Special Edition for CSEDU Students

TOUCH-N-PASS EXAM CRAM GUIDE SERIES

OPERATING SYSTEMS

Prepared By

Sharafat Ibn Mollah Mosharraf

CSE, DU

12th Batch (2005-2006)

Includes DU OS Final Exam Questions of 7 Years (1999-2006)

TABLE OF CONTENTS

CHAPTER 2
PROCESSES

Questions

2.1	Discuss the differences between a process and a program. *[1999. Marks: 4]*
2.2	Define sequential and concurrent processing. Write with at least a real life example for each. *[1999. Marks: 3]*
2.3	Describe process control block. *[2003. Marks: 4]* OR, What is process control block (PCB) of a process? Explain the information that are maintained by the PCB. *[2002. Marks: 1 +3]*
2.4	What is context switch? Why are context switch times highly dependent on hardware support of a computing system? Explain briefly. *[2004. Marks: 1 + 2]*
2.5	Why does context switch create overhead? *[2003. Marks: 3]*
2.6	**Define process states. If a process is in execution state, show the next state for each of the following conditions:** ***[1999. Marks: 3 + 3]*** **(i) Process makes a page fault** **(ii) Process requests secondary storage to store "I print it" to tt.txt file.** **(iii) The process is interrupted by another process.** (i) Blocked (ii) Blocked (iii) There may be three cases: a. If the process has an interrupt handler available for that interrupt, it will be in execution state. b. If the process accepts the default action of the interrupt, which is to terminate itself, then it will be in termination state. c. If the interrupt is of type SIGSTOP which suspends the process, the process will be in ready state.
2.7	Define the states of process. *[2000. Marks: 3]* OR, Draw the process state diagram. Briefly explain the state transitions. *[2004. Marks: 1 + 2]* OR, In the *three-state process model* (*ready*, *running* and *waiting*), what do each of the three states signify? What transitions are possible between each of the states, and what causes a process (or thread) to undertake such a transition? *[2002. Marks: 2 + 2]*
2.8	Write down the differences between a process and a thread. *[In-course 2, 2004. Marks: 3]*
2.9	How do threads differ from processes? What are the advantages and disadvantages of using threads? *[2004. Marks: 2 + 2]*
2.10	A single process contains fifteen threads. What are the item(s)/resource(s) shared by these threads? *[2006. Marks: 2]*
2.11	Compare reading file using a single-process file server and a multi-process file server. Within the file server, it takes 15ms to get a request for work and do all the necessary processing; assuming the required block is in the main memory disk block cache. A disk operation is required for one-third of the requests, which takes an additional 75ms during which the process sleeps. How many requests/sec can a server handle if it is single process server and if it is multiprocess server? *[2002. Marks: 4]*
2.12	Distinguish between message passing model and shared memory model. *[2003. Marks: 4]*
2.13	What is critical section problem? *[2004, 2003, 2000. Marks: 1]*
2.14	Define the properties that a solution to critical section problem must satisfy. *[2004. Marks: 2]*

OR, What conditions must be met to solve critical section problem? *[2003. Marks: 3]*
OR, Define the requirements that satisfy critical section problem. *[2000. Marks: 2]*

2.15 Consider the following code that shows the structure of a process in an algorithm to solve the critical section problem for two processes.

```
Var Flag[2] of Boolean; //initialized to false

Repeat
   Flag[i] := true;
   While Flag[j] do no-op;

   Critical Section

   Flag[i] := false

   Remainder Section
Until false;
```

Does the algorithm satisfy all requirements for critical section problem? Justify your answer. *[2004. Marks: 3]*

2.16 Use of TEST_AND_SET to solve the synchronization problem creates a new problem race condition. What do you understand by race condition? How can we overcome race condition in programming using semaphores? *[In-course 1, 2008. Marks: 5]*

2.17 Suppose you are given two large matrix (40000 × 40000) to multiply and you are asked to use one process for calculating one row of the resultant matrix. How many critical section(s) are there? *[In-course 1, 2008. Marks: 5]*

2.18 What do you understand by atomic instruction? Why is it necessary to solve the critical section problem? *[1999. Marks: 2 + 2]*

2.19 What is busy waiting? *[2004, 1999. Marks: 1]*
Can it be avoided altogether? Why or why not? *[2004, 2000. Marks: 1]*

2.20 Explain the difference between busy waiting and blocking. *[1999. Marks: 2]*

2.21 Give the definition of TEST-and-SET instructions. How is it used to achieve mutual exclusion? Show whether the solution satisfies the other requirements of critical section problem. *[2004. Marks: 2 + 2 + 2]*

2.22 What is meant by process cooperation? "Producer-consumer" problem is a paradigm for cooperating process." – Explain. *[2002, Marks: 1 + 5]*

2.23 Show that if the wait and signal operations are not executed atomically, then mutual exclusion may be violated. *[2000. Marks: 3]*

2.24 **Following are some operations related with given data structure.**
(a) Find out the critical region.
(b) Rewrite the code to solve the critical section problem residing in the above code. Your solution must be free from busy waiting. *[2006. Marks: 1 + 3]*

```
struct queue {
    item_type *start;
    item_type *end;
};

struct queue *Q;

void init() {
    Q = NULL;
}

int insert(item_type *item) {
    if (!Q) {
        Q = (struct queue*) malloc(sizeof(struct queue));
        Q->start = NULL;
        Q->end = NULL;
    }
    if (Q->start == NULL)
        Q->start = item;
    item_type *temp;
    temp = Q->end;
    Q->end = item;
    item->next = temp;
}

item_type *delete() {
    item_type *temp;
```

```c
    if (Q->start == Q->end == NULL)
        return NULL;
    if (Q->start == Q->end != NULL) {
        temp = Q->start;
        Q->start = NULL;
        Q->end = NULL;
        return temp;
    }
    temp = Q->start;
    Q->start = temp->next;
    return temp;
}
```

(a) The code statements where the queue is accessed are part of the critical section. The functions insert() and delete() are included in the critical region.

(b) The rewritten code solving the critical section problem is as follows:

```c
struct queue {
    item_type *start;
    item_type *end;
};

struct queue *Q;

struct semaphore *mutex;

void init() {
    Q = NULL;
}
```

```c
int insert(item_type *item) {
    item_type *temp;
    down(mutex);
    if (!Q) {
        Q = (struct queue*) malloc(sizeof(struct queue));
        Q->start = NULL;
        Q->end = NULL;
    }
    if (Q->start == NULL)
        Q->start = item;
    temp = Q->end;
    Q->end = item;
    item->next = temp;
    up(mutex);
}

item_type *delete() {
    item_type *temp;
    down(mutex);
    if (Q->start == Q->end == NULL) {
        up(mutex);
        return NULL;
    }
    if (Q->start == Q->end != NULL) {
        temp = Q->start;
        Q->start = NULL;
        Q->end = NULL;
        up(mutex);
        return temp;
    }
    temp = Q->start;
    Q->start = temp->next;
    up(mutex);
    return temp;
}
```

2.25 Can CPU scheduling reduce the total execution time of processes? Based on your opinion, discuss the necessity of CPU scheduling. *[2002. Marks: 1 + 2]*

2.26 **When scheduling is performed? *[2006. Marks: 2]***

(i) A child process is created
(ii) Parent process terminated
(iii) A process sleeps on semaphore
(iv) Process goes for I/O operation

Scheduling is performed in all the cases mentioned.

2.27	**Which of the following scheduling algorithms could result in starvation?** **a. FCFS** **b. SJF** **c. RR** **d. Priority** SJF and Priority.
2.28	What is the advantage of multilevel scheduling? Explain with the help of an example. *[In-course 2, 2004. Marks: 2]*
2.29	Why most of the operating systems use multilevel-feedback scheduling for process scheduling? *[1999. Marks: 3]*
2.30	What are the parameters that define the multilevel feedback scheduling method? How does it prevent starvation? *[2004. Marks: 3]*
2.31	**What advantage is there in having different time-quantum sizes on different levels of a multilevel queuing system? *[2004. Marks: 2]*** Processes that need more frequent servicing, for instance, interactive processes such as editors, can be in a queue with a small time quantum. Processes with no need for frequent servicing can be in a queue with a larger quantum, requiring fewer context switches to complete the processing, making more efficient use of the computer.
2.32	Discuss the impact of the size of time quantum of Round-Robin scheduling on CPU efficiency. *[2002. Marks: 3]*
2.33	Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 milliseconds and that all processes are long-running tasks. What is the CPU utilization for a RR scheduler when the time quantum is 1ms and 10 ms?
2.34	**The exponential average algorithm with α = 0.5 is being used to predict run times. The previous four runs, from oldest to most recent, are 40, 20, 40, and 15 msec. What is the prediction of the next time? *[In-course 2, 2004. Marks: 3]*** $\tau_1 = 0.5 \times 40 + 0.5 \times 40 = 40$ $\tau_2 = 0.5 \times 40 + 0.5 \times 20 = 30$ $\tau_3 = 0.5 \times 30 + 0.5 \times 40 = 35$ $\tau_4 = 0.5 \times 35 + 0.5 \times 15 = 25$ ∴ The prediction of the next time is 25 msec.
2.35	With an example, prove that for real time periodic scheduling $\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$, for *m* periodic events, event *i* occurring with period P_i requires C_i seconds. *[2006. Marks: 3]*
2.36	**A soft real-time system has four periodic events with periods of 50, 100, 200, and 250 msec each. Suppose that the four events require 35, 20, 10, and *x* msec of CPU time, respectively. What is the largest value of *x* for which the system is schedulable?** The fraction of the CPU used is 35/50 + 20/100 + 10/200 + *x*/250. To be schedulable, this must be less than 1. Thus *x* must be less than 12.5 msec.

