ISSN 2349-6193
Impact Factor: 2.805

IJESMR

International Journal OF Engineering Sciences & Management Research

STUDY ABOUT INTERRUPT, TRAP AND FATAL ERROR

Nikita Sharma*
*SOC &E, IPS Academy, Rajendra Nagar, Indore, M.P., India

Keywords: Interrupt Service Routine (ISR), Hardware Interrupts (Asynchronous Interrupt) , Software Interrupts (Synchronous Interrupt),Interrupt Request (IRQ), Disk Controller, Interrupt Vector, Interrupt Vector Table (*Dispatch Table*), Trap (Exception or Fault), Subroutine Calls , Computer Multitasking, Real-Time Computing, Interrupt-Driven, Fatal Error, Fatal Exception Error, Exception Handling.

ABSTRACT
The main aim of this paper is to make a clear vision between the interrupt, trap and fatal error. Or we can say that the paper will be able to clarify the behavior and working of interrupt, trap and fatal error. This paper have justified definition and data about all these three terms ,and the paper itself is capable to differentiate and clear every confusion between these three terms called interrupt ,trap and fatal error.

INTRODUCTION

Interrupt
An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing. The processor responds by suspending its current activities, saving its state, and executing a function called an interrupt handler (or an interrupt service routine, ISR) to deal with the event. This interruption is temporary, and, after the interrupt handler finishes, the processor resumes normal activities. [1]

In other words When a Process is executed by the CPU and when a user Request for another Process then this will create disturbance for the Running Process. This is also called as the **Interrupt**. [2]

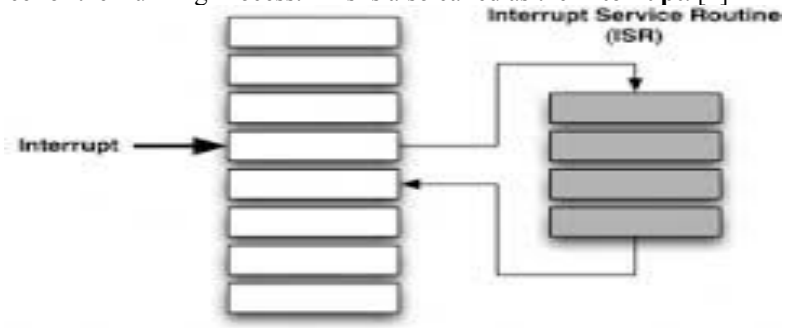

Figure 1: Interrupt and interrupt Service Routine.

Interrupt Vector and Interrupt Vector Table
An interrupt vector is the memory location of an interrupt handler, which prioritize interrupt and saves them in a queue if more than one interrupt is waiting to be handled. [3] A table of *interrupt vectors* is called interrupt vector Table, it is also known as *dispatch table. [4]*

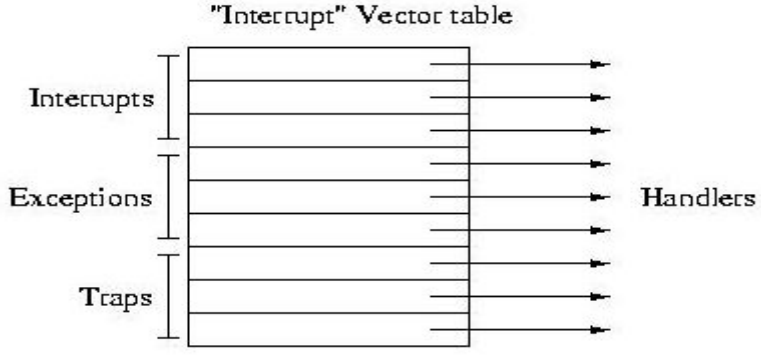

Figure 2: Interrupt Vector Table.

There are two types of interrupts.

