3. Bounded waiting

→ Bound on number of times other process can enter their Critical section after process has made and request is granted.

Critical section in Operating System

1. preemptive kernel : allow a process to be preempted while it is running in kernel mode.

1. Non-preeemptive kernel: a kernel process will run until it exits kernel mode, blocks

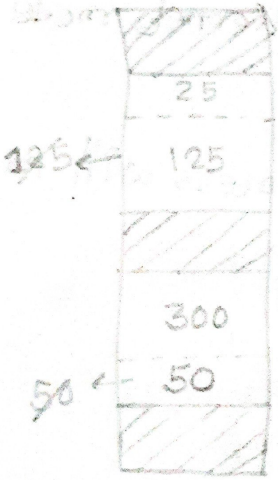
Example

Request from process are 300KB, 25KB, 125KB, 50KB respectively.

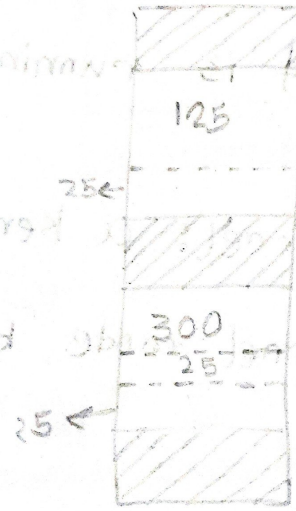
Above request could be satisfied with.

Solution.

first fit.

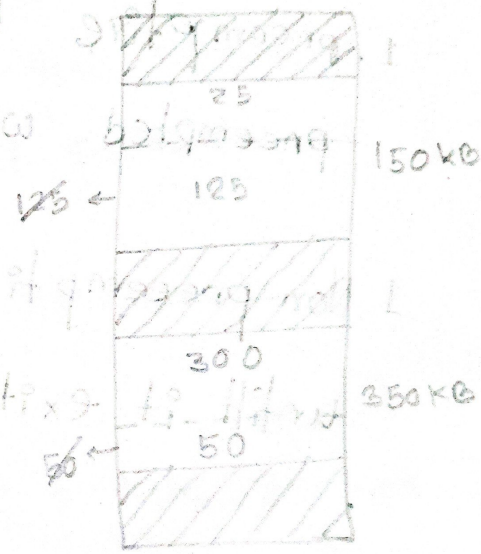


Best fit.



Not applicable

Worst fit.



Binary Semaphore

- Value Range 0 and 1
- Use to control access to a single resource
- enforce mutual exclusion for a critical section.

mutex

Counting Semaphore

- Value Range 0 to n
- Use when we have more than one process in critical section at same time.

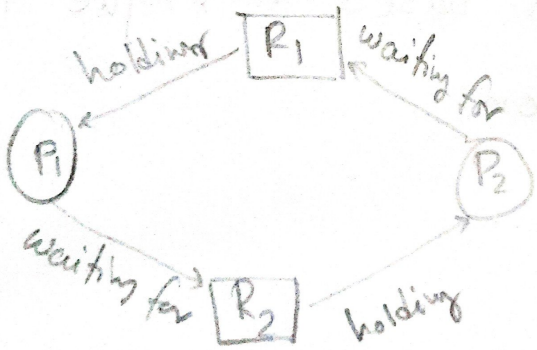
Mutex

- to give access to a resource to only one process at a time
- It allows to use same resource at a time
- It is locked based technique to handle the critical section.

Deadlock

Deadlock: Deadlock is a situation that occures in os where any process enters a waiting state because another waiting process is holding demanded resources.

- Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as soft lock or software.



Necessary Condition for Deadlock.

1. Mutual Exclusion
2. Hold and wait
3. No preemption
4. Circular wait

" If one of them is not present in a system, no deadlock will arise".

- Mutual Exclusion: One process at a time can use one resource

- A process must be held one resource and waiting acquire additional resources that are currently being held by other process

- A resource can be released when it's completed its task.

- Po is waiting for P1 . P1 is waiting for Pn+1 in circular way.

Resource Allocation Graph

Cycle: Possibility of Deadlock [may or may not]

No Cycle: No Deadlock.

Methods for Handling Deadlock.

Prevention

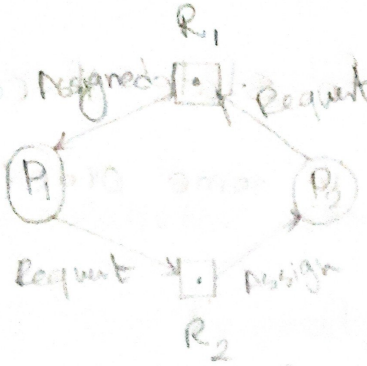
→ Ensure that the system will never nter a deadlock

Avoidance

→ Resource will be granted if resulting state does not cause deadlock.

