8086 Microprocessor and Intefacing

Ms. Roop Shikha

Ms. Manik Dhiman

Swami Vivekanand Institute
of Engineering & Technology

Electrical Engineering

KESHAV BOOKS

DELHI-110032

8086 Microprocesssor and Interfacing

First Published- 2023

ISBN : 978-93-87393-74-5

Price : 1325/-

Published by
KESHAV BOOKS
F-7, Gali No. 1, IIIrd Floor, Panchsheel Garden
Ext. Naveen Shahdara, Delhi-110032
Mob. : 9871603557
E-mail : keshavbooks30@gmail.com

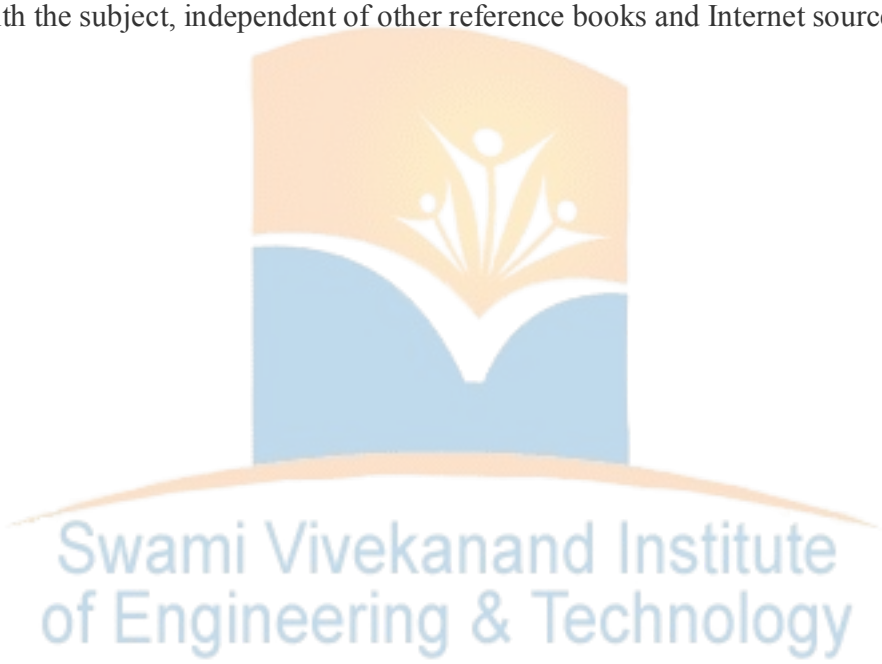
Printed at: GS Offset, Delhi.

PREFACE

We take an opportunity to present this Text Book on "8086 Microprocessor and Interfacing" to the students of engineering, this book offers an introduction to microprocessors and microcontrollers. The book is designed to explain basic concepts underlying programmable devices and their interfacing.

It provides complete knowledge of the Intel's 8086 microprocessors and 8255 microprocessor and their architecture, Pin configuration of 8255 PPI, 8259 PIC and assembly language.

The text has been organized in such a manner that a student can understand and get well-acquainted with the subject, independent of other reference books and Internet sources.



Ms. Roop Shikha

Ms. Manik Dhiman

Swami Vivekanand Institute
of Engineering & Technology

CONTENTS

CHAPTER 1

INTRODUCTION TO MICROPROCESSORS

Ms. Yukti Gupta[1] Ms. Roop Shikha[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

1.1 DIFFERENCES BETWEEN: MICROCOMPUTER, MICROPROCESSOR AND MICROCONTROLLER

- Microcomputer is a computer with a microprocessor as its CPU. Includes memory, I/O etc.
- Microprocessor is a silicon chip which includes ALU, register circuits & control circuits
- Microcontroller is a silicon chip which includes microprocessor, memory & I/O in a single package.

1.2 WHAT IS MICRO?

Micro is a new addition. In the late 1960's, processors were built using discrete elements. These devices performed the required operation, but were too large and too slow. It went directly from discrete elements to a single chip. However, comparing today's microprocessors to the ones built in the early 1970's you find an extreme increase in the amount of integration.

1.3 WHAT IS A MICROPROCESSOR?

The word comes from the combination of micro and processor. Processor means a device that processes whatever. In this context processor means a device that processes numbers, specifically binary numbers, 0's and 1's.To process means to manipulate. It is a general term that describes all manipulation. Again in this content, it means to perform certain operations on the numbers that depend on the microprocessor's design. It is a programmable device that takes in numbers, performs on them arithmetic or logical operations according to the program stored in memory and then produces other numbers.

As a Programmable device:

- The microprocessor can perform different sets of operations on the data it receives depending on the sequence of instructions supplied in the given program.
- By changing the program, the microprocessor manipulates the data in different ways as Instructions, Words, Bytes, etc.

- They processed information 8-bits at a time. That's why they are called —8-bit processors. They can handle large numbers, but in order to process these numbers, they broke them into 8-bit pieces and processed each group of 8-bits separately.

1.4 BLOCK DIAGRAM OF A MICROPROCESSOR

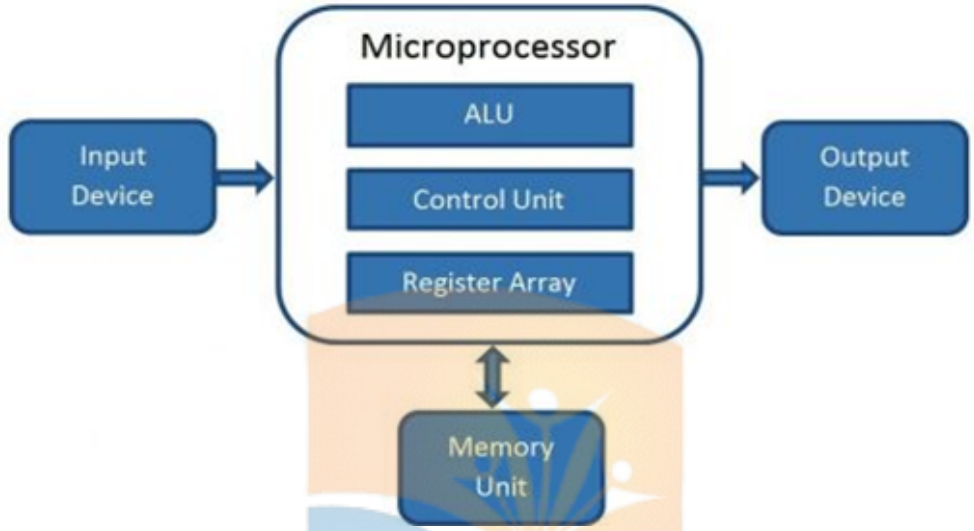


Fig. 1.1 Block diagram of a microprocessor

A microprocessor consists of an ALU, control unit and register array. Where **ALU** performs arithmetic and logical operations on the data received from an input device or memory. Control unit controls the instructions and flow of data within the computer. And, **register array** consists of registers identified by letters like B, C, D, E, H, L, and accumulator.

1.5 WHAT IS MEMORY?

Memory is the location where information is kept while not in current use. It is stored in memory. Memory is a collection of storage devices. Usually, each storage device holds one bit. Also, in most kinds of memory, these storage devices are grouped into groups of 8. These 8 storage locations can only be accessed together. So, one can only read or write in terms of bytes to and from memory. Memory is usually measured by the number of bytes it can hold. It is measured in Kilos, Megas and lately Gigas. A Kilo in computer language is 210 =1024. So, a KB (KiloByte) is 1024 bytes. Mega is 1024 Kilos and Giga is 1024 Mega. When a program is entered into a computer, it is stored in memory. Then as the microprocessor starts to execute the instructions, it brings the instructions from memory one at a time. Memory is also used to hold the data. The microprocessor reads (brings in) the data from memory when it needs it and writes (stores) the results into memory when it is done.

1.6 A MICROPROCESSOR-BASED SYSTEM

From the above description, we can draw the following block diagram to represent a microprocessor-based system as shown in fig 1.2. In this system, the microprocessor is the master and all other peripherals are slaves. The master controls all peripherals and initiates all operations. The buses are group of lines that carry data, address or control signals. The CPU interface is provided to demultiplex the multiplexed lines, to generate the chip select signals and additional control signals. The system bus has separate lines for each signal.

All the slaves in the system are connected to the same system bus. At any time instant communication takes place between the master and one of the slaves. All the slaves have tristate logic and hence normally remain in high impedance state. The processor selects a slave by sending an address. When a slave is selected, it comes to the normal logic and communicates with the processor.

The EPROM memory is used to store permanent programs and data. The RAM memory is used to store temporary programs and data. The input device is used to enter program, data and to operate system. The output device is also used for examining the results. Since the speed of IO devices does not match with speed of microprocessor, an interface device is provided between system bus and IO device.

CENTRAL PROCESSING UNIT

The CPU consists of ALU (Arithmetic and Logic Unit), Register unit and control unit. The CPU retrieves stored instructions and data word from memory; it also deposits processed data in memory.

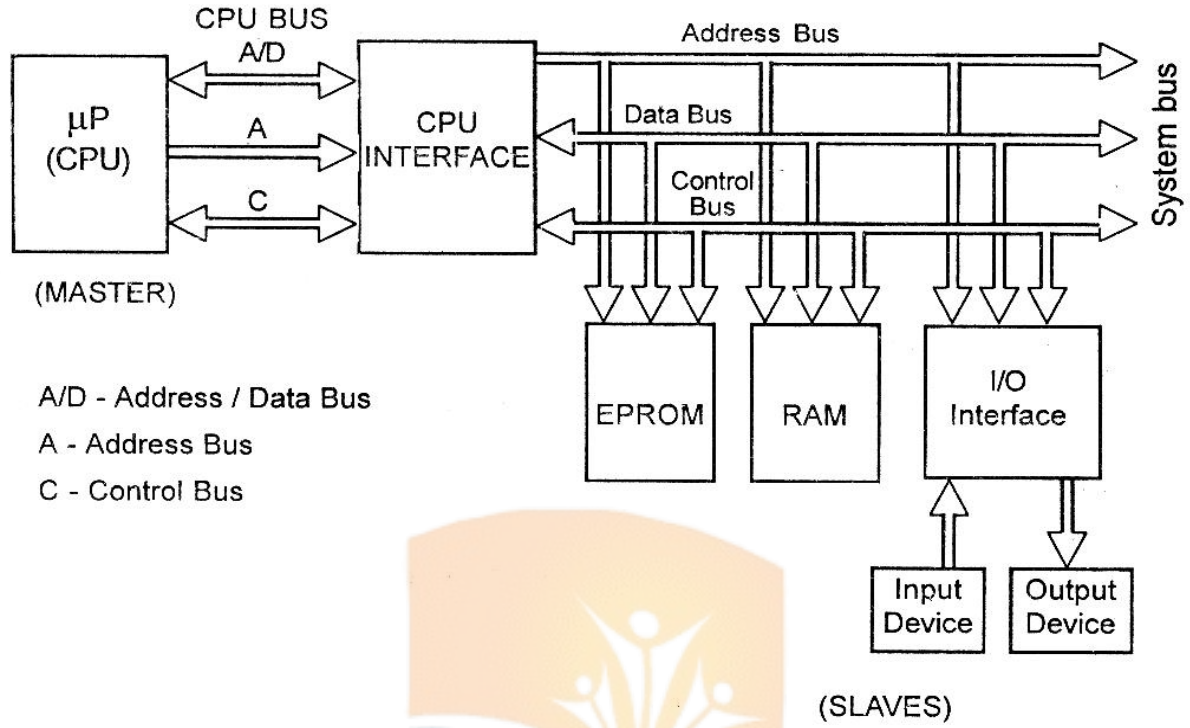


Fig.1.2 Microprocessor based system (organization of microcomputer)

ALU (ARITHMETIC AND LOGIC UNIT)

This section performs computing functions on data. These functions are arithmetic operations such as additions subtraction and logical operation such as AND, OR rotate etc. Result are stored either in registers or in memory or sent to output devices.

REGISTER UNIT

It contains various register. The registers are used primarily to store data temporarily during the execution of a program. Some of the registers are accessible to the uses through instructions.

CONTROL UNIT

It provides necessary timing & control signals necessary to all the operations in the microcomputer. It controls the flow of data between the p and peripherals (input, output & memory). The control unit gets a clock which determines the speed of the p.

The CPU basic functions

- It fetches an instructions word stored in memory.
- It determines what the instruction is telling it to do.(decodes the instruction)

- It executes the instruction. Executing the instruction may include some of the following major tasks.
- Transfer of data from reg. to reg. in the CPU itself.
- Transfer of data between a CPU reg. & specified memory location.
- Performing arithmetic and logical operations on data from a specific memory location or a designated CPU register.
- Directing the CPU to change a sequence of fetching instruction, if processing the data created a specific condition.
- Performing housekeeping function within the CPU itself in order to establish desired condition at certain registers.
- It looks for control signal such as interrupts and provides appropriate responses.
- It provides states, control, and timing signals that the memory and input/output section can use.

There are three buses:

ADDRESS BUS:

It is a group of wires or lines that are used to transfer the addresses of Memory or I/O devices. It is unidirectional. In Intel 8085 microprocessor, Address bus was of 16 bits. This means that Microprocessor 8085 can transfer maximum 16 bit address which means it can address 65,536 different memory locations. This bus is multiplexed with 8 bit data bus. So the most significant bits (MSB) of address goes through Address bus (A7-A0) and LSB goes through multiplexed data bus (AD0-AD7).

DATA BUS:

Data Bus is used to transfer data within Microprocessor and Memory/Input or Output devices. It is bidirectional as Microprocessor requires sending or receiving data. The data bus also works as address bus when multiplexed with lower order address bus. Data bus is 8 Bits long. The word length of a processor depends on data bus, thats why Intel 8085 is called 8 bit Microprocessor because it have an 8 bit data bus.

CONTROL BUS:

Microprocessor uses control bus to process data that is what to do with the selected memory location. Some control signals are Read, Write and Opcode fetch etc. Various operations are performed by microprocessor with the help of control bus. This is a dedicated bus, because all timing signals are generated according to control signal. The microprocessor is the master, which controls all the activities of the system. To perform a specific job or task, the microprocessor has to execute a program stored in memory. The program consists of a set of instructions stored in consecutive memory location. In order to execute the program the microprocessor issues address

and control signals, to fetch the instruction and data from memory one by one. After fetching each instruction it decodes the instruction and carries out the task specified by the instruction.

1.7 ADVANTAGES OF MICROPROCESSORS

- High-speed processing
- Brings intelligence to the system
- Is flexible in nature
- Has a compact size
- Is easy to maintain

1.8 DISADVANTAGES OF MICROPROCESSORS

- Leads to overheating due to continuous use.
- The data size decides the performance
- Larger than microcontrollers
- Doesn't support floating-point operations

1.9 APPLICATIONS OF THE MICROPROCESSORS

- It is present in single-board microcomputers as they use low configuration with software and hardware.
- It is embedded in the PC making it suitable to access and use applications.
- It's present in superminis and CAD enhancing their performance.
- It acts as an instrument because of its accepting programmability.
- It acts as a controller in many home appliances like toasters, televisions, stereo systems, etc. In the science industry, it is useful for measuring speed, temperature, moisture, etc.
- The telecom sector uses it for a digital telephone system, telephone exchange, and modem while the hospitality sector uses it for railway and airline reservation systems.
- Office automation uses it for word processing, spreadsheet operations, storage, etc.
- The publication uses it for automatic photocopies, high-quality printing, and good speed.
- Consumers are using it for toys, amusement devices, and house held devices frequently nowadays.
- It is also present in wireless communication equipment allowing them to interact and connect with devices.

1.10 REFERENCES

1. https://sist.sathyabama.ac.in/sist_coursematerial/uploads/SEC1310.pdf
2. https://www.javatpoint.com/microprocessor-introduction
3. https://data-flair.training/blogs/basics-of-microprocessor-evolution-types-applications-working/

CHAPTER 2

TYPES OF COMPUTERS

Ms. Vandana[1] Ms. Nisha[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

A computer is a device that transforms data into meaningful information. It processes the input according to the set of instructions provided to it by the user and gives the desired output.

2.1 TYPES OF COMPUTER

There are hundreds of different tasks that require the need for computers. But, a single type of computer performing each of these tasks is not an easy option. This is why with the advancement in technology we have come up with different types of computers. Each serves a different purpose.

In this chapter, let us look at some of these computers which are classified based on the-

1. Data type handling
2. Purpose they serve
3. Functionality

2.2 TYPES OF COMPUTERS BASED ON DATA TYPE

Based on the data type handling, computers can be categorized as Digital, Analog, and Hybrid.

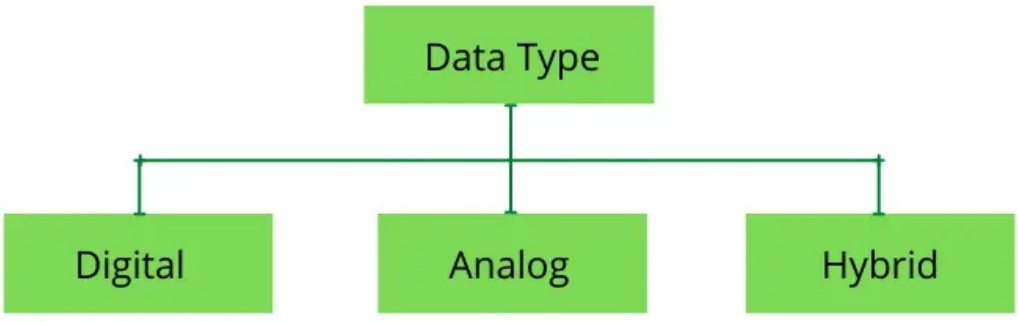


Fig. 2.1 Types of Computers Based on Data Type

Digital

Personal computers are an example of a digital computer. These computers accept input in the form of 0s and 1s. The computer processes binary input and provides the output. These computers perform all the logical & arithmetical operations. Any input given in any language is first converted into binary language and then the computer processes the information. Examples – laptops, PCs, mobile phones, desktops, etc.

Advantages of digital computers:

- It allows you to store a large amount of information and to retrieve it easily whenever you need it.
- You can easily add new features to digital systems more easily.
- Different applications can be used in digital systems just by changing the program without making any changes in hardware
- The cost of hardware is less due to the advancement in the IC technology.
- It offers high speed as the data is processed digitally.
- It is highly reliable as it uses error correction codes.
- Reproducibility of results is higher as the output is not affected by noise, temperature, humidity, and other properties of its components

Analog

These computers process analog data. Analog data keep varying. Hence, it does not have any discrete value. They read the continuous change in the input, process it, and then provide the output. Analog computers perform with equal diligence and accuracy. They are however slower than digital computers. They are also slightly less precise. Analog computers are for a Speedometer, thermometer, frequency, and signal of voltage, measuring the resistance of a capacitor.

Advantages of using analogue computers:

- It allows real-time operations and computation at the same time and continuous representation of all data within the rage of the analogue machine.
- In some applications, it allows performing calculations without taking the help of transducers for converting the inputs or outputs to digital electronic form and vice versa.

- The programmer can scale the problem for the dynamic range of the analogue computer. It provides insight into the problem and helps understand the errors and their effects.

Types of analogue computers:

- **Slide Rules**: It is one of the simplest types of **mechanical analogue computers**. It was developed to perform **basic mathematical calculations**. It is made of two rods. To perform the calculation, the hashed rod is slid to line up with the markings on another rod.

- **Differential Analysers**: It was developed to perform **differential calculations**. It performs integration using wheel-and-disc mechanisms to solve differential calculations.

- **Castle Clock**: It was invented by Al-Jarazi. It was able to save programming instructions. Its height was around 11 feet and it was provided with the display of time, the zodiac, and the solar and lunar orbits. This device also could allow users to set the length of the day as per the current season.

- **Electronic Analogue Computer**: In this type of analogue computer, electrical signals flow through capacitors and resistors to simulate physical phenomena. Here, the mechanical interaction of components does not take place. The voltage of the electrical signal generates the appropriate displays.

Hybrid

Hybrid computers are a mix of both analog and digital computers. These computers perform a high level of calculations. Hybrid computers are quick and efficient. They take input in analog form, convert it into digital form, and then process it to produce an output. Scientists are also using hybrid computers for complex calculations. For example, in hospitals to measure the heartbeat of the patients and at research institutes to measure earthquakes and other natural calamities.

Advantages of using hybrid computers:

- Its computing speed is very high due to the all-parallel configuration of the analogue subsystem.
- It produces precise and quick results that are more accurate and useful.
- It has the ability to solve and manage big equation in real-time.
- It helps in the on-line data processing.

2.3TYPES OF COMPUTERS BASED ON THE PURPOSE

Types of computers based on the purpose are shown in figure 2.2.

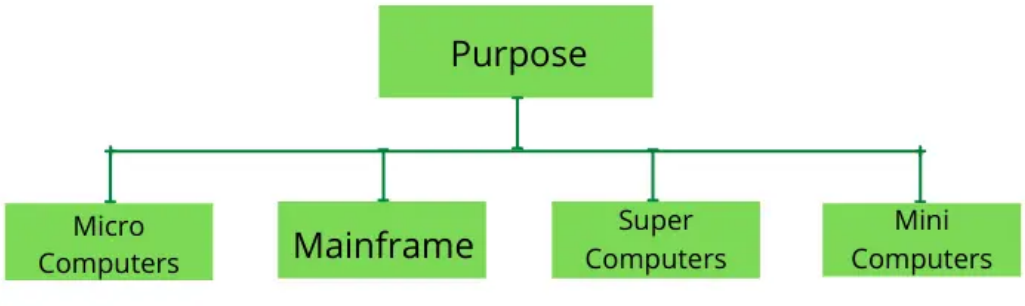


Fig. 2.2 Types of Computers Based on the Purpose

Microcomputer

Microcomputers are nothing but personal computers. These are single-chip systems. These are useful for personal use and can perform all the basic functions of the computer. Microcomputers require very little space and are comparatively inexpensive. Such computers have the most minimalistic requirement in terms of I/O devices. And have all the circuitry mounted on a single PCB. For example tablets, I pads, smartwatches, laptops, desktops.

Characteristics of a microcomputer:

- It is the smallest in size among all types of computers.
- A limited number of software can be used.
- It is designed for personal work and applications. Only one user can work at a time.
- It is less expansive and easy to use.
- It does not require the user to have special skills or training to use it.
- Generally, comes with single semiconductor chip.
- It is capable of multitasking such as printing, scanning, browsing, watching videos, etc.

Minicomputer

Standing in between a microcomputer and a mainframe computer is the minicomputer. These computers are useful if people around 5 to 300. Those who want to operate the system at the same time. You can see such computers at the billing counters of malls or large institutions.

Characteristics of miniframe or minicomputer:

- It is light weight that makes it easy to carry and fit anywhere.
- It is less expensive than mainframe computers.
- It is very fast compared to its size.
- It remains charged for a long time.
- It does not require a controlled operational environment.

Applications of minicomputers:

A minicomputer is mainly used to perform three primary functions, which are as follows:

- **Process control**: It was used for process control in manufacturing. It mainly performs two primary functions that are collecting data and feedback. If any abnormality occurs in the process, it is detected by the minicomputer and necessary adjustments are made accordingly.
- **Data management**: It is an excellent device for small organizations to collect, store and share data. Local hospitals and hotels can use it to maintain the records of their patients and customers respectively.
- **Communications Portal**: It can also play the role of a communication device in larger systems by serving as a portal between a human operator and a central processor or computer.

Mainframe

Mainframe computers are useful when a large number of people are involved. Like in the health care or retail sector who want to access data simultaneously. These computers process large amounts of data.

In addition, mainframe computers have evolved a lot over the years in terms of speed, size, and efficiency. These computers are just below the supercomputers. And sometimes are even more useful than a supercomputer. Examples – IBM z Series, System z9, etc.

Characteristics of Mainframe Computers:

- It can process huge amount of data, e.g. millions of transactions in a second in the banking sector.
- It has a very long life. It can run smoothly for up to 50 years after proper installation.
- It gives excellent performance with large scale memory management.
- It has the ability to share or distribute its workload among other processors and input/output terminals.

- There are fewer chances of error or bugs during processing in mainframe computers. If any error occurs it can fix it quickly without affecting the performance.
- It has the ability to protect the stored data and other ongoing exchange of information and data.

Applications of mainframe computers:

- In health care, it enabled hospitals to maintain a record of their millions of patients in order to contact them for treatment or related to their appointment, medicine updates or disease updates.
- In the field of defense, it allows the defense departments to share a large amount of sensitive information with other branches of defense.
- In the field of education, it helps big universities to store, manage and retrieve data related to their courses, admissions, students, teachers, employees and affiliated schools and colleges.
- In the retail sector, the retail companies that have a huge customer base and branches use mainframe computers to handle and execute information related to their inventory management, customer management, and huge transactions in a short duration.

Supercomputers

The biggest and fastest computers are supercomputers. Such computers can process trillions of functions within a few seconds. We generally use MPIS (Million Instructions per Second) to measure their performance. These computers are specifically designed for scientific applications such as –

1. Encryption decryption of passwords
2. Weather forecasting
3. Testing of nuclear weapons
4. Scientific research of earth and other planetary systems, etc.

Examples – PARAM supercomputer series, Gravity Pipe for astrophysics, Deep Crack for deciphering codes, etc.

2.4 TYPES OF COMPUTERS BASED ON THE FUNCTIONALITY

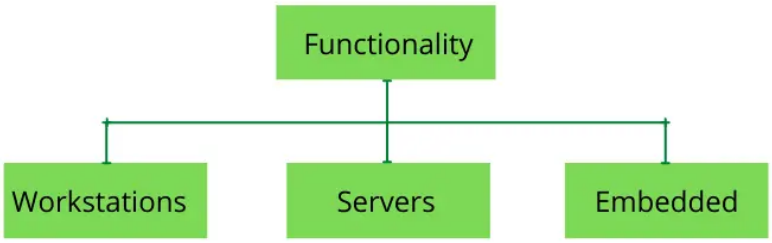


Fig. 2.3 Types of Computers Based on the Functionality

Workstations

Workstation computers are for single usage and professional purposes. These are like our basic laptops and desktops but with added superior features. For example, double-processor motherboard, added graphic card, ECC RAM, etc. The workstations are more powerful as compared to generic PCs. These can handle heavy-duty functions. Like animation, CAD, audio & video editing, professional gaming, etc. Examples: Apple PowerBook G4, SPARC CPU, MIPS CPU, etc.

Characteristics of workstation computer:

- It is a high-performance computer system designed for a single user for business or professional use.
- It has larger storage capacity, better graphics, and more powerful CPU than a personal computer.
- It can handle animation, data analysis, CAD, audio and video creation and editing.

Any computer that has the following five features can be termed as a workstation or can be used as a workstation.

- **Multiple Processor Cores**: It has more processor cores than simple laptops or computers.
- **ECC RAM**: It is provided with Error-correcting code memory that can fix memory errors before they affect the system's performance.
- **RAID (Redundant Array of Independent Disks)**: It refers to multiple internal hard drives to store or process data. RAID can be of different types, for example, there can be multiple drives to process data or mirrored drives where if one drive does not work than other starts functioning.
- **SSD**: It is better than conventional hard-disk drives. It does not have moving parts, so the chances of physical failure are very less.

- **Optimized, Higher end GPU**: It reduces the load on CPU. E.g., CPU has to do less work while processing the screen output.

Servers

These are hardware components or software programs that are built to assist other computers termed as clients. Together this architecture is called the client-server model. The client sends a request to the server and the server responds in return with a result or a solution. This proves that these server computers are more powerful than standard computers. The main purpose of these computers is to share data and resources with other computers. Different types of servers are useful for different needs and applications. For example, cloud server, application server, database server, file server, etc. Each of these servers has a different purpose for different client needs.

Embedded

These computers are mainly microcontroller-based systems. Used for processing specific tasks. Embedded computers have a combination of software and hardware components. But, these are usually a part of a larger system. Each of its components is designed from scratch to serve a specific purpose or complete a specific task. Another characteristic that distinguishes them from a standard PC is that all of its components are integrated into a single PCB or motherboard. They are most helpful for industrial use. This is because of their ruggedness. Examples: GPS systems, centralized heating systems, fitness trackers, digital watches, electronic calculators, etc.

2.5 REFERENCES

1. https://www.geeksforgeeks.org/types-of-computers/
2. https://artoftesting.com/types-of-computers
3. https://www.javatpoint.com/types-of-computer

CHAPTER 3

MICROPROCESSOR EVOLUTION AND TYPES

Mr. Manik Dhiman[1] Dr. Indu Batra[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Associate Professor, Swami Vivekanand Institute of Engineering & Technology*

When the central processor unit is made of only one integrated circuit, it becomes a microprocessor. It has millions of transistors and electronic components to process multiple instructions at a time. All this is on one silicon chip which has memory and other special features to support the computer system.

It is programmable in a way to read binary instructions from memory and then execute the task to deliver the needed output. It is useful for storing data, device interaction, and sending/receiving data simultaneously.

A Microprocessor has many components like transistors, registers, and diodes which come together to perform. The ability of the chip has become more complex with technology evolution. The functionality has become better and the speed has become faster.

Now, most devices need to have a microprocessor to function. It is the element that brings intelligence to a device. Be it a computer or mobile phone, all devices need an interface to handle data that only a microprocessor provides. And it still has a long way to go with the development of artificial intelligence.

3.1 BLOCK DIAGRAM OF A MICROPROCESSOR

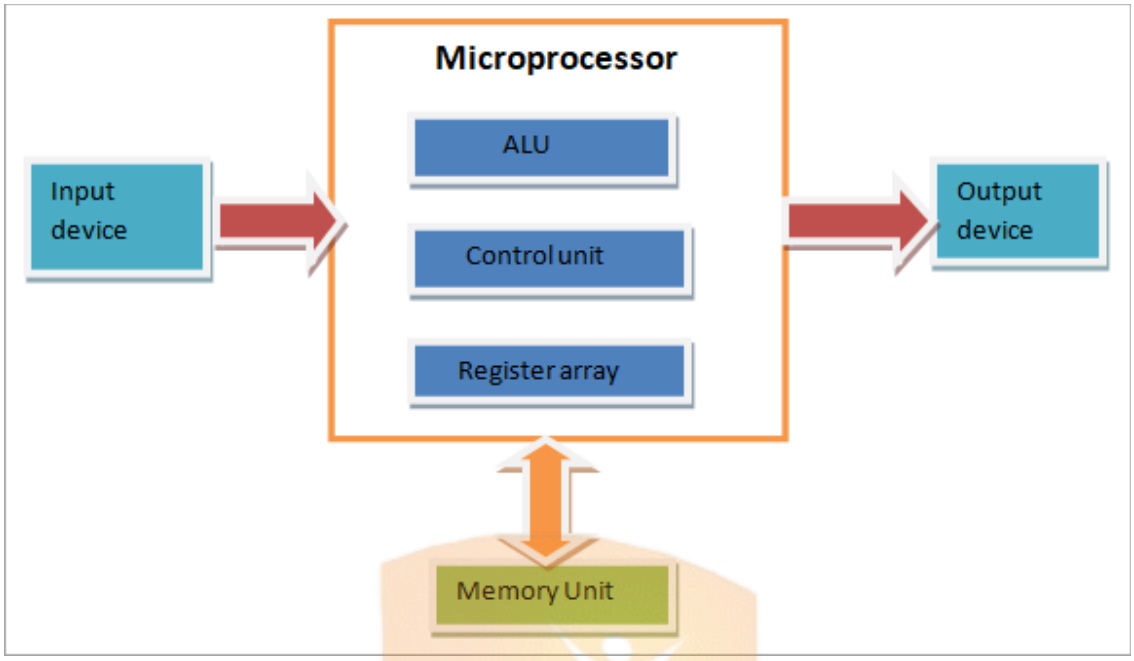


Fig. 3.1 Block Diagram of A Microprocessor

Arithmetic and Logical Unit, Control Unit, and Register array make up the microprocessor. The ALU deals with input devices or memory for receiving data. The control unit takes care of instructions and structure. Register array identifies and saves the registers like B, C, and accumulator.

3.2 WORKING OF A MICROPROCESSOR

There are three steps that a microprocessor follows –

1. Fetch – The instructions are in storage from where the processor fetches them.

2. Decode – It then decodes the instruction to assign the task further. During this, the arithmetic and logic unit also performs to register the data temporarily.

