SONY

CE Linux Forum

Kernel Probes for MIPS, ARM and PPC32

Sony India Software Centre
Sony Corporation

Madhvesh Sulibhavi (madhvesh.s@ap.sony.com)
Lubna Badkar (lubna@ap.sony.com)
Tim Bird (tim.bird@am.sony.com) Presenting

Contents

Kprobes Overview

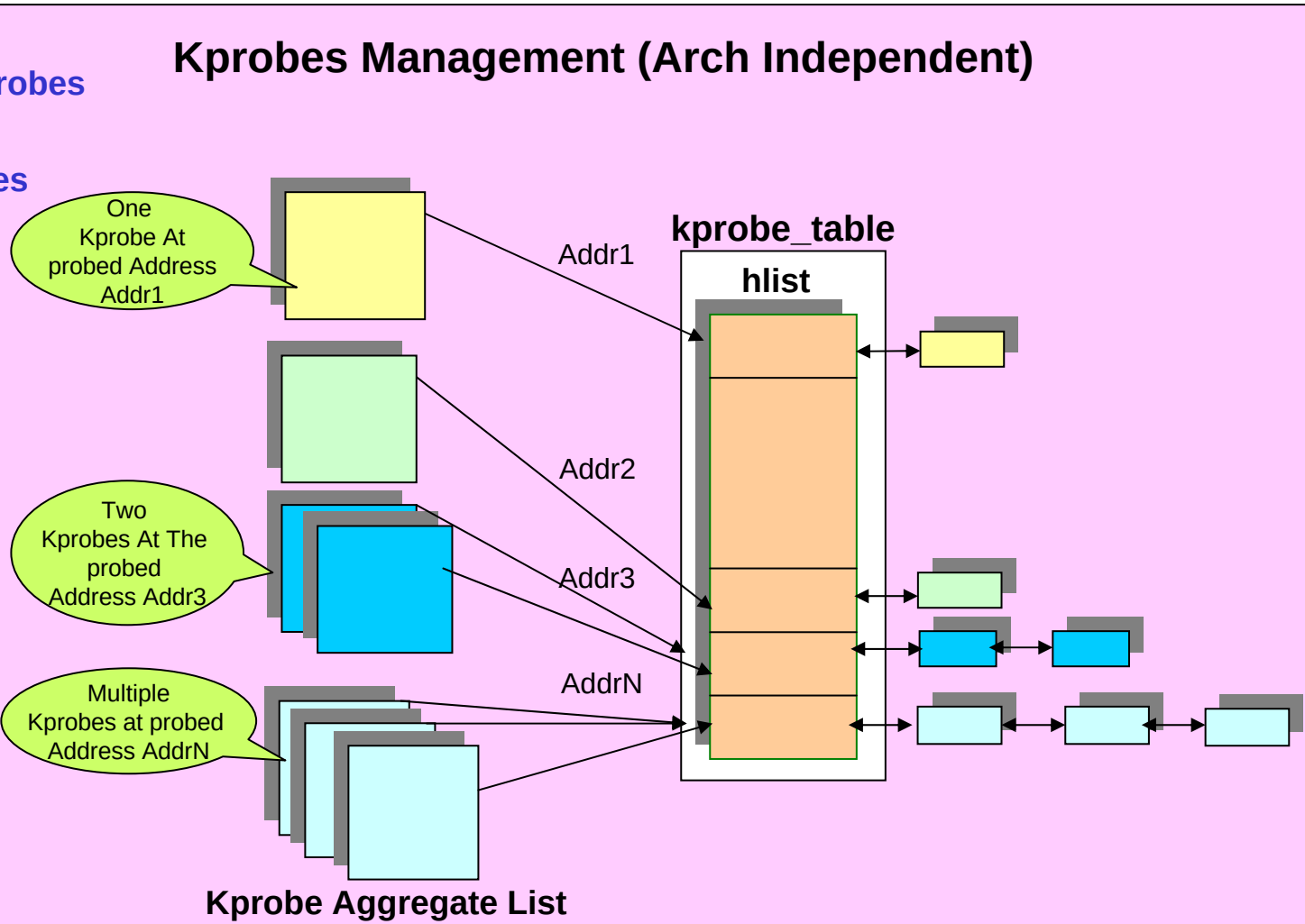
- Kprobes is simple and lightweight mechanism to collect debugging information dynamically.
- Linux main line kernel 2.6.16 (and later) includes support for X86, PPC64, Sparc, IA64, x86_64
- Kprobes support for MIPS, ARM and PPC32 arch is implemented by Sony for vanilla kernel 2.6.16-39.

Kprobes Overview (Contd)

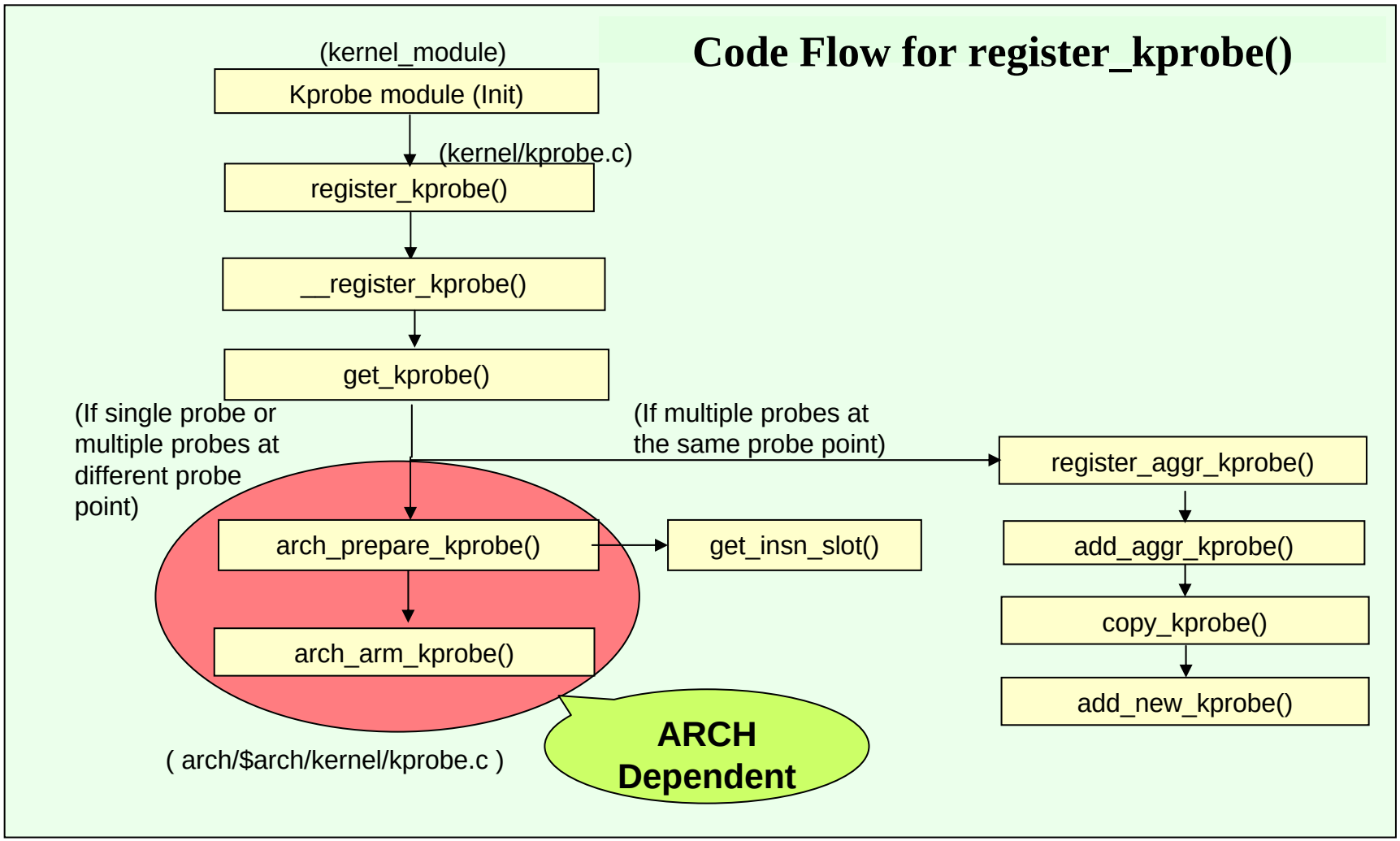
- Kprobe mainly contains following three handlers
- Pre-Handler
  - Executed before the probed instruction
  - It can be used to dump the register contents before executing the probed instruction
- Post-Handler
  - Executed after the probed instruction
  - It can be used to dump the register contents after executing the probed instructions
- Fault-Handler
  - Executed when some fault occurs in the pre handler or post handler or in the instruction being debugged

Kprobes Overview (Contd)

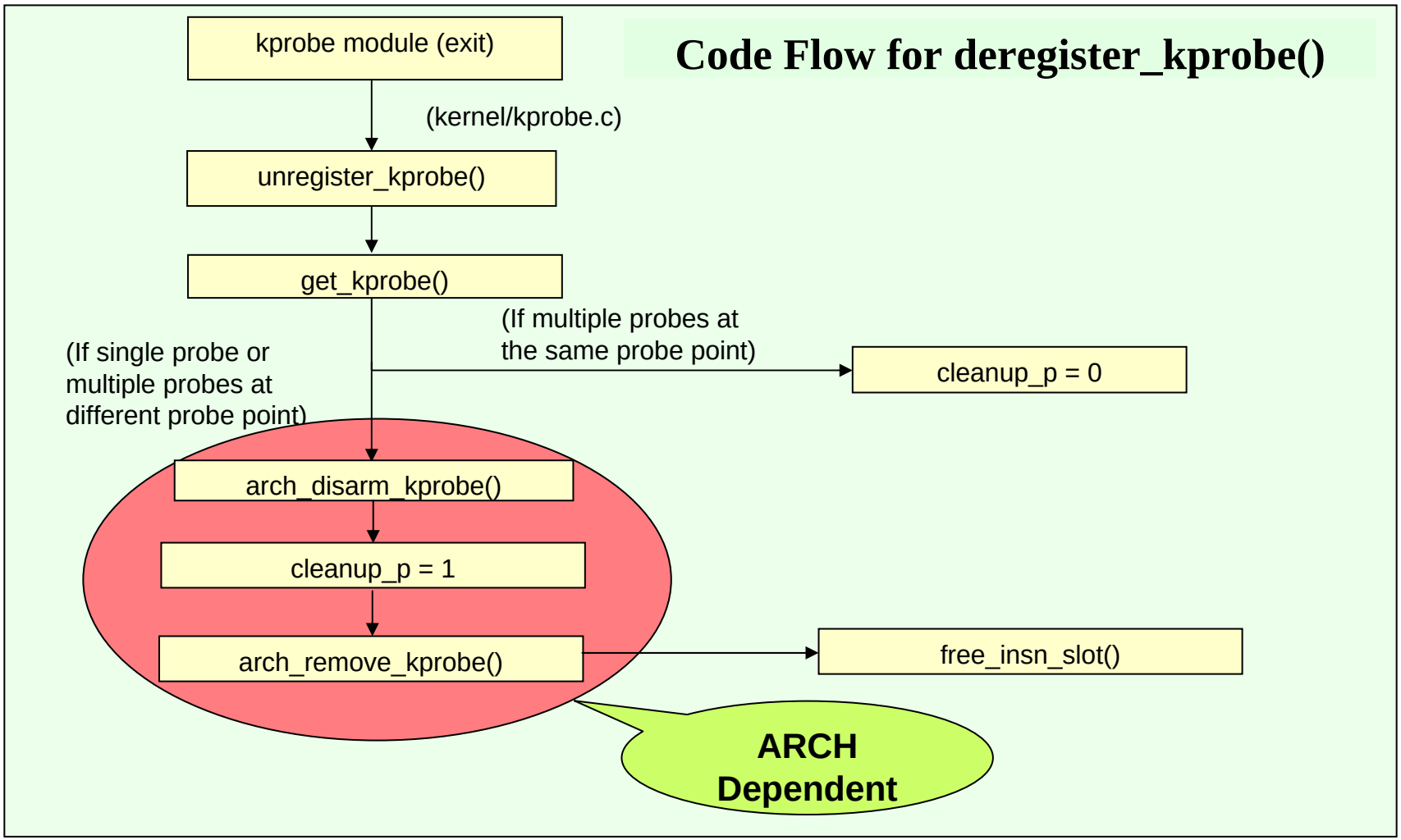
Includes:
-Initializing the kprobes
-Register kprobes
-Deregister kprobes



