DR.BABASAHEB AMBEDKAR
OPEN UNIVERSITY

BCA

BACHELOR OF COMPUTER APPLICATION

BCAR-303

System Programming and Introduction to Microprocessor

SYSTEM PROGRAMMING AND INTRODUCTION TO MICROPROCESSOR

DR. BABASAHEB AMBEDKAR OPEN UNIVERSITY

AHMEDABAD

Editorial Panel

Authors : Amit Suthar
Assistant Professor,
AMPICS| Department of Computer Science,
Ganpat University – Kherva.

Amit Bardhan
Assistant Professor,
Som–Lalit Education & Research Foundation,
Computer Science Department,
Opp. St. Xavier's College,
Navrangpura Ahmedabad.

Editor : Mr. Parimal Patel
I/C Director,
Khyati School of Computer Application,
Ahmedabad.

Language Editor : Dr. Vasant K. Joshi
Associate Professor,
G. B. Shah Commerce College,
Ahmedabad

ISBN 978-93-91071-08-0

Edition : 2021

Acknowledgment

Every attempt has been made to trace the copyright holders of material reproduced in this book. Should an infringement have occurred, we apologize for the same and will be pleased to make necessary correction/amendment in future edition of this book.

The content is developed by taking reference of online and print publications that are mentioned in Bibliography. The content developed represents the breadth of research excellence in this multidisciplinary academic field. Some of the information, illustrations and examples are taken "as is" and as available in the references mentioned in Bibliography for academic purpose and better understanding by learner.

ROLE OF SELF–INSTRUCTIONAL MATERIAL IN DISTANCE LEARNING

The need to plan effective instruction is imperative for a successful distance teaching repertoire. This is due to the fact that the instructional designer, the tutor, the author(s) and the student are often separated by distance and may never meet in person. This is an increasingly common scenario in distance education instruction. As much as possible, teaching by distance should stimulate the student's intellectual involvement and contain all the necessary learning instructional activities that are capable of guiding the student through the course objectives. Therefore, the course / self–instructional material are completely equipped with everything that the syllabus prescribes.

To ensure effective instruction, a number of instructional design ideas are used and these help students to acquire knowledge, intellectual skills, motor skills and necessary attitudinal changes. In this respect, students' assessment and course evaluation are incorporated in the text.

The nature of instructional activities used in distance education self–instructional materials depends on the domain of learning that they reinforce in the text, that is, the cognitive, psychomotor and affective. These are further interpreted in the acquisition of knowledge, intellectual skills and motor skills. Students may be encouraged to gain, apply and communicate (orally or in writing) the knowledge acquired. Intellectual–skills objectives may be met by designing instructions that make use of students' prior knowledge and experiences in the discourse as the foundation on which newly acquired knowledge is built.

The provision of exercises in the form of assignments, projects and tutorial feedback is necessary. Instructional activities that teach motor skills need to be graphically demonstrated and the correct practices provided during tutorials. Instructional activities for inculcating change in attitude and behavior should create interest and demonstrate need and benefits gained by adopting the required change. Information on the adoption and procedures for practice of new attitudes may then be introduced.

Teaching and learning at a distance eliminates interactive communication cues, such as pauses, intonation and gestures, associated with the face–to–face method of teaching. This is particularly so with the exclusive use of print media. Instructional activities built into the instructional repertoire provide this missing interaction between the student and the teacher. Therefore, the use of instructional activities to affect better distance teaching is not optional, but mandatory.

Our team of successful writers and authors has tried to reduce this.

Divide and to bring this Self Instructional Material as the best teaching and communication tool. Instructional activities are varied in order to assess the different facets of the domains of learning.

Distance education teaching repertoire involves extensive use of self–instructional materials, be they print or otherwise. These materials are designed to achieve certain pre–determined learning outcomes, namely goals and objectives that are contained in an instructional plan. Since the teaching process is affected over a distance, there is need to ensure that students actively participate in their learning by performing specific tasks that help them to understand the relevant concepts. Therefore, a set of exercises is built into the teaching repertoire in order to link what students and tutors do in the framework of the course outline. These could be in the form of students' assignments, a research project or a science practical exercise. Examples of instructional activities in distance education are too numerous to list. Instructional activities, when used in this context, help to motivate students, guide and measure students' performance (continuous assessment)

PREFACE

We have put in lots of hard work to make this book as user-friendly as possible, but we have not sacrificed quality. Experts were involved in preparing the materials. However, concepts are explained in easy language for you. We have included many tables and examples for easy understanding.

We sincerely hope this book will help you in every way you expect.

All the best for your studies from our team!

SYSTEM PROGRAMMING AND INTRODUCTION TO MICROPROCESSOR

Contents

Dr. Babasaheb Ambedkar Open University Ahmedabad **BCAR-303**

System Programming and Introduction to Microprocessor

BLOCK 1 : SYSTEM PROGRAMMING AND ASSEMBLY LANGUAGE

SYSTEM PROGRAMMING AND ASSEMBLY LANGUAGE

Block Introduction :

In earlier days of computing, on/off switches were used as primitive devices for communication. To transfer information, system programs were developed to make computers better suited to the needs of users. Some of the system programs are operating systems, compilers, interpreters, loaders, linkers, macro processors etc.

In this block students will be aware of about system programming and its techniques with certain use of Assembly language. The students will find the block effective as it explains about operating programs and how one should be benefitted.

The block will detailed about system program and use of assembly language. The block will stress on 8085 Assembly Language, Assemblers, Macros and Macro processor and Loaders. The students will be trained with practical examples and exercises that will help to learn and grab the subject easily.

Block Objectives :

After learning this block, you will be able to understand :

- Components of System Programming
- Evolution of Operating Systems
- Programming Languages
- Idea about 8085 Assembly Language
- Study of Assemblers
- Knowledge about Macros and Macro processor
- Concept about Loaders

Block Structure :

Unit 01

INTRODUCTION TO SYSTEM PROGRAMS AND SYSTEM PROGRAMMING

UNIT STRUCTURE

1.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The Concept of System programming
- Basic of Operating System as a main system program
- Various foundations of systems programming
- Role of assemblers in low level programming
- Idea about working of a loader
- Working of linker in a system.
- Detailed about compiler and interpreter
- Evolution of Operating Systems

1.1 Introduction :

During the early days of computing, the on/off switches were used as primitive devices for communication. However, for communicating complex information these traditional devices were not found useful. For this purpose, system programs were developed to make computers better suited to the needs of users. Examples of system programs are operating systems, compilers, interpreters, loaders, linkers, macro processors etc. Further system programs provided assistance to the user to prepare their programs. In this unit, you will study the components of system programming and evolution of operating system.

1.2 Components of System Programming :

Operating systems and file systems allow flexible storing and retrieval of information. Compilers are system programs that accept spoken languages, for example, English and translate into machine language. Loaders are system programs that prepare machine language programs for execution. Macro processors allow programmers to use abbreviations.

The productivity of computer system is heavily dependent on the effectiveness, efficiency and sophistication of a system program. Fig. 1.1 shows the foundation of system programming.

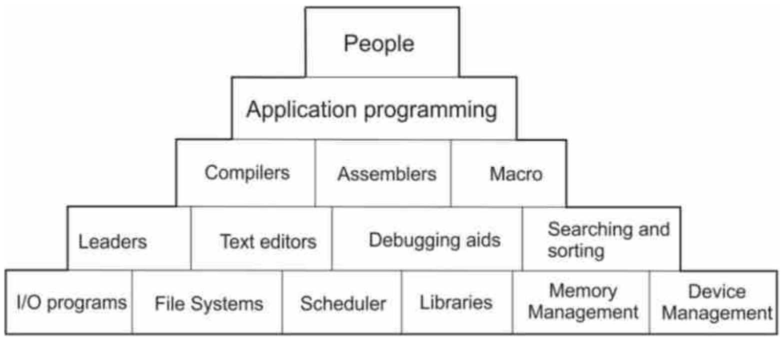


Fig. 1.1 Foundation of System Programming

As shown in Figure 1.1, the system programming is mainly developed for people who use computer systems. The application program packages are run with system program as a background. For example, an operating system would be required to execute an application program. The system program comprises of operating systems, compilers, interpreters, loaders, linkers, macro processors etc. Utility tools such as text editors, debuggers, search engines etc. are also based on system programs. The management of files, libraries, IO devices also depends largely on system programs.

The major components of a programming system are assembler, macro processor, loader, compilers and operating system :

1. **Assembler –** Assembler is the internal program that translates assembly language to machine language. Machine language is in terms of binary numbers that is 0 and the example of assembler is Microsoft Assembler [MASM]. The Input to an assembler is a type of an assembly language program, while output is an object program along with certain information which will allow the loader to arrange the object program for implementation.

2. **Macro Processor –** In macro processor, macro call is contraction for several codes. It is found that a macro definition is a series of code having particular name. So, a macro processor is a program to facilitate and specializes macro description for macro calls.

3. **Loader –** In micro programming, a loader is a routine so as to loads an object program and furthermore get ready for implementation. There are variety of loading schemes such as :

   - Absolute
   - Relocating
   - Direct linking

   Generally, a loader should load, relocate as well as link the object program.

4. **Compilers –** A compiler is a program whose purpose is to acknowledge a source program 'in high–level language' in addition to manufactures an equivalent object program. The programming language spoken in

simple English is identified as High Level Language. For example, BASIC is a high level language.

5. **Operating System** – An operating system is related with the allotment of resources as well as services like :
   - Memory
   - Processors
   - Devices
   - Information

An operating system is a significant constituent in a computing system. An operating system is related with distribution of resources as well as services as shown. It corresponds to the program which helps in managing resources like :

- traffic controller
- a scheduler
- memory management module
- I/O programs
- file system

❑ **Check Your Progress – 1 :**

1. Which is not an example of system program ?

   a. Operating systems c. Compilers

   b. MS Office d. Interpreters

2. ______ are system programs that prepare machine language programs for execution.

   a. loaders c. Compilers b. MS Office d. Interpreters

3. Which program will translates assembly language to machine language ?

   a. Assembler c. Compilers b. Loader d. Interpreters

1.3 Evolution of Operating Systems :

The evolution of operating system took place in early sixties. At those time, personal computers with 'on line' operations were not available. It was required to input programs/data through punch cards. The instruction/data would be created by punching holes in cards. Hence these cards are called punch cards. The set of such punched cards would constitute a program. This set would be fed to computer memory. This phenomenon is known as batch processing.

As discussed above, in early computers, there were no operating systems. But when the services of commercial computers started for execution of jobs the scheduling started on batch processing systems. Now, with the advancement of commercial computer services, a number of operating systems software has come into the picture.

Initially, starting with DOS, many operating systems software's were developed such as Unix, Oracle etc. depending on the requirement. The most commonly used operating system is Microsoft Windows. Most of the servers use FreeBSD, Linux and other Unix like systems a lot.

In simple batched system, the memory resource was allocated totally to a single program. Thus, if a program did not need the entire memory, a portion of that resource was wasted. Often in these partitioned memory systems, certain portion could not be used as it was too small to contain a program. The problem of 'holes' or unused portion of core is called fragmentation.

❖ **Operating System – User Viewpoint : Functions**

The purpose of an operating system (monitor) is to assist him in the mechanics of solving problems. Specifically, the following functions are performed by the system :

- Job sequencing, scheduling and traffic controller operation
- Input/output programming
- Protecting itself from the user
- Protecting the user from other users
- Secondary storage management
- Error handling

Secondary storage management is a task performed by an operating system in conjunction with the use of disks, tapes and other secondary storage for a user's programs and data.

An operating system must respond to errors. For example, if the programmer should overflow a register, it is not economical for the computer to simply stop and wait for an operator to intervene. Whenever an error occurs, the operating system must take appropriate action to resolve that.

❖ **Batch Control Language :**

A number of users view an operating system just through the batch system control cards by which they should preface their programs the monitor is a term which refers to the control programs of an operating system. Generally, in a batch system the jobs are stacked in a card reader and the monitor system processes each job in a sequential manner.

A job can consist of numerous separate programs to be executed sequentially, every individual program being called a job step. In a batch monitor system, the user communicates with the system through a control language. In a simple batch monitor system, there are two classes of control cards : execution cards and definition cards.

❖ **Operating system – user viewpoint : Facilities**

The function of an operating system is to provide all those facilities, which help to solve problems. This function is applicable for an oriented user the following facilities are conventionally provided by modern operating systems :

- Compilers, such as FORTRAN, COBOL and PL/ I
- Subroutine libraries, such as sine, cosine, square root
- Assemblers
- Linkage editors and program loaders, which bind subroutines together and prepare programs for execution
- Utility routines, such as SORT/MERGE and TAPE COPY
- Application packages, such as circuit analysis or simulation

- Management of system hardware
- Debugging facilities; such as program tracing and 'core dumps'
- Data management and file processing

❖ **Functions of Operating Systems :**

An Operating System contains situation of exceptional programs to facilitate computer system that will require working appropriately. It does various work like

- Identifying input from keyboard
- keeping track of files and directories on disk
- sending output to display screen
- controlling peripheral devices

OS is considered to provide the following functions :

- Control distribution of resources of computing system between various users as well as tasks.
- Provide interface among computer hardware as well as programmer which shortens in addition to makes possible for coding, creation, debugging of different application programs.

In addition to above, OS also serves following purposes :

- **Convenience** – It creates a computer suitable to exercise
- **Efficiency** – It increases the efficiency of computer system in an efficient manner.
- **Ability to evolve** – It is constructed in such a way that efficient development, testing as well as occurrence of new system functions are permitted.

❖ **System call :**

Modern processors provide processors that can be used as system call. System calls provide interface between a process and operating system. They generate interrupt signal that causes operating system to gain control of the processor.

❖ **Operating System Services :**

The operating system offers various services to programs and to the users of these programs. Some of the common tasks provided by the operating systems are as follows :

- **Error detection** – It is found that error occurs in CPU, I/O device or in memory hardware. It ought to obtain suitable action to make sure for correct as well as reliable computing.
- **File system manipulation** – In this, the program will either read or write a file. Here the O/S will allow the program to work on file.
- **I/O operation** – It is related to I/O file or device. The program here will run with the use of any I/O device. This I/O is made available by the operating system.
- **Program execution** – The program will get executed inside the memory where the O/S loads the data. Here the program will stop executing normally or unusually.

- **Communication –** Data transfer sandwiched between two processes is essential for some time. Together processes behave on single computer or on various computer however, it is associated all the way through computer network. Communication may be put into practice by two methods shared memory as well as message passing.

❑ **Check Your Progress – 2 :**

1. Set of such punched cards will constitute a :

   a. link c. software b. program d. all

2. Which is not part of Operating System Services ?

   a. Error Handling c. Program Executive

   b. File System Manipulation d. Program deletion

1.4 Let Us Sum Up :

The way we write the program to run a computer system is known as system programming. The system program is also called system software.

Examples of system programs are operating systems, compilers, interpreters, loaders, linkers, macro processors etc. Further, system programs provided assistance to the user to prepare their programs.

Operating systems and file systems allow flexible storing and retrieval of information. Compilers are system programs that accept spoken languages, for example, English and translate into machine language. Loaders can be referred to as system programs, which prepare machine language programs for execution. Macro processors allow programmers to use abbreviations.

Thus, the major components of a programming system are assembler, macro processor, loader, compiler and operating system. The evolution of operating system took place in early sixties. At that time, personal computer with 'on line' operations were not available. It was required to input programs/ data through punch cards. The instruction/data would be created by punching holes in cards. Hence, these cards are called punch cards. The set of such punched cards would constitute a program. This set would be fed to computer memory. This phenomenon is known as batch processing. In those days, a FORTRAN programmer would approach the computer with his source deck in his left hand and a green deck of cards that would be a FORTRAN Compiler in his right hand.

From the user's point of view, the purpose of an operating system (monitor) is to assist him in the mechanics of solving problems. Specifically, the following functions are performed by the system :

- Job sequencing, scheduling and traffic controller operation.
- Input/output programming
- Protecting itself from the user; protecting the user from other users
- Secondary storage management

Time–sharing is a method to allocate processor time. It is typically characterized by interactive processing and time–slicing of the CPU's time to allow quick response to each user.

Paging can be referred to as a method of memory allocation through which the program is sub–divided into equal portions or pages and core is sub–divided into equal portions or blocks. The pages are loaded into blocks.

Many users view an operating system only through the batch system control cards by which they must preface their programs.

The following facilities are typically provided by modern operating systems :

- Assemblers
- Compilers, are FORTRAN, COBOL and PL/ I
- Subroutine libraries, such as sine, cosine, square root
- Linkage editors and program loaders that bind subroutines together and prepare programs for execution.
- Utility routines, such as SORT/MERGE and TAPE COPY
- Application packages, such as circuit analysis or simulation.
- Debugging facilities; such as program tracing and 'core dumps'
- Data management and file processing
- Management of system

  hardware OS is designed to :
- Control allocation of resources of the computing system among various users and tasks
- Allocate a boundary among computer hardware and the programmer for use of coding, creation, debugging programs

1.5 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (b), 2. (a) 3. (a)

❑ **Check Your Progress 2 :**

1. (b), 2. (d)

1.6 Glossary :

1. **System software** – It is a system program.
2. **Loaders** – It is a system program which prepares machine language programs for execution.
3. **Paging** – A method of memory allocation by which a program is sub–divided into equal portions or pages.

1.7 Assignment :

Get information about various System Calls and explain function of any 10 system calls.

1.8 Activities :

What are different building blocks in foundation of System Programming ? Explain with a diagram.

1.9 Case Study :

Discuss with example how the evolution of Operating System took place.

1.10 Further Readings :

1. Systems Programming, John J. Donovan, Mc–Graw Hill Publishing
2. Principles of System Programming, R. M. Graham, Wiley Publishing
3. Microprocessor and Assembly Language Programming, D. A. Godse, A. P. Godse Technical Publications

Unit 02

ELEMENTS OF ASSEMBLY LANGUAGE PROGRAMMING

UNIT STRUCTURE

2.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The Concept of Assembly Language Programming
- Features of 8085 assembly language

2.1 Introduction :

The computer performs activities as per the instructions given by the user. The set of such instruction written for a particular task is known as computer program. Program is the instruction that tells the computer how to process the data into the form you want. After studying the evolution of operating system concepts, this unit will focus on concepts and features of assembly language programming.

2.2 Programming Languages :

The language in which computer program is written is known as programming language. The programming languages are classified as high level language and Low level language. See Fig. 2.1.

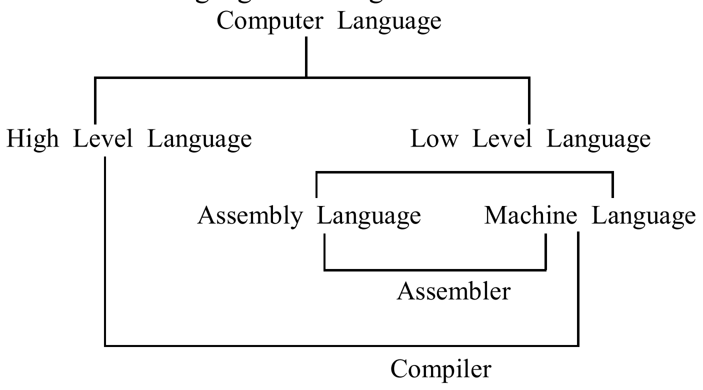


Fig. 2.1 Classification of Programming Languages

Low level language is further classified as assembly language and machine language. Each processor has a fixed set of instructions in form of binary pattern called machine language.

Number of bits that computer can recognise and process at a time is known as a word. The word ranges from 8 bit to 64 bits or even more. One byte comprises of 8 bits. Therefore, the word length ranges from 1 byte to 8 byte.

In machine language, the number of words is fixed for a given processor. For example, 8085 has a fixed word format. Therefore, the number of combinations are 2^8 i.e. 256. The manufacturer develops instructions from these words. One instruction may have one word or more than one word. The manufacturer selects combination of a bit pattern and gives a specific meaning to each combination by using electronic logic circuits. These combinations are called instructions. Set of such instructions is known as assembly language. It is not mandatory for a manufacturer to utilise all 256 word formats for creating instructions.

Machine language is expressed in terms of binary numbers i.e. 0 and 1 as the processor understands binary numbers only. However, for a human being, it is difficult to read and write the program in terms of 0s and 1s.

For example, in order to stop the processing of program execution, the following command is given by 8085 processor in machine language.

01110110

It is really difficult to remember and represent the above instruction expressed in machine language. Therefore, it is expressed in terms of the hexadecimal number as follows :

76 (H)

This code is called hex code or operation code or simply opcode. The opcode for halt i.e. 76 H which is abbreviated by the 8085 manufacturers as HLT. Now it is easy to remember the code HLT for stoppage of the program.

The machine language code is thus simplified by converting it to the code called op code. The op code depends upon type of processor. The program written in op code is known as assembly language code. During the run time, it is necessary to convert op code into machine language so that the processor will understand and process the code. The internal program that translates op code to machine code is known as assembler. The examples of assembler are Microsoft Assembler [MASM], Z–80, 8085, 8086 etc. The assembler for each processor is different.

In assembly language, the binary instructions of machine language are abbreviated. These abbreviated names are known as mnemonics. Thus, assembly language comprises of mnemonics.

Use of assembly language requires knowledge of assembly language and even computer hardware. For everyone it is much convenient to write a program in a High Level Language. The High Level Language comprises of instructions in simple English. Examples of High Level Language are BASIC, FORTRAN and COBOL etc. Compiler is the internal program that translates high level language to machine language. These high level languages are English–like languages such as BASIC, FORTRAN etc.

Each processor recognises different sets of instructions in machine language and therefore assembly language. For example, the machine language form 8085, 6800 and Z80 kits are different. Therefore, machine and assembly language is machine dependent. Using high level language, the instructions common to all processors can be written. Therefore, the high level language is machine independent.

A high-level language is a programming language that allows a programmer to develop programs that are not dependent on the type of computer being used. High-level languages are distinguished from machine-level languages by their resemblance to human languages.

When writing a program in a high-level language, the logic of the problem must be given complete attention.

Advantages of a high-level language

- Because it is written in English-like words, the high-level language is simple to read, write, and maintain.
- The purpose of high-level languages is to overcome the constraint of low-level languages, namely portability. The high-level language is portable, which means that it is machine-independent.

What is programming ?

"Simply put, programming is the act of giving a set of instructions to a computer to execute. If you've ever cooked from a recipe, imagine yourself as the computer and the recipe's author as the programmer. The recipe author gives you a set of instructions to read and then follow. The more complicated the instructions, the more complicated the outcome !"

And programming languages are the means by which we write instructions for computers to follow. Computers "think" in binary, which is made up of strings of 1s and 0s. Programming languages enable us to convert 1s and 0s into something humans can understand and write. A programming language is made up of a series of symbols that act as a bridge between humans and computers, allowing us to translate our thoughts into instructions that computers can understand.

Despite the fact that many languages share similarities, each has its own syntax. After learning the language's rules, syntax, and structure, a programmer writes the source code in a text editor or integrated development environment (IDE). The programmer will then frequently compile the code into machine language that the computer can understand.

Types of programming languages

Each of the programming languages mentioned in the following section can be classified as one or more of the following types (paradigms).

- High-level (most common) / low-level
- Declarative / imperative / procedural
- General-purpose / domain-specific
- Object-oriented / concurrent
- Command / Compiled / Script language

List of computer programming languages

- **Application and program development**

Application and program development is concerned with the development of programs that you use on a daily basis in your workplace. The Internet browser that you are currently using to view the web page, for example, is considered to be a program. If you are interested in creating a program, you might want to consider the following programming languages :

  - o C
  - o C#
  - o C++
  - o D
  - o Java
  - o Swift
  - o Tcl
  - o Visual Basic

- **Artificial intelligence development**

AI and related fields include the development of character interactions in computer games, portions of programs that make decisions, chatbots, and other applications of artificial intelligence. If you're interested in creating an artificial intelligence, you might want to consider the following programming languages :

  - o AIML
  - o C
  - o C#
  - o C++
  - o Prolog
  - o Python

- **Database development**

Database developers are responsible for the creation and maintenance of databases. When creating or maintaining a database, you should consider using one of the programming languages listed below :

  - o DBASE
  - o FoxPro
  - o MySQL
  - o SQL
  - o Visual FoxPro

- **Game development**

Game development is the process of designing and developing computer games and other forms of entertainment software. If you're thinking about creating a game, you might want to consider one of the following programming languages :

- o C
- o C#
- o C++
- o DarkBASIC
- o Java

- **Computer drivers or other hardware development**

Computer drivers and programming hardware interface support are required in order for hardware functionality to be achieved. If you're interested in creating drivers or software interfaces for hardware devices, you might want to consider learning one of the following programming languages :

- o Assembly
- o C

- **Internet and web page development**

The Internet and web page development are at the heart of the Internet's operation. The Internet would not exist if it were not for the efforts of developers. If you're interested in developing web pages, Internet applications, or other Internet-related tasks, you might want to consider learning one of the programming languages listed below :

- o HTML
- o Java
- o JavaScript
- o Perl
- o PHP
- o Python
- o XML

- **Script development**

It is unlikely that script creation and development will lead to a career, but knowing how to do so can increase productivity for you or your company, resulting in time savings in the thousands of hours. If you're interested in script development, you might want to think about learning one of the following languages :

- o AutoHotkey
- o awk
- o bash
- o Batch file
- o Perl
- o Python
- o Tcl

❑ Check Your Progress – 1 :

1. The language in which computer _______ is written is known as programming language.

   a. Program   b. Data   c. Information   d. All

2. In 8085, the possible word format is :

a. 256 b. 128 c. 512 d. none

2.3 8085 Assembly Language :

The assembly language of 8085 comprises 74 instructions having 246 bit pattern. In addition to the instruction set, the 8085 accepts 8 bit data from the input devices and sends it to output devices as per command. The programmer first writes the program in assembly language. He then manually translates the instructions to opcode using table. Then he enters the code the hex keypad of the microprocessor kit. Even though we can write a program using opcode, it is still difficult to understand. It is therefore a good programming practice to write a program using mnemonics. The set of 8085 mnemonics is called 8085 assembly language. In other devices such as Motorola 6800, Zilog Z80 as well as Intel 8085, the assembly language is different. Therefore, the assembly language program written for one microprocessor is not transferable to the other. The machine language and assembly language both are processor specific and non–transferable.

The instruction sets are instruction codes to perform some task. It is classified into five categories.

- Control Instructions (e.g. NOP, HLT, DI, etc.)
- Logical Instructions (e.g. CMP, CPI, ANA, etc.)
- Branching Instructions (e.g. JMP, CC, RET, etc.)
- Arithmetic Instructions (e.g. ADD, ADI, SUB, etc.)
- Data Transfer Instructions (e.g. MOV, MVI, LDA, etc.)

Addressing Mode in 8085

These are the instructions used to transfer the data from one register to another register, from the memory to the register, and from the register to the memory without any alteration in the content. Addressing modes in 8085 is classified into 5 groups

- **Register addressing mode :** In this mode, the data is copied from one register to another. For example : MOV K, B: means data in register B is copied to register K.
- **Direct addressing mode :** In this mode, the data is directly copied from the given address to the register. For example : LDB 5000K: means the data at address 5000K is copied to register B.
- **Indirect addressing mode :** In this mode, the data is transferred from one register to another by using the address pointed by the register. For example : MOV K, B: means data is transferred from the memory address pointed by the register to the register K.
- **Implied addressing mode :** This mode doesn't require any operand; the data is specified by the opcode itself. For example : CMP.

The procedure to write and execute 8085 assembly language using microprocessor kit is given below :

- Write a source code in mnemonics.
- Convert source code to hex code using table.

- Enter (type) the opcode in user memory in sequential order using the hex keypad.
- Execute program by pressing 'execute' key. The answer will be displayed on 7 segments LED.

Let us now understand how the microprocessor works. The monitor program of the microprocessor translates hex code to binary code and stores in R/w memory in a sequential manner. When the execute command is given, microprocessor fetches each instruction decodes it and executes it in a sequence till the end of program. Using ASCII keyboard, we can directly enter mnemonics instead of opcode.

Let us write a Machine language program to add two numbers as follows :

```
00111110    ;Copy value 2H in register A
00000010
00000110    ;Copy value 4H in register B
00000100
10000000    ;A = A + B
```

It uses English like words to convey the action / meaning called as MNEMONICS. For Example,

MOV	to indicate the data transfer
ADD	to add two values
SUB	to subtract two values

Now, let us write an assembly language program to add two numbers as follows :

```
MVI A, 2H   ;Copy value 2H in register A
MVI B, 4H   ;Copy value 4H in register B
ADD B       ;A = A + B
```

The above program has the immediate type of instruction called as MVI i.e. Move Immediate and register type of instruction called ADD which will accumulate the value of ACCUMULATOR (Register A) with register B and store the result in register A.

Assembly language is specific to a given processor. For example, assembly language of 8085 is different than that of Motorola 6800 microprocessor.

❑ **Check Your Progress – 2 :**

1. The assembly language of 8085 contains _______ instructions with ______ bit pattern.

   a. 74, 246 b. 246, 74 c. 74, 74 d. 246, 246

2. The 8085 microprocessor have _________ addressing mode to store the value inside register.

   a. Immediate b. Register c. Implicit d. None

3. Accumulator (Register A) is used to store the answer.

   a. True b. False

2.4 Let Us Sum Up :

The computer performs activity as per instruction given by the user. The set of such instruction written for a particular task is known as computer program. Program is the instruction that tell computer how to process the data into the form you want. Low level language is further classified as Assembly language and machine language. Each processor has a fixed set of instructions in form of binary pattern called machine language. Machine language is expressed in terms of binary numbers i.e. 0 and 1 as the processor understands binary numbers only. However, for a human being, it is difficult to read and write the program in terms of 0s and 1s.

The assembly language of 8085 comprises of 74 instructions having 246 bit pattern. In addition to the instruction set, the 8085 accepts 8 bit data from the input devices and sends it to output devices as per command. The programmer first writes the program in assembly language. He then manually translates the instructions to opcode using table. Then he enters the code the hex key pad of the microprocessor kit. Even though we can write a program using opcode, it is still difficult to understand. It is therefore a good programming practice to write a program using mnemonics. The set of 8085 mnemonics is called 8085 assembly language.

Let us now understand how the microprocessor works. The monitor program of the microprocessor translates hex code to binary code and stores in R/w memory in a sequential manner, When the execute command is given, microprocessor fetches each instruction decodes it and executes it in a sequence till the end of program. Using ASCII keyboard, we can directly enter mnemonics instead of opcode.

2.5 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (a), 2. (a)

❑ **Check Your Progress 2 :**

1. (a), 2. (a), 3. (a)

2.6 Glossary :

1. **Programming language –** It is a language in which computer program is written.
2. **Machine language –** It is a set of instructions in form of binary pattern.

2.7 Assignment :

Describe the various addressing modes available in 8085 microprocessor.

2.8 Activities :

Write an assembly language program to subtract two numbers.

2.9 Case Study :

Define the usage of microprocessor in various applications.

2.10 Further Readings :

1. Systems Programming, John J. Donovan, Mc–Graw Hill Publishing
2. Microprocessor and Assembly Language Programming, D. A. Godse, A. P. Godse Technical Publications
3. Principles of System Programming, R. M. Graham, Wiley Publishing

Unit

03

ASSEMBLERS

UNIT STRUCTURE

3.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The Concept of Assembler
- The basic functions of assembler
- The architecture of an assembler

3.1 Introduction :

Generally we focus on procedures for producing the machine language. However, the reader must keep in mind that the assembler must also produce other information for the loader to use. For example, externally defined symbols must be noted and passed on to the loader; the assembler does not know the address (value) of these symbols and it is up to the loader to find the programs containing them, load them into core, and place the values of these symbols in the calling program. In this unit we are primarily concerned with the production of machine language.

We have studied that assembler is the program that translates the mnemonics into its machine code. Usually the assembler is not provided on microprocessor kit (single board microcomputer). The program can be entered in mnemonics through ASCII keyboard (Using hex keyboard the program can be types directly in hex code or opcode). Examples of ASCII keyboard are TRS–80 by radio shack and Micro Division by Morrow Design. The assembler translates mnemonics into 8085 machine language code and allocates memory location to each machine code. This avoids manual assembly and errors associated with it. The code can be easily edited. The corresponding changes are made by the assembler in the memory locations and jump sequences.

In addition, the assembler calculates constant expressions and solves symbolic names for memory and other entities. The use of symbolic references is an important assembly feature, which saves tedious calculations and manual address updates after changes to the program. Most assembled macro devices also include textual replacement machines - e.g. to generate shorter common sequences of instructions as inline instead of so-called substitutions.

Additionally, some assemblers may be able to perform some simple types of instruction set-specific optimizations, depending on their design. x86 assemblers from various vendors, for example, are widely used and can be considered an example of this. Most of them are capable of performing jump-instruction replacements (long jumps replaced by short or relative jumps) in any number of passes upon request. This process is referred to as "jump-sizing." Another type of assembler is one that simply rearranges or inserts instructions, as is the case with some RISC architecture assemblers, which can assist in optimizing a sensible instruction scheduling in order to exploit the CPU pipeline as efficiently as possible.

Since the 1950s, assemblers have existed as a step above machine language and before high-level programming languages like Fortran, Algol, COBOL, and Lisp. Speedcode is perhaps the best-known example of a class of translators and semi-automatic code generators with properties similar to both assembly and high-level languages.

There may be multiple assemblers with varying syntaxes for a given CPU or instruction set architecture. For example, an instruction to add memory data to a register in an x86-family processor may be written as add eax,[ebx], in the original Intel syntax, but as addl (percent ebx), percent eax in the AT&T syntax used by the GNU Assembler. Despite their differences in appearance, various syntactic forms generate the same numeric machine code. Additionally, a single assembler may have multiple modes to accommodate variations in syntactic forms as well as their precise semantic interpretations (such as FASM-syntax, TASM-syntax, ideal mode, etc., in the special case of x86 assembly programming).

Basic Elements of Assembler

There is considerable variation in how assembler authors categorize statements and in the nomenclature they use. Specifically, some refer to any operation that is not a machine mnemonic or an extended mnemonic as a pseudo-operation (pseudo-op). It is composed of three distinct types of instruction statements used to define program operations:

- Opcode mnemonics
- Data definitions
- Assembly directives

❖ **Opcode mnemonics and extended mnemonics**

Instructions (statements) in assembly language, in contrast to high-level languages, are typically very short and straightforward. Mnemonics are symbolic names that are used to identify a single executable machine language instruction (an opcode). There is typically one opcode mnemonic for each machine language instruction that is defined in a computer program. Each instruction is typically composed of an operation code (opcode) and zero or more operands (also known as operands). In most cases, a single value or a pair of values is referred to in the instruction. Operands can be immediate (a value coded directly into the instruction itself), registers specified explicitly or implicitly in the instruction, or the addresses of data stored elsewhere in storage. This is determined by the underlying processor architecture; the assembler merely reflects the way in which this architecture operates and behaves. In many cases, extended mnemonics are used to specify a combination of an opcode and a specific operand. For example, the System/360 assemblers use B as an extended mnemonic for BC with a mask of 15 and NOP ("NO OPeration" - do nothing for one step) as an extended mnemonic for BC with a mask of 0.

Extended mnemonics are frequently used to support specialized uses of instructions, which are frequently for purposes that are not immediately apparent from the instruction name itself. For example, many CPUs do not have an explicit NOP instruction, but they do have instructions that can be used to accomplish the same result. Nop is implemented in 8086 CPUs using the instruction xchg ax, ax, with nop serving as a pseudo-opcode to encode the instruction xchg ax,ax. Some disassemblers are aware of this and will treat the xchg ax,ax instruction as a nop instruction. For BC and BCR with zero masks, the extended mnemonics NOP and NOPR are used by IBM assemblers for the System/360 and System/370 systems, respectively. These are referred to as synthetic instructions in the context of the SPARC architecture.

Additionally, some assemblers include built-in macros that generate two or more machine instructions. Using some Z80 assemblers, for example, the instruction ld hl,bc is recognised and translated into the values ld l,c followed by ld h,b. These are referred to as pseudo-opcodes in some circles.

Mnemonics are arbitrary symbols; in 1985, the IEEE published Standard 694, requiring all assemblers to use the same set of mnemonics. Since then, the standard has been withdrawn.

❖ **Data directives**

Data elements are used to hold data and variables, and instructions are used to define them. The types of data, the length, and the positioning of data are all defined by them. They can also specify whether or not the data is accessible to outside programs (programs that have been assembled separately) or whether it is only accessible by the program in which the data section is defined. Some assemblers group these under the category of pseudo-operations.