3. Execute – The assigned tasks undergo execution and reach the output port in binary form.

3.3 EVOLUTION OF MICROPROCESSORS

We can categorize the microprocessor according to the generations or according to the size of the microprocessor:

First Generation (4 - bit Microprocessors)

The first generation microprocessors were introduced in the year 1971-1972 by Intel Corporation. It was named **Intel 4004** since it was a 4-bit processor.

It was a processor on a single chip. It could perform simple arithmetic and logical operations such as addition, subtraction, Boolean OR and Boolean AND.

I had a control unit capable of performing control functions like fetching an instruction from storage memory, decoding it, and then generating control pulses to execute it.

Second Generation (8 - bit Microprocessor)

The second generation microprocessors were introduced in 1973 again by Intel. It was a first 8 - bit microprocessor which could perform arithmetic and logic operations on 8-bit words. It was Intel 8008, and another improved version was Intel 8088.

Third Generation (16 - bit Microprocessor)

The third generation microprocessors, introduced in 1978 were represented by **Intel's 8086, Zilog Z800 and 80286**, which were 16 - bit processors with a performance like minicomputers.

Fourth Generation (32 - bit Microprocessors)

Several different companies introduced the 32-bit microprocessors, but the most popular one is the **Intel 80386**.

Fifth Generation (64 - bit Microprocessors)

From 1995 to now we are in the fifth generation. After 80856, Intel came out with a new processor namely Pentium processor followed by **Pentium Pro CPU**, which allows multiple CPUs in a single system to achieve multiprocessing.

Other improved 64-bit processors are **Celeron, Dual, Quad, Octa Core processors**.

Table 3.1: Important Intel Microprocessors

Microprocessor	Year of Invention	Word Length	Memory addressing Capacity	Pins	Clock	Remarks
4004	1971	4-bit	1 KB	16	750 KHz	First Microprocessor
8085	1976	8-bit	64 KB	40	3-6 MHz	Popular 8-bit

						Microprocessor
8086	1978	16-bit	1MB	40	5-8 MHz	Widely used in PC/XT
80286	1982	16-bit	16MB real, 4 GB virtual	68	6-12.5 MHz	Widely used in PC/AT
80386	1985	32-bit	4GB real, 64TB virtual	132 14X14 PGA	20-33 MHz	Contains MMU on chip
80486	1989	32-bit	4GB real, 64TB virtual	168 17X17 PGA	25-100 MHz	Contains MMU, cache and FPU, 1.2 million transistors
Pentium	1993	32-bit	4GB real,32-bit address,64-bit data bus	237 PGA	60-200	Contains 2 ALUs,2 Caches, FPU, 3.3 Million transistors, 3.3 V, 7.5 million transistors
Pentium Pro	1995	32-bit	64GB real, 36-bit address bus	387 PGA	150-200 MHz	It is a data flow processor. It contains second level cache also,3.3 V
Pentium II	1997	32-bit	-	-	233-400 MHz	All features Pentium pro plus MMX technology,3.3 V, 7.5 million transistors
Pentium III	1999	32-bit	64GB	370 PGA	600-1.3 MHz	Improved version of Pentium II; 70 new SIMD instructions
Pentium 4	2000	32-bit	64GB	423 PGA	600-1.3 GHz	Improved version of Pentium III
Itanium	2001	64-bit	64 address lines	423 PGA	733 MHz-1.3 GHz	64-bit EPIC Processor

Where,

- **PGA** - Pin Grid Array
- **MMX** - multimedia extensions
- **EPIC** - Explicitly Parallel Instruction Computing
- **SIMD** - Single Instruction Multiple Data
- **ALU** - Arithmetic and Logic Unit
- **MMU** - Memory Management Unit
- **FPU** - Floating Point Unit

3.4 BASIC TERMS USED IN MICROPROCESSOR

Here is a list of some basic terms used in microprocessor:

Instruction Set - The group of commands that the microprocessor can understand is called Instruction set. It is an interface between hardware and software.

Bus - Set of conductors intended to transmit data, address or control information to different elements in a microprocessor. A microprocessor will have three types of buses, i.e., data bus, address bus, and control bus.

IPC (Instructions per Cycle) - It is a measure of how many instructions a CPU is capable of executing in a single clock.

Clock Speed - It is the number of operations per second the processor can perform. It can be expressed in megahertz (MHz) or gigahertz (GHz). It is also called the Clock Rate.

Bandwidth - The number of bits processed in a single instruction is called Bandwidth.

Word Length - The number of bits the processor can process at a time is called the word length of the processor. 8-bit Microprocessor may process 8 -bit data at a time. The range of word length is from 4 bits to 64 bits depending upon the type of the microcomputer.

Data Types - The microprocessor supports multiple data type formats like binary, ASCII, signed and unsigned numbers.

3.5 FEATURES OF MICROPROCESSOR

- **Low Cost** - Due to integrated circuit technology microprocessors are available at very low cost. It will reduce the cost of a computer system.

- **High Speed** - Due to the technology involved in it, the microprocessor can work at very high speed. It can execute millions of instructions per second.

- **Small Size** - A microprocessor is fabricated in a very less footprint due to very large scale and ultra large scale integration technology. Because of this, the size of the computer system is reduced.

- **Versatile** - The same chip can be used for several applications, therefore, microprocessors are versatile.

- **Low Power Consumption** - Microprocessors are using metal oxide semiconductor technology, which consumes less power.

- **Less Heat Generation** - Microprocessors uses semiconductor technology which will not emit much heat as compared to vacuum tube devices.

- **Reliable** - Since microprocessors use semiconductor technology, therefore, the failure rate is very less. Hence it is very reliable.

- **Portable** - Due to the small size and low power consumption microprocessors are portable.

3.6 ADVANTAGES OF A MICROPROCESSOR

- The microprocessor is that these are general purpose electronics processing devices which can be programmed to execute a number of tasks
- Compact size
- High speed
- Low power consumption
- It is portable
- It is very reliable
- Less heat generation
- The microprocessor is very versatile
- The microprocessor is its speed, which is measured in basically Hertz. For instance, a microprocessor with a measured speed 3 GHz, shortly GHz is capable of performing 3 billion tasks per second
- The microprocessor is that it can quickly move data between the various memory location

3.7 DISADVANTAGES OF A MICROPROCESSOR

- The main disadvantages are it's overheating physically
- It is only based on machine language

- The overall cost is high
- The large size of PCB is required for assembling all components
- The physical size of the product is big
- Overall product design requires more time
- A discrete component is used, the system is not reliable
- Most of the microprocessor does not support floating point operations
- The processor has a limitation on the size of data
- This processor should not contact with the other external devices
- The microprocessor does not have any internal peripheral like ROM, RAM and other I/O devices

3.8 APPLICATIONS OF A MICROPROCESSOR

- We can use a microprocessor in our personal computers
- These can be used in laser printers
- They are also used in television and mobile phone
- it is used for various military applications
- It is also be used in various game machines and calculators

3.9 REFERENCES

1. https://www.javatpoint.com/microprocessor-introduction
2. https://data-flair.training/blogs/basics-of-microprocessor-evolution-types-applications-working/
3. https://www.ecstuff4u.com/2019/05/advantages-and-disadvantages-of-microprocessor.html
4. watelectronics.com/what-is-a-microprocessor-architecture-types-its-applications/

CHAPTER 4

8086 INTERNAL ARCHITECTURE

Ms. Roop Shikha[1] Dr. Shashi Jawla[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Associate Professor, Swami Vivekanand Institute of Engineering & Technology*

8086 Microprocessor is an enhanced version of 8085Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20 address lines and16 data lines that provides up to 1MB storage. It consists of powerful instruction set, which provides operations like multiplication and division easily.

It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor.

4.1 FEATURES OF 8086 MICROPROCESSOR

1. 8086 includes an Instruction Queue. While an instruction is getting executed, it can store 6 instructions bytes at most at a time. It improves the performance.
2. The 8086 microprocessor has registers, internal and external buses and ALU, all of 16-bit. As a result it performs faster operations.
3. Microprocessor 8086 can simultaneously execute an instruction and fetch the next one.
4. The 8086 is a 16-bit microprocessor. The term "16-bit" means that its arithmetic logic unit, internal registers and most of its instructions are designed to work with 16-bit binary words.
5. The 8086 has a 16-bit data bus, so it can read data from or write data to memory and ports either 16 bits or 8 bits at a time. The 8088, however, has an 8-bit data bus, soil can only read data from or write data to memory and ports 8 bits at a time.
6. The 8086 has a 20-bit address bus, so it can directly access 220 or 10,48,576 (1Mb) memory locations. Each of the 10, 48, 576 memory locations is byte Therefore, a sixteen-bit words are stored in two consecutive memory locations. The 8088 also has a 20-bit address bus, so it can also address 220 or 10, 48, 576 memory locations.
7. The Features of 8086 Microprocessor can generate 16-bit I/O address, hence it can access 216 = 65536 I/O ports.
8. The 8086 provides fourteen 16-bit registers.
9. The 8086 has multiplexed address and data bus which reduces the number of pins needed, but does slow down the transfer of data (drawback).
10. The 8086 requires one phase clock with a 33% duty cycle to provide optimized internal timing.

11. The Features of 8086 Microprocessor is possible to perform bit, byte, word and block operations in 8086. It performs the arithmetic and logical operations on bit, byte, word and decimal numbers including multiply and divide.
12. The Intel 8086 is designed to operate in two modes, namely the minimum mode and the maximum mode. When only one 8086 CPU is to be used in a microcomputer system, the 8086 is used in the minimum mode of operation. In this mode the CPU issues the control signals required by memory and I/O In multiprocessor (more than one processor in the system) system 8086 operates in maximum mode. In maximum mode, control signals are generated with the help of external bus controller (8288).
13. The Intel 8086 supports multiprogramming. In multiprogramming, the code for two or more processes is in memory at the same time and is executed in a time-multiplexed fashion.
14. An interesting feature of the 8086 is that it fetches up to six instruction bytes (4 instruction bytes for 8088) from memory and queue stores them in order to speed up instruction execution.
15. The Features of 8086 Microprocessor provides powerful instruction set with the following addressing modes: Register, immediate, direct, indirect through an index or base, indirect through the sum of a base and an index register, relative and implied.
16. Intel 8086 microprocessor has two units; Execution Unit (EU) and Bus Interface Unit (BIU). These 2 units depend on each other to function.

4.2 INTERNAL ARCHITECTURE OF 8086

Fig. 4.1 shows a block diagram of Internal Architecture of 8086. It is internally divided into two separate functional units. These are the **Bus Interface Unit (BIU)** and the **Execution Unit (EU)**. These two functional units can work simultaneously to increase system speed and hence the throughput. Throughput is a measure of number of instructions executed per unit time.

Bus Interface Unit:

The bus interface unit is the Internal Architecture of 8086 to the outside world. It provides a full 16-bit bi-directional data bus and 20-bit address bus. The bus interface unit is responsible for performing all external bus operations, as listed below:

Functions of Bus Interface Unit:

1. It sends address of the memory or I/O.
2. It fetches instruction from memory.
3. It reads data from port/memory.
4. It Writes data into port/memory.
5. It supports instruction queuing.

6. It provides the address relocation facility.

To implement these functions the BIU contains the instruction queue, segment registers instruction pointer, address summer and bus control logic.

Instruction Queue:

To speed up program execution, the BIU fetches **six instruction bytes** ahead of time from the memory. These prefetched instruction bytes are held for the execution unit in a group of registers called **Queue**.

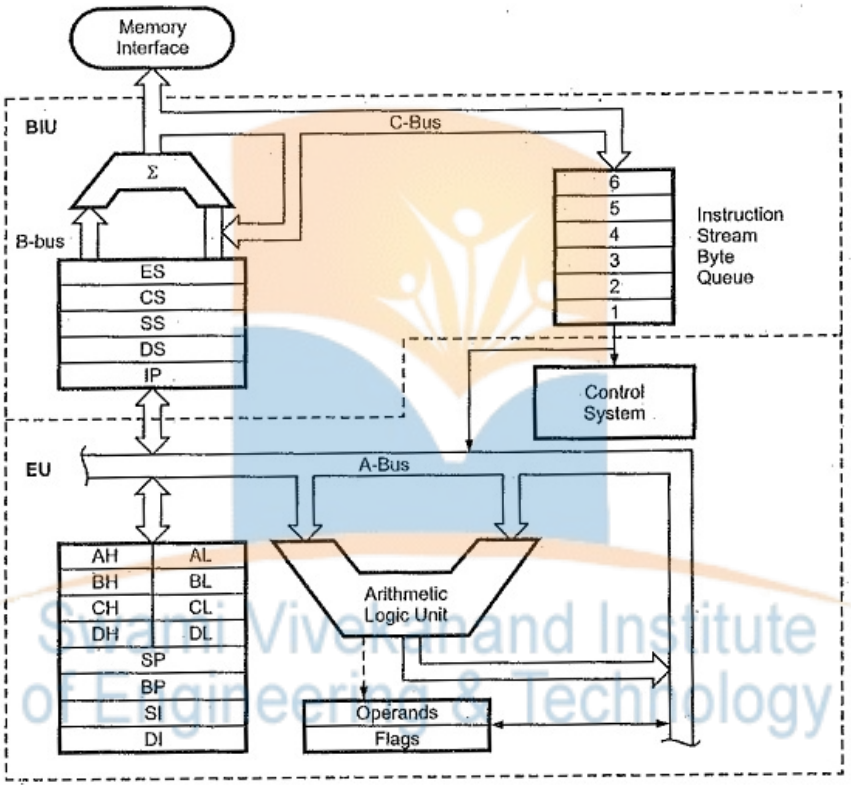


Fig. 4.1 8086 Internal block diagram

With the help of queue it is possible to fetch next instruction when current instruction is in execution. For example, current instruction in execution is a multiplication instruction. In Internal Architecture of 8086, operands for multiplication operations are within registers. Still it requires 100 clock cycles to execute multiply instruction. Like multiplication there are number of other instructions in 3086 which need a quite a large number- of clock cycles for execution. During this execution time the BIU fetches the next instruction or instructions from memory into the instruction queue instead of remaining idle. The BIU continues this process as long as the

queue is not full. Due to this, execution unit gets the ready instruction in the queue and instruction fetch time is eliminated. This is illustrated in Fig. 4.2.

The queue operates on the principle first in first out (FIFO). So that the execution unit gets the instructions for execution in the order they are fetched. In case of JUMP and CALL instructions, instruction already fetched in queue is of no use. Hence, in these cases queue is dumped and newly formed by loading instructions from new address specified by JUMP or CALL instruction. Feature of fetching the next instruction while the current instruction is executing is called **pipelining**.

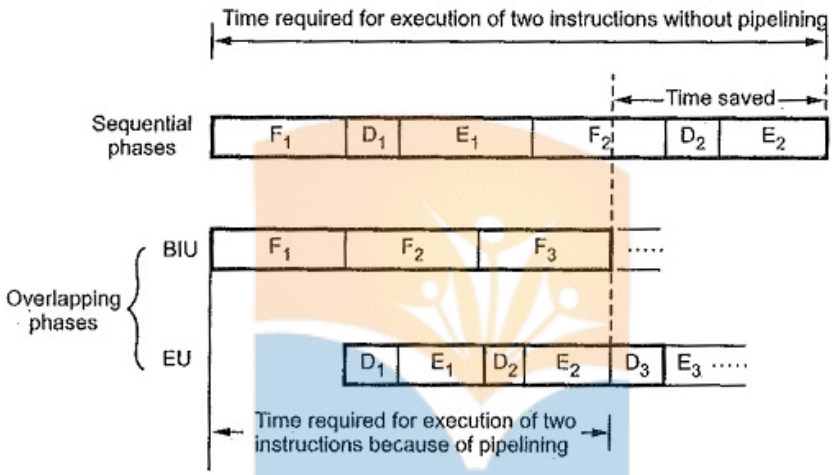


Fig. 4.2 Pipelining

Segment Registers:

The physical address of the Internal Architecture of 8086 is 20-bits wide to access 1 Mbyte memory locations. However, its registers and memory locations which contain logical addresses are just 16-bits wide. Hence 8086 uses memory segmentation. It treats the 1 Mbyte of memory as divided into segments, with a maximum size of a segment as 64 Kbytes. Thus any location within the segment can be accessed using 16 bits. The Internal Architecture of 8086 allows only four active segments at a time, as shown in the Fig. 4.3. For the selection of the four active segments the 16-bit segment registers are provided within the BIU of the 8086. These four registers are:

Code segment (CS) register, the data segment (DS) register, the stack segment (SS) register, and the extra segment (ES) register. These are used to hold the upper 16-bits of the starting addresses of the four memory segments, on which 8086 works at a particular time. For example, the value in CS identifies the starting address of 64 K-byte segment known as code segment. By "**starting address**", we mean the lowest addressed byte in the active code segment. The starting address is also known as **base address** or **segment base**.

The BIT) always inserts zeros for the lowest 4 bits (nibble) in the contents of segment register to generate 20-bit base address. For example, if the code segment register contains 348AH, then code segment will start at address 348A0H.

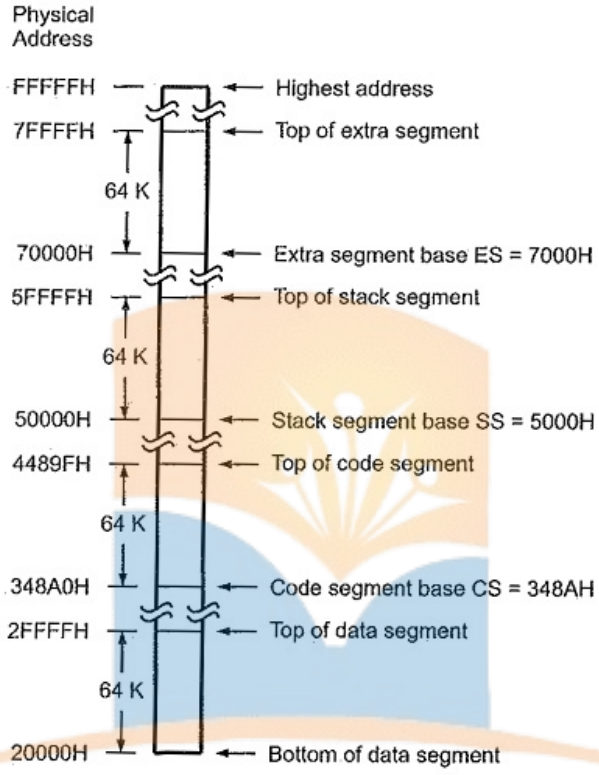


Fig. 4.3 Memory segmentation

Functions of Segment Registers:

1. The CS register holds the upper 16-bits of the starting address of the segment from which the BIU is currently fetching the instruction code byte.
2. The SS register is used for the upper 16-bits of the starting address for the program stack (all stack related instructions will operate on stack)
3. ES register and DS register are used to hold the upper 16-bits of the starting address of the two memory segments which are used for data.

Rules for Memory Segmentation:

1. The four segments can overlap for small programs. In a minimum system all four segments can start at the address 00000H.
2. The segment can begin/start at any memory address which is divisible by 16.

Advantages of Memory Segmentation:

1. It allows the memory addressing capacity to be 1 Mbyte even though the address associated with individual instruction is only 16-bit.
2. It allows instruction code, data, stack, and portion of program to be more than 64 KB long by using more than one code, data, stack segment, and extra
3. It facilitates use of separate memory areas for program, data and stack.
4. It permits a program or its data to be put in different areas of memory, each time the program is executed i.e. program can be relocated which is very useful in multiprogramming.

Instruction Pointer:

The instruction pointer register holds the 16-bit address of the next Code byte within the code segment. The value contained in the IP is referred to as an **offset**. This value must be offset from (added to) the segment base address in CS to produce the required 20-bit physical address.

Generation of 20-bit Address:

The contents of the CS register are multiplied by 16 i.e. shifted by 4 position to the left by inserting 4 zero bits and then the offset i.e. the contents of IP register are added to the shifted contents of CS to generate physical address. As shown in the Fig 4.4, the contents of CS register are 348AH; therefore the shifted contents of CS register are 348A0H. When the BIU adds the offset of 4214H in the IP to this starting address, the result is 20-bit physical of 38AB4H.

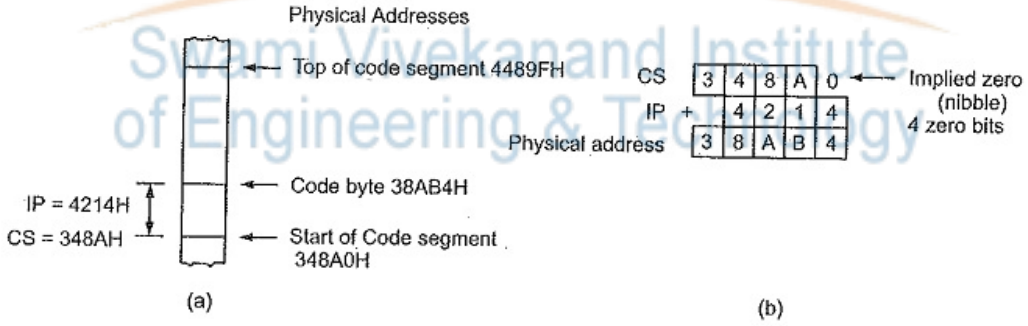


Fig. 4.4 Generation of 20-bit address

Execution Unit [EU]:

The execution unit of Internal Architecture of 8086 tells the BIU from where to fetch instructions or data, decodes instructions and executes instructions. It contains

- Control Circuitry

- Instruction Decoder
- Arithmetic Logic Unit (ALU)
- Flag Register
- General Purpose Registers

- Pointers and Index Registers

Control Circuitry, Instruction Decoder, ALU:

The control circuitry in the EU directs the internal operations. A decoder in the EU translates the instructions fetched from memory into a series of actions wlifeh the EU performs. ALU is 16-bit. It can add, subtract, AND, OR, XOR, increment, decrements, complement and shift binary numbers.

Flag Register:

A flag is a flip–flop which indicates some condition produced by the execution of an instruction or controls certain operations of the EU. The flag register contains nine active flags as shown in the Fig. 4.5.

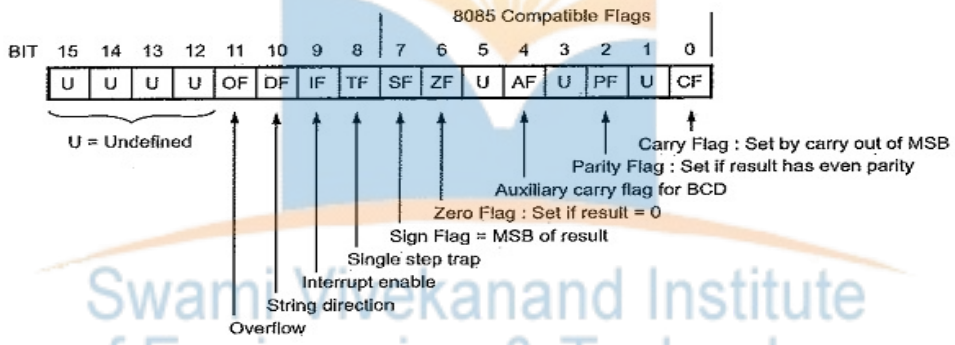


Fig. 4.5 Flag register

General Purpose Registers:

The EU has. 8 general purposes registers labeled ATI, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually for temporary storage of 8 bit data. The AL register is also called accumulator. Certain pairs of these general purpose registers can be used together to store 16-bit data, such as AX, BX, CX and DX.

Pointers and Index Registers:

All segment registers are 16-bit. But it is necessary to put 20-bit address (physical address) on the address bus. To get 20-bit physical address one more register is associated with each segment register the way IP is associated with CS.

These additional registers belong to the pointer and index group. The pointer and index group consists of instruction pointer (IP), stack pointer (SP), BP (base pointer), source index (SI) and destination index (DI) registers.

Stack Pointer (SP): The stack pointer (SP) register contains the 16-bit offset from the start of the segment to the top of stack. For stack operation, physical address is produced by adding the contents of stack pointer register to the segment base address in SS. To do this the contents of the stack segment register are shifted four bits left and the contents of SP are added to the shifted result. If the contents of SP are 9F20H and SS are 4000H then the physical address is calculated as follows. (Refer Fig. 4.6)

SS = 4000H after shifting four bits left SS = 40000H

Now

SS	40000H
+ SP	9F20H
Physical address	49F20H

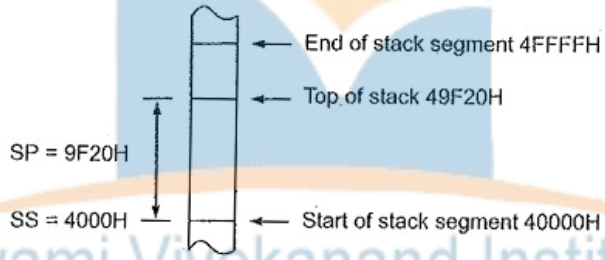


Fig. 4.6 Stack and stack pointer

Base Pointer, Source Index and Destination Index (BP, SI and DI)

These three 16-bit registers can be used as general purpose registers. However, their main use is to hold the 16-bit offset of the data word in one of the segments.

Base pointer: We can use the BP register instead of SP for accessing the stack using the based addressing Mode. In this case, the 20-bit physical stack address is calculated from BP and SS.

Source Index: Source index (SI) can be used to hold the offset of a data word in the data segment.

Destination Index: The ES register points to the extra segment in which data is stored. String instructions always use ES and DI to determine the 20-bit physical address for the destination.

Default and Alternate Register Assignments:

Table 4.1 shows that some memory references and their default and alternate segment definitions. For example, instruction codes can only be stored in the code segment with IP used as an offset. Similarly, for stack operations only SS and SP or BP registers can be used to give segment and offset addresses respectively. On the other hand, for accessing general data, string source; data pointed by BX and BP registers; it is possible to use alternate segments by using segment override prefix. See examples given in Table 4.1.

Table 4.1 Default and Alternate Register Assignments

Type of Memory Reference	Default Segment	Alternate Segment	Offset (Logical Address)
Instruction fetch	CS	None	IP
Stack operation	SS	None	SP, BP
General data	DS	CS, ES, SS	Effective address
String source	DS	CS, ES, SS	SI
String destination	ES	None	DI
BX used as pointer	DS	CS, ES, SS	Effective Address
BP used as pointer	SS	CS, ES, DS	Effective Address

4.3 REFERENCES

1. https://www.eeeguide.com/internal-architecture-of-8086/
2. https://www.tutorialspoint.com/microprocessor/microprocessor_8086_overview.htm
3. https://onlineclassnotes.com/draw-internal-architecture-of-8086/

CHAPTER 5

MICROPROCESSOR - 8086 ADDRESSING MODES

Ms. Neha Garg[1] Ms. Sujata Tondon[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

The different ways in which a source operand is denoted in an instruction is known as **addressing modes**. This specifies that the given data is an immediate data or an address. It also specifies whether the given operand is register or register pair.

5.1 TYPES OF ADDRESSING MODES:

The 8086 microprocessors have 8 addressing modes. Two addressing modes have been provided for instructions which operate on register or immediate data.

These two addressing modes are:

1. **Register Addressing mode**

In register addressing, the operand is placed in one of the 16-bit or 8-bit general purpose registers.

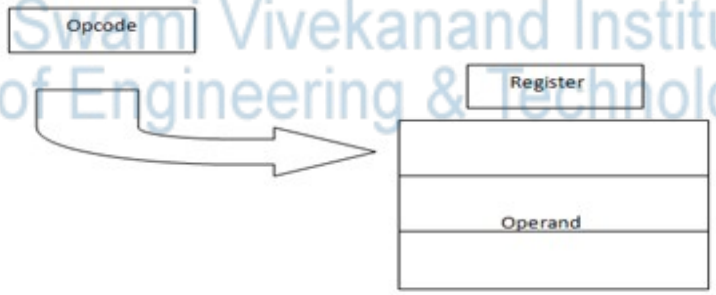


Fig. 5.1 Register addressing mode

Example

- MOV AX, CX
- ADD AL, BL

- ADD CX, DX

2. Immediate Addressing mode

In immediate addressing, the operand is specified in the instruction itself.

Example

- MOV AL, 35H
- MOV BX, 0301H
- MOV [0401], 3598H
- ADD AX, 4836H

The remaining 6 addressing modes specify the location of an operand which is placed in a memory.

These 6 addressing modes are:

3. Direct Addressing

In direct addressing mode, the operand offset is given in the instruction as an 8-bit or 16-bit displacement element.

In Absolute/ Direct Addressing Mode the effective address of memory location where operand is present is written directly in the instruction.

Example: Mov Ax, [5000H]

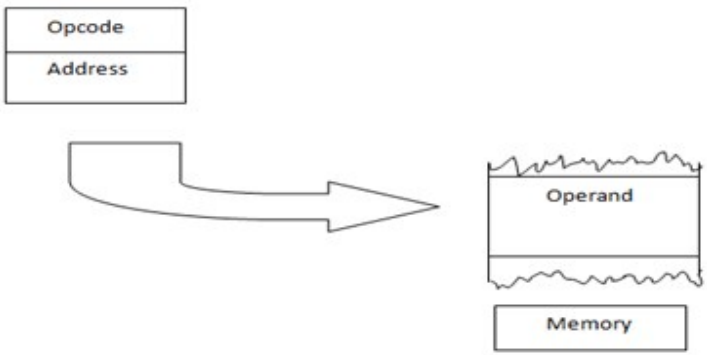


Fig. 5.2 Direct addressing mode

4. Register Indirect Addressing

The operand's offset is placed in any one of the registers BX, BP, SI or DI as specified in the instruction.

Example

- MOV AX, [BX]

It moves the contents of memory locations addressed by the register BX to the register AX.

5. Based Addressing mode

The operand's offset is the sum of an 8-bit or 16-bit displacement and the contents of the base register BX or BP. BX is used as base register for data segment, and the BP is used as a base register for stack segment.

Effective address (Offset) = [BX + 8-bit or 16-bit displacement].

Fig. 5.3 Based addressing mode

Example

- MOV AL, [BX+05]; an example of 8-bit displacement.
- MOV AL, [BX + 1346H]; example of 16-bit displacement.

6. Indexed Addressing

The offset of an operand is the sum of the content of an index register SI or DI and an 8-bit or 16-bit displacement.

Offset (Effective Address) = [SI or DI + 8-bit or 16-bit displacement]

Example

- MOV AX, [SI + 05]; 8-bit displacement.
- MOV AX, [SI + 1528H]; 16-bit displacement.

7. Based Indexed Addressing

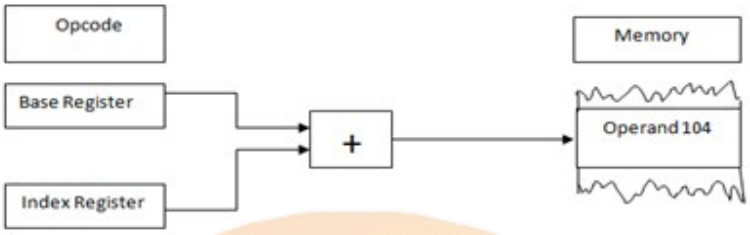


Fig. 5.4 Based Indexed Addressing

The effective address of data is formed by adding content of base register any one of Bx or Bp to the content of index register any one of SI or DI. The default segment may be ES or DS. Example: ADD CX, [AX+SI],

MOV AX, [AX+DI]

Here Bx is the base register and SI is the index register.

8. Based Indexed with Displacement

In this mode of addressing, the operand's offset is given by:

Effective Address (Offset) = [BX or BP] + [SI or DI] + 8-bit or 16-bit displacement

Example

- MOV AX, [BX + SI + 05]; 8-bit displacement
- MOV AX, [BX + SI + 1235H]; 16-bit displacement

5.2 REFERENCES

1. https://www.geeksforgeeks.org/addressing-modes-8086-microprocessor/

2. https://www.tutorialspoint.com/microprocessor/microprocessor_8086_addressing_modes.htm

3. https://www.javatpoint.com/8086-microprocessor

4. https://medium.com/@rashika1985/addressing-modes-8086-microprocessor-fc62cb962254

CHAPTER 6

DATA TRANSFER AND ARITHMETIC INSTRUCTIONS OF 8086

Ms. Ritika Mishra[1] Mr. Vikas Zandu[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

We know that instructions are the binary commands used for the execution of any operation. Instructions are classified on the basis of functions they perform.

