Microprocessors

Unit – 2: Chapter – 2

8086 Architecture and Instruction Set

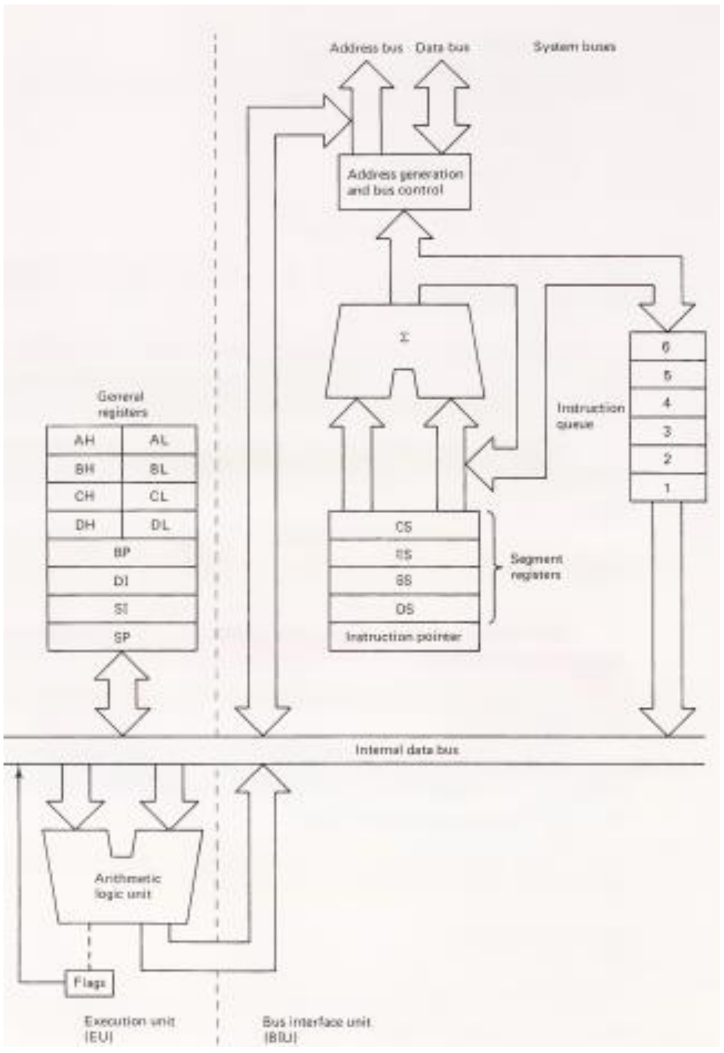
When studying microprocessors there are two major topics to consider:

1. The CPU architecture, including the internal registers, flags and the instruction set.
2. The electrical interface, including the data bus, address bus, and control buses; and the clock generation circuitry.

1. The 8086 CPU Architecture

The microprocessor functions as the CPU in the stored program model of the digital computer. Its job is to generate all system timing signals and synchronize the transfer of data between memory, I/O, and itself. It accomplishes this task via the three-bus system architecture. The microprocessor also has a software function. It must recognize, decode, and execute program instructions fetched from the memory unit. The software function of MPU requires an arithmetic-logic unit (ALU) within the CPU to perform arithmetic and logical functions.

The following figure shows a model of the 8086 CPU.



The 8086 is organized as two separate processors:

a) the bus interface unit (BIU)

b) the execution unit (EU)

The BIU provides hardware functions, including generation of the memory and I/O addresses for transferring the data to and from the CPU. The EU receives program instructions and data from the BIU, executes these instructions, and stores the results in the general registers. By passing the data back to the BIU, data can also be stored in a memory location or written to an output device.

