Q.2   a. What are the basic functions of an Operating System?

Answer:
Operating system controls and coordinates the use of the hardware among the various applications programs for various uses. Operating system acts as resources allocator and manager. Since there are many possibly conflicting requests for resources the operating system must decide which requests are allocated resources to operating the computer system efficiently and fairly. Also operating system is control program which controls the user programs to prevent errors and improper use of the computer. It is especially concerned with the operation and control of I/O devices.

b. Explain the batch processing systems.

Answer:
Batch jobs are set up so they can be run to completion without manual intervention, so all input data is preselected through scripts or command-line parameters. This is in contrast to "online" or interactive programs which prompt the user for such input. A program takes a set of data files as input, processes the data, and produces a set of output data files. This operating environment is termed as "batch processing" because the input data are collected into batches of files and are processed in batches by the program.

c. Explain the following terms: (any TWO)
      (i)   Time sharing OS
      (ii) Multiprogramming systems
      (iii) Real time OS
      (iv) SPOOLING

Answer:
Time – sharing is sharing a computing resource among many users by means of multiprogramming and multi-tasking. Its introduction in the 1960s, and emergence as the prominent model of computing in the 1970s, represents a major technological shift in the history of computing. By allowing a large number of users to interact concurrently with a single computer, time-sharing dramatically lowered the cost of providing computing capability, made it possible for individuals and organizations to use a computer without owning one, and promoted the interactive use of computers and the development of new interactive applications**.**

A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time application requests.

A key characteristic of a RTOS is the level of its consistency concerning the amount of time it takes to accept and completed an application's task; the variability is jitter. A hard real-time operating system has less jitter than a soft real-time operating system. The chief design goal is not high throughput, but rather a guarantee of a soft or hard performance category. A RTOS that can usually or generally meet a deadline is a soft real-time OS,

but if it can meet a deadline deterministically it is a hard real-time OS. Scheduler flexibility enables a wider, computer-system orchestration of process priorities, but a real-time OS is more frequently dedicated to a narrow set of applications. Key factors in a real-time OS are minimal interrupt latency and minimal thread switching latency, but a real-time OS is valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time.

As machines with more and memory became available, it was possible to extend the idea of multiprogramming (or multiprocessing) as used in spooling batch systems to create systems that would load several jobs into memory at once and cycle through them in some order, working on each one for a specified period of time.

One difficult with simple batch systems is that the computer still needs to read the deck of cards before it can begin to execute the job. This means that the CPU is idle (or nearly so) during these relatively slow operations.
Since it is faster to read from a magnetic tape than from a deck of cards, it became common for computer centers to have one or more less powerful computers in addition to there main computer. The smaller computers were used to read a deck of cards onto a taps, so that the tape would contain many batch jobs. This tape was then loaded on the main computer and the jobs on the tape were executed. The output from the jobs would be written to another tape which would then be removed and loaded on a less powerful computer to produce any hardcopy or other desired output.
It was a logical extension of the timer idea described above to have a timer that would only let jobs execute for a short time before interrupting them so that the monitor could start an IO operation could proceed while the CPU was couching on a user program, little degradation in performance was noticed.
Since the computer can now perform IO in parallel with computation, it became possible to have the computer read a deck of cards to a tape, drum or disk and to write out to a tape printer while it was computing. This process is called SPOOLING:
Simultaneous Peripheral Operation OnLine.

Spooling batch systems were the first and the simplest of the multiprogramming systems.

Q.3   a. what are deadlock prevention techniques?

Answer:
Mutual exclusion: some resources such as read only files shouldn't be mutually exclusive. They should be sharable. But some resources such as printers must be mutually exclusive.

Hold and wait: To avoid this condition we have to ensure that if a process is requesting for a resources it should not hold any resources.

No pre-emption: If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is the process must wait) then all the resources currently being held are pre-empted (released autonomously)

Circular wait: the way to ensure that this condition never holds is to impose a total ordering of all the resources types, and to require that each process requests resources in an increasing order of enumeration.

b. Describe the FCFS scheduling algorithm.

Answer:
First-Come-Served algorithm is the simplest scheduling algorithm is the simplest scheduling algorithm**.** Processes are dispatched according to their arrival time on the ready queue. Being a non preemptive discipline once a process has a CPU it runs to completion. The FCFS scheduling is fair in the formal sense or human sense of fairness but it is unfair in the sense that long jobs make short jobs wait and unimportant jobs make important jobs wait**.**

FCFS is more predictable than most of other schemes since it offers time. FCFS scheme is not useful in scheduling interactive users because it cannot guarantee good response time. The code for FCFS scheduling is simple to write and understand. One of the major drawbacks of this scheme is that the average time is often quite long. The First-Come-First-Served algorithm is rarely used as a master scheme in modern operating systems but it is often embedded within other schemes.

c. What do you mean by deadlock avoidance?

 Answer:
Avoidance by careful resource allocation prevention by structurally negating one of the four necessary conditions.
Deadlock Prevention
Difference from avoidance is that here, the system itself is built in such a way that there are no deadlocks.
Make sure atleast one of the 4 deadlock is never satisfied.
This may however be even more conservative than deadlock avoidance strategy.
Attacking Mutex condition never grants exclusive access. But this may not be possible for several resources.
Attacking preemption not something you want to do.
Attacking hold and wait condition make a process holds at the most I resources at a time make all the requests at the beginning. All or nothing policy. If you feel, retry, eg.2-phase locking.
Attacking circular wait Order all the resources.
Make sure that the requests are issued in the correct order so that there are no cycles present in the resources graph.
Resources numbered 1...n. Resources can be requested only in increasing order. ie. you cannot request a resources whose no is less than any you may be holding.

Deadlock Avoidance

Avoid actions that may lead to deadlock.

Think of it as a state machine moving from 1 state to another as each instruction is executed.

Safe State
Safe state is one where
It is not a deadlocked state
There is some sequence by which all requests can be satisfied.

To avoid deadlocks, we try to make only those transitions that will take you from one safe state to another. We avoid transitions to unsafe state that is not deadlocked and is not safe).

Q.4   a. What is Context Switch?

Answer:
Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is known as a context switch. Context-switch time is pure overhead, because the system does no useful work while switching. Its speed varies from machine to machine, depending on the memory speed the number of registers which must be copied, the existed of special instructions(such as a single instruction to load or store all registers).

b. Explain the allocation methods of disk space.

Answer:
Disk Allocation Methods

Contiguous allocation each file occupies a set of consecutive addresses on disk each directory entry contains: file name starting address of the first block address = sector id (e.g., block = 4k) length in blocks usual dynamic storage allocation problem use first fit, or worst fit, or worst fit algorithms to manage storage if the file can increase in size either leave no extra space, and copy file elsewhere if it expands leave extra space.

Linked allocation each data block contains the block address of the next block in the file each directory entry contains the block address of the first block sometimes, also have a pointer) indexed allocation store all pointers together in an index table the index table is stored in several index blocks assume index table has been loaded into main memory.

c. Explain the critical section problem.

Answer:
A critical section is a piece of code that accesses shared resources (data structure or device) that must not be concurrently accessed by more than one thread of execution. A critical section will usually terminate in fixed time, and a thread task or process will have to wait a fixed time to enter and exit of the critical section to ensure exclusive use, for example a semaphore.

By carefully controlling which variables are modified inside and outside the critical section (usually by accessing important state only from within) concurrent access to that state is prevented. A critical section is typically used when a multithreaded program must update multiple related variables without a separate thread making conflicting changes to that data. In a related situation, a critical section may be used to ensure a shared resource for example a printer can only be accessed by one process at a time.

How critical sections are implemented varies among operating systems.

The simplest method is to prevent any change of processor control inside the critical section. On uni-processor systems this can be done by disabling interrupts on entry into the critical section avoiding systems, calls that can cause a context switch while inside the section and restoring interrupts to their previous state on exit. Any thread of execution entering any critical section anywhere in the system will with this implementation prevent any other thread including an interrupt from getting the CPU and therefore from entering any other critical section or indeed any code whatsoever, until the original any other critical section or, indeed any code whatsoever, until the original thread leaves its critical section.

