OBJECTIVES: At the end of this section, the student will be able to do the following:

- Discuss the functional hierarchy of H/W, BIOS, & DOS
- Describe the functions of the SYSTEM BIOS
- Discuss the functions of POST.
- Find the POST Error Codes in a technical reference.
- Discuss the use of BIOS Interrupts & the I/O services performed by BIOS.
- Discuss the use of basic DOS Functions.
- Describe the DOS BOOT sequences of events.
FUNCTIONAL HIERARCHY
BIOS AND DOS

This Chapter discusses BIOS and DOS.

The layered concept of an operating system is used to show how BIOS and DOS work together to control the hardware.
FUNCTIONAL HIERARCHY

APPLICATIONS

O/S: [BIOS / KERNEL / COMMAND PROCESSOR]

SYSTEM BIOS (RESIDENT)

HARDWARE
FUNCTIONAL HIERARCHY

The PC/AT has a "layered" operating system.

- BIOS provides the low-level interaction with the hardware.
- BIOS services perform I/O by directly addressing hardware registers on I/O chips.
- BIOS services provide a standard software interface that hide the hardware details from the O/S and applications.
- DOS operates at a higher level and manages "logical" organization of the system.
FUNCTIONAL HIERARCHY

The **BIOS** interacts directly with the hardware.

- It is generally associated with input and output devices and contains drivers for such devices as the hard disk, floppy drives, parallel ports, and serial ports to perform their hardware related tasks.

- The **SYSTEM BIOS** is called the *resident* portion and is built into each computer by the computer manufacturer.

- A secondary part of the BIOS is **non-resident** as it is read into main system memory (DRAM) from disk or network when a computer loads the operating system.
  - In MS-DOS this file is called *IO.SYS*
  - In PC-DOS it is called *IBMBIO.COM*. 
SYSTEM BIOS OVERVIEW

- ROM retains the stored code and data when the computer is turned off.
- ROM usually contains the following:
  - **POST--Power-On-Self-Test**
    - POST initializes installed components on the system board, tests the system hardware, detects the presence or absence of devices, reports system configuration and diagnostic status, and loads an operation system.
  - **BIOS--Basic Input Output System.**
    - Low level interface to devices used during run-time operations.
    - Isolates O/S and applications from the low-level hardware.
SYSTEM BIOS OVERVIEW

SYSTEM BIOS IS USED FOR:

- **POST--POWER-ON-SELF-TEST**
- **INITIALIZING THE SYSTEM**
  - MEMORY, DMA, PICs, PIT, KEYBOARD CONTROLLER
  - INTERRUPT VECTORS
- Configuring the System According to:
  - Built-in set-up program
  - Battery backed CMOS Configuration RAM
- Disk boot sector loading
- Storing graphics characters for certain graphics modes
SYSTEM BIOS OVERVIEW

SYSTEM BIOS IS USED FOR: (Cont.)

- Storing Low-Level I/O Routines such as
  - KEYBOARD SERVICES
  - CONSOLE SERVICES
  - VIDEO SERVICES
  - PRINTER SERVICES
  - DISK AND DISKETTE SERVICES
  - COM SERVICES

- Storing System Interrupt Routines such as
  - KEYBOARD INTERRUPT
  - TIMER INTERRUPT
  - REAL-TIME-CLOCK INTERRUPT
SYSTEM BIOS OVERVIEW

- When the system is reset, the processor fetches and executes the first instruction from address FFFF0H (FFFFFF0H for 80386 and later CPUs) in the ROM.
  - The first instruction is typically a FAR jump to the actual POST (POWER-ON-SELF-TEST) program.
- The CPU Executes the Code fetched from the SYSTEM BIOS and the POST is executed.
  - POST detects, checks, & initializes installed components on the system board.
  - POST writes a CODE to I/O Port 80 at the start of each new POST test.
SYSTEM BIOS OVERVIEW

POST normally stops on critical Failures.

- The LAST POST CODE written to Port 80 is an indication of the failure. The majority of failures will occur before POST has completed.

If everything is OK, POST gives control to a Bootstrap Loader to load an Operating System.

The bootstrap loader simply loads the boot sector and transfers control to it.

- If booting DOS, the boot sector program looks for the DOS files IO.SYS and MSDOS.SYS.
- Finally (if booting DOS), COMMAND.COM is executed, producing a prompt on the screen.
Power-On-Self-Test
POST (Power-On-Self-Test)

- The POST (POWER-ON-SELF-TEST) runs through a series of tests to determine whether the machine is functional.

- POST indicates errors in hardware or software by these messages:
  - If the video is operational, POST will beep and display a message on the console.
  - If the video is not initialized, POST sounds a series of beeps.
  - A POST CARD (Port 80H Decoder) may be used to display the current POST CODE on LED’s.
Post and Boot Sequence

This Information taken from a generic PC System Board POST (PCI INITIALIZATION NOT INCLUDED)

1 - POST CHECKS THE SYSTEM

- CPU REGISTERS
- KEYBOARD CONTROLLER
- REAL-TIME CLOCK READ/WRITE CHECK
- SYSTEM BIOS CHECKSUM
- PROGRAMMABLE INTERVAL TIMER
- DMA CHIPS AND PAGE REGISTERs
- RAM REFRESH
- FIRST 64K RAM TEST AND PARITY TEST
Post and Boot Sequence (Cont.)

1 - POST CHECKS THE SYSTEM (Cont.)

- INITIALIZE MASTER AND SLAVE DMA
- INITIALIZE MASTER AND SLAVE PIC
- INITIALIZE (LOADS) INTERRUPT VECTORS
- REAL-TIME CLOCK CHECKSUM & configuration
- SCREEN INITIALIZATION AND ROM SEARCH
- POST THEN CHECKS AND INITIALIZES THE REMAINING MEMORY.

2 - JUMP TO THE BOOTSTRAP LOADER.

3 - GIVE CONTROL TO THE BOOT SECTOR CODE.

- IF DOS, GIVE CONTROL TO COMMAND.COM.
Post and Boot Sequence (Cont.)

This is a sample of the beginning POST Sequence for AMI BIOS (PORT80H.DOC).

- POST Loader starts.
- Reset Initialization.
- ChipSet Initialization (PCI initialization).

0D0h

- Initialize DS, ES, GS and FS.
- Check if keyboard system bit is set.
- Check whether a hard or soft reset has occurred.
Post and Boot Sequence (Cont.)

Sample of PORT80H.DOC

- **0D1h**
  - Power On Initialization
  - Initialize special ChipSets in power on / hard reset
    - Check Cache size and type
    - Write reserved Cache size information to CMOS
    - Determine CPU speed (optional)

- **0D2h**
  - Disable NMI Reporting

- **0D3h**
  - Reset Video Adapter
Post and Boot Sequence (Cont.)

Sample of PORT80H.DOC

- **0E1h**
  - Preset any defaults needed to ChipSet registers
  - Start the refresh timer(s) running
  - Size all L2/L3 Cache (if present/required)
  - Detect EDO SIMM, etc.

- **0E2h**
  - Initialize timer channel 2 for speaker

- **0E3h**
  - Initialize timer channel 0 for system timer

- etc ...
POST (Power-On-Self-Test)

- **BEEP CODES** are generated on many versions of BIOS in addition to the Diagnostic Code written to PORT 80.
  - BEEP CODES are useful when you do not have a POST Card or other Diagnostic Tool.
  - BEEP CODES are BIOS VENDOR dependent.
- The system board technical reference gives a complete list of beep codes for the version of POST in your machine.
- The next page gives an illustration of the beep codes.
  - Notice also that Port 80H is used to display diagnostic information.
BEEP CODE EXAMPLES

The following are examples of BEEP CODES from Phoenix BIOS.

Port 80 -> BEEP CODE

- 01H -> None -> CPU Register Test in Progress
- 02H -> 1-1-3 -> Real-Time Clock Write/Read Failure
- 03H -> 1-1-4 -> SYSTEM BIOS Checksum Failure
- 04H -> 1-2-1-> Programmable Interval Timer Failure
- 08H -> 1-3-1-> RAM Refresh Verification Failure
- 0AH -> 1-3-3 -> 1St 64K Ram Failure
- 27H -> 3-2-4 -> Keyboard Controller Test Failure
- 2BH -> 3-3-4 -> Screen Memory Test Failure
POST MESSAGES

- If the display is working, POST will beep and put a message on the screen.

- **Message: Floppy Disk Subsystem Reset Failed**
  - Possible Cause: The floppy disk drive control cable has failed.
  - Solution: Check the floppy disk drive control cable. If the problem persists, contact your Service Representative.

- **Message: Hard Disk Configuration Error**
  - Possible Cause: The specified configuration is incorrect.
  - Solution: Rerun the set-up program and enter the correct fixed disk drive type number. If the problem persists, contact your Service Representative.
POST MESSAGES

Message: Gate A20 Failure

- Possible Cause: The computer cannot switch into protected mode.
- Solution: Contact your Service Representative.

NOTE: It would be nice if POST could tell us the source of every problem. The hard work comes when POST can't run; for example, if the FLASH can't be accessed.
BIOS Interrupts
BIOS INTERRUPTS

- Generally, all the interrupt-type numbers from 0 to 1FH are considered BIOS interrupts.
- The purpose of interrupts is to suspend the execution of the current program and execute an interrupt handler (ISR).

  - The ISRs reside (at least initially) within the BIOS. Some BIOS interrupts are re-vectored. Examples include:
    1) Expansion ROM BIOS such as Video may take over the INT 10H vector from the System BIOS during the POST.
    2) Operating systems point to their own handlers which are usually more robust and often add functionality to an existing interrupt handler.
BIOS INTERRUPTS

Interrupts fall into three broad categories:

1. An **Exception** is an event which typically occurs when a CPU operation causes an error. It is a form of internally generated interrupt that generally results when the processor attempts to execute an instruction. For example, an attempt to DIVIDE BY ZERO.

2. **Software interrupts** providing services to DOS and application programs. These include most of those listed from 10H to 1FH & are issued with the “INT (x)” instruction.

3. **Hardware interrupts** provide support to system peripherals. These include all interrupts signals connected to the PICs and the NMI.
BIOS INTERRUPTS

There are two types of SOFTWARE CONDITIONS that can cause an interrupt to an Intel x86 microprocessor.

- **Exceptions.**
  - The Interrupt Type Number is set by the Processor itself.

- **Software interrupts**
  - Software interrupts generated by the INT (interrupt) instruction.
  - This instruction take the following format: INT 10
BIOS INTERRUPTS

Exceptions: The Interrupt Type Number is generated within the CPU by microcode instructions.

- Exceptions occur when the CPU attempts to execute an instruction which results in an error.
- A example would be an attempt to divide a number by zero.
- Some other Exception examples include:
  - Invalid OP CODE (Type 6);
  - General Protection Error (Type 0Dh)

» NOTE: Some protected mode exceptions Type Numbers overlap the definitions adopted by IBM, which have become a standard. To eliminate possible conflicts, the exception handlers (ISRs) have the responsibility of determining if this was caused by an exception or hardware interrupt.
BIOS INTERRUPTS

Software interrupts

- Software interrupts are under the control of the programmer. This instruction takes the following Assembly Language format: **INT 13**

- The interrupt instruction is a special type of indirect call that allows the programmer to invoke a subroutine without knowing the address of the function called.

- When the microprocessor executes a Software Interrupt, the process is similar to a hardware interrupt request received on the maskable interrupt request line (INTR).
USING BIOS SERVICES--INTRODUCTION

- BIOS services provide a way of testing some of the peripheral chips.
  - Technicians can use these built in routines with an ITP instead of writing routines from scratch.
    - e.g. - Test Video (INT 10)
    - e.g - Test Floppy Drive (INT 13)

- All BIOS services are used the same way:
  1. We must set up a function number in the AH register.
  2. We may need to set up other information in other registers.
  3. We execute the INT instructions.
## BIOS INTERRUPTS

### BIOS INTERRUPTS-- LISTED BY TYPE NUMBER

- ** INDICATES INTEL DEFINED

<table>
<thead>
<tr>
<th>TYPE #</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H</td>
<td>DIVIDE BY ZERO **</td>
</tr>
<tr>
<td>01H</td>
<td>DEBUG **</td>
</tr>
<tr>
<td>02H</td>
<td>NON-MASKABLE **</td>
</tr>
<tr>
<td>03H</td>
<td>SOFTWARE BREAK POINT **</td>
</tr>
<tr>
<td>05H</td>
<td>PRINT SCREEN</td>
</tr>
</tbody>
</table>
BIOS INTERRUPTS (Cont.)

BIOS INTERRUPTS-- LISTED BY TYPE NUMBER

- The following are hardware interrupts generated by PIC#1

<table>
<thead>
<tr>
<th>TYPE #</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>08H</td>
<td>SYSTEM TIMER (TOD)</td>
</tr>
<tr>
<td>09H</td>
<td>KEYBOARD INTERRUPT</td>
</tr>
<tr>
<td>0BH</td>
<td>COM2</td>
</tr>
<tr>
<td>0CH</td>
<td>COM1</td>
</tr>
<tr>
<td>0DH</td>
<td>ALTERNATE PRINTER</td>
</tr>
<tr>
<td>0EH</td>
<td>DISKETTE</td>
</tr>
<tr>
<td>0FH</td>
<td>PRINTER</td>
</tr>
</tbody>
</table>
## Hardware Interrupts

### PIC #1 Ports

<table>
<thead>
<tr>
<th>Type</th>
<th>ISA Bus</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRQ0</td>
<td>08H</td>
<td>TOD</td>
</tr>
<tr>
<td>IRQ1</td>
<td>09H</td>
<td>KEYBOARD</td>
</tr>
<tr>
<td>(IRQ2)</td>
<td>0AH)</td>
<td>(X)</td>
</tr>
<tr>
<td>IRQ3</td>
<td>0BH</td>
<td>COM2</td>
</tr>
<tr>
<td>IRQ4</td>
<td>0CH</td>
<td>COM1</td>
</tr>
<tr>
<td>IRQ5</td>
<td>0DH</td>
<td>ALT PRINTER (HARD DISK)</td>
</tr>
<tr>
<td>IRQ6</td>
<td>0EH</td>
<td>DISKETTE</td>
</tr>
<tr>
<td>IRQ7</td>
<td>0FH</td>
<td>PRINTER</td>
</tr>
</tbody>
</table>

### PIC #2 Ports

<table>
<thead>
<tr>
<th>Type</th>
<th>ISA Bus</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRQ8</td>
<td>70H</td>
<td>REALTIME CLK</td>
</tr>
<tr>
<td>IRQ9</td>
<td>71H</td>
<td>REDIRECT TO 0AH</td>
</tr>
<tr>
<td>IRQ10</td>
<td>72H</td>
<td>FREE</td>
</tr>
<tr>
<td>IRQ11</td>
<td>73H</td>
<td>FREE</td>
</tr>
<tr>
<td>IRQ12</td>
<td>74H</td>
<td>FREE (PS/2 Mouse)</td>
</tr>
<tr>
<td>IRQ13</td>
<td>75H</td>
<td>CO-PROCESSOR</td>
</tr>
<tr>
<td>IRQ14</td>
<td>76H</td>
<td>HARD DISK</td>
</tr>
<tr>
<td>IRQ15</td>
<td>77H</td>
<td>FREE</td>
</tr>
</tbody>
</table>

### IRQ2 Redirect Note

IRQ9 ISR (Type 71) invokes S/W Interrupt 0AH (old IRQ2 was Type 0AH)
BIOS INTERRUPTS (Cont.)

- BIOS INTERRUPTS-- LISTED BY TYPE NUMBER

- The following are **software services** available to DOS and Applications

<table>
<thead>
<tr>
<th>TYPE #</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>10H</td>
<td>VIDEO SERVICES</td>
</tr>
<tr>
<td>11H</td>
<td>EQUIPMENT DETERMINATION</td>
</tr>
<tr>
<td>12H</td>
<td>MEMORY SIZE DETERMINATION</td>
</tr>
<tr>
<td>13H</td>
<td>DISK AND DISKETTE I/O</td>
</tr>
<tr>
<td>14H</td>
<td>RS232 I/O</td>
</tr>
<tr>
<td>15H</td>
<td>SYSTEM SERVICE ROUTINES</td>
</tr>
<tr>
<td>16H</td>
<td>KEYBOARD I/O</td>
</tr>
</tbody>
</table>
**BIOS INTERRUPTS (Cont.)**

- BIOS INTERRUPTS-- LISTED BY TYPE NUMBER

The following are *software services* available to DOS and Applications

<table>
<thead>
<tr>
<th>TYPE #</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>17H</td>
<td>PRINTER I/O</td>
</tr>
<tr>
<td>19H</td>
<td>SYSTEM BOOT</td>
</tr>
<tr>
<td>1AH</td>
<td>CLOCK SERVICES</td>
</tr>
<tr>
<td>1BH</td>
<td>CTRL-BREAK</td>
</tr>
<tr>
<td>1CH</td>
<td>TIMER-TICK, A HOOK FOR APPLICATIONS</td>
</tr>
<tr>
<td>1EH</td>
<td>DISKETTE PARAMETER TABLE</td>
</tr>
<tr>
<td>1FH</td>
<td>GRAPHICS CHARACTER EXTENSIONS</td>
</tr>
</tbody>
</table>
**BIOS INTERRUPTS (Cont.)**

**BIOS INTERRUPTS-- LISTED BY TYPE NUMBER**

- The following are hardware interrupts generated by PIC#2

<table>
<thead>
<tr>
<th>TYPE #</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>70H</td>
<td>REAL-TIME CLOCK (Alarm)</td>
</tr>
<tr>
<td>71H</td>
<td>Redirected Type 0Ah for PC/ XT compatibility</td>
</tr>
<tr>
<td>75H</td>
<td>COPROCESSOR ERROR</td>
</tr>
<tr>
<td>76H</td>
<td>FIXED DISK</td>
</tr>
</tbody>
</table>
BIOS SERVICES EXAMPLES

Example 1 -- Getting a character from the keyboard

• MOV AH,00       ;RETURN A CHARACTER
• INT 16H           ;AL WILL HAVE CHARACTER

Example 2 -- Initializing the COM PORT to
Baud Rate = 2400; No Parity; 1 Stop Bit; 8 Data Bits

• MOV AH,00       ;INITIALIZATION FUNCTION
• MOV AL,10100011B ;2400, NP,1 S, 8 DATA
• MOV DX,0         ;COM1
• INT 14H           ;BIOS INTERRUPT
BIOS SERVICES EXAMPLES

Example 3 -- Change to a Graphics Video Mode

- MOV AH, 0H ; Set Video Mode Function
- MOV AL, 10H ; Set Video Mode 10H (640x480)
- INT 10H ; Now using A0000p Video Buffer

Example 4 -- Writing a character to the Console

- MOV AH, 0EH ; Write and Move Cursor Function
- MOV AL, 'A' ; Character to Output - ‘A’ = 41H
- INT 10H ; BIOS Interrupt
BIOS SERVICES EXAMPLES

Example 5 -- Read Diskette Boot Sector into memory (DMA Transfer).

- MOV AX,800 ;Buffer area defined by ES:BX
- MOV ES,AX ;Data will be put in 800:25 (8025 Physical)
- MOV BX,25 ;
- MOV AH,02 ;Function 2 of INT 13 is READ SECTOR
- MOV AL,01 ;AL= # of Sectors to read = 1
- MOV CX,1 ;CH = Track # 0, CL= Sector #1
- MOV DX,0 ;DX= Drive #0 = A:
- INT 13 ;Transfer Boot Sector to 800:25
DOS FUNCTIONS
INTRODUCTION TO DOS FUNCTIONS

- BIOS service routines have a standard software interface that hides the details of the hardware from DOS and applications.
- DOS operates at a higher level and manages "logical" organization of the system.
- Typical DOS activities are these:
  - Character I/O functions
    - KEYBOARD; DISPLAY; PRINTER; COM PORTS
  - Managing various switches and other information
    - BREAK; VERIFY; DATE/TIME; DOS VERSION NUMBER
DOS FUNCTION CALLS

For the sake of completeness, we will briefly look at DOS function calls.

- Like BIOS, DOS provides services (function calls) via the software interrupt instruction INT 21H.
- Also like BIOS, DOS requires a function number in the AH register and possibly other numbers in other registers.
- DOS functions are easy to use, although probably not relevant to the task of finding a problem with a computer. BIOS functions would more likely be useful.
- A few typical DOS function calls are shown on the following pages.
Use of DOS Function Calls

Example 1 -- Get a character from the keyboard and echo to screen

- MOV AH, 1 ; FUNCTION NUMBER
- INT 21H ; DOS INTERRUPT
- AL = CHARACTER

Example 2 -- Send a character to the Console

- MOV AH, 2 ; FUNCTION NUMBER
- MOV DL, 'R' ; OUTPUT AN 'R'
- INT 21H ; DOS INTERRUPT
Use of DOS Function Calls (Cont.)

Example 3 -- Display a message; message must end with a '$' character

- MOV AX,SEG MESSAGE1 ;Put the segment of the
- MOV DS,AX ; Message in DS
- MOV DX,OFFSET MESSAGE1 ;Offset goes in DX
- MOV AH,9 ;Function Number
- INT 21H ;DOS Interrupt
DOS BOOT PROCESS
DOS BOOT PROCESS

- The “BOOT SECTOR” is located on a system disk (floppy disk for our discussion) in logical sector 0.
- The BOOT SECTOR contains critical information regarding the disk’s physical characteristics.
- The major sections in the BOOT SECTOR are:
  - The first byte in the boot sector is always an 80 x 86 jump instruction to the bootstrap code in the final section.
  - The next section is where an OEM computer software manufacturer’s name and version can be found.
  - The next section contains information about the disk’s physical characteristics which is needed by MS-DOS.
  - The final section in the boot sector contains the disk bootstrap.
DOS BOOT PROCESS

The process of loading DOS is outlined below:

- It is a high level view starting from power up.

  1 - Power up.

  2 - POST and hardware initialization.

Step 3 is accomplished via INT 19 at the end of the POST and is considered the beginning of the DOS Boot process.

- INT 19 is a software interrupt invoked via the “INT 19” instruction imbedded in the BIOS EPROM's.

  3A - Check for system disk.
DOS BOOT PROCESS (Cont.)

3B- Load bootstrap routine from the system disk into DRAM.

- Read diskette track 0, head 0, sector 1 into DRAM starting at physical address 00007C00.
  - If unable to load the disk bootstrap routine an INT 18 is often performed.

3C - Transfer program flow to disk bootstrap loader contained in DRAM at address 7C00.

- Check the data at address 7C00. If it is not a boot diskette, the error message “NON SYSTEM DISK” appears.
- Jump to physical address 7C00 and begin program execution. This is the first stage of the boot-strap loader.
DOS BOOT PROCESS (Cont.)

4 - Bootstrap Loader loads system files.
5 - Program control transferred to the IO.SYS file.
6 - IO.SYS calls the SYSINIT module.
7 - SYSINIT calls MSDOS.SYS.
8 - MSDOS.SYS initializes the interrupt vectors.
9 - If a CONFIG.SYS file exists, then it is called and any drivers designated within CONFIG.SYS are loaded.

- If CONFIG.SYS exists, modify DOS parameters and install user-specified device drivers.
- Otherwise, use default DOS values.
DOS BOOT PROCESS (Cont.)

10 - SYSINIT loads COMMAND.COM as the default shell (Command Interpreter).

11 - If AUTOEXEC.BAT is present, then perform the commands in AUTOEXEC.BAT
   • Otherwise, issue DOS DATE and TIME commands.

12 - SYSINIT is then of no more use so it gets discarded.

13 - COMMAND.COM displays the DOS prompt (A:) and waits for user input.
   • The PC is now ready for user input.
SUMMARY

WE HAVE DISCUSSED THE FOLLOWING:

- The functional hierarchy of H/W, BIOS, & DOS
- The functions of the SYSTEM BIOS
- The functions of POST and some details of the initial sequence.
- The POST Error Codes
- The use of BIOS Interrupts & the I/O services performed by BIOS.
- The use of basic DOS Functions.
- The DOS BOOT sequences of events.