Date / /
Page No.
Shivalal

Que:① What is Deadlock ? How you prevent system from Hold & wait and Circular Wait Condition ?

Answer: Deadlock is a condition in which one process is waiting for resource from defined time.

We can prevent System from Hold and wait from three method:

1) Conservative approach - assign all resource for a process at initial.

2) Do not Hold - If you process need a resource then it must release all the holded resources.

3) wait time out - After a maximum time wait by process for a resource, it must release all holding resource

We can prevent System from Circular wait using:

① first giving a natural number of every resource

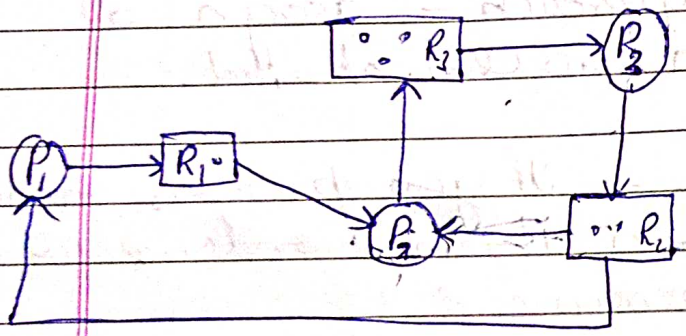
② all every process to either only in the increasing of decreasing order of resource number.

Que: ⑦ Draw the Resource allocation graph

$P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$

$P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_2$ and find out whether deadlock exist or not? Justify.

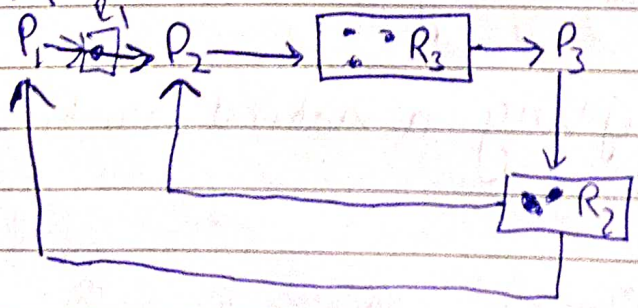
(Note - R_1, R_2, R_3 has one, two, three instance respectively.)



1) Mutual exclusion
2) Hold and wait

In simple way ←

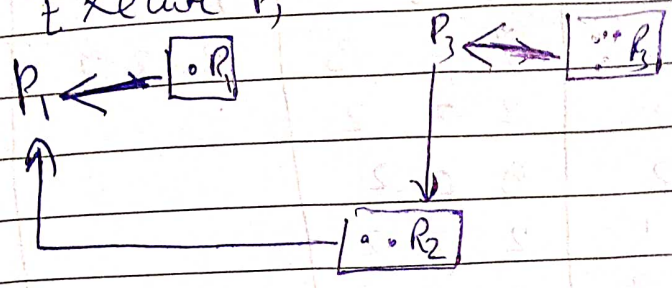
Step 1:



Execute: P_2

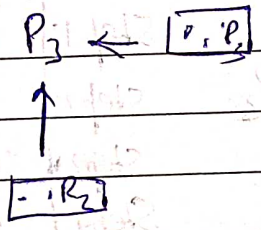
$P_2 \rightarrow$

Step 2:
Execute P_1



$\langle P_2, P_1 \rangle$

Step 3:
Execute P_3



$\langle P_2, P_1, P_3 \rangle$ is one safe sequence

No, deadlock exist

Ans (3) Attempt all of the following:

Process	Allocation A	B	C	Max A	B	C	Available A	B	C	Need A	B	C
P_0	0	1	0	7	5	3	3	3	2	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

a) Need Matrix

b) Execute - P_1

Step 1: $< P_1$

Execute - P_3

Step 2: $< P_1, P_3$

Execute - P_4

Step 3: $< P_1, P_3, P_4$

Execute - P_0

Step 4: $< P_1, P_3, P_4, P_0$

Execute - P_2

Step 5: $< P_1, P_3, P_4, P_0, P_2 >$

	Available A	B	C
Step 1:	5	3	2
Step 2:	7	4	3
Step 3:	7	4	5
Step 4:	7	5	5
Step 5:	10	5	7

c) Safe sequence: $< P_1, P_3, P_4, P_0, P_2 >$

d) P_4 request for additional resource (3, 3, 0)

Date / /
Page No.

