



BIRZEIT UNIVERSITY

Electrical and Computer Engineering Department  
ENCS339 Operating Systems 1<sup>st</sup> Semester 2018/2019  
Midterm Exam Instructor: Dr. Adnan H. Yahya Time: 90min

Q	ABET	Max	Earned
Q1	e	18	
Q2	e	15	
Q3	a	16	
Q4	c	20	
Q5	c	18	
Q6		20	
Σ		107	

Student Name: \_\_\_\_\_ Student Number: \_\_\_\_\_

Please answer all questions using the exam sheets ONLY.

Please show all steps of your solutions. Max grade is:107.

**Question 1 (18%)** A computer system has only 32GB of **physical memory (RAM)**. The system has a 16KB **page size** and 48-bit logical address space. CPU generated addresses are 6 bytes each[yes: not a power of 2!].

- (a) 2% Indicate on the diagram below which of the bits of the **logical address** of 48 bits are used for page number (**p**) and for offset (**d**). Most significant (MSB) is bit #0 and least significant (LSB) is bit #47  
16KB=2\*\*14 Bytes, thus 14 bits [34:47] are used for offset (displacement) and

0	10	20	30	34	47
				<b>++++Displacement++</b>	

- (b) 2% How many **frames** are there in the RAM?

**RAM is 32GB, each frame is 16KB, # of Frames= 32GB/16KB=2MFrames [addressable using 21 bits]**

- (c). 2% Ignoring page table overhead and OS needs, how many **pages** can a process have (max) to be runnable in **contiguous** memory allocation mode?

**32GB=2MPages (2 mega pages)**

- (d). 2% How many **bits** are **minimally** needed for frame number of this computer in page map tables (PMTs)?  
**21 bits to address the 2MFrames,**

- (e) 2% Given a 4GB Process what is the size of the Page Map Table (PMT) in **bytes** and **pages** if the PMT is flat (one level)?

**Flat means the table has 4GB/16KB= 1/4MPages= 256Kpages. Each page needs 21 bits or 4bytes for addresses of frames for a total of 256x4K=1MB. 1MB =1MB/16KB=2\*\*20/2\*\*14=2\*\*6=64 pages.**

- (f) 2% Given the 4GB Process: how many levels are needed for the PMT using multi-level paging of PMT, if needed?

**First level has 16KB/4Bytes=4KPages=4K\*16KB=64MB.**

**Second level has 4K\*4KPages=16M\*16KB=256GB.**

**So we have only 2 levels.**

- (g)3% With a **two level** paging of PMT, find the maximum size (address space, in bytes) that a job **can** have?  
**256GM, as shown earlier.**

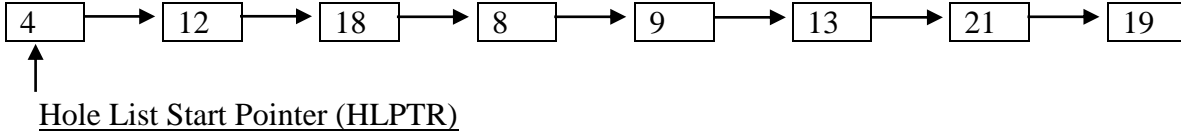
- (h) 3% How many levels of page tables would be required to map a full 48 bit virtual address space (top level: one page max)? Explain.

**2 levels gave 256GB or 2\*\*38B, 3 Levels will give (2\*\*38) x 4K=(2\*\*38) x (2\*\*14)=2\*\*52Bytes; So we need 3 levels.**

**Another way: Each page has 4K entries. Needs 14 bits. So 14bits for displacement, 12bits for first level, 12 for second for a total of 14+12+12=38bits. The last 10 bits are for the third level! Note that size of such job with these 3 levels (last level has only 1K entries out of 4K) is 2\*\*48B= 256TB**

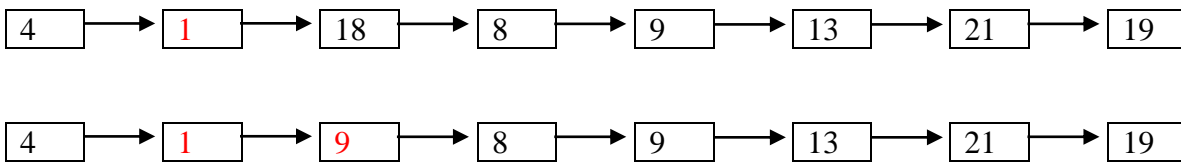


**Question 3 (16%: 4% each)** Consider a dynamic (contiguous) partitioning system in which the (free) memory consists of the following list of **holes** (free partitions), sorted by increasing memory address (all sizes are in Megabytes):

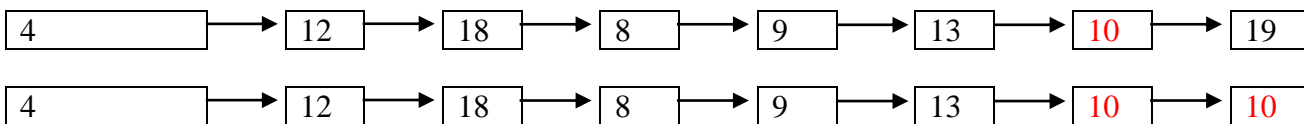


Suppose a new process Pa requiring 11 MB arrives, followed by a process Pb needing 9MB of memory. Show the list of holes **after both of** these processes are placed in memory for each of the following algorithms (start with the original list of holes for each algorithm). Assume that **the hole List Start Pointer is moved to the closest hole** to the allocated (or to the newly created after each allocation): from left to right and **circular**.

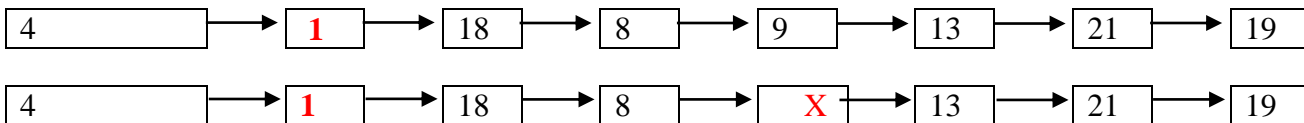
i) First Fit-5%:



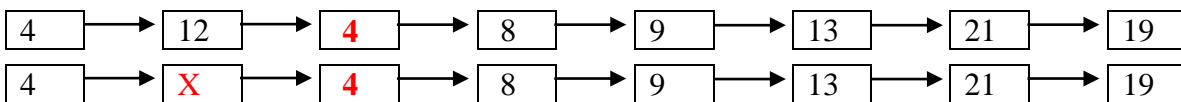
ii) Worst Fit -5%:



iii) Best Fit-:



iv) Best Fit Plus 3- meaning best fit but each process gets exactly (size +3) hole:



**Question 4 (20%, 5% each)** Consider the following process arrival, CPU burst (in milli-seconds) and explicit **priorities** of the processes A, B, C and D. Assume that **5** represents highest (preferred) priority and 1 lowest.

Draw the Gantt charts for and compute the turnaround and wait times and fill the table entries.  
 F: Finish Time, TA: TurnAround Time, W: Wait Tim

Process	Arrival time	CPU burst time	Priority	Priority/P			FCFS			SJF			SRTF		
				F	TA	W	F	TA	W	F	TA	W	F	TA	W
A	3	10	1	34	31	21	34	31	21	20	17	7	20	17	7
B	14	10	1	44	30	20	44	30	20	30	16	6	30	16	6
C	0	10	5	10	10	0	10	10	0	10	10	0	10	10	0
D	2	15	5	24	22	7	24	22	7	44	42	27	44	42	27
<b>Avg</b>				X	23.25	12	X	23.25	12	X	21.25	10	X	21.25	10

(a) Priority/preemptive:  
0

Time	910	24	34	44
Process	CCCCC DDDDDDDDD AAAAAAAAAABBBBBBBBBBBB			

(b) FCFS (First Come First Served).  
0

Time	910	24	34	44
Process	CCCCC DDDDDDDDD AAAAAAAAAABBBBBBBBBBBB			

(c) SJF (Shortest Job First).  
0

Time	910	20	30	44
Process	CCCCC AAAAAAAAAABBBBBBBBBB DDDDDDDDD CCCCC BBBBBBBBBB AAAAAAAAAA DDDDDDDDD			

(d) SRTF (Shortest Remaining Time First).  
0

Time	910	20	30	44
Process	CCCCC AAAAAAAAAABBBBBBBBBB DDDDDDDDD CCCCC BBBBBBBBBB AAAAAAAAAA DDDDDDDDD			

**Question 5 (18%)** The producer-consumer problem is a common example of cooperating processes. A **producer process** produces information that is consumed by a **consumer process**. Here, the producer process and the consumer process communicate using a **bounded buffer** implemented in shared memory.

Producer Process	Consumer Process
Memory region shared by both processes: <pre>#define BUFFER_SIZE 10  typedef struct {     . . . } item;  item buffer[BUFFER_SIZE]; int in = 0; int out = 0;</pre>	
<pre>item nextProduced; 1: 2: while (1) { 3:     /* produce an item in nextProduced 4:    */ 5:     while (((in + 1) % BUFFERSIZE) == 6:    out) 7:         ; /* do nothing */ 8:     buffer[in] = nextProduced; 9:     in = (in + 1) % BUFFER_SIZE;     }</pre>	<pre>item nextConsumed; 1: 2: while (1) { 3:     while (in == out) 4:         ; /* do nothing */ 5:     nextConsumed = buffer[out]; 6:     out = (out + 1) % BUFFER_SIZE; 7:     /* consume the item in 8:    nextConsumed */ 9: }</pre>

a. (5%) Assume that the **Consumer Process** happens to be the first to run. Assume that the Consumer Process is allowed to run for a long time. **Select what happens. Explain your answer.**

- 1- **The Consumer will be busy waiting**      2- **The buffer is full which produces an exception (fault).**
  - 3- **Buffer will be filled due to the long time**      4- **Control will be passed immediately to Producer process.**
- in=out=0 and nothing can happen except busy waiting (2% for explanation)

b. (5%) Assume that the Consumer Process eventually is swapped out (or the very long time quantum is finished), and the **Producer Process** gets its chance to run. Assume that the Producer Process is allowed to run for a long time, (enough time to fill the buffer). **Select what happens. Explain your answer.**

- 1- **The producer process will be busy waiting**      2- **Control will be passed immediately to Producer process.**
- 3- **Buffer will be filled due to the long time then process goes to busy waiting.**
- 4- **The buffer will overflow and an exception (interrupt) will be generated.**

After the buffer is full. (in+1) % BUFFERSIZE=out and nothing can happen except busy waiting (2% for explanation)

- c. (4%) For this part of the problem, assume we have re-started both processes, so they are just ready to start, with the shared memory variable having their values as initialized in the code. Describe one very fortunate (**optimistic**) sequence of executions which allows the processes to keep doing useful work. Your answer might take the form: **Producer** process runs until **\_once\_** then **Consumer** process runs **once** until **\_OR\_** **Producer** process runs until **\_buffer is full\_** then **Consumer** process runs until **Buffer is empty** (or any alternating arrangement: add 2 remove 2, add 3 remove 3 and so on).
- d. (4%) Is this program in need of improvement? If so, Suggest at least one way to improve performance. **YES, E.g. remove busy waiting for example by sleep awake,**

