Operating Systems–2: CS3523 2019
Theory Assignment 2:
Homework on Memory Management, Virtual Memory and Flash memory architecture

Sai Harsha Kottapalli
CS17BTECH11036

1. Page number = floor(4095 / (2 * (2^{10})) = 1
   Offset = address – (page_number * page_size) = 4095 – (1 * 2 * 2^{10})
   = 2047
2. Number of entries in inverted page table
   = physical address space / page size.
   Physical address space = 2^{24} bytes
   Page size = 2KB = 2^{11} bytes
   So, Number of entries = $2^{24-11} = 2^{13.}$

3. We have 512KB chunk available and kernel requests 57KB.
   - We split it into A_L and A_R of 256KB each.
   - Then we further split A_L into B_L and B_R of 128KB each.
   - Then we further split B_L into C_L and C_R of 64KB each, and one of them is used to satisfy the request.
   - (Upon further division, C_L is split into 32KB chunks which cannot satisfy the request)
4. a) 345 < 420, so it is a valid address
   physical address = 239 + 345 = 584
   b) 666 is not less than 555, so it is an invalid address
   c) 876 < 1400, so it is a valid address
   physical address = 5450 + 876 = 6326

5.

Segment	Base	Length
0	1100	700
1	9350	550
2	5600	600
3	2200	3400
4	6200	2500

6. Page length register is used to store length of page.
We can use this register to check if address is valid as we know page size , we compare if it is in the range of page starting address and page starting address + its length.

7. The outer most page table has 536870912 entries which is 2^{29} entries.
So,29 bits
The second-level page table has 8192 entries which is 2^{13} entries.
So, 13 bits
The third level page table has 512 entries which is 2^{9} entries.
So, 9 bits
The fourth-level page table has 64 entries which is 2^{6} entries.
So, 6 bits
Total bits required for page number = 29 + 13 + 9 + 6 = 57
Bits for page offset = logical address space in bits – bits required for page number
= 64 -57 = 7

8. <u>BEST FIT</u>
   a. Search through the free memory partitions - 320 KB, 580 KB, 480 KB, 220 KB, 890 KB, 600 KB and 155 KB
   Allots to the 155 KB Hole as it the next highest size after 135 KB.
   b. Search through the free memory partitions - 320 KB, 580 KB, 480 KB, 220 KB, 890 KB, 600 KB and 20 KB
   Allots to the 890 KB Hole as it the next highest size after 650 KB.
   c. Search through the free memory partitions - 320 KB, 580 KB, 480 KB, 220 KB, 240 KB, 600 KB and 20 KB
   Allots to the 480 KB Hole as it the next highest size after 398 KB.

d. Search through the free memory partitions - 320 KB, 580 KB, 82 KB, 220 KB, 240 KB, 600 KB and 20 KB
Allots to the 220 KB Hole as there is a hole of size 220 KB.

e. Search through the free memory partitions - 320 KB, 580 KB, 82 KB, 0 KB, 240 KB, 600 KB and 20 KB
Allots to the 580 KB Hole as it the next highest size after 520 KB.

f. Search through the free memory partitions - 320 KB, 60 KB, 82 KB, 0 KB, 240 KB, 600 KB and 20 KB
Allots to the 600 KB Hole as it the next highest size after 440 KB.

<u>WORST FIT</u>

a. Search through the free memory partitions - 320 KB, 580 KB, 480 KB, 220 KB, 890 KB, 600 KB and 155 KB
Allots to the 890 KB Hole as it is the largest hole.

b. Search through the free memory partitions - 320 KB, 580 KB, 480 KB, 220 KB, 755 KB, 600 KB and 155 KB
Allots to the 755 KB Hole as it is the largest hole.

c. Search through the free memory partitions - 320 KB, 580 KB, 480 KB, 220 KB, 105 KB, 600 KB and 155 KB
Allots to the 600 KB Hole as it is the largest hole.

d. Search through the free memory partitions - 320 KB, 580 KB, 480 KB, 220 KB, 105 KB, 202 KB and 155 KB
Allots to the 580 KB Hole as it is the largest hole.

e. Search through the free memory partitions - 320 KB, 360 KB, 480 KB, 220 KB, 105 KB, 202 KB and 155 KB
Can’t Allot the process with 520KB as the 480 KB is the largest hole. So it is not allocated

f. Search through the free memory partitions - 320 KB, 360 KB, 480 KB, 220 KB, 105 KB, 202 KB and 155 KB
Allots to the 480 KB Hole as it is the largest hole.

9.

0	5	INVALID	4	INVALID
6	INVALID	2	1	INVALID
FREE	8	3	7	INVALID

(Only pages is 0-index based, rest i.e. blocks are 1-index based)
After write operation of 2^{nd} write operation on 0^{th} page, for the 3^{rd} write operation on the 5^{th} page in logical space an erase was performed on 2^{nd} block.
After write operation of 3^{rd} write operation on 8^{th} page, for the 3^{rd} write operation on the 2^{nd} page in logical space an erase was performed on 3^{rd} block.
After write operation of 2^{nd} write operation on 3^{rd} page, for the 2^{nd} write operation on the 4^{th} page in logical space an erase was performed on 4^{th} block.
After write operation of 2^{nd} write operation on 7^{th} page, for the 3^{rd} write operation on the 0^{th} page in logical space an erase was performed on 1^{st} block.
So, there were 4 erase operations.

10. Table Size = number_of_logical_pages * log(number_of_physical_pages)
Number of logical pages = 64 GB / 4 KB = $64 * 2^{30} / 4 * 2^{10}$ = 2^{24}
As, Number of physical pages is 4 times the logical pages.
Number of physical pages = 2^{26}
Table Size = $2^{24} * \log(2^{26})$