Department of Computer Science & Engineering, IIT Guwahati

CS 343 - Operating Systems: Mid Semester Exam (21.09.2022)

Maximum Marks = 35, Duration 120 minutes.

Instructions: There are 8 questions for a total of 35 marks. Index the facing sheet of answer booklet with question number and page numbers. No clarifications on the questions will be entertained. Write necessary steps used for solving. No marks for answers without proper justifications.

1. [2 marks] Match the elements from Set A to a unique element in Set B using one to one mapping.

Set A	Set B
CPU scheduler	Fragmentation
Working Set	Deadlock Avoidance
Job Scheduler	Semaphore
Claim Edge	Degree of multiprogramming
Compaction	Deadlock Detection
Safe State	Thrashing
	Dispatcher

	Set A	**Set B**
1	CPU Scheduler	Dispatcher
2	Working Set	Thrashing
3	Job Scheduler	Degree of multiprogramming
4	Claim Edge	Deadlock Detection
5	Compaction	Fragmentation
6	Safe state	Deadlock Avoidance

1 mark if minimum 3 is correct, 2 marks if all 6 are correct. No further partial marks.
0 marks if the mapping is not one to one.

2. [2 marks] The TLB of a system that uses 2 KB pages has a reach of 1 MB. The system has a virtual address space of 1 TB and physical address space of 1 GB. What will be the size of TLB (in bytes) if each TLB entry stores a 6-bit process id, virtual page number, physical frame number and 10 control bits?

Solution:
Virtual Address space = 1 TB = 2^{40} bytes: Physical Address space= 1 GB = 2^{30} bytes
Page size = 2 KB = 2^{11} bytes
Total Virtual Pages = 2^{29} VA = 29-bit page number + 11-bit page offset
Total Physical Frames = 2^{19} PA= 19-bit frame number + 11-bit page offset
Since, TLB Reach = (TLB Size) x (Page Size) → 2^{20} Bytes = (TLB Size) x 2^{11} Bytes
TLB size = 2^{9} = 512 entries
So, number of rows in TLB = 2^{9} OR 512 [1 mark]
Therefore, Number of bits in each TLB entry
= 6 (process id) +29 (Virtual Page no.) +19 (Physical Frame no.) + 10 (control bits)
= 64 bits = 8 Bytes
size of TLB = No of rows x size of row = 512*8 Bytes = **2^{12} bytes** (4096 bytes) [1 mark]

3. [5 marks] Consider three processes P1, P2, and P3, that will print a single instance of '#', '&' and '$', respectively. Context switching can happen between these processes in arbitrarily any order and any time during the execution. Instructions inside the processes are atomic. These three processes are designed in such way that they synchronize their execution for generating the pattern $ $ $ # # & $ $ $ # # & $ $ $ # # & …. as output. You are permitted to use exactly 3 counting semaphores, (A, B and C), one binary semaphore (M) and a shared integer variable (X). It is given that the initial value of X, A, C, M, and B is 0, 0, 0, 1, and 3, respectively. Write the structure of P1, P2 and P3.

Solution:

[4 mark for correctness with proper semaphore usage, 1 mark for efficiency]

Initial Values: X=0, A=0, C=0, M=1 and B=3. Many possible solutions

Semaphores A and C can be interchanged.

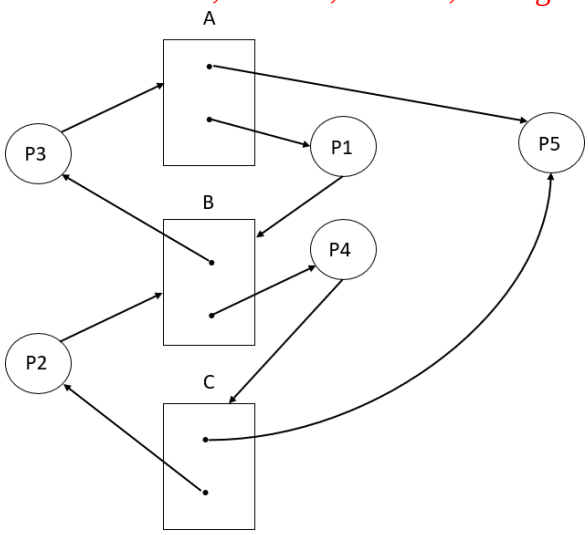
```
P1: while(1) {
        Wait(A);
        Wait(M);
        Print(#);
        X++;
        if(X==2){
              Signal(C);
              X=0;

        }
        Signal(M);
}

P2: while(1) {
        Wait(C);
        Wait(M);
        Print(&);
        X++;
        if(X==1){
              Signal(B);
              Signal(B);
              Signal(B);
              X=0;
        }
        Signal(M);
}

P3: while(1) {
        Wait(B);
        Wait(M);
        Print($);
        X++;
        if(X==3){
              Signal(A);
              Signal(A);
              X=0;
        }
        Signal(M);
}
```

4. [4 marks] Consider a system with 5 processes: P1, P2, P3, P4, & P5 and 3 resource types A, B, & C, each with 2 instances. The following single instance resource requests {Process, Resource type} arrive in sequence.

