8086 Interrupts

Lecture References:

- Book:
  - *Microprocessors and Interfacing: Programming and Hardware, Chapter # 08,* **Author:** Douglas V. Hall
  - *Microprocessor and Microcomputer – Based System Design,* **Author:** Mohamed Rafiquzzaman

- Lecture Materials:
 - *M. A. Sattar, BUET, Dhaka, Bangladesh.*

Interrupt and Its Classifications

- Interrupt is a process where a normal program execution to be interrupted by some external signal or by a special instruction in the program.

- Microprocessor pay attention to the interrupt stopping the the current execution.

- <u>Classification of Interrupts</u>
  - Interrupts can be classified into two types:
    - <u>Maskable Interrupts</u> (Can be delayed or Rejected)
      - □ User-defined interrupts

    - <u>Non-Maskable Interrupts</u> (Can not be delayed or Rejected)
      - □ System Interrupts

Interrupt & Its Consequences over MP

▸ An interrupt is considered to be an emergency signal that may be serviced.

  ▸ The Microprocessor may respond to it as soon as possible.

▸ **What happens when MP is interrupted ?**

  ▸ When the Microprocessor receives an interrupt signal, it suspends the currently executing program and jumps to an **Interrupt Service Routine** (ISR) to respond to the incoming interrupt.

  ▸ Each interrupt will have its own ISR.

  ▸ After finishing the second program/interrupt, automatically return to the first program and start execution from where it was left
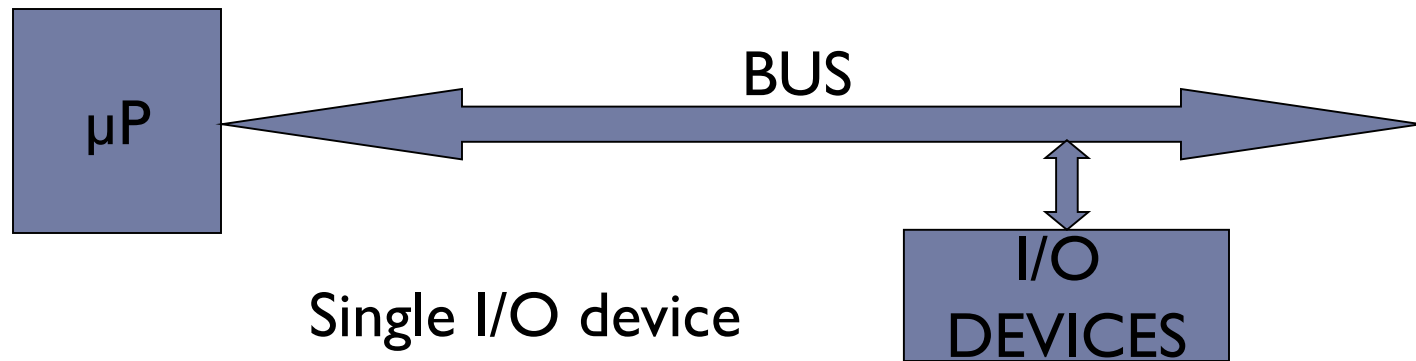
Why Interrupt is Necessary?

▸ To understand this we will have to review how μP/μC communicate with the outside world.

