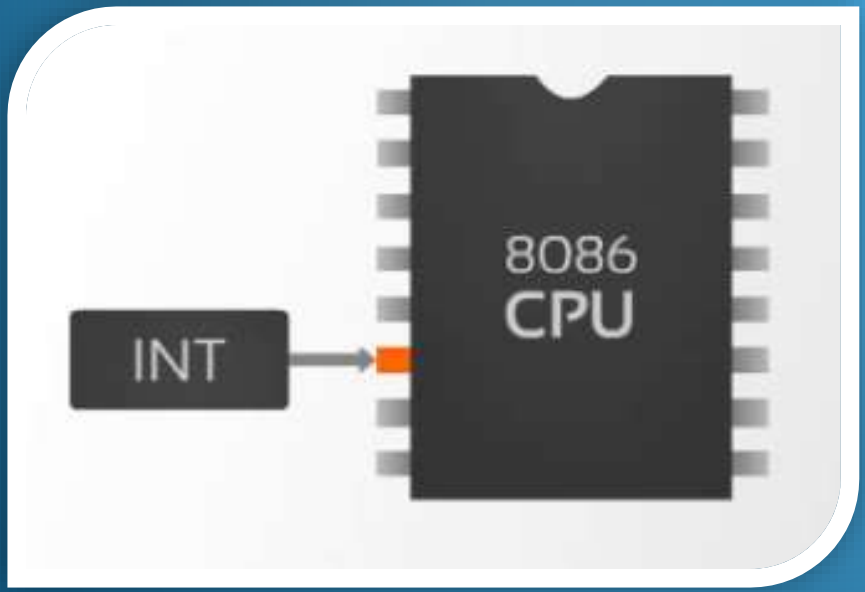
Interrupts on 8086 Microprocessor



- When your phone rings during a lecture, what will happen?
- When you are studying then your cell phone rings – what will you do?

- When you finish talking on the phone then you will continue with your study.

- Now your phone rings again and someone also knocking at your door then what will you do?

- When being interrupted, you will perform some pre-defined action.
- Interrupt has priority – some interrupt is more important than the others. For example, answering your phone is more important

Introduction

- An interrupt is used to cause a temporary halt in the execution of program.
- 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 ISR, IRET returns the control back again to the main program. Interrupt processing is an alternative to polling.

Need for Interrupt:

- Interrupts are particularly useful when interfacing I/O devices, that provide or require data at relatively low data transfer rate.

Sources of Interrupts:

Three types of interrupts sources are there:

1. An external signal applied to NMI or INTR input pin(hardware interrupt)
2. Execution of Interrupt instruction(software interrupt)
3. Interrupt raised due to some error condition produced in 8086 instruction execution process. (Divide by zero, overflow errors etc)

8086 Interrupt Sources

An 8086 interrupt can come from any one of the following three sources:

1. An external signal applied to the non-maskable interrupt (NMI 17 pin) pin or to the interrupt (INTR 18 pin) pin. An interrupt caused by a signal applied to one of these inputs is called *hardware interrupt*.
2. The execution of the instruction INT n, where n is the interrupt type that can take any value between 00H and FFH. This is called *software interrupt*.
3. An error condition such as divide-by-0, which is produced in the 8086 by the execution of the DIV/IDIV instruction or the trap interrupt.

How to get key typed in the keyboard or a keypad?

- **Polling :-**

The CPU executes a program that check for the available of data, If a key is pressed then read the data, otherwise keep waiting or looping!!!

- **Interrupt:-**

The CPU executes other 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"

Polling vs Interrupt

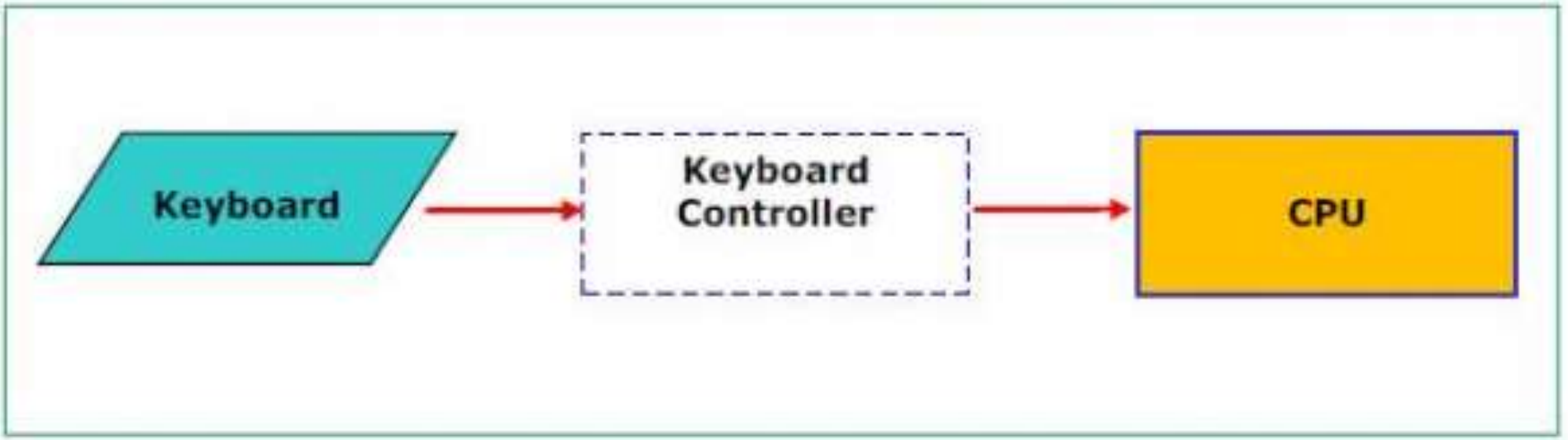
- The keyboard controller can hold only a single keystroke. Therefore, the keyboard controller must be freed before the next keystroke arrives.
- The keystroke is passed to the CPU by putting it in the keyboard buffer. So, the keyboard controller keeps on passing the keystroke input to the CPU,

But how does the CPU attend to it?

The CPU is not at the disposal of the keyboard controller; it is usually busy doing several other operations. So, we need some mechanism to indicate to the CPU that a keystroke has arrived. How is this done? There are two approaches to making sure that the CPU pays attention:

- Polling-based
- Interrupt-based

Example: Polling Vs Interrupt

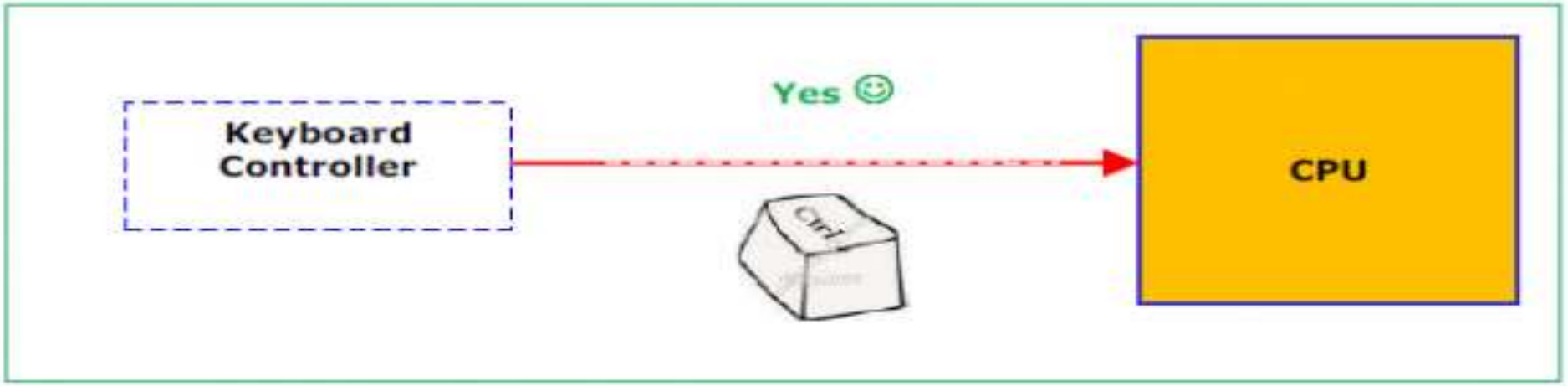


Keystroke causes interrupt

Polling Based System:-

Figure 2: Polling-based interrupt handling

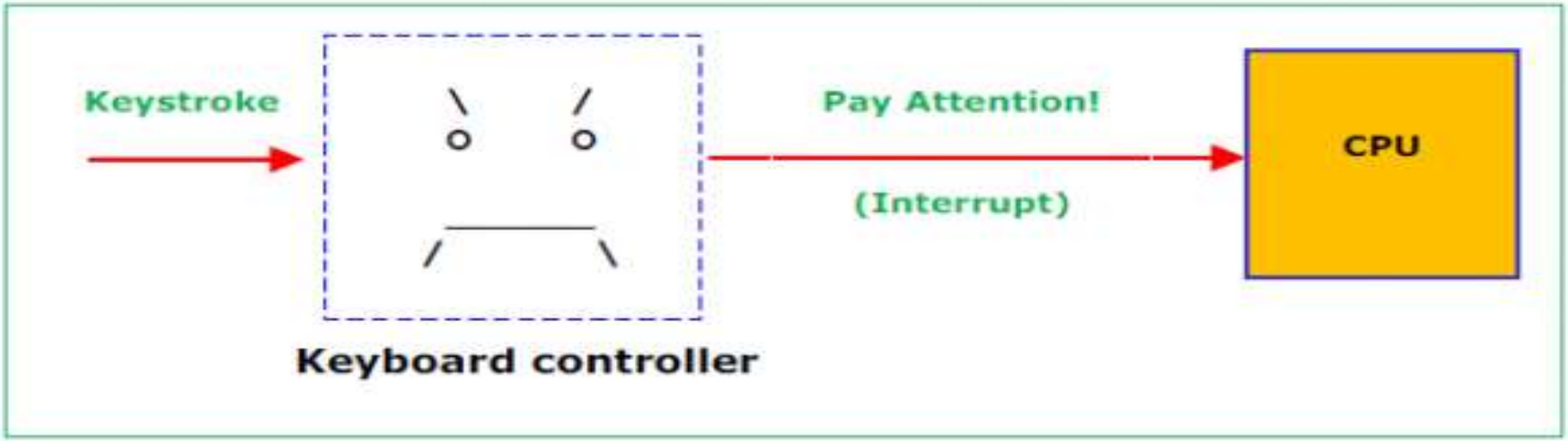
- The CPU executes a program that check for the available of data If a key is pressed then read the data, otherwise keep waiting or looping!!!



Keystroke passed to the CPU

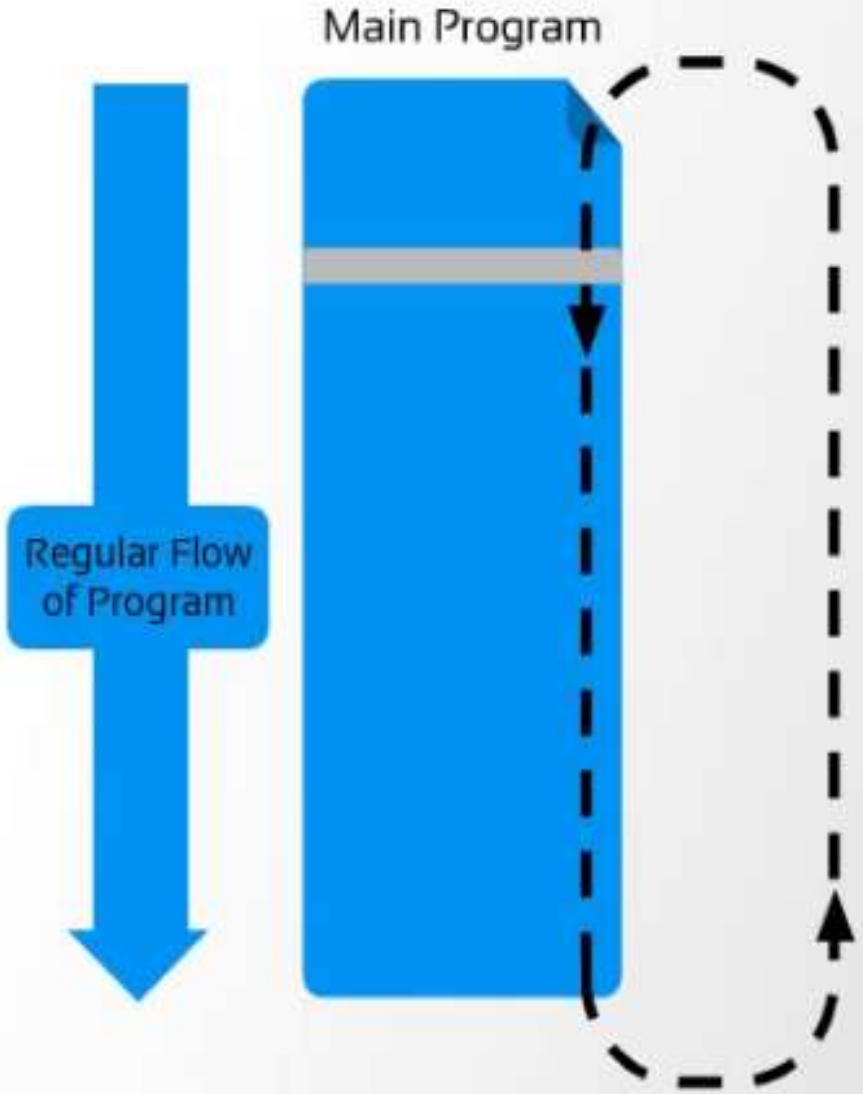
No keystroke for CPU

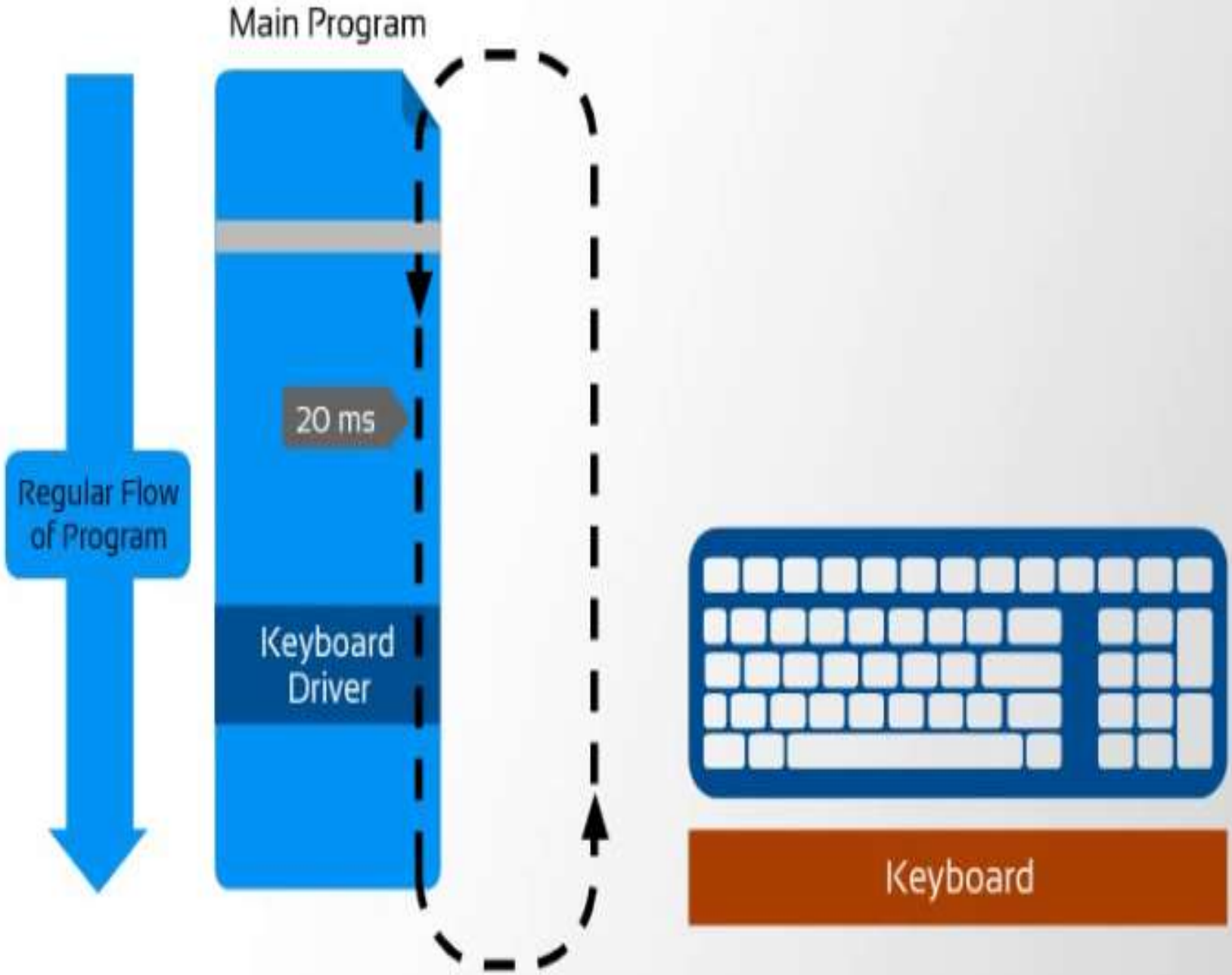
Interrupt-based systems

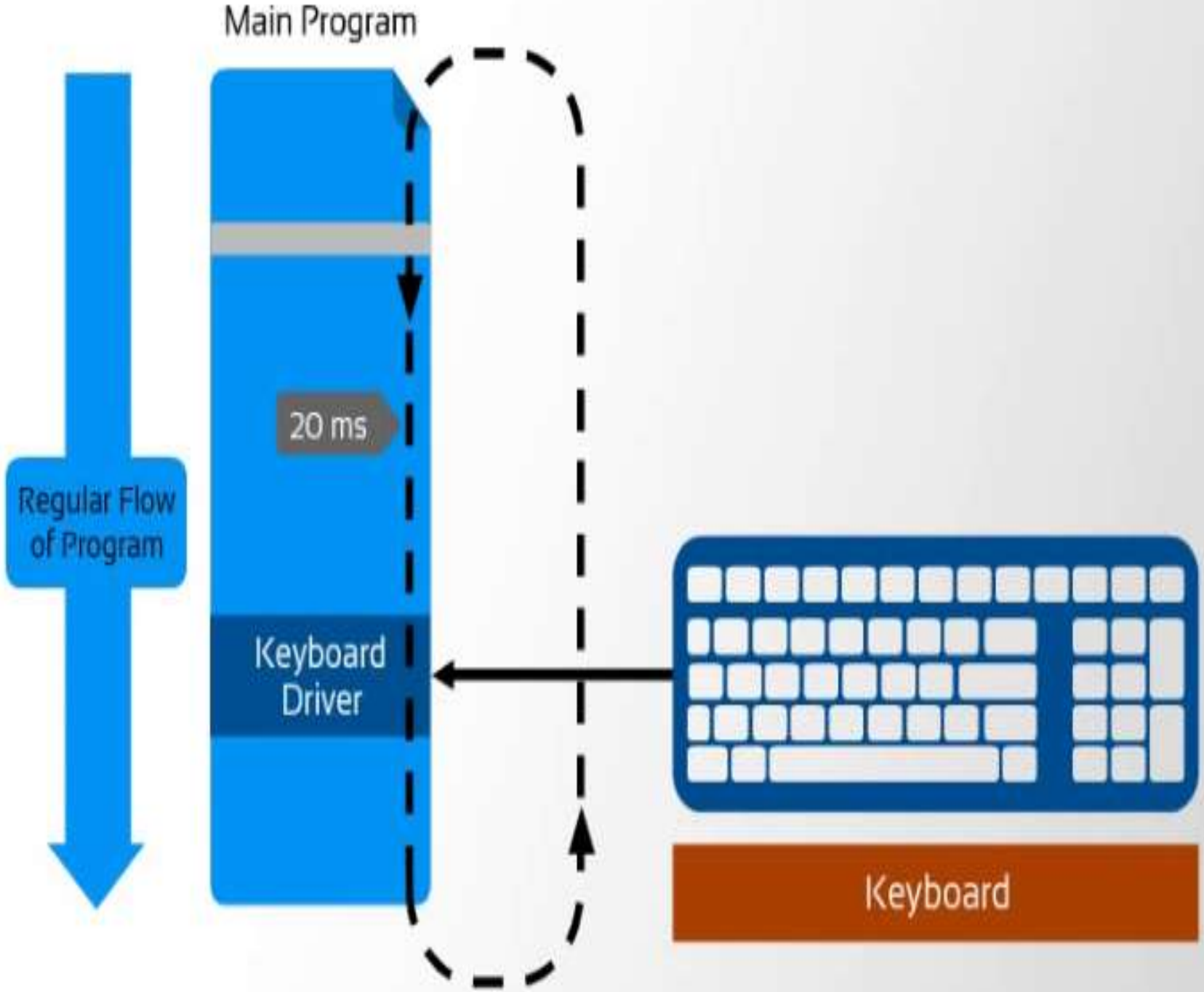


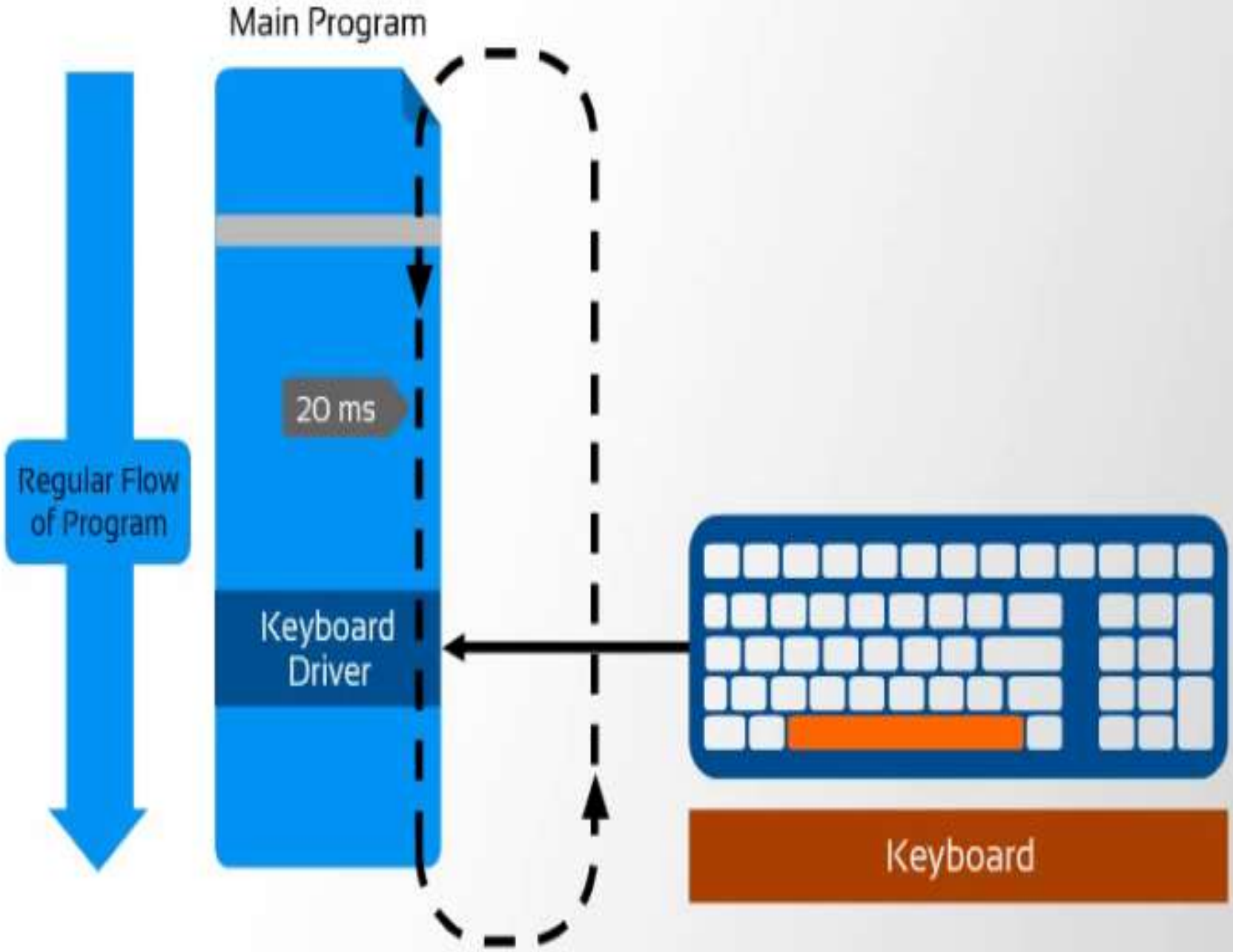
Interrupt-based approach

- The CPU executes other 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"

