Department of Computer Science & Engineering
BRAC University.

Interrupts

Course ID: CSE341
Course Title: Microprocessors

Lecture References:

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

Interrupt

- 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 current execution.

- **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

Interrupt

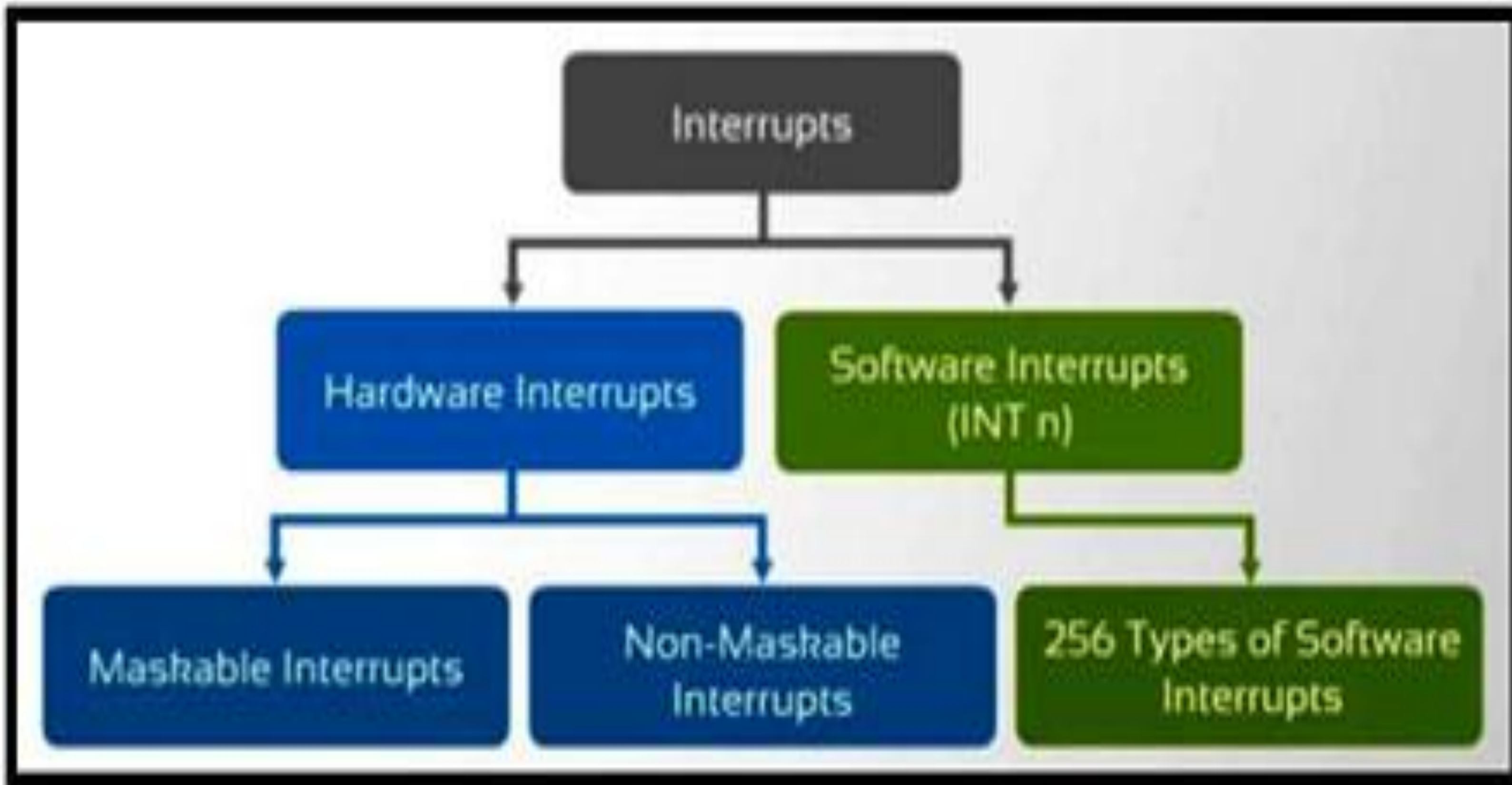
- The processor can be interrupted in the following ways:

  i) by an external signal generated by a peripheral, such as Keyboard, Mouse, Printer, etc.

  ii) by an internal signal generated by a special instruction in the program,

  iii) by an internal signal generated due to an exceptional condition which occurs while executing an instruction.

Classification of Interrupts



Hardware Interrupt

Hardware Interrupts:

- The interrupts initiated by external hardware by sending an appropriate signal to the interrupt pin of the processor is called hardware interrupt.
- The 8086 processor has two interrupt pins INTR (18) and NMI (17).
- The interrupts initiated by applying appropriate signal to these pins are called hardware interrupts of 8086.
- Hardware Interrupts Used to handle external hardware peripherals such as key boards , mouse , hard disks , floppy disks , DVD drivers, printers, mouse, DVD drivers, floppy disks, key boards etc.

Maskable and Non-Maskable

- The processor has the facility for accepting or rejecting hardware interrupts.
  - **Maskable Interrupts** (Can be delayed or Rejected) The interrupts whose request can be either accepted or rejected or delayed by the processor are called maskable interrupts.
  - Example: User-defined interrupts. In 8086 processor all the hardware interrupts initiated through INTR pin are maskable by clearing interrupt flag (IF).

  - **Non-Maskable Interrupts** (Can not be delayed or Rejected) The interrupts whose request has to be definitely accepted (or cannot be rejected) by the processor are called non-maskable interrupts.
  - Example: The interrupt initiated through NMI pin and all software interrupts are non-maskable.

Examples

- Non-Maskable Interrupts
  - Used during power failure
  - Used during critical response time
  - Used during non-recoverable hardware errors
  - Used during memory parity errors

- Maskable Interrupts
  - Interrupt from mouse, keyboard or other peripherals

Software Interrupts

- The software interrupts are program instructions.
- These instructions are inserted at desired locations in a program.
- While running a program, if software interrupt instruction is encountered then the processor initiates an interrupt.
- The 8086 processor has 256 types (00 to FF) or (00 to 255) of software interrupts. The software interrupt instruction is INT n, where n is the type number in the range 0 to 255.

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***
  - Execution of interrupt instruction ***INT.***
    - referred as ***software interruption***
    - *It is also a* ***user-defined interrupt***
  - 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***

