OVP Trace User Guide

Imperas Software Limited
Imperas Buildings, North Weston,
Thame, Oxfordshire, OX9 2HA, UK
docs@imperas.com

Author:	Imperas Software Limited
Version:	1.0.0
Filename:	OVP_Trace_User_Guide.doc
Project:	Trace Reference
Last Saved:	November 4, 2021
Keywords:	

Copyright Notice

Right to Copy Documentation

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

Table of Contents

1 Introduction

The OVP simulator implements a powerful generic instruction tracing capability that can display instructions as they are executed together with information such as registers changed or written, or memory accesses performed by a processor as each instruction executes. Tracing can be controlled either by simulator command line arguments, or interactively by a debugger, or by functions in the OP API, or by model commands.

This document describes how trace output can be controlled using these four methods.

2 Command Line Trace Options

Tracing can be controlled through the command line interface. The primary available trace options are summarized in the following table:

`--trace`	enable tracing
`--traceafter <count>`	enable tracing after the given instruction count
`--tracecount <count>`	trace this number of instructions
`--tracechange`	show changed registers after each trace line
`--tracewrite`	trace all written registers, even if they don't change
`--tracemode`	include processor mode in trace line
`--traceshowicount`	include instruction/cycle count in trace line
`--tracelowpc <pc>`	only trace instructions at address >= pc
`--tracehighpc <pc>`	only trace instructions at address <= pc
`--tracemem <code>`	trace load/store, fetch and system memory accesses based on code
`--tracefile <name>`	write all trace output to this file
`--traceshowcpuname`	include CPU name on all trace lines, not just instruction line

There are also some legacy trace options not discussed here.

Trace options can either be applied globally on a command line or to specific processors. For example, this line turns on tracing for all processors in a simulation:

```
iss.exe \
    --trace \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf
```

To see all trace control flags available (and other command line flags) include `-helpall` in the run line.

This line turns on tracing for a specific processor in a multicore system:

```
iss.exe \
    --override        iss/cpu0/trace=T \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf
```

To see the trace overrides available (and all other overrides) include `-showoverrides` in the run line.

Following sections describe these trace flags in detail, with examples. These examples all use the OVP RISC-V model, but tracing is a general capability and can be used with any OVP processor model.

2.1 Option `--trace`

The option `--trace` turns on instruction tracing at the start of simulation.

```
iss.exe \
    --trace \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf

Info 'iss/cpu0', 0x0000000080000000(_start): 5000506f j       80005500
Info 'iss/cpu0', 0x0000000080005500(START_TEST): fffff297 auipc   t0,0xfffff
Info 'iss/cpu0', 0x0000000080005504(START_TEST+4): b0028293 addi    t0,t0,-1280
Info 'iss/cpu0', 0x0000000080005508(START_TEST+8): 30529073 csrw    mtvec,t0
Info 'iss/cpu0', 0x000000008000550c(START_TEST+c): 00000297 auipc   t0,0x0
. . . etc . . .
```

The format of the line is:

1. The `Info` tag;
2. The processor name;
3. The program counter, with optional label read from the executable;
4. The instruction disassembly as provided by the processor model.

Depending on the processor model implementation, the instruction disassembly may vary. For the RISC-V processor, the disassembly is the instruction pattern (a number of hex characters corresponding to the instruction size) followed by mnemonic and operands.

2.2 Option `--traceshowicount`

The option `--traceshowicount` enhances the trace output to add information about the current instruction count (or nominal cycle count, in simulations with such information enabled).

```
iss.exe \
    --trace \
    --traceshowicount \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf

Info 1: 'iss/cpu0', 0x0000000080000000(_start): 5000506f j       80005500
Info 2: 'iss/cpu0', 0x0000000080005500(START_TEST): fffff297 auipc   t0,0xfffff
Info 3: 'iss/cpu0', 0x0000000080005504(START_TEST+4): b0028293 addi    t0,t0,-1280
Info 4: 'iss/cpu0', 0x0000000080005508(START_TEST+8): 30529073 csrw    mtvec,t0
Info 5: 'iss/cpu0', 0x000000008000550c(START_TEST+c): 00000297 auipc   t0,0x0
. . . etc . . .
```

Note that `--traceshowicount` does not itself enable tracing: instead, it modifies what is reported when tracing is enabled by `--trace` or `--traceafter`.

