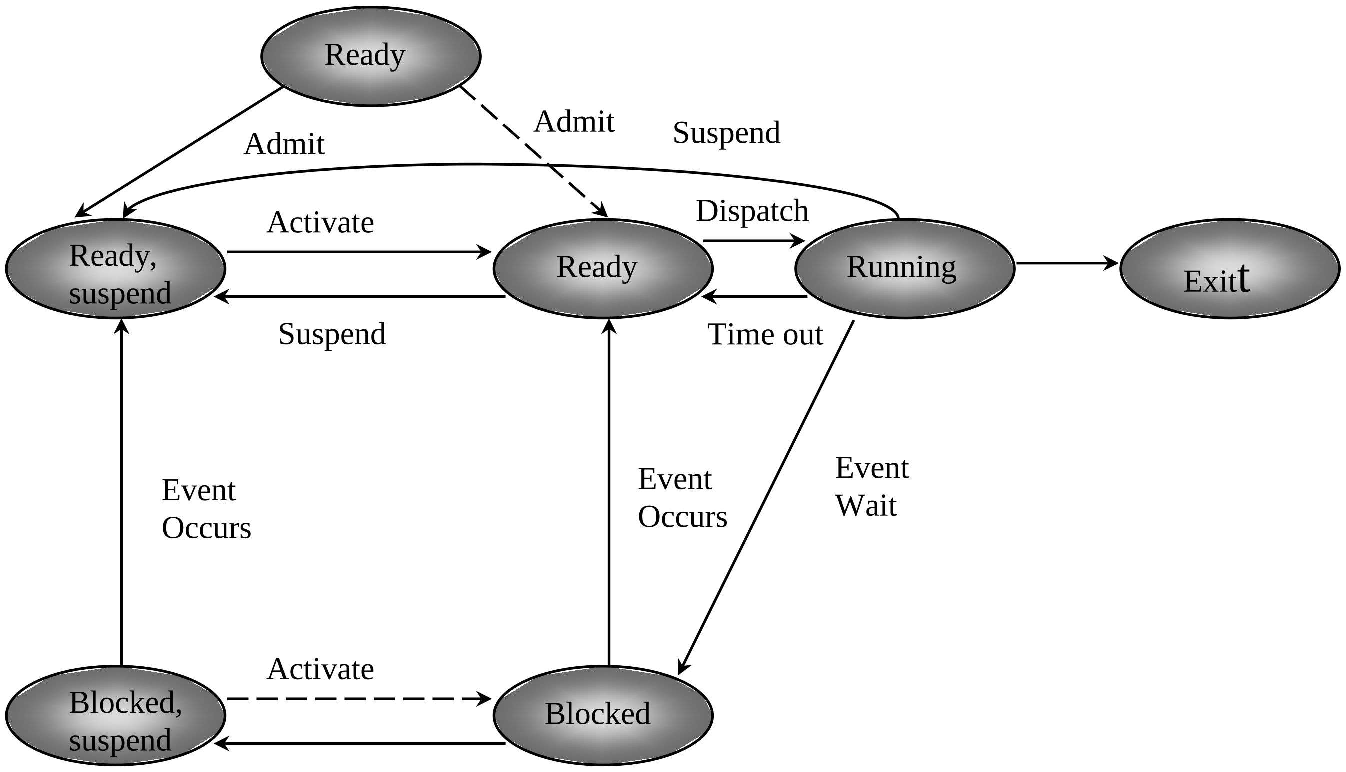
SYSC 4001 Operating Systems
Solution for Sample Final Exam 1
Instructor: T. Kunz

Question 1. Processes and Threads (10 marks)

a) Draw the process state model with 5 basic states plus support for swapping as discussed in class. Clearly label all relevant state transitions in the diagram.
Answer (3 marks):



b) Briefly describe the difference between a process and a thread.
Answer (3 marks):

- Process:
  - Has a virtual address space which holds the process image
  - Protected access to processors, other processes, files, and I/O resources
- Threads:
  - Has an execution state (running, ready, etc.)
  - Saves thread context when not running
  - Has an execution stack
  - Has some per-thread static storage for local variables
  - Has access to the memory and resources of its process, all threads of a process share this

c) What are the advantages of threads, compared to processes? What are the disadvantages?

