bug-binutils@gnu.org

[Bug gas/13215] New: ARM Cortex M3 strexh strexb instructions with same registers generates error

hazelnusse at gmail dot com 12 years ago

Permalink

http://sourceware.org/bugzilla/show_bug.cgi?id=13215

Bug #: 13215
Summary: ARM Cortex M3 strexh strexb instructions with same
registers generates error
Product: binutils
Version: 2.21
Status: NEW
Severity: critical
Priority: P2
Component: gas
AssignedTo: ***@sources.redhat.com
ReportedBy: ***@gmail.com
Classification: Unclassified

Created attachment 5939
--> http://sourceware.org/bugzilla/attachment.cgi?id=5939
example of ARM provided code that makes use of strexh strexb with same
registers

In gas version 2.21.53, when compiling for the Cortex-M3 with -mcpu=cortex-m3
-march=armv7-m, the following instructions generate assembler errors:

strexh r0, r0, [r1]
strexb r0, r0, [r1]

The error messages are:
Error: registers may not be the same -- `strexh r0,r0,[r1]'
Error: registers may not be the same -- `strexb r0,r0,[r1]'

However, according to the ARM documentation of STREX for the Cortex M3 [0],
only the word sized (32-bit) version of the instruction strex has the
restriction that the three registers be unique:

Selection Permalink:
https://narkive.com/493kemg7:1.1093.59

for STREX, Rd must be different from both Rt and Rn

This problem shows up, for example, when trying to compile vendor provided
peripheral libraries that make use of CMSIS. In particular, the core_cm3.[ch]
of CMSIS v1.3, and core_cmInstr.h of CMSIS v2.0 have all of these instructions.
Curiously, even strex r0, r0, [r1] shows up in CMSIS, even though this is
explicitly restricted in [0]. So perhaps even some of the CMSIS code is
breaking the specification of the ARM documentation, not sure.

[0] --
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0552a/BABFFBJB.html

--
Configure bugmail: http://sourceware.org/bugzilla/userprefs.cgi?tab=email
------- You are receiving this mail because: -------
You are on the CC list for the bug.

hazelnusse at gmail dot com 12 years ago

http://sourceware.org/bugzilla/show_bug.cgi?id=13215

Luke Peterson <hazelnusse at gmail dot com> changed:

What |Removed |Added

Status|NEW |RESOLVED
Resolution| |INVALID

--- Comment #1 from Luke Peterson <hazelnusse at gmail dot com> 2011-09-22 16:22:03 UTC ---
A closer read of the ARMv7m reference manual confirms that for any of the strex
instructions (word, half word, byte), if the register are the same, the result
is undefined. So James Greenhalgh's patch is correct, and this bug is invalid.

DaniBoy 12 years ago

To solve this problem with ARM cortex m3 I changed the supplied CMSIS file
core_cm3.c

I'm now using:

```
uint32_t __STREXH(uint16_t value, uint16_t *addr)
{
//uint32_t result=0;
register uint32_t result asm ("r2");
__ASM volatile ("strexh %0, %2, [%1]" : "=r" (result) : "r" (addr), "r"
(value) );
return(result);
}
...
uint32_t __STREXB(uint8_t value, uint8_t *addr)
{
//uint32_t result=0;
register uint32_t result asm ("r2");
__ASM volatile ("strexb %0, %2, [%1]" : "=r" (result) : "r" (addr), "r"
(value) );
return(result);
}
```

instead of:

```
uint32_t __STREXH(uint16_t value, uint16_t *addr)
{
uint32_t result=0;
__ASM volatile ("strexh %0, %2, [%1]" : "=r" (result) : "r" (addr), "r"
(value) );
return(result);
}
...
uint32_t __STREXB(uint8_t value, uint8_t *addr)
```

```
{
uint32_t result=0;
__ASM volatile ("strexb %0, %2, [%1]" : "=r" (result) : "r" (addr), "r"
(value) );
return(result);
}
```

...

--
View this message in context: http://old.nabble.com/-Bug-gas-13215--New%3A-ARM-Cortex-M3-strexh-strexb-instructions-with-same-registers-generates-error-tp32516436p33011958.html
Sent from the Gnu - Binutils - Bugs mailing list archive at Nabble.com.

Hans-Peter Nilsson 12 years ago

> *Post by DaniBoy*
> To solve this problem with ARM cortex m3 I changed the supplied CMSIS file
> core_cm3.c
> uint32_t __STREXH(uint16_t value, uint16_t *addr)
> {
> //uint32_t result=0;
> register uint32_t result asm ("r2");
> __ASM volatile ("strexh %0, %2, [%1]" : "=r" (result) : "r" (addr), "r"
> (value));
> return(result);

With the base version //-commented out and replaced with that on the next line? Wrong or at least suboptimal (I don't recall if GCC is _documented_ to be free to use r2 as a _source_ operand when it's an asm-register-declared _output_ operand as above but I think it's free to do that).

Write "=&r" instead of "=r" to get the result in a register not overlapping any of the source operands.

brgds, H-P

DaniBoy 12 years ago

Yep, I commented the base version (uint32_t result=0;).
Because it was offending the GNU assembler with:

- strexh r0, r0, [r1]
- strexb r0, r0, [r1]

Then I replaced it by:
- register uint32_t result asm ("r2");

According to ARM documentation, for STREX, Rd must be different from both Rt and Rn.

now the compiler output is:

- strexh r2, r0, [r1]
- strexb r2, r0, [r1]

With the above assembler code, the GNU assembler (GAS) doesn't complain anymore.
I believe this is a great workaround for that problem.

--
View this message in context: http://old.nabble.com/-Bug-gas-13215--New%3A-ARM-Cortex-M3-strexh-strexb-instructions-with-same-registers-generates-error-

Benjamin Green 12 years ago

I had the same problem with my cross compiler, as you can see the code translates into the same number of instructions.

DaniBoy's solution:

register uint32_t result asm ("r2");

00000000 <__STREXB>:
0: e8c1 0f42 strexb r2, r0, [r1]
4: 4610 mov r0, r2
6: 4770 bx lr

Hans-Peter Nilsson's solution:

__ASM volatile ("strexb %0, %2, [%1]" : "=&r" (result) : "r" (addr), "r" (value));

00000000 <__STREXB>:
0: 4603 mov r3, r0
2: e8c1 3f50 strexb r0, r3, [r1]
6: 4770 bx lr

r0, r1, r2 and r3 are all argument registers and r0 is the integer function result register (alternate names are a1, a2, a3 and a4) so I cannot see an obvious problem with either of these solutions. I think perhaps I would lean twards Hans-Peter Nilsson's solution as if this function were inlined the compiler may be able to optimise register usage. I also did not check the impact ordering would have on the instruction pipeline.

This bug is invalid (as Bugzilla says) and also more concern to ARM Cortex-M3 users... sorry for the clutter.

Benjamin.