What is the difference between Trap and Interrupt?

Asked 9 years, 9 months ago Active 1 year, 8 months ago Viewed 166k times

153

90

What is the difference between Trap and Interrupt?

If the terminology is different for different systems, then what do they mean on x86?

x86 operating-system kernel interrupt cpu-architecture

edited Jun 1 '16 at 16:43
Peter Cordes
186k 29 303 464

asked Jun 30 '10 at 12:23
David
2,465 5 19 28

9 Answers

Active Oldest Votes

195

A **trap** is an exception in a user process. It's caused by division by zero or invalid memory access. It's also the usual way to invoke a kernel routine (a system call) because those run with a higher priority than user code. Handling is synchronous (so the user code is suspended and continues afterwards). In a sense they are "active" - most of the time, the code expects the trap to happen and relies on this fact.

An **interrupt** is something generated by the hardware (devices like the hard disk, graphics card

You can also see a trap as a kind of CPU-internal interrupt since the handler for trap handler looks like an interrupt handler (registers and stack pointers are saved, there is a context switch, execution can resume in some cases where it left off).

edited Aug 27 '14 at 9:52

answered Jun 30 '10 at 12:28

Aaron Digulla
280k 91 513 745

4 It's interesting that lxr.free-electrons.com/source/arch/x86/kernel/… divide by zero is initialized as a hardware interrupt, why is that so? – Alex Kreimer May 22 '12 at 8:09

7 Because it's really an interrupt which the CPU sends when the ALU finds this problem. Just like a segmentation fault. Not all math errors cause interrupts (overflow doesn't), though. – Aaron Digulla May 22 '12 at 8:13

3 That makes sense. But then, what's a bit confusing is that why in earlier Linux kernels it was initialized as a software trap: set_trap_gate(0,&divide_error); – Alex Kreimer May 22 '12 at 9:56

9 What do you mean, "a **bit** confusing"? It's **very** confusing :-) The problem here is that divide by zero is a hardware interrupt (IRQ/vector 0) but the kernel developers have several choices how to handle it. So from a user process, it's a trap but from the CPU side, it's an interrupt. Who is right? None? Both? – Aaron Digulla May 22 '12 at 11:06

2 Of course, this is only true for x86 CPUs. Other CPUs work differently. – Aaron Digulla May 22 '12 at 11:07

103

Traps and **interrupts** are closely related. Traps are a type of **exception**, and exceptions are similar to interrupts.

Intel x86 defines two overlapping categories, vectored events (**interrupts** vs **exceptions**), and exception classes (**faults** vs **traps** vs **aborts**).

All of the quotes in this post are from the April 2016 version of the Intel Software Developer Manual. For the (definitive and complex) x86 perspective, I recommend reading the SDM's chapter on Interrupt and Exception handling.

Vectored Events

Vectored Events (**interrupts** and **exceptions**) cause the processor to jump into an interrupt handler after saving much of the processor's state (enough such that execution can continue from that point later).

Exceptions and interrupts have an ID, called a vector, that determines which interrupt handler the processor jumps to. Interrupt handlers are described within the Interrupt Descriptor Table.

Interrupts occur at random times during the execution of a program, in response to signals from hardware. System hardware uses interrupts to handle events external to the processor, such as requests to service peripheral devices. Software can also generate interrupts by executing the INT n instruction.

Exceptions

Exceptions occur when the processor detects an error condition while executing an instruction, such as division by zero. The processor detects a variety of error conditions including protection violations, page faults, and internal machine faults.

Exception Classifications

Exceptions are classified as **faults**, **traps**, or **aborts** depending on the way they are reported and whether the instruction that caused the exception can be restarted without loss of program or task continuity.

Summary: **traps** increment the instruction pointer, **faults** do not, and **aborts** 'explode'.

Trap

A **trap** is an exception that is reported immediately following the execution of the trapping instruction. Traps allow execution of a program or task to be continued without loss of program continuity. The return address for the trap handler points to the instruction to be executed after the trapping instruction.

Fault

A **fault** is an exception that can generally be corrected and that, once corrected, allows the program to be restarted with no loss of continuity. When a fault is reported, the processor restores the machine state to the state prior to the beginning of execution of the faulting instruction. The return address (saved contents of the CS and EIP registers) for the fault handler points to the faulting instruction, rather than to the instruction following the faulting instruction.

Example: A page fault is often recoverable. A piece of an application's address space may have been swapped out to disk from ram. The application will trigger a page fault when it tries to access memory that was swapped out. The kernel can pull that memory from disk to ram, and hand control back to the application. The application will continue where it left off (at the faulting instruction that was accessing swapped out memory), but this time the memory access should succeed without faulting.

An **abort** is an exception that does not always report the precise location of the instruction causing the exception and does not allow a restart of the program or task that caused the exception. Aborts are used to report severe errors, such as hardware errors and inconsistent or illegal values in system tables.

Edge Cases

Software invoked interrupts (triggered by the INT instruction) behave in a trap-like manner. The instruction completes before the processor saves its state and jumps to the interrupt handler.

edited Jun 1 '16 at 3:42

answered Jun 1 '16 at 3:05

ruthafjord
1,301 1 8 12

5 This is such a good answer I was positive the "Late Answer from a New User" review queue was giving me a test to make sure I was paying attention. – Noumenon Jun 1 '16 at 3:33

Thanks! That means a lot to me :) – ruthafjord Jun 1 '16 at 3:46

This is a great answer for x86. The question originally was pretty general and didn't mention x86. I made an edit to try to strike a balance. Perhaps you could add another paragraph at the top of this answer to address the question of terminology outside of the x86 world? And / or leave another edit on the question if you think I made it worse :P I only really know x86 myself, but hopefully it's true that other systems use very similar terminology, and you can just say that :) – Peter Cordes Jun 1 '16 at 16:44

1 Skimmed the PPC aarchitecture book, and it looks like their definitions are largely overlapping. They have new names for edge cases, and treat exceptions like a subtype of interrupts, rather than part of a distinct category. – ruthafjord Jun 1 '16 at 18:29

1 I think this answer describes it best. It discusses the blurry line that can exist between the two. And mentions that page faults result in the CPU reattempting a instruction a trap skips over the instruction and moves on. – in70x Aug 3 '16 at 23:05

9

Generally speaking, terms like exceptions, faults, aborts, **Traps**, and **Interrupts** all mean the same thing and are called "Interrupts".

Coming to the difference between Trap and Interrupt:

Trap: Is a programmer initiated and expected transfer of control to a special handler routine. (For ex: 80x86 **INT** instruction is a good example)

Where as

Interrupt(Hardware): Is a program control interruption based on an external hardware event external to the CPU (For ex: Pressing a key on the keyboard or a time out on a timer chip)

Cheshar
191 1 2 12

Good definitions. Source? – alexlomba87 Feb 3 at 22:39

6

A *trap* is a special kind of *interrupt* which is commonly referred to as a *software interrupt*. An *interrupt* is a more general term which covers both *hardware interrupts* (interrupts from hardware devices) and *software interrupts* (interrupts from software, such as *traps*).

answered Jun 30 '10 at 13:17
Paul R
186k 26 324 490

4 It confuses matters even more that some authors (Tanenbaum) refer to "hardware traps." If we can have hardware traps and software interrupts, clearly the definitions are fairly muddy and can go either way, always requiring the word hardware or software. – The111 Apr 3 '12 at 2:33

3

A trap is called by code like programs and used e. g. to call OS routines (i. e. normally synchronous). An interrupt is called by events (many times hardware, like the network card having received data, or the CPU timer), and - as the name suggests - interrupts the normal control flow, as the CPU has to switch to the driver routine to handle the event.

answered Jun 30 '10 at 12:28
Frank
2,390 11 13

2

An interrupt is a hardware-generated change-of-flow within the system. An interrupt handler is summoned to deal with the cause of the interrupt; control is then returned to the interrupted context and instruction. A trap is a software-generated interrupt. An interrupt can be used to signal the completion of an I/O to obviate the need for device polling. A trap can be used to call operating system routines or to catch arithmetic errors.

answered Jul 13 '14 at 16:40
Dinesh Reddy
21 3

2

I think Traps are caused by the execution of current instruction and thus they are called as synchronous events. where as interrupts are caused by an independent instruction that is running in the processor which are related to external events and thus are known as asynchronous ones.

answered Aug 28 '14 at 0:28
chetan pawar

2

Interrupts are hardware interrupts, while traps are software-invoked interrupts. Occurrences of hardware interrupts usually disable other hardware interrupts, but this is not true for traps. If you need to disallow hardware interrupts until a trap is served, you need to explicitly clear the interrupt flag. And usually the interrupt flag on the computer affects (hardware) interrupts as opposed to traps. This means that clearing this flag will not prevent traps. Unlike traps, interrupts should preserve the previous state of the CPU.

answered Feb 17 '15 at 16:09

Anamik Sarvaiya
21 2

1

A trap is a software interrupt.If you write a program in which you declare a variable having divide by zero value then it is treated as a trap.Whenever you run this program it will throw same error at the same time.System call is a special version of trap in which a program asks os for its required service. In case of interrupt(a general word for hardware interrupts)like an i/o error,the cpu is interrupted at random time and off course it is not the fault of our programmers.It is the hardware that brings them up.

edited Mar 18 '15 at 18:11

answered Jan 24 '15 at 8:55

Sanketssj5
585 4 16

1 Can you explain how system call is a trap ? – Radha Gogia Dec 28 '15 at 6:28
