BCA 4TH SEMSESTER

System Software & DCA2203

SET.:- 1

Q1. A. Explain different types of addressing modes.

Answer.:- In computer architecture, addressing modes are used to define the process for determining a data or instruction's memory address. There are several addressing modes, all of which are intended to promote quick and flexible access to memory regions.

- Immediate Addressing Mode: In this mode, the instruction explicitly specifies the operand value. For literal values and constants, this is helpful.
- Direct Addressing mode: The address of the memory region where the data is stored is the operand in the direct addressing mode. While simple and effective, this mode has a limited amount of RAM that is available.
- Indirect addressing mode: The operand in the indirect addressing mode is a memory region containing the address of the actual data. This mode makes memory access more flexible and is beneficial for data structures.
- Indexed Addressing Mode: In this mode, the operand's memory location is determined by adding an index register to the base address. This mode is helpful for accessing data structure or array items.
- Register Addressing Mode: In this mode, a register rather than a memory location is used to hold the operand. While quick and effective, this mode may have a limit on the amount of RAM available.
- Relative addressing mode: The operand in the relative addressing mode is a signed displacement value that has been added to the running programme counter. For branching and looping processes, this mode is helpful.
- The best addressing mode to use depends on the particular needs of the application being run. Each addressing mode has benefits and drawbacks of its own.

The best addressing mode to use depends on the particular needs of the application being run. Each addressing mode has benefits and drawbacks of its own.

Parsing.:- is the process of examining a string of symbols or code in a computer language to ascertain its grammatical structure in accordance with a set of rules or syntax. The goal of parsing is to discover the code's fundamental structure and translate it into a format that a computer can readily comprehend and process.

There are many reasons why parsing is important. First and foremost, it enables the computer to comprehend the syntax and organizational structure of the code, which is essential for accurate interpretation and execution. Second, it lets the computer to find mistakes and inconsistencies in the code and provide the proper error messages for the user. Lastly, it may be used to optimize the code for quicker and more effective execution.

There are several parsing algorithms, such as top-down and bottom-up parsing, as well as distinct parsing methods, such recursive descent and shift-reduce parsing. Depending on the kind of code being processed, each has different advantages and disadvantages and is used.

The term "parsing" refers to one of the most significant processes in the process of computer programming. At this point, the computer is able to understand the syntax and structure of the code, find faults and inconsistencies, and optimize the code so that it may be executed more quickly and with more efficiency.

Q2. A. What is Assembler? Explain different types of Assemblers.

Answer.:- An assembler is a program that translates assembly language code into machine code that can be executed by a computer. Assembly language is a low-level programming language that uses mnemonics to represent machine instructions.

According to the particular requirements of the programmer , many kinds of assemblers may be used.:-

- One-pass Assemblers: These assemblers just read the source code once, producing machine code without the need for any intermediary files.
- With a two-pass assembler, the source code is read twice. It creates a symbol table and determines the positions of labels and variables in memory during the first iteration. The information acquired in the first pass is used to produce the machine code in the second.

- Macro Assemblers: These assemblers enable the usage of pre-defined sequences of instructions called macros that may be reused throughout the code.
- Cross Assembler: A cross assembler produces code for platforms other than the one it is currently executing on. A cross-assembler, for instance, may produce code for an embedded system operating on a microcontroller when it is installed on a Windows computer.
- Integrated Development Environment (IDE) Assembler: An IDE is a comprehensive environment for authoring, editing, and debugging code. This kind of assembler is integrated with an IDE.

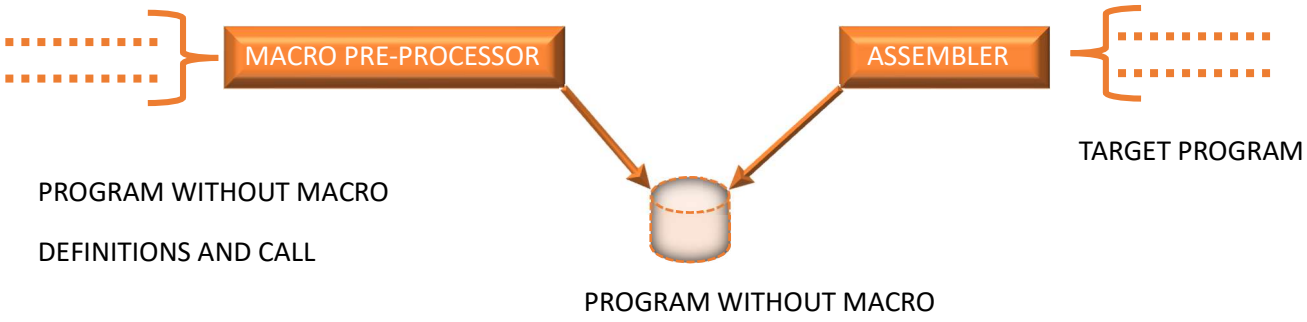
Programming for computers requires the use of assemblers, particularly when creating low-level systems like operating systems, device drivers, and firmware. They provide a quick and effective means to translate assembly language code into computer executable machine code, and the many assembler types offer flexibility and adaptation to various programming requirements.

Q2. B. Explain various Data Structures used in Macro Processor Design.

Answer.:- A programme called a macro processor is used in software development to automate the creation of code. By enabling programmers to generate reusable code chunks, or macros, it is intended to streamline and improve the efficiency of the coding process. Popular data structures include.:-

- "Symbol Table" is used to keep track of the names and values of symbols used in macros. It aids in locating and resolving any symbol references present in the macro code.
- Macro Definition Table: This table is used to record the appropriate macro names as well as the associated macro definitions. It aids in the resolution of macro calls and their expansion with the appropriate macro code.
- Argument Stack: During macro expansion, arguments are provided to macros and stored in this stack. It helps in giving macro definitions dynamic values.
- Expansion Buffer: During macro expansion, this data structure is utilized to hold the expanded macro code. By lowering the overhead of macro expansion, it aids in the efficient production of code.

- Conditional compilation tables are used to record data on the conditional compilation clauses that are used in macros. They provide effective conditional statement handling during macro expansion.
- The input text stream that is read by the macro processor is stored in this buffer. It aids in the text input being processed effectively.
- The output text stream produced by the macro processor is stored in this buffer, which is called the output buffer. It aids in the production of effective results.



Due to their assistance in the effective handling and processing of macros, these data structures are essential to the design of macro processors. They make it possible to write reusable sections of code and facilitate the simplification of the programming process. The programming language being used, and the unique requirements of the macro processor architecture influence the choice of data structures.

Q3.A. Explain program relocation concepts in detail.

Answer.:- The concept of "programme relocation" is used to alter the memory address references in a programme when it is loaded into a memory location different than the one in which it was meant to run. This is necessary because programmes are often created using absolute memory addresses that assume a certain position in memory. If the programme is loaded into a different location, those absolute addresses will no longer be acceptable. For the programme to operate correctly at the new memory location, these addresses must be adjusted during relocation.

Most of the time, the relocation process is executed when a programme is loaded into memory by the operating system loader. In order to determine which addresses need modification, the loader examines the program's code and data segments. It then performs the appropriate

adjustments. This allows the programme to access the correct memory areas regardless of where it is stored in memory.

There are several different types of relocation that may be done, including absolute relocation, relative relocation, and base relocation. Absolute relocation involves altering absolute addresses in the program's code and data segments. Modifying addresses that are relative to a certain memory location, such as a base address, is what relative relocation implies. Base relocation moves addresses in relation to a base address that is software-specific.

Q3.B. Describe Compiler. Explain different phases of compilation.

Answer.:- A software known as a compiler converts high-level programming languages into machine languages. It goes through a number of steps to transform the source code into a machine-executable format, including lexical analysis, syntax analysis, semantic analysis, code creation, and optimization.

The following are the different phases of compilation:

- Lexical Analysis: The lexical analysis, sometimes referred to as scanning, is the initial stage of compilation. At this stage, specific tokens including keywords, identifiers, operators, and literals are recognised by reading and analysing the source code.
- Semantic Analysis: The third stage of compilation is semantic analysis. This step checks the AST created in the preceding phase for semantic correctness, making sure that variable types are accurate and consistent, that functions and procedures are called accurately, and that the code complies with the semantics of the language.
- Intermediate Code Generation: The fourth stage of compilation is intermediate code generation. The AST created in the preceding step is transformed into an intermediate representation, such three-address code, in this phase.
- Code Optimization: The fifth stage of compilation is code optimization. To enhance the performance of the final machine code, the intermediate code created in the previous step is examined and optimised in this phase.
- Code Generation: The last stage of compilation is code generation, when the optimised intermediate code is converted into machine code for a particular target platform. Thisentails

producing assembly language or machine code that the computer's processor may use directly.

SET.:- 2

Q4.A. Differentiate between Line editors and Stream editors.

Answer.:- Text files may be edited using line editors and stream editors, two different kinds of text editors. Their methods and skills vary, however. The following are the variations between line editors and stream editors.:-

- **Interface.:-** Line editors provide a command-line interface via which users may communicate with the editor by inputting instructions, to which the editor reacts by altering the text file. Stream editors, in contrast, provide a non-interactive interface where users give a series of instructions in a script, and the editor performs them on the text file.
- **Text Manipulation.:-** Line editors let users edit a text file one line at a time. By employing line-oriented commands, users may modify, remove, or add lines. On the other hand, stream editors work on the full text file, applying a set of rules or transformations to the input stream and producing the updated output stream.
- **Flexibility.:-** Line editors are more versatile than stream editors because they let users interact with the text file and are better suited for making modest changes to the file. Stream editors, on the other hand, are better suited for batch processing, when the same set of alterations must be done to several files.
-

Q4.B. Explain the basic building blocks of an UPnP enabled network.

A. Answer.:- Devices may connect to one another and communicate without a human setup thanks to a group of networking protocols known as Universal Plug and Play (UPnP). A UPnP network is made up of these components:

An item of hardware or software known as a control point locates and controls UPnP devices on a network. It is capable of searching for objects, learning about their capabilities, and controlling their behaviour. A control point might be a computer, tablet, or phone running UPnP control software.

A UPnP control point can manage and control a piece of hardware known as a UPnP device. Routers and smart home appliances like thermostats and security cameras are examples of UPnP devices.

A UPnP device responds to requests from a control point and uses UPnP protocols to promote its services and capabilities.

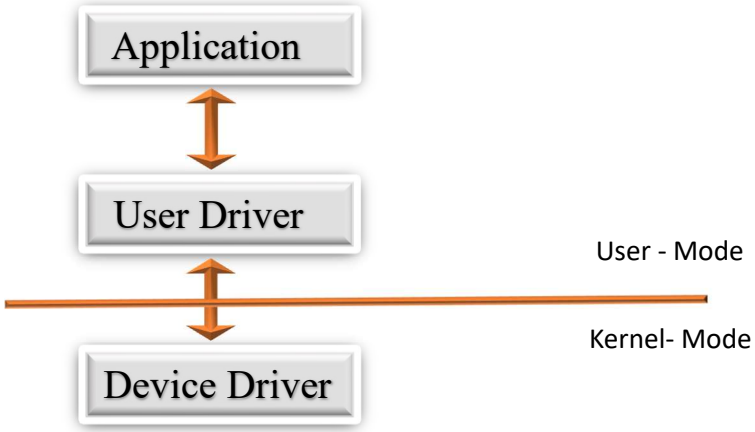
- **Service.:-** A UPnP device's software component that offers a particular capability or feature is known as a service. A control point may transmit print jobs to a printer, for instance, if the printer is a UPnP device. Several services that are each designated by a different service type may be found on a UPnP device.

  Discovery Protocol.:- UPnP devices and control points may find one another in a network using the discovery protocol. The protocol makes use of the Simple Service Discovery Protocol (SSDP) to send and receive messages across the network, enabling devices to inform the network of their existence and functionalities.
- Control Protocol.:- A control point may use a collection of rules and instructions that are collectively referred to as the control protocol in order to guide the operations of a UPnP device. The control protocol enables the control point to provide instructions to the device and receive responses from the device via the exchange of messages between the two parties.
- Due to their assistance in the effective handling and processing of macros, these data structures are essential to the design of macro processors. They make it possible to write reusable sections of code and facilitate the simplification of the programming process. The programming language being used, and the unique requirements of the macro processor architecture influence the choice of data structures.

Q5. What is a device driver? Explain the role of device drivers in terms of mechanism and policy?

With a piece of software called a device driver, an operating system may interact with and control a specific piece of hardware, such as a network card, printer, or scanner. Serving as a bridge between hardware and the operating system is the primary responsibility of a device driver. They can effectively work together because of this.

Mechanism-based and policy-based functions for device drivers may be distinguished. The precise method through which the device driver interacts with the operating system and hardware is referred to as a mechanism. This includes tasks like resource allocation, interrupt management, and sending and receiving data.



A device driver's policy incorporates a number of important factors. The first step is addressing mistakes, which entails finding them and taking action. Timeouts, damaged data, and hardware malfunctions are a few examples of this. The second factor is security, which entails making sure the device driver runs safely and doesn't bring vulnerabilities into the system. Data encryption, access control, and secure boot are a few examples of what falls under this category. The third factor is performance optimization, which entails raising device driver performance while lowering its demand on system resources. This may include practises like buffer management, data compression, and traffic shaping.

The ability to allow communication between various kinds of hardware and software is one of device drivers' distinctive qualities in computer networking. An example of a network card device driver is a local area network (LAN) device driver, which enables a computer to connect to it and interact with servers, printers, scanners, and other networked devices. A printer device driver, which allows a computer to send print jobs to a specific printer independent of the manufacturer or model of the printer, may offer functionality that is analogous to that described above.

Device drivers may also provide complex functionality that is not available via the operating system's built-in interfaces. A network card device driver, for instance, could include sophisticated traffic shaping features that let the computer prioritise certain network traffic over others. When the network is overcrowded or when certain types of traffic, such audio or video, need to take precedence in order to retain the service's quality, this may be useful.

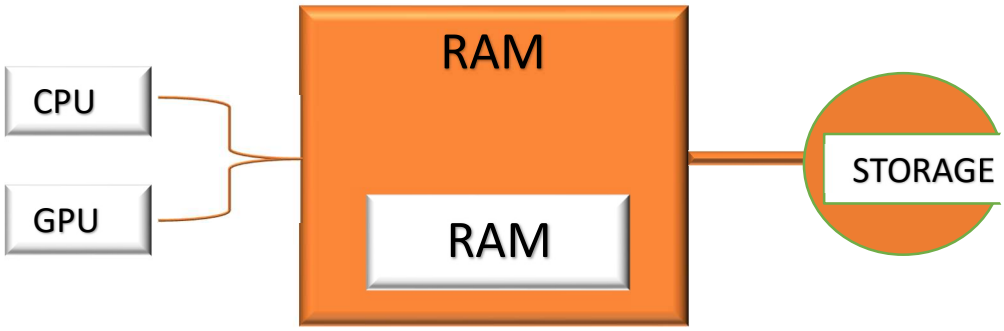
Q6.A. Write short notes on: (i) Worker threads (ii) Thread-safe methods.

Answer.:- Important ideas like worker threads and thread-safe procedures are essential for maintaining effective and secure communication between various nodes in a network.

i. A sort of thread called a worker thread is employed in a software to carry out certain duties. Several tasks may be carried out simultaneously since they are generated by the main thread and run separately from it. A server may manage numerous connections at once by using worker threads in networking to handle incoming requests from multiple clients. The server's speed and responsiveness may be considerably enhanced as a result of its increased ability to process requests concurrently.

ii. On the other hand, thread-safe techniques are ones that provide concurrent access and modification by many threads without risking conflicts or failures. The security and integrity of data being sent between various network nodes are ensured in networking through the use of thread-safe mechanisms. A data race may occur and lead to unexpected behaviour, for instance, if two threads attempt to write to the same memory region concurrently. Thread-safe techniques make sure that only one thread at a time may access a certain resource by using synchronisation mechanisms like locks, semaphores, and mutexes to prevent this.

iii. worker threads and thread-safe techniques are critical ideas in computer networking that assist guarantee effective and secure communication between various nodes in a network.



Q6.B. What are the diverse types of memory in phones? List some of the usages of memory of the android system.

Answer.:- Important ideas like worker threads and thread-safe procedures are essential for maintaining effective and secure communication between various nodes in a network.

- A sort of thread called a worker thread is employed in a software to carry out certain duties. Several tasks may be carried out simultaneously since they are generated by the main thread

and run separately from it. A server may manage numerous connections at once by using worker threads in networking to handle incoming requests from multiple clients. The server's speed and responsiveness may be considerably enhanced as a result of its increased ability to process requests concurrently.

- On the other hand, thread-safe techniques are ones that provide concurrent access and modification by many threads without risking conflicts or failures. The security and integrity of data being sent between various network nodes are ensured in networking through the use of thread-safe mechanisms. A data race may occur and lead to unexpected behaviour, for instance, if two threads attempt to write to the same memory region concurrently. Thread-safe techniques make sure that only one thread at a time may access a certain resource by using synchronisation mechanisms like locks, semaphores, and mutexes to prevent this.

worker threads and thread-safe techniques are critical ideas in computer networking that assist guarantee effective and secure communication between various nodes in a network.

Performance, storage capacity, and user experience are all impacted by memory, which is a crucial component of smartphones. Mobile devices employ a variety of memory types, including **internal storage, RAM, and external storage.**

The operating system, applications, and user data are all stored on internal storage, sometimes referred to as flash memory. Being non-volatile, it keeps the data it contains long after the device is switched off. The type and specs of the gadget determine the internal storage's size.

The operating system and running applications are stored in volatile memory called RAM, or random access memory. The number of applications and processes that may run concurrently on the device depends on the RAM amount and is quicker than internal storage.

The phone's storage may be increased beyond its internal storage using external storage, such a microSD card. It is also non-volatile and has the ability to store user data, documents, and media files.

Memory utilisation is crucial to Android devices' performance and user interface. By giving active applications and background processes priority, the operating system controls memory utilisation.

Among the several uses of RAM:

1) Installation and storage of application data: Applications are installed, and their data is saved in the device's internal storage.

2) Android caches app data in RAM in order to increase efficiency and decrease load times.

3) Android requires RAM to execute background activities and services, like push notifications and system upgrades.
4) Virtual memory: Android employs virtual memory to shift data between RAM and internal storage, enabling the device to manage greater quantities of data.

5) External storage: Android enables users to enhance the storage capacity of their smartphone by utilising external storage such as a microSD card.