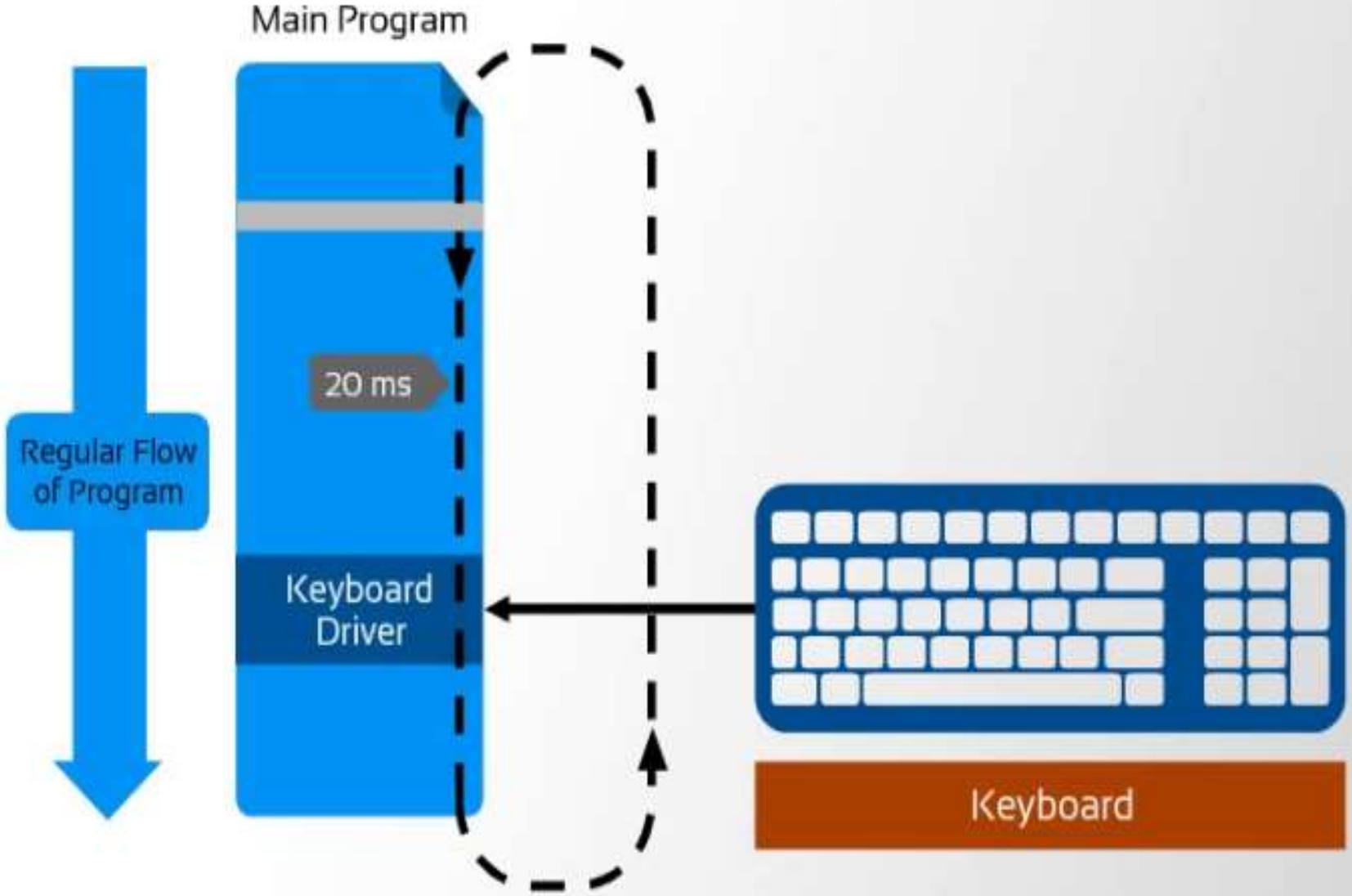

Main Program
Regular Flow of Program


Main Program
Regular Flow of Program
20 ms
Keyboard Driver
Keyboard

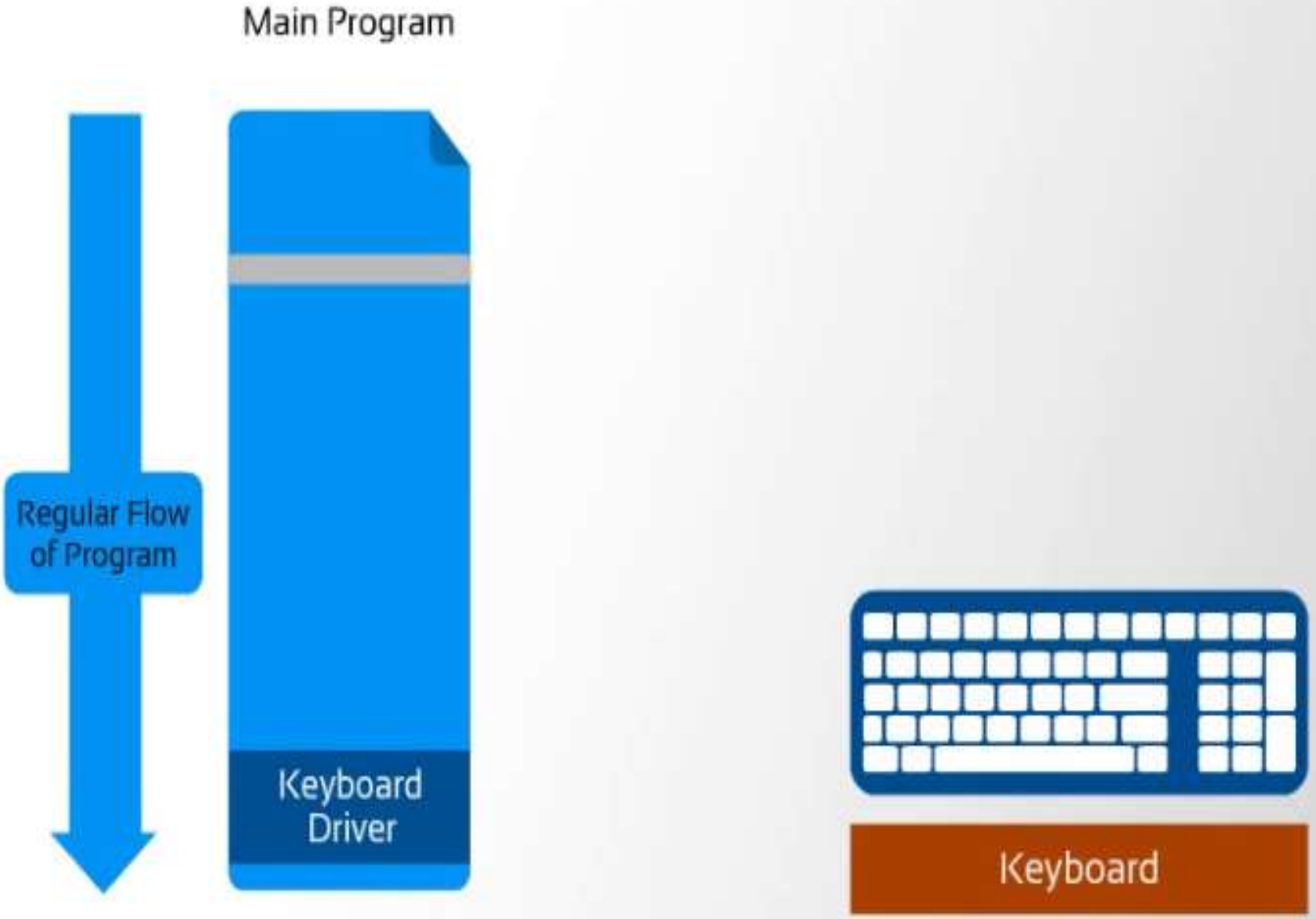

Main Program
Regular Flow of Program
20 ms
Keyboard Driver
Keyboard

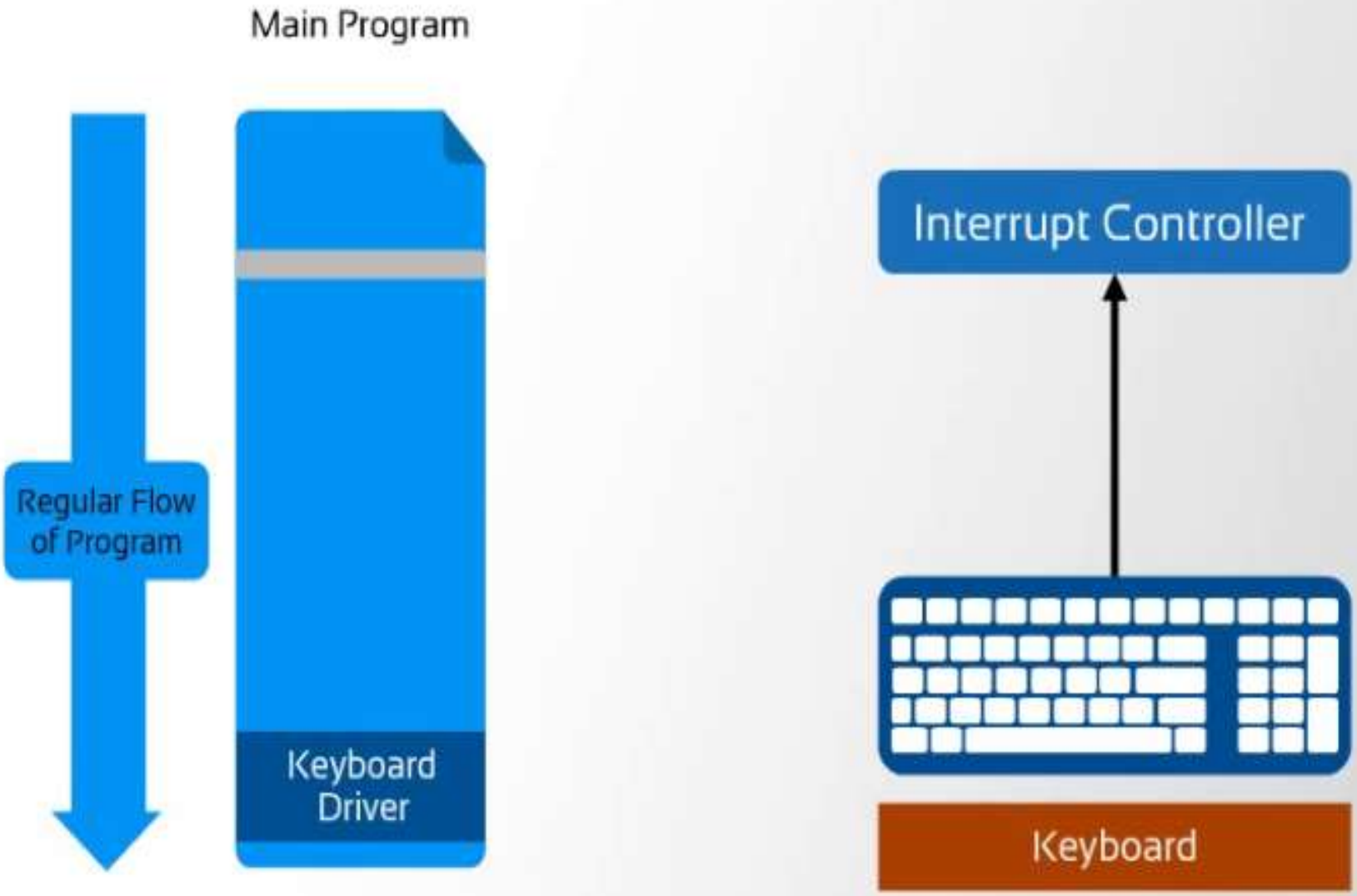

Main Program
Regular Flow of Program
20 ms
Keyboard Driver
Keyboard

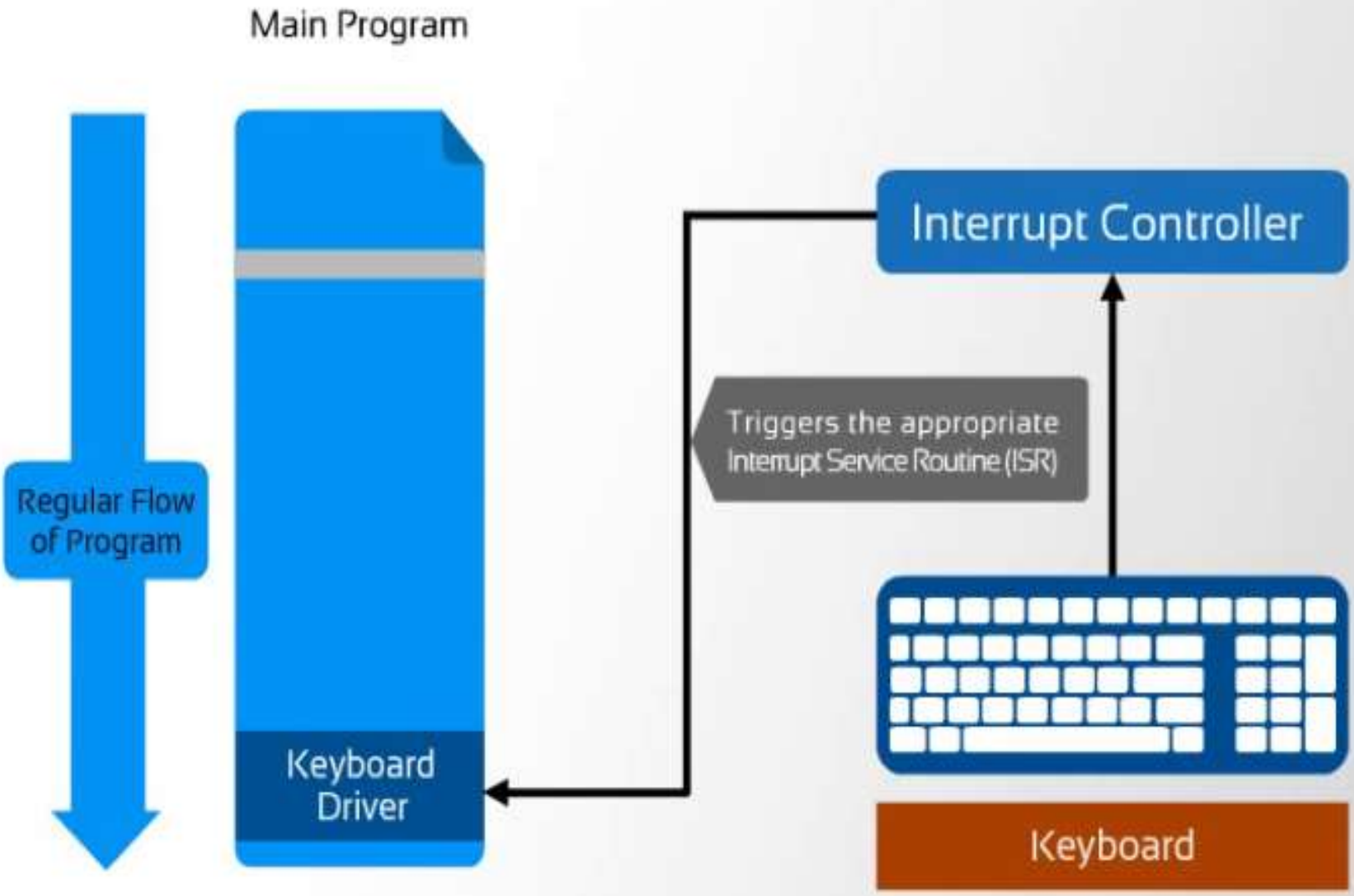
This is called Polling.

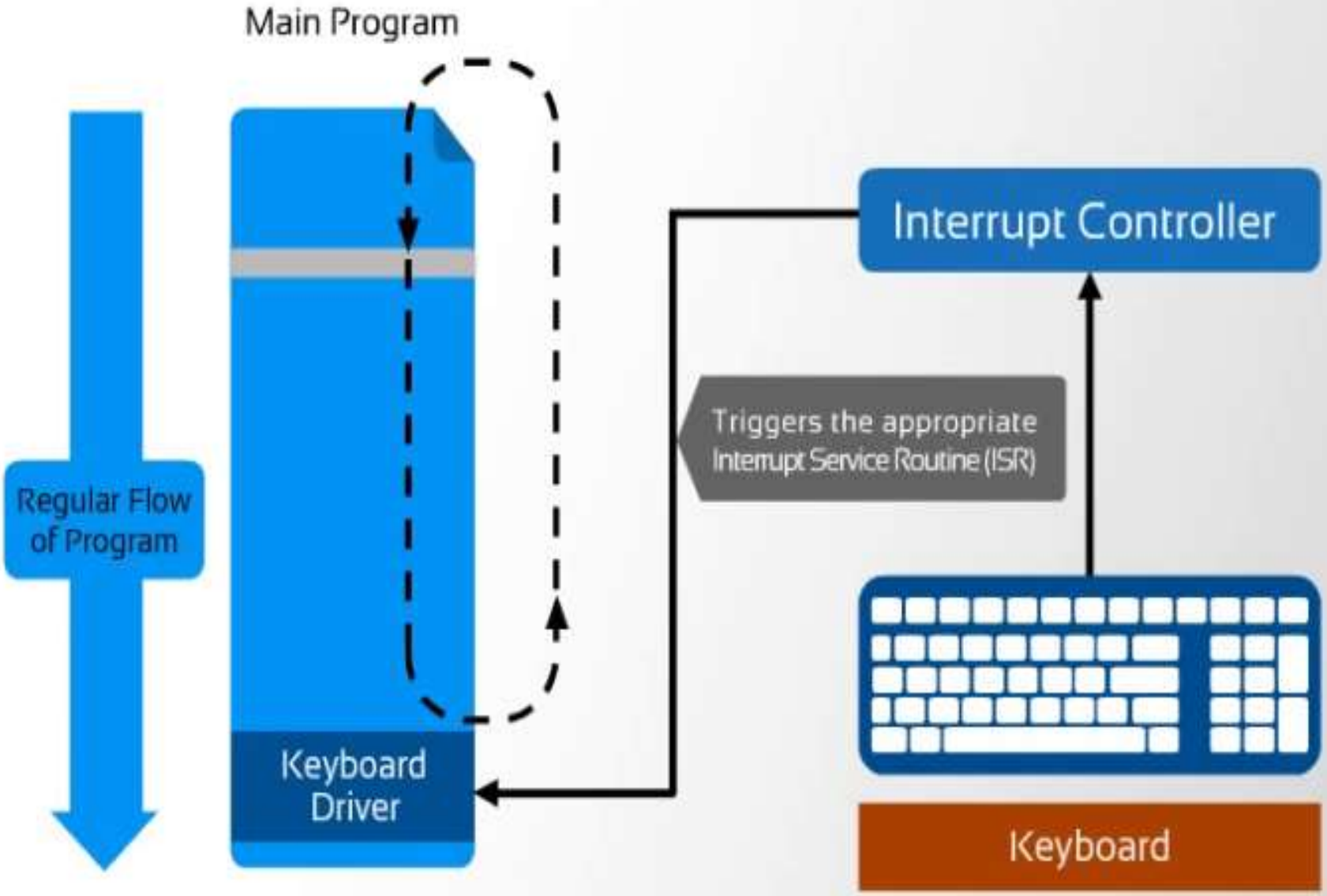


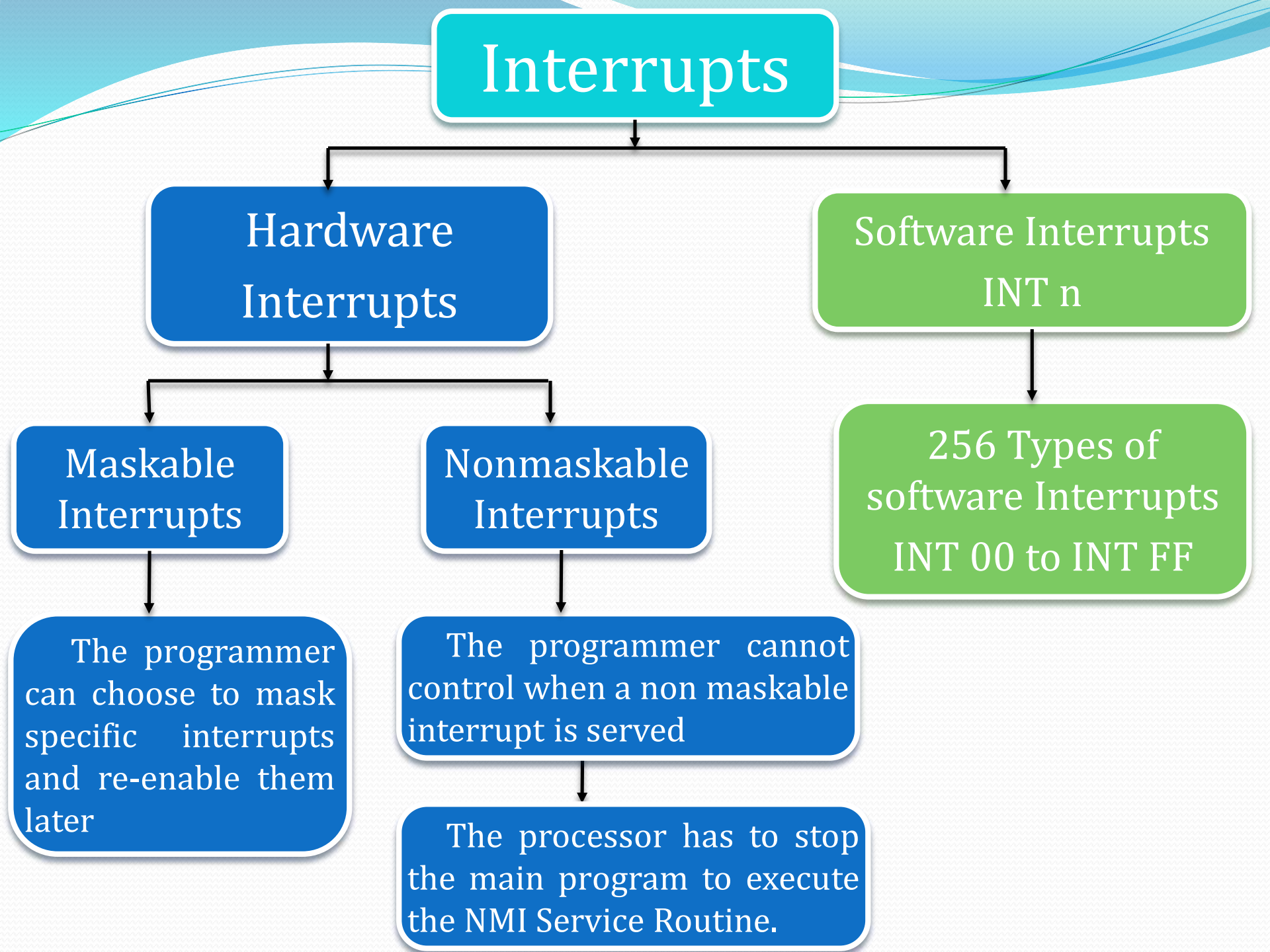
How would the processor work with an Interrupt

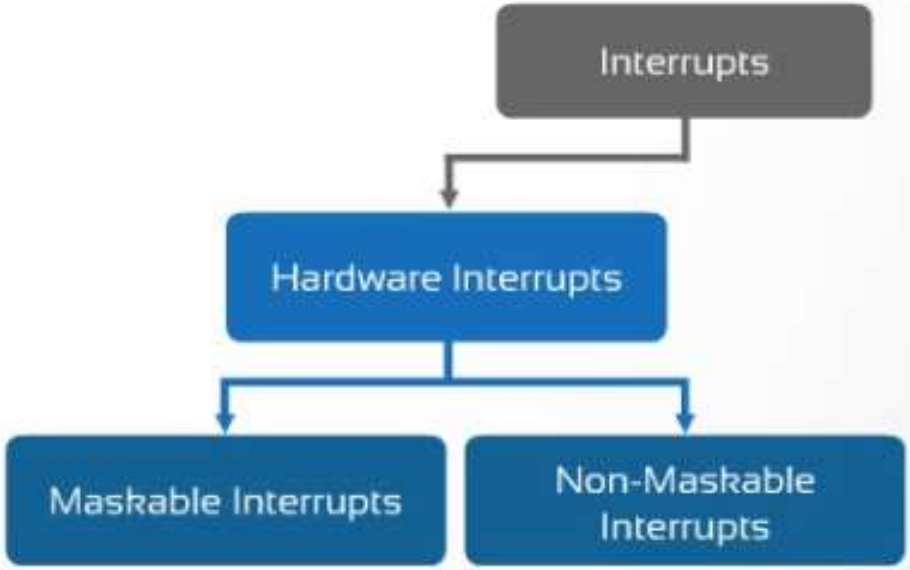



Main Program
Regular Flow of Program
Keyboard Driver
Interrupt Controller
Keyboard

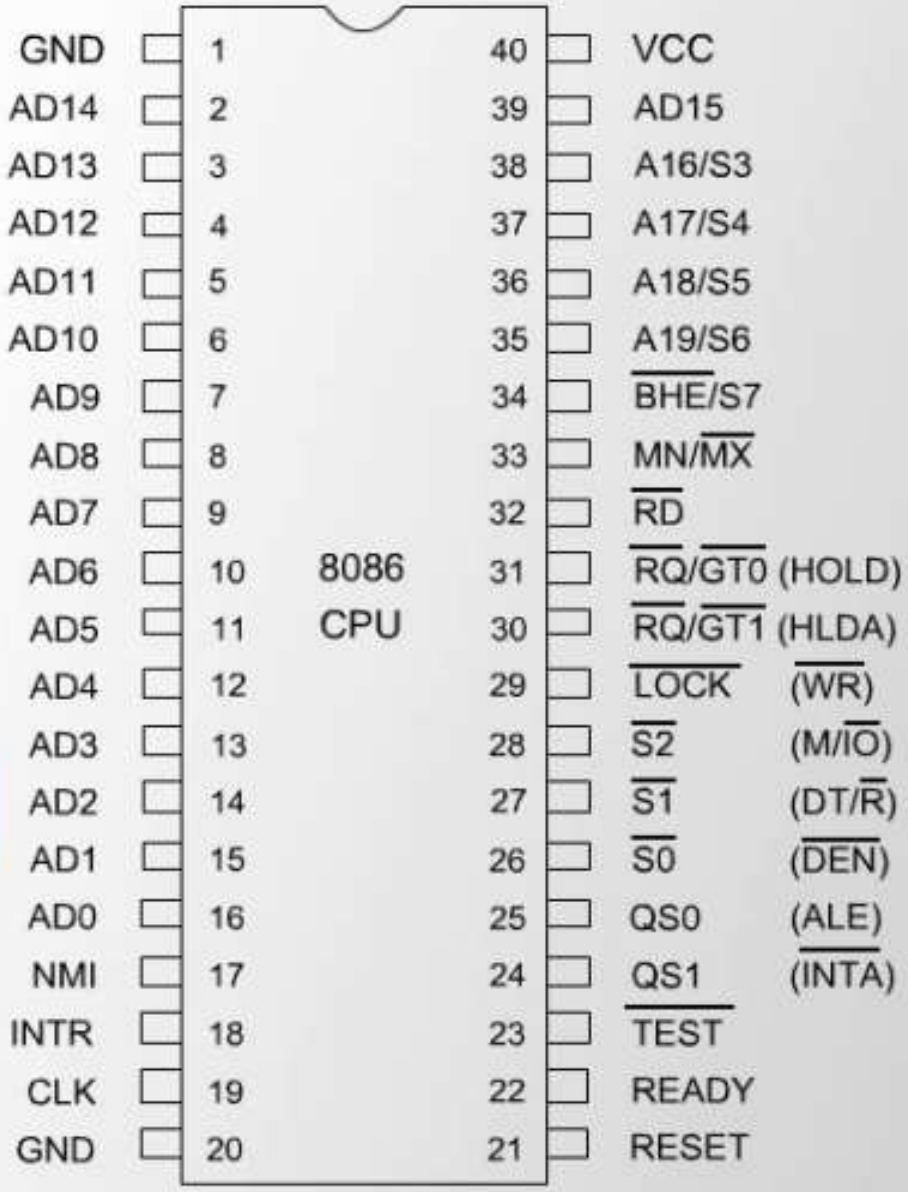

Main Program
Regular Flow of Program
Keyboard Driver
Interrupt Controller
Triggers the appropriate Interrupt Service Routine (ISR)
Keyboard

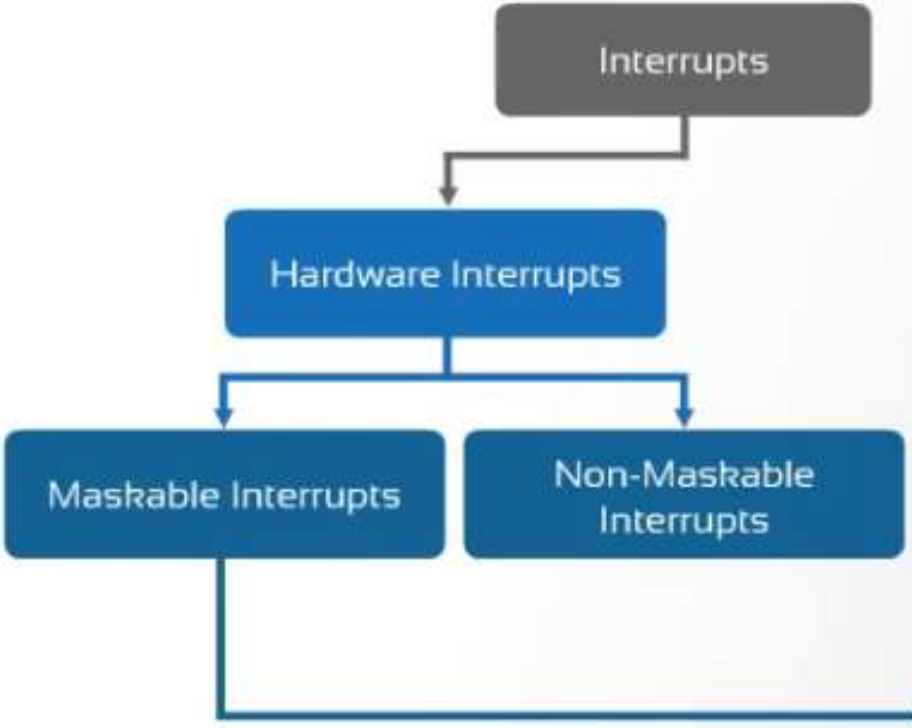

Main Program
Regular Flow of Program
Keyboard Driver
Interrupt Controller
Triggers the appropriate Interrupt Service Routine (ISR)
Keyboard


Interrupts
Hardware Interrupts
Software Interrupts INT n
Maskable Interrupts
Nonmaskable Interrupts
256 Types of software Interrupts INT 00 to INT FF
The programmer can choose to mask specific interrupts and re-enable them later
The programmer cannot control when a non maskable interrupt is served
The processor has to stop the main program to execute the NMI Service Routine.

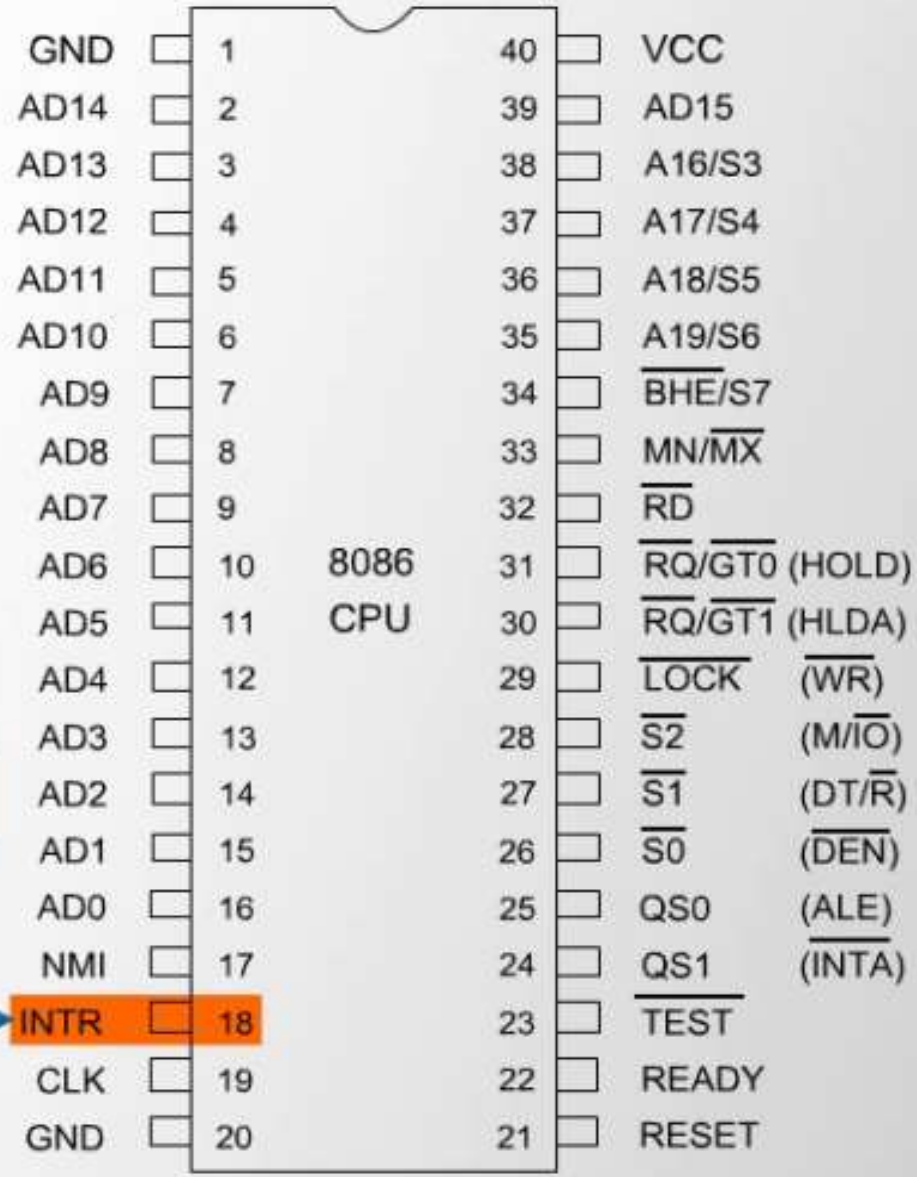

Interrupts
Hardware Interrupts
Maskable Interrupts
Non-Maskable Interrupts

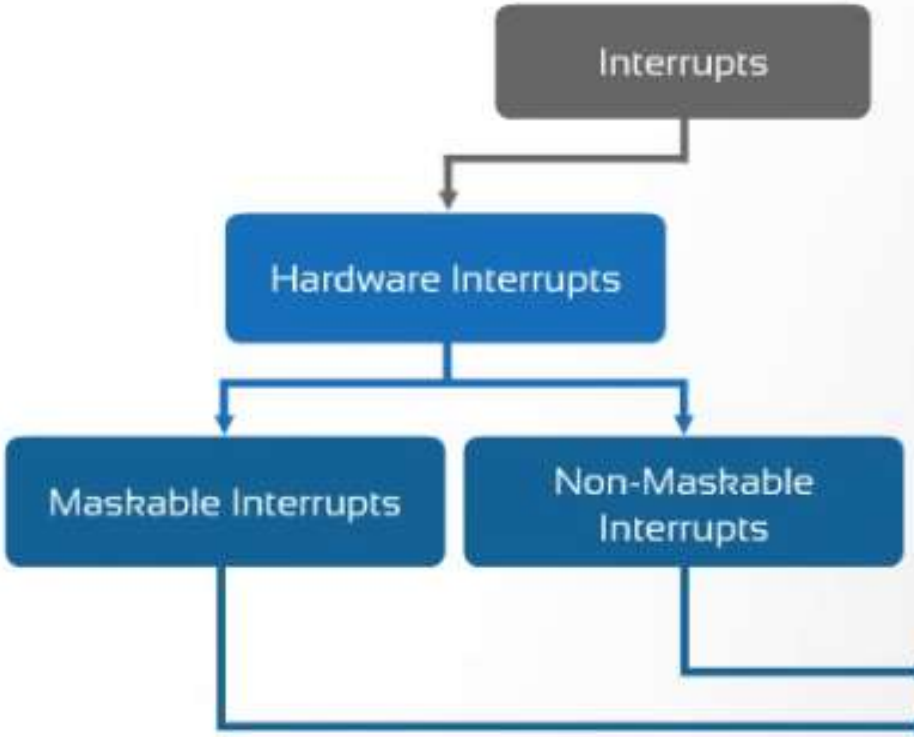
8086 CPU


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

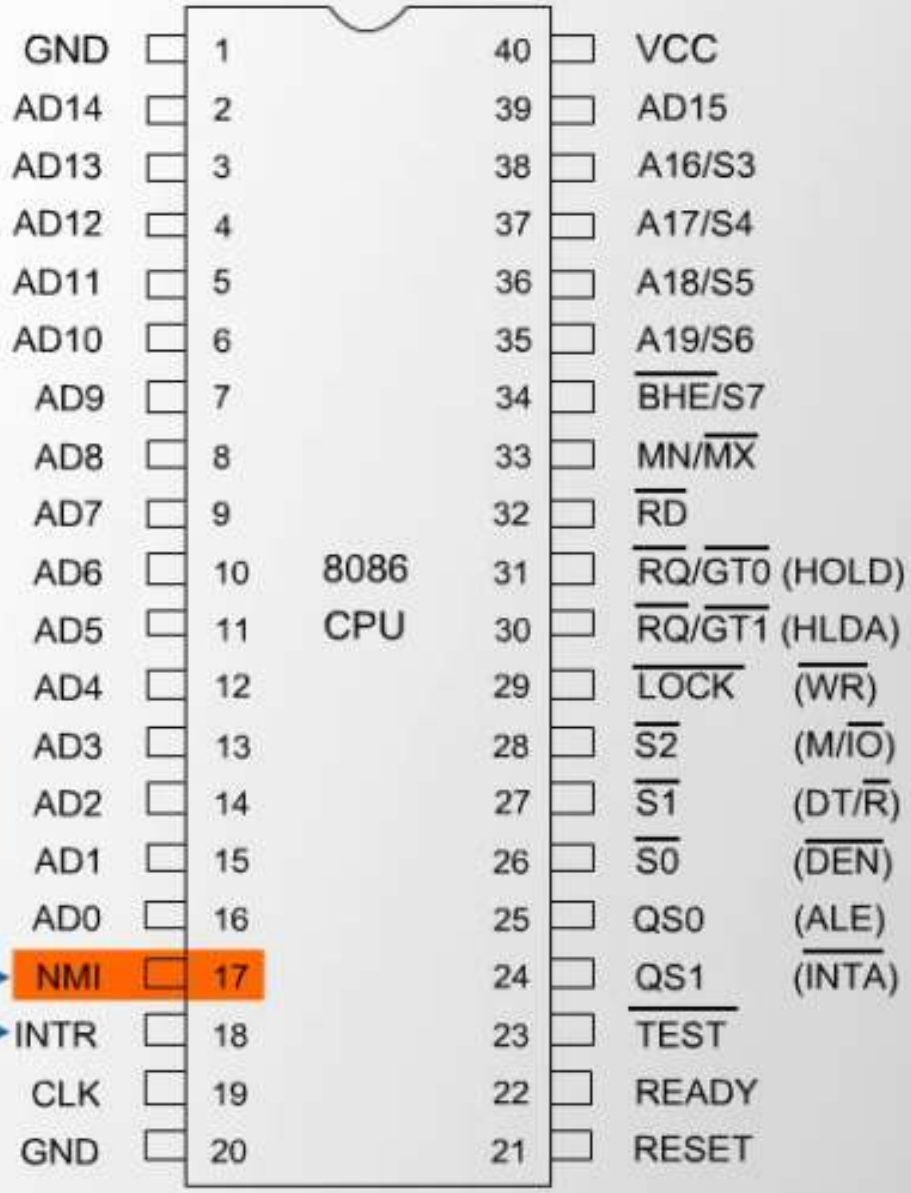

Interrupts
Hardware Interrupts
Maskable Interrupts
Non-Maskable Interrupts

8086 CPU


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


Interrupts
Hardware Interrupts
Maskable Interrupts
Non-Maskable Interrupts

8086 CPU


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

Processing of Interrupt by the Processor

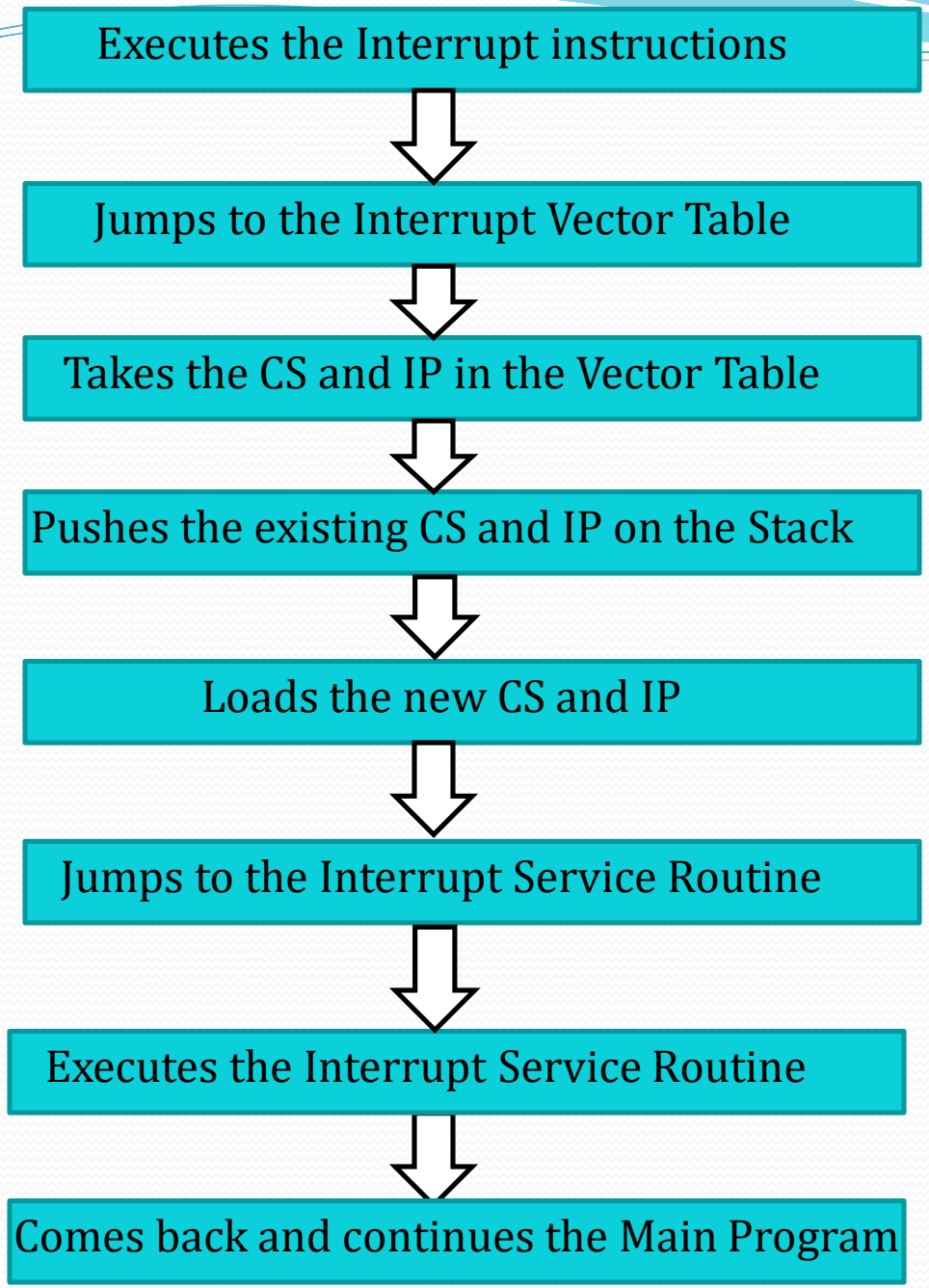
1. Executes the INT instruction
2. Interrupts the INT instruction during the assembly time
3. Moves the INT instruction to the Vector Table
   - Vector Table occupies location 00000H to 0003FFh of the program memory.
   - It contains the code segment (CS) and Instruction Pointer (IP) for each kind of interrupt.

8086 Interrupt Processing

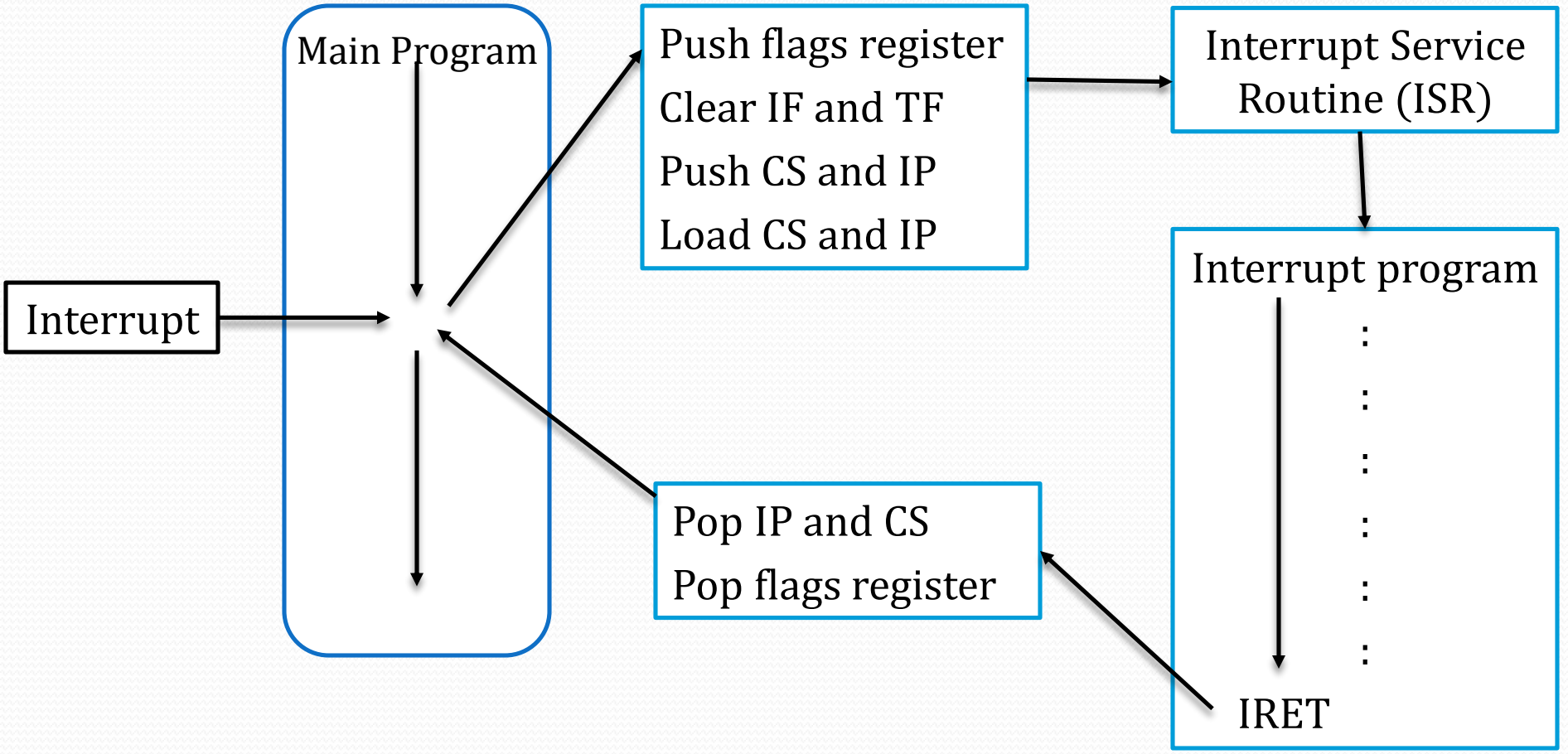
If an interrupt has been requested, the 8086 Microprocessor processes it by performing the following series of steps:

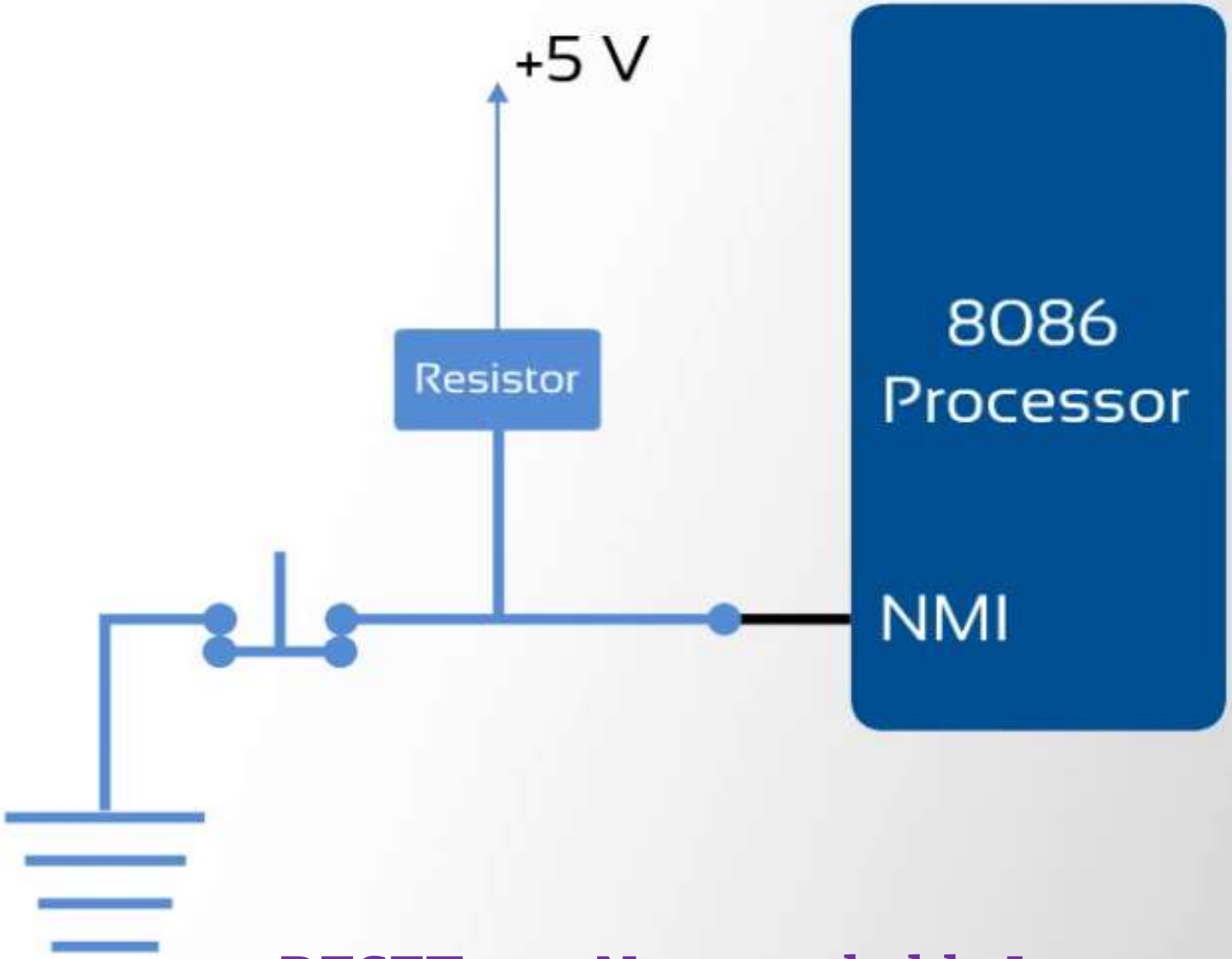
1. Pushes the content of the flag register onto the stack to preserve the status of the interrupt (IF) and trap flags (TF), by decrementing the stack pointer (SP) by 2
2. Disables the INTR interrupt by clearing IF in the flag register
3. Resets TF in the flag register, to disable the single step or trap interrupt
4. Pushes the content of the code segment (CS) register onto the stack by decrementing SP by 2
5. Pushes the content of the instruction pointer (IP) onto the stack by decrementing SP by 2
6. Performs an indirect far jump to the start of the interrupt service routine (ISR) corresponding to the received interrupt.

Steps involved in processing an interrupt instruction by the processor

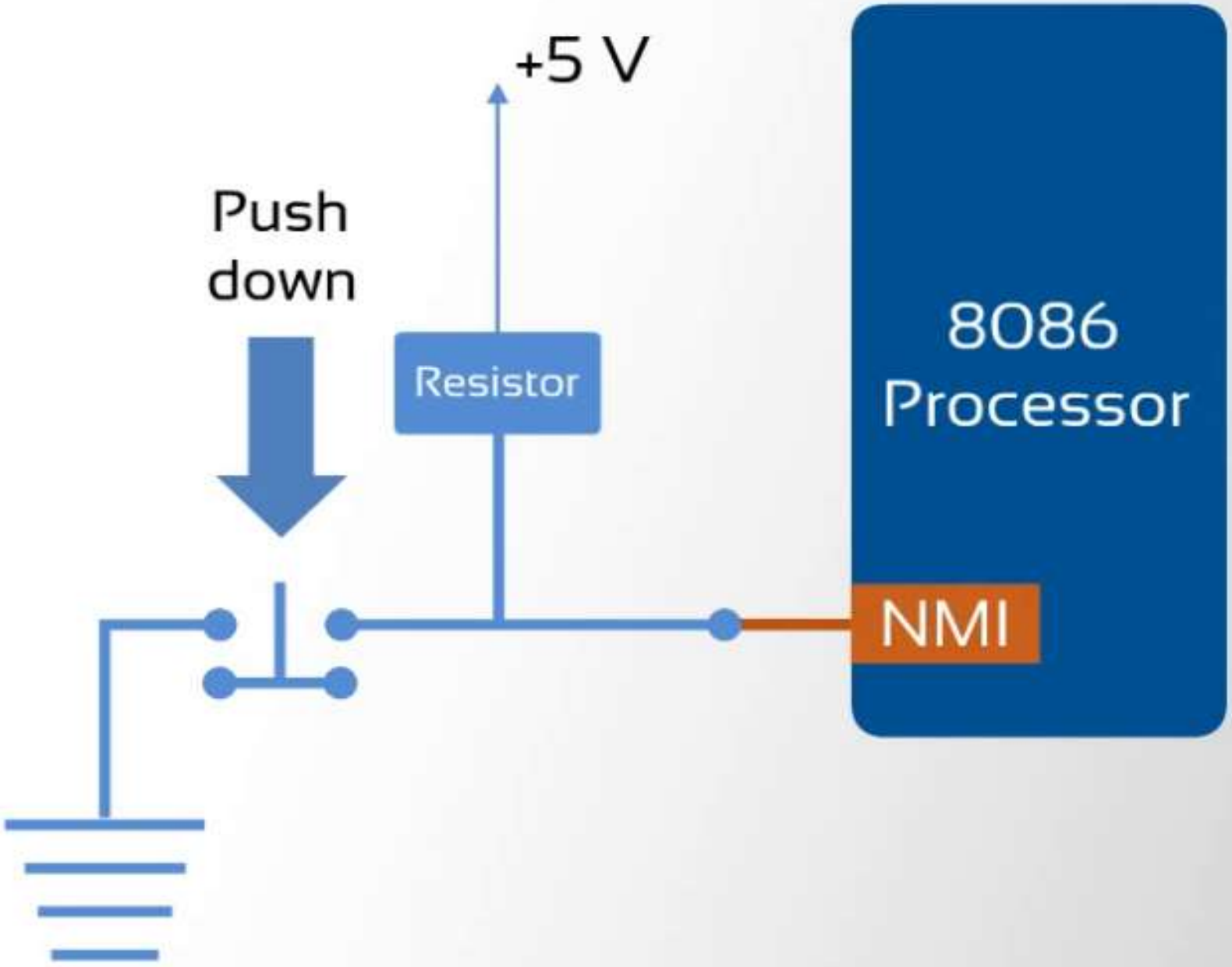


Processing of an Interrupt by the 8086





RESET as a Non-maskable Interrupts


+5 V
Push
down
Resistor
8086
Processor
NMI

Process sequence in the processor

Completes the current instruction that is in progress

↓

Pushes the Flag Register values on to the Stack

↓

Pushes the CS value and IP value of the return address on to the stack

↓

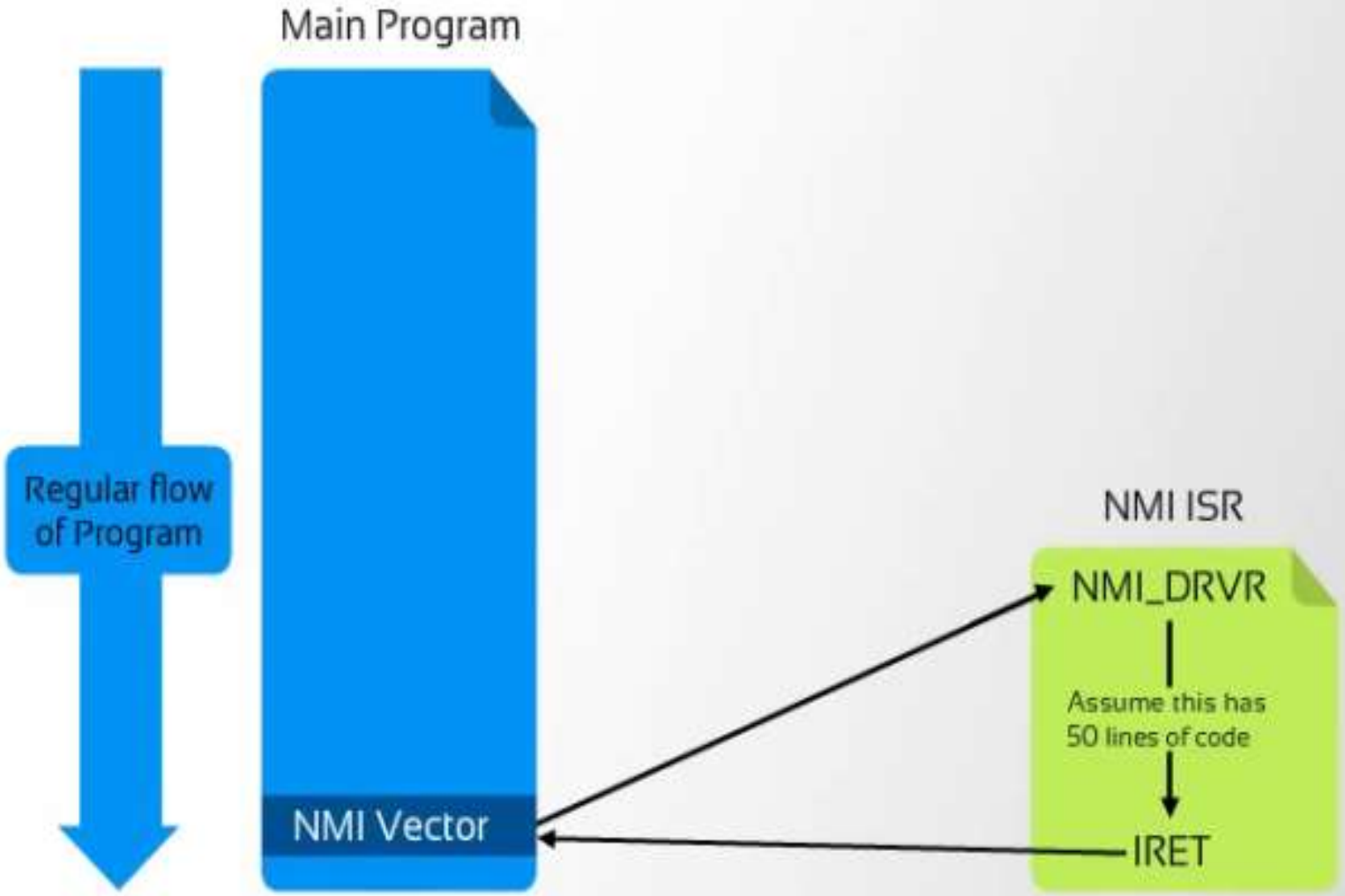
IP is loaded from contents of the word location 00008H

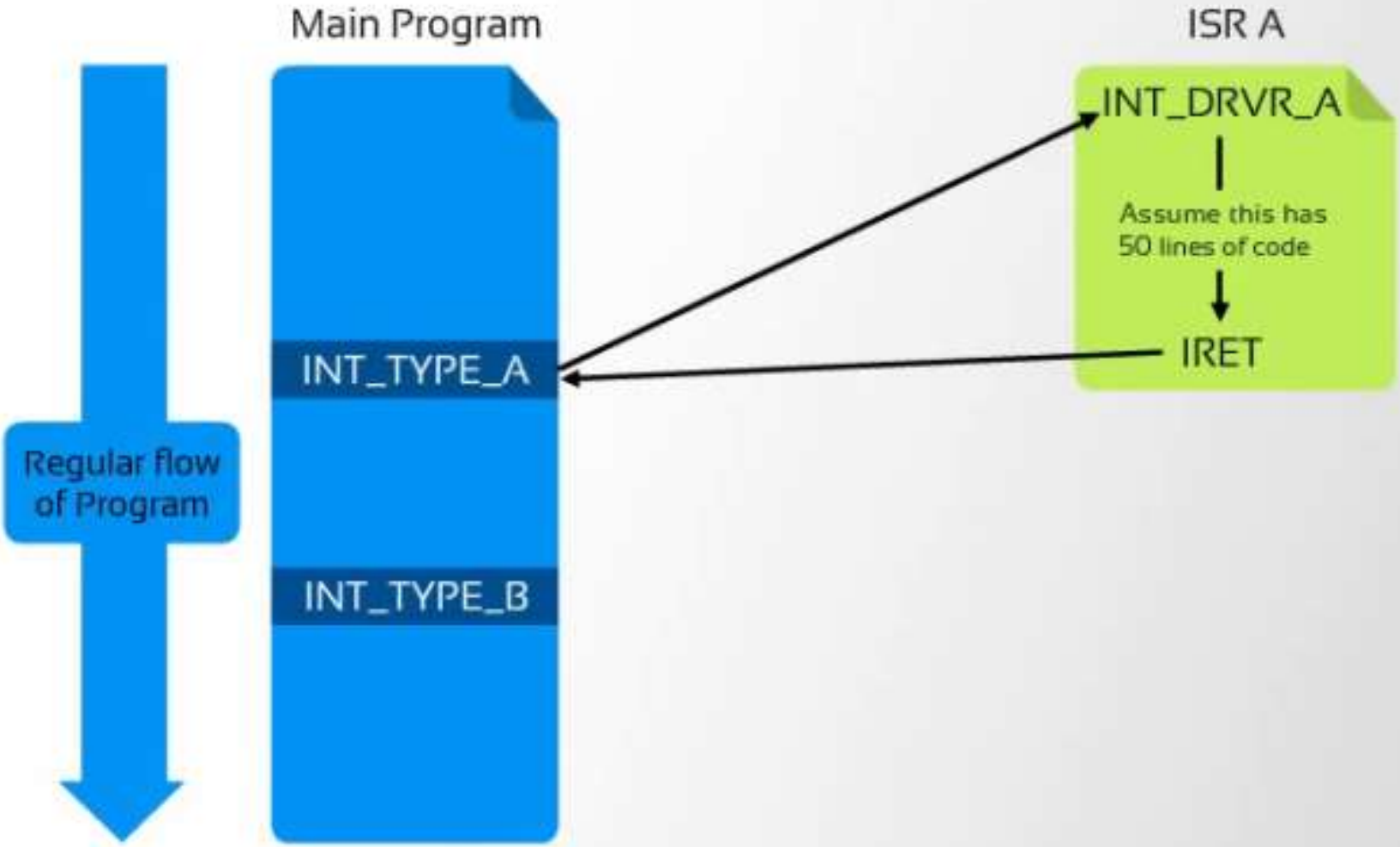
↓

CS is loaded from contents of next word location 0000AH

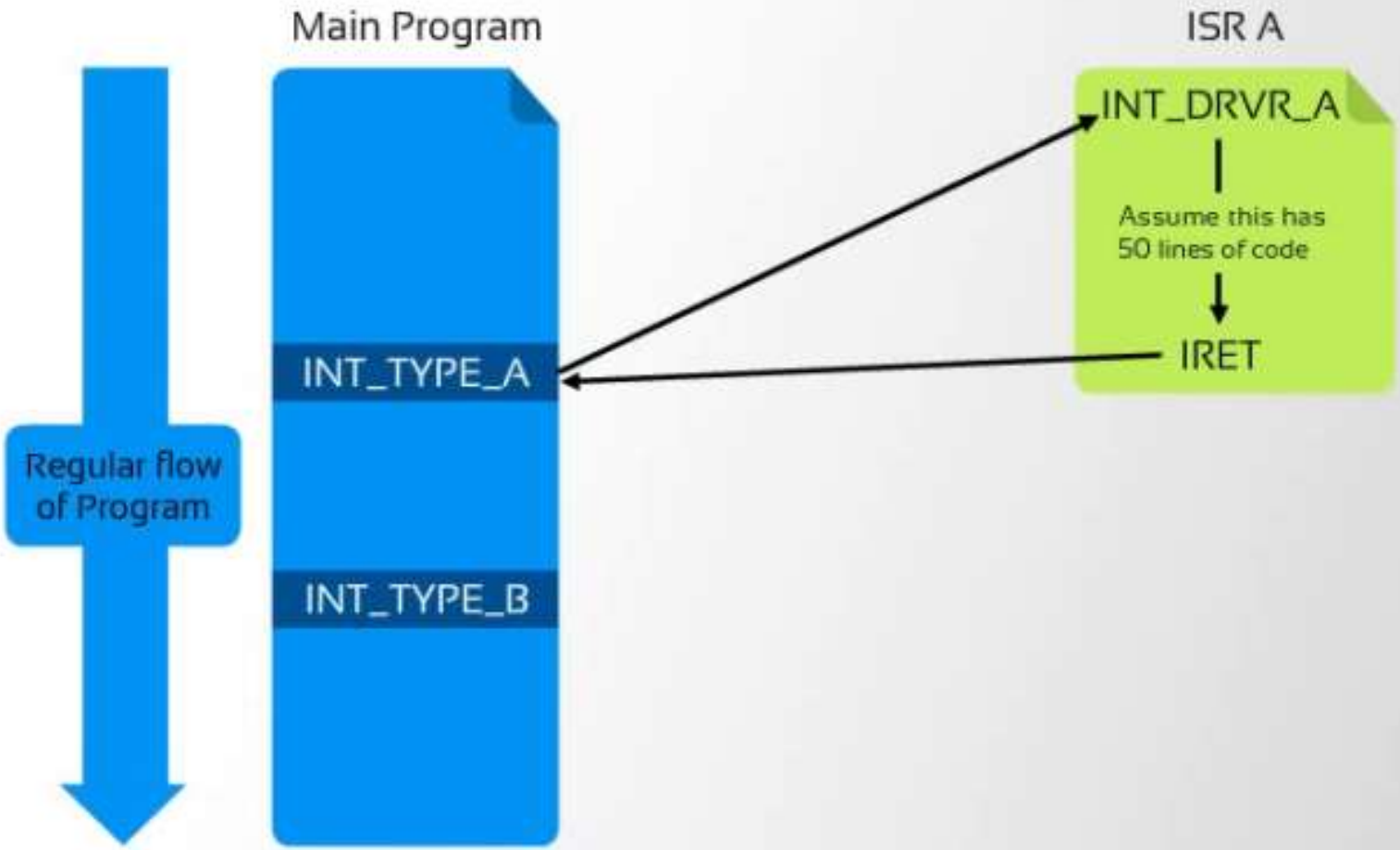
↓

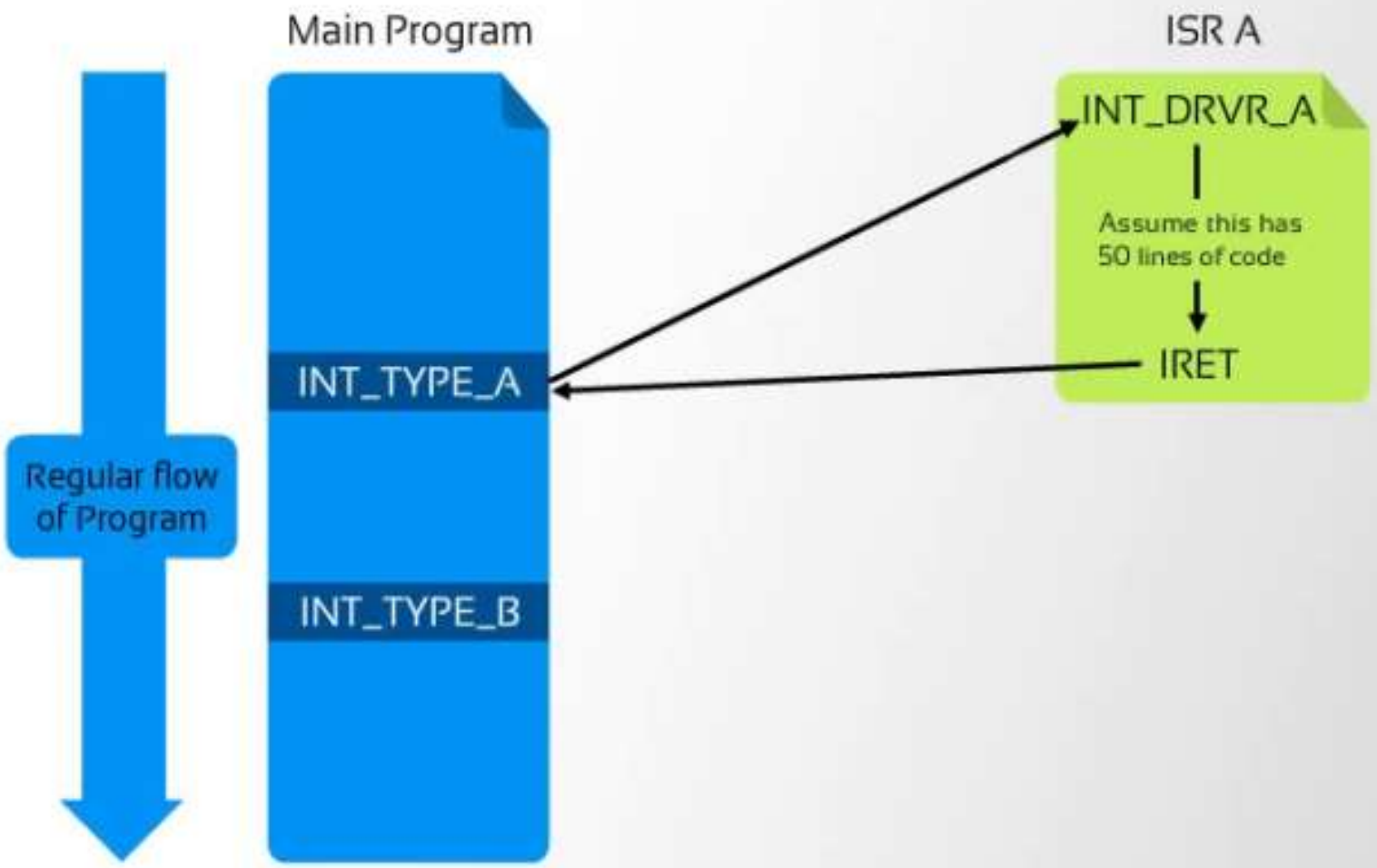
Interrupt Flag and Trap Flag are reset to 0.


Main Program
Regular flow of Program
NMI Vector
NMI ISR
NMI_DRVR
Assume this has 50 lines of code
IRET

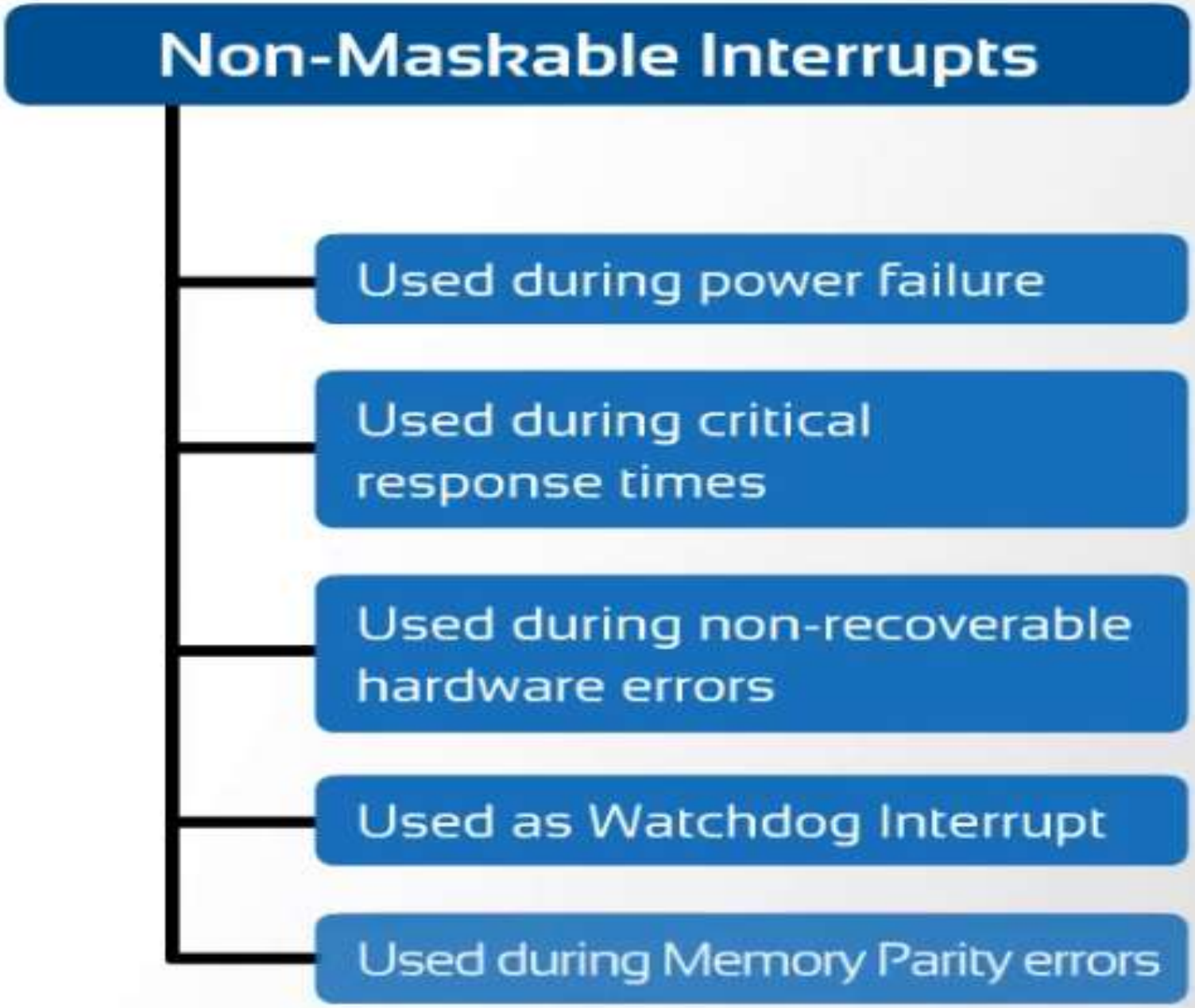

Main Program
ISR A
INT_DRVR_A
Assume this has
50 lines of code
IRET
INT_TYPE_A
INT_TYPE_B
Regular flow
of Program

The processor pushes the Flag, the CS, and the IP before executing the ISR.





Interrupt Return Instruction


Non-Maskable Interrupts
Used during power failure
Used during critical response times
Used during non-recoverable hardware errors
Used as Watchdog Interrupt
Used during Memory Parity errors

Software Interrupts

- Used by Operating Systems to provide hooks into various functions
- Used as a communication mechanism between different parts of a program

Hardware Interrupts

Used to handle external hardware peripherals, such as keyboards, mouse, hard disks, floppy disks, DVD drives, and printers

Interrupts are expensive in terms of time and processing power.

Hardware Interrupts

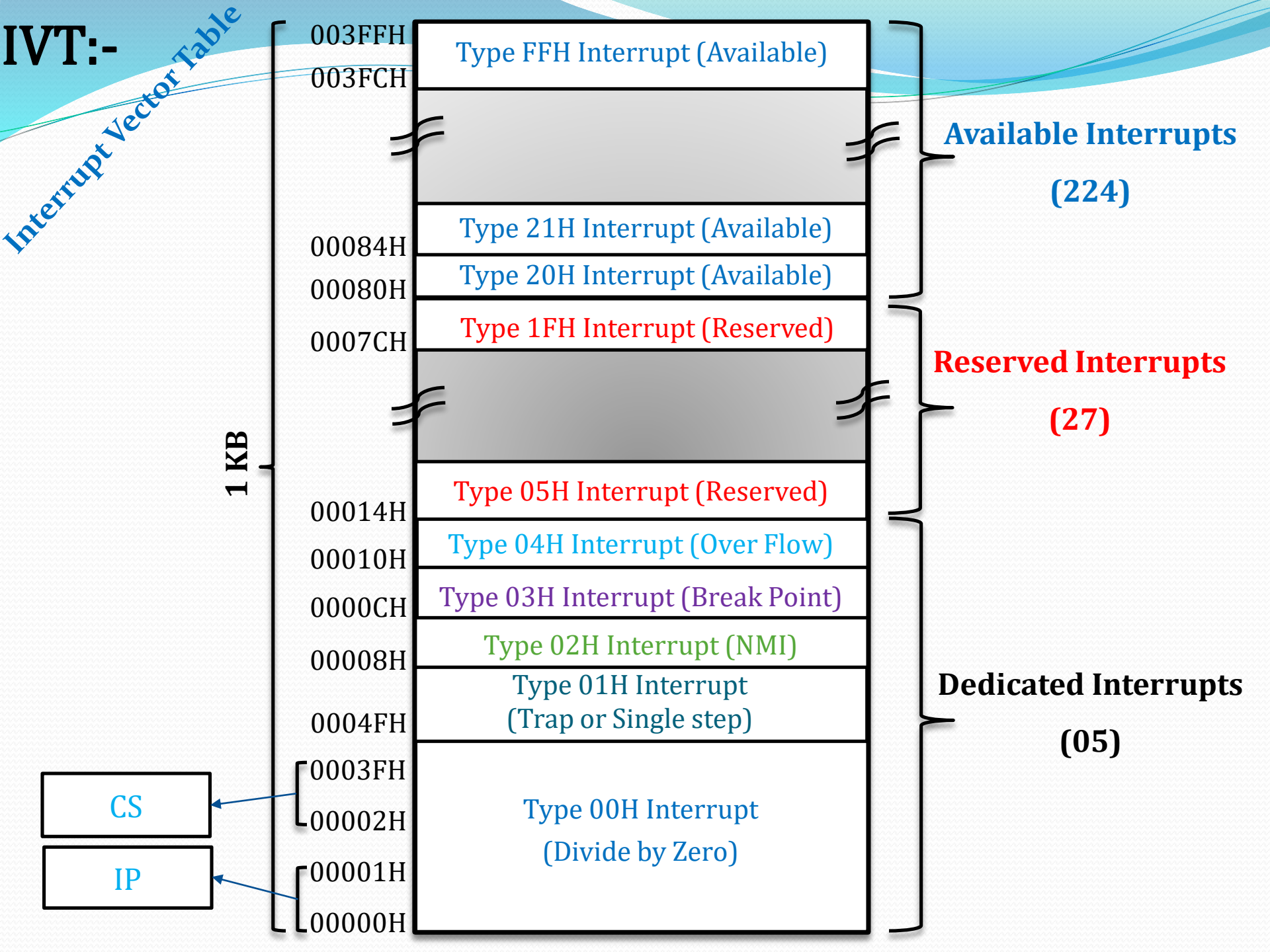
Used to handle external hardware peripherals, such as keyboards, mouse, hard disks, floppy disks, DVD drives, and printers

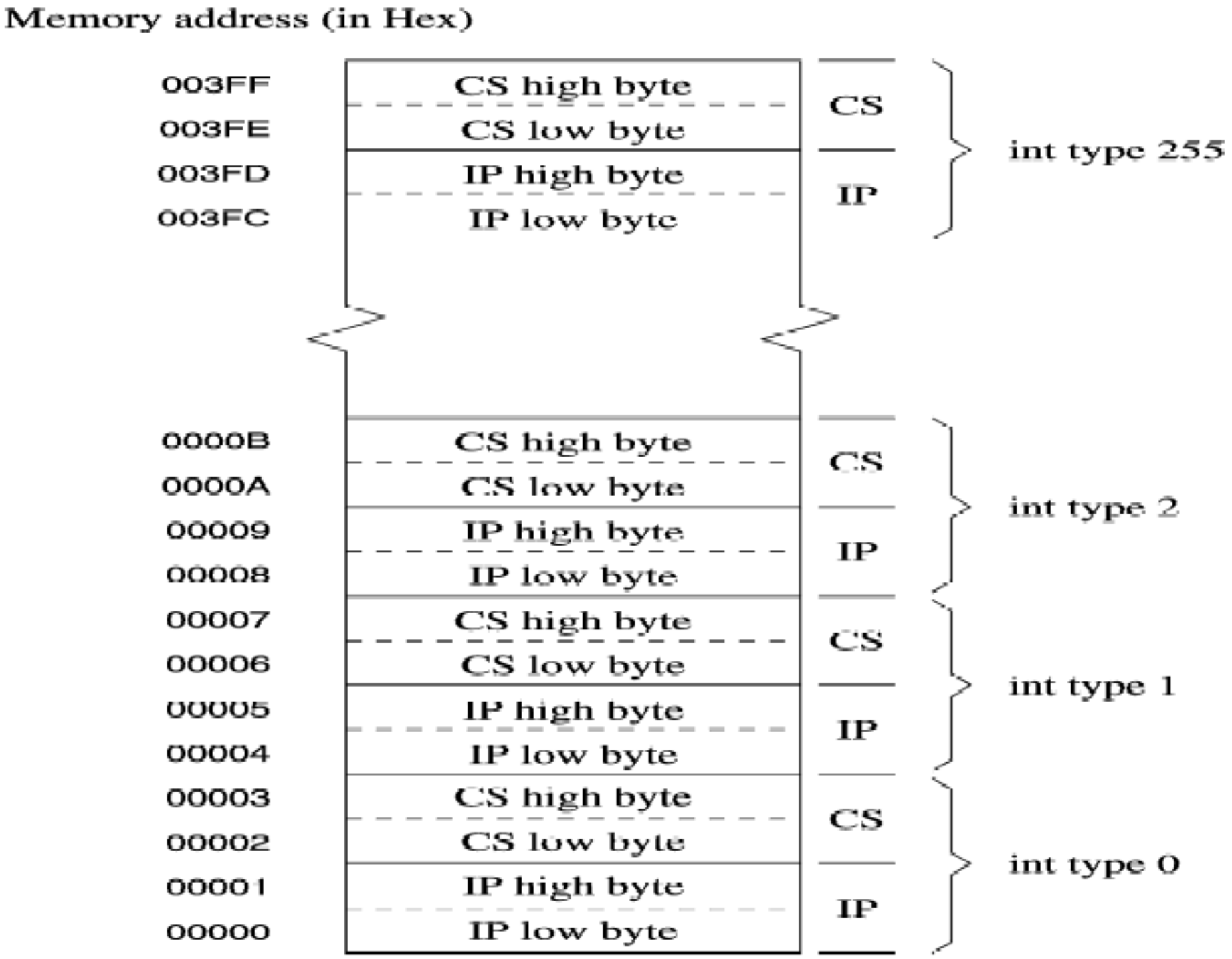
Without Interrupts, the systems are very simple.

All microprocessor architectures have in-built interrupt service capability.

IVT:-

Interrupt Vector Table




Memory address (in Hex)
003FF
003FE
003FD
003FC
CS high byte
CS low byte
IP high byte
IP low byte
CS
IP
int type 255
0000B
0000A
00009
00008
00007
00006
00005
00004
00003
00002
00001
00000
int type 2
int type 1
int type 0

256 Interrupts Of 8086 are Divided in To 3 Groups

1. **Type 00 to Type 04 interrupts-**
   These are used for fixed operations and hence are called dedicated interrupts

2. **Type 05 to Type 31 interrupts**
   Not used by 8086,reserved for higher processors like 80286 80386 etc.

3. **Type 32 to Type 255 interrupts**
   Available for user, called user defined interrupts these can be H/W interrupts and activated through INTR line or can be S/W interrupts.

- **Type – 0 :- Divide by Zero Error Interrupt**

  Quotient is large cant be fit in al/ax or divide by zero

- **Type –1:- Single step or Trap Interrupt**

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

- **Type – 2:- Non-Maskable Interrupt**

  This interrupt is used for executing ISR of NMI pin (positive edge signal), NMI can't be masked by S/W.

- **Type – 3:- One-byte INT instruction interrupt**

  Used for providing break points in the program

- **Type – 4 Over flow Interrupt**

  Used to handle any overflow error after signed arithmetic.

Thank You