Microprocessor And Its Applications

BCA- EB

UNIT I: INTRODUCTION TO MICROPROCESSOR

1.0 Learning Objectives

After going through this unit, you will be able to:

- Know the history of Microprocessor
- Define Microprocessor
- Know the internal architecture of a microprocessor
- Explain different types of busses
- Know different addressing techniques
- Perform decimal, binary and hexadecimal number conversions
- Define bit, byte and nibble

1.1 Introduction

Prehistoric man used primitive tools were used for hunting, gathering, building, and animal husbandry[1]. The use of tools evolved as the needs of man did, and new tools were invented as new needs arose. These tools were used to craft the world as me know it today. Many types of tools have been created for the benefit of mankind. The invention of the computer is no exception. But the computer has not always been what we know it today.

The origin of modern computer is arguable. This is due in part to ambiguity of the word "computer." In the strictest sense, a computer is something (or someone) that performs mathematical calculations. There are plenty of devices that fit this description ranging all the way back to the first known calculating device, the abacus. According to American Heritage Dictionary, a computer is "a device that computes, especially a programmable electronic machine that performs high-speed mathematical or logical operations or that assembles, stores, correlates, or otherwise processes information" (computer). The most commonly known form of a computer is, of course, the Personal Computer or PC. The modern PC is built almost entirely around one essential piece of hardware, the microprocessor.

[1] http://microprocs.wikispaces.com/Introduction

The microprocessor is commonly described as the "brain" of any electronic device that employs it. It is the device that performs arithmetic and logical function at speeds that are many orders of magnitude faster than any human brain can perform. And, just as tools have evolved, many types of microprocessors have been invented to suite the many needs of man.

1.2 History

In the early 1960s, Fairchild Semiconductor and Texas instruments introduced the first commercially available integrated circuits (Warner). The integrated circuit, or IC, revolutionized the electronics industry by placing multiple circuit components onto a miniaturized chip. The miniaturization of these components allowed for more complex circuits to be constructed. The increase in complexity of these chips can roughly be modeled by Moore's Law, which states that the amounts of transistors, the basic building block of digital electronics, will double every eighteen months (Moore's Law).

In 1971, Intel introduced the world's first single chip microprocessors which was invented by Intel engineers Federico Faggin, Ted Hoff, and Stan Mozor. Intel labeled it the 4004 Microprocessor that utilized the new silicon-gated MOS technology that many other engineers and scientist at that time believed that it was not yet possible for a single chip was able to contain the central processor unit (CPU), memory, input and output controls onto one, single chip. The chip was only 4-bit CPU, which compared to today's 64-bit microprocessors yet the microprocessor is still based on the same methods and designs even close to forty years later (Bellis).

In 1973, Texas instruments released their first single-chip microprocessor, the TI TMS 1000 yet it wasn't until 1974 in which the standalone version was create for mass production at $2 apiece. Both the 4004 and the TI TMS 1000 chips were designed for use in calculators (Warner).

Intel was working on a new design, an upgraded model of the 4004 series, the 8008 which was introduced in mid 1972. The 8008 was considered to be the "first truly usable microprocessor (Warner)." The future upgrade in improvements in the memory and features such as improvements in stacks in the memory bank is what promised Intel a future in the microcomputer and microprocessor market with their development of the 8080 (Warner).

The demand for microprocessors skyrocketed. Ever since the early 1980s, new developments in technology have increased the desire as well as the performance for the use of microprocessors. The idea of a small electronic computing device that was easy and cheap to manufacture was incredibly popular. Eventually, engineers realized that the functionality of the microprocessor could be extended far beyond the scope of a calculator. Today, almost every electronic has some form of microprocessor control the electronics functions such as digital clocks, televisions, MP3 Players, personal computers, cell phones, car ignitions, etc. All of these advancements have developed since the release of the original two chips.

1.2.2 Evolution of Intel Microprocessor Family

The evolution of Intel Microprocessor Family is summarized in the Table 1below:

Table 1: Evolution of Intel Microprocessor Family

Processor	Year of Introduction	Transistor	Clock Rate(MHz)	External Data Bus	Internal Data Bus	Address Bus
4004	1971	2250	0.108	4	8	12
8008	1972	3500	0.200	8	8	14
8080	1974	6000	3	8	8	16
8085	1976	6000	6	8	8	16
8086	1978	29000	10	16	16	20
8088	1979	29000	10	8	16	20
80286	1982	134000	12.5	16	16	25
80386DX	1985	275000	33	32	32	32
80386SX	1988	275000	33	16	32	24
Pantium C	1993	3100000	66-200	64	32	32
Pantium MMX	1997	4500000	300	64	32	32
Pantium Pro	1995	5500000	200	64	32	36
Pantium II	1997	7500000	233-450	64	32	36
Pantium III	1999	9500000	550-733	64	32	36
Itanium	2001	30000000	800-...	128	64	64

1.3 Microcomputer and Microprocessors

There are three major parts of a Computer System[2].

1. **Central Processing Unit (CPU):** Also simply called as the microprocessor acts as the brain coordinating all activities within a computer.
2. **The Memory:** The program instructions and data are primarily stored.
3. **The Input/Output(I/O) Devices:** Allow the computer to input information for processing and then output the results. I/O Devices are also known as computer peripherals.

The integrated Circuit (IC) chip containing the CPU is called the microprocessor and the entire computer including the microprocessor, memory and I/O is called a microcomputer.

1.4 Internal organization of a microcomputer

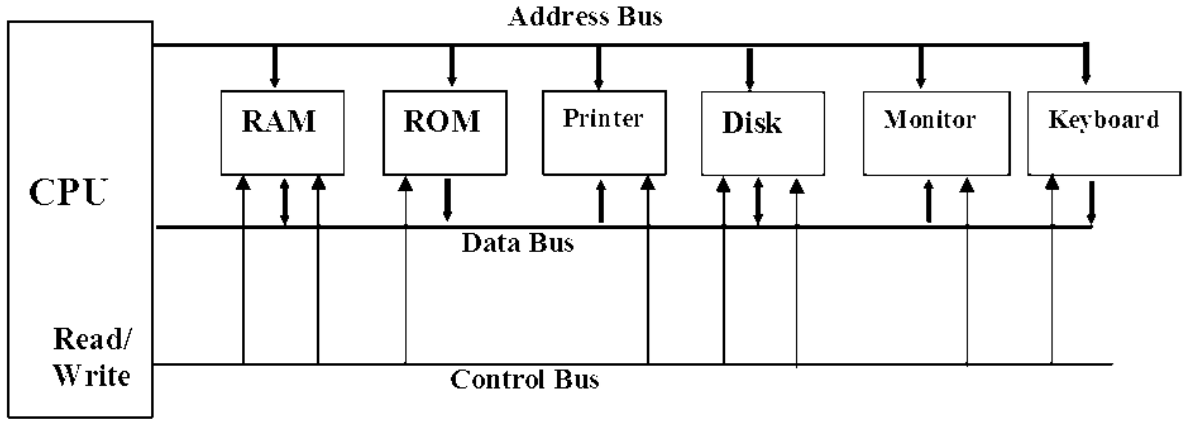


Figure 1: Internal organization of a microcomputer

The CPU is connected to memory and I/O devices through a strip of wires called a bus. The bus inside a computer carries information from place to place. In every computer there are three types of busses:

[2] http://faraday.ee.emu.edu.tr/eeng410/lectures/eeng410_Lecture1.pdf

1. **Address Bus:** The address bus is used to identify the memory location or I/O device the processor intends to communicate with. The width of the Address Bus rages from 20 bits (8086) to 36 bits for (Pentium II).
2. **Data Bus:** Data bus is used by the CPU to get data from / to send data to the memory or the I/O devices. The width of a microprocessor is used to classify the microprocessor. The size of data bus of Intel microprocessors vary between 8-bit (8085) to 64-bit (Pentium).
3. **Control Bus:** How can we tell if the address on the bus is memory address or an I/O device address? This where the control bus comes in. Each time the processor outputs an address it also activates one of the four control bus signals: Memory Read, Memory Write, I/O Read and I/O Write.

The address and control bus contains output lines only, therefore it is unidirectional, but the data bus is bidirectional.

A program stored in the memory provides instructions to the CPU to perform a specific action. This action can be a simple addition. It is function of the CPU to fetch the program instructions from the memory and execute them.

1. The CPU contains a number of registers to store information inside the CPU temporarily. Registers inside the CPU can be 8-bit, 16-bit, 32-bit or even 64-bit depending on the CPU.
2. The CPU also contains Arithmetic and Logic Unit (ALU). The ALU performs arithmetic (add, subtract, multiply, divide) and logic (AND, OR, NOT) functions.
3. The CPU contains a program counter also known as the Instruction Pointer to point the address of the next instruction to be executed.
4. Instruction Decoder is a kind of dictionary which is used to interpret the meaning of the instruction fetched into the CPU. Appropriate control signals are generated according to the meaning of the instruction.
5. Accumulator is a Special Function Register(SFR) closely related to the operation of the ALU. It is a kind of working desk used for storing all data upon which some operation should be performed (addition, shift/move etc.). It also stores the results ready for use in further processing. One of the SFRs, called a Status Register (PSW), is closely related to

the accumulator. It shows at any given moment the "status" of a number stored in the accumulator (number is greater or less than zero etc.).

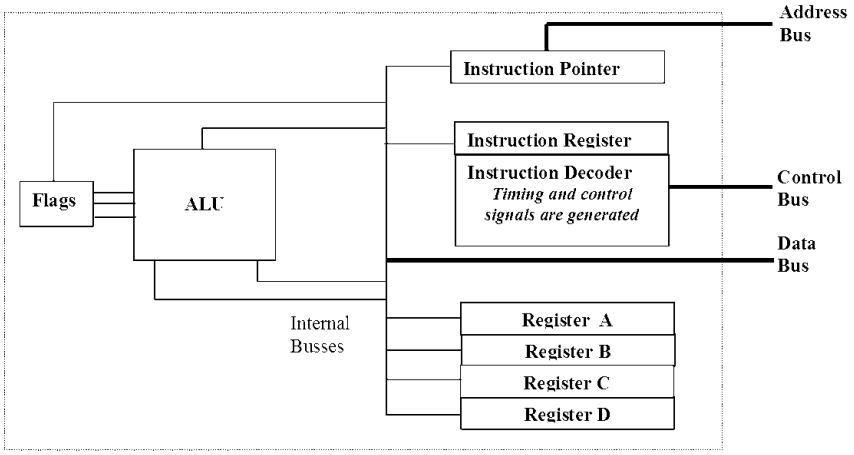


Figure 2: Internal block diagram of a CPU

The Figure 3 below demonstrates the interaction between the CPU, memory and I/O Devices.

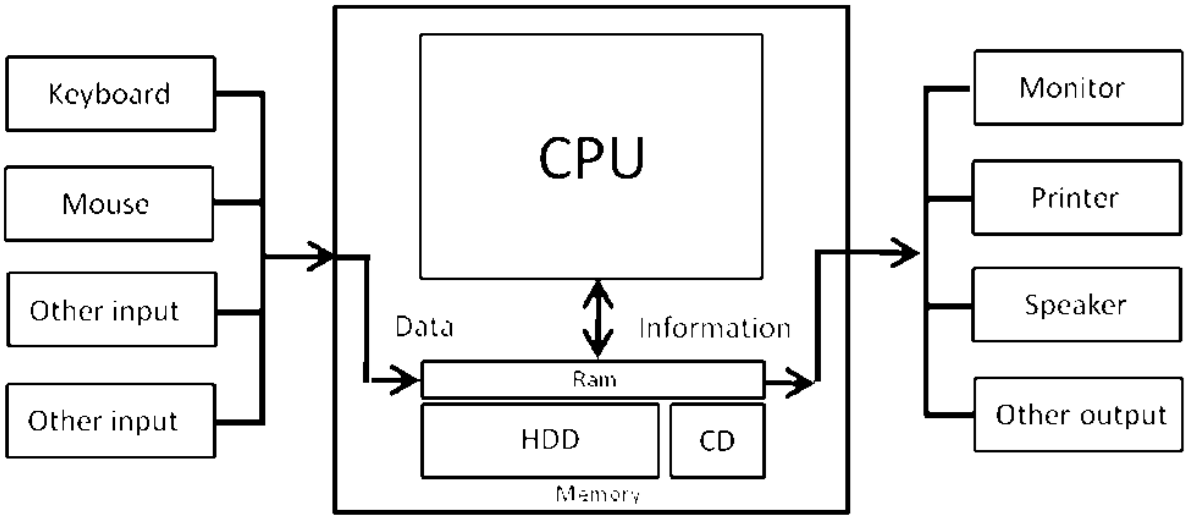


Figure 3: Interaction between the CPU, memory and I/O Devices inside a microcomputer[3]

1.4.1 Memory Unit

Memory is part of the microcomputer used for data storage. The easiest way to explain it is to compare it with a filing cabinet with many drawers. Suppose, the drawers are clearly marked so

[3] Image adopted from https://en.wikiversity.org/wiki/Hardware#/media/File:Computer2.png available under creative common license

that it is easy to access any of them. It is easy enough to find out the contents of the drawer by reading the label on the front of the drawer.

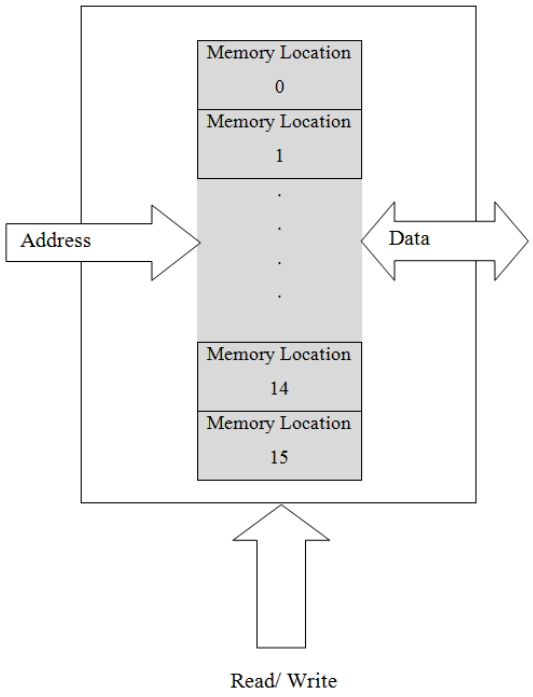


Figure 4: Memory

Each memory address corresponds to one memory location. The content of any location becomes known by its addressing. Memory can either be written to or read from.

1.4.1.1 Read Only Memory (ROM)

ROM (Read Only Memory) is used to permanently save the program being executed. The size of a program that can be written depends on the size of this memory. Today's microcontrollers commonly use 16-bit addressing, which means that they are able to address up to 64 Kb of memory, i.e. 65535 locations. As a novice, your program will rarely exceed the limit of several hundred instructions. There are several types of ROM.

1. **Masked ROM**: Microcontrollers containing this ROM are reserved for the great manufacturers. Program is loaded into the chip by the manufacturer. In case of large scale manufacture, the price is very low.

2. **One Time Programmable ROM (OTP ROM):** If the microcontroller contains this memory, you can download a program into this memory, but the process of program downloading is a "one-way ticket", meaning that it can be done only once. If an error is detected after downloading, the only thing you can do is to download the corrected program to another chip.
3. **UV Erasable Programmable ROM (UV EPROM):** Both the manufacturing process and characteristics of this memory are completely identical to OTP ROM. However, the package of this microcontroller has a recognizable "window" on the upper side. It enables the surface of the silicon chip inside to be lit by an UV lamp, which effectively erases and program from the ROM. Installation of this window is very complicated, which normally affects the price.

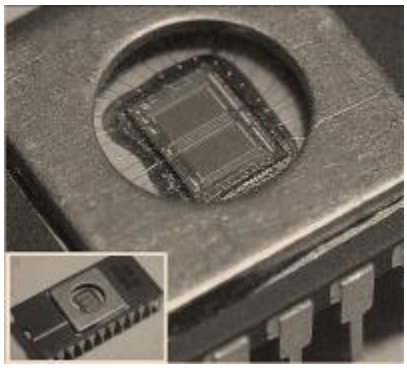


Figure 5: UV erasable programmable ROM[4]

4. **Flash memory:** This type of memory was invented in the 80s in the laboratories of INTEL and were represented as the successor to the UV EPROM. Since the contents of this memory can be written and cleared practically an unlimited number of times, the microcontrollers with Flash ROM are ideal for learning, experimentation and small-scale manufacture. Because of its popularity, the most microprocessor and microcontrollers are manufactured in flash versions today.
5. **Electrically Erasable Programmable ROM (EEPROM):** The contents of the EEPROM may be changed during operation (similar to RAM), but remains permanently saved even upon the power supply goes off (similar to ROM). Accordingly, an EEPROM

[4] Image adopted from http://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/front-matter/introduction-to-the-world-of-microcontrollers/ available under creative commons license.

is often used to store values, created during operation, which must be permanently saved. For example, if you design an electronic lock or an alarm, it would be great to enable the user to create and enter a password, but useless if it is lost every time the power supply goes off. The ideal solution is the microcontroller with an embedded EEPROM.

1.4.1.2 Random Access Memory (RAM)

Once the power supply is off the contents of RAM (Random Access Memory) is cleared. It is used for temporary storing data and intermediate results created and used during the operation of the microcontroller. For example, if the program performs an addition (of whatever), it is necessary to have a register representing what in everyday life is called the "sum". For that purpose, one of the registers in RAM is called the "sum" and used for storing results of addition.

A CPU is referred to as the brain of the computer. For any processing on data, the data need to be brought inside the CPU. The data is transferred inside the CPU via input devices and the processed information is displayed via output devices. These input/output devices like, printer, keyboard, mouse, monitor, etc. are collectively known as Peripheral devices. At times the information is stored in the secondary storage devices for future reference. Clearly, the CPU needs to communicate with memory and peripheral devices. Therefore, the CPU requires some path, known as bus, through which it can communicate to these devices. Figure 6 shows how CPU is connected to these devices.

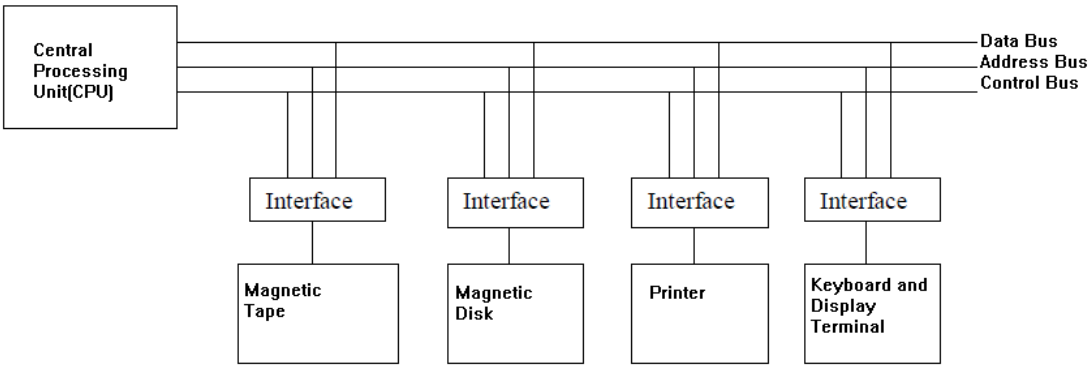


Figure 6: Processor Interconnection with Peripheral Devices

In the figure above, the peripheral devices are connected to CPU via interface. There are many differences in the implementation of CPU and the peripherals devices, some of the major differences are listed below:

1. CPU is an electronic device whereas the peripheral devices are electromechanical (like printer, which have mechanical motors for page movement) and electromagnetic devices (like secondary memory). Therefore, the technology behind these two entities and the manner of operation is different.
2. CPU transmits and processes the data at very fast rate whereas the data transfer rate of peripheral devices is comparatively very slow. Moreover, CPU transmits the data in parallel whereas peripheral devices usually transmit data serially.
3. Data codes and formats in peripherals differ from the word format in the CPU and memory.

To overcome these mismatches, a special hardware, known as interface is connected between CPU and these peripherals device to take care of all the above issues and to supervise and synchronize all input and output transfers. These components are called **interface.**

CPU needs to communicate with memory and other peripherals devices. There are many peripheral devices attached to a computer, so to locate a specific device, an address need to be specified. Similarly, to read/write data from/to memory, an address need to be specified. Also, CPU need to specify the control information i.e., what operation is to be performed. Like in case of memory, CPU needs to specify whether it is a read operation or a write operation. After specifying the address and the control information, the data need to be transferred between CPU and peripheral devices/memory. For this purpose, bus is required. Bus is a conducting path that connects the CPU with other devices. Based on the purpose for which the bus is used, it is categorised into three categories.

a. *Address Bus:* An address bus is used by the CPU to transmit the address of the peripherals/memory location. Since, it is used by CPU only, this bus is unidirectional.

b. *Control Bus:* It is used by the CPU to control and regulate the various devices attached to it. This bus is bidirectional.

c. *Data Bus:* It is used by CPU and the devices attached to the CPU to transfer data. This Bus is bidirectional.

Apart from peripheral devices, the CPU is also connected to main memory and needs to frequently communicate with it. So, bus is also required to connect the CPU with memory. There are three possible architectures to connect the CPU with memory and peripheral devices.

1. Different set of address, data and control buses are used to connect memory and other Peripheral Devices (PD), as shown in Figure 7. In this case a separate Input/Output processor (IOP) is required to control the peripheral devices. The memory communicates with CPU and IOP using memory bus whereas the IOP communicates with its separate address data and control bus.

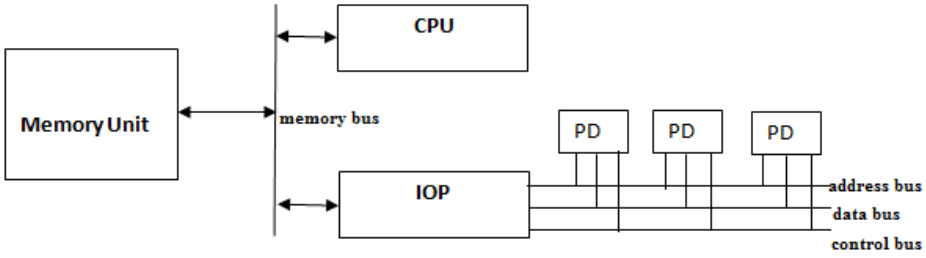


Figure 7: Interconnection of Peripheral Devices and Memory using different set of Buses

2. The second alternative is to connect CPU to memory and peripheral is using different set of control buses but data and address buses are common.

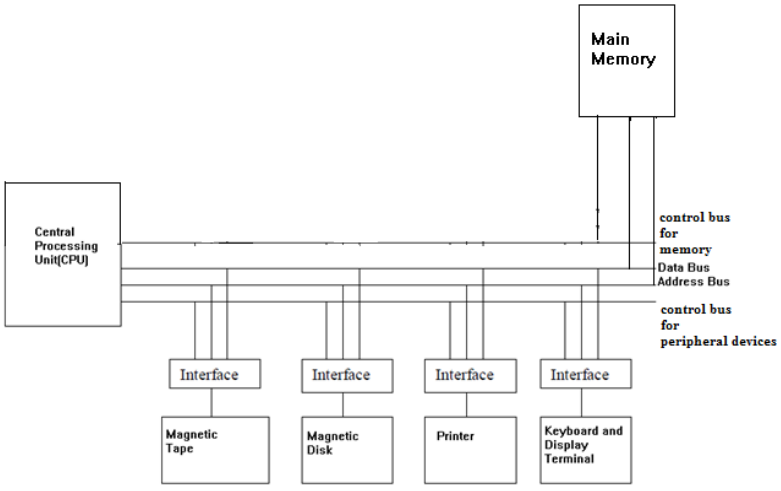


Figure 8: Interconnection of CPU and Peripheral Devices using Common Address & Data Bus and different Control Buses

The same is shown in Figure 8. When CPU needs to communicate with memory or I/O device, it places the data in the data bus and specifies the address using address bus. The distinction between I/O device and memory is made via control lines. If CPU wants to interact with memory, it will use the control lines dedicated for memory. This is will isolate all the I/O devices and the memory will clearly recognize that the address is directed to memory, not I/O devices. Similarly, if CPU wants to interact with I/O devices, it will use the control lines dedicated for I/O devices. Whenever the CPU communicates with memory, the I/O device remains isolated and when CPU communicates with I/O devices, memory becomes isolated. Therefore this configuration is also known as Isolated I/O method.

3. The third alternative is to use all the busses i.e., data, addresses and control bus to connect the CPU with peripheral and memory. The same is shown below in Figure 9.

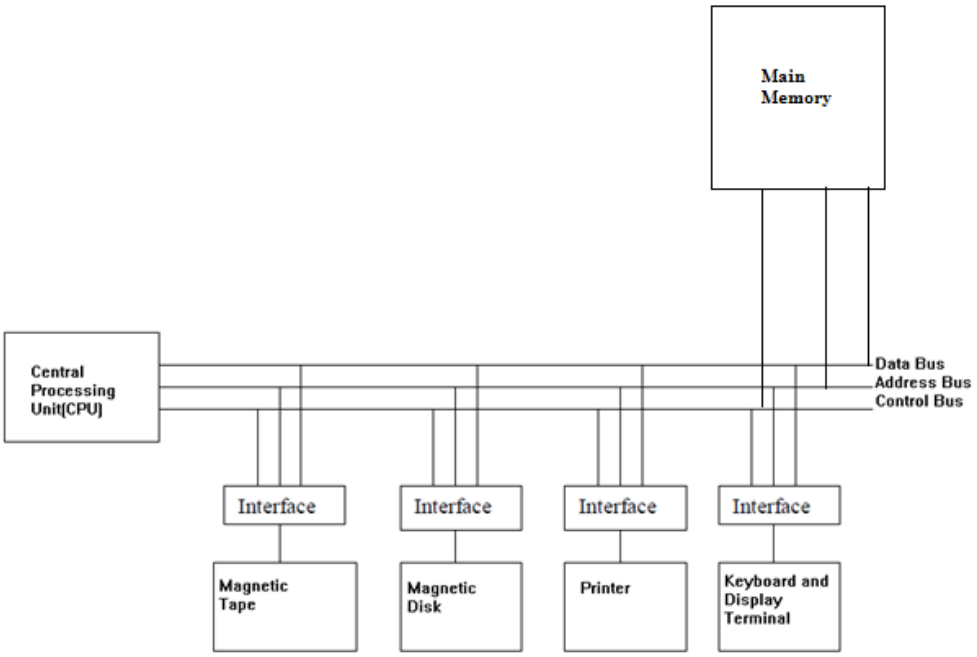


Figure 9: Interconnection of CPU and Peripheral Devices using Common Address, Data Bus Control Bus

In this case, some of the address range of the main memory is reserved to peripheral devices. This configuration is known as memory-mapped I/O. In this case, the CPU does not make any distinction between memory and peripheral devices and use the same set of read and write

signals for both memory-read/ peripheral-read and memory-write/peripheral-write operations as the interface of the peripheral devices are treated similar to memory address(refer Figure 10). Thus, this reduces the number of instructions in the instruction set of the computer as the same instructions can be used for memory as well as peripheral devices. The drawback of this scheme is, the memory cannot be fully utilized as some of the address from the address range of memory is reserved for peripheral address.

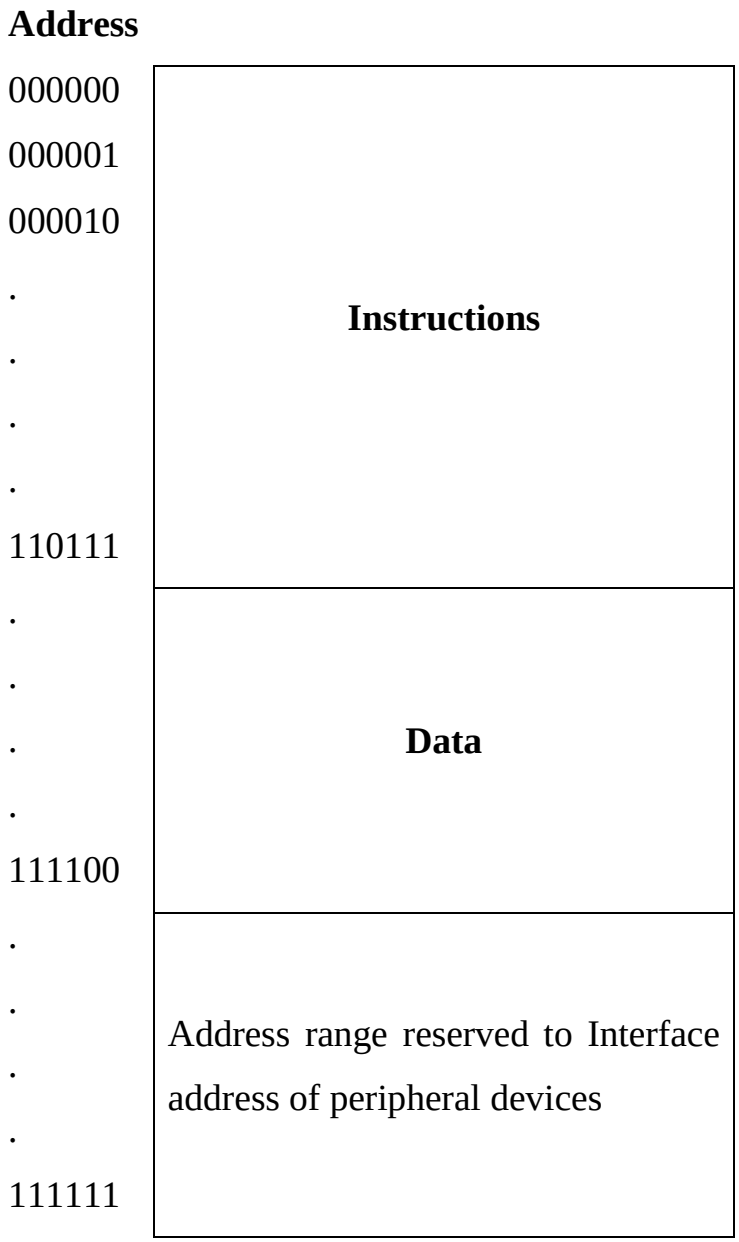


Figure 10: Address Range in Memory-Mapped I/O Configuration

1.4.2 Timers/Counters

The microcomputer oscillator uses quartz crystal for its operation. Even though it is not the simplest solution, there are many reasons to use it. Namely, the frequency of such oscillator is precisely defined and very stable, the pulses it generates are always of the same width, which makes them ideal for time measurement. Such oscillators are used in quartz watches. If it is necessary to measure time between two events, it is sufficient to count pulses coming from this oscillator. That is exactly what the timer does.

1.4.3 Input/Output Ports

In order to make the microcomputer useful, it has to be connected to additional electronics, i.e. peripherals. Each microcontroller has one or more registers (called a "port") connected to the microprocessor pins. Usually, each I/O port is under control of another SFR, which means that each bit of that register determines the state of the corresponding microcontroller pin. For example, by writing logic one (1) to one bit of that control register SFR, the appropriate port pin is automatically configured as input. It means that voltage brought to that pin can be read as logic 0 or 1. Otherwise, by writing zero to the SFR, the appropriate port pin is configured as an output. Its voltage (0V or 5V) corresponds to the state of the appropriate bit of the port register.

1.5 BASICS

1.5.1 World of Numbers

Mathematics is such a good science![5] Everything is so logical and simple as that. The whole universe can be described with ten digits only. But, does it really have to be like that? Do we need exactly ten digits? Of course not, it is only a matter of habit. Remember the lessons from the school. For example, what does the number 764 mean: four units, six tens and seven hundreds. Simple! Could it be described in a bit more complicated way? Of course it could: 4 + 60 + 700. Even more complicated? Naturally: 4*1 + 6*10 + 7*100. Could this number look a bit more scientific? The answer is yes: 4*10^0 + 6*10^1 + 7*10^2. What does it actually mean? Why do we use exactly these numbers: 100, 101 and 102? Why is it always about the number

[5] http://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/front-matter/introduction-to-the-world-of-microcontrollers/ available under creative commons license.

10? That is because we use ten different digits (0, 1, 2, … 8, 9). In other words, because we use base-10 number system, i.e. decimal number system.

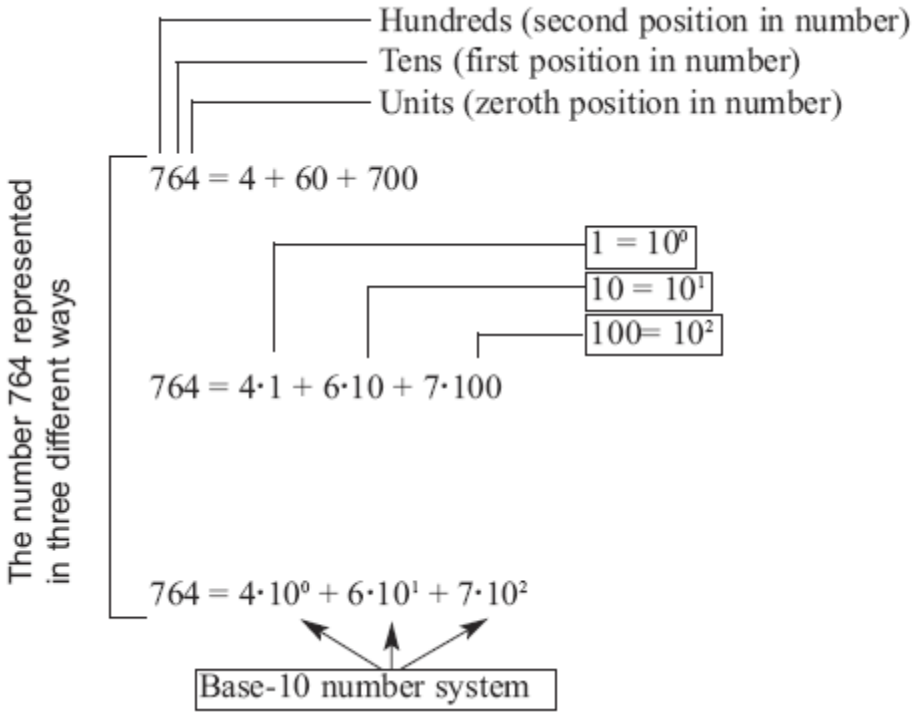


Figure 11: The number 764 represented in three different ways

1.5.1.1 Binary Number System

What would happen if only two digits would be used- 0 and 1? Or if we would not know to determine whether something is 3 or 5 times greater than something else? Or if we would be restricted when comparing two sizes, i.e. if we could only state that something exists (1) or does not exist (0)? Nothing special would happen, we would keep on using numbers in the same way, but they would look a bit different. For example: 11011010. How many pages of a book does the number 11011010 include? In order to learn that, follow the same logic like in the previous example, but in reverse order. Bear in mind that all this is about mathematics with only two digits- 0 and 1, i.e. base-2 number system (binary number system).

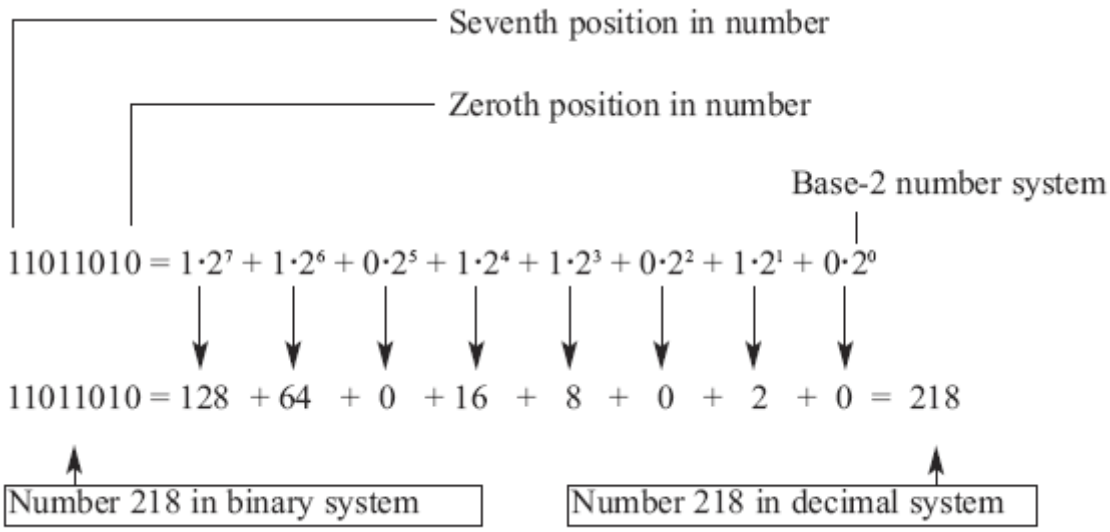


Figure 12: The number 218 represented in binary and decimal system

Clearly, it is the same number represented in two different ways. The only difference is in the number of digits necessary for writing some number. One digit (2) is used to write the number 2 in decimal system, whereas two digits (1 and 0) are used to write that number in binary system. Do you now agree that there are 10 groups of people? Welcome to the world of binary arithmetic! Do you have any idea where it is used?

Excepting strictly controlled laboratory conditions, the most complicated electronic circuits cannot accurately determine the difference between two sizes (two voltage values, for example) if they are too small (lower than several volts). The reasons are electrical noises and something called the "real working environment" (unpredictable changes of power supply voltage, temperature changes, tolerance to values of built in components etc.). Imagine a computer which would operate upon decimal numbers by recognizing 10 digits in the following way: 0=0V, 1=5V, 2=10V, 3=15V, 4=20V… 9=45V !? Did anybody say batteries? A far simpler solution is the use of binary logic where 0 indicates that there is no voltage and 1 indicates that there is voltage. It is easier to write 0 or 1 instead of "there is no voltage" or "there is voltage". It is called logic zero (0) and logic one (1) which electronics perfectly conforms with and easily performs all those endlessly complex mathematical operations. It is electronics which in reality applies mathematics in which all numbers are represented by two digits only and in which it is only important to know whether there is voltage or not. Of course, we are talking about digital electronics.

1.5.1.2 Hexadecimal Number System

At the very beginning of computer development it was realized that people had many difficulties in handling binary numbers. Because of this, a new numbering system had to be established. This time, a number system using 16 different digits. The first ten digits are the same as digits we are used to (0, 1, 2, 3,… 9) but there are six digits more. In order to keep from making up new symbols, the six letters of alphabet A, B, C, D, E and F are used. A hexadecimal number system consisting of digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F has been established. What is the purpose of this seemingly bizarre combination? Just look how perfectly everything fits the story about binary numbers.

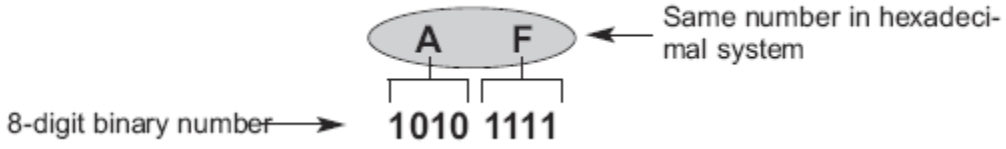


Figure 13: Binary and Hexadecimal number

The largest number that can be represented by 4 binary digits is the number 1111. It corresponds to the number 15 in decimal system. That number is in hexadecimal system represented by only one digit F. It is the largest one digit number in hexadecimal system. Do you see how skillfully it is used? The largest number written with eight digits is at the same time the largest two digit hexadecimal number. Bear in mind that the computer uses 8-digit binary numbers.

1.5.2 CODES

1.5.2.1 BCD Code

BCD code is actually a binary code for decimal numbers only. It is used to enable electronic circuits to communicate in a decimal number system with peripherals and in a binary system within “their own world”. It consists of four digit binary numbers which represent the first ten digits (0, 1, 2, 3 … 8, 9). Even though four digits can give a total of 16 possible combinations, only the first ten are used.

1.5.3 Number System Conversion

The binary numbering system is the most commonly used, the decimal system is the most understandable while the hexadecimal system is somewhere between them. Therefore, it is very important to learn how to convert numbers from one numbering system to another, i.e. how to turn a series of zeros and units into values understandable to us.

1.5.3.1 Binary to Decimal Number Conversion

Digits in a binary number have different values depending on their position in that number. Additionally, each position can contain either 1 or 0 and its value may be easily determined by its position from the right. To make the conversion of a binary number to decimal it is necessary to multiply values with the corresponding digits (0 or 1) and add all the results. The magic of binary to decimal number conversion works…You doubt? Look at the example:

110 = 1*2^2 + 1*2^1 + 0*2^0 = 6

It should be noted that for decimal numbers from 0 to 3 you only need two binary digits. For greater values, extra binary digits must be added. Thus, for numbers from 0 to 7 you need three digits, for numbers from 0 to 15- four digits etc. Simply speaking, the largest binary number consisting of n digits is obtained when the base 2 is raised by n. The result should be then subtracted by 1. For example, if n=4:

2^4 – 1 = 16 – 1 = 15

Accordingly, using 4 binary digits it is possible to represent decimal numbers from 0 to 15, including these two digits, which amounts to 16 different values in total.

1.5.3.2 Hexadecimal to Decimal Number Conversion

In order to make conversion of a hexadecimal number to decimal, each hexadecimal digit should be multiplied with the number 16 raised by its position value. For example:

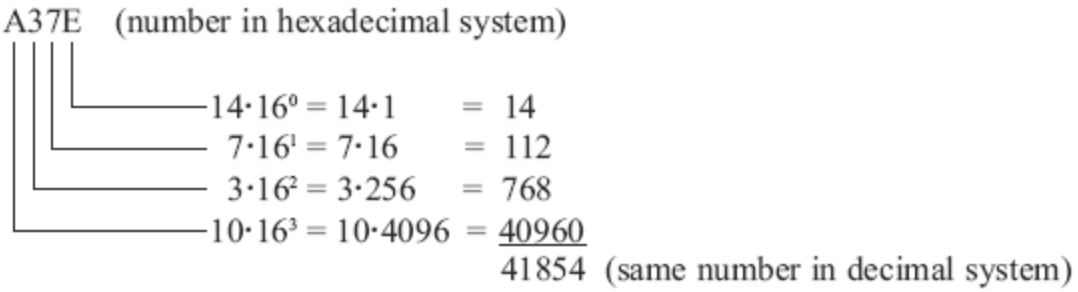


Figure 14: Hexadecimal to decimal number conversion

1.5.3.3 Hexadecimal to Binary Number Conversion

It is not necessary to perform any calculation in order to convert hexadecimal numbers to binary numbers. Hexadecimal digits are simply replaced by the appropriate four binary digits. Since the maximum hexadecimal digit is equivalent to decimal number 15, we need to use four binary digits to represent one hexadecimal digit. For example:

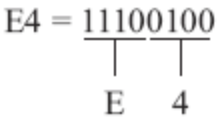


Figure 15: Hexadecimal to binary number conversion

Comparative table below contains the values of numbers 0-255 in three different numbering systems.

1.5.4 Marking Numbers

The hexadecimal numbering system is along with binary and decimal number systems considered to be the most important for us. It is easy to make conversion of any hexadecimal number to binary and it is also easy to remember it. However, these conversions may cause confusion. For example, what does the statement “It is necessary to count up 110 products on assembly line” actually mean? Depending on whether it is about binary, decimal or hexadecimal, the result could be 6, 110 or 272 products, respectively! Accordingly, in order to avoid misunderstanding, different prefixes and suffixes are directly added to the numbers. The prefix $ or 0x as well as the suffix h marks the numbers in hexadecimal system. For example,

hexadecimal number 10AF may look as follows $10AF, 0x10AF or 10AFh. Similarly, binary numbers usually get the suffix % or 0b, whereas decimal numbers get the suffix D.

DEC.	BINARY								HEX.
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	2
3	0	0	0	0	0	0	1	1	3
4	0	0	0	0	0	1	0	0	4
5	0	0	0	0	0	1	0	1	5
6	0	0	0	0	0	1	1	0	6
7	0	0	0	0	0	1	1	1	7
8	0	0	0	0	1	0	0	0	8
9	0	0	0	0	1	0	0	1	9
10	0	0	0	0	1	0	1	0	A
11	0	0	0	0	1	0	1	1	B
12	0	0	0	0	1	1	0	0	C
13	0	0	0	0	1	1	0	1	D
14	0	0	0	0	1	1	1	0	E
15	0	0	0	0	1	1	1	1	F
16	0	0	0	1	0	0	0	0	10
17	0	0	0	1	0	0	0	1	11
	.....								
	.....								
	.....								
253	1	1	1	1	1	1	0	1	FD
254	1	1	1	1	1	1	1	0	FE
255	1	1	1	1	1	1	1	1	FF

Figure 16: Conversion table[6]

1.5.5 Bit

Theory says a bit is the basic unit of information… Let's forget this dry explanation for a moment and take a look at what it is in practice. The answer is nothing special a bit is a binary digit. Similar to decimal number system in which digits in a number do not have the same value (for example digits in the number 444 are the same, but have different values), the "significance" of the bit depends on the position it has in the binary number. Therefore, there is no point talking about units, tens etc. Instead, here it is about the zero bit (rightmost bit), first bit (second from the right) etc. In addition, since the binary system uses two digits only (0 and 1), the value of one bit can be 0 or 1.

[6] Adopted from http://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/front-matter/introduction-to-the-world-of-microcontrollers/

Don’t be confused if you find some bit has value 4, 16 or 64. It means that bit’s values are represented in decimal system. Simply, we have got so much accustomed to the usage of decimal numbers that these expressions became common. It would be correct to say for example, “the value of sixth bit in binary number is equivalent to decimal number 64”. But we are human and habits die hard… Besides, how would it sound “number: one-onezero- one-zero…”

1.5.6 Byte

A byte or a program word consists of eight bits grouped together. If a bit is a digit, it is logical that bytes represent numbers. All mathematical operations can be performed upon them, like upon common decimal numbers. As is the case with digits of any other number, byte digits do not have the same significance. The largest value has the leftmost bit called the most significant bit (MSB). The rightmost bit has the least value and is therefore called the least significant bit (LSB). Since eight zeros and units of one byte can be combined in 256 different ways, the largest decimal number which can be represented by one byte is 255 (one combination represents zero).

A nibble is referred to as half a byte. Depending on which half of the byte we are talking about (left or right), there are “high” and “low” nibbles.

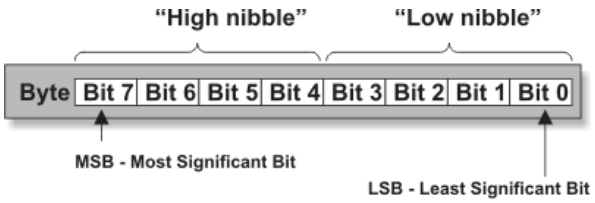


Figure 17: High and Low nibbles[7]

1.6 Summary

1. 1946 The first generation of Computer ENIAC was started to be used on the vacuum tube technology.
2. 1958 The first transistorized computer TRADIC was announced by IBM.
3. 1959 The first IC was invented.
4. 1960s ICs were started to be used in CPU boards.

[7] Adopted from http://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/front-matter/introduction-to-the-world-of-microcontrollers/

5. 1970s entire CPU was put in a single chip. (1971 the first microprocessor of Intel 4004 with 4-bit data bus and around 2300 transistors was introduced).
6. Late 1970s Intel 8080/85 appeared with 8-bit data bus and 16-bit address bus and used from traffic light controllers to homemade computers.1981 First PC was introduced by IBM Intel 8088 microprocessor.
7. Motorola emerged with 6800. Apple Macintosh computers started to use 68000 series of microprocessors.
8. The increase in complexity of these chips can roughly be modeled by Moore's Law, which states that the amount of transistors, the basic building block of digital electronics, will double every eighteen months.
9. A program stored in the memory provides instructions to the CPU to perform a specific action.
10. Memory is part of the microcomputer used for data storage.
11. Each memory address corresponds to one memory location.
12. A CPU is referred to as the brain of the computer. For any processing on data, the data need to be brought inside the CPU. The data is transferred inside the CPU via input devices and the processed information is displayed via output devices.
13. The peripheral devices are connected to CPU via interface.
14. CPU needs to communicate with memory and other peripherals devices. There are many peripheral devices attached to a computer, so to locate a specific device, an address need to be specified.

1.7 Check your progress

Fill in the blanks

1. IC stands for _____.
2. The CPU is connected to memory and I/O devices through a strip of wires called a ____.
3. The address and control bus contains output lines only, therefore it is _____, but the data bus is _____ .
4. SPR stands for _____.
5. _____ shows at any given moment the “status” of a number stored in the accumulator (number is greater or less than zero etc.).
6. The input/output devices like, printer, keyboard, mouse, monitor, etc. are collectively known as _____ devices.

Answers to fill in the blanks

1. Integrated circuit.
2. Bus
3. Unidirectional, bidirectional
4. Special Function Register
5. Program Status Word
6. Peripheral

1.8 Model Questions

1. What is a computer?
2. Define microprocessor.
3. State Moor's law,
4. What are the major components of a microcomputer system?
5. What is an accumulator?
6. With a help of a diagram, demonstrates the interaction between the CPU, memory and I/O Devices.
7. Explain the difference between OTP ROM and UV erasable programmable ROM.
8. List out the differences in the implementation of CPU and the peripherals devices.
9. Explain the three possible architectures to connect the CPU with memory and peripheral devices.
10. What is a timer? Why it is used in a microcomputer?

UNIT II: MEMORY

2.0 Learning Objectives

After reading this unit, you will be able to:

- Know the development of memory
- Understand the working of D- flip flop
- Explain the working of m-bit register
- Explain the development of memory chips
- Understand memory read/write operations
- Classify RAM, ROM, PROM, EPROM and SRAM
- Explain the characteristics of memory
- Calculate the access time of the memory
- Explain the architecture of Intel 2716 ROM, Intel 6116 RAM, Dynamic RAM and 4116 DRAM

2.1 Introduction

Memory is a storage device. It stores program data and the results. There are two kind of memories; semiconductor memories & magnetic memories[8]. Semiconductor memories are faster, smaller, and lighter and consume less power. Semiconductor memories are used as the main memory of a computer. Magnetic memories are slow but they are cheaper than semiconductor memories. Magnetic memories are used as the secondary memories of a computer for bulk storage of data and information's. With the development in technology, semiconductor memories are used everywhere. Let us see how semiconductor memories are developed.

[8] Adopted from http://nptel.ac.in/courses/108107029/4 available under creative commons license.

2.2 Development of Memory

The smallest unit of information a digital system can store in a binary digit which has a logic value of 0 or 1. A bit of data is stored in electronics devices called a flip – flop or a 1-bit register. A flip – flop is a general memory and has two stable states in which it can remain indefinitely as long as the operating power is not interrupted. The output can be changed only if the input signals allow for it. A very simple type of flip – flop is D- type flip- flop as shown in Figure 18.

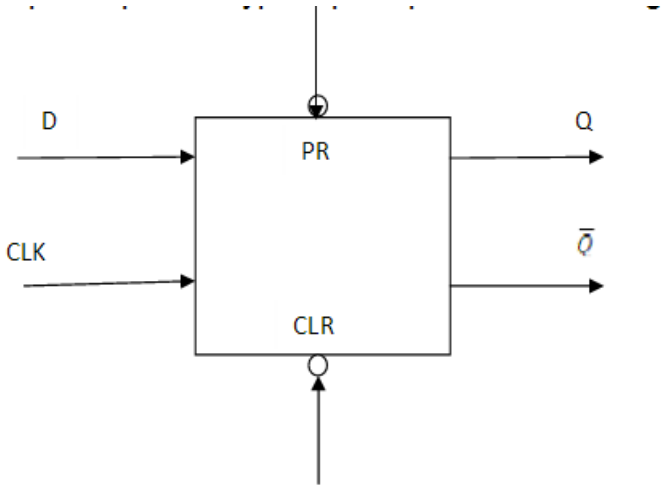


Figure 18:D-Filp flop

It has a single data input D and two input, Q and $\bar{Q}$. Output Q represents the state of flip- flop; $\bar{Q}$ represent the complement of the flip- flop's state. The logic value at a flip- flop's D input when a clock signal, CLK occurs is stored in the flip- flop. If the store value is equal to 1 (Q = 1) the flip– flop is set. If the stored value is equal to 0 (Q = 0) the flip – flop is clear.

The logical operation of a D type flip – flop is expressed by the characteristic equation $Q_{n+1} = D_n$. This equation indicates that the output of a D- type flip – flop after the accordance of a clock pulse, Q_{n+1} is equal to the logic value of the D – input before the accordance of the clock pulse D_n.

But D – type flip-flop differ with regard to the praise time at which the clock pulse causes the input data to be accepted, the output to change in accordance with the input, and the output to be

held or latched. Two clock pulse or strokes are shown in fig positive clock pulse. This signal is logic 0 in its quiescent state, makes a transition to logic 1 remains at logic 1 momentarily, and then returns to logic 0. The leading edge of the pulse is a 0 to 1 or positive transition and the trailing edge is a 1 to 0 or negative transition. The clock pulse shown is a negative clock pulse, its quiescent value is logic 1 and it makes a momentary negative transition to logic 0 followed by a positive transition back to logic 1. A positive transition is also referred to as a using edge, and a negative transition is also referred as the following edge.

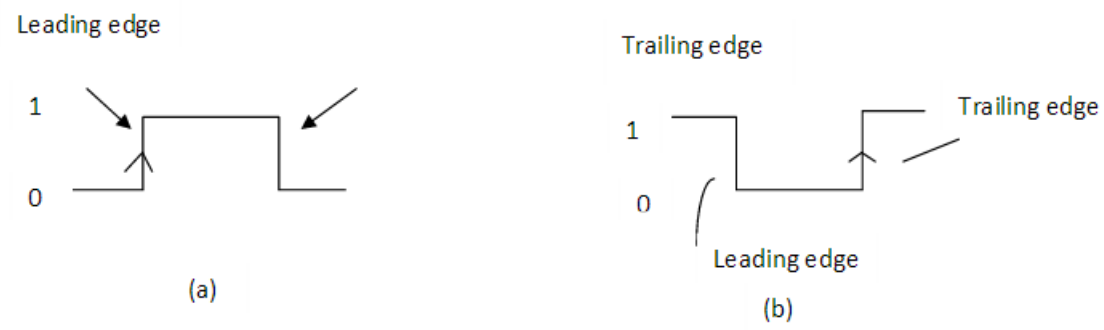


Figure 19: Leading and trailing edge of a pulse

An edge triggered D – type flip-flop latches the logic value at the D input during the clock pulse's transition from one logic value to the other. The sensitivity of the flip-flop to the transition (edge) of the clock is indicated on the flip-flop logic symbol by a dynamic indicator, a triangle <, at the clock input. Positive edge triggered flip – flops latch on the positive transition of the clock. Negative edge triggered Flip-flop/ latch on the negative transition of the clock. If the clock pulse of fig (a) is applied to the +ve edge pulse triggered flip-flop, the data is latched at the trailing edge of the pulse. If the clock pulse of (b) is applied to a +ve edge triggered flip-flop, the data is latched at the trailing edge of the pulse. Note that in the edge triggered flip-flop, the input is accepted, and the output changes and is latched during a single clock transition.

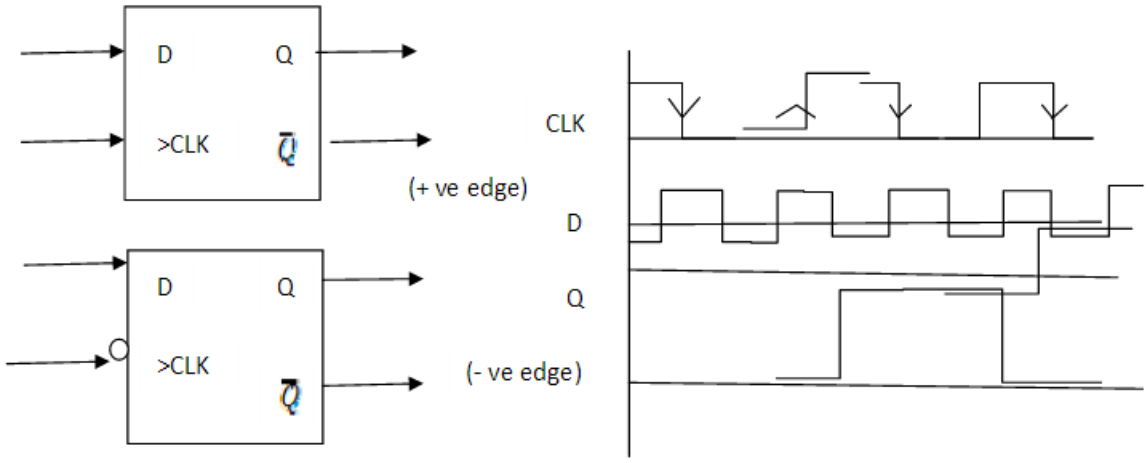
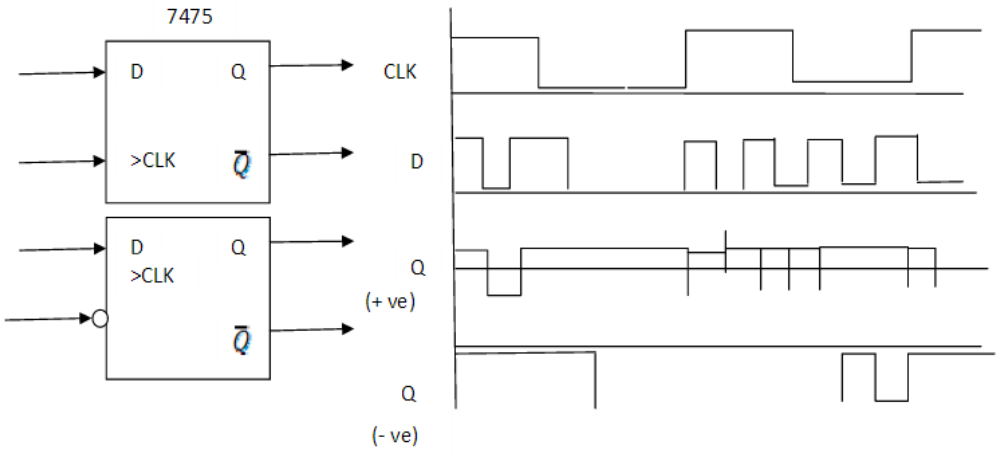


Figure 20: +ve and –ve edge triggered flip flop

A level triggered flip-flop usually referred to simply as a latch has a clock input that is sensitive to the level of the clock signal. The output of a positive level triggered D flip-flop follows the D input when the clock signal is high. When the edge makes a transition from 1 to 0, the data present at the D input is latched. The output of a negative level triggered D flip-flop follows the input. When the clock is logic '0' and latches the input on a 0 to 1 transition. Thus for a level triggered flip-flop, the output follows the input, when the clock is at the trigger level. During this condition the flip-flop is referred to as being the input data is latched on the transition from the trigger level to the quiescent level.