6.1 INSTRUCTION SET OF 8086 MICROPROCESSOR

The instruction set in 8086 microprocessor is classified as follows:

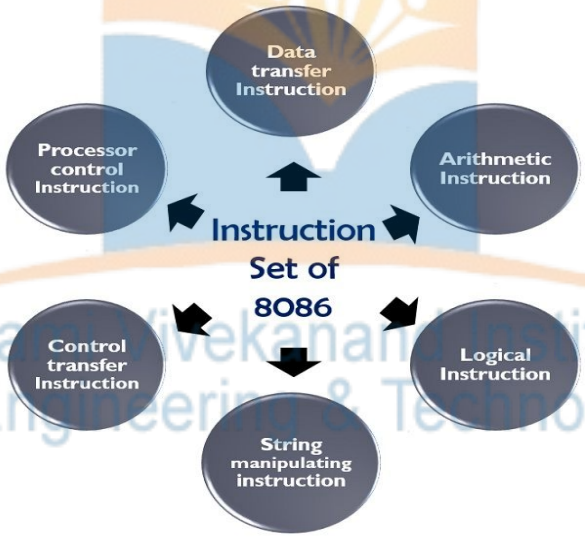


Fig. 6.1 Instruction set of 8086 microprocessor

In this chapter, we will discuss the data transfer and arithmetic instructions in detail.

6.2 DATA TRANSFER INSTRUCTION

This group includes the instructions used for moving the data from one place to another. In data transfer group of instructions data or address can be transferred to either register, memory or I/O ports.

In order to accomplish any transfer, the source and destination must be known. So, these instructions involve two operands i.e., the source and the destination. The size of the operand must be the same.

The source holds either the immediate data, register or memory location. While the destination holds either the address of any memory location or register.

6.3 TYPES OF DATA TRANSFER INSTRUCTIONS

1. Move instructions:

These instructions are used to move data from one memory location to another or between a memory location and a register. They include the following instructions:

- MOV: Moves data from a source operand to a destination operand.
- XCHG: Swaps the contents of two operands.
- XLAT: Translates a byte in memory using a lookup table pointed to by the contents of the AL register.
- LEA: Loads a 16-bit offset address into a register.

2. Load instructions:

These instructions are used to load data from a memory location or I/O device into a register. They include the following instructions:

- LDS: Loads a 16-bit pointer value from a memory location into a register pair and loads the 8-bit value from the next memory location into another register.
- LSS: Loads a 16-bit pointer value from a memory location into a register pair and loads the 16-bit value from the next memory location into another register.
- LXI: Loads a 16-bit value into a register pair.
- MOV with memory operand: Loads data from a memory location into a register.

3. Store instructions:

These instructions are used to store data from a register into a memory location or I/O device. They include the following instructions:

- MOV with memory operand: Stores data from a register into a memory location.
- STA: Stores the contents of the accumulator register (AL or AX) in memory.
- STAX: Stores the contents of a register pair (BC, DE, or HL) in memory using either the indirect addressing mode or the direct addressing mode.

- SHLD: Stores a 16-bit data word from registers H and L in memory using the direct addressing mode.
- PUSH: Stores the contents of a register onto the stack.

4. Input/Output instructions:

These instructions are used to communicate with external input/output (I/O) devices. They include the following instructions:

- IN: Reads a byte or word of data from an I/O port into a register.
- OUT: Writes a byte or word of data from a register to an I/O port.
- INS: Reads a block of data from an I/O port into a memory location.
- OUTS: Writes a block of data from a memory location to an I/O port.

5. String instructions:

These instructions are used for manipulating strings of data, such as moving, copying, or comparing strings. They operate on consecutive bytes or words in memory, and can be used for fast and efficient string processing. Some examples of string instructions include:

- MOVS: Moves a byte or word from a source location to a destination location, and updates the index registers to point to the next byte or word.
- CMPS: Compares a byte or word in memory to a byte or word in a register, and updates the index registers accordingly.
- LODS: Loads a byte or word from a memory location into a register, and updates the index registers to point to the next byte or word.
- STOS: Stores a byte or word from a register into a memory location, and updates the index registers to point to the next byte or word.

6.4 ARITHMETIC INSTRUCTIONS

These instructions are used in order to execute arithmetic instructions like addition, subtraction, multiplication, division, increment or decrement.

The flags of the 8086 microprocessor are altered when arithmetic and logical instructions are executed. Basically, the status of the result of the operations is reflected by the flag.

ADD:

The add instruction adds the contents of the source operand to the destination operand. Eg. ADD AX, 0100H

ADD AX, BX

ADD AX, [SI]

ADD AX, [5000H]

ADD [5000H], 0100H

ADD 0100H

ADC: Add with Carry

This instruction performs the same operation as ADD instruction, but adds the carry flag to the result.
Eg. ADC 0100H

ADC AX, BX

ADC AX, [SI]

ADC AX, [5000]

ADC [5000], 0100H

SUB: Subtract

The subtract instruction subtracts the source operand from the destination operand and the result is left in the destination operand.

Eg. SUB AX, 0100H

SUB AX, BX

SUB AX, [5000H]

SUB [5000H], 0100H

SBB: Subtract with Borrow

The subtract with borrow instruction subtracts the source operand and the borrow flag (CF) which may reflect the result of the previous calculations, from the destination

operand
Eg. SBB AX, 0100H

SBB AX, BX

SBB AX, [5000H]

SBB [5000H], 0100H

INC: Increment

This instruction increases the contents of the specified Register or memory location by 1. Immediate data cannot be operand of this instruction.

Eg. INC AX

INC [BX]

INC [5000H]

DEC: Decrement

The decrement instruction subtracts 1 from the contents of the specified register or memory location.

Eg. DEC AX

DEC [5000H]

NEG: Negate

The negate instruction forms 2's complement of the specified destination in the instruction. The destination can be a register or a memory location. This instruction can be implemented by inverting each bit and adding 1 to it.

Eg. NEG AL

AL = 0011 0101 35H Replace number in AL with its 2's complement

AL = 1100 1011 = CBH

CMP: Compare

This instruction compares the source operand, which may be a register or an immediate data or a memory location, with a destination operand that may be a register or a memory location

Eg. CMP BX, 0100H

CMP AX, 0100H

CMP [5000H], 0100H

CMP BX, [SI]

CMP BX, CX

MUL: Unsigned Multiplication Byte or Word

This instruction multiplies an unsigned byte or word by the contents of AL.

Eg.

MUL BH ; (AX) (AL) x (BH)

MUL CX ; (DX)(AX) (AX) x (CX)

MUL WORD PTR [SI] ; (DX)(AX) (AX) x ([SI])

IMUL: Signed Multiplication

This instruction multiplies a signed byte in source operand by a signed byte in AL or a signed word in source operand by a signed word in AX.

Eg. IMUL BH

IMUL CX

IMUL [SI]

CBW: Convert Signed Byte to Word

This instruction copies the sign of a byte in AL to all the bits in AH. AH is then said to be sign extension of AL.

Eg. CBW

AX= 0000 0000 1001 1000 Convert signed byte in AL signed word in AX.

Result in AX = 1111 1111 1001 1000

CWD: Convert Signed Word to Double Word

This instruction copies the sign of a byte in AL to all the bits in AH. AH is then said to be sign extension of AL.

Eg. CWD

Convert signed word in AX to signed double word in DX: AX

DX= 1111 1111 1111 1111

Result in AX = 1111 0000 1100 0001

DIV: Unsigned division

This instruction is used to divide an unsigned word by a byte or to divide an unsigned double word by a word.

Eg.

DIV CL ; Word in AX / byte in CL

; Quotient in AL, remainder in AH

DIV CX ; Double word in DX and AX / word

; in CX, and Quotient in AX,

; remainder in DX

AAA: ASCII Adjust After Addition

The AAA instruction is executed after an ADD instruction that adds two ASCII coded operand to give a byte of result in AL. The AAA instruction converts the resulting contents of Al to a unpacked decimal digits.

Eg.

```
ADD CL, DL ; [CL] = 32H = ASCII for 2
           ; [DL] = 35H = ASCII for 5
           ; Result [CL] = 67H
MOV AL, CL ; Move ASCII result into AL since
           ; AAA adjust only [AL]
AAA        ; [AL]=07, unpacked BCD for 7
```

AAS: ASCII Adjust AL after Subtraction

This instruction corrects the result in AL register after subtracting two unpacked ASCII operands. The result is in unpacked decimal format. The procedure is similar to AAA instruction except for the subtraction of 06 from AL.

AAM: ASCII Adjust after Multiplication

This instruction, after execution, converts the product available In AL into unpacked BCD format.

Eg.

```
MOV AL, 04 ; AL = 04
MOV BL ,09 ; BL = 09
MUL BL     ; AX = AL*BL ; AX=24H
AAM        ; AH = 03, AL=06
```

AAD: ASCII Adjust before Division

This instruction converts two unpacked BCD digits in AH and AL to the equivalent binary number in AL. This adjustment must be made before dividing the two unpacked BCD digits in AX by an unpacked BCD byte. In the instruction sequence, this instruction appears Before DIV instruction.

Eg.
AX 05 08

AAD result in AL 00 3A ,since 58D = 3AH in AL

The result of AAD execution will give the hexadecimal number 3A in AL and 00 in AH. Where 3A is the hexadecimal Equivalent of 58 (decimal).

DAA: Decimal Adjust Accumulator

This instruction is used to convert the result of the addition of two packed BCD numbers to a valid BCD number. The result has to be only in AL.

Eg.
AL = 53 CL = 29

ADD AL, CL ; AL <– (AL) + (CL)

; AL 53 + 29

; AL 7C

DAA; AL 7C + 06 (as C>9)

; AL 82

DAS: Decimal Adjust after Subtraction

This instruction converts the result of the subtraction of two packed BCD numbers to a valid BCD number. The subtraction has to be in AL only.

Eg.
AL = 75, BH = 46

SUB AL, BH ; AL 2F = (AL) – (BH)

; AF = 1

DAS ; AL 29 (as F>9, F – 6 = 9)

6.4 REFERENCES

1. https://www.javatpoint.com/instruction-set-of-8086
2. https://www.tutorialspoint.com/microprocessor/microprocessor_8086_instruction_sets.htm
3. https://electronicsdesk.com/instruction-set-of-8086-microprocessor.html
4. https://www.geeksforgeeks.org/data-transfer-instructions-8086-microprocessor/
5. https://8086up.wordpress.com/2014/03/06/arithmetic-instructions/

CHAPTER 7

LOGICAL, STRING MANIPULATION, CONTROL TRANSFER AND PROCESSOR CONTROL INSTRUCTIONS

Ms. Kulbir Kaur[1] Ms. Saneha[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

7.1 LOGICAL INSTRUCTIONS

These instructions perform operations like AND, OR, complement, shift and rotate on the binary data. The outcome of the operation executed by logical instructions is represented as the status of flag register.

Let us see the logical instructions of 8086 microprocessor. Here the D, S and C are destination and source and count respectively. D, S and C can be either register, data or memory address.

Opcode	Operand	Description
AND	D,S	Used for adding each bit in a byte/word with the corresponding bit in another byte/word.
OR	D,S	Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.
NOT	D	Used to invert each bit of a byte or word.
XOR	D,S	Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.
TEST	D,S	Used to add operands to update flags, without affecting operands.
SHR	D,C	Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.
SHL/SAL	D,C	Used to shift bits of a byte/word towards left and put zero(S) in LSBs.

ROR	D,C	Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].
ROL	D,C	Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].
RCR	D,C	Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.
RCL	D,C	Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

7.2 STRING MANIPULATION INSTRUCTION

A sequence of bytes or words forms a string. This instruction set contains instructions for movement, comparison, scanning, loading or storing of the string.

The following instructions come under this category:

Instruction	**Description**
MOVS/MOVSB/MOVSW	Moves 8-bit or 16-bit data from the memory location(s) addressed by SI register to the memory location addressed by DI register.
CMPS/CMPSB/CMPSW	Compares the content of memory location addressed by DI register with the content of memory location addressed by SI register.
SCAS/SCASB/SCASW	Compares the content of accumulator with the content of memory location addressed by DI register in the extra segment ES.
LODS/LODSB/LODSW	Loads 8-bit or 16-bit data from memory location addressed by SI register into AL or AX register.
STOS/STOSB/STOSW	Stores 8-bit or 16-bit data from AL or AX register in the memory location addressed by DI register.
REP	Repeats the given instruction until CX ≠ 0

REPE/ REPZ	Repeats the given instruction till CX ≠ 0 and ZF = 1
REPNE/REPNZ	Repeats the given instruction till CX ≠ 0 and ZF = 0

7.3 CONTROL TRANSFER INSTRUCTION

This set consists of instructions like call, jump, loop and software interrupt instructions that basically controls the operation of the processor. These instructions do not modify the status of flag registers.

The following instructions come under this category:

Instruction	**Description**
JA or JNBE	Jump if above, not below, or equal i.e. when CF and ZF = 0
JAE/JNB/JNC	Jump if above, not below, equal or no carry i.e. when CF = 0
JB/JNAE/JC	Jump if below, not above, equal or carry i.e. when CF = 0
JBE/JNA	Jump if below, not above, or equal i.e. when CF and ZF = 1
JCXZ	Jump if CX register = 0
JE/JZ	Jump if zero or equal i.e. when ZF = 1
JG/JNLE	Jump if greater, not less or equal i.e. when ZF = 0 and CF = OF
JGE/JNL	Jump if greater, not less or equal i.e. when SF = OF
JL/JNGE	Jump if less, not greater than or equal i.e. when SF ≠ OF
JLE/JNG	Jump if less, equal or not greater i.e. when ZF = 1 and SF ≠ OF
JMP	Causes the program execution to jump unconditionally to the memory address or label given in the instruction.
CALL	Calls a procedure whose address is given in the instruction and saves their return address to the stack.
RET	Returns program execution from a procedure (subroutine) to the next instruction or main program.

IRET	Returns program execution from an interrupt service procedure (subroutine) to the main program.
INT	Used to generate software interrupt at the desired point in a program.
INTO	Software interrupts to indicate overflow after arithmetic operation.
LOOP	Jump to defined label until CX = 0.
LOOPZ/LOOPE	Decrement CX register and jump if CX ≠ 0 and ZF = 1.
LOOPNZ/LOOPNE	Decrement CX register and jump if CX ≠ 0 and ZF = 0.

Here, CF = Carry Flag

ZF = Zero Flag

OF = Overflow Flag

SF = Sign Flag

CX = Register

7.4 PROCESSOR CONTROL INSTRUCTION

This set includes instructions for setting or clearing the carry, direction and interrupt flags inside the processor.

Process control instructions are the instructions which control the processor's action by setting (1) or resetting (0) the values of flag registers.

Following is the table showing the list of process control instructions:

OPCODE	**OPERAND**	**EXPLANATION**	**EXAMPLE**
STC	None	sets carry flag to 1	STC
CLC	None	resets carry flag to 0	CLC
CMC	None	compliments the carry flag	CMC
STD	None	sets directional flag to 1	STD

CLD	None	resets directional flag to 0	CLD
STI	None	sets the interrupt flag to 1	STI
CLI	None	resets the interrupt flag to 0	CLI

7.5 REFERENCES

1. https://www.javatpoint.com/instruction-set-of-8086
2. https://www.tutorialspoint.com/microprocessor/microprocessor_8086_instruction_sets.htm
3. https://electronicsdesk.com/instruction-set-of-8086-microprocessor.html
4. https://www.geeksforgeeks.org/data-transfer-instructions-8086-microprocessor/
5. https://8086up.wordpress.com/2014/03/06/arithmetic-instructions/

CHAPTER 8

ASSEMBLER DIRECTIVES OF THE 8086 MICROPROCESSOR

Ms. Komal Sood[1] Mr. Rajat Gupta[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

8.1 INTRODUCTION

Assembly languages are low-level languages for programming computers, microprocessors, microcontrollers, and other IC. They implement a symbolic representation of the numeric machine Codes and other constants needed to program a particular CPU architecture. This representation is usually defined by the hardware manufacturer, and is based on abbreviations that help the programmer to remember individual instructions, registers. An assembler directive is a statement to give direction to the assembler to perform task of the assembly process.

It control the organization if the program and provide necessary information to the assembler to understand the assembly language programs to generate necessary machine codes. They indicate how an operand or a section of the program is to be processed by the assembler.

An assembler supports directives to define data, to organise segments to control procedure, to define macros. It consists of two types of statements: instructions and directives. The instructions are translated to the machine code by the assembler whereas directives are not translated to the machine codes.

8.2 WHAT IS AN ASSEMBLER?

We know that assembly language is a less complex and programmer-friendly language used to program the processors. In assembly language programming, the instructions are specified in the form of mnemonics rather in the form of machine code i.e., 0 and 1. But the microprocessor or microcontrollers are specifically designed in a way that they can only understand machine language.

Thus assembler is used to convert assembly language into machine code so that it can be understood and executed by the processor. *Therefore, to control the generation of machine codes from the assembly language, assembler directives are used*. However, machine codes are only generated for the program that must be provided to the processor and not for assembler directives because they do not belong to the actual program.

8.3 ASSEMBLER DIRECTIVES

Assembler directives help the assembler to correctly understand the assembly language programs to prepare the codes. Another type of hint which helps the assembler to assign a particular constant with a label or initialize particular memory locations or labels with constants is called an operator. Rather, the operators perform the arithmetic and logical tasks unlike directives that just direct the assembler to correctly interpret the program to code it appropriately. The following directives are commonly used in the assembly language programming practice using Microsoft Macro Assembler (MASM) or Turbo Assembler (TASM).

8.4 ASSEMBLER DIRECTIVES OF 8085

The assembler directives given below are used by 8085 and 8086 assemblers:

DB: Define Byte

The DB directive is used to reserve byte or bytes of memory locations in the available memory. While preparing the EXE file, this directive directs the assembler to allocate the specified number of memory bytes to the said data type that may be a constant, variable, string, etc. Another option of this directive also initializes the reserved memory bytes with the ASCII codes of the characters specified as a string.

```
AREA DB 30H, 52H, 35H
```

Memory name AREA has three consecutive locations where 30H, 52H and 35H are to be stored.

DW: Define Word

It is used for initialising single or multiple data words (16-bit).

```
MARK DW 1020H, 4216H
```

These two 16-bit data 1020H and 4216H are stored at 4 consecutive locations in the memory MARK.

END: *End of program*
This directive is used at the time of program termination.

EQU: *Equate*
It is used to assign any numerical value or constant to the variable.

```
DONE EQU 10H
```

Variable name 'DONE' has value 10H

MACRO: *Represents beginning*
Shows the beginning of macro along with defining name and parameters.

ENDM: *End of macro*
ENDM indicates the termination of macro.

```
STEP  MACRO [x1, x2, x3]
      -----
      -----
      -----   } statements
      -----
STEP  ENDM
```


where macroname (STEP) is specified by the user.

ORG: *Origin*
This directive is used at the time of assigning starting address for a module or segment.

```
ORG 1050H
```

By this instruction, the assembler gets to know that the statements following this instruction, must be stored in the memory location beginning with address 1050H.

8.5 ASSEMBLER DIRECTIVES OF 8086

These assembler directives are specifically used by 8086:

ASSUME: *Shows the segment name to the assembler*
It provides information to the assembler regarding the name of the program or data segment for that particular segment.

```
ASSUME CS : _DONE
```

This directive specifies that the instruction of the source program is stored in logical segment _DONE.

DD: *Define Double word*
This directive allows the initialization of single or multiple data in the form of double words (i.e., 4 bytes). The is used to inform the assembler that the stored data in memory is a double word.

```
MEM DD 10D50F7B
```

Thus memory stores the given data in the form:

MEM	
	7BH
	0FH
	D5H
	10H
	00H
	00H
	00H
	00H

DQ: *Define Quad words*
It is used to initialise quad words (8-bytes) either one or more than one. Thereby informing the assembler that the data stored in memory is quad-word.

DT: *Define ten bytes*
It is used to allocate and initialize 10 bytes of a variable.

DUP: *Duplicate*
DUP allows initialization of multiple locations and assigning of values to them. This allows storing of repeated characters or variables in different locations.

```
Book DB 8 DUP (10H, 20H, 30H, 40H)
```

So this permits the storing of these data in memory and creating 8 identical sets in the memory identified as Book.

Book	10H
	20H
	30H
	40H
	10H
	20H
	30H
	40H
	10H
	20H
	30H
	40H

DWORD: *Double word*
This directive is used to indicate that the operand is of double word size.

PROC: *Procedure*
It defines the starting of a procedure/subroutine.

FAR: This directive is a type specifier that is used by the assembler to declare intersegment call (i.e., call from different segment).
NEAR: This is used for intrasegment call i.e., a call within the same segment.
ENDP: *End of procedure*
This directive shows the termination of a procedure.

```
SUB32  PROC NEAR
       -----
       -----
       -----   statements
       -----
SUB32  ENDP
```


SEGMENT: *Beginning of a memory segment.*
It is used to show the beginning of a memory segment with a specific name.

ENDS: *End of segment*
This directive defines the termination of a particular memory segment as it is specified by its name.

```
DATA  SEGMENT
      ------
      ------
      ------   statements
      ------
DATA  ENDS
```

The statements within the segment are nothing but the program code.

EVEN: *It is used to inform the assembler to align the data beginning from an even address.*
As data specified with an odd starting address requires 2 byte accessing. Thus using this directive, data can be aligned with an even starting address.

PTR: *Pointer*
This directive shows information regarding the size of the operand.

```
JMP BYTE PTR [BX]
```

This shows legal near jump to BX.

PUBLIC: This directive is used to provide a declaration to variables that are common for different program modules.
STACK: This directive shows the presence of a stack segment.

```
STACK [size]
```

SHORT: This is used in reference to jump instruction in order to assign a displacement of one byte.
THIS: It is used along with EQU directive for setting the label to either, byte, word or double-word.
So, these assembler directives are used by the processors for controlling the generation of machine code and organization of the program.

8.6 REFERENCES

1. https://electronicsdesk.com/assembler-directives.html#:~:text=Assembler%20Directives%20of%208086,-These%20assembler%20directives&text=It%20provides%20information%20to%20the,stored%20in%20logical%20segment%20_DONE.

2. https://www.ukessays.com/essays/engineering/assembler-directive-of-8086-microprocessor.php

3. https://www.brainkart.com/article/Instruction-set-and-assembler-directives-of-8086-Microprocessor_7846/

CHAPTER 9

INTRODUCTION TO ASSEMBLY LANGUAGE PROGRAMMING

Ms. Kulbir Kaur[1] Mr. Hardeep Singh[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

9.1 PROGRAMMING LANGUAGES

To run a program, a microcomputer must have the program stored in binary form in successive memory locations. There are three language levels that can be used to write a program for a microcomputer.

1. Machine Language

2. Assembly Language

3. High-level Languages

1. Machine Language

You can write programs as simply a sequence of the binary codes for the instructions you want the microcomputer to execute. This binary form of the program is referred to as machine language because it is the form required by the machine. However, it is very difficult, not possible, for a programmer to memorize the thousands of binary instruction codes for a microprocessor. Also, it is very easy for an error to occur when working with long series of 1's and 0's. Using hexadecimal representation for the binary codes might help some, but there are still thousands of instruction codes to cope with.

2. Assembly Language

To make programming easier, many programmers write programs in assembly language. They then translate the assembly language program to machine language so that it can be loaded into memory and run.

Assembly language uses 2, 3, or 4- letter mnemonics to represent each instruction type. A mnemonic is advice to help you remember something. The letters in an assembly language mnemonic are usually initials or shortened form of the English word(s) for the operation performed by the instruction. For example, the mnemonic for addition is ADD, the mnemonic

for subtraction is SUB and the mnemonic for the instruction to copy data from one location to another is MOV. Assembly language statements are usually written in a standard form that has four fields, as shown in figure 9.1.

LABEL FIELD	OPCODE/MNEMONIC FIELD	OPERAND FIELD	COMMENT FIELD
NEXT:	ADD	AL,07H	;Add immediate number 07H to the contents of AL register

Fig. 9.1 Assembly Language statement format.

The first field in an assembly language statement is the Label field. A label is a symbol or group of symbols used to represent an address which is not specially known at the time the statement is written. Labels are usually followed by a colon.

The opcode field of the instruction contains the mnemonic for the instruction to be performed. Instruction mnemonics are sometimes called operation codes or opcodes.

The operand field of the statement contains the data, the memory address. The port address, or the name of the register on which the instruction is to be performed. Operand is just another name for the data item(s) acted on by the instruction. In the above example there are two operands, AL and 07H, specified in the operand field. AL represents the AL register, and 07H represents the number 07H. This assembly language statement thus says, "Add the number 07H to the contents of the AL register." By Intel convention, the result of the addition will be put in the register or the memory location specified before the comma in the operand field. For the example, the result will be left in the register AL.

The final field in an assembly language statement is comment field, which starts with a semicolon. Comments do not become the part of the machine language program, but they are very important.

3. High-level Language

Another way of writing a program for a microcomputer is with a high-level language, such as BASIC, Pascal, or C. These language use program statements which are even more English-like than those of assembly language. Each high level statement may represent many machine code instructions. An interpreter or a compiler program is used to translate higher-level language

statements to machine codes. Programs can usually be written faster in high level languages than in assembly language because a high –level language work with bigger building blocks. However, programs written in a high –level language and interpreted or compiled almost always execute more slowly and require more memory than the same program written in assembly language.

Programs that involve a lot of hardware control, such as robots and factory control systems, or programs that must run as quickly as possible are usually best written assembly language. Complex data processing programs that manipulate massive amounts of data, such as insurance company records, are usually best written in a high-level language.

9.2 PROGRAM DEVELOPMENT STEPS

Developing a program however requires more than just writing down series of instructions. When you write a computer program, it is good idea to start by developing a detailed plan or outline for the entire program. You should never start writing an assembly language program by just writing down instructions!

The program development steps are:

1. Defining a Problem

2. Representing program operations

3. Finding the right instruction

4. Writing a program

9.3 ASSEMBLY LANGUAGE PROGRAM DEVELOPMENT TOOLS

For all but the very simplest assembly language programs, you will probably want to use some type of microcomputer development system and program development tools to make your work easier. Most of the program development tools are programs which you run to perform some function on the program you are writing.

Program development tools are:

1. Editor

2. Assembler

3. Linker

4. Locator

5. Debugger

6. Emulator

1. **Editor**: An editor is a software tool that allows user, the construction of an assembly language program by providing a set of commands. By using this tool, the user can type and modify the program in assembly code. The program created by an editor is termed as *source program*.
2. **Assembler**: We have already discussed that an assembler allows conversion of assembly level language into machine language. Assemblers are of different types like one pass, two pass, cross, meta macro and resident assembler. *Source program generated by editor acts as input to the assembler*.
3. **Library Builder**: Library builder is a tool used to generate library files. Basically library files contain the functions that are frequently used by a program. For any particular application, whenever software is developed, then the programmers can link the library files with the programs. Linking library files with the program permits the copying of procedure needed by the program from library files to the program.
4. **Linker**: A linker allows the combining of relocatable object files of program modules with the library functions to generate a single executable file. Basically **program modules** are nothing but separate procedures of subdivisions of a large program into smaller tasks. This is so because whenever a program is divided into smaller tasks then each task has its individual procedure. Library files are also used by certain tasks according to their availability. Each program module allows separate assembling, testing and debugging. Further to have a complete executable file the object files of the modules and library files are linked together. The *linker creates a link map file* the holds the information about the address of the linked files.
5. **Debugger**: A debugger executes a program according to the controlling of the user. A debugger allows the location and correction of errors if present in any program, and this is known as **debugging**.
6. **Simulator**: A program that runs on the PC for simulating the operations of the recently designed system is known as a simulator. It also displays the contents of registers and memory locations on the computer screen. So, as the program operates, the change in contents can be monitored.
7. **Emulator**: An emulator is a tool utilized for testing and debugging both hardware and software of a microprocessor-based system. The system designer loads and runs the program on the emulator, and allows examining and changing of information inside the registers and memory locations.

It is to be noted here that assembly language programs are machine-dependent. This is so because every microprocessor supports different mnemonics. Thus different programs are developed for different microprocessors.

However, upward compatibility is provided by the manufacturers to the same class of microprocessors. This means programs that are developed for lower versions of the processor can be supportable even on higher versions without introducing any modifications.

9.3 PROGRAM DEVELOPMENT AND EXECUTION

The steps involved in Program Development and Execution of assembly language programs. Fig. 9.2 shows these steps. The left side of the figure shows the time period, at which each step in the overall process takes place.

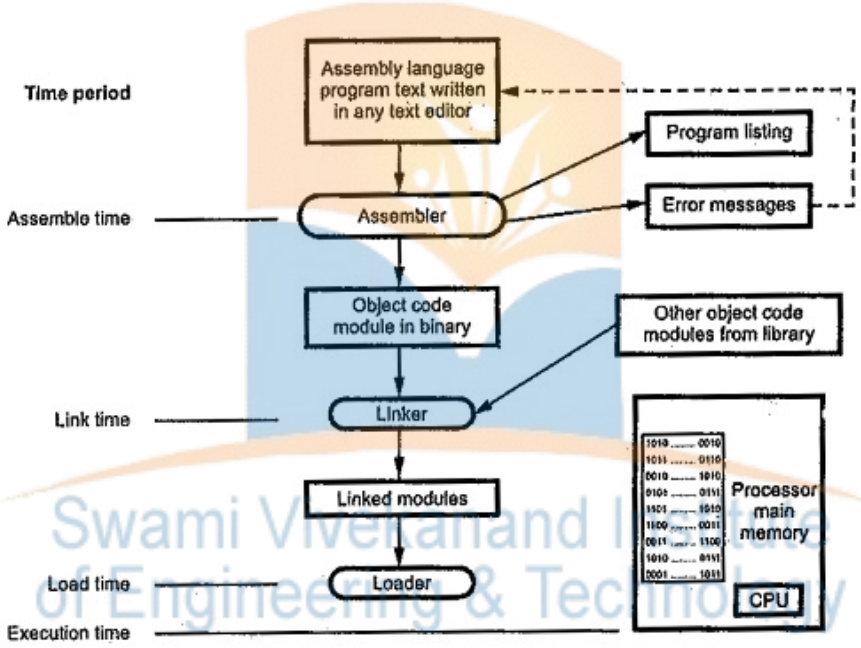


Fig. 9.2 Steps in program development and execution

The first step in the development process is to write an assembly language program. The assembly language program can be written with an ordinary text editor such as word star, edit and so on. The assembly language program text is an input to the assembler.

The assembler translates assembly language statements to their binary equivalents, usually known as object code. Time required translating assembly code to object code is called **Assemble Time**. During assembling process assembler checks for syntax errors and displays them before giving object code module.

The object code module contains the information about where the program or module to be loaded in memory. If the object code module is to be linked with other separately assembled modules then it contains additional linkage information. At link time, separately assembled modules are combined into one single load module, by the linker.

The linker also adds any required initialization or finalization code to allow the operating system to start the program running and to return control to the operating system after the program has completed.

Most linkers allow assembly language modules to be linked with object code modules compiled from high-level languages as well. This allows the programmer to insert time-critical assembly language routines, library modules into a program.

At load time, the program loader copies the program into the computer's main memory, and at execution time, program execution begins
Syntax for Assembly Language Instruction

Assembly language programming follows a unique syntax, in which the instructions are written in the format:

```
Label : Mnemonic  Operand1, Operand2  ;Comment
```

Label is an optional field of instruction in assembly language and acts as an identifier that provides a symbolic name to the first byte of the instruction and is generally used at the time of branching.

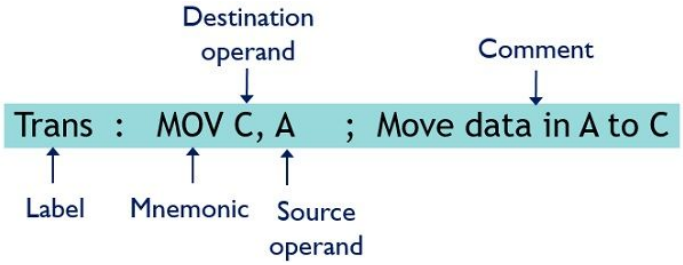
Mnemonics provide the information about the instruction to be performed and is a compulsory field of the instruction format.

Operand specifies the data over which operation is performed. It is to be noted here that the number of operations in an instruction depends on the type of instruction.

This is so because some hold no operand while some hold one or two operands. A comma is used to separate two operands of an instruction.

The comment field starts with a semicolon and it signifies the objective of a particular instruction.

