Microprocessors (0630371)
Fall 2010/2011 – Lecture Notes # 21

Interrupts
MS-DOS Function Calls (INT 21h)

Objectives

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

MS-DOS provides a lot of functions for displaying and reading the text on the console (200 functions). The general syntax for calling the functions is

```
mov ah, function number
     ; input parameters
int 21h
     ; return values
```

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
```

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**

Selected Output Functions

- **02h, 06h** - Write a single character to standard output
- **05h** - Write a single character to default printer
- **09h** - Write string (terminated by a $ character)to standard output.
- **40h** - Write an array of bytes (block of data) to a file or device

1. **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
```

Write a backspace to standard output:

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

The difference between Functions **02h** and **06h** is that the 06h function returns the **ASCII code** of the character in **AL** , if **ZF =0.**

2. **INT 21h Function 05h: Write Character to Default Printer**

Write the letter 'A':

```
mov ah,05h      ; select printer output
mov dl,65       ; character to be printed
int 21h         ; call MS-DOS
```

Write a horizontal tab:

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

3. **INT 21h Function 09h: Write a $-terminated 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
```

Use the **SEG** operator to set DS to the segment containing the data, The following statements do this:

```
.data
inBuffer BYTE 80 DUP(0)
.code
mov ax, SEG inBuffer
mov ds, ax
mov dx, OFFSET inBuffer
```

4. **INT 21h Function 40h: Write a block of data(array of byts) to a File or Device**

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

```
.data
message "Writing a string to the console"
bytesWritten WORD ?
.code
mov ax,@data
mov ds,ax
mov ah,40h
mov bx,1
mov cx,LENGTHOF message
mov dx,OFFSET message
int 21h
mov bytesWritten,ax
```

Programming Examples

```
                        ;Example 1
TITLE Hello World Program          (Hello.asm)
.MODEL small
.STACK 100h
.386
.data
message BYTE "Hello, world!",0dh,0ah
.code
main PROC
     mov  ax,@data                  ; initialize DS
     mov  ds,ax
     mov  ah,40h                    ; write to file/device
     mov  bx,1                      ; output handle
     mov  cx,SIZEOF message         ; number of bytes
     mov  dx,OFFSET message         ; addr of buffer
     int  21h
     .EXIT
main ENDP
END main
```

```
                        ;Example 2
TITLE Hello World Program          (Hello2.asm)
.MODEL small
.STACK 100h
.386
.data
message BYTE "Hello, world!",0dh,0ah
.code
main PROC
     .STARTUP
     mov  ah,40h                ; write to file/device
     mov  bx,1                  ; output handle
     mov  cx,SIZEOF message     ; number of bytes
     mov  dx,OFFSET message     ; addr of buffer
     int  21h
   .EXIT
main ENDP
END
```

Selected Input Functions

- **01h, 06h** - Read a single 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

1. **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
```

2. **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)
- If **ZF =0**, AL contains the character's ASCII code.
- 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
```

3. **INT 21h Function 0Ah: Read buffered array from standard input**

- 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

- Executing the interrupt:

```
.data
kybdData KEYBOARD <>
.code
     mov ah,0Ah
     mov dx,OFFSET kybdData
     int 21h
```

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

- Can be interrupted by Ctrl-Break (^C)
- If the character is waiting, AL =0FFh; otherwise, AL=0.
- 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
```

Example: String Encryption

- Reads from standard input, encrypts each byte, writes to standard output.

```
                    ;Example 3
TITLE Encryption Program                (Encrypt.asm)
; This program uses MS-DOS function calls to
; read and encrypt a file. Run it from the
; command prompt, using redirection:
;    Encrypt < infile.txt > outfile.txt
; Function 6 is also used for output, to avoid
; filtering ASCII control characters.
INCLUDE Irvine16.inc
XORVAL = 239              ; any value between 0-255
.code
main PROC
     mov   ax,@data
     mov   ds,ax
L1:
     mov  ah,6            ; direct console input
     mov  dl,0FFh         ; don't wait for character
     int  21h             ; AL = character
     jz   L2              ; quit if ZF = 1 (EOF)
     xor  al,XORVAL
     mov  ah,6            ; write to output
     mov  dl,al
     int  21h
     jmp  L1              ; repeat the loop
L2:  exit
main ENDP
END  main
```

5. **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
```

```
                                    ;Example 4
TITLE Buffered Keyboard Input          (Keybd.asm)
; Test function 3Fh, read from file or device with the keyboard. Flush ;the
buffer.
INCLUDE Irvine16.inc
.data
firstName BYTE 15 DUP(?),0
lastName  BYTE 30 DUP(?),0
.code
main PROC
   mov  ax,@data
   mov  ds,ax
; Input the first name:
     mov  ah,3Fh
     mov  bx,0                          ; keyboard handle
     mov  cx,LENGTHOF firstName
     mov  dx,OFFSET firstName
     int  21h
;  Disable the following line to see what happens when the buffer is not
;flushed:
     ;call FlushBuffer
; Input the last name:
     mov  ah,3Fh
     mov  bx,0                          ; keyboard handle
     mov  cx,LENGTHOF lastName
     mov  dx,OFFSET lastName
     int  21h
; Display both names:
     mov  dx,OFFSET firstName
     call WriteString
     mov  dx,OFFSET lastName
     call WriteString
quit:
     call Crlf
    exit
main ENDP
FlushBuffer PROC
; Flush the standard input buffer.; Receives: nothing. Returns: nothing
.data
oneByte BYTE ?
.code
     pusha
L1:
    mov ah,3Fh                      ; read file/device
     mov bx,0                       ; keyboard handle
    mov cx,1                        ; one byte
    mov dx,OFFSET oneByte  ; save it here
    int 21h                         ; call MS-DOS
    cmp oneByte,0Ah                 ; end of line yet?
    jne L1                  ; no: read another
     popa
     ret
FlushBuffer ENDP
END main
```

Date/Time Functions

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

1. **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
```

2. **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
```

3. **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
```

4. **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

```
                        ;Example 5
TITLE Display the Date and Time     (DateTime.asm)
; This Real-mode program displays the date and time.
Include Irvine16.inc
Write PROTO char:BYTE
.data
```

```
str1 BYTE "Date: ",0
str2 BYTE ",  Time: ",0
.code
main PROC
    mov  ax,@data
    mov  ds,ax
; Display the date:
    mov   dx,OFFSET str1
    call  WriteString
    mov   ah,2Ah        ; get system date
    int   21h
    movzx eax,dh        ; month
    call  WriteDec
    INVOKE Write,'-'
    movzx eax,dl        ; day
    call  WriteDec
    INVOKE Write,'-'
    movzx eax,cx        ; year
    call  WriteDec
; Display the time:
    mov   dx,OFFSET str2
    call  WriteString
    mov   ah,2Ch        ; get system time
    int   21h
    movzx eax,ch        ; hours
    call  WritePaddedDec
    INVOKE Write,':'
    movzx eax,cl        ; minutes
    call  WritePaddedDec
    INVOKE Write,':'
    movzx eax,dh        ; seconds
    call  WritePaddedDec
    call  Crlf
    exit
main ENDP
;-----------------------------------------------
Write PROC char:BYTE
; Display a single character.
;-----------------------------------------------
    push eax
    push edx
    mov  ah,2
    mov  dl,char
    int  21h
    pop  edx
    pop  eax
    ret
Write ENDP
;-----------------------------------------------
```

```
WritePaddedDec PROC
; Display unsigned integer in EAX, padding
; to two digit positions with a leading zero.
;-----------------------------------------------
    .IF eax < 10
       push eax
       push edx
       mov  ah,2
       mov  dl,'0'
       int  21h
       pop  edx
       pop  eax
    .ENDIF
    call WriteDec
    ret
WritePaddedDec ENDP
END main
```