Operating Systems

B.Tech. CSE

Gurmukh Singh

Instructor:
Mr. Amit Chauhan

Contents

1 Operating systems

<u>*Def^n* :</u>
Operating system is an interface between user and the hardware. Without operating system the user cannot access any hardware.

<u>*Def^n* :</u>
When we have many programs running at one given time, we need to allocate CPU to one of them and memory with osem other application. This is known as resource sharing. Sometimes resources are not sharable so we allocate some process or program and later after taking it back we can allocate it to some other process.

1. Operating system acts like a manager. It is a kind of book keeping like which resource is currently allocated to which process.
2. memory can be managed like :
    - how much memory do we have?
    - how much memory we can allocate?
    - how much free memocy is available?

1.1 Goals of operating systems:

1. Primary Goal : Convenience (personal computers)
2. Secondary Goal: Efficiency (Supercomputers)

1.2 Types of operating systems:

1. Batch OS
2. Multiprogramming OS
3. Multitasking Os
4. Multiprocessing OS
5. Real time OS

1.2.1 Batch OS

There used to be a single computer to which everyone was given access. This computer is called mainframe and everyone should give a program to this mainframe computer in a queue. Then the comuter will pick them one after the other and execute them and later at some point the user will pick the program.

If any of the jobs require less amount of CPU time and more I/O time since here CPU time is greater than I/O time still CPU is not allocated to any other job until the current process is completed. It is not interactive as well which means thatresponse is not given immediately

1.2.2 Multiprogramming OS

It is basically an extension to Batch processing using multiprogramming CPU will be used efficiently. e.g. suppose the first job requires I/O then it will go to I/O device then CPU will be allocated to second job at the same time in such a way that CPU remains busy as longer as the are jobs.

Advantages of Multiprogramming OS:

1. We need not to keep CPU ideal.
2. CPU will be busy all the time with maximum utilization

1.2.3 Multitasking OS

It is an extension to multiprogramming OS. CPU will be multiplexing among all the jobs without completing any first job. Like for some time CPU is allocated to job1 then to job2 then to job3 and so on. And repeat this cycle again and again.

e.g. round robin approach.

1.2.4 Multiprocessing OS

In this instead of having one CPU we will be having lots of CPU's in the same computer.

Advantages:

1. Many jobs can be run simultaneously and parallelism can be achieved.
2. Throughput can be improved. (number of jobs completed per unit time.)
3. if one CPU fails, jobs can be rescheduled between other CPU's

1.2.5 Real time OS

In this scenario, we are given some jobs and jobs will be having some deadlines. then we are supposed to finish those jobs in the assigned deadline.
In this scenario timing is everything.

2 Process management

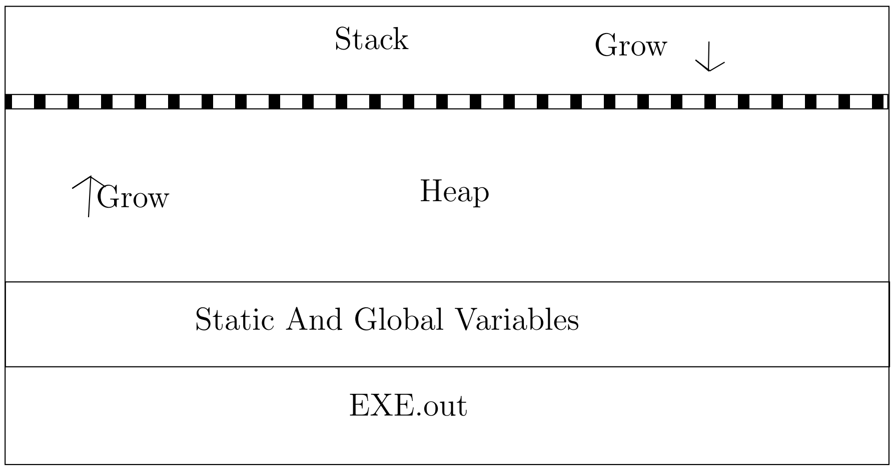
suppose we write a file "exa.c" in C language. now the program has to be first converted from high level language to low level language or machine language. This process is done by a compiler.

Program generally resides in the secondary memory. Operating system will take it from secondary memory and place it in main memory. Then start the execution. When the OS puts the program in main memory, it creates a datastructure called a process.

2.1 Process

It is something which is created by the operating sysetem in order to execute a program. So before we execute a program we need to create a process. The proces is real and the program is virtual.

For example: Making of the food is the process and the recipie is the program


Stack
Grow
Grow
Heap
Static And Global Variables
EXE.out

Once the Datastructure is created, the OS will start executing the executable code line by line.

2.1.1 Attributes of a process

1. Process ID
2. Program counter
3. Process state
4. Priority
5. General purpose register
6. List of open files
7. List of open devices
8. Protection

Process ID:
Each process is given a unique number called the process ID. the number of bits depends on the OS.

Program counter:
While we are executing a process, we suddenly start a process and then we restart it at the end of execution.

Process state:
This is the current state of the process.

Priority:
Whenever a process is created by an OS, it is assigned some priority. So a process with high priority will be executed first by the CPU.

GPR:
When a process is prempted and some other process comes for execution then the contents of the registers related to the prempted process should also be preserved so that when it resumes it's execution consistency will be maintained.

List of open files:
During the execution of a process some files need to be opened , some of the files are opened for reading purpose and others for writing purpose.

List of oped devices:
Devices which are currently used by the process

Protection:
One process should not go into other process workspace and vice versa. For every process OS is going to create a process control block (PCB) which stores all the above information about a process.

2.2 States of a process

A process from it's creation to completion. Depending upon the requirements a process will transition from one state to another. When we have a lot of processes that are ready to rum is known as multiprogramming.

States of a process

1. New: Whenever a process is created It is in the new state.
2. Ready: whenever processes are created they are ready to run in the main memory. There are various processes which are ready to run In multiprogramming there are two types of processes:
   (a) With premption
   (AKA multitasking or time sharing) With premption the process is forcefully stopped
   (b) Without premption
   Without premption the process is allowed to complete it's execution, it is not forced to stop.
3. Run
   simultaneously only process will be running if we have one CPU and if we have n CPU's then n processes will be running simultaneously.
4. Block/wait
   Sometimes a process which is running might need some resources like a file or access to a device then we pull it out and put it in block state or wait state until it finishes with I/O. Once it finishes the I/O we again bring it to previous state i.e. ready state.
5. Termination/completion
   Once a process finishes or completes it's execution it is known to be in completion stage or termination stage. Once a process completes it's execution it's PCB(Process control block) is also deleted.
6. Suspend ready
   Whenever space in main memory is not sufficient and whenever we want to accomodate more important processes(high priority processes) we just move the low priority processes out of the main memory to the secondary memory. These are the set of processes which work initially in the ready state and now because of lack of resources we are throwing them outside into the secondary memory.
7. Suspend wait/ suspend block
   performing processes are performing some I/O and are waiting for initialization so we let them wait in secondary memory or move to wait or block state of secondary memory.

2.3 Generation of OS:

1. First generation: (1940-1948)
   large and expensive.
2. Second Generation: (1956-1963)
   transistors replaced vaccuum tubes
   generated a lot of heat.
3. Third Generation(1964-1971)
   Introduced IC.
   transistors were placed on silicon chips
   keyboards and monitors became widespread.
4. Fourth Generation(1971-present)
   Microprocessors, GUI's, mouse and handheld device.
5. Fifth Generation(Present and beyond)
   Based on AI, Early stage
   voice recognition
   parallel processing and superconductors are aiding this transition.
   quantum computers and nanotechnology.

2.4 Services of OS:

1. provide programs an environment to execute.
2. provide users means to run programs.

2.5 Program execution

1. leads a program into memory.
2. execution the program.
3. handles program's execuion.
4. Provides a mechanism for process syncronization.
5. Provides a mechanism for process communication.
6. Provides a mechanism for deadlock handling.

