8086 Interrupts

Source:

'Microprocessors and Interfacing', by Douglas V Hall

Contents

Introduction

- The meaning of “interrupts” is to break the sequence of operation.
- While the microprocessor is executing a program, an “interrupt” breaks the normal sequence of execution of instructions, diverts its execution to some other program called Interrupt Service Routine (ISR).
- After executing, control returns the back again to the main program.

Interrupt

- Keeping moving until interrupted by the sensor.
- Interrupt received then do pre-defined operation.
- After finishing the interrupt service return to normal operation i.e. keep moving forward again.

The processor can be interrupted in the following ways –

- By an external signal generated by the peripheral.
- By an external signal generated by a special instruction in the program.
- By an internal signal generated due to an exceptional condition which occurs while executing an instruction.

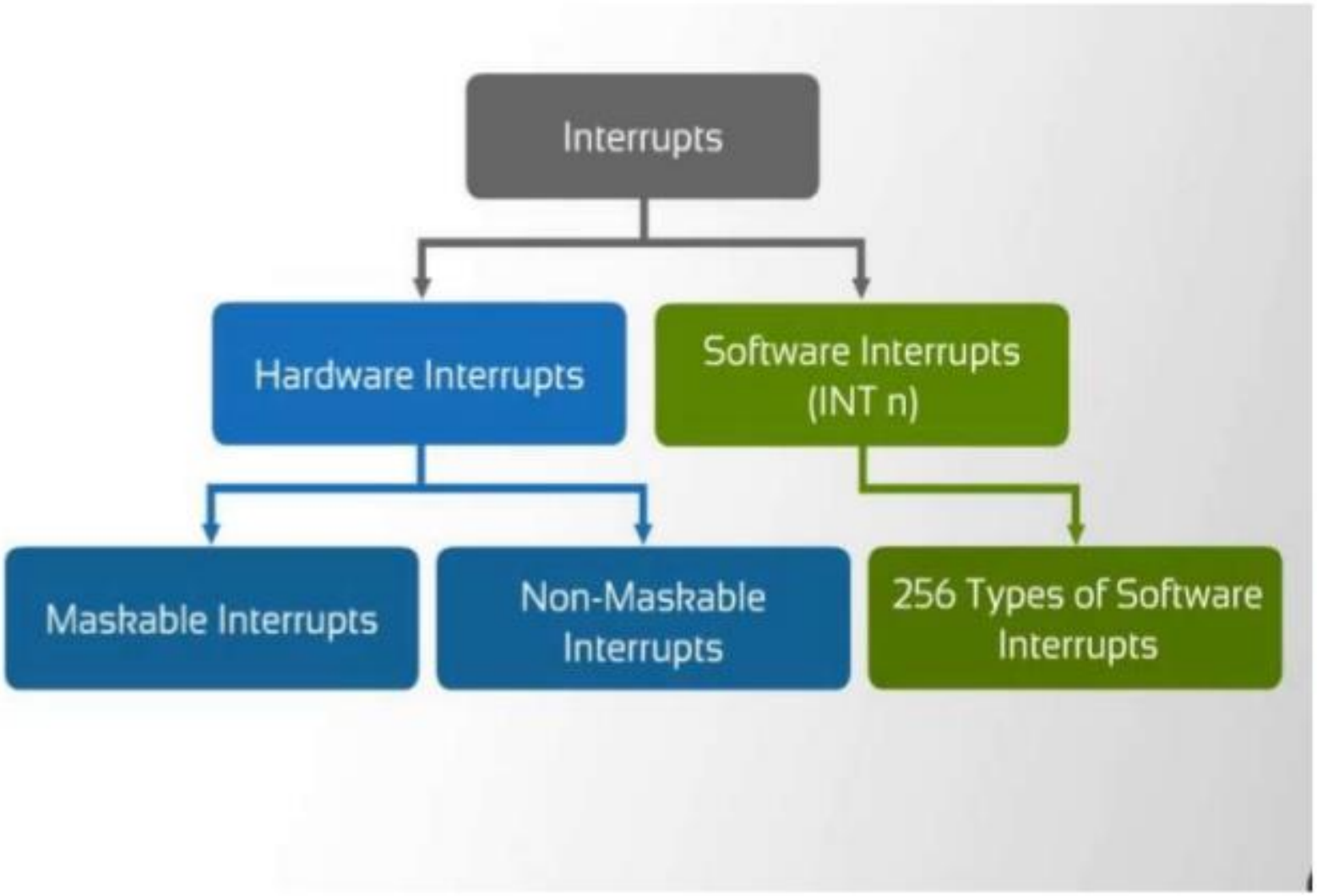
Interrupt

- Normal program execution is interrupted by
  - Some external signal, or
  - A special instruction in the program

- In response to an interrupt,
  - the microprocessor stops executing its normal program &
  - calls a procedure which ‘services’ the interrupt.
  - An IRET instruction at the end of the interrupt-service procedure returns execution to the interrupted program.

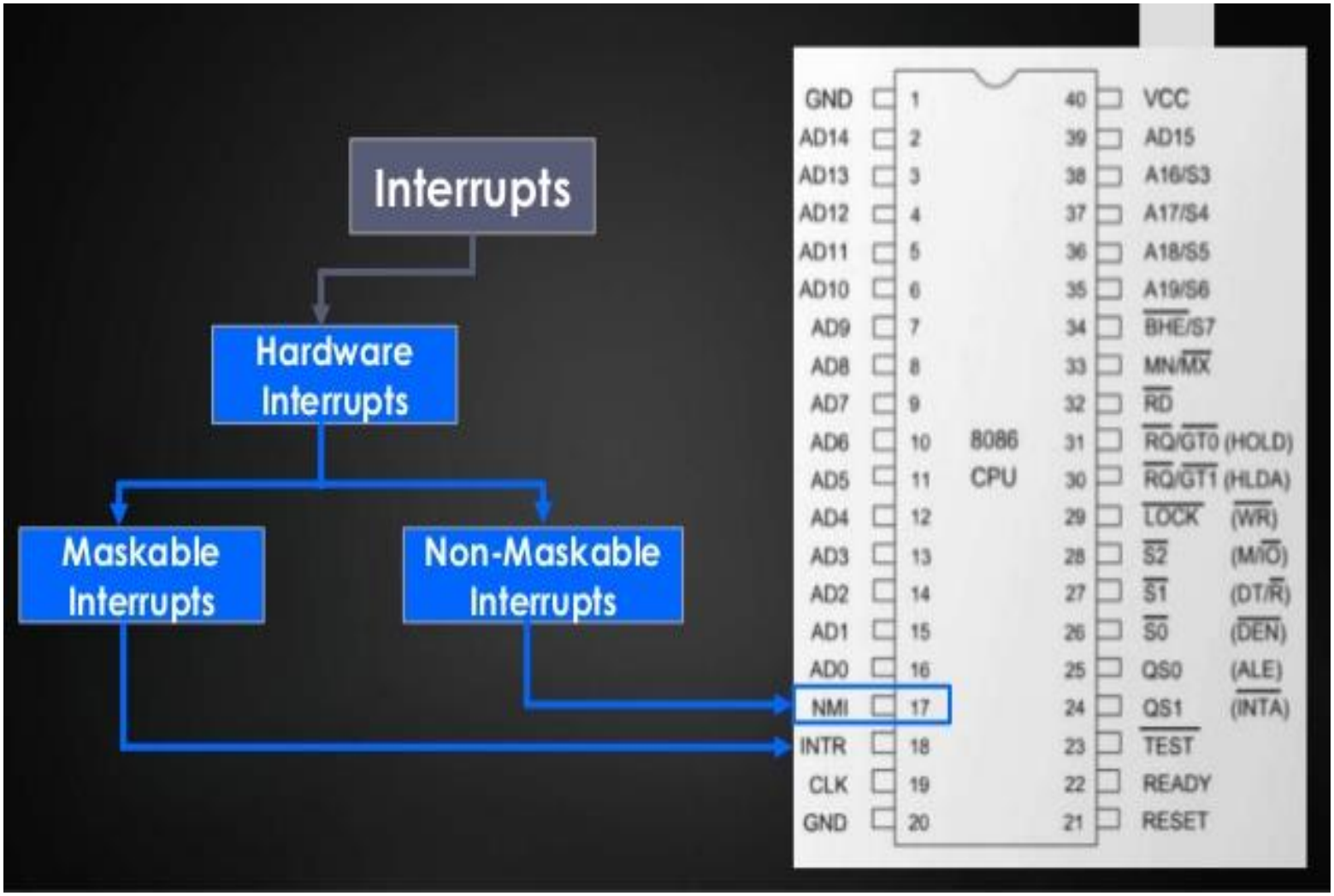
8086 interrupt can come from 3 sources:

1. **Hardware interrupt:** An external signal – applied
   - To the *nonmaskable interrupt* (NMI) input pin, or
   - To the *interrupt* (INTR) input pin
2. **Software interrupt:** Execution of the Interrupt instruction, INT
3. **Error condition:** If some error condition occur by the execution of an instruction. e.g.,
   - *Divide-by-zero interrupt*: If we attempt to divide an operand by zero, the 8086 will automatically interrupt the currently executing program


Interrupts
Hardware Interrupts
Software Interrupts (INT n)
Maskable Interrupts
Non-Maskable Interrupts
256 Types of Software Interrupts

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 microprocessor has two interrupt pins INTR and NMI. The interrupt initiated by applying appropriate signal to these pins are called hardware interrupts of 8086.


Interrupts
Hardware Interrupts
Maskable Interrupts
Non-Maskable Interrupts
GND 1
AD14 2
AD13 3
AD12 4
AD11 5
AD10 6
AD9 7
AD8 8
AD7 9
AD6 10
AD5 11
AD4 12
AD3 13
AD2 14
AD1 15
AD0 16
NMI 17
INTR 18
CLK 19
GND 20
8086 CPU
40 VCC
39 AD15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 BHE/S7
33 MN/MX
32 RD
31 RQ/GT0 (HOLD)
30 RQ/GT1 (HLDA)
29 LOCK (WR)
28 S2 (M/IO)
27 S1 (DT/R)
26 S0 (DEN)
25 QS0 (ALE)
24 QS1 (INTA)
23 TEST
22 READY
21 RESET

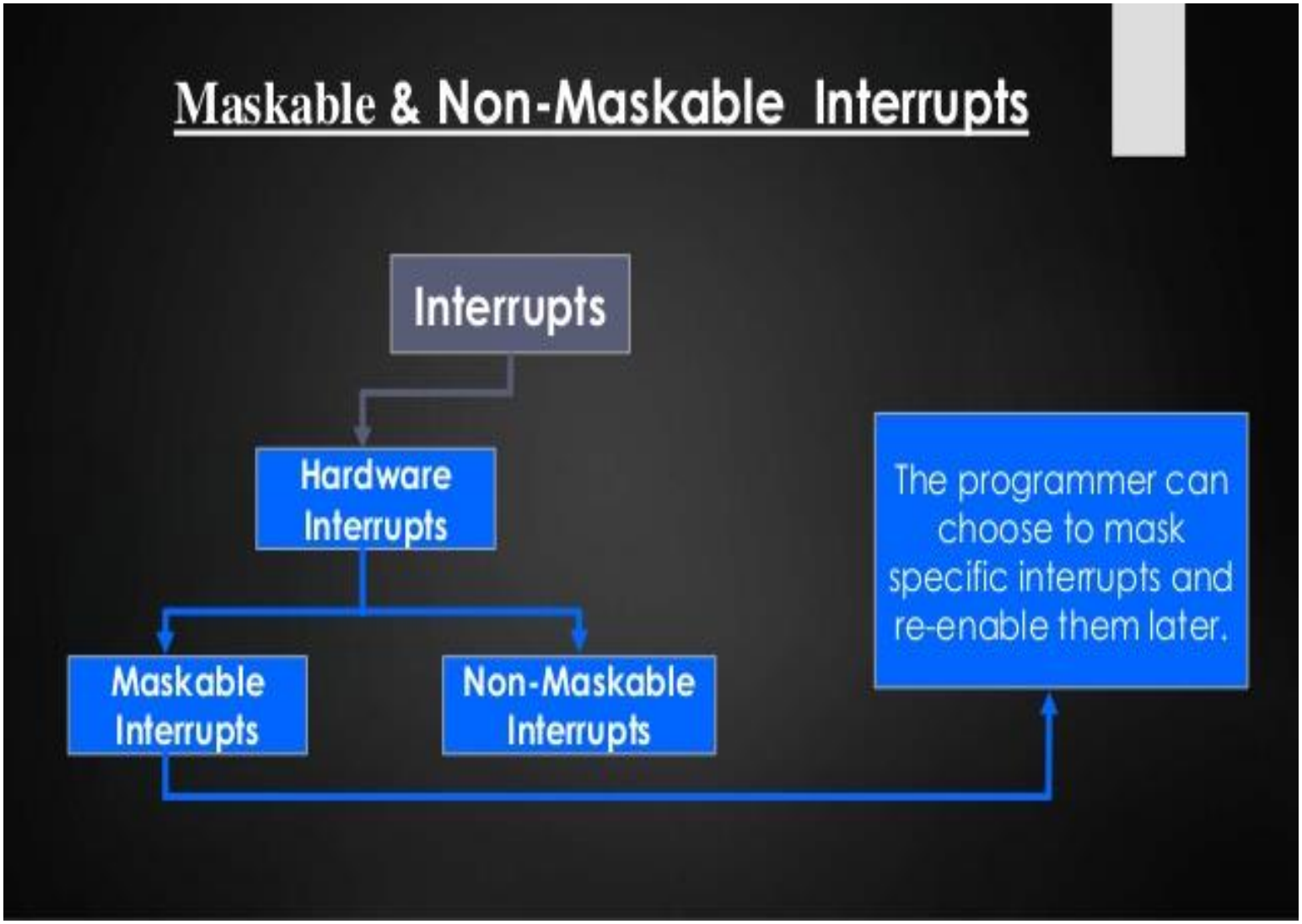
Hardware Interrupts

Used to handle external hardware peripherals , such as key boards , mouse , hard disks , floppy disks , DVD drivers, and printers.

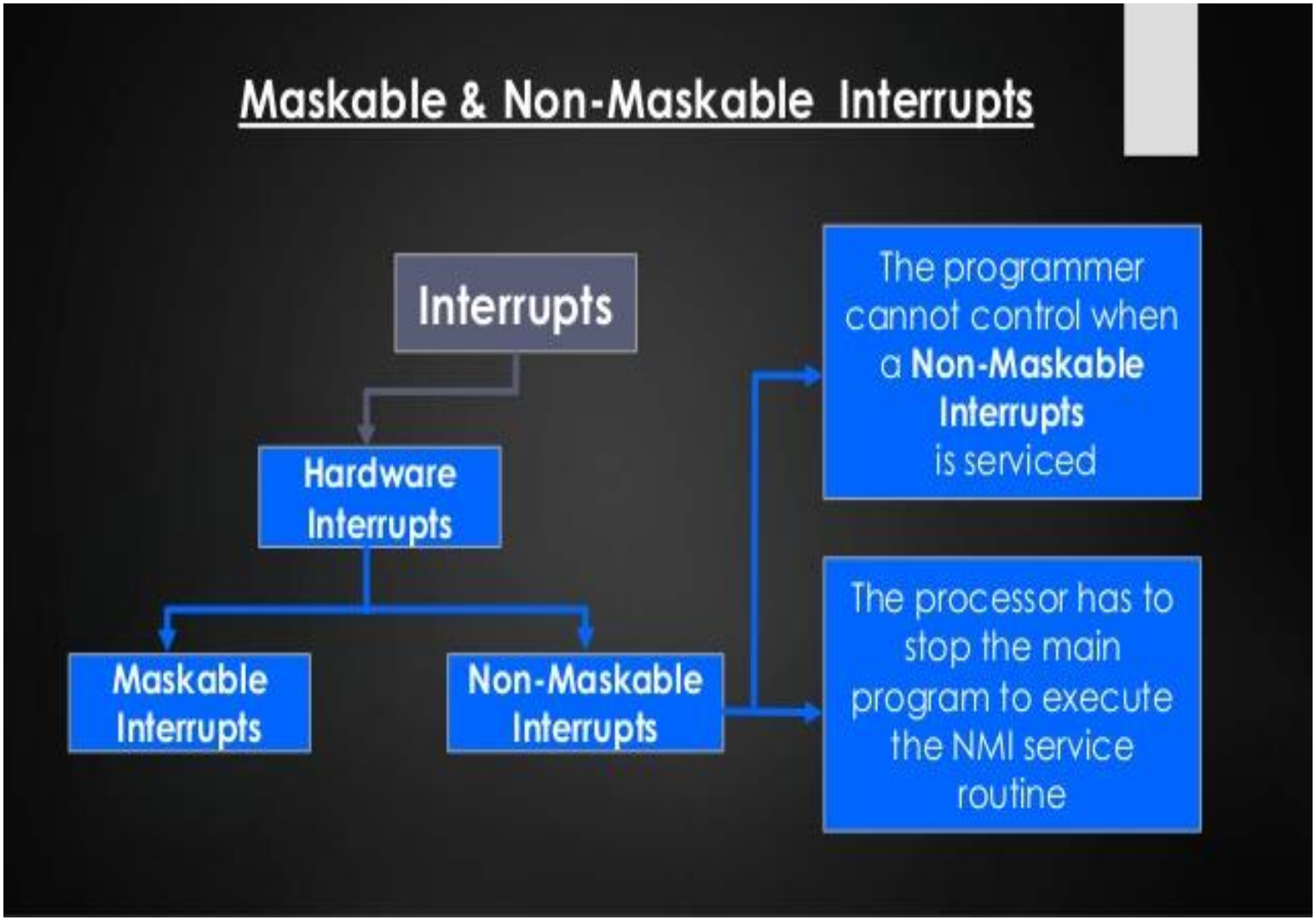
Maskable and Non-Maskable Interrupts

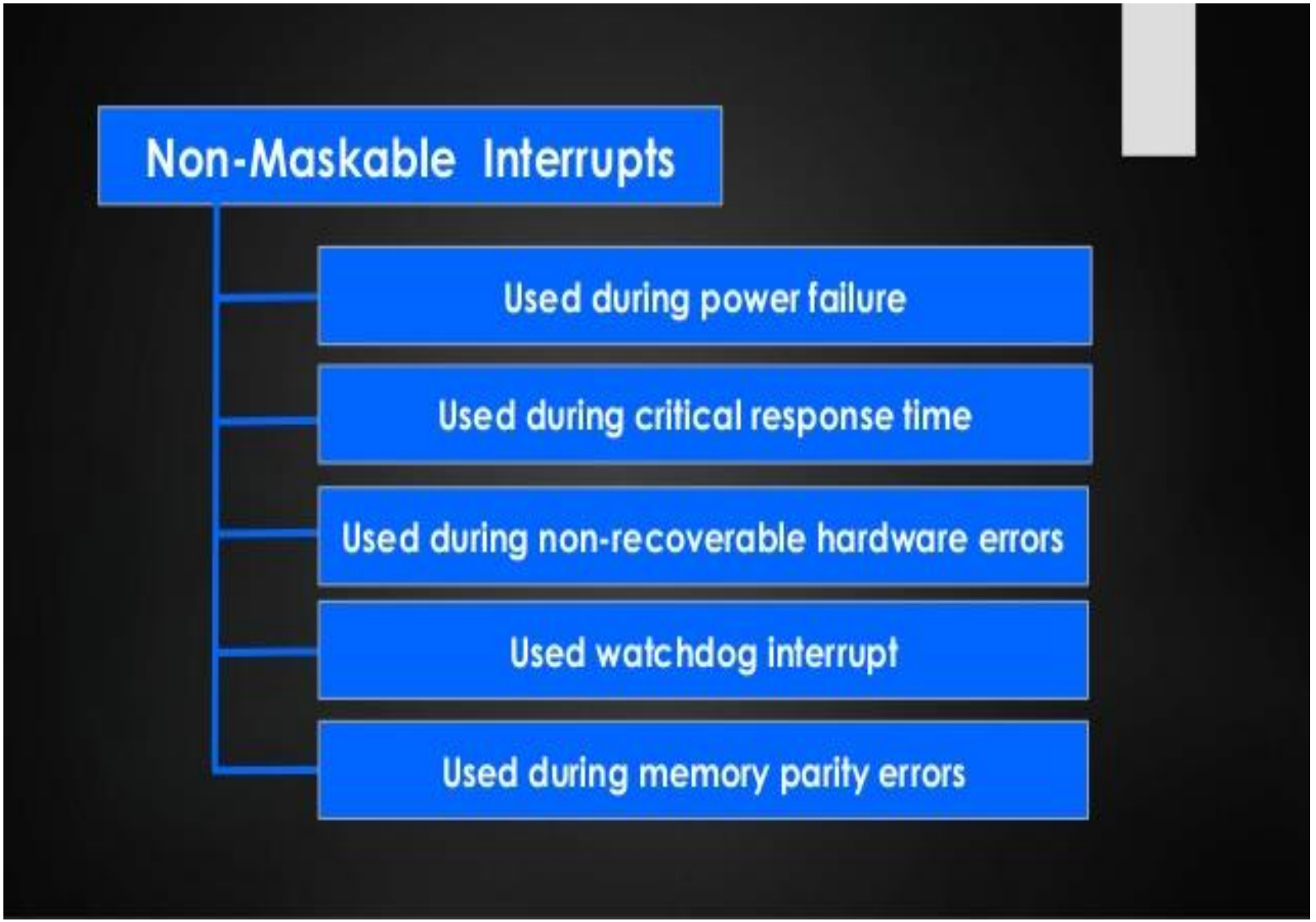
The processor has the facility for accepting or rejecting hardware interrupts. Programming the processor to reject an interrupt is referred to as masking or disabling and programming the processor to accept an interrupt is referred to as unmasking or enabling. In 8086 the interrupt flag (IF) can be set to 1 to unmask or enable all hardware interrupts except NMI. The interrupt whose request can be either accepted or rejected by the processor are called maskable interrupts.

Maskable & Non-Maskable Interrupts



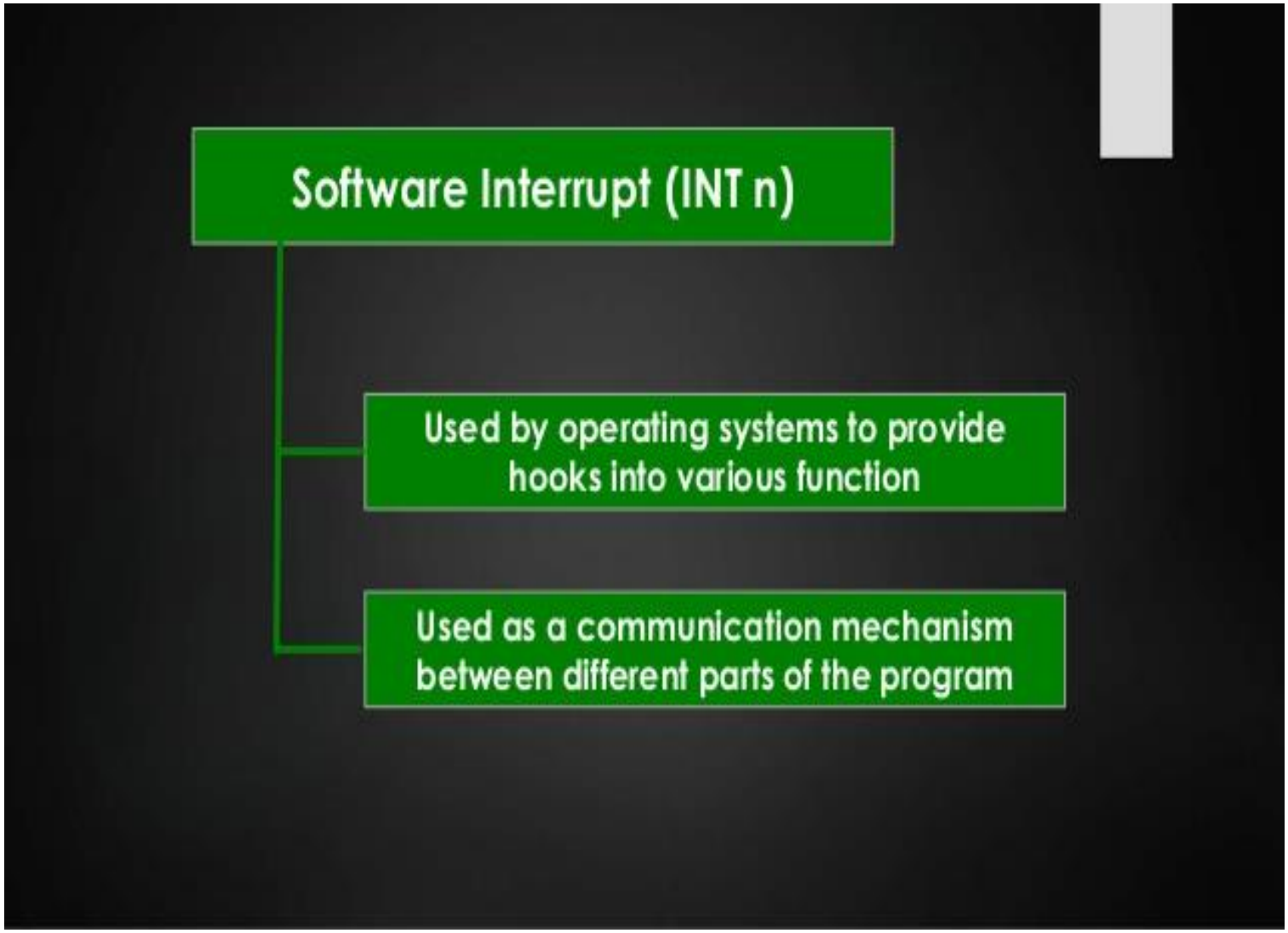
Maskable & Non-Maskable Interrupts




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

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 of software interrupts. The software interrupt instruction is INT n, where n is the type number in the range 0 to 255.


Software Interrupt (INT n)
Used by operating systems to provide hooks into various function
Used as a communication mechanism between different parts of the program

8086 interrupt types

256 interrupts of 8086 are divided into 3 groups

1. **Type 0 to Type 4 interrupts**

   These are used for fixed operations and hence are called dedicated interrupts.

2. **Type 5 to Type 31 interrupts**

   Not used by 8086, reserved for higher processor like 80286, 80386 etc.

3. **Type 32 to Type 255 interrupts**

   Available for user, called user defined interrupts. These can be hardware interrupts and activated through INTR line or can be software interrupts.

- Type – 0 Divide Error Interrupt

  Quotient is large. Can't be fit in AX or divide by 0.

- Type – 1 Single Step Interrupt

  Used for executing the program in single step mode by setting Trap flag

- Type – 2 Non-Maskable Interrupt

  This interrupt is used for execution of NMI pin.

- Type – 3 Break Point Interrupt

  Used for providing break points in the program.

- Type – 4 Overflow Interrupt

  Used to handle any overflow error.

Interrupt Vectors

- The Interrupt Vector contains the address of the interrupt service routine.
- The Interrupt Vector Table is located in the first 1024 bytes of memory at address 00000H-003FFH.
- It contains 256 different 4-byte interrupt vectors, grouped in 18 types
  - 000H: Type 0 (Divide error)
  - 004H: Type 1 (Single-step)
  - 008H: Type 2 (NMI)
  - 00CH: Type 3 (1-byte breakpoint)
  - 010H: Type 4 (Overflow)

Interrupt Vectors

- 014H: Type 5 (BOUND)
- 018H: Type 6 (Undefined opcode)
- 01CH: Type 7 (Coprocessor not available)
- 020H: Type 8 (Double fault)
- 024H: Type 9 (Coprocessor segment overrun)
- 028H: Type 10 (Invalid task state segment)
- 02CH: Type 11 (Segment not present)

Interrupt Vectors

- 030H: Type 12 (Stack segment overrun)
- 034H: Type 13 (General protection)
- 038H: Type 14 (Page fault)
- 03CH: Type 15 (Unassigned)
- 040H: Type 16 (Coprocessor error)
- 044H-07CH: Type 17-31 (Reserved)
- 080H: Type 32-255 (User)

Interrupt Types –

- Type 0: Divide error – Division overflow or division by zero
- Type 1: Single step or Trap – After the execution of each instruction when trap flag set
- Type 2: NMI Hardware Interrupt – '1' in the NMI pin
- Type 3: One-byte Interrupt – INT3 instruction (used for breakpoints)
- Type 4: Overflow – INTO instruction with an overflow flag
- Type 5: BOUND – Register contents out-of-bounds
- Type 6: Invalid Opcode – Undefined opcode occurred in program
- Type 7: Coprocessor not available – MSW indicates a coprocessor
- Type 8: Double Fault – Two separate interrupts occur during the same instruction
- Type 9: Coprocessor Segment Overrun – Coprocessor call operand exceeds FFFFH

Interrupt Types –

- Type 10: Invalid Task State Segment – TSS invalid (probably not initialized)
- Type 11: Segment not present – Descriptor P bit indicates segment not present or invalid
- Type 12: Stack Segment Overrun – Stack segment not present or exceeded
- Type 13: General Protection – Protection violation in 286 (general protection fault)
- Type 14: Page Fault – 80386 and above
- Type 16: Coprocessor Error – ERROR′ = '0' (80386 and above)
- Type 17: Alignment Check – Word/Doubleword data addressed at odd location (486 and above)
- Type 18: Machine Check – Memory Management interrupt (Pentium and above)

- At the end of each instruction cycle, 8086 checks to see if any interrupts have been requested.
- If an interrupt has been requested – the 8086 responds to the interrupt by stepping through the following series of major actions:

1. It decrements the stack pointer by 2 and pushes the flag register on the stack.
2. It disables the INTR interrupt input – by clearing the interrupt flag (IF) in the flag register.
3. It resets the trap flag (TF) in the flag register.
4. It decrements the stack pointer by 2 and pushes the current code segment register contents on the stack.

5. It decrements the stack pointer again by 2 and pushes the current instruction pointer contents on the stack.
6. CS and IP are loaded with new addresses derived from the interrupt type number.
7. 8086 microprocessor executes the ISR to service the interrupt.

Divide-by-zero interrupt – Type 0

- 8086 will automatically do a type – 0 interrupt if
  - The result of a DIV operation or
  - An IDIV operation

  is too large to fit in the destination register

  → Too large!? Infinity!

Write program in a way so that can manage –

- E.g., make sure that the divisor is not zero

  [to avoid infinity result]; and

- Do the division in several steps so that result of the division will never be too large.

Another way –

- Write an interrupt-service procedure which takes the desired action when an invalid division occurs.

Nonmaskable interrupt – type 2

- 8086 will automatically do a *type 2* interrupt response when – it received a low-to-high [0 to 1] transition on its **NMI** input pin.

- This interrupt **can not** be disabled/masked – by any program instructions.
- As this input can’t be intentionally or accidentally disabled, external system must be taken care of.

Example 1 –

- We could have a **pressure sensor** on a large steam boiler – connected to the **NMI** input.
- If the pressure goes above a preset limit, the sensor will send an interrupt signal to the 8086.
- Type 2 interrupt-service procedure for this case might
  - turn off the fuel to the boiler
  - open a pressure-relief valve and
  - sound an alarm!

Example 2 –

To save program data in case of a system power failure.

- When AC power fails – some external circuitry detects it.
- Sends an interrupt signal to the **NMI** input pin.
- Because of the large filter capacitors in most power supplied, the dc system power will remain for perhaps 50ms – after the AC power is gone.

- This is enough time for a type 2 interrupt-service procedure to copy program data to some RAM, which has a battery backup power supply.
- When the AC power returns, program data can be restored from the battery-backed RAM,
- and the program can resume execution where it left off.

Conclusion

The CPU executes program, as soon as a key is pressed, the keyboard generates an interrupt. The CPU will response to the interrupt – read the data. After that returns to the original program. So by proper use of interrupt, the CPU can serve many devices at the same time.