Kprobes Overview (Contd)



Kprobes Overview (Contd)

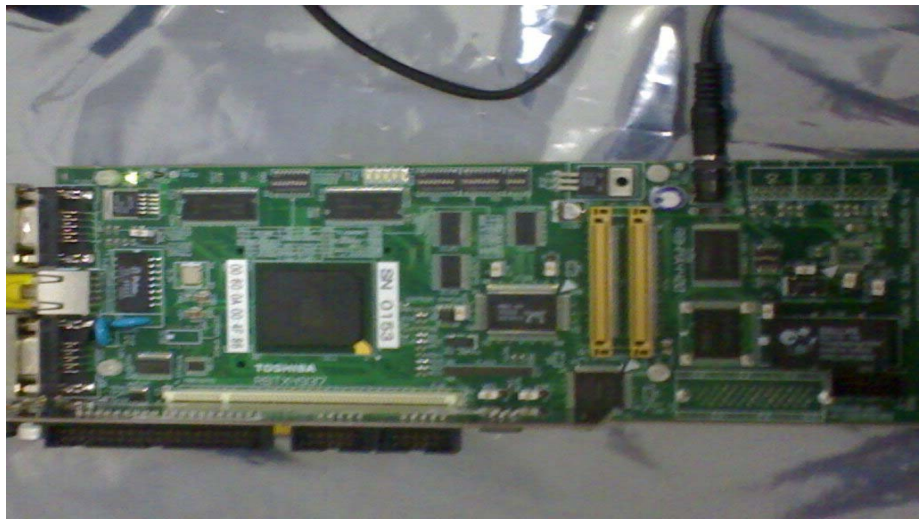


MIPS Arch Kprobes Contents

- Target Environment Details
- Kprobes Design
- Sample Example Output
- Sample Example Disassembly
- Limitations

MIPS Target Environment Details

- **Target Board**
  - Toshiba RBHMA4400 (TX4937)
- **OS**
  - Linux 2.6.16.39
- **Configuration**
  - **CONFIG_PREEMPT=y**
  - **CONFIG_KALLSYMS=y**
  - **CONFIG_KALLSYMSALL=y**
  - **CONFIG_DEBUG_KERNEL=y**



Hardware Features:
TX4937 @ 300Mhz.
System Clock @ 66Mhz
USB 1.1 Host Controller

MIPS Kprobes Design

- MIPS contains “break” instruction using which break point can be achieved.
- MIPS does not have hardware single step support and hence required alternate mechanism.
- Single step feature is implemented in software using “break” instruction with different code value
- MIPS Linux Kernel supports the following different break codes and two of these are used to implement Kprobes

MIPS Kprobes Design (Contd)

Opcode		Use
BRK_USERBP	0	User bp (used by debuggers)
BRK_KERNELBP	1	Break in the kernel
BRK_ABORT	2	Sometimes used by abort(3) to SIGIOT
BRK_BD_TAKEN	3	For bd slot emulation - not implemented
BRK_BD_NOTTAKEN	4	For bd slot emulation - not implemented
BRK_SSTEPBP	5	User bp (used by debuggers)
BRK_OVERFLOW	6	Overflow check
BRK_DIVZERO	7	Divide by zero check
BRK_RANGE	8	Range error check
BRK_STACKOVERFLOW	9	For Ada stack checking
BRK_NORLD	10	No rld found - not used by Linux/MIPS
_BRK_THREADBP	11	For threads, user bp (used by debuggers)
BRK_BUG	512	Used by BUG()
BRK_KDB	513	Used in KDB_ENTER()
BRK_MULOVF	1023	Multiply overflow

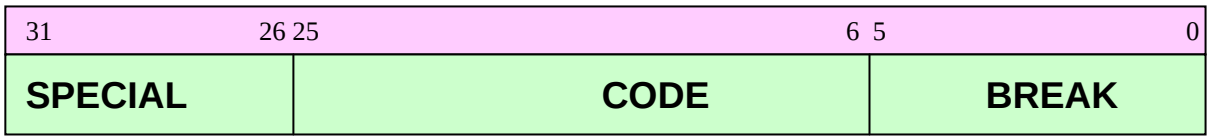
Linux Kernel Supported Break Codes in asm-mips/break.h

This is Used for Kprobe to Implement BreakPoint

This is Used for Kprobe to Implement Single Step Operation

MIPS Kprobes Design (Contd)

