Assembly Language for Intel-Based Computers, 4th Edition

Kip R. Irvine

Chapter 13: 16-Bit MS-DOS Programming

Interrupts

Slide show prepared by Kip R. Irvine, Revision date: 08/04/02

Modified by Dr. Nikolay Metodiev Sirakov

February 21,2007

- Chapter corrections (Web) Assembly language sources (Web)

Chapter Overview

- MS-DOS and IBM-PC
- MS-DOS Function Calls (INT 21h)
- Standard MS-DOS File I/O Services

MS-DOS and the IBM-PC

- Real-Address Mode
- MS-DOS Memory Organization
- MS-DOS Memory Map
- Redirecting Input-Output
- Software Interrupts
- 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
  - Derived by Microsoft from Gary Kildall's Digital Research CP/M
  - Later renamed to MS-DOS, owned by Microsoft

INT Instruction

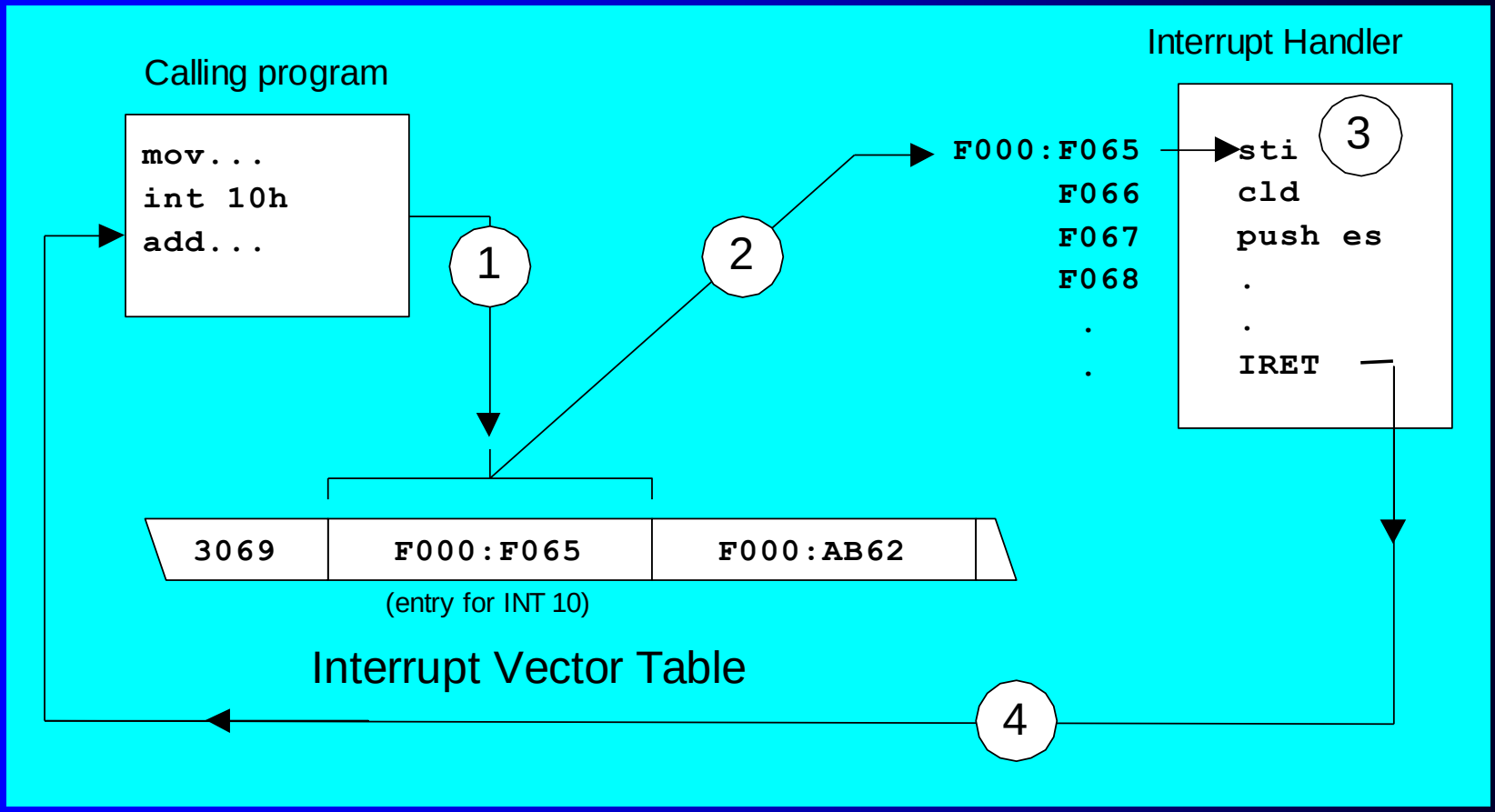
- The INT instruction executes a software interrupt.
- The code that handles the interrupt is called an interrupt handler.
- Syntax:

```
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

MS-DOS Function Calls (INT 21h)

- ASCII Control Characters
- Selected Output Functions
- Selected Input Functions
- Example: String Encryption
- Date/Time Functions

INT 4Ch: Terminate Process

- Ends the current process (program), returns an optional 8-bit return code to the calling process.
- A return code of 0 usually indicates successful completion.

```
mov ah,4Ch              ; terminate process
mov al,0                ; return code
int 21h

; Same as:

.EXIT 0
```

Selected Output Functions

- ASCII control characters
- 02h, 06h - Write character to standard output
- 05h - Write character to default printer
- 09h - Write string to standard output
- 40h - Write string to file or device

ASCII Control Characters

Many INT 21h functions act upon the following control characters:

- 08h - Backspace (moves one column to the left)
- 09h - Horizontal tab (skips forward n columns)
- 0Ah - Line feed (moves to next output line)
- 0Ch - Form feed (moves to next printer page)
- 0Dh - Carriage return (moves to leftmost output column)
- 1Bh - Escape character

INT 21h Functions 02h and 06h:
Write Character to Standard Output

Write the letter 'A' to standard output:

```
mov ah,02h
mov dl,'A'
int 21h
```

or: mov ah,2

Write a backspace to standard output:

```
mov ah,06h
mov dl,08h
int 21h
```

INT 21h Function 05h:
Write Character to Default Printer

Write the letter 'A':

```
mov ah,05h
mov dl,65
int 21h
```

Write a horizontal tab:

```
mov ah,05h
mov dl,09h
int 21h
```

INT 21h Function 09h:
Write String to Standard Output

- The string must be terminated by a '$' character.
- DS must point to the string's segment, and DX must contain the string's offset:

```
.data
string BYTE "This is a string$"

.code
mov  ah,9
mov  dx,OFFSET string
int  21h
```

INT 21h Function 40h:
Write String to File or Device

Input: BX = file or device handle (console = 1), CX = number of bytes to write, DS:DX = address of array

```
.data
message "Writing a string to the console"
bytesWritten WORD ?

.code
    mov ah,40h
    mov bx,1
    mov cx,LENGTHOF message
    mov dx,OFFSET message
    int 21h
    mov bytesWritten,ax
```

Selected Input Functions

- 01h, 06h - Read character from standard input
- 0Ah - Read array of buffered characters from standard input
- 0Bh - Get status of the standard input buffer
- 3Fh - Read from file or device

INT 21h Function 01h:
Read single character from standard input

- Echoes the input character
- Waits for input if the buffer is empty
- Checks for Ctrl-Break (^C)
- Acts on control codes such as horizontal Tab

```
.data
char BYTE ?
.code
mov ah,01h
int 21h
mov char,al
```

INT 21h Function 06h:
Read character from standard input without waiting

- Does not echo the input character
- Does not wait for input (use the Zero flag to check for an input character)
- Example: repeats loop until a character is pressed.

```
.data
char BYTE ?
.code
L1: mov  ah,06h          ; keyboard input
    mov  dl,0FFh         ; don't wait for input
    int  21h
    jz   L1              ; no character? repeat loop
    mov  char,al         ; character pressed: save it
    call DumpRegs        ; display registers
```

INT 21h Function 0Ah:
Read buffered array from standard input (1 of 2)

- Requires a predefined structure to be set up that describes the maximum input size and holds the input characters.
- Example:

```
count = 80

KEYBOARD STRUCT
   maxInput BYTE count              ; max chars to input
   inputCount BYTE ?                ; actual input count
   buffer BYTE count DUP(?)         ; holds input chars
KEYBOARD ENDS
```

INT 21h Function 0Ah (2 of 2)

Executing the interrupt:

```
.data
kybdData KEYBOARD <>

.code
    mov ah,0Ah
    mov dx,OFFSET kybdData
    int 21h
```

INT 21h Function 0Bh:
Get status of standard input buffer

- Can be interrupted by Ctrl-Break (^C)
- Example: loop until a key is pressed. Save the key in a variable:

```
L1: mov ah,0Bh        ; get buffer status
    int 21h
    cmp al,0          ; buffer empty?
    je  L1            ; yes: loop again
    mov ah,1          ; no: input the key
    int 21h
    mov char,al       ; and save it
```

INT 21h Function 3Fh:
Read from file or device

- Reads a block of bytes.
- Can be interrupted by Ctrl-Break (^C)
- Example: Read string from keyboard:

```
.data
inputBuffer BYTE 127 dup(0)
bytesRead WORD ?
.code
mov  ah,3Fh
mov  bx,0                       ; keyboard handle
mov  cx,127                     ; max bytes to read
mov  dx,OFFSET inputBuffer      ; target location
int  21h
mov  bytesRead,ax               ; save character count
```

Date/Time Functions

- 2Ah - Get system date
- 2Bh - Set system date *
- 2Ch - Get system time
- 2Dh - Set system time *

* may be restricted by your user profile if running a console window under Windows NT, 2000, and XP.

INT 21h Function 2Ah:
Get system date

- Returns year in CX, month in DH, day in DL, and day of week in AL

```
mov   ah,2Ah
int   21h
mov   year,cx
mov   month,dh
mov   day,dl
mov   dayOfWeek,al
```

INT 21h Function 2Bh:
Set system date

- Sets the system date. AL = 0 if the function was not successful in modifying the date.

```
mov   ah,2Bh
mov   cx,year
mov   dh,month
mov   dl,day
int   21h
cmp   al,0
jne   failed
```

INT 21h Function 2Ch:
Get system time

- Returns hours (0-23) in CH, minutes (0-59) in CL, and seconds (0-59) in DH, and hundredths (0-99) in DL.

```
mov   ah,2Ch
int   21h
mov   hours,ch
mov   minutes,cl
mov   seconds,dh
```

INT 21h Function 2Dh:
Set system time

- Sets the system date. AL = 0 if the function was not successful in modifying the time.

```
mov   ah,2Dh
mov   ch,hours
mov   cl,minutes
mov   dh,seconds
int   21h
cmp   al,0
jne   failed
```

Example: Displaying the Date and Time

- Displays the system date and time, using INT 21h Functions 2Ah and 2Ch.
- Demonstrates simple date formatting
- View the source code
- Sample output:

```
Date: 12-8-2001,   Time: 23:01:23
```

ToDo: write a procedure named ShowDate that displays any date in mm-dd-yyyy format.

The End