End of chapter questions

1 a) FCFS = 17.5 ms

b) SJF = 6.25 ms

c) SRTF = 6.0 ms

d) round robin = 12.75 ms

2 a)

- The page is present in **memory**.
- Loaded at/stored/present in page frame 542//its memory address is 542.

b) i)

- The next instruction is first instruction in Page 6.
- Page 6 is not present in memory.
- The instruction can only be executed if present in memory.
- The program cannot continue until Page 6 is loaded.

ii) When there is an attempt to load an instruction for a Page not in memory

- a page default occurs/Page 5 finishes …
- … this generates an interrupt
- ISR code is executed
- causes the OS to load Page 6 into memory.

c) i) time of entry

ii)

page	presence flag	page frame address	additional data
6	1	221	12:07:34:49

iii)

- When the procedure call is made, Page 1 is swapped out and Page 3 is swapped in.
- At the end of the procedure call, Page 3 is swapped out and Page 1 is swapped in.
- Page 1/3 is always in memory the shortest amount of time.
- The entire sequence is repeated for every iteration.

iv) Thrashing/continuously swapping pages

3 a) quantum

b) pre-emptive

c) virtual memory

d) low level scheduler

e) context switching

f) paging

g) non-preemptive

h) burst

i) segmentation

j) starvation

4 a) i) **From blocked to ready**

- Process is waiting for resource/IO operation to complete (blocked state).
- When IO operation completed, process goes into ready queue (ready state).

ii) **From running to ready**

- When process is executing, it is allocated a time slice (running state).
- The process is allocated time in processor.
- When time slice completed, interrupt occurs …
- … process can no longer use processor even though it is capable of further processing (ready state).

b) **A process cannot move directly from ready state to blocked state because**

- to be in blocked state, process must initiate some IO operation
- to initiate operation, process must be executing
- if process is in ready state, it cannot be executing/must be in running state.

c) i) exit/termination/completion

ii) when process has finished execution

d) **A low-level scheduler**

- decides which of processes is in ready state
- should get use of processor/be put in running state
- is based on position/priority
- is invoked after interrupt/OS call.

5 a) **Programs can access data from memory when using virtual memory because**

- program executes load process with a virtual address
- computer translates virtual address to give a physical address in memory
- if physical address not in memory, the OS loads it from the HDD
- computer then reads RAM using physical address and returns the data to program.

b) i) FIFO:

- When using first in first out (FIFO), the OS keeps track of all pages in memory using a queue structure
- The oldest page is at the front of the queue and is the first to be removed when a new page needs to be added.
- FIFO algorithms don't consider page usage when replacing pages; a page may be replaced simply because it arrived earlier than another page.
- It suffers from, what is known as, Belady's Anomaly where it is possible to have more page faults when increasing the number of page frames.

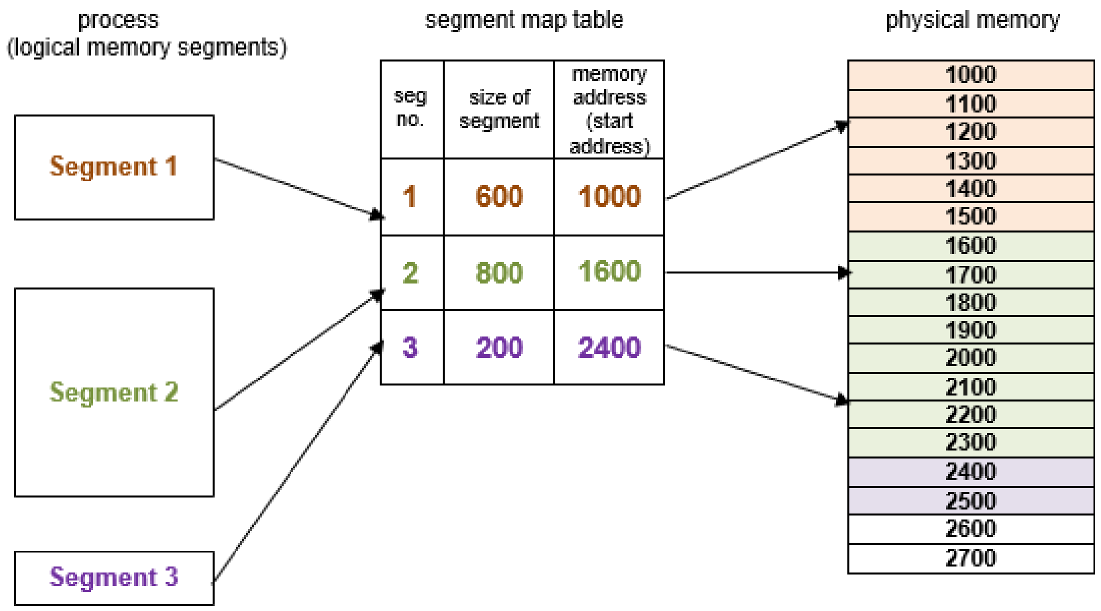
ii) OPR:

- Optimal page replacement looks forward in time to see which frame it can replace in the event of a page fault.
- The algorithm is actually impossible to implement; at the time of a page fault, the OS has no way of knowing when each of the pages will be replaced next.
- It tends to get used for comparison studies but has the advantage that it is free of Belady's Anomaly and also has the fewest page faults.

iii) LRU:

- With least recently used page replacement (LRU), the page which has not been used for the longest time is replaced.
- To implement this method, it is necessary to maintain a linked list of all pages in memory with the most recently used page at the front and the least recently used page at the rear.

6 a)



seg no.	size of segment	memory address (start address)
1	600	1000
2	800	1600
3	200	2400

b) paging is fixed size; segmentation is variable size; pages are smaller than segments

c) Types of interrupts

- device interrupt (e.g. printer out of paper)
- exception (e.g. division by zero)
- trap/software interrupt (e.g. software needs to access/use a resource).