Microprocessors (0630371)
Fall 2010/2011 – Lecture Notes # 20

16-Bit MS-DOS Programming

(*MS-DOS & BIOS-level Programming*)

Objectives

- **Real-Address Mode**
- **MS-DOS Memory Organization**
- **MS-DOS Memory Map**
- **Interrupts Mechanism—Introduction**
- **Interrupts Mechanism — Steps**
- **Types of Interrupts**
- **8086/8088 Pinout Diagrams**
- **Redirecting Input-Output**
- **INT Instruction**
- **Interrupt Vectoring Process**
- **Common Interrupts**

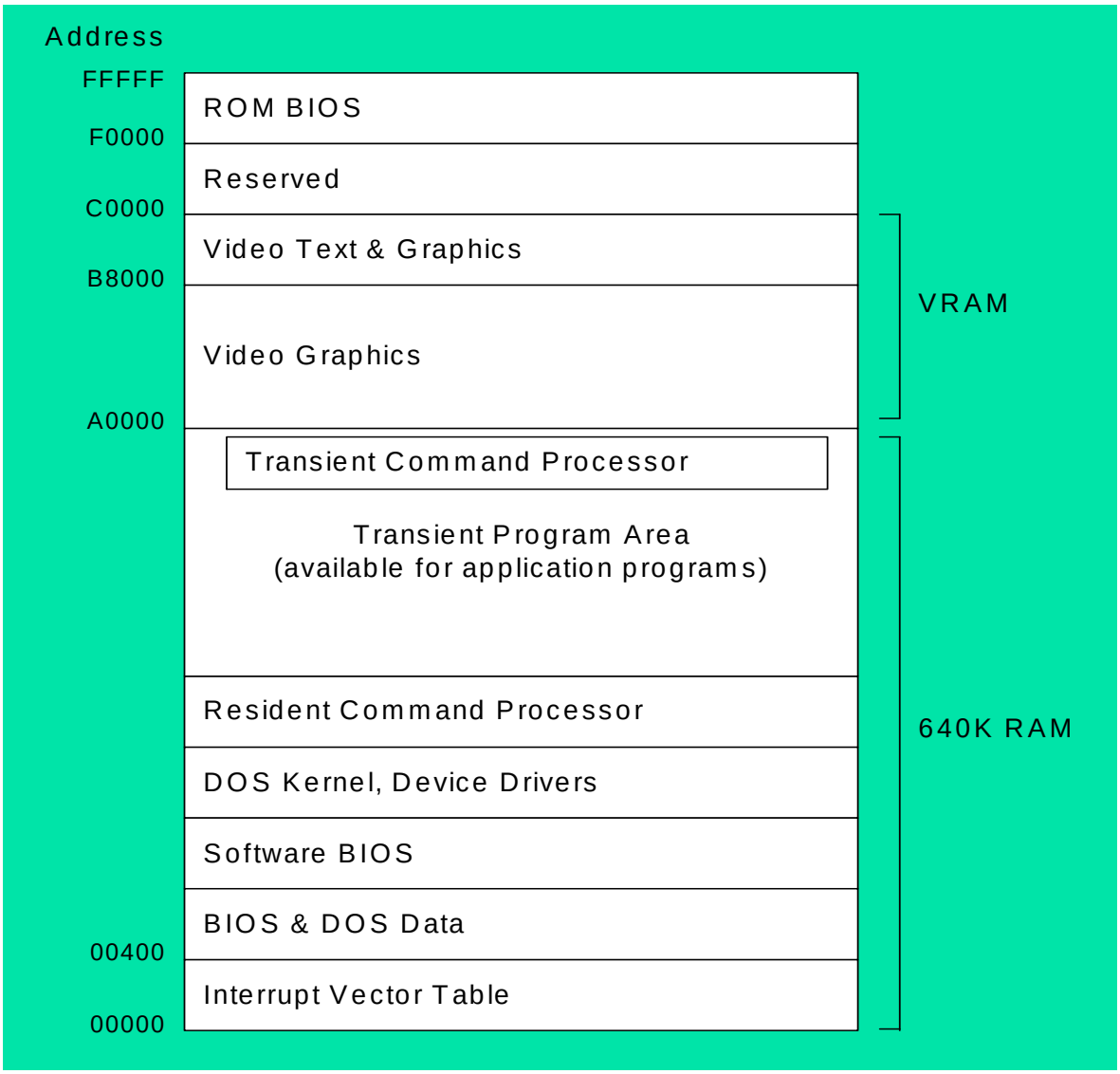
Real-Address Mode

- Real-address mode (16-bit mode) programs have the following characteristics:
  - Max 1 megabyte addressable RAM
  - Single tasking
  - No memory boundary protection
  - Offsets are 16 bits
- IBM PC-DOS: first Real-address OS for IBM-PC
- Later renamed to MS-DOS, owned by Microsoft

MS-DOS Memory Organization

- Interrupt Vector Table
- BIOS & DOS data
- Software BIOS
- MS-DOS kernel
- Resident command processor
- Transient programs
- Video graphics & text
- Reserved (device controllers)
- ROM BIOS

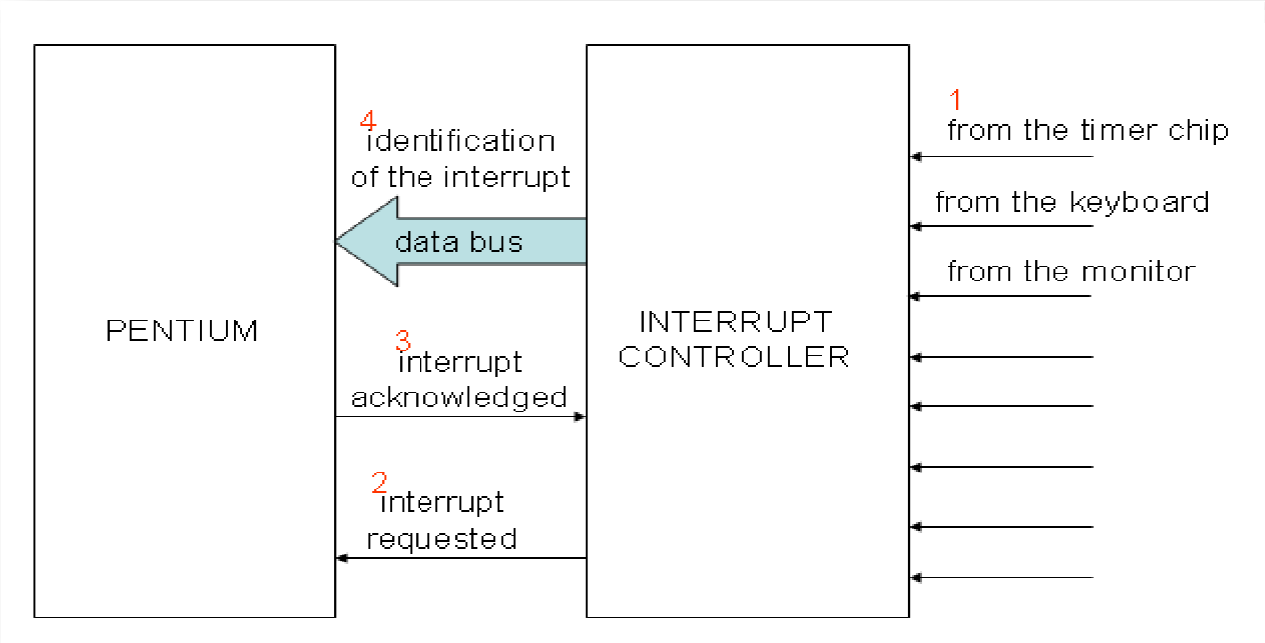
MS-DOS Memory Map



Interrupt Mechanism—Introduction

- Devices such as the keyboard, the monitor, hard disks etc. can cause such interrupts, when they require service of some kind, such as to get or receive a byte. For example, when you press a key on the keyboard this causes an interrupt.
- When the Microprocessor is interrupted, it completes the current instruction, and then pushes onto the stack the flags register plus the address of the next instruction (the **return** address).
- It then carries out the procedure that services the interrupt involved.
- Then it uses a special return instruction **iret** which pops the flags register from the stack, and pops and uses the return address to resume doing whatever it was doing before the interrupt occurred.
- x86 Recognizes 256 Different Interrupts

Interrupt Mechanism Steps:



1. The device signals its request, for an interrupt service, to the interrupt controller.
2. The interrupt controller sends a signal to the Microprocessor requesting an interrupt.
3. The Microprocessor is hard-wired to respond by finishing its current instruction, pushing the flags register, and the address of the next instruction (the return address) onto the stack, and then sending an acknowledgment signal back to the interrupt controller.
4. The interrupt controller then puts onto the data bus a code that identifies which of its input lines (and hence which device) is requesting this interrupt.

Types of Interrupts:

- **External** - generated outside CPU by other hardware
- **Internal** - generated within CPU as a result of an instruction or operation
  - x86 has internal interrupts**: int, into, Divide Error and Single Step**
  - **Trap** generally means any processor generated interrupt
  - int x86, Trap usually means the **Single Step interrupt**
- **Software** Interrupt - Internal - from **int** or **into**
- **Hardware** Interrupt - External Uses **INTR** and **NMI**

Software Interrupts

- The **INT** instruction executes a software interrupt.
- The code that handles the interrupt is called an **interrupt handler**.
- The Interrupt Vector Table (**IVT**) holds a 32-bit segment-offset address for each possible interrupt handler.
- Interrupt Service Routine (**ISR**) is another name for interrupt handler.

Hardware Interrupts

- Generated by the Intel **8259 Programmable Interrupt Contoller** (**PIC**)
  - in response to a hardware signal

Interrupt Control Instructions

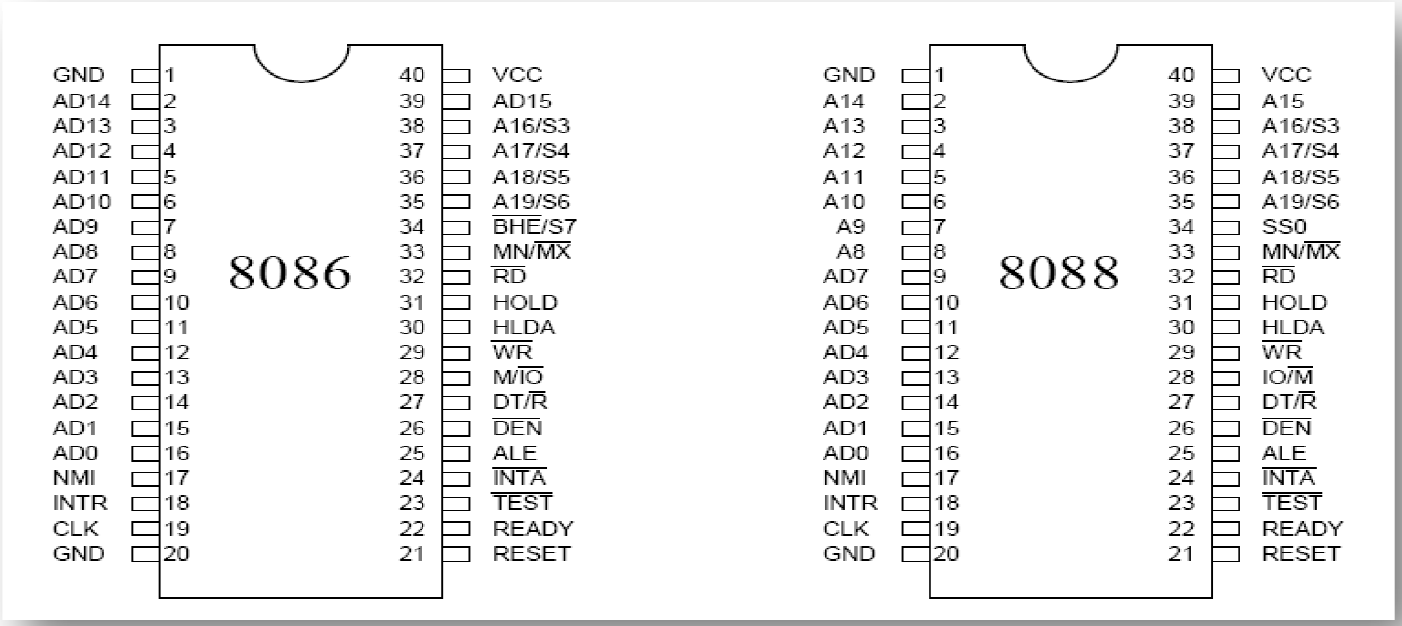
- **STI** – set interrupt flag
  - enables external interrupts

- always executed at beginning of an interrupt handler

➢ **CLI** – clear interrupt flag

- disables external interrupts
- used before critical code sections that cannot be interrupted

8086/8088 Pinout Diagrams



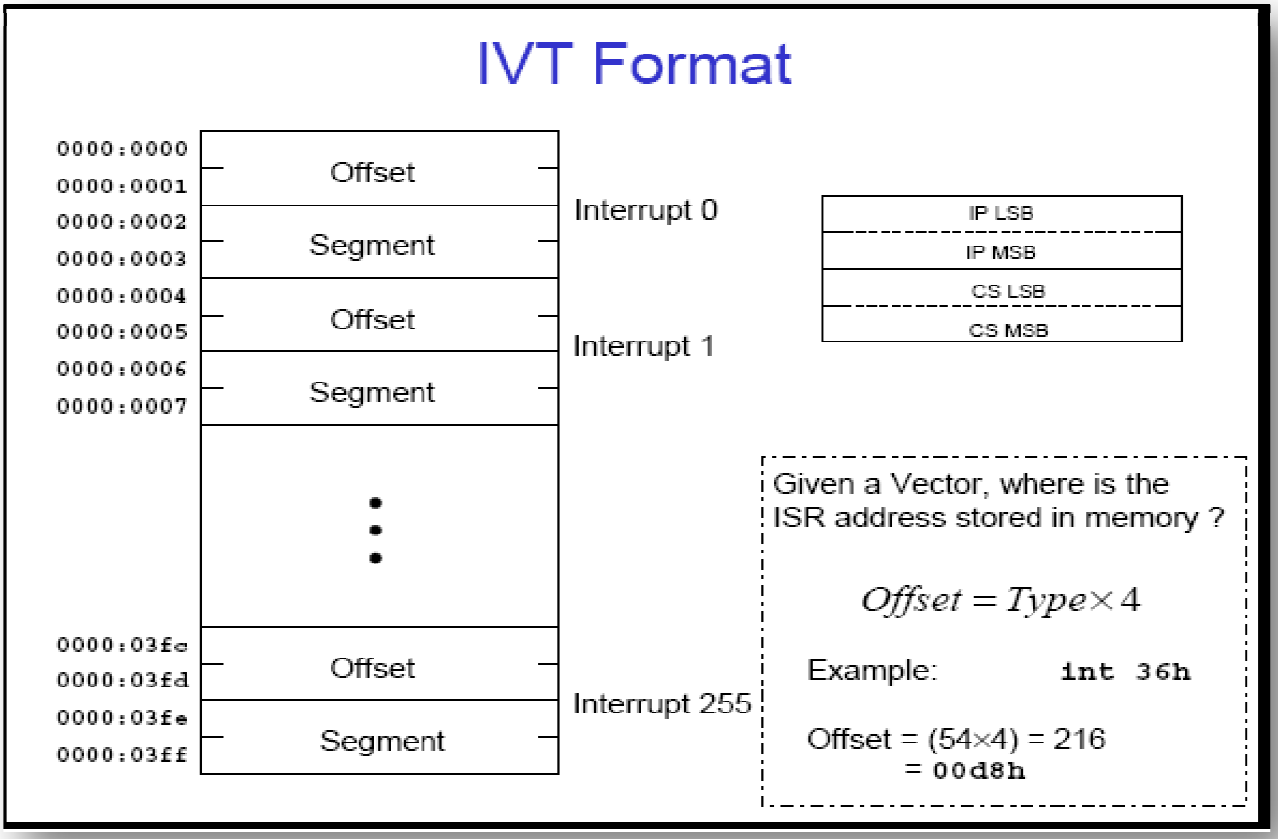
Interrupt Vector Table

➢ Each entry contains a 32-bit segment/offset address that points to an interrupt service routine

```
Offset = interruptNumber * 4
```

The following are only examples:

Interrupt Number	Offset	Interrupt Vectors
00-03	0000	02C1:5186 0070:0C67 0DAD:2C1B 0070:0C67
04-07	0010	0070:0C67 F000:FF54 F000:837B F000:837B
08-0B	0020	0D70:022C 0DAD:2BAD 0070:0325 0070:039F
0C-0F	0030	0070:0419 0070:0493 0070:050D 0070:0C67
10-13	0040	C000:0CD7 F000:F84D F000:F841 0070:237D



- • IVT Contains 256 Far Pointer Values
- – Far Pointer is CS:IP Values
- – Specified by Type Number or Vector

Redirecting Input-Output

- Input-output devices and files are interchangeable
- Three primary types of I/O:
  - Standard input (console, keyboard)
  - Standard output (console, display)
  - Standard error (console, display)
- Symbols borrowed from Unix:
  - < symbol: *get input from*
  - > symbol: *send output to*
  - | symbol: pipe output from one process to another
- Predefined device names:
  - **PRN** (printer), **CON** (keyboard, display), **LPT1** , **LPT2** (parallel printer), **NUL**, **COM1** , **COM2** (serial ports).
  - Standard input, standard output can both be redirected
  - Standard error cannot be redirected
  - Suppose we have created a program named myprog.exe that reads from standard input and writes to standard output. Following are MS-DOS commands that demonstrate various types of redirection:

```
myprog < infile.txt
myprog > outfile.txt
```

```
myprog < infile.txt > outfile.txt
```

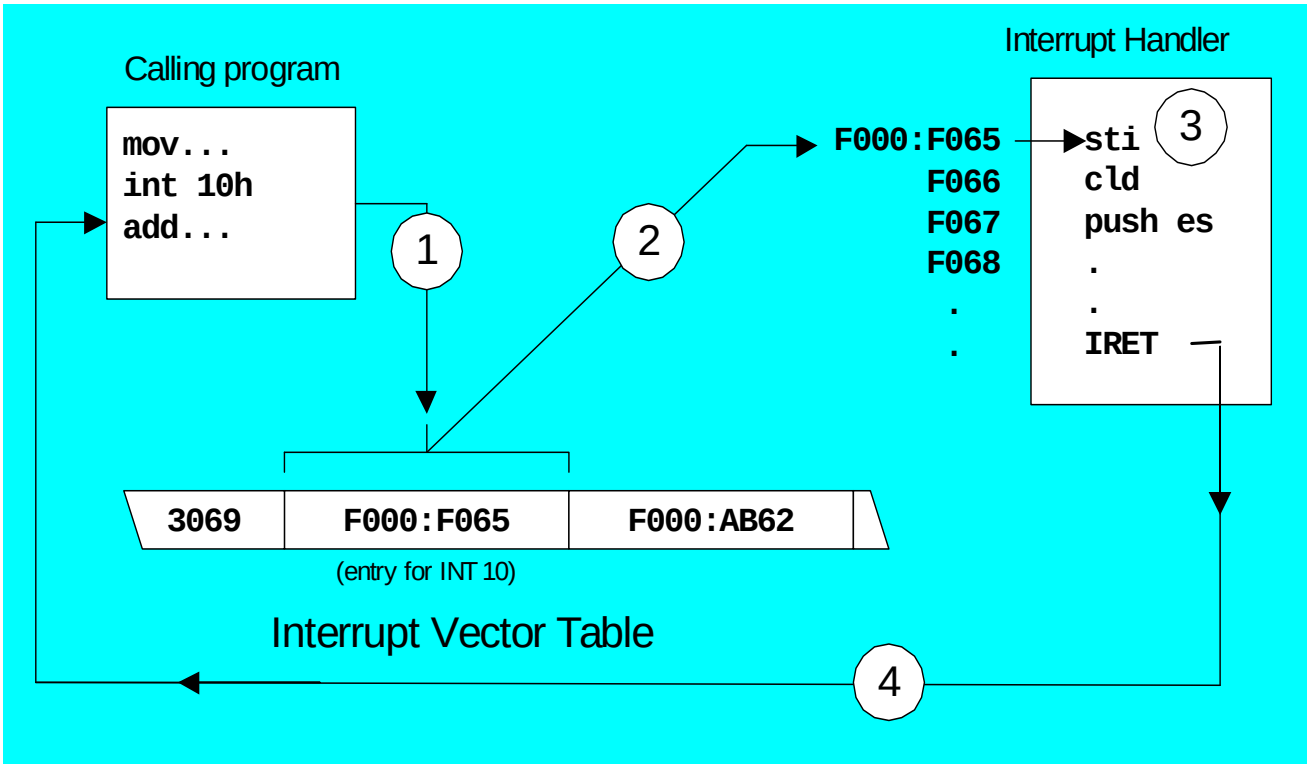
INT Instruction

- The INT instruction executes a software interrupt.
- The code that handles the interrupt is called an interrupt handler.
- Syntax:

```
INT number
(number = 0..FFh)
```

- The Interrupt Vector Table (**IVT**) holds a 32-bit segment-offset address for each possible interrupt handler.
- Interrupt Service Routine (ISR) is another name for interrupt handler.

Interrupt Vectoring Process



Common Interrupts

- INT 10h Video Services
- INT 16h Keyboard Services
- INT 17h Printer Services
- INT 1Ah Time of Day
- INT 1Ch User Timer Interrupt
- INT 21h MS-DOS Services