Intel® System Studio – "System Debugger – Processor Trace Sample"

Overview

Intel® System Debugger is the complete system debug solution provides deep insight into memory and system configuration.

Key features of Intel® System Debugger:

- JTAG debug for Intel® Atom™, Core™, Xeon® & Quark™ SoC-based platforms
- EFI/UEFI Firmware, bootloader debug, Linux* OS awareness
- Dynamically loaded Linux kernel module debug
- In depth visualization of memory configuration, system state and register sets
- LBR & Intel® Processor Trace On-Chip instruction trace support
- JTAG debug & instruction trace to Microsoft* WinDbg* kernel debugger
- System Trace: System-wide hardware and software event trace

Intel® Processor Trace is the hardware based low overhead code execution logging on instruction level and provides a powerful and deep insight into past instruction flow combined with interactive debug.

Key features of Intel® Processor Trace:

- Low-overhead execution tracing feature
- Capturing information about software execution
- Reconstruct the exact program flow

Terminology

This document discusses 'host' and 'target' systems. The 'host' system is the Linux or Windows desktop computer that you use for most of your work. The 'target' system is some x86-based hardware (or model of such hardware) on which the example Linux or Windows distribution is running.

System Requirements

Recommended hardware:

- A system based on 2nd Generation Intel® Core™ i3, i5, i7 or newer processor

Hardware requirements:

- A system based on Intel® Pentium® processor, Intel® Xeon® processor, or subsequent IA-32 architecture based system;
- A system based on a processor with Intel® 64 architecture;

Software requirements:

- For the basic system requirements on operating system, Intel® System Debugger and refer to the corresponding Release Notes document.

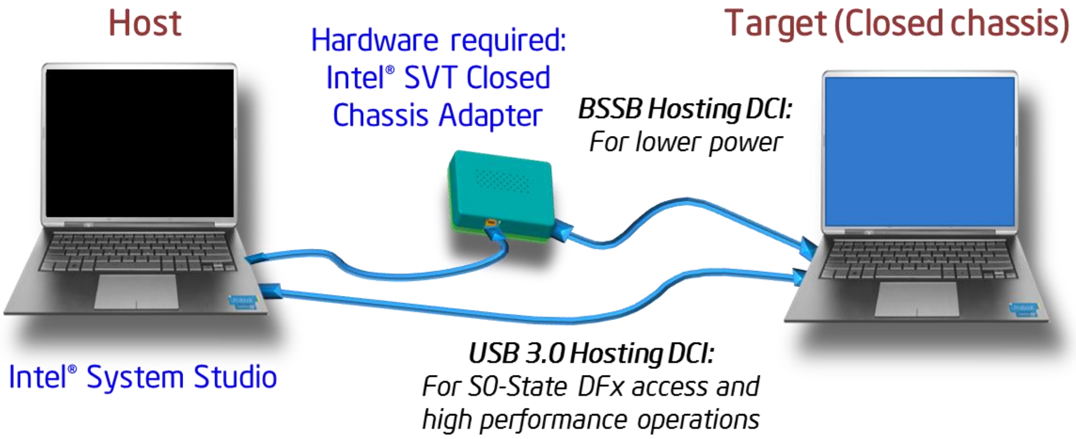
Product specific requirements:

JTAG Probe : ITP-XDP3 / Intel® SVT CCA

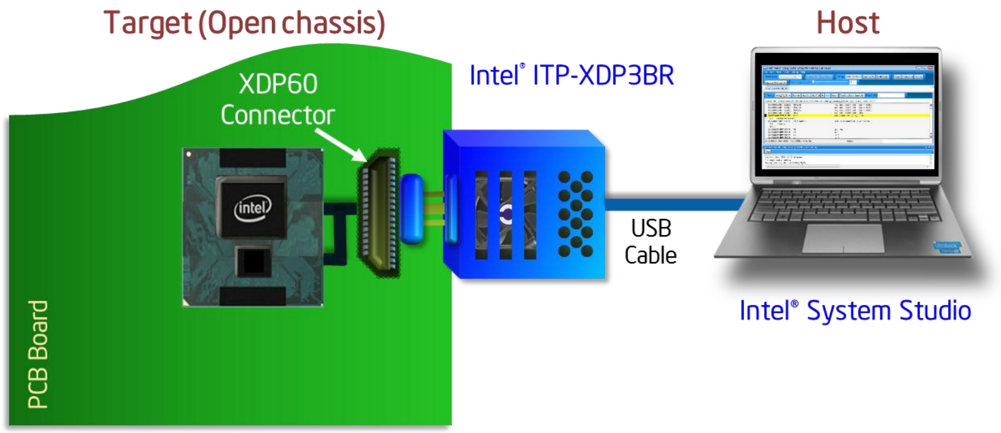
For more information please see Intel® System Studio Requirements.