Interrupt Vectors and Vector Table

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

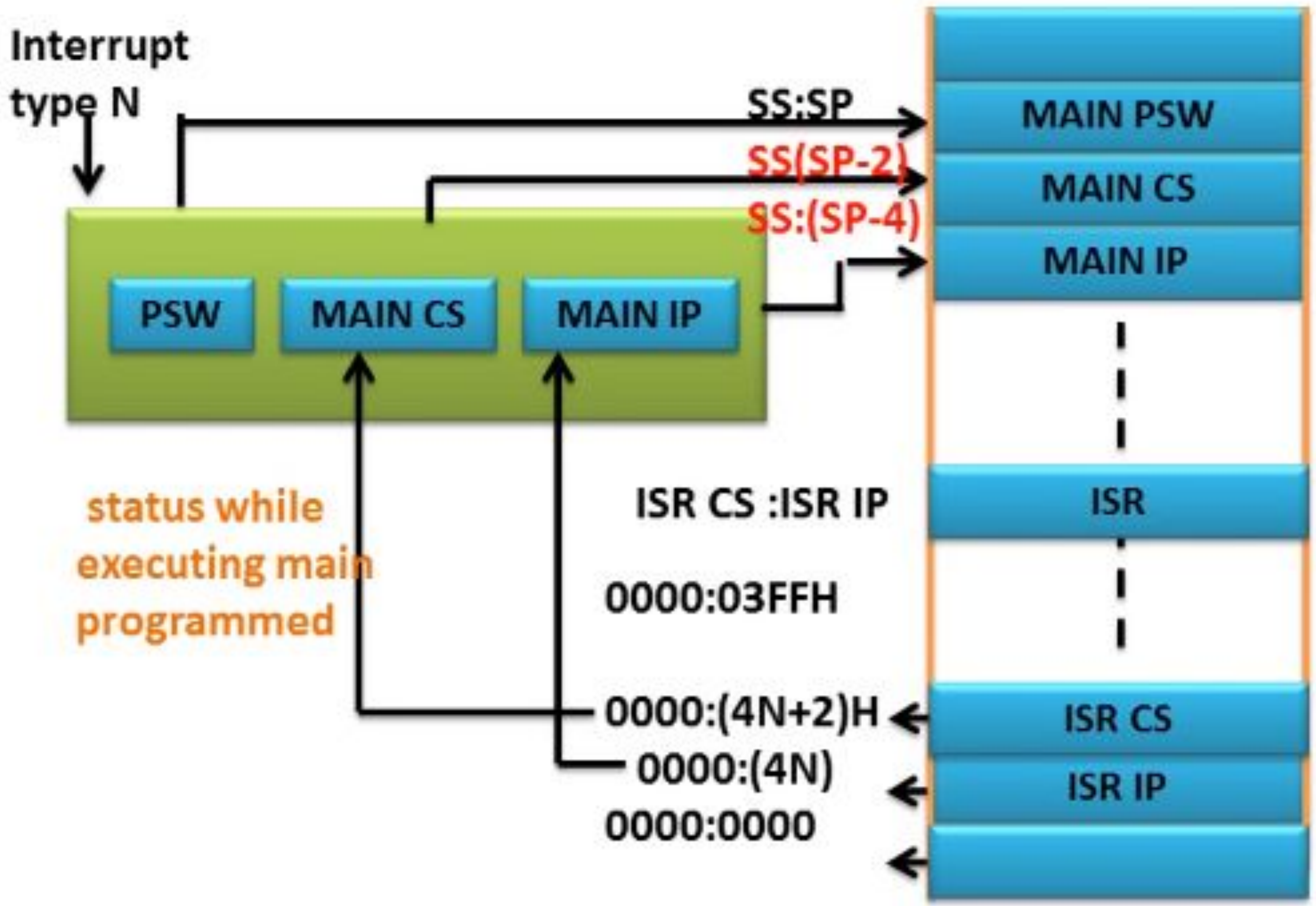
AVAILABLE FOR USER (224)	003FFH	TYPE 255
		...
	00080H	TYPE 32
RESERVED (27)		TYPE 31
		...
	00014H	TYPE 5
Predefined/ Dedicated/Internal Interrupts Pointers (5)		TYPE 4
	00010H	INTO OVERFLOW
		TYPE 3
	0000CH	INT
		TYPE 2
	00008H	NON-MASKABLE
		TYPE 1
	00004H	SINGLE STEP
		TYPE 0
	00000H	DIVIDE ERROR

CS Base Address
IP Offset

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
- CS = 2 bytes, IP = 2 Bytes, so total 4 bytes are needed for Interrupt and there are total 256 interrupts, where 256 x 4 = 1024 Bytes = 1 KB are located in the starting part of memory
- CS:IP = 0000:0000 to 0000:03FF

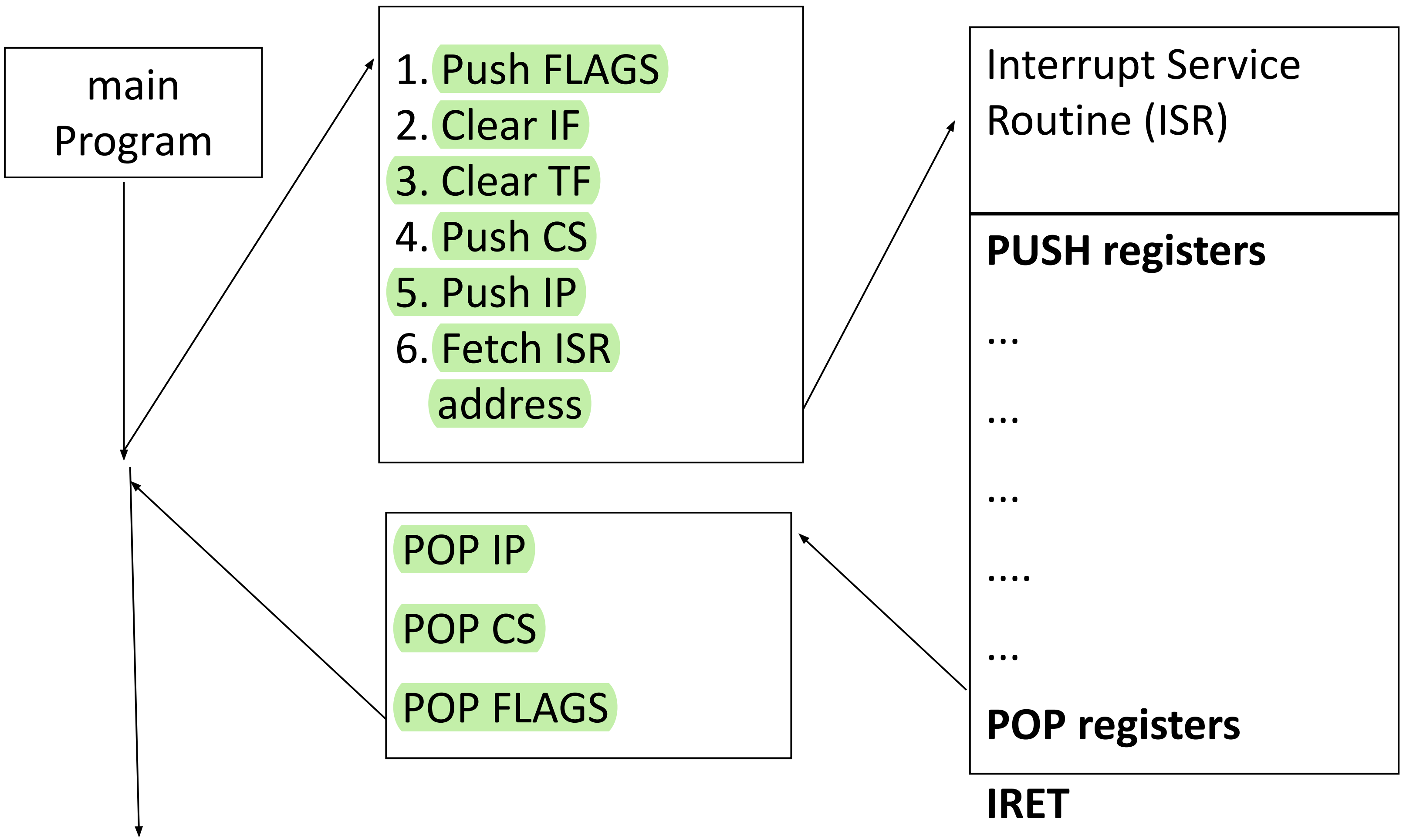
How Interrupt is serviced



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
  - 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 written to respond to the interrupt.

