CSC 718 -- Operating System Design
Exam 2

Answer all the questions. The maximum credit for each question is as shown.

1. (15) Give a solution to the Producer-Consumer problem using the Wait(s) and Signal(s) instructions. The producer and Consumer will be sharing a common buffer BUF[0..N-1]. The Producer will be putting items in the array and the Consumer will be taking items from the array. We need to synchronize the two processes so that the Producer will not be overwriting a full buffer and the Consumer will not be taking items from an empty buffer.

   Answer:

   - Global Variable
     1. B[0..N-1] – an array of size N (Buffer)
     2. P – a semaphore, initialized to N
     3. C – a semaphore, initialized to 0
   - Local Variable
     1. In – a ptr(integer) used by the producer, in=0 initially
     2. Out – a ptr(integer) used by the consumer, out=0 initially

   ```
   Producer Process
   producer: produce(w)
             wait(p)
             B[in]=w
             in=(in+1)mod N
             signal(c)
             goto producer

   Consumer Process
   consumer: wait(c)
              w=B[out]
              out=(out+1)mod N
              signal(p)
              consume(w)
              goto consumer
   ```

2. (20) Define the Test-and-Set instruction and show how it can be used to solve the Mutual Exclusion problem. The solution to the Mutual Exclusion problem suffers from the fact that a process can be starved. Show how the starvation problem can be fixed.

   Answer:

   (a) Definition of Test-and-Set:

Boolean TS(i)= true if i=0; it will also set i to 1
false if i=1

(b) The solution to the Mutual Exclusion:

Global Variable – lock, Initially, lock=0

P_i
$Prefix_i$
While(¬ TS(lock)) do {}
CS_i
Lock=0
$suffix_i$

(c) Fix the starvation problem:

To avoid the starvation problem, we can do the following:
Global variables: waiting[0..n-1] (boolean)
lock (integer)
Local variable: Key_i (boolean)
Initially, lock=0 and waiting[i]=false for all $0 \le i \le n-1$
P_i
Waiting[i]=true;
Key_i=true;
While(waiting[i] and Key_i) do {Key_i=¬ TS(lock)}
Waiting[i]=false;
CS_i
j=(i+1) mod n
While(j≠i and ¬ waiting[j]) do {j=(j+1) mod n}
if (j=i) then lock=0 else Waiting[j]=false

3. (15) Consider the Deadlock Avoidance problem. Shown below are the Claim Matrix, Allocation Matrix at time t, and the Available Vector at time t. Suppose Process P2 asks for (2, 1, 2). Will the Operating System grant this request now?

Available Vector V = (4, 2, 3)

$$Claim\ Matrix = \begin{pmatrix} 3 & 4 & 3 \\ 6 & 4 & 5 \\ 7 & 3 & 4 \\ 8 & 2 & 3 \end{pmatrix} \begin{matrix} P1 \\ P2 \\ P3 \\ P4 \end{matrix}$$

$$Allocation\ Matrix = \begin{pmatrix} 3 & 3 & 2 \\ 1 & 2 & 1 \\ 1 & 0 & 1 \\ 2 & 2 & 1 \end{pmatrix} \begin{matrix} P1 \\ P2 \\ P3 \\ P4 \end{matrix}$$

Answer:

$$Claim\ Matrix = \begin{pmatrix} 3 & 4 & 3 \\ 6 & 4 & 5 \\ 7 & 3 & 4 \\ 8 & 2 & 3 \end{pmatrix} \begin{matrix} P1 \\ P2 \\ P3 \\ P4 \end{matrix}$$

Pretend to give (2,1,2) to P2, then

$$Allocation\ Matrix = \begin{pmatrix} 3 & 3 & 2 \\ 3 & 3 & 3 \\ 1 & 0 & 1 \\ 2 & 2 & 1 \end{pmatrix} \begin{matrix} P1 \\ P2 \\ P3 \\ P4 \end{matrix}$$

Available Vector = (4,2,3) - (2,1,2) = (2,1,1)

So maximum request that can be made becomes

$$R = \begin{pmatrix} 3 & 4 & 3 \\ 6 & 4 & 5 \\ 7 & 3 & 4 \\ 8 & 2 & 3 \end{pmatrix} - \begin{pmatrix} 3 & 3 & 2 \\ 3 & 3 & 3 \\ 1 & 0 & 1 \\ 2 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 3 & 1 & 2 \\ 6 & 3 & 3 \\ 6 & 0 & 2 \end{pmatrix}$$

