Q1. Buddy Memory Allocation

The Buddy method for allocating memory to processes shall be used for a memory with a capacity of 1024 kB. Perform the provided operations and give the occupancy state of the memory after each operation.

0 128 256 384 512 640 768 896 1024

<table>
<tr><td>Initial state</td><td colspan="8">1024 KB</td></tr>
<tr><td>65 KB request => A</td><td>A</td><td colspan="3">128</td><td colspan="2">256</td><td colspan="2">512</td></tr>
<tr><td>30 KB request => B</td><td>A</td><td>B</td><td>32</td><td>64</td><td colspan="2">256</td><td colspan="2">512</td></tr>
<tr><td>90 KB request => C</td><td>A</td><td>B</td><td>32</td><td>64</td><td>C</td><td>128</td><td colspan="2">512</td></tr>
<tr><td>34 KB request => D</td><td>A</td><td>B</td><td>32</td><td>D</td><td>C</td><td>128</td><td colspan="2">512</td></tr>
<tr><td>130 KB request => E</td><td>A</td><td>B</td><td>32</td><td>D</td><td>C</td><td>128</td><td>E</td><td>256</td></tr>
<tr><td>Free C</td><td>A</td><td>B</td><td>32</td><td>D</td><td colspan="2">256</td><td>E</td><td>256</td></tr>
<tr><td>Free B</td><td>A</td><td colspan="2">64</td><td>D</td><td colspan="2">256</td><td>E</td><td>256</td></tr>
<tr><td>275 KB request => F</td><td colspan="8">Not Available → wait</td></tr>
<tr><td>145 KB request => G</td><td>A</td><td colspan="2">64</td><td>D</td><td colspan="2">G</td><td>E</td><td>256</td></tr>
<tr><td>Free D</td><td>A</td><td colspan="3">128</td><td colspan="2">G</td><td>E</td><td>256</td></tr>
<tr><td>Free A</td><td colspan="4">256</td><td colspan="2">G</td><td>E</td><td>256</td></tr>
<tr><td>Free G</td><td colspan="6">512</td><td>E</td><td>256</td></tr>
<tr><td>Free E</td><td colspan="8">1024</td></tr>
</table>

Q2. Page Replacement Strategies

a. **Perform the access sequence with the replacement strategies Optimal, LRU, FIFO and CLOCK once with a cache with a capacity of 4 pages and once with 5 pages. Also calculate the number of page faults attempted in each strategy.**

→ Optimal replacement strategy (OPT):

Requests	1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4	5
Page 1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	2	1	1	1	1
Page 2		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 3			5	5	2	2	2	2	2	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Page 4				4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Fault?					X					X	X									X	X			

Number of page faults: 5

→ Replacement strategy Least Recently Used (LRU):

Requests	1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4	5
Page 1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 2		3	3	3	3	3	3	3	3	3	5	5	5	5	5	5	5	5	5	5	1	1	1	1
Page 3			5	5	5	5	5	5	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4
Page 4				4	4	4	4	4	4	0	0	0	0	0	0	0	0	0	0	2	2	2	2	5
Fault?					X				X	X	X	X			X					X	X			X

Number of page faults: 9

→ Replacement strategy FIFO:

Requests	1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4	5
Page 1	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	5
Page 2		3	3	3	3	3	3	3	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4
Page 3			5	5	5	5	5	5	5	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2
Page 4				4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	1	1	1	1
Fault?					X				X	X	X	X			X					X	X			X

Number of page faults: 9

→ Replacement strategy CLOCK:

Rqs	1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4	5
P1	1*	1*	1*	→ 1*	2*	2*	2*	2*	2*	2	→ 2	3*	3*	3*	3*	3*	3*	3*	3*	3	→ 3	→ 3*	→ 3*	5*
P2	→	3*	3*	3*	→ 3	→ 3	→ 3*	→ 3*	3	0*	0*	→ 0*	→ 0*	→ 0*	0	0	0	0	0	2*	2*	2*	2*	→ 2
P3		→	5*	5*	5	5	5	5	1*	→ 1*	1	1	1	1	4*	4*	4*	4*	4*	→ 4*	4	4	4*	4
P4			→	4*	4	4*	4*	4*	→ 4*	4	5*	5*	5*	5*	→ 5*	→ 5*	→ 5*	→ 5*	→ 5*	5	1*	1*	1*	1
F?					X				X	X	X	X			X					X	X			X

Number of page faults: 9

b. Why is it impossible to implement the optimal replacement strategy OPT?

We can't know future requests

c. Deduce which of the strategies is closest to the Optimal one

LRU and FIFO and CLOCK have same number of faults

Q3. Consider the following workload:

Process	Burst Time	Priority	Arrival Time
P1	50 ms	4	0 ms
P2	20 ms	1	20 ms
P3	100 ms	3	40 ms
P4	40 ms	2	60 ms

a. **Show the schedule using shortest remaining time, non-preemptive priority (a smaller priority number implies higher priority) and round robin with quantum 30ms. Use time scale diagram as shown below for the FCFS example to show the schedule for each requested scheduling policy.**

Example for FCFS (1 unit = 10ms):

P1	P1	P1	P1	P1	P2	P2	P3	P3	P3	P3	P3	P3	P3	P3	P3	P3	P4	P4	P4	P4

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

Shortest Remaining Time First (Preemptive SJF)

P1	P1	P2	P2	P1	P1	P1	P4	P4	P4	P4	P3	P3	P3	P3	P3	P3	P3	P3	P3	P3

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

Priority

P1	P1	P1	P1	P1	P2	P2	P4	P4	P4	P4	P3	P3	P3	P3	P3	P3	P3	P3	P3	P3

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

RR (q=30)