The D flip-flop shown two additional inputs common to most ICs preset & clear. Both the input is active low signals. Preset & clear are asynchronous input; they affect the state of the flip-flop

independent of the clock's level or transition. Thus preset & clear have over side influence on clock & synchronous input. Logic '0' at the clear input clears it for proper operation preset & clear are not absorbed simultaneously.

2.3 Examples of latches

Typical examples of transparent latches are 74LS373 & the Intel 8282 shown in fig. Both are functionally similar; however, they are not pin compatible. These octal latches are suitable to latch 8-bit data.

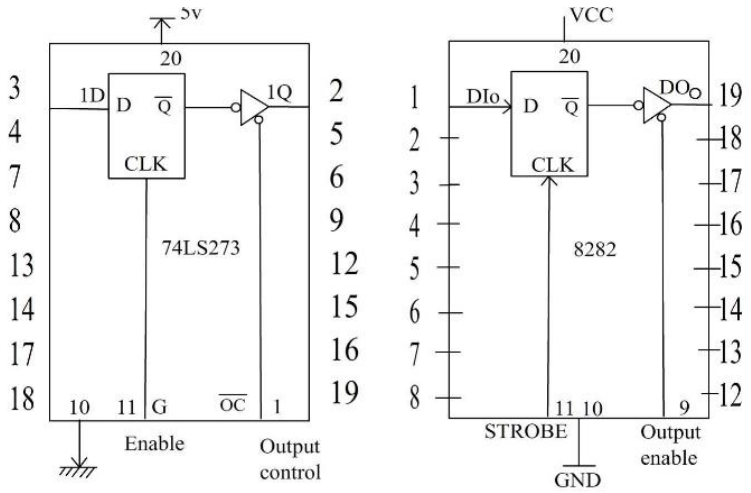


Figure 21: Latches

Function table

Output control	Enable G	Input	Output
L	H	H	H
L	H	L	L
L	L	X	Q
H	X	X	Hi edge

Output enable	STB	Input DI	O/P
L	H	H	H
L	H	L	L
L	L	X	D type latch
H	X	X	L

These devices include eight D latches with tri-state buffers. They require two input signals, enable (G) &($\bar{O}E^{-}$) the output control for the 74LS373 which are synchronous to the stroke (STB) & the output control (OE^{-}) for the 82072. The enable is an active high signal connected

to the clock signal input of the flip-flop. When this signal goes low data are latched from the data bus. When the output control is low (active) the data latched is accessible to the display devices.

2.3.1 m- Bit register

Memory is storage device and it is used to store both instructions and data. The smallest unit of information a digital system can store is a binary digit which has a logic value of '0' and '1'. A bit of data is stored in a flip flop. It is general memory cell and has two states in which it can remain indefinitely as long as power is not interrupted. The output can be changed only if the output signals allow for it. D flip flop is the simplest flip flop to store one bit of information. To store several bits of data simultaneously, the clock inputs of several D flip-flops are connected in parallel to form an m bit register (m may be 4, 6 or 8). Such registers store m bits of data D_0 to D_{m-1} under the control of the clock and provide m bits of data output Q_0 to Q_{m-1}.

To store m-bits of data simultaneously, the clock input of several D- flip-flops are connected in parallel to form an m- bit register (m may be 4, 6 or 8). Such register store m bit of data D_0 to

D_{m-1}under control of the clock and provide m data outputs O_0 to O_{m-1}. The act of storing data in a register is a writer operation. Determining the value of the content of a register is a read operation.

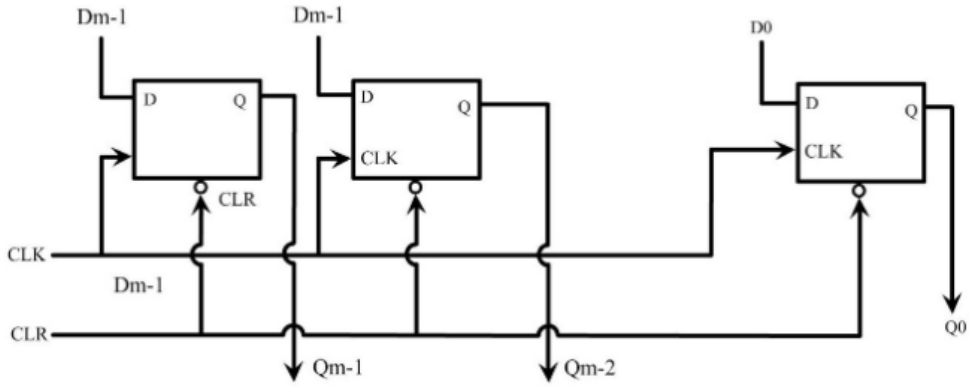


Figure 22: m-bit register

The m- bit of data stored in a register make up a word. A word is simply a number of contiguous bits operated upon or considered by the hardware as a group. The number of bit in the word m is a word length. The m input to the register are provided by an m- bit input data bus and the m-

output by an m- bit output data bus. A bus is a number of signal line grouped together because of similarity of function, which connect two or more systems or subsystem. Eight bit register are often called octal registers (examples). Several equal length registers can be incorporated in a single IC and share a common set of inputs, a common set of inputs and a single clock such as circuit is referred to as a memory. Each register occupies a distinct location, which has a unique numerical address. Thus, memory can be thought of as a collection of addressable registers. Logic is necessary to decode address inputs to ensure that only a single register outputs its contents when data is being read from the memory, and only a single register has data stored in it when the data is being written into the memory.

2.3.2 Development of Memory Chip

Let us consider a memory of 16 words, each of 8 bits is to be stored. 16 words can be stored in 16 memory locations, each having a unique 4 bit memory address (0000 to 1111) and each location being capable of containing 8 bits of data.

To set up this memory system using ICs, 16 bit flip-flop registers are required. To identify the correct address a 4 line-16 line decoder can be used to decode the 4 bit location address to select the appropriate data register (1 to 16) for input/output. Figure shows the circuit used to implement the memory system.

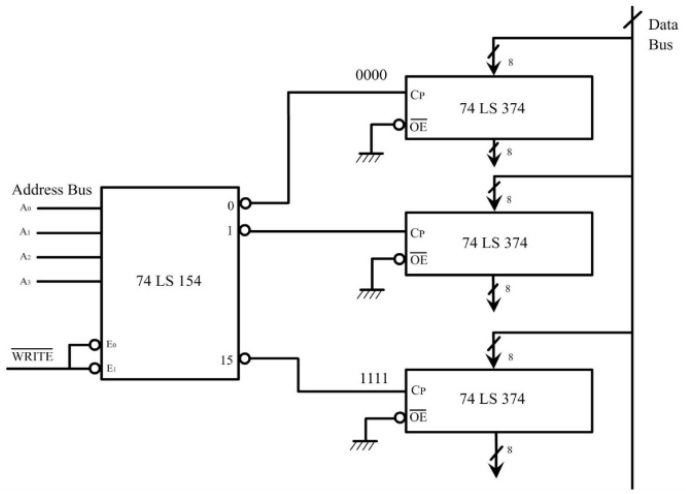


Figure 23: Memory system

The 74LS374s are octal D flip flops with three state of outputs. To store data in them, 8 bits of data are put on the D_0 to D_7data inputs via the data bus. Then the low to high edge on the clock input will cause the data a D_0 to D_7 to be latched into each flip-flop. the stored value in the D flip-flop is observed at the outputs Q_0 to Q_7 by making the output enable ($\overline{OE}$) pin low. To select the appropriate memory location, a 4 bit address is input to 74154 (4 Line to 16 Line decoder), which outputs a low pulse on one of the output lines when $\overline{W}\overline{R}\overline{I}\overline{T}\overline{E}$ the is pulsed low. The timing of setting up the address bus, data bus and pulsing the $\overline{W}\overline{R}\overline{I}\overline{T}\overline{E}$ line is critical. The following **Figure 24** shows standard timing diagram bus driven devices. Rather than showing all four address lines and all data lines, they are grouped and X is used to show where any or all of the lines are allowed to change digital levels.

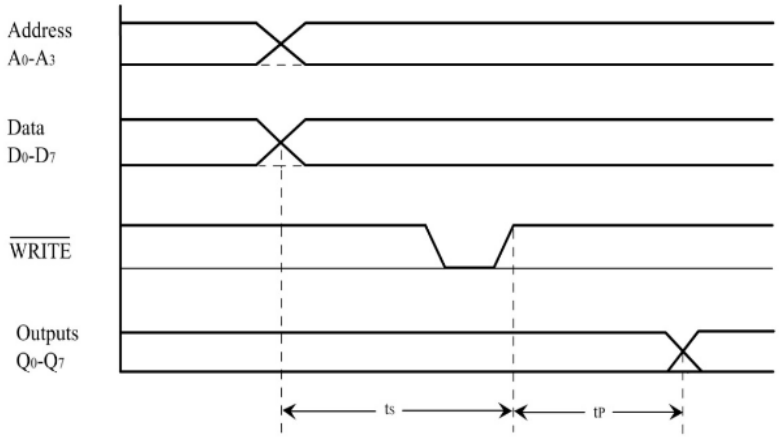


Figure 24: Timing diagram

The address and the data lines must be set up some time (t_s) before the low to high edge of $\overline{WRITE}$. In other words the address and the data lines must be valid some period of time (t_s) before the low to high edge of $\overline{WRITE}$ in order for the74374 to interpret them correctly.

When the $\overline{WRITE}$ line is pulsed, the decoder outputs a low pulse on one of its 16 outputs which clocks the appropriate memory location to receive data from the data bus. After the propagation delay, (t_p) the data output at $Q_0$to Q_7 will be the new data just entered into the D flip-flop. Then the t_p will include the propagation delay of decoder and of the D flip-flop.

In the figure all the three state outputs are continuously enabled so that their Q outputs always active. To reduce the number of lines the 8 outputs Q_0 to Q_7 of all 16 memory locations back to

the data bus. The $\overline{OE}$ enables of the 16 memory locations have to be individually selected at the appropriate time to avoid a conflict on the data bus, called bus contention. Bus contention occurs when two or more devices are trying to send their digital levels to the shared data bus at the same time. To individually select each group of Q outputs, the grounds on the $\overline{OE}$ enables would be removed and instead be connected to the output of another 74154 1 of 16 decoder.

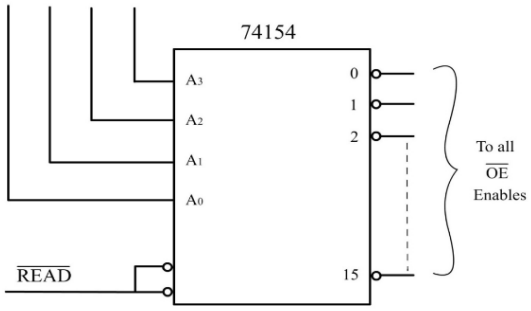


Figure 25: 1 to 16 decoder

2.4 Memory

If a memory stores N- words of information each word being of m bits, we say it is a Nxm memory. e.g. 8x4 memory means there are 8 words 4 each word containing 4- bit of information (called nibble). 8 words are stored at 8-memory locations and these memory locations are clearly identified by addresses. Addresses are formulated by bit combinations available in wires known as address lines. To identify 8 memory locations we require 3 address lines designed $A_2A_1A_0$. The memory locations identify and the corresponding content stored is shown in Table 2.

Table 2: The memory locations and the corresponding content

A_2	A_1	A_0	Decimal Equivalent	Memory Location	Contents of the memory location
0	0	0	0	0	M(0)
0	0	1	1	1	M(1)
0	1	0	2	2	M(2)
0	1	1	3	2	M(3)
1	0	0	4	3	M(4)
1	0	1	5	5	M(5)
1	1	0	6	6	M(6)
1	1	1	7	7	M(7)

M(0) is the content of memory location '0'. It has 4 bits here. M(1) is the content of memory location '1' and so on. It can also be shown us.

$\frac{3(AB)}{(A2-A0)}$ AB ----- Address Bus

The three address lines $A_2A_1A_0$ together is known as address bus. It is a unidirectional bus. The microprocessor always sends the addresses.

In general, a N x m memory shall have K address lines designated $A_{k-1}A_{k-2}A_{k-3}$..... $A_2A_1A_0$, such that K is the smallest integer satisfying the inequality $2k \geq N$. e.g. 200x8 memory shall have

200 memory locations 8 bit of information. To identify 200 memory locations we require a minimum of K= 8 lines designated A7A6A5A4A3A2A1A0. However K = 8 address lines can identify a total of 256 memory location starting from $(0000\ 0000)_2$ to $(1111\ 1111)_2$ or 00_H to FF_H. But we are using only 200 memory locations and rest locations are redundant. The 200 memory location shall be identified starting from $(0000\ 0000)_2$ to $(1110\ 0111)_2$ or 00_H to $E7_H$. The other combinations 1110 1000 B to 1111 1111 B or $E8_H$ to FF_H are not used in this memory & are redundant addresses.

Since it is too tiring & boring to use binary numbers for identifying the addresses we normally make use of hexadecimal number notation. E.g. 200 memory location are identified starting from (00)H to (C7)H, (C0)H to(FF)H are redundant memory locations. Using 10 address lines designated $A_9A_8A_7$.....$A_2A_1A_0$, we can directly address 2^{10}= 1024 memory locations. This is conventionally known as 1K memory locations.

The capacity of a memory is specified terms of the maximum number of words the memory can store. In general, if the memory has an R bit address and each word is of length m, then the memory has a capacity of 2^{R*}m bits, organized as 2^Kwords each of m bits. If R=10, then the memory can store 1024 words or 1K words. In most of the 8 bit Microprocessor the Microprocessors has 16 address lines. Therefore it can address directly

2^{16}memory locations = 2^0x 2^{10}memory locations

= 64 K memory locations

= 65536.

Thus, 8-bit microprocessor provides a maximum of 2^{16} or 64 K memory addresses ranging from 0000 to $FFFF_H$.

2.4.1 Characteristics of Memory

An important characteristic of a memory is whether it is volatile or non volatile. The contents of volatile memory are lost if the power is turned off. On the other hand, a non volatile memory retains its contents after the power is switched off. The best known non volatile memory is magnetic core.

In the broad sense, a Microprocessor's memory system can be logically divided into three groups:

1. Processor memory
2. Primary or main memory
3. Secondary memory

Processor memory refers to a set of Microprocessor registers. These registers hold temporary results when a computation is progress. There is no speed disparity between these registers and the Microprocessor because they are fabricated on the same chip using the same technology.

Primary memory is the external memory to store both program and data. The Microprocessor can access their memories directly. In earlier days, the primary memory was designed using magnetic cores. In modern Microprocessor s, MOS technology is employed in the primary memory design. Usually, the size of primary memory is much larger than the processor memory and its operating speed is slower than that the processor registers by a factor of 25 or 30.

Secondary memory refers to the storage medium compositing slow devices such as hard disks and floppies. These devices are used to hold large data and huge program that are not needed by the processor frequently. Sometimes, secondary memories are also referred to as auxiliary or back up storage.

In order to design an efficient memory system, the following characteristics of memory must be known. The most important factor of a memory system is its cost, expressed in dollars per bit. A good design implies a very low cost per bit.

There are two parameters that will indicate the speed with which information can be transferred in and out of a memory.

1. Access time, t_A
2. Cycle time, t_C

The access time t_A is defined as the average time taken to read a unit of information from the memory. Sometimes the access time is also referred to as read access time. Similarly, one can define write access time. Usually, the write access time will be equal to read access time. The cycle time t_C of a memory unit is defined as the average time lapse between two successive read operations.

The reciprocal of access time is called the access rate ($r_A=1/t_A$), which is expressed in bits per second. Similarly, the reciprocal of cycle time is referred to as date transfer rate or bandwidth also expressed in bits per second.

The third important characteristic of memory unit is its access mode. The access mode refers to the manner in which information can be accessed from the memory. There are two major access modes. They are the random access mode and sequential access modes. In random access mode, any memory location access time is independent of the location from which the date is read. In sequential access mode, the memory is accessed strictly in a sequential manner. In this mode, the access time depends on the location in which data is stored. They are also referred as serial access memories. A bipolar memory and magnetic tape are typical example for random & serial –access memories.

Random access memories than its sequential access memory 2^{nd} order to active a compromise. Some memories combine both access modes and called semi-random memories. A typical example is magnetic access whit one read/write head for each track. This arrangement permits any track to be accessed. At random however, Access within a track must be made in a serial fashion.

2.4.2 Classification of Memory

In general, semiconductor memories can be clarified in two main groups' random access memories (RAM) and sequential access memories (SAM).

RAM can be classified in three main groups as shown below.

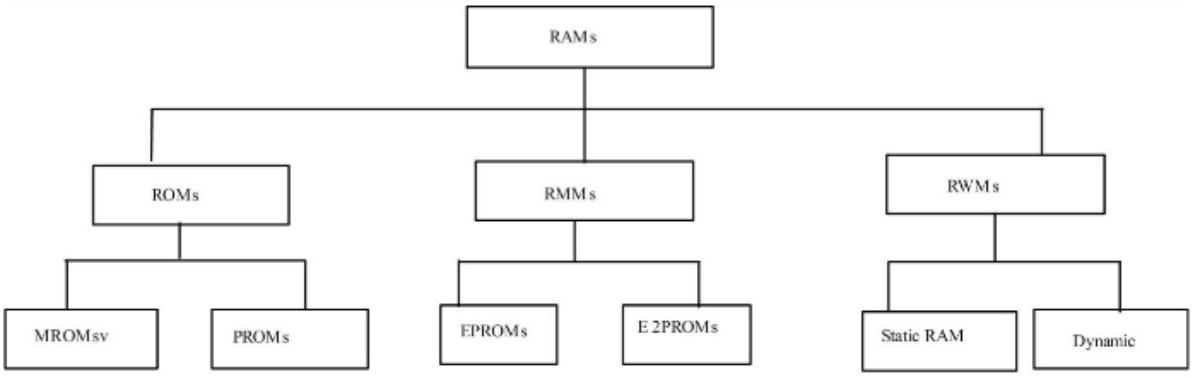


Figure 26: Classification of RAM

1. ROMS: The type of memory means the content of an address location can only be read and cannot be written into. The contents of the memory location are not destroyed whether the power is ON or OFF. Such a memory is known as non volatile memory. ROMs are used to store data on permanent basis. They are random access memories and this makes them very useful for the storage of computer operating systems, software language computers, look-up tables and programs for dedicated microprocessor applications. ROMs can only be read are not written into.

2. MASK ROMs: Mask ROMs are programmed by a masking operation performed on the chip during the manufacturing process. The contents of a ROM are decided by the manufacture. These contents are permanently stored in a ROM at the time of manufacturing. The contents of MROMs cannot be changed by the user. Most desktop computers use MROMs to contain there operating system and for execution fixed procedures, such as decoding the keyboard and the generation of characters for the CRT.

3. PROMs: If user needs relatively few ROMs, there is a variation, which cost more per devices but allows the user to in rest the information. To avoid the high one-time cost of producing custom mask ROMs, IC manufacturing provides user programmable ROMs. This device is called programmable Read only memory. Using special equipments, called PROM programmers, user can program a PROM- once. Subsequently one can read the information out of PROM as often as one wish, but one can never write into it again. Therefore once the PROM is programmed with correct information it can be used as a ROM only in microcomputer. If one needs to change or correct the information stored in the PROM, one must pull it out, throw it away and replace is with a fresh unused PROM, writing the new on corrected information into this inused devices. The PROM will hold its contents indefinitely. These PROMs are provided with fusible links which are burned during the programming. Once the data are permanently stored in the PROMs, it can be read again and again by just accessing the correct memory location.

4. EPROMs: If a mistake is done in programming ROM and PROMs, the correction cannot be made. The solution of this problem is erasable PROM (EPROM). An EPROM isan erasable PROM. The contents are erased by ultra violet light. Therefore, they are also called

5. UVEPROM: The user can not erase the content of a single memory location, the entire contents are erased. EPROMs can be reprogrammed using EPROM programmer. Once programmed, it can be used as ROM in microcomputer. Later, if one needs to correct the information stored, it is taken out from the system, erase the program written, write new program into it and use it. The EPROM is erased by exposing an open window in the IC to an ultraviolet light source for a specified length of time, typical erase time vary between two and 30 minutes, the EPROM programmed and providing proper addresses. An EPROM also holds the information indefinitely once it has been programmed. One can read the contents of an EPROM as often as one like.

6. RMMs: They are read mostly memories, since they have much slower write time then read time, there memories are usually suited for operation where mostly reading rather than, sorting will be performed.

7. E^2PROMs: E^2PROM are electrically erasable PROM. They need not to be removed from a microcomputer board for erasing. Erasing & programming E^2PROM is much easier as the ultraviolet sources are not required. The stored information can be erased by applying a high voltage of about 21V, a singly byte or the entire chip can be erased in10 mille sec. This is faster than UV erasing and it can be done easily while the chip is still in circuit, It is also known as EAPROM (electrically alterable PROM). One can write into at any time without erasing prior contents. The problems with EAROM are that electronically they are relatively difficult to use also, they slowly lose their information. One application of the E^2PROM is in the tuner of a modern TV set. The E^2PROM remember (i) the channel, you were watching when your tuned off the set (2) the volume setting of the audio amplifier.

8. RWMS: In this type of memory one can either read the contents of an addressed location in a MEMORY READ operation or one can write a m bit of data in the addressed memory location in a MEMORY WRITE operation. It is a volatile memory. It is normally known as RANDOM ACCESS MEMORY (RAM). The content of RWM shall be destroyed when the power is OFF. During of MEMORY REALD operation the content of the addressed location is not destroyed. It is only read onto the external data bus. During a MEMORY WRITE operation, however, the original content of the addressed location is destroyed and the new content takes it placed which is just now written. Read/write memories are used for temporary storage of data and program instruction in a up based septum there are also RAMs, RWMs are generally called RAMs, RAMMs a specific terms it tells that the data can be read on written to any memory location, RAM is classified as either static on dynamic, static RAMs (SRAMs) flip flops as basic storage elements; whereas the dynamic RAMs (DRAMs) use internal capacitor as basic storage elements, additional refresh circuitry is needed to maintain the charge on the internal capacitor of a dynamic RAM; they have more packing density, there it has more storage capacity per unit area team a static RAM, the cast per bit of dynamic RAMs is also much less than that of the static RAMs.

9. Non-volatile RAM: A Non-volatile RAM combines a static RAM and E^2PROM into the same chip. Such a device operates as a normal RAM. If power supply fails the entire

content of RAM are stored in E^2PROM by a single signal $\overline{STORE}$. A signal $\overline{RECALL}$ can transfer data from E2PROM back into the RAM. E.g. x2201 is a non-volatile RAM of 1K bit. The transfer time is 4msec.

The symbolic diagram of a static RAM and ROM are shown below:

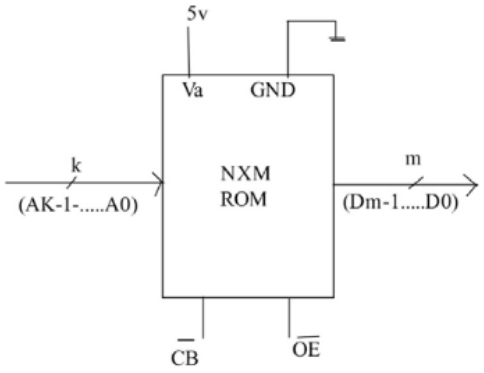


Figure 27: Symbolic diagram of ROM

$\overline{CB}$= Chip selection to control input.

$\overline{OE}$= output enables signals.

Both the control signals are active low (in general) but they may be active high, when the chip is selected then only addressed memory location data is available on data lines provided $\overline{OE}$ is active.

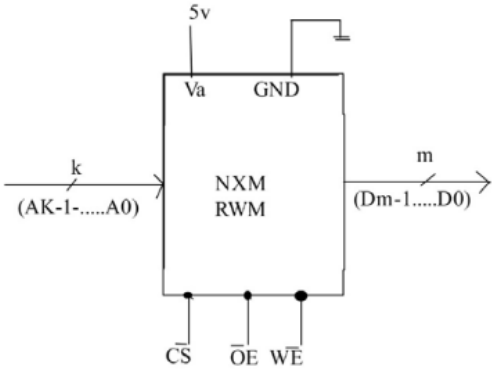


Figure 28: Symbolic diagram of RAM

$\overline{CS}$= Chip selection to control input.

$\overline{OE}$= output enables signal

$\overline{WE}$= write enable signal

These are representative signal. These signals may be LOW or HIGH for other ROMs. For READ operation, first address is placed and then make $\overline{CS}$= LOW, the output is available. Under WRITE operation $\overline{OE}$ may be HIGH or LOW but $\overline{WE}$ is active LOW. Under READ operation $\overline{OE}$ is ACTIVE LOW but $\overline{WE}$ must be HIGH.

All these memories are random access memories. In RAM any memory location can be accessed in a random fashion without regard to another location. The access time is same for each memory locations.

2.5 INTEL 2716 EPROM

This is an Ultra Violet Erasable Programmable ROM {UVEPROM}. The pin connection & logic symbolism is shown in Figure 29 and Figure 30 respectively. It is a 2Kx8 ROM. It has 2048 memory. It has 11 address lines. While programming V_{pp} must be held at 25 volts. It this chip is used in a microcomputer after programming this voltage must be held at +5V.

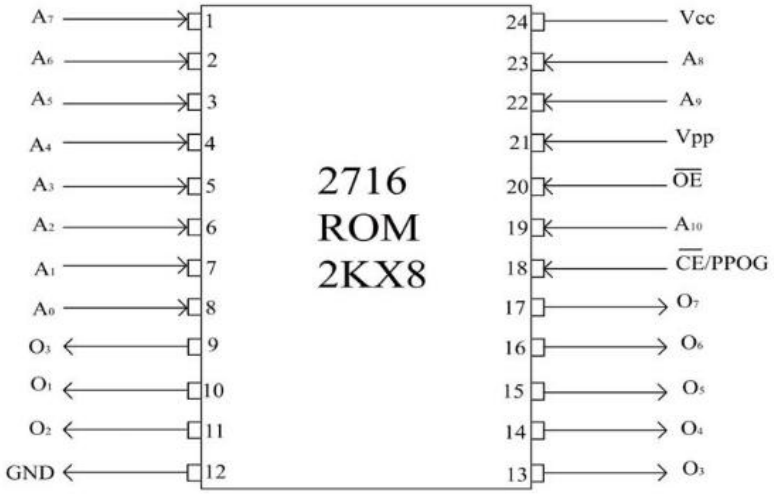


Figure 29: Intel 2716 pin diagram

The pin details are given below:

V_{CC}, GND = +5V and Ground

A_{11}-A_0 = address lines

D_7-D_0=data lines

V_{PP}=Programming voltage

$\overline{OE}$= output enables (to enable the output data buffer)

$\overline{CE}$/PROG= dual function pin. While programming HIGH pulse is applied at this pin and during read operation the chip is selected enabled by making $\overline{CE}$ pin low.

When it is completely erased then each bit must be '1'. If we want to store '0' we write '0' there.

Before programming each address stores FF_H but to store any data at the addressed location a 50ms pulse is given to the PROG.

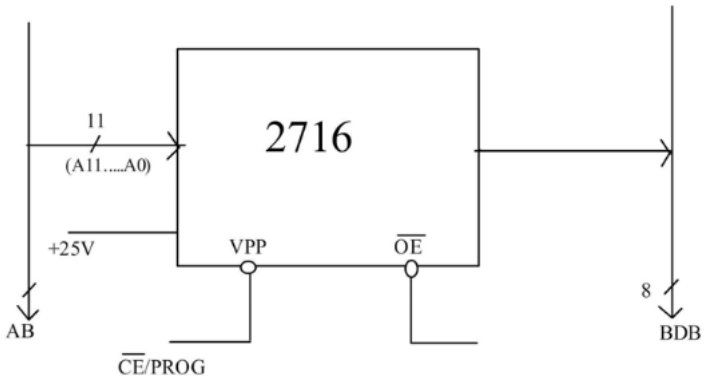


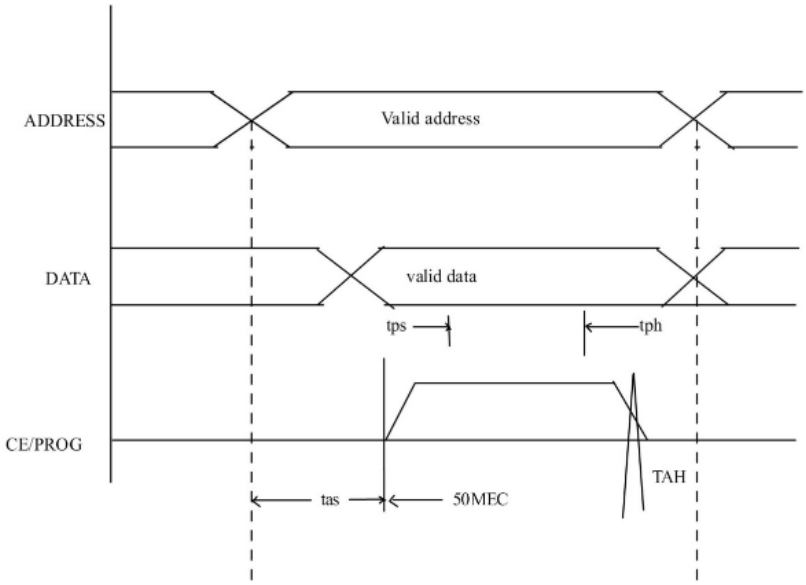
Figure 30: Logic symbolism of Intel 2716

For programming 2716 is connected as shown in figure and following operations are done in sequence:

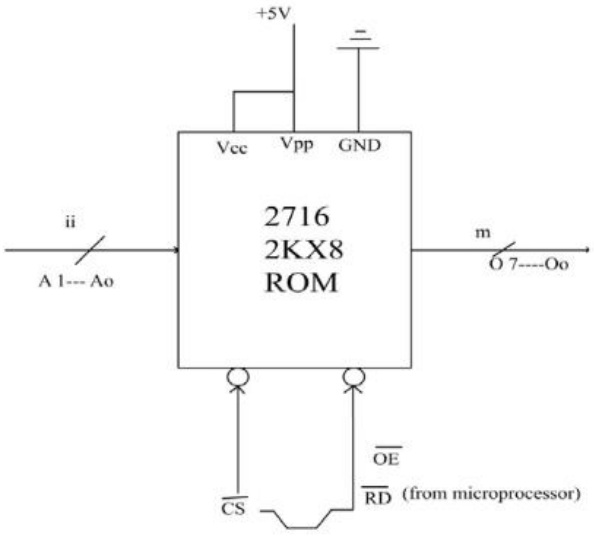
1. Apply 25V dc to pin no 21(V_{pp}).
2. Keep the (output enable bar) $\overline{OE}$ high (+5v).
3. Establish the address at the address bus.

4. Established the desired data to be stored at the addressed location on then data bus.
5. A positive TTL pulse of 50msec duration is applied to the pin no. 18 ($\overline{CE}$/PROM).

The waveforms during programming are shown in figure below:



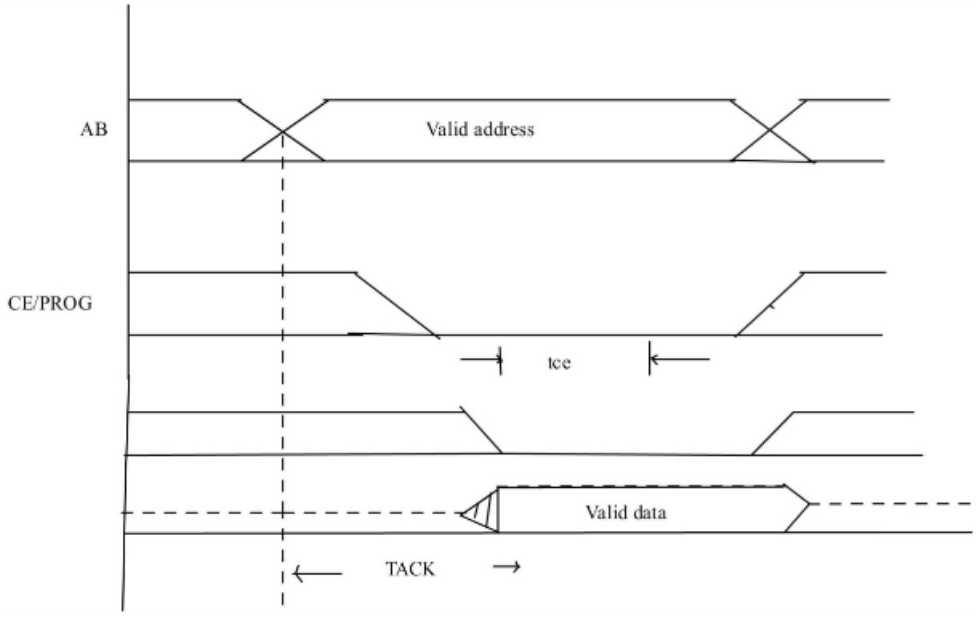
The above procedure is repeated for all location to programme all the 2K memory location. One can programme 2716 partly as required. All the above actions are carried out in separate unit known as EPROM programmer. It requires only 100sec to programme all the memory locations. Once the programme is written down in memory chip, it cannot be erased. If we want to change it we put it in UV eraser and erase it and then programme it again.



Once the chip is programmed, it can be used to read data again as again. When the two inputs $\overline{CE}$ and $\overline{OE}$ are in their normal state (HIGH) the output is tri-stated. Now one can only perform a MEMORY READ operation from this device. The following is the procedure for a MEMORY READ operation.

1. Establish the addresses of the memory location to be read on the address bus.
2. Make the $\overline{CE}$ signal ACTIVE LOW.
3. Apply a $\overline{RD}$ control signal to $\overline{OE}$ terminal i.e. make the $\overline{OE}$ ACTIVE LOW.

$\overline{RD}$ is normally HIGH by the microprocessor and to read date low is generated. This is known as MEMORY READ operation. The waveforms during read operation are shown in figure below:



The wave forms of the chip show that the data out puts became valid after a delay for setting up the addresses on enable the chip on inability the output whichever is completed last.

2.7 INTEL 6116 RAM

It is 2Kx8 memory. It is a static RWM (2Kx8). This is pin by pin compatible with 2716 ROM. The pin connection is shown in Figure 31 and the logic symbolism is shown in Figure 32.

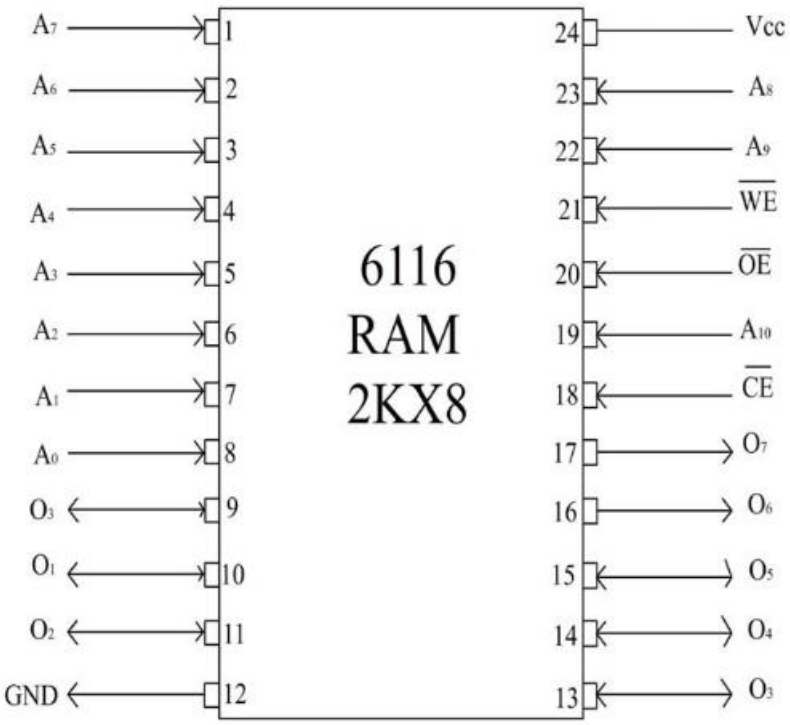


Figure 31: Intel 6116 RAM

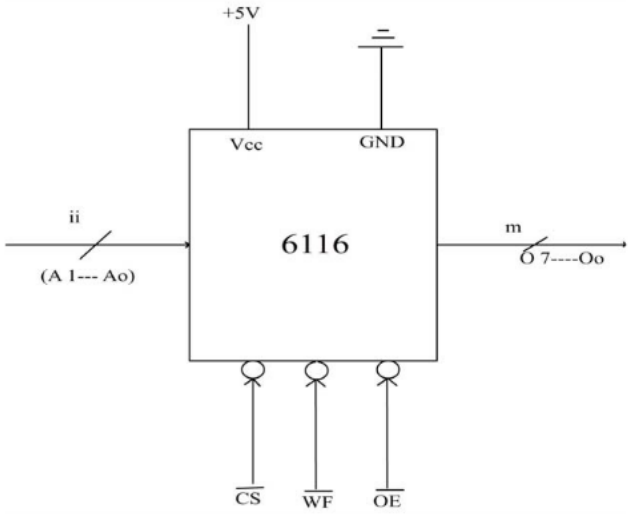


Figure 32: Logic symbolism of Intel 6116 RAM

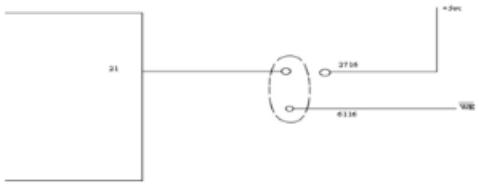
The truth table for control signals is as follows:

Table 3 Table for control signals

$\overline{CE}$	$\overline{WE}$	$\overline{OE}$	Operation	REMARKS
0	0	X	WRITE	The data available on the data bus shall be written on the addressed location. The original contents are lost. The new labes it place
0	1	0	READ	The content of the addressed location is READ on to the output data line 0-00. The content other addressed location is not destroyed.
0	1	1	No operation *tri stated	Output is Tri state.
1	*	*	No operation tri stated	Output is Tri state.

The two chips 2716 and 6116 are pin by pin compatible and can be used in place of other. The pin by pin compatibility of 6116 with 2716 has a advantage. In the initial stage of a programme development we fix up 6116 in the 24 pin socket provided on the microcomputer and develop the programme. After a lot of effort we are ready with a permanent programme. Once the programme is completely tested and satisfactory then using a PROM programmer the programme can be transferred to 2716 ROM for permanent storage.

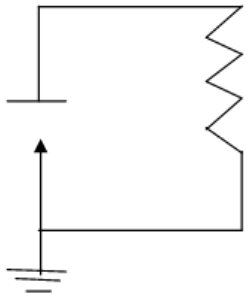
Thereafter, 2716 can be put directly to the same socket occupied by 6116. A simple jumper should be provided for pin no 21. This is shown in figure below.



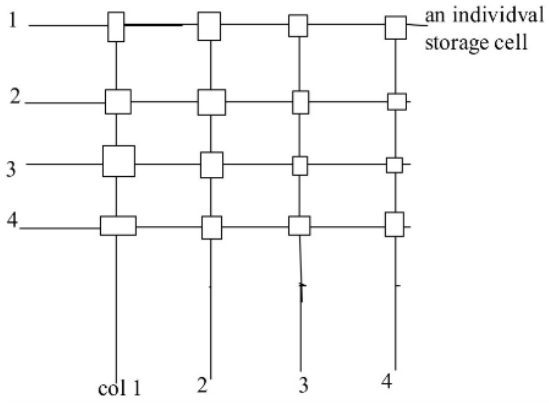
2.6 Dynamic RAM chip

A dynamic RAM comprises storage cells that may be thought of eclectically as capacitors there are many thousands of these capacitors, or storage cells on a dynamic RAM chip, each all is capable of storing one bit of information.

The capacitor that makes this storage cell is not ideal. That is, change placed on this capacitor will leak off given enough time. In a DRAM, the change on the capacitor represents the stored data. Therefore, the data stored in the cell can be lost. A more accurate model of the storage cell is a capacitor in parallel with a resistor.

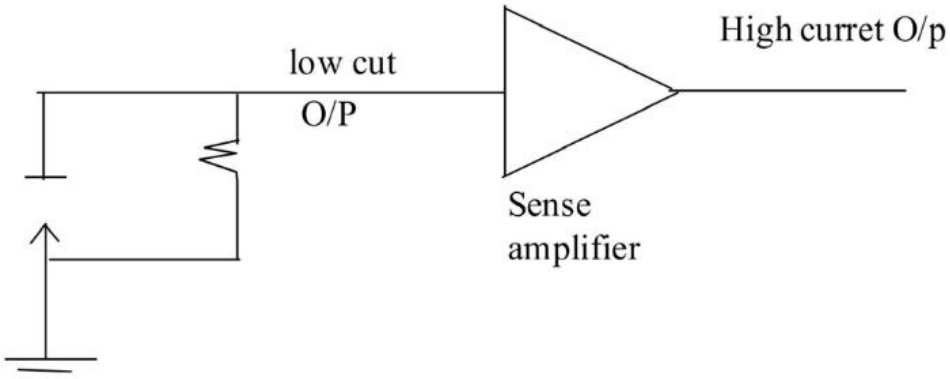


To keep the stored information in a cap cell, DRAM is refreshed again & again. In a dynamic RAM, the storage wills are organized in a matrix form. Fig shows, the organization of 16cells in a matrix of 4 rows & 4 columns. Each cell in a matrix has a unique position specified by the intersection of a row& a column.

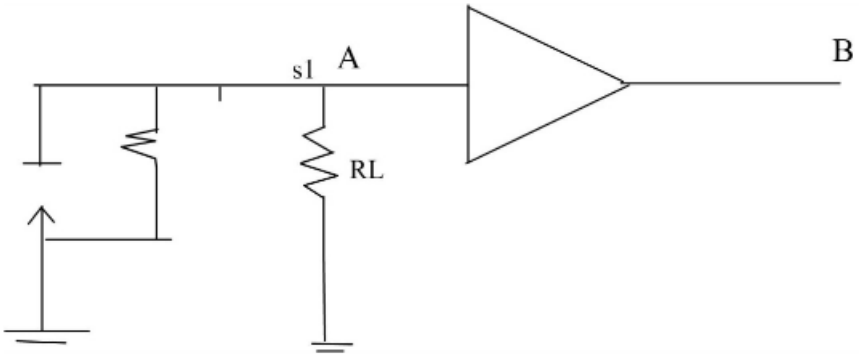


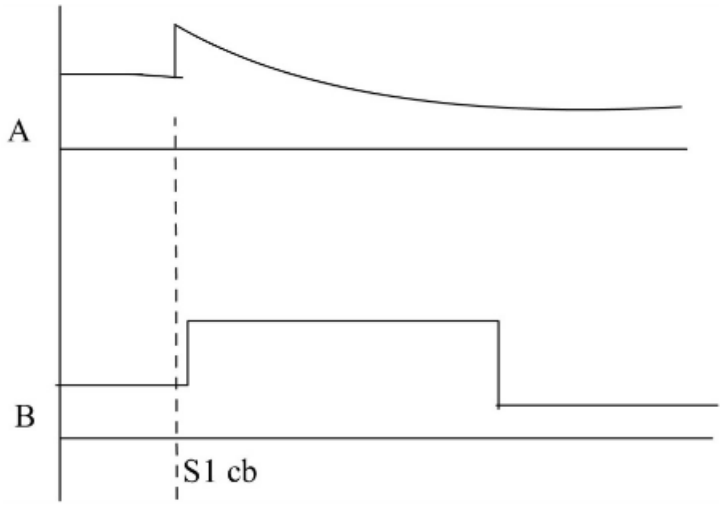
Using Row & Column, any cell can be uniquely identified. External signal lines are used to indicate which storage cell in the internal matrix is to be accessed. These lines are called row address lines & column address lines.

Due to the way in which DRAMS are fabricated, the storage capacitor is not capable of providing large Q/P current to an external load. Therefore a circuit called sense amplifier is placed at the output of the storage cap to increase the Q/P current drive capability of the cell.

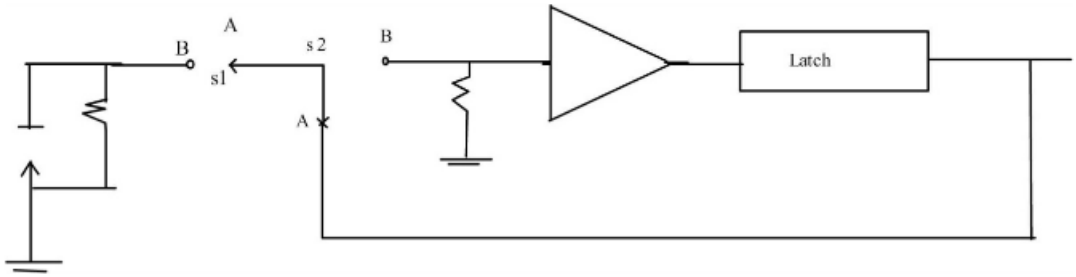


Solid state switch S, is fabricated in the DRA chip to isolate the storage cell from all circuit external to that cell $S_1$is closed when the row address & etc. address corresponding to the storage cell are selected upon clearing s1, the change stored on the capacitor flame through s1 & R_L. Current flawing through R_L causes a voltage drop arose R_L, the voltage developed across R_L is increased by the sense amplifier to a lever suitable for during an external load (which is usually TTL),





When $S_1$is closed, current flews thought r1 and change on the capture is lost, this type of read is called destructive read operation the data should not disappear in read operation. A latch is used to refresh the stored data as shown in fig.



2.7 4116 dynamic RAM chip

4116 is a 161kv dynamic RAM, the chip has 16,384 storage cells in its matrix from and as commonly referred to as a 16kk X 1 bit DRAM, there are 16k unique memory location, storage cell A block diagram of the chip is shown below;

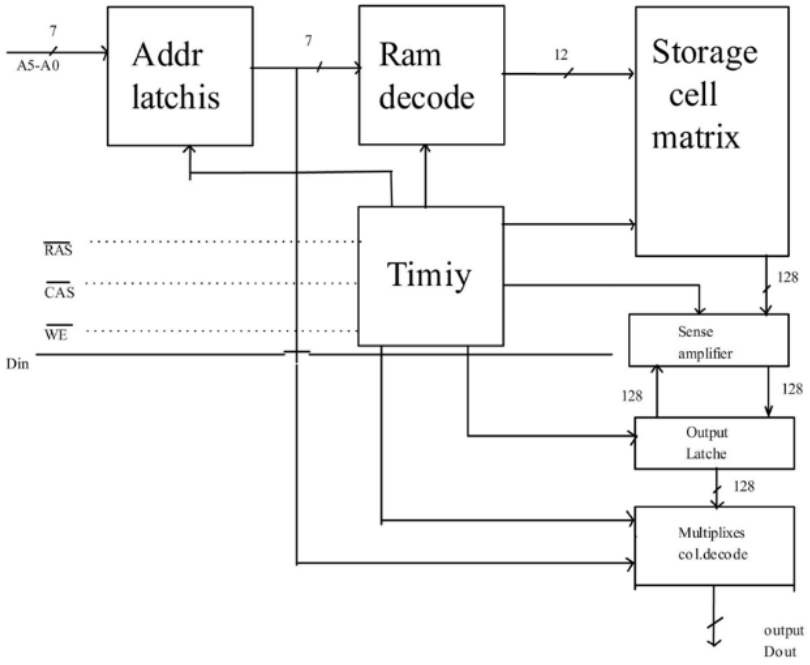


Figure 33 Dynamic RAM chip:

Storage cell matrix block: There are 16,384 storage cell in the 4116 organized in a cell contains a storage cap & an.ssw which isolates the cap from the rest of the incant in DRAM.

Address latch block: Seven address lives A1-AO are input to this block, after proper time, an address is latched in the address latch block, the output of this block is input to both the how decode & the col multiplexer block.

Row address Recorder block: The block have seven input line &128 output lines, seven output line represent one now of the storage matrix cell.

Sense address Decoder block: The output of the col of cells in the storage matrix are tied together on a single line celled a bit line; there are 128 bit line in the matrix and 128 sense amplifier.

Data latch block: After data is read from a now of cells, it is mitten into the latches in this block.

Column Multiplexer block: Only are of 128 line input to this block is switched through to the DOUT output lines the seven address signals input to the DRAM, chip determine which one is selected.

Time block: The sequence of events that take place within the DRAM is determined by this block also provides control signal to most of the other block the DRAM.

Pin out of 4116:

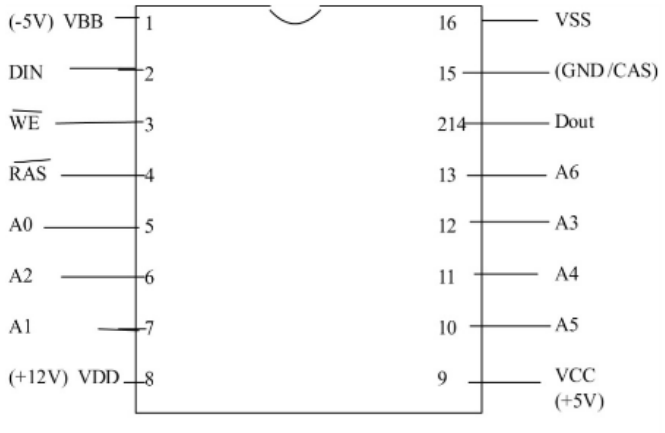


Figure 34: Pin out of 4166

A_6-A_0: The up accesses are memory cell by outputting the row& column address on these seven lines.

$\overline{C}\overline{A}\overline{S}$: When the up has output the col address on the A5-A0 lines, the $\overline{C}\overline{A}\overline{S}$ line is assented internally.

$\overline{R}\overline{A}\overline{S}$: When $\overline{R}\overline{A}\overline{S}$ is asserted, the read add on AR-AO line is latched internally.

DIN: Data is stored in the cap cell by making this line ‘o’ or 1. After the row & ecol address are latched internally, the up writer to the cell by places data on this line.

DOUT: Data is read from the RAM chip their line, after the row & col, address are latched internally, the selected cell entreats are output to the line.

2.8 Summary

1. There are two kind of memories; semiconductor memories & magnetic memories.
2. Semiconductor memories are faster, smaller, and lighter and consume less power.
3. Magnetic memories are slow but they are cheaper than semiconductor memories.
4. The smallest unit of information a digital system can store in a binary digit which has a logic value of 0 or 1.
5. A flip – flop is a general memory and has two stable states in which it can remain indefinitely as long as the operating power is not interrupted.
6. To store m-bits of data simultaneously, the clock input of several D- flip-flops are connected in parallel to form an m- bit register (m may be 4, 6 or 8).
7. The 74LS374s are octal D flip flops with three state of outputs.
8. Primary memory is the external memory to store both program and data.
9. Secondary memory refers to the storage medium compositing slow devices such as hard disks and floppies.
10. A dynamic RAM comprises storage cells that may be thought of eclectically as capacitors there are many thousands of these capacitors, or storage cells on a dynamic RAM chip, each all is capable of storing one bit of information.

2.9 Check Your Progress

1. Memory is storage device and it is used to store both _____ and _____.
2. Semiconductor memories are used as the _____ of a computer.
3. Magnetic memories are used as the _____ of a computer for bulk storage of data and information's.
4. A bit of data is stored in electronics devices called a _____ .
5. A _____ is simply a number of contiguous bits operated upon or considered by the hardware as a group.
6. The contents of _____ memory are lost if the power is turned off.
7. A _____ memory retains its contents after the power is switched off.
8. The reciprocal of access time is called the_____ .

9. The _____ refers to the manner in which information can be accessed from the memory.

2.10 Answers to Check Your Progress

1. Instructions, data
2. Main memory
3. Secondary memories
4. Flip–flop
5. Word
6. Volatile
7. Non-volatile
8. Access rate
9. Access mode

2.11 Model Questions

1. What is the difference between semiconductor memory and magnetic memory?
2. What is flip-flop?
3. Explain the working of D-Flip flop.
4. Explain the working of m-bit register.
5. Define access time and cycle time.
6. What is access mode? What are the different types of access modes?
7. Explain the classification of memories.
8. Draw the symbolic diagram of a static RAM and ROM.
9. Explain the functioning of INTEL 2716 EPROM with the help of a pin diagram.
10. Explain the functioning of INTEL 6116 RAM with the help of a pin diagram.
11. What is dynamic RAM? Explain the working of 4116 dynamic RAM chip.

CHAPTER III: ARCHITECTURE OF 8085 MICROPROCESSOR

3.0 Learning Objectives

After reading this chapter, you will be able to:

- Know about the 8085A microprocessor
- Explain the pin diagram of 8085A microprocessor
- Know the signal groups of Intel 8085 microprocessor
- Understand the Internal Architecture of 8085 microprocessor
- Know the Register Set of 8085 microprocessor
- Understand the working of DMA
- Differentiate between maskable and non-maskable interrupts
- Explain priority order of the interrupts

3.1 Introduction to 8085 Microprocessor

Intel launched its first 8-bit microprocessor in 1972 and names it Intel 8008. Soon after, it improved version, Intel 8080 was launched. Finally Intel launched 8085 in 1977, which was much powerful then its two earlier versions. Intel 8985 is an 8-bit general purpose microprocessor capable of addressing 64K memory. It has 40 pins and runs on +5 V power supply. It operates with 3 MHz clock. In 8085, the 8-bit data bus was multiplexed with the lower part of 16-bit address bus so that the number of pins are limited to 40. It has a 16-bit address bus, hence it is capable of addressing 2^{16}= 64 KB memory.

It consists of:

- **Control unit**: control microprocessor operations.
- **ALU**: performs data processing function.
- **Registers**: provide storage internal to CPU.
- **Interrupts**
- **Internal data bus**

The pin diagram of Intel 8085 is shown in Figure 35. The pins on the chip can be grouped into 6 groups:

- Address Bus.

- Data Bus.
- Control and Status Signals.
- Power supply and frequency.
- Externally Initiated Signals.
- Serial I/O ports.