We can sequence P1 first
V=(2,1,1)+(3,3,2)=(5,4,3)
We can sequence P2 next
V=(5,4,3)+(3,3,3)=(8,7,6)
Then we can sequence the rest easily

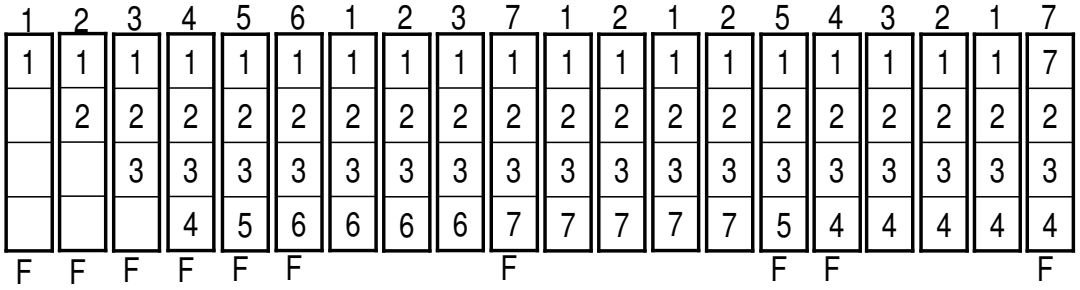
So it is safe ans OS will grant to P2.

4. (15) Shown below is a page reference stream. Show the contents of the main memory if 4 page frames are allocated to this process and the page replacement algorithm used are (a) Belady's optimal algorithm, (b) Least Recently Used algorithm (LRU), (c) First-in-first-out (FIFO).

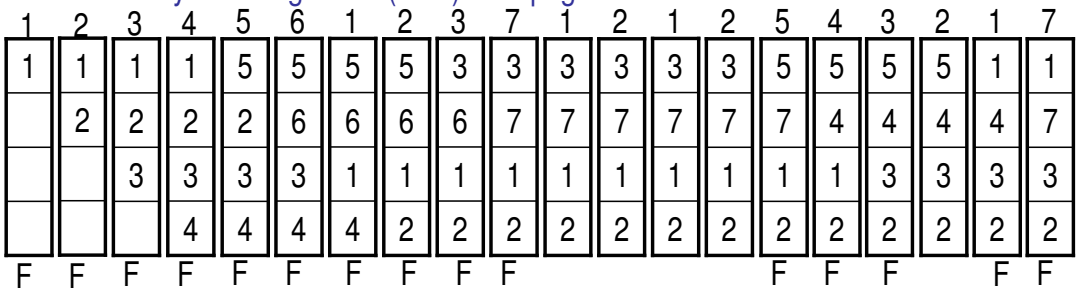
   1 2 3 4 5 6 1 2 3 7 1 2 1 2 5 4 3 2 1 7

Answer:

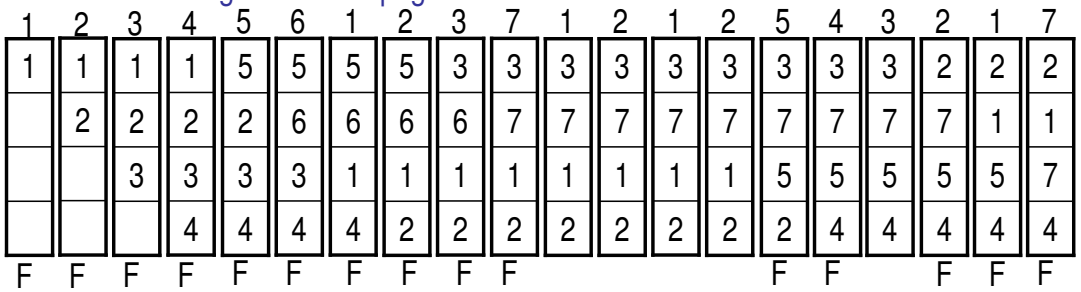
Belady's optimal algorithm: # of page faults=10



Least Recently Used algorithm (LRU): # of page faults=15



First-in-first-out algorithm: # of page faults=15



5. (15) Explain paging system. What is the dynamic address translation scheme used in paging system? How do we implement virtual memory in paging system?

