Ruby master - Bug #5244

Continuation causes Bus Error on Debian sparc

08/28/2011 07:48 PM - lucas (Lucas Nussbaum)

Status:	Closed		
Priority:	Normal		
Assignee:	ngoto (Naohisa Goto)		
Target version:			
ruby -v:	-	**Backport:**	2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

Hi,

```
$ ./miniruby -I./lib -I. -I.ext/common ./tool/runruby.rb --extout=.ext -rcontinuation -e 'callcc { |c| c.call }'
-e:1: [BUG] Bus Error
ruby 1.9.3dev (2011-08-26) [sparc-linux]

-- Control frame information -----------------------------------------------
c:0004 p:---- s:0009 b:0009 l:000008 d:000008 CFUNC  :callcc
c:0003 p:0009 s:0006 b:0006 l:000fcc d:001d74 EVAL   -e:1
c:0002 p:---- s:0004 b:0004 l:000003 d:000003 FINISH
c:0001 p:0000 s:0002 b:0002 l:000fcc d:000fcc TOP

-- Ruby level backtrace information ----------------------------------------
-e:1:in <main>'
-e:1:incallcc'

-- C level backtrace information -------------------------------------------
Bus error
```

gdb says:

```
(gdb) run -I./lib -I. -I.ext/common ./tool/runruby.rb --extout=.ext -rcontinuation -e 'callcc { |c| c.call }'
Starting program: /home/lucas/ruby1.9.1-1.9.3~preview1+svn33077/miniruby -I./lib -I. -I.ext/common ./tool/runruby.rb --extout=.ext
-rcontinuation -e 'callcc { |c| c.call }'
[Thread debugging using libthread_db enabled]
[New Thread 0xf7fc7b70 (LWP 31418)]
[Thread 0xf7fc7b70 (LWP 31418) exited]
process 31417 is executing new program: /home/lucas/ruby1.9.1-1.9.3~preview1+svn33077/ruby1.9.1
[Thread debugging using libthread_db enabled]
[New Thread 0xf79e5b70 (LWP 31419)]

Program received signal SIGBUS, Bus error.
0xf7f4d304 in cont_capture (stat=Cannot access memory at address 0x49
) at cont.c:439
439     if (ruby_setjmp(cont->jmpbuf)) {

(gdb) print cont
Cannot access memory at address 0xfffffff9
```

Associated revisions

Revision 9c497531 - 10/20/2011 04:09 AM - nobu (Nobuyoshi Nakada)

- include/ruby/defines.h (flush_register_windows): use software trap on Debian Sparc 32-bit userspace. [Bug #5244]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@33492 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 33492 - 10/20/2011 04:09 AM - nobu (Nobuyoshi Nakada)

- include/ruby/defines.h (flush_register_windows): use software trap on Debian Sparc 32-bit userspace. [Bug #5244]

Revision 33492 - 10/20/2011 04:09 AM - nobu (Nobuyoshi Nakada)

- include/ruby/defines.h (flush_register_windows): use software trap on Debian Sparc 32-bit userspace. [Bug #5244]

Revision 33492 - 10/20/2011 04:09 AM - nobu (Nobuyoshi Nakada)

- include/ruby/defines.h (flush_register_windows): use software trap on Debian Sparc 32-bit userspace. [Bug #5244]

Revision 33492 - 10/20/2011 04:09 AM - nobu (Nobuyoshi Nakada)

- include/ruby/defines.h (flush_register_windows): use software trap on Debian Sparc 32-bit userspace. [Bug #5244]

Revision 33492 - 10/20/2011 04:09 AM - nobu (Nobuyoshi Nakada)

- include/ruby/defines.h (flush_register_windows): use software trap on Debian Sparc 32-bit userspace. [Bug #5244]

Revision 33492 - 10/20/2011 04:09 AM - nobu (Nobuyoshi Nakada)

- include/ruby/defines.h (flush_register_windows): use software trap on Debian Sparc 32-bit userspace. [Bug #5244]

Revision fa00b651 - 11/15/2011 04:42 AM - ngoto (Naohisa Goto)

- include/ruby/defines.h (FLUSH_REGISTER_WINDOWS): move sparc asm code to a separete file sparc.c for preventing inlining optimization. Patched by Jurij Smakov. [Bug #5244] [ruby-core:40685]
- sparc.c (rb_sparc_flush_register_windows): ditto.
- configure.in: ditto.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@33757 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 33757 - 11/15/2011 04:42 AM - ngoto (Naohisa Goto)

- include/ruby/defines.h (FLUSH_REGISTER_WINDOWS): move sparc asm code to a separete file sparc.c for preventing inlining optimization. Patched by Jurij Smakov. [Bug #5244] [ruby-core:40685]
- sparc.c (rb_sparc_flush_register_windows): ditto.
- configure.in: ditto.

