指令调度

Agenda

- 指令调度原理
- LLVM中的指令调度器及其工作过程
- 调度pass的定制

指令调度原理

- 指令调度原理

```
load r1, a
load r2, b
add r3, r1, r2
load r4, c
load r5, d
mul r6, r4, r5
add r7, r3, r6
store e, r7
```

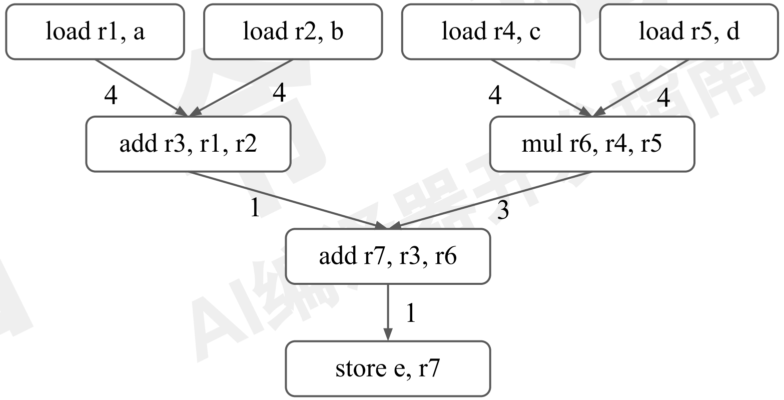


图4-10　数据依赖图示例

```
1 : load r1, a
2 : load r2, b
3-5 : stall
6 : add r3, r1, r2
7 : load r4, c
8 : load r5, d
9-11 : stall
12 : mul r6, r4, r5
13-14 : stall
15 : add r7, r3, r6
16 : store e, r7
```

指令调度原理

- 指令调度原理 - 关键路径优先

```
1 : load r5, d
2 : load r4, c
3 : load r2, b
4 : load r1, a
5 : stall
6 : mul r6, r4, r5
7 : stall
8 : add r3, r1, r2
9 : add r7, r3, r6
10 : store e, r7
```

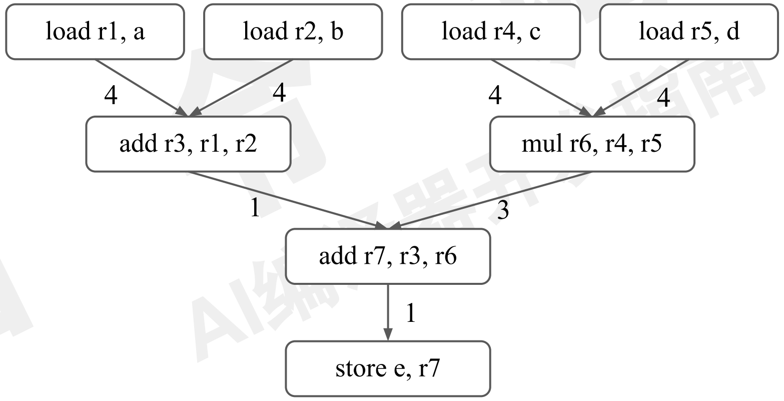


图4-2 LLVM后端执行流程

LLVM中的指令调度器及其工作过程

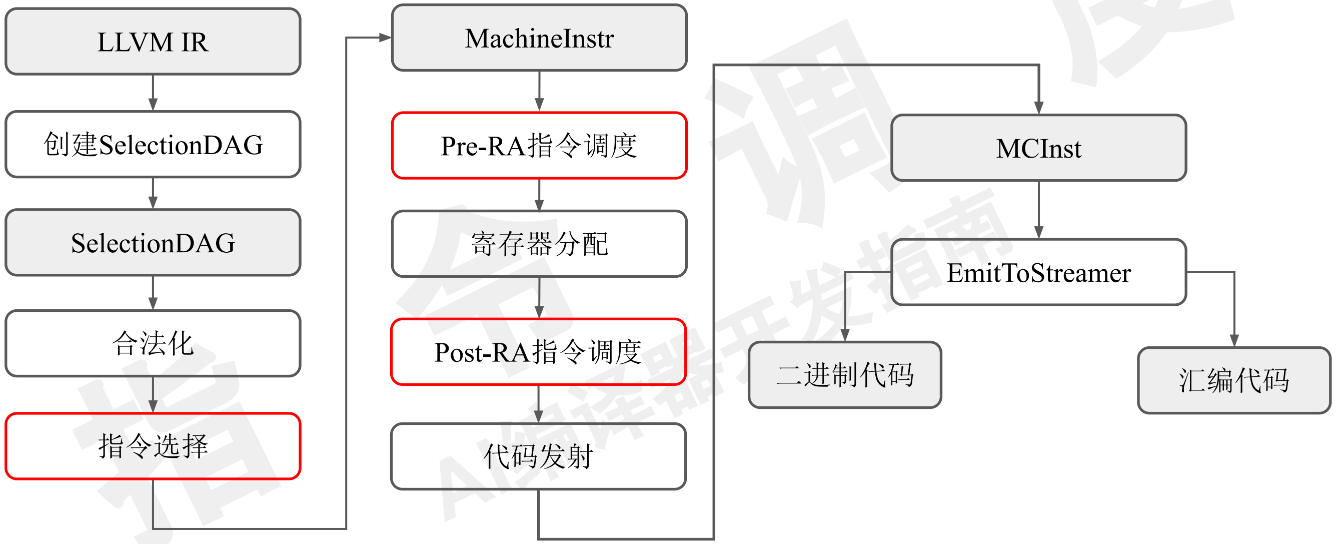
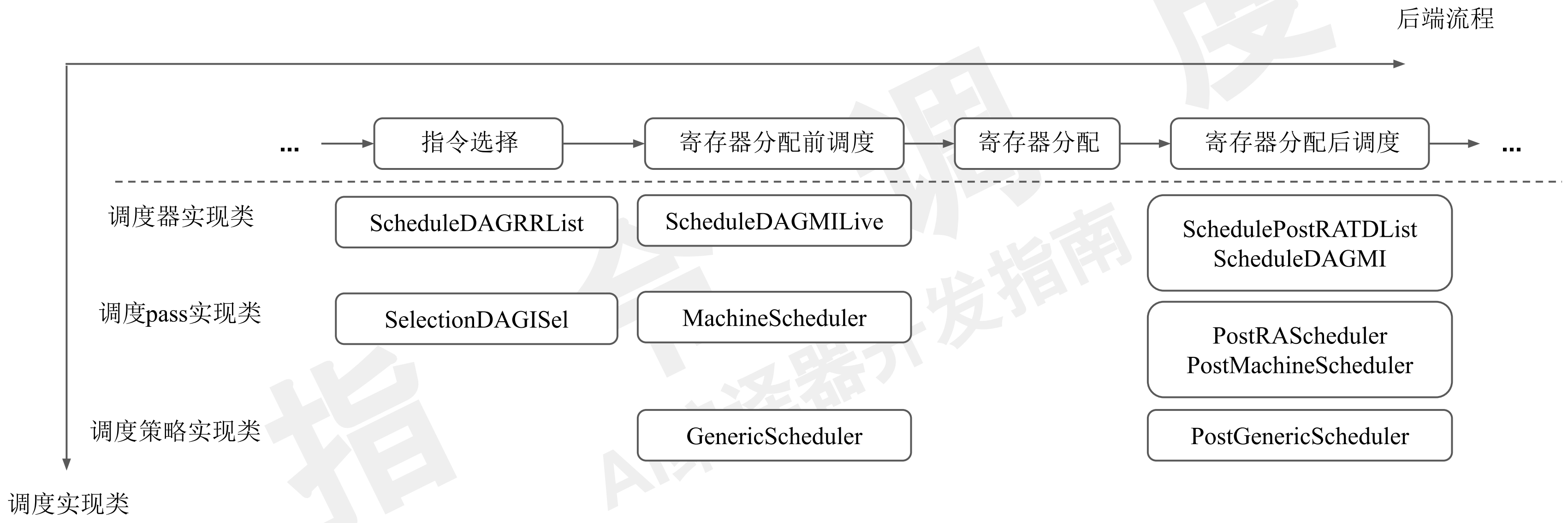


图4-2　LLVM后端执行流程

LLVM中的指令调度器及其工作过程



指令选择阶段的调度器 - 0

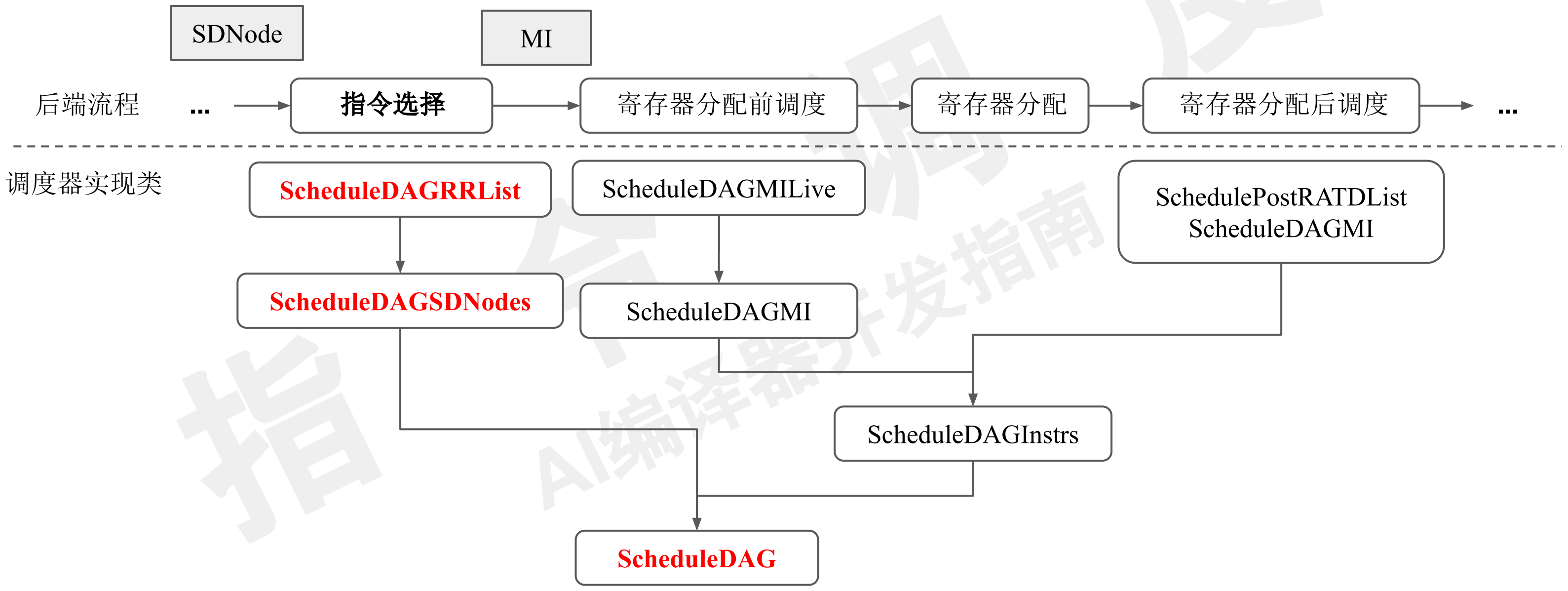


图4-14　调度器类继承关系

指令选择阶段的调度器 - 1

- ScheduleDAGRRList
  - burrListDAGScheduler：减少寄存器用量
  - sourceListDAGScheduler：按源码顺序调度
  - hybridListDAGScheduler：寄存器压力敏感
  - ILPListDAGScheduler：兼顾指令级并行度和寄存器压力

指令选择阶段的调度器 - 2

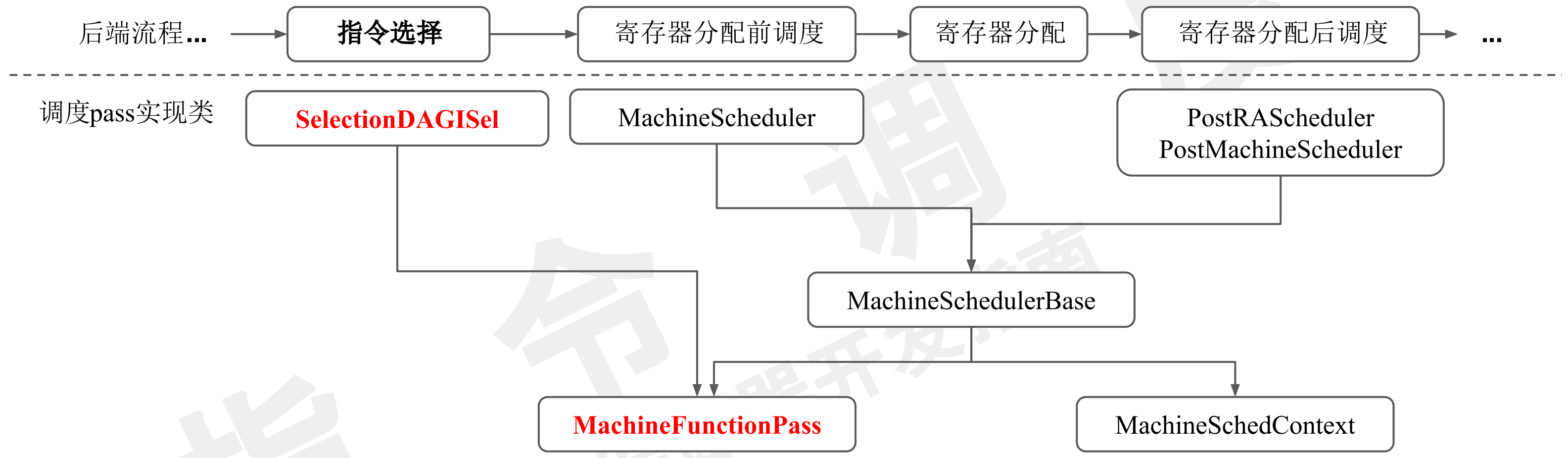


图4-13　调度 pass 类继承关系

指令选择阶段的调度器 - 3

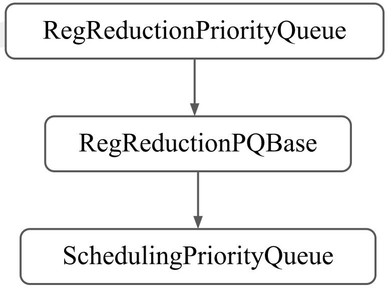
```
void SelectionDAGISel::CodeGenAndEmitDAG() {
...
// Run the DAG combiner in pre-legalize mode.
CurDAG->Combine(BeforeLegalizeTypes, AA, OptLevel);
CurDAG->LegalizeTypes();

// Run the DAG combiner in post-type-legalize mode.
CurDAG->Combine(AfterLegalizeTypes, AA, OptLevel);
CurDAG->LegalizeVectors();

CurDAG->LegalizeTypes();
CurDAG->Combine(AfterLegalizeVectorOps, AA, OptLevel);

CurDAG->Legalize();
CurDAG->Combine(AfterLegalizeDAG, AA, OptLevel);

DoInstructionSelection();
...
ScheduleDAGSDNodes *Scheduler = CreateScheduler();
...
Scheduler->Run(CurDAG, FuncInfo->MBB);
...
}
```

```
ScheduleDAGSDNodes *SelectionDAGISel::CreateScheduler() {
  return ISHeuristic(this, OptLevel);
}

static cl::opt<RegisterScheduler::FunctionPassCtor, false,
               RegisterPassParser<RegisterScheduler>>
ISHeuristic("pre-RA-sched",
               cl::init(&createDefaultScheduler), cl::Hidden,
               cl::desc("Instruction schedulers available (before register"
                        " allocation):"));
```

```
ScheduleDAGSDNodes* createDefaultScheduler(...) {
          ...
          if (OptLevel == CodeGenOpt::None ||
          (ST.enableMachineScheduler() && ST.enableMachineSchedDefaultSched()) ||
          TLI->getSchedulingPreference() == Sched::Source)
          return createSourceListDAGScheduler(IS, OptLevel);
          if (TLI->getSchedulingPreference() == Sched::RegPressure)
          return createBURRListDAGScheduler(IS, OptLevel);
          if (TLI->getSchedulingPreference() == Sched::Hybrid)
          return createHybridListDAGScheduler(IS, OptLevel);
          if (TLI->getSchedulingPreference() == Sched::VLIW)
          return createVLIWDAGScheduler(IS, OptLevel);
          assert(TLI->getSchedulingPreference() == Sched::ILP &&
          "Unknown sched type!");
          return createILPListDAGScheduler(IS, OptLevel);
}
```

指令选择阶段的调度器 - 4

- 调度器生成函数生成优先级队列
  - burrListDAGScheduler优先级队列 : BURegReductionPriorityQueue
  - sourceListDAGScheduler优先级队列 : SrcRegReductionPriorityQueue
  - hybridListDAGScheduler优先级队列 : HybridBURRPriorityQueue
  - ILPListDAGScheduler优先级队列 : ILPBURRPriorityQueue



```
ScheduleDAGSDNodes *
llvm::createBURRListDAGScheduler(SelectionDAGISel *IS, CodeGenOpt::Level OptLevel) {
...
  BURegReductionPriorityQueue *PQ = new BURegReductionPriorityQueue(*IS->MF, false, false, TII, TRI, nullptr);
  ScheduleDAGRRList *SD = new ScheduleDAGRRList(*IS->MF, false/*needlatency*/, PQ, OptLevel);
  PQ->setScheduleDAG(SD);
  return SD;
}

using BURegReductionPriorityQueue = RegReductionPriorityQueue<bu_ls_rr_sort>;
using SrcRegReductionPriorityQueue = RegReductionPriorityQueue<src_ls_rr_sort>;
using HybridBURRPriorityQueue = RegReductionPriorityQueue<hybrid_ls_rr_sort>;
using ILPBURRPriorityQueue = RegReductionPriorityQueue<ilp_ls_rr_sort>;
```

指令选择阶段的调度器 - 5

- 调度过程
  - 建立依赖图
  - 执行列表调度

```
void SelectionDAGISel::CodeGenAndEmitDAG() {
...
ScheduleDAGSDNodes *Scheduler = CreateScheduler();
...
Scheduler->Run(CurDAG, FuncInfo->MBB);
...
}

void ScheduleDAGSDNodes::Run(SelectionDAG *dag, MachineBasicBlock *bb) {
...
  Schedule();
}

void ScheduleDAGRRList::Schedule() {
...
  // Build the scheduling graph.
  BuildSchedGraph(nullptr);
...
  AvailableQueue->initNodes(SUnits);
  ListScheduleBottomUp();
  AvailableQueue->releaseState();
...
}
```

指令选择阶段的调度器 - 6

- 建立依赖图
  - 节点聚类
  - 生成SUnit节点
  - 添加调度边

```
void ScheduleDAGRRList::Schedule() {
...
  // Build the scheduling graph.
  BuildSchedGraph(nullptr);
...
  AvailableQueue->initNodes(SUnits);
  ListScheduleBottomUp();
  AvailableQueue->releaseState();
...
}
```

```
void ScheduleDAGSDNodes::BuildSchedGraph(AAResults *AA) {
  ClusterNodes();
  BuildSchedUnits();
  AddSchedEdges();
}
```

```
bool SIInstrInfo::areLoadsFromSameBasePtr(SDNode *Load0,
                                          SDNode *Load1,
                                          int64_t &Offset0,
                                          int64_t &Offset1) const {
  if (!Load0->isMachineOpcode() || !Load1->isMachineOpcode())
    return false;
…
}
```

```
bool SIInstrInfo::shouldScheduleLoadsNear(SDNode *Load0,
                                          SDNode *Load1,
                                          int64_t Offset0, int64_t Offset1,
                                          unsigned NumLoads) const {
  assert(Offset1 > Offset0 &&
         "Second offset should be larger than first offset!");

  return (NumLoads <= 16 && (Offset1 - Offset0) < 64);
}
```

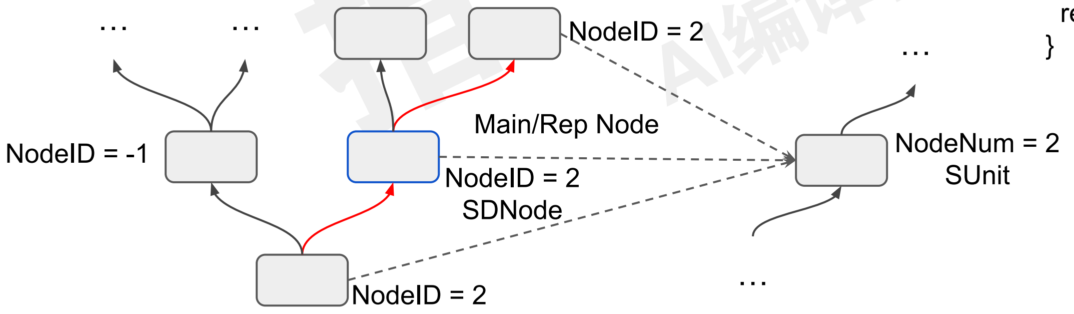
指令选择阶段的调度器 - 7

- 建立依赖图
  - 节点聚类
  - 生成SUnit节点
  - 添加调度边

```
void ScheduleDAGRRList::Schedule() {
...
  // Build the scheduling graph.
  BuildSchedGraph(nullptr);
...
  AvailableQueue->initNodes(SUnits);
  ListScheduleBottomUp();
  AvailableQueue->releaseState();
...
}
```

```
void ScheduleDAGSDNodes::BuildSchedGraph(AAResults *AA) {
  ClusterNodes();
  BuildSchedUnits();
  AddSchedEdges();
}
```

```
unsigned int R600InstrInfo::getInstrLatency(
                              const InstrItineraryData *ItinData,
                              const MachineInstr &,
                              unsigned *PredCost) const {

  if (PredCost)
    *PredCost = 2;
  return 2;
}
```



指令选择阶段的调度器 - 8

- 建立依赖图
  - 节点聚类
  - 生成SUnit节点
  - 添加调度边

```
void ScheduleDAGRRList::Schedule() {
...
  // Build the scheduling graph.
  BuildSchedGraph(nullptr);
...
  AvailableQueue->initNodes(SUnits);
  ListScheduleBottomUp();
  AvailableQueue->releaseState();
...
}
```

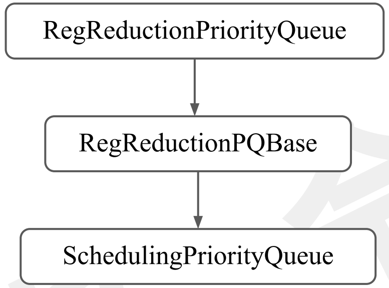
```
void ScheduleDAGSDNodes::BuildSchedGraph(AAResults *AA) {
  ClusterNodes();
  BuildSchedUnits();
  AddSchedEdges();
}
```

```
enum Kind {
  Data,
  Anti,
  Output,
  Order
};
```

```
enum OrderKind {
  Barrier,
  MayAliasMem,
  MustAliasMem,
  Artificial,
  Weak,
  Cluster
};
```

指令选择阶段的调度器 - x

- 优先级队列类的继承关系



Sort Function

RegPressure Queue

scheduledNode()
unscheduledNode()
tracksRegPressure()

initNodes
releaseState()

pop()
empty()
push()
remove()
…

寄存器分配前的MI调度器 - 0

-pre-RA-sched - Instruction schedulers available (before register allocation):
-enable-misched - Enable the machine instruction scheduling pass.
-enable-post-misched - Enable the post-ra machine instruction scheduling pass.

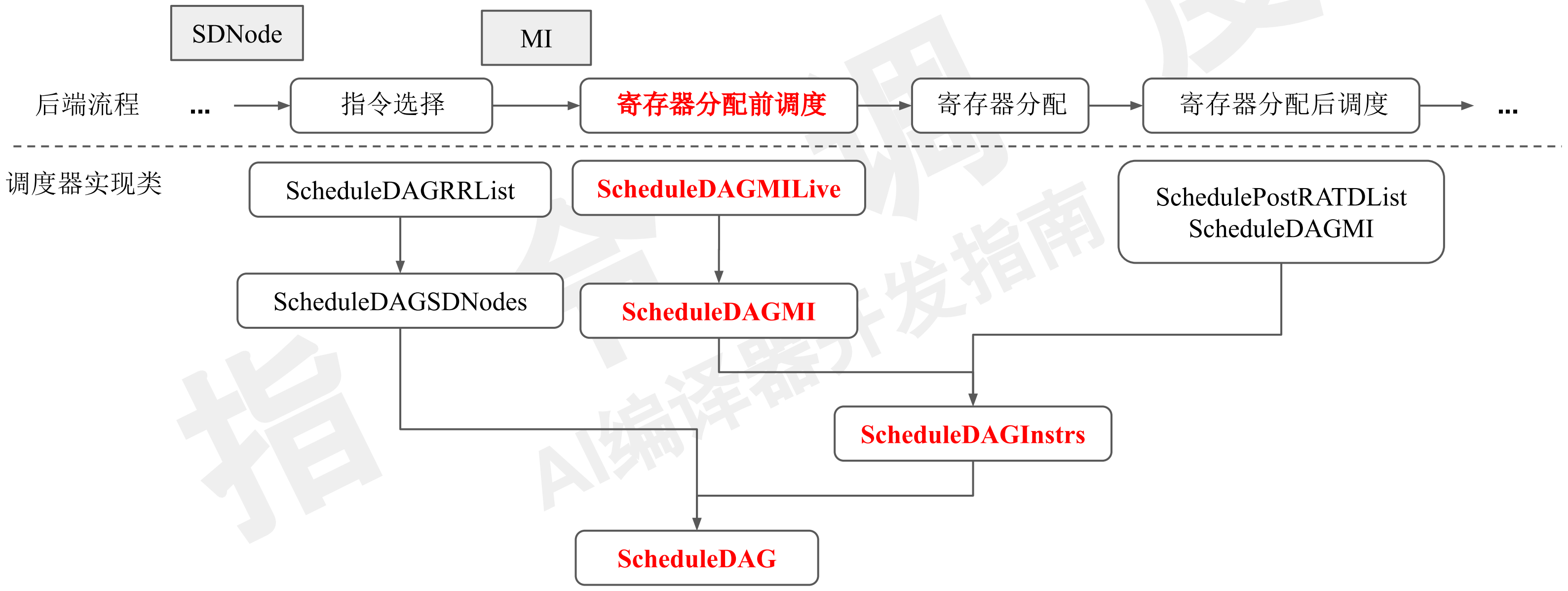


图4-14　调度器类继承关系

寄存器分配前的MI调度器 - 1

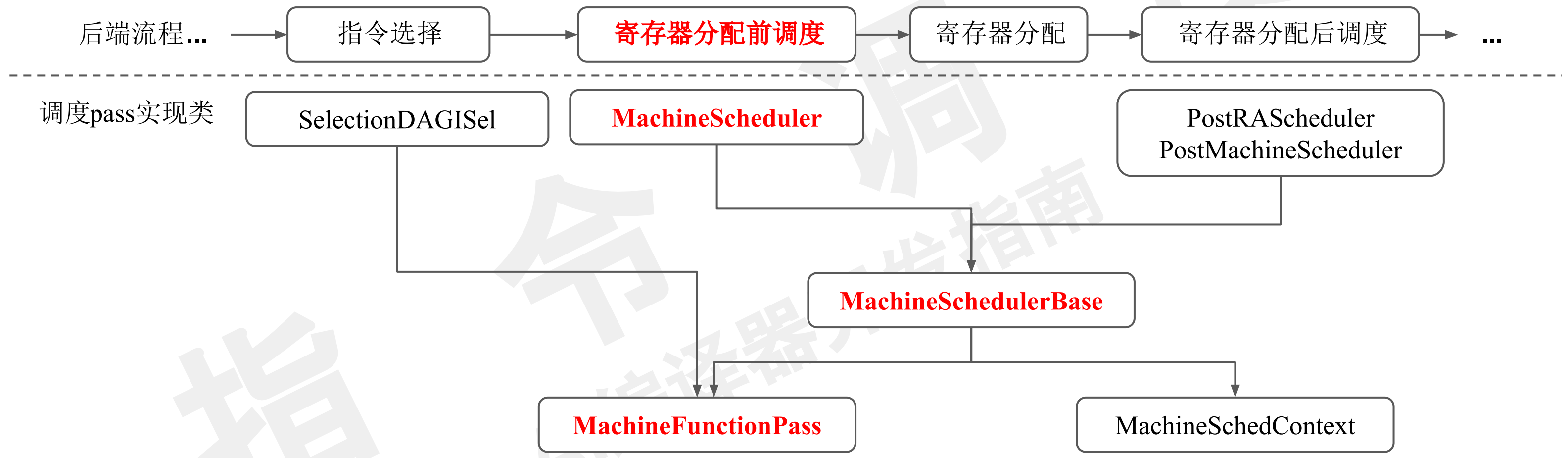


图4-13　调度 pass 类继承关系

寄存器分配前的MI调度器 - 2

```
bool MachineScheduler::runOnMachineFunction(MachineFunction &mf) {
  if (skipFunction(mf.getFunction()))
    return false;

  if (EnableMachineSched.getNumOccurrences()) {
    if (!EnableMachineSched)
      return false;
  } else if (!mf.getSubtarget().enableMachineScheduler())
    return false;

  // Initialize the context of the pass.
  MF = &mf;
  MLI = &getAnalysis<MachineLoopInfo>();
  MDT = &getAnalysis<MachineDominatorTree>();
  PassConfig = &getAnalysis<TargetPassConfig>();
  AA = &getAnalysis<AAResultsWrapperPass>().getAAResults();

  LIS = &getAnalysis<LiveIntervals>();
  RegClassInfo->runOnMachineFunction(*MF);

  // Instantiate the selected scheduler for this target, function, and optimization level.
  std::unique_ptr<ScheduleDAGInstrs> Scheduler(createMachineScheduler());
  scheduleRegions(*Scheduler, false /*FixKillFlags*/);

  return true;
}
```

```
ScheduleDAGInstrs *MachineScheduler::createMachineScheduler() {
  // Select the scheduler, or set the default.
  MachineSchedRegistry::ScheduleDAGCtor Ctor = MachineSchedOpt;
  if (Ctor != useDefaultMachineSched)
    return Ctor(this);

  // Get the default scheduler set by the target for this function.
  ScheduleDAGInstrs *Scheduler = PassConfig->createMachineScheduler(this);
  if (Scheduler)
    return Scheduler;

  // Default to GenericScheduler.
  return createGenericSchedLive(this);
}
```

```
ScheduleDAGInstrs *GCNPassConfig::createMachineScheduler(
  MachineSchedContext *C) const {
  const GCNSubtarget &ST = C->MF->getSubtarget<GCNSubtarget>();
  if (ST.enableSIScheduler())
    return createSIMachineScheduler(C);
  return createGCNMaxOccupancyMachineScheduler(C);
}
```

寄存器分配前的MI调度器 - 3

```
void MachineSchedulerBase::scheduleRegions(ScheduleDAGInstrs &Scheduler,
                                           bool FixKillFlags) {
  for (MachineFunction::iterator MBB = MF->begin(), MBBEnd = MF->end();  MBB != MBBEnd; ++MBB) {
    Scheduler.startBlock(&*MBB);
    MBBRegionsVector MBBRegions;
    getSchedRegions(&*MBB, MBBRegions, Scheduler.doMBBSchedRegionsTopDown());
    for (const SchedRegion &R : MBBRegions) {
      MachineBasicBlock::iterator I = R.RegionBegin;
      MachineBasicBlock::iterator RegionEnd = R.RegionEnd;
      unsigned NumRegionInstrs = R.NumRegionInstrs;

      Scheduler.enterRegion(&*MBB, I, RegionEnd, NumRegionInstrs);
      …
      Scheduler.schedule();
      Scheduler.exitRegion();
    }
    …
}
```

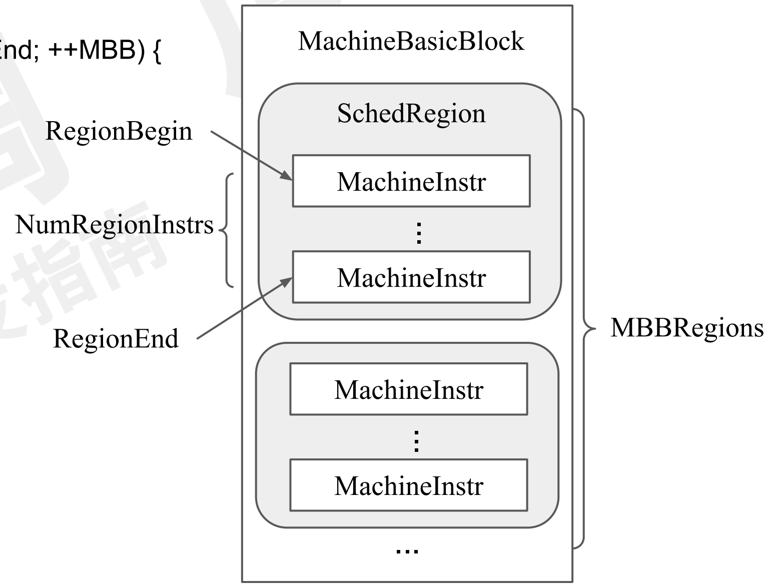


图4-11 调度区域结构

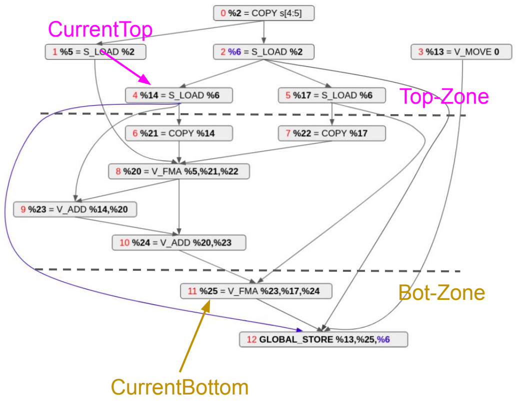
Scheduler.schedule()函数可由后端定制

- AMDGPU GCN子目标实现GCNScheduleDAGMILive::schedule()
- AMDGPU SI子目标实现SIScheduleDAGMI::schedule()

寄存器分配前的MI调度器 - 调度代码框架

- 调度区域划分为三个区域：顶部调度区域（Top-Zone）、底部调度区域（Bottom-Zone）和未调度区域。
- buildDAGWithRegPressure
- postprocessDAG
- findRootsAndBiasEdges
- initQueues
- pickNode
- scheduleMI
- schedNode
- updateQueues

```
void ScheduleDAGMILive::schedule() {
  buildDAGWithRegPressure();
  postprocessDAG();
  findRootsAndBiasEdges(TopRoots, BotRoots)
  initQueues(TopRoots, BotRoots);
  while (true) {
    SUnit *SU = SchedImpl->pickNode(IsTopNoc
    if (!SU) break;
    scheduleMI(SU, IsTopNode);
    SchedImpl->schedNode(SU, IsTopNode);
    updateQueues(SU, IsTopNode);
  }
}

GCNMaxOccupancySchedStrategy::pickNode(bool &IsTopNode) {
  SU = pickNodeBidirectional(IsTopNode);
}

GCNMaxOccupancySchedStrategy::pickNodeBidirectional(bool &IsTopNode) {
  pickNodeFromQueue(Bot, ... , BotCand);
  pickNodeFromQueue(Top, ... , TopCand);
  tryCandidate(BotCand, TopCand, nullptr);
}

GCNMaxOccupancySchedStrategy::pickNodeFromQueue(..., SchedCandidate &Cand) {
  for (SUnit *SU : Q) {
    SchedCandidate TryCand;
    initCandidate();
    GenericScheduler::tryCandidate(Cand, TryCand, ZoneArg);
    Cand.setBest(TryCand);
  }
}
```



寄存器分配前的MI调度器 - 举例

```
llc -march=amdgcn test_amd.ll -debug-only=machine-scheduler -filetype=asm -o -
```

```
__kernel void test(float a, float *x, float *y)
{
  float b = a*x[0] + y[0];
  float c = b + x[1];
  float d = c * y[1];
  y[2] = b + c + d;
}
```

```
s_load_dword s6, s[4:5], 0x0
s_load_dwordx4 s[0:3], s[4:5], 0x8
v_mov_b32_e32 v0, 0
s_mov_b32 s33, 0
s_waitcnt lgkmcnt(0)
s_load_dwordx2 s[0:1], s[0:1], 0x0
s_load_dwordx2 s[4:5], s[2:3], 0x0
s_waitcnt lgkmcnt(0)
v_mov_b32_e32 v1, s0
v_mov_b32_e32 v2, s4
v_fma_f32 v1, s6, v1, v2
v_add_f32_e32 v2, s1, v1
v_add_f32_e32 v1, v1, v2
v_fma_f32 v1, v2, s5, v1
global_store_dword v0, v1, s[2:3] offset:8
s_endpgm
```

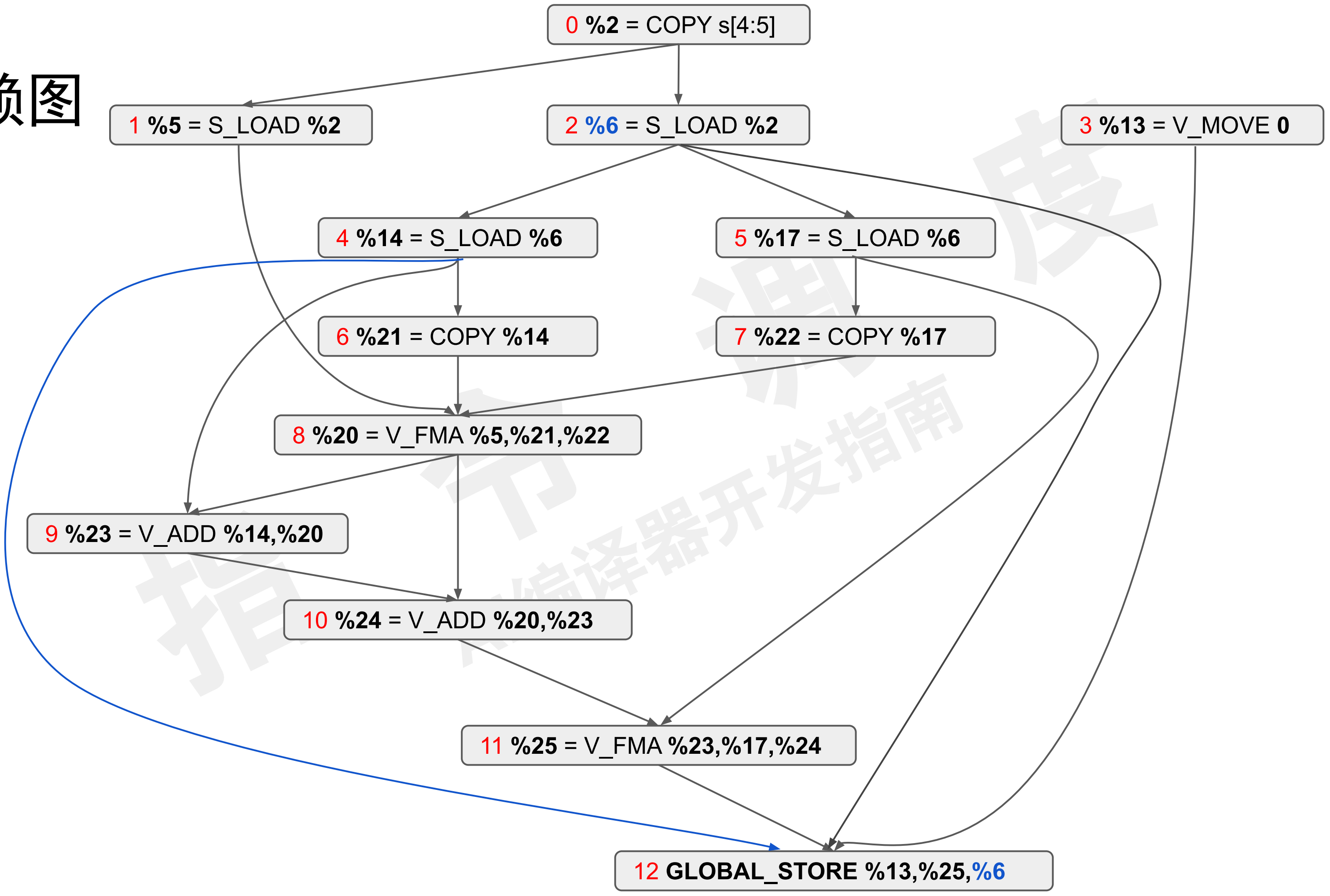
```
%4 = tail call nonnull align 8 dereferenceable(24) i8 addrspace(4)*
        @llvm.amdgcn.kernarg.segment.ptr()
%5 = bitcast i8 addrspace(4)* %4 to float addrspace(4)*
%6 = load float, float addrspace(4)* %5, align 8, !invariant.load !14
%7 = getelementptr inbounds i8, i8 addrspace(4)* %4, i64 8
%8 = bitcast i8 addrspace(4)* %7 to float* addrspace(4)*
%9 = load float*, float* addrspace(4)* %8, align 8, !invariant.load !14
%10 = getelementptr inbounds i8, i8 addrspace(4)* %4, i64 16
%11 = bitcast i8 addrspace(4)* %10 to float* addrspace(4)*
%12 = load float*, float* addrspace(4)* %11, align 8, !invariant.load !14
%13 = load float, float* %9, align 4, !tbaa !15
%14 = fmul contract float %6, %13
%15 = load float, float* %12, align 4, !tbaa !15
%16 = fadd contract float %14, %15
%17 = getelementptr inbounds float, float* %9, i64 1
%18 = load float, float* %17, align 4, !tbaa !15
%19 = fadd contract float %16, %18
%20 = getelementptr inbounds float, float* %12, i64 1
%21 = load float, float* %20, align 4, !tbaa !15
%22 = fmul contract float %19, %21
%23 = fadd contract float %16, %19
%24 = fadd contract float %23, %22
%25 = getelementptr inbounds float, float* %12, i64 2
store float %24, float* %25, align 4, !tbaa !15
ret void
```

寄存器分配前的MI调度器 - MI

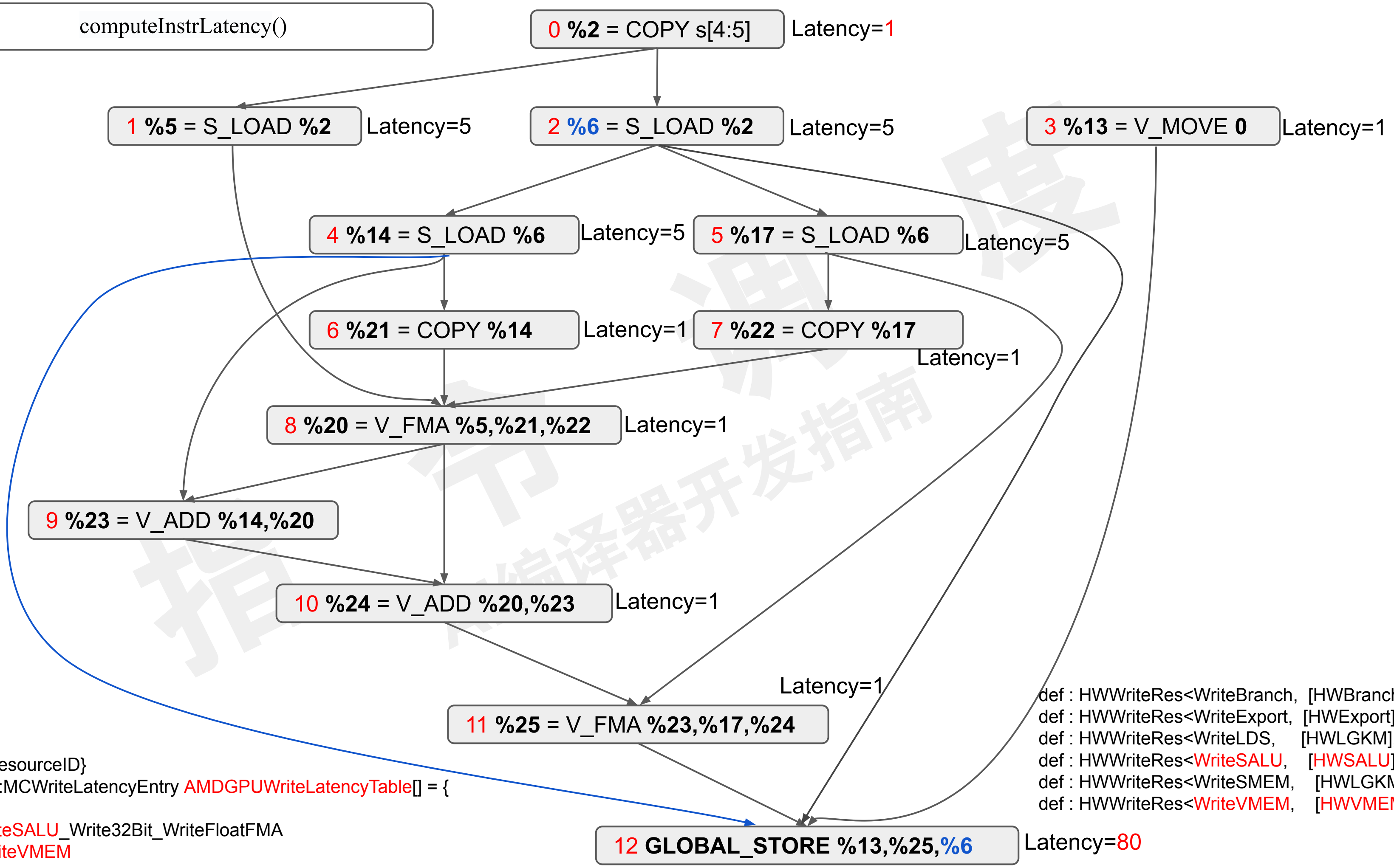
```
__kernel void test(float a, float *x, float *y)
{
  float b = a*x[0] + y[0];
  float c = b + x[1];
  float d = c * y[1];
  y[2] = b + c + d;
}
```

```
bb.0 (%ir-block.3):
liveins: $sgpr4_sgpr5
%2:sgpr_64(p4) = COPY $sgpr4_sgpr5
%5:sreg_32_xm0_xexec = S_LOAD_DWORD_IMM %2:sgpr_64(p4), 0, 0, 0 :: (dereferenceable invariant load 4 from %ir.5, align 8, addrspace 4)
%6:sgpr_128 = S_LOAD_DWORDX4_IMM %2:sgpr_64(p4), 8, 0, 0 :: (dereferenceable invariant load 16 from %ir.9, align 8, addrspace 4)
%13:vgpr_32 = V_MOV_B32_e32 0, implicit $exec
%14:sreg_64_xexec = S_LOAD_DWORDX2_IMM %6.sub0_sub1:sgpr_128, 0, 0, 0 :: (load 8 from %ir.19, align 4, !tbaa !15, addrspace 1)
%17:sreg_64_xexec = S_LOAD_DWORDX2_IMM %6.sub2_sub3:sgpr_128, 0, 0, 0 :: (load 8 from %ir.24, align 4, !tbaa !15, addrspace 1)
%21:vgpr_32 = COPY %14.sub0:sreg_64_xexec
%22:vgpr_32 = COPY %17.sub0:sreg_64_xexec
%20:vgpr_32 = contract nofpexcept V_FMA_F32 0, %5:sreg_32_xm0_xexec, 0, %21:vgpr_32, 0, %22:vgpr_32, 0, 0, implicit $mode, implicit $exec
%23:vgpr_32 = contract nofpexcept V_ADD_F32_e32 %14.sub1:sreg_64_xexec, %20:vgpr_32, implicit $mode, implicit $exec
%24:vgpr_32 = contract nofpexcept V_ADD_F32_e32 %20:vgpr_32, %23:vgpr_32, implicit $mode, implicit $exec
%25:vgpr_32 = contract nofpexcept V_FMA_F32 0, %23:vgpr_32, 0, %17.sub1:sreg_64_xexec, 0, %24:vgpr_32, 0, 0, implicit $mode, implicit $exec
GLOBAL_STORE_DWORD_SADDR %13:vgpr_32, %25:vgpr_32, %6.sub2_sub3:sgpr_128, 8, 0, 0, 0, implicit $exec :: (store 4 into %ir.33, !tbaa
!15, addrspace 1)
S_ENDPGM 0
```

依赖图


0 %2 = COPY s[4:5]
1 %5 = S_LOAD %2
2 %6 = S_LOAD %2
3 %13 = V_MOVE 0
4 %14 = S_LOAD %6
5 %17 = S_LOAD %6
6 %21 = COPY %14
7 %22 = COPY %17
8 %20 = V_FMA %5,%21,%22
9 %23 = V_ADD %14,%20
10 %24 = V_ADD %20,%23
11 %25 = V_FMA %23,%17,%24
12 GLOBAL_STORE %13,%25,%6

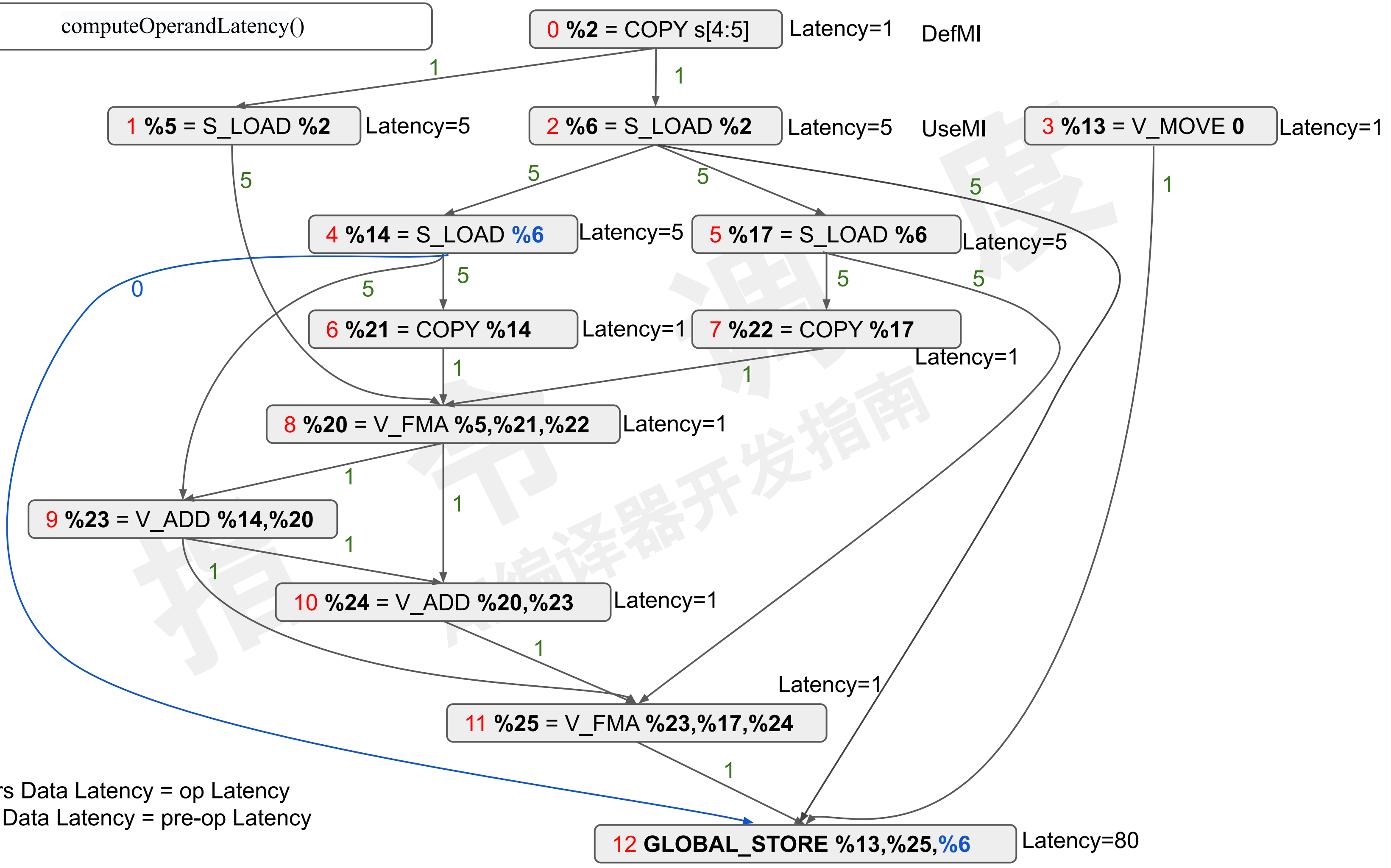
computeInstrLatency()



```
// {Cycles, WriteResourceID}
extern const llvm::MCWriteLatencyEntry AMDGPUWriteLatencyTable[] = {
  { 0,  0}, // Invalid
  { 1,  0}, // #1 WriteSALU_Write32Bit_WriteFloatFMA
  {80,  0}, // #2 WriteVMEM
...
```

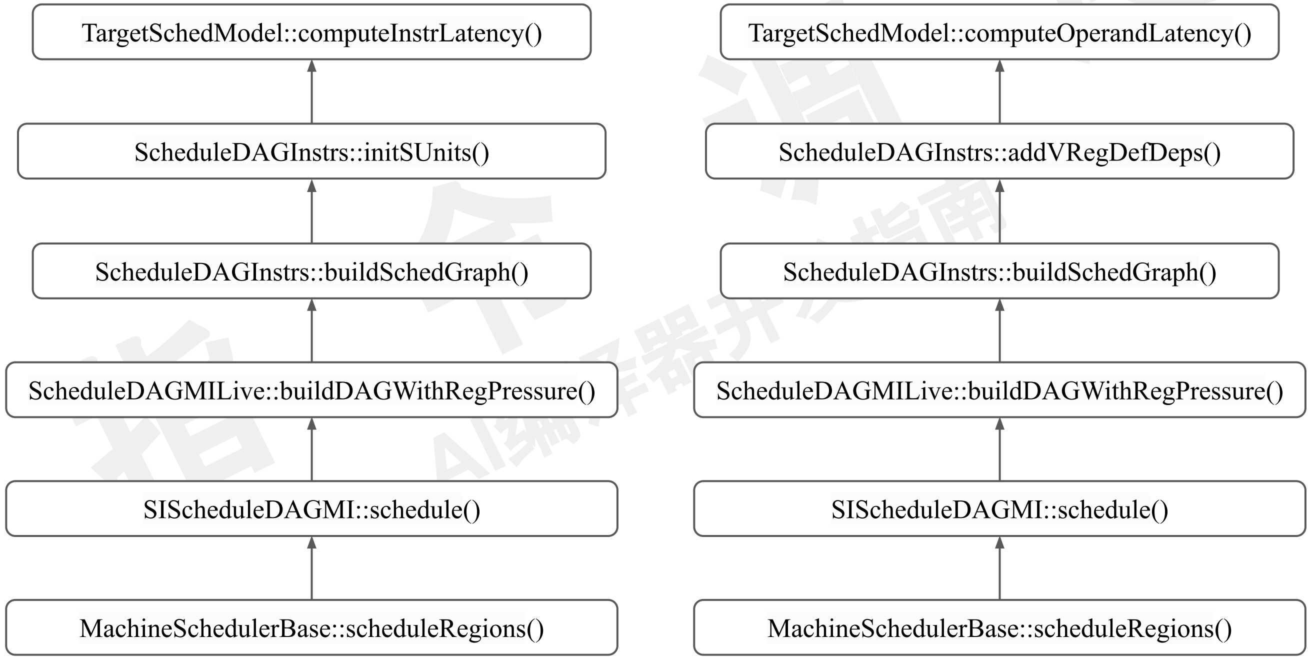
```
def : HWWriteRes<WriteBranch,  [HWBranch], 8>;
def : HWWriteRes<WriteExport,  [HWExport], 4>;
def : HWWriteRes<WriteLDS,     [HWLGKM],   5>;
def : HWWriteRes<WriteSALU,    [HWSALU],   1>;
def : HWWriteRes<WriteSMEM,    [HWLGKM],   5>;
def : HWWriteRes<WriteVMEM,    [HWVMEM],   80>;
```

computeOperandLatency()

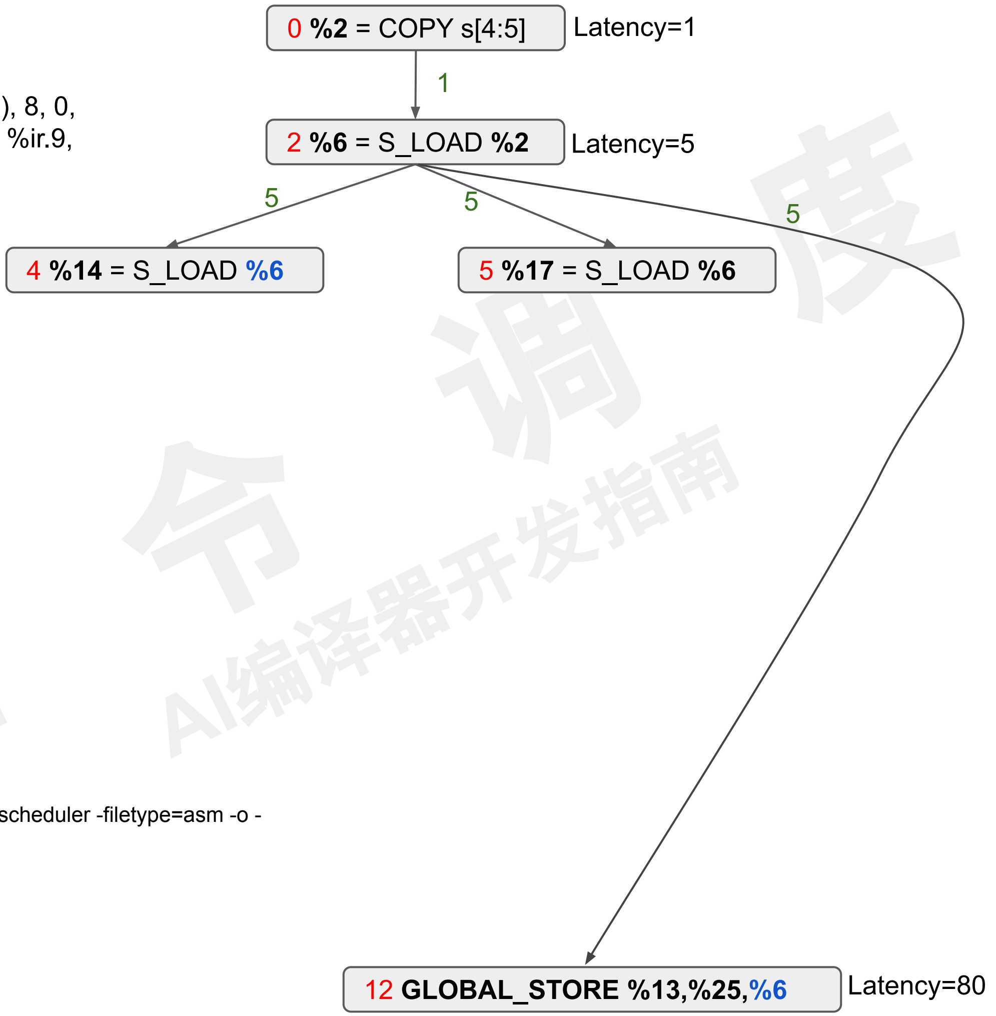


Successors Data Latency = op Latency
Predecessors Data Latency = pre-op Latency

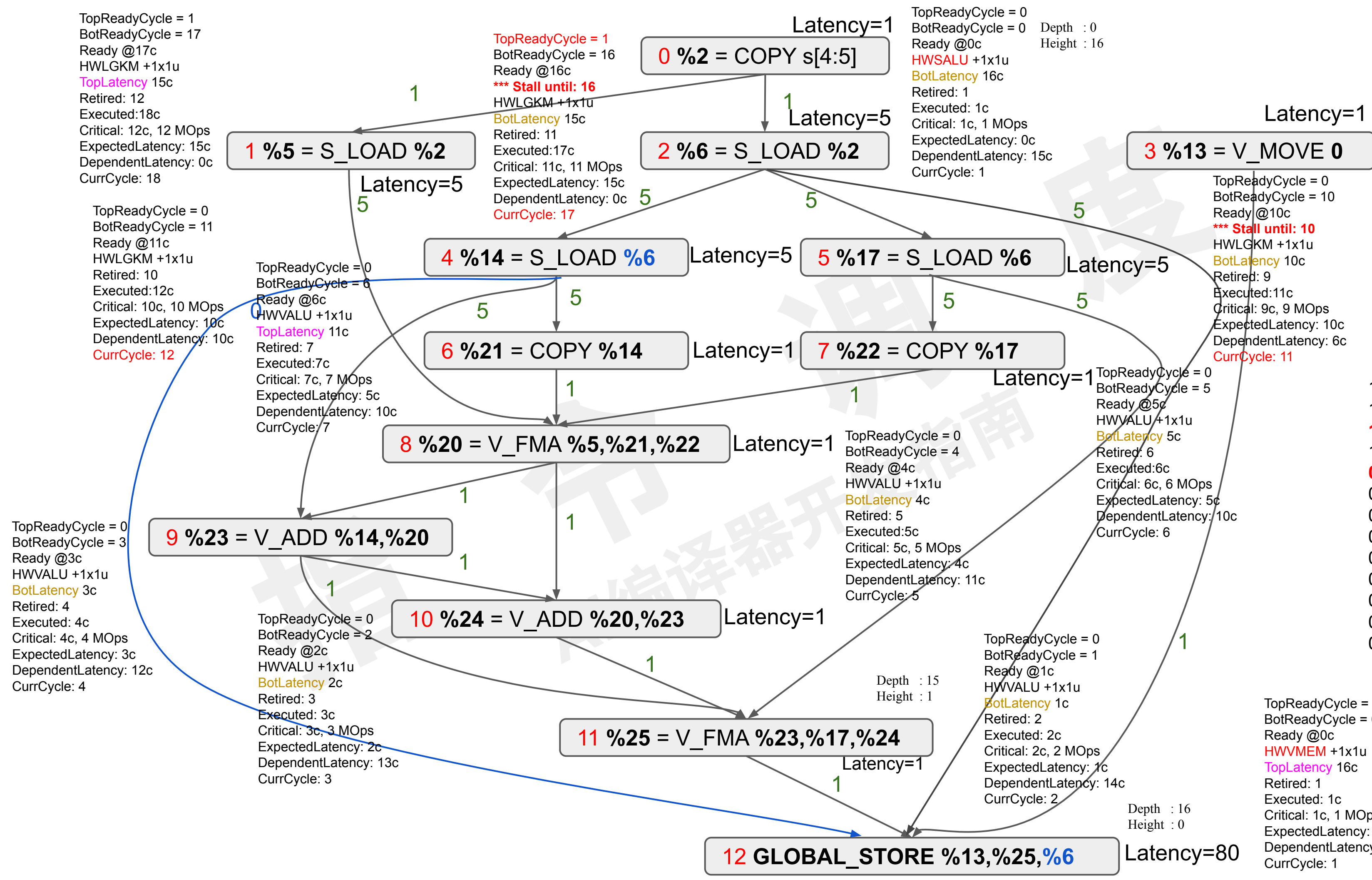
寄存器分配前的MI调度器 - 延迟计算函数调用栈


TargetSchedModel::computeInstrLatency()
ScheduleDAGInstrs::initSUnits()
ScheduleDAGInstrs::buildSchedGraph()
ScheduleDAGMILive::buildDAGWithRegPressure()
SIScheduleDAGMI::schedule()
MachineSchedulerBase::scheduleRegions()
TargetSchedModel::computeOperandLatency()
ScheduleDAGInstrs::addVRegDefDeps()
ScheduleDAGInstrs::buildSchedGraph()
ScheduleDAGMILive::buildDAGWithRegPressure()
SIScheduleDAGMI::schedule()
MachineSchedulerBase::scheduleRegions()

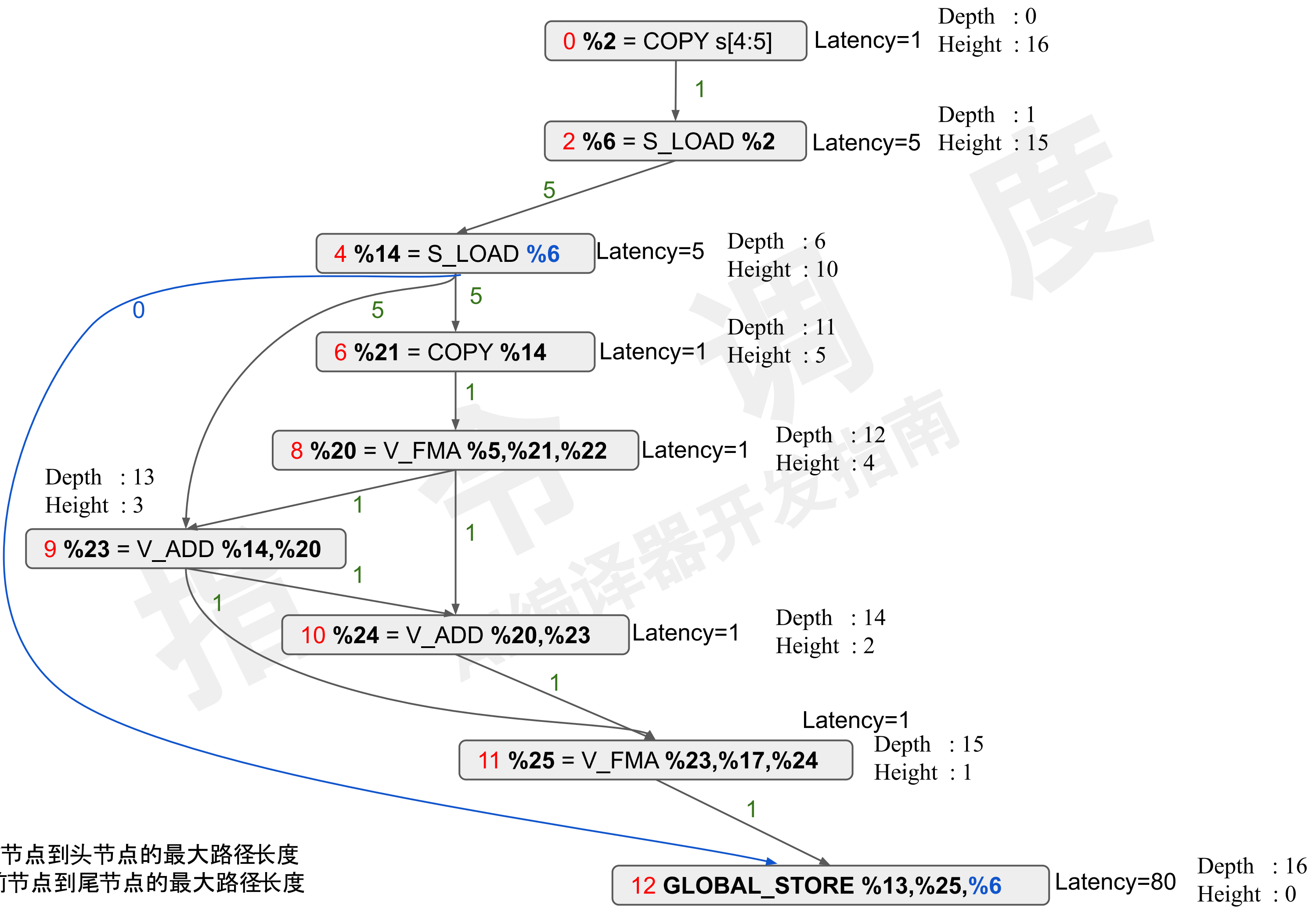
```
SU(2):   %6:sgpr_128 =
S_LOAD_DWORDX4_IMM %2:sgpr_64(p4), 8, 0,
0 :: (dereferenceable invariant load 16 from %ir.9,
align 8, addrspace 4)
  # preds left       : 1
  # succs left       : 3
  # rdefs left      : 0
  Latency           : 5
  Depth              : 1
  Height             : 15
  Predecessors:
    SU(0): Data Latency=1 Reg=%2
  Successors:
    SU(12): Data Latency=5 Reg=%6
    SU(5): Data Latency=5 Reg=%6
    SU(4): Data Latency=5 Reg=%6
  Pressure Diff      : SReg_32 -2
  Single Issue       : false;
```



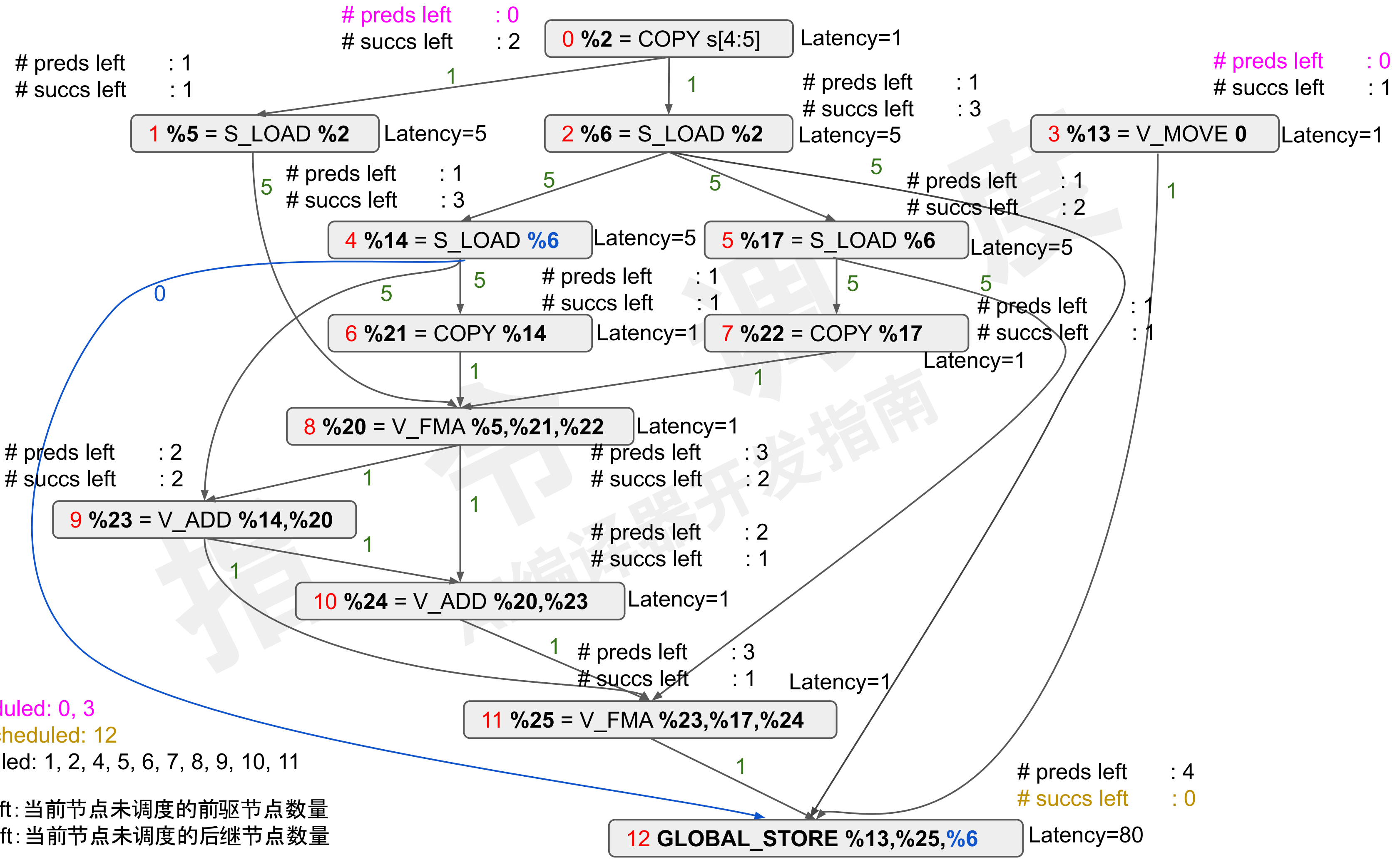
```
llc -march=amdgcn test_amd.ll -debug-only=machine-scheduler -filetype=asm -o -
```



17 Scheduling SU(1)
16 Scheduling SU(2)
11 Scheduling SU(4)
10 Scheduling SU(5)
07 Scheduling SU(3)
06 Scheduling SU(6)
05 Scheduling SU(7)
04 Scheduling SU(8)
03 Scheduling SU(9)
02 Scheduling SU(10)
01 Scheduling SU(11)
00 Scheduling SU(0)
00 Scheduling SU(12)

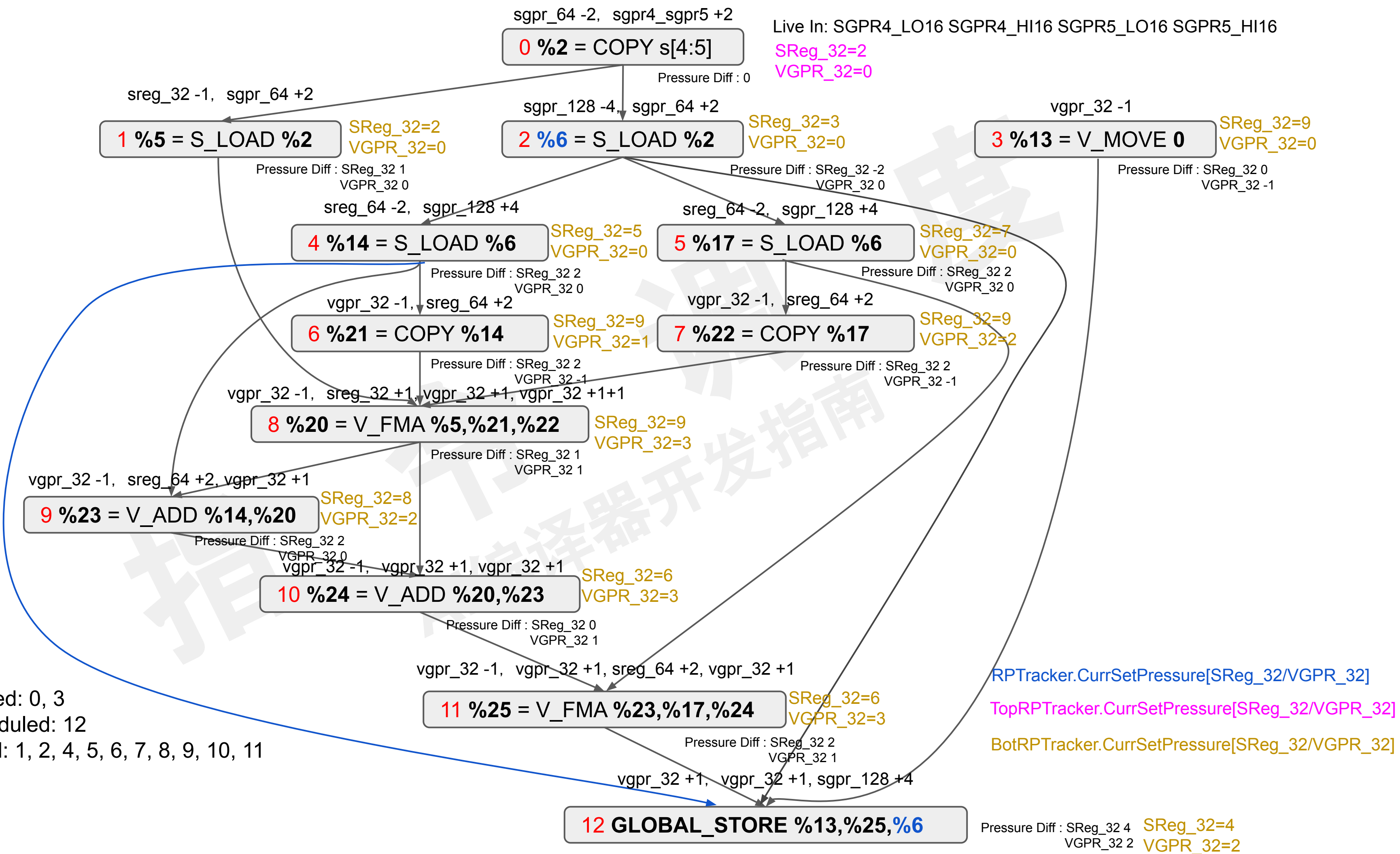


# Depth: 当前节点到头节点的最大路径长度
# Height: 当前节点到尾节点的最大路径长度

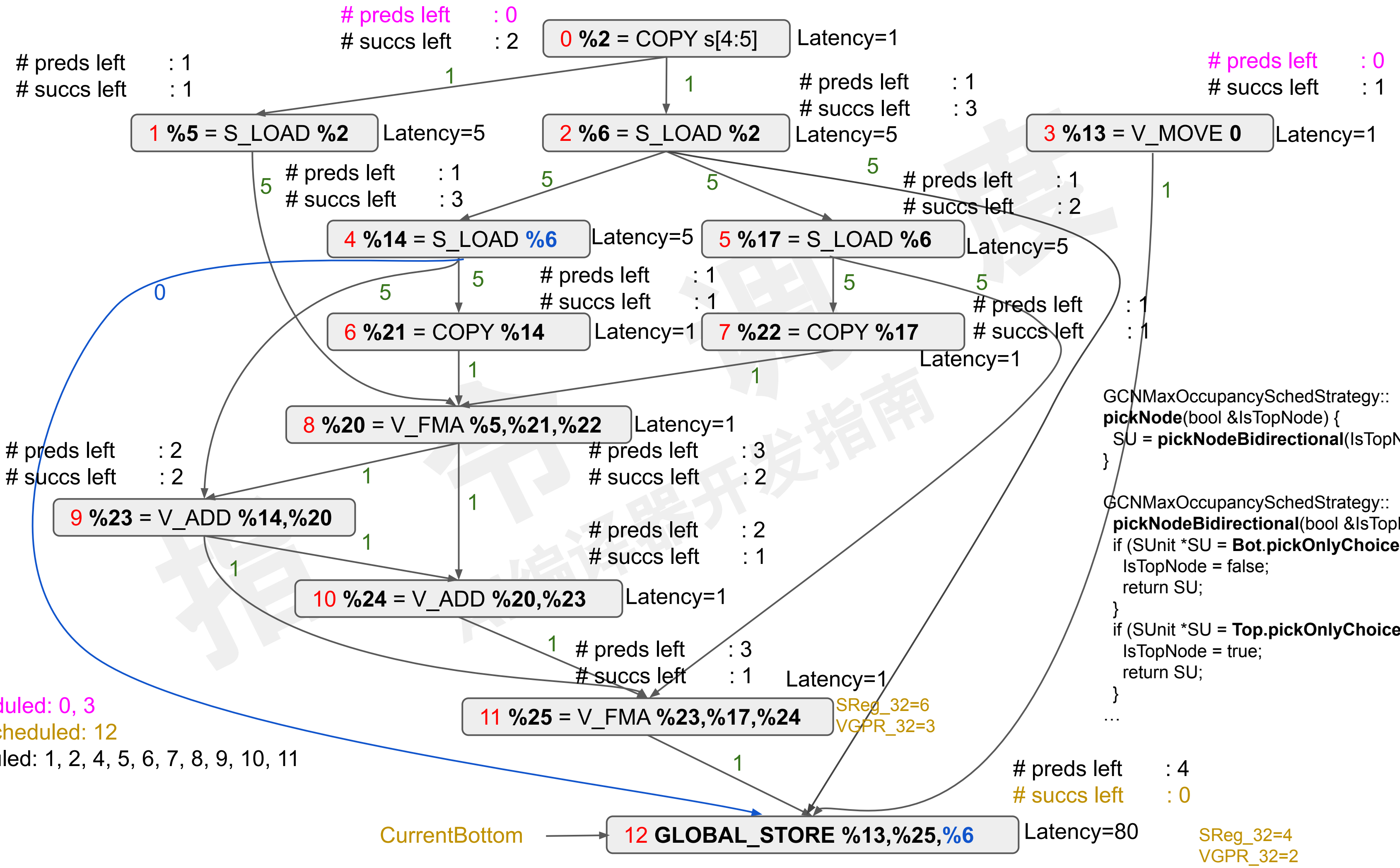


Top-scheduled: 0, 3
Bottom-scheduled: 12
Unscheduled: 1, 2, 4, 5, 6, 7, 8, 9, 10, 11

# preds left: 当前节点未调度的前驱节点数量
# succs left: 当前节点未调度的后继节点数量

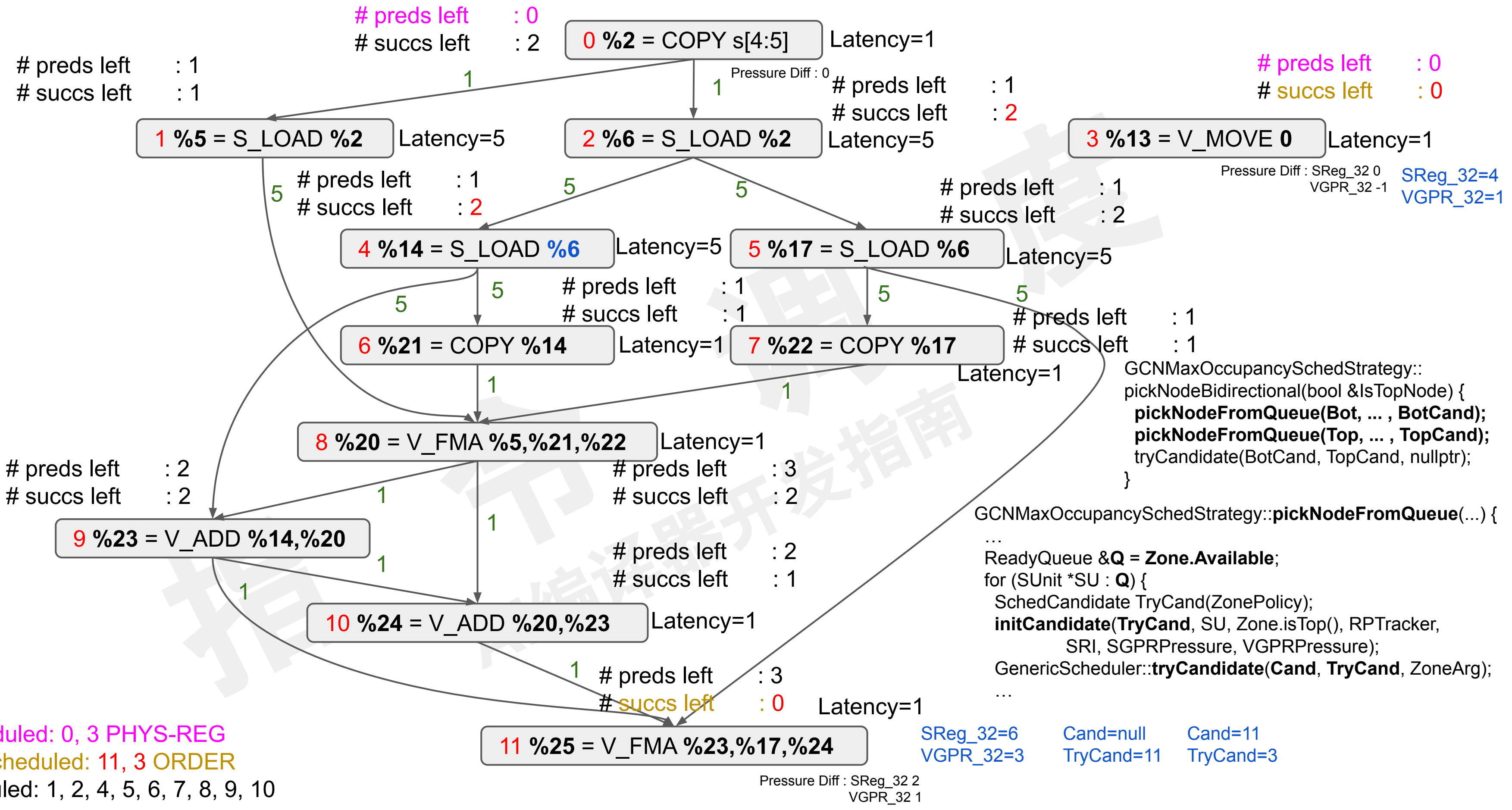

sgpr_64 -2, sgpr4_sgpr5 +2
0 %2 = COPY s[4:5]
Pressure Diff : 0
Live In: SGPR4_LO16 SGPR4_HI16 SGPR5_LO16 SGPR5_HI16
SReg_32=2
VGPR_32=0
sreg_32 -1, sgpr_64 +2
1 %5 = S_LOAD %2
SReg_32=2
VGPR_32=0
Pressure Diff : SReg_32 1
VGPR_32 0
sgpr_128 -4, sgpr_64 +2
2 %6 = S_LOAD %2
SReg_32=3
VGPR_32=0
Pressure Diff : SReg_32 -2
VGPR_32 0
vgpr_32 -1
3 %13 = V_MOVE 0
SReg_32=9
VGPR_32=0
Pressure Diff : SReg_32 0
VGPR_32 -1
sreg_64 -2, sgpr_128 +4
4 %14 = S_LOAD %6
SReg_32=5
VGPR_32=0
Pressure Diff : SReg_32 2
VGPR_32 0
sreg_64 -2, sgpr_128 +4
5 %17 = S_LOAD %6
SReg_32=7
VGPR_32=0
Pressure Diff : SReg_32 2
VGPR_32 0
vgpr_32 -1, sreg_64 +2
6 %21 = COPY %14
SReg_32=9
VGPR_32=1
Pressure Diff : SReg_32 2
VGPR_32 -1
vgpr_32 -1, sreg_64 +2
7 %22 = COPY %17
SReg_32=9
VGPR_32=2
Pressure Diff : SReg_32 2
VGPR_32 -1
vgpr_32 -1, sreg_32 +1, vgpr_32 +1, vgpr_32 +1+1
8 %20 = V_FMA %5,%21,%22
SReg_32=9
VGPR_32=3
Pressure Diff : SReg_32 1
VGPR_32 1
vgpr_32 -1, sreg_64 +2, vgpr_32 +1
9 %23 = V_ADD %14,%20
SReg_32=8
VGPR_32=2
Pressure Diff : SReg_32 2
VGPR_32 0
vgpr_32 -1, vgpr_32 +1, vgpr_32 +1
10 %24 = V_ADD %20,%23
SReg_32=6
VGPR_32=3
Pressure Diff : SReg_32 0
VGPR_32 1
vgpr_32 -1, vgpr_32 +1, sreg_64 +2, vgpr_32 +1
11 %25 = V_FMA %23,%17,%24
SReg_32=6
VGPR_32=3
Pressure Diff : SReg_32 2
VGPR_32 1
vgpr_32 +1, vgpr_32 +1, sgpr_128 +4
12 GLOBAL_STORE %13,%25,%6
Pressure Diff : SReg_32 4
VGPR_32 2
SReg_32=4
VGPR_32=2
RPTracker.CurrSetPressure[SReg_32/VGPR_32]
TopRPTracker.CurrSetPressure[SReg_32/VGPR_32]
BotRPTracker.CurrSetPressure[SReg_32/VGPR_32]

