New issue

Jump to bottom

Is Zmmul a subset of M? #869

Open palmer-dabbelt opened this issue on Jul 14 · 16 comments

palmer-dabbelt commented on Jul 14 Member

This came up during the GNU tools meeting earlier today, we couldn't quite figure out how M and Zmmul are related to each other. It's not entirely clear whether Zmmul is a subset of the M extension, or a different extension that implements a subset of the functionality of the M extension. The wording is slightly different here, "The Zmmul extension implements the multiplication subset of the M extension." vs "he Zfhmin extension is a subset of the Zfh extension, ...". I'm not sure if those different wordings are meant to imply a different relationship between the extensions.

It seems like folks are leaning towards Zmmul being a subset of M, which means `-march=rv32im` will define `__riscv_zmmul` and `-march=rv32imzmmul` will be legal. That seems fine to me, just wanted to check it's what the spec is trying to say.

👍 1

scottj97 commented on Jul 14 • edited Contributor

It's not quite a proper subset because of `misa.M`. See discussions here and here.

Quoting myself:

> One difference between M and Zmmul is that the latter is not controlled by misa.M. If you have Zmmul then the multiply instructions are always enabled. If you have M then the multiply instructions are enabled iff misa.M==1. That's why saying "M implies Zmmul" is inaccurate -- because that would mean that misa.M only controls divide and not multiply.

But the toolchain probably doesn't care about that.

aswaterman commented on Jul 14 Member extension.

If people are finding it confusing, we can rephrase it to avoid the word "subset", but I think informal use of the word "subset" is going to crop up again as more overlapping extensions are defined over time.

👍 1

aswaterman commented on Jul 14 • edited Member

As a separate matter, it is the case that RV32IM_Zmmul is a legal ISA string (and it is equivalent to RV32IM, modulo the detail Scott mentioned that probably isn't important in the this context). It would then seem to follow logically that `__riscv_zmmul` is defined when M is implemented. It also seems seems pragmatic, since programmers wouldn't need to write as much boilerplate to determine if they can use the MUL* instructions.

👍 1

kito-cheng commented on Jul 14 • edited Member

I guess I more care about what is the canonical form of -march=rv64im_zmmul and -march=rv64im, toolchain need that to manage multi-lib, that might not super important from the ISA spec view, but I really want preventing from define toolchain's own canonical form to resolve this issue or make bunch of special rule in toolchain.

Having a complete canonical rule for those stuffs would be easier I think.

palmer-dabbelt commented on Jul 14 Member Author

If I'm understanding this correctly, for systems with writable and implemented misa:

- `rv32im` allows the M bit to be set/cleared, when cleared `mul` isn't a valid instruction.
- `rv32i_zmmul` doesn't allow the M bit to be set/cleared, and `mul` is always a valid instruction.
- `rv32im_zmmul` allows the M bit to be set/cleared, when cleared `mul` is valid but `div` is not.

aswaterman commented on Jul 14 • edited Member

> I guess I more care about what is the canonical form of -march=rv64im_zmmul and -march=rv64im,

RV64IM is canonical even though both are legal. No one *wants* to write Zmmul just for redundancy...

scottj97 commented on Jul 14 Contributor

> If I'm understanding this correctly, for systems with writable and implemented misa:
>
> - `rv32im` allows the M bit to be set/cleared, when cleared `mul` isn't a valid instruction.
> - `rv32i_zmmul` doesn't allow the M bit to be set/cleared, and `mul` is always a valid instruction.
> - `rv32im_zmmul` allows the M bit to be set/cleared, when cleared `mul` is valid but `div` is not.

I suppose that's correct, with the subtle qualifier that "writable and implemented misa" does not necessarily imply that every bit of misa is modifiable, e.g. an rv32im system might have misa.M hardwired to 1 even when other bits are writable.

