Microprocessors and Interfacing

8086, 8051, 8096, and advanced processors

N. Senthil Kumar
Professor
Department of Electrical and Electronics Engineering
Mepco Schlenk Engineering College
Sivakasi, Tamil Nadu

M. Saravanan
Professor
Department of Electrical and Electronics Engineering
Thiagarajar College of Engineering
Madurai, Tamil Nadu

S. Jeevananthan
Professor
Department of Electrical and Electronics Engineering
Pondicherry Engineering College
Puducherry

S.K. Shah
Professor and Head
Department of Electrical Engineering
MS University of Baroda
Vadodara, Gujarat

Oxford University Press

OXFORD
UNIVERSITY PRESS

© Oxford University Press. All rights reserved.

OXFORD
UNIVERSITY PRESS

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide. Oxford is a registered trade mark of
Oxford University Press in the UK and in certain other countries.

Published in India by
Oxford University Press
YMCA Library Building, 1 Jai Singh Road, New Delhi 110001, India

© Oxford University Press 2012

The moral rights of the author/s have been asserted.

First published in 2012

All rights reserved. No part of this publication may be reproduced, stored in
a retrieval system, or transmitted, in any form or by any means, without the
prior permission in writing of Oxford University Press, or as expressly permitted
by law, by licence, or under terms agreed with the appropriate reprographics
rights organization. Enquiries concerning reproduction outside the scope of the
above should be sent to the Rights Department, Oxford University Press, at the
address above.

You must not circulate this work in any other form
and you must impose this same condition on any acquirer.

ISBN-13: 978-0-19-807906-4
ISBN-10: 0-19-807906-0

Typeset in Times New Roman
by Trinity Designers & Typesetters, Chennai
Printed in India by Tara Art Printers (P) Ltd, Noida

Oxford University Press

© Oxford University Press. All rights reserved.

Brief Contents

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Detailed Contents

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

Oxford University Press

© Oxford University Press. All rights reserved.

1

Microprocessors—Evolution and Introduction to 8085

LEARNING OUTCOMES

After studying this chapter, you will be able to understand the following:

- Importance of microprocessors
- Origin and evolution of microprocessors
- Classification of microprocessors and memories
- Common input and output devices for computers
- Bus structures used in computers and technology improvements
- 8085 microprocessor architecture and instruction set

Oxford University Press

1.1 INTRODUCTION

The microprocessor is an electronic chip that functions as the central processing unit (CPU) of a computer. In other words, the microprocessor is the heart of any computer system. Microprocessor-based systems with limited resources are called microcomputers. Today, microprocessors can be found in almost all consumer electronic devices such as computer printers, washing machines, microwave ovens, mobile phones, fax machines, and photocopiers and in advanced applications such as radars, satellites, and flights. Any middle-class household will have about a dozen microprocessors in different forms inside various appliances. The recent developments in the electronics industry and the large-scale integration of devices have led to rapid cost reduction and increased application of microprocessors and their derivatives.

Typically, basic microprocessor chips have arithmetic and logic functional units along with the associated control logic to process the instruction execution. Almost all microprocessors use the basic concept of *stored-program execution*. Programs or instructions to be executed by the microprocessor are stored sequentially in memory locations. The microprocessor, or the processor in general, fetches the instructions one after another and executes them in its arithmetic and logic unit. So all microprocessors have a built-in memory access and management part as well as some amount of memory.

A microprocessor can be programmed to perform any task that can be written and programmed by the user. Without a program, the microprocessor unit is a piece of useless electronic circuit. The programmer must take care of all the resources of the microprocessor and use them efficiently for implementing the required functionality. So to work with the microprocessor, it is necessary for the programmer to know about its internal resources and features. The programmer

© Oxford University Press. All rights reserved.

must also understand the instructions that a microprocessor can support. Every microprocessor has its own associated set of instructions; this list is given by all microprocessor manufacturers. The instruction set for microprocessors is in two forms—one in mnemonic, which is comparatively easy to understand and the other in binary machine code, which the microprocessor works with and is difficult for us to understand. Generally, programs are written using mnemonics called assembly-level language and then converted into binary machine-level language. This conversion can be done manually or using an application called assembler.

In general, programs are written by the user for the microprocessor to work with real world data. Data are available in many forms and from many sources. To input these data to the microprocessor, the microprocessor-based systems need some input interfacing circuits and some electronic processing circuits. These circuits include data converters and ports. After processing the real world data, the output from the microprocessor must be taken out to give to the output devices or circuits. This again needs interfacing circuits and ports. So a microprocessor-based system will need a set of memory units and interfacing circuits for inputs and outputs. The circuits, together with the microprocessor, make the *microcomputer system*. The physical components of the microcomputer system are called *hardware*. The program that makes this hardware useful is called *software*.

The semiconductor manufacturing technology for chips has developed from transistor–transistor logic (TTL) to complementary metal-oxide-semiconductor (CMOS). Microprocessor manufacturing also has gone through these technological changes. The other semiconductor manufacturing technology available is emitter-coupled logic (ECL). TTL technology is most commonly used for basic digital integrated circuits; CMOS is favoured for portable computers and other battery-powered devices because of its low power consumption.

1.2 EXPLANATION OF BASIC TERMS

The terms relevant to the use of microprocessors are explained in this section. These explanations will give the reader an understanding of various microprocessor-related terms, technologies, and topics.

Chip A chip or an integrated circuit is a small, thin piece of silicon with the required circuits and transistors etched on it to perform a particular function. Simpler processors may consist of a few thousand transistors etched onto a silicon base just a few millimeters square.

Bit A bit means a single binary digit. The bit is also the fundamental storage unit of computer memory. In binary form, a bit can have only two values, 0 or 1, whereas a decimal digit can have 10 values, represented by symbols 0 through 9.

Bit size The bit size of a microprocessor refers to the number of bits that can be processed simultaneously by the basic arithmetic circuits of the microprocessor.

Word A word is a number of bits grouped together for processing. In microprocessors, a word refers to the basic data size or bit size that can be processed

© Oxford University Press. All rights reserved.

by the arithmetic and logic unit (ALU) of the processor. A 16-bit binary number is called a word in a 16-bit processor.

Memory word The number of bits that can be stored in a register or memory element is called memory word. Mostly, all memory units use eight bits for their memory word.

Byte An 8-bit word is referred to as a byte.

Nibble A 4-bit word is referred to as a nibble.

Kilobyte A collection of 1024 bytes is called a kilobyte (2^{10} bytes).

Megabyte A collection of 1024 kilobytes is called a megabyte (2^{20} bytes).

RAM or R/W memory Random access memory or read/write memory is a type of semiconductor memory in which a particular memory location can be erased and written with new data at any time. These memory units are volatile, which means that the contents of the memory are erased when the power to the chip is disrupted. The access of the individual memory location can be done randomly. In microprocessors, the RAM is used to store data.

DRAM Dynamic random access memory is a semiconductor memory in which the stored contents need to be refreshed repeatedly at about thousands of times per second. Without refreshing, the stored data will be lost. These memory chips are preferred in a computer system as these are slower but economical.

SRAM Static random access memory chips keep the data stored in it as long as power is available. There is no need for refreshing. In terms of speed, SRAM is faster.

ROM Read only memory devices are memory devices whose contents are retained even after removing the power supply.

Arithmetic and logic unit ALU is a digital circuit present in the microprocessor to perform arithmetic and logic operations on digital data. The typical operations performed by the ALU are addition, subtraction, logical AND, logical OR, and comparison of binary data. Generally, the functions of the ALU of a microprocessor will decide the processor's functionality.

Microcontroller A microcontroller is a chip that includes microprocessor, memory, and input/output signal ports. Microcontrollers can be called single-chip microcomputers.

Microcomputer The system formed by interfacing the microprocessor with the memory and I/O devices to execute the required programs is called microcomputer.

Bus A bus is a group of wires/lines that carry similar information.

System bus The system bus is a group of wires/lines used for communication between the microprocessor and peripherals.

Oxford University Press

© Oxford University Press. All rights reserved.

Firmware Software written for a microprocessor application without provision for changes is called firmware. These are stored in the permanent storage or ROM of the computer system.

Input device The devices that are used for providing data and instructions to the microprocessor or microcomputer system are called input devices. Keyboard and mouse are the common input devices.

Output device The devices that are used for transferring data out of the microprocessor or microcomputer system are called output devices. Display screen, printer, and other forms of display are the common output devices.

Floppy disk A removable-type magnetic disk used for storing programs and data for transferring from and to the computer is called floppy disk.

Disk drive The hardware component that is used to read or write data to devices such as floppy disks is called disk drive.

Computer architecture The design, internal configuration, and accesses in a digital computer are together called computer architecture.

Von–Neumann architecture The architecture in which the same memory is used for storing programs as well as data.

Harvard architecture The architecture in which programs and data are stored in two separate memory units.

CISC processor Complex instruction set computer is a processor architecture that supports many machine language instructions.

RISC processor Reduced instruction set computer is a processor architecture that supports limited machine language instructions. RISC processors are expected to execute the programs faster than CISC processors.

High-level language A computer programming language in which programs are written without the knowledge of the processor in which the program will be executed. BASIC, Fortran, C, Pascal, and Java are examples of high-level languages.

Assembly language A programming language written using the mnemonics or the instruction set of a particular microprocessor is called assembly language. Assembly language programming is microprocessor-specific. It is not as easily understood as a high-level language program, but is easier than a machine language program.

Machine language Machine language refers to binary code programs that are specific to the processor and can be directly executed by the processor. Machine language is the lowest level language and cannot be easily understood.

Assembler A computer application program that converts the assembly language program into machine-level language program.

Compiler A computer program that converts the high-level language program into machine-level language program.

Oxford University Press

© Oxford University Press. All rights reserved.

Interpreter A computer program that reads the high-level or assembly-level program one line at a time and converts it into machine-level program. Compiler and assembler can function only on the entire program in a file.

Algorithm A sequence of operations or instructions that defines how to solve a problem using a computer or microcomputer. An algorithm must be definite, must follow a clear instruction flow without ambiguity, and must have definite start and end points.

BIOS Basic input/output system is a set of programs that handles the input and output functions and interacts with the hardware directly. A new hardware installed must be provided with the corresponding BIOS routines.

Clock The circuit in the computer that generates the sequence of evenly spaced pulses to synchronize the activities of the processor and its peripherals is called clock. The clock speed determines the speed of the operation of the computer. The computer with a high frequency clock works faster. Normally the clock frequency is in the range of megahertz (MHz) or gigahertz (GHz).

MIPS Million instructions per second is a measure of the speed at which the instructions are executed in a processor.

Tri-state logic It is the logic used by digital circuits. The three logic levels used are high (1), low (0), and high impedance state (Z). The logic high state of a digital circuit can source current and the logic low can sink current in a computer system, but the high impedance state neither sources nor sinks current and so the other devices connected to it are not affected.

Operating system The program that controls the entire computer and its resources and enables users to access the computer and its resources is called operating system. It is required for any computer system to become operational and user friendly. Under the control of the operating system, the computer recognizes and obeys commands typed by the user. In addition, the operating system provides built-in routines that allow the user's program to perform input/output operations without specifying the exact hardware configuration of the computer. In low-level microprocessor-based systems, the program that controls the hardware is called *monitor routine* or *monitor software*.

1.3 MICROPROCESSORS AND MICROCONTROLLERS

The microprocessor (also called CPU) is the principal element of a computer as it executes lists of instructions. These instruction lists are commonly called *programs*. This programming language is complex to use since it is machine- or processor-specific and coded into hexadecimal and binary.

Two types of processors are manufactured—the microprocessor and the microcontroller. At the data processing level, the two are practically equivalent. The distinction comes from the established functionalities.

The general-purpose microprocessors give the computers all the necessary computing power. These microprocessors need additional circuitry elements such

© Oxford University Press. All rights reserved.

as memory devices and I/O ports to connect the input and output devices. All microprocessor-based systems need two types of memories—RAM and ROM. RAM is used for storage of data while ROM is used for storage of programs, especially the start-up program that runs when the microprocessor is powered on.

There are numerous microprocessors developed by many companies. The evolution of microprocessors, from 4-bit microprocessors to 64-bit microprocessors, has been discussed later in this chapter. This book is devoted to the discussion of two groups of microprocessors—Intel's 8-bit 8085 microprocessor series in brief and 16-bit 8086 series in detail.

Microcontrollers are microprocessors designed specially for control applications. Microcontrollers contain memory units and I/O ports inside a chip, in addition to the CPU. Microcontrollers are otherwise called embedded controllers; they are generally used to control and operate smart machines. Some of the machines using microcontrollers are microwave ovens, washing machines, sewing machines, automobile ignition systems, computer printers, and fax machines. You will be amazed to know that out of 100 processor chips manufactured, 99 are embedded processors; only one goes into a general computer! A plethora of semiconductor companies are in the microcontroller market and any application development engineer is flooded with a variety of microcontrollers to choose from. This book discusses Intel's 8-bit 8051 series and 16-bit 8096 series as also other advanced microcontrollers.

1.4 MICROPROCESSOR-BASED SYSTEM

A computer system developed using a basic general-purpose microprocessor is called a microcomputer system. The system consists of CPU, memory, and I/O ports as shown in Fig. 1.1.

