The following program first loads CL with value 55H, then moves this value around to various registers inside the CPU.

```
MOV   CL,55H      ;move 55H into register CL
MOV   DL,CL       ;copy the contents of CL into DL (now DL=CL=55H)
MOV   AH,DL       ;copy the contents of DL into AH (now AH=DL=55H)
MOV   AL,AH       ;copy the contents of AH into AL (now AL=AH=55H)
MOV   BH,CL       ;copy the contents of CL into BH (now BH=CL=55H)
MOV   CH,BH       ;copy the contents of BH into CH (now CH=BH=55H)
```

The use of 16-bit registers is demonstrated below.

```
MOV   CX,468FH    ;move 468FH into CX (now CH=46,CL=8F)
MOV   AX,CX       ;copy contents of CX to AX (now AX=CX=468FH)
MOV   DX,AX       ;copy contents of AX to DX (now DX=AX=468FH)
MOV   BX,DX       ;copy contents of DX to BX (now BX=DX=468FH)
MOV   DI,BX       ;now DI=BX=468FH
MOV   SI,DI       ;now SI=DI=468FH
MOV   DS,SI       ;now DS=SI=468FH
MOV   BP,DI       ;now BP=DI=468FH
```

In the 8086 CPU, data can be moved among all the registers shown in Table (1-2) (except the flag register) as long as the source and destination registers match in size. such as "MOV AL,DX" will cause an error, since one cannot move the contents of a 16-bit register into an 8-bit register. The exception of the flag register means that there is no such instruction as "MOV FR,AX". Loading the flag register is done through other means, discussed in later chapters.

If data can be moved among all registers including the segment registers, can data be moved directly into all registers?

The answer is no. Data can be moved directly into non segment registers only, using the May instruction. For example, look at the following instructions to see which are legal and which are illegal.

```
MOV   AX,58FCH    ;move 58FCH into AX    (LEGAL)
MOV   DX,6678H    ;move 6678H into DX    (LEGAL)
MOV   SI,924BH    ;move 924B  into SI    (LEGAL)
MOV   BP,2459H    ;move 2459H into BP    (LEGAL)
MOV   DS,2341H    ;move 2341H into DS    (ILLEGAL)
MOV   CX,8876H    ;move 8876H into CX    (LEGAL)
MOV   CS,3F47H    ;move 3F47H into CS    (ILLEGAL)
MOV   BH,99H      ;move 99H into BH      (LEGAL)
```

From the discussion above, note the following three points:

1. Values cannot be loaded directly into any segment register (CS, OS, ES, or SS). To load a value into a segment register, first load it to a non segment register and then move it to the segment register, as shown next.

```
MOV   AX,2345H      ;load 2345H into AX
MOV   DS,AX         ;then load the value of AX into DS

MOV   DI,1400H      ;load 1400H into DI
MOV   ES,DI         ;then move it into ES, now ES=DI=1400
```

2. If a value less than FFH is moved into a 16-bit register, the rest of the bits are assumed to be all zeros.

For example, in "MOV BX,5" the result will be BX ~ 0005; that is, BH ~ 00 and BL ~ 05.

3. Moving a value that is too large into a register will cause an error.

```
MOV   BL,7F2H       ;ILLEGAL: 7F2H is larger than 8 bits
MOV   AX,2FE456H    ;ILLEGAL: the value is larger than AX
```

ADD instruction:

The ADD instruction has the following format:

ADD destination, source ; ADD the source operand to the destination

The ADD instruction tells the CPU to add the source and the destination operands and put the result in the destination. To add two numbers such as 25H and 34H, each can be moved to a register and then added together

```
MOV   AL,25H        ;move 25 into AL
MOV   BL,34H        ;move 34 into BL
ADD   AL,BL         ;AL = AL + BL
```

Executing the program above results in AL ~ 59H (25H + 34H ~ 59H) and BL ~ 34H. Notice that the contents of BL do not change. The program above can be written in many ways, depending on the registers used. Another way might be:

```
MOV   DH,25H        ;move 25 into DH
MOV   CL,34H        ;move 34 into CL
ADD   DH,CL         ;add CL to DH: DH = DH + CL
```

The program above results in DH = 59H and CL = 34H. There are always many ways to write the same program. One question that might come to mind after looking at the program above is whether it is necessary to move both data items into registers before adding them together. The answer is no, it is not necessary.

Look at the following variation of the same program:

```
MOV   DH,25H        ;load one operand into DH
ADD   DH,34H        ;add the second operand to DH
```

In the case above, while one register contained one value, the second value followed the instruction as an operand. This is called an immediate operand. The examples shown so far for the ADD and MOV instructions show that the source operand can be either a register or **immediate data**. In the examples above, the destination operand has always been a register.

The largest number that an 8-bit register can hold is FFH. To use numbers larger than FFH (255 decimal), 16-bit registers such as AX, BX, CX, or DX must be used.

For example, to add two numbers such as 34EH and 6A5H, the following program can be used:

```
MOV   AX,34EH       ;move 34EH into AX
MOV   DX,6A5H       ;move 6A5H into DX
ADD   DX,AX         ;add AX to DX: DX = DX + AX
```

Running the program above gives DX = 9F3H (34E + 6A5 = 9F3) and AX = 34E. Again, any 16-bit non segment registers could have been used to perform the action above:

```
MOV   CX,34EH       ;load 34EH into CX
ADD   CX,6A5H       ;add 6A5H to CX (now CX=9F3H)
```

The general-purpose registers are typically used in arithmetic operations. Register AX is sometimes referred to as the accumulator.