CS342 - Homework #2

Berk Gakar - Section 2 - 22003021

Solution to Question 1

Total Resource Matrix → [15, 6, 9, 10]

Based on the given matrices we can construct the need matrix:

	A	B	C	D
P_0	7	5	3	4
P_1	2	1	2	2
P_2	3	4	4	2
P_3	2	3	3	1
P_4	4	1	2	1
P_5	3	4	3	3

Moreover, the available resource matrix becomes → [6, 3, 5, 4]
Now check the case for each process

→ P_1 can be satisfied, the available matrix → [6, 4, 6, 5]
→ P_2 can be satisfied, " " " → [10, 5, 6, 7]
→ P_3 can be satisfied, " " " → [11, 5, 6, 8]
→ P_4 can be satisfied, " " " → [12, 6, 6, 8]
→ P_5 can be satisfied, " " " → [13, 6, 7, 9]
→ P_0 can be satisfied, " " " → [15, 6, 9, 10]

Hence, the current state is safe.

Suppose that we granted P5(3,2,3,3), the new need matrix is =>

	A	B	C	D
P0	7	5	3	4
P1	2	1	2	2
P2	3	4	4	2
P3	2	3	3	1
P4	4	1	2	1
P5	0	2	0	0

Available resources matrix → [3, 1, 2, 1]

As can be seen granting P5(3,2,3,3) causes all processes to become deadlocked. So P5(3,2,3,3) should not be granted.

Solutions to Question 2

a) 2 physical memory accesses required. Hence:

$$EAT = 150\,ns * 2 = 300\,ns$$

b) In the case of a TLB hit, access time = 150 + 10 = 160 ns
In the case of a TLB miss, access time = 150 + 150 + 10 = 310 ns
Hence, $EAT = (0.85)\,160 + (0.15)\,310 = 136 + 46.5 =$ **182.5 ns**

c) In two level paging, if there is a TLB miss access time becomes → 150 + 150 + 150 + 10 = 460 ns. The other case's access time remains the same
Hence, $EAT = (0.85)\,160 + (0.15)\,460 = 205\,ns$

Solutions to Question 3

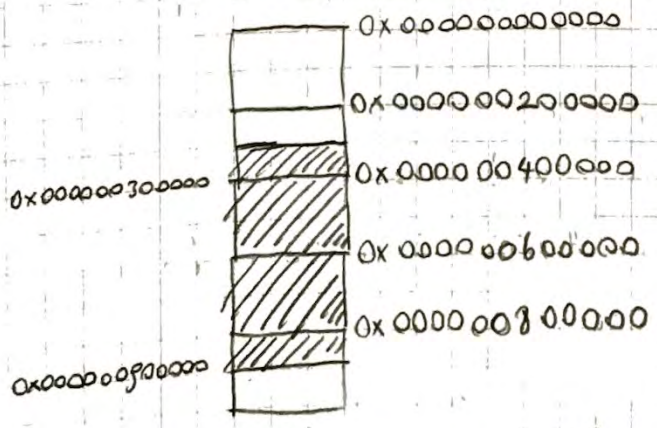
a) $48 - 12 = 36$, so 36 bits are required to store a frame number
(12 → offset)

b) Since the offset is 12 bits, page size is 2^{12} bytes, which makes 4 KB

A fourth level table can map $2^9 * 2^{12} = 2^{21}$ bytes = 2 MB
A third level table can map $2^9 * 2^{21} = 2^{30}$ bytes = 1 GB
A second level table can map $2^9 * 2^{30} = 2^{39}$ bytes = 512 GB

Since the virtual memory addresses are 48 bits and a fourth level table is 2 MB (uses the last 21 bits of the virtual address), we can say that the following address ranges are mapped to the same fourth level table: 0x000000000000 – 0x000000200000, 0x000000200000 – 0x000000400000 ...

Hence, if we draw our virtual memory



6 MB is mapped as shown.
Hence, we need 4 fourth level, 1 third level, 1 second level and 1 first level table.
In total 7 tables are required

A table requires $2^9 * 8$ bytes = 4KB of memory
Hence we need 16 KB of memory for fourth level, 4 KB for third, 4 KB for second, and 4 KB for first level table. In total 28 KB of memory is required

c)

First level table = 1 table is required

Second level table = A second level table can map 512 GB via 0x8000000000. Hence, the code segment and the data segment are in the first block. However, the stack part is in 30th block. As a result, [2] 2nd level tables are required.

Third level table = A third level table can map 1 GB via 0x40000000. The code segment is in the first block. However, the data segment starts from block 32 and takes up 2 GB. Hence 1 third-level table is needed for code segment and 2 third-level table is needed for data segment. Lastly we need another third level table for the stack. Hence, we need 4 third-level tables in total.

Fourth level table = 2 MB is mapped by 0x200000. Code segment is 128 KB and it is enough to have 1 table. For the data segment, 2048 / 2 = 1024 tables are required. For the stack, 4/2 = 2 tables are required. In total 1 + 1024 + 2 = 1027 fourth level tables are required.

How much memory is consumed?

First-level page tables: 1 * 4 = 4KB

Second-level page tables: 2 * 4 = 8KB

Third-level page tables: 4 * 4 = 16 KB

Fourth-level page tables: 1027 * 4 = 4108 KB

In total: 4136 KB

Solutions to Question 4

2: 2(1) f

4: 2(1) 4(1) f

1: 2(1) 4(1) 1(1) f

3: 3(1) 4(0) 1(0) f

4: 3(1) 4(1) 1(0)

—— clear ——

6: 3(0) 6(1) 1(0) f

3: 3(1) 6(1) 1(0)

6: 3(1) 6(1) 1(0)

1: 3(1) 6(1) 1(1)

2: 3(0) 6(0) 2(1) f

—— clear ——

1: 1(1) 6(0) 2(0) f

5: 1(1) 5(1) 2(0) f

2: 1(1) 5(1) 2(1)

5: 1(1) 5(1) 2(1)

4: 1(0) 5(0) 4(1) f

—— clear ——

1: 1(1) 5(0) 4(0)

2: 1(0) 2(1) 4(0) f

4: 1(0) 2(1) 4(1)

3: 3(1) 2(1) 4(0) f

b)

2: 2 f

4: 2 4 f

1: 2 4 1 f

3: 3 4 1 f

4: 3 4 1

6: 3 6 1 f

3: 3 6 1

6: 3 6 1

1: 3 6 1

2: 3 2 1 f

1: 3 2 1

5: 5 2 1 f

2: 5 2 1

5: 5 2 1

4: 4 2 1 f

1: 4 2 1

2: 4 2 1

4: 4 2 1

3: 4 2 3 f (smallest page number is replaced)

Note: To save up space, the frames are represented horizontally

i.e., 

→ 1 2 3

Solutions to Question 5

=> FCFS

5200 - 4500 = 700
5200 - 2000 = 3200
8500 - 2000 = 7500
8500 - 4300 = 5200
4300 - 1500 = 2800
5100 - 1500 = 3600
8600 - 5100 = 4500
8600 - 4000 = 5600
4600 - 4000 = 600

Total = 33700

=> SCAN

1500 2000 4000 4300 4500 [>>>] 4600 5100 5200 8500 8600

(8888 - 4500) + (8888 - 1500) = 13388

=> SSTF

4500 → 4600 → 4300 → 4000 → 5100 → 5200 → 2000 → 1500 → 8500 → 8600

100 300 300 1100 100 3200 500 8000 100

100 + 300 + 300 + 1100 + 100 + 3200 + 500 + 8000 + 100 = 13700

Solutions to Question 6

a) Transfer time for 4KB: 4 KB / 30MB/s $\cong$ 0.13 ms

3600 RPM means that 1 rotation takes $\frac{1}{60}$ s.

Average rotational latency is $\frac{1}{120}$ s $\cong$ 8.3 ms

Then, total time to transfer 4 KB = 8.3 + 4 + 0.13 = 12.43

Number of I/O per second = 80 (1000 / 12.43)

Throughput = (80 × 4) / 1024 = 0.3125 MB/s

b) 512 * 4 KB / 30 MB/s $\cong$ 66.56 ms

Time for I/O = 8.3 + 4 + 66.56 = 78.86 ms

Number of I/Os per second = 12 (1000 / 78.86)

Throughput = 24 MB/s

Solutions to Question 7

a) An inode can store $2^{12}/2^3 = 512$ pointers.

The total file size:

$10 + 512 + 512^2 + 512^3 = 134480394 \cong 2^{27}$

Hence,

$2^{27} * 2^{12} = 2^{39}$ bytes = 512 GB

b) A leaf index block is able to map $2^9 * 2^{12} = 2^{21} = 2$ MB

8 GB $\rightarrow 2^{33}$ bytes $2^{33}/2^{21} = 2^{12} = 4096$ leaf nodes

Hence, we need to use the single level index block.

All of the 512 second level index blocks will be used.

Remaining blocks will come from the third level index blocks. 10 direct pointers do not change the number of used second level index tables

$4096 - 513 = 3583$ $3583/512 = 7$ → number of second level index tables from three level

in total = 7 + 512 = 519

c) i. $2^{15} / 2^{12} = 2^3 = 8$. The offset 2^{15} is in block 8. 1 disk access is needed to retrieve the block.

ii. $2^{23} = 8$ MB. A single index node can map $2^{12} * 2^9 = 2^{21} = 2$MB which is not sufficient. Hence, two level index structure is required since it can map 1 GB. So, we need to access the first level index block of the two level index, then a second level index block at this structure and then the data block. in total, we need 3 disk accesses

iii. $2^{32} = 4$GB. A third level index block is required since it can map 512 GB. Using similar logic given in ii, 4 disk accesses are required.