

Re: Debug.exe for win32

Source: <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2004-09/0291.html>

From: Eric P. (*Eric.P_at_discussions.inter.nl.net*)

Date: 09/07/04

Date: Tue, 07 Sep 2004 09:24:01 +0200

I don't know this is exactly what you need!
Years ago I bought the book "Pentium Processor Optimization Tools" by
Michael L. Schmit ISBN 0-12-627230-1.
The book came with a floppy containing amongst others DEBUG32.

DEBUG32 help file

Address format:

segment:offset is a real mode segment plus offset.
selector|offset is a protected mode selector plus offset.
linear+offset is a protected mode linear address plus offset.
Segment addresses (":" identifier) refer to a segment address in both
real and protected mode. Selector addresses ("|" identifier) must be
used in protected mode to refer to a selector+offset value.

Assemble <address>

Assembles instructions to memory.

BP <address> <R|W|RW|I>

Sets a breakpoint. If no R/W/RW/I is specified, then the breakpoint is
for execution only. "R" causes a break on reading data. "W" breaks on
writing data. "RW" breaks on reading or writing. "I" uses the debug
registers to break on execution (works with ROM).

BC <address|*>

Clears a breakpoint (<*> for all breakpoints).

BL

Lists breakpoints. Format is: n) e address t=n
"t=n" is total executions.

CLS

Clears the screen

CPU

Displays cpu type, operating mode, A20 status, and prefetch length.

Compare <address_1 <address_2> <L length>>>

Compares data at <address_1> to data at <address_2> for length <L length>.

DA <address>

Dump data at <address> in Ascii format.

DB <address>

Dump data at <address> in byte format.

DD <address>

Dump data at <address> in doubleword (32 bit) format.

DF <address>

Dump data at <address> in far pointer (16:32) format.

DP <address>

Dump data at <address> in pointer (16:16) format.

DW <address>

Dump data at <address> in word (16 bit) format.

Dump <address>

Dump data at <address> in the current format.

DR

Display the debug registers. They may be directly modified with the R DRx command. The BP command provides a more convenient method of setting up the debug registers.

Enter <address <byte<,byte>>>

'character_string'

Enters data into memory.

Fill <address <end_address <byte<,byte<...>>>>

<L length>

Fills memory with a repeating value.

FLip ON|OFF

ON (default) causes switching to the application screen on T and P commands.

OFF does not switch on T and P commands, and thus may confuse the application and debugger screens.

Go <=<address> <temp_breakpoint <temp_breakpoint2 <...>>>

Resumes execution. <=<address> sets new IP value.

<temp_breakpoint> sets breakpoints temporary breakpoint(s).

HELP <initial_command_letters>

HELP or ? commands alone produce the full help text.

The help output can be limited by specifying the first letter(s) of the

command(s) to be described.

Hex value1 value2

Calculates the hexadecimal sum and difference of value1 & value2.

In <address>

Inputs a byte from port <address> and displays it in hex.

Load <program_name <address>>

Loads the specified program.

An address may be specified for non-EXE format files.

Load <address>

Loads the file identified by a NAME command.

An address may be specified for non-EXE format files.

Load address drive_number first_sector number_of_sectors

Loads data on a sector basis from the specified disk.

drive_number is 0 for A, 1 for B, 2 for C, etc.

first_sector is the starting sector on the disk (in hex).

number_of_sectors is the number of 512 byte sectors (in hex).

LOG <filename>

Copies all command window output to the specified file.

LOG without a filename closes the logging file.

Use of the logging facility is limited to non-DOS, non-interrupt areas of code. Use inside DOS or interrupt handlers does not work.

MEMory

Shows memory control block allocations.

MORE ON|OFF

ON (default) pauses command output on each screenfull.

OFF allows scrolling to proceed uninhibited. This is useful when the data is simply being logged to a file, rather than being read.

When the MORE prompt is issued, any key continues for another full window.

ESC cancels the command, and a digit allows that many more lines output.

Move <address_1> <address_2> <L length>

Copies data at <address_1> to <address_2> for length <L length>.

