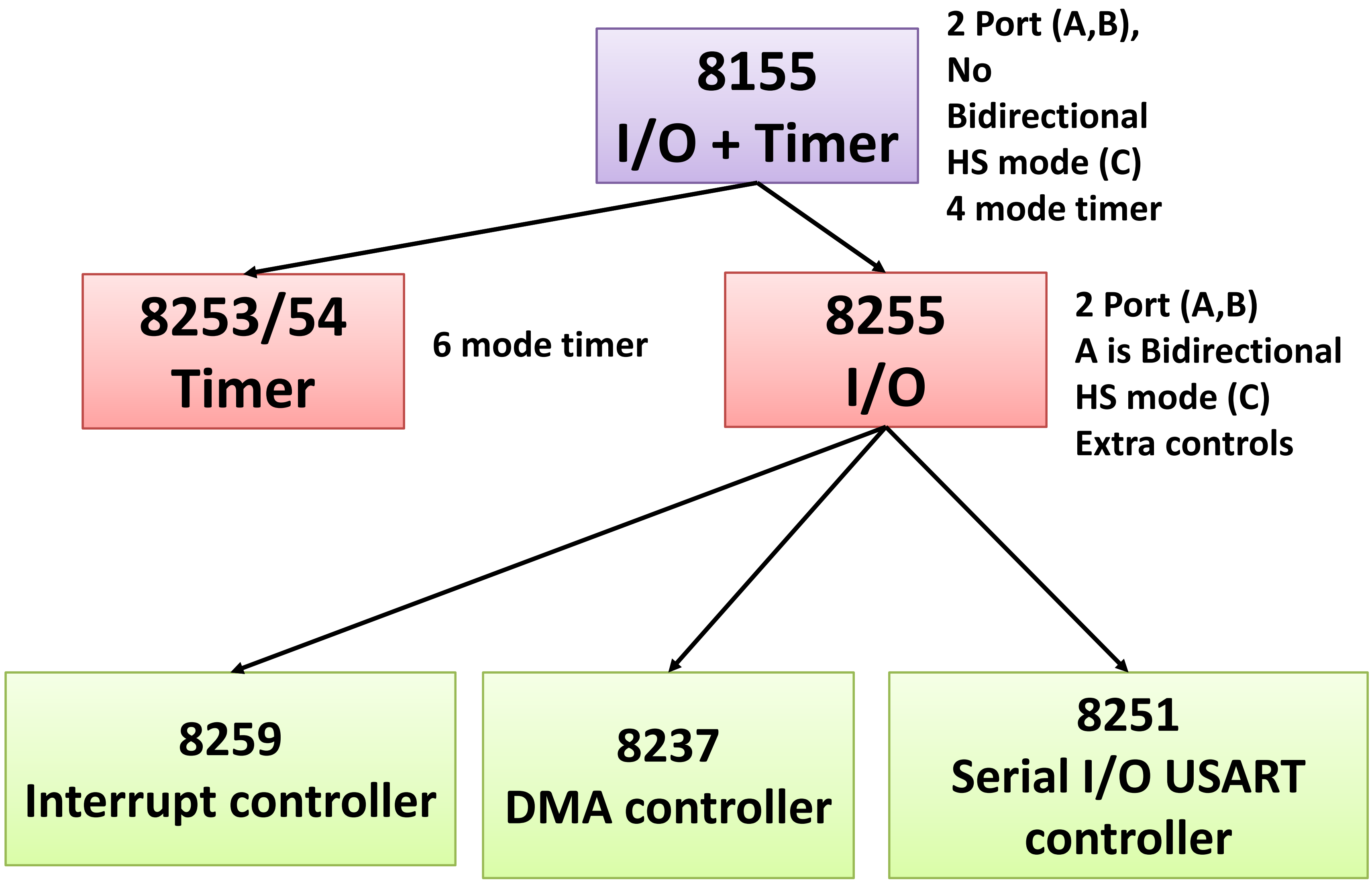
Interrupt Controller (Introduction to 8259)