Function of 8086 during Interrupts



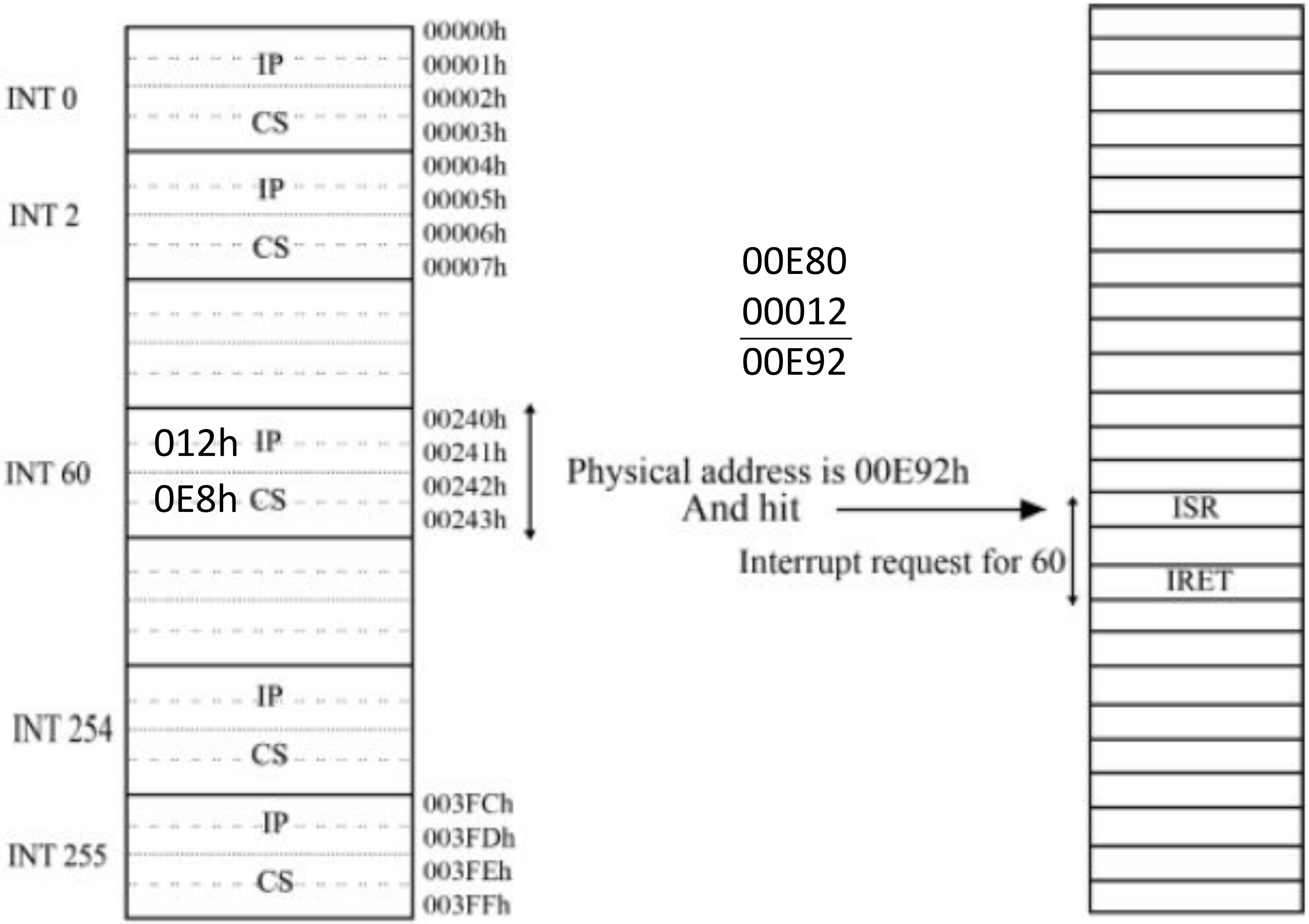
Calculation

- Now we know that the interrupt number (nn) is 60. So we can easily get the effective address from the interrupt vector table. For any interrupt number there are 4 line address first two for IP and last two for CS.
- The equation to get effective address from IP:CS is,

  IP = (nn x 4) and CS = (nn x 4) + 2

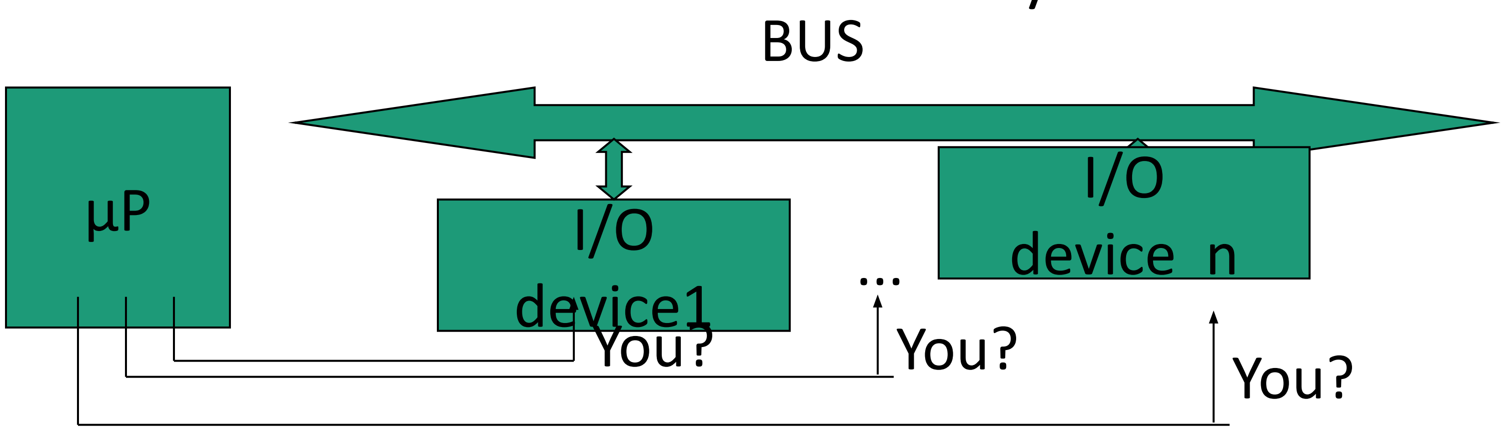
- So for 60, the RAM location of IP is (60 x 4) = 240 or 000F0h & 241 or 000F1h. And the RAM location of CS are 242 or 000F2h & 243 or 000F3h. The value of IP & CS are 012h & 0E8h respectively.
- Now from CS:IP we get the physical address where the interrupt service routine exist. Here the 20 bit physical address is 0E92h. Through this address we get the ISR and it will execute until the IRET which declare the last instruction of ISR.

Calculation



Why Interrupt is Necessary?

- **First Type:** POLLED I/O or PROGRAMMED I/O communication between MP and I/O devices.

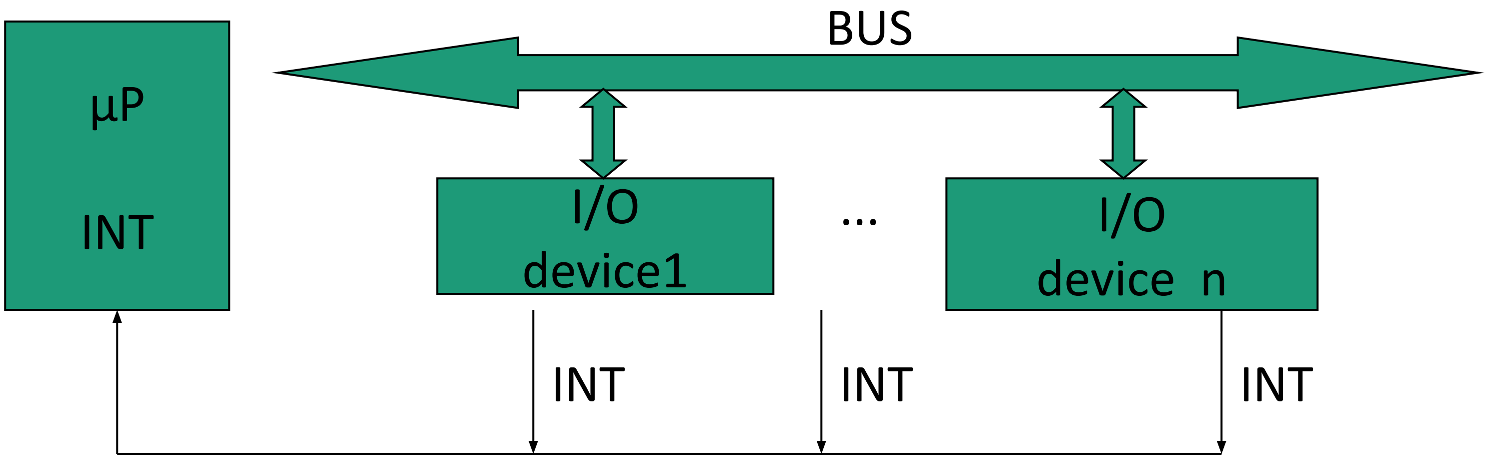


Disadvantages of Second Type Communication:

- 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*!
  - Handled by CPU
- **Interrupt**
  - fast, but more complicated
  - processor is notified by I/O device (interrupted) when device needs attention
  - Handled by Interrupt - handler
  - *similar to a telephone with bells*

Interrupt Types based on ISR ID

AVAILABLE FOR USER (224)	003FFH	TYPE 255
		...
	00080H	TYPE 32
RESERVED (27)		TYPE 31
		...
	00014H	TYPE 5
Predefined/ Dedicated/Internal Interrupts Pointers (5)	00010H	TYPE 4 — INTO OVERFLOW
	0000CH	TYPE 3 — INT
	00008H	TYPE 2 — NON-MASKABLE
	00004H	TYPE 1 — SINGLE STEP
	00000H	TYPE 0 — DIVIDE ERROR

CS Base Address
IP Offset

Divide by zero interrupt- Type 0

- It occurs automatically when the result of DIV or IDIV is too large
- Example:

DIV BL //This will do AX÷BL

//will put result in AL(quotient) and AH(reminder)

If AX was 4000H and BL was 02H,

The quotient is 2000H and reminder is 00H.

