Name: Sayma Akter Shumi **ID: IT-18032**

1. a) What is deadlock? Write down the conditions of deadlock? 2+4

b) Explain the deadlock detection? 4

c) Write down deadlock avoidance in banker's algorithm? 4

2.a) What is banker's algorithm? Why banker's algorithm is used? 2+2

b) How can we prevent the occurrence of a deadlock occurs? 5

c) How can recovery from deadlock state? 5

3. a) Draw the multistep processing of a user program? 5

b) Differentiate between logical & physical address space? 5

c) Differentiate between static & dynamic linking? 4

4. a) What is dynamic linking? Why we used dynamic linking in OS? 2+2

b) Explain advantages & disadvantages of dynamic linking? 6

c) What is swapping? Write down the purpose of swapping? 2+2

5. a) What is memory protection? Differentiate between MMU & MPU? 2+4

b) Write down advantages & disadvantages of demand paging? 4

c) What are the aspects of demand paging? 4

6. a) Draw the schematic view of swapping? 5

b) What is ARM architecture? Why we used ARM architecture? 2+2

c) Differentiate between ARM & x86 architecture? 5

7. a) What is thrashing? Write down the causes of thrashing? 2+4

b) Write is page fault? How we can handle page fault? 2+4

c) Define LRU algorithm? 2

8. a) What is file system? Mention all objectives of file system? 2+3

b) What is file sharing? What is importance of file sharing? 2+3

c) What is file structure? Write down properties of file system? 2+2

Answer to the question number (1)

a)

Deadlock:

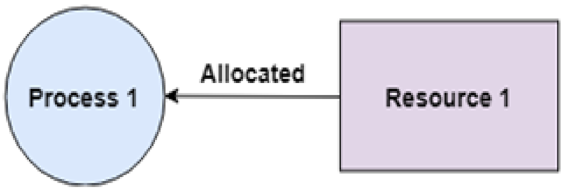
Deadlock is a situation where the execution of two or more processes is blocked because each process holds some resource and waits for another resource held by some other process.

Conditions of deadlock:

A deadlock occurs if the four Coffman conditions hold true. But these conditions are not mutually exclusive. They are given as follows −

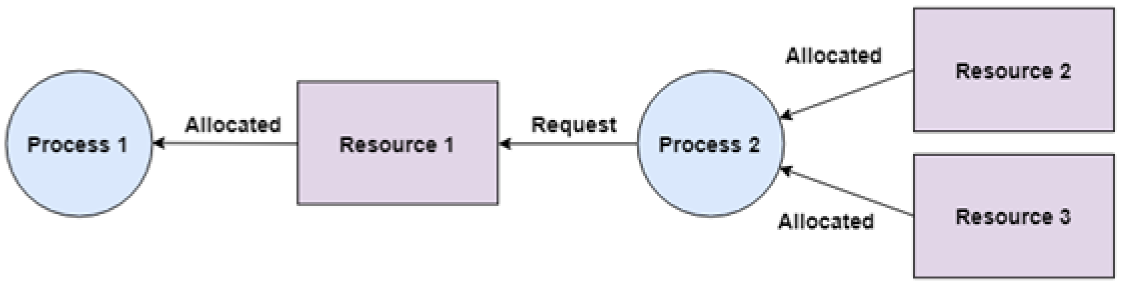
Mutual Exclusion:

There should be a resource that can only be held by one process at a time. In the diagram below, there is a single instance of Resource 1 and it is held by Process 1 only.



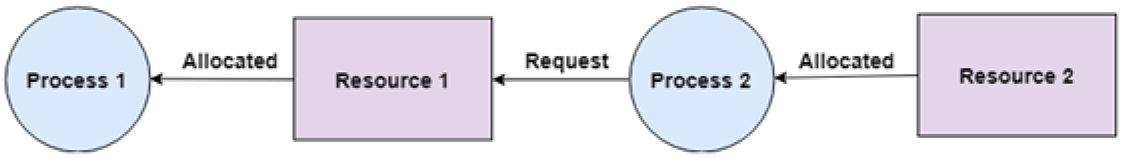
Hold and Wait:

A process can hold multiple resources and still request more resources from other processes which are holding them. In the diagram given below, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.



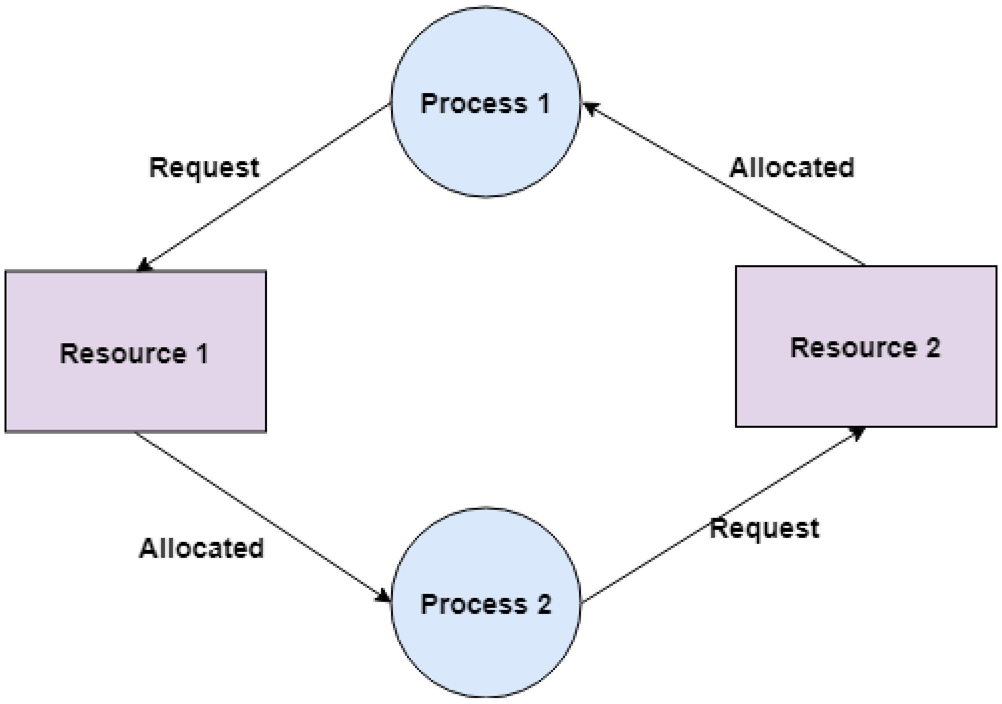
No Preemption:

A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In the diagram below, Process 2 cannot preempt Resource 1 from Process 1. It will only be released when Process 1 relinquishes it voluntarily after its execution is complete.



Circular Wait:

A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example: Process 1 is allocated Resource2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.



b)

<u>Deadlock detection</u>:

If a system does not employ either a deadlock-prevention or a deadlock avoidance algorithm then a deadlock situation may occur. The system must provide:

1. An algorithm that examines the state of the system to determine whether a deadlock has occurred.
2. An algorithm to recover from the deadlock.

A detection and recovery will incur a considerable overhead in computation time that includes:

1) Run time cost of maintaining the necessary information.
2) Executing the detection algorithm.
3) The potential losses inherent in recovery from a deadlock.

c)

<u>Deadlock avoidance</u>:

Deadlock avoidance in banker's algorithm-

- In a deadlock state, the execution of multiple processes is blocked.
- Deadlock avoidance strategy involves maintaining a set of data.
- The data is used to make a decision whether to entertain any new request or not.
- If entertaining the new request causes the system to move in an unsafe state, then it is discarded.

<u>**Answer to the question number (2)**</u>

a)

<u>Banker's Algorithm</u>:

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

Banker's Algorithm is used majorly in the banking system to avoid deadlock. It helps you to identify whether a loan will be given or not. This algorithm is used to test for safely simulating the allocation for determining the maximum amount available for all resources.

b)

By ensuring that at least one of deadlock conditions cannot hold.
As follow:

1. Mutual Exclusion: The mutual-exclusion condition must hold for non-sharable resources.

2. Hold and wait: we must guarantee that, whenever a process requests a resource, it does not hold any other resources.

3. No Preemption: we can use the following protocol. If a process is holding some resources and requests another resource that cannot be immediately allocated to it (the process must wait), then all resources currently being held are preempted.

4. Circular Wait: By impose a total ordering of all resource types and to require that each process requests resources in an increasing order of enumeration.

c)

When a deadlock exists, several alternatives are available.

1. One possibility is to inform the operator that a deadlock has occurred and to let the operator deal with the deadlock manually.

2. The system recovers from the deadlock automatically. There are two options for breaking a deadlock. A) Process Termination: Aborting processes to eliminate the deadlock. There are two methods:
1)Abort all deadlocked processes.
2) Abort one process at a time until the deadlock cycle is eliminated. B) Resource Preemption: preempt: some resources from processes and give these resources to other processes until the deadlock cycle is broken. three issues need to be addressed:

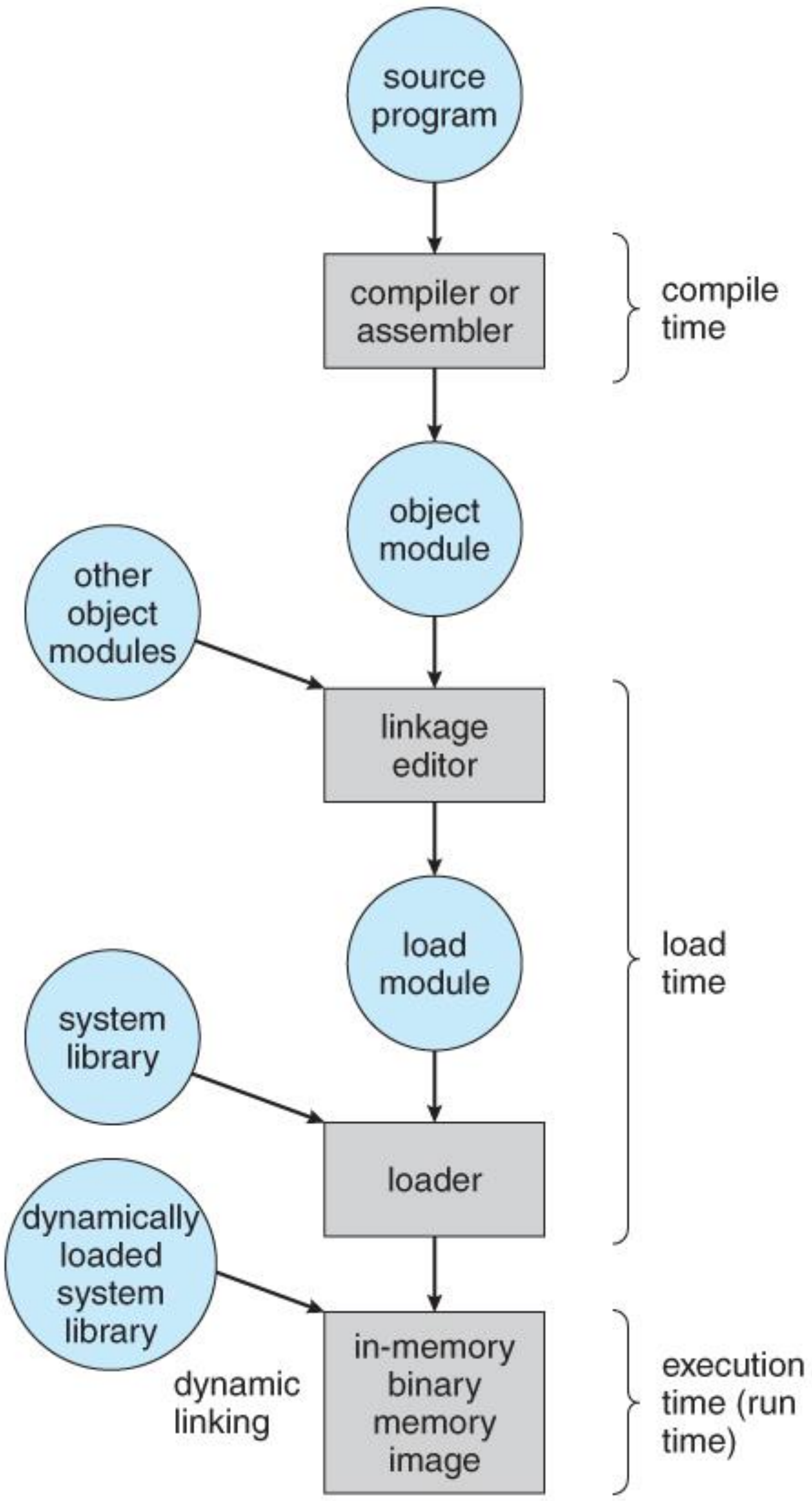
1) Selecting a victim. Which resources and which processes are to be preempted?
2) Rollback. If we preempt a resource from a process, what should be done with that process?

3) Starvation. How do we ensure that starvation will not occur, guarantee that resources will not always be preempted from the same process?

Answer to the question number (3)

a)

The multistep processing of a user program-



b)

The difference between logical and physical address space-

1. Logical address is the address that is generated by the central processing unit (CPU) in perspective of a program. Logical address can also be referred to as a virtual address. Physical address on the other hand is a location that exists in the memory; it allows accessing a particular storage cell in the main memory.

2. Logical address space is the set of all logical addresses generated by CPU for a program whereas physical address space is the set of all physical address mapped to corresponding logical addresses.

3. The physical address is an accessible physical location existing within the memory whereas the logical address exist virtually and does not have a specific location to exist physically in memory unit hence it is also known as virtual address.

4. The logical address is generated by the central processing unit (CPU) whereas physical address is computed by Memory Management Unit (MMU). MMU is a hardware device that maps virtual to physical address.

5. Physical and logical addresses are same in compile-time and load-time addressing-binding schemes. The two only differ in the execution-time address-binding scheme.

6. The physical address helps to identify a location in the main memory whereas the logical address helps to obtain the physical address.

c)

Difference between static & dynamic linking:

1. In static linking, all the library modules are copied to the final executable image. When the program is loaded, OS places only a single file to the memory which contain both the source code and the referencing libraries. Whereas in dynamic linking only the names of external or shared libraries is placed into the memory. Dynamic linking lets many programs use single copy of executable module.
2. Static linking is done by the linkers in the final step of the compilation whereas the dynamic linking is done at run time by the OS.
3. Statically linked files consume more disk and memory as all the modules are already linked. But in Dynamic linking**,** only one copy of the reference module is stored which is used by many programs thereby saving memory and disk space.
4. In Static linking**,** if external source program is changed then they have to be recompiled and relinked. But in case of dynamic linking only a single module needs to be updated and recompiled.
5. Statically linked programs are faster than their dynamic counterpart.
6. Since the statically linked file contains every package and module, no compatibility issues occur. Whereas in dynamic linking, since the library files are separately stored there may be compatibility issues (say one library file is compiled by new version of compiler).
7. Statically linked programs always take constant load time whereas the time is variable in dynamically linked programs.

Answer to the question number (4)

a)

Dynamic linking: Dynamic linking consists of compiling and linking code into a form that is loadable by programs at run time as well as link time. The ability to load them at run time is what distinguishes them from ordinary object files. Various operating systems have different names for such loadable code: UNIX: Sharable Libraries.

Every program generated must contain copies of exactly the same common system **library** functions. In terms of both physical memory and disk-space usage, it is much more efficient to load the system libraries into memory only once. **Dynamic linking** allows this single loading to happen.

b)

Advantages of dynamic linking:

- Load time might be reduced because the shared library code might already be in memory.
- Run-time performance can be enhanced because the operating system is less likely to page out shared library code that is being used by several applications, or copies of an application, rather than code that is only being used by a single application. As a result, fewer page faults occur.
- The routines are not statically bound to the application but are dynamically bound when the application is loaded. This permits applications to automatically inherit changes to the shared libraries, without recompiling or rebinding.

Disadvantages of dynamic linking include the following:

- From a performance viewpoint, there is "glue code" that is required in the executable program to access the shared segment. There is a performance cost in references to shared library routines of about eight machine cycles per reference. Programs that use shared libraries are usually slower than those that use statically-linked libraries.
- A more subtle effect is a reduction in "locality of reference." You may be interested in only a few of the routines in a library, and these routines may be scattered widely in the virtual address space of the library. Thus, the total number of pages you need to touch to access all of your routines is significantly higher than if these routines were all bound directly into your executable program. One impact of this situation is that, if you are the only user of these routines, you experience more page faults to get them all into real memory. In addition, because more pages are touched, there is a greater likelihood of causing an instruction translation lookaside buffer (TLB) miss.
- When a program references a limited number of procedures in a library, each page of the library that contains a referenced procedure must be individually paged into real memory. If the procedures are small enough that using static linking might have linked several procedures that are in different library pages into a single page, then dynamic linking may increase paging thus decreasing performance.
- Dynamically linked programs are dependent on having a compatible library. If a library is changed (for example, a new compiler release may change a library), applications might have to be reworked to be made compatible with the new version of the library. If a library is removed from the system, programs using that library will no longer work.

c)

Swapping:

Swapping is a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes. It is used to improve main memory utilization.

Purpose of swapping:

The purpose of the swapping in operating system is to access the data present in the hard disk and bring it to RAM so that the application programs can use it. The thing to remember is that swapping is used only when data is not present in RAM.

Answer to the question number (5)

a)

Memory protection:

Memory protection is a way to control memory access rights on a computer, and is a part of most modern instruction set architectures and operating systems. This prevents a bug or malware within a process from affecting other processes, or the operating system itself.

Difference between Memory Management Unit & Memory Protection Unit:

- Memory is one important component in modern computing. As such, it is necessary that its contents are not corrupted by any errant application. This function can be done by an MMU (Memory Management Unit) or by an MPU (Memory Protection Unit). Although they both do the same basic function, there are a number of differences between an MMU and an MPU. An MMU is considered to be a more advanced device than an MPU. An MMU is capable of doing the job of an MPU along with other more advanced features that are absent in the latter.

- The features that are present in an MMU and not in the MPU include cache control, bus arbitration, and bank switching. All these features are necessary in more complex computers as they allow the flow of information to be smoother and without any problem. By using an MMU, you can also optimize the performance of your computer as it would off-load the said tasks from the microprocessor.

- But if you are dealing with a very simple computer that does not do multi-tasking and other related processes, using an MMU may not be the best thing to do. The primary consideration would be price and complexity as employing the more advanced MMU would be more costly and more complex. It may also tax the system unnecessarily as MMU units utilize more overhead compared to the far simpler MPUs. If the main processor used is not fast enough, using an MMU may cause problems.

- Choosing between an MMU and MPU is largely dependent on the scale and complexity of the system being built. An MMU is recommended if your system is large enough to benefit from its capabilities. If you are building a relatively simply system, using an MMU is not advisable. An MPU would provide the necessary capabilities while keeping complexity and cost down.

b)

Advantages of demand paging:

Demand paging, as opposed to loading all pages immediately:

- Only loads pages that are demanded by the executing process.
- As there is more space in main memory, more processes can be loaded, reducing the context switching time, which utilizes large amounts of resources.
- Less loading latency occurs at program startup, as less information is accessed from secondary storage and less information is brought into main memory.
- As main memory is expensive compared to secondary memory, this technique helps significantly reduce the bill of material (BOM) cost in smart phones for example. Symbian OS had this feature.

Disadvantages of demand paging:

- Individual programs face extra latency when they access a page for the first time.
- Low-cost, low-power embedded systems may not have a MMU that supports page replacement.
- Memory management with page replacement algorithms becomes slightly more complex.
- Possible security risks, including vulnerability to timing attacks.
- Thrashing which may occur due to repeated page faults.

c)

Aspects of Demand Paging-

1. Extreme case – start process with no pages in memory
 · OS sets instruction pointer to first instruction of process, non- memory-resident -> page fault
 · And for every other process pages on first access
 · Pure demand paging .
2. Actually, a given instruction could access multiple pages -> multiple page faults
 · Consider fetch and decode of instruction which adds 2 numbers from memory and stores result back to memory

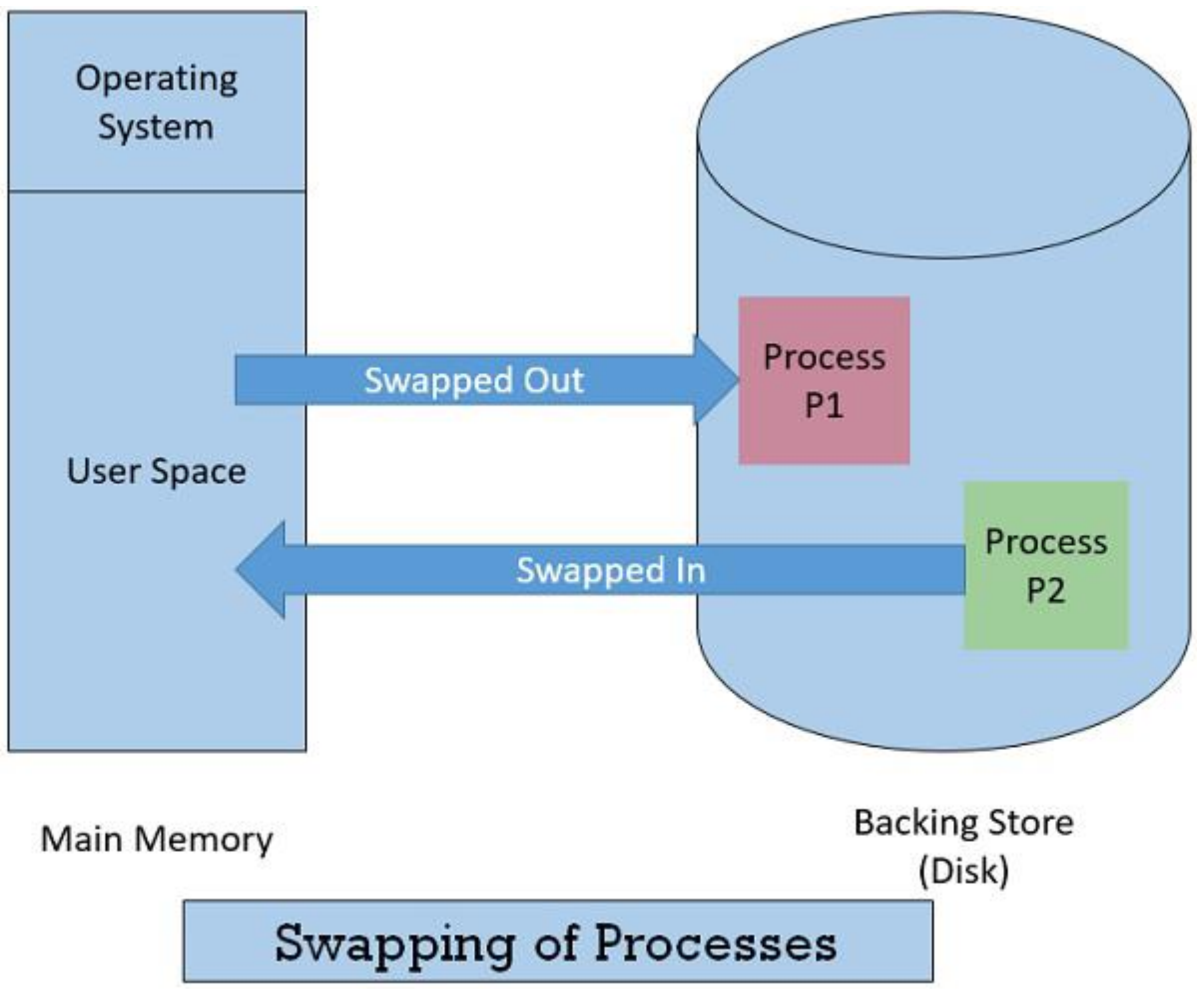
 · Pain decreased because of locality of reference

3. Hardware support needed for demand paging
 · Page table with valid / invalid bit
 · Secondary memory (swap device with swap space)
 · Instruction restart.

<u>**Answer to the question number (6)**</u>

a)

<u>**Schematic view of swapping**</u>:



b)

ARM architecture:

ARM (stylized in lowercase as arm, previously an acronym for Advanced RISC Machines and originally Acorn RISC Machine) is a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments.

ARM processors are extensively used in consumer electronic devices such as smartphones, tablets, multimedia players and other mobile devices, such as wearables. Because of their reduced instruction set, they require fewer transistors, which enables a smaller die size for the integrated circuitry (IC).

c)

Difference between ARM & x86 architecture:

1.Performance

In any case, computers with X86 structure are much faster and stronger than ARM structure systems in terms of performance. X86 CPU is more than 1g, dual-core, quad-core is popular, usually using 45nm (or even more advanced) process for production; While ARM: CPU is usually several hundred megabytes, recently, about 1g

CPU has just appeared, and the process usually uses less than 65nm Process. It can be said that ARM is not the rival of X86 structure system in terms of performance and production process.

However, the advantage of ARM lies not in its powerful performance but in its efficiency. ARM Adopts RISC pipeline instruction set, which is fundamentally at a disadvantage in completing comprehensive work, however, in some applications with relatively fixed tasks, its advantages can be fully exerted.

2. Capacity expansion

X86-structured computers are connected to expansion devices (such as hard disks and memory) in a “bridge” way. Moreover, x86-structured computers have appeared for nearly 30 years, there are many kinds of supporting extended devices and the price is relatively low, so x86 computers can easily expand their performance, such as adding memory and hard disks.

ARM-structured computer connects CPU with data storage device through dedicated data interface, so ARM’s storage, performance expansion such as memory is difficult to carry out (generally, the memory and data storage capacity have been determined during product design). Therefore, expansion is generally not considered for systems with ARM structure. Basically follow the principle of “enough is good.

3. Compatibility of the operating system

The X86 system is dominated by the Wintel alliance built by Microsoft and Intel. It has monopolized the personal computer operating system for nearly 30 years, forming a huge user group and deeply solidifying the usage habits of many users, at the same time, x86 system has formed a unified standard in terms of hardware and software development. Almost all x86 hardware platforms can directly use Microsoft Windows system and almost all popular tools and software, therefore, x86 systems have incomparable advantages in compatibility.

Almost all ARM systems adopt Linux operating systems, and almost all hardware systems have to build their own systems separately, which are incompatible with other systems, resulting in the inconvenient transplantation of their application software, these 1.1 straight severely restrict the development and application of ARM system. After GOOGLE developed the open Android system, it unified the operating system of computer with ARM structure, which enabled the newly launched computer system based on ARM structure to have a unified, open and free operating system, it provides strong support and impetus for the development of ARM.

4. Convenience of software development and diversity of available tools

It has been nearly 30 years since the X86 structure system was launched. During this period, x86 computers have gone through a golden period of rapid development. Users’ applications, software matching, software development tools matching and compatibility, etc, it has reached a very mature or even perfect state. Therefore, using X86 computer systems not only has a large number of third-party software to choose from, but also has a large number of software programming tools to help you complete the work you want to complete.

Answer to the question number (7)

a)

Thrashing:

In computer science, thrashing occurs when a computer's virtual memory resources are overused, leading to a constant state of paging and page faults, inhibiting most application-level processing. This causes the performance of the computer to degrade or collapse. This condition is referred to as thrashing.

Causes of Thrashing :

1. High degree of multiprogramming : If the number of processes keeps on increasing in the memory than number of frames allocated to each process will be

decreased. So, less number of frames will be available to each process. Due to this, page fault will occur more frequently and more CPU time will be wasted in just swapping in and out of pages and the utilization will keep on decreasing.
For example:
Let free frames = 400
Case 1: Number of process = 100
Then, each process will get 4 frames.
Case 2: Number of process = 400
Each process will get 1 frame.
Case 2 is a condition of thrashing, as the number of processes are increased, frames per process are decreased. Hence CPU time will be consumed in just swapping pages.

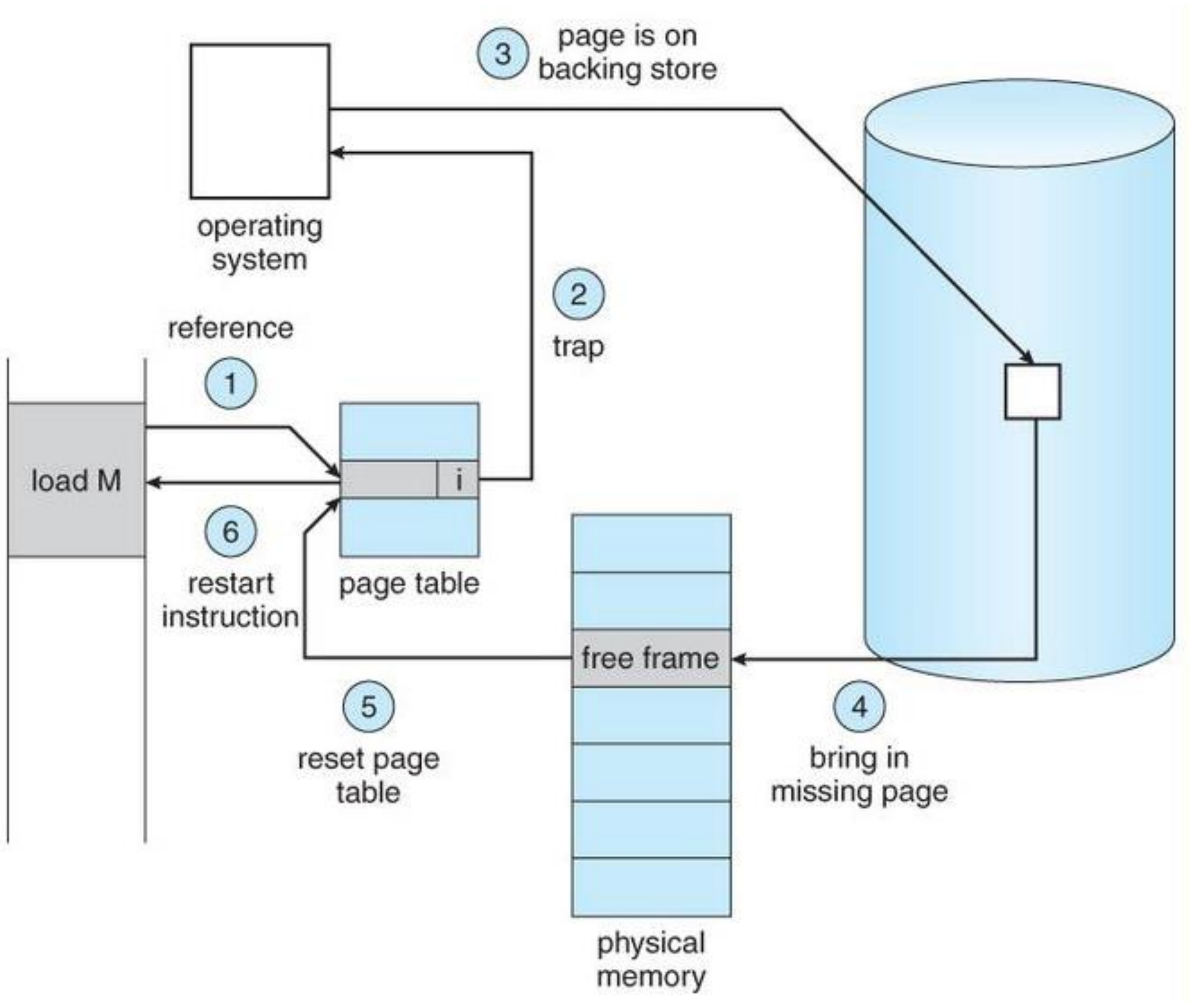
2. Lacks of Frames: If a process has less number of frames then less pages of that process will be able to reside in memory and hence more frequent swapping in and out will be required. This may lead to thrashing. Hence sufficient amount of frames must be allocated to each process in order to prevent thrashing.