Dr A Sahu
Dept of Comp Sc & Engg.
IIT Guwahati

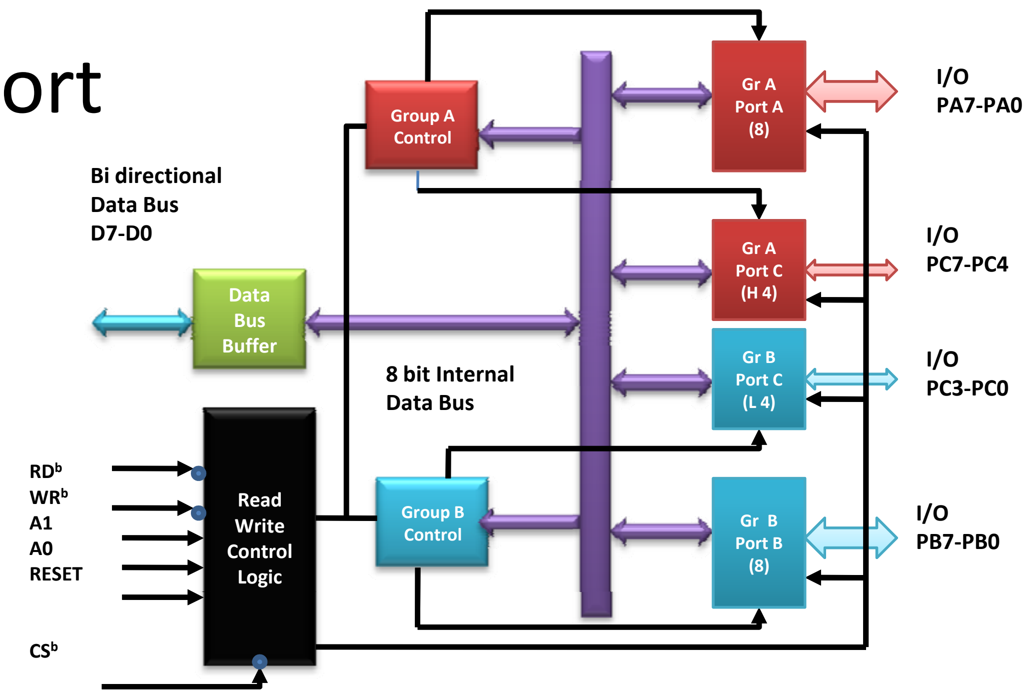
Hierarchy of I/O Control Devices



How many I/O can be connected using 8255

- Two Port A & Port B
- Port C as HS/INT port
- Both can work

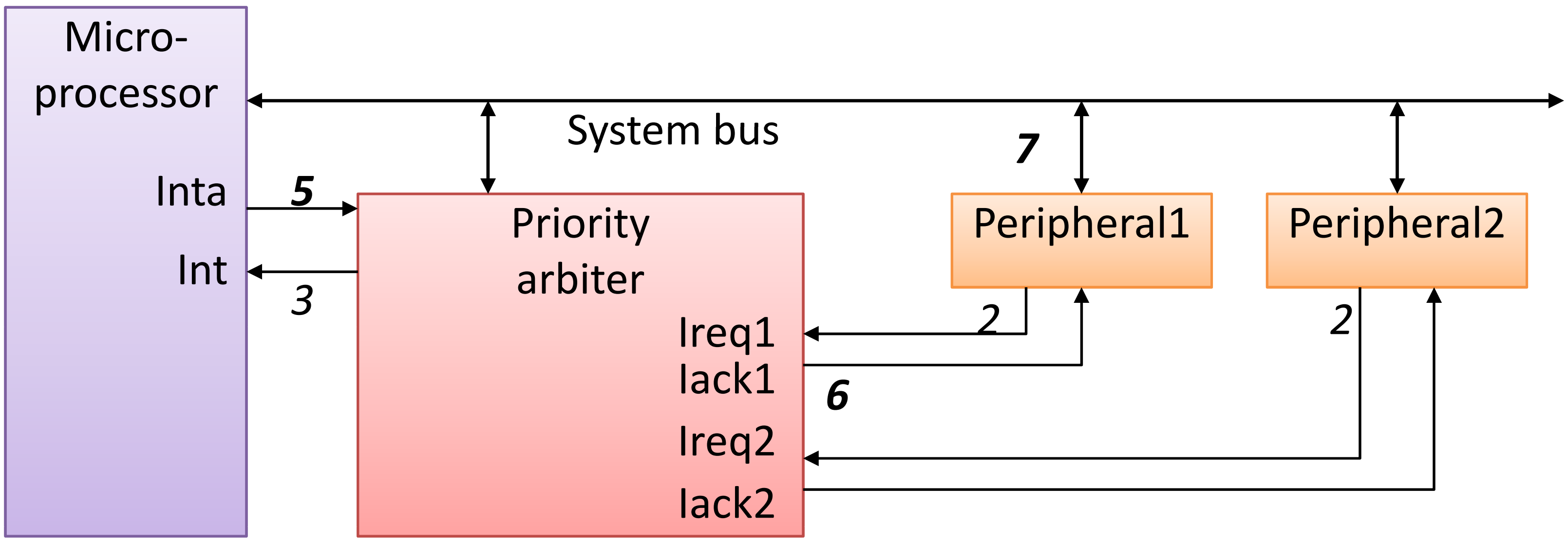
  Simultaneously



How to connect multiple I/O Device

- Use Interrupt as generalized mechanism to connect
- Use priority resolver to connect them

Arbitration using a priority arbiter



Peripheral1 needs servicing so asserts *Ireq1*. Peripheral2 also needs servicing so asserts *Ireq2*.

Priority arbiter sees at least one *Ireq* input asserted, so asserts *Int*.

Microprocessor stops executing its program and stores its state.

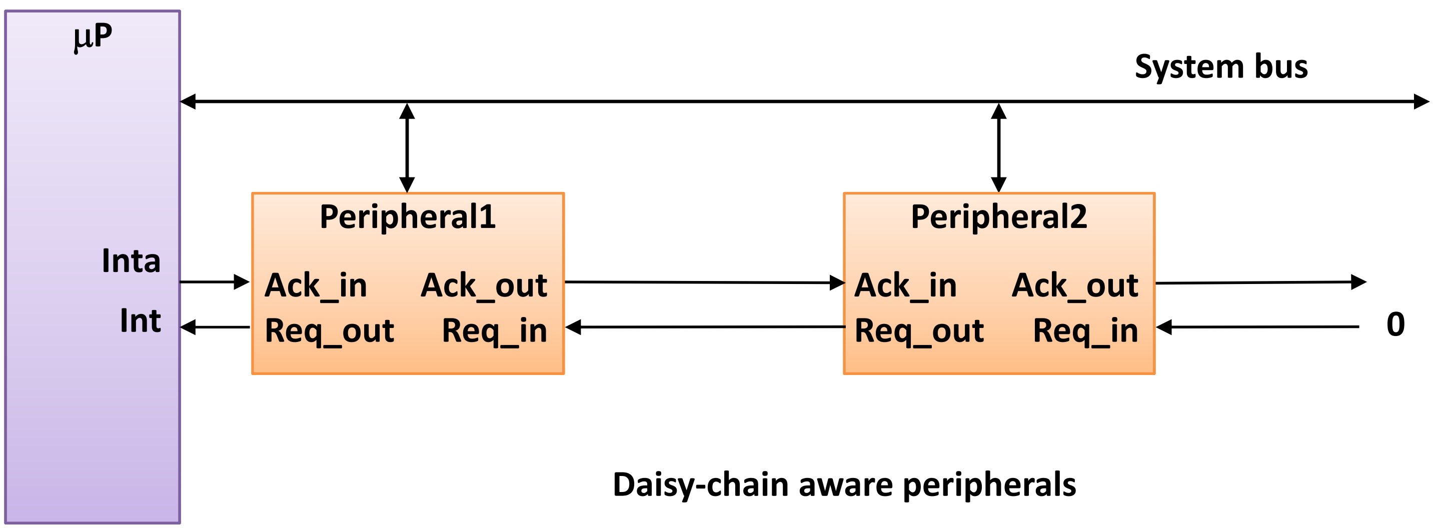
Microprocessor asserts *Inta*. Priority arbiter asserts *Iack1* to acknowledge Peripheral1.

