Figure 8-5.
Control program for Example 8.4.

Figure 8-6.
Using device-select pulses to control a relay. An IN AL, 0 instruction will turn the relay on; an OUT 0, AL instruction will turn it off.
the contents of register AL is a “don’t care” when an output instruction is executed, and indeterminate when an input instruction is executed.

Memory-Mapped I/O
The address space of the 80x86 processors is divided into 4 GB of memory space and 64K of I/O space.\(^3\) To ensure that these two regions do not overlap, the control bus signal M/IO is provided. Memory instructions cause the control signal to be high; I/O instructions cause this signal to be low.

However, consider designing a 1-byte (or word or doubleword) read/write memory. We would use latches to store the data written during a memory write cycle, and tristate gates to drive the bus during a memory read cycle—exactly the same hardware we would use for an output or input port.

This is the essence of memory-mapped I/O. In hardware, it appears to be a conventional I/O port. But because it is mapped to a memory address, it is accessible in software using any of the memory read or write instructions. For example, the instruction MOV BH,[1000] becomes an input instruction (input the data at “port” 1000 to register BH). Indirect I/O is also possible. The instruction sequence

\[
\text{MOV SI}, 1000H \quad ; \text{Point SI at the port} \\
\text{MOV [SI]}, CX \quad ; \text{Output CX to the port}
\]

allows CX to be output to the 16-bit port at address DS:1000H.

As you can see, the advantage of memory-mapped I/O is the large number of instructions and addressing modes available for referencing memory. All of these now become potential I/O instructions. With I/O-mapped ports we are restricted to the simple IN and OUT instructions. Even using string I/O, the port address must be specified in register DX. Memory-mapped ports allow this address to be computed using any of the 80x86’s addressing modes.

---

**Self-Review 8.1 (Answers on page 464)**

8.1.1 When the instruction IN AL,01H is executed, the 80486 inputs the data byte over the _______ data bus lines.
(a) D0–D7
(b) D8–D15
(c) D16–D32
(d) D24–D31

8.1.2 For direct I/O, the 80x86 processors are restricted to _______ I/O ports with addresses in the range _______ to _______.

8.1.3 Indicate the logic levels required (1 or 0) on W/R and M/IO for (a) an I/O read cycle and (b) an I/O write cycle.

---

\(^3\)The 8086/88 has 1 MB of memory space; the 286 has 16 MB.
8.1.4 When an I/O port select signal is combined with the \texttt{IOR} or \texttt{IOW} control signals, the resulting signal is referred to as a \underline{________} \underline{________} \underline{________}.

8.1.5 Describe the change required to the output port in Figure 8-4 if it is to be mapped to port 3.

8.1.6 Which of the following instructions could \textit{not} be used for memory-mapped input?

(a) INC AX
(b) CMP DX,[1000]
(c) MOV BX,[1000]
(d) MOV BP,[BX]

8.2 Programmed I/O

Introduction

The I/O ports of a computer typically operate at different data rates. A hard disk drive, for example, might require the computer to input data at 10 MB/s. CD-ROM drives operate at 300–600KB/s. However, when inputting keystrokes from the operator, the data rate may fall to only one or two characters per second. If the processor is to operate efficiently, we will need to develop a strategy to control or \textit{synchronize} the flow of data between the processor and the widely varying data rates of its I/O devices.

In this section we will:

- Explain the "handshaking" signals exchanged between a parallel printer and microcomputer.
- Write a printer driver using programmed I/O.
- Explain why polling is an inefficient method of controlling an I/O device.

Parallel Printer Interface Example\textsuperscript{4}

Printers typically have buffers that can be filled by the computer at high speed. Once full, the computer must wait while the data in the buffer is printed. The buffer is then refilled and the process repeated. To facilitate the control process, most printer manufacturers have settled on a standard set of control and data signals that is now referred to as the \textit{Centronics Parallel Printer Interface}. In this section we will study this interface as an example of programmed I/O.

Signal Definitions. In the PC, the parallel printer port is usually referred to as LPT1 or LPT2. Each of these "ports" is actually made up of three ports:

- Port A—the printer data port
- Port B—the printer control signal status port
- Port C—the printer control port

\textsuperscript{4}In Appendix C we provide a schematic diagram of the parallel printer port used in the PC. The discussion in this section is based on an analysis of this hardware.
Table 8–2.
Parallel Printer Port Bit Definitions

<table>
<thead>
<tr>
<th>Port A—DATA</th>
<th>8-Bit Output Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA7</td>
<td>PA6</td>
</tr>
<tr>
<td>Data 8</td>
<td>Date 7</td>
</tr>
<tr>
<td>PA5</td>
<td>PA4</td>
</tr>
<tr>
<td>Data 6</td>
<td>Data 5</td>
</tr>
<tr>
<td>PA3</td>
<td>PA2</td>
</tr>
<tr>
<td>Data 4</td>
<td>Data 3</td>
</tr>
<tr>
<td>PA2</td>
<td>PA1</td>
</tr>
<tr>
<td>Data 2</td>
<td>Data 1</td>
</tr>
<tr>
<td>PA1</td>
<td>PA0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port B—Status</th>
<th>5-Bit Input Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB7</td>
<td>PB6</td>
</tr>
<tr>
<td>BUSY</td>
<td>ACKNLG</td>
</tr>
<tr>
<td>PB5</td>
<td>PB4</td>
</tr>
<tr>
<td>PE</td>
<td>SLCT</td>
</tr>
<tr>
<td>PB3</td>
<td>PB2</td>
</tr>
<tr>
<td>ERROR</td>
<td>Not Used</td>
</tr>
<tr>
<td>PB2</td>
<td>PB1</td>
</tr>
<tr>
<td>Not Used</td>
<td>Not Used</td>
</tr>
<tr>
<td>PB1</td>
<td>PB0</td>
</tr>
<tr>
<td>Not Used</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port C—Control</th>
<th>5-Bit Output Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC7</td>
<td>PC6</td>
</tr>
<tr>
<td>Not Used</td>
<td>Not Used</td>
</tr>
<tr>
<td>PC5*</td>
<td>PC4</td>
</tr>
<tr>
<td>PC3</td>
<td>PC2</td>
</tr>
<tr>
<td>DIR</td>
<td>IRQEN</td>
</tr>
<tr>
<td>SLCT IN</td>
<td>INIT</td>
</tr>
<tr>
<td>PC1</td>
<td>PC0</td>
</tr>
<tr>
<td>AUTO</td>
<td>STROBE</td>
</tr>
</tbody>
</table>

In modern computers port A is bidirectional. When bit 5 of port C is set, the bidirectional mode is enabled. Software can then control the direction of the port by writing to an additional register.

Table 8–2 gives the (parallel printer) signal names associated with the bits in each port. When interfaced as LPT1, the three ports are mapped to I/O addresses 378–37AH; when interfaced as LPT2, the addresses are 278–27AH.5

In the Centronics standard the computer and printer are connected via a cable with 25 pins at the computer end and 36 pins at the printer end. Table 8–3 provides the details on the specific pin assignments: Figure 8–7 shows the hardware interface.

The 8-bit data byte to be output to the printer is sent to port A. Data output to this port will be latched when the STROBE input (bit PC0) is pulsed low for 0.5 μs or longer.6 Figure 8–8 illustrates the timing. The data requires a 0.5 μs setup time before the leading edge of STROBE occurs. A 0.5 μs hold time is also required after the STROBE signal returns high.

Two other control signals are provided by the printer, labeled BUSY and ACKNLG. Returning to Table 8–3, BUSY is an active-high signal that indicates that the printer is “busy” printing a character, has some error condition, or is in an OFFLINE state. Taken literally, it means I am busy now and can’t accept data from you.

5A parallel printer port may also be found at address 3BCH (this is used on early monochrome video adapters). In addition, a fourth parallel port may be mapped to base address 2BCH. During the boot-up process, the BIOS scans ports 3BCH, 378H, 278H, and then 2BCH. The first active port it finds is assigned the name LPT1, the second LPT2, etc. Four different ports are thus possible. Some BIOS versions support only LPT1 and LPT2—even though the hardware for LPT3 and LPT4 may be present.

6The PB7, PC3, PC1, and PC0 signals in Table 8–2 are input or output through inverters in the PC's parallel port hardware (see Appendix C). Thus, writing a logic 1 to bit 0 of port C will actually result in a logic 0 being output by the parallel interface card. Similarly, a logic 1 input from bit 7 of port B will be input as a logic 0.
**Table 8-3.**
Parallel Printer Signal Descriptions

<table>
<thead>
<tr>
<th>Port</th>
<th>Printer 36-pin</th>
<th>Computer 25-pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC0</td>
<td>1</td>
<td>1</td>
<td>STROBE</td>
<td>In</td>
<td>STROBE pulse to read data in. Pulse width must be more than 0.5μs at receiving terminal. The signal level is normally “HIGH”; read-in of data is performed at the “LOW” level of this signal.</td>
</tr>
<tr>
<td>PA0</td>
<td>2</td>
<td>2</td>
<td>DATA 1</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>PA1</td>
<td>3</td>
<td>3</td>
<td>DATA 2</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>PA2</td>
<td>4</td>
<td>4</td>
<td>DATA 3</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>PA3</td>
<td>5</td>
<td>5</td>
<td>DATA 4</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>PA4</td>
<td>6</td>
<td>6</td>
<td>DATA 5</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>PA5</td>
<td>7</td>
<td>7</td>
<td>DATA 6</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>PA6</td>
<td>8</td>
<td>8</td>
<td>DATA 7</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>PA7</td>
<td>9</td>
<td>9</td>
<td>DATA 8</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>PB6</td>
<td>10</td>
<td>10</td>
<td>ACKNLG</td>
<td>Out</td>
<td>Approx. 5-μs pulse. “LOW” indicates that data has been received and that the printer is ready to accept other data.</td>
</tr>
<tr>
<td>PB7</td>
<td>11</td>
<td>11</td>
<td>BUSY</td>
<td>Out</td>
<td>A “HIGH” signal indicates that the printer cannot receive data. The signal becomes “HIGH” in the following cases: 1. During data entry 2. During printing operation 3. In OFFLINE state 4. During printer error status</td>
</tr>
<tr>
<td>PB5</td>
<td>12</td>
<td>12</td>
<td>PE</td>
<td>Out</td>
<td>A “HIGH” signal indicates that the printer is out of paper.</td>
</tr>
<tr>
<td>PB4</td>
<td>13</td>
<td>13</td>
<td>SLCT</td>
<td>Out</td>
<td>This signal indicates that the printer is in the selected state.</td>
</tr>
<tr>
<td>PCI</td>
<td>14</td>
<td>14</td>
<td>AUTO FEED XT</td>
<td>In</td>
<td>With this signal being at “LOW” level, the paper is automatically fed one line after printing.</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>—</td>
<td>NC</td>
<td>—</td>
<td>Not used.</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>—</td>
<td>OV</td>
<td>—</td>
<td>Logic GND level.</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>—</td>
<td>CHASSIS-GND</td>
<td>—</td>
<td>Printer chassis GND. In the printer, the chassis GND and the logic GND are isolated from each other.</td>
</tr>
</tbody>
</table>
### Table 8-3. (continued)

<table>
<thead>
<tr>
<th>Port</th>
<th>Printer 36-pin</th>
<th>Computer 25-pin</th>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18</td>
<td>—</td>
<td>NC</td>
<td>—</td>
<td>Not used.</td>
</tr>
<tr>
<td></td>
<td>19 to 30</td>
<td>18-25</td>
<td>GND</td>
<td>—</td>
<td>TWISTED-PAIR RETURN signal GND level.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INIT</td>
<td>In</td>
<td>When the level of this signal becomes “Low,” the printer controller is reset to its initial state and the print buffer is cleared. This signal is normally at “HIGH” level, and its pulse width must be more than 50 μs at the receiving terminal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>ERROR</td>
<td>Out</td>
<td>The level of this signal becomes “LOW” when the printer is in: 1. PAPER END state 2. OFFL INF state 3. Error state</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>18-25</td>
<td>GND</td>
<td>—</td>
<td>Same as with printer pins 19 to 30.</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>—</td>
<td>NC</td>
<td>—</td>
<td>Not used.</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Pulled up to +5V through 4.7-kΩ resistance.</td>
</tr>
<tr>
<td>PC3</td>
<td>36</td>
<td>17</td>
<td>SLCT IN</td>
<td>In</td>
<td>Data entry to the printer is possible only when the level of this signal is “LOW.”</td>
</tr>
</tbody>
</table>

**Notes**
1. “Direction” refers to the direction of signal flow as viewed from printer.
2. “Return” denotes “TWISTED-PAIR RETURN” and is to be connected at signal ground level. As to the wiring for the interface, be sure to use a twisted-pair cable for each signal and never fail to complete connection on the Return side. To prevent noise effectively, these cables should be shielded and connected to the chassis of the host computer and the printer, respectively.
3. All interface conditions are based on TTL level. Both the rise and fall times of each signal must be less than 0.2μs.
4. Data transfer must not be carried out by ignoring the ACKNLG or BUSY signal. Data transfer to this printer can be carried out only after confirming the ACKNLG signal or when the level of the BUSY signal is “LOW.”

Source: Seiko Epson Corporation.

The ACKNLG signal is an active-low pulse provided by the printer after a character has been accepted and printed. Note the difference between these two signals. BUSY is best suited for a level-triggered input; ACKNLG is most appropriate with an edge-triggered input.7

---

7As shown in Figure 8–7, setting bit PC4 allows the ACKNLG signal to drive the IRQ7 interrupt input of the PC. Interrupts are covered in detail in Chapter 9. In addition, an interrupt-driven parallel printer interface is also discussed in that chapter.
Figure 8-7.
PC to parallel printer interface. Three ports—A, B, and C—are required. In this example the port assignments correspond to LPT1.

Figure 8-8.
Parallel printer timing showing the relationship between the handshaking signals. (Courtesy of Seiko Epson Corporation.)
FIGURE 12.16  Centronics parallel printer connections

This sequence, illustrated in Figure 12.17, performs handshaking between the printer and the computer, and guarantees that the computer does not send data to the printer faster than the printer can accept it.

FIGURE 12.17  Printer/computer handshaking
The STROBE, BUSY, and ACKNLG signals form a set of "handshaking" signals exchanged between the computer and printer. The computer extends its "hand" with the STROBE pulse, saying, *Here is the data.* The printer acknowledges via the ACKNLG pulse saying, *I've got it. You can send me some more.* BUSY is the same as ACKNLG but provides a level to be monitored instead of a pulse.

**Controlling the Printer.** Before data can be writing to the printer, the printer must be *initialized.* This is accomplished by writing to the control port (Port C).

---

**Example 8.5** Determine the code to be written to the LPT1 control port in Figure 8–7 such that the printer port operates unidirectionally, IRQ7 is disabled, the printer is selected, the printer controller is reset, line feeds are automatically added after each line, and the STROBE input is inactive.

**Solution** Referring to Table 8–2 for the Port C bit definitions, the following bit pattern is required:

(a) PC7 and PC6 are "don't cares"

(b) PC5 = 0 (The printer port is unidirectional)

(c) PC4 = 0 (Disable IRQ7)

(d) PC3 = 0 (Put the printer in the selected state)

(e) PC2 = 0 (Reset the printer controller)

(f) PC1 = 0 (Enable automatic line feeds)

(g) PC0 = 1 (STROBE = high)

The code to output would appear to be XX00 0001. However, as shown in Figure 8–7, bits PC3, PC1, and PC0 are inverted by the printer adapter before being output to the printer. Taking this into account, the proper initialization code is XX00 1010 = 0AH (choosing the Xs to be 0).

---

Once the printer is initialized, it is ready to receive data. This is accomplished by outputting the data to port A, and then taking the STROBE input momentarily low. The next character is then output, and the process repeated. The STROBE signal can be generated in software by outputting a 1, a 0, and then a 1 again. The result will be a brief active-low pulse.

---

**Example 8.6** Determine the codes to write to the LPT1 control port in Figure 8–7 such that (a) STROBE is high; and (b) STROBE is low. Assume the printer operates as described in Example 8.5.

**Solution** The printer's INIT input must first be made inactive so that the printer controller is not continuously being reset. This requires PC2 = 1. Now to make the STROBE signal high, PC0 = 0 (recall that this bit is inverted). The code is thus: XX00 1110 = 0EH. To make the STROBE signal low, PC0 = 1. The code is XX00 1111 = 0FH.
Polling. Figure 8.9 flowcharts the printer driver software required to transfer data between the computer and printer. In this example, we assume the data to be printed is stored in consecutive memory locations thought of as a print buffer. A pointer is used to identify the head of this buffer; a counter is used to store the total number of bytes to be printed.

The decision block, BUSY = 1?, forms a polling loop in which the computer continually tests the printer's BUSY flag. When this line goes low, the processor fetches a byte from the buffer, outputs it to the printer, pulses the STROBE input, advances the table pointer, and decrements the byte counter. If more data remains to be printed, control returns to the polling loop.

Control Program. Figure 8-10 (on page 433-435) lists the 80x86 code required to control the parallel printer interface in Figure 8-7. We begin by setting up a PRINT_DATA segment, where the number of bytes to be printed and the starting address of this data (in segment:offset format) are stored. [Figure 8-10 (a)]. The program is written as a procedure, so it is up to the calling program to initialize this PRINT_DATA segment before calling the main program.

Several equates are used to make the program more readable. Note that the I/O addresses are assumed to be 378–37AH corresponding to LPT1 on the PC. S_HIGH and S_LOW are defined such that, when written to the printer control port, the STROBE signal will go high or low accordingly.

The program code begins in Figure 8-10(b). Register CX is used as the byte counter and DS:SI as the data pointer. Following printer initialization, the program drops into the polling loop waiting for the printer to be ready. When it is, the LODSB instruction fetches the data byte to AL, where it is then output to the data port. The STROBE signal is then pulsed low and back high (via software). The sequence is then repeated until all bytes have been written (CX = 0).

The most important point to note about this program is the use of the STROBE and BUSY signals to synchronize the microprocessor and peripheral. Because the transfer of data to the printer is done under program control, the technique is referred to as programmed I/O.

The Polling Loop. In Figure 8-10, the four instructions

```
POLL:  MOV  DX, STATUS
       IN   AL, DX
       TEST  AL, 10000000B
       JZ    POLL
```

form the polling loop. In effect, they ask the printer, “Are you ready for more data?” Initially, when the printer’s buffer is empty, the answer will be Yes. However, because the data rate of the computer is so fast (even a 5 MHz 8086 can output more than 60,000 characters/s), the buffer will quickly fill up. Several seconds may then elapse before the printer is ready for more data. During this time, the computer will be repeatedly asking the printer if it is ready. Several thousand such requests may occur before the answer is finally Yes.

This may suggest that polling is a rather inefficient way of controlling the printer. Most of the time the processor is simply waiting for the printer to be READY. And even
Figure 8–9.
Flowchart for the printer control program.

START

- Initialize printer

- Get number of bytes to be printed and store as byte counter

- Get address of the first byte to be printed

- BUSY = 1?
  - Yes
  - Fetch byte from buffer
  - Output byte to printer
  - Pulse STROBE for 0.5 µs
  - Advance buffer pointer
  - Decrement byte counter

- Counter = 0?
  - Yes
  - RETURN
  - No

- No
when it is, only a few milliseconds are required to refill the print buffer. Then it’s back to waiting for the data to be printed.

But maybe you are thinking, “What does it matter? What else does the processor have to do anyway?” Depending on the system, perhaps nothing. But if the printer is being used to print a 100-page report, you may have a long wait before the system can be used for some other job. It also seems intuitively wrong to have a microprocessor as powerful as the Pentium (or 386, or 486, etc.) simply “spinning its wheels” waiting for the slow printer to be done.

So now you ought to be thinking about a way to improve the efficiency of this process. Ideally, it would let the processor “periodically check in on the printer” while most of its time is devoted to other (more useful) tasks. In Chapter 9 we will see exactly how to do this.

;Function: Polled printer driver for LPT1.
;Written as a far procedure.

;Inputs: PRINT_DATA segment holds number of bytes
;to be printed and the address of the buffer.
;Outputs: Characters in the buffer are output to LPT1.
;Calls: None
;Destroys: AX, CX, SI, DS, flags

0000  PRINT_DATA SEGMENT WORD
0000 0000       NUMB   DW  ?           ;Number of bytes to print
0002 00000000   ADR    DD  ?           ;Address of first byte

0006  PRINT_DATA ENDS

= 0378   LPT1 EQU 378H   ;Printer data port
= 0379   STATUS EQU 379H  ;Printer status port
= 037A   Control EQU 37AH  ;Printer control port
= 000A   INIT EQU 0AH   ;Unidirectional, no TRQ,
             ;select printer, init, auto
             ;STROBE=1
= 000E   S_HIGH EQU 0EH   ;STROBE=1 and no init
= 000F   S_LOW EQU 0FH   ;STROBE=0 and no init

Figure 8–10.
Control program for the parallel printer interface in Figure 8–7.
; Initialize pointers:
; DS:SI to start of data
; CX with number of bytes to be printed
0000 B8 ---- R 
0003 8E D8
0005 8B 0E 0000 R
0009 C5 36 0002 R
000D FC

; Initialize and select printer, auto line feed, STROBE=1
000E B0 0A
0010 BA 037A
0013 EE

; Poll the printer waiting for BUSY to be low
0014 B0 0E
0016 EE
0017 BA 0379
001A EC
001B A8 80
001D 74 F8

Figure 8–10. (continued)
;Printer is ready, fetch and output a byte
LODSB  ;Get byte
        ;and advance pointer.
MOV     DX, LPT1  ;Data port access
OUT     DX, AL    ;Output to printer

;Strobe the printer
MOV     AL, S_LOW  ;STROBE=0
MOVV    DX, CONTROL ;Control port access
OUT     DX, AL     ;Write to LPT1 control port

;Repeat the polling loop until all data has been printed
LOOP    NEXT       ;Do CX times
        RET         ;Then return
FIG8_10 ENDP
CODE    ENDS
END

Figure 8–10. (continued)