Chapter 8: I/O Organization and Interrupts

I. CUSP has 5 I/O Devices
   A. Keyboard
   B. Display screen, CRT
   C. Printer
   D. Tape drive
   E. Timer

II. I/O Ports
   A. CUSP has 4096 8-bit ports. Many are unused-this allows for future expansion.
   B. Used to transfer data to and from device controllers.
      1. Each device has a Control Register that is used to set device specific characteristics.
      2. All devices but the CRT have a Status Register that is used to determine the state of the device.
      3. Status Register and Control Register share the same I/O port address. Why is this typical?
   C. Four instructions to read and write to I/O Ports.
      1. All addressing modes allowed, except immediate mode.
      2. OUTB = output byte. Writes lower 8 bits of the ACCUM to the selected port.
      3. INB = input byte. Reads byte from I/O port and stores it in the lower 8 bits of the ACCUM, and clears the upper 16 bits of the ACCUM.
      4. OUTW = output word. Writes ACCUM to 3 consecutive I/O ports, with the most significant 8 bits stored in the lowest address, i.e. big endian fashion. The operand specifies the address of the lowest port.
      5. INW = input word. Reads into the ACCUM the values from 3 consecutive I/O ports with the bits stored in the lowest addressed I/O port being stored in the most significant 8 bits of the ACCUM. The operand specifies the address of the lowest port.

III. Keyboard
   A. Shift, Ctrl, and Alt keys just modify the ASCII codes of non-control keys pressed at the same time.
   B. Some special keys, notably cursor movement keys, place two characters in the buffer, the first of which is always ASCII Escape code $1B. The second character is the two-character sequence varies depending on the key pressed. In UNIX this produces the all too familiar A's, B's, C's, and D's when pressing the cursor movement keys in vi while in insert mode.
   C. CUSP Keyboard has three registers associated with it.
      1. Keyboard Data Register (I/O port $001). Contains the first character of a hidden 10 character FIFO (first-in, first-out) buffer, or queue.
      2. Keyboard Status Register (I/O port $000).
         a) Bit 7: 1 = ready (data available). 0 = not ready (no data available). Since INB sets LT when this is set it is an easy way to determine if there is data available.
         b) Bit 6: 1 = interrupt enabled, 0 = interrupt disabled.
         c) Bits 0-5: unused (always 0).
      3. Keyboard Control Register (I/O port $000)
         a) Bit 7: 1 = enable interrupt. 0 = disable interrupt.
         b) Bit 6: 1 = flush buffer. 0 = no operation.
         c) Bits 0-5: unused (no affect).

IV. Display Screen (CRT)
   A. Two representations: pixels elements and characters.
      1. A pixels (picture element) is a small (.28 mm) phosphorescent dot on the CRT. Common dimensions are 640 W x 480 H, 800 x 600, 1024 x 768, 1280 x 1024.
      2. Characters, usually 80 W x 25 H. In CUSP 38 W x 14 H.
   B. Mapped displays
      1. Have a video buffer that contains a mirror of what is on the CRT.
         a) Buffer size determined by dimensions and maximum number of colors that can be displayed.
(1) 2 colors /pixel requires only one bit for each pixel, 16 colors / pixel requires a nibble (4 bits) for each pixel, 256 colors / pixel requires a byte for each pixel, 65536 / pixel requires 2 bytes per pixel.

(2) For character representation there is one byte per character, and a separate nibble, byte, or 2-byte, table for the color of the character.

b) Video buffer now on graphics card using VRAM.

c) CUSP video buffer $100 - $313. Since it uses character representation has set colors it has no color table. May be read or written to.

C. Unmapped displays
1. Character codes are sent one at a time to a CRT Data Register (I/O port $317).
   a) Non-control characters are displayed at the current cursor location, and then the cursor is automatically moved one position to the right, with CR and LF if necessary.
   b) Control characters, BS, CR, LF, FF, BEL, and ESC, cause cursor to move appropriate to their definitions.

2. Cursor location can be specified and ascertained with Cursor X register (I/O port $314) and the Cursor Y register (I/O port $315). Upper Left is (1,1).

D. CUSP CRT Control Register (I/O port $316) is used to tell CRT to execute specific operations.
1. $01 = Clear display and move cursor home.
2. $02 = Scroll display down one line.
3. $03 = Scroll display up one line
4. $04 = Cursor home.
5. $05 = Cursor to start of next line
6. $06 = Ring bell.

V. Printer
A. Similar to non-mapped displays, i.e., serial and automatically move one space to right (usually with wrapping).

B. Character codes
1. Control codes are printer commands that start with ESC ($1B) and then followed with one or more characters as required.
   a) On an Epson printer $1B $50 will cause the printer to change to Pica pitch.
   b) On another printer $1B $54 $05 $08 $13 $00 will set the tabs to 5, 8, and 19.

2. "Non-printable" ASCII characters affect printer as expected, e.g. LF, FF, BS, TAB, and CR.
   a) Note that BS causes the print head to backup one space but cannot erase the previously printed character. In CUSP though, BS will erase the character.

C. CUSP has three printer registers.
1. Printer Data Register (I/O port $011) used to write characters to printer.
2. Printer Status Register (I/O port $010)
   a) Bit 7: 1 = ready (to receive a character).
   b) Bit 6: 1 = interrupts enabled
   c) Bit 5: 1 = printer on-line. If zero, then Bit 7 will also be zero.
   d) Bit 4: 1 = interrupt pending.
   e) Bits 0-3 : unused (always zero).
3. Printer Control Register (I/O port $010)
   a) Bit 7: 1 = enable interrupts, 0 = disable interrupts.
   b) Bit 6: 1 = clear interrupt request, 0 = no operation.
   c) Bits 0-5: unused (no effect).

VI. Tape Drive
A. Data is organized sequentially as a series of records.
1. Each record ends with an end-of-record marker. In CUSP this a the CR, LF combination used by DOS to indicate EOLN. Therefore, in CUSP each record is on its own line. End of record marker is not transferred.
2. The last record on the tape is followed with an end-of-tape marker. In CUSP this is the end of the file.
B. Direct Memory Access (DMA) allows the device to access memory directly without using the CPU.
   1. Device requests must interleave with CPU memory requests. Control bus determines when device can use the bus.
   2. Devices are connected directly to system bus.
   3. Much faster than Program Controlled Transfer that transfers data one character at a time between I/O port and memory.

C. CUSP has four Tape Drive registers
   1. Transfer Length Register (I/O port $021) holds the maximum number of characters to be read from the tape, or the number of characters to write to the tape.
   2. Transfer Address Register (I/O ports $022 - $024) . The lower 12 bits hold the address of the first memory location to be used in transferring data. The upper 12 bits are ignored. Three ports are used to allow the use of INW and OUTW.
   3. Tape Status Register (I/O port $020)
      a) Bit 7: 1 = ready (for an operation). Set after completion of operation when tape is mounted.
      b) Bit 6: 1 = interrupts enabled.
      c) Bit 5: 1 = tape mounted. Errors cause tape dismount. If dismounted then not ready.
      d) Bit 4: 1 = interrupt pending
      e) Bit 3: 1 = end-of-tape encountered on read
      f) Bits 0-2: unused (always zero.)
      g) When testing whether tape is ready, you should check both Bit 5 and Bit 7.
   4. Tape Control Register (I/O port $020)
      a) Bit 7: 1 = enable interrupts, 0 = disable interrupts.
      b) Bit 6: 1 = clear interrupt request, 0 = no operation
      c) Bits 4-5: 00 = no operation, 01 = read record, 10 = write record, 11 = rewind tape.
      d) Bits 0 -3 : unused (no effect).

VII. Timer

A. CUSP Timer has four registers
   1. Timer Control Register (I/O Port $030)
      a) Bit 7: 1= enable interrupts, 0 = disable interrupts.
      b) Bit 6: 1= Clear ready bit, 0 = no operation
      c) Bits 4-5: 00 = no operation, 01 = start timer (after loading counter), 10 stop timer, 11 = start timer (without loading counter).
      d) Bits 0-3: Unused (no affect)
   2. Timer Status Register (I/O Port $030)
      a) Bit 7: 1 = ready (count complete)
      b) Bit 6: 1 = interrupts enabled, 0 = interrupts disabled
      c) Bits 0-5: Unused (always zero).
   3. Reload Value Register (I/O ports $031 - $033),
      a) Value is copied into the Counter Register when the Counter Register value reaches zero, or when Timer Control Register is set with $10.
      b) Read/write, but seldom read.
   4. Counter Register (I/O ports $034 - $036)
      a) Decremented with each machine cycle and then value is checked. Therefore longest possible count will be when it is initialized with 0-which will be $1000000 ticks.
      b) When it reaches zero, after decrementing:
         (1) The ready bit in TSR is set.
         (2) Counter Register is loaded with the value in the Reload Value Register.
         (3) Interrupt is thrown.
      c) Read/write, but seldom written.

VIII. Interrupts

A. Why? Because polling doesn't work well.
   1. Too frequent will cause the program to slow down unnecessarily.
2. Too sparse will make the program unresponsive.

B. Interrupt Service Routine, ISR
1. Two requirements to call ISR: 1) Device requests an interrupt, and 2) Interrupt control flag is set.
2. Called before fetch of instruction so no increment of PC.
3. ISR will call the interrupt handler of the device requesting the interrupt. Interrupt vectors are used to store the addresses of the interrupt handlers.
   a) CUSP uses $FF8 - $FFB for its vectors. $FF8 for keyboard, $FF9 for printer vector, $FFA for tape, $FFB for timer.
4. If more than one device request an interrupt they are called in a specific pecking order. In CUSP these are: 1) Timer, 2) Tape, 3) Printer, and 4) Keyboard.
5. ISR should reset the device's ready bit before returning.
6. Only when all interrupt requests have been handled will the ISR return control to the program.

C. CUSP Interrupt instructions
1. SIE = Set the interrupt enable flag, EI := 1
2. CIE = Clear the interrupt enable flag, EI := 0
3. INT = Start the ISR of a device.
   a) Similar to JSR
   b) Pushes onto the stack $FF0000 + $8000 * OV + $4000 * EQ + $2000 * LT + $1000 * IE + PC (unincremented)
   c) Set PC to address contained in the device's interrupt vector.
   d) After the above push, clear interrupt enable flag. This prevents other devices interrupting the ISR.
4. IRTN = Return from ISR
   a) Similar to RTN
   b) Pops from the stack the value stored by INT and places the appropriate values in OV, EQ, LT, IE, and PC.

D. Tape and Printer interrupts
1. The ready bit doesn't automatically cause an interrupt because it will be set even when nothing is pending for the device.
2. Caused by change of status of the device, either the bit #7 (ready bit) or bit #5 (on-line bit in the case of the printer, mounted bit in the case of a tape).

IX. Queues
A. A first-in, first-out (FIFO) data structure; so items are dequeued in the same order in which they are enqueued.
B. Circular buffer implementation.
1. Head pointer points to next data to be dequeued.
2. Tail pointer points to where next data is to be enqueued.
3. Count is the number of items in the queue. Necessary because Head Pointer == Tail pointer can mean either the buffer is full (count == size of buffer) or empty (count == 0).
4. Enqueue operation:
   a) Ring[tail] := data
   b) tail := (tail + 1) mod size
   c) count := count + 1
5. Dequeue operation:
   a) Data := Ring[head]
   b) head := (head + 1) mod size
   c) count := count - 1
6. For character queues use FP to point to start of queue structure.

X. Uses of Buffers
A. Keyboard buffers.
   1. Allows user to keystrokes while program works on other operations.
   2. Allows editing of line without analysis.
   3. Keyboard typeahead
a) May not echo keystrokes until dequeued to prevent interference with any messages. No true in UNIX.

B. Printer buffers.
   1. Allows program to continue without waiting for slow printer.
   2. CUSP printer requires that the program start the printer ISR.
      a) Declare a program flag that indicates whether printing is running. When a print job is running, i.e. flag is true, then do nothing because the ISR will read the buffer when ready. If a print job is not running then start ISR by explicit call of INT, which will set running flag. ISR will clear running flag when buffer is empty.