**UNIT - I OVERVIEW AND INSTRUCTION**

**COMPUTER ARCHITECTURE**

• **Computer Architecture** deals with designing and implementation of instruction set, information format and memory addressing techniques of a computer.

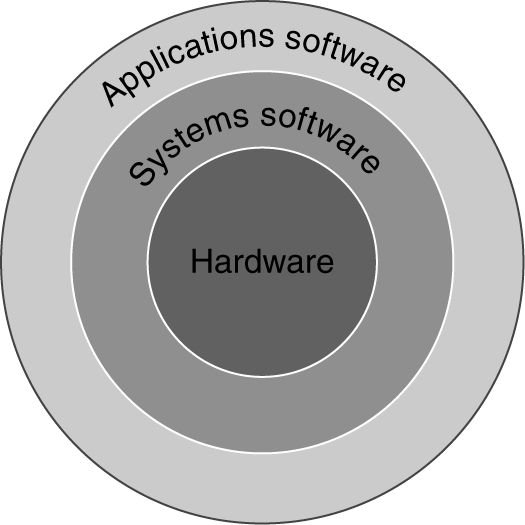
• **Computer Organization** refers to the operational units and their interconnections that describe the function and design of various units of a computer.

• A **Computer Architect** performs instruction set design, and memory addressing modes.

**1.1. Functional Unit - Components of a computer**

1. Hardware component

2. Software component



**1.1.1 Hardware component**

The electronic components interconnected in the computer system constitute the hardware components of a system.

A computer consists of the following functional units.

 Input Unit

 Central Processing Unit

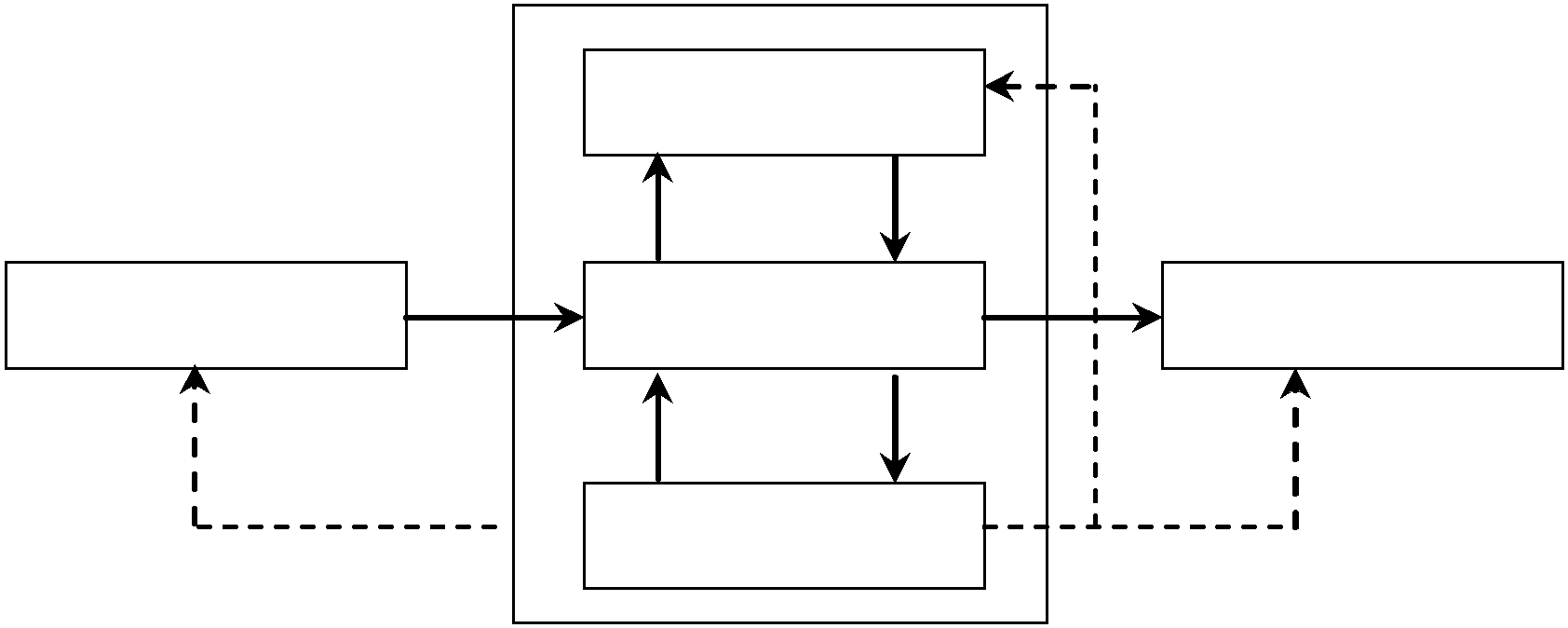
o Memory Unit – Primary memory, secondary memory, Cache memory, Registers

o Arithmetic and logic Unit

o Control Unit

 Output Unit

**Functional units of a Computer**



ALU

Input unit

Memory unit

Output unit

Control unit

**1.1.1.1 Input Unit**

 They are electromechanical devices that allow the user to provide information into the computer for analysis and storage inside the CPU.

 Input device captures information and translates it into a form that can be processed by the CPU.

 Computer accepts input in two ways. They are,

o Manual entry  the information is entered using keyboard or mouse.

o Direct entry  the information is fed into the computer automatically from a source document like barcode.

 Examples for Input devices: Keyboard, pointing devices like Mouse, Joystick.

 Whenever a key is pressed, the corresponding letter or digit is automatically translated into its corresponding binary code

 It is then transmitted over a cable to either the computer memory or the processor.

**1.1.1.2 Central Processing Unit (CPU)**

 It is referred as **‘the brain of a computer system’**.

 It converts data (input) into meaningful information (output).

 It is a highly complex, extensive set of electronic circuitry which executes stored program instructions called software.

 It controls all internal and external devices and performs arithmetic and logic operations

 It controls the usage of main memory to store data and instructions and controls the sequence of operations.

 It consists of three main subsystems.

o **Arithmetic and Logic Unit (ALU)**

o **Memory Unit**

o **Control Unit (CU)**

**1.1.1.3 Arithmetic Logic Unit (ALU)**

It contains the electronic circuitry that executes all arithmetic and logical operations on the data. ALU comprises of two units

**Arithmetic Unit**

 Contains the circuitry that is responsible for performing the actual computing and carrying out the arithmetic calculations (+, -, \*, /).

 To perform these operations, operands from the main memory is bought into processor

 After performing the operation results are stored in the memory location

 It can perform these operations at very high speed.

**Logic Unit**

 It enables the CPU to perform logical operations based on the instructions provided to it.

 Example: Logical comparison between data. ( Logical Operations : =, <, > conditions)

**1.1.1.4 Memory Unit**

 Memory refers to the electronic holding place for instructions and data.

 Memory also stores the intermediate results and output.

 Memory is mainly classified into two categories: primary and secondary.

**Primary Memory**

 It is also knows as main memory/ internal memory/ in-built memory.

 It stores data and instructions for processing.

 It is an integral component of CPU.

 It is a fast memory that operates at electronic speeds.

 The memory contains large no of semiconductor storage cells.

 Each cell carries 1 bit of information.

 The Cells are processed in a group of fixed size called Words.

 To provide easy access to any word in a memory, a distinct address is associated with each word location.

 Addresses are numbers that identify successive locations.

 The number of bits in each word is called the word length.

 The word length ranges from 16 to 64 bits.

 It can be classified as random access memory (RAM) and Read only memory (ROM).

**Differences between RAM and ROM**

|  |  |  |
| --- | --- | --- |
| **Features** | **RAM** | **ROM** |
| **Stands for** | Random Access Memory | Read-only memory |
| **Volatility** | RAM is volatilei.e. its contents are lost when the device is powered off. | It is non-volatile i.e. its contents are retained even when the device is powered off. |
| **Types** | The two main types of RAM are static RAM and dynamic RAM. | The types of ROM include PROM, EPROM and EEPROM. |
| **Use** | RAM allows the computer to read data quickly to run applications. It allows reading and writing. | ROM stores the program required to initially boot the computer.  It only allows reading. |
| **Definition** | Random Access Memory or RAM  is a form of data storage that can be accessed randomly at  any time, in any order and from  any physical location. | Read-only memory or ROM is also a form of data storage that cannot be easily altered or reprogrammed. |

**Secondary Memory**

 It is also known as auxiliary memory or external memory.

 It is used for storing software programs and data.

 It is less expensive and stores huge volume of data than primary memory.

 The data and instructions stored on such devices are permanent in nature.

 It can be removed only if the user wants or if the device is destroyed.

 Example: Pen drive, Floppy Disk, Compact Disk, External hard disk, etc.

**Differences between primary and secondary memory**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Primary Memory** | **Secondary Memory** |
| 1. | It is also called as Main or Internal or Built-in memory. | It is also known as Secondary or  Auxiliary or External memory. |
| 2. | It is present inside the computer. | It is present external to the computer that can be connected. |
| 3. | It is smaller in size and holds limited data. | It is larger in size and holds massive amount of data. |
| 4. | Due to its locality, it transfers data faster. | Since it is outside the CPU, it is slower when compared to primary memory. |
| 5. | It is costlier than secondary memory. | It is cheaper than main memory. |
| 6. | Example: RAM, ROM | Example: Disk drives, optical disks, magnetic tape drives. |

**Cache memory**

 Cache is a high speed memory located in between RAM and the CPU.

 It increases the speed of processing since it holds the most frequently used data.

 It is highly expensive and smaller in size.

 It is present in two or three forms in a system – L1, L2 L3 cache memories.

 The memory ranges from 256 KB to 2 MB.

**Register memory**

 Registers are special-purpose, high speed temporary memory units

 It holds various types of information such as data, instructions, addresses and the intermediate result of calculations.

 It holds the information that the CPU is currently working on.

 It is said to be **“CPU’s working memory**” or an additional storage location that offers the advantage of speed.

 Registers are of two types: general purpose and special purpose registers.

o **General purpose registers**

 These are a set of registers that store temporary data and addresses by the programmer.

 It includes floating point registers, constant registers, vector registers etc.

o **Special purpose registers**

 These are registers that are meant for performing special purposes by the CPU.

 They include Program counter (PC) , accumulator (ACC), instruction register (IR), memory address register (MAR), memory buffer register (MBR), memory data register (MDR) etc.

 Program counter contains the address of the next instruction to be processed.

 Accumulator stores the result of various arithmetic and logical operations.

 Instruction register holds the current instruction that is fetched.

 MAR contains the address of the next location in the memory that is to be accessed.

 MBR stores the temporary data and MDR stores the operands of the expression processed.

**1.1.1.5 Control Unit**

 This unit checks the correctness of sequence of operations.

 It fetches the instructions from the primary memory, interprets them and ensures correct execution of the program.

 It also controls the input and output devices

 Directs the overall functioning of the other units of the computer.

 Control Unit

o Supervises and controls the path of information that runs over the processor.

o Organizes the various activities of those units that lie inside it.

o Guides the flow of data through the different parts of the computer.

o Interprets the instructions.

o Regulates the time controls of the processor.

o Sends and receives control signals from various peripheral devices.

**1.1.1.6 Output Devices**

 Output Devices take the machine-coded output results from the CPU and convert them into a form that is easily readable by human beings.

 The output can be obtained in two forms: hardcopy and softcopy.

o The physical form of the output is called as hardcopy.

o The output which resides in the memory is called as softcopy.

 Example: Monitors, Printers, Plotters and audio response etc.

**1.1.2 Software Component**

 Software is the collection of instructions written in a computer language.

 It is responsible for controlling, integrating and managing the hardware components of a computer and to accomplish a specific task.

 Software instructs the hardware to perform the desirable task to be done.

**Types**

 System Software

 Application Software

**1.1.2.1 System Software**

 System software is a program that manages and supports the computer resources and operations of a computer system.

 It executes various tasks such as processing data and information, controlling hardware components, and allowing users to use application software.

o It is more transparent and less noticed by the users.

o They usually interact with the hardware or the applications.

o Basic functionality includes file management, visual display, keyboard input, etc.

 Example : **Operating systems, device drivers, language translators, text editors, utilities, loaders, linkers, etc.**

**Operating System**

 An **operating system** (**OS**) is a collection of software that manages computer hardware resources and provides common services for computer programs.

 The operating system is a vital component of the system software in a computer system. Application programs require an operating system to function.

 The functions of OS include Disk Access, Memory Management, Task Scheduling, and User Interfacing.

 Provides a software platform on top of which other programs run.

 Example: MS –DOS, WINDOWS, LINUX, UNIX.

**Device Drivers**

 These are system programs which are responsible for proper functioning of devices.

 Whenever a new device is added to a computer system the driver must be installed before the device is used.

 It acts as a translator between the device and the program that uses the device.

 It is not an independent program; it assists or is assisted by the OS for proper functioning.

 Example: printer, monitor, mouse, keyboard.

**Programming Language Translators**

 The language translators transform the instructions prepared by programmers in a high level language into the form which can be understood by the computer.

 Translators are divided into 3 categories: compiler, interpreter, and assembler.

 **Compiler**

 Compiler translates the high level programming language into machine language.

 It translates source code into object code.

 It can be used for larger applications.

 Example: C, C++, PASCAL compilers.

 **Interpreter**

 Interpreters translate the source code into object code in line-by-line manner, without looking at the entire program.

 Programs produced by compilers run much faster than interpreter.

 It is easier to modify the source code.

 Example: Basic.

 **Assembler**

 Assemblers translate assembly language program into machine language.

**System Utility**

 These programs perform day-to-day tasks related to the maintenance of the computer system.

 They are used to support, enhance and secure existing programs and data in the computer system.

 They are generally small programs having specific task to perform.

**1.1.2.2 Application Software**

 This is the most used software by the users.

 It is used to accomplish specific tasks.

 Application software consists of a single program (Ex. Notepad) or a collection of programs (software package) (Ex. Microsoft Office Suite).

Some of the most commonly used application software are as follows.

**Word Processors**

 A word processor is software used to compose, format, edit and print electronic documents.

 We can include pictures, graphs, and charts and allows changes in alignments, margins, font, and color and also allows spell checking.

 Example: Microsoft word, word perfect

**Spreadsheets**

 A spreadsheet application is a rectangular grid, which allows text, numbers and complex functions to be entered into a matrix of thousands of individual cells.

 Applications include payroll processing, financial record maintenance.

 Example: Microsoft Excel, Lotus 1-2-3.

**Image Editors**

 Image editor programs are designed specifically for capturing, creating, editing and manipulating images.

 The programs provide a variety of special features for creating and altering images.

 They also enable the user to create and superimpose layers, import and export graphic files, adjust an image and improve its appearance

 Example: Adobe Photoshop, Adobe Corel Draw.

**Database Management Systems**

 Database Management software is a collection of computer programs that supports structuring of the database in a standard format

 It provides tools for data input, verification, storage, retrieval, query and manipulation in an efficient manner.

 It controls the security and integrity of the database from unauthorized access.

 Example: Oracle, FoxPro.

**Presentation Applications**

 A Presentation is a means of assessment, which requires presentation providers to present their work orally in the presence of an audience.

 It combines both visual and verbal elements.

 Presentation software allows the user to create presentations by producing slides/hand-outs for the presentation of projects.

 Example: Microsoft PowerPoint.

**Desktop Publishing Software**

 The desktop publishing is a technique of using a personal computer to design images and pages, and assemble type and graphics, then using a printer to output the assembled pages onto paper.

 This software is used for creating magazines, books etc.

 Example: Adobe PageMaker, Quark Express.

**1.2 Basic Operational Concepts**

* The primary function of a computer system is to execute a program, sequence of instructions. These instructions are stored in computer memory.
* These instructions are executed to process data which are already loaded in the computer memory through some input devices.
* After processing the data, the result is either stored in the memory for further reference, or it is sent to the outside world through some output port.
* To perform the execution of an instruction, in addition to the arithmetic logic unit, and control unit, the processor contains a number of registers used for temporary storage of data and some special function registers.
* The special function registers include program counters (PC), instruction registers (IR), memory address registers (MAR) and memory and memory data registers (MDR).
* The Program counter is one of the most critical registers in CPU.
* The Program counter monitors the execution of instructions. It keeps track on which instruction is being executed and what the next instruction will be.
* The instruction register IR is used to hold the instruction that is currently being executed.
* The contents of IR are available to the control unit, which generate the timing signals that control, the various processing elements involved in executing the instruction.
* The two registers MAR and MDR are used to handle the data transfer between the main memory and the processor.
* The MAR holds the address of the main memory to or from which data is to be transferred.
* The MDR contains the data to be written into or read from the addressed word of the main memory.
* Whenever the processor is asked to communicate with devices, we say that the processor is servicing the devices. The processor can service these devices in one of the two ways.
* One way is to use the polling routine, and the other way is to use an interrupt.
* Polling enables the processor software to check each of the input and output devices frequently. During this check, the processor tests to see if any devices need servicing or not.
* Interrupt method provides an external asynchronous input that informs the processor that it should complete whatever instruction that is currently being executed and fetch a new routine that will service the requesting device.

**1.3 PERFORMANCE**

 The prime factors of the success of a computer are the speed and cost.

 Performance depends on how fast machine instructions can be brought into the processor for execution and how fast they can be executed.

 The performance of a computer is dependent on

o the design of the compiler o the machine instruction set o the hardware

**Terminologies**

**Response time**

 It is the time between the start and completion of a task.

 It refers the execution time of a set of instructions.

 The faster execution of instructions leads to reduction in response time, thus increases throughput.

**Throughput**

 It is the total work done in a unit time period.

**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 machine instruction.

**Clock**

 The Processor circuits are controlled by a timing signal called a clock.

**Clock Cycle**

 The clock defines a regular time interval called clock cycle.

**Bandwidth**

 The amount of data that can be transferred from one point to another in a given time period is called bandwidth.

 It is expressed in bits per second (bps).

**Clock cycles per instruction (CPI)**

 Average number of clock cycles per instruction for a program or program fragment. CPU clock cycles = Instructions for a program × Average clock cycles per instruction

**Execution of an Instruction**

 At the start of execution all program instructions and the required data are stored in the main memory.

 As execution proceeds, instructions are fetched one by one over the bus into the processor, and a copy is placed in the cache.

 When the execution of an instruction calls for data located in the main memory, the data are fetched and a copy is placed in the 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.

 To execute a machine instruction, the processor divides the action to be performed into a sequence of basic steps; each step can be completed in one clock cycle.

Clock Rate, R =1/P (measured in cycles per second)

where, P  Length of one clock cycle

**Basic Performance Equation**

Let

T  the processor time required to execute a program. N  the actual number of instruction executions

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

If the **clock rate** is **R** cycles/second, the program execution time is given by

T = (N\*S)/R

where, TPerformance Parameter

RClock Rate in cycles/sec

NActual number of instruction executions

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

To achieve high performance, the computer designer must reduce the value of T, which means reducing N and S, increasing R.

N, S<R

The value of N is reduced if the source program is compiled into fewer machine instructions. The value of S is reduced if instructions have a smaller number of basic steps to perform or if the execution of instructions is overlapped.

**Pipelining**

A considerable improvement in performance can be achieved by overlapping the execution of successive instructions. This technique is called pipelining.

**Superscalar Operation**

Multiple instruction pipelines can be implemented in the processor that allows several instructions 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 logic circuits faster, which reduces the time needed to complete a basic step. This allows the clock period, P, to be reduced and the clock rate, R, to be increased.

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

**Performance Improvement**

 To maximize the performance, minimize the response time or execution time for some task.

 The performance of the computer is directly related to performance and execution time for computer, X.

*Performance X*

 1

*Execution T ime X*

 For two computers X and Y, the performance of X is greater than Y then we have

*PerformanceX*

 *PerformanceY*

1  1

*Execution TimeX*

*Execution TimeY*

*E xecution T im eY*

**Example**

 *E xecution T im e X*

*Execution TimeY*  *Execution TimeX*

Time taken to run a program = 10s on A, 15s on B

Relative performance =Execution Time

B

A

/ Execution Time

=15s/10s

=1.5

So A is 1.5 times faster than B

**Measuring Performance**

 Measured in terms of seconds per program

 Defined as the total time taken to complete a task. The task includes

o disk access

o memory access

o I/O activities

o Execution of Instructions

 This time taken is known as wall-clock time / response time.

**CPU Time**

 Time the CPU spends computing for particular task and does not include the time waiting for I/O.

 This is also called as CPU execution time

**Formula**

*CPU time*   *CPU time spent in the program*(**user CPU time**)

*CPU time spent in the* operating system(**system CPU time**)

**Example**

Let

 User CPU time = 90.7 seconds

 System CPU time = 12.9 seconds

 Elapsed time = 2 minutes and 39 seconds (159 seconds)

 CPU time

CPU Time  90.7  12.9  0.65

159

**Performance Equation I**

*CPU execution time for a program*  *CPU clock cycles for a program*  *Clock cycle time*

Clock rate is inverse of clock cycle time



 CPU Execution time

*CPU execution time for a program*   *CPU clock cycles for a program*

*Clock rate*

 Performance is improved by reducing the length of the clock cycle or number of clock cycle required for a program

 Execution time depends on the number of instructions in a program

 Number of clock cycles

*CPU clock cycles*  *Instructions for a program*  *Average clock cycles per instruction*

 Clock cycles per instruction (CPI)

o Average number of clock cycles in which each instruction takes to execute

**Performance Equation II**

 CPU Execution Time in terms of instruction count, CPI and clock cycle time

CPU Execution time = Instruction count  CPI  Clock cycle time

(OR)

CPU Execution Time  Instruction Count CPI Clock Rate

**1.4 INSTRUCTIONS**

 An instruction is a piece of a program that performs an operation issued by the computer processor.

 Every instruction is defined by the instruction set of the processor.

**1.4.1 Instruction Set**

 A list of all the instructions with all their variants that can be executed by a processor is called instruction set.

 It is a group of commands defined by the processor in machine understandable language.

**Example**

 Arithmetic instructions Add, Subtract, Multiply and Divide

 Logic instructions  And (Conjunction) , Or (Disjunction), Not (negation)

 Control flow instructions  Goto, if ... goto, call, and return

 Data handling and memory instructions  Read, Write, Copy, Set, Load, Store

Instruction Sets are differentiated based on

 Operand storage in the CPU (data can be stored in a stack structure or in registers)

 Number of explicit operands per instruction (zero, one, two, and three address)

 Operand location (instructions can be classified as register-to-register, register- to-memory or memory-to-memory)

 Operations

 Type and size of operands (operands can be addresses, numbers, or even characters)

**Format**

An instruction has three fields, namely-

 **Operation code** (Opcode) specifies which type of operation to be performed.

 **Mode Field** specifies the way the operand or effective address is determined.

 **Address Field** specifies memory address or a processor register.

Opcode Mode field Address field

**1.4.2 Types of Instruction format**

**0 – Operand instruction**

 These instructions have no address fields.

 They are also called as Zero address instruction or Stack instruction.

 They do not have source / destination addresses. The address is implicit.

 All the operations are done using stack data structure.

 The operands are present on the top of the stack.

 In other words, the absolute address of the operand is held in a special register that is automatically incremented (or decremented) to point to the location of the top of the stack.

 Syntax

 Example

Stack\_Operation / Operation

 **Advantages**

To perform C = A + B, the instructions are,

PUSH A //Inserts the data A onto the stack PUSH B //Inserts the data B onto the stack ADD //Adds the value of A and B

POP C //Gets the added value from the stack

o It is a simple model of expression evaluation.

o The instructions are short.

 **Disadvantages**

o A stack can’t be randomly accessed.

o This makes it hard to generate effective code.

o Since the same stack is used for every operation, it creates a bottleneck.

**1 – Operand instruction**

 These instructions contain one address field.

 They are also known as one address instruction or Accumulator instruction.

 Accumulator (ACC) register is used for manipulation of data.

o All the operations are carried out between the accumulator register and a memory operand.

 Syntax

Operation Destination\_Location

 Example

ADD A is equivalent to ACC ACC + A Where, A  Destination operand

The Arithmetic Operation, C = A + B is performed as, LOAD A

ADD B STORE C

 **Advantage**

o The instructions are short.

 **Disadvantage**

o The accumulator is only a temporary storage so memory traffic is the highest for this approach also.

**2 - Operand Instructions**

 These instructions contain two address fields namely, the source and the destination.

 Each address field specifies either a processor register or a memory.

 They are also called as two – Address instructions or general purpose register instructions.

 Syntax

Operation Destination\_Location, Source\_Location

 Example

To perform C = A + B, an intermediate register R1 is used as,

 **Advantage**

LOAD R1, A

ADD R1, B or

STORE C, R1

LOAD R1,A LOAD R2, B ADD R1, R2

STORE C, R1

o Makes code generation easy.

 **Disadvantage**

o All operands must be named leading to longer instructions.

**3 - Operand instruction**

 These instructions contain three address fields.

 They are also called as three address instructions or general purpose register instruction.

 Register address field may be a processor register or a memory operand.

 **Syntax**

Operation Source1\_Location, Source2\_location, Destination\_Location

 **Example:** To perform C = A + B, the code is

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ADD A, B, C | or | MOVE R1, A | or | LOAD R1, A |
|  |  | ADD C, R1,B |  | LOAD R2, B |
|  |  |  |  | ADD R3, R1, R2 |
|  |  |  |  | STORE C, R3 |

 **Advantage**

o Makes code generation easy.

 **Disadvantage**

o All operands must be named leading to longer instructions.

**1.4.3 Instruction Execution**

The four phases in instruction execution are

 Fetch the Instruction from memory - the instruction is fetched from the memory location whose address is in Program Counter (PC) and is placed in the instruction register.

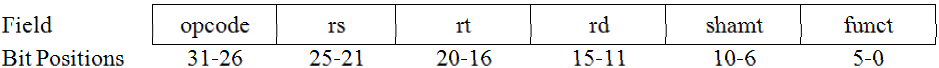
 Decode the Instruction.

 Execute the Instruction - the operands are fetched from the memory or processor registers, and the operation is performed.

 Store the result in the destination location.

**1.4.4 MIPS Instruction Formats**

  **R - Format**



o Opcode = 0

o Three register operands: rs, rt, and rd

 rs and rt - sources

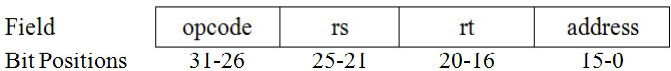
 rd - destination

o shamt field - used only for shifts

o funct field - The ALU function (add, sub, and, or, and slt) and is decoded by the ALU control design

|  |  |
| --- | --- |
| **ALU control lines** | **Function** |
| 000 | AND |
| 001 | OR |
| 010 | Add |
| 110 | Subtract |
| 111 | Set on less than |

 **I - Format**



o *For load and store instructions*

 Opcode = 35(for load) and Opcode = 43( for store)

 rs - the base register

 rt is

 For loads, the destination register for the loaded value

 For stores, the source register whose value should be stored into memory

 The memory address is computed as

Memory address = base register +16-bit address field

o *For branch instructions*

 Opcode = 4

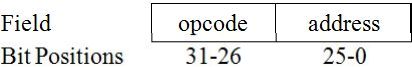
 rs and rt are the source registers that are compared for equality

 The branch target address is computed as

Target address = PC + (signed-extended 16-bit offset address

<< 2)

 **J – Format**



o Opcode = 2

o The destination address is computed as

Target address = PC [31-28]  (offset address << 2)

**1.5 LOGICAL INSTRUCTIONS**

 Instructions that perform logical operations and manipulate Boolean values are called as logical instructions.

 They include Logical AND, OR, and NOT.

**1.5.1 AND instruction**

 It contains three register operands.

 These instructions perform bitwise AND operation between the source registers and stores the result in the destination register.

 It is also called as conjunction operation.

 Syntax

Operation destination, source1, source2

 Example

AND R3, R1, R2 //Equivalent to R3 = R1 & R2

**1.5.2 OR instruction**

 OR instruction contains three register operands.

 It performs bitwise OR operation between the source registers and stores the result in the destination register.

 This is also called as disjunction operation.

 Syntax

Operation destination, source1, source2

 Example

OR R3, R1, R2 //Equivalent to R3 = R1 | R2

**1.5.3 NOR instruction**

 These instructions have three register operands.

 It performs bitwise NOR operation (OR operation followed by NOT) between two source registers and stores the result in the destination register.

 Syntax

Operation destination, source1, source2

 Example

NOR R1, R2, R3 // Equivalent to R1 = ~ (R2 **|** R3)

**1.5.4 AND Immediate (ANDI) instruction**

 This instruction contains three register operands.

 It performs bitwise AND operation between a source register and specified immediate value and stores the result in the destination register.

 Syntax

Operation destination, source1, Immediate\_Value

 Example

AND R1, R2, Immediate\_Value // Equivalent to R1 = R2 & Imm\_Val

**1.5.5 OR Immediate (ORI) instruction**

 This instruction has three register operands.

 It perform bitwise OR operation between a source registers and specified immediate value and finally stores the result in the destination register.

 Syntax

Operation destination, source1, Immediate\_Value

 Example

OR $1, $2, immediate\_Value //Equivalent to R1 = R2 | Imm\_Val

**1.5.6 Shift Left Logical instruction**

 This instruction contains three register operands.

 It shifts the given register value left by the shift amount listed in the instruction and stores the result in a third register.

 Syntax

Operation destination, source1, constant

 Example

SLL R1, R2, 10 // Equivalent to R1 = R2 << 10

**1.5.7 Shift Right Logical instruction**

 This instruction has three register operands.

 It shifts the specified register value right by the shift amount listed in the instruction and stores the result in a third register.

 Syntax

Operation destination, source1, Constant

 Example

SRL R1, R2, 10 // Equivalent to R1 = R2 >> 10

**1.5.8 Shift Right Arithmetic instruction**

 This instruction possesses three register operands.

 It shifts a register value right by the shift amount listed in the instruction and places the result in a third register.

 Syntax

Operation destination, source1, Constant

 Example

SRA R1, R2, 10 // Equivalent to R1 = R2 >> 10

**1. 6 CONTROL OPERATIONS / DECISION MAKING**

 Control statements are those that take a decision out of or without a condition.

 These instructions perform a test by evaluating a logical condition.

 Depending on the outcome of the condition, it modifies the PC to take the branch or continue to the next instruction.

 Control operations are of two types.

 Conditional branch  Performs branching based on a condition

 Unconditional branch  Performs branching without any condition

**1.6.1 Conditional Branch**

An instruction that directs the computer to another part of the program based on the results of a comparison is called conditional branching.

**1.6.1.1 BEQ Instruction**

 BEQ stands for branch on equal.

 The instruction checks if the two register values are equal. If so, it branches to the specified offset.

 Syntax

Operation source1, source2, offset

 Example

BEQ R1, R2, OFFSET // Equivalent to

if (R1==R2) goto L1;

a=b+c;

L1: a=b-c;

**16.1.2 BNE Instruction**

 BNE stands for branch on not equal.

 The instruction performs branching if the two registers values are not equal.

 Syntax

Operation source1, source2, offset

 Example

BNE R1, R2, OFFSET // Equivalent to

if (R1 != R2) goto L1;

a=b+c;

L1: a=b-c;

**1.6.2 Unconditional Branch**

An instruction that directs the computer to another part of the program without any test/condition operation is called unconditional branching.

**1.6.2.1 J Instruction**

 J stands for jump.

 It Jumps to the specified address.

 Syntax

Operation offset

 Example

J Target\_Address;

**1.6.2.2 JAL Instruction**

 JAL stands for Jump and link.

 The instruction jumps to the specified address and stores the return address.

 Syntax

Operation offset

 Example

JAL Target\_Address;

**1.6.2.3 JR Instruction**

 JR stands for Jump Register.

 It jumps to the address contained in the specified register R.

 Syntax

Operation source

 Example

JAL R1;

**1.7 MIPS ADDRESSING AND ADDRESSING MODES**

 There are various ways to specify the address of the operands for any given operations such as load, add or branch. The different ways of determining the address of the operands specified in an instruction are called addressing modes. In other words, addressing mode specifies how address is determined which may be either memory or register.

 MIPS addressing modes are as follows:

 **Immediate addressing:** The operand is a constant within the instruction itself

 **Register addressing:** The operand is a register

 **Base or displacement addressing:** The operand is at the memory location whose address is the sum of a register and a constant in the instruction

 **PC-relative addressing:** The branch address is the sum of the PC and a constant in the instruction

 **Pseudo-direct addressing:** The jump address is the 26 bits of the instruction concatenated with the upper bits of the PC

**1.7.1 Immediate Addressing**

 The instruction contains an immediate operand that has a constant value or an expression.

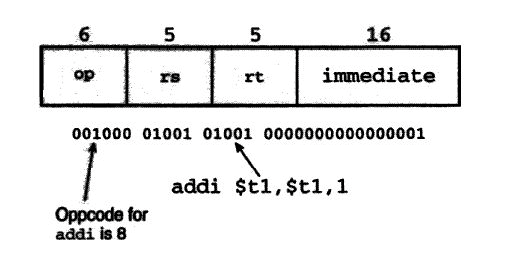
**Instruction**

op rs rt Immediate

 The operand is embedded inside the instruction

 Since the instruction does not require an extra memory access to fetch the operand, it executes faster.