ISSN 2349-6193
Impact Factor: 2.805

IJESMR
International Journal OF Engineering Sciences & Management Research

1. Hardware interrupt

Hardware interrupts are used by devices to communicate that they require attention from the operating system. Internally, hardware interrupts are implemented using electronic alerting signals that are sent to the processor from an external device, which is either a part of the computer itself, such as a disk controller, or an external peripheral. For example, pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position. Unlike the software type (described below), hardware interrupts are asynchronous and can occur in the middle of instruction execution, requiring additional care in programming. The act of initiating a hardware interrupt is referred to as an interrupt request (IRQ). [5]

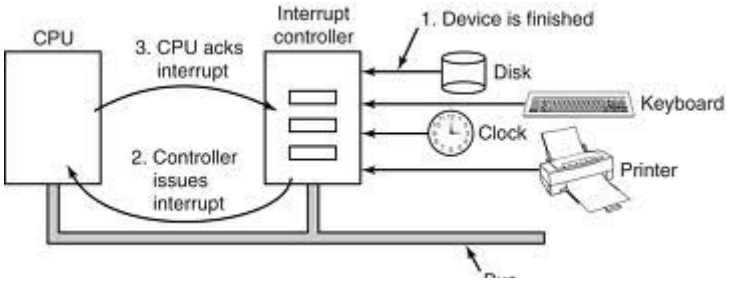

Figure 3: Interrupt Handling

2. Software interrupt

Software interrupt (synchronous interrupt)is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. The former is often called a **trap** or *exception* and is used for errors or events occurring during program execution that is exceptional enough that they cannot be handled within the program itself. For example, if the processor's arithmetic logic unit is commanded to divide a number by zero, this impossible demand will cause a divide-by-zero exception, perhaps causing the computer to abandon the calculation or display an error message. Software interrupt instructions function similarly to subroutine calls and are used for a variety of purposes, such as to request services from low-level system software such as device drivers. For example, computers often use software interrupt instructions to communicate with the disk controller to request data be read or written to the disk. [5]

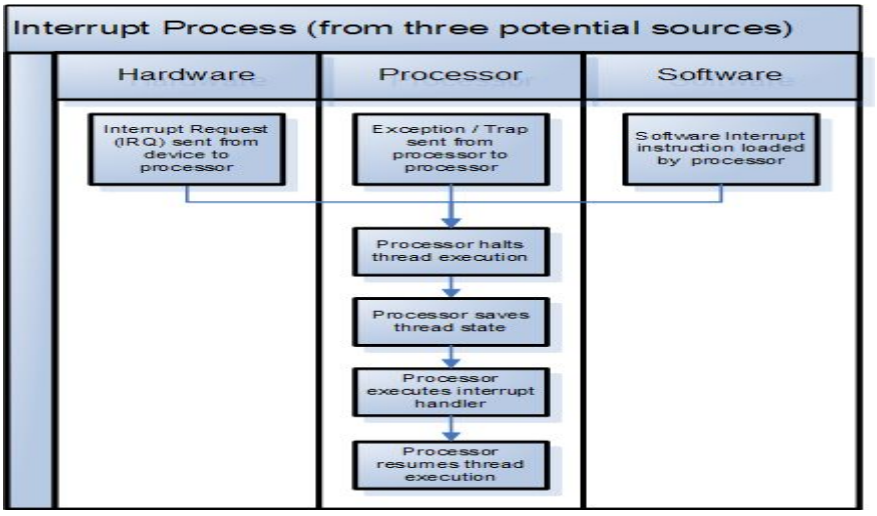

Figure 4: Interrupt Processing

Each interrupt has its own interrupt handler. The number of hardware interrupts is limited by the number of interrupt request (IRQ) lines to the processor, but there may be hundreds of different software interrupts. Interrupts are a commonly used technique for computer multitasking, especially in real-time computing. Such a system is said to be interrupt-driven. [6]

ISSN 2349-6193
Impact Factor: 2.805

IJESMR
International Journal OF Engineering Sciences & Management Research

A trap is a software-generated interrupt; 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 can be used to call operating system routines or to catch arithmetic errors. [7]

