1.4 INTRODUCTION TO PROGRAM SEGMENTS

A typical assembly language program consists of at least three segments: a **code segment**, a **data segment**, and a **stack segment**.

The **code segment** contains the Assembly language instructions that perform the tasks that the program was designed to accomplish. The **data segment** is used to store information (data) that needs to to be processed by the instructions in the code segment. The **stack** is used to store information temporarily.

Origin and definition of the segment

A segment is an area of memory that includes up to 64K bytes and begins on an address evenly divisible by 16 (such an address ends in OH). The segment size of 64K bytes came about because the 8085 microprocessor could address a maximum of 64K bytes of physical memory since it had only 16 pins for the address lines (2 16 = 64K). This limitation was carried into the design of the 8088/86 to ensure compatibility. Whereas in the 8085 there was only 64K bytes of memory for all code, data, and stack information, in the 8088/86 there can be up to 64K bytes of memory assigned to each category. Within an Assembly language program, these categories are called the code segment, data segment, and stack segment. For this reason, the 8088/86 can only handle a maximum of 64K bytes of code and 64K bytes of data and 64K bytes of stack at any given time, although it has a range of 1 Mega byte of memory because of its 20 address pins (2^{20} = 1 megabyte). How to move this window of 64K bytes to cover all 1 megabyte of memory is discussed below, after we discuss logical address and physical address

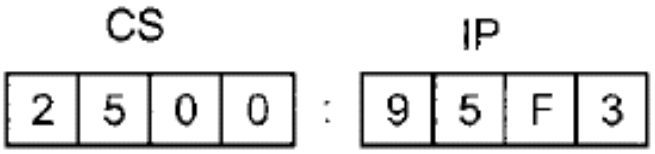
Logical address and physical address

In Intel literature concerning the 8086, there are three types of addresses mentioned frequently: the **physical address**, the **offset address**, and the **logical address**.

The **physical address** is the 20-bit address that is actually put on the address pins of the 8086 microprocessor and decoded by the memory interfacing circuitry. This address can have a range of 00000H to FFFFFH for the 8086 and real-mode 286,386, and 486 CPUs. This is an actual physical location in RAM or ROM within the 1 megabyte memory range.

The **offset address** is a location within a 64K-byte segment range. Therefore, an offset address can range from 0000H to FFFFH. The **logical address** consists of a segment value and an offset address. The differences among these addresses and the process of converting from one to another are best understood in the context of some examples, as shown next.

Code segment



To execute a program, the 8086 fetches the instructions (Op codes and operands) from the code segment. The logical address of an instruction always consists of a CS (code segment) and an IP (instruction pointer), shown in CS:IP format. The physical address for the location of the instruction is generated by shifting the CS left one hex digit and then adding it to the IP. IP contains the offset address. The resulting 20-bit address is called the physical address since it is put on the external physical address bus pins to be decoded by the memory decoding circuitry.

To clarify this important concept, assume values in CS and IP as shown in the diagram. The offset address is contained in IP; in this case it is 95F3H. The logical address is CS:IP, or 2500:95F3H. The physical address will be 25000 + 95F3 = 2E5F3H. The physical address of an instruction can be calculated as follows:

1. Start with CS. 2 5 0 0
2. Shift left CS. 2 5 0 0 0
3. Add IP. 2 E 5 F 3

The microprocessor will retrieve the instruction from memory locations starting at 2E5F3. Since IP can have a minimum value of0000H and a maximum of FFFFH, the logical address range in this example is 2500:0000 to 2500: FFFF. This means that the lowest memory location of the code segment above will be 25000H (25000 + 0000) and the highest memory location will be 34FFFH (25000 + FFFF).

What happens if the desired instructions are located beyond these two limits? The answer is that the value of CS must be changed to access those instructions. See Example 1-1.

Example 1-1

If CS = 24F6H and IP = 634AH, show:
(a) The logical address
(b) The offset address
and calculate:
(c) The physical address
(d) The lower range
(e) The upper range of the code segment

Solution:
(a) 24F6:634A
(b) 634A
(c) 2B2AA (24F60 + 634A)
(d) 24F60 (24F60 + 0000)
(e) 34F5F (24F60 + FFFF)

Logical vs. physical address in the code segment

In the code segment, CS and IP hold the logical address of the instructions to be executed. The following Assembly language instructions have been assembled (translated into machine code) and stored in memory. The three columns show the logical address of CS:IP the machine code stored at that address and the corresponding Assembly language code. This information can easily be generated by the DEBUG program using the Unassemble command.

Logical address CS:IP	Machine language opcode and operand	Assembly language mnemonics and operand
1132:0100	B057	MOV AL,57
1132:0102	B686	MOV DH,86
1132:0104	B272	MOV DL,72
1132:0106	89D1	MOV CX,DX
1132:0108	88C7	MOV BH,AL
1132:010A	B39F	MOV BL,9F
1132:010C	B420	MOV AH,20
1132:010E	01D0	ADD AX,DX
1132:0110	01D9	ADD CX,BX
1132:0112	05351F	ADD AX,1F35

The program above shows that the byte at address 1132:0 100 contains BO, which is the opcode for moving a value into register AL, and address 1132:0101 contains the operand (in this case 57) to be moved to AL. Therefore, the instruction "MOV