2.3 Option `--traceafter`

The option `--traceafter` causes tracing to be enabled only after the given number of instructions or nominal cycles have elapsed:

```
iss.exe \
    --traceafter      10 \
    --traceshowicount \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf

Info 10: 'iss/cpu0', 0x0000000080005520(START_TEST+20): b61fa0ef jal     ra,80000080
Info 11: 'iss/cpu0', 0x0000000080000080(writeS): 82aa     mv      t0,a0
Info 12: 'iss/cpu0', 0x0000000080000082(writeS+2): 0002c503 lbu     a0,0(t0)
Info 13: 'iss/cpu0', 0x0000000080000086(writeS+6): 0285     addi    t0,t0,1
Info 14: 'iss/cpu0', 0x0000000080000088(writeS+8): 00050563 beqz    a0,80000092
Info 15: 'iss/cpu0', 0x000000008000008c(writeS+c): 0005200b custom0
. . . etc . . .
```

2.4 Option `--tracecount`

The option `--tracecount` switches tracing off after the given number of instructions have been traced. It is commonly used in combination with `--traceafter` to obtain an instruction trace within a long simulation run:

```
iss.exe \
    --traceafter      10 \
    --tracecount      5 \
    --traceshowicount \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf

Info 10: 'iss/cpu0', 0x0000000080005520(START_TEST+20): b61fa0ef jal     ra,80000080
Info 11: 'iss/cpu0', 0x0000000080000080(writeS): 82aa     mv      t0,a0
Info 12: 'iss/cpu0', 0x0000000080000082(writeS+2): 0002c503 lbu     a0,0(t0)
Info 13: 'iss/cpu0', 0x0000000080000086(writeS+6): 0285     addi    t0,t0,1
Info 14: 'iss/cpu0', 0x0000000080000088(writeS+8): 00050563 beqz    a0,80000092
```

2.5 Option `--tracemode`

The option `--tracemode` enhances the trace output to add information about the current processor mode:

```
iss.exe \
    --traceafter      10 \
    --tracecount      5 \
    --traceshowicount \
    --tracemode \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf

Info 10: 'iss/cpu0', 0x0000000080005520(START_TEST+20): Machine b61fa0ef jal
ra,80000080
Info 11: 'iss/cpu0', 0x0000000080000080(writeS): Machine 82aa     mv      t0,a0
```

```
Info 12: 'iss/cpu0', 0x0000000080000082(writeS+2): Machine 0002c503 lbu     a0,0(t0)
Info 13: 'iss/cpu0', 0x0000000080000086(writeS+6): Machine 0285     addi    t0,t0,1
Info 14: 'iss/cpu0', 0x0000000080000088(writeS+8): Machine 00050563 beqz    a0,80000092
```

Note that `--tracemode` does not itself enable tracing: instead, it modifies what is reported when tracing is enabled by `--trace` or `--traceafter`. The model-specific mode is shown just before the instruction disassembly.

2.6 Option `--tracechange`

The option `--tracechange` enhances the trace output to add information about registers that are changed as the result of an instruction being executed. The changed registers are shown after the instruction line, with old and new values in hexadecimal. There can in general be multiple register change lines for each instruction:

```
iss.exe \
    --traceafter      10 \
    --tracecount      5 \
    --traceshowicount \
    --tracemode \
    --tracechange \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf
```

```
Info 10: 'iss/cpu0', 0x0000000080005520(START_TEST+20): Machine b61fa0ef jal
ra,80000080
Info   ra 0000000000000000 -> 0000000080005524
Info 11: 'iss/cpu0', 0x0000000080000080(writeS): Machine 82aa     mv      t0,a0
Info   t0 0000000080004d40 -> 0000000080001014
Info 12: 'iss/cpu0', 0x0000000080000082(writeS+2): Machine 0002c503 lbu     a0,0(t0)
Info   a0 0000000080001014 -> 000000000000004c
Info 13: 'iss/cpu0', 0x0000000080000086(writeS+6): Machine 0285     addi    t0,t0,1
Info   t0 0000000080001014 -> 0000000080001015
Info 14: 'iss/cpu0', 0x0000000080000088(writeS+8): Machine 00050563 beqz    a0,80000092
```

Note that `--tracechange` does not itself enable tracing: instead, it modifies what is reported when tracing is enabled by `--trace` or `--traceafter`.

