 01 00 00 00 00 00 |........C.......|
000000a0 00 00 00 00 c8 c3 00 00 32 00 00 00 33 08 80 1c |........2...3...|
000000b0 c4 e1 1c 66 14 01 3d 88 43 38 84 c3 8c 42 80 07 |...f..=.C8...B..|
000000c0 79 78 07 73 98 71 0c e6 00 0f ed 10 0e f4 80 0e |yx.s.q..........|
000000d0 33 0c 42 1e c2 c1 1d ce a1 1c 66 30 05 3d 88 43 |3.B.......f0.=.C|
000000e0 38 84 83 1b cc 03 3d c8 43 3d 8c 03 3d cc 78 8c |8.....=.C=..=.x.|
000000f0 74 70 07 7b 08 07 79 48 87 70 70 07 7a 70 03 76 |tp.{..yH.pp.zp.v|
00000100 78 87 70 20 87 19 cc 11 0e ec 90 0e e1 30 0f 6e |x.p ....0.n|
00000110 30 0f e3 f0 0e f0 50 0e 33 10 c4 1d de 21 1c d8 |0.....P.3....!..|
00000120 21 1d c2 61 1e 66 30 89 3b bc 83 3b d0 43 39 b4 |!..a.f0.;..;.C9.|
00000130 03 3c bc 83 3c 84 03 3b cc f0 14 76 60 07 7b 68 |.<..<..;...v`.{h|
00000140 07 37 68 87 72 68 07 37 80 87 70 90 87 70 60 07 |.7h.rh.7..p..p`.|
00000150 76 28 07 76 f8 05 76 78 87 77 80 87 5f 08 87 71 |v(.v..vx.w.._..q|
00000160 18 87 72 98 87 79 98 81 2c ee f0 0e ee e0 0e f5 |..r..y..,.......|
00000170 c0 0e ec 00 71 20 00 00 02 00 00 00 06 40 30 d4 |....q ....@0.|
00000180 32 01 00 00 61 20 00 00 07 00 00 00 13 04 81 09 |2...a ....|
00000190 81 08 32 08 07 02 00 00 02 00 00 00 16 10 00 26 |..2...........&|
000001a0 10 04 00 00 00 00 00 00 00 00 00 00 |............|
000001ac
$

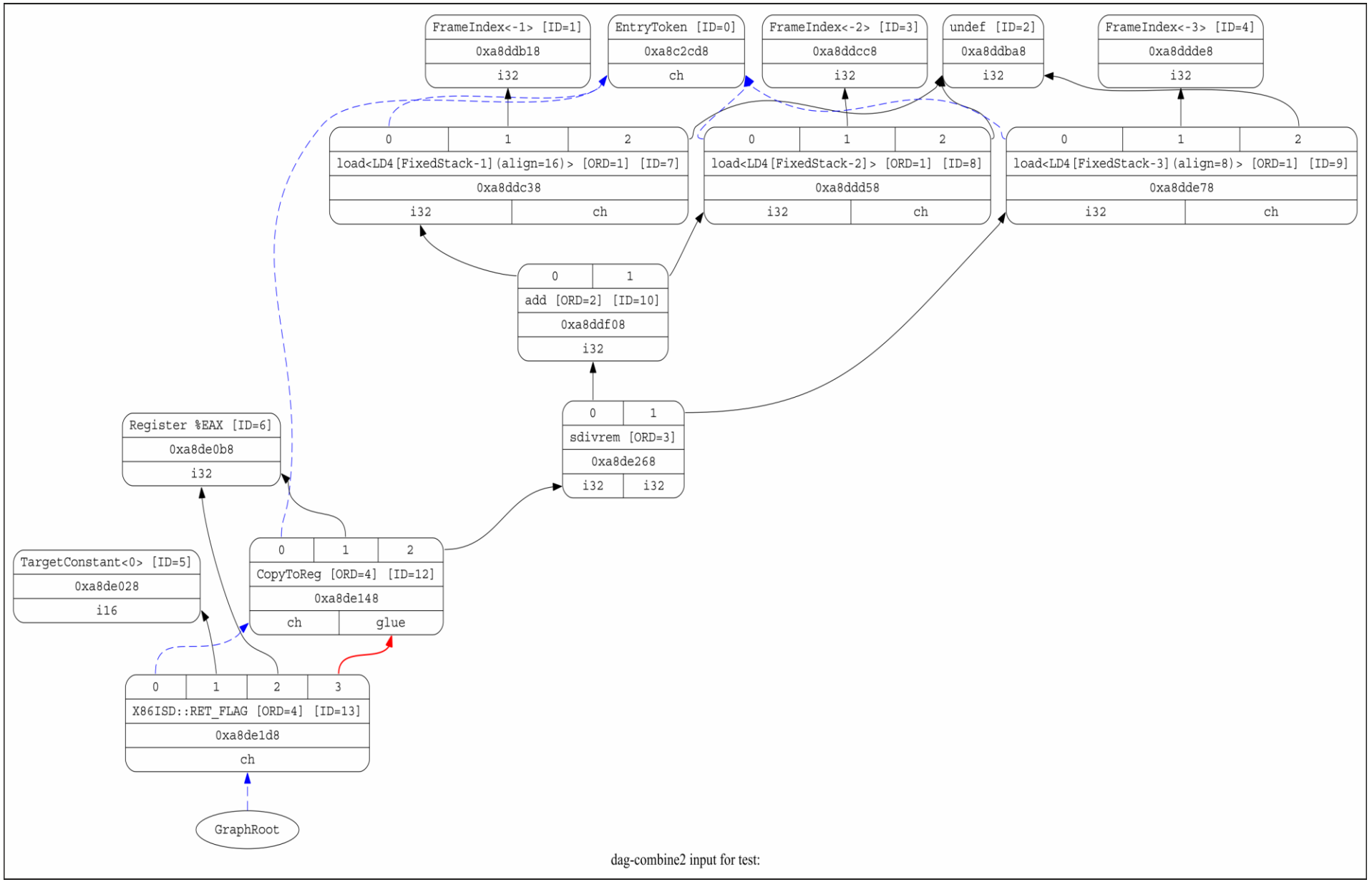

MyOptimizer.cpp
PassManager PM;
PM.add(createPassA());
PM.add(createPassB());
PM.add(createMYPass());
...
PassA.o
PassC.o
PassB.o
PassD.o
LLVMPasses.a
MyPass.o
MyPasses.a