Peripheral1 puts its interrupt address vector on the system bus

Microprocessor jumps to the address of ISR read from data bus, ISR executes and returns

Arbitration: Daisy-chain arbitration

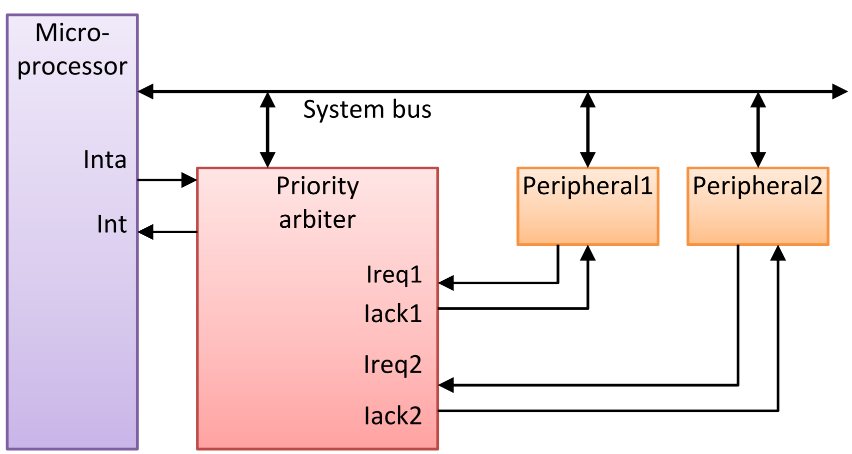
- Arbitration done by peripherals
  - Built into peripheral or external logic added
    - *req* input and *ack* output added to each peripheral
- Peripherals connected to each other in daisy-chain manner
  - One peripheral connected to resource, all others connected "upstream"
  - Peripheral's *req* flows "downstream" to resource, resource's *ack* flows "upstream" to requesting peripheral
  - Closest peripheral has highest priority

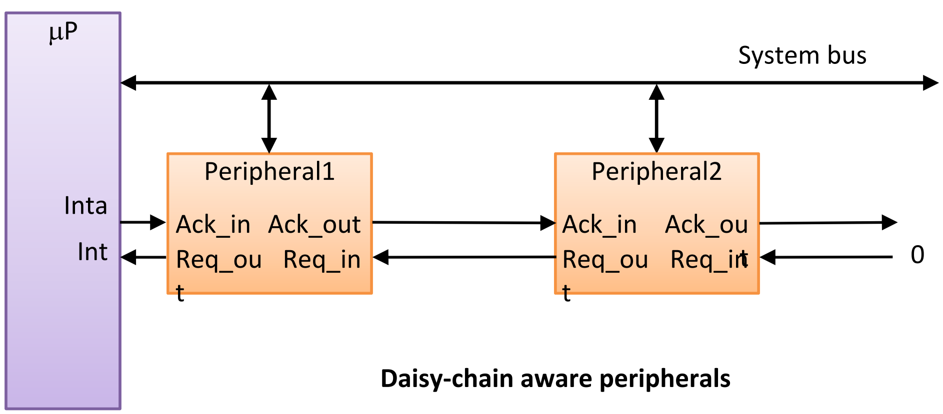


Daisy-chain aware peripherals

Arbitration: Daisy-chain arbitration

- Pros/cons
  - Easy to add/remove peripheral - no system redesign needed
  - Does not support rotating priority
  - One broken peripheral can cause loss of access to other peripherals



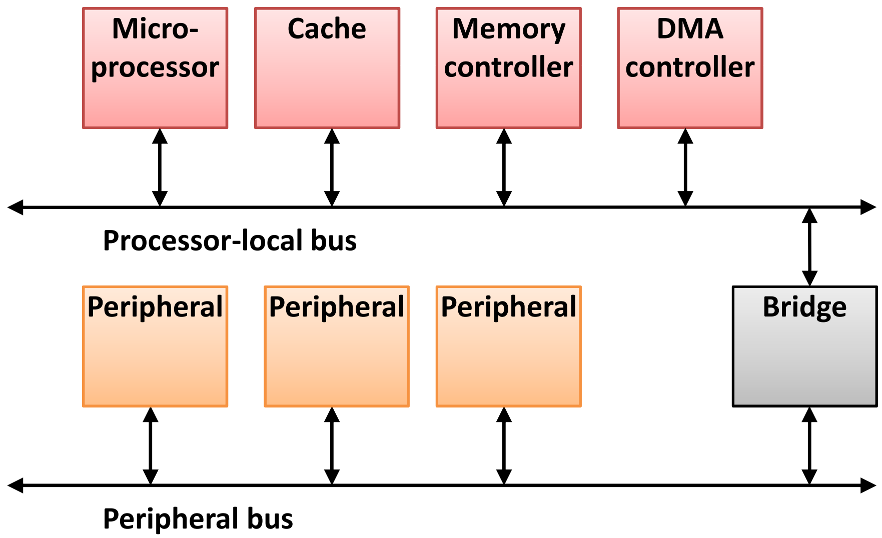


Daisy-chain aware peripherals

Multilevel bus architectures

- Don't want one bus for all communication
  - Peripherals would need high-speed, processor-specific bus interface
  - Too many peripherals slows down bus

