## **Part 1: Memory Management**

The diagram on the 2nd page shows a logical address space. It has a 5-bit addressing scheme - allowing 32 possible locations. The logical address is broken up into two segments. The higher order 3 bits represent the page number. The lower-order two bits represent the page offset. This addressing scheme gives us 8 pages per process, with a page size of 4 bytes. Notice, the physical memory contains 16 total page frames, I have put a gap between 5 and 13 just to save space.

- 1) Using the diagram below, find the data corresponding to logical addresses 2, 10, 21, and 30. (10 Points)
- 2) Alternatively, we could have split the 5-bit addressing scheme differently, using the 2 higher order bits for the page index and the 3 lower-order bits for the page offset. What size page frames would we have? How many pages in the page table? (5 Points)
- 3) Assume that you have 32MB of main memory, divided into 8 4MB frames. Now assume that you have 3 processes. Process 1 is 12MB, Process 2 is 5 MB, and Process requires 15 MB. Can we fit all 3 processes in memory using our standard paging technique? If we expand our main memory to be 48MB, how much memory is left for other processes? (5 Points)
- 4) What is the purpose of paging the page tables? (5 Points)
- Assume we have a 3-tiered paging system, consisting of an outer page table that translates a logical page number into an index into a second set of inner page tables. The inner page table then provides an index into a third set of page tables that actually have the frame number for a given logical address. Also assume that we have a TLB hit rate of 85% (and that a TLB hit will directly translate a logical address into a frame number + offset). Assume that each main memory access takes 120ns and each TLB access takes 25ns. What is the effective memory access time of the system? (10 Points)

## Part 2: Virtual Memory

Assume a program has been allocated <u>four page frames and uses eight pages</u>. After observing the program run, the memory addresses used follow the following sequence:

Reference String: 0, 1, 7, 2, 3, 2, 7, 1, 0, 3

- a) Calculate the number of page faults (assuming all frames were initially empty) for FIFO page replacement. **(5 points)**
- b) LRU? (5 points)
- c) Optimum Page Replacement algorithm (5 points)

Logical Address Space										
Decimal	Binary									
0	0	0	0	0	0					
1	0	0	0	0	1					
2	0	0	0	1	0					
3	0	0	0	1	1					
4	0	0	1	0	0					
5	0	0	1	0	1					
6	0	0	1	1	0					
0 1 2 3 4 5 6 7 8 9	00000000000	0	1		1					
8	0	1	0	1 0	0					
9	0	1	0	0	1					
10	0	1	0	1	0					
11	0	1	0	1	1					
12	0	1	1	0	0					
13	0	1	1	0	1					
14	0	1	1	1	0					
15	0	1	1	1	1					
16	1	0	0	1 0 0	0					
17	1	0	0	0	1					
18	1	0	0	1	0					
19	1	0	0	1	1					
20	1	0	1	0	0					
21	1	0	1	1 0 0	1					
22	1	0	1	1	0					
23	1	0	1	1	1					
24	1	1	0	0	0					
25	1	1	0	0	1					
26	1	1	0	1	0					
27	1	1	0	1 1 0	0101010101010101010101010101010					
28	1	1	1	0	0					
29	1 1	1	1	0	1					
30		1			0					
31	1	1	1	1	1					

PAGE TABLE							
Page #	Frame #						
(Decimal)	(Decimal)						
0	5						
1	13						
2	14						
3	3						
4	0						
5	4						
6	1						
7	2						

Main Memory (RAM)									
Frame #	Physical Address (Decimal)	Physical Address (Binary)					Data (Decimal)		
0	0	0	0	0	0	0	0	868	
	1	0	0	0	0	0	1	983	
	2	0	0	0	0	1	0	780	
	3	0	0	0	0	1	1	943	
1	4	0	0	0	1	0	0	946	
	5	0	0	0	1	0	1	28	
	6	0	0	0	1	1	0	829	
	7	0	0	0	1	1	1	266	
2	8	0	0	1	0	0	0	612	
	9	0	0	1	0	0	1	246	
	10	0	0	1	0	1	0	911	
	11	0	0	1	0	1	1	893	
3	12	0	0	1	1	0	0	322	
	13	0	0	1	1	0	1	393	
	14	0	0	1	1	1	0	573	
	15	0	0	1	1	1	1	557	
4	16	0	1	0	0	0	0	463	
	17 18	0	1	0	0	0	1	514 431	
	19	0	1	0	0	1	0	118	
5	20	0	1	0	1	0	0	316	
'	21	ľ	1	0	1	0	1	949	
	22	ľ	1	0	1	1	0	400	
	23	0	1	0	1	1	1	932	
13	52	0	1	0	1	0	0	45	
	53	0	1	0	1	0	1	852	
	54	0	1	0	1	1	0	647	
	55	0	1	0	1	1	1	758	
14	56	0	1	1	0	0	0	317	
	57	0	1	1	0	0	1	622	
	58	0	1	1	0	1	0	867	
	59	0	1	1	0	1	1	425	
15	60	0	1	1	1	0	0	759	
	61 62	0	1	1	1	0	1	98 140	
	63	0	1	1	1	1	1	703	
	03	U	1	1	1	1	1	/03	