FATAL ERROR

An error that causes a program to abort. Sometimes a fatal error returns you to the operating system. When a fatal error occurs, you may lose whatever data the program was currently processing. [8]

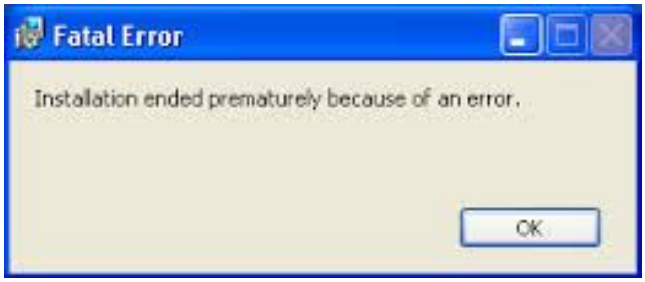

Figure 5: Fatal Error regarding installation.

For example When a program like Microsoft Word or Excel "crashes," it means that something has gone seriously wrong during the program's execution. The operating system often recognizes that there is a serious problem and kills off the offending application in a clean way. When it does this, the operating system will say something cryptic like "fatal exception error" (and often display a large collection of hexadecimal numbers that are totally useless to you, the user, but might be of some use to the original programmer). The other way for a program to crash is for it to take the operating system down with it, meaning that you have to reboot. [9]

Figure 6: Fatal error regarding uncaught memory exception.

An application program like Microsoft Word is made up of many layers and components. There is the core operating system, an operating system services layer, perhaps an encapsulation layer on top of the system services, hundreds of software libraries, internal function/class libraries and DLLs, and finally the main application layer. Most modern operating systems and languages (like C++, Java, etc.) support programming concepts known as **exceptions** and **exception handling**. Exceptions allow different layers to communicate problems to each other. For example, say that a program needs some memory, so it asks the operating system to reserve a block of memory. If the operating system is unable to honor the memory request (because the requested block is too big, or the system is low on memory, or whatever), it will "throw a memory exception" up to the layer that made the request. Various layers may continue to throw the exception upward. Somewhere along the line, one of the layers needs to "catch the exception" and deal with the problem. The program needs to say, the system is out of memory. I need to tell the user about this with a nice dialog box. If the program fails to catch the exception (because for some reason the programmer never wrote the code to handle that particular exception), the exception makes it all the way to the top of all the layers, and the operating system recognizes it as an "unhandled exception." The operating system then shuts down the program. Well-designed software handles all exceptions. [9]

CONCLUSION

This paper has included basic definition, their related terms, and proper example with appropriate diagrams of interrupt, trap and fatal error. By this paper we can easily differentiate among interrupt, trap and fatal error.

ISSN 2349-6193
Impact Factor: 2.805

IJESMR

International Journal OF Engineering Sciences & Management Research

REFERENCES

1. *Jonathan Corbet; Alessandro Rubini; Greg Kroah-Hartman (2005). "Linux Device Drivers, Third Edition, Chapter 10. Interrupt Handling" (PDF). O'Reilly Media. p. 269. Retrieved December 25, 2014. Then it's just a matter of cleaning up, running software interrupts, and getting back to regular work. The "regular work" may well have changed as a result of an interrupt (the handler could wakeup a process, for example), so the last thing that happens on return from an Interrupt is a possible rescheduling of the processor.*
2. *http://ecomputernotes.com/fundamental/input-output-and-memory/what-is-interrupt-types-of-interrupts.*
3. *http://whatis.techtarget.com/definition/interrupt-vector*
4. *http://www.webopedia.com/TERM/I/interrupt vector table.html*
5. *"Hardware interrupts". Retrieved 2014-02-09.*
6. *"Basics of Interrupts". Retrieved 2010-11-11.*
7. *http://www.geekinterview.com/question details/62604.*
8. *http://www.webopedia.com/TERM/F/fatal error.html.*
9. *http://computer.howstuffworks.com/question288.htm*