Answer (2 marks):
Advantages:
- Takes less time to create a new thread than a process
- Less time to terminate a thread than a process
- Less time to switch between two threads within the same process
- Since threads within the same process share memory and files, they can communicate with each other without invoking the kernel

Disadvantage:
- Less protection from each other

d) What is the difference between kernel-level threads and user-level threads?
Answer (2 marks):
User-level threads:
- All thread management is done by the application
- The kernel is not aware of the existence of threads
- Thread switching does not require kernel mode privileges
- Scheduling is application specific

Kernel-level threads:
- Kernel maintains context information for the process and the threads
- Switching between threads requires the kernel

Question 2. Synchronization (10 marks)

a) What are the requirements any mutual exclusion solution should fulfill?
Answer (3 marks):
- Only one process at a time is allowed in the critical section for a resource
- If a process halts in its critical section, it must not interfere with other processed
- A process requiring the critical section must not be delayed indefinitely; no deadlock or starvation
- A process must not be delayed access to a critical section when there is no other process using it
- No assumptions are made about relative process speeds or number of processes
- A process remains inside its critical section for a finite time only

b) What are monitors? Briefly explain the concept and sketch a solution to the producer-consumer problem using monitors.
Answer (4 marks):
- **Is a software module containing:**
  - one or more procedures
  - an initialization sequence
  - local data variables
- **Characteristics:**
  - local variables accessible only by monitor’s procedures
  - a process enters the monitor by invoking one of it’s procedures
  - only one process can be in the monitor at any one time

Two processes and a monitor to manage the shared buffer:

```
Producer:
repeat
  produce v;
  append(v);
forever

Consumer:
repeat
  take(v);
  consume v;
forever
```

```
Monitor boundedbuffer:
  buffer: array[0..k-1] of items;
  nextin, nextout, count: integer;
  notfull, notempty: condition;

  append(v):
    while (count=k) cwait(notfull);
    buffer[nextin]:= v;
    nextin:= nextin+1 mod k;
    count++;
    csignal(notempty);

  take(v):
    while (count=0) cwait(notempty);
    v:= buffer[nextout];
    nextout:= nextout+1 mod k;
    count--;
          csignal(notfull);
```

c) What are the necessary and sufficient conditions for deadlock to occur?

Answer (3 marks):

- Mutual exclusion
  - only one process may use a resource at a time
- Hold-and-wait
  - a process may hold allocated resources while awaiting assignment of others
- No preemption
  - no resource can be forcibly removed from a process holding it
- Circular wait
  - a closed chain of processes exists, such that each process holds at least one resource needed by the next process in the chain
  - consequence of the first three conditions
  - first three conditions are necessary but not sufficient for deadlock

Question 3. Memory Management (20 marks)

a) Briefly explain how memory management with fixed partitions works? Discuss the case of both equal and unequal partitions.

   Answer (2 marks):

   Fixed partitions with equal size:

   - Partition available memory into equal-sized regions with fixed boundaries
   - Equal-size partitions
     - any process whose size is less than or equal to the partition size can be loaded into an available partition
     - if all partitions are full, the operating system can swap a process out of a partition
     - a program may not fit in a partition. The programmer must design the program with overlays

   Unequal partitions:

   - Partition available memory into different-size regions with fixed boundaries
   - Lessons problems with equal-size partitions (better chance of finding a partition that “fits” process better).