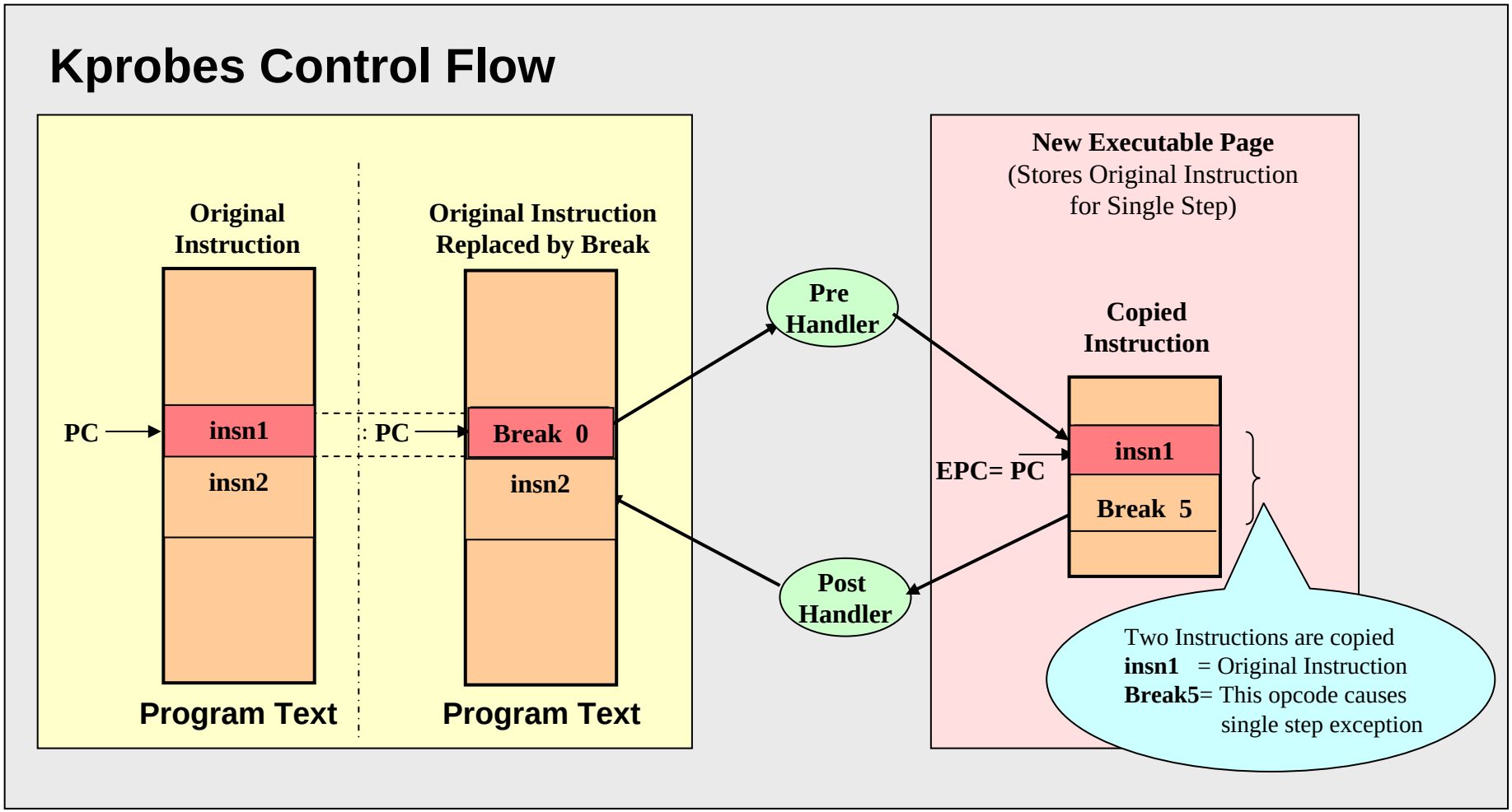
- Break instruction format is



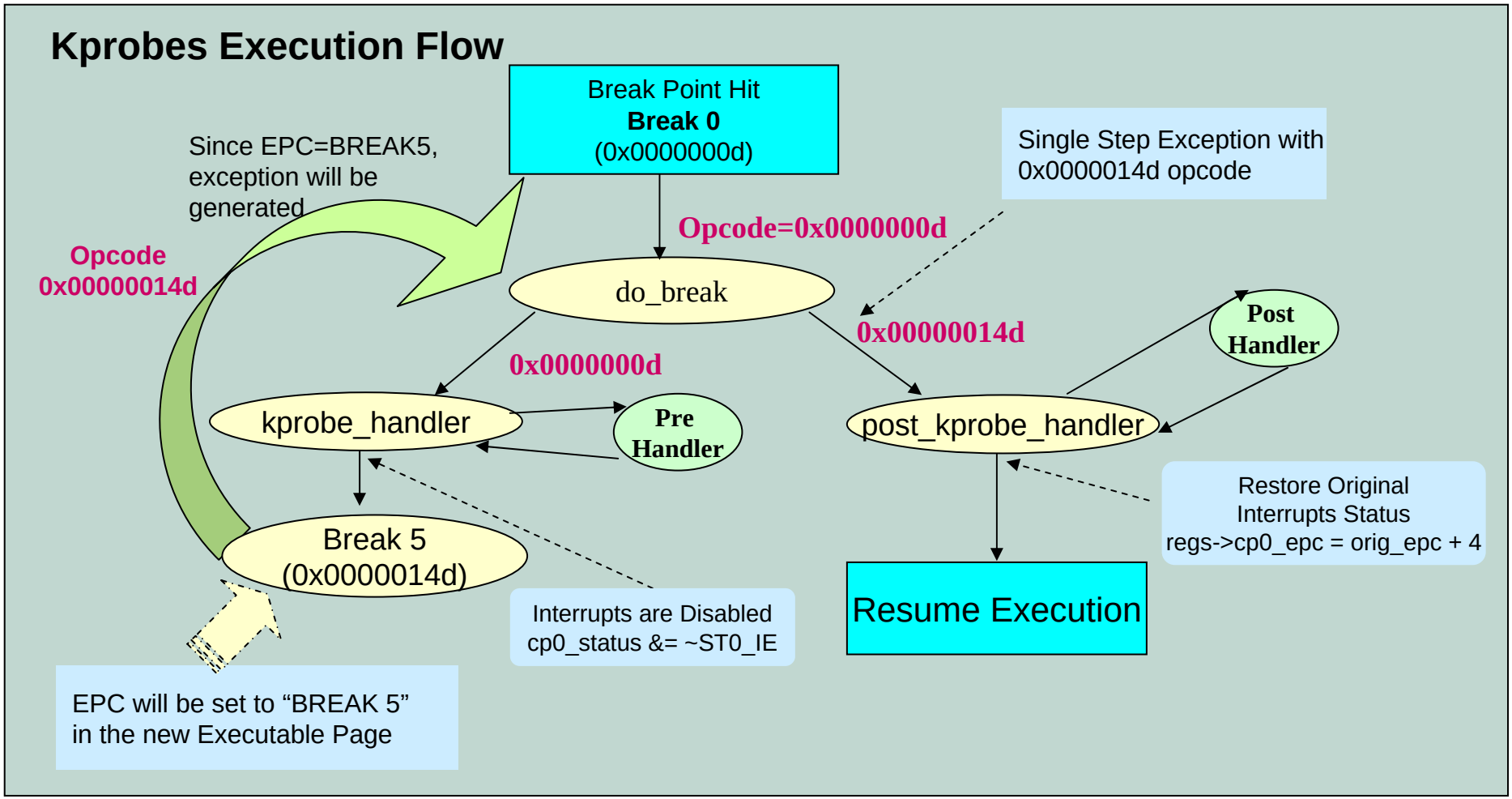
- When exception is generated, Exception Program Counter (EPC) register holds the instruction which raised the exception. Using the EPC value, the “code” field is extracted to check whether the exception is of BREAK0 or BREAK5 type

- By using different values for ***CODE*** field, the break point and single step operations are achieved

MIPS Kprobes Design (Contd)



MIPS Kprobes Design (Contd)



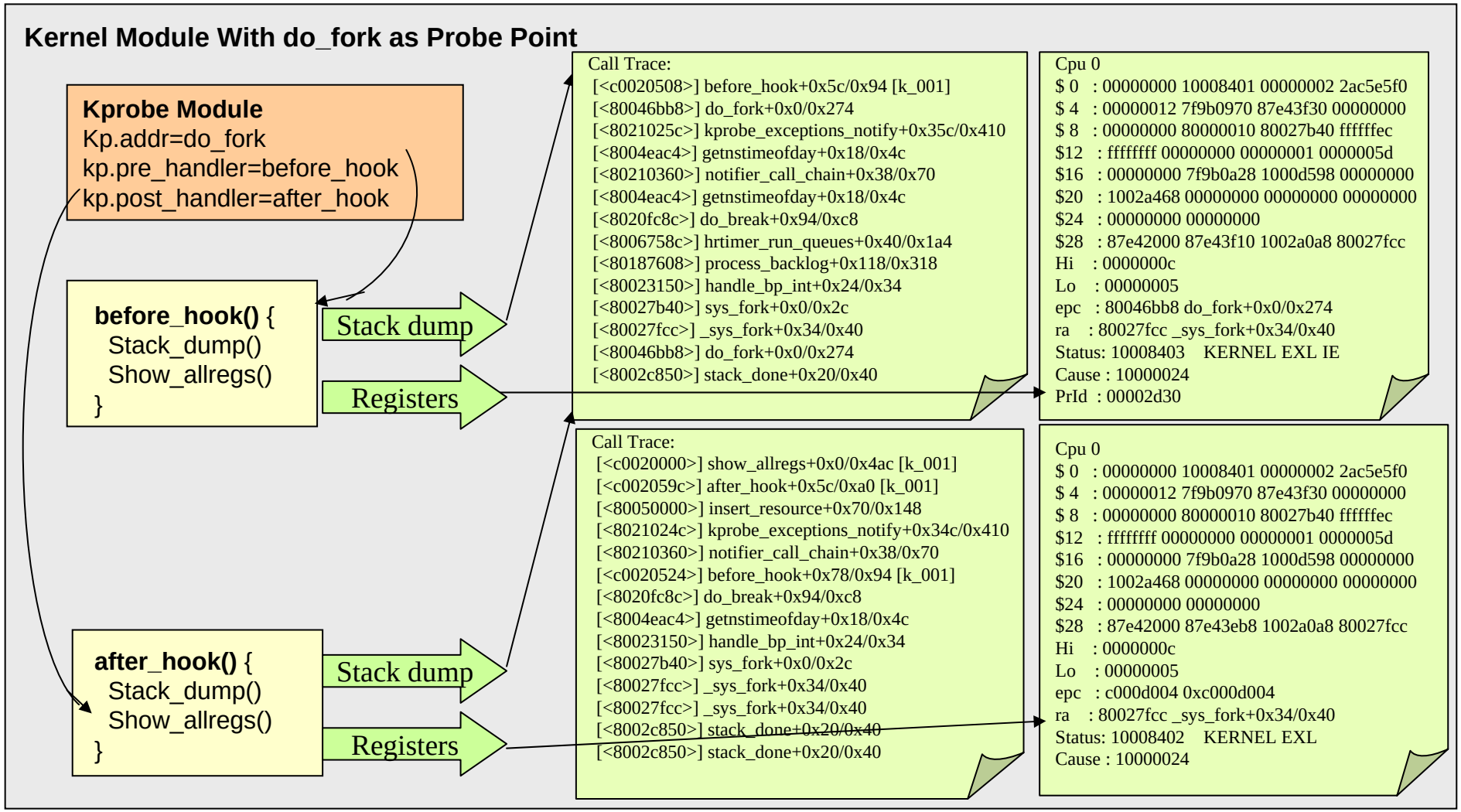
MIPS Kprobes Design (Contd)

- Kprobes Control Flow Details
  - When a particular instruction “insn1” is probed, the original instruction will be copied on the executable page and replaces the probed instruction by ‘break 0’ i.e 0x0000000d
  - ‘break 5’ (0x0000014d) will be copied on the executable page after the original probed instruction
  - A breakpoint exception occurs, immediately and unconditionally transferring control to the exception handler
  - When break 0 is hit, kprobes exception notifier will be called by the exception handler do_break() with the die value DIE_BREAK. This will in turn call kprobe_handler() which performs executing pre_handler() associated with kprobes

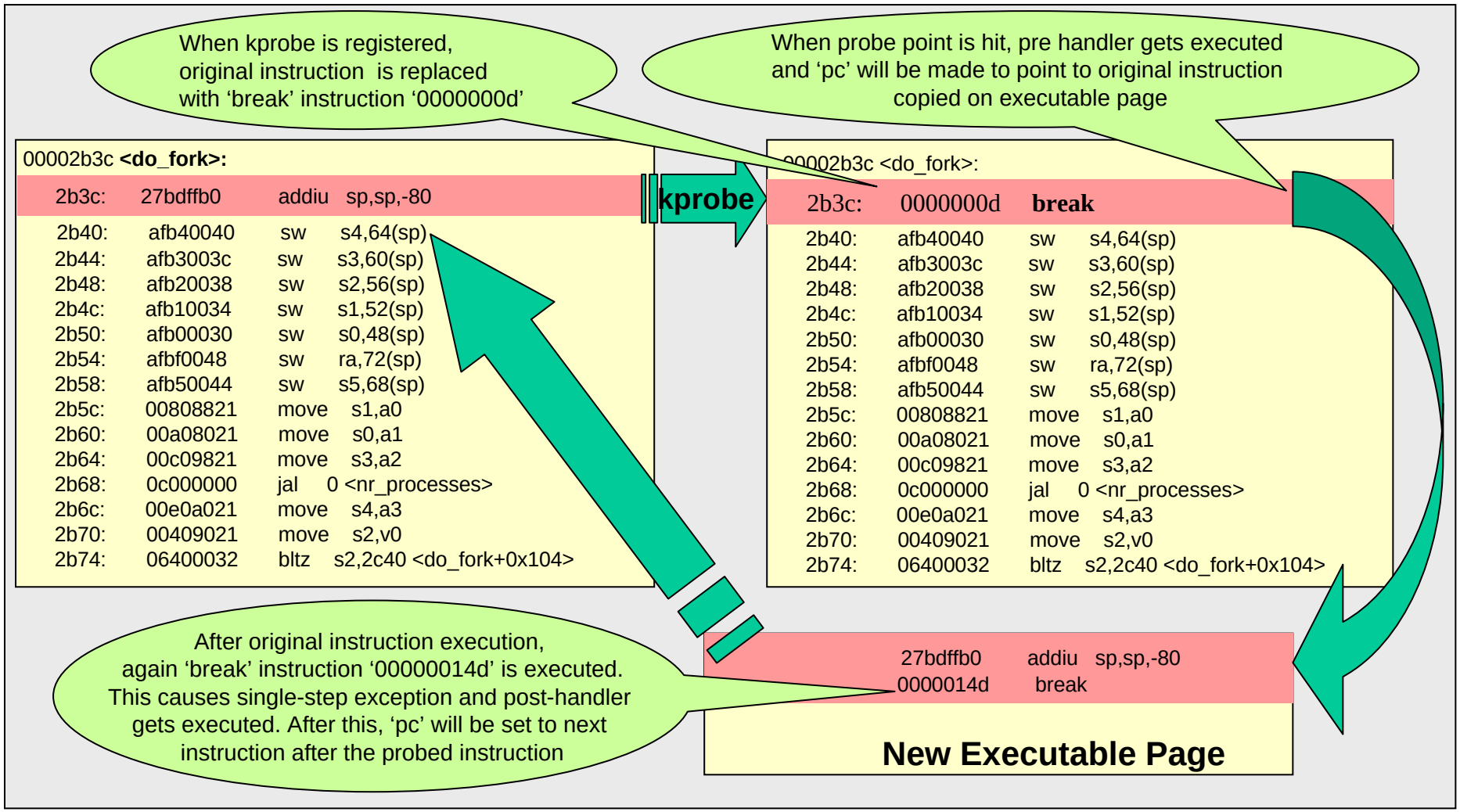
MIPS Kprobes Design (Contd)

- Kprobes Control Flow Details (Contd)
  - kprobe_handler() takes care of moving program counter to the copied instruction by calling prepare_single_step().
  - After executing the probed instruction, the next instruction on the copied page ‘break 5’ will be executed
  - When break 5 is hit, again kprobes exception notifier will be called by the exception handler do_break() with the die value DIE_SSTEPBP. This will in turn call user defined post_handler().
  - After the instruction debugging, the execution resumes to the next instruction “insn2”

MIPS Sample Example Output



MIPS Example Disassembly



MIPS Kprobes Limitations

- Only minimal testing is performed and needs to be tested using system tap test suites
- Not tested under SMP configuration
- Following instruction types are not supported
  - Jump register or jump and link register (ex: jr, jalr)
  - Jump And Link and Jump (ex: jal, j)
  - Branch instructions (ex: bltz, bgez, beq, bne, blez, bgtz)
  - coprocessor instructions (ex: cop1)

