Lecture 13: I/O

- Interrupts
- MS-DOS Function Calls
  - Input,Output, File I/O
- Video
- Keyboard

I/O

- Getting data into your program:
  - define it in the data area
  - use immediate operands
- Very limiting…
- Most programs require input from an external source: keyboard, disk, mouse, or modem.
- Most programs need to provide output in a useable fashion: screen, printer, or disk.

How?

- Using the INT (Interrupt) instruction:
  - INT 21h -> DOS function calls
  - INT 10h -> BIOS level video control
  - INT 16h -> BIOS level keyboard input

(BIOS? Basic Input Output System – special programs built into the computer that handle basic I/O)

Interrupts

- What's an interrupt?
  - processing of one program is interrupted to respond to an event or take some action
- Two types we'll talk about:
  - Hardware
  - Software

Hardware Interrupts

- A hardware interrupt is a signal generated by part of the system hardware that needs immediate attention from the CPU.
- An example of this is hitting a key on the keyboard. The CPU will suspend the current program and execute a BIOS level routine.
- This is needed because if the keyboard character is not saved by the CPU then it will be lost.
- Sometimes programs have to disable interrupts. This is done using the CLI (clear interrupt flag) and STI (set interrupt flag) instructions.

Software Interrupts

- Not really interrupts but have some similarities.
- INT instruction requests services from the operating system (in other words: INT calls an OS subroutine)
- The value in AH tells which subroutine to call.
- Also, other registers may need to hold data needed by the subroutine.

How does it work?

- The CPU processes an interrupt instruction using the *interrupt vector table*: a table of addresses in the lowest 1,024 bytes of memory.
- Each entry in the table points to an operating system subroutine.
- Steps taken:
  1. Use the number after the INT (such as 21h or 10h) to find the entry in the interrupt vector table.
  2. Jump to the address stored at that location in the interrupt vector table.
  3. Execute the DOS subroutine at that location.
  4. Return to the calling program (using the IRET instruction)

Common Software 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 – DOS services

MS-DOS Function Calls

- INT 21h
- 90 or so different functions!
- We'll look at:
  - Output
  - Input
  - File I/O

But first…

- Remember ASCII?
- When you do I/O using INT 21h you are working with ASCII characters.
- For example, 'A' = 41h
- This ALSO holds true if you are reading in or displaying numbers!
- If you want to display 123, you have to tell DOS to display 31 (the ASCII code for '1'), then 32, then 33
- If you are reading in a number, you will need to convert it from a string of ASCII characters (such as "123") into a number (stored internally as binary).

Output

- We've seen 09h (String Output)
  - example, p. 148
- 02h: Character Output
  - sends a character to standard output (your screen) and advances the cursor.
  - Input: AH = 2, DL = the character you want to print
  - Output: AL is modified

Character Output Example

- From write procedure in Homework 3:

```
    ;Display the buffer, using CX as a counter.
A2:     mov     ah, 2     ;function: display
                          ;character
        mov dl, [di]      ;get digit from buffer
        int   21h         ;call DOS
        inc   di          ;point to next digit
        loop A2
```

Input

- A bewildering array of DOS input functions:
  - 01h – filtered input with echo
  - 06h – direct input without waiting
  - 07h – direct input, no control-break
  - 08h – direct input with control-break
  - 0ah – buffered input
  - 0bh – get input status
  - 0ch – clear input buffer, invoke input function
  - 3fh – read from file or device

A few terms

- Keyboard typeahead buffer – a 15 character circular buffer that stores keystrokes as you type (lets you type faster than DOS can respond without losing data)
- Input characteristics:
  - waits for keystroke – does the function wait for you to type or assume the character is in the buffer?
  - echos – does it display the character it reads?
  - cntrl-break – can you terminate it using control break?
  - filtered input – does it filter out control characters (such as enter, tab, backspace)

Some Input Functions

- 01h – filtered input with echo
  - waits for a single character to be entered (or, if one is in the input buffer already just grabs it)
  - stores it in AL
  - Input: ah = 1
  - Output: al = the character read

  (filtered? filters out control characters)

  (echo? it displays the character you as you type it. If you weren't using echo you would not be able to see what you typed!)

  example, p. 150

Buffered Input

- 0Ah reads a character string of up to 255 characters from standard input (your DOS window) and stores it in a buffer.
  - Backspace key can be used to erase characters and back up the cursor
  - Enter key terminates input
  - Non-ASCII keys are filtered out
  - Input: ah = 0ah, DX contains offset of record containing the keyboard parameters.

Keyboard Parameters?

- You (the programmer) need to define the data area where you tell the DOS function how many characters you want to input (maximum), and where it should put them:

```
.data
maxlen   db      20   ; max chars to input
actuallen db     ?    ;DOS will put the
   number read here
inbuf    db      20 dup (‘ ‘) ; where DOS
   will put the data read in
```

input buffer area (last char will be enter - 0Dh)

chars in buffer (not counting enter)

max characters your program will allow (<= 255) – includes enter!

Example

- example on p. 152, Irvine

File Processing

- DOS sees no distinction between disk files and devices (keyboard, display, etc.)
- A handle is a 16-bit number DOS uses to identify an open file or device.
- Standard handles that are pre-defined (that you don’t have to open):
  - 0 – keyboard
  - 1 – console
  - 4 – printer
- For all functions, if an error occurs then the carry flag is set and an error code is returned in AX.

Basic File Functions

- table 1, p. 439 in Irvine

Using Files

- 3Ch – creating a file.
  - Creates a new file or truncates an existing file to 0 bytes.
  - Automatically opens file for reading and writing
  - Input:
    - AH = 3Ch
    - DS:DX points to an ASCIIZ string with the name of the file (ASCIIZ? null terminated).
    - CX contains an attribute value: 00 normal file, 01 read only, 02 hidden, 04 system.
  - Output:
    - File Handle returned in AX

File Create Example

```
.data
newfile  db    "NEWFILE.DOC", 0
handle   dw    ?

mov dx, offset newfile      ;name offset
mov ah, 3ch                 ;create file
mov cx, 0                   ;normal file
int 21h                     ;DOS call
jc  display_error           ;jump if error
mov handle, ax              ;save handle
```

This could be dangerous!

- Why? it does not prevent you from writing over an existing file.
  - You can check if the file exists (by trying to open it) or
  - You can use function 5Bh (create new file instead)

File Error Codes

- When the carry flag is set, you have an error.
- AX will have the error code:
  - 03 – path not found. The file specifier (pointed to by DX) probably has a non-existant directory name
  - 04 – too many open files. The max number of open files defaults to 8. The first five are used by DOS (standard file handles), that leaves you only three. You can change this by editing your CONFIG.SYS file to add the files command (such as files=32)
  - 05 – access denied. The file exists, and is read only, or the file name matches a subdirectory name, or you're adding a new entry into a full root directory.

Opening a File

- 3dh – opens an existing file in input (read only), output (write only), or input-output mode.
  - Input:
    - AH = 3dh
    - AL – contains the file mode
      input = 0
      output = 1
      input-output = 2
    - DX contains the offset of the filename
  - Output: file handle in AX

Example

- p. 450 in Irvine

- Error codes returned in AX:
  1 invalid function (trying to share file)
  2 file not found
  3 path not found
  4 too many files
  5 access denied

Closing a File

- 3Eh – closes a file.
  - Flushes the DOS internal file buffer by writing any remaining data to disk
  - Makes the file handle available to other programs (important since number of files is limited!)
  - Input:
    - AH = 3eh
    - BX – contains the file handle
  - Output:
    - Only if there's an error. The only error is 6 – invalid handle (file handle does not refer to an open file)

Example

- p. 451, Irvine

Read from File or Device

- 3Fh – read from file or device
  - reads from a file or another device (such as the keyboard if your handle is zero!)
  - If you're using it for keyboard input, it will terminate when you type enter, the CR,LF will be stored and included in the count of characters.
  - Input:
    - AH = 3F
    - BX – file handle (0 for keyboard)
    - CX – number of bytes to read
    - DX – pointer to the buffer area
  - Output:
    - AX – number of bytes read
  - The number of bytes read is useful because you can use it to check for end of file!

Example

- p. 452, Irvine

Write to a File or Device

- 40h – write to a file or device
  - writes to a file or other device (such as the terminal if the handle is 1)
  - Input:
    - AH = 40h
    - BX – file handle (1 for terminal)
    - CX – number of bytes to write
    - DX – pointer to the buffer area
  - Output:
    - AX – number of bytes written
  - If the number of bytes written (AX) is less than the number of bytes to write (CX) then the disk might be full!
  - Possible error codes are 5 – access denied and 6 – invalid handle

Example

- p. 452, Irvine

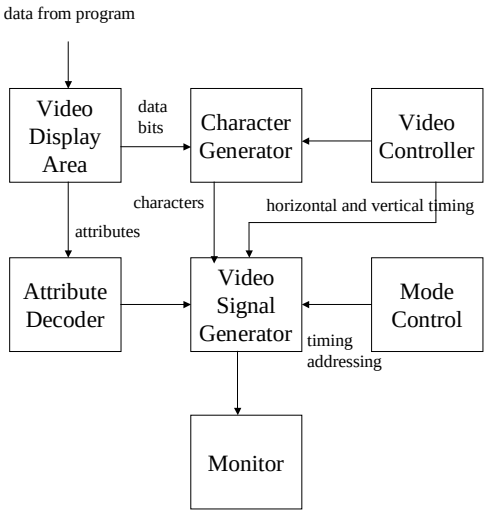
Recap

- These were a few of the INT 21h functions.
- We talked about functions for:
  - output (strings and single chars)
  - input (single char and buffered)
  - file I/O (create, open, close, read, write)
- Your text describes many more!

BIOS Operations

- The INT 21h functions can handle much of your I/O
- There are also lower-level BIOS operations (which the procedures called by INT 21h will use) that handle more complex operations.
- Some that are described in Irvine:
  - INT 10h – BIOS video operation
  - INT 16h – Keyboard operation

Video Systems



Components of a Video System

- Basic components:
  - Monitor – what displays the data. The screen consists of a group of closely spaced horizontal lines -> the raster. Each line is made of hundreds of pixels (points)
  - Video Display Area – this is where a program transfers data for display. It can be in characters (for text mode – what we've been using) or in pixels (for graphics mode). Data is stored in pages where one page is displayed at a time.
  - Video Controller – generates horizontal and vertical timing signals. It synchronizes delivery of data with these signals. It also handles size and location of the cursor and selecting the page to be displayed.

More components

- ASCII generator – converts ASCII codes from the video display area into dot patterns that make up the characters.
- Attribute decoder – translates the attribute byte from the video display area into signals to give the characters characteristics (background and foreground color, intensity, blinking…)
- Mode control – the video mode determines things like text vs. graphics, color or monochrome, etc.

Video Attributes

- If you're using the color text mode (ASCII, not graphics) you can display text in colors with blinking and reverse video.
- Video attribute:
  - picture and example from p. 159.

INT 10h (BIOS Video Operation) Functions

- table from p. 163, Irvine

Keyboard Operations

- We've already mentioned the keyboard buffer (when talking about buffered I/O).
- When you press a key, the keyboards processor generates the *scan code* for the key and requests BIOS INT 09H.
  - scan code? yet another character encoding. This is NOT the same as ASCII!
- INT 09h gets the scan code, converts it to ASCII, and delivers it to the keyboard buffer ara.

INT 16h (Keyboard Operation) Functions

- table from p. 156, Irvine.