   Placement algorithms:

   - Equal-size partitions
     - because all partitions are of equal size, it does not matter which partition is used
   - Unequal-size partitions
     - can assign each process to the smallest partition within which it will fit
     - queue for each partition
     - processes are assigned in such a way as to minimize wasted memory within a partition

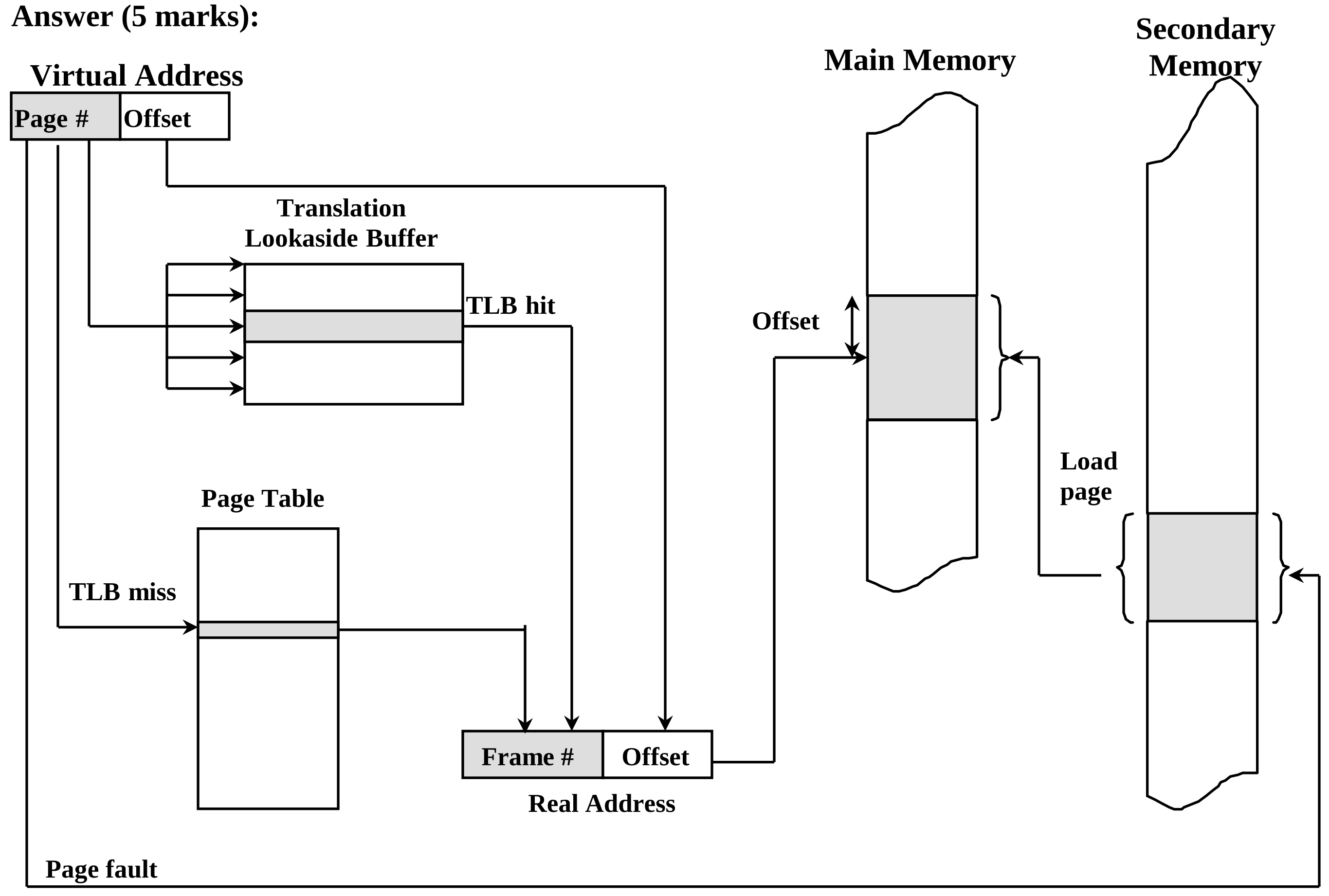
b) Explain and discuss the relative advantages of at least three dynamic partitioning placement algorithms.

   Answer (3 marks):

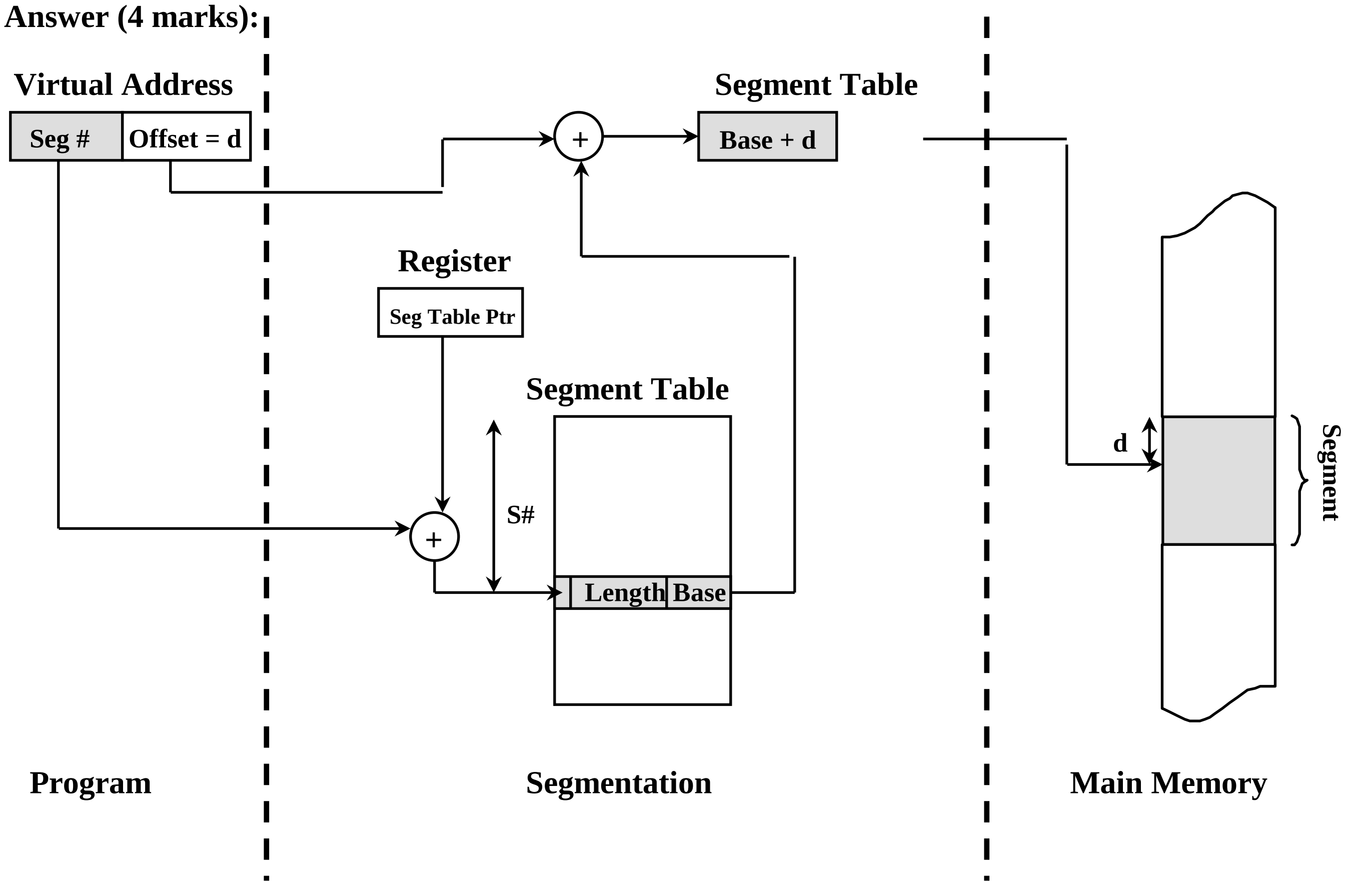
   - Best-fit algorithm
     - chooses the block that is closest in size to the request
     - worst performer overall
     - since smallest block is found for process, the smallest amount of fragmentation is left memory compaction must be done more often
   - First-fit algorithm
     - starts scanning memory from the beginning and chooses the first available block that is large enough.
     - fastest
     - may have many process loaded in the front end of memory that must be searched over when trying to find a free block
   - Next-fit
     - starts scanning memory from the location of the last placement and chooses the next available block that is large enough
     - more often allocate a block of memory at the end of memory where the largest block is found

