Interrupts and interrupt applications

8086 Interrupts And Interrupt Applications

Introduction:

Microprocessors are computers built on single IC. There can be more IC’s also used for this. Most microprocessors allow normal program execution to be interrupted by some external signal or by a special instruction in the program. In response to an interrupt the microprocessor stops executing its current program and calls a procedure which sevices the interrupt. An IRET instruction at the end

of the interrupt service procedure returns execution to the interrupted program.

8086 Interrupts And Interrupt Responses:

An 8086 interrupt can come from any one of three sources. One source is an external signal applied to the non-maskable interrupt (NMI) input pin or to the interrupt input pin. An interrupt caused by a signal applied to one of these inputs is referred to as a hardware interrupt. A second source of an interrupt is execution of the interrupt instruction. This is referred to as a software interrupt. The third source of an interrupt is some error condition produced in the 8086 by the execution of an instruction. An example of this is the divide by zero interrupt. If you attempt to divide an operand by zero, the 8086 will automatically interrupt the currently executing program. At the end of each instruction cycle, the 8086 checks to see if any interrupts have been requested. If an interrupt has been requested, the 8086 responds to the interrupt by stepping through the following series of major actions:

1) It decrements the stack pointer by 2 and pushes the flag register on the stack.

2) It disables the 8086 INTR interrupt input by clearing the interrupt flag in the flag register.

3) It resets the trap flag in the flag register.

4) It decrements the stack pointer by 2 and pushes the current code segment register contents on the stack.

5) It decrements the stack pointer again by 2 and pushes the current instruction pointer contents on the stack.

8086 Interrupt Types:

The preceding sections used the type 0 interrupts an example of how the 8086 interrupts function. It has hardware caused NMI interrupt, the software interrupts produced by the INT instruction, and the hardware interrupt produced by applying a signal to the INTR input pin.

DIVIDE-BY-ZERO INTERRUPT-TYPE 0:

The 8086 will automatically do a type 0 interrupt if the result of a DIV operation or an IDIV operation is too large to fit in the destination register. For a type 0 interrupt, the 8086 pushes the flag register on the stack, resets IF and TF and pushes the return addresses on the stack.

SINGLE STEP INTERRUPT-TYPE 1:

The use of single step feature found in some monitor programs and debugger programs. When you tell a system to single step, it will execute one instruction and stop. If they are correct we can tell a system to single step, it will execute one instruction and stop. We can then examine the contents of registers and memory locations. In other words, when in single step mode a system will stop after it executes each instruction and wait for further direction from you. The 8086 trap flag and type 1 interrupt response make it quite easy to implement a single step feature direction.

NONMASKABLE INTERRUPT-TYPE 2:

The 8086 will automatically do a type 2 interrupt response when it receives a low to high transition on its NMI pin. When it does a type 2 interrupt, the 8086 will push the flags on the stack, reset TF and IF, and push the CS value and the IP value for the next instruction on the stack. It will then get the CS value for the start of the type 2 interrupt service procedure from address 0000AH and the IP value for the start of the procedure from address 00008H.

BREAKPOINT INTERRUPT-TYPE 3:

The type 3 interrupt is produced by execution of the INT3 instruction. The main use of the type 3 interrupt is to implement a breakpoint function in a system. When we insert a breakpoint, the system executes the instructions up to the breakpoint and then goes to the breakpoint procedure. Unlike

the single step which stops execution after each instruction, the breakpoint feature executes all the instructions up to the inserted breakpoint and then stops execution.

OVERFLOW INTERRUPT-TYPE4:

The 8086 overflow flag will be set if the signed result of an arithmetic operation on two signed numbers is too large to be represented in the destination register or memory location. For example, if you add the 8 bit signed number 01101100 and the 8 bit signed number 010111101, the result will be 10111101. This would be the correct result if we were adding unsigned binary numbers, but it is not the correct signed result.