ARM Arch Kprobes Contents

- Target Environment Details
- Kprobes Design
- Sample Example Output
- Sample Example Disassembly
- Limitations

ARM Target Environment Details

- **Target Board**
  - OMAP Starter Kit (OSK 5912)
- **OS**
  - Linux 2.6.16.39
- **Configuration**
  - **CONFIG_PREEMPT=y**
  - **CONFIG_KALLSYMS=y**
  - **CONFIG_KALLSYMSALL=y**
  - **CONFIG_DEBUG_KERNEL=y**

Hardware Features:
OSK5912 @ 192 Mhz.
32 M Byte Flash ROM
USB Host Port

ARM Kprobes Design

- ARM arch does not have break or single step instructions
- Kprobes is implemented using one of the exception handling mechanism

ARM Supports Below Exception Types

Exception	Mode	Purpose
Fast Interrupt Request	FIQ	Fast interrupt handling
Interrupt Request	IRQ	Normal interrupt handling
SWI and RESET	SVC	Protected mode for OS
Pre-fetch or data abort	ABT	Memory protection handling
Undefined Instruction	UND	SW emulation of HW coprocessors

Undefined Exception Type is used for Kprobes

ARM Kprobes Design (Contd)

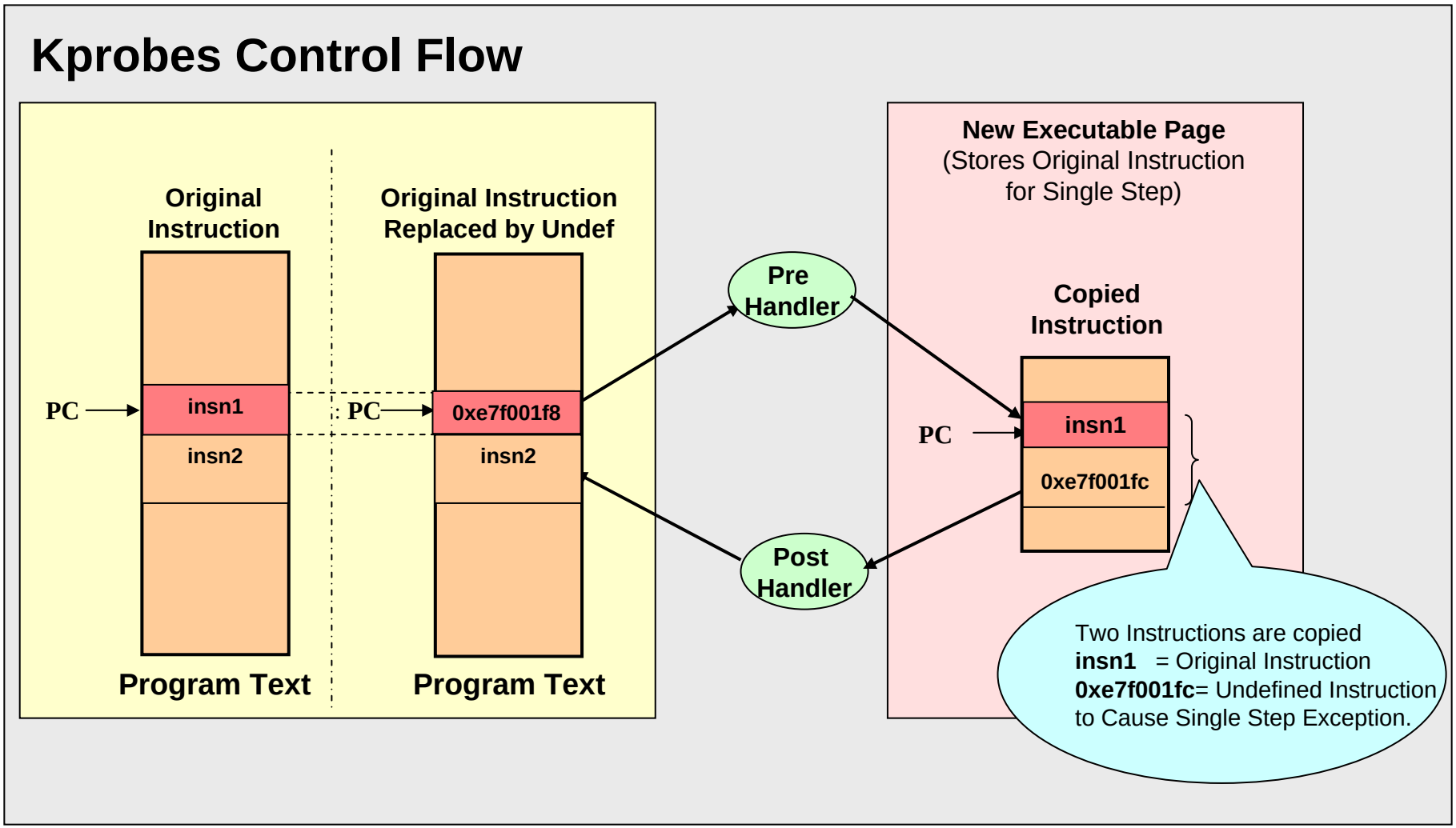
- Undefined instructions are those which the processor cannot decode in the absence of coprocessor. So this is used for kprobes implementation.
- Undefined Instruction opcode is as below

- If the “cond” field opcode[31:28] value = 1110, then it indicates the unconditional type.

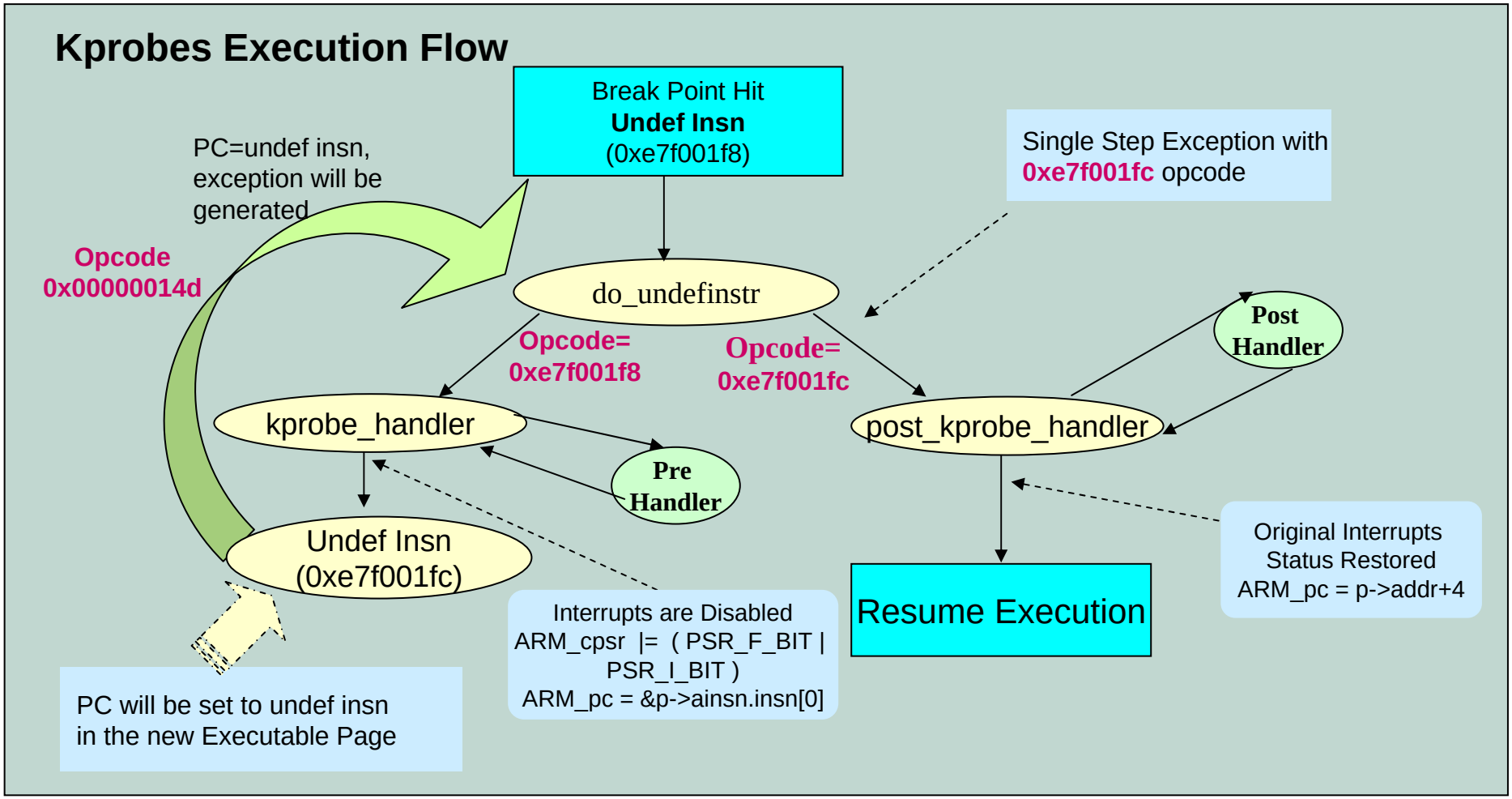
ARM Kprobes Design (Contd)

- When undefined exception is generated, Link Register (LR) will be updated with appropriate return address and this is used to implement single step operation and to resume the next instruction of the probe point
- In the exception handler, “die notifier” kernel feature is used to differentiate breakpoint and single step exceptions

ARM Kprobes Design (Contd)



ARM Kprobes Design (Contd)