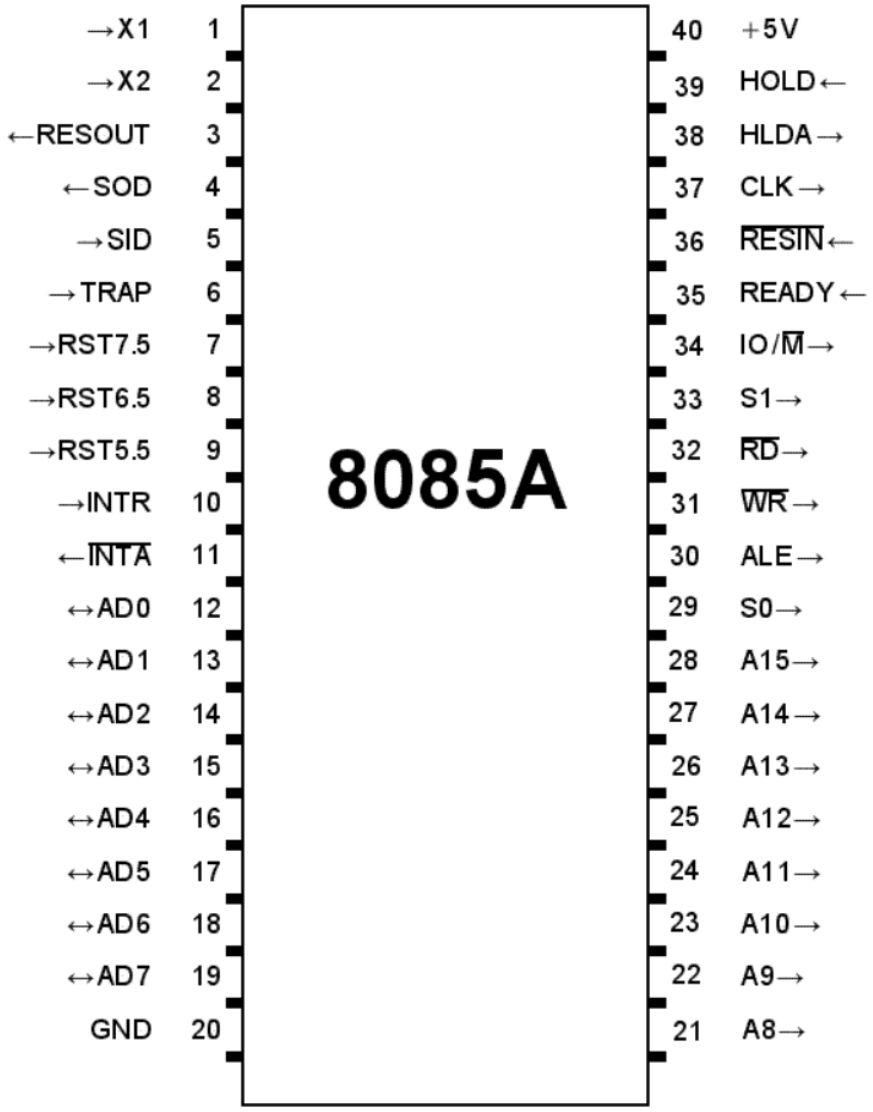


Figure 35: Pin diagram of Intel 8085[9]

1. *Address and Data Signals:* The address bus has 8 signal lines A8 – A15 which are unidirectional. The other 8 address bits are multiplexed (time shared) with the 8 data bits. The bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time. During the execution of the instruction, these lines carry the address bits during the

[9] Image adopted from: https://upload.wikimedia.org/wikipedia/commons/a/ab/Anschlussbelegung_8085.gif available under creative common license.

early part, then during the late parts of the execution, they carry the 8 data bits. In order to separate the address from the data, we use a latch to save the value before the function of the bits changes.

2. *Control and Status Signals*: There are 4 main control and status signals. These are:
   a. ALE(pin 30): Address Latch Enable. This signal is a pulse that become 1 when the AD0 – AD7 lines have an address on them. It becomes 0 after that. This signal can be used to enable a latch to save the address bits from the AD lines.
   b. RD(pin 32): Read (Active low).
      WR(pin 31): Write(Active low).
   c. IO/M(pin 34): This signal specifies whether the operation is a memory operation (IO/M=0) or an I/O operation (IO/M=1).
   d. S1(pin 33) and S0(pin 29) : Status signals to specify the kind of operation being performed. Usually not used in small systems.

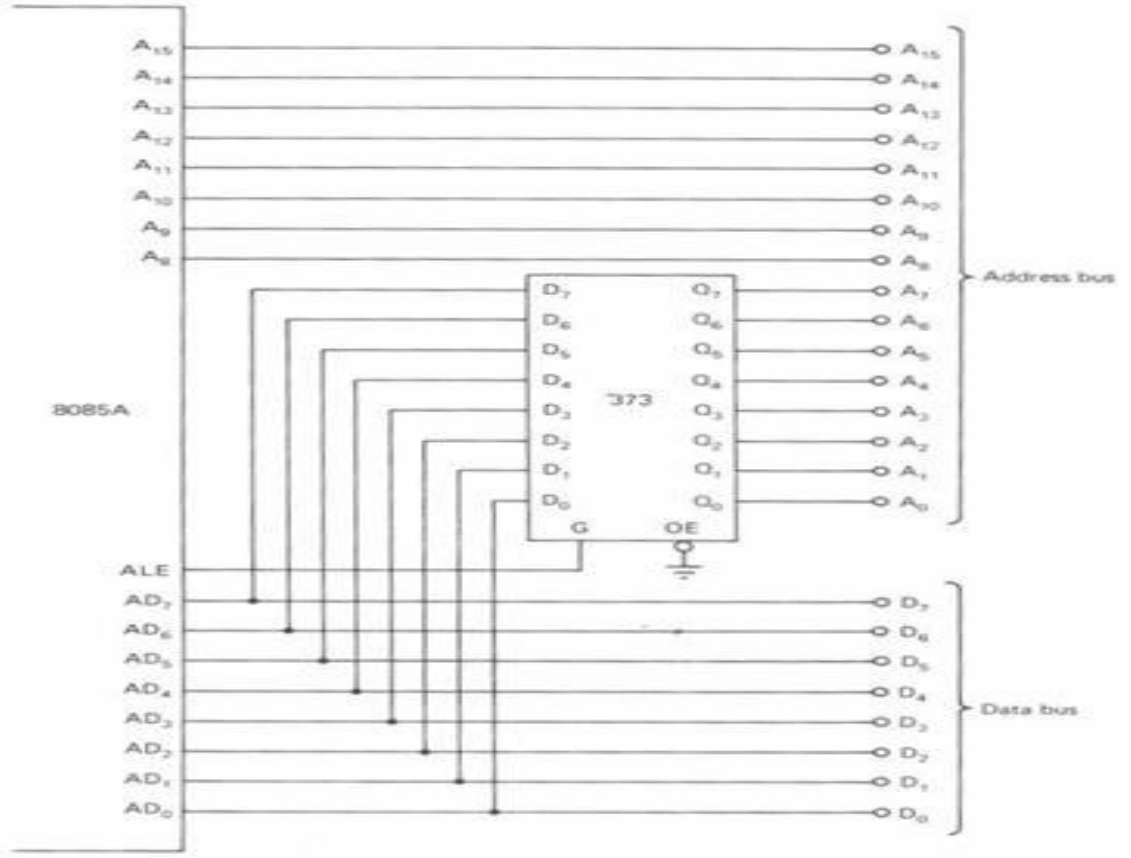


Figure 36: Multiplexed Data and Address Bus in Intel 8085

3. *Power Supply and Frequency:* There are 3 important pins in the frequency control group.
    a. X1(pin 1) and X2(pin 2) are the inputs from the crystal or clock generating circuit. The frequency is internally divided by 2. So, to run the microprocessor at 3 MHz, a clock running at 6 MHz should be connected to the X1 and X2 pins.
    b. CLK (OUT)(Pin 37): An output clock pin to drive the clock of the rest of the system.
4. *Interrupts and Externally initiated signals*:
    a. INTR(pin 10): Input Request. This signal is used as a general-purpose interrupt.
    b. INTA(Active Low)(pin 11): Interrupt Acknowledge. This signal is used to acknowledge an interrupt.
    c. RST 7.5(pin 7): RST 6.5(pin 8 RST 5.5(pin 9): } Restart Interrupts. These are vectored interrupts that transfer the control of the program to specified memory locations.
    d. TRAP(pin 6): This is a non-maskable interrupt and has a highest priority.
    e. HOLD(pin 39): Whenever a peripheral device, such as DMA want the hold of the address and the data bus, this signal is initiated by the peripheral device.
    f. HLDA(pin 38): Hold Acknowledge. This signal acknowledges the HOLD request.
    g. READY(pin 35): This signal is used to delay the microprocessor Read or Write cycles until a slow peripheral device is ready to send or receive data. When this signal is low, the microprocessor waits for a integral number of clock cycles until it goes high.

    There are two kinds of RESET signals in 8085:
    h. RESET IN(pin 36): an active low input signal, Program Counter (PC) will be set to 0 and thus MPU will reset.
    i. RESET OUT(pin 3): an output reset signal to indicate that the μp was reset (i.e. RESET IN=0). It also used to reset external devices.
5. *Serial I/O Ports:*
    a. SID (Input)(pin5): Serial input data line The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
    b. SOD (output)(pin 4): Serial output data line. The output SOD is set or reset as specified by the SIM instruction.

    There are two more pins left:
    a. Vcc(pin 40): +5 volt supply.
    b. Vss(pin 20): Ground Reference.

The signal groups of Intel 8085 microprocessor is shown in Figure 37.

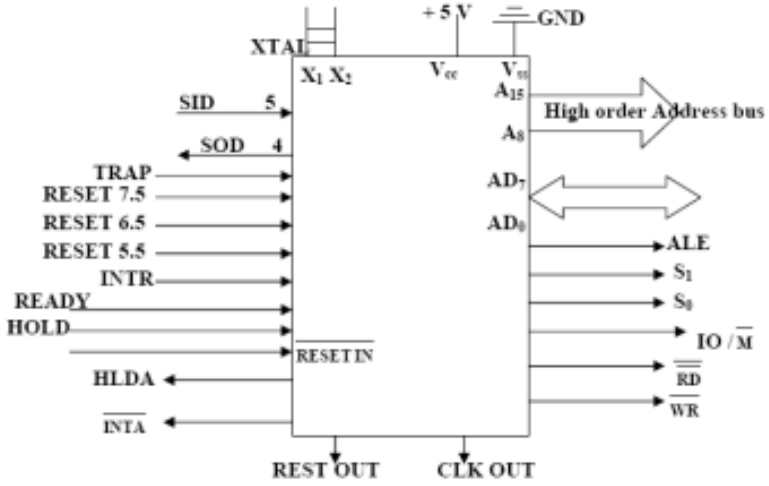


Figure 37: Signal Groups of Intel 8085

3.2 The Internal Architecture of 8085

Now let discuss the internal architecture of the 8085 microprocessor in detail. Fig. 4 explains the internal architecture of the 8085.

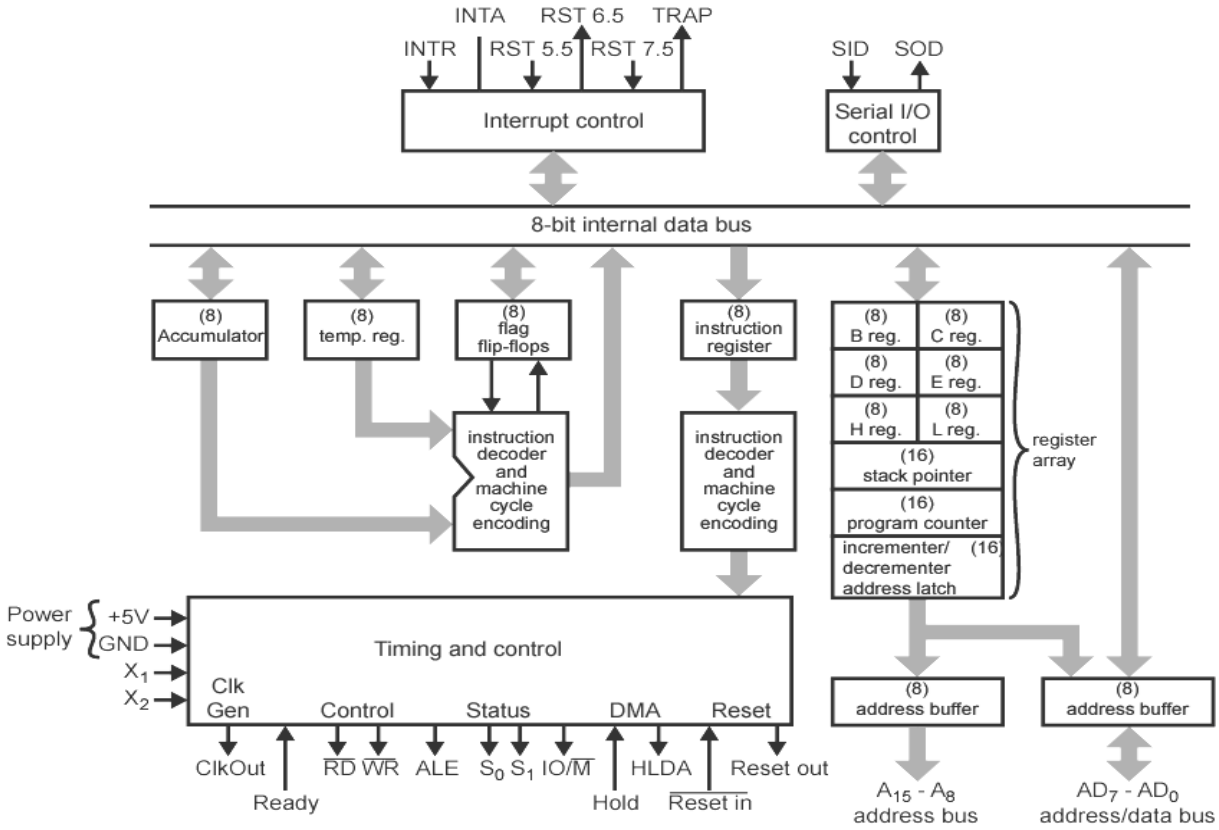


Figure 38: Internal Architecture of 8085 Microprocessor

The ALU consists of arithmetic and logic circuits to perform arithmetic and logic operations. It also consists of register set to perform these operations on the data. The details of these registers are discussed below.

The 8085 microprocessor contains seven 8-bit register which are directly accessible to the programmer. These registers are named as A, B, C, D, E, H, and L.

A	Status Register
B	C
D	E
H	L

Program Counter

Stack Pointer

Figure 39: Register Set of 8085 Microprocessor

A is the 8-bit accumulator where all the operation on data takes place. The other six registers can be used as 8- bit register or a 16-bit register pair to manipulate 16-bit data. The register pair is as follows: BC, DE, and HL. In case a operation on a 16-bit data is to be performed, the HL pair can be used as a 16-bit accumulator. Apart from that, it also consists of two 16-bit registers. PC, program counter, controls the sequencing of the execution of instructions and is used to store the address of the next instruction to be executed. And SP, stack pointer, is used to point the address of the top-most element of the stack. The register set is shown in Figure 39.

It register set also contains 8-bit status register. Each bit of this status register contains a flag, which is a –bit flip-flop. These status flags are affected by the arithmetic and logic operations before or after the operation. There are six status flags in the status register and these are S (sign flag), Z (zero flag), AC (auxiliary carry flag), P (parity flag) & CY (carry flag).
Figure 40 represents the status register.

S	**Z**		**AC**		**P**		**CY**

Figure 40: Status Flags

The use of these flags is explained below:

a. S(sign flag): The sign flag is set if bit D7 of the accumulator is set after an arithmetic or logic operation.
b. Z(zero flag): Set if the result of the ALU operation is 0. Otherwise is reset. This flag is affected by operations on the accumulator as well as other registers. (DCR B).
c. AC(Auxiliary Carry): This flag is set when a carry is generated from bit D3 and passed to D4 . This flag is used only internally for BCD operations.
d. P(Parity flag): After an ALU operation, if the result has an even # of 1s, the p-flag is set. Otherwise it is cleared. So, the flag can be used to indicate even parity.
e. CY(carry flag): This flag is set when a carry is generated from bit D7 after an unsigned operation.
f. OV(Overflow flag): This flag is set when an overflow occurs after a signed operation.

Whenever an Instruction from the memory is fetched, the instruction is placed inside a 8-bit register, known as Instruction Register(IR). A decoder is attached to the Instruction Register which enables the CPU to decode instruction and take appropriate action. Suppose, after decoding the instruction, it was found that it was an addition instruction, the CPU will generate necessary control signals to initialize the adder and fetch the operand either from the memory or the input device.

The 8085 microprocessor have 8-bit data bus and 16-bit address bus. The address bus has 8 signal lines A8 –A15 which are unidirectional. The other lower order 8 address bits are multiplexed (time shared) with the 8 data bits. So, the bits AD0 –AD7 are bi-directional and serve as A0 –A7and D0 –D7at the same time. During the execution of the instruction, these lines carry the address bits during the early part, and then during the late parts of them execution, they carry the 8 data bits. In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

Now let us discuss how the demultiplexing of AD7-AD0 is done to serve the dual purpose i.e. same line are used as address lines and data lines. The high order bits of the address remain on the bus for three clock periods. However, the low order bits remain for only one clock period and they would be lost if they are not saved externally. Therefore, an external latch is used to save the value of AD7–AD0 when the lines are carrying the address bits. Address Latch Enbale(ALE) signal is used to enable the latch. Whenever AD7- AD0 is to be used for data bus, the ALE goes low.

Direct Memory Access(DMA) technique is used when a fast speed I/O device want to transmit the data to memory at high speed and the speed of CPU limits the speed of transfer. In this case

the CPU is bypassed and the control of the data and address buses is given to the transmitting device. Once the transfer is complete, the control of Data and Address bus is relinquished to the CPU. To facilitate DMA transfer, HOLD and HLDA signals are used. Whenever an I/O device request for DMA transfer, it enables the HOLD line. As soon as the HOLD line is enabled, the microprocessor data and the address bus of the CPU are placed in the high impedance state and the control of the buses is transferred to I/O device. After this, the HLDA signal is initialized by the CPU which is a signal for the I/O device to start the transfer of data. Once the data transfer is complete, the HOLD signal is disabled and the control of buses is returned to CPU.

Interrupt is a mechanism by which an I/O or an instruction can suspend the normal execution of processor and get itself serviced. 8085 microprocessor have few maskable and non-maskable Interrupts. There are four hardware interrupts in 8085:

- TRAP
- RST 7.5
- RST6.5
- RST5.5

Interrupts are generally used to stop the normal execution sequence of the instructions by the CPU and address a higher priority task first. This usually happens when the peripheral device want to transmit data to either to memory or CPU. The device which seeks CPU attention sends an interrupt signal INTR to CPU. The CPU holds the operation which it was performing, save the intermediately data and the registers value, so that it could resume the task later. The CPU sends the interrupt acknowledgement INTA to the device, which is a signal that the CPU is now ready to service the request of the device which initiated the interrupt and the vectored address, the address of the Interrupt Service Routine to handle the interrupts is stored in the Program Counter(PC). After this the CPU executes the *Interrupt Service Routine(ISR)*, which is a small program/routine to service the corresponding interrupting source. The interrupts are of two categories, maskable and non-maskable.

a. *Maskable interrupts*: these are the class of interrupts which a CPU can ignore if it is servicing an important task. TRAP is the an example of a non-maskable interrupt.
b. *Non-maskable interrupts*: these are the class of interrupts which the CPU cannot afford to ignore and has to be serviced immediately, come what may. Typically this category of interrupts is used for critical condition. RST 7.5, RST 6.5 and RST 5.5 are the examples of maskable interrupt.

The priority order of the interrupts is as follows:

TRAP > RST 7.5 > RST 6.5> RST5.5> INTR

3.3 Summary

1. A group of lines that are used to send a memory address or a device address from the MPU to the memory location or a peripheral.
2. The 8085 microprocessor have 8-bit data bus and 16-bit address bus.
3. The data bus is bi-directional because the data flow in both directions between the MPU and memory and peripheral devices.
4. The accumulator is the register used to store the 8-bit data to perform the arithmetic and logical operations.
5. The data conditions, after arithmetic or logical operations, are indicated by setting or resetting the flip-flops called flags.
6. The number of bits stored in a register is called a memory word.
7. Direct Memory Access(DMA) technique is used when a fast speed I/O device want to transmit the data to memory at high speed and the speed of CPU limits the speed of transfer.
8. The instruction Enable Interrupt sets the Interrupt Enable flip-flop and enables the interrupt process.
9. Interrupts are generally used to stop the normal execution sequence of the instructions by the CPU and address a higher priority task first.
10. The priority order of the interrupts is as follows: TRAP > RST 7.5 > RST 6.5> RST5.5> INTR.

3.4 Check Your Progress

1. The 8085 microprocessor has _____ address lines.
2. _____ is single line that is generated by the MPU to provide timing of various operations.
3. _____ register pair can be used as data pointer or memory pointer.
4. 8085 can access _____I/O ports.

5. The _____ signal obtained from pin 37 of 8085 Microprocessor is used for synchronizing external devices.
6. In 8085, the signals can be divided into _____ groups according to their functions.
7. 8085 operates at a frequency of ____ Mhz.
8. The interrupts are of two categories: _____ and _____ .
9. DMA stands for _____.

3.5 Answers to Check Your Progress

1. 16
2. Control bus
3. HL
4. 256
5. CLK_{OUT}
6. Seven
7. 3
8. Maskable, non-maskable
9. Direct Memory Access

3.6 Model Questions

1. What are the general purpose registers in 8085?
2. What are the control signals used by the 8085?
3. What are the different type of registers used in 8085?
4. Which Register handles the arithmetic operations in the 8085?
5. What are the different flags included in the ALU?
6. What is the purpose of the sign flag in 8085?
7. How many instruction bits does 8085 processor support?
8. What is the maximum clock frequency used by the 8085 microprocessor?
9. What was is the primary usage of a 8085 microprocessor?
10. Define the architecture of the 8085 microprocessor with the help of a Pin diagram?
11. What is the significance of the control unit in the 8085?

12. Name the unit that controls the sequential execution of instructions?
13. State the total number of pins in the 8085 microprocessor?
14. What is the significance of the HOLD and HLDA pins?
15. The Input/Output signals are related to which pins?
16. In 8085, power and frequency can be checked by connecting the wire with which pins?
17. What are the total number of input and output ports in 8085?
18. Define the types of bus used in the 8085?
19. How many bit address bus does the 8085 use?
20. Why is the address bus in the 8085 tri-stated and unidirectional?
21. What is the total addressable memory size in a 8085 microprocessor?
22. What are the steps involved in communication of 8085 microprocessor with the memory?

CHAPTER IV: OPERATION AND CONTROL OF-8085 MICROPROCESSOR

4.0 Learning Objectives

After going through this unit, you will be able to:

- Understand the working for timing and control unit of 8085 Microprocessor
- Define machine cycle and instruction cycle
- Draw and Understand the timing diagram of various instructions

4.1 Timing and Control Unit

The timing and control unit of the 8085 is responsible for the generation of timing and control signals so that instructions can be executed. These operations synchronize the process of communication between Microprocessor(μP) and peripheral devices by generating the control signals necessary for communication for e.g. $\overline{RD}$ and $\overline{WR}$ signals which indicate the availability of data on the data bus. Whenever Microprocessor has to execute any instruction, the instruction is first received by the instruction register (8-bit) through A/D (address/data) bus. The instruction is then passed on to the instruction decoder where it is split, decoded and then passed to the control unit to generate necessary control signals for its execution. Besides doing this timing and control unit also checks whether any internal or external interrupt has occurred. If there is any kind of interrupt request this unit stops the execution of normal sequence of instructions to respond to it (interrupt) by generating the required control signals.

To execute any instruction Microprocessor has to pursue the following steps:

1. Identify the memory location from where the instruction is to be fetched
2. Decode the instruction using instruction decoder
3. Perform the function specified by the decoded instruction.

All these operations are performed within a given time interval, which is provided by the clock of the system. Before discussing the concept further, let us define some frequently encountered terms throughout the unit:

- **Timing Diagram:** It is a graphical representation and it represents the execution time taken by each instruction in a graphical format. The execution time is represented in T-states.
- **Instruction Cycle:** The time required to execute an instruction is called an Instruction cycle.
- **Machine cycle:** The time required to access the memory or input/output devices is called machine cycle.
- **T-States:** The machine cycle and instruction cycle takes multiple clock periods. A portion of an operation carried out in one system clock period is called as T-state.

The time required by the 8085 to fetch and execute one machine language instruction is defined as an Instruction Cycle[10]. Each instruction is divided into one to five Machine Cycles. Each machine cycle is essentially the result of the need, by the instruction being executed, to access the RAM. The shortest instruction would require just one machine cycle, in which the instruction itself is obtained from RAM. The longest, of five machine cycles, would consist of five RAM accesses, the first to obtain the instruction byte itself, and the remaining four to be divided into fetching and saving other bytes. For example, cycle numbers 2 & 3 may be needed to fetch two more bytes of an address, while numbers 4 & 5 may be needed to save a 2-byte address somewhere else in RAM.

4.2 Machine Cycle

The 8085 divides the clock frequency at x1 and x2 inputs by 2 which is called operating frequency. T-state can be represented as:

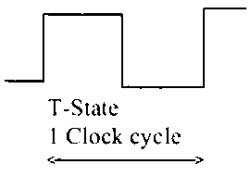


Figure 41: T-State

In the above figure the rising and the falling edge of the pulse is shown as a straight line which represent that it takes non-zero time to change the state from 0 to1. But practically it is a wrong representation. It takes a non-zero time, however small it may be, to change the state from 0 to 1 and vice versa. The actual representation is shown in the diagram below:

[10] http://microprocessor-8085.blogspot.in/2009/01/processor-cycle-of-8085.html

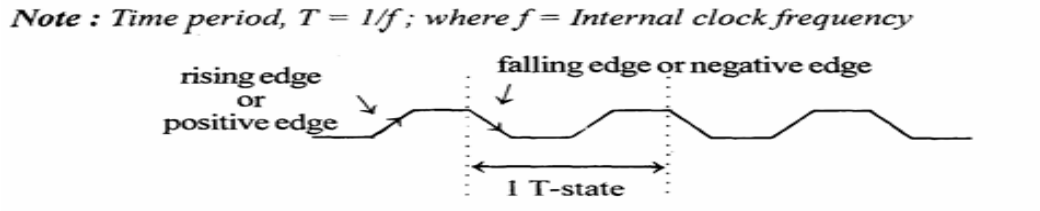


Figure 42: Rising and falling edge of a clock pulse

The processor takes a definite time to execute the machine cycles. The time taken by the processor to execute a machine cycle is expressed in T-states. One T-state is equal to the time period of the internal clock signal of the processor. The T-state starts at the falling edge of a clock. The type of machine cycle being executed is specified by the status lines IO/$\overline{M}$, S_0, and S_1, and the control lines $\overline{RD}$, $\overline{WR}$, and $\overline{INTA}$. These six lines can define seven different machine cycle types as follows. Refer to the table below which defines the bit patterns of each.

Table 4: Machine cycles

IO/$\overline{M}$	S_1	S_0	Data Bus Status
0	0	0	Halt
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode Fetch
1	1	1	Interrupt Acknowledge

The 8085 microprocessor has 7 basic machine cycles. They are:

1. Opcode fetch cycle (4T)
2. Memory read cycle (3T)
3. Memory write cycle (3T)
4. I/O read cycle (3T)
5. I/O write cycle (3T)
6. Interrupt acknowledge cycle (6T or 12T)

7. Bus idle cycle (2T or 3T)

4.2.1 OP CODE FETCH

This is the first machine cycle of any instruction. It is defined with S_0 and S_1 asserted high, and IO/$\overline{M}$ and $\overline{RD}$ low. It is a read cycle from RAM to obtain an instruction byte. Each instruction of the processor has one byte opcode. The opcodes are stored in memory. So, the processor executes the opcode fetch machine cycle to fetch the opcode from memory. Hence, every instruction starts with opcode fetch machine cycle. The time taken by the processor to execute the opcode fetch cycle is 4T. In this time, the first, 3 T-states are used for fetching the opcode from memory and the remaining T-states are used for internal operations by the processor.

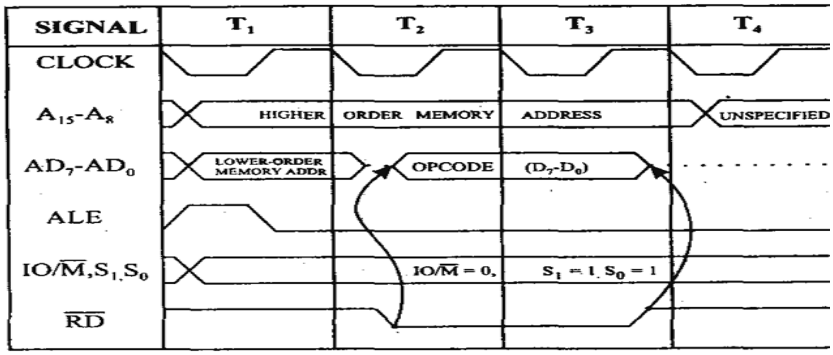


Figure 43: Opcode Fetch cycle

4.2.2 MEMORY READ

This is a normal read cycle of any byte except the OP code. It is defined with S0 and S1 set to 0, 1 respectively, and I-O/M and /RD low. It is a read cycle from RAM to obtain a data or address byte.

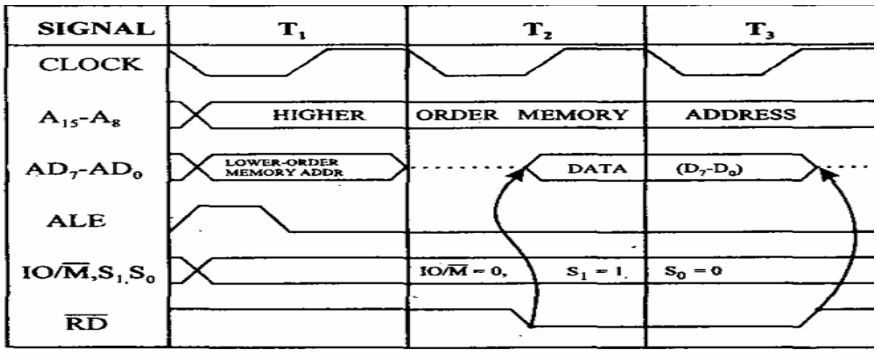


Figure 44: Machine read cycle

The memory read machine cycle is executed by the processor to read a data byte from memory. The processor takes 3T states to execute this cycle. The instructions which have more than one byte word size will use the machine cycle after the opcode fetch machine cycle.

4.2.3 MEMORY WRITE

This is a normal write cycle to memory. It is defined with S0 and S1 set to 1, 0 respectively, and I-O/M and /WR low. It is a write cycle to RAM to store one byte in the specified address.

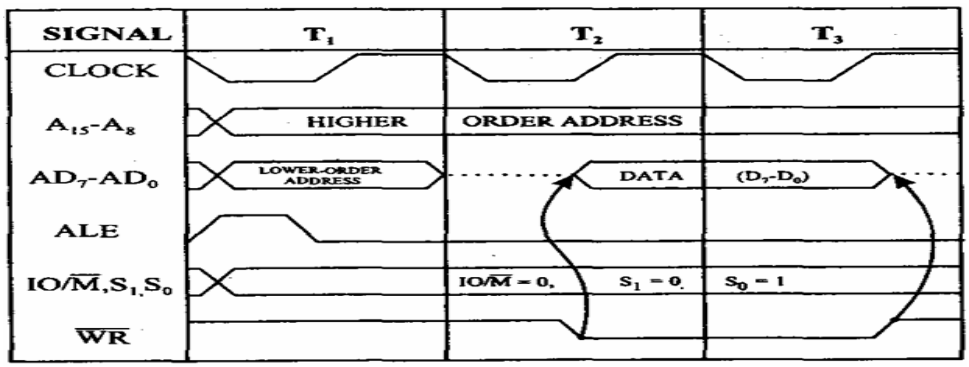


Figure 45: Machine write cycle

The memory write machine cycle is executed by the processor to write a data byte in a memory location. The processor takes, 3T states to execute this machine cycle

4.2.4 I/O READ

This is a normal read cycle from an I/O device. It is defined with S0 and S1 set to 0, 1 respectively, and with I-O/M high and /RD low. It is a read cycle which will bring one byte into the MP from the input device specified.

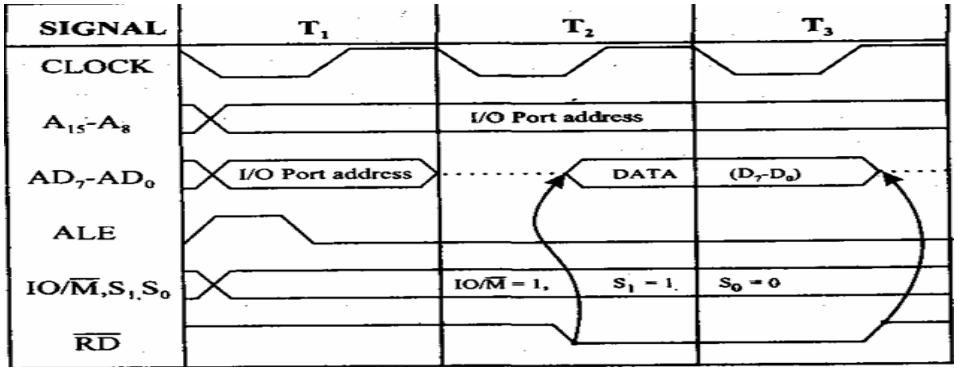


Figure 46: I/O read cycle

The I/O write machine cycle is executed by the processor to write a data byte in the I/O port or to a peripheral, which is I/O, mapped in the system. The processor takes, 3T states to execute this machine cycle.

4.2.5 I/O WRITE

This is a normal write cycle to an I/O device. It is defined with S0 and S1 set to 1, 0 respectively, and with I-O/M high and /WR low. It is a write cycle which will send one byte outbound from the MP to the specified output device.

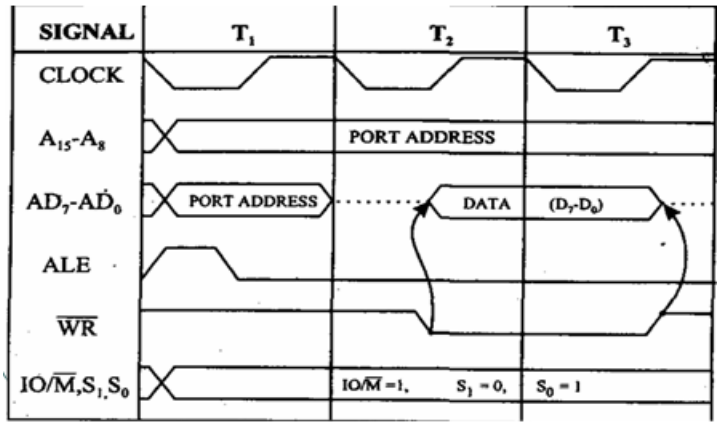


Figure 47: I/O write cycle

The I/O write machine cycle is executed by the processor to write a data byte in the I/O port or to a peripheral, which is I/O, mapped in the system. The processor takes, 3T states to execute this machine cycle.

4.2.6 Interrupt Acknowledge Cycle

This is a response to an interrupt request applied to the MP via the INTR line. It is defined with S0 and S1 set to 1, 1 respectively, I-O/M set high, and both /RD and /WR also high. The Interrupt Acknowledge pin is also held to a low asserted level. It is neither a read nor write cycle, although the interrupting device will jam an interrupt vector onto the D0-D7 lines on the next machine cycle.

The 8085 Microprocessor check for the Interrupt signal at the end of the second T-state of the of the last machine cycle of every instruction. If there is a valid Interrupt request and if INTR is enabled then the processor completes the current instruction execution and then executes and

Interrupt Acknowledgement Machine Cycle. If there is a valid Interrupt request and if INTA is enabled then the processor completes the current instruction execution and then executes then executes an Interrupt Acknowledgement machine cycle. The Interrupt Acknowledgement machine cycle is executed to get either a RST n instruction from the interrupting device or to get a CALL instruction with CALL address from the interrupting device. It also stores content of Program Counter in stack so that the Microprocessor can return back to the current program it is executing after servicing the interrupt.

4.2.6.1 Interrupt Acknowledgement Machine Cycle with RST n instruction

In the first T-state of the Interrupt acknowledgement cycle, the address is placed on the AD_7-AD_0 and A_8-A_{15} address lines and ALE is asserted HIGH. The other control signals are asserted as follows:

$IO/\overline{M}$ =1; S_0=1 and S_1=1.

In the middle of T_1, ALE is asserted LOW. The INTR signal will go to LOW state once the Interrupt is accepted.

In the second T-Cycle(T_2), the $\overline{INTA}$ is asserted LOW and this enables the interrupting device to place the opcode of RST n instruction on the data bus.

At the end of T_3, $\overline{INTA}$ is asserted HIGH and the RST n opcode is latched into the Microprocessor.

The next three T states T_4, T_5 and T_6 are used for internal operations. The internal operating performed are decoding of instruction and encoding into various machine cycles and generation of vector address of RST n interrupt.

The T-states, T_7,T_8 and T_9 are used to store the high byte of the program Counter in stack.

In T_7, the content of Stack Pointer(SP) is decremented by 1, and placed on AD_0-AD_7 and A_8-A_{15} lines. ALE is asserted HIGH and then LOW, to latch the low byte of address into external latch. The status signals are asserted as $IO/\overline{M}$ =0, S_0=1 and S_1=0.

In T_8, the high byte of PC is placed in AD_0-AD_7 and $\overline{WR}$ is asserted LOW to enable the stack memory for write operation. At the end of T_9, $\overline{WR}$ is asserted HIGH.

T-state T_{10},T_{11} and T_{12} are used to store the low byte of the program counter into stack.

In T_{10}, the content of SP is again decremented by one and placed on AD_0-AD_7 and A_8-A_{15} lines. ALE is asserted HIGH and then LOW, to latch the low byte of address into external latch. The status signal are IO/$\overline{M}$ =0, S_0=1 and S_1=0.

In T_{11}, the low byte of PC is placed on AD_0-AD_7 lines and $\overline{WR}$ is asserted LOW to enable the stack memory for write operation. At the end of T_{12} $\overline{WR}$ is asserted HIGH.

After the Interrupt Acknowledgement Machine Cycle, the Program Counter will have the vector address of RST n instruction and so the Microprocessor start servicing the interrupt by executing the interrupt service subroutine stored at this address.

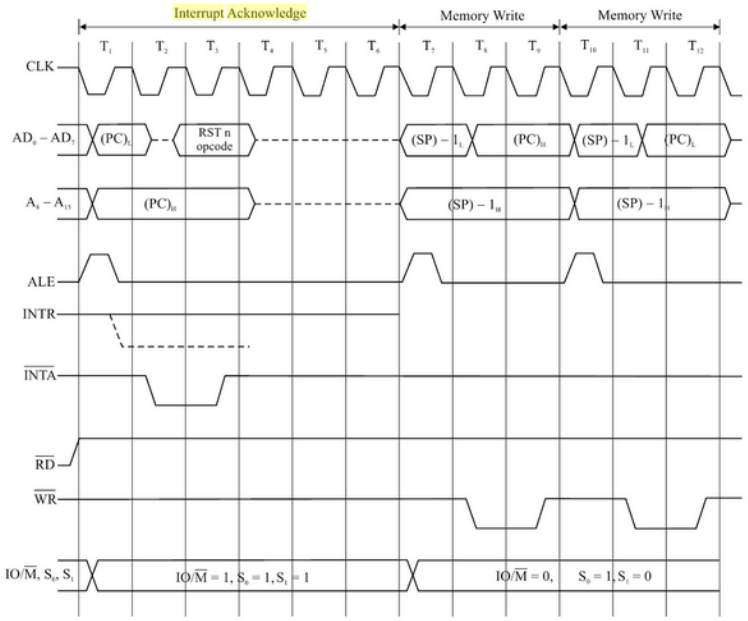


Figure 48: Interrupt Acknowledgement Machine Cycle with RST n instruction

4.2.6.2 Interrupt Acknowledgement Machine Cycle with CALL instruction

This cycle is executed by machine to service the interrupt, when an interrupt request is made through 8259(Interrupt Controller) to the INTR pin of the 8085 Microprocessor. The Intel 8259

can accept 8 interrupt requests and allow one by one to the INTR pin of the 8085 Microprocessor. It also supplies CALL opcode and CALL address, when it receives $\overline{INTA}$ signal form the Microprocessor.

The Microprocessor checks for an interrupt at the second T-state of the last machine cycle of every instruction. If there is a valid interrupt request and if INTR is enabled then the processor completes the current instruction execution and then executes an interrupt acknowledge machine cycle. The timings of various signals during interrupt acknowledgement cycle when CALL instruction is supplied by the interrupting device is shown in the figure below.

1. At the falling edge of T_1 the address is place in the address lines and ALE is asserted HIGH. But the address is not used to read from memory. The other control signals are asserted as IO/$\overline{M}$ =1, S_0=1 and S_1=1. In the middle of T_1, ALE is asserted LOW. The INTR signal can remain HIGH or it can go LOW once the interrupt is accepted by executing acknowledgement cycle.
2. In T_2, $\overline{INTA}$ asserted LOW and this enables the interrupt controller 8259 to place a CALL opcode on the data bus.
3. At the end of T_2, the $\overline{INTA}$ is asserted HIGH and the CALL opcode is latched into the processor.
4. The T-states T_4,T_5 and T_6 are used for internal operations. The internal operations performed are decoding the opcode and encoding into various machine cycles.
5. The T states T_7,T_8 and T_9 are used to fetch the LOW byte of call address from 8059. In T_7, the content of Program Counter is placed on address bus but nit used for memory operations. In T_8, the $\overline{INTA}$ is asserted LOW and this enables the interrupt controller 8259 to place the low byte of call address on data bus. At the end of T_9 the $\overline{INTA}$ is asserted HIGH and the high byte call address on the data bus is latched into the Microprocessor.
6. T_{10} of Program Counter is placed on address bus, but not used for memory operation. In T_{11} the INTA is asserted LOW and 8259 is enabled for placing the high byte of CALL address on data bus. At the end of T_{12} the $\overline{INTA}$ is asserted HIGH and the high byte call address on the data bus is latched into the Microprocessor.
7. The T states T_{13},T_{14} and T_{15} are used to store the high byte of the program counter in stack memory.

8. The T states T_{16}, T_{17} and T_{18} are used to store the low byte of the Program Counter in Stack memory. In T_{16}, the content of Stack Pointer is again decremented by one and placed in address bus. ALE is asserted HIGH and then LOW, to latch the low byte of address into external latch. The other control signal are asserted as IO/$\overline{M}$ =0, S_0=1 and S_1=0. In T_{17}, the low byte of Program Counter is placed on the lower order address lines and $\overline{WR}$ is asserted LOW to enable the Stack Memory for write operation. At the end of T_{18} $\overline{WR}$ is asserted HIGH.

After the interrupt acknowledge machine cycle, the Program counter will have the CALL address and so the processor start servicing the interrupt by executing the interrupt service subroutine stored at the address.

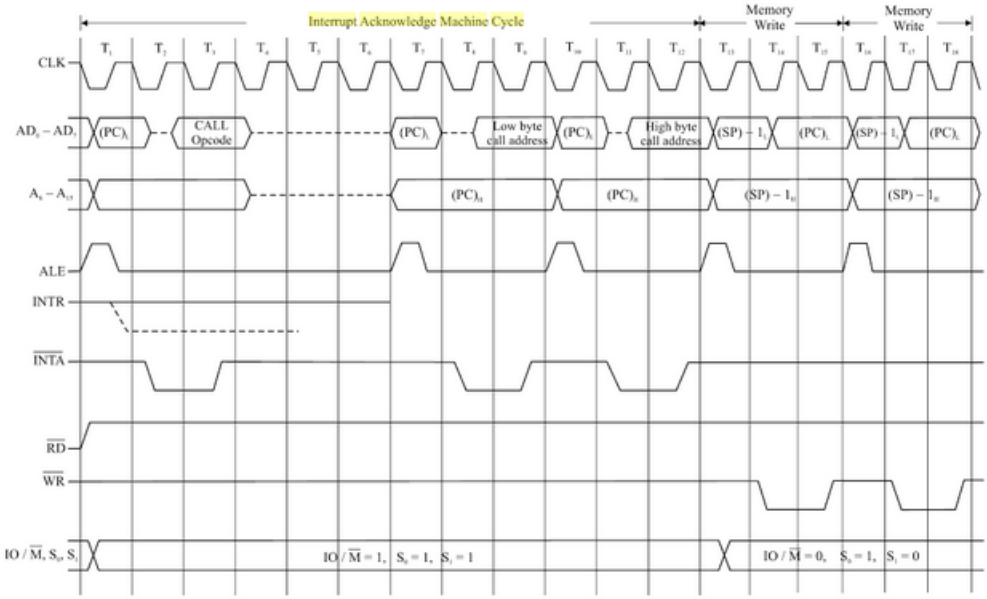


Figure 49: Interrupt Acknowledgement Machine Cycle with CALL instruction

4.2.7 Bus Idle Machine Cycle

The bus idle machine cycle is executed, when extra time or more time is needed for and internal operation of the processor. During this cycle, the status signal S0 and S1 are asserted LOW. The data, address and control pins are driven to high impedance state. The ready signal will not be sampled by the processor during this cycle.

This is an idle cycle in which no specific bus activity is defined. It occurs under following conditions:

4.2.7.1 Double Add Instruction(DAD)

DAD instruction is used to ADD the content of the HL register pair with the content of register specified in the instruction. This instruction requires enough execution time to merit its own Idle cycle. It is defined with S_0 and S_1 set to 0, 1 respectively, I-O/$\overline{M}$ set low, and neither $\overline{R}\overline{D}$ nor $\overline{W}\overline{R}$ asserted (both high). Since neither a read nor a write are specified, no bus action takes place. This instruction requires 10 T-states.

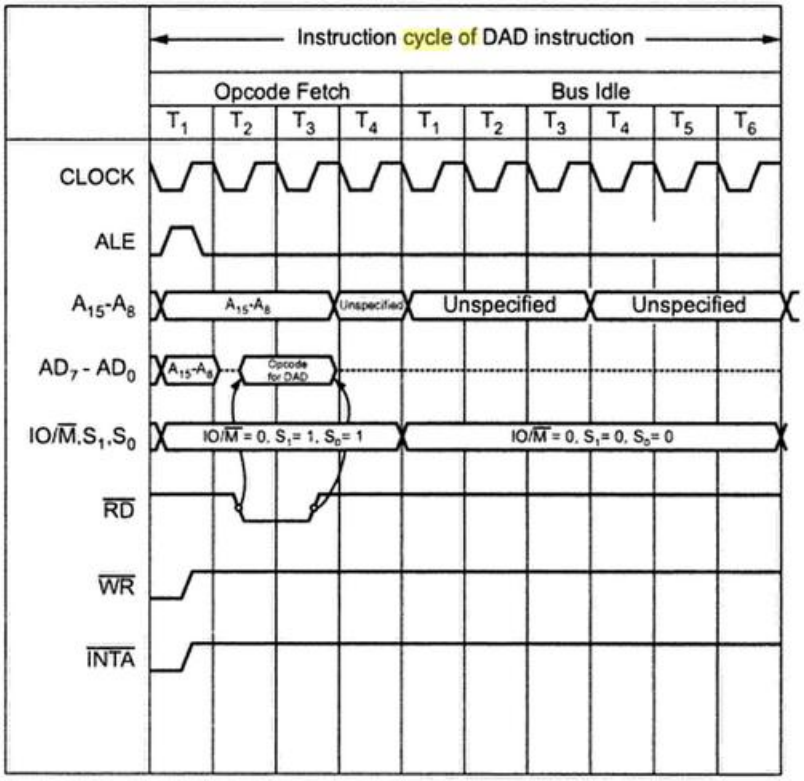


Figure 50: Timing diagram for DAD instruction

4.2.7.2 Acknowledge of Restart or Trap

This idle cycle allows time for the 8085 Microprocessor to cope with a RST or Trap interrupt request. The figure below shows the Bus Idle machine Cycle for TRAP. In response to the TRAP interrupt, 8085 enters into a Bus Idle machine cycle during which it invokes restart instruction,

stores the content of Program Counter into the stack and places the vector address oof TRAP i.e. 0024H into the Program Counter.

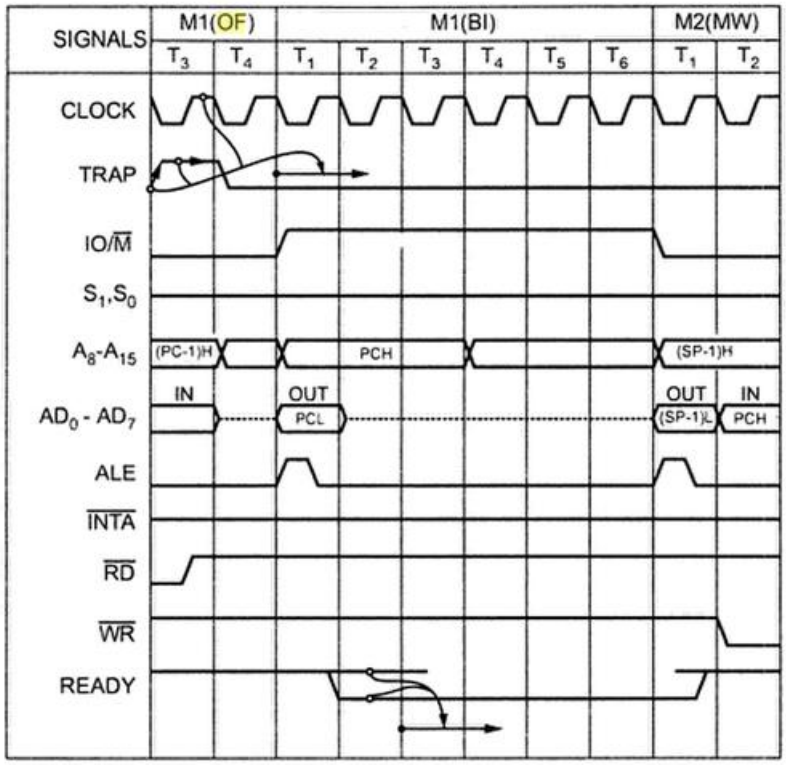


Figure 51: Bus Idle machine cycle for TRAP

The number of machine cycles required to fetch complete instruction depends on the instruction type. For one byte instruction it does not requires additional machine cycle. For two byte instruction it required one additional machine cycle for memory read operation. And for three byte instruction it requires two additional machine cycles for memory read operation.

4.3 Examples

Now let us discuss the timing diagram of some of the 8085 instruction. Let us start with STA 526AH

4.3.1 STA(Store Accumulator)

This instruction stores the contents of the accumulator specified address, which is 526A in this case. The opcode of the STA instruction is said to be 32H. It is fetched from the memory

41FFH(see fig). - OF machine cycle. Then the lower order memory address is read(6A). - Memory Read Machine Cycle. Read the higher order memory address (52).- Memory Read Machine Cycle. The combination of both the addresses are considered and the content from accumulator is written in 526A. - Memory Write Machine Cycle. Assume the memory address for the instruction and let the content of accumulator is C7H. So, C7H from accumulator is now stored in 526A.

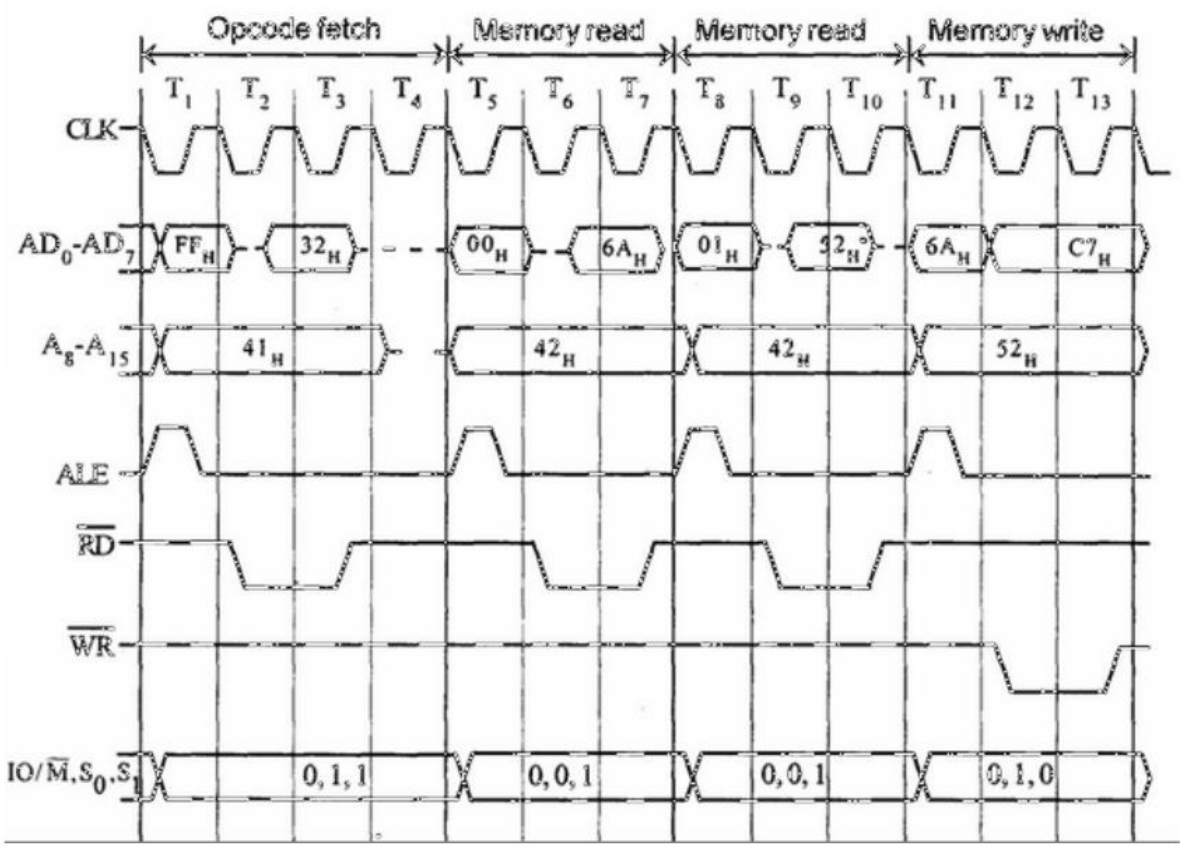


Figure 52: Timing diagram of STA 526AH

4..3.2 IN Instruction

Now let us discuss the timing diagram of another instruction, IN C0H. The steps are explained below:

- Fetching the Opcode DBH from the memory 4125H.

- Read the port address C0H from 4126H.
- Read the content of port C0H and send it to the accumulator.
- Let the content of port is 5EH.

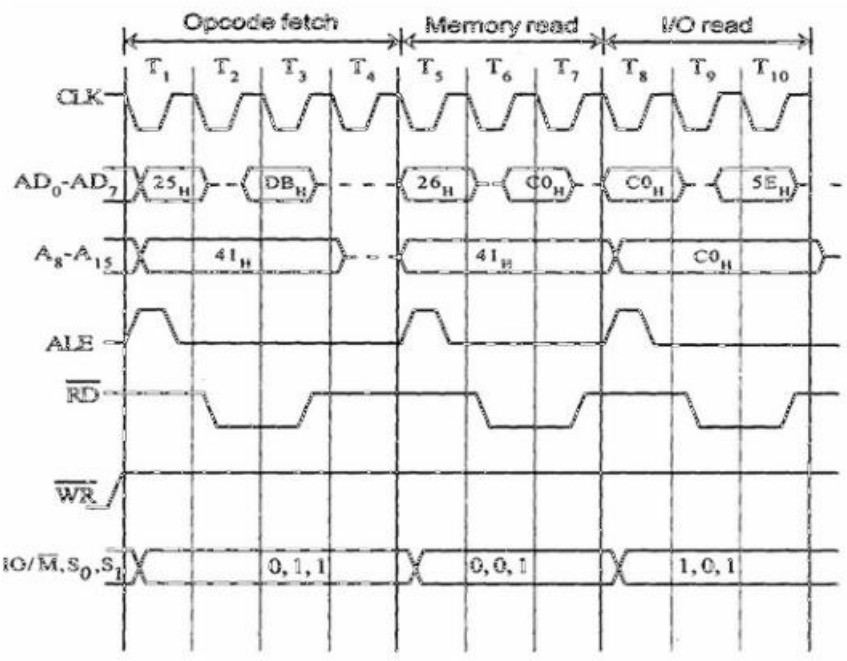


Figure 53: Timing diagram of IN instruction

4.4 SUMMARY

1. Timing diagram is the display of initiation of read/write and transfer of data operations under the control of 3-status signals IO/$\overline{M}$, S_1, and S_0.
2. The execution of instruction always requires read and writes operations to transfer data to or from the µP and memory or I/O devices.
3. The function of the microprocessor is divided into fetch and execute cycle of any instruction of a program.
4. The program is nothing but number of instructions stored in the memory in sequence. In the normal process of operation, the microprocessor fetches (receives or reads) and executes one instruction at a time in the sequence until it executes the halt (HLT) instruction.
5. An instruction cycle is defined as the time required to fetch and execute an instruction.

6. An instruction cycle consists of many machine cycles. Each machine cycle consists of many clock periods or cycles, called T-states.
7. A transparent latch is a flip-flop; its output changes according to input when the clock signal is high, and it latches the input on the trailing edge of the clock. The latch is necessary for output devices to return the result; otherwise the result will disappear.

4.5 Check Your Progress

1. Which of the following statement is true?
    a. The group of machine cycle is called a state.
    b. A machine cycle consists of one or more instruction cycle.
    c. An instruction cycle is made up of machine cycles and a machine cycle is made up of number of states.
    d. None of the above
2. If Microprocessor is interrupted, It
    a. Stops execution of instructions
    b. Acknowledges interrupt and branches to service subroutine
    c. Acknowledge interrupt and continue
    d. Acknowledge interrupt and waits for the next instruction from the interrupting device.
3. It is a graphical representation and it represents the execution time taken by each instruction in a graphical format.
4. The time required to execute an instruction is called an _____.