SOFTWARE INTERRUPTS-TYPE O THROUGH 255:

The 8086 INT instruction can be used to cause the 8086 to do any one of the 256 possible interrupt types. The desired interrupt type is specified as part of the instruction. The instruction INT32, for example will cause the 8086 to do a type 32 interrupt response. The 8086 will push the flag register on the stack, reset TF and IF, and push the CS and IP values of the next instruction on the stack.

INTR INTERRUPTS-TYPES 0 THROUGH 255:

The 8086 INTR input allows some external signal to interrupt execution of a program. Unlike the NMI input, however, INTR can be masked so that it cannot cause an interrupt. If the interrupt flag is cleared, then the INTR input is disabled. IF can be cleared at any time with CLEAR instruction.

PRIORITY OF 8086 INTERRUPTS:

If two or more interrupts occur at the same time then the highest priority interrupt will be serviced first, and then the next highest priority interrupt will be serviced. As a example suppose that the INTR input is enabled, the 8086 receives an INTR signal during the execution of a divide instruction, and the divide operation produces a divide by zero interrupt. Since the internal interrupts-such as divide error, INT, and INTO have higher priority than INTR the 8086 will do a divide error interrupt response first.

Hardware Interrupt Applications:

Simple Interrupt Data Input:

One of the most common uses of interrupts is to relieve a CPU of the burden of polling. To refresh your memory polling works as follows. The strobe or data ready signal from some external device is connected to an input port line on the microcomputer. The microcomputer uses a program loop to read and test this port line over and over until the data ready signal is found to be asserted. The microcomputer then exits the polling loop and reads in the data from the external device. The disadvantage of polled input or output is that while the microcomputer is polling the strobe or data ready signal, it cannot easily be doing other tasks. I n this case the data ready or strobe signal is connected to an interrupt input on the microcomputer. The microcomputer then goes about doing its other tasks until it is interrupted by a data ready signal from the external device. An interrupt service procedure can read in or send out the desired data in a few microseconds and return execution to the interrupted program. The input and output operation then uses only a small percentage of the microprocessors time.

Counting Applications:

As a simple example of the use of an interrupt input for counting , suppose that we are using an 8086 to control a printed circuit board making machine in our computerized electronics factory. Further suppose that we want to detect each finished board as it comes out of the machine and to keep a count with the number of boards fed in. This way we can determine if any boards were lost in the machine. To do this count on an interrupt basis, all we have to do is to detect when a board passes out of the machine and send an interrupt signal to an interrupt input on the 8086. The interrupt service procedure for that input can simply increment the board count stored in a named memory location. To detect a board coming out of the machine, we use an infrared LED, a photoresistor and two conditioning gates. The LED is positioned over the track where the boards come out, and the photoresistor is positioned below the track. When no board is between the LED and the photoresistor, the light from the LED will strike the photoresistor and turn it on. The collector of the photoresistor will then be low, as will the NMI input on the 8086. When a board passes between the LED and photoresistor, the light will not reach the photoresistor and turn it on. The collector of the photoresistor will then be low, as will the NMI input on the 8086.

Timing Applications:

In this it is shown that how delay loop could be used to set the time between microcomputer operations. In the example there, we used a delay loop to take in data samples at 1 ms intervals. The obvious disadvantage of a delay loop is that while the microcomputer is stuck in the delay loop, it cannot easily be doing other useful work. In many cases a delay loop would be a waste of the microcomputer’s valuable time, so we use an interrupt approach. Suppose for example, that in our 8086 controlled printed circuit board making machine we need to check the ph of a solution approximately every 4 min. If we used a delay loop to count off the 4 min, either the 8086 wouldn't be able to do much else or what points in the program to go check the ph.

8254 Software-Programmable Timer/Counter:

Because of many tasks that they can be used for in microcomputer systems, programmable timer/counters are very important for you to learn about. As you read through following sections, pay particular attention to the applications of this device in systems and the general procedures for initializing a programmable device such as 8254.

Basic 8253 And 8254 Operation:

The intel 8253 and 8254 each contain three 16 bit counters which can be programmed to operate in several different modes. The major differences are as follows:

1) The maximum input clock frequency for the 8253 is 2. 6 MHz, the maximum clock frequency for the 8254 is 8MHz.

2) The 8254 has a read back feature which allows you to latch the count in all the counters and the status of the counter at any point. The 8253 does not have this read back feature. The big advantage of these counters, however, is that you can load a count in them, start them and stop them with instructions in your program. Such a device is said to be software programmable.

8259a Priority Interrupt Controller:

In a small system, for example, we might read ASCII characters in from a keyboard on an interrupt basis; count interrupts from timer to produce a real time clock of second, minutes and hours and detect several emergency or job done conditions on an interrupt basis. Each of these interrupt applications requires a separate interrupt input. If we are working with an 8086 , we have problem here because the 8086 has only two interrupt inputs, NMI and INTR. If we save NMI for a power failure interrupt, this leaves only one input for all the other applications. For applications where we have interrupts from multiple sources, we use an external device called a priority interrupt controller.

Software Interrupt Applications:

The software interrupt instruction INT N can be used to test any type of interrupt procedure. For example to test a type 64 interrupt procedure without the need for external hardware, we can execute the instruction INT 64.

Another important use of software interrupts is to call Basic Input Output System, or BIOS, procedures in an IBM PC-type computer. These procedures in the system ROMS perform specific input or output functions, such as reading a character from the keyboard, writing some characters to the CRT, or reading some information from a disk. To call one of these procedures, you load any required parameters in some specified registers and execute an INT N instruction. N in this case is the interrupt type which vectors to the desired procedure. Suppose that, as part of an assembly language program that you are writing to run on an IBM PC type computer, you want to send some characters to the printer. The header for the INT 17H procedure from the IBM PC BIOS listing. The DX, AH, and AL registers are used to pass the required parameters to the procedure. The procedure is used for two different operations: initializing the printer port and sending a character to the printer. The operation performed by the procedure is determined by the number passed to the procedure in the AH register. AH= 1 means initialize the printer port, AH= 0 means print the characters in AL, and AH= 2 means read the printer status and returned in AH. If an attempt to print a character was not successful for some reason, such as the printer not being turned on, not being selected, or being busy, 01 is returned in AH. The main advantage of calling procedures with software interrupts is that you don't need to worry about the absolute address where the procedure actually resides or about trying to link the procedure into your program. So at last every microcomputer system uses a variety of interrupts and this is all about 8086 interrupts and applications.

Conclusion:

Microprocessors are computers built on single IC. There can be more IC's also used for this. Most microprocessors allow normal program execution to be interrupted by some external signal or by a special instruction in the program. In response to an interrupt the microprocessor stops executing its current program and calls a procedure which sevices the interrupt. An IRET instruction at the end of the interrupt service procedure returns execution to the interrupted program. An 8086 interrupt can come from any one of three sources. One source is an external signal applied to the non-maskable interrupt (NMI) input pin or to the interrupt input pin. An interrupt caused by a signal applied to one of these inputs is referred to as a hardware interrupt. A second source of an interrupt is execution of the interrupt instruction. This is referred to as a software interrupt. The third source of an interrupt is some error condition produced in the 8086 by the execution of an instruction. An example of this is the divide by zero interrupt. If you attempt to divide an operand by zero, the 8086 will automatically interrupt the currently executing program. At the end of each instruction cycle, the 8086 checks to see if any interrupts have been requested. One of the most common uses of interrupts is to relieve a CPU of the burden of polling. To refresh your memory polling works as follows. The strobe or data ready signal from some external device is connected to an input port line on the microcomputer. The microcomputer uses a program loop to read and test this port line over and over until the data ready signal is found to be asserted. The software interrupt instruction INT N can be used to test any type of interrupt procedure. For example to test a type 64 interrupt procedure without the need for external hardware, we can execute the instruction INT 64. So at last we conclude that every microcomputer system uses a variety of interrupts and this is all about 8086 interrupts and applications.

References:

1) DOUGLAS V. HALL, “ microprocessors and interfacing” TaMcGRaw-Hill edition

2) www. wikipedia. com

3) www. google. com