

## OVERVIEW OF MICROPROCESSORS

---

### 1.1 GENERAL

A microprocessor is one of the most exciting technological innovations in electronics since the appearance of the transistor in 1948. This wonder device has not only set in the process of revolutionizing the field of digital electronics, but it is also getting entry into almost every sphere of human life. Applications of microprocessors range from the very sophisticated process controllers and supervisory control equipment to simple game machines and even toys.

It is, therefore, imperative for every engineer, specially electronics engineer, to know about microprocessors. Every designer of electronic products needs to learn how to use microprocessors. Even if he has no immediate plans to use a microprocessor, he should have knowledge of the subject so that he can intelligently plan his future projects and can make sound engineering judgements when the time comes.

The subject of microprocessors is overviewed here with the objective that a beginner gets to know what a microprocessor is, what it can do, how it fits in a system and gets an overall idea of the various components of such a system. Once he has understood signam of each component and its place in the system, he can go deeper into the working details and design of individual components without difficulty.

### 1.2 WHAT IS A MICROCOMPUTER?

To an engineer who is familiar with mainframe and mini computers, a microcomputer is simply a less powerful mini computer. Microcomputers have smaller instruction sets and are slower than mini computers, but then they are far less expensive and smaller too.

To an engineer with a hardware background and no computer experience, a microcomputer will look like a sequential state machine that can functionally replace thousands of random logic chips, but occupies a much lesser space, costs much lesser and the number of device interconnections being fewer in it, is much more reliable.

A microcomputer is primarily suited, because of its very low cost and very small size, to dedicated applications. On the same grounds, the mainframe computer is as a rule suitable as a general purpose computer. Mini computer finds applications in both areas.

### 1.3 WHAT IS A MICROPROCESSOR?

A computer, large or small, can be represented functionally (in a simplified form) by the block diagram in Figure. 1.1. As shown, it comprises of three basic parts or sub-systems:

## 2 Advanced Microprocessors

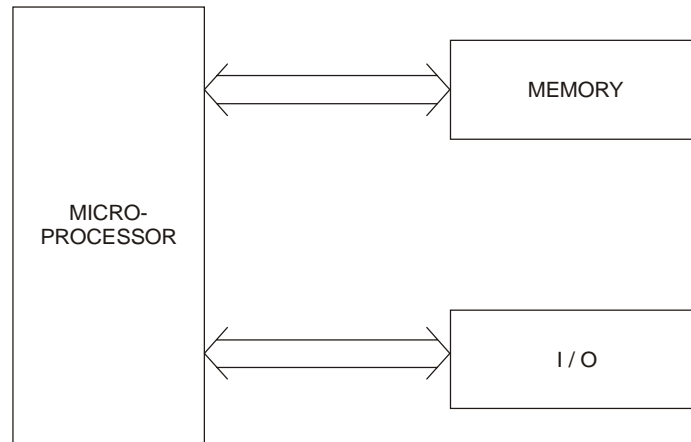


Figure 1.1 Block Diagram of Microcomputer

### (a) Central Processing Unit (CPU)

It performs the necessary arithmetic and logic operations and controls the timing and general operation of the complete system.

### (b) Input/Output (I/O) Devices

Input devices are used for feeding data into the CPU, examples of these devices are toggle switches, analog-to-digital converters, paper tape readers, card readers, keyboards, disk etc. The output devices are used for delivering the results of computations to the outside world; examples are light emitting diodes, cathode ray tube (CRT) displays, digital-to-analog converters, card and paper-tape punches, character printers, plotters, communication lines etc. The input-output subsystem thus allows the computer to usefully communicate with the outside world. Input-output devices are also called as peripherals.

### (c) Memory

It stores both the instructions to be executed (i.e., the program) and the data involved. It usually consists of both RAMs (random-access memories) and ROMS (read-only memories).

A microprocessor is an integrated circuit designed to function as the CPU of a microcomputer.

## 1.4 WHAT IS INSIDE A MICROPROCESSOR ?

The microprocessor or CPU reads each instruction from the memory, decodes it and executes it. It processes the data as required in the instructions. The processing is in the form of arithmetic and logical operations. The data is retrieved from memory or taken from an input device and the result of processing is stored in the memory or delivered to an appropriate output device, all as per the instructions.

To perform all these functions, the  $\mu\text{P}$  (microprocessor) incorporates various functional units in an appropriate manner. Such an internal structure or organizational structure of  $\mu\text{P}$ , which determines how it operates, is known as its architecture.

A typical microprocessor architecture is shown in Figure 1.2. The various functional units are as follows:

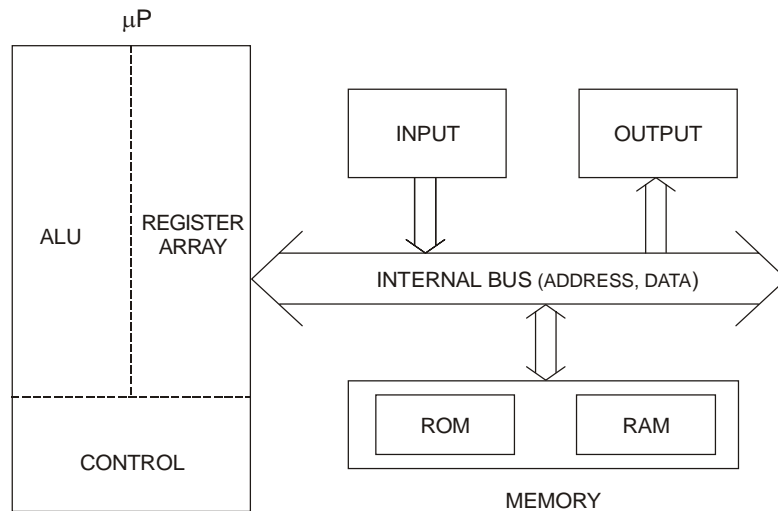


Figure 1.2 Architecture of Microprocessor

### 1.4.1 Busses

$\mu$ C (microcomputer), like all computers, manipulates binary information. The binary information is represented by binary digits, called bits.  $\mu$ C operates on a group of bits which are referred to as a word. The number of bits making  $\mu$ P a word varies with the  $\mu$ P. Common word sizes are 4, 8, 12 and 16 bits ( $\mu$ Ps with 32 bit-word have also of late entered the market). Another binary terms that will be of interest in subsequent discussions are the byte and the nibble, which represent a set of 8 bits and 4 bits, respectively.

Figure 1.2 shows busses interconnecting various blocks. These busses allow exchange of words between the blocks. A bus has a wire or line for each bit and thus allows exchange of all bits of a word in parallel. The processing of bits in the  $\mu$ P is also in parallel. The busses can thus be viewed as data highways. The width of a bus is the number of signal lines that constitute the bus.

The figure shows for simplicity three busses for distinct functions. Over the *address bus*, the  $\mu$ P transmits the address of that I/O device or memory locations which it desires to access. This address is received by all the devices connected to the processor, but only the device which has been addressed responds. The *data bus* is used by the  $\mu$ P to send and receive data to and from different devices (I/O and memory) including instructions stored in memory. Obviously the address bus is unidirectional and the data bus is bi-directional. The control bus is used for transmitting and receiving control signals between the  $\mu$ P and various devices in the system.

### 1.4.2 Arithmetic-Logic Unit (ALU)

The arithmetic-logic unit is a combinational network that performs arithmetic and logical operations on the data.

### 1.4.3 Internal Registers

A number of registers are normally included in the microprocessor. These are used for temporary storage of data, instructions and addresses during execution of a program. Those in the Intel

#### 4 Advanced Microprocessors

8085 microprocessor are typical and are described below:

##### **(i) Accumulator (Acc) or Result Register**

This is an 8-bit register used in various arithmetic and logical operations. Out of the two operands to be operated upon, one comes from accumulator (Acc), whilst the other one may be in another internal register or may be brought in by the data bus from the main memory. Upon completion of the arithmetic/logical operation, the result is placed in the accumulator (replacing the earlier operand). Because of the later function, this register is also called as result register.

##### **(ii) General Purpose Registers or Scratch Pad Memory**

There are six general purpose 8-bit registers that can be used by the programmer for a variety of purposes. These registers, labelled as B, C, D, E, H and L, can be used individually (e.g., when operation on 8-bit data is desired) or in pairs (e.g., when a 16-bit address is to be stored). Only B-C, D-E and H-L pairs are allowed.

##### **(iii) Instruction Register (IR)**

This 8-bit register stores the next instruction to be executed. At the proper time this stored word (instruction) is fed to an instruction decoder which decodes it and supplied appropriate signals to the control unit. When the execution has been accomplished the new word in the instruction register is processed.

##### **(iv) Program Counter (PC)**

This is a 16-bit register which holds the address of the next instruction that has to be fetched from the main memory and loaded into the instruction register. The program controlling the operation is stored in the main memory and instructions are retrieved from this memory normally in order. Therefore, normally the address contained in the PC is incremented after each instruction is fetched. However, certain classes of instruction can modify the PC so that the programmer can provide for branching away from the normal program flow. Examples are instructions in the "jump" and 'call subroutine' groups.

##### **(v) Stack Pointer (SP)**

This is also a 16-bit register and is used by the programmer to maintain a stack in the memory while using subroutines.

##### **(vi) Status Register or Condition Flags**

A status register consisting of a few flip-flops, called as condition flags (in 8085 the number of flags is five) is used to provide indication of certain conditions that arise during arithmetic and logical operations. These are:

'zero'	Flag is set if result of instruction is 0.
'sign'	Set if MSB of result is 1.
'parity'	Set if result has even parity.
'carry'	Set if carry or borrow resulted.
'auxiliary carry'	Set if instruction caused a carry out of bit 3 and into bit 4 of the resulting value.

### (vii) Dedicated Registers

Several other registers are incorporated in the  $\mu$ P for its internal operation. They cannot be accessed by the programmer and hence do not concern much a  $\mu$ P user.

### 1.4.4 Instruction Decoder and Control Unit

It decodes each instruction and under the supervision of a clock controls the external and internal units ensuring correct logical operation of the system.

## 1.5 SEMICONDUCTOR MEMORIES

As mentioned earlier, semiconductor memories are required in a microcomputer for storing information which may comprise of (a) the data to be used for computation, (b) instructions and (c) computational results. A program starts as a set of instructions on a paper, then this is transferred to a set of cards with the instructions punched in code on them. These instructions also can be transferred to magnetic tape, paper tape or directly into semiconductor memory which is the eventual storage space for a program. The semiconductor memory chips are connected to the  $\mu$ P through the address bus, data bus and control bus. (This is also the way that I/O devices are connected to the  $\mu$ P). See Figure 1.3.

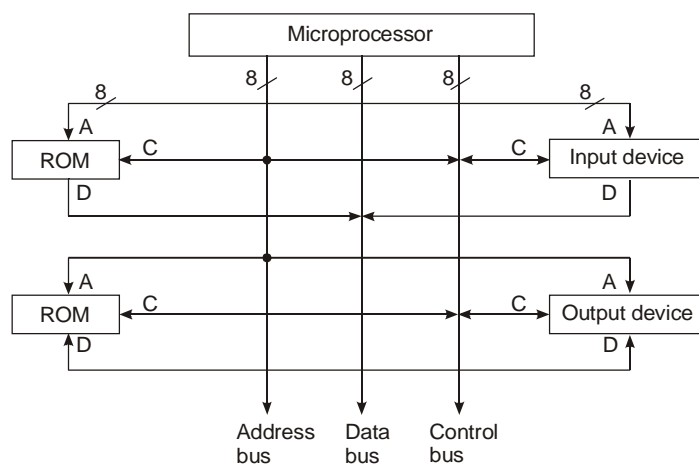


Figure 1.3 Connection of I/O Devices and Memory

### 1.5.1 Memory Classes

Memories may be broadly divided into two classes:

#### (a) Random Access Memory (RAM) or Read/Write Memory (RWM)

There is provision in RAMs (RWMs) for writing information into the memory and reading it when the microcomputer is in operation. It is, therefore, used to store information which changes or may change during the operation of the system, viz. data for calculations and results of calculations. It is also used to store the programs which are to be changed frequently. Semiconductor RAM is a volatile memory.

A RAM can be of static or dynamic type. Dynamic RAMs have higher packing densities, are faster and consume less power in the quiescent state. However, because of external refreshing circuitry requirement, the dynamic RAMs are profitable only in large sizes.

### **(b) Read-Only Memory (ROM)**

The ROM functions as a memory array whose contents once programmed, are permanently fixed and can not be altered by the  $\mu\text{P}$  while the system is operating. It is non-volatile. ROMs exist in many forms.

