→ 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

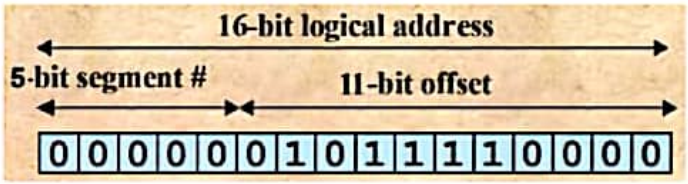
Offset:

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.....

Ex 1

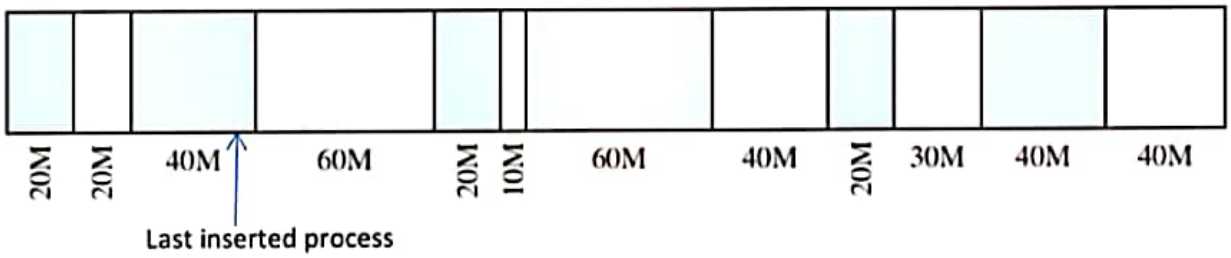
Consider a fixed partitioning scheme with equal-size partitions of 2^{16} bytes and a total main memory size of 2^{24} bytes. A process table is maintained that includes a pointer to a partition for each resident process. How many bits are required for the pointer?

Answer:

The number of partitions equals the number of bytes of main memory divided by the number of bytes in each partition: $2^{24}/2^{16} = 2^8$. Eight bits are needed to identify one of the 2^8 partitions.

Ex 2

A dynamic partitioning scheme is being used, and the following is the memory configuration at a given point in time:



The shaded areas are allocated blocks; the white areas are free blocks. The next three memory requests are for 40M, 20M, and 10M. Indicate the starting address for each of the three blocks using the following placement algorithms:

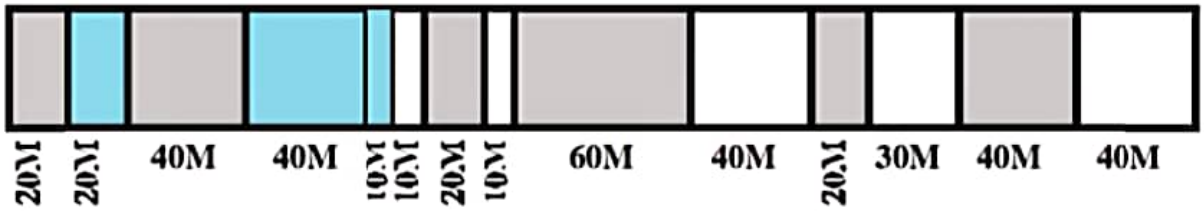
a. First-fit

b. Best-fit

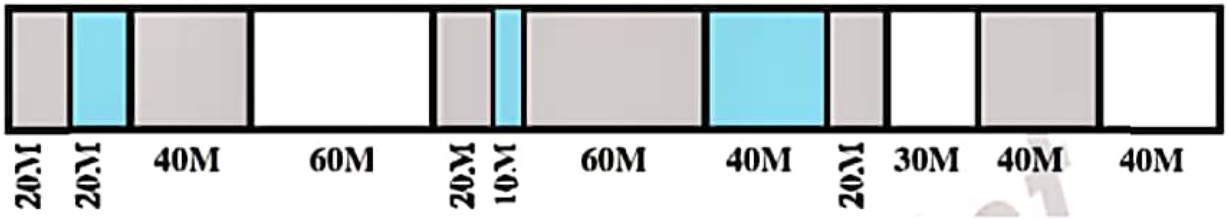
c. Next-fit. Assume the most recently added block is the first 40M block.

Answer:

a. The 40 M block fits into the second hole, with a starting address of 80M. The 20M block fits into the first hole, with a starting address of 20M. The 10M block is placed at location 120M.



b. The three starting addresses are 230M, 20M, and 160M, for the 40M, 20M, and 10M blocks, respectively.



c. The three starting addresses are 80M, 120M, and 160M, for the 40M, 20M, and 10M blocks, respectively.

Ex 3

A 1-Mbyte block of memory is allocated using the buddy system.

a. Show the results of the following sequence in a figure similar to Figure 7.6 (slide 21 in chapter 6): Request 70; Request 35; Request 80; Return A; Request 60; Return B; Return D; Return C.

b. Show the binary tree representation after the operation of Return B.

Ex 4

Consider a simple paging system with the following parameters: 2^{32} bytes of physical memory; page size of 2^{10} bytes; 2^{16} pages of logical address total space.

a. How many bits are in a logical address?

b. How many bytes in a frame?

c. How many bits in the physical address specify the frame?

d. What is the maximum possible number of entries in the page table?

e. How many bits in each page table entry? Assume each page table entry contains a valid/invalid bit.

Answer:

a. The number of bytes in the logical address space is (2^{16} pages) × (2^{10} bytes/page) = 2^{26} bytes. Therefore, 26 bits are required for the logical address.

b. A frame is the same size as a page, 2^{10} bytes.

c. The number of frames in main memory is (2^{32} bytes of main memory)/(2^{10} bytes/frame) = 2^{22} frames. So 22 bits are needed to specify the frame.

d. There is one entry for each page in the logical address space. Therefore there are 2^{16} entries.

e. In addition to the valid/invalid bit, 22 bits are needed to specify the frame location in main memory, for a total of 23 bits.

Ex 5

In a simple segmentation system, a certain process has the following segment table:

Starting Address	Length (bytes)
660	248
1752	422
222	198
996	604
NULL	370

For each of the following logical addresses, determine the physical address or indicate if a segment fault occurs:

a. 0, 198

b. 2, 530

c. 4, 110

Answer:

a. Segment 0 starts at location 660. With the offset, we have a physical address of 660 + 198 = 858

b. Segment 2 has a length of 198 bytes, so this address triggers a segmentation fault since the offset is greater than the segment size.

c. Segment 4 has NULL in its starting address, which means that it doesn't exist in main memory, so this address triggers a segmentation fault.

Ex 6

Suppose the page table for the process currently executing on the processor looks like the following figure. Page numbers start from zero and all addresses are memory byte addresses. The page size is 1024 bytes. The total number of frames in main memory is 16, and the maximum size of the process in virtual memory is 8KB.

Present bit	Modify bit	Page frame number
1	0	0100
1	1	0111
0	0	—
1	0	0010
0	0	—
1	1	0000

What physical address (or Page Fault) would each of the following logical addresses correspond to?

a. 0010000011100 = 1052 in decimal

b. 0100010101101 = 2221 in decimal

c. 1010101111011 = 5499 in decimal

Answer:

a. 1052 = 1024 + 28 maps to VPN 1 in PFN 7, (7 × 1024+28 = 7196) = 01110000011100

b. 2221 = 2 × 1024 + 173 maps to VPN 2, page fault

c. 5499 = 5 × 1024 + 379 maps to VPN 5 in PFN 0, (0 × 1024+379 = 379) = 00000101111011

Ex 7

Consider a memory management system that uses simple paging mechanism where each process has a single page table. Assume that the necessary page table is always in main memory.

a. If a memory reference takes 200 ns, how long does it take to translate a logical address into a physical address and obtain the data from the physical address in main memory (which is called the Effective Memory Access Time or EMAT)? Note that a memory reference happens each time the processor accesses any location in main memory.

b. Now we add a Memory Management Unit (MMU) that contains a TLB cache. The MMU adds an overhead of 20 ns to search in the TLB regardless whether the search results in a hit or a miss. If we assume that 85% of all memory references in the MMU result in a TLB hit, what is the Effective Memory Access Time (EMAT)?

c. Explain how the TLB hit rate affects the EMAT.

Answer:

a. 400 nanoseconds: 200 to get the page table entry, and 200 to access the memory location.

b. This is a familiar effective time calculation: In case of a hit we need a total time of 20 to get physical address + 200 to get data. In case of a miss we need 200 to access TLB, then 200 to get physical address from page table, then 200 to get data. Hence:

$$\text{EMAT} = (220 \times 0.85) + (420 \times 0.15) = 250 \text{ ns}$$

Two cases: First, when the TLB contains the entry required. In that case we pay the 20 ns overhead on top of the 200 ns memory access time. Second, when the TLB does not contain the item, then we pay an additional 200 ns to get the required entry into the TLB.

c. The higher the TLB hit rate is, the smaller the EMAT is, because the additional 20 ns penalty to get the physical address from the TLB is much less than the 200 ns to get the physical address from memory. Hence, if hypothetically we have 100% TLB hit, the minimum possible EMAT will be equal to (220 x 1) + (420 x 0) = 220 ns. (This is hypothetically of course because we can never reach a 100% TLB hit unless the TLB size is equal to all page tables).

Ex 8

Assume that a process is divided into four equal-sized segments and that the O.S. divides each segment into exactly eight pages. Thus, the O.S. uses a combination of segmentation and paging. Assume also that the page size is 2 Kbytes.

a. What is the maximum size of each segment?

b. How many number-of-bits are needed for the logical address?

c. Assume that an element in physical location 00021ABC (hexadecimal) is accessed by this process. What is the format of the logical address that was translated to get this physical address?

Answer:

a. 8 × 2K = 16 Kbytes

b.

Maximum size of process = 16K × 4 = 64 Kbytes = 2^{16}.

Maximum offset inside segment = 16KB = 2^{14}. Hence, number of bits needed for segment offset = 14.

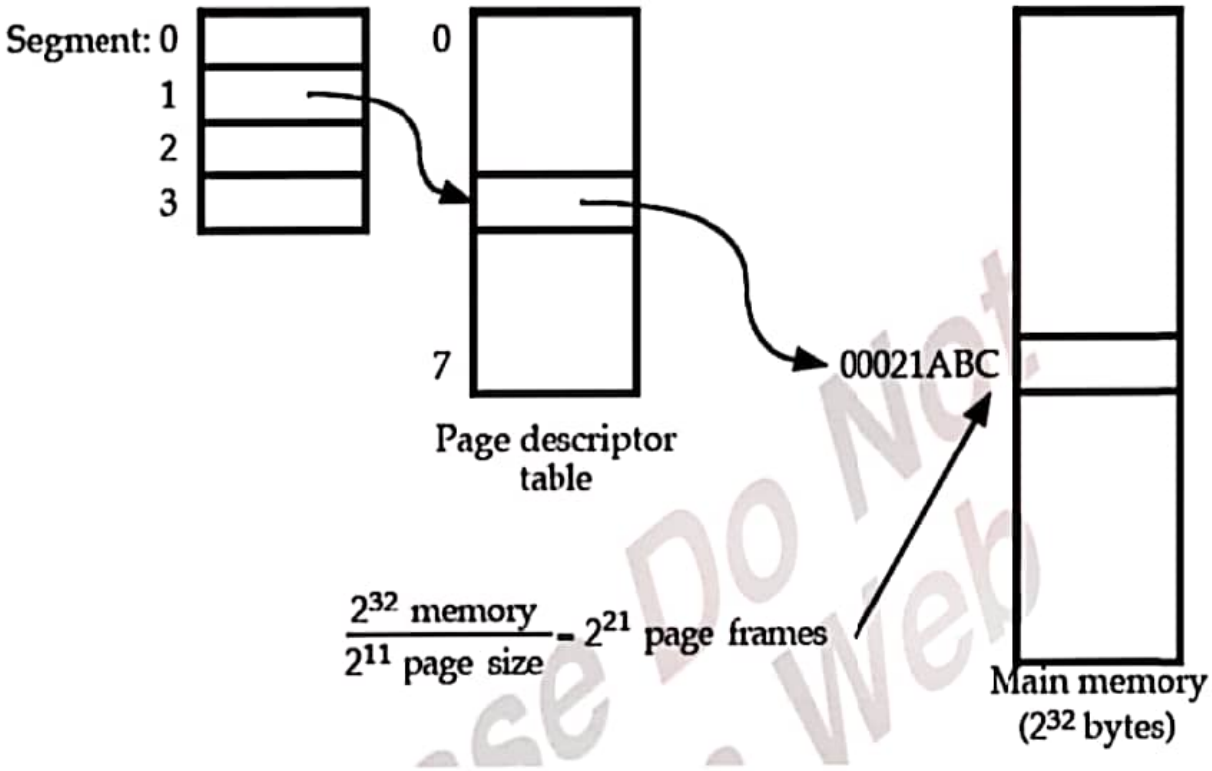
Maximum number of segments = 4 = 2^2. Hence, the number of bits needed for segment number is 2.

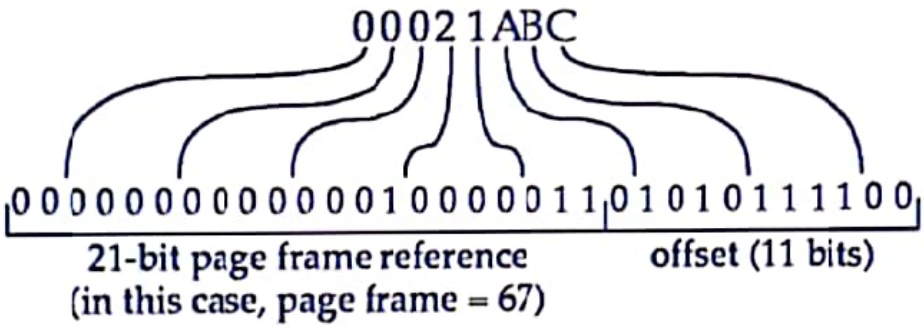
Therefore, the total number of bits needed for logical address = 14+2 = 16.

c. Number of bits in the main memory address (00021ABC) = 8 x 4 = 32 bits

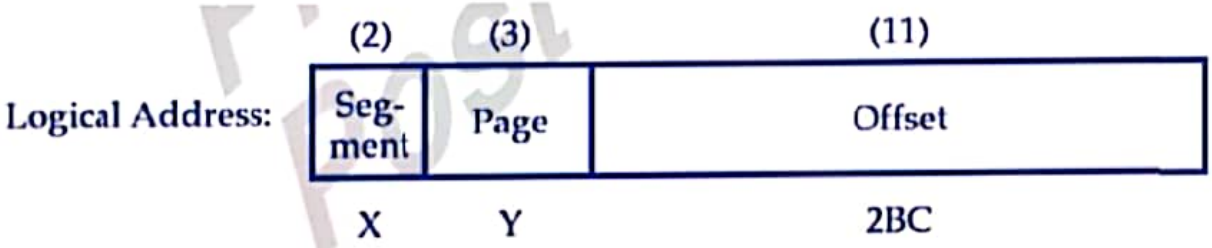
- Size of main memory = 2^{32} = 4 GBytes

- Number of frames in main memory = 2^{32} /2^{11} (page size is 2^{11}) = 2^{21} frames





Format of the logical address: 2 bits for segment number followed by 3 bits for page number followed by 11 (01010111100) bits for Offset:

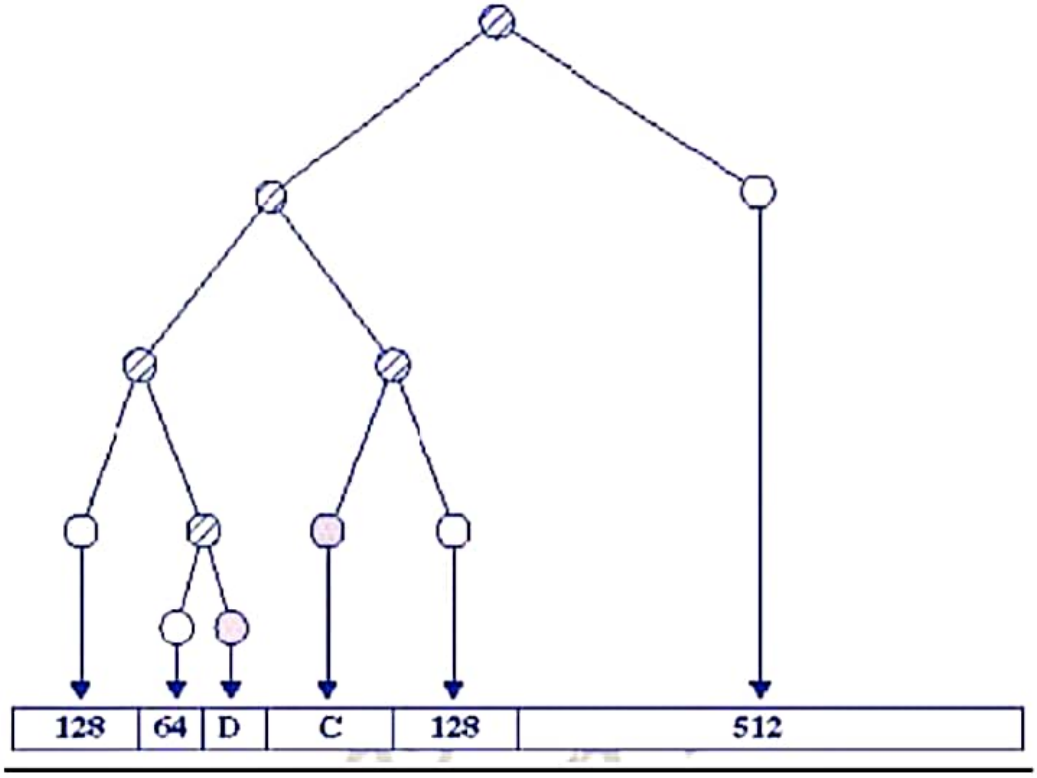


Answer:

a.

Request 70	A	128		256		512
Request 35	A	B	64	256		512
Request 80	A	B	64	C	128	512
Retum A	128	B	64	C	128	512
Request 60	128	B	D	C	128	512
Return B	128	64	D	C	128	512
Retum D	256			C	128	512
Retum C	1024					

b.



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
Initial state	1024 KB								
65 KB request => A	A	128		256		512			
30 KB request => B	A	B	32	64	256		512		
90 KB request => C	A	B	32	64	C	128	512		
34 KB request => D	A	B	32	D	C	128	512		
130 KB request => E	A	B	32	D	C	128	E	256	
Free C	A	B	32	D	256		E	256	
Free B	A	64		D	256		E	256	
275 KB request => F	Not Available → wait								
145 KB request => G	A	64		D	G		E	256	
Free D	A	128			G		E	256	
Free A	256				G		E	256	
Free G	512						E	256	
Free E	1024								

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