Revision 33757 - 11/15/2011 04:42 AM - ngoto (Naohisa Goto)

- include/ruby/defines.h (FLUSH_REGISTER_WINDOWS): move sparc asm code to a separete file sparc.c for preventing inlining optimization. Patched by Jurij Smakov. [Bug #5244] [ruby-core:40685]
- sparc.c (rb_sparc_flush_register_windows): ditto.
- configure.in: ditto.

Revision 33757 - 11/15/2011 04:42 AM - ngoto (Naohisa Goto)

- include/ruby/defines.h (FLUSH_REGISTER_WINDOWS): move sparc asm code to a separete file sparc.c for preventing inlining optimization. Patched by Jurij Smakov. [Bug #5244] [ruby-core:40685]
- sparc.c (rb_sparc_flush_register_windows): ditto.
- configure.in: ditto.

Revision 33757 - 11/15/2011 04:42 AM - ngoto (Naohisa Goto)

- include/ruby/defines.h (FLUSH_REGISTER_WINDOWS): move sparc asm code to a separete file sparc.c for preventing inlining optimization. Patched by Jurij Smakov. [Bug #5244] [ruby-core:40685]
- sparc.c (rb_sparc_flush_register_windows): ditto.
- configure.in: ditto.

Revision 33757 - 11/15/2011 04:42 AM - ngoto (Naohisa Goto)

- include/ruby/defines.h (FLUSH_REGISTER_WINDOWS): move sparc asm code to a separete file sparc.c for preventing inlining optimization. Patched by Jurij Smakov. [Bug #5244] [ruby-core:40685]
- sparc.c (rb_sparc_flush_register_windows): ditto.
- configure.in: ditto.

Revision 33757 - 11/15/2011 04:42 AM - ngoto (Naohisa Goto)

- include/ruby/defines.h (FLUSH_REGISTER_WINDOWS): move sparc asm code to a separete file sparc.c for preventing inlining optimization. Patched by Jurij Smakov. [Bug #5244] [ruby-core:40685]
- sparc.c (rb_sparc_flush_register_windows): ditto.
- configure.in: ditto.

Revision 31cf6801 - 01/03/2012 10:12 AM - kosaki (Motohiro KOSAKI)

merge revision(s) %s: 33757:33758

```
    * include/ruby/defines.h (FLUSH_REGISTER_WINDOWS): move sparc asm code
      to a separete file sparc.c for preventing inlining optimization.
      Patched by Jurij Smakov. [Bug #5244] [ruby-core:40685]
    * sparc.c (rb_sparc_flush_register_windows): ditto.
    * configure.in: ditto.
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_1_9_3@34199 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 09/06/2011 05:46 AM - jurij (Jurij Smakov)

I poked at it a bit. While I don't understand fully what's going on, it looks like it's using _setjmp/longjmp, and one thing in setjmp page caught my eye: "setjmp() saves the stack context/environment in env for later use by longjmp(3). The stack context will be invalidated if the function which called setjmp() returns." I think that such invalidation takes place here. Function rb_callcc (in cont.c) does the following:

rb_callcc(VALUE self)
{
volatile int called;
volatile VALUE val = cont_capture(&called);

```
if (called) {
    return val;
}
else {
    return rb_yield(val);
}
```

}

In cont_capture() the _setjmp is invoked via