- (i) *Mask ROM*: It is custom programmed or mask programmed when manufactured and can not be altered thereafter. The cost of a custom built mask for programming is so high that thousands of ROMs storing the same information must be produced to pay for the mask.
- (ii) *Programmable ROM (PROM)* : This type is programmable by the user (typically by electrically overheating fusible links in selected manner). Once programmed, the contents can not be altered. The memory may be programmed one at a time by the user and is thus suitable for the cases where small quantities of a ROM are needed.
- (iii) *Electrically Alterable ROM (EAROM)* : In this type of memory, the contents can be electrically erased (by applying a large negative voltage to control gates of memory cells) and the memory can be then reprogrammed (by applying a large positive voltage to control gates). This type is convenient when the user is not sure of the program and may wish to modify it. This is a typical requirement in prototype development.
- (iv) *Erasable Programmable ROM (EPROM)* : Like EAROM, this type of memory can also be erased and reprogrammed. However, erasing is by exposing the memory chips to high intensity ultraviolet light of a wavelength close to 2537 Å. It has the same application filed as the EAROM.

## **1.5.2 Bipolar v/s MOS Memories**

Basically there are two semiconductor technologies, namely, bipolar and MOS unipolar. Mask ROMs and PROMs are available in both types whereas EAROMs and EPROMs are made with MOS technology only.

In general, bipolar devices (including memories) are faster and have higher drive capabilities. On the other hand, MOS devices consume less space and power and are cheaper. Therefore, MOS memories are preferred where speed is not a critical factor.

## **1.6 PERIPHERAL INTERFACING**

### **1.6.1 Functions**

When one or more I/O devices (peripherals) are to be connected to a  $\mu\text{P}$ , an interface network for each device, called peripheral interface, is required.

The interface incorporate commonly the following four functions:

- (a) *Buffering*: Which is necessary to take care of incompatibility between the  $\mu\text{P}$  and the peripheral.
- (b) *Address Decoding*: Which is required to select one of the several peripherals connected in the system.

- (c) *Command Decoding*: Which is required for peripherals that perform actions other than data transfers.
- (d) *Timing and Control*: All the above functions require timing and control.

### 1.6.2 Data Transfer

Data exchange or transfers which occur between a peripheral device and the  $\mu\text{C}$  fall into one of the following two broad categories:

#### (i) *Programmed Data Transfer*

A software routine residing in memory requests the peripheral device for data transfer to or from the  $\mu\text{P}$ . Generally, the data is transferred to or from the accumulator though in some  $\mu\text{Ps}$ , others internal registers may also participate in the transfer.

Programmed data transfers are generally used when a small amount of data is transferred with relatively slow I/O devices, e.g., A/D and D/A converters, peripheral multiplier, peripheral floating point arithmetic unit etc. In these cases, usually one word of data is transferred at a time.

#### (ii) *Direct Memory Access (DMA) Transfer or Cycle Stealing Transfer*

In this mode, the data transfer is controlled by the peripheral device. The  $\mu\text{P}$  is forced to hold on by an I/O device until the data transfer between the device and the memory is complete. Since the data control transfer is controlled entirely by hardware, the interface is more complex than that required for a programmed data transfer. DMA transfer is used when a large block of data is to be transferred, for example, for transferring data from peripheral mass storage devices like the floppy disk and high-speed card reader.

### 1.6.3 Interfacing Devices

Extensive hardware (IC chips) is now available for designing custom interfaces. Included in this are multiplexer, and demultiplexers, line drivers and receivers, level translators and buffers a stable and monostable multivibrators, latches, gates, shift registers, and so on.

Then there are more sophisticated interfaces-the programmable inter-faces, whose functions may be altered by an instruction from the  $\mu\text{P}$ . These interfaces can be of general purpose type or special purpose (dedicated function) type.

## 1.7 $\mu\text{P}$ SOFTWARE AND PROGRAM LANGUAGES

The foregoing was a brief account of the  $\mu\text{C}$  hardware and now follows a concise picture of the  $\mu\text{C}$  software. A sequence of instructions designed to perform a particular task in a computer is known as the program and a set of programs written for a computer-based system is called software for that system.

### 1.7.1 Programming Languages

#### (i) *Machine Language (ML)*

Since a machine (computer) can handle (i.e., store and process) information in binary form only, the instruction (program) must be finally encoded in binary for feeding to the machine. A program in this form is thus in machine language.

Comparison between Various Microprocessors

Name	Date	Transistors	Microns	Clock speed	Data width	MIPS
8080	1974	6,000	6	2 MHz	8	0.64 MIPS
8088	1979	29,000	3	5 MHz	16 bits, 8 bit bus	0.33 MIPS
80286	1982	134,000	1.5	6 MHz	16 bits	1 MIPS
80386	1985	275,000	1.5	16 MHz	32 bits	5 MIPS
80486	1989	1,200,000	1	25 MHz	32 bits	20 MIPS
Pentium	1993	3,100,000	0.8	60 MHz	32 bits, 64 bit bus	100 MIPS
Pentium II	1997	7,500,000	0.35	233 MHz	32 bits, 64 bit bus	400 MIPS ?
Pentium III	1999	9,500,000	0.25	450 MHz	32 bits, 64 bit bus	1,000 MIPS ?

First home computers  
 First IBM PC  
 IBP ATs. Up to 2.66  
 MIPS at 12MHz  
 Eventually 33 MHz, 11.4  
 MIPS  
 Eventually 50 MHz, 41  
 MIPS  
 Eventually 200 MHz  
 Eventually 200 MHz.  
 Eventually 450 MHz,  
 800 MIPS ?



**(ii) Assembly Language (AL)**

In an assembly language programme, the instructions including the storage locations are represented by alphanumeric symbols, called *mnemonics*. It is relatively easier to write a programme in assembly language than in machine language. However, if written in AL, it must be translated to ML before it can be stored and executed in  $\mu\text{C}$ . Mostly one statement in AL translates to one instruction in ML.

**(iii) High-Level Language (HLL)**

Programming in AL is very tedious and time consuming. High Level languages like Fortran, Cobol, Algol, Pascal and PL/M can be used for programming and then the program translated into ML program. One statement in HL generally corresponds to several ML instructions.

**1.7.2 Software Tools****(i) Assembler**

It is a computer program that translates an AL program to ML program (also called object code). A *cross assembler* is an assembler that is executed on a machine other than the one for which it is producing the ML program. A *self assembler* or *resident assembler*, on the other hand, is meant to be run on the machine for which ML program is required to be produced.

**(ii) Compiler**

A computer program that translates a HLL program to ML program (object code). Like the assembler, the compiler can be a cross compiler or a self (resident) compiler.

**(iii) Editor**

During the process of program entry into the memory or debugging, it may be necessary to make changes in the program text in order to correct any errors or modify the logic. An editor helps the programmer to do this.

**(iv) System Monitor**

All single boards  $\mu\text{Cs}$ , design kits and  $\mu\text{P}$  development systems have system monitor as an integrated component. The monitor (a ML program) resides in ROM. It helps a programmer in performing such functions as entering a program into memory, getting a program executed, modifying contents of one or more memory locations or  $\mu\text{P}$  registers, displaying the contents of any memory location of a register in the  $\mu\text{P}$ , entering ML program in convenient hexadecimal format, etc.

**1.8 MICROCOMPUTER INSTRUCTION SET**

The instruction set of a microprocessor will typically comprise of *five groups* of instructions:

**(i) Data Transfer Group**

These instructions help to move data between registers within the microprocessor, between a register and a memory location or between memory location.

**(ii) Arithmetic Group**

Instructions in this group add, subtract, increment or decrement data in registers or in memory (e.g., an instruction to add the contents of two registers within the microprocessor).

### **(iii) Logical Group**

These are used to AND, OR, EXCLUSIVE-OR, Compare, rotate or complement data in registers or in memory (e.g., OR the contents of two registers within the microprocessor).

### **(iv) Branch Group**

This group includes conditional and unconditional jump instructions, subroutine-call instructions and return-from-subroutine instructions. A conditional instruction specifies that a certain operation be performed only if a certain condition has been met (e.g., jump to a particular instruction if the result of the last operation was zero). Conditional instructions provide decision making capability in programs.

### **(v) Stack, I/O and Machine Control Group**

This group of instructions performs data transfer between the microprocessor and I/O devices, manipulates the stack and alters internal control flags. These instructions make it possible for the programmer to halt the microprocessor, put it in a “no operation” state, enable/disable its interrupt system and so on.

Instructions, which are stored alongwith data in the memory, may be one or more bytes in length. Multiple byte instructions are stored in successive memory locations: the address of the first byte is always used as the address of the instruction. Also, the first byte is always the operation code (abbreviated as OPCODE).

## **1.9 MICROPROCESSOR DEVELOPMENT CHRONOLOGY**

The first microprocessor was announced in 1971 by Intel Corporation, U.S.A. This was the Intel 4004. It was on a single chip and was a 4-bit microprocessor (i.e., operated on 4 bits of data at a time).

Encouraged by the success of 4004, Intel Corp. introduced its enhanced version, the Intel 4040. Many other companies also announced 4-bit microprocessors, examples are Rockwell International's PPS4, NEC's  $\mu$ COM 4 and Toshiba's T3472.

The first 8-bit microprocessor was announced in 1973, again by Intel Corp. This was the Intel 8008. An improved version, Intel 8030, followed. Several other companies followed the suit. Today the better known 8-bit mPs are Intel's 8085, Motorola's M6800, NEC's  $\mu$ COM85AF, National \* SC/MP, Zilog Corporation's Z80 and Fairchild's F8.

Then followed 12-bit and 16-bit  $\mu$ Ps. Examples of 12-bit  $\mu$ Ps are Intersil's IM 6100 and Toshiba's T3190 and those of 16-bit  $\mu$ Ps Intel's 8086, Fairchild's 9440, Texas Instrument's TMS 9940 and TMS 9980, Zilog's Z8000, Motorola's M68000.

The developments in  $\mu$ P since 1971 have been in the direction of (a) improving architecture, (b) improving instruction set, (c) increasing speeds, (d) simplifying power requirements and (e) incorporating more and more memory space and I/O facilities in the same chip (thus giving use to single chip computers).

- The *date* is the year that the processor was first introduced. Many processors are re-introduced at higher clock speeds for many years after the original release date.
- *Transistors* is the number of transistors on the chip. You can see that the number of transistors on a single chip has risen steadily over the years.
- *Microns* is the width, in microns, of the smallest wire on the chip. For comparison, a human hair is 100 microns thick. As the feature size on the chip goes down, the number of transistors rises.

- *Clock speed* is the maximum rate that the chip can be clocked. Clock speed will make more sense in the next section.
- *Data Width* is the width of the ALU. An 8-bit ALU can add/subtract/multiply/etc. two 8-bit numbers, while a 32-bit ALU can manipulate 32-bit numbers. An 8-bit ALU would have to execute 4 instructions to add two 32-bit numbers, while a 32-bit ALU can do it in one instruction. In many cases the external data bus is the same width as the ALU, but not always. The 8088 had a 16-bit ALU and an 8-bit bus, while the modern Pentiums fetch data 64 bits at a time for their 32-bit ALUs.
- *MIPS* stands for Millions of Instructions Per Second, and is a rough measure of the performance of a CPU. Modern CPUs can do so many different things that MIPS ratings lose a lot of their meaning, but you can get a general sense of the relative power of the CPUs from this column.

From this table you can see that, in general, there is a relationship between clock speed and MIPS. The maximum clock speed is a function of the manufacturing process and delays within the chip. There is also a relationship between the number of transistors and MIPS. For example, the 8088 clocked at 5 MHz but only executed at 0.33 MIPS (about 1 instruction per 15 clock cycles). Modern processors can often execute at a rate of 2 instructions per clock cycle. That improvement is directly related to the number of transistors on the chip.

## 1.10 MANUFACTURING TECHNOLOGIES

Broadly two technologies have been used in the manufacture of  $\mu$ Ps: MOS and Bipolar. The majority of  $\mu$ Ps available in the market use MOS technology because of its two distinct merits, namely, a higher component density and a lower manufacturing cost. The bipolar-technology-based  $\mu$ Ps are limited to special applications that call for high speeds in which respect MOS devices are inferior. Because of the size problem the bipolar  $\mu$ Ps are usually made in bit-slice configuration; examples being Intel's 3002 (2-bit slice, TTL), Transiltron's 1601 (4-bit slice, TTL) and Texas Instrument's SBP 0400 (4-bit slice, TTL).

The first few types of  $\mu$ Ps to be announced (e.g., 4004, 4040, 8008) were based on PMOS technology, which is now obsolete for  $\mu$ Ps because of its speed limitation. The NMOS is the main technology today in use for low cost  $\mu$ Ps (e.g., 8080, 8085, Z-80, 6800, 808, 6800, 8086, Z-8000, 68000). The CMOS technology based  $\mu$ Ps (e.g., RCA's COSMAC) have limited application because of lower packing density and higher cost. The exceptions are the less cost-sensitive military and aerospace applications, where low power dissipation (typical of the CMOS devices) is of prime importance.

## ASSIGNMENTS

1. What is a general purpose microcomputer?
2. Name three classes of computers.
3. What is the heart of microcomputer system called?
4. How much ROM is provided in the PC?
5. What are the standard data word lengths for which microprocessors have been developed?
6. What is the difference between microprocessor and microcomputer?
7. Define bit, byte, word and instruction.
8. What is an assembler?
9. Explain the difference between compiler and interpreter.
10. What is an assembler?