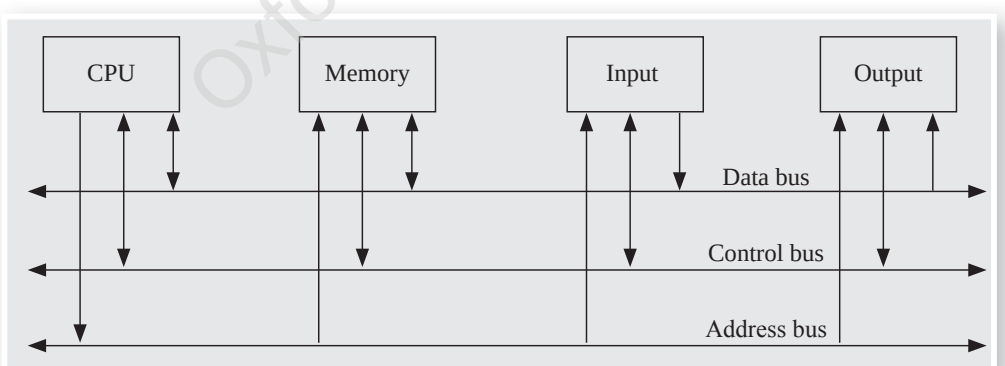

Fig. 1.1 Microcomputer system (Von–Neumann model)

Figure 1.2 shows a typical personal computer system. The interfacing of the processor with the other parts of the microcomputer system needs a three-bus architecture. The three buses are data bus, address bus, and control bus.

Each memory location or I/O port is identified by a specific address similar to a postal address. In microprocessor systems, the addresses are all in binary, and in general, represented in hexadecimal number format. The address is a

© Oxford University Press. All rights reserved.

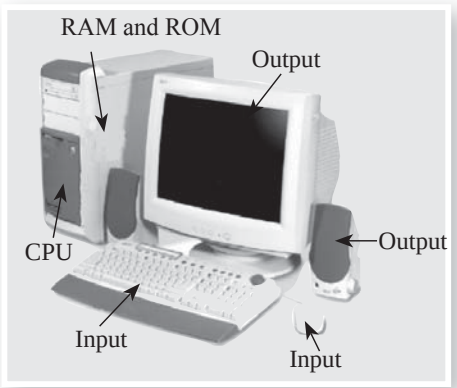

Fig. 1.2 Personal computer

unique pattern used to identify a location in the memory or an I/O port. The address bus consists of many lines that transport the digital data sent by the processor. An address bus of eight bits corresponds to eight lines of addresses and can thus address 2^8 different memory locations. These addresses are written in hexadecimal number format as 00H–FFH and can be used to for 256 different locations. Similarly, the 16-bit address bus can address 2^{16} different addresses. Its address range is 0000H–FFFFH. The greater the number of lines in the address bus, the greater the number of locations the processor is able to manage.

The address on the address bus can locate a specific memory or I/O location. After selecting the location, the data transfer between the memory and processor or between the I/O device and the processor is done through the data bus. The width of the data bus determines the data size that can be transferred. An 8-bit processor will generally have an 8-bit data bus and a 16-bit processor will have a 16-bit data bus. The memory locations in microprocessors are accessed as 8-bit or one-byte units only. So the transfer of a 16-bit data from memory needs two memory addresses. A 1 KB memory chip will have 1024 bytes of memory locations.

A control bus is needed for proper data transfer between the processor and the peripherals. The control bus basically consists of signals for selection of the correct memory or I/O device from the address, indication of the direction of data transfer, and synchronization of data transfer between slow devices. Many of the control signals are given by the processor itself because the processor is the master of the computer system. Some control signals such as selection of the correct memory chip can be generated externally by the logic circuits. The timing of the control signal is very important; the entire timing of the operation is controlled by the microprocessor in synchronization with the clock signal input.

1.5 ORIGIN OF MICROPROCESSORS

The microprocessor is the greatest invention of the 20th century. Its evolution started from the earlier mechanical calculating devices, in the 1930s. These devices used mechanical relays. Later, in the 1950s, these devices were replaced by vacuum tubes. The vacuum tubes were quickly replaced by transistors. The breakthrough in transistor technology led to the introduction of minicomputers in the 1960s and the personal computer revolution in the 1970s.

The transistor technology led to the development of complex devices called integrated circuits (ICs). The microprocessor, or microprocessing unit (MPU), later evolved as an IC and was designed to fetch instructions and execute the predefined arithmetic and logic functions. Intel was the first MPU producer and has been holding a large share of the world market for this product. The evolution

© Oxford University Press. All rights reserved.

of microprocessors is categorized into five generations: first, second, third, fourth, and fifth.

1.5.1 First Generation (1971–1973)

The microprocessors that were introduced from 1971 to 1973 were referred to as the first-generation systems. First-generation microprocessors processed their instructions serially—they fetched the instruction, decoded it, and then executed it. The first microprocessor, the 4004, was introduced in 1971. It was co-developed by Busicom, a Japanese manufacturer of calculators, and Intel, a US manufacturer of semiconductors. The 4-bit 4004 microprocessors ran at 108 kHz and contained 2300 transistors. They were fabricated using p-channel metal-oxide-semiconductor (PMOS) technology, which provided low cost, slow speed, and low output currents. They were not compatible with TTL. In 1972, Intel made the 8-bit 8008 and 8080 microprocessors.

1.5.2 Second Generation (1974–1978)

As the technology evolved, the number of circuits that could be fabricated on a chip grew. Very large-scale integration (VLSI) led to chips that had speeds up to hundreds of millions of switchings per second. The second generation marked the beginning of very efficient 8-bit microprocessors. Some of the popular processors were Motorola's 6800 and 6809, Intel's 8085, and Zilog's Z80. The second-generation devices marked a sharp contrast with the use of newer semiconductor technology to fabricate chips. They were manufactured using n-channel metal-oxide-semiconductor (NMOS) technology. This technology offered faster speed and higher density than PMOS. It resulted in a five-fold increase in instruction execution speed and higher chip densities.

1.5.3 Third Generation (1978–1980)

The third generation, introduced in 1978, was dominated by Intel's 8086 and Zilog's Z8000, which were 16-bit processors with minicomputer-like performance. These processors had the technology of 16-bit arithmetic and pipelined instruction processing. The third generation came with IC transistor counts of about 250,000. In Motorola's MC68020, for example, an on-chip cache was incorporated for the first time and the depth of the pipeline was increased to five or more stages. It was designed using high density metal-oxide-semiconductor (HMOS) technology. HMOS provides some advantages over NMOS: Its speed–power product is four times better than that of NMOS; it can accommodate twice the circuit density of NMOS.

1.5.4 Fourth Generation (1981–1995)

The microprocessors entered their fourth generation with designs containing more than a million transistors in a single package. This era marked the beginning of 32-bit microprocessors. Intel introduced 80386 and Motorola introduced 68020/68030. They were fabricated using high density/high speed complementary metal-oxide-semiconductor (HCMOS), a low-power version of the HMOS technology.

Oxford University Press

© Oxford University Press. All rights reserved.

1.5.5 Fifth Generation (1995–till date)

The fifth generation microprocessors employ decoupled super scalar processing and their design contains more than 10 million transistors. This generation marks the introduction of devices that carry on-chip functionalities. It has also paved the way for high speed memory I/O devices along with the introduction of 64-bit microprocessors. Intel leads the show here with Pentium, Celeron, and very recently, dual- and quad-core processors working with up to 3.5 GHz speed. This generation is characterized by a low-margin single-microprocessor PC business, which is complemented by high-volume sales. Table 1.1 gives the comparison of the major processors based on specific parameters such as clock speed and data word size.

Table 1.1 Comparison of general-purpose processors

General-purpose processors	Transistors	CPU speed	Data length (bits)
8080	6,000	2 MHz	8
8085	6,500	3 MHz	8
8088	29,000	3 MHz	16
8086	30,000	4 MHz	16
80286	1,34,000	6 MHz	16
80386	2,75,000	16 MHz	16/32
80486	12,00,000	33 MHz	16/32
Athalon XP	37,00,000	2.8 GHz	16/32/64
Celeron	75,00,000	1.06–2 GHz	32
Pentium II	75,00,000	233–450 MHz	32
Pentium III	95,00,000	450 MHz–1 GHz	32
Pentium III Xeon	2,81,00,000	500 MHz–1 GHz	32
Pentium 4	5,50,00,000	1.4–2.2 GHz	32
IBM PowerPC G3	65,00,000	233–333 MHz	32
PowerPC G4	1,05,00,000	400–800 MHz	32

Oxford University Press

1.5.6 Timeline of Microprocessor Evolution

(i) 1971—Intel 4004 microprocessor with 2300 transistors, working at a speed of 108 kHz

(ii) 1971—Intel 8008, twice as powerful as the 4004, with 3500 transistors and speed of 200 kHz

(iii) 1974—Intel 8080 processor with 6000 transistors and speed up to 2 MHz

(iv) 1976—Intel 8085 processor with about 6500 transistors and speed of 3–5 MHz came into existence. There were multiple versions of 8085 microprocessors. The original version of the 8085 microprocessor without suffix A was manufactured by Intel. It was quickly replaced with the 8085A, which had a bug-fixer. A few years later, in the 1980s, Intel introduced the 8085AH, the HMOS version of 8085A followed by the 80C85A, the CMOS version of the 8085A.

© Oxford University Press. All rights reserved.

(v) 1978—Intel 80X86 families of microprocessors. The first generation of the 80X86 families included the 8086 and the 8088. It was followed by the 80186, 80286, 80386, and 80486.

(vi) 1979—Intel 8088, which was similar in architecture to the 8086; the difference was in the available number of data bits of the data bus. Number of transistors: 29,000; speed: 5 MHz, 8 MHz, 10 MHz

(vii) 1985—Intel 80386, the first 32-bit chip that contained 275,000 transistors, processing five million instructions per second, and running all popular operating systems, including Windows.

(viii) 1989—Intel 486 with an 8 KB cache memory (shared for data and instructions), operating at clock frequencies from 25 to 100 MHz

(ix) 1993—Intel Pentium processor retains the 32-bit address bus of the 80486 but doubles the data bus to 64 bits. It includes two 8 KB cache memories—one for instructions and the other for data. It was based on dual pipeline method known as superscalar architecture and currently operates with frequencies up to 1.75 GHz, 20-stage pipeline, and three-level cache memory architectures.

(x) 1997—Intel Pentium II processor was designed specifically to process video, MMX audio, and graphics data efficiently with speeds of 200 MHz, 233 MHz, 266 MHz, and 300 MHz.

(xi) 1999—Intel Celeron processor and Intel Pentium III processor

(xii) 2000—Intel Pentium 4 processor

Various other companies such as Motorola, NEC, Mitsubishi, Siemens, AMD, Toshiba, and Texas Instruments also manufacture processor chips. These companies have their own chips and architectures in addition to the regular Intel-based architectures.

1.6 CLASSIFICATION OF MICROPROCESSORS

Microprocessors can be classified based on their specifications, applications, and architecture.

Based on the size of the data that the microprocessors can handle, they are classified as 4-bit, 8-bit, 16-bit, 32-bit, and 64-bit microprocessors.

Based on the application of the processors, they are classified as follows:

(i) General-purpose processors
(ii) Microcontrollers
(iii) Special-purpose processors

General-purpose processors are those that are used in general computer system integration and can be used by the programmer for any application. Common microprocessors from Intel 8085 to Intel Pentium are examples of general-purpose processors. *Microcontrollers* are microprocessor chips with built-in hardware for the memory and ports. These chips can be programmed by the user for any generic control application. *Special-purpose processors* are designed specifically

© Oxford University Press. All rights reserved.

to handle special functions required for an application. Digital signal processors are examples of special-purpose processors; these have special instructions to handle signal processing. Application-specific integrated circuit (ASIC) chips are also examples of this category of microprocessors.

Based on the architecture and hardware of the processors, they are classified as follows:

(i) RISC processors
(ii) CISC processors
(iii) VLIW processors
(iv) Superscalar processors

RISC is a processor architecture that supports limited machine language instructions. *RISC processors* can execute programs faster than CISC processors. *CISC processors* have about 70 to a few hundred instructions and are easier to program. However, CISC processors are slower and more expensive than RISC processors. *Very long instruction word (VLIW) processors* have instructions composed of many machine operations. These instructions can be executed in parallel. This parallel execution is called instruction-level parallelism. VLIW processors also have a large number of registers. *Superscalar processors* use complex hardware to achieve parallelism. It is possible to have overlapping of instruction execution to increase the speed of execution.

1.7 TYPES OF MEMORY

Memory unit is an integral part of any microcomputer system. Its primary purpose is to hold program and data. The main objective of the memory unit design is to enable it to operate at a speed close to that of the processor. Although technology is available to design such a high speed memory, cost is the major limiting factor. To strike a balance between cost and operating speed, a memory system is usually designed using different materials such as solid state, magnetic, and optical materials.

A microcomputer memory can be logically divided into four groups:

(i) Processor memory/register
(ii) Cache memory
(iii) Primary or main memory
(iv) Secondary memory

Processor memory refers to a set of CPU registers. Processor registers are the first set of storage devices available for the programmers to store any data, but they are generally few in number—up to a few tens or hundreds. As these registers are available within the processor, they are the fastest memory registers. The main disadvantage is the cost involved, which restricts the number of registers and their bytes.

Cache memory is the fastest external memory; it is placed close to the processor. The instructions to be executed are placed in the cache memory for access by the processor. These are a few kilobytes in size. Cache memory contains volatile semiconductor RAMs. The processor fetches instructions from the cache memory and if an instruction is not in cache, it refers to the primary memory.

© Oxford University Press. All rights reserved.