4.6 Answers to Check Your Progress

1. B
2. B
3. Timing Diagram
4. Instruction Cycle

4.6 Model Questions

1. Explain the role of timing and control unit in the operation of Microprocessor.
2. Define instruction cycle, machine cycle and timing diagram.
3. What is the use of ALE signal?
4. Define T-state.
5. Define instruction cycle.
6. Write down the control and status signals?
7. What is the function of SID and SOD pins in 8085 Microprocessor
8. Indicate the different machine cycles of 8085 Microprocessor.
9. Draw and explain the timing diagram for the memory read instruction.
10. Draw and explain the timing diagram for the I/O write instruction.
11. Draw the interrupt acknowledge machine cycle for
    a. RST instruction
    b. CALL instruction
12. What is meant by Bus Idle machine cycle?
13. Explain the DAD instruction and draw its timing diagram.
14. What are WAIT states in microprocessors?
15. What are the different steps involved in a Fetch Cycle?

INSTRUCTION SET OF 8085 MICROPROCESSOR

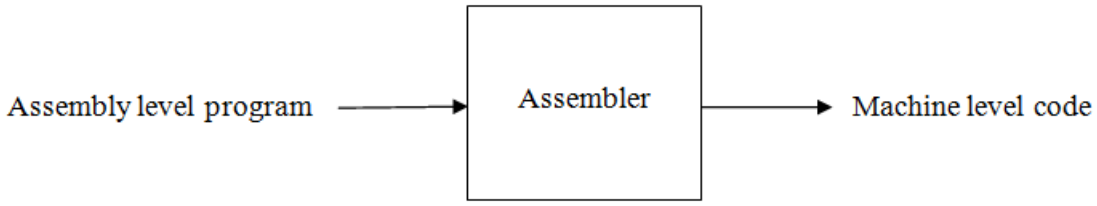
5.0 Learning Objectives

After reading this chapter, you will be able to:

- Define instruction
- Know the different instruction formats
- Define different 8085 Addressing modes
- Classify 8085 instruction set
- Understand various instructions of 8085 microprocessor

5.1 Introduction

Computer can perform all the operations for which it is designed i.e. perform operations that are defined in its instruction set. An instruction is a binary pattern designed inside a microprocessor to perform a specific function. To perform a specific task, a program, which is a sequence of instructions, is written and through this sequence of instructions we instruct the computer to perform what operation is to be performed, on what data and in which sequence. The programmer can write a program in assembly language using these instructions. The microprocessor understands machine language only which is in the form of strings of 0's and 1's. So the instructions written in assembly language are converted to machine language using an assembler. An assembler is a computer program which translates assembly language to machine language format.



5.2 8085 Instruction Format

An instruction is a command given to the microprocessor which specifies what task is to be performed on specified data. The instruction consists of two parts viz. operation code and operand.

Op-code	Operand

Op-code specifies the task to be performed like addition, subtraction, multiplication, etc. and the operand specifies the data to be operated upon. The Operand can be used in many different ways e.g. 8 bit data or 16 bit data or internal register or memory location or 8 bit or 16 bit address. 8085 instructions are classified into following three groups of instructions:

1. One-word or 1-byte instructions: These instructions include the op-code and the operand in the same byte. For example:

Operation	Opcode	Operand	Binary Code	Hex code
Compliment each bit of the accmulator register.	CMA	-	0010 1111	2FH
Add the content of regsister B to the content of Accumulator.	ADD	B	1000 0000	80H

2. Two-word or 2-byte instructions: The first byte specifies the op-code and the second byte specifies the operand. For example:

Operation	Opcode	Operand	Binary Code	Hex code
Load an 8-bit data byte in the accumulator register.	MVI	A, Data	0011 1110 Data	3E Data

3. Three-word or 3-byte instructions: The first specifies the op-code and the following 2 bytes specify the 16- bit address.

Operation	Opcode	Operand	Binary Code	Hex code
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011 1100 0011 0010 0000	C3 85 20

5.3 Addressing Modes in 8085

The various ways of specifying data (or operands) for instructions are called as **addressing modes.** The 8085 addressing modes are classified into following types:

1. Immediate addressing mode
2. Direct addressing mode
3. Register addressing mode
4. Register indirect addressing mode
5. Implicit addressing mode

5.3.1 Immediate Addressing mode

In immediate addressing mode operand is a part of the instruction itself. If the immediate data is 8-bit, the instruction will be of two bytes. If the immediate data is 16 bit, the instruction is of 3 bytes. For example:

1. ADI DATA; Add immediate the data to the contents of the accumulator.
2. LXIH 8500H; Load immediate the H-L pair with the operand 8500H.
3. MVI 08H; Move the data 08 H immediately to the accumulator.
4. SUI 05H; Subtract immediately the data 05H from the accumulator.

5.3.2 Direct Addressing mode

The mode of addressing in which the 16-bit address of the operand is directly available in the instruction itself is called Direct Addressing mode. i.e., the address of the operand is available in the instruction itself. This is a 3-byte instruction. For example:

1. LDA 9525H; Load the contents of memory location into Accumulator.
2. STA 8000H; Store the contents of the Accumulator in the location 8000H.
3. IN 01H; Read the data from port whose address is 01H.

5.3.3 Register Addressing Mode

In this mode the operands are microprocessor registers only i.e. the operation is performed within various registers of the microprocessor. For example:

1. MOV A, B; Move the contents of B register to A register.
2. SUB D; Subtract the contents of D register from Accumulator.
3. ADD B, C; Add the contents of C register to the contents of B register.

5.3.4 Register indirect addressing mode

The 16-bit address location of the operand stored in a register pair (H-L) is given in the instruction. The address of the operand is given in an indirect way with the help of a register pair. So it is called Register indirect addressing mode. For example:

1. LXIH 9570H : Load immediate the H-L pair with the address of the location 9570H.
2. MOV A, M : Move the contents of the memory location pointed by the H-L pair to accumulator.

5.3.5 Implicit Addressing mode

The mode of instruction which do not specify the operand in the instruction but it is implicated, is known as implicit addressing mode. i.e., the operand is supposed to be present generally in accumulator. For example:

1. CMA; complement the contents of Accumulator.
2. CMC; Complement carry.
3. RLC; Rotate Accumulator left by one bit.
4. RRC; Rotate Accumulator right by one bit..
5. STC; Set carry.

5.4 Instruction Set Classification

The 8085 microprocessor have 246 instructions. Each instruction is represented by an 8-bit binary value. These instructions have been classified into the following groups:

- Data Transfer Group
- Arithmetic Group
- Logical Group
- Branch Control Group
- I/O and Machine Control Group

5.4.1 Data Transfer Group

This category of instructions are used to transfer the content of source register to another destination register and after the transfer of the data, the content of the source remains unaltered. It is similar to copy command, where the selected contents from the source are transferred to destination without the affecting of content at the source. The example of this category instructions are MOV, MVI, LDA, LXI, STA, etc.

Example #1: MOV R1, R2

This instruction moves/copies the content of the source register R2 to Destination register R1.

MOV [Destination Register], [Source Register]

Suppose before the execution of the above instruction, the content of register R1 and R2 are 20 and 30 respectively.

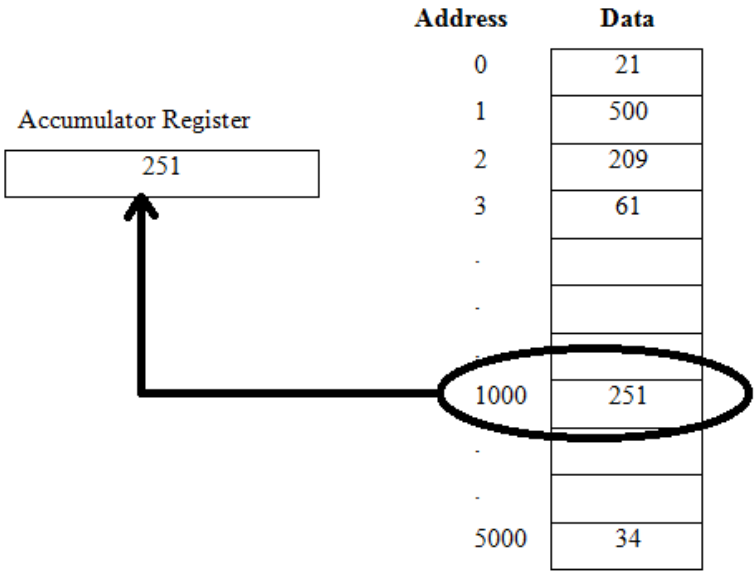
Destination Register R1	**Source Register R2**
20	30

After the execution of the statement MOV R1, R2, the content of R1= 30 and R2=30 i.e. the content of the source register remains same even after the execution of the instruction. Only the content of destination register is altered.

Destination Register R1	**Source Register R2**
30	30

Example #2: LDA 1000

This instruction load the content of the memory location, as specified in the instruction to the accumulator register and the after the execution of the instruction, the content of the memory location remains intact.



LDA [Address of Memory Location]

In the above example, after the LDA 1000 instruction is executed, the content of location 1000 is transferred to Accumulator register and after the transfer; the content at memory location remains unchanged.

The list of instructions which falls in this category is:

- MOV r1, r2 (Move Data; Move the content of the one register to another).
  [r1] ← [r2]
- MOV r, m (Move the content of memory register).
  r ← [M]
- MOV M, r. (Move the content of register to memory).
  M ← [r]
- MVI r, data. (Move immediate data to register).
  [r] ← data.

- MVI M, data. (Move immediate data to memory).
  M ← data.
- LXI rp, data 16. (Load register pair immediate).
  [rp] ← data 16 bits, [rh] ← 8 LSBs of data.
- LDA addr. (Load Accumulator direct).
  [A] ← [addr].
- STA addr. (Store accumulator direct).
  [addr] ← [A].
- LHLD addr. (Load H-L pair direct).
  [L] ← [addr], [H] ← [addr+1].
- SHLD addr. (Store H-L pair direct)
  [addr] ← [L], [addr+1] ← [H].
- LDAX rp. (LOAD accumulator indirect)
  [A] ← [[rp]]
- STAX rp. (Store accumulator indirect)
  [[rp]] ← [A].
- XCHG. (Exchange the contents of H-L with D-E pair)
  [H-L] ↔ [D-E].

5.4.2 Arithmetic Group

All the arithmetic operations like addition, subtraction; increment or decrement comes under this category.

Example #1:

DAA (Decimal adjust accumulator)- The instruction DAA is used in the program after ADD, ADI, ACI, ADC, etc instructions. After the execution of ADD, ADC, etc instructions the result is in hexadecimal and it is placed in the accumulator. The DAA instruction operates on this result and gives the final result in the decimal system. It uses carry and auxiliary carry for decimal adjustment. 6 is added to 4 LSBs of the content of the accumulator if their value lies in between A and F or the AC flag is set to 1. Similarly, 6 is also added to 4 MSBs of the content of the accumulator if their value lies in between A and F or the CS flag is set to 1. All status flags are affected. When DAA is used data should be in decimal numbers.

The example of this category instructions are:

- ADD r. (Add register to accumulator)

  [A] ← [A] + [r].
- ADD M. (Add memory to accumulator)

  [A] ← [A] + [[H-L]].
- ADC r. (Add register with carry to accumulator).

  [A] ← [A] + [r] + [CS].
- ADC M. (Add memory with carry to accumulator)

  [A] ← [A] + [[H-L]] [CS].
- ADI data (Add immediate data to accumulator)

  [A] ← [A] + data.
- ACI data (Add with carry immediate data to accumulator)

  [A] ← [A] + data + [CS].
- DAD rp. (Add register paid to H-L pair)

  [H-L] ← [H-L] + [rp].
- SUB r. (Subtract register from accumulator)

  [A] ← [A] – [r].
- SUB M. (Subtract memory from accumulator)

  [A] ← [A] – [[H-L]].
- SBB r. (Subtract register from accumulator with borrow)

  [A] ← [A] – [r] – [CS].
- SBB M. (Subtract memory from accumulator with borrow)

  [A] ← [A] – [[H-L]] – [CS].
- SUI data. (Subtract immediate data from accumulator)

  [A] ← [A] – data.
- SBI data. (Subtract immediate data from accumulator with borrow).

  [A] ← [A] – data – [CS].
- INR r (Increment register content)

  [r] ← [r] +1.
- INR M. (Increment memory content)

  [[H-L]] ← [[H-L]] + 1.

- DCR r. (Decrement register content).
  [r] ← [r] – 1.
- DCR M. (Decrement memory content)
  [[H-L]] ← [[H-L]] – 1.
- INX rp. (Increment register pair)
  [rp] ← [rp] – 1.
- DCX rp (Decrement register pair)
  [rp] ← [rp] -1.

5.4.3 Logical Group

All the instructions related to logical operations like AND, OR, compare, etc. in data are grouped under logic group instructions.

The example of this category instructions are:

- ANA r. (AND register with accumulator)
  [A] ← [A] ^ [r].
- ANA M. (AND memory with accumulator).
  [A] ← [A] ^ [[H-L]].
- ANI data. (AND immediate data with accumulator)
  [A] ← [A] ^ data.
- ORA r. (OR register with accumulator)
  [A] ← [A] v [r].
- ORA M. (OR memory with accumulator)
  [A] ← [A] v [[H-L]]
- ORI data. (OR immediate data with accumulator)
  [A] ← [A] v data.
- XRA r. (EXCLUSIVE – OR register with accumulator)
  [A] ← [A] v [r]
- XRA M. (EXCLUSIVE-OR memory with accumulator)
  [A] ← [A] v [[H-L]]
- XRI data. (EXCLUSIVE-OR immediate data with accumulator)
  [A] ← [A]

- CMA. (Complement the accumulator)

  [A] ← [A]
- CMC. (Complement the carry status)

  [CS] ← [CS]
- STC. (Set carry status)

  [CS] ← 1.
- CMP r. (Compare register with accumulator)

  [A] – [r]
- CMP M. (Compare memory with accumulator)

  [A] – [[H-L]]
- CPI data. (Compare immediate data with accumulator)

  [A] – data.

  The 2nd byte of the instruction is data, and it is subtracted from the content of the accumulator. The status flags are set according to the result of subtraction. But the result is discarded. The content of the accumulator remains unchanged.
- RLC (Rotate accumulator left)

  [An+1] ← [An], [A0] ← [A7],[CS] ← [A7].

  The content of the accumulator is rotated left by one bit. The seventh bit of the accumulator is moved to carry bit as well as to the zero bit of the accumulator. Only CS flag is affected.
- RRC. (Rotate accumulator right)

  [A7] ← [A0], [CS] ← [A0], [An] ← [An+1].

  The content of the accumulator is rotated right by one bit. The zero bit of the accumulator is moved to the seventh bit as well as to carry bit. Only CS flag is affected.
- RAL. (Rotate accumulator left through carry)

  [An+1] ← [An], [CS] ← [A7], [A0] ← [CS].
- RAR. (Rotate accumulator right through carry)

  [An] ← [An+1], [CS] ← [A0], [A7] ← [CS]

5.4.4 Branch Control Group

Normally the order of execution of the instructions of the program is sequential. Often we encounter a situation in real world programming where we want the control of the program jump to some location as specified by the instruction. For which, a condition is tested. If the condition holds true, the instruction at the location specified in the instruction is executed, else the instruction in the next sequential location is executed. This is known as conditional jump. At times, we encounter a situation where we want to execute an instruction which is not located at the next sequential location at any cost and we don't want any condition to hold true for that. This is unconditional jump. And such situation arises when the CPU is executing as program and an urgent operating system related subroutine need to be executed, or an interrupt occurs. Branch control group contains such instructions which are used for conditional and unconditional jump, subroutine call and return, and restart purposes.

This group includes the instructions. Examples are:

- JMP addr (label)- (Unconditional jump: jump to the instruction specified by the address). [PC] ←Label.
- Conditional Jump addr (label)- After the execution of the conditional jump instruction the program jumps to the instruction specified by the address (label) if the specified condition is fulfilled. The program proceeds further in the normal sequence if the specified condition is not fulfilled. If the condition is true and program jumps to the specified label, the execution of a conditional jump takes 3 machine cycles: 10 states. If condition is not true, only 2 machine cycles; 7 states are required for the execution of the instruction.
- JZ addr (label)- (Jump if the result is zero)
- JNZ addr (label)- (Jump if the result is not zero)
- JC addr (label)- (Jump if there is a carry)
- JNC addr (label)- (Jump if there is no carry)
- JP addr (label)- (Jump if the result is plus)
- JM addr (label)- (Jump if the result is minus)
- JPE addr (label)- (Jump if even parity)
- JPO addr (label)- (Jump if odd parity)

- CALL addr (label)- (Unconditional CALL: call the subroutine identified by the operand)
- CALL instruction is used to call a subroutine- Before the control is transferred to the subroutine, the address of the next instruction of the main program is saved in the stack. The content of the stack pointer is decremented by two to indicate the new stack top. Then the program jumps to subroutine starting at address specified by the label.
- RET (Return from subroutine)
- RST n (Restart)- Restart is a one-word CALL instruction. The content of the program counter is saved in the stack. The program jumps to the instruction starting at restart location.

5.4.5 I/O and Machine Control Group

This group includes the instructions for input/output ports, stack and machine control.

The example of this category instructions are:

- IN port-address. (Input to accumulator from I/O port)
  [A] ← [Port]
- OUT port-address (Output from accumulator to I/O port)
  [Port] ← [A]
- PUSH rp (Push the content of register pair to stack)
- PUSH PSW (PUSH Processor Status Word)
- POP rp (Pop the content of register pair, which was saved, from the stack)
- POP PSW (Pop Processor Status Word)
- HLT (Halt)
- XTHL (Exchange stack-top with H-L)
- SPHL (Move the contents of H-L pair to stack pointer)
- EI (Enable Interrupts)
- DI (Disable Interrupts)
- SIM (Set Interrupt Masks)
- RIM (Read Interrupt Masks)
- NOP (No Operation)

5.5 Unspecified Instruction Set

Examine the instruction set of 8085a, one finds that out of the 256 possible op-code with 2 bits, Intel announces only 246, there are no announced instruction corresponding to the 10 missing opcode[11]. User have since reported that these missing opcode. The flags specified by Intel are S, Z, AC, P, CY located in the flag register as

S	**Z**	**X**	**AC**	**X**	**P**	**X**	**CY**

Users have reported that there are two more flag bits having same useful meaning.

S	**Z**	**X_5**	**AC**	**X**	**P**	**V**	**CY**

V = overflow bit. This bit is set to 1 of overflow occurs in 2's complement addition / subtraction/ DCX/INX for8 and 16- arithmetic operation.

X_5 their bit has been named for its position in the condition code byte 2t does not resemble any normal flag bat this bit is affected as per the following expression.

$X5 = S_1.S_2 + SI.R + SL.R$

Where, S_1= sign of operand 1

S_2= sign of operand 2

R= sign of result.

For subtraction & compression replace S_1 by $\overline{S}_2$ where operand 2 the subtraction i.e . result, operand 1- operand 2. The only use for the bit termed are as an unsigned overflow indication resulting from a data change of FFFF to 0000 on executing the instruction INX and as an unsigned underflow indicator from a data change of 0000 to FFFF an executing DCX.

The ten new instructions, include seven opcode that involve the processing of register pairs, the involve jump operation with X_5 flag bit are that perform a conditional flag bit V.

[11] Adopted from: http://nptel.ac.in/courses/108107029/28 available under creative commons license.

The various instructions are:

1. DSUB: (double subtraction): the macro RTL implements is

   $(H, L) \leftarrow (H, L) - (B, C)$

   This is a single byte instruction. The opcode is $(OB)_H$. The meaning of the instruction is the content of register pair BC are subtracted foam the content of register pair (HL) and result placed back in register pair (H,L). All seven condition flag are affected. It requires 3 m/c cycles and 10 states like DAD instruction. The addressing mode is register addressing mode.

2. ARHL: (arithmetic right shift HL register) the macro RTL implemented is

   $H_7 \leftarrow (H_7)$, $H_{n-1} \leftarrow H_n$,

   $L_7 \leftarrow H_0$, $L_{n-1} \leftarrow L_n$, $CY \leftarrow L_0$

   This is a single byte construction the opcode is $(10)_H$. the meaning of the instruction is the contents of register pair (HL) are shifted right by one bit. The upper most bit is duplicated and the lower bit is shifted into the carry bit. The result is placed back into the (H, L) register pair only ax flag is affected It requires 2 m/c cycles and 7 statues the addressing mode is register address mode.

3. RDEL: (Rotate D.E reg. pair left through carry) the macro RTL implements

   $CY \leftarrow D_7$, $D_{n+1} \leftarrow D_E$,

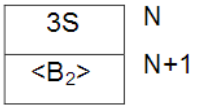
   $D_0 \leftarrow E_7$, $E_{n+1} \leftarrow E_n$, $E_0 \leftarrow CY$.

   This is a single byte instruction. The operation code is 18_4. The meaning of the instruction is the content of repair DE is rotated left by one b through the carry flag. The low order but is set equal to the CY flag and the CY flag is set to the value shifted out of the V flag bits are affected. The result is placed book into the DE register pair. It requires 3m/c cycle and 10 states. The addressing mode is register addressing mode.

4. LDHI: (Lead D, E reg pair with (H, L) flow immediate data). The macro RTL implements is

   (D,E) ← (H,L)+<B2>

   This is a 2 byte instruction the operation code format is

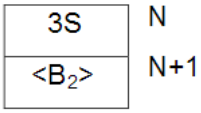


   The meaning of the instruction in the contain of register pair (H, L) are added to the immediate byte. The result is placed in register pair (D, E) no condition flags are affected. It requires 3m/c cycles and 10 states. The addressing mode is immediate & register addressing mode the second byte is called offset.

5. LDSI: (Load (D, E) reg pain with SP flow immediately byte). The macro RTL implements is

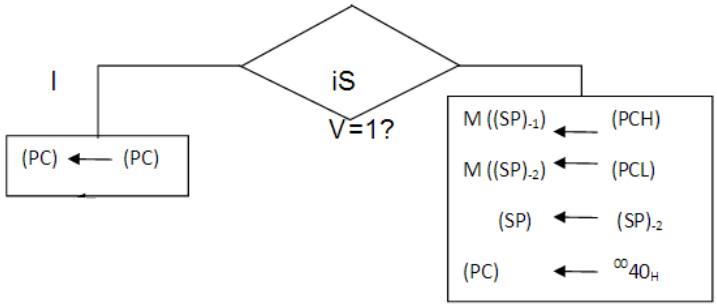
   (D.E) ← (BP) + <B2>

   This is a byte instruction the opcode format is



   The meaning of the instruction is the content of register pair SP are added to the immediate byte & the results is placed in register pair (D,E). No condition flag are affected. It requires 3m/c cycle and 10 states. The addressing mode is immediate register addressing mode.

6. RSTV (Restart in overflow): The macro RTL implements is shown below



This is a single byte instruction. The opcode is $(CS)_H$. The meaning is the overflow flag V is set, the actions specified above are performed, otherwise control continues sequentially.

It requires 1 or 3 m/c cycles and 6 or 12 states. The addressing mode is register indirect addressing mode none flags are affected.

7. SHLX: (Store H,L indirect through D,E) the macro RTL implemented is

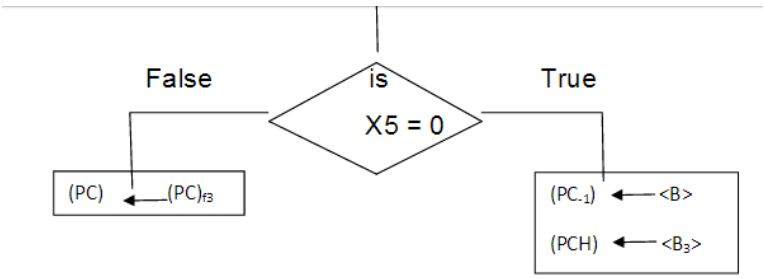
   M ((D,E)) ← (L)

   M ((DE)+1) ← (H)

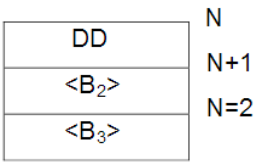
   This is a single byte instruction. The opcode is $(D9)_H$. The meaning is the content of register L are moved to the memory location whose addressing register DE. The contents of register are moved to the succeeding memory location.

   It requires 3m/c cycles and 10 states. The addressing mode is register indirect addressing mode. None flags are affected.

8. JNX5: (Jump on not X5) The macro RTL implemented is shown below

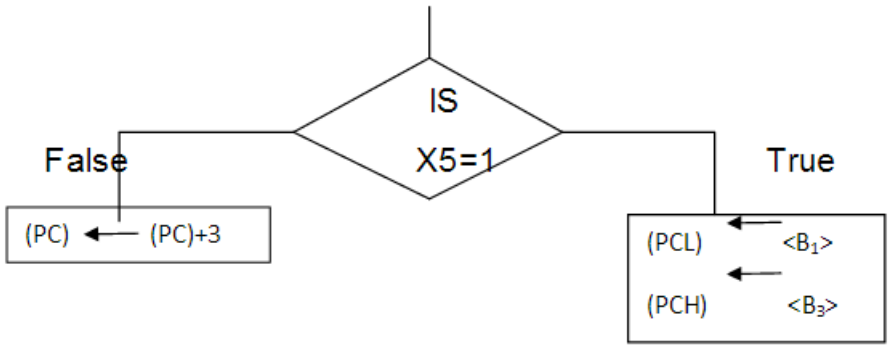


Where (PC) is the address for opcode. This is an S byte instruction. The operation code format is

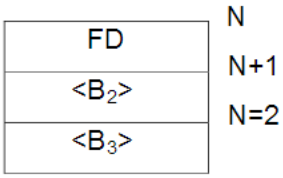


This instruction tests the X5 bit, if the X5 bit is '0', the control is transferred to the instruction whose address is specified in byte 2 & byte 3 of the current instruction, otherwise control continues sequentially. It requires 2 or 3 machine cycle of 7 to 10 states. The addressing mode is immediate mode. None flag are affected.

9. JX5: (Jump on X_5): the macro RTL implemented is shown below.



This is an 8 byte instruction. The operation code formats



This instruction test X_5 bit if X_5 bit is set, the control is transferred to the instruction whose address is specified in byte 3&2 of the current instruction otherwise the control continues sequentially.

It requires 2 or 3 m/c cycles and 7 or10 states. The addressing mode is immediate none flags are affected

10. LLX: (Load H,L indirect through DE) The macro RTL implemented is (DE)

(L) ← M (D, E)

(H) ← M ((D, E) +1)

This is a single byte instruction. The operation code is ED_H. The meaning is the content of the memory location whose address is in (DE) reg. pair are moved to register L, & the content of the succeeding memory location are moved to register H. it requires 3m/c cycles and 10 states. The addressing mode is register indirect none flags are affected

5.6 Summary

1. An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
2. To perform a specific task, a program, which is a sequence of instructions, is written and through this sequence of instructions we instruct the computer to perform what operation is to be performed, on what data and in which sequence.
3. The instruction consists of two parts viz. operation code and operand.
4. The various ways of specifying data (or operands) for instructions are called as addressing modes.
5. The 16-bit address location of the operand stored in a register pair (H-L) is given in the instruction. The address of the operand is given in an indirect way with the help of a register pair. So it is called Register indirect addressing mode.
6. The mode of instruction which do not specify the operand in the instruction but it is implicated, is known as implicit addressing mode.
7. All the arithmetic operations like addition, subtraction; increment or decrement comes under arithmetic group.
8. All the instructions related to logical operations like AND, OR, compare, etc. in data are grouped under logic group instructions.

5.7 Check Your Progress

1. An _____ is a computer program which translates assembly language to machine language format.
2. Op-code stands for _____.

3. In _____ addressing mode operand is a part of the instruction itself.
4. The mode of addressing in which the 16-bit address of the operand is directly available in the instruction itself is called _____ Addressing mode.
5. In _____ addressing mode the operands are microprocessor registers only i.e. the operation is performed within various registers of the microprocessor.
6. The 8085 microprocessor have _____ instructions.
7. _____ category of instructions are used to transfer the content of source register to another destination register and after the transfer of the data, the content of the source remains unaltered.
8. _____ group includes the instructions for input/output ports, stack and machine control.

5.8 Answers to Check Your Progress

1. Assembler
2. Operation code
3. Immediate
4. Direct
5. Register
6. 246
7. Data transfer group
8. I/O and Machine Control

5.9 Model Questions

1. Explain 1 byte, 3 byte and 3 byte instructions with example.
2. What are the three groups of instructions into which 8085 instructions are classified?
3. What is an addressing mode? What are the different addressing modes in 8085 microprocessor?
4. What is a conditional jump? Explain.
5. Explain the difference between a JMP instruction and CALL instruction.
6. What is Unspecified Instruction Set? Explain.

ASSEMBLY LANGUAGE PROGRAMMING USING 8085

6.0 Learning Objectives

After going through this unit, you will be able to:

- Define assembly language
- Differentiate machine language with assembly language
- Understand programming model of 8085 microprocessor
- Classify instruction set of 8085 into data transfer, arithmetic, logical, branch and machine control instructions
- Write assembly level programs for 8085 microprocessor

6.1 Introduction

The microprocessor can perform the tasks for which it is designed and can perform only those tasks which are defined in its instruction set. The instruction set contains the set of all the possible operations which a microprocessor can perform. To perform a particular task, the corresponding instruction designed to execute the particular task is given by the user. These instructions should be given in the language known to the microprocessor i.e. machine language which is in the form of strings of 0's and 1's. For example 00000111is a valid machine level instruction in 8085 which rotate each bit of accumulator by one position to the left.

A machine language program to add two numbers is as follows:

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

You must have observed that it is very difficult to remember the 8 bit binary code for the instructions and remember, 8085 have 246 unique instructions. Therefore, it is very difficult to remember the instruction code in machine language as we are not in a habit of using the language of 0's and 1's in our daily life. Moreover it is a time consuming process. Therefore the instructions are written in assembly language by the user. Assembly language is easier for a human to read and can be written faster, but it is still much harder for a human to use than a high-level programming language which tries to mimic human language.

An assembly language is a programming language that can be used to directly tell the computer what to do[12]. An assembly language is almost exactly like the machine language that a computer can understand, except that it uses words in place of numbers. A computer cannot really understand an assembly program directly. However, it can easily change the program into machine code by replacing the words of the program with the numbers that they stand for. A program that does that is called an assembler.

Programs written in assembly language are usually made of instructions, which are small tasks that the computer performs when it is running the program. They are called instructions because the programmer uses them to instruct the computer what to do. The part of the computer that follows the instructions is the processor.

The assembly language of a computer is a low-level language, which means that it can only be used to do the simple tasks that a computer can understand directly. In order to perform more complex tasks, one must tell the computer each of the simple tasks that are part of the complex task. For example, a computer does not understand how to print a sentence on its screen. Instead, a program written in assembly must tell it how to do all of the small steps that are involved in printing the sentence.

With assembly language, each instruction can be written as a short word, called a mnemonic, followed by other things like numbers or other short words. The mnemonic is used so that the programmer does not have to remember the exact numbers in machine code needed to tell the computer to do something. Examples of mnemonics in assembly language include ADD, which adds data, and MOV, which moves data from one place to another. Because 'mnemonic' is an

[12] https://simple.wikipedia.org/wiki/Assembly_language

uncommon word, the phrase instruction type or just instruction is sometimes used instead, often incorrectly. The words and numbers after the first word give more information about what to do. For instance, things following an add might be what two things to add together and the things following mov say what to move and where to put it. A simple program to add two numbers in assembly language is as follows:

MVI A, 2H; Copy value 2H in register A

MVI B, 4H; Copy value 4H in register B

ADD B; A= A+B

You must have easily observed that it is comparatively easy to remember the mnemonics in assembly code then the binary code. However, the assembly language is platform dependent or machine dependent. Assemble language of Intel 8085 is different from Intel 8086 although both the microprocessors are manufactured by same manufacturer i.e. Intel.

6.2 Programming Model of 8085

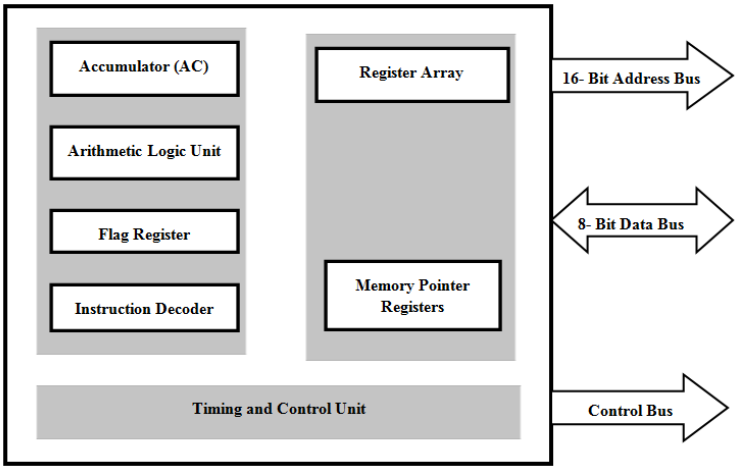


Figure 54: Block Diagram of 8085 Microprocessor

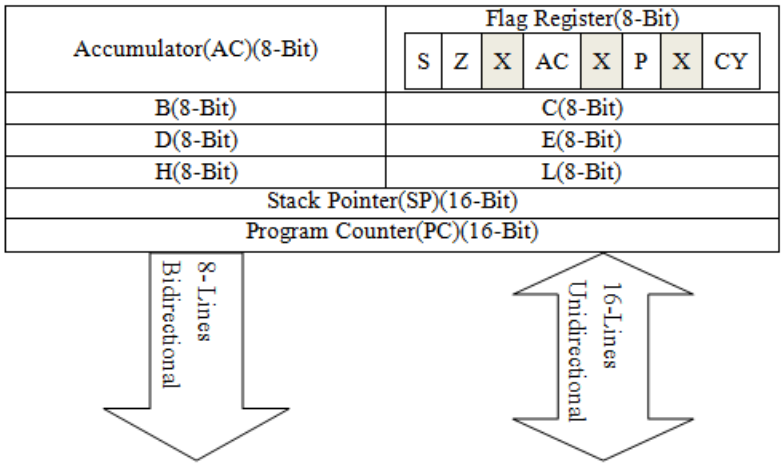


Figure 55: Registers in 8085 microprocessor

Following is the detail of the programming model of 8085:

1. Intel 8085 microprocessor have six 8-bit general purpose registers viz. B,C,D,E,H and L. However these registers can be paired as BC, DE and HL to perform 16-bit operations.
2. 8-bit flag register, which is the part of ALU. There are following five flip-flops in the flag register:
    a. Z(Zero flag)
    b. CY(Cary Flag)
    c. S(Sign Flag)
    d. P(Parity Flag)
    e. AC(Auxiliary Flag)

    These flags are set/ reset by execution of arithmetic and logical operations. The status of the flag registers can be tested through the software instructions and microprocessor uses these flags for decision making process.

3. 8-bit general purpose register, known as Accumulator (AC) is used to perform arithmetic and logic operation by ALU. The result of the operation is stored in the accumulator.
4. 16-bit Program Counter (PC), which is used to store the address of the next instruction to be executed by the microprocessor. It is used to sequence the execution of the program instructions. Most of the time the program is stored in the sequential locations in the memory. While the current instruction is executed by the processor, the PC is

incremented by 1. Therefore, the PC always holds the address of the next instruction to be executed by the microprocessor. In case there is a conditional or unconditional branching in the program, PC is updated by the address of the branching instruction in the memory. Hence, after the execution of the current instruction, the new instruction is fetched from the branching address specified by the PC and not from the next sequential memory location.

5. 16- bit Stack Pointer(SP) register, which points to the top-most occupied location of the stack memory. Initially when the stack is empty, the base address or the starting address of the stack is loaded in the SP. After the addition of each element in the stack memory, SP is incremented by1.

6.3 Instruction Set of 8085

The 8085 microprocessor instructions can be classified into following groups:

1. Data transfer
2. Arithmetic
3. Logical and bit manipulation
4. Branch
5. Machine Control

6.3.1 Data Transfer Instructions

The data transfer instructions are similar to copy instruction where the content of the source are retained after the transfer of data from source to destination. These instructions are used to:

- Load 8-bit number in a register. For example, MOV B, 24H; This instruction loads a 8-bit number 24 in register B.
- Copy from register to register. For example, MOV A,B; This instruction copy the content of register B to A.
- Copy between register and memory. For example, MOV M, B; This instruction copy the content of register B to the memory location specified by M.
- Copy between I/O ports and accumulator register. For. Example, OUT 01H; This instruction copy the content of the Accumulator to output port 01.

- Load a 16-bit address in register pairs BC, DE and HL. For example, LXI H, 1040H; A 16-bit number 1040 is loaded in the HL register pair.
- Copy between register pair and stack memory. For example SPHL; This instruction copy the data from the HL register pair to Stack Pointer (SP).

6.3.2 Arithmetic Instructions

These types of instructions perform arithmetic operations on the data. Some of the arithmetic instructions supported by 8085 are:

- Addition. For example, ADI 32H; This instruction add the 8-bit number 32 to the content of Accumulator.
- Subtraction. For example, SUI 32H; This instruction subtract the 8-bit number 32H from the content of Accumulator.
- Increment. For example, INR D; This instruction increments the content of register D by1.
- Decrement. For Example DCR E; This instruction decrements the content of the register E by1.

6.3.3 Logical and Bit Manipulation Instructions

These instructions perform logical and bit manipulation on the data stored in the registers. Some of the logical and bit manipulation instructions supported by 8085 are:

- AND. For example, ANA H; This instruction logically AND the content of H register with the content of an Accumulator.
- OR. For example, ORA L; This instruction Logically OR the content of L register with the contents of an Accumulator.
- Exclusive OR. For example, XRA B; This instruction logically XOR the contents of B register with the contents of an Accumulator.
- Compare. For example, CMP C; This instruction compares the contents of register C with the contents of an Accumulator.
- Complement. For example, CMA; This instruction compliments the contents of an Accumulator.
- Rotate. For example, RAL; This instruction rotates the contents of an Accumulator left.

6.3.4 Branching Instructions

These instructions are used to control the flow of program execution. Some of the branching instructions supported by 8085 are:

- JUMP. For example, JC 2080H; This instruction is a conditional jump statement and the program to the 16-bit address 2080H in the carry flag of the status register is SET. Otherwise the next instruction is fetched from the next sequential location of the memory.
- Unconditional JUMP. For example, 2050H; This instruction jumps to the 16-bit address 2050H unconditionally without checking any conditions.
- Unconditional CALL. For example, CALL 3050H; This instruction call a subroutine with its 16-bit address as 3050H.
- Return Back from Unconditional CALL. For example, RET; This instruction return back from the CALL.
- Conditional CALL. For example, CNC 3050H; This instruction CALL a subroutine with is 16-bit address as 3050H if the Carry flag of the status register is RESET.
- Conditional RETURN. For example, RZ; This is a conditional RETURN from the subroutine call if the ZERO flag of the status register is SET.

6.3.5 Machine Control Instructions

These instructions affect the operation of the processor. Some of the machine control instructions supported by 8085 are:

- HLT. This instruction stops the program execution.
- NOP. This instruction does not perform any operation.

6.4 Writing an Assembly Level Program

The assembly language is machine dependent i.e. every series of microprocessor have its own unique instruction set. For example Intel 8085 has different instruction set than Motorola 6800. But the basic concept of writing an assembly level program is same. The only difference is in the step 5 mentioned below. Following are the different steps to write an assembly level program for 8085 microprocessor:

1. Analyze the problem
2. Develop program logic
3. Write and algorithm
4. Make a flowchart
5. Write program Instruction using Assembly language of 8085

6.4.1 Addressing modes of 8085

The various formats of specifying operands are called addressing modes. Following are the addressing modes supported by 8085 microprocessor.

1. Register Addressing: In register addressing mode, the operand resides in one of the internal registers of the microprocessor. For example:
   MOV A,B;

   ADD C;
2. Immediate Addressing: In immediate addressing mode, the operand is specified in the instruction itself. For example:
   MVI A, 20H;

   LXI H, 2050H;
3. Memory Addressing: In memory addressing mode, one of the operand is a memory location which holds the operand. Depending on how address of memory location is specified, memory addressing is of two types:
   a. Direct memory addressing: In case of direct memory addressing mode, 16-bit address of the memory location which holds the operand is specified in the instruction only. For example:
      LDA 2050H; Load Accumulator with the operand which is located at the memory location 2050H of the memory.
   b. Indirect memory addressing: In case of indirect memory addressing mode, a memory pointer register is used to store the address of the memory location. For example:
      MOV M,A; This instruction move the contents of Accumulator to the memory location, whose address is specified by the contents of HL register pair.
4. Input/output Addressing: In Input/ Output addressing mode, the 8-bit address of the port is directly specified in the instruction itself. For example:
   IN 07H;

6.5 Sample Assembly Level Programs in 8085 Microprocessor

Program #1: Store the data byte 32H into memory location 4000H.

Step 1: Analyze the problem

- 8-bit data (32H) is to be stored in the specified memory location(4000H).

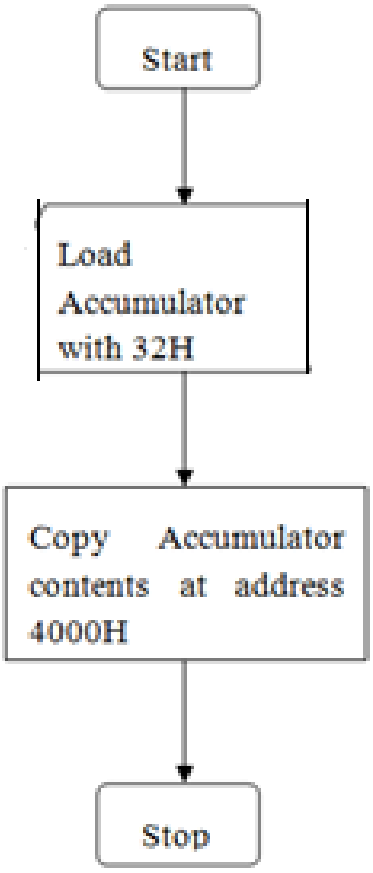
Step 2: Program Logic

- Load the 8-bit data in Accumulator
- Store the content of Accumulator in memory

Step 3: Algorithm

- Store 32H in Accumulator
- Copy the content of an Accumulator to memory location 4000H.

Step 4: Flow Chart



Step 5: Write an assembly level program using 8085 instruction set.

MVI A, 32H : Store 32H in the accumulator

STA 4000H : Copy accumulator contents at address 4000H

HLT : Terminate program execution

Program #2: Subtract the contents of memory location 4001H from the memory location 4000H and place the result in memory location 4002H.

Memory Location	Data
0000H	
.	
.	
4000H	**51H**
4001H	**19H**
4002H	

Step 1: Analyze the problem

- Subtract two 8-bit numbers

Step 2: Program Logic

- Load the first operand into the accumulator from memory
- Subtract the second operand from the content of AC
- Store the result in memory

(4000H) = 51H

(4001H) = 19H

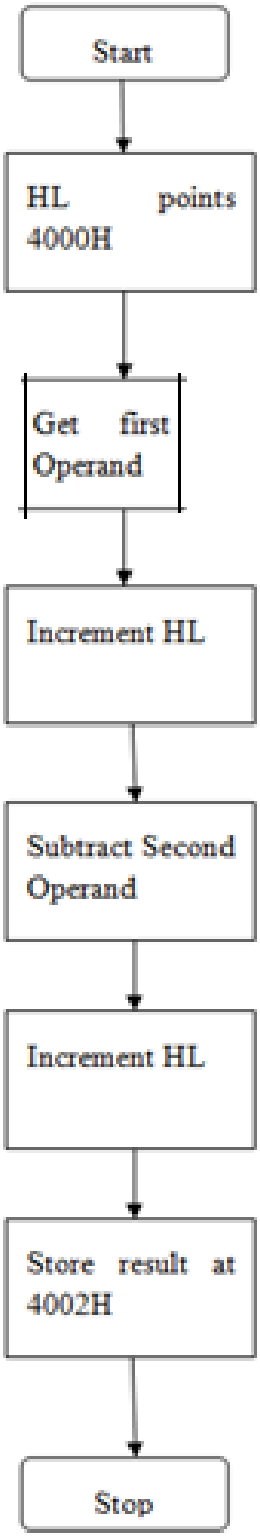
Result = 51H - 19H = 38H

(4002H) = 38H

Step 3: Algorithm

- Move the first operand from memory to accumulator
- Subtract the second operand from the content of accumulator
- Store the result in memory

Step 4: Flowchart



Step 5: Write an assembly level program using 8085 instruction set.

LXI H, 4000H : HL points 4000H

MOV A, M : Get first operand

INX H : HL points 4001H

SUB M : Subtract second operand

INX H : HL points 4002H

MOV M, A : Store result at 4002H.

HLT : Terminate program execution

Program #3: Divide two 8-bit numbers by repeated subtraction.

Step 1: Analyze the program

- Divide two 8-bit numbers by repeated subtraction.

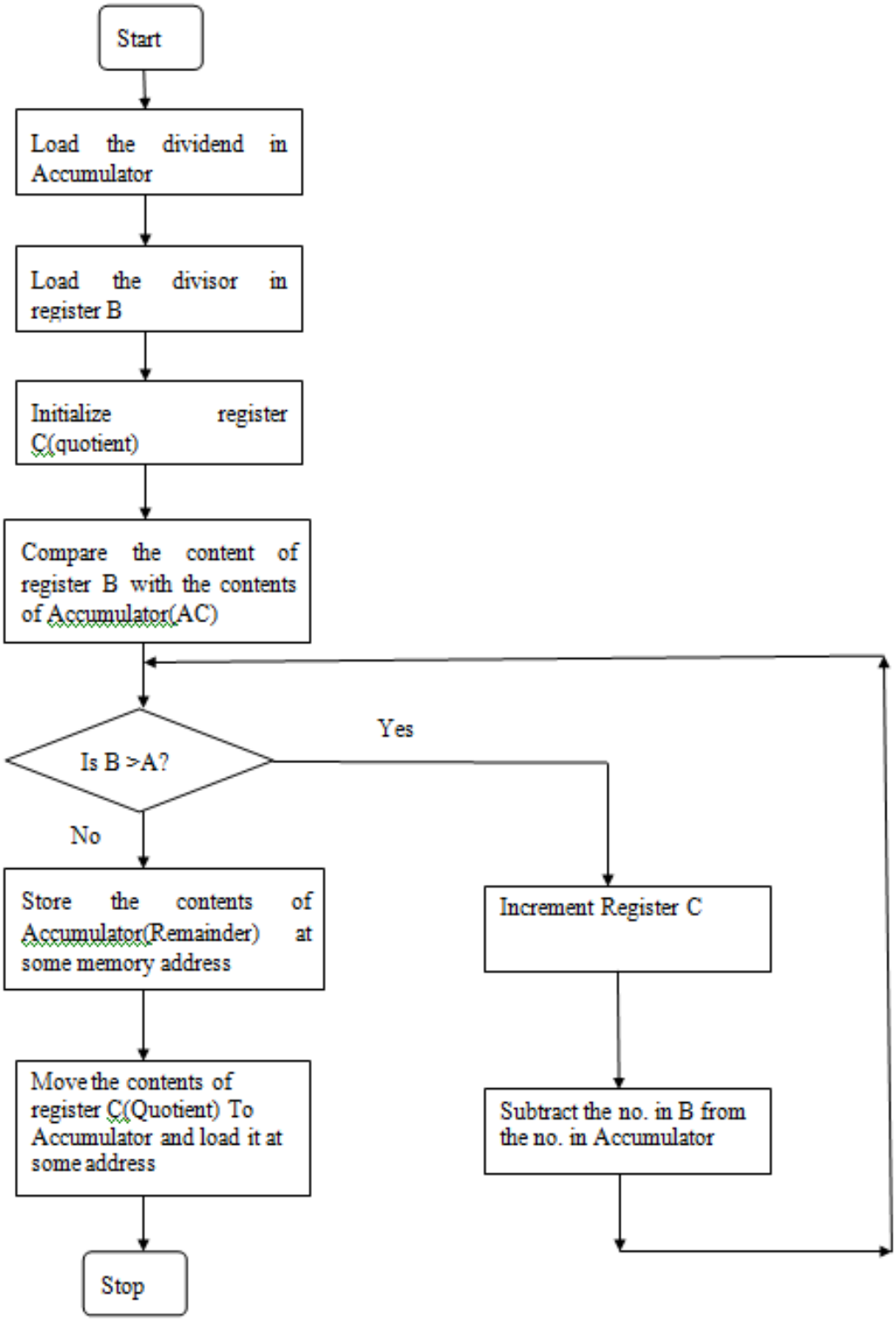
Step 2: Program Logic

- Load dividend and devisor from the two successive memory locations.
- Subtract the divisor from dividend repeatedly until the difference becomes less than devisor.
- Calculate quotient and reminder.

Step 3: Algorithm

1. Load dividend and divisor from the two successive memory locations.
2. Subtract the divisor from the dividend repeatedly until the difference becomes less than devisor.
3. Store the number of times the subtraction took place as quotient and the remaining difference as reminder.
4. End of program.

Step 4: Flow Chart



Step 5: Write an assembly level program using 8085 instruction set.

LDA 2050H; Load Divisor in Accumulator

MOV B,A; Copy Divisor to Register B

LDA 2051H; Load Dividend in Accumulator

MVI C, 00H; Initialize register C for Quotient

CMP B; Compare the no. in B with the no. in Accumulator

JC 2010H; Jump if B>A to address 2010

INR C; Increment Register C (Quotient)

SUB B; Subtract B from A

JMP 2007H; Repeat the above steps till A becomes smaller than B

STA 2052H; Store the remainder at memory address 2052

MOV A,C; Move the contents of C to Accumulator

STA 2053H; Store the Quotient at memory address 2053

HLT; Stop the execution of the program

Program #4: Add two 8-bit numbers using direct addressing mode.

Step 1: Analyze the problem

- Add two 8-bit numbers using direct addressing mode.

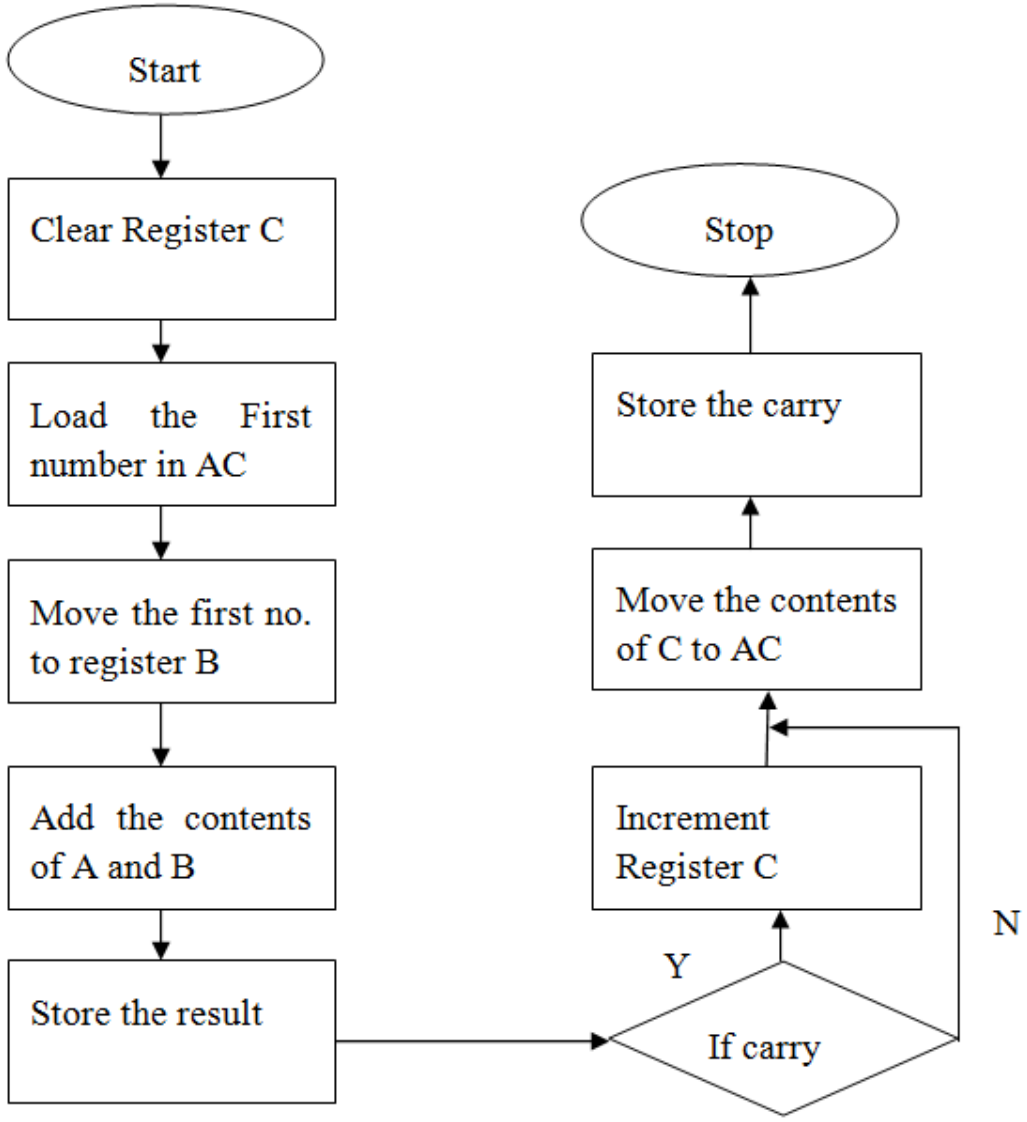
Step 2: Program Logic

- Input two 8-bit numbers
- Add them

Step 3: Algorithm

1. Enter two 8bit numbers.
2. Add them.
3. Store the result and carry.
4. End of Program

Step 4: Flowchart



Step 5: Write an assembly level program using 8085 instruction set.

LDA ,2050H;	Clear Accumulator.
MOV C,A;	Clear Register C.

LDA ,2051H;	Store the First no. in Accumulator.
MOV B,A;	Copy this No. to Register B.
LDA, 2052H;	Store the second Number in Accumulator.
ADD B;	Add the two Numbers.
STA ,2053H;	Store the result .
JNC , 2013;	Check for carry.
INR C;	Increment Register C.
MOV A,C;	Copy the Carry to Accumulator.
STA ,2054H;	Store the Carry.
HLT;	Program Terminated.

Program # 5: Add two 8-bit hexadecimal numbers, with carry using Indirect addressing mode.

Step 1: Analyze program

- Add the two hexadecimal numbers using indirect addressing mode.

Step 2: Program Logic

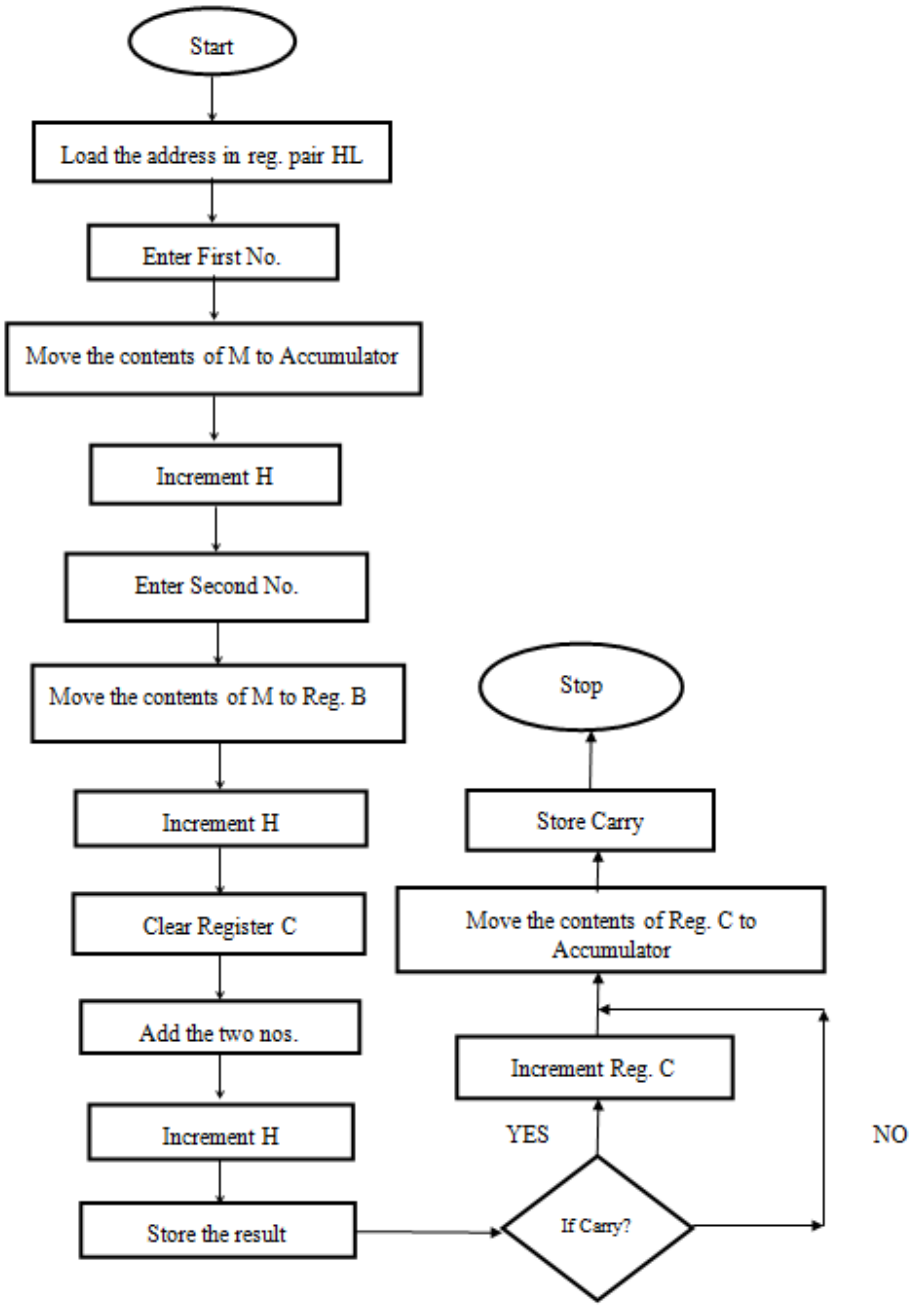
- Load the two hexadecimal numbers in memory at two consecutive locations.
- Load the address of the first operand in the register pair.
- Load the first operand in the Accumulator.
- Load the second operand in the microprocessor register.
- Add the two hexadecimal numbers.
- Store the result and carry in memory.

