Parallel I/O Ports

This lecture covers the design of parallel I/O ports. These simple ports are used to interface the CPU to I/O devices. After this lecture you should be able to: (1) design simple input, output and bidirectional I/O ports using registers, tri-state buffers and and open-collector buffers; and (2) write 8088 assembly language programs to read and write the individual bits of an I/O port.

I/O Ports

All useful microcomputer systems have input/output (I/O) devices. These I/O devices move data between the outside world and the computer. The interface between the CPU and these I/O devices is through registers in the address or I/O space of the processor. Through these registers the CPU can input (read) or output (write) a number of bits (typically a byte) at a time.

Typical examples of I/O port include output ports that drive LEDs, ports to scan a keypad, ports to control machinery, etc. More complex I/O interfaces such as floppy disk controllers or serial interface chips usually contain several I/O ports. Some ports are used to obtain status information about the interface through "status registers" and other ports can control the interface's operation through "control registers."

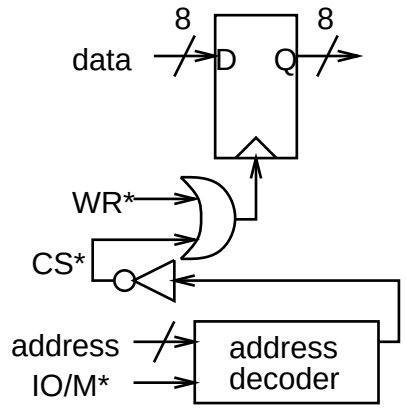
For example, each printer interface on the IBM PC has associated with it a status port that can be used to obtain certain status information (busy, on-line, out of paper, etc). The printer interface also has a control port that can be used to reset the printer and set the automatic line feed mode. In addition, there is an output port that is used to output the character to be printed.

Implementation of I/O Ports

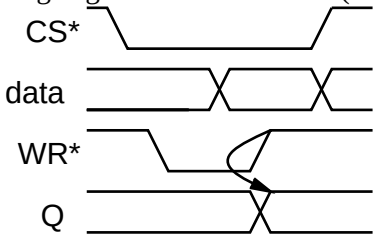
Output

Output ports are implemented using registers – multi-bit flip-flops with a common clock. The register's data inputs (D) are connected to the CPU data bus and the register's clock input is driven by the CPU write strobe (WR*). In addition, an address decoder is used to make sure the clock is only asserted when the CPU is addressing the desired IO or memory address. The rising edge of the write strobe loads the data into the register output (Q) and this output stays fixed until the register is written again.

The following schematic shows how a register could be connected to operate as an output port. The CPU's write strobe (WR*) is used to clock the data into the register, but only if the address on the CPU bus corresponds to the address of the output port:



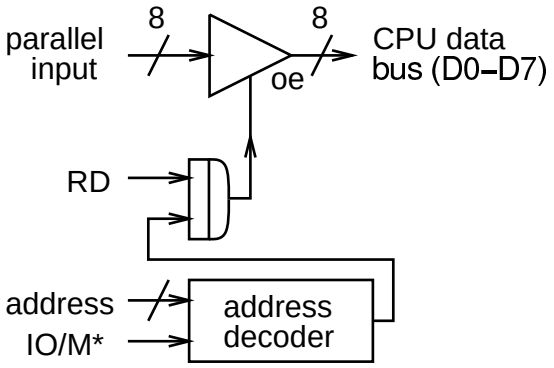
The following timing diagram shows the relationship between the signals. Note that the output is held after the rising edge of the write strobe (WR*):



Input

Input ports can also be implemented with a minimum of hardware. A tri-state buffer is used to connect the external digital input to the CPU's data bus during a read cycle if the CPU is addressing the memory or IO address assigned to the input port. The read strobe (RD*) is used to enable the buffer so that it connects the input to the CPU data bus.

The following schematic shows how a register could be connected to operate as a parallel input port. The CPU's inverted RD* strobe (RD) is used to enable the output of a tri-state buffer when RD is active and the address corresponds to the address of the input port:



The value read by the CPU will be the value on the input port at the time that the IN (if I/O mapped) or MOV (if memory-mapped) instruction is executed. This type of input port samples the value of the input at the time the instruction is executed.

Bi-Directional I/O Ports

By using open-collector outputs on an output port it's possible to use the same signal pins for both input and output. The open collector outputs are driven high by pull-up resistors and can be driven low by either the output port or by an external device. An input port is attached to these lines. The state of the I/O interface lines can be read by reading the input port.

Exercise: To what value must the outputs be set in order to be able to read from an external device?

Address and I/O Decoding

The design of address decoders for I/O ports is similar to the design for memory systems. A typical an I/O interface will only require a few (typically less than 16) ports (addresses). On some CPUs (such as the 8088) there are separate I/O and memory address spaces. In this case the decoder must enable the port only for the appropriate address space.

Software Aspects

The value on the output port is set with MOV (if the port is memory-mapped) or OUT (if the port is mapped into the I/O space) instructions. Similarly, the value on an input port is read with a MOV or IN instruction.

It's often necessary to set or clear a particular bit on an output port or to test the value of a particular bit on an input port. This can be done with bit masks and the bit-wise logical operations AND and OR.

To set a particular bit(s), the current output value is OR'ed with a bit-map which contains 1's in the bit positions to be set. To clear a particular bit(s), the current output value is AND'ed with a bit-map which contains 0's in the bit positions to be cleared. To test the value of a particular bit, the input value is ANDed with a bit-map which contains 1's in the bit position(s) to be tested.

```
in      al,60H      ; read from I/O port at 60H
and     al,80H      ; test MS bit
or      al,07H      ; set LS 3 bits
and     al,0BFH     ; clear bits 5 and 4
out     70H,al      ; write to I/O port at 70H
```

Often it's not possible to read the value written to an output port. If individual bits need to be changed, it's necessary to store the output value to RAM each time it's changed and then obtain the current output value from RAM.

For example, the following code clears the LS bit of a value that is being output to an 8-bit output port which is IO-mapped at port 80H. In this case the port is output-only so a copy of the output value is kept in the RAM memory location 'outval'.

```
mov     al,outval
and     al,0FEH,
out     80H,al
mov     outval,al
```

Exercise: The status port for a serial interface chip is located at I/O port 55H. Bit 2 (bits are usually numbered from 0 starting with the LS bit) will have the value 1 if a received character is available to be read (from another port on the chip). Write a section of 8088 assembly language code that checks to see if there is a character ready to be read.