Department: CSE

Course No: CSE 3213

Examination: Final

ID: 170204004

Program: B.SC in CSE

Course Title: Operating System

Semester: Spring 2020

Sign and date: Rukaiya
24.05.21

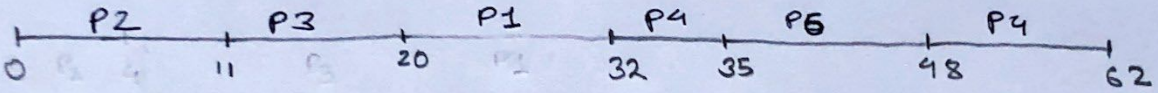
ID: 1702040004
Course: CSE 3213
Sign: Rukaiya

Answer to Question 2.c

- Using Shortest Remaining Time Next:

Process	A.T	CPU Time
P1	5	12
P2	0	11
P3	4	9
P4	12	17
P6	35	13

Gnatt chart:

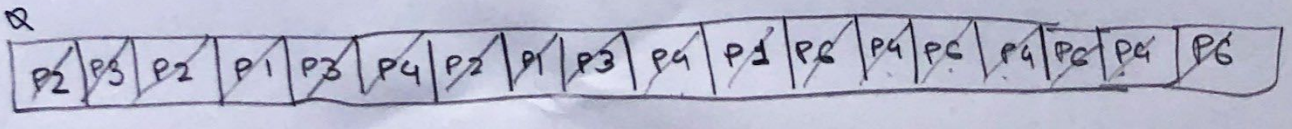


Avg. waiting time = (0 + 7 + 15 + 33 + 0) / 5
= 11

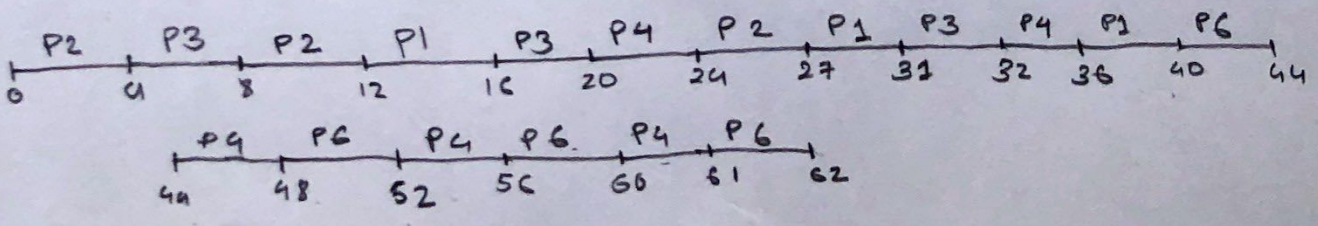
Avg. response time = (15 + 0 + 7 + 32 - 12 + 0) / 5
= 8.4

- Using Round-Robin;

P1	5	12/8/4/0	q=4
P2	0	11/7/3/0	
P3	4	9/5/1/0	
P4	12	17/13/9/5/1/0	
P6	35	13/9/5/1/0	



Gnatt chart:



ID: 170204004
Course: CSE 3213
Sign: Rukaiya

~~Avg waiting time = {(20-5)+0+(11-4)+(32-12)+(48-35)~~

Total Waiting time = (12-5)+(27-16)+(36-31)+(8-4)+(24-12)
+(4-4)+(16-8)+(31-20)+(20-12)+(32-24)+(44-36)
+(52-48)+(60-56)+(40-35)+(48-44)+(56-52)
+(61-60)
= 104

Avg waiting time = 104/5 = 20.8

Avg Response time = {(12-5)+0+0+(20-12)+(40-35)}/5 = 4

Now, if we compare these two algorithms it can be said that while using RR the response time is lower than that of SRTN. on the other hand, the avg. waiting time is higher for RR. So if we required the processes to respond as soon as possible when called, we should opt ~~w~~ for RR algorithm. ~~Similarly if we require the processes to be On~~ Generally, we should use algorithms that gives us lower Avg waiting time. Therefore, in this case SRTN would perform better.

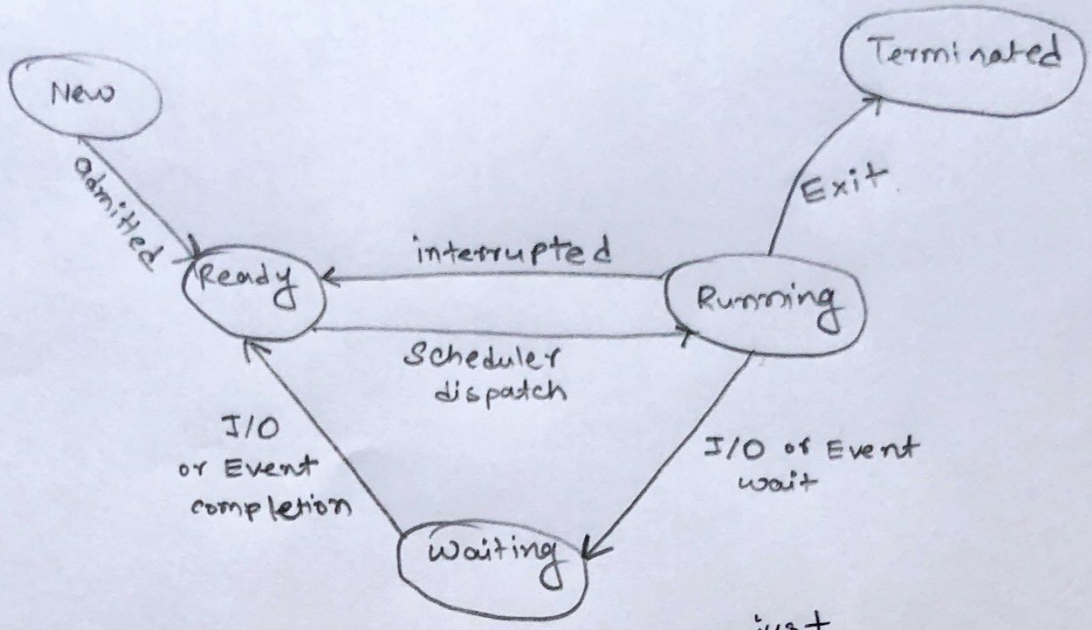
Answer to question 2.b

The dispatcher module gives control of the CPU to the process selected ~~b~~ by the short-term scheduler.
The function of dispatcher would include:

1. Context switching = when a process needs to be taken off the CPU and into the ready queue and also when a process needs to be taken to the CPU from the ready queue, ~~the~~ its data ~~d~~ needs to be stored ~~and r~~ in the PCB while the other process is being loaded. Dispatcher takes care of it. It saves and restores the processes.
2. Switching to User mode: Dispatcher switches the processes to user mode. All ~~low level~~ operating system processes run on kernel mode including dispatcher. When it is done storing and restoring processes it switches ~~t~~ to User mode so the user application can run.
3. Addressing: It jumps to the proper location ~~in~~ in the newly loaded program. The PC points toward the ~~ne~~ next processes that is to be executed. The dispatcher is responsible for ~~that~~ addressing the address.

ID: 170204004
Course: CSE3213
Sign: Rukaiya

The diagram below shows the state transition diagram of a process.



1. New State: A process that has been just created but has not been yet to the queue of execution processes by the OS. Each of the new operation which is requested to the system is known as the new born process.

2. Ready state: When the new born process is ready to be executed but is waiting for the CPU to be assigned then it is ~~called~~ in the ready-state. After completion of the I/O or an event the process will be on ready state which means it will wait for the processor to execute.

3. Running State: The process currently being executed.

4. Waiting or Blocked state: A process that cannot execute until some event occurs or an I/O completion is in this state. In this state, the process is stored out of the memory and will again go to the Ready state when the I/O or event is completed.

5. Terminated State: After completion of the process, it will be terminated by the CPU. This state is called terminated-state. In this state, the processor will also deallocate the memory which was allocated to the process.

Answer to Question 5.a

In paging mechanism, the RAM or memory is divided into fixed sized partitions called frames. Similarly the process that needs to be loaded is divided into partions called pages in such a way that ~~they have~~ pages' size is equal to the frames' size. Now, the pages are loaded into the frames and their addresses are mapped on to Page table. A page fault occurs when the OS tries to search for a page location in the page table and find it empty. ~~In this way,~~ In the page table, there is a Valid-invalid bit. If it is v, then the OS knows that it is valid and a page fault would not occur. And if the bit is set to i, page fault will occur. In this way, OS determines whether a page fault will occur or not.

Given that,

RAM = 16 MB

Page = 16 KB

Process = 32 MB

No. of total pages = $\frac{32\text{ MB}}{16\text{ KB}}$ = 2048 byte = 11 bits

" " " frame = $\frac{16\text{ MB}}{16\text{ KB}}$ = 1024 byte = 10 bits

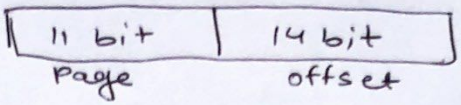
Here, address space in each frame/page = 16 × 1024

= 16384 bytes

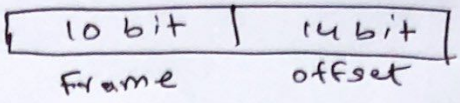
= 14 bits

ID: 1702041004
Course: CSE3213
Sign : Rukaiya

∴ logical address:



∴ Physical address:



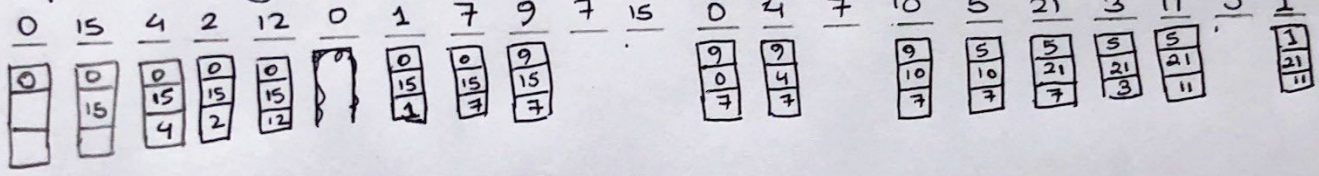
Here,
Page No = 200
Frame No = 28
offset = 15200

logical Address = 00011001000 | 11101101100000

Physical Address = 0000011100 | 11101101100000

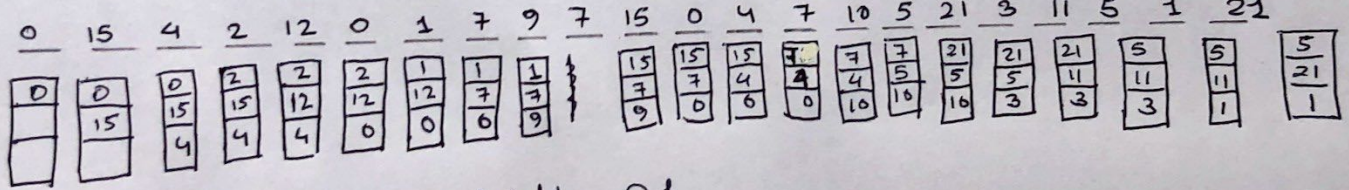
Answer to question 5.b

• Optimal Page Replacement:



∴ total no. of page fault = 16

• LRU:



∴ total no. of page fault = 21

Now if we compare the number of page faults, LRU has the greater number. Therefore we can say that, Optimal page replacement algorithm would give the best result in this case.

ID: 170204004
Course: CSE3213
Sign: Rukaiya

Ans to Question 5.c

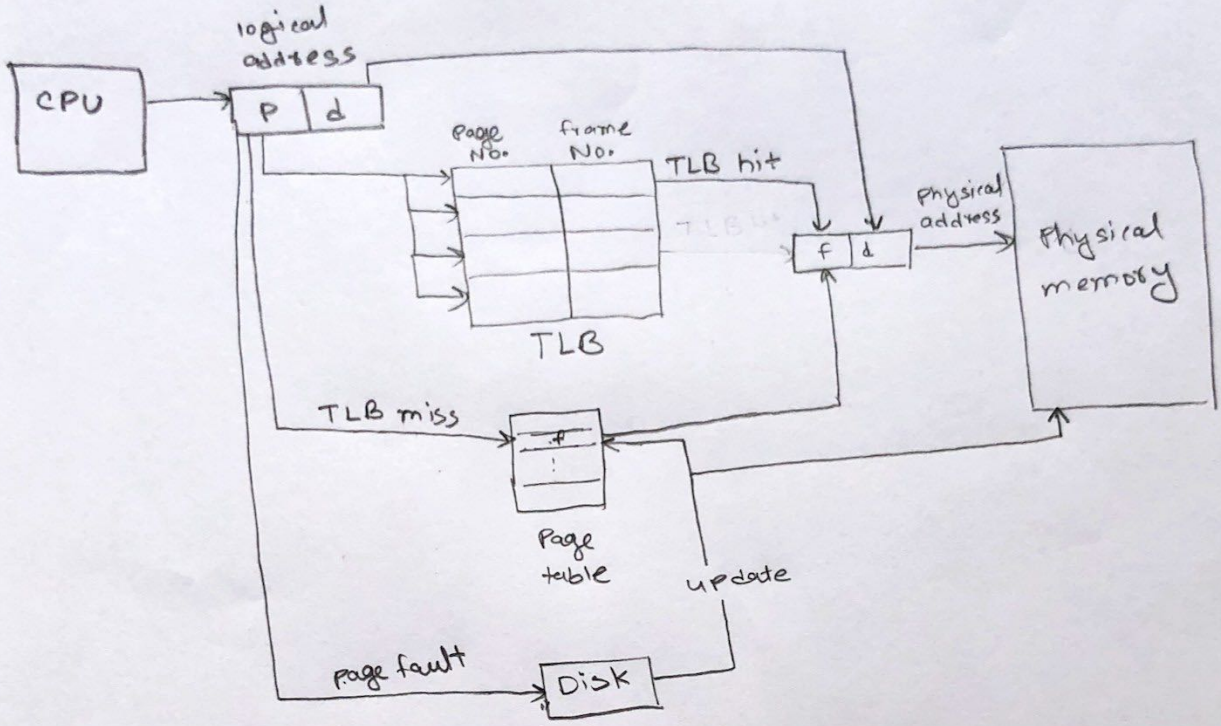


Fig: Diagram for address translation process with TLB

Answer to question 2.a

I+ is expected of a CPU to be busy for the most time, ideally 100%. so the CPU cycles are not wasted. On a real system, CPU utilization ranges from 50%-90%. The OS wants to optimize the average values of ~~e~~ things like, Maximum CPU utilization & throughput. So maximum work can be done and the CPU is not left idle.
A solution is provided here for a race free condition for the given case:

ID: 170204004
Course: CSE3213
sign : Rukaiya

```
Reader( ){
while (true){
down (mutex) //acquire lock
    read_count++
 if (read_ count==1)
        down (db);
 up(mutex); //release lock

 Read_DB( ){
 down (mutex)
   read_count--;
 if(read_count==0)
     up(db);
 up (mutex);
 }
 }
writer (i)
{ while (true){
      prepare Data (i);
 down (writermutex);
    writer_ count++;
    if (writer_ count==1)
      down (db)
    up (writer mutex);
    down (Exclusivelock)
    writer Data(i);
    up(Exclusive lock)
     down (writer mutex);
     writer_ cout--;
     up (writer mutex)
     if. (writer_ count==0
        up(db);
  {
 {
```

ID:1702040004

Course: CSE3213

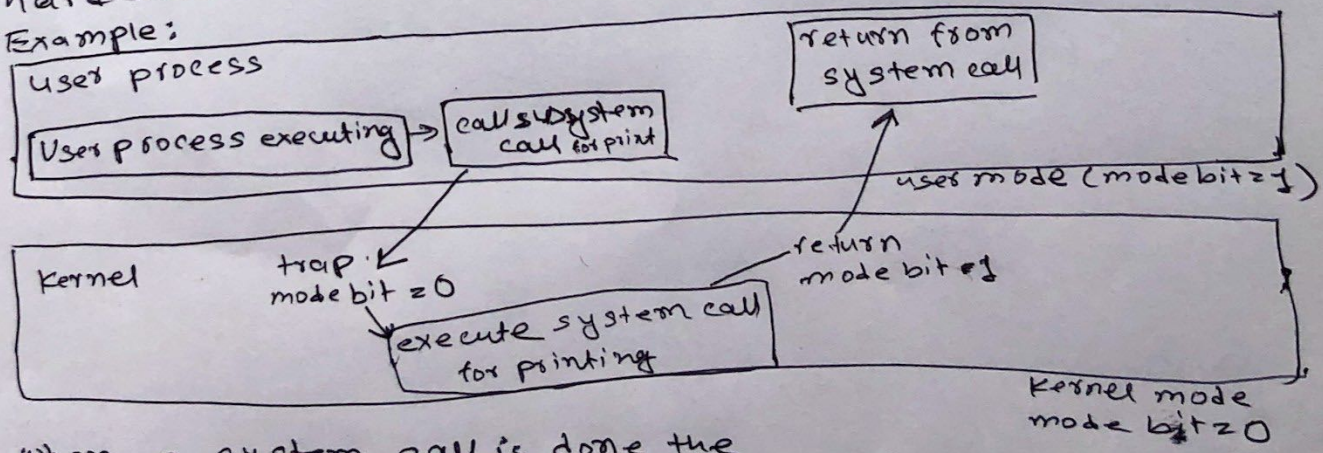
Sign: Rukaiya

We used mutex to lock reader_count. in context switch. So, Multiple reader can enter, but when any of the reader is present ~~in~~ db is down so no writer can enter. Because of writer code, multiple writer can enter but as, a binary semaphore, Exclusivelock is used., not more than 1 writer can write. When writer is 0, db is up, only then readers can enter.

Answer to question 3.9

A system call is a way in which the user program requests a service from the kernel ~~in~~ of the OS. This may include hardware related services, creation and execution of new processes or scheduling. Users programs are not allowed to access hardware directly for protection and safety reasons. This is why system call is extremely important to communicate with user program and kernel and ~~so~~ ~~got~~ therefore ~~the~~ access hardware.

Example:



When a system call is done the mode bit is set to zero, ~~th~~ traping ~~the~~ it and shifting to kernel mode to execute the system request, when it is done the mode bit is set to 1 and shifted to user mode again.

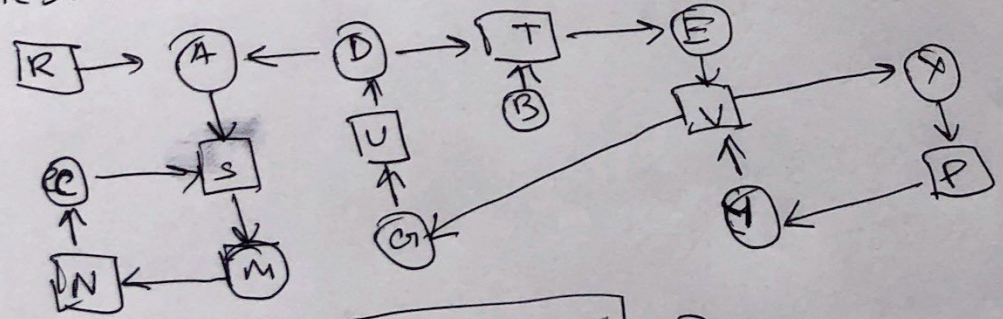
ID: 1702040004
Course: CSE 3213
Sign: Rukaiya

Answer to 1.b

Ways in which the OS deals with deadlocks are given below:

1. Dynamic Avoidance by careful resource allocation.
2. Deadlock Prevention by carefully negating one of the four required condition for deadlocks: a. Mutual Exclusion
   b. No preemption
   c. Circular wait
   d. Hold and wait
3. Detection and Recovery:
   Lets deadlock occur, detects them and then takes action
4. Deadlock Ignorance:
   Just ignores the problem, it uses Ostrich algorithm.

Diagram below is a RAG with 10 processes, resources and 3 deadlocks.



L = { R, A, S, M, N, C, S }

Pre deadlock = { S, M, N, C }

L = { }

Then, L = { B, T, E, V, G, U, D, T }

Pro deadlock = { T, E, V, G, U, D }

ID: 170204004
Course: CSE3213
Sign: Rukaiya

$L = \{B, T, E, V, X, P, Y, V\}$

Here,
Pro deadlock = {V, X, P, Y}

$L = \{\{$

Here 3 Deadlocks found because there are 3 cycles.

Ans to 1c

Theards are called Light weight processes. because they have their own stack but can access shared data. Because threads share the same address space as process and other threads within the process, the communication cost between them is low. The kernel-level is recognised and implemented by OS. Its implementation is complicated. It takes time for context switch. It needs hardware support. If one kernel thread performs blocking operation then another thread ca continue execution.

Answer to 7.a

Process means a program in execution where as thread means a segment of process. Process isn't lightweight, They take more time and space. For instance, if a process wants to deliver some data of shoppers, the process has to get required info from multiple shoppers, fetch data from the server and then deliver it. If the no. of shoppers is very large, creating same processes every time will be very costly. In this case, threads would work much more efficiently.

ID: 170209004

Course:

Sign: Rukaiya

Answer to 7.b

A type-1 hypervisor runs directly on the host machine's physical hardware, and it is reffered to as bare metal hypervisor. This does not have to load an underlying OS. with direct access to the hardware, and no other software, this type is regarded as the most efficient and best performing hypervisor.

Type-2 hypervisor is installed on top of an existing OS. It is called a hosted hypervisor, because it relises on the host machine's OS to manage calls to CPU, memory network etc.
Eg. VMware Fusion.

Ans. to 7.a

when dead lock is determined OS can recover it by 2 approaches.

1. Process Termination: To eliminate processes to get rid of deadlock. For this we can use 2 methods: (a) Abort All deadlock processes
   (b) Abort one process at a time until it is gone.
2. Recourse Preemption:
   We preempt some resources. Its issue is (a) selecting a victim
   (b) Roll back
   (c) Starvation.