Computer Architecture

Lecture 1
Introduction to Microprocessor

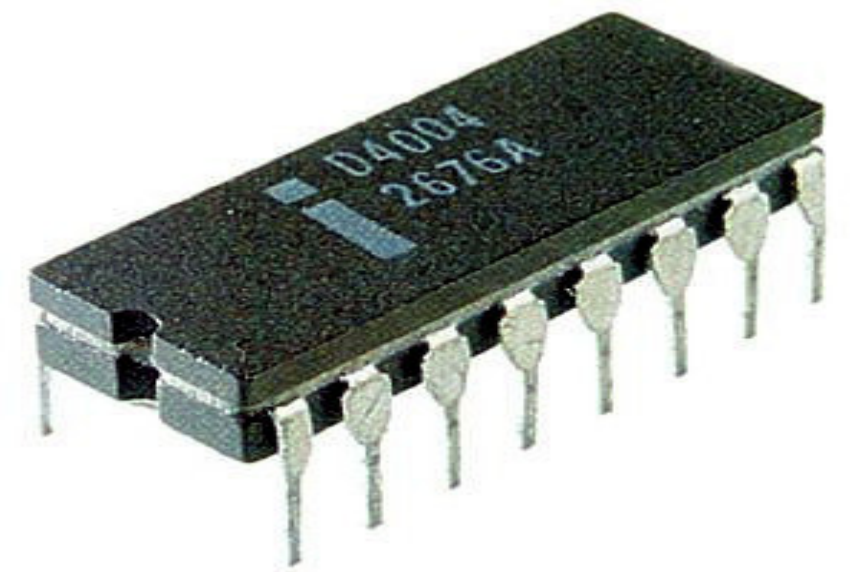
Asst. Lect.:Noor H. Hassoon

Introduction

- A microprocessor is a very small electronic circuit typically inch (12mm) across.
- It is easily damaged by moisture or abrasion so to offer it some protection it is encapsulated in plastic or ceramic.
-
- To provide electrical connections directly to the circuit would be impractical owing to the size and consequent fragility, so connecting pins are moulded into the case and the microprocessor then plugs into a socket on the main circuit board.

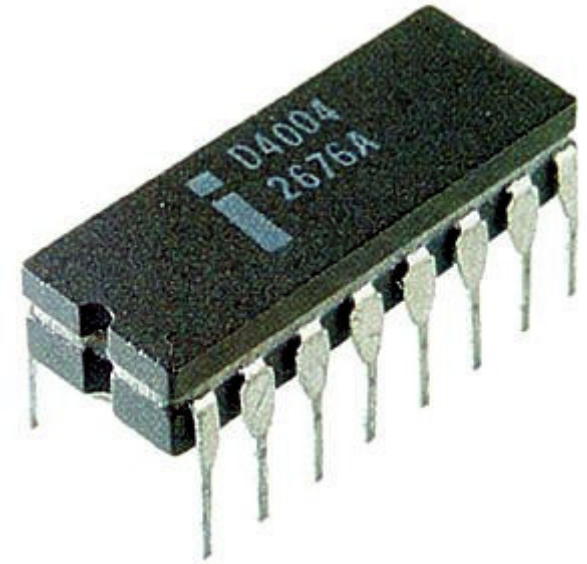
Introduction

- The size, shape and number of pins on the microprocessor depend on the amount of data that it is designed to handle. The trend, as in many fields, is forever upward. Typical microprocessor is shown in Figure below.



Introduction

- **Evolution Of Microprocessors**
  - The first microprocessor was introduced in the year 1971. It was introduced by Intel and was named Intel 4004.
  - Intel 4004 is a 4 bit microprocessor and it was not a powerful microprocessor. It can perform addition and subtraction operation on 4 bits at a time.



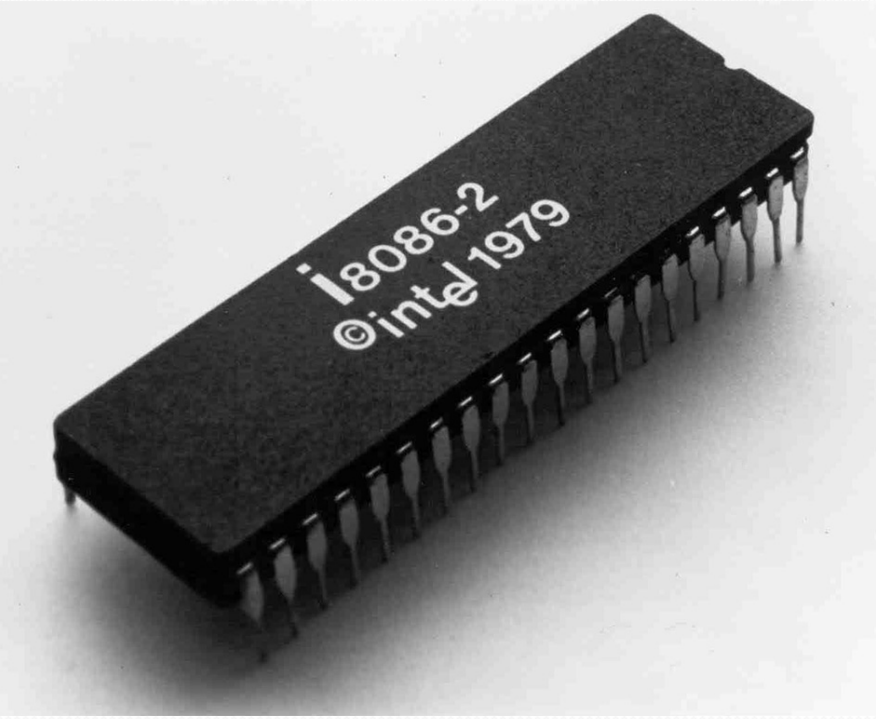
Introduction

- However it was Intel's 8080 was the first microprocessor to make it to Home computers. It was introduced during the year 1974 and it can perform 8 bit operations.

- Then during the year 1976, Intel introduced 8085 processors which is nothing but an update of 8080 processors.8080 processors are updated by adding two Enable/Disable Instructions, Three added interrupt pins and serial I/O pins.

Introduction

- Intel introduced 8086 pins during the year 1976. The major difference between 8085 and 8086 processor is that 8085 is an 8 bit processor, but 8086 processor is a 16 bit processor.



Introduction

- The **greatest advantage** of the above processors are that it do not contain Floating point instructions. Here floating point refers to the radix point or decimal point. For example: 123.456 is a floating point representation. Processors such as 8085 and 8086 do not support such representations and instructions.

- Intel later introduced 8087 processor which was the first math co-processor and later the 8088 processor which was incorporated into IBM personal computers.

Introduction

- As the years progressed lots of processors from 8088, 80286, 80386, 80486, Pentium II, Pentium III, Pentium IV and now Core2Duo, Dual Core, Quad core processors and core I3, core I5 and core I7 are the latest in the market.

- Apart from Intel, there are some other manufacturers who produce the CMOS version of 8085 microprocessor. Such manufacturers are called second source manufacturers.

Introduction

- The second source manufacturers include: AMD ,Mitsubishi ,NEC, OKI, Toshiba and Siemens.
- CMOS stands for complementary metal oxide semiconductor. It is a technology used in Microprocessors and Microcontrollers for making Integrated circuits
- The devices which are made of CMOS have high immunity towards noise and the static power consumption is low.

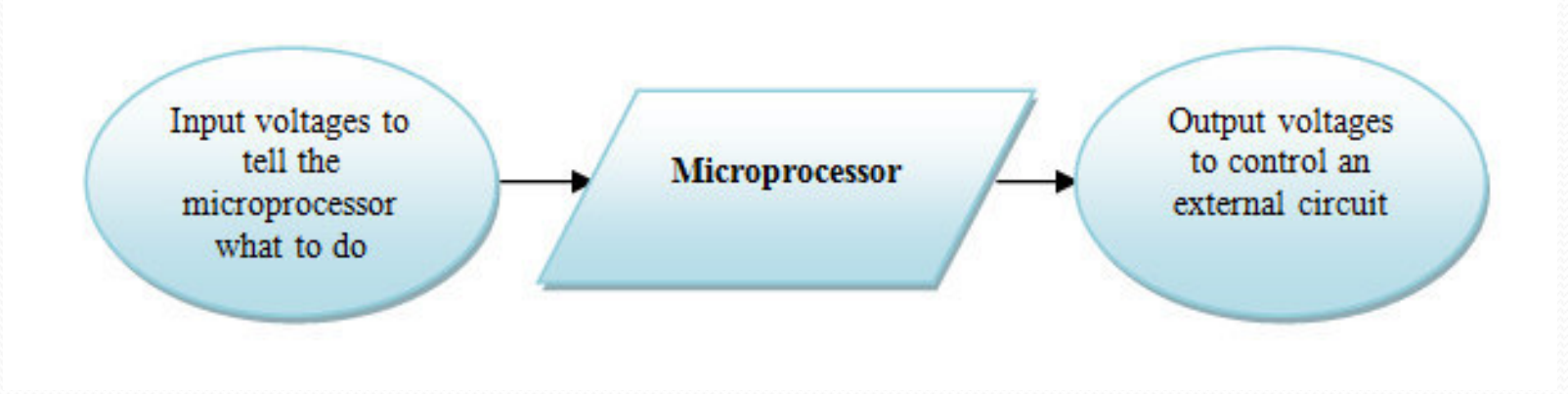
Introduction

A microprocessor system:

- Like any other system, a microprocessor has inputs, outputs and a process.
- The inputs and outputs of a microprocessor are a series of voltages that can be used to control external devices.
- The process involves analyzing the input voltages and using them to decide on the required output voltages.

Introduction

- The decision is based on previously entered instructions that are followed quite blindly, sensible or not.



Introduction

Terminology:

An electronic circuit fabricated out of a solid block of semiconductor material. This design of circuit, often called a solid state circuit, allows for very complex circuits to be constructed in a small volume. An integrated circuit is also called a (chip).

Introduction

Microprocessor-based system

- This is any system that contains a microprocessor, and does not necessarily have anything to do with computing. In fact, computers use only a small proportion of all the microprocessors manufactured.

- For example: Our garage door opening system is a microprocessor-based system or is sometimes called a microprocessor- controlled system.

Introduction

Microcomputer

- The particular microprocessor-based systems that happen to be used as a computer are called microcomputers. The additional circuits required for a computer can be built into the same integrated circuit giving rise to a single chip microcomputer.

Microcontroller

- This is a complete microprocessor-based control system built onto a single chip. It is small and convenient but doesn't do anything that could not be done with a microprocessor and a few additional components.

Introduction

CPU and MPU

- **C.P.U.=central processing unit**.
  - A central processing unit (CPU), or sometimes simply processor, is the component in a digital computer that interprets computer program instructions and processes data.

- **M.P.U=micro processing unit.**
  - A microcontroller (or MCU) is a computer-on-a-chip used to control electronic devices.

Introduction

- From the above terminologies, we can draw the following block diagram to represent the microcomputer and a microprocessor-based system is apart from it. Microcomputer a computer with a microprocessor as its CPU, and includes memory, I/O etc.

- Inside the Microprocessor, Internally, the microprocessor is made up of three main units represented by the Arithmetic/Logic Unit (ALU), the Control Unit (CU), and an array of registers for holding data while it is being manipulated.

Introduction

That mean the Microprocessor is silicon chip which includes ALU, register circuits & control circuits. While Microcontroller silicon chip which includes microprocessor, memory, and I/O in a single package.

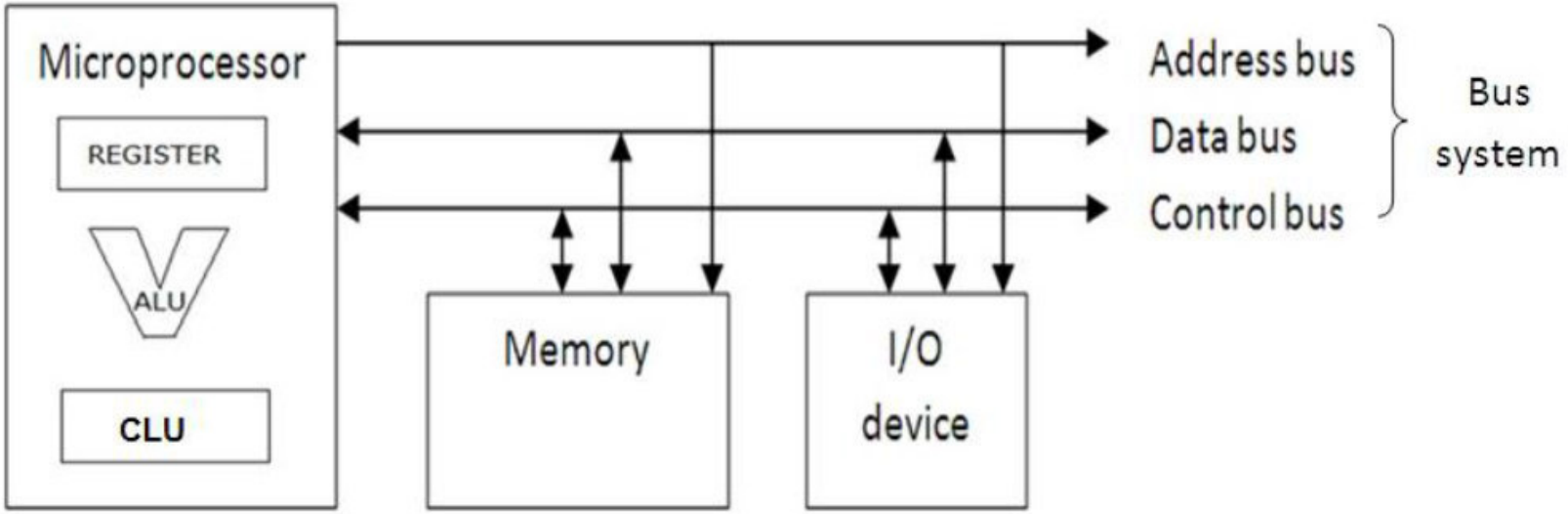


Figure 1.1 Microprocessor-based system

Introduction

Micro (small)

- The word micro is used in electronics and in science generally, to mean: one-millionth, or 1 : 106. It has also entered general language to mean something very small like a very small processor or microprocessor. It has also become an abbreviation for micro-processor, microcomputer, microprocessor-based system or a micro- controller, indeed almost anything that has (micro) in its name

Introduction

- In the scientific sense, the word micro is represented by the Greek letter ( ). It was only a small step for microprocessor to become abbreviated to P. Some confusion can arise unless we make sure that everyone concerned is referring to the same thing.


The End

Computer Architecture

Lecture 2
Registers and Memories

Asst. Lect.:Noor H. Hassoon

Registers and Memories

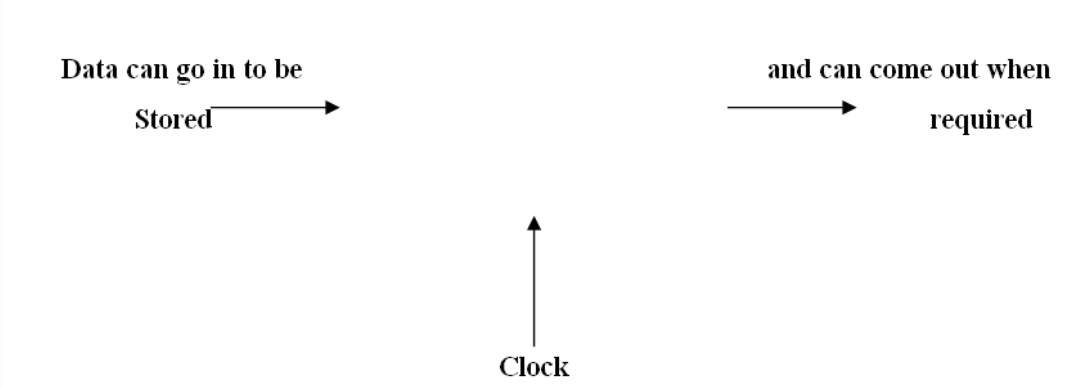
- The logic gates occur in their millions in microprocessors and in the surrounding circuitry. They are to be found in all microprocessors from the oldest and simplest.

- When logic gates are used in a microprocessor, they are usually grouped together into circuits, called flip-flops, each one being able to store a single binary digit.

Registers and Memories

A flip-flop or bistable:

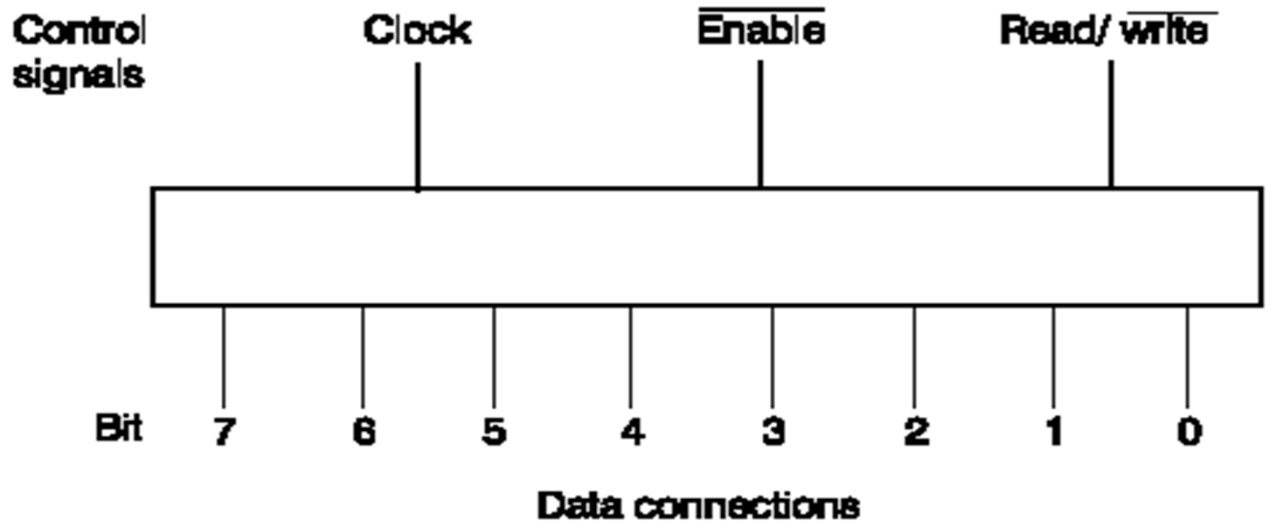
- A flip-flop or bistable is a circuit that can store a single binary bit either 0 or 1. One useful characteristic of the flip-flop is that it can only have an output of 0 or 1. It cannot hover somewhere in between. The flip-flop is shown in following figure. The purpose of the clock input is to tell the flip-flop when to accept the new input level.



Registers and Memories

A register:

- A register is just a collection of flip-flops. A flip-flop can only store one bit so to handle 32 bits at a time we would need 32 flip-flops and would refer to this as a 32-bit register. To save space, following figure shows an 8-bit register.



Registers and Memories

- The register has two distinct groups of connections: the data bits 0 to 7 and the control signals. The data connections or data lines carry the binary levels in or out of the register.

- The number of data lines determines the size of the register so a 64-bit register would have 64 data connections.

Registers and Memories

What are registers for?

- Registers are storage areas inside the microprocessor. Almost the whole of the microprocessor is made of registers. They store the data that is going to be used, they store the instructions that are to be used and they store the results obtained.

- Nearly all registers involve tri-state buffers to control the direction of data flow.

Registers and Memories

How long can it be stored?

- It will be stored until the power supplies are removed ‘either’ by an equipment fault or, more usually by the system being switched off. The data does not deteriorate in storage.

Registers and Memories

Memories:

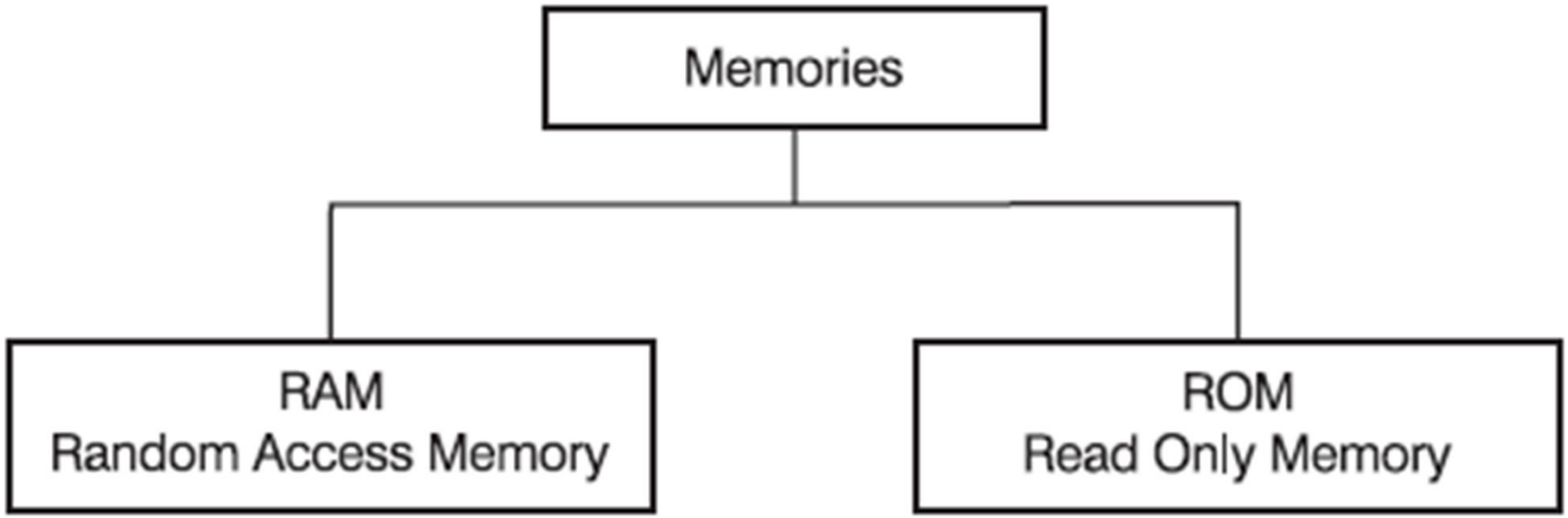
- The function of a memory is to store information almost the same as we said for the register. Generally, a register lives within the microprocessor and stores small quantities of data for immediate use and it can do useful little tricks like shift and rotate. **A memory is designed for bulk storage of data.**

Registers and Memories

- Some types can remember the data even when the power is switched off. The ability to remember data after the power is switched off is the dividing line between the two main types of memory.

- If it loses its data when the power is switched off, then we call the memory RAM or volatile memory. If it can hold on to the data without power, we call it ROM or non-volatile memory (volatile means able to evaporate). This is seen in following figure.

Registers and Memories



Registers and Memories

Random Access Memory (RAM) :

- It is used to store user programs and data, and can be altered at any time, i.e. temporary storage. The information stored in RAM or RWM can be easily read and altered by the CPU. The contents (data or programs) stored is lost if power supply to this chip is turned off.

Registers and Memories

- The letters RAM stands for Random Access Memory. The memory comes in an integrated circuit looking like a small microprocessor and is usually called a memory chip.

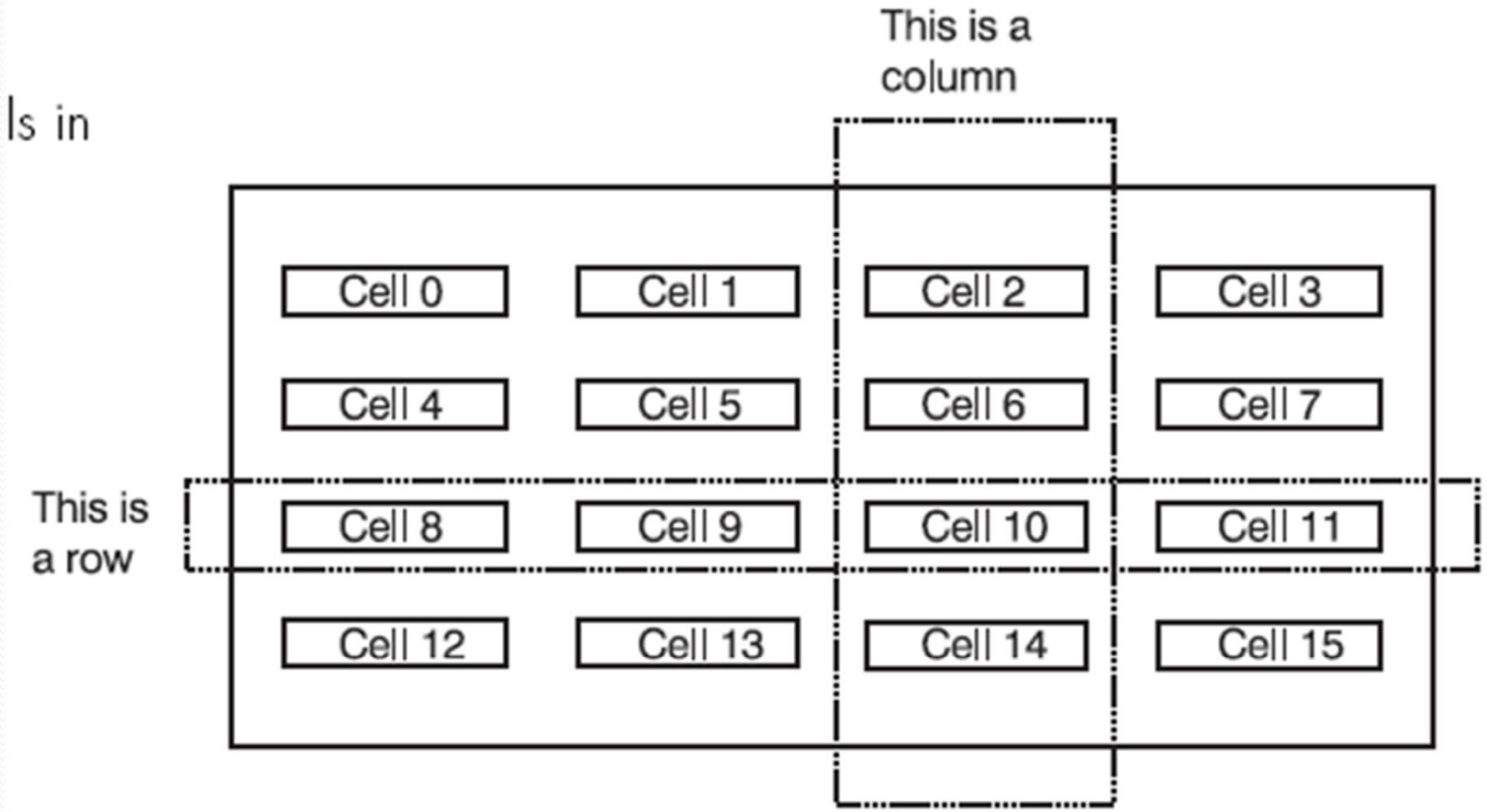
- Inside, there are a large number of registers, hundreds, thousands, millions depending on the size of the memory.

Registers and Memories

- Incidentally, when we are referring to memories, we use the word ‘cell’ instead of register even though they are the same thing.

- So, each of the internal cells may have 4, 8, 16, 32, or 64 bits stored in flip-flops. Following figure shows the register layout in a very small memory containing only 16 cells or locations, each of which can hold 4 bits and is given a memory number or address.

Registers and Memories



The cells or 'memory locations' are numbered from 0 to 15.
Each holds binary data – anywhere between 1 and 8 bits in each.

Registers and Memories

This RAM word:

- In prehistoric computing days, the memory would be loaded in order. The first group of bits would go into location 0, the next would go into location 2, then location 3 and so on rather like a shift register.

- This meant that the time to load or recover the information would increase as we started to fill the memory and have to move further down the memory.

Registers and Memories

- This was called sequential access memory (or serial access memory), abbreviated to SAM. This was OK when a large computer may hold 256 bits of information but would be impossibly slow if we tried this trick with a gigabyte.

- To overcome this problem, we developed a way to access any memory location in the same amount of time regardless of where in the memory it happens to be stored.

Registers and Memories

- This system was called random access memory or RAM. All memory, whether volatile or non-volatile is now designed as random access memory so it would be much better to divide the two types of memory into read/write and read only memory. But it won't happen, RAM is too firmly entrenched.

Registers and Memories

Accessing memory:

- Each location in a memory is given a number, called an address. In previous figure, the 16 locations of memory would be numbered from 0 to 15, or in binary 0000_11112. The cells are formed into a rectangular layout, in this case a (4*4) square with four columns and four rows.

Registers and Memories

- To use a cell, the row containing the cell must be selected and the column containing the cell must also be activated. The shaded cell in following figure has the address 0110 which means that it is in row 01 and in column 10.

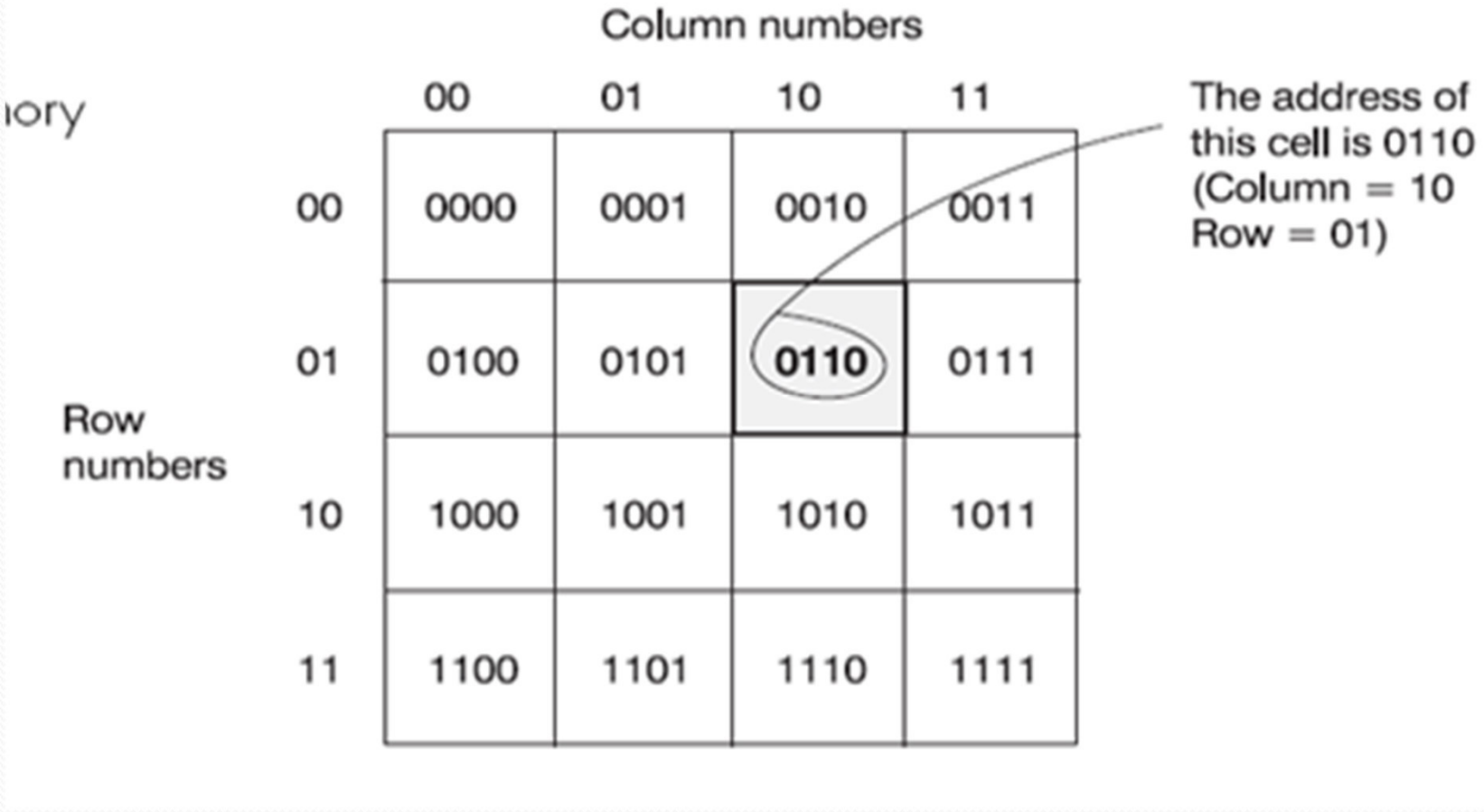
- To access this cell we need to apply the binary address to the row and column decoders.

Registers and Memories

- When the address 0110 is applied, the first half of the address, 01, is applied to the row decoder and the second half of the address is applied to the column decoder.

- A decoder circuit is a small logic circuit that, when fed with the address of the location, is able to switch on the appropriate row and column.

Registers and Memories



Registers and Memories

- The maximum number of locations that can be addressed will depend on the number of bits in the address. We have already seen that a 4-bit address can access 16 locations.

- This was because 24 = 16, so, generally 2n = number of locations where n is the number of bits in the address. To take a more realistic example, if we had 20 address lines we would have 220 = 1048576 or 1Meg locations.


The End

Computer Architecture

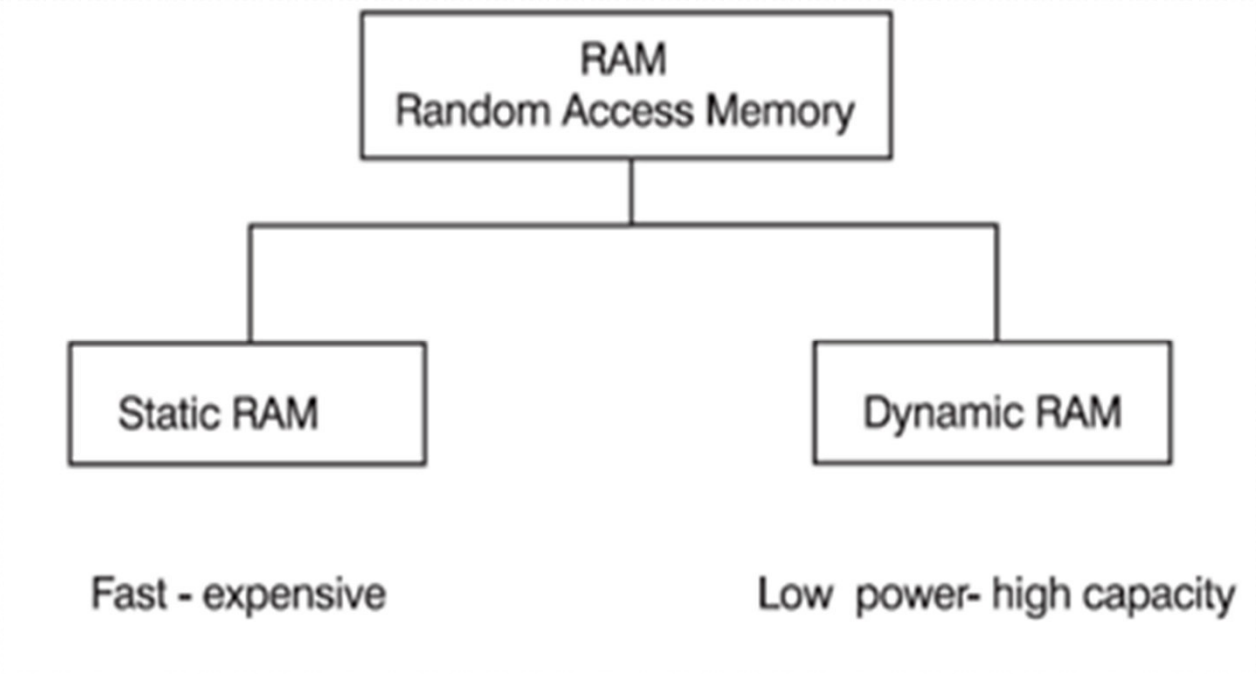
Lecture 3
Registers and Memories

Asst. Lect.:Noor H. Hassoon

Registers and Memories

Two types of RAM:

- Ram chips can be designed in two different forms which we call static RAM (SRAM) and dynamic RAM (DRAM), as seen in following figure.



Registers and Memories

Static RAM (SRAM):

- SRAM is a type of RAM (cache memory) and it is a volatile memory, which looses its data when the power is turned off. In a SRAM, each bit that stores data is made up of four or six transistors that make up a flip-flop.

- There are additional transistors that are used to control read and write accesses of storage cells. Even though typical SRAMs use six transistors to store each bit, there are SRAMs that use eight, ten or more transistors to store a single bit

Registers and Memories

- When the number of transistors is reduced, the size of the memory cell decreases. Each SRAM cell can be in three different states called read, write and standby.

- A cell is in the reading state when data has been requested and it is in writing state when the data in the cell is modified. The cell is in the standby state when it is idling.

Registers and Memories

Dynamic RAM (DRAM)

- DRAM is also a volatile memory that uses separate capacitors to store each bit. Capacitors when not charged represent the value 0 of a bit and when charged represent the value 1. Since the capacitors discharge with time, they need to be refreshed periodically to maintain the values stored in them.

- Each memory cell in a DRAM consists of a capacitor and a transistor and these cells are arranged in a square array.

Registers and Memories

- DRAMS are widely used for main memories in personal computers and game stations since they are cheaper. DRAMs are manufactured as integrated circuits (ICs) that come in plastic packages with metal pins that could be connected in to busses.

- Currently there are DRAMs in the market that are manufactured as plug-in modules, which are easier to handle. Single In-line Pin Package (SIPP), Single In-line Memory Module (SIMM) and Dual In-line Memory Module (DIMM) are some examples of such modules.

Registers and Memories

The basic difference between Static and Dynamic RAM lies mainly in structure and work principal.

1. Firstly the main difference in the structure varies due to transistor and capacitor number and setting as just three to four transistors are required for a Dynamic RAM, but six to eight MOS transistors are necessary for a Static RAM.

2. Secondly Dynamic RAM memory can be deleted and refreshed while running the program, but in case of Static RAM it is not possible to refresh programs.

Registers and Memories

3. Data is stored as a charge in a capacitor in Dynamic RAM, where data is stored in flip flop level in Static RAM.

4. For refreshing a data another capacitor is required in case of Dynamic capacitor, but no refreshing option is available in Static RAM.

5. A Dynamic RAM possesses less space in the chip than a Static RAM.

Registers and Memories

6. Dynamic RAM is used to create larger RAM space system, where Static RAM create speed- sensitive cache.
7. Static ram is 4 times more expensive than Dynamic RAM.
8. Dynamic RAM consumes less power than Static RAM.
9. For accessing a data or information, Static RAM takes less time than Dynamic RAM.
10. Dynamic RAM has higher storage capacity. In fact it can store 4 times than Static RAM.

Registers and Memories

Memory Organization:

- A memory contains a number of cells or registers that, themselves store a number of bits. The memory organization is always quoted as number of locations x bits stored in each, so this memory would have an organization of anywhere between 16 * 1, 16 * 4 or 16 * 8.

Registers and Memories

- Static RAMs usually store 8 bits in each location so a typical chip size would be 131072 * 8 giving a total storage capacity of 1 048 576 bits. This is often referred to as 128K * 8.
-
- Dynamic RAMs store either 1 or 4 bits in each location. One bit in each is very popular, so a typical chip organization would be 1048576*1 which, as we can see, would actually hold the same total number of bits as the example SRAM – it's just the organization that has been changed.

Registers and Memories

Read Only Memory (ROM)

- It is used to store programs and data that need not to be altered, i.e. permanent storage. Programs and data stored in ROMs can only be read by the CPU. Special equipment is used to write programs and data into the ROMs.