Some processor registers are whole or partial aliases of others; for example, in the RISC-V processor the `sstatus` system register is a partial alias of the `mstatus` system register. If such a register is modified by an instruction, then all register aliases that have changed will be reported.

2.7 Option `--tracewrite`

The option `--tracewrite` enhances the trace output to add information about registers that are identified as written as the result of an instruction being executed. The written registers are shown after the instruction line, with old and new values in hexadecimal – unlike for `--tracechange`, old and new values may be identical. There can in general be multiple register write lines for each instruction:

```
iss.exe \
    --traceafter      10 \
```

```
    --tracecount      5 \
    --traceshowicount \
    --tracemode \
    --tracewrite \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf
```

```
Info 10: 'iss/cpu0', 0x0000000080005520(START_TEST+20): Machine b61fa0ef jal
ra,80000080
Info   ra 0000000000000000 -> 0000000080005524
Info 11: 'iss/cpu0', 0x0000000080000080(writeS): Machine 82aa     mv      t0,a0
Info   t0 0000000080004d40 -> 0000000080001014
Info 12: 'iss/cpu0', 0x0000000080000082(writeS+2): Machine 0002c503 lbu     a0,0(t0)
Info   a0 0000000080001014 -> 000000000000004c
Info 13: 'iss/cpu0', 0x0000000080000086(writeS+6): Machine 0285     addi    t0,t0,1
Info   t0 0000000080001014 -> 0000000080001015
Info 14: 'iss/cpu0', 0x0000000080000088(writeS+8): Machine 00050563 beqz    a0,80000092
```

Note that `--tracewrite` does not itself enable tracing: instead, it modifies what is reported when tracing is enabled by `--trace` or `--traceafter`.

Option `--tracewrite` only reports registers that are written as the result of an instruction executing *normally*, without taking an exception. If an instruction takes an exception, nothing is reported by `--tracewrite`: this is because in general such an instruction my update only some or none of the destination registers implied by the opcode and therefore the information about written registers is unreliable.

Some processor registers are whole or partial aliases of others; for example, in the RISC-V processor the `sstatus` system register is a partial alias of the `mstatus` system register. If such a register is written by an instruction, then which register is reported is model-specific.

2.8 Options `-tracelowpc` and `-tracehighpc`

These options constrain tracing so that it is shown only when executing at addresses in the range `lowpc:highpc`:

```
iss.exe \
    --trace \
    --traceshowicount \
    --tracelowpc      0x80005504 \
    --tracehighpc     0x80005514 \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf
```

```
Info 3: 'iss/cpu0', 0x0000000080005504(START_TEST+4): b0028293 addi    t0,t0,-1280
Info 4: 'iss/cpu0', 0x0000000080005508(START_TEST+8): 30529073 csrw    mtvec,t0
Info 5: 'iss/cpu0', 0x000000008000550c(START_TEST+c): 00000297 auipc   t0,0x0
Info 6: 'iss/cpu0', 0x0000000080005510(START_TEST+10): 83428293 addi    t0,t0,-1996
Info 7: 'iss/cpu0', 0x0000000080005514(START_TEST+14): 34029073 csrw    mscratch,t0
```

Note that these options do not themselves enable tracing: instead, they modify what is reported when tracing is enabled by `--trace` or `--traceafter`.

2.9 Option `--tracemem`

The option `--tracemem` enhances the trace output to add information about memory accesses caused by an instruction. Different types of access will be reported, based on the argument to `--tracemem`, as described below. The argument can be either a number (1-7) or a string.

If a number 1-7:

1. If bit 0 is set, load/store accesses are reported with prefix `MEMR` or `MEMW`;
2. If bit 1 is set, fetch accesses are reported with prefix `MEMX`;
3. If bit 2 is set, other system memory accesses are reported with prefix `MEMS`. This will typically be accessed as a result of activities such as hardware page table walks, but could also be hardware stack updates on exception entry and exit.

