# OS Final Answer

# Q1

- FCFS

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P1 | P1 | P1 | P1 | P1 | P2 | P2 | P2 | P2 | P3 | P3 | P3 | P4 | P4 |

- Preemptive SJF

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P2 | P2 | P2 | P2 | P4 | P4 | P3 | P3 | P3 | P1 | P1 | P1 | P1 | P1 |

- Non-preemptive priority

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P1 | P1 | P1 | P1 | P1 | P4 | P4 | P3 | P3 | P3 | P2 | P2 | P2 | P2 |

| | P1 | P2 | P3 | P4 |
|---|---|---|---|---|
| Turnaround | 6 | 14 | 8 | 4 |
| Wait | 0 | 10 | 5 | 2 |
| Response | 0 | 10 | 5 | 2 |

# Q2

- EDF

| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P2 | P2 | P2 | P1 | P2 | P1 | P2 | P2 | P1 | P2 | P2 | P1 | P2 |

- RM

| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P2 | P2 | P1 | P2 | P2 | P1 | P2 | P2 | P1 | P2 | P2 | P1 | P2 |

## Q3

- CPU affinity means a process can only be ran on a specific CPU.
- It can improve performance by reducing cache miss.
- But it can also lower CPU utilization due to the scheduling constraint

# Q4

- Initial status (2%)

| | Max | | | Allocated | | | Needed | | |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | A | B | C | A | B | C |
| P0 | 2 | 2 | 4 | 0 | 1 | 1 | 2 | 1 | 3 |
| P1 | 6 | 5 | 3 | 0 | 0 | 2 | 6 | 5 | 1 |
| P2 | 7 | 5 | 3 | 0 | 1 | 0 | 7 | 4 | 3 |
| P3 | 3 | 2 | 2 | 3 | 0 | 2 | 0 | 2 | 0 |

- Available instance (3%): (7,3,2); (10,3,4); (10,4,5)
- Safe sequence: P3,P0,P2,P1

# Q5

- Mutual exclusion: only 1 process at a time can use a resource
- Hold & Wait: a process holding some resources and is waiting for another resource
- No preemption: a resource can be only released by a process voluntarily
- Circular wait: there exists a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that $P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P_n \rightarrow P_0$

# Q6

(a) spin-lock will be more efficient for small or short critical section. Because non-busy waiting requires system call.

(b) Monitor is easier to use for solving more complex synchronization problem.

But software approach can is more efficient.

## Q7

(a)

Use “w” lock by  “wait(w)” and “signal(w)”

(b)

When writer come first, the first writer will lock “r” lock to avoid other readers come into critical section.

When writer come after readers, the writer will compete “r” lock with incoming one reader(use mutex_3 to ensure). Once writer get “r” lock, it will prevent other reader come into critical section.

# Q8: Bakery Algorithm (*n* processes)

```
//Process i:
do {
  choosing [ i ] = TRUE ;
  num[ i ] = max(num[0],num[1],…,num[n-1]) + 1;
  choosing [ i ] = FALSE ;
  for (j = 0; j < n; j++) {
     while (choosing [ j ] ) ;
     while ((num[ j ] != 0) &&
              ((num[ j ], j) < (num[ i ], i))) ;
  }
      critical section
  num[ i ] = 0 ;
      reminder section
} while (1) ;
```

Get ticket

FCFS

Cannot compare when num is being modified

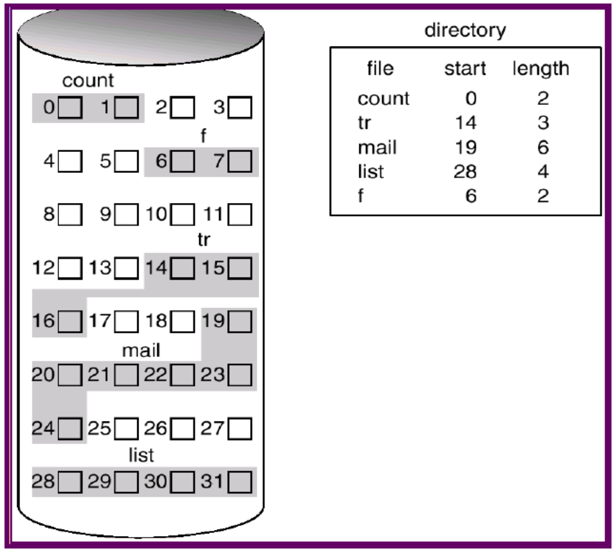
release ticket

- Bounded-waiting because processes enter CS on a First-Come, First Served basis

## Q9