Exercises

2.37 Assume you have the following processes to execute with one processor, with the processes arriving at the following times and having the following CPU burst times and priorities:

Process	Arrival Time	CPU Burst Time	Priority
A	0	8	3
B	3	4	1
C	5	7	4
D	8	3	2

For each of the following scheduling algorithms:

i. Shortest Job First (SJF), Preemptive
ii. Priority, Preemptive

1. Draw a Gantt chart.
2. Calculate average waiting time.
3. Calculate turnaround time of each process. *[2007. Marks: 2 + 2 + 1]*

2.39 Consider the following set of processes, with the length of CPU burst given in milliseconds:

Process	Arrival Time	Burst Time
P1	0	16
P2	0	13
P3	4	4
P4	3	6
P5	11	9

i) Draw Gantt charts illustrating the execution of these processes using Preemptive SJF and RR (with time-slice = 4 ms) scheduling.

ii) Find the average waiting time and average response time for each algorithm. *[2004. Marks: 3 + 4]*

2.40 Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

Process	Arrival Time	CPU Burst Time	Priority (lower value higher priority)
P_0	0	16	2
P_1	0	6	1
P_2	3	8	0
P_3	6	1	2
P_4	12	3	0
P_5	17	2	1
P_6	20	5	1

i) Draw Gantt charts describing the execution of these processes using FCFS, SJF (preemptive), preemptive priority, and RR (quantum size 2 ms) scheduling.

ii) Compute the average waiting time and average turnaround time for each algorithm. *[2003. Marks: 10]*

2.41 Consider the following table:

Process	Arrival Time	Burst Time
P1	1.2	2
P2	0.8	5
P3	0.0	4
P4	1.4	7

i) What are the average turnaround times for FCFS, SJF (preemptive) and RR (time quantum = 1) scheduling?

ii) What are the average waiting times for each of the algorithms mentioned in (i)? *[2000.*

Marks: 5]

2.42 What is the average waiting time for the following processes with RR scheduling? Note that time quantum is 2*ms*. *[In-course, 2003. Marks: 5]*

Process	Arrival Time (ms)	Burst Time (ms)
P_1	0.0	10.0
P_2	1.2	15.0
P_3	2.3	8.0
P_4	3.4	20.0
P_5	5.0	27.0

2.43 Five batch jobs *A* through *E*, arrive at a computer center at almost the same time. They have estimated running times of 10, 6, 2, 4, and 8 minutes. Their (externally determined) priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead.

(a) Round robin.
(b) Priority scheduling.
(c) First-come, first-served (run in order 10, 6, 2, 4, 8).
(d) Shortest job first.

For (a), assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For (b) through (d) assume that only one job at a time runs, until it finishes. All jobs are completely CPU bound. *[In-course 2, 2004. Marks: 4]*

2.44 Followings are the portion of codes for reader and writer processes:

<u>Reader Process</u>

```
wait(mutex);
readcount += 1;
if (readcount == 1)
      wait(wrt);
signal(mutex);
.......................................
reading is performed
.......................................
wait(mutex);
readcount -= 1;
if (readcount == 0)
      signal(wrt);
signal(mutex);
```

<u>Writer Process</u>

```
wait(wrt);
.......................................
writing is performed
.......................................
signal(wrt);
```

By using the following information, determine the turnaround time of each operation: *[In-course, 2003. Marks: 5]*

Operation	Arrival Time (ms)	Execution Time (ms)
Read1	0.0	5
Read2	2.0	6
Write1	2.5	9
Read3	2.6	7
Read4	3.0	5
Write2	3.0	8

Chapter 3
Deadlock

Questions

3.1 What is deadlock? What are the conditions that turn a system into deadlock? *[2003, Marks: 4. 2002, Marks: 3]*

3.2 Give an example and explain that use of semaphore in inappropriate sequence may cause deadlock. *[In-course 1, 2008. Marks: 5]*

3.3 Show that if we allow one process to access more than one critical regions concurrently, then the system may be in deadlock. *[1999. Marks: 3]*

3.4 What is a knot? Give your comment on the statement: "A cycle in the resource allocation graph is a necessary but not a sufficient condition for the existence of deadlock but a knot is a sufficient condition for deadlock." *[2004. Marks: 3]*

3.5 Describe a method to prevent deadlock by ensuring circular wait does not occur. Prove the correctness of the method. *[2004. Marks: 2 + 2]*

3.6 What are the limitations of deadlock prevention techniques? How can deadlock avoidance algorithms overcome those? Explain. *[2004. Marks: 1 + 2]*

3.7 Why none of the basic approaches (prevention, avoidance and recovery from deadlock) alone is not appropriate for the operating system to keep the system deadlock free? *[1999. Marks: 4]*

3.8 Differentiate between deadlock and starvation. *[1999. Marks: 4]*

3.9 Can a system detect starvation? If 'yes', how? If 'no', how starvation can be dealt with? *[2000. Marks: 3]*

3.10 What factors should you consider to determine which process is to be terminated in order to recover from deadlock? *[2004. Marks: 3]*

3.11 **Following is a snapshot of a set of processes in deadlock. Which process(es) should be aborted to break deadlock to get best throughput?** ***[2002. Marks: 3]***

Process	**Allocated Resource(s)**	**Request for Resource(s)**	**Execution Completed**	**Future Demand Resource(s)**
P_1	**F**	**D**	**20%**	**B, A**
P_2	**C**	**E**	**30%**	**A, B, C**
P_3	**B**	**A**	**40%**	**C, D, E**
P_4	**D, A**	**C, B**	**60%**	**E, F**

P_2 and P_4.

[To answer this, first construct the matrices:

Process	**Allocation**						**Requests**						**Available**					
	A	**B**	**C**	**D**	**E**	**F**	**A**	**B**	**C**	**D**	**E**	**F**	**A**	**B**	**C**	**D**	**E**	**F**
P_1	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1	0
P_2	0	0	1	0	0	0	1	1	1	0	1	0						
P_3	0	1	0	0	0	0	1	0	1	1	1	0						
P_4	1	0	0	1	0	0	0	1	1	0	1	1						

Now, kill a single process, add its resources to the available matrix and run banker's algorithm to find a safe sequence. Repeat this for any combination of processes until you get one or more processes so that killing them will bring a safe sequence. In this case, Killing P_2 and P_4 would bring a safe sequence.]

3.12 Define the safe and unsafe states of a system. *[2004. Marks: 2]*

3.13	Show that, in a system with unsafe state, processes can complete their execution without entering a deadlock state. *[2000, 1999. Marks: 2]*
3.14	Suppose there are three real time processes and two resources. Show that the system will never go to deadlock. *[In-course, 2003. Marks: 2]*
3.15	Suppose a system has three resources each with two instances; and has four processes. Each process required two resources (one instance of each resource type) at a time. Show that the system never goes to deadlock. *[In-course 1, 2008. Marks: 5]*
3.16	A system has thousands of processes (> *m*) and limited number of resource (< *n*) and each resource has multiple instances. The processes have some special behaviors. If any of the process request for one or a set of instances of one or many type of resources, the instances of such resource(s) will be allocated immediately if available. If instances of one/more resource(s) type is not available (as requested), it will give up any instances of such resource(s) which have been allocated previously. Show that the system will never be in deadlock or unsafe state. (Assume that a process will never request more instances than the system has for a resource type.) *[In-course 2, 2008. Marks: 5]*
3.17	A system has 2 processes and 3 identical resources. Each process needs a maximum of two resources. Is deadlock possible? Explain your answer. Consider the previous problem again, but now with *p* processes each needing a maximum of *m* resources and a total of *r* resources available. What condition must hold to make the system deadlock free?
3.18	Consider a system consisting of *m* resources of the same type being shared by *n* processes. Resources can be requested and released by processes only one at a time. Show that the system is deadlock free if the following two conditions hold: (a) The maximum need of each process is between *l* and *m* resources. (b) The sum of all maximum needs is less than *m* + *n*. *[2004, Marks: 3; 2000, Marks: 6]*
3.20	**Consider the following resource allocation policy. Requests and releases for resources are allowed at any time. If a request for resources cannot be satisfied because the resources are not available, then we check any processes that are blocked, waiting for resources. If they have the desired resources, then these resources are taken away from them and are given to the requesting process. The vector of resources for which the waiting process is waiting is increased to include the resources that were taken away.** **For example, consider a system with three resource types and the vector *Available* initialized to (4, 2, 2). If process P_0 asks for (2, 2, 1), it gets them. If P_1 asks for (1, 0, 1), it gets them. Then, if P_0 asks for (0, 0, 1), it is blocked due to resource unavailability. If P_2 asks for (2, 0, 0), it gets the available one (1, 0, 0) and one that was allocated to P_0 (since P_0 is blocked). P_0's *Allocation* vector goes down to (1, 2, 1), and its *Need* vector goes up to (1, 0, 1).** **(a) Can deadlock occur? If so, give an example. If not, which necessary condition cannot occur?** **(b) Can indefinite blocking occur? Why? *[2004. Marks: 4]*** (a) Deadlock will not occur. The necessary condition that cannot occur is the *hold and wait* condition. (b) Indefinite blocking can occur. Because, processes might continuously come and take resources from the blocked process; thus the blocked process might not get a chance to fulfill its needs.

Exercises

3.19 **Using the Banker's algorithm, determine if the following system is in deadlock. If it is, which process(es) are deadlocked? If not in deadlock, what is the safe sequence? You need to show all intermediate steps to get full marks. $P_1 – P_5$ are processes, and A, B, C, D are resource types.**

(a) Determine if a request from process P_2 of (1, 0, 2, 1) instances of resource A, B, C and D respectively will be granted immediately or not. Explain your answer.

(b) After fulfilling the request of question (*a*), will the system grant request of P_1 (0, 2, 0, 3)? *[2006. Marks: 6]*

Process	Allocation					Max					Available			
	A	B	C	D		A	B	C	D		A	B	C	D
P_1	0	1	0	2		7	5	3	6		3	3	2	4
P_2	2	0	0	1		3	2	2	3					
P_3	3	0	2	1		9	0	2	1					
P_4	2	1	1	2		2	2	2	3					
P_5	0	0	2	1		4	3	3	4					

Solution:

The matrix "still needed":

Still Needed			
A	B	C	D
7	4	3	4
1	2	2	2
6	0	0	0
0	1	1	1
4	3	1	3

From the matrix "still needed", we can find the following sequence of processes that can be satisfied with the available resources one after another. The available resources after each process finishes are also given:

Process	Available			
	A	B	C	D
P_4	5	4	3	6
P_2	7	4	3	7
P_3	10	4	5	8
P_5	10	4	7	9
P_1	10	5	7	11

As all the processes in the above sequence can be satisfied, therefore, the system is not in deadlock.

The safe sequence is <P_4, P_2, P_3, P_5, P_1>.

(a) Yes, the request can be granted immediately; because a safe sequence <P_4, P_5, P_2, P_1, P_3> can still be found.

[**Note:** to answer this type of question, you'll have to do the following:

1. Add (1, 0, 2, 1) to P_2 in the *Allocation* matrix.
2. Delete (1, 0, 2, 1) from P_2 in the *Still Needed* matrix.
3. Delete (1, 0, 2, 1) from the *Available* matrix.
4. Now again apply the Banker's algorithm to determine whether a safe sequence can be found.

The matrices should look like the following after these 3 steps are done:

Process	Allocation					Max					Available				Still Needed			
	A	B	C	D		A	B	C	D		A	B	C	D	A	B	C	D
P₁	0	1	0	2		7	5	3	6		2	3	0	3	7	4	3	4
P₂	3	0	2	2		3	2	2	3						0	2	0	1
P₃	3	0	2	1		9	0	2	1						6	0	0	0
P₄	2	1	1	2		2	2	2	3						0	1	1	1
P₅	0	0	2	1		4	3	3	4						4	3	1	3

]

(b) No, the request cannot be granted; because none of the processes can be satisfied with the available resources (2, 1, 0, 0) after the request is granted.

3.22 **Answer the following questions about the tables given below: *[2002. Marks: 1+2+2+2+3]***

Process	Current Allocation					Maximum Demand					Still Needs					Available			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P_1	0	0	1	2		0	0	1	2							2	1	0	0
P_2	2	0	0	0		2	7	5	0										
P_3	0	1	3	4		6	6	5	6										
P_4	2	3	5	4		4	3	5	6										
P_5	0	3	3	2		0	6	5	2										

i) Compute what each process still might request and display in the columns labeled "still needs".

ii) Is the system in a safe or unsafe state? Why?

iii) Is the system deadlocked? Why or why not?

iv) Which processes, if any, are or may become deadlocked?

v) Assume a request from P4 arrives for (0, 1, 0, 0).

a. Can the request be safely granted immediately?

b. In what state (deadlocked, safe and unsafe) the system would go if the request is granted immediately?

c. Which processes, if any, are or may become deadlocked if the request is granted immediately?

Solution:

i)

Still Need			
A	B	C	D
0	0	0	0
0	7	5	0
6	5	2	2
2	0	0	2
0	3	2	0

ii) The system is in a safe state. Because all the processes in the sequence <P_1, P_4, P_5, P_2, P_3> can be satisfied with the available resources.

iii) The system is not deadlocked. Because there exists a safe sequence.

iv) No processes are or may become deadlocked.

v) a. Yes, the request can be safely granted. The safe sequence is <P_1, P_4, P_5, P_2, P_3>.
b. The state would be in safe state if the request is granted immediately.
c. No processes may become deadlocked if the request is granted immediately.

3.21 **Consider the following snapshot of a system:**

Process	Allocation				Max				Available			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P_1	0	0	1	2	0	0	1	2	1	5	2	0
P_2	1	0	0	0	1	7	5	0				
P_3	1	3	5	4	2	3	5	6				
P_4	0	6	3	2	0	6	5	2				
P_5	0	0	1	4	0	6	5	6				

Answer the following questions using the Banker's algorithm:

i) **Is the system in safe state? [Why?] If it is, then write the safe sequence.**

ii) **What is the content of the matrix *Need*? If a request from process P_1 arrives for (0, 4, 2, 0), can the request be granted immediately? Why?** ***[2000. Marks: 4 + 2 + 3]***

iii) **If a request from process P_2 arrives for (0, 4, 2, 0), can the request be granted immediately? Why?** ***[2004. Marks: 4]***

iv) **If a request from process P_4 arrives for (0, 3, 4, 0), can the request be granted immediately?** ***[2003. Marks: 4]***

CHAPTER 4
MEMORY MANAGEMENT

Questions

4.1 **Discuss the advantages and disadvantages of multiprogramming with fixed partitions and variable partitions.** ***[1999. Marks: 5]***

4.2 **Given memory partitions of 1000KB, 500 KB, 200 KB, 300 KB and 600 KB (in order), how would each of the first-fit, best-fit and worst-fit algorithms place processes of 212KB, 417KB, 112KB and 426KB (in order)? Which algorithm makes the most efficient use of memory from your opinion?** ***[2003. Marks: 6]***

First-fit:

a. 212K is put in 500K partition
b. 417K is put in 600K partition
c. 112K is put in 288K partition (new partition 288K = 500K - 212K)
d. 426K must wait

Best-fit:

a. 212K is put in 300K partition
b. 417K is put in 500K partition
c. 112K is put in 200K partition
d. 426K is put in 600K partition

Worst-fit:

a. 212K is put in 600K partition
b. 417K is put in 500K partition
c. 112K is put in 388K partition
d. 426K must wait

In this example, best-fit turns out to be the best.

4.3 **Calculate the physical memory address in multi-partition memory allocation if base register is 0x235F and limit register is 2000 for the following program address:** ***[In-course 1, 2008. Marks: 5]***

100, 200, 2002, 300, 324, 400, 5000, 6234, 700, 933

The base register is 0x235F or 9055.

Program address	Physical address
100	9055 + 100 = 9155
200	9055 + 200 = 9255
2002	Illegal reference
300	9055 + 300 = 9355
324	9055 + 324 = 9379
400	9055 + 400 = 9455
5000	Illegal reference
6234	Illegal reference
700	9055 + 700 = 9755
933	9055 + 933 = 9998

4.4 **Distinguish between external and internal fragmentation.** ***[2004. Marks: 3]***
OR, What is internal and external fragmentation? How this problem can be overcome? ***[2000. Marks: 2 + 1]***
OR, What is fragmentation? How fragmentation can be overcome by paging? ***[2003. Marks: 6]***

See the topic "Memory Fragmentation" under Theory 4.6.