µP can put 00H in AH(reminder)

But it cannot put 2000H in AL(quotient)

This condition is called divide error!

Again by mistake, we do a division operation where divisor is 0, such as AX = 4000H and BL = 00H, then the result will be infinity and it is the largest number that can not be stored in AL or AX whatever.

This is also divide by 0 error or Type 0 interrupt.

Single Step Interrupt- Type 1

- In single step mode, a system will **stop after it executes each instructions** and wait for further direction
- If the 8086 **trap flag** is set, the 8086 will automatically do a type 1 interrupt after each instruction executes
- The trap flag is reset when the 8086 does a type 1 interrupt, so the single step mode will be disabled during the interrupt-service procedure
- Tasks for implementing single stepping:
  - Set the trap flag
  - Write an interrupt service procedure which saves all registers on the stack
  - The stack where they can later be examined
  - Load the starting address of the type 1 interrupt service procedure into address 00004H and 00006H

Non-Maskable Interrupt- Type 2

- The 8086 will automatically do a type 2 interrupt response when it receives a low to high transition on its NMI input pin
- The 8086 gets 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
- **The type 2 interrupt response cannot be disabled** by any program instruction that's why we use it to signal the 8086 that some condition in an external system must be taken care of
- **The type 2 interrupt is useful to save program data in case of a system power failure**

Breakpoint Interrupt- Type 3

- The main use of type 3 interrupt is **to implement a breakpoint function** in a system
- Unlike the single step feature, which stops execution after each instruction, the breakpoint feature **executes all the instruction up to the inserted breakpoint** and then stops execution
- The 8086 gets the CS value for the start of the type 3 interrupt service procedure from address 0000EH and the IP value for the start of the procedure from address 0000CH
- **It is useful in debugging large programs when single stepping is inefficient**

Overflow Interrupt- Type 4

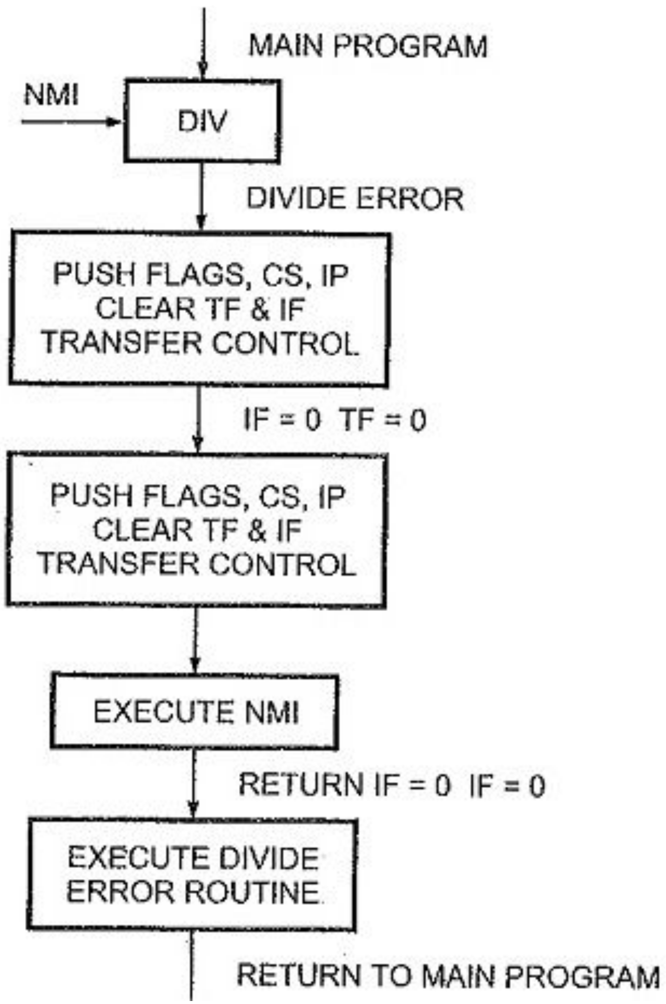
- The 8086 **overflow flag (OF) will be set if the signed result of an arithmetic operation on two signed numbers is too large to be represented or memory location**
- The 8086 will get the CS value for the start of the type 4 interrupt service procedure from address 00012H and the IP value for the start of the procedure from address 00010H
- It is **useful to detect overflow error** in signed arithmetic operations
- There are **two ways to detect and respond** to an overflow error in a program
  - **Put the jump if overflow instruction (JO)** immediately after the arithmetic instruction
  - **Put the interrupt on overflow instruction (INTO)** immediately after the arithmetic instruction

Summary of 8086 Interrupt Function …

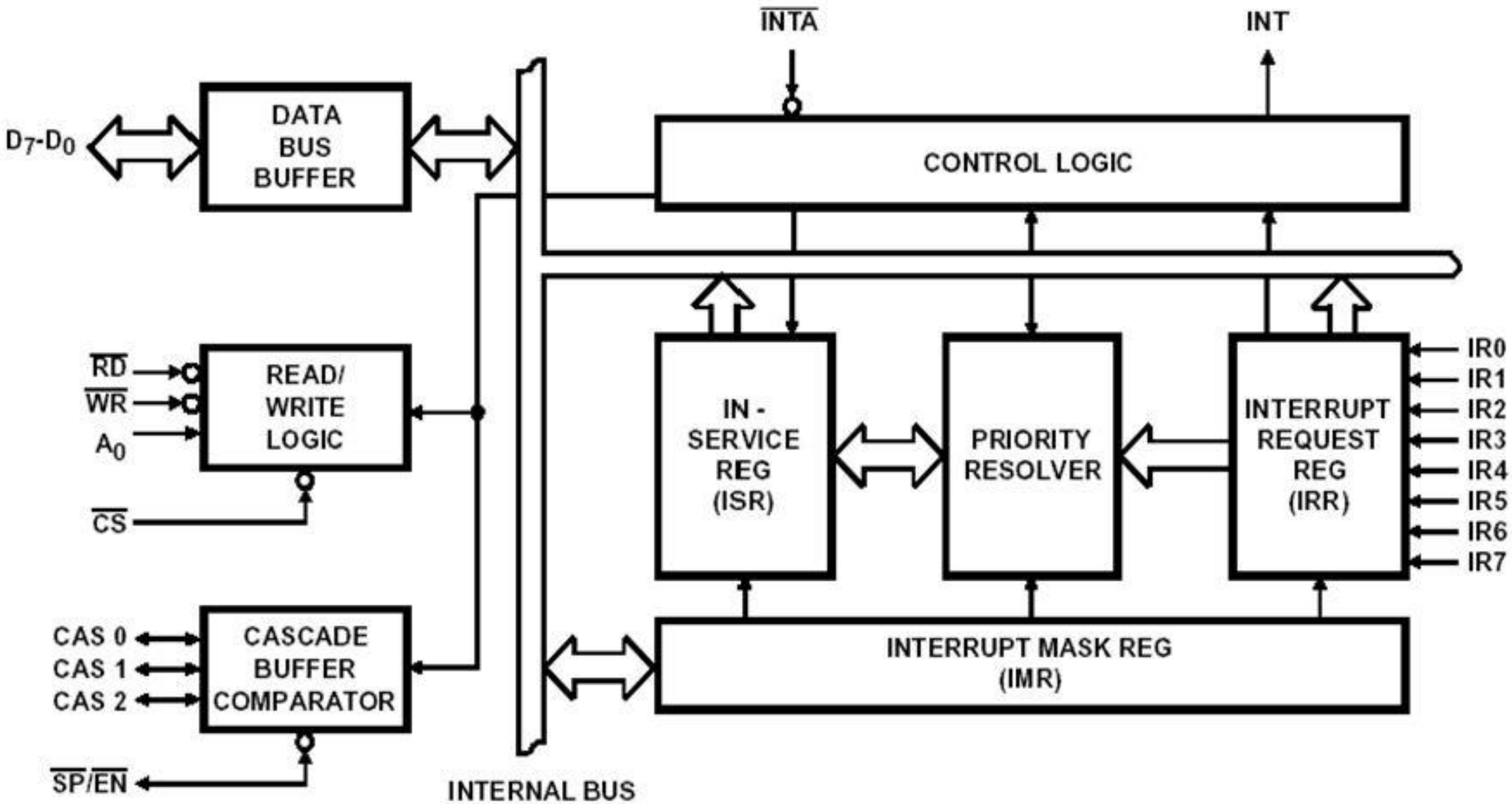
- **What happens if two or more interrupts occur at the same time?**
  - Higher priority interrupts will be served first

Interrupt Type	Priority
DIVIDE ERROR, INTn, INT0	HIGHEST
NMI	
INTR	
SINGLE STEP	LOWEST

Flow Chart for Divide Error Routine



8259A
Programmable Interrupt Controller (PIC)

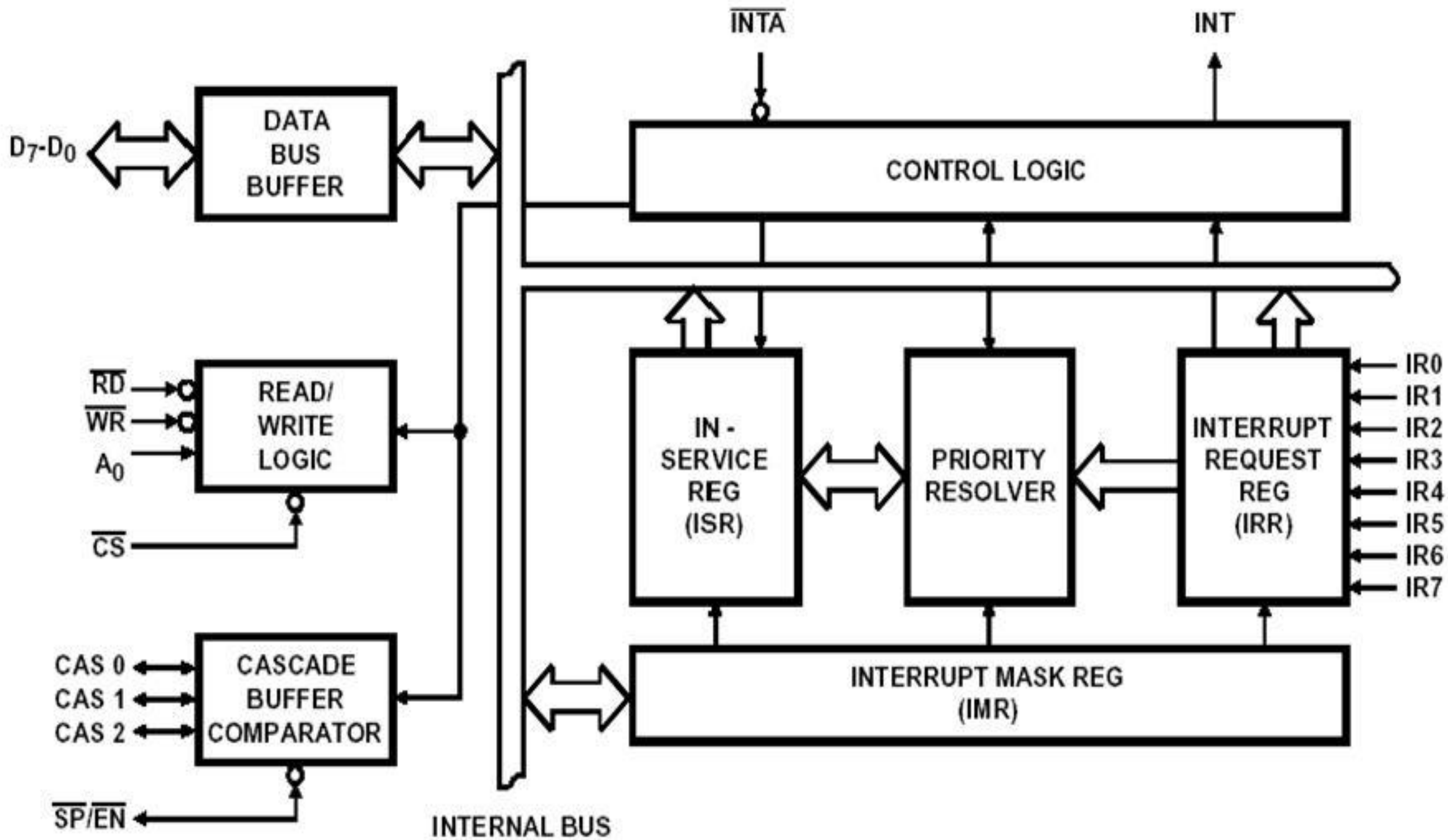


How 8086 INTR input works

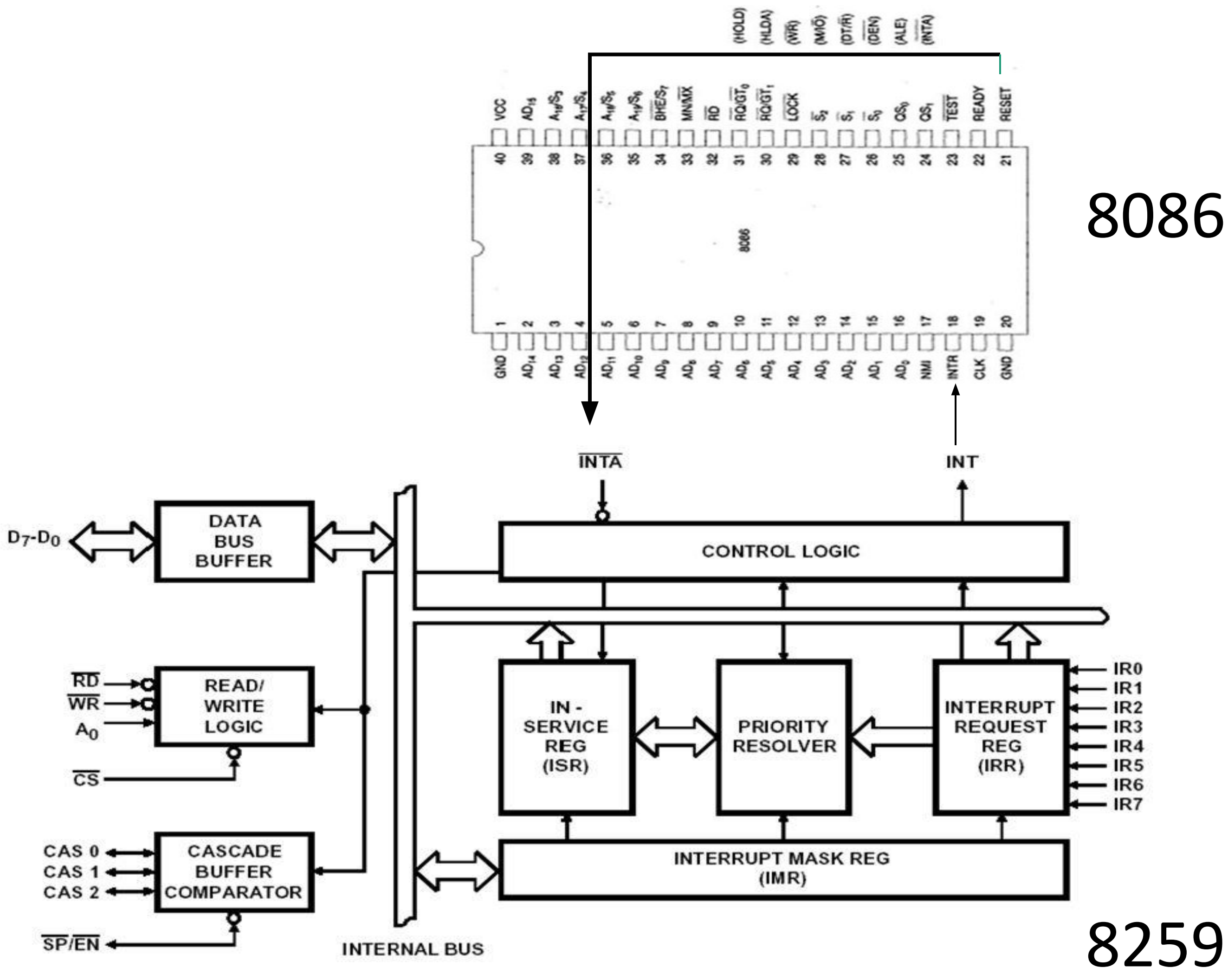
If the 8086 Interrupt flag is set and the INTR input receives a high signal, the 8086 will:

1. Send out two interrupt acknowledgement pulses on its INTA pin to the INTA pin of 8259 PIC. The INTA pulses tell the 8259A to send the desired interrupt type to the 8086 on the data bus.
2. Multiply the interrupt type it receives from the 8259A by 4 to produce an address in the Interrupt Vector Table. INT60 means 60 X 4 = 240h
3. Push the flags on the stack
4. Clear IF and TF
5. Push the return address on the stack.
6. Get the starting address for the interrupt procedure from the IVT and load thar address in CS and IP.
7. Execute the interrupt service procedure.

8259A Internal Block Diagram



Connection with 8086



Registers

- IRR:

0	0	0	0	0	0	0	0
IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0

- ISR:

0	0	0	0	0	0	0	0
ISR7	ISR6	ISR5	ISR4	ISR3	ISR2	ISR1	ISR0

- IMR:

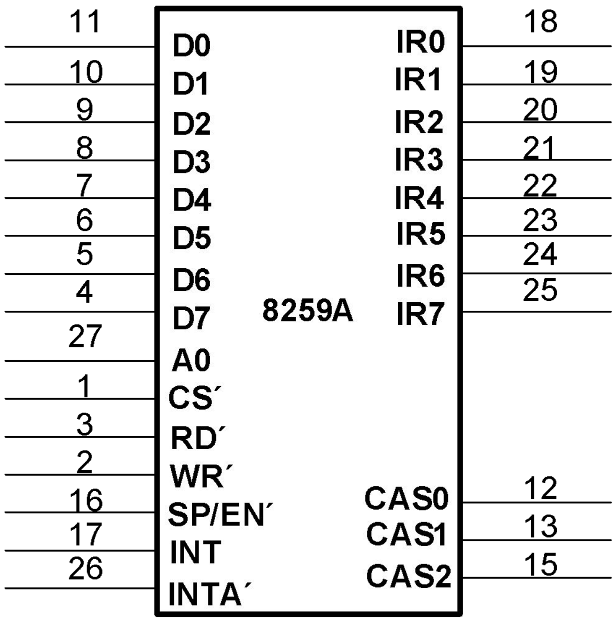
0	0	0	0	0	0	0	0
IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0

The 8259A System Connections

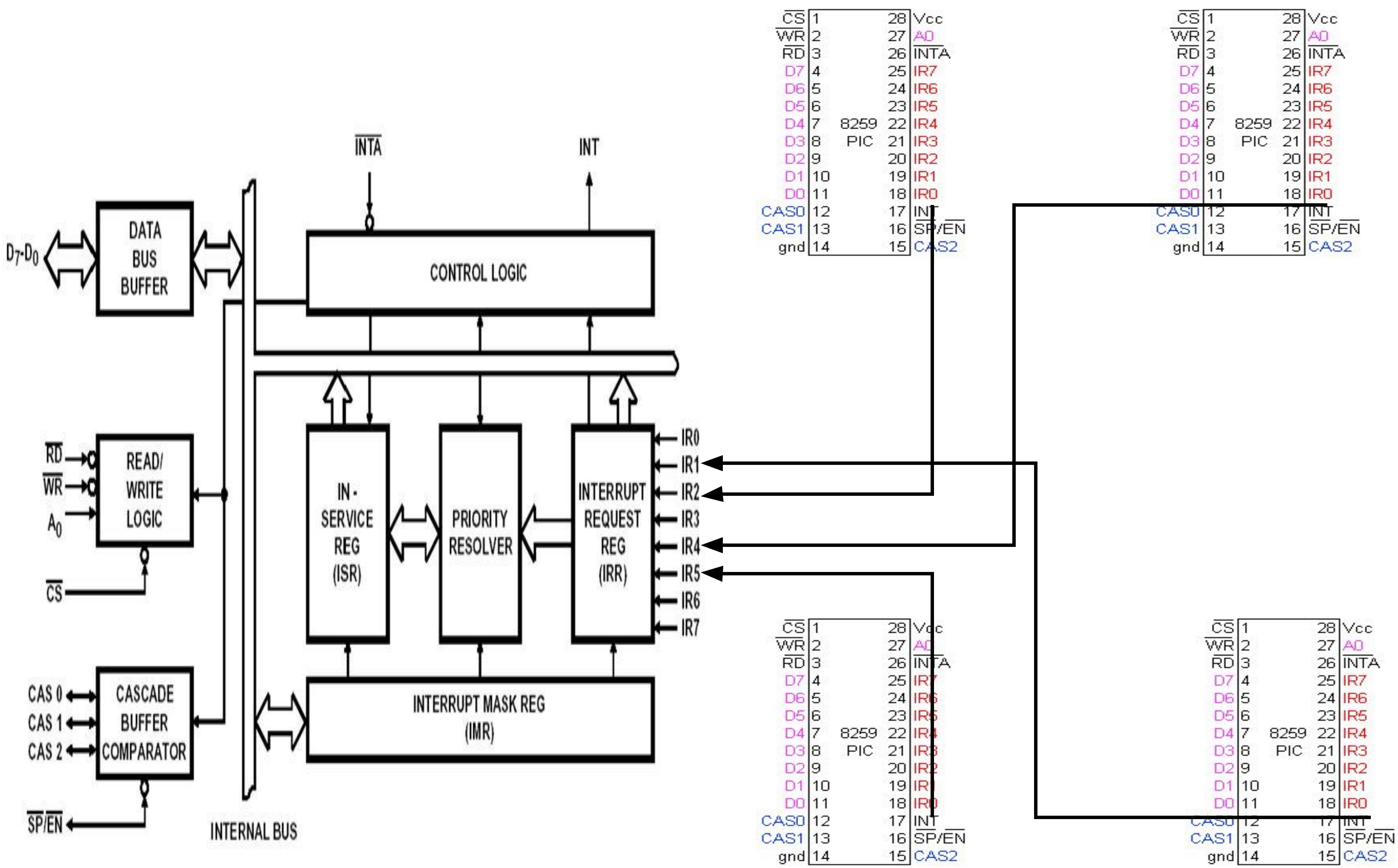
- The 8259A is used to increase the number of interrupts
- The 8 bit data bus allows 8086 to send control words to the 8259A and read a status word from the 8259A and also allows to send interrupt types to the 8086
- The RD’ and WR’ inputs control the transfer
- When an interrupt occurs corresponding bit becomes 1 in (IR0-IR7), it checks IRR to know which interrupt has occurred, it checks IMR to know which interrupts are masked and it checks ISR to know which interrupts are in service
- If the interrupt occurs is in higher priority, not masked and its in higher priority than the interrupt which was in service then its validate and it will be send to the µp on INTR

The 8259A Priority Interrupt Controller

- Adds 8 vectored priority encoded interrupts to the microprocessor
- Can be expanded without additional hardware to accept up to 64 IRQ (master and slave PIC's)
- D0-D7: Bidirectional data connections
- IR0-IR7: Interrupt request inputs
- WR´: Write input strobe
- RD´: Read input connects to the IORC
- INT: Output, connects to µP INTR pin
- INTA´: Input, connects to µP INTA´ pin
- A0: Command word select
- CS´: Chip select input
- SP/EN´: Slave program/enable buffer pin
- CAS0-CAS2: Outputs from master to slave for cascading multiple 8259A chips



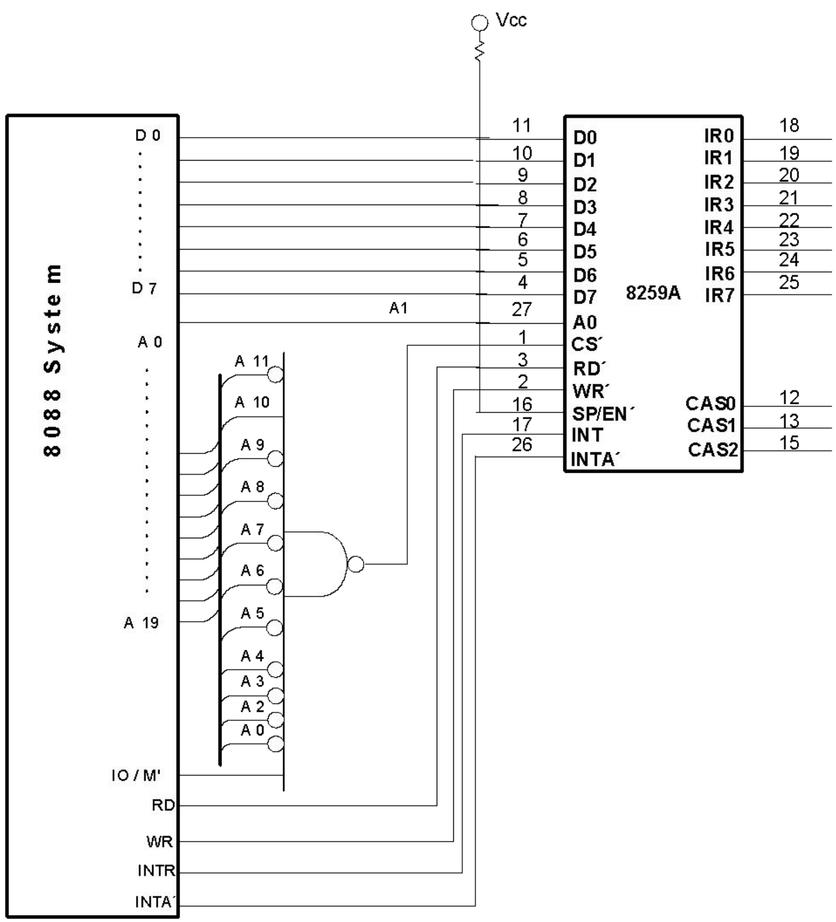
Cascading with other 8259



Cascading with other 8259

CAS2	CAS1	CAS0	PIC
0	0	0	PIC1
0	0	1	PIC2
0	1	0	PIC3
0	1	1	PIC4
1	0	0	PIC5
1	0	1	PIC6
1	1	0	PIC7
1	1	1	PIC8

Connecting a single 8259A controller



Thank You !!

The programmable peripheral Interface

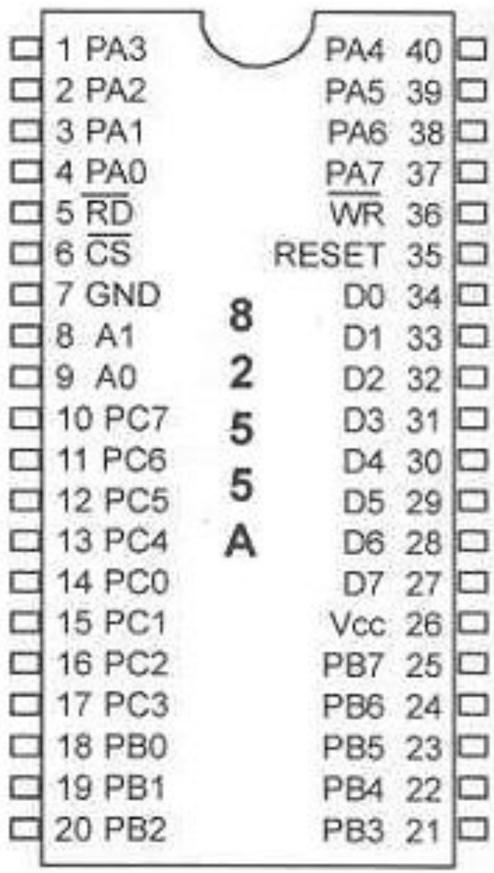
The 82C55 **programmable peripheral interface (PPI)** is a very popular low-cost interfacing component that is used found in many applications.

Applications range from 7-segment display, stepper motor connection, counters to key pad management.

This device is still in use today in P4 Computers and is used to interface and detect key presses on modern keyboards, parallel printers and other interfacing chipsets.

For those of you who are doing computer interfacing course you will be extensively using this device to interface various devices with the PC.

82C55 Programmable Peripheral Interface

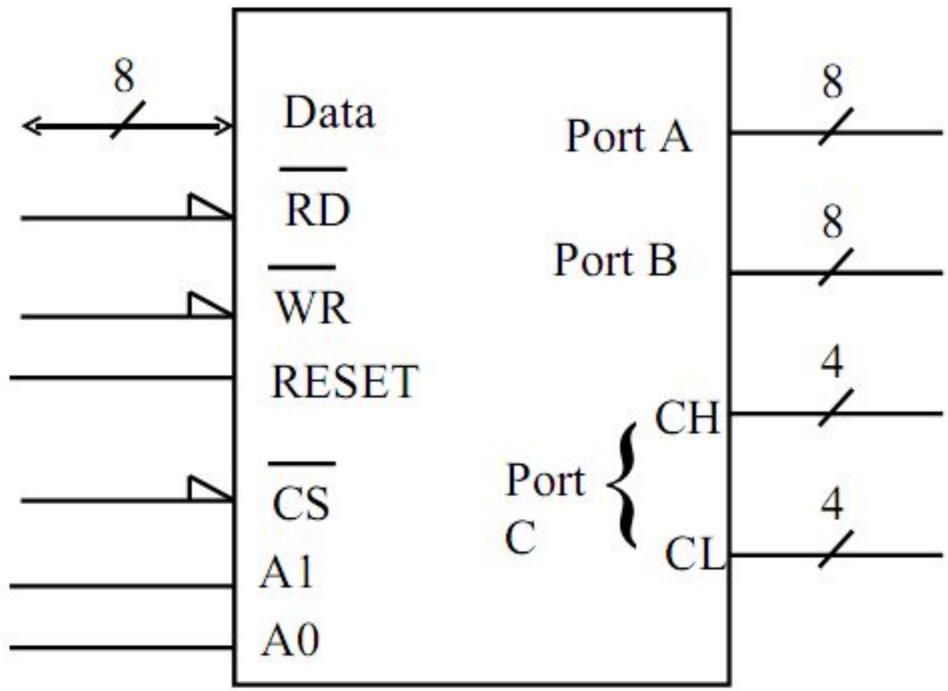


The 8255 Programmable Peripheral Interface (PPI)

- It is one of the most widely used I/O chips.
- It has three separately accessible ports: A, B, and C
- The individual ports can be programmed to be input or output.
- Port A (PA0-PA7) -> all inputs or all outputs.
- Port B (PB0-PB7) -> all inputs or all outputs.
- Port C (PC0-PC7) -> all inputs or all outputs or can be programmed individually.
  - CU : upper bits (PC4-PC7)
  - CL : lower bits (PC0-PC3)

8255 PPI

(Programmable Peripheral Interface)
(Parallel Port Interface)



82C55 Features

- It consists of 3 ports for I/O.
- Each port is 8-bit. So total 24 pins.
- The I/O pins can be programmed in groups of 12 pins.
- Address A0 and A1 (these are input of 8255 not 8086) are used to select the port to read/write.
- Data are transferred through a 8-bit bidirectional data bus.
- Chip select ($\overline{CS}$)of the 8255 must be enabled.
- There are three distinct modes of operation Mode 0, Mode 1 and Mode 2.
- Group A connections consists of Port A(PA0-PA7) and the upper half of port C (PC4-PC7).
- Group B connections consists of Port B (PB0-PB7) and the lower half of port C (PC0-PC3).

82C55 port selection table

CS’	A1	A0	SELECTI ON	ADDRES S
0	0	0	PORT A	80 H
0	0	1	PORT B	81 H
0	1	0	PORT C	82 H
0	1	1	Control Register	83 H
1	X	X	No Selection	X

CS pin is used to select the device for reading or writing.

A0 and A1 select the specific port with 82C55

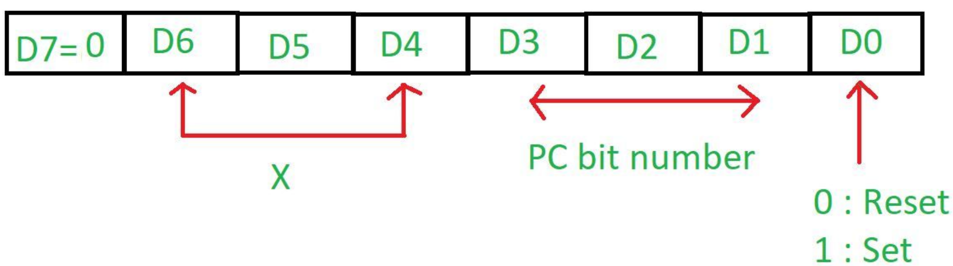
Following Table gives the basic operation,

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Input operation
0	0	0	1	0	PORT A $\longrightarrow$ Data bus
0	1	0	1	0	PORT B $\longrightarrow$ Data bus
1	0	0	1	0	PORT C $\longrightarrow$ Data bus
					<u>Output operation</u>
0	0	1	0	0	Data bus $\longrightarrow$ PORT A
0	1	1	0	0	Data bus $\longrightarrow$ PORT B
1	0	1	0	0	Data bus $\longrightarrow$ PORT C
1	1	1	0	0	Data bus $\longrightarrow$ control

82C55 operation modes

There are 2 modes in 8255 microprocessor:

- **Bit set reset (BSR) mode –** This mode is used to set or reset the bits of port C only, and selected when the most significant bit (D7) in the control register is 0. Control Register is as follows:



82C55 operation modes(Contd.)

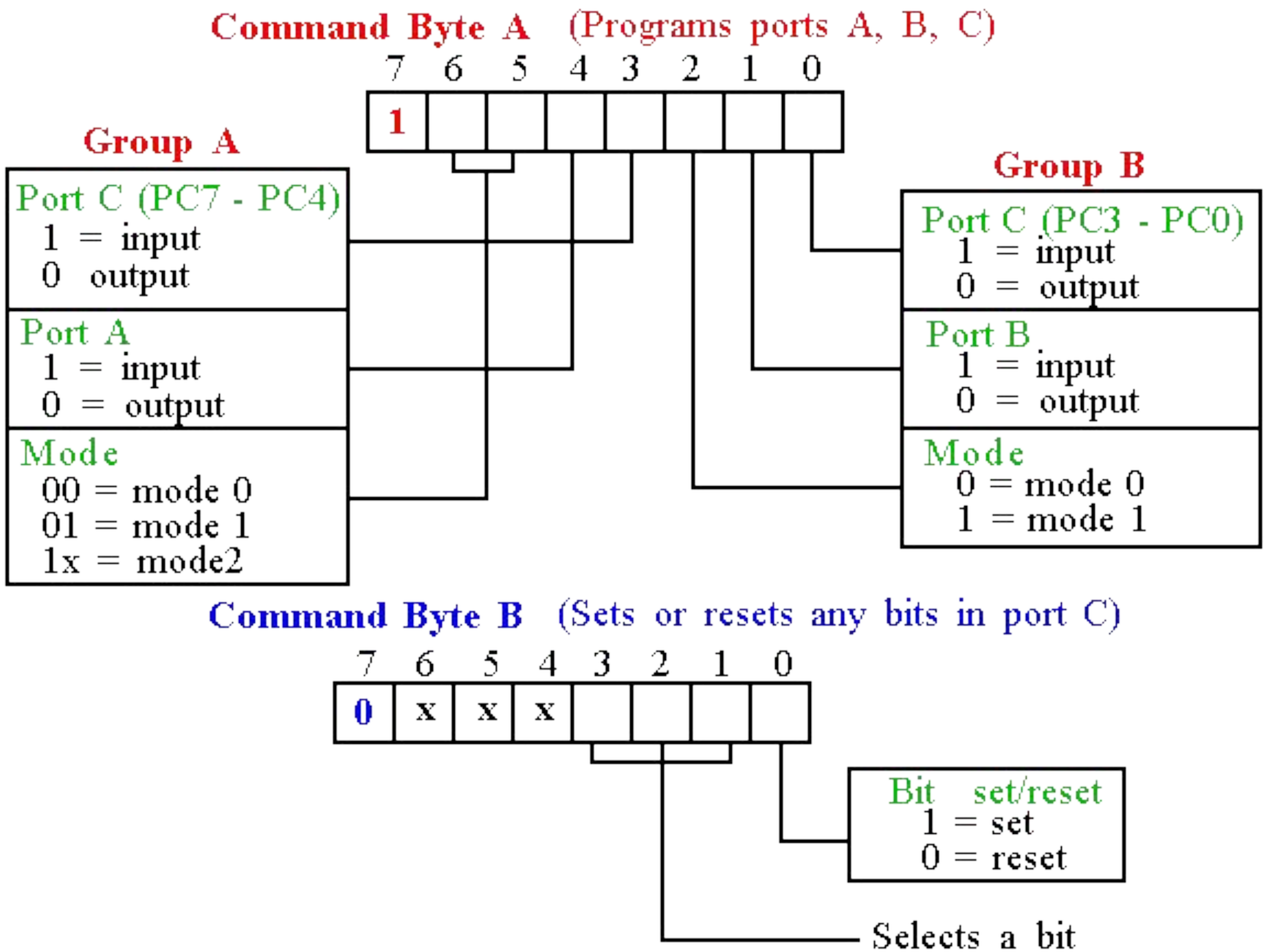
- **Input/output mode (I/O) –** This mode is selected when the most significant bit (D7) in the control register is 1.
  - **Mode 0 – Simple or basic I/O mode**
  - **Mode 1 – Handshake or strobbed I/O**
  - **Mode 2 – Bidirectional I/O**

82C55 operation modes(Contd.)

- **Mode 0 –**In this mode all the three ports (port A, B, C) can work as simple input function or simple output function.
- **Mode 1 –** Handshake I/O mode or strobbed I/O mode. In this mode either port A or port B can work as simple input port or simple output port, and port C bits are used for handshake signals before actual data transmission.
  - Example: A CPU wants to transfer data to a printer. In this case since speed of processor is very fast as compared to relatively slow printer, so before actual data transfer it will send handshake signals to the printer for synchronization of the speed of the CPU and the peripherals.
- **Mode 2:** Bi-directional data bus mode. In this mode only port A works, and port B can work either in mode 0 or mode 1. 6 bits port C are used as handshake signals. It also has interrupt handling capacity.

Programming 8255

- 8255 has three operation modes: *mode 0, mode 1, and mode 2*



Example - 82C55 Keyboard Interrupt Circuit