- Processor-local bus
  - High speed, wide, most frequent communication
  - Connects microprocessor, cache, memory controllers, etc.
- Peripheral bus
  - Lower speed, narrower, less frequent communication
  - Typically industry standard bus (ISA, PCI) for portability



- Bridge
  - Single-purpose processor converts communication between busses

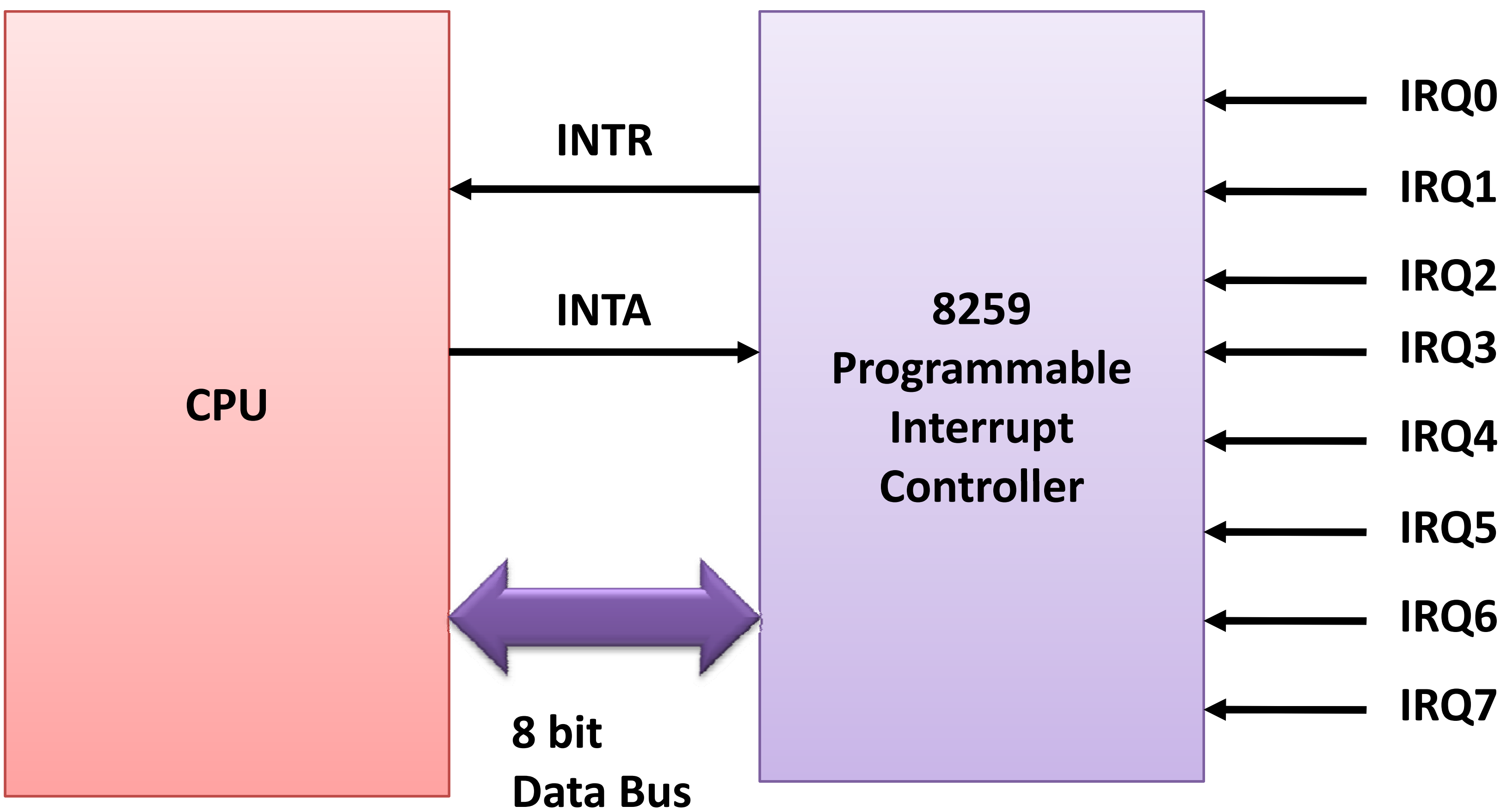
What is Interrupt

- Interrupts alter a program's flow of control
  - Behavior is similar to a procedure call
    - Some significant differences between the two
- Interrupt causes transfer of control to an *interrupt service routine* (ISR)
    - ISR is also called a *handler*
- When the ISR is completed, the original program resumes execution
- **Interrupts are used to interface I/Os**
- Interrupts provide an efficient way to handle unanticipated events