- the largest block of memory is broken up into smaller blocks
- compaction is required to obtain a large block at the end of memory

c) Sketch the address translation process for a virtual memory system based on paging, using a Translation Lookaside Buffer.
Answer (5 marks):



d) Sketch the address translation process for a virtual memory system based on segmentation.
Answer (4 marks):



e) Explain the following page replacement algorithms and explain their relative advantages and disadvantages: LRU, FIFO, and Clock Policy.
Answer (6 marks):

- Least Recently Used (LRU)
  - Replaces the page that has not been referenced for the longest time
  - By the principle of locality, this should be the page least likely to be referenced in the near future
  - Each page could be tagged with the time of last reference. This would require a great deal of overhead.
- First-in, first-out (FIFO) treats page frames allocated to a process as a circular buffer
  - Pages are removed in round-robin style
  - Simplest replacement policy to implement
  - Page that has been in memory the longest is replaced
  - These pages may be needed again very soon
- Replacement Policy - Clock Policy
  - Additional bit called a use bit
  - When a page is first loaded in memory, the use bit is set to 0
  - When the page is referenced, the use bit is set to 1
  - When it is time to replace a page, the first frame encountered with the use bit set to 0 is replaced.
  - During the search for replacement, each use bit set to 1 is changed to 0

Question 4. Scheduling (10 marks)

a) List short-term scheduling criteria.
Answer (2 marks):

- User-oriented:
  - Response Time
  - Elapsed time between the submission of a request until there is output.
- System-oriented
  - effective and efficient utilization of the processor
- Performance-related
  - measurable such as response time and throughput
- Not performance related
  - predictability

b) Briefly describe and discuss the advantages and disadvantages of the following short-term scheduling policies: FCFS, round robin, shortest process next, shortest remaining time.
Answer (4 marks):

- First-Come-First-Served:
  - A short process may have to wait a very long time before it can execute
  - Favors CPU-bound processes
  - I/O processes have to wait until CPU-bound process completes
- Round Robin:
  - Uses preemption based on a clock
  - An amount of time is determined that allows each process to use the processor for that length of time
- Shortest Process Next
  - Nonpreemptive policy
  - Process with shortest expected processing time is selected next
  - Short process jumps ahead of longer processes
  - Predictability of longer processes is reduced
  - If estimated time for process not correct, the operating system may abort it
  - Possibility of starvation for longer processes
- Shortest Remaining Time:
  - Preemptive version of shortest process next policy
  - Must estimate processing time

c) Briefly explain the concept of “gang scheduling”.
Answer (2 marks):

- Simultaneous scheduling of threads that make up a single process
- Useful for applications where performance severely degrades when any part of the application is not running
- Threads often need to synchronize with each other

d) What four categories for real-time scheduling can be distinguished?
Answer (2 marks):

- Static table-driven
  - determines at run time when a task begins execution
- Static priority-driven preemptive
  - traditional priority-driven scheduler is used
- Dynamic planning-based
- Dynamic best effort

Question 5. Disks and File Systems (20 marks)

a) Briefly describe the following disk scheduling algorithms: FIFO, priority, last-in-first-out, shortest seek time first, SCAN, and C-SCAN. Which of these algorithms provide fairness (i.e., avoid starvation)?

