OS-2 Assignment

V. Sri Charan Reddy
CS16BTECH11044

1) $3095_{(10)} \equiv 110000010111_{(2)}$

$2kB \Rightarrow$ last $1+10=11$ digits are page offset (memory access has byte level granularity)

Given address has 12 digits

Page number $=1$ Page offset $= 10000010111$

2) Page size $= 4kB \Rightarrow$ last $2+10$ digits i.e last 12 digit are page offset (memory access has byte level granularity)

Number of entries in inverted page table is same as number of physical frames.

Physical address space is 24 bits, last 12 bits are page offset

$24-12=12$ bits are page number bits i.e there are ~~[illegible]~~

$2^{12} = 2048$ entries in the inverted page table

3) (a) 0,345

$229 + 345 = 574$

(b) 3,666

$666 > 555$ Hence it gives rise to a memory trap

(c) 2,876

$5500 + 876 = 6376$

4) For processes with pending I/O, swapping can't be directly be done as I/O would occur at wrong location. So, pending I/O processes are always transferred to kernel space, I/O is completed, and then the process is swapped out. This is known as double buffering. It adds overhead.

5)

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

6) Page-table length register (PLTR) indicates the size of page table. This value is checked against every logical address to verify that the address is in the valid range for the process. Failure of this test causes an error trap to the operating system

7)

Levels	Bits
1	29
2	13
3	9
4	6

29	13	9	6	7
1 level table	2 level table	3 level	4 level	offset

As first 57 bits are for page number, 7 bits can be used for page offset (7+57=64 bits)

8) Best Fit:- Increasing order:

155, 220, 310, 380, 580, 600, 790 //125 → 155

30, 220, 310, 380, 580, 600, 790 //550 → 580

30, 30, 220, 310, 380, 600, 790 //378 → 380

2, 30, 30, 220, 310, 600, 790 //210 → 220

2, 10, 30, 30, 310, 600, 790 //510 → 600

2, 10, 30, 30, 90, 790 //400 → 790

2, 10, 30, 30, 90, 390

We are able to accomodate all the given memory requirements with suitable holes

Worst Fit:-

Decreasing order:-

790, 600, 580, 380, 310, 220, 155 //125 → 790

665, 600, 580, 380, 310, 220, 155 //550 → 665

600, 580, 380, 310, 220, 155, 115 //378 → 600

580, 380, 310, 222, 220, 155, 115 //210 → 580

380, 370, 310, 222, 220, 155, 115 // 510

No big enough hole to accomodate 510 kB and 400 kB even though the total free mem is 1762 which is > (510+400)

9) (a) When the page table is also stored in main memory, two memory accesses are needed to access a byte (one for the page-table entry, one for byte). Thus memory access is slowed by a factor of 2.
Hence, 300 ns are required for paged memory reference

(b) TLB check is done in instruction pipeline. Hence for a TLB hit there will be no penalty i.e, 150 ns is taken
However, for a TLB miss, Page table reference is made hence 2* 150ns is taken $0.75*150 + 0.25*300 = 187.5$ ns

10) 4mplies $P/2 + 1$ memory accesses are made to find a particular element in memory (avg).