Primary memory is the storage area from which all the programs are executed. All the programs and corresponding data for execution must be within the primary memory. The primary memory is much larger than the processor memory and the cache memory but its operating speed is slower. The primary memory in a system varies from few KB to a few MB.

Secondary memory refers to the storage medium for huge files such as program source codes, compilers, operating systems, etc. These are not accessed directly or very frequently by the microprocessor in a computer system. Secondary memory consists of slow devices such as magnetic tapes and optical disks. Sometimes, they are referred to as auxiliary or backup store. Stored information in a magnetic tape or magnetic disk is not lost when power is turned off. Therefore, these storage devices are called non-volatile memories.

Classification of primary memory Primary memory normally includes ROM and RAM, which are further classified as shown in Fig. 1.3. Microprocessor-based systems have at least one RAM and one ROM chip.

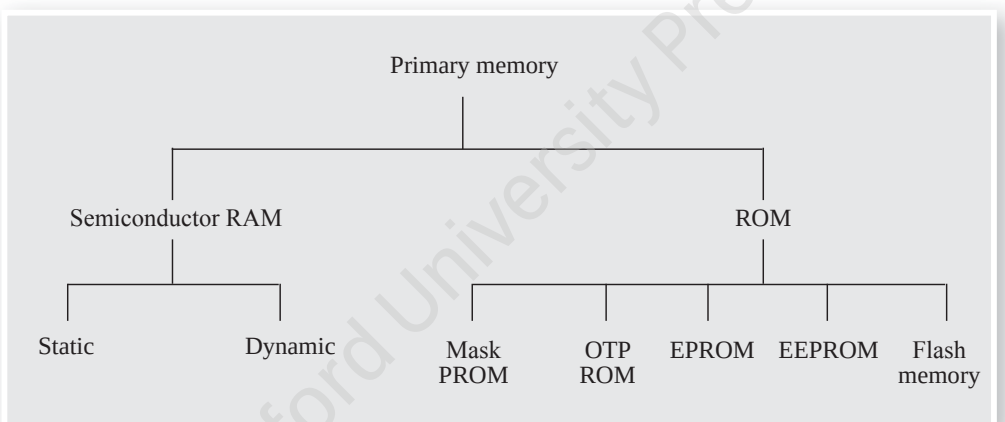

Fig. 1.3 Classification of primary memories

RAM devices allow both reading and writing to their memory cells. In static RAM devices, bits are stored as the status of on/off switches. There are no charges involved and hence, no charges to leak. However, static RAM devices have complex construction and hence larger size per unit storage. So they are more expensive. Static RAMs are comparatively faster and are used in cache memories.

In dynamic RAM devices, the data bits are stored as charge in capacitors. Since capacitor charge has a tendency to leak, these devices need refreshing even when they are powered. However, they have simpler construction and smaller size per unit storage. These devices are less expensive and comparatively slower.

As the name implies, a ROM permits only read access. There are many kinds of ROMs:

(i) Mask programmable ROMs (MPROMs) are custom-made for the customer; their contents are programmed by the manufacturer. Since they are mass produced, they are inexpensive. The customer cannot erase or program it afterwards.

Oxford University Press

© Oxford University Press. All rights reserved.

(ii) Programmable ROMs (PROMs) or one-time programmable (OTP) ROMs are devices that can be programmed by the user in his/her place using special equipments. The main disadvantage of PROMs is that they cannot be erased and reprogrammed.

(iii) Erasable and programmable ROMs (EPROMs) allow the erasure and reprogramming of the content by the user. In an EPROM, programs are entered using electrical impulses and the stored information is erased using ultraviolet rays.

(iv) Electrically erasable PROMs (EEPROMs) or electrically alterable ROMs (EAROMs) allow the users to electrically erase and reprogram its contents. EEPROMs are different from RAMs in that electrical signals are required to erase and program them. EEPROMs require a higher voltage for erasing and programming than the normal 5 V supply.

(v) Flash memory devices are a group of single transistor cell EPPROMs. Cell sizes are about half the size of a two-transistor EEPROM. The operation requires bulk erasure of a large portion of the memory array.

1.8 INPUT AND OUTPUT DEVICES

Input and output devices permit the user to feed data to the computer and retrieve the computed result from it. Sometimes, the input and output devices can communicate among themselves. In general, computer systems have I/O ports; I/O devices are connected to these ports for data transfer. Basically, the ports are digital registers that allow the computer to transfer data between the I/O devices using additional control signals. These control signals allow error-free transfer of data.

The common input device used in almost all systems is the keypad. Microprocessor-based basic microcomputer systems use simple numeric keypads. However, advanced computer systems use keyboards with a large number of keys involving alphabets, numbers, and special characters. Nowadays, a number of optical devices and scanners such as mouse, joystick, and bar code scanners are also being used as input devices. Microcomputer systems also use different types of sensors for data input. These sensors need data converters such as analog to digital converters. Any introductory course on microprocessors should cover the interfacing of data converters, keypads, and switches.

An output device is a device through which the user can receive the results from the computer. The output can be a rapidly changing display or printed material. Other forms of output are sounds and alarms. The simplest output devices, used in almost all microprocessor-based systems and computer systems, are LEDs, seven-segment LED displays, and LCD displays. The advanced video display terminals (either cathode-ray tubes or LCDs) and ink-jet and laser printers are the common output devices nowadays. Some output devices can be used to directly control machineries. Some devices, such as display terminals with touch screen, may provide both input and output. Modems and other network interface cards can also be called output devices as they enable the transmission and reception of data between computers.

Oxford University Press

© Oxford University Press. All rights reserved.

1.9 TECHNOLOGY IMPROVEMENTS ADAPTED TO MICROPROCESSORS AND COMPUTERS

Technological improvements are taking place rapidly in microprocessor, microcomputer, and personal computer systems. Some of these improvements are listed here:

(i) Increase in data bus/address bus width: The processing capability of the microprocessor can be drastically improved by increasing data size. This development can be seen clearly from the advancements in microprocessors (Section 1.5).

(ii) Increase in speed: As the data to be processed by the microprocessors and computers increased in volume, it became necessary to increase the speed of the processor. With high speed processors, the user can get results quickly, even with large data volumes.

(iii) Reduction in size and increase in capability: The trend in microprocessor technology is to include a large number of peripherals such as memory and I/O ports within a single chip. Microcontrollers are manufactured in this fashion. In addition, developments in large scale integration have led to the manufacture of small microprocessor chips with large built-in peripherals. Processors with a large amount of flash memory are now available in the market.

(iv) Development of external peripherals: The use of computers in all fields have resulted in the development of many fast and advanced peripheral devices. For example, the application of microprocessors in medicine has resulted in the development of many handheld electronic devices with specialized input sensors, output printers, etc. Faster peripherals can increase the speed of processor execution and provide a good user interface.

(v) Increase in memory unit size and speed: The developments in IC technology have led to a reduction in the size of the memory units and an increase in memory speed. This reduces the memory access time of the processor and results in higher speed of execution. More amount of memory per unit area is possible.

(vi) Microprocessors are largely used in handheld devices operated from a battery source. This has resulted in research on the reduction of power consumption in microprocessor chips. As power consumption is reduced, these devices work for more time once the batteries are fully charged. There are many devices operating at 3.3 V or even lower voltages and have low power consumption.

1.10 INTRODUCTION TO 8085 PROCESSOR

The microprocessor is a semiconductor device consisting of electronic logic circuits manufactured using either large-scale integration (LSI) or very large-scale integration (VLSI) technique. It basically contains registers, an arithmetic and logic unit, flip-flops, and timing and control circuits. All microprocessors work using Von–Neumann architecture. In this architecture, the CPU or the processor

Oxford University Press

© Oxford University Press. All rights reserved.

fetches instructions from the memory, decodes it (i.e., interprets the nature of the instruction/command and develops clock-synchronized steps for execution), generates appropriate control signals, and finally executes it. The program is stored in consecutive memory locations. The execution steps are repeated for all the instructions of the program until the execution is terminated by hardware or software. The data required may be taken either from memory or from input ports; the results of the program may be either stored in the memory or transferred out through output ports.

A program is a list of instructions for the microprocessor to execute. Before the start of execution, the complete program must be stored in the memory. Let us assume that the starting address of the stored program is 8800H. While running the program, the microprocessor must be directed to 'go' from 8800H. Once it has executed the instruction in 8800H, it goes to the next address 8801H (assuming single-byte instructions) and so on until it reaches the end of the program.

Intel 8085 is an 8-bit microprocessor manufactured by Intel Corporation and is usually called a general-purpose 8-bit processor. It is upward compatible with microprocessor 8080, which was Intel's earlier product. There are several faster versions of the 8085 microprocessor such as 8085AH, 8085AH-1, and 8085AH-2.

A microprocessor system consists of three functional blocks—central processing unit (CPU), input and output units, and memory units, as shown in Fig. 1.4. The CPU contains several registers, an arithmetic and logic unit (ALU),

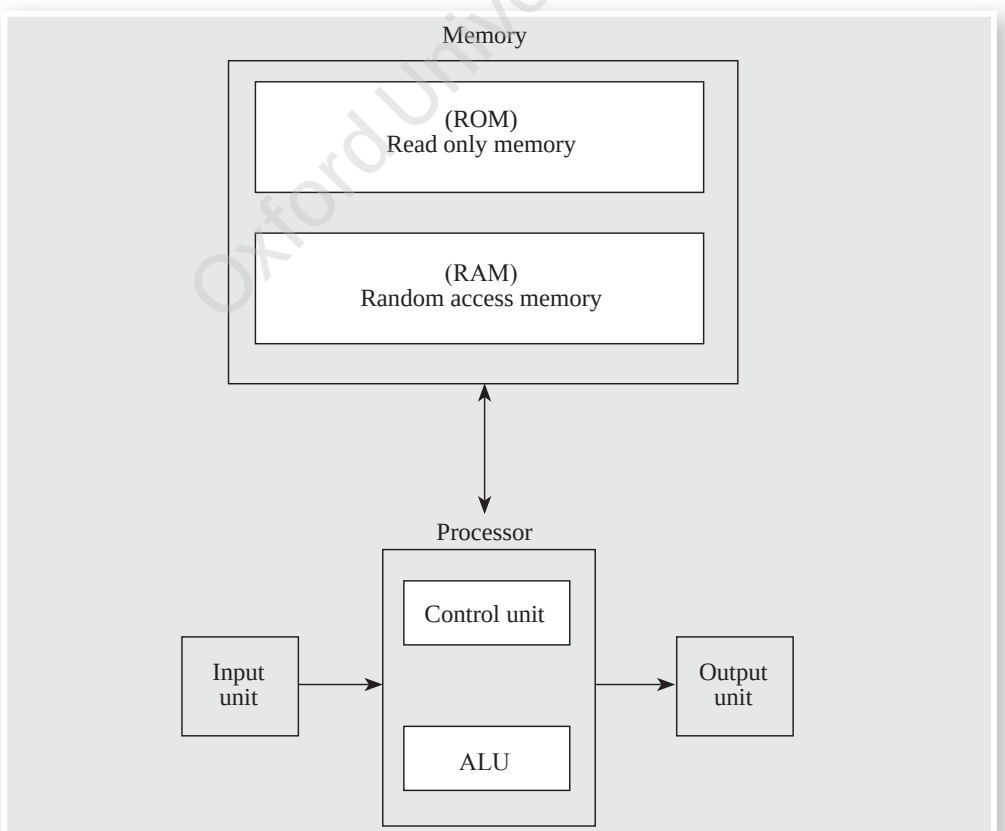

Fig. 1.4 A microprocessor system

© Oxford University Press. All rights reserved.

and a control unit. The function of ALU, as the name implies, is to perform arithmetic and logical operations. The control unit translates the instructions and executes the desired task.

1.11 ARCHITECTURE OF 8085

The block diagram explaining the architecture of Intel 8085 microprocessor is shown in Fig. 1.5. It is generally available as a 40-pin IC package and uses +5 V for power. It can run at a maximum frequency of 3 MHz. The modified versions of the 8085 processor have these minimum common features and functional similarities.

The 8085 is called an 8-bit processor since its data length and data bus width is eight bits. It has an addressing capability of 16 bits, i.e., it can address $2^{16} = 64$ KB of memory (1 KB = 1024 bytes). The processor contains five functional units:

(i) Arithmetic and logic unit
(ii) General-purpose registers
(iii) Special-purpose registers
(iv) Instruction register and decoder
(v) Timing and control unit

1.11.1 Arithmetic and Logic Unit

ALU is the circuitry that performs the actual numerical and logical operations. Addition (ADD), subtraction (SUB), increment (INR), decrement (DCR), and comparison (CMP) are the arithmetic operations possible in the 8085

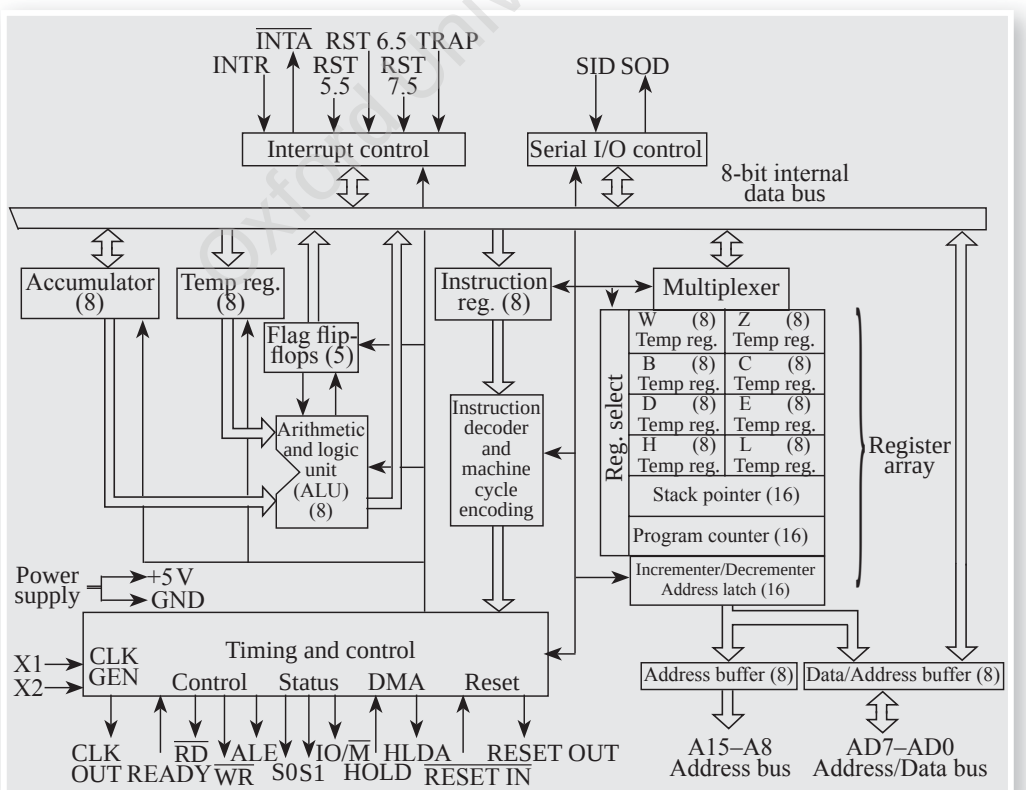

Fig. 1.5 Functional block diagram of Intel 8085

© Oxford University Press. All rights reserved.

microprocessor. The possible logical operations are AND (AND), OR (OR), exclusive OR (EXOR), complement (CMA), etc.

The ALU of the 8085 processor is called accumulator-oriented ALU as one of the data used in arithmetic and logic operations must be stored in the accumulator. The other data is taken from a memory location or register. The results of the arithmetic and logical operations are stored in the accumulator. If the operation needs only one data, that data must be stored in the accumulator.

1.11.2 General-purpose Registers

A register is a collection of eight D-type flip-flops with parallel-in and parallel-out operation. A flip-flop can only store one bit at a time. Therefore, to handle eight bits at a time, eight flip-flops are required and hence the term 8-bit register. Though the registers are all storage areas inside the microprocessor, they differ in the purpose of storage. The general-purpose registers are used to store only the data that is being used by the program under execution and the results obtained from it. These general-purpose registers are user accessible through programs.

Registers B, C, D, E, H, and L are the general-purpose registers in the 8085, as shown in Fig. 1.6. They can also be called scratchpad registers. In almost all arithmetic and logical operations, these registers are used as the second operands, the first operand being the accumulator (A). The general-purpose registers are all 8-bit registers but they can be handled as 16-bit registers as well. This can be achieved by combining the register pairs B and C, D and E, and H and L to perform 16-bit operations. They are then named register pairs BC, DE, and HL, respectively.

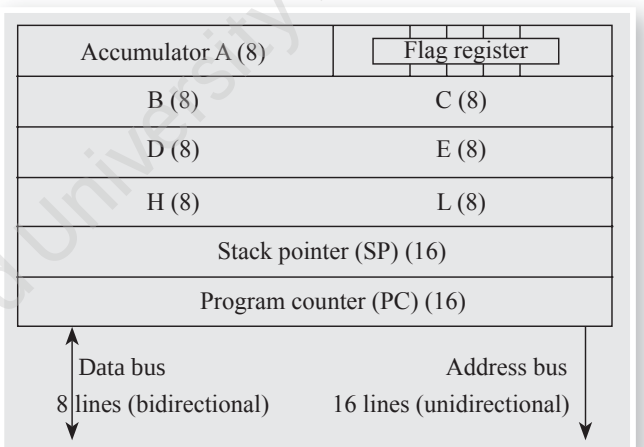

Fig. 1.6 Registers of Intel 8085

Among these pairs, HL has a special significance. A few memory-related instructions of the 8085 (refer instruction set) use the HL pair as a memory pointer. For example, the instruction MOV A, M transfers the content of the memory location to which the HL pair is pointing, to the accumulator. The HL pair is pre-loaded with the memory address in which data is available.

1.11.3 Special-purpose Registers

There are also special-purpose registers that are dedicated to a specific function. The accumulator, flag register, program counter (PC), and stack pointer (SP) constitute the special registers in the 8085 microprocessor.

1.11.3.1 *Accumulator*

The accumulator is an 8-bit register; it is a part of the ALU and is the most

© Oxford University Press. All rights reserved.

important register. It is used to store 8-bit data and to perform arithmetic and logical operations. The output of an operation is also stored in the accumulator. The accumulator is identified as register A in the instruction set of the 8085. The programmer can use it at any time to store an 8-bit binary number. Being only eight bits long, it can only hold one byte at a time. Any previous data stored in this register will be overwritten as soon as new data is stored. The 8085 microprocessor communicates with input/output devices only through the accumulator.

1.11.3.2 *Flag Register*

This is a special 8-bit register. Each bit of the flag register is quite independent of the others. In all other registers, each bit is part of a single binary byte value and hence each bit would have a numerical value. The flag is an 8-bit register used to indicate the status of a recent arithmetic or logical operation. It may be set or reset after an arithmetic or logical operation according to the condition of the processed data. The five flag bits are zero (Z), carry (CY), sign (S), parity (P), and auxiliary carry (AC); their bit positions in the flag register are shown in Fig. 1.7. The remaining three bits (D1, D3, and D5) of the flag register remain unassigned; they are marked with an X to show that they are not used and are *don't cares*.

S	Z	X	AC	X	P	X	CY
D7	D6	D5	D4	D3	D2	D1	D0

Fig. 1.7 Flag register

Any flag register bit is said to be 'set' when its value is 1 and 'cleared' when its value is 0. The most commonly used flags are zero, carry, and sign. AC flag cannot be accessed externally.

Sign flag (S) The sign flag is just a copy of the bit D7 (most significant bit—MSB) of the accumulator. A negative number has a 1 in bit 7 and a positive number has a 0 in 2's complement representation. This flag indicates the sign of the number. (It may be recalled that signed magnitude numbers use 1 to indicate a negative number and 0 to indicate a positive number.) This flag can be used in signed arithmetic operations.

Zero flag (Z) The zero flag is set if an arithmetic operation results in a zero. It sets, i.e., it changes to binary 1 if the result in the accumulator is zero; if not, it remains reset, i.e., at binary 0.

Carry flag (C) The carry flag is set when a carry is generated in the process of an arithmetic operation in the accumulator. When addition is carried out, it sometimes results in a ninth bit being carried over to the next byte. The C flag copies the value of the carry, which is an extra bit, from D7. It also reflects the value of the borrow in subtractions.

Auxiliary carry flag (AC) The auxiliary carry flag is set when an auxiliary carry is generated in the process of an arithmetic operation in the accumulator, i.e., when a carry results from bit D3 and passes on to D4 (from the lower nibble to the higher nibble). This carry is also called half-carry. It may also occur in the process of a subtraction operation. In other words, this flag is set if the subtraction operation results in borrowing from the higher nibble.

© Oxford University Press. All rights reserved.

Parity flag (P) The parity flag is set if the content of the accumulator after an arithmetic operation has an even number of 1s. Otherwise, the parity flag is reset. It is set for operation in the even parity mode.

1.11.3.3 *Program Counter*

Program counter (PC) is a 16-bit register that always points to the address of the next instruction to be executed. In other words, this register is used to sequence the execution of the instructions. After execution of every instruction, the content of the memory location indicated by the PC is moved to the instruction register and the PC is loaded with the next address. It keeps track of a program by counting the memory address from which the next byte is to be fetched, and hence the name program counter.

1.11.3.4 *Stack Pointer*

Stack is an array of memory locations organized in last-in, first-out (LIFO) or first-in, last-out (FILO) fashion. It is accessed using a 16-bit pointer register called stack pointer (SP), which holds the address of the memory location of the top of the stack. The programmer can reserve and allocate a series of RAM locations to be used as a stack and accordingly initialize the stack pointer. The range of stack memory locations must be chosen carefully so that it does not affect the program space. In all microprocessor-based systems, the stack is mainly used to store the return address of the main program when a subroutine is called. While the programmer uses the stack for storage and retrieval of data, the microprocessor uses the stack during subroutine calls. Care must be taken by the programmer to ensure that the data stored in the stack is retrieved properly, so that the data stored in the stack by the processor is not affected.

1.11.4 Instruction Register and Decoder

It is an 8-bit register that temporarily stores the instructions drawn from memory locations, before their actual execution. The content of the register is decoded by the decoder circuitry, where the nature of the operation to be performed is decided (interpreted). In addition, there are two temporary registers W and Z, which are controlled internally and not available for user access.

1.11.5 Timing and Control Unit

The timing and control unit gets commands from the instruction decoder and issues signals on the data bus, address bus, and control bus. The following sections explain the operation of the various buses and the timing.

A typical microprocessor communicates with memory and input/output devices using buses. There are three types of buses—the address bus, the data bus, and the control bus.

1.11.5.1 *Data Bus*

The microprocessor performs its functions using wires or lines called buses. For example, an 8-bit microprocessor normally uses eight wires to carry data between the microprocessor and the memory. To make their representation simple, the data wires with common functions are grouped together and referred to as the data bus.

Oxford University Press

© Oxford University Press. All rights reserved.

The data bus (D0–D7) is a two-way bus carrying data around the system. Information going into the microprocessor and results coming out of the microprocessor are through this data bus. It is used for transfer of binary information between the microprocessor, memory, and peripherals. The lower group of eight address lines A0–A7 is multiplexed with the data bus in order to reduce the pin count. Therefore, the multiplexed lower group of address lines and data lines is more generally denoted as AD0–AD7.

1.11.5.2 *Address Bus*

The address bus carries addresses and is a one-way bus from the microprocessor to the memory or other devices. It is a group of sixteen unidirectional lines that allows flow of address from the processor to its peripheral devices. Each peripheral and memory location is identified by a 16-bit binary number called address. It follows that the maximum number of memory locations that can be addressed by the 8085 processor is 2^{16} bytes = 64 KB. Its basic function is to identify a peripheral or memory location.

The address bus lines are generally identified as A0–A15. The address bus has eight higher-order address lines (A8–A15), which are unidirectional. The lower-order eight lines (A0–A7) are multiplexed (time-shared) with the eight data bits (D0–D7) and hence, they are bidirectional. When the instruction is executed, these lines carry the address bits during the early part, and the eight data bits during the later part. To separate the address from the data, a latch is used externally to save the address before the function of the bits changes.

1.11.5.3 *Control Bus*

The control bus carries control signals that are partly unidirectional and partly bidirectional. For a microprocessor to function correctly, these control signals are vital. The control bus typically consists of a number of single lines that coordinate and control microprocessor operations. For example, a read/write control signal will indicate whether memory is being written into or read from. Thus, they are individual lines that provide a pulse to indicate the operation of the microprocessor. In fact, the microprocessor generates specific control signals for every operation, which in turn are used to identify the type of device the processor intends to communicate with. The following points describe the control and status signals of the 8085 processor:

(i) ALE (output): Address Latch Enable is a pulse that is provided when an address appears on the AD0–AD7 lines, after which it becomes 0. This signal can be used to enable a latch to save the address bits from the AD lines, thereby de-multiplexing the address bus and data bus.

(ii) $\overline{\text{RD}}$ (active low output): The Read signal indicates that data are being read from the selected I/O or memory device and that they are available on the data bus.

(iii) $\overline{\text{WR}}$ (active low output): The Write signal indicates that the data on the data bus are to be written into a selected memory or I/O location.

(iv) IO/$\overline{\text{M}}$ (output): It is a signal that distinguishes between a memory operation and an I/O operation. An active low on this signal shows it is a memory

© Oxford University Press. All rights reserved.

operation (IO/$\overline{M}$ = 0) and a high on this line indicates an I/O operation (IO/$\overline{M}$ = 1).

(v) S1 and S0 (output): These are status signals used to specify the kind of operation being performed. The status signals combine with I/O signals to govern various operations; they are listed in Table 1.2. If both S0 and S1 are low, the operation of the processor tends to halt. If S0 is low and S1 is high, the processor reads data. If S0 is high and S1 is low, the processor writes data onto a memory or I/O device. If both S0 and S1 are high, the fetch operation is performed.

Table 1.2 Status signals and associated operations

S1	S0	States
0	0	Halt
0	1	Write
1	0	Read
1	1	Fetch

The schematic representation of the 8085 bus structure, shown in Fig. 1.8, explains how the movement of data within the computer is accomplished by a series of buses. Address information, data, and control signals have to be carried around inside the microprocessor as well as in the external system. Hence, the buses are present both internally and externally.

(vi) Interrupts: These signals are used to make the microprocessor respond to high priority externally initiated signals. When an interrupt signal is detected by the processor, it suspends the execution of the current program and executes the program corresponding to the interrupt signal instead. Five interrupt signals (INTR, RST 5.5, RST 6.5, RST 7.5, and Trap) are available to facilitate the processor to receive and acknowledge the interrupt call of peripherals. The 8085 processor accepts three more externally initiated signals—$\overline{\text{RESET IN}}$, Hold, and Ready as inputs. The following points explain these signals in brief:

(a) INTR (input): It is a general-purpose interrupt request signal. It is an active high signal.