Answer (8 marks):

- First-in, first-out (FIFO)
  - process request sequentially
  - fair to all processes
  - approaches random scheduling in performance if there are many processes
- Priority
  - goal is not to optimize disk use but to meet other objectives
  - short batch jobs may have higher priority
  - provide good interactive response time
- Last-in, first-out
  - good for transaction processing systems
  - the device is given to the most recent user so there should be little arm movement
  - possibility of starvation since a job may never regain the head of the line
- Shortest Service Time First
  - select the disk I/O request that requires the least movement of the disk arm from its current position
  - always choose the minimum Seek time
- SCAN
  - arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction
  - direction is reversed
- C-SCAN
  - restricts scanning to one direction only
  - when the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again

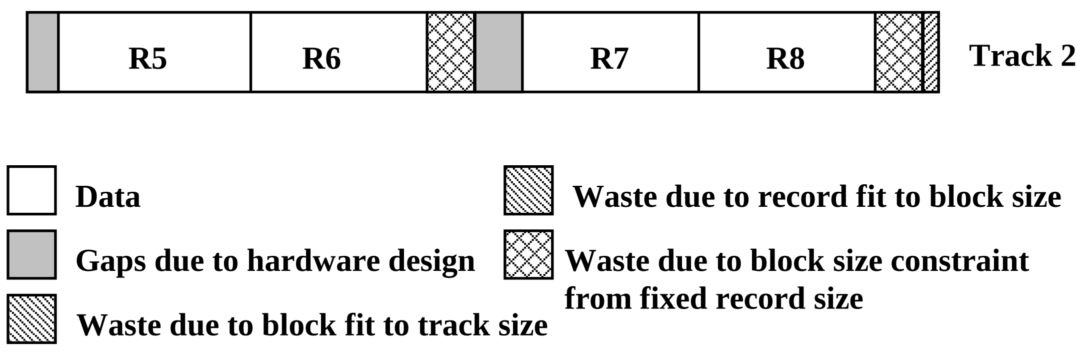
Other than FIFO, none of these policies guarantee fairness: there could always be requests with higher priority, more recent requests, or later requests that access the disk at the same track the head is currently positioned over (which will give these requests priority over other requests in the last three policies).

(2 marks for correct answer to last part…..)

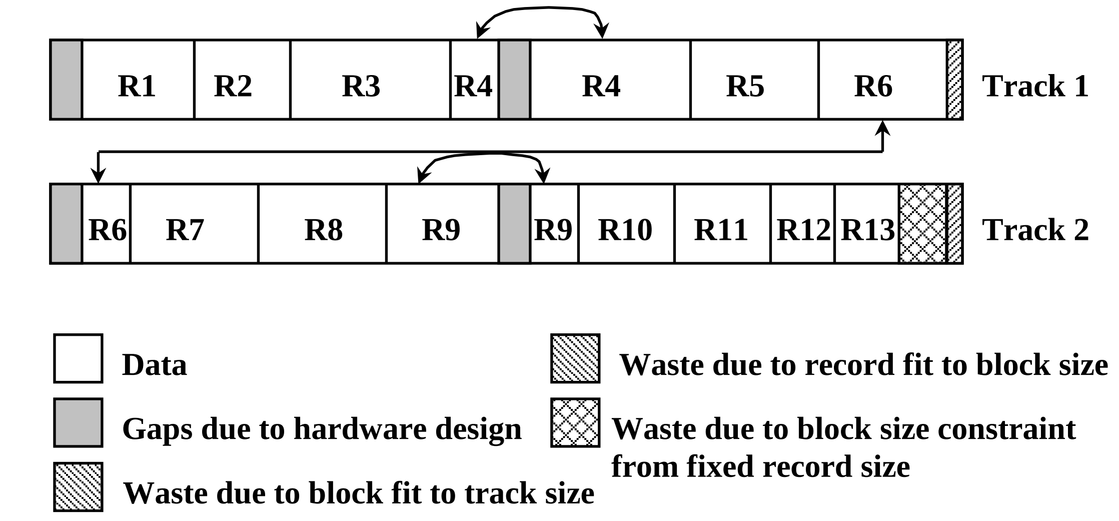
b) Briefly sketch the disk layout for a file with multiple records, using one of the following three blocking methods: fixed blocking, variable-length spanned blocking, and variable-length unspanned blocking. Your diagrams should indicate which parts of the blocks are filled with data and which parts or waste (and due to which reason does this waste occur).