Step 3: Algorithm

1. Load the two numbers at consecutive addresses.
2. Load the address in HL Register pair.

3. Move the numbers from addresses to Accumulator and another register using the register pair H.
4. Add the two numbers.
5. Store the result and carry using the HL register pair.

Step 4: Flowchart



Step 5: Write an assembly level program using 8085 instruction set.

LXI H,2050H;	Load the memory address
MOV A,M;	First no. in Accumulator.
INX H;	Increment mem. Addr.
MOV B,M;	Second No. in Register B
ADD B;	Add the two no.s
INX H;	Increment the mem. Addr.
MOV C,M;	Clear Register C
INX H;	Increment Memory. Address
MOV M,A;	Store the result
JNC ,2010H;	Check for Carry and store the result
INX H;	Increment memory Address
INR C;	Add Carry
MOV A,C;	Copy contents of Carry in Accumulator
MOV M,A;	Store Carry
HLT;	Program terminated.

Program #6: Multiply two 8-bit numbers by repeated addition.

Step 1: Analyze Problem

Multiplication of two 8-bits numbers using repeated addition method.

Step 2: Program Logic

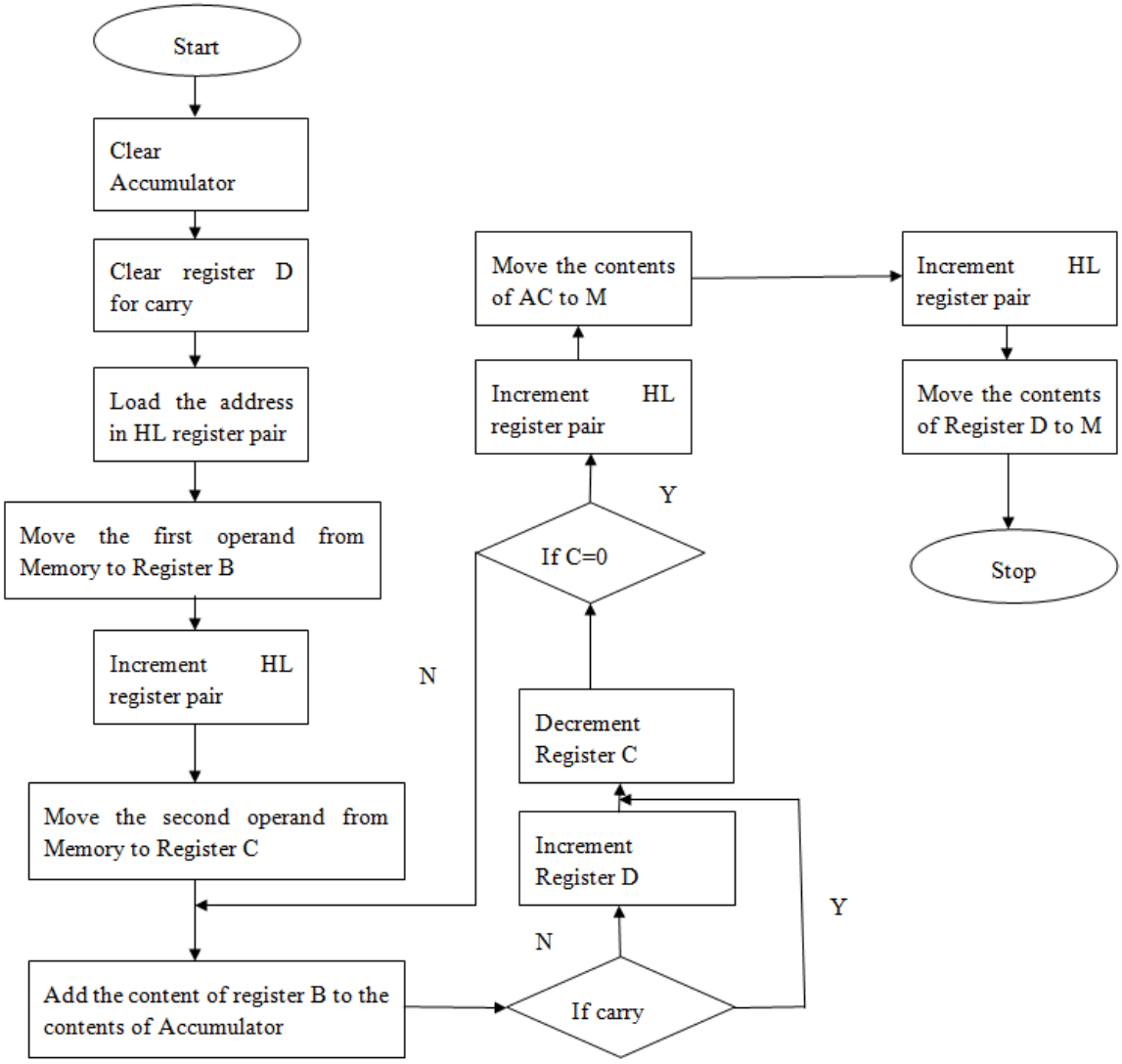
- Load the two operands in registers.
- Add the first number to itself as many times as equal to the value of the second operand.

- Stop

Step 3: Algorithm

1. Load the two numbers from two successive memory locations.
2. Add the first number to itself and decrement the second number.
3. Repeat the second step till the second no. becomes 0.
4. Store the result and carry at some address.
5. End of Program.

Step 4: Flowchart



Step 5: Write an assembly level program using 8085 instruction set.

XRA A;	Clear Accumulator
MOV D,A;	Initialize carry as 0
LXI H 2050H;	Load the address in H reg. pair.
MOV B,M;	Copy the contents(no.), of the memory to reg. B
INX H;	Increment the contents of HL register pair
MOV C,M;	Copy the contents of the memory to the reg. C
ADD B;	Add the the contents of reg. B to the contents of accumulator.
JNC 200DH;	Jump if no carry to address 200D
INR D;	Increment the contents of reg. D(if there is a carry)
DCR C;	Decrement the contents of reg. C
JNZ 2008H;	Jump if the contents of reg. C are not zero, to the addr. 2008
INX H;	Increment the contents of Reg. pair H
MOV M,A;	Copy the contents of the accumulator to the memory(the product)
INX H;	Increment the contents of Reg. pair H
MOV M,D;	Copy the contents of the Reg. D to the memory(carry).
HLT;	End of Program

6.6 Summary

1. The microprocessor can perform the tasks for which it is designed and can perform only those tasks which are defined in its instruction set.

2. Assembly language is easier for a human to read and can be written faster, but it is still much harder for a human to use than a high-level programming language which tries to mimic human language.
3. An assembly language is almost exactly like the machine language that a computer can understand, except that it uses words in place of numbers.
4. The mnemonic is used so that the programmer does not have to remember the exact numbers in machine code needed to tell the computer to do something.
5. Intel 8085 microprocessor have six 8-bit general purpose registers viz. B,C,D,E,H and L. However these registers can be paired as BC, DE and HL to perform 16-bit operations.
6. Program Counter register is used to sequence the execution of the program instructions.
7. The data transfer instructions are similar to copy instruction where the content of the source are retained after the transfer of data from source to destination.
8. Arithmetic instructions perform arithmetic operations on the data.
9. Logical and bit-manipulation instructions perform logical and bit manipulation on the data stored in the registers.

6.7 Check Your Progress

1. With assembly language, each instruction can be written as a short word, called a _____ , followed by other things like numbers or other short words.
2. The _____ are set/ reset by execution of arithmetic and logical operations.
3. 8-bit general purpose register, known as _____ is used to perform arithmetic and logic operation by ALU.
4. 16-bit register called _____ is used to store the address of the next instruction to be executed by the microprocessor.
5. 16- bit _____ register points to the top-most occupied location of the stack memory.
6. _____ instructions are used to control the flow of program execution.
7. _____ instruction does not perform any operation.

6.8 Answer to Check your progress

1. Mnemonic
2. Flags
3. Accumulator
4. Program Counter
5. Stack pointer
6. Branching

7. NOP

6.9 Model Questions

1. Differentiate between machine language and assembly language.
2. Explain the block diagram of 8085 microprocessor.
3. What is a flag register? Why it is used? How many flags are there in 8085 and what is their role in programming?
4. What is the role of program counter (PC)?
5. What are the different steps to write an assembly level program for 8085 microprocessor?
6. Write an assembly level program for 8085 to check the parity of a given number.
7. Write an assembly level program for 8085 to subtract two 8-bit Hexadecimal numbers using Direct Addressing mode.
8. Write an assembly level program for 8085 to subtract two 8-bit hexadecimal numbers, using Indirect addressing mode.

INTERFACING

7.0 Learning Objectives

After reading this chapter, you will be able to:

- Know the concept of bus
- Define interface
- Perform memory interfacing
- Differentiate between absolute and partial decoding
- Understand different address decoding techniques in 8085
- Understand I/O interfacing
- Differentiate between I/O and memory mapped interfacing

7.1 Introduction

The microprocessor need to interact with the memory for fetching the instructions and operands. Also, interaction with memory is required to store the final result of the operation. Therefore, the Microprocessor requires some path, known as bus, through which it can communicate to these devices.

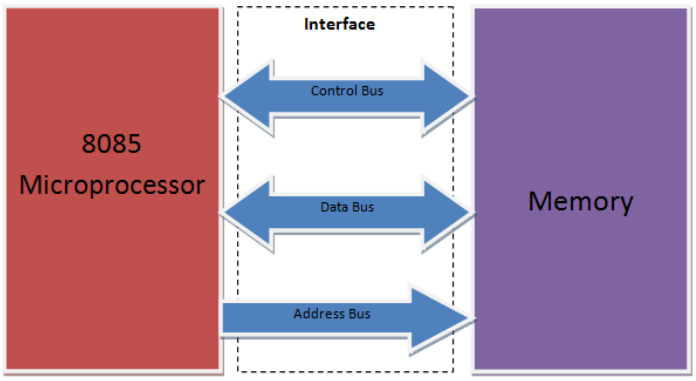


Figure 56: Microprocessor interfacing with memory

In the figure above, the memory is connected to Microprocessor via interface. There are many differences in the implementation of a Microprocessor and memory like a Microprocessor is an electronic device whereas the memory is an electromagnetic devices Therefore, the technology behind these two entities and the manner of operation is different. Also, Microprocessor transmits and processes the data at very fast rate whereas the data transfer rate of memory is comparatively very slow. Moreover, Microprocessor transmits the data in parallel whereas peripheral devices usually transmit data serially. To overcome these mismatches, a special hardware, known as interface is connected between Microprocessor and memory to take care of all the above issues and to supervise and synchronize all input and output transfers. These components are called **interface.**

Microprocessor needs to communicate with memory. So to locate a specific location in the memory, an address need to be specified Also, Microprocessor need to specify the control information i.e., what operation is to be performed. Like in case of memory, CPU needs to specify whether it is a read operation or a write operation. After specifying the address and the control information, the data need to be transferred between Microprocessor and memory. For this purpose, bus is required. Bus is a conducting path that connects the Microprocessor with other devices. Based on the purpose for which the bus is used, it is categorised into following categories.

a. *Address Bus:* An address bus is used by the Microprocessor to transmit the address of the memory location. Since, it is used by Microprocessor only, this bus is unidirectional.

b. *Control Bus:* It is used by the Microprocessor to control and regulate the various devices attached to it. This bus is bidirectional.

c. *Data Bus:* It is used by Microprocessor and the devices attached to the Microprocessor to transfer data. This Bus is bidirectional.

7.2 Memory Interfacing

8085 has a 16-bit address bus (A_{15}-A_0) and hence can support 64K memory (2^{16} = 65,536). The higher-order address bus is unidirectional and it carries the most significant 8-bits (A_{15}-A_8) of a

16-bit memory address and this address is available in the address line till the operation is not completed. The lower-order address bus is bidirectional and multiplexed with the data bus using time division multiplexing and contains the least significant 8-bits (A_7-A_0) memory address. During the first clock cycle, the Address Latch Enable (ALE) is high and 8-bits of the (AD_7-AD_0) lines are transferred to Address latch and constitutes to the 8 least significant bits of the 16-bit memory address. In the second and third clock cycle, when the ALE signal is low, 8-bits of the (AD_7-AD_0) lines are transferred to Data buffer, from where it is transferred to the data line (D_7-D_0).

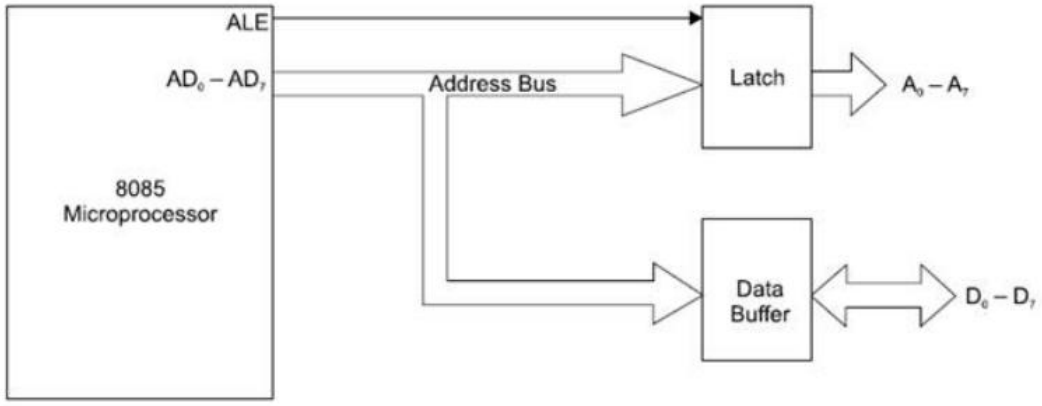


Figure 57: Multiplexing of address and data bus

For performing a memory operation, the following signals are required.

1. n address lines for selecting between 2^n addresses in the memory.
2. Chip Select(CS) signal to selected between the chips in case the memory is implemented using more than one memory chip.
3. IO/$\overline{M}$ signal to select between Input/Output or Memory operation.
4. Read or Write control signal to differentiate between read and write operation.

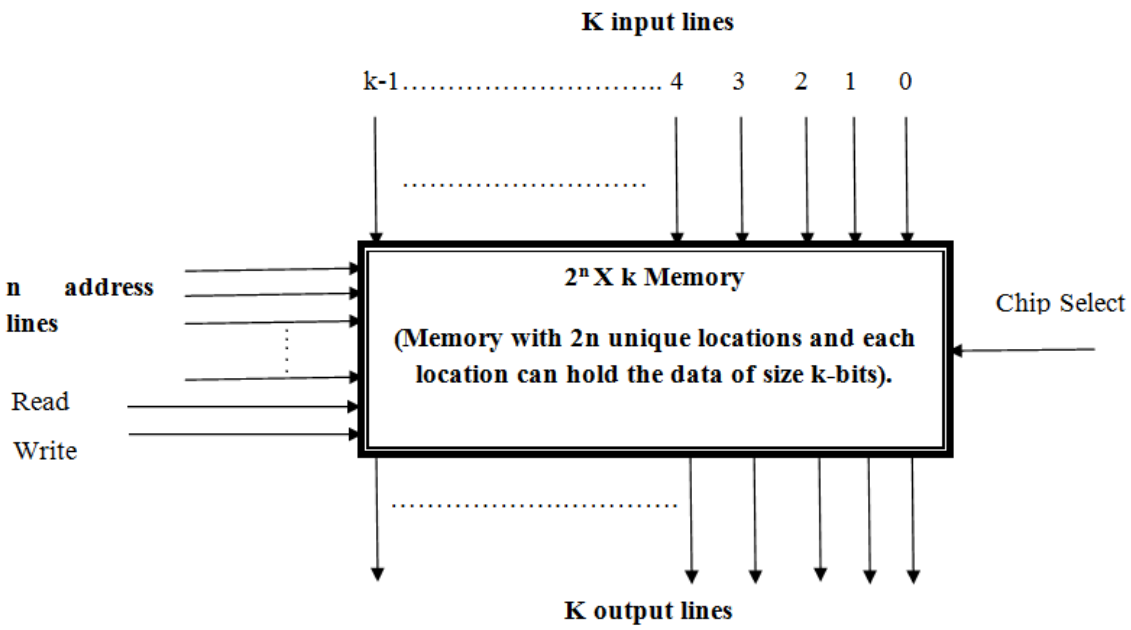


Figure 58: Memory Select Logic

Whenever a memory related operation is to be performed, the most significant 8-bits are placed in the higher order address lines (A_{15}-A_8) during the first clock pulse and the 8 least significant address bits are place in the lower order address line (AD_7-AD_0) with Address Line Enable (ALE) signal set to high. The IO/$\overline{M}$ pin of the microprocessor, which is attached to the $\overline{CS}$ pin of the memory, goes low to enable the chip select. Depending on the Memory Read or the Memory Write operation, the $\overline{RD}$ or the $\overline{WR}$ signals goes to low state to enable read or write operation. In the second clock pulse, the ALE signal goes down and the data is placed in AD_7-AD_0 lines.

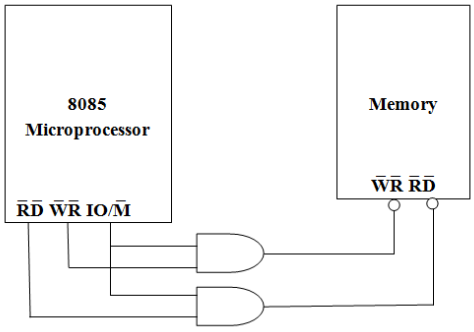


Figure 59: Memory Read/Write Select

In case the memory is implemented using more than one memory chip, the address selection logic is different than that is explained above. To simplify the concept, let us suppose there are two memory chips with four addresses each.

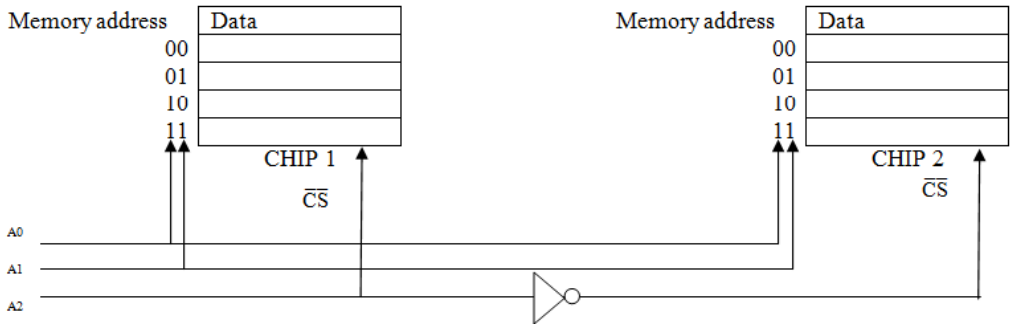
Memory address	Data
00	
01	
10	
11	

CHIP 1

Memory address	Data
00	
01	
10	
11	

CHIP 2

In the above architecture, we need three address lines A_0,A_1 and A_2. Line A_0 and A_1 are used to select one of the four addresses from 00,01,10 and 11 and line A_2 is used to select between the chip 1 and chip 2.



In case there are multiple chips, a decoder like 74LS138, known as address decoder are used. This can be illustrated using a following example. Suppose there are 4 memory chips each with 1024 addresses. The overall picture of can be summarized using the diagram below:

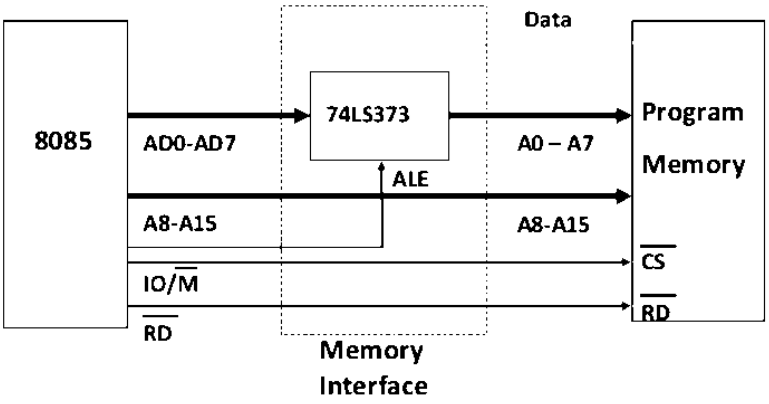


Figure 60: Memory interfacing

There are two types of address decoding techniques in 8085:

1. **Exhaustive/ Absolute Decoding:** It is an address decoding technique when all the higher order address lines remaining after completely generating the full address range of the memory are used for generating Chip Select (CS) signal. For example, If we a have a memory of size 4K(2^{12}=4096), 12 address lines out of 16 lines are required for addressing and the remaining 4 lines are used for generating Chip Select signals.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	0	1	1	0	1	1	0	0	1	0	1	0	1	1	1
Chip Select				Address Decoding											

2. **Partial Decoding:** It is an address decoding technique in which only some of the address lines of microprocessor left after using the required signals for memory are used for generating Chip Select (CS) signal. If total memory space is not required for the system then, this type of address decoding can be used. For example, If we a have a memory of size 512 (2^9=4096), 12 address lines out of 16 lines are required for addressing and the remaining 4 lines are used for generating Chip Select signals. The advantage of this technique is fewer components are required for memory interfacing because of this board size reduces and in turn cost reduces.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	0	1	1	X	X	X	0	0	1	0	1	0	1	1	1
Chip Select				Don't Care			Address Decoding								

7.2.1 Memory Map

Microprocessor does there Major functions[13].

1. **Read from Memory** to internal register
2. **Write to Memory** from internal register
3. **Execute** using internal registers

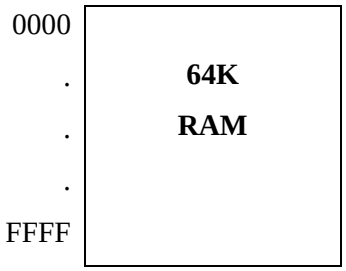
Microprocessor does not execute using external memory. It performs the execution using its internal registers. So Microprocessor performs two major activities externally. Read from Memory and Write to Memory are these two major activities. The 8085 Microprocessor has 16 address and 8 data lines. Therefore it can access up to 64 K locations because it has 16 address lines (2^{16} = 64K). Since this microprocessor has 8 data lines each location contains 8 bits (1 byte). Now let us see the address range of 8085 Microprocessor:

Address Lines	**A_{15}**	**A_{14}**	**A_{13}**	**A_{12}**	**A_{11}**	**A_{10}**	**A_9**	**A_8**	**A_7**	**A_6**	**A_5**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**
Starting Address	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ending Address	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

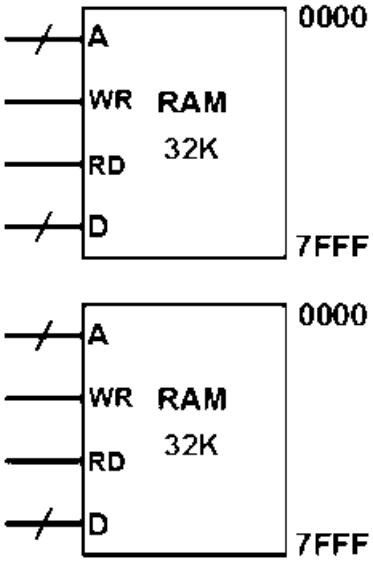
Since representing in binary is quite laborious, we use Hexa-Decimal Representation which is nothing but a short form of Binary. So the microprocessor which has 16 address lines can address 0000 - FFFF locations. The same way the 64K memory has 16 address lines has locations 0000 - FFFF. Since both are identical and made for each other. Every address location addressed by Microprocessor has corresponding location in Memory. Every address of Microprocessor is mapped to every location of Memory (0000 -> 0000, 0001 -> 0001, 0002 -> 0002, FFFE -> FFFE, FFFF -> FFFF). This is the simplest Memory Mapping. 1 Microprocessor, 1 Memory and Microprocessor to memory are directly connected.

Here the Memory Map of Microprocessor. Let us assume the Memory is RAM.

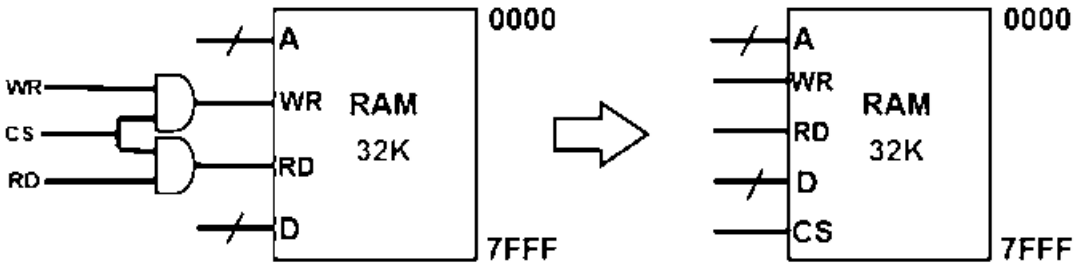
[13] Adapted from https://www.quora.com/What-is-Memory-Mapping-in-Microprocessor-based-systems



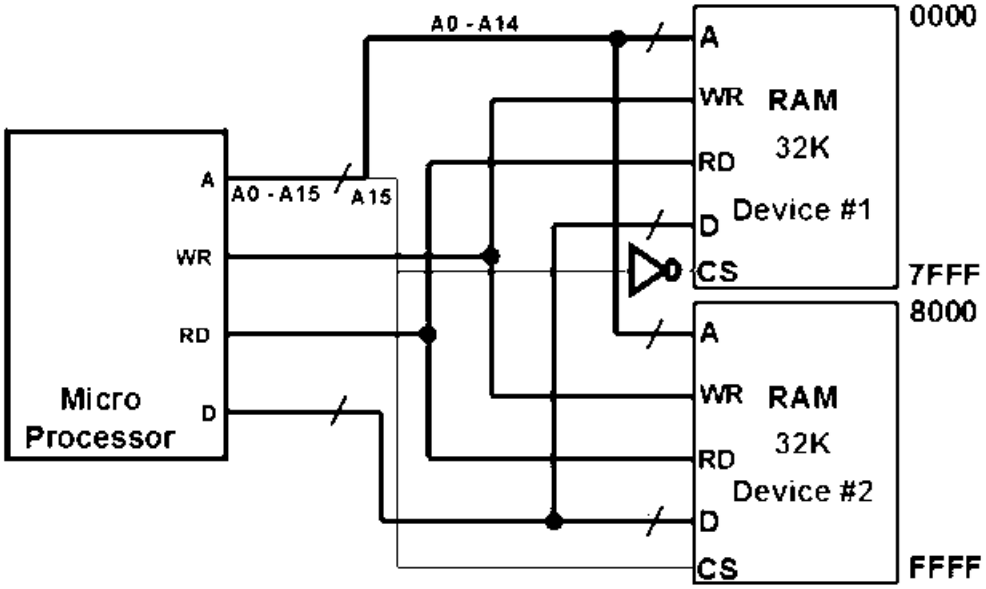
The above memory map is very simple because there is only one device and that is also have the maximum size addressable by Microprocessor. Now let us consider how to connect two devices of 32K size.



Since we have multiple device, we have introduce a new signal to select the Chip or we can call that signal as Chip Select. Now let see how to introduce the Chip Select in the above 32K device.



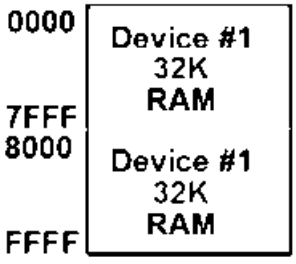
Now let us connect two 32K memories with the microprocessor and see how CS signal is used.



In the above diagram Microprocessor gives 16 Address line A0 - A15. But the memories have only 15 Address lines A0 - A14. So the 15 Address lines A0 - A14 are connected to the Memories and the address line A15 is used to select the Chip. If A15 is 0, Device #1 is selected and if A15 is 1, Device #2 is selected. Now let see how to create a Memory map for this circuit. Individually both the memories are 32K. So their address Range is 0000 – 7FFF (000 0000 0000 0000 - 111 1111 1111 1111). But when both the memories are placed inside the circuit, the address line A_{15} decides which Device has to be selected. Since A_{15} = 0 selects Device #1 and A_{15} = 1 selects Device #1, here is the address Range

Device #1 - **0**000 0000 0000 0000 - 0000 to **0**111 1111 1111 1111 = 0000 - 7FFF

Device #2 - **1**000 0000 0000 0000 - 8000 to **1**111 1111 1111 1111 = 8000 - FFFF



So Memory Map in a Microprocessor based is system is nothing but the address range (Low - High) of each device within the available address space. For example in the above design we have two devices. The available address space is 64K, because the microprocessor has 16 address lines (2^{16}). Device #1's address range is 0000 - 7FFF and Device #2's address range is 8000 - FFFF.

If the microprocessor is have 16 address bits then we can connect the devices with the following memory sizes.

$64K - 2^{16}$: $32K - 2^{15}$: $16K - 2^{14}$: $8K - 2^{13}$

$4K - 2^{12}$: $2K - 2^{11}$: $1K - 2^{10}$: $512 - 2^{9}$

$256 - 2^{8}$: $128 - 2^{7}$: $64 - 2^{6}$: $32 - 2^{5}$

$16 - 2^{4}$: $8 - 2^{3}$: $4 - 2^{2}$: $2 - 2^{1}$

$1 - 2^{0}$

So we have devices with 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1K, 2K, 4K, 8K, 16K, 32K and 64K size are used with Microprocessor which has 16 address lines. We can have any number of devices connected with the Microprocessor. But we have to follow some rules while designing. Let us try to understand the rules by solving a simple problem.

Rule No 1: The sum of all given devices memory sizes should be less than the maximum memory addressable by the microprocessor. For example:

Maximum addressable memory by microprocessor = 2^{16} = 64k

Sum of all given memories = 32K + 16K + 8K + 4K + 2K = 62K

Since 62K < 64K, so this connection is possible.

But 32k,16k,16k,8k memories cannot be connected because sum of given memory sizes(32K + 16K + 16K + 8K) is 72k which is more than memory addressable by microprocessor(64K).

Rule No 2: If Rule No 1 is OK then move to Rule No 2. If range of all memories is Descending order in the number. Like

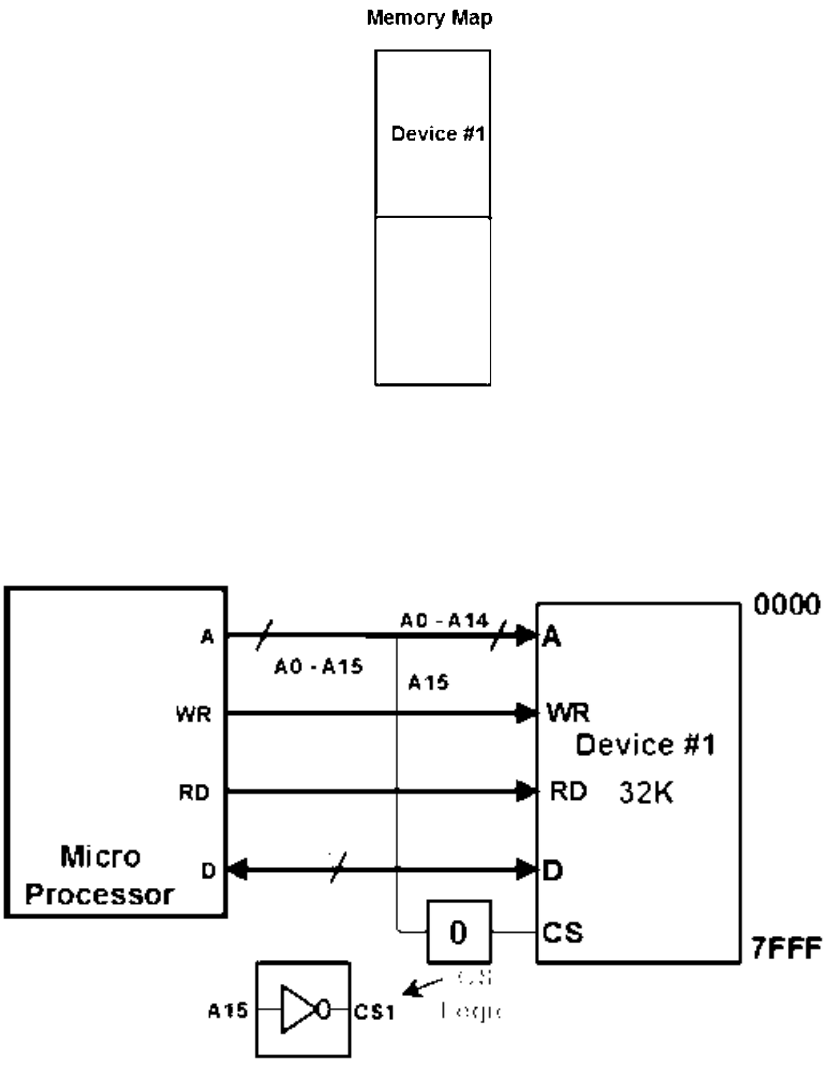
32K, 16K, 8K, 4K, 2K

Rule No 3: Connect first device to last device one by one with microprocessor

Now let us perform some exercise to further elaborate the concept.

Exercise: Connect the following devices of size 32k,16k,8k,4k and 2k with microprocessor whose address lines are 16.

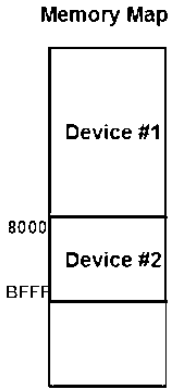
Step 1: First let us connect the Device #1 (32K) with Microprocessor. Since this device has 32K memory, it requires 15 address lines. Microprocessor gives 16 address lines. The A0-A14 (15 lines) are connected to the Device and Address line A15 is used to control the Chip Select. In this case we have assumed that if A15 is 0 Device 1 will be selected.

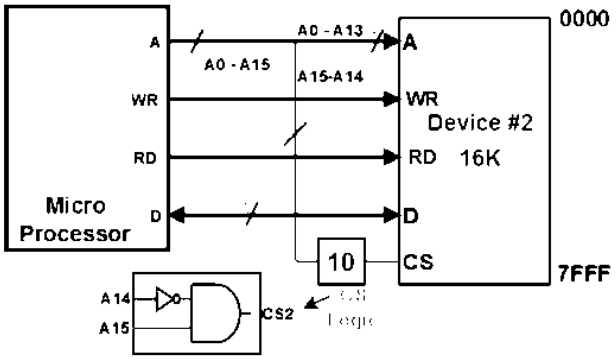


Here is how the starting and ending address of device #1 is calculated.

Address Lines	**A_{15}**	**A_{14}**	**A_{13}**	**A_{12}**	**A_{11}**	**A_{10}**	**A_9**	**A_8**	**A_7**	**A_6**	**A_5**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**	
Starting Address	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0000**
Ending Address	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**7FFF**

Step 2: Connect the (16K) with Microprocessor whose address lines are 16. Since this device has 16K memory, it requires 14 address lines. Microprocessor gives 16 address lines. The A0-A13 (14 lines) are connected to the Device and Address line A15 & A14 are used to control the Chip Select. Since A15 = 0 is used by the device #1 we can use A15 = 1. But A14 can be 0 or 1. We assume A14 is 0. So when A15 is 1 and A14 is 0, Device 2 will be selected.

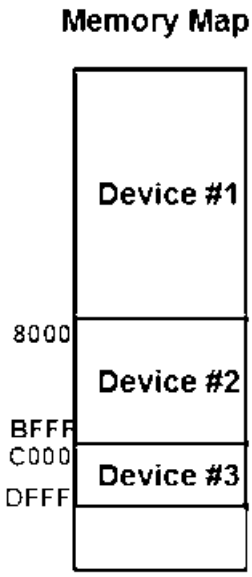


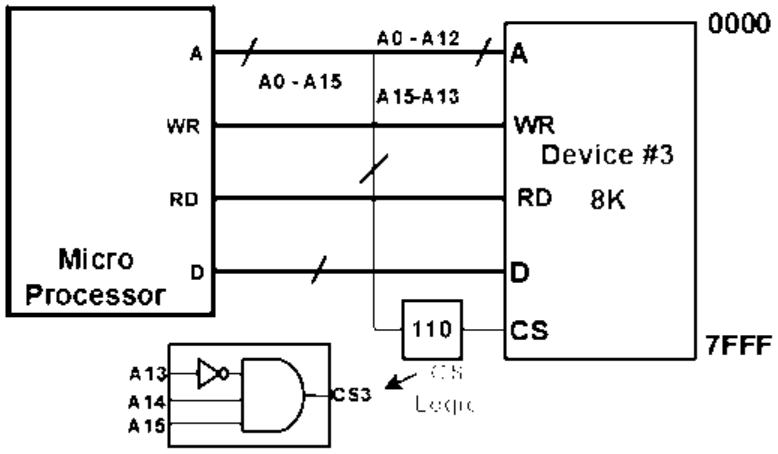


Here is how the starting and ending address of device #2 is calculated.

Address Lines	**A_{15}**	**A_{14}**	**A_{13}**	**A_{12}**	**A_{11}**	**A_{10}**	**A_9**	**A_8**	**A_7**	**A_6**	**A_5**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**	
Starting Address	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**8000**
Ending Address	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**BFFF**

Step 3: Let us connect the Device #3 (8K) with Microprocessor. Since this device has 8K memory, it requires 13 address lines. Microprocessor gives 16 address lines. The A0-A12 (13 lines) are connected to the Device and Address line A15, A14 & A13 are used to control the Chip Select. Since A15 = 0 is used by the device #1 we can use A15 = 1. A15 = 1 & A14 =0 are used by the device #2, so we can use A14 = 1. A13 can be 0 or 1. We assume A13 is 0. So when A15 is 1, A14 is 1 & A13 is 0, Device #3 will be selected.

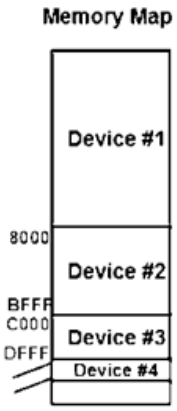


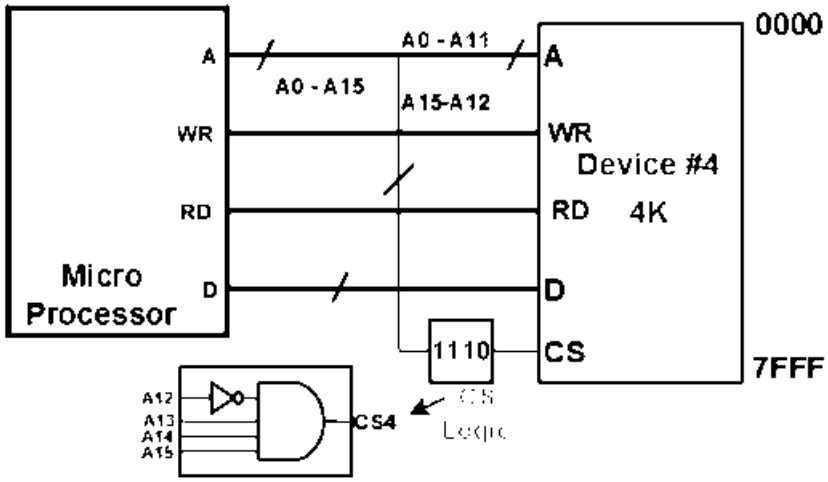


Here is how the starting and ending address of device #3 is calculated.

Address Lines	**A_{15}**	**A_{14}**	**A_{13}**	**A_{12}**	**A_{11}**	**A_{10}**	**A_9**	**A_8**	**A_7**	**A_6**	**A_5**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**	
Starting Address	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**C000**
Ending Address	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	**DFFF**

***Step* 4:** Let us connect the Device #4 (4K) with Microprocessor. Since this device has 4K memory, it requires 12 address lines. Microprocessor gives 16 address lines. The A0-A11 (12 lines) are connected to the Device and Address line A15, A14, A13 & A12 are used to control the Chip Select. Since A15 = 0 is used by the device #1 we can use A15 = 1. A15 = 1 & A14 =1 are used by the device #2, so we can use A14 = 1. A15 = 1 , A14 =1 and A13 = 0 are used by the device #3, so we can use A13 = 1. A12 can be 0 or 1. We assume A12 is 0. So when A15 is 1, A14 is 1, A13 is 1 & A12 is 0, Device #4 will be selected.



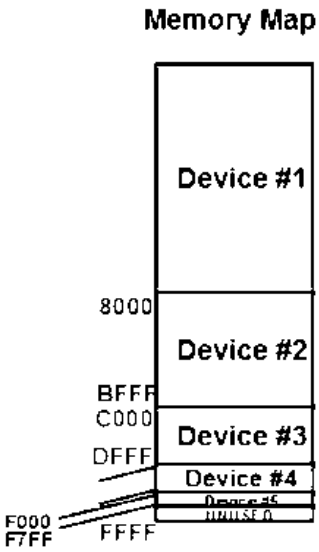


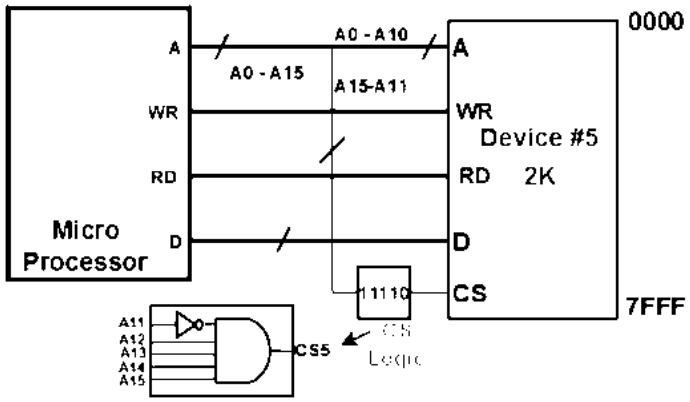
Here is how the starting and ending address of device #4 is calculated.

Address Lines	**A_{15}**	**A_{14}**	**A_{13}**	**A_{12}**	**A_{11}**	**A_{10}**	**A_9**	**A_8**	**A_7**	**A_6**	**A_5**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**	
Starting Address	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**E000**
Ending Address	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	**EFFF**

Step 5: Let us connect the Device #5 (2K) with Microprocessor. Since this device has 2K memory, it requires 11 address lines. Microprocessor gives 16 address lines. The A0-A10 (11 lines) are connected to the Device and Address line A15, A14, A13,A12 & A11 are used to

control the Chip Select. Since A15 = 0 is used by the device #1 we can use A15 = 1. A15 = 1 & A14 =1 are used by the device #2, so we can use A14 = 1. A15 = 1 , A14 =1 and A13 = 0 are used by the device #3, so we can use A13 = 1. A15 = 1 , A14 =1 A13 = 1 and A12 = 0 are used by the device #4, so we can use A12 = 1. A11 can be 0 or 1. We assume A12 is 0. So when A15 is 1, A14 is 1, A13 is 1, A12 is 1 & A11 is 0, Device #5 will be selected.



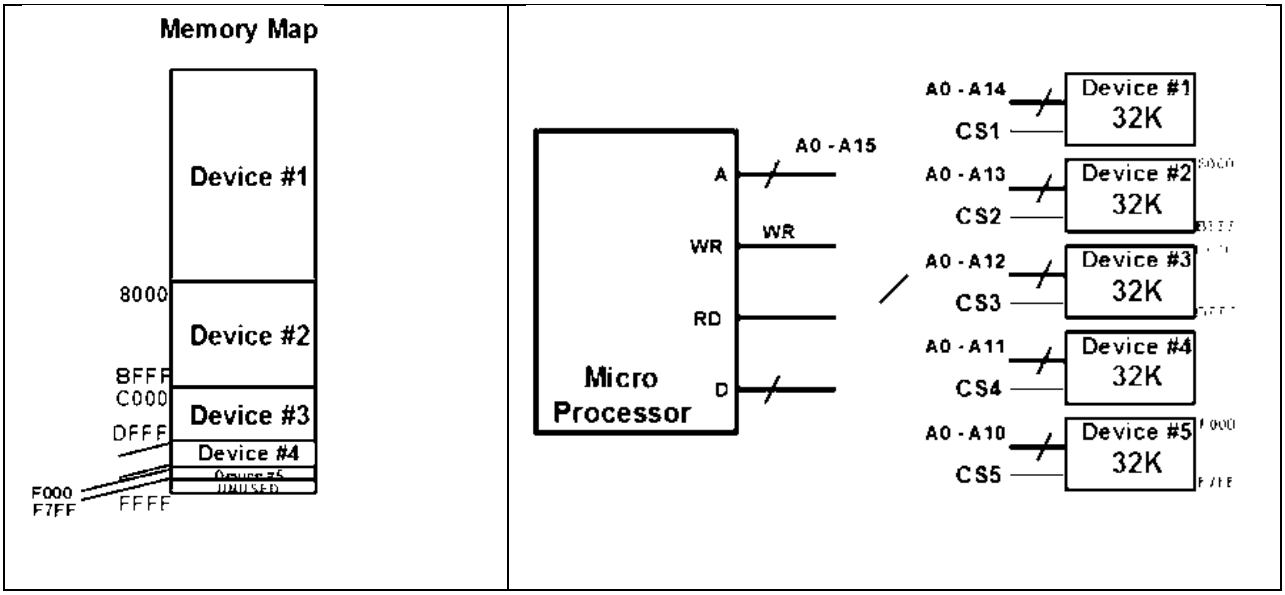


Here is how the starting and ending address of device #5 is calculated.

Address	**A_{15}**	**A_{14}**	**A_{13}**	**A_{12}**	**A_{11}**	**A_{10}**	**A_9**	**A_8**	**A_7**	**A_6**	**A_5**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**

Lines																	
Starting Address	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	**F000**
Ending Address	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	**F7FF**

Here is the consolidated Circuit with all the devices. I have not mentioned the Chip Select logic, Control & Data Lines.



7.3 I/O Interfacing

We know that keyboard and Displays are used as communication channel with outside world. So it is necessary that we interface keyboard and displays with the microprocessor. This is called I/O interfacing. In this type of interfacing we use latches and buffers for interfacing the keyboards and displays with the microprocessor. Interfacing with I/O devices can be done in two ways:

- Parallel I/O: In this case entire group of 8-bits of data bus is used for the communication.
- Serial I/O: In this method one bit is transferred at a time using SID or SOD pins on the microprocessor.

There are two techniques through which devices can be interfaced in parallel data transfer mode to microprocessor.

1. Memory mapped I/O
2. Peripheral mapped I/O or I/O mapped I/O

7.3.1 Memory mapped I/O

I/O devices are interfaced using address from memory space. That means IO device address are part of addresses given to memory locations.8085 uses 16-bit address to memory interfacing. So any address between 0000H-FFFFH can be given to each peripheral. But the addresses given to peripheral can't be used for memory.

Memory control signals are used as read and write control signals for peripherals. And all the operations that can be performed on memory can also be performed on peripherals. No need of using IO instructions such as IN, OUT.

7.3.2 I/O mapped I/O

In this method separate address space is given to IO devices. Each IO device is given a 8-bit address. Hence maximum 256 devices can be interfaced to the processor. The address range for the IO devices is 00H-FFH. IO control signals are used to perform read, write operations.

For reading data from IO device or writing data to IO device IN, OUT instructions needs to be used. Arithmetic and logical operations can't be performed directly on IO devices as in memory mapped IO.

IO devices can be interfaced, by using buffers for simple IO i.e. by using address decoding circuit to enable buffer. For handshake IO or to interface more peripherals ICs like 8255 peripheral programmable interface (PPI) can be used.

7.3.3 Comparison of memory mapped I/O and I/O mapped I/O

Memory Mapped I/O	I/O Mapped I/O
I/O is treated as Memory	I/O is treated as I/O
16-bit addressing	8-bit addressing
More decoder hardware is required	Less decoder hardware is required

Can address 2^{16}=64K locations	Can address 2^{8}=256 locations
Less memory is available	Whole memory range is available
Memory instructions are used for both memory and I/O related operations	Special instructions line IN, OUT etc. are used
Memory control signals are used	Special control signals are used
Arthimetic and logic operations can be performed on data	Arthimetic and logic operations cannot be performed on data
Data transfer is possible between register and I/O	Data transfer is only possible between accumulator and I/O

7.4 SUMMARY

1. The entire group of instructions, determines what functions the microprocessor can perform is called instruction set.
2. A program is a sequence of instructions written to tell the computer to perform a specific function.
3. A latch is necessary to hold the output data for display. The input data byte is obtained by enabling a tri-state buffer and placed in the accumulator.
4. We know that keyboard and Displays are used as communication channel with outside world. So it is necessary that we interface keyboard and displays with the microprocessor. This is called I/O interfacing.

7.5 Check Your Progress

1. The operation to be performed is called _____ .
2. The data to be operated is called _____.
3. The _____ instructions copy data from one source in to a destination without modifying the content of the source.
4. Which is used to store critical pieces of data during subroutines and interrupts:

a. Stack
b. Queue
c. Accumulator
d. Data register

5. The point where control returns after a subprogram is completed is known as the :
   a. Return address
   b. Main Address
   c. Program Address
   d. Current Address
6. The subprogram finish the return instruction recovers the return address from the:
   a. Queue
   b. Stack
   c. Program counter
   d. Pointer
7. MAR stands for:
   a. Memory address register
   b. Memory address recode
   c. Micro address register
   d. None of these
8. The standard I/O is also called:
   a. Isolated I/O
   b. Parallel I/O
   c. both a and b
   d. none of these

7.6 Answers to Check Your Progress

1. Opcode
2. Operand
3. Data transfer
4. A
5. A
6. B
7. A
8. A

7.7 Model Questions

1. Give the functional categories of 8085 micro instructions?
2. Define the types of branching operations.

3. What is STA in data transfer instruction?
4. Why the number of out ports in the peripheral-mapped I/O is restricted to 256 ports?
5. Define Memory mapped I/O?
6. What is an interrupt I/O?
7. What is Partial Decoding?
8. Define absolute decoding.
9. Compare memory mapped I/O with I/O mapped I/O.

INTERRUPTS

8.0 Learning Outcomes

After going through this chapter, you will be able to:

- Define an Interrupt
- Classify maskable and non-maskable interrupts
- Differentiate software and hardware interrupts
- Perform operation using SIM and RIM instruction
- Know the vector address of the software interrupts.
- Understand the priority of the different interrupts

8.1 Introduction

Interrupt is a mechanism by which an I/O or an instruction can suspend the normal execution of processor and get itself serviced. Generally, a particular task is assigned to that interrupt signal. In the microprocessor based system the interrupts are used for data transfer between the peripheral devices and the microprocessor.

As soon as the microprocessor receives an interrupts signal, it suspends the currently executing program and jumps to the Interrupt Service Routine(ISR) to respond to the incoming interrupt. ISR is a small program or a routine that when executed services the corresponding interrupting source and each interrupt have its own ISR. This can be explained using the figure below:

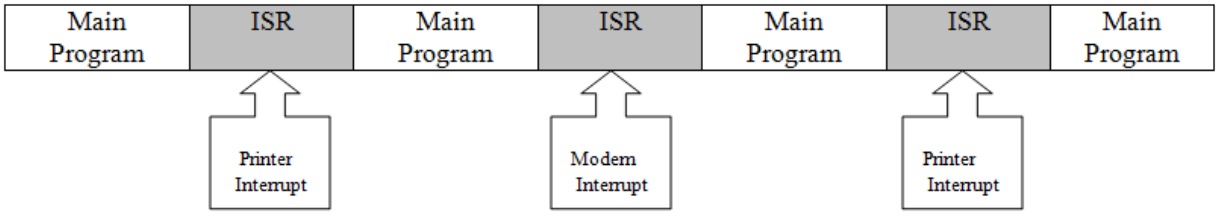


Figure 61:Execution of interrupt in between a main program

The interrupt processing flow could be explained using the flow chart below. Initially, before the interrupt is initiated by an attached device, the Microprocessor is executing the Main program.

As soon as the Microprocessor received an interrupt request, it checks whether the device generating the interrupt has the greater priority than the current task it is executing. If yes, then the Microprocessor accepts the Interrupt and it completes the execution of the current instruction. Then it attempts to get the Interrupt vector or the address of the ISR and save the content of Program Counter(PC) in the stack so that it can come back to the Main Program once the ISR is serviced.. Then it jumps to the starting location of ISR and start executing it. Once the ISR is serviced, the Microprocessor jump bank to the Main Program using the address that it stored in the Stack.

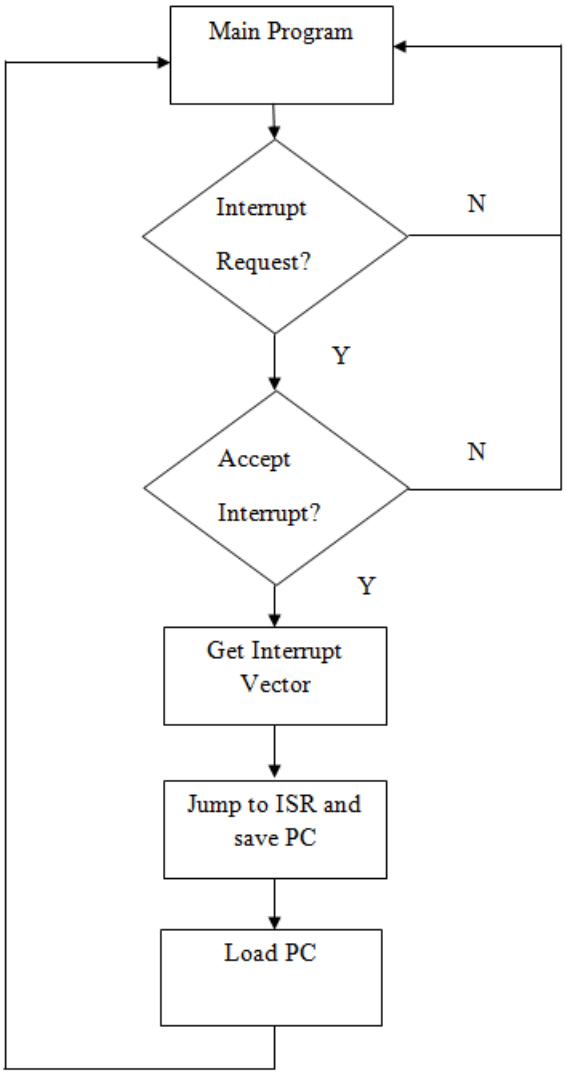


Figure 62: Flow chart of an interrupt

8.2 Types of Interrupts

8.2.1 Vector Interrupt

In this type of interrupt, Processor knows the address of Interrupt. In other word processor knows the address of interrupt service routine. The example of vector interrupt are RST 7.5, RST 6.5, RST 5.5, TRAP.

8.2.2 Non-Vector Interrupt

In this type of interrupt, Processor cannot know the address of Interrupt. It should give externally. In the device will have to send the address of interrupt service routine to processor for performing Interrupt. The example of Non-vector interrupt is INTR.

8.2.3 Maskable interrupts

An interrupt which can be disabled by software that means we can disable the interrupt by sending appropriate instruction, is called a maskable interrupt. RST 7.5, RST 6.5, RST 5.5 are the example of Maskable Interrupt.

8.2.4 Non-Maskable interrupts

As the name suggests we cannot disable the interrupt by sending any instruction is called Non Maskable Interrupt. TRAP interrupt is the non-maskable interrupt for 8085. It means that if an interrupt comes via TRAP, 8085 will have to recognize the interrupt we cannot mask it.

8.2.5 Software Interrupt

It is an instruction based Interrupt which is completely control by software. That means programmer can use this instruction to execute interrupt in main program. There are eight software interrupt available in 8085 microprocessor. See the example with their hex code and vector address.

Instruction	Corresponding HEX code	Vector addresses
RST 0	C7	0000H
RST 1	CF	0008H
RST 2	D7	0010H
RST 3	DF	0018H
RST 4	E7	0020H
RST 5	EF	0028H
RST 6	F7	0030H
RST 7	FF	0038H

8.2.6 Hardware Interrupt

As name suggest it is interrupt which can get the interrupt request in hardware pin of microprocessor 8085. There is mainly six pin is available for dedicatedly for interrupt purpose. Those are

- TRAP
- RST 5.5
- RST 6.5
- RST 7.5
- INTR
- INTA (It is not an Interrupt pin but it is used to send acknowledgement of the Interrupt request getting from other interrupt pin.)