Shivaldi

Make change in max-need and ~~Allocation~~ need

Process	Allocation			Max			Available			Need		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2	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

~~Request~~

Execute P_1

Step 1: $< P_1$

Execute P_3

Step 2: $< P_1, P_3$

Execute P_0

Step 3: $< P_1, P_3, P_0$

Execute P_2

Step 4: $< P_1, P_3, P_0, P_2$

	Available		
Step 1:	5	3	2
Step 2:	7	4	3
Step 3:	7	5	3
Step 4:	10	5	5

for executing P_4 we have not sufficient B resource

Hence, we cannot grant additional resource for P_4

Que: ① Consider the snapshot of the system and solve the following Banker's Algorithm,

Process	Allocation	Max	Available	Need
Resource	A B C D	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 1 2	1 5 2 0	0 0 0 0
P_1	1 0 0 0	1 7 5 0		0 7 5 0
P_2	1 3 5 4	2 3 5 6		1 0 0 2
P_3	0 6 3 2	0 6 5 2		0 0 2 0
P_4	0 0 1 4	0 6 5 6		0 6 4 2

↑ a) Need matrix

P_0 Executed

Step 1: $< P_0$

	Available
Step 1:	1 5 3 2
Step 2:	2 8 8 6
Step 3:	3 8 8 6
Step 4:	3 14 11 8
Step 5:	3 14 12 12

P_2 Executed

Step 2: $< P_0\ P_2$

P_1 Executed

Step 3: $< P_0\ P_2\ P_1$

P_3 Executed

Step 4: $< P_0\ P_2\ P_1\ P_3$

P_4 Executed

Step 5: $< P_0\ P_2\ P_1\ P_3\ P_4 >$

b) Safe Sequence Exist $< P_0\ P_2\ P_1\ P_3\ P_4 >$

iii) Check new ~~order~~ resource allocated state

	Allocation	Max	Available	Need
	A B C D	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 1 2	1 5 2 0	0 0 0 0
P_1	1 0 0 0	1 7 5 0		0 7 5 0
P_2	1 3 5 4	2 3 5 6		1 0 0 2
P_3	0 6 3 2	0 6 5 2		0 0 2 0
P_4	0 0 1 4	0 6 5 6		0 6 4 2

Execute P_0

Step 1: $< P_0$

Execute P_2

Step 2: $< P_0 P_2$

Execute P_3

Step 3: $< P_0 P_2 P_3$

Execut P_4

Step 4: $< P_0 P_2 P_3 P_4$

Execute P_1

Step 5: $< P_0 P_2 P_3 P_4 P_1 >$

	Available
Step 1:	1 5 3 2
Step 2:	2 8 8 6
Step 3:	2 14 11 8
Step 4:	2 19 12 12
Step 5:	3 19 12 12

We can allow request of P_1

Safe sequence exist

Que: A computer system has 6 tape drives with n processes competing for them. Each process may need 3 tape drives. What is the maximum value of n for which the system is guranted to be no deadlock? Justify your answer.

Answer: (5)

Process	Max need	Allocated
P_0	3	0
P_1	3	0
:	:	0
:	:	0
P_n	3	0

$6 >= 3n$

$n <= 2$

$n = 2$

Step 1: Calculate the total number of available resources
Total resources = 6 tape drives

Step 2: Calculate the total resource allocation for all processes.
Total allocation = 0 tape drives

Step 3: Calculate the total resource needs for all processes
Total needs = 3n tape

Step 4: Calculate the total remaing resource
Remaing resources = Total resources - Total allocation
= 6 - 0
= 6 tapes drives

Step 5: Check if the remaing resource are greater than or equal to the total needs of any process
Remaining resources >= Total need of any process

Date / /
Page No.
Shivalal

Que: 6 An OS contain 3 resource classes. The number of resource units in these classes is 7, 7, 10 respectively. The current resource allocation state is shown:

Answer:

Process	Allocated R_1 R_2 R_3	Max R_1 R_2 R_3	Need R_1 R_2 R_3	Available R_1 R_2 R_3
P_1	2 2 3	3 6 8	1 4 5	2 3 0
P_2	2 0 3	4 3 3	2 3 0	
P_3	1 2 4	3 4 4	2 2 0	
	5 4 10			

Execute P_2

step 1: $< P_2$

	Available A B C
step1	4 3 3
step2	5 5 7
step3	7 7 10

Exeat P_3

step 2: $< P_2\ P_3$

Execute P_1

step 1: $< P_2\ P_3\ P_1 >$

Safe sequence EXIST

a) yes

b) P_1 request (1, 1, 0)

granted since, Available after step 2: 5 5 7

need for P_1 is 2 5 5

ii) P_2 request (0, 1, 0)
new resource allocated

Process	Allocated			Max			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
										2	3	0
P_1	2	2	3	3	6	8	1	4	5			
P_2	2	0	3	4	4	3	2	4	0			
P_3	1	2	4	3	4	4	2	2	0			

Execute P_3

Step 1: $\langle P_3$

	Available		
	A	B	C
Step 1:	3	5	4
Step 2:	5	5	7
Step 3:	7	7	10

Execute P_2

step 2: $\langle P_3 P_2$

Execute P_1

Step 3: $\langle P_3 P_2 P_1 \rangle$

Safe sequence exist
we can grant

iii) P_1 request (0, 1, 0)
from part a) need (1, 5, 5)
Available (5, 5, 7)
need $\leq$ available

we can grant request

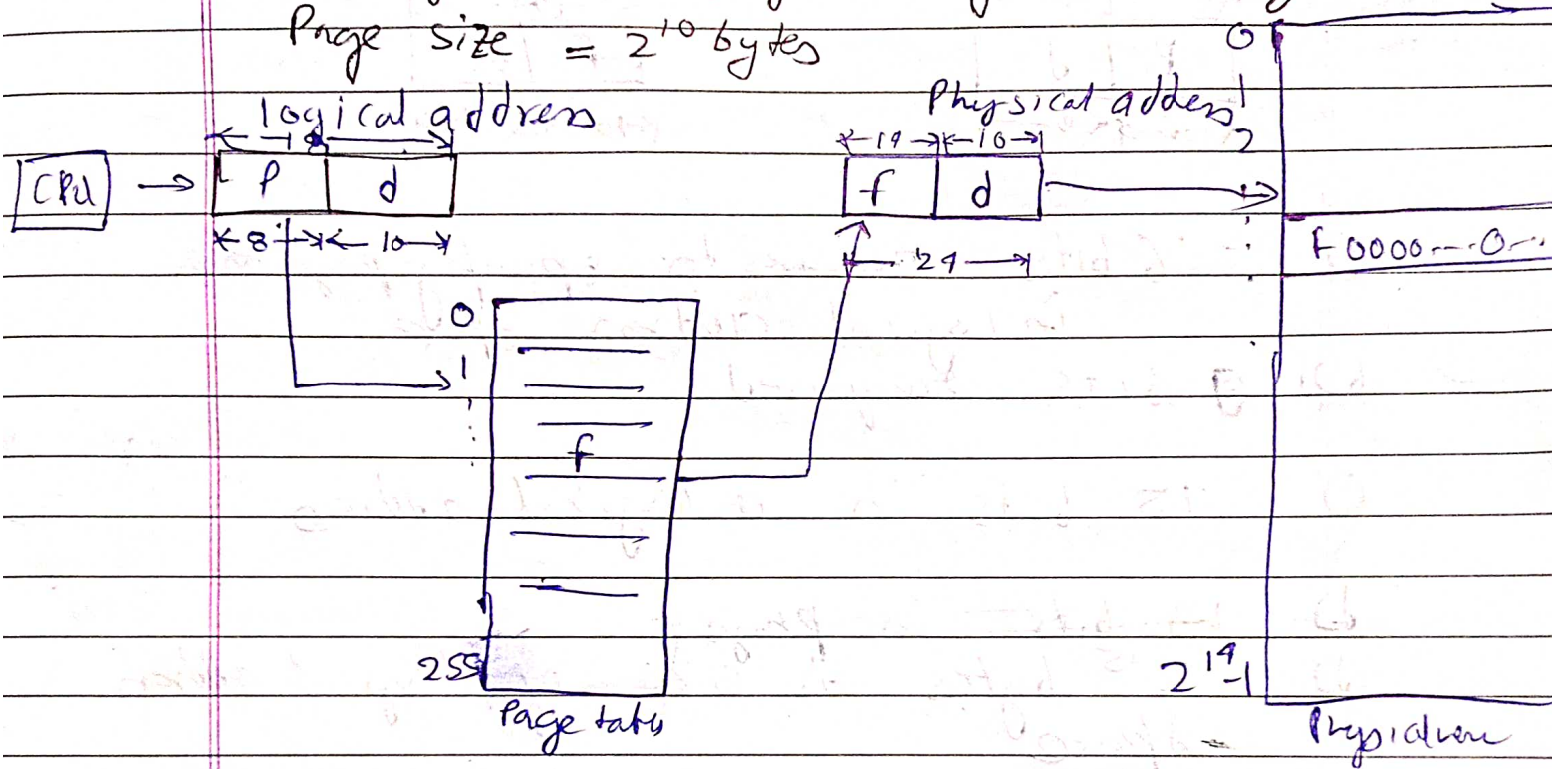
Solution: (7) $n = 2$

same question as question 5

Question: (8) Physical memory $= 2^{24}$ bytes

Total Pages $= 256$ pages in logical memory

Page size $= 2^{10}$ bytes



Size of Physical memory $= \frac{2^{24}}{2^{10}} = 2^{14}$ bytes

a) 18 bits are there in logical address i.e 2^{18} bytes total logical address space

b) 2^{10} bytes in a page frame

c) 24 bit address is required

d) 256 entries in a Page table

e) each entry — 1 bit for page is valid or not
14 bit for frame number

Solution: (5)

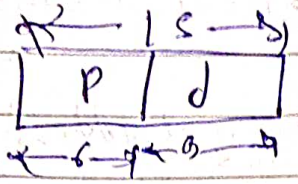
a) Total page = 64 entries

Page size = 16 bytes

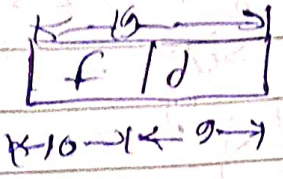
Frames size = 2^{10} bytes

Page size = 512

Logical



Physical



a) 6 bits required to specify page number in logical address space

b) 9 bits required

c) 15 bits in a logical address

d) ~~19 bits in physical~~

d) 2^{15} bytes is size of logical address space

e) 10 bits required

f) 9 bits required

g) 19 bits in physical address

h) 2^{19} bytes is size of physical address space.

Date / /
Page No.
Shivalal

Solution: (10) address space is specified by 24 bits, which means that there are 2^{24} possible addresses.

⇒ each address corresponds to one word
number of words in address space is 2^{24}

⇒ memory space = 16 bits
Possible memory locations $= 2^{16}$
number of words in memory space $= 2^{16}$

⇒ Page consist $= 2^{11}$ words $= 2^{11}$ bytes is each page

Number of pages in Logical memory $= \frac{2^{24}}{2^{11}} = 2^{13}$

Number of frames in Physical memory $= \frac{2^{16}}{2^{11}} = 2^5$

There are 2^{13} Pages and 2^5 frames
↓ ↓
8192 Pages 32 Frames

Solution: (a)

Page size = 8K = $2^3 \times 2^{10} = 2^{13}$

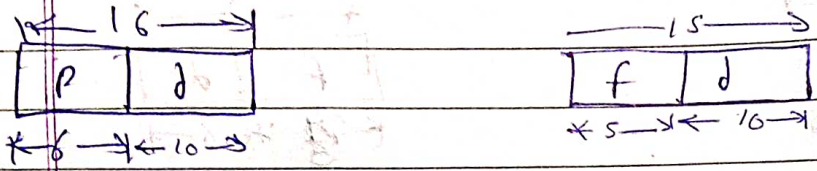
Virtual address space = 36 bit = 2^{36}

(a) Total Page = $\frac{2^{36}}{2^{13}} = 2^{23}$ pages

(b) maximum size of addressable physical memory in this system is determined by the number of bits used for Physical addresses. Since there are 36 bits for virtual addresses we need to determine how many bits are used for physical addresses. This is determined by the size of the physical memory that can be addressed.

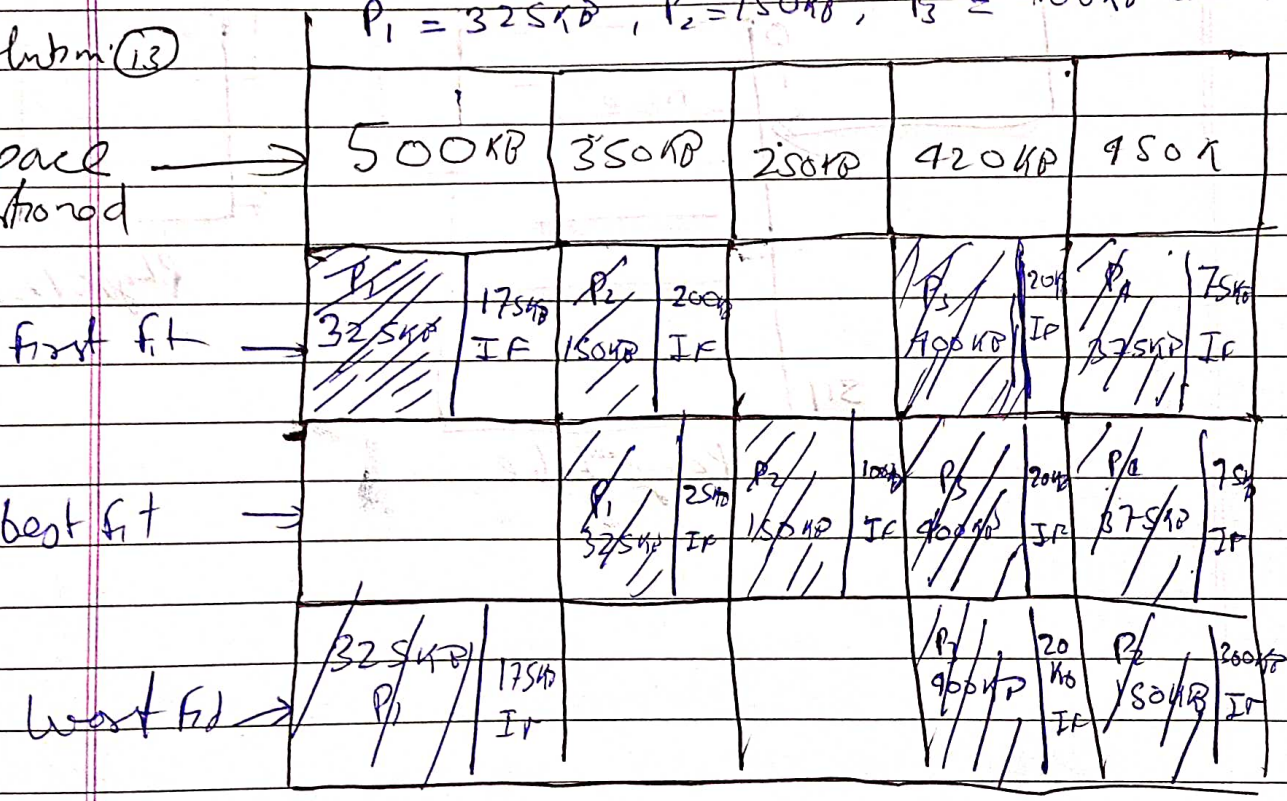
Let's assume that the system uses a 38-bit physical address space, which is common for modern processors. This means that there are 13 more bits available for physical addresses than for virtual addresses. Therefore the max size of addressable physical memory in this system is = 2^{45} bytes

Solution (12) Total pages = 64 pages
each page size = 1024 words
frames = 32



a) 16 bits in logical address space
b) 15 bits in physical address space

Solution (13) $P_1 = 325KB$, $P_2 = 150KB$, $P_3 = 400KB$ and $P_4 = 375KB$



IF → Internal fragmentation

~~In~~ In worst fit we cannot fit Process P_4 of size 375KB

Solution: (11)

Logical memory space = 32K = 2^{15} words
Physical memory space = 8K words = 2^{9} words
Page size = 64 words = 2^{6} words