Name <program name | parms>

Sets the program name for a subsequent Load command.

Sets the command line parameters for an already loaded program.

Out <address> <value>

Outputs <value> to port <address>.

PRegs

Displays the privileged registers. These include the GDT, LDT, IDT,

and TSS pointers, as well as EFLAGS and the control registers.

Ptrace <=address> <count>

Traces a single inline instruction or procedure.

<=address> sets resume location.

<count> specifies the number of instructions to trace.

Quit

Quits the debugger and returns to DOS.

R <register_name <new_value>>

R alone displays all of the registers in the current (16/32) mode.

R <register_name> displays the current value of <register_name> and prompts for input of a new value.

R <register_name> <new_value> sets a register to a new value.

R16/R32 ...

R16 sets 16 bit register mode.

R32 sets 32 bit register mode.

Both commands perform the same as the "R" command, with the additional function of setting the 16/32 bit mode.

RC

Displays the changed registers.

Read <file_name <address>>

Reads the specified file. An address may be specified, or defaulted to the prefix area + 100h. EXE formatting is read as ordinary data (it is not interpreted).

REASON

Displays the reason for entry into the debugger.

Search <address <end_address <byte<,byte<,...>>>>
<L length>

Searches for the specified byte or character values.

SElector value

Displays the linear address active for the specified selector.

SKIPCR

Disables access to the control and debug registers for operating environments that do not allow such access.

Trace <=address> <count>

Traces a single instruction.

<=address> overrides the resume address.

<count> sets the number of instructions to trace.

U16 <address>

Interprets memory at <address> as USE16 format assembly language.

U32 <address>

Interprets memory at <address> as USE32 format assembly language.

Unassemble <address>

Interprets memory at <address> as assembly language.

Viewswap

Swaps the display to the application screen. Any key returns.

See also the FLIP command.

Write address drive_number first_sector number_of_sectors

Writes data on a sector basis from the specified disk.

drive_number is 0 for A, 1 for B, 2 for C, etc.

first_sector is the starting sector on the disk (in hex).

number_of_sectors is the number of 512 byte sectors (in hex).

XA page_count

Allocates page_count EMS pages.

XD handle

Deallocates specified EMS handle.

XM physical_page logical_page handle

Maps the specified EMS handle/logical page into a physical page.

XS

Displays the current EMS status.

* comment

Enters a comment.

Bare Metal Debugger 2.0

DEBUG32 (The command-line debugger shipped with this book) is the predecessor of BMD (Bare Metal Debugger). Some features of BMD:

- full screen symbolic debugging with source code
- can run as a TSR
- can isolate itself from the hardware to allow debugging TSR's and device drivers. This is done by using built-in keyboard and screen I/O routines.
- remote machine debugging
- can be placed in extended memory

To receive upgrades for DEBUG32, send your name and address to the address below. (There is a \$10 shipping & handling fee.)

Receive a \$50 discount on the purchase of the Bare Metal Debugger with purchase of this book. Call or write:

Quantasm Corp. (800) 765-8086 sales
19672 Stevens Creek Blvd., #307-B (408) 244-6826 tech
support
Cupertino, CA 95014 (408) 244-7268 FAX
76347,3661

CompuServe

Robert wrote:

> *Greetings everyone,*
>
> *Is it possible to work with debug.exe (which comes with every MS os*
> *that I know of) to create win32 programs? I know that debug.exe is*
> *limited to creating 64k com files, and that windows programs probably*
> *need this much for their headers and such. Still, I was wondering if*
> *there was a way to use debug.exe to cut apart the important headers*
> *off of existing win32 files, and use them as the headers for your*
> *programs that you build with debug.exe*
>
> *I'd like to be able to sit down at any MS-OS and start writing*
> *programs. There aren't any default C/C++ compilers in these OS's like*
> *there are in Unix unfortunately. I don't know if there is some*
> *default preinstalled compiler/assembler that will natively compile*
> *win32 code on a Win32 platform. Thank you for any help you can*
> *provide. If this is the wrong newsgroup, please point me to the*
> *correct one. Thank you.*