|  |  |  |
| --- | --- | --- |
| Bochs User Manual | | |
| [Prev](http://docs.google.com/using-slirp.html) | Chapter 8. Tips and Techniques | [Next](http://docs.google.com/debugging-with-gdb.html) |

8.13. Using Bochs internal debugger

You can now conditionally compile in a GDB like command line debugger, that allows you to set breakpoints, step through instructions, and other useful functions. If there isn't a command for something you believe is generally useful for the debugger, let me know and I'll implement it if possible.

**Note:** This section describes how to enable and use the Bochs command line debugger. For it's builtin graphical front-end please see the [debugger gui](http://docs.google.com/internal-debugger.html#DEBUGGER-GUI) section how to enable it.

To use the debugger, you must configure Bochs with the --enable-debugger and --enable-disasm flags. For example:

./configure --enable-debugger --enable-disasm

**Note:** You must use flex version 2.5.4 or greater. I have heard that version 2.5.2 will not work.

When you first start up Bochs, you will see the command line prompt

bochs:1>

From here, you may use the following commands:

8.13.1. Execution Control

c continue executing  
 cont  
 continue  
  
 s [count] execute count instructions, default is 1  
 step [count]  
  
 s [cpu] [count] for SMP simulation, execute count instructions on cpu, default is 1  
 step [cpu] [count]  
  
 s all [count] for SMP simulation, execute count instructions on all cpus  
 step all [count]  
  
 Ctrl-C stop execution, and return to command line prompt  
 Ctrl-D if at empty line on command line, exit  
  
 q quit debugger and execution  
 quit  
 exit

8.13.2. BreakPoints

NOTE: The format of 'seg', 'off', and 'addr' in these descriptions,  
 are as follows. I don't have any way to set the current radix.  
  
 hexidecimal: 0xcdef0123  
 decimal: 123456789  
 octal: 01234567  
  
 vbreak seg:off Set a virtual address instruction breakpoint  
 vb seg:off  
  
 lbreak addr Set a linear address instruction breakpoint  
 lb addr  
  
 pbreak [\*] addr Set a physical address instruction breakpoint  
 pb [\*] addr (the '\*' is optional for GDB compatibility)  
 break [\*] addr  
 b [\*] addr  
  
 info break Display state of all current breakpoints  
 bpe n Enable a breakpoint  
 bpd n Disable a breakpoint  
 delete n Delete a breakpoint  
 del n  
 d n

8.13.3. Memory WatchPoints

watch read addr Insert a read watch point at physical address addr  
 watch r addr Insert a read watch point at physical address addr  
  
 watch write addr Insert a write watch point at physical address addr  
 watch w addr Insert a write watch point at physical address addr  
  
 watch Display state of current memory watchpoints  
  
 watch stop Stop simulation when a watchpoint is encountered (default)  
 watch continue Do not stop simulation when a watchpoint is encountered  
  
 unwatch addr Remove watchpoint to specific physical address  
 unwatch Remove all watch points  
  
 trace-mem on/off Enable/Disable memory access tracing

8.13.4. Manipulating Memory

x /nuf addr Examine memory at linear address addr  
 xp /nuf addr Examine memory at physical address addr  
 n Count of how many units to display  
 u Unit size; one of  
 b Individual bytes  
 h Halfwords (2 bytes)  
 w Words (4 bytes)  
 g Giant words (8 bytes)  
 NOTE: these are \*not\* typical Intel nomenclature sizes,  
 but they are consistent with GDB convention.  
 f Printing format. one of  
 x Print in hexadecimal  
 d Print in decimal  
 u Print in unsigned decimal  
 o Print in octal  
 t Print in binary  
  
 n, f, and u are optional parameters. u and f default to the last values  
 you used, or to w(words) and x(hex) if none have been supplied.  
 n currently defaults to 1. If none of these optional parameters are  
 used, no slash should be typed. addr is also optional. If you don't  
 specify it, it will be the value the next address (as if you had  
 specified n+1 in the last x command).  
  
 setpmem addr datasize val Set physical memory location of size  
 datasize to value val.  
  
 writemem dump a number of bytes of virtual memory starting from  
 the specified linear address into a file  
  
 crc addr1 addr2 Show CRC32 for physical memory range addr1..addr2

8.13.5. Info commands

r|reg|regs|registers List of CPU integer registers and their contents  
 fp|fpu List of all FPU registers and their contents  
 mmx List of all MMX registers and their contents  
 sse|xmm List of all SSE registers and their contents  
 ymm List of all AVX registers and their contents  
 sreg Show segment registers and their contents  
 dreg Show debug registers and their contents  
 creg Show control registers and their contents  
  
 info cpu List of all CPU registers and their contents  
 info eflags Show decoded EFLAGS register  
 info break Information about current breakpoint status  
 info tab Show paging address translation  
 info device Show state of the specified device

8.13.6. Manipulating CPU Registers

set reg = expr Change a CPU register to value of expression.  
 Currently only general purpose registers and instruction pointer  
 are supported. You may not change eflags, segment registers,  
 floating point or SIMD registers.  
  
 Examples: set eax = 2+2/2  
 set esi = 2\*eax+ebx  
  
 registers List of CPU registers and their contents  
 regs  
 reg  
 r

8.13.7. Disassembly commands

disassemble start end Disassemble instructions in given linear address  
 range, inclusive of start, exclusive of end.  
 Use "set $disassemble\_size =" to tell  
 debugger desired segment size. Use a value for  
 end of less than start (or zero) if you only  
 want the first instruction disassembled.  
  
 disassemble switch-mode Switch between Intel and AT&T disassebly styles  
 for debugger disassembler.  
  
 disassemble size = n Tell debugger what segment size to use when  
 the "disassemble" command is used. Use values  
 of 0, 16 or 32 for n. Value of 0 means  
 "use segment size specified by current CS  
 segment". Default is 0.  
  
 set $auto\_disassemble = n Cause debugger to disassemble current instruction  
 every time execution stops if n=1. Default is 0.  
 Segment size of current CPU context is used for  
 disassembly, so the "disassemble size" variable is  
 ignored.  
  
 set disassemble on The same as 'set $auto\_disassemble = 1'  
 set disassemble off The same as 'set $auto\_disassemble = 0'

8.13.8. Instruction tracing

trace on Disassemble every executed instruction. Note  
 that instructions which caused exceptions are  
 not really executed, and therefore not traced.  
  
 trace off Disable instruction tracing.

8.13.9. Instrumentation

To use instrumentation features in bochs, you must compile in support for it. You should build a custom instrumentation library in a separate directory in the "instrument/" directory. To tell configure which instrumentation library you want to use, use the --enable-instrumentation option. The default library consists of a set of stubs, and the following are equivalent:

./configure [...] --enable-instrumentation  
 ./configure [...] --enable-instrumentation="instrument/stubs"

You could make a separate directory with your custom library, for example "instrument/myinstrument", copy the contents of the "instrument/stubs" directory to it, then customize it. Use:

./configure [...] --enable-instrumentation="instrument/myinstrument"

8.13.10. Instrumentation commands

instrument [command] calls BX\_INSTR\_DEBUG\_CMD instrumentation callback with [command]

8.13.11. Other Commands

ptime

Print the current time (number of ticks since start of simulation).

sb delta

Insert a time break point "delta" instructions into the future ("delta" is a 64-bit integer followed by "L", for example 1000L).

sba time

Insert a time break point at "time" ("time" is a 64-bit integer followed by "L", for example 1000L).

print-stack [num words]

Print the num words top 16-bit words on the stack. Num words defaults to 16. Only works reliably in protected mode when the base address of the stack segment is zero.

modebp

Toggles CPU mode switch breakpoint.

ldsym [global] filename [offset]

Load symbols from file filename. If the global keyword is added, then the the symbols will be visible in all contexts for which symbols have not been loaded. Offset (default is 0) is added to every symbol entry. The symbols are loaded in the current (executing) context.

The symbol file consists of zero or more lines of the format

"%x %s"

.

show [string]

Toggles show symbolic info (calls to begin with).  
 show - shows current show mode  
 show mode - show, when processor switch mode  
 show int - show, when interrupt is happens  
 show call - show, when call is happens  
 show ret - show, when iret is happens  
 show off - toggles off symbolic info  
 show dbg-all - turn on all show flags  
 show dbg-none - turn off all show flags

8.13.12. The Bochs debugger gui

The graphical front-end for the Bochs command line debugger is available for Windows and GTK2 hosts.

To use the gui debugger, you must configure Bochs with the default debugger switches and the --enable-debugger-gui flag. For example:

./configure --enable-debugger --enable-disasm --enable-debugger-gui

At runtime you need to add the value gui\_debug to the [display\_library](http://docs.google.com/bochsrc.html#BOCHSOPT-DISPLAYLIBRARY) options parameter in order to use the gui instead of the command line debugger. This example shows how to use it with the 'x' gui:

display\_library: x, options="gui\_debug"

The gui debugger consists of a gui window with a menu bar, a button bar and some child windows that show the cpu registers, disassembler output, memory dump and the internal debugger output. A command prompt for entering debugger commands is also available.

 List the features here.

Most of the gui debugger settings are now saved to an INI file on exit and restored at the next run.

8.13.13. Related links

 For information on advanced debugger usage see the [developer documentation](http://docs.google.com/development/debugger-advanced.html) (under construction).

|  |  |  |
| --- | --- | --- |
| [Prev](http://docs.google.com/using-slirp.html) | [Home](http://docs.google.com/index.html) | [Next](http://docs.google.com/debugging-with-gdb.html) |
| Using the 'slirp' networking module | [Up](http://docs.google.com/howto.html) | Using Bochs and the remote GDB stub |