BUS

μP

MEMORY
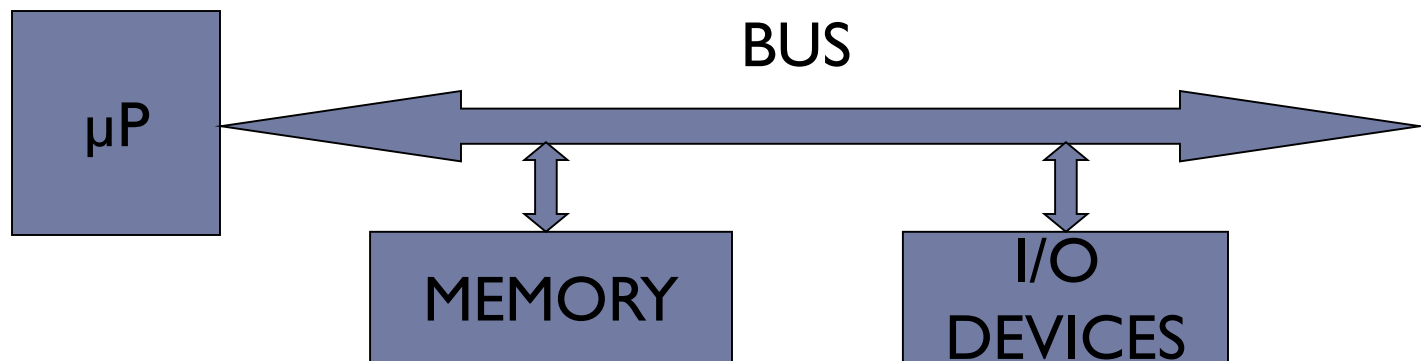
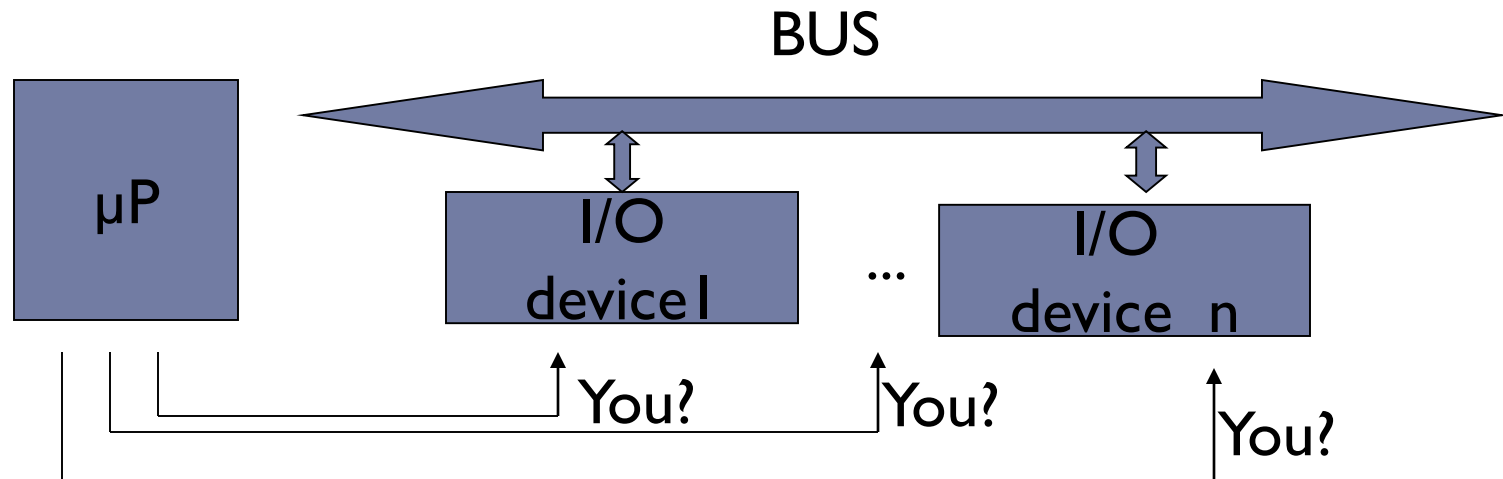I/O DEVICES

Why Interrupt is Necessary?

▸ **First Type:** DEDICATED communication between MP and I/O devices.

μP

BUS

I/O DEVICES

Single I/O device

Why Interrupt is Necessary?

▸ **Second Type:** POLLED I/O or PROGRAMMED I/O communication between MP and I/O devices.

BUS

μP

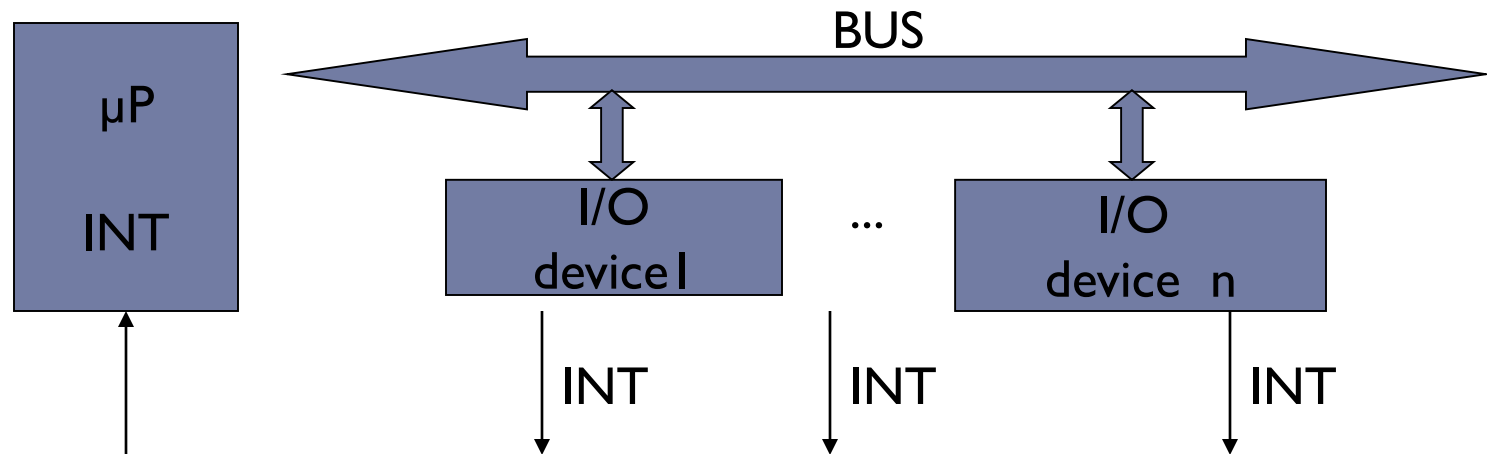I/O device 1 … I/O device n

You? You? You?

Disadvantages of Second Type Communication:
▸ not fast enough
▸ waste too much microprocessor time

Why Interrupt is Necessary?

▸ **Third Type:** INTERRUPTED I/O communication between MP and I/O devices.

Interrupts are particularly useful when I/O devices are slow

Polling and Interrupt

- Both are methods to notify processor that I/O device needs attention
- **Polling**
  - simple, but slow
  - processor check status of I/O device regularly to see if it needs attention
  - *similar to checking a telephone without bells*!
- **Interrupt**
  - fast, but more complicated
  - processor is notified by I/O device (interrupted) when device needs attention
  - *similar to a telephone with bells*

Interrupt Concept

- Intel processors include two hardware pins (INTR and NMI) that request interrupts.

- And one hardware pin (INTA) to acknowledge the interrupt requested through INTR.

- The processor also has software interrupts INT, INTO and INT 3.

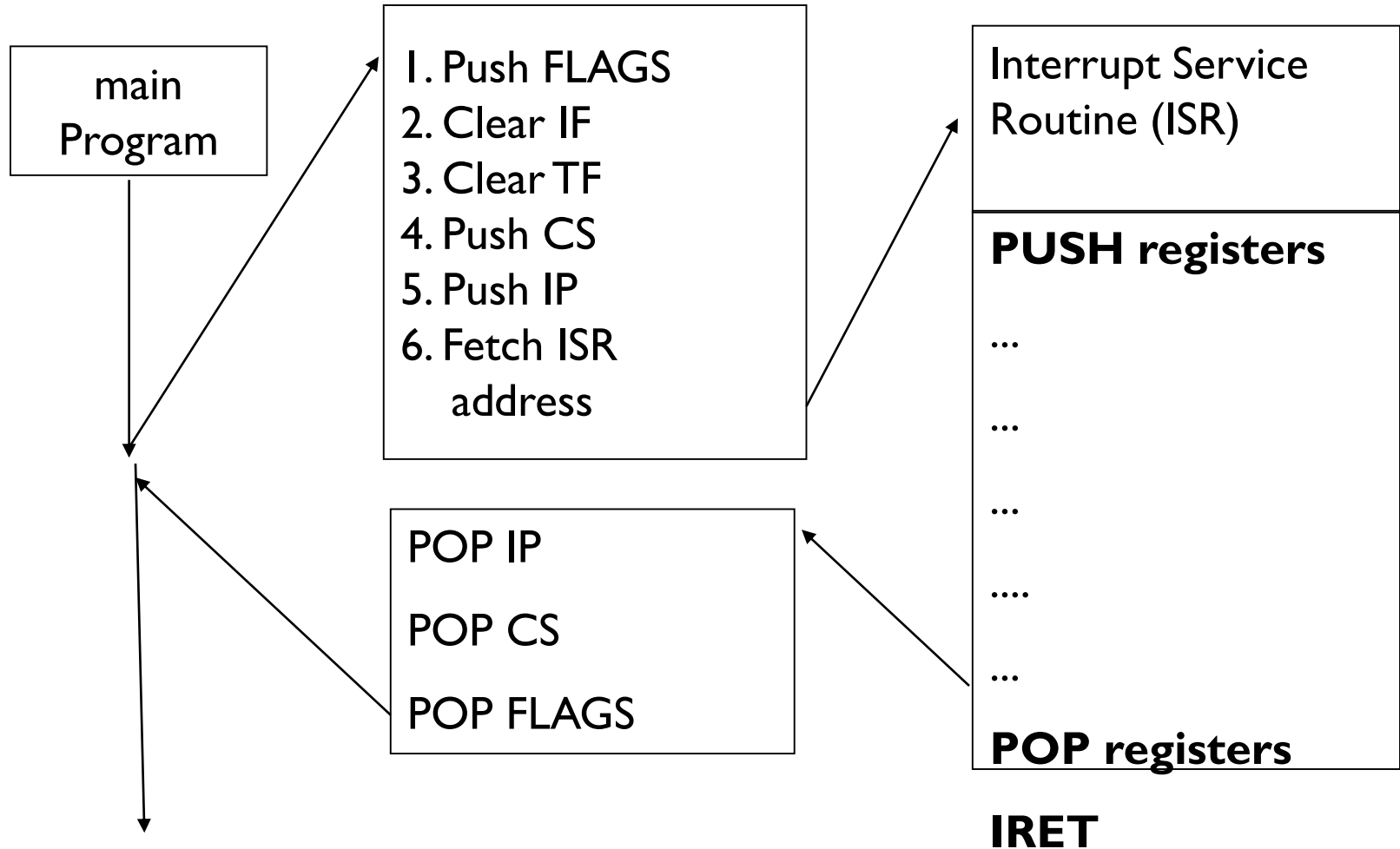- Flag bits IF (interrupt flag) and TF (trap flag), are also used with the interrupt structure.

Classifications of 8086 Interrupts

▶ An **8086** interrupt can come from any of the ***three*** sources:

  ▶ An ***external signal*** applied to NMI or INTR pin.
    – known as ***hardware interruption***
    – *It is a **user-defined interrupt***
    – *Example: Connecting I/O Device*

  ▶ Execution of interrupt instruction **INT.**
    – referred as ***software interruption***
    – *It is also a **user-defined interrupt***
    – *Example: INT 21h, INTO, INT 3*

  ▶ Some error condition produced by execution of an instruction, e.g., trying to divide some number by zero.
    ▶ It is known as ***pre-defined interrupt***

Function of 8086 during Interrupts

▶ At the end of each instruction cycle, 8086 checks to see if any interrupts have been requested.

▶ If yes, then 8086 responds to the interrupt by stepping through the following series of major actions:

  ▶ It decremented SP by 2 and pushes *Flag register* on the stack.

  ▶ It disables 8086 **INTR** input by clearing **IF (Interrupt) flag** in Flag register, IF=1.

  ▶ It resets the **TF (Trap) flag** in Flag register

  ▶ It decremented SP again by 2 and pushes current **CS (Code Segment)** contents on the stack.

  ▶ It decremented SP again by 2 and pushes current **IP (Instruction Pointer)** contents on the stack.

  ▶ It does an indirect far **Jump** to the start of the procedure (**ISR**) written to respond to the interrupt.

Function of 8086 during Interrupts

main Program

1. Push FLAGS
2. Clear IF
3. Clear TF
4. Push CS
5. Push IP
6. Fetch ISR address

Interrupt Service Routine (ISR)

PUSH registers

...

...

...

....

...

POP registers

IRET

POP IP

POP CS

POP FLAGS

Interrupt Vectors and Vector Table

- An **interrupt vector** is a **pointer** to where the ISR is stored in memory.

- All interrupts (vectored or otherwise) are mapped onto a memory area called the **Interrupt Vector Table (IVT)**.

  - The IVT is usually located in the first 1 Kbyte of memory segment (from 00000 H - 003FF H).

  - The purpose of the IVT is to hold the vectors that redirect the microprocessor to the right place when an interrupt arrives.

- The starting address of an ISR is often called

  - the *interrupt vector* or the *interrupt pointer.*

- So the Table is referred to as

  - *interrupt-vector table* or *interrupt-pointer table.*

Interrupt Types based on ISR ID

▸ Note that

  ▸ The **IP** value is put in as the **low word** of the vector

  ▸ **CS** as **high word** of the vector

▸ 4 bytes are required to store the CS and IP values for each interrupt service procedure, the *interrupt-vector table* can hold starting addresses for up to 256 interrupt procedures.

▸ Each *Double Word* interrupt vector is identified by a number from 0 to 255

▸ *INTEL* calls this number the *TYPE* of the interrupt

Interrupt Types based on ISR ID

AVAILABLE FOR USER (224)	3FFH	TYPE 255
		...
	080H	TYPE 32
RESERVED (27)		TYPE 31
		...
	014H	TYPE 5
Predefined/ Dedicated/Internal Interrupts Pointers (5)		TYPE 4
	010H	INTO OVERFLOW
		TYPE 3
	00CH	INT
		TYPE 2
	008H	NON-MASKABLE
		TYPE 1
	004H	SINGLE STEP
		TYPE 0
	000H	DIVIDE ERROR

CS Base Address

IP Offset

Interrupt Types based on ISR ID

▸ The first five interrupt vectors are identical in all Intel processors

▸ Intel reserves the first 32 interrupt vectors

▸ The last 224 interrupt vectors are user-available

▸ Each is four bytes long in real mode and contains the starting address of the interrupt service procedure.

  ▸ The first two bytes contain the offset address

  ▸ The last two contain the segment address

Interrupt Types based on ISR ID

▶ **Type 0**

  ▶ The **divide error** whenever the result from a division overflows or an attempt is made to divide by zero.

▶ **Type 1**

  ▶ Single-step or trap occurs after execution of each instruction if the trap (TF) flag bit is set.

  ▶ Upon accepting this interrupt, TF bit is cleared so the interrupt service procedure executes at full speed.

Interrupt Types based on ISR ID

- ## **Type 2**
  - The **non-maskable interrupt** occurs when a logic 1 is placed on the NMI input pin to the microprocessor.
  - Non-maskable—it cannot be disabled

- ## **Type 3**
  - A special one-byte instruction (INT 3) that uses this vector to access its interrupt-service procedure.
  - Often used to store a breakpoint in a program for debugging

Interrupt Types based on ISR ID

- **Type 4**
  - **Overflow** is a special vector used with the INTO instruction. The INTO instruction interrupts the program if an overflow condition exists.
  - As reflected by the overflow flag (OF)

Summary of 8086 Interrupt Function ...

- **How does 8086 get to Interrupt Service Routine (ISR)?**
  - Simple. It loads its CS and IP registers with the address of ISR.
  - So, the next instruction to be executed is the first instruction of ISR.

- **How does 8086 get the address of Interrupt Service Routine (ISR)?**
  - It goes to **specified memory location** to fetch **four consecutive bytes**
    - higher two bytes to be used as CS (Code Segment)
    - lower two bytes to be used as IP (Instruction Pointer)

Summary of 8086 Interrupt Function …

▶ **How does 8086 get the address of that specified memory location?**

  ▶ In an 8086 system, the first 1Kbytes of memory, from 00000 to 003FF, is set aside as a **Table** for storing the starting addresses of **Interrupt Service Routines** (ISR).

  ▶ Since 4 bytes are required to store **CS and IP** values for each ISR, the **Table** can hold the starting addresses for up to 256 ISRs.

Summary of 8086 Interrupt Function …

- **How does 8086 get the address of a particular ISR?**

  - In an 8086 system, each "interrupter" has an id #

  - 8086 treat this id # as interruption type #

  - After receiving INTR signal, 8086 sends an INTA signal

  - After receiving INTA signal, interrupter releases it's id #, i.e., type # of the interruption.

  - 8086 multiplies this id# or type# by 4 to produced the desired address in the *vector table*

  - 8086 reads 4 bytes of memory starting from this address to get the starting address of ISR

    - lower 2 byte is loaded in to IP

    - higher 2 bytes to CS

Summary of 8086 Interrupt Function …

▶ **What happens if two or more interrupts occur at the same time?**

  ▶ Higher priority interrupts will be served first

Interrupt Type	Priority
DIVIDE ERROR, INT n, INTO NMI INTR	HIGHEST
SINGLE STEP	LOWEST

Thank You !!