Types of ROM

- There are different variations on the classic ROM chips which were manufacturer produced and could not change. The most common are:

Registers and Memories

1. Programmable Read-Only Memory (PROM)

- This type of ROM can be re-programmed by using a special device called a PROM programmer. Generally, a PROM can only be changed/updated once.

2. Erasable Programmable Read-Only Memory (EPROM)

- This type of ROM can have its contents erased by ultraviolet light and then reprogrammed by an RPROM programmer.

Registers and Memories

- This procedure can be carried out many times; however, the constant erasing and rewriting will eventually render the chip useless.

3. Electrically Erasable Programmable Read-Only Memory (EEPROM)

- This type of ROM works in a similar way to Flash memory in that it can its contents can be 'flashed' for erasure and then written to without having to remove the chip from its environment.

Registers and Memories

- EEPROMs are used to store a computer system's BIOS, and can be updated without returning the unit to the factory.
- In many cases, BIOS updates can be carried out by computer users wishing a BIOS update.

Registers and Memories

Flash Memory

- Flash memory is an example of quite a recent type of storage technology known as solid state devices. This type of portable storage has become very popular because of its low price and high storage capacity compared to its rivals, e.g. floppy disk.

Registers and Memories

- Unlike ROM, flash memory can be read form and written to and unlike RAM does not require power to retain its data.

- Although these devices typically cannot hold as much data as hard disks, CD-ROMs and DVDs, the storage capacity is continually increasing.


The End

Computer Architecture

Lecture 4
Microprocessor Basics

Assit. Lect.:Noor H. Hassoon

Microprocessor Basics

- At present there are many types and sizes of computers available. These computers are designed and constructed based on digital and Integrated Circuit (IC) fabrication technology.

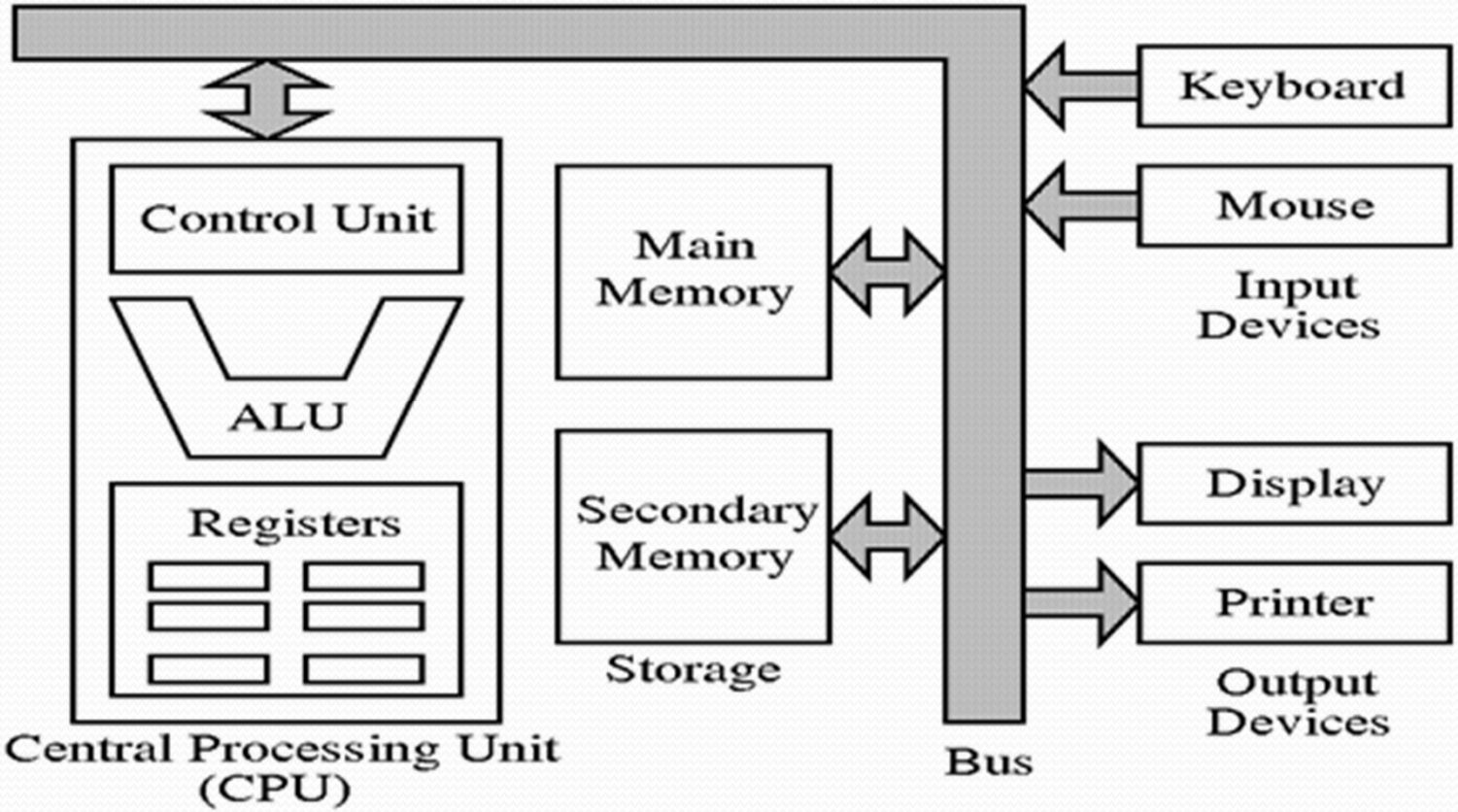
- A digital computer is a machine that can be used to solve problems for people by carrying out the task by following the instructions given to it. A sequence of instructions describing how to perform a certain task or job is called a program.

Microprocessor Basics

- A personal computer (PC) uses its memory to store the instructions, typically memory size is more than 32Mb of RAM. PCs were used for word processing, spreadsheets, and numerous highly interactive applications.

Microprocessor Basics

- Following figure show the structure of PC.



Microprocessor Basics

Structure of a PC:

- All types of computers, regardless of their size, type and use, consist of at least the following devices:-

1. Microprocessor or Central Processing Unit (CPU)
2. Storage device (Memory)
3. Input/Output devices (I/O)
4. Instructions or programs

Microprocessor Basics

- When these devices are connected and make to work together as a system, it is called a microcomputer system.

- The CPU, memory and I/O are called hardware since they are physical devices, whereas the instructions or programs (non-physical component) is called software.

Microprocessor Basics

Microprocessor or Central Processing Unit (CPU):

- **A microprocessor is a multipurpose, programmable logic device (IC)** that reads binary instructions from a storage device **called memory**, accepts binary data as input and **processes data according to those instructions**, and provide results as output.

- A multipurpose device means it can be used to perform various sophisticated computing tasks or functions, as well as simple tasks.

Microprocessor Basics

- **A programmable device** means that it can be instructed to perform given tasks within its capability. Today's microprocessor is designed to understand and execute many binary instructions.

- Microprocessor is also called Central Processing Unit (CPU) since it is the functional center of the computer system and it is used to process data.

Microprocessor Basics

- The CPU is the functional center of the microcomputer system. Its internal construction can be broadly divided into three sections : **Control section**, **Arithmetic and Logic Unit** and **Register section.**

1.Control Section:

- The control section/unit is the part of the microcomputer that controls its basic operations. It is made up of the control signal generating circuitry (clock) and the command (instruction) decoder.

Microprocessor Basics

- The control section fetches pre-programmed instructions from memory (op-code fetch cycle) as needed and temporarily stores them in the command register (also known as Instruction Register IR).

- These instructions are then decoded by the operation decoder (decode cycle), which sends control signals to the relevant parts of the microcomputer system (via the system busses) to cause them to carry out the required operation (execute cycle).

Microprocessor Basics

- The timing with which these control signals are generated is determined by the clock. The number of T-states tells the time taken for the CPU to execute that particular instruction.

- The major types of operations controlled by the control signals are:-

1. Sending of data from one part of the microcomputer to another (read or write cycle).

Microprocessor Basics

2. Inputting and outputting of data to/from the microcomputer (I/O read or write cycle).

2. Arithmetic and Logic calculations.

2. Halting of computer instructions.

3. Jumping to another instruction during running (execution) of a program.

Microprocessor Basics

2. Arithmetic and Logic Unit (ALU):

- This section, under the control of control section carries out the actual processing of data, normally describe as data manipulation.

- This consists largely of arithmetic operations (ADDition, SUBtraction, INCrementing, DECrementing etc) and logical operations (ANDing, ORing, XORing, NOTing etc).

Microprocessor Basics

The ALU carries out these operations in the following manner :-

1. Stores data fetched from memory or I/O in the registers
2. Fetches this data as needed from the registers and/or from relevant accumulators.
3. Send this data either to its arithmetic circuitry or logical circuitry, where necessary, where the necessary arithmetic or logical operations are carried out.
4. Send results of its arithmetic or logical operation to relevant accumulator, to the memory, or to the I/O interfaces.

Microprocessor Basics

3. Register Section:

- The register section/array consists completely of circuitry used to temporarily store data or program codes until they are sent to the ALU or to the control section or to memory.

- The number of registers are different for any particular CPU, and the more register a CPU have will result in easier programming tasks.

Microprocessor Basics

System Bus:

- The three components of the microcomputer system is connected by three buses, also known as system bus. These busses are used to transfer information (data) internally and externally to the microprocessor.

Microprocessor Basics

1. Address Bus:

- The address bus is 'unidirectional', over which the microprocessor sends an address code to the memory or input/output. The size (width) of the address bus is specified by the number of bits it can handle.

- The more bits there are in the address bus, the more memory locations a microprocessor can access. A 16 bit address bus is capable of addressing 65,536 (64K) addresses.

Microprocessor Basics

2. Data Bus:

- The data bus is two ways 'bi-directional', on which data or instruction codes are transferred into the microprocessor or on which the result of an operation or computation is sent out from the microprocessor to the memory or input/output. Depending on the particular microprocessor, the data bus can handle 8 bit or 16 bit data.

Microprocessor Basics

3. Control Bus:

- The control bus is used by the microprocessor to send out or receive timing and control signals in order to coordinate and regulate its operation and to communicate with other devices, i.e. memory or input/output.

Microprocessor Basics

Memory

- Memory is the term used to the various storage devices in which are used to store the programs and data for the microprocessor. These storage devices are made of semiconductor devices, and also known as Primary Storage Devices

- The semiconductor memory is of 2 types that is Read Only Memory (ROM) and Read Write Memory (RWM). RWM is popularly known as Random Access Memory (RAM).

Microprocessor Basics

Input/Output:

- The input/output unit allows the microprocessor to communicate with the outside world, either to receive or to send data.

- Most of the time, the input/output unit will also act as an interface for the microprocessor, that is to convert the data into a suitable format for the microprocessor.

- Data can be in the form of parallel (8 bit) or serial format (single line).

Microprocessor Basics

- Input devices are devices that input data or send data to the computer. Input devices are such as **keyboard**, **punched card readers**, **sensors**, **switches**, **Mouse**, etc.

- Output devices are devices that output data or perform various operations under the control of the CPU. Output devices are LEDs, speaker, printer, etc

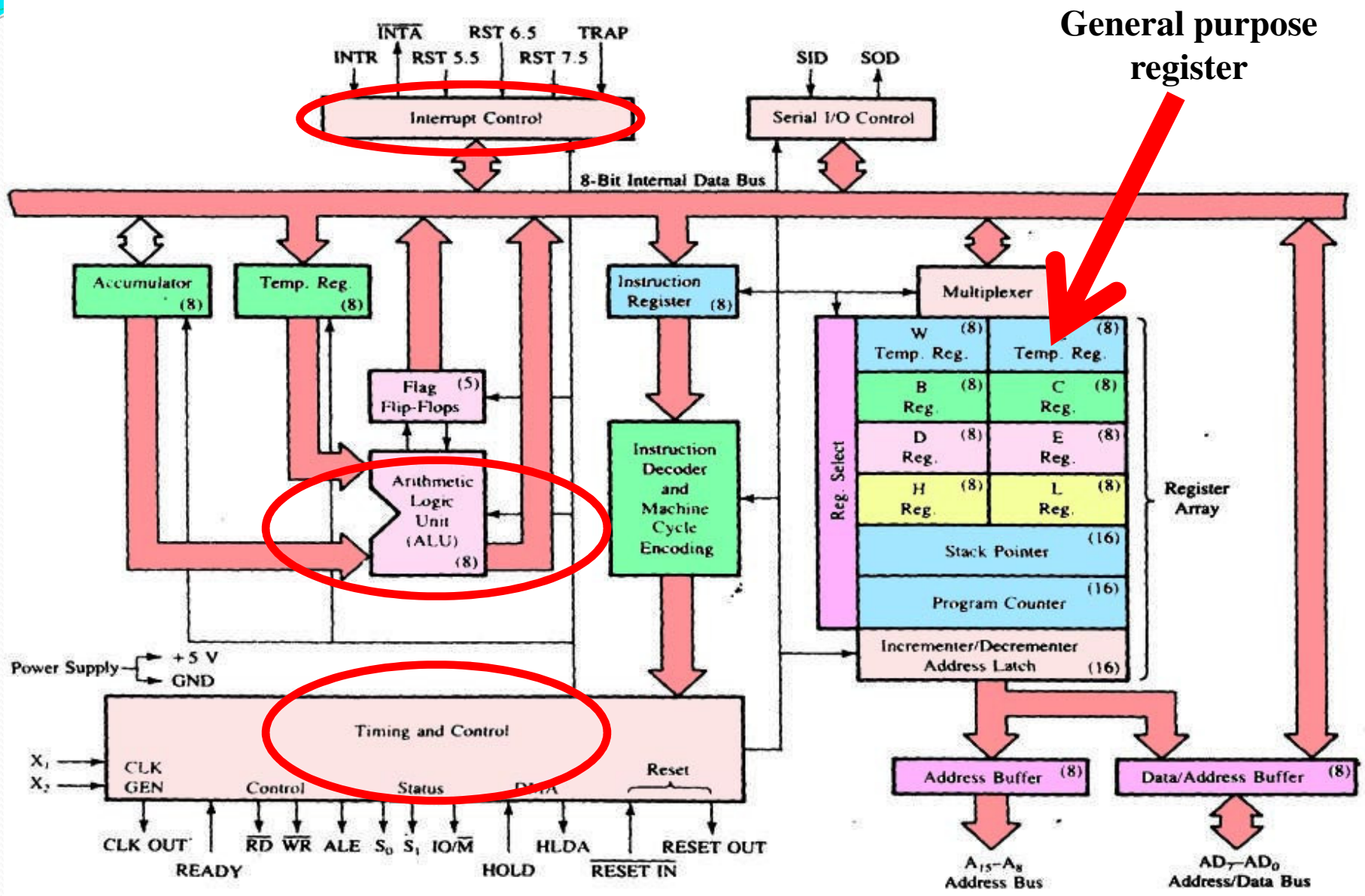

The End

Computer Architecture

Lecture 5
8085 Architecture and Programming

Assit. Lect.:Noor H. Hassoon

Internal Architecture of 8085 Microprocessor



Internal Architecture of 8085 Microprocessor

1- Control Unit:

- Generates signals within microprocessor to carry out the instruction, which has been decoded. In reality causes certain connections between blocks of the microprocessor to be opened or closed, so that data goes where it is required, and so that ALU operations occur.

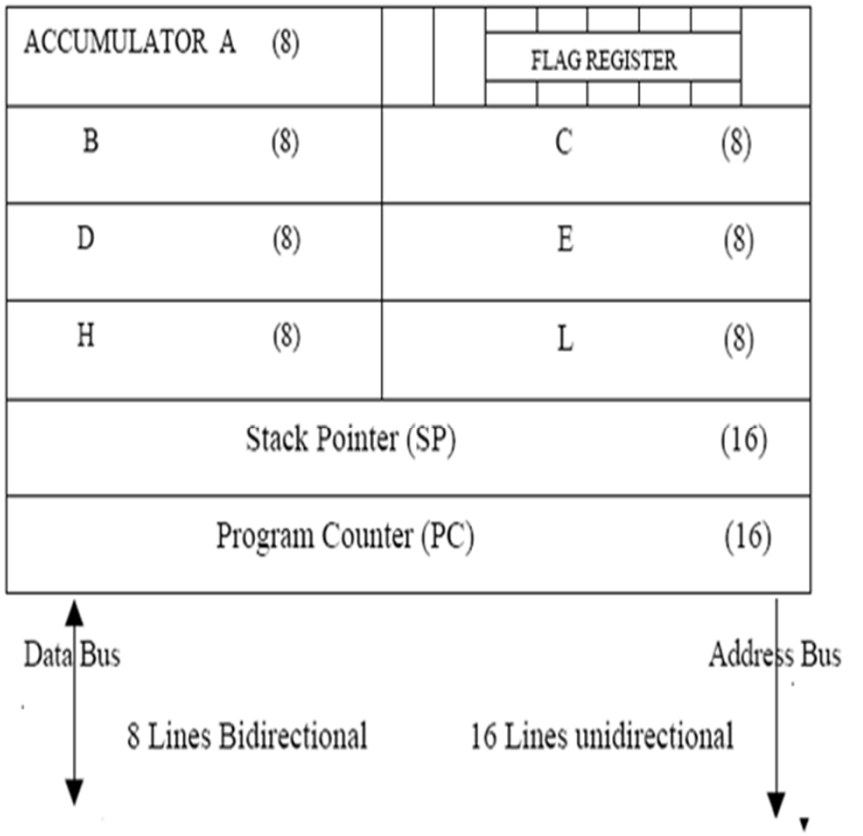
2- Arithmetic Logic Unit:

- The ALU performs the actual numerical and logic operation such as ‘add’, ‘subtract’, ‘AND’, ‘OR’, etc. Uses data from memory and from Accumulator to perform arithmetic. Always stores result of operation in Accumulator.

Internal Architecture of 8085 Microprocessor

3- Registers:

- The8085/8080A-programming model includes six registers, one accumulator, and one flag register, as shown in previous figure. In addition, it has two 16-bit registers: the stack pointer and the program counter. They are described briefly as follows.



Internal Architecture of 8085 Microprocessor

- The 8085/8080A has six general-purpose registers to store 8-bit data; these are identified as B,C,D,E,H, and L as shown in the figure. 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 or copy data into the registers by using data copy instructions.

Accumulator:

- The accumulator is an 8-bit register that is a part of arithmetic/logic unit (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. The accumulator is also identified as register A.

Internal Architecture of 8085 Microprocessor

FLAG REGISTER:

- The Status Flags of the 8080 and 8085 are single bits which indicate the logical conditions that existed as a result of the execution of the instruction just completed. This allows instructions following to act accordingly, such as a branch as a result of two values comparing equal.

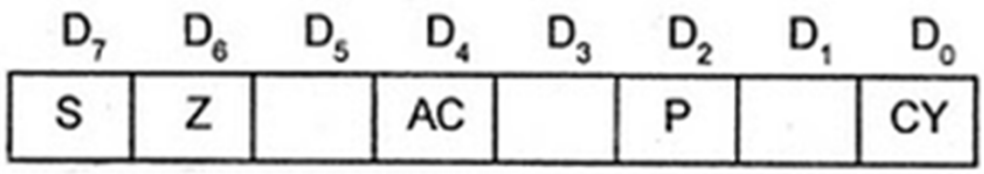


Fig 1.7 : Bit positions of various flags in the flag register of 8085

The flags are:

Internal Architecture of 8085 Microprocessor

The flags are:

- **(CY) Carry Flag**: If an arithmetic result in a carry, the flag is set otherwise the flag is reset.
- **(P) Parity Flag**: After the arithmetic and logic operation, if the result have even number of (1's) the flag is set. If it has odd number of (1's) the flag is reset.
- **(AC) Auxiliary Carry Flag**: In arithmetic operation, when a carry is generated by digit B3 and passed on the digit B4, the AC is set else it is reset.

Internal Architecture of 8085 Microprocessor

- **(Z) Zero Flag**: This flag is set if the result is zero, and reset if the result is not zero.
- **(S) Sign Flag**: After the execution the operation arithmetic or logical, if B7 of the result is 1 then the sign flag is set.

EXAMPLE:

- Write the contents of flags after executing the operation A5H + 77H

Convert from HEX to Binary system:

0111 0111

1010 0101

0001 1100 ⟶ the carry is (1)

ZF¬= 0, SF= 0, CF= 1, PF=0, AC= 0

Internal Architecture of 8085 Microprocessor

Program Counter (PC):

- This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register.
- The microprocessor uses this 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 (machine code) is being fetched, the program counter is incremented by one to point to the next memory location

Internal Architecture of 8085 Microprocessor

Stack Pointer (SP):

- The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer. The stack concept is explained in the chapter "Stack and Subroutines."

Instruction Register/Decoder:

- Temporary store for the current instruction of a program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and ‘decodes’ or interprets the instruction. Decoded instruction then passed to next stage.

Internal Architecture of 8085 Microprocessor

Memory Address Register

- Holds address, received from PC, of next program instruction. Feeds the address bus with addresses of location of the program under execution.

Control Generator

- Generates signals within microprocessor to carry out the instruction which has been decoded. In reality causes certain connections between blocks of the microprocessor to be opened or closed, so that data goes where it is required, and so that ALU operations occur.

Internal Architecture of 8085 Microprocessor

Register Selector:

- This block controls the use of the register stack in the example. Just a logic circuit which switches between different registers in the set will receive instructions from Control Unit.

General Purpose Registers:

- Microprocessor requires extra registers for versatility. Can be used to store additional data during a program. More complex processors may have a variety of differently named registers.

8085 System Bus:

- The microprocessor performs four operations primarily.

- Memory Read
- Memory Write
- I/O Read
- I/O Write

- All these operations are part of the communication processes between microprocessor and peripheral devices. The 8085 performs these operations using three sets of communication lines called buses - the address bus, the data bus and the control bus.

8085 System Bus:

1- Address Bus:

- The address bus is a group of 16 lines represent by (A0 A1…A15). The address bus is unidirectional bits flow only in one direction from the 8085 processor to the peripheral devices. The microprocessor uses the address bus to perform the first function: identifying a peripheral or memory location. Each peripheral or memory location is identified by a 16 bit address. The 8085 with its 16 lines is capable of addressing 64 K memory locations.

8085 System Bus:

2- Data Bus

- The data bus is a group of eight lines represent (D0 D1…D7) used for data flow. They are bidirectional: data flows in both direction between the 8085 microprocessor, and memory or peripheral devices. The 8 lines enable the microprocessor to manipulate 8-bit data at-a-time only ranging from 00 to FF.

8085 System Bus:

3- Control Bus

- The control bus consists of ten single lines that carry synchronization signals. These are not groups of lines like address of data bus but individual lines that provide a pulse to indicate an operation. The 8085 generates specific control signal for each operation it performs. These signals are used to identify a device type which the processor intends to communicate.


The End

Computer Architecture

Lecture 6
Instruction Set

Assit. Lect.:Noor H. Hassoon

Instruction Set

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the **instruction set**, determines what functions the microprocessor can perform. An Instruction is a command given to the computer to perform a specified operation on given data. The instructions described are of Intel 8085. These instructions are of Intel Corporation.

- They cannot be used by other microprocessor manufactures. The programmer can write a program in assembly language using these instructions.

Instruction Set

- These instructions can be classified into the following five functional categories:
  - **Data Transfer Instruction,**
  - **Arithmetic Instructions,**
  - **Logical Instructions,**
  - **Branching Instructions,**
  - **Control Instructions,**

Instruction Set

Machine Control Operations

- These instructions control machine functions such as Halt, Interrupt, or do nothing. The microprocessor operations related to data manipulation can be summarized in four functions:

1. Copying data
2. Performing arithmetic operations
3. Performing logical operations
4. Testing for a given condition and alerting the program sequence

Instruction Set

Some important aspects of the instruction set are noted below:

1. In data transfer, the contents of the source are not destroyed; only the contents of the destination are changed. The data copy instructions do not affect the flags.

2. Arithmetic and Logical operations are performed with the contents of the accumulator, and the results are stored in the accumulator (with some expectations). The flags are affected according to the results.

3. Any register including the memory can be used for increment and decrement.

4. A program sequence can be changed either conditionally or by testing for a given data condition.

Instruction Set

Instruction Format

- An instruction is a command to the microprocessor to perform a given task on a specified data. Each instruction has two parts: the first is used to the task to be performed, called the operation code (opcode), and the second is the data to be operated on, called the operand. The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit.

Instruction Set

Instruction Word Size

- The 8085 instruction set is classified into the following three groups according to word size:

1. One-word or 1-byte instructions

2. Two-word or 2-byte instructions

3. Three-word or 3-byte instructions

- 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.

Instruction Set

1- One – Byte Instructions

- A 1-byte instruction includes the opcode and operand in the same byte. Operand(s) are internal register and are coded into the instruction. For example:

Task	Op code	Operand	Binary Code	Hex Code
Copy the contents of the accumulator in the 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
Invert (compliment) each bit in the accumulator.	CMA		0010 1111	2FH

Instruction Set

- These instructions are 1-byte instructions performing three different tasks.
  - ***In the first instruction***, both operand registers are specified.
  - ***In the second instruction***, the operand B is specified and the accumulator is assumed.
  - ***Similarly, in the third instruction***, the accumulator is assumed to be the implicit operand. These instructions are stored in 8- bit binary format in memory; each requires one memory location.

Instruction Set

2- Two – Byte Instructions

- In a two-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Source operand is a data byte immediately following the opcode. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A, Data	0011 1110	3E	First Byte
			DATA	Data	Second Byte

Instruction Set

3- Three – Byte Instructions

- In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address. opcode + data byte + data byte.

Task	Opcode	Operand	Binary code	Hex Code	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011 1000 0101 0010 0000	C3 85 20	First byte Second Byte Third Byte

Instruction Set

Data Transfer Instruction

- The data transfer instructions move data between registers or between memory and registers.