(b) $\overline{\text{INTA}}$ (output): It is used to acknowledge an interrupt. It is an active low signal.

(c) Restart interrupts (input): These are vectored interrupts that transfer the

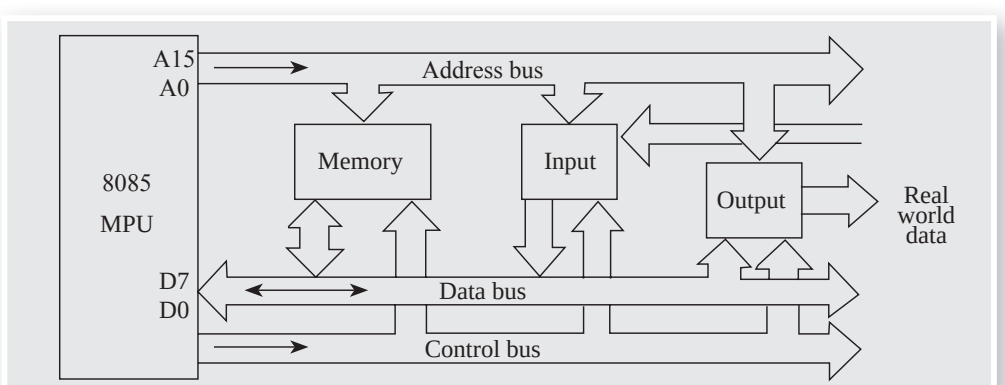

Fig. 1.8 Schematic representation of the 8085 bus structure

© Oxford University Press. All rights reserved.

program control to specific memory locations. They have higher priority than INTR interrupts. The priority order is RST 7.5, RST 6.5, and RST 5.5.

(d) Trap (input): It is a non-maskable interrupt, i.e., it cannot be stopped or overridden by any command. It has the highest priority among all 8085 interrupts.

(e) $\overline{\text{RESET IN}}$ (input): When the signal on this pin goes low the program counter is set to zero and the processor is reset. It is an active low signal.

(f) RESET OUT (output): This signal can be used to reset other devices that are connected to the processor. It is an active high signal.

(g) Hold (input): This signal indicates that a peripheral such as a direct memory access (DMA) controller is requesting the use of the address and data buses.

(h) HLDA (output): It is an acknowledge signal that is sent in response to the Hold request. During the Hold state, the peripheral (I/O) devices get control over the data and address buses for data transfer to and from memory. This operation is called direct memory access (DMA). DMA is useful when high-speed peripherals want to transfer data to and from memory. The processor does not intervene during this period.

(i) Ready (input): It is a signal that serves to delay the microprocessor read/write signals until a slow-responding peripheral is ready to send or accept data. If this signal goes low, then the processor is allowed to wait for an integral number of clock cycles until Ready becomes high. The Ready signal must be synchronized with the processor clock.

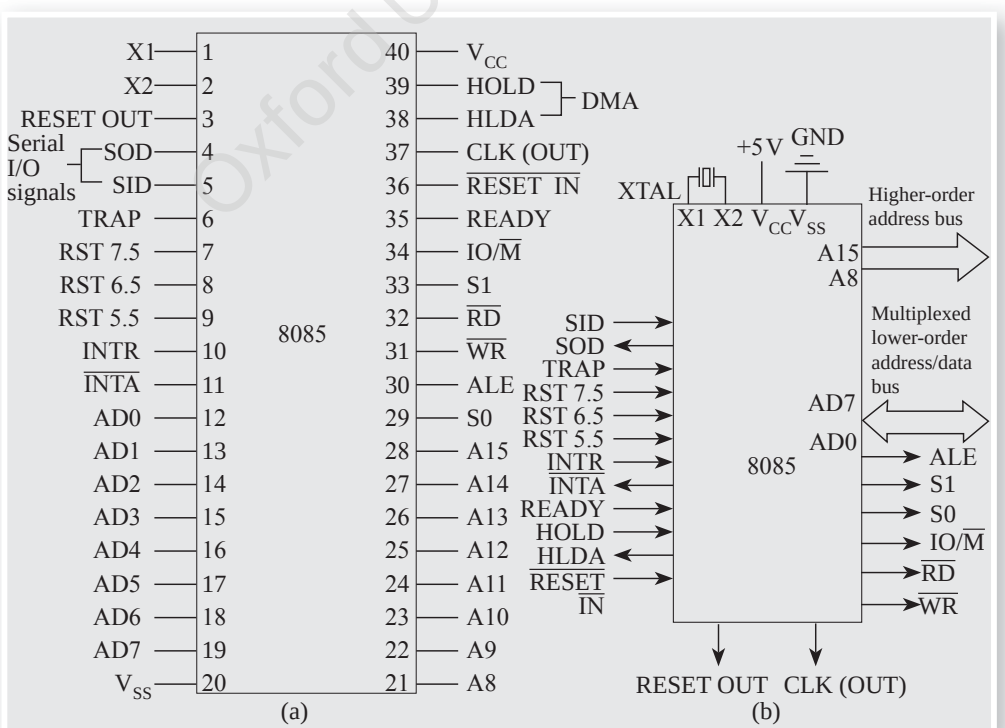

Fig. 2.7 8085 details (a) Pin diagram (b) Signal groups

© Oxford University Press. All rights reserved.

The typical pin layout and signal groups of the 8085 microprocessor are shown in Figs 1.9 (a) and 1.9 (b), respectively. The 8085 is available with 40 pins as a dual in-line package (DIP).

Intel 8085 has 40 pins, operates at 3 MHz clock frequency, and requires +5 V for power supply.

1.11.5.4 *Serial I/O Signals*

There are two signals to implement serial transmission. They are serial input data (SID) and serial output data (SOD). The data bits are sent over a single line, one bit at a time, in serial transmission.

(i) SID (input): The bit data on this line is loaded in the seventh bit of the accumulator whenever a RIM instruction is executed.

(ii) SOD (output): The output SOD is set or reset as specified by the SIM instruction.

RIM and SIM instructions have been explained in detail in Chapter 5.

1.11.5.5 *Power Supply and System Clock*

The following pins are available in the 8085 chip to provide power and clock signal to the processor:

(i) X1, X2 (input): A microprocessor needs a square wave (clock) signal to ensure that all internal operations are synchronized. A crystal or R–C or L–C network is connected to these two pins. The crystal frequency is internally divided by two to give the operating system frequency. There are three advantages in increasing the frequency of a crystal—as frequency increases, the crystal size becomes smaller, and the crystal becomes lighter and cheaper. Therefore, clock circuits include a divide-by-two circuit so that a double-frequency crystal can be used. So, to run the microprocessor at 3 MHz, a 6 MHz crystal should be connected to the X1 and X2 pins. The crystal is preferred as a clock source because of its high stability, large Q (quality factor), and absence of frequency drifting with aging. Without a clock signal, the microprocessor cannot execute any program.

(ii) CLK (output): This output clock pin is used to provide the clock signal to the rest of the system.

Power supplies: V_{CC}—+5 V supply; V_{SS}—ground reference.

1.12 MICROPROCESSOR INSTRUCTIONS

Every microprocessor has its own instruction set. Based on the design of the ALU and the decoding unit, microprocessor manufacturers generally list out the instructions for every microprocessor manufactured. The instruction set consists of both assembly language mnemonics and the corresponding machine code.

The purpose of the instruction set is to facilitate the development of efficient programs by the users. The instruction set is based on the architecture of the processor. So to understand the instruction set of a processor, it is necessary to understand the basic architecture of the microprocessor and the user-accessible

Oxford University Press

© Oxford University Press. All rights reserved.

registers in it. An *instruction* is a bit pattern that is decoded inside a microprocessor to perform a specific function. The assembly language mnemonics are the codes for these binary patterns so that the user can easily understand the functions performed by these instructions. The entire group of instructions that a microprocessor can handle is called its *instruction set*; this determines the microprocessor's functionality. The Intel 8085 processor has its own set of instructions listed both in mnemonics and machine code, also called as object code. As the 8085 is an 8-bit processor, the machine codes for the instructions are also 8 bits wide.

The syntax for 8085 instructions may contain one or more of the following notations:

R = 8-bit register (A, B, C, D, E, H, and L)
Rs = Source register
Rd = Destination register } (A, B, C, D, E, H, and L)
Rp = Register pair (BC, DE, HL, and SP)
P = Port address (8-bit binary number or two hex digits)
8-bit = 8-bit data or two hex digits
16-bit = 16-bit data/address or four hex digits
( ) = Contents of

1.13 CLASSIFICATION OF INSTRUCTIONS

Microprocessor instructions can be classified based on parameters such as functionality, length, and operand addressing.

1.13.1 Based on Functionality

Based on the functionality, the instructions are classified into the following five categories:

(i) Data transfer (copy) operations
(ii) Arithmetic operations
(iii) Logical operations
(iv) Branching operations
(v) Machine control operations

1.13.1.1 *Data Transfer (Copy) Operations*

This group of instructions copies data from a location called source register to another location called destination register. Generally, the contents of the source register are not modified. Although the term *data transfer* is used for the copy operation, it is misleading because it implies that the contents of the source memory location are destroyed. The various types of data transfer are listed in Table 1.3 along with examples of each type.

Table 1.3 Types of data transfer

Type	Example
Transferring data between one register and another	MOV A, D—Copies the content of register D to the accumulator
Storing a data byte in a register or memory location	MVI C, 66H—Loads register C with the data 66H

(Contd)

© Oxford University Press. All rights reserved.

Table 1.3 Types of data transfer *(Contd)*

Type	Example
Transferring data between a memory location and a register	LDA 8800H—Loads the contents of memory location 8800H in the accumulator
Transferring data between an I/O device and the accumulator	IN PORT1—Transfers data from an input device to the accumulator

1.13.1.2 *Arithmetic Operations*

Arithmetic operations include addition, subtraction, increment, and decrement. As the 8085 has an accumulator-oriented ALU, one of the data used in the arithmetic operations is stored in the accumulator; the result is also stored in the accumulator. Arithmetic and logical operations cannot be executed without the accumulator.

Addition (ADD) The addition instructions of the 8085 add the contents of a register or memory location with the contents of the accumulator. The result is stored in the accumulator. The Intel 8085 instruction set supports two types of addition instructions—with and without addition of the carry flag content to the least significant bit of the numbers. The instruction set also supports 16-bit addition, i.e., the content of the HL register pair can be added to that of another register pair and the result stored in the HL register pair.

Subtraction (SUB) The instruction set of the 8085 supports two types of subtraction—with borrow and without borrow. Like addition, the subtraction operation also uses the accumulator as reference, i.e., it subtracts the content of a register or memory location from that of the accumulator and stores the result in the accumulator.

Increment/Decrement These operations can be used to increment or decrement the contents of any register, register pair, or memory location. Unlike the arithmetic and logical operations, the increment and decrement operations need not be based upon the accumulator.

1.13.1.3 *Logical Operations*

Logical instructions are also accumulator-oriented, i.e., they require one of the operands to be placed in the accumulator. The other operand can be any register or memory location. The result is stored in the accumulator. The operations that use two operands are logical AND, OR, and EXOR. The operation that uses a single operand (i.e., the accumulator) is the logical complement or NOT operation.

The instruction set of the 8085 supports rotation of the data stored in accumulator. The data can be rotated left or right, through the carry or without the carry.

The most important 8085 instruction is the compare instruction. This instruction is used to compare register or memory content with the accumulator content. The result of comparison such as equal to, greater than, or less than is reflected in the flag register bits.

1.13.1.4 *Branching Operations*

Branching instructions are important for programming a microprocessor. These

© Oxford University Press. All rights reserved.

instructions can transfer control of execution from one memory location to another, either conditionally or unconditionally. Branching can take place in the following two ways:

(i) Execution control cannot return to the point of branching. Example: Jump instructions
(ii) Execution control can return to the point of branching, which is stored by the 8085. Example: Subroutine call instructions

1.13.1.5 *Machine Control Operations*

These instructions can be used to control the execution of other instructions. They include halting the operation of the microprocessor, interrupting program execution, etc. Detailed explanations for 8085 instructions are given in Section 1.14.

1.13.2 Based on Length

Based on the length of the machine language code, 8085 instructions can be classified into the following three types:

(i) One-byte instructions (ii) Two-byte instructions
(iii) Three-byte instructions

Assembly language instructions should be converted into machine code for storage and execution by the processor. So the length of the machine language code instructions determines the length of the program. This in turn determines the amount of memory required for the program.

1.13.2.1 *One-byte Instructions*

Instructions that require only one byte in machine language are called one-byte instructions. These instructions just have the machine code or opcode alone to represent the operation to be performed. The common examples are the instructions that have their operands within the processor itself. Some examples of one-byte instructions are given in Table 1.4. Even though the instruction ADD M adds the content of a memory location to that of the accumulator, its machine code requires only one byte.

Table 1.4 One-byte instructions

Opcode	Operand	Machine code/Opcode/ Hex code
MOV	A, B	78
ADD	M	86
XRA	A	AF

Let us now understand the instruction MOV Rd, Rs. This instruction copies the contents of source register Rs to destination register Rd. (Rd ← Rs)

It is coded as 01dddsss. Here, ddd is the binary code of one of the seven general-purpose registers that is the destination of the data and sss is the binary code of the source register.

