INTERRUPT

Interrupt and Interrupt Application

Introduction

An Interrupt is an event that causes the processor to suspend its present task and transfer the present task and control to new program called ISR(Interrupt Service Routine). Most MPU 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 MPU stops executing its current program and calls a procedure which services the interrupt. The actual code that is invoked when an interrupt occurs is called the **Interrupt Service Routine** (ISR).

Interrupts and exceptions alter the normal program flow in order to handle external events, report errors or exceptional conditions. The difference between interrupts and exceptions is that interrupts are used to handle asynchronous external events while exceptions handle instruction faults. Although a program can generate a software interrupt via an INT N instruction, the processor treats software interrupts as exceptions.

Hardware interrupts occur as the result of an external event and are classified into two types: maskable or non-maskable. Interrupts are serviced after the execution of the current instruction. After the interrupt handler is finished servicing the interrupt, execution proceeds with the instruction immediately after the interrupted instruction. Exceptions are classified as faults, traps, or aborts, depending on the way they are reported and whether or not restart of the instruction causing the exception is supported. Faults are exceptions that are detected and serviced before the execution of the faulting instruction. Traps are exceptions that are reported immediately after the execution of the instruction which caused the problem. Aborts are exceptions which do not permit the precise location of the instruction causing the exception to be determined. Thus, when an interrupt service routine has been completed, execution proceeds from the instruction immediately following the interrupted instruction. On the other hand, the return address from an exception fault routine will always point to the instruction causing the exception and will include any leading instruction prefixes.

Sources of Interrupts

There are three sources of an interrupts and they are as follows.

1. PROCESSOR INTERRUPT

These interrupts are generated by the processor itself, usually in response to an error condition.

For example a type 0 interrupt occurs when attempt to divide by zero.

2. SOFTWARE INTERRUPT

These are special 80x86 instructions that trigger an interrupt response to processor. The general

form of the software interrupt instruction is INT 10H.

3. HARDWARE INTERRUPT

Hardware interrupts are interrupt requests initiated by external hardware. 8086 has 2 pins

reserved for this purpose (NMI and INTR)

The Purpose of Interrupts

Interrupts are particularly useful when interfacing I/O device that provides or require

data relatively low data transfer rates.Unlike the polling sequence, interrupt processing allows the MPU to execute other software while the keyboard operator is thinking about what key to type next .As soon as the key is pressed, the keyboard encoder de-bounces the switch and puts out one pulse that interrupts the MPU.

Trap is a software-invoked interrupt. To execute a trap, you use the 80x86 int (software interrupt) instruction. The main purpose of a trap is to provide a fixed subroutine that various programs can call without having to actually know the run-time address. MS-DOS is the perfect example. The int 21h instruction is an example of a trap invocation. Your programs do not have to know the actual memory address of DOS' entry point to call DOS. Instead, DOS patches the interrupt 21h vector when it loads into memory. When you execute int 21h, the 80x86 automatically transfers control to DOS' entry point, wherever in memory that happens to be.

TYPES OF INTERRUPT

1. MASKABLE INTERRUPT

Those interrupts which can be blocked by user using instructions are called maskable interrupt.

STI-> Enable INTR Input.

CLI-> Disable INTR Input.

INTR is a maskable hardware interrupt. The interrupt can be enabled/disabled using STI/CLI instructions or using more complicated method of updating the FLAGS register with the help of the POPF instruction. When an interrupt occurs, the processor stores FLAGS register into stack, disables further interrupts, fetches from the bus one byte representing interrupt type, and jumps to interrupt processing routine address of which is stored in location 4 * <interrupt type>. Interrupt processing routine should return with the IRET instruction.

2. NON MASKABLE INTERRUPT

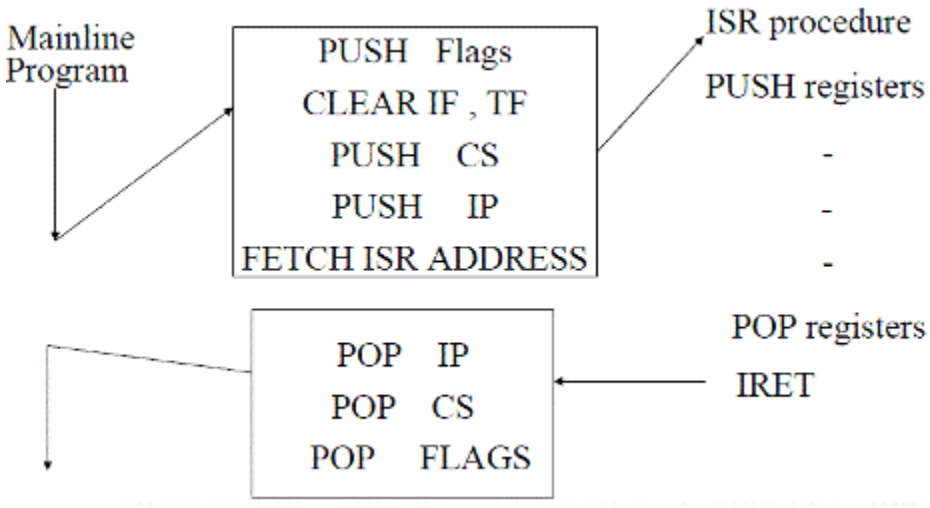
Those interrupts which cannot be blocked by instructions are termed as non maskable interrupt.

NMI-> Non Maskable Interrupt

NMI is a non-maskable interrupt. Interrupt is processed in the same way as the INTR interrupt. Interrupt type of the NMI is 2, i.e. the address of the NMI processing routine is stored in location 0008h. This interrupt has higher priority than the maskable interrupt.

Interrupt Processing

When an interrupt occurs, the following actions happen. First, the current program address and Flags are saved on the stack to allow resumption of the interrupted program. Next, an 8-bit vector is supplied to the Microprocessor which identifies the appropriate entry in the interrupt table. The table contains the starting address of the interrupt service routine. Then, the user supplied interrupt service routine is executed. Finally, when an IRET instruction is executed the old processor state is restored and program execution resumes at the appropriate instruction. The 8-bit interrupt vector is supplied to the Microprocessor in several different ways: exceptions supply the interrupt vector internally; software INT instructions contain or imply the vector; maskable hardware interrupts supply the 8-bit vector via the interrupt acknowledges bus sequence. Non- Maskable hardware interrupts are assigned to interrupt vector 2.



To summarize:

When an interrupt occurs, 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.

8086 INTERRUPTS

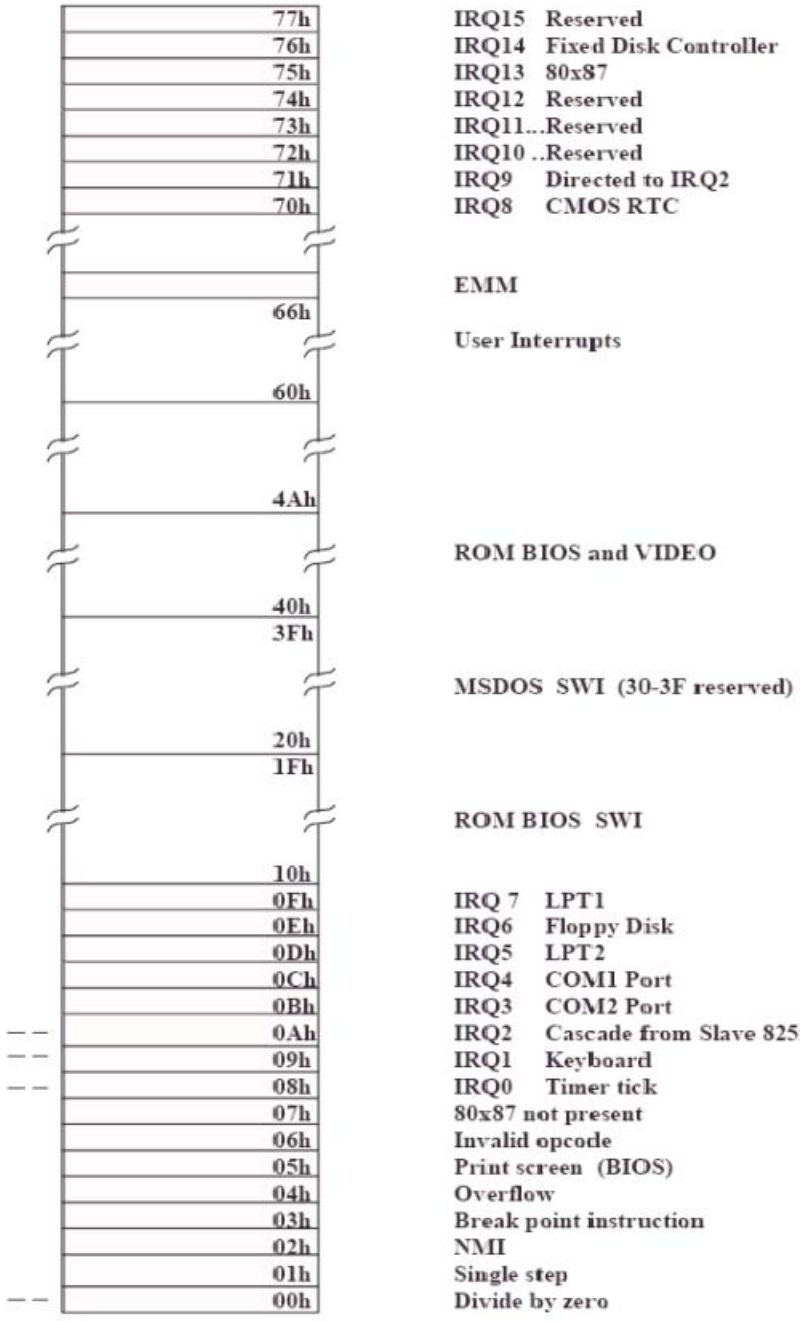
The interrupt of entire Intel family of MPU include 2 hardware pins that request interrupts (INTR and NMI).And other one hardware pin INTA acknowledges the interrupt request through INTR. An 8086 interrupt can come from any one of 2 sources. One source is an external signal applied to the Nonmaskable Interrupt (NMI) or to the (INTR) input pin. An interrupt caused by a signal applied to one of these inputs (NMI or INTR) is referred as Hardware Interrupt. A second source of an interrupt is execution of the interrupt instruction INT. This is referred as Software Interrupt. Software Interrupts can be caused by:

INT instruction - breakpoint interrupt. This is a type 3 interrupt.
INT <interrupt number> instruction - any one interrupt from available 256 interrupts.
INTO instruction - interrupt on overflow
Single-step interrupt is generated if the TF flag is set. This is a type 1 interrupt. When the
CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.

The third source of an interrupt is some error condition produced in the 8086 by the Execution of an instruction. Example: Processor Exceptions: Divide Error (Type 0), Unused Opcode (type 6) and Escape opcode (type 7).

INTERRUPT VECTORS

The Interrupt Vector Table is located in the first 1024 bytes of memory at address 00000H – 0003FFH.It contains 256 different 4 byte Interrupt Vectors. An Interrupt Vector contains the address (segment and offset) of the Interrupt Service Procedure. Each vector is a 4 byte long and contains the starting address of the Interrupt Service Routine.



Interrupt Vector Table

The first 2 bytes of the vector contain the offset address and the last two bytes contain the segment address.

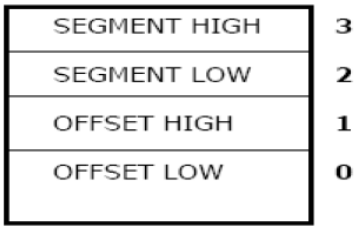


Figure : Interrupt Vector

8086 INTERRUPT TYPES

1. DIVIDE BY ZERO INTERRRUPT: TYPE 0

Divide error occurs when the result of division overflow or whenever an attempt is made to divide by zero.The 8086 type 0 is automatic and cannot be disabled in any way

2. SINGLE STEP INTERRUPT: TYPE 1

If the 8086 trap flag is set, the 8086 will automatically do a type 1 interrupt after each instruction executes. Upon accepting this interrupt, the TF bit is cleared so that Interrupt Service Procedure executes. When the 8086 does a type 1 interrupt, it pushes the flag register on the stack, resets TF and IF and pushes CS and IP value for the next instruction on the stack.It gets CS value for the start of type 1 interrupt service procedure from address 00006H and gets IP value for the start of the procedure from the address 00004H. When you tell a system to single step, it will execute one instruction and stop.You can then examine the contents of registers and memory locations. In other words, in single step mode, a system will stop after it executes each instruction and waits for further direction from you.

3. NON MASKABLE INTERRUPT: TYPE 2

A result of placing logic 1 on the NMI input pin causes type 2 interrupt. This input is non-maskable, which means that it cannot be disabled. Another common use of type 2 interrupt is to save program data in case of a system power failure or to deal with some other catastrophic failure conditions. Some external circuitry detects when the AC power to the system fails and sends an Interrupt signal to the NMI.

4. BREAKPOINT INTERRUPT: TYPE 3

The INT 3 instruction is often used to store a breakpoint in a program for debugging. A break point is used to examine the CPU and memory after the execution of a group of Instructions. It is one byte instruction whereas other instructions of the form “INT nn” are 2 byte instructions.

5. OVERFLOW INTERRUPT: TYPE 4

A special vector used with the INT0 instruction. The INT0 instruction interrupts the program if an overflow condition exists as reflected by overflow flag (OF). There is an instruction associated with this INT 0 (interrupt on overflow).If INT 0 is placed after a signed number arithmetic as IMUL or ADD the CPU will activate INT 04 if 0F = 1.In case where 0F = 0 , the INT 0 is not executed but is bypassed and acts as a NOP.

Real Mode and Protected mode operation of Interrupt

In protected mode , interrupts have exactly the same assignments as in real mode but IVT is different. In place of interrupt vector , protected mode uses a set of 256 interrupt descriptors stored in **interrupt descriptors table(IDT)** . The interrupt descriptor table is 256x8(2k) bytes long with each descriptor containing eight bytes. The interrupt descriptor table is located at any memory location within the system by interrupt descriptor table address register (IDTR). Each entry in the IDT contains the address of the interrupt service procedure in the form of a segment selector and a 32-bit offset address.

Real mode interrupt vectors can be converted into protected mode interrupts by copying the interrupt procedure addresses from the interrupt vector table and converting them to 32-bit offset addresses that are stored in the interrupt descriptors. A single selector and segment descriptor can be placed in the global descriptor table that identifies the first 1M byte of memory as the interrupt segment.

PRIORITY OF 8086 INTERRUPTS

INTERRUPTS	PRIORITY
DIVIDE ERROR,INT n , INT0	HIGHEST
NMI	
INTR	
SINGLE STEP	LOWEST

INTERRUPT PRIORITY

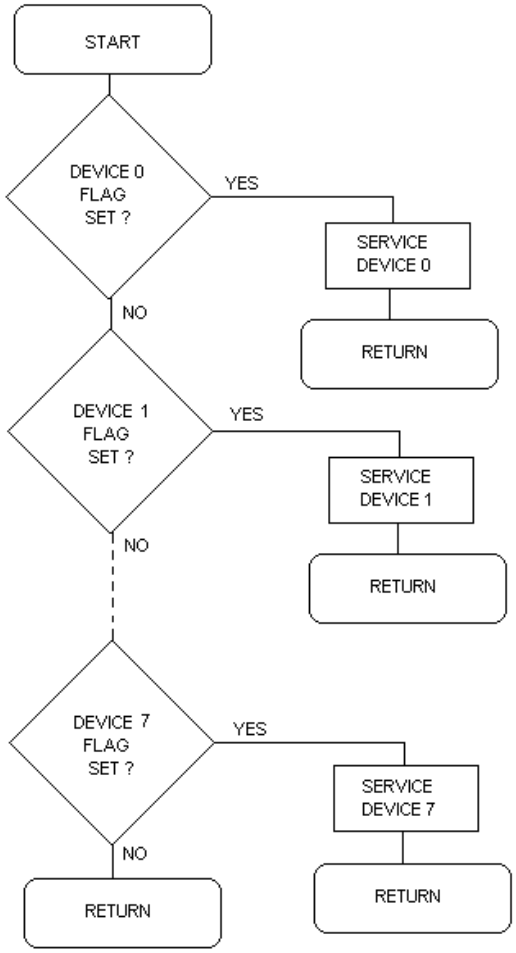
Multiple Interrupts

If more than one device is connected to the interrupt line, the processor needs to know to which device service routine it should branch to. The identification of the device requesting service can be done in either hardware or software, or a combination of both. The three main methods are:

1. Software Polling,
2. Hardware Polling, (Daisy Chain),
3. Hardware Identification (Vectored Interrupts).

Software Polling Determination of the Requesting Device

A software routine is used to identify the device requesting service. A simple polling technique is used, each device is checked to see if it was the one needing service. Having identified the device, the processor then branches to the appropriate interrupt-handling-routine address for the given device. The order in which the devices appear in the polling sequence determines their priority.



SOFTWARE POLLING FLOWCHART

Summary of Software Polled I/O

Polling is the most common and simplest method of I/O control. It requires no special hardware and all I/O transfers are controlled by the CPU programme. Polling is a synchronous mechanism, by which devices are serviced in sequential order.

The polling technique, however, has limitations.

1) it is wasteful of the processors time, as it needlessly checks the status of all devices all the time,

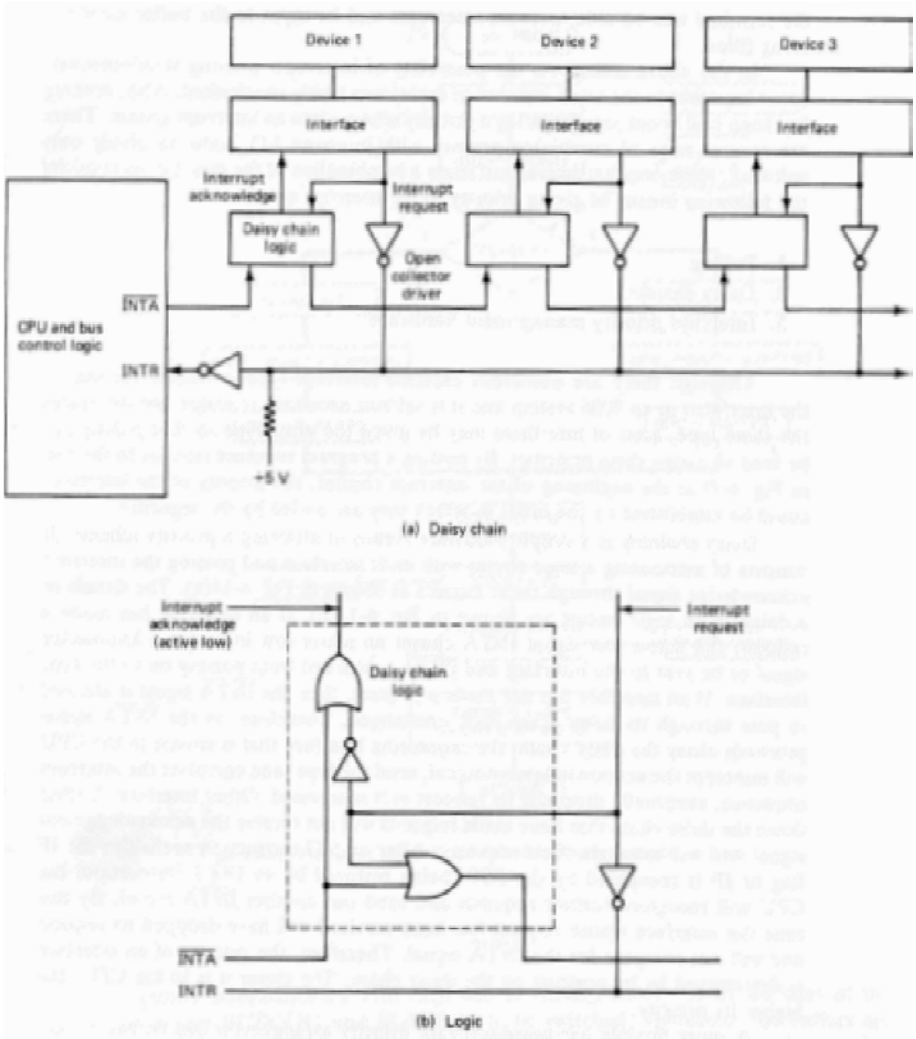
2) it is inherently slow, as it checks the status of all I/O devices before it comes back to check any given one again,

3) when fast devices are connected to a system, polling may simply not be fast enough to satisfy the minimum service requirements,

4) priority of the device is determined by the order in the polling loop, but it is possible to change it via software.

Software/Hardware Driven Identification (Daisy Chain)

This is significantly faster than a pure software approach. A daisy chain is used to identify the device requesting service.



Daisy Chain Polling Arangement

Daisy chaining is used for level sensitive interrupts, which act like a wired 'OR' gate. Any requesting device can take the interrupt line low, and keep it asserted low until it is serviced.

Because more than one device can assert the shared interrupt line simultaneously, some method must be employed to ensure device priority. This is done using the interrupt acknowledge signal generated by the processor in response to an interrupt request.

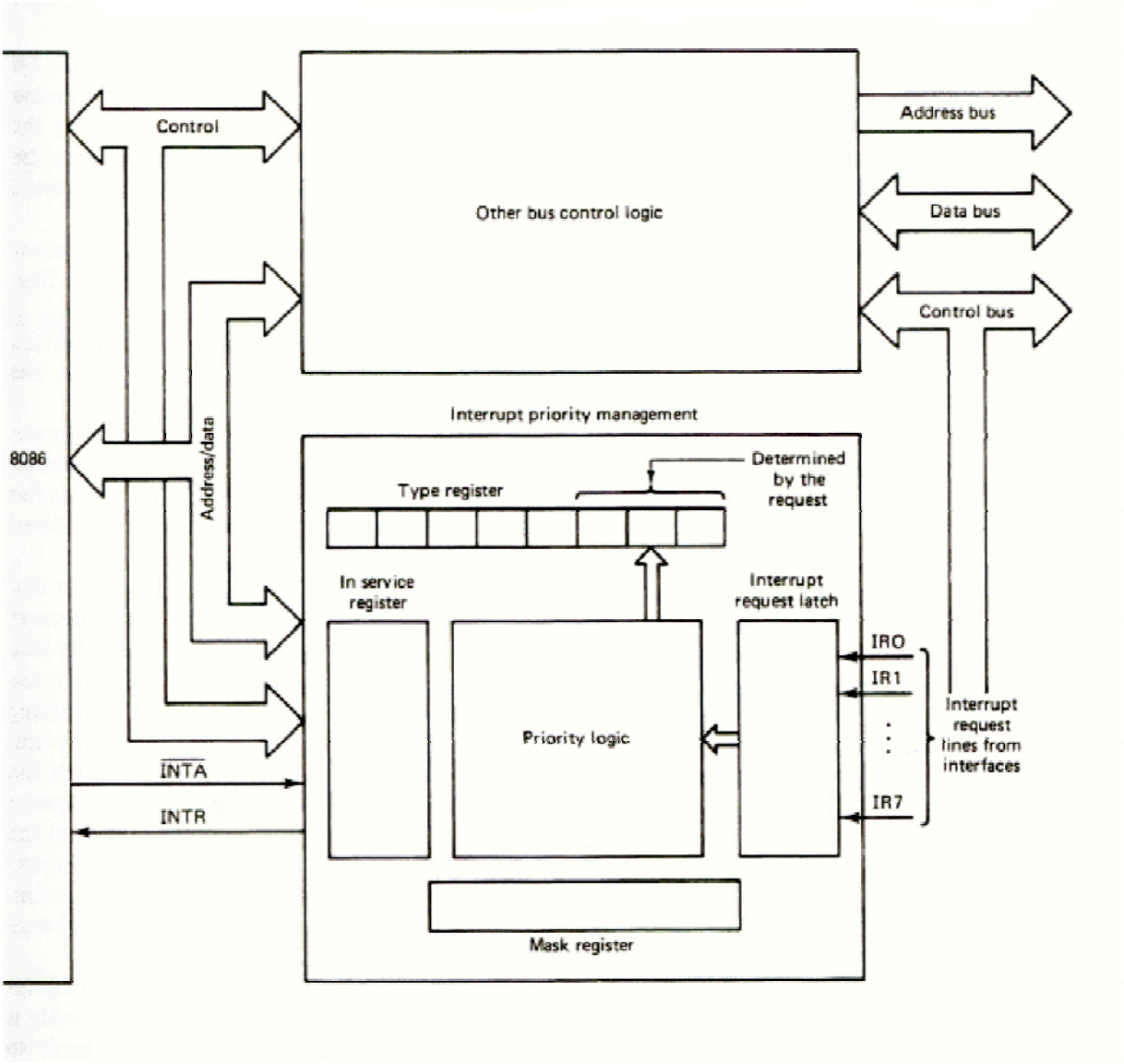
Each device is connected to the same interrupt request line, but the interrupt acknowledge line is passed through each device, from the highest priority device first, to the lowest priority device last.

After preserving the required registers, the microprocessor generates an interrupt acknowledge signal. This is gated through each device. If device 1 generated the interrupt, it will place its identification signal on the data bus, which is read by the processor, and used to generate the address of the interrupt-service routine. If device 1 did not request the servicing, it will pass the interrupt acknowledge signal on to the next device in the chain. Device 2 follows the same procedure, and so on.

Hardware Identification (Vectored Interrupts)

This is the fastest system. The focus is placed on the requesting device to request the interrupt, and identify itself. The identity could be a branching address for the desired interrupt-handling routine.

If the device just supplies an identification number, this can be used in conjunction with a lookup table to determine the address of the required service routine. Response time is best when the device requesting service also supplies a branching address.



Priority Interrupt Management Controller

Priority Interrupt Controller Chips (PIC's) are hardware chips designed to make the task of a device presenting its own address to the CPU simple. The PIC also assesses the priority of the devices connected to it. Modern PIC's can also be programmed to prevent the generation of interrupts which are lower than a desired level.

The decoded location is connected to the output of a priority encoder. The input of the priority encoder is connected to each device. When a device requests service, the priority encoder presents a special code combination (unique for each device) to the decoded memory location. The port thus holds the value or address associated with the highest device requesting service.

The priority encoder arranges all devices in a list, devices given a lower priority are serviced when no other higher priority devices need servicing. This simplifies the software required to determine the device, resulting in an increase in speed.

The disadvantages are:

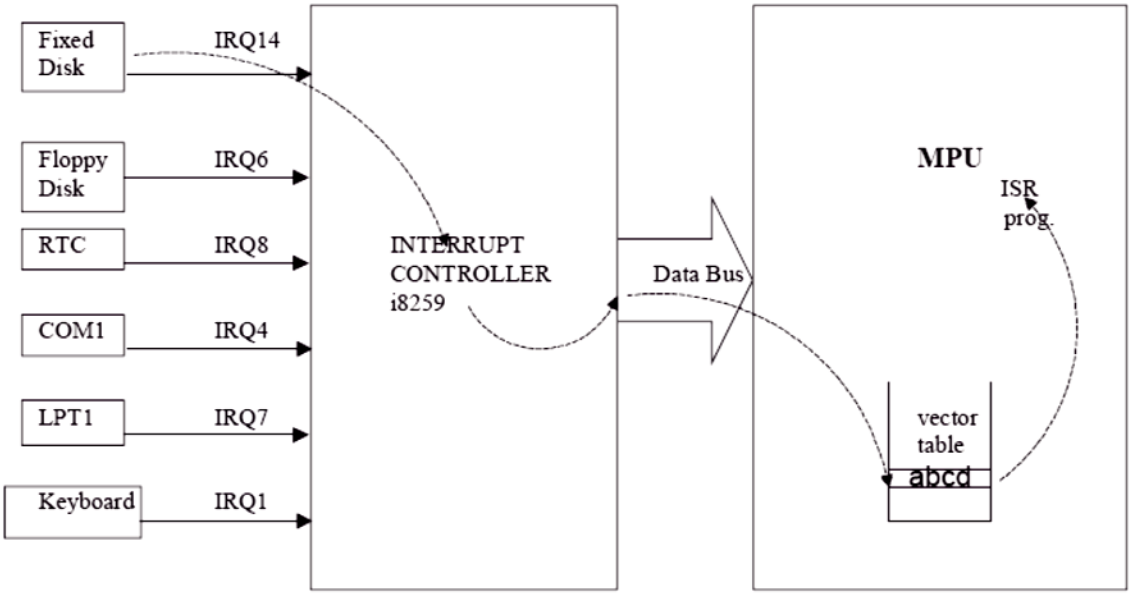
1) the extra chip required,

2) resultant increases in cost,

3) more board space and power consumption,

4) fixed priority in hardware.

8259A PRIORITY INTERRUPT CONTROLLER



8086 have only 2 interrupt input NMI and INTR. If we save NMI for a power failure, only one pin is left for all other Interrupts. Providing all the interrupts from a single pin is difficult. To solve above problem, we use an external device called a Priority Interrupt Controller. It funnels the interrupt signals into a single interrupt input on the processor. The process is as follows :

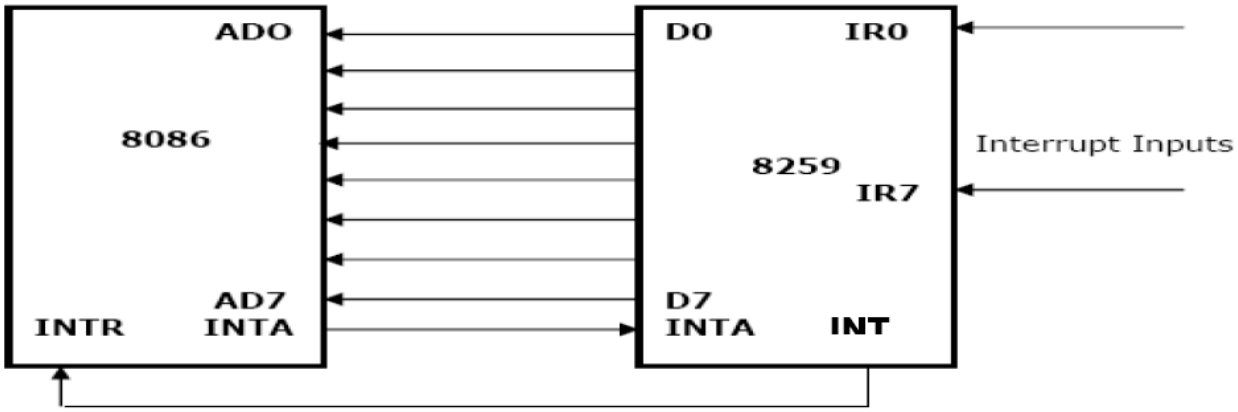
- INTR Interrupt Request comes to 8086 at INTR pin .
- 8086 will send out Interrupt Ack pulse on its INTA pin to the INTA pin of the 8259A PIC.

- The INTA pulse tells the 8259A to send the desired interrupt type on the data bus.
- Multiply the interrupt type it receives from the 8259A by 4 to produce an address in the Interrupt Vector Table.
- Push the flags on the stack.
- Clear IF and TF
- Push the return address on the stack.
- Get the starting address for ISR (CS: IP).
- Execute Interrupt Service Routine.
- Return from Interrupt Service Routine.

In Real Mode the address of the ISR is stored in four consecutive memory locations (Double Word) in an IVT.When an Interrupt occurs an 8 bit number is supplied to the processor, which identifies the appropriate entry in this table.

The method for determining the type number depends on the interrupt source.

- Software Interrupts supply the number as a part of instruction (INT n, where n is the type number).
- Internal interrupts have predefined type number for e.g. TYPE 0 – Divide by Zero.
- NMI hardware interrupt is predefined as type 2.
- INTR however no predefined type number has and must get its type number onto data bus D0-D7.



COMPUTING THE ISR ADDRESS

LIST OF PROBLEMS

PROBLEM 1

A particular real mode interrupt has a type no n = 41H. If the corresponding ISR begins at address 09E3:0010, determine the location in the vector table to store this address.

SOLUTION

- The vector address is calculated by multiplying 41H by 4.
-This is done most easily by rotating 41H left twice
41H = 000001000001 -> Rotate Twice -> 0001 0000 0100 = 104H
Hence 00104H is the starting address.
00107 = 09 |
00106 = E3 | -> CODE SEGMENT
00105 = 00 |
00104 = 10 | -> INSTRUCTION POINTER

PROBLEM 2

Calculate the IVT address when operated in real mode of the 80x86 processor's NMI interrupt input. Assume the vector points to memory location E010:1000H

PROBLEM 3

An 80x86 processor stores the byte 3A49:2F1C beginning at address 0008CH. What interrupt type number does this address correspond to?

INTERRUPT LATENCY

The term interrupt latency refers to the amount of time it takes a system to respond to an interrupt. It is the duration between an interrupt request and its type identification. It is the time interval from when the interrupt is first asserted to the time the CPU recognizes it. This will depend much upon whether interrupts are disabled, prioritized and what the processor is currently executing. At times, a processor might ignore requests whilst executing some indivisible instruction stream (read-write-modify cycle). The figure that matters most is the longest possible interrupt latency time. Interrupt Response Time is the time interval between the CPU recognizing the interrupt to the time when the first instruction of the interrupt service routine is executed. This is determined by the processor architecture and clock speed.

VECTORED VS POLLED INTERRUPT

1. VECTORED INTERRUPT

In vectored interrupt, the interrupt signal includes the identity of the device sending the interrupt signal. In a computer, a vectored interrupt is an I/O interrupt that tells the part of the computer that handles I/O interrupts at the hardware level that a request for attention from an I/O device has been received and also identifies the device that sent the request. In a vectored interrupt, the source that interrupts supplies the branch information (Interrupt Vector) to the CPU.

2. POLLED INTERRUPT (Hardware Polled)

It is an interrupt handler that requires the interrupt handler poll or sends a signal to each device in order to find out which one sent the interrupt request. In a computer, a polled interrupt is a specific type of I/O interrupt that notifies that part of the computer containing the I/O interface that a device is ready to be read or otherwise handled but does not indicate which device. The interrupt controller must poll each device to determine which one made the request. In a non-vectored interrupt, the branch address is assigned to a fixed location in the memory. It generally implements daisy chain type of hardware polling .

Interrupt Processing Sequence for Software interrupt .