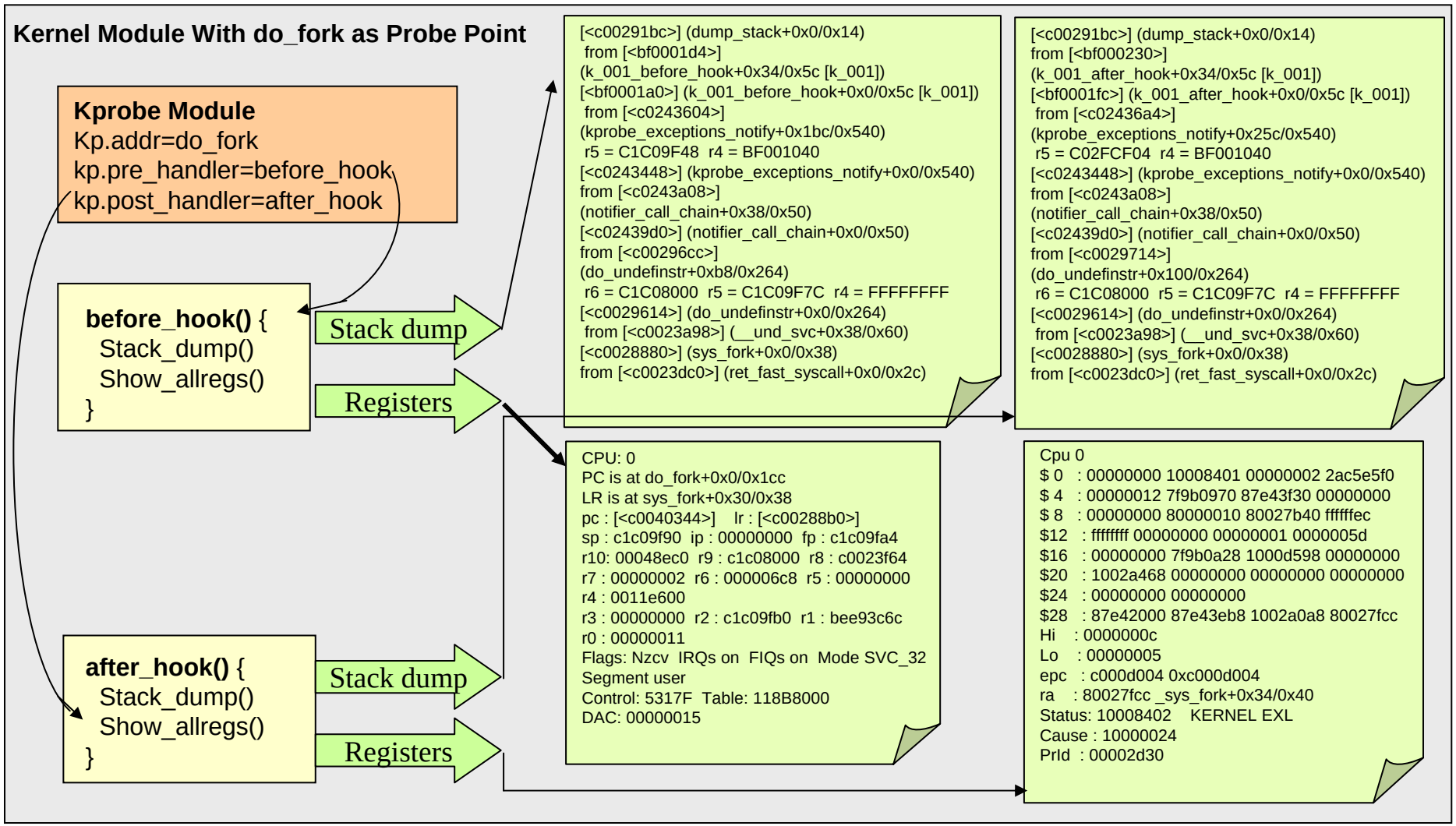
ARM Kprobes Design (Contd)

- Kprobes Control Flow Details
  - When a particular instruction “insn1” is probed, the original instruction will be copied on the executable page and replaces the probed instruction by undefined instruction ‘0xe7f001f8’ i.e BREAKPOINT_INSTRUCTION
  - Undefined Instruction with opcode 0xe7f001fc (BREAKPOINT_INSTRUCTION_2) will be copied on the executable page
  - When “BREAKPOINT_INSTRUCTION” is hit, kprobes exception notifier will be called by the exception handler do_undefinstr() with the die value DIE_UNDEF_1. This will in turn call kprobe_handler() which executes pre_handler() associated with kprobes

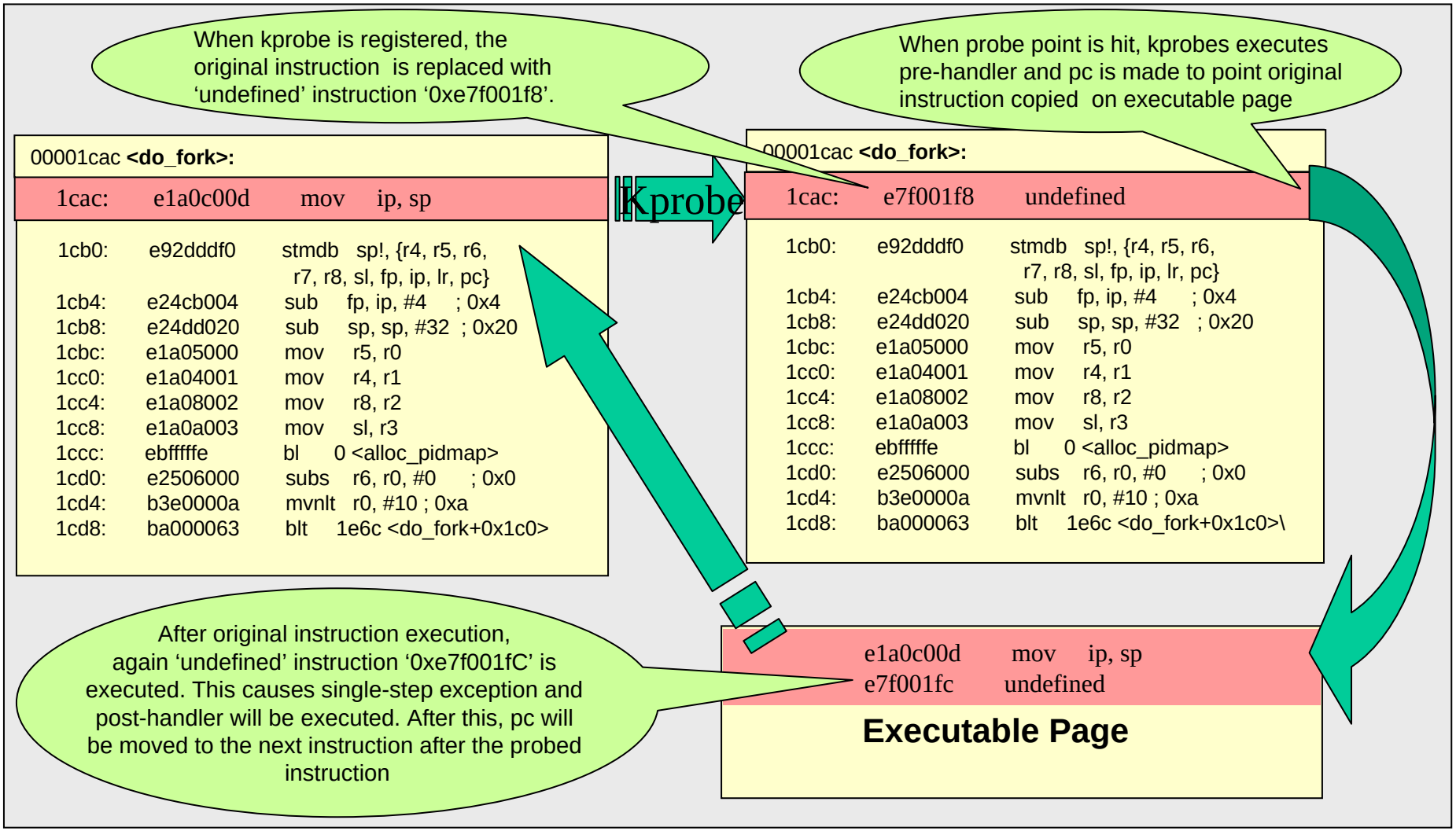
ARM Kprobes Design (Contd)

- Kprobes Control Flow Details (Contd)
  - kprobe_handler() takes care of moving program counter to the copied instruction by calling prepare_single_step().
  - After executing the probed instruction, the next instruction on the copied page ‘BREAKPOINT_INSTRUCTION_2’ will be executed
  - When BREAKPOINT_INSTRUCTION_2 is hit, again kprobes exception notifier will be called by the exception handler do_break() with the die value DIE_UNDEF_2. This will in turn calls post_kprobe_handler ().
  - After the instruction debugging, the execution resumes to the next instruction “insn2”

ARM Sample Example Output



ARM Example Disassembly



ARM Kprobes Limitations

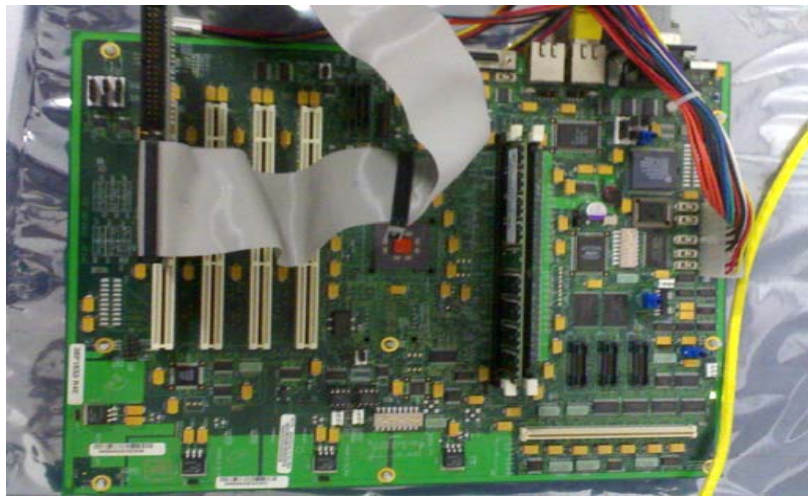
- Only minimal testing is performed and needs to be tested using system tap test suites
- Not tested under SMP configuration
- Following instruction types are not supported
  - Branch/link with exchange instruction type (ex: bx or blx)
  - Data processing instructions whose destination register is a PC (ex: AND, EOR, SUB, RSB, ADD, ADC, SBC, RSC, MOV, BIC, MVN)
  - load instructions which destination register is PC (ex:ldr)
  - Multiple load type instructions (ex: ldm)
  - Branch and branch with link type (Ex: b, bl)

PPC32 Arch Kprobes Contents

- Target Environment Details
- Kprobes Design
- Sample Example Output
- Sample Example Disassembly
- Limitations

PPC32 Target Environment Details

- **Target Board**
  - Ebony (PPC440gp)
- **OS**
  - Linux 2.6.16.39
- **Configuration**
  - **CONFIG_PREEMPT=y**
  - **CONFIG_KALLSYMS=y**
  - **CONFIG_KALLSYMSALL=y**
  - **CONFIG_DEBUG_KERNEL=y**



Hardware Features:

440GP Processor @ 500 Mhz

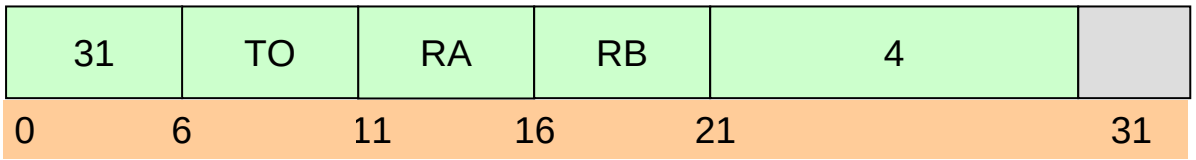
System Clock @ 66.66Mhz

PPC32 Kprobes Design

- Kprobe support for ppc64 is present in linux main line tree
- Kprobe support for PPC32 was posted recently http://patchwork.ozlabs.org/linuxppc/patch?id=9244
- But the above patch addresses only for non BookE versions.
- Sony has provided kprobes support for PPC32 BookE versions
- PPC32 with BookE versions involved technical challenges while addressing single step exception handling. These issues are highlighted and also the solution to overcome these is discussed in detail.