But does the toolchain really care about what bits of misa are writable, and/or what features those bits control? If I tell the compiler to target rv32im then it should assume that mul/div are both valid. If I'm planning to run on a system where I've written misa.M to 0, then I should compile with rv32i, not rv32im.

aswaterman commented on Jul 14 Member

> But does the toolchain really care about what bits of misa are writable

No, it’s a distraction. If you tell the toolchain you have M, you better have M enabled, and at that point M_Zmmul == M.

kasanovic commented on Jul 14 Collaborator

Whether misa.M is writable is a separate M-mode option that should not affect software, and certainly not <M-mode software.
We don't have compiler flags to indicate QEMU's option settings in case we're running under QEMU - writable misa is similar.

palmer-dabbelt commented on Jul 14 Member Author

> Whether misa.M is writable is a separate M-mode option that should not affect software, and certainly not <M-mode software. We don't have compiler flags to indicate QEMU's option settings in case we're running under QEMU - writable misa is similar.

I don't think anyone was planning on having compiler flags for the presence of CSR bits (aside from those that filter in via ISA extensions), the real goal here was to figure out if Zmmul was implied by M. It sounds like the answer to that is no, at least in the general case, as there's this difference in behavior between the two related to misa.M.

kasanovic commented on Jul 15 Collaborator

Zmmul *is* implied by M - it is a proper subset of the instructions in M. Here the ISA string is being used to tell the compiler what instructions the compiler can use. I really don't think you want to encode possible ISA-related EE differences beyond what the instructions do. The dependence is the other way around - the binary tells the EE what instructions it uses, and the EE has to be set up to support that. As for #defines/paths etc. - what software is going to want to do different things based on whether misa.M is writable?

palmer-dabbelt commented on Jul 15 Member Author

Sorry, now I'm confused again. You're saying '"Zmmul is implied by M", but Scott's saying in #869 (comment) that there's a behavioral difference between `rv32im` and `rv32im_zmmul`.

scottj97 commented on Jul 15 Contributor

That behavior difference is not anything that the toolchain would care about. We're telling the toolchain what instructions are available -- not what instructions might possibly be dynamically disabled.

In this context, it's fair to say that Zmmul is implied by M.

👍 1

allenjbaum commented on Jul 15

Note that if you implement an RV32IM, and misa.M is read/write, and you
set misa.M to 0
- that doesn't necessarily mean that a MUL instruction won't multiple two
source registers and put it into a destination register.
It might trap, it might multiply, it might do something custom (e.g. a
custom scaled multiply)?

Zmmul doesn't have that subtlety; if it is present, then executing a MUL op
better perform the multiply.

The case with a RV32IM_Zmmul with misa.m begin RW does effectively allow
the Divide instructions to be... undefined.
It would be pretty weird to do that and not have those opcodes do something
custom in that case, but it is totally implementation dependent.
…

And I would not say that Zmmul is implied by M; I would say that Zmmul ops
are implied by M to be a bit more precise.
And pedantic, which is my new middle name.

On Fri, Jul 15, 2022 at 7:45 AM Allen Baum ***@***.***>
wrote:
…

kasanovic commented on Jul 16 Collaborator

Zmmul unprivileged ISA string should not imply anything about privileged support for disabling/enabling features.
Zmmul-only hardware might have a "chicken bit" that turns off multiplies.
We would need a separate privileged Sm* string to indicate possible M-mode switching variants (including writable misa.M).
So M=>Zmmul.

alistair23 pushed a commit to alistair23/qemu that referenced this issue on Jul 19

`RISC-V: Allow both Zmmul and M` … `cdf5f68`

alistair23 pushed a commit to alistair23/qemu that referenced this issue on Jul 21

`RISC-V: Allow both Zmmul and M` … `b43a94d`

alistair23 pushed a commit to alistair23/qemu that referenced this issue on Jul 25

`RISC-V: Allow both Zmmul and M` … `3007d1a`

alistair23 pushed a commit to alistair23/qemu that referenced this issue on Jul 28

`RISC-V: Allow both Zmmul and M` … `44602af`

Assignees

No one assigned

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

6 participants