DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE PLAN

Faculty Details

Name of the Faculty	:	V.BALU, R.PREETHI
Designation	:	Assistant Professor
Department	:	CSE

Course Details

Name of the Course	:	BE
Branch	:	CSE
Year	:	I I
Subject Code	:	PCC-CS402
Title of the Subject	:	Computer Organization and Architecture
Batch	:	2019-2020

COURSE PLAN

Select a course (a theory subject): **Computer Organization and Architecture**

Select a unit: **Unit I**

Select a Topic: **Computer Evolution and performance**

1. **Objectives**
   - To understand different development stages of digital computer
   - To know about the performance of various evolutionary components involved in each developmental stage
   - To understand the need for futuristic advancements yet to be made.
2. **Outcomes**

   Students will be able to

   - Present an overview of the evolution of computer technology from early digital computers to the latest microprocessors.
   - Understand the key performance issues that relate to computer design.
   - Explain the reasons for the move to advanced technologies in upcoming generations
   - Distinguish among chips, integrated circuits and various processors.
   - Summarize some of the issues in computer performance assessment.
3. **Pre-requisites**

   Knowledge about fundamentals of computer system
4. **Terminology used other than normal known scientific / engg terms and their fundamental explanations / relations**

   Not Applicable

Plan for the lecture delivery

1. **How to plan for delivery – black board / ppt / animated ppt / (decide which is good for this topic)**

   Power Point Presentation

2. **How to explain the definition and terms with in it**
   - Different generations of computer evolution
   - Technology used in each stage of evolution

- Comparison of each stage with respect to performance, speed and throughput
- Performance analysis and techniques used for improvement.

3. **How to start the need for any derivation / procedure / experiment / case study**

   Not Applicable

4. **Physical meaning of math equations / calculations**

   Not Applicable

5. **Units and their physical meaning to make them understand practical reality and comparison between different units**

   We have discussed about various evolutionary stages of a computer. Understood the pros and cons of various technologies used in each stage of development. Analyzing the performance measures helps us to select better technology in designing the future architecture.

6. **What is the final conclusion?**

   How various technological improvements resulted in a digital computer system today

7. **How to put it in a nut shell**

   Digital computers today is not a end. It will be continuing its evolution to more generations.

8. **Important points for understanding / memorizing / make it long lasting**

   5 generations of computer evolution:

   1st generation – vacuum tubes and valves

   2nd generation – transistors

   3rd generation- integrated circuits

   4th generation – VLSI Microprocessor

   5th generation - ULSI microprocessor

   Future – Artificial Intelligence

9. **Questions and cross questions in that topic to make them think beyond the topic.**

   1. How many major companies around the world manufacture microprocessors?
   2. What is the main electronic component used in first generation computers?

3. In most IBM PCs, the CPU, the device drives, memory expansion slots and active components are mounted on a single board. What is the name of this board?
4. Which is the first computer in India?

10. Final conclusions

Knowing the evolution history plays an important role in understanding the current technologies better and aids to implement futuristic innovations

History of Computers

Computer Generations

Generation	Approximate Dates	Technology	Typical Speed (operations per second)
1	1946–1957	Vacuum tube	40,000
2	1958–1964	Transistor	200,000
3	1965–1971	Small and medium scale integration	1,000,000
4	1972–1977	Large scale integration	10,000,000
5	1978–1991	Very large scale integration	100,000,000
6	1991-	Ultra large scale integration	1,000,000,000

It has become widely accepted to classify computers into generations based on the fundamental hardware technology employed. Each new generation is characterized by greater processing performance, larger memory capacity, and smaller size than the previous one.

A Brief History of Computers

The first Generation: Vacuum Tubes

1. **ENIAC**

The ENIAC (Electronic Numerical Integrator And Computer), designed by and constructed under the supervision of Jonh Mauchly and John Presper Eckert at the University of Pennsylvania, was the world's first general-purpose electronic digital computer. The project was a response to U.S. wartime needs. Mauchly, a professor of electrical engineering at the University of Pennsylvania and Eckert, one of his graduate students, proposed to build a general-purpose computer using vacuum tubes. In 1943, this proposal was accepted by the Army, and work began on the ENIAC. The resulting machine was enormous, weighting 30 tons, occupying 15,000 square feet of floor space, and containing more than 18,000 vacuum tubes. When operating, it consumed 140 kilowatts of power. It was aloes substantially faster than any electronic-mechanical computer, being capable of 5000 additions per second.

The ENIAC was decimal rather than a binary machine. That is, numbers were represented in decimal form and arithmetic was performed in the decimal system. Its memory consisted of 20 “accumulators”, each capable of holding a 10-digit decimal number. Each digit was represented by a ring of 10 vacuum tubes. At any time, only one vacuum tube was in the ON state, representing one of the 10 digits. The major drawback of the ENIAC was that it had to be programmed manually by setting switches and plugging and unplugging cables.

The ENIAC was completed in 1946, too late to be used in the war effort. Instead, its first task was to perform a series of complex calculations that were used to help determine the feasibility of the H-bomb. The ENIAC continued to be used until 1955.

2. The von Neumann Machine

The programming process could be facilitated if the program could be represented in a form suitable for storing in memory alongside the data. Then, a computer could get its instructions by reading them from memory, and a program could be set of altered by setting the values of a portion of memory.

This idea, known as the Stored-program concept, is usually attributed to the ENIAC designers, most notably the mathematician John von Neumann, who was a consultant on the ENIAC project. The idea was also developed at about the same time by Turing. The first publication of the idea was in a 1945 proposal by von Neumann for a new computer, the EDVAC (Electronic Discrete Variable Computer).

In 1946, von Neumann and his colleagues began the design of a new stored-program computer, referred to as the IAS computer, at the Princeton Institute for Advanced Studies. The IAS computer, although not completed until 1952, is the prototype of all subsequent general-purpose computers. Figure 1.5 shows the general structure of the IAS computer. It consists of:

- A main memory, which stores both data and instructions.
- An arithmetic-logical unit (ALU) capable of operating on binary data.
- A control unit, which interprets the instructions in memory and causes them to be executed.
- Input and output (I/O) equipment operated by the control unit.

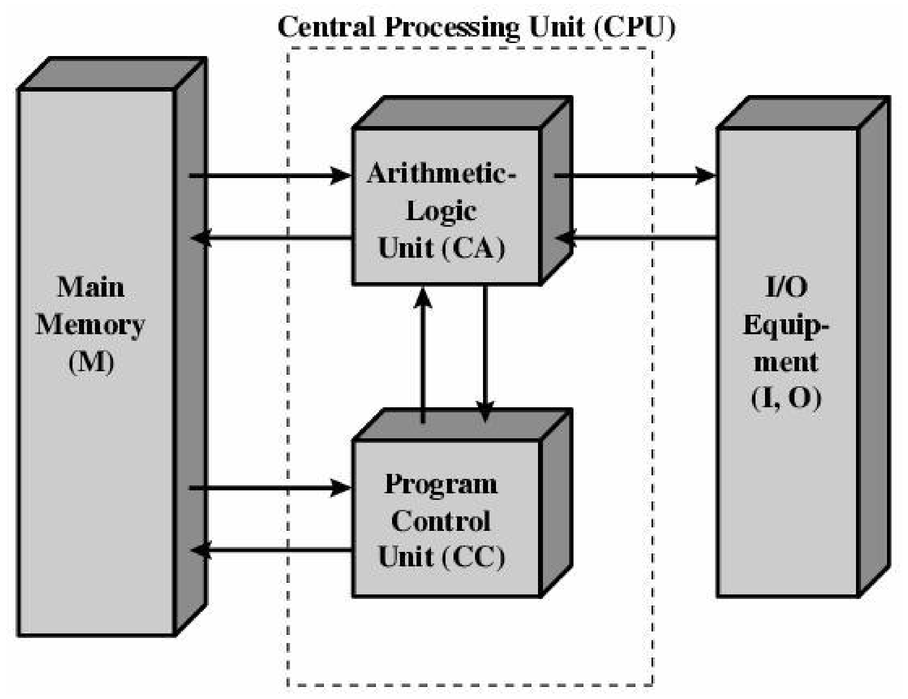

Central Processing Unit (CPU)
Main Memory (M)
Arithmetic-Logic Unit (CA)
Program Control Unit (CC)
I/O Equip-ment (I, O)

Figure 1.5 Structure of the IAS computer

3. **Commercial Computers**

The 1950s saw the birth of the computer industry with two companies, Sperry and IBM, dominating the marketplace.

In 1947, Eckert and Mauchly formed the Eckert-Maunchly computer Corporation to manufacture computers commercially. Their first successful machine was the UNIVAC I (Universal Automatic Computer), which was commissioned by the Bureau of the Census for the 1950 calculations. The Eckert-Maunchly Computer Corporation became part of the UNIVAC division of Sperry-Rand Corporation, which went on to build a series of successor machines.

The UNIVAC II, which had greater memory capacity and higher performance than the UNIVAC I, was delivered in the late 1950s and illustrates several trends that have remained characteristic of the computer industry. First, advances in technology allow companies to continue to build larger, more powerful computers. Second, each company tries to make its new machines upward compatible with the older machines. This means that the programs written for the older machines can be executed on the new machine. This strategy is adopted in the hopes of retaining the customer base; that is, when a customer decides to buy a newer machine, he is likely to get it from the same company to avoid losing the investment in programs.

The UNIVAC division also began development of the 1100 series of computers, which was to be its bread and butler. This series illustrates a distinction that existed at one time. The first model, the UNIVAC 1103, and its successors for many years were primarily intended for scientific applications, involving long and complex calculations. Other companies concentrated on business applications, which involved processing large amounts of text data. This split has largely disappeared but it was evident for a number of years.

IBM, which was then the major manufacturer of punched-card processing equipment, delivered its first electronic stored-program computer, the 701, in 1953. The 70l was intended primarily for scientific applications. In 1955, IBM introduced the companion 702 product, which had a number of hardware features that suited it to business applications. These were the first of a long series of 700/7000 computers that established IBM as the overwhelmingly dominant computer manufacturer.