(i) {P4, B} (ii){P5, C} (iii){P2, C} (iv){P3, B} (v){P2, B}
(vi) {P5, A} (vii){P1, A} (viii){P1, B} (ix){P3, A} (x){P4, C}

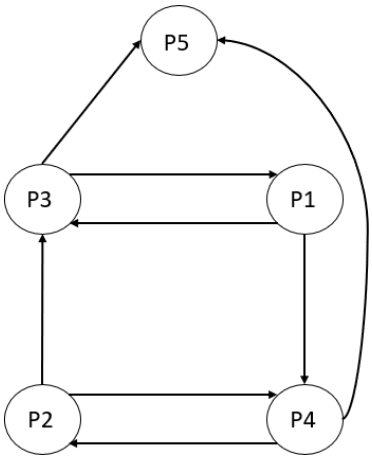
Draw the Resource Allocation Graph and Wait for Graph of the above system. The edges of the graph should not crossover each other.

a) Required Resource allocation graph is drawn below:
[2 marks for correct, labelled, directed, no edge crossover graph. No partial marking]



- Arrows from instances of a resource to process represent the allocation of resource to a process that have been made.
- Arrows from processes to resources represent the allocation request for that resource which the process in context has made.

b) Required Wait For Graph is drawn below :
[2 marks for correct, labelled, directed, no edge crossover graph. No partial marking]



- Arrows from a Process to other Process represent that the former process is waiting for an instance of a resource to be released by the later process.

5. [4 marks] Consider a system with 4 active processes P, Q, R, and S and 3 resource types A, B, and C. There are a total of 6 instances each for A and B and 3 instances for C that are available initially. Consider the following MAX and ALLOCATION matrices that are fed to Banker's algorithm module for deadlock avoidance. Three independent resource requests: R1: P[1 0 1], R2: Q[1 0 1], and R3: R [1 0 1] come to the deadlock management at the same time. Which of these requests can be granted now? Explain with necessary justifications and illustrations.

	MAX				ALLOCATION		
	A	B	C		A	B	C
P	3	2	2		2	0	1
Q	3	4	0		1	2	0
R	4	2	1		1	1	0
S	2	3	2		1	1	1

Solution: Given Total Resources of A B C = [6 6 3]

From the table allocation we have:

	MAX	ALLOCATION	NEED
	A B C	A B C	A B C
P	3 2 2	2 0 1	1 2 1
Q	3 4 0	1 2 0	2 2 0
R	4 2 1	1 1 0	3 1 1
S	2 3 2	1 1 1	1 2 1

Total Allocation is [5 4 2] and hence Available = is [6 6 3] – [5 4 2] = [1 2 1] [1 Mark]

The requests are as follows, R1: P[1,0,1] R2: Q[1,0,1] R3: R[1,0,1]

Case 1: R1: P[1 0 1]

Let us assume we grant R1 now, Available will be [0 2 0] and revised need of all process can be satisfied by following P first. Safe sequence is P followed by Q/R/S any order. Safe.

Hence R1: P[1 0 1] can be granted as safe sequence exists even after granting R1. [1 Mark]

Case 2: R2: Q [1 0 1]

Request of Q [1 0 1] is > Need of Q [2 2 0]. Hence, as per Bankers algorithm it is error. Request cannot be granted. No calculation of safe state. [1 Mark]

Case 3: R2: R [1 0 1]

Let us assume we grant R3 now, Available will be [0 2 0]. The revised need of all process is larger than Available. So none of the process can complete its resource requirements. Hence allocation of R3: R [1 0 1] cannot be granted as granting this will lead the system to an unsafe state. [1 Mark]

Conclusion:

R1 can be granted, R2 gives error condition and R3 cannot be granted.

6. [4 marks] A page table of a computer system that resides in main memory has a page fault rate of 10%. The system uses a TLB that has an access time of 50 ns and miss rate of 5%. Each main memory access takes 300 ns and each page transfer to or from the disk takes 2500 ns each. During page replacement, dirty victim frames have to be written back to disk before the required page is read in from the disk. Assume that page table update, subsequent TLB update and referring to updated TLB / page table entry take negligible time. If average memory access time is 380 ns, what fraction of victim frames are clean?

Solution:
Page Fault, pf = 10 % = 0.1
TLB Access Time , T = 50 ns
TLB Miss Rate mr = 5% = 0.05. Hence, hr=(1-mr)= 0.95
MM Access Time, M = 300 ns
Disk Access Time, D = 2500 ns
AMAT = 380 ns and dirty rate, dr = ? clean rate cr = 1- dr

AMAT= hr.(T+M) + mr.{(1-pf)x(T+M+M) + pf.[(1-dr).(T+M+D+M)+dr.(T+M+D+D+M)]}

OR AMAT = (T+M) + mr.{ M + pf.[(1-dr) x D + dr x 2D)]} …. (2) [2 marks]

380 = 0.95x350 + 0.05.{(0.9)x(650) + 0.1.[(1-dr)x(3150)+dr x(5650)]}

Solving we get, dr = 0.2 (20% frames are dirty)
Fraction of clean frames = (1-dr) = 0.8 or 80% [2 marks]

Remarks:
Every access will have one TLB access and one Memory access (actual data access)
Every TLB miss will access page table in memory for sure.
Why 2 M in case of no page faults? You need to look up the page table (M) for the entry and then access the required location in memory (M).
Why 2 Disk Access in case of dirty page? You'll need to write a dirty page back to disk (D) and bring the page (which caused the page fault) back to main memory (D).

7. [6 marks] A system with 1 MB virtual address space has 256 KB physical memory that uses a paged memory scheme with a page size of 16 KB. The virtual to physical address translation is done with a hashed page table that is stored in the physical memory. The virtual page number is hashed using the function I=V%(F/4), where I is the index in the hashed page table, V is the virtual page number and F is the total number of page frames. Initially the hashed page table is kept in the zeroth frame in the physical memory. All the other frames are empty. During a page fault, frame is allocated from lower order memory address to higher. Average processing time of page fault (swap in and swap out of pages + updating page table and TLB) is 450 ms, average access time (read/write) for a main memory data unit is 50 ms and hash function index computation takes 10 ms. The hashed page table entry and data in an element of a chain are considered as single data unit for reading/writing. After every page fault, once the page table is updated, it takes one chain element reading to get the frame address. For executing a program, the CPU generated 10 virtual addresses in the following sequence.
Virtual Address sequence: 0x73400, 0x25200, 0xE5304, 0x70240, 0x7C008, 0x47A88, 0x84840, 0x44100, 0xA1328, 0x867D0

(a) Draw the hashed page table and its associated chains after execution of the above 10 sequence of address.
(b) Calculate the total memory access time taken in executing the above program.

Virtual Memory: 1 MB = 2^{20} bytes VA: 6 bit page number + 14 bit page offset
Physical Memory: 256 KB 2^{18} bytes PA: 4 bit page number + 14 bit page offset
Page Size : 16 KB = 2^{14} bytes (last 14 bits in address is page offset) [1 Mark]

Hash Function: I = V%(F/4).
I : Index in Hash Table, V: Virtual Page Number, F: Total Number of Physical Frames = 16

[1 mark for computation of virtual page number correctly]

Virtual Address	Virtual Page No Msb 6 bits (in decimal)	Index in Hash Table (I)	Physical Frame No (in Decimal)
V1= 0x73|400	011100 = 28	28%4 = 0	1 (Page Fault)
V2=0x25|200	001001 = 9	9%4 = 1	2 (Page Fault)
V3=0xE5|304	111001 = 57	57%4 = 1	3 (Page Fault)
V4=0x70|240	011100 = 28	28%4 = 0	1
V5=0x7C|008	011111 = 31	31%4 = 3	4 (Page Fault)
V6=0x47|A88	010001 = 17	17%4 = 1	5 (Page Fault)
V7=0x84|840	100001 = 33	33%4 = 1	6 (Page Fault)
V8=0x44|100	010001 = 17	17%4 = 1	5
V9=0xA1|328	101000 = 40	40%4 = 0	7 (Page Fault)
V10=0x86|7D0	100001 = 33	33%4 = 1	6

(a) Final Hash-Page Table (Page Number and frame number in decimal) [2 marks]

0→	(28,1) → (40,7)
1→	(9,2) → (57,3) → (17,5) →(33,6)
2	---
3→	(31,4)

(b) Individual memory access time.

Virtual Address	Access Time Components	Total Access Time
V1= 0x73|400 Page 28	10+50+450+50+50	610
V2=0x25|200 Page 9	10+50+450+50+50	610
V3=0xE5|304 Page 57	10+50+50+450+50+50	660
V4=0x70|240 Page 28 *	10+50+50+50	160
V5=0x7C|008 Page 31	10+50+450+50+50	610
V6=0x47|A88 Page 17	10+50+50+50+450+50+50	710
V7=0x84|840 Page 33	10+50+50+50+50+450+50+50	760
V8=0x44|100 Page 17 *	10+50+50+50+50+50	260
V9=0xA1|328 Page 40	10+50+50+450+50+50	660
V10=0x86|7D0 Page 33 *	10+50+50+50+50+50+50	310

Time Taken for Main Memory Access = 5350 ms [2 marks]

Special case: If the access to chain element after page fault and page table update is not counted, answer is 5350-(7x50) = 5000 ms [1 mark]

8. [8 marks] Consider the memory allocation and deallocation for a 1 MB user space hole which is initially empty. The following are the sequence of events with respect to arrival and termination of processes. Perform memory allocation and deallocation using the following two independent cases and answer the questions given at the end for each of the case.
Case 1: Buddy system allocator.
Case 2: Best fit allocator that allocates at granularity of 4 KB.
T1: P1 with 280 KB arrives.
T2: P2 with 10 KB arrives.
T3: P3 with 198 KB arrives.
T4: P4 with 25 KB arrives.
T5: P2 terminates.
T6: P5 with 100 KB arrives.
T7: P1 terminates.
T8: P6 with 177 KB arrives.
T9: P4 terminates.
T10: P7 with 305 KB arrives.
(a) How much internal and external fragmentation is there after T5?
(b) How much internal and external fragmentation is there after T10?
(c) Draw the process allocation and hole layout of the total memory after T5 and T10.

Case1: Buddy system allocation.

T1	P1 (280)		512					
T2	P1 (280)		P2(10)	16	32	64	128	256
T3	P1 (280)		P2(10)	16	32	64	128	P3 (198)
T4	P1 (280)		P2(10)	16	P4 (25)	64	128	P3 (198)
T5	P1 (280)		32		P4 (25)	64	128	P3 (198)
T6	P1 (280)		32		P4 (25)	64	P5 (100)	P3 (198)
T7	512		32		P4 (25)	64	P5 (100)	P3 (198)
T8	P6 (177)	256	32		P4 (25)	64	P5 (100)	P3 (198)
T9	P6 (177)	256	128				P5 (100)	P3 (198)
T10*	P6 (177)	256	128				P5 (100)	P3 (198)

(a) How much internal and external fragmentation is there after T5? [1 mark]
Internal Fragmentation = P1:232 KB + P3:58 KB + P4:7 KB= 297 KB
External Fragmentation = all process allocated, 0 KB

(b) How much internal and external fragmentation is there after T10? [1 mark]
Internal Fragmentation = P3:58 KB + P5:28 KB + P6:79 KB= 165 KB
External Fragmentation = P7 is unallocated, add all the holes= 256 + 128 = 384 KB

(c) Draw the process allocation and hole layout of the total memory after T5 and T10?
Figure given above or its equivalent. [2 Marks]

Case 2: Best fit allocator that allocates at granularity of 4 KB.

T1	P1 (280)	744					
T	P1 (280)	P2 (12)	732				
T3	P1 (280)	P2 (12)	P3 (200)	532			
T4	P1 (280)	P2 (12)	P3 (200)	P4 (28)	504		
T5	P1 (280)	12	P3 (200)	P4 (28)	504		
T6	P1 (280)	12	P3 (200)	P4 (28)	P5 (100)	404	
T7	292		P3 (200)	P4 (28)	P5 (100)	404	
T8	P6 (180)	112	P3 (200)	P4 (28)	P5 (100)	404	
T9	P6 (180)	112	P3 (200)	28	P5 (100)	404	
T10	P6 (180)	112	P3 (200)	28	P5 (100)	P7 (308)	96

(a) How much internal and external fragmentation is there after T5? [1 mark]
Internal Fragmentation = P3:2 KB + P4:3 KB = 5 KB
External Fragmentation = all process allocated, 0 KB

(b) How much internal and external fragmentation is there after T10? [1 mark]
Internal Fragmentation = P3:2 KB + P6:3 KB + P7:3 KB = 8 KB
External Fragmentation = all process allocated, 0 KB

(c) Draw the process allocation and hole layout of the total memory after T5 and T10?
Figure given above or its equivalent. [2 Marks]

****