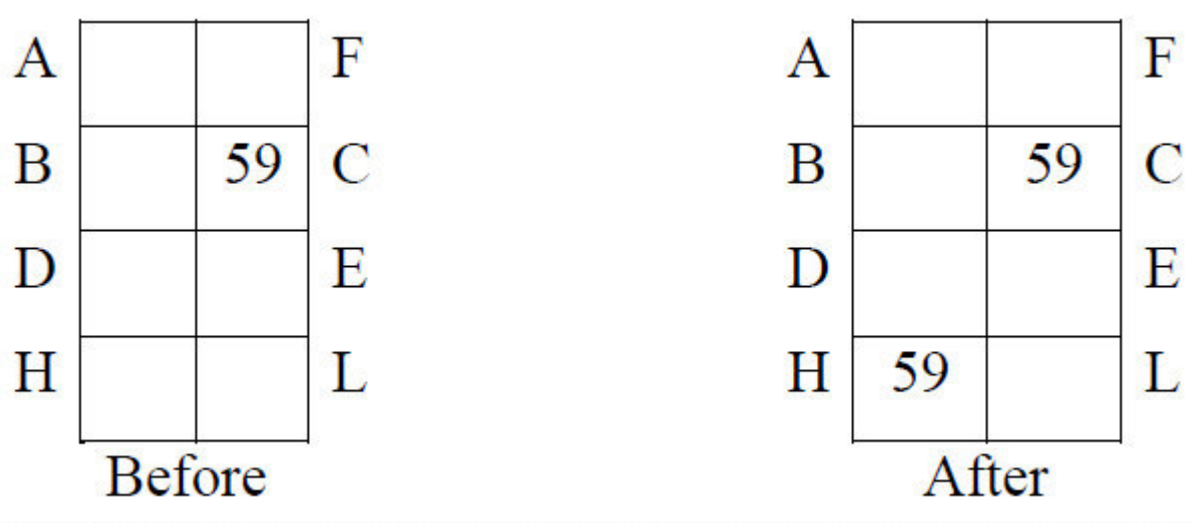
Copy from source to destination		
MOV	Rd, Rs	This instruction copies the contents of the source register into the destination register, the contents of Rd, M the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers. **Example:** MOV B, C or MOV B, M
	M, Rs	
	Rd, M	

Instruction Set

Example

- Copy the content of the register C to Register H

MOV H, C



Instruction Set

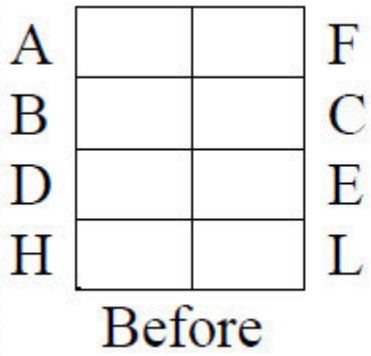
Move immediate 8-bit		
MVI	Rd, data	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers. **Example:** MVI B, 57H or MVI M, 57H
	M, data	

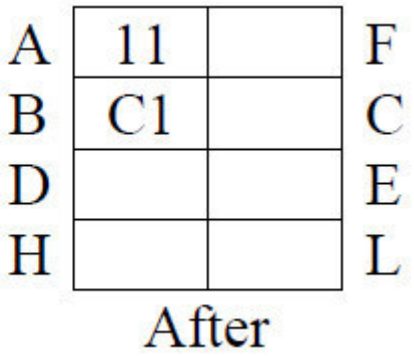
Example

- Load the register B with C1 h and the accumulator with 11 h

MVI B, C1 h

MVI A, 11h





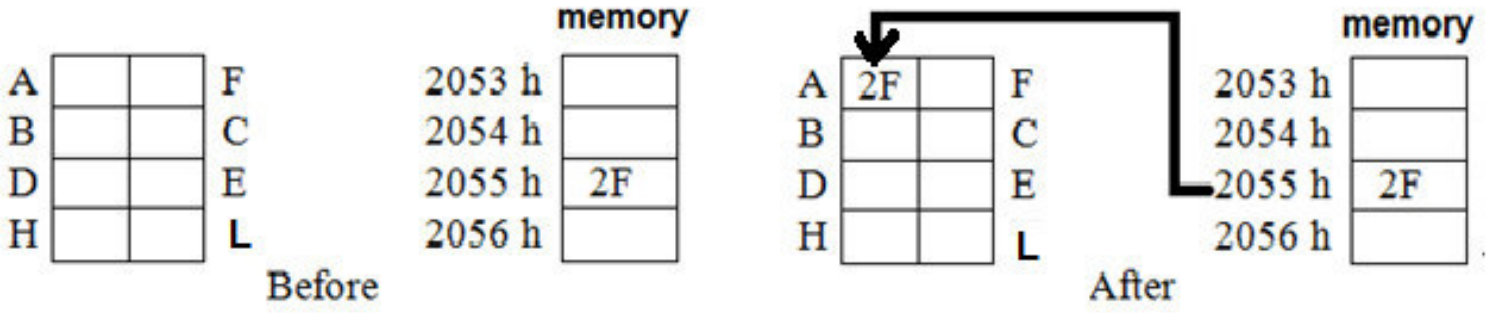
Instruction Set

Load accumulator		
LDA	16-bit address	The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. **Example:** LDA 2034H

Example

- Load the accumulator with content of memory location 2055 h

LDA 2055h



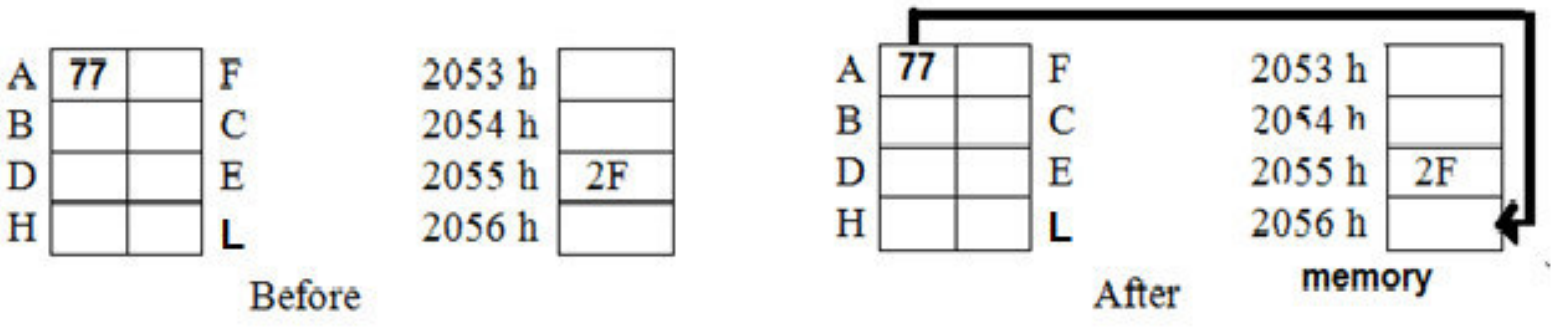
Instruction Set

Store accumulator direct		
STA	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. **Example:** STA 4350H

Example

- Store the content of accumulator to memory location 2056 h

STA 2056h



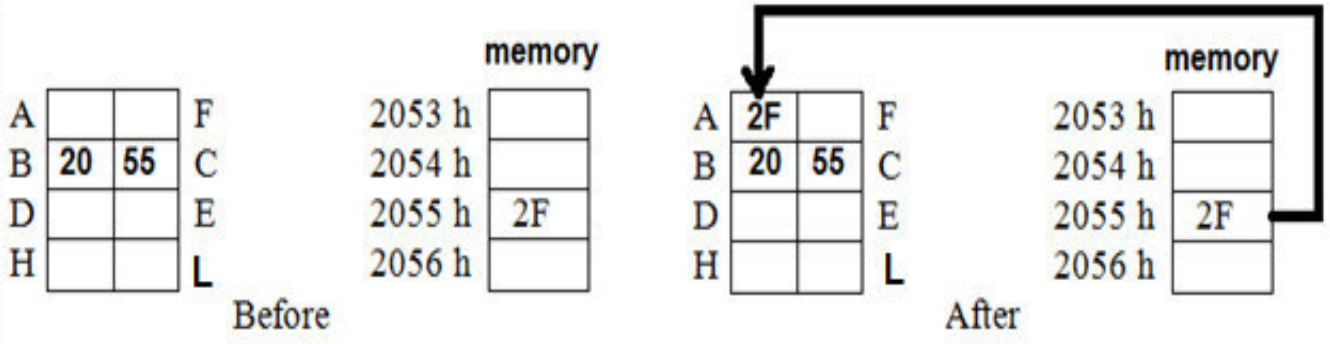
Instruction Set

Load accumulator indirect		
LDAX	B/D Reg. pair	The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. **Example:** LDAX B

Example

- Load the content of memory location to accumulator if the address specify in register pair B

LDAX B



Instruction Set

Store accumulator Indirect		
STAX	Reg. pair	The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered. **Example:** STAX B

Example

- Store the content of register H to memory location if the address specify in register pair B

```
MOV A, H
STAX B
```

Instruction Set

Load register pair immediate		
LXI	Reg. pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand. **Example:** LXI H, 2034H or LXI H, XYZ

Example

- Copy the content of register B to the memory location 2053 h

  LXI H, 2053 H

  MOV M, B

Example

- Load the memory location 2053 h with data F8 h.

  LXI H, 2053h

  MVI M, F8 h

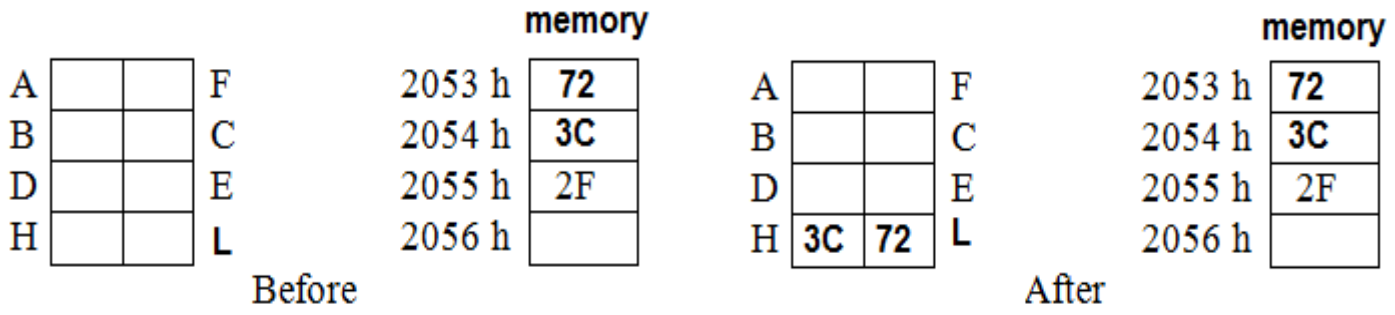
Instruction Set

Load H and L registers direct		
LHLD	16-bit address	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered. **Example**: LHLD 2040H

Example

- Load the content of the memory location 2053 h and 2054 to H and L registers

LHLD 2053h



Instruction Set

Store H and L registers direct		
SHLD	16-bit address	The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. **Example:** SHLD 2470H

Example

- Store the content of the H and L registers to memory location 2053 h and 2054

  SHLD 2053 h

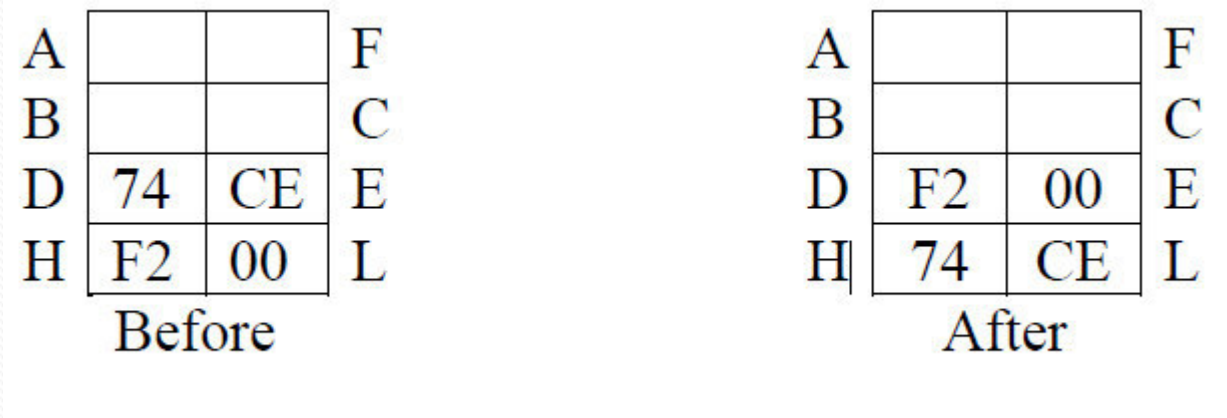
Instruction Set

Exchange H and L with D and E		
XCHG	none	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. **Example:** XCHG

Example

- Swap the content registers pair D with H

XCHG



Instruction Set

Copy H and L registers to the stack pointer		
SPHL	none	The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered. **Example:** SPHL

Example

- Load the content of the HL register pair to stack pointer

  SPHL

Instruction Set

Push register pair onto stack		
PUSH	Reg. pair	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high- order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. **Example:** PUSH B or PUSH A

Example

- Store the content of register pair D in to stack memory

  PUSH D

Instruction Set

Pop off stack to register pair		
POP	Reg. pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. **Example:** POP H or POP A

Example

- Load the register pair D from stack memory content.