The Second Generation: Transistors

The first major change in the electronic computer came with the replacement of the vacuum tube by the transistor. The transistor is smaller, cheaper, and dissipates less heal than a vacuum tube but can be used in the same way as a vacuum tube to construct computers. Unlike the vacuum tube, which requires wires, metal plates, a glass capsule, and a vacuum, the transistor is a solid-state device, made from silicon.

The transistor was invented at Bell Labs in 1947 and by the 1950s had launched an electronic revolution. It was not until the late 1950s, however, that fully transistorized computers were commercially available. IBM again was not the first company to deliver the new technology. NCR and. more successfully. RCA were the front-runners with some small transistor machines. IBM followed shortly with the 7000 series.

The use of the transistor defines the second generation of computers. It has become widely accepted to classify computers into generations based on the fundamental hardware technology employed. Each new generation is characterized by greater processing performance, larger memory capacity, and smaller size than the previous one.

The Third Generation: Integrated Circuits

A single, self-contained transistor is called a discrete component. Throughout the 1950s and early 1960s, electronic equipment was composed largely of discrete components—transistors, resistors, capacitors, and so on. Discrete components were manufactured separately, packaged in their own containers, and soldered or wired together onto circuit boards, which were then installed in computers, oscilloscopes, and other electronic equipment. Whenever an electronic device called for a transistor, a little lube of metal containing a pinhead-sized piece of silicon had to be soldered to a circuit hoard. The entire manufacturing process, from transistor to circuit board, was expensive and cumbersome.

These facts of life were beginning to create problems in the computer industry. Early second-generation computers contained about 10,000 transistors. This figure grew to the hundreds of thousands, making the manufacture of newer, more powerful machines increasingly difficult.

In 1958 came the achievement that revolutionized electronics and started the era of microelectronics: the invention of the integrated circuit. It is the integrated circuit that defines the third generation of computers. Perhaps the two most important members of the third generation are the IBM System/360 and the DEC PDP-8.

Chip Growth

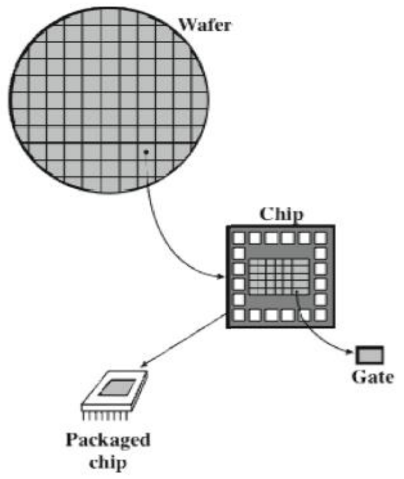


Figure 2.7 Relationship Among Wafer, Chip, and Gate

A thin **wafer** of silicon is divided into a matrix of small areas, each a few millimeters square. The identical circuit pattern is fabricated in each area, and the wafer is broken up into **chips.** Each chip consists of many gates and/or memory cells plus a number of input and output attachment points. This chip is then packaged in housing that protects it and provides pins for attachment to devices beyond the chip. A number of these packages can then be interconnected on a printed circuit board to produce larger and more complex circuits.

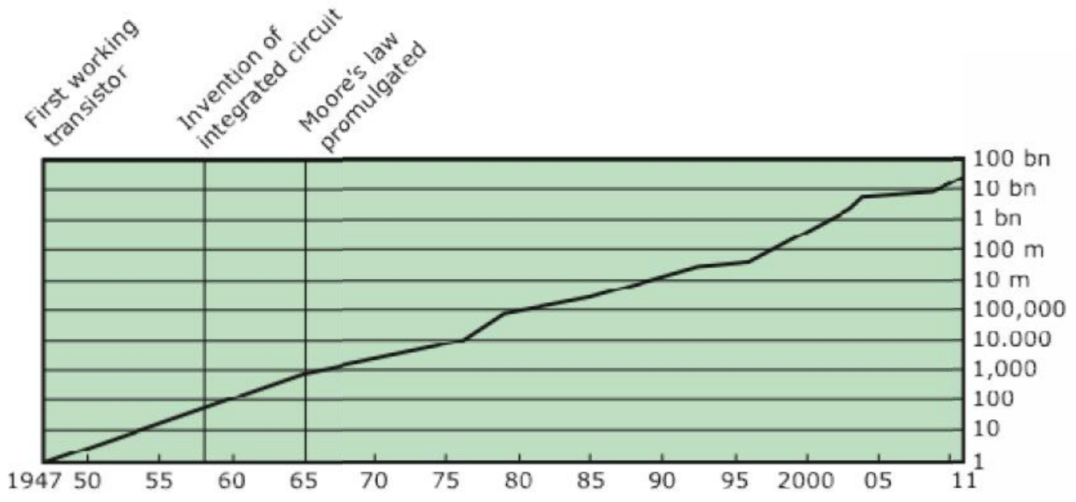


Figure 2.8 Growth in Transistor Count on Integrated Circuits (DRAM memory)

Moore's Law

1965; Gordon Moore – co-founder of Intel

Observed number of transistors that could be put on a single chip was doubling every year

The pace slowed to a doubling every 18 months in the 1970's but has sustained that rate ever since

Consequences of Moore's law:

- The cost of computer logic and memory circuitry has fallen at a dramatic rate
- The electrical path length is shortened, increasing operating speed
- Computer becomes smaller and is more convenient to use in a variety of environments
- Reduction in power and cooling requirements
- Fewer interchip connections

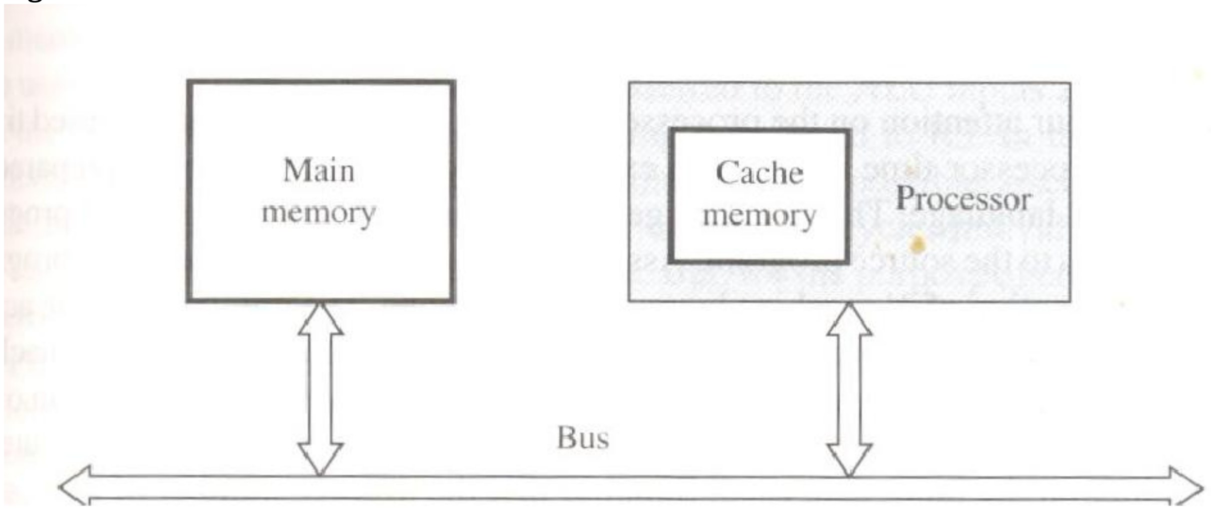
Later Generations

Beyond the third generation there is less general agreement on defining generations of computers. There have been a fourth and a fifth generation, based on advances in integrated circuit technology. With the introduction of large-scale integration (LSI), more than 1000,000 components can be placed on a single integrated circuit chip. Very-large-scale integration (VLSI) achieved more than 1000,000,000 components per chip, and current VLSI chips can contain more than 1000.000 components.

PERFORMANCE:

For best performance, it is necessary to design the compiler, machine instruction set and hardware in a co-ordinate way.
Elapsed Time - the total time required to execute the program is called the elapsed time. It depends on all the units in computer system.
Processor Time - The period in which the processor is active is called the processor time. It depends on hardware involved in the execution of the instruction.

Fig: The Processor Cache



A Program will be executed faster if the movement of instruction and data between the main memory and the processor is minimized, which is achieved by using the Cache.

Processor clock:
Clock - The Processor circuits are controlled by a timing signal called a clock.
Clock Cycle - The cycle defines a regular time interval called clock cycle.
Clock Rate,R =1/P
Where, P - Length of one clock cycle.

Basic Performance Equation:

$T = (N*S)/R$

Where, T - Performance Parameter

R - Clock Rate in cycles/sec

N - Actual number of instruction execution

S - Average number of basic steps needed to execute one machine instruction.

To achieve high performance,

$N,S<R$

Pipelining and Superscalar operation:

Pipelining : A Substantial improvement in performance can be achieved by overlapping the execution of successive instruction using a technique called pipelining.

Superscalar Execution : It is possible to start the execution of several instruction in everey clock cycles (ie)several instruction can be executed in parallel by creating parallel paths.This mode of operation is called the Superscalar execution.

Clock Rate:

There are 2 possibilities to increase the clock rate(R).They are,

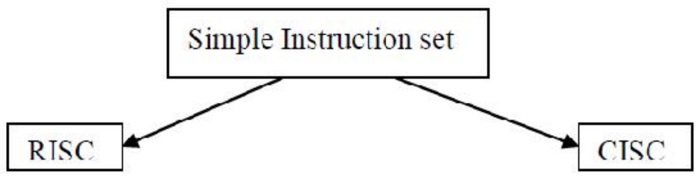
Improving the integrated Chip(IC) technology makes logical circuits faster.

Reduce the amount of processing done in one basic step also helps to reduce the clock period P.

Instruction Set:CISC AND RISC:

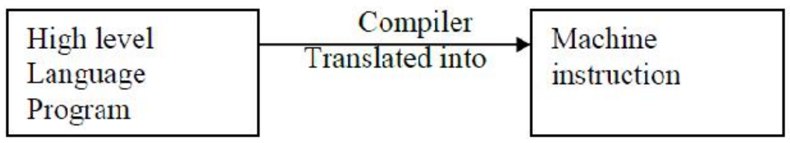
The Complex instruction combined with pipelining would achieve the best performance.

It is much easier to implement the efficient pipelining in processor with simple instruction set.



(Reduced Instruction Set Computer)
It is the design of the instruction set
of a processor with simple instruction

(Complex Instruction Set Computer)
It is the design of the instruction set
of a processor with complex instruction.



Functions of Compiler:

The compiler re-arranges the program instruction to achieve better performance.

The high quality compiler must be closely linked to the processor architecture to reduce the total number of clock cycles.

Performance Measurement:

The Performance Measure is the time it takes a computer to execute a given benchmark.

A non-profit organization called SPEC(System Performance Evaluation Corporation) selects and publishes representative application program.

$$\boxed{\text{SPEC rating} = \frac{\text{Running time on reference computer}}{\text{Running time on computer under test}}}$$

The Overall SPEC rating for the computer is given by,

$$\boxed{\text{SPEC rating} = \left(\prod_{i=1}^{n} \text{SPEC}_i \right)^{1/n}}$$

COURSE PLAN

Select a course (a theory subject): **Computer Organization and Architecture**

Select a unit: **Unit I**

Select a Topic: **Functional components and operations of general computer system**

1. **Objectives**
   - Students will know various components of a computer system and how these components are organized .
   - Students will be able to understand the functions of different components.
   - Students will have a thorough understanding of the basic structure and operation of a digital computer.
2. **Outcomes**
   - Students will be able to describe the fundamental organisation of a computer system
   - Students can have a clear cut understanding about the functional units of a processor
   - Ability to select appropriate computer systems for given application domains for future design of computer architecture.
   - Understand and develop processor for future computing hardwires to solves the problems of high end computing applications
3. **Pre-requisites**

   Basic knowledge about computer and exposure to system usage
4. **Terminology used other than normal known scientific / engg terms and their fundamental explanations / relations**

   Not Applicable

Plan for the lecture delivery

1. **How to plan for delivery – black board / ppt / animated ppt / (decide which is good for this topic)**

   Power Point Presentation/animated ppt

2. **How to explain the definition and terms with in it**

   What is Computer?

   Function of a computer system

   Various componens of computer

   Functions of each sub component

3. **How to start the need for any derivation / procedure / experiment / case study**

   Case Study

   Not Applicable

4. **Physical meaning of math equations / calculations**

   Not Applicable

5. **Units and their physical meaning to make them understand practical reality and comparison between different units**

   We have discussed about the basic structure of a computer system and how the components operate. It is sufficient to differentiate the functions of various components such as input,output, processing and storage counterparts.

6. **What is the final conclusion?**

   Understands the structural and functional model of a digital computer system.

7. **How to put it in a nut shell**

   Physical and behavioural model of a digital compuer

8. **Important points for understanding / memorizing / make it long lasting**

   Understand various functional components of a computer system

   Input/output unit

   Memory unit

   Control unit

   Arithmetic & Logic unit

9. **Questions and cross questions in that topic to make them think beyond the topic.**

   1. What do you understand by the term Computer Architecture?
   2. Is Computer Architecture different from a Computer Organization?
   3. What is the purpose of control unit ?
   4. State the role of input & output units.

5. What are the components of a processor?

10. Final conclusions

This course plan typically describes the general structural architecture to be followed in order to build a computer system. Various functional units of a processor and their primary roles were discussed. How these functional components are interconnected and how interoperability leads to achieve system performance as a whole can be learned in future sessions.

What is computer?

A **computer** is an electronic device that manipulates information, or data. It has the ability to store, retrieve, and process data. You may already know that you can use a **computer** to type documents, send email, play games, and browse the Web.

What is computer organization?

Computer organization is a study of computer architecture that includes CPU, memory, registers and so on. Data processing takes place in a CPU of a computer.

What is computer Architecture?

Computer architecture consists of rules and methods or procedures which describe the implementation, functionality of the computer systems.

Computer organization VS computer architecture

S.NO	COMPUTER ARCHITECTURE	COMPUTER ORGANIZATION
1.	Architecture describes what the computer does.	Organization describes how it does it.
2.	Computer Architecture deals with functional behavior of computer system.	Computer Organization deals with structural relationship.

3.	it deals with high-level design issue.	it deals with low-level design issue.
4.	Architecture indicates its hardware.	Where, Organization indicates its performance.
5.	For designing a computer, its architecture is fixed first.	For designing a computer, organization is decided after its architecture.
6.	Computer Architecture is also called as instruction set architecture.	Computer Organization is frequently called as micro architecture.
7.	Computer Architecture comprises logical functions such as instruction sets, registers, data types and addressing modes.	Computer Organization consists of physical units like circuit designs, peripherals and adders.
8.	Architecture coordinates between the hardware and software of the system.	Computer Organization handles the segments of the network in a system.

Functional Units of a computer system

A computer consists of five functionally independent main parts input, memory, arithmetic logic unit (ALU), output and control unit.

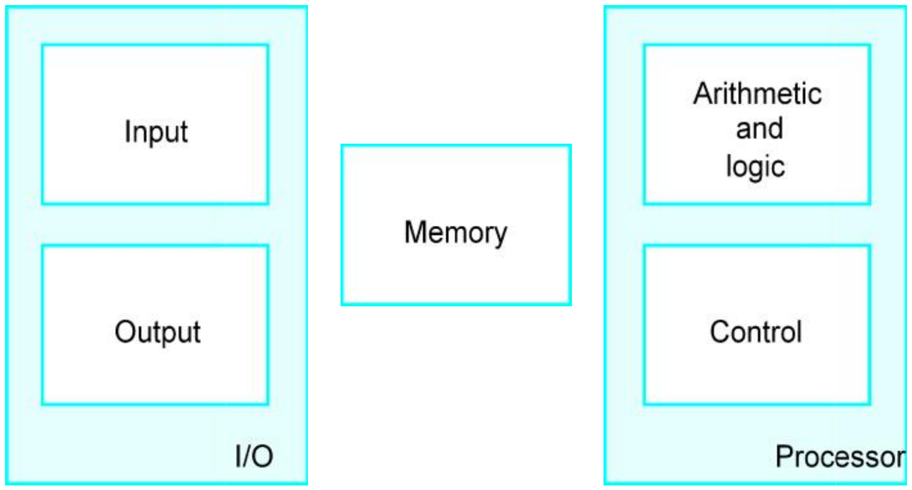


Figure 1.1. Basic functional units of a computer.

Input device accepts the coded information as source program i.e. high level language. This is either stored in the memory or immediately used by the processor to perform the desired operations. The program stored in the memory determines the processing steps. Basically the computer converts one source program to an object program. i.e. into machine language. Finally the results are sent to the outside world through output device. All of these actions are coordinated by the control unit.

1. **Input unit:** - The source program/high level language program/coded information/simply data is fed to a computer through input devices keyboard is a most common type. Whenever a key is pressed, one corresponding word or number is translated into its equivalent binary code over a cable & fed either to memory or processor. Joysticks, trackballs, mouse, scanners etc are other input devices.
2. **Memory unit:** - Its function into store programs and data. It is basically to two types
    a. Primary memory
    b. Secondary memory

Primary memory: -

- Is the one exclusively associated with the processor and operates at the electronics speeds programs must be stored in this memory while they are being executed.
- The memory contains a large number of semiconductors storage cells. Each capable of storing one bit of information. These are processed in a group of fixed site called word.
- To provide easy access to a word in memory, a distinct address is associated with each word location.
- Addresses are numbers that identify memory location. Number of bits in each word is called word length of the computer. Programs must reside in the memory during execution.
- Instructions and data can be written into the memory or read out under the control of processor. Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called random-access memory (RAM).
- The time required to access one word in called memory access time.
- Memory which is only readable by the user and contents of which can't be altered is called read only memory (ROM) it contains operating system.

- ➢ Caches are the small fast RAM units, which are coupled with the processor and are aften contained on the same IC chip to achieve high performance.
- ➢ Although primary storage is essential it tends to be expensive.

Secondary memory: -

Is used where large amounts of data & programs have to be stored, particularly information that is accessed infrequently.

Examples: - Magnetic disks & tapes, optical disks (ie CD-ROM's), floppies etc.,

3. **Arithmetic logic unit (ALU):-**
    - ➢ Most of the computer operators are executed in ALU of the processor like addition, subtraction, division, multiplication, etc.
    - ➢ the operands are brought into the ALU from memory and stored in high speed storage elements called register.
    - ➢ Then according to the instructions the operation is performed in the required sequence.
    - ➢ The control and the ALU are many times faster than other devices connected to a computer system.
    - ➢ This enables a single processor to control a number of external devices such as key boards, displays, magnetic and optical disks, sensors and other mechanical controllers.
4. **Output unit:-** These actually are the counterparts of input unit. Its basic function is to send the processed results to the outside world.
   Examples:- Printer, speakers, monitor etc.
5. **Control unit:-** It effectively is the nerve center that sends signals to other units and senses their states. The actual timing signals that govern the transfer of data between input unit, processor, memory and output unit are generated by the control unit.

<u>Basic operational concepts:</u> -

- Activity in a computer is governed by instructions.

- To perform a task, an appropriate program consisting of a list of instructions is stored in the memory.
- Individual instructions are brought from the memory into the processor, which executes the specified operations.
- Data to be used as operands are also stored in the memory.

Examples: **- Add LOCA, R0**

This instruction adds the operand at memory location LOCA, to operand in register R0 & places the sum into register. This instruction requires the performance of several steps,

1. First the instruction is fetched from the memory into the processor.
2. The operand at LOCA is fetched and added to the contents of R0
3. Finally the resulting sum is stored in the register R0

The preceding add instruction combines a memory access operation with an ALU Operations. In some other type of computers, these two types of operations are performed by separate instructions for performance reasons.

Load LOCA, R1

Add R1, R0

Separate Memory Access and ALU Operation- Transfers between the memory and the processor are started by sending the address of the memory location to be accessed to the memory unit and issuing the appropriate control signals. The data are then transferred to or from the memory.

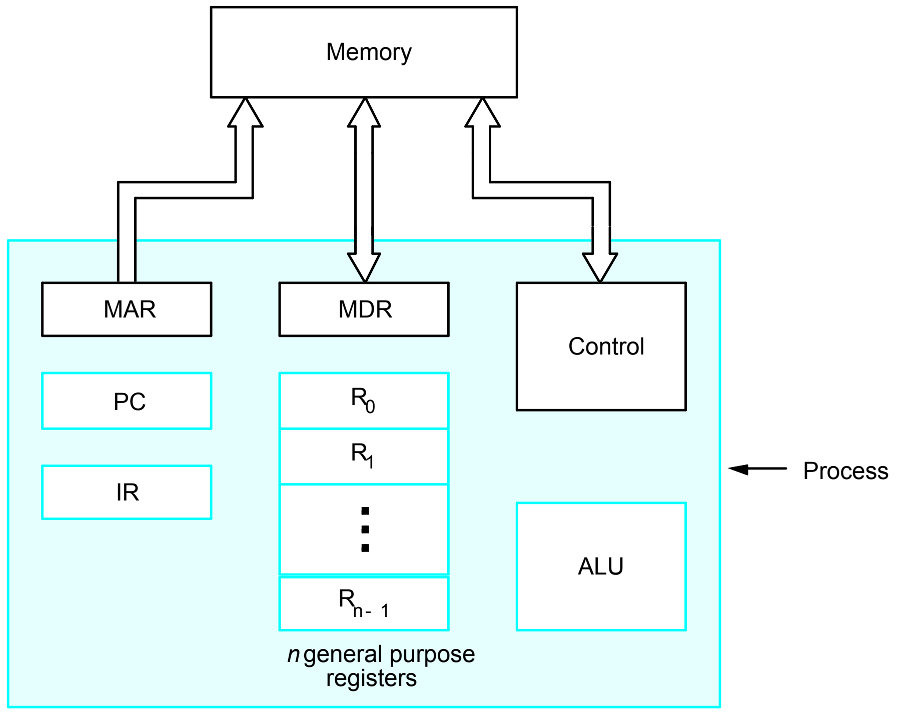


Figure 1.2. Connections between the processor and the memory.

The fig shows how memory & the processor can be connected. In addition to the ALU & the control circuitry, the processor contains a number of registers used for several different purposes.

The instruction register (IR):- Holds the instructions that is currently being executed. Its output is available for the control circuits which generates the timing signals that control the various processing elements in one execution of instruction.

The program counter PC:- This is another specialized register that keeps track of execution of a program. It contains the memory address of the next instruction to be fetched and executed.

Besides IR and PC, there are n-general purpose registers R0 through Rn1. The other two registers which facilitate communication with memory are: -

1. MAR – (Memory Address Register):- It holds the address of the location to be accessed.

2. MDR – (Memory Data Register):- It contains the data to be written into or read out of the address location.

Typical Operating steps are

1. Programs reside in the memory & usually get these through the I/P unit.
2. Execution of the program starts when the PC is set to point at the first instruction of the program.
3. Contents of PC are transferred to MAR and a Read Control Signal is sent to the memory.
4. After the time required to access the memory elapses, the address word is read out of the memory and loaded into the MDR.
5. Now contents of MDR are transferred to the IR & now the instruction is ready to be decoded and executed.
6. If the instruction involves an operation by the ALU, it is necessary to obtain the required operands.
7. An operand in the memory is fetched by sending its address to MAR & Initiating a read cycle.
8. When the operand has been read from the memory to the MDR, it is transferred from MDR to the ALU.
9. After one or two such repeated cycles, the ALU can perform the desired operation.
10. If the result of this operation is to be stored in the memory, the result is sent to MDR.
11. Address of location where the result is stored is sent to MAR & a write cycle is initiated.
12. The contents of PC are incremented so that PC points to the next instruction that is to be executed.

- Normal execution of a program may be preempted (temporarily interrupted) if some devices require urgent servicing, to do this one device raises an Interrupt signal.

- An interrupt is a request signal from an I/O device for service by the processor. The processor provides the requested service by executing an appropriate interrupt service routine.
- The Diversion may change the internal stage of the processor its state must be saved in the memory location before interruption. When the interrupt-routine service is completed the state of the processor is restored so that the interrupted program may continue.

COURSE PLAN

Select a course (a theory subject): **Computer Organization and Architecture**

Select a unit: **Unit I**

Select a Topic: **Machine Instruction and Instruction Representation**

1. **Objectives**

   To understand how different Instruction format in the Computer System.

   To know about how to write an instruction.

2. **Outcomes**
   a. Students will be able to understand the Instruction format and ability to write machine instruction
   b. Understand various type of instruction

3. **Pre-requisites**

   Prior knowledge on Functional components and operations of general computer system

4. **Terminology used other than normal known scientific / engg terms and their fundamental explanations / relations**

   Not Applicable

Plan for the lecture delivery

1. **How to plan for delivery – black board / ppt / animated ppt / (decide which is good for this topic)**

   Power Point Presentation/animated PPT

2. **How to explain the definition and terms with in it**

   What is machine instruction, instruction format, Classification for Instructions and Types of Instruction?

3. **How to start the need for any derivation / procedure / experiment / case study**

   What is machine instruction and explain the instruction format and various types of Instruction available in the Processor.

4. **Physical meaning of math equations / calculations**

   Not Applicable

5. **Units and their physical meaning to make them understand practical reality and comparison between different units**

   Machine instruction and applied in the addressing modes for specifying the operand is presented in what memory location.

6. **What is the final conclusion?**

   Understand the concept of what is Instruction, Instruction format and various types of Instruction.

7. **How to put it in a nut shell**

   Machine instruction format, Types of instruction, Classification of Instruction

8. **Important points for understanding / memorizing / make it long lasting**

   - Machine Instruction is Command Or Program
   - An instruction is normally made up of a combination of an operation code and some way of specifying an operand, most commonly by its location or address in memory
   - Data transfer instructions, Arithmetic instructions, Logic instructions, String manipulation instruction, Control transfer instructions, Loop control instructions
   - Three address instructions, Two address instructions, One address instructions, Zero address instruction

9. **Questions and cross questions in that topic to make them think beyond the topic.**

   What is Machine Instruction?

   Explain the various of types of Machine Instruction.

10. **Final conclusions**

    Understand the concept of what is Instruction, Instruction format and various types of Instruction.

What is Machine Instruction?

Machine Instructions are commands or programs written in machine code of a machine (computer) that it can recognize and execute. A machine instruction consists of several bytes in memory that tells the processor to perform one machine operation. The processor looks at machine instructions in main memory one after another, and performs one machine operation for each machine instruction. The collection of machine instructions in main memory is called a machine language program.

Machine code or machine language is a set of instructions executed directly by a computer's central processing unit (CPU). Each instruction performs a very specific task, such as a load, a jump, or an ALU operation on a unit of data in a CPU register or memory. Every program directly executed by a CPU is made up of a series of such instructions.

Machine Instruction Format

The general format of a **machine instruction** is

[Label:]   Mnemonic   [Operand, Operand]   [; Comments]

- Brackets indicate that a field is optional
- Label is an identifier that is assigned the address of the first byte of the instruction in which it appears. It must be followed by **":"**
- Inclusion of spaces is arbitrary, except that at least one space must be inserted; no space would lead to an ambiguity.
- Comment field begins with a semicolon **" ; "**

Example:

```
Here:        MOV    R5,#25H        ;load 25H into R5
```

Machine instructions used in 8086 microprocessor

1. Data transfer instructions– move, load exchange, input, output.

MOV	:Move byte or word to register or memory .
IN, OUT	: Input byte or word from port, output word to port.
LEA	: Load effective address
LDS, LES	: Load pointer using data segment, extra segment .
PUSH, POP	: Push word onto stack, pop word off stack.
XCHG	: Exchange byte or word.
XLAT	: Translate byte using look-up table.

2. Arithmetic instructions – add, subtract, increment, decrement, convert byte/word and compare.

ADD, SUB	: Add, subtract byte or word
ADC, SBB	:Add, subtract byte or word and carry (borrow).
INC, DEC	: Increment, decrement byte or word.
NEG	: Negate byte or word (two’s complement).
CMP	: Compare byte or word (subtract without storing).
MUL, DIV	: Multiply, divide byte or word (unsigned).
IMUL, IDIV	: Integer multiply, divide byte or word (signed)
CBW, CWD	: Convert byte to word, word to double word
AAA, AAS, AAM,AAD	: ASCII adjust for add, sub, mul, div .
DAA, DAS	: Decimal adjust for addition, subtraction (BCD numbers)

3. Logic instructions – AND, OR, exclusive OR, shift/rotate and test

NOT	: Logical NOT of byte or word (one’s complement)
AND	: Logical AND of byte or word
OR	: Logical OR of byte or word.
XOR	: Logical exclusive-OR of byte or word
TEST	: Test byte or word (AND without storing).
SHL, SHR	: Logical Shift rotate instruction shift left, right byte or word? by 1or CL
SAL, SAR	: Arithmetic shift left, right byte or word? by 1 or CL
ROL, ROR	: Rotate left, right byte or word? by 1 or CL .
RCL, RCR	: Rotate left, right through carry byte or word? by 1 or CL.

4. String manipulation instruction – load, store, move, compare and scan for byte/word

MOVS: Move byte or word string

MOVSB, MOVSW: Move byte, word string.

CMPS: Compare byte or word string.

SCAS S: can byte or word string (comparing to A or AX)

LODS, STOS: Load, store byte or word string to AL.

5. Control transfer instructions – conditional, unconditional, call subroutine and return from subroutine.

JMP:Unconditional jump .it includes loop transfer and subroutine and interrupt instructions.

JNZ:jump till the counter value decreases to zero.It runs the loop till the value stored in CX becomes zero

6. Loop control instructions-

LOOP: Loop unconditional, count in CX, short jump to target address.

LOOPE (LOOPZ): Loop if equal (zero), count in CX, short jump to target address.

LOOPNE (LOOPNZ): Loop if not equal (not zero), count in CX, short jump to target address.

JCXZ: Jump if CX equals zero (used to skip code in loop).

Subroutine and Interrupt instructions-

CALL, RET: Call, return from procedure (inside or outside current segment).

INT, INTO: Software interrupt, interrupt if overflow.IRET: Return from interrupt.

7. Processor control instructions-

Flag manipulation:

STC, CLC, CMC: Set, clear, complement carry flag.

STD, CLD: Set, clear direction flag.STI, CLI: Set, clear interrupt enable flag.

PUSHF, POPF: Push flags onto stack, pop flags off stack.

INSTRUCTION REPRESENTATION

Each instruction is represented by a Sequence of bits. The instruction is divided into fields Duration Instruction Execution an instruction is read into an instruction register (IR) The CPU must be able to extract the data from various instruction fields to perform the required Operation Opcodes are represented by abbreviations called mnemonics that indicates the operation

Example

Mnemonics	**Description**
ADD	Addition Operation
SUB	Subtraction Operation
MPY	Multiply Operation
DIV	Division Operation
LOAD	Load data from Memory
STOR	Store data from Memory

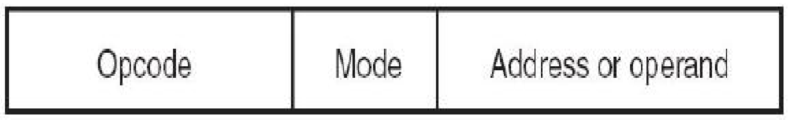
Simple Instruction

Example ADD R,Y

ADD R,Y means add the value contained in data location in memory Y to the contents of register R. In this example Y refers to the address of a location in memory and R refers to the particular register Note that the operation formed on the content of a location not on its address Instruction represented in the form of Sequence of bits

What is instruction format?

An instruction is normally made up of a combination of an operation code and some way of specifying an operand, most commonly by its location or address in memory



Computer perform a task based on instruction. Instruction is a group of fields the fields are Operation field which specified the operation to be performed like addition, subtraction etc Address field which contain the location of operand ie. Register or memory location Model field which specified how operand is to be founded.

Classification of Instructions

Three address instructions

Two address instructions

One address instructions

Zero address instruction

Three address instructions

Memory addresses for the two operands and one destination need to be specified. It is also called General register organization.

Instruction: ADD R1, R2, R3

Micro operation: R1 $\leftarrow$ R2 + R3

EVALUATE X=(A+B)*(C+D)

ADD R1, A, B	R1 $\leftarrow$ M[A]+M[B]
ADD R2, C, D	R2 $\leftarrow$ M[C]+M[D]
MUL X, R1, R2	M[X] $\leftarrow$ R1*R2

Two address instructions

Two address registers or two memory locations are specified Assumes that the destination address is the same as that of the first operand.

Instruction: ADD R1, R2

Micro operation: R1 ← R1 + R2

EVALUATE X=(A+B)*(C+D)

MOV R1, A	R1 ← M[A]
ADD R1, B	R1 ← R1+M[B]
MOV R2, C	R2 ← M[C]
ADD R2, D	R2 ← R2+M[D]
MUL R1,R2	R1 ← R1*R2
MOV X,R1	M[X] R1

Two address instructions

Two address registers or two memory locations are specified Assumes that the destination address is the same as that of the first operand.

Instruction: ADD R1, R2

Micro operation: R1 ← R1 + R2

EVALUATE X=(A+B)*(C+D)

MOV R1, A	R1 ← M[A]
ADD R1, B	R1 ← R1+M[B]
MOV R2, C	R2 ← M[C]
ADD R2, D	R2 ← R2+M[D]
MUL R1,R2	R1 ← R1*R2
MOV X,R1	M[X] ← R1

One address instructions

One address can be a register name or memory address.

SINGLE ACCUMULATOR ORGANIZATION

It uses AC register for all data manipulation

Instruction: ADD X

Micro operation: AC ← AC + M[X]

EVALUATE X=(A+B)*(C+D)

LOAD A	AC ← M[A]
ADD B	AC ← AC+M[B]
STORE T	M[T] ← AC
LOAD C	AC ← M[C]
ADD D	AC ← AC+M[D]
MUL T	AC ← AC*M[T]
STORE X	M[X] ← AC

Zero address instruction

Stack is used. Arithmetic operation pops two operands from the stack and pushes the result.

Also called stack organization

Zero address instruction

Evaluate X = (A +B) * (C +D)

PUSH A	TOS ← A
PUSH B	TOS ← B
ADD	TOS ← (A+B)
PUSH C	TOS ← C
PUSH D	TOS ← D
ADD	TOS ← (C+D)
MUL	TOS ← (C+D)*(A+B)
POP X	M[X] ← TOS

COURSE PLAN

Select a course (a theory subject): **Computer Organization and Architecture**

Select a unit: **Unit I**

Select a Topic: **Addressing Modes and Instruction Cycle**

1. **Objectives**

   To understand various addressing modes in Computer.
2. **Outcomes**
   a) Understand the various ways of Specifying the addressing Modes.
   b) Understand the Types of addressing modes
3. **Pre-requisites**

   Prior knowledge on machine instruction

4. **Terminology used other than normal known scientific / engg terms and their fundamental explanations / relations**

   Not Applicable

Plan for the lecture delivery

1. **How to plan for delivery – black board / ppt / animated ppt / (decide which is good for this topic)**
   Power Point Presentation/animated PPT / Video

2. **How to explain the definition and terms with in it**
   What is addressing mode, and explain various addressing modes.

3. **How to start the need for any derivation / procedure / experiment / case study**
   Defining the addressing modes, Explain various addressing modes, Steps involved in Instruction Life Cycle

4. **Physical meaning of math equations / calculations**

   Not Applicable

5. **Units and their physical meaning to make them understand practical reality and comparison between different units**

   Understand the various addressing mode and life cycle of the Instruction set are in this topic will help to write the machine instruction

6. **What is the final conclusion?**
   Understand what are the various addressing modes are specifying in the computer and how the instruction cycle is performed

7. **How to put it in a nut shell**

   Different addressing modes, Instruction Cycle

8. **Important points for understanding / memorizing / make it long lasting**

   Implied and Immediate Addressing Modes

   Direct or Indirect Addressing Modes

   Register Addressing Modes

   Register Indirect Addressing Mode

   Auto-Increment and Auto-Decrement Addressing Modes

   Displacement Based Addressing Modes

   Instruction Cycle (Fetch, Decode, Execute, Store)

9. **Questions and cross questions in that topic to make them think beyond the topic.**
    a) Explain the various addressing modes in a Computer System.
    b) Explain the Instruction Life Cycle.

10. **Final conclusions**

Conclusion is that, the method by which the addressing of source of data or the address of destination of result is given in the instruction is called addressing mode.

And whatever the addressing mode is there, the instruction consist of the operand, address of the source data, and destination of the result

Or, we can say that addressing modes for the instruction is also depends on that what bye of instruction we input

What is Addressing Modes ?

The operation field of an instruction specifies the operation to be performed. And this operation must be performed on some data. So each instruction need to specify data on which the operation is to be performed. But the operand(data) may be in accumulator, general purpose register or at some specified memory location. So, appropriate location (address) of data is need to be specified, and in computer, there are various ways of specifying the address of data. These various ways of specifying the address of data are known as "Addressing Modes"

So Addressing Modes can be defined as "The technique for specifying the address of the operands " And in computer the address of operand i.e., the address where operand is actually found is known as **"Effective Address".** Now, in addition to this, the two most prominent reason of why addressing modes are so important are:

First, the way the operand data are chosen during program execution is dependent on the addressing mode of the instruction.

Second, the address field(or fields) in a typical instruction format are relatively small and sometimes we would like to be able to reference a large range of locations, so here to achieve this objective i.e., to fit this large range of location in address field, a variety of addressing techniques has been employed. As they reduce the number of field in the addressing field of the instruction.

Thus, Addressing Modes are very vital in Instruction Set Architecture(ISA).

some notations are

A=	Contents of an address field in the instruction
R=	Contents of an address field in the instruction that refers to a register
EA=	Effective Address(Actual address) of location containing the referenced operand.
(X)=	Contents of memory location x or register X.

Types Of Addressing Modes

Various types of addressing modes are:

1. Implied and Immediate Addressing Modes
2. Direct or Indirect Addressing Modes
3. Register Addressing Modes
4. Register Indirect Addressing Mode
5. Auto-Increment and Auto-Decrement Addressing Modes
6. Displacement Based Addressing Modes

1.Implied and Immediate Addressing Modes:

Implied Addressing Mode:

Implied Addressing Mode also known as "Implicit" or "Inherent“ addressing mode is the addressing mode in which, no operand(register or memory location or data) is specified in the instruction. As in this mode the operand are specified implicit in the definition of instruction.

“Complement Accumulator” is an Implied Mode instruction because the operand in the accumulator register is implied in the definition of instruction. In assembly language it is written as:

CMA: Take complement of content of AC Similarly, the instruction,

RLC: Rotate the content of Accumulator is an implied mode instruction.

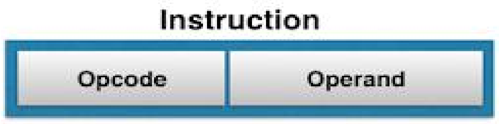
All Register-Reference instruction that use an accumulator and Zero-Address instruction in a Stack Organised Computer are implied mode instructions, because in Register reference operand implied in accumulator and in Zero-Address instruction, the operand implied on the Top of Stack.

Immediate Addressing Mode:

In Immediate Addressing Mode operand is specified in the instruction itself. In other words, an immediate mode instruction has an operand field rather than an address field, which contain actual operand to be used in conjunction with the operand specified in the instruction. That is, in this mode, the format of instruction is:

As an example: The Instruction:

MVI 06	Move 06 to the accumulator
ADD 05	ADD 05 to the content of accumulator



- One of the operand is mentioned directly.
- Data is available as a part of instruction.
- Data is 8 0r 16 bit long.
- No memory reference is needed to fetch data

Immediate Mode :Eg.

Example 1 :

MOV CL, 03H

03 – 8 bit immediate source operand

CL – 8 bit register destination operand

Example 2:

ADD AX, 0525H

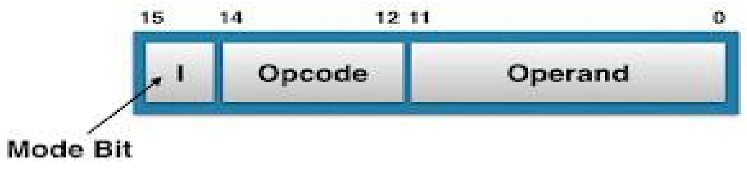
0525 – 16 bit immediate source operand

AX – 16 bit register destination operand.

2. Direct and Indirect Addressing Modes

The instruction format for direct and indirect addressing mode is shown below:

It consists of 3-bit opcode, 12-bit address and a mode bit designated as(I).**The mode bit (I) is zero for Direct Address and 1 for Indirect Address**. Now we will discuss about each in detail one by one.



Instruction Format

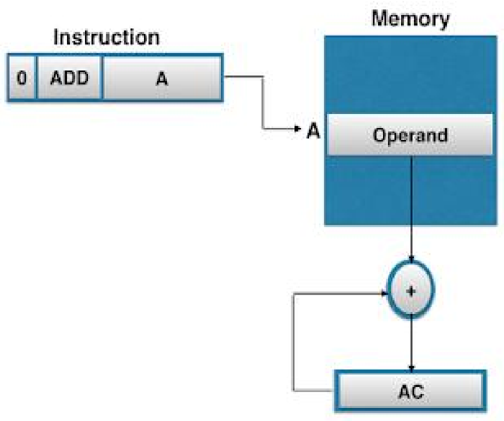
Direct Addressing Mode

Direct Addressing Mode is also known as "Absolute Addressing Mode". In this mode the address of data(operand) is specified in the instruction itself. That is, in this type of mode, the operand resides in memory and its address is given directly by the address field of the instruction. Means, in other words, in this mode, the address field contain the Effective Address of operand i.e., EA=A

As an example: Consider the instruction:

ADD A Means add contents of cell A to accumulator .

It Would look like as shown below:



Here, we see that in it Memory Address=Operand.

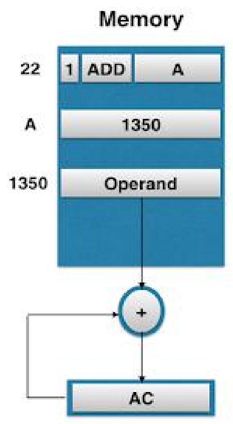
Indirect Addressing Mode:

In this mode, the address field of instruction gives the memory address where on, the operand is stored in memory. That is, in this mode, the address field of the instruction gives the address where the "Effective Address" is stored in memory. i.e., EA=(A)

Means, here, Control fetches the instruction from memory and then uses its address part to access memory again to read Effective Address.

As an example: Consider the instruction:

ADD (A) Means adds the content of cell pointed to contents of A to Accumulator. It look like as shown in figure below:



Thus in it, AC <-- M[M[A]]

[M=Memory]

i.e., (A)=1350=EA

3.Register Addressing Mode:

In Register Addressing Mode, the operands are in registers that reside within the CPU. That is, in this mode, instruction specifies a register in CPU, which contain the operand. It is like Direct Addressing Mode, the only difference is that the address field refers to a register instead of memory location.

i.e., EA=R

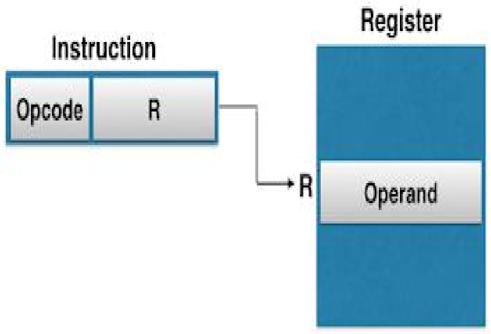
It look like as:

Example of such instructions are:

MOV AX, BX	Move contents of Register BX to AX
ADD AX, BX	Add the contents of register BX to AX

Here, AX, BX are used as register names which is of 16-bit register.

Thus, for a Register Addressing Mode, there is no need to compute the actual address as the operand is in a register and to get operand there is no memory access involved



4. Register Indirect Addressing Mode:

In Register Indirect Addressing Mode, the instruction specifies a register in CPU whose contents give the operand in memory. In other words, the selected register contain the address of operand rather than the operand itself. That is,

i.e., EA=(R)

Means, control fetches instruction from memory and then uses its address to access Register and looks in Register(R) for effective address of operand in memory.

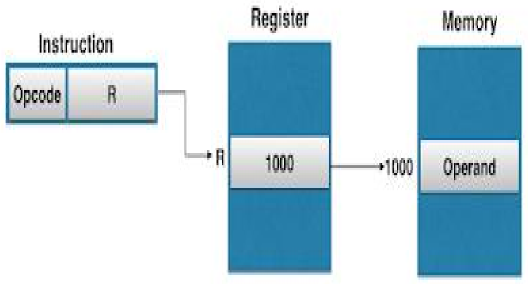
It look like as:

Here, the parentheses are to be interpreted as meaning contents of.

Example of such instructions are:

MOV AL, [BX]

Code example in Register:



MOV BX, 1000H

MOV 1000H, operand

From above example, it is clear that, the instruction(MOV AL, [BX]) specifies a register[BX], and in coding of register, we see that, when we move register [BX], the register contain the address of operand(1000H) rather than address itself.

5. Auto-increment and Auto-decrement Addressing Modes

These are similar to Register indirect Addressing Mode except that the register is incremented or decremented after(or before) its value is used to access memory. These modes are required because when the address stored in register refers to a table of data in memory, then it is necessary to increment or decrement the register after every access to table so that next value is accessed from memory.

Thus, these addressing modes are common requirements in computer.

Auto-increment Addressing Mode:

Auto-increment Addressing Mode are similar to Register Indirect Addressing Mode except that the register is incremented after its value is loaded (or accessed) at another location like accumulator(AC).

That is, in this case also, the Effective Address is equal to

EA=(R)

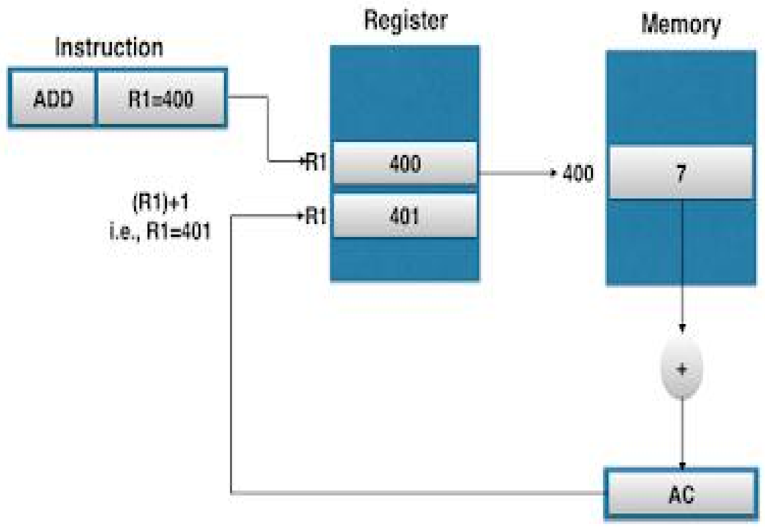
But, after accessing operand, register is incremented by 1.

As an example:

It look like as shown below:

Here, we see that effective address is (R)=400 and operand in AC is 7. And after loading R1 is incremented by 1.It becomes 401.

Means, here we see that, in the Auto-increment mode, the R1 register is increment to 401 after execution of instruction.



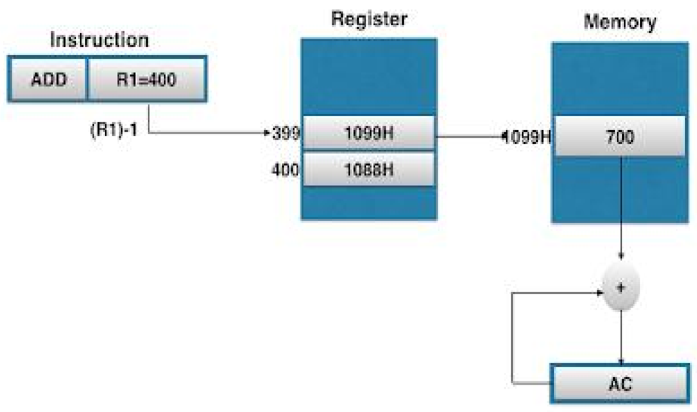
Auto-decrement Addressing Mode:

Auto-decrement Addressing Mode is reverse of auto-increment , as in it the register is decrement before the execution of the instruction. That is, in this case, effective address is equal to

EA=(R) - 1

As an example:

It look like as shown below:



Here, we see that, in the Auto-decrement mode, the register R1 is decremented to 399 prior to execution of the instruction, means the operand is loaded to accumulator, is of address 1099H in memory, instead of 1088H.Thus, in this case effective address is 1099H and contents loaded into accumulator is 700.

6. Displacement Based Addressing Modes:

Displacement Based Addressing Modes is a powerful addressing mode as it is a combination of direct addressing or register indirect addressing mode. i.e., EA=A+(R)

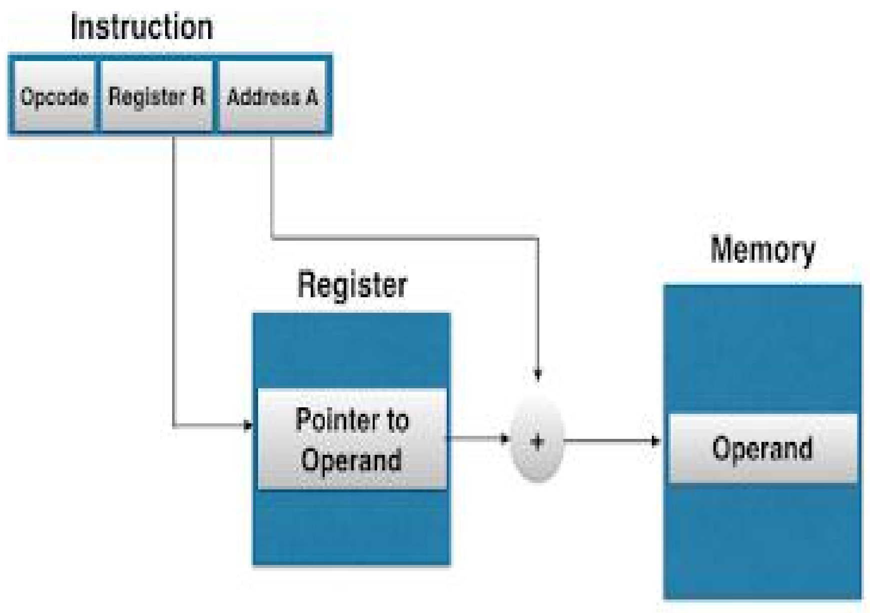
Means, Displacement Addressing Modes requires that the instruction have two address fields, at least one of which is explicit means, one is address field indicate direct address and other indicate indirect address.

That is, value contained in one addressing field is A, which is used directly and the value in other address field is R, which refers to a register whose contents are to be added to produce effective address.

There are three areas where Displacement Addressing modes are used. In other words, Displacement Based Addressing Modes are of three types. These are:

1. Relative Addressing Mode
2. Base Register Addressing Mode
3. Indexing Addressing Mode

Now we will explore to each one by one.



1. Relative Addressing Mode:

In Relative Addressing Mode , the contents of program counter is added to the address part of instruction to obtain the Effective Address.

That is, in Relative Addressing Mode, the address field of the instruction is added to implicitly reference register Program Counter to obtain effective address.

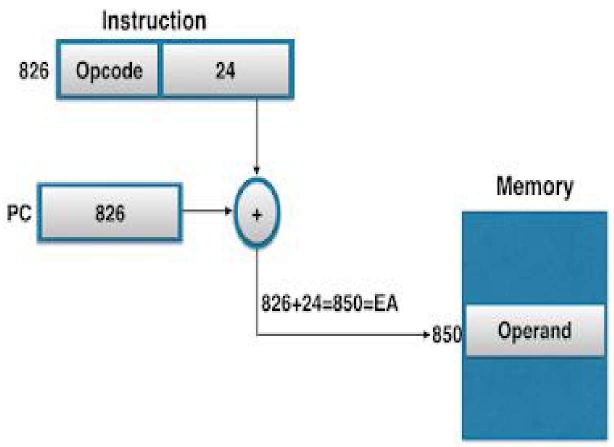
i.e., EA=A+PC

It becomes clear with an example:

Assume that PC contains the no.- 825 and the address part of instruction contain the no.- 24, then the instruction at location 825 is read from memory during fetch phase and the Program Counter is then incremented by one to 826.

The effective address computation for relative address mode is 26+24=850

Thus, Effective Address is displacement relative to the address of instruction. Relative Addressing is often used with branch type instruction



2. Index Register Addressing Mode

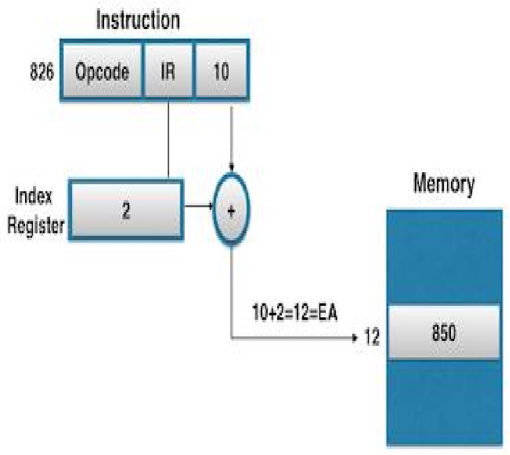
In indexed addressing mode, the content of Index Register is added to direct address part(or field) of instruction to obtain the effective address. Means, in it, the register indirect addressing field of instruction point to Index Register, which is a special CPU register that contain an Indexed value, and direct addressing field contain base address.

As, indexed type instruction make sense that data array is in memory and each operand in the array is stored in memory relative to base address. And the distance between the beginning address and the address of operand is the indexed value stored in indexed register.

Any operand in the array can be accessed with the same instruction, which provided that the index register contains the correct index value i.e., the index register can be incremented to facilitate access to consecutive operands.

Thus, in index addressing mode

EA=A+Index



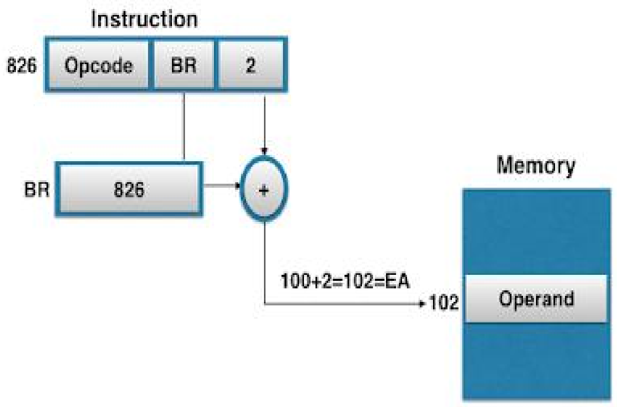
3. Base Register Addressing Mode:

In this mode, the content of the Base Register is added to the direct address part of the instruction to obtain the effective address.

Means, in it the register indirect address field point to the Base Register and to obtain EA, the contents of Instruction Register, is added to direct address part of the instruction.

This is similar to indexed addressing mode except that the register is now called as Base Register instead of Index Register.

That is, the EA=A+Base

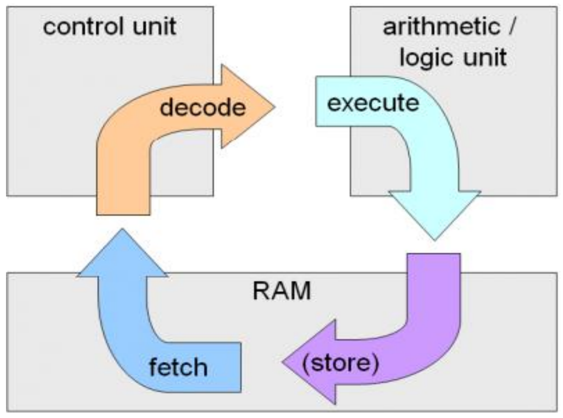


Thus, the difference between Base and Index mode is in the way they are used rather than the way they are computed. An Index Register is assumed to hold an index number that is relative to the address part of the instruction. And a Base Register is assumed to hold a base address and the direct address field of instruction gives a displacement relative to this base address.

Thus, the Base register addressing mode is used in computer to facilitate the relocation of programs in memory. Means, when programs and data are moved from one segment of memory to another, then Base address is changed, the displacement value of instruction do not change.

So, only the value of Base Register requires updation to reflect the beginning of new memory segment.

Instruction Cycle



Instructions are processed under the direction of the control unit in a step-by-step manner.

There are four fundamental steps in the instruction cycle:

1. **Fetch the instruction** The next instruction is fetched from the memory address that is currently stored in the Program Counter (PC), and stored in the Instruction register (IR). At the end of the fetch operation, the PC points to the next instruction that will be read at the next cycle.

2. **Decode the instruction** The decoder interprets the instruction. During this cycle the instruction inside the IR (instruction register) gets decoded.

3. **Execute** The Control Unit of CPU passes the decoded information as a sequence of control signals to the relevant function units of the CPU to perform the actions required by the instruction such as reading values from registers, passing them to the ALU to perform mathematical or logic functions on them, and writing the result back to a register. If the ALU is involved, it sends a condition signal back to the CU.

4. **Store result** The result generated by the operation is stored in the main memory, or sent to an output device. Based on the condition of any feedback from the ALU, Program Counter may be updated to a different address from which the next instruction will be fetched.

Data Bus: A data bus is a connection between the different parts of a computer that information is sent on.

Address Bus: The address bus is a data bus that is used to specify a physical address. A CPU will specify the memory location

You Tube Video :https://www.learncomputerscienceonline.com/instruction-cycle/

https://www.youtube.com/watch?list=RDCMUC0HzEBLlJxlrwBAHJ5S9JQg&v=OTDTdTYld2g&feature=emb_rel_end

COURSE PLAN

Select a course (a theory subject): **Computer Organization and Architecture**

Select a unit: **Unit I**

Select a Topic: **Interconnection of components**

1. **Objectives**

   To understand how different functional units of a computer system are connected.
   To know about the components which aids in interconnectivity.

2. **Outcomes**
   - Understand how various components are interconnected and integrated working of various hardware counterparts.
   - Ability to select appropriate computer systems for given application domains for future design of computer architecture.
   - Understand and develop processor for future computing hardwires to solves the problems of high end computing applications

3. **Pre-requisites**

   Prior knowledge on Functional components and operations of general computer system

4. **Terminology used other than normal known scientific / engg terms and their fundamental explanations / relations**

   Not Applicable

Plan for the lecture delivery

1. **How to plan for delivery – black board / ppt / animated ppt / (decide which is good for this topic)**
   Power Point Presentation/animated PPT

2. **How to explain the definition and terms with in it**
   What is BUS structure?

Types of bus structure

Different types of bus

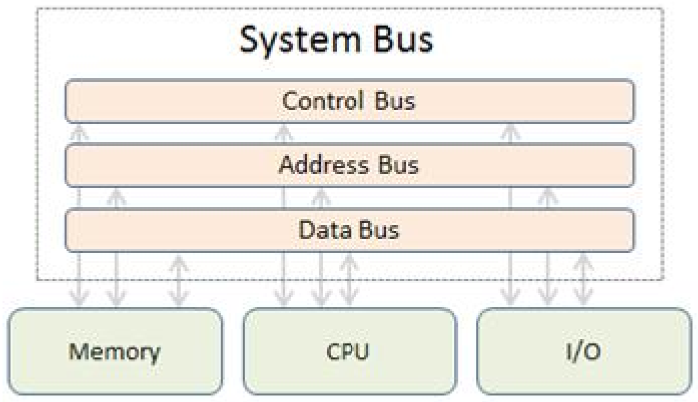
3. **How to start the need for any derivation / procedure / experiment / case study**

Case Study

We will explore two real-world examples of bus architectures: USB and SCSI.
We will provide an overview of the basics of the architecture.

Get on the Bus!
Recall that a bus in a computer is a path between sources and destinations. It's more like a set of train tracks than a Greyhound, but the concept remains. It is a conduit between CPU, Input/output (I/O), and Memory. As such it has three main tracks that include data, the address of the data, and control that manages the transfer of information and addresses. It's job is to join the CPU with memory and other devices.

Figure 1 shows an example:

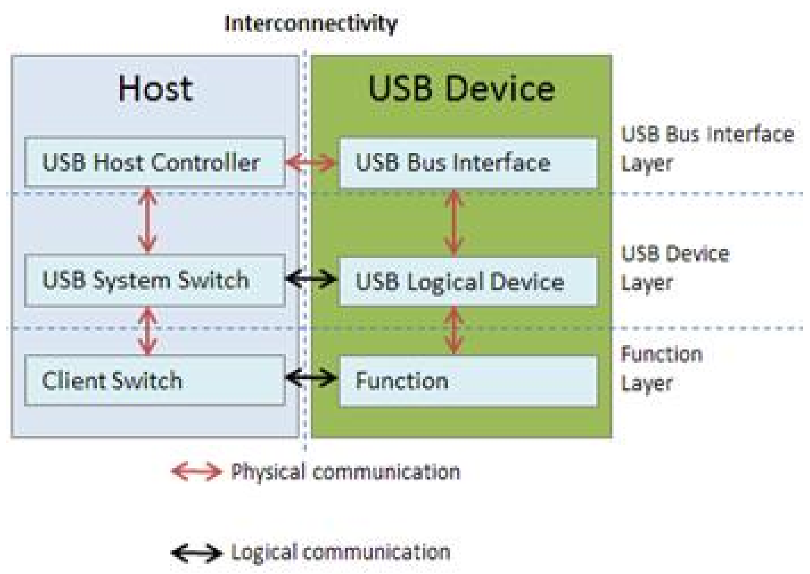


If you are using a computer you likely have a wireless or wired mouse connected. In either case, you've attached a USB device (either the mouse or the signal reader). The USB is a universal serial bus. Let's take a look at the architecture of the USB.

i. **USB Architecture**

The USB architecture was built to create a universal method for connecting peripherals to a computer.

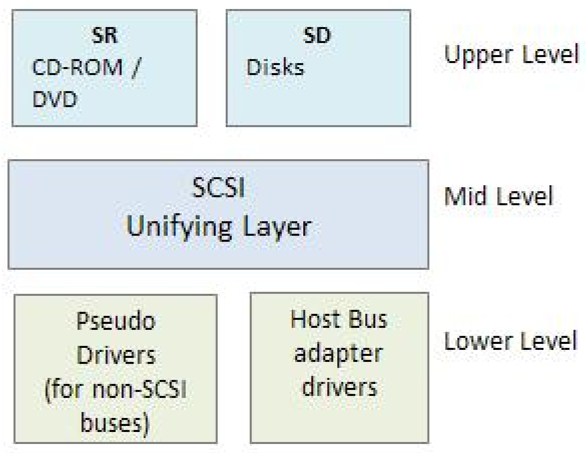
Figure 2 shows the architecture of a USB.



Comparing with Figure 1, we see the flow of information/communication between both the USB device and the host system. Inside the USB device, there is a Serial Interface Engine that allows for the sending and receiving of data. The Function layer represents the software within the USB.

ii. SCSI architecture

Like USB, Small Computer System Interface (SCSI) is a standard interface used to connect devices to a computer. SCSI is used to boost data transfer rates and performance for DVD drives, scanners, RAID, and other devices.



4. Physical meaning of math equations / **calculations**

   Not Applicable

5. **Units and their physical meaning to make them understand practical reality and comparison between different units**

   We have discussed different types of buses and its functions. Two types of bus architectures have been discussed. Functionalities of each bus should be differentiated and suitable architecture should be selected for implementation.

6. **What is the final conclusion?**

   Understand the role of different types of buses and how the bus acts as a important counterpart in interconnecting various functional units of a computer system.

7. **How to put it in a nut shell**

   BUS- interconnecting various functional units of a computer

8. **Important points for understanding / memorizing / make it long lasting**
   - BUS- A group of lines that serves as a connecting path for several devices
   - 3 types of buses – Address bus, Data Bus and Control Bus.

9. **Questions and cross questions in that topic to make them think beyond the topic.**
   1. What is a BUS structure?
   2. Which bus is used to connect the monitor to the CPU?
   3. What is the advantage of multiple bus organization over single Bus?
   4. Differentiate between address ,data and control bus.

10. **Final conclusions**

We have discussed about various bus structures and their benefits. We had a outline regarding how bus structure can be selected to increase the performance of hardware counterparts with varying speeds.

What is BUS?

In computer architecture,a bus is a communication system that transfers data between components inside a computer, or between computers.

BUS Structure

- There are many ways to connect different parts inside a computer together.

- The simplest and most common way of interconnecting various parts of the computer.
- To achieve a reasonable speed of operation, a computer must be organized so that all its units can handle one full word of data at a given time.
- A group of lines that serve as a connecting port for several devices is called a **bus**.
- In addition to the lines that carry the data, the bus must have lines for address and control purpose.

Types of BUS Structure

1. Single Bus structure
2. Multiple Bus structure

Single Bus structure:

The bus can be used for only one transfer at a time, only two units can actively use the bus at any given time. Bus control lines are used to arbitrate multiple requests for use of one bus.

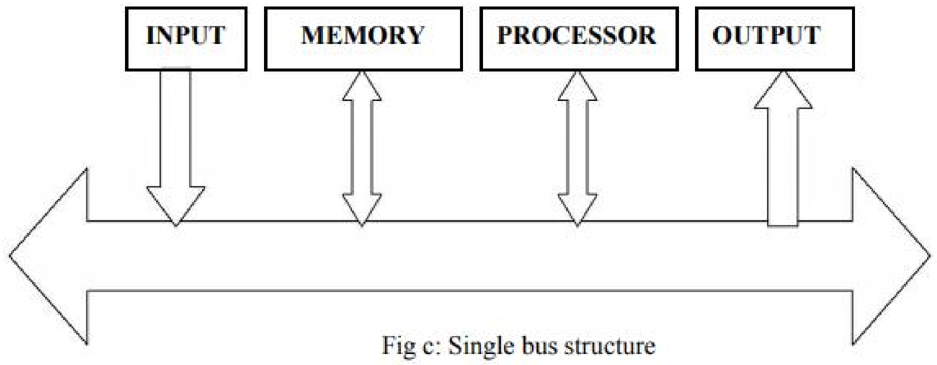


Fig c: Single bus structure

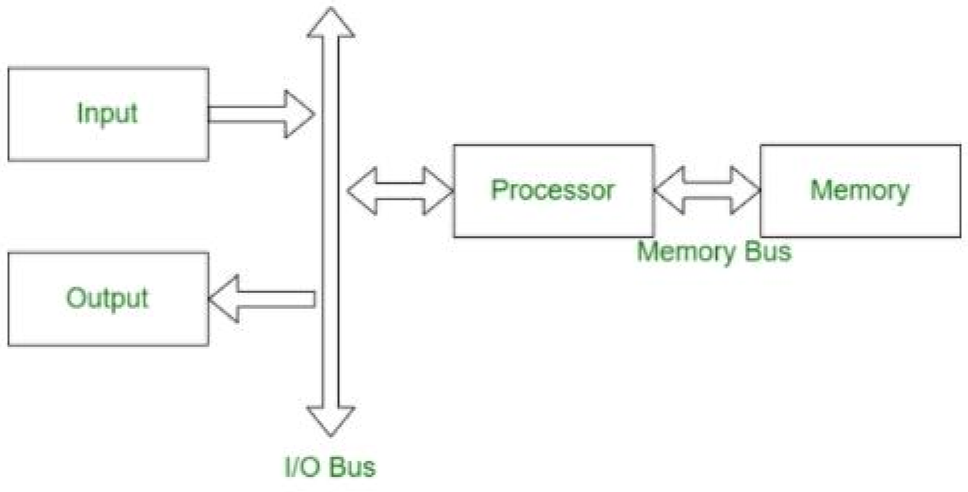
Single bus structure is

- Low cost
- Very flexible for attaching peripheral devices

Multiple Bus structure:

- Multiple bus structure certainly increases, the performance but also increases the cost significantly.
- All the interconnected devices are not of same speed & time, leads to a bit of a problem. This is solved by using cache registers (ie buffer registers).

- These buffers are electronic registers of small capacity when compared to the main memory but of comparable speed.
- The instructions from the processor at once are loaded into these buffers and then the complete transfer of data at a fast rate will take place.



Double Bus Structure

Benefits of multiple bus architecture are:

- Allows more number of devices to be connected to the computer
- Faster execution because many devices can work simultaneously resulting in performance enhancement
- Compatible with both old and new devices alike
  Multi-core processor that transfers more information and minimizes the wait time.

<u>Types of BUS:</u>

There are three types of buses.

1. **Address bus –**
   - It is a group of conducting wires which carries address only.
   - Address bus is unidirectional because data flow in one direction, from microprocessor to memory or from microprocessor to Input/output devices (That is, Out of Microprocessor).
   - The Length of the address bus determines the amount of memory a system can address. Such as a system with a 32-bit address bus can address 2^32 memory locations.
   - If each memory location holds one byte, the addressable memory space is 4 GB.
   - However, the actual amount of memory that can be accessed is usually much less than this theoretical limit due to chipset and motherboard limitations.

2. **Data bus –**

- It is a group of conducting wires which carries Data only.
- Data bus is bidirectional because data flow in both directions, from microprocessor to memory or Input/Output devices and from memory or Input/Output devices to microprocessor.
- When it is write operation, the processor will put the data (to be written) on the data bus, when it is read operation, the memory controller will get the data from specific memory block and put it into the data bus.
- The width of the data bus is directly related to the largest number that the bus can carry, such as an 8 bit bus can represent 2 to the power of 8 unique values, this equates to the number 0 to 255.A 16 bit bus can carry 0 to 65535.

3. **Control bus –**

- It is a group of conducting wires, which is used to generate timing and control signals to control all the associated peripherals, microprocessor uses control bus to process data, that is what to do with selected memory location.
- Some control signals are:
  - Memory read
  - Memory write
  - I/O read
  - I/O Write
  - Opcode fetch
- If one line of control bus may be the read/write line.If the wire is low (no electricity flowing) then the memory is read, if the wire is high (electricity is flowing) then the memory is written.