Lecture 11

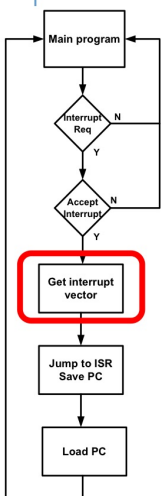
Interrupts are similar to exceptions

ISR: Interrupt Service Routine

ISP: Interrupt service procedure

German Universit

INTERRUPT VECTORS



- To find the interrupt vector for interrupt number *n*, we just need to multiply *n* by 4.
- EXAMPLE: Interrupt 9
  - **For INT 9:** → Dec
  - $9 \times 4 = (36)_{10} = 0024H$, The physical addresses **00024H, 00025H,00026H,00027H** hold the CS:IP for INT 9 Interrupt Service Routine (ISR).
  - The logical address is **0000:0024H – 0000:0027H**

Cs 0000
+ 0024
As 0024

INT → Push F
For Call → to get Address of ISR

IRET same as → ① Pop F
② ret

- When IF=0 → all hardware interrupt requests through the **INTR** are ignored.
  - The IF flag bit has no effect on the **NMI pin** or the "**INT nn**"

IF=0 [Ignore any Interrupt]
IF=1

For Software Interrupts: They are Called only using INT

Hardware Interrupt: Triggred by external events

INTRO TO INTERRUPTS

- An interrupt is a **hardware-generated CALL** externally derived from a hardware signal
- Or a **software-generated CALL** internally derived from the execution of an instruction or by some other internal event
  - at times an internal interrupt is called an *exception*
- Either type interrupts the program by calling an **interrupt service procedure** (ISP) or interrupt handler.

WHAT Is an Interrupt Service Routine (ISR) ?

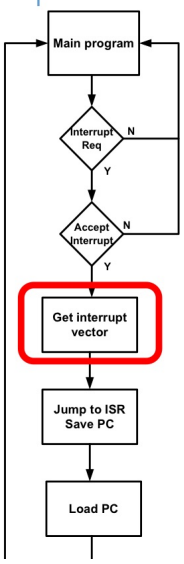
- A program associated with an interrupt.
- When an interrupt is invoked → it is asked to run a program to perform a certain service. → a Certain procedure
- This program is known as the ***Interrupt Service Routine*** or ***Interrupt handler***.

Interrupts is a mechanism that allows CPU to stop it's normal execution flow and immediately execute some other code (ISR)

Key Differences:

• ISR is a more common term used in low-level programming to refer to the function that directly handles an interrupt.
• ISP could be used in some contexts to describe a similar concept but may be used in more formal or high-level system frameworks. The term might be less commonly used compared to ISR.
2. Contextual Use:
• ISR is used to describe the routine that immediately responds to and handles an interrupt, typically with minimal delay.
• ISP might refer to the broader concept of interrupt handling within a system, but often it's used interchangeably with ISR.

INTERRUPT VECTORS



- To find the interrupt vector for interrupt number *n*, we just need to multiply *n* by 4.
- EXAMPLE: Interrupt 9
  - **For INT 9:** → Dec
  - $9 \times 4 = (36)_{10} = 0024H$, The physical addresses **00024H, 00025H,00026H,00027H** hold the CS:IP for INT 9 Interrupt Service Routine (ISR).
  - The logical address is **0000:0024H – 0000:0027H**

As

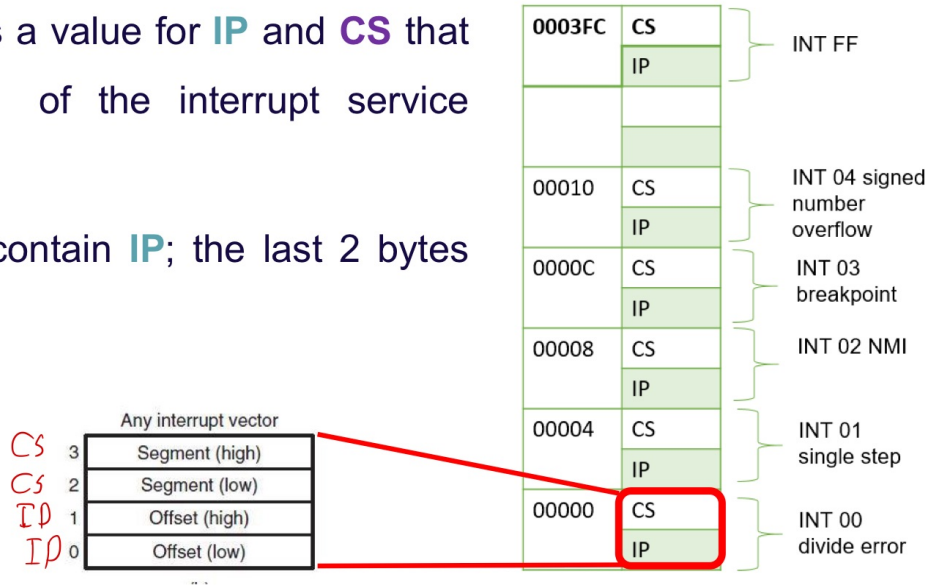
Cs 0000 + 0024 = 0024

(36, 37, 38, 39) decimal

INTERRUPT VECTORS

Interrupt vector tables

- Each vector contains a value for **IP** and **CS** that forms the address of the interrupt service procedure.
  - the first 2 bytes contain **IP**; the last 2 bytes **CS**.



INTERRUPT INSTRUCTIONS

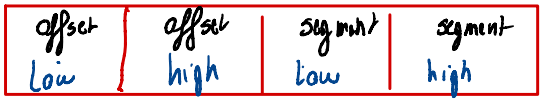
- Three different interrupt instructions are available:
  - INT, INTO, and INT 3 → Breakpoint (INTO → int overflow)
- In real mode, each fetches a vector from the vector table, and then calls the procedure stored at the location addressed by the vector.
- Similar to a **far CALL** instruction because it places the return address (IP/EIP and CS) on the stack.

Explanation (No need to go through)

INTERRUPT VECTORS

- A 4-byte number stored in the first 1024 bytes of memory (00000H–003FFH) in real mode.
- 256 different interrupt vectors.
  - each vector contains the address of an **interrupt service procedure**.

INT Number	Physical Address	Logical Address
INT 00	00000	0000:0000
INT 01	00004	0000:0004
INT 02	00008	0000:0008
INT 03	0000C	0000:000C
.....	.....	.....
.....	.....	.....
INT FF	003FC	0000:03FC



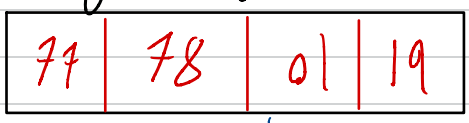
The address that is stored is the address of Interrupt procedure

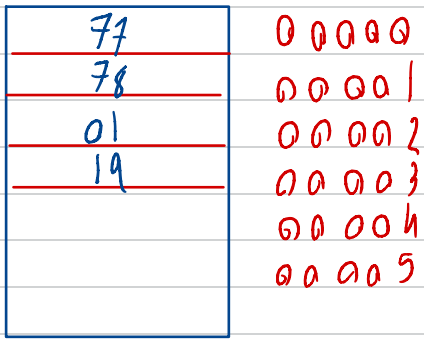
So The IVT (Interrupt vector table) has The address of each interrupt procedure store at ex

INT 00 → 00000 → This Contains The Address of ISP

INT 01 → 00004

Let's say Address of ISP of INT 00 is





So the first 1024 bytes of Memory Store the Addresses of ISP

each 4 byte Contain a single address

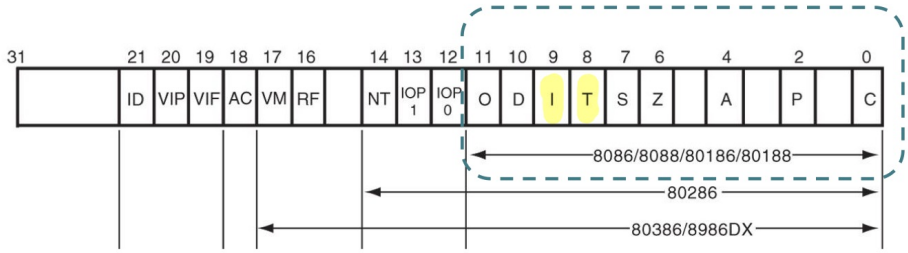
So if I want to get Address of INT 05

→ go to IVT 5 * 4 = 20 (change to Hex)

4 bytes

14h This is the start of the Address in IVT

FLAGS



- ***I (interrupt):*** control the operation of the INTR (interrupt request) input pin.
  - 0 disables INTR pin, 1 enables INTR pin.

IF=0
IF=1

- ***T (trap):*** enables trapping through an on-chip debugging feature.

single step

  - 0 disable trapping, 1 enable trapping.
  - If enabled, allows the microprocessor to interrupt the flow of the program on conditions indicated by the debug registers and control registers.
  - **Microsoft Visual Studio debugging tool uses this feature.**

Software Interrupts

Flag high
Flag low
Cs high
Cs low
IP high
IP low

- **When a software interrupt executes**, it:

SP →

  - pushes the flag register (FR) onto the stack and SP is decremented by 2.

Why?

SP 16 bits 2Bytes

  - clears the TF (Trap flag) and IF (Interrupt flag) bits.

Why?

So the program can't accept another interrupt

  - pushes current CS onto the stack and SP is decremented by 2.
  - pushes the current IP/EIP onto the stack and SP is decremented by 2.
  - The INT number (type) is multiplied by 4 in order to calculate the physical address of the locations that holds the CS and IP of the ISR

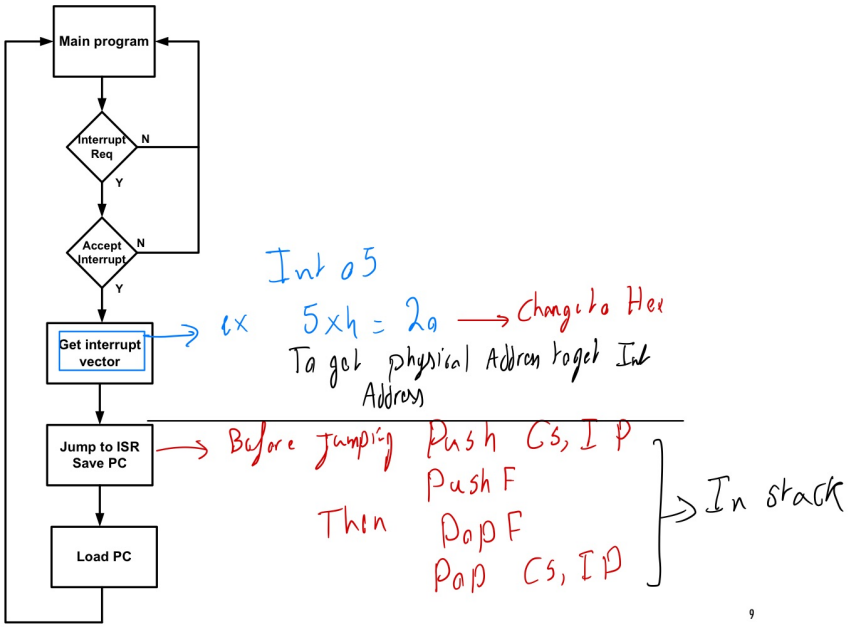
20

  - fetches the new value for CS from the interrupt vector.
  - fetches the new value for IP/EIP from the vector.
  - jumps to the new location addressed by CS and IP/EIP and the CPU starts to fetch and execute instructions belonging to the ISR program.
  - IRET is the last instruction of the Interrupt service routine in order to get the IP, CS and FR back from the stack to allow the CPU to run the code where it left off.

IRET same as → ① popf ② ret

Software Interrupts: They are called only using INT

Hardware Interrupt: Triggred by external events



Interrupt Processing Flow:

1)Interrupt Occurs:

An interrupt (e.g., from hardware or software) happens while the program is running.

2)Check the Interrupt Flag (IF):

If IF = 1 (interrupts are enabled), the CPU will process the interrupt.
If IF = 0 (interrupts are disabled), the CPU will ignore the interrupt, unless it's a Non-Maskable Interrupt (NMI), which always gets processed.

3)Save Current Program State:

When the CPU accepts the interrupt, it saves the current state (IP, CS, and Flags) on the stack so it can return later.

4)Disable Interrupts (IF = 0):

The CPU clears the IF flag to prevent further interrupts from happening while it's handling the current one.

5)Look Up the Interrupt Service Routine (ISR):

The CPU checks the Interrupt Vector Table (IVT) to find out where the corresponding ISR (Interrupt Service Routine) is located.
The IVT has the address (segment and offset) of the ISR for each interrupt type.

6)Jump to ISR:

The CPU jumps to the ISR address found in the IVT and starts executing the ISR to handle the interrupt.

7)Execute the ISR:

The ISR performs the necessary task (e.g., handling data, clearing interrupt flags) to respond to the interrupt.

8)Return from Interrupt:

The CPU uses the IRET instruction to restore the saved state (IP, CS, Flags) from the stack.
The CPU also restores the IF flag, re-enabling interrupts if needed.

9)Continue Program Execution:

The program continues from where it left off, right after the instruction that was interrupted.

Key Points:

IF flag: Controls whether interrupts are allowed. IF = 1 means interrupts are enabled, and IF = 0 means interrupts are disabled (except NMI).

Interrupt Vector Table (IVT): Stores addresses of ISRs for each interrupt type.

The CPU saves the state, executes the ISR, and then restores the state to continue the program.

- **INT** performs as a **far CALL**
  - not only pushes CS & IP onto the stack, also pushes the flags onto the stack.
- The **INT** instruction performs the operation of a ***PUSHF***, followed by a **far CALL** instruction.
- The interrupts often control printers, video displays, and disk drives.

INT ⟶ Push F
For call → to get Address of ISR

- **INT nn** instruction is 2 bytes long (the first byte is for the **opcode** while the second byte is for the **interrupt number**), whereas the **far CALL** is 5 bytes long.
- Each time that the **INT** instruction replaces a **far CALL**, it saves 3 bytes of memory.
- This can amount to a sizable saving if INT often appears in a program, as it does for system calls.

IRET

- Used only with software or hardware interrupt service procedures.
- IRET instruction will
  - pop stack data back into the IP
  - pop stack data back into CS
  - pop stack data back into the flag register
- Accomplishes the same tasks as the **POPF** followed by a **far RET** instruction.

for Both Hardware & Software

Interrupt Occurs:

An interrupt (e.g., from hardware or software) happens while the program is running:

Hardware interrupts are triggered by external events (e.g., I/O devices, timers, etc.).
Software interrupts are triggered by the program itself using the INT instruction (e.g., for system calls, debugging, or invoking specific functions).

Check the Interrupt Flag (IF):

The CPU checks the Interrupt Flag (IF) in the Flags Register to determine whether it should process the interrupt:
IF = 1: Interrupts are enabled, and the CPU will process the interrupt, whether it is hardware or software.
IF = 0: Interrupts are disabled, and the CPU will ignore the interrupt, unless it's a Non-Maskable Interrupt (NMI), which cannot be ignored and is always processed.

Save Current Program State:

When the CPU accepts the interrupt, it saves the current state (such as IP, CS, and Flags) onto the stack to ensure the program can return to the same point after handling the interrupt.

Disable Interrupts (IF = 0):

The CPU clears the IF flag to prevent further interrupts from happening while it is handling the current interrupt. This is done to avoid nested interrupts.

Look Up the Interrupt Service Routine (ISR):

The CPU retrieves the corresponding Interrupt Service Routine (ISR) for the interrupt by consulting the Interrupt Vector Table (IVT):
For hardware interrupts, the IVT entry for the specific interrupt type will contain the address of the ISR.
For software interrupts, the program specifies the interrupt number when issuing the INT instruction, and the CPU uses this to look up the ISR in the IVT.

Jump to ISR:

The CPU loads the segment and offset of the ISR from the IVT and jumps to the ISR to begin processing the interrupt.
For hardware interrupts, this involves handling the interrupt triggered by an external device (e.g., reading data from an I/O port).
For software interrupts, this could involve executing a system call or invoking a specific routine (e.g., for handling an exception or calling an operating system function).

Execute the ISR:

The ISR executes and performs the necessary task to handle the interrupt:
For hardware interrupts, this may involve interacting with hardware, handling I/O, or clearing interrupt flags from the hardware device.
For software interrupts, this may involve performing operations like system calls, exception handling, or debugging tasks.

Return from Interrupt:

The CPU uses the IRET (Interrupt Return) instruction to restore the saved state from the stack (including IP, CS, and Flags registers).
The CPU also restores the IF flag to its original state, re-enabling interrupts if IF = 1 was set before the interrupt occurred.

Continue Program Execution:

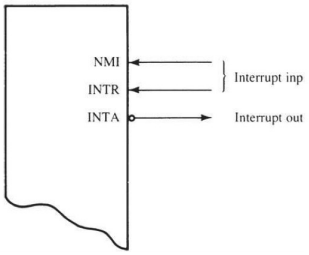
The program resumes execution at the point immediately after the instruction that was interrupted, using the values restored by the IRET instruction.

For Hardware Interrupts

HARDWARE INTERRUPT

- Allows hardware devices to interrupt the operation of the CPU.
  - For example, when a keyboard key is pressed, the CPU must be interrupted to read the key code in the keyboard buffer.
- The Intel processor has three pins that are related to hardware interrupts:
  - **INTR (Interrupt Request)**
  - **NMI (non-maskable interrupt)**
  - **INTA (Interrupt Acknowledge)**

ex: shut down restart



27

INTERRUPT CONTROL

- Two instructions control the INTR pin.
- The **set interrupt flag** instruction (STI) places 1 in the **I flag bit**.
  - which enables the INTR pin
- The **clear interrupt flag** instruction (CLI) places a 0 into the **I flag bit**.
  - which disables the INTR pin
- The STI instruction enables INTR and the CLI instruction disables INTR.

instruction

Interrupt flag

Set

INTR (Interrupt Request Pin):

Triggered by external devices to request an interrupt.
The CPU will process it if the IF flag is 1 (interrupts enabled). If IF = 0, interrupts are ignored, unless it's an NMI.

NMI (Non-Maskable Interrupt Pin):

Used for urgent, critical interrupts (e.g., hardware errors) that cannot be ignored, regardless of the IF flag.
The CPU will always respond to NMI.

INTA (Interrupt Acknowledge Pin):

Used by the CPU to acknowledge an interrupt request.
The CPU triggers this pin after it accepts an interrupt, allowing the interrupting device to provide the interrupt vector or additional data.