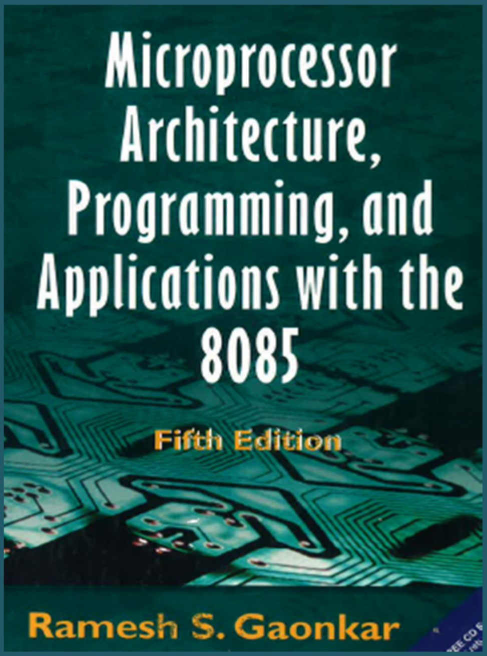

Microprocessor Architecture, Programming, and Applications with the 8085
Fifth Edition
Ramesh S. Gaonkar

Unit – 1

Chapter-1

Microprocessors, Microcomputers, and Assembly Language

Topics Covered

1. **Microprocessors**
2. **Microprocessor Instruction Set and Computer Languages**
3. **From Large Computers to Single Chip Microcontrollers**

1. Microprocessors

A microprocessor is a

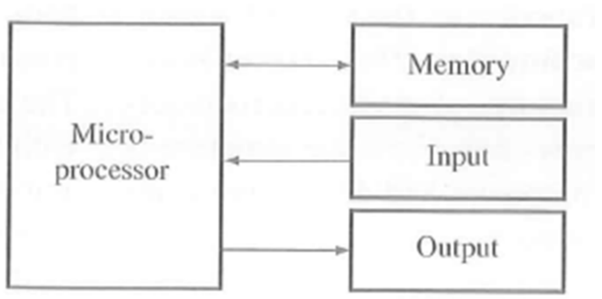
- multipurpose
- programmable
- clock-driven
- register-based

electronic device

It reads binary instructions from a storage device called memory.

It accepts binary data as input and processes data according to those instructions, and provides results as output.

A Programmable Machine



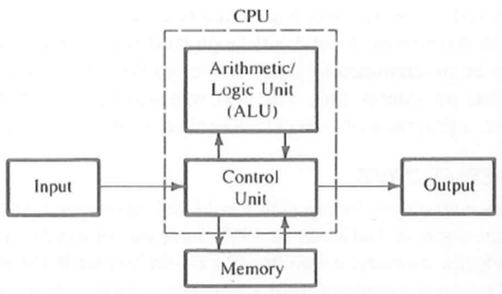
A programmable machine has two types of components:

1. Hardware components
2. Software components

Microprocessor as a CPU

We can view the microprocessor as a primary component of a computer.

Traditionally, the Computer is represented in block diagram as shown below.



The block diagram shows that the computer has four components: memory, input, output, and the CPU, which consists of the ALU and the control unit.

The CPU contains various registers to store data, the ALU is to perform arithmetic and logical operations, instruction decoders, counters, and control lines.

The CPU reads instructions from the memory and performs the tasks specified.

The CPU also communicates with I/O either to accept or to send data.

The CPU is the primary and central player in communicating with devices such as memory, and I/O.

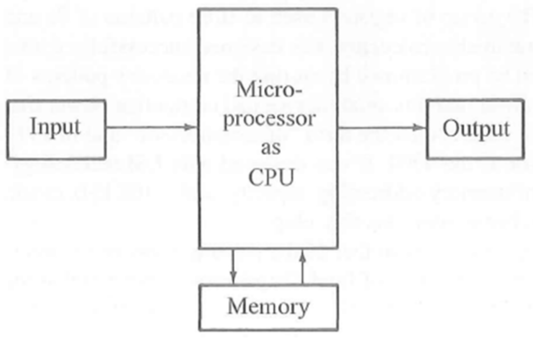
However, the timing of the communication process is controlled by the group of circuits called the ***control unit***.

Microprocessor as a CPU (Contd …)

In the late 1960s, the CPU was designed with discrete components on various boards.

With the advent of integrated circuit technology, it became possible to build the CPU on a single chip.

This chip came to be known as a ***microprocessor*** and can be shown by the below block diagram.



Microprocessor as a CPU (Contd ...)

A computer with a microprocessor as its CPU is known as a ***microcomputer***.

The terms microprocessor and microprocessor unit (MPU) are often used synonymously.

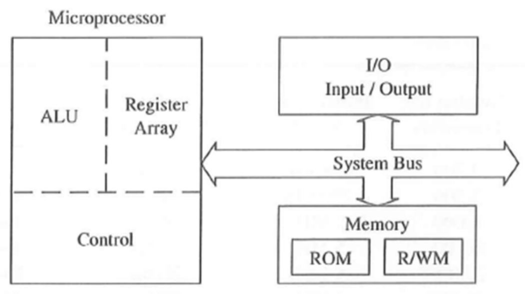
MPU implies a complete processing unit with the necessary control signals.

The microprocessor is a ***clock-driven semiconductor device*** consisting of electronic logic circuits manufactured by using either a large-scale integration (LSI) or very-large-scale integration (VLSI) technique.

The microprocessor is capable of performing various computing functions and making decisions to change the sequence of program execution.

The microprocessor is in many ways similar to the CPU, but includes all the logic circuitry, including the control unit, on one chip.

The microprocessor can be divided into three segments: ALU, register array, and control unit.