The differences between an 8086 microprocessor and an 8088 microprocessor:

1. In the 8088, the BIU data bus path is 8 bits wide Vs., the 8086's 16-bit data bus.

2. The 8088 instruction queue is four bytes long, whereas in 8086 it is six bytes long.

Fetch and Execute

1. The BIU outputs the contents of the instruction pointer register (IP) onto the address bus, causing the selected byte or word to be read into the BIU.
2. Register IP is incremented by 1 to prepare for the next instruction fetch.
3. Once inside the BIU, the instruction is passed to the queue. This is a first-in, first-out storage register.
4. Assuming that the queue is initially empty, the EU immediately draws the instruction from the queue and begins execution.
5. While the EU is executing this instruction, the BIU proceeds to fetch a new instruction. Depending on the execution time of the first instruction, the BIU may fill the queue with several new instructions before the EU is ready to draw its next instruction.

Programming Model

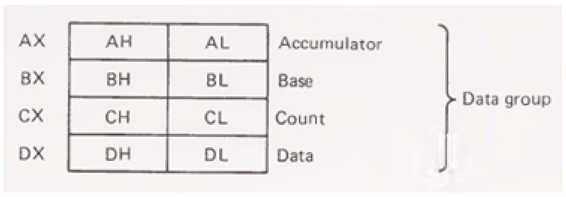
A programmer for the 8086/8088 must become familiar with the various registers in the EU and BIU.

Data Group

The data group consists of the accumulator and the BX, CX, and DX registers.

Each can be accessed as a byte or a word. Thus BX refers to the 16-bit base register but BH refers only to the high-order 8 bits of this register and BL refers to the low-order 8 bits of this register.

The data registers are normally used for storing temporary results that are needed for instruction execution.



Pointer and Index Group

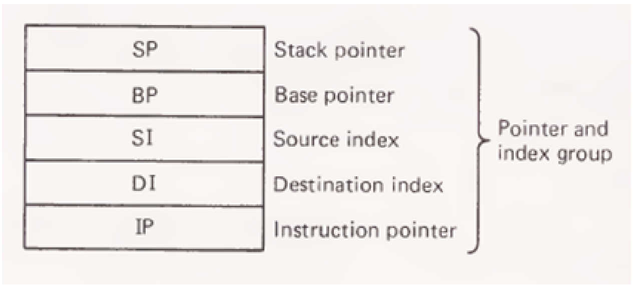
The pointer and index group are all 16-bit registers.

These registers are used as memory pointers.

For example, the instruction MOV AH, [SI] means "Move the byte whose address is contained in register SI to register AH."

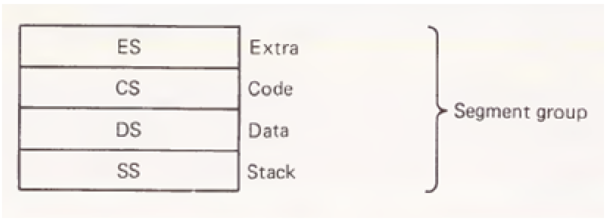
SI thus "points" at the desired memory location.

The brackets around SI are used to indicate the contents of memory pointed to by SI, not the value of SI itself.



Segment Group

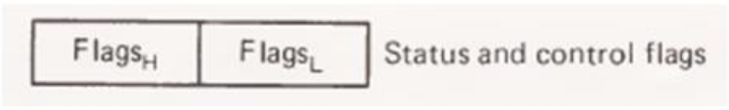
The segment group consists of the registers that are used by the BIU to determine the memory address output by the CPU when it is reading or writing from the memory unit.

