Chapter 5

PROGRAM CONTROL INSTRUCTIONS

Prepared By Bekretsyon B.
24/10/2022

Contents

Introduction

- This chapter explains the program control instructions, including
  - jumps
  - calls
  - returns
  - interrupts
  - machine control instructions.

5–1 THE JUMP GROUP

- Allows programmer *to skip program sections and branch to any part of memory for the* next instruction.
- A *conditional jump instruction* allows decisions based upon numerical tests.
  - results are held in the flag bits, then tested by conditional jump instructions
- LOOP and conditional LOOP are also forms of the jump instruction.

5.1.1 Unconditional Jump (JMP)

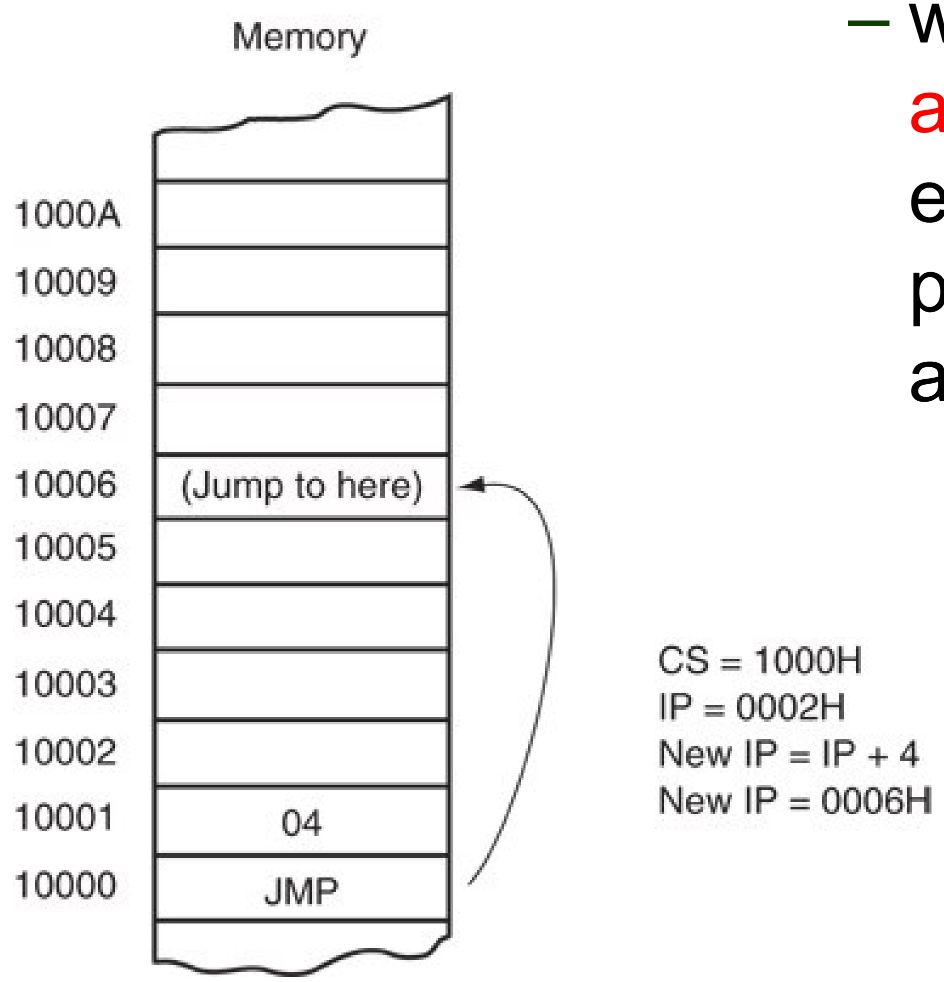
- **Three types:**
  - *short jump, near jump, far jump.*
- **Short jump** is a 2-byte instruction that allows jumps or branches to memory locations *within +127 and –128 bytes.*
  - from the address following the jump
- **3-byte near jump** allows a branch or jump within ±32K bytes from the instruction in the current code segment.

- **5-byte far jump** allows a jump to any memory location within the real memory system.
- The short and near jumps are often called **intersegment jumps.**
- Far jumps are called **intersegment jumps.**

Short Jump

- Called **relative jumps** because they can be moved, with related software, *to any location in the current code segment without a change*.
  - jump address is not stored with the opcode
  - a **distance**, or displacement, follows the opcode
- The short jump displacement is a distance represented by a 1-byte signed number whose value ranges between +127 and –128.

Figure 5–1 A short jump to four memory locations beyond the address of the next instruction.



- when the microprocessor executes a short jump, the displacement is sign-extended and added to the instruction pointer (IP/EIP) to generate the jump address within the current code segment
  - The instruction branches to this new address for the next instruction in the program

- When a jump references an address, a label normally identifies the address.
- The JMP NEXT instruction is an example.
  - it jumps to label NEXT for the next instruction
  - very rare to use an actual hexadecimal address with any jump instruction
- The label NEXT must be followed by a colon (NEXT:) to allow an instruction to reference it
  - if a colon does not follow, you cannot jump to it
- The only time a colon is used is when the label is used with a jump or call instruction.

Cont..

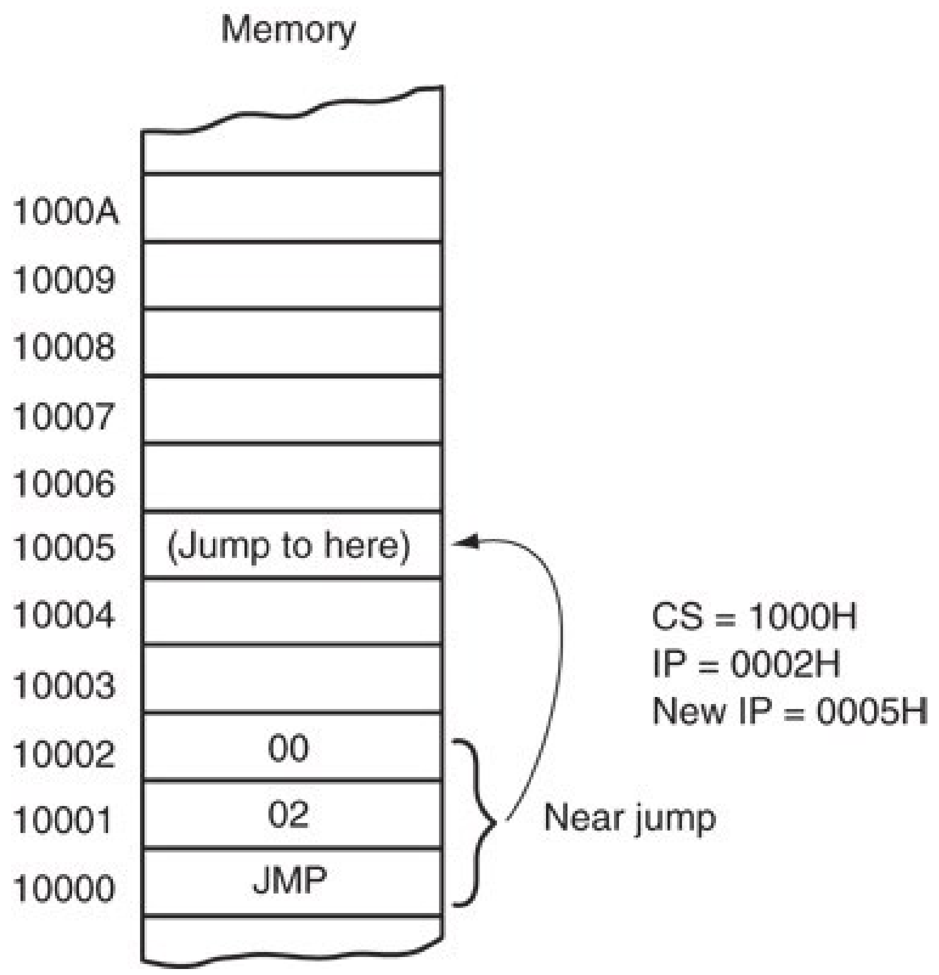
- Another way of defining a label as global is to use a *double colon* (LABEL::)
  - required inside procedure blocks defined as
    near if the label is accessed from outside the procedure block
- When the program files are joined, the linker inserts the address for the UP label into the JMP UP instruction.
- Also inserts segment address in JMP START instruction.

Near Jump

- A near jump passes control to an instruction in the current code segment located within ±32K bytes from the near jump instruction.
  - distance is ±2G in 80386 and above when operated in protected mode
- Near jump is a 3-byte instruction with opcode followed by a signed 16-bit displacement.
  - 80386 - Pentium 4 displacement is 32 bits and the near jump is 5 bytes long

- Signed displacement adds to the instruction pointer (IP) to generate the jump address.
  - because signed displacement is ±32K, a near jump can jump to any memory location within
    the current real mode code segment
- The protected mode code segment in the 80386 and above can be 4G bytes long.
  - 32-bit displacement allows a near jump to any location within ±2G bytes
- Figure 6–3 illustrates the operation of the real mode near jump instruction.

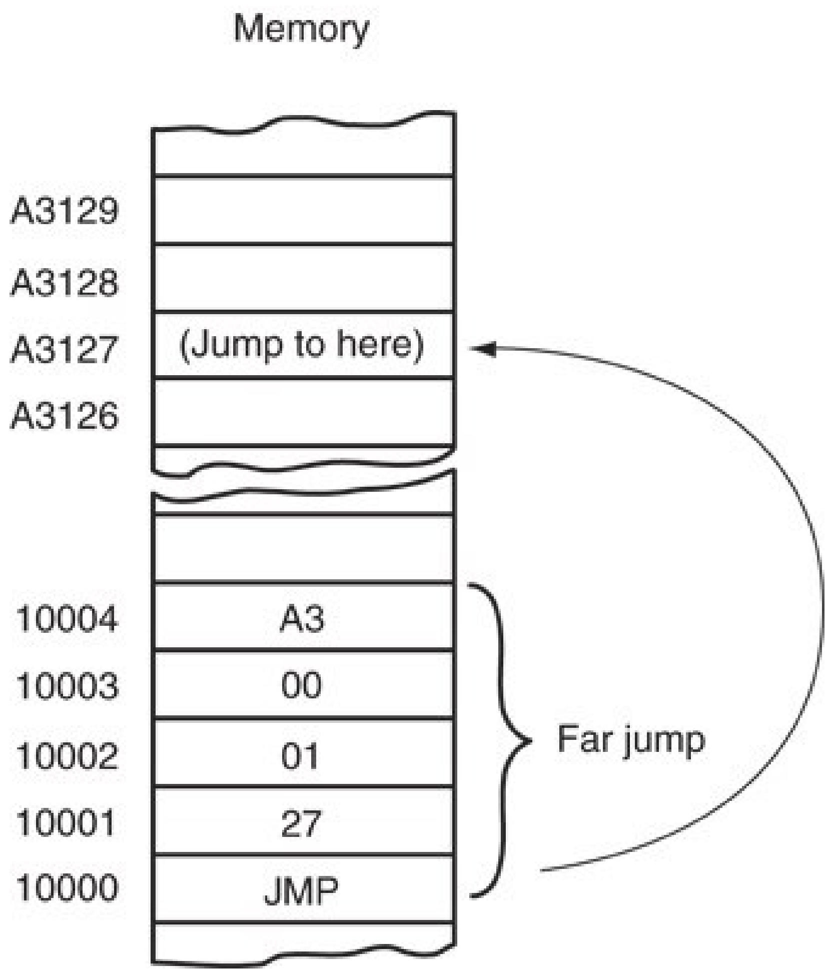
Figure 5–2 A near jump that adds the displacement (0002H) to the contents of IP.



Far Jump

- Obtains a new segment and offset address to accomplish the jump:
  - bytes 2 and 3 of this 5-byte instruction contain the new offset address
  - bytes 4 and 5 contain the new segment address
  - in protected mode, the segment address accesses a descriptor with the base address of the far jump segment
  - offset address, either 16 or 32 bits, contains the offset address within the new code segment

Figure 5–3 A far jump instruction replaces the contents of both CS and IP with 4 bytes following the opcode.



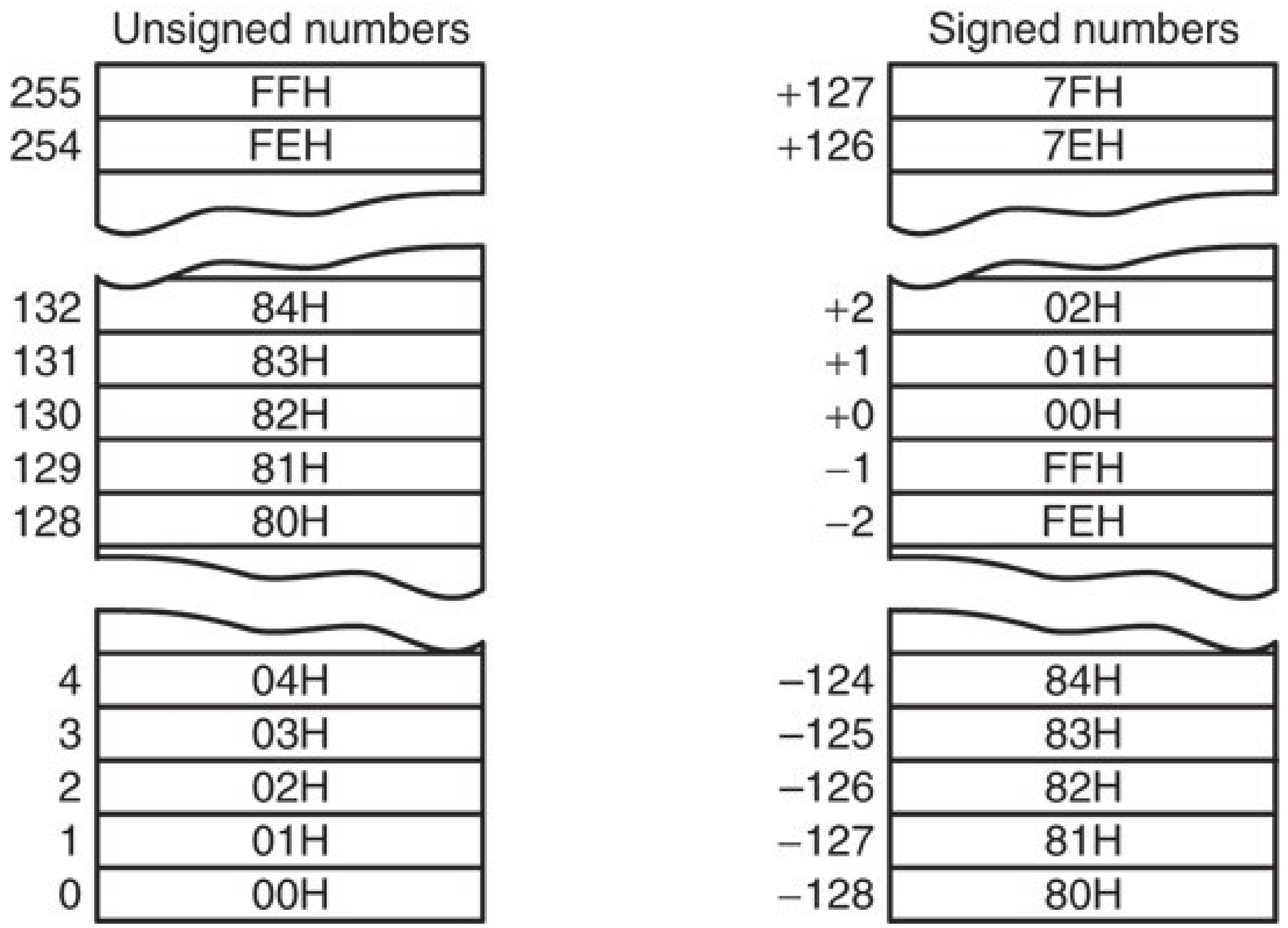
5.1.2 Conditional Jumps and Conditional Sets

- Always short jumps in 8086 - 80286.
  - limits range to within +127 and –128 bytes from the location following the conditional jump
- In 80386 and above, conditional jumps are either short or near jumps (±32K).
- Allows a conditional jump to any location within the current code segment.

- Conditional jump instructions test flag bits:
  - sign (S), zero (Z), carry (C)
  - parity (P), overflow (0)
- If the condition under test is true, a branch to the label associated with the jump instruction occurs.
  - if false, next sequential step in program executes
  - for example, a JC will jump if the carry bit is set
- Most conditional jump instructions are straightforward as they often test one flag bit.
  - although some test more than one

- Because both signed and unsigned numbers are used in programming.
- Because the order of these numbers is different, *there are two sets of conditional jump instructions for magnitude comparisons.*
- 16- and 32-bit numbers follow the same order as 8-bit numbers, except that they are larger.
- Figure 6–5 shows the order of both signed and unsigned 8-bit numbers.

Figure 6–5 Signed and unsigned numbers follow different orders.

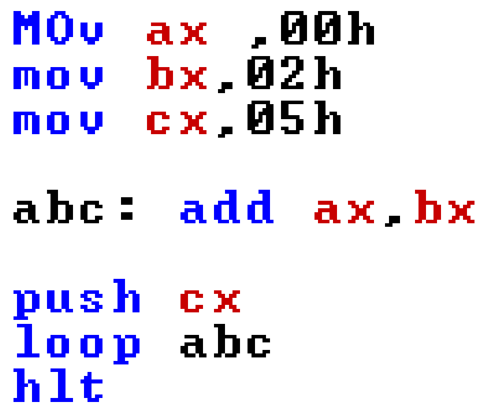


The Conditional Set Instructions

- 80386 - Core2 processors also contain conditional set instructions.
  - conditions tested by *conditional jumps put to work with the conditional set instructions*
  - conditional set instructions *set a byte to either 01H or clear a byte to 00H,* depending on the outcome of the condition under test
- Useful where a condition must be tested at a point much later in the program.

5.2.1 LOOP

- A combination of a decrement CX and the JNZ conditional jump.
- In 8086 - 80286 LOOP decrements CX.
  - if CX != 0, it jumps to the address indicated by the label
  - If CX becomes 0, the next sequential instruction executes
- In 80386 and above, LOOP decrements either CX or ECX, depending upon instruction mode.



```
MOv  ax ,00h
mov  bx,02h
mov  cx,05h

abc: add  ax,bx

push cx
loop abc
hlt
```

- In 16-bit instruction mode, LOOP uses CX; in the 32-bit mode, LOOP uses ECX.
  - default is changed by the LOOPW (using CX) and LOOPD (using ECX) instructions 80386 - Core2
- In 64-bit mode, the loop counter is in RCX.
  - and is 64 bits wide
- There is no direct move from segment register to segment register instruction.

Conditional LOOPs

- LOOP instruction also has conditional forms: LOOPE and LOOPNE
- LOOPE (**loop while equal**) instruction jumps if CX != 0 while an equal condition exists.
  - will exit loop if the condition is not equal or the CX register decrements to 0
- LOOPNE (**loop while not equal**) jumps if CX != 0 while a not-equal condition exists.
  - will exit loop if the condition is equal or the CX register decrements to 0

- In 80386 - Core2 processors, *conditional LOOP can use CX or ECX as the counter.*
  - LOOPEW/LOOPED or LOOPNEW/LOOPNED override the instruction mode if needed
- Under 64-bit operation, the loop counter uses RCX and is 64 bits in width
- Alternates exist for LOOPE and LOOPNE.
  - LOOPE same as LOOPZ
  - LOOPNE instruction is the same as LOOPNZ
- In most programs, only the LOOPE and LOOPNE apply.

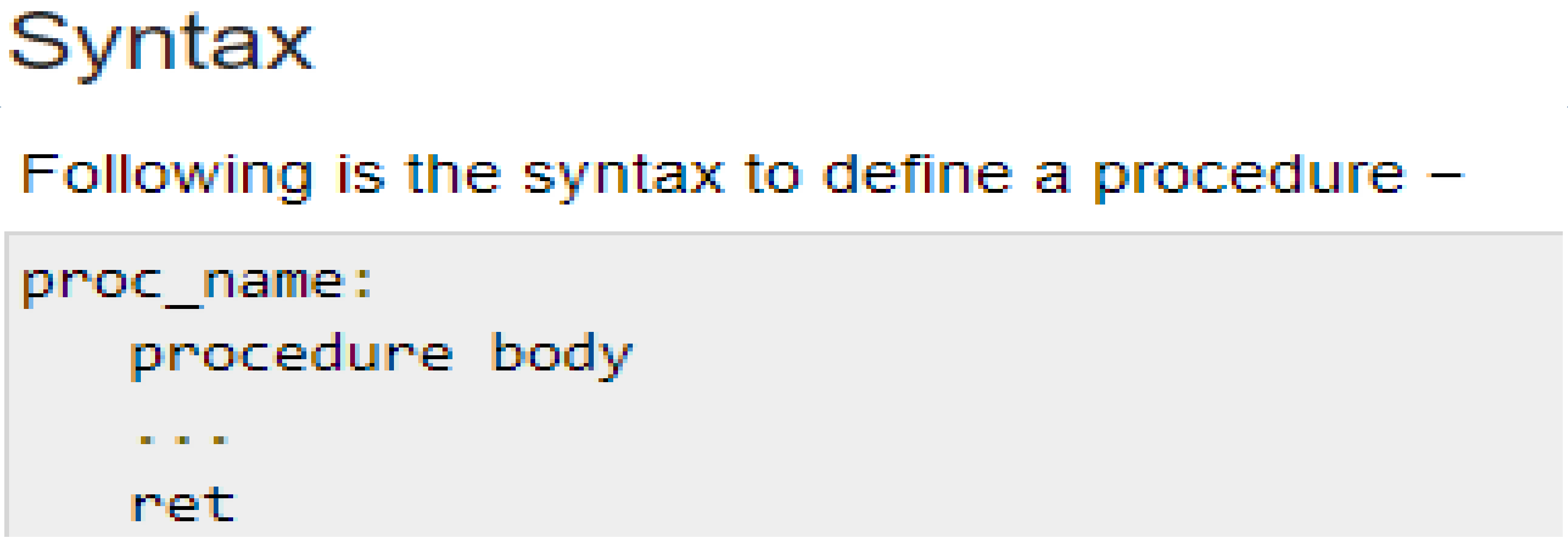
REPEAT-UNTIL Loops

- A series of instructions is repeated until some condition occurs.
- The .REPEAT statement defines the start of the loop.
  - end is defined with the .UNTIL statement, which contains a condition
- An .UNTILCXZ instruction uses the LOOP instruction to check CX for a repeat loop.
  - .UNTILCXZ uses the CX register as a counter
    to repeat a loop a fixed number of times

5–3 PROCEDURES

- A procedure is a group of instructions that usually performs one task.
  - subroutine, method, or **function** is an important part of any system’s architecture
- A procedure is a reusable section of the software stored in memory once, used as often as necessary.
  - saves memory space and makes it easier to develop software
- Disadvantage of procedure is time it takes the computer to link to, and return from it.
  - CALL links to the procedure; the RET (**return**) instruction returns from the procedure

- CALL pushes *the address of the instruction following the CALL (**return address**) on the stack.*
  - the stack stores the return address when a procedure is called during a program
- RET instruction removes an address from the stack so the program returns to the instruction following the CALL.
- A procedure begins with the PROC directive and ends with the ENDP directive.
  - each directive appears with the procedure name



Syntax

Following is the syntax to define a procedure −

```
proc_name:
   procedure body
   ...
   ret
```

- Procedures that are to be *used by all software (global) should be written as far procedures.*
- Procedures that are used by a given task (local) are normally defined as near procedures.
- Most procedures are near procedures.

CALL

- Transfers the flow of the program to the procedure.
- CALL instruction *differs from the jump instruction because a CALL saves a return address on the stack.*
- The return address returns control to the instruction that immediately follows the CALL in a program when a RET instruction executes.

Near CALL

- 3 bytes long.
  - the first byte contains the opcode; the second and third bytes contain the displacement
- When the near CALL executes, it first pushes the offset address of the next instruction onto the stack.
  - offset address of the next instruction appears in the instruction pointer (IP or EIP)
- It then adds displacement from bytes 2 & 3 to the IP to transfer control to the procedure.

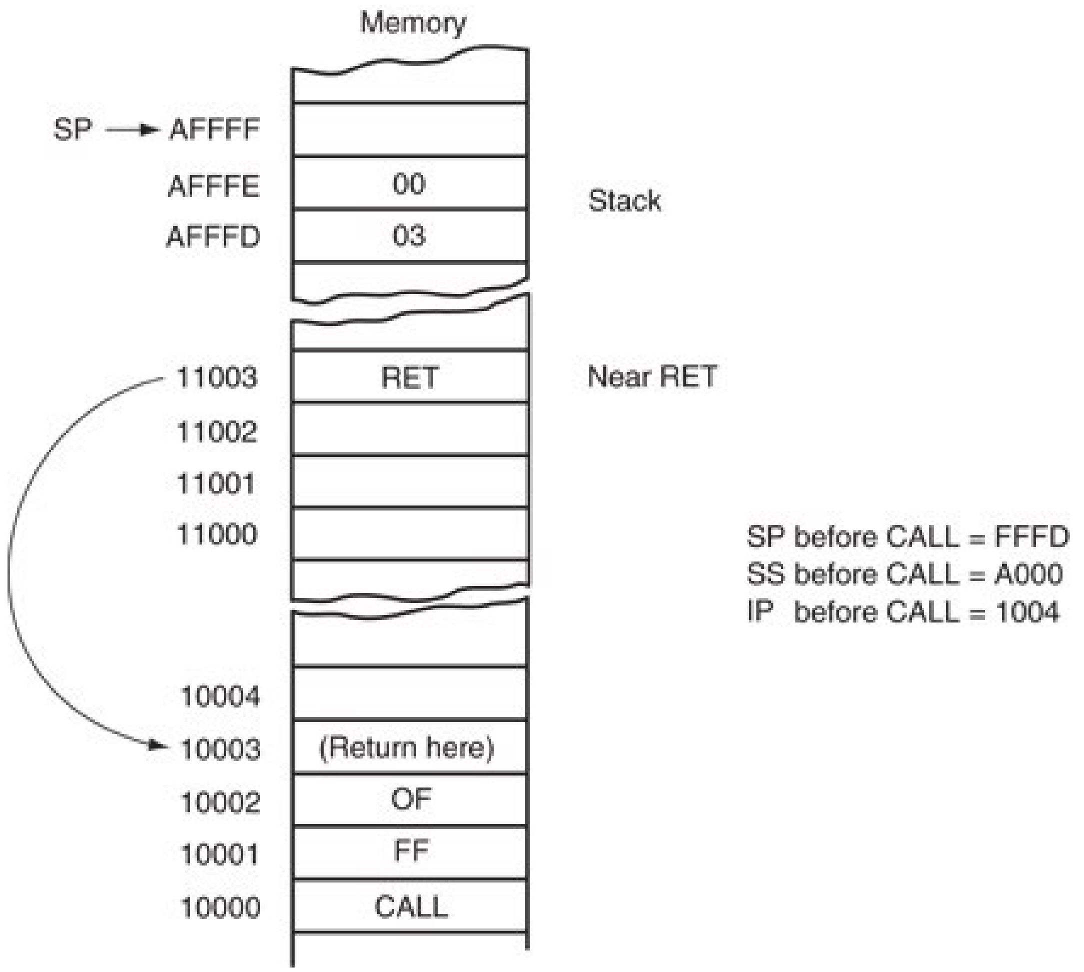
Far CALL

- 5-byte instruction contains an opcode followed by the next value for the IP and CS registers.
  - bytes 2 and 3 contain new contents of the IP
  - bytes 4 and 5 contain the new contents for CS
- Far CALL places the contents of both IP and CS on the stack before jumping to the address indicated by bytes 2 through 5.
- This allows far CALL to call a procedure located anywhere in the memory and return from that procedure.

RET

- Removes a 16-bit number (**near return**) from the stack placing it in IP, or removes a 32-bit number (**far return**) and places it in IP & CS.
  - near and far return instructions in procedure’s PROC directive
  - automatically selects the proper return instruction
- Figure 6–8 shows how the CALL instruction links to a procedure and how RET returns in the 8086–Core2 operating in the real mode.

The effect of a near return instruction on the stack and instruction pointer.



5–4 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.
- Will be discussed in chapter 8 in detail

Interrupt Vectors

- A 4-byte number stored in the first 1024 bytes of memory (00000H–003FFH) in real mode.
  - in protected mode, the vector table is replaced by an interrupt descriptor table that uses 8-byte descriptors to describe each of the interrupts
- 256 different interrupt vectors.
  - each vector contains the address of an interrupt service procedure

- 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
- Intel reserves the first 32 interrupt vectors for the present and future products.
  - interrupt vectors (32–255) are available to users
- Some reserved vectors are for errors that occur during the execution of software
  - such as the divide error interrupt

- Some vectors are reserved for the coprocessor.
  - others occur for normal events in the system
- In a personal computer, reserved vectors are used for system functions
- Vectors 1–6, 7, 9, 16, and 17 function in the real mode and protected mode.
  - the remaining vectors function only in the protected mode

Interrupt Instructions

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

INTs

- 256 different software interrupt instructions (INTs) available to the programmer.
  - each INT instruction has a numeric operand whose range is 0 to 255 (00H–FFH)
- For example, INT 100 uses interrupt vector 100, which appears at memory address 190H–193H.
  - address of the interrupt vector is determined by multiplying the interrupt type number by 4

- When a software interrupt executes, it:
  - pushes the flags onto the stack
  - clears the T and I flag bits
  - pushes CS onto the stack
  - fetches the new value for CS from the interrupt vector
  - pushes IP/EIP onto the stack
  - fetches the new value for IP/EIP from the vector
  - jumps to the new location addressed by CS and IP/EIP

- 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.
- Software interrupts are most commonly used to call system procedures because the address of the function need not be known.
- The interrupts often control printers, video displays, and disk drives.

- INT replaces a far CALL that would otherwise be used to call a system function.
  - INT instruction is 2 bytes long, 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/IRETD

- 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.

- When IRET executes, it restores the contents of I and T from the stack.
  - preserves the state of these flag bits
- If interrupts were enabled before an interrupt service procedure, they are automatically re-enabled by the IRET instruction.
  - because it restores the flag register
- IRET is used in real mode and IRETD in the protected mode.

INT 3

- A special software interrupt designed to function as a breakpoint.
  - a 1-byte instruction, while others are 2-byte
- Common to insert an INT 3 in software to interrupt or break the flow of the software.
  - function is called a breakpoint
  - breakpoints help to debug faulty software
- A breakpoint occurs for any software interrupt, but because INT 3 is 1 byte long, it is easier to use for this function.

INTO

- Interrupt on overflow (INTO) is a conditional software interrupt that tests overflow flag (O).
  - If O = 0, INTO performs no operation
  - if O = 1 and an INTO executes, an interrupt occurs via vector type number 4
- The INTO instruction appears in software that adds or subtracts signed binary numbers.
  - eith these operations, it is possible to have an overflow
- JO or INTO instructions detect the overflow.

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.

- In software interrupt service procedure, hardware interrupts are enabled as one of the first steps.
  - accomplished by the STI instruction
- Interrupts are enabled early because just about all of the I/O devices in the personal computer are interrupt-processed.
  - if interrupts are disabled too long, severe system problems result

Interrupts in the Personal Computer

- Interrupts found in the personal computer only contained Intel-specified interrupts 0–4.
- Access to protected mode interrupt structure in use by Windows is accomplished through kernel functions Microsoft provides.
  - and cannot be directly addressed
- Protected mode interrupts use an interrupt descriptor table.

5–5 MACHINE CONTROL AND MISCELLANEOUS INSTRUCTIONS

- These instructions provide control of the carry bit, sample the BUSY/TEST pin, and perform various other functions.
- **Controlling the Carry Flag Bit**
- The carry flag (C) propagates the carry or borrow in multiple-word/doubleword addition and subtraction.
  - can indicate errors in assembly language procedures
- Three instructions control the contents of the carry flag:
  - STC (set carry), CLC (clear carry), and CMC (complement carry)

WAIT

- Monitors the hardware *BUSY* pin on 80286 and 80386, and the *TEST* pin on 8086/8088.
- If the WAIT instruction executes while the *BUSY* pin = 1, nothing happens and the next instruction executes.
  - pin inputs a busy condition when at a logic 0 level
  - if *BUSY* pin = 0 the microprocessor waits for the pin to return to a logic 1

HLT

- Stops the execution of software.
- There are three ways to exit a halt:
  - by interrupt; a hardware reset, or DMA operation
- Often synchronizes external hardware interrupts with the software system.
- DOS and Windows both use interrupts extensively.
  - so HLT will not halt the computer when operated under these operating systems

5.6 NOP

- In early years, before software development tools were available, a NOP, which performs absolutely no operation, was often used to pad software with space for future machine language instructions.
- When the microprocessor encounters a NOP, it takes a short time to execute.

- If you are developing machine language programs, which are extremely rare, it is recommended that you place 10 or so NOPS in your program at 50-byte intervals.
  - in case you need to add instructions at some future point
- A NOP may also find application in time delays to waste time.
- A NOP used for timing is not very accurate because of the cache and pipelines in modern microprocessors.

Chapter 6

8086/8088 HARDWARE SPECIFICATIONS

Contents

6.1 Introduction to 8086/8088 spec

- both are packaged in 40-pin dual in-line packages (DIPs)
  - 8086 is a 16-bit microprocessor with a 16-bit data bus;
  - 8088 has an 8-bit data bus.
  - 8086 has pin connections AD0–AD15
  - 8088 has pin connections AD0–AD7
- Data bus width is the only major difference.
- thus 8086 transfers 16-bit data more efficiently

Cont..

- Both microprocessors require +5.0 V with a supply voltage tolerance of ±10 percent.
- 8086 uses a maximum supply current of 360 mA
- 8088 draws a maximum of 340 mA
- Both microprocessors operate in ambient temperatures of between 32°F and 180°F (0°C and 82°C).

6.2 Clock Generator (8284A)

- A clock generator or system clock is a circuit that produces a timing signal (known as a clock signal) for use in synchronizing a circuit's operation.
- The 8284A is an integrated circuit designed to be used in the 8086/8088 microprocessor.
- A clock signal is a particular type of signal that oscillates between ahigh and a low state.

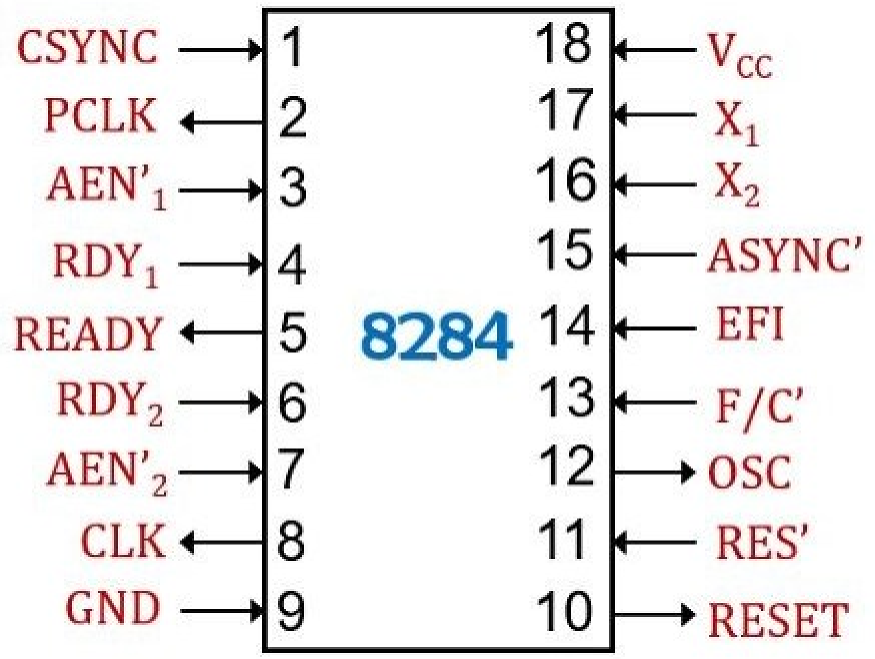
Cont..

- 8284 clock generator is an IC developed by Intel to provide clock frequency, ready and reset signal to the 8086/8088 microprocessor.
- It is an 18 pin chip.
- 8284 produces the clock signal, synchronizes it with the ready and reset signal and provides it to the microprocessor.

8284A PIN CONFIGURATION

- Pins are divided into three category:
  - Power Supply Pins
  - Input Pins
  - Output Pins

Pin Description of 8284



Pin Configuration of 8284

1. Power supply pins:

a) VCC…..(18# pin)

- VCC is the power input pin connected to +5.0V with a tolerance of+10 percent

b) GND…..(9# pin)

- pin is used for the ground connection.

2.Input pins:

A. Reset in…..(11 # pin)

B. X1 and X2(crystal in)…..(17# and 16# pins) resp.

- required while connecting quartz crystal. When EFI is provided then X1 is connected with V_{CC} or GND.

C. F/C(frequency/clock select)…..(13 #pin).

D. EFI(external freq in)…..(14# pin).

- provides external input frequency to 8284 when F/C’ is high.

Cont.

E. CSYNC(clock synchronization)…..(1# pin)

- It is an active high signal that synchronizes the clock signal of various 8284 chips present in a single system

F. RDY1 and AEN1(ready1 and address enable1)…..(4# and 3# pins) resp.

G. RDY2 and AEN2(ready2 and address enable2)…..(6# and 7# pins) resp.

- These are active high pins and these signals are provided by devices present on the data bus showing the availability or reception of the data.

H. ASYNC(synchronization select)…..(15# pin)

- gives the information regarding the synchronization provided to the inputs.

OUTPUT SIGNALS:

A. RESET…..(10# pin).
B. OSC(Oscillator)…..(12# pin).
C. CLK(clock)…..(8# pin).
D. PCLK(peripheral clock)…..(2# pin).
E. Ready…..(5# pin).

Functions of 8284

- It **provides a stable clock** to the processor.
- In the case of a multiprocessor system, *it **facilitates synchronization** of multiple clock signals*.
- Provides **resetting to the processor** along with the clock signal.

6.3 Bus Buffering and Latching

- A temporary storage area, usually in RAM.
- The purpose of most buffers is to act as a holding area, enabling the CPU to manipulate data before transferring it to a device.
- Then when you save the file, the word processor updates the disk file with the contents of the buffer.
- Buffers pass an input through to output after some propagation time, possibly increasing drive strength (increasing fanout). Latches additionally add memory, to capture and persist the input value at some point in time (memory). This latching behavior is triggered by a third signal, control.

Cont..

- Before the 8086/8088 microprocessors *can be used with memory or I/O interfaces, their multiplexed buses must be demultiplexed.*
- This section provides the detail required to demultiplex the buses and illustrates how the buses are buffered for very large systems.
- Because the maximum fan-out is 10, the system must be buffered if it contains more than 10 other components.

Cont..

- All computer systems have three buses:

1. An address bus that provides the memory and I/O with the memory address or the I/O port number,
2. A data bus that transfers data between the micro- processor and the memory and I/O in the system,
3. A control bus that provides control signals to the memory and I/O.

- These buses must be present in order to interface to memory and I/O.

The Buffered System

- If more than 10 unit loads are attached to any bus pin, the entire 8086 or 8088 system must be buffered.
- The demultiplexed pins are already buffered by the 74LS373 or 74LS573 latches, which have been designed to drive the high-capacitance buses encountered in microcomputer systems.
- The buffer's output currents have been increased so that more TTL unit loads may be driven: A logic 0 output provides up to 32 mA of sink current, and a logic 1 output provides up to 5.2 mA of source current.

6.4 Ready and the Wait State

- the READY input causes wait states for slower memory and I/O components.
- A wait state (Tw) is an *extra clocking period, inserted between T2 and T3 to lengthen the bus cycle.*
- If one wait state is inserted, then the memory access time, normally 460 ns with a 5 MHz clock, is lengthened by one clocking period (200 ns) to 660 ns.
- RDY is the synchronized ready input to the 8284A clock generator.

6.5. Minimum Mode versus Maximum Mode

- 8080 and 8085 were used for single processor applications.
- Here 8086 is backwards Compatible with 8080 and 8085 , that means when minimum mode is used it function is similar to 8085/8080
- But 8086 is advanced built for complex application involving multiprocessor
- In max mode the control signals are generated by 8288 bus controller and 8289 bus arbiter IC

Differences between the Minimum and Maximum mode of operation:

Maximum mode	**Minimum Mode**
When pin 33 MN / Mx' is connected to GND.	When pin 33 MN/Mx' is connected to high.
In maximum mode 8086 generates QS1,QS0,S0' ,S1',S2', LOCK(bar),RQ(bar)/GT1,RQ(bar)/GT0 control signals. and other signals are generated with the help of S0', S1' and S2'	8086 generates INTA(bar), ALE, DEN(bar), DT/R(bar), M/IO(bar), HLDA,HOLD and WR(bar) control signals.
It is used for multi-processors system.	Used in single-processor applications
Whereas in maximum mode interfacing, master/slave and multiplexing and several such control signals are required	In minimum mode no interfacing or master/slave signals is required.
In maximum mode a bus controller is required to produce control signals. This bus controller produces MEMRDC, MEMWRC, IORDC, IOWRC, ALE, DEN, DT/R control signals.	In minimum mode direct RD / WR signals can be used. No bus controller required. A simple demultiplexer would do the job. of producing the control signals. This demultiplexer produces MEMRD, MEMWR, IORD, IOWR control signals.

?

Thanks

Chapter 7

INTERFACE

Contents

Introduction

- Any application of a microprocessor based system requires the transfer of data between external circuitry to the microprocessor and microprocessor to the external circuitry.
- Most of the peripheral devices are designed and interfaced with a CPU either to enable it to communicate with the user or an external process and to ease the circuit operations so that the microprocessor works more efficiently.

Cont..

- **Interface** is the path for communication between two components.
- Interfacing is of two types,
  - Memory interfacing
  - I/O interfacing.

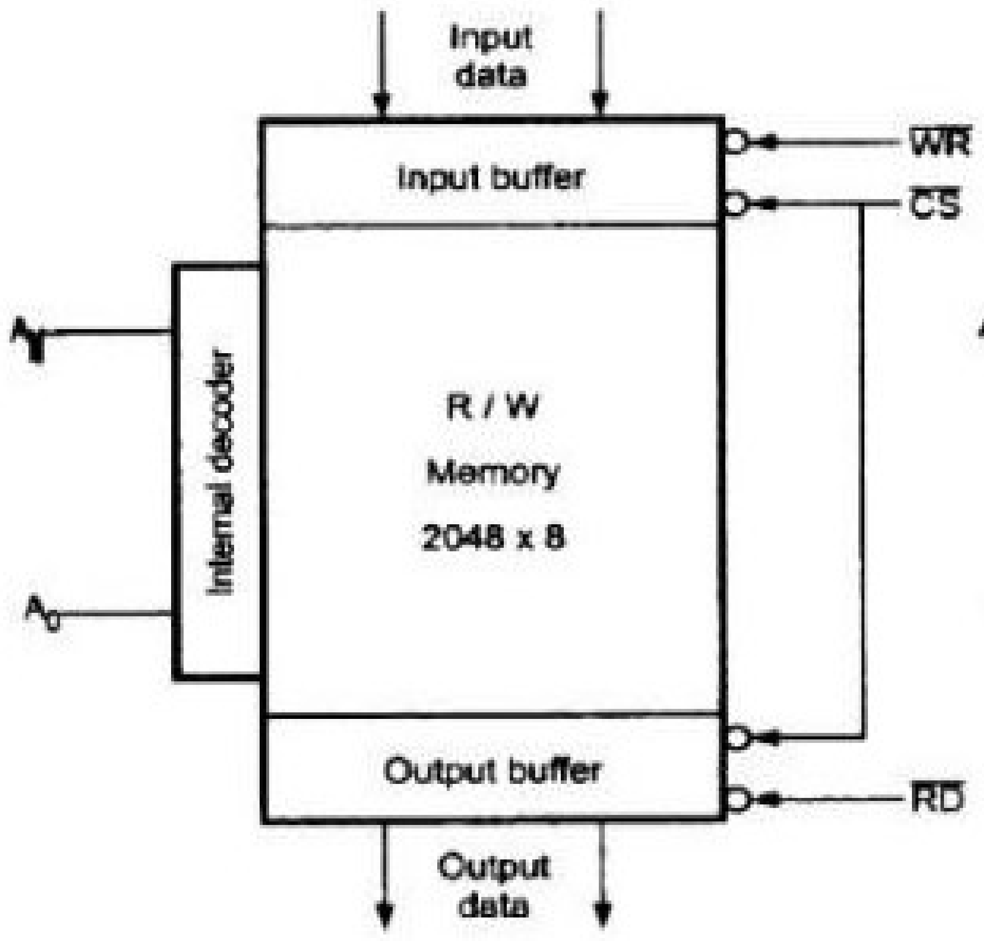
7.1 Memory Interfacing

- When we are executing any instruction, we need the microprocessor to access the memory for reading instruction codes and the data stored in the memory.
- For this, *both the memory and the microprocessor requires some signals to read from and write to registers.*
- The interfacing process includes some key factors to match with the memory requirements and microprocessor signals.
- The interfacing circuit therefore should be designed in such a way that it matches the memory signal requirements with the signals of the microprocessor.

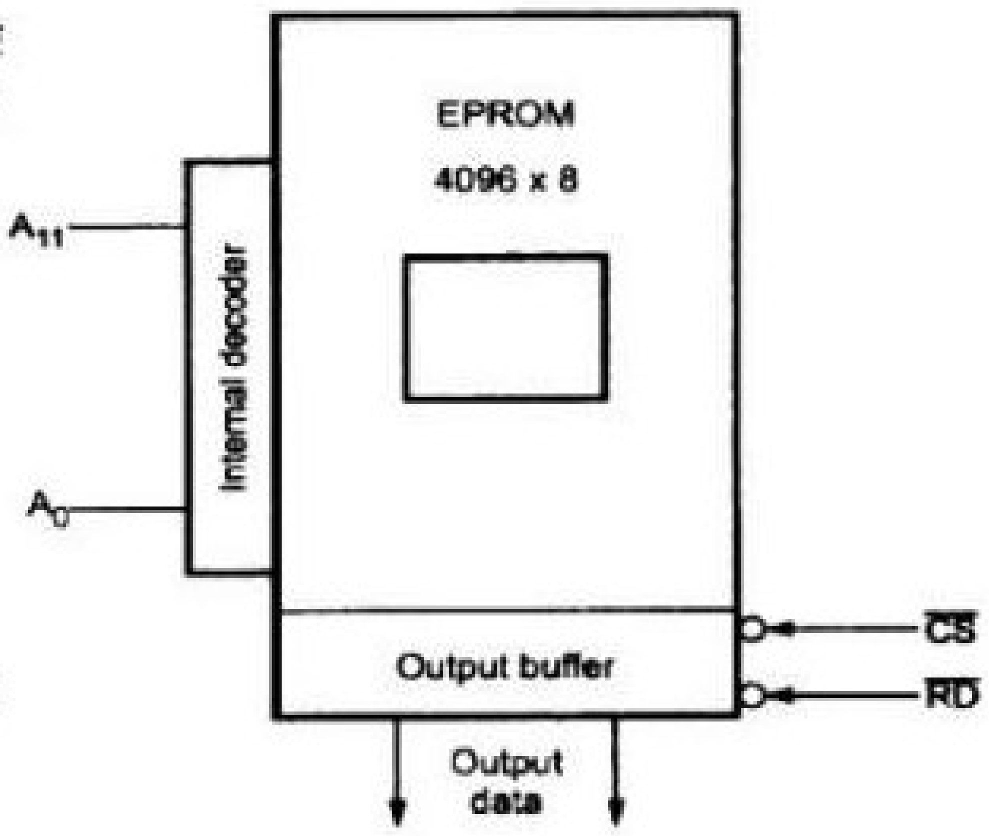
Cont..

- The memory interfacing circuit is used to access memory quit frequently *to read instruction codes and data stored in the memory*.
- The read / write *operations are monitored by control signals.*
- *Semiconductor memories are of two types.*
  - *RAM (Random Access Memory) and*
    - *The Semiconductor RAM's are broadly two types- static Ram and dynamic RAM*
  - *ROM (Read Only Memory)*

cont..



(a) Logic diagram for RAM

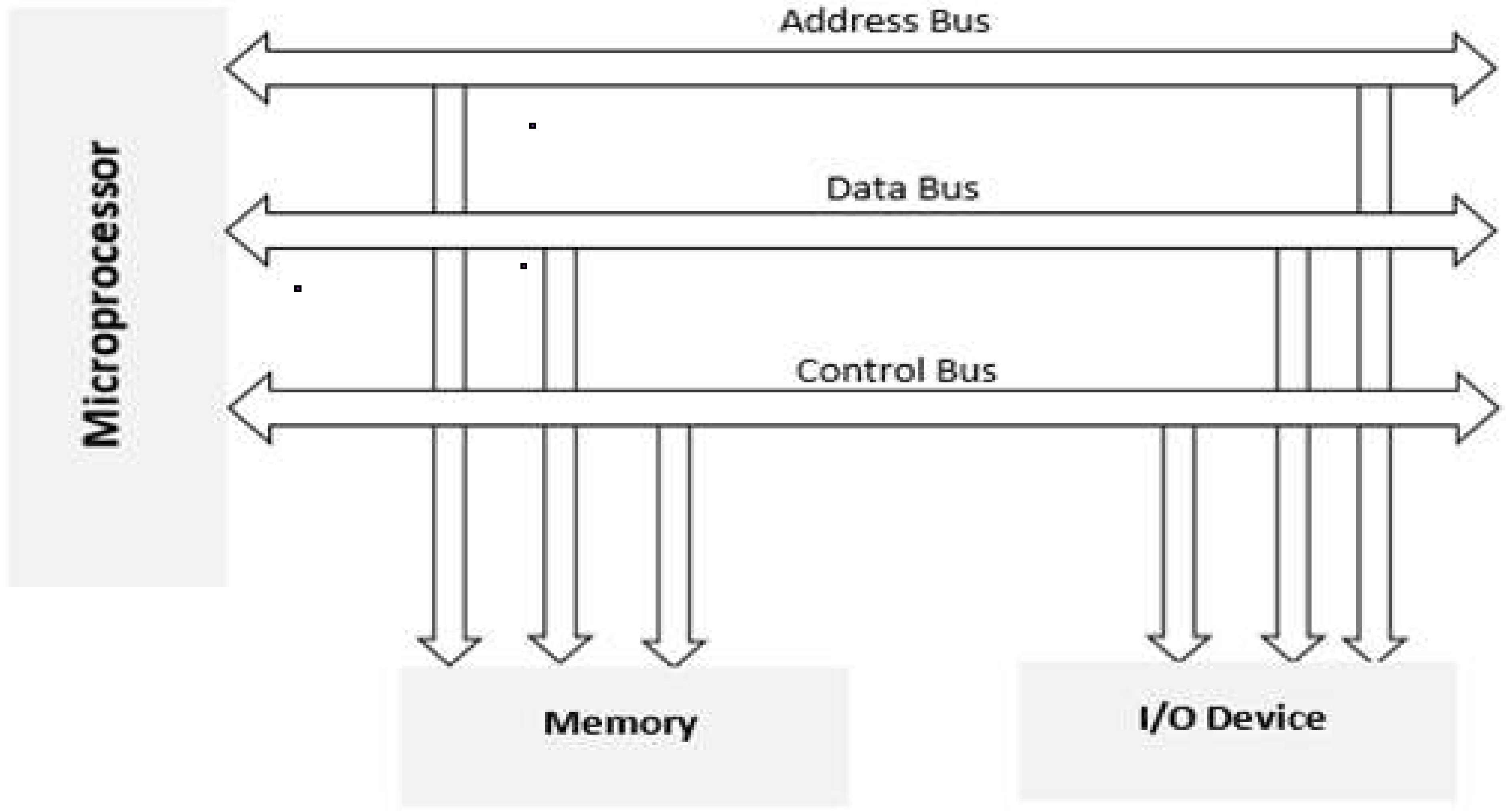


(b) Logic diagram for EPROM

7.2 IO Interfacing

- There are various communication devices like the keyboard, mouse, printer, etc.
- So, we need to interface the keyboard and *other devices with the microprocessor by using latches and buffers.*
- This type of interfacing is known as I/O interfacing.

Block Diagram of Memory and I/O Interfacing



7.2.1 I/O Interfacing Techniques

- Input/output devices can be interfaced with microprocessor systems in two ways:

1. I/O mapped I/O
2. Memory mapped I/O

1. I/O mapped I/O:

- 8086 has special instructions IN and OUT to transfer data through the input/output ports in I/O mapped I/O system.
- **The IN instruction copies data** from a port to the Accumulator.
- If an 8-bit port is read data will go to AL and if 16-bit port is read the data will go to AX.
- **The OUT instruction copie**s a byte from AL or a word from AX to the specified port.
- The M/IO signal is always low when 8086 is executing these instructions.

2. Memory mapped I/O

- In this type of I/O interfacing, *the 8086 uses 20 address lines to identify an I/O device.*
- The I/O device is *connected as if it is a memory device*.
- The 8086 uses same *control signals and instructions to access I/O as those of memory,* here RD and WR signals are activated indicating memory bus cycle.

Ways of Communication Microprocessor with the Outside World?

- There are two ways of communication in which the microprocessor can connect with the outside world.
  - Serial Communication Interface
  - Parallel Communication interface

Cont..

- **Serial Communication Interface –**
  - the interface gets a *single byte of data from the microprocessor* and *sends it bit by bit to the other system serially and vice-a-versa.*
- **Parallel Communication Interface –**
  - The interface gets a byte of data from the microprocessor and *sends it bit by bit to the other systems in simultaneous (or) parallel fashion and vice-a-versa*.

7.4 Programmable peripheral interface

- PPI 8255 is a general purpose programmable I/O device designed to interface the CPU with its outside world such as
  - ADC -A ADC is a device that turns a analogue signal into a digital one
  - DAC -A DAC is a device that turns a digital signal into an analogue one
  - keyboard etc.
- We can program it according to the given condition.
- It can be used with almost any microprocessor.
- It consists of three 8-bit bidirectional I/O ports i.e.
  - *PORT A, PORT B and PORT C.*
- We can assign different ports as input or output functions.

Chapter 8

INTERRUPTS

Contents

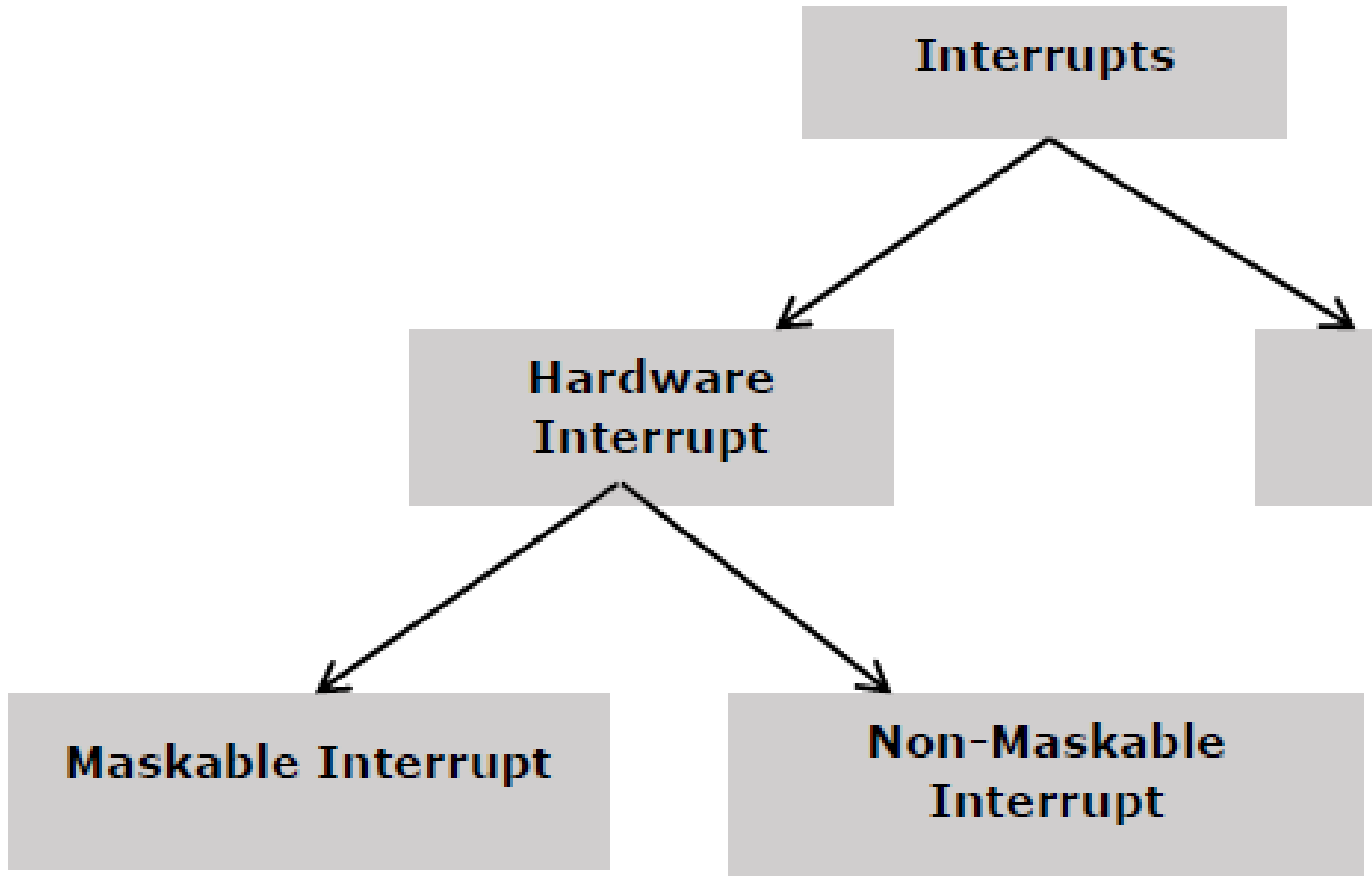
8.1 Interrupts

- Interrupt is the method of **creating a temporary halt during program execution** and allows peripheral devices to access the microprocessor.
- The microprocessor responds to that interrupt with an ISR (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.
- An interrupt is a condition that halts the microprocessor temporarily to work on a different task and then return to its previous task.
- Interrupt is an event or signal that request to attention of CPU.
- This halt allows peripheral devices to access the microprocessor.

Cont..

- 8086 μP can implement 256 different interrupts.

Interrupts…


Interrupts
Hardware Interrupt
Software Interrupt
Maskable Interrupt
Non-Maskable Interrupt

Hardware Interrupts

- Hardware interrupt is *caused by any peripheral device by sending a signal through a specified pin to the microprocessor.*
- The 8086 has two hardware interrupt pins,
  - NMI and INTR.
  - NMI is a non-maskable interrupt and INTR is a *maskable interrupt having lower priority.*
  - One more interrupt pin associated is INTA called interrupt acknowledge

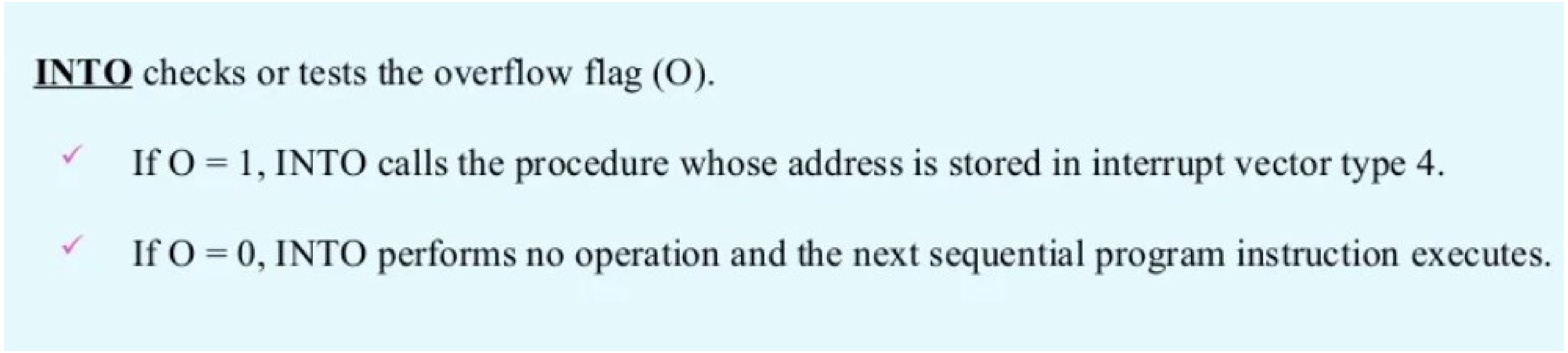
Software Interrupts

- Some instructions are inserted at the desired position into the program to create interrupts.
- These interrupt instructions can be used to test the working of various interrupt handlers. It includes −

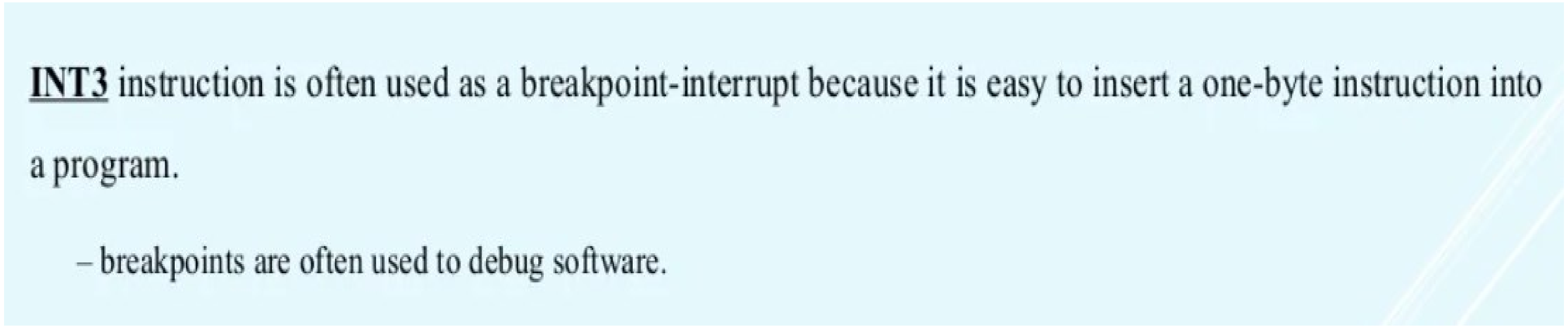
Interrupt Instructions:

- 8086 has the following Software interrupt instructions
  - INT and INT3 Are Very similar
  - INTO are Conditional
  - IRET is a special interrupt return instruction

Cont..

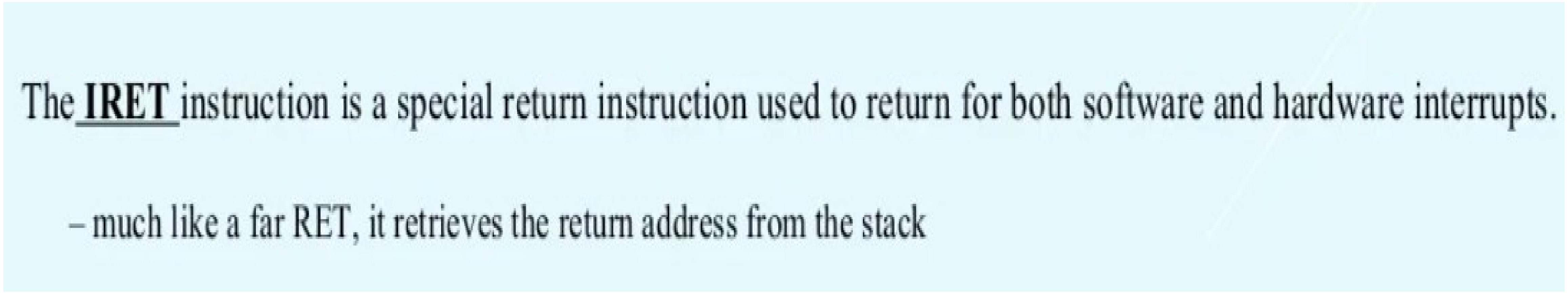

INTO checks or tests the overflow flag (O).

- ✓ If O = 1, INTO calls the procedure whose address is stored in interrupt vector type 4.
- ✓ If O = 0, INTO performs no operation and the next sequential program instruction executes.


INT3 instruction is often used as a breakpoint-interrupt because it is easy to insert a one-byte instruction into a program.

– breakpoints are often used to debug software.

Cont….



The **IRET** instruction is a special return instruction used to return for both software and hardware interrupts.

– much like a far RET, it retrieves the return address from the stack

Interrupt flag

- In 8086 the interrupt flag (IF) can be set to one to unmask or enable all hardware interrupts and IF is cleared to zero to mask or disable a hardware interrupts except NMI.
- The interrupts whose request can be either accepted or rejected by the processor are called maskable interrupts.