Value Range Tracking in NIR

Ian Romanick – X.org Developers Conference 2018 (Lightning Talk)
Overview

- Existing “0th-order” tracking
- WIP 1st-order tracking
- Existing 2nd-order tracking
- Future 2nd-order tracking
Existing 0\textsuperscript{th}-order tracking

Rely on known range produced by certain operations

\[
(('\text{fabs}', ('b2f', a)), ('b2f', a))
\]
WIP 1\textsuperscript{st}-order tracking

Gather information about SSA values based on known properties of operation results.

- Analysis conceptually similar to existing 0\textsuperscript{th}-order
  - Result of fabs must be $\geq 0$, etc.
  - $(\text{value} \geq 0) \times (\text{value} \leq 0) \rightarrow \text{result must be} \leq 0$
  - Analysis is on demand, but results are cached.

- Add simple predicates for use in nir\_opt\_algebraic

  $((\text{'fge'}, \text{'b(is_not_negative)'}, \text{'a(is_not_positive)'}), \text{True})$
WIP 1\textsuperscript{st}-order tracking

Results so far are good

- Two main commits:
  - nir: Use value range analysis to eliminate tautological compares
  - nir: Use value range analysis to convert a fmin to an fsat

\begin{itemize}
  \item total instructions in shared programs: 15088355 -> 15027041 (-0.41%)
  \item instructions in affected programs: 2823740 -> 2762426 (-2.17%)
  \item helped: 10614
  \item HURT: 2
  \item helped stats (abs) min: 1 max: 294 \( \bar{x} \): 5.78 \( \bar{x} \): 2
  \item helped stats (rel) min: 0.05% max: 58.33% \( \bar{x} \): 3.23% \( \bar{x} \): 1.37%
  \item HURT stats (abs) min: 6 max: 6 \( \bar{x} \): 6.00 \( \bar{x} \): 6
  \item HURT stats (rel) min: 0.30% max: 0.30% \( \bar{x} \): 0.30% \( \bar{x} \): 0.30%
\end{itemize}

95\% mean confidence interval for instructions value: -5.99 -5.56
95\% mean confidence interval for instructions \% -change: -3.32\% -3.15\%

Instructions are helped.
Existing 2\textsuperscript{nd}-order tracking

Tim Arceri’s recently did some work to propagate compare results into branches.
Future 2\textsuperscript{nd}-order tracking

Infer value ranges from if-statement conditions, loop conditions, etc.

- Add NIR instructions similar to clang's / MSVC's “assume” built-in.
  - ssa_4 = assume_gt ssa_3, 0
  - Could expose directly in GLSL / SPIR-V

- Allows tracking of ranges after if-statements are replaced with bcsel

- Interferes with copy prop, CSE, etc.
  - Run optimization loop, strip assume instructions, run loop again?

- Or – hash values based on SSA and block ID
  - Harder to deal with bcsel
Questions?

https://cgit.freedesktop.org/~idr/mesa/log/?h=simple-range-analysis
Value Range Tracking in NIR
Ian Romanick – X.org Developers Conference 2018 (Lightning Talk)
Overview

- Existing “0th-order” tracking
- WIP 1st-order tracking
- Existing 2nd-order tracking
- Future 2nd-order tracking
Existing 0\textsuperscript{th}-order tracking

Rely on known range produced by certain operations

\[ (\text{'fabs'}, (\text{'b2f'}, a)), (\text{'b2f'}, a)) \]
WIP 1st-order tracking

Gather information about SSA values based on known properties of operation results.

- Analysis conceptually similar to existing 0th-order
  - Result of fabs must be $\geq 0$, etc.
  - $(value \geq 0) \times (value \leq 0) \Rightarrow$ result must be $\leq 0$
  - Analysis is on demand, but results are cached.

- Add simple predicates for use in nir_opt_algebraic

```python
(('fge', 'b(is_not_negative)', 'a(is_not_positive)'), True)
```
WIP 1\textsuperscript{st}-order tracking

Results so far are good

- Two main commits:
  - nir: Use value range analysis to eliminate tautological compares
  - nir: Use value range analysis to convert a fmin to an fsat

total instructions in shared programs: 15088355 \rightarrow 15027041 (-0.41%)
instructions in affected programs: 2823740 \rightarrow 2762426 (-2.17%)
helped: 10614
HURT: 2
helped stats (abs) min: 1 max: 294 \bar{x}: 5.78 x
helped stats (rel) min: 0.05% max: 58.33% \bar{x}: 3.23% x
HURT stats (abs) min: 6 max: 6 \bar{x}: 6.00 x
HURT stats (rel) min: 0.30% max: 0.30% \bar{x}: 0.30%
95\% mean confidence interval for instructions value: -5.99 -5.56
95\% mean confidence interval for instructions %-change: -3.32% -3.15%
Instructions are helped.
Existing 2\textsuperscript{nd}-order tracking

Tim Arceri's recently did some work to propagate compare results into branches.
Future 2\textsuperscript{nd}-order tracking

Infer value ranges from if-statement conditions, loop conditions, etc.

- Add NIR instructions similar to clang's / MSVC's "assume" built-in.
  - `ssa_4 = assume_gt ssa_3, 0`
  - Could expose directly in GLSL / SPIR-V

- Allows tracking of ranges after if-statements are replaced with bcsel

- Interferes with copy prop, CSE, etc.
  - Run optimization loop, strip assume instructions, run loop again?

- Or – hash values based on SSA and block ID
  - Harder to deal with bcsel
Questions?

https://cgit.freedesktop.org/~idr/mesa/log/?h=simple-range-analysis