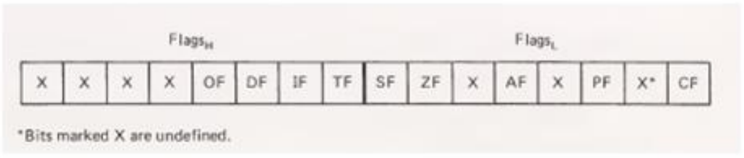


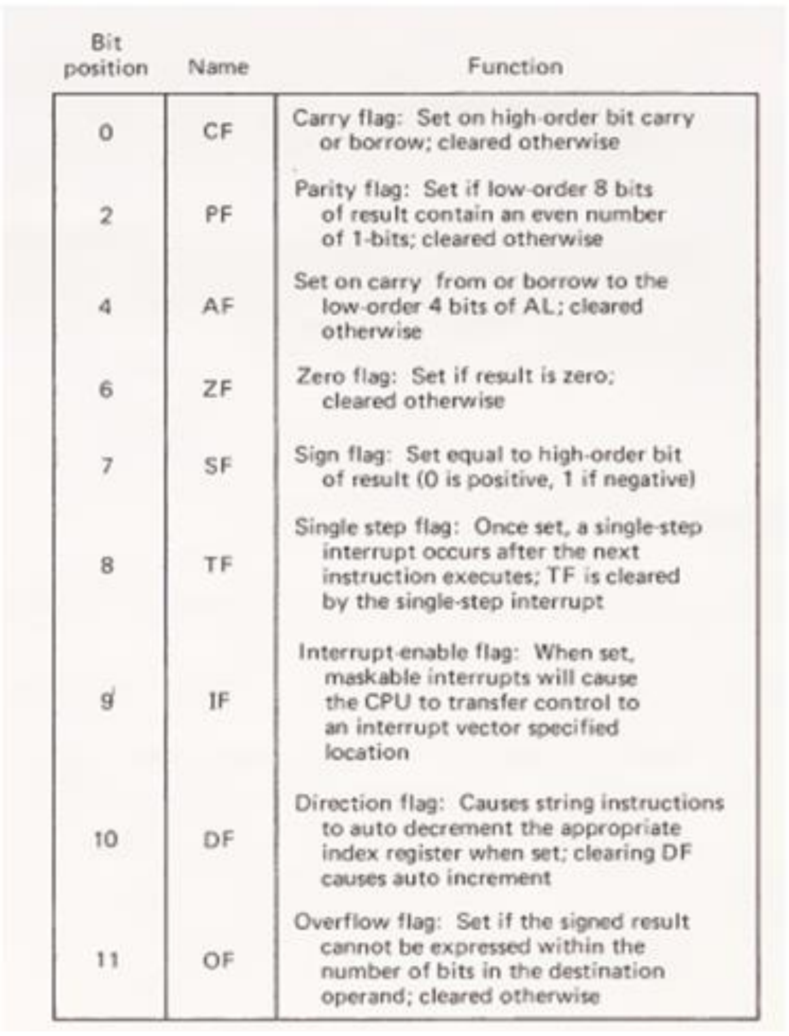
Status and Control Flags

The 8086 programming model consists of a 16-bit flag register.

Six of the flags are status indicators, reflecting properties of the result of the last arithmetic or logical instruction.







Bit position	Name	Function
0	CF	Carry flag: Set on high-order bit carry or borrow; cleared otherwise
2	PF	Parity flag: Set if low-order 8 bits of result contain an even number of 1-bits; cleared otherwise
4	AF	Set on carry from or borrow to the low-order 4 bits of AL; cleared otherwise
6	ZF	Zero flag: Set if result is zero; cleared otherwise
7	SF	Sign flag: Set equal to high-order bit of result (0 is positive, 1 if negative)
8	TF	Single step flag: Once set, a single-step interrupt occurs after the next instruction executes; TF is cleared by the single-step interrupt
9	IF	Interrupt-enable flag: When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location
10	DF	Direction flag: Causes string instructions to auto decrement the appropriate index register when set; clearing DF causes auto increment
11	OF	Overflow flag: Set if the signed result cannot be expressed within the number of bits in the destination operand; cleared otherwise

2. Segmented Memory

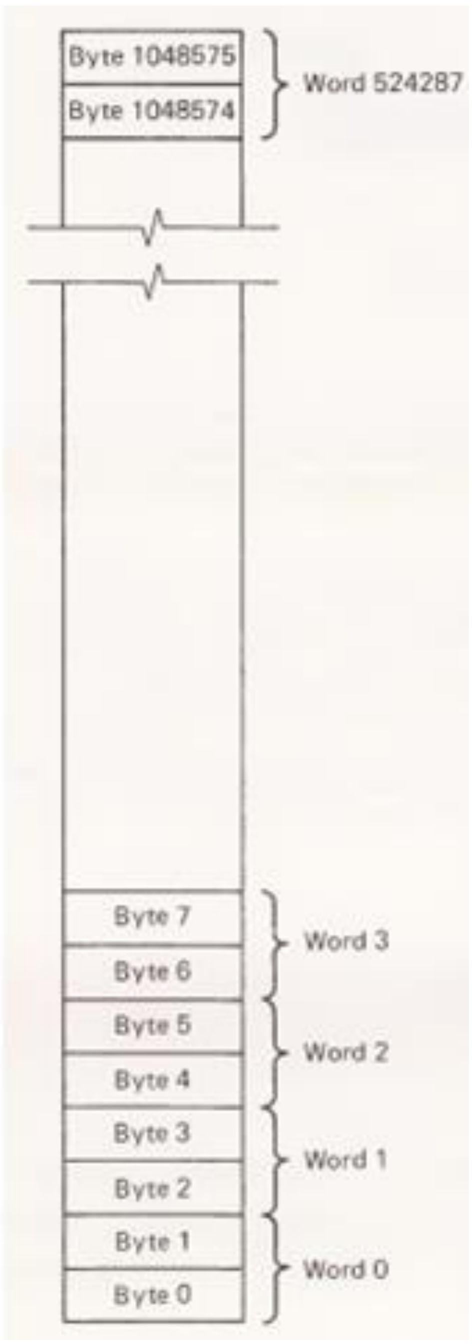
Even though the 8086 is considered a 16-bit microprocessor, its memory is still thought of in bytes.

Actually, there are a couple of good reasons, here.

1. It allows the processor to work on bytes as well as words. 'This is especially important with I/O devices such as printers, terminals, and modems, all of which are designed to transfer ASCII-encoded (7- or 8-bit) data.
2. Many of the 8086's (and 8088's) operation codes are single bytes. Other instructions may require anywhere from two to seven bytes. By being able to access individual bytes, these odd-lengthed instructions can be handled.

The 8086/88 has a 20-bit address bus, allowing it to output 2^{20} (1,048,576) different memory addresses.

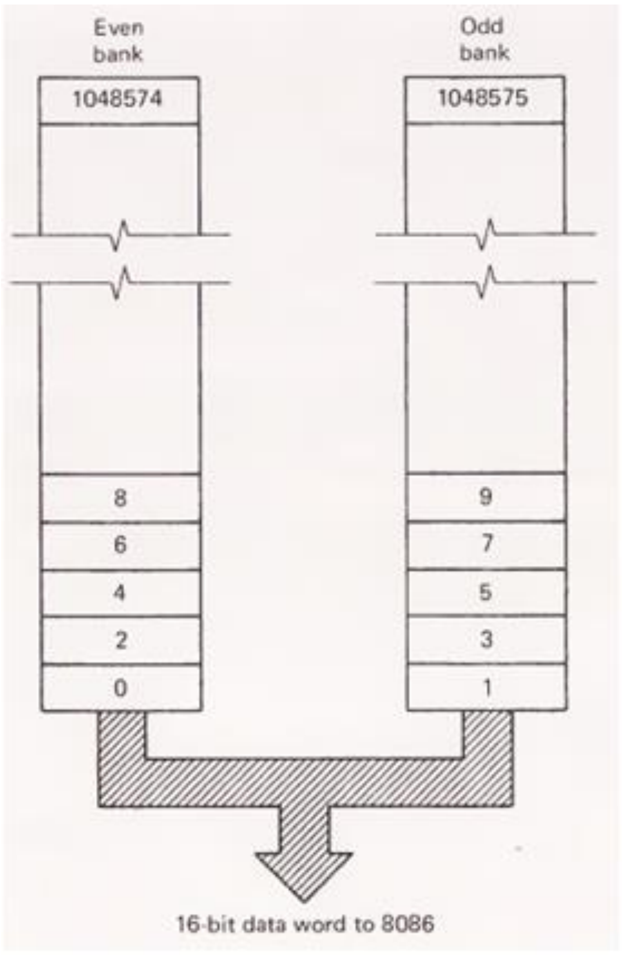
In the figure below, it is shown how 524,287 words can also be visualized in this memory.



The 8086 reads 16 bits from memory by simultaneously reading an odd-addressed byte and an even-addressed byte. For this reason, the 8086 organizes its memory into an even-addressed bank and an odd-addressed bank, as shown in figure below. The 8086 provides control bus signals that can be decoded by the memory to determine if the access is for a byte or a word.

For accessing a tow-byte word, the CPU must perform two memory read cycles:

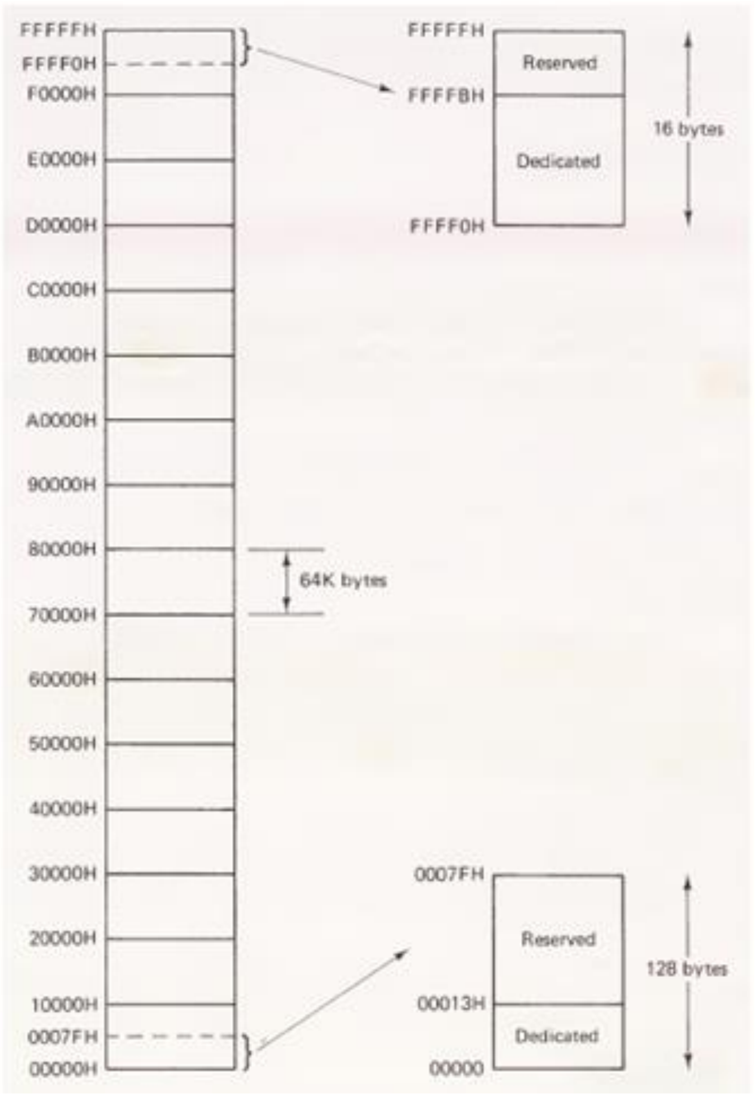
1. one to fetch the low-order byte
2. second to fetch the high-order byte.



Memory Map

The figure below shows the memory map of 8086/88.

1. There are 16 memory blocks with 64K bytes each, beginning at hex address 00000 and ending at address FFFFFH.
2. The figure shows RAM from 00000 to 3FFFFH and ROM from FF000H to FFFFFH.
3. The remainder of the memory space unused.
4. This information is vital to know exactly where programs can be safely loaded.
5. Some memory locations are marked reserved and others dedicated.
6. The ***dedicated locations*** are used for processing specific system interrupts and the reset function.



Segment Registers

Within the 1MB of memory space the 8086/88 defines four 64K-byte memory blocks.

These blocks are, the ***code segment, stack segment, data segment,*** and ***extra segment***.

Each of these memory blocks is used differently by the processor.

Code segment	The code segment holds the program instruction codes.
Stack segment	The stack segment is used to store interrupt and subroutine return addresses.
Data segment	The data segment stores data for the program.
Extra segment	The extra segment is an extra data segment (often used for shared data).

Logical and Physical Addresses

Addresses within a segment can range from address 0 to address FFFFH.

This corresponds to the 64K-byte length of the segment.

An address within a segment is called an ***offset*** or ***logical address***. To get the physical address, the offset should be added to the base address of the segment.

Advantages of Segmented Memory

In 8086, the program op-codes will be fetched from the code segment, while program data variables will be stored in the data and extra segments.

Stack operations use registers BP or SP and the stack segment.

An immediate advantage of having separate data and code segments is that one program can work on several different sets of data. This is done by reloading register DS to point to the new data.

Another advantage of segmented memory is that programs that reference logical addresses only can be loaded and run anywhere in memory. This is because the logical addresses always range from 0000 to FFFFH, independent of the code segment base.

Advantages of Segmented Memory (Contd ...)

Example:

Consider a multitasking environment in which the 8086/88 is doing several different jobs at once.

An inactive program can be temporarily saved on a magnetic disk and a new program brought in to take its place—without concern for the physical location of this new program.

Such programs are said to be relocatable, meaning that they will run at any location in memory.

The requirements for writing relocatable programs are that no references be made to physical addresses, and no changes to the segment registers are allowed.

3. Addressing Modes

A computer instruction is made up of an operation code and zero, one, or two operands.

The op-code identifies the operation to be performed while the operands identify the source and destination of the data operated on.

The operands can specify a CPU register, a memory location in one of the memory segments, or an I/O port.

The different ways in which the microprocessor generates these operand addresses are called the addressing modes.

SNO	Addressing Mode	Purpose
1	Immediate	Source data is within instruction.
2	Register	Source and destination of data are CPU registers.
3	Direct	Memory address is supplied within the instruction.
4	Register indirect	Memory address is supplied in an index or pointer register.
5	Indexed	Memory address is the sum of the index register plus a displacement within the instruction.
6	Based	Memory address is the sum of the BX or BP base registers plus a displacement within instruction.
7	Based and indexed	Memory address is the sum of an index register and a base register.
8	Based and indexed with displacement	Memory address is the sum of an index register, a base register, and a displacement within instruction.
9	Strings	The memory source address is register SI in the data segment, and the memory destination address is register DI in the extra segment.

4. Data Transfer Instructions

There are over 3000 different 8086/88 instructions to consider if each op-code is combined with each addressing mode!

The data transfer group of instructions allows the microprocessor to communicate with the outside world via the input and output instructions.

These instructions also provide the means for moving data into and out of memory and between the CPU registers.

MOV Instruction

The MOV instruction transfers data from source to destination.

The operand part is always written in the form destination, source.

The data moved can be a byte or a word.

The destination and source cannot both be memory locations.

The flags are unaffected by MOV of instructions.

Following are some example combinations with the MOV instruction.

```
MOV AX,[BX]
MOV AX,[BX + 4]
MOV AX,[SI - 6]
MOV AX,[BP + SI]
MOV AX,[BP + SI + 3EH]
```

These are all memory-to-register instructions but use the various forms of the memory indirect addressing modes.

Special Data Transfer Instructions

There are some data transfer instructions that do not use the MOV mnemonic.

The instructions are listed below

Mnemonic	Operands	Description
XCHG	Dest., source	Switch the contents od the word or byte source operand with the destination operand. No flag is affected.
LAHF		Copy the low-order flag byte into AH.
SAHF		Copy AH into the low-order flag byte.
IN	AC, port	Input a byte or a word from direct I/O ports 0-255.
OUT	Port, AC	Output a byte or word to direct I/O ports 0-255.
LEA	dest., source	The effective address of the source operand is transferred to the destination operand. No flags are affected.
LDS LES	dest., source dest., source	Transfer a 32-bit pointer variable from the source operand in memory to the destination register DS or ES. No flag is affected.
XLAT		Replace the byte in AL with a byte from the 256-byte table beginning at [BX]. Use AL as an offset into the table. No flag is affected.

5. String Instructions

The 8086/88 has several instructions for moving large blocks of data or strings.

The following table lists the string instructions.

For all these instructions the memory source is DS:SI and the memory destination is ES:DI.

The destination must be the extra segment.

Mnemonic	Description
STOSB STOSW STOS	Transfers a string byte, word or stream from register AL or AX to the string element addressed by DI in the extra segment.
LODSB LODSW LODS	Transfers a byte, word or a stream from the string element addressed by SI in the data segment to the register AL or AX.
MOVSB MOVSW MOVS	Transfer a byte, word or stream from a string element addressed by SI in the data segment to the string element addressed by DI in the extra segment.
SCASB SCASW SCAS	Subtract the byte, word or stream of the string element addressed by DI in the extra segment from AL or AX.
CMPSB CMPSW CMPS	Subtract the byte, word or string of the destination string element addressed by DI in the extra segment from the byte, word or stream of the source string element addressed by SI in the data segment.

6. Logical Instructions

The logical instructions refer to the Boolean logic functions, such as AND, OR, NOT, exclusive-OR, and to the rotate and shift instructions.

These instructions are all performed in the ALU and usually affect all the flags.

Boolean Functions

Mnemonic	Description
NOT	Complement all bits of the byte or word operand.
AND	Perform a bit-by-bit AND of the source and destination byte or word operands, storing the result in the destination operand.
OR	Perform a bit-by-bit OR of the source and destination byte or word operands, storing the result in the destination operand.
XOR	Perform a bit-by-bit exclusive-OR (XOR) of the source and destination byte or word operands, storing the result in the destination operand.
TEST	Perform a bit-by-bit AND of the source and destination byte or word operands; the operands remain unchanged.

Shift and Rotate Instructions

The main difference between a shift and a rotate is that the shifted bits "***fall off***" the end of the register, whereas the rotated bits "***wrap around***."

Within the shift group of instructions there are both arithmetic (SAL and SAR) and logical (SHL and SHR) shift instructions.

Mnemonic	Description
SAL/SAR SHL/SHR	Shift word or byte operand left or right once. Note: SAL/SAR are arithmetic shift instructions. SHL/SHR are logical shift instructions.
RCL RCR ROL ROR	Rotate word or byte operand left or right once.

7. Arithmetic Instructions

The arithmetic instructions give the microprocessor its computational capabilities.

But unlike earlier 8-bit processors, these instructions are not limited to addition and subtraction of 8-bit numbers in the accumulator.

The 8086/88 can add and subtract 8-bit and 16-bit numbers in any of the general CPU registers, and using certain dedicated registers, perform multiplication and division of signed or unsigned numbers.

Addition and Subtraction Instructions

Mnemonic	Operands	Description
ADD	Dest., source	Replace the destination byte or word with the sum of source and destination operands. Update all flags.
ADC	Dest., source	Replace the destination byte or word with the sum of source and destination operands plus the carry. Update all flags.
SUB	Dest., source	Replace the destination byte or word with the difference between the destination operand and the source operand. Update all flags.
SBB	Dest., source	Replace the destination byte or word with the difference between the destination operand and the source operand plus carry. Update all flags.
INC	Dest.	Increment the byte or word destination operand by one and store the result in the destination operand. All flags except CF are updated.
DEC	Dest.	Decrement the byte or word destination operand by one and store the result in the destination operand. All flags except CF are updated.
NEG	Dest.	Find 2's complement of the byte or word destination operand. Update all flags.
CMP	Dest., source	Complement the byte or word destination operand.

Multiplication and Division Instructions

Assignment to students

8. Transfer of Control Instructions

The stored program computer repeatedly follows the sequence:

- fetch the instruction whose address is in IP
- increment IP
- execute the instruction

This implies that all programs will execute in a sequential manner.

However, there are times when it is necessary to transfer program control to an address that is not the next instruction in sequence.

Unconditional Jump Instructions

The stored program computer repeatedly follows the sequence:

- fetch the instruction whose address is in IP
- increment IP
- execute the instruction

This implies that all programs will execute in a sequential manner.

However, there are times when it is necessary to transfer program control to an address that is not the next instruction in sequence.

9. Processor Control Instructions

Assignment to students