Arithmetic/Logic Unit (ALU)	Various computing functions are performed on data in the ALU. The ALU performs arithmetic operations, and such logic operations.
Register Array	This consists of various registers identified by letters such as B, C, D, E. H, and L. These registers are primarily used to store data temporarily during the program execution These are accessible to the user through instructions.
Control Unit	The control unit provides the necessary timing and control signals to all the operations in the microcomputer. It controls the flow of data between the micro processor and memory and peripherals.

Microprogramming

Example:

A full adder circuit can be designed with registers, logic gates, and a clock.

- The clock initiates the adding operation.
- The bit pattern of an instruction initiates a sequence of clock signals.
- The appropriate logic circuits in the ALU are activated to perform the task

This is called ***microprogramming***, which is done in the design stage of the microprocessor.

Memory

Memory stores binary information as instructions and data, and provides that information to the microprocessor whenever necessary.

To execute programs, the microprocessor reads instructions and data from memory and performs the computing operations in its ALU section.

Results are either transferred to the output section for display or stored in memory for later use.

The memory has two sections:

- Read Only Memory (ROM)
- Random Access Memory (RAM)

I/O (Input/Output)

The I/O communicates with the outside world.

I/O includes two types of devices: input and output.

The I/O devices are also known as ***peripherals***.

System Bus

The system bus is a communication path between the microprocessor and peripherals.

It is a group of wires to carry bits.

All peripherals and memory share the same bus; however, the microprocessor communicates with only one peripheral at a time.

The timing is provided by the control unit of the microprocessor.

How does the microprocessor work?

Assume that a program and data are already entered in the RAM.

When the microprocessor is given a command to execute the program, it reads and executes one instruction at a time and finally sends the result for display.

2. Microprocessor Instruction Set and Computer Languages

Microprocessors recognize and operate in binary numbers.

However, each microprocessor has its own binary words, meanings, and language.

The words are formed by combining a number of bits for a given machine.

The ***word*** is defined as the number of bits the microprocessor recognizes and processes at a time.

The word length ranges from four bits for small, microprocessor-based systems to 64 bits for high-speed large computers.

Another term commonly used to express word length is ***byte***, a group of 8 bits.

Machine Language

Each machine has its own set of instructions based on the design of its CPU or of its microprocessor.

To communicate with the computer, one must give instructions in binary language **(machine language).**

Because it is difficult for most people to write programs in sets of 0s and 1s, computer manufacturers have devised English-like words to represent the binary instructions of a machine.

Programs written using these English-like words are called ***assembly language*** programs.

An assembly language is specific to a given machine, so programs written in assembly language are not transferable from one machine to another.

To circumvent this limitation, high-level languages like as BASIC, FORTRAN, Pascal etc. have been devised.

A program written in the high-level language can be machine-in dependent.

The number of bits in a word for a given machine is fixed.

The words are formed through various combinations of these bits.

For example, a machine with a word length of eight bits can have 2^8 (256) combinations of eight bits, thus a language of 256 words.

These words are used in different combinations to formulate instructions.

The set of instructions designed for the machine is called the ***machine language***, or a ***binary language***,

8085 Machine Language

The 8085 is a microprocessor has 8-bit word length.

Its **instruction set** (or language) is designed by using various combinations of these eight bits.

The 8085 is an improved version of the earlier processor 8080A.

For example,

0011 1100	It is an instruction that increments the number in the register called the accumulator by one.
1000 0000	It is an instruction that adds the number in the register called B to the number in the accumulator, and keeps the sum in the accumulator.

However, It is tedious and error-inducing to recognize and write instructions in binary language, these instructions are written in hexadecimal code for convenience and entered in a single-board microcomputer by using Hex keys.

For example,

The binary instruction 0011 1100 is equivalent to 3C in hexadecimal (3 for 0011 and C for 1100).

This instruction can be entered in a single-board microcomputer system with a Hex key board by pressing two keys: 3 and C.

Then these keys are translated into their equivalent binary pattern for execution.

8085 Assembly Language

Even though the instructions can be written in hexadecimal code, it is still difficult to understand a program written in hexadecimal numbers.

Therefore, for each instruction, a symbolic code has been devised.

These symbolic codes are called ***mnemonics***.

The mnemonic for a particular instruction consists of letters suggesting the operation to be performed by that instruction.

For example,

Binary	Hexadecimal	Mnemonic	Description
0011 1100	3C	INR A	Increment the contents of Accumulator by 1.
1000 0000	80	ADD B	Adds the contents of the register B to the contents of the accumulator and store the result in the accumulator.

Machine language and assembly language are microprocessor-specific and hence they both considered ***low-level languages***.

The machine language is in binary, and the assembly language is in English-like words.

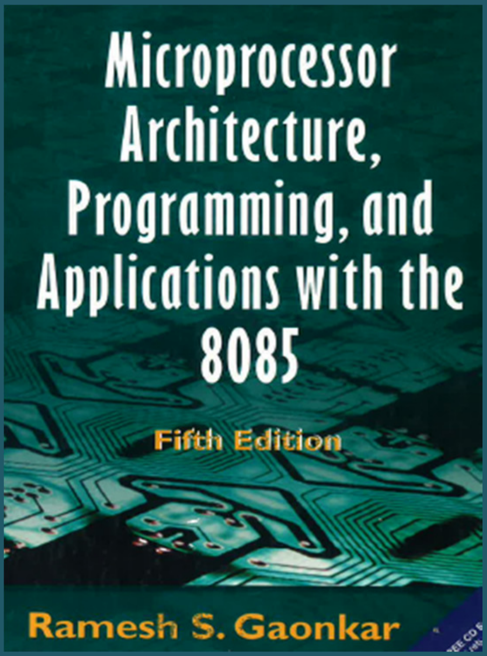
However, the microprocessor understands only the binary.

Using the ***assembler***, the mnemonics are translated into the binary.

3. From Large Computers to Single Chip Microcontrollers

Assignment

1. Large Computers
2. Medium-size computers
3. Microcomputers
4. Personal computers
5. Workstations
6. Single-board microcomputers
7. Single-chip microcomputers (Microcontrollers)


Microprocessor Architecture, Programming, and Applications with the 8085
Fifth Edition
Ramesh S. Gaonkar

Unit – 1

Chapter-2

8085 Assembly Language Programming

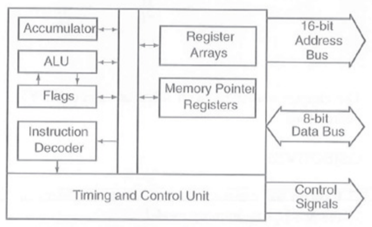
Topics Covered

1. The 8085 Programming Model
2. Instruction Classification
3. Instruction, Data Format, and Storage
4. Writing, Assembling and Executing a Simple Program
5. Overview of 8085 Instruction Set
6. Writing and Hand Assembling a Program

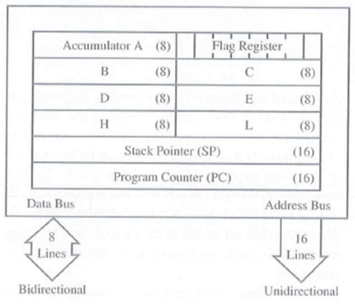
1. The 8085 Programming Languge

The microprocessor can be represented in terms of its hardware and a programming model.

A simplified hardware model of a microprocessor is shown below:



Programming Model



Flag Register

REGISTERS

The 8085 has six general-purpose registers to store 8-bit data.

These registers are identified as B, C, D, E, H, and L.

They can be combined as register pairs BC, DE, and HL-to perform some 16-bit operations.

The programmer can use these registers to store data into the registers using data copy instructions.

ACCUMULATOR

The accumulator (identified as A) is an 8-bit register that is part of the ALU.

This register is used to store 8-bit data and to perform arithmetic and logical operations.

The result of an operation is stored in the accumulator.

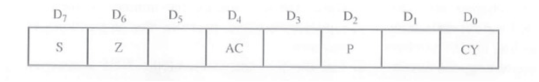
FLAGS

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers.

These flip-flops are:

1. Zero (Z)
2. Carry (CY)
3. Sign (S)
4. Parity (P)
5. Auxiliary Carry (AC)

The flip-flop bit positions in the flag register are shown below. data conditions.



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

The microprocessor uses these flags to test data conditions.

PROGRAM COUNTER (PC) AND STACK POINTER (SP)*

The PC and SP are two 16-bit registers used to hold memory addresses.

The microprocessor uses the PC register to sequence the execution of the instructions.

The function of the program counter is to point to the memory address from which the next byte is to be fetched.

When a byte, representing the machine code is being fetched, the program counter is incremented by one to point to the next memory location.

The SP is also a 16-bit register used as a memory pointer.

2. Instruction Classification

The 8085 instructions can be classified into five functional categories:

1. Data transfer operations
2. Arithmetic operations
3. Logical operations
4. Branching operations
5. Machine-control operations

Data transfer operations	Move, Copy
Arithmetic operations	Addition, Subtraction, Increment/Decrement
Logical operations	AND, OR, Rotate, Compare, Complement
Branching operations	Jump, Call, Return, Restart
Machine control operations	Halt, Interrupt, Do nothing

3. Instruction, Data Format, and Storage

An **instruction** is a command to the microprocessor to perform a given task on specified data.

Each instruction has two parts:

1) The opcode part that specifies the task to perform
2) The operand part that specifies the data to be operated on

The operand part may include 8-bit (or 16-bit) data, an internal register, a memory location, or an 8-bit (or 16-bit) address.

Instruction Word Size

The 8085 instruction set is classified into the following three groups according to word size or byte size.

In the 8085, "byte" and "word" are synonymous because it is an 8-bit microprocessor.

However, instructions are commonly referred to in terms of bytes rather than words.

1. One-byte instructions
2. Two-byte instructions
3. Three-byte instructions

1. One-byte instructions

A one-byte instruction includes the opcode and the operand in the same byte.

For example:

Task	Opcode	Operand	Binary Code	Hex Code
Copy the contents of the accumulator in register C	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator	ADD	B	1000 0000	80H
Complement each bit in the accumulator	CMA		0019 1111	2FH

2. Two-byte instructions

In a 2-byte instruction, the first byte specifies the operation code and the second byte specifies the operand.

For example:

Task	Opcode	Operand	Binary Code	Hex Code
Load an 8-bit data in AC	MVI	A, 32H	0011 1110 0011 0010	3E 32
Load an 80bit data in register B	MVI	B, F2H	0000 0110 1111 0010	06 F2

3. Three-byte instructions

In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address.

Note: The second byte is the low-order address and the third byte is the high-order address.

For example:

Task	Opcode	Operand	Binary Code	Hex Code
Load contents of memory 2050H into AC.	LDA	2050H	0011 1010 1010 0000 0010 0000	3A 50 20
Branch to memory location 2085H	JMP	2085H	1100 0011 1000 0101 0010 0000	C3 85 20

Addressing Modes

These various ways of specifying data are called the **addressing modes.**

In the microprocessor the mnemonic letters used to specify a command are chosen by the manufacturer.

When an instruction is stored in memory, it is stored in binary code, the only code the microprocessor is capable of reading and understanding.

The important point to remember is that the microprocessor neither reads nor understands mnemonics or hexadecimal numbers.

Opcode Formats

In general, in microprocessor user's manual, opcodes are specified in binary format and 8-bits are divided in various groups.

However, this information is not necessary to understand assembly language programming.

In the design of the 8085 microprocessor chip, all operations, registers, and status flags are identified with a specific code.

For example, all internal registers are identified as follows:

Code	Registers	Code	Register Pairs
000	B	00	BC
001	C	01	DE
010	D	10	HL
011	E	11	AF OR SP
100	H		
101	L		
111	A		
110	Reserved for Memory-Related operation		

Data Formats

The 8085 is an 8-bit microprocessor, and it processes only binary numbers.

However, the real world operates in decimal numbers and languages of alphabets and characters.

Therefore, we need to represent binary numbers in a different way.

In 8-bit processor systems, commonly used codes and data for mats are ASCII, BCD, signed integers, and unsigned integers.

Code	Description
ASCII Code	This is a 7-bit alphanumeric code that represents decimal numbers, English alphabets, and nonprintable characters such as carriage return.
Extended ASCII Code	Extended ASCII is an 8-bit code. The additional numbers (beyond 7-bit ASCII code) represent graphical characters.
BCD (Binary Coded Decimal) Code	It is used for decimal numbers. The decimal numbering system has ten digits, 0 to 9. Therefore, we need only four bits to represent ten digits from 0000 to 1001. An 8-bit register in the 8085 can accommodate two BCD numbers.
Signed Integer	A signed integer is either a positive number or a negative number. In an 8-bit processor, the most significant digit, D_7, is used for the sign; 0 represents the positive sign and 1 represents the negative sign. The remaining seven bits, D_6-D_0, rep resent the magnitude of an integer. The largest positive integer that can be processed by the 8085 at one time is 0111 1111 (7FH). The remaining Hex numbers, 80H to FFH, are considered negative numbers. **Note:** All negative numbers in this microprocessor are represented in 2's complement format.
Unsigned Integer	An integer without a sign can be represented by all the 8 bits in a microprocessor register. The largest number that can be processed at one time is FFH.

An integer without a sign can be represented by all the 8 bits in a microprocessor register.

Therefore, the largest number that can be processed at one time is FFH.

We need to understand that the 8085 microprocessor is not limited to handling only 8-bit numbers.

Numbers larger than 8 bits (such as 16-bit or 24-bit numbers) are processed by dividing them in groups of 8 bits.

4. Writing, Assembling and Executing a Simple Program

A program is a sequence of instructions written to tell a computer to perform a specific task.

The instructions are selected from the instruction set of the microprocessor.

To write a program,

- ➢ divide a given problem in small steps in terms of the operations the 8085 can perform
- ➢ then translate these steps into instructions

Example: Writing a program of adding two numbers in the 8085 language.

1. PROBLEM STATEMENT

Write instructions to load the two hexadecimal numbers 32H and 48H in registers A and B, respectively. Add the numbers, and display the sum at the LED output port PORT1.

2. PROBLEM ANALYSIS

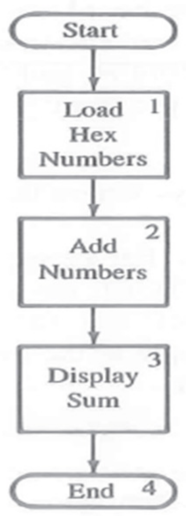
Divide the problem into small steps to examine the process of writing programs.
The wording of the problem provides sufficient clues for the necessary steps.
They are as follows:

1. Load the numbers in the registers.
2. Add the numbers.
3. Display the sum at the output port PORT1.

Example: (Contd ...)

3. FLOWCHART

The steps listed in the problem analysis and the sequence can be represented in a flowchart.



Example: (Contd …)

4. ASSEMBLY LANGUAGE PROGRAM

To write an assembly language program, we need to translate the blocks shown in the flowchart into 8085 operations and then into mnemonics.

By examining the blocks, we can classify them into three types of operations:

Blocks l and 3 are copy operations;

Block 2 is an arithmetic operation;

Block 4 is a machine-control operation.

To translate these steps into assembly and machine languages, you should review the instruction set.

The translation of each block into mnemonics with comments is shown as follows:

Block 1:	MVI A, 32H	Load register A with 32H
	MVI B, 48H	Load register B with 48H
Block 2:	ADD B	Add two bytes and save the sum in A
Block 3:	OUT 01H	Display the accumulator content at port 01H
Block 4:	HALT	End

Example: (Contd …)

5. FROM ASSEMBLY LANGUAGE TO HEX CODE

To convert the mnemonics into Hex code, we need to look up the code in the 8085 instruction set; this is called either manual or hand assembly.

Mnemonics		**Hex Code**	
MVI	A,32H	3E	2-byte instruction
		32	
MVI	B,48H	06	2-byte instruction
		48	
ADD	B	80	1-byte instruction
OUT	0lH	D3	2-byte instruction
		01	
HLT		76	1-byte instruction

Example: (Contd ...)

6. STORING IN MEMORY AND CONVERTING FROM HEX CODE TO BINARY CODE

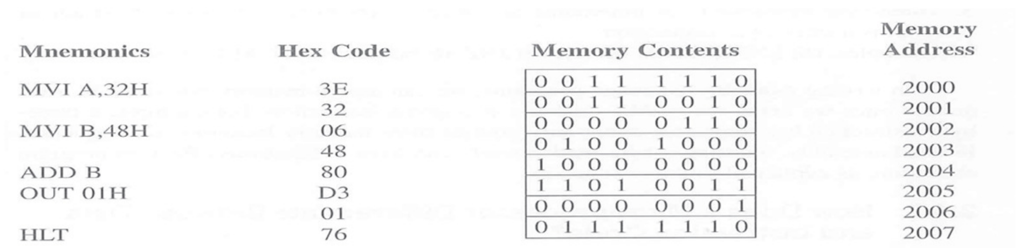
To store the program in R/W memory of a single-board microcomputer and display the output, we need to know the memory addresses and the output port address.
Let us assume that R/W memory ranges from 2000H to 20FFH, and the system has an LED output port with the address 0lH.

Now, to enter the program:

1. Reset the system by pushing the RESET key.
2. Enter the first memory address using Hex keys where the program should be stored. Let us assume it is 2000H.
3. Enter each machine code by pushing Hex keys. For example, to enter the first machine code, push the 3, E, and STORE keys. (The STORE key may be labeled differently in different systems.) When you push the STORE key, the program will store the machine code in memory location 2000H and upgrade the memory address to 2001H.
4. Repeat Step 3 until the last machine code, 76H.
5. Reset the system.