Answer:

Partition memory into small equal-size chunks and divide each process into the same size chunks; the chunks of a process are called pages and chunks of memory are called frames. When a process is to be executed, its pages are loaded into any available memory frames from the backing store.

Operating system maintains a page table for each process. This page table contains the frame location for each page in the process. In paging system, memory address consists of a page number and offset within the page. The page number is used as an index into a page table. The page table contains the base address of each page in physical memory. This base address is combined with the page offset to define the physical memory address that is sent to the

memory unit. The dynamic address translation scheme used in paging system is shown as follows:

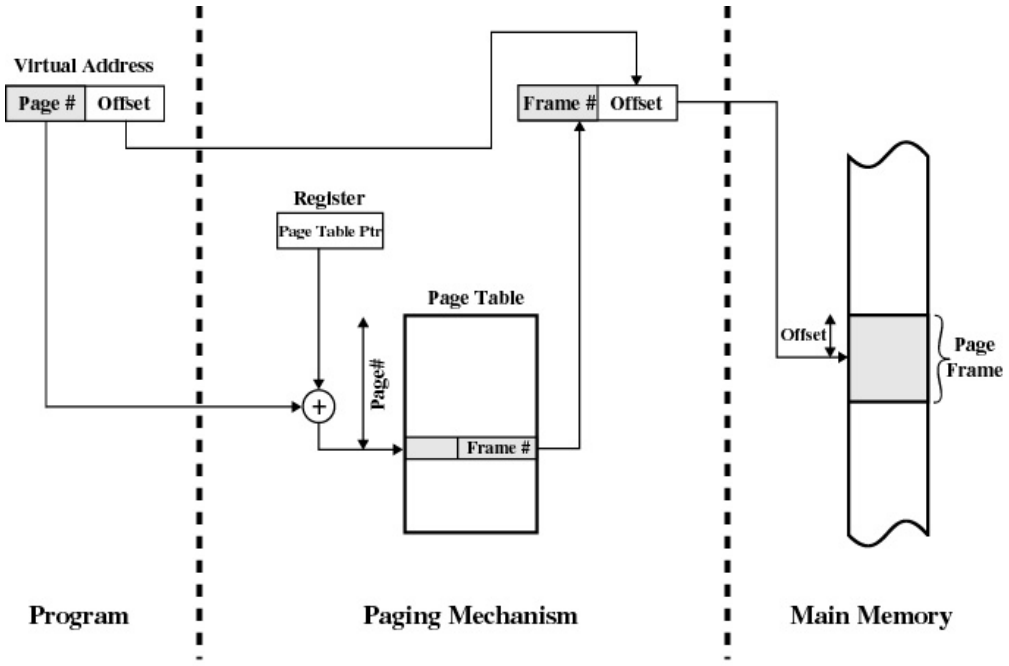


Figure 8.3 Address Translation in a Paging System

Implementation of virtual memory in paging system:
Virtual memory is a technique that allows the execution of processes that are not completely in memory. In paging system, virtual memory technique means that not all pages of a process are in main memory. In order to support virtual memory, OS needs to decide on the following issues:

- Fetch Policy:
  Fetch Polich determines when a page should be brought into memory. There are two methods to be used:
  Demand paging – bring pages into main memory only when it is needed
  Prepaging – brings in more pages then needed even though it is not needed now.
- Placement Policy:
  Placement Policy decides where a process piece resides in main memory. It is a trivial issue. We just find the free frames and put the pages in.
- Replacement Policy:
  Replacement Policy determines which page to replace when a new page needs to be brought in and there is no empty page frame around. Page removed should be the page least likely to be referenced in the near future. Most policies predict the future behavior on the basis of past behavior. There are four algorithms: Belady's Optimal Algorithm, Least Recently Used Algorithm (LRU), First-in-first-out Algorithm (FIFO), Clock (approximation of LRU)

6. (20) Construct the schedule for the following set of jobs if multi-level feedback queue is used. Assume we use 3 levels in the multi-level feedback queue and that the time quantum is 1.5 time units.

Jobs	Arrival Time	Service Time
A	0	4
B	1	2
C	2	5
D	4	3
E	5	4
F	6	5

Answer: