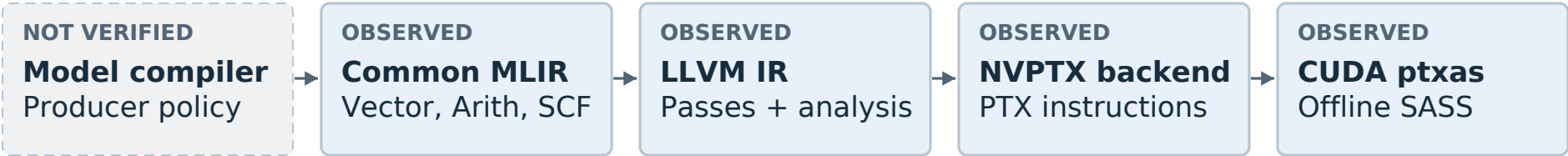


Qwen static compiler analysis

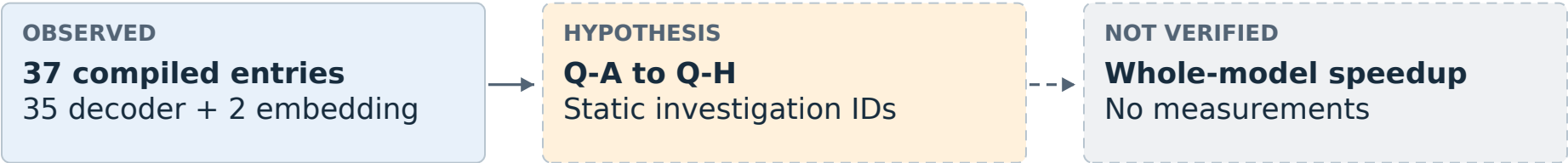
Q-A to Q-H | Controlled English based on ASD-STE100 Issue 9

Figure 1. Compiler boundaries



Q-A: MLIR | Q-B/C/E: LLVM | Q-D/H: boundary open | Q-F/G: NVPTX hypothesis
Offline SASS is not verified driver JIT SASS.

Figure 2. Evidence and status



Instruction counts do not establish execution cost.

Scope	Evidence	Reading key
The guide explains eight static hypotheses. It covers 35 Qwen2.5-7B entries and two Qwen3.5-4B embedding entries.	The review uses saved MLIR, LLVM IR, PTX and offline SASS. No new compiler or GPU run supports this guide.	Blue means observed code. Green means an arithmetic relation. Dashed amber means a hypothesis. Dashed gray means an unverified boundary.

These IDs are not CAND numbers. The guide does not establish a measured hotspot or speedup.

Eight items, different compiler layers

The source location and the unresolved boundary are separate facts.

Q-A	Common MLIR Vector	Shifted contiguous gather
Q-B	LLVM analysis	Correlated row-address alignment
Q-C	LLVM InstCombine	Alignment assume removal
Q-D	LLVM or common MLIR	Split and rebuilt address
Q-E	LLVM analysis	Nested address and depth limit
Q-F	NVPTX hypothesis	Exact integer division setup
Q-G	NVPTX hypothesis	Exact f32 division setup
Q-H	Boundary open	Zero-or-one iteration proof

Q-F and Q-G start from code that ptxas expands. They do not establish an LLVM defect.

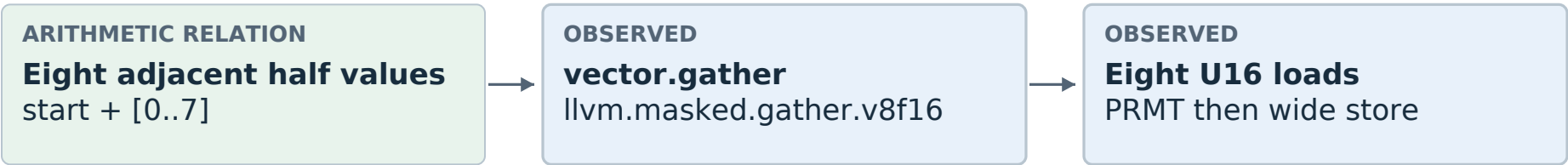
Q-H and Softmax exp recomputation still need earlier boundary evidence.

Producer-only changes and external-tool internals remain outside the contribution scope.