 Example



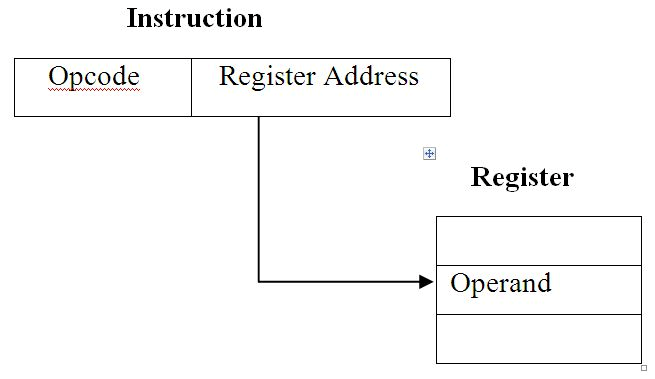
:

**1.7.2 Register Addressing**

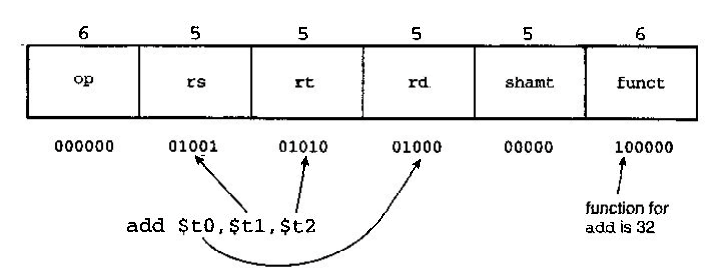
 It is the simplest addressing modes of all. The instruction works on register operands. It works much faster than other addressing modes because it does not involve with memory access.

 The register address is specified as a part of the instruction.

Effective Address, EA = Register Address, A



 **Example**



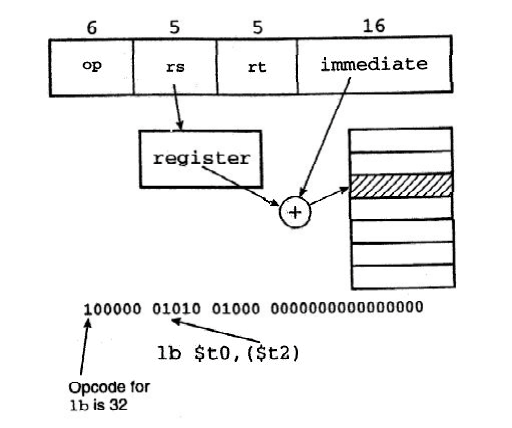
**1.7.3 Base Addressing**

 The address of the operand is the sum of the immediate and the value in a register

(rs). 16-bit immediate is a two’s complement number

 Effective Address = A + (R)

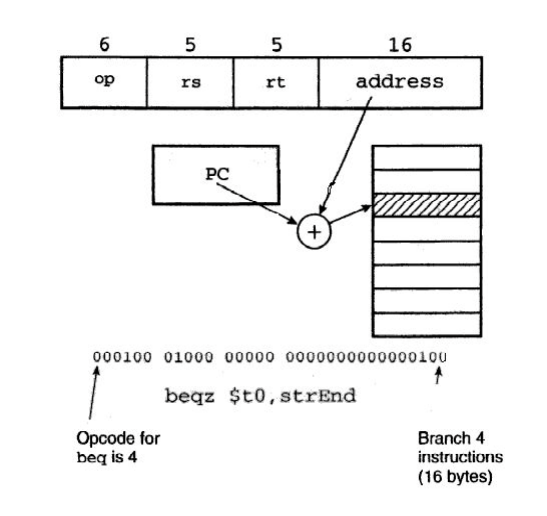
 Examples:



**1.7.4 PC Relative Addressing**

 For relative addressing, the implicitly referenced register is the program counter (PC).

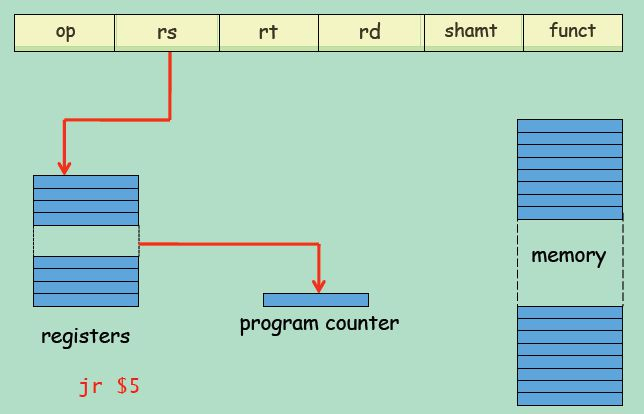
 PC-relative addressing is used for conditional branches. The address is the sum of the program counter and a constant in the instruction.Example



**1.7.5 Pseudo Direct Addressing**

In Register Direct Addressing, the value the (memory) effective address is in a register. It is also called “Indirect Addressing”. Special case of base addressing where offset is 0. Used with the jump register instructions.

Example: jr $31



**Direct Addressing**: the address is “the immediate”. 32-bit address cannot be embedded in a 32- bit instruction.

**Pseudodirect addressing**: 26 bits of the address is embedded as the immediate, and is used as the instruction offset within the current 256MB (64MWord) region defined by the MS 4 bits of the PC.

Example: j Label