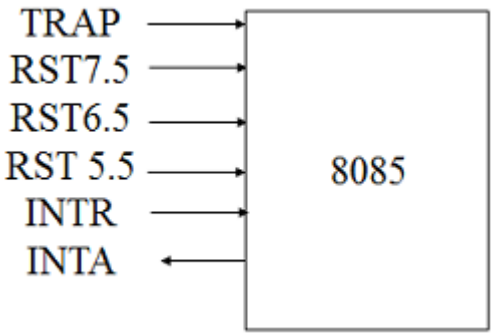


Figure 63: Different types of hardware interrupts

Any device attached to the microprocessor based system can raise an interrupt signal to seek the attention of the microprocessor. The I/O devices can be interfaced with the microprocessor using two possible configurations to handle the interrupts.

1. A microprocessor may have only one interrupt level and many I/O devices are connected to it. In this case a special device called an Interrupt Controller(Intel 8259), which supports 8 I/O devices at a time, is connected between the microprocessor and I/O devices to take care of the interrupts generated by different devices.
2. A microprocessor may have several interrupt levels and one I/O device is to be connected to each interrupt level as the number of interrupts levels are greater than or equal to the number of I/O devices attached to the microprocessor. Whenever an interrupt is generated by a device, the microprocessor knows the address of the Interrupt Service Routine in advance. All it needs is that the interrupting device sends its unique vector via a data bus and through its I/O interface to the Microprocessor. The Microprocessor takes this vector, checks the interrupt table in memory, and then carries out the correct ISR for that device. This type of interrupt scheme is known as vectored interrupt.

8.3 Interrupts in 8085 Microprocessor

8085 has 5 interrupts viz. TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. The priority of the interrupts is as follows:

TRAP > RST 7.5 > RST 6.5 > RST 5.5 > INTR

The detailed description of the interrupts is as follows:

1. TRAP: It is non-maskable edge and level triggered interrupt. TRAP has the highest priority and vectored interrupt. Edge and level triggered means that the TRAP must go high and remain high until it is acknowledged. In case of sudden power failure, it executes an ISR and sends the data from main memory to backup memory. As we know that TRAP cannot be masked but it can be delayed using HOLD signal. This interrupt transfers the microprocessor's control to location 0024H. TRAP interrupts can only be masked by resetting the microprocessor. There is no other way to mask it.
2. RST7.5: It has the second highest priority. It is maskable and edge level triggered interrupt. The vector address of this interrupt is 003CH. Edge sensitive means input goes high and no need to maintain high state until it is recognized. It can also be reset or masked by resetting microprocessor. It can also be resetted by DI instruction. EI is a one byte instruction used to enable non-maskable interrupts and DI is a one byte instruction used to disable non-maskable interrupts.
3. RST6.5: This is a level triggered and maskable interrupts. When RST6.5 pin is at logic 1, INTE flip-flop is set. RST 6.5 has third highest priority. It can be masked by giving DI and SIM instructions or by resetting microprocessor.

4. RST5.5: This is also a level triggered and maskable interrupts and has fourth highest priority. It can be masked by giving DI and SIM instructions or by resetting microprocessor.
5. INTR: It is level triggered and maskable interrupt. The following sequence of events occurs when INTR signal goes high:
    a. The 8085 checks the status of INTR signal during execution of each instruction.
    b. If INTR signal is high, then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled.
    c. On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction.

    It has the lowest priority. It can be disabled by resetting the microprocessor or by DI and SIM instruction.

8.4 The 8085 Vectored/Maskable Interrupts

Intel 8085 has three masked/Vectored interrupts viz. RST 7.5, RST 6.5 & RST 5.5 and they are automatically vectored according to the following table:

Interrupt	Vector
RST 5.5	002CH
RST 6.5	0034H
RST 7.5	003CH

These interrupts are masked at two levels:

- Through the Interrupt Enable flip-flop and EI/DI instructions.
- Through individual mask flip-flop that control the availability of the individual interrupts.

The Interrupt Enable(IE) and Disable Interrupt(DI) instructions are the instructions through which authorize the Microprocessor to accept or reject and Interrupt. As soon as the EI instruction is encountered by the Microprocessor, it saves the address of the next instruction in the stack and starts executing the instruction followed by EI instruction. This allows at least one more instruction like JMP or RET to be executed before the microprocessor allows itself to be again interrupted. In case of DI, the interrupts are disabled immediately and no flags are affected. The figure below explains the schematic diagram of interrupts:

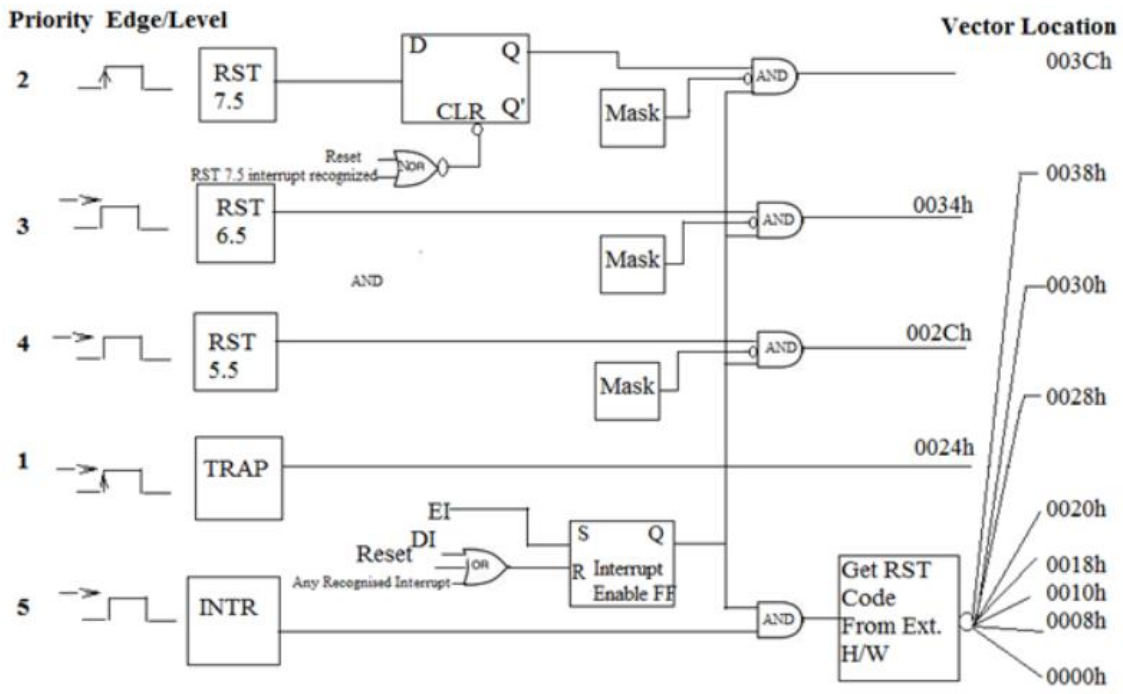


Figure 64: Schematic diagram of interrupts

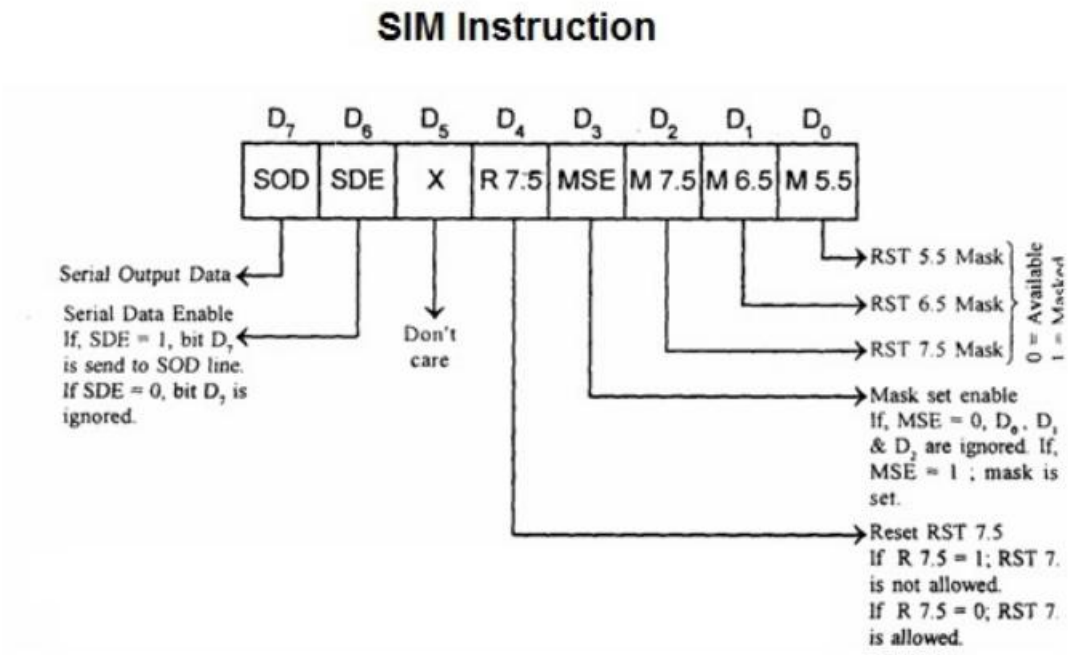
1. The interrupt process should be enabled using the EI instruction.
2. The 8085 checks for an interrupt during the execution of every instruction.
3. If there is an interrupt, and if the interrupt is enabled using the interrupt mask, the microprocessor will complete the executing instruction, and reset the interrupt flip-flop.
4. The microprocessor then executes a call instruction that sends the execution to the appropriate location of the interrupt vector.
5. When the microprocessor executes the call instruction, it saves the address of the next instruction on the stack.
6. The microprocessor jumps to the specific service routine.
7. The service routine must include the instruction EI to re-enable the interrupt process.
8. At the end of the service routine, the RET instruction returns the execution to where the program was interrupted.

The RIM(READ INTERRUPT MASK) and SIM(SET INTERRUPT MASK) instructions are used by the 8085 Microprocessor to provide the interrupt services with the help of Serial Input Data(SID) and Serial Output Data(SOD) pins on the device.

8.4.1 Manipulating the mask

The interrupt enable flip flop is mask is manipulated using EI/ DI instructions and the individual mask for RST 7.5, RST 6.5 and RST 5.5 are manipulated using SIM(SET INTERRUPT MASK)

instruction, which takes the bit pattern in the Accumulator register and applies it to the interrupt mask enabling and disabling the specific interrupt.

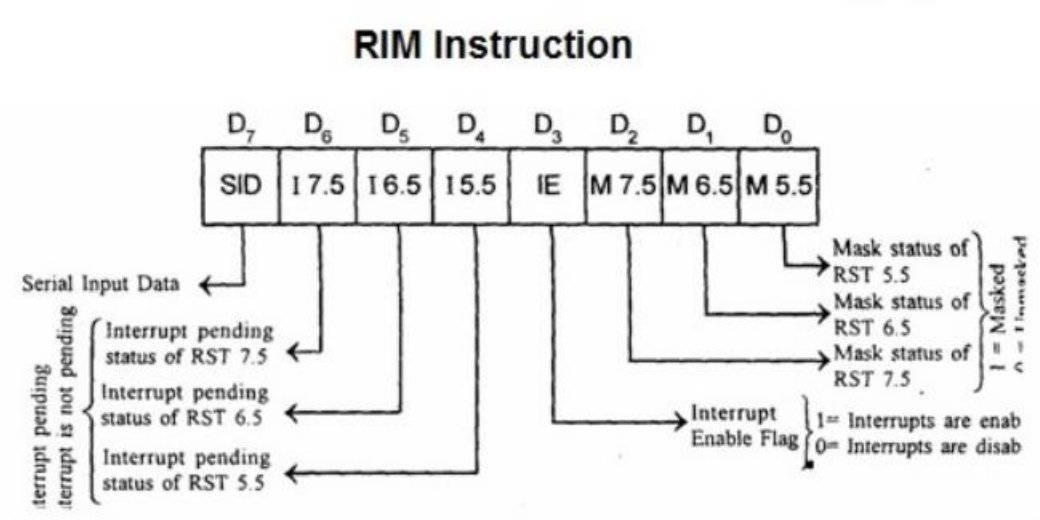


The SIM instruction uses the data in the accumulator as follows:

- D7-D6 - The left two bits are related to the serial interface. When D6 (SDE-Serial Data Enable) is 1, then whatever is in D7 (SOD-Serial Data Output) is written to the serial data output (pin 4 of the 8085). If D6=0, nothing is written. This allows a SIM instruction to be executed altering interrupt masks without affecting serial data.
- Bit D5 is not used.
- Bit D4 (R 7.5-Reset RST 7.5) This bit allows the SIM instruction to reset the interrupt pending flag indicated by bit D6 in the RIM instruction layout. The 7.5 interrupt can indicate that it is pending via the RIM instruction even though it is masked off. This bit allows that pending request to be reset.
- Bit D3 (MSE-Mask Set Enable) is like SDE -- it indicates whether the lower three bits (D2-D0) are ignored or not. This allows the serial data output to occur without affecting the interrupt masks. If a SIM is executed with this bit low, the condition of the mask bits will not change. If a SIM is executed with this bit set high, the mask bits will be set according to the lower three bits of the accumulator.
- Bits D2-D0 (RST 7.5 Mask, RST .5 Mask, RST 5.5 Mask) These are the interrupt masks for the 8085 interrupts 7.5, 6.5, and 5.5. If the corresponding bit is 0, the interrupt is enabled. If the bit is 1, the interrupt is masked (ignored).

8.4.2 Determining the current mask settings

The RIM(READ INTERRUPT MASK) instruction permits the system to examine the interrupt mask by loading into the Accumulator register a byte which defines the condition of the mask bits for the maskable interrupts, the condition of the interrupts pending for the maskable interrupts, the condition of the Interrupt Enable flag, and the condition of the Serial Input Data (SID) pin on the Microprocessor. The RIM instruction reads the following bits into the accumulator:



- Bit D7 (SID-Serial Input Data) This is the input pin of the serial data interface which is connected to pin 5 of the 8085, and indicates the high/low status of that pin.
- Bits D6-D4 (I 7.5, I 6.5, I 5.5) These bits indicates that an interrupt is pending for these three 8085 interrupts 7.5, 6.5, and 5.5. If interrupts 5.5 or 6.5 have been masked off by bits D0 or D1, bits D4 and D5 will not be set. Bit D6, which corresponds to the 7.5 interrupt, will be set on to indicate that an interrupt 7.5 was requested, even if it was masked off.
- Bit D3 (IE-Interrupt Enable) This bit indicates whether interrupts are enabled (1) using the EI (Enable Interrupts) instruction, or disabled (0) using the DI (Disable Interrupts) instruction.
- Bits 2-D0 (M 7.5, M6.5, M5.5) Mask status of interrupts 7.5, 6.5, and 5.5. Corresponds to bits D2-D0 of the SIM instruction. 1 if masked, 0 if enabled.

8.5 The 8085 Non-Vectored Interrupts

The non-vectored interrupts is processed by 8085 Microprocessor as follows:

1. The interrupt process should be enabled using the EI instruction.
2. The 8085 Microprocessor checks for the interrupts during the execution of every instruction.
3. If INTR is high, Microprocessor completes the execution of the current instruction. Disable the interrupt and sends INTA(INTerrupt Acknowledgement) signal to the device from where the interrupt signal was generated.
4. INTA allows the I/O device to send a RST instruction through the data bus.
5. As soon as the Microprocessor receives the INTA signal, it saves the address of the next instruction received from the Program Counter to the Stack and the RST instruction specifies the starting address of the "CALL" subroutine (IST Cell).
6. The microprocessor performs the ISR. ISR must include the "EI" instruction to enable further interrupt within the program.
7. RET instruction at the end of the ISR allows the Microprocessor to retrieve the return address from the stack and the program control is again transferred to the program which was interrupted in between.

The 8085 recognizes 8 restart instructions. These are RST0, RST1, RST2, RST3, RST4, RST5, RST6, RST7. Whenever the restart instruction is initiated, each of these would send the execution to a predetermined memory location specified below:

Restart Instruction	Equivalent To
RST0	CALL 000H
RST1	CALL 0008H
RST2	CALL 0010H
RST3	CALL 0018H
RST4	CALL 0020H
RST5	CALL 0028H
RST6	CALL 0030H
RST7	CALL 0038H

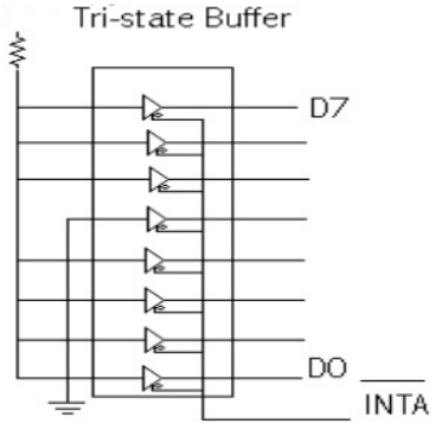
The Restart sequence is made up of three machine cycles:

Machine Cycle 1: In the 1st machine cycle the microprocessor sends the INTA signal and while INTA is active, the microprocessor reads the data lines expecting to receive, from the interrupting device, the Opcode for the specific RST signal.

Machine Cycle 2 and 3: In order to return back to the interrupted program which the Microprocessor is currently executing, the 16-bit address of the next instruction is saved on the stack and the Microprocessor jumps to the address reserved for the RST instruction.

One of the important question that may come to the mind of the learners is how does the external device produce the Opcode for the appropriate RST instruction? To know how it works we have to first understand what Opcode(Operation Code) is. Opcode is simply a collection of bits. So, the device needs to set the bits of the data bus to the appropriate value in response to the INTA signal. This can easily be implemented through a tri-state buffer as shown in figure below which explains the generation of RST5 Opcode, which is EF.

	$D_7D_6D_5D_4D_3D_2D_1D_0$
Opcode of RST5 EF	1 1 1 0 1 1 1 1



During the Interrupt Acknowledge machine cycle(1st Machine Cycle of RST operation), the microprocessor activates the INTA signal which enables the Tri-state buffers and place the value of EF_H on the data bus. Therefore, sending the RST5 signal to the microprocessor. The RST5 instruction is exactly equivalent to CALL 0028H

8.6 Issues in Implementing INTR Interrupts

Before going through the discussion, there are two important terms that the learners should know:

1. **Interrupt Latency:** The time interval from when the interrupt is first asserted to the time the CPU recognizes it. This will depend much upon whether interrupts are disabled, prioritized and what the processor is currently executing. At times, a processor might ignore requests whilst executing some indivisible instruction stream (read-write-modify cycle). The figure that matters most is the longest possible interrupt latency time.
2. **Interrupt Response Time:** The time interval between the CPU recognizing the interrupt to the time when the first instruction of the interrupt service routine is executed. This is determined by the processor architecture and clock speed.
3. **Interrupt priority:** In systems with more than one interrupt inputs, some interrupts have a higher priority than other. They are serviced first if multiple interrupts are triggered simultaneously.
4. **Interrupt vector:** Code loaded on the bus by the interrupting device that contains the Address (segment and offset) of specific interrupt service routine
5. **Interrupt Masking:** Ignoring (disabling) an interrupt

Now the next big question is, how long must INTR signal remain **high**? As we have already discussed above that the microprocessor checks the INTR line one clock cycle before the last T-state of each instruction. The interrupt process is asynchronous and INTR must remain active long enough to allow for the longest instruction, the conditional CALL instruction which requires 18 T-states. Therefore, the INTR must remain active for 17.5 T- states.

The INTR line must be deactivate before the EI is executed. Otherwise, the Microprocessor will be interrupted again. Once the Microprocessor starts to respond to an INTR interrupt, INTA must be active(=0). Therefore, INTR should be turned off as soon as INTA signal is received.

One more important question is: Can the Microprocessor be interrupted again before the completion of the ISR?

The answer to this question is Yes. As soon as the 1^{st} interrupt arrives, all the maskable interrupts are disabled. They will only be enabled after the execution of EI instruction. Therefore, to interrupt the Microprocessor before the completion of the ISR, the EI instruction should be placed early in the ISR, so that the other interrupts may occur before the ISR is done.

8.7 Handling Multiple Interrupts and Priorities

The microprocessor can only respond to one signal on INTR at a time. Therefore, we must allow the signal from only one of the device to reach the Microprocessor. We must assign some priority to the different devices attached to the Microprocessor and allow their signals to reach the microprocessor according to the priority.

The solution is to use a circuit called the priority encoder(74LS148). This circuit has 8-inputs and 3 outputs. The inputs are assigned increasing priorities according to the increasing index of the input i.e. Input 7 has the highest priority and input 0 has the lowest. The 3 outputs carry the index of the highest priority active input.

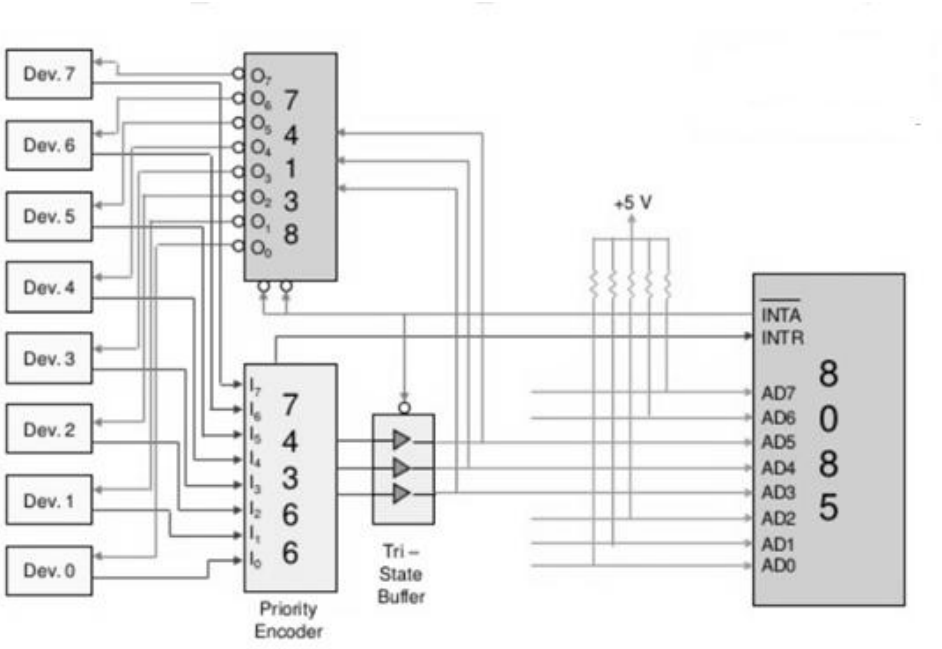


Figure 65: Priority encoder

Note that the opcode for the different RST instructions follows a set pattern. Bit D5,D4 and D3 of the opcode change in a binary sequence from RST 7 down to RST0. The one drawback to this scheme is that the only way to change the priority of the devices connected to the 74366 is to reconnect the hardware.

8.8 SUMMARY

1. Interrupt is a process where an external device can get the attention of the Microprocessor.
2. Interrupts may come and be acknowledged (provided masking of any interrupt is not done) by the microprocessor without any reference to the system clock. That is why interrupts are asynchronous in nature.
3. There are five (5) interrupt pins of 8085—from pin 6 to pin 10. They represent TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR interrupts respectively. These five interrupts are 'hardware' interrupts.
4. TRAP interrupt is the non-maskable interrupt for 8085. It means that if an interrupt comes via TRAP, 8085 will have to recognize the interrupt.
5. The interrupts of 8085 have their priorities fixed—TRAP interrupt has the highest priority, followed by RST 7.5, RST 6.5, RST 5.5 and lastly INTR.
6. TRAP interrupt is both positive edge and level triggered, RST 7.5 is positive edge triggered while RST 6.5, RST 5.5 and INTR are all level triggered.
7. The EI instruction sets the interrupt enable flip-flop, thereby enabling RST 7.5, RST 6.5, RST 5.5 and INTR interrupts.
8. The DI instruction resets the interrupt enable flip-flop, thereby disabling RST 7.5, RST 6.5, RST 5.5 and INTR interrupts.
9. The important properties of the Interrupts can be summarized below:

Interrupt	Maskable	Masking method	Vectored	Memory	Triggering method
INTR	Yes	DI/EI	No	No	Level sensitive
RST 5.5	Yes	DI/EI SIM	Yes	No	Level sensitive
RST 6.5	Yes	DI/EI SIM	Yes	No	Level sensitive
RST 7.5	Yes	DI/EI SIM	Yes	Yes	Edge sensitive
TRAP	No	None	Yes	No	Level and edge sensitive

10. Interrupts occurring within interrupts are called 'nested interrupts'. While handling nested interrupts, care must be taken to see that the stack does not grow to such an extent as to foul the main program—in that case the system program fails.
11. Multiple interrupts can be handled if a separate interrupt is allocated to each peripheral.

8.9 Check Your Progress

1. Which interrupt has the highest priority?
    a. INTR
    b. TRAP
    c. RST6.5
    d. All have the same priority
2. What is SIM?
    a. Select Interrupt Mask
    b. Sorting Interrupt Mask
    c. Set Interrupt Mask
    d. None of the above
3. Which one of the following is not a vectored interrupt?
    a. TRAP
    b. INTR
    c. RST 7.5
    d. RST 3
4. In 8085 microprocessor, the RST6 instruction transfer programme execution to following location :
    a. 0030H
    b. 0024H
    c. 0048H
    d. 0060H
5. The number of hardware interrupts that the processor 8085 consists of is:
    a. 1
    b. 3
    c. 5
    d. 7
6. The interrupt control logic:
    a. manages interrupts
    b. manages interrupt acknowledge signals
    c. accepts interrupt acknowledge signal
    d. all of the above

8.10 Answers to Check Your Progress

1. C
2. C
3. D
4. A
5. C
6. D

8.11 Model Questions

1. Explain maskable and non-maskable interrupts.
2. What is meant by priority of interrupts?
3. In how many categories can ‘interrupt requests’ be classified?
4. What are software interrupts of 8085? Mention the instructions, their hex codes and the corresponding vector addresses.
5. In what way INTR is different from the other four hardware interrupts?
6. Discuss the INTR interrupt of 8085.
7. Draw the SIM instruction format and discuss.
8. Show the RIM instruction format and discuss the same.
9. Draw the interrupt circuit diagram for 8085 and explain.
10. Comment on the TRAP input of 8085.
11. Discuss the utility of RST software instruction.

PROGRAMMABLE PERIPHERAL INTERFACE

9.0 Learning Objectives

After going through this unit, you will be able to:

- Understand the importance of peripheral devices
- Know the main IC's which are to be interfaced with 8085
- Understand the features of Intel 8255 PPI
- Know the different ports of 8255 PPI
- Know the three modes on which the port A of Intel 8255 can operate
- Understand the pin diagram of Intel 8255
- Perform the programming in Intel 8055
- Perform the interfacing of Intel 8055 with Intel 8085 Microprocessor.

9.1 Introduction

Peripheral Interfacing is considered to be a main part of Microprocessor, as it is the only way to interact with the external world. The interfacing happens with the ports of

the Microprocessor. The main IC's which are to be interfaced with 8085 are:

1. 8255 PPI
2. 8259 PIC
3. 8251 USART
4. 8279 Key board display controller
5. 8253 Timer/ Counter
6. A/D and D/A converter interfacing.

9.2 Programmable Peripheral Interface(Intel 8255)

Intel 8255 is a programmable peripheral interface chip designed for parallel communication between microprocessor and I/O devices, which have a speed mismatch between each other.

Features of 8255:

- It has three 8-bit ports

- It can be operated in three different modes in I/O mode and in BSR mode

IC 8255 has three ports A, Band C. The ports A and B are 8 bit parallel ports. Port A can be programmed to work in any one of the three modes as input or output port. The three operating modes are:

Mode-0 - Simple I/O port

Mode-l - Handshake I/O port

Mode-2 - Bidirectional I/O port.

The port B can be programmed to work either in mode-0 or mode-1. The port C pins (8-pins) have different assignments depending on the mode of port A and B. If port A and B are programmed in mode-0, then the port C can perform anyone of the following function. As 8 bit parallel port in mode-0 for input or output.

1. As two numbers of 4 bit parallel port in mode-O for input or output.
2. The individual pins of port C can be set or reset for various control applications.

The various functions (assignments) of port C during the different operating modes of port A and B are listed in Table below.

Functions of port A & B	**PC_7**	**PC_6**	**PC_5**	**PC_4**	**PC_3**	**PC_2**	**PC_1**	**PC_0**
Port A & B in mode 0 Input/Output	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
Port A & B in mode 1 Input ports	I/O	I/O	IBF_A	$\overline{STB}_A$	$INTR_A$	$\overline{STB}_B$	IBF_A	$INTR_B$
Port A & B in mode 1 Output ports	$\overline{OB}$ $\overline{F}_A$	$\overline{AC}\overline{K}_A$	I/O	I/O	$INTR_A$	$\overline{AC}\overline{K}_B$	$\overline{OB}\overline{F}_B$	$INTR_B$
Port A in mode 2 Port B in mode 0	$\overline{OB}$ $\overline{F}_A$	$\overline{AC}\overline{K}_A$	IBF_A	$\overline{STB}_A$	$INTR_A$	I/O	I/O	I/O

Where,

I/O is Input/Output line
$\overline{STB}$ is Strobe
IBF is Input Buffer Full
INTR is interrupt request
OBF is Output Buffer Full
ACK is Acknowledgement
Subscript A is Port A signal
Subscript B is Port B signal

If ports A and Bare programmed in mode-l or mode-2, then some of the pins of port C are used for handshake signals and the remaining pins can be used as input/output lines or individually set/reset for control applications.

9.2.1 Pins, Signals and internal block diagram of 8255

The Intel 8255 have 40 pins and operates on +5V.

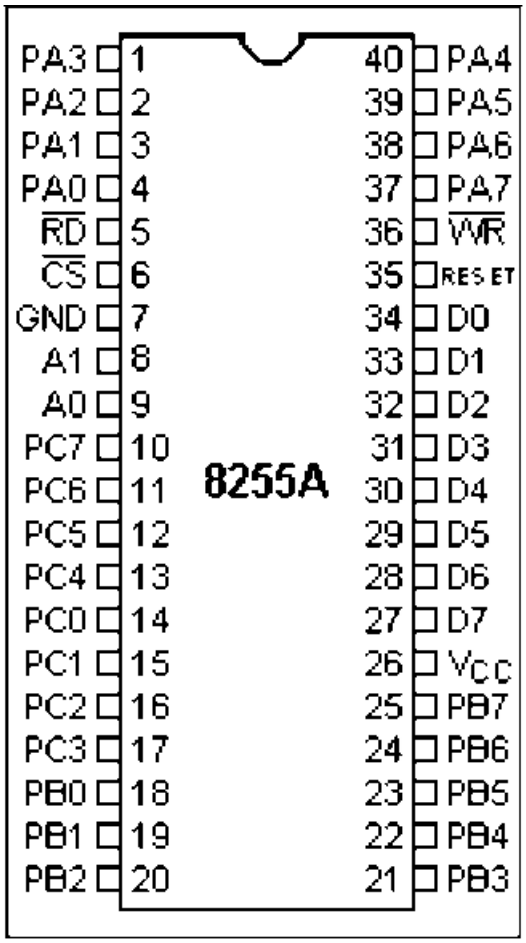


Figure 66: Pin diagram of Intel 8255 PPI

The pin description for Intel 8255 is as follows:

- Pin D_0-D_7 are bi-directional data lines
- RESET pin Resets input
- $\overline{CS}$ pin is for Chip Select
- $\overline{RD}$ pin is for Read Control
- $\overline{WR}$ pin is for Write control
- A_0 and A_1 pins are for Internal address
- PA_7-PA_0 pins are Port-A pins
- PB_7-PB_0 pins are Port-B pins
- PC_7-PC_0 pins are Port-C pins
- V_{CC} pin is for power(+5V)
- V_{SS} pin is connected to GROUND(0V)

9.2.2 Block Diagram of Intel 8255

The block diagram of Intel 8255 is shown below:

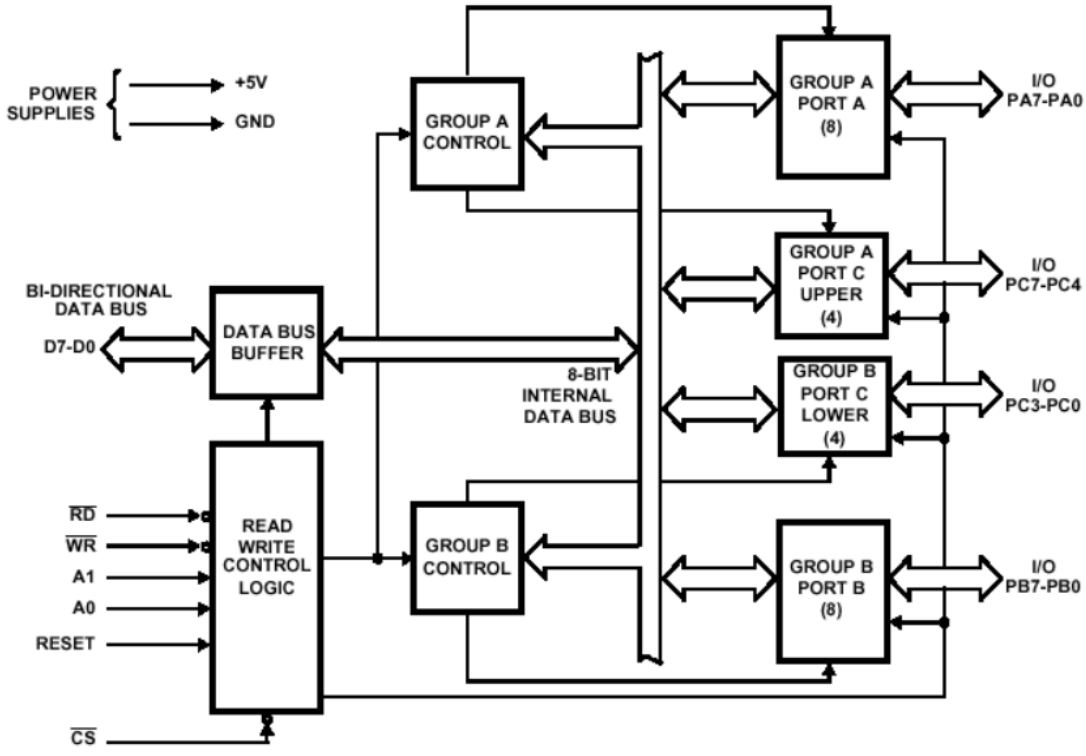


Figure 67: Block diagram on Intel 8255

The ports are grouped as Group A and Group B. The group A has port A, port C upper and its control circuit. The group B has port B, port C lower and its control circuit. The Read/Write control logic requires six control signals. These signals are given below.

RD (Read): This control signal enables the read operation. When this signal is LOW, the microprocessor reads data from a selected I/O port of the 8255A.

WR (Write): This control signal enables the write operation. When this signal goes LOW, the microprocessor writes into a selected I/O port or the control register.

RESET: This is an active HIGH signal. It clears the control register and set all ports in the input mode.

CS, A0 and A1: These are device select signals. The CS is connected to the decoder in the system. A0 and A1 are generally connected to A0 and A1 of the processor (Alternatively, A0 and A1 can be connected to any two-address lines of the processor). 8255 can be either Memory mapped or I/O mapped in the system.

A0 and A1 address lines can be made to select any one of the following four internal devices as shown on right side.

Internal Address		Device Selected
A_1	A_0	
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control Register

9.3 Modes of Intel 8255

Mode-0: In this mode, all the three ports can be programmed either as input or output port. In mode-O, the outputs are latched and the inputs are not latched. The ports do not have handshake or Interrupt capability. The ports in mode-o can be used to interface DIP switches, Hexa-keypad, LED's and 7-segment LED's to the processor.

Mode-l: In this mode, only ports A & B can be programmed either as input or output port. In mode-1, handshake signals are exchanged between the processor and peripherals prior to data transfer. The port C pins are used for handshake signals. Input and output data are latched. Interrupt driven data transfer scheme is possible.

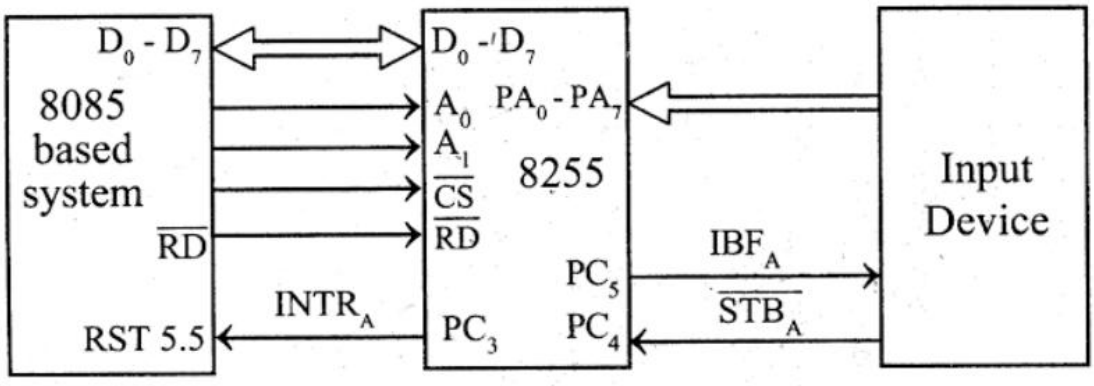


Figure 68: Intel 8255 Handshake Input Mode

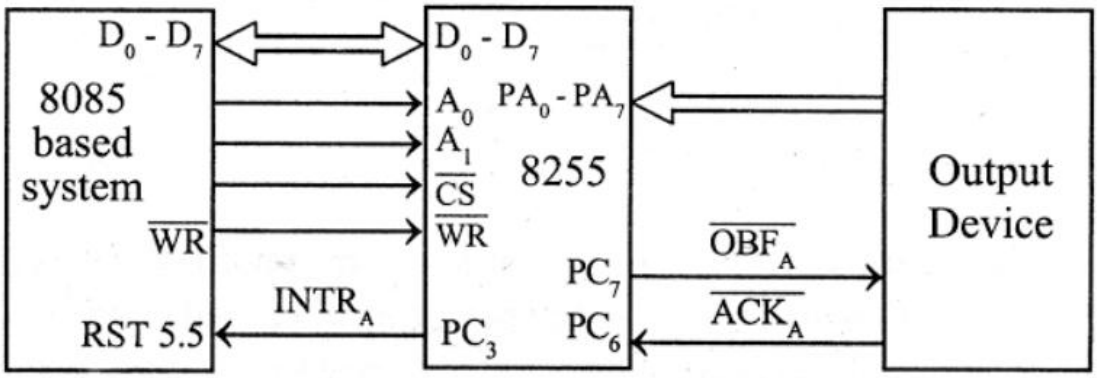


Figure 69: Intel 8255 Handshake Output Mode

Mode-2: In this mode, the port will be a bi-directional port (i.e., the processor can perform both read and write operations with an I/O device connected to a port in mode-2). Only port-A can be programmed to work in mode-2. Five pins of port C are used for handshake signals. This mode is used primarily in applications such as data transfer between two computers or floppy disk controller interface.

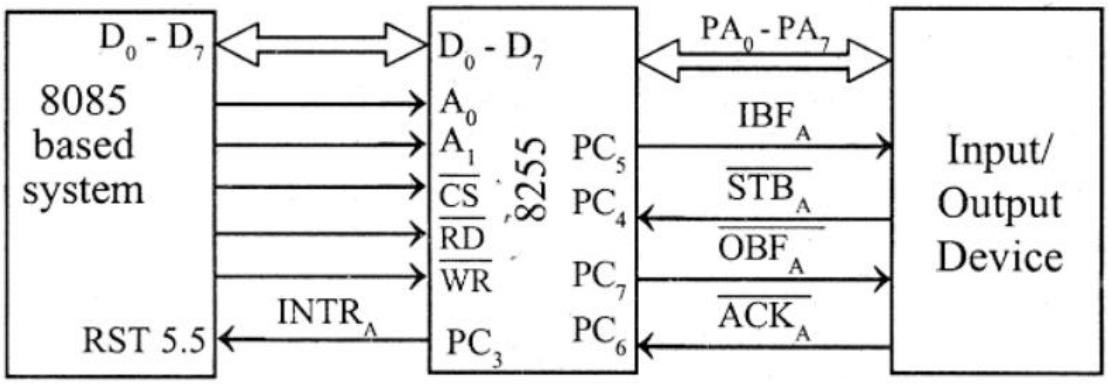


Figure 70: Intel 8255 configured in Mode 2

9.4 Programming in Intel 8255

The 8255 has two control words, one for specifying 1/O functions and another for bit set/reset mode of port C. Both the control words are written in the same control register. The control register differentiate them by the value of bit D7 .The bit set/reset control word does not affect the functions of ports A and B.

Bit D7 of the control register specifies either the 1/0 function or the bit set / reset function.

If bit D7 = 1, then the bits D6 – D0 determine 1/0 functions in various modes.

If bit D7 = 0, then the bits D6 – D0 determine the pin of port C to be set or reset.

The 8255 ports are programmed (or initialized) by writing a control word in the control register. For setting 1/0 functions and mode of operation the 1/0 mode control word is send to control register. The format of the 1/0 mode set control word is shown below.

For setting/resetting (BSR mode) a pin of port C, the bit set/ reset control word is sent to control register. The format of bit set/reset control word is shown below.

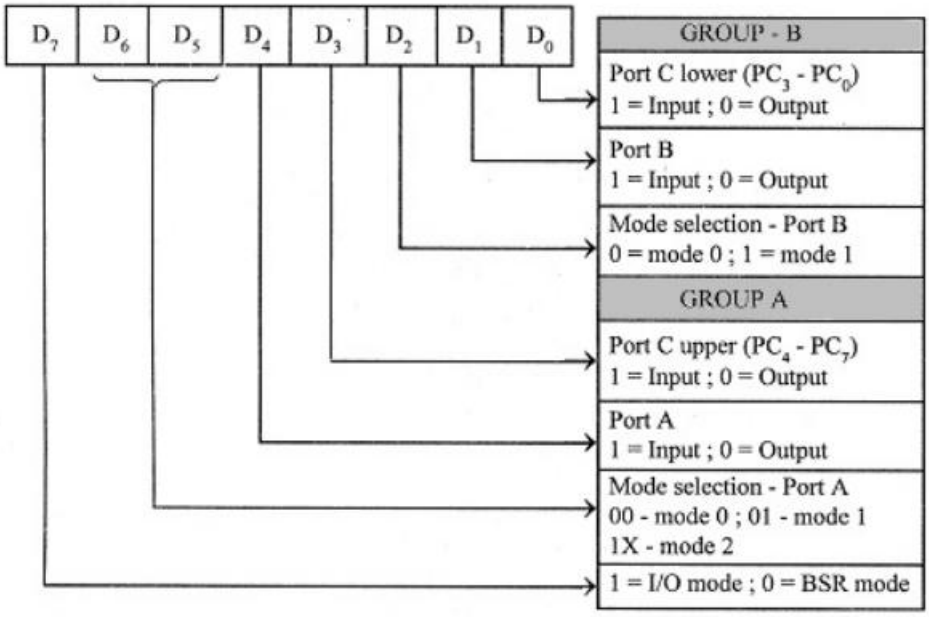


Figure 71: Intel 8255 Control word

The data transfer between the processor and the port can be either interrupt driven or through status check. In the interrupt driven data transfer scheme, when the port is ready, it interrupts the processor for a read or write operation. In status check technique, the processor polls the status of the port and checks whether the port is ready for data transfer or not. The status of the ports A and B can be known by reading the port C. When the port is ready for data transfer, the processors executes a read or write cycle.

9.5 Interfacing of 8255 with 8085 processor

A simple schematic for interfacing the 8255 with 8085 processor is shown in Figure 72. The 8255 can be either memory mapped or I/O mapped in the system. In the schematic shown in above is I/O mapped in the system. Using a 3-to-8 decoder generates the chip select signals for I/O mapped devices. The address lines A4, A5 and A6 are decoded to generate eight chip select signals (IOCS-0 to IOCS-7) and in this, the chip select IOCS- 1 is used to select 8255. The address line A7 and the control signal IO/M (low) are used as enable for the decoder. The address line A0 of 8085 is connected to A0 of 8255 and A1 of 8085 is connected to A1 of 8255 to provide the internal addresses. The data lines D0-D7 are connected to D0-D7 of the processor to achieve parallel data transfer.

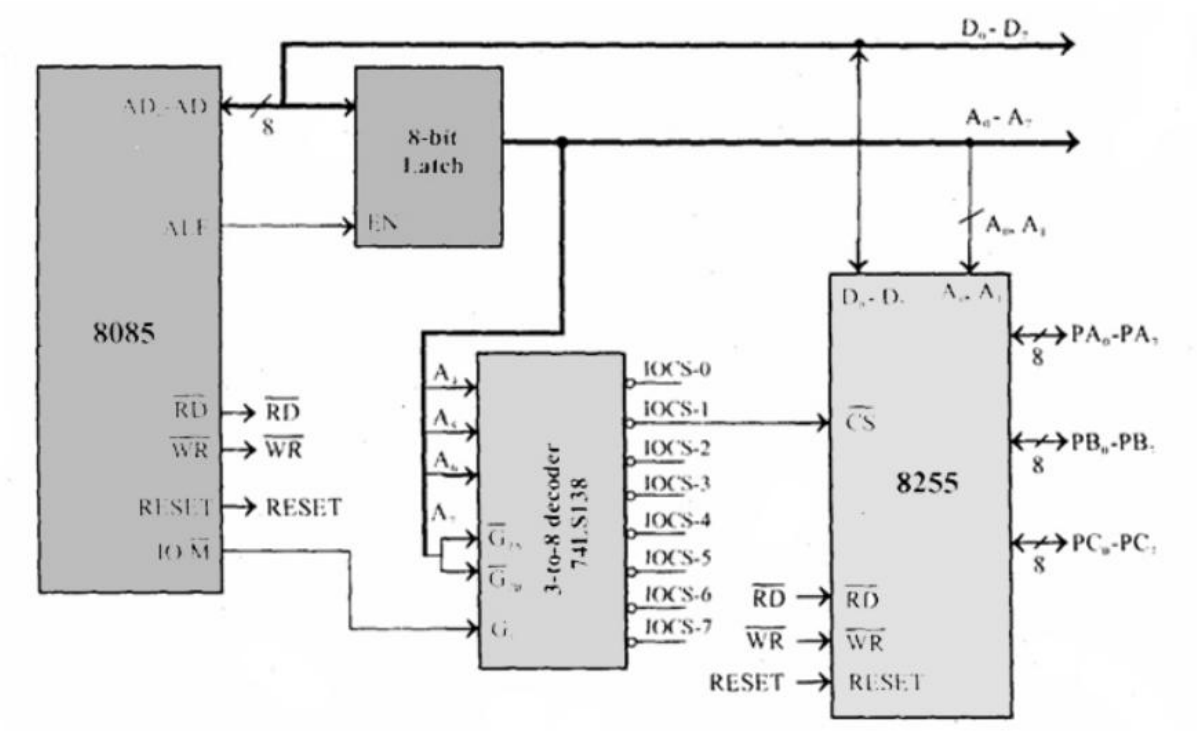


Figure 72: Interfacing of Intel 8255 PPI with Intel 8085

The I/O addresses allotted to the internal devices of 8255 are listed in Table 5.

Table 5 : The I/O addresses allottment in Intel 8255

Internal Device	Binary Address								Hexa Address
	Decoder input and enable				Input to address pins of 8055				
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Port A	0	0	0	1	x	x	0	0	10
Port B	0	0	0	1	x	x	0	1	11
Port C	0	0	0	1	x	x	1	0	12
Control Register	0	0	0	1	x	x	1	1	13

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

9.6 Summary

1. Peripheral Interfacing is considered to be a main part of Microprocessor, as it is the only way to interact with the external world.
2. The interfacing happens with the ports of the Microprocessor.
3. Intel 8255 is a programmable peripheral interface chip designed for parallel communication between microprocessor and I/O devices, which have a speed mismatch between each other.
4. The 8255A is widely used, programmable, parallel I/O device .It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O.
5. The mode is used primarily in applications such as data transfer between two computers or floppy disk controller interface.
6. In the I/O mode, the 8255 ports work as programmable I/O ports.0
7. In BSR mode, only port C can be used to set and reset individual ports.
8. The strobed input/output mode is another name of Mode 2.
9. If the value of the pin STB (Strobe Input) falls to low level, then input port is loaded into input latches.
10. The signals that are provided to maintain proper data flow and synchronization between the data transmitter and receiver are known as handshaking signals.

9.7 Check Your Progress

1. PPI stands for _____.
2. Specify the bit of a control word for the 8255, which differentiates between the I/O mode and the BSR mode.
3. Port C of 8255 can function independently as:
    a. input port
    b. output port
    c. either input or output ports
    d. both input and output ports
4. All the functions of the ports of 8255 are achieved by programming the bits of an internal register called:

a. data bus control
b. read logic control
c. control word register
d. none

5. The data bus buffer is controlled by
    a. control word register
    b. read/write control logic
    c. data bus
    d. none
6. The port that is used for the generation of handshake lines in mode 1 or mode 2 is
    a. port A
    b. port B
    c. port C Lower
    d. port C Upper
7. If A1=0, A0=1 then the input read cycle is performed from
    a. port A to data bus
    b. port B to data bus
    c. port C to data bus
    d. CWR to data bus
8. The pin that clears the control word register of 8255 when enabled is:
    a. CLEAR
    b. SET
    c. RESET
    d. CLK

9.8 Answers to Check Your Progress

1. Programmed Peripheral Interface
2. BSR mode D7=0, and I/O mode D5=1
3. C
4. C
5. B

6. D
7. B
8. C

9.9 Model Questions

1. Where are the Handshake signals in 8255?
2. What are the features used mode 1 in 8255?
3. How many ports are present in 8255?
4. How port C works in 8255?
5. How many modes are present in 8255?
6. What is the function of BSR mode in 8255? In which application, BSR mode can be used?
7. What are the functions of mode1, mode2 and mode3?
8. Give the Control Word Format of 8255 in I/O and BSR mode.
9. Discuss mode –2 (bi-directional mode) of 8255 (Programmable Peripheral Interface).

PROGRAMMABLE INTERVAL TIMER: INTEL 8253

10.0 Learning Objectives

After going through this unit, you will be able to:

- Understand the features of Intel 8253
- Know the different ports of 8253
- Understand the pin diagram of Intel 8253

10.1 Introduction

The Intel 8253 is a programmable counter / timer chip designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP. It is organized as 3 independent 16-bit counters, each with a counter rate up to 2 MHz. All modes of operation are software programmable. The 82C54 is pin compatible with the HMOS 8254, and is a superset of the 8253. Six programmable timer modes allow the 82C54 / 8253 to be used as an event counter, elapsed time indicator, programmable one-shot, and in many other applications.

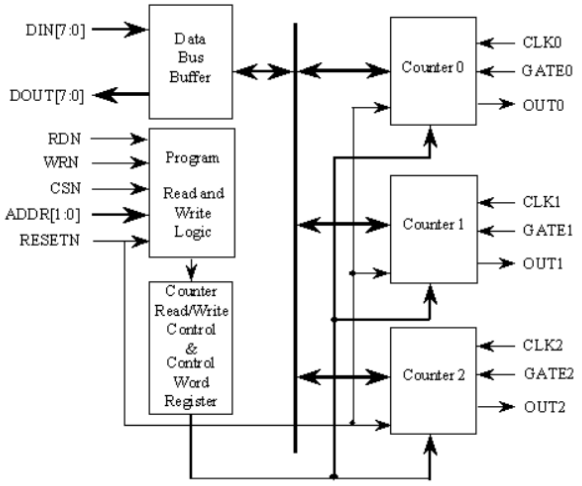


Figure 73: Block diagram of an 8253 programmable interval timer

The block labeled data bus buffer contains the logic to buffer the data bus to / from the microprocessor, and to the internal registers. The block labeled read / write logic controls the reading and the writing of the counter registers. The final block, the control word register, contains the programmed information that is sent to the device from the microprocessor. In effect this register defines how the 8253 logically works.

Each counter in the block diagram has 3 logical lines connected to it. Two of these lines, clock and gate, are inputs. The third, labeled OUT is an output. The function of these lines changes and depends on how the device is initialized or programmed.

10.2 Pin Configuration

The following picture shows the pin configuration of the 8253 and a general definition of the lines follows:

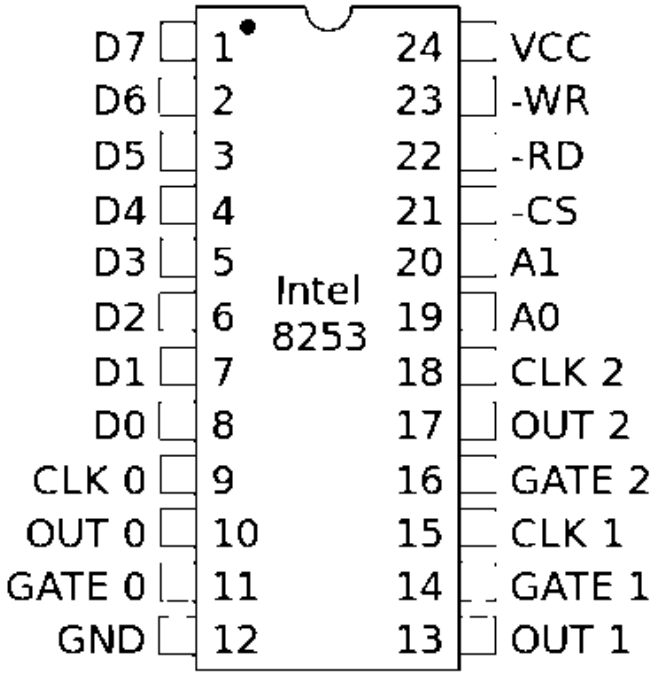


Figure 74: Pin diagram of Intel 8253[14]

[14] Adopted from https://en.wikipedia.org/wiki/Intel_8253#/media/File:Intel_8253_and_8254.svg

Clock: This is the clock input for the counter. The counter is 16 bits. The maximum clock frequency is 1 / 380 nanoseconds or 2.6 megahertz. The minimum clock frequency is DC or static operation.

Out: This single output line is the signal that is the final programmed output of the device. Actual operation of the outline depends on how the device has been programmed.

Gate :This input can act as a gate for the clock input line, or it can act as a start pulse, depending on the programmed mode of the counter.

This table shows the different uses of the 8253 gate input pin. Each mode of operation for the counter has a different use for the GATE input pin.

Table 6 : Different uses of the 8253 gate input pin

Signal Status	Low or going High	Rising	High
Mode			
0	Disable counting	--	Enables Counting
1	--	Initiates counting Resets output after next clock	--
2	Disable counting Sets output immediately high	Reloads counter Initiates counting	Enables Counting
3	Disable counting Sets output immediately high	Initiates counting	Enables Counting
4	Disable counting	--	Enables Counting
5	--	Initiates counting	--

10.3 Internal 8253 registers

Here is a list of the internal 8253 registers that will program the internal counters of the 8253:

Table 7: List of the internal 8253 registers

	$\overline{RD}$	$\overline{WR}$	A0	A1	Function
Counter 0	1	0	0	0	Load counter 0
	0	1	0	0	Load counter 0
Counter 1	1	0	0	1	Load counter 1
	0	1	0	1	Load counter 1
Counter 2	1	0	1	0	Load counter 2
	0	1	1	0	Load counter 2
MODE WORD or CONTROL WORD	1	0	1	1	Write mode word
—	0	1	1	1	No-operation

Counter: #0, #1, #2 Each counter is identical, and each consists of a 16-bit, pre-settable, down counter. Each is fully independent and can be easily read by the CPU. When the counter is read, the data within the counter will not be disturbed. This allows the system or your own program to monitor the counter's value at any time, without disrupting the overall function of the 8253.

Control Word Register: This internal register is used to write information to, prior to using the device. This register is addressed when A0 and A1 inputs are logical 1's. The data in the register controls the operation mode and the selection of either binary or BCD (binary coded decimal) counting format. The register can only be written to. You can't read information from the register.

10.3.1 Control Word Register

CONTROL BYTE D7 - D0							
D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCP

All of the operating modes for the counters are selected by writing bytes to the control register. This is the control word format.

D7 SC1	D6 SC0	Counter Select
0	0	counter 0
0	1	counter 1
1	0	counter 2
1	1	illegal value

Bits D7 and D6 are labeled SC1 and SC0. These bits select the counter to be programmed, it is necessary to define, using the control bits D7 and D6, which counter is being set up. Once a counter is set up, it will remain that way until it is changed by another control word.

Bits D5 and D4 (RL1 / RL0) of the control word shown above are defined as the read / load mode for the register that is selected by bits D7 and D6. Bits D5 and D4 define how the particular counter is to have data read from or written to it by the CPU.

D5 RL1	D4 RL0	R / L Definition
0	0	Counter value is latched. This means that the selected counter has its contents transferred into a temporary latch, which can then be read by the CPU.
0	1	Read / load least-significant byte only.
1	0	Read / load most-significant byte only.
1	1	Read / load least-significant byte first, then most-significant byte.