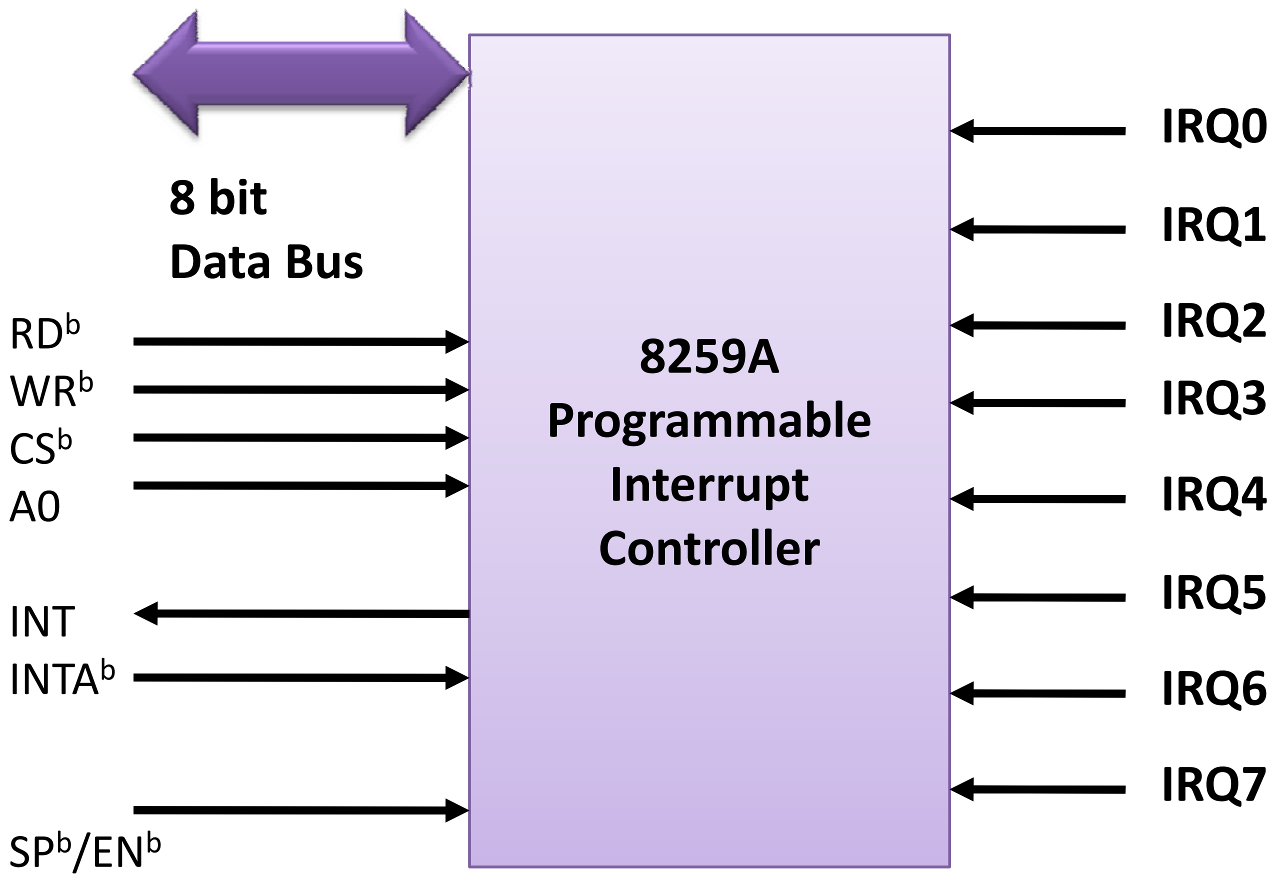
8259 Programmable Interrupt Controller (PIC)

- It is a tool for managing the interrupt requests.
- 8259 is a very flexible peripheral controller chip:
  - PIC can deal with up to 64 interrupt inputs
  - interrupts can be masked
  - various priority schemes can also programmed.
- originally (in PC XT) it is available as a separate IC
- Later the functionality of *(two PICs)* is in the motherboards chipset.
- In some of the modern processors, the functionality of the *PIC* is built in.

8259: Programmable Interrupt Controller



Block Diagram of 8259



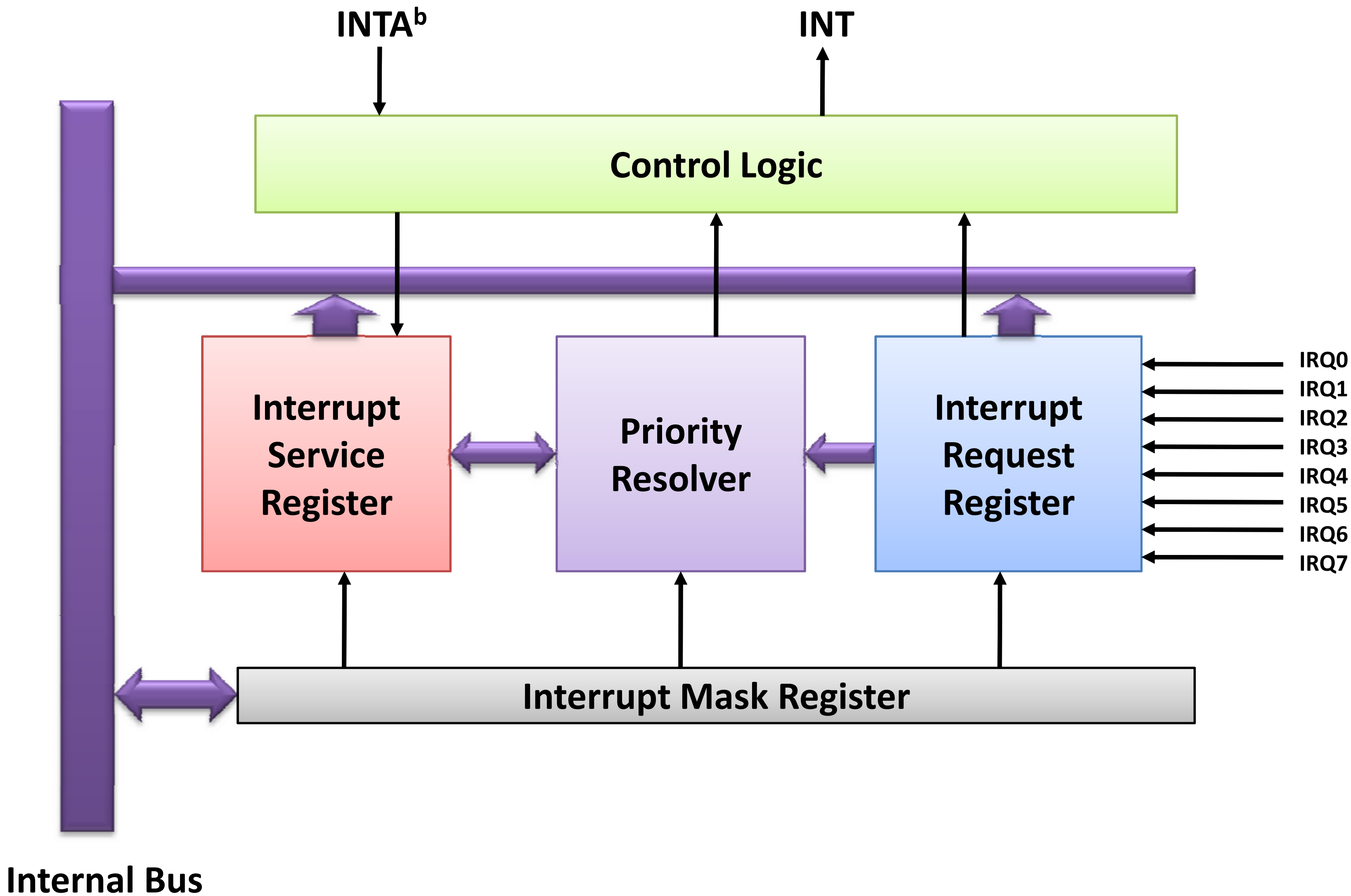
Pin description

- 8-bit bi-directional data bus, one address line is needed,

  PIC has two control registers to be programmed, you can think of them as two output ports or two memory location.

- The direction of data flow is controlled by RD and WR.
- CS is as usual connected to the output of the address decoder.
- Interrupt requests are output on INT which is connected to the INTR of the processor. Int. acknowledgment is received by INTA.
- IR0-IR7 allow 8 separate interrupt requests to be inputted to the PIC.
- sp/en=1 for master , sp/en=0 for slave.
- CAS0-3 inputs/outputs are used when more than one PIC to cascaded.

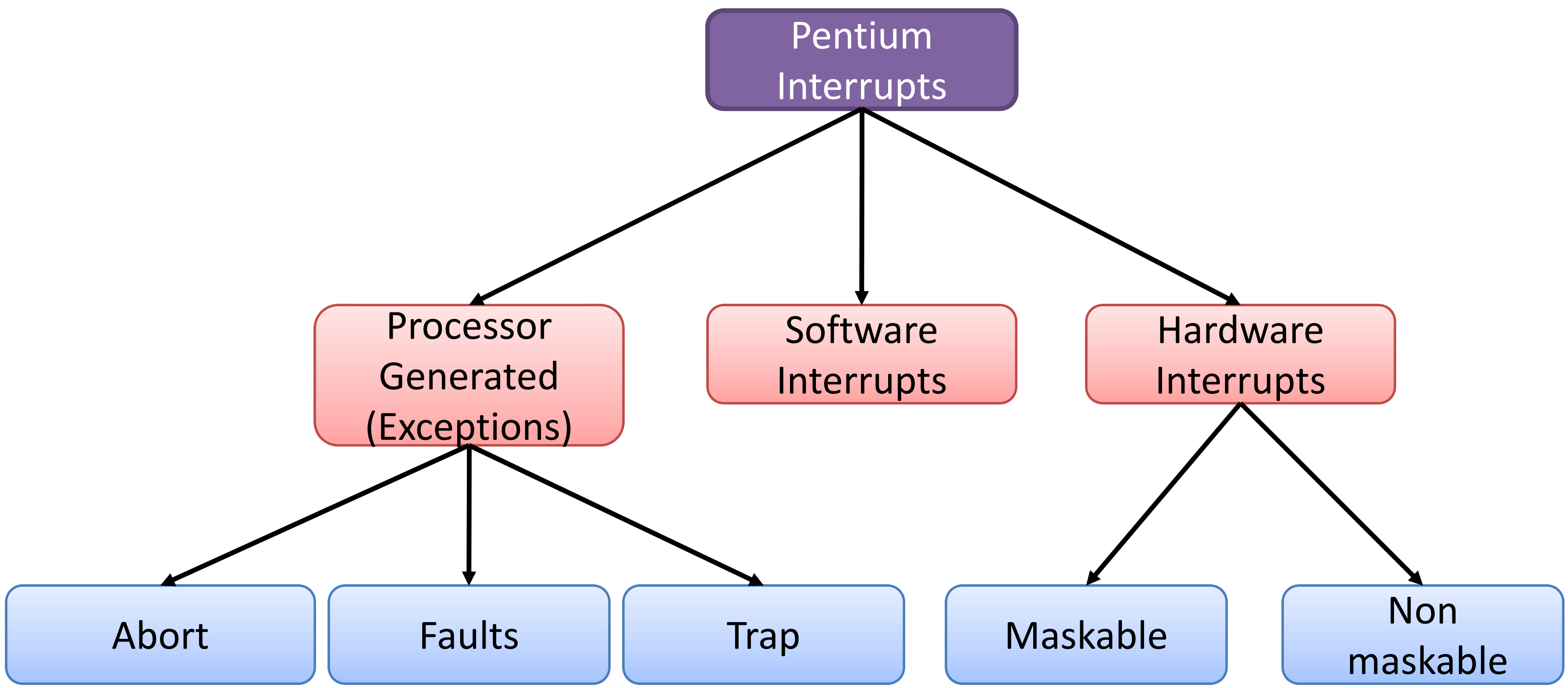
Block Diagram Architecture of 8259



8259 Programmable Interrupt Controller

- 8259 can service up to eight hardware devices
  - Interrupts are received on IRQ0 through IRQ7
- 8259 can be programmed to assign priorities in several ways
  - Fixed priority scheme is used in the PC
    - IRQ0 has the highest priority and IRQ7 lowest
- 8259 has two registers
  - Interrupt Command Register (ICR)
    - Used to program 8259
  - Interrupt Mask Register (IMR)

A Taxonomy of Pentium Interrupts



Exceptions

- Exception Classification (processor-generated)
  - Fault
    - *Return to the faulting instruction*
    - Reported during the execution of the faulting instruction
    - Virtual memory faults
      - TLB miss, page fault, protection
    - Illegal operations
      - divide by zero, invalid opcode, misaligned access
  - Trap
    - *Return to the next instruction* (after the trapping instruction)
      - For a JMP instruction, the next instruction should point to the target of the JMP instruction
    - Reported immediately following the execution of the trapping instruction
    - Examples: breakpoint, debug, overflow

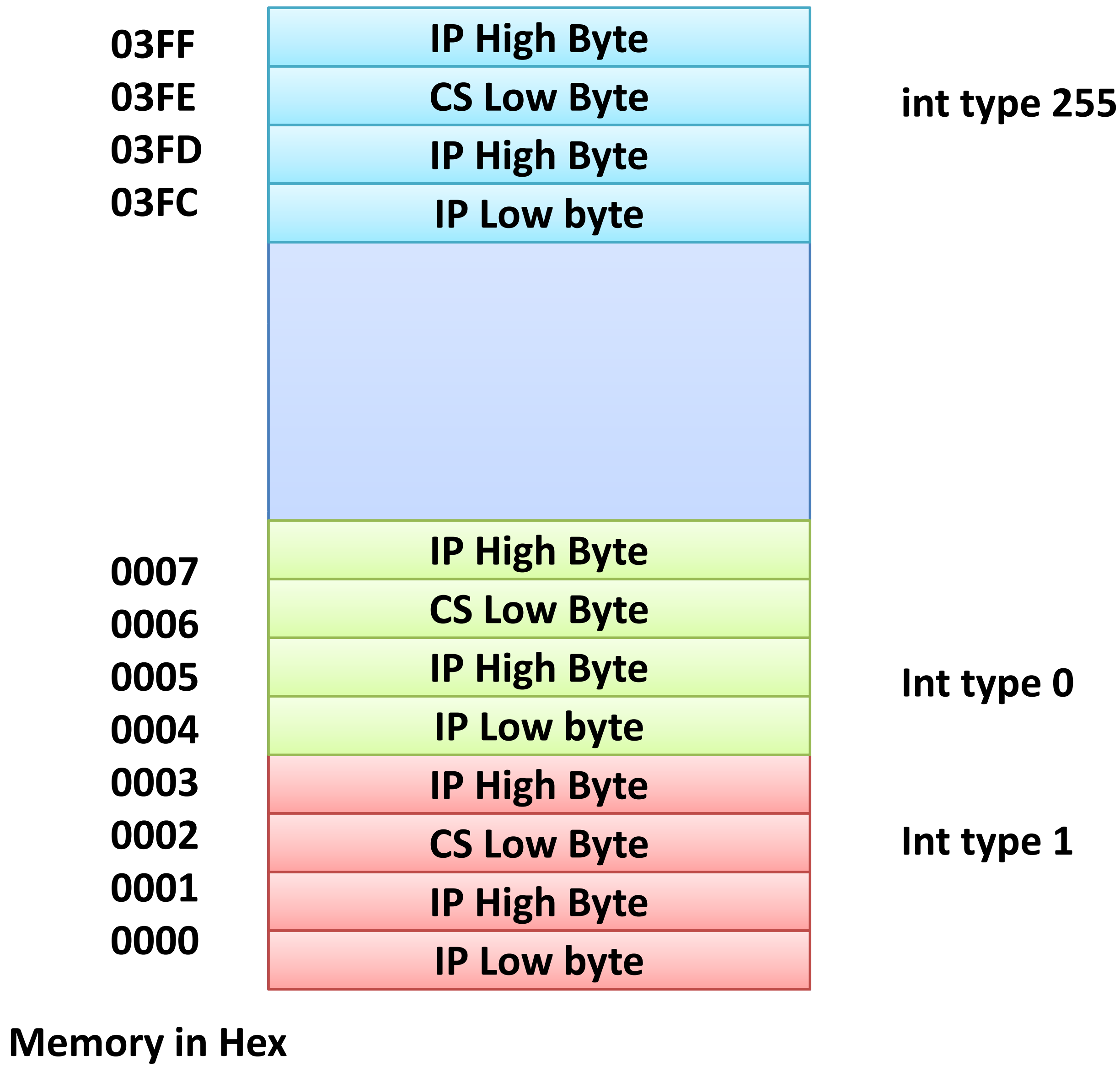
Exceptions

- Abort
  - Suspend the process at an unpredictable location
    - Does not report the precise location of the instruction causing the exception
    - Does not allow restart of the program
  - Severe errors or malfunctions
  - *Abort handlers* are designed to *collect diagnostic* information about the processor's state and then perform a *graceful system shutdown*
  - Examples: bit error (parity error), inconsistent or illegal values in system tables
- Software-generated exception
  - *INT n* instruction generates an exception with an exception number (n) as an operand

Interrupt Processing in Real Mode

- Uses an interrupt vector table that stores pointers to the associated interrupt handlers.
  - This table is located at base address zero.
- Each entry in this table consists of a CS:IP pointer to the associated ISRs
  - Each entry or vector requires four bytes:
    - Two bytes for specifying CS
    - Two bytes for the offset
- In PC: up to 256 interrupts are supported (0 to 255).

Interrupt Vector Table



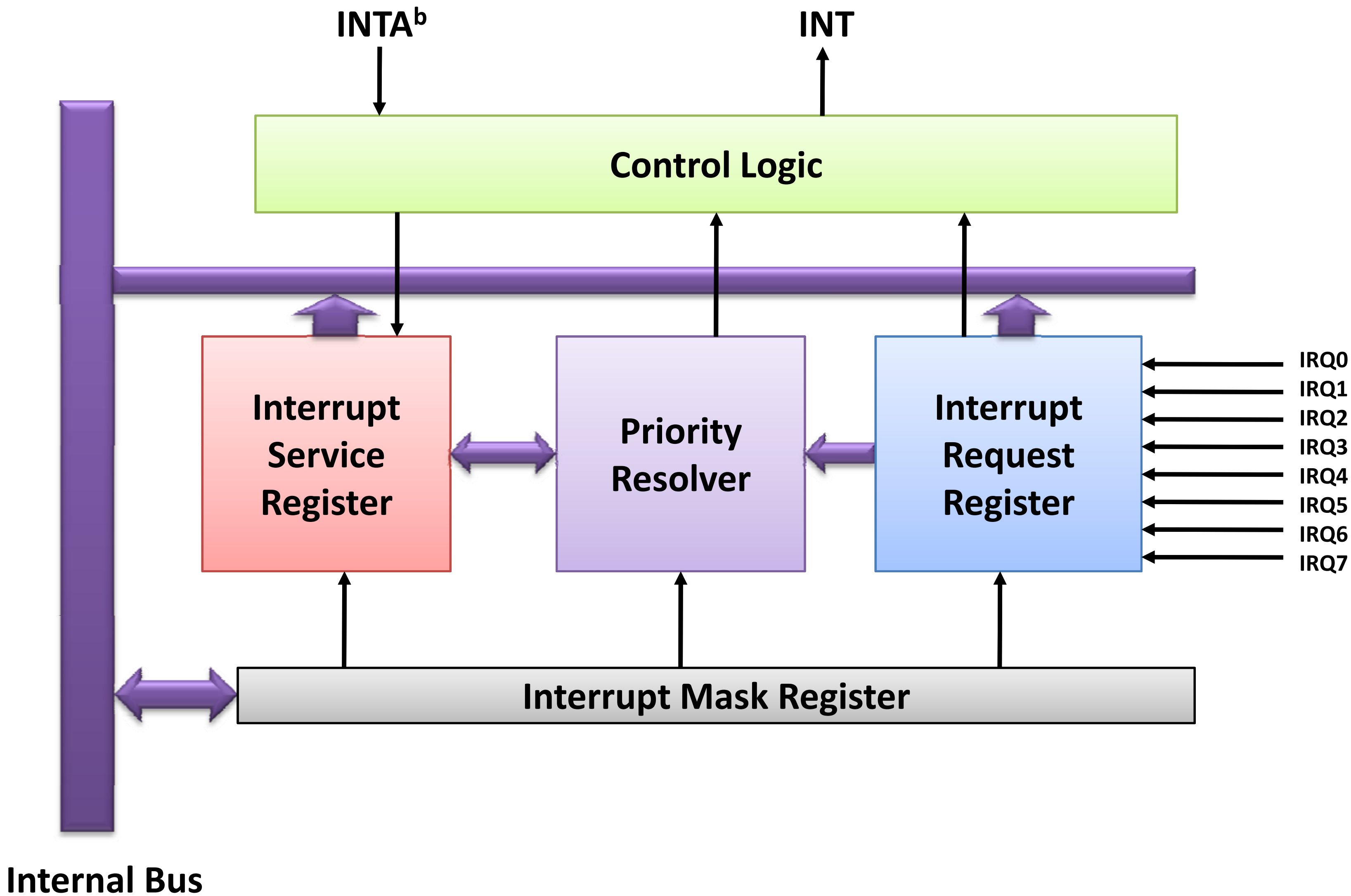
Interrupt Number to Vector Translation

- Interrupt numbers range from 0 to 255
- Interrupt number acts as an index into the interrupt vector table
- Since each vector takes 4 bytes, interrupt number is multiplied by 4 to get the corresponding ISR pointer

What Happens When An Interrupt Occurs?

- Push flags register onto the stack
- Clear interrupt enable and trap flags
  - This disables further interrupts
  - Enable interrupts
- Push CS and IP registers onto the stack
- Load CS with the 16-bit data at memory address
- Load IP with the 16-bit data at memory address

Block Diagram Architecture of 8259



Priority Modes

- Fully Nested Modes
  - IR are arranged in IR0-IR7 and Any IR can be assigned Highest or lowest priority IR4=0 (high), IR3=7 (low)
- Automatics Rotation Mode
  - A device after being served, receive the lowest priority with value 7 01234567➔12345670➔ 23456701
- Specific Rotation Mode
  - User can select any IR for lowest priority
- EOI: End of interrupt
  - Specific EOI Command
  - Automatic EOI: no command necessary
  - Non-Specific EOI: it reset the ISR bit

Control Word (initialization)

CS	A0	Initialization
0	0	ICW1
0	1	ICW2,ICW3,ICW4
1	X	Not Address

ICW1 & ICW2

AD0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	LTIM	0	SGNL	IC4
	0 for x86				1 for Level Trigger 0 for Edge Trigger		1=single 0=Cascade	

AD0	D7	D6	D5	D4	D3	D2	D1	D0
1	T7	T6	T5	T4	T3	T2	T1	T0
	T7=T0 is the assign to IR0, Vector address for ISR							

Masking and Prioritization

- OCW (operation command word)

CS	A0	Operation Command Word
0	0	OCW1
0	1	OCW2,OCW3,OCW4
1	X	Not Address

Programming OCWs: OCW1, OCW2

AD0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0
	Interrupt Masks: 1= Mask Set, 0 =Mask reset							

AD0	D7	D6	D5	D4	D3	D2	D1	D0
0	R	SL	EOI	0	0	L2	L1	L0
	Rote ate	Speci fic	EOI			IR Level to be acted Upon (0-7)		

Reference

- R S Gaonkar, “Microprocessor Architecture”, Chapter 15

Thanks