```
if (ruby_setjmp(cont->jmpbuf)) {
    volatile VALUE value;

    value = cont->value;
    if (cont->argc == -1) rb_exc_raise(value);
    cont->value = Qnil;
    *stat = 1;
    return value;
}
else {
    *stat = 0;
    return cont->self;
}
```

So, after invoking _setjmp, cont_capture returns which, according to the man page, should invalidate the saved stack context. Later, when something in the rb_yield(val) call chain tries to do the longjmp, we arrive at the location where _setjmp was originally called (in cont_capture) with a smashed stack, resulting in a SIGBUS.

The fact that it works for other arches (and even on sparc, if I rebuild everything with -O0 instead of -O2) is somewhat surprising, but it might be that it just working "by accident" in most cases (i.e. the saved stack is preserved, even though it's not guaranteed).

#2 - 09/06/2011 10:15 AM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to ngoto (Naohisa Goto)

#3 - 09/06/2011 05:38 PM - ngoto (Naohisa Goto)

- Status changed from Assigned to Feedback

This cannot be reproduced on Solaris10 on sparc, with both Sun cc and gcc 4.4.3.

(using svn ruby_1_9_3 branch r33165)

The bug might depend on OS (kernel), gcc and/or libc.
Which version of OS (kernel), gcc, and libc do you use?
Could you try rebuilding with -O0 option?

#4 - 09/06/2011 08:50 PM - lucas (Lucas Nussbaum)

You can find a full build log at
https://buildd.debian.org/status/fetch.php?pkg=ruby1.9.1&arch=sparc&ver=1.9.3~preview1%2Bsvn33077-3&stamp=1314689360

Kernel: Linux lebrun 2.6.32-5-sparc64-smp #1 SMP Tue Jun 14 12:44:14 UTC 2011 sparc GNU/Linux
GNU Libc 2.13
GCC 4.6.1

Jurij Smakov said above that it works fine when built with -O0

#5 - 09/07/2011 06:43 AM - jurij (Jurij Smakov)

My kernel and toolchain versions are slightly different because I'm running Debian sid, but I don't think that's the issue here. The main problem is that the approach used in implementation of continuations is not guaranteed to work (even though it *may* work in vast majority of cases). Quoting http://en.wikipedia.org/wiki/Setjmp.h#Caveats_and_limitations :

"If the function in which setjmp was called returns, it is no longer possible to safely use longjmp with the corresponding jmp_buf object. This is because the stack frame is invalidated when the function returns. Calling longjmp restores the stack pointer, which—because the function returned—would point to a non-existent and potentially overwritten/corrupted stack frame.[4][5]"

I've verified that ruby_setjmp() translates to a simple _setjmp() in preprocessed code, and after calling it the function returns immediately.

#6 - 09/07/2011 01:23 PM - shugo (Shugo Maeda)

- ruby -v changed from 1.9.3 to -

Hi,

2011/9/7 Jurij Smakov jurij@wooyd.org:

> "If the function in which setjmp was called returns, it is no longer possible to safely use longjmp with the corresponding jmp_buf object. This is because the stack frame is invalidated when the function returns. Calling longjmp restores the stack pointer, whichâ€”because the function returnedâ€"would point to a non-existent and potentially overwritten/corrupted stack frame.[4][5]"
>
> I've verified that ruby_setjmp() translates to a simple _setjmp() in preprocessed code, and after calling it the function returns immediately.

Ruby's callcc copies stack to heap, then calls setjmp(). Stack frames
are restored from the copy before longjmp().

--
Shugo Maeda

#7 - 09/08/2011 08:32 PM - ngoto (Naohisa Goto)

The bug does not occur with older Debian sparc running on qemu.

1. download image http://people.debian.org/~aurel32/qemu/sparc/debian_etch_sparc_small.qcow2
2. qemu-system-sparc -hda debian_etch_sparc_small.qcow2 -M SS-10 -m 1G
3. change /etc/apt/sources.list
4. install subversion and many packages by using aptitude
5. manually install yaml-0.1.4.tar.gz and libffi-3.0.10.tar.gz
6. svn co http://svn.ruby-lang.org/repos/ruby/branches/ruby_1_9_3 (fetched revision r33190)
7. autoconf; ./configure --prefix=/home/user/ruby/193 optflags="-O2"
8. make
9. ./miniruby -I./lib -I. -I.ext/common ./tool/runruby.rb --extout=.ext -rcontinuation -e 'callcc { |c| c.call }'
10. finished with exit code 0

