1. (a) All four Coffman conditions are satisfied here.

Mutual exclusive: Assume resources cannot shared by multiple threads at the same time.

No preemption: Assume there is no preemption.

Hold and wait & circular wait: Thread 0 owns Resource Y and waits for Resource X. Thread 2 owns Resource X and waits for Resource Y.

(b) There is no deadlock. After thread 1 finishes using an instance of resource X, it releases the resource. Thread 0 gets resource X and resource Y, and it can progress and finish working. Then, thread 0 releases resource Y, thread 2 get both resources and it can finish working. There is no deadlock. For multiple instance resources, the fact that Coffman conditions are true doesn't guarantee deadlock.

2. Each segment may have different length. In contrast, each page has the same size.

Different content in segment table and page table comes from different information needed to find physical address.

To convert logical address to physical address:

(1) Segment: Extract segment number and segment offset from logical address.
    Using segment number to find starting address and length in the segment table.
    We need to check if offset < segment length to make sure the visit is valid.
    Physical address = starting address of segment + offset.

(2) Page: Extract page number and page offset from logical address.
    Using page number to find frame number in the page table. Frame size is equal to page size.
    Frame number * frame size + offset is the physical address.
    If bits that are used to store offset are all 1, we get the page size. Offset is guaranteed to be smaller than page size. Starting address = frame number * frame size.
    Therefore, we don't need to store page length and starting address of a page in page table. It is enough to store frame number.

We can record base in a page table. Then just use base address of page + offset to get physical address. However, if we have the frame number in page table, base address of page is redundant information, we can calculation it by frame number * frame size.

The length of each page is not needed. Each page has the same size. Offset is guaranteed to be smaller than page size.

3.

(a) LRU

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reference	7	2	3	1	2	5	3	4	6	7	7	1	0	5	4
Frame 0	7	7	7	1	1	1	3	3	3	7	7	7	7	5	5
Frame 1	X	2	2	2	2	2	2	4	4	4	4	1	1	1	4
Frame 2	X	X	3	3	3	5	5	5	6	6	6	6	0	0	0
Page Fault(Y/N)	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y

Total page faults = 13

(b) FIFO

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reference	7	2	3	1	2	5	3	4	6	7	7	1	0	5	4
Frame 0	7	7	7	1	1	1	1	1	6	6	6	6	0	0	0
Frame 1	X	2	2	2	2	5	5	5	5	7	7	7	7	5	5
Frame 2	X	X	3	3	3	3	3	4	4	4	4	1	1	1	4
Page Fault(Y/N)	Y	Y	Y	Y	N	Y	N	Y	Y	Y	N	Y	Y	Y	Y

Total page faults = 12

4.

(a) From 8 entries in page table, we can see there are 3 bits in logical address to store page number.

Since there are 16 bytes for page size, we can see there are 4 bits in logical address to store offset.

Page faults will be caused by:

Page 2: Page number bits: 010 Page offset: 0000 to 1111 (15 in decimal). Convert this to decimal:

0100000 = 2 ^ 5 = 32

0101111 = 2 ^ 5 + 15 = 47

[32, 47]

Page 3: Page number bits: 011. Page offset: 0000 to 1111.

2^5 + 2^4 + 0 = 32 + 16 = 48

2^5 + 2^4 + 15 = 32 + 16 + 15 = 63

[48, 63]

Page 5: Page number bits: 101. Page offset: 0000 to 1111.

1010000: 2^6 + 2 ^4 = 64 + 16 = 80

1011111: 2^6 + 2 ^4 + 15 = 64 + 16 + 15 = 95

[80, 95]

Page 7: Page number bits: 111. Page offset: 0000 to 1111.

1110000: 2 ^ 6 + 2 ^ 5 + 2 ^ 4 = 64 + 32 + 16 = 112

1111111: 112 + 15 = 127

[112, 127]

All logical address ranges that could cause page faults:

[32, 47] U [48, 63] U [80, 95] U [112, 127]

(b) Convert logical address to physical address:

(1) 0. 0 / 16 = 0: page number 0. 0 % 16 = 0 offset 0. Page number 0 corresponds to frame number 3. Frame number * frame size + offset = 3 * 16 + 0 = 48.

(2) 17. 17 / 16 = 1. Page number 1. 17 % 16 = 1. Offset 1. Page number 1 corresponds to frame number 0. Frame number * frame size + offset = 0 * 16 + 1 = 1.

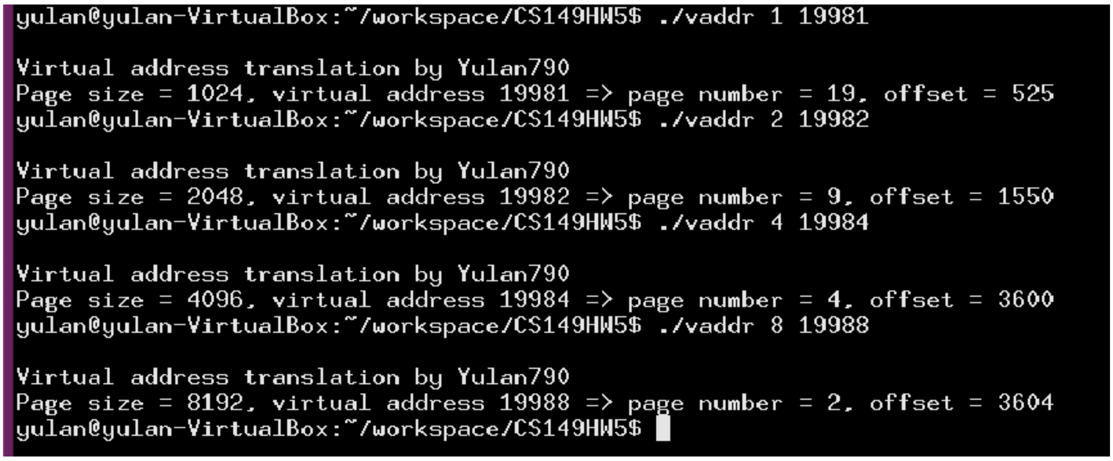
(3) 31. 31 / 16 = 1. Page number 1. 31 % 16 = 15. Offset 15. Page number 1 corresponds to frame number 0. Frame number * frame size + offset = 0 * 16 + 15 = 15.

(4) 32/ 16 = 2. Page number 2. There is no corresponding frame in main memory. Conversion is not possible.

(5) 100 / 16 = 6. Page number 6. 100 % 16 = 4. Offset 4. Page number 6 corresponds to frame

number 1. Frame number * frame size + offset = 1 * 16 + 4 = 20.

5. (Extra Credit):



```
yulan@yulan-VirtualBox:~/workspace/CS149HW5$ ./vaddr 1 19981

Virtual address translation by Yulan790
Page size = 1024, virtual address 19981 => page number = 19, offset = 525
yulan@yulan-VirtualBox:~/workspace/CS149HW5$ ./vaddr 2 19982

Virtual address translation by Yulan790
Page size = 2048, virtual address 19982 => page number = 9, offset = 1550
yulan@yulan-VirtualBox:~/workspace/CS149HW5$ ./vaddr 4 19984

Virtual address translation by Yulan790
Page size = 4096, virtual address 19984 => page number = 4, offset = 3600
yulan@yulan-VirtualBox:~/workspace/CS149HW5$ ./vaddr 8 19988

Virtual address translation by Yulan790
Page size = 8192, virtual address 19988 => page number = 2, offset = 3604
yulan@yulan-VirtualBox:~/workspace/CS149HW5$ 
```