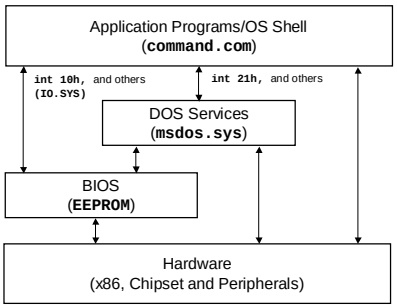
Interrupt Services



BIOS - Basic Input Output System
DOS - Disk Operating System

Which Way is Best ?

	Portability	Speed
Direct `in`, `out`	Least	Most
BIOS	Average	Average
DOS	Most	Least

- Direct is Fastest but Least Portable

Example: Many Draw programs use Direct for Speed

  - Must Provide Support for ALL Video Cards
  - Attempt to Standardize Video Card Drivers
- BIOS Has Average Portability Since Few Versions
  - AWARD, AMI (others DELL, Mr. BIOS, ACER, etc.)
  - Typically Lower Level Services than DOS
  - Faster Since Closer to HWE
- DOS Services are Slowest but Most Portable
  - Huge Percentage of PCs run MS-OS
  - DOS ISRs Generally Invoke BIOS ISRs

Characteristics

- BIOS and DOS Services can be Called from ANY Program
  - **IVT** contains FAR Pointers
- BIOS (and most DOS) ISRs Preserve Register Contents
  - Except **ax**
- Most DOS Services Return Error Code in **ax**
  - Error Code Present if **CF=1**
- DOS and BIOS Services are Interfaces to Devices
  - Collection of **DEVICE DRIVERS**
  - Some Specialized HWE May Not Have DOS/BIOS Service
  - A **DEVICE DRIVER** Must be Used

Customized Services

- Some Devices Do Not have Services in BIOS or MSDOS.SYS
- Must Write a Customized ISR for Device and Load at Boot Time
  - *Device Drivers*
- DOS Services and BIOS Services are Standard Device Drivers
  - DOS is Actually Little More than a Collection of ISRs
  - Clone of the CP/M OS with Shell Similarities to UNIX

DOS/BIOS Services were Created to Aid Programmers in IBM PC

Interrupt Classification by Type Number

Processor	00h-04h 06h-07h
BIOS	05h 10h-1fh 40h-5fh 70h-85h
DOS	20h-3fh

- Remaining Used by DOS/BIOS Extensions or User
- 10 DOS Services Intended for Programmer Access

DOS Service Interrupts

Type	*Description*	*Intention*
20h*	Program Terminate	System Service
21h*	General Services	System Service
22h	Terminate Address	User Can Specify
23h	CTRL-C Handler	User Can Specify
24h	Critical Error Handler	User Can Specify
25h*	Absolute Disk Read	System Service
26h*	Absolute Disk Write	System Service
27h*	Terminate and Stay Resident	System Service
28h	DOS Idle	User Can Specify
2fh	Multiplex Interrupt	Mem Resident Communication

2fh Allows Memory Resident Programs to Communicate (Mailbox)
** Intended to be Invoked using* `int` *Instruction*

DOS System Service Interrupt – **20h**

PROGRAM TERMINATE SERVICE

- Original Service to Pass Control Back to DOS
- Later versions of DOS Added More Functionality
  - Automatically Close Opened Files
  - Free up Heap Memory
- Later PROGRAM TERMINATE Services
  - **int 21h**, func **00h**, **0fh**, **16h**, 31h or **4ch**
- Should Use **int 21h**, func **4ch** unless

Compatibility with Early DOS Versions Required

DOS System Service Interrupts – **25h**, **26h**

ABSOLUTE DISK READ, WRITE

- Read/Write Specific PHYSICAL Sectors
  - Ignores the Logical Structure
- Parameters Present in **al**, **cx**, **bx**, **ds**
  - **al** Indicates the Disk
  - **ds:bx** Points to Memory Location (data to read/write)
  - **cx** Contains Number of Sectors
  - Free up Heap Memory

al	Disk Drive
00h	A:
01h	B:
02h	C:

- Result Code Returned in al, ah
  - **CF=1** Indicates Error, **CF=0** Indicates No Error

DOS System Service Interrupt – **27h**

TERMINATE AND STAY RESIDENT (TSR)

- Ends Program Like **int 20h** But Leaves Portion in Memory
  - Better to Use **int 21h**, func **31h** Unless Need Compatibility
- TSR Programs Designed in 2 Parts
  1) Resident – Initializes Data and Calls **int 27h**
  2) Transient – Loads During Event and Executes

DOS Multiplex Interrupt – **2fh**

MULTIPLEX HANDLER

- Allows Communication Between TSR Programs
  - Better to Use **int 21h**, func **31h** Unless Need Compatibility
- Each TSR has Unique ID Number
  - Mailbox Number
- **PRINT.EXE** (DOS Print Spooler) Uses This

DOS User Interrupts – **22h, 23h, 24h, 28h**

"ADDRESS INTERRUPTS"
Intended For User to Create ISRs and "hook" Vector, not to Use **int**

- Handle 3 Types of Exceptions:
  1) End of a Program
  2) "break" Action – CTRL-C or CTRL-BREAK
  3) Critical Errors (Typically a Disk Error)
  - Better to Use **int 21h**, func **31h** Unless Need Compatibility
- Type 22h – Terminate Address
  - Specifies Address to Transfer Control to When int 20h, 27h or 21h func 00h, 0fh, 16h, 31h or 4ch occurs
- Type 23h – CTRL-C (CTRL-BREAK) Handler
- Type 24h – Critical Error
  - Default Produces "Abort, Retry, Ignore?" or "Abort, Retry, Fail?" (3.3)
- Type 28h – DOS Idle
  - Used by DOS When Waiting for Event (I.e. "wait" for keystroke)

DOS System Service Interrupt – **21h**

GENERAL SERVICES

- Function Specified in ah – Over 100!!!!
- Can Be Grouped Into Categories
  1) I/O Services – Char. I/O Only – Keyboard, Mon., Port (eg. COM1)
  2) Printer Services – func 5 only – output ASCII in dl to LPT1
  3) Disk Services – Read, Write, Open, Close, Mod., Structure, etc.
  4) System Services – Set "hooks", Device Information, etc.
  5) Network Services – Get Name, Redirect to Remote Device
  6) Date/Time Services – Retrieve and Format from BIOS

FUNCTION	DESCRIPTION
2ah	Get Date
2bh	Set Date
2ch	Get Time
2dh	Set Time

al Contains Day (0-6) **dh** Contains Month (1-12)

cx Contains Year (1980-2099) **dl** Contains Day (1-31)

ROM BASIC Interrupt - Type 18

BIOS INTERRUPT

- This Short MASM Program Accesses the ROM BASIC

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                       ;
; Example program that invokes the BASIC interpreter                    ;
; that was present in a ROM on the motherboard of                       ;
; the original IBM PCs, XTs, ATs and some other                         ;
; later models, but is generally not there in later                     ;
; AT clone type machines.                                               ;
;                                                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

int18st SEGMENT STACK 'STACK'
   DB 100h DUP(?)
int18st ENDS

int18cd SEGMENT 'CODE'
STRT:
   int  18h
int18cd ENDS
END STRT
```

Divide Error Interrupt - Type 0

PROCESSOR INTERRUPT

- This Short MASM Program Intentionally Does a Divide-by-0

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                       ;
; Example program that intentionally does a divide                      ;
; zero in order to invoke the int 0 interrupt.                          ;
;                                                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
int0st SEGMENT STACK
       DB 100h DUP(?)
int0st ENDS

int0cd SEGMENT
STRT:
   xor  ax, ax
   mov  bx, 0001h
   div  bx
   mov  ah, 4ch
   int  21h
int0cd ENDS
END STRT
```

IVT Entry Interchange (*hooking*)

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                       ;
; Example program that copies the interrupt vector                      ;
; in the IVT for the ROM BASIC service (int 18h)                        ;
; into the location for the Divide Error trap                           ;
; (int 0).  It then intentionally does a divide-by-0                    ;
; causing the ROM BASIC ISR to be invoked.                              ;
;                                                                       ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
swapst SEGMENT STACK
       DB 100h DUP(?)
swapst ENDS

swapcd SEGMENT
STRT:
   xor  ax, ax                    ;Clear ax
   mov  es, ax                    ;Set up es to contain segment 0000
   ;Interrupt 18h is the ROM BASIC Service
   mov  cl,  2                    ;Prepare to shift left twice
   mov  si,  0018h                ;Interrupt 18h in si
   shl  si,  cl                   ;Compute true offset by mult. by 4
   cli                            ;Disable interrupts
   mov  ax,  WORD PTR es:[si]     ;Read int 18h vector offset into ax
   mov  bx,  WORD PTR es:[si+2]   ;Read int 18h vector segment into bx
   ;Interrupt 00 is the Divide overflow handler
   mov  WORD PTR es:[0000],  ax   ;int 00h offset <- int 18h offset
   mov  WORD PTR es:[0002],  bx   ;int 00h segment <- int 00h segment
   sti                            ;Enable interrupts
   mov  bx, 0001h                 ;bx <- 1
   xor  ax, ax                    ;ax <- 0
   div  bx                        ;bx <- 1/0 --OVERFLOW!!!!!
swapcd ENDS
END STRT
```

BIOS - 3 Major Parts

1) Power-On Self-Test (POST) Program

*reset vector (**ffffh:0000h**) points to POST*

a) Reset, Power-ON
b) HWE Reset Button (on MB)
c) Warm-Boot (Ctrl-Alt-Del)
skips some of POST
d) POST invokes **int 19h** as final command
e) Searches for Bootable Disk
f) Copies Boot Sector into 0000:7c00 *OR* **int 18h** ROM BASIC
g) Sets CS:IP to Point to Bootstrap in Memory

2) Bootstrap Program

a) Copies **MSDOS.SYS**, **IO.SYS** and **COMMAND.COM** into memory
b) Transfers **CS:IP** to First Instruction in **COMMAND.COM**
c) **MSDOS.SYS** and **IO.SYS** just reside there
d) **COMMAND.COM** configures itself,
generates prompt and then is a loader
e) During **COMMAND.COM** config, Device Drivers loaded via **config.sys**
f) io.sys Interface to BIOS Routines
g) msdos.sys Contain DOS ISR Code

3) Input/Output Routines

Contains BIOS ISR Code

BIOS – Service Routines

- Compatibility – OS Can Change, BIOS Change not as Likely
- 12 Basic BIOS Services – Categorized in 5 Groups

1)	Peripheral Devices	(10h, 13h, 14h, 15h, 16h, 17h)
2)	Equipment Status	(11h, 12h)
3)	Time/Date Service	(1ah)
4)	Print Screen Key	(05h)
5)	Special Services	(18h, 19h)

- Many ISRs Have Several Different Functions

eg. int 10h (video) has 25 functions