```
$ uname -a
Linux debian-sparc 2.6.18-6-sparc32 #1 Fri Dec 12 16:29:52 UTC 2008 sparc GNU/Linux
$ dpkg -l gcc libc6
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name           Version           Description
+++-==============-==================-============================================
ii  gcc            4.1.1-15          The GNU C compiler
ii  libc6          2.3.6.ds1-13etch10 GNU C Library: Shared libraries
```

The bug might be specific to the svn revision 33077 and might have already been fixed in the latest svn revision.

#8 - 09/10/2011 03:51 PM - lucas (Lucas Nussbaum)

The problem is that it's quite hard to investigate this using qemu, because Debian dropped support for sparcv8 after etch, and qemu doesn't have working support for anything > sparcv8.

#9 - 09/10/2011 04:45 PM - lucas (Lucas Nussbaum)

I took a look on a Debian porterbox.

When building everything with -O2, except cont.c with -O0, it works.
With cont.c built with -O1, it fails.

However, what I don't understand is that building with -O0 and all the optimizations that are normally enabled with -O1 (determined using gcc -Q -O1 --help=optimizers), it works.

#10 - 09/10/2011 08:44 PM - lucas (Lucas Nussbaum)

It fails with gcc 4.4 and 4.5 too (in addition to 4.6).

#11 - 10/07/2011 12:33 AM - jurij (Jurij Smakov)

I looked at it some more (ruby1.9.1-1.9.3~preview1+svn33236 now), and tried to figure out what goes wrong by comparing the binaries compiled with -O0 and -O2. The call to ruby_longjmp does not look suspicious, I've verified that in both cases comp->setjmpbuf gets correctly passed to the function. Eventually we get to the point where the actual long jump is performed, which is **longjmp in eglibc-2.13/sysdeps/sparc/sparc32/** longjmp.S. Actual jumping is done with the following code:

```
LOC(thread):
/*
* Do a "flush register windows trap".  The trap handler in the
* kernel writes all the register windows to their stack slots, and
* marks them all as invalid (needing to be sucked up from the
* stack when used).  This ensures that all information needed to
* unwind to these callers is in memory, not in the register
* windows.
/
ta  ST_FLUSH_WINDOWS
#ifdef PTR_DEMANGLE
ld  ENV(g1,JB_PC), %g5 / Set return PC. /
ld  ENV(g1,JB_SP), %g1 / Set saved SP on restore below. /
PTR_DEMANGLE2 (%o7, %g5, %g4)
PTR_DEMANGLE2 (%fp, %g1, %g4)
#else
ld  ENV(g1,JB_PC), %o7 / Set return PC. /
ld  ENV(g1,JB_SP), %fp / Set saved SP on restore below. /
#endif
sub %fp, 64, %sp    / Allocate a register frame. /
st  %g3, RW_FP  / Set saved FP on restore below. /
retl
restore %g2, 0, %o0    / Restore values from above register frame. */
```

I've verified that in both cases the value of %o7 which is used by retl (it's essentially %o7 + 8) is correct, pointing to the address from where setjmp has been previously called. However, in the optimized case (built with -O2) something goes wrong with the register frame restore (which is executed in retl delay slot), and we jump to the correct address, but with an obviously broken value of 0x5 in %fp, which eventually leads to a SIGBUS once we start dereferencing memory with it. I'll need to do quite a bit of reading here to understand why the broken values end up on the register frame, so it may take a while.

#12 - 10/14/2011 08:19 AM - jurij (Jurij Smakov)

Discussion of this issue is ongoing in this thread on sparclinux mailing list: http://marc.info/?t=131806608400002&r=1&w=2.

#13 - 10/19/2011 08:05 AM - jurij (Jurij Smakov)

- File ruby.patch added

I think we figured it out. The problem arises in cont_save_machine_stack() function, where the register windows are flushed using 'flushw' assembler instruction, and the machine stack is then saved by memcpy'ing it from cont->machine_stack_src to cont->machine_stack. However, 'flushw' does not flush the current register window, so we end up copying incorrect memory contents, because the source address lies withing the last stack frame. For a detailed analysis see my message:

http://article.gmane.org/gmane.linux.ports.sparc/15410

and David Miller's suggestions for fixing it:

http://article.gmane.org/gmane.linux.ports.sparc/15411

If you decide to follow the first suggestion (replacing 'flushw' with 'ta 0x03'), attached patch implements it. I've just ran a successful build with it applied on my SunBlade 1000 machine.

#14 - 10/19/2011 11:29 AM - nobu (Nobuyoshi Nakada)

- Status changed from Feedback to Open

At r3285, defined(**FreeBSD**) was lost.
I have no idea if 'flushw' is preferable to 'ta 0x03' on FreeBSD.

#15 - 10/19/2011 02:28 PM - nobu (Nobuyoshi Nakada)

knu says that 'flushw' is correct for SparcV9, but not 'ta 0x03'.
And your platform seems 32bit, right?
Then why defined(**sparc_v9**) || defined(**sparcv9)** || **defined(**arch64__) is true?

Can't you show the result from:
$ gcc -E -dM -xc /dev/null | grep -i -e sparc -e arch64

#16 - 10/19/2011 04:11 PM - jurij (Jurij Smakov)

My machine is UltraSparc III based, so it's a v9 and 64-bit. For historical reasons though Debian is using 64-bit kernel and 32-bit userspace:

jurij@debian:~$ gcc -E -dM -xc /dev/null | grep -i -e sparc -e arch64
#define sparc 1
#define **sparc** 1
#define **sparc 1**
#define __sparc_v9 1
jurij@debian:~$

For the purposes of the continuation code it's not appropriate to say that 'flushw' is correct for sparcv9, as 'flushw' has slightly different effect compared to 'ta 0x03' (at least, on sparc/linux). It appears that Ruby wants to save the entire stack, including the current stack frame, and relies on the contents of current register window being flushed before that. Well, 'flushw' is going to flush all windows *except* the current one, this behavior is described in Sparc Architecture Manual (version 9). On the other hand, 'ta 0x03' is a software trap, which flushes all register windows of the process invoking the trap (including the current one), and that's the behavior wanted here.

#17 - 10/20/2011 11:35 AM - ngoto (Naohisa Goto)

I think a possible workaroud to distinguish 32-bit on Debian Sparc (or on Sparc Linux) is to check SIZEOF_VOIDP (value of sizeof(void *) set by configure) in addition to sparc specific macros.

FYI, on Solars10, Sun compiler (Sun Studio, Oracle Solaris Studio) defines __sparcv9 only when generating 64-bit code for SPARC V9.
http://developers.sun.com/sunstudio/documentation/ss12u1/mr/READMEs/c++_faq.html#Vers6

GCC on Solaris 10 also does so.

$ gcc-4.4 -E -dM -xc /dev/null | grep -i -e sparc -e arch64
#define sparc 1
#define **sparc** 1
#define **sparcv8 1**
#define __sparc 1
$ gcc-4.4 -m64 -E -dM -xc /dev/null | grep -i -e sparc -e arch64
#define sparc 1
#define __sparc 1
#define **sparcv9 1**
#define __sparc 1
#define __arch64 1

#18 - 10/20/2011 01:09 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r33492.
Lucas, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- include/ruby/defines.h (flush_register_windows): use software trap on Debian Sparc 32-bit userspace. [Bug #5244]

#19 - 10/22/2011 08:04 AM - jurij (Jurij Smakov)

Sorry, but this is not a proper fix. While it will fix the immediate problem for Debian, other systems will still be affected. Out of curiosity I tried building the latest svn snapshot (including this fix) on a freebsd/sparc64 system, and, sure enough, it still crashes there:

```
$ uname -a
FreeBSD free.wooyd.org 8.2-RELEASE FreeBSD 8.2-RELEASE #0: Thu Feb 17 06:57:44 UTC 2011     root@araz.cse.buffalo.edu
:/usr/obj/usr/src/sys/GENERIC  sparc64
$ export
RUBYLIB=/usr/home/jurij/snapshot:/usr/home/jurij/snapshot/.ext/common:/usr/home/jurij/snapshot/.ext/sparc64-freebsd8.2:/usr/home/jurij/snapshot/lib
$ ./ruby -rcontinuation -e 'callcc { |c| c.call }'
-e:1: [BUG] Segmentation fault
ruby 2.0.0dev (2011-10-22 trunk 33503) [sparc64-freebsd8.2]