HW Configuration

- Intel® CCA (Closed Chassis Adapter) is used for low power debugging and tracing and connection is DCI over the USB3.0 (Target side).
- USB3.0 DbC is used for non-low power debugging and tracing.



- Intel® ITP-XDP3BR JTAG probe is used for open chassis with XDP60 port target.

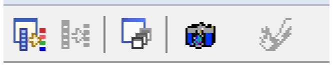


SW Configuration

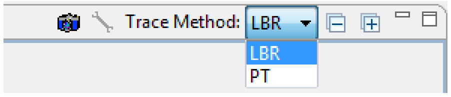
Enable Processor Trace feature in the EFI BIOS:

- CPU Configuration → Processor Trace Memory → No Disabled (4kB~)
- CPU Configuration → Processor Trace → Enabled

Run the XDB and open the Trace Window by clicking Instruction Trace (camera icon):



Select the trace method from the trace window:

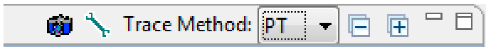


Trace Method: LBR

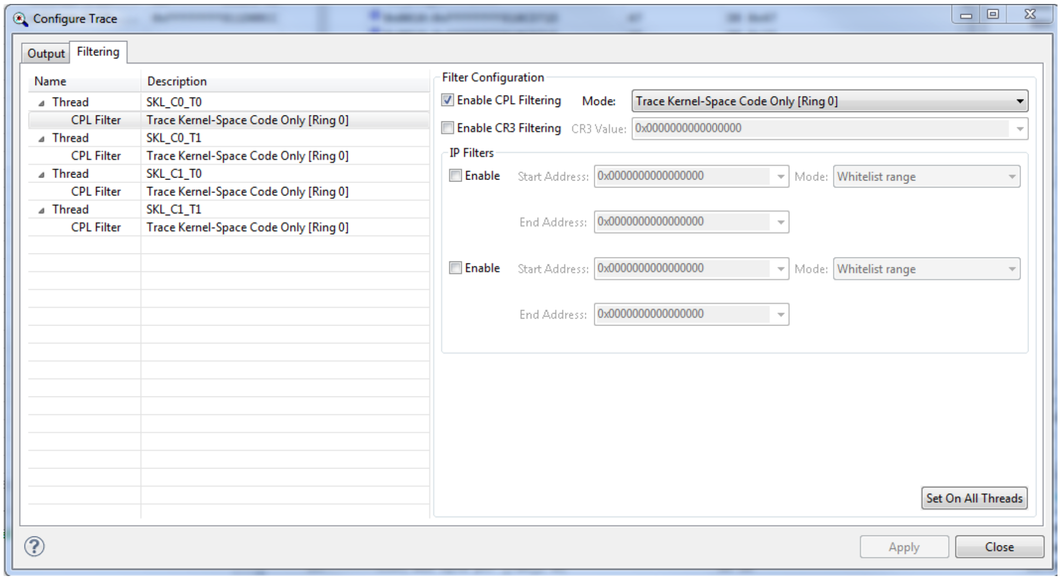
LBR
PT

Trace Configuration:

Click the Configure Trace icon in the trace window toolbar (spanner icon)



Trace Method: PT



Configure Trace

Output | Filtering

Name	Description
Thread	SKL_C0_T0
CPL Filter	Trace Kernel-Space Code Only [Ring 0]
Thread	SKL_C0_T1
CPL Filter	Trace Kernel-Space Code Only [Ring 0]
Thread	SKL_C1_T0
CPL Filter	Trace Kernel-Space Code Only [Ring 0]
Thread	SKL_C1_T1
CPL Filter	Trace Kernel-Space Code Only [Ring 0]

Filter Configuration

☑ Enable CPL Filtering Mode: Trace Kernel-Space Code Only [Ring 0]

□ Enable CR3 Filtering CR3 Value: 0x0000000000000000

IP Filters

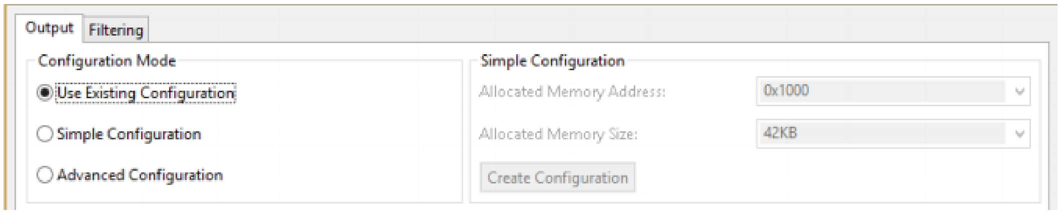
□ Enable Start Address: 0x0000000000000000 Mode: Whitelist range

End Address: 0x0000000000000000

□ Enable Start Address: 0x0000000000000000 Mode: Whitelist range

End Address: 0x0000000000000000

In the output tab and configuration mode menu, you can customize the address and size of trace buffer but you only need to do this step if the trace collection was not configured (by the BIOS). Select Use Existing Configuration is enough when you already configure it in BIOS. In this sample we are using first option – already configured by BIOS.



Output | Filtering

Configuration Mode

● Use Existing Configuration

○ Simple Configuration

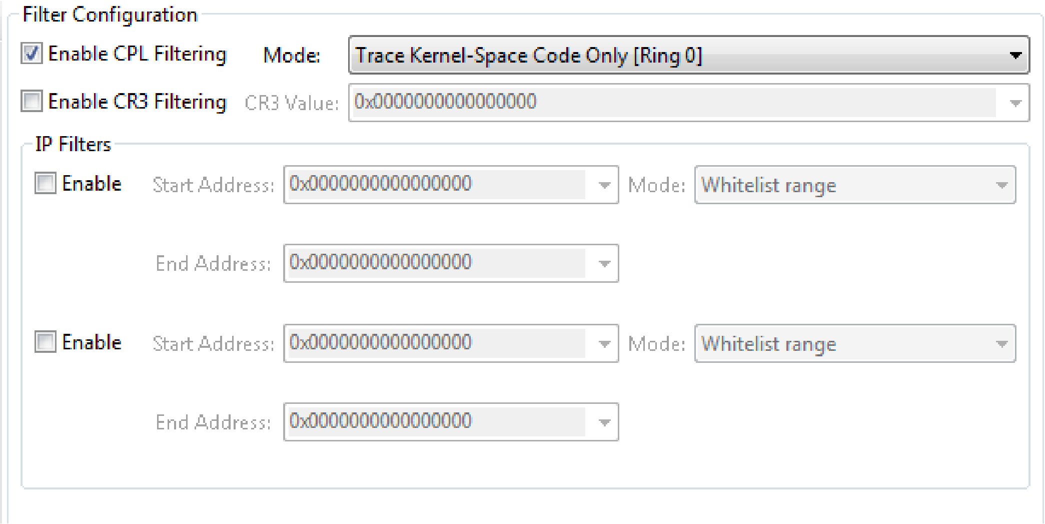
○ Advanced Configuration

Simple Configuration

Allocated Memory Address: 0x1000

Allocated Memory Size: 42KB

In the filtering tab, check Enable CPL Filtering for each Thread and select Trace Kernel-Space Code Only [Ring 0] Mode to collect Kernel space instruction only. If you want to gather user space only, then set filter as Trace User-Space Code Only [Ring 3] Mode



Filter Configuration

☑ Enable CPL Filtering Mode: Trace Kernel-Space Code Only [Ring 0]