Example: (Contd ...)

In this example, the program will be stored in memory as follows:



Mnemonics	Hex Code	Memory Contents	Memory Address
MVI A,32H	3E	0 0 1 1 1 1 1 0	2000
	32	0 0 1 1 0 0 1 0	2001
MVI B,48H	06	0 0 0 0 0 1 1 0	2002
	48	0 1 0 0 1 0 0 0	2003
ADD B	80	1 0 0 0 0 0 0 0	2004
OUT 01H	D3	1 1 0 1 0 0 1 1	2005
	01	0 0 0 0 0 0 0 1	2006
HLT	76	0 1 1 1 1 1 1 0	2007

Example: (Contd …)

7. EXECUTING THE PROGRAM

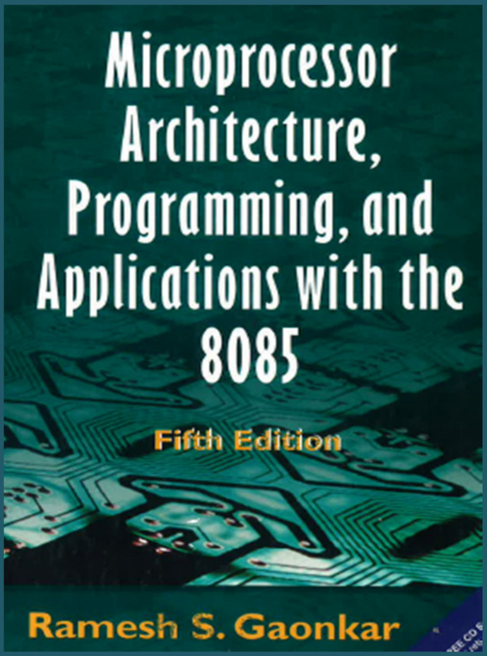
- To execute the program, we need to enter the memory address 2000H to tell the microprocessor where the program begins.
- Now, we push the Execute key to begin the execution.
- As soon as the Execute key is pushed, the microprocessor loads 2000H in the program counter, and the program control is transferred from the Monitor program (i.e., OS) to our program.

5. Overview of 8085 Instruction Set

Assignment

6. Writing and Hand Assembling a Program

Assignment


Microprocessor Architecture, Programming, and Applications with the 8085
Fifth Edition
Ramesh S. Gaonkar

Unit – 1

Chapter-3

Microprocessor Architecture And Microcomputer Systems

Topics Covered

1. **Microprocessor Architecture and its Operations**
2. **Memory**
3. **I/O Devices**
4. **Example of a Microcomputer System**

A microcomputer system consists of

- Microprocessor
- Memory
- I/O

The micro processor manipulates data, controls the timing of various operations, and communicates with memory and I/O.

The internal logic design of the microprocessor determines how and when various operations are performed by the microprocessor.

The system bus provides paths for the flow of data and instructions.

1. Microprocessor Architecture and its Operations

The microprocessor is a programmable digital device, designed with registers, flip-flops, and timing elements.

The microprocessor has a set of instructions, designed internally, to manipulate data and communicate with peripherals.

The process of data manipulation and communication is determined by the logic design of the microprocessor, called the architecture.

The microprocessor can be programmed to perform functions on given data by selecting necessary instructions from its set.

These instructions are given to the microprocessor by writing them into its memory.

The microprocessor executes one instruction at a time.

The result can be stored in memory or sent to such output devices as **LED**s or a **CRT terminal**.

In addition, the microprocessor can respond to external signals.

It can be interrupted, reset, or asked to wait to synchronize with slower peripherals.

All the various functions performed by the microprocessor can be classified in three general categories:

- Microprocessor-initiated operations
- Internal operations
- Peripheral (or externally initiated) operations

To perform these functions, the microprocessor requires a group of logic circuits and a set of signals called control signals.

However, early processors did not have the necessary circuitry on one chip; the complete units were made up of more than one chip.

Therefore, the term ***microprocessing unit*** (MPU) is defined here as a group of devices that can perform the functions with the necessary set of control signals.

This term is similar to the term ***central processing unit*** (CPU).

However, later microprocessors include most of the necessary circuitry to perform these operations on a single chip.

1. Microprocessor-Initiated Operations and 8085 Bus Organization

The MPU performs primarily four operations:

- Memory Read: Reads data (or instructions) from memory.
- Memory Write: Writes data (or instructions) into memory.
- I/0 Read: Accepts data from input devices.
- I/0 Write: Sends data to output devices.

All these operations are part of the communication process between the MPU and peripheral devices.

To communicate with a peripheral (or a memory location), the MPU needs to perform the following steps:

Step 1: Identify the peripheral or the memory location (with its address).

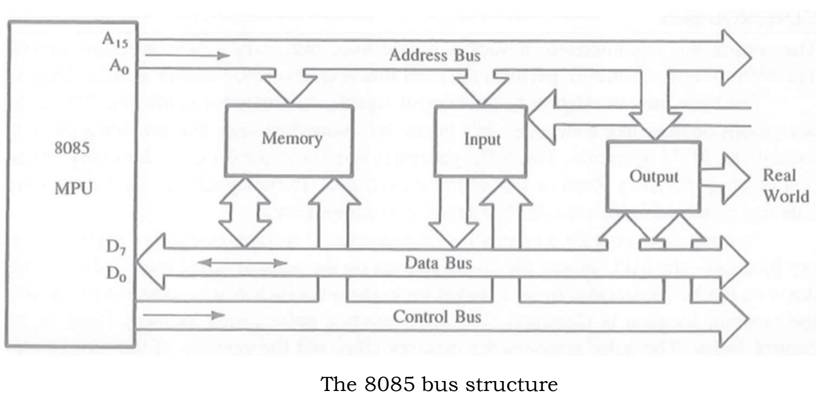
Step 2: Transfer binary information (data and instructions).