❖ **Assembly directives**

Assembly directives, also known as pseudo-opcodes, pseudo-operations, or pseudo-ops, are instructions given to an assembler that tell it to "do things other than assemble instructions." Directives affect the assembler's behavior and "may affect the object code, the symbol table, the listing file, and the values of internal assembler parameters," according to the manual. The term pseudo-opcode is sometimes used to refer to directives that generate object code, such as data generation.

To distinguish pseudo-ops from machine instructions, their names frequently begin with a dot. Pseudo-ops can make the program's assembly dependent on parameters entered by the programmer, allowing one program to be assembled in a variety of ways, possibly for different applications. A pseudo-op can also be used to change the way a program is presented to make it easier to read and maintain. Another common application of pseudo-ops is to set aside storage areas for run-time data and, if desired, to set their contents to known values.

Programmers can use symbolic assemblers to associate arbitrary names (labels or symbols) with memory locations and constants. Every constant and variable is usually given a name so that instructions can refer to them by name, resulting in self-documenting code. Each subroutine's name is associated with its entry point in executable code, so any calls to it can use that name. GOTO destinations are labelled inside subroutines. Local symbols, which are often lexically distinct from normal symbols, are supported by some assemblers (for example, the use of "10$" as a GOTO destination).

Some assemblers, such as NASM, offer flexible symbol management, allowing programmers to manage multiple namespaces, calculate offsets within data structures automatically, and assign labels to literal values or the results of simple assembler computations. Labels can also be used to create relocatable addresses for constants and variables.

Like most other computer languages, assembly languages allow comments to be added to program source code that will be ignored during assembly. Assembly language programs require careful commenting because the meaning and purpose of a sequence of binary machine instructions can be difficult to decipher. When changes must be made, the "raw" (uncommented) assembly language generated by compilers or disassemblers is difficult to read.

3.2 Functions of Assembler :

The fundamental functions of an assembler are to translate mnemonic operation codes to their machine language equivalents, and then it assigns machine addresses to symbolic labels by the programmer.

The basic assembler functions are as follows :

- Translating mnemonic language to its equivalent object code.
- Assigning machine addresses to symbolic labels.

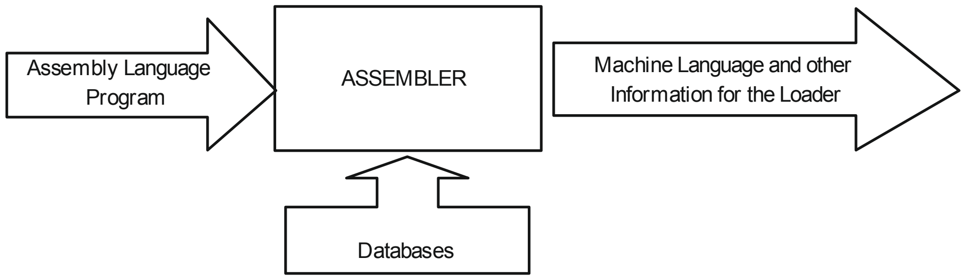

Assembly Language Program
ASSEMBLER
Machine Language and other Information for the Loader
Databases

Fig. 3.1 Function of an Assembler

Let us understand the working of assembler with some example.

The on/off switches are connected to the input port having address 00H and turn on devices are connected to the output port with address 01H. The suffix H indicates that the addresses are given in hexadecimal numbers.

The input has eight switches connected to the data bus through a tri–state buffer. Any one of the switches can be connected to 5 V (logic 1) or ground (logic and each switch controls the corresponding devices at output port. The microprocessor is required to read the bit pattern on the switches and send the same bit pattern to the output port in order to turn on the corresponding devices :

Instructions IN 00H

OUT 01H

HLT

When the microprocessor executes instruction H, it enables the tri–state buffer. The bit pattern formed by the switch position is placed on data bus and transferred to the accumulator. This is called reading of the input port.

When the microprocessor executes next instruction OUT 01H, it places the controls of the accumulator on the data bus and enables output port 01H. This is also called writing data to an output port. The output port latches the bit pattern and turns on/off the devices connected to the output port according to the bit pattern.

The drawback of the above program is the program reads switch positions once and stops. Therefore, if you want to turn on/off different devices, you have to reset the system and start it over again. This is impractical in real life situations. However, the unconditional jump condition (JNP), in place of HLT instruction will allow the microprocessor to monitor switch positions continuously.

The above program can be modified to read the switch positions continuously and turn on the appliances accordingly as follows :

START : IN 00H

OUT 01H

JMP START

The machine code and comments can be included in the above program as follows :

Memory Address	Machine Code	Label	Mnemonics	Comments
2000 2001	DB 00	START	IN 00H	Read input switches
2002 2003	D3 01		OUT 01H	Turn on the devices according to the switch positions
2004	C3		JMP START	Go back to beginning and read the switches again

The above program includes more than one column called label. Memory location of 2000H is defined with a label START. Therefore the operand of jump instruction is specified with label START. The program sets up an endless loop and the microprocessor monitors the input port continuously, the output will reflect any change in the switch position.

The above program is written considering manual assembly. That is, first we write mnemonics and then we translate corresponding hex code (machine code) as shown in the above example. This hexcode is entered through the hex keypad provided on microprocessor kit.

Now using ASCII keyboard, we can enter mnemonics directly using assembler. The assembler will convert mnemonics to machine code. The assembler program includes following assembler directives.

Label	Mnemonics	Comments
PORT0	EOQ 00H	Input Port Address
PORT1	OUT 01H	Output Port Address
	ORG 2000H	Start assembling the program from 2000H
START	IN PORT0	Read input switches
	OUT PORT1	Turn on devices
	JMP START	Go back and read switches
END		

The commands given through assembler are called assembler directives. The above program illustrates following assembler directives.

- **ORG** – The object code will be stored starting at location 2000 H.
- **EQU** – The directive EQU is the abbreviation of the word 'equate'. The program defines two equates : PORT0 and PORT1. In this program it would be easier to write port addresses directly with the instructions. The equates are essential in development projects in which hardware and software design are done concurrently. In such a situation, equates are convenient. Equates are also useful in long programs because it is easy to change or define port addresses by defining equates.
- **Label** – The program illustrates one label – START.
- **END** – End of assembly.

❑ **Check Your Progress – 1 :**

1. The system program that accepts assembly language program as input and produces its machine language equivalent know as ___________.

   a. Assembler b. Loader c. Compiler d. Interpreter

2. JMP mnemonic can be used for _______________.

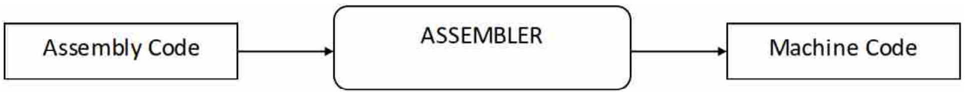
   a. Jump on specified label b. Describe the information

   c. Retrieve information d. None

3.3 Architecture of Assembler :

❖ **Single Pass Assembler :**

The assemblers that complete the whole assembly process within one pass, i.e. one reading of the program are called single pass assemblers. The assembler scans the source program instruction by instruction and translates them in machine codes. If there is instruction which is defined later in the program, but referenced earlier (i.e. forward reference problem) then the assembler leaves the address of that particular symbol or operand and process further. This forward reference symbol is entered in the symbol table and marked as undefined. Another list of instructions which use this symbol is also maintained. When the symbol definition is encountered, the assembler inserts the address in the symbol table. It also inserts the address to the instructions which use this symbol in the list of forward references. The object code in single pass assemblers can be produced in two ways. First, compile and loading scheme, in which the object code is directly loaded into the main memory for execution. So, no separate loaders are required. Secondly, the object program is stored secondary memory and loaded later in main memory for execution. Thus, separate loader is used.



It generates instructions by evaluating the mnemonics (symbols) in operation field and finds the value of symbol and literals to produce machine code. Now, if assembler do all this work in one scan then it is called single pass assembler, otherwise if it does in multiple scans then called multiple pass assembler.

❖ **Two Pass Assembler :**

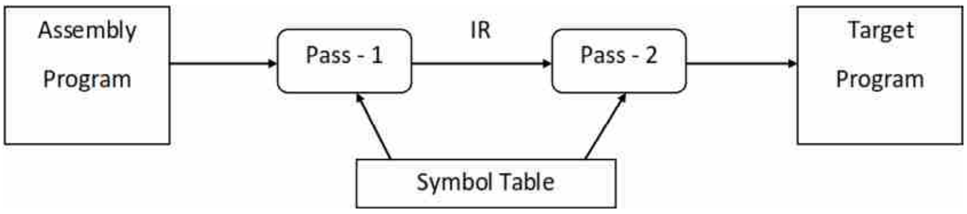
To arrange the program, the assembler examines throughout the program twice whose process is termed as two pass assembler.

- **Pass 1 :**

1. Define symbols and literals and remember them in symbol table and literal table respectively.
2. Keep track of location counter.
3. Process pseudo–operations.

- **Pass 2 :**

1. Generate object code by converting symbolic op–code into respective numeric op–code.
2. Generate data for literals and look for values of symbols.



During initial period, the first memory place is calculated from ORG statement furthermore makes the location counter to initialize. Further, the assembler will go through every instructions as well as records in address column of initial byte of instruction. The vision of location counter is to look all the way to every byte in the program. The assembler in addition, will produce a symbol table for the initial period. When it comes across a label, it records a label and its location.

During the next stage, every instruction is observe furthermore the mnemonics as well as labels are changed by machine hex notation which can be visualized as :

❖ **Pass 1 :**

Address Hex	**Machine Code Hex**	**Label Opcode**	**Operand**	**Symbol Table**
2000		START : IN	PORT0	PORT0 00H
2002		OUT	PORT1	PORT1 01H
2004		JUMP	START	START 2000H

❖ **Pass 2 :**

2000	DB00
2002	D301
2004	C30020

❖ **Assembled Print File :**

The assembled print file lists the memory address of the first byte of each instruction.

This program monitors the switch positions of the input port and turn on/off devices connected to the output port.

```
0000 = PORT0 EQU 00H; Input port address
0001 = PORT1 EQU 01H; Output port address
2000 ORG 2000H; Start assembling program from location
; 2000H
2000 DB00 IN PORT0; Read input switches
2002 D301 OUT PORT1; Turn on devices
2004 C30020 JMP START; Go back and read switches again
2007 END
```

The example of assembled print file is shown above. The comments are started with; mark. The equate relationship is indicated by = mark.

❑ **Check Your Progress – 2 :**

1. In the program

```
START : IN 00H
OUT 01H
JMP START
```

IN 00H denotes

a. Read input switches b. Read output switches

c. Turn on the devices d. Turn off the devices

2. All the machine opcode is stored inside ___________

a. MOT Table b. Base Table c. POT Table d. LC

3. The forward reference is a main problem in One Pass Assembler.

a. True b False

3.4 Let Us Sum Up :

Assembler is the program that translates the mnemonics into its machine code. Usually the assembler is not provided on microprocessor kit (single board microcomputer). The program can be entered in mnemonics through ASCII keyboard (Using hex keyboard the program can be types directly in hex code or opcode). Examples of ASCII keyboard are TRS–80 by radio shack and Micro Division by Morrow Design. The assembler translates mnemonics into 8085 Machine language code and allocates memory location to each machine code. This avoids manual assembly and errors associated with it. The code can be easily edited. The corresponding changes are made by the assembler in the memory locations and jump sequences. In order to assemble the program, the assembler scans through program twice, this is known as a two pass assembler.

3.5 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1. (a), 2. (a)

- **Check Your Progress 2 :**

  1. (a), 2. (a) 3. (a)

3.6 Glossary :

1. **Assembler –** It is system program that translate mnemonic operation codes to their machine language equivalents.

2. **Single Pass Assembler –** Forward reference is the problem in this type of assembler

3. **Two Pass Assembler –** Pass 1 define the symbols (assign addresses) and Pass 2 assemble instructions and generate object program.

3.7 Assignment :

Describe the one pass assembler with its limitation in your words.

3.8 Activities :

Discuss how the two pass assembler is used in 8085 microprocessor.

3.9 Case Study :

Elaborate the Machine – Dependent Assembler features and Machine – Independent Assembler features.

3.10 Further Reading :

1. Systems Programming, John J. Donovan, Mc–Graw Hill Publishing
2. Microprocessor and Assembly Language Programming, D. A. Godse, A. P. Godse Technical Publications
3. Principles of System Programming, R. M. Graham, Wiley Publishing

Unit 04

THE MACRO PROCESSOR AND LOADERS

UNIT STRUCTURE

4.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The Macro Instructions and concepts of Macro Processor
- Role of loaders
- Working of compile and go loaders
- Functions of general loader scheme
- Application of absolute loaders
- Concept of Direct linking loader

4.1 Introduction :

A macro processor is a program that copies a stream of text from one place to another, making a systematic set of replacements as it does so. Macro processors are often embedded in other programs, such as assemblers and compilers. Sometimes they are standalone programs that can be used to process any kind of text.

Macro processors have been used for language expansion (defining new language constructs that can be expressed in terms of existing language components), for systematic text replacements that require decision making, and for text reformatting

4.2 Macro Instructions :

A Macro instruction is the notational convenience for the programmer. For every occurrence of macro the whole macro body or macro block of statements gets expanded in the main source code. Thus Macro instructions make writing code more convenient.

A macro instruction is a group of programming instructions that have been compressed into a simpler form and appear as a single instruction. When used, a macro expands from its compressed form into its actual instruction details. Both the name of the macro definition and other variable parameter attributes are included within the macro statement.

Macro instructions were first used in the assembler language rather than a higher–level programming language. The way a macro expands to a set of instructions depends on the macro definition, which converts the macro into its detailed instructional form.

Macros save developers much time and effort, especially when dealing with a certain sequence of commands that is repeated more than once within the program body. Macros also save space and spare the programmer time spent on a long code block that may pertain to performing a single function.

The concept of macros is used within some precompilers, while higher–level languages focus on simplifying program and function writing, which makes the macro instruction a common element among most high–level programming languages. Macro instructions are generated together with the rest of the program by the assembler.

For example, before calling a subroutine, the contents of all registers may need to be stored. This routine work can be done using a macro. This requires a sequence of instructions. We can define and use a macro, SAVEREGS, to represent this sequence of instructions.

4.3 Macro Processor :

In order to relieve a programmer from the need to repeat identical parts of his program, operating systems provide macro processing facilities, which allow programmers to define abbreviations for a part of his program and to use these abbreviations in his program so as to avoid repeated use of identical code. The macro processor treats abbreviation as a macro definition. The macro processor substitutes definitions for all occurrences of abbreviation in the program such abbreviations are called macro calls.

In addition to this task, the macro processors are also used as general text handlers and for specializing operating systems to individual computer installations. To specialise an operating system, series of macro calls are written. These are processed by the macro processor by substituting appropriate definitions, thereby producing all the programs for an operating system. As we have seen above, macro is the simplest form of abbreviation for a sequence of operations. Consider the following program.

```
A 1, DATA "Add contents of DATA to register 1"
A 2, DATA "Add contents of DATA to register 2"
A 3, DATA "Add contents of DATA to register 3"
.
.
.
A 1, DATA "Add contents of DATA to register 1"
A 2, DATA "Add contents of DATA to register 2"
A 3, DATA "Add contents of DATA to register 3"
DATA DC, F5
```

In the above program, the following sequence occurs twice :

A 1, DATA

A 2, DATA

A 3, DATA

Using macro allows us to assign name to a sequence and to use that name in it's a macro facility permits us to assign name to this sequence and to use this name in its place. We can invent a macro language that allows us to specify the above as macro definition and allows us to refer the definition later. The macro processor effectively constitutes a separate language processor with its own language.

The name is attached to a sequence by means of macro instruction definition. This definition is formed in the following manner :

MACRO	===> Start of definition
INCR	===> Macro Name
A 1, DATA A 2, DATA A 3, DATA	$\Big\}$ Sequence of instructions to be abbreviated.
MEND	===> End of definition

The MACRO pseudo–op is the first line of definition and identifies the following line as a micro instruction name. Following the name line is a sequence of instructions being abbreviated the instructions comprising the macro instruction. This definition is terminated by a line with MEND (macro end) pseudo op.

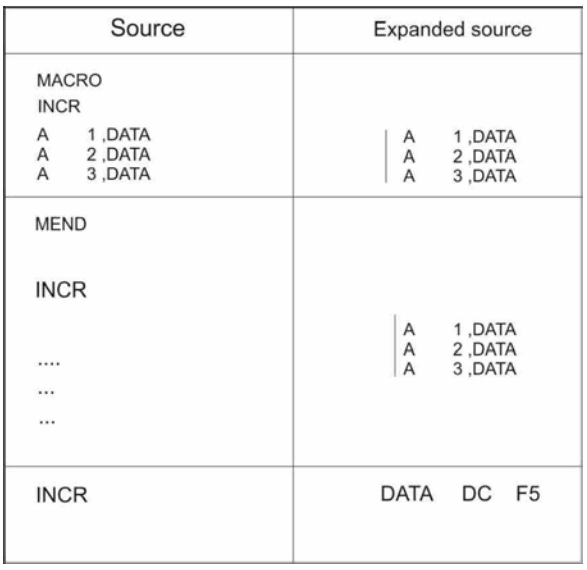
Once the macro has been defined, the use of macro name as an operation mnemonic in an assembly system is equivalent to the use of corresponding instruction sequence. The above example is re–written as follows, assigning the name INCR to the repeated sequence.

In this case the macro processor replaces the macro call with the lines :

A 1, DATA

A 2, DATA

A 3, DATA



Source	Expanded source
MACRO INCR A 1 ,DATA A 2 ,DATA A 3 ,DATA	A 1 ,DATA A 2 ,DATA A 3 ,DATA
MEND INCR	A 1 ,DATA A 2 ,DATA A 3 ,DATA
INCR	DATA DC F5

The process of replacement is called expanding the macro. The macro definition does not appear in the expanded source code. This definition is saved by the macro processor. As the macro name occurs in the source program it has to be expanded and the process is called a macro call.

❑ **Check Your Progress – 1 :**

1. If the instructions are repeating in the program, then ________is used to lower the length of such program.

   a. procedure   b. subroutine

   c. macro   d. none of the above

2. In programming, the macro present inside the macro is known as_______.

   a. macro in macro   b. nested macro

   c. macro inside macro   d. none of the above

3. Which of the following language provide built in facilities for writing macros

   a. C   b. C++   c. Ada   d. All of the above

4.4 Loaders :

We have studied that the user programs which normally gets converted to object programs (machine language) by assemblers, compilers or interpreters. Loader is a program, which accepts object programs, prepares them for execution by the processor and initiates the execution. This is shown in the general loading scheme (Fig. 4.1).

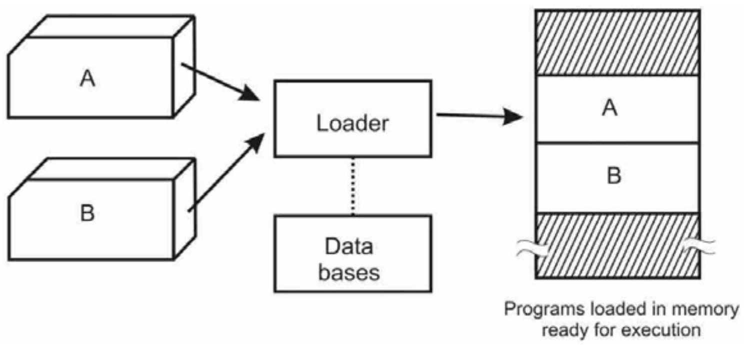


Fig. 4.1 General Loading Scheme

Basically, the loader performs following four functions :

- Allocate space in memory for the program (allocation).
- Resolve symbolic references between object desks (linking).
- Adjust all address dependent locations such as address constants to respond to the allocated space (relocation).
- Physically place the machine instructions and data into the memory (loading).

Loader Schemes : The loader schemes provide various ways of loading.

The main loader schemes are discussed below :

1. Compile and Go Loader :

In this scheme, the assembler runs in this part of memory. The assembled machine instructions and data are placed directly in the assigned memory locations. When the assembly gets completed, the assembler causes a transfer to staring instruction of the program. This is the simple solution involving no extra procedures. Such a loading scheme is commonly called "compile–and–go" or "assemble–and–go". See the Fig. 4.2.

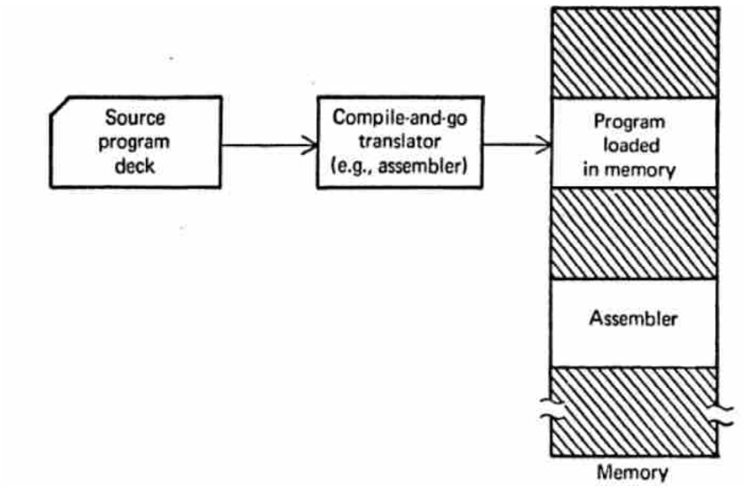


Fig. 4.2 "Compile and go" loader scheme

The assembler just places code into core and the loader comprises one instruction which transfers to the starting instruction of the newly assembled program. Apart from this, there are certain disadvantages of using this method. First, portion of the memory is wasted because the core occupied by the assembler is unavailable to the object program. Second, it is required to retranslate (assemble) the user's program every time during its run. Third, it is very difficult to handle multiple segments, especially if the source programs are in different languages.

2. General Loader Scheme :

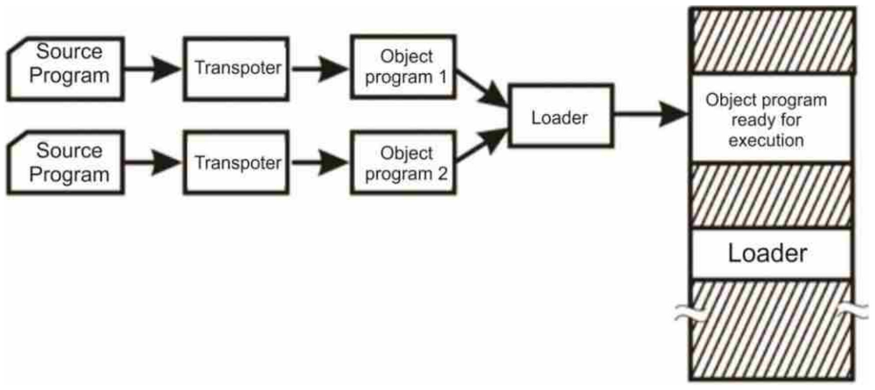


Fig. 4.3 General Loader Scheme

The outputting of instructions and data as they are assembled circumvents the problem of wasting core for the assembler. That type of an output could be saved and loaded whenever the code has to be executed. The assembled programs could be loaded into the same area in core that the assembler occupied (since the translation will have been completed). This type of output form, which may be on cards containing a coded form of instructions, is called an object deck.

The use of object deck as intermediate data to avoid one disadvantage of preceding 'compile–and–go' scheme requires addition of new program to the system, a loader.

The function of the loader is to allow the machine instructions, data and useful information in shape of object layout and keeps the instructions and data in the centre of an executable structure. It is found that more memory can be stored in the loader as it is smaller as compared to assembler.

So the assemblers and compilers being the program converter that will form an equivalent object program and with the help of linkage conventions will write subroutines in many different languages for further action by the loader in machine language.

❖ **Absolute Loader :**

Such loader is simple and carries general model of loader structure. Here the assembler will generate an output same as complier and go loader generate. The machine language translation of source program will be shown by the assembler.

❖ **Disadvantages :**

- In this, programmer needs to show the location address to load the program.
- The multiple subroutine will makes them hard to reallocate further.
- Programmer has to remember the address of each subroutine and use that absolute address explicitly in other subroutines to perform subroutine linkage.

The figure 4.4 indicates the operation of absolute assembler and the absolute loader. The programmer should be careful not to assign 2 subroutines to the same or overlapping locations.

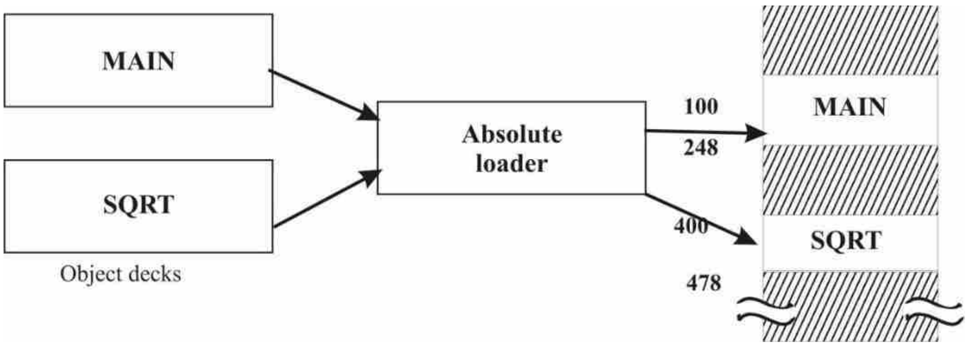


Fig. 4.4 Absolute Assembler and Loader

The four loader functions are accomplished as follows in the absolute loading scheme :

- Allocation – by programmer
- Linking – by programmer
- Relocation – by assembler
- Loading – by loader

❖ **Direct linking loaders :**

It is a normal type of relocatable loader which uses famous loading schemes. It makes the programmer to work with multiple procedure and data segments.

This type linking loader will feasibly allow programmer to work with in referencing data or instructions thereby allowing flexible intersegment referencing and accessing ability with standalone program execution.

The information provided to the loader by the assembler is mentioned below :

- The length of the segment.
- A list of all symbols in the segment that may be referenced by the other segments and their relative location within the segment.
- List of the symbols not defined in the segment but referenced in the segment.
- Information regarding the location of the constants in the segment and a description how to revise their values. The machine code translation of a source program and the relative addresses assigned.

❑ **Check Your Progress – 2 :**

1. A loader is

   a. program which keeps programs in memory and helps them for execution.

   b. program which translates assembly language into machine language.

   c. program which acknowledges program written in high level language and gives object program.

   d. program which work out source program in machine language.

2. Load address for the first word of the program is called

   a. Linker address origin b. Load address origin

   c. Phase library d. Absolute library

4.5 Let Us Sum Up :

A macro processor enables you to define and to use macros in your assembly programs. When you define a macro, you provide text (usually assembly code) that you want to associate with a macro name. Then, when you want to include the macro text in your assembly program, you provide the name of the macro. The assembler replaces the macro name with the text specified in the macro definition.

Salient features of Macro Processor :

- Macro represents a group of commonly used statements in the source programming language.
- Macro Processor replaces each macro instruction with the corresponding group of source language statements. This is known as the expansion of macros.
- Using Macro instructions programmer can leave the mechanical details to be handled by the macro processor.
- Macro Processor designs are not directly related to the computer architecture on which it runs.
- Macro Processor involves definition, invocation, and expansion.

Loader is a program which accepts object programs, prepares them for execution by the processor and initiates the execution. Basically, the loader performs following four functions :

- To arrange place in memory for program execution
- To solve symbolic references among object linking
- To alter related locations address of dependent locations
- To put the machine instructions and data in memory

The main loader schemes are :

- Compile and go loader
- General Loader Scheme
- Absolute Loader
- Direct Linking Loader

4.6 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (c), 2. (b), 3. (d)

❑ **Check Your Progress 2 :**

1. (a), 2. (b)

4.7 Glossary :

1. **Macro :** A macro allows you to define instruction sequences that are used repetitively throughout your program.

2. **Loader :** It is a program that intakes object programs for execution by processor.

4.8 Assignment :

Obtain information about various loader schemes and prepare a report on the same in your own words.

4.9 Activities :

Describe the Macro in 'C' Programming Language.

4.10 Case Study :

Create a report on functionalities of macro in various programming languages.

4.11 Further Readings :

1. Systems Programming, John J. Donovan, Mc–Graw Hill Publishing
2. Microprocessor and Assembly Language Programming, D. A. Godse, A. P. Godse Technical Publications
3. Principles of System Programming, R. M. Graham, Wiley Publishing

BLOCK SUMMARY :

The block will give detail knowledge and understanding about system programming and its techniques with certain use of Assembly language. The students will be trained with extra examples and deep knowledge about the subject. The block focus more on system program and usage of assembly language with more stress on 8085 Assembly Language, Assemblers, Macros and Macro processor and Loaders. The students will give extra practical examples and exercises that will help to learn and grab the subject easily.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Explain Loaders ?
2. What is OpCode ?
3. What are the differences between Compiler and Interpreter ?
4. Explain the difference between low – level V/s high – level languages.
5. Write the procedure to write and execute 8085 assembly language.

❖ **Long Questions :**

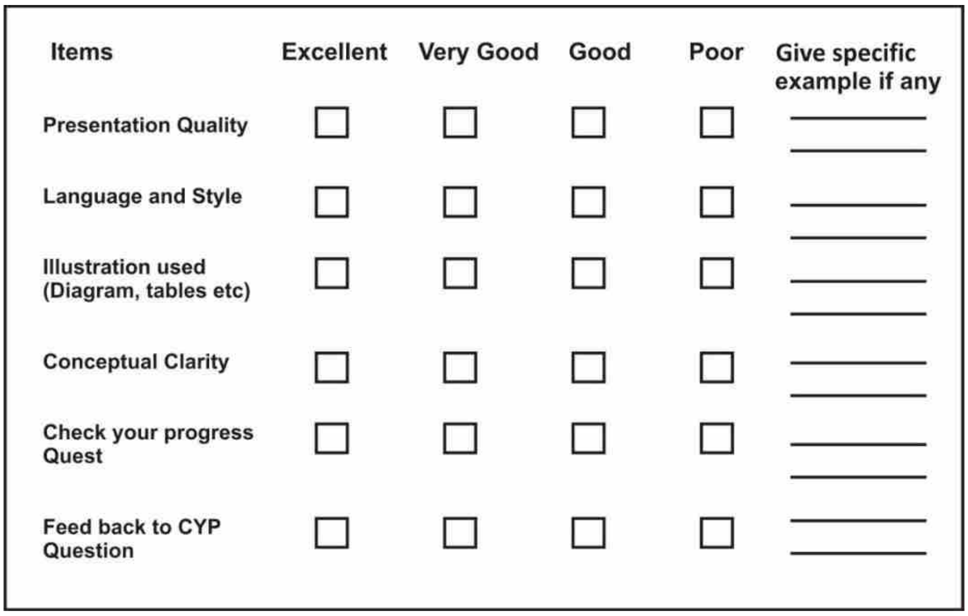
1. Discuss the working of macro processors.
2. Discuss the various functions of Operating Systems.
3. Write any five mnemonic codes of assembly language along with their description

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	1	2	3	4
No. of Hrs.				

2. Please give your reactions to the following items based on your reading of the block :



Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	________
Language and Style	☐	☐	☐	☐	________
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	________
Conceptual Clarity	☐	☐	☐	☐	________
Check your progress Quest	☐	☐	☐	☐	________
Feed back to CYP Question	☐	☐	☐	☐	________

3. Any other Comments

....

....

....

....

....

....

....

....

Dr. Babasaheb Ambedkar Open University Ahmedabad

BCAR-303

System Programming and Introduction to Microprocessor

BLOCK 2 : SYSTEM PROGRAMMING AND ASSEMBLY LANGUAGE

INTRODUCTION TO MICROPROCESSOR

Block Introduction :

The instructions or commands are entered through input devices such as keyboard or switch. The set of such instructions is known as program. The set of programs is called software. The input given to the computer is processed by the Central Processing Unit (CPU). The CPU sends output to the output devices such as Monitor, Liquid Crystal Display (LCD) and Light Emitting Diode (LED).

In this block students will be aware with the basic idea about microprocessor and its origination. The students will find the block effective as it explains about what a processor and how it is used in today's computer.

The block will detailed about microprocessor and its history with knowledge about various types of microprocessors. The block will stress on 8085 microprocessor and its related circuit arrangement with pins and sockets. The block will guide about aspects of microprocessor. The students will be trained with practical examples and exercises that will help to learn and grab the subject easily.

Block Objectives :

After learning this block, you will be able to understand :

- Concept of Microprocessor
- Understanding the Internal Architecture and Bus System of microprocessor
- Idea about history and overview of Microprocessors
- Detail about Intel 8085 Processing Architecture
- Knowledge about SSI, MSI, VLSI
- Basic of Microprocessors and Microcomputers
- Examine about 8085 Microprocessor
- Different Registers in 8085

Block Structure :

Unit 05

INTRODUCTION TO MICROPROCESSORS

UNIT STRUCTURE

5.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The basic principle of working of microprocessors.
- The difference between microcomputer and microprocessors.
- The internal architecture of microprocessor.
- Bus system architecture.
- The purpose of address signals.
- How the transmission of data signals takes place.
- The role of control and timing signals.
- How the execution of instructions takes place ?

5.1 Introduction :

The digital computer is based on the model 'input–process–output'. The instructions or commands are entered through input devices such as keyboard or switch. The set of such instructions is known as program. The set of programs is called software. The input given to the computer is processed by the Central Processing Unit (CPU). The CPU sends output to the output devices such as Monitor, Liquid Crystal Display (LCD) and Light Emitting Diode (LED). The timing of operations is controlled by the Control unit. The hardware constitutes physical components of computer. This unit will deal with the discussion on microprocessor.

5.2 Microprocessors :

In simple words, computer is defined as an electronic device that accepts data from Input device processes it and stores in a disk and displays it on output device such as monitor and on printer. The basic block diagram of a computer is shown in Fig. 5.1. For all types of computers such as personal computer, laptop, palm top etc., the fundamental principle of working is the same.

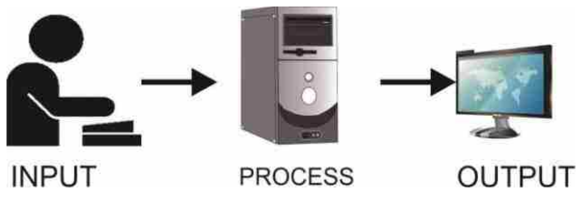


Fig. 5.1 Block Diagram of Computer

As shown in Fig 5.1, there are four main building blocks of a computer organisation – input, processor, output and memory. The data is inputted through input device. Keyboard, disks, mouse are the input devices. The input devices translate data and programs that the human being can understand into the form that the computer can process.

The CPU recognises and processes binary numbers. The binary numbers are constituted from binary digits called bits. The bits are expressed as 0 and 1. The bits 0 and 1 are generated from low and high electrical voltage signals respectively. The bit is stored in digital electronic circuit called register. One bit data requires one register for storage. Therefore 8 bit data requires 8 bit register for storage. Ten rows of 8 bit register will constitute 2^{10} i.e. 1024 bytes or 1 kilobyte memory.

CPU is the main component of Computer or Microprocessor system. It comprises of Arithmetic Logic Unit (ALU), registers, decoders, encoders, counters etc. It receives instructions from keyboard and performs a specified task. It communicates with I/O devices to accept or send the data.

The microprocessor is capable of performing computing functions and making decisions to change a sequence of programming execution. The main parts of microprocessors are Arithmetic and Logic Units (ALU), registers and control units. The ALU comprises of arithmetic circuits such as half and full adder, subtractor etc. These circuits perform arithmetic operations such as addition, subtraction, complementing etc. The ALU also have logic circuits such as AND, NOT, NAND, XOR gates which perform logical operations. The data can be temporarily stored in the circuit called register. The control unit comprises of control and timing signals.