Chapter 3

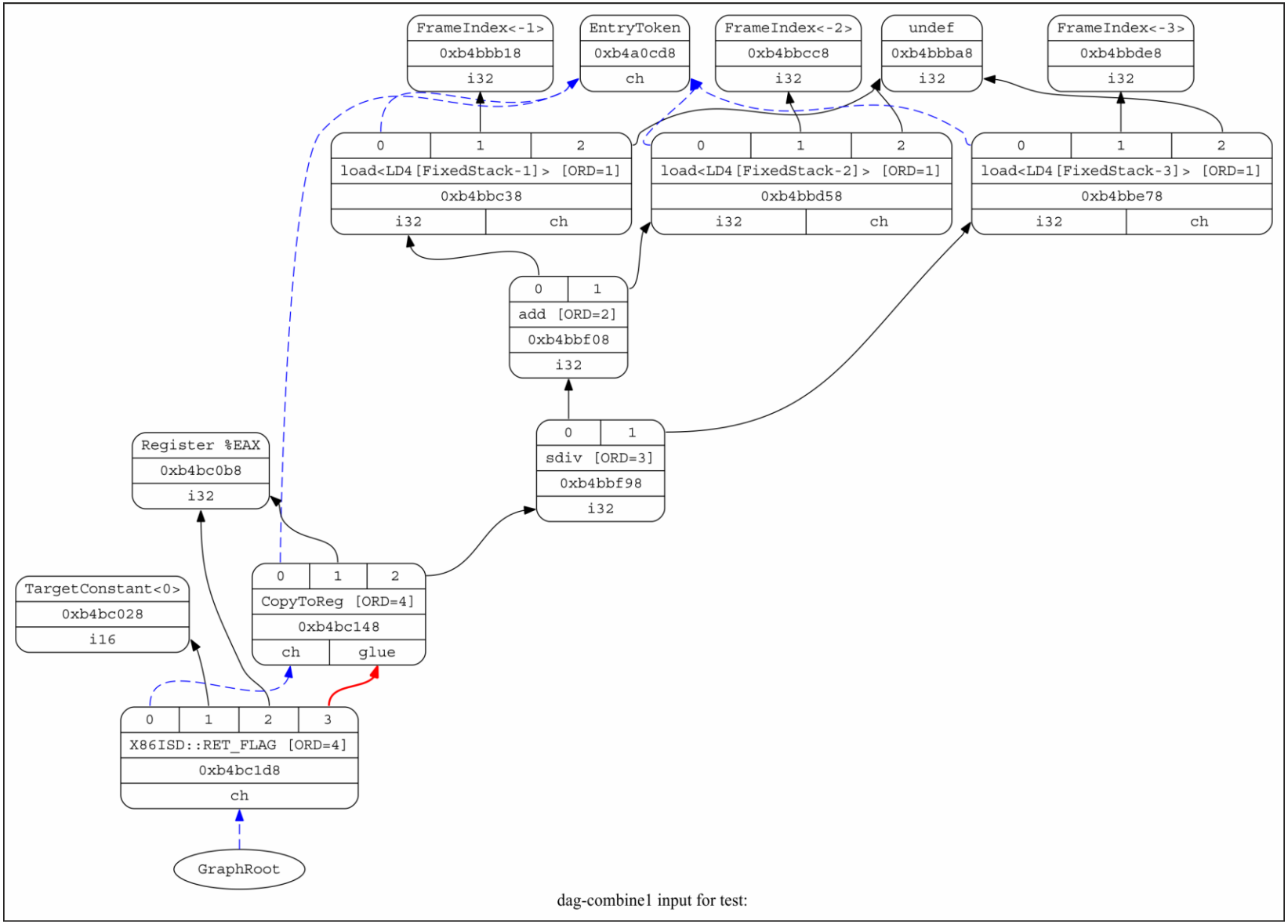

suyog@ubuntu: ~
suyog@ubuntu:~$ cat example5
def fib(x)
  if x < 3 then
    1
  else
    fib(x-1)+fib(x-2);
suyog@ubuntu:~$ ./toy example5
; ModuleID = 'my compiler'
target datalayout = "e-m:e-p:32:32-f64:32:64-f80:32-n8:16:32-S128"
define i32 @fib(i32 %x) {
entry:
  %cmptmp = icmp ult i32 %x, 3
  br i1 %cmptmp, label %ifcont, label %else
else: ; preds = %entry
  %subtmp = add i32 %x, -1
  %calltmp = call i32 @fib(i32 %subtmp)
  %subtmp1 = add i32 %x, -2
  %calltmp2 = call i32 @fib(i32 %subtmp1)
  %addtmp = add i32 %calltmp2, %calltmp
  br label %ifcont
ifcont: ; preds = %entry, %else
  %iftmp = phi i32 [%addtmp, %else], [1, %entry]
  ret i32 %iftmp
}



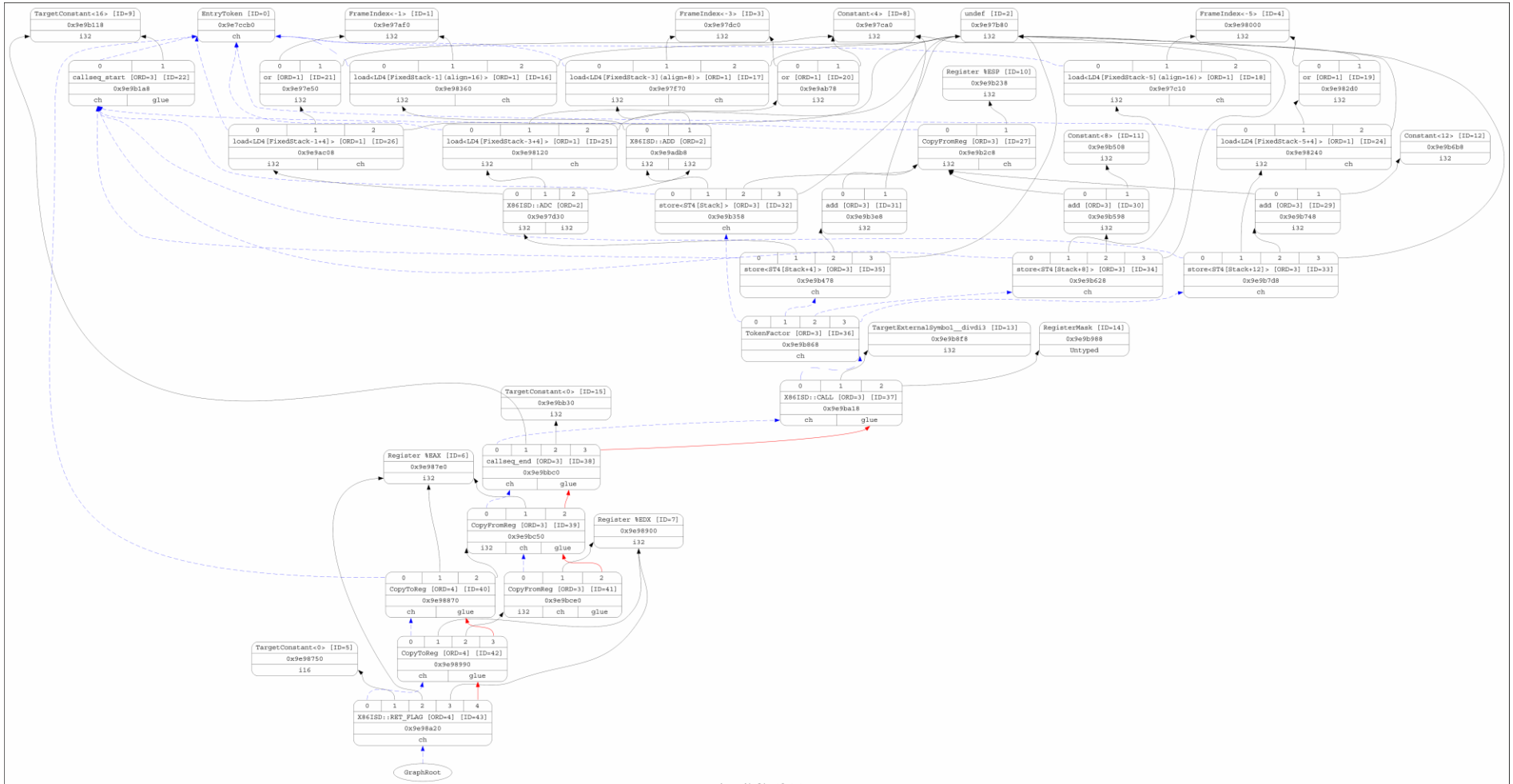
isel input for test:



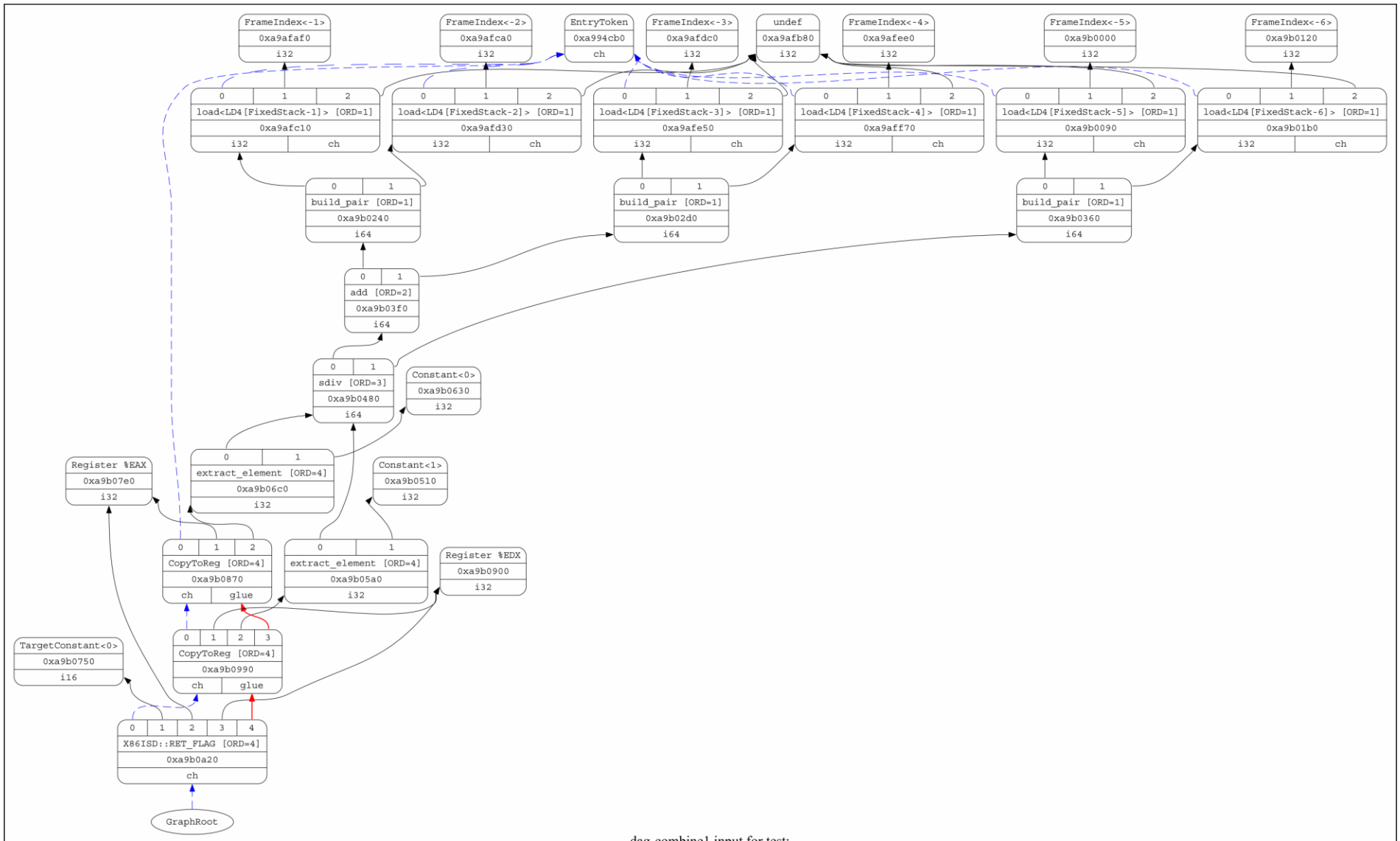
dag-combine2 input for test:



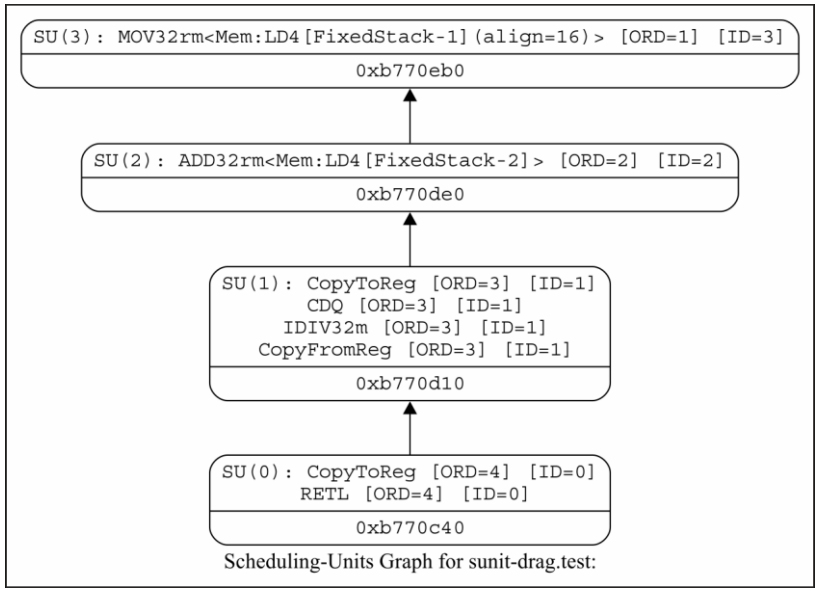
dag-combine1 input for test:



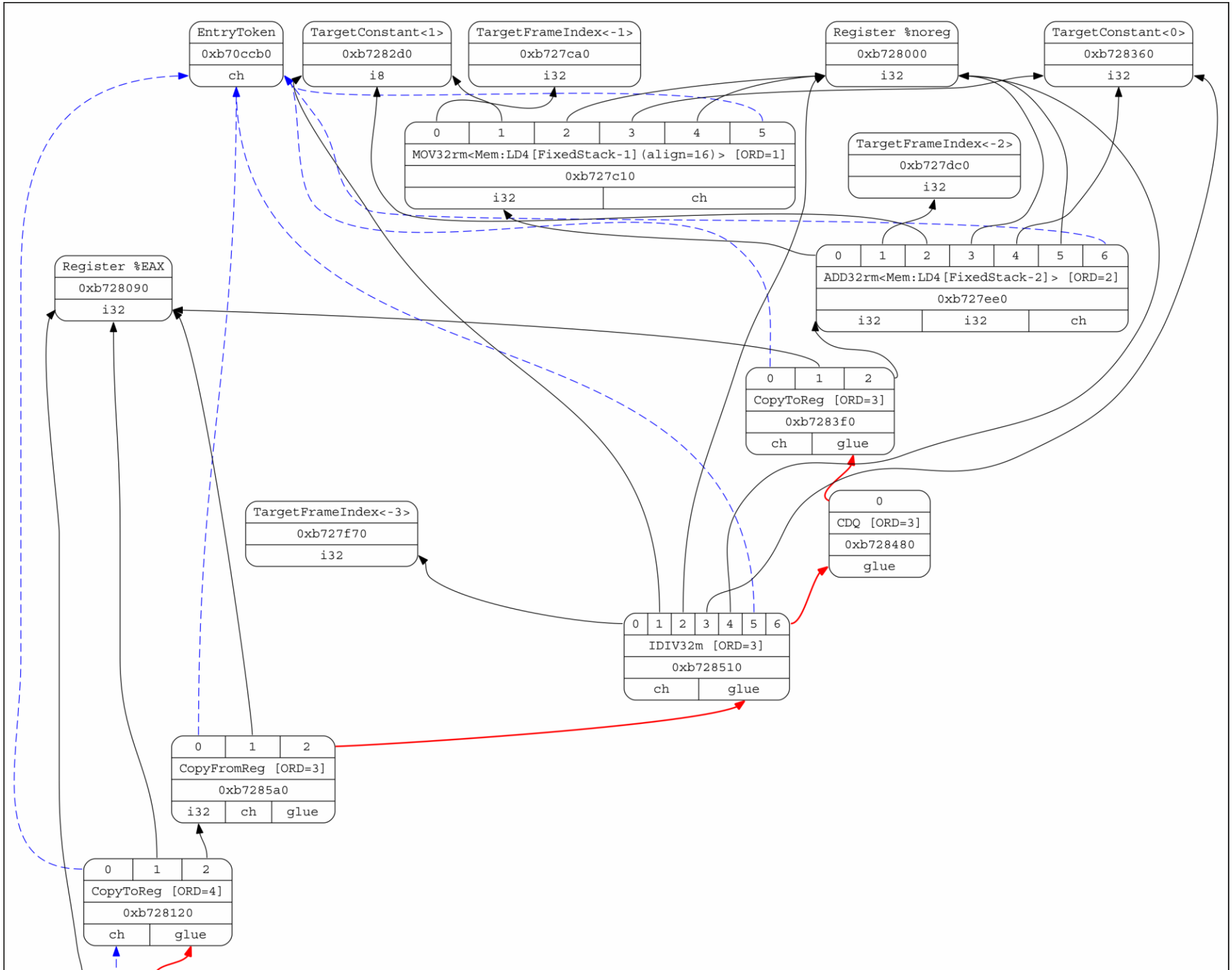
dag-combine2 input for test:

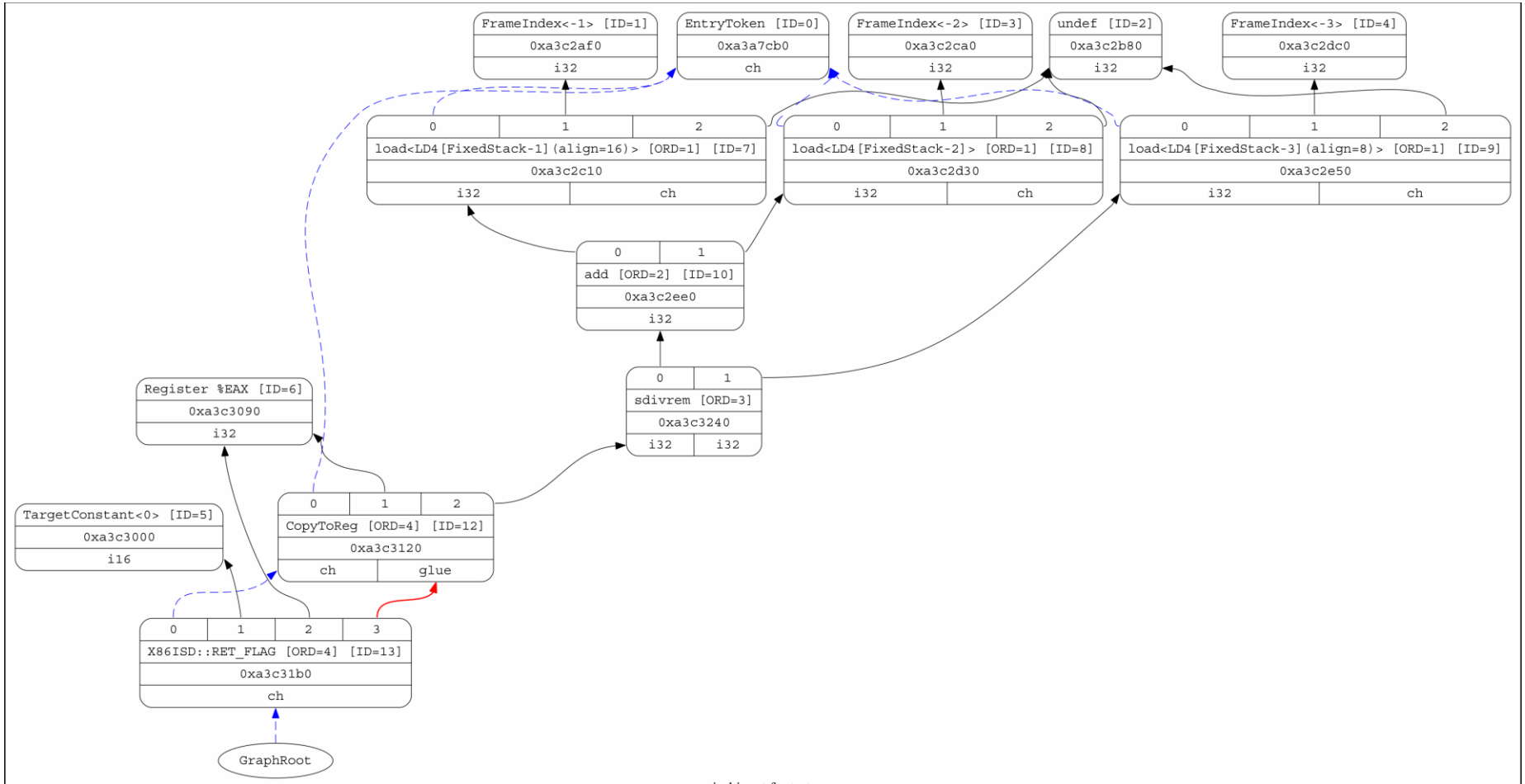


dag-combine1 input for test:

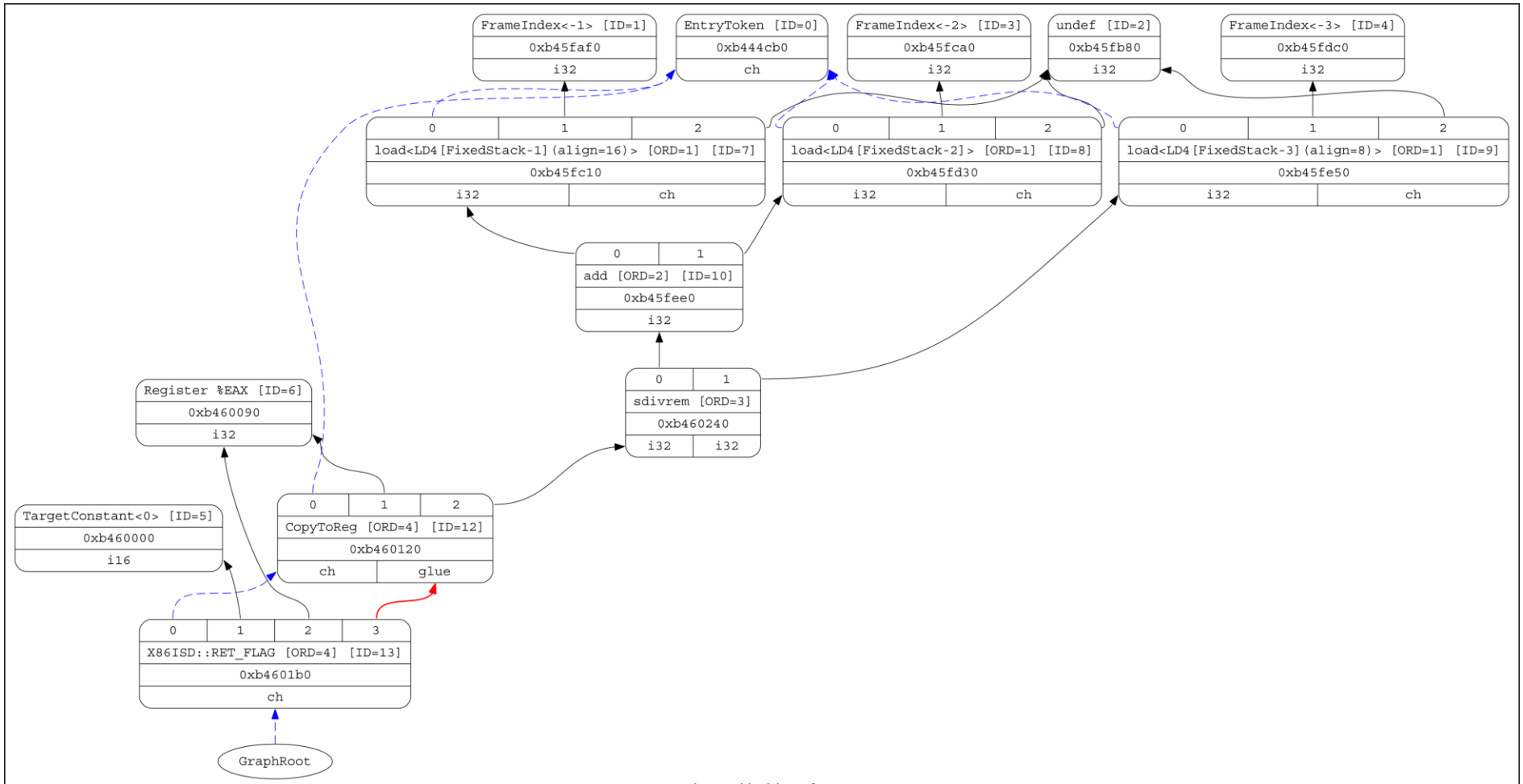


Scheduling-Units Graph for sunit-drag.test:

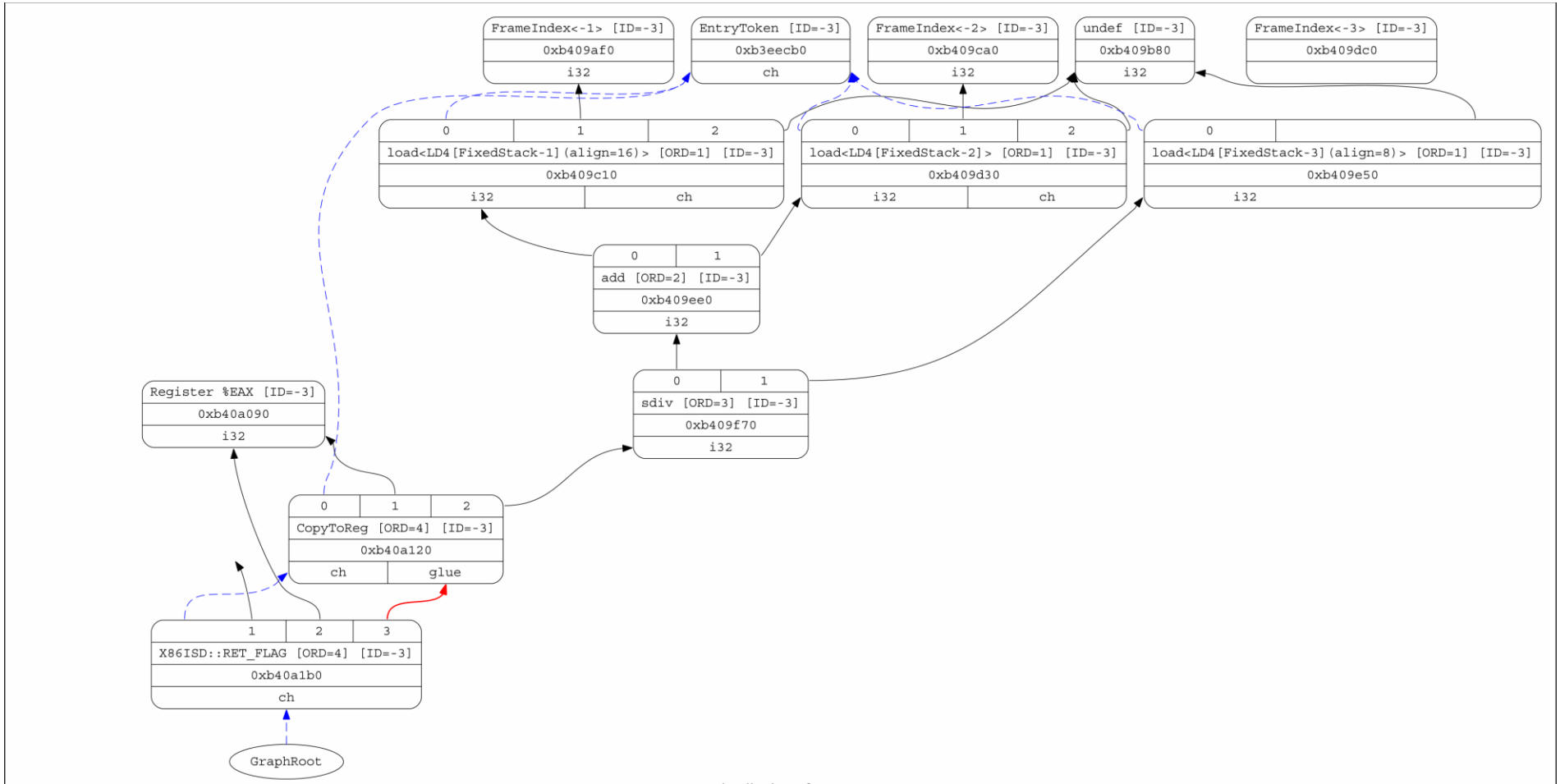

EntryToken
0xb70ccb0
ch
TargetConstant<1>
0xb7282d0
i8
TargetFrameIndex<-1>
0xb727ca0
i32
Register %noreg
0xb728000
i32
TargetConstant<0>
0xb728360
i32
0 1 2 3 4 5
MOV32rm<Mem:LD4[FixedStack-1](align=16)> [ORD=1]
0xb727c10
i32 ch
TargetFrameIndex<-2>
0xb727dc0
i32
0 1 2 3 4 5 6
ADD32rm<Mem:LD4[FixedStack-2]> [ORD=2]
0xb727ee0
i32 i32 ch
Register %EAX
0xb728090
i32
0 1 2
CopyToReg [ORD=3]
0xb7283f0
ch glue
TargetFrameIndex<-3>
0xb727f70
i32
0
CDQ [ORD=3]
0xb728480
glue
0 1 2 3 4 5 6
IDIV32m [ORD=3]
0xb728510
ch glue
0 1 2
CopyFromReg [ORD=3]
0xb7285a0
i32 ch glue
0 1 2
CopyToReg [ORD=4]
0xb728120
ch glue



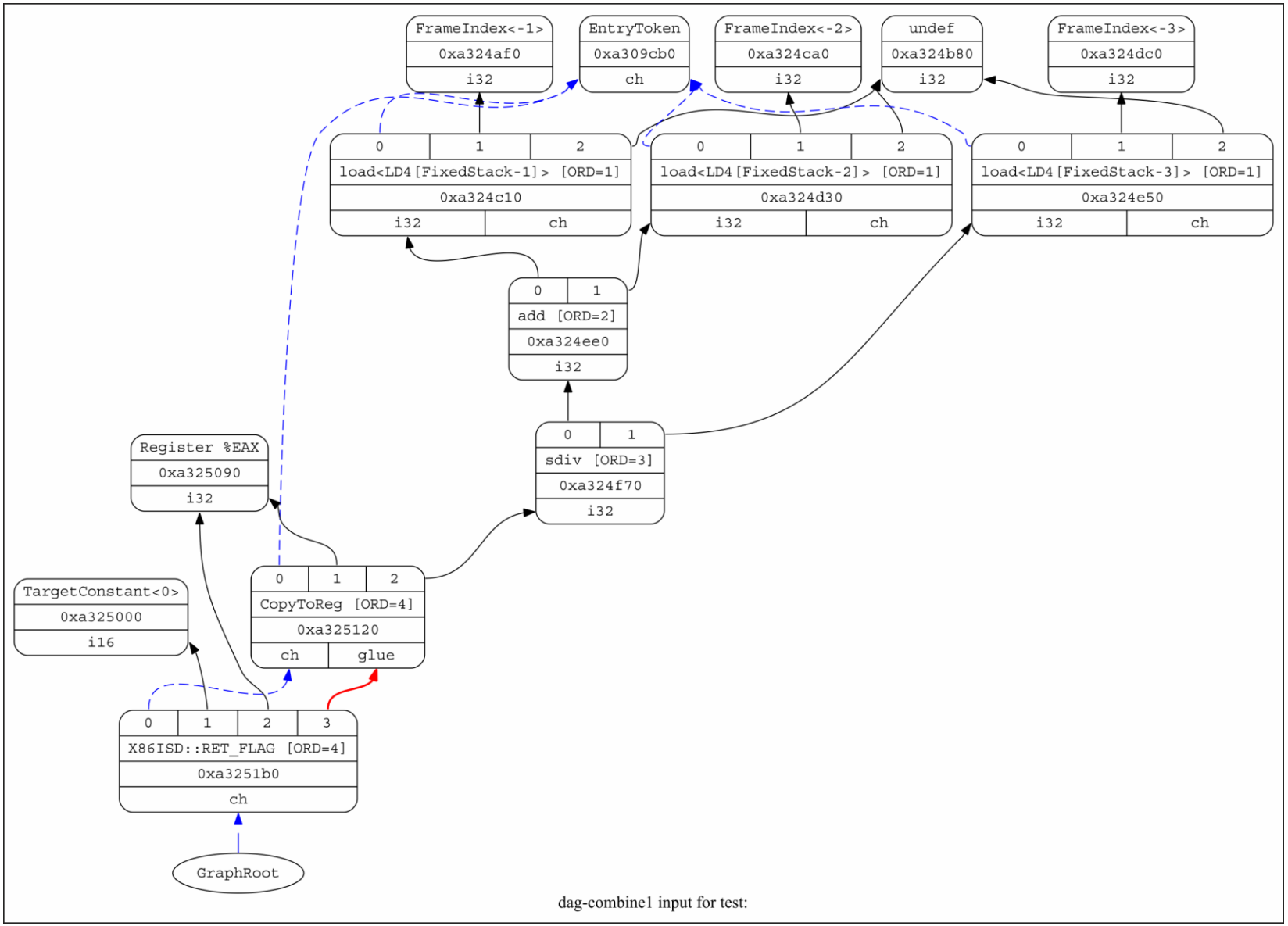
isel input for test:



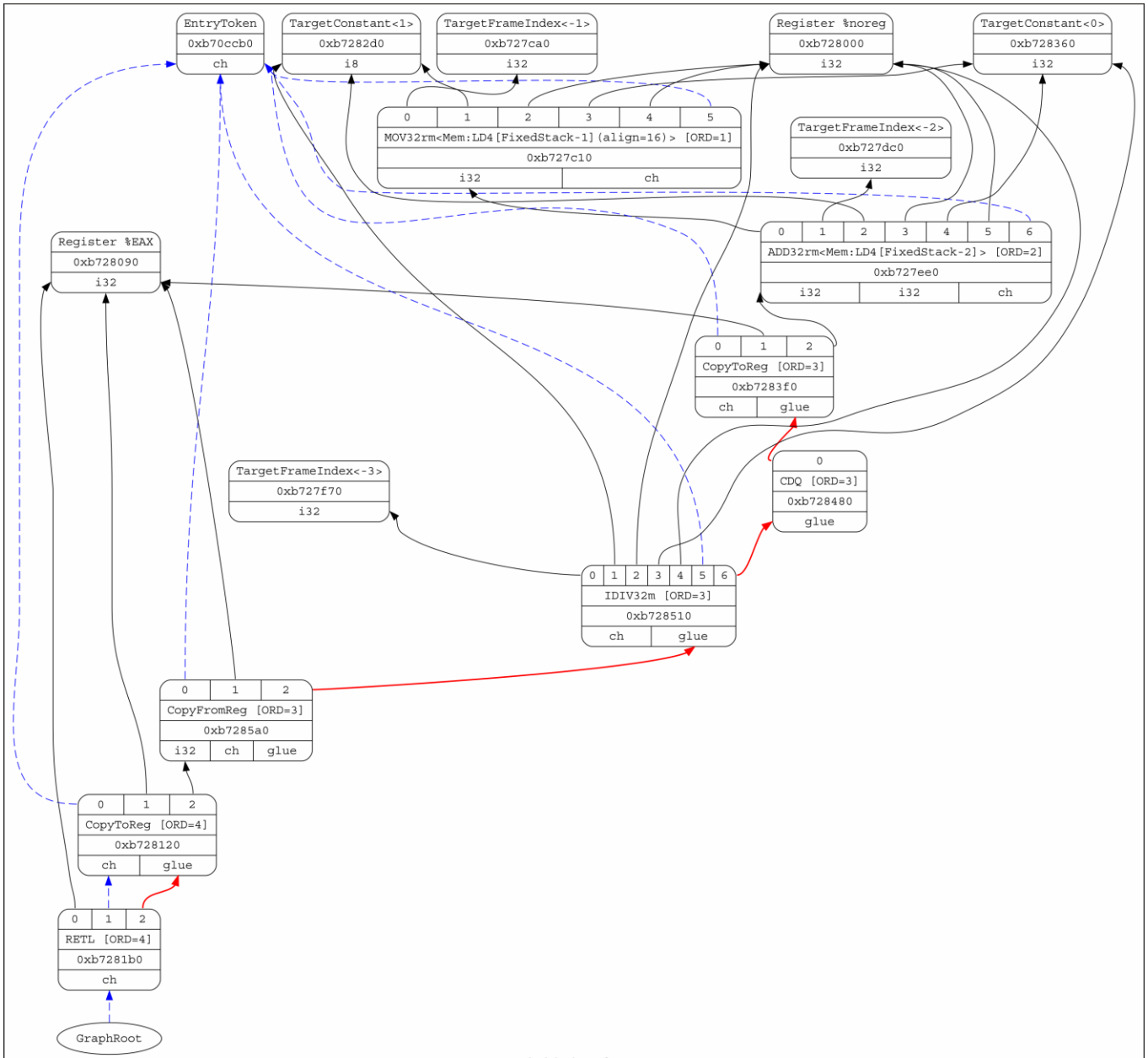
dag-combine2 input for test:



legalize input for test:

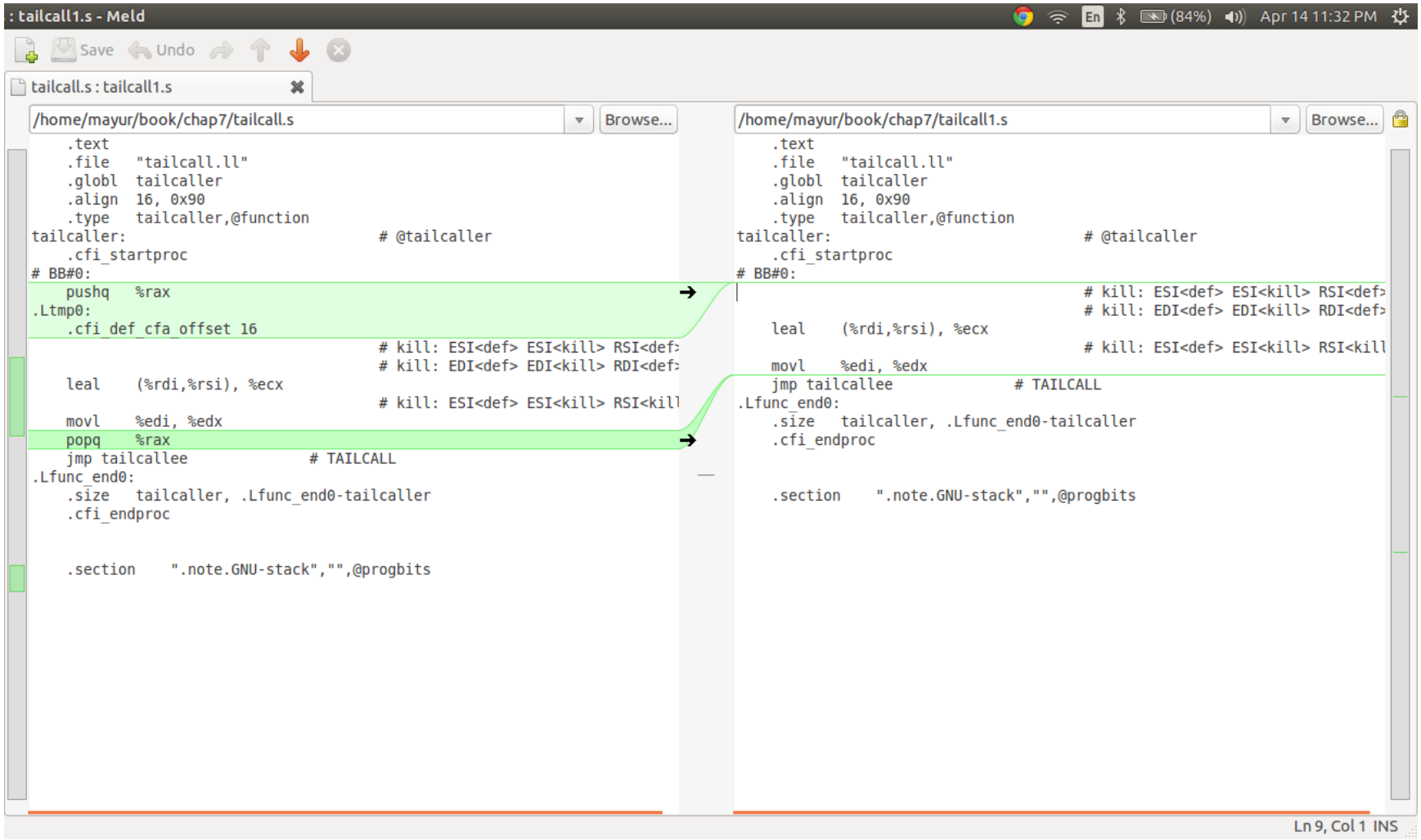


dag-combine1 input for test:



scheduler input for test:

Chapter 7

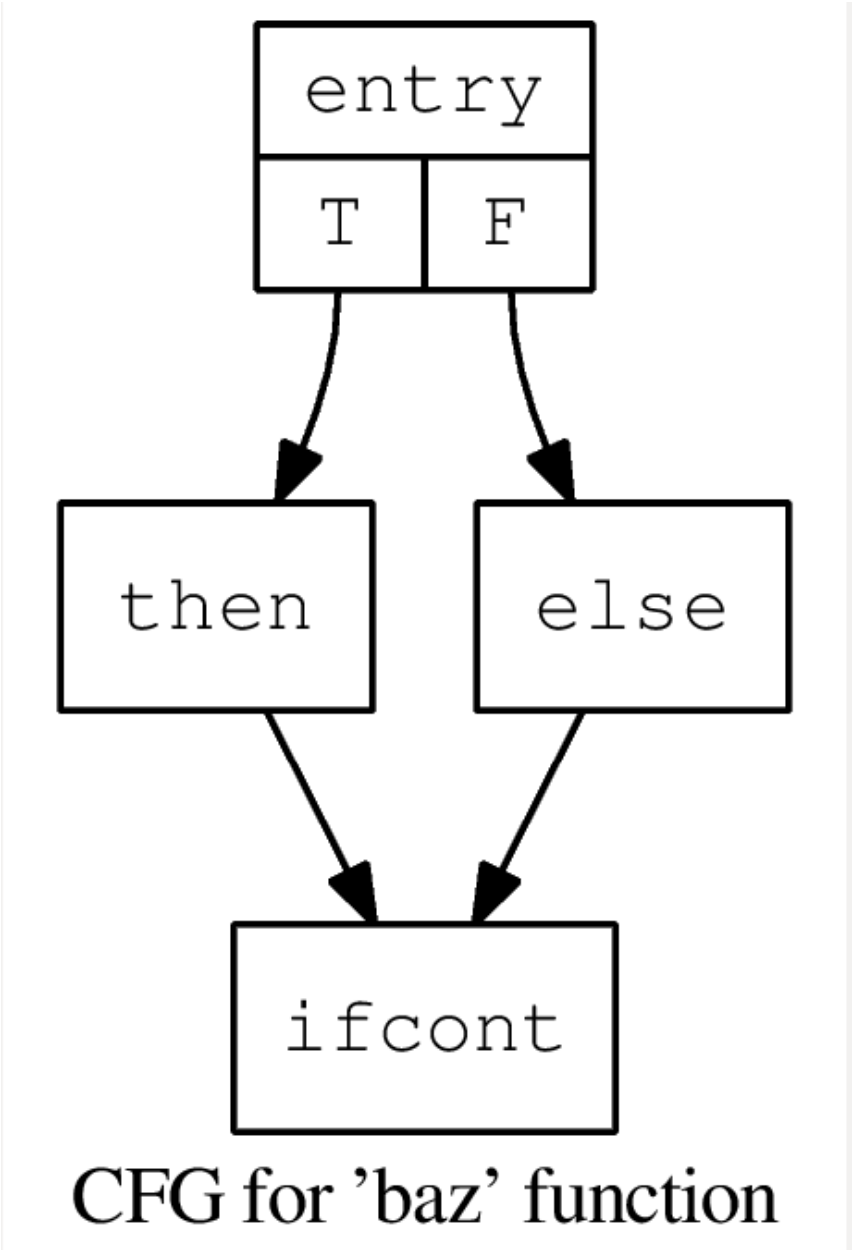

: tailcall1.s - Meld
Save
Undo
tailcall.s : tailcall1.s
/home/mayur/book/chap7/tailcall.s
Browse...
/home/mayur/book/chap7/tailcall1.s
Browse...
.text
.file "tailcall.ll"
.globl tailcaller
.align 16, 0x90
.type tailcaller,@function
tailcaller: # @tailcaller
.cfi_startproc
BB#0:
pushq %rax
.Ltmp0:
.cfi_def_cfa_offset 16
kill: ESI<def> ESI<kill> RSI<def>
kill: EDI<def> EDI<kill> RDI<def>
leal (%rdi,%rsi), %ecx
# kill: ESI<def> ESI<kill> RSI<kill>
movl %edi, %edx
popq %rax
jmp tailcallee # TAILCALL
.Lfunc_end0:
.size tailcaller, .Lfunc_end0-tailcaller
.cfi_endproc
.section ".note.GNU-stack","",@progbits
jmp tailcallee # TAILCALL
Ln 9, Col 1 INS

Chapter 9

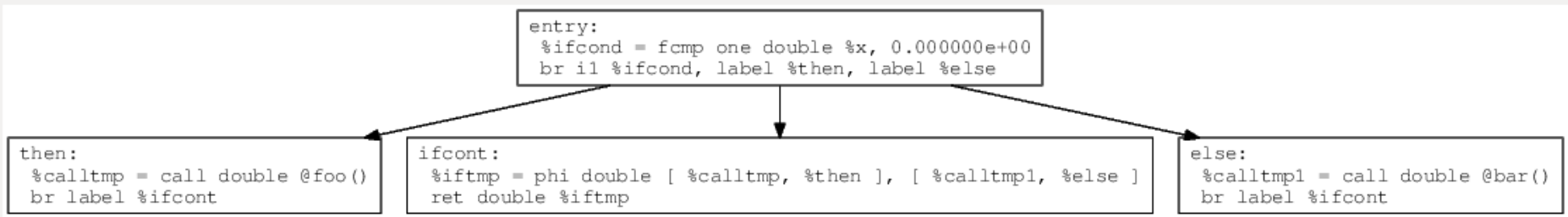
```
mayur@vaio-linux:~/book/chap9$ ASAN_SYMBOLIZER_PATH=/usr/local/bin/llvm-symbolizer ./a.out
=================================================================
==22656==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7fff273c21b8 at pc 0x0000004d406b bp 0x7fff273c2150 sp 0x7fff273c2148
READ of size 4 at 0x7fff273c21b8 thread T0
    #0 0x4d406a in main (/home/mayur/book/chap9/a.out+0x4d406a)
    #1 0x7f8673850ec4 in __libc_start_main /build/buildd/eglibc-2.19/csu/libc-start.c:287
    #2 0x4176a5 in _start (/home/mayur/book/chap9/a.out+0x4176a5)

Address 0x7fff273c21b8 is located in stack of thread T0 at offset 56 in frame
    #0 0x4d3f4f in main (/home/mayur/book/chap9/a.out+0x4d3f4f)

  This frame has 1 object(s):
    [32, 52) 'a' <== Memory access at offset 56 overflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanism or swapcontext
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow (/home/mayur/book/chap9/a.out+0x4d406a) in main
Shadow bytes around the buggy address:
  0x100064e703e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100064e703f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100064e70400: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100064e70410: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100064e70420: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x100064e70430: f1 f1 f1 f1 00 00 04[f3]f3 f3 f3 f3 00 00 00 00
  0x100064e70440: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100064e70450: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100064e70460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100064e70470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100064e70480: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Heap right redzone:      fb
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack partial redzone:   f4
  Stack after return:      f5
```



CFG for 'baz' function



Dominator tree for 'baz' function