□ Enable CR3 Filtering CR3 Value: 0x0000000000000000

IP Filters

□ Enable Start Address: 0x0000000000000000 Mode: Whitelist range

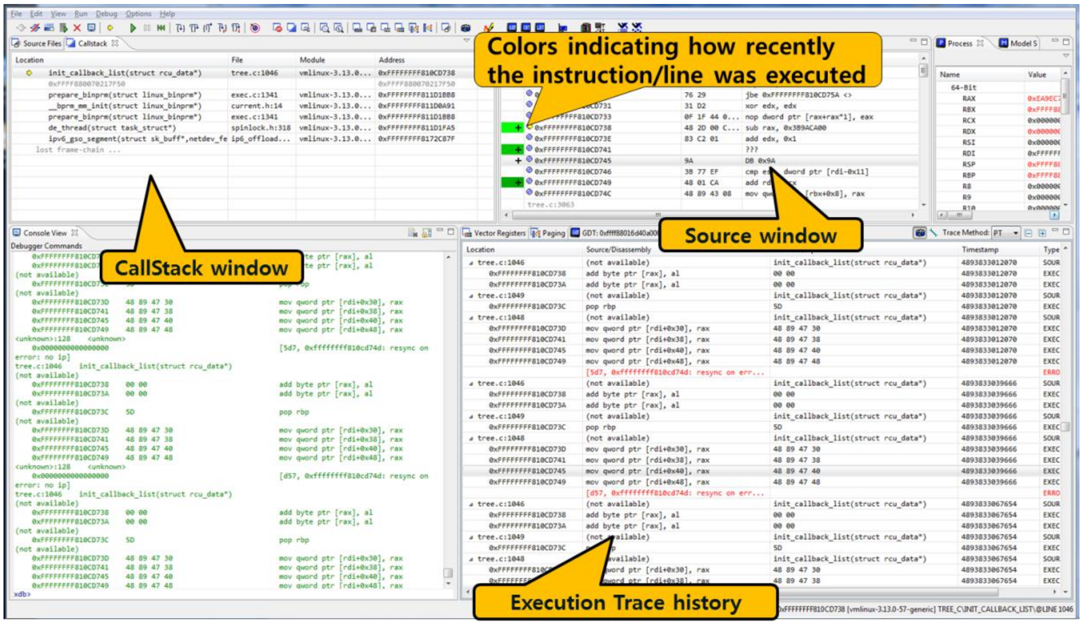
End Address: 0x0000000000000000

□ Enable Start Address: 0x0000000000000000 Mode: Whitelist range

End Address: 0x0000000000000000

Intel® Processor Trace Sample

Below picture is the layout of Intel® System Debugger with Processor Trace Window.



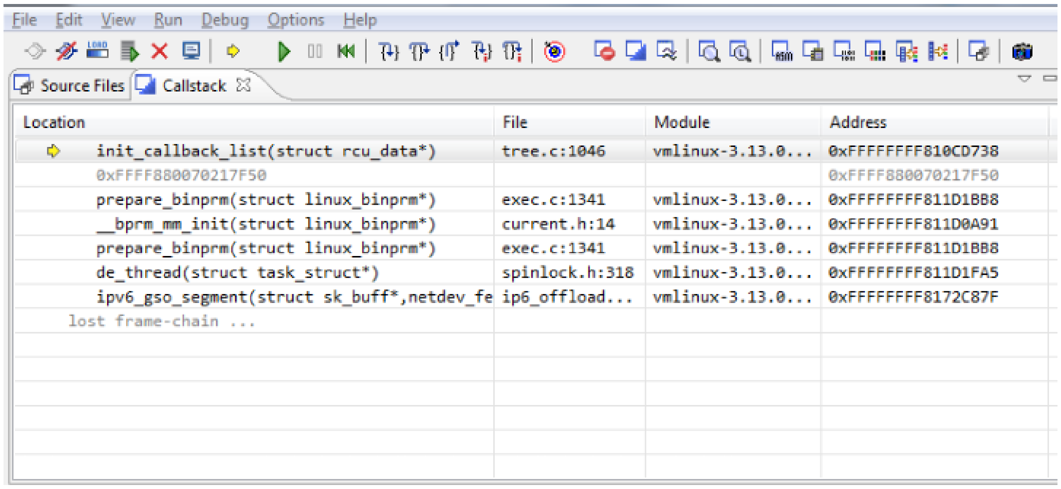
Colors indicating how recently the instruction/line was executed