This brute-force approach can be improved upon by using semaphores. to enter a critical section, a thread must obtain a semaphore, which it releases on leaving the section. Other thread are prevented from entering the critical section at the same time as the original thread but are free to gain control of the CPU and execute other code, including other critical sections that are protected by different semaphores.

Q.5   a. Briefly describe 'paging' in memory management.

Answer:
Paging is one of the memory-management schemes by which a computer can store and retrieve data from secondary storage for uses in main memory. In the paging memory-management scheme, the operating system retrieves data from secondary storage in same –size blocks called pages. The main advantage of paging is that it allows the physical address space of a process to be noncontiguous. Before the time paging was used systems had to fit whole programs into storage contiguously which caused various storage and fragmentation problems.[1]

Paging in an important part of virtual memory implementation in most contemporary general – purpose operating systems allowing them to use disk storage for data that does not fit into physical random – access memory (RAM)

The main functions of paging are performed when a program tries to access pages that are not currently mapped to physical memory (RAM). This situation is known as a page fault. The operating system must then take control and handle the page fault, in a manner invisible to the program. Therefore, the operating system must:

Determine the location of the data in auxiliary storage.
Obtain an empty page frame in RAM to use as a container for the data.
Load the requested data into the available page frame.
Update the page table to show the new data.
Return control to the program, transparently retrying the instruction that caused the page fault.

b. Explain the virtual memory concept.

Answer:
Virtual memory is a memory management technique developed for multitasking kernels. This technique virtualizes a computer architecture's various forms of computer data storage (such as random-access memory and disk storage) allowing a program to be designed as through there is only one kind of memory, "virtual" memory, which behaves liked directly addressable read/write memory (RAM) In order to implement virtual memory it is necessary for the computer system to have special memory management hardware. This hardware is often known as an MMU (Memory Management Unit). Without an MMu, when the CPU accesses RAM, the actual Ram locations never change.

c. What is fragmentation? Explain different types of fragmentation.

Answer:
Fragmentation occurs in a dynamic memory allocation system when many of the free blocks are too small to satisfy any request. External Fragmentation: External Fragmentation happens when a dynamic memory allocation algorithm allocates some memory and a small piece is left over that cannot be effectively used. If too much external fragmentation occurs, the amount of usable memory is drastically reduced. Total memory space exists to satisfy a request but it is not contiguous. Internal Fragmentation: is the space wasted inside of allocated memory blocks because or restriction on the allowed size of allocated blocks. Allocated memory may be slightly larger than requested memory this size difference is memory internal to a partition but not being used.

Q.6   a. What are the benefits of using "language processors"?

Answer:
Natural language processing (NLP) is a field of computer science and linguistics concerned with the interactions between computers and human (natural) languages; it began as a branch of artificial intelligence. In theory natural language processing is a very attractive method of human-computer interaction. Natural language understanding is sometimes referred to as an AI- complete problem because it seems to require extensive knowledge about the outside world and the ability to manipulate it.
Whether NLP is distinct from or identical to the field of computational linguistics is a matter of perspective. The Association for Computational Linguistics defines the latter as focusing on the theoretical aspects of NLP. On the other hand the open-access journal

"Computational Linguistics", styles itself as the longest running publication devoted exclusively to the design and analysis of natural language processing systems".

b. What do you understand by the term System Software?

Answer:
System software are general programs designed for performing tasks such as controlling all operations required to move data into and out of the computer. It communicates with printers card reader disk tapes etc. Monitor the use of various hardware like memory, CPU etc. Also system software are essential for the development of applications software.
System software allows application packages to be run on the computer with less time and effort.
It is not possible to run application software system software. Development of system software is a complex task and it requires extensive knowledge of computer technology. Due to its complexity it is not developed in house.

c. What are the various *language processing* activities in the domain of system software? What do you understand by *cross-compilation?*

Answer:
The language processing activities are
a) Program Generation Activities:
   Program generator is a software, which accepts the specification of a program to be generated; and produces a program in target language.
   Initially the semantic gap between source language domain and target language domain. But now with the program generation activities, the semantic gap exists between source language domain is close to source language domain, and it is easy for the designer/programmer to writ the specification of the program to be generated.
   This arrangement also reduces the testing effort. This is so because to test an application generated by the generator, it is necessary to only verify the correctness of specification that is input to the generator.

b) Program Execution Activities:

   The execution of program is segregated in two activities
   These two activities are:
   1) Program Translation Activities.
   2) Program Interpretation Activities.

Q7.   a. What is parsing? Write down the drawback of top down parsing of backtracking.

Answer:
Parsing is the process of analyzing a text, made of a sequence of tokens, to determine its grammatical structure with respect to a given formal grammar. Parsing is also known as syntactic analysis and parser is used for analyzing a text. The task of the parser is essentially to determine if and how the input can be derived from the start symbol of the grammar. The input is a valid input with respect to a given formal grammar if it can be derived from the start symbol of the grammar.

Following are drawbacks of top down parsing of backtracking:

(i)   Semantic actions cannot be performed while making a prediction. The actions must be delayed until the prediction is known to be a part of a successful parse.

(ii)   Precise error reporting is not possible. A mismatch merely triggers backtracking. A source string is known to be erroneous only after all predictions have failed.

b. Mention some advantages of assembly language over machine language.

Answer:
Assembly language is a symbolic representation of a processor's native code. Using machine code allows the programmer to control precisely what the processor does. It offers a great deal of power to use all of the features of the processor. The resulting program is normally very fast and very compact. In small programs it is also very predictable. Timings for example can be calculated very precisely and program flow is easily controlled. It is often used for small, real time applications.

However, the programmer needs to have a good understanding of the hardware being used. As programs become large, assembly language get very cumbersome. Maintenance of assembly language is notoriously difficult, especially if another programmer is brought in to carry out modifications after the code has been written. Assembly language also has no support of an operating system, nor does it have any complex instructions. Storing and retrieving data is a simple task with high level languages; assembly needs the whole process to be programmed step by step. Mathematical processes also have to be performed with binary addition and subtraction when using assembly language. Use a new processor and you need to learn a new language each time.

c. What are *assembler directives* in assembly languages? Illustrate with an example the importance of assembler directives.

Answer:
Assembler directives or pseudo-operations (pseudo-ops) are commands to the assembler directives are:

Section Control Directives
Symbol Attribute Directives
Assignment Directives
Data Generating Directives
Optimizer Directives

Q8.   a. What are the major stages in the process of *compilation*?

Answer:
Stages from Source to Executable
Compilation: source code = => relocatable object code (binaries)
Linking: many relocatable binaries (modules plus libraries) ==> one relocatable binary (with all external references satisfied)
Loading: relocatable ==> absolute binary (with all code and data references bound to the addresses occupied in memory)
Execution: control is transferred to the first instruction of the program
At compile time (CT), absolute addresses of variables and statement labels are not known.
In static languages (such as Fortran), absolute addresses are bound at load time (LT).
In block- structured languages, binding can change at run time (RT).
Phases of the Compilation process
 Syntactic analysis (parsing): tokens are combined to form syntactic structures, typically represented by a parse tree.
The parser may be replaced by a syntax- directed editor, which directly generates a parse tree as a product of editing.
Semantic analysis: intermediate code is generated for each syntactic structure.
Type checking is performed in this phase. Complicated features such as generic declarations and operator overloading are also processed.
Machine – independent optimization: intermediate code is optimized to improve efficiency.
Code generation: intermediate code is translated to relocatable object code for the target machine.
Machine – dependent optimization: the machine code is optimized.

b. Explain positional and keyword parameters used in lexical expansion.

Answer: Page Number 134 of Text Book

c. Define and explain program relocation.

Answer: Page Number 223 of Text Book

Q9.   **Write short notes on:**

   (i) A toy code generator for expressions

   (ii) Interpreter and their use

Answer: Page Number 180, 212 of Text Book

Text Book

Systems Programming & Operating Systems, D.M. Dhamdhere, Tata McGraw-Hill, II Revised Edition, 2005