Q-A | Shifted contiguous gather

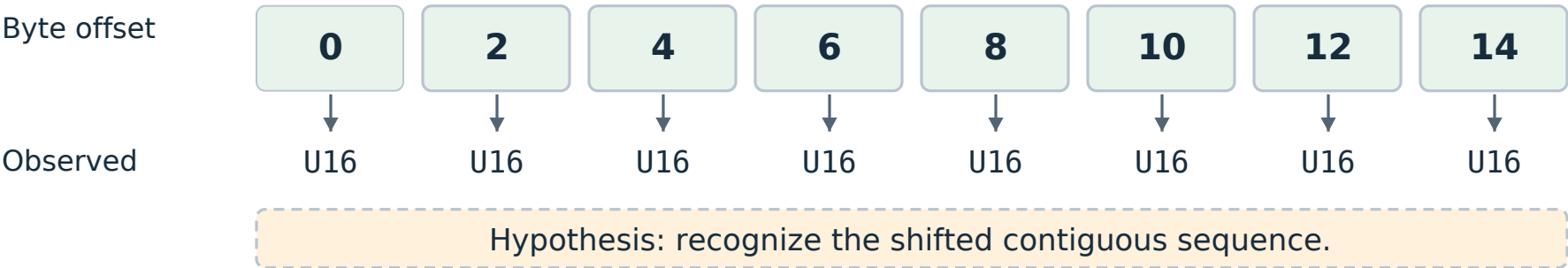
7B: 0 | 4B embedding: 0 | Common MLIR Vector and Vector-to-LLVM

Figure 3. Q-A: The gather path



7B dispatch 0 / 4B embedding dispatch 0.

Figure 4. Q-A: Addresses and the proposed recognition



Computation

The embedding kernel copies one table row. Each thread reads eight adjacent half values.

Observed code

The IR keeps a shifted gather. Offline SASS has eight U16 loads and packing instructions.

Source hypothesis

The common matcher identifies zero-based sequences. A source extension can identify the shifted sequence.

Meaning conditions: The transform needs unit stride and valid cast ranges. It must keep the mask and pass-through behavior. Negative-token handling must stay unchanged.

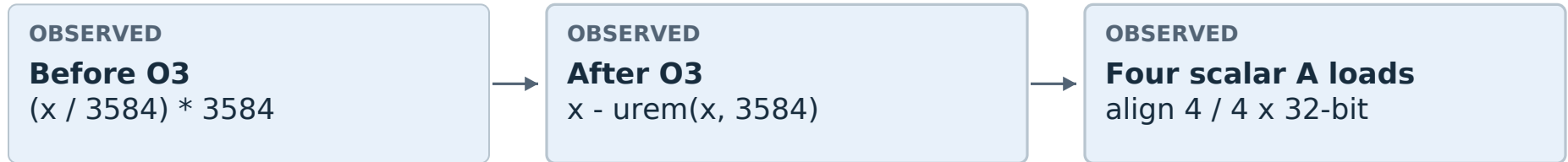
Open boundary: The first gather creation and fold positions remain unknown. The generic draft has no execution result.

Source locations: page 15. Exact code mapping: page 16.

Q-B | Row-address alignment relationship

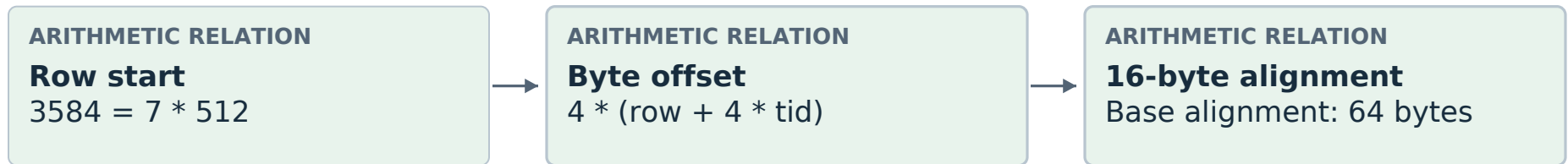
7B: 35, 61 | LLVM KnownBits and InstCombine

Figure 5. Q-B: The row address changes form



The A base keeps align 64. The lost relationship differs from Q-C.

Figure 6. Q-B: The alignment relationship



Hypothesis: KnownBits can use the relationship between x and its remainder.

Computation

The output projection reduces a dot product across 28 warps. Each thread reads four adjacent A values.

Meaning conditions: The analysis must keep freeze, undef and poison semantics. It must also respect the divisor and integer width.

Open boundary: The first missed alignment query has no trace. The A base still has align 64.

Source locations: page 15. Exact code mapping: page 16.

Observed code

O3 changes the row address to x minus its remainder. Four scalar A loads keep only align 4.

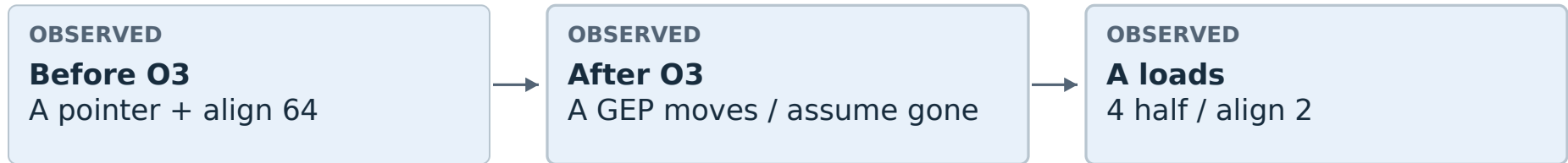
Source hypothesis

KnownBits treats the operands separately. A bounded relation can show the zero low bits of the row address.

Q-C | Alignment fact and instruction sinking

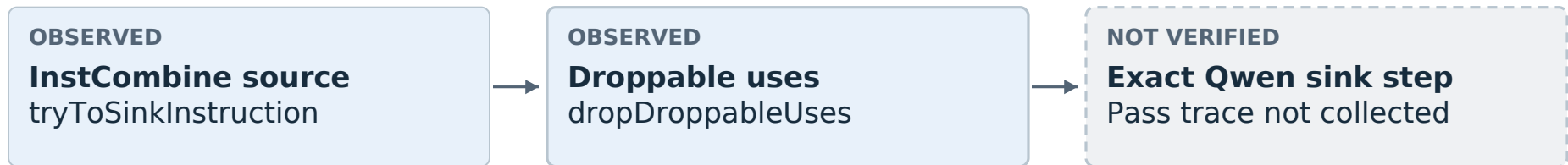
7B: 15, 37, 38, 743 | LLVM InstCombine

Figure 7. Q-C: The alignment fact disappears



Dispatch 15, 37, 38 and 743. This is a recurrence of the G01 investigation.

Figure 8. Q-C: Source behavior and the open trace



Historical HS014 patch exists. Its Qwen effect is not verified.

Computation

These projections reduce dot products.
The A input supplies adjacent half values to each thread.

Observed code

The A alignment assume disappears in O3. The moved A address then supplies loads with align 2.

Source hypothesis

Instruction sinking can remove droppable uses. The source behavior agrees with the observed change.

Meaning conditions: A source change must keep dominance and execution-path rules. The historical patch does not establish a Qwen speedup.

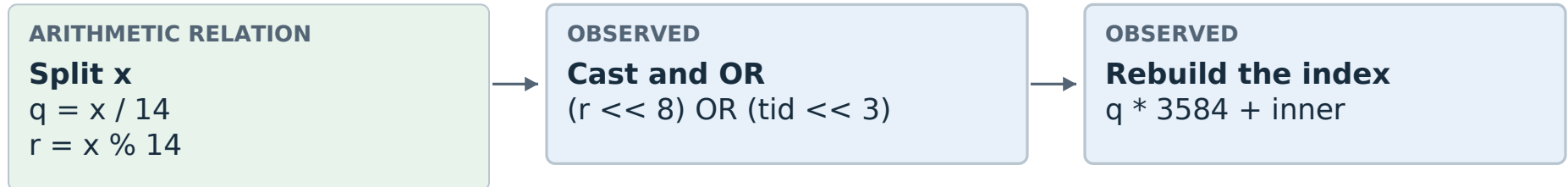
Open boundary: The exact Qwen sink step has no trace. This item continues G01. It is not a new independent candidate.

Source locations: page 15. Exact code mapping: page 16.

Q-D | Split and rebuilt address

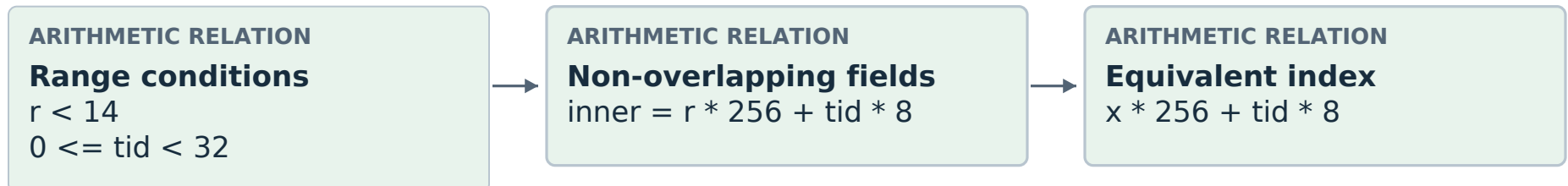
7B: 13, 31 | 4B embedding: 1 | LLVM InstCombine or common MLIR arithmetic

Figure 9. Q-D: Split and rebuild the address



SASS keeps quotient work in the loop. Wide load and store already exist.

Figure 10. Q-D: The same address with fewer terms



The implementation must keep cast, wrap, poison and memory semantics.

Computation

The kernels convert or scale eight values per thread. Their addresses split a flat index into quotient and remainder.

Meaning conditions: The transform must keep cast and integer-wrap semantics. Signed correction needs a separate proof.

Open boundary: The first missed fold remains unknown. Wide loads and stores already exist.

Source locations: page 15. Exact code mapping: page 16.

Observed code

Casts and OR separate the related terms. SASS keeps quotient work inside the loop.

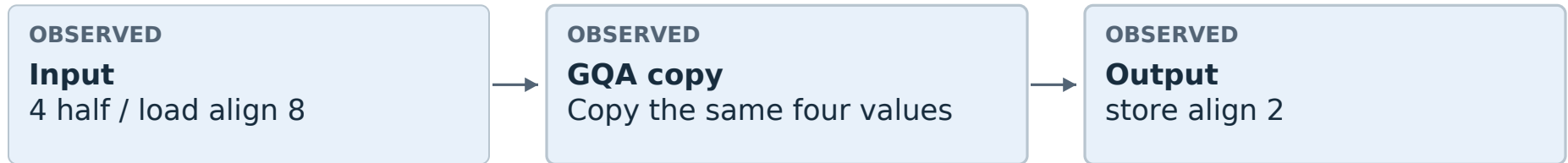
Source hypothesis

The existing direct matcher cannot see this form. Range and bit-field relations can show an equivalent index.

Q-E | Nested output-address alignment

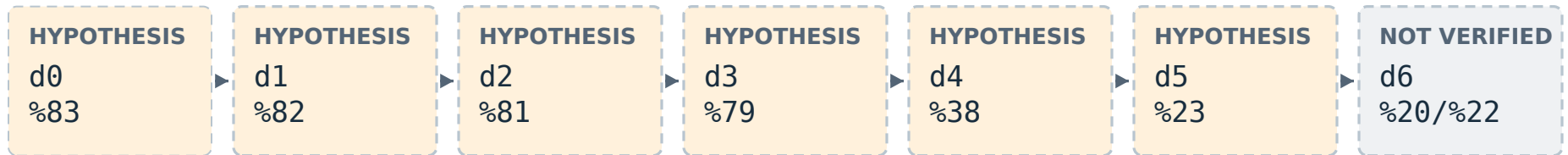
7B: 28 | LLVM KnownBits and alignment analysis

Figure 11. Q-E: One copy has different load and store widths



SASS: one LDG64 at 0a50 / four STG.U16 at 0b80, 0ba0, 0bb0 and 0bc0.

Figure 12. Q-E: Static path toward the extent facts



Depth limit: 6. The diagram follows a source-based query hypothesis.

Hypothesis only: the actual query stack is not verified.

Computation

The GQA copy repeats four KV heads to make 28 heads. Each thread copies four half values.

Meaning conditions: The extent is a multiple of 128. The byte offset is a multiple of eight.

Open boundary: The actual query stack has no trace. GQA materialization removal would be a separate producer change.

Source locations: page 15. Exact code mapping: page 16.

Observed code

The input uses one 64-bit load. The output uses four 16-bit stores.

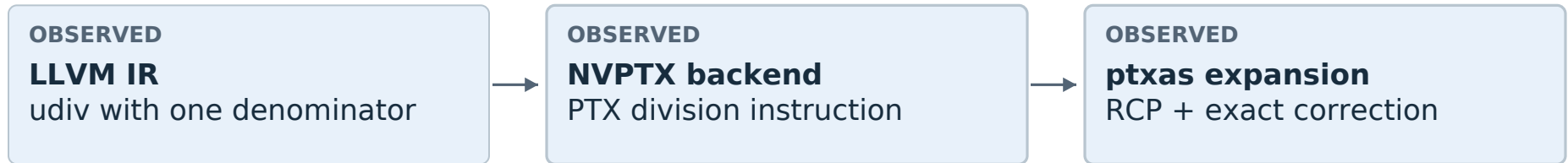
Source hypothesis

The output query reaches depth 6 before the extent facts. This depth path can explain the alignment difference.

Q-F | Invariant work in exact integer division

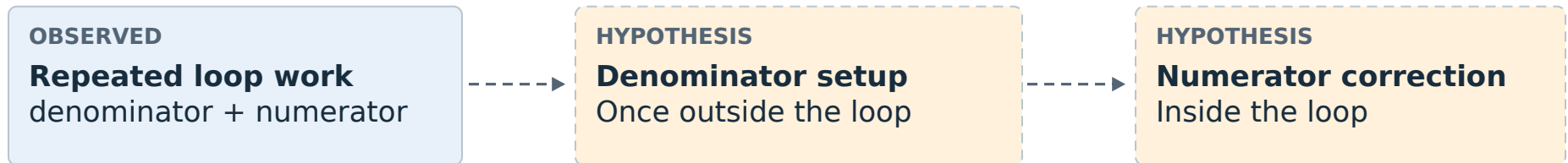
7B: 2, 4 | Possible LLVM NVPTX source implementation

Figure 13. Q-F: The denominator work becomes visible late



Regular LICM does not see the ptxas expansion. The 64-bit fallback remains necessary.

Figure 14. Q-F: A possible separation of exact division



This investigation has no source implementation or exact arithmetic proof.

Computation

Index and mask loops divide different numerators by an invariant denominator. The original arithmetic has 64-bit semantics.

Observed code

ptxas adds reciprocal setup and quotient correction. The setup occurs inside the observed loop.

Source hypothesis

A NVPTX source implementation can expose denominator work before PTX emission. LLVM can then share that work.

Meaning conditions: Each numerator needs exact quotient correction. The implementation must keep the 64-bit fallback and signed corner cases.

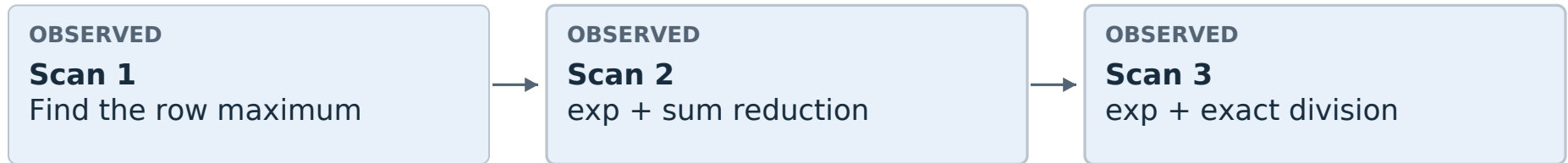
Open boundary: This investigation has no exact algorithm or source patch. This observation does not establish a defect in LLVM.

Source locations: page 15. Exact code mapping: page 16.

Q-G | One denominator for exact Softmax division

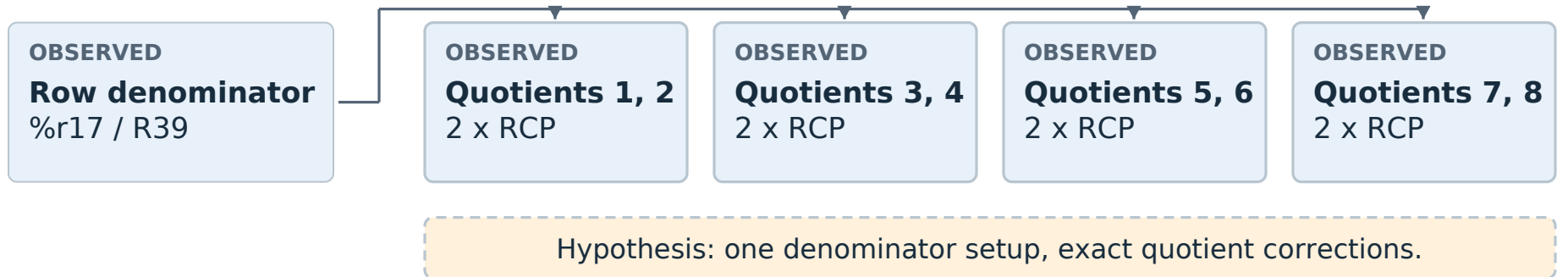
7B: 33 | Possible LLVM NVPTX source implementation

Figure 15. Q-G: Softmax has three scans



Scores and mask are read again. Four barriers order reduction and buffer reuse.

Figure 16. Q-G: Eight quotients use the same denominator



Computation

Softmax uses maximum, sum and output scans. Eight quotients use one row denominator.

Observed code

Eight PTX divisions use the same denominator. Offline SASS has eight reciprocal operations on R39.

Source hypothesis

A source implementation can share denominator setup. Each quotient must keep its own rounding and error correction.

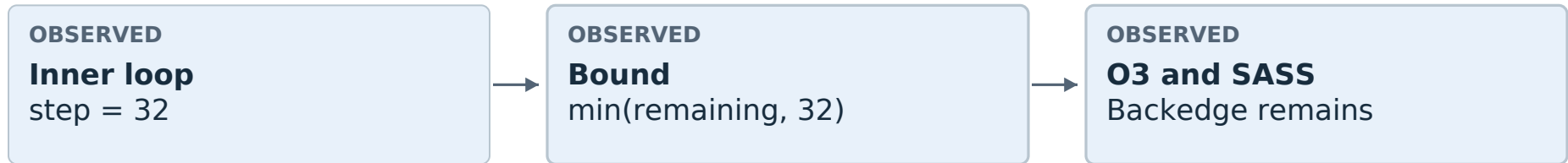
Meaning conditions: Reciprocal multiplication can change exact division results. Zero, subnormal, NaN and infinity cases need the original behavior.
Open boundary: The exp recomputation boundary remains unknown. The frozen common Linalg decomposition shares one exp tensor.

Source locations: page 15. Exact code mapping: page 16.

Q-H | Range of a small tile loop

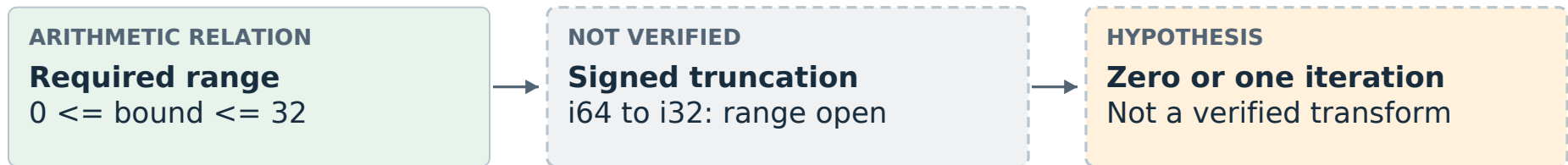
7B examples: 1, 3, 7 | Common SCF, arithmetic or LLVM loop analysis / boundary open

Figure 17. Q-H: The small loop still has a backedge



Dispatch 1, 3 and 7 are examples.

Figure 18. Q-H: The missing range proof



The SCF-to-CF snapshot and the first lost relationship remain open.

Computation

The tile loop has step 32. Its upper bound is the smaller of remaining and 32.

Observed code

The backedge remains in O3 and SASS. Signed truncation prevents an unconditional zero-or-one iteration claim.

Source hypothesis

A valid range proof can remove the backedge. The responsible common transform remains unknown.

Meaning conditions: The proof must follow the signed range through i64-to-i32 truncation. It must also cover the outer loop bounds.

Open boundary: The SCF-to-CF snapshot is missing. A producer-only loop change would not qualify.

Source locations: page 15. Exact code mapping: page 16.

Validation procedures

These steps are proposals. This guide does not report their execution.

Q-A

1. Run the generic shifted-gather test.
2. Add tests for partial masks and non-unit stride.
3. Compare the source change in SASS.

Q-B

1. Compare the two row-address forms in generic IR.
2. Record the first missed alignment query.
3. Measure the cost of the new analysis.

Q-C

1. Record the exact sink step for the A address.
2. Run the focused test on the current revision.
3. Compare Qwen outputs and SASS after the source change.

Q-D

1. Run the generic unsigned address test.
2. Add tests for casts, multiple uses and poison.
3. Locate the first common transform that misses the relation.

Q-E

1. Record the input and output alignment queries.
2. Compare shallow and nested GEPs in generic IR.
3. Do not increase the depth limit without a cost analysis.

Q-F

1. Create a producer-free exact-division test.
2. Separate denominator work from numerator correction.
3. Compare SASS dependencies and register use.

Q-G

1. Create a strict same-denominator division test.
2. Do exact-output tests for exceptional values.
3. Find the decomposition and fusion steps before the repeated exp calculations.

Q-H

1. Collect the SCF-to-CF boundary.
2. Show the signed range before and after truncation.
3. Locate the first loss of the loop-bound relationship.

Technical glossary 1

Each technical term has one defined use in this guide.

IR	Intermediate representation. A compiler uses this form of a program.
dispatch	One compiled entry. A model can call the same entry more than once.
producer	The model compiler that creates the initial computation and memory organization.
common MLIR	Upstream MLIR operations and transforms that do not require a producer-specific dialect.
lowering	A compiler conversion from one representation to another representation.
NVPTX	The LLVM backend that produces PTX for NVIDIA GPUs.
PTX	The virtual instruction form that the NVIDIA toolchain accepts.
ptxas	The NVIDIA tool that converts PTX into machine code.
SASS	NVIDIA machine instructions. This guide uses offline ptxas output.
driver JIT	The compiler in the GPU driver. This guide does not verify its loaded machine code.
half / f32	A 16-bit floating-point value / a 32-bit floating-point value.
align N	An IR contract that the address is a multiple of N bytes.
gather	A vector memory operation that reads addresses from an index vector.
mask / pass-through	Active-lane selection / the value for an inactive load lane.
GEP	The LLVM getelementptr instruction. It calculates an address.
KnownBits	LLVM analysis that identifies known zero or one bits.

Technical glossary 2

Each technical term has one defined use in this guide.

assume	An IR statement that gives facts which the optimizer can use.
instruction sinking	A transform that moves an instruction toward its use.
droppable use	An IR use that a transform can remove without a change to required program results.
freeze / poison / undef	LLVM terms for indeterminate values. Their distinct rules restrict valid transforms.
relinearization	The calculation that rebuilds a flat address from separate coordinates.
backedge	The control-flow edge that returns to a loop body.
invariant	A value that does not change during the specified loop.
LICM	Loop Invariant Code Motion. This LLVM transform moves eligible work outside a loop.
reciprocal / RCP	The inverse of a number / a reciprocal machine instruction.
RN	Round to nearest, ties to even. Each exact quotient must keep this rule.
subnormal	A floating-point value with a magnitude below the normal range.
NaN	Not a Number. A special floating-point value.
GQA / KV	Grouped Query Attention / key and value tensors.
CTA / warp / tid	GPU thread block / a group of 32 threads / the thread identifier.
technical verbs	This guide uses compile, truncate, decompose, expose and share as software-process verbs.
technical symbols	Code names, formulas and instruction mnemonics keep their original spelling.

Evidence, provenance and promotion gates

Source excerpts and hashes accompany this document.

Figure 19. From a static hypothesis to a contribution



No Qwen item has passed these gates. A pass-only change does not qualify.

LLVM baseline: 0025dca2990ec75672b5dff25050f103a016d530.

Producer: IREE 442c605b3f85b0728389abb83ad842db38d5108e.

Offline code: CUDA 13.3 ptxas, target sm_89, PTX feature +ptx78.

7B original: Qwen/Qwen2.5-7B-Instruct@a09a35458c702b33eeacc393d103063234e8bc28.

7B ONNX: liodon-ai/Qwen2.5-7B-Instruct-ONNX@84b028251eb9a044bfa523e10651ef087fcb73e0.

4B original: Qwen/Qwen3.5-4B@851bf6e806efd8d0a36b00ddf55e13ccb7b8cd0a.

4B ONNX: onnx-community/Qwen3.5-4B-ONNX@74d8caba2117fd5f41d655e9cc27eda1338662b3.

The evidence directory contains function excerpts and pinned source files. The excerpts are not standalone reproducers.

The manifest records relative paths, sizes and SHA256 hashes. No model weights or tool binaries are included.

Language reference: ASD-STE100 Issue 9, 2025-01-15. The text uses controlled English and a technical glossary.

The length checks do not establish complete dictionary or grammar compliance. The guide makes no certified compliance claim.

Source locations

References use the frozen baseline and the exact saved entry excerpts.

Q-A

VectorOps.cpp:6433-6568 / ConvertVectorToLLVM.cpp:305-366
isZeroBasedContiguousSeq / FoldContiguousGather

Q-B

InstCombineMulDivRem.cpp:461-493 / ValueTracking.cpp:517-543,1689
computeForAddSub

Q-C

InstructionCombining.cpp:5563-5613
tryToSinkInstruction / dropDroppableUses

Q-D

InstCombineAddSub.cpp:1160-1230
SimplifyAddWithRemainder

Q-E

Local.cpp:1579 / ValueTracking.cpp:1717 / ValueTracking.h:47
getOrEnforceKnownAlignment / computeKnownBits

Q-F

NVPTX division emission / CUDA ptxas expansion
No exact source implementation or algorithm is prepared.

Q-G

NVPTXInstrInfo.td:1145-1258 / NVPTXISelLowering.cpp:991-1090
Separate exp boundary: SoftmaxOp::decomposeOperation,
LinalgOps.cpp:3052-3235

Q-H

SCF-to-CF boundary / common loop analysis: not localized
No first missed transform is identified.

These references identify static code. PC values do not establish execution time.

Exact code mapping

References use the frozen baseline and the exact saved entry excerpts.

Q-A

7B table [152064,3584]. 4B table [248320,2560]. LLVM gather: %43, align 2.

4B U16: 04c0, 04f0, 0520, 0550, 0570, 0580, 0590, 05a0. PRMT: 0670-06a0. STG128: 06b0.

Q-B

35 A base: %1. 61 A base: %0. Scalar loads: %45, %47, %49, %51.

35 A loads: 03c0, 03e0, 0400, 0420. 61 A loads: 03b0, 03d0, 03f0, 0410.

Q-C

15: Q projection. 37: up projection. 38: gate and SiLU. 743: lm_head.
The surviving %27 assume in dispatch 38 refers to another epilogue input, not A.

Q-D

7B divisor: 14. Row extent: 3584. 4B divisor: 10. Row extent: 2560.
31: address work 0260-0390, backedge 0480 to 0260. LDG128: 03a0. STG128: 0470.

Q-E

Load: %78, align 8. Output base: %42, align 64. Extent: %23. Fact: (%20 & 127) == 0.

LDG64: 0a50. STG.U16: 0b80, 0ba0, 0bb0, 0bc0. Depth limit: 6.

Q-F

Dispatch 2 uses a 32-bit fast path and a separate 64-bit fallback.
2: reciprocal setup and correction 02e0-0410. Backedge: 05c0 to 0280.

Q-G

PTX denominator: %r17. Division lines: 1173, 1213, 1254, 1294, 1334, 1374, 1415, 1455.

MUFU.RCP on R39: 3c50, 3e80, 4260, 4560, 4850, 4b50, 4e50, 5150.

Q-H

Examples: dispatch 1, 3, 7. Step: 32. Bound: min(remaining,32). Signed cast: i64 to i32.

Zero or one iteration requires the signed-range proof. No exact backedge PC is claimed here.

These references identify static code. PC values do not establish execution time.