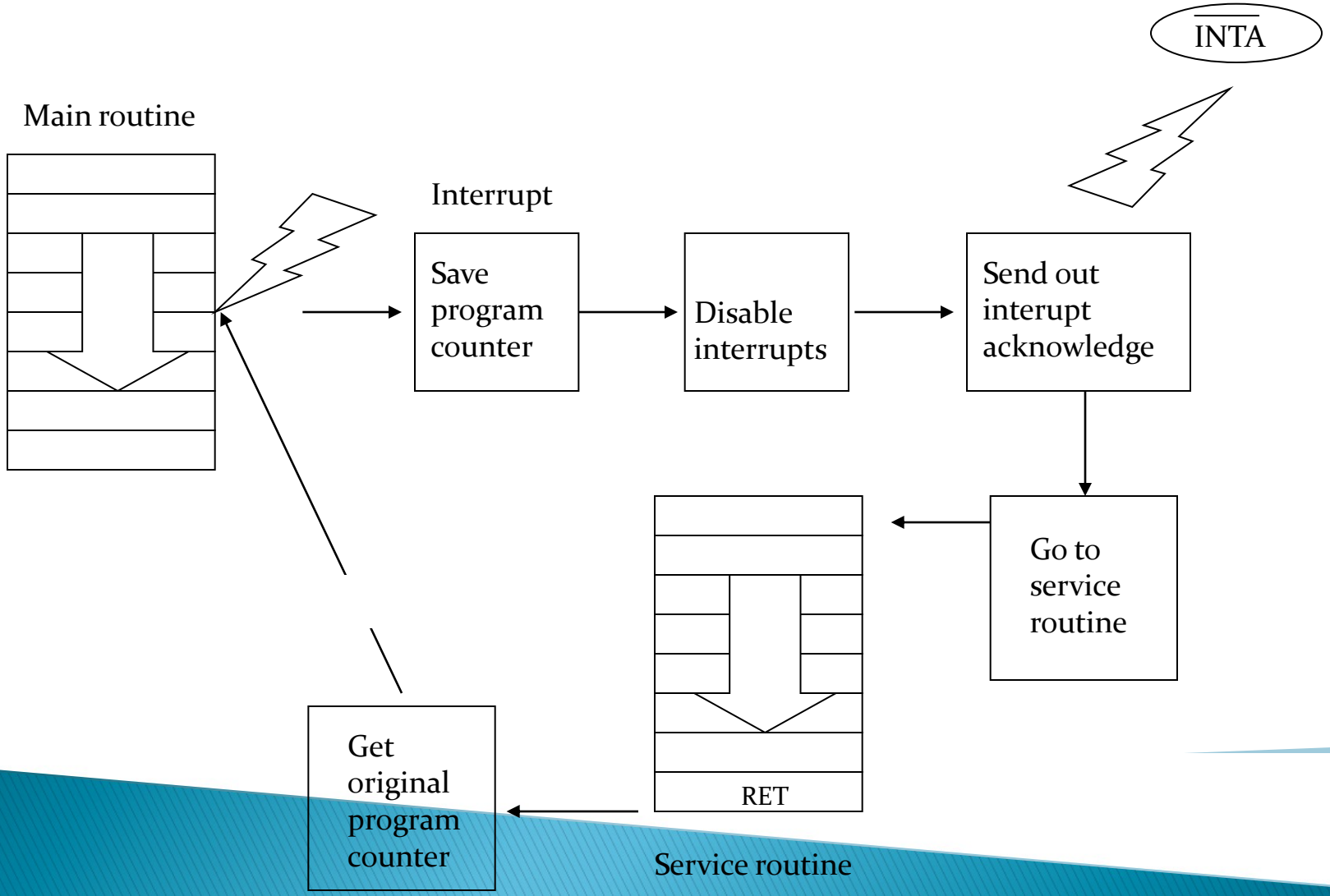
Unit 3

LECTURE 5

Interrupts

- An interrupt is considered to be an emergency signal.
  - The Microprocessor should respond to it as soon as possible.

- 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 most probably have its own ISR.


INTA
Main routine
Interrupt
Save program counter
Disable interrupts
Send out interupt acknowledge
Go to service routine
RET
Service routine
Get original program counter

Responding to Interrupts

- Responding to an interrupt may be immediate or delayed depending on whether the interrupt is maskable or non-maskable and whether interrupts are being masked or not.

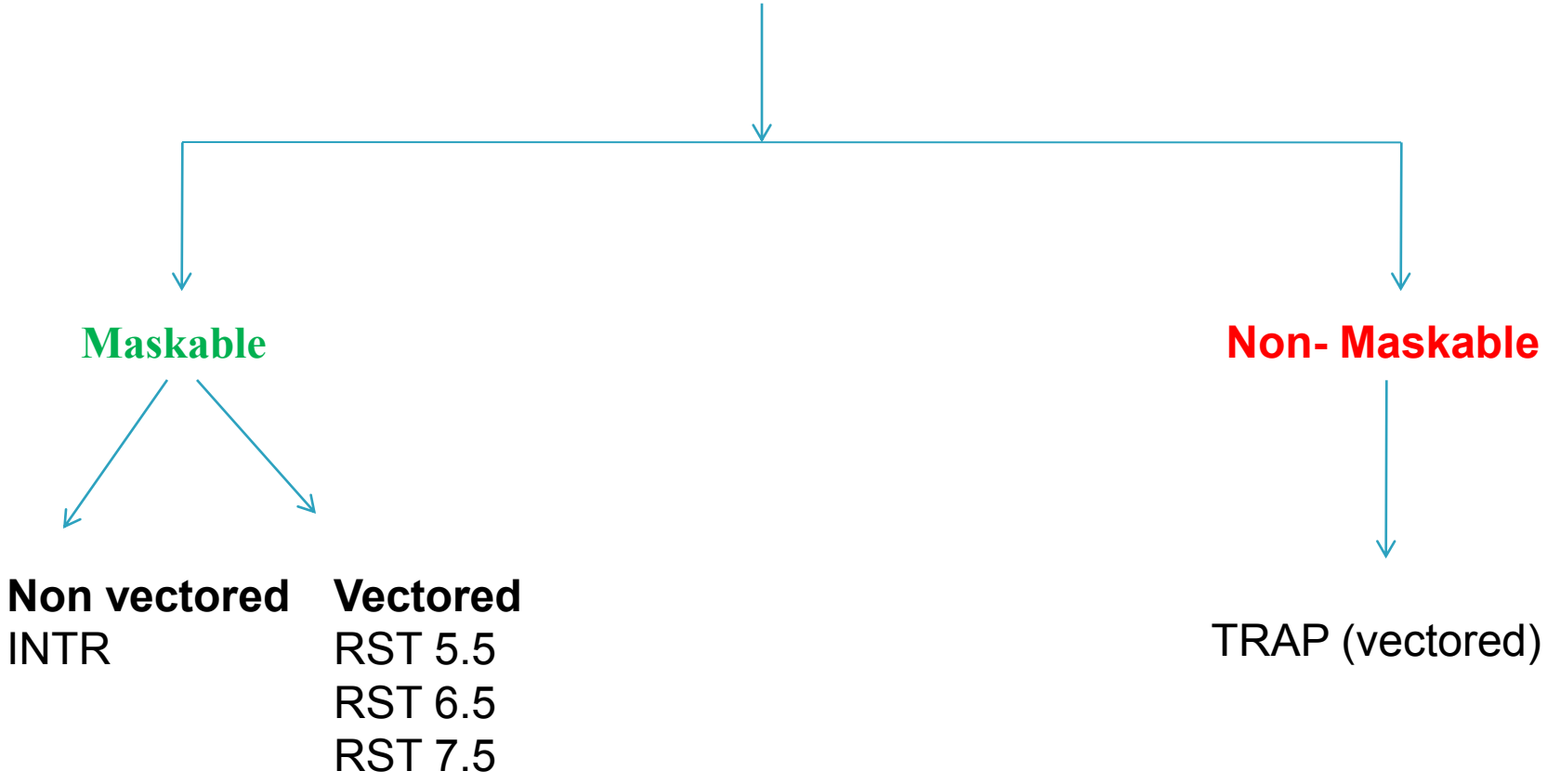
- There are two ways of redirecting the execution to the ISR depending on whether the interrupt is vectored or non-vectored.
  - The vector is already known to the Microprocessor
  - The device will have to supply the vector to the Microprocessor

The 8085 Interrupts

- The maskable interrupt process in the 8085 is controlled by a single flip flop inside the microprocessor. This Interrupt Enable flip flop is controlled using the two instructions "EI" and "DI".

- The 8085 has a single Non-Maskable interrupt.
  - The non-maskable interrupt is not affected by the value of the Interrupt Enable flip flop.

Types of interrupts



The 8085 Interrupts

- The 8085 has 5 interrupt inputs.
  - The INTR input.
    - The INTR input is the only non-vectored interrupt.
    - INTR is maskable using the EI/DI instruction pair.
  - RST 5.5, RST 6.5, RST 7.5 are all automatically vectored.
    - RST 5.5, RST 6.5, and RST 7.5 are all maskable.
  - TRAP is the only non-maskable interrupt in the 8085
    - TRAP is also automatically vectored

The 8085 Interrupts

Interrupt name	Maskable	Vectored
INTR	Yes	No
RST 5.5	Yes	Yes
RST 6.5	Yes	Yes
RST 7.5	Yes	Yes
TRAP	No	Yes

Interrupt Vectors and the Vector Table

- An interrupt vector is a pointer to where the ISR is stored in memory.
- All interrupts (vectored or otherwise) are mapped onto a memory area called the Interrupt Vector Table (IVT).
  - The IVT is usually located in memory page 00 (0000H - 00FFH).
  - The purpose of the IVT is to hold the vectors that redirect the microprocessor to the right place when an interrupt arrives.
  - The IVT is divided into several blocks. Each block is used by one of the interrupts to hold its "vector"

Maskable/Non vectored Interrupt Process

1. The interrupt process should be enabled using the EI instruction.
2. The 8085 checks for an INTR line (interrupt signal/pin 10) during the execution of every instruction.
3. If there is an interrupt(INTR is high) , and if the interrupt is enabled, the microprocessor will complete the executing instruction, and reset/disables the interrupt flip flop and sends signal INTA' (interrupt acknowledge active low signal).
4. The microprocessor then executes a RST restart instruction (or a call instruction) through external hardware.RST sends the execution to the appropriate location on page 00H in the interrupt vector table.
5. When the microprocessor executes the call instruction, it saves the address of the next instruction on the stack.
6. The microprocessor jumps to the specific service routine and performs the task.
7. The service routine must include the instruction EI to re-enable the interrupt process.
8. At the end of the service routine, the RET instruction returns the execution to where the program was interrupted.

The 8085 Non-Vectored Interrupt Process

- The 8085 recognizes 8 RESTART instructions: RST0 - RST7.
  - each of these would send the execution to a predetermined hard-wired memory location:

Restart Instruction	Equivalent to
RST0	CALL 0000H
RST1	CALL 0008H
RST2	CALL 0010H
RST3	CALL 0018H
RST4	CALL 0020H
RST5	CALL 0028H
RST6	CALL 0030H
RST7	CALL 0038H

Restart Sequence

- The restart sequence is made up of three machine cycles
  - In the 1st machine cycle:
    - The microprocessor sends the INTA signal.
    - While INTA is active the microprocessor reads the data lines expecting to receive, from the interrupting device, the opcode for the specific RST instruction.
  - In the 2nd and 3rd machine cycles:
    - the 16-bit address of the next instruction is saved on the stack.
    - Then the microprocessor jumps to the address associated with the specified RST instruction.

Restart Sequence

- The location in the IVT associated with the RST instruction can not hold the complete service routine.
  - The routine is written somewhere else in memory.
  - Only a JUMP instruction to the ISR's location is kept in the IVT block.

Hardware Generation of RST Opcode

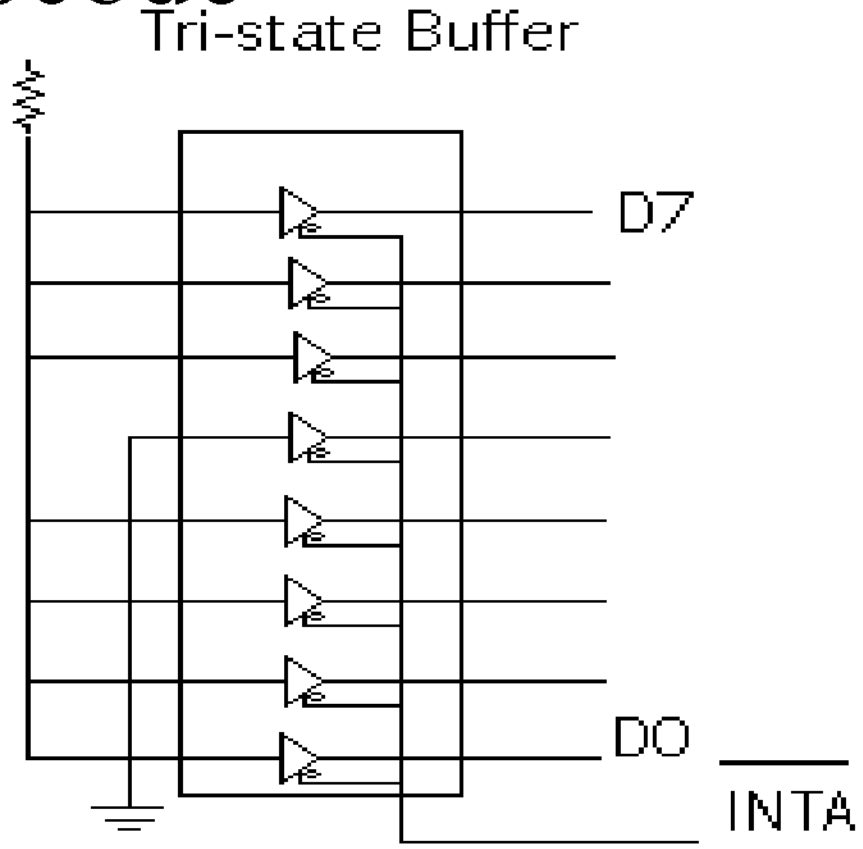
- How does the external device produce the opcode for the appropriate RST instruction?
  - The opcode is simply a collection of bits.
  - So, the device needs to set the bits of the data bus to the appropriate value in response to an INTA signal.

Hardware Generation of RST Opcode

The following is an example of generating RST 5:

RST 5's opcode is EF =

D D
76543210
11101111



Hardware Generation of RST Opcode

- During the interrupt acknowledge machine cycle, (the 1st machine cycle of the RST operation):
  - The Microprocessor activates the INTA signal.
  - This signal will enable the Tri-state buffers, which will place the value EFH on the data bus.
  - Therefore, sending the Microprocessor the RST 5 instruction.

- The RST 5 instruction is exactly equivalent to CALL 0028H

Multiple Interrupts & Priorities

- How do we allow multiple devices to interrupt using the INTR line?
  - The microprocessor can only respond to one signal on INTR at a time.
  - Therefore, we must allow the signal from only one of the devices to reach the microprocessor.
  - We must assign some priority to the different devices and allow their signals to reach the microprocessor according to the priority.

The Priority Encoder

- The solution is to use a circuit called the priority encoder (74366).
  - This circuit has 8 inputs and 3 outputs.
  - The inputs are assigned increasing priorities according to the increasing index of the input.
    - Input 7 has highest priority and input 0 has the lowest.
  - The 3 outputs carry the index of the highest priority active input.
  - Figure 12.4 in the book shoes how this circuit can be used with a Tri-state buffer to implement an interrupt priority scheme.
    - The figure in the textbook does not show the method for distributing the INTA signal back to the individual devices.

Multiple Interrupts and Priority