These bits are defined as:

The 1st value, 00, is the counter latch mode. If this mode is specified, the current counter value is latched into an internal register at the time of the I/O write operation to the control register. When a read of the counter occurs, it is this latched value that is read.

Caution: If the latch mode is not used, then it is possible that the data read back may be in the process of changing while the read is occurring. This could result in invalid data being input by the CPU. To read the counter value while the counter is still in the process of counting, one must

first issue a latch control word, and then issue another control word that indicates the order of the bytes to be read.

The next 3 bits of the control word are D3, D2, and D1. These bits determine the basic mode of operation for the selected counter. The mode descriptions follows:

D3 M2	D2 M1	D1 M0	Mode value
0	0	0	mode 0: interrupt on terminal count
0	0	1	mode 1: programmable one-shot
x	1	0	mode 2: rate generator
x	1	1	mode 3: square wave generator
1	0	0	mode 4: software triggered strobe
1	0	1	mode 5: hardware triggered strobe

The final bit **D0** of the control register determines how the register will count:

D0	counts down in
0	binary
1	BCD

The **maximum values** for the count in each count mode are 10^4 (10,000 decimal) in BCD, and 2^{16} (65,536 decimal) in binary.

10.4 Modes

The following text describes all possible modes. The modes used in the MZ-700 and set by the monitor's startup are mode 0, mode 2, and mode 3.

Mode 0: Interrupt on Terminal Count

The counter will be programmed to an initial value and afterwards counts down at a rate equal to the input clock frequency. When the count is equal to 0, the OUT pin will be a logical 1. The output will stay a logical 1 until the counter is reloaded with a new value or the same value or until a mode word is written to the device.

Once the counter starts counting down, the GATE input can disable the internal counting by setting the GATE to a logical 0.

Mode 1: Programmable One-Shot

In mode 1, the device can be setup to give an output pulse that is an integer number of clock pulses. The one-shot is triggered on the rising edge of the GATE input. If the trigger occurs during the pulse output, the 8253 will be retriggered again.

Mode 2: Rate Generator

The counter that is programmed for mode 2 becomes a "divide by n" counter. The OUT pin of the counter goes to low for one input clock period. The time between the pulses of going low is dependent on the present count in the counter's register. I mean the time of the logical 1 pulse.

For example, suppose to get an output frequency of 1,000 Hz (Hertz), the period would be 1 / 1,000 s = 1 ms (millisecond) or 1,000 µs (microseconds). If an input clock of 1 MHz (Mega-Hertz) were applied to the clock input of the counter #0, then the counter #0 would need to be programmed to 1000 µs. This could be done in decimal or in BCD. (The period of an input clock of 1 MHz is 1 / 1,000,000 = 1 µs.)

The formula is: $n=f_i$ divided by f_{out}.

f_i = input clock frequency, f_{out} = output frequency, n = value to be loaded.

Mode 3: Square Wave Generator

Mode 3 is similar to the mode 2 except that the output will be high for half the period and low for half. If the count is odd, the output will be high for (n + 1) / 2 and low for (n - 1) / 2 counts.

Mode 4: Software Triggered Strobe

In this mode the programmer can set up the counter to give an output timeout starting when the register is loaded. On the terminal count, when the counter equals to 0, the output will go to a logical 0 for one clock period and then returns to a logical 1. First the mode is set, the output will be a logical 1.

Mode 5: Hardware Triggered Strobe

In this mode the rising edge of the trigger input will start the counting of the counter. The output goes low for one clock at the terminal count. The counter is retriggerable, thus meaning that if the trigger input is taken low and then high during a count sequence, the sequence will start over.

When the external trigger input goes to a logical 1, the timer will start to time out. If the external trigger occurs again, prior to the time completing a full timeout, the timer will retrigger.

10.5 Summary

1. The 8253 is programmable interval timer/counter specifically designed for use with the Intel micro- computer systems.
2. The 8253 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control.
3. Instead of setting up timing loops in systems software, the programmer configures the 8253 to match his requirements, initializes one of the counters of the 8253 with the desired quantity, then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks.
4. It is easy to see that the software overhead is minimal and that multiple delays can easily be maintained by assignment of priority levels.

10.6 Check Your Progress

1. The _____ is a programmable counter/timer chip designed for use as an Intel Microprocessor peripheral.
2. Intel 8253 is packaged in a ____ pin plastic DIP.
3. _____ programmable timer modes allow the 8253 to be used.
4. Intel 8253 has _____ counters.

10.7 Answers to Check Your Progress

1. Intel 8253
2. 24
3. Six
4. 3

10.8 Model Questions

1. Discuss the various modes of Intel 8253.
2. Discuss internal registers of Intel 8253.
3. Draw the pin diagram of Intel 8253.
4. Explain the working of Intel 8253 using block diagram.

PROGRAMMABLE INTERRUPT CONTROLLER 8253/8254

11.0 Learning Objectives

After going through this unit, you will be able to:

- Know the functioning of Intel 8254
- Know the pin diagram and pin description of Intel 8254
- Know the Control Word of Intel 8254 PIT
- Understand the various modes on which the 8054 operates.

11.1 Introduction

The Intel 8253 and 8254 are Programmable Interval Timers(PITs), which perform timing and counting functions. They were primarily designed for the Intel 8080/8085- processors. After the desired delay, the 8254 will interrupt the CPU. It is 24 pin IC requires +5 V. It consists of 3 16 bit counters operates in 6 modes. It generates accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the interval timer to match system requirements and programs the counter for the desired delay or for the desired output. Some of the other counter/timer functions common to microcomputers which can be implemented with the 8254 are:

- Real Time Clock
- Event-counter
- Digital one-shot
- Programmable rate generator
- Square wave generator
- Binary rate multiplier
- Complex waveform generator
- Complex motor generator

11.2 Block Diagram of Intel 8254/8253

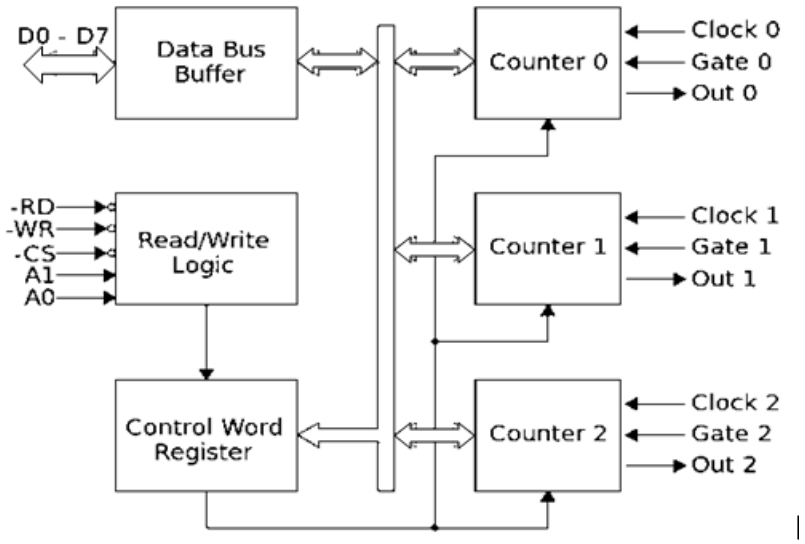


Figure 75: Block diagram of Intel 8254/8253

Data/Bus Buffer

This block contains the logic to buffer the data bus to / from the microprocessor, and to the internal registers. It has 8 input pins, usually labeled as $D_7..D_0$, where D7 is the MSB.

Read/Write Logic

RD: read signal

$\overline{W}\overline{R}$: write signal

$\overline{CS}$: chip select signal

A0, A1: address lines

Operation mode of the PIT is changed by setting the above hardware signals. For example, to write to the Control Word Register, one needs to set /CS=0, /RD=1, /WR=0, A1=A0=1.

The timer has three counters, called channels. Each channel can be programmed to operate in one of six modes. Once programmed, the channels can perform their tasks independently. The

timer is usually assigned to IRQ-0 (highest priority hardware interrupt) because of the critical function it performs and because so many devices depend on it.

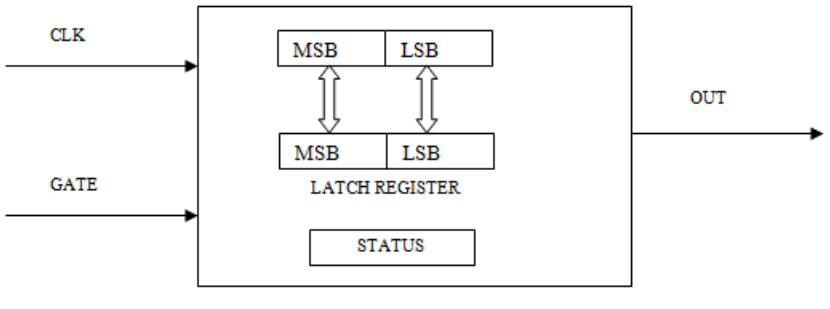


Figure 76; Timer/Counter Channel

Counters

There are 3 counters (or timers), which are labeled as "Counter 0", "Counter 1" and "Counter 2". Each counter has 2 input pins – "CLK" (clock input) and "GATE" and 1-pin, "OUT", for data output. The 3 counters are 16-bit down counters independent of each other, and can be easily read by the CPU.

Counters are programmed by writing a Control Word and then an initial count. GATE=1 enables counting, GATE=0 disables counting. All 3 counters are 16-bits. PIT has only an 8-bit data bus and can read or write only one byte at a time. Thus to read or write a 16-bit value, you must do so one byte at a time.

For example, to load counter0 with 38370 decimal, the LSB is 226 decimal, and its MSB is 149 decimal.

1	0	0	1	0	1	0	1	1	1	1	0	0	0	1	0	=38370
MSB= 149 Decimal								LSB= 226 Decimal								Decimal

Figure 77: A 16-bit number has LSB and MSB

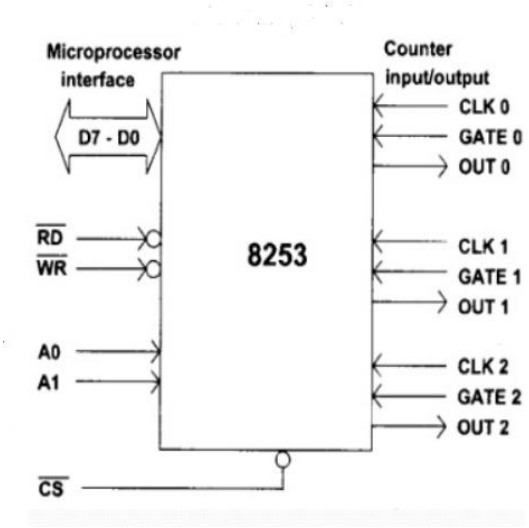


Figure 78: Block diagram of Intel 8254/8253

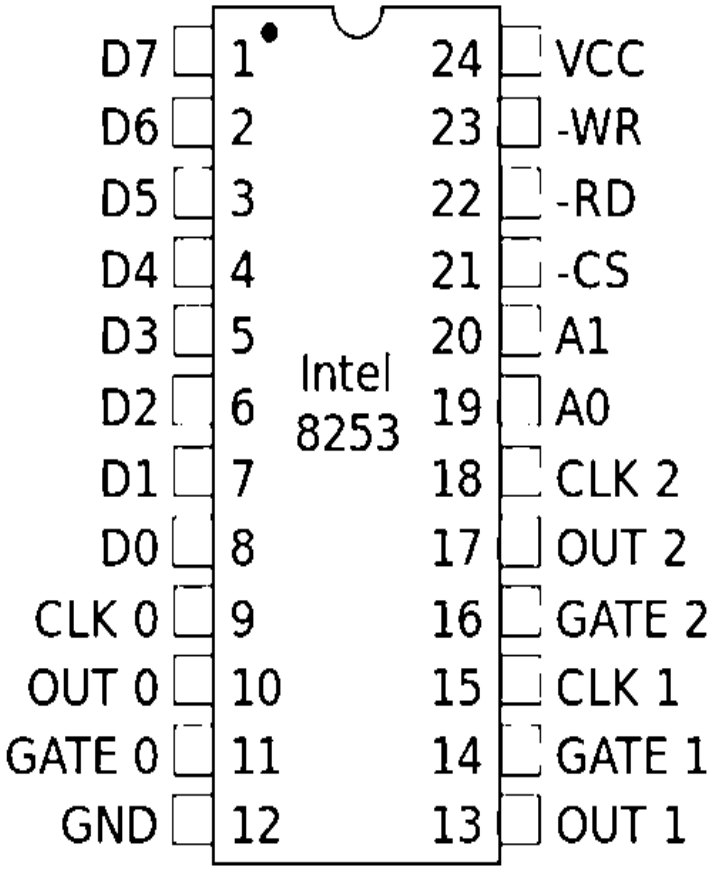


Figure 79: Pin diagram of Intel 8254/8253

Pin Description

<table>
<tr><th>Symbol</th><th>Pin No.</th><th>Type</th><th>Name and Function</th></tr>
<tr><td>D7-D0</td><td>1-8</td><td>I/O</td><td>DATA: Bi-directional three state data bus lines, connected to the system data bus.</td></tr>
<tr><td>CLK 0</td><td>9</td><td>I</td><td>CLOCK 0: Clock input of Counter 0.</td></tr>
<tr><td>OUT 0</td><td>10</td><td>O</td><td>OUTPUT 0: Output of Counter 0.</td></tr>
<tr><td>GATE 0</td><td>11</td><td>I</td><td>GATE 0: Gate input of Counter 0.</td></tr>
<tr><td>GND</td><td>12</td><td></td><td>GROUND: Power supply connection.</td></tr>
<tr><td>VCC</td><td>24</td><td></td><td>POWER: A 5V power supply connection.</td></tr>
<tr><td>$\overline{WR}$</td><td>23</td><td>I</td><td>WRITE CONTROL: This input is low during CPU write operation</td></tr>
<tr><td>RD</td><td>22</td><td>I</td><td>READ CONTROL: This input is low during CPU read operation.</td></tr>
<tr><td>CS</td><td>21</td><td>I</td><td>CHIP SELECT: A low on this input enables the 8254 to respond to RD and WR signals. RD and WR are ignored otherwise.</td></tr>
<tr><td>A1,A0</td><td>20-9</td><td>I</td><td>ADDRESS: Used to select one of the three counters or the control word register for read or write operations. Normally connected to the system address bus
<table>
<tr><th>A1</th><th>A0</th><th>Select</th></tr>
<tr><td>0</td><td>0</td><td>Counter 0</td></tr>
<tr><td>0</td><td>1</td><td>Counter 1</td></tr>
<tr><td>1</td><td>0</td><td>Counter 2</td></tr>
<tr><td>1</td><td>1</td><td>Control Word Register</td></tr>
</table></td></tr>
<tr><td>CLK 2</td><td>18</td><td>I</td><td>CLOCK 2: Clock input of counter 2.</td></tr>
<tr><td>OUT 2</td><td>17</td><td>O</td><td>OUT 2: Output of Counter 2.</td></tr>
<tr><td>GATE 2</td><td>16</td><td>I</td><td>GATE 2: Gate Input of Counter 2.</td></tr>
<tr><td>CLK 1</td><td>15</td><td>I</td><td>CLOCK 1: Clock Input of Counter 1.</td></tr>
<tr><td>GATE 1</td><td>14</td><td>I</td><td>GATE 1: Gate Input of Counter 1.</td></tr>
<tr><td>OUT 1</td><td>OUT 1</td><td>O</td><td>OUT 1: Output of Counter 1.</td></tr>
</table>

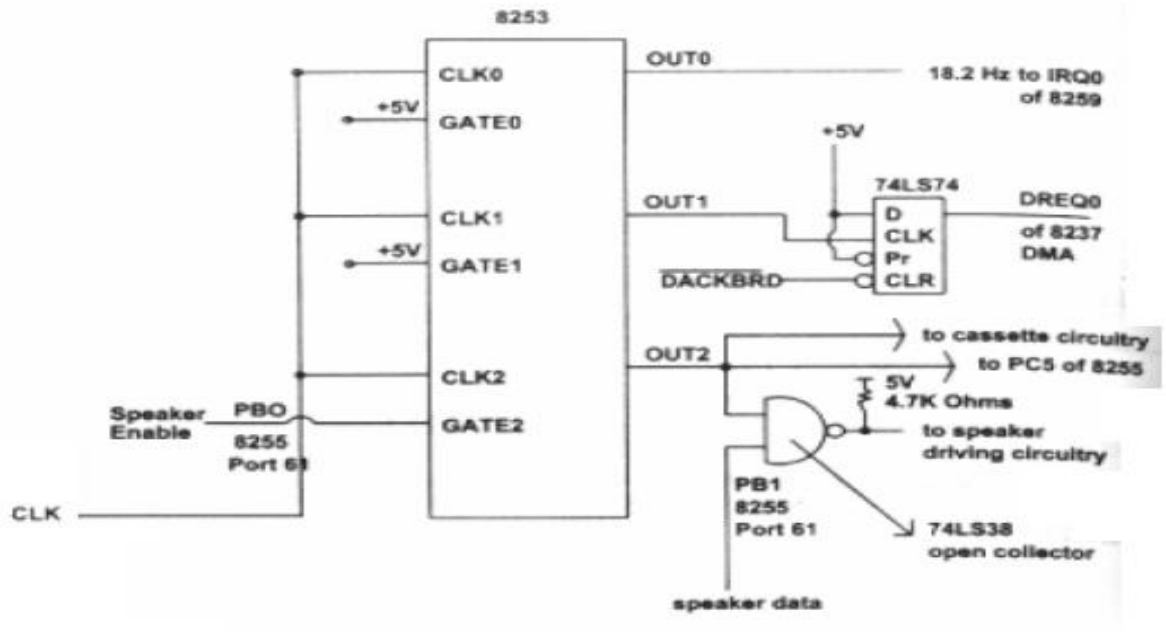


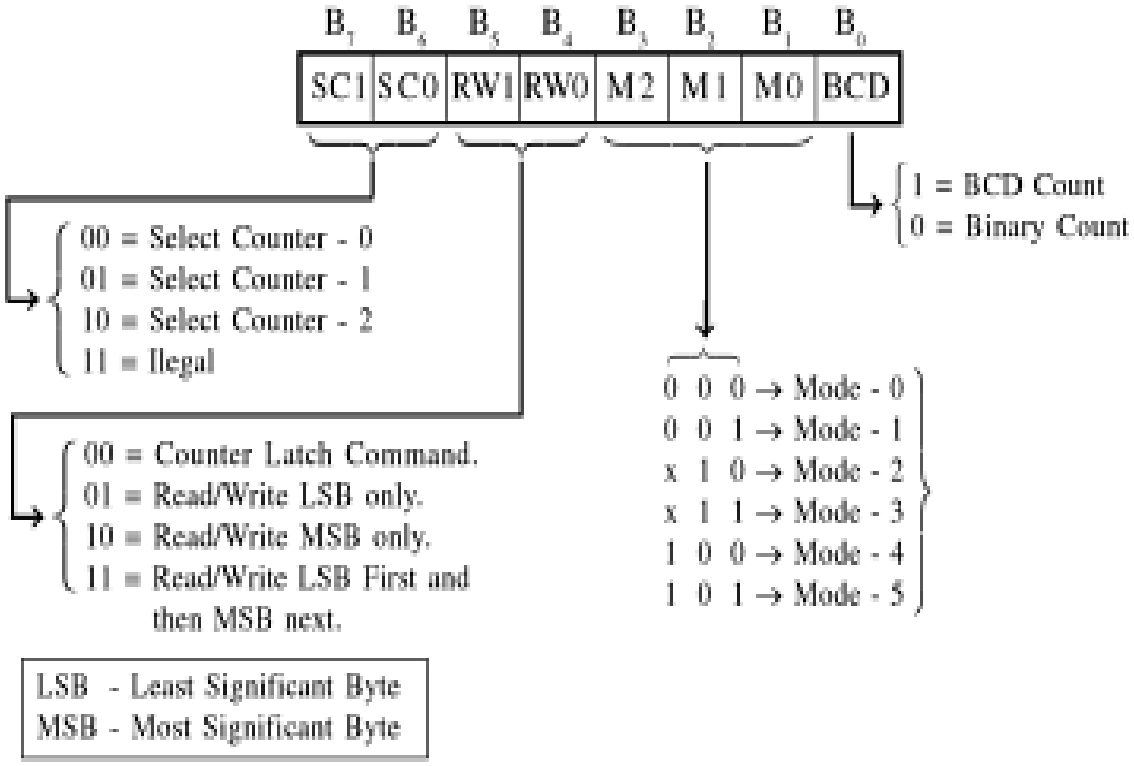
Figure 80: 8254/ 8253 chip connections

Counters are programmed by writing a Control Word and then an initial count. Control Words are written into the Control Word Register, which is selected when A0,A1=11. The Control Word itself specifies which Counter is being programmed. Initial counts are written into the Counters, not the Control Word Register. The A0,A1 inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

11.3 Control Word of PIT 8254

7	6	5	4	3	2	1	`0
CHANNEL		COMMAND		OUTPUT Mode			Binary/BCD
CHANNEL ID 00= Chn 0 01= Chn 1 10= Chn 2		COMMAND ID 00=Latch 01= LSB r/w 10= MSB r/w 11= LSB-MSB r/w		OUTPUT MODE 000= one-shot level 001= retriggerable 010 rate-generator 011= square –wave 100= software strobe 101- hardware strobe			Counting Mode 0= Binary 1= BCD

Figure 81: Control word of PIT 8254



There are 5 steps to calculate the proper control word number:

- Step 1: Chose one counter (SC1 and SC0)
- Step 2: Chose method to load (RW1 and RW0)
- Step 3: Chose programming mode (M2, M1 and M0)
- Step 4: Chose binary or BCD number (BCD).
- Step 5: Convert the final binary word into decimal

A0 and A1 select one of the three counters or the control word register to be read from/written into as specified in the following table.

Table 8 : Counter table

A1	A0	Selected Block
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word Register

For example: Suppose you want to make counter 0 a binary counter that generates square waves, and uses LSB and MSB read/write loading. What is the proper control word?

Figure shows the answer. From Step 1, SC1 and SC0 both are 0 if you want to use counter 0. Step 2 say that LSB and MSB are both loaded. For square wave generation, you use Mode 3 and Step 3 says that M2,M1, M0 = 011. Step 4 makes BCD = 0 to have a binary counter. Thus the resulting binary number is 00110110 or, 54 decimal. Thus you would load the control register with control Word with 54:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	0	1	1	0

Example has the 8-bit control word- 00110110 binary which in decimal is equal to 54.

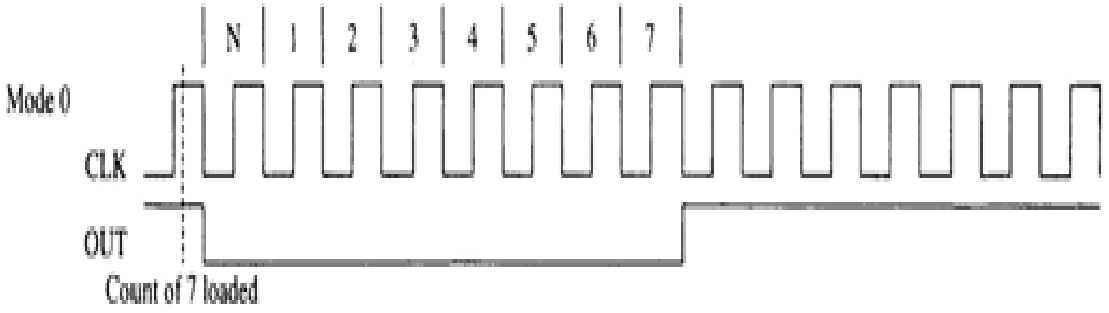
11.4 Operating Modes PIT 8254

There are six modes of operation for the counters. In all modes the counters operate as down counters. They are defined as follows:

- Mode 0: Interrupt on Terminal Count/ Event counter
- Mode 1: Hardware Re triggerable One-Shot
- Mode 2: Rate Generator
- Mode 3: Square Wave Mode
- Mode 4: Software Triggered Strobe
- Mode 5: Hardware Triggered Strobe (Re triggerable)

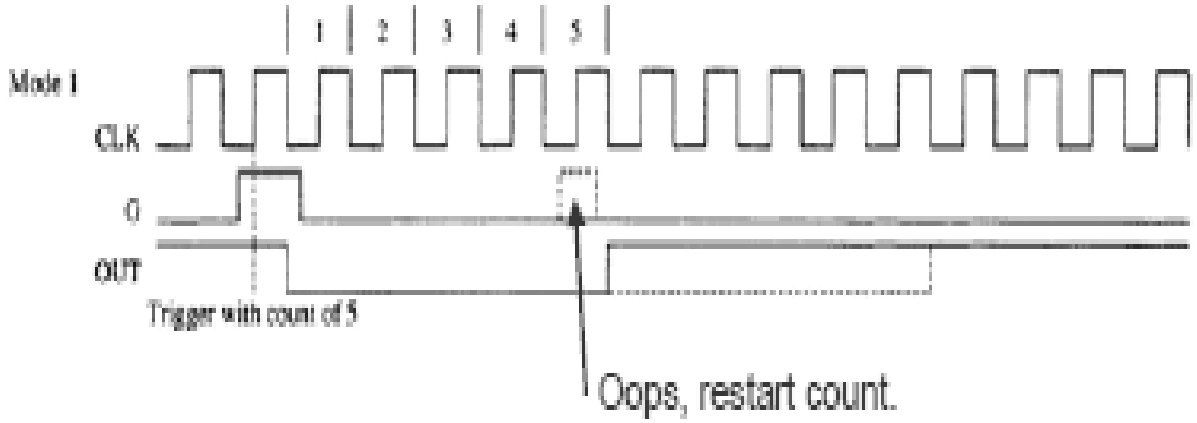
11.4.1 MODE 0: Interrupt on terminal count

1. Event counting.
2. After the Control Word is written, OUT is initially low and remains low.
3. When the counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written into the Counter.



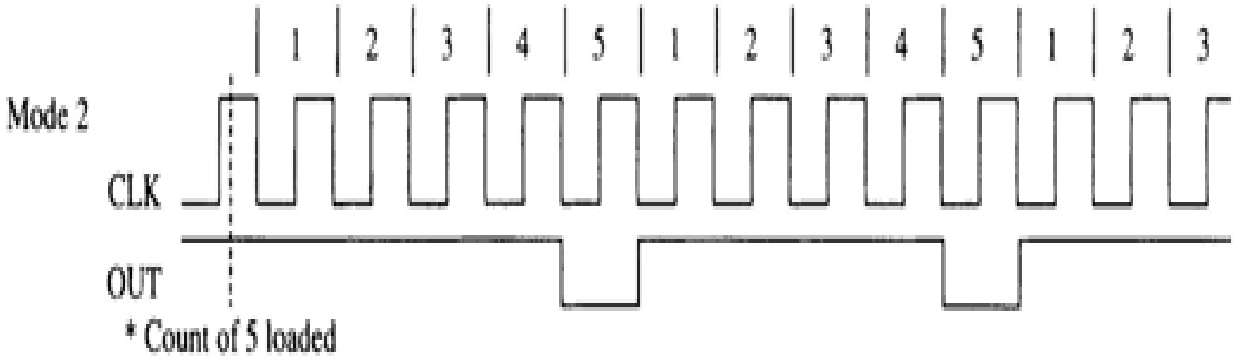
11.4.2 MODE 1: Hardware re-triggerable one-shot

1. OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and remain low until the Counter reaches zero.
2. OUT will then go high and remain high until the CLK pulse after the next trigger.



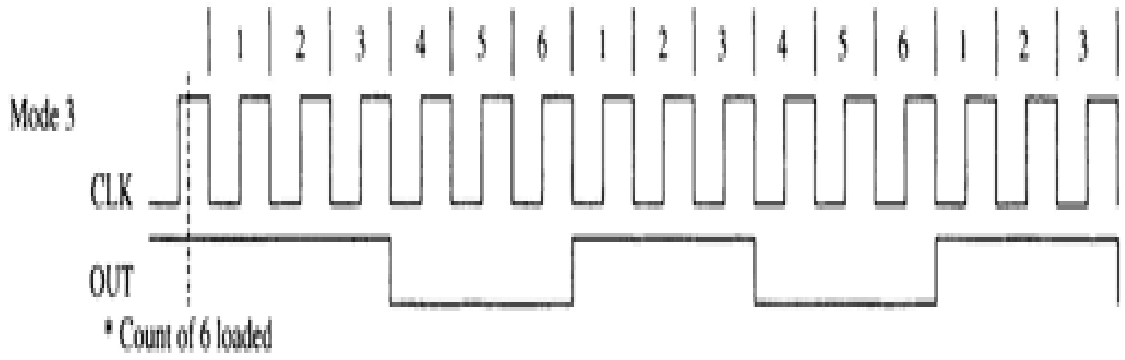
11.4.3 MODE 2: Rate generator

1. Functions like a divide-by-N counter and used to generate a Real Time Clock interrupt.
2. OUT will initially be high.
3. When the initial count has decremented to one, OUT goes low for one CLK pulse.
4. Out then goes high again, the Counter reloads the initial count and the process is repeated.
5. MODE 2 is periodic. The same sequence is repeated indefinitely.



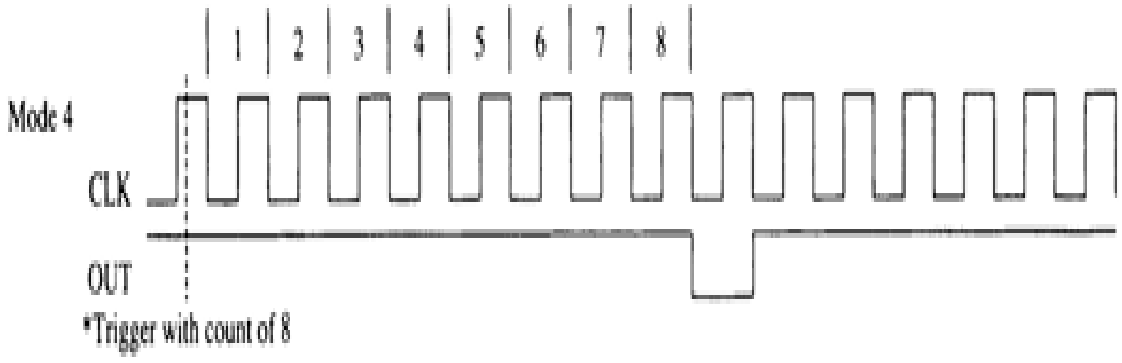
11.4.4 MODE 3: Square wave mode

1. Typically used for baud rate generation.
2. Out will initially be high.
3. When half the initial count is expired, OUT goes low for the remainder of the count.
4. MODE 3 is periodic. The same sequence is repeated indefinitely.



MODE 4: Software triggered strobe

1. OUT will initially be high.
2. When the initial count expires, OUT will go low for one CLK pulse and then go high again.
3. The counting sequence is "triggered" by writing the initial count.
4. The Counter is loaded on the next CLK pulse following writing a Control Word and initial count.



11.4.5 MODE 5: Hardware triggered strobe (retriggerable)

1. OUT will initially be high.
2. Counting is triggered by a rising edge of GATE.
3. When the initial count expires, OUT will go low for one CLK pulse and then go high again.
4. The difference between MODE 4 and MODE 5 is that in MODE 5 the count will not be loaded until the CLK pulse after a trigger.

11.5 SUMMARY

1. Intel 8054 generates accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the interval timer to match system requirements and programs the counter for the desired delay or for the desired output.
2. Some common timer/counter/output functions which microprocessors require are: real time clock, event counter, digital one-shot, programmable rate generator, square wave generator, binary rate multiplier, complex wave form generator, and complex motor control.
3. The 8253/54 includes three identical 16 bit counters that can operate independently.
4. 8254 can handle inputs from DC to 10 MHz (5MHz 8254-5 8MHz 8254 10MHz 8254-2) where as 8253 can operate upto 2.6 MHz.
5. Three counters are identical presettable, and can be programmed for either binary or BCD count.

6. There are six modes on which Intel 8254 operates. They are: Mode 0:interrupt or terminal count, Mode 1:Rate generator, Mode 3:square wave generator, Mode 4:software triggered strobe, Mode 5:hardware triggered strobe.
7. Compatible with all Intel and most other microprocessors.
8. 8254 has powerful command called READ BACK command which allows the user to check the count value, programmed mode and current mode and current status of the counter.

11.6 Check Your Progress

1. The number of counters that are present in the programmable timer device 8254 is:
    a. 1
    b. 2
    c. 3
    d. 4
2. The operation that can be performed on control word register is
    a. read operation
    b. write operation
    c. read and write operations
    d. none
3. The mode that is used to interrupt the processor by setting a suitable terminal count is
    a. mode 0
    b. mode 1
    c. mode 2
    d. mode 3
4. The generation of square wave is possible in the mode
    a. mode 1
    b. mode 2
    c. mode 3
    d. mode 4
5. If BCD=0, then the operation is
    a. decimal count

b. hexadecimal count

c. binary count

d. octal count

6. If the programmable counter timer 8254 is set in mode 1 and is to be used to count six events, the output will remain at logic 0 for _____ number of counts

   a. 5

   b. 6

   c. 0

   d. All of the above

11.7 Answers to Check Your Progress

1. C
2. B
3. A
4. C
5. B
6. B

11.8 Model Questions

1. What is the function of 8254 Programmable Interval Timer? Discuss any one of its applications in detail.
2. Draw the pin of Intel 8554 and define the pin functions.
3. Define and explain the Control word of Intel 8254.
4. Explain with proper diagram all the six modes of operation of programmable interval timer 8254.

PROGRAMMABLE INTERRUPT CONTROLLER 8259A

12.0 Learning Objectives

After going through this unit, you will be able to:

- Understand the functioning of Programmable Interrupt Controller
- Know the functional block diagram of Intel 8259A
- Know the various operating modes of Intel 8259A
- Understand the interfacing of Intel 8259A with 8085 Microprocessor

12.1 Introduction

The Programmable Interrupt Controller(PIC) functions as an overall manager in an interrupt-driven system environment. It accepts requests from the peripheral equipment, determines which of the incoming requests is of the highest importance(priority), ascertains whether the incoming request has the higher priority value than the level currently being serviced, and issue an interrupt to the CPU based on this determination.

12.2 Intel 8259

Intel 8259A is a Programmable Interrupt Controller from Intel specially programmed to work with either 8085 or 8086 processor. It manages 8-interrupts according to the instructions written into its control registers. In 8085 Microprocessor, the interrupt vector address is programmable. The priorities of the interrupts are programmable. The interrupts can be masked or unmasked individually. The 8259s can be cascaded to accept a maximum of 64 interrupts.

12.2.1 Functional block diagram of Intel 8059A

It has eight functional blocks. They are,

1. Control logic

2. Read Write logic
3. Data bus buffer
4. Interrupt Request Register (IRR)
5. In-Service Register (ISR)
6. Interrupt Mask Register (IMR)
7. Priority Resolver (PR)
8. Cascade buffer.

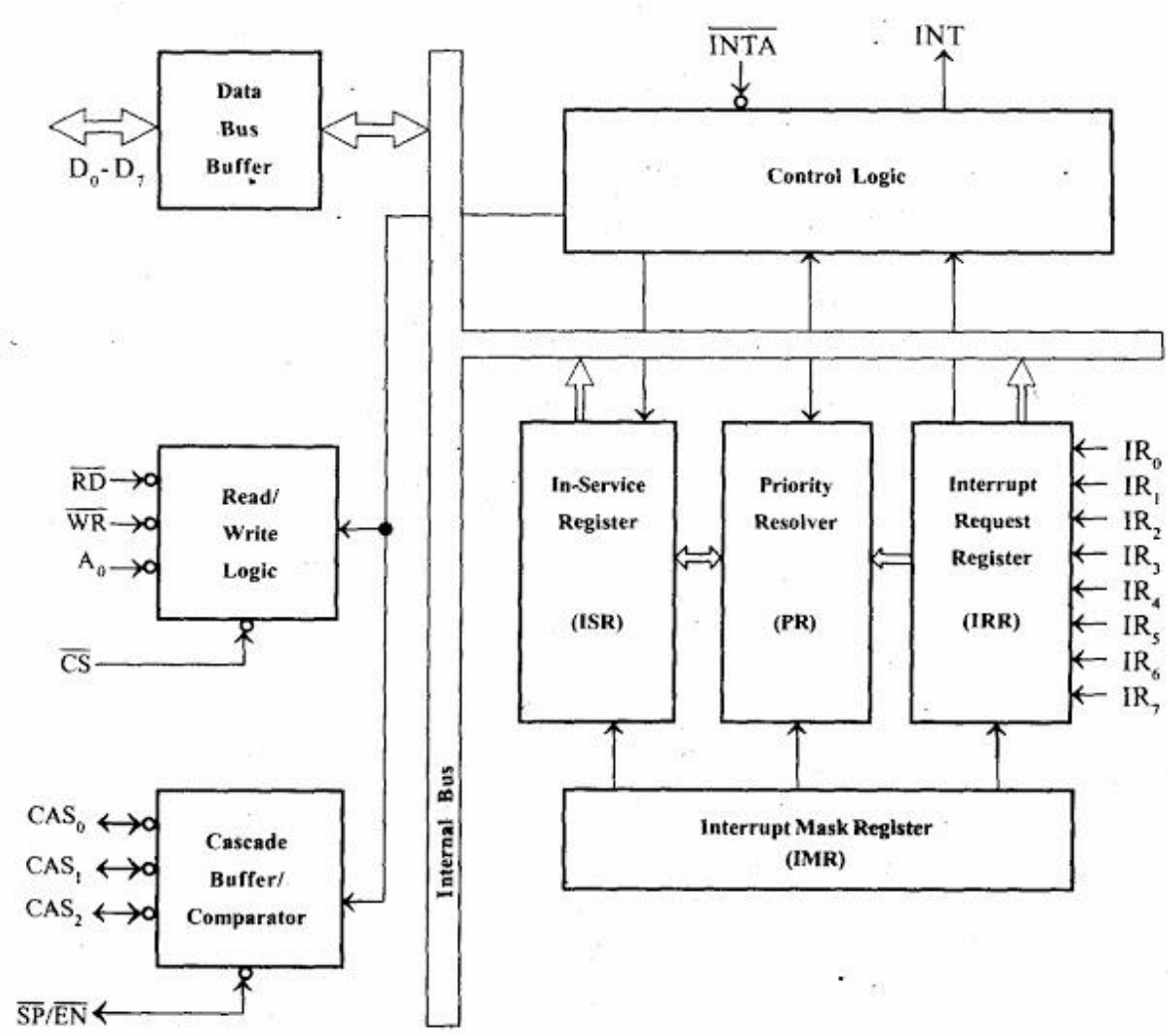


Figure 82: Functional block diagram of Intel 8259A

The data bus and its buffer are used for the following activities:

- The processor sends control word to data bus buffer through D0-D7.
- The processor read status word from data bus buffer through D0-D7.

From the data bus buffer the 8259 send the call opcode and address through D0-D7 to the processor.

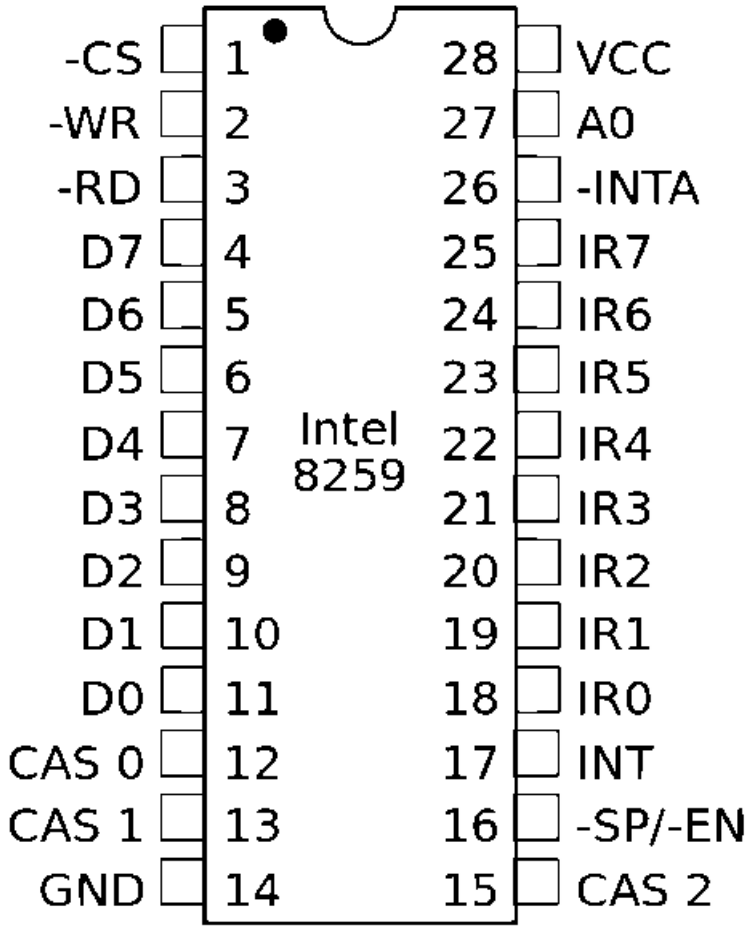


Figure 83: Intel 8259 pin diagram[15]

[15] http://www.wikiwand.com/en/Intel_8259

12.2.2 Functional Description

The main signal pins on an 8259 are as follows: eight interrupt input request lines named IRQ0 through IRQ7, an interrupt request output line named INTR, interrupt acknowledgment line named INTA, D0 through D7 for communicating the interrupt level or vector offset. Other connections include CAS0 through CAS2 for cascading between 8259s.

Up to eight slave 8259s may be cascaded to a master 8259 to provide up to 64 IRQs. 8259s are cascaded by connecting the INT line of one slave 8259 to the IRQ line of one master 8259.

There are three registers, an Interrupt Mask Register (IMR), an Interrupt Request Register (IRR), and an In-Service Register (ISR). The IRR maintains a mask of the current interrupts that are pending acknowledgement, the ISR maintains a mask of the interrupts that are pending an EOI, and the IMR maintains a mask of interrupts that should not be sent an acknowledgement.

End Of Interrupt (EOI) operations support specific EOI, non-specific EOI, and auto-EOI. A specific EOI specifies the IRQ level it is acknowledging in the ISR. A non-specific EOI resets the IRQ level in the ISR. Auto-EOI resets the IRQ level in the ISR immediately after the interrupt is acknowledged.

Edge and level interrupt trigger modes are supported by the 8259A. Fixed priority and rotating priority modes are supported.

The 8259 may be configured to work with an 8080/8085 or an 8086/8088. On the 8086/8088, the interrupt controller will provide an interrupt number on the data bus when an interrupt occurs. The interrupt cycle of the 8080/8085 will issue three bytes on the data bus (corresponding to a CALL instruction in the 8080/8085 instruction set).

The 8259A provides additional functionality compared to the 8259 (in particular buffered mode and level-triggered mode) and is upward compatible with it.

The processor uses the RD (low), WR (low) and A0 to read or write 8259. The 8259 is selected by CS (low).The IRR has eight input lines (IR0-IR7) for interrupts. When these lines go high, the request is stored in IRR. It registers a request only if the interrupt is unmasked.

Normally IR0 has highest priority and IR7 has the lowest priority. The priorities of the interrupt request input are also programmable.

First the 8259 should be programmed by sending Initialization Command Word (ICW) and Operational Command Word (OCW). These command words will inform 8259 about the following:

- Type of interrupt signal (Level triggered / Edge triggered).
- Type of processor (8085/8086).
- Call address and its interval (4 or 8)
- Masking of interrupts.
- Priority of interrupts.
- Type of end of interrupts.

The interrupt mask register (IMR) stores the masking bits of the interrupt lines to be masked. The relevant information is send by the processor through OCW. The in-service register keeps track of which interrupt is currently being serviced. The priority resolver examines the interrupt request, mask and in-service registers and determines whether INT signal should be sent to the processor or not.

12.3 Programming the 8259A

The Intel 8259A accepts two types of command words generated by the CPU:

1. Initialization Command Words(ICWs): Before normal operation can begin, each 8259A in the system must be brought to a starting point by a Sequence of 2 to 4 bytes timed by WR pulses.
2. Operation Command Words(OCWs): These are the command words which command the 8259A to operate in various interrupt modes. The OCWs can be written into 8259A anytime after initialization. The various operational modes of 8259A are:
    a. Fully nested mode
    b. Rotating priority mode
    c. Special mask mode

d. Polled mode

12.3.1 Initialization Command Word(ICW) Format

The 8259A is initialized by a sequence of initialization command words (ICWs). When a byte is issued with A0=0 and bit 4 (D4) = 1, it is interrupted as initialization command word 1, ICW1. ICW1 starts the initialization sequence during which the following automatically occur.

a. The edge sense circuit is reset, which means that following initialization, an interrupt request (IR) input must make a low to high transition to generate an interrupt.
b. The interrupt mask register (IMR) is cleared.
c. IR7 input is assigned priority 7.
d. The slave mode address is set to 7.
e. Special mask mode is cleared and status read is set to IRR.

Initialization Control Word 1 (ICW1)

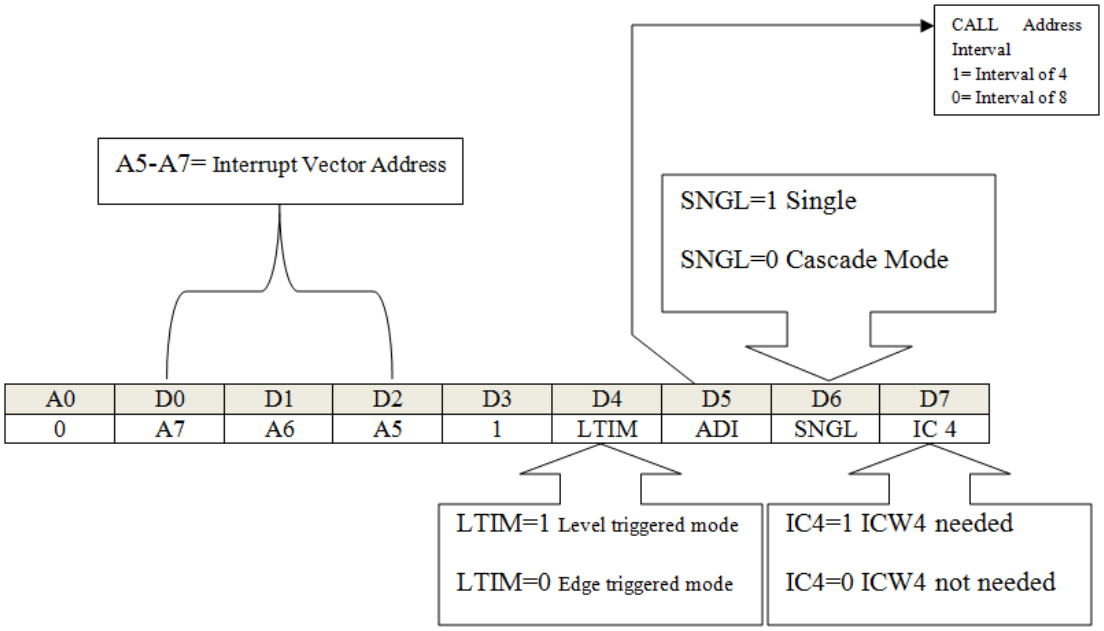


Figure 84: ICW1 Format

Initialization Control Word 2 (ICW2)

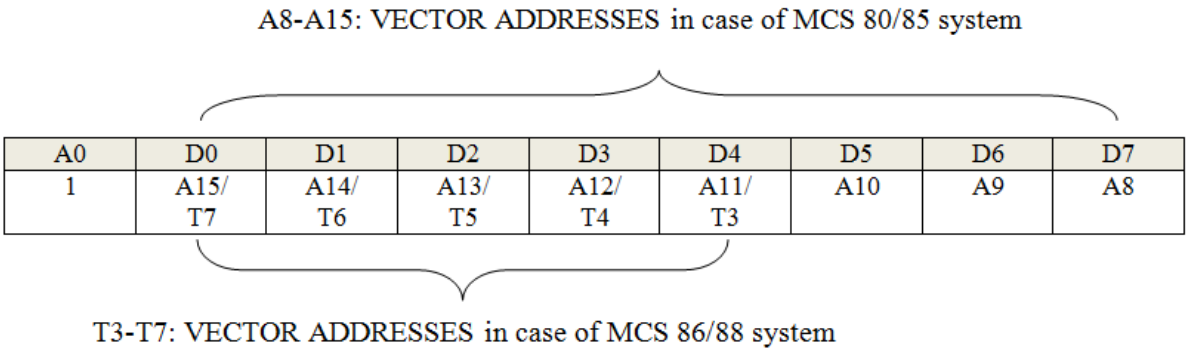


Figure 85: ICW2 format

Initialization Control Word 3 (ICW3: MASTER MODE)

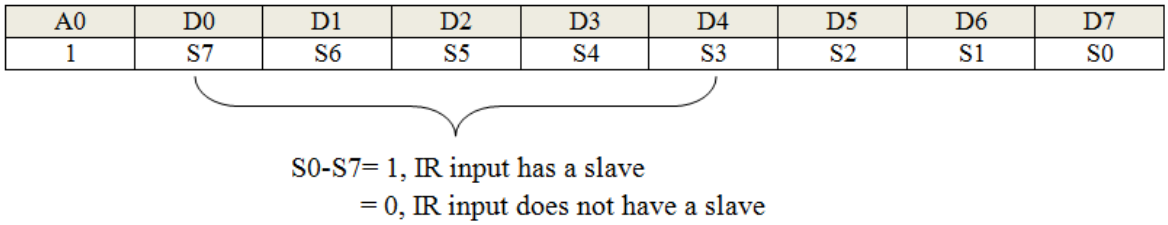


Figure 86: ICW3(Master mode) format

Initialization Control Word 3 (ICW3: SLAVE MODE)

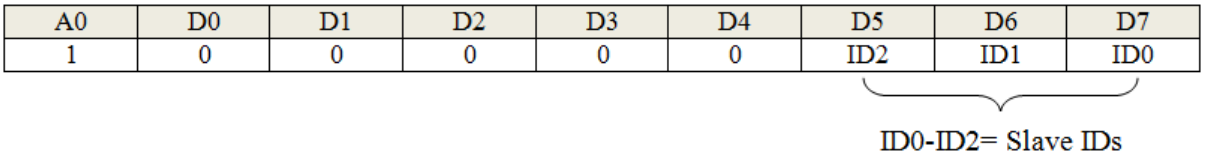


Figure 87: ICD3(Slave mode) format

12.3.2 Operation Command Words(OCWs)

These are the command words which command the 8259A to operate in various interrupt modes. The OCWs can be written into 8259A anytime after initialization. The various operational modes of 8259A are:

a. Fully nested mode
b. Rotating priority mode
c. Special mask mode
d. Polled mode

Operation Command Word1 (OCW1)

A0	D0	D1	D2	D3	D4	D5	D6	D7
1	M7	M6	M5	M4	M3	M2	M1	M0

Interrupt Mask=1 Mask set
=0 Mask Reset

Figure 88: OCW1 format

Operation Command Word2 (OCW2)

A0	D7	D6	D2	D3	D4	D5	D6	D7
0	R	SL	EOI	0	0	L2	L1	L0

0	0	1	Non- Specific EOI Command
0	1	1	Specific EOI Command
1	0	1	Rotate on Non- Specific EOI Command
1	0	0	Rotate on automatic EOI mode(Set)
0	0	0	Rotate on automatic EOI mode(Clear)
1	1	1	Rotate on Specific EOI mode command
1	1	0	Set priority Command
0	1	0	No operation

L0-L2=IR Level to be acted upon

Figure 89:OCW2 format

Operation Command Word3 (OCW3)

P=1 Poll Command
=0 No poll Command

A0	D0	D1	D2	D3	D4	D5	D6	D7
0	0	ESMM	SMM	0	1	P	RR	RIS
	No action	0	0			No action	0	0
	No action	0	1			No action	0	1
	Reset Special Mask	1	0			Read IR on next $\overline{RD}$ pulse	1	0
	Set Special Mask	1	1			Read IR on next $\overline{RD}$ pulse	1	1

Figure 90: OCW3 format

ESMM(Enable Special Mask Mode)- When this bit is set to 1 it enables the SMM bit to set or reset the Special Mask Mode. When ESMM is set to 0 the SMM bit becomes a “don’t care”.

SMM(Special Mask Mode): If ESMM =1 and SMM=1 the 8259A will enter Special Mask Mode. If ESMM=1 and SMM=0 the 8259A will revert to normal mask mode. When ESMM=0, SMM has no effect.

12.4 Fully Nested Mode

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority from 0 through 7(0 highest). When an interrupt is acknowledged the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service Register(ISO-7) is set. This bit remains set unit the microprocessor issues an End of Interrupt(EOI) command immediately before returning from the service routine, or if AEOI(Automatic End of Interrupt) bit is set, until the trailing edge of the last INTA. While the IS bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels will generate an interrupt (which will be acknowledged only if the microprocessor internal interrupt enable flip-flop has been re-enabled through software). After the initialization sequence, IR0 has the highest priority and IR7 the lowest.

End of Interrupt

The In Service(IS) bit can be reset either automatically following the trailing edge of the last in sequence $\overline{INTA}$ pulse (when AEOI bit in ICW1 is set) or by a command word that must be issued to the 8259A before returning from a service routine (EOI command). An EOI command must be issued twice if in the Cascade mode, once for the master and once for the corresponding slave. There are two forms of EOI command: Specific and Non-specific. When the 8259A is operated in modes which preserves the fully nested structure, it can determine which IS bit to reset on EOI. When a Non-Specific EOI command is issued the 8059A will automatically reset the highest IS bit of those that are set, since in the fully nested mode the highest IS level was necessarily the last level acknowledged and serviced. A non-specific EOI can be issued with OCW2(EOI=1, SL=0, R=0).

When a mode is used which may disturb the fully nested structure, the 8259A may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt(EOI) must be issued which includes as part of the command the IS level to be reset. A specific EOI can be issued with OCW2(EOI=1, SL=1, R=0 and L0-L2 is the binary level of the IS bit to be reset). It should be noted that an IS bit that is masked by an IMR bit will not be cleared by a non-specific EOI if the 8259A is in the Special Mask Mode.

12.5 Automatic End of Interrupt(AEOI) Mode

If AEOI=1 in ICW4, then the 8059A will operate in AEOI mode continuously until reprogrammed by ICW4, in this mode the 8259A will automatically perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse.

Note that form a system standpoint, this mode should be sued only when a nested multilevel interrupt structure is not required within a single 8259A. The AEOI mode can only be used in a master 8259A and not a slave. 8259As with a copyright date of 1985 or later wil operate in the AEOI mode as a master or a slave.

Automatic Rotation

In some applications there are a number of interrupting devices of equal priority. In this mode a device, after being serviced, receives the lowest priority, soa device requesting an interrupt will

have to wait, in the worst case until each of the 7 other devices are serviced at most once. For example, if the priority and " in service" status is:

Before rotate (IR4 the highest priority requiring service):

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0	
0	1	0	0	0	0	0	0	IS status
7	6	5	4	3	2	1	0	Priority Status

After rotate (IR4 the highest priority requiring service):

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0	
0	1	0	0	0	0	0	0	IS status
2	1	0	7	6	5	4	3	Priority Status

Specific Rotation(Specific Priority)

The programmer can change priorities y programming the bottom ppriority a d thus fixing all other priorities i.e. if IR 5 is programmed as the bottom priority device, then IR6 will have the highest one. The Set Priority command is issued in OCW2 where: R=1, SL=1, L0-L2 is the binary priority level code of the priority device. However, it is independent of the End of Interrupt(EOI) command(also executed by OCW2). Priority changes can be executed during an EOI command by using the Rotate on Specific EOI command in OCW2 (R=1, SL-1, EOI=1 and L0-L2 IR level to receive bottom priority).

Interrupt mask

Each Interrupt Request input can be masked individually by the Interrupt Mask Register programmed through OCW1. Each bit in the IMR register masks one interrupt channel if ti is set(1). Bit 0 masks IR0, Bit 1 masks IR1 and so forth. Masking an IR channel does not affects the other channels operation.

12.6 Special mask Mode

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish

to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The difficulty here is that if an Interrupt Request is acknowledged and ab End of Interrupt command did not reset its IS bit (i.e. while executing a service routine), the 8259A would have inhibited all lower priority requests with no easy way for the routine to enable them. That is where the Special mask Mode comes in. In the special Mask Mode, when a mask bit is set in OCW1, it inhibits further interrupts at that level and enables interrupt from all other levels(lower as well as higher) that are not masked. Thus, any interrupts may be selectively enabled by loading the mask register.

The special Mask Mode is set by OCW3 where: SSMM=1, SMM=1 and cleared SSMM=1 , SMM=0.

12.7 Poll Command

In Poll mode the INT output functions as it normally does. The microprocessor should ignore this output. This can be accomplished either by not connecting the INT output or by masking interrupts within the microprocessor, thereby disabling its interrupt input. The Poll command is issued by setting P=1 in OCW3. The 8259A treats the next $\overline{RD}$ pulse to the 8259A (i.e. RD is 0 and CS is 0) as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupt is frozen from $\overline{WR}$ to $\overline{RD}$. The word enabled onto the data bus during $\overline{RD}$ is:

D7	D6	D5	D4	D3	D2	D1	D0
1	-	-	-	-	W2	W1	W0

W0-W2 is binary code of the highest priority level requesting service and I is equal to 1 if there is an interrupt. This mode is useful if there is a routine command common to several levels so that the $\overline{INTA}$ sequence is not needed (saves ROM space). Another application is to use the poll mode to expand the number of priority levels to more than 64.

12.8 Cascade Mode

The 8259A can easily interconnected in a system of one master with up to eight slaves to handle up to 64 priority levels. The master controls the slaves through the 3 line cascade bus. The cascade bus acts like chip selects to the slaves during the INTA sequence. In a cascade configuration, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master will enable the corresponding slave to release the device routine address during bytes 2 and 3 of INTA.

The cascade bus lines are normally low and will contain the slave address code from the trailing edge of the first INTA pulse to the trailing edge of the third pulse. Each 8259A in the system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI command must be issued twice, once for the master and second for the corresponding slave. An address decoder is required to activate the Chip Select($\overline{CS}$) input of each 8259A. The cascade lines of the master 8259A are activated only for salve input, non-slave inputs leave the cascade line inactive(low).

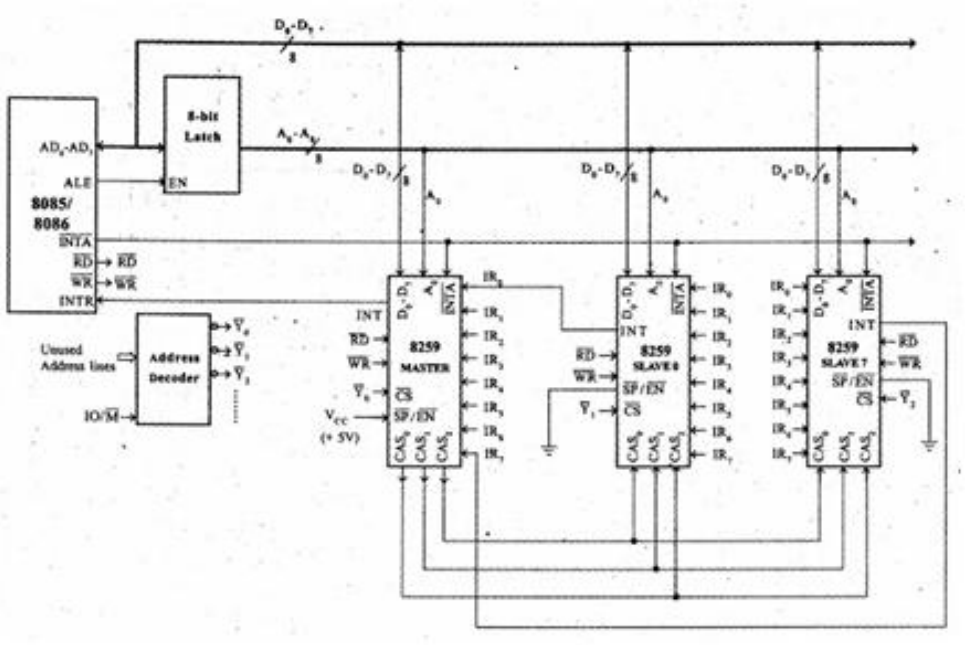


Figure 91: Cascade Connection of 8259[16]

[16] Image adopted from https://mediatoget.blogspot.in/2013/02/interfacing-8259-with-8085.html

12.9 Interfacing 8259 with 8085 Microprocessor

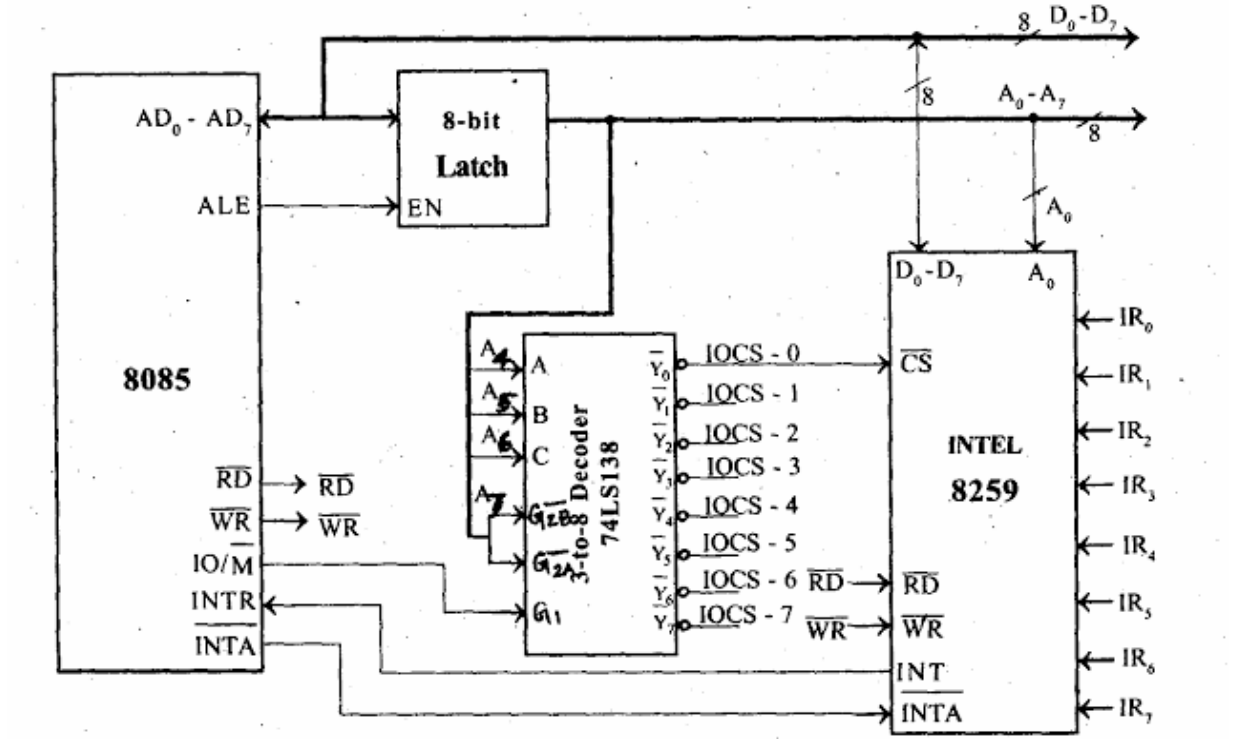


Figure 92: Interfacing 8259 with 8085[17]

It requires two internal address and they are A =0 or A = 1. It can be either memory mapped or I/O mapped in the system. The interfacing of 8259 to 8085 is shown in figure is I/O mapped in the system. The low order data bus lines D0-D7 are connected to D0-D7 of 8259. The address line A0 of the 8085 processor is connected to A0 of 8259 to provide the internal address. The 8259 require one chip select signal. Using 3-to-8 decoder generates the chip select signal for 8259.

The address lines A4, A5 and A6 are used as input to decoder. The control signal IO/M (low) is used as logic high enables for decoder and the address line A7 is used as logic low enable for decoder. The I/O addresses of 8259 are shown in Table 9.

[17] Image adopted from https://mediatoget.blogspot.in/2013/02/interfacing-8259-with-8085.html

Table 9: The I/O addresses of 8259

	Binary Address		Hexa Address
	Decoder input/enable	**Input to address pin of 8259**	
	A_7 A_6 A_5	A_4 A_3 A_2 A_1 A_0	00
For A_0 of 8059 to be zero	0 0 0	0 x x x x 0	01
For A_0 of 8259 to be one	0 0 0	0 x x x x 1	

12.9.1 Working of 8259 with 8085 processor

First the 8259 should be programmed by sending Initialization Command Word (ICW) and Operational Command Word (OCW). These command words will inform 8259 about the following:

- Type of interrupt signal (Level triggered / Edge triggered).
- Type of processor (8085/8086).
- Call address and its interval (4 or 8)
- Masking of interrupts.
- Priority of interrupts.
- Type of end of interrupts.

Once 8259 is programmed it is ready for accepting interrupt signal. When it receives an interrupt through any one of the interrupt lines IR0-IR7 it checks for its priority and also checks whether it is masked or not. If the previous interrupt is completed and if the current request has highest priority and unmasked, then it is serviced. For servicing this interrupt the 8259 will send INT signal to INTR pin of 8085. In response it expects an acknowledge INTA (low) from the processor. When the processor accepts the interrupt, it sends three INTA (low) one by one. In response to first, second and third INTA (low) signals, the 8259 will supply CALL opcode, low byte of call address and high byte of call address respectively. Once the processor receives the

call opcode and its address, it saves the content of program counter (PC) in stack and load the CALL address in PC and start executing the interrupt service routine stored in this call address.

12.10 Summary

5. The Intel 8259A Programmable Interrupt Controller handles up to eight vectored priority interrupts for the CPU.
6. The Intel 8259A Programmable Interrupt Controller handles up to eight vectored priority interrupts for the CPU.
7. It is cascadable for up to 64 vectored priority interrupts without additional circuitry.
8. It is packaged in a 28-pin DIP, uses NMOS technology and requires a single a5V supply.
9. Circuitry is static, requiring no clock input.
10. The 8259A is designed to minimize the software and real time overhead in handling multi-level priority interrupts.
11. It has several modes, permitting optimization for a variety of system requirements.
12. The Programmable Interrupt Controller (PIC) functions as an overall manager in an Interrupt-Driven system environment.
13. It accepts requests from the peripheral equipment, determines which of the in- coming requests is of the highest importance (priority), ascertains whether the incoming request has a higher priority value than the level currently being serviced, and issues an interrupt to the CPU based on this determination.
14. The 8259A is a device specifically designed for use in real time, interrupt driven microcomputer systems.
15. It manages eight levels or requests and has built-in features for expandability to other 8259A's (up to 64 levels).
16. It is programmed by the system's software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by the 8259A can be configured to match his system requirements.
17. The priority modes can be changed or reconfigured dynamically at any time during the main program. This means that the complete interrupt structure can be defined as required, based on the total system environment.

12.11 Check Your Progress

1. The register that stores all the interrupt requests in it in order to serve them one by one on priority basis is:
    a. Interrupt Request Register
    b. In-Service Register
    c. Priority resolver
    d. Interrupt Mask Register
2. The register that stores the bits required to mask the interrupt inputs is
    a. In-service register
    b. Priority resolver
    c. Interrupt Mask register
    d. None
3. The interrupt control logic
    a. manages interrupts
    b. manages interrupt acknowledge signals
    c. accepts interrupt acknowledge signal
    d. all of the mentioned
4. In cascaded mode, the number of vectored interrupts provided by 8259A is
    a. 4
    b. 8
    c. 16
    d. 64
5. When non-specific EOI command is issued to 8259A it will automatically
    a. set the ISR
    b. reset the ISR
    c. set the INTR
    d. reset the INTR
6. In the application where all the interrupting devices are of equal priority, the mode used is:
    a. automatic rotation
    b. automatic EOI mode

c. specific rotation

d. EOI

12.12 Answers to Check Your Progress

1. A
2. C
3. D
4. D
5. B
6. A

12.13 Model Questions

1. What are the functions performed by 8259A?
2. How many interrupts can 8259A handle?
3. How many 8259A are required to handle 64 interrupts?
4. What is the purpose of IR0-IR7 pins in 8259A?
5. Where is the slave INT pin is connected on the master 8259A in a cascaded system?
6. What is an ICW?
7. What is an OCW?

CHAPTER 13: APPLICATIONS OF MICROPROCESSOR

13.0 Learning Objectives

After reading this unit, you will be able to:

- Understand the various applications of Intel 8085 Microprocessor
- Implement Microprocessor based stepper motor control system using 8085
- Implement traffic light controller using 8085
- Implement ADC interface using 8085
- Implement DAC interface using 8085

13.1 Introduction

The impact of microprocessor in different lures of fields is significant. The availability of low cost, low power and small weight, computing capability makes it useful in different applications. Now-a-days microprocessor based systems are used in instructions, automatic testing product, speed control of motors, traffic light control, light control of furnaces etc. Some of the important applications are discussed in the forthcoming sections,

13.2 8085 Microprocessor Based Stepper Motor Control System

The motor is controlled by ON/OFF the control winding[18]. The popular stepper motor used for demonstration in laboratories has a step size of 1.8° (i.e., 200 steps per revolution). The basic step size of the motor is called full-step. By altering the switching sequence, the motor can be made to run with incremental motion of half the full-step value. The switching sequence for half step rotation is shown.

[18] https://mediatoget.blogspot.in/2013/02/8085-microprocessor-based-stepper-motor.html

Table 10: Switching sequence for full step rotation

Switching Sequence	Clockwise Roatation				Anticlockwise Roatation			
	PA_3	PA_2	PA_1	PA_0	PA_3	PA_2	PA_1	PA_0
Sequence-1	1	1	0	0	0	0	1	1
Sequence-2	0	1	1	0	0	1	1	0
Sequence-3	0	0	1	1	1	1	0	0
Sequence-4	1	0	0	1	1	0	0	1

Table 11: Switching sequence for half step rotation

Clockwise Roatation				Anticlockwise Roatation			
PA_3	PA_2	PA_1	PA_0	PA_3	PA_2	PA_1	PA_0
1	1	0	0	0	0	1	1
0	1	0	0	0	0	1	0
0	1	1	0	0	1	1	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0
1	0	0	1	1	0	0	1
1	0	0	0	0	0	0	1

A two-phase or four winding stepper motor is shown. The system consist of 8085 microprocessor as CPU, EPROM and RAM memory for program & data storage and for stack.

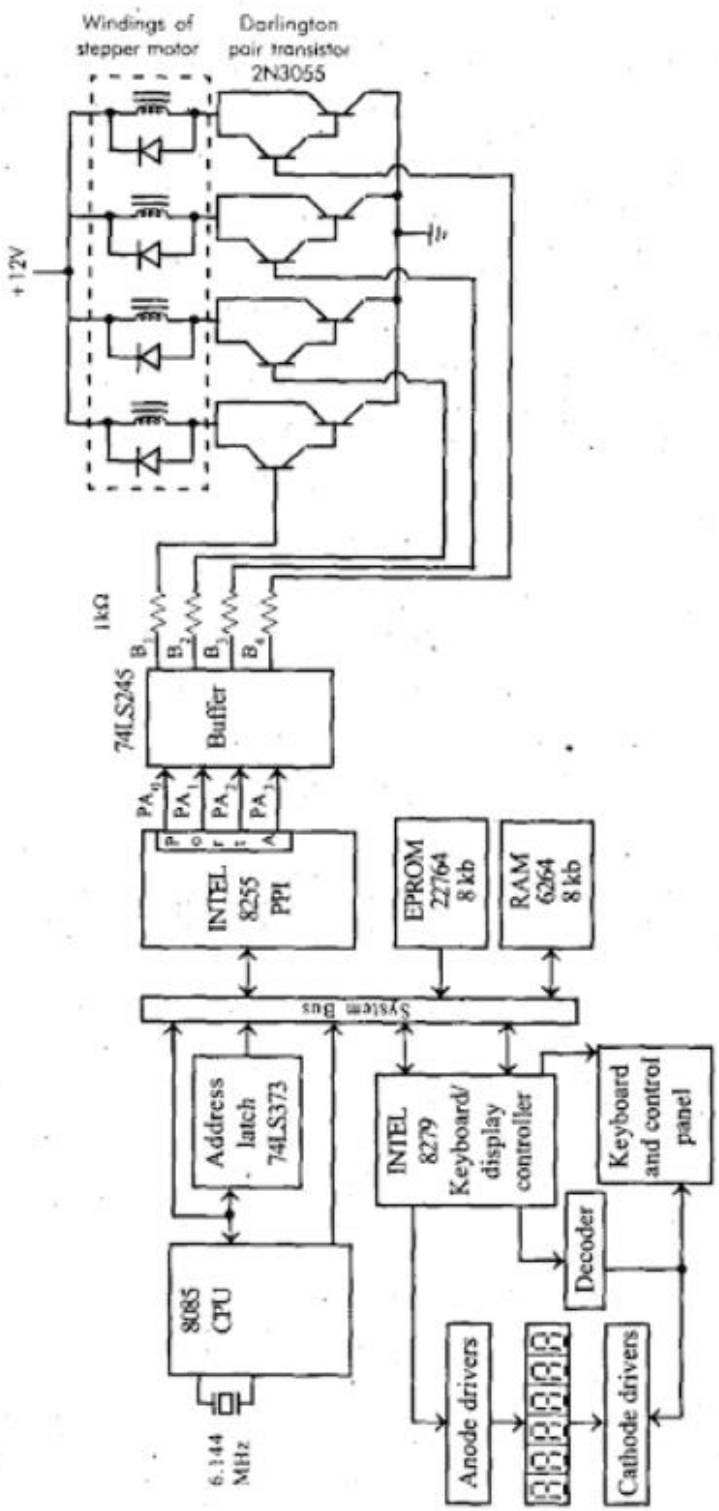


Figure 93 : 8085 microprocessor based stepper motor control system[19]

[19] Figure adopted from https://mediatoget.blogspot.in/2013/02/8085-microprocessor-based-stepper-motor.html

Using INTEL 8279, a keyboard and six number of 7-segment LED display have been interfaced in the system. Through the keyboard the operator can issue commands to control the system. The LED display has been provided to display messages to the operator. The windings of stepper motor are connected to the collector of Darlington pair transistors. The transistors are switched ON/OFF by the microprocessor through the ports of 8255 and buffer (74LS245). A freewheeling diode is connected across each winding for fast switching.

The flowchart for the operational flow of the stepper motor control system is shown.

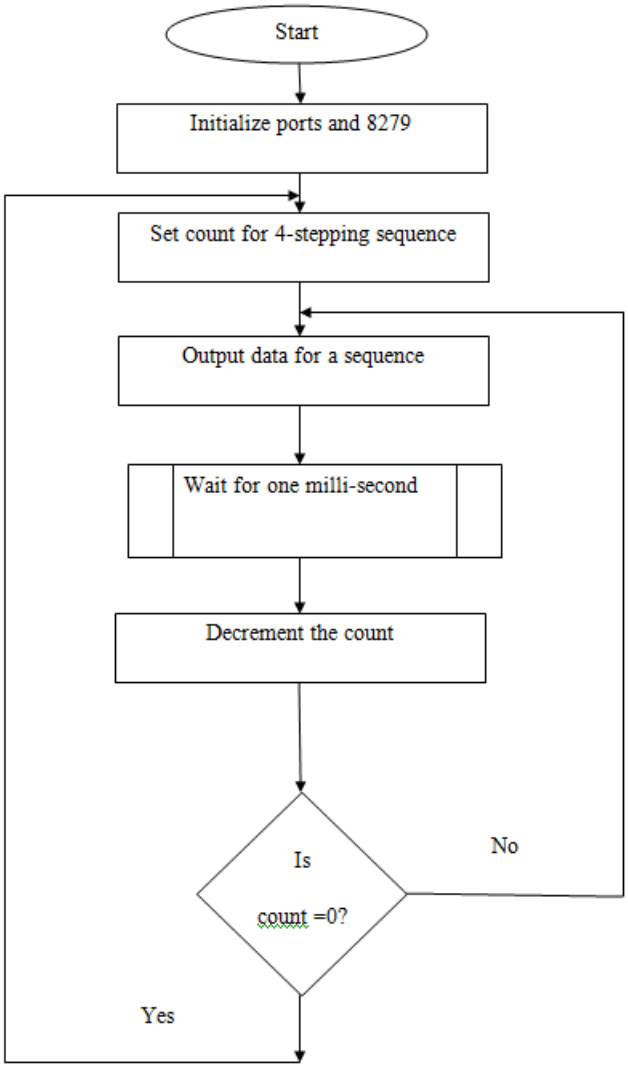


Figure 94: Operational flow of the stepper motor control system

13.3 Keyboard and Display Interface

In a microprocessor based system, when keyboard and 7-segment LED display is interfaced using ports or latches then the processor has to carry the following task[20].

- Keyboard scanning
- Key debouncing
- Key code generation
- Sending display code to LED
- Display refreshing

A typical Hexa keyboard and 7-segment LED display interfacing circuit using 8279 is shown.

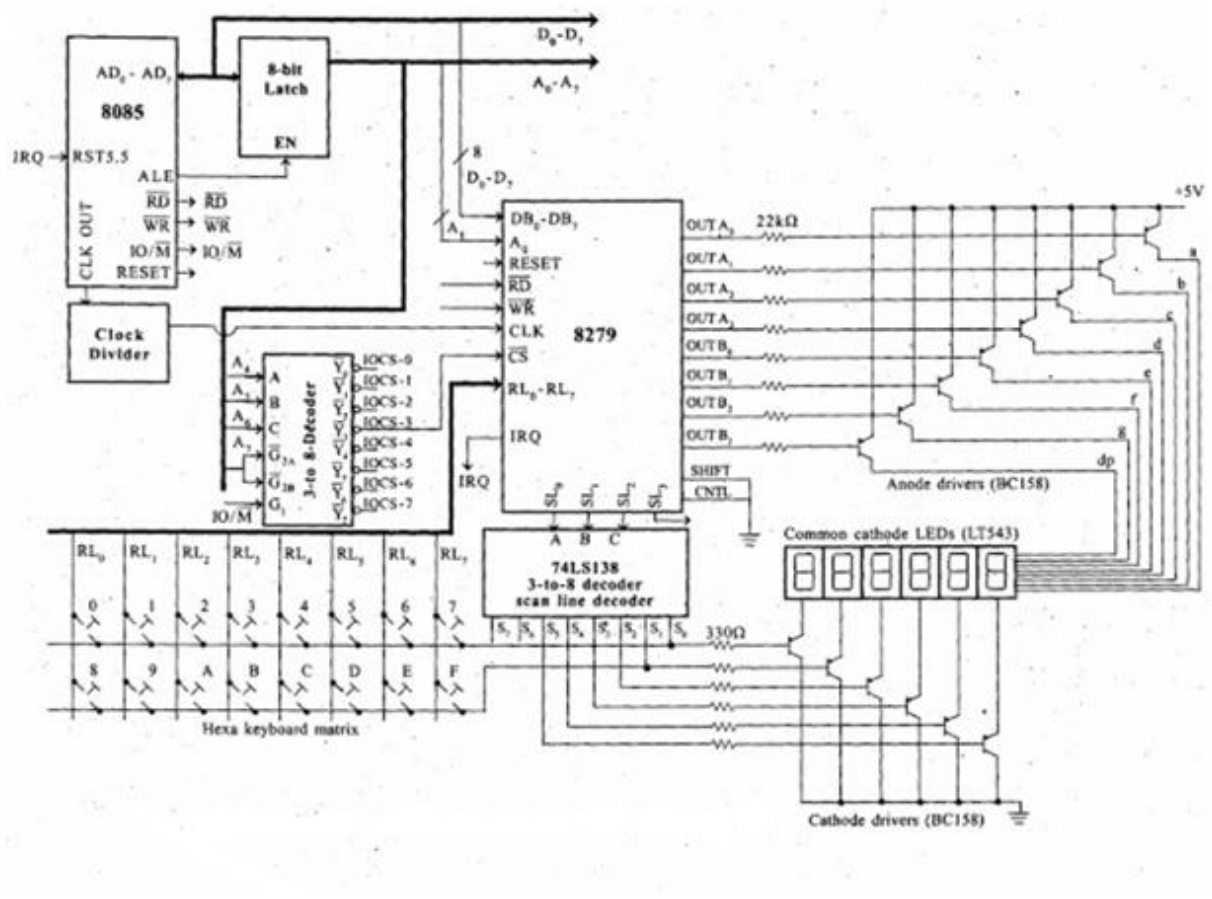


Figure 95: Keyboard and display interface

[20] https://mediatoget.blogspot.in/2012/12/keyboard-and-display-interface-using.html

The circuit can be used in 8085 microprocessor system and consist of 16 numbers of hexa-keys and 6 numbers of 7-segment LEDs.

- The 7-segment LEDs can be used to display six digit alphanumeric character.
- The 8279 can be either memory mapped or I/O mapped in the system. In the circuit shown is the 8279 is I/O mapped.
- The address line A0 of the system is used as A0 of 8279.
- The clock signal for 8279 is obtained by dividing the output clock signal of 8085 by a clock divider circuit.
- The chip select signal is obtained from the I/O address decoder of the 8085 system. The chip select signals for I/O mapped devices are generated by using a 3-to-8 decoder.
- The address lines A4, A5 and A6 are used as input to decoder.
- The address line A7 and the control signal IO/M (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 are shown in table.

Table 12 : I/O address of the internal devices of 8279

Internal device	**Binary Address**								Hexa Address
	Decoder input and enable				**Input to address of 8279**				
	A_7	**A_6**	**A_5**	**A_4**	**A_3**	**A_2**	**A_1**	**A_0**	
Data register	0	0	1	1	X	X	X	0	30
Control register	0	0	1	1	X	X	X	1	31

- 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-8 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 wait 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.

13.4 Traffic Light Controller

Figure below shows the interfacing diagram to control 12 electric bulbs. Port A is used to control lights on N-S road and Port B is used to control lights on W-E road.

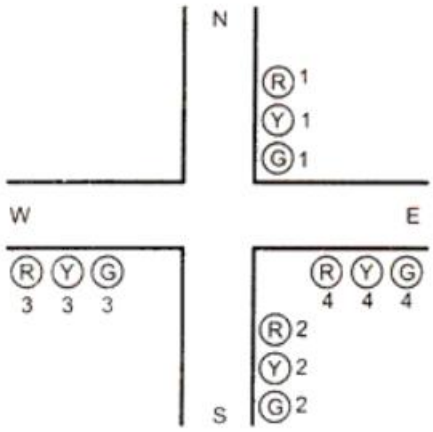


Figure 96 : Interfacing diagram to control 12 electric bulbs of traffic light

Actual pin connections are listed in Table 13 below.

Table 13 : Pin connections of trafic light

Pins	Light	Pins	Light
PA_0	R_1	PB_0	R_3
PA_1	Y_1	PB_1	Y_3
PA_2	G_1	PB_2	G_3
PA_3	R_2	PB_3	R_4
PA_4	Y_2	PB_4	Y_4
PA_5	G_2	PB_5	G_4

The electric bulbs are controlled by relays. The 8255 pins are used to control relay on-off action with the help of relay driver circuits. The driver circuit includes 12 transistors to drive 12 relays. Fig. also shows the interfacing of 8255.

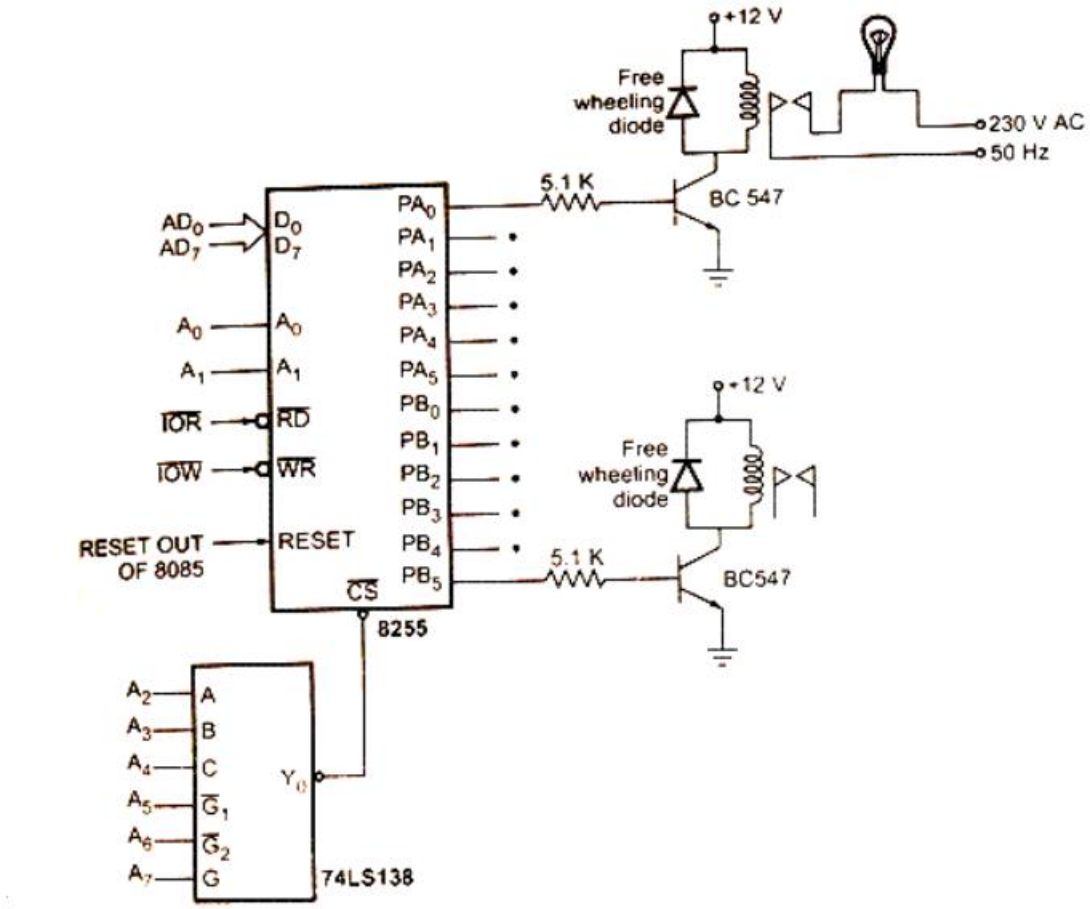


Figure 97 : Interfacing of 8255

13.3.1 I/O Map

Table 14 : I/O map

Ports/ Control register	Address lines								Address
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Port A	1	0	0	0	0	0	0	0	80H
Port B	1	0	0	0	0	0	0	1	81H
Port C	1	0	0	0	0	0	1	0	82H
Control Register	1	0	0	0	0	0	1	1	83H

13.3.2 Control Word for initialization of 8255

BSR/IO	MODE A		P_A	PC_H	MODE B	P_B	PC_L
1	0	0	0	X	0	0	X

=80H

To glow	PB_7	PB_6	PB_5	PB_4	PB_3	PB_2	PB_1	PB_0	PA_7	PA_6	PA_5	PA_4	PA_3	PA_2	PA_1	A_0	Port B Output	Port A Output
R_1,R_2,G_3 and G_4	X	X	1	0	0	1	0	0	X	X	0	0	1	0	0	1	24H	09H
Y_1,Y_2,Y_3 and Y_4	X	X	0	1	0	0	1	0	X	X	0	1	0	0	1	0	12H	12H
R_3,R_4,G_1 and G_2	X	X	0	0	1	0	0	1	X	X	1	0	0	1	0	0	09H	24H

13.3.3 Source program

```
        MVI A, 80H          : Initialize 8255, port A and port B
        OUT 83H (CR)        : in output mode
START:  MVI A, 09H
        OUT 80H (PA)        : Send data on PA to glow R1 and R2
        MVI A, 24H
        OUT 81H (PB)        : Send data on PB to glow G3 and G4
        MVI C, 28H          : Load multiplier count (40₁₀) for delay
        CALL DELAY          : Call delay subroutine
        MVI A, 12H
        OUT (81H) PA        : Send data on Port A to glow Y1 and Y2
        OUT (81H) PB        : Send data on port B to glow Y3 and Y4
        MVI C, 0AH          : Load multiplier count (10₁₀) for delay
CALL:   DELAY               : Call delay subroutine
```

```
    MVI A, 24H
    OUT (80H) PA          : Send data on port A to glow G1 and G2
    MVI A, 09H
    OUT (81H) PB          : Send data on port B to glow R3 and R4
    MVI C, 28H            : Load multiplier count (40₁₀) for delay
    CALL DELAY            : Call delay subroutine
    MVI A, 12H
    OUT PA                : Send data on port A to glow Y1 and Y2
    OUT PB                : Send data on port B to glow Y3 and Y4
    MVI C, 0AH            : Load multiplier count (10₁₀) for delay
    CALL DELAY            : Call delay subroutine
    JMP START
```

Delay Subroutine:

```
DELAY: LXI D, Count       : Load count to give 0.5 sec delay
BACK: DCX D               : Decrement counter
    MOV A, D
    ORA E                 : Check whether count is 0
    JNZ BACK              : If not zero, repeat
    DCR C                 : Check if multiplier zero, otherwise repeat
    JNZ DELAY
    RET                   : Return to main program
```

Assume Operating Frequency= 2 MHz

Time for one T-state= $\frac{1}{2MHz}$ = 0.5 μsec

Count= $\frac{Required\ Delay}{Time\ required\ for\ 1\ loop}$

=$\frac{0.5\ Sec}{0.5\mu sec x 24}$, since 1 loop needs 24 T-states

=14666_{10}=A2C2H

13.4 ADC INTERFACE

In many applications, an analog device has to be interfaced to digital system. But the digital devices cannot accept the analog signals directly and so the analog signals are converted into equivalent digital signal (data) using Analog-to-Digital Converter (ADC).

The analog to digital (A/D) conversion is the reverse process of digital to analog (D/A) conversion. The A/D conversion is also called Quantization, in which the analog signal is represented binary data. The analog signals varies continuously and defined for any interval of time. The digital signals (or data) can take only finite values for discrete instant of time. If the digital data is represented by n-bit binary then it can have 2n different values. The given analog signal has to divided into steps of 2n values, and each step is represented by one of the 2n values. The Analog to Digital Converters can be classified into two groups based on the technique involved for conversion.

The **first group** includes *successive-approximation, counter and flash type converters.* The technique involved in these devices is that the given analog signal is compared with internally generated analog signal.

The **second group** includes *integrator converters and voltage to frequency converters.* In the devices of second group, the given analog signal is converted to time or frequency and the new parameters (time or frequency) is compared with known values to produce digital signal.

The trade-off between the two techniques is based on Accuracy Vs Speed.

The successive approximation and flash type converters are accurate than the integrator and the voltage-to-frequency converters. Also, flash type is costlier. The successive-approximation type converters are used for high-speed conversion and the integrating type converters are used for high accuracy.

The **resolution** of the converter is the minimum analog value that can be represented by the digital data. If the ADC gives n-bit digital output and the full-scale analog input is X volts, then the resolution is 1/2n x X volts.

In ADC, another critical parameter is **conversion time**. The conversion time is defined as the total time required to convert an analog signal into its digital equivalent. It depends on the conversion technique and the propagation delay in various circuits.

13.4.1 Successive-Approximation ADC

A successive approximation ADC consist of D/A converter, successive approximation register and comparator. The figure below shows the functional blocks of a typical successive approximation A/D converter.

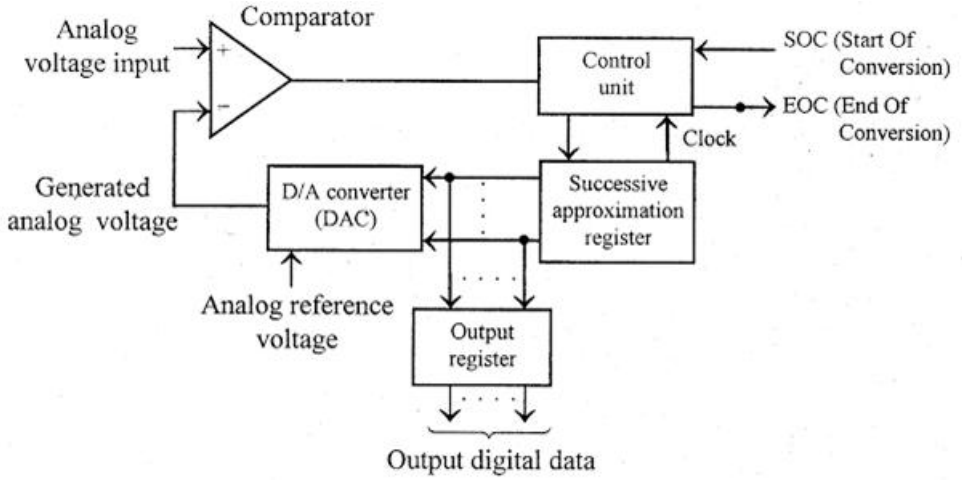


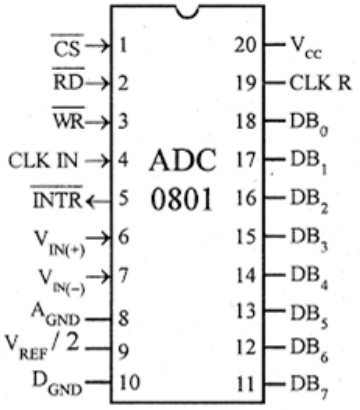
Figure 98: Successive approximation ADC converter

The conversion process is initiated by a start of conversion (SOC) signal from the processor to ADC. On receiving the SOC, the control unit of ADC will give a start command to successive approximation register and it starts generating digital signal by successive approximation method. The generated digital data is converted to analog signal by Dl A converter and then compared with given analog signal. When the analog signals are equal the comparator output informs the control unit to stop generation of digital signal. The digital data available at this instant is given as output through output register. Also the control unit generates a signal to indicate the End of Conversion (EOC) process to the processor.

In this method the MSD (Most Significant Digit) is first set to "1" and all other digits are reset to "0". The analog signal generated for this digital data is compared with given analog signal. (Initially the comparator output will be HIGH. After comparison the output of comparator remains in HIGH state, if the given analog signal is higher than generated analog signal. Otherwise, if the given signal is less than generated signal, then the output of comparator changes from HIGH to LOW state). If the output state of comparator changes then the MSD is reset to "0" otherwise it is retained as '1'. Then the above process is repeated by setting the next higher order bit to '1'. The process is continued for each bit starting from MSD to LSD. (During a process, the higher order bits are the bits determined in earlier steps and the lower order bits are reset to "0"). After one complete cycle through MSD to LSD, the data available on the successive approximation register will be the digital equivalent of the given analog signal.

13.4.2 ADC interfacing to 8085 microprocessor system

The ADC can be interfaced to 8085 microprocessor system through tri-state buffer or port devices such as 8255/8155. The interfacing of ADC0801 is presented in this section. The ADC0801 is a single channel, 8-bit successive approximation type A/D converter from National Semiconductor Corporation. It is a 20-pin IC available in DIP. The pin configuration of ADC080 1 is shown in figure below.



Pin	Description
DB_0 - DB_7	Output digital data
$V_{IN(+)}$	Positive Analog input voltage
$V_{IN(-)}$	Negative Analog input voltage
A_{GND}	Ground for analog circuit
D_{GND}	Ground for digital circuit
$V_{REF}/2$	Half the value of Analog reference voltage
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Control
$\overline{WR}$	Write Control (SOC)
$\overline{INTR}$	Interrupt (EOC)
CLK IN	Clock Input
CLK R	Resistance connection for clock
V_{cc}	Power supply

Figure 99: Pin configuration of ADC 0801

The ADC080 I has two analog inputs V IN(+) and V IN (-) .Both the analog inputs are used for differential mode of operation. When the analog signal is single ended positive, then V IN (+) is used as input and V IN(-) is grounded. When the analog signal is single ended negative, then V IN (-) is used as input and V IN (+) is grounded. The ADC requires an external clock in the frequency range l00 kHz to 800 kHz or the clock can be generated by connecting a RC circuit between pin 4 and 19. Typically, the clock frequency is chosen as 640 kHz to provide a conversion time of l00 μsec. A typical circuit to interface ADC0801 to 8085 processor is shown in figure below.

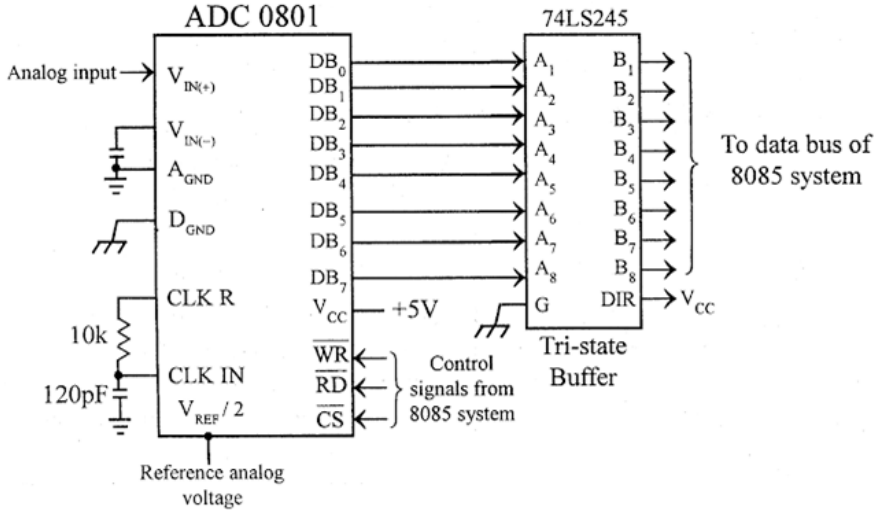


Figure 100: Interfacing ADC 0801 to 8085

The ADC is connected to system bus through tri-state buffer, 74LS245. The ADC can be either memory mapped or I/O mapped in the system. The chip select signal (CS) is obtained from the address decoder of the 8085 system. The conversion is initiated when both CS and WR are asserted LOW. The write control signal (WR) is used to reset the successive approximation register (SAR) of ADC and to give start of conversion (SOC). The WR of ADC can be connected directly to WR of the 8085 processor. At the falling edge of WR, the SAR is resetted and at the rising edge of WR, the conversion starts.

The end of conversion is indicated by asserting INTR LOW and this signal can be inverted to interrupt the 8085 processor. The processor reads the digital data using RD and when the data is read, the DAC will set INTR HIGH.

3.5 DAC INTERFACE

In many applications, the microprocessor has to produce analog signals for controlling certain analog devices. Basically the microprocessor system can produce only digital signals. In order to convert the digital signal to analog signal a Digital-to-Analog Converter. (DAC) has to be employed. The DAC will accept a digital (binary) input and convert to analog voltage or current. Every DAC will have "n" input lines and an analog output.

The DAC require a reference analog voltage (Vref) or current (Iref) source. The smallest possible analog value that can be represented by the n-bit binary code is called resolution. The resolution of DAC with n-bit binary input is 1/2n of reference analog value. Every analog output will be a multiple of the resolution. In some converters the input reference analog signal will be multiplied or divided by a constant to get full scale value. Now the resolution will be 1/2n of full scale value. For example, consider an 8-bit DAC with reference analog voltage of 5 volts. Now the resolution of the DAC is (1/28) x 5 volts. The 8-bit digital input can take, 28 = 256 different values. The analog values for all possible digital input are as shown in table below.

Table 15 : The analog values for all possible digital input

Digital Input	Analog Output
0000 0000	$\frac{0}{2^8} \times 5$ Volts
0000 0001	$\frac{1}{2^8} \times 5$ Volts
0000 0010	$\frac{2}{2^8} \times 5$ Volts
0000 0011	$\frac{3}{2^8} \times 5$ Volts
⋮	⋮
1111 1111	$\frac{255}{2^8} \times 5$ Volts

The maximum input digital signal will have an analog value which is equal to reference analog value minus resolution. The digital-to-analog converters can be broadly classified into three categories, and they are :

- Current output
- Voltage output
- Multiplying type

The current output DAC provides an analog current as output signal. In voltage output DAC, the analog current signal is internally converted to voltage signal. In multiplying type DAC, the output is given by the product of the input signal and the reference source and the product is linear over a broad range. Basically, there is not much difference between these three types and any DAC can be viewed as multiplying DAC.

13.5.1 Typical DAC circuit

The basic components of a DAC are resistive network with appropriate values, switches, a reference source and a current to voltage converter as shown in figure below.

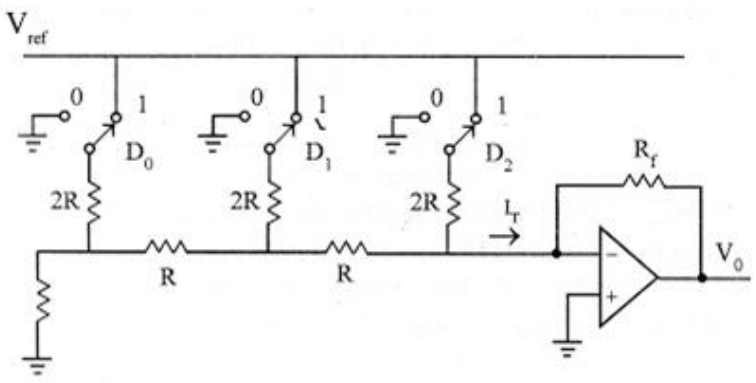


Figure 101: A typical R/2R ladder resistive network as DAC

The switches in the circuit of figure above can be transistors which connect the resistance either to ground or Vref. The resistors are connected in such a way that for any number of inputs, the total current is in binary proportion. The operational amplifier converts the current to a voltage signal V0, which can be calculated from the following equation. The circuit of figure shown above can be modified as 8-bitDAC, by increasing the number of R/2R ladder. For an 8-bit DAC the output voltage is given by:

$$V_0 = V_{ref}\frac{R_f}{R}\left(\frac{D_2}{2^1}+\frac{D_1}{2^2}+\frac{D_0}{2^3}\right)$$

The time required for converting the digital signal to analog signal is called conversion time. It depends on the response time of the switching transistors and the output amplifier. If the DAC is interfaced to microprocessor then the digital data (Signal) should remain at the input of DAC, until the conversion is complete. Hence to hold the data a latch is provided at the input of DAC.

$$V_0 = V_{ref}\frac{R_f}{R}\left(\frac{D_7}{2^1}+\frac{D_6}{2^2}+\frac{D_5}{2^3}+\frac{D_4}{2^4}+\frac{D_3}{2^5}+\frac{D_2}{2^6}+\frac{D_1}{2^7}+\frac{D_0}{2^8}\right)$$

The Digital-to-Analog converters compatible to microprocessors are available with or without internal latch and I to V converting amplifier. The AD558 of Analog Devices is an example of 8-bit DAC with an internal latch and I to V converting amplifiers. The output of AD558 is an analog voltage signal. The AD558 can be directly interfaced to 8085 microprocessor bus and it requires only two control signals: Chip Select (CS) and Chip Enable (CE). [No handshake signals are necessary for interfacing a DAC. The time between loading two digital data to DAC is controlled by software time delay].

The DAC0808 of National Semiconductor Corporation is an example of 8-bit DAC without

internal latch and I to V converting amplifier. The internal block diagram and the pin configuration of DAC0808 are shown in figure below.

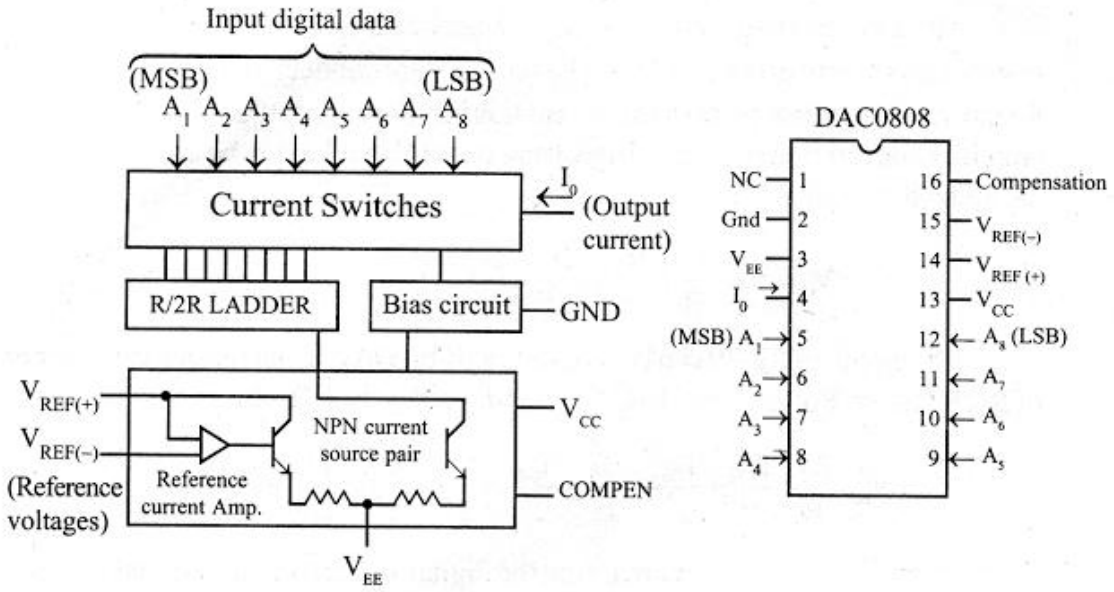


Figure 102: Block diagram and Pin configuration of DAC0808

The DAC0800 can be interfaced to 8085 system through an 8-bit latch as shown in figure below. The chip select (CS) signal from the decoder of the microprocessor system is delayed and inverted to clock the latch. If the DAC is memory mapped then the CS is from memory decoder. If the DAC is I/O mapped then CS is from I/O decoder.

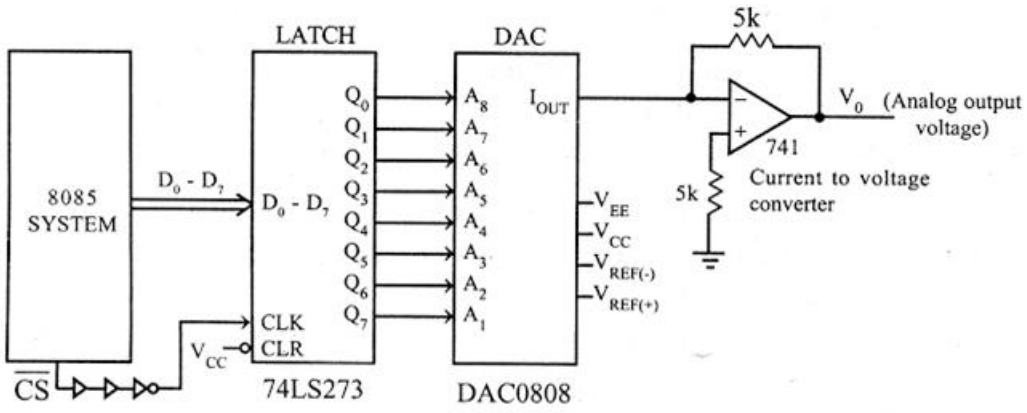


Figure 103: Interfacing DAC0808 to 8085 Microprocessor system

The processor sends an address, which is decoded by decoder in the microprocessor system to produce chip select signal. Then the processor sends a digital data to latch. The buffer and inverter will produce sufficient delay for CS signal so that, the latch is clocked only after the data

is arrived at the input lines of the latch. When the latch is clocked the digital data is send to DAC. The DAC will produce a corresponding current signal, which is converted to voltage signal by the op-amp 741. The typical settling time of DAC0800 is 150nsec. Therefore the processor need not wait for loading next data.

13.6 Summary

1. The popular stepper motor used for demonstration in laboratories has a step size of 1.8° (i.e., 200 steps per revolution).
2. The basic step size of the motor is called full-step.
3. By altering the switching sequence, the motor can be made to run with incremental motion of half the full-step value.
4. In many applications, an analog device has to be interfaced to digital system. But the digital devices cannot accept the analog signals directly and so the analog signals are converted into equivalent digital signal (data) using Analog-to-Digital Converter (ADC).
5. The A/D conversion is also called Quantization, in which the analog signal is represented binary data.
6. In many applications, the microprocessor has to produce analog signals for controlling certain analog devices. Basically the microprocessor system can produce only digital signals. In order to convert the digital signal to analog signal a Digital-to-Analog Converter. (DAC) has to be employed.

QUESTION BANK

1. What is a Microprocessor?
2. Why crystal is being preferred as a clock source?
3. Name High order / Low order Register in 8085 microprocessor?
4. Describe Tri-state logic?
5. What will happen if HLT instruction is executed in processor?
6. What type of architecture used in 8085 microprocessor?
7. What is the function of accumulator?
8. What are the different types of flags in 8085 microprocessor?
9. What are the types of general purpose registers in 8085?
10. What is the length of stack pointer in 8085 microprocessor?
11. What is the memory size of 8085 microprocessor?
12. How many bits is 8085 microprocessor?
13. What are the various interrupts in 8085 microprocessor? Which is the highest priority interrupt?
14. Which type of cycle is used for fetch and execute instruction?
15. How many address lines are there in 8085 microprocessor?
16. What do you mean by address bus?
17. Why is the data bus bi-directional?
18. Which Stack is used in 8085?
19. What is a flag?
20. Define memory word.
21. Describe briefly Program counter?
22. Name the 1st / 2nd / 3rd / 4th generation processor?
23. Name the processor lines of two major manufacturers?
24. Where’s MBR located on the disk?
25. What are the types of buses?
26. What is meant by Maskable interrupts?
27. What is Non-Maskable interrupts?

28. Which interrupts are generally used for critical events?
29. What is SIM and RIM instructions?
30. Explain the function of ALU and IO/M signals in the 8085 architecture?
31. Define T state.
32. Write down the control and status signals.
33. Give the bit position reserved for the flags.
34. Define Instruction cycle.
35. Give the functional categories of 8085 micro instructions?
36. What is an IN instruction?
37. What is an OUT instruction?
38. Give the difference between JZ and JNZ?
39. What is CMA?
40. What is CALL instruction?
41. How is the instruction set classified?
42. What is STA in data transfer instruction?
43. Why the number of out ports in the peripheral-mapped I/O is restricted to 256 ports?
44. If an input and output port can have the same 8-bit address how does the 8085 differentiate between the ports?
45. Why a latch is used for the output port and a tri-state buffer is used for the input port?
46. Define Memory mapped I/O?
47. What is an interrupt I/O?
48. What is Partial Decoding?
49. Define absolute decoding?
50. What is the purpose of an interrupt enable?
51. Give the commonly used priority modes.
52. Give the additional features of 8259A controller?
53. What is the purpose of 8255 PPI?
54. List the operating modes of 8255A PPI?
55. Specify the bit of a control word for the 8255, which differentiates between the I/O mode and the BSR mode?
56. Write the input /output feature in Mode 9 for the 8255A PPI?

57. Write down the output control signals used in 8255A PPI?
58. What is the use of mode 2 in 8255A PPI?
59. What is the purpose for scan section in keyboard interface?
60. List the major components of 8251A programmable communication interface?
61. Give the various modes of 8254 timer.
62. Define A/D and D/A converters?
63. To interface an A/D converter with the microprocessor, what does the microprocessor do?

Reference

(n.d.). Retrieved Sep. 2016, 19, from http://www.sharpmz.org/mz-700/8253ovview.htm

Agarwal, P. (2013, Nov. 18). *Microprocessor*. Retrieved Sep. 22, 2016, from NPTEL: http://nptel.ac.in/courses/108107029/4 available under creative commons license.

Agarwal, P. (2013, Nov. 18). *Microprocessor*. Retrieved Sep. 18, 2016, from NPTEL: http://nptel.ac.in/courses/108107029/44 available under creative commons license

Basheer, N. (2013, Feb). *8085 Microprocessor Based Stepper Motor Control System* . Retrieved Oct. 02, 2016, from Figure adopted from https://mediatoget.blogspot.in/2013/02/8085-microprocessor-based-stepper-motor.html available under creative commons license.

Basheer, N. (2012, Sep. 22). *INTEL 8255 Programmable Peripheral Interface*. Retrieved Sep. 18, 2016, from https://mediatoget.blogspot.in/2012/11/intel-8255-ppi.html available under creative commons 3.0 unported license

Basheer, N. (2011, Dec.). *Interrupts in 8085 microprocessor*. Retrieved Sep. 24, 2016, from https://mediatoget.blogspot.in/2011/12/interrupts-of-intel-8085.html available under a Creative Commons Attribution 3.0 Unported License.

Godse, A. P. (2010). *Microprocessor and Microcontrollers.* Technical Publications.

Kani, A. N. *8085 Microprocessors and its Applications.* Mc Graw Hill.

Microprocessor 8085 – Timing and control; Interrupts; Stack Memory; Interfacing device. (n.d.). Retrieved Sep. 20, 2016, from vle.du.ac.in/mod/resource/view.php?inpopup=true&id=13342 available under creative commons license

Microprocs. (2009, Feb. 10). Retrieved Sep. 23, 2016, from http://microprocs.wikispaces.com/Introduction available under a Creative Commons Attribution Share-Alike 3.0 License.

MikroElektronika. (n.d.). *Introduction to the World of Microcontrollers*. Retrieved Sep. 23, 2016, from http://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/front-

matter/introduction-to-the-world-of-microcontrollers/ available under a Creative Commons Attribution 4.0 license.

Peripheral Interfacing. (n.d.). Retrieved Sep. 18, 2016, from http://8085projects.info/Peripheral-Interfacing.html

Seshadri, B. (2016, Jan. 02). *What is Memory Mapping in Microprocessor based systems?* Retrieved Sep. 22, 2016, from Adapted from https://www.quora.com/What-is-Memory-Mapping-in-Microprocessor-based-systems

The Processor Cycle of 8085 . (2009, Jan. 07). Retrieved Sep. 20, 2016, from http://microprocessor-8085.blogspot.in/2009/01/processor-cycle-of-8085.html