Lecture8: 8086 interrupt

Outline:

1. *Introduction*
2. *Hardware interrupt*
3. *Nonmaskable interrupt*
4. *Maskable interrupt*
5. *Soft interrupt*
6. *reset*

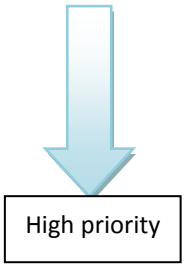
1. Introduction

An interrupt is the method of processing the microprocessor by peripheral device. An interrupt is used to cause a temporary halt in the execution of program. Microprocessor responds to the interrupt with an interrupt service routine, which is short program or subroutine that instructs the microprocessor on how to handle the interrupt.

There are two basic type of interrupt, maskable and non-maskable, non-maskable interrupt requires an immediate response by microprocessor, it usually used for serious circumstances like power failure. A maskable interrupt is an interrupt that the microprocessor can ignore depending upon some predetermined upon some predetermined condition defined by status register.

Interrupt can divide to five groupe:

1. hardware interrupt
2. Non-maskable interrupt
3. Software interrupt
4. Internal interrupt
5. Reset



Hardware,software and internal interrupt are service on priority basis. each interrupt is given a different priority level by assign it a type number. Type 0 identifies the highest-priority and type 255 identifies the lowest- priority interrupt.

The 80x86 chips allow up to 256 vectored interrupts. This means that you can have up to 256 different sources for an interrupt and the 80x86 will directly call the service routine for that interrupt without any software processing. This is in contrast to nonvectored interrupts that transfer control directly to a single interrupt service routine, regardless of the interrupt source.
The 80x86 provides a 256 entry interrupt vector table beginning at address 0:0 in memory. This is a 1K table containing 256 4-byte entries. Each entry in this table contains a segmented address that points at the interrupt service routine in memory. The lowest five types are dedicated to specific interrupts such as the divide by zero interrupt and the non maskable interrupt. The next 27 interrupt types, from 5 to 31 are

reserved by Intel for use in future microprocessors. The upper 224 interrupt types, from32 to 255, are available to use for hardware and software interrupts.

When an interrupt occurs (shown in figure 1), regardless of source, the 80x86 does the following:

1. The CPU pushes the flags register onto the stack.
2. The CPU pushes a far return address (segment:offset) onto the stack, segment value first.
3. The CPU determines the cause of the interrupt (i.e., the interrupt number) and fetches the four byte interrupt vector from address 0:vector*4.
4. The CPU transfers control to the routine specified by the interrupt vector table entry.

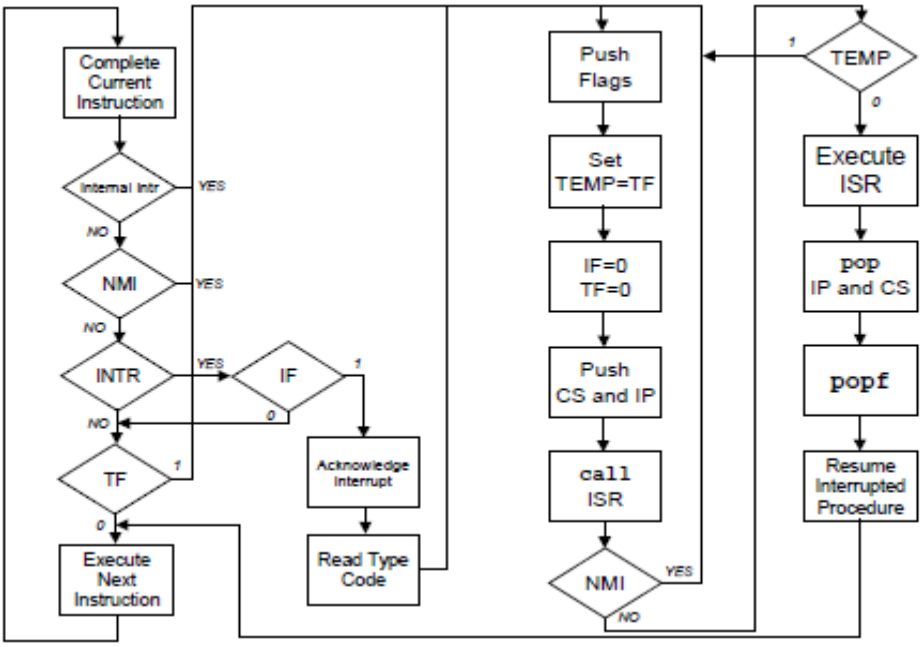


Figure 1:block diagram to interrupt handle

When the interrupt service routine wants to return control, it must execute an IRET (interrupt return) instruction. The interrupt return pops the far return address and the flags off the stack. Note that executing a far return is insufficient since that would leave the flags on the stack.

