Choose the correct answer from the given alternatives for each of the following. Explain your choice. One mark for correct answer, and one mark for the explanation.

1. Which of the following four models corresponds to a serial computer.
   (a) Single-Instruction Multiple data
   (b) Multiple-instruction multiple-data
   (c) Multiple-instruction single-data
   (d) Single-instruction single-data

   **Option D.** Single instruction multiple data, is a form of domain decomposition where the same instruction is executed on the various parts of the data. The different portions of the data are handled by the multiple processors available. Multiple instruction is various operations being performed on the same data. The different tasks can be allocated to different processors in order to parallelise the procedure. The same holds for multiple-instruction multiple-data.

2. Consider a program which uses persistent data-structures. Persistent data-structures are data-structures which change over time, but the program needs to have access to past versions as well. Assume various subroutines/functions of the program manipulate the data-structure. Which of the following is the appropriate way to pass this data-structure between various subroutines.
(a) By reference
(b) By copy
(c) One of the two methods chosen randomly at the time of passing.
(d) The method of passing is irrelevant.

**Option B.** Passing by copy results in the called function modifying only a copy of the data, while the version present with the calling function is unaltered. Thus several different versions can be maintained, which is impossible under passing by reference.

3. Consider a program for computing the minimum of a set of \( n \) numbers. Which is the appropriate model of parallelisation for this program?

(a) Task decomposition
(b) Domain decomposition
(c) Pipe lining
(d) A combination of two of the above.

**Option B.** Here, the domain consisting of the data of \( n \) numbers is split into smaller units, and the same task of finding the minimum is performed on each portion. These results again have a minimum finding algorithm applied to them. Thus domain decomposition is the appropriate model of parallelisation applicable to this problem.

4. Which of the following is true?

(a) If there is a set of cooperating user-level threads then the operating system is responsible for context-switches among them.
(b) If there is a set of kernel-level threads then when one of the threads makes a system call, execution of the other threads may continue.
(c) If one of a set of user-level threads makes a system call then the other threads in the group can continue execution.
(d) A group of cooperating threads can be a mix of system-level and user-level threads.

**Option B.** The kernel is aware of different kernel level threads within a task. The user-level thread structure within a task is abstracted away from the operating system, and it treats the entire task as a single block. Hence, it places the entire task in the waiting queue if any thread makes a system call.
5. Consider interprocess communication by message-passing. In this context what is the best description of a directed link connecting a set of three or more processes?

(a) There is exactly one transmitter process and all other processes are receivers.
(b) There is exactly one receiver process and all other processes are transmitters.
(c) Each process is either a transmitter or a receiver, and there is at least one process of each type associated with the link.
(d) Any process can transmit or receive messages via the link.

**Option C**. This generalises the directed link involving exactly two processes. The main feature there is that only one of the processes can transmit and only the other can receive. To maintain the directionality, in the generalised case, each process can only either transmit or receive. Also, for the link to function there must be at least one process of each type.

6. A file-map table (FMT) is used in any system where the memory management scheme is:

(a) Segmentation without virtual memory.
(b) Paging with virtual memory.
(c) Paging without virtual memory.
(d) Segmentation with virtual memory.

**Option B**. In a virtual memory environment, each process has only a certain active portion of its address space loaded into the main memory at any point of time. An image of the entire process is resident in secondary storage. Under paging scheme (contiguous allocation) with virtual memory a page a process requests may not be present in memory resulting in a page fault. The page map table (PMT) only has pointers to memory page frames containing loaded pages. In order to know the secondary storage address from which to load a missing page, the file map table (FMT) is used.

7. Consider the problem of finding the length of a shortest-path between each pair of vertices in a graph and then the largest value among these. The appropriate model of parallelisation for this program is:
(a) Domain decomposition  
(b) Task decomposition  
(c) Pipe lining  
(d) A combination of the above

**Option D.** The problem of finding the shortest path for each pair of vertices in a graph can be solved using domain decomposition and dynamic programming. The task of finding the maximum among the shortest paths between each pair can again be found using domain decomposition. The path lengths which are the output of the first part are passed as input to the maximum finding algorithm. This is a case of pipelining. It is thus a combination of domain decomposition and pipelining.

8. In a paged virtual memory system, the number of pages of a process loaded in memory remains unchanged under which of the following situations?

(a) All the page-frames in the main memory are full and the process faults, and there is a global replacement scheme in place.  
(b) All the page-frames in the main memory are full and the process does not fault.  
(c) There are some unused page-frames in the main memory and the process faults.  
(d) All the page-frames in main memory are full and the process faults, and there is a local replacement scheme in place.

**Options B,D.** When no page-fault occurs, then clearly the number of pages of any process (and hence this particular process) remains unaltered. This is the reason for option B. When a page fault occurs and all page frames in memory are full and there is a local replacement scheme in place, it means all the page frames allotted to this process are full and one of them will be swapped out to make way for the page being demanded. The number of pages of the faulting process thus remains unchanged. This is the reason for option D.

9. The main advantage of parallel computing is:

(a) To reduce chances of error in the program output as more computers/processors are working on the program.
(b) To reduce the end-to-end running time of the program.
(c) To reduce the total computation time; the total being taken over all the working processors working on the program.
(d) To guarantee that the program always terminates and does not get stuck in infinite loops.

**Option B.** The total time summed over all parallel processors actually increases, if we take into account the overhead of parallelisation. Thus the aim of parallelising is NOT to reduce the total computation in this sense, but to reduce the end-to-end (turnaround) time or the *wall-clock time*.

10. Which is the allocation algorithm which leads to the least memory wastage (unusability) due to external fragmentation in a segmented memory management scheme? Answer this question assuming any possible sequence of process loads (memory allocation) and terminations/swap-outs (memory frees).

(a) First-fit
(b) Best-fit
(c) Worst-fit
(d) It depends on the specific sequence.

**Option D.** It depends on the specific sequence. Notice that the first-fit is clearly not a universal option, since the best-fit or worst-fit might very well be the first to occur. And among these two, neither can be most successful in all sequences. There is a variation from sequence to sequence.

**Section B**

(2 X 10=20 marks)

11. In a contiguous allocation memory management scheme, when there is enough free space (in total) but not as a segment, then the system sometimes performs compaction, which is movement of segments together to combine all the available free-space. This is an expensive operation and its frequency should be minimised. Consider a system with 1 MB (1024 KB) of main memory available for user processes. The following are the memory requirements for a set of processes which request to be loaded and moved out in the indicated sequence.

\[ P_1 = 109KB \text{ IN}, \ P_2 = 222KB \text{ IN}, \ P_3 = 266KB \text{ IN}, \ P_4 = 111KB \text{ IN}, \]
\[ P_5 = 125KB \text{ IN}, \quad P_6 = 120KB \text{ IN}, \quad P_1 \text{ OUT}, \quad P_4 \text{ OUT}, \quad P_7 = 70KB \text{ IN}, \]
\[ P_8 = 100KB \text{ IN}, \quad P_9 = 100KB \text{ IN}. \]
Sort the schemes first-fit, best-fit and worst-fit in the order in which compaction becomes necessary under this sequence of requests. Assume same sequence upto \( P_4 \) OUT, and generate a sequence of requests (and releases if you need) such that compaction is needed under best-fit earlier than it is needed under worst-fit.

**Solution** After the first set of load operations the memory is as follows.

- 1 – 109: Process \( P_1 \).
- 110 – 331: Process \( P_2 \).
- 332 – 597: Process \( P_3 \).
- 598 – 708: Process \( P_4 \).
- 709 – 833: Process \( P_5 \).
- 834 – 953: Process \( P_6 \).
- 954 – 1024: Free space.

The memory configuration after the next two operations, which are both process removal from memory, is as follows.

- 110 – 331: Process \( P_2 \).
- 332 – 597: Process \( P_3 \).
- 598 – 708: Free space (111 KB).
- 709 – 833: Process \( P_5 \).
- 834 – 953: Process \( P_6 \).
- 954 – 1024: Free space (71 KB).

The next three operations are process loads of sizes 70 KB, 100 KB and 100 KB respectively.

- **First-fit.** This places the process the process \( P_7 \) in the first free slot (1-70), the process \( P_8 \) in the second free slot (598-697). Notice that the remaining slot (71-109) is insufficient for the process \( P_8 \). The remaining free slots are insufficient for process \( P_9 \), and hence compaction must take place before \( P_9 \) is loaded.
• **Best-fit.** This places the process $P_7$ in the last free slot (954-1023), the process $P_8$ in the first free slot (1-100) and the process $P_9$, in the second free slot (598-697). Thus all processes can be loaded without any compaction needed.

• **Worst-fit.** This places the process $P_7$ in the second free slot (598-667), $P_8$ in the first free slot (1-100) and there is insufficient contiguous block space to load process $P_9$. Thus compaction needs to be performed before $P_9$ can be loaded.

Thus, both first-fit and worst-fit require compaction before $P_9$ can be loaded, whereas under bedt-fit scheme, all processes can be loaded without compaction. This is thus the order of the schemes according to earliest requirement of compaction for this sequence of memory requests.

The request sequence of loading 4 processes of size 40 KB, 71 KB, 71 KB, 71 KB, would be serviced uninterrupted, by a worst-fit algorithm, whereas a best-fit algorithm would need compaction after the third request. This is assuming the sequence upto $P_4$ OUT in the sequence given in the question. Any other valid sequence will, of course, get full credit. Otherwise zero.

Please note, that I am using a breakup of 5 marks each for the first and second part of the question. This breakup is indicated by a horizontal line separating them in the solution. This breakup is based on the amount of thinking that might go into the second part being more, even though the writing is less.

12. Describe the page-map tables (PMTs), and file-map tables (FMTs) for systems with virtual memory and give an estimate on the amount of storage needed. Describe the two registers which can be used to reduce the total memory needed to store this information over all the active processes. Give a detailed description of the address translation process and the subsequent access to the physical address on the basis of the virtual/logical address.

**Solution** Paging is a scheme used in noncontiguous management of main memory. Here the memory is divided into fixed size units called page-frames. The address space associated with processes are also divided into fixed size chunks called pages. In a virtual memory environment where paging is used, an image of the entire address space of a process is present in secondary storage and this image is divided into
fixed size units called pages. In a virtual memory environment, the set of pages of a process loaded in main memory at any time is typically less than its actual number of pages. This is in order to enable greater use of the main memory to load more processes and thereby increase the degree of multiprogramming and the system performance. When a process needs one of its pages not currently resident in memory, it is retrieved from the image in secondary storage, if necessary, by first freeing a page-frame in main memory, by evicting a page of that or another process. This is known as a fault.

The page map table (PMT) is used to keep track of the sequence of logical pages of a process loaded in memory with regard to the page-frame numbers in which they reside. There are as many entries the PMT of a process as the number of page-frames it consumes, each entry corresponding to the location of one of its logical pages. The size of the individual portions of the PMT corresponding to a particular process is maintained using two registers. One is the page map table limit register (PMTLR) and indicates the number of page frames allocated to that process. The page map table base register (PMTBR) indicates the base address of the place where the PMT of that process begins. This information together specifies precisely, the PMT of that process in the overall PMT.

When a process requests a page, it may or may not be present in main memory in a virtual memory environment. This presence of absence information is provided by the file map table (FMT). It contains information on whether the page is present, and if not it contains a pointer to the location in secondary storage where the corresponding page resides within the image of the process.

The address translation process consists of figuring the logical page number and word of the process within its logical/virtual address space. The next step is the word number within that page. The presence or absence of that page in the memory is determined using FMT. The PMT indicates the page-frame number, and the multiplication by page-frame size and an offset for the word number within the page yields the exact physical address being sought.

This the lines along which I expect the solution. It can of course be significantly shorter and concise, but this is the information required for full marks. The precise breakup is description of paging memory management (3 marks); PMT
and the registers to reduce the total size consumed by them (2 marks); FMT and faults (3 marks); address translation (2 marks). This is the template on the basis of which marking is being done. Anyone arguing wrongly over marks will get -10; so think carefully before asking for reevaluation.