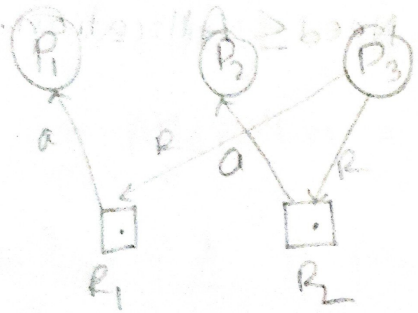
Detection and Recovery

→ Allow to system to allow deadlock and then recover.



Process No	Allocated		Request	
	R_1	R_2	R_1	R_2
P_1	1	0	0	1
P_2	0	1	1	0

Availability → (R_1 R_2) = (0 0) Deadlock



No Deadlock

Process No	Allocated		Request	
	R_1	R_2	R_1	R_2
P_1	1			
P_2		1		
P_3			1	1

Availability → (R_1, R_2)

Difference between Deadlock and Starvation.

Deadlock	Starvation
1. Also known as Circular lock	2. Also known as lived lock
2. Resource are blocked by the process	2. Resources are continuously utilized by high priority process
3. Can be prevented by the necessary condition for deadlock	3. It can ve prevented by Aging

- Safe state: System is in safe state if can allocate resources to each process in some order and still avoid deadlock.

  safe state = there exists a safe sequence.

Banker Algorithm.

Need = Max − Allocation
Need ≤ A

Available

3	3	2

	Allocation			Max			Need		
P_0	0	1	0	7	5	3	7	4	3
P_1	2	0	0	3	2	2	1	2	2 ✓
P_2	3	0	2	9	0	2	6	0	0 ✓
P_3	2	1	1	2	2	2	0	1	1 ✓
P_4	0	0	2	4	3	3	4	3	1 ✓

work

3	3	2
+ 2	0	0
5	3	2
+ 2	1	1
7	4	3
0	0	2
7	4	5
0	1	0
7	5	5
3	0	2
10	5	7

finish

P_T	P_T	P_T	P_T	P_T
0	1	2	3	4

$P_1 \rightarrow P_3 \rightarrow P_4 \rightarrow P_0 \rightarrow P_2$

Resource - Request Algorithm for Process P_i

Thi request should be

$$Request_i \leq Need_i \text{ and } Request_i \leq Available_i$$

Then.

$$Available = Available - Request$$

$$Need_i = Need_i - Request$$

$$Allocation = Allocation + Request.$$

what if P_1 Request [1,0,2]

P_1: Need (1.2.2) $\geq$ (1.0.2) request

Available (3.2.2) $\geq$ (1.0.2) reaquest

	Allocation			Need		
P_0	0	1	0	7	4	3
P_1	3	0	2	0	2	0
P_2	3	0	2	6	0	0
P_3	2	1	1	0	1	1
P_4	0	0	2	4	3	1

Available: [2 3 0]

$P_1 \rightarrow P_3 \rightarrow P_4 \rightarrow P_0 \rightarrow P_2$

work

2 3 0
+ 3 0 2
5 3 2
+ 2 1 1
7 4 3
+ 0 0 2
7 4 5
+ 0 1 0
7 5 5
+ 3 0 2
10 5 7

Semaphore

A semaphore is a integer variable which can accessed two standard atomic operation

- wait ()
- signal().

Type of Semaphore

- Counting Semaphore
- Binary Semaphore.

Counting Semaphore

→ Initialize to the number resource available S = N

→ if S = 0. all resource are being used.
will block

→ all resource ^ until S > 0

→ Solver various synchronization problem.

→

SEMAPHORE Implementation

→ During semaphore operation. if semaphore >0, it must wait.

→ Rather than busy waiting, we block itself and replace into a waiting queue.

→ State of the process is switched to waiting queue and control is transfer to the CPU scheduler.

→ It will restarted when other process executes a signal() operation.

→ Restarted by a wake up operation that change it forom waiting state to ready state.

- Page size = 2kB = 2048 Bytes
- Process size = 73,506 Bytes

1 KB → 1024 Bytes
1 Byte → 8 bits

Solution:

$$\text{Number of frames } n(\text{frames}) = \frac{\text{Process Size}}{\text{Page Size}}$$

$$= \frac{73,506}{2048} = 35.892 \approx 36$$

[Internal fragmentation because of point value occure]

$$\therefore \text{Internal fragmentation} = (36 \times 2048) - 73,506$$

$$= 222$$

Main Memory

Difference Between Logical Address and physical Address

Logical Address	Physical Address
1. Generated by CPU	1. Computed by MMU
2. User can view the logical address of a program.	2. User can never view the logical address of a program.
3. Logical address can be changed.	3. Physical address can not be changed.
4. Virtual Address	4. Real Address.

Dynamic Loading

→ Routine is not loaded until it is called

→ Better memory space Utilization

→ All routine kept on disc relocatable load format.

→ Useful when handle large amount data

→ No support in OS.

Dynamic Linking

→ System also know as share librarier

→ Useful for librarier

→ linking prospond untill execution time.

Effective Access Time

- Consider $\alpha = 80\%$, $\epsilon = 20ns$ for TLB search, 100ns for memory access.

EAT = Hit (TLB + Memory) + Miss (Page Table + Main Memory)

$= \frac{80}{100}(20 + 100) + \frac{20}{100}(100 + 100)$

$= 136ns$

- Consider more realistic hit ration $\alpha = 99\%$, $\epsilon = 20ns$ for TLB search, 100 ns for memory access.

EAT = Hit (TLB + Memory) + Miss (Page + Memory)

$= \frac{99}{100}(20 + 100) + \frac{1}{100}(100 + 100)$

$= 120.8 ns.$

• How we reduce exterral fragmentation

→ We can use compaction to reduce probability of external fragmentation. In compaction, all the free partition are made contiguous and all the load partition are brought together. By apply this technique. we can store bigger process in memory.

But compaction is possible when relocation is dynamic. and is done at execution time.

1. Paging

Page No	Bytes	
0	0	1
1	2	3

P_1

Frame No	Bytes	
0	0	1
1	2	3
2	4	5
3	6	7
4	8	9
5	10	11
6	12	13
7	14	15

Main memory

Main Memory Size = 16B

Frame size = 2B

No of frame = $\frac{16}{2}$

= 8

Process Size = 4B (Bytes)

Page size = 2B

No of Page/Process = $\frac{4}{2}$

= 2

 # Page size = # frame size

Example

Page size = 4 bytes

memory Size = 32 bytes

0	5
1	0
2	7
3	2

find the corresponding logical address of the physical address.

① 1001

Solution

① 1001 → offset

↳ represent 2 in decimal

and from page table 2 has 7. binary value of 7 is 111.

we need 2 bit for represent 4byte. So Last 2 bit is offset.

Physical Address = 111 01

- Consider a system which has

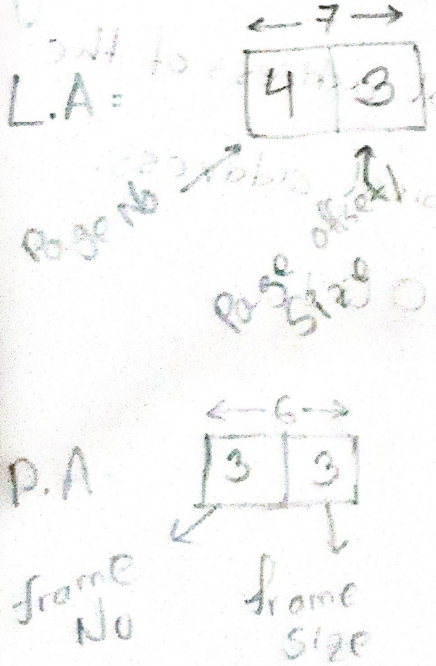
Physical Address = 6 bits

Logical Address = 7 bits

Page Size = 8 words/byte

Calculate no of pages and no of framen.

Solution.



Page Size = 8 words

In bits = $\log_2(8)$

= 3 bits

No. of PageNo = $2^4 = 16$

No. of frame No = $2^3 = 8$

No. of offset = No. of frame size = $2^3 = 8$

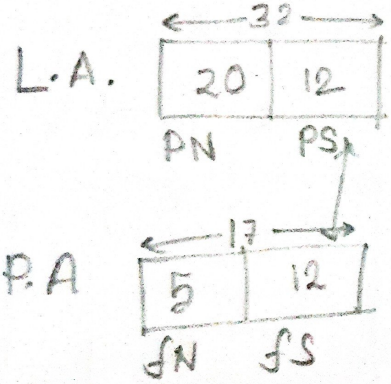
• Consider a virtual (Logical Address) Address space of 32bits and Page size of 4kB. System is having a RAM of 128 kB. Then what will be the ration of (Page No.) page table and Inverted page table size if each entry in both is of size 4B.

(P.A)

Solution:

Page size = 4kB
$= 2^2 \times 2^{10}$ B
$= 2^{12}$ Bits

RAM = 128 kB
$= 2^7 \times 2^{10}$
$= 2^{17}$



• Page table = 20 bits

Page table size = 2^{20}

• Inverted page table = 5 bits

Inverted page table size = 2^5 ~~bits~~

• ratio = $\frac{2^{20} \times 4B}{2^5 \times 4B}$

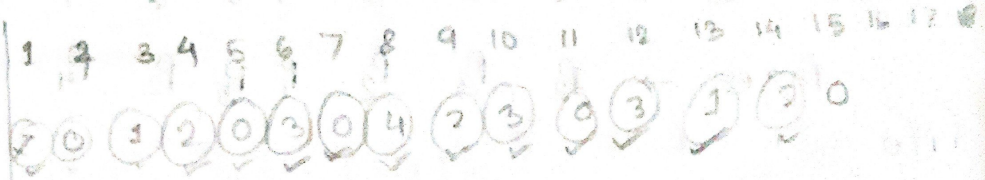
$= 2^{15}$

∴ ratio : 2^{15} : 1

Virtual Memory

Page Replacement

- FIFO
- LRU
- Optimal



FIFO

	7	0	1	2	0	3	0	4	2	3	0	3	1	2	0
	7	7	7	2	2	2	2	4	4	4	0	0	0	0	1
		0	0	0	0	3	3	3	2	2	2	2	1	1	0
			1	1	1	1	0	0	0	3	3	3	3	2	2
	*	*	*	*	hit	*	*	*	*	*	*	hit	*	*	hit

hit ratio = $\frac{3}{15} \times 100\%$

= 20%

Optimal

	7	0	1	2	0	3	0	4	2	3	0	3	1	2	0
	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2
		0	0	0	0	0	0	0	0	0	0	0	0	0	0
			1	1	1	3	3	4	4	3	3	3	1	1	1
	*	*	*	*	hit	*	hit	*	hit	*	hit	hit	*	hit	hit

hit Ratio = $\frac{7}{15} \times 100\%$

= 46%

LRU

	7	0	1	2	0	3	0	4	2	3	0	3	1	2	0
	7	7	7	2	2	2	2	4	4	4	0	0	0	2	2
		0	0	0	0	0	0	0	2	3	3	3	3	3	0
			1	1	1	3	3	3	3	2	2	2	1	1	1
	*	*	*	*	hit	*	hit	*	*	*	*	hit	*	*	*

hit Ratio = $\frac{3}{15} \times 100$

= 20%

time →	1	2	3	4	5	6	7	8	9	10	11	12
page →	P_2	P_3	P_2	P_1	P_5	P_2	P_4	P_5	P_3	P_2	P_5	P_2

fifo

	P_2	P_3	P_2	P_1	P_5	P_2	P_4	P_5	P_3	P_2	P_5	P_2
f_1	P_2	P_2	P_2	P_2	P_5	P_5	P_5	P_5	P_3	P_3	P_3	P_3
f_2		P_3	P_3	P_3	P_3	P_2	P_2	P_2	P_2	P_2	P_5	P_5
f_3				P_1	P_1	P_1	P_4	P_4	P_4	P_4	P_4	P_2
	*	*	hit	*	*	*	*	hit	*	hit	*	*

Hit Ratio = $\frac{3}{12} \times 100\% = 25\%$

LRU

	P_2	P_3	P_2	P_1	P_5	P_2	P_4	P_5	P_3	P_2	P_5	P_2
f_1	P_2	P_2	P_2	P_2	P_2	P_2	P_2	P_2	P_3	P_3	P_3	P_3
f_2		P_3	P_3	P_3	P_5	P_5	P_5	P_5	P_5	P_5	P_5	P_5
f_3				P_1	P_1	P_4	P_4	P_4	P_4	P_2	P_2	P_2
	*	*	hit	*	*	hit	*	hit	*	*	hit	hit

Hit Ratio = $\frac{5}{12} \times 100\% = 41.66\%$

Optimal

	P_2	P_3	P_2	P_1	P_5	P_2	P_4	P_5	P_3	P_2	P_5	P_2
f_1	P_2	P_2	P_2	P_2	P_2	P_2	P_4	P_4	P_4	P_2	P_2	P_2
f_2		P_3	P_3	P_3	P_3	P_3	P_3	P_3	P_3	P_3	P_3	P_3
f_3				P_1	P_5	P_5	P_5	P_5	P_5	P_5	P_5	P_5
	*	*	hit	*	*	hit	*	hit	hit	*	hit	hit

Hit Ratio = $\frac{6}{12} \times 100\% = 50\%$

Optimal > LRU > FIFO

[Optimal will be best Solution]