-- Control frame information -----------------------------------------------
c:0004 p:---- s:0009 b:0009 l:000008 d:000008 CFUNC  :callcc
c:0003 p:0009 s:0006 b:0006 l:0007f8 d:000fb8 EVAL   -e:1
c:0002 p:---- s:0004 b:0004 l:000003 d:000003 FINISH
c:0001 p:0000 s:0002 b:0002 l:0007f8 d:0007f8 TOP

-- Ruby level backtrace information ----------------------------------------
-e:1:in <main>'
-e:1:incallcc'

-- Other runtime information -----------------------------------------------
```

- Loaded script: -e
- Loaded features:

  ```
  0 enumerator.so
  1 /usr/home/jurij/snapshot/.ext/sparc64-freebsd8.2/enc/encdb.so
  2 /usr/home/jurij/snapshot/.ext/sparc64-freebsd8.2/enc/trans/transdb.so
  3 /usr/home/jurij/snapshot/lib/rubygems/defaults.rb
  4 /usr/home/jurij/snapshot/rbconfig.rb
  5 /usr/home/jurij/snapshot/lib/rubygems/deprecate.rb
  6 /usr/home/jurij/snapshot/lib/rubygems/exceptions.rb
  7 /usr/home/jurij/snapshot/lib/rubygems/custom_require.rb
  8 /usr/home/jurij/snapshot/lib/rubygems.rb
  9 /usr/home/jurij/snapshot/.ext/sparc64-freebsd8.2/continuation.so
  ```

```
[NOTE]
You may have encountered a bug in the Ruby interpreter or extension libraries.
Bug reports are welcome.
For details: http://www.ruby-lang.org/bugreport.html

Abort trap (core dumped)
$ gdb
GNU gdb 6.1.1 [FreeBSD]
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "sparc64-marcel-freebsd".
(gdb) file ./ruby
Reading symbols from ./ruby...done.
(gdb) run -rcontinuation -e 'callcc { |c| c.call }'
Starting program: /usr/home/jurij/snapshot/ruby -rcontinuation -e 'callcc { |c| c.call }'

Program received signal SIGSEGV, Segmentation fault.
0x0000000040a41368 in __sparc_utrap_install () from /lib/libc.so.7
(gdb) bt
#0  0x0000000040a41368 in __sparc_utrap_install () from /lib/libc.so.7
#1  0x0000000040a4148c in __sparc_utrap_install () from /lib/libc.so.7
#2  0x0000000040a41730 in __sparc_utrap_install () from /lib/libc.so.7
#3  0x0000000040a40f6c in __sparc_utrap_install () from /lib/libc.so.7
#4  0x00000000002489b4 in cont_capture (stat=Error accessing memory address 0x882: Bad address.
) at cont.c:440
Previous frame inner to this frame (corrupt stack?)
(gdb) The program is running.  Exit anyway? (y or n) y
$
```

Unfortunately, using 'ta 0x03' instead of 'flushw' is not an option, as it causes an illegal instruction trap.

#20 - 11/03/2011 07:58 AM - jurij (Jurij Smakov)

- File flush_windows.patch added

Attached is a patch for this problem, fixing the issue by moving the windows-flushing instruction into a separate function on sparc. It will still use flushw any sparcv9-capable machine irrespective of the OS. I've verified that it fixes the crashes on both Debian/sparc/unstable and FreeBSD/sparc64/8.2. Thanks a lot to David Miller for explaining how to do it correctly.

#21 - 11/12/2011 03:10 AM - lucas (Lucas Nussbaum)

Dear Ruby developers,

Could you follow up on this issue?
The fix that was commited is not correct, as explained in comment #19. A correct fix is in comment #20.

Also, could you backport this to the ruby1_9_3 branch?

#22 - 11/12/2011 06:34 AM - kosaki (Motohiro KOSAKI)

- Status changed from Closed to Assigned

Goto-san, ping?

#23 - 11/12/2011 11:08 AM - ngoto (Naohisa Goto)

Sorry for delay. I'll try on it Solaris10 with SUN cc, Fujitsu fcc and gcc, with 32 and 64-bit compiler options.

#24 - 11/13/2011 03:14 PM - ngoto (Naohisa Goto)

In Solaris10, with Sun Studio 11 cc, with 64-bit compile option -xarch=v9, compile error occur with the patch.

compiling sparc.c
"sparc.c", line 21: syntax error before or at: :
cc: acomp failed for sparc.c
make: *** [sparc.o] Error 2

It seems that the description ("flushw" : : : "%o7") is not portable.

#25 - 11/14/2011 05:38 AM - jurij (Jurij Smakov)

Does it work if you replace ("flushw" : : : "%o7") with just ("flushw")? If it is, then it just has to be protected by #ifdef **GNUC**, i.e. the body of the function should look something like this (untested):

void flush_sparc_register_windows()
{
asm
#if defined(**sparcv9) || defined(**sparc_v9__)

ifdef GNUC

`__volatile__ ("flushw" : : : "%o7")`

else

`("flushw")`

endif /* GNUC */

#else
("ta 0x03")
#endif
;
}

As long as the asm instructions remain in a separate function, it should still work.

#26 - 11/15/2011 01:42 PM - ngoto (Naohisa Goto)

- Status changed from Assigned to Closed

This issue was solved with changeset r33757.

Lucas, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- include/ruby/defines.h (FLUSH_REGISTER_WINDOWS): move sparc asm code to a separete file sparc.c for preventing inlining optimization. Patched by Jurij Smakov. [Bug #5244] [ruby-core:40685]
- sparc.c (rb_sparc_flush_register_windows): ditto.
- configure.in: ditto.

#27 - 11/15/2011 04:13 PM - ngoto (Naohisa Goto)

The patch is applied to trunk as r33757, r33758 (and r33760) with little modification.
Backport request to ruby 1.9.3 is submitted as #5636.
http://redmine.ruby-lang.org/issues/show/5636

#28 - 09/21/2016 08:56 PM - genunix (Dennis Clarke)

This may still be a problem on Solaris sparc64 where the link stage fails thus :

