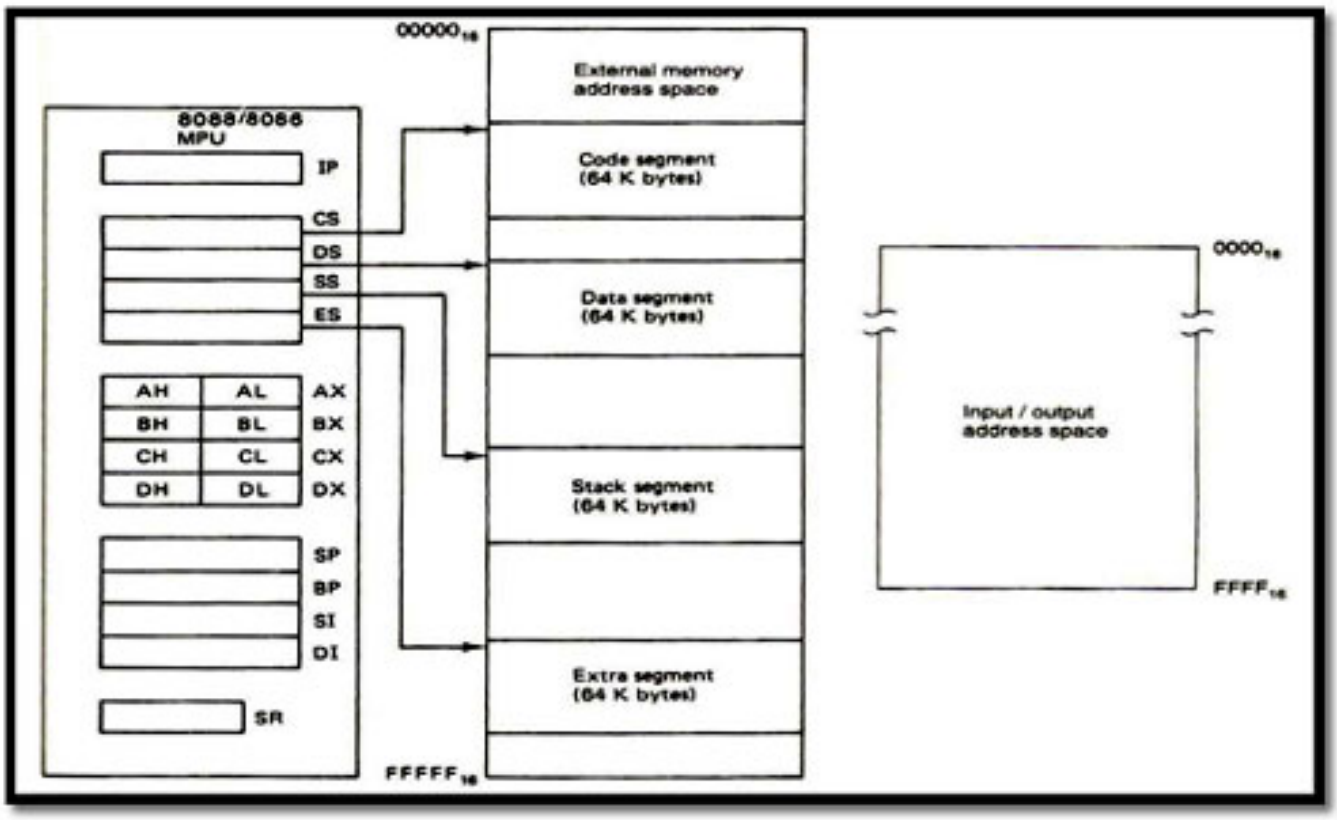
Software Model of the 8086 Microprocessor

As a programmer of the 8086 you must become familiar with the various registers in the EU and BIU. The 8086 microprocessor has a total of fourteen registers that are accessible to the programmer. It is divided into four groups. They are:

- Four General purpose registers
- Four Index/Pointer registers
- Four Segment registers
- Two other register

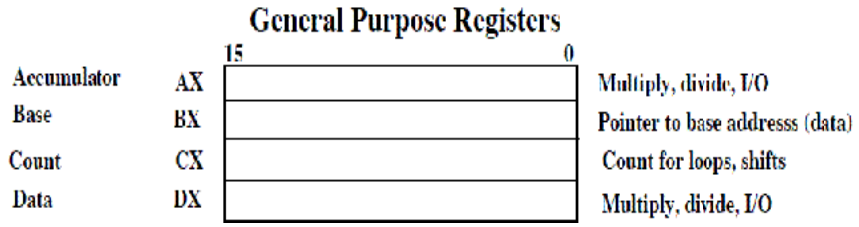


software model of 8086 microprocessor

General purpose registers:

Accumulator register consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX. AL in this case contains the loworder byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations and string manipulation.

Base register consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX. BL in this case contains the low-order byte of the word, and BH contains the high-order byte. BX register usually contains a data pointer used for based, based indexed or register indirect addressing.



Count register consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX. When combined, CL register contains the loworder byte of the word, and CH contains the high-order byte. Count register can be used in Loop, shift/rotate instructions and as a counter in string manipulation

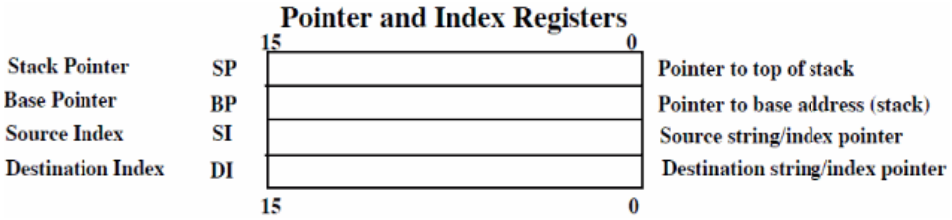
Data register consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low order byte of the word, and DH contains the high-order byte. Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

Index or Pointer Registers

These registers can also be called as Special Purpose registers.

Source Index (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions. Used in conjunction with the DS register to point to data locations in the data segment.

Destination Index (DI) is a 16-bit register. Used in conjunction with the ES register in string operations. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions. In short, Destination Index and SI Source Index registers are used to hold address.



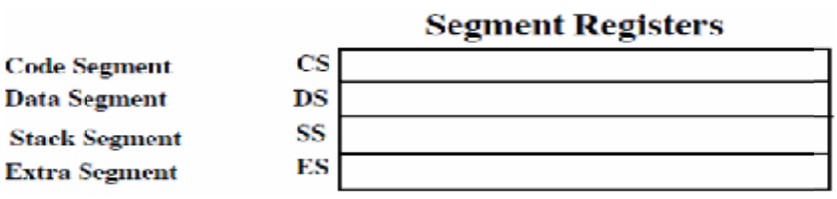
Stack Pointer (SP) is a 16-bit register pointing to program stack, ie it is used to hold the address of the top of stack. The stack is maintained as a LIFO with its bottom at the start of the stack segment (specified by the SS segment register).Unlike the SP register, the BP can be used to specify the offset of other program segments.

Base Pointer (BP) is a 16-bit register pointing to data in stack segment. It is usually used by subroutines to locate variables that were passed on the stack by a calling program. BP register is usually used for based, based indexed or register indirect addressing.

Segment Registers

Most of the registers contain data/instruction offsets within 64 KB memory segment. There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses four segment registers.

Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.



Stack segment (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

Data segment (DS) is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions.

Extra segment (ES) used to hold the starting address of Extra segment. Extra segment is provided for programs that need to access a second data segment. Segment registers cannot be used in arithmetic operations.

Other registers of 8086

Instruction Pointer (IP) is a 16-bit register. This is a crucially important register which is used to control which instruction the CPU executes. The ip, or program counter, is used to store the memory location of the next instruction to be executed. The CPU checks the program counter to ascertain which instruction to carry out next. It then updates the program counter to point to the next instruction. Thus the program counter will always point to the next instruction to be executed.

Flag Register determines the current state of the processor. They are modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program. 8086 has 9 flags and they are divided into two categories:

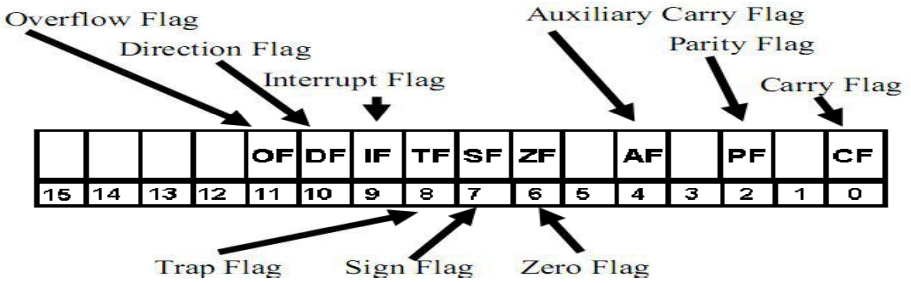
1. Status Flags

Status Flags represent result of last arithmetic or logical instruction executed. Conditional flags are as follows:

- **Carry Flag (CF):** This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.
- **Auxiliary Flag (AF):** If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 D3) to upper nibble (i.e. D4 – D7), the AF flag is set i.e. carry given by D3 bit to

D4 is AF flag. This is not a general-purpose flag, it is used internally by the processor to perform Binary to BCD conversion.

- **Parity Flag (PF):** This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1‟s, the Parity Flag is set and for odd number of 1‟s, the Parity Flag is reset.
- **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.
- **Sign Flag (SF):** In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.
- **Overflow Flag (OF):** It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine.



2. Control Flags

Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

1. **Trap Flag (TP):**
   - It is used for single step control.
   - It allows user to execute one instruction of a program at a time for debugging.
   - When trap flag is set, program can be run in single step mode.
2. **Interrupt Flag (IF):**
   - It is an interrupt enable/disable flag.
   - If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.
   - It can be set by executing instruction sit and can be cleared by executing CLI instruction.

3. Direction Flag (DF):

- It is used in string operation.
- If it is set, string bytes are accessed from higher memory address to lower memory address.
- When it is reset, the string bytes are accessed from lower memory address to higher memory address.

MEMORY SEGMENTATION

The 8086 microprocessor operate in the Real mode memory addressing. Real mode operation allows the microprocessor to address only the first 1M byte of memory space. The first 1M byte of memory is called either the **real memory** or **conventional memory** system. Even though the 8086 has a 1M byte address space, not all this memory is active at one time. Actually, the 1M bytes of memory are partitioned into 64K byte (65,536) segments. The 8086-80286 microprocessors allow four memory segments. Figure 2-3 shows these memory segments.

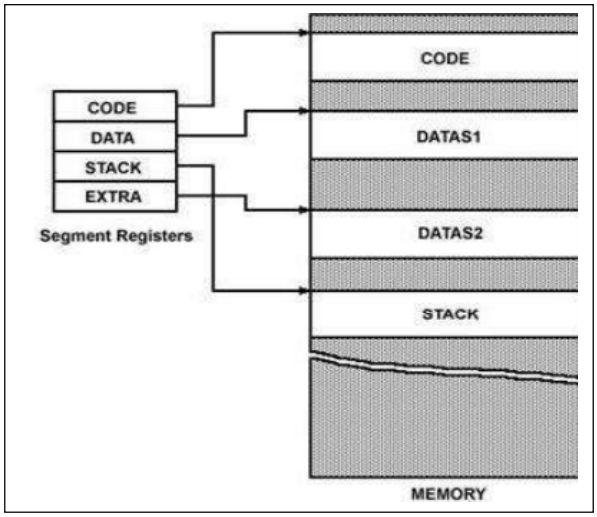
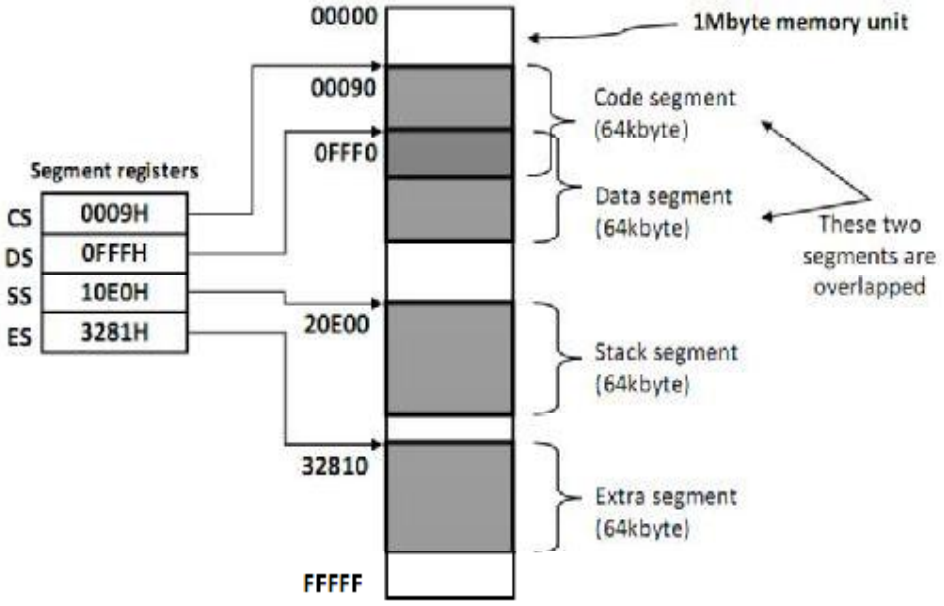


Fig. (2.3): Real Mode, Segmented Memory Model.

Think of segments as windows that can be moved over any area of memory to access data or code. Also note that a program can have more than four segments, but can only access four segments at a time.

Example: Let the segment registers be assigned as follow: CS = 0009H, DS = 0FFFH, SS = 10E0, and ES = 3281H. We note here that code segment and data segment are overlapped while other segments are disjointed.



In the real mode a combinational of a segment address and offset address access a memory location. All real mode memory address must consist of a segment address plus an offset address. The microprocessor has a set of rules that apply to segments whenever memory is addressed. These rules define the segment register and offset register combination (see Table 2-1). For example, the code segment register is always used with the instruction pointer to address the next instruction in a program. This combination is CS:IP. The code segment register defines the start of the code segment and the instruction pointer locates the next instruction within the code segment

TABLE (2-1): 8086 default 16 bit segment and offset address combinations

Segment	Offset	Special Purpose
CS	IP	Instruction address
SS	BP	Stack address
SS	SP	Top of the stack
DS	BX, DI,SI, an 8-bit number, or a 16-bit number	Data address
ES	DI for string instructions	String destination address

This combination (CS:IP) locates the next instruction executed by the microprocessor. For example if CS = 1400H and IP = 1200H, the microprocessor fetches its next instruction from memory location:

Physical address=Segment base address*10+Offset (Effective) address

PA = SBA * 10+ EA

=1400H*10+1200H=15200H.

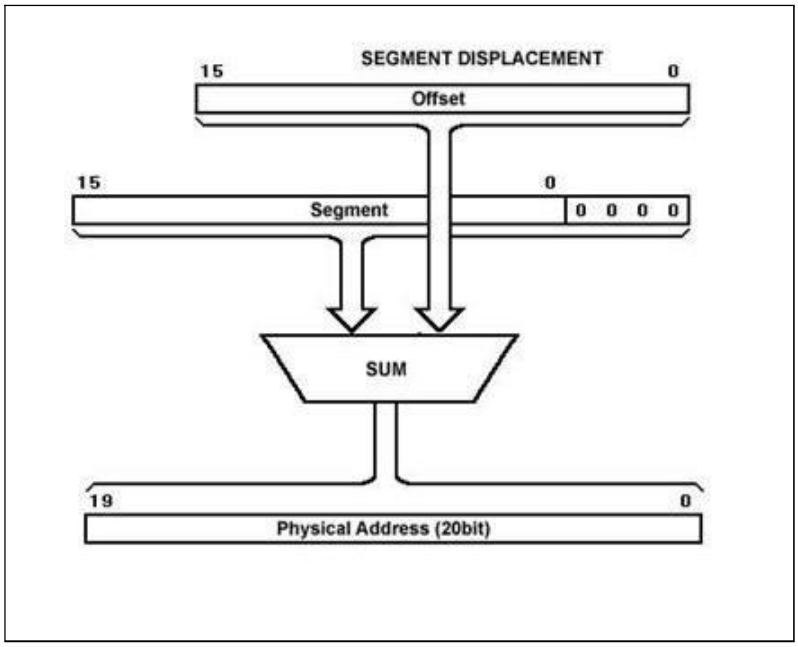


Fig. (2.3): Generating a physical address

Q: What is segmentation? What are its advantages? How is segmentation implemented in typical microprocessors?

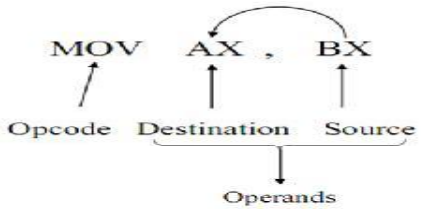
Ans:

Segment memory addressing divides the memory into many segments. Each of these segments can be considered as a linear memory space. Each of these segment is addressed by a segment register. However since the segment register is 16 bit wide and the memory needs 20 bits for an address the 8086 appends four bits segment register to obtain the segment address. Therefore, to address the segment 10000H by , say the SS register, the SS must contain 1000H. The first advantage that memory segmentation has is that only 16 bit registers are required both to store segment base address as well as offset address. This makes the internal circuitry easier to build as it removes the requirement for 20 bits register in case the linear addressing method is used. The second advantage is relocatability.

8086 Addressing Mode

Introduction

- Program is a sequence of commands used to tell a microcomputer what to do.
- Each command in a program is an instruction
- Programs must always be coded in machine language before they can be executed by the microprocessor.
- A program written in machine language is often referred to as *machine code*.
- Machine code is encoded using **0**s and **1**s
- A single machine language instruction can take up one or more bytes of code
- In assembly language, each instruction is described with alphanumeric symbols instead of with **0**s and **1**s
- Instruction can be divided into two parts : its *opcode* and *operands*
- Op-code identify the operation that is to be performed.
- Each opcode is assigned a unique letter combination called a *mnemonic*.
- Operands describe the data that are to be processed as the microprocessor carried out, the operation specified by the opcode.
- For example, the move instruction is one of the instructions in the data transfer group of the 8086 instruction set.
- Execution of this instruction transfers a byte or a word of data from a source location to a destination location.



Addressing Mode of 8086

An addressing mode is a method of specifying an operand. The 8086 addressing modes categorized into three types:

1. Register Addressing
2. Immediate Addressing
3. Memory Addressing

Register addressing mode

In this addressing mode, the operands may be:

- reg16: 16-bit general registers: AX, BX, CX, DX, SI, DI, SP or BP.
- reg8 : 8-bit general registers: AH, BH, CH, DH, AL, BL, CL, or DL.