4.5	**How can the problem with external fragmentation be solved by compaction? Illustrate with example.** ***[2004. Marks: 6]***
4.6	**Discuss memory management with bit maps and linked-lists with their merits and demerits.** ***[1999. Marks: 3]*** *See the relative topics under Theory 4.6.*
4.7	**Calculate the absolute (physical) address for the following relative address in paging system where frame size is 4 KB: 3F2BC0H.** ***[1999. Marks: 3]***
4.8	**A certain computer provides its users with a virtual memory space of 2^{32} bytes. The computer has 2^{20} bytes of physical memory. The virtual memory is implemented by demand paging, and page size is 2048 bytes. Determine the physical address from the following logical addresses: (consider pages are loaded sequentially)** ***[In-course 3, 2003. Marks: 7.5]*** **a. 123456** **b. 23465976** **c. 456** **d. 4789** **e. 124967**
4.9	**Differentiate between program address [*virtual address*] and physical address. How program addresses are mapped to physical address in demand paging system?** ***[2006. Marks: 3]*** **OR, What is paging? Describe the mechanism of paging.** ***[2000. Marks: 1 + 4]*** Program address is the address generated by CPU which corresponds to the virtual address space of a process. On the other hand, physical address is the address of the actual memory location of the process. Paging is a technique of implementing multiprogramming with variable partitions where the logical address space of the processes as well as the physical memory is divided into segments of the same length. *For the mechanism of paging, see the topic "How the Paging System / MMU Works" under Theory 4.8.*
4.10	**How can you measure the performance of demand paging?** ***[2003. Marks: 4]*** *See the topic "Performance of Demand Paging" under Theory 4.8.*
4.11	**When do page faults occur? Describe the actions taken by the operating system when a page fault occurs.** ***[2004. Marks: 1 + 2]*** **OR, write the basic steps in replacing a page when page faults occur.** ***[2003. Marks: 3]*** **OR, Describe the sequence of events when a page fault occurs. Describe which actions are taken by hardware and which by software.** ***[2002. Marks: 5]*** **OR, What is virtual memory? Describe what happens when a page fault occurs.** ***[2000. Marks: 1 + 4]*** Page faults occur when a process tries to access a page that is not available in main memory. Virtual memory is a technique of implementing multiprogramming with variable partitions where the logical address space of the processes is permitted to be noncontiguous, thus allowing a process to be allocated physical memory wherever the latter is available. *For steps in page fault servicing, see the relevant topic under Theory 4.8.*
4.12	**Each OS has its own methods for storing page tables. Most allocate a page table for each process. The hardware implementation of the page tables can be done by a set of dedicated registers. But the use of registers for the page table is satisfactory if the page table is reasonably small. Most contemporary computers, however, allow the page table to be very large. How can the page tables be implemented for these computers using main memory? What are the limitations of this approach? Propose a standard solution to this page table implementation problem and explain it shortly.** ***[In-course 3, 2004; Marks: 3 + 2 +3. 2004, Marks: 1 +3]***

	For the computers mentioned, the page tables can be implemented using multiple levels. *Describe the multilevel page table structure in short.* The limitation of this approach is that the lookup performance would be poor. A standard solution to this problem is to use Translation Look-aside Buffer, which is a cache in CPU storing page table entries. *Describe TLB in short.*
4.13	**What is an associative memory? With example, discuss how it functions and what advantages it offers. *[1999. Marks: 3]*** To speed up the lookup process of page tables, computers are equipped with a small hardware device for mapping virtual addresses to physical addresses without going through the page table. This device is called an associative memory. *For how associative memory functions, see the topic "Working Procedure of TLB" under Theory 4.8.*
4.14	**What type of hardware support do you prefer for storing information in a page table? Mention the purpose of valid and invalid bit of page table. *[2004. Marks: 5]*** I would prefer a small hardware device for mapping virtual addresses to physical addresses without going through the page table. *Describe TLB in short.* The purpose of valid/invalid bit of page table [careful, it's page table – not TLB] is to determine whether the page is loaded in main memory or not. If the bit is 0, it means that the page is not currently present in memory. If the bit is 1, it means the page is available in memory.
4.15	**What are the differences between *paging* and *segmentation*?**

Paging	Segmentation
1. Paging was invented to get a large linear address space without having to buy more physical memory.	1. Segmentation was invented to allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection.
2. The programmer does not need to be aware of this technique being used.	2. The programmer needs to be aware of this technique being used.
3. There is only *one* linear address.	3. There are many linear addresses.
4. Procedure and data cannot be distinguished and separately protected.	4. Procedure and data can be distinguished and separately protected.
5. Sharing of procedures among users is not facilitated.	5. Sharing of procedures among users is facilitated.

4.16 **What are the pros and cons of paging and segmentation? Why are segmentation and paging sometimes combined into one scheme? *[2004. Marks: 5]***

Pros of Paging:

1. The programmer does not need to be aware of this technique being used.
2. There is only *one* linear address.

Cons of Paging:

1. Procedure and data cannot be distinguished and separately protected.
2. Sharing of procedures among users is not facilitated.

Pros of Segmentation:

1. Sharing of procedures among users is facilitated.

2. Procedure and data can be distinguished and separately protected.

Cons of Segmentation:

1. The programmer needs to be aware of this technique being used.
2. There are many linear addresses.

Why segmentation and paging are sometimes combined into one scheme:

Segmentation and paging are often combined in order to improve upon each other. Segmented paging is helpful when the page table becomes very large. A large contiguous section of the page table that is unused can be collapsed into a single segment table entry with a page-table address of zero. Paged segmentation handles the case of having very long segments that require a lot of time for allocation. By paging the segments, we reduce wasted memory due to external fragmentation as well as simplify the allocation.

4.17 **Briefly describe the 2nd chance algorithm for page replacement.** ***[2003. Marks: 5.5]***

See Theory 4.10.

4.18 **What are the main causes of thrashing? Briefly describe the working set model that may be used for locality.** ***[2003. Marks: 3]***

The main cause of thrashing is that after a process is brought back from disk to memory, it causes many page faults until its working set has been loaded.

The Working Set Model:

This model uses a parameter, Δ, to define the working-set window. The idea is to examine the most recent Δ page references. The set of pages in the most recent Δ page references is the working set. If a page is in active use, it will be in the working set. If it is no longer being used, it will drop from the working set Δ time units after its last reference. Thus, the working set is an approximation of the program's locality.

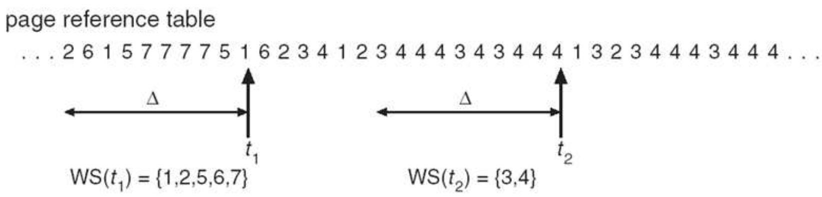


Figure: Working Set Model.

For example, given the sequence of memory references shown in the above figure, if Δ = 10 memory references, then the working set at time t_1 is {1, 2, 5, 6, 7}. By time t_2, the working set has changed to {3, 4}.

4.19 **A computer has four page frames. The time of loading, time of last access, and the *R* and *M* bits for each page are as shown below (the times are in clock ticks):**

Page	Loaded	Last ref.	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

(a) Which page will NRU replace?
(b) Which page will FIFO replace?
(c) Which page will LRU replace?
(d) Which page will second chance replace?

(a) 2 (b) 3 (c) 1 (d) 2

Exercises

4.1	**Consider a paging system with page table stored in memory. If we add associative registers [*TLB*], and 75% of all page-table references are found in the associative registers, what is the effective memory reference time? (Memory reference take 200 ns and associative memory reference takes 5 ns)** ***[2002; In-course 3, 2003; Marks: 3]*** Effective memory reference time = .75 × (5 + 200) + (1 – .75) × (5 + 200 + 200) ns = 255 ns
4.2	**Suppose 80% of a process page table is in TLB. TLB access time is 5ns and memory access time is 100ns. Determine the effective memory access time. If we want to limit the memory access time in 150ns, determine the page fault ratio. (Disk access time is 10 ms)** ***[In-course 2, 2008. Marks: 5 + 5]*** Effective memory access time = .80 × (5 + 100) + (1 – .80) × (5 + 100 + 100) = 125 ns Let p denote the page fault ratio. $\therefore (1 - p) \times 100$ ns $+ p \times (10 \times 10^6$ ns$) = 150$ ns $\therefore p = \frac{1}{199{,}998}$
4.3	**In a demand paging system, what will be the maximum tolerable page fault if a memory access is not larger than 40% of the original access time? Disk access time is 10^{10} times larger than memory access time.** ***[1999. Marks: 4]*** Let, memory access time = x ns $\therefore$ Maximum effective memory access time = $1.4x$ ns Let p denote the maximum tolerable page fault rate. According to the question, $(1 - p)x + p \times 10^{10}x = 1.4x$ $\Rightarrow 1 - p + 10^{10}p = 1.4$ [Dividing by x, as $x \neq 0$] $\therefore p = \frac{1}{25 \times 10^{10}}$
4.4	**Suppose we have a demand paging memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page fault rate for an effective access time of no more than 200 ns?** ***[2004. Marks: 3]*** We know, Effective Memory Access Time = (1 – p) × *memory access time* + p × *page fault service time* Where p is the page fault rate. $\therefore$ According to the question, $(1 - p) \times 100$ ns $+ p \times \{(.70 \times 20 + .30 \times 8) \times 10^6$ ns$\} \leq 200$ ns $\Rightarrow 100 - 100p + 16400000p \leq 200$ $\Rightarrow 16399900p \leq 100$ $\Rightarrow p \leq \frac{1}{163999}$ $\therefore$ The maximum acceptable page fault rate is $\frac{1}{163999}$.

4.5 **A system has five processes. Each memory reference takes 200 ns. To load page table from memory to TLB, it takes 400 ns (by using special hardware). On average, 80% of each process' page table can be loaded to the TLB. TLB access time is 50 ns. System is using round robin scheduling and time quantum is 10 ms. Within this time quantum, each process referenced 25 memory locations in 5 (working set model) pages. Determine the average memory access time.** ***[2006. Marks: 4]***

[Actually, এখানে round-robin, working set model এসব ডেটা ফাউ। Average memory access time বের করার জন্য এগুলো লাগবে না। কিন্তু একটা সমস্যা হয়ে গেছে এই কথা বলাতে যে, To load page table from memory to TLB, it takes 400 ns. কারণ, স্বাভাবিক নিয়মে এই অঙ্ক সলভ করার সময় TLB মিস হলে সেটাকে আমরা Memory access time দিয়ে গুণ করি (যেটা এখানে 200ns)। এখন, এখানে আমরা ২০০ দিয়ে গুণ করব, না কি ৪০০ দিয়ে, সেটা একটা সমস্যা। এটা পরীক্ষার হলে স্যারকে জিজ্ঞেস করে নিয়ে করাই ভাল। সেটা সম্ভব না হলে আমার মনে হয় ৪০০ ব্যবহার করাই উত্তম।]

Avg. Memory Access Time = $.80 \times (50 + 200) + (1 – .80) \times (50 + 400 + 200)$ ns = 330 ns

4.6 **Consider a demand-paging system with a paging disk that has an average access and transfer time of 20 milliseconds. Addresses are translated through a page table in main memory, with an access time of 1 microsecond per memory access. Thus, each memory reference through the page table takes two accesses. To improve this time, we have added an associative memory that reduces access time to one memory reference, if the page-table entry is in the associative memory.**

Assume that 80 percent of the accesses are in the associative memory and that, of the remaining, 10 percent (or 2 percent of the total) cause page faults. What is the effective memory access time?

Effective memory access time = $0.8 \times 1\ \mu s + 0.18 \times 2\ \mu s + 0.02 \times (2 + 20\times10^3)\ \mu s = 401.64\ \mu s$

4.7 **Consider the following reference string:**

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2

How many page faults occur for the following page replacement algorithms, for four frames? All frames are initially empty; as a result, unique pages will cost one fault each. ***[2004. Marks: 6]***

i) LRU
ii) FIFO
iii) Optimal

i) LRU:

1	**2**	**3**	**4**	**5**	**3**	**4**	**1**	**6**	**7**	**8**	**7**	**8**	**9**	**7**	**8**	**9**	**5**	**4**	**5**	**4**	**2**
1	1	1	1	5			5	6	6	6			6				5	5			5
	2	2	2	2			1	1	1	1			9				9	9			9
		3	3	3			3	3	7	7			7				7	4			4
			4	4			4	4	4	8			8				8	8			2

∴ Number of page faults = 13

i) FIFO:

1	**2**	**3**	**4**	**5**	**3**	**4**	**1**	**6**	**7**	**8**	**7**	**8**	**9**	**7**	**8**	**9**	**5**	**4**	**5**	**4**	**2**
1	1	1	1	5			5	5	5	8			8				8	8			2
	2	2	2	2			1	1	1	1			9				9	9			9
		3	3	3			3	6	6	6			6				5	5			5
			4	4			4	4	7	7			7				7	4			4

∴ Number of page faults = 13

i) OPT:

1	2	3	4	5	3	4	1	6	7	8	7	8	9	7	8	9	5	4	5	4	2
1	1	1	1	1				6	7	7			7					4			4
	2	2	2	5				5	5	5			5					5			5
		3	3	3				3	3	8			8					8			2
			4	4				4	4	4			9					9			9

∴ Number of page faults = 11

4.8 **How many page faults will occur for the following mentioned page replacement algorithms for the page reference string R in a memory system with 4 frames?** ***[2003. Marks: 4.5]***
How many page faults with 3 frames? ***[2000. Marks: 5]***

R = 1, 2, 3, 4, 2, 1, 5, 6, 7, 2, 1, 2, 3, 5, 7, 6, 3, 2, 1, 2, 3, 6

i) FIFO
ii) OPT
iii) LRU

4.9 **From the following page reference string, determine the number of page faults occurred in a system, which consists of five memory frames for page replacement algorithm LRU and FCFS.** ***[1999. Marks: 4]***

0, 1, 3, 0, 1, 5, 6, 2, 9, 5, 3, 2, 0, 1, 2, 3, 16, 17, 10, 9, 5, 15, 20, 2

4.10 **Consider the following list of virtual memory references. Values given should be read from left to right then top to bottom:**

020	021	047	104	105	207	210	211	024
025	026	363	164	423	424	563	642	643
020	021	022	047	141	725	726	727	730
564	565	027	030	031	365	366	367	426
427	750	751	105	106	107	254	255	370

Suppose that the process executing the program has four page frames available to it, and the size of each page is 100 (i.e., addresses 0-99 are on page 0 etc.).

Using each of the page replacement algorithms listed below; trace the sequence of page faults generated by the execution of this program, assuming that it starts with no pages in main memory. Indicate which pages are swapped in and out and when this occurs. ***[2002. Marks: 8]***

i) First-In First Out
ii) Least Recently Used
iii) Least Frequently Used

Note:

1. First, convert the memory address references into page numbers by dividing the address by page frame size and take the integer value. For example, [020 / 100] = [0.2] = 0.
2. LFU is just an implementation of LRU. So, LFU and LRU are the same.

4.11 **Followings are the snapshot of memory references (virtual addresses):**

125, 3890, 236, 1420, 1345, 2100, 312, 567, 89, 2345, 6780, 3120, 82, 61234

System has three 512-byte memory frames, and using one byte flag where 1 is inserted in MSB by shifting the flag one bit right; and 0 is inserted in MSB (right shifting) if it is not referenced. This operation is done for every memory references. Pages in memory frames are replaced based on lowest flag value. Determine the number of page faults and compare it with optimal page replacement algorithm. ***[2006. Marks: 3]***

The system described in the question is an implementation of the LRU algorithm. So, we have to determine the number of page faults using the LRU algorithm and then compare it with the optimal page replacement algorithm.

First, we convert the memory address references into page numbers:

125	**3890**	**236**	**1420**	**1345**	**2100**	**312**	**567**	**89**	**2345**	**6780**	**3120**	**82**	**61234**
0	7	0	2	2	4	0	1	0	4	13	6	0	119

Now,

Page References	**LRU**			**Optimal**		
0	0			0		
7	0	7		0	7	
0						
2	0	7	2	0	7	2
2						
4	0	4	2	0	4	2
0						
1	0	4	1	0	4	1
0						
4						
13	0	4	13	0	13	1
6	6	4	13	0	6	1
0	6	0	13			
119	6	0	119	119	6	1

[Note: পেজ রেফারেন্স-এর সংখ্যা বেশি দেওয়া থাকলে পরীক্ষার খাতায় পাশাপাশি লিখতে গেলে আঁটবে না। সেজন্য কিভাবে নিচে-নিচে লিখে করতে পার, সেটা দেখায় দিলাম।]

CHAPTER 5
INPUT / OUTPUT

Questions

5.1	**What is RAID? What are the significant properties of a stable storage system? Discuss.** ***[2002. Marks: 4]*** RAID is a technology which makes use of two or more hard drives in order to improve performance, reliability or create larger data volumes.
5.2	**Describe different RAID levels.** ***[2004. Marks: 4]*** *See Theory 5.18.*
5.3	**"Raid level 01 or 10 increase disk access performance and availability", discuss with a suitable example.** ***[2007. Marks: 4]***
5.4	**Why might a system use interrupt-driven I/O to manage a single serial port, but polling I/O to manage a front-end processor such as a terminal concentrator?** ***[2002. Marks: 3]***
5.5	**What is meant by device-independent I/O software?** ***[1999. Marks: 2]*** Device-independent IO software performs the I/O functions that are common to all devices and provides a uniform interface to the user-level software.
5.6	**Nowadays operating system uses device-independent I/O software, why?** ***[2006. Marks: 2]*** Because device-independent IO software performs the I/O functions that are common to all devices and provides a uniform interface to the user-level software; hence the programming becomes very easy.
5.7	**What are the impact of buffering in data transfer between I/O devices and operating system kernel?** ***[2006. Marks: 2]***
5.8	**What do you understand by disk formatting? Why boot block is necessary?** ***[1999. Marks: 3]***

Exercises

5.1	**Suppose that a disk drive has 5000 cylinders numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order is:** **85, 1465, 920, 1784, 948, 1510, 1025, 1745, 128** **Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk scheduling algorithms?** **i. SSTF** **ii. SCAN** **iii. C-SCAN** **iv. C-LOOK** ***[2006. Marks: 4]*** **v. LOOK** ***[2004. Marks: 6]*** **vi. FIFO** ***[2002. Marks: 6]*** The requests *along with the initial head position* sorted in ascending order are as follows: 85, 128, 143, 920, 948, 1025, 1465, 1510, 1745, 1784

	i. SSTF: [*Steps:* 1. Sort the queue *along with the initial head position* in ascending order. 2. Locate the start position (143 in this exercise) and determine whether its left or right value is closer to it. (in this case, it's 128 – the value to the left.) 3. Find the distance with the closer value (|143 – 128| = 15) and then delete the start position. 4. Repeat steps 2-3 starting from this new value (128).] Total distance = |143 – 128| + |128 – 85| + |85 – 920| + |920 – 948| + |948 – 1025| + |1025 – 1465| + |1465 – 1510| + |1510 – 1745| + |1745 – 1784| = 1757 **ii. SCAN:** Total distance = |143 – 920| + |920 – 948| + |948 – 1025| + |1025 – 1465| + |1465 – 1510| + |1510 – 1745| + |1745 – 1784| + |1784 – 4999| + |4999 – 128| + |128 – 85| = 9770 **iii. C-SCAN:** Total distance = |143 – 920| + |920 – 948| + |948 – 1025| + |1025 – 1465| + |1465 – 1510| + |1510 – 1745| + |1745 – 1784| + |1784 – 4999| + |4999 – 0| + |0 – 85| +|85 – 128| = 9983 **iv. C-LOOK:** Total distance = |143 – 920| + |920 – 948| + |948 – 1025| + |1025 – 1465| + |1465 – 1510| + |1510 – 1745| + |1745 – 1784| + |1784 – 85| + |85 – 128| = 3383 **v. LOOK:** Total distance = |143 – 920| + |920 – 948| + |948 – 1025| + |1025 – 1465| + |1465 – 1510| + |1510 – 1745| + |1745 – 1784| + |1784 – 128| + |128 – 85| = 3340 **vi. FIFO:** Total distance = |143 – 85| + |85 – 1465| + |1465 – 920| + |920 – 1784| + |1784 – 948| + |948 – 1510| + |1510 – 1025| + |1025 – 1745| + |1745 – 128| = 7067
5.2	**Determine the total head movement for algorithms SSTF and C-LOOK for the following requests (in track number) where the current position of head is on track number 0. *[1999. Marks: 3 + 3]*** **98, 181, 23, 65, 122, 14, 72, 36, 26, 67** The requests *along with the initial head position* sorted in ascending order are as follows: 14, 23, 26, 36, 65, 67, 72, 98, 122, 181 **SSTF:** Total head movement = |0 – 14| + |14 – 23| + |23 – 26| + |26 – 36| + |36 – 65| + |65 – 67| + |67 – 72| + |72 – 98| + |98 – 122| + |122 – 181| = 181 **C-LOOK:** Total head movement is the same as SSTF, i.e., 181.
5.3	**A disk has an average seek time of 5ms, a rotational speed of 15,000 rpm, and 500 sectors per track. What is the average access time to read a single sector? What is the expected time to read 500 contiguous sectors on the same track? What is the expected time to read 500 sectors scattered over the disk? *[2002. Marks: 3]*** Given, Avg. seek time, $T_s = 5\text{ms} = 5\times10^{-3}$ s

	Rotational speed, r = 15000 rpm = $\frac{15000}{60}$ rps = 250 rps Sectors per track = 500 $\therefore$ Avg. access time to read a single sector = $T_s + \frac{1}{2r} + \frac{1}{r \times 500}$ $= 5\times10^{-3} + \frac{1}{2\times250} + \frac{1}{250\times500}$ $= 7.008\times10^{-3}$ s = 7.008 ms $\therefore$ Expected time to read 500 contiguous sectors on the same track = 7.008 ms + $\frac{1}{250\times500} \times 499$ s = 7.008 ms + 3.992 ms = 11 ms $\therefore$ Expected time to read 500 sectors scattered over the disk = 7.008 ms × 500 = 3.504 s
5.4	**Consider a disk with a mean seek time of 8ms, a rotational rate of 15,000 rpm, and 262,144 bytes per track. What are the access times for block sizes of 1 KB, 2 KB, and 4 KB, respectively?** Given, Avg. seek time, T_s = 8ms = 8×10^{-3} s Rotational rate, r = 15000 rpm = $\frac{15000}{60}$ rps = 250 rps Number of bytes per track, N = 262144 $\therefore$ Access time for 1KB block = $T_s + \frac{1}{2r} + \frac{1024}{r\times N} = 8\times10^{-3} + \frac{1}{2\times250} + \frac{1024}{250\times262144}$ = 10.0156 ms $\therefore$ Access time for 2KB block = $T_s + \frac{1}{2r} + \frac{1024}{r\times N} = 8\times10^{-3} + \frac{1}{2\times250} + \frac{1024\times2}{250\times262144}$ = 10.03125 ms $\therefore$ Access time for 4KB block = $T_s + \frac{1}{2r} + \frac{1024}{r\times N} = 8\times10^{-3} + \frac{1}{2\times250} + \frac{1024\times4}{250\times262144}$ = 10.0625 ms

CHAPTER 6
FILE SYSTEMS

Questions

6.1 **What do you understand by file system? What are the different layers of a standard file system?** ***[2000. Marks: 2 + 2]***

The part of the operating system dealing with files is known as the file system.

Files are managed by the operating system. How they are structured, named, accessed, used, protected, and implemented are the issues of file management.

The layers of a standard file system are as follows:

3. Super Block
4. Free Space Management
5. Inodes
6. Files and Directories

6.2 **Describe how traditional Unix systems maintain the list of free disk blocks.** ***[2004. Marks: 4]***

Describe the bitmap technique. See Theory 6.12.

6.3 **What is bit vector? Consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26 and 27 are free and the rest of the blocks are allocated. Construct the free-space bitmap.** ***[2006. Marks: 3]***

Bit vector is a technique to locate free blocks in a disk. Each block in the disk is represented by 1 bit. If the block is free, the bit is 1, otherwise, the bit is 0.

Free-space bitmap for the given blocks: 00111100111111000110000001110000…

6.4 **Suppose tree-directory structure and acyclic-graph directory structure is implemented in two different file systems to organize files and directories. Which one would be better and why?** ***[2000. Marks: 4]***

OR, What are the advantages of directory structure in acyclic graph over tree and graph directory structure? ***[1999. Marks: 4]***

OR, Compare the data structures tree and acyclic graph for directory implementation. ***[2002. Marks: 3]***

The acyclic-graph directory structure would be better than tree-directory structure. Because, when several users are working together on a project, they often need to share files. As a result, it is often convenient for a shared file to appear simultaneously in different directories belonging to different users.

Again, the graph directory structure might contain loops which could be costly to detect. So, acyclic-graph directory structure is better than graph directory structure, too.

Comparison of tree and acyclic graph directory implementation:

Tree Directory Structure	Acyclic Graph Directory Structure
A file can appear under only one directory.	A file can appear under multiple directories.
Sharing files among users is difficult.	Sharing files among users is very easy.
Less complication in managing files than acyclic graph structure.	Complicates managing files as hard link counts need to be processed.

6.5	**What is virtual file system (VFS)? How multiple file systems are handled by virtual file system?** ***[2006. Marks: 3]*** *See Theory 6.13.*
6.6	**What is indexed-allocation method? Is multilevel indexed allocation is a better solution for applications that need files with very large size? Explain your answer.** ***[2000. Marks: 2 + 4]*** Indexed allocation method is a method of implementing files. In this method, each file has its own index block, which is an array of disk-block addresses. The i^{th} entry in the index block points to the i^{th} block of the file. Yes, multilevel indexed allocation is a better solution for applications that need files with very large size. If there were a single index, it would need to be contiguous; and adding records to the file from time to time would require a large free space to fit in the index. That would create fragmentation and other problems from which the sequential file implementation suffers. That's why using multilevel indexed allocation would yield better space optimization for disks containing very large files.
6.7	**Compare linked file allocation method with indexed allocation method.** ***[2004. Marks: 3]***

Linked File Allocation	Indexed Allocation
Better for sequential files, but random access is extremely slow.	Random access is fast, sequential access is not as fast as random access.
The amount of data storage in a block is no longer a power of two because the pointer takes up a few bytes.	The amount of data storage in a block is a power of two.

6.8	**Write a short note on – journaling a file system.** ***[2004. Marks: 4]*** *See Theory 6.17.*
6.9	**Why access protection is necessary in file systems? How protection can be implemented by the identity of the users?** ***[2000. Marks: 2 + 4]*** Access protection is necessary in file systems for protecting a user's data from theft and also to provide privacy. *For the second part of the question, see Theory 6.15.*
6.10	**How many disk operations are needed to fetch the inode for the file** ***/usr/ast/courses/os/handout.t*****? Assume that the inode for this root directory is in memory, but nothing else along the path is in memory. Also assume that all directories fit in one disk block.** The following disk reads are needed: directory for / i-node for */usr* directory for */usr* i-node for */usr/ast* directory for */usr/ast* i-node for */usr/ast/courses* directory for */usr/ast/courses* i-node for */usr/ast/courses/os* directory for */usr/ast/courses/os* i-node for */usr/ast/courses/os/handout.t* In total, 10 disk reads are required.

Exercises

6.1 **Assume you have an inode-based file system. The file system has 512 byte blocks. Each inode has 10 direct, 1 single indirect, 1 double indirect and 1 triple indirect block pointers. Block pointers are 4 bytes each. Assume the inode and any block free list is always in memory. Blocks are not cached.**

i) **What is the maximum file size that can be stored before**
  1. **the single indirect pointer is needed?**
  2. **the double indirect pointer is needed?**
  3. **the triple indirect pointer is needed?**

ii) **What is the maximum file size supported?**

iii) **What is the number of disk block reads required to read 1 byte from a file**
  1. **in the best case?**
  2. **in the worst case?** ***[2002. Marks: 3 + 1 + 2]***

Solution:

i) 1. 10×512 bytes = 5 KB
   2. $10 \times 512 + (512 / 4) \times 512$ bytes = 69 KB
   3. $10 \times 512 + (512 / 4) \times 512 + (512 / 4)^2 \times 512$ bytes ≈ 8 MB

ii) $10 \times 512 + (512 / 4) \times 512 + (512 / 4)^2 \times 512 + (512 / 4)^3 \times 512$ bytes ≈ 1 GB

iii) 1. 1
   2. 4

6.2 **Consider a Unix file system with 12 direct pointers, 1 indirect pointer, 1 double-indirect pointer, and 1 triple-indirect pointer in the inode. Assume that the disk blocks are 8Kbytes and that each pointer to a disk block requires 4 bytes.**

i) **What is the largest possible file size that can be supported with this design?**

ii) **How many disk reads will this file system require to read block 14 of the file named /*a*? Assume that nothing relevant is in the file cache (e.g. no inodes and no data blocks), and the root directory contains very few entries (i.e., is one block long). Describe each disk read.** ***[2004. Marks: 2 + 3]***

i) $12 \times 8 + (8192 / 4) \times 8 + (8192 / 4)^2 \times 8 + (8192 / 4)^3 \times 8$ KB ≈ 64 TB

ii) The following disk reads are needed:

directory for /
inode for /*a*
single-indirect block of /*a*
block 14 of /*a*

In total, 4 disk reads are required.

6.3 **Consider a file currently consisting of 100 records of 400 bytes. The file system uses fixed blocking, i.e., one 400 byte record is stored per 512 byte block. Assume that the file control block (and the index block in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one record, the following conditions hold. In the contiguous-allocation case, assume that there is no room to grow at the beginning, but there is room to grow at the end of the file. Assume that the record information to be added is stored in memory.**

i) **The record is added at the beginning.**
ii) **The record is added in the middle.**
iii) **The record is added at the end.**
iv) **The record is removed from the beginning.**
v) **The record is removed from the middle.**
vi) **The record is removed from the end.** ***[2002. Marks: 1.5 each]***

[Note: In linked allocation, the FCB (File Control Block) contains the *starting* block number as well as the *ending* block number of a file. Hence, to access the last block of a file, only 1 read is needed as the ending block number is already known from FCB.]

Contiguous	(i)	201	(100 reads + 100 writes) for shifting the blocks + 1 write for the new block
	(ii)	101	(50 reads + 50 writes) for shifting the blocks of the second half + 1 write for the new block
	(iii)	1	1 write for the new block
	(iv)	198	(99 reads + 99 writes) for shifting the remaining blocks
	(v)	98	(49 reads + 49 writes) for shifting the remaining blocks
	(vi)	0	
Linked	(i)	1	1 write for the new block
	(ii)	52	50 reads for finding the link info of 50th block + 1 write for updating the link info of 50th block + 1 write for the new block
	(iii)	3	(1 read + 1 write) for updating the link information of the last block + 1 write for the new block
	(iv)	1	1 read for finding the link info of 2nd block [to update the FCB]
	(v)	52	51 reads for finding the link info of 50th and 51st block + 1 write for updating the link info of 50th block [by writing the link info of 51st block into 50th block]
	(vi)	100	(99 reads + 1 write) for updating the link info of 99th block [by putting a -1 as the next block number in the link field]
Indexed	(i)	1	1 write for the new block
	(ii)	1	1 write for the new block
	(iii)	1	1 write for the new block
	(iv)	0	
	(v)	0	
	(vi)	0	

CHAPTER 9
SECURITY

Questions

9.1 **What is virus? How virus code can be extracted?** ***[2006. Marks: 2]***

A virus is a program that can reproduce itself by attaching its code to another program. In addition, the virus can also do other things in addition to reproducing itself.

Virus code can be extracted by scanning and looking for any of the viruses in the database of known viruses. Additionally, fuzzy search can be used for this purpose.

CHAPTER
MIPS SYSTEM CALLS

Roadmap

In this chapter, we'll learn how system calls are handled by OS/161 operating system which is based on MIPS processor. First, we'll see how MIPS handles any type of exception. Then we'll learn how system calls are handled by generating a *syscall* exception.

Theories

1

Coprocessor 0 (CP0)

In MIPS R2000/R3000 processor, the processor control registers (such as exception management registers) are located in CP0.

CP0 Exception Management Registers and Their Contents

There are 4 registers that deal with exceptions:

1. **c0_cause:** stores cause of the most recent exception.
2. **c0_status:** stores the current status of the CPU.
3. **c0_epc** [*Exception Program Counter*]**:** stores the address of the instruction that caused the exception. In other words, stores the return address (i.e., where to restart execution) after handling the exception.
4. **c0_badvaddr** [*Bad Virtual Address*]**:** Stores the address accessed that caused the exception. This is used if an exception occurs because of a bad (illegal / restricted) memory address reference. In that case, the address referenced is stored here.

The c0_status Register

[Note that we're interested only in the fields with yellow background. We don't need to know what the other fields contain.]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CU3	CU2	CU1	CU0	0		RE	0		BEV	TS	PE	CM	PZ	SwC	IsC

15	8	7	6	5	4	3	2	1	0
IM		0		KUo	IEo	KUp	IEp	KUc	IEc

KU:
- 0: Kernel mode
- 1: User mode

IE:
- 0: All interrupts disabled (masked)
- 1: All interrupts enabled

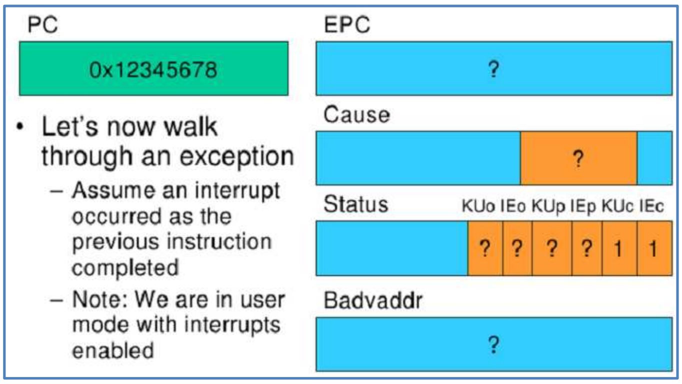
c, p, o: Current, Previous, Old

The c0_cause Register

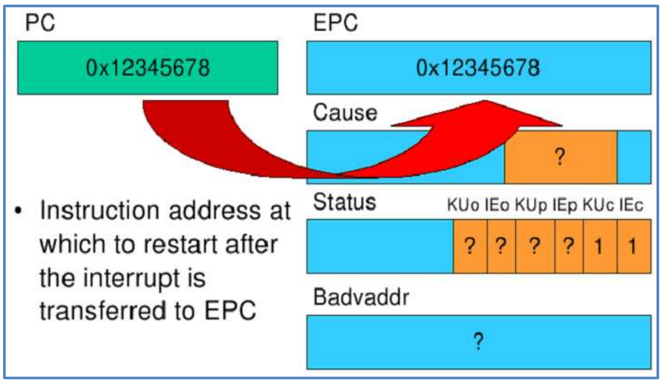
31	30	29	28	27	16	15	8	7	6	2	1	0
BD	0	CE		0		IP		0	ExcCode		0	

ExcCode: The code number of the exception taken.

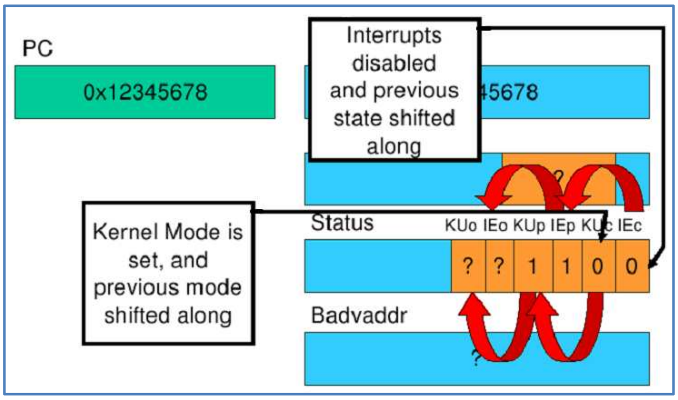
2 Hardware Exception Handling



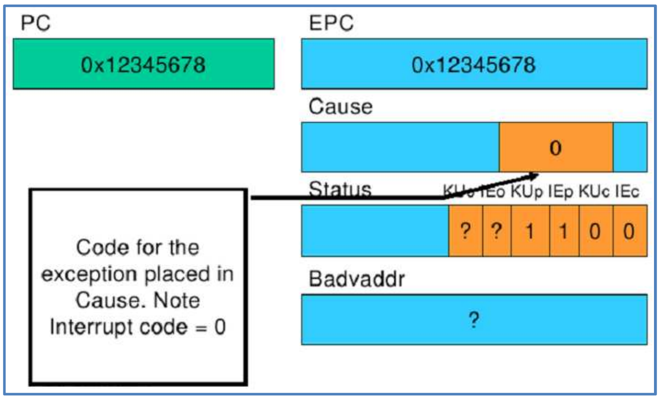
1



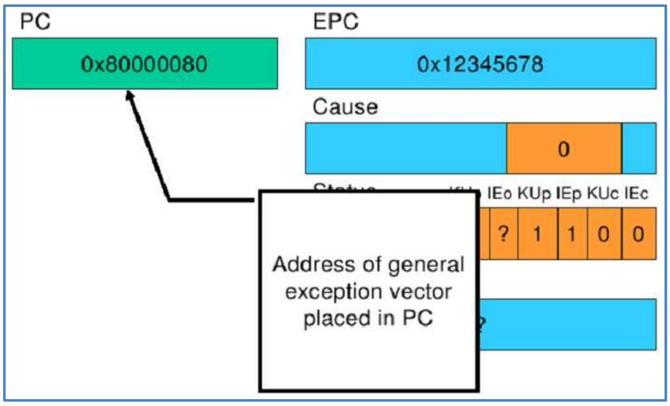
2



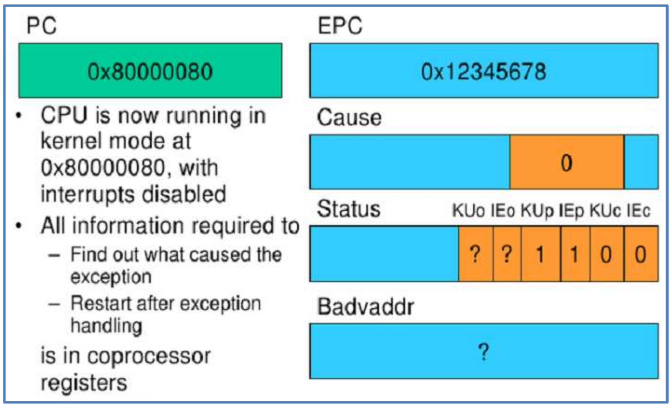
3



4



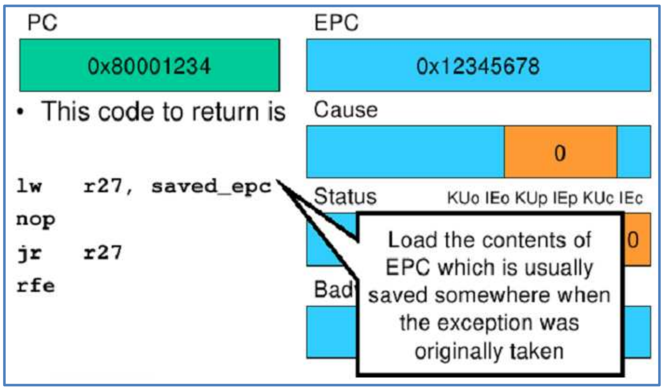
5



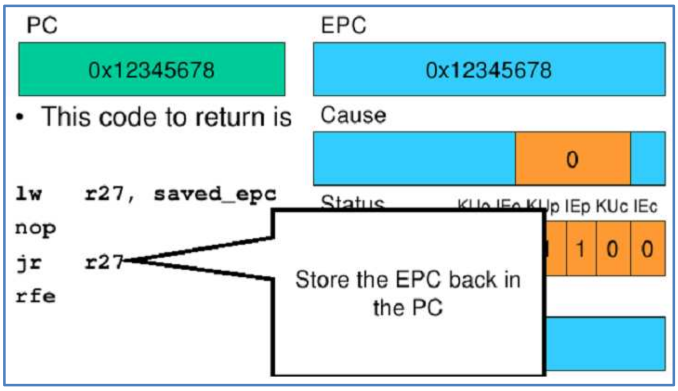
6

Returning From an Exception

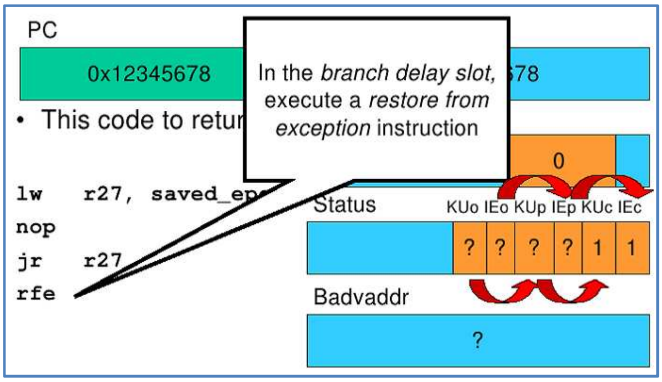
For now, let's ignore how the exception is actually handled and how user-level registers are preserved. Let's simply look at how we return from the exception.



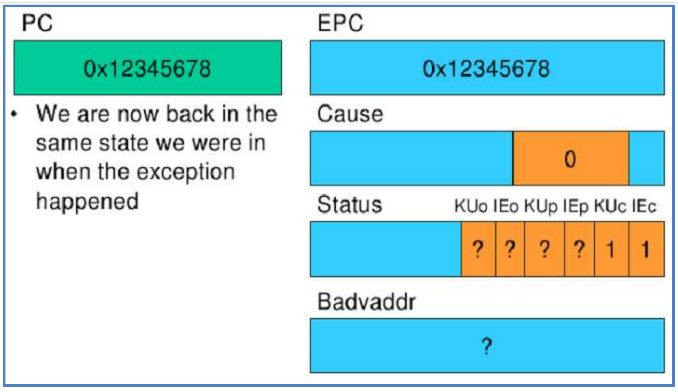
1



2



3



4

3 **MIPS System Calls**

- System calls are invoked via a *syscall* instruction.
  - The *syscall* instruction causes an exception and transfers control to the general exception handler.
  - A convention (an agreement between the kernel and the applications) is required as to how user-level software indicates
    - Which system call is required
    - Where its arguments are
    - Where the results should go
- OS/161 uses the following conventions:
  - Arguments are passed and returned via the normal C function calling convention.
  - Additionally,
    - Register v0 contains the system call number.
    - On return, register a3 contains
      - 0: if success, and in this case, v0 contains successful result.
      - Not 0: if failure, and in that case, v0 contains the error number.
        - However, when returning from the system call, the content of v0 register is copied to the `errno` global variable, and -1 is returned in v0.

4 **User-Level System Call Walk Through**

Consider the system call *read*:

```
int read(int fd, void *buffer, int size)
```

Note that there are three arguments and one return value.

The steps in executing this system call are as follows:

1. First, appropriate registers are preserved (for example, the arguments are loaded into registers a0~a2, return address is loaded into register *ra* etc.).
2. The syscall number (5) is loaded into v0.
3. A jump (not jump and link) to the common syscall routine is made.
4. In the common syscall routine, a syscall exception is generated.
5. Instruction address at which to restart after the interrupt is transferred from PC to EPC register. Note that both the KUc and IEc bits in c0_status register contain 1 as we are still in user mode with interrupts enabled.
6. The interrupts are disabled and previous state shifted along in the c0_status register. Thus, the KUp and IEp bits now contain 1s and KUc and IEc bits 0s.
7. Code for the exception (8 – code for *syscall*) is placed in the ExcCode field in c0_cause register.
8. Address of the syscall exception is loaded into PC and the exception handling routine is executed, which in turn completes the tasks of the *read* procedure.
9. If the system call succeeds, then program execution returns from the common syscall routine.

	10. If the system call fails, then error code is stored in the errno variable and -1 is returned to v0. 11. Program execution returns to the location after where read() was called. *[Note that steps 5 to 8 are none other than the steps mentioned in the section "Hardware Exception Handling" in Theory 2.]*

The Above Steps Depicted in Code Segments [*Provided for your understanding only*]

Code fragment calling the read function:

```
... ... ... ... ... ...
40012c:   0c1001a3    jal   40068c <read>
... ... ... ... ... ...
```

Code fragment of the read function:

```
0040068c <read>:
    40068c:     08100190    li    v0, 5                   //step 2
    400690:     24020005    j     400640 <__syscall>      //step 3
```

Code fragment of the syscall function:

```
00400640 <__syscall>:
    400640:     0000000c    syscall                          //step 4
    400644:     10e00005    beqz  a3, 40065c <__syscall+0x1c>  //step 9
    400648:     00000000    nop
    40064c:     3c011000    lui at, 0x1000                   //step 10
    400650:     ac220000    sw    v0, 0(at)
    400654:     2403ffff    li    v1, -1
    400658:     2402ffff    li    v0, -1
    40065c:     03e00008    jr    ra                         //step 11
    400660:     00000000    nop
```

Questions

1	**Determine the steps to the service of system call such as read. Write your answer in details (including c0_status register values). *[In-course 2, 2009. Marks: 5]*** *See Theory 4* (*Section "User-Level System Call Walk Through"*).
2	**"syscall" is an instruction in MIPS processor family to generate unconditional exception which can be useful for system call. For a system call (such as fork, open, read etc.) how this instruction can be useful? Write your answer describing invoking and returning from system call. Assume that address of syscall exception vector is 0x9000215 and current program counter (PC) value is 0x1000234. *[2006. Marks: 5]*** *See Theory 2.*
3	**Suppose a kernel library function (system call) has starting address at 0x4EFA120A (which is in kernel address space). How this library function can be used (or called) by the user program? What happens to c0_status register before execution, in execution and after execution of the function? *[In-course 1 & 2, 2008. Marks: 5]*** The kernel library function can be used by the user program by using the *syscall* instruction and providing it with the number of that library function. *For the next part of the question, see Theory 2.*
4	**How arguments like array, linked-list or more than four arguments can be passed from user mode to kernel mode using system call? Assume that a0~a3 can hold only four arguments and v0 holds the system call identifier. *[2006. Marks: 3]*** The first four arguments are passed via registers a0~a3. The remaining arguments are passed via pushing them onto the stack. (…কার স্ট্যাক? প্রসেস স্ট্যাক? না কি কারনেল স্ট্যাক? কনফিউজড...☹)