Example:

```
MOV A, B  (coded as 01111000 = 78H)
```

1.13.2.2 *Two-byte Instructions*

Instructions that require two bytes in machine code are called as two-byte instructions. The first byte of the two-byte instructions is the opcode, which

© Oxford University Press. All rights reserved.

specifies the operation to be performed. The second byte is the 8-bit operand, which is either an 8-bit number or an address. Some common examples of two-byte instructions are listed in Table 1.5.

Table 1.5 Two-byte instructions

Opcode	Operand	Machine code/Opcode/Hex code	Byte description
MVI	A, 7FH	3E	First byte
		7F	Second byte
ADI	0FH	C6	First byte
		0F	Second byte
IN	40H	DB	First byte
		40	Second byte

The instruction is stored in two consecutive memory locations.

MVI R, data—(R ← data)

Example:

```
MVI A, 32H (coded as 3E 32 in two contiguous bytes)
```

This is an example of immediate addressing.

The following two instructions are also examples of two-byte instructions:

(i) ADI data (A ← A + data)

(ii) OUT port (where port is an 8-bit device address. (Port) ← A) Since the byte is not the data itself, but points directly to where it is located, this is called direct addressing. For a detailed account of addressing modes, see Section 1.13.3.

1.13.2.3 *Three-byte Instructions*

Instructions that require three bytes in machine code are called three-byte instructions. In 8085 machine language, the first byte of the three-byte instructions is the opcode which specifies the operation to be performed. The next two bytes refer to the 16-bit operand, which is either a 16-bit number or the address of a memory location. Some common examples of three-byte instructions are listed in Table 1.6.

Table 1.6 Three-byte instructions

Opcode	Operand	Machine code/Opcode/Hex code	Byte description
JMP	9050H	C3	First byte
		50	Second byte
		90	Third byte
LDA	8850H	3A	First byte
		50	Second byte
		88	Third byte
LXI	H, 0520H	21	First byte
		20	Second byte
		05	Third byte

Oxford University Press

© Oxford University Press. All rights reserved.

The instruction LXI Rp, 16-bit data can be explained as follows:

Rp is one of the pairs of registers BC, DE, or HL, which are used as 16-bit registers. The two data bytes are to be stored as a 16-bit number in L and H in sequence. LXI H, 0520H is coded as 21H 20H 05H in three bytes. (This is an example of immediate addressing.)

In executing the instruction LDA addr, the accumulator is loaded with the memory content of the address given in the instruction. Addr is a 16-bit address. LDA 8850H is coded as 3AH 50H 88H. (This is an example of direct addressing.)

1.13.3 Addressing Modes in Instructions

Every instruction in a program has to operate on data. The process of specifying the data to be operated on by the instruction is called *addressing*. Efficient software development for the microprocessor requires complete familiarity with the addressing mode employed for each instruction. For example, the instructions MOV B, A and MVI A, 82H are used to copy data from a source to a destination. In these instructions, the source can be a register or an 8-bit number (00H to FFH); the destination is a register. The source and destination are operands. The various formats for specifying operands are called *addressing modes*. The 8085 has the following five types of addressing:

(i) Immediate addressing
(ii) Memory direct addressing
(iii) Register direct addressing
(iv) Indirect addressing
(v) Implied or implicit addressing

1.13.3.1 *Immediate Addressing*

Immediate addressing transfers the operand given in the instruction—a byte or word—to the destination register or memory location. The operand is part of the instruction. The format for immediate addressing is given in Fig. 1.10.

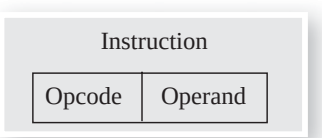

Fig. 1.10 Format of immediate addressing

Example:

```
MVI A, 9AH
```

(a) The operand is part of the instruction.
(b) The operand is stored in the register mentioned in the instruction.

Example:

```
ADI 05H
```

(a) Add 05H to the contents of the accumulator.
(b) 05H is the operand.

Immediate addressing has no memory reference to fetch data. It executes faster, but has limited data range.

1.13.3.2 *Memory Direct Addressing*

Memory direct addressing moves a byte or word between a memory location and register. The memory location address is given in the instruction. The instruction set does not support memory-to-memory transfer. Memory direct addressing is illustrated in Fig. 1.11.

© Oxford University Press. All rights reserved.

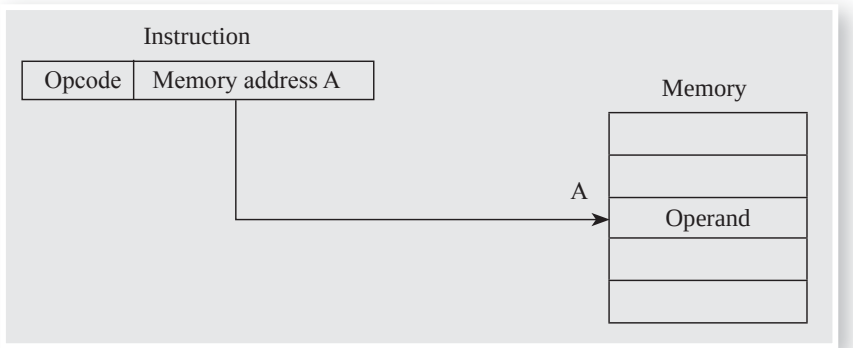

Fig. 1.11 Format of memory direct addressing

Example:

```
LDA 850FH
```

This instruction is used to load the contents of the memory location 850FH in the accumulator.

Example:

```
STA 9001H
```

This instruction is used to store the contents of the accumulator in the memory address 9001H.

In these instructions, the memory address of the operand is given in the instruction.

Direct addressing is also used for data transfer between the processor and input/output devices. For example, the IN instruction is used to receive data from the input port and store it in the accumulator; the OUT instruction is used to send the data from the accumulator to the output port.

Example:

```
IN 00H and OUT 01H
```

1.13.3.3 *Register Direct Addressing*

Register direct addressing transfers a copy of a byte or word from the source register to the destination register. The operand is in the register named in the instruction. It executes very fast, has very limited register space, and requires good assembly programming. The operand is within in the processor itself; so the execution is faster. Register direct addressing is illustrated in Fig. 1.12.

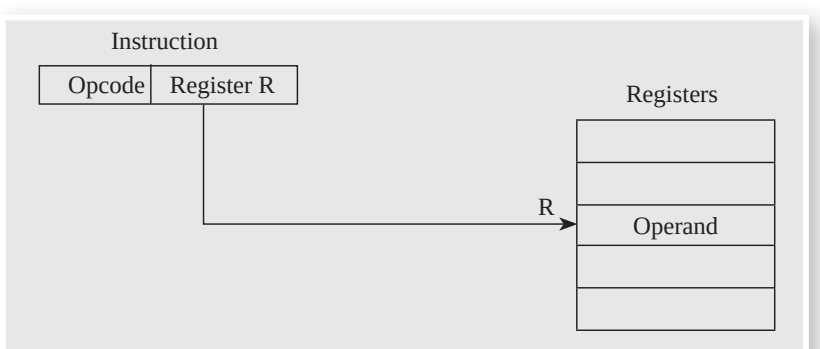

Fig. 1.12 Format of register direct addressing

© Oxford University Press. All rights reserved.

Example:

```
MOV Rd, Rs
MOV B, C
```

It copies the contents of register C to register B.

Example:

```
ADD B
```

It adds the contents of register B to the accumulator and saves it in the accumulator.

1.13.3.4 *Indirect Addressing*

Indirect addressing transfers a byte or word between a register and a memory location. The address of a memory location is stored in a register and that register is specified in the instruction. This is illustrated in Fig. 1.13.

In indirect addressing, the effective address is calculated by the processor using the contents of the register specified in the instruction. This type of addressing employs several accesses—two accesses to retrieve the 16-bit address and a further access (or accesses) to retrieve the data which is to be loaded in the register.

Example:

```
MOV A, M
```

Here, the data is in the memory location pointed to by the contents of the HL pair. The data is moved to the accumulator.

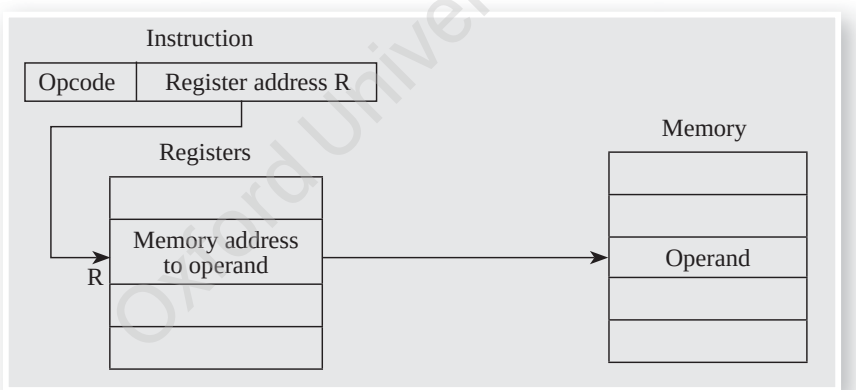

Fig. 1.13 Format of indirect addressing

1.13.3.5 *Implied or Implicit Addressing*

In implied addressing mode, the instruction itself specifies the data to be operated upon. For example, CMA complements the contents of the accumulator. No specific data or operand is mentioned in the instruction.

1.14 INSTRUCTION SET OF 8085

The 8085 microprocessor instruction set has 74 operation codes and 246 instructions. It is compatible with that of its predecessor, the 8080A, but has two additional instructions—SIM (set interrupt mask) and RIM (read interrupt mask)—related to serial I/O. The complete instruction set is listed in Appendix 1 with additional information such as number of clock states required for execution and the flags affected.

© Oxford University Press. All rights reserved.

1.14.1 Format of Assembly Language Instructions and Programs

Assembly language programs are written for performing specific functions, converted into machine language code, and then stored in the memory of the microprocessor-based system. The conversion of an assembly language program into machine language code is called assembling; the application that performs this task is called assembler. This conversion or assembling can also be done manually by the programmers. To facilitate the process of assembling, the assembly language programs are written in a specific format as shown in Fig. 1.14.

Memory address	Machine code/Opcode	Label	Mnemonics with operands	Comments

Fig. 1.14 Format for writing assembly language programs

In general, the assembly language mnemonics with their operands are written first. The address where the instructions are stored is given a dummy name called label. The purpose of labels is to give the correct branch addresses in instructions. Labels are separated from mnemonics with a colon.

The comments column is essential for any program as it helps the programmer understand the logic of the program at any point in time. Without comments, it is difficult to understand an assembly language program. Comments are separated from the mnemonics with a semicolon.

The first two columns correspond to the physical memory address and the actual machine code. These two columns are filled in after completing the assembly language programming. These columns must contain only binary numbers, but for easy understanding, hexadecimal numbers are used. For manual assembling, these two columns are filled in by the programmer. An assembler can generate these columns automatically.

An example of the assembly language program format is given in Table 1.7.

Table 1.7 Sample assembly language program

Memory address	Machine code/ Opcode	Label	Mnemonics with operands	Comments
8000 8001 8002	3E 5F	START:	MVI A, 5FH	; Load data in the accumulator. ; Address of the next memory location

The instruction in Table 1.7 moves the data 5FH to the accumulator.

1.14.2 Data Transfer Instructions

Data transfer instructions are used to transfer data between two registers in the microprocessor or between a peripheral device and the microprocessor. Some instructions and their features are given in the following points. The complete list with explanations is given in Table 1.8.

(i) MVI instruction is used for storing 8-bit data in a microprocessor register.
(ii) LXI instruction is used for storing 16-bit data in a register pair.

Oxford University Press

© Oxford University Press. All rights reserved.

(iii) In direct addressing mode, MOV instruction is used for data transfer between registers. In indirect addressing mode, MOV is used for data transfer between a memory location and a register. If the instruction has M in the operand field, the memory location pointed to by the HL pair is considered for data transfer.

Table 1.8 Data transfer instructions

Mnemonics	Tasks performed on execution	Addressing mode	Instruction length	Example
MVI R, 8-bit	Moves the 8-bit data to the register	Immediate	Two bytes	MVI B, 3FH
LXI Rp, 16-bit	Loads the 16-bit data in the register pair	Immediate	Three bytes	LXI B, 5AF3H
MOV Rd, Rs	Copies the data from the source register to the destination register	Register direct	One byte	MOV A, B
LDA 16-bit	Loads the accumulator with the data from the memory location indicated by the 16-bit address	Memory direct	Three bytes	LDA 905FH
LHLD 16-bit	Loads the H and L registers directly from the two consecutive memory locations indicated by the 16-bit address	Memory direct	Three bytes	LHLD 900AH
STA 16-bit	Stores the contents of the accumulator in the memory location indicated by the 16-bit address	Memory direct	Three bytes	STA 9050H
SHLD 16-bit	Stores the contents of the H and L registers in two consecutive memory locations indicated by the 16-bit address	Memory direct	Three bytes	SHLD 809FH
PUSH Rp	Pushes the contents of the register pair onto a stack	Register direct	One byte	PUSH B
POP Rp	Pops the top two memory locations of the stack onto a register pair	Register direct	One byte	POP H
OUT 8-bit	Outputs the data in the accumulator to the port indicated by the 8-bit address	I/O	Two bytes	OUT 40H

(Contd)

Oxford University Press

© Oxford University Press. All rights reserved.

