Microprocessors & Interfacing

Interrupts

BITS Pilani

Dr. Gargi Prabhu
Department of CS & IS

Interrupt vectors defined by Intel

innovate achieve lead

Number	*Address*	*Microprocessor*	*Function*
0	0H–3H	All	Divide error
1	4H–7H	All	Single-step
2	8–BH	All	NMI pin
3	CH–FH	All	Breakpoint
4	10H–13H	All	Interrupt on overflow
5	14H–17H	80186–Core2	Bound instruction
6	18H–1BH	80186–Core2	Invalid opcode
7	1CH–1FH	80186–Core2	Coprocessor emulation
8	20H–23H	80386–Core2	Double fault
9	24H–27H	80386	Coprocessor segment overrun
A	28H–2BH	80386–Core2	Invalid task state segment
B	2CH–2FH	80386–Core2	Segment not present
C	30H–33H	80386–Core2	Stack fault
D	34H–37H	80386–Core2	General protection fault (GPF)
E	38H–3BH	80386–Core2	Page fault
F	3CH–3FH	—	Reserved
10	40H–43H	80286–Core2	Floating-point error
11	44H–47H	80486SX	Alignment check interrupt
12	48H–4BH	Pentium–Core2	Machine check exception
13–1F	4CH–7FH	—	Reserved
20–FF	80H–3FFH	—	User interrupts

Interrupt Instructions

- The microprocessor has three different interrupt instructions that are available to the programmer:

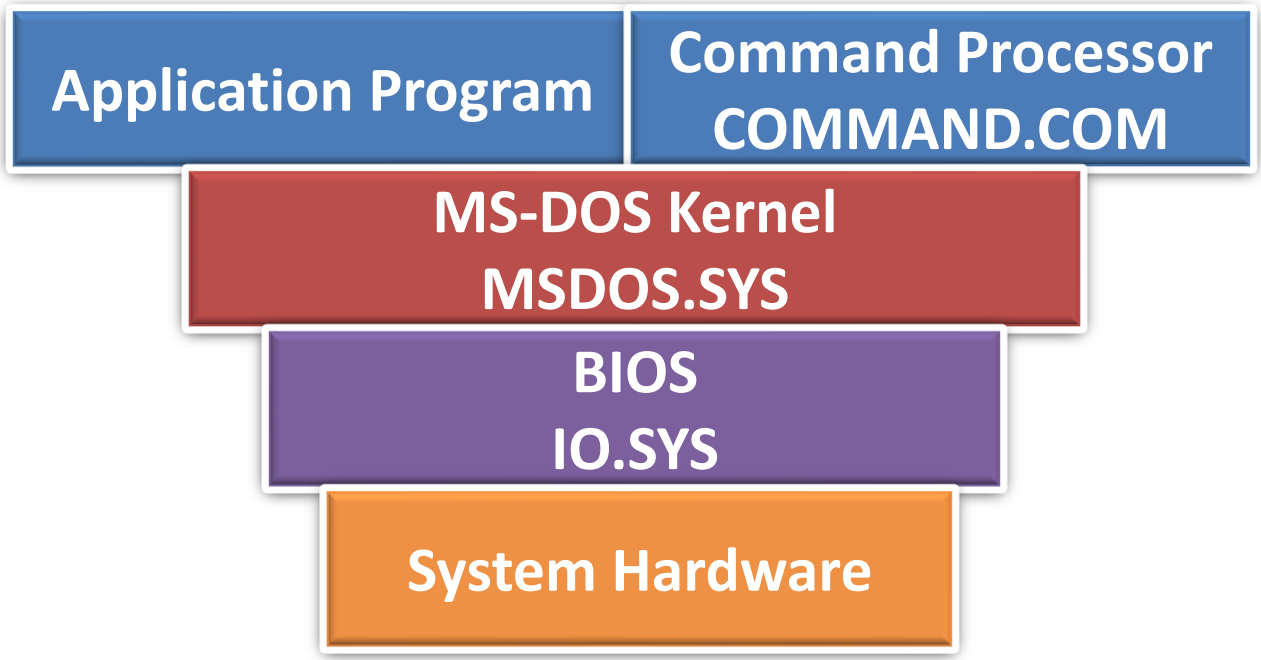
  INT , INTO , INT 3

- Each of these instructions fetches a vector from the vector table, and then calls the procedure stored at the location addressed by the vector.

BIOS

- BIOS (Basic Input/Output System) is firmware embedded on a motherboard that initializes and controls hardware during the boot process of a computer.
- It provides a standardized way for software to interact with hardware components such as the keyboard, display, disk drives, and other peripherals.
- Hardware-specific, provided by manufacturer
- Resident portion –> ROM
- IO.SYS -> adding new features to BIOS

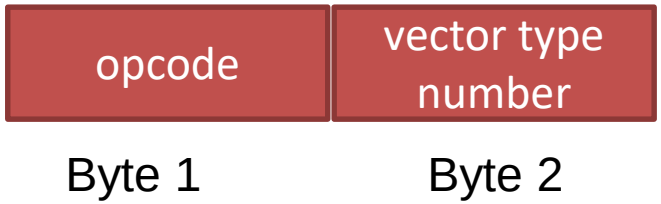
MS-DOS Functions and BIOS Calls



INT

- There are 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).
- The address of the interrupt vector is determined by multiplying the interrupt type number by 4.
- INT 10H instruction calls the interrupt service procedure whose address is stored beginning at memory location 40H (10H × 4) in the real mode.
- In the protected mode, the interrupt descriptor is located by multiplying the type number by 8 instead of 4 because each descriptor is 8 bytes long.

INT



- Only exception to this is INT 3, a 1-byte special software interrupt used for breakpoints.
- The INT instruction performs as a far CALL except that it not only pushes CS and IP onto the stack, but it also pushes the flags onto the stack.
- The INT instruction performs the operation of a PUSHF, followed by a far CALL instruction.

INT

Whenever a software interrupt instruction executes, it

(1) pushes the flags onto the stack

(2) clears the T and I flag bits

(3) pushes CS onto the stack

(4) fetches the new value for CS from the interrupt vector

(5) pushes IP/EIP onto the stack

(6) fetches the new value for IP/EIP from the vector

(7) jumps to the new location addressed by CS and IP/EIP

IRET/IRETD

- The interrupt return instruction (IRET) is used only with software or hardware interrupt service procedures.
- Unlike a simple return instruction (RET), the IRET instruction will

(1) pop stack data back into the IP

(2) pop stack data back into CS

(3) pop stack data back into the flag register

- The IRET instruction accomplishes the same tasks as the POPF, followed by a far RET instruction

INT 3

- An INT 3 instruction is a special software interrupt designed to function as a breakpoint
- INT 3 is a 1-byte instruction, while the others are 2-byte instructions
- It is common to insert an INT 3 instruction in software to interrupt or break the flow of the software, called a breakpoint
- Why we need breakpoints?

INT 3

How INT3 works?

1. Flag register value is pushed on to the stack.
2. CS value of the return address and IP value of the return address are pushed on to the stack.
3. IP is loaded from the contents of the word location 3×4 = 0000CH
4. CS is loaded from the contents of the next word location.
5. Interrupt Flag and Trap Flag are reset to 0

INTO

- Interrupt on overflow (INTO) is a conditional software interrupt that tests the overflow flag (O).
- If O = 0, the INTO instruction performs no operation; if O = 1 and an INTO instruction executes, an interrupt occurs via vector type number 4.
- Either the JO instruction or INTO instruction detects the overflow condition

INTO

It is active only when the overflow flag is set to 1 and branches to the interrupt handler whose interrupt type number is 4 using following steps.

- Flag register values are pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of word location 4×4 = 00010H
- CS is loaded from the contents of the next word location.
- Interrupt flag and Trap flag are reset to 0

INT 21H

- It invokes a software interrupt to call the DOS function pointed to by the value in the AH (accumulator high) register.
- **Set up AH register**: Before calling INT 21H, you load the AH register with the number corresponding to the specific DOS function you want to call.
- Reading or writing data to a file, the DX register is commonly used to specify the memory buffer or file handle.

E.g.

```
MOV DL, 'A'    ; Set DL register with ASCII value of 'A'
MOV AH, 02H    ; Set AH register with function code 02H for write character
INT 21H        ; Call DOS interrupt 21H
```

Common AH values

AH = 01H: Read character from standard input (keyboard).
AH = 02H: Write character to standard output (console).
AH = 06H: Direct console output (write string to standard output).
AH = 07H: Direct console input (read string from standard input).
AH = 09H: Write string to standard output with '$' terminator.
AH = 0AH: Read string from standard input with buffer length.
AH = 0CH: Check keyboard status without waiting.
AH = 0EH: Write character to standard output, with advanced control.
AH = 19H: Get current default drive.
AH = 25H: Set interrupt vector.
AH = 26H: Create new PSP (Program Segment Prefix).
AH = 3DH: Open existing file or create new file.
AH = 3EH: Close file.
AH = 3FH: Read from file or device.
AH = 40H: Write to file or device.
AH = 42H: Move file pointer.
AH = 4BH: Execute program.
AH = 4CH: Terminate program with return code.

Example: Print a numerical value (integer) on the screen


```
.MODEL SMALL
.STACK 100H

.DATA
    num DW 1234          ; The numerical value to be printed

.CODE
MAIN PROC
    MOV AX, @DATA        ; Initialize DS register
    MOV DS, AX

    MOV AX, num          ; Move the numerical value into AX register
    CALL PRINT_NUM       ; Call the procedure to print the number

    MOV AH, 4CH          ; DOS function to terminate the program
    INT 21H

MAIN ENDP
```

Example: Print a numerical value (integer) on the screen


```
; Procedure to print a numerical value
PRINT_NUM PROC
    ; Convert the number to a string representation
    MOV CX, 10          ; Initialize divisor for division by 10
    MOV BX, 10          ; Initialize digit place counter

    MOV SI, OFFSET buffer  ; Set SI to point to buffer for storing the string
    ADD SI, 6           ; Move SI to the end of the buffer

    MOV WORD PTR [SI], '$'  ; Null-terminate the string

    MOV DX, 0           ; Clear DX register
```

Example: Print a numerical value (integer) on the screen


```
; Procedure to print a numerical value
PRINT_NUM PROC
    ; Convert the number to a string representation
    MOV CX, 10          ; Initialize divisor for division by 10
    MOV BX, 10          ; Initialize digit place counter

    MOV SI, OFFSET buffer  ; Set SI to point to buffer for storing the string
    ADD SI, 6           ; Move SI to the end of the buffer

    MOV WORD PTR [SI], '$'  ; Null-terminate the string

    MOV DX, 0           ; Clear DX register
    ; Loop to extract digits from the number
```

Example: Print a numerical value (integer) on the screen


```
    ; Loop to extract digits from the number
    EXTRACT_LOOP:
        XOR DX, DX      ; Clear DX register
        DIV CX          ; Divide AX by 10, quotient in AX, remainder in DX
        ADD DL, '0'     ; Convert the remainder to ASCII character
        DEC SI          ; Move SI to the next position in the buffer
        MOV [SI], DL    ; Store the ASCII character in the buffer
       INC BX           ; Increment digit place counter
        TEST AX, AX     ; Check if quotient is zero
        JNZ EXTRACT_LOOP  ; If not, continue the loop

    ; Display the string on the screen
    MOV AH, 09H         ; DOS function to display string
    MOV DX, SI          ; Load DX with the address of the string
    INT 21H
    RET
PRINT_NUM ENDP
END MAIN
```

An Interrupt Service Procedure

- Suppose that, in a particular system, a procedure is required to add the contents of DI, SI, BP, and BX and then save the sum in AX.
- Because this is a common task in this system, it may be worthwhile to develop the task as a software interrupt.

An Interrupt Service Procedure

Example

```
0000                    INTS    PROC   FAR USES AX
0000 03 C3                      ADD    AX,BX
0002 03 05                      ADD    AX,BP
0004 03 C7                      ADD    AX,DI
0006 03 C6                      ADD    AX,SI
0008 CF                         IRET
0009                    INTS    ENDP
```

Opening the File

```
Open a New File/Rewrite a File: INT 21h, function 3Ch
Input:  AH = 3Ch
        DS:DX = address of file name, which is an ASCII string
                (a string ending with a 0 byte)
        CL = attribute
Output: if successful, AX = file handle
        Error if CF = 1, error code in AX (3, 4, or 5)
```

Example: Write a program to open a new read-only file called FILE1.

```
.model small
.stack 100h
.data
FNAME           DB      'FILE1', 0
HANDLE  DB      ?
.code
.startup
        MOV AH, 3Ch     ; open file function
        LEA DX, FNAME   ; DX has filename address
        MOV CL, 1       ; read-only attribute
        INT 21H         ; open file
        MOV HANDLE, AX  ; save handle or error code
        JC OPEN_ERROR   ; jump if error
        ...
.exit
END
```

Opening the existing file

innovate achieve lead

```
Open an Existing File: INT 21h, function 3Dh
Input:  AH = 3Dh
        DS:DX = address of file name, which is an ASCII string
                  (a string ending with a 0 byte)
        AL = access code:       0 means open for reading
                                1 means open for writing
                                2 means open for both
Output: if successful, AX = file handle
        Error if CF = 1, error code in AX (2, 4, 5 or 12)
```

Read from the existing file


```
Read from a File: INT 21h, function 3Fh
Input:  AH = 3Fh
        BX = file handle
        CX = number of bytes to read
        DS:DX = memory buffer address
Output: AX = number of bytes actually read
        if AX=0 or AX < CX, end of file encountered
        Error if CF = 1, error code in AX (5, 6)
```

Example: Write some code to read a 512-byte from a file. Assume file handle is stored in variable HANDLE, and BUFFER is a 512 byte buffer.

```
.data
        HANDLE DW       ?
        BUFFER DB 512 DUP(0)
        ...
.code
        ...
        MOV AX, 3Fh     ; read file function
        MOV BX, HANDLE  ; get handle
        MOV CX, 512     ; read 512 bytes
        INT 21H         ; read file, AX = bytes read
        JC READ_ERROR   ; jump if error
        ...
```

Writing to a File


```
Write to a File: INT 21h, function 40h
Input:  AH = 40h
        BX = file handle
        CX = number of bytes to write
        DS:DX = data address
Output: AX = number of bytes written
        if AX=0 or AX < CX, error (full disk)
        Error if CF = 1, error code in AX (5, 6)
```

Interrupt Control

- The set interrupt flag instruction (STI) places a 1 into 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.
- In a software interrupt service procedure, hardware interrupts are enabled as one of the first steps.
- This is accomplished by the STI instruction.
- The reason interrupts are enabled early in an interrupt service procedure is all of the I/O devices in the personal computer are interrupt-processed.
- If the interrupts are disabled too long, severe system problems result.

Controlling the Carry Flag Bit

- The carry flag (C) propagates the carry or borrow in multiple-word/doubleword addition and subtraction.
- It also can indicate errors in assembly language procedures.
- Three instructions control the contents of the carry flag:
  - STC (set carry)
  - CLC (clear carry)
  - CMC (complement carry)

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You