P1	P1	P1	P2	P2	P1	P1	P3	P3	P3	P4	P4	P4	P3	P3	P3	P4	P3	P3	P3	P3

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

b. **What is the average waiting time of the above scheduling policies (shortest remaining time, non-preemptive priority, round robin, and FCFS)?**

	P1	P2	P3	P4	Average
SRT	20	0	70	10	25
PRIORITY	0	30	70	10	27.5
RR	20	10	70	70	42.5
FCFS	0	30	30	110	42.5

Q4. We consider a mass medium for which the allocation is carried out in blocks of 2 KB. The mass medium comprises 40 blocks. The blocks are numbered from 0 to 39. At time t, the following disk blocks are allocated: 1, 5, 7, 10, 11, 21, 22, 26, 33, 37 and 38.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

The allocation method used on the mass support is that of contiguous allocation. In this context, we want to allocate a new file "FICH1" of 10 KB.

a. **Give the block numbers allocated to the file "FICH1" in the case of a First Fit allocation, and in the case of a Best Fit allocation.**

First Fit allocation: 12, 13, 14, 15 and 16

Best Fit allocation: 27, 28, 29, 30 and 31

b. **Another file "FICH2" of 14 KB will be allocated. Give the block numbers allocated to the "FICH2" file if possible in the case of a First Fit allocation, and in the case of a Best Fit allocation.**

First Fit allocation: Stuck

Best Fit allocation: 12, 13, 14, 15, 16, 17 and 18

c. **According to the above scenario, which algorithm uses memory most efficiently?**

Best Fit

Q5. Under the following hypothetical memory management schemes

a. **Write the binary translation of the logical address 0000010110111010 to the physical address under a paging system with a 256-address (2^8) page size where the corresponding frame number in the page table is 00010101 (8-bits)**

Page Number:

00000101 → 5

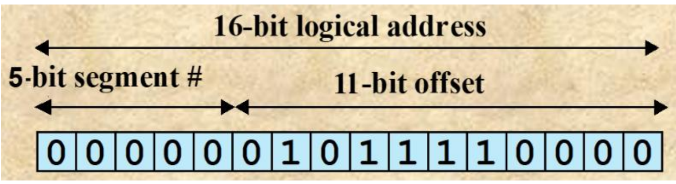
Offset:

10111010

Binary Physical Address:

0001010110111010

b. **Write the binary translation of the following logical address to the physical address under the segmentation system using the below Process Segment table**



	Length	Base
0	001011101110	0000010000000000
1	011110011110	0010000000100000

Process segment table

Binary Physical Address:

0000010000000000

\+ 01011110000

0000011011110000

Q6. Banker's Algorithm

Consider the following snapshot of a system at time T0:

Processes	Allocation R0 R1 R2	Max R0 R1 R2	Available R0 R1 R2
P_0	1 1 2	4 3 3	2 1 0
P_1	2 1 2	3 2 2	
P_2	4 0 1	9 0 2	
P_3	0 2 0	7 5 3	
P_4	1 1 2	1 1 2	

a. Determine the number of instances of each type of resource (R1, R2 and R3).

The total amount of resources = sum of columns of allocation + Available = [8 5 7] + [2 1 0] = [10 6 7]

b. Calculate the content of the need matrix.

Need = Max – Allocation

Process	Need		
	R1	R2	R3
P_0	3	2	1
P_1	1	1	0
P_2	5	0	1
P_3	7	3	3
P_4	0	0	0

c. Is the system in a safe state? If yes, show a safe sequence.

1. For process P_0, Need = (3, 2, 1) and

   Available = (2, 1, 0)

Need ? Available = False
So, the system will move for the next process.

2. For Process P_1, Need = (1, 1, 0)
Available = (2, 1, 0)
Need ? Available = True
Request of P_1 is granted.
? Available = Available +Allocation
= (2, 1, 0) + (2, 1, 2)
= (4, 2, 2) (New Available)

3. For Process P_2, Need = (5, 0, 1)
Available = (4, 2, 2)
Need ? Available = False
So, the system will move to the next process.

4. For Process P_3, Need = (7, 3, 3)
Available = (4, 2, 2)
Need ? Available = False
So, the system will move to the next process.

5. For Process P_4, Need = (0, 0, 0)
Available = (4, 2, 2)
Need ? Available = True
Request of P_4 is granted.
? Available = Available + Allocation
= (4, 2, 2) + (1, 1, 2)
= (5, 3, 4) now, (New Available)

6. Now again check for Process P_2, Need = (5, 0, 1)
Available = (5, 3, 4)
Need ? Available = True
Request of P_2 is granted.
? Available = Available + Allocation
= (5, 3, 4) + (4, 0, 1)
= (9, 3, 5) now, (New Available)

7. Now again check for Process P_3, Need = (7, 3, 3)
Available = (9, 3, 5)
Need ? Available = True
Request of P_3 is granted.
? Available = Available +Allocation
= (9, 3, 5) + (0, 2, 0) = (9, 5, 5)

8. Now again check for Process P_0, = Need (3, 2, 1)
= Available (9, 5, 5)
Need ? Available = True
So, the request will be granted to P_0.
Safe sequence: < P_1, P_4, P_2, P_3, P_0>. **The system allocates all the needed resources to each process. So, we can say that system is in a safe state.**

Q7. Write "True" if you agree on the statement or correct it if you believe it is false.

a. The physical addresses are usually much larger than logical addresses

……………False………

b. Computing the physical address from a logical address is harder with paging than with segmentation

…………… False ……

c. The OPT page replacement algorithm is the most efficient but unrealizable one

……………True……

d. The clock algorithm is better than both LRU and FIFO

……………True……

e. Paging may cause external fragmentation

……………False……

f. If the page table is large, then it may be paginated itself

……………True……