For understanding consider the instruction given below:



9.4 REFERENCES

1. https://electronicsdesk.com/assembly-language-programming.html
2. https://maheshelectronics.files.wordpress.com/2015/06/assembly-language-programming1.pdf
3. https://www.eeeguide.com/program-development-and-execution/

CHAPTER 10

ASSEMBLY LANGUAGE PROGRAMMING

Ms. Tanika Thakur[1] Mr. Navdeep Randhawa[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

10.1 NEED OF ASSEMBLY LANGUAGE PROGRAMMING

We know that programs are a well-defined set of instructions used by programming devices like microprocessor, to perform a specific task.

The instructions in machine level language are written in binary form i.e., using 0 and 1. However, machine level language is quite complex and is not very much user-friendly, leading to cause difficulty in program development.

Hence, a less complex and user-friendly language was developed in which the instructions are specified using a few letters of an English word, thus making it quite easier. This language is known as assembly level language.

But it is noteworthy here that a microprocessor does not hold the ability to execute the programs written in assembly language. Thus it becomes necessary to convert assembly language programs into machine language so that the microprocessor can execute it.

So, a software tool known as assembler is used for converting assembly language into a machine language.

Thus a program written in assembly language is first converted into machine language and then these programs are executed by the microprocessor.

10.2 ASSEMBLY LANGUAGE PROGRAMS

Simple programs

1. Write an ALP in 8086 to perform an addition of two 8-bit numbers.

```
ASSUME CS: CODE

ORG 2000H

CODE SEGMENT
```

```
START: MOV SI, 3000H
MOV AL, [SI]
INC SI
MOV BL, [SI]
ADD AL, BL
INT 03H
CODE ENDS
END
```

Using data segment declaration

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
N1 DB 08H
N2 DB 02H
DATA ENDS
ORG 3000H
CODE SEGMENT
MOV AX, DATA
MOV DS, AX
MOV AL, N1
MOV BL, N2
ADD AL, BL
INT 03H
CODE ENDS
```

END

2. Write an ALP in 8086 to perform subtraction of two 8-bit numbers.

```
ASSUME CS: CODE
ORG 2000H
CODE SEGMENT
MOV SI, 3000H
MOV AL, [SI]
INC SI
MOV BL, [SI]
SUB AL, BL
INT 03H
CODE ENDS
END
```

3. Write an ALP in 8086 to perform multiplication of two 8-bit numbers.

```
ASSUME CS: CODE
ORG 2000H
CODE SEGMENT
MOV SI, 3000H
MOV AL, [SI]
INC SI
MOV BL, [SI]
MUL BL
INT 03H
```

```
CODE ENDS

END
```

4. Write an ALP in 8086 to perform 16-bit by 8-bit division.

```
ASSUME CS: CODE

ORG 2000H

CODE SEGMENT

MOV SI, 3000H

MOV AL, [SI]

INC SI

MOV AH, [SI]

INC SI

MOV BL, [SI]

DIV BL

INT 03H

CODE ENDS

END
```


5. Write an ALP in 8086 to perform an addition of two 16-bit numbers.

```
ASSUME CS: CODE

ORG 2000H

CODE SEGMENT

START: MOV SI, 3000H

MOV AX, [SI]

INC SI
```

```
INC SI

MOV BX, [SI]

ADD AX, BX

INT 03H

CODE ENDS

END
```

10.3 REFERENCES

1. https://maheshelectronics.files.wordpress.com/2015/06/assembly-language-programming1.pdf
2. https://electronicsdesk.com/assembly-language-programming.html
3. Gaonkar, Ramesh S. Microprocessor Architecture, Programming and Applications with the 8085, Penram International
4. Ram B, Fundamentals of Microprocessors and Microcomputers, Dhanpat Rai and Sons,

CHAPTER 11

PIN DIAGRAM AND DESCRIPTION OF 8086 MICROPROCESSOR

Ms. Yashu[1] Dr. Pertik Garg[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Associate Professor, Swami Vivekanand Institute of Engineering & Technology*

11.1 DEFINITION

8086 is a **16-bit microprocessor** and was created by Intel in 1978. The 8086 microprocessor contains **40 pins** dual in line.

However, unlike the 8085 microprocessor, an 8086 to have better performance, operates in 2 modes that are minimum and maximum mode.

The minimum mode is a single processor configuration while the maximum mode is a multiple processor configuration.

Due to this reason, in the 40 pin IC of 8086 microprocessor, 8 pins i.e., pin numbered from 24 to 32 are assigned different configurations separately according to the two modes.

11.2 8086 PIN DIAGRAM

8086 was the first 16-bit microprocessor available in 40-pin DIP (Dual Inline Package) chip. Let us now discuss in detail the pin configuration of 8086 Microprocessor.

Let us now discuss the signals in detail −

Power supply and frequency signals

It uses 5V DC supply at V_{CC} pin 40, and uses ground at GND pin 1 and 20 for its operation.

Clock signal

Clock signal is provided through Pin-19. It provides timing to the processor for operations. Its frequency is different for different versions, i.e. 5MHz, 8MHz and 10MHz.

Address/data bus

AD0-AD15. These are 16 address/data bus. AD0-AD7 carries low order byte data and AD8AD15 carries higher order byte data. During the first clock cycle, it carries 16-bit address and after that it carries 16-bit data.

Address/status bus

A16-A19/S3-S6. These are the 4 address/status buses. During the first clock cycle, it carries 4-bit address and later it carries status signals.

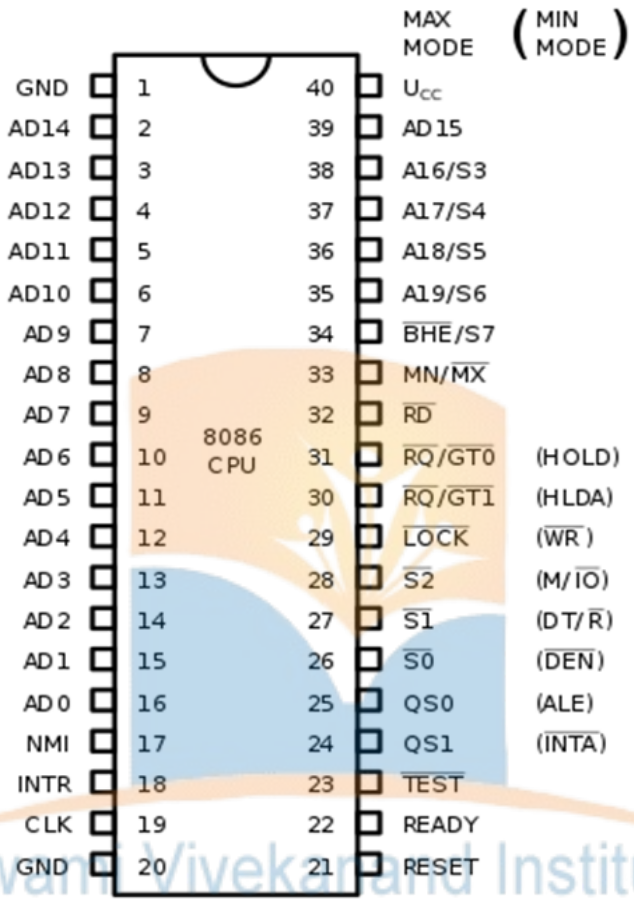


Fig. 11.1 Pin diagram of 8086 microprocessor

S7/BHE

BHE stands for Bus High Enable. It is available at pin 34 and used to indicate the transfer of data using data bus D8-D15. This signal is low during the first clock cycle, thereafter it is active.

Read (RD')

It is available at pin 32 and is used to read signal for Read operation.

Ready

It is available at pin 22. It is an acknowledgement signal from I/O devices that data is transferred. It is an active high signal. When it is high, it indicates that the device is ready to transfer data. When it is low, it indicates wait state.

RESET

It is available at pin 21 and is used to restart the execution. It causes the processor to immediately terminate its present activity. This signal is active high for the first 4 clock cycles to RESET the microprocessor.

INTR

It is available at pin 18. It is an interrupt request signal, which is sampled during the last clock cycle of each instruction to determine if the processor considered this as an interrupt or not.

NMI

It stands for non-maskable interrupt and is available at pin 17. It is an edge triggered input, which causes an interrupt request to the microprocessor.

TEST

This signal is like wait state and is available at pin 23. When this signal is high, then the processor has to wait for IDLE state, else the execution continues.

MN/MX'

It stands for Minimum/Maximum and is available at pin 33. It indicates what mode the processor is to operate in; when it is high, it works in the minimum mode and vice-versa.

INTA

It is an interrupt acknowledgement signal and id available at pin 24. When the microprocessor receives this signal, it acknowledges the interrupt.

ALE

It stands for address enable latch and is available at pin 25. A positive pulse is generated each time the processor begins any operation. This signal indicates the availability of a valid address on the address/data lines.

DEN

It stands for Data Enable and is available at pin 26. It is used to enable Transreceiver 8286. The transreceiver is a device used to separate data from the address/data bus.

DT/R

It stands for Data Transmit/Receive signal and is available at pin 27. It decides the direction of data flow through the transreceiver. When it is high, data is transmitted out and vice-a-versa.

M/IO

This signal is used to distinguish between memory and I/O operations. When it is high, it indicates I/O operation and when it is low indicates the memory operation. It is available at pin 28.

WR

It stands for write signal and is available at pin 29. It is used to write the data into the memory or the output device depending on the status of M/IO signal.

HLDA

It stands for Hold Acknowledgement signal and is available at pin 30. This signal acknowledges the HOLD signal.

HOLD

This signal indicates to the processor that external devices are requesting to access the address/data buses. It is available at pin 31.

QS_1 and QS_0

These are queue status signals and are available at pin 24 and 25. These signals provide the status of instruction queue. Their conditions are shown in the following table −

QS_0	QS_1	Status
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty the queue
1	1	Subsequent byte from the queue

S_0, S_1, S_2

These are the status signals that provide the status of operation, which is used by the Bus Controller 8288 to generate memory & I/O control signals. These are available at pin 26, 27, and 28. Following is the table showing their status −

S_2	S_1	S_0	Status
0	0	0	Interrupt acknowledgement
0	0	1	I/O Read
0	1	0	I/O Write

0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

LOCK

When this signal is active, it indicates to the other processors not to ask the CPU to leave the system bus. It is activated using the LOCK prefix on any instruction and is available at pin 29.

RQ/GT_1 and RQ/GT_0

These are the Request/Grant signals used by the other processors requesting the CPU to release the system bus. When the signal is received by CPU, then it sends acknowledgment. RQ/GT_0 has a higher priority than RQ/GT_1.

11.3 REFERENCES

1. https://www.tutorialspoint.com/microprocessor/microprocessor_8086_pin_configuration.htm
2. https://www.geeksforgeeks.org/pin-diagram-8086-microprocessor/
3. https://electronicsdesk.com/pin-diagram-of-8086-microprocessor.html
4. https://www.includehelp.com/embedded-system/pin-diagram-of-8086-microprocessor.aspx
5. https://techatronic.com/what-is-8086-microprocessor-8086-pin-diagram-8086-architecture/

CHAPTER 12

MINIMUM MODE CONFIGURATION OF 8086 MICROPROCESSOR (MIN MODE)

Ms. Kiran Bala[1] Ms. Komal Dhiman[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

12.1 INTRODUCTION

In the 8086 microprocessor, there are two modes of operation: minimum mode and maximum mode.

Minimum mode is used when the 8086 microprocessor is operating as a standalone processor without any external coprocessors or support chips. In this mode, the 8086 uses a single 8-bit bus for both data and instructions, and a single 20-bit address bus. The minimum mode requires a minimum set of support chips, such as clock generator, address latch, and bus controller.

Maximum mode is used when the 8086 microprocessor is operating with one or more external coprocessors or support chips. In this mode, the 8086 uses a multiplexed bus for data and instructions, and a 20-bit address bus. The maximum mode requires additional support chips, such as a bus controller, a clock generator, and a data buffer.

An 8086 is a 16-bit HMOS microprocessor. It is available in 40 pin DIP chip. It uses a 5V DC supply for its operation. The 8086 uses a 20-line address bus. It has a 16-line data bus. The 20 lines of the address bus operate in multiplexed mode. The 16-low order address bus lines have been multiplexed with data and 4 high-order address bus lines have been multiplexed with status signals.

12.2 MINIMUM MODE

The Minimum Mode of the 8086 microprocessor is designed for systems wherein the processor shares the gadget bus with different devices and does not act as the bus controller. This mode is commonly utilized in single-processor systems or while the device calls for fewer assets and a much less complex configuration.

12.3 KEY FEATURES OF THE MINIMUM MODE ENCOMPASS

1. **Bus Controller:** In Minimum Mode, the microprocessor relies on an outside bus controller, which includes the 8284 clock generator, to address bus control indicators and bus arbitration. The bus controller coordinates the activities of various devices linked to the machine bus.

2. **Single Processor:** The Minimum Mode is generally used in unmarried-processor structures wherein the microprocessor is the number one processing unit and does not require the complexities related to multiprocessor structures.

3. **Limited Address and Data Lines:** In Minimum Mode, most effective 16 address traces and 16 facts lines of the 8086 microprocessor are utilized, proscribing the reminiscence addressing skills and I/O options compared to the Maximum Mode.

4. **Reduced Complexity:** The Minimum Mode configuration simplifies the machine design with the aid of offloading bus manager obligations to an external bus controller, decreasing the overall complexity and price of the gadget.

Choosing the correct mode relies upon the unique requirements of the system. The Maximum Mode is appropriate for big-scale systems with a couple of processors, massive reminiscence and I/O requirements, and the need for full control over the device bus. On the other hand, the Minimum Mode is favored for easier structures with a single processor and restrained useful resource requirements, where the microprocessor stocks the bus with different gadgets.

12.4 PIN DEFINITIONS (24 TO 31) IN MINIMUM MODE

INTA (Interrupt Acknowledge) Output: This indicates recognition of an interrupt request. It consists of two negative going pulses in two consecutive bus cycles. The first pulse informs the interface that its request has been recognized and upon receipt of the second pulse, the interface is to send the interrupt type to the processor over the data bus.

ALE (Address Latch Enable) output: This signal is provided by 8086 to demultiplex the AD_0-AD_{15} into A_0-A_{15} and D_0-D_{15} using external latches.

DEN (Data Enable) output: This signal informs the transceivers that the CPU is ready to send or receive data.

DT/R (Data transmit/Receive) output: This signal is used to control data flow direction. High on this pin indicates that the 8086 is transmitting the data and low indicates that the 8086 is receiving the data.

M/IO output: It is used to distinguish memory data transfer, (M/I0 = HIGH) and I/O data transfer (M/I0 = LOW).

WR: Write output: WR is low whenever the 8086 is writing data into memory or an I/O device.

HOLD input, HLDA output: A HIGH on HOLD pin indicates that another master (DMA) is requesting to take over the system bus. On receiving HOLD signal processor outputs HLDA signal HIGH as an acknowledgment. At the same time, processor tristates the system bus. A low on HOLD gives the system bus control back to the processor. Processor then outputs low signal on HLDA.

12.5 MINIMUM MODE CONFIGURATION

Fig. 12.1 shows the typical Minimum Mode Configuration of 8086. As shown in the figure, AD_0-AD_{15}, A_{16}/S3-A_{19}/S_6, and BHE/S_7 signals are multiplexed. These signals are demultiplexed by external latches and ALE signal generated by the processor. This is accomplished by using three latch ICs (Intel 8282/8283), two of them are required for a 16-bit address and three are needed if a full 20-bit address is used. Fig. 12.2 shows the internal block diagram of 8282/8283 latches. The 8282 provides noninverting outputs while the 8283 version inverts the input data. In addition to their demultiplexing function, these chips also buffer the address lines, providing increased output driving capability. The output low level is specified as 0.45 V maximum with a sink current of 32 mA maximum. The high level is specified as 2.4 V minimum while supplying a 5 mA maximum high level load current.

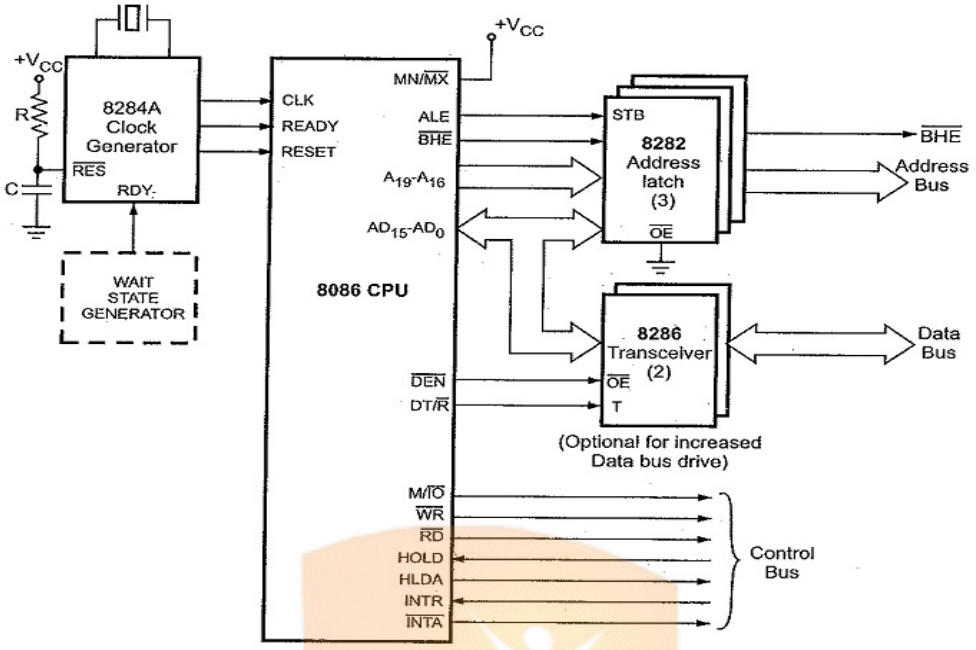


Fig. 12.1 Typical minimum mode configuration

If a system includes several interfaces then to increase current sourcing/sinking capacities it is necessary to use drivers and receivers (transceiver) for data bus also. The Intel 8286 device is used to implement the transceiver block shown in Fig. 12.1. The 8286 contains 16 tristate elements, eight receivers, and eight drivers. Therefore two 8286s are required to service 16 data lines of 8086.

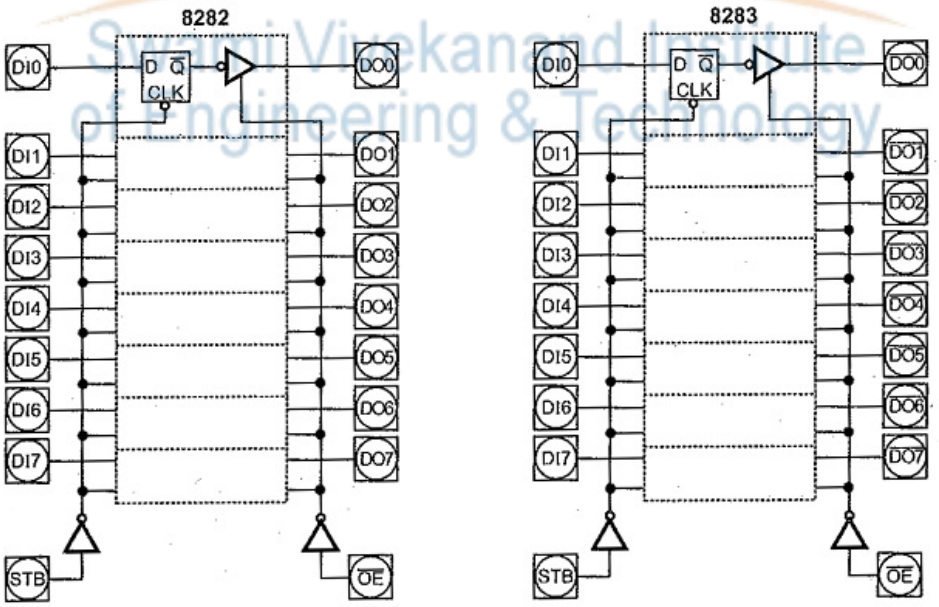


Fig. 12.2 Internal diagram of 8282 and 8283

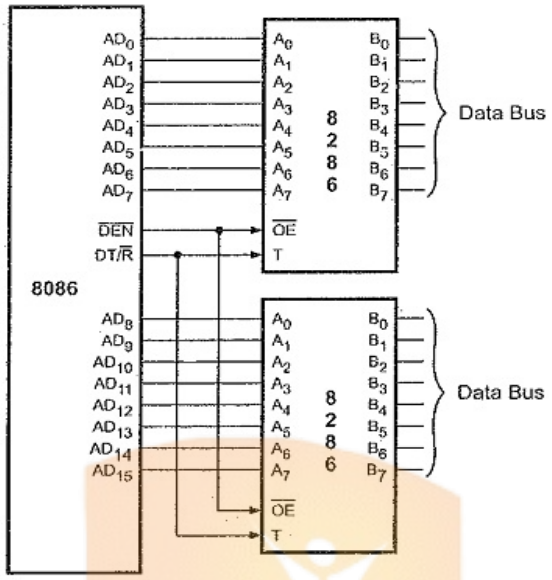


Fig. 12.3 Connections details of 8286

Fig. 12.3 shows the detailed connections of 8286. DT/R signal is connected to the T input, which controls the direction of the data flow. When this signal is low, receivers are enabled, so that 8086 can read data from memory or input device. To write data into memory or output device, the 8086's DT/R signal goes high. Due to this drivers are enabled to transfer data from 8086 to the memory or the output device. At the time of data transfer, to enable output of transceiver its OE should be low. This is accomplished by connecting DEN signal of 8086 to the OE pin of 8286, since DEN signal goes low when CPU is ready to send or receive data.

The third component other than the processor that appears in Fig. 12.1 is an 8284 clock generator. The 8284 clock generator does the following functions

- Clock generation
- RESET synchronization
- READY synchronization
- Peripheral clock generation

The Fig. 12.4 shows the internal logic diagram of 8284.

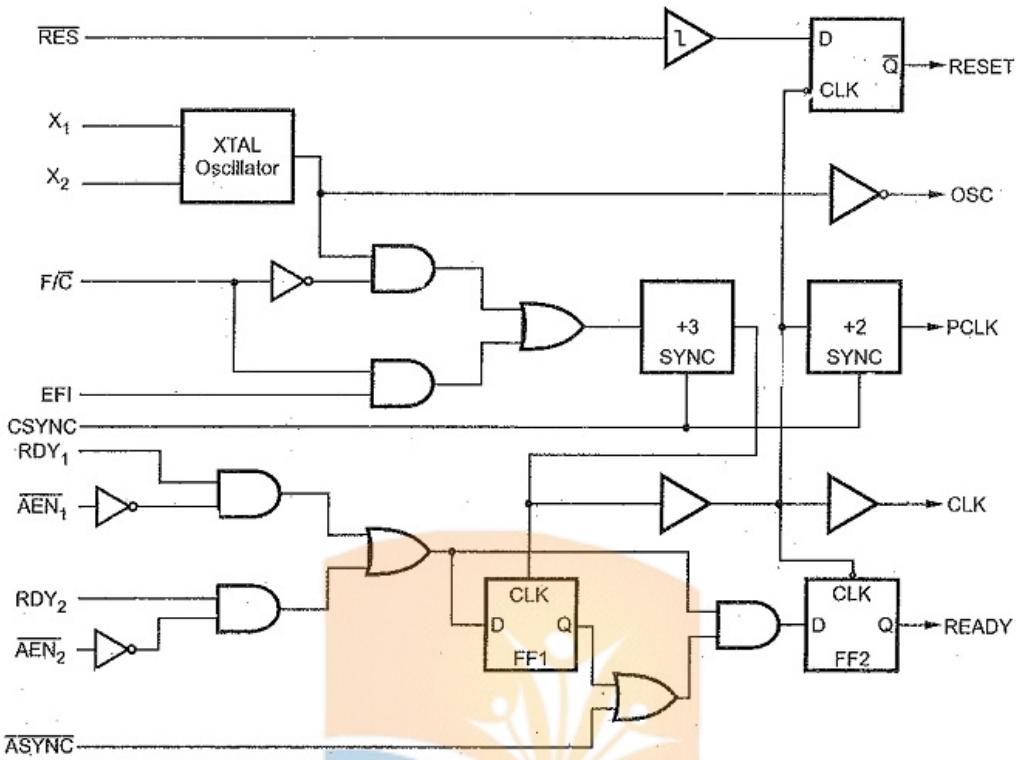


Fig. 12.4 Internal logic diagram of 8284

The top half of the logic diagram represents the clock and reset synchronization section of the 8284 clock generator. As shown in the logic diagram, the crystal oscillator has two inputs: X_1 and X_2. If a crystal is attached to X_1 and X_2, the oscillator generates a square-wave signal at the same frequency as the crystal. The output of oscillator is fed to an AND gate and also to an inverter buffer that provides the osc output signal. The F/C signal selects one of the oscillator inputs. When F/C input is 1, the EFI input determines the frequency; otherwise oscillator determines the frequency. When EFI input is used, CSYNC signal is used for multiple buffered before it leaves the clock generator. As shown in the Fig. 12.4, the output of the divide-by-3 counter generates the timing for ready synchronization, a signal for another counter (divide-by-2), and the CLK signal to the 8086/8088 microprocessors. The two cascaded counters (divide-by-3 and divide-by-2) provide the divide-by-6 output at PCLK, which can be used to provide clock input for peripherals. The address enable pins, AEN_1 and AEN_2 are provided to qualify the bus ready signals, RDY_1 and RDY_2, respectively.

The reset circuit of 8284 consists of a schmitt trigger buffer and a single D flip-flop circuit. The D flip-flop ensures that the timing requirements of the 8086/8088 RESET input are met. This circuit applies the RESET signal to the microprocessor on the negative edge (1 to 0 transition) of each clock. The 8086/8088 microprocessors sample RESET at the positive edge (0 to 1 transition) of the clocks; therefore, this circuit meets the timing requirements of the 8086/8088.

The RC circuit provides a logic 0 to the RES input pin when power is first applied to the system. After a short time, the RES input becomes a logic 1 because the capacitor charges toward + 5.0 V through the register. A push button switch allows the microprocessor to be reset by the operator.

The status on the M/IO, RD, and WR lines decides the type of data transfer, as listed in the Table 12.1.

Table 12.1 Type of data transfer

$M/\overline{IO}$	$\overline{RD}$	$\overline{WR}$	Operation
0	0	1	I/O read
0	1	0	I/O write
1	0	1	Memory read
1	1	0	Memory write

HOLD and HLDA signals are used to interface other bus masters like DMA controller. Interrupt request (INTR) and interrupt acknowledge (INTA) are used to extend the interrupt handling capacity of the 8086 with the help of interrupt controller.

12.6 REFERENCES

1. https://www.eeeguide.com/minimum-mode-configuration-of-8086/
2. https://www.geeksforgeeks.org/difference-between-minimum-mode-and-maximum-mode-in-8086-microprocessor/
3. https://www.javatpoint.com/maximum-mode-and-minimum-mode-in-8086-microprocessor

CHAPTER 13

MAXIMUM MODE CONFIGURATION OF 8086 MICROPROCESSOR (MAX MODE)

Mr. Ishant Premi[1] Ms. Tanika Thakur[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

13.1 8086 MICROPROCESSOR CHARACTERISTICS

- It contains 20 bit address bus.
- It contains 16-bit data bus, therefore 8086 is called as **16-bit microprocessor.**
- It is 2-stage pipelined processor. It can prefetch 6 bytes from memory and store into queue to increase the speed of the execution.
- It's control bus carries signals for executing operations such as read, write etc.
- It has Memory Banks. 2 banks of 512KB each. These banks are called as lower Bank (even) and higher Bank (odd).
- In 8086 the entire memory is divided into four memory segments which are code, stack, data and extra segment.
- 8086 has 16 bit IO address.
- It has 256 interrupts.

13.2 8086 HAS TWO OPERATING MODES

1. Minimum mode
2. Maximum mode

Minimum mode:

- In this 8086 is the only processor in the system. In a minimum mode 8086 system.
- 8086 is operated in minimum mode when MN/MX' pin to logic 1.
- In this mode, all the control signals are given out by the 8086 itself.

Maximum mode:

The Maximum Mode of the 8086 microprocessor is supposed for use in structures where the processor is the significant element and has full control over the device bus. In this mode, the microprocessor acts because of the bus controller and interacts without delay with other gadgets together with memory, I/O ports, and co-processors. The Maximum Mode configuration calls for

extra help chips, including the 8288 bus controller, to manipulate bus arbitration and generate control alerts.

- In this we can connect more processors to 8086 (8087/8089).
- 8086 max mode is basically for implementation of allocation of global resources and passing bus control to other coprocessor (i.e. second processor in the system), because two processors cannot access system bus at same instant.
- All processors execute their own program.
- The resources which are common to all processors are known as global resources.
- The resources which are allocated to a particular processor are known as local or private resources.

13.3 KEY FEATURES OF THE MAXIMUM MODE CONSIST OF:

1. **Bus Control:** The 8086 microprocessor takes control of the machine bus, permitting it to initiate and control all records transfers among diverse devices related to the gadget.

2. **Multiple Processors:** The Maximum Mode supports the usage of a couple of processors, making it suitable for multiprocessor structures. Each processor is assigned specific obligations, and that they talk through shared memory.

3. **External Bus Controller:** In Maximum Mode, an outside bus controller, such as the 8288, is required to address bus arbitration and generate bus manage indicators. This controller facilitates efficient statistics transfer and prevents conflicts amongst distinctive gadgets.

4. **Expanded Address and Data Lines:** The Maximum Mode lets in the usage of all 20 cope with lines and sixteen records strains of the 8086 microprocessor, allowing admission to a larger memory space and extra great I/O abilities.

13.4 CIRCUIT EXPLANATION OF MAXIMUM MODE

- When MN/ MX' =0, 8086 works in max mode.

- Clock is provided by 8284 clock generator.

- **8288 bus controller**- Address form the address bus is latched into 8282 8-bit latch. Three such latches are required because address bus is 20 bit. The ALE (Address latch enable) is connected to STB(Strobe) of the latch. The ALE for latch is given by 8288 bus controller.

- The data bus is operated through 8286 8-bit transceiver. Two such transceivers are required, because data bus is 16-bit. The transceivers are enabled the DEN signal, while the direction of data is controlled by the DT/R signal. DEN is connected to OE' and DT/ R' is connected to T. Both DEN and DT/ R' are given by 8288 bus controller.

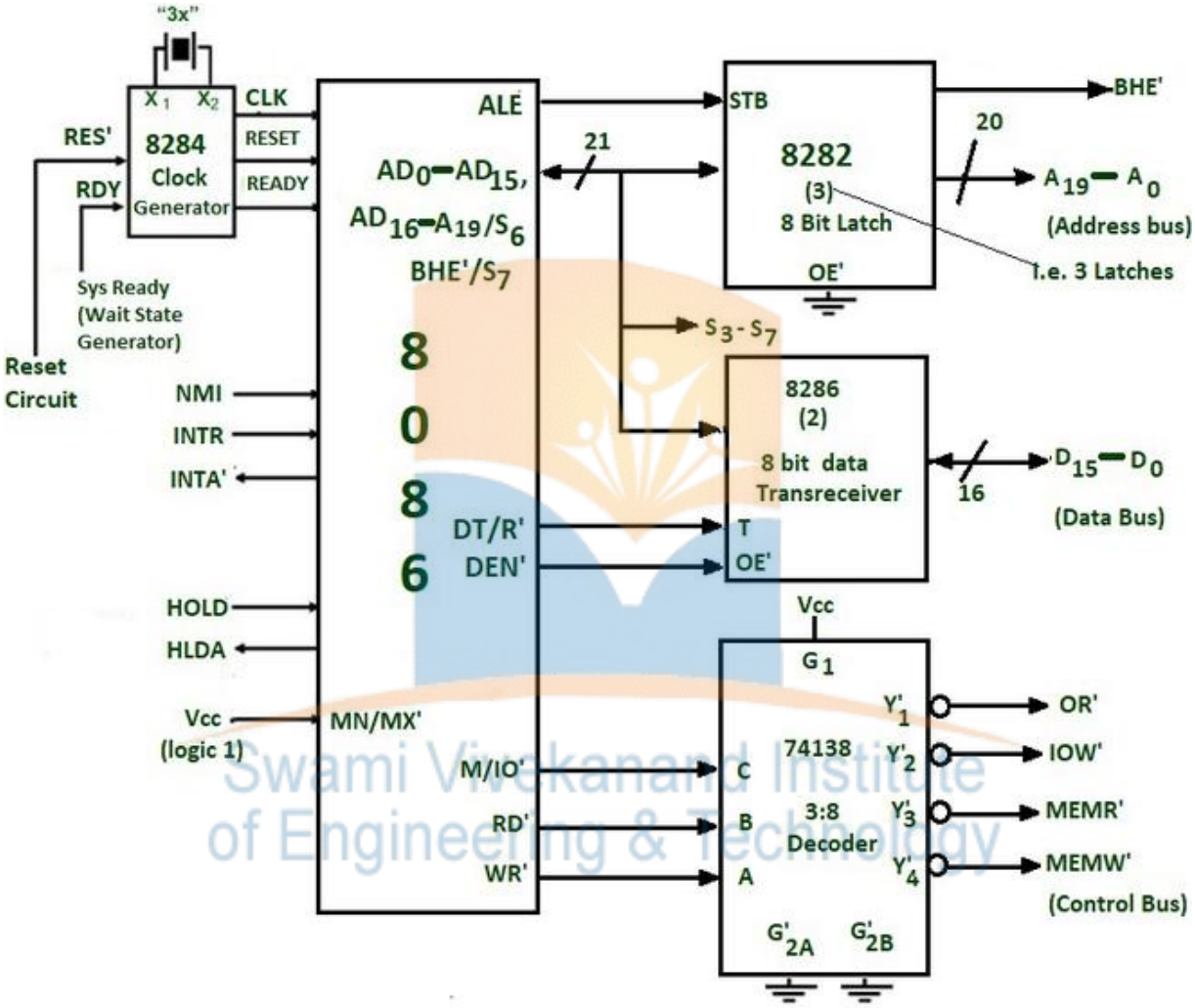


Fig. 13.1 Maximum mode circuit

DEN (Of 8288)	DT/ R'	Action
0	X	Transreceiver is disabled
1	0	Receive data
1	1	Transmit data

- Control signals for all operations are generated by decoding S'_2, S'_1 and S'_0 using 8288 bus controller.

S'_2	S'_1	S'_0	Processor State (What the µP wants to do)	8288 Active Output (What Control signal should 8288 generate)
0	0	0	Interrupt Acknowledge	INTA'
0	0	1	Read I/O Port	IORC'
0	1	0	Write I/O Port	IOWC' and AIOWC'
0	1	1	Halt	None
1	0	0	Instruction Fetch	MRDC'
1	0	1	Memory Read	MRDC'
1	1	0	Memory Write	MWTC' and AMWTC'
1	1	1	Inactive	None

- Bus request is done using RQ' / GT' lines interfaced with 8086. RQ_0/GT_0 has more priority than RQ_1/GT_1.
- INTA' is given by 8288, in response to an interrupt on INTR line of 8086.
- In max mode, the advanced write signals get enabled one T-state in advance as compared to normal write signals. This gives slower devices more time to get ready to accept the data, therefore it reduces the number of cycles.

13.5 ADVANTAGES OF MAX MODE

- It helps to interface more devices like 8087.This interface is also called a **closely coupled co-Processor configuration**. In this 8086 is called as the host and 8087 as Co-processor.
- It supports multiprocessing, Therefore it helps to increase the efficiency.
- The 8087 was the first floating-point coprocessor for the 8086 series of microprocessors. The purpose of the 8087 was to increase calculations for floating point operations, such as add, sub, multiply, divide, and square root.

13.6 DISADVANTAGES OF MAX MODE OVER MIN MODE

- It has more complex circuit than min mode.

13.7 REFERENCES

1. https://www.eeeguide.com/minimum-mode-configuration-of-8086/
2. https://www.geeksforgeeks.org/difference-between-minimum-mode-and-maximum-mode-in-8086-microprocessor/
3. https://www.javatpoint.com/maximum-mode-and-minimum-mode-in-8086-microprocessor

CHAPTER 14

BUS TIMINGS FOR MINIMUM MODE AND MAXIMUM MODE

Ms. Komal Dhiman[1] Mr. Ishant Premi[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

14.1 BUS TIMINGS FOR MINIMUM MODE

The timing diagrams of input and output transfers for Minimum Mode Configuration of 8086 are shown in the Fig. 14.1 and 14.2 respectively. These are explained in steps.

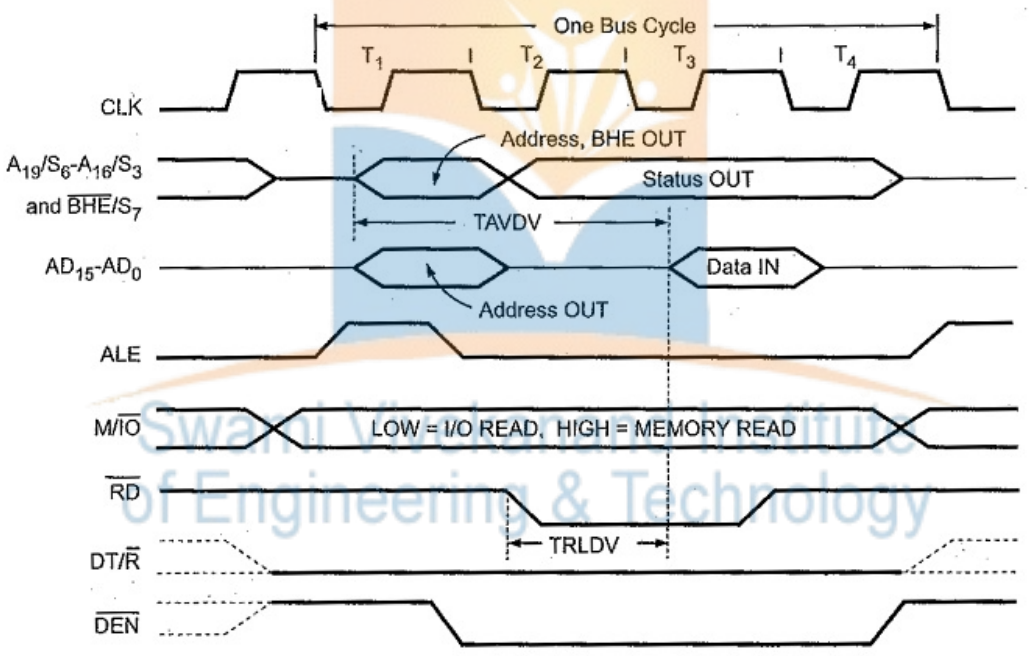


Fig. 14.1 (a) Input (read operation)

1. When processor is ready to initiate the bus cycle, it applies a pulse to ALE during T_1. Before the falling edge of ALE, the address, BHE, M/IO, DEN and DT/R must be stable i.e. DEN = high and DT/R = 0 for input or DT/R = 1 for output.

2. At the trailing edge of ALE, ICs 74LS373 or 8282 latches the address.

3. During T_2 the address signals are disabled and S_3-S_7 are available on AD_{16}/S_3-AD_{19}/S_6 and BHE/S_7. Also DEN is lowered to enable transceiver.

4. In case of input operation, RD is activated during T_2 and AD_0 to AD_{15} go in high impedance preparing for input.

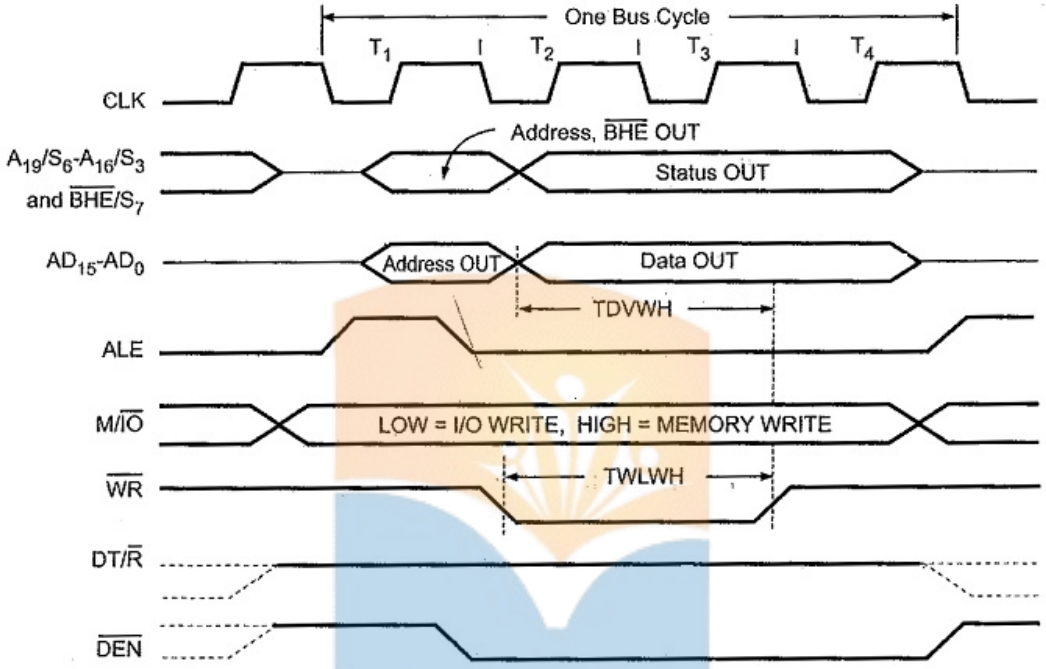


Fig. 14.2 Output (write operation)

5. If memory or I/O interface can perform the transfer immediately; there are no wait states and data is output on the bus during T_3.

6. After the data is accepted by the processor, RD is raised high at the beginning of T_4.

7. Upon detecting this transition during T_4, the memory or I/O device will disable its data signals.

8. For an output operation, processor applies WR = 0 and then the data on the data bus during T_2.

9. In T_4, WR is raised high and data signals are disabled.

10. For either input or output operation, DEN is raised during T_4 to disable the Also M/IO is set according to the next transfer at this time or during next T_1 state. Thus length

of bus cycle in 8086 is four clock cycle. If the bus is to be inactive after completion of bus cycle, then the gap between the successive cycles is filled by ideal state clock cycles.

When the memory or I/O device is not able to respond quickly during transfer, wait states (T_w) are inserted between T_3 and T_4 by disabling the READY input of the 8086. The bus activity during wait state is same as during T_3.

14.2 BUS TIMING DIAGRAM OF MAXIMUM MODE

The Bus Timing Diagram of 8086 of input and output transfers are shown in the Fig. 14.3 and 14.4 respectively.

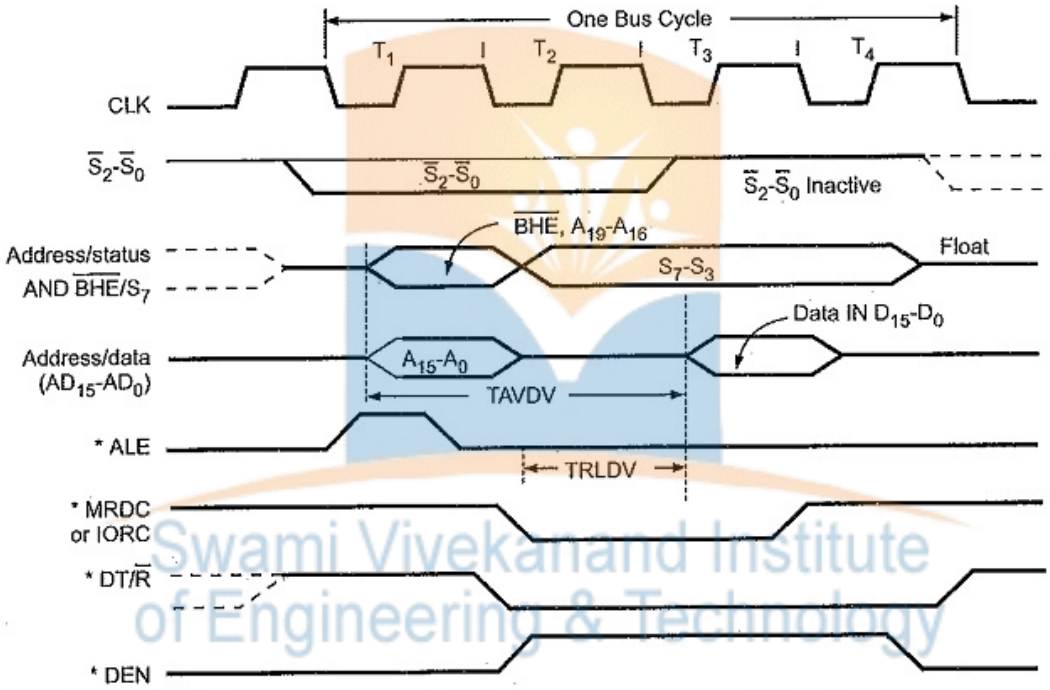


Fig 14.3 Input (read operation)

These are explained in steps.

1. S_0,S_1,S_2 are set at the beginning of bus cycle. On detecting the change on passive state $S_0 = S_1 = S_2 = 1$, the 8288 bus controller will output a pulse on its ALE and apply a required signal to its DT/R pin during T_1.

2. In T_2, 8288 will set DEN = 1 thus enabling transceiver. For an input, 8288 it will activates MRDC or IORC. These signals are activated until T_4. For an output, the

AMWC or AIOWC is activated from T_2 to T_4 and MWTC or IOWC is activated from T_3 to T_4.

3. The status bits S_0 to S_2 remain active until T_3, and become passive during T_3 and T_4.
4. If ready input is not activated before T3, wait state will be inserted between T_3 and T_4.

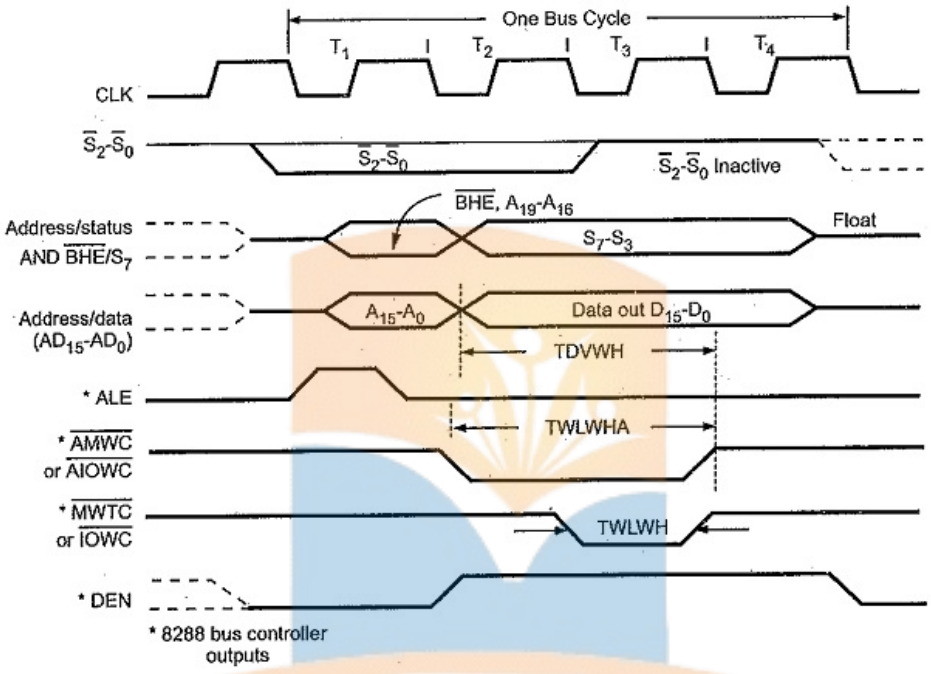


Fig. 14.4 Output (write operation)

14.3 DIFFERENCE BETWEEN MAXIMUM MODE AND MINIMUM MODE

Feature	Maximum Mode	Minimum Mode
Bus Control	8086 acts as bus controller and manages the gadget bus.	8086 does not act as a bus controller and relies on an external bus controller.
Bus Controller	Requires an external bus controller (e.G.8288) for bus arbitration and manipulate sign generation.	Requires an external bus controller (e.G., 8284) for bus manage signal era and coordination.

Address Lines	Utilizes all 20 address traces for extended reminiscence addressing.	Utilizes handiest sixteen cope with traces for limited reminiscence addressing.
Data Lines	Uses all 16 records strains for statistics switch.	Uses the handiest 16 data traces for facts switch.
Processor	Supports a couple of processors for multiprocessor structures.	Designed for unmarried-processor systems.
System Control	Full control over the device bus and gadgets linked to it.	Share management of the device bus with other gadgets.
System Complexity	More complicated device design because of bus manipulation obligations and support chips.	Simple gadget layout as bus manager is treated by means of an external bus controller.
Memory and I/O	Can get right of entry to a larger memory space and accommodate extra good sized I/O abilities.	Limited reminiscence addressing and I/O options.
System Cost	Generally higher gadget value because of extra support chips.	Lower machine fee due to reduced complexity and less aid chips.
Typical	Use Suitable for larger systems with more than one processor and sizable memory/I/O requirements.	Preferred for simpler systems with an unmarried processor and restricted resource desires.

These are the important differences between Maximum Mode and Minimum Mode in the 8086 Microprocessor. The preference among the 2 modes relies upon the particular necessities and complexity of the system being designed.

14.4 REFERENCES

1. https://www.eeeguide.com/minimum-mode-configuration-of-8086/
2. https://www.geeksforgeeks.org/difference-between-minimum-mode-and-maximum-mode-in-8086-microprocessor/
3. https://www.javatpoint.com/maximum-mode-and-minimum-mode-in-8086-microprocessor

CHAPTER 15

INTERRUPTS IN 8086 MICROPROCESSOR

Dr. Pertik Garg[1] Ms. Vandana[2]

[1]*Associate Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

An interrupt is a condition that halts the microprocessor temporarily to work on a different task and then returns to its previous task. An interrupt is an event or signal that requests the CPU's attention. This halt allows peripheral devices to access the microprocessor. Whenever an interrupt occurs, the processor completes the current instruction and starts the implementation of an Interrupt Service Routine (ISR) or Interrupt Handler. ISR is a program that tells the processor what to do when the interrupt occurs. After the ISR execution, control returns to the main routine where it was interrupted. In the 8086 microprocessors following tasks are performed when the microprocessor encounters an interrupt:

1. The value of the flag register is pushed into the stack. It means that first, the value of SP (Stack Pointer) is decremented by two then the value of the flag register is pushed to the memory address of the stack segment.
2. The value of starting memory address of CS (Code Segment) is pushed into the stack.
3. The value of IP (Instruction Pointer) is pushed into the stack.
4. IP is loaded from word location (Interrupt type) * 04.
5. CS is loaded from the following word location.
6. Interrupt and Trap flags are reset to 0.

15.1 INTERRUPT STRUCTURE OF 8086

An Interrupt Structure of 8086 can come from any one the three sources:

- External signal
- Special Instruction in the program
- Condition produced by instruction

External Signal (Hardware Interrupt):

An 8086 can get interrupt from an external signal applied to the non-maskable interrupt (NMI) input pin; or the interrupt (INTR) input pin.

Special Instruction:

Interrupt Structure of 8086 supports a special instruction, INT to execute special program. At the end of the interrupt service routine, execution is usually returned to the interrupted program.

Condition Produced by Instruction:

An 8086 is interrupted by some condition produced in the 8086 by the execution of an instruction. For example divide by zero: Program execution will automatically be interrupted if you attempt to divide an operand by zero.

At the end of each instruction cycle 8086 Interrupts checks to see if there is any interrupt request. If so, 8086 responds to the interrupt by performing series of actions (Refer Fig. 15.1).

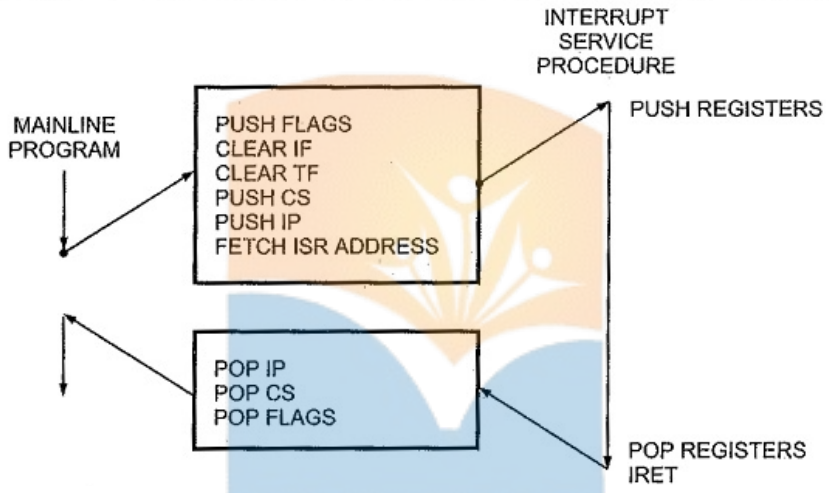


Fig. 15.1 Interrupt response

1. It decrements stack pointer by 2 and pushes the flag register on the stack.
2. It disables the INTR interrupt input by clearing the interrupt flag in the flag
3. It resets the trap flag in the flag register.
4. It decrements stack pointer by 2 and pushes the current code segment register contents on the stack.
5. It decrements stack pointer by 2 and pushes the current instruction pointer contents on the stack.
6. It does an indirect far jump at the start of the procedure by loading the CS and IP values for the start of the interrupt service routine (ISR).

An IRET instruction at the end of the interrupt service procedure returns execution to the main program.

15.2 INTERRUPT VECTOR TABLE 8086

Now the question is "How to get the values of CS and IP register?" The 8086 gets the new values of CS and IP register from four memory addresses. When it responds to an interrupt, the 8686 goes to memory locations to get the CS and IP values for the start of the interrupt service routine. In an Interrupt Structure of 8086 system the first 1 Kbyte of memory from 00000H to 003FFH is reserved for storing the starting addresses of interrupt service routines. This block of memory is often called the **Interrupt Vector Table in 8086** or the **interrupt pointer table**. Since 4 bytes are required to store the CS and IP values for each interrupt service procedure, the table can hold the starting addresses for 256 interrupt service routines. Fig. 15.2 shows how the 256 interrupt pointers are arranged in the memory table.

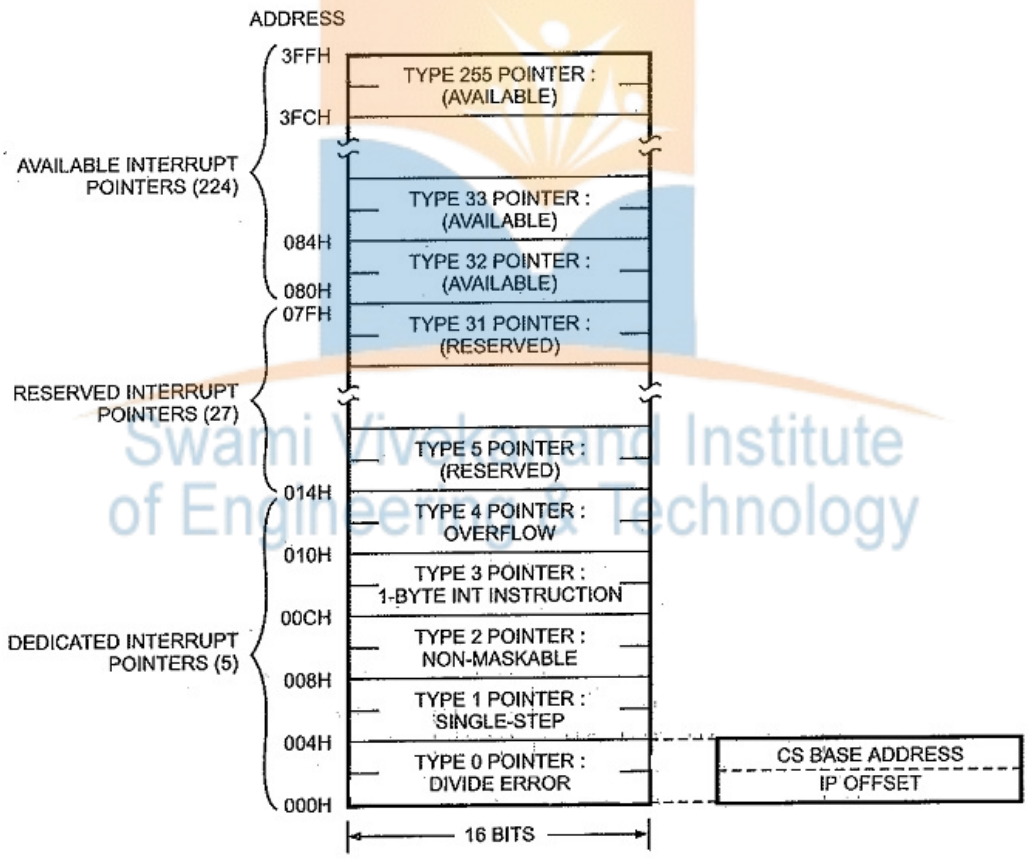


Fig. 15.2 Interrupt vector table

Each interrupt type is given a number between 0 to 255 and the address of each interrupt js found by multiplying the type by 4 e.g. for type 11, interrupt address is 11 x 4 = 44_{10}= 0002CH

Only first five types have explicit definitions such as divide by zero and non-maskable interrupt. The next 27 interrupt types, from 5 to 31, are reserved by Intel for use in future microprocessors. The upper 224 interrupt types, from 32 to 255, are available for user for hardware or software interrupts.

When the 8086 responds to an interrupt, it automatically goes to the specified location in the Interrupt Vector Table in 8086 to get the starting address of interrupt service routine. So user has to load these starting addresses for different routines at the start of the program.

15.3 TYPES OF INTERRUPTS

Figure 15.3 shows the types of interrupts we have in 8086 microprocessor −

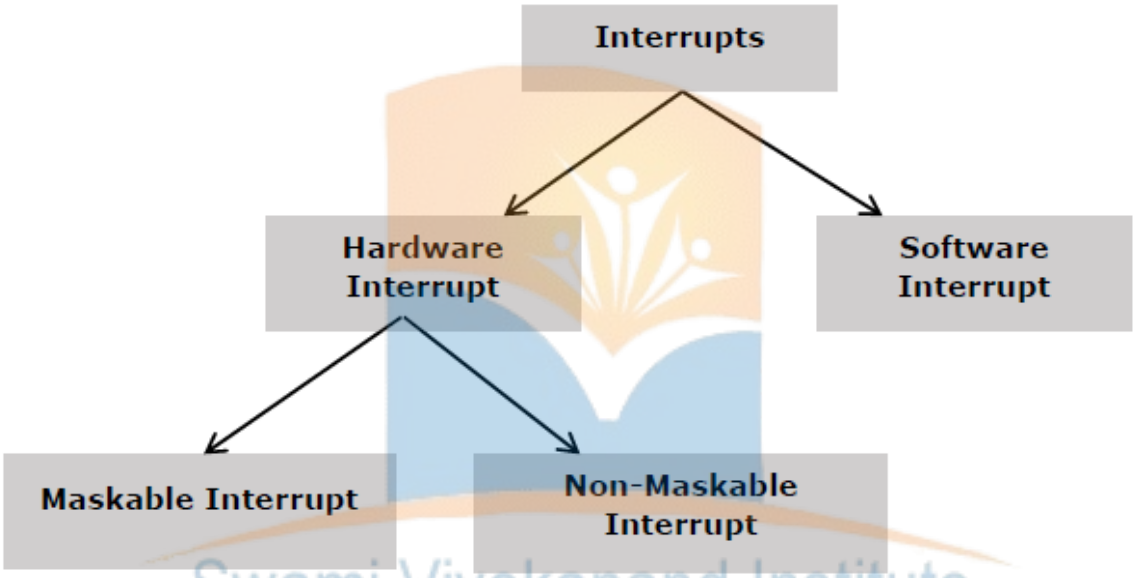


Fig. 15.3 Types of interrupts

Hardware Interrupts

Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.

The 8086 has two hardware interrupt pins, i.e. NMI and INTR. NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority. One more interrupt pin associated is INTA called interrupt acknowledge.

NMI

It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR) and it is of type 2 interrupt.

When this interrupt is activated, these actions take place −

- Completes the current instruction that is in progress.
- Pushes the Flag register values on to the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.
- IP is loaded from the contents of the word location 00008H.
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

INTR

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction. It should not be enabled using clear interrupt Flag instruction.

The INTR interrupt is activated by an I/O port. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends '0' on INTA pin twice. The first '0' means INTA informs the external device to get ready and during the second '0' the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

These actions are taken by the microprocessor −

- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location X × 4
- CS is loaded from the contents of the next word location.
- Interrupt flag and trap flag is reset to 0

Software Interrupts

Some instructions are inserted at the desired position into the program to create interrupts. These interrupt instructions can be used to test the working of various interrupt handlers. It includes −

INT- Interrupt instruction with type number

It is 2-byte instruction. First byte provides the op-code and the second byte provides the interrupt type number. There are 256 interrupt types under this group.

Its execution includes the following steps −

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 'type number' × 4
- CS is loaded from the contents of the next word location.

- Interrupt Flag and Trap Flag are reset to 0

The starting address for type0 interrupt is 000000H, for type1 interrupt is 00004H similarly for type2 is 00008H and ……so on. The first five pointers are dedicated interrupt pointers. i.e. −

- **TYPE 0** interrupt represents division by zero situation.
- **TYPE 1** interrupt represents single-step execution during the debugging of a program.
- **TYPE 2** interrupt represents non-maskable NMI interrupt.
- **TYPE 3** interrupt represents break-point interrupt.
- **TYPE 4** interrupt represents overflow interrupt.

The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interrupts from 32 to Type 255 are available for hardware and software interrupts.

INT 3-Break Point Interrupt Instruction

It is a 1-byte instruction having op-code is CCH. These instructions are inserted into the program so that when the processor reaches there, then it stops the normal execution of program and follows the break-point procedure.

Its execution includes the following steps −

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 3×4 = 0000CH
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

INTO - Interrupt on overflow instruction

It is a 1-byte instruction and their mnemonic **INTO**. The op-code for this instruction is CEH. As the name suggests it is a conditional interrupt instruction, i.e. it is active only when the overflow flag is set to 1 and branches to the interrupt handler whose interrupt type number is 4. If the overflow flag is reset then, the execution continues to the next instruction.

Its execution includes the following steps −

- Flag register values are pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of word location 4×4 = 00010H
- CS is loaded from the contents of the next word location.
- Interrupt flag and Trap flag are reset to 0

15.4 REFERENCES

1. https://www.eeeguide.com/interrupt-structure-of-8086/
2. https://www.tutorialspoint.com/microprocessor/microprocessor_8086_interrupts.htm
3. https://www.geeksforgeeks.org/interrupts-in-8086-microprocessor/
4. https://www.lkouniv.ac.in/site/writereaddata/siteContent/20200410131017434 7kalpana_singh_engg_Microprocessor_interrupt.pdf

CHAPTER 16

8254- PROGRAMMABLE TIMER/COUNTER

Mr. Navdeep Randhawa[1] Mr. Manik Dhiman[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

In a microprocessor Based system, we come across two important modes, i.e **timer -> to provide delay** and **counter -> to count incoming pulses**. Presently, the way we implement timer/counter in 8085 based system has the following drawbacks:

1. **Delay:** The 8085 can provide delays of any value, but it uses software to implement the delay. The instructions are arranged to waste time. The main disadvantage of this scheme is 8085 is executing some instructions; it means that it is busy in doing work.

2. **Counter:** The 8085 can count number of pulses arriving at port. To implement this 8085 goes on checking port, if it is active it increasing counter by 1 and again goes on checking port. The same disadvantage, 8085 will have to execute instructions.

In a large system, where 8085 wastage time is critical, a separate time IC 8254 can be used. 8254 programmable interval timer consists of 3 identical 16-bit counters. These counters can work as counters or can provide accurate time delays. To operate as a counter, a 16-bit count is loaded and the desired mode of operation is selected. The counters will work independently and generate the desired output. Now the 8085 Microprocessor job is to initialize and load counters.

16.1 DIFFERENCE BETWEEN 8253 AND 8254

The following table differentiates the features of 8253 and 8254 −

8253	**8254**
Its operating frequency is 0 - 2.6 MHz	Its operating frequency is 0 - 10 MHz
It uses N-MOS technology	It uses H-MOS technology
Read-Back command is not available	Read-Back command is available
Reads and writes of the same counter cannot be interleaved.	Reads and writes of the same counter can be interleaved.

16.2 FEATURES OF 8253 / 54

The most prominent features of 8253/54 are as follows −

- It has three independent 16-bit down counters.
- It can handle inputs from DC to 10 MHz.
- These three counters can be programmed for either binary or BCD count.
- It is compatible with almost all microprocessors.
- 8254 has a powerful command called READ BACK command, which allows the user to check the count value, the programmed mode, the current mode, and the current status of the counter.

16.3 BLOCK DIAGRAM OF 8254 PROGRAMMABLE INTERVAL TIMER

The block diagram of 8254 Programmable Interval Times is as shown in the figure 16.1.

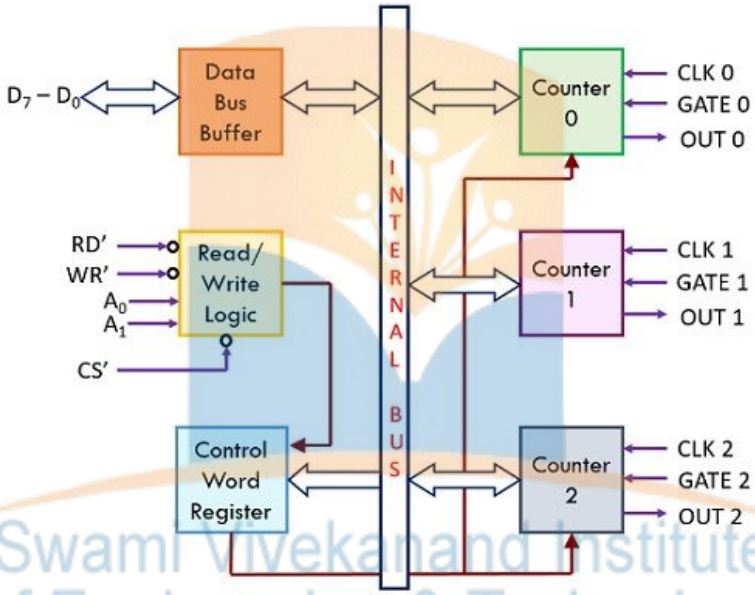


Fig. 16.1 Block Diagram of 8254 Programmable Interval Timer

It includes a data bus buffer, read/write logic, control word register, and counters.

1. Data Bus Buffer:

- It is tristate, bidirectional 8 bit data bus buffer.
- It is used to interface 8254 data bus with system data bus.
- It is internally connected to internal data bus and its outer pins D0-D7 are connected to system data bus. The direction of data buffer is decided by read and writes control signals.

2. Read/Write Logic:

- This block accepts inputs from system control bus and address bus.
- In I/O mapped I/O, the signals *RD'* and *WR'* are connected to *IOR'* and *IOW'*.
- In memory mapped I/O *RD'* and *WR'*, are connected to *MEMR'* and *MEMW'*.
- A_0 and A_1 are directly connected to address lines A_0 and A_1.
- *CS'* is connected to address decoder.
- The 8254 operation/selection is enables/disabled by *CS'* signal. A_0, A_1 selects a specific part *WR'*, *RD'* decides writing data to 8254 or reading data from 8254.
- The control word registers and the counters are selected according to the signals on line A_0 and A_1.

A_1	A0	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control word register

3. Control Word Register:

- This register of 8254 programmable interval timer gets selected when A_0=1 and A_1=1.
- It is used to specify the BCD or binary counter to be used, its mode of operation and the data transfer to be used i.e. read or write the data bytes.
- If the CPU performs a write operation, the data is stored in the control word register and is preferred to as control word. It is used to define counter operation.
- The data can only be written into control word register, no read operation is allowed. Status information is available with the help of read back command.

4. Counters:

- There are three independent, 16-bit down counters.
- They can be programmed separately through control word register to decide mode of counter.
- Each counter is having 2 inputs viz. CLK and GATE.
- CLK is used as an input to counter and GATE is used to control the counter.
- The counters give output on OUT pin.
- The loaded count value in counter will be decremented by counter at each clock input pulse. The programmer can read counter without disturbing counter operation.

16.4 PROGRAMMING OF 8254

- At the time of programming the timer it is necessary that each individual counter of 8254 is to be programmed separately using control word and count value.
- The figure 16.2 shows the format of the control word:

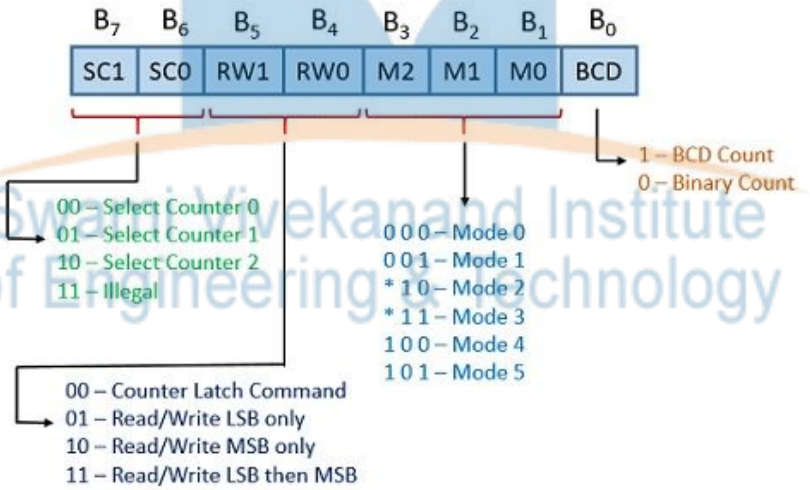


Fig. 16.2 Format of control word of 8254 timer

- Here in the control word format, B_0 selects the BCD or binary count while B_1, B_2, and B_3 are used to select one of the modes of operation for the counter which bits B_6 and B_7 specify. For the operation to take place, the control word is needed to be sent for each separate counter at the same control address register. The identification of the control word of the particular counter is done using bits B_6 and B_7 of the control word format.

- The read/ write operations are performed using bits B_4 and B_5. Through these bits, the 16-bit count value is read and written in an orderly sequence. Also, each time the read operation will take place the count value is to be read by stopping the counter. Here basically the count value is latched to an internal latch present at the output of each counter before the read operation.
- Unlike 8253, there is a separate read-back control word format in 8254 through which the count value is latched. The figure 16.3 shows the format of the read-back control word of 8254:

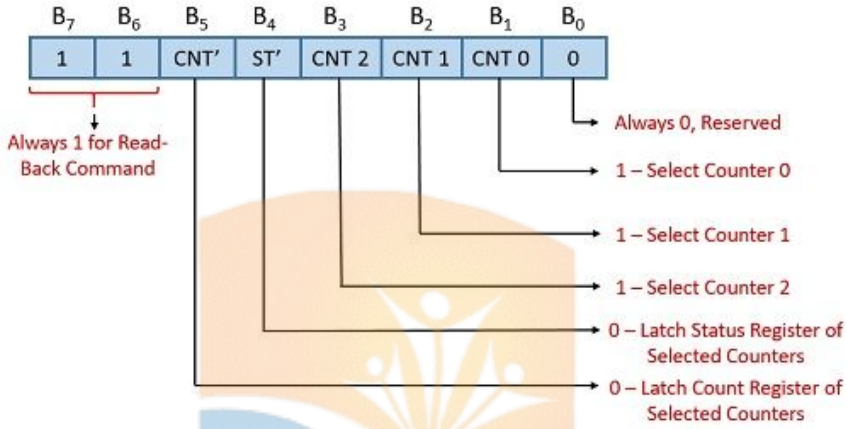


Fig. 16.3 Format of read-back control word of 8254 timer

- In order to latch the count value before the read operation takes place the control word is sent to the same control register address. The control word is identified using the bit values of B_6 and B_7 by the control register. By sending one control word, one or all the counters can be latched by the read-back control word. The status register is also latched to the output latch of the counters through the control word, this allows reading the status register by the respective address of the counter.
- It is to be noted here that at any specific instant either one can latch the count value by programming the bit B_5 as zero or latch the status register by programming the bit B_4 as zero.
- Figure 16.4 shows the format of the status register of each counter:

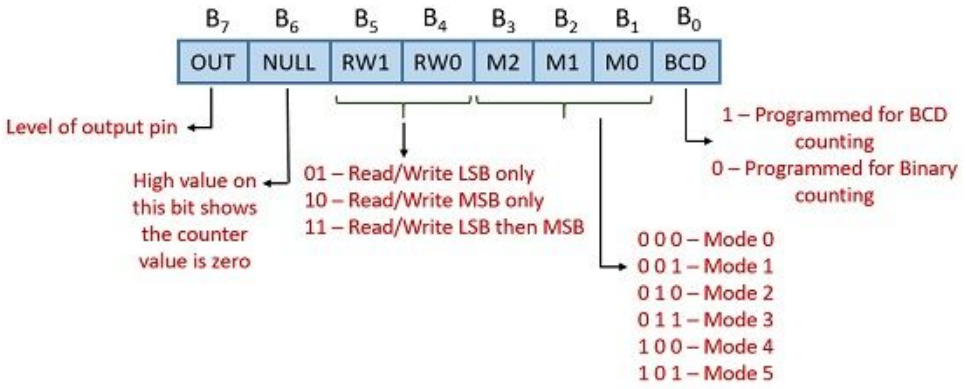


Fig. 16.4 Format of the status register of each counter

- The programmed status of the counter is judged by the status word of the counter and through this one can also check whether the count value has become zero or not.

16.5 PIN DESCRIPTION OF 8254

- The figure 16.5 shows the representation of 24 pins IC packed in DIP:

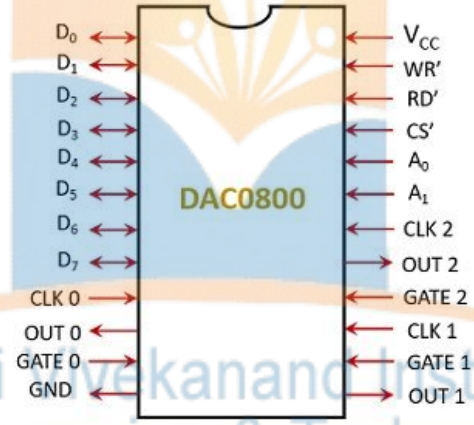


Fig. 16.5 Pin diagram of 8254

- Here D_0 to D_7 represents the bidirectional 3 states data bus lines used for I/O operations. V_{cc} represents the power supply connection of +5V while GND corresponds to ground. There are CLK 0 and OUT 0 which is used for clock input of counter 0 and output of counter 0. Similarly, CLK 1 and CLK 2 are the clock inputs of counter and 2 whereas OUT 1 and OUT 2 are outputs. Other than these GATE 0 and GATE 1 pins correspond to the gate input of counter 0 and counter 1 whereas OUT 0 and OUT 1 are the outputs of counter 0 and 1.
- A_0 and A_1 are the address lines by which selection of one of the three counters or the control word register is done for the read/write operation to take place.

A_1	A_0	Device Selected
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Register

- WR' represents the write control pin which is kept at a low signal during the CPU write operations. RD' is for read control and is low at the time when the CPU performs the read operation. Another crucial pin is the chip select denoted by CS' and whenever it is low then the timer responds to the read and write signals otherwise such signals are ignored.

16.6 APPLICATIONS

1. To generate an accurate time delay
2. As an event counter
3. Square wave generator
4. Rate generator
5. Digital one shot

16.7 REFERENCES

1. https://www.geeksforgeeks.org/microprocessor-8254-programmable-interval-timer/
2. https://easyelectronics.co.in/8254-programmable-interval-timer/
3. https://electronicsdesk.com/programmable-interval-timer.html
4. https://www.tutorialspoint.com/microprocessor/microprocessor_intel_8253_programmable_interval_timer.htm

CHAPTER 17

8259-PRIORITY INTERRUPT CONTROLLER

Mr. Hardeep Singh[1] Ms. Roop Shikha[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

The 8259 is known as the Programmable Interrupt Controller (PIC) microprocessor. In 8085 and 8086 there are five hardware interrupts and two hardware interrupts respectively. But adding 8259, we can increase the interrupt handling capability. This chip combines the multi-interrupt input source to single interrupt output. This provides 8-interrupts from IR0 to IR7.

17.1 NEED OF PROGRAMMABLE INTERRUPT CONTROLLER

We know whenever an interrupt occurs then the microprocessor suspends the current program and switches to the Interrupt Service Routine (ISR).

We know 8085 has 5 interrupts, which are: Trap, RST7.5, RST6.5, RST5.5 and INTR.

Among all these, only INTR is a non-vectored type of interrupt and reset are vectored interrupts.

We know vectored interrupts are those interrupts whose ISR address is known to the processor. Or we can say in case of vectored interrupts, the processor holds the address of the memory location where ISR is stored.

But in case of non-vectored interrupts, the processor has to reach the ISR but it does not hold the address of ISR. So, in this case, the interrupt generating device provides the ISR address to the microprocessor.

An 8085 has 5 major interrupts for which a fixed number of lines are present in the chip. But there are many devices connected to a processor. So, for such a case the processor must have more number of lines to handle several interrupts.

But it is not practically possible to increase the number of lines each time with the increase in the number of interrupts.

So, to overcome this problem 8259 PIC chip is used. 8259 allows the combining of multiple interrupts and providing them to the processor based on priority through a common line.

As we have already discussed that the processor holds the address of ISR in case of vectored interrupts. So, it is not possible to combine a non-vectored interrupt with a vectored one.

Therefore, 8259 is used to combine various interrupts which are non-vectored in nature.

Also, suppose in some way or the other, two devices generate interrupt simultaneously through a common line without the involvement of 8259.

So, the processor gets two INTR signals at the same time but how does the processor get to know that from where the interrupt is generating and where to send the INTA in order to have the ISR address.

This shows the necessity of 8259. The programmable interrupt controller tells the microprocessor about the interrupt. Basically the external devices initially interrupt the 8259 and further the 8259 interrupts the microprocessor.

17.2 FEATURES OF 8259

- It is an LSI chip that manages 8 levels of interrupts i.e. it is used to implement 8 level interrupt systems.
- It can be cascaded in a master-slave configuration to handle up to 64 levels of interrupts.
- It can identify the interrupting device.
- It can resolve the priority of interrupt requests i.e. it does not require any extra
- It can be operated in various priority modes such as fixed priority and rotating priority.
- The interrupt requests are individually maskable.
- It provides an 8-bit vector number as interrupt information.
- It does not require a clock signal.
- It can be used in polled as well as interrupt modes.
- The starting address of the vector number is programmable.
- It can be used in buffered mode. (Buffered mode is applicable for multiprocessor systems).

17.3 ARCHITECTURE OF 8259

The figure 17.1 shows the architectural representation of 8259 programmable interrupt controller.

It has an 8-bit of data bus. As we have already discussed that 8259 never services the interrupt, it simply forwards the interrupts to the microprocessor.

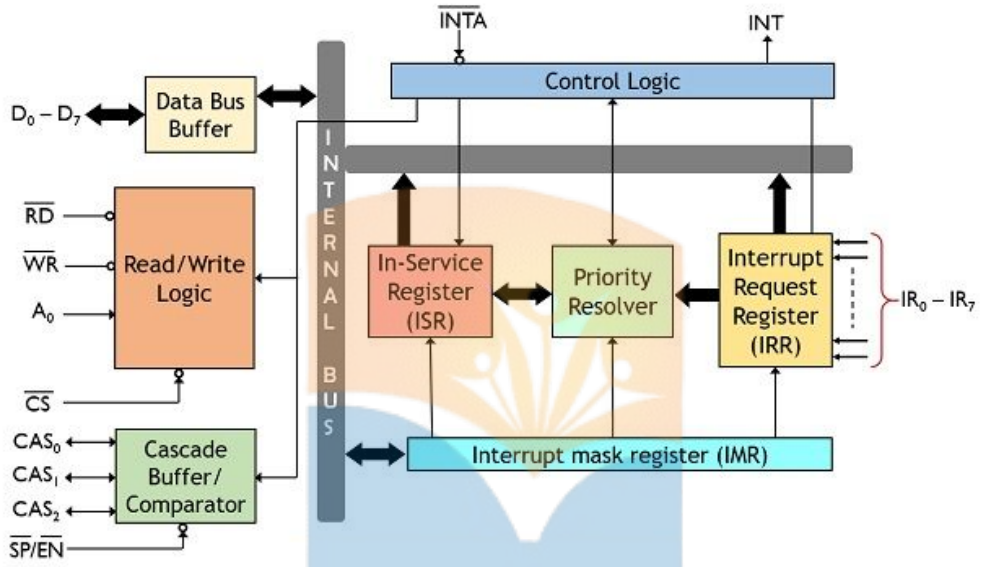


Fig. 17.1 Architecture of 8259 PIC

Thus, the above architecture has different units those combinely functions to increase the interrupts handled by the processor.

Let us understand the operation performed by each unit in detail:

1. Data Bus Buffer: 8259 has tri-stated bidirectional 8-bit data bus buffer (i.e., D_0 to D_7) that interfaces with the internal bus of the processor. The 8085 microprocessor sends/ receives control or status words to / from the 8259 using data bus buffer.

2. Read/ Write Logic: This unit is responsible for controlling the internal read-write operations of the system. It holds initialization command word register and operation command word register inside which various control formats exist that are needed for the device operation.

RD, WR, A_0 and CS are the pins that are associated with this unit. Basically, these pins are used by the processor for read and write operations.

A low signal at CS i.e., chip select shows that now the communication has been set up between the processor and 8259.

3. Control Logic: This unit is the heart of the architecture of 8259. It controls the overall operation of the system by sending the INTR signal to the processor whenever an interrupt request is generated.

Also, it receives INTA signal by the processor when microprocessor demands for the address of the interrupt service routine. The control logic is responsible for sending the address of the desired interrupt service routine through the data bus.

4. Interrupt request register (IRR): This unit stores the interrupt requests generated by the peripheral devices. We know that 8259 has 8 interrupt request pins (i.e., IR_0 to IR_7). So, the unit can store 8 interrupt requests that are requesting the service from the processor.

5. Priority Resolver: This logic unit decides that among all the interrupt request present in the IRR which holds the highest priority and needs to be executed first.

Suppose at the time of servicing an interrupt, another incoming interrupt request gets generated then that request will be ignored as the one in-service is holding the highest priority.

But in case the incoming request has greater priority than the one which is being in current execution then that respective bit will be set in ISR and INTR signal is sent to the microprocessor.

This simply means that only the interrupt holding the highest priority will be forwarded by the 8259 to the processor.

6. In-service register: Here the name of the unit is itself indicating the operation performed by it. This register unit stores the interrupts which are currently being executed by the processor.

The priority resolver sets each bit of ISR and after getting interrupt word command by the processor, the bits get reset. As the processor holds the ability to directly read the status of in-service register.

7. Interrupt mask register: This register unit holds the masking bit of those interrupts which are to be masked. Through operation command word (OCW) the processor sends the required information and programs the interrupt mask register.

8. Cascade buffer/comparator: As we have already discussed that by cascading multiple 8259, the number of interrupts handled by 8259 can be expanded up to 64. The unit allows the comparison of IDs of different 8259s cascaded together.

It permits the operation of the system in two modes: **master mode** and **slave mode**.

In the master mode of operation, it acts as a cascaded buffer. Whereas in slave mode, this unit acts as a comparator.

Among the various cascaded 8259, one 8259 directly handles the interrupts by forming a connection with the processor and it is known to be master 8259. While the other 8259s that interrupts the master 8259 are known as slave 8259.

Each of the 8259s can be separately programmed as all of them holds a specific address. The cascading pins of the master 8259, CAS0, CAS1 and CAS2 forms connection with the corresponding pins of slave 8259s.

For the slave devices, these pins act as input pins while for a master device these acts as output pins. An active-low signal at SP/EN for a device shows that it is operating in slave mode.

In this way, a programmable interrupt controller operates.

17.4 REFERENCES

1. https://www.tutorialspoint.com/8259-pic-microprocessor
2. https://electronicsdesk.com/8259-programmable-interrupt-controller.html
3. https://easyelectronics.co.in/8259a-programmable-interrupt-controller/
4. https://www.tutorialspoint.com/8259-pic-microprocessor

CHAPTER 18

PIN CONFIGURATION OF 8259 PIC

Mr. Harjinder Singh[1] Ms. Neha Garg[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

Programmable interrupt controllers are used to enhance the number of interrupts of a microprocessor. 8259 is a programmable interrupt controller which shows compatibility with 8085 microprocessor.

It is also known as a **priority interrupt controller** and was designed by **Intel** to increase the interrupt handling ability of the microprocessor. An 8259 PIC never services an interrupt; it simply forwards the interrupt to the processor for the execution of interrupt service routine.

18.1 THE PIN CONFIGURATION OF 8259 PIC

The pin configuration of 8259 PIC (Programmable Interrupt Controller) is as shown in the figure 18.1.

Fig. 18.1 Pin diagram of 8259 PIC

D_0-D_7	I/O	Data Bus
RD'	I	I/O read
WR'	I	I/O write
CS'	I	Chip select
INT	O	Interrupt request (To 8085)
INTA'	I	Interrupt acknowledge (From 8085)
A_0	I	Address line
IR_0-IR_7	I	Interrupt request (To 8085)
SP'/ EN'	I/O	Slave program / Enable buffer
CAS0-CAS2	I/O	Cascade lines
Symbol	**Description**	
D_0-D_7	It is an active low-control input line. It is used to write data into registers. it is connected to *IOWR'* or *MEMWR'* of the system bus.	
RD'	It is an active low-control input line. It is used to read the contents of the internal registers. it is connected to *IOR'* or *MEMR'* of the	

	system bus.
WR'	It is an interrupt acknowledge input line. This signal is generated by the microprocessor. The 8259A accepts two *INTA'* pulses to release a one-byte vector number. The 8259 A does not work without *INTA'* pulses in vectored mode.
A_0	It is an address input line. it is used to select the appropriate control register. It is connected to one of the address lines of the system address bus.
CS'	It is an active low-chip select input line. it is used to select the 8259 A chip. This signal is generated by the address decoder.
