Chapter-3

Organization of the IBM Personal Computers

3.2 Organization of the 8086/8088 Microprocessors

3.2.1. Registers

The registers are classified according to they perform. In general, data registers hold data for an operation, address registers hold the address of an instruction or data and a status register keeps current status of the processor.

The 8086 microprocessor has four general data registers; the address registers are divided into segment, pointer, and index registers; and the status register is called FLAGs register. In total, there are fourteen 16-bit registers (Fig. 3.2).

3.2.2. Data Registers: AX, BX, CX, DX

They are: AX, BX, CX, and DX. The high and low bytes of the data registers can be accessed separately. The high bytes are AH, BH, CH, DH and low bytes are AL, BL, CL, DL respectively. This arrangement gives us more registers to use when dealing with byte-size data.

AX (Accumulator Register)

AX is preferred for arithmetic, logic and data transfer instructions. In multiplication and division operations, one of the numbers involved must be in AX or AL. Input and output operations also require the use of AL and AX.

BX (Base Register)

BX is also serve as an address register.

CX (Count Register)

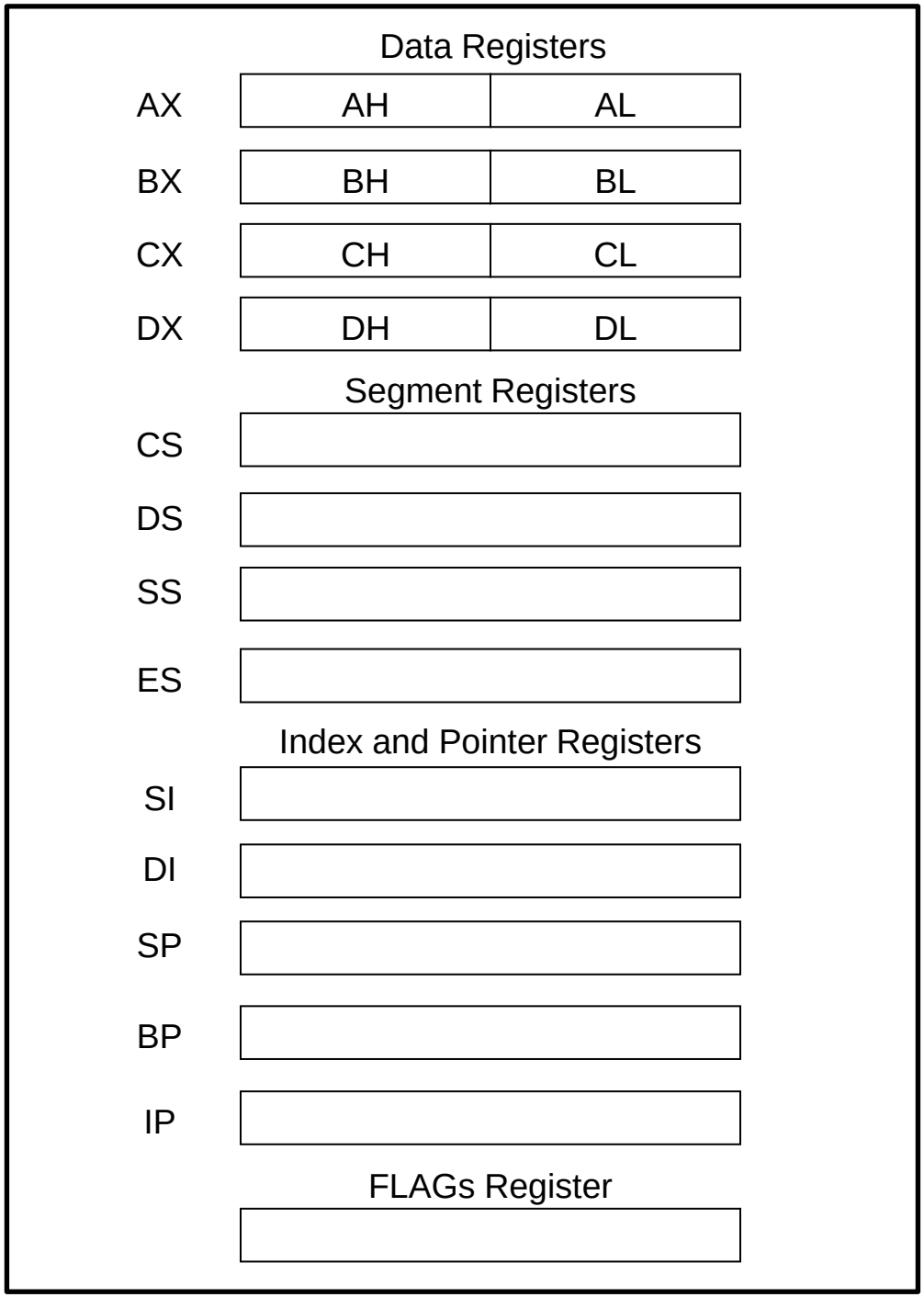
Program loop constructions are facilitated by the use of CX, which serves as a loop counter. This register has important role in string, shift and rotate operations.

DX (Data Register)

DX is used in multiplication and division. It is also used in I/O operations.

Figure 3.1

8086 Registers



3.2.3. Segment Registers: CS, DS, SS,ES

The bus interface unit contains four 16-bit segment registers. They are: code segment (CS) register, data segment (DS) register, stack segment (SS) register, extra segment (ES) register.

These registers are used to hold the upper 16-bit of the starting address of four memory segment. The 8086 bus interface unit sends out 20-bit addresses, so it can address any of 2^{20} bytes (1 megabyte) of memory. However, at any time the 8086 only works with four 65,536 (normally called 64K) segments.

The 8086 processor assigns a 20-bit **physical address** to its memory locations. The first five bytes in memory have the following addresses:

00000000000000000000
00000000000000000001
00000000000000000010
00000000000000000011
00000000000000000100

In hexadecimal form the addresses are expressed:

00000H
00001H
00002H
:
:
FFFFFH

Memory Segment

A **memory segment** is a block of 2^{16} (or 64K) consecutive memory bytes. Each segment is identified by a **segment number**, starting with 0. A segment number is 16 bits, so the highest segment number is FFFFH.

Within a segment, a memory location is specified by giving an **offset**. This is the number of bytes from the beginning of the segment. With a 64-KB segment, the offset can be given as a 16-bit number. The first byte in a segment has offset 0. The last offsets in a segment is FFFFH.

Segment : Offset Address

A memory location may be specified by providing a segment number and an offset, written in the form *segment:offset*; this is known as a **logical address**. For example, A4FBH:4872H means offset 4872H within a segment A4FBH. To obtain a 20-bit physical address, the 8086 processor first shifts the segment address 4 bits to the left (this is equivalent to multiplying by 10H), and then adds the offset. Thus the physical address for A4FB:4872 is

```
  A4FB0H
+ 4872H
--------
  A9822H        (20-bit physical address)
```

Location of Segments

Segment 0 starts at address 0000:0000H=00000H and ends at 0000:FFFFH=0FFFFH. Segment 1 starts at address 00001:0000H = 00010H and ends at 0001:FFFFH = 1000FH.

As we can see, there is a lot of overlapping between segments. Figure 3.2 shows the locations of the first three memory segments. The segments start every 10H=16 bytes and the starting address of a segment always ends with a hex digit 0. This 16 bytes is called a **paragraph**. The last address (divisible by 16 or ends with 0H) is called a **paragraph boundary**.

Because segments may overlap, the segment:offset form of an address is not unique, as the following example shows.

Example 3.1: For the memory location whose physical address is specified by 1256AH, give the address in segment:offset form for segments 1256H and 1240H.

Solution: Let X be the offset in segment 1256H and Y the offset in segment 1240H. We have

1256AH = 12560H + X and 1256AH = 12400H + Y

and so

X = 1256AH – 12560H = AH and Y = 1256AH – 12400H = 16AH

thus

1256AH = 1256:000AH = 1240:016AH

It is possible to calculate the segment number when the physical and the offset are given.

Example 3.2: A memory location has physical address 80FBDH. In what segment does it have offset BFD2H?

Solution: We know that

physical address = segment x 10H + offset

Thus

segment x 10H = physical address - offset

in this example

$$\begin{array}{r} \text{physical address} = 80FD2H \\ -\ \text{offset} = \quad BFD2H \\ \hline \text{segment} \times 10H = 75000H \end{array}$$

So the segment must be 7500H.

3.2.4. Pointer and Index Registers: SP, BP, SI, DI

The SP, BP, SI, and DI registers normally point to memory locations. They contain the offset address of memory locations. Unlike segment registers, the pointer and index registers can be used in normal arithmetic operations.

SP (Stack Pointer)

The SP register is used in conjunction with SS for accessing the stack segment.

BP (Base Pointer)

The BP register is used primarily to access data on the stack. However, unlike SP, BP can also be used to data in other segments.

SI (Source Index)

The SI register is used to point memory locations in the data segment. By incrementing the contents of SI, consecutive memory locations can be accessed.

DI (Destination Index)

The DI register perform the same functions as SI. There is a class of instructions called string operations, that use DI to access memory locations addressed by ES.

IP (Instruction Pointer)

To access instructions, the 8086 uses the register CS and IP. The CS register contains, the segment number of the next instruction, and the IP contains the offset. IP is updated each time an instruction is executed.

3.2.6. Flags Register

A flag is a flip-flop that indicates some condition produced by the execution of an instruction. It also controls certain operations of the execution unit. A 16-bit flag register in the execution unit contains nine active flags.

Six of the nine flags are called status flags and used to indicate some conditions produced by execution of some instructions. For example, when a subtraction operation results in a 0, the ZF (zero flag) is set to 1 (true).

The three flags that are used to control certain operation of the processor are called control flags. They are used for interruption (interrupt flag - IF) and some string operations.