Step 3: Provide tinting or synchronization signals.

The 8085 MPU performs these functions using three sets of communication lines called buses: the address bus, the data bus, and the control bus.

The 8085 MPU performs the functions using three sets of buses: the address bus, the data bus, and the control bus.

These buses called as the system bus.



The 8085 bus structure

ADDRESS BUS

- The address bus is a group of 16 lines identified as A_0 to A_{15}.
- The address bus is **unidirectional:** bits flow in one direction: from the MPU to peripheral devices.
- The MPU uses the address bus to perform the first function: identifying a peripheral or a memory location.
- The 8085 MPU with its 16 address lines is capable of addressing 2^{16} = 65,536 memory locations.

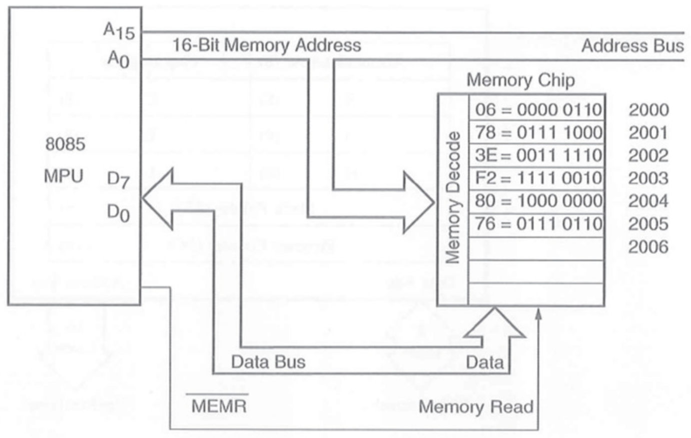
DATA BUS

- The data bus is a group of eight lines used for data flow.
- These lines are **bidirectional:** data flow in both directions between the MPU and memory or periph eral devices.
- The eight data lines enable the MPU to manipulate 8-bit data ranging from 00 to FF (2^8 = 256 numbers).
- The 8085 is known as an 8-bit microprocessor.
- Microprocessors such as the Intel 8086, Zilog Z8000, and Motorola 68000 have 16 data lines and are known as 16-bit microprocessors.
- The Intel 80386/486 have 32 data lines; thus they are classified as 32-bit microprocessors.

CONTROL BUS

- The control bus is comprised of various single lines that carry synchronization signals.
- The MPU uses such lines to provide timing signals.
- The MPU generates specific control signals for every operation it performs (such as Memory Read or I/0 Write).
- These signals are used to identify a device type with which the MPU intends to communicate.

BUS Communication for Memory Read/Write



2. Internal Data Operations and 8085 Registers

The internal architecture of the 8085 microprocessor determines how and what operations can be performed with the data.

These operations are:

1. Store 8-bit data.
2. Perform arithmetic and logical operations.
3. Test for conditions.
4. Sequence the execution of instructions.
5. Store data temporarily during execution in the defined R/W memory locations called the stack.

To perform the operations, the microprocessor requires registers, an ALU and control logic, and internal buses.

3. Peripherals

External devices (or signals) can initiate the following operations, for which individual pins on the microprocessor chip are assigned: Reset, Interrupt, Ready, Hold.

Reset	When the reset pin is activated by an external key, all internal operations are suspended and the program counter is cleared to 0000H. Now the program execution can again begin at the zero memory address.
Interrupt	The microprocessor can be interrupted from the normal execution of instructions and asked to execute some other instructions called a service routine. The microprocessor resumes its operation after completing the service routine.
Ready	The 8085 has a pin called READY. If the signal at this READY pin is low, the microprocessor enters into a ***Wait*** state. This signal is used primarily to synchronize slower peripherals with the microprocessor.
Hold	When the HOLD pin is activated by an external signal, the microprocessor relinquishes control of buses and allows the external peripheral to use them. For example, the HOLD signal is used in Direct Memory Access (DMA) data transfer.

2. Memory

Memory is an essential component of a microcomputer system.

It stores binary instructions and data for the microprocessor.

There are various types of memory, which can be classified in two groups: ***prime (or main) memory*** and ***storage (or secondary) memory***.

The ***RAM*** (or R/W memory) is made of registers, and each register has a group of flip-flops that store bits of information.

These flip-flops are called memory cells.

The number of bits stored in a register is called a **memory** word.

Memory devices (chips) are available in various word sizes.

The user can use RAM to hold programs and store data.

On the other hand, the ***ROM*** stores information permanently in the form of diodes.

The group of diodes can be viewed as a register.

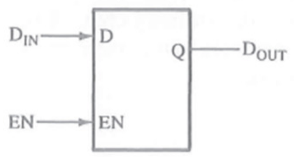
To communicate with memory, the MPU should be able to

1. select the chip,
2. identify the register, and
3. read from or write into the register.

1. Flip-Flop or Latch as a Storage Element

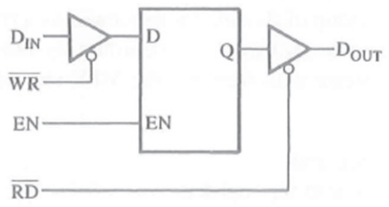
Memory is a circuit that can store bits, 1s and 0s.
A flip-flop or a latch is a basic element of memory.

To write or store a bit in the flip-flop, we need an input data bit (D_{IN}) and an enable signal (EN), as shown below.



In this flip-flop, the stored bit is always available on the output line D_{OUT}.

To avoid unintentional change in the input and control the availability of the output, we can use two tri-state buffers on the flip-flop, as shown below.



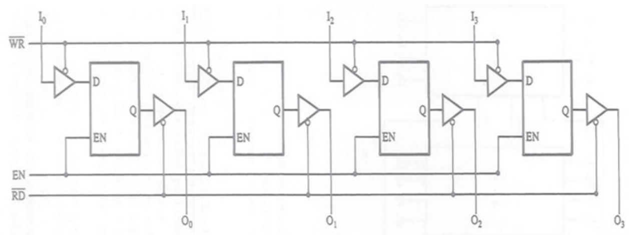
Now we can write into the flip-flop by enabling the input buffer and read from it by enabling the output buffer.

WR is the *Write signal* and RD is the *Read signal.*

By default these are in low state, i.e., at 0 state.

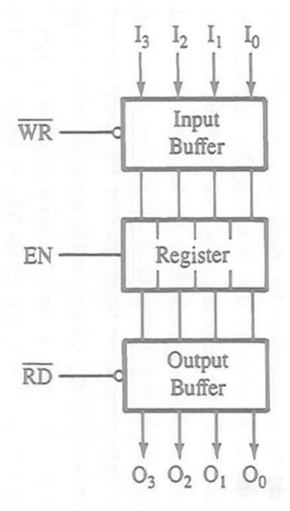
This flip-flop, which can store one binary bit, is called a memory cell.

The following figure shows four such cells or latches grouped together; this is a register, which has four input lines and four output lines and can store four bits; thus the size of the memory word is four bits, in this case.



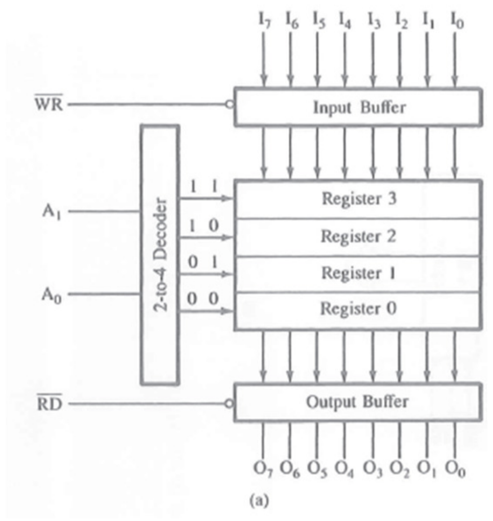
The size of this register is specified either as 4-bit or 1 x 4-bit, which indicates one register with four cells or four I/O lines.

The figure below shows block diagram of the 4-bit register.

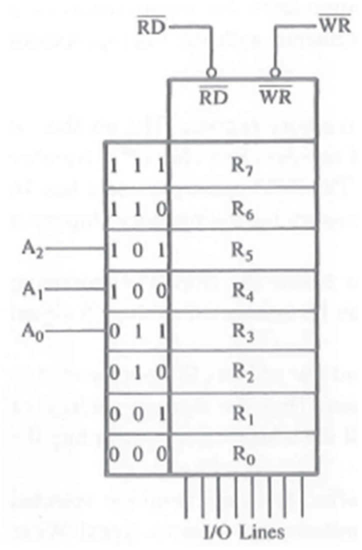


In the following figure, four 8-bit registers are arranged in a sequence.

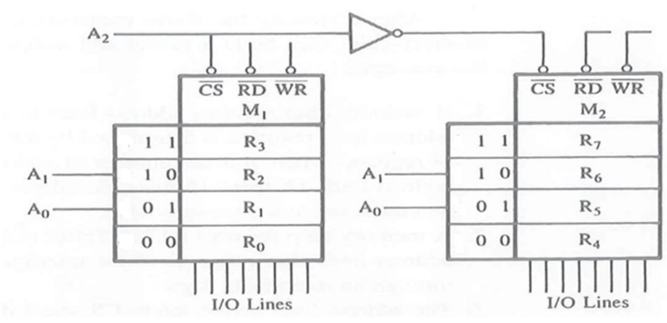
To write into or read from any one of the registers, a specific register should be enabled.



Case-1: Eight 8-bit registers on a single chip.



Case-2: Eight 8-bit registers on two chips.



Important points of Memory

1. A memory chip requires address lines to identify a memory register. The number of address lines required is determined by the number of registers in a chip (2^n = Number of registers where n is the number of address lines). The 8085 microprocessor has 16 address lines. Of these 16 lines, the address lines necessary for the memory chip must be connected to the memory chip.

2. A memory chip requires a Chip Select (CS) signal to enable the chip. The remaining address lines of the microprocessor can be connected to the CS signal through an interfacing logic.

3. The address lines connected to CS select the chip, and the address lines connected to the address lines of the memory chip select the register. Thus the memory address of a register is determine by the logic levels (0/1) of all the address lines.

4. The control signal Read (RD) enables the output buffer, and data from the selected register are made available on the output lines. Similarly, the control signal Write (WR) enables the input buffer, and data on the input lines are written into memory cells. The microprocessor can use its Memory Read and Memory Write control signals to enable the buffers and the data bus to transport the contents of the selected register between the microprocessor and memory.

2. Memory Map and Address

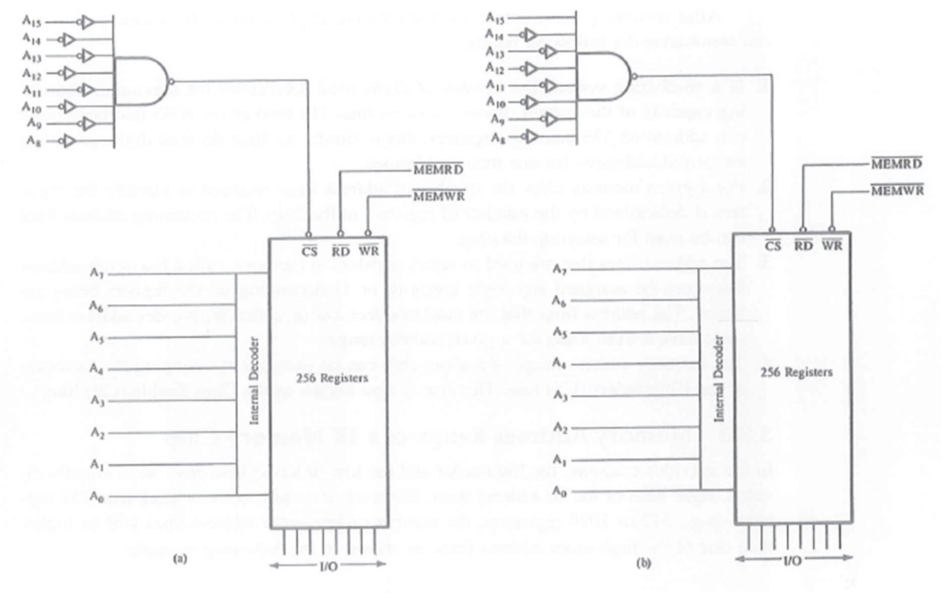
Typically, in an 8-bit microprocessor system, 16 address lines are available for memory.