  POP D

Instruction Set

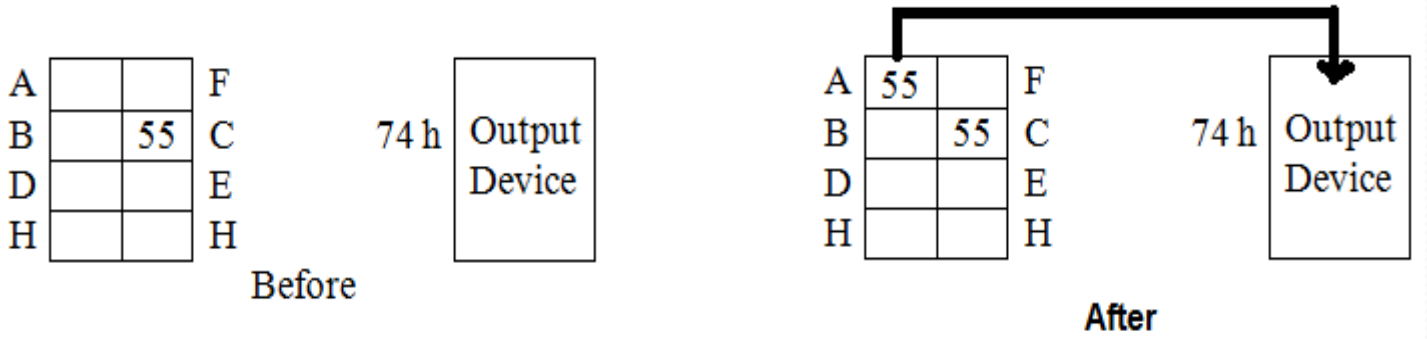
Output data from accumulator to a port with 8-bit address		
OUT	8-bit port address	The contents of the accumulator are copied into the I/O port specified by the operand. **Example:** OUT F8H

Example

- Store the content of register C to output device port 74 h

MOV A, B

OUT 74 h



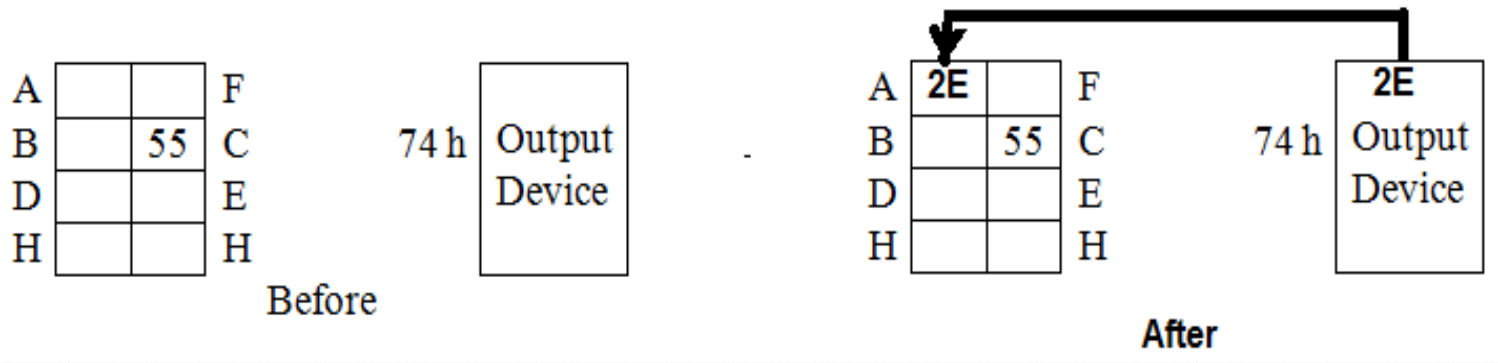
Instruction Set

Input data to accumulator from a port with 8-bit address		
IN	8-bit port address	8-bit port address The contents of the input port designated in the operand are read and loaded into the accumulator. **Example:** IN 8CH

Example

- Load the data from output device port 74 h to accumulator

IN 74 h




The End

Computer Architecture

Lecture 7
Arithmetic Instructions

Asst. Lect.:Noor H. Hassoon

Arithmetic Instructions

- Arithmetic instructions include some mathematical operations such as addition, subtraction, increment, and decrement. For addition and subtraction the quantity is always stored in accumulator. Accumulator is a register which can hold 8 bit data where the arithmetic values can be stored in it. All flags are effected by the arithmetic instruction.

- ***FLAGS REGISTER***

B7	B6	B5	B4	B3	B2	B1	B0
S	Z		AC		P		CY

- **(CY) Carry Flag**: If an arithmetic result in a carry, the flag is set otherwise the flag is reset.

Arithmetic Instructions

- **(P) Parity Flag:** After the arithmetic and logic operation, if the result have even number of (1's) the flag is set. If it has odd number of (1's) the flag is reset.
- **(AC) Auxiliary Carry Flag:** In arithmetic operation, when a carry is generated by digit B3 and passed on the digit B4, the AC is set else it is reset.
- **(Z) Zero Flag:** This flag is set if the result is zero, and reset if the result is not zero.
- **(S) Sign Flag:** After the execution the operation arithmetic or logical, if B7 of the result is 1 then the sign flag is set.

Arithmetic Instructions

1- Addition Instructions

(1)

ADD	R M	1. Add register or memory to accumulator 2. The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. 3. If the operand is a memory location, its location is specified by the contents of the HL registers. 4. All flags are modified to reflect the result of the addition. 5. Example: ADD B or ADD M

Example: Write the 8085 program to perform the following function

1. Load the number $8B_H$ in register D and register C with $6F_H$

2. Add the contents of registers and display the result at the output 01

Arithmetic Instructions

Sol:

MVI D, $8B_H$

MVI C, $6F_H$

MOV A, D

ADD C

OUT 01_H

Result in A = F2H

CY	= 0	P	= 1	AC	= 1	Z	= 0	S	= 1

Arithmetic Instructions

Example: Load the accumulator with 6CH and register D with 2EH. And then add these value use the Instruction ADD performs the addition:

```
2EH = 00101110
6CH = 01101100
-------------
9AH = 10011010
```

The accumulator contains the value 9AH following execution of the ADD D instruction. The contents of the D register remain unchanged. The condition flags are set as follows:

Carry	= 0
Sign	= 1
Zero	= 0
Parity	= 1
Aux. Carry	= 1

Arithmetic Instructions

(2)

ADC	R M	1. Add register to accumulator with carry 2. The contents of the operand (register or memory) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. 3. If the operand is a memory location, its location is specified by the contents of the HL registers. 4. All flags are modified to reflect the result of the addition. 5. Example: ADC B or ADC M

Example:

- If the memory location 2006 hold the data $(FF)_H$ and memory location 2002 holds $(01)_H$. Add these contents and if the carry is set add it then store the result in memory location 2001.

Arithmetic Instructions

LDA 2006

LXI H, 2002

ADC M

STA 2001

CY= P= AC= Z= S=

(3)

ADI 8-bit data	1. Add immediate to accumulator 2. The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. 3. All flags are modified to reflect the result of the addition. 4. Example: ADI 45

Arithmetic Instructions

Example: Write 8085 program to add the 05_H to register B that hold $F3_H$, and then output the result on port FF_H.

MOV A, B

ADI 05_H

OUT FF_H

CY= P= AC= Z= S=

(4)

ACI 8-bit data	1. Add immediate to accumulator with carry 2. The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. 3. All flags are modified to reflect the result of the addition. 4. Example: ACI 45

Arithmetic Instructions

Example: Write 8085 program to add with carry the 05_H to register B that can be load with FF_H, and then store the result in memory location $EF00_H$.

MVI B, FF_H

MOV A, B

ACI 05_H

STA $EF00_H$

CY= P= AC= Z= S=

Arithmetic Instructions

(5)

DAD	Register pair R_B R_D	1. Add register pair to H and L registers 2. The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. 3. The contents of the source register pair are not altered. 4. If the result is larger than 16 bits, the CY flag is set. No other flags are affected. 5. Example: DAD H, DAD B, DAD D

Example

- Write the 8085 program to add the content of the memory location 2000 and 2001 to the content of memory location 3000 and 3001, the store the result to memory location 4000, 4001.

Arithmetic Instructions

Sol:

LHLD 2000_H

XCHG

LHLD 3000_H

DAD D

SHLD 4000_H

2000_H	51
2001_H	1E
2002_H	FF
3000_H	44
3001_H	BC
4000_H	
4001_H	
4002_H	

NOTE

If there are carry, only carry flag is effect and other flags are not effect by this instruction

Arithmetic Instructions

2- Subtraction Instructions

- The 8085 perform subtraction by using the method of 2's complement. The 8085 performs the following steps internally to execute the instruction.

Step 1: Coverts subtracted to its 1's complement by convert all (1 to 0) and (0 to 1)

Step 2: Add 1 to 1's complement to obtain the 2's complement of the subtracted value.

Step 3: Add the 2's complement to subtract value

Step 4: Complement the carry flag

Arithmetic Instructions

(6)

SUB	R M	1. Subtract register or memory from accumulator 2. The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator 3. If the operand is a memory location, its location is specified by the contents of the HL registers. 4. All flags are modified to reflect the result of the subtraction. 5. Example: SUB B or SUB M

Example

Load the register B with 65_H and the accumulator with 97_H. Add the content of register B from the accumulator and store the result to memory location 2001_H

Arithmetic Instructions

Answer

MVI A, 97_H

MVI B, 65_H

SUB B

STA 2001_H

Step 1	Subtracted (B): 65		0 1 1 0 0 1 0 1
	1's complement of 65		1 0 0 1 1 0 1 0
Step 2	Add 1 to obtain 2's Comp	+	0 0 0 0 0 0 0 1
	2's Complement of 65		1 0 0 1 1 0 1 1
Step 3	Add to Accumulator 97_H		1 0 0 1 0 1 1 1
Step 4	Complement Cy	Cy = 1	0 0 1 1 0 0 1 0
	Result is 32_H	Cy = 0	0 0 1 1 0 0 1 0
S= 0 Z=0 CY=0 p=1 AC=0			

Arithmetic Instructions

Example: Write the 8085 program to do the following:

1. Load the Register B with 30_H, and the Register C with 39_H

2. Subtract with register C from register B and display the result on output port 06_H.

Answer

MVI B, 30_H

MVI C, 39_H

MOV A,B

SUB C

OUT 06_H

Step 1	Subtracted : 39		0 0 1 1 1 0 0 1
	1's complement of 39		1 1 0 0 0 1 1 0
Step 2	Add 1 to obtain 2's Comp	+	0 0 0 0 0 0 0 1
	2's Complement of 39		1 1 0 0 0 1 1 1
Step 3	Add to Accumulator 30_H		0 0 1 1 0 0 0 0
Step 4	Complement Cy	Cy = 0	1 1 1 1 0 1 1 1
	Result is $F7_H$	Cy = 1	1 1 1 1 0 1 1 1
S= 1 Z=0 CY=1 p=0 AC=0			

Arithmetic Instructions

Note: The subtract operation is perform by using 2's complement method. If the subtraction result in negative number, the answer in 2's complement and the carry (the Borrow flag) is set.

(6)

SBB	R M	1. Subtract register and borrow from accumulator 2. The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. 3. If the operand is a memory location, its location is specified by the contents of the HL registers. 4. All flags are modified to reflect the result of the subtraction. 5. Example: SBB B or SBB M

Arithmetic Instructions

Example: If the register B holds the FE value, subtract with barrow the content $(89)_H$ of memory location $30AB_H$ form Register B.

MOV A, B

LXI H, $30AB_H$

SBB M

(9)

SUI 8-bit data	1. Subtract immediate from accumulator 2. The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. 3. All flags are modified to reflect the result of the subtraction. 4. Example: SUI 89_H

Arithmetic Instructions

Example: Write the 8085 program to subtract the $5E_H$ from the memory location 2000_H that hold 55_H and then send the result to output port $1F_H$.

LDA 2000_H

SUI $5E_H$

(10)

SBI 8-bit data	1. Subtract immediate from accumulator with borrow 2. The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. 3. All flags are modified to reflect the result of the subtraction. 4. Example: SBI 45

Arithmetic Instructions

Example: Write the 8085 program to subtract with borrow the 47_H from register H that hold 12_H and then store the result in memory location 2100_H.

Answer

MOV A, H
SBI 47_H
STA 2100_H

3- Increment and Decrement Instruction

(11)

INR	R M	1. Increment register or memory by 1 2. The contents of the designated register or memory) are incremented by 1 and the result is stored in the same place. 3. If the operand is a memory location, its location is specified by the contents of the HL registers. 4. Example: INR B or INR M

Arithmetic Instructions

Example: Load the memory location 3012_H with $5F_H$ and the accumulator with 73_H, increment the content of the memory and then add it to the accumulator.

answer

MVI A, 73_H

LXI H, 3012_H

MVI M, $5F_H$

INR M

ADD M

Arithmetic Instructions

(12)

INX	Register Pair R_B R_D R_H	1. Increment register pair by 1 2. The contents of the designated register pair are incremented by 1 and the result is stored in the same place. 3. Example: INX H

Example: Write the 8085 program to subtract the content of memory location 2001_H from 2002_H and store the result in memory location 2003.

Answer

LXI H, 2001_H
MOV B, M
INX H
MOV A, M
SUB B
STA 2003_H

Arithmetic Instructions

(13)

DCR	R M	1. Decrement register or memory by 1 2. The contents of the designated register or memory are decremented by 1 and the result is stored in the same place. 3. If the operand is a memory location, its location is specified by the contents of the HL registers. 4. Example: DCR B or DCR M

Example: Write 8085 program to load the accumulator with 94H, registers B, C with FE, 7A respectively, then decrement the content of both B, C and add them to the Accumulator.