```
compiling vm.c
"./include/ruby/st.h", line 85: warning: nonportable bit-field type
"vm_insnhelper.c", line 1838: warning: statement not reached
"vm.inc", line 1443: warning: statement not reached
"vm.inc", line 1671: warning: statement not reached
"vm_exec.c", line 125: warning: statement not reached
"vm_method.c", line 1220: warning: statement not reached
"vm_method.c", line 1259: warning: statement not reached
"vm_eval.c", line 249: warning: statement not reached
"vm_eval.c", line 1066: warning: statement not reached
"vm.c", line 442: warning: statement not reached
"vm.c", line 1786: warning: "__typeof__" is an extension of ANSI C
gmake[1]: Leaving directory `/usr/local/build/ruby-2.2.5_SunOS5.10_sparcv9.001'
processing probes in object files
linking miniruby
Undefined                       first referenced
 symbol                             in file
asm                                 sparc.o
ld: fatal: symbol referencing errors. No output written to miniruby
gmake: *** [miniruby] Error 2
$
```

This is Solaris 10 sparc on a T5240 with Oracle Studio compilers version 12.5 and using strict ANSI compliance mode and c99 compiler flags :

$ echo $CFLAGS
-errfmt=error -erroff=%none -errshort=full -xstrconst -xildoff -m64 -xmemalign=8s -xnolibmil -Xc -xcode=pic32 -xregs=no%appl -xlibmieee -mc -g -xs -ftrap=%none -Qy -xbuiltin=%none -xdebugformat=dwarf -xunroll=1 -xtarget=ultraT2 -xcache=8/16/4:4096/64/16

Perhaps I need to use just cc and a transition mode option that is not so strict about C code compliance ?

#29 - 09/26/2016 10:56 AM - shyouhei (Shyouhei Urabe)

- Status changed from Closed to Open

(Just reopening... Sorry I don't have this specific hardware so can't reproduce)

#30 - 09/26/2016 12:35 PM - ngoto (Naohisa Goto)

- Status changed from Open to Closed

> This may still be a problem on Solaris sparc64 where the link stage fails thus :

No, different issue from the original report.

The error message shows that the symbol "asm" is not found.
With "-Xc" option (strict ISO C mode), "asm" as an reserved word (in-line assembler) is disabled and is treated as a normal function.
Ruby uses "asm" inline assembler and thus "-Xc" can not be used.

Files

ruby.patch	480 Bytes	10/19/2011	jurij (Jurij Smakov)

flush_windows.patch 2.19 KB 11/03/2011 jurij (Jurij Smakov)