b)

Page fault:

A page fault occurs when a program attempts to access a block of memory that is not stored in the physical memory, or RAM. The fault notifies the operating system that it must locate the data in virtual memory, then transfer it from the storage device, such as an HDD or SSD, to the system RAM.

Steps for handling page fault:



- The basic idea behind paging is that when a process is swapped in, the pager only loads into memory those pages that it expects the process to need (right away.)
- Pages that are not loaded into memory are marked as invalid in the page table, using the invalid bit. (The rest of the page table entry may either be blank or contain information about where to find the swapped-out page on the hard drive.)
- If the process only ever accesses pages that are loaded in memory (memory resident pages), then the process runs exactly as if all the pages were loaded in to memory.
- On the other hand, if a page is needed that was not originally loaded up, then a page fault trap is generated, which must be handled in a series of steps:
    1. The memory address requested is first checked, to make sure it was a valid memory request.
    2. If the reference was invalid, the process is terminated. Otherwise, the page must be paged in.
    3. A free frame is located, possibly from a free-frame list.
    4. A disk operation is scheduled to bring in the necessary page from disk. (This will usually block the process on an I/O wait, allowing some other process to use the CPU in the meantime.)
    5. When the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to indicate that this is now a valid page reference.
    6. The instruction that caused the page fault must now be restarted from the beginning, (as soon as this process gets another turn on the CPU.)
- In an extreme case, NO pages are swapped in for a process until they are requested by page faults. This is known as pure demand paging**.**
- In theory each instruction could generate multiple page faults. In practice this is very rare, due to locality of reference, covered in section 9.6.1.
- The hardware necessary to support virtual memory is the same as for paging and swapping: A page table and secondary memory. (Swap space, whose allocation is discussed in chapter 12.)
- A crucial part of the process is that the instruction must be restarted from scratch once the desired page has been made available in memory. For most simple instructions this is not a major difficulty. However there are some architectures that allow a single instruction to modify a fairly large block of data, (which may span a page boundary), and if some of the data gets modified before the page fault occurs, this could cause problems. One

solution is to access both ends of the block before executing the instruction, guaranteeing that the necessary pages get paged in before the instruction begins.

c)

LRU algorithm:

In Least Recently Used (LRU) algorithm is a Greedy algorithm where the page to be replaced is least recently used. The idea is based on locality of reference, the least recently used page is not likely.

Answer to the question number (8)

a)

File system:

A file is a collection of correlated information which is recorded on secondary or non-volatile storage like magnetic disks, optical disks, and tapes. It is a method of data collection that is used as a medium for giving input and receiving output from that program.

Objective of file system:

- It provides I/O support for a variety of storage device types.
- Minimizes the chances of lost or destroyed data
- Helps OS to standardized I/O interface routines for user processes.
- It provides I/O support for multiple users in a multiuser systems environment.

b)

File sharing:

File sharing is the public or private sharing of computer data or space in a network with various levels of access privilege. File sharing can also mean having an allocated amount of personal file storage in a common file system.

Importance of file sharing:

1. The purpose and main importance of file sharing is to give copies of digital files, information and media with ease and speed to all concern users.

2. File-sharing can be easily executed using file-sharing software or programs and websites like: Google Drive, One Drive, and Dropbox.

c)

File structure:

A File Structure needs to be predefined format in such a way that an operating system understands . It has an exclusively defined structure, which is based on its type.

Properties of file system:

Here, are important properties of a file system:

- Files are stored on disk or other storage and do not disappear when a user logs off.
- Files have names and are associated with access permission that permits controlled sharing.
- Files could be arranged or more complex structures to reflect the relationship between them.

End