Arithmetic Instructions

Answer

MVI A, 94H

MVI B, FEH

DCR B

MVI C

DCR C

ADD B

(14)

DCX	Register pair R_B R_D R_H	1. The contents of the designated register pair are decremented by 1 and the result is stored in the same place. 2. Example: DCX H

Arithmetic Instructions

Example: Write the 8085 program to subtract the content of memory location 2002_H from 2001_H and store the result in memory location 2003.

Answer

LXI H, 2002

MOV M , B

DCX H

MOV A, M

SUB B

STA 2003

Arithmetic Instructions

Example: Write 8085 program to add the data $(9F, 33)_H$ from input port 00_H, 01_H. And then decrement the result and store the result in memory location $3F54_H$.

Answer

IN 00_H

MOV C, A

IN 01_H

ADD C

DCR A

STA $3F54_H$


The End

Computer Architecture

Lecture 8
Logic Instructions

Asst. Lect.:Noor H. Hassoon

Logic Instructions

(1)

ORA	R M	1. Logical OR register or memory with accumulator 2. The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. 3. If the operand is a memory location, its address is specified by the contents of HL registers. 4. S, Z, P flags are modified to reflect the result of the operation. 5. CY and AC are reset. 6. Example: ORA 45_H

Truth Table of OR instruction

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	0

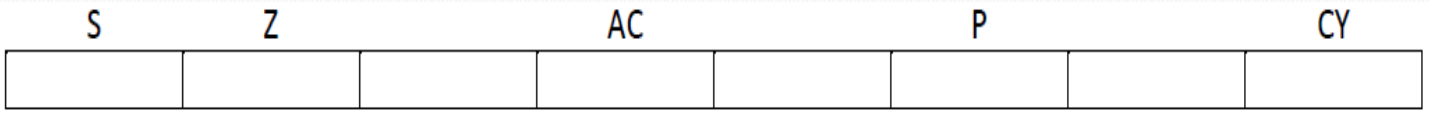
Logic Instructions

***Example*:**

- Write program in 8085 to store $9D_H$ in memory location 5060_H use indirect method. Then ORed them with the data from input port $F1_H$ that represented by switch number 1 and 2 are on and other switches are off. Show what are the flags?.

Answer

- LXI H, 5060_H
- MVI M, $9D_H$
- IN $F1_H$
- ORA M



Logic Instructions

(2)

ORI	8-bits data	1. Logical OR immediate with accumulator 2. The contents of the accumulator are logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. 3. S, Z, P flags are modified to reflect the result of the operation. 4. CY and AC are reset. 5. Example: ORI 55_H

Logic Instructions

Example:

- Write program in 8085 to load $9D_H$ from memory location 5060_H use indirect method. Then ORed immediate the data with register C that hold F0. Explain which switches on port 00_H are turn off?

Answer

- LXI H, 5060_H
- MOV A, M
- ORI C
- OUT 00

Logic Instructions

(3)

ANA	R M	1. Logical AND register or memory with accumulator 2. The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. 3. If the operand is a memory location, its address is specified by the contents of HL registers. 4. S, Z, P flags are modified to reflect the result of the operation. 5. CY is reset. AC is set. 6. Example: ANA 63_H

Truth Table of AND instruction

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Logic Instructions

Example:

- Write program in 8085 to store the Values (55, 65, 75, 85, 95) in memory locations (5000, 5001, 5002, 5003, 5004)$_H$ respectively use indirect method. Then ANDed these data and store the result in memory location 5005$_H$?

Answer

```
LXI  H, 5000H
MVI  M, 55 H
INX  H
MVI  M, 65 H
INX  H
MVI  M, 75H
INX  H
MVI  M, 85 H
INX  H
MVI  M, 95 H

MOV A,M
DCX H
ANA M

DCX H
ANA M

DCX H
ANA M

STA 5005H
```

Logic Instructions

(4)

ANI	8-bits data	1. Logical AND immediate with accumulator 2. The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. 3. S, Z, P flags are modified to reflect the result of the operation. 4. CY is reset. AC is set. 5. Example: ANI $8D_H$

Example:

- *Write program in 8085 ANDed immediate the value of register C with value of register H, if register C has 78 while register H holds 22. What are the values of flags.*

Answer

- *MOV A, C*
- *ANI H*

Logic Instructions

(5)

XRA	R M	1. Exclusive OR register or memory with accumulator 2. The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. 3. If the operand is a memory location, its address is specified by the contents of HL registers. 4. S, Z, P flags are modified to reflect the result of the operation. 5. CY and AC are reset. 6. Example: XRA 55_H

Truth Table of AND instruction

X	Y	X *XRA* Y
0	0	0
0	1	1
1	0	1
1	1	0

Logic Instructions

Example:

- Write program in 8085 to clear the Accumulator if it has $B3_H$. Add the content of memory location $30FF_H$ with register L. Show the result on port 50_H

Answer

- XRA A
- LDA $30FF_H$
- ADD L
- OUT 50_H

Logic Instructions

(6)

XRI	8-bits data	1. Exclusive OR immediate with accumulator 2. The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. 3. S, Z, P flags are modified to reflect the result of the operation. 4. CY and AC are reset. 5. Example: XRI 63_H

Logic Instructions

Example:

- Write program in 8085 to clear the Accumulator if it has $B3_H$. XRed Immediate the content of memory location $30FF_H$ that has 33_H with 77_H. Show which switch on port 50H is turn on.

Answer

- XRA A
- LDA $30FF_H$
- XRI 77_H
- OUT 55_H

Logic Instructions

(7) RRC

RRC	1. Rotate accumulator right 2. Each binary bit of the accumulator is rotated right by one position. 3. Bit D0 is placed in the position of D7 as well as in the Carry flag. 4. CY is modified according to bit D0. S, Z, P, AC are not affected. 5. Example: RRC

Logic Instructions

Example

- Store the data 55_H in memory location 3001 and the register B with 37_H. Add these data then rotate the result right.

Answer

- MVI A, 55_H
- STA 3001
- MVI B, 37
- LDA 3001
- ADD B

Logic Instructions

		1	1	1		1	1	1	
A	=	0	1	0	1	0	1	0	1
B	=	0	0	1	1	0	1	1	1
Result A =		1	0	0	0	1	1	0	0

Flags

S	Z			AC		p		CY
1	0			0		1		0

RRC

Result after RRC

0	1	0	0	0	1	1	0

Flags after RRC

S	Z			AC		p		CY
1	0			0		1		0

Logic Instructions

(8) RLC

RLC	1. Rotate accumulator left 2. Each binary bit of the accumulator is rotated left by one position. 3. Bit D7 is placed in the position of D0 as well as in the Carry flag. 4. CY is modified according to bit D0. S, Z, P, AC are not affected. 5. Example: RLC

Logic Instructions

Example

- Store the data FF_H in memory location 3001 and the register B with 87_H. Add these data then rotate the result Left.

Answer

- LXI H, 3001
- MVI M, FF_H
- MVI B, 87_H
- MOV B,A
- ADD M

Logic Instructions

		1	1	1	1	1	1	1	1
A	=	1	0	0	0	0	1	1	1
M	=	1	1	1	1	1	1	1	1
Result A =		1	0	0	0	0	1	1	0

Flags

S	Z			AC		p		CY
1	0			1		1		1

RLC

Result after RLC

0	0	0	0	1	1	0	1

Flags after RLC

S	Z			AC		p		CY
1	0			1		1		1

Logic Instructions

(9) RAR

RAR	1. Rotate accumulator right through carry 2. Each binary bit of the accumulator is rotated right by one position through the Carry flag. 3. Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7. 4. CY is modified according to bit D0. S, Z, P, AC are not affected. 5. Example: RAR

Logic Instructions

Example

- Store the data FF_H in memory location 3001 and the register B with 87_H. Add these data then rotate the result Right with carry.

Answer

- LXI H, 3001
- MVI M, FF_H
- MVI B, 87_H
- MOV B,A
- ADD M

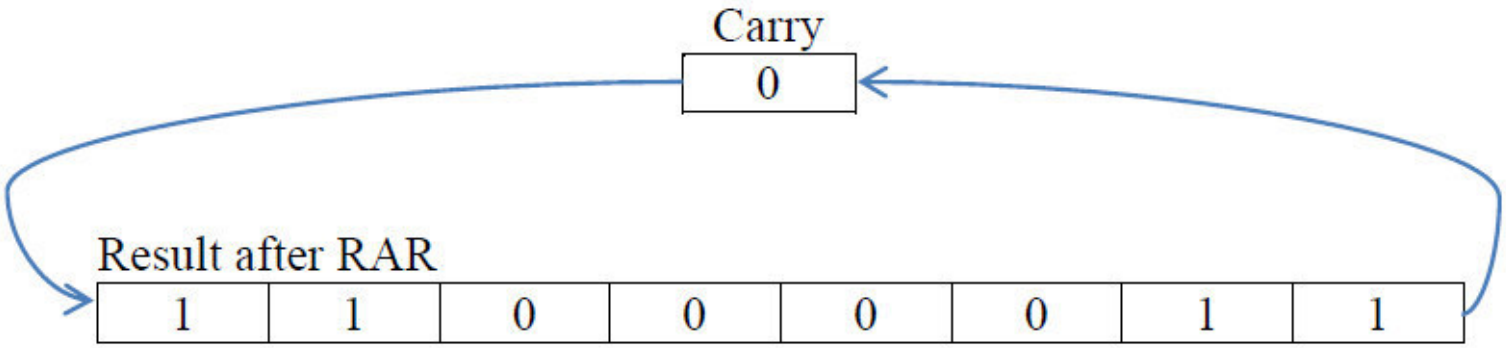
Logic Instructions

		1	1	1	1	1	1	1	1
A	=	1	0	0	0	0	1	1	1
M	=	1	1	1	1	1	1	1	1
Result A =		1	0	0	0	0	1	1	0

Flags

S	Z			AC		p		CY
1	0			1		1		1

RAR



Flags after RAR

S	Z			AC		p		CY
1	0			1		1		0

Logic Instructions

(10) RLR

RLR	1. Rotate accumulator right through carry 2. Each binary bit of the accumulator is rotated left by one position through the Carry flag. 3. Bit D7 is placed in the Carry flag, and the Carry flag is placed in the most significant position D0. 4. CY is modified according to bit D7. S, Z, P, AC are not affected. 5. Example: RLR

Logic Instructions

Example

- Store the data FF_H in memory location 3001 and the register B with 87_H. Add these data then rotate the result left with carry.

Answer

- LXI H, 3001
- MVI M, FF_H
- MVI B, 87_H
- MOV B,A
- ADD M

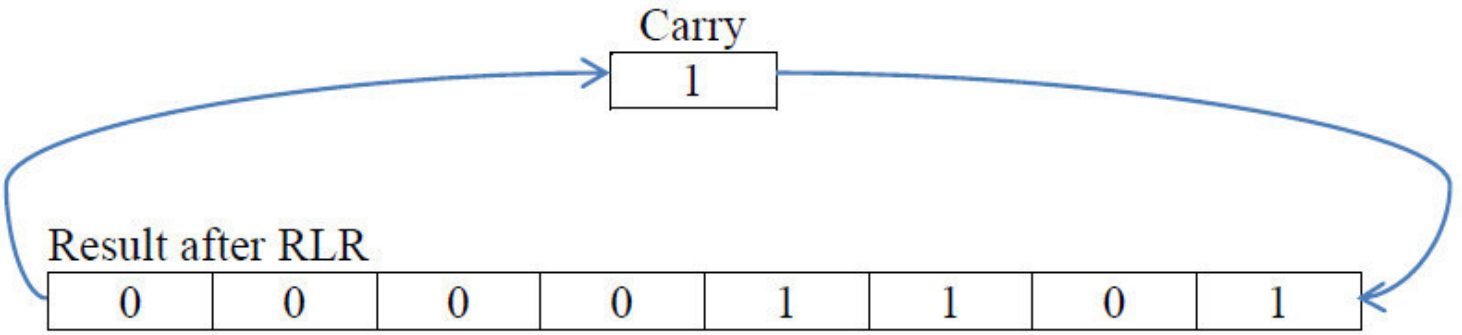
Logic Instructions

		1	1	1	1	1	1	1	1
A	=	1	0	0	0	0	1	1	1
M	=	1	1	1	1	1	1	1	1
Result A =		1	0	0	0	0	1	1	0

Flags

S	Z			AC		p		CY
1	0			1		1		1

RLR



Flags after RLR

S	Z			AC		p		CY
1	0			1		1		1

Logic Instructions

(11) CMA

CMA	1. Complement accumulator 2. The contents of the accumulator are complemented. 3. No flags are affected. 4. Example: CMA

Example

- Store the data FFH in memory location 3001 and the register B with 87H. Add these data then complement the result.

Answer

- LXI H, 3001
- MVI M, FFH
- MVI B, 87

MOV B,A
ADD M

Logic Instructions

		1	1	1	1	1	1	1	1
A	=	1	0	0	0	0	1	1	1
M	=	1	1	1	1	1	1	1	1
Result A =		1	0	0	0	0	1	1	0

Flags

S	Z			AC		p		CY
1	0			1		1		1

CMA

Result after CMA

0	1	1	1	1	0	0	1

Flags after CMA

S	Z			AC		p		CY
1	0			1		1		1


The End