Top-scheduled: 0, 3
Bottom-scheduled: 12
Unscheduled: 1, 2, 4, 5, 6, 7, 8, 9, 10, 11



```
GCNMaxOccupancySchedStrategy::
pickNode(bool &IsTopNode) {
  SU = pickNodeBidirectional(IsTopNode);
}

GCNMaxOccupancySchedStrategy::
 pickNodeBidirectional(bool &IsTopNode) {
 if (SUnit *SU = Bot.pickOnlyChoice()) {
   IsTopNode = false;
   return SU;
 }
 if (SUnit *SU = Top.pickOnlyChoice()) {
   IsTopNode = true;
   return SU;
 }
…
```

Top-scheduled: 0, 3
Bottom-scheduled: 12
Unscheduled: 1, 2, 4, 5, 6, 7, 8, 9, 10, 11



```
GCNMaxOccupancySchedStrategy::
pickNodeBidirectional(bool &IsTopNode) {
  pickNodeFromQueue(Bot, ... , BotCand);
  pickNodeFromQueue(Top, ... , TopCand);
  tryCandidate(BotCand, TopCand, nullptr);
}
```

```
GCNMaxOccupancySchedStrategy::pickNodeFromQueue(...) {
  …
  ReadyQueue &Q = Zone.Available;
  for (SUnit *SU : Q) {
    SchedCandidate TryCand(ZonePolicy);
    initCandidate(TryCand, SU, Zone.isTop(), RPTracker,
                  SRI, SGPRPressure, VGPRPressure);
    GenericScheduler::tryCandidate(Cand, TryCand, ZoneArg);
    …
```

Top-scheduled: 0, 3 PHYS-REG
Bottom-scheduled: 11, 3 ORDER
Unscheduled: 1, 2, 4, 5, 6, 7, 8, 9, 10

RPTracker.CurrSetPressure =
SReg_32=4, VGPR_32=2

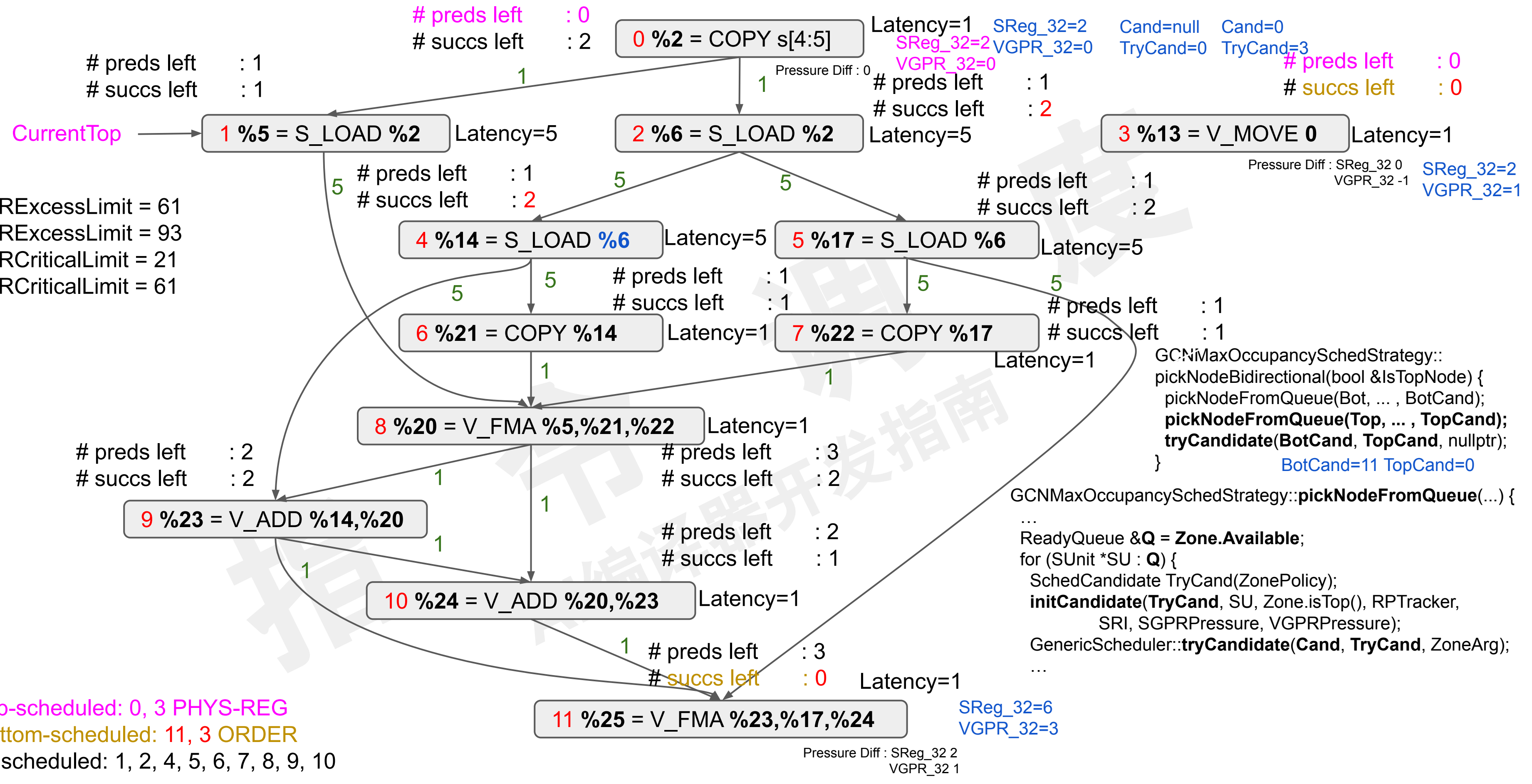
Scheduling SU(12) GLOBAL_STORE %13, %25, %6

```
void GCNMaxOccupancySchedStrategy::initCandidate(SchedCandidate &Cand, SUnit *SU, bool AtTop, const RegPressureTracker &RPTracker,
                                   const SIRegisterInfo *SRI, unsigned SGPRPressure, unsigned VGPRPressure) {
...

  if (AtTop)  TempTracker.getDownwardPressure(SU->getInstr(), Pressure, MaxPressure);
  else TempTracker.getUpwardPressure(SU->getInstr(), Pressure, MaxPressure);

  unsigned NewSGPRPressure = Pressure[AMDGPU::RegisterPressureSets::SReg_32];
  unsigned NewVGPRPressure = Pressure[AMDGPU::RegisterPressureSets::VGPR_32];

  // FIXME: Better heuristics to determine whether to prefer SGPRs or VGPRs.
  const unsigned MaxVGPRPressureInc = 16;
  bool ShouldTrackVGPRs = VGPRPressure + MaxVGPRPressureInc >= VGPRExcessLimit;
  bool ShouldTrackSGPRs = !ShouldTrackVGPRs && SGPRPressure >= SGPRExcessLimit;

  if (ShouldTrackVGPRs && NewVGPRPressure >= VGPRExcessLimit) {
    Cand.RPDelta.Excess = PressureChange(AMDGPU::RegisterPressureSets::VGPR_32);
    Cand.RPDelta.Excess.setUnitInc(NewVGPRPressure - VGPRExcessLimit);
  }

  if (ShouldTrackSGPRs && NewSGPRPressure >= SGPRExcessLimit) {
    Cand.RPDelta.Excess = PressureChange(AMDGPU::RegisterPressureSets::SReg_32);
    Cand.RPDelta.Excess.setUnitInc(NewSGPRPressure - SGPRExcessLimit);
  }

  int SGPRDelta = NewSGPRPressure - SGPRCriticalLimit;
  int VGPRDelta = NewVGPRPressure - VGPRCriticalLimit;
  if (SGPRDelta >= 0 || VGPRDelta >= 0) {
    if (SGPRDelta > VGPRDelta) {
      Cand.RPDelta.CriticalMax =
        PressureChange(AMDGPU::RegisterPressureSets::SReg_32);
      Cand.RPDelta.CriticalMax.setUnitInc(SGPRDelta);
    } else {
      Cand.RPDelta.CriticalMax =
        PressureChange(AMDGPU::RegisterPressureSets::VGPR_32);
      Cand.RPDelta.CriticalMax.setUnitInc(VGPRDelta);
    }
  }
}
```

SReg_32=6
VGPR_32=3

VGPRExcessLimit = 61
SGPRExcessLimit = 93
VGPRCriticalLimit = 21
SGPRCriticalLimit = 61

VGPRExcessLimit = 61
SGPRExcessLimit = 93
VGPRCriticalLimit = 21
SGPRCriticalLimit = 61



```
GCNMaxOccupancySchedStrategy::
pickNodeBidirectional(bool &IsTopNode) {
  pickNodeFromQueue(Bot, ... , BotCand);
  pickNodeFromQueue(Top, ... , TopCand);
  tryCandidate(BotCand, TopCand, nullptr);
}
```

BotCand=11 TopCand=0

```
GCNMaxOccupancySchedStrategy::pickNodeFromQueue(...) {
  ...
  ReadyQueue &Q = Zone.Available;
  for (SUnit *SU : Q) {
    SchedCandidate TryCand(ZonePolicy);
    initCandidate(TryCand, SU, Zone.isTop(), RPTracker,
                  SRI, SGPRPressure, VGPRPressure);
    GenericScheduler::tryCandidate(Cand, TryCand, ZoneArg);
    ...
```

Top-scheduled: 0, 3 PHYS-REG
Bottom-scheduled: 11, 3 ORDER
Unscheduled: 1, 2, 4, 5, 6, 7, 8, 9, 10

RPTracker.CurrSetPressure =
SReg_32=2, VGPR_32=0

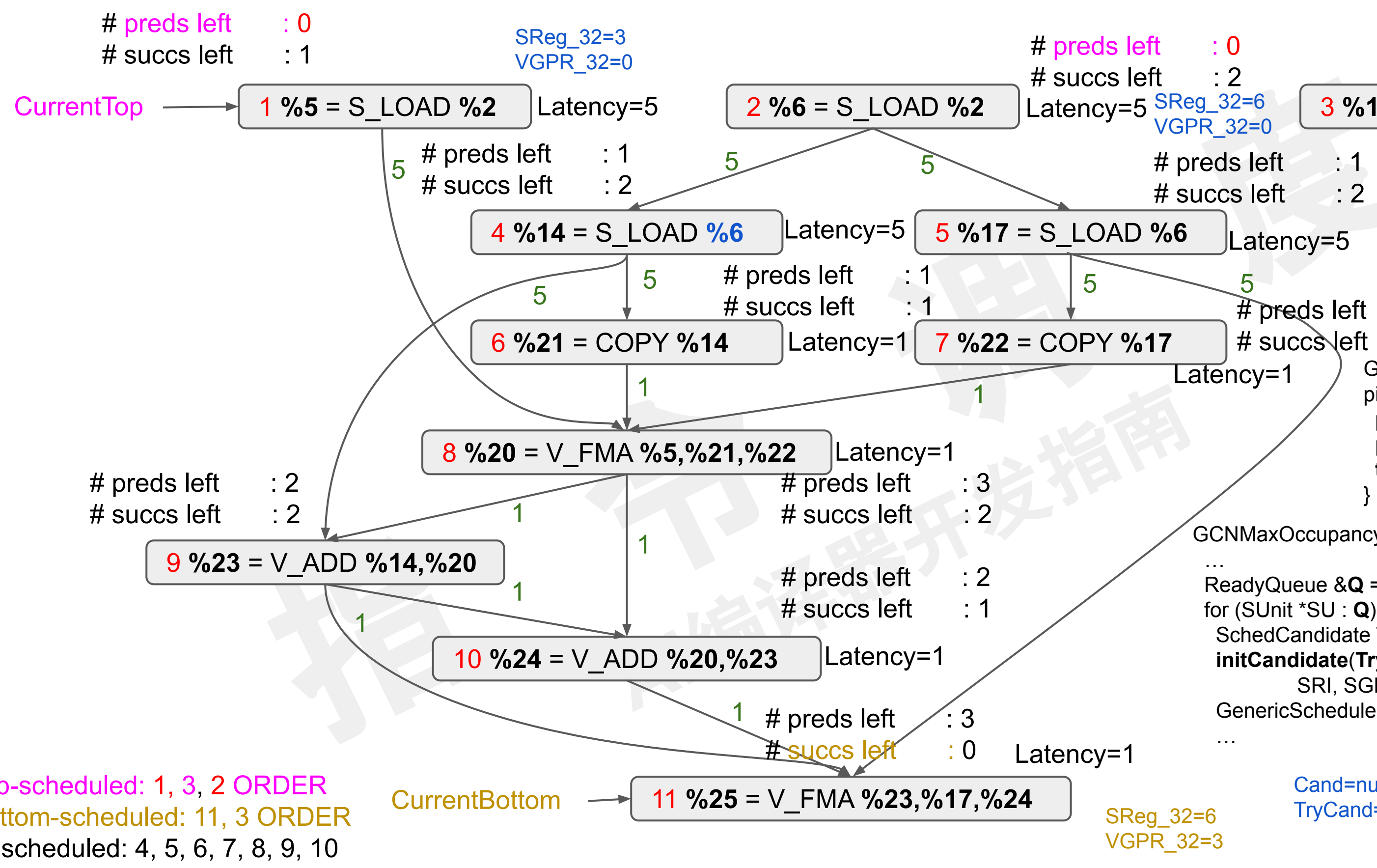
Scheduling SU(12) GLOBAL_STORE %13, %25, %6

```
void GenericScheduler::tryCandidate(SchedCandidate &Cand,
                                    SchedCandidate &TryCand,
                                    SchedBoundary *Zone) const {
  // Initialize the candidate if needed.
  if (!Cand.isValid()) {
    TryCand.Reason = NodeOrder;
    return;
  }

  // Bias PhysReg Defs and copies to their uses and defined respectively.
  if (tryGreater(biasPhysReg(TryCand.SU, TryCand.AtTop),
               biasPhysReg(Cand.SU, Cand.AtTop), TryCand, Cand, PhysReg)) {
    return;
  }
…


int biasPhysReg(const SUnit *SU, bool isTop) {
  const MachineInstr *MI = SU->getInstr();

  if (MI->isCopy()) {
    unsigned ScheduledOper = isTop ? 1 : 0;
    unsigned UnscheduledOper = isTop ? 0 : 1;
    if (Register::isPhysicalRegister(MI->getOperand(ScheduledOper).getReg()))
      return 1;
    …
  }

  if (MI->isMoveImmediate()) {
    bool DoBias = true;
    for (const MachineOperand &Op : MI->defs()) {
      if (Op.isReg() && !Register::isPhysicalRegister(Op.getReg())) {
        DoBias = false;
        break;
      }
    }
  …
  return 0;
}
```

```
0 %2 = COPY s[4:5]
```

```
3 %13 = V_MOVE 0
```

```
void ScheduleDAGMILive::schedule() {
  …
  while (true) {
    SUnit *SU = SchedImpl->pickNode(IsTopNode);
    if (!SU) break;
    scheduleMI(SU, IsTopNode);
    SchedImpl->schedNode(SU, IsTopNode);
    updateQueues(SU, IsTopNode);
  }
}


void ScheduleDAGMI::updateQueues(SUnit *SU, bool IsTopNode) {
  // Release dependent instructions for scheduling.
  if (IsTopNode)
    releaseSuccessors(SU);
  else
    releasePredecessors(SU);

  SU->isScheduled = true;
}
```



Top-scheduled: 1, 3, 2 ORDER
Bottom-scheduled: 11, 3 ORDER
Unscheduled: 4, 5, 6, 7, 8, 9, 10

```
GCNMaxOccupancySchedStrategy::
pickNodeBidirectional(bool &IsTopNode) {
  pickNodeFromQueue(Bot, ... , BotCand);
  pickNodeFromQueue(Top, ... , TopCand);
  tryCandidate(BotCand, TopCand, nullptr);
}
```

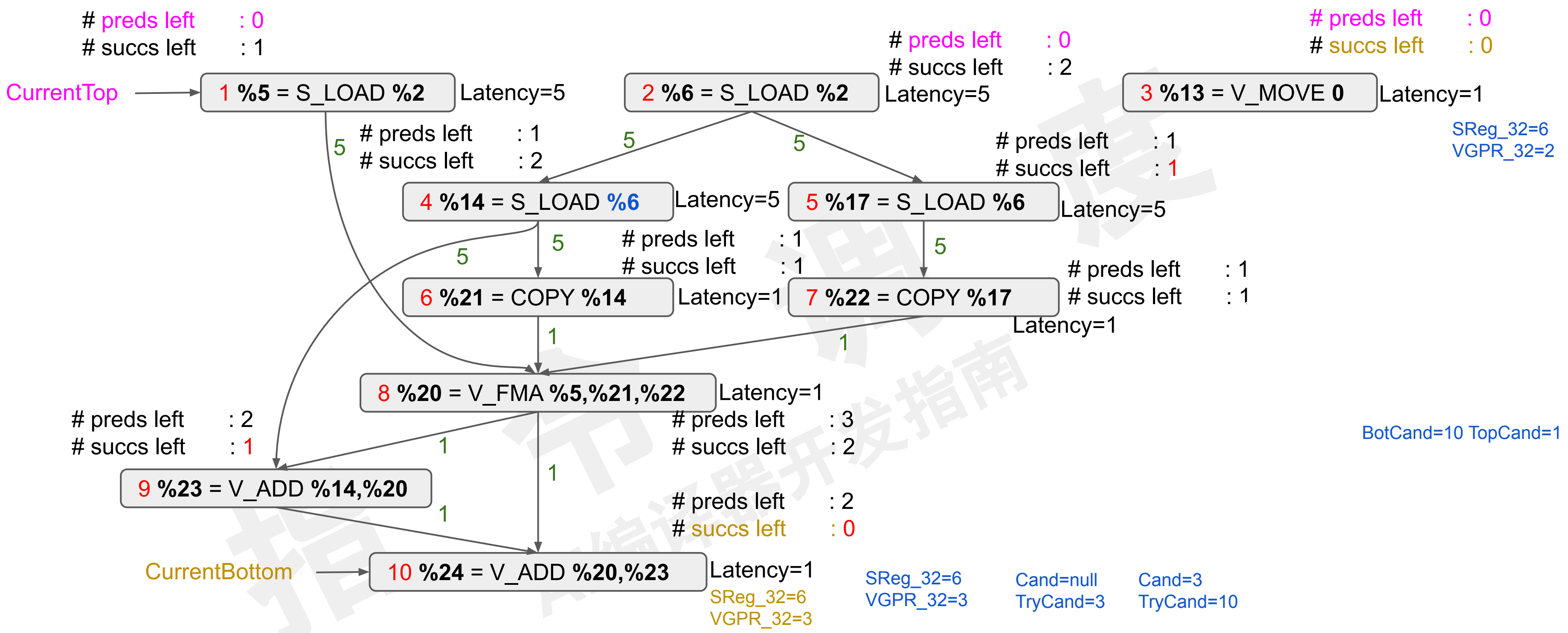
BotCand=11 TopCand=1

```
GCNMaxOccupancySchedStrategy::pickNodeFromQueue(...) {
  …
  ReadyQueue &Q = Zone.Available;
  for (SUnit *SU : Q) {
    SchedCandidate TryCand(ZonePolicy);
    initCandidate(TryCand, SU, Zone.isTop(), RPTracker,
                  SRI, SGPRPressure, VGPRPressure);
    GenericScheduler::tryCandidate(Cand, TryCand, ZoneArg);
  …
```

Cand=null	Cand=3	Cand=3
TryCand=3	TryCand=2	TryCand=1

RPTracker.CurrSetPressure =
SReg_32=2, VGPR_32=0

```
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5 PHYS-REG
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```



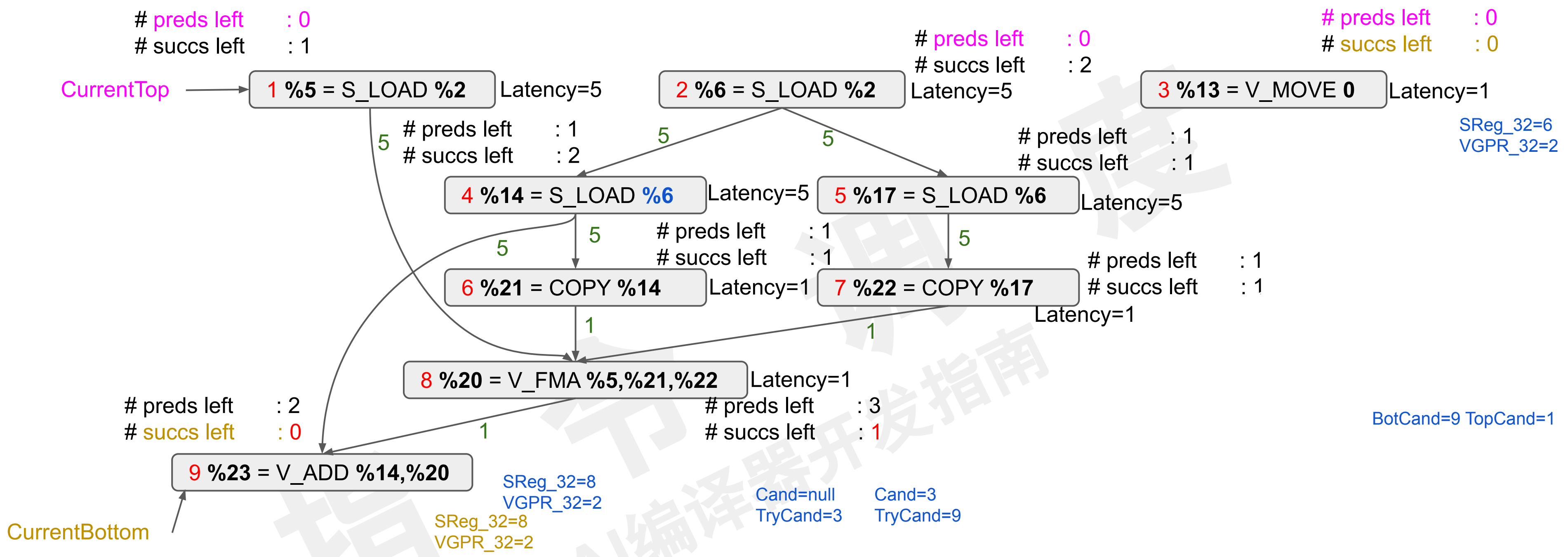
Top-scheduled: 1, 3, 2 NOCAND

Bottom-scheduled: 10, 3 ORDER

Unscheduled: 4, 5, 6, 7, 8, 9

RPTracker.CurrSetPressure =
SReg_32=6, VGPR_32=3

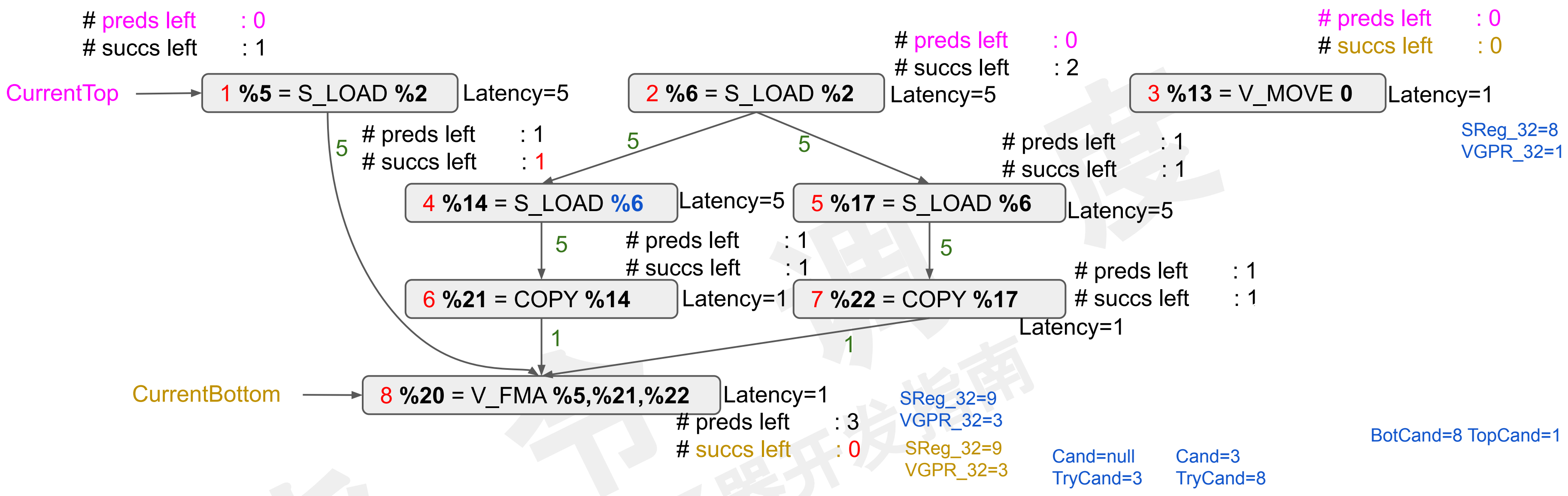
```
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```



Top-scheduled: 1, 3, 2 NOCAND
Bottom-scheduled: 9, 3 ORDER
Unscheduled: 4, 5, 6, 7, 8

RPTracker.CurrSetPressure =
SReg_32=6, VGPR_32=3

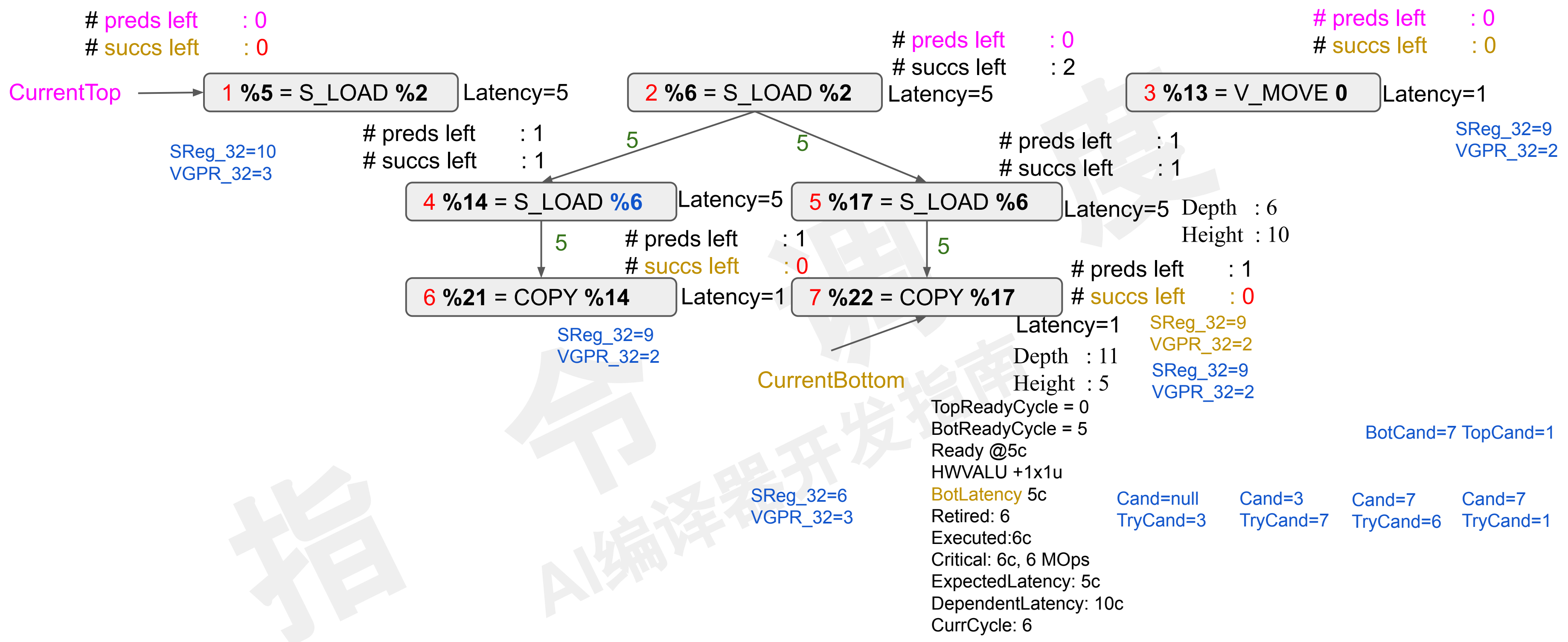
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6



Top-scheduled: 1, 3, 2 NOCAND
Bottom-scheduled: 8, 3 ORDER
Unscheduled: 4, 5, 6, 7

RPTracker.CurrSetPressure =
SReg_32=8, VGPR_32=2

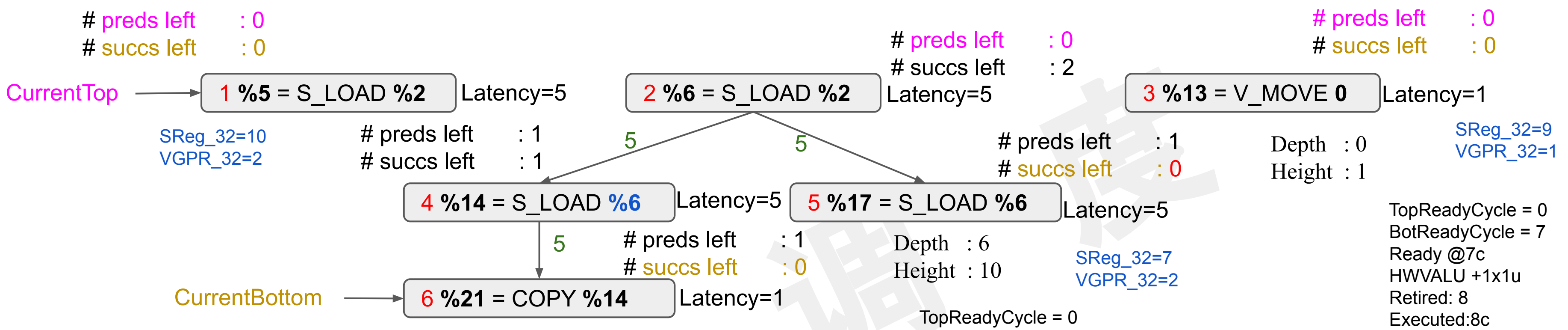
```
Scheduling SU(9) %23 = V_ADD %14, %20
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```



Top-scheduled: 1, 3, 2 NOCAND
Bottom-scheduled: 7, 3, 6, 1 WEAK
Unscheduled: 4, 5

RPTracker.CurrSetPressure =
SReg_32=9, VGPR_32=3

```
Scheduling SU(8) %20 = V_FMA 0, %5, %21, %22
Scheduling SU(9) %23 = V_ADD %14, %20
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```



TopReadyCycle = 0
BotReadyCycle = 7
Ready @7c
HWVALU +1x1u
Retired: 8
Executed:8c
Critical: 8c, 8 MOps
ExpectedLatency: 5c
DependentLatency: 9c
CurrCycle: 8

BotCand=6 TopCand=1

TopReadyCycle = 0
BotReadyCycle = 5->6
Ready @6c
HWVALU +1x1u
TopLatency 11c
Retired: 7
Executed:7c
Critical: 7c, 7 MOps
ExpectedLatency: 5c
DependentLatency: 10c
CurrCycle: 7

Depth : 11
Height : 5

SReg_32=9
VGPR_32=1

SReg_32=9
VGPR_32=1

TopReadyCycle = 0
BotReadyCycle = 10
Ready @10c
***** Stall until: 10**
HWLGKM +1x1u
BotLatency 10c
Retired: 9
Executed:11c
Critical: 9c, 9 MOps
ExpectedLatency: 10c
DependentLatency: 6c
CurrCycle: 11

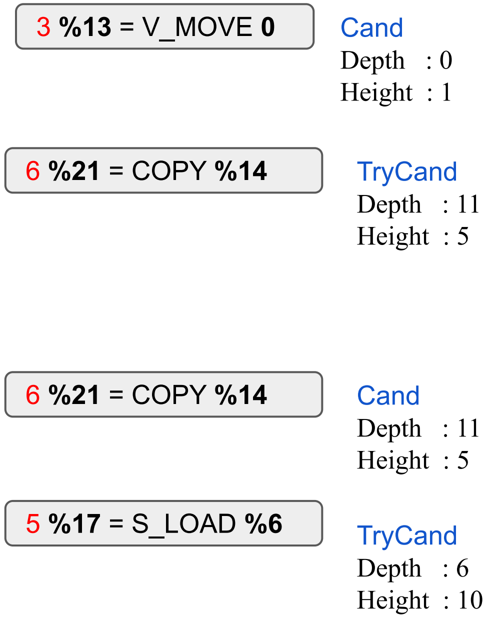
Cand=null	Cand=3	Cand=3	Cand=6
TryCand=3	TryCand=1	TryCand=6	TryCand=5

Top-scheduled: 1, 3, 2 NOCAND
Bottom-scheduled: 6, 3, 1, 5 BOT-HEIGHT
Unscheduled: 4

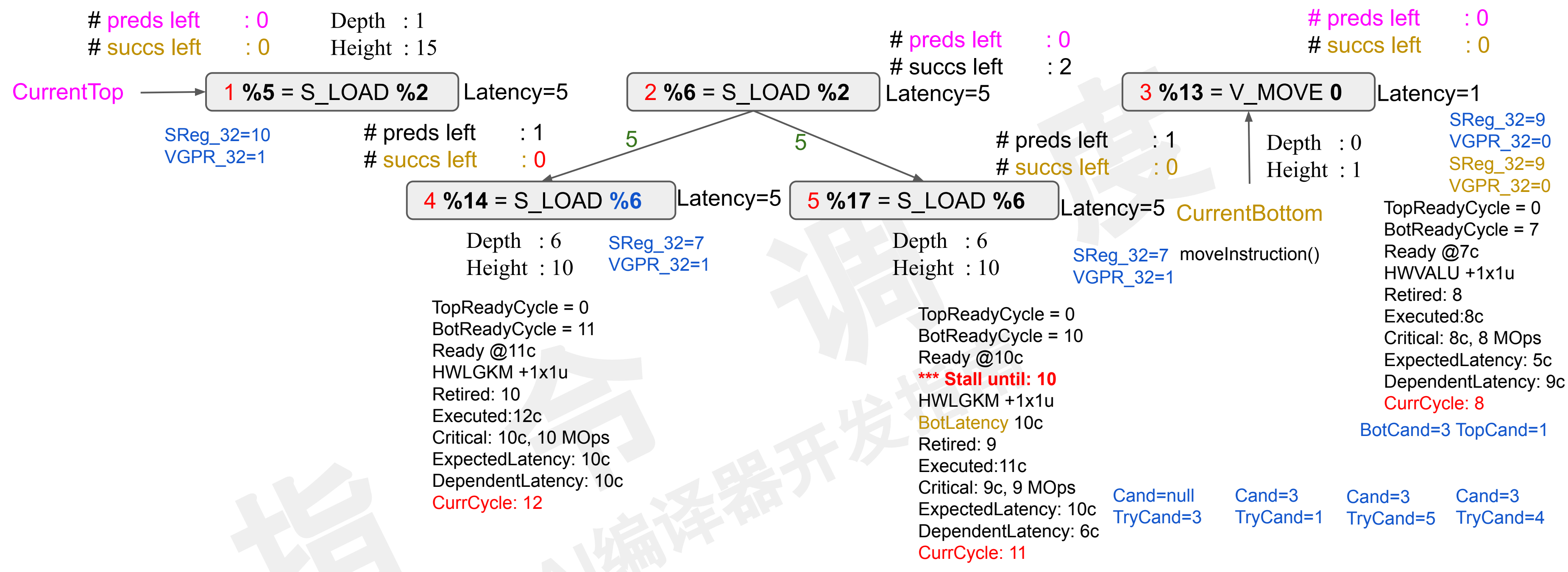
RPTracker.CurrSetPressure =
SReg_32=9, VGPR_32=2

```
Scheduling SU(7) %22 = COPY %17 WEAK
Scheduling SU(8) %20 = V_FMA 0, %5, %21, %22
Scheduling SU(9) %23 = V_ADD %14, %20
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```

```
bool tryLatency(GenericSchedulerBase::SchedCandidate &TryCand,
                GenericSchedulerBase::SchedCandidate &Cand,
                SchedBoundary &Zone) {
  if (Zone.isTop()) {
    …
  } else {
    // Prefer the candidate with the lesser height, but only if one of them has
    // height greater than the total latency scheduled so far, otherwise either
    // of them could be scheduled now with no stall.
    if (std::max(TryCand.SU->getHeight(), Cand.SU->getHeight()) >
        Zone.getScheduledLatency()) {
      if (tryLess(TryCand.SU->getHeight(), Cand.SU->getHeight(),
                  TryCand, Cand, GenericSchedulerBase::BotHeightReduce))
        return true;
    }
    if (tryGreater(TryCand.SU->getDepth(), Cand.SU->getDepth(),
                   TryCand, Cand, GenericSchedulerBase::BotPathReduce))
      return true;
  }
  return false;
}
```



```
3 %13 = V_MOVE 0        Cand
                        Depth  : 0
                        Height : 1

6 %21 = COPY %14        TryCand
                        Depth  : 11
                        Height : 5

6 %21 = COPY %14        Cand
                        Depth  : 11
                        Height : 5

5 %17 = S_LOAD %6       TryCand
                        Depth  : 6
                        Height : 10
```



TopReadyCycle = 0
BotReadyCycle = 11
Ready @11c
HWLGKM +1x1u
Retired: 10
Executed:12c
Critical: 10c, 10 MOps
ExpectedLatency: 10c
DependentLatency: 10c
CurrCycle: 12

TopReadyCycle = 0
BotReadyCycle = 10
Ready @10c
***** Stall until: 10**
HWLGKM +1x1u
BotLatency 10c
Retired: 9
Executed:11c
Critical: 9c, 9 MOps
ExpectedLatency: 10c
DependentLatency: 6c
CurrCycle: 11

TopReadyCycle = 0
BotReadyCycle = 7
Ready @7c
HWVALU +1x1u
Retired: 8
Executed:8c
Critical: 8c, 8 MOps
ExpectedLatency: 5c
DependentLatency: 9c
CurrCycle: 8

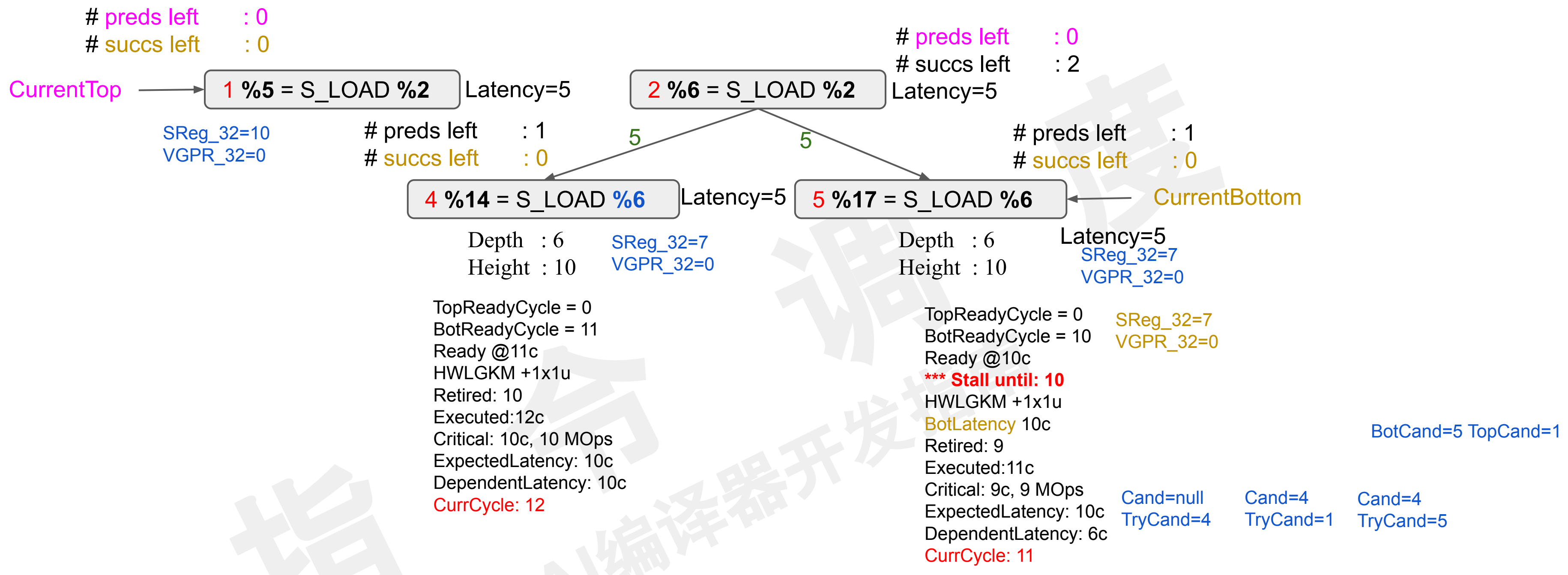
BotCand=3 TopCand=1

Cand=null	Cand=3	Cand=3	Cand=3
TryCand=3	TryCand=1	TryCand=5	TryCand=4

Top-scheduled: 1, 3, 2 NOCAND
Bottom-scheduled: 3, 1, 5, 4 WEAK
Unscheduled:

RPTracker.CurrSetPressure =
SReg_32=9, VGPR_32=1

```
Scheduling SU(6) %21 = COPY %14 BOT-HEIGHT
Scheduling SU(7) %22 = COPY %17
Scheduling SU(8) %20 = V_FMA 0, %5, %21, %22
Scheduling SU(9) %23 = V_ADD %14, %20
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```



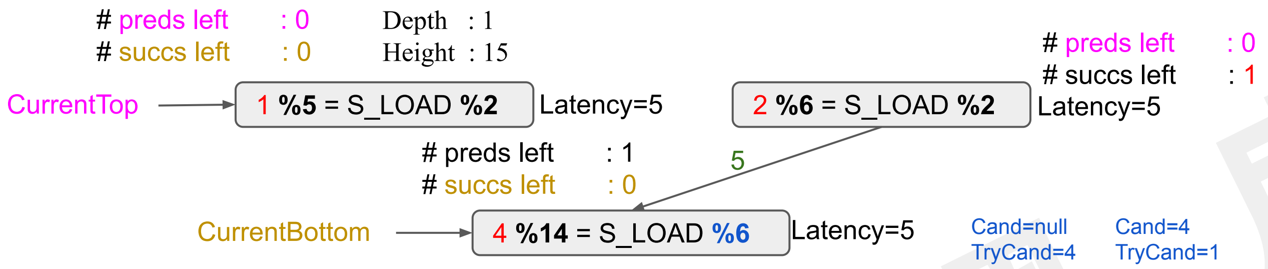
```
TopReadyCycle = 0
BotReadyCycle = 11
Ready @11c
HWLGKM +1x1u
Retired: 10
Executed:12c
Critical: 10c, 10 MOps
ExpectedLatency: 10c
DependentLatency: 10c
CurrCycle: 12
```

```
TopReadyCycle = 0
BotReadyCycle = 10
Ready @10c
*** Stall until: 10
HWLGKM +1x1u
BotLatency 10c
Retired: 9
Executed:11c
Critical: 9c, 9 MOps
ExpectedLatency: 10c
DependentLatency: 6c
CurrCycle: 11
```

Top-scheduled: 1, 2 NOCAND
Bottom-scheduled: 5, 1, 4 ORDER
Unscheduled:

RPTracker.CurrSetPressure =
SReg_32=9, VGPR_32=0

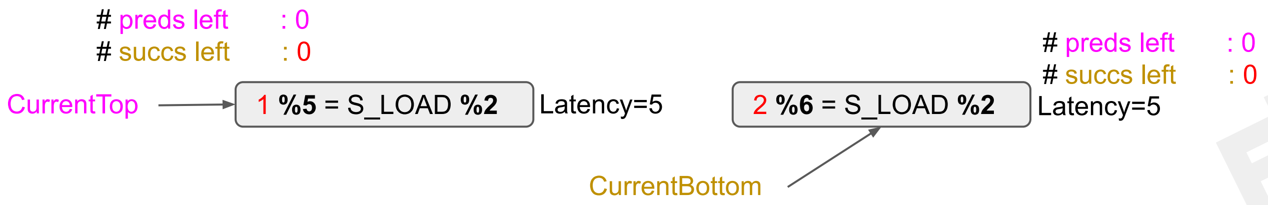
```
Scheduling SU(3) %13 = V_MOV 0 WEAK
Scheduling SU(6) %21 = COPY %14
Scheduling SU(7) %22 = COPY %17
Scheduling SU(8) %20 = V_FMA 0, %5, %21, %22
Scheduling SU(9) %23 = V_ADD %14, %20
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```



BotCand=4 TopCand=1

Top-scheduled: 1, 2 NOCAND
Bottom-scheduled: 4, 1 WEAK
Unscheduled:

```
Scheduling SU(5) %17 = S_LOAD %6 ORDER
Scheduling SU(3) %13 = V_MOV 0
Scheduling SU(6) %21 = COPY %14
Scheduling SU(7) %22 = COPY %17
Scheduling SU(8) %20 = V_FMA 0, %5, %21, %22
Scheduling SU(9) %23 = V_ADD %14, %20
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```



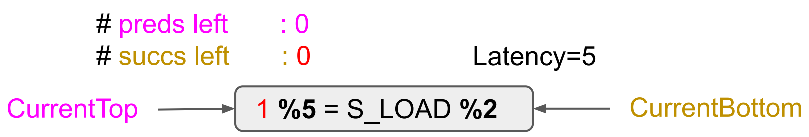
Cand=null
TryCand=2

Cand=2
TryCand=1

BotCand=2 TopCand=1

```
Scheduling SU(4) %14 = S_LOAD %6 WEAK
Scheduling SU(5) %17 = S_LOAD %6
Scheduling SU(3) %13 = V_MOV 0
Scheduling SU(6) %21 = COPY %14
Scheduling SU(7) %22 = COPY %17
Scheduling SU(8) %20 = V_FMA 0, %5, %21, %22
Scheduling SU(9) %23 = V_ADD %14, %20
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```

Top-scheduled: 1, 2 NOCAND
Bottom-scheduled: 2, 1 WEAK
Unscheduled:



```
void ScheduleDAGMILive::schedule() {
  buildDAGWithRegPressure();
  postprocessDAG();
  findRootsAndBiasEdges(TopRoots, BotRoots);
  initQueues(TopRoots, BotRoots);
  while (true) {
    SUnit *SU = SchedImpl->pickNode(IsTopNode);
    if (!SU) break;
    scheduleMI(SU, IsTopNode);
    SchedImpl->schedNode(SU, IsTopNode);
    updateQueues(SU, IsTopNode);
  }
}
```

```
Scheduling SU(2) %6 = S_LOAD %2 WEAK
Scheduling SU(4) %14 = S_LOAD %6
Scheduling SU(5) %17 = S_LOAD %6
Scheduling SU(3) %13 = V_MOV 0
Scheduling SU(6) %21 = COPY %14
Scheduling SU(7) %22 = COPY %17
Scheduling SU(8) %20 = V_FMA 0, %5, %21, %22
Scheduling SU(9) %23 = V_ADD %14, %20
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```

Top-scheduled: 1
Bottom-scheduled: 1
Unscheduled:

```
void ScheduleDAGMILive::schedule() {
  buildDAGWithRegPressure();
  postprocessDAG();
  findRootsAndBiasEdges(TopRoots, BotRoots);
  initQueues(TopRoots, BotRoots);
  while (true) {
    SUnit *SU = SchedImpl->pickNode(IsTopNode);
    if (!SU) break;
    scheduleMI(SU, IsTopNode);
    SchedImpl->schedNode(SU, IsTopNode);
    updateQueues(SU, IsTopNode);
  }
}
```

```
Scheduling SU(1) %5 = S_LOAD %2
Scheduling SU(2) %6 = S_LOAD %2
Scheduling SU(4) %14 = S_LOAD %6
Scheduling SU(5) %17 = S_LOAD %6
Scheduling SU(3) %13 = V_MOV 0
Scheduling SU(6) %21 = COPY %14
Scheduling SU(7) %22 = COPY %17
Scheduling SU(8) %20 = V_FMA 0, %5, %21, %22
Scheduling SU(9) %23 = V_ADD %14, %20
Scheduling SU(10) %24 = V_ADD %20, %23
Scheduling SU(11) %25 = V_FMA 0, %23, %17, %24
Scheduling SU(0) %2 = COPY $sgpr4_sgpr5
Scheduling SU(12) GLOBAL_STORE %13, %25, %6
```

Top-scheduled:
Bottom-scheduled:
Unscheduled: