Software Architecture of the 8086 Microprocessor

2-1 MICROARCHITECTURE OF THE 8086 MICROPROCESSOR:

The microarchitecture of a microprocessor is its internal architecture that is, the circuit building blocks that implement the software and hardware architectures of the 8086 microprocessors.

The microarchitecture of the 8086 microprocessors employs *parallel processing-that* is, they are implemented with several simultaneously operating processing units. Figure 2-1(a) illustrates the internal architecture of the 8086 microprocessor. They contain two processing units: the **B**us **I**nterface **U**nit (**BID**) and the **E**xecution **U**nit (**EU**). Each unit has dedicated functions and both operate at the same time. In essence, this parallel processing effectively makes the fetch and execution of instructions independent operations. This results in efficient use of the system bus and higher performance for 8086 microcomputer systems.

The BID: It is the 8086's connection to the outside world. By interface, we mean the path by which it connects to external devices. The BID is responsible for performing all external bus operations, such as instruction fetching, reading and writing of data operands for memory, and inputting or outputting data for input/output peripherals. These information transfers take place over the system bus. The BID is not only responsible for performing bus operations; it also performs other functions related to instruction and data obtained. For instance, it is responsible for instruction queuing and address generation.

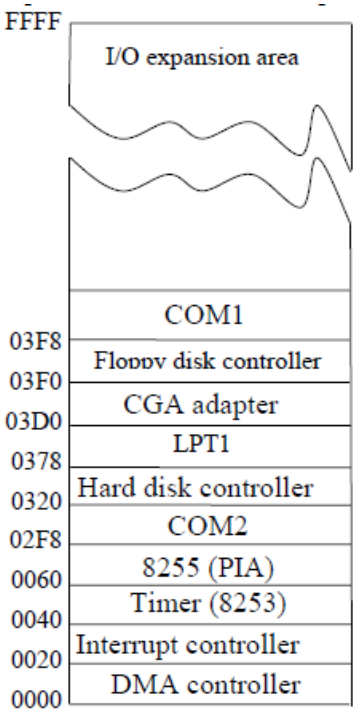


Figure (1-5) The I/O map of a personal computer.

As shown in Figure 2-1(b) the BID contains the segment registers, the instruction pointer, the address generation adder, bus control logic, and an instruction queue.

The BID uses a mechanism known as an ***instruction queue*** to implement a pipelined architecture. This queue permits the 8086 to prefetch up to 6 bytes of instruction code. The BID is free to read the next instruction code when:

- The queue is not full-that is, it has room for at least 2 more bytes.
- The execution unit is not asking it to read or write data from memory.

(b) Execution and interface units.

Pre-fetched instructions are held in the first-in first-out (FIFO) queue. Since instructions are normally waiting in the queue, the time needed to fetch many instructions of the microcomputer's program is eliminated. If the queue is full and the ED is not requesting access to data in memory, the BID does not need to perform any bus operations. These intervals of no bus activity, which occur between bus operations, are known as **idle states.**

The EU: The execution unit is responsible for decoding and executing instructions. Notice in Fig. 2-1 (b) that it consists of the *arithmetic logic unit* (ALU), status and control flags, general-purpose registers, and temporary-operand registers. The EU accesses instructions from the output end of the instruction queue and data from the general-purpose registers or memory. It reads one instruction byte after the other from the output of the queue, decodes them, generates data addresses if necessary, passes them to the BIU and requests it to perform the read or write operations to memory or I/O, and performs the operation specified by the instruction. The ALU performs the arithmetic, logic, and shift operations required by an instruction. During execution of the instruction, the EU may test the status and control flags, and updates these flags based on the results of executing the instruction. If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to the top of the queue.

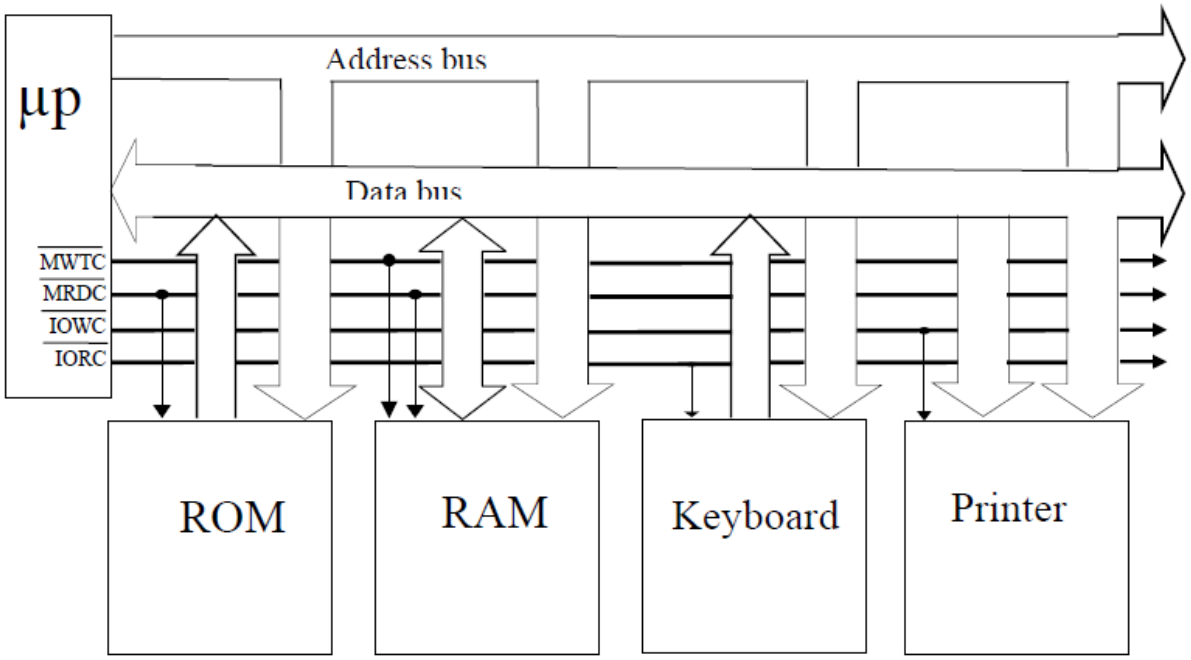


Figure (1-6) The block diagram of computer system showing the buses structure.

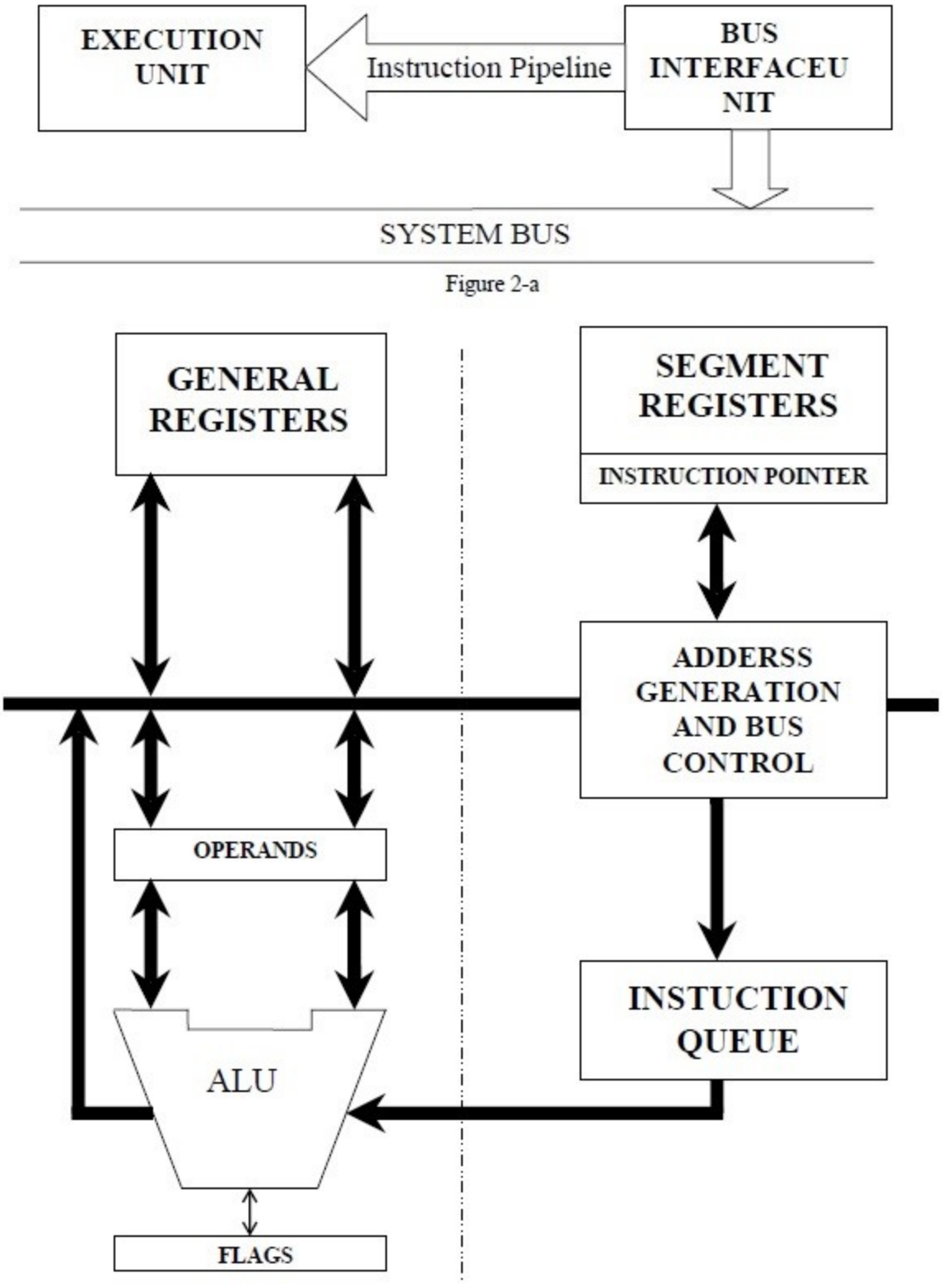


Fig 2-1 (a) Pipelined architecture of 8086 microprocessor

2.2 SOFTWARE MODEL OF THE 8086 MICROPROCESSOR:

To be able to program a microprocessor, one does not need to know all of its hardware architectural features. What is important to the programmer is to know the various registers within the device and to understand their purpose, functions, operating capabilities, and limitations. Fig. 2-2 illustrates the software architecture of the 8086 microprocessor. From this diagram, we see that it includes fourteen l6-bit internal registers: the *instruction pointer* (IP), four *data registers* (AX, BX, CX, and DX), two *pointer registers* (BP and SP), two *index registers* (SI and DI), four *segment registers* (CS, DS, SS, and ES) and *status register* (SR), with nine of its bits implemented as status and control flags.

• **INSTRUCTION POINTER (IP):** IP is 16 bits in length and identifies the next word of instruction code to be fetched from the current code segment however; it contains the offset of the instruction code instead of its actual address. This is because IP and CS are both 16 in length, but a 20-bit address is needed to access memory. The vale of the next address for the next code access is often denoted as CS: IP.

Actual address=CS+IP.

• **DATA REGISTER:** Fig 2-3 shows the 8086 has four **general-purpose data registers.** Notice that they are referred to as the accumulator register (A), the base register (B), the count register (C), and the data register (D). Any of the general-purpose data registers can be used as the source or destination of an operand during an arithmetic operation such as ADD or a logic operation such as AND.

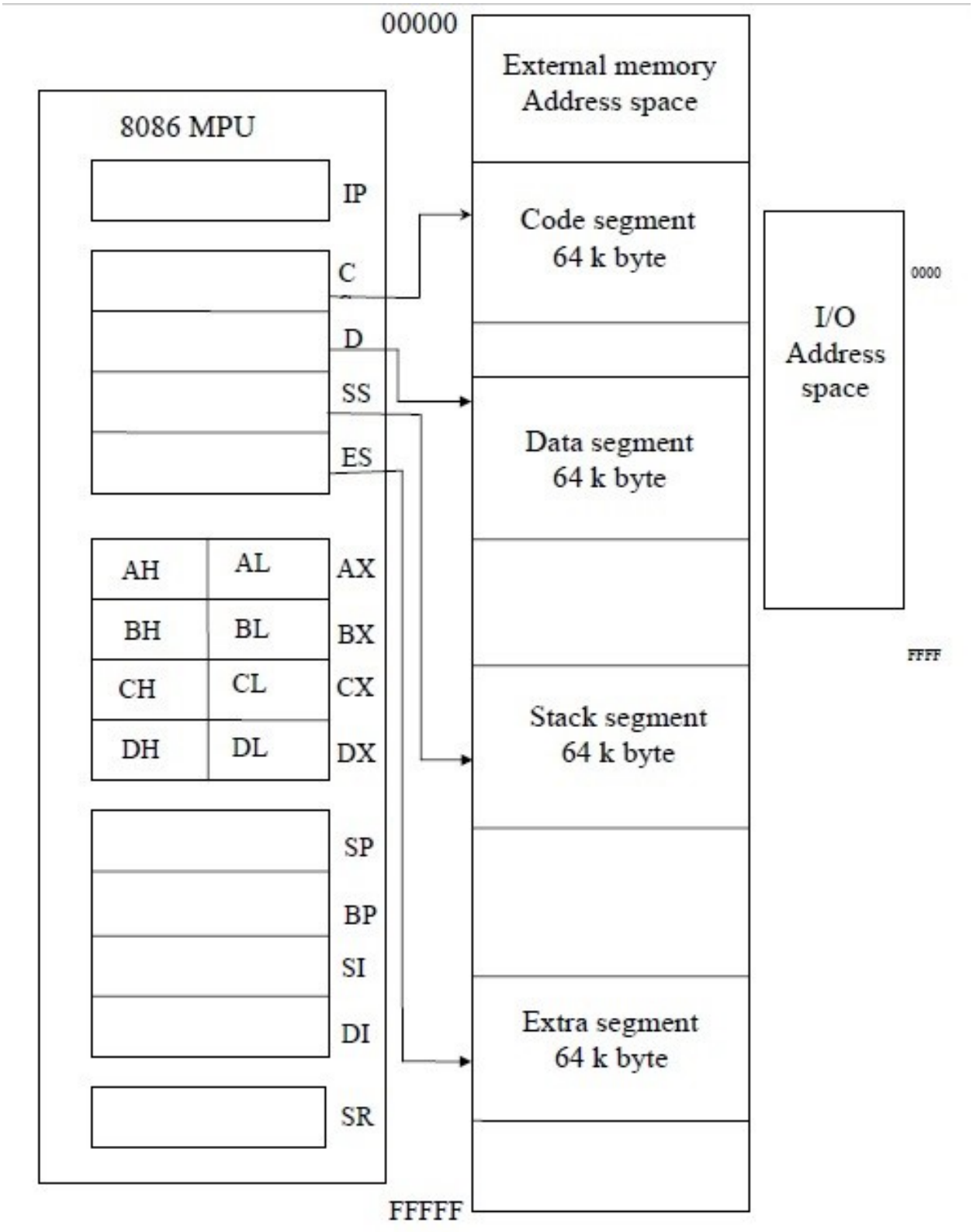
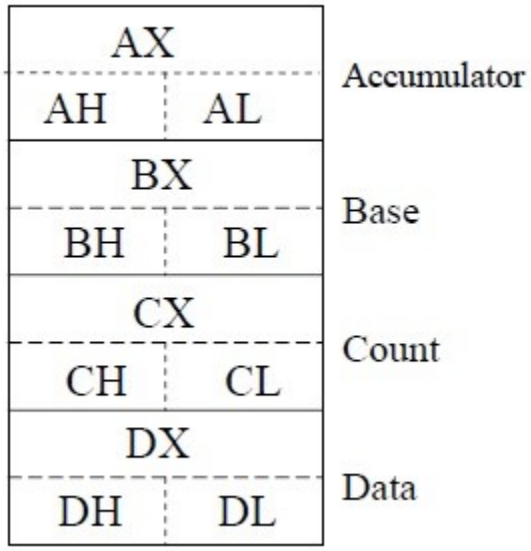


Figure 2-2 Software model of 8086 microprocessor.



(a)

Figure 2-3(a) General-purpose data registers.

Operation	Word
AX	multiply, word divide, word I/O
AL	Byte multiply, byte divide, byte I/O, translate, decimal arithmetic.
AH	Byte multiply, byte divide.
BX	Translate
CX	String operation, loops.
CL	Variable shift and rotate.
DX	Word multiply, word divide, indirect I/O.

(b)

Figure 2-3(b) Dedicated register functions.

- **POINTER AND INDEX REGISTERS**: Figure 2-4 shows these register. These registers are four general-purpose registers: two pointer registers and two index registers. They store what are called **offset addresses**. An **offset address** selects any location within 64 k byte memory segment.

SP	Stack pointer
BP	Base pointer
SI	Source index
DI	Destination index

Figure 2-4 pointer and index registers.

- **SEGMENT REGISTERS AND 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-even if it is the Pentium 4 microprocessor. Note that 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 a. Figure 2-5 shows these memory segments. Note that a memory segment can touch or even overlap if 64K bytes of memory are not required for a segment. 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. 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.

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

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

The code segment register defines the start of the code segment and the instruction pointer locates the next instruction within the code segment. This combination (CS:IP) locates the next instruction executed by the microprocessor. Figure 2-6 show an example that 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.

Table 2-2 shows more examples of memory addresses.

Segment register	Offset address	Physical address
002A	0023	002C3
4900	0A23	49A23
1820	FE00	28000

Table 2-2

FLAG (STATUS) REGISTER: This register is another 16-bit register within the 8086. Figure (2-7) shows the organization of this register. Notice that just nine of its bits are implemented. Six of these bits represent status flags:

- ❖ The carry flag (CF): CF is set if there is a carry-out or a borrow-in for the most significant bit of the result during the execution of an instruction. Otherwise, CF is reset.
- ❖ The parity flag (PF): PF is set if the result produced by the instruction has even parity-that is, if it contains an even number of bits at the 1 logic level. If parity is odd, PF is reset.
- ❖ The auxiliary carry flag (AF): AP is set if there is a carry-out from the low nibble into the high nibble or a borrow-in from the high nibble into the low nibble of the lower byte in a l6-bit word. Otherwise, AF is reset.
- ❖ The zero flag (ZF): ZF is set if the result produced by an instruction is zero. Otherwise, ZF is reset.
- ❖ The sign flag (SF): The MSB of the result is copied into SF. Thus, SF is set if the result is a negative number or reset if it is positive.
- ❖ The overflow flag (OF): When OF is set, it indicates that the signed result is out of range. If the result is not out of range, OF remains reset.

- The other three flags are control flags. These three flags provide control functions of the 8086 as follows:

- ❖ The trap flag (TF): If TF is set, the 8086 goes into the single-step mode of operation.
- ❖ The interrupt flag (IF): The IF controls the operation of the INTR (interrupt request) input pin If IF=1, the INTR pin is enabled; If IF=0, the INTR pin is disabled. The state of IF bit is controlled by the STI **(set IF)** and CLI **(clear IF)** instructions.
- ❖ The direction flag (DF): The direction flag select either the increment or decrement mode for the DI and/or SI registers during string instructions. If D=1, the registers are automatically decremented; if D=0, these registers are automatically incremented. The DF is set with STD **(set direction))** and cleared with CLD **(clear direction)** instructions.

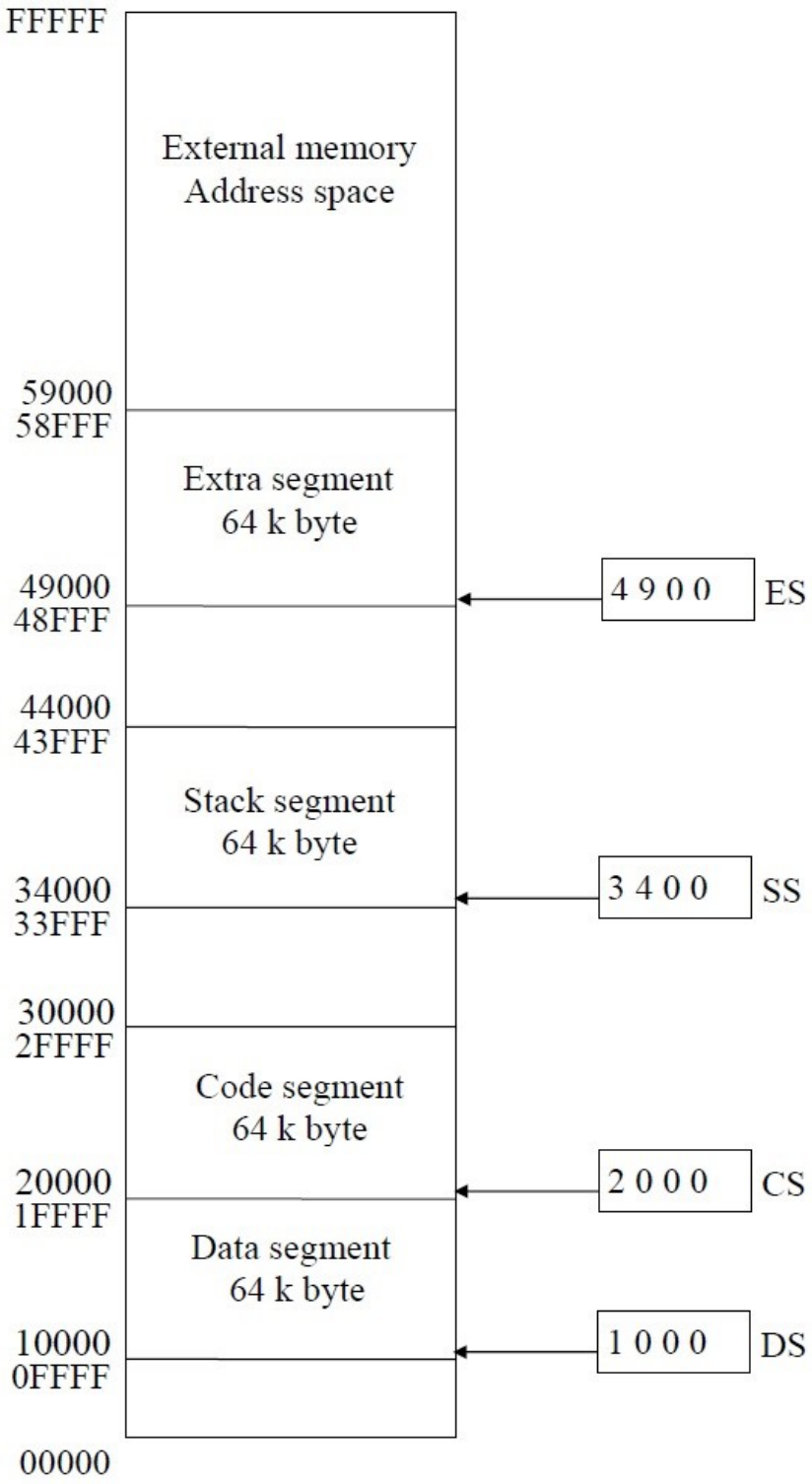


Figure 2-5 A memory system showing the placement of four memory segments.

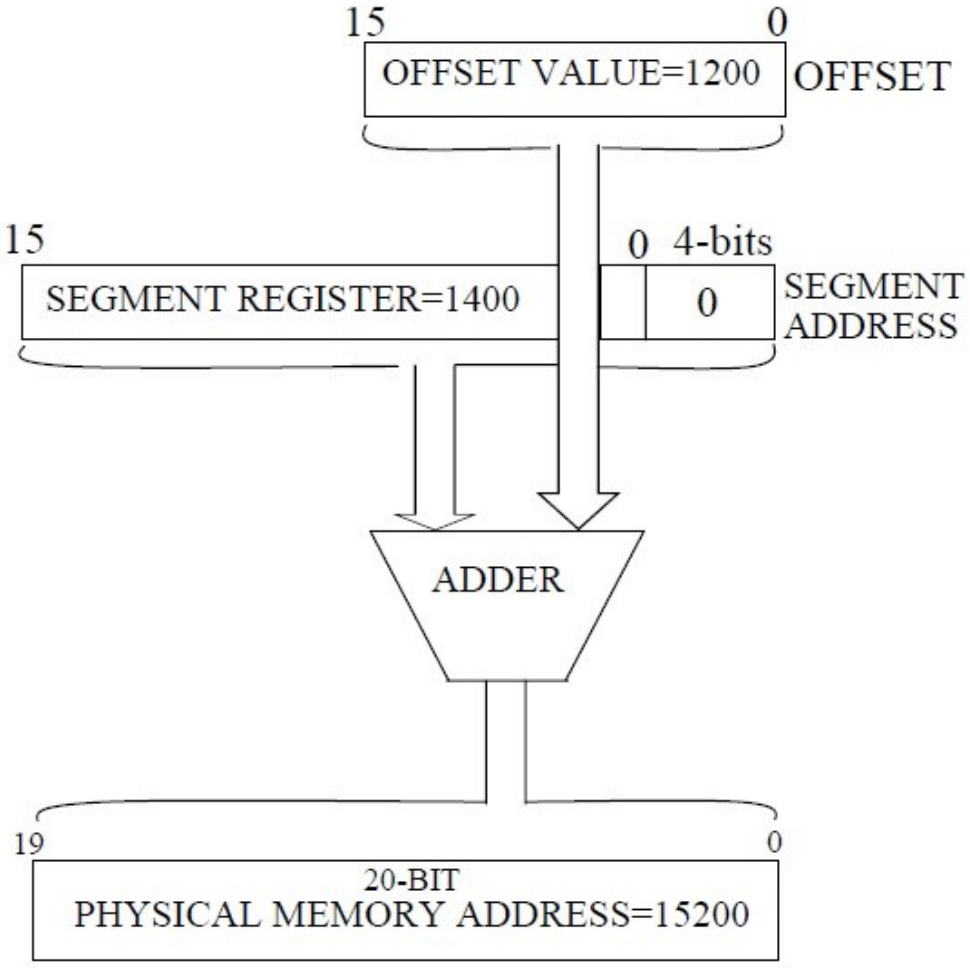


Figure 2-6 generating a physical address

				O	D	I	T	S	Z		A		P		C

Figure 2-7 Status and control flags.