1. At the time of getting input for a system call, it is required for the operating system to be in a high level cautious mode, when such information is available in memory rather than in registers. Identify the need for the constraint.

The constraint requiring the operating system (OS) to operate in a high-level cautious mode when handling input for a system call.

1. Data validation and security:-

Memory data can be modified by malicious or faculty processes before the OS processess the system call.

2. consistency and synchronization:-

Memory data may be shared across multiple threads or processes, creating a risk of concurrent modificati -on during system call execution.

3. Privilege checking:-

This constraint ensures the OS verifies proper memory permissions and boundaries.

4. complexity of Memory Access:-

The OS must carefully interpret and process such data to avoid crashes or misbehaviour.

5. Performance and Resource Management:-

The OS needs to handle these delays efficiently to ensure system performance while maintaining accuracy.

6. Interrupt handling and Fault Recovery.-

A cautious mode enables the OS to recover from such faults without compromising the systems stability.

2.)

To ensure operating system (OS) protection while imposing minimal constraints on users, a minimal set of protected instructions must be identified.

1.) change to user Mode:-

1.) protection status: Not protected.

a.) Justification:-

Changing to user mode is typically initiated by the OS when a user program is about to

2. Change to Monitor Mode:-

1. Protection status:- Must be protected

2. Justification:- Switching to monitor mode grants access to privileged instructions and sensitive system resources.

3. Read from Monitor Memory:-

1. protectionstatus:- Must be protected

2. Justification:-

Monitor memory contains sensitive data such as process control blocks, page tables, and device driver code.

4. Write into Monitor Memory:-

1.) protection status:- Must be protected

2.) Justification:- Writing into Monitor memory poses a significant threat to system integrity.

5.) Fetch an Instruction from Monitor Memory:-

1.) protectionstatus:-

Not protected

(2.) Justification:-

Fetching instructions from memory is a part of normal as operation and does not necessarily require protection as long as execution of three instructions remains controlled by the OS.

(6.) Turn on Timer interrupt:-

1.) protection status:-

Not protected

2.) Justification:-

Enabling a timer interrupt is generally safe and is often done by the os to manage process scheduling or ensure fairness.

7.) Turn off timer Interrupt:-

1.) protection status:-

Must be protected

2.) Justification:-

Disabling timer interrupts would allow user programs to monopolize the cpu, bypass time sharing mechanisms, or prevent the os from regaining control.

Minimal set of protected Instructions:-

1.) change to Monitor Mode
2.) Read from Monitor Memory
3.) Writes into Monitor Memory
4.) Turn off Timer Interrupt

3.)

process	Arrival Time	Burst Time
P_1	0	10
P_2	3	6
P_3	7	1
P_4	8	3

To solve this using the First-come-First-serve (FCFS) algorithm:-

Step1:-

Calculate completion Time (CT):-

* FCFS processes are executed in the order of their arrival times.

process	Arrival time	Burst Time	completion Time
P_1	0	10	10
P_2	3	6	16
P_3	7	1	17
P_4	8	3	20

Step2:-

calculate Turn around Time (TAT):-

TAT = CT- Arrival Time

process	completion Time	Arrival Time	Turnaround Time
P_1	10	0	10
P_2	16	3	13
P_3	17	7	10
P_4	20	8	12

Step 3:-

Calculate Average Turnaround Time:-

* Average TAT = (Sum of TATs) / Number of processes
* Average TAT = (10 + 13 + 10 + 12) / 4
  = 45/4 = 11.25

∴ The average turn around time using the FCFS algorithm is 11.25 units.

(4.)

process	Arrival Time	Burst Time
P_1	0	7
P_2	3	3
P_3	5	5
P_4	6	2

To solve for the average waiting Time using both preemptive SIF and non-preemptive STF

step1:- calculate Gant chart:-

* At every time unit, the process with the shortest remaining burst time is executed

Time	Running process	Ready Queue
0-3	P_1	—
3-6	P_2	P_1
6-8	P_4	P_1, P_3
8-13	P_3	P_1
13-14	P_1	—

Gant chart: $P_1 \rightarrow P_2 \rightarrow P_4 \rightarrow P_3 \rightarrow P_1$

Step2:-

Completion Time (CT)

process	completion Time
P_1	14
P_2	6
P_3	13
P_4	8

Step3:-

Turn around Time (TAT) and waiting Time (WT)

* TAT = CT - Arrival Time
* WT = TAT - Burst Time

process	Arrival Time	Burst Time	Completion Time	TAT = CT-AT	WT = TAT-BT
P_1	0	7	14	14-0= 14	14-7 = 7
P_2	3	3	6	3	0
P_3	5	5	13	8	3
P_4	6	2	8	2	0

Average waiting Time (AWT) - preemptive SJF:-

$$AWT = \frac{\text{sum of WT}}{\text{No. of process}} = \frac{7+0+3+0}{4}$$

$$= 10/4 = 2.5$$

(2) Non- preemptive SJF:-

Step1:- calculate gant chart:-

Time	Running process	Ready Queue.
0-7	P_1	P_2, P_3, P_4
7-10	P_2	P_3, P_4
10-15	P_4	P_3
15-20	P_3	-

Gant chart : $P_1 \rightarrow P_2 \rightarrow P_4 \rightarrow P_3$