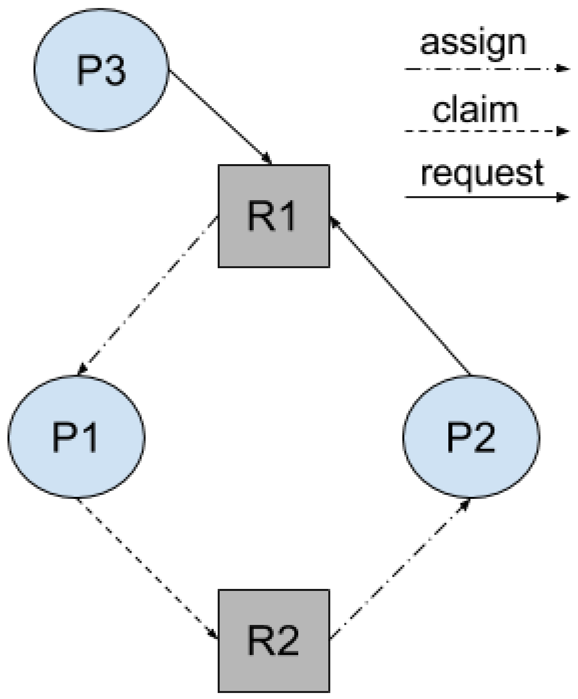
Short answers

<u>Chapter 7:</u>

1. Example of system:



a. This is an unsafe state because there is a cycle (P1, R2, P2, R1). The system can deadlock if P1 requests R2.

b. All the processes could complete without entering deadlock if P1 finishes using R1 and releases it. Then P2's request for R1 can be granted and it will then finish its work using R1 and R2. Then P3's request for R1 can be granted and it will then finish its work using R1.

c. CPU scheduler can affect if the system deadlocks or not. Suppose there's two processes P1 and P2 and two resources A and B. If P1 acquires the resources in the order of A->B, but P2 acquires them in the order of B->A, then it is possible for deadlock to occur. However, if the processes are given long time-slices (for example using FCFS) then they are seldom interrupted and hence fewer opportunities for another process to jump in and holding the resource (assuming on a single core CPU). On the other hand, using a scheduler that switches more frequently (again assuming on a single core CPU), such as the Round Robin with a short time quantum, gives more chances to interrupt P1 (or P2) in the time between acquiring A and B (or B and A) and hence more likely to deadlock.

3. a. An order in which processes may complete: P0 -> P3 -> P1 -> P2 -> P4

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P_0	4 2 1 2	4 2 1 2	1 1 1 0 (5 3 2 2)
P_3	1 4 2 4	1 4 2 4	5 2 1 0 (6 6 3 4)
P_1	5 2 5 2	5 2 5 2	4 5 0 3 (9 7 5 5)
P_2	2 3 1 6	2 3 1 6	9 5 4 2 (11 8 5 8)
P_4	3 6 6 5	3 6 6 5	9 6 2 5 (12 12 8 10)

b. Yes, if we allocated P_1 (1, 1, 0, 0) the available remaining resources is (2, 2, 2, 1) and then if we follow the same sequence as before (P_0 -> P_3 -> P_1 -> P_2 -> P_4) we will end up with a safe state.

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P_0	4 2 1 2	4 2 1 2	0 0 1 0 (4 2 2 2)
P_3	1 4 2 4	1 4 2 4	4 1 1 0 (5 5 3 4)
P_1	5 2 5 2	5 2 5 2	4 5 0 3 (9 7 5 5)
P_2	2 3 1 6	2 3 1 6	9 5 4 2 (11 8 5 8)
P_4	3 6 6 5	3 6 6 5	9 6 2 5 (12 12 8 10)

c. No, if we allocated P_4(0, 0, 2, 0) the available remaining resources is (3, 3, 0, 1) and neither of the remaining processes can finish (this includes P_4) because there is not enough resources to even satisfy any process to completion.

d. The request will not be granted immediately, there is not enough available resources to satisfy P_3's request for 3 instances of resource D because the current available instances of D is only 1. Depending on the system, it will either deny the request, place it on hold, or only allocate the remaining resources it has if it even has any left.

Chapter 8:

1. It is most efficient to break the address into X page bits and Y offset bits, rather than to perform arithmetic on the address to calculate the page number and offset. Because each bit position represents a power of 2, splitting an address between bits results in a page size with a power of 2.

2. Internal fragmentation is the wasted space within each allocated block because of rounding up from the actual requested allocation to the allocation granularity. External fragmentation is the various free spaced holes that are generated in either your memory or disk space. External fragmented blocks are available for allocation, but may be too small to be of any use.

3.

	External Fragmentation	Internal Fragmentation	Ability to share code
contiguous memory allocation (fixed size)	There is no external fragmentation	There is internal fragmentation	Does not allow processes to share code.
contiguous memory allocation (variable size)	There is external fragmentation	There is no internal fragmentation	Does not allow processes to share code.
pure segmentation	There is external fragmentation	There is no internal fragmentation	Able to share code between processes
pure paging	There is no external fragmentation	There is internal fragmentation	Able to share code between processes

4. In certain situations the page tables could become large enough that by paging the page tables, one could simplify the memory allocation problem (by ensuring that everything is allocated as fixed-size pages as opposed to variable-sized chunks) and also enable the swapping of portions of page table that are not currently used.