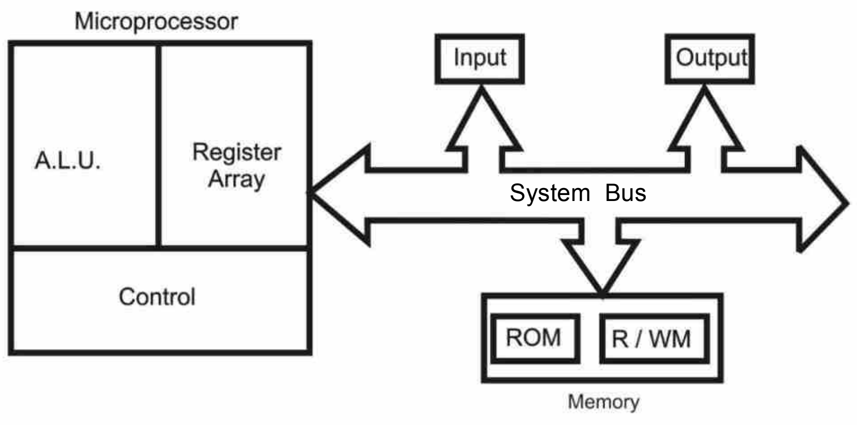


Fig. 5.2 Block Diagram of Microprocessor System

Input : The input section of microprocessor mainly comprises of keyboard, teletypewriter and analog to digital converters. There are two types of keyboards – Hex Keyboard and ASCII Keyboard. As name indicates, hex keyboard comprises of 16 numbers of hexadecimal keys 0 to 9 and A to F. Additionally, it has function keys such as execute, store etc. ASCII keyboard is similar to English typewriter keyboard.

Output : This section of the microprocessor is connected with peripherals such as Light Emitting Diodes (LED), Liquid Crystal Display (LCD), Printers and Cathode Ray Tube (CRT). The 8085 microprocessor is connected to 7 segment LED display.

Memory : The memory section of the microprocessor basically has two types of memories – Read Only Memory (ROM) and Read/Write Memory

(R/WM). The ROM comprises of stored programs which we cannot change. Example of ROM is a monitor program. Monitor program reads instructions received from input, compares with standard set and provides equivalent binary code to the processor for execution.

On other hand, R/WM is a user memory. User can save his programs and data in R/WM. The contents of R/WM can be edited.

System bus : It is the group of wires carrying bits. It comprises three types of buses – address bus, data bus and control bus. The monitor program monitors the keys of hex keyboard and stores instruction and data for further processing.

❖ **Principle of Working of the Microprocessor :**

The instructions received from keyboard are stored sequentially in the R/W memory. The microprocessor fetches the first instruction, decodes and executes it. The sequence of fetch, decode and execute continues till the microprocessor comes across an instruction which instructs to stop. During the entire process, microprocessor uses system bus to fetch the binary instruction and data from memory. It stores data in general purpose registers temporarily. It performs computing in ALU. Finally, it sends out results in binary using system bus to 7 segments LED.

❑ **Check Your Progress – 1 :**

1. Microprocessor is also known as :

   a. Memory   b. CPU   c. Bits   d. None of these

2. Which is not a part of microprocessor ?

   a. Arithmetic and Logic Units (ALU)

   b. Registers

   c. Control units

   d. Memory

3. ALU contains :

   a. NOT   b. NAND   c. XOR   d. all

5.3 Internal Architecture :

The internal working of the microprocessor system takes place as follows :

- The 8 bit data is stored in the register.

- The Arithmetic–Logic operation is performed as per command.
- Sequencing of execution of instructions is done.
- The data is stored temporarily in R/W memory during execution.

❑ **Check Your Progress – 2 :**

1. In microprocessor, 8 bit data is stored in ________.

   a. ALU b. register c. memory d. None of these

5.4 Bus System Architecture :

We have studied that the system bus carries various types of signals. The 8085 bus architecture comprises three types of buses – address, data and control bus.

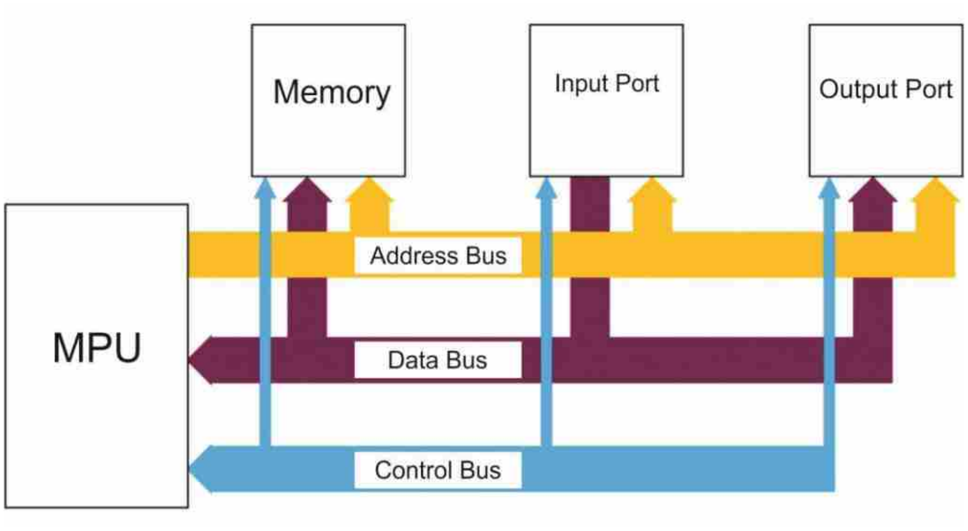


Fig. 5.3 Bus System of 8085

Address Bus : The location of memory and peripherals are identified by placing a 16–bit address on address bus. The address bus is a 16–bit unidirectional bus, therefore number of memory locations will be 2^{16} i.e. 65536 or 64 KB. The 16 bit address signals are denoted by A0, A1, A2… A15.

Data Bus : The data bus is 8 bit bidirectional bus. These eight bits are denoted as D0, D1, D2…. D7. Unlike address bus, data bus is bidirectional.

The 8 bit data has 2^8 i.e. 256 combinations. Therefore, the data ranges from 00 to FF H. The microprocessor is specified in terms of data bus signals. For example, 8085 is specified as an 8–bit microprocessor based on 8 bit data bus. On similar lines, the microprocessors Intel 8086, Motorola 68000, Zilog Z8000 are specified as a 16 bit processor.

Control Bus : The control bus comprises of individual lines that carry pulses generated by the microprocessor during its operation. For example, while reading I/O device, the microprocessor generates a signal I/OR (Input / Output Read). The overall working of these buses takes place sequentially as follows. Let us read data from a memory :

- The microprocessor places 16 bit address on address bus.
- The microprocessor then generates a signal MEMR to enable memory chip for reading. In case of a low active signal, it will generate a signal $\overline{\text{MEMR}}$

The 8 bit data is loaded from the specified location of memory chip on data bus and sent to microprocessor for further processing.

❑ **Check Your Progress – 3 :**

1. The address bus is ______ unidirectional bus

   a. 16–bit b. 8–bit c. 32–bit d. 64–bit

2. Which is not a function of Microprocessor ?

   a. Communicates with all peripherals with system bus.

   b. Stores binary instructions and data.

   c. Performs the computing tasks specified in a program.

5.5 Let Us Sum Up :

The digital computer is based on the model 'input–process–output'. The instructions or commands are entered through input devices such as keyboard or switch. The set of such instructions is known as program. The set of programs is called software. The input given to the computer is processed by the Central Processing Unit (CPU). The CPU sends output to the output devices such as Monitor, Liquid Crystal Display (LCD) and Light Emitting Diode (LED). The timing of operations is controlled by the control unit. The hardware constitutes physical components of computer.

The CPU recognizes and processes binary numbers. The binary numbers are constituted from binary digits called bits. The bits are expressed as 0 and 1. The bits 0 and 1 are generated from low and high electrical voltage signals respectively. The bit is stored in digital electronic circuit called register. One bit data requires one register for storage. Therefore 8 bit data requires 8 bit register for storage. Ten rows of 8 bit register will constitute 2^{10} i.e. 1024 bytes or 1 kilobyte memory,

CPU is the main component of computer or microprocessor system. It comprises of Arithmetic Logic Unit (ALU), registers, decoders, encoders, counters etc. It receives instructions from keyboard and performs a specified task. It communicates with IO devices to accept or send the data.

The microprocessor is capable of performing computing functions and making decisions to change a sequence of programming execution. The main parts of microprocessors are Arithmetic and Logic Units (ALU), registers and control units. The ALU comprises of arithmetic circuits such as half and full adder, subtractor etc.

The internal working of the microprocessor system takes place as follows :

- The 8 bit data is stored in the register.
- The Arithmetic–Logic operation is performed as per command.
- Sequencing of execution of instructions is done.
- The data is stored temporarily in R/W memory during execution.

Address Bus : The location of memory and peripherals are identified by placing a 16–bit address on address bus. The address bus is a 16–bit unidirectional bus, Therefore number of memory locations will be 2^{16} i.e. 65536 or 64 KB. The 16 bit address signals are denoted by A0, A1, A2… A15.

Data Bus : The data bus is 8 bit bidirectional bus. These eight bits are denoted as D0, D1, D2…. D7. Unlike address bus, data bus is bidirectional.

The 8 bit data has 2^8 i.e. 256 combinations. Therefore, the data ranges from 00 to FF H. The microprocessor is specified in terms of data bus signals. For example, 8085 is specified as an 8–bit microprocessor based on 8 bit data bus. On similar lines, the microprocessors Intel 8086, Motorola 68000, Zilog Z8000 are specified as a 16 bit processor.

Control Bus : The control bus comprises of individual lines that carry pulses generated by the microprocessor during its operation. For example, while reading I/O device, the microprocessor generates a signal I/OR (Input / Output Read). The overall working of these buses takes place sequentially as follows.

5.6 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (b), 2. (d), 3. (d)

❑ **Check Your Progress 2 :**

1. (b)

❑ **Check Your Progress 3 :**

1. (a), 2. (d)

5.7 Glossary :

1. **Programs** – These are set of instructions.
2. **System bus** – It is the group of wires carrying bits.

5.8 Assignment :

Explain with block diagram the working of microprocessor systems.

5.9 Activities :

Gather more information about working of microprocessors and explain the same along with proper diagrams.

5.10 Case Study :

Find out information about the bus system of 8085 and using a proper diagram explains the same.

5.11 Further Readings :

1. Systems Programming, John J. Donovan, Mc–Graw Hill Publishing.
2. Microprocessor and Assembly Language Programming, D. A. Godse, A. P. Godse Technical Publications.
3. Principles of System Programming, R. M. Graham, Wiley Publishing.

Unit 06

HISTORY AND OVERVIEW OF MICROPROCESSORS

UNIT STRUCTURE

6.0 Learning Objectives :

After learning this unit, you will be able to understand :

- 8085 Processing Architecture.
- Various Scale level Integrations.
- Medium Scale level Integration.
- The history of Microprocessors and Minicomputers.
- The general overview of Processors.

6.1 Introduction :

The microprocessor system comprises of devices as a unit that communicate with peripherals, that provide timing signals, direct data flow and perform computing tasks as specified by the instructions in the memory. This unit will explain about the necessary lines for address bus, data bus and control signals and would require power supply and crystal. You will also discuss about the small, medium and very large scale integrations.

6.2 Intel 8085 Processing Architecture :

The Intel 8085 Microprocessor was considered to be the first microprocessor released by Intel in late 1970s. It was the successor to Intel's 8080 processing unit whose features were increased compatibility and much simpler voltage requirement. This processor was mostly used as a microcontroller for many hardware applications.

- **Memory :** Program, data and stack memories occupy the same memory space. The total addressable memory size is 64 KB.

- **Program memory :** The program can be located anywhere in memory. Jump, branch and call instructions use 16–bit addresses, i.e. they can be used to jump/branch anywhere within 64 KB. All jump/branch instructions use absolute addressing.
- **Data memory :** The data can be placed anywhere as the 8085 processor always uses 16–bit addresses.
- **Stack memory :** It is limited only by the size of memory. Stack grows downward.

First 64 bytes in a zero memory page should be reserved for vectors used by RST instructions.

❖ **Interrupts :**

There are 5 interrupts in 8085 as :

INTR – It is a maskable 8080A compatible interrupt. In this, when the interrupt occurs, the processor carry out one of the instruction from the bus :

- 8 RST instructions (RST0 – RST7)
- CALL instruction (3 byte instruction)

RST5.5 – This maskable interrupt will allow processor to keep the data of PC register in stack with 2Ch address.

RST6.5 – This maskable interrupt make the processor to keep data of PC register in that stack with branches to 34h address.

RST7.5 – This maskable interrupt directs the processor to keep the data of PC register in stack and in branches to 3Ch address.

Trap is a non–maskable interrupt. When this interrupt is received the processor saves the contents of the PC register into stack and branches to 24h (hexadecimal) address.

I/O ports – There are 256 Input as well as Output ports **Registers**

Accumulator or register is 8–bit that does arithmetic, logic, I/O and load/ store operations.

Flag is also an 8–bit register that carries 5 1–bit flags as :

- **Sign :** It will set when MSB of the result is set.
- **Zero :** It will set when result is 0.
- **Auxiliary carry :** It will set when a carry out from bit 3 to bit 4 exists.
- **Parity :** It will set when parity is even.
- **Carry :** It will set when there was carry at addition or borrow at subtraction.

❖ **General Registers :**

- In this, an 8–bit B as well as and 8–bit C registers are applied to make 16–bit BC register pair. In pair, the C register has low order byte, where few instructions are used by BC register as data pointer.
- Further, 8–bit D as well as 8–bit E registers combines to form 16–bit DE register. In pair, such E bit register has low order byte. Many instructions having DE register is used as data pointer.

- 8–bit H and 8–bit L registers can be used as one 16–bit HL register pair. When used as a pair the L register contains low–order byte. HL register usually contains a data pointer used to reference memory addresses.

Stack pointer is a 16 bit register. This register is always incremented /decremented by 2.

Program counter is a 16–bit register.

❖ **Applications :**

The 8085 microprocessor actually saw very little use as a central processing unit. Rather, it's most prevalent implementation was that of a microcontroller. The Intel 8085 microcontroller was most notably used on instrumentation by NASA during physics missions until the early 2000s.

❖ **Instructions and Registers :**

The operational instructions of the Intel 8085 were the same as its predecessor, the 8080, except for two new instructions. The 8085 had seven processing registers. These registers were eight bits wide and had the capability of performing 16–bit operations if they were paired together with other registers. This process worked by placing the operating instruction in the first 8–bit register and additional instruction or execution data in the subsequent register space.

❑ **Check Your Progress – 1 :**

1. Intel 8085 Microprocessor was released by Intel in.

   a. 1971 b. 1969 c. 1968 d. 1965

2. 8085 microprocessorhas_____interrupts.

   a. 4 b. 5 c. 7 d. 9

6.3 SSI, MSI, VLSI :

In 1960s, computers were accessible and affordable to large establishments such as defense sectors, universities, government agencies etc. Due to advances in semiconductor technology, computers became more efficient and cheaper.

During the early days of electronic devices and circuits, the semiconductor devices such as transistors and diodes were invented after valve tubes. The Printed Circuit Boards (PCBs) were used on which these components were mounted. During late fifties, the integrated circuits comprising of number of semiconductor components were developed. In early sixties, the logic gates came in existence.

The integration of logic gates on single chip was designed. This was known as Small Scale Integration (SSI). During the later period the era of Medium Scale Integration (MSI) came into existence. The MSI comprised of more than hundred components mounted on a single chip. This integration was known as Large Scale Integration (LSI). The LSI technology was advanced to Very Large Scale Integration (VLSI). In VLSI, more than ten thousand components can be embedded in a single integrated circuit. Microprocessor is the semiconductor device consisting of digital electronic logic circuits manufactured from LSI or VLSI.

❑ **Check Your Progress – 2 :**

1. The integration of logic gates on single chip was known as :

   a. Small Scale Integration (SSI)

   b. Medium Scale Integration (MSI)

   c. Medium Scale Integration (MSI)

   d. all

6.4 History of Microprocessors and Microcomputers :

Robert Noyce, the founder of Intel in 1969 invented innovative function for silicon technology named as microcomputer. Apart from this, Hoff thought it would be better to use MOS LSI technology to manufacture computer. Owing to increasingly growing compactness of large scale integrated (LSI) tracks, a computer on a chip was predictable.

Intel's first microcomputer comes into view in November 1971. They break new ground in integrated electronics. Intel transports two dissimilar microcomputers five months apart.

Intel commence as memory Chip Company in 1969. Each one of custom chip had a different function such as :

• Keyboard
• Printer
• Display
• serial arithmetic
• control

With a team of 2 designers, Intel solves problems with lesser chip designs. During 1970, Intel implement 4–b computer on 3 LSI chips which was kept in 16–pin packages. Lowering of data word to 4–b will compromise for 1–b serial calculator chips and 16–b computers. The extent of 4–b word size allows CPU chip size which Intel used as 16–pin package.

The multiplex logic will amend chip of ROM/RAM memory chips so as to allow built–in address registers.

❖ **Features :**

• 256 x 8 Read Only Memory (2kb ROM) with :
• 4–b I/O port
• 80 x 4 Random Access Memory (320 b RAM) with :
• 4–b output port
• 4–b CPU chip with :
• 16x 4–b index registers
• 45 1 and 2 byte instructions
• 4–level Subroutine Address Stack
• 12–b Program Counter (4 k addresses)
• Brief of Microcomputers above A. ROM Chip (4001)

A. **ROM Chip (4001) :** Earlier calculators make use of special chips that are used in keyboard, display and printer. MCS–4 will handle control logic operations where program are kept inside the ROM. CPU's 12–b program will counter addresses till 16 ROM chips. Such work uses single ROM chip while calculator used four such chips.

The ROM chip contains :

- multiple address register
- output data register
- multiplexers
- control and timing logic

B. **RAM Chip (4002) :** Since the calculator requires 16–digit decimal floating–point numbers, the RAM was amended with 20–digit word (80 b) as :

- 16 digits for the fraction
- 2 digits for the fraction
- 2 digits for signs and control
- 20 digits x 4 b/digit

As RAM chip handles four 80–b numbers with the chip having output port. The built–in refresh counter was applied to handle data integration. At time of instruction fetch cycles, the refreshment took place when data inside RAM could not be able to access.

C. **Input / Output Ports :** The ROM and RAM chips carries 16 bits which was integrated with 4–b ports that utilises direct connection for I/O devices to save chip count and making current power/clock pins in use. For an output, a program choose RAM/ROM chip which further was forwarded to 4–b of Accumulator data to choose output port. For desk calculation, the display, keyboard and printer were connected with such ports.

D. **Microprocessor – CPU Chip (4004) :** As seen, in calculator, every single key operation will make thousands of processor instructions that were done from ROM. As seen in a subroutine, if a 10–byte loop exists for serial addition, it uses 80 µs/digit. Over here, a subroutine is added and a CPU index register will address every 16 digits which are kept in RAM. The main difference in comparison with many computers results in MCS–4's programs and data. Earlier computers will be able to run from RAM memory. In a minicomputer, a subroutine will instruct execution of PDP–8 and HP 2114 as a program's having a return address.

E. **Distributed Logic Architecture :** The architecture carries a time division multiplexing of 4–b bus, where CPU's address stack and on–chip DRAM memories are shown with MCS–4 architecture where a distributed decoding of instructions takes place. ROM/RAM chips inspected bus structure that was locally decoded with port instructions send by ROM.

F. **MCS–4 Applications :** It seems that, a minimum system configuration carries two chips : CPU and ROM. It is found that a calculator will have 4 ROM's and RAM chip having five I/O ports and twenty connecting wires for devices. In a fully loaded system, there are 16 ROM and 16 RAM chips with applications as :

- Gas Pumps traffic light
- Digital Scales taxi meters
- Medical Instruments
- Elevator Control vending machines

❑ **Check Your Progress – 3 :**

1. The 4–b computer was designed with _____ pin packages.

   a. 8   b. 32   c. 16   d. 64

2. The Chip 4004 is a :

   a. RAM chip   b. ROM chip

   c. Microprocessor   d. all

3. A fully loaded system will have _____ ROM and ____ RAM chips.

   a. 16, 16   b. 8, 16   c. 8, 8   d. 32, 16

6.5 Let Us Sum Up :

In early sixties, the logic gates came in existence. The integration of logic gates on single chip was designed. This was known as Small Scale Integration (SSI). During the later period the era of Medium Scale Integration (MSI) came in existence.

The MSI comprised of more than hundred components mounted on a single chip. The example of MSI is decade counter (IC 7450). Later on, the technology was advanced to the extent that it was possible to mount thousands of gates on a single chip. This integration was known as Large Scale Integration (LSI). The LSI technology was advanced to Very Large Scale Integration (VLSI). In VLSI, more than ten thousand components can be embedded in a single integrated circuit.

A microprocessor is an essential component of computer architecture, without which you will be unable to perform any tasks on your computer. It is a programmable device that accepts input, performs arithmetic and logical operations on it, and produces the desired output.

A microcomputer is a system that includes at a minimum a microprocessor, programme memory, data memory, and input-output (I/O). Timers, counters, analogue-to-digital converters (ADCs), and other components are included in some microcomputer systems. As a result, a microcomputer system can range from a large computer with hard discs, floppy discs, and printers to a single-chip embedded controller.

The MCS-4 (Micro-Computer Set-4) or 4000 Series or Busicom Chip Set was a family of Intel's 4-bit microprocessor chipsets. The entire chipset was made up of four individual chips, one of which was the 4004 CPU, the first commercial microprocessor.

6.6 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (a),   2. (b)

❑ **Check Your Progress 2 :**

1. (a)

- **Check Your Progress 3 :**

  1. (c), 2. (c), 3. (a)

6.7 Glossary :

1. **Stack memory** – It is limited only by the size of memory.
2. **INTR** – It is a maskable 8080A compatible interrupt.
3. **Trap** – It is a non–maskable interrupt.

6.8 Assignment :

Discuss the registers of 8085 microprocessors. Also explain the applications of microprocessors.

6.9 Activities :

Prepare a report on 8086 microprocessors using Internet in not less than 100–150 words.

6.10 Case Study :

Obtain information about LSI and VLSI and prepare a report on the same.

6.11 Further Readings :

1. Systems Programming, John J. Donovan, Mc–Graw Hill Publishing
2. Microprocessor and Assembly Language Programming, D. A. Godse A. P. Godse Technical Publications
3. Principles of System Programming, R. M. Graham, Wiley Publishing

Unit 07

THE 8085 MICROPROCESSOR

UNIT STRUCTURE

7.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The features of Microprocessors
- The instruction cycle of 8085
- About registers in 8085
- The architecture of 8085
- The machine cycle status and clock cycle of 8085

7.1 Introduction :

In the previous unit, you have studied the history and overview of different types of microprocessors. In continuation with the same, we will now concentrate upon the other type of microprocessor called 8085. We will also be discussing about the different types of registers in this architecture, the machine cycle status and clock cycle. The detail study of its architecture is also provided in this unit which will definitely help you to gain substantial knowledge about 8085 microprocessor.

7.2 8085 Microprocessors :

The 8085 is 8 bit, 40 pin, 3 MHz general purpose microprocessor manufactured by Intel, capable of addressing 64 Kilobyte of memory.

The 8085 microprocessor has following limitations :–

- The low order address bus is multiplexed with data bus. Therefore it is required to demultiplex these signals.
- Appropriate control signals are required to be generated for interfacing 8085 with memory and IO devices.

❖ **Multiplexing and De–multiplexing :**

As mentioned above, the low order address signals (A0–A7) are multiplexed with data bus signals (D0–D7). This gives combination of address–cum–data signals, AD0–AD7. Hence DE multiplexing is applied for such signals.

The signal lines AD0 to AD7 works in both the directions. They give out double purpose. They are applied to as small order address bus as well as data bus. The execution of instruction in previous part of cycle will allow lines to be used for low order data bus. In the second half of cycle, such lines are used as data bus, which is formally known as multiplexing.

The process of separating address–cum–data signals, AD0–AD7 into address signals A0–A7 and data signal D0–D7 is known as DE multiplexing. The DE multiplexing is performed by the circuit called latch.

Address Latch Enable (ALE) is constructive going pulse produced every time when 8085 begins operation. It shows that the bits on AD0–AD7 are address bits. This low order latch signal will be multiplexed by bus which produces different set of address lines ranging from A0 to A7.

❑ **Check Your Progress – 1 :**

1. Which is incorrect about 8085 microprocessor ?

   a. 16 bit   b. 40 pin

   c. 3 MHz speed   d. none

2. The 8085 can address _____Kilobyte of memory.

   a. 8   b. 16   c. 32   d. 64

7.3 Registers in 8085 :

The given figure shows various programmable registers in 8085 microprocessor system.

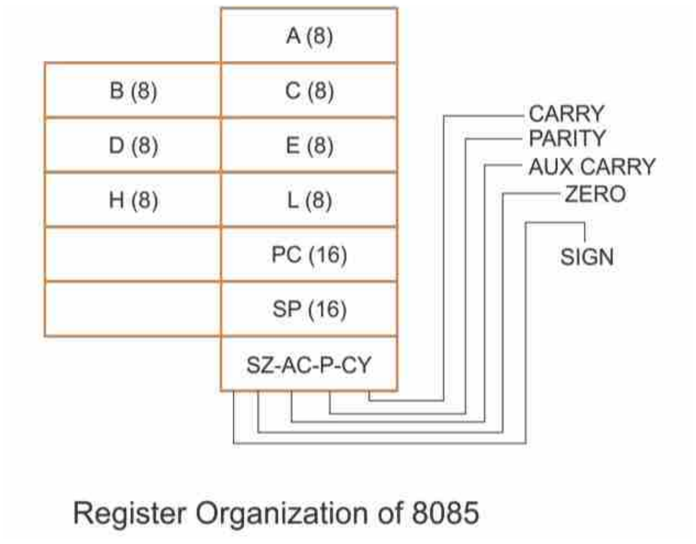


Fig. 7.1 8085 Programmable Registers

The 8085 Microprocessor System (MPS) comprises of the following registers :

1. **Accumulator :** Accumulator is the 8 bit register that performs arithmetic and logical operations and store the 8 bit data. The accumulator can also be used for temporary storage of 8 bit data.

2. **General Purpose Registers (B, D, H, C, E, L) :** The general purpose registers are 8 bit registers used for temporary storage of 8 bit data. These registers are programmable. That is, we write a source code using the general purpose registers. For example, in order to copy the contents of register A to register B, we write the instruction : MOV B, A.

   The register pairs B–C, D–E and H–L can be used as a 16 bit register to store a 16 bit data.

3. **Flags :** The ALU includes 5 types of flip flops that are set or reset according to data conditions in accumulator and other registers. For example, i) when carry is generated from addition of two numbers, the flip flop called carry flag is set to one. ii) When any arithmetic operation results in zero, the flip flop called carry flag is set to one. There are five flags in 8085 architecture – Sign (S), Zero (Z), Carry (CY), Auxiliary Carry (AC) and Parity (P).

   The flag is the eight bit register adjacent to accumulator. The five bit positions out of eight are stored in flag from output of flip flop. The programmer tests these positions through instructions. For example, the instruction JZ changes the program sequence if zero flag is set.

4. **Program Counter (PC) :** We have studied that there are 16 bit address lines in 8085 Microprocessor system. Program Counter (PC) is a memory pointer that points the memory address from which the next byte is to be fetched. When the byte is being fetched, the PC is incremented by one to point out next memory location. Obliviously, PC is a 16 bit register that deals with sequencing of execution of instructions.

5. **Stack Pointer (SP) :** It is also a 16 bit register used as a memory pointer. It points a memory location in R/W memory called stack. The beginning of the stack is defined by loading a 16 bit address in the SP register.

❖ **8085 pin out :**

Fig. 7.2 shows the pin connection diagram of 8085 microprocessor. There are 40 pins provided for various signals. According to functions, these pins are formed in following groups :

Address Bus – A0 to A8 are high order, unidirectional address bus pins.

Multiplexed Address/Data bus – The pins AD0 to AD7 are provided for address–cum–data signals.

Control and Status Signals – This group comprises of the following pins.

ALE – Address latch enable

RD Read – It is an active low read signal which shows selected I/O or memory device to read as well as to arrange data on data bus.

WR Write – It is used for active Write control signal which shows data on data bus that to be written in preferred memory or I/O device.

IO/M – This signal is used to discriminate among I/O as well as Memory operation. When signal is high, then it is I/O operation, when low, then shows memory operation.

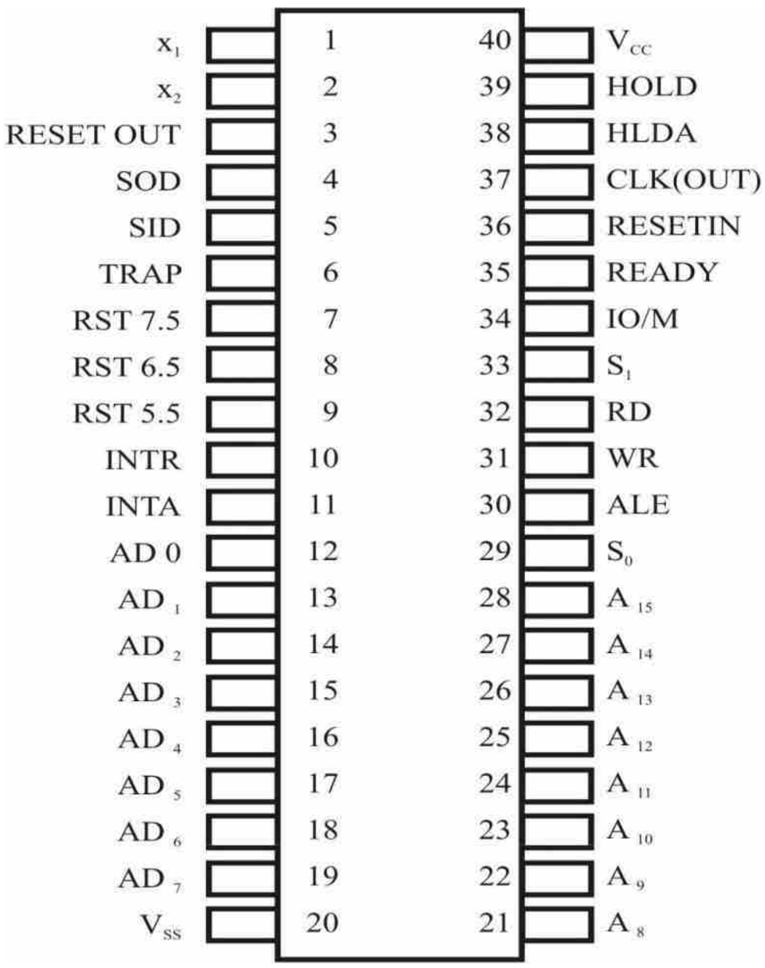


Fig. 7.2 8085 Pin Out

S1 and S0 – These are status signals which identify various operations. These operations are shown in Table 7.1

Power Supply and Clock Frequency

Vcc : Here the voltage will be + 5 Volt power supply.

Vss : This is the ground reference point.

X1, X2 : These are the connection required for a crystal.

Frequency is internally divided by two so to work with a system of 3 Mhz, the crystal frequency should be 6 MHz.

CLK (OUT) : It is the output of the clock. In this, the signal uses system clock for certain devices.

Table 7.1 8085 Machine Cycle Status and Control Signals

	Status			*Controls*		
Machine cycle	$IO/\overline{M}$	S_1	S_0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
Opcode Fetch (OF)	0	1	1	0	1	1
Memory Read	0	1	0	0	1	1
Memory Write	0	0	1	1	0	1
I/O Read (I/OR)	1	1	0	0	1	1
I/O Write (I/OW)	1	0	1	1	0	1
Acknowledge of INTR (INTA)	1	1	1	1	1	0
BUS Idle (BI) : DAD	0	1	0	1	1	1
ACK of RST, TRAP	1	1	1	1	1	1
HALT	Z	0	0	Z	Z	1
HOLD	Z	X	X	Z	Z	1
X ⇒ Unspecified, and Z ⇒ High impedance state						

Interrupts and externally initiated operations execution. These five signals are INTR, $\overline{\text{INTA}}$, RESET, HOLD and READY.

INTR is the interrupt request input signal. This signal is acknowledged by 8085 as $\overline{\text{INTA}}$ that is interrupt acknowledged. To respond HOLD request, it has the signal HLDA called hold acknowledge.

The RESET signal has two signals RESET IN and RESET OUT as described below :

$\overline{\text{RESET IN}}$

If there is a low signal on the pin, program counter is set to 0, further, the buses, tri–stated as well as microprocessor unit are all set to 0.

$\overline{\text{RESET OUT}}$

This signal indicates that the microprocessor unit is being reset.

Table 7.2 shows various interrupt signals for 8085 and externally initiated signals.

Name of the Signal	Input / Output	Description
INTR	Input	Interrupt request. This is used as a general purpose interrupt.
$\overline{\text{INTA}}$	Output	Interrupt Acknowledge. It is provided to acknowledge interrupt request (INTR)
RST 7.5	Input	Restart interrupt. It is a vector interrupt and transfers the program control to specific memory location. It has priority that INTR interrupt.
RST 6.5	Input	Restart interrupt. It is a vector interrupt and transfers the program control to specific memory location. It has priority that INTR interrupt.
RST 5.5	Input	Restart interrupt. It is a vector interrupt and transfer the program control to specific memory location. It has priority that INTR interrupt. Amongst the three, the priority order is 7.5, 6.5 and 5.5.
TRAP	Input	It is the non–maskable interrupt having topmost priority.
HOLD	Input	This signal indicates that the peripherals such as DMA (Direct Memory Access) controller are requesting the use of address and data buses.
HLDA	Output	It acknowledges the hold request.
READY	Input	It delays the microprocessor Read/Write cycles till the time a slow moving device sends or accepts data.

❖ **Serial I/O Ports :**

The 8085 has two signals to implement the serial transmission – SID (Serial Input Data) and SOD (Serial Output Data).

❑ **Check Your Progress – 2 :**

1. The accumulator can store up to _____ bit temporary data.

   a. 16 b. 8 c. 32 d. 64

2. Stack Pointer is a _____bit register used as a memory pointer.

   a. 16 b. 8 c. 32 d. 64

7.4 8085 Architecture :

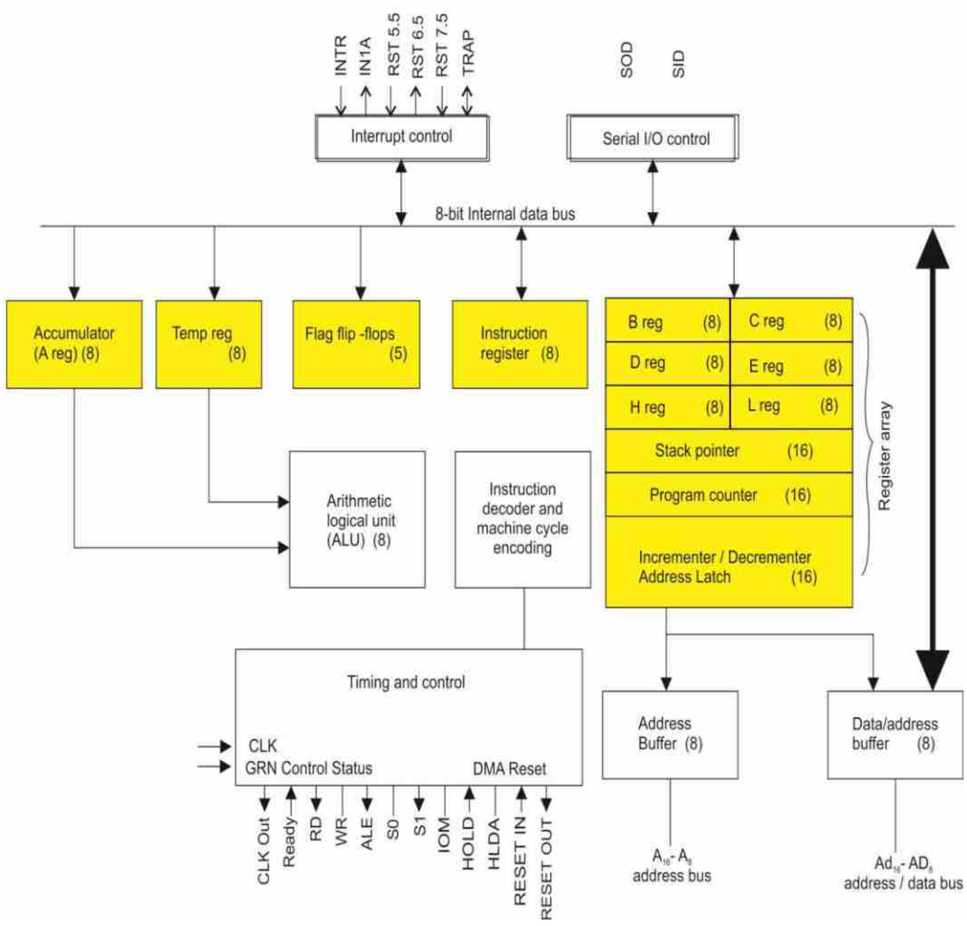


Fig. 7.3 8085 Architecture

The figure shows internal architecture of 8085 processor. It comprises of Arithmetic and Logic Unit (ALU), timing and control unit, Instruction register and decoder, Register array, Interrupt control and serial I/O/ control.

❖ **ALU :**

Arithmetic and Logic Unit will do all arithmetic and logic operations. The ALU carries an accumulator, general purpose register, arithmetic and logic circuit and five flags. In this, the temporary register will keep the data at the time of arithmetic and logic operation.

Here we find that the flags get deflected with arithmetic and logic operation as performed in ALU. After the operations, the result gets saved in accumulator. The details of flag operation are as follows :

s	z		Ac		P		CY
D7	D6	D5	D4	D3	D2	D1	D0

Sign Flag (S) : From the above arrangement, if output of arithmetic or logic operation contains bit D7 as 1, the sign flag is set. If the bit D7 is zero, the sign flag is reset. This flag is used with signed numbers. In a given byte if D7 is 1, the number will be considered as a negative number. On other hand, if bit D7 is zero, the number will be treated as positive number. In arithmetic operations with signed numbers, bit D7 is reserved for indicating sign and the remaining 7 bits are used to represent the magnitude of a number.

Zero Flag (Z) : Such flag will hold the answer of arithmetic and logic operation as zero. In this, the zero flag is reset if result will not be zero. Such types of flags gets altered by results inside accumulator along with other registers.

Auxiliary Carry (AC) Flag : From the above arrangement, when a carry occurs in arithmetic operations from digit D3 which is forwarded to D4, then the flag gets set. It is an internal flag for Binary Coded Decimal operations.

Parity Flag (P) : The result having even 1s after arithmetic or logical operation, the flag is set and when result contains odd numbers of 1s, then the flag gets reset.

Carry Flag (CY) : Once the result is a carry after arithmetic operation, it sets the carry flag otherwise the flag gets reset. Such flag will act as borrow flag in subtraction.

Timing and Control Unit : Here the synchronization exists in all 8085 operations with a clock that carries control signals for communication that exists among microprocessor and other devices.

❖ **Instruction Register and Decoder :**

The registers and decoders are the part of ALU. The decoder decodes the instruction and establishes the sequence of events to follow. The instruction register isn't programmable and cannot be accessed via any instruction.

❖ **Register Array :**

Such registers are used for holding 8–bit data during the execution of some instruction. However, since they are used internally they are not available to the programmer.

❖ **Instruction Cycles, Machine Cycles and T states :**

A program or instructions are stored in the memory of a microcomputer.

To execute an instruction, the MPU must locate the memory location, fetch the code via data bus, decode it in the instruction register and perform the function specified in the code. In 2–byte and 3–byte instructions, the subsequent codes are fetched, decoded and executed in the same way as the first code. All these operations are performed at the given moment and within a given period. The timing is provided by the clock of the system and the sequencing is done by the control unit of the microprocessor.

The 8085 identifies various operations called machine cycles through three status signals :

$IO/\overline{M}$, S0 and S1 as follows :

Table 7.3 8085 Operations

The 8085 Operation	**Status**			**Control**	
	$IO/\overline{M}$	S1	S0	$\overline{RD}$	$\overline{WR}$
Opcode Fetch (OF)	0	1	1	0	1
Memory Read (MR)	0	1	0	0	1
Memory Write (MW)	0	0	1	1	0
I/O Read (IOR)	1	1	0	0	1
I/O Write (IOW)	1	0	1	1	0

In addition to these operations, 8085 performs other operations such as interrupt acknowledge, which are not shown here for clarity.

❖ **Timing Cycle for instruction MOV C, A :**

Let us suppose that the instruction MOV C, A (Code 4FH) is stored at memory location 2005H. The accumulator has data byte 7AH. We have to illustrate the execution of instruction and calculate execution time if the system clock frequency is 2 Mhz.

Memory Location	Machine Code
2005	4F

The execution of all the instructions is sequenced by the program counter.

Before execution of instruction, the memory address must be in the program counter. Before the execution of an instruction, its memory address must be in its program counter. It is assumed that the instruction located in the previous memory location 2004H is already executed and that the program counter has address 2005 H. To execute the instruction located at 2005 H, the following sequence takes place. See the Figure 3.3.

- The 8085 places the contents of program counter (2005 H) on the address bus 20H on the high order bus and 05 H on the low order bus AD0–AD7.
- The 8085 causes the ALE signal to go high, that can be used to demultiplex the bus.
- The 8085 identifies the nature of machine cycle – Opcode Fetch (Instruction fetch) – by using status signals. ($IO/\overline{M}$ = 0, S1 = 1, S0 = 1).
- During T2, the 8085 send control signal $\overline{RD}$ to enable memory and increment the program counter by 1 to 2006 H. The controls of memory location 4FH (MOV C, A) are placed on data bus.
- The 8085 reads the byte 4FH and places it in instruction register T3.
- The 8085 decodes the instruction, places the accumulator content (7AH) in the temporary register and then transfers it to register C (T4).

❖ **8085 timing for execution of instruction MVI A, 32 H :**

In the following example, the execution of a 2–byte instruction MVI, A, 32 H (load the accumulator with the data 32 H) is explained with instruction cycle, machine cycles and T states.

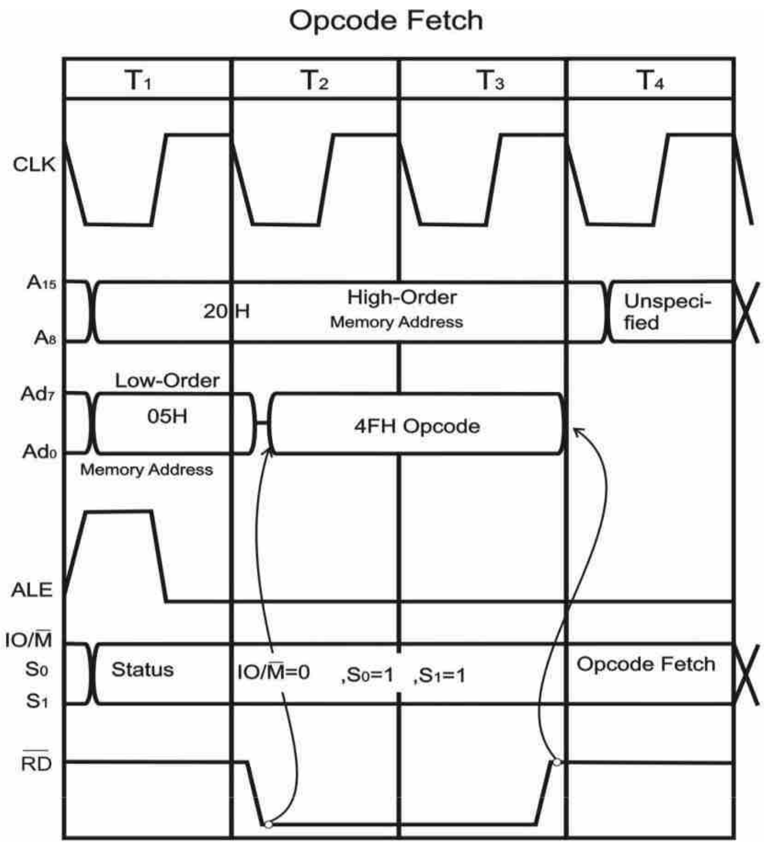


Fig. 7.4 Timing Cycle for instruction MOV C, A

The memory locations and machine codes are as follows :

Memory Location	Machine Code	Mnemonics
2000	3 E	MVIA, 32 H
2001	3 2	

This instruction requires two machine cycles with a total of seven T–states. The first machine cycle M1 has four T states, the second machine cycle M2 has three T states. See Figure 7.4.

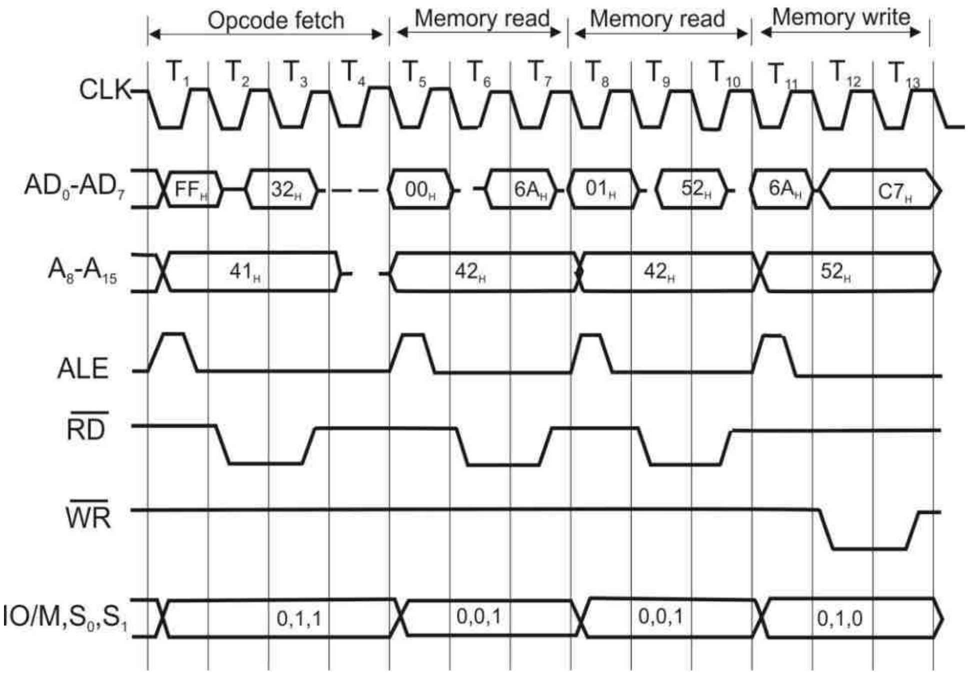


Fig. 7.5 8085 Timing for execution of instruction (MVI A, 32 H)

The machine cycle M1 is identified as Opcode fetch cycle by signals $IO/\overline{M}$ = 0, S1 = 1, S0 = 1. During the state T1 the high order address 20 H is placed on the bus AD7–AD0 and the ALE (Address Latch Enable) signal is made to go high. In the state T2, the control signal $\overline{RD}$ goes low and the data 3EH from the memory location 2000H is placed on the data bus. The fetch cycle is completed in the state T3 and the instruction is decoded in the state T4. During the state T4, the contents of bus are unknown.

In the machine cycle M2 which is identified as a Memory Read Cycle ($IO/\overline{M}$ = 0, S1 = 1, S0 = 0) is similar to M1 cycle. In this cycle, the address 2000H is placed on the bus in M1, the data byte 32 H is fetched via the data bus and the instruction is executed during the T3 state. Machine cycles M1 and M2 are similar, they both perform the memory read function. However the first machine cycle of each instruction is executed during the T3 state. Machine cycles M1 and M2 are similar, they both perform the memory read function. However, the machine cycle of each instruction is identified as Opcode Fetch cycle rather memory read cycle.

From the above examples, the following points can be summarized :

- The instruction cycle consists of the execution of one or more machine cycles and the execution time is determined by T states.
- The machine cycle is defined as a sequence of operations required to complete one of the functions.
- The T state is internally defined operation and is synchronized with the system clock.
- The type of operation is defined at the beginning of the machine cycle by the status signals.
- The control signal identifies timing of operation.

❑ **Check Your Progress – 3 :**

1. Which is incorrect in case of an ALU in 8085 ?

   a. It has an accumulator

   b. It carries a general purpose register

   c. It contains arithmetic and logic circuit

   d. It carries 4 flags

2. The zero flag is set if :

   a. result of arithmetic operation is zero

   b. result of logic operation is zero

   c. result of arithmetic and logic operation is zero

   d. result of arithmetic and logic operation is one

7.5 Let Us Sum Up :

The Intel 8085 microprocessor was one amongst the first microprocessors released by Intel in the late 1970s. The Intel 8085 microprocessor was the successor to Intel's 8080 processing unit, whose most notable features were that of increased compatibility and a much simpler voltage requirement. The Intel 8085 microprocessor saw its greatest use not in the form of a system processor, but as a microcontroller for several other hardware applications.

The Intel 8085 was an 8–bit microprocessor design. This is to state that the biggest chunk of information the 8085 could process at any given time was limited to 8 bits. The 8085 microprocessor came in three variants : the 8085A, 8085AH and M8085AH. The difference between these three variants is their clock operating speed of 3, 5 and 6 MHz respectively. The final distinguishing characteristic of the 8085 microprocessor was that of the single +5 Volt rail requirement and its 6500 transistor count—500 more than its predecessor.

The 8085 microprocessor actually saw very little use as a central processing unit. Rather, it's most prevalent implementation was that of a microcontroller.

The operational instructions of the Intel 8085 were the same as that of its predecessor, the 8080, except for 2 new instructions. The 8085 had seven processing registers. These registers were eight bits wide and had the capability of performing 16–bit operations if they were paired together with other registers.

The 8085 is 8 bit, 40 pin, 3 Mhz general purpose microprocessor manufactured by Intel, capable of addressing 64 kilobyte of memory.

The low order address signals (A0–A7) are multiplexed with data bus signals (D0–D7). This gives combination of address–cum–data signals, AD0–AD7. Therefore it is required to demultiplex these signals.

The process of separating address–cum–data signals, AD0–AD7 into address signals A0–A7 and data signal D0–D7 is known as demultiplexing. The demultiplexing is performed by the circuit called latch.

Registers used in 8085 : The 8085 Microprocessor System (MPS) comprises of the following registers.

1. Accumulator
2. General Purpose Registers (B, D, H, C, E and L)
3. Flags
4. Program Counter (PC)
5. Stack Pointer (SP)

These registers are used to hold 8–bit data during the execution of some instruction. However, since they are used internally they are not available to the programmer. The following points should be noted :

- The machine cycle is defined as a sequence of operations required to complete one of the functions.
- The T state is internally defined operation and is synchronized with the system clock.
- The type of operation is defined at the beginning of the machine cycle by the status signals.

7.6 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (d), 2. (d)

❑ **Check Your Progress 2 :**

1. (b), 2. (a)

❑ **Check Your Progress 3 :**

1. (d), 2. (c)

7.7 Glossary :

1. **Demultiplexing –** It is a process of separating address and data signals.
2. **Address Latch Enable –** It is a constructive going pulse generated when 8085 begins operation.
3. **Accumulator –** It is an 8 bit register which does arithmetic and logical operations and stores 8 bit data.
4. **Flag –** It is an 8 bit register adjacent to accumulator.

7.8 Assignment :

Write instruction cycle, machine cycles and T states in serpent of command OUT 01H ?

7.9 Activities :

Explain the pin connection diagram of 8085 microprocessors.

7.10 Case Study :

Using proper diagram explain the internal architecture of 8085.

7.11 Further Readings :

1. Systems Programming, John J. Donovan, Mc–Graw Hill Publishing
2. Microprocessor and Assembly Language Programming, D. A. Godse, A. P. Godse Technical Publications
3. Principles of System Programming, R. M. Graham, Wiley Publishing

BLOCK SUMMARY :

The block explained about the detail knowledge and understanding of microprocessor and its techniques with stress on 8085 processor. The students will get clear idea about the architecture and pin layout of 8085 processor. The block focuses more on working and features of 8085 with knowledge about its programming instructions. The students have been practically demonstrated with circuit diagrams about the uniqueness and characteristics of 8085 microprocessor. The students will give extra practical examples and exercises that will help to learn and grab the subject easily.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Explain the control bus of 8085 ?
2. Explain the memory of 8085 microprocessor ?
3. Explain registers ?
4. Explain the architecture of 8008 microprocessor ?
5. Explain SSI and MSI in detail ?

❖ **Long Questions :**

1. Discuss the registers of 8085 microprocessors. Also explain the applications of microprocessors ?
2. Explain the internal working of microprocessors ?
3. Discuss various operations of instructions cycles ?

❖ **Enrolment No. :**

1. How many hours did you need for studying the units ?

Unit No.	5	6	7
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :

Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	
Language and Style	☐	☐	☐	☐	
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	
Conceptual Clarity	☐	☐	☐	☐	
Check your progress Quest	☐	☐	☐	☐	
Feed back to CYP Question	☐	☐	☐	☐	

3. Any other Comments

....

....

....

....

....

....

....

....

Dr. Babasaheb Ambedkar Open University Ahmedabad **BCAR-303**

System Programming and Introduction to Microprocessor

BLOCK 3 : 8085, 8086/8088 PROGRAMMING

8085, 8086/8088 PROGRAMMING

Block Introduction :

Interrupts are the input signals that suspend the operation of the program and transfer the control to the subroutine called Interrupt Service Routine (ISR). After execution of ISR, the control again comes back at the location at which interrupt signal was inputted.

In this block students will be aware of about 8085 Interrupts and 8259 programmable Interrupt controller. The students will find the block effective as it explains about interfacing techniques of 8251 programmable communication interface.

The block will detailed about addressing modes and use of 8085 instruction set. The students will details with 8085 instruction set with an understanding of 16 bit microprocessor. The students will be trained with practical examples and exercises that will help to learn and grab the subject easily.

Block Objectives :

After learning this block, you will be able to understand :

- 8085 Interrupts
- Idea about 8259 Programmable Interrupt Controller
- Detail of 8251 Programmable Communication Interface
- Generalization of Direct Memory Access (DMA)
- Concept of Addressing Modes
- Idea about 8085 Instruction Set
- Knowledge about 16 Bit Microprocessors

Block Structure :

Unit 08

INTERRUPTS

UNIT STRUCTURE

8.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The concept of vector interrupt input signals.
- About hardware and software interrupt signals.
- The priority analysis of input signals.
- How to enable and disable interrupt input signals.
- SIM and RIM instructions.
- The working of 8259 Programmable Interrupt Controller.
- The working of 8251 Programmable Communication interface.
- The concept of maskable and non–maskable interrupts.

8.1 Introduction :

Interrupt is a procedure through which an input/output or instruction will remove the standard working of processor and gets charged itself. Normally it is seen that a type of work is alloted to particular interrupt signal, where the data transfer occurs among the peripheral devices and microprocessor. Here the control gets transferred to subroutine and is termed as Interrupt Service Routine.

After execution of ISR, the control again comes back at the location at which interrupt signal was inputted. This unit will deal with interrupts. We will also discuss about the concept of direct memory access.

8.2 8085 Interrupts :

We have studied that the 8085 has interrupt inputs namely TRAP, RST 7.5, RST 6.5 and RST 5.5. The call locations of these interruption signals are given below in Table 8.1.

Table 8.1 8085 Interrupt Input Signals

Name of the signal	Call location	Description
INTR	–	Interrupt request. This is used as a general purpose interrupt.
RST 7.5	003CH	Restart interrupt. It is a vector interrupt and transfers the program control to specific memory location. It has priority that INTR interrupt.
RST 6.5	0034 H	Restart interrupt. It is a vector interrupt and transfers the program control to specific memory location. It has priority that INTR interrupt.
RST 5.5	002C H	Restart interrupt. It is a vector interrupt and Programming transfers the program control to specific memory location. It has priority that INTR interrupt. Amongst the three, the priority order is 7.5, 6.5 and 5.5.
TRAP	0024H	It is the non–maskable interrupt having topmost priority.

❖ **TRAP :**

TRAP is the non–maskable interrupt known as NMI. It has highest priority. It need not be enabled and cannot be disabled. It is level and edge sensitive, meaning that the input should go high and stay high to be acknowledged. It cannot be acknowledged again until it makes transition from high to–low–to–high.

When this interrupt is triggered, the program control is transferred to the location 0024 H without any external hardware or interruption enables instruction.

TRP is generally used in case of critical events such as power failure and emergency shut off.

❖ **RST Interrupts :**

We have studied that RST 7.5, RST 6.5 and RST 5.5 are the vector interrupt signals having priority order 7.5., 6.5, 5.5. These signals are RST hardware interrupts. 8085 instruction set also provides RST instructions, called RST software interrupts. These are 1 byte call instructions and transfer the program execution to a specific location on page 00H as listed in table 8.2

Table 8.2 Restart Instructions

Software Interrupt	Hex Code	Call location in hex
RST0	C7	0000
RST1	CF	0008
RST2	D7	1010
RST3	DF	0018
RST4	E7	0020
RST5	EF	0028
RST6	F7	0030

❖ **EI and DI Instructions :**

The interrupt process should be enabled by writing the instruction EI in the main program. The instruction EI sets the interrupt enable flip–flop. The instruction DI resets flip–flop and disables the interrupt process. EI and DI are one byte instructions. The instruction DI should be included in a program where interrupt from outside source cannot be tolerated.

❖ **Set Interrupt Mask (SIM) :**

It is a 1 byte instruction. This instruction reads the contents of accumulator and enables or disables the interrupts according to the contents of accumulator.

Bits D7 and D6 are used for serial IO and do not affect interrupts. D6 = 1 enables serial IO. Bit D3 is the control bit and it should be equal to 1 in order for bits D0, D1 and D2 to be effective. Logic 0 in D0, D1 and D2 will enable the corresponding interrupts and logic 1 will disable interrupts. Bit D4 is an additional control for RST 7.5. If D4=1, the RST 7.5 flip–flop is reset. This is used to override RST 7.5 without servicing it. See Fig. 8.1

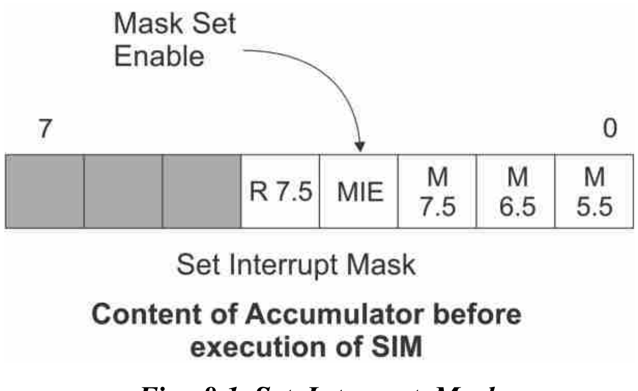


Fig. 8.1 Set Interrupt Mask

❖ **Read Interrupt Mask (RIM) :**

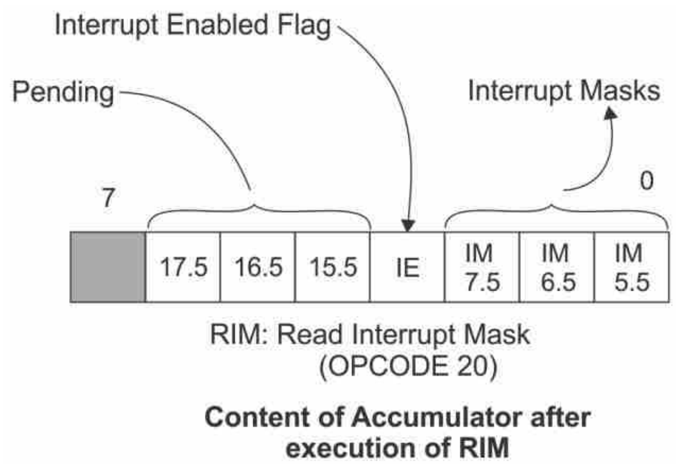

Interrupt Enabled Flag
Pending
Interrupt Masks
7
0
17.5
16.5
15.5
IE
IM 7.5
IM 6.5
IM 5.5
RIM: Read Interrupt Mask (OPCODE 20)
Content of Accumulator after execution of RIM

Fig. 8.2 Read Interrupt Mask

It is a 1 byte instruction. It loads the accumulator with 8 bits indicating the current status of the interrupt masks. If it is pending, enable RST 6.5 without affecting any other interrupts, otherwise return to the main program.

❑ **Check Your Progress – 1 :**

1. Which is not an 8085 interrupt ?

   a. RST 7.5 b. RST 6.5 c. RST 5.5 d. RST 4.5

2. The call location 0034 H is of which interrupt ?

   a. RST 7.5 b. RST 6.5 c. RST 5.5 d. RST 4.5

8.3 8259 : Programmable Interrupt Controller :

8259 is a programmable interrupt–managing device. It is particularly designed to use interrupt signal, INTR. The primary features of 8259 are as follows :

- It manages 8 interrupt requests.
- It vectors the interrupt as required anywhere in memory map by way of program control without extra hardware.
- It solves eight levels of interrupt priorities in different modes.
- In this the priority scheme gets enlarged up to 64 levels.

Implementing interrupts in a simplest format without cascading requires two specific instructions. The instructions are written by the processor in the device registers. The first instruction specifies features such as mode and/or memory space between two consecutive interrupt levels. The second instruction specifies high order memory address. After these instructions have been written, the following sequence of events should occur. See Fig 8.3.

- One or more interrupt lines go high requesting the service.
- The 8259 resolves the priorities and sends INT signal to microprocessor.
- The processor acknowledges the interrupt by sending INTA.
- After INTA have been received, the opcode for the CALL instruction (CDH) is placed on the data bus.
- Because of CALL instruction, the processor sends two more INTA signals.
- At first INTA, the 8259 places low order 8 bit address of interrupt vector. This completes the 3–byte CALL instruction.

- The program sequence of the processor is transferred to the memory location specified CALL instruction.

The 8259 includes additional features such as reading the status and changing the interrupt during the program execution.

❖ **Read / Write Logic :**

When the address line A0 is at logic 0, the controller is selected to write a command or read a status. The chip select logic and A0 determine the port address of controller.

❖ **Control Logic :**

This block has two pins : INT (Interrupt) as an output and $\overline{\text{INTA}}$ (Interrupt Acknowledge) as an input. The INT is connected to the interrupt pin of the microprocessor. Whenever, the valid interrupt is asserted, this signal goes high. The $\overline{\text{INTA}}$ is the interrupt acknowledge signal from the processor.

❖ **Interrupt Registers And Priority Resolver :**

The Interrupt Request Register (IRR) has eight input lines (IR0–IR7) for interrupts. When these lines go high, the requests are stored in the register. The in–service register (ISR) stores all the levels that are currently being serviced; the interrupt mask register (IMR) stores the masking bits of the interrupt lines to be masked. The priority resolver (PR) examines these three registers and determines whether INT should be sent to the processor.

❖ **Cascade Buffer / Comparator :**

This block is used to expand the number of interrupt levels by cascading two or more 8259s.

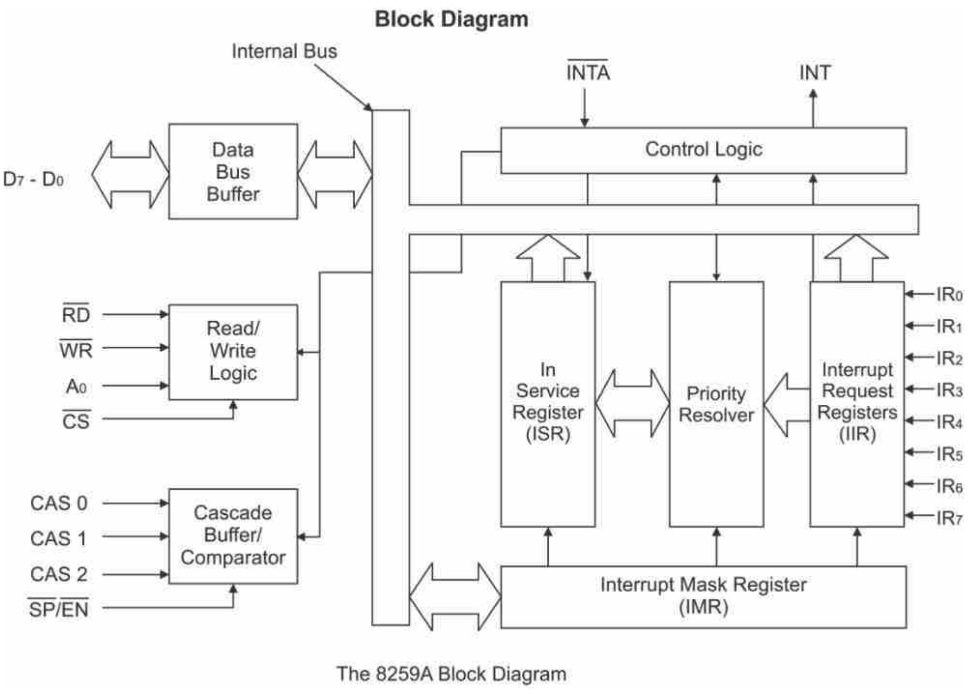


Fig. 8.3 8259 Programmable Interrupt Controller

❑ **Check Your Progress – 2 :**

1. Interrupt Request Register has ______ input lines for interrupts.

a. 4 b. 8 c. 16 d. 32

8.4 8251 : Programmable Communication Interface :

It is a programmable chip designed for synchronous and asynchronous serial data communication, packaged in 28 pin DIP. The figure shows the block diagram of 8251. It includes following sections :

- Read/Write Control Logic
- Transmitter / receiver
- data bus buffer
- modem control

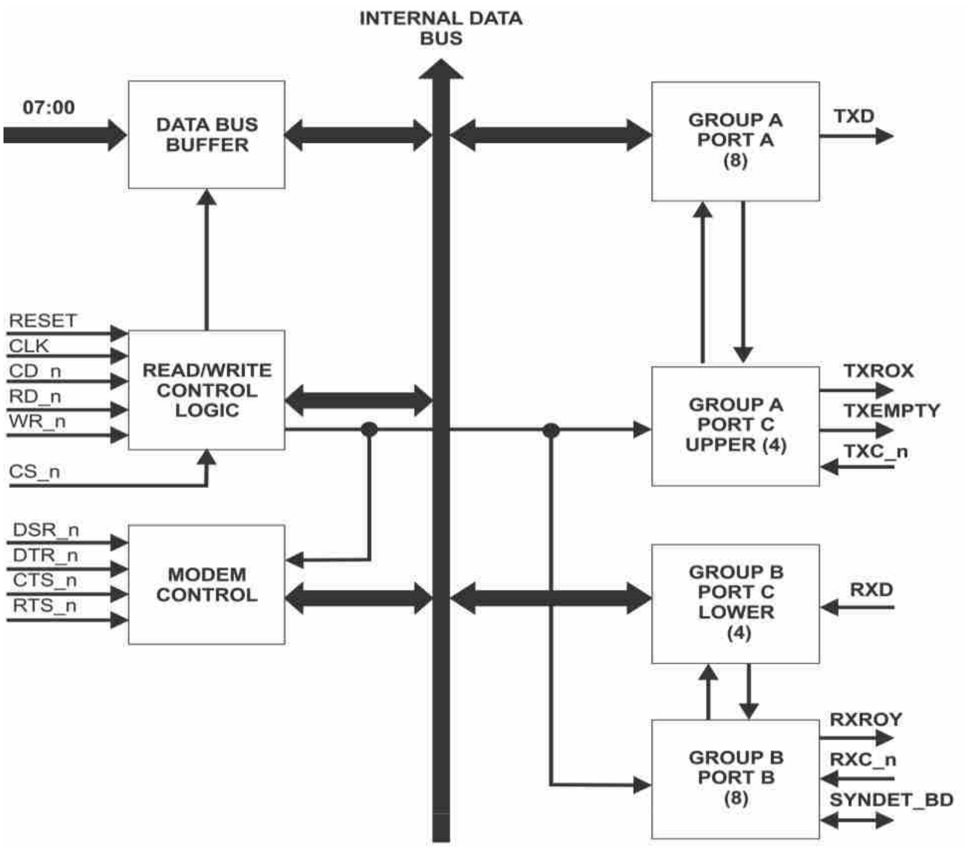


Fig. 8.4 : Programmable Communication Interface

From the figure it is analysed that control logic interfaces a communicable chip that carries a processor which shows the working of chip as per the control word in its register and monitors the data flow. 8251 transmission section be able to transform parallel word that is accepted from processor in serial bits additionally will transmits at TxD line to peripheral

The receiver section of 8251 will get serial bits from peripheral section which further will transform into parallel a word which again sends it to CPU. Here we see that a modem control is applied in order to set up data communication all the way through modem with the help of telephone lines. It is analysed that 8251 appears to be a multifaceted device which is competent of doing different functions. In this, the asynchronous mode is frequently applied for data communication which is flanked by CPU as well as serial peripherals like terminals along with floppy disks. The brief summary about interfacing and control signals of 8251 is shown :

Table 8.3 Summary of Control Signals for 8251

$\overline{CS}$	$C/\overline{D}$	$\overline{RD}$	$\overline{WR}$	Function
0	1	1	0	Processor writes instructions in control register.
0	1	0	1	Processor reads status from status register.
0	0	1	0	Processor outputs data to the data buffer.
0	0	0	1	Processor accepts data from data buffer.
1	X	X	X	USART is not selected.

❖ **Transmitter Section :**

It comprises of 3 output signals and 1 Input/output signal such as :

- **TxD :** It is a signal that transmits serial bits.
- **TxC :** It is an input signal controls which transmit the bits by USART with clock frequency of 1, 16 and 64.
- **TxRDY :** it is an output signal which shows transmitter ready. In this, if transmitter is high, it shows empty buffer register. It can be reset when data byte is loaded in buffer.
- **TxE :** It is also an output signal which shows that transmitter is empty. Here Logic1 shows that output register is empty. Such type of signal gets reset if the byte is moved from buffer to output register.

❖ **Receiver Section :**

- **RxD :** Here the bits are received serially which gets transformed into parallel byte in receiver input register.
- $\overline{\textbf{RxC}}$: It is a clock signal which can control the bits receiving rate by USART. In asynchronous mode, clock sets to 1, 16 or 64 times baud.
- **RxRDY :** It is an output signal which becomes high when USART has character in buffer register.

❑ **Check Your Progress – 3 :**

1. In 8251 control signal, the processor will write the instructions in control register, when read and write signals are :

   a. 1, 0 b. 1, 1 c. 0, 1 d. 0, 0

2. In transmitter section of 8251, it has ___ output signals and ___ Input/output signal.

   a. 3, 2 b. 3, 1 c. 2, 3 d. 1, 3

8.5 Direct Memory Access (DMA) :

The Direct Memory Access (DMA) is the process of communication of data controlled by an external peripheral. In situations in which the processor controlled data transfer is too slow, the DMA is generally used e.g. data transfer between floppy a floppy disk and R/W memory of the system.

❖ **8237 Programmable DMA Controller :**

The 8237 Multimode Direct Memory Access Controller is a peripheral interface circuit for microprocessor systems. It is designed to improve system performance by allowing external devices to directly transfer information to or from system memory. Memory to memory transfer is also provided. The 8237 offers a wide variety of programmable control features to enhance data throughput and system optimization and to allow dynamic reconfiguration under program control. See Fig 8.5

The 8237 is designed to be used in conjunction with an external 8–bit address register such as 8282. It has 4 independent channels which are expandable to more channels through cascading extra controller chips. With the help of 3 transfer modes the programming of DMA service becomes easy. Every channel can be programmed singly just to initialize its original condition by using End of Process that carries 64 K address and word count competence.

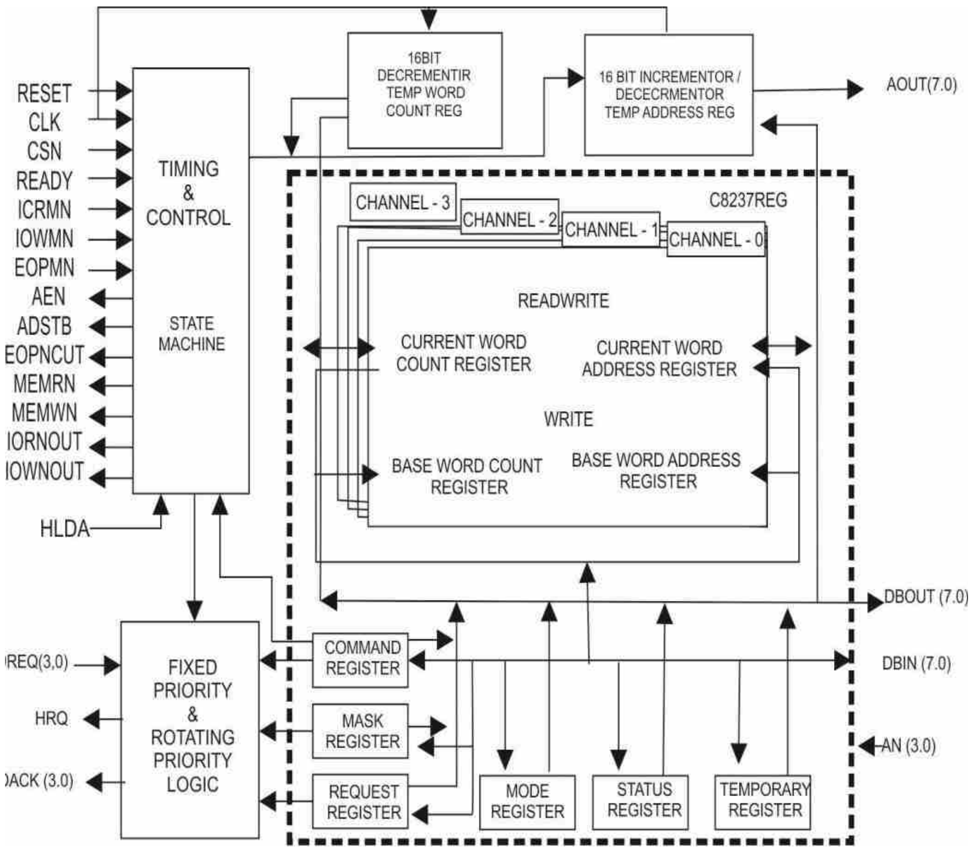


Fig. 8.5 8237 Programmable DMA Controller

The features of 8237 are as follows :

- Enable/disable control of every DMA requests.
- 4 free DMA channels
- Self–governing auto initialisation of all channels.
- Memory to Memory relocation
- Memory block initialisation
- Address / increment or decrement
- High routine relocation up to 1.6 M Bytes/sec.
- Directly flexible to some numeral of channels
- Closing stage of process input for concluding transfers
- Software DMA requirements

- Autonomous polarity organizer for DREQ as well as DACK signals The 40–pins of 8237 are defined as follows –

1. Vcc is +5 Volt DC supply
2. Vss is ground
3. **CLK (Clock Input)**

   This input controls the internal operations of 8237 and its rate of data transfers. The input may be driven at upto 3 MHz.

4. **$\overline{\text{CS}}$ (Chip Select) Input**

   It is the lively low input that selects 8237 as I/O component in idle cycle.

5. **RESET (Reset input)**

   An asynchronous high input which comprehend the command, status, appeal as well as temporary registers along with flip–flops.

6. **READY (Ready Input)**

   It extends the memory read as well as write pulses starting from 8237 to provide slow memories or else I/O peripheral devices.

❖ **Interfacing with 8085 microprocessor system :**

The figure 8.6 shows interfacing with 8085 microprocessor system. The multimode DMA controller issues the HRQ to the processor whenever there is at least one valid DMA request from the peripheral device. When the processor replies with HLDA signal, the 8237 takes the control of address bus, data bus and control bus. The address for the first transfer operation comes out in two bytes – the least significant 8 bits on the eight address outputs and the most significant 8 bits on the data bus.

The contents of data bus are then latched into the 8282 bit latch to complete full 16 bits of the address bus. The 8282 is a high speed, 8 bit three–state latch in a 20–pin package. After the initial transfer takes place, the latch is updated only after a carry or borrow is generated in the least significant address byte. Four DMA Channels are provided when one 8237 is used.

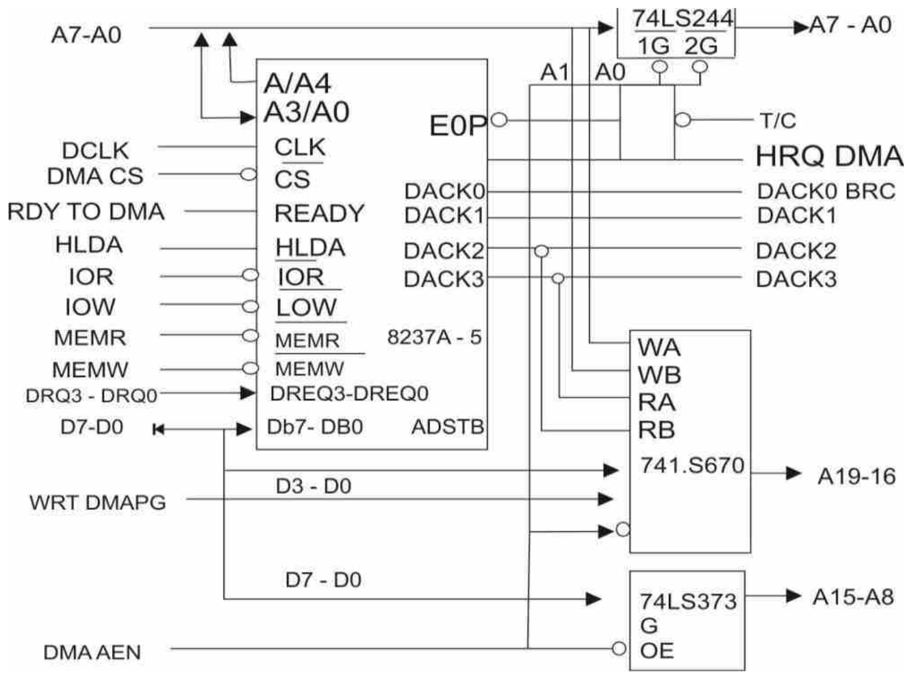


Fig. 8.6 8237 Interface with 8085

❑ **Check Your Progress – 4 :**

1. Which is not a feature of 8237 ?

   a. Enable/disable control of every DMA requests.

   b. 8 DMA channels

   c. self–governing auto initialisation of all channels.

   d. memory block initialisation

2. Which is incorrect in case of 8282 ?

   a. It has high speed   b. It carries 8 bit three–state latch

   c. It is a 20–pin package   d. none

8.6 Let Us Sum Up :

In this unit, we have learnt that, interrupts are input signals that suspend the operation of the program and transfer the control to the subroutine called Interrupt Service Routine (ISR). After execution of ISR, the control again comes back at the location at which interrupt signal was inputted.

TRAP is the non–maskable interrupt known as NMI. It has highest priority. It need not be enabled and cannot be disabled. It is level and edge sensitive, meaning that the input should go high and stay high to be acknowledged. It cannot be acknowledged again until it makes transition from high to low to high.

When this interrupt is triggered, the program control is transferred to the location 0024 H without any external hardware or interruptions enable instruction.

RST 7.5, RST 6.5 and RST 5.5 are the vector interrupt signals having priority order 7.5., 6.5, 5.5. These signals are RST hardware interrupts. 8085 instruction set also provides RST instructions, called RST software interrupts.

The interrupt process should be enabled by writing the instruction EI in the main program. The instruction EI sets the interrupt enable flip–flop. The instruction DI resets flip–flop and disables the interrupt process. EI and DI are one byte instructions. The instruction DI should be included in a program where interrupt from outside source cannot be tolerated.

Set Interrupt Mask (SIM) is a 1 byte instruction. This instruction reads the contents of accumulator and enables or disables the interrupts according to the contents of accumulator.

Bits D7 and D6 are used for serial I/O and do not affect interrupts. D6 =1 enables serial I/O. Bit D3 is the control bit and it should be equal to 1 in order for bits D0, D1 and D2 to be effective. Logic 0 in D0, D1 and D2 will enable the corresponding interrupts and logic 1 will disable interrupts. Bit D4 is an additional control for RST 7.5. If D4=1, the RST 7.5 flip–flop is reset. This is used to override RST 7.5 without servicing it.

Read Interrupt Mask is a 1 byte instruction. It loads the accumulator with 8 bits indicating the current status of the interrupt masks. If it is pending, enable RST 6.5 without affecting any other interrupts, otherwise return to the main program.

8259 is a programmable interrupt–managing device. It is specially designed to use interrupt signal, INTR.

Implementing interrupts in a simplest format without cascading requires two specific instructions. The instructions are written by the processor in the device registers. The first instruction specifies features such as mode and/or memory space between two consecutive interrupt levels. The second instruction specifies high order memory address. After these instructions have been written, the following sequence of events should occur.

- One or more interrupt lines go high requesting the service.
- The 8259 resolves the priorities and sends INT signal to microprocessor.
- The processor acknowledges the interrupt by sending $\overline{\text{INTA}}$.
- After $\overline{\text{INTA}}$ have been received, the opcode for the CALL instruction (CDH) is placed on the data bus.
- The program sequence of the processor is transferred to the memory location specified CALL instruction.

The Direct Memory Access (DMA) is the process of communication of data controlled by an external peripheral. In situations in which the processor controlled data transfer is too slow, the DMA is generally used e.g. data transfer between floppy a floppy disk and R/W memory of the system.

The 8237 Multimode Direct Memory Access Controller is a peripheral interface circuit for microprocessor systems. It is designed to improve system performance by allowing external devices to directly transfer information to or from system memory. Memory to memory transfer is also provided. The 8237 offers a wide variety of programmable control features to enhance data throughput and system optimization and to allow dynamic reconfiguration under program control. The 8237 is designed to be used in conjunction with an external 8–bit address register such as 8282.

8.7 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (d), 2. (b)

❑ **Check Your Progress 2 :**

1. (b)

❑ **Check Your Progress 3 :**

1. (a), 2. (b)

❑ **Check Your Progress 4 :**

1. (b), 2. (d)

8.8 Glossary :

1. **TRAP** – It is a non–maskable interrupt known as NMI.
2. **DMA** – It is Direct Memory Access, a process of communication of data which is controlled by external peripheral.

8.9 Assignment :

Draw the block diagram of 8251 and explain the same in your own words.

8.10 Activities :

Obtain more information about DMA from Internet and explain the same in your own words.

8.11 Case Study :

Obtain information about various interrupt masks and write the same in your own words.

8.12 Further Readings :

1. 8080/8085 Microprocessor Book, Intel Marketing Communications, Wiley–Intel Series.
2. 8085 Microprocessors : Programming and Interfacing, N. K. Srinath, PHI Learning Private Ltd.
3. 8085 Software Design, C. A. Titus, SAMS Publishing.

Unit 09

8085, 8086/8088 PROGRAMMING

UNIT STRUCTURE

9.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Various addressing modes
- One byte , two byte and three byte instruction
- Various types of instructions in 8085
- The concept of 16–bit microprocessors
- The basics of 8086/8088 processors

9.1 Introduction :

The 8085 instructions are classified as one byte, two byte and three byte. The one byte instructions have one byte opcode. For example, MOV A, B, CMA, RST0, HLT are one byte instructions. Two byte instructions have a two byte opcode. Examples of two byte instructions are MVI A, 05 H, ADI 09 H, ANI FA H. In these examples, the first byte is assigned to the instruction such as MVI A, ADI, ANI etc. The second byte is the data such as 05 H, 09 H and FA H. In case of three byte instructions, the first byte is for instruction; second and third bytes are assigned to low order and high order 16 bit number respectively.

9.2 Addressing Modes :

The various formats of specifying the sources and destinations are called addressing modes. The 8085 instruction set comprises of the following addressing modes. These are also called addressing schemes.

- Immediate addressing : MVI Rs , data
- Register addressing : MOV Rd , Rs
- Direct Addressing : IN/OUT Port #
- Indirect addressing : LDA , LDAX , STA , STAX

❑ **Check Your Progress – 1 :**

1. Which is not an addressing modes of 8085 instruction set.

a. Immediate addressing	b. Register addressing
c. Direct Addressing	d. Centre Addressing

9.3 8085 Instruction Set :

The instruction set of 8085 carries 74 operation codes which give 246 instructions. Many of the notations are used in description of instructions–

R – 8 bit register

M – Memory register

Rs – Register source

Rd – Register destination

Rp – Register pair

Data Transfer instructions

Under this category, the data is transferred or copied in following situations :

- From register to register
- Load an 8 bit number in register
- Between memory and register
- Between IO and accumulator
- Load 16 bit number in register pair

MOV Rd, Rs : This will keep content of source register Rs with destination register Rd.

MVI Rd, 8 bit : Load 8 bit data in the destination register Rd.

OUT 8 bit (Port Address) : Sends data byte from accumulator to output device.

IN 8 bit (Port Address) : Reads data byte from input device and place it accumulator.

LXI Rp, 16 bit : Load 16 bit in register pair

MOV R, M : Copy the data byte from memory location (source) to register

LDAX Rp : Copy the data byte into the accumulator from memory location indicated by the register pair.

LDA 16 bit : Copy the data byte into the accumulator from the memory location specified by the 16 bit address.

The simple example of data transfer is given below :

If we have to write instruction to read ON/OFF switches which is joined to input port is having address as 00 H, then make the device ON and join with output devices having address as 01 H. Here, port addresses is in hexadecimal with input as 8 switches attached to data bus by tri–state buffer. In such case, the microprocessor will read bit pattern of switches and send similar pattern to the output port so as to turn on. The code would be written as follows :

IN 00H	Read the contents of input port.
OUT 01H	Send the information to output port.
HLT	Stop the program.

When the microprocessor executes instruction 00H, it enables tri–state buffer. The bit pattern formed by the switch positions is placed on data bus and transferred to the accumulator. This is called reading of the input port.

When the microprocessor executes next instruction, OUT 01H, it places contents of accumulator on data bus and enables output port 01H. The output port turns ON/OFF the devices connected to it.

Arithmetic Instructions : There are four basic arithmetic instructions in 8085 instruction set these are Add, Subtract, Increment (Add 1) and Decrement (Subtract 1). The mnemonics and tasks to be performed are as follows–

ADD R This will add data of register to accumulator and keeps output in accumulator.

ADI 8 bit It adds 8–bit data to accumulator.

SUB R It will minus content of register with accumulator and keeps the output in accumulator.

SUI 8 bit This will minus 8–bit data to accumulator.

INR R Increases data of register.

DCR R Deceases data of register.

INX Rp Increases data of register pair.

DCX Rp Decreases data of register pair.

ADD M It will add memory data to place to accumulator and keeps result in accumulator.

SUB M It will minus data of memory place from data of accumulator and keeps the output in accumulator.

INR M Increment the contents of memory location.

DCR M Decrement the contents of memory location.

Logical Instructions : The logical instructions are based on the operations of logic gates such as AND, OR, NOT. The 8085 can process 5 types of logical instructions viz. AND, OR, XOR, Compare and Rotate.

Branch Instructions : The branch instructions change the program sequence either unconditionally or subject to the specified condition.

Machine Code instructions : The instructions are written to terminate or stop the program. The programmer can either stop the program temporarily or permanently. The two machine code instructions used are HLT or NOP.

HLT Stop processing and wait.

NOP Do not perform any operation.

9.3.1 8085 Programs :

(a) To perform addition of two 8 bit numbers using 8085.

PROGRAM :

```
MVI        C, 00  ;Initialize C register to 00
LDA        4150   ;Load the value to Accumulator.
MOV        B, A   ;Move the content of Accumulator to B register.
LDA        4151   ;Load the value to Accumulator.
ADD        B      ;Add the value of register B to A
JNC        LOOP   ;Jump on no carry.
INR        C      ;Increment value of register C
LOOP: STA  4152   ;Store the value of Accumulator (SUM).
MOV        A, C   ;Move content of register C to Acc.
STA        4153   ;Store the value of Accumulator (CARRY)
HLT               ;Halt the program.
```

SAMPLE INPUT AND OUTPUT :

```
Input :   80 (4150)
          80 (4251)
Output :  00 (4152)
          01 (4153)
```

(b) To perform the subtraction of two 8 bit numbers using 8085.

PROGRAM :

```
MVI        C, 00  ;Initialize C to 00
LDA        4150   ;Load the value to Acc.
MOV        B, A   ;Move the content of Acc to B register.
LDA        4151   ;Load the value to Acc.
SUB        B
JNC        LOOP   ;Jump on no carry.
CMA               ;Complement Accumulator contents.
INR        A      ;Increment value in Accumulator.
INR        C      ;Increment value in register C
LOOP: STA  4152   ;Store the value of A-reg to memory address.
MOV        A, C   ;Move contents of register C to Accumulator.
STA        4153   ;Store the value of Accumulator memory address.
HLT               ;Terminate the program.
```

SAMPLE INPUT AND OUTPUT :

```
Input :   06 (4150)
          02 (4251)
Output :  04 (4152)
          01 (4153)
```

(c) To perform the multiplication of two 8 bit numbers using 8085

PROGRAM :

```
        MVI   D,00     ;Initialize register D to 00
        MVI   A, 00    ;Initialize Accumulator content to 00
        LXI   H, 4150
        MOV   B, M     ;Get the first number in B - reg
        INX   H
        MOV   C, M     ;Get the second number in C- reg.
LOOP:   ADD   B        ;Add content of A - reg to register B.
        JNC   NEXT     ;Jump on no carry to NEXT.
        INR   D        ;Increment content of register D
NEXT:   DCR   C        ;Decrement content of register C.
        JNZ   LOOP     ;Jump on no zero to address
        STA   4152     ;Store the result in Memory
        MOV   A, D
        STA   4153     ;Store the MSB of result in Memory
        HLT            ;Terminate the program.
```

SAMPLE INPUT AND OUTPUT :

```
            Input :    FF (4150)
                       FF (4151)
            Output :   01 (4152)
                       FE (4153)
```

(d) To perform the division of two 8 bit numbers using 8085

PROGRAM :

```
        LXI   H, 4150
        MOV   B, M     ;Get the dividend in B - reg.
        MVI   C, 00    ;Clear C - reg for quotient
        INX   H
        MOV   A, M     ;Get the divisor in A - reg.
NEXT:   CMP   B        ;Compare A - reg with register B.
        JC    LOOP     ;Jump on carry to LOOP
        SUB   B        ;Subtract A - reg from B- reg.
        INR   C        ;Increment content of register C.
        JMP   NEXT     ;Jump to NEXT
LOOP:   STA   4152     ;Store the remainder in Memory
        MOV   A, C
        STA   4153     ;Store the quotient in memory
        HLT            ;Terminate the program.
```

SAMPLE INPUT AND OUTPUT :

```
            Input :    FF (4150)
                       FF (4251)
            Output :   01 (4152) Remainder
                       FE (4153)Quotient
```

❑ **Check Your Progress – 2 :**

1. In 8085 instruction set, what is the notation for register source ?

   a. R b. M c. Rs d. Rd

2. Which instruction is used copy contents of source register to destination register ?

   a. MOV Rd, Rs b. MVI Rd, 8 bit

   c. OUT 8 bit d. IN 8 bit

9.4 16 Bit Microprocessors :

The 16–bit microprocessors are premeditated first and foremost to contend with minicomputers as well as leaning towards high level languages. Their applications from time to time go beyond those of 8–bit microprocessor furthermore might fight among those of mainframe computers. They have influential instruction set as well as proficiency of addressing megabytes of memory. Characteristically, some of the famous 16–bit microprocessors are Intel 8086/8088, Zilog Z8001/8002, LSI–11, Texas Instruments, TMS 9900, Motorola 68000 as well as National Semiconductors NS16000.

One critical feature in 16 Bit processor design is quantity of pins. It shows single tendency of having 40 pin package size moreover it takes benefit of accessible production as well as testing facilities. Intel preferred to design 8086/8088 using 40 pin configurations. However the ICs having more number of pins were selected due to advancement of technology.

The 16–bit microprocessors have been designed with following objectives

- Increase memory addressing capacity.
- Increase execution speed.
- Provide powerful instruction set.
- Facilitate programming in high level languages.

❖ **Intel 8086/8088 :**

8086/8088 is 16 bit/40 pin microprocessor which is invented by Intel. It is competent of addressing 1 MB of memory. Other types of chip configuration work from 4 MHz to 8 MHz clock frequencies. Figure 9.1 (a) demonstrate internal registers where the coloured part is similar to 8085 registers. It carries 16 bit registers, where upper most four registers (AX, BX, CX and DX) are applied for general purpose accumulators which can be used as 8–bit registers.

Now the next four 16 bit registers are applied mainly to memory pointers as well as index registers, which is also general purpose registers. After this, the next four 16–bit registers are mainly for identifying segment of 1 MB memory. The last two registers are comparable to program counter as well as flag register.

The 8088 works similarly as 8086, but the only thing is that, it carries 8–bit data bus. Its internal architecture as well as instruction set is similar to 8086. The only difference between 8086 and 8088 is that it carries 16–bit data word that is transferred in two segments in case of 8088. Finally it is seen that 8088 can be seen as 8–bit processor having a power execution of 16 bit processor.

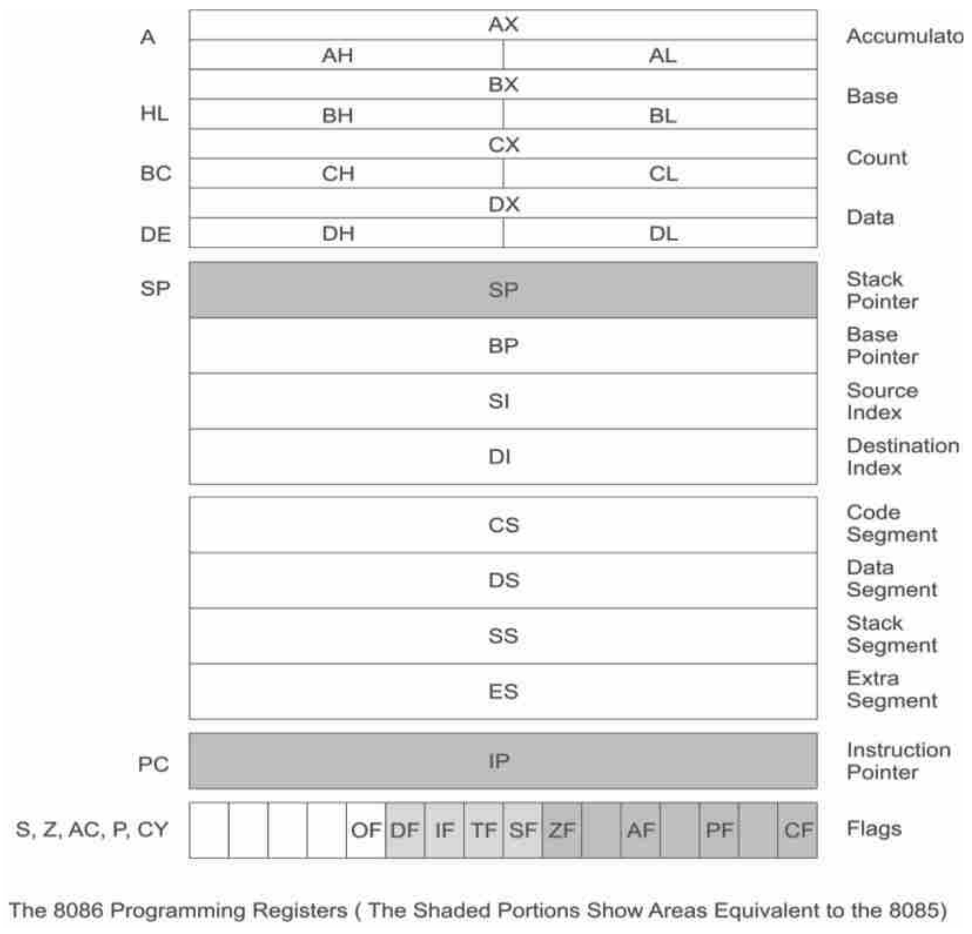


Fig. 9.1 8086 Programming registers

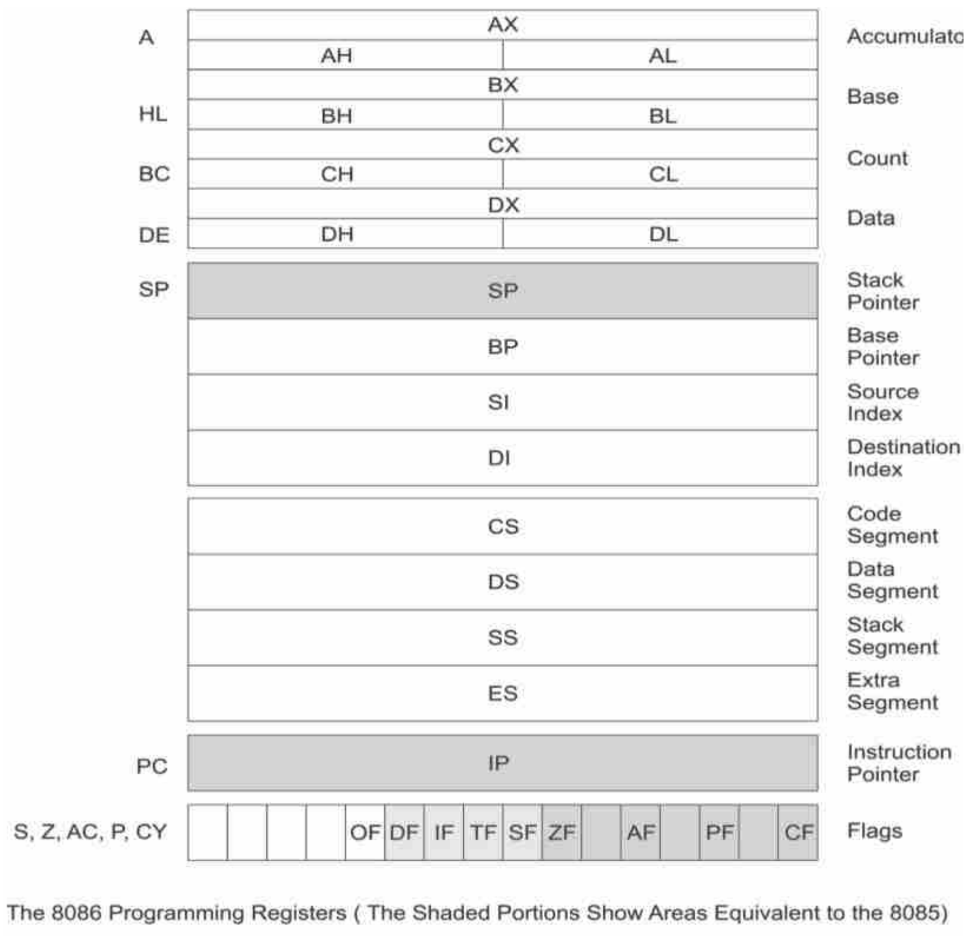


Fig. 9.2 8086 Programming Register

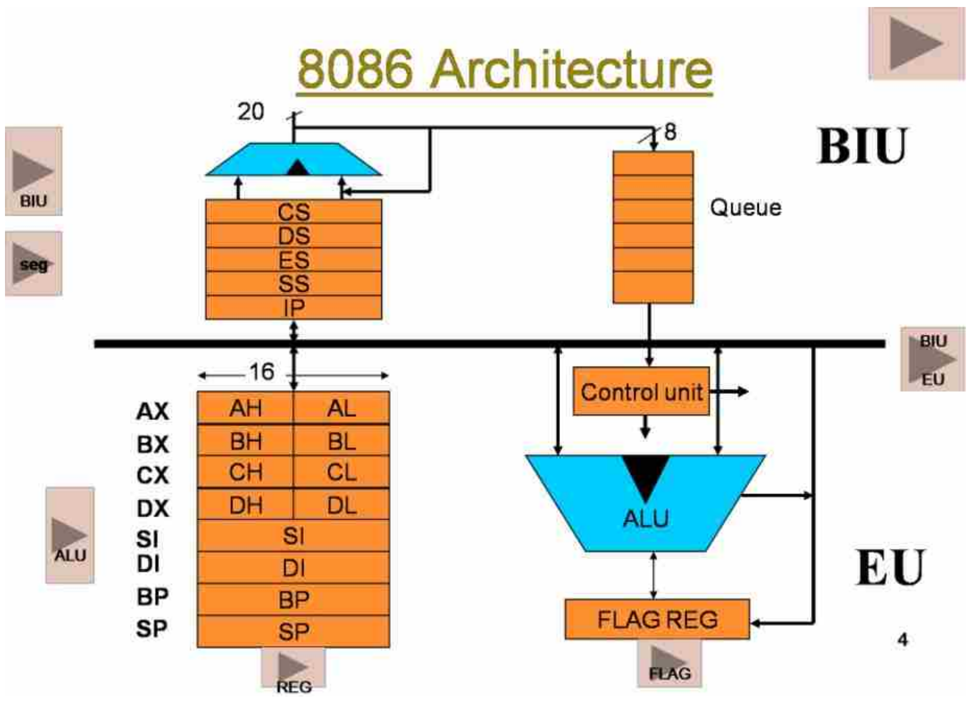


Fig. 9.3 Architecture 8086

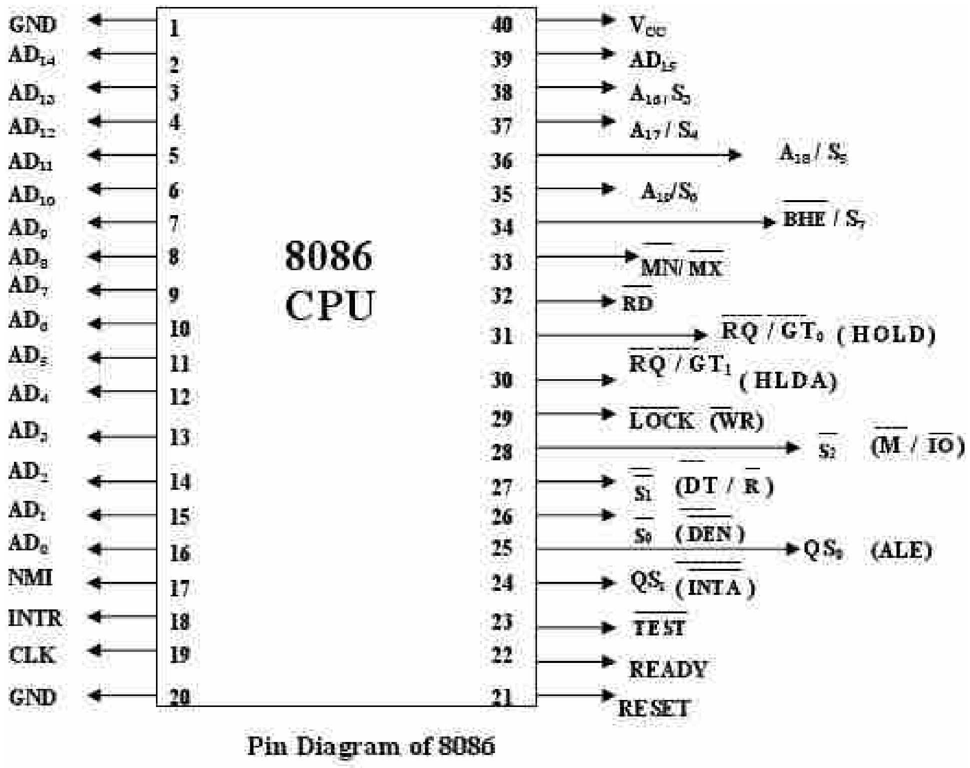


Fig. 9.4 Pin diagram of 8086

Pin Name	Description	Type
Ad0–AD15	Data/Address Bus	Bidirectional, tristate
A16/S3, A17/S4	Address/Segment Identrifier	Output, tristate
A18/55	Address/Interrupt enable status	Output, tristate
A19/S6	Address/Status	Output, tristate
BHE/S7	High order byte status	Output, tristate
RD	Read Control	Output, tristate
Ready	Wait State Request	Input
Test	Wait for Test Control	Input
INTR	Interrupt Request	Input
NMI	Non–Maskable Interrupt Request	Input
RESET	System Request	Input
CLK	System Clock	Input
MN/MX	=GND for maximum system	
S0,S1,S2	Machine Cycle Status	Output tristate
RQ,GT0,RQ,GT1	Local bus Priority control	Bidirectional
QS0, QS1	Instruction queue status	Output
LOCK	Bus hold control	Output tristate
MN/MX	Vcc for a min System	
M/IO	Memory or I/O access	Output tristate
WR	Write Control	Output tristate
ALE	Address Latch Enable	Output
DT/R	Data Transmit/receive	Output tristate
DEN	Data Enable	Output tristate
INT A	Interrupt Acknowledge	Output tristate
HOLD	Hold Request	Input
HLD A	Hold Acknowledgement	Output
Vcc, GND	Power, Ground	

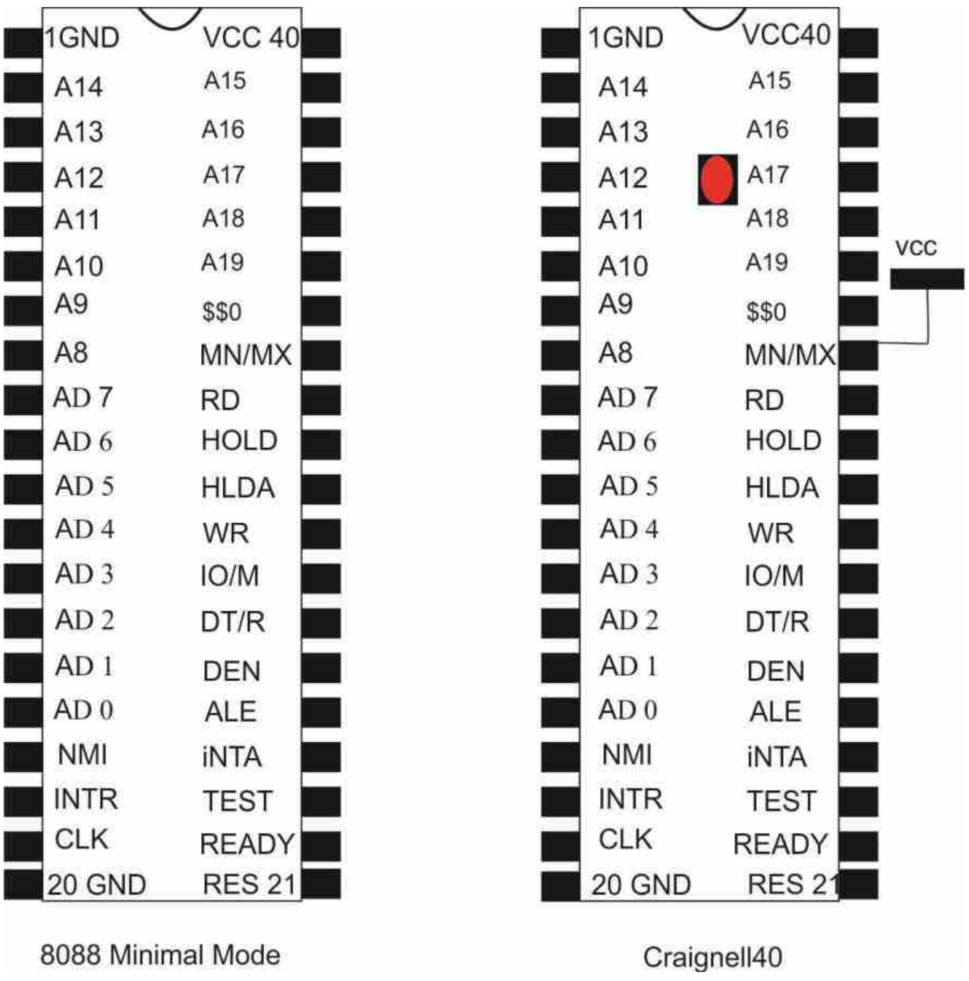


Fig. 9.5 8088 Pin out

❑ **Check Your Progress – 3 :**

1. Which among the following are 16–bit microprocessors ?

   a. Intel 8086 b. Zilog Z8002 c. TMS 9900 d. all

2. 8086/8088 is ____ bit microprocessor.

   a. 4 b. 8 c. 16 d. 32

9.5 Let Us Sum Up :

In this unit, we have learnt that, there are various formats of specifying the sources and destinations are called addressing modes. The 8085 instruction set comprises of the following addressing modes. These are also called addressing schemes.

- Immediate addressing : MVI Rs, data
- Register addressing : MOV Rd, Rs
- Direct Addressing : IN/OUT Port #
- Indirect addressing : LDA, LDAX, STA, STAX

The 8085 instruction set provides 74 operation codes that result in 246 instructions. Some important instructions are furnished below. The following notations are used in the description of instructions :

R – 8 bit register

M – Memory register (Location)

Rs – Register source

Rd – register destination

Rp – Register pair

Arithmetic Instructions : There are four basic arithmetic instructions in 8085 instruction set these are Add, Subtract, Increment (Add 1) and Decrement (Subtract 1).

Logical Instructions : The logical instructions are based on the operations of logic gates such as AND, OR, NOT. The 8085 can process 5 types of logical instructions viz. AND, OR, XOR Compare and Rotate.