2. hardware interrupt

The primary sources of interrupts, however, are the PCs timer chip, keyboard, serial ports, parallel ports, disk drives, CMOS real-time clock,

mouse, sound cards, and other peripheral devices. These devices connect to an Intel 8259A programmable interrupt controller (PIC) that prioritizes the interrupts and interfaces with the 80x86 CPU. The 8259A chip adds considerable complexity to the software that processes interrupts.

2.1 programmable interrupt controller

The 8259A programmable interrupt controller chip accepts interrupts from up to eight different devices, which shown in figure (2), If any one of the devices requests service, the 8259 will toggle an interrupt output line (connected to the CPU) and pass a programmable interrupt vector to the CPU. You can *cascade* (show in figure(3))the device to support up to 64 devices by connecting nine 8259s together: eight of the devices with eight inputs each whose outputs become the eight inputs of the ninth device. A typical PC uses two of these devices to provide 15 interrupt inputs (seven on the *master* PIC with the eight input coming from the *slave* PIC to process its eight inputs)7. The sections following this one will describe the devices connected to each of those inputs, for now we will concentrate on what the 8259 does with those inputs. Nevertheless, for the sake of discussion, the following table lists the interrupt sources on the PC:

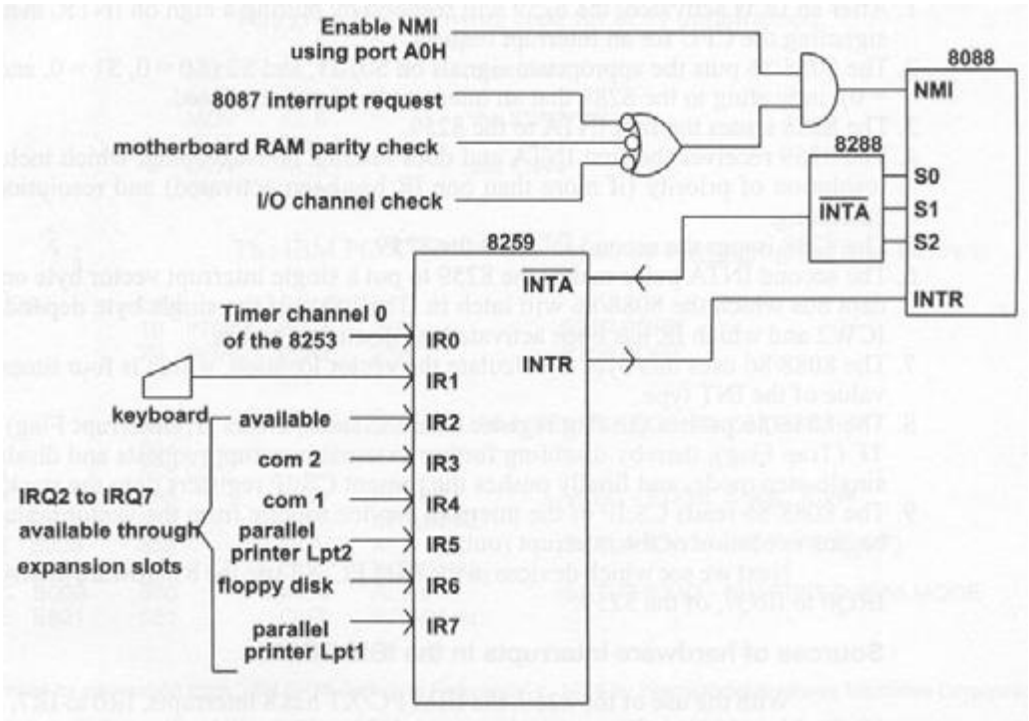


Figure 2: general block diagram 8086 interrupt

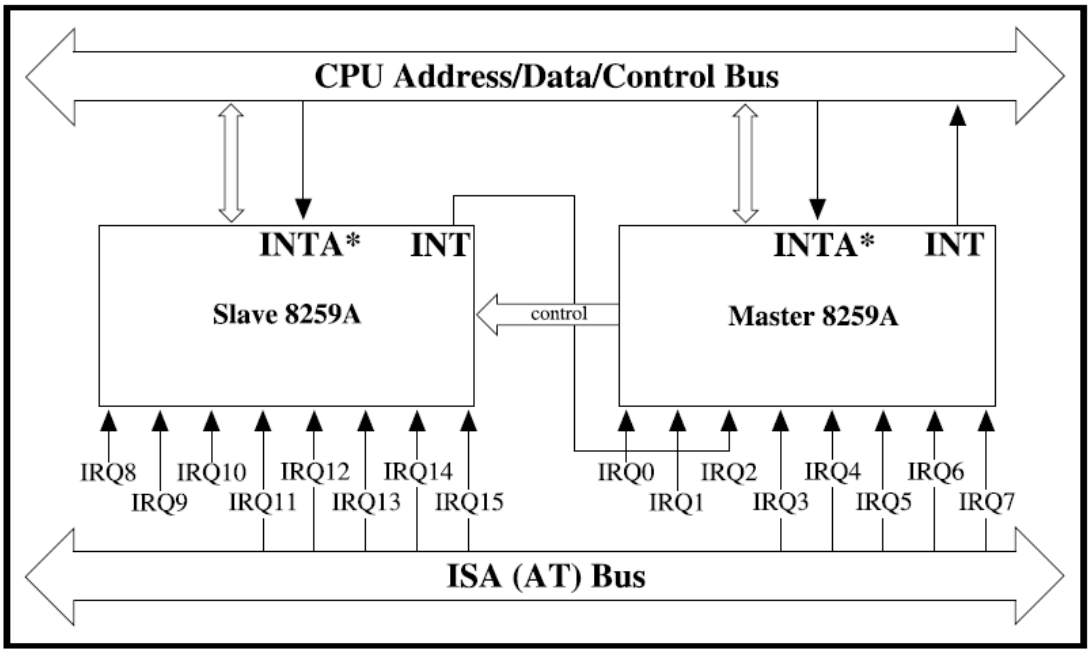


Figure 3:cascade 8259A

3. NON-MASKABLE INTERRUPT (NMI)

The processor provides a single non-maskable interrupt pin (NMI) which has higher priority than the maskable interrupt request pin (INTR). A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW-to-HIGH transition. The activation of this pin causes a type 2 interrupt. NMI is required to have a duration in the HIGH state of greater than two CLK cycles, but is not required to be synchronized to the clock. Any high-going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves of a block-type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another

high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

4. MASKABLE INTERRUPT

Whenever an external signal activates the INTR pin, the microprocessor will be interrupted only if interrupts are enabled using set interrupt Flag instruction. If the interrupts are disabled using clear interrupt Flag instruction, the microprocessor will not get interrupted even if INTR is activated. That is, INTR can be masked. INTR is a non vectored interrupt, which means, the 8086 does not

know where to branch to service the interrupt. The 8086 has to be told by an external device like a Programmable Interrupt controller regarding the branch. Whenever the INTR pin is activated by an I/O port, if Interrupts are enabled and NMI is not active at that time, the microprocessor finishes the current instruction that is being executed and gives out a '0' on INTA pin twice. When INTA pin goes low for the first time, it asks the external device to get ready. In response to the second INTA the microprocessor receives the 8 bit, say N, from a programmable Interrupt controller. The action taken is as follows.

1. Complete the current instruction.
2. Activates INTA output, and receives type Number, say N
3. Flag register value, CS value of the return address & IP value of thereturn address are pushed on to the stack.
4. IP value is loaded from contents of word location N x 4.
5. CS is loaded from contents of the next word location.
6. 2Interrupt Flag and trap Flag are reset to 0.

At the end of the ISS, there will be an IRET instruction. This performs popping off from the stack top to IP, CS and Flag registers. Finally, the register values which are also saved on the stack at the start of ISS, are restored from the stack and a return to the interrupted program takes place using the IRET instruction.

Table 1:type of interrupt

Vector No.	Mnemonic	Description	Source
0	#DE	Divide Error	DIV and IDIV instructions.
1	#DB	Debug	Any code or data reference.
2		NMI Interrupt	Non-maskable external interrupt.
3	#BP	Breakpoint	INT 3 instruction.
4	#OF	Overflow	INTO instruction.
5	#BR	BOUND Range Exceeded	BOUND instruction.
6	#UD	Invalid Opcode (UnDefined Opcode)	UD2 instruction or reserved opcode.[1]
7	#NM	Device Not Available (No Math Coprocessor)	Floating-point or WAIT/FWAIT instruction.
8	#DF	Double Fault	Any instruction that can generate an exception, an NMI, or an INTR.

Vector No.	Mnemonic	Description	Source
9		CoProcessor Segment Overrun (reserved)	Floating-point instruction.[2]
10	#TS	Invalid TSS	Task switch or TSS access.
11	#NP	Segment Not Present	Loading segment registers or accessing system segments.
12	#SS	Stack Segment Fault	Stack operations and SS register loads.
13	#GP	General Protection	Any memory reference and other protection checks.
14	#PF	Page Fault	Any memory reference.
15		(Intel reserved. Do not use.)	
16	#MF	Floating-Point Error (Math Fault)	Floating-point or WAIT/FWAIT instruction.
17	#AC	Alignment Check	Any data reference in memory.[3]
18	#MC	Machine Check	Error codes (if any) and source are model dependent.[4]
19-31		(Intel reserved. Do not use.)	
32-255		Maskable Interrupts	External interrupt from INTR pin or INT *n* instruction.

1. The UD2 instruction was introduced in the Pentium® Pro processor.
2. Intel Architecture processors after the Intel386™ processor do not generate this exception.
3. This exception was introduced in the Intel486™ processor.
4. This exception was introduced in the Pentium processor and enhanced in the Pentium Pro processor.

4. Software interrupt Instructions

There are instructions in 8086 which cause an interrupt. They are

- INT instructions with type number specified.
- INT 3, Break Point Interrupt instruction.
- INTO, Interrupt on overflow instruction.

These are instructions at the desired places in a program. When one of these instructions is executed a branch to an ISS takes place. Because their execution results in a branch to an ISS, they are called interrupts. Software Interrupt instructions can be used to test the working of the various Interrupt handlers- For example, we can execute INTO instruction to execute type 0 ISS, without really having to divide a number by 0. Similarly, we can execute INT 2 instruction to test NMI ISS.

4.1 INT-Interrupt Instruction with Type number Specified

The mnemonic for this is INT. It is a 2 byte instruction. The first byte provides the op-code and the second byte the Interrupt type number. Op-code for this instruction is CDH

The execution of an INT instruction, say INTN, when N is the value in the range 00H to FFH, results in the following:

1. Flag register value is pushed on to the stack.
2. CS value of the Return address and IP value of the Return address are push edon to the stack.
3. IP is loaded from the contents of the word location N x 4.
4. CS is loaded from the contents of the next word location.
5. Interrupt Flag and Trap Flag are reset to 0.

Thus a branch to the ISS take place. During the ISS, interrupt are disabled because the Interrupt flag is reset to 0.At the end of the ISS, there will be an IRET instruction. Thus a return back to the interrupted program takes place with Flag registers unchanged.

4.2. INT 3-Break Point Interrupt Instruction

When a break point is inserted, the system executes the instructions up to the breakpoint, and then goes to the break point procedure. Unlike the single-Step feature which stops execution after each instruction, the breakpoint feature executes all the instructions up to the inserted breakpoint and then stops execution. The mnemonic for the instruction is INT3. It is a 1 byte instruction Op-code for this is CCH.

The execution of INT3 instruction results in the following.

1. Flag register value is pushed on to the Stack.
2. CS value of the return address and IP value of the return address are pushed onto the Stack.
3. IP is loaded from the contents of the word location 3x4 = 0000CH.
4. CS is loaded from the contents of the next word location.
5. Interrupt Flag and Trap Flag are reset to 0.

Thus a branch to the ISS takes place. During the ISS, interrupts are disabled because Interrupt flag is reset to 0. At the end of the ISS, there will be an IRET instructionto return back to the interrupted program.A break point interrupt service procedure usually saves all the register contents onthe Stack. Depending upon the system, it may then send the

register contents to the CRTdisplay and wait for the next command from the user.

4.3. INTO - Interrupt on overflow instruction

The 8086 overflow flag, OF, 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 we add the 8-bit signed number 01101100 and the 8- bit signed number 01010001, the signed result will be 10111101. This is correct if we add unsigned binary numbers, but it is not the correct signed result. There are two ways to detect and respond to an overflow error in a program. One way is to put the jump if overflow instruction, JO, immediately after the arithmetic instruction. If the overflow flag is Set, execution will jump to the address specified in the JO instruction. At this address an error routine may be put which respond to the overflow. The second way is to put them INTO instruction immediately after the arithmetic Instruction in the program. The mnemonic for the instruction is INTO. It is a 1 byte instruction. The op-code for this is CEH. It is a conditional interrupt instruction. Only if the overflow flag is Set, a branch takes place to an interrupt handler whose interrupt type number is 4. If the overflow flag is reset, the execution continues with the next instruction. The execution of INTO results in the following.

1. Flag register values are pushed on to the Stack.
2. CS value of the return address and IP value of the return address and IP value of the return address are pushed on to the stack.
3. IP is loaded from the contents of word location 4x4 = 00010H.
4. CS is loaded from the contents of next word location.
5. Interrupt flag and Trap flag are reset to 0.

Thus a branch to ISS takes place. During the ISS, interrupts are disabled. At the end of ISS, there will be an IRET instruction, returning back to the interrupted program. Instructions in the ISS procedure perform the desired response to the error condition.

6.RESET

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin which it shows in table (2). The 8086 RESET is required to be HIGH for greater than 4 CLK cycles. The 8086 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 10 CLK cycles. After this interval the 8086 operates normally beginning with the instruction in absolute location FFFF0H.

Table 2:process initialization register content

CPU Asset	Content
FLAGS Register	0000h
IP	0000h
CS	ffffh
DS	0000h
SS	0000h
ES	0000h
Instruction Queue	Empty