This means it is a numbering system of 16 binary bits and is capable of identifying 2^{16} (65,536) memory registers, each register with a 16-bit address.

The entire memory addresses can range from 0000 to FFFF in Hex.

A memory map is a pictorial representation in which memory devices are located in the entire range of addresses.

Memory addresses provide the locations of various memory devices in the system, and the interfacing logic defines the range of memory addresses for each memory device.



Important points

1. In a numbering system, the number of digits used determines the maximum addressing capacity of the system. Sixteen address lines (16 bits) of the 8085 microprocessor can address 65,536 memory registers.

2. For a given memory chip, the number of address lines required to identify the registers is determined by the number of registers in the chip. The remaining address lines can be used for selecting the chip.

3. The address lines that are used to select registers in memory, called low-order address lines, can be assigned any logic levels (0 or 1) depending on the register being selected. The address lines that are used to select a chip, called high-order address lines, must have a fixed logic for a given address range.

4. The memory address range of a given chip can be changed by changing the hardware of the Chip Select (CS) line. This line is also known as the Chip Enable (CE) line.

3. Memory and Instruction Fetch

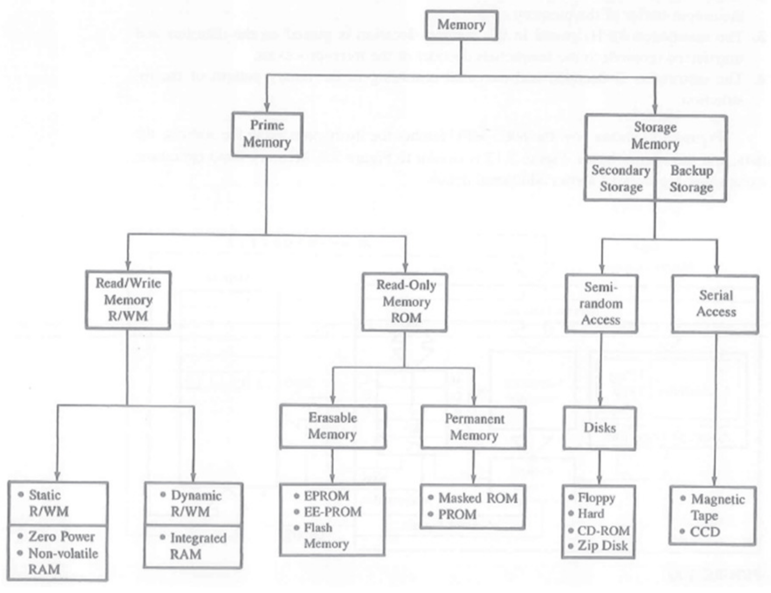
The primary function of memory is to store instructions and data and to provide that in formation to the MPU whenever the MPU requests it.

The MPU requests the information by sending the address of a specific memory register on the address bus and enables the data flow by sending the control signal, as illustrated in the next example.

To fetch the instruction located in memory location 2005H, the following steps are per formed:

1. The program counter places the 16-bit address 2005H of the memory location on the address bus.
2. The control unit sends the Memory Read control signal (MEMR, active low) to enable the output buffer of the memory chip.
3. The instruction (4FH) stored in the memory location is placed on the data bus and transferred to the instruction decoder of the microprocessor.
4. The instruction is decoded and executed according to the binary pattern of the instruction.

4. Memory Classification



3. I/O Devices

Input/output devices are the means through which the MPU communicates with "the out side world."

The MPU accepts binary data as input from devices such as keyboards and Analogue/Digital (AID) converters and sends data to output devices such as LEDs or printers.

There are two different methods by which I/O devices can be identified:

1) one that uses an 8-bit address
2) one that uses a 16-bit address.

1. I/Os with 8-bit Addresses (Peripheral-Mapped I/O)

In this case, the MPU uses eight address lines to identify an I/O device.

This is known as peripheral-mapped I/O (or I/O-mapped I/0).

This is an 8-bit numbering system for I/Os used in conjunction with Input and Output instructions.

he eight address lines can have 256 (2^8 combinations) addresses; thus, the MPU can identify 256 input devices and 256 output devices with addresses ranging from OOH to FFH.

The entire range of I/O addresses from 00H to FFH is known as an I/O map, and individual addresses are referred to as I/O device addresses or I/O port numbers.

The steps in communicating with an I/O device are summarized as follows:

1. The MPU places an 8-bit address on the address bus, which is decoded by external decode logic.
2. The MPU sends a control signal (I/O Read or I/O Write) and enables the I/O device.
3. Data are transferred using the data bus.

2. I/Os with 16-bit Addresses (Memory-Mapped I/O)

In this type of I/O, the MPU uses 16 address lines to identify an I/O device.

An I/O is connected as if it is a memory register. This is known as memory-mapped I/O.

The MPU uses the same control signal (Memory R/W) and instructions as those of memory.

In memory-mapped I/O, the MPU follows the same steps as if it is accessing a memory register.