CSCI 3453: Operating System Concepts
HW Assignment # 5
Due Date: April 4, 2021 @ 11:55 PM

1) Explain the difference between internal and external fragmentation.
Internal fragmentation: Allocated memory may be slightly larger than requested memory. Size difference is memory internal to a partition, but not being used Break the physical memory into fixed sized blocks and allocate memory in units based on block size. Memory is allocated to a process that may be slightly larger than request memory

External fragmentation: as files are allocated and deleted, free storage is broken into chunks or Total memory space exists to satisfy a request, but not contiguous it exists when there is not enough total memory space to satisfy a request but the available spaces are not contiguous. Storage is fragmented into large number of small holes

2) Given six memory partitions of 100 MB, 170 MB, 40 MB, 205 MB, 300 MB, and 185 MB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 200 MB, 15 MB, 185 MB, 75 MB, 175 MB, and 80 MB (in order)? Indicate which—if any—requests cannot be satisfied.

Process 200 MB, 15 MB, 185 MB, 75 MB, 175 MB, 80 MB
Memory (M1)100 MB, (M2)170 MB, (M3)40 MB, (M4)205 MB, (M5)300 MB, (M6)185 MB

FIRST FIT - goes through the list and finds the first one that it can fit in
Process1 200mb -> M4 205 mb, leaving a space of 5 mb extra
Process2 15mb -> M1 100 mb, leaving 85
Process3 185 mb -> M5 300 Mb, leaving 115
Process4 75 mb -> M1, leaving 10
Process5 175 mb -> M6 185mb. leaving 10
Process6 80 mb -> M3, 115-80 = 35

Memory (M1)100 MB, (M2)170 MB, (M3)40 MB, (M4)205 MB, (M5)300 MB, (M6)185 MB

BEST FIT - finds the closest memory size to its process
Process1 200mb -> M4, leaving 5
Process2 15mb -> M3, leaving 35
Process3 185 mb -> M6, leaving 0
Process4 75 mb -> M1, leaving 25
Process5 175 mb -> M5, leaving 125
Process6 80 mb -> M5, leaving 45

Memory (M1)100 MB, (M2)170 MB, (M3)40 MB, (M4)205 MB, (M5)300 MB, (M6)185 MB

WORST FIT - find the largest free one
Process1 200mb -> M5, leaving 100
Process2 15mb -> M4, leaving 205-15 = 190
Process3 185 mb -> M4, leaving 5
Process4 75 mb -> M6, leaving 110
Process5 175 mb -> cannot be allocated
Process6 80 mb -> M2, leaving 90

3) Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers)?

page size = 2^n = 1kb = 1024 = 2^{10}
offset = 10

1) 21205
Binary = 10100 1011010101
Page # = 10100 = 20
Offset = 1011010101 = 725

2) 164250
Binary = 10100000 0110011010
Page # = 10100000 = 160
Offset = 0110011010 = 410

3) 121357
Binary = 1110110 1000001101
Page # = 1110110 = 118
Offset = 1000001101 = 525

4) 16479315
Binary = 11111011011101 0001010011
Page # = 11111011011101 = 16093
Offset = 0001010011 = 83

5) 27253187
Binary = 110011111110110 0111000011
Page # = 110011111110110 = 26614
Offset = 0111000011 = 451

4) Consider a logical address space of 2,048 pages with a 4-KB page size, mapped onto a physical memory of 512 frames.

1) How many bits are required in the logical address?

logical address space = 2^m

logical address space = # of pages x page size

= 2048 x 4kb(4096)

= 8388608 = $2^{23} \rightarrow m = 23$

2) How many bits are required in the physical address?

physical address space = 2^m

physical address space = # of frames x page size

= 512 x 4kb(4096)

= 2097152 = $2^{21} \rightarrow m = 21$

5) Consider a paging system with the page table stored in memory.

1) If a memory reference takes 50 nanoseconds, how long does a paged memory reference take?

2 memory accesses = 2* 50 ns = 100 ns

2) If we add TLBs, and if 75 percent of all page-table references are found in the TLBs, what is the effective memory reference time? (Assume that finding a page-table entry in the TLBs takes 2 nanoseconds, if the entry is present.)

.75 * (TLB(hit) + 2ns(finding page)) + .25 * (TLB(miss) + 2ns(finding page)) =

.75(50+2) + .25(100+2) + 2 = 64.5

6) The following is a page table for a system with 12-bit virtual and physical addresses and 256-byte pages. Free page frames are to be allocated in the order 9, F, D. A dash for a page frame indicates that the page is not in memory.

Page	**Page Frame**
0	0 x 4 = 0100
1	0 x B = 1011
2	0 x A = 1010
3	0 x F –
4	0 x 9 –
5	0 x 2 = 0010
6	0 x D –
7	0 x 0 = 0000
8	0 x C = 1100
9	0 x 1 = 0001

Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. All numbers are given in hexadecimal. In the case of a page fault, you must use one of the free frames to update the page table and resolve the logical address to its corresponding physical address.

12 bit virtual

12 bit physical

256 byte pages = 2^8

1) 0x2A1

Binary = 0010 10100001
0010 = 2
Page number 2 → page frame 0xA
Answer = 1010 1010 0001 → 0xA A1

2) 0x4E6

Binary = 0100 1110 0110
0100 = 4
Entry 4 → page frame empty

Offset = 0xE6
Page fault
Frame number be 0x9, since 9 is first in line
Physical Address = 0x9 E6

3) 0x94A
Binary = 1001 0100 1010
1001 = 9
Entry 9 → page frame 0x1
Answer = 0001 0100 1010 → 0x1 4A

4) 0x316
Binary = 0011 0001 0110
0011 = 3
Entry 3 → page frame empty, F is next in line
Offset = 0x16
Physical Address = 0xF 16

7) Apply the (1) FIFO, (2) LRU, and (3) optimal (OPT) replacement algorithms for the following page-reference strings: 2,6,9, 2,4,2,1,7,3,0,5,2,1,2,9,5,7,3,8,5
**ASSUMING 3 frames

FIFO = 385 , 18 page faults
LRU = 538, 17 page faults
OPT = 358, 13 page faults

String	FIFO	Fault	LRU	Fault	OPT	Fault
2	2	1	2	1	2	1
6	26	2	26	2	26	2
9	269	3	269	3	269	3
2	269	x	269	x	269	x
4	469	4	249	4	249	4

2	429	5	249	x	249	x
1	421	6	241	5	219	5
7	721	7	271	6	217	6
3	731	8	371	7	213	7
0	730	9	370	8	210	8
5	530	10	350	9	215	9
2	520	11	250	10	215	x
1	521	12	251	11	215	x
2	521	x	251	x	215	x
9	921	13	291	12	915	10
5	951	14	295	13	915	x
7	971	15	795	14	975	11
3	371	16	735	15	375	12
8	381	17	738	16	385	13
5	385	18	538	17	385	x

Create the text file using Microsoft Word, Open Office Writer, or a text editor. Name the document as follows:
FirstName_LastName_HW5