If a string:

1. If string contains `L`, load/store accesses are reported with prefix `MEMR` or `MEMW`;
2. If string contains `X`, fetch accesses are reported with prefix `MEMX`;
3. If string contains `S`, other system memory accesses are reported with prefix `MEMS`, as described above.

```
iss.exe \
    --traceafter      10 \
    --tracecount      5 \
    --traceshowicount \
    --tracemode \
    --tracechange \
    --tracemem        L \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf

Info 10: 'iss/cpu0', 0x0000000080005520(START_TEST+20): Machine b61fa0ef jal
ra,80000080
Info   ra 0000000000000000 -> 0000000080005524
Info 11: 'iss/cpu0', 0x0000000080000080(writeS): Machine 82aa     mv      t0,a0
Info   t0 0000000080004d40 -> 0000000080001014
Info 12: 'iss/cpu0', 0x0000000080000082(writeS+2): Machine 0002c503 lbu     a0,0(t0)
Info   MEMR 0x80001014 0x80001014 1 4c
Info   a0 0000000080001014 -> 000000000000004c
Info 13: 'iss/cpu0', 0x0000000080000086(writeS+6): Machine 0285     addi    t0,t0,1
Info   t0 0000000080001014 -> 0000000080001015
Info 14: 'iss/cpu0', 0x0000000080000088(writeS+8): Machine 00050563 beqz    a0,80000092
```

In the above example, the `MEMR` line indicates a read from memory; following fields give this information:

1. The read is from virtual address `0x80001014`;
2. The read is from physical address `0x80001014`;
3. The read is of one byte;
4. The value read is `0x4c`.

```
iss.exe \
    --traceafter      10 \
    --tracecount      5 \
```

```
    --traceshowicount \
    --tracemode \
    --tracechange \
    --tracemem        LX \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf

Info 10: 'iss/cpu0', 0x0000000080005520(START_TEST+20): Machine b61fa0ef jal
ra,80000080
Info   MEMX 0x80005520 0x80005520 2 a0ef
Info   MEMX 0x80005522 0x80005522 2 b61f
Info   ra 0000000000000000 -> 0000000080005524
Info 11: 'iss/cpu0', 0x0000000080000080(writeS): Machine 82aa     mv      t0,a0
Info   MEMX 0x80000080 0x80000080 2 82aa
Info   t0 0000000080004d40 -> 0000000080001014
Info 12: 'iss/cpu0', 0x0000000080000082(writeS+2): Machine 0002c503 lbu     a0,0(t0)
Info   MEMX 0x80000082 0x80000082 2 c503
Info   MEMX 0x80000084 0x80000084 2 0002
Info   MEMR 0x80001014 0x80001014 1 4c
Info   a0 0000000080001014 -> 000000000000004c
Info 13: 'iss/cpu0', 0x0000000080000086(writeS+6): Machine 0285     addi    t0,t0,1
Info   MEMX 0x80000086 0x80000086 2 0285
Info   t0 0000000080001014 -> 0000000080001015
Info 14: 'iss/cpu0', 0x0000000080000088(writeS+8): Machine 00050563 beqz    a0,80000092
Info   MEMX 0x80000088 0x80000088 2 0563
Info   MEMX 0x8000008a 0x8000008a 2 0005
```

With `--tracemem LX`, the trace also includes information about instruction *fetches* with the `MEMX` prefix. All fetches are 2-byte accesses in this processor model; longer instructions require more than one 2-byte fetch access.

