Analogous to the AND are three versions of the OR instruction:

OR immediate	*OR register*	*OR memory*
ORI data	ORA r	ORA M
(A) ← (A) ∨ (byte 2)	(A) ← (A) ∨ (r)	(A) ← (A) ∨ ((H)(L))

Flags: All; CY and AC are cleared

The OR instructions are used to merge the bits of two operands, i.e., to set specific bits of the accumulator. The accumulator is left with 1s in the bit positions corresponding to 1s in either of the operands. For example, the following program segment outputs the ASCII character equivalent of a binary coded decimal digit in register B:

```
MOV A, B
ORI 30H
OUT ASCII
```

To change a single bit of an output port without changing any other bits—a necessity when several bits of an output latch are controlling separate devices—a copy of the last control word output to the port, its ***image***, is kept in memory, in location CWORD. To set the bit, bit 4 in this case, the following program segment is used:

```
LXI H, CWORD    ;load address of control word into HL
MOV A, M        ;transfer copy of present control word
                ;to accumulator
ORI 10H         ;set bit 4
MOV M, A        ;update copy of control word
OUT PORT0       ;output new control word to external latch
```

To reset bit 4:

```
LXI H, CWORD    ;load address of control word into HL
MOV A, M        ;transfer copy of present control word
                ;to accumulator
ANI 0EFH        ;reset bit 4
MOV M, A        ;update copy of control word
OUT PORT0       ;output new control word
```

The ORA A instruction is also frequently used to affect the flags following a data transfer to the accumulator without altering the contents of the accumulator.

The EX-OR operation sets the bits of the accumulator corresponding to the bit positions where the two operands differ; the bit positions corresponding to those where the operands are the same are cleared. In other words, the bits of the accumulator that correspond to 1 bits in the data EX-ORed with the accumulator are complemented. The other bits in the accumulator are not altered. The three

versions of EX-OR are

EX-OR immediate	*EX-OR register*	*EX-OR memory*
XRI data	XRA r	XRA M
(A) ← (A) ∀ (byte 2)	(A) ← (A) ∀ (r)	(A) ← (A) ∀ ((H)(L))

Flags: All; the CY and AC are cleared

When data is EX-ORed with itself, the result is zero. XRA A EX-ORs the accumulator with itself, thus clearing the accumulator and the carry flag.

Frequently, an input port transmits status information to the microprocessor concerning one or more external devices or processes. To ascertain whether any change has occurred in the status of the external process, a copy of the prior status is kept in a memory location and compared with the present status:

```
LXI H, STATUS   ;load HL with address of copy of prior status
MOV B, M        ;transfer copy of prior status to register B
IN PORT1        ;input present status
MOV M, A        ;update status copy
XRA B           ;test for status change
```

At the end of this sequence, the zero flag is set to indicate no change in the status or cleared to indicate a change.

The comparison of 2 bytes to determine which is greater in terms of binary magnitude can be carried out in hardware by using MSI comparator circuits. The software equivalent of the hardware comparator is the compare instruction. This instruction compares 2 bytes, one of which is in the accumulator, by subtracting the second byte from it. This instruction is unusual in that neither of the bytes compared is altered. The compare instruction has three forms:

Compare immediate	*Compare register*	*Compare memory*
CPI data	CMP r	CMP M
(A) - (byte 2)	(A) - (r)	(A) - ((H)(L))

Flags: All; Z = 1 if (A) = (r); CY = 1 if (A) < (r)

After execution, the flag register is tested to determine whether the bytes are equal or, if unequal, which is greater. Assuming that the bytes of data being compared are interpreted as unsigned binary numbers, the zero flag is set if they are equal. If the content of the accumulator is greater than the byte with which it is compared, the carry flag is cleared. Since the compare instruction is used exclusively for setting or clearing flags on which branch decisions are made, examples of its application are deferred until branch instructions are introduced later in this chapter.

The remaining logic instructions in the instruction set are those that rotate the contents of the accumulator. The action of the *rotate accumulator left*, RLC, instruction is indicated by the following register transfer expressions and diagram: