Why does CUDA PTX have clz but no ctz, and CUDA headers have "fake ffs" but no fls?

Asked 7 years, 8 months ago Modified 7 years, 8 months ago Viewed 1k times

PTX is an intermediary representation for compiling C/C++ GPU code into, eventually, individual micro-architecture's SASS assembly language. Thus it is not supposed to be encumbered by specific holes/gaffs/flukes/idiosyncrasies in the actual instruction sets of specific nVIDIA GPU micro-architectures.

Now, PTX has an instruction for counting the number leading zeros in a register: `clz`. Yet - it lacks a corresponding `ctz` instruction, which counts the number trailing zeros. These operations are 'symmetric' and one would certainly expect to see either both or none in an instruction set - again, especially if its abstract and not bound to what's available on a specific piece of hardware. Popular CPU architectures have had both for many years.

Strangely enough, the CUDA header `device_functions.h` declares the function

```
 * \brief Find the position of the least significant bit set to 1 in a 32 bit
integer.
 *
 * [etc.]
 *
 * \return Returns a value between 0 and 32 inclusive representing the position
of the first bit set.
 * - __ffs(0) returns 0.
 */
__DEVICE_FUNCTIONS_DECL__ __device_builtin__ int                    __ffs(int
x);
```

This function:

- has almost the same semantics as count-trailing-zeros - only differing on an all-zero input.
- does *not* translate into a single PTX instruction, but rather two: bitwise negation, then a `clz`.
- is also missing its potential counterpart, `__fls` - find last set.

So, why is that? Why is an apparently obvious-to-have instruction missing from PTX, and a "fake builtin" that's almost identical to it present in the headers?

assembly cuda ptx

Share

Improve this question

Follow

edited Apr 23, 2017 at 10:26

asked Apr 23, 2017 at 9:39

einpoklum

131k • 76 • 408 • 832

Try and keep the gratuitous nonsense tag creation to a minimum. Tags are intended for search and question classification, not haiku. – talonmies Apr 23, 2017 at 10:22

@talonmies: (1) A tag regarding count-trailing-zeros (not just in CUDA) seems reasonable to have. (2) I hope that's not why you've downvoted. – einpoklum Apr 23, 2017 at 10:25

2 At least ARM doesn't have `ctz` but has `clz`. To count the number of trailing zeroes, you first use `rbit` to reverse bits and then `clz`. There is really no point in wasting opcode space for both rarely used functions when one can easily be expressed through the other. – fuz Apr 23, 2017 at 10:43

4 And note that `__ffs()` is probably present because it is part of POSIX. See ffs() in POSIX. – fuz Apr 23, 2017 at 10:44

1 @einpoklum: Let's put this into some perspective. There are 13.7 million questions on Stack Overflow today. Somehow, we have gotten by without a CTZ tag for almost 10 years. Somehow, I suspect we will get by without something which is obviously a meta tag (see stackoverflow.com/help/tagging) for another 10 years. If you feel strongly about this -- meta.stackoverflow.com is the place to go and have a meta discussion about tagging. You'll love it there, I'm sure. – talonmies Apr 23, 2017 at 11:14

1 Answer

Sorted by: Highest score (default)

3

Generally speaking, as with the x86 architecture, many features of CUDA and GPU architecture have accumulated organically, based on customer feedback and demands, rather than originating in some grand unified orthogonal design.

I personally added the `__ffs()` and `__ffsll()` device function intrinsics to CUDA. They were included because they represent useful bit-manipulation primitives and exactly match the `ffs()` functionality defined by POSIX.

For bit manipulations, in particular for the implementation of fixed-point operations and floating-point emulation, `CLZ` is a much more important operation than `CTZ`. Initially, I implemented `__clz()` and `__clz()` in CUDA as short emulation sequences. Hardware support for `CLZ` was added later. I was not part of the hardware architecture team, but I am reasonably sure the instruction was added based on customer feedback.

One of the major goals of PTX is to expose underlying hardware functionality in an abstracted form, as each GPU generation makes significant changes to the actual microarchitecture of the hardware. This virtual ISA is intended as a thin wrapper

around native instruction sets. Native GPU instruction sets are quite minimal. For example, there are no instructions for divisions. Simple hardware makes for smaller die area and higher core counts.

To provide a practical target for existing compilers (CUDA has used the Open64 and LLVM toolchains), some higher level operations were added to PTX, even though underlying hardware support is lacking. As this represents a software support burden, there is probably little incentive to add more such ops. Not all of the existing emulations are of the highest possible performance. During my tenure at NVIDIA I worked on optimizing the emulation sequences for the *most important* PTX operations.

CUDA users can submit requests for enhancements (such as inclusion of `CTZ` as a PTX operation) via the bug reporting mechanism.

Share Improve this answer Follow

answered Apr 23, 2017 at 19:53

njuffa

26k ● 5 ● 86 ● 140

Can you elaborate, if only briefly, on why adding an instruction, which is not radically different from existing ones - such as `ctz` - would be a software support burden? Anyway, +1 for the explanation. Also, you mentioned the goal of exposing underlying hardware functionality; why, then, do we have `__ffs()` but not `__fls()`, which is actually available in hardware? – einpoklum Apr 23, 2017 at 20:00

1 @einpoklum: Adding `CTZ` as a PTX-level instruction for which there is no hardware support would require maintaining emulation code across the about four different GPU architectures that must be supported at any given time (currently: 2.x, 3.x, 5.x, 6.x). I do not recall *any* user requests for `__fls()` and in fact I am not familiar with that function [where is that specified?]. My suggestion: File RFEs for what you want and see what happens. – njuffa Apr 23, 2017 at 20:04

I will file RFEs, but I wanted to understand whether such an RFE makes sense or whether there's something I ignoring. anyway, there's an FLO - find last one - SASS operation. See the answers to this question. – einpoklum Apr 23, 2017 at 20:12

No idea when `FLO` was added to the native GPU instruction set, and which GPU generations support it (sometimes instructions that used to be available disappear in future architectures; thus the need for PTX). `FLO` could be a low-throughput instruction, you may want to benchmark. – njuffa Apr 23, 2017 at 20:21