Table 1.8 Data transfer instructions *(Contd)*

Mnemonics	Tasks performed on execution	Addressing mode	Instruction length	Example
IN 8-bit	Inputs the data from the port indicated by the 8-bit address to the accumulator	I/O	Two bytes	IN 30H
MOV Rd, M	Copies the contents of the memory location pointed to by the HL register pair to the register	Indirect	One byte	MOV B, M
MOV M, Rs	Copies the contents of the register to the memory location pointed to by the HL register pair	Indirect	One byte	MOV M, C
LDAX Rp	Loads accumulator with the contents of the memory location pointed to by the register pair	Indirect	One byte	LDAX B
STAX Rp	Stores the contents of the accumulator in the memory location pointed to by the register pair	Indirect	One byte	STAX D
XCHG	Exchanges the contents of the HL register pair with that of the D and E register pair	Implicit	One byte	XCHG
SPHL	Copies the contents of the H and L registers to the stack pointer	Implicit	One byte	SPHL
XTHL	Exchanges the contents of the HL register pair with the top of stack	Implicit	One byte	XTHL

(iv) LDA and STA use memory direct addressing mode and a 16-bit memory address as operand.

(v) LDAX and STAX use indirect addressing mode for data transfer. The operand given in the instruction is one of the register pairs BC or DE. Register pair HL is not used with LDAX due to the availability of the alternative instruction MOV A, M.

(vi) LHLD and SHLD are the instructions used to transfer 16-bit data between the HL register pair and two consecutive memory locations. For example, executing SHLD 9000H instruction will store the contents of L register in 9000H and the contents of H register in 9001H.

(vii) PUSH and POP instructions are used for data transfer between a register

Oxford University Press

© Oxford University Press. All rights reserved.

pair and a stack. The stack is a set of memory locations configured as a last-in, first-out (LIFO) or first-in, last-out (FILO) array. The top of the stack locations is pointed to by a special register, the stack pointer, which is within the microprocessor. PUSH instruction will store the register pair given in the instruction to the top two memory locations of the stack. Similarly, POP instruction will copy the last two bytes stored in the stack to the register pair mentioned in the instruction. Care must be taken in using these instructions as the stack is configured as a LIFO array. Another instruction to store data in the stack is XTHL, which exchanges the top two memory locations of the stack with the contents of the HL register pair.

(viii) Stack pointer can be initialized using LXI or SPHL instructions. SPHL instruction will copy the contents of the HL register pair to the stack pointer.

(ix) IN and OUT instructions use 8-bit port addresses as operand. IN instruction is used to get data from the input port and the data obtained is stored in the accumulator. OUT instruction is used to issue data from the accumulator to an output port.

(x) XCHG instruction is used to exchange the contents of the HL and DE register pairs.

1.14.3 Arithmetic Instructions

The arithmetic instructions supported by the 8085 are addition, subtraction, and their variants. The arithmetic instructions are listed in Table 1.9.

Table 1.9 Arithmetic instructions

Mnemonics	Tasks performed on execution	Addressing mode	Instruction length	Example
ADI 8-bit	Adds the 8-bit data to the contents of the accumulator	Immediate	Two bytes	ADI 30H
ACI 8-bit	Adds the 8-bit data and the carry flag to the contents of the accumulator	Immediate	Two bytes	ACI 4FH
SUI 8-bit	Subtracts the 8-bit data from the contents of the accumulator	Immediate	Two bytes	SUI 2AH
SBI 8-bit	Subtracts the 8-bit data and the borrow from the contents of the accumulator	Immediate	Two bytes	SBI 5CH
ADD R	Adds the contents of the register to the contents of the accumulator	Register direct	One byte	ADD C

(Contd)

Oxford University Press

© Oxford University Press. All rights reserved.

Table 1.9 Arithmetic instructions *(Contd)*

Mnemonics	Tasks performed on execution	Addressing mode	Instruction length	Example
ADC R	Adds the contents of the register and the carry to the contents of the accumulator	Register direct	One byte	ADC E
SUB R	Subtracts the contents of the register from that of the accumulator	Register direct	One byte	SUB B
SBB R	Subtracts the contents of the register and the borrow from that of the accumulator	Register direct	One byte	SBB C
DAD Rp	Adds the contents of the register pair to that of the H and L registers	Register direct	One byte	DAD B
INR R	Increments the register by 1	Register direct	One byte	INR B
INX Rp	Increments the register pair by 1	Register direct	One byte	INX B
DCR R	Decrements the register by 1	Register direct	One byte	DCR E
DCX Rp	Decrements the register pair by 1	Register direct	One byte	DCX D
ADD M	Adds the contents of the memory location pointed to by the HL register pair to that of the accumulator	Indirect	One byte	ADD M
ADC M	Adds the contents of the memory location pointed to by the HL register pair and the carry to that of the accumulator	Indirect	One byte	ADC M
SUB M	Subtracts the contents of the memory location pointed to by the HL register pair from that of the accumulator	Indirect	One byte	SUB M
SBB M	Subtracts the borrow and the contents of the memory location pointed to by the HL pair from that of the accumulator	Indirect	One byte	SBB M

(Contd)

Oxford University Press

© Oxford University Press. All rights reserved.

Table 1.9 Arithmetic instructions *(Contd)*

Mnemonics	Tasks performed on execution	Addressing mode	Instruction length	Example
INR M	Increments the memory location pointed to by the HL register pair by 1	Indirect	One byte	INR M
DCR M	Decrements the memory location pointed to by the HL register pair by 1	Indirect	One byte	DCR M
DAA	Converts the contents of the accumulator from binary to BCD (Decimal-Adjust Accumulator)	Implicit	One byte	DAA

The following points list some key features of arithmetic operations:

(i) For arithmetic operations, one of the data must be stored in the accumulator and the other given or addressed in the instruction.
(ii) Add-with-carry instructions are used for multi-byte and higher-order byte addition.
(iii) Similarly, subtract-with-borrow instructions are used in multi-byte and higher-order byte subtraction.
(iv) Increment and decrement instructions can be operated not only on the accumulator, but also on other registers including memory locations.
(v) The contents of a register pair can be incremented or decremented using INX and DCX instructions.
(vi) DAA is the 8085 instruction that supports BCD addition. The addition of BCD data is done like binary addition, using the ADD instruction. DAA is used to convert the result of the binary addition of BCD numbers into a BCD number. This instruction cannot be used to directly convert binary numbers into BCD numbers.

Oxford University Press

1.14.4 Logical Instructions

The most important logical instructions supported by the 8085 are AND, OR, EXOR, and NOT. The complete list is given in Table 1.10.

Table 1.10 Logical instructions

Mnemonics	Tasks performed on execution	Addressing mode	Instruction length	Example
ANI 8-bit	The 8-bit data is logically ANDed with the contents of the accumulator.	Immediate	Two bytes	ANI 0FH
XRI 8-bit	The 8-bit data is logically EXORed with the contents of the accumulator.	Immediate	Two bytes	XRI 01H
ORI 8-bit	The 8-bit data is logically ORed with the contents of the accumulator.	Immediate	Two bytes	ORI 80H

(Contd)

© Oxford University Press. All rights reserved.

Table 1.10 Logical instructions *(Contd)*

Mnemonics	Tasks performed on execution	Addressing mode	Instruction length	Example
ANA R	The contents of the register are logically ANDed with the contents of the accumulator.	Register direct	One byte	ANA C
XRA R	The contents of the register are logically EXORed with the contents of the accumulator.	Register direct	One byte	XRA D
ORA R	The contents of the register are logically ORed with the contents of the accumulator.	Register direct	One byte	ORA E
ANA M	The contents of the memory location pointed to by the HL register pair is logically ANDed with the contents of the accumulator.	Indirect	One byte	ANA M
XRA M	The contents of the memory location pointed to by the HL register pair is logically EXORed with the contents of the accumulator.	Indirect	One byte	XRA M
ORA M	The contents of the memory location pointed to by the HL register pair is logically ORed with the contents of the accumulator.	Indirect	One byte	ORA M
RLC	Rotates the bits of the accumulator left by one position	Implicit	One byte	RLC
RRC	Rotates the bits of the accumulator right by one position	Implicit	One byte	RRC
RAL	Rotates the bits of the accumulator left by one position, through the carry	Implicit	One byte	RAL
RAR	Rotates the bits of the accumulator right by one position, through the carry	Implicit	One byte	RAR
CPI 8-bit	Compares the 8-bit data with the contents of the accumulator	Immediate	Two bytes	CPI FFH
CMP R	Compares the contents of the register with that of the accumulator	Register direct	One byte	CMP B
CMP M	Compares the contents of the memory location pointed to by the HL register pair with that of the accumulator	Indirect	One byte	CMP M
CMA	Complements the contents of the accumulator	Implicit	One byte	CMA
CMC	Complements the carry	Implicit	One byte	CMC
STC	Sets the carry	Implicit	One byte	STC

Oxford University Press

© Oxford University Press. All rights reserved.

For logical operations, one of the data must be stored in the accumulator and the other given or addressed in the instruction. Logical operations can be performed with immediate data, data stored in a register, or indirectly addressed memory location content.

Besides the instructions already mentioned, two types of rotate instructions are available in the 8085. One set—RLC and RRC—rotates the accumulator contents within itself. The RLC instruction shifts the accumulator content left by one bit. In the process, the most significant bit of the accumulator becomes the least significant bit. The RRC instruction shifts the accumulator content right by one bit.

The other set of rotate instructions—RAL and RAR—rotates the accumulator content along with the carry flag. The RAL instruction shifts the accumulator content left by one bit and in the process, the most significant bit will be shifted to the carry flag and the carry flag content will be shifted to the least significant bit of the accumulator.

The instruction set of the 8085 supports a compare instruction for comparing the magnitude of two binary numbers. The compare instructions are used to compare the accumulator content with the operand specified in the instruction. CPI instruction uses immediate addressing and CMP uses registers or indirectly addressed memory location for comparing with the accumulator. The result of the compare instruction is indicated in the flag register, as follows:

If [(A) − operand] = 0, i.e., (A) = operand, the zero flag is set.

If [(A) − operand] < 0, i.e., (A) < operand, the carry flag is set.

If [(A) − operand] > 0, i.e., (A) > operand, the zero and carry flags are reset.

1.14.5 Branching Instructions

Branching instructions are used to transfer the program execution to a different address. Branching instructions are of two types—jump instructions and subroutine instructions. The jump instructions merely transfer the execution from one location in the program to another, whereas the subroutine instructions in the main program transfer execution to a new location and also return to the main program. Return instructions are used for this purpose. The branching can take place unconditionally or conditionally, based on the flag conditions shown in Table 1.11. PCHL instruction is a special instruction used to branch to the address stored in the HL register pair.

RST *n* is the restart instruction supported by the 8085. Upon execution of the RST *n* instruction, the program execution will be transferred to the address given by $n \times 8$. For example, RST 4 instruction will transfer the execution to the address 0020H which is the hexadecimal equivalent of 32 (in decimal form).

In machine code or opcode, the16-bit or 4 hex digit addresses in the branching instructions are given such that the lower-order byte of the address follows the higher-order byte. For example, JMP 8030H is coded as C3 30 80. The opcode for JMP, C3, is stored first, followed by 30 and then by 80.

Oxford University Press

© Oxford University Press. All rights reserved.

Table 1.11 Branching instructions

Mnemonics	Tasks performed on execution	Instruction length	Example
JMP 16-bit	Jump unconditionally	Three bytes	JMP 9500
JC 16-bit	Jump if carry is set	Three bytes	JC 9500
JNC 16-bit	Jump on no carry	Three bytes	JNC 9500
JP 16-bit	Jump on positive	Three bytes	JP 9500
JM 16-bit	Jump on minus	Three bytes	JM 9500
JZ 16-bit	Jump on zero	Three bytes	JZ 9500
JNZ16-bit	Jump on no zero	Three bytes	JNZ 9500
JPE 16-bit	Jump on parity even	Three bytes	JPE 9500
JPO 16-bit	Jump on parity odd	Three bytes	JPO 9500
CALL 16-bit	Call unconditionally	Three bytes	CALL 9500
CC 16-bit	Call on carry	Three bytes	CC 9500
CNC 16-bit	Call on no carry	Three bytes	CNC 9500
CP 16-bit	Call on positive	Three bytes	CP 9500
CM 16-bit	Call on minus	Three bytes	CM 9500
CZ 16-bit	Call on zero	Three bytes	CZ 9500
CNZ 16-bit	Call on no zero	Three bytes	CNZ 9500
CPE 16-bit	Call on parity even	Three bytes	CPE 9500
CPO 16-bit	Call on parity odd	Three bytes	CPO 9500
RET	Return unconditionally	One byte	RET
RC	Return on carry	One byte	RC
RNC	Return on no carry	One byte	RNC
RP	Return on positive	One byte	RP
RM	Return on minus	One byte	RM
RZ	Return on zero	One byte	RZ
RNZ	Return on no zero	One byte	RNZ
RPE	Return on parity even	One byte	RPE
RPO	Return on parity odd	One byte	RPO
PCHL	Copy HL contents to the program counter	One byte	PCHL
RST 0/1/2/3/4/5/6/7	Restart	One byte	RST 5

1.14.6 Machine Control Instructions

Machine control instructions are used to control the microprocessor execution and functioning and are listed in Table 1.12. They are explained in detail in the following points:

(i) NOP means no operation. When this instruction is executed, nothing is done; no changes occur in the contents of the registers. The program counter alone is incremented to fetch and execute the next instruction.

(ii) HLT instruction is used to halt the execution of the program. The operation of the microprocessor is suspended when HLT instruction is executed. The only way to exit the halt state is to apply the hardware reset signal.