Answer (8 marks):

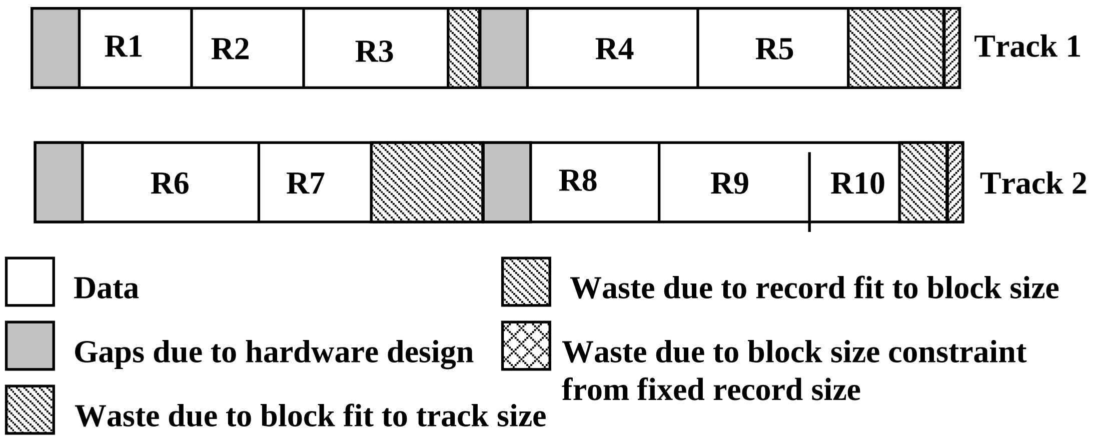
Fixed Blocking:


R5
R6
R7
R8
Track 2
Data
Gaps due to hardware design
Waste due to block fit to track size
Waste due to record fit to block size
Waste due to block size constraint from fixed record size

Variable-length Spanned Blocking:


R1
R2
R3
R4
R4
R5
R6
Track 1
R6
R7
R8
R9
R9
R10
R11
R12
R13
Track 2
Data
Gaps due to hardware design
Waste due to block fit to track size
Waste due to record fit to block size
Waste due to block size constraint from fixed record size

Variable-length Unspanned Blocking:


R1
R2
R3
R4
R5
Track 1
R6
R7
R8
R9
R10
Track 2
Data
Gaps due to hardware design
Waste due to block fit to track size
Waste due to record fit to block size
Waste due to block size constraint from fixed record size

c) Explain contiguous and chained file allocation and their relative advantage and disadvantage.

Answer (4 marks):

- Contiguous allocation
    - Single set of blocks is allocated to a file at the time of creation
    - Only a single entry in the file allocation table
    - starting block and length of the file
    - Fragmentation will occur
    - Will become difficult to find contiguous blocks of sufficient length
- Chained allocation
    - Allocation on basis of individual block
    - Each block contains a pointer to the next block in the chain
    - Only single entry in the file allocation table
    - Starting block and length of file
    - No fragmentation
    - Any free block can be added to the chain
    - No accommodation of the principle of locality