Branch Instructions : The branch instructions change the program sequence either unconditionally or subject to the specified condition.

Machine Code instructions : The instructions are written to terminate or stop the program. The programmer can either stop the program temporarily or permanently. The 16–bit microprocessors are designed primarily to compete with minicomputers and are oriented towards high level languages. Their applications sometimes overlap those of 8–bit microprocessor and may compete with those of mainframe computers.

The 16–bit microprocessors have been designed with following objectives :

- Increase memory addressing capacity.
- Increase execution speed.
- Provide powerful instruction set.
- Facilitate programming in high level languages.

9.6 Answers for Check Your Progress :

❑ **Check Your Progress 1 :**

1. (d)

❑ **Check Your Progress 2 :**

1. (c), 2. (a)

❑ **Check Your Progress 3 :**

1. (d), 2. (c)

9.7 Glossary :

1. **Addressing mode –** It is a type of format that will specify source and destination.
2. **MOV R, M –** Copy the data byte from memory location (source) to register
3. **LDAX Rp –** Copy the data byte into the accumulator from memory location indicated by the register pair.
4. **LDA 16 bit –** Copy the data byte into the accumulator from the memory location specified by the 16 bit address.

9.8 Assignment :

Discuss the various logical and branch instructions of 8085.

9.9 Activities :

Obtain information about the various types of addressing modes and discuss it in your own words.

9.10 Case Study :

Collect information about microprocessors and discuss the same in your own words.

9.11 Further Readings :

1. 8080/8085 Microprocessor Book, Intel Marketing Communications, Wiley–Intel Series.
2. 8085 Microprocessors : Programming and Interfacing, N. K. Srinath, PHI Learning Private Ltd.
3. 8085 Software Design, C. A. Titus, SAMS Publishing.
4. Schaum's Outline Series of theory and Problems on Microprocessor Fundamentals, R. L. Tokheim, Mc–Graw Hill Publishing.
5. Microprocessor Architecture, Programming and Applications with the 8085 / 8080 A., Ramesh Gaonkar, Wily Eastern Limited.

Unit 10

MULTIPROCESSOR CONFIGURATION & MICROCONTROLLERS

UNIT STRUCTURE

10.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The microcontroller and its various types
- Areas microcontrollers are used.
- When to use microcontroller and when to use microprocessor ?

10.1 Introduction :

Multiprocessor means a numerous set of processors that implements instructions concurrently. There are three elementary multiprocessor formations.

10.2 Coprocessor Configuration :

A Coprocessor is a particularly designed circuit on microprocessor chip which can achieve the same assignment very quickly, which the microprocessor executes. It decreases the work load of the core processor. The coprocessor allocates the similar memory, IO system, bus, control logic and clock generator. The coprocessor manages specialized jobs like mathematical calculations, graphical display on screen, etc.

The 8086 and 8088 can execute most of the tasks but their instruction set is not able to accomplish difficult mathematical operations, so in these circumstances the microprocessor requires the math coprocessor like Intel 8087 math coprocessor, which can effortlessly execute these operations very rapidly.

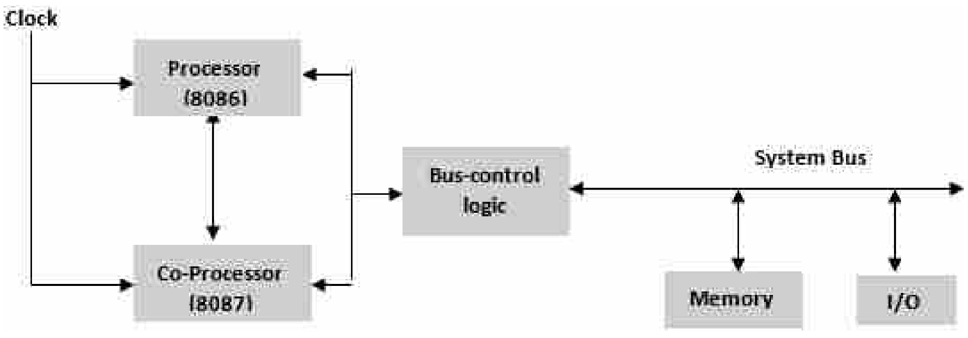


Fig. 10.1 Block Diagram of Coprocessor Configuration

How is the coprocessor & the processor linked ?

- The coprocessor and the processor is linked via TEST, RQ–/GT– and QS0 & QS1 signals.
- The TEST signal is linked to BUSY pin of coprocessor and the lasting 3 pins are linked to the coprocessor's 3 pins of the identical label.
- TEST signal takes care of the coprocessor's movement, i.e. the coprocessor is busy or idle.
- The RT–/GT–is used for bus arbitration.

10.2.1 Closely Coupled Configuration :

Closely coupled configuration is like the coprocessor configuration, i.e. both share the similar memory, I/O system bus, control logic, and control generator with the host processor. However, the coprocessor and the host processor gets and accomplishes their own instructions. The system bus is managed by the coprocessor and the host processor independently.

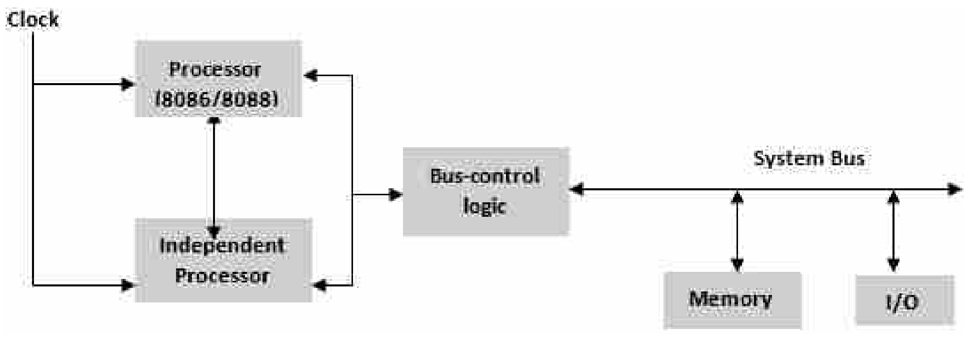


Fig. 10.2 Block Diagram of Closely Coupled Configuration

How is the processor & the independent processor linked ?

- Interaction between the host and the independent processor is done through memory space.
- None of the instructions are used for communication, like WAIT, ESC, etc.
- The host processor accomplishes the memory and wakes up the independent processor by sending instructions to one of its ports.
- Afterwards the independent processor acquires the memory to run the job.

- After finishing the job, it sends a response to the host processor by using the status signal or an interrupt request.

10.2.2 Loosely Coupled Configuration :

Loosely coupled configuration comprises of the number of modules of the microprocessor based systems, which are linked through a mutual system bus. Each module comprises of their own clock generator, memory, I/O devices and are linked through a local bus.

❖ **Merits :**

- Acquiring more than one processor results in improved efficiency.
- Every processors have their own local bus to access the local memory/ I/O devices. This makes it simple to achieve parallel processing.
- The system structure is flexible, i.e. the collapse of one module doesn't affect the whole system; damaged module can be swapped later.

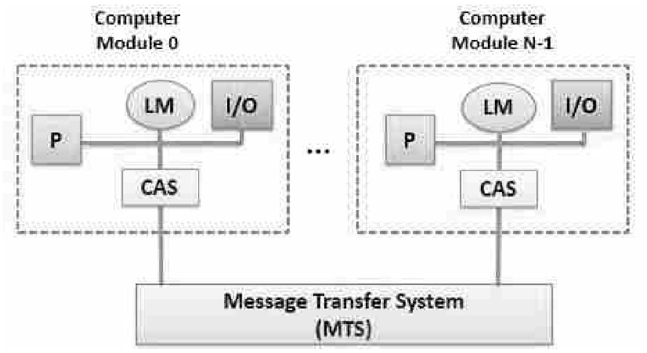


Fig. 10.3 Block Diagram of Loosely Coupled Configuration

10.3 Microcontroller :

A microcontroller is a compact and cost–effective microcomputer, which is designed to carry out the particular jobs of embedded systems like displaying microwave's information, receiving remote signals, etc.

The general microcontroller contains the processor, the memory (RAM, ROM, EPROM), Serial ports, peripherals (timers, counters), etc.

A micro-controller is a single IC, normally with the subsequent functionalities :

- CPU(central processing unit) - ranging from tiny and simple 4-bit processors to complex 32-bit or 64-bit processors
- RAM (volatile memory) for data storage
- ROM, EPROM, EEPROM or Flash memory for program and operating factor storage
- Distinct input and output bits, allowing manage or discovery of the logic state of an individual enclosed pin
- Serial input/output such as serial ports i.e. Universal Asynchronous Receiver Transmitter (UARTs)
- Additional serial communications interfaces like I^{2}C, Serial Peripheral Interface and Controller Area Network for system interconnection
- Peripherals such as timers, event counters, PWM (Pulse width modulation)generators, and watchdog

- Clock generator - often an oscillator for a quartz timing crystal, resonator or RC circuit
- Many include analog-to-digital converters, some include digital-to-analog converters
- In-circuit programming and in-circuit debugging shore up

Interrupt Latency :

Microcontrollers used in embedded systems often look for to optimize interrupt latency over instruction throughput.

When an electronic device causes an interrupt, through the framework switch the transitional consequences data have to be saved before the software responsible for handling the interrupt can execute. They must also be restored later than that interrupt handler is finished. If there are more processor registers, this cutback and restoring process may take additional time, increasing the latency.

Ways to reduce such restore latency comprise having comparatively few registers in their central processing units (unwanted since it slows down the majority non-interrupt processing considerably), or at least having the hardware not save them all (this fails if the software then needs to reimburse by saving the rest "manually"). Another technique involves spending silicon gates on "shadow registers": One or more photocopy registers used only by the interrupt software, perhaps behind a devoted stack.

Other factors affecting interrupt latency include :

Cycles needed to finish existing CPU activities. To diminish those costs, microcontrollers tend to have short pipelines, small write buffers, and ensure that longer instructions can be continued or restarted. RISC design principles ensure that most instructions take the similar numeral of cycles, serving keep away from the need for most such continuance/resume logic.

The span of any significant segment that needs to be interrupted. Entry to a significant segment restricts simultaneous data structure entrance. When a data structure must be accessed by an interrupt handler, the significant section must lump that interrupt. Accordingly, interrupt latency is bigger by however long that interrupt is barred. When there are hard peripheral constraints on system latency, developers often need apparatus to calculate interrupt latencies and note down which significant sections causes delays.

o One common technique just blocks all interrupts for the period of the significant section. This is easy to put into practice, but sometimes significant sections get awkwardly extended.

o A more multifaceted method just blocks the interrupts that may activate access to that data structure. This is often based on interrupt priority, which tend to not correspond well to the applicable system data structures. Accordingly, this technique is used mostly in very unnatural environments.

o Processors may have hardware support for some significant sections. Examples include at the bottom of atomic access to bits or bytes inside a word, or other atomic access instructions like the LDREX (Load Register Exclusive) /STREX (Store Register Exclusive) are introduced in the ARMv6 architecture.

- o Interrupt nesting in certain microcontrollers permit advanced priority interrupts to interrupt lower priority ones. This allows software to administer latency by openhanded time-critical interrupts higher priority (and thus lower and more predictable latency) than less-critical ones.
- o Trigger rate is taken in to consideration while interrupts occur back-to-back, microcontrollers may keep away from an additional context to keep/ reinstate cycle by a appearance of tail call optimization.

Memory Technology :

Two special kinds of memory are frequently used with microcontrollers, a non-volatile memory for storing firmware and a read-write memory for provisional data.

Data :

From the initial microcontrollers to today, six-transistor SRAM is roughly used as the read/write operational memory, with a few additional transistors per bit used in the register file.

In accumulation to the SRAM, some microcontrollers also have internal EEPROM for data storage; and even ones that do not have any are often connected to outside sequential EEPROM chip or external serial flash memory chip. A small number of microcontrollers commencement in 2003 had "self-programmable" flash memory.

Firmware :

Microcontroller firmware is entrenched software which is written into the program memory of an MCU. Since microcontroller interacts with the end user, in order to experiment its firmware, one needs to emulate input signals and calculate or confine its output signals. If these output signals match to what you have stated in a requirement, then the firmware is functioning as it should.

There are a lot of devices that help a developer do this testing: voltmeters, oscilloscopes, signal generators, logical analyzers, protocol analyzers, interface converters etc. So the basic idea of firmware testing is to implement an input signal and observe the reaction. If you see a trouble, you update the main code and then replicate this cycle.

The initial microcontrollers used masking in ROM to save the firmware. Afterward microcontrollers such as the Free scale 68HC11 and PIC microcontrollers had EPROM memory, which used a transparent casement to permit removal using UV light, while manufacture versions had no such casement, being one-time-programmable. Firmware updates were comparable to replacing the microcontroller itself, thus many products were not upgradeable.

List of common micro-controllers :

(1) INTEL

a. 8 BIT

i. MCS-48, 8048 family

ii. MCS-51, 8051 family

iii. MCS-515, High performance 8051 instruction set

b. 16 BIT
   i. MCS-96, 8096 family
   ii. Intel MCS-296

10.4 Types of Microcontroller :

Microcontrollers are divided into different categories depending on memory, architecture, bits and instruction sets. Following are the list of their types :

10.4.1 Bit Based Configuration :

Depending on bit configuration, the microcontroller is further divided into three categories.

8–bit microcontroller : This form of microcontroller is used to perform arithmetic and logical operations like addition, subtraction, multiplication division, etc. For example, Intel 8031 and 8051 are 8 bits microcontroller.

16–bit microcontroller : This form of microcontroller is used to perform arithmetic and logical operations where more accuracy and performance is required. For example, Intel 8096 is a 16–bit microcontroller.

32–bit microcontroller : This form of microcontroller is generally used in automatically controlled appliances like automatic operational machines, medical appliances, etc.

10.4.2 Memory Based Configuration :

Depending on the memory configuration, the microcontroller is further divided into two categories.

External memory microcontroller : This form of microcontroller is designed in such a way that they don't have a program memory on the chip. So, it is labled as external memory microcontroller. For example : Intel 8031 microcontroller.

Embedded memory microcontroller : This form of microcontroller is designed in such a way that the microcontroller has all programs and data memory, counters and timers, interrupts, I/O ports are embedded on the chip. For example : Intel 8051 microcontroller.

10.4.3 Instruction Based Configuration :

Depending on the instruction set configuration, the microcontroller is further divided into two categories.

CISC : CISC stands for complex instruction set computer. It permits the user to insert only one instruction as an alternative to many simple instructions.

RISC : RISC stands for Reduced Instruction Set Computers. It decreases the operational time by shortening the clock cycle per instruction.

10.5 Applications of Microcontroller :

Microcontrollers are extensively used in various devices such as :

- Light sensing and controlling devices like LED.
- Temperature sensing and controlling devices like microwave oven, chimneys.

- Fire detection and safety devices like Fire alarm.
- Measuring devices like Volt Meter.

10.6 Difference between Microprocessor and Micro-controller :

The following table highlights the differences between a microprocessor and a microcontroller :

Microcontroller	Microprocessor
Microcontrollers are used to perform a single job within an application.	Microprocessors are used for big applications.
It is used in designing and hardware is cost-effective.	It is used in designing and hardware is not cost-effective.
Easy to replace.	Not so easy to replace.
It is built with CMOS technology, which needs less power to operate.	Its power consumption is high because it has to manage the whole system.
It contains CPU, RAM, ROM, I/O ports.	It does not contain RAM, ROM, I/O ports. It uses its pins to interface to peripheral devices.

❑ **Check Your Progress :**

1. A ______ is a compact and cost–effective microcomputer.

   a. microcontroller b. microprocessor

   c. Both a and b d. None of these

2. CISC stands for ______.

   a. Complex Instruction Set Computing

   b. Common Instruction Set Computing

   c. Command Instruction Set Computing

   d. None of these

3. Interaction between the host and the independent processor is done through ______ space.

   a. microcontroller b. memory

   c. cache d. None of these

4. A ______ is a specially designed circuit on microprocessor chip which can perform the same task very quickly

   a. Config b. Coprocessor

   c. Both a and b d. None of these

5. In ______ power consumption is high.

   a. Microcontroller b. Microprocessor

   c. Microcomputer d. None of these

10.7 Let Sum It Up :

- In this unit, we have learnt how multiprocessors can execute instructions simultaneously with the help of coprocessor configurations.

- Closely coupled configuration is similar to the coprocessor configuration, i.e. both share the same memory, I/O system bus, control logic, and control generator with the host processor.
- Loosely coupled configuration consists of the number of modules of the microprocessor based systems, which are connected through a common system bus.
- The general microcontroller consists of the processor, the memory (RAM, ROM, EPROM), Serial ports, peripherals (timers, counters), etc.
- Microcontrollers can be based on bit, memory and instruction set.
- There are various applications of microcontroller :
    - o Consumer electronics products
    - o Instrumentation process and control
    - o Office Equipment
    - o Multimedia Applications
    - o Automobile
    - o Communication
- Microcontroller are built with CMOS technology.
- Microprocessor are used in :
    - o Handheld devices
    - o General purpose computing
    - o High performance computing
- Embedded firmware is the flash memory chip that stores dedicated software executing in a IC in an embedded device to manage its functions. Firmware in embedded systems fills the similar reason as a ROM other than it can be updated more with no trouble for improved flexibility to circumstances or interconnecting with supplementary equipment.

10.8 Answers to Check Your Progress :

❑ **Check Your Progress :**

1. (a), 2. (a), 3. (b), 4. (b), 5. (a)

10.9 Glossary :

- **RQ–/GT** – Request and Grant
- **QS0 & QS1 –** It provides status to allow external tracking of the internal 8086 instruction queue.
- **CAS –** Channel and arbiter switch that allocate access to shared resources.

10.10 Assignment :

Discuss various types of Microcontroller and its usage in various areas of technology.

10.11 Activities :

Obtain the information on various types of microcontrollers and state its applications in solving the problems of real life scenarios.

10.12 Case Study :

Collect the information of energy efficient microcontrollers and state its use in implementing internet of things.

10.13 Further Readings :

1. Microprocessor Architecture, Programming and Applications with the 8085 6/e, By Ramesh Gaokar
2. https://en.wikipedia.org/wiki/Microcontroller
3. 8080/8085 Microprocessor Book, Intel Marketing Communications, Wiley–Intel Series.
4. 8085 Microprocessors : Programming and Interfacing, N. K. Srinath, PHI Learning Private Ltd.
5. 8085 Software Design, C. A. Titus, SAMS Publishing.

BLOCK SUMMARY :

The block will aware the students about 8085 Interrupts and 8259 programmable Interrupt controller. The generalization and understanding of concepts on microprocessors will explains about interfacing techniques of 8251 programmable communication interface.

The block will give detailed knowledge about addressing modes and use of 8085 instruction set. The concept of 8085 instruction set along with understanding of 16 bit microprocessor is also well explained. The students will be trained with practical examples and exercises that will help to learn and grab the subject easily.

Additionally this block also gives an introduction of microcontrollers and coprocessor configuration and discusses its different categories and uses in our day to life.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Explain addressing modes ?
2. Explain TRAP interrupts ?
3. Explain the interrupt registers of 8259 ?
4. What are immediate and register addressing modes ?
5. Explain the significance of DMA ?
6. Discuss the applications of microcontroller.
7. Differentiate between microprocessor ad microcontroller.

❖ **Long Questions :**

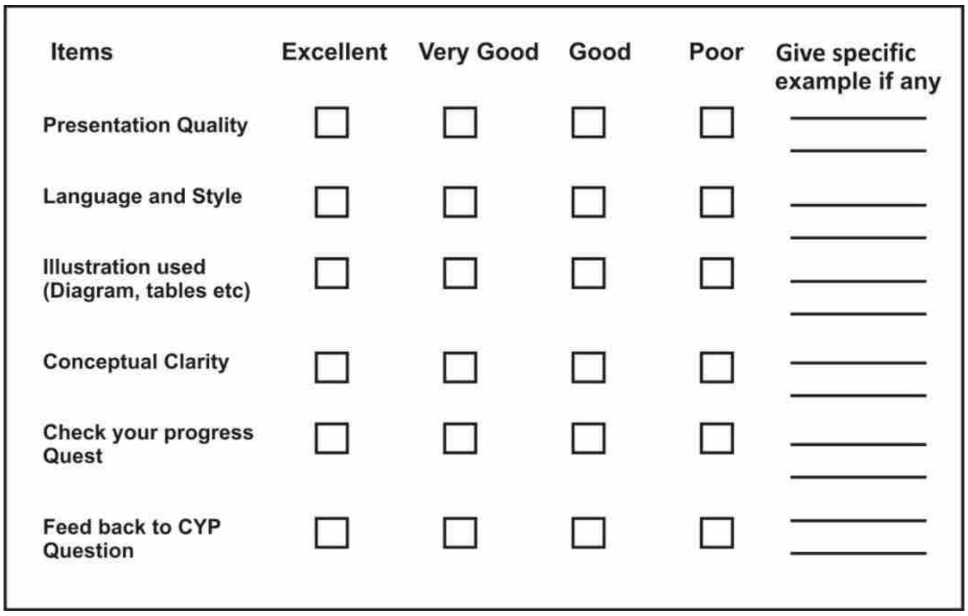
1. Explain the notations used in the description of instructions of 8085 instruction set ?
2. Explain the transmitter and receiver sections of 8251 ?
3. Explain the arithmetic instructions in 8085 instruction set ?
4. Explain different types of microcontroller in detail.
5. Differentiate between closely coupled and loosely coupled configuration.

❖ **Enrolment No. :** ☐

1. How many hours did you need for studying the units ?

Unit No.	8	9	10
No. of Hrs.			

2. Please give your reactions to the following items based on your reading of the block :



Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	________
Language and Style	☐	☐	☐	☐	________
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	________
Conceptual Clarity	☐	☐	☐	☐	________
Check your progress Quest	☐	☐	☐	☐	________
Feed back to CYP Question	☐	☐	☐	☐	________

3. Any other Comments

....

....

....

....

....

....

....

....

Dr. Babasaheb Ambedkar Open University Ahmedabad **BCAR-303**

System Programming and Introduction to Microprocessor

BLOCK 4 : MEMORY

MEMORY

Block Introduction :

Memory is an important part of microcomputer system as it stores binary instructions and data received from microprocessor. The storage systems can be classified as primary storage and secondary storage. The primary storage mainly comprises of semiconductor memory devices such as Read Only Memory (ROM), Random Access Memory (RAM) whereas secondary storage systems mainly comprises of magnetic memory and optical memory. This unit will explain all concepts of memory.

In this block students will be aware of about memory storage devices with reference to primary and secondary storage. The students will find the block effective as it explains about different storage media used by user to keep their data.

The block will detailed about different microprocessors invented by Intel. The students will get detail knowledge about 80286, 80386, 80486 and Intel Pentium microprocessors. The students will be trained with practical examples and exercises that will help to learn and grab the subject easily.

Block Objectives :

After learning this block, you will be able to understand :

- About Primary Storage
- Basic of Secondary Storage
- Overview of Higher Versions
- About Intel 80286, Intel 80386, Intel 80486, Intel Pentium

Block Structure :

Unit 11

MEMORY – I

UNIT STRUCTURE

11.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The basic concepts of primary and secondary storage
- The basic concepts of address and data
- How the data is recorded in ROM
- Various types of ROMs viz. PROM, EPROM and EEPROM
- The working of RAM & interfacing with microprocessors
- The working of static RAM and Dynamic RAM
- The difference between RAM and ROM

11.1 Introduction :

Memory is the essential component of microcomputer system; it stores binary instructions and data for the microprocessor.

The storage systems can be classified as primary storage and secondary storage. The primary storage mainly comprises of semiconductor memory devices such as Read Only Memory (ROM), Random Access Memory (RAM) whereas secondary storage systems mainly comprises of magnetic memory and optical memory. This unit will explain all concepts of memory.

11.2 Primary Storage :

The microprocessor is dealt with two types of memories : Read Only Memory (ROM) and Read/Write Memory (R/W M).

❖ **Read Only Memory (ROM) :**

The name is given as Read Only Memory (ROM) because data stored in ROM cannot be modified. Unlike RAM, the 'write' feature is not available in ROM. The ROM is mainly used to distribute firmware. (Firmware is software that is very closely tied to specific hardware). The desired data is permanently stored in ROM and thus can never be modified.

❖ **Diode ROM :**

The data is stored in terms of bits in various cells in ROM. The word comprises of number of cells thus forming a register. The semiconductor devices such as diodes, transistors or MOSFETs are used to store a bit.

The given figure shows a Diode ROM having diodes provided in cells.

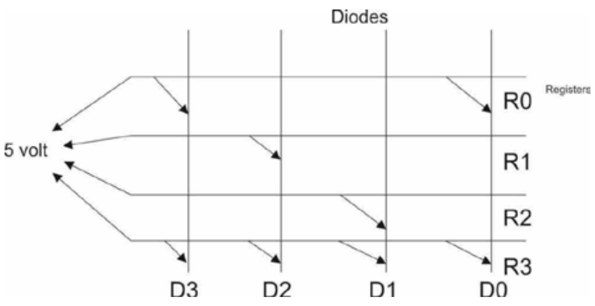


Fig. 11.1 Diode ROM

As shown in the Fig 11.1, The 4 bit word is stored in 4 bit registers R0, R1, R2 and R3. The bits D0, D1, D2 and D3 are stored in registers R0, R1, R2 and R3 by using diodes. Hence the circuit is called diode ROM. Let us suppose that the switches are named as 0, 1, 2 and 3 for registers R0, R1, R2 and R3 respectively.

Then the data would be stored as follows :

Register	**Address**	**Word**			
		D3	D2	D1	D0
R0	0	1	0	0	1
R1	1	0	1	0	0
R2	2	0	0	1	0
R3	3	1	1	1	1

When the switch for register R0 is connected to 5 V battery, the contents of R0 are displayed at output terminals D0, D1, D2 and D3. When the switch of R1 is connected to the 5 V battery, the contents of R1 are displayed and so on.

In this way, the data stored in ROM is displayed by switch changeover.

Now instead of switch, we can use decoder. If 2 bit decoder is provided, it would have 2^4 i.e. 4 outputs. These four outputs will be available by giving inputs 00, 01, 10 and 11. See the table 11.1.

Table 11.1 Decoder

Decoder		**Register**	**Address**	**Word**			
A1	A0			D3	D2	D1	D0
0	0	R0	0	1	0	0	1
0	1	R1	1	0	1	0	0
1	0	R2	2	0	0	1	0
1	1	R3	3	1	1	1	1

In this way, by giving input to decoder, corresponding data can be read. The Decoder – Diode ROM is shown in the Fig. 11.2

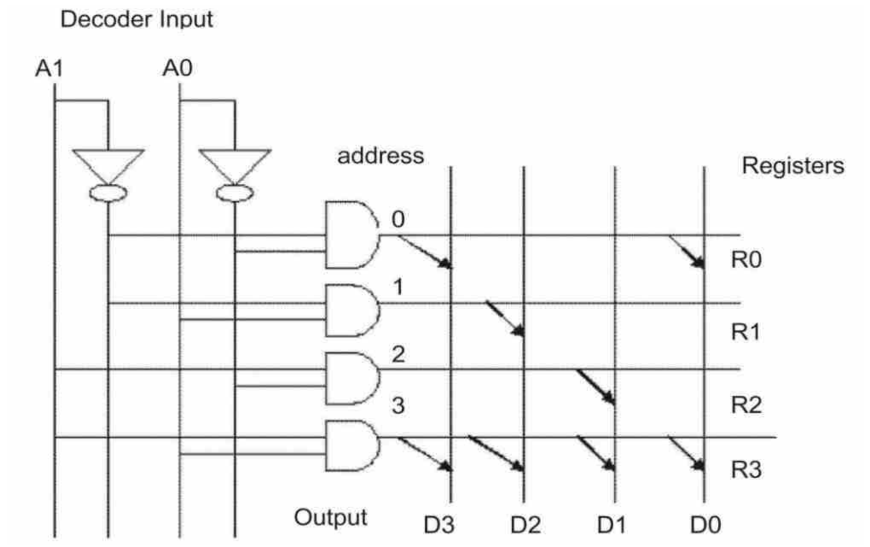


Fig. 11.2 Decoder–Diode ROM

The ROM chips are manufactured by the manufacturer as per your order. You have to provide details to manufacturer regarding data to be stored at various memory locations. Accordingly the manufacturer produces a mask i.e. a photographic template of circuit. The mask can be used to undertake mass production of ROM.

Other types of non–volatile solid–state memory are :

- **Programmable Read–Only Memory (PROM)**

In PROM, all diodes are supplied with fusible links which if not used gets burnt off by PROM Programmer. In this, excess current is put inside the cell which will damage the link by PROM programmer. The drawback with PROM is that it gets programmed only once as after burning of links, they cannot be joined further.

- **Erasable Programmable Read–Only Memory (EPROM)**

In EPROMs, the data gets erased when is brought under strong ultraviolet light. It can be written again by applying high voltage. If the EPROM is exposed to UV light more frequently than the memory chip gets damage. Such type of chip can be easily located by quartz window by way of UV light.

- **Electrically Erasable Programmable Eead–Only Memory (EEPROM) :**

EEPROM is same as EPROM which passes all its contents to be electrically erased, and then rewritten electrically due to the requirement of computer accessories as camera, MP3 player. Writing on EEPROM is very slow process as compared to reading from ROM or writing to RAM.

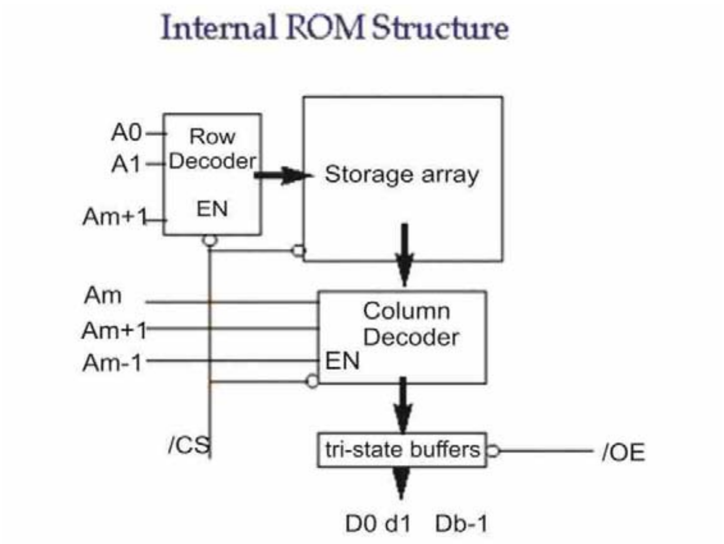


Fig. 11.3 Internal Structure of a Read–only Memory

The Figure 11.3 shows the arrangement of an internal structure of read–only memory where a memory size is of 16 words having 8 bits. In this, the address–decoder is a basic demultiplexer. Here with each binary address input, one of the word lines gets charged through address decoder. The arrangement of 4 :16 demultiplexer uses to control 16 vertical lines which is similar to 16 memory addresses.

It is seen that for each vertical lines, a connection is given to bit lines which gets charged for similar memory word. Here, the diodes made connections with word lines to bit lines.

❖ **Random Access Memory (RAM) :**

It is a computer memory which will be worked out randomly. In this any number of byte of memory worked without touching previous bytes. It is most famous memory device used in computers and printers.

It is called as read/write memory, as the data is read from and written into RAM. It is a volatile memory in which the data gets lost when power is switch off.

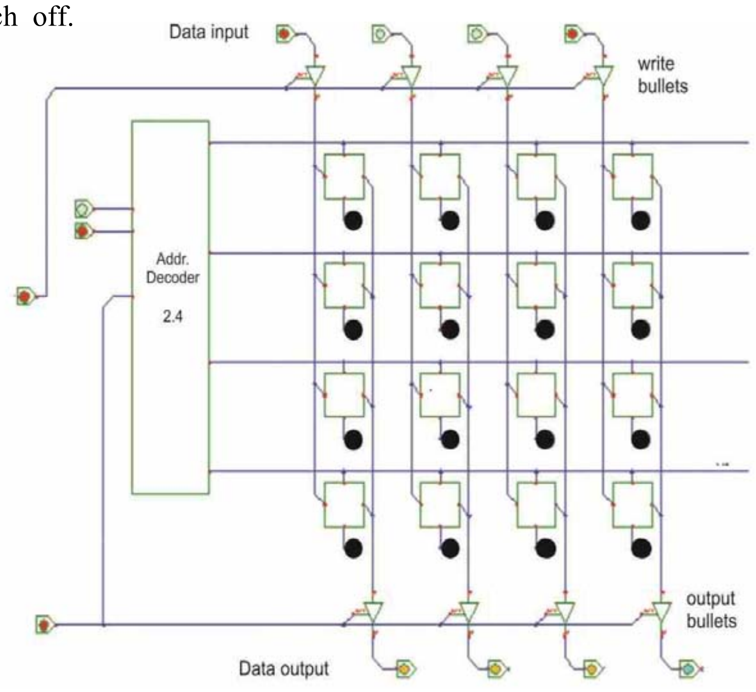


Fig. 11.4 General Arrangement of RAM

The figure 11.4 shows a general arrangement of RAM. The RAM cells are arranged in form of matrix elements. These cells are connected to the address decoder. Each cell has input and output terminal for data. The address for the given row of cells is inputted to the decoder as 00, 01, 10 and 11. According to the address, the corresponding data pertaining to the row of cells is selected.

There are two main types of RAM, static and dynamic. These are termed as SRAM (Static Random Access Memory) and DRAM (Dynamic Random Access Memory).

Static RAM uses flip–flops to store bits and so consumes current whether they are storing a 1 or a 0.

Dynamic RAM uses capacitors to store charges and use less power. However, these stored charges leak away and have to be continually REFRESHED which makes the circuitry more complicated.

- SRAM is faster as compared to DRAM
- DRAM refreshes thousands of times/second while SRAM needs no refreshing
- DRAM access times is 60 nanoseconds, while SRAM access time is 10 nanoseconds
- SRAM and DRAM are both volatile

It is found that RAM is synonymous having main memory. As seen, a computer with 8MB RAM carries 8 million bytes of memory that are utilized by programs.

Also, ROM is read–only memory which is a special memory which stores programs. Many computers contain ROM. Both RAM and ROM allows random access. Finally, it is examined that RAM is read/write memory, while ROM is read–only memory. Fig 11.9 shows 4118 RAM structure.

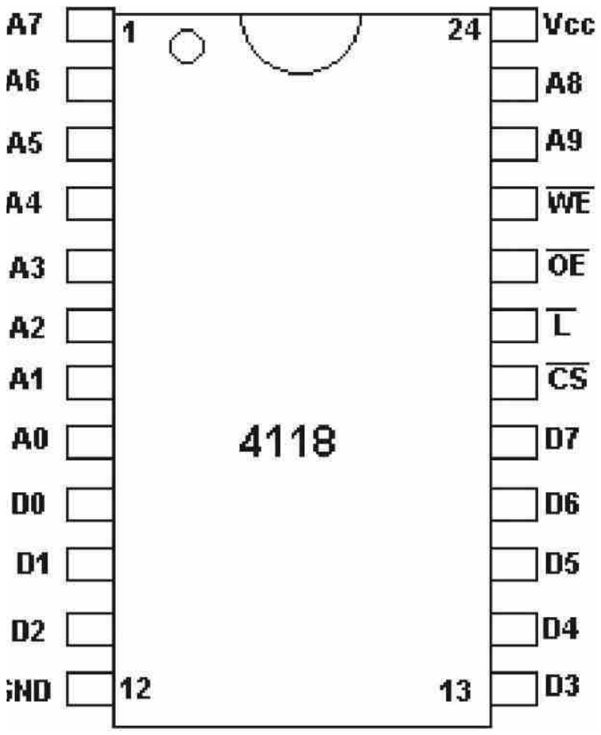


Fig. 11.5 4118 (8 bit x 1k static RAM)

D0–07	**DATA IN/OUT**
A0–09	**ADDRESS INPUTS**
Vss	**GROUND**
Vcc	**POWER +5 VOLTS**
$\overline{\text{WE}}$	**WRITE ENABLE**
$\overline{\text{OE}}$	**OUTPUT ENABLE**
$\overline{\text{L}}$	**LATCH**
$\overline{\text{CS}}$	**CHIP SELECT**

❖ **Cache Memory :**

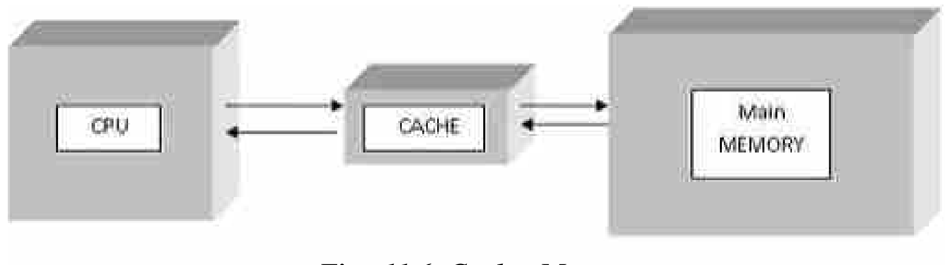


Fig. 11.6 Cache Memory

A cache memory is a memory used by the central processing unit to reduce the average time in order to access the memory. The cache is a smaller, faster memory which stores copies of the data from the most frequently used main memory locations.

When the processor needs to read from or write to a location in main memory, it first checks whether a copy of that data is in the cache. If so, the processor immediately reads from or writes to the cache, which is much faster than reading from or writing to main memory. See Fig. 11.6.

Most modern desktop and server CPUs have at least three independent caches : an instruction cache to speed up executable instruction fetch, a data cache to speed up data fetch and store and a translation look aside buffer used to speed up virtual–to–physical address translation for both executable instructions and data.

❖ **Read / Write Memory (R/W M) :**

Read /Write memory is Random Access Memory (RAM).

For an 8–bit microprocessor, the memory is required to store 8–bits of information as a group.

To communicate with memory, 8085 should

1. Select the chip
2. Identify the register
3. Read from or write into the register

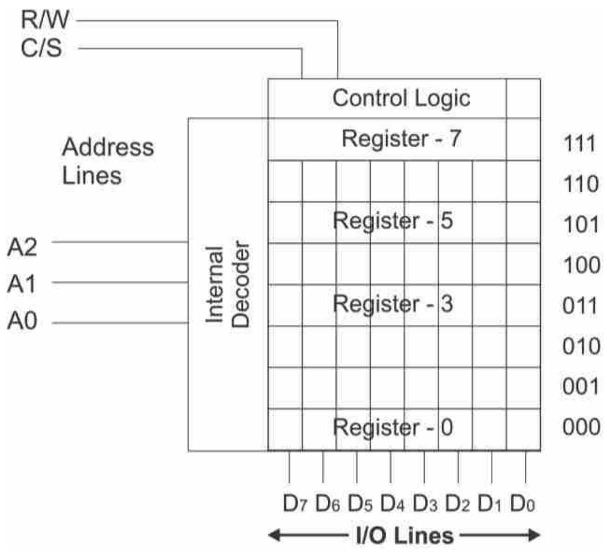


Fig. 11.7 Memory Chip with 8 registers

The Fig. 11.7 shows memory chip with eight registers. This memory chip will be selected for read/write operation when the signal chip select is low i.e. $\overline{CS} = 0$. The bubble provided for chip select indicates that it is a low active chip.

When the memory chip is selected, it is ready for read/write operation. Now the R/W signal is also low active. Therefore, $R/\overline{W} = 0$ indicates that the write operation can be performed. The 8 bit data from D0 to D7 can be stored through I/O lines.

The 8–bit data is stored into or read from the 8–bit register. As shown in the figure, there are eight numbers of eight bit registers. How would you select a particular register for read/write operation ? The answer is, through decoder. The diagram shows that the memory chip is connected to 3–bit decoder. The three address lines A0 to A2 will select a particular register as per address. This is shown in the following table 11.2.

Table 1.2 Decoder – RAM

Address input to decoder			Register
A2	A1	A0	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

❖ **Memory Map :**

Memory map is defined as the assignment of address to memory registers in various memory chips in the system. In 8085 system, entire memory map ranges from 0000 H to FFFF H (216 = 65536).

Let us design the memory map for 256 byte memory chip. There are 256 registers provided. Therefore, the number of address lines for decoder input will be calculated as follows :

$2^n = 256$ Therefore n= 8

The 8085 processor is provided with 16 address lines (A0 – A15). Let us select eight lines (A0 – A7) for giving address input to the decoder. Remaining eight lines (A8 – A15) are utilized for chip select function. The bobbled chip select terminal is connected to eight lines using bobbled NOT Gate and NAND gate as shown in the figure 11.8.

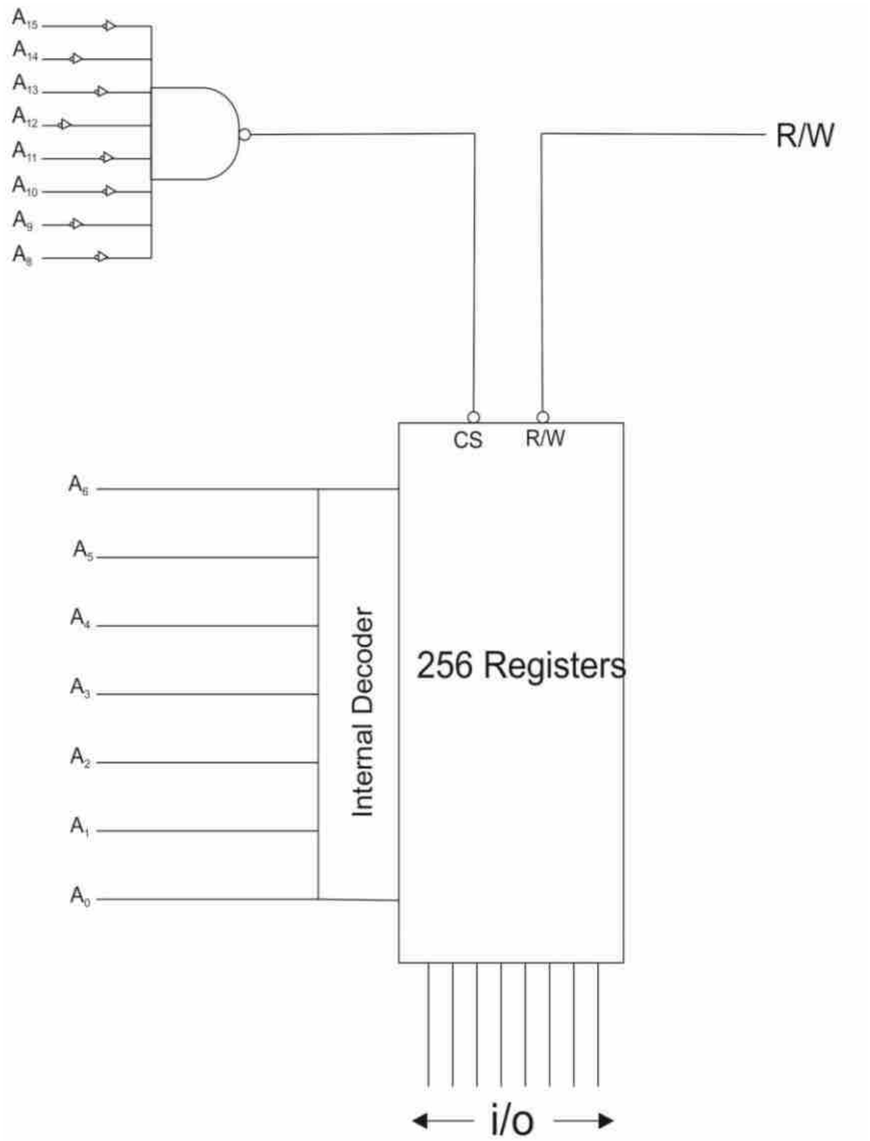


Fig. 11.8 Memory Map for 256 Byte Memory

The range of lines A0–A7 will be 00 to FF H. The value of all signals from A8 to A15 must be 0 for selecting the chip.

❖ **Memory Map of 1 K Chip :**

The 1 K chip is 1024 X 8 bit combination. In order to select 1024 registers, number of address lines required would be calculated as follows –

2^n =1024 Therefore n = 10

Thus, out of 16 address lines (A15 – A0), ten lines (A9–A10) will be required for decoder input. Remaining six lines (A15–A10) are utilized for enabling the 1 K chip.

The chip select terminal is connected to bobbled NOT and NAND gate as shown in Figure 11.9.

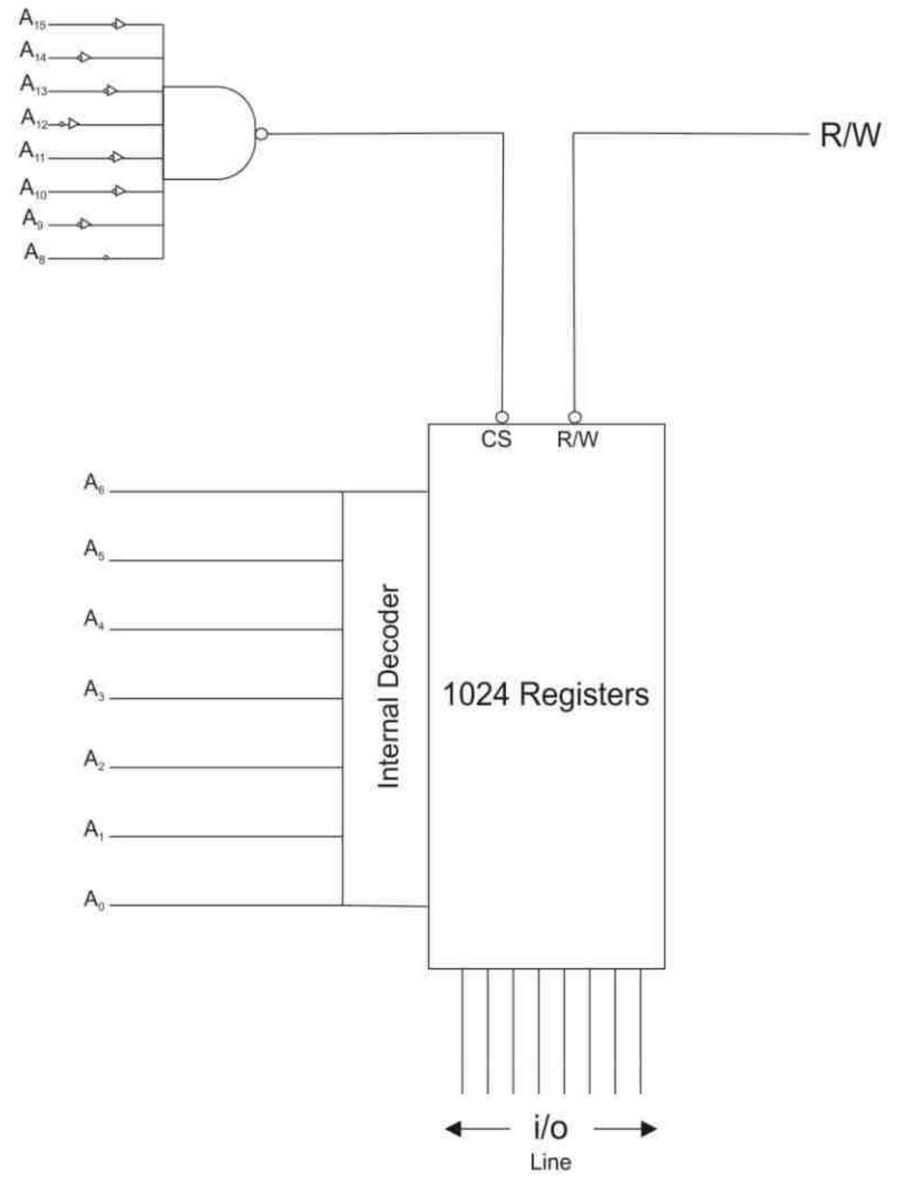


Fig. 11.9 Memory Map of 1 K memory chip

In this, the memory map ranges from 0000 H to 03FF H. In terms of page analogy, we can devise the 1 K memory into four pages, each page having 256 lines, as follows :

00 00 00 FF	Page 0 with 256 lines
01 00 01 FF	Page 1 with 256 lines
02 00 02 FF	Page 2 with 256 lines
03 00 03 FF	Page 3 with 256 lines

❖ **Memory Mapped I/O and I/O Mapped I/O :**

The microprocessor generates low active signals MEMR, MEMW, IOR and IOW from IO/M, RD and WR, as shown in Fig 11.10.

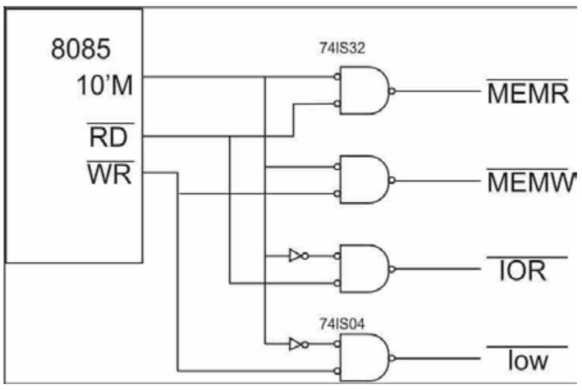


Fig. 11.10 Schematic to Generate Read/Write Control Signals for memory and IO

❑ **Check Your Progress :**

1. In Diode ROM ________ is used to store a bit.

   a. RAM b. Diode c. Memory d. none

2. In diode ROM, if a 2 bit decoder is presentthen it would have ______ outputs

   a. 8 b. 4 c. 16 d. 32

3. Which is not a non–volatile memory ?

   a. RAM

   b. Programmable read–only memory (PROM)

   c. Erasable programmable read–only memory (EPROM)

   d. Electrically erasable programmable read–only memory (EEPROM)

4. Cache memory is used by ________.

   a. RAM b. CPU c. HDD d. ROM

5. ________ RAM uses flip flop.

   a. Static b. Dynamic c. DDR d. DDR3

11.3 Let Us Sum Up :

In this unit, we have learnt that storage systems can be classified as primary storage and secondary storage. The primary storage semiconductor memory devices are Read Only Memory (ROM), Random Access Memory (RAM) whereas secondary storage systems mainly comprises of magnetic memory and optical memory. The name is given as Read Only Memory (ROM) because data stored in ROM cannot be modified. Unlike RAM, the 'write' feature is not available in ROM. The ROM is mainly used to distribute firmware. (Firmware is software that is very closely tied to specific hardware). The desired data permanently stored in ROM and thus can never be modified. In case of PROM, initially all the diodes are provided with fusible links. The links which are not required are burnt using a device called PROM Programmer.

RAM : It is read/write computer memory which can be accessed randomly. RAMs are of two types : static and dynamic. Static RAM uses flip–flops to store bits and so consumes current whether they are storing a 1 or a 0. Dynamic RAM uses capacitors to store charges and use less power.

11.4 Answers for Check Your Progress :

❑ **Check Your Progress :**

1. (b), 2. (b), 3. (a), 4. (b), 5. (a)

11.5 Glossary :

1. **RAM** – It is called as Random Access Memory an internal memory that can be read from as well as write.

2. **ROM** – It is called as Read Only Memory a memory that can only read and cannot write from.

3. **Firmware** – It is software that is very closely tied to specific hardware

4. **Memory Map** – It is an assignment of address to memory registers in various memory chips in the system

11.6 Assignment :

Explain diode ROM.

11.7 Activities :

Draw the memory map of 2K memory

11.8 Case Study :

Explain the internal structure of ROM along with proper diagram.

11.9 Further Readings :

1. 8080/8085 Microprocessor Book, Intel Marketing Communications, Wiley–Intel Series
2. 8085 Microprocessors : Programming and Interfacing, By N. K. Srinath, PHI Learning Private Ltd.
3. 8085 Software Design, C. A. Titus, SAMS Publishing
4. Schaum's Outline Series of theory and Problems on Microprocessor Fundamentals, R. L. Tokheim, Mc–Graw Hill Publishing
5. Microprocessor Architecture, Programming and Applications with the 8085 / 8080 A., Ramesh Gaonkar, Wily Eastern Limited

Unit 12

MEMORY – II

UNIT STRUCTURE

12.0 Learning Objectives :

After learning this unit, you will be able to understand :

- The concept of magnetic storage
- The comparison between primary storage and secondary storage
- The working of Hard Disk Drive
- The working of floppy disk drive
- The working of CD and DVD

12.1 Introduction :

In the previous unit you have studied about the primary memories that is, about RAM, ROM and cache memories. We have also discussed about the general arrangement of these memories.

This unit will also discuss about the type of memory but here it is secondary memory. Secondary memories are used to save, take the back up of data and once the power supply is switched off, the contents stored in these memories are not erased. Let's discuss the same types of memories now.

12.2 Secondary Storage :

Secondary storage devices are used to save, back up and even transport files consisting of data or programs from one location or computer to other. Let us study the secondary storage devices.

In microprocessor based systems, the secondary storage devices are connected to DMA Controllers.

❖ Hard Disk Drives :

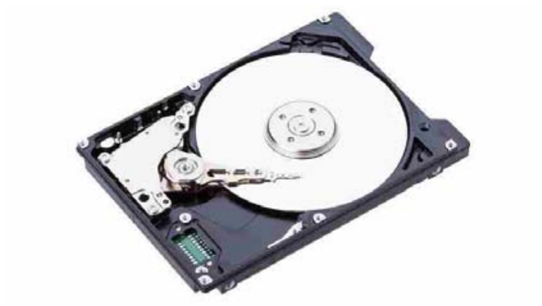


Fig 12.1 Hard disk

A hard disk drive is a non–volatile storage device which stores digitally encoded data on rapidly rotating platters with magnetic surfaces. The word 'drive' refers to a device distinct from its medium, such as a tape drive and its tape, or a floppy disk drive and its floppy disk. Early HDDs had removable media; however, an HDD today is typically a sealed unit with fixed media

HDDs use magnetising technology to record data which is shown as binary 0 or a 1. It reads data by locating magnetization of material. It carries a spindle that keeps one or more flat circular disks known as platters. Platters are obtained from non–magnetic material and are coated with thin layer of magnetic material.

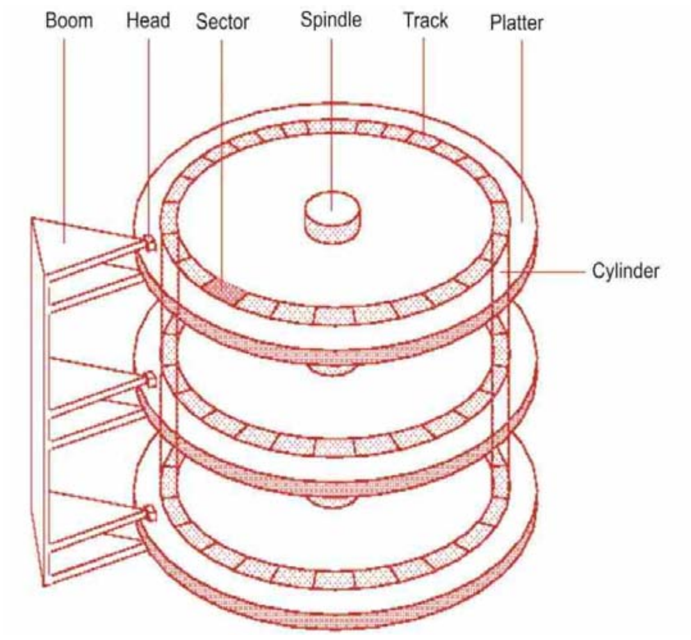


Fig. 12.2 Hard disk Cylinders

The platters are spun at very high speeds. Information is written to a platter as it rotates fast devices called read–and–write heads that operate very close (tens of nanometers in new drives) over the magnetic surface. The read–and–write head is used to detect and modify the magnetization of the material immediately under it. There is one head for each magnetic platter surface on the spindle, mounted on a common arm. An actuator arm (or access arm) moves the heads on an arc (roughly radially) across the platters as they spin, allowing each head to access almost the entire surface of the platter as it spins. The arm is moved using a voice coil actuator or in some older designs a stepper motor.

❖ **Floppy Disk Drive (FDD) :**

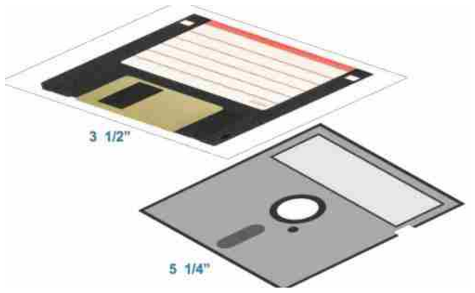


Fig. 12.3 Floppy Disk

Just like hard disk, floppy disk is also stores data which is made of disk of thin, flexible magnetic storage medium encased in a square or rectangular plastic shell called jacket. These are made up of Mylar. Sata is stored as electromagnetic charges on a metal oxide film coating the Mylar plastic. Data and programs are represented by a presence or absence of these charges using ASCII or EBCDIC code.

Floppy disks are read and written by a floppy disk drive or FDD. The floppy disk was invented by IBM. The floppy disks are available in three sizes of 8–inch (200 mm), 5¼–inch (133.35 mm) and 3½–inch (90 mm), the 3½–inch disks are labeled as 2HD which means "double sided, high density". These disks have the capacity of 1.44 Megabytes.

Fig. 12.4 Floppy Disk Drive

On the surface the disks are having circular rings called tracks. These tracks are concentric circles. Each track is divided into wedge shaped sections called sectors.

❖ **Compact Disks :**

CD–ROM is a hardware which is used to read and write CD or DVD disks

Fig. 12.5 CD ROM

Five years later, CD–ROM drives were being introduced on to computers. In 1994, they called a computer with a CD–ROM a Multimedia computer since it could play music and specially coded videos. Companies like Creative Technologies created a Sound Blaster multimedia upgrade kit which at the time gave the user a CD–ROM Drive with driver, a sound card and speakers. This in 1994 was a $200 product. Windows 95 in 1995 was introduced on either

10, 3.5–inch disks or 1 CD–ROM. CD–ROMs are used mostly to store and distribute computer software, games, applications and data.

In optical disk technology, a laser beam alters the surface of plastic or metallic disk to represent data. Unlike floppy and hard–disks, which use magnetic charge to represent 1s and 0s, optical disks use reflected light. The 1s and 0s are represented by flat areas called lands and bumpy areas called pits respectively on the disk surface. The disk is read by the laser that projects a tiny beam of light on these areas. The amount of reflected light determines whether the area represents 1 or 0.

Optical disks are available in market in various sizes such as 3½, 4¼, 5¼, 8, 12 and 14 inches. The most common size is 4¼ inch. The data is stored on these disks in different ways and different formats. The most common are CD and DVD.

Compact Disk format : The CD format is the most widely used format today. CD drives are standardized on many microcomputer systems. The CD drive can store data from 10 MB to 750 MB. The rotational speed of the CD determines how fast the data can be transferred from the CD. For example, the 24X CD can transfer 3.6 MB per second; while a 32X CD can transfer 4.8 MB per second. The faster the drive, the faster data can be read from the CD and CD–RW.

CD–ROM (Compact Disk Read Only) Memory : It is similar to commercial music CD. Read only means it cannot be written on or erased by the user. User has an access to data imprinted by the publisher. CD–ROM's are used to distribute large databases and references. These are used to distribute large software application packages.

CD–R (CD Recordable) : Can be written only once. After, they can be read many times without deterioration but cannot be written or erased.

CD–RW (Compact Disk Rewritable) : It is also known as erasable optical disk. The surface of CD–RW disk is not altered permanently when the data is recorded. As they can be changed, CD–RW is often used to develop multimedia applications.

DVD (Digital Versatile Disk) : DVD and DVD drives are very similar to CD except that more data can be packed in same amount of space. The DVDs are available in four forms – DVD–ROM, DVD–R (DVD Recordable), DVD RW (DVD rewritable) and DVD–RAM (DVD Random Access Memory).

Data play : It is the optical write–once format similar to CD–RW. This format is designed for optical disks the size of the quarter with the capacity of 500 MB. Users are able to record data such as music files, digital photographs etc. However the prerecorded music in Data Play format is more difficult to copy than with music stored in traditional CDs.

The summary of optical storage system is given in the Table 12.3.

Table 12.1 Summary of Optical Storage System

Format	Capacity	Type	Description
CD	650 MB to 1 GB	CD–ROM	Read only, used to distribute databases, reference books and software.
		CD–R	Write one time only, used to archive large amount of data.
		CD–RW	Reusable, used to create and edit large multimedia presentations.
DVD	4.7 GB to 17 GB	DVD–ROM	Read only, used to distribute theatrequality video and sound presentations.
		DVD–R	Written to one time only, used to archive large amount of data.
		DVD+ RW	Reusable, able to create and read CD disks, used to create and edit large scale multimedia presentations.
		DVD–RAM	Similar to DVD+RW except unable to create and read CDs.
Data play	500 MB		Specialized format for digital photography and music.

❖ **Magnetic Tape :**

If you have to select a particular song from the storage device, there are two ways of doing this. Either you have direct access to that song or you have to access song by a method of fast forward or reverse. The direct access is provided by disks. On the other hand, the sequential access is provided by tapes.

The magnetic tape or tape cartridge is a plastic tape coated with magnetic oxide. Application of magnetic tapes was more in mainframes.

Compared to disks, the data access performed by tapes is slower. However, these can be used for storage of backup data.

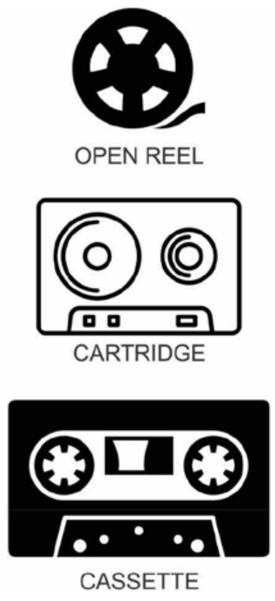


Fig. 12.6 CD Cartridge and Cassette

❖ **Universal Serial Bus (USB) :**

Universal Serial Bus (USB) is a serial bus standard to connect devices to a host computer. USB was designed to allow many peripherals to be connected using a single standardized interface socket and to improve plug and play capabilities by allowing hot swapping; that is, by allowing devices to be connected and disconnected without rebooting the computer or turning off the device. Other convenient features include providing power to low–consumption devices, eliminating the need for an external power supply; and allowing many devices to be used without requiring manufacturer–specific device drivers to be installed.

USB is intended to replace many varieties of serial and parallel ports. USB can connect computer peripherals such as mouse, keyboards, PDAs, gamepads and joysticks, scanners, digital cameras, printers, personal media players, flash drives and external hard drives. For many of those devices, USB has become the standard connection method. USB was designed for personal computers, but it has become commonplace on other devices such as PDAs and video game consoles and as a power cord between a device and an AC adapter plugged into a wall plug for charging.

The design of USB is standardised by the USB Implementers Forum (USB–IF), an industry standards body incorporating leading companies from the computer and electronics industries.

❖ **Bluetooth :**

It is a wireless protocol which can share data over long distances from fixed as well as mobile devices with the help of personal area networks. It is a kind of standard communications protocol which is utilized for low power consumption having short range with low cost.

❑ **Check Your Progress :**

1. Secondary Storage Devices are :

   a. Hard Disk b. Floppy Disk c. CD ROM d. All of above

2. Which doesn't come under CDs ?

   a. CD–R b. CD–RW c. Pen Drive d. CD ROM

3. The magnetic tape is a _______ tape coated with magnetic oxide

   a. Metal b. glass c. plastic d. None

4. ________ is a serial bus standard to connect devices to a host computer.

   a. USB b. VGA c. PS/2 d. None

5. Compared to disks, the data access performed by tapes is ________.

   a. faster b. equal c. slower d. None

12.3 Let Us Sum Up :

It is studied that a hard disk is non–volatile device in which the data is stored for longer duration.

Floppy disk is also a storage device made of thin disk which is rectangular in shape.

Data play is the optical write–once format similar to CD–R. This format is designed for optical disks the size of the quarter with the capacity of 500 MB.

The magnetic tape or tape cartridge is a plastic tape coated with magnetic oxide.

Internet Hard Drives (Also called i–drive or on–line storage) is a new concept in database management systems. The data storage is done by using space available on internet.

Universal Serial Bus (USB) is a serial bus standard to connect devices to a host computer. USB was designed to allow many peripherals to be connected using a single standardized interface socket and to improve plug and play capabilities by allowing hot swapping; that is, by allowing devices to be connected and disconnected without rebooting the computer or turning off the device.

Infrared technology is what most TV remotes use. The distance an infrared signal can travel varies based on the strength of the remote, but is usually less than 50 feet for household electronics. In order for an infrared signal to be detected, there must be a direct line of sight between the transmitter (remote) and the receiver (TV). If there is a wall or large object between them, the signal will not pass through

12.4 Answers for Check Your Progress :

❑ **Check Your Progress :**

1. (d), 2. (c), 3. (c), 4. (a), 5. (c)

12.5 Glossary :

1. **Hard Disk** – It is storage media inside the Computer System.

2. **CD ROM** – It is Compact Disc Read Only Memory used for recording information's.

3. **CD–RW** – It is Compact Disk Rewritable where you can store and delete information any time

12.6 Assignment :

Give the summary of optical storage systems.

12.7 Activities :

Why do you think that secondary storage devices are important ?

12.8 Case Study :

Obtain more information about bluetooth from internet and discuss its advantages.

12.9 Further Readings :

1. 8080/8085 Microprocessor Book, Intel Marketing Communications, Wiley–Intel Series.
2. 8085 Microprocessors : Programming and Interfacing, N. K. Srinath, PHI Learning Private Ltd software Design, C. A. Titus, SAMS Publishing.
3. Schaum's Outline Series of theory and Problems on Microprocessor Fundamentals, R. L. Tokheim, Mc–Graw Hill Publishing.
4. Microprocessor Architecture, Programming and Applications with the 8085 / 8080 A., Ramesh Gaonkar, Wily Eastern Limited.

Unit 13

CACHE MEMORY ORGANIZATION

UNIT STRUCTURE

13.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Importance of cache in memory organization.
- Cache performance can be measured in terms hit ratio.
- Significance of implementing multilevel cache in CPU organization.
- Data mapping between main memory and cache.
- Working principle of cache.

13.1 Introduction :

Cache memory is used to increase the execution of large programs. But as we increase the main memory the cost of the system will also increase, which will further increase the requirement of the power supply as well as more cooling requirements are needed. So, the capacity of the main memory cannot be increased beyond a certain limit. This problem can be solved by using a cache memory.

It is one of the smallest and fastest memory in the memory hierarchy. As shown the figure shows that the cache memory is always deployed between the main memory and the CPU. The main objective of the cache memory is to reduce the access time. Because the main memory is slow, it takes more time for data to either read or write.

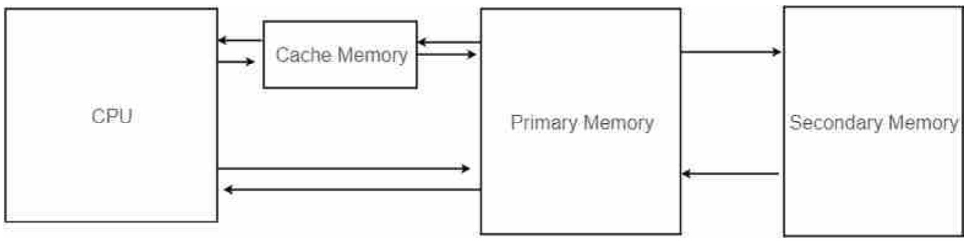


Fig. 13.1 : Cache memory

The fundamental of the cache memory is to keep frequently used data and instructions in the cache, thus reducing the access time. It keeps a part of the program from the main memory. When the CPU needs to access memory, the cache is searched for the data first. If the word is found in the cache, it is accessed from the cache, but if not then access to the main memory is done and simultaneously the word is copied into the cache.

Generally the transfer of words from the main memory to the cache is done in blocks. A block consists of many words. As the capacity of the cache memory is small there is a possibility of the cache being full and some existing item has to be replaced. Replacement of the item is based on different algorithm.

13.2 Levels of Cache Memory :

There are basically three levels of cache, L1 cache, L2 cache and L3 cache.

L1 cache, or main cache, is exceptionally fast but comparatively small, and is usually embedded in the processor chip as CPU cache.

L2 cache, or secondary cache, is often more voluminous than L1. L2 cache may be embedded on the CPU, or it can be on a different chip or coprocessor and have a high–speed substitute system bus linking the cache and CPU. That way it doesn't get slowed by traffic on the main system bus.

Level 3 (L3) cache is dedicated memory developed to progress the performance of L1 and L2. L1 or L2 can be appreciably quicker than L3, though L3 is typically twice the speed of DRAM. With multicore processors, every core can have devoted L1 and L2 cache, but they can share an L3 cache. If an L3 cache references an instruction, it is usually prominent to a advanced level of cache.

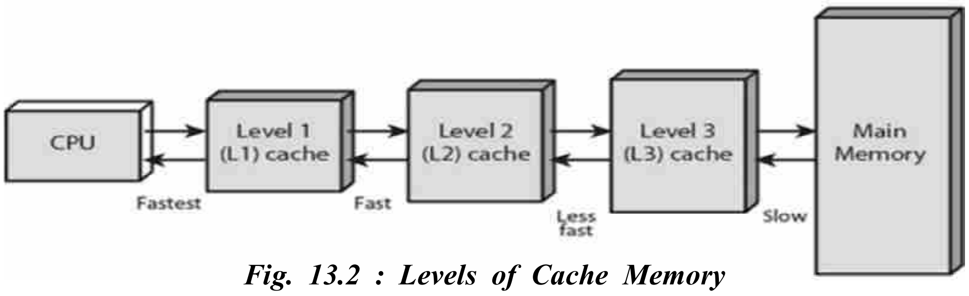


Fig. 13.2 : Levels of Cache Memory

In the history, L1, L2 and L3 caches have been formed by means of combined processor and motherboard elements. Recently, the drift has been toward consolidating every three levels of memory caching on the CPU itself. That's why the main means for increasing cache size has begun to move from the attainment of a precise motherboard with dissimilar chipsets and bus architectures to buying a CPU with the correct amount of combined L1, L2 and L3 cache.

13.3 Measuring the Cache Performance :

When the CPU access the main memory the cache controller checks to see whether the data is inside the cache or not. If the data is in the cache it is termed as "CACHE HIT" and if the data is not present it is known as "CACHE MISS".

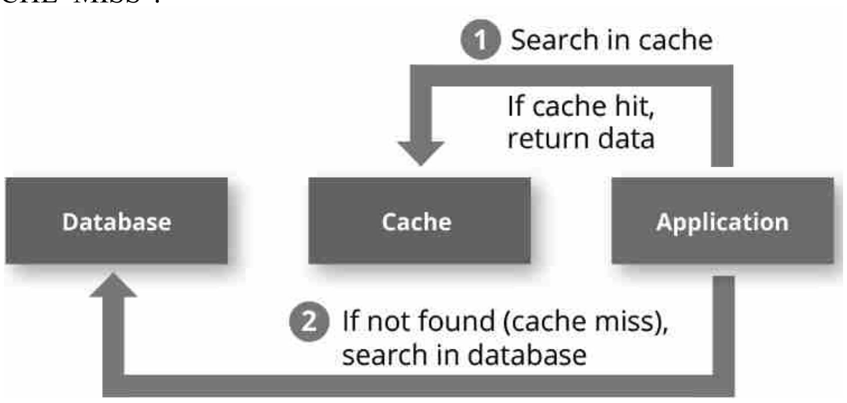

1 Search in cache
If cache hit, return data
Database
Cache
Application
2 If not found (cache miss), search in database

Fig. 13.3 : Cache HIT/MISS

Hence the whole block from the primary memory is brought into the cache and at the same time the processor is given the data that is required.

The performance of the cache is measured in terms of HIT ratio. It is the ratio of the number of hits divided by the total number of references to the memory.

HIT RATIO = Number of HITS / Total number of memory references.

13.4 Cache Mapping :

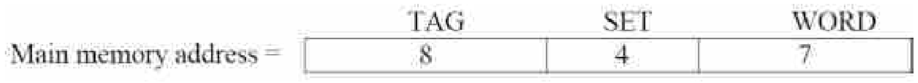
The basic feature of cache memory is its fast access time. Thus, no time should be wasted in searching the data in cache. The processor generates the address of the word to be accessed from the main memory. But the word is searched in the cache memory which is smaller in size. The transformation of data from main memory to cache memory is known as mapping.

There are basically three types of mapping :

- Direct mapping
- Associative mapping
- Set–associative mapping

13.4.1 Direct Mapping :

In this scheme, block i of the main memory maps into the block frame i modulo 128 of the cache. The memory address consists of three fields : tag, block and word as shown the figure.