Oxford University Press

© Oxford University Press. All rights reserved.

(iii) Interrupts are disabled and enabled using DI and EI signals, respectively. Once the DI instruction has been executed, the processor ignores any interrupt request received. To enable interrupts again, the EI instruction has to be executed.

(iv) The SIM instruction is used to send serial data on the serial output data (SOD) line of the microprocessor and the RIM instruction is used to receive serial data on the serial input data (SID) line of the processor. The SIM and RIM instructions are also associated with the setting and reading of interrupt masks for RST hardware interrupts.

Table 1.12 Machine control instructions

Mnemonics	Tasks performed on execution	Addressing mode	Instruction length
NOP	No operation	Implicit	One byte
HLT	Halts the microprocessor execution	Implicit	One byte
DI	Disables interrupts	Implicit	One byte
EI	Enables interrupts	Implicit	One byte
RIM	Reads interrupt mask	Implicit	One byte
SIM	Sets interrupt mask	Implicit	One byte

1.15 SAMPLE PROGRAMS

1. Write an assembly language program to add two numbers.

The program given in Table 1.13 uses immediate addressing for the two data to be added. The data to be added are stored in memory locations 8001H and 8003H. The sum is stored in the memory location 8500H. This program assumes that no carry is generated from the addition.

Table 1.13 Program for adding two 8-bit numbers

Memory address	Machine code/ Opcode	Labels	Mnemonics with operands	Comments
8000	3E	START:	MVI A, 32H	; Load the first number in
8001	32			the accumulator.
8002	C6		ADI 64H	; Add the second number with
8003	64			the contents of the accumulator.
8004	32		STA 8500H	; Store the sum in the memory
8005	00			location 8500H.
8006	85			
8007	76		HLT	; Terminate program execution.

2. Write an assembly language program to add two numbers of 16 bits each.

This program also uses immediate addressing for loading the data in the processor registers. The sum is stored in the memory locations 8500H and 8501H, as shown in Table 1.14.

© Oxford University Press. All rights reserved.

Table 1.14 Program for adding two 16-bit numbers

Memory address	Machine code/ Opcode	Labels	Mnemonics with operands	Comments
8000	21	START:	LXI H, 805FH	; Load the first 16-bit number in
8001	5F			the HL register pair.
8002	80			
8003	01		LXI B, 123AH	; Load the next number in the BC
8004	3A			register pair.
8005	12			
8006	09		DAD B	; Add the two numbers using double addition instruction.
8007	22		SHLD 8500H	; Store the result in the HL pair in
8008	00			the memory locations 8500H and
8009	85			8501H.
800A	76		HLT	; Terminate program execution.

3. Write an assembly language program to add the two numbers stored in the memory locations 8500H and 8501H and store the result in 8502H.

This program uses indirect addressing instructions to load the numbers to be added in the processor registers. The carry, if generated, is ignored. The program is shown in Table 1.15.

Table 1.15 Program for adding two numbers from memory

Memory address	Machine code/ Opcode	Labels	Mnemonics with operands	Comments
8000	21	START:	LXI H, 8500H	; Initialize HL register pair to point
8001	00			to the memory location of the
8002	85			first number.
8003	7E		MOV A, M	; Load the first number in the accumulator.
8004	23		INX H	; Increment the HL pair to point to the memory location of the next number.
8005	86		ADD M	; Add the two numbers.
8006	23		INX H	; Increment the HL pair to point to the next memory location.
8007	77		MOV M, A	; Store the contents of the accumulator in the memory location pointed to by the HL register pair.
8008	76		HLT	; Terminate program execution.

Oxford University Press

© Oxford University Press. All rights reserved.

1.16 INSTRUCTION EXECUTION

The 8085 microprocessor is designed to fetch the instruction pointed to by the program counter, and then decode and execute the instruction within the processor. If necessary, further operand fetch takes place before completing the execution. Each instruction, as we have already seen, has two parts—operation code (known as opcode) and operand. The opcode is a command such as ADD and the operand is an object to be operated on, such as a byte or the contents of a register.

Instruction cycle is the time taken by the processor to complete the execution of an instruction. An instruction cycle consists of one to six machine cycles.

Machine cycle is the time required to complete one operation—accessing either the memory or an I/O device. A machine cycle consists of three to six T-states.

T-state is the time corresponding to one clock period. The T-state is the basic unit used to calculate the time taken for execution of instructions and programs in a processor.

To execute a program, the 8085 performs various operations such as opcode fetch, operand fetch, and memory read/write or I/O read/write. The microprocessor's external communication function can be divided into three categories:

(i) Memory read/write (ii) I/O read/write
(iii) Interrupt request acknowledge

POINTS TO REMEMBER

- The microprocessor is an electronic circuit that functions as the central processing unit (CPU) of a computer, providing computational control.
- The microprocessor is the controlling element in a computer system. The microprocessor performs data transfers, does simple arithmetic and logical operations, and makes simple decisions.
- The basic operation of the microprocessor is to fetch instructions stored in the memory and execute them one by one in sequence.
- Microprocessors are used in almost all advanced electronic systems.
- Microcontrollers are advanced forms of microprocessors, with memory and ports present within the chip.
- A microcomputer system is made by interfacing memory and I/O devices to a microprocessor.
- Microprocessor evolution is classified into five generations. The processors that are currently in use belong to the fifth generation.
- The microprocessor is a semiconductor device consisting of electronic logic circuits manufactured using either large-scale integration (LSI) or very large-scale integration (VLSI) technique. It works at a fixed clock frequency.
- A bus is a collection of wires connecting two or more chips.
- A typical microprocessor communicates with memory and other input/output devices using three buses—address bus, data bus, and control bus.
- Salient features of the 8085 microprocessor manufactured by Intel
  - It is an 8-bit microprocessor.
  - It has a 16-bit address bus (A0–A15) and hence, can address up to $2^{16} = 65,536$ bytes (64 KB).

Oxford University Press

© Oxford University Press. All rights reserved.

  - The 8085 has a multiplexed bus (AD0–AD7), which is used as the lower-order address bus and the data bus. It can be de-multiplexed using a latch and the ALE signal.
  - The data bus is a group of eight lines (D0–D7).
  - It supports external interrupt request.
  - It has a 16-bit program counter (PC) and a 16-bit stack pointer (SP).
  - It has six 8-bit general-purpose registers, which can be arranged in pairs as BC, DE, and HL.
  - It requires a +5 V power supply and operates at 3 MHz clock frequency.
  - It contains 40 pins and is available as a dual in-line package (DIP).
  - It has five flags—sign, zero, auxiliary carry, parity, and carry.
- The microprocessor operations related to data manipulation can be summarized in the following four functions:
  (i) Transferring data
  (ii) Performing arithmetic operations
  (iii) Performing logical operations
  (iv) Testing for a given condition and altering the program sequence
- The instructions are classified into three groups according to word size: one-, two-, and three-byte instructions.
- An instruction has two parts—opcode (operation to be performed) and operand (data to be operated on). The operand can be data (8-bit or 16-bit), addresses, registers, or implicit in the opcode. The method of specifying an operand (directly, indirectly, etc.) is called addressing mode.
- The instructions are executed in steps of machine cycles and each machine cycle requires many T-states.

KEY TERMS

Accumulator It is an 8-bit register; it is a part of the ALU and is the most important register. It is used to store 8-bit data and to perform arithmetic and logical operations. The output of an operation is also stored in the accumulator. The accumulator is identified as register A.

Address bus This bus carries the binary number (i.e., the address) used to access a memory location. Binary data can then be written into or read from the addressed memory location. The address bus consists of 16 wires and can, therefore, handle 16 bits.

Addressing mode It is the method of specifying the data to be operated on by the instruction.

Bus It is a group of conducting lines that carry data, address, and control signals

Clock speed This determines how many instructions per second the processor can execute. It is specified in megahertz (MHz).

Control bus This bus has various lines for coordinating and controlling microprocessor operations. For example, $\overline{RD}$ and $\overline{WR}$ lines.

Data bus This bus carries data in binary form between the microprocessor and external units such as memory. Typical size is eight or 16 bits.

DMA controller It is used to take control of the system bus by placing a high signal on the Hold pin.

Flag It is a flip-flop used to store information about the status of the processor and the status of the instruction executed most recently.

© Oxford University Press. All rights reserved.

Hold and HLDA These signals are used for direct memory access (DMA) type of data transfer. The Hold request makes the 8085 drive all its tri-stated pins to high impedance state. The HLDA signal goes high to acknowledge the receipt of the Hold signal.

Immediate addressing It transfers the operand given in the instruction—a byte or word—to the destination register or memory location.

Implied addressing In this addressing mode, the instruction itself specifies the data to be operated on.

IN This instruction is used to move data from an I/O port to the accumulator.

Indirect addressing It transfers a byte or word between a register and a memory location addressed by another register.

Instruction cycle It is the time required to execute an instruction.

IO/$\overline{\text{M}}$ signal This signal is used to differentiate memory access and I/O access. For input/output instructions it is high; for memory reference instructions it is low.

JMP and **CALL** JMP instruction permanently changes the program counter. CALL instruction leaves information on the stack so that the original program execution sequence can be resumed.

Machine cycle It is the time required to access the memory or input/output devices.

Memory direct addressing It moves a byte or word between a memory location and a register.

Opcode It is the part of the instruction that specifies the operation to be performed.

Operand It is the data on which the operation is performed.

OUT This instruction is used to move data from the accumulator to an I/O port.

Ready It is an input signal to the processor. It is used by the memory or I/O devices to get extra time for data transfer or to introduce wait states in the bus cycles.

Register direct addressing It transfers a copy of a byte or word from a source register to a destination register.

Timing diagram It is a graphical representation of the time taken by each instruction for execution. The execution time is represented in T-states.

Trap It is a non-maskable interrupt of the 8085 and is not disabled by processor reset or after reorganization of interrupt.

T-state It is the basic unit used to calculate the time taken for execution of instructions and programs in a processor. It is the time corresponding to one clock period.

Oxford University Press

REVIEW QUESTIONS

1. What is the main function of a computer?
2. Name any three input devices of a computer.
3. Name any two output devices of a computer.
4. Name any three storage devices of a computer.
5. Name any three places where computers can be used.
6. Draw a block diagram of a computer and label its components.
7. Who developed the world's first microprocessor?
8. What is the data bus width of the 8085 microprocessor?
9. When did Intel introduce the Pentium 4 microprocessor?
10. What is the amount of memory that the Pentium 4 processor can address?
11. What are the basic units of a microprocessor?
12. What is the function of microprocessor in a system?

© Oxford University Press. All rights reserved.

13. How many memory locations can be addressed by a microprocessor with 14 address lines?
14. Name any two types of memories that are used in a computer.
15. Define computer hardware.
16. Define computer software.
17. What is the role of CPU in a computer?
18. What are input and output devices?
19. Describe and draw the diagram of Von–Neumann model.
20. Define the following abbreviations: CPU, RAM, and ROM
21. Name any three features of the 8085.
22. What are the operations performed by the ALU of the 8085?
23. What are the various registers in the 8085?
24. What is a flag? List its types. What is the structure of the flag register? Explain each flag with an example.
25. List the 16-bit registers of the 8085 microprocessor.
26. What is a bus?
27. Why is the data bus bidirectional?
28. How are the signals of the 8085 classified?
29. How are clock signals generated in the 8085? What is the frequency of the internal clock?
30. How does the 8085 processor differentiate a memory access (read/write) signal from an I/O access (read/write) signal?
31. Why is crystal a preferred clock source?
32. Which interrupt has the highest priority in the 8085? What is the priority of the other interrupts?
33. When and where is the Ready signal used?
34. What are Hold and HLDA? How are they used?
35. Draw a general block diagram of a microprocessor-based system. Explain briefly the various blocks of the system. Give some examples of the types of devices used for each block.
36. What is a microprocessor? Sketch and explain the various pins of the 8085.
37. Explain the operation of these 8085 signals: Ready, S1 and S0, Hold and HLDA, and ALE.
38. Explain the architecture of the 8085 with the help of its internal block schematic diagram.
39. List the four categories of 8085 instructions that are used for data manipulation.
40. Define opcode and operand. Identify the opcode and the operand in the instruction MOV H, L.
41. Explain the instruction XCHG.
42. What is an instruction? List any four arithmetic instructions and their uses.
43. Define stack. Explain the instructions related to stack operations.
44. When is the instruction XRA A used?
45. How many operations are there in the instruction set of the 8085 microprocessor?
46. Explain with examples the different instruction formats, based on the length of the instructions.
47. List the four instructions which control the interrupt structure of the 8085 microprocessor.
48. What is the last instruction executed in a program? Why?

Oxford University Press

© Oxford University Press. All rights reserved.

49. What is the significance of XCHG and SPHL instructions?
50. Explain the operation carried out when the 8085 executes the instruction RST 0.
51. What is addressing? What are the various addressing modes available in the 8085?
52. Explain direct addressing with an example.
53. Explain implied addressing with an example.
54. What are the machine cycles in the 8085 microprocessor?

THINK AND ANSWER

1. Compare the instructions CALL and PUSH.
2. What is the difference between the shift and rotate instructions?
3. How many address lines are there in a 4096 × 8 EPROM chip?
4. Explain the difference between the instructions JMP and CALL.
5. If the instructions CALL and RET were not available in the 8085, would it still be possible to write subroutines? How would the subroutine be called? How would one return to the main program?

Oxford University Press

© Oxford University Press. All rights reserved.