PPC32 Kprobes Design (Contd)

- PPC32 Arch does have trap instructions ‘tw'.
- Trap instruction format is



- There are different variations of trap instruction.
- This variation is mainly due to ‘TO’ field of the trap instruction
- In general trap instruction will be of the type 0x7c000008
- The trap used for kprobe is an unconditional trap “tw” i.e ‘0x7fe00008’ for which TO =31 and RA=RB=0

PPC32 Kprobes Design (Contd)

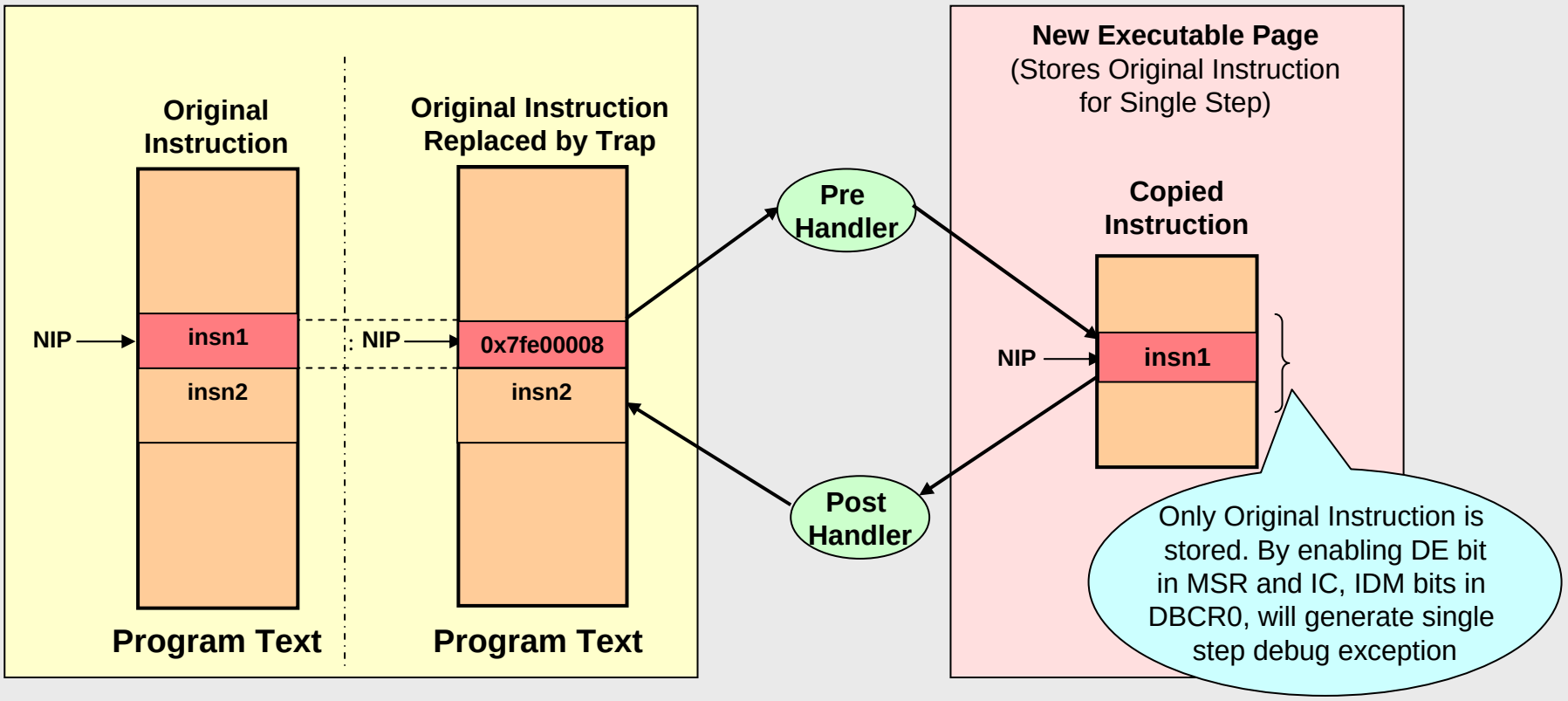
Different types of trap instruction are as follows

Code	Description	TO Encoding
twlt	Less than	16
Twle	Less than or equal	20
Tweq	Equal	4
Twge	Greater than or equal	12
Twgt	Greater than	8
Twnl	Not less than	12
Twne	Not equal	24
Twng	Not greater than	20
Twllt	Logically less than	2
Twlle	Logically less than or equal	6
Twlge	Logically greater than or equal	5
Twlgt	Logically greater than	1
Twlnl	Logically not less than	5
Twlng	Logically not greater than	6
Tw	**Unconditional**	**31**

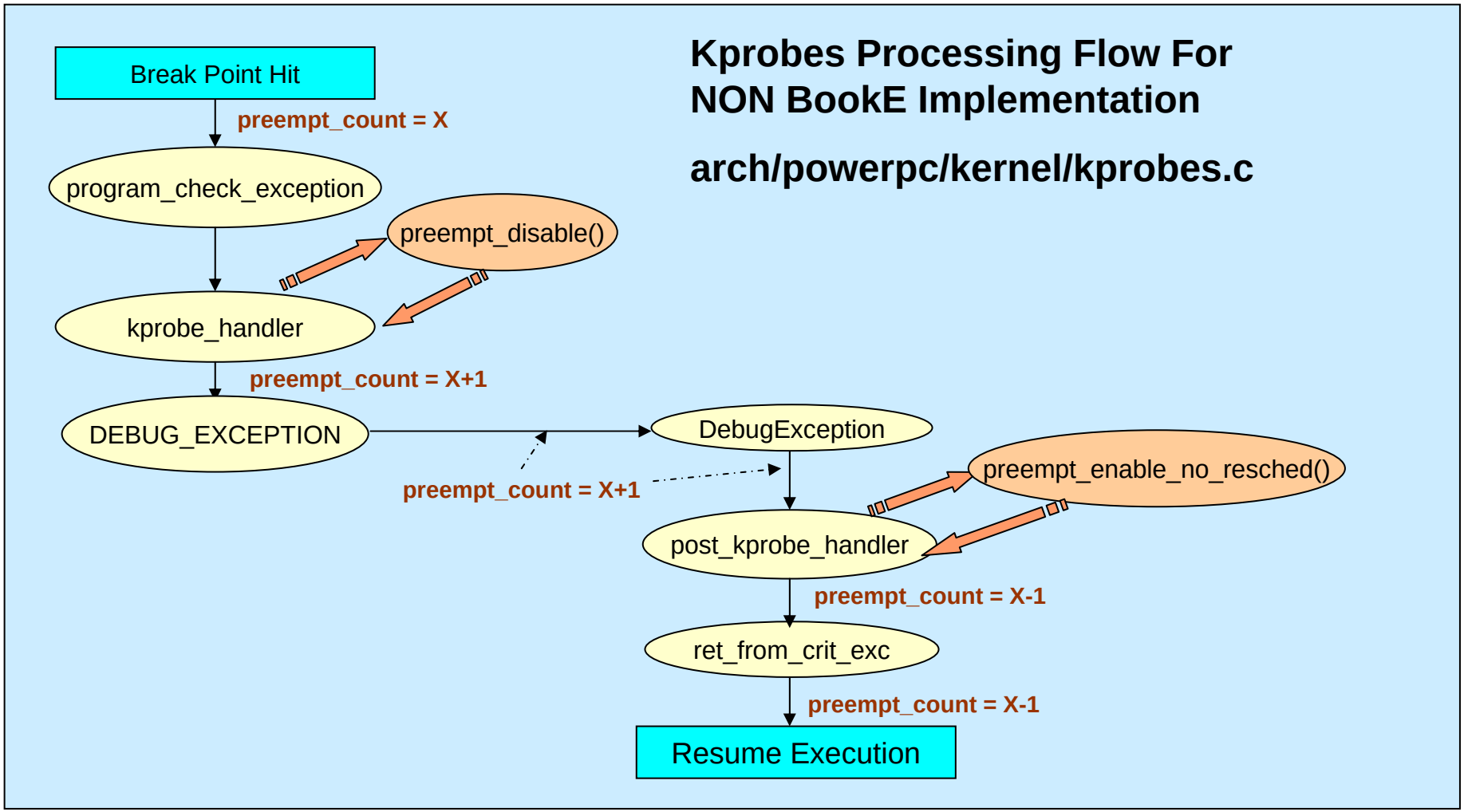
This is Used by Kprobes to Implement Break Point Exception

PPC32 Kprobes Design (Contd)