	TAG	SET	WORD
Main memory address =	8	4	7

Fig. 13.4 : Main Memory Address Format

Direct mapping scheme is the simplest of all. It has the advantage f permitting simultaneous access to the desired data and the tag. Also it does not requires trivial replacement algorithms as from all the blocks that map into a block frame, only one can actually be in the cache at a time.

Direct mapping has disadvantage that the cache hit ratio reduces sharply if two or more blocks, used alternately, map onto the same block frame in the cache. The possibility of this contention is high in multiple–stream cache system compared to uni–processor as many concurrently active streams are sharing the cache.

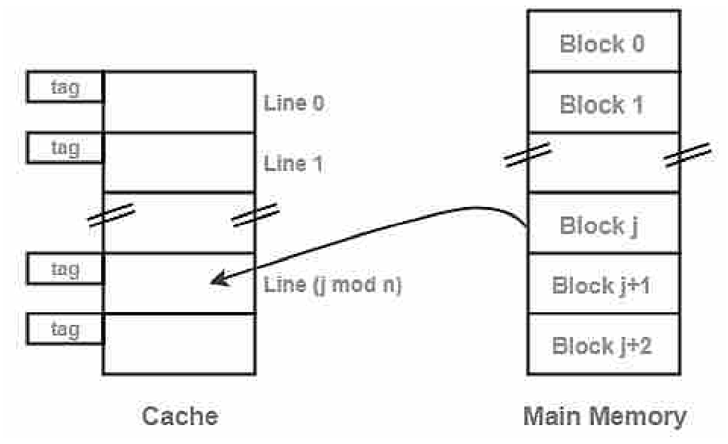


Fig. 13.5 : Direct Mapping

13.4.2 Associative Mapping :

The most fast and flexible cache uses associative memory organization. In associative mapping, any block in memory can be in any block frame of cache memory. When a request comes for a block, all the entries are compared simultaneously to determine if the requested block is present or not.

As in our example the format of the memory address is shown in the figure.

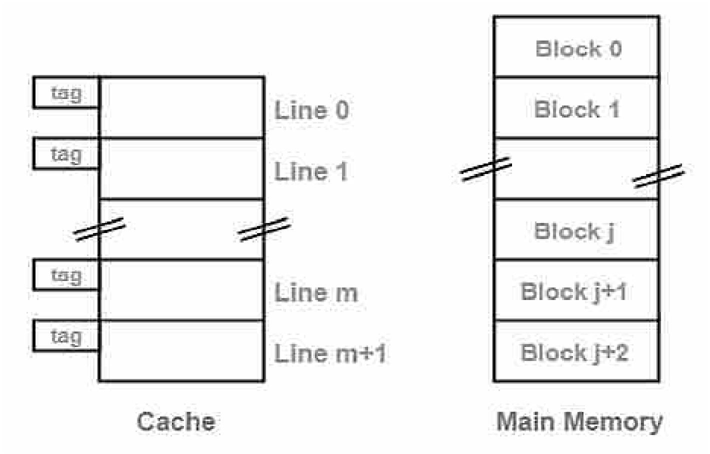


Fig. 13.6 : Associative Mapping

Here the tag filed is required to identify the memory block when it is present in the cache. The mapping flexibility permits wide variety of replacement algorithms. Although associative mapping gives the flexibility of placing a block anywhere in cache, it also encounters longer access time because of associative search.

13.4.3 Set Associative Mapping :

In set associative mapping there is a compromise between direct and associative mapping organization. Set–associative mapping is an improvement over the direct mapping and here the cache is divided into S sets. A block i in memory can be in any block frame belonging to the set i modulo S, as shown in the figure.

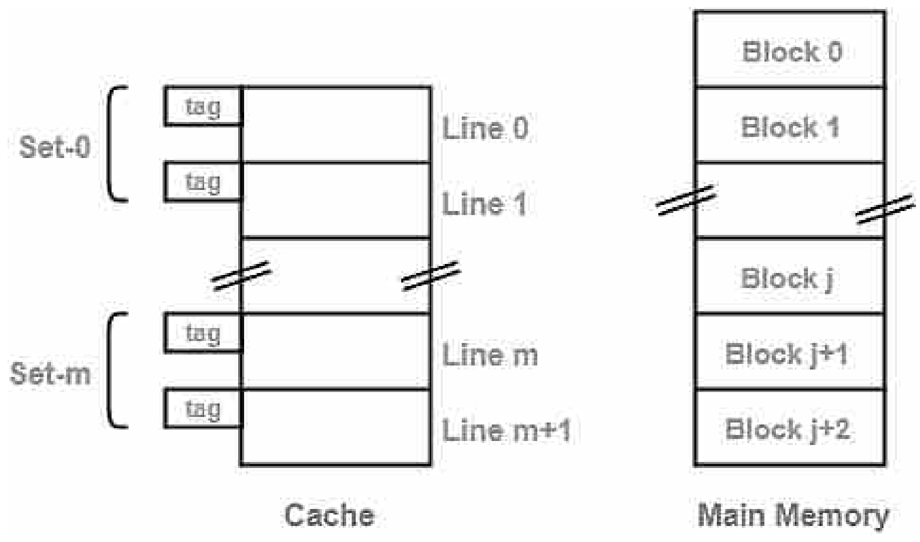


Fig. 13.7 : Two Way Set Associative Mapping

13.5 Cache Coherence :

In a multiprocessor system, data discrepancy may occur among neighboring levels or inside the similar level of the memory hierarchy. For instance, the cache and the main memory may have incompatible copies of the same item.

As multiple processors operate in parallel, and separately several caches may possess dissimilar copies of the similar memory block, this creates **cache coherence problem. Cache coherence schemes** help to keep away from this hitch by maintaining a consistent state for each cached block of data.

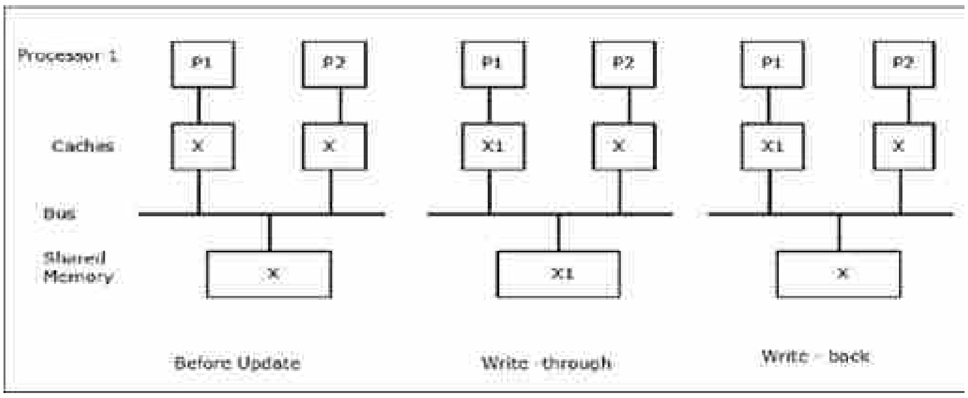


Fig. 13.8 : Two Way Set Associative Mapping

Let X be an element of shared data which has been referenced by two processors, P1 and P2. In the beginning, three copies of X are constant. If the processor P1 writes a new data X1 into the cache, by using **write-through policy**, the same copy will be written right away into the common memory. In this case, discrepancy occurs between cache memory and the main memory. When a write-back policy is used, the main memory will be restructured when the customized data in the cache is replaced or invalidated.

In general, there are three sources of discrepancy problem –

- Sharing of writable data
- Process migration
- I/O activity

Snoopy Bus Protocols

Snoopy protocols achieve data consistency between the cache memory and the shared memory through a bus-based memory system. Write-invalidate and write-update policies are used for maintaining cache constancy.

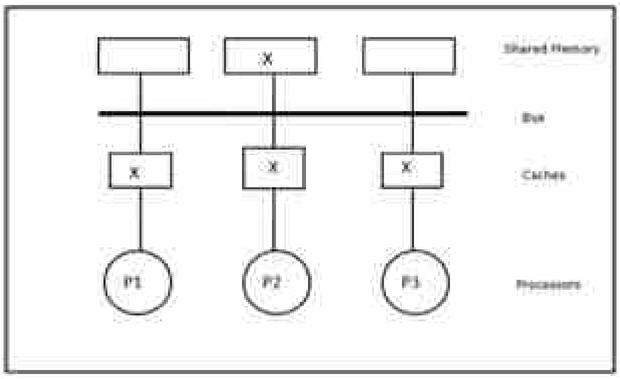


Fig. 13.9(a) : Block X in shared memory

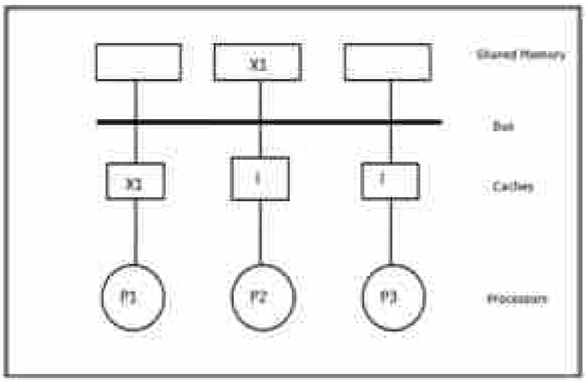


Fig. 13.9(b) : After write invalidate operation by P1

In this case, we have three processors P1, P2, and P3 having a constant copy of data element 'X' in their local cache memory and in the shared memory (Figure-13.9(a)). Processor P1 writes X1 in its cache memory using **write-invalidate protocol**. So, all other copies are invalidated via the bus. It is denoted by 'I' (Figure-13.9(b)). Invalidated blocks are also known as **dirty**, i.e. they should not be used. The **write-update protocol** updates all the cache copies via the bus. By using **write back cache**, the memory copy is also updated (Figure-13.10).

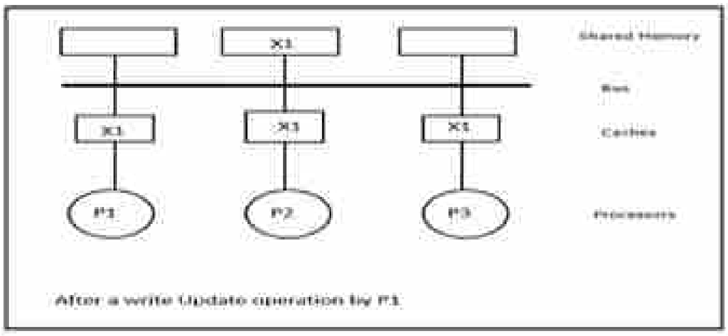


Fig. 13.10 : Write back cache

Cache Events and Actions

Following events and actions occur on the execution of memory-access and invalidation commands –

- **Read-miss** – When a processor wants to read a block and it is not in the cache, a read-miss occurs. This initiates a **bus-read** operation. If no dirty copy exists, then the main memory that has a constant copy, supplies a copy to the requesting cache memory. If a dirty copy exists in a isolated cache memory, that cache will hold back the main memory and send a copy to the requesting cache memory. In both the cases, the cache copy will enter the suitable state following a read miss.
- **Write-hit** – If the copy is in unclean or **reserved** state, write is done close by and the new state is dirty. If the new state is valid, write-invalidate command is broadcasted to all the caches, invalidating their copies. When the common memory is written through, the ensuing state is reserved after this first write.
- **Write-miss** – If a processor fails to write in the local cache memory, the reproduction must come moreover from the main memory or from a distant cache memory with a unclean block. This is done by sending a **read-invalidate** command, which will invalidate all cache copies. Then the home copy is updated with unclean state.
- **Read-hit** – Read-hit is always performed in local cache memory devoid of causing a transition of state or using the snoopy bus for invalidation.
- **Block replacement** – When a copy is dirty, it is to be written back to the main memory by block replacement technique. However, when the copy is either in valid or reserved or invalid state, no substitute will take place.

Directory-Based Protocols

By using a multistage system for construction a huge multiprocessor with hundreds of processors, the snoopy cache protocols need to be customized to suit the network capabilities. Broadcasting being very costly to perform in a multistage network, the consistency commands is sent only to those caches that keep a copy of the block. This is the reason for expansion of directory-based protocols for network-connected multiprocessors.

In a directory-based protocols system, data to be shared are located in a general directory that maintains the coherence among the caches. Here, the directory acts as a strain where the processors ask authorization to load an entry from the primary memory to its cache memory. If an entry is changed the directory either updates it or invalidates the other caches with that entry.

Hardware Synchronization Mechanisms

Synchronization is a special form of communication where in its place of data control, in order is exchanged between communicating processes residing in the same or dissimilar processors.

Multiprocessor systems use hardware mechanisms to put into practice low-level organization operations. Most multiprocessors have hardware mechanisms to enforce atomic operations such as memory read, write or read-modify-write operations to execute some synchronization primitives. Other than atomic memory operations, some inter-processor interrupts are also used for synchronization purposes.

Cache Coherency in Shared Memory Machines

Maintaining cache coherency is a problem in multiprocessor system when the processors contain local cache memory. Data inconsistency between different caches easily occurs in this system.

The major concern areas are –

- Sharing of writable data
- Process migration
- I/O activity

13.6 Applications of Cache Memory :

- Generally, the cache memory can accumulate a realistic number of blocks at any specified time, but this number is little compared to the total number of blocks in the main memory.
- The communication among the main memory blocks and those in the cache is given by a mapping function.
- It is used in minimizing the read write overheads.

13.7 Locality of Reference :

The cache works on the principle of locality of reference. Generally the main memory locations that are to be accessed are physically very close to the main memory locations which are being currently accessed. The reason for this may be understood by considering that a program flows in a straight line until a function or a loop is encountered. When a loop is executed the CPU refers to the same set of instructions in the memory. And when a subroutine is executed, its set of instructions is fetched from the memory.

Thus the subroutines and the loops localize the references made to the memory. So while accessing the main memory, not only the instruction which is needed is fetched but several consecutive instructions are also fetched and stored in the cache memory.

As shown in the figure the main memory is divided into blocks each consisting of fixed number of locations. While a location is accessed the entire block is brought and stored into the cache memory.

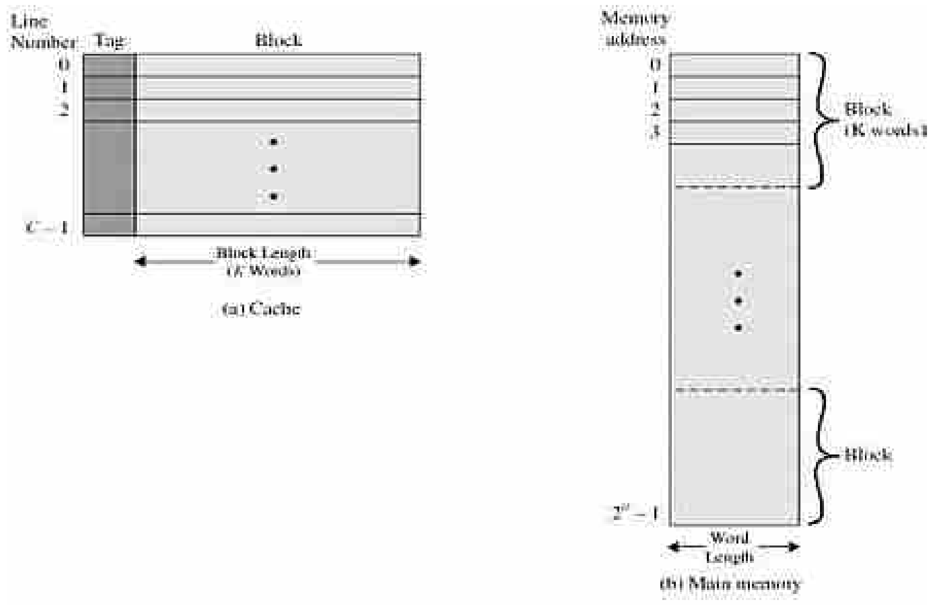


Fig. 13.8 : Locality of Reference

Also when a block is stored in cache memory it referred as a line or block frame which is of same size as of as the block in the main memory. Hence any memory block can be mapped into one of the cache line.

❑ **Check Your Progress :**

1. _______ cache is considered to be the primary cache.

   a. L1 b. L2 c. L3 d. L4

2. Cache principle is based on locality of ___________.

   a. Hit ratio b. Reference c. Miss ratio d. None of these

3. _______ mapping is the simplest of all cache mapping algorithms.

   a. Associative b. Set–Associative

   c. Direct d. None of these

4. Performance of the cache is based on _____ ratio.

   a. Hit b. Miss c. Both a and b d. None of these

5. In ______ mapping cache lines are grouped together into sets.

   a. Associative b. Set–Associative

   c. Direct d. None of these

13.8 Lets Us Sum Up :

Cache memory is used to increase the execution of large programs.

The fundamental of the cache memory is to keep frequently used data and instructions in the cache, thus reducing the access time.

There are basically three levels of cache, L1 cache, L2 cache and L3 cache.

The cache works on the principle of locality of reference.

There are basically three types of mapping : Direct mapping, associative mapping and set associative mapping.

The cache performance is measured in terms of HIT ratio.

For superior performance in a multiprocessor system, every processor will frequently have its individual cache. Cache coherence refers to the difficulty of maintaining the data in these caches consistent. The main problem is dealing with writes by a processor.##x0A;##x0A;There are two all-purpose strategies for dealing with writes to a cache:##x0A;##x0A;€ Write-through - All data written to the cache is also written to memory at the identical time.##x0A;€ Write-back - When data is written to a cache, a dirty bit is set for the affected block. The customized block is written to memory only as soon as the block is replaced.

13.9 Answers to Check Your Progress :

❑ **Check Your Progress :**

1. (a), 2. (b), 3. (c), 4. (a), 5. (b)

13.10 Glossary :

- **L1 :** Level one cache
- **L2 :** Level two cache
- **L3 :** Level three cache

- **Routine or Subroutine :** They are referred as procedure or a function or a piece of code which can be executed or called anywhere in the program.

13.11 Assignment :

Check your computing devices in which different levels of cache are used and explore its usage.

13.12 Activities :

Search and list out the devices where cache memory is used extensively and state the impact of it.

13.13 Case Study :

Why not register memory is used in place of cache memory ? Can cache be replaced by any other memory technology ?

13.14 Further Readings :

1. 8080/8085 Microprocessor Book, Intel Marketing Communications, Wiley–Intel Series.
2. 8085 Microprocessors : Programming and Interfacing, N. K. Srinath, PHI Learning Private Ltd software Design, C. A. Titus, SAMS Publishing.
3. Schaum's Outline Series of theory and Problems on Microprocessor Fundamentals, R. L. Tokheim, Mc–Graw Hill Publishing.
4. Microprocessor Architecture, Programming and Applications with the 8085/8080 A., Ramesh Gaonkar, Wily Eastern Limited.

Unit 14

OVERVIEW OF HIGHER VERSIONS

UNIT STRUCTURE

14.0 Learning Objectives :

After learning this unit, you will be able to understand :

- Higher versions of Intel family
- Comparison between the performance of 80286 and 80386 processors
- How to write simple programs in 8085
- The overview of different processors
- The features of Intel Pentium processors

14.1 Introduction :

In the previous units you have studied about the features of different types of microprocessors. Now, in this unit we will continue our discussion about the same, that is, we will be talking about the other types of processors. The 80286, 80386 and features of other processors are explained in this unit.

After going through this unit, you'll be able to gain much knowledge about different types of processors.

14.2 Intel 80286 :

The microprocessor 80286 was introduced by Intel Corporation in the year 1982. It is the 68 pin integrated circuit having a maximum clock rate of 25 MHz. It is a 16–bit microprocessor with 134,000 transistors. It was the first Intel processor that could run all the software written for its predecessors, the Intel 8086 and Intel 8088. It was widely used in IBM PC compatible computers during the mid–1980s to early 1990s, starting when IBM first used it in the IBM PC/AT in 1984.

The 80286's performance per clock cycle is considered more than twice than to its predecessors, that is, Intel 8086 and Intel 8088. Access to the addressing modes was fast as compared to 8086. Complex mathematical operations such as MUL/DIV took fewer cycles as compared to 8086. With the help of 24–bit address bus, 80286 will perform addressing up to 16 MB of RAM.

The 286 was designed to run multitasking applications, including communications (such as automated PBXs, real time process control and multi–user systems.

Features of 80286

- High performance microprocessor with memory management and protection
- 80286 is the first member of the family of advanced microprocessors with built in/on-chip memory management and protection abilities primarily designed for multi-user/multitasking systems.
- Available in 12.5MHz, 10MHz & 8MHzclock frequencies
- The 80286 CPU, with its 24-bit address bus is able to address 16MB of physical memory.
- Intel 80286 has 2 operating modes:
  - o Real Address Mode :
    - ■ 80286 is just a fast 8086 --- up to 6 times faster
    - ■ All memory management and protection mechanisms are disabled
    - ■ 286 is object code compatible with 8086
  - o Protected Virtual Address mode :
    - ■ 80286 works with all of its memory management and protection capabilities with the advanced instruction set.
    - ■ It is source code compatible with 8086
- 286 includes special instructions to support operating system.
- For example, one instruction can
  - o ends the current task
  - o save its states
  - o switch to a new task
  - o load its states and
  - o begin executing the new task

❑ **Check Your Progress – 1 :**

1. Which is incorrect in case 80286 ?

   a. It is a 68 pin integrated circuit

   b. It is having clock speed of 25 MHz

   c. It is a 32–bit microprocessor

   d. All

14.3 Intel 80386 :

The 80386 was 32–bit microprocessor, which was introduced by Intel in 1985. The initial description of 80386 carries 275,000 transistors moreover these were used as the central processing unit of several computers as well as workstations. The first 80386 microprocessor based computer was designed by Compaq.

It was seen that 8086 architecture is an actual configuration of 32–bit extensions; the 80386 instruction set, programming model as well as binary encodings are at present the similar denominator for 32–bit x86 processors. As seen the codes intend for previous 16–bit x86 processors configuration such as 8088 and 80286 were a result of problem which was corrected by 80386.

Today, 64–bit x86 processors can handle and perform many programs that was used for earlier chips with successively todays implementations of similar architecture that have become much faster than original 80386. A 33 MHz 80386 was reportedly measured to operate at about 11.4 MIPS.

After 80386, there appears 80286 with 16–bit processor. 80386 carries 32 bit architecture and paging translation which is easy to work with operating systems.

There exist three operating modes in 80386 such as :

- real mode
- protected mode
- virtual mode

The protected mode in 80286 is limited in comparison to 80386 that uses address up to 4 GB. A 32–bit memory model of 80386 featured for x86 processor family with the release of AMD. After 80386 will be 80486 and Intel Pentium.

❖ **Features of 80386 :**

- 8, 16, 32-Bit Data Types
    - o 8 General Purpose 32-Bit Registers
- Very Large Address Space
    - o 4 Gigabyte Physical
    - o 64 Terabyte Virtual
    - o 4 Gigabyte Maximum Segment Size
- Integrated Memory Management Unit
    - o Virtual Memory Support
    - o Optional On-Chip Paging
    - o 4 Levels of Protection
    - o Fully Compatible with 80286
- Object Code Compatible with All 8086 Family Microprocessors
- Virtual 8086 Mode Allows Running of 8086 Software in a Protected and Paged System
- Hardware Debugging Support

- Optimized for System Performance
  - o Pipelined Instruction Execution
  - o On-Chip Address Translation Caches
  - o 20, 25 and 33 MHz Clock
  - o 40, 50 and 66 Megabytes/Sec Bus Bandwidth
- Numeric Support via Intel387TM DX Math Coprocessor
- Complete System Development Support
  - o Software : C, PL/M, Assembler
  - o System Generation Tools
  - o Debuggers : PSCOPE, ICETM-386
- High Speed CHMOS IV Technology
  - o 132 Pin Grid Array Package
  - o 132 Pin Plastic Quad Flat Package

❑ **Check Your Progress – 2 :**

1. Which is not the quality of 80386 ?

   a. It is a 16–bit microprocessor

   b. It was introduced by Intel in 1985

   c. It carries roughly 275,000 transistors

   d. The first computer 80386 was designed by Compaq

14.4 Intel 80486 :

It was launched in year 1989 by Intel. The common manufacturers of this processor are Intel, IBM, AMD, Texas Instruments, UMC, Thomson and SGS. The CPU clock rate ranges from 16 MHz to 50 MHz. The initial 80486 chip uses millions of transistors having large on–chip cache and integrated floating point unit.

❖ **Differences between 386 and 486 :**

- An 8 KB on–chip SRAM cache stores the most recently used instructions and data. The 386 had no such internal cache but supported a slower off–chip cache.
- Tightly coupled pipelining allows the 486 to complete a simple instruction like ALU reg, reg or ALU reg in every clock cycle. The 386 needed two clock cycles for this.
- Integrated FPU, which is disabled or absent in SX models, with a dedicated local bus gives faster floating point calculations compared to the i386+i387 combination.
- Improved MMU performance : The 486 has a 32–bit data bus and a 32–bit address bus. This required either four matched 30–pin (8 bit) SIMMs or one 72 pin (32 bit) SIMM on a typical PC motherboard. The 32 bit address bus means that 4 GB of memory can be directly addressed.

❖ **Features of 80486 :**

- One of the most obvious feature included in a 80486 is a built in math coprocessor.

- This coprocessor is essentially the same as the 80387 processor used with a 80386, but being integrated on the chip allows it to execute math instructions about three times as fast as a 80386/387 combination
- 80486 is an 8Kbyte code and data cache.
- It can execute 40 million instructions per second on average.
- From performance point of view the architecture of 80486 is far better than its precedent
- Updated MMU performance
- Tightly coupled pipe lines.
- New instructions are added such as XADD, BSWAP, CMPXCHG, INVD, WBINVD, INVLPG
- There are several other variants that are based on 80486 like Intel RapidCAD, i486SL-NM, i487SX, i486 OverDrive etc.
- 80486 yields almost doubled ALU performance at the same clock rate.

❑ **Check Your Progress – 3 :**

1. Which is a quality of 486 microprocessor ?

a. It has no internal cache	b. It requires two clock cycles
c. It has 32–bit data bus	d. It has 16–bit address bus

14.5 Intel Pentium :

Intel filed a U.S trademark for the name 'Pentium' on July 2, 1992, more than 8 months before the public release of the Intel Pentium chip with the description 'computer hardware : namely, microprocessors'.

❖ **Pentium Pro :**

The Pentium Pro is a sixth–generation x86 microprocessor developed and manufactured by Intel and was introduced in November 1995. It had a CPU clock speed rating from 150 MHz to 200 MHz and FSB speed from 60 MHz to 66 MHz.

❖ **Pentium II :**

The Pentium II processor was launched by Intel in 1997. It had a CPU clock rate from 233 MHz to 450 MHz. Its FSB speed ranged from 66 MHz to 100 MHz. It is provided with MMX technology.

❖ **Intel Pentium III :**

It is called as Intel's 32–bit x86 desktop as well as mobile microprocessors as per sixth–generation Intel P6 micro technology. The maximum CPU clock will range from 450 MHz to 1.4 GHz with 100 MHz to 133 MHz FSB.

❖ **Pentium IV :**

Pentium IV processor was launched by Intel in 2000. It carries CPU clock from 1.3 GHz to 3.8 GHz with speed of 400 MT/s to 1066 MT/s.

❖ **Intel Pentium V :**

It is a P5 micro architecture which is the advancement of 80486 architecture having dual integer pipelines. Earlier, Pentium MMX was invented with similar micro architecture with larger caches.

❑ Check Your Progress – 4 :

1. Which Intel series has clock speed between 100 MHz to 133 MHz ?

   a. Pentium Pro b. Pentium II c. Pentium III d. Pentium IV

2. The 80386 was ______ bit microprocessor.

   a. 16 b. 32 c. 64 d. 128

❖ Features of Intel Pentium :

- 64 bit data bus
- 8 bytes of data information can be transferred to and from memory in a single bus cycle
- Supports burst read and burst write back cycles
- Supports pipelining
- Instruction cache
- 8 KB of dedicated instruction cache
- Two Integer execution units, one Floating point execution unit
- Dual instruction pipeline to process more than one instruction per clock and achieve a high level of performance
- 256 lines between instruction cache and pre-fetch buffers; allows 32 bytes to be transferred from cache to buffer
- Data cache
- 8 KB dedicate data cache gives data to execution units
- 32 byte lines
- Two parallel integer execution units
- Allows the execution of two instructions to be executed simultaneously in a single processor clock
- Floating point unit
- It includes
- Faster internal operations
- Local advanced programmable interrupt controller
- Speeds up to 5 times for common operations including add, multiply and load, than 80486
- Branch Prediction Logic
- To reduce the time required for a branch caused by internal delays
- When a branch instruction is encountered, microprocessor begins pre-fetch instruction at the branch address
- Data Integrity and Error Detection
- Has significant error detection and data integrity capability
- Data parity checking is done on byte - byte basis
- Address parity checking and internal parity checking features are added
- Dual Integer Processor
- Allows execution of two instructions per clock cycle
- Functional redundancy check

- To provide maximum error detection of the processor and interface to the processor
- A second processor 'checker' is used to execute in lock step with the 'master' processor
- It checks the master's output and compares the value with the internal computed values
- An error signal is generated in case of mismatch
- Superscalar architecture
- Three execution units
- One execution unit executes floating point instructions
- The other two (U pipe and V pipe) execute integer instructions
- Parallel execution of several instructions - superscalar processor
- Integrated GPU (Graphics Processing Unit)
- Implementation of Hyper-threading technology that allows to improve parallelization of calculations performed on x86 microprocessors.
- Enhanced smart cache and shared L3chache

14.6 Let Us Sum Up :

In this unit, we learnt that, microprocessor 80286 was introduced by Intel Corporation in the year 1982. It is the 68 pin integrated circuit having a maximum clock rate of 25 MHz. It is a 16–bitmicroprocessor with 134,000 transistors. It was the first Intel processor that could run all the software written for its predecessors, the Intel 8086 and Intel 8088. It was widely used in IBM PC compatible computers during the mid–1980s to early 1990s, starting when IBM first used it in the IBM PC/AT in 1984.

The 80386 was 32–bit microprocessor, which was introduced by Intel in 1985. To make use of 80386, the first personal computer was designed and manufactured by Compaq.

The 80486 and Intel Pentium line of processors were descendents of the 80386 design.

Intel 80486 represents fourth generation of binary compatible CPU's.

The CPU clock rate ranges from 16 MHz to 50 MHz. It was also the first x86 chip, which used more than a million transistors, due to a large on–chip cache and an integrated floating point unit.

Intel filed a U.S trademark for the name 'Pentium' on July 2, 1992, more than 8 months before the public release of the Intel Pentium chip with the description 'computer hardware : namely, microprocessors'.

The Pentium Pro is sixth–generation x86 microprocessor by Intel with CPU clock speed from 150 MHz to 200 MHz and FSB speed from 60 MHz to 66 MHz.

The Pentium II processor by Intel was having CPU clock speed from 233 MHz to 450 MHz with FSB from 66 MHz to 100 MHz.

The Pentium III is Intel's 32–bit x86 desktop which is sixth–generation P6 micro architecture. It has CPU clock ranges from 450 MHz to 1.4 GHz and FSB speed from 100 MHz to 133 MHz.

14.7 Answers for Check Your Progress :

- **Check Your Progress 1 :**

  1. (c)

- **Check Your Progress 2 :**

  1. (a)

- **Check Your Progress 3 :**

  1. (c)

- **Check Your Progress 4 :**

  1. (c), 2. (b)

14.8 Glossary :

1. **Microprocessor** – It is a hardware chip comprises of chips and ICs that are fabricated.
2. **Pentium** – It is brand of Intel that manufactures different series microprocessors.

14.9 Assignment :

Discuss the various versions of Pentium.

14.10 Activities :

Obtain more information about 80486 and prepare a report on the same in not less than 150–180 words

14.11 Case Study :

Collect more information about 80286 microprocessors and explain it in your own words.

14.12 Further Readings :

1. 8080/8085 Microprocessor Book, Intel Marketing Communications, Wiley–Intel Series
2. 8085 Microprocessors : Programming and Interfacing, N. K. Srinath, PHI Learning Private Ltd.
3. 8085 Software Design, C. A. Titus, SAMS Publishing
4. Schaum's Outline Series of theory and Problems on Microprocessor Fundamentals, R. L. Tokheim, Mc–Graw Hill Publishing
5. Microprocessor Architecture, Programming and Applications with the 8085 / 8080 A., Ramesh Gaonkar, Wily Eastern Limited

BLOCK SUMMARY :

The block will aware the students about different storage devices used by microprocessor. The generalization and understanding of concepts of data storage on different devices is also explained with respect to their latest techniques.

Additionally this block also explains the importance of cache memory in speeding up the execution process of the CPU. It also explains how data is mapped from main memory to cache memory by using various mapping techniques.

The block will give detailed knowledge about different types of microprocessors manufactured by Intel. The students will be trained with practical examples and exercises that will help to learn and grab the subject easily.

BLOCK ASSIGNMENT :

❖ **Short Questions :**

1. Give the summary of optical storage systems ?
2. Explain diode ROM ?
3. Discuss the performance per clock cycle of 80286 ?
4. What do you mean by compact disk ? Explain its working. Explain various types of Compact disks ?
5. Discuss the salient features of PROM and EPROM ?
6. Explain the principle of cache in detail.
7. Discuss how the performance of cache can be measured by giving example.

❖ **Long Questions :**

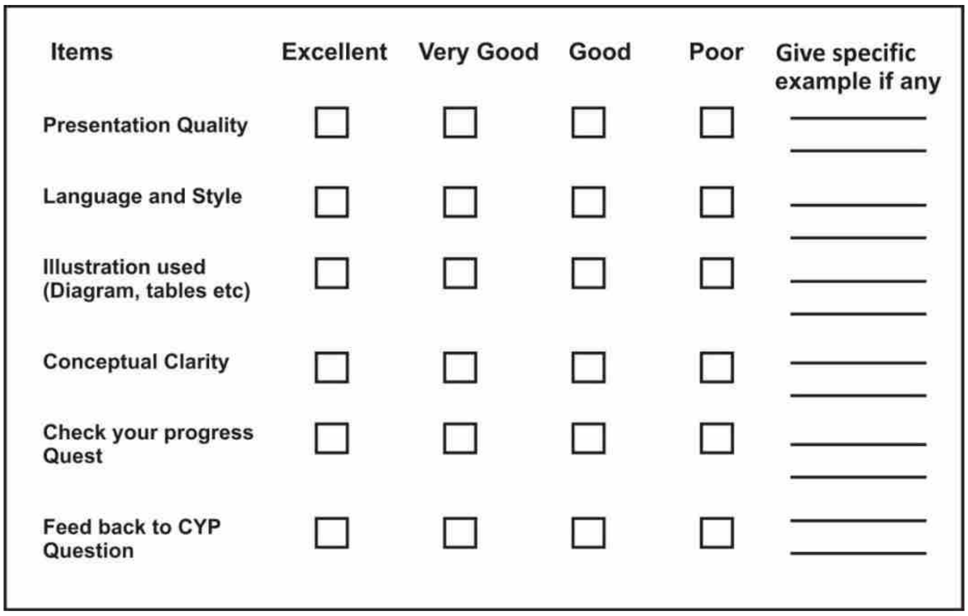
1. Explain hard disk and floppy disk drives ?
2. Explain the various operating modes of 80386 ?
3. Explain the salient features of Pentium V ?
4. Write a shot on the mapping algorithms used in cache.

❖ **Enrolment No. :**

1. How many hours did you need for studying the units ?

Unit No.	11	12	13	14
No. of Hrs.				

2. Please give your reactions to the following items based on your reading of the block :



Items	Excellent	Very Good	Good	Poor	Give specific example if any
Presentation Quality	☐	☐	☐	☐	
Language and Style	☐	☐	☐	☐	
Illustration used (Diagram, tables etc)	☐	☐	☐	☐	
Conceptual Clarity	☐	☐	☐	☐	
Check your progress Quest	☐	☐	☐	☐	
Feed back to CYP Question	☐	☐	☐	☐	

3. Any other Comments

....

....

....

....

....

....

....

....

BAOU
Education for All

स्वाध्यायः परमं तपः

DR.BABASAHEB AMBEDKAR
OPEN UNIVERSITY

'Jyotirmay' Parisar,
Sarkhej-Gandhinagar Highway, Chharodi, Ahmedabad-382 481.
Website : www.baou.edu.in