2.6 Operations on process

1. Creation:
   Once a process is created it is in ready state and is present in main memory.

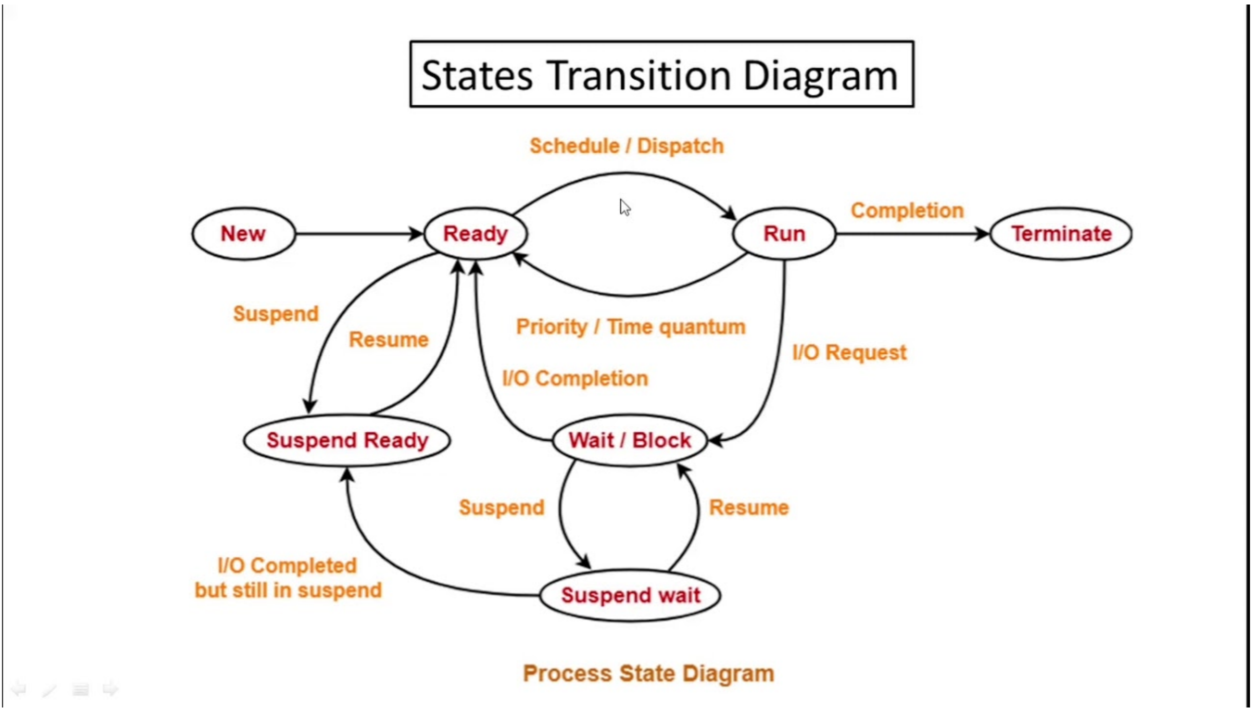
2. Scheduling
   When we have many processes in the ready queue we schedule it and give it to the CPU i.e. allocate the CPU to one of them on the basis of Scheduling.

   - New state: Process is about to be created and it is currently present in the secondary memory.

3. Executing:
   Once we give a process to the CPU it will start the execution.

4. Killing/Delete:
   Whenever we are done with the process we can simply kill the process.

2.7 Process state transition Diagram



NOTE:

- The minimum number of states a process can go through is 4. (New, Ready, Run, Terminate)
- Priority or time quantum may cause a process to premp
- Suspend wait or suspend ready are in secondary memoryare in secondary memory.
- I/O can occur anywhere (in main memory as well as in secondary memory) without any interruption once it is initiated.

2.8 Types of Schedulars in OS

There are 3 types of Schedulars in OS:

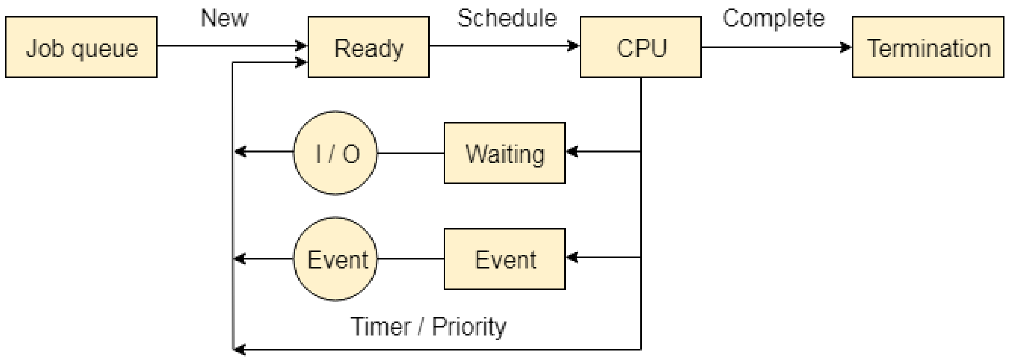
1. Long term Schedular:
   Long term Schedular is responsible for the decisions about how many processes have to be created.