2.10 Option `--traceshowcpuname`

The option `--traceshowcpuname` enhances output when `--tracechange`, `--tracewrite` or `--tracemem` are specified so that CPU name and instruction count information are reported on register/memory update lines as well. This simplifies post-processing of log files from multiprocessor simulations when results for a particular processor need to be extracted.

```
iss.exe \
    --traceafter      10 \
    --tracecount      5 \
    --traceshowicount \
    --tracemode \
    --tracewrite \
    --tracemem        LX \
    --traceshowcpuname \
    --processorvendor riscv.ovpworld.org \
    --processorname   riscv \
    --variant         RV64GC \
    -f                test.elf

Info 10: 'iss/cpu0', 0x0000000080005520(START_TEST+20): Machine b61fa0ef jal
ra,80000080
Info 10: 'iss/cpu0',    MEMX 0x80005520 0x80005520 2 a0ef
Info 10: 'iss/cpu0',    MEMX 0x80005522 0x80005522 2 b61f
Info 10: 'iss/cpu0',    ra 0000000000000000 -> 0000000080005524
Info 11: 'iss/cpu0', 0x0000000080000080(writeS): Machine 82aa     mv      t0,a0
Info 11: 'iss/cpu0',    MEMX 0x80000080 0x80000080 2 82aa
```

```
Info 11: 'iss/cpu0',    t0 0000000080004d40 -> 0000000080001014
Info 12: 'iss/cpu0', 0x0000000080000082(writeS+2): Machine 0002c503 lbu     a0,0(t0)
Info 12: 'iss/cpu0',    MEMX 0x80000082 0x80000082 2 c503
Info 12: 'iss/cpu0',    MEMX 0x80000084 0x80000084 2 0002
Info 12: 'iss/cpu0',    MEMR 0x80001014 0x80001014 1 4c
Info 12: 'iss/cpu0',    a0 0000000080001014 -> 000000000000004c
Info 13: 'iss/cpu0', 0x0000000080000086(writeS+6): Machine 0285     addi    t0,t0,1
Info 13: 'iss/cpu0',    MEMX 0x80000086 0x80000086 2 0285
Info 13: 'iss/cpu0',    t0 0000000080001014 -> 0000000080001015
Info 14: 'iss/cpu0', 0x0000000080000088(writeS+8): Machine 00050563 beqz    a0,80000092
Info 14: 'iss/cpu0',    MEMX 0x80000088 0x80000088 2 0563
Info 14: 'iss/cpu0',    MEMX 0x8000008a 0x8000008a 2 0005
```

2.11 Option `--tracefile`

The option `--tracefile` causes all trace output to be redirected to the named log file, instead of to the standard simulator log file. This makes it easy to separate trace output from standard simulator output.

3 Debugger Trace Control

Tracing can be controlled interactively from the Imperas Multiprocessor Debugger or a GDB when connected to the Imperas Simulator. The semantics are the same as when controlled from the command line.

3.1 Tracing from MPD

By default, tracing goes to the simulator standard output and the simulator log file if enabled. It can be redirected to the MPD console if required using the TCL command:

`iredirect -debugger`	Send output to the debugger console
`iredirect -simulator`	Send output back to the simulator console

3.1.1 Debug mode

Tracing is controlled using options to the `set` command. Options are summarized here:

`help set itrace`	Print a summary of MPD trace commands
`set itrace on`	Turn on plain instruction tracing
`set itrace off`	Turn off instruction tracing
`set itrace count`	Turn on instruction tracing, showing the instruction count
`set itrace processorname`	Turn on tracing, showing the processor name on every trace line
`set itrace registers`	Turn on instruction tracing, showing all registers after each trace
`set itrace change`	Turn on instruction tracing, showing changed registers
`set itrace mem-x`	Turn on instruction tracing, showing associated code-fetches
`set itrace mem-l`	Turn on instruction tracing, showing associated loads and stores
`set itrace mem-s`	Turn on instruction tracing, showing any associated system access
`set itrace buffer on`	Turn on the trace buffer (but no tracing)
`set itrace buffer off`	Turn off the trace buffer
`set itrace … after <i>`	Control of instruction tracing can be deferred by a number of instructions `<i>`.

3.1.2 TCL mode

The Imperas Debugger's TCL interpreter also has trace commands summarized here:

`itrace`

`-help`	Show this summary
`-after <i>`	Defer the action for <i> instructions
`-on`	Turn on tracing
`-off`	Turn off tracing
`-instructioncount`	Include the instruction count
`-processorname`	Include the processor name
`-memory <L|S|X]`	Include tracing of memory operations

		L Only show loads and stores X Only show code fetch S Only show system access
	-registers	Show all registers after each instruction
	-registerchange	Show only the changed registers
	-processor	Apply to this processor(s accepts wildcards)
itracebuffer		
	-on	Enable the trace buffer
	-off	Disable the trace buffer
	-dump	Dump the trace buffer contents
	-processor	Apply to this processor(s accepts wildcards)
	-after <i>	Defer the action for <i> instructions

3.2 Tracing from GDB

Tracing is controlled using options to the GDB `monitor` command summarized here:

`monitor set itrace on`	Turn on instruction tracing
`monitor set itrace off`	Turn off instruction tracing
`monitor set itrace buffer on`	Capture to the trace buffer
`monitor set itrace buffer off`	Stop capture to the trace buffer
`monitor set itrace registers on`	Show all registers
`monitor set itrace change on`	Show changing registers
`monitor set itrace count on`	Show instruction count
`monitor set itrace processorname on`	Show processor name in every trace line
`monitor set itrace mem on`	Show all memory access
`monitor set itrace mem-l on`	Show memory loads and stores
`monitor set itrace mem-s on`	Show memory system accesses
`monitor set itrace mem-x on`	Show memory fetches

4 OP Harness Trace Control

Tracing can be turned on and off from within a test harness that uses the OP API. To control the format and contents of the trace, use the command line options described above.

4.1 Tracing

4.1.1 Enabling and disabling

To turn processor tracing on or off after it has executed the given number of instructions:

```
void opProcessorTraceOnAfter (
    optProcessorP processor,
    Uns64         delta
);

void opProcessorTraceOffAfter (
    optProcessorP processor,
    Uns64         delta
);
```

4.1.2 Limiting to an address range

To limit the PC address range where tracing is enabled:

```
void opProcessorTraceHighPCSet (
    optProcessorP processor,
    Addr          upperBounds
);

void opProcessorTraceLowPCSet (
    optProcessorP processor,
    Addr          lowerBounds
);
```

4.2 Using the trace buffer.

Tracing reduces simulator performance by orders of magnitude. Using the trace buffer also reduces performance but by a lesser amount. The trace buffer stores a limited number of instructions (256 in the current implementation) in a circular buffer. The contents of the buffer can then be displayed when the simulator is stopped.

To begin capturing to the trace buffer:

```
opProcessorTraceBufferEnable(optProcessorP processor)
```

To finish capturing to the trace buffer:

```
opProcessorTraceBufferDisable(optProcessorP processor)
```

To dump the contents of the trace buffer:

```
opProcessorTraceBufferDump(optProcessorP processor)
```

5 Trace Control using Model Commands

All processor models have model commands to control tracing. Model commands can be called before simulation from the command line or during simulation from a harness that uses the OP API (This functionality duplicates what is available from the command line, in the debugger and in the OP API).

Processor trace commands

`itrace`	
`-after <i>`	Defer the command by <i> instructions
`-on`	Turn on tracing
`-off`	Turn off tracing
`-instructioncount`	Show instruction count number in each trace line
`-processorname`	Show the processor name in every trace line
`-registers`	Show all registers after each instruction
`-registerchange`	Show only the changed registers after each instruction
`-memory [LXS]`	Show memory accesses associated with each instruction
	`L` Show only loads and stores
	`X` Show code fetches
	`S` Show system access

5.1 How to call model commands

5.1.1 From the command line

To get a list of commands:

```
platform.exe -showcommands
  --callcommand top/cpu/debugflags [show or modify the processor debug flags]
  --callcommand top/cpu/itrace [enable or disable instruction tracing]
```

To get a list of options to a command (note quotes around all command options):

```
platform.exe -callcommand “top/cpu/itrace -help”
  itrace         : enable or disable instruction tracing
    -after       :       apply after this many instructions
    -instructioncount : include the instruction number in each trace
    -off         :               disable instruction tracing
    -on          :               enable instruction tracing
```

To call a command:

```
platform.exe -callcommand “top/cpu/itrace -on -after 10000”
```

5.1.2 From the OP API

Commands can be called from the OP API in a harness:

```
opCommandCallByName(
  optModuleP   root,        (root module)
  const char  *name,        (hierarchical processor name)
  const char  *plugin,      (name of plugin, null in this case)
  const char  *command      (name of the command)
  Uns32        argc,        (number of arguments including the command name)
  const char **argv         (list of arguments including the command name)
);
```

For example:

```
const char *argv[] = {“itrace”, “-on”};
opCommandCallByName(root, “top/cpu0”, 0, “itrace” 2, argv);
```

Other OP functions are available to call a command; refer to the OP API.