Source window

CallStack window

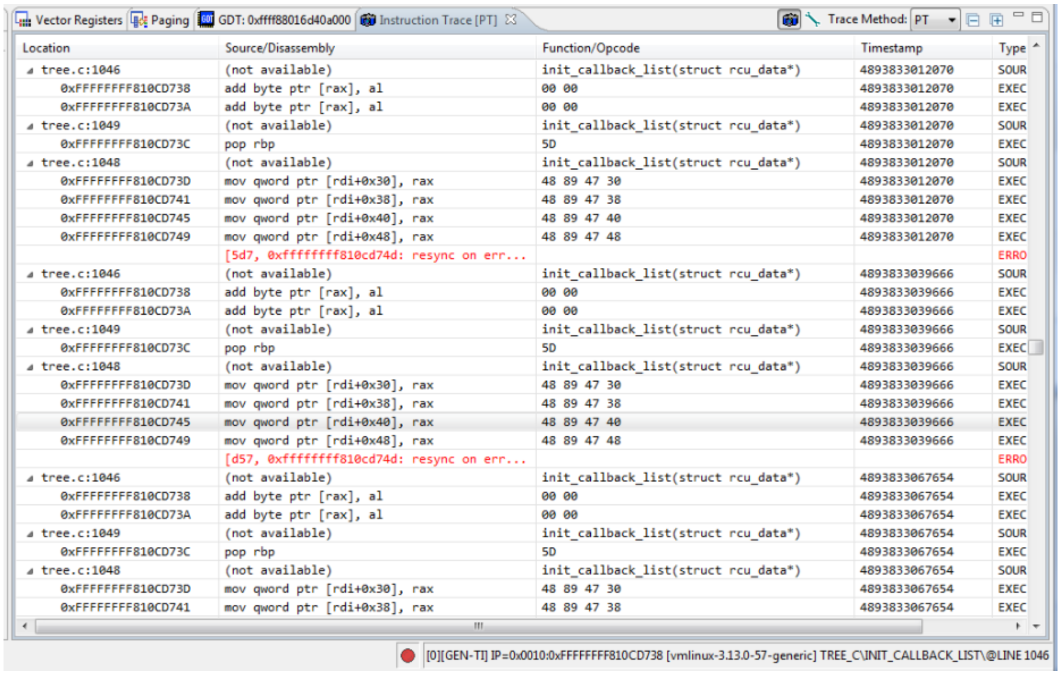
Execution Trace history

After stopping (suspend the execution), the call stack gives that it stays in the init_callback_list(struct rcu_data*) function in tree.c line number 1046.



Location	File	Module	Address
init_callback_list(struct rcu_data*)	tree.c:1046	vmlinux-3.13.0...	0xFFFFFFFF810CD738
0xFFFF880070217F50			0xFFFF880070217F50
prepare_binprm(struct linux_binprm*)	exec.c:1341	vmlinux-3.13.0...	0xFFFFFFFF811D1BB8
__bprm_mm_init(struct linux_binprm*)	current.h:14	vmlinux-3.13.0...	0xFFFFFFFF811D0A91
prepare_binprm(struct linux_binprm*)	exec.c:1341	vmlinux-3.13.0...	0xFFFFFFFF811D1BB8
de_thread(struct task_struct*)	spinlock.h:318	vmlinux-3.13.0...	0xFFFFFFFF811D1FA5
ipv6_gso_segment(struct sk_buff*,netdev_fe	ip6_offload...	vmlinux-3.13.0...	0xFFFFFFFF8172C87F
lost frame-chain ...			

In this example, system does not response any user input and whenever we stop it, it stay same file (tree.c). We easily guess that there might be a kind of loop.



Trace Method: PT

Location	Source/Disassembly	Function/Opcode	Timestamp	Type
tree.c:1046	(not available)	init_callback_list(struct rcu_data*)	4893833012070	SOUR
0xFFFFFFFF810CD738	add byte ptr [rax], al	00 00	4893833012070	EXEC
0xFFFFFFFF810CD73A	add byte ptr [rax], al	00 00	4893833012070	EXEC
tree.c:1049	(not available)	init_callback_list(struct rcu_data*)	4893833012070	SOUR
0xFFFFFFFF810CD73C	pop rbp	5D	4893833012070	EXEC
tree.c:1048	(not available)	init_callback_list(struct rcu_data*)	4893833012070	SOUR
0xFFFFFFFF810CD73D	mov qword ptr [rdi+0x30], rax	48 89 47 30	4893833012070	EXEC
0xFFFFFFFF810CD741	mov qword ptr [rdi+0x38], rax	48 89 47 38	4893833012070	EXEC
0xFFFFFFFF810CD745	mov qword ptr [rdi+0x40], rax	48 89 47 40	4893833012070	EXEC
0xFFFFFFFF810CD749	mov qword ptr [rdi+0x48], rax	48 89 47 48	4893833012070	EXEC
	[5d7, 0xffffffff810cd74d: resync on err...			ERRO
tree.c:1046	(not available)	init_callback_list(struct rcu_data*)	4893833039666	SOUR
0xFFFFFFFF810CD738	add byte ptr [rax], al	00 00	4893833039666	EXEC
0xFFFFFFFF810CD73A	add byte ptr [rax], al	00 00	4893833039666	EXEC
tree.c:1049	(not available)	init_callback_list(struct rcu_data*)	4893833039666	SOUR
0xFFFFFFFF810CD73C	pop rbp	5D	4893833039666	EXEC
tree.c:1048	(not available)	init_callback_list(struct rcu_data*)	4893833039666	SOUR
0xFFFFFFFF810CD73D	mov qword ptr [rdi+0x30], rax	48 89 47 30	4893833039666	EXEC
0xFFFFFFFF810CD741	mov qword ptr [rdi+0x38], rax	48 89 47 38	4893833039666	EXEC
0xFFFFFFFF810CD745	mov qword ptr [rdi+0x40], rax	48 89 47 40	4893833039666	EXEC
0xFFFFFFFF810CD749	mov qword ptr [rdi+0x48], rax	48 89 47 48	4893833039666	EXEC
	[d57, 0xffffffff810cd74d: resync on err...			ERRO
tree.c:1046	(not available)	init_callback_list(struct rcu_data*)	4893833067654	SOUR
0xFFFFFFFF810CD738	add byte ptr [rax], al	00 00	4893833067654	EXEC
0xFFFFFFFF810CD73A	add byte ptr [rax], al	00 00	4893833067654	EXEC
tree.c:1049	(not available)	init_callback_list(struct rcu_data*)	4893833067654	SOUR
0xFFFFFFFF810CD73C	pop rbp	5D	4893833067654	EXEC
tree.c:1048	(not available)	init_callback_list(struct rcu_data*)	4893833067654	SOUR
0xFFFFFFFF810CD73D	mov qword ptr [rdi+0x30], rax	48 89 47 30	4893833067654	EXEC
0xFFFFFFFF810CD741	mov qword ptr [rdi+0x38], rax	48 89 47 38	4893833067654	EXEC

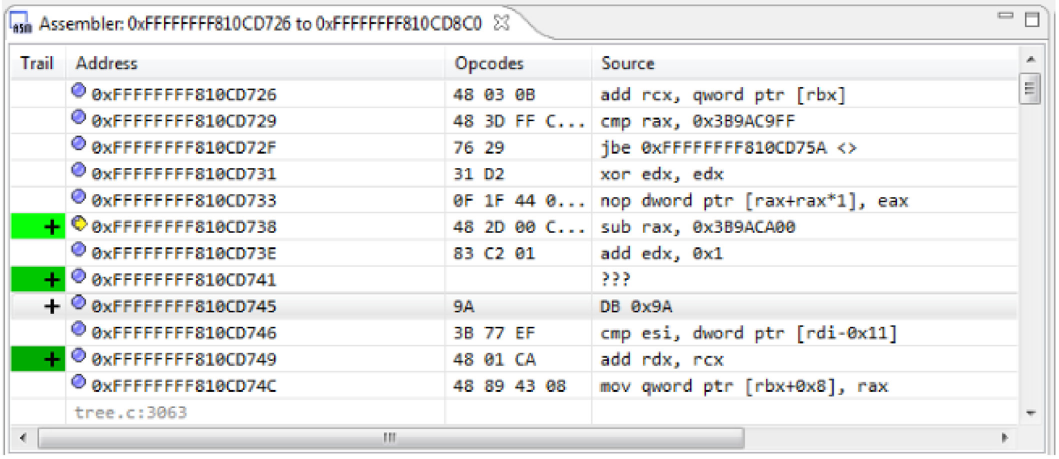
[0][GEN-TI] IP=0x0010:0xFFFFFFFF810CD738 [vmlinux-3.13.0-57-generic] TREE_C\INIT_CALLBACK_LIST\@LINE 1046

And you check the Instruction Trace windows. It shows that filename: tree.c and line number: 1046~1049 and disassembled instructions with functions and opcodes. It helps you:

- Find tracks the instruction flow in the last function of call stack.
- Find tracks the functions which are not exposed in the call stack because it returns already.

These capabilities help you to easily identify the problematic codes and functions when you debug the system and software.

Note the coloring that shows up on some lines in the assembler. We annotate the last hundred (or so) instructions that were executed with coloring information that indicates how recently an instruction was executed. The deeper the green, the more recently this instruction was executed. A "+" sign indicates that this instruction was executed multiple times in the last hundred instructions



Assembler: 0xFFFFFFFF810CD726 to 0xFFFFFFFF810CD8C0

Trail	Address	Opcodes	Source
	0xFFFFFFFF810CD726	48 03 0B	add rcx, qword ptr [rbx]
	0xFFFFFFFF810CD729	48 3D FF C...	cmp rax, 0x3B9AC9FF
	0xFFFFFFFF810CD72F	76 29	jbe 0xFFFFFFFF810CD75A <>
	0xFFFFFFFF810CD731	31 D2	xor edx, edx
	0xFFFFFFFF810CD733	0F 1F 44 0...	nop dword ptr [rax+rax*1], eax
+	0xFFFFFFFF810CD738	48 2D 00 C...	sub rax, 0x3B9ACA00
	0xFFFFFFFF810CD73E	83 C2 01	add edx, 0x1
+	0xFFFFFFFF810CD741		???
+	0xFFFFFFFF810CD745	9A	DB 0x9A
	0xFFFFFFFF810CD746	3B 77 EF	cmp esi, dword ptr [rdi-0x11]
+	0xFFFFFFFF810CD749	48 01 CA	add rdx, rcx
	0xFFFFFFFF810CD74C	48 89 43 08	mov qword ptr [rbx+0x8], rax
	tree.c:3063		

You can save the trace data as a file by using command line. (Saved example file is attached in the appendix.)

(1) set protfile “c:\temp\processor-trace-example.log”

(2) itrace “print”

Summary of Intel® Processor Trace key benefits:

- Intel® PT provides the context around all kinds of events. Performance profilers can use PT to discover the root causes of 'response-time' issues.
- Intel® PT enables a much deeper view into execution than has previously been commonly available; for example, loop behavior, from entry and exit down to specific back edges and loop trip counts.
- Reconstruct the code flow that led to the current location. Whether this is a crash site, a breakpoint, a watch point, or simply the instruction following a function call we just stepped over.
- Another important use case is debugging stack corruptions. When the call stack has been corrupted, normal frame unwinding usually fails or may not produce reliable results. Intel® PT can be used to reconstruct the stack back trace

Intel® PT can also help to narrow down data races in multi-threaded operating system and user program code. It can log the execution of all threads with a rough time indication.

References

- Processor Tracing
  https://software.intel.com/en-us/blogs/2013/09/18/processor-tracing
- Intel® 64 and IA-32 Architectures Software Developer’s Manual

Appendix

Example: Intel® Processor Trace and Call Stack Saved in XDB via command console

```
!F ! --------------------------------------------------------------------------------
!F !       Copyright (C) 2014-2015 Intel Corporation. All rights reserved.
!F !
!F ! This file is owned by Intel Corporation or its suppliers or licensors,
!F ! and is furnished under license.  It may only be used or copied in accordance
!F ! with the terms of that license.  Unless otherwise provided in that license,
!F ! it is provided AS IS, with no warranties of any kind, express or implied.
!F ! Except as expressly permitted by that license, neither Intel Corporation nor
!F ! its suppliers assumes any responsibility or liability for any errors or
!F ! inaccuracies that may appear herein. Except as expressly permitted by that
!F ! license, no part of the Software may be reproduced, stored in a retrieval
!F ! system, transmitted in any form, or distributed by any means without the
!F ! express written consent of Intel Corporation. Title to this material remains
!F ! with Intel Corporation or its suppliers and licensors.
!F ! --------------------------------------------------------------------------------
!F ! This hook does nothing by default. Override this behaviour if needed.
!F exit
!(EVA) execution stopped by "Halt Command break"
!(EVA) [vmlinux-3.13.0-57-generic] TREE_C¥INIT_CALLBACK_LIST¥@LINE 1046
STOP
!(ERR) E-2201: The requested translation table cannot be accessed.
show calls
!(EVA) current stack-frames:
!(EVA) 0xFFFFFFFF810CD738   init_callback_list(struct rcu_data*)   [ tree.c : line=1046 ]
!(EVA) 0xFFFF880070217F50   0xFFFF880070217F50
!(EVA) 0xFFFFFFFF811D1BB8   prepare_binprm(struct linux_binprm*)   [ exec.c : line=1341 ]
!(EVA) 0xFFFFFFFF811D0A91   __bprm_mm_init(struct linux_binprm*)   [ current.h : line=14 ]
!(EVA) 0xFFFFFFFF811D1BB8   prepare_binprm(struct linux_binprm*)   [ exec.c : line=1341 ]
!(EVA) 0xFFFFFFFF811D1FA5   de_thread(struct task_struct*)   [ spinlock.h : line=318 ]
```

```
!(EVA) 0xFFFFFFFF8172C87F   ipv6_gso_segment(struct sk_buff*,netdev_features_t)
[ ip6_offload.c : line=150 ]

!(ERR) E-2201: The requested translation table cannot be accessed.

!(EVA) lost frame-chain ...

itrace "print"

!(EVA) tree.c:1046    init_callback_list(struct rcu_data*)

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD738    00 00                                   add byte ptr [rax], al

!(EVA)     0xFFFFFFFF810CD73A    00 00                                   add byte ptr [rax], al

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD73C    5D                                      pop rbp

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD73D    48 89 47 30                             mov qword ptr [rdi+0x30], rax

!(EVA)     0xFFFFFFFF810CD741    48 89 47 38                             mov qword ptr [rdi+0x38], rax

!(EVA)     0xFFFFFFFF810CD745    48 89 47 40                             mov qword ptr [rdi+0x40], rax

!(EVA)     0xFFFFFFFF810CD749    48 89 47 48                             mov qword ptr [rdi+0x48], rax

!(EVA) <unknown>:128    <unknown>

!(EVA)

!(EVA)     0x0000000000000000                                            [5d7, 0xffffffff810cd74d:
resync on error: no ip]

!(EVA) tree.c:1046    init_callback_list(struct rcu_data*)

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD738    00 00                                   add byte ptr [rax], al

!(EVA)     0xFFFFFFFF810CD73A    00 00                                   add byte ptr [rax], al

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD73C    5D                                      pop rbp

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD73D    48 89 47 30                             mov qword ptr [rdi+0x30], rax

!(EVA)     0xFFFFFFFF810CD741    48 89 47 38                             mov qword ptr [rdi+0x38], rax

!(EVA)     0xFFFFFFFF810CD745    48 89 47 40                             mov qword ptr [rdi+0x40], rax

!(EVA)     0xFFFFFFFF810CD749    48 89 47 48                             mov qword ptr [rdi+0x48], rax

!(EVA) <unknown>:128    <unknown>
```

```
!(EVA)

!(EVA)     0x0000000000000000                                      [d57, 0xffffffff810cd74d:
resync on error: no ip]

!(EVA) tree.c:1046    init_callback_list(struct rcu_data*)

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD738    00 00                             add byte ptr [rax], al

!(EVA)     0xFFFFFFFF810CD73A    00 00                             add byte ptr [rax], al

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD73C    5D                                pop rbp

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD73D    48 89 47 30                       mov qword ptr [rdi+0x30], rax

!(EVA)     0xFFFFFFFF810CD741    48 89 47 38                       mov qword ptr [rdi+0x38], rax

!(EVA)     0xFFFFFFFF810CD745    48 89 47 40                       mov qword ptr [rdi+0x40], rax

!(EVA)     0xFFFFFFFF810CD749    48 89 47 48                       mov qword ptr [rdi+0x48], rax

!(EVA) <unknown>:128    <unknown>

!(EVA)

!(EVA)     0x0000000000000000                                      [4d7, 0xffffffff810cd74d:
resync on error: no ip]

!(EVA)     0x0000000000000000                                      [4d7, 0x0: sync error:
reached end of trace stream]

itrace "print"

!(EVA) tree.c:1046    init_callback_list(struct rcu_data*)

!(EVA) (not available)

!(EVA)     0xFFFFFFFF810CD738    00 00                             add byte ptr [rax], al

!(EVA)     0xFFFFFFFF810CD73A    00 00                             add byte ptr [rax], al
```