CAS0-CAS2	These are bi-directional 3-bit cascade lines. These lines are used in cascade mode only. In master mode, these lines function as output lines. In this mode, the pIC places a 3-bit slave identification number on cascade lines.
SP'/ EN'	It is an active low bi-directional control line. In non-buffered mode, it functions as *SP'* input line. In this mode, *SP'* is used to distinguish between master and slave PICs. In buffered mode, it functions as an *EN'* output line. In this mode, it is used to enable data buffers.
INT	It is an interrupt output line. it goes high whenever a valid interrupt request is activated. It must be connected to the INTR input of the microprocessor.
INTA'	It is an interrupt acknowledge input line. This signal is generated by the microprocessor. The 8259A accepts two *INTA'* pulses to release one-byte vector number. The 8259 A does not work without *INTA'* pulses in vectored mode.

IR_0-IR_7	These are asynchronous interrupt request input lines. These signals are generated by peripherals. They can be used whether in edge-triggered or level-triggered mode. The IR input should make low to high transition n level as well as edge-triggered mode.

18.2 PRIORITY MODES OF 8259A PROGRAMMABLE INTERRUPT CONTROLLER

The various priority modes of 8259A are:

1. Fully nested mode
2. Special fully nested mode
3. Rotating Priority mode
    - The rotating priority mode can be set as:
        1. Automatic rotation
        2. Specific rotation
4. Special masked Mode

18.3 ADVANTAGES

Interrupt Management: The 8259 PIC is designed to handle interrupts efficiently and effectively, allowing for faster and more reliable processing of interrupts in a system.

Flexibility: The 8259 PIC is programmable, meaning that it can be customized to suit the specific needs of a given system, including the number and type of interrupts that need to be managed.

Compatibility: The 8259 PIC is compatible with a wide range of microprocessors, making it a popular choice for managing interrupts in many different systems.

Multiple Interrupt Inputs: The 8259 PIC can manage up to 8 interrupt inputs, allowing for the management of complex systems with multiple devices.

Ease of Use: The 8259 PIC includes simple interface pins and registers, making it relatively easy to use and program.

18.4 DISADVANTAGES

Cost: While the 8259 PIC is relatively affordable, it does add cost to a system, particularly if multiple PICs are required.

Limited Number of Interrupts: The 8259 PIC can manage up to 8 interrupt inputs, which may be insufficient for some applications.

Complex Programming: Although the interface pins and registers of the 8259 PIC are relatively simple, programming the 8259 can be complex, requiring careful attention to interrupt prioritization and other parameters.

Limited Functionality: While the 8259 PIC is a useful peripheral for interrupt management, it does not include more advanced features, such as DMA (direct memory access) or advanced error correction.

18.5 REFERENCES

1. https://www.geeksforgeeks.org/8259-pic-microprocessor/
2. https://electronicsdesk.com/8259-programmable-interrupt-controller.html
3. https://easyelectronics.co.in/8259a-programmable-interrupt-controller/

CHAPTER 19

8255 MICROPROCESSOR: ARCHITECTURE AND WORKING

Ms. Saneha[1] Ms. Yukti Gupta[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

Actually connecting I/O devices with the data bus of the processor is not possible directly. So in its place, there must be some device to which I/O ports must be there for connecting I/O devices like 8255 microprocessor. This processor is from the Family of MCS-85 which Intel designed and it can be used with an 8086 & 8085 microprocessor. The 8255 is a Programmable peripheral interface device that is used to achieve the basic communication method between the microprocessor & machines. It is a peripheral device used for a machine that is programmed to perform as an interface. This 8255 PPI is an interface between the microprocessors and the I/O devices.

19.1 WHAT IS AN 8255 MICROPROCESSOR?

8255 microprocessor is a very popularly used programmable peripheral interface chip or PPI chip. The function of the 8255 microprocessor is to transmit data in various conditions from simple I/O to interrupt I/O. This microprocessor is also designed for interfacing the CPU with its external world like ADC, keyboard, DAC, etc. This microprocessor is economical, functional, and flexible although is a little complex, so it can be used with any microprocessor. This microprocessor is used to connect peripheral devices & also for interfacing. So this peripheral device is also called an I/O device because the I/O ports of this microprocessor are used for connecting I/O devices. This processor includes three 8-bit bidirectional I/O ports which can be configured based on necessity.

19.2 FEATURES

- It is a programmable parallel I/O device.
- It contains 24 programmable I/O pins arranged as 2-8 bit ports and 2-4 bit ports.
- It has 3, 8-bit ports: Port A, Port B, and Port C, which are arranged in two groupss of 12 pins.
- Fully compatible with Intel microprocessor families.
- TTL is compatible.

- Direct bit set/reset capability is available for port C.
- Improved DC driving capability.
- It can operate in 3 Modes:
    1. Mode 0: Simple I/O
    2. Mode 1: Strobed I/O
    3. Mode 2: Strobed bi-directional I/O

19.3 ARCHITECTURE OF 8255 PPI

The figure 19.1 represents the architectural representation of 8255 PPI.

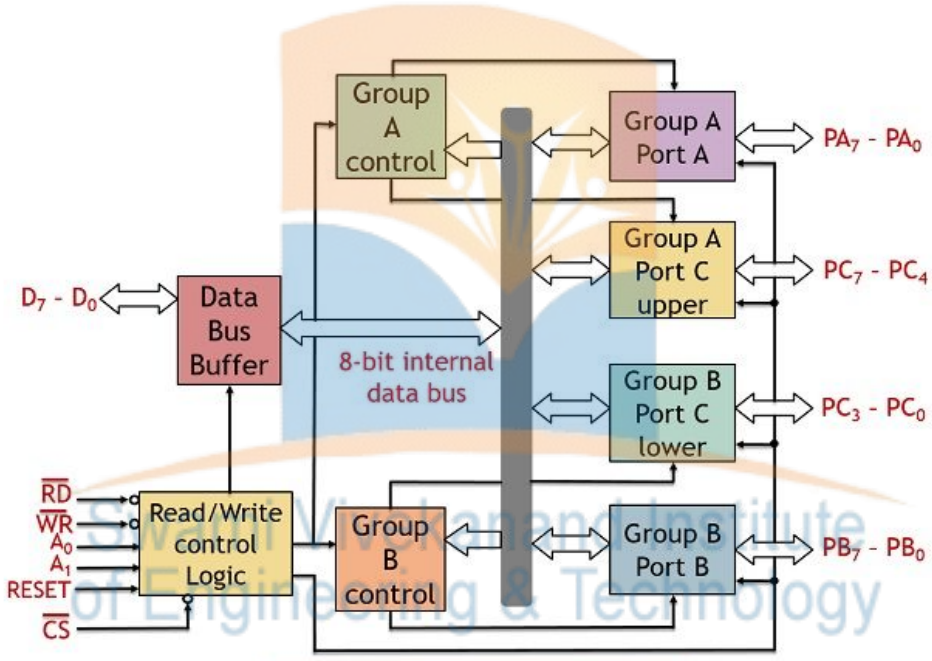


Fig. 19.1 Architecture of 8255 PPI

Let us understand the operation performed by each unit separately.

Data bus buffer: It is used to connect the internal bus of 8255 with the system bus so as to establish proper interfacing between the two. The data bus buffer allows the read/write operation to be performed from/to the CPU.

The buffer allows the passing of data from ports or control register to CPU in case of write operation and from CPU to ports or status register in case of read operation.

Read/ Write control logic: This unit manages the internal operations of the system. This unit holds the ability to control the transfer of data and control or status words both internally and externally.

Whenever there exists a need for data fetch then it accepts the address provided by the processor through the bus and immediately generates command to the 2 control groups for the particular operation.

Group A and Group B control: These two groups are handled by the CPU and functions according to the command generated by the CPU. The CPU sends control words to the group A and group B control and they in turn sends the appropriate command to their respective port.

As we have discussed that group A has the access of the port A and higher order bits of port C. While group B controls port B with the lower order bits of port C.

CS: It stands for chip select. A low signal at this pin shows the enabling of communication between the 8255 and the processor. More specifically we can say that the data transfer operation gets enabled by an active low signal at this pin.

RD – It is the signal used for read operation. A low signal at this pin shows that CPU is performing read operation at the ports or status word. Or we can say that 8255 is providing data or information to the CPU through data buffer.

WR – It shows write operation. A low signal at this pin allows the CPU to perform write operation over the ports or control register of 8255 using the data bus buffer.

A_0 and A_1: These are basically used to select the desired port among all the ports of the 8255 and it does so by forming conjunction with RD and WR. It forms connection with the LSB of the address bus.

The table below shows the operation of the control signals:

A_1	A_0	RD'	WR'	CS'	Input/Output Operation
0	0	0	1	0	Port A - Data Bus
0	1	0	1	0	Port B - Data Bus
1	0	0	1	0	Port C - Data Bus
0	0	1	0	0	Data Bus - Port A
0	1	1	0	0	Data Bus - Port B

1	0	1	0	0	Data Bus - Port C
1	1	1	0	0	Data Bus - Control register

Reset: It is an active high signal that shows the resetting of the PPI. A high signal at this pin clears the control registers and the ports are set in the input mode.

Initializing the ports to input mode is done to prevent circuit breakdown. As in case of reset condition, if the ports are initialized to output mode then there exist chances of destruction of 8255 along with the processor.

19.4 OPERATING MODES

- The 8255 IC provides one control word register.
- It is selected when A0=1, A1=1, *CS'* =0 and *WR'* = 0.
- The read operation is not allowed for the control register.
- The bit pattern loaded in the control word register specifies an I/O function for each port and the mode of operation in which the ports are to be used.
- There are 2 different control word formats that specify 2 basic modes:
  1. BSR-Bit set-reset mode
  2. I/O mode
- The two basic modes are selected by the D7 bit of the control register. When D7=1, it is an I/O mode, and when D_7 =0, it is a BSR mode.

1. BSR Mode

- The BSR mode is a port C bit set/reset mode.
- The individual bit of port C can be set or reset by writing the control word in the control register.
- The control word format of BSR mode is as shown in the figure 19.2.
- The pin of port C is selected using bit select bits and set or reset is decided by bit S/R.
- The BSR mode affects only one bit of port C at a time.
- The bit set using BSR mode remains set unless and until you change the bit. So to set any bit of port C, the bit pattern is loaded in the control register.

- If a BSR mode is selected, it will not affect the I/O mode.

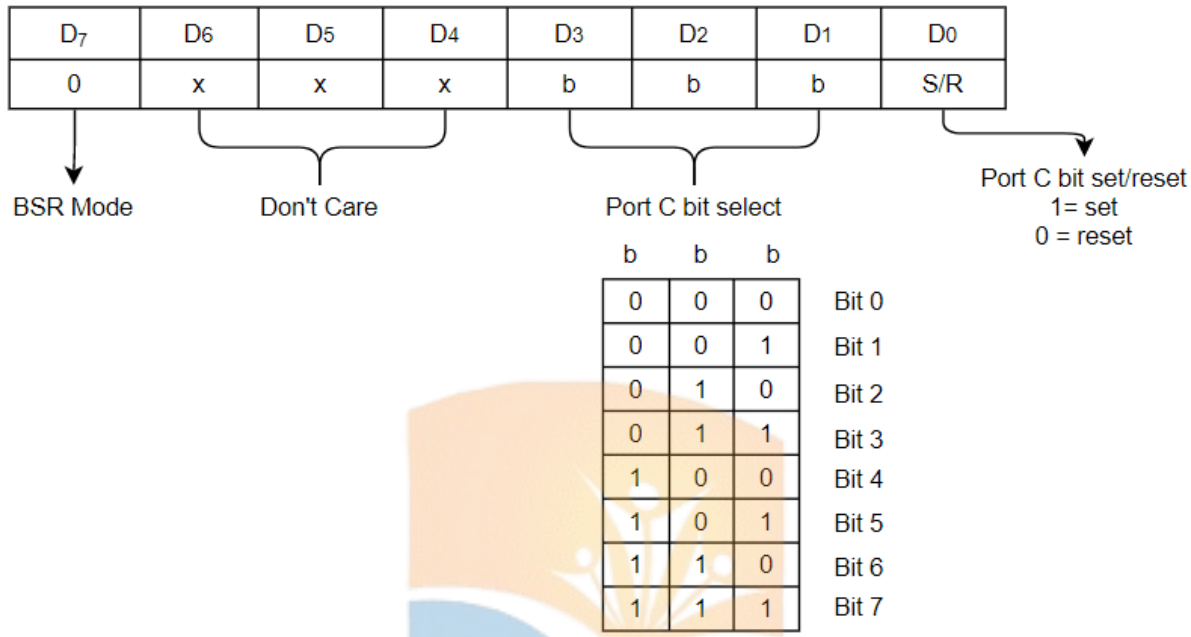


Fig. 19.2 Control word format of BSR mode

2. I/O Modes

There are three I/O modes of operation:

1. Mode 0: Basic I/O
2. Mode 1: Strobed I/O
3. Mode 2: Bidirectional I/O

I/O modes are programmed using a control register. The function of each bit is as follows:

1. **D_7:** When the bit D7=1 then I/O mode is selected, if D7=0 then BSR mode is selected. The function of bits D0 to D6 is dependent on mode.
2. **D_6 and D_5:** In I/O mode the bit D6 and D5 specify the different I/O modes for group A.
3. **D_4 and D_3:** In I/O mode the bits D4 and D3 select the port function for groups A. If these bits =1 the respective port specified is used as the input port. But if bit=0, the port is used as the output port.
4. **D_2:** In I/O mode the bit D2 specifies the different I/O modes for group B.

5. **D_1 and D_0:** In I/O mode the bits D1 and D0 select the port function for group B. If these bits =1 the respective port specified is used as the input port. But if bit=0, the port is used as the output port.

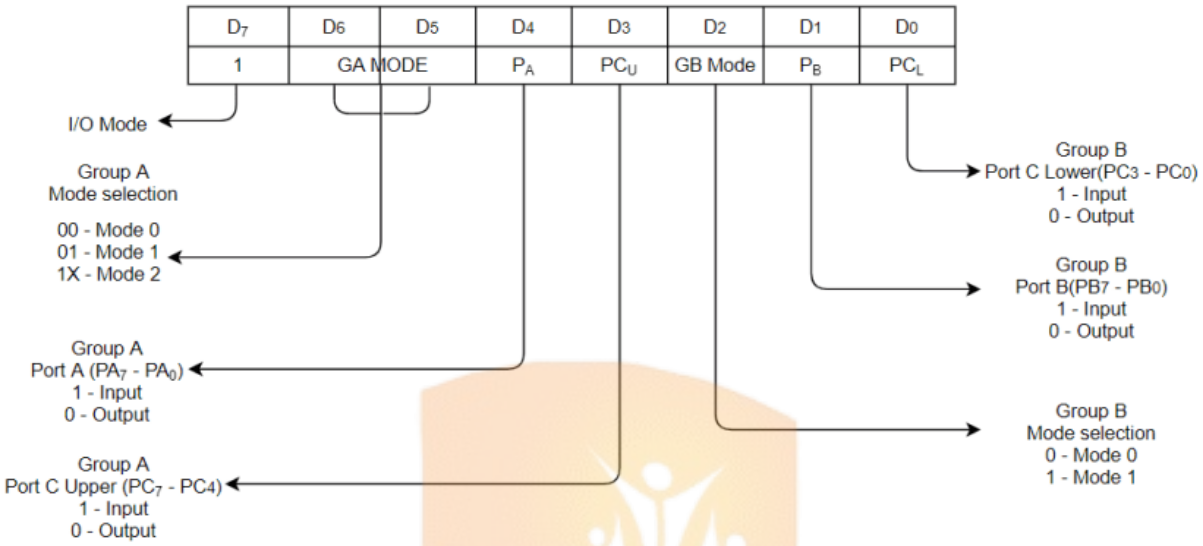


Fig. 19.3 Control register format

19.5 REFERENCES

1. https://www.tutorialspoint.com/microprocessor/microprocessor_intel_8255a_programmable_peripheral_interface.htm
2. https://www.brainkart.com/article/Parallel-Communication-Interface--8255-Programmable-Peripheral-Interface-and-Interfacing_7872/
3. https://easyelectronics.co.in/8255-ppi/
4. https://www.elprocus.com/8255-microprocessor/
5. https://electronicsdesk.com/8255-ppi.html
6. https://www.eeeguide.com/features-of-8255-microprocessor/

CHAPTER 20

PIN DIAGRAM OF 8255 PPI

Mr. Vikas Zandu[1] Ms. Kulbir Kaur[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

The 8255A is a general purpose programmable I/O device designed to transfer the data from I/O to interrupt I/O under certain conditions as required. It can be used with almost any microprocessor.

It consists of three 8-bit bidirectional I/O ports (24I/O lines) which can be configured as per the requirement.

20.1 PORTS OF 8255A

8255A has three ports, i.e., PORT A, PORT B, and PORT C.

- **Port A** contains one 8-bit output latch/buffer and one 8-bit input buffer.
- **Port B** is similar to PORT A.
- **Port C** can be split into two parts, i.e. PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word.

These three ports are further divided into two groups, i.e. Group A includes PORT A and upper PORT C. Group B includes PORT B and lower PORT C. These two groups can be programmed in three different modes, i.e. the first mode is named as mode 0, the second mode is named as Mode 1 and the third mode is named as Mode 2.

20.2 PIN DIAGRAM OF THE 8255 MICROPROCESSOR

The pin diagram of the 8255 microprocessor is shown in figure 20.1. This microprocessor includes 40-pins like PA7-PA0, PC7-PC0, PC3-PC0, PB0-PB7, RD, WR, CS, A1 & A0, D0-D7 and RESET. These pins are discussed below.

PA7 to PA0 (PortA Pins)

The PA7 to PA0 are Port A data lines pins (1 to 4 & 37 to 40) which are distributed equally on two sides of the top of the microprocessor. These eight port A pins work as either buffered input lines or latched output based on the loaded control word into the control word register.

PB0 to PB7 (Port B Pins)

The PB0 to PB7 from 18 to 25 are the data line pins that carry the port B data.

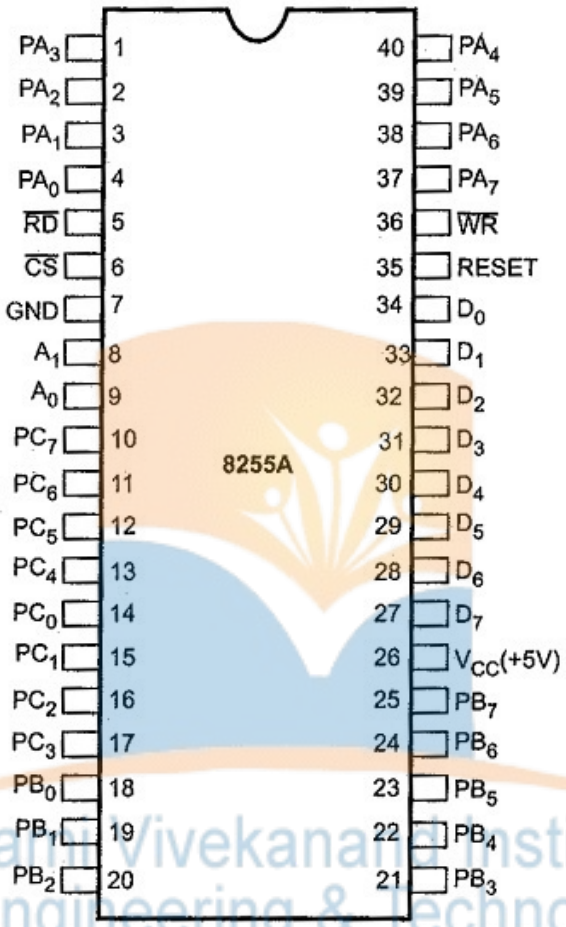


Fig 20.1 Pin configuration of 8255

PC0 to PC7 (Port C Pins)

PC0 to PC7 pins are port C pins which include pin10 to pin17 which carry the port A data bits. From there, pins 10 – pin13 are known as Port C upper pins & pin14 to pin17 are known as lower pins. The pins from these two sections can be used individually to transmit 4 data bits using two separate port C parts.

D0 to D7 (Data bus pins)

These D0 to D7 pins are data I/O lines which include 27-pin to 34-pin. These pins are used to carry the 8-bit binary code and it is utilized to train the entire IC work. These pins are jointly known as the control register/control word which carries the data of the control word.

A0 & A1

A0 and A1 pins at pin8 & pin9 simply make a decision about which port will be preferred for transmitting the data.

If A0 = 0 & A1=0 then Port-A is selected.

If A0 = 0 & A1=1 then Port-B is selected.

If A0 = 1 & A1=0 then Port-C is selected.

If A0 = 1 & A1=1 then the control register is selected.

CS'

The pin6 like CS' is a chip select input pin which is responsible for selecting a chip. A low signal at CS' pin simply allows the communication between the 8255 & the processor which means at this pin, the operation of data transfer gets allowed by an active low signal.

RD'

The pin5 like RD' is a read input pin that puts the chip within the reading mode. A low signal at this RD's pin provides data to the CPU by a data buffer.

WR'

The pin36 like WR' pin is a write input pin that puts the chip within writing mode. So, a low signal at WR' pin simply allows the CPU to execute the write operation above the ports otherwise microprocessor's control register through the data bus buffer.

RESET

The pin35 like the RESET pin resets the whole data available in all the keys to their default values when it is in set mode. It is an active high signal where the high signal at the RESET pin clears the control registers &the ports are placed within the input mode.

GND

The pin7 is a GND pin of IC.

VCC

The pin26 like VCC is the 5V input pin of IC.

20.3 MODES OF OPERATION

8255 has two modes of operation. These are as follows:

Bit Set-Reset mode: When port C is utilized for control or status operation, then by sending an OUT instruction, each individual bit of port C can be set or reset.

I/O mode: As we know that the I/ O mode is sub-classified into 3 modes. So, let us now discuss the 3 modes here.

Mode 0: Input/Output mode

This mode is the simple input output mode of 8255 which allows the programming of each port as either input or output port. The input/output feature of mode 0 includes:

- It does not support handshaking or interrupt capability.
- The input ports are buffered while outputs are latched.

Mode 1: Input/Output with handshaking

Mode 1 of 8255 supports handshaking with the ports programmed as either input or output mode. We know that it is not necessary that all the time the data is transferred between two devices operating at same speed. So, handshaking signals are used to synchronize the data transfer between two devices that operates at different speeds.

The figure 20.2 shows the data transferring between CPU and an output device having different operating speeds:

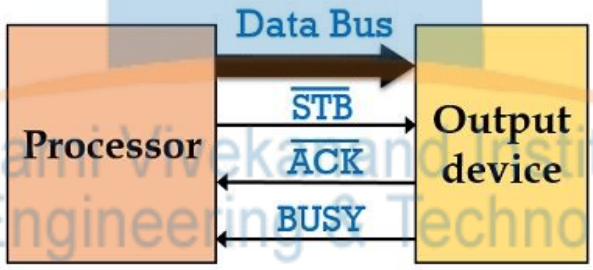


Fig. 20.2 Data transferring between CPU and an output device

- Here STB signal is used to inform the output device that data is available on the data bus by the processor.
- Here port A and port B can be separately configured as either input or output port.
- Both the port utilizes 3-3 lines of port C for handshaking signals. The rest two lines operates as input/output port.
- It supports interrupt logic.
- The data at the input or output ports are latched.

Mode 2: Bidirectional I/O port with handshaking

In this mode, the ports can be utilized for the bidirectional flow of information by handshaking signals. The pins of group A can be programmed to acts as bidirectional data bus and the port C upper ($PC_7 – PC_4$) are used by the handshaking signal. The rest 4 lower port C bits are utilized for I/O operations.

As the data bus exhibits bidirectional nature thus when the peripheral device request for a data input only then the processor load the data in the data bus. Port B can be programmed in mode 0 and 1. And in mode 1 the lower bits of port C of group B are used for handshaking signals.

So, from the above discussion it is clear that how interfacing of peripheral devices is performed with the processor.

20.4 APPLICATIONS OF 8255 PPI

8255 PPI is the most widely used chip for many applications:

- LED / Relay Interface
- Display Interface
- Stepper Motor Interface
- Lift Controller etc.
- Keyboard Interface
- ADC / DAC Interface
- Traffic Signal Controller

20.5 REFERENCES

1. https://www.tutorialspoint.com/microprocessor/microprocessor_intel_8255a_programmable_peripheral_interface.htm
2. https://www.brainkart.com/article/Parallel-Communication-Interface--8255-Programmable-Peripheral-Interface-and-Interfacing_7872/
3. https://easyelectronics.co.in/8255-ppi/
4. https://www.elprocus.com/8255-microprocessor/
5. https://electronicsdesk.com/8255-ppi.html
6. https://www.eeeguide.com/features-of-8255-microprocessor/

CHAPTER 21

KEYBOARD/DISPLAY CONTROLLER – 8279

Ms. Sujata Tondon[1] Ms. Komal Dhiman[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

The **Intel 8279** is a keyboard/display controller that interfaces keyboard and display devices with the microprocessor-based system. Through the use of this controller, processor utilization reduces in correspondence to time-consuming tasks such as keyboard scanning and display refreshing.

Basically, it is a general-purpose controller that performs the function of driving the display unit as well as interfacing the keyboard and the CPU simultaneously.

21.1 Introduction

Intel's 8279 is a dedicated controller designed by Intel that offers simultaneous keyboard and display operations. It provides interfacing for 64-contact keys that are arranged in 8*8 matrix format. Also, through 8279 multiplexed interfacing can be obtained for 7-segment LEDs as well as other popular display devices.

It has a display RAM of **16*8** matrix, however; the arrangement can be organized in dual **16*4** RAM. The RAM is loaded by the CPU however, once data loading is done, the 8279 performs data displaying and refreshing functions.

It works in a way that the keyboard display interface keeps the keyboard under scanning in order to check if any key is pressed. If a pressed key is detected, then the key code of the pressed key is sent to the CPU. Also, after the operation, the data which the CPU forwards it, is transmitted to the display device. These two operations take place within the system in a synchronized manner without disturbing the other ongoing operations of the CPU.

21.2 FEATURES OF INTEL 8279 PROGRAMMABLE KEYBOARD DISPLAY INTERFACE

The Intel 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel microprocessors. The Features of Intel 8279 Programmable Keyboard Display Interface is:

1. It provides a scanned interface to a 64-contact key matrix, with two more keys CONTROL and SHIFT.

2. It provides three input modes for keyboard interface;

- Scanned Keyboard Mode
- Scanned Sensor Matrix Mode
- Strobed Input Mode

3. It has built-in hardware to provide key debounce.

4. It allows key depressions in 2 key lockout or N-key rollover mode, which eliminates software required to implement 2 key lockout and N-key rollover

5. The interrupt output of 8279 can be used to tell CPU that the keypress is This eliminates the need of software polling.

6. Features of Intel 8279 provides 8 byte FIFO RAM to store keycodes. This allows to store 8 key board inputs when CPU is busy in performing his own computation.

7. It provides multiplexed display interface with blanking and inhibit options.

8. It provides sixteen byte display RAM to store display codes for 16 digits, allowing interfacing 16 digits.

9. In auto increment mode, address of display RAM and FIFO RAM is incremented automatically which eliminates extra command after each read/write operation to access successive locations of display RAM and FIFO

10. Features of Intel 8279 provide two output modes for display interface.

- Left Entry (typewriter type)
- Right Entry (calculator type)

11. Simultaneous keyboard and display operation facility allows interleaving keyboard and displaying software.

21.3 METHODS OF INTERFACING KEYBOARD WITH CPU

Mainly there are two ways by which the keyboard can be interfaced with the CPU, these are as follows:

- **Interrupt Mode**: In this mode of operation, the CPU continues to perform its original task until and unless any request is generated to that a key is pressed and is required to be serviced.

- **Polled Mode**: In the polled mode of operation the CPU itself checks for any pressed key in a periodic manner by reading the flags of 8279. So, in this case whenever the CPU gets a signal that shows the pressed key then the requested service is served.

There is a 40 pin IC of 8279 which exists in the Dual In-line Package (DIP).

21.4 BLOCK DIAGRAM OF 8279

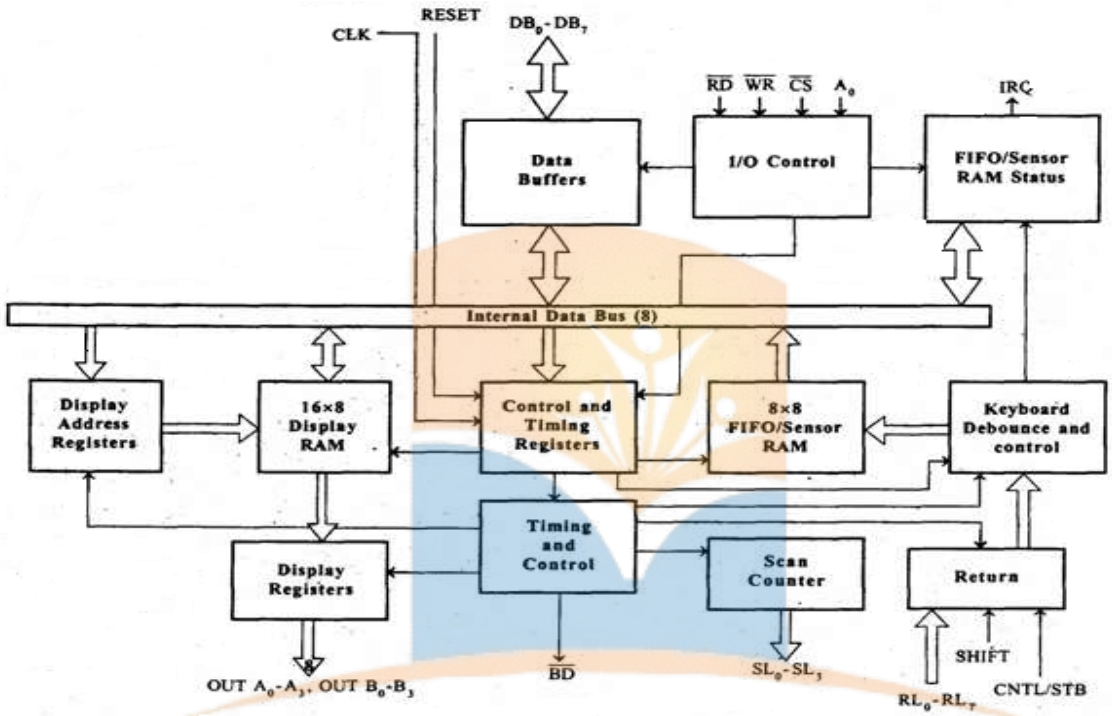


Fig. 21.1 Block Diagram of 8279

I/O Control and Data Buffer

This unit controls the flow of data through the microprocessor. It is enabled only when D is low. Its data buffer interfaces the external bus of the system with the internal bus of the microprocessor. The pins A0, RD, and WR are used for command, status or data read/write operations.

Control and Timing Register and Timing Control

This unit contains registers to store the keyboard, display modes, and other operations as programmed by the CPU. The timing and control unit handles the timings for the operation of the circuit.

Scan Counter

It has two modes i.e. **Encoded mode** and Decoded mode. In the encoded mode, the counter provides the binary count that is to be externally decoded to provide the scan lines for the keyboard and display.

In the **decoded scan mode**, the counter internally decodes the least significant 2 bits and provides a decoded 1 out of 4 scan on SL_0-SL_3.

Return Buffers, Keyboard Debounce, and Control

This unit first scans the key closure row-wise, if found then the keyboard debounce unit debounces the key entry. In case, the same key is detected, then the code of that key is directly transferred to the sensor RAM along with SHIFT & CONTROL key status.

FIFO/Sensor RAM and Status Logic

This unit acts as 8-byte first-in-first-out (FIFO) RAM where the key code of every pressed key is entered into the RAM as per their sequence. The status logic generates an interrupt request after each FIFO read operation till the FIFO gets empty.

In the scanned sensor matrix mode, this unit acts as sensor RAM where its each row is loaded with the status of their corresponding row of sensors into the matrix. When the sensor changes its state, the IRQ line changes to high and interrupts the CPU.

Display Address Registers and Display RAM

This unit consists of display address registers which holds the addresses of the word currently read/written by the CPU to/from the display RAM.

21.5 8279 – PIN DESCRIPTION

The figure 21.2 shows the pin diagram of 8279.

Data Bus Lines, DB_0 - DB_7

These are 8 bidirectional data bus lines used to transfer the data to/from the CPU.

CLK

The clock input is used to generate internal timings required by the microprocessor.

RESET

As the name suggests this pin is used to reset the microprocessor.

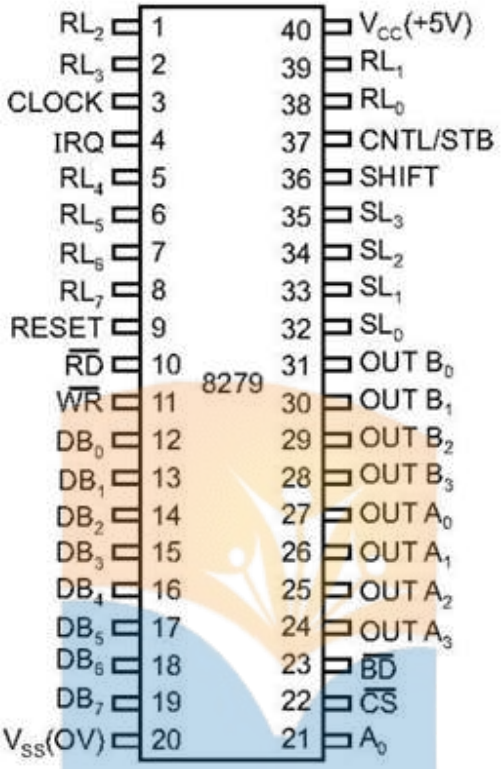


Fig. 21.2 Pin diagram of 8279

CS Chip Select

When this pin is set to low, it allows read/write operations; else this pin should be set to high.

A_0

This pin indicates the transfer of command/status information. When it is low, it indicates the transfer of data.

RD, WR

This Read/Write pin enables the data buffer to send/receive data over the data bus.

IRQ

This interrupt output line goes high when there is data in the FIFO sensor RAM. The interrupt line goes low with each FIFO RAM read operation. However, if the FIFO RAM further contains

any key-code entry to be read by the CPU, this pin again goes high to generate an interrupt to the CPU.

V_{ss}, V_{cc}

These are the ground and power supply lines of the microprocessor.

SL_0 – SL_3

These are the scan lines used to scan the keyboard matrix and display the digits. These lines can be programmed as encoded or decoded, using the mode control register.

RL_0 – RL_7

These are the Return Lines which are connected to one terminal of keys, while the other terminal of the keys is connected to the decoded scan lines. These lines are set to 0 when any key is pressed.

SHIFT

The Shift input line status is stored along with every key code in FIFO in the scanned keyboard mode. Till it is pulled low with a key closure, it is pulled up internally to keep it high

CNTL/STB - CONTROL/STROBED I/P Mode

In the keyboard mode, this line is used as a control input and stored in FIFO on a key closure. The line is a strobe line that enters the data into FIFO RAM, in the strobed input mode. It has an internal pull up. The line is pulled down with a key closure.

BD

It stands for blank display. It is used to blank the display during digit switching.

$OUTA_0$ – $OUTA_3$ and $OUTB_0$ – $OUTB_3$

These are the output ports for two 16x4 or one 16x8 internal display refresh registers. The data from these lines is synchronized with the scan lines to scan the display and the keyboard.

21.6 REFERENCES

1. https://www.tutorialspoint.com/microprocessor/microprocessor_8279_programmable_keyboard.htm
2. https://electronicsdesk.com/keyboard-display-controller-8279.html
3. https://www.eeeguide.com/operating-modes-of-8279/

CHAPTER 22

INTERFACING 8279 WITH 8086 PROCESSOR

Dr. Shashi Jawla[1] Mr. Manik Dhiman[2]

[1]*Associate Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

22.1 8279 PROGRAMMABLE KEYBOARD

The Intel 8279 is a programmable keyboard interfacing device. Data input and display are the integral part of microprocessor kits and microprocessor-based systems.

8279 has two sections namely **keyboard section** and **display section**.

The function of the **keyboard section** is to interface the keyboard which is used as input device for the microprocessor. It can also interface toggle or thumb switches.

The purpose of the **display section** is to drive alphanumeric displays or indicator lights. It is directly connected to the microprocessor bus.

The microprocessor is relieved from the burden of scanning the keyboard or refreshing the display.

Some important Features are:

- Simultaneous keyboard display operations
- Scanned sensor mode
- Scanned keyboard mode
- 8-character keyboard FIFO
- Strobed input entry mode
- 2-key lock out or N-key roll over with contact debounce
- Single 16-charcter display
- Dual 8 or 16 numerical display
- Interrupt output on key entry

- Programmable scan timing and mode programmable from CPU

22.2 INTERFACING A MICROPROCESSOR TO KEYBOARD

When you press a key on your computer, you are activating a switch. There are many different ways of making these switches. An overview of the construction and operation of some of the most common types are:

Mechanical key switches: In mechanical-switch keys, two pieces of metal are pushed together when you press the key. The actual switch elements are often made of a phosphor-bronze alloy with gold platting on the contact areas. The key switch usually contains a spring to return the key to the nonpressed position and perhaps a small piece of foam to help damp out bouncing.

Some mechanical key switches now consist of a molded silicon dome with a small piece of conductive rubber foam short two trace on the printed-circuit board to produce the key pressed signal.

Mechanical switches are relatively inexpensive but they have several disadvantages. First, they suffer from contact bounce. A pressed key may make and break contact several times before it makes solid contact.

Second, the contacts may become oxidized or dirty with age so they no longer make a dependable connection.

Higher- quality mechanical switches typically have a rated life time of about 1 million keystrokes. The silicone dome type typically last 25 million keystrokes.

Membrane key switches: These switches are really a special type of mechanical switches. They consist of a three-layer plastic or rubber sandwich.

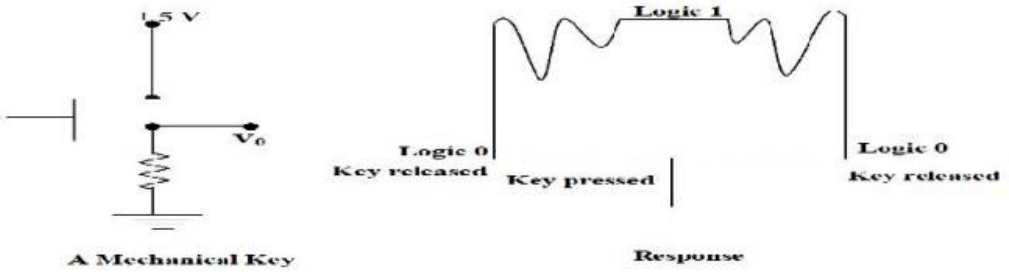


Fig. 22.1 Mechanical key and its response to key press

The top layer has a conductive line of silver ink running under each key position. The bottom layer has a conductive line of silver ink running under each column of keys.

The key board interfaced is a matrix keyboard. This key board is designed with a particular rows and columns. These rows and columns are connected to the microcontroller through its ports of the micro controller 8051. We normally use 8*8 matrix keyboard. So only two ports of 8051 can be easily connected to the rows and columns of the keyboard.

Whenever a key is pressed, a row and a column gets shorted through that pressed key and all the other keys are left open. When a key is pressed only a bit in the port goes high which indicates microcontroller that the key is pressed. By this high on the bit key in the corresponding column is identified.

Once we are sure that one of key in the key board is pressed next our aim is to identify that key. To do this we firstly check for particular row and then we check the corresponding column the key board.

To check the row of the pressed key in the keyboard, one of the rows is made high by making one of bit in the output port of 8051 high. This is done until the row is found out.

Once we get the row next out job is to find out the column of the pressed key. The column is detected by contents in the input ports with the help of a counter. The content of the input port is rotated with carry until the carry bit is set.

The contents of the counter is then compared and displayed in the display. This display is designed using a seven segment display and a BCD to seven segment decoder IC 7447.

22.3 INTERFACING 8279 WITH 8086 PROCESSOR

A typical Hexa keyboard and 7-segment LED display interfacing circuit using 8279 and 8086 based system is shown in figure 22.2.

- The system consists of 16 numbers of hexa-keys and numbers of 7-segment LEDs. The 7- segment LEDs can be used to display eight-digit alphanumeric character.
- The 8279 can be either memory mapped or I/O mapped in the system. In the circuit shown is I/O mapped.
- The address line A1 of the system is used as A0 of 8279.
- The clock signal for 8279 is obtained by dividing the PCLK (peripheral clock) of 8284 by a clock divider circuit.
- The chip select signals, for I/O mapped devices are generated by using a 3-to-8 decoder.
- The address lines A5, A6 and A7 are used as input to decoder.
- The address line A0 and the control signal M/IO (low) are used as enable for decoder. The chip select signal IOCS-3 is used to select 8279.
- The I/O address of the internal devices of 8279 is shown in table.

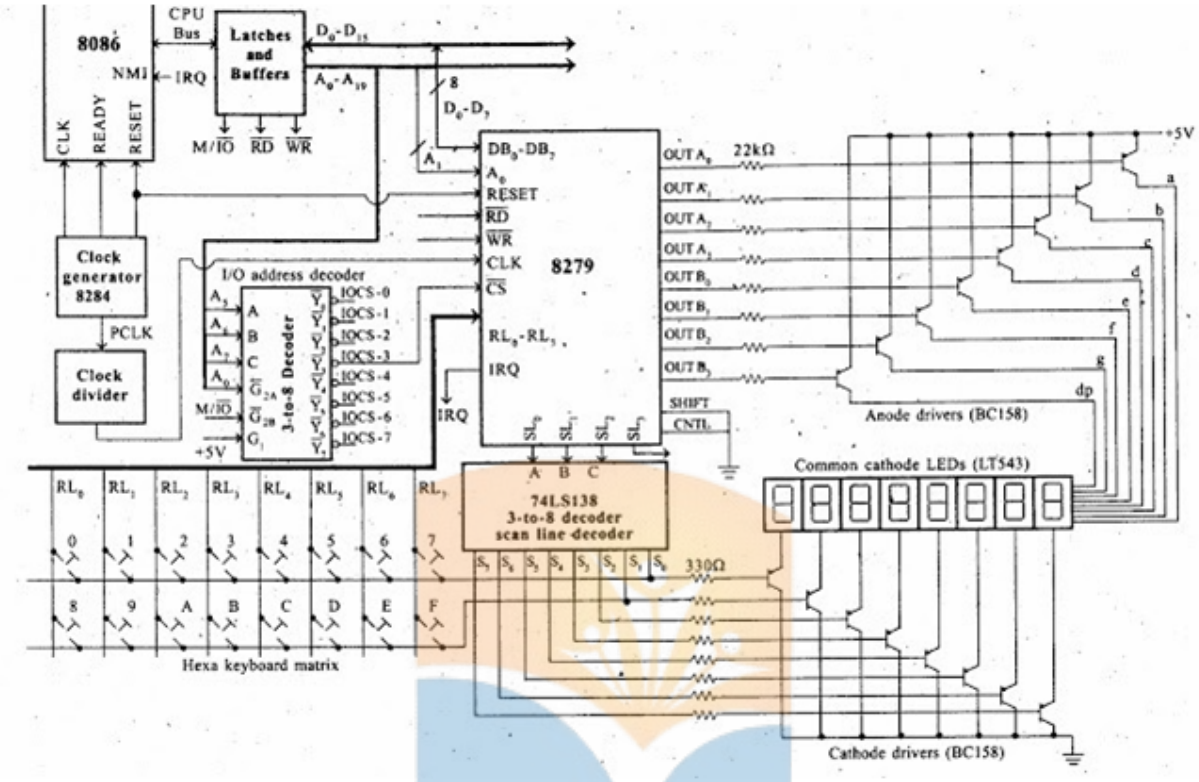


Fig. 22.2 Interfacing 8279 with 8086 Processor

Internal Device	Binary Address								Hexa Address
	Decoder input			Input to address pin of 8279				Decoder enable	
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Data register	0	1	1	x	x	x	0	0	60
Control register	0	1	1	x	x	x	1	0	62

Note : *Don't care "x" is considered as zero.*

- The circuit has 6 numbers of 7-segment LEDs and so the 8279 has to be programmed in encoded scan. (Because in decoded scan, only 4 numbers of 7-segment LEDs can be interfaced).
- In encoded scan the output of scan lines will be binary count. Therefore an external, 3-to-decoder is used to decode the scan lines SL0, SL1 and SL2 of 8279 to produce eight scan lines S0 to S7.
- The decoded scan lines S0 and S1 are common for keyboard and display.

- The decoded scan lines S2 to S5 are used only for display and the decoded scan lines S6 and S7 are not used in the system.
- Anode and Cathode drivers are provided to take care of the current requirement of LEDs.
- The pnp transistors, BC 158 are used as driver transistors.
- The anode drivers are called segment drivers and cathode drivers are called digit drivers.
- The 8279 output the display code for one digit through its output lines (OUT A0 to OUT A3 and OUT B0 to OUT B3) and send a scan code through, SL0- SL3.
- The display code is inverted by segment drivers and sent to segment bus.
- The scan code is decoded by the decoder and turns ON the corresponding digit driver. Now one digit of the display character is displayed. After a small interval (10 milli-second, typical), the display is turned OFF (i.e., display is blanked) and the above process is repeated for next digit. Thus multiplexed display is performed by 8279.
- The keyboard matrix is- formed using the return lines, RL0 to RL3 of 8279 as columns and decoded scan lines S0 and S1 as rows.
- A hexa key is placed at the crossing point of each row and column. A key press will short the row and column. Normally the column and row line will be high.
- During scanning the 8279 will output binary count on SL0 to SL3, which is decoded by decoder to make a row as zero. When a row is zero the 8279 reads the columns. If there is a key press then the corresponding column will be zero.
- If 8279 detects a key press then it waits for debounce time and again read the columns to generate key code.
- In encoded scan keyboard mode, the 8279 stores an 8-bit code for each valid key press. The keycode consist of the binary value of the column and row in which the key is found and the status of shift and control key.
- After a scan time, the next row is made zero and the above process is repeated and so on. Thus 8279 continuously scan the keyboard.

22.4 REFERENCES

1. https://electronicsdesk.com/keyboard-display-controller-8279.html
2. https://www.eeeguide.com/operating-modes-of-8279/
3. https://www.idc-online.com/technical_references/pdfs/electronic_engineering/Interfacing_8279_with_8086_processor.pdf
4. https://www.brainkart.com/article/Interfacing-a-Microprocessor-to-Keyboard_7890/
5. https://www.javatpoint.com/memory-and-io-interfacing

CHAPTER 23

8086 MICROPROCESSOR INTERFACING WITH DAC

Dr. Indu Batra[1] Ms. Tanika Thakur[2]

[1]*Associate Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

23.1 DIGITAL TO ANALOG CONVERTER

The Digital to Analog Converter is an advanced CMOS 8-bit DAC designed to interface directly with the 8080, 8048, 8085, Z80, and other popular microprocessors. A deposited silicon-chromium R-2R resistor ladder network divides the reference current and provides the circuit with excellent temperature tracking characteristics (0.05% of Full Scale Range maximum linearity error over temperature). The circuit uses CMOS current switches and control logic to achieve low power consumption and low output leakage current errors. Special circuitry provides TTL logic input voltage level compatibility.

Double buffering feature allows this DAC to output a voltage corresponding to one digital word while holding the next digital word. This permits the simultaneous updating of any number of DACs.

The DAC0830 Digital to Analog Converter series (DAC0830/DAC0831/DAC0832) are the 8-bit members of a family of microprocessor-compatible DACs. For applications demanding higher resolution, the DAC1000 series (10-bits) and the DAC1208 and DAC1230 (12-bits) are available alternatives.

23.2 FEATURES

- Double-buffered, single-buffered or flow-through digital data inputs.
- Easy interchange and pin-compatible with 12-bit DAC1230 series.
- Direct interface to all popular microprocessors.
- Built-in facility for zero adjustment.
- Works with ± 10V reference voltage.
- Can be used in the voltage switching mode.
- Logic inputs which meet TTL voltage level specifications.
- Operates “STAND ALONE” (without up) if desired.
- Available in 20-pin small-outline or molded chip carrier package.

23.3 PIN DIAGRAM

Fig. 23.1 shows the pin diagram of Digital to Analog Converter. DAC is a twenty pin device and description for each pin is given in table 23.1.

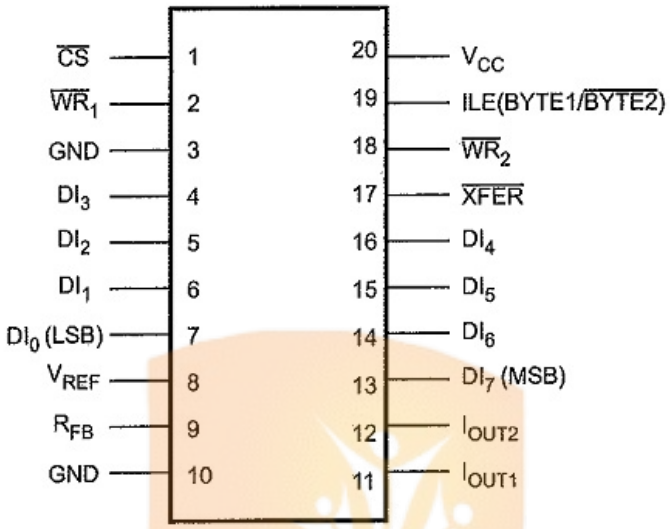


Fig. 23.1 Pin diagram of DAC

Table 23.1 Pin description of DAC

Pin	Name	Description
1	CS	Chip Select (Active Low) needed to be grounded for enabling the chip
2	WR	Write (Active Low) needed to be grounded for successful data transmission
3	GND	Ground
4	DI3	Digital Input 3 will take data for conversion
5	DI2	Digital Input 2 will take data for conversion
6	DI1	Digital Input 1 will take data for conversion
7	DI0(LSB)	Digital Input 0(Least Significant Bit) will take data for conversion
8	VREF	Voltage Reference for the chip is given at this pin
9	Rfb	Feedback Resistor will be connected to this pin
10	GND	Ground
11	IOUT1	Current Output 1 will give analog output after conversion
12	IOUT2	Current Output 2 will give analog output after conversion

13	DI7(MSB)	Digital Input 7(Most Significant Bit) will take data for conversion
14	DI6	Digital Input 6 will take data for conversion
15	DI5	Digital Input 5 will take data for conversion
16	DI4	Digital Input 4 will take data for conversion
17	XFER	Transfer Control Signal (Active Low)
18	WR2	Write 2 (Active Low)
19	ILE	Input Latch Enable pin needed to be high for successful data transmission
20	VCC	Positive Power Supply

23.4 HOW TO USE DAC

Before going for working let us consider **functional diagram of the IC DAC0832** as shown in figure 23.2.

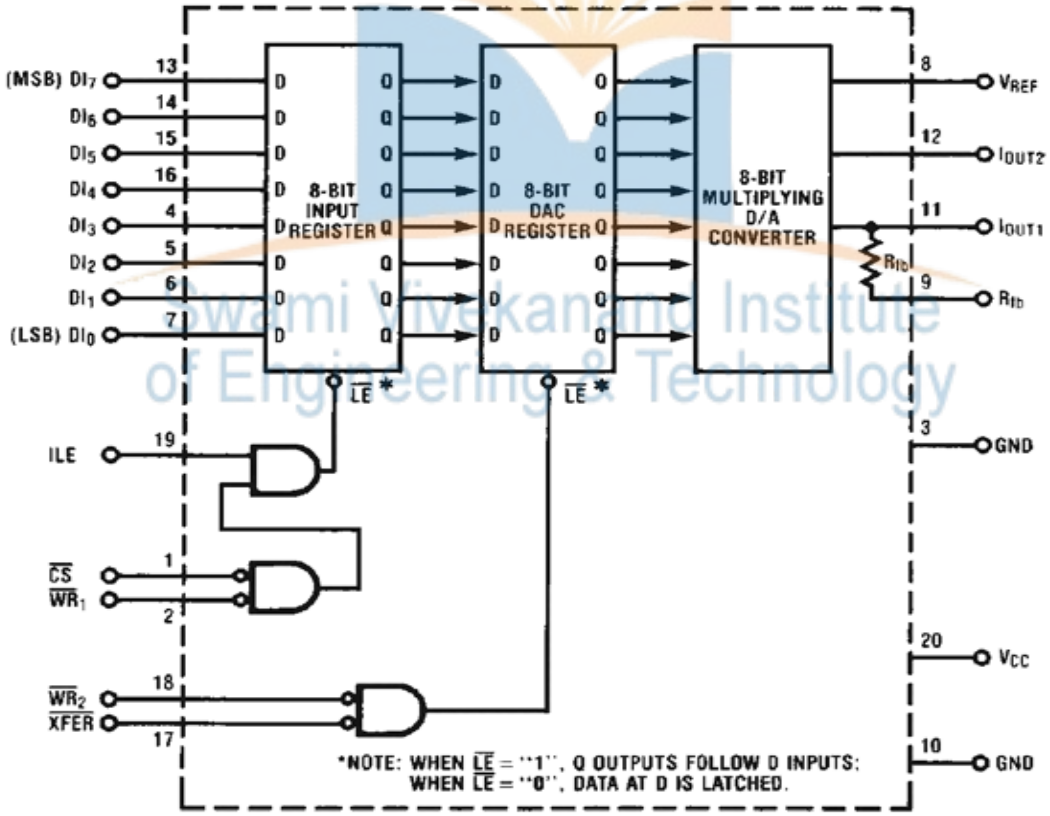


Fig. 23.2 Functional Diagram of the IC DAC0832

Based on the functional diagram we can write,

$$\overline{LE} = ILE * (\overline{CS} * \overline{WR})$$

$$\overline{LE} = \overline{WR} * \overline{XFER}$$

As mentioned in the functional diagram in both cases LE needs to be high for successful data flow. At any time if LE goes low the data flow is interrupted and the output will not be updated.

In the first case for LE to be HIGH we need:

- ILE = HIGH
- CS = LOW
- WR = LOW

In the second case for LE to be HIGH we need:

- WR = LOW
- XFER = LOW

All these control pins need to be setup appropriately as mentioned above for successful **DAC conversion** of the device.

Now for understanding the working let us consider the example application circuit given in figure 23.3.

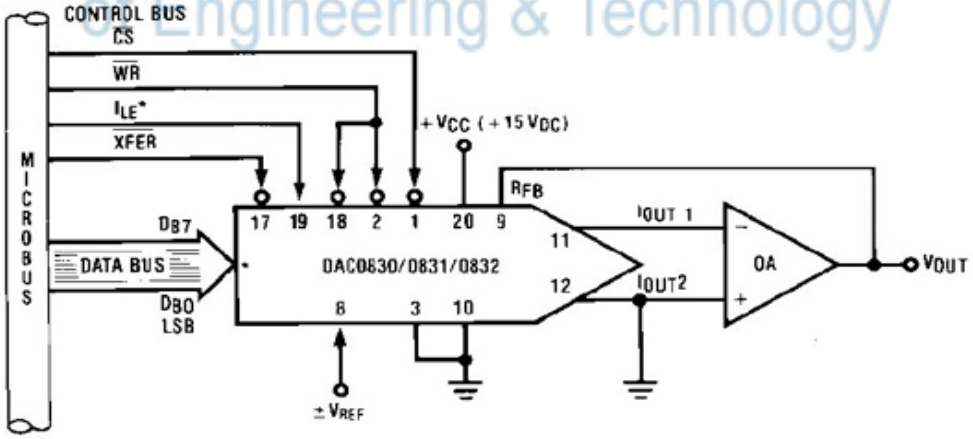


Fig. 23.3 Circuit Diagram of the IC DAC0832

In circuit:

- For working of device DAC0832 we have connected a voltage source of 15V as shown in figure 23.3.
- Eight digital inputs are given to the chip and are supposed to be in order from MSB to LSB.
- Vref is optional but remember the maximum output voltage cannot go higher than Vref

Working

DAC0832 takes in parallel 8 bit data from a microcontroller or microprocessor and converts that data in to analog signal at the output. This analog signal is the current quantity that is directly proportional to the input digital value and this current parameter needed to be converted in to voltage parameter for further use. So to convert the current parameter in to voltage parameter we will use op-amp circuit as shown in **circuit diagram**. This **op-amp circuit** is called current-to-voltage converter.

The analog voltage output from the op-amp circuit is in direct relation with inputted digital value and hence **DAC conversion with DAC0832** is achieved.

23.5 APPLICATIONS OF DAC

- Microprocessor interface
- Analog and digital circuits
- Electrical measurements
- Audio conversion
- Hobbyist applications

23.6 INTERFACING DIGITAL TO ANALOG ONVERTERS

The digital to analog converters convert binary numbers into their analog equivalent voltages or currents. Several techniques are employed for digital to analog conversion.

i. Weighted resistor network

ii. R-2R ladder network

iii. Current output D/A converter

23.7 INTERFACING DAC 0830 WITH 8086

The DAC0830 Digital to Analog Converter is connected to 8086 microprocessor, as shown in the Fig. 23.4. Here, I/O port address is decoded using OR gate. The digital data is loaded into DAC0830 when A_0-A_7 lines, WR and IO/M signals are low. This gives us the address for DAC0830 as 00H and the data can be loaded in the DAC0830 by OUT 00H, AL instruction,

where AL register contains the digital data to be sent to DAC0830. The IC 741, the operational amplifier is used to convert current output of DAC0830 to voltage output. The voltage output of the operational amplifier is used to drive the DC motor after increasing the driving capacity. The driving capacity is increased by using the darlington transistor.

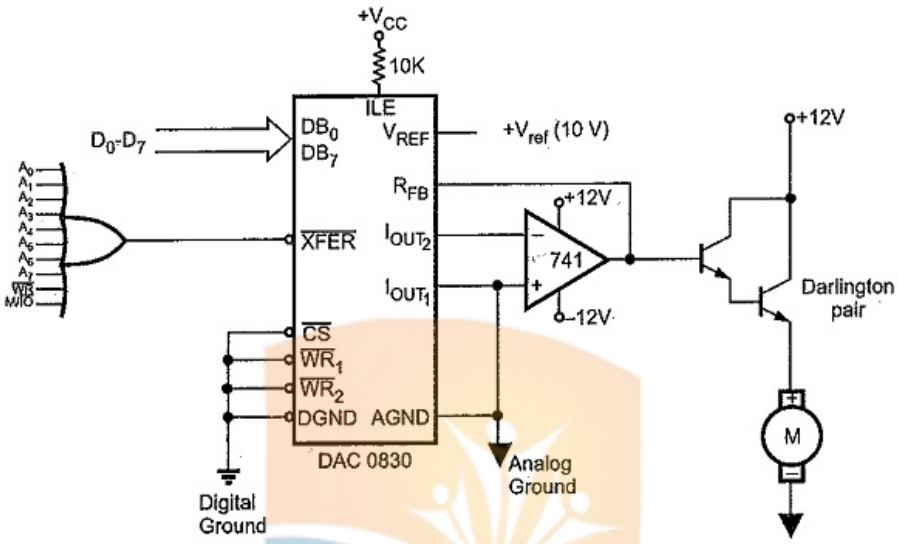


Fig. 23.4 Interfacing DAC0830 to 8086 Microprocessor

23.8 INTERFACING DAC0830 WITH 8086 USING 8255

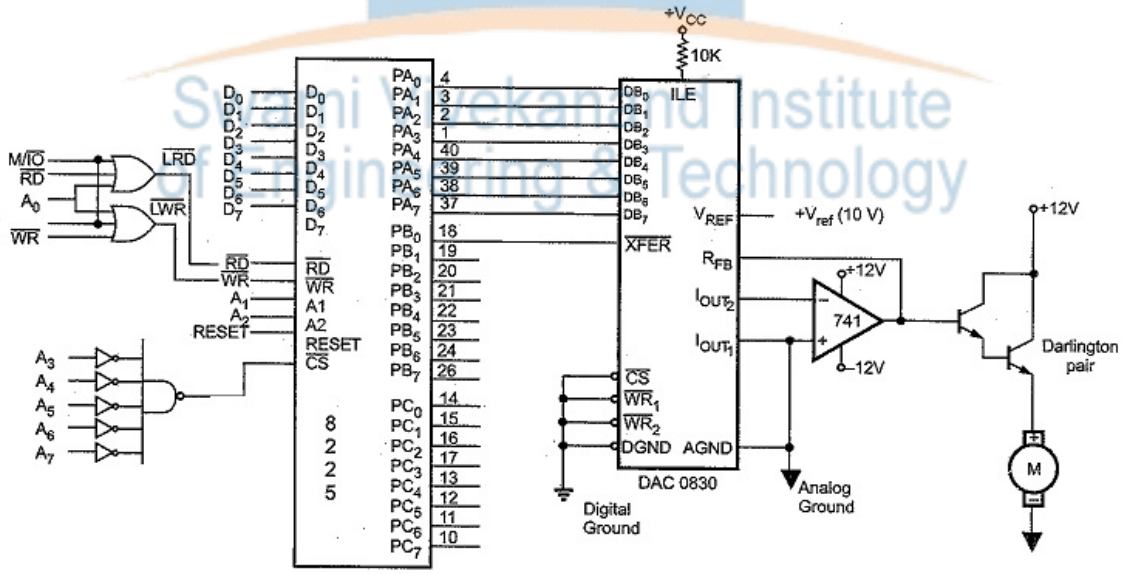


Fig. 23.5 The interfacing of DAC0830 to 8086 Microprocessor using 8255

Fig. 23.5 shows the interfacing of DAC0830 to 8086 Microprocessor using 8255. Here, port A of 8255 is used to send data to the DAC0830 and the XFER signal is generated by programming PB_0 pin of 8255. The 8255 is interfaced to 8086 system in I/O mapped I/O with address: PA = 00H, PB = 02H, PC = 04H, PC= 06H.

23.9 REFERENCES

1. https://www.eeeguide.com/dac0830-digital-to-analog-converter/
2. https://components101.com/ics/dac0832-pinout-datasheet
3. https://www.brainkart.com/article/D-A-and-A-D-Interface_7874/

CHAPTER 24

DIFFERENCE BETWEEN MACRO AND PROCEDURE

Ms. Harpreet Kaur[1] Ms. Yashu[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

A microprocessor is a computer system processor that accomplishes the functions of a CPU on a single integrated circuit. It manages the output devices, processes instructions stored in its memory, and shows the results. These processors are made up of both combinational and sequential digital circuitry. Furthermore, Assembly language is a programming language that aids in the programming of microprocessors. Both Macro and Procedure are two ideas that are utilized in microprocessor programming. The primary distinction between these processes is that the Macro is utilized for a small number of commands. In contrast, the Procedure is utilized for a large number of instructions.

In this chapter, we will discuss about the difference between Macro and Procedure. But before discussing the differences, you must know about Macro and Procedure.

24.1 MACRO

Macro is a set of instruction and the programmer can use it anywhere in the program by using its name. It is mainly used to achieve modular programming. So same set of instructions can be used multiple times when ever required by the help of macro. Wherever macro's identifier is used, it is replaced by the actual defined instructions during compilation thereby no calling and return occurs.

Syntax of macro:

%macro macro_name number_of_parameters

<macro body>

%endmacro

Advantages:

- Macro reduces the amount of repetitive coding.
- Program becomes more readable and simple.

- Execution time is less as compared to calling procedures.
- Reduces errors caused by repetitive coding.

Disadvantage of macro is that the memory requirement of a program becomes more

24.2 PROCEDURE

Procedures are also like macro, but they are used for large set of instruction when macro is useful for small set of instructions. It contains a set of instructions which performs a specific task. It contains three main parts i.e. Procedure name to identify the procedure, procedure body which contains set of instructions, and RET statement which denotes return statement. Unlike macros, procedures follow call-return method thereby achieving true modularity.

Syntax of Procedure:

procedure_name:

procedure body

…….

RET

To call a procedure

CALL procedure_name

After execution of procedure control passes to the calling procedure using RET statement.

A procedure can be of two types.

1) Near Procedure 2) Far Procedure

Near Procedure: A procedure is known as NEAR procedure if is written(defined) in the same code segment which is calling that procedure. Only Instruction Pointer(IP register) contents will be changed in NEAR procedure.

FAR procedure: A procedure is known as FAR procedure if it is written (defined) in the different code segment than the calling segment. In this case both Instruction Pointer(IP) and the Code Segment(CS) register content will be changed.

Directives used for procedure:

PROC directive: The PROC directive is used to identify the start of a procedure. The PROC directive follows a name given to the procedure. After that the term FAR and NEAR is used to specify the type of the procedure.

ENDP Directive: This directive is used along with the name of the procedure to indicate the end of a procedure to the assembler. The PROC and ENDP directive are used to bracket a procedure.

Advantages

- **Reusability of code:** The procedures provide us an ease in our code by making the set of instructions reusable. So, we need not write the same set of instructions again and again when required.
- **Less usage of memory:** The procedure is a subprogram which is stored in the memory only one. But it can use as many times as required. So, this occupies less memory space.
- **Development becomes easier:** By using procedures in a code, the modular programming can be well implemented and so, each part of the code is written in a different module which can be developed by a separate developer. So, this reduces the burden on one developer and the programming becomes simple for each of them.
- **Reduced development time:** As separate developers can work on different modules of programs separately, multiple developers can work on the project simultaneously which reduces the total development time of the project.
- **Debugging and error fixing becomes easier:** It becomes easier to detect and fix errors if the program is present in different modules rather than the entire program being present in a single code fragment.

Disadvantages

- **Extra code is required to integrate the procedures:** Every time a procedure is to be implemented in the program, we require CALL and RET instructions to integrate the procedures with the calling program.
- **Extra time required to link the procedures:** Whenever a call to procedure is made, the control of the processor goes to the procedure code, and then returns to the calling program code. This takes a lot of extra time.
- **Not efficient way to use for a small set of instructions:** When the number of instructions is much less, then the calling and returning from the procedures itself takes a lot more time than executing the instructions itself. So, it is better to use procedures in cases where the set of instructions to be included in the procedure is large.

- **Extra load on the processor:** The processor needs to do extra work to save the status of the current procedure and load status of the called procedure. Also, the instruction queue must be emptied so that the instructions of the procedure can be filled in the queue.

24.3 DIFFERENCES BETWEEN MACRO AND PROCEDURE

Some of the main differences between the Macro and Procedure are as follows:

1. A macro definition defines a macro as a series of instructions that facilitate modular programming. On the other hand, a procedure is a sequence of instructions that a programmer can repeatedly invoke and that carry out a particular task.
2. When a macro is called, a new machine code is created. In contrast, only one instance of the machine code is generated during the procedure.
3. A macro also removes the overhead time involved in calling the procedure and starting the program again. In contrast, calling a process and returning to the calling procedure from the calling procedure require greater overhead time.
4. The parameters in the macro are given as part of a sentence that calls the macro. In contrast, parameters are supplied in registers and memory locations on the stack in a procedure.
5. CALL and RET instructions are not required for the macro. On the other hand, a procedure necessitates CALL and RETS commands.
6. A macro is utilized for a small number of instructions, usually less than ten. In contrast, a procedure is utilized for a large number of instructions, usually more than ten.
7. A macro necessitates extra memory. On the other hand, a procedure necessitates less memory.
8. The execution speed of a macro is faster. In contrast, the execution speed of a procedure is slower than a macro.

24.4 HEAD-TO-HEAD COMPARISON BETWEEN MACRO AND PROCEDURE

Here, we will discuss the head-to-head comparisons between Macro and Procedure. The main differences between Macro and Procedure are as follows:

Features	Macro	Procedure
Definition	It is a series of rules or programmable patterns that decrypts a specified input sequence into a predefined output sequence.	It is a sequence of instructions that a programmer can repeatedly call to execute a specified purpose.

Memory Requirement	It needs large memory.	It needs less memory.
Machine Code	When a macro is called, a new machine code is created.	Only one instance of the machine code is generated during the procedure.
Sets of Instructions	It is utilized for a small number of instructions, usually less than ten.	It is utilized for a large number of instructions, usually more than ten.
CALL and RET	It doesn't need CALL and RET instructions.	It needs CALL and RET instructions.
Overhead Time	It removes the overhead time involved in calling the procedure and starting the program again.	Calling a process and returning to the calling procedure from the calling procedure require greater overhead time.
Execution Speed	The execution speed of a macro is faster.	The execution speed of a procedure is slower than a macro.
Passing Parameters	A parameter is passed as part of the statement that calls the macro in a macro.	The parameters are transmitted through registers and stack memory addresses.
Assembler Directives	Assembly directive MACRO is utilized to define macros, while the assembler directive ENDM is utilized to signify that the body has finished.	Assembler directive PROC is utilized to specify the procedure, and ENDP is utilized to signify that the body has finished.

24.5 REFERENCES

1. https://www.includehelp.com/embedded-system/macros-in-the-8086-microprocessor.aspx
2. https://www.javatpoint.com/difference-between-macro-and-procedure
3. https://www.scribd.com/document/342781063/6-Procedure-and-Macro-in-Assembly-Language-Program
4. https://www.geeksforgeeks.org/difference-between-macro-and-procedure/
5. https://www.snjb.org/polytechnic/up-images/downloads/chapter%206-MAPupFile_058d4fa990abaa.pdf

CHAPTER 25

THE JUMP INSTRUCTIONS IN 8086 MICROPROCESSOR

Ms. Yashu[1] Ms. Kiran Bala[2]

[1]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*
[2]*Assistant Professor, Swami Vivekanand Institute of Engineering & Technology*

The control flow relates to altering the execution path of instructions in a program. For example, a control flow decision may cause a sequence of instructions to be repeated or a group of instructions to not be executed at all. The jump instruction is provided in the 8086 instruction set for implementing control flow operations.

In the 8086 architecture, the code segment register and instruction pointer keep track of the next instruction to be fetched for execution. Thus, to initiate a change in control flow, a jump instruction must change the contents of these registers.

In this way, execution continues at an address other than that of the next sequential instruction. That is, a jump occurs to another part of the program.

25.1 JUMP INSTRUCTIONS

Jump Instructions are used for changing the flow of execution of instructions in the processor. If we want jump to any instruction in between the code, then this can be achieved by these instructions. **There are two types of Jump instructions**:

1. Unconditional Jump Instructions
2. Conditional Jump Instructions

25.2 UNCONDITIONAL JUMP INSTRUCTIONS

These instructions are used to jump on a particular location unconditionally, i.e. there is no need to satisfy any condition for the jump to take place.

There are three types of procedures used for unconditional jump. They are:

NEAR – This procedure targets within the same code segment (Intra-segment). A near-type jump instruction can cause the next instruction to be fetched from anywhere in the current code segment.

- A 16 bit signed displacement means that the jump can be to a location anywhere from +32,767 to -32,768 bytes from the current instruction pointer location.

- A *positive displacement* is an ahead jump and a *negative displacement* is a backward jump.

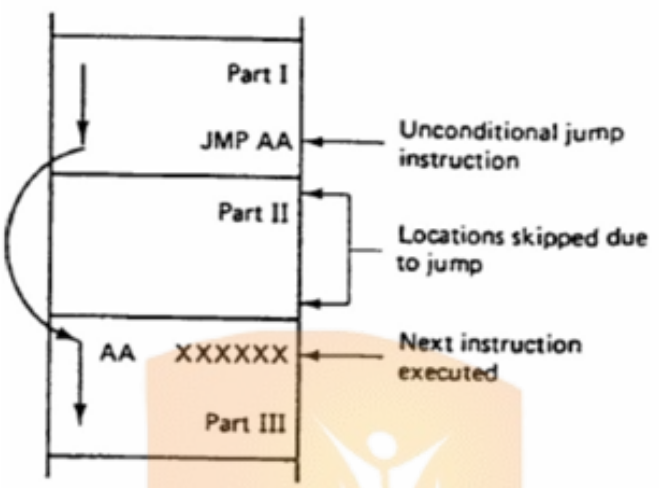


Fig. 25.1 Unconditional jump

SHORT - This procedure also targets within the same code segment, but the offset is 1 byte long. (Intra-segment)

FAR - In this procedure, the target is outside the segment and the size of the pointer is double word. (Inter-segment)

Syntax: JMP procedure_namememory_location

Example: JMP short target

Near and far jumps are further described as either

i) Direct

- If the destination address for the jump is specified directly as a part of the instruction, then the jump is described as direct. It is further classified as *direct near jump* and *direct far jump*.

ii) Indirect

- If the destination address for the jump is contained in a register or memory location, the jump is referred to as an indirect jump.

25.3 CONDITIONAL JUMPS

In these types of instructions, the processor must check for the particular condition. If it is true, then only the jump takes place else the normal flow in the execution of the statements is maintained.

The ALU operations set flags in the status word (Flag register). The conditional jump statements test the flag and jump is the flag is set.

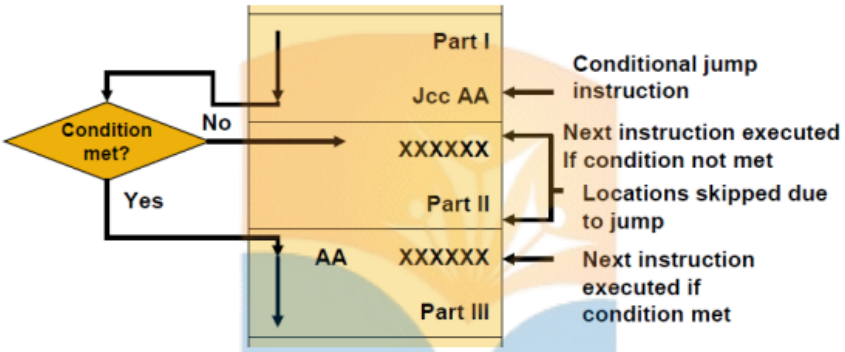


Fig. 25.2 conditional jump

There are following types of conditional jump instructions:

i) JC: Stands for 'Jump if Carry'

It checks whether the carry flag is set or not. If yes, then jump takes place, that is: **If CF = 1, then jump.**

ii) JNC: Stands for 'Jump if Not Carry'

It checks whether the carry flag is reset or not. If yes, then jump takes place, that is: **If CF = 0, then jump.**

iii) JE / JZ: Stands for 'Jump if Equal' or 'Jump if Zero'

It checks whether the zero flag is set or not. If yes, then jump takes place, that is: **If ZF = 1, then jump.**

iv) JNE / JNZ: Stands for 'Jump if Not Equal' or 'Jump if Not Zero'

It checks whether the zero flag is reset or not. If yes, then jump takes place, that is: **If ZF = 0, then jump.**

v) JP / JPE: Stands for 'Jump if Parity' or 'Jump if Even Parity'

It checks whether the Parity flag is set or not. If yes, then jump takes place, that is: **If PF = 1, then jump.**

vi) JNP / JPO: Stands for 'Jump if Not Parity' or 'Jump if Odd Parity'

It checks whether the Parity flag is reset or not. If yes, then jump takes place, that is: **If PF = 0, then jump.**

25.4 REFERENCES

1. https://www.includehelp.com/embedded-system/jump-instructions-in-8086-microprocessor.aspx
2. https://coeng.uobaghdad.edu.iq/wp-content/uploads/sites/3/2021/09/LC12.pdf
3. https://csenotesforyou.blogspot.com/2016/12/jumps-flags-and-conditional-jumps.html
4. https://opencourses.emu.edu.tr/pluginfile.php/63699/mod_resource/content/1/eeng410-Lecture7.pdf