2. Short term schedular:
   Once we have a pool of processes waiting in ready queue to be alloted to CPU for execution. This decision is taken by short term scheduler AKA dispatcher.

3. Medium term scheduler:
   Suspension decisions are taken by Medium term scheduler. this decision can be taken either in ready state or in wait or block state.

NOTE:
Long term decisions of how many processes to be created is also known as degree of multi-programming. (Number of processes that are present in the ready queue at maximum)

When the processes are present in the states, they're basically presented as queues. These queues are implemented as linked lists.



fork() command is used for creating a new process and that process will go to the ready queue.

2.9 Various times related to processes

1. Arrival time:
   Whenever a process gets in the ready queue it is the time at which the process is ready for execution.

2. Burst time:
   The amount of CPU time required by a process to finish (hard to predict/detect) in real time.

3. Completion time:
   The time at which the process completes it's execution is called completion time.

4. Turn around Time:
   Completion time and arrival time are basically points in time whereas waiting time and burst time are durations in time. In between arrival time and completion time we can have a process either waiting or executing. The difference between completion time and arrival time is called Turn around time.

5. Waiting time:
   The difference between Turnaround Time and Burst time is called Waiting time.

6. Response time:
   What is the first time the process hits the CPU after it arrives.

2.10 CPU Scheduling:

Picking up a process from the ready queue and giving it to the CPU is called CPU scheduling. The short term scheduler will perform CPU scheduling.

- Who $\rightarrow$ short term scheduler
- Where $\rightarrow$ Ready state to running
- When $\rightarrow$ When a process moves

2.11 Scheduling algorithms

FCFS(First Come First Serve):

- if two processes have the same arrival time then pick the one with lower process numbers.

Note: one disadvantage of FCFS is convey effect (convey: when a process is having large burst time then all the processes which are greater will have to starve. This is also known as starvation process)

SJF(Shortest job first):

- Criteria: burst time
- Mode: Non premption

Whichever process has the least burst time is executed first. The process is not stopped until it is finished. Whenever two processes have the same burst time pick the one which has arrived earlier. SJF is the best known scheduling algorithm as of now. Advantages:

- maximum throughput
- minimum average waiting time.

Disadvantages:

- Starvation to longer jobs.

- It is not implemented because burst time of processes cannot be known as ahead

Throughput is maximum at any point of schedule for SJF because we complete as many jobs as possible depending upon test burst time.

2.12 Priority Scheduling

Priority Scheduling is of two types:

- Static: it does not change throughout the execution of the process
- Dynamic: it changes at regular intervals of time.

Note: Whenever the process enters the ready queue, it comes alongwith a number associated with it This number is known as priority. Implementation may vary.

Sometimes highest number means lowest priority and sometimes lowest number means highest priority. Among all available processes we pick the one with the highest priority.

Priority scheduling algorithm can be classified into two types:

1. Preemptive: We are choosing a process with highest process numbers and run it either till completion or till some other processes with higher priority entering the ready queue.
2. Non-premptive: When we choose the process with the highest priority we are going to run it till completion without stopping it.

For example: (Lower number higher priority)

Process	Priority	Arrival time (AT)	Burst time (BT)
P_1	3	0	8
P_2	4	1	2
P_3	4	3	4
P_4	5	4	1
P_5	2	5	6
P_6	6	6	5
P_7	1	10	1

2.12.1 Preemptive Scheduling Algorithm

Process	Priority(higher the more)	Arrival time (AT)	Burst time (BT)
P_1	10	0	5
P_2	20	1	4
P_3	30	2	2
P_4	40	4	1

Criterion: Priority
Mode: Premptive

run a process till a process with higher priority is available

2.13 Process synchronization:

As long as processes do not interfere with each other i.e. they run independently, there will be no problem at all as long as the OS protects memory space of one process from another. There can be some problems if two processes are trying to communicate with each other. Here in operating systems, interprocess communication refers to communication between the processes running on the same computer.

2.14 System call

The fork system call is used to create a new process or linux or unix system which is called child processes which runs concurrently with the processes which makes the fork command as parent process. after a new child process is created, both processes execute following command with the fork system call.

$$fork = \{$$

below are the different values returned by fork():

1. negative value The creation of a child process was unsuccessful
2. zero returns to the child process.
3. positive value returns to the parent process and the value is the PID of the process.

2.15 Deadlock

a set of processes are said to be in deadlock if they wait for happening of an event caused by others in the same set.

Note: Starvation is long waiting and deadlock is infinite waiting.

Every deadlock is a starvation but every starvation needn't be a deadlock.
Resource: A resource can either be a file or a variable or a semaphor or a register or even a cpu. everything in a computer is known as a resource and every process some of it's processes for it's execution. And whenever it wants a resource it is going to ask the operating system.

printer only one process should access the printer at a time. non preemption: while some process is using some resource we are not allowed to pull that resource e.g. while some process is printing a file, we are not allowed to stop the printing and give the resource (or printer) to some other process.

circular wait: Mutual exclusion: Let us say we have a printer and the user wants to use it. so in order to use a device like a printer we can use spooling method. so if we have a printer and if we want to use it we can just say what we want to do with the printer (we just write the job and leave it be). printer follows FCFS(First come first serve) So these processes are not completely for the printer. Instead they are competing for the school memory. Then we put the jobs in the spool memory and later printer will read the jobs from the spool memory and execute them one-by-one. Therefore at any time only one process will be accessing the printer but this spooling process is not a feasible process for all the resources. In fact it is also not feasible for the printer because now the processes will be competing for the spool memory. Using spooling we cannot completely solve the problem but to some extent, yes we can. Hold and wait condition:

Condition	**Approach**
Mutual Exclusion	Spool everything
Hold and Wait	Request all resources initially
No premption	Take resources array
Circular Wait	Order resources Numerically

If we want to resolve the situation then we need to allot all the resources to a process initially before it starts executing. The solution is Hold or Wait.

1. Suppose we are having a process which needs two resources for it's execution and both the resources are available only and only we execute the process, otherwise we not.

2. Do not hold. If we want to make a new request for a new resource, then release all the resources which you're holding and then make the request together for all the resources. The problem with this approach is we cannot predict well ahead of time how many resources we need and what are the resources we need. So disabling this condition is practically not feasible because it is very difficult to guess the requirements of a program in advance.

3. No premption says that why the process is using a resource. We cannot pull it forcefully. Now in order to break or disable this condition, we can take the resources away but this is not a good idea. because it leads to inconsistent results.

4. Circular wait condition. Let us say all the resources are numbered as {R1, R2, R3, R4} Now suppose a process P1 has requested a resource R2 and got it, then it should not ask for resource R1 but instead of this it should ask for something greater than R2. Now since every process is asking for the resources in increasing order then obviously no process will come back and ask for a lower number. Therefore the process will not be in a cycle. In this way we can avoid circular wait. Therefore the procedure is number all the resources and give them some ordering in sequence and the processes should be restricted in such a way that you should ask for the resources in an increasing order. In case at any point of execution, if it requires a resource, which is lower than the resources it already holds then it is supposed to release all the resoures it is holding. and only after then it can request the lower numbered resources.

Note: This is the only approach that can be implemented practically and all other approaches cannot be implemented practically

(So all these three conditions cannot be disabled practically)

Banker's algorithm.
Max Resources: A-10, B-5, C-7

3 Virtual Machine

The main objectives/advantages of Virtual machines is whenever the process size is very much big than the main memory, then virtual machine will help the OS to run the processes. Due to this, the user will get an illusion that the main memory is available to run the complete process. E.G. there are two processes, P1 and P2, and here we are not going to load them entirely into the main memory as the main memory will not be sufficient for them.
Suppose the two processes P1 and P2 are in the secondary memory where both the processes are divided into 8-8 pages and processes. Let us now assume that the main memory is divided into 16 frames.
In the main memory, top three frames are reserved for the OS. The rest of the frames will be allocated to the user.

4 Semaphors

It is an integer variables that, apart from initialization, is accessed only through two standard atomic operations - wait and signal. This is used for - critical section and to decide the order of execution among processes and resource management.
A semaphor is a variable that provides an abstraction for controlling the access of a shared resource by multiple processes in a parallel programming environment. in wait operation the semaphor will decrement and in signal operation the semaphor will increment. Wait can also be denoted as $P(S)$.

Problem A

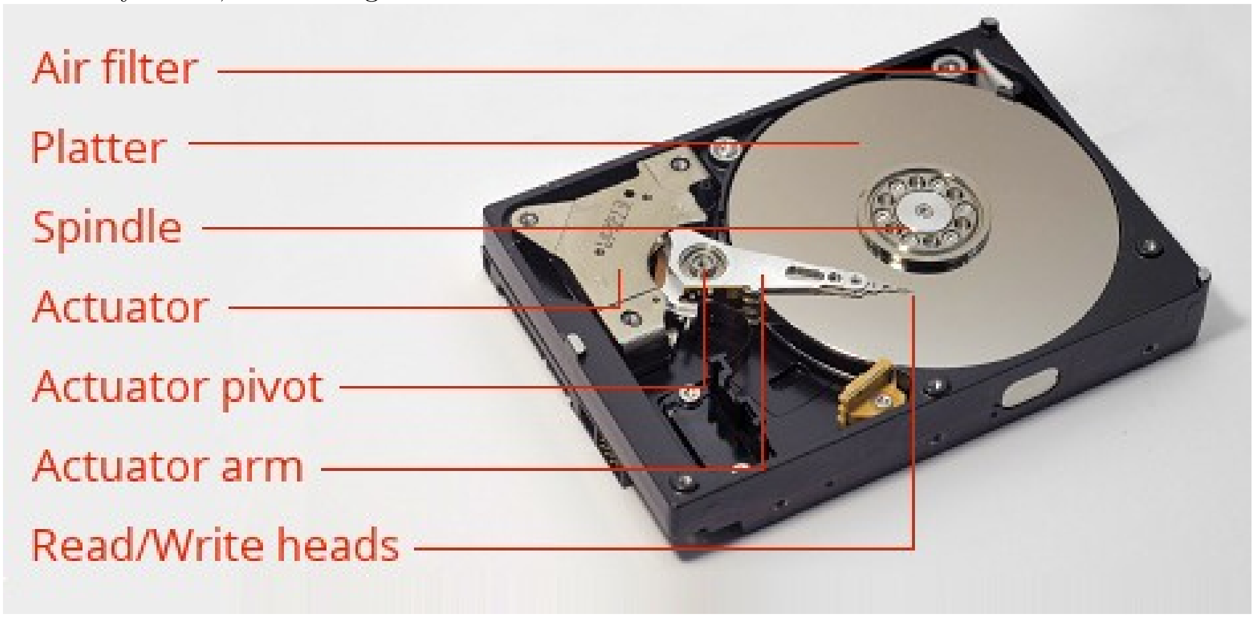
counting semaphor was initialized to 8 then to P. Operations and seven V (signal) operations were completed on this semaphor What is the resulting value of semaphor.

Problem A

Initial counting: 10
P ops: 6
V ops: 4
Find the final counting (Ans: 8)

5 OS scheduling/ Disk arm scheduling algorithm

Theres an arm, theres a spindle
theres my heart, slit through the middle.



The goal of disk arms scheduling is to minimize the seek time. seek time is the time taken to reach upto the desired track.

1. FCFS/FIFO
2. Shortest seek time first (SSTF)
3. Scan AKA Elevator Algorithm
4. Look
5. C-Scan (circular scan)
6. C-Look (circular look)

Problem A

disk contains 200 tracks (0- 199) and request queue contains track ID 82, 170, 43, 140, 24, 16, 190 respectively. The current position of read-write head is 50. Calculate total number of track moments by read-write head. (by FIFO)

6 Paging

Paging is a storage mechanism used to retrieve processes from the secondary storage into the main memory, RAM, in the form of pages. The main idea behind the paging is to divide each process in the form of pages. The main memory will also be divided in the form of frames. One page of the process is to be stored in one of the frames of the memory. In a system we have three main components:

1. RAM (upto 16GB)
2. Secondary memory (upto 2TB)

note that page size = frame size.
Suppose you are having a page of 2B and a process of size 4B. The number of pages per process will be 2. If the main memory is of 16B then the frame size is going to be 2B and the number of frames will be 8. The pages can be stored under different locations of the memory but the priority is to always find the contiguous frames. Pages of the process are brought into the main memory only when they are required otherwise they reside in the secondary storage. the size of each frame must be equal considering the fact that the pages are mapped to the frames in paging. The page size needs to be as same as the frame size. Different OS define different frame sizes.

Problem A

Consider a system which has logical address 7 bits, physical address 6 bits, page size 8 words. Then calculate the number of pages and number of frames.