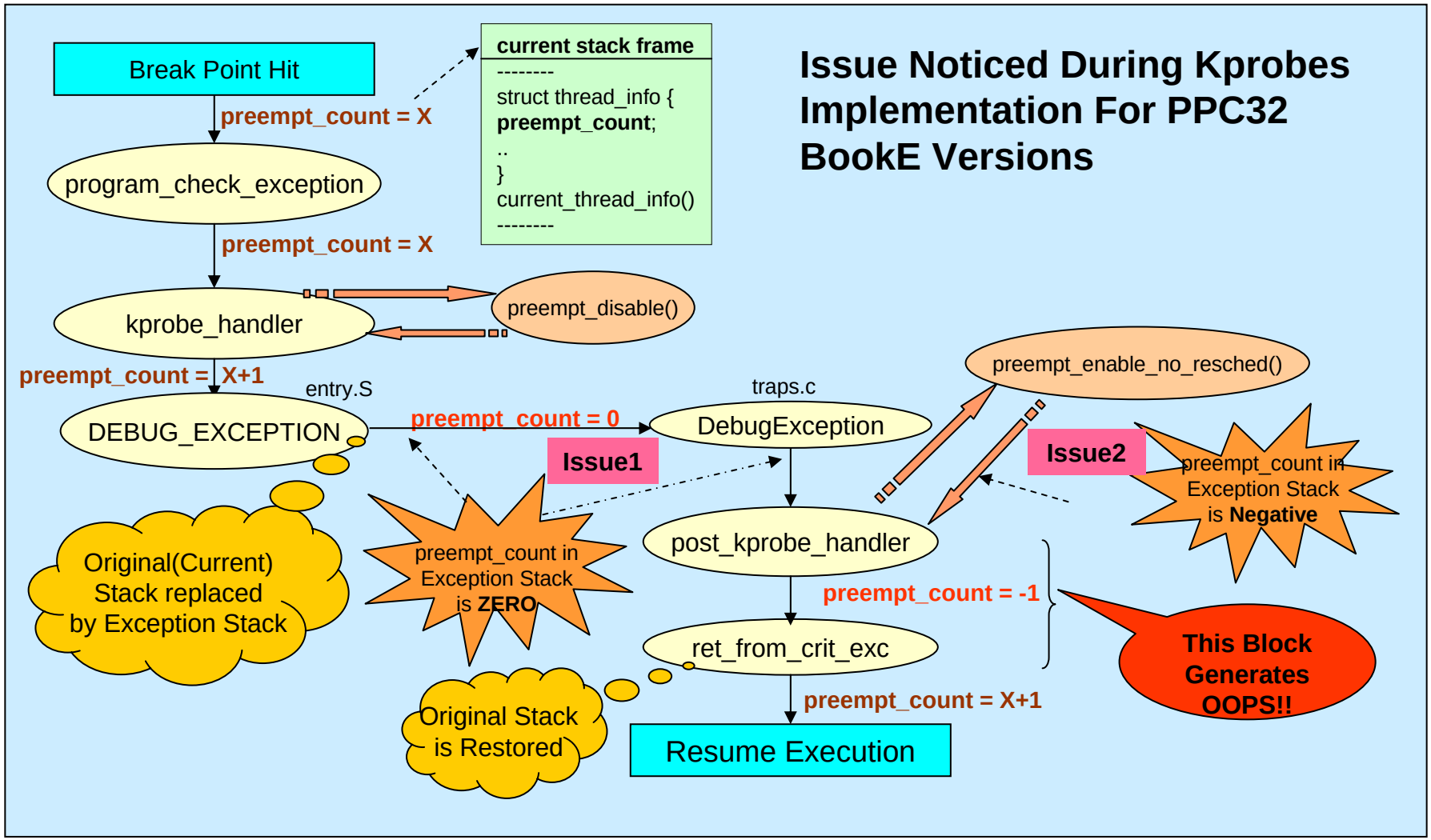
Kprobes Control Flow



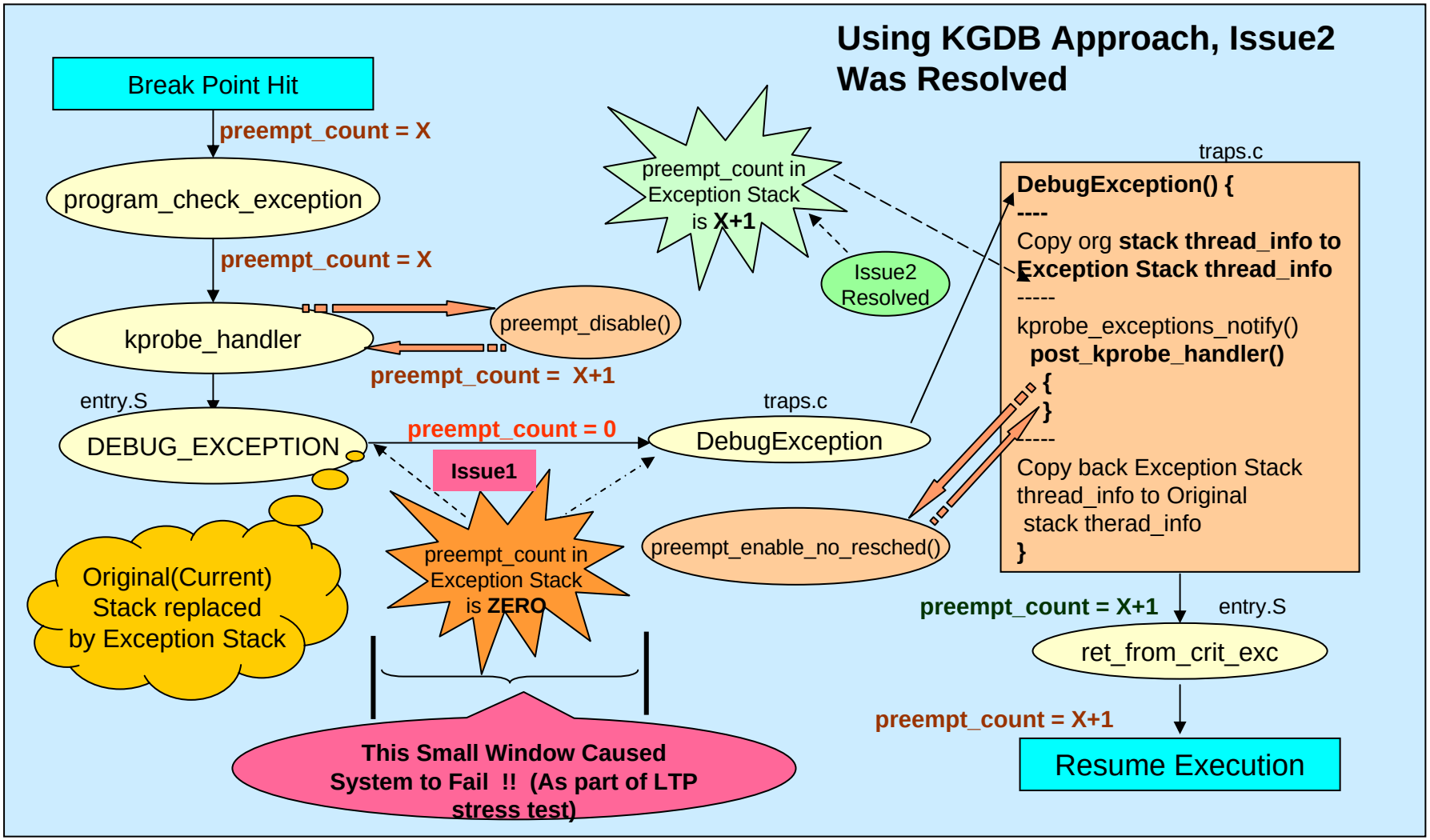
PPC32 Kprobes Design (Contd)



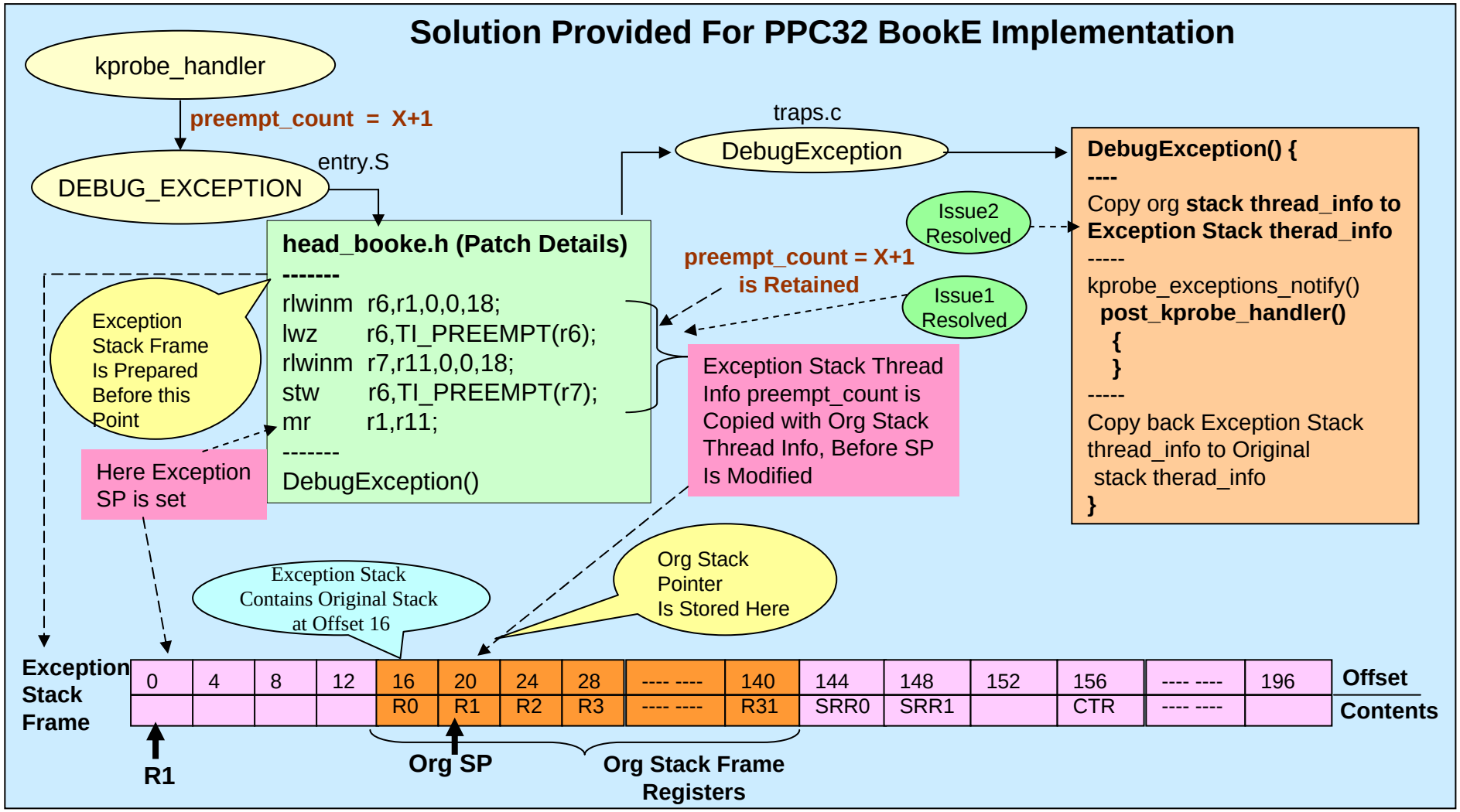
PPC32 Kprobes Design (Contd)



PPC32 Kprobes Design (Contd)

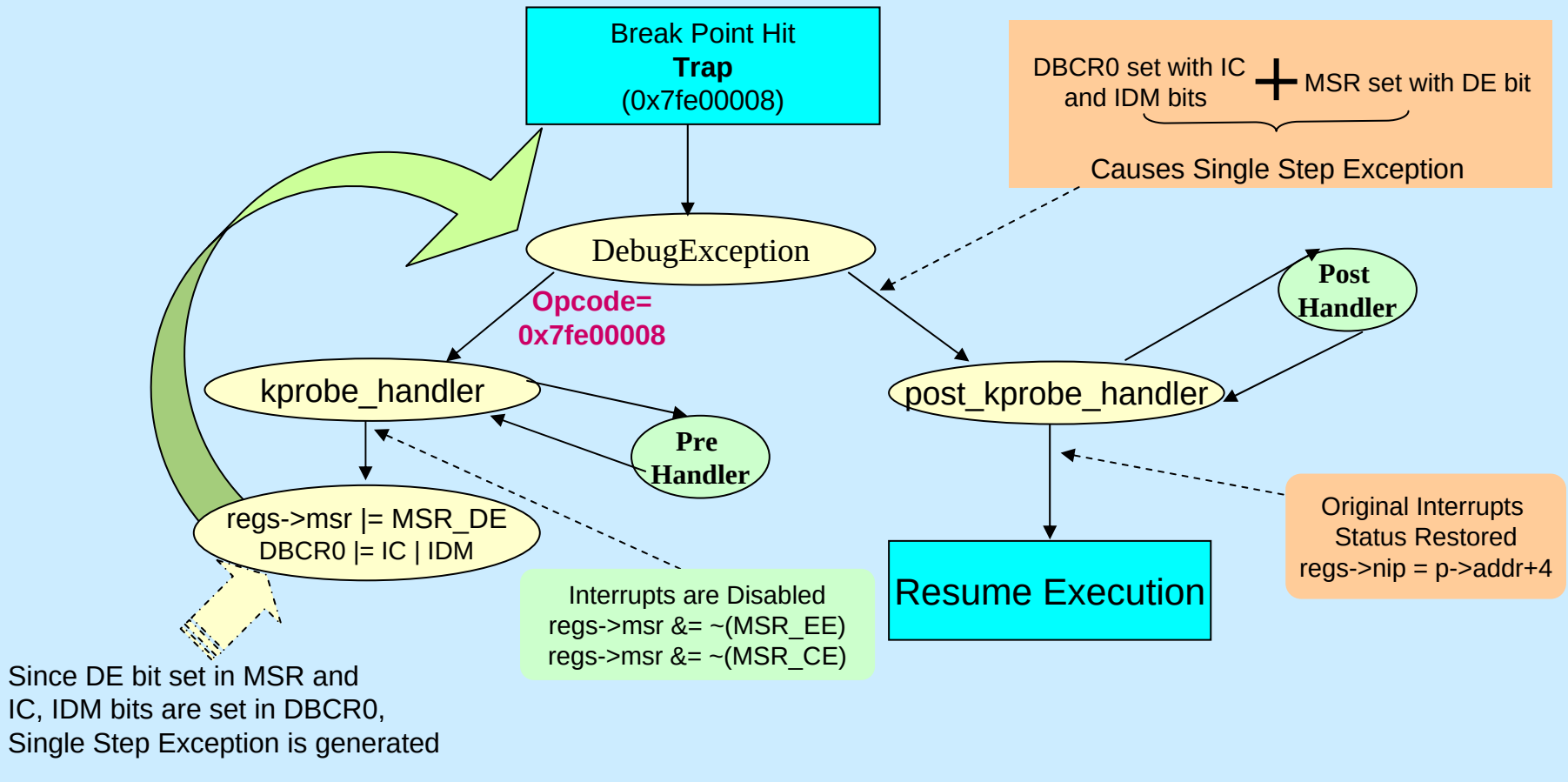


PPC32 Kprobes Design (Contd)



PPC32 Kprobes Design (Contd)

Working Kprobes PPC32 BookE Execution Flow



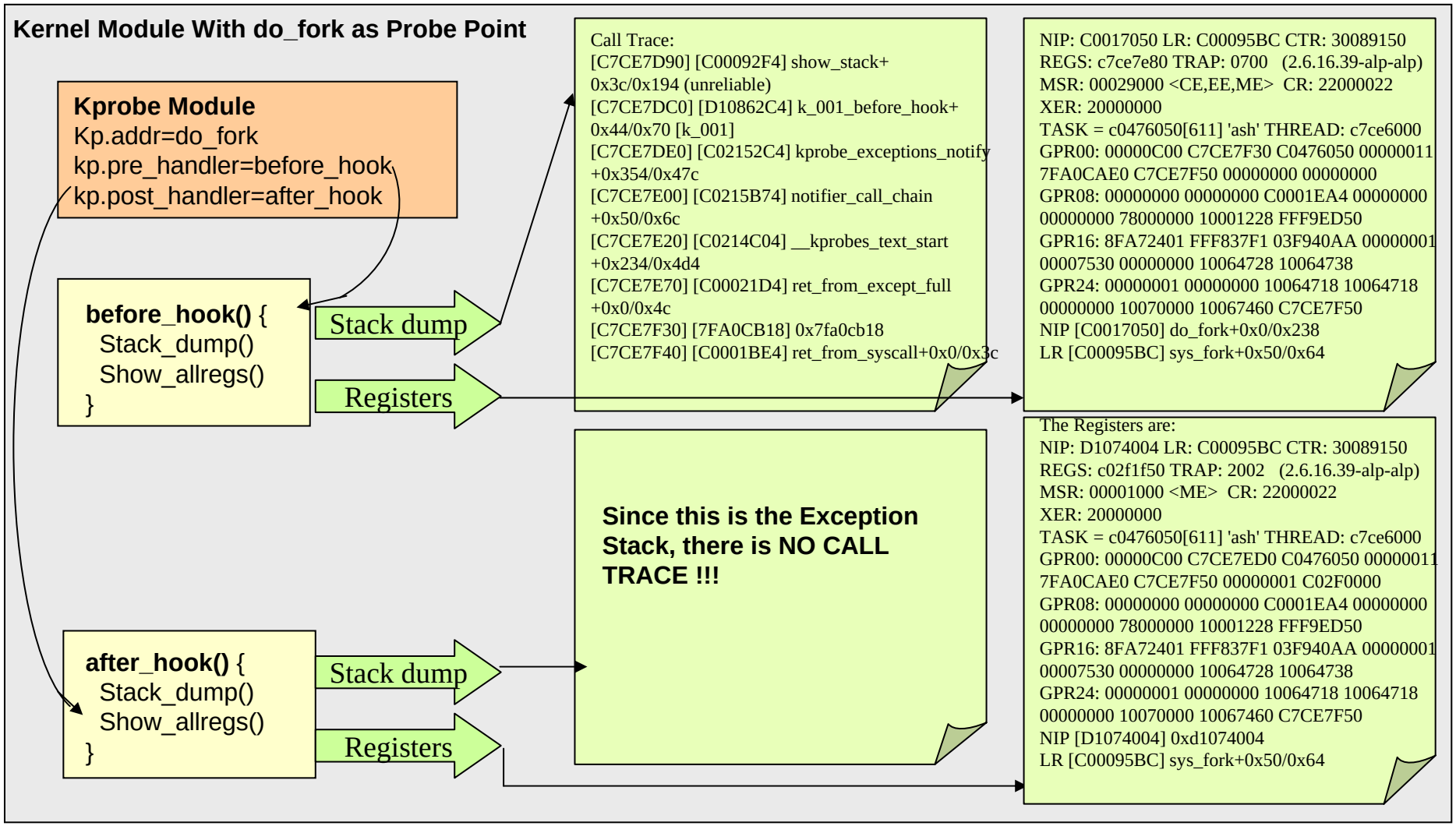
PPC32 Kprobes Design (Contd)

- Kprobes Control Flow Details
  - When a particular instruction “insn1” is probed, the original instruction will be copied on the executable page and replaces the probed instruction by ‘trap’ i.e 0x7fe00008
  - A breakpoint exception occurs, immediately and unconditionally transferring control to the exception handler
  - When ‘tw’ is hit, kprobes exception notifier will be called by the exception handler program_check_exception () with the die value DIE_BPT. This will in turn call kprobe_handler() which performs executing pre_handler() associated with kprobes

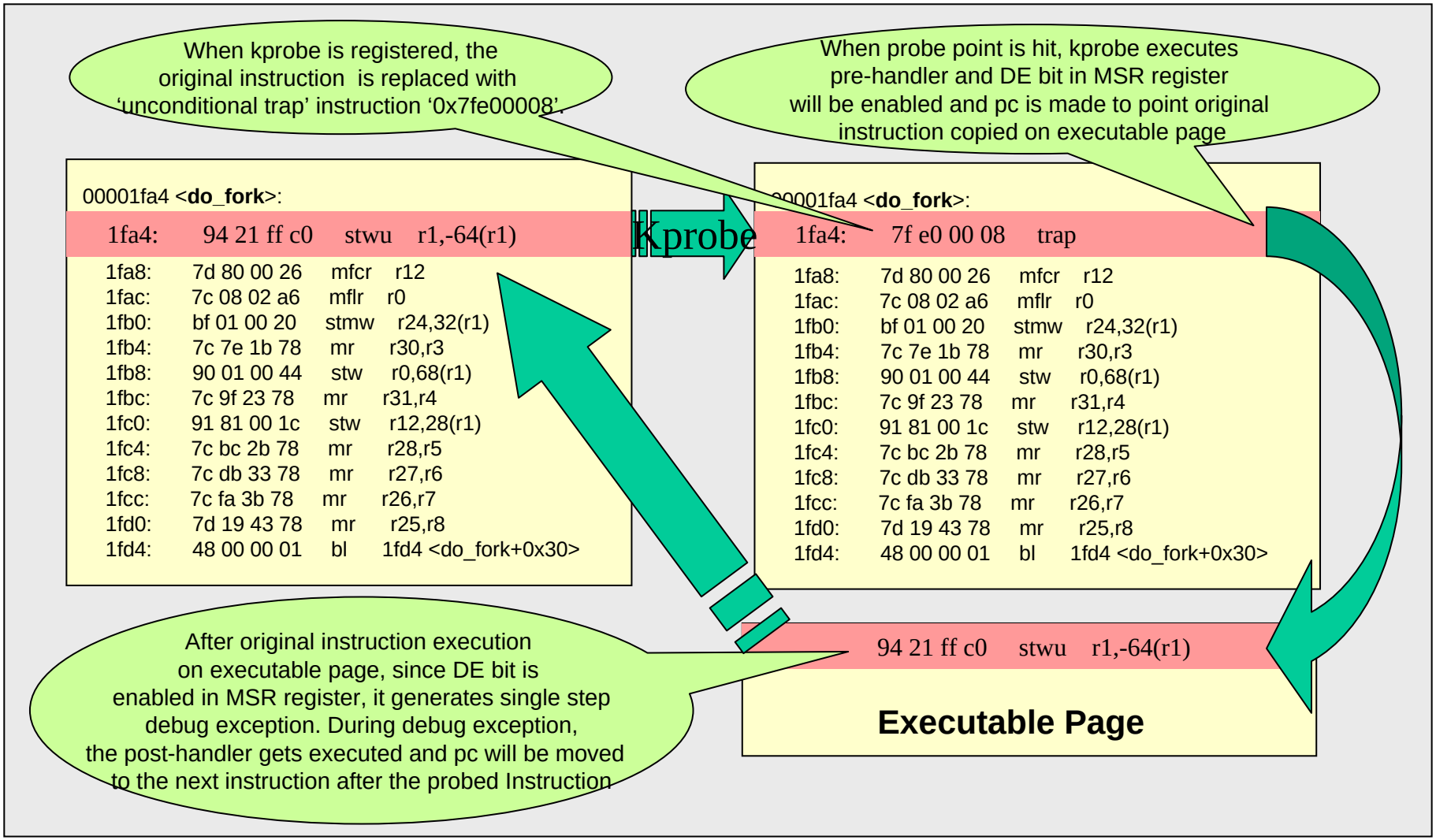
PPC32 Kprobes Design (Contd)

- Kprobes Control Flow Details (Contd)
  - kprobe_handler() takes care of moving program counter to the copied instruction by calling prepare_single_step(). DE bit of MSR and IC, IDM bits of DBCR0 will also be set as part of prepare_single_step() and this will generate debug exception.
  - Since MSR[DE] bit is set, single step operation will be performed on the copied instruction in the executable page and kprobes exception notifier will be called by the exception handler DebugException() with the die value DIE_SSTEP. This will in turn call user defined post_handler()
  - After the instruction debugging, the execution resumes to the next instruction “Insn2”

PPC32 Sample Example Output



PPC32 Example Disassembly



PPC32 Kprobes Limitations

- If post handler tries to dump stack contents, then call trace cannot be displayed because during post handler execution, the stack corresponds to Exception Stack.
- Only minimal testing is performed and needs to be tested using system tap test suites
- Not tested under SMP configuration

Overhead Measurement Details

Kprobes Overhead Measurement Data for MIPS, ARM and PPC32 Arch
Overhead Measurement Test Sample: http://sourceware.org/ml/systemtap/2006-q1/msg00318.html

Overhead	MIPS(usec/call)	ARM(usec/call)	PPC32(usec/call)
Time taken to execute gettimeofday	1.16	5.22	0.99
Time taken to execute gettimeofday when a kprobe is inserted at do_gettimeofday with only Pre-handler	2.98	11.85	3.26
pre_handler Overhead	1.81	6.62	2.27
Time taken to execute gettimeofday when a kprobe is inserted at do_gettimeofday with only Post-handler	2.99	12.04	3.27
post_handler Overhead	1.82	6.81	2.28
Time taken to execute gettimeofday when a kprobe is inserted at do_gettimeofday with both Pre and post handler	3.01	12.10	3.30
(pre+post)_handler Overhead	1.84	6.87	2.31

Source Patches

- MIPS patches can be extracted from below link under patches for 2.6.16
  - http://tree.celinuxforum.org/CelfPubWiki/PatchArchive
- ARM patches can be extracted from below link under patches for 2.6.16
  - http://tree.celinuxforum.org/CelfPubWiki/PatchArchive
- PPC32 patches will be uploaded very soon in CELF Patch Archive page

References

- http://www-128.ibm.com/developerworks/library/l-kprobes.html?ca=dgr-lnxw42Kprobe
- http://www-users.cs.umn.edu/~boutcher/kprobes/
- http://tree.celinuxforum.org/pipermail/celinux-dev/2006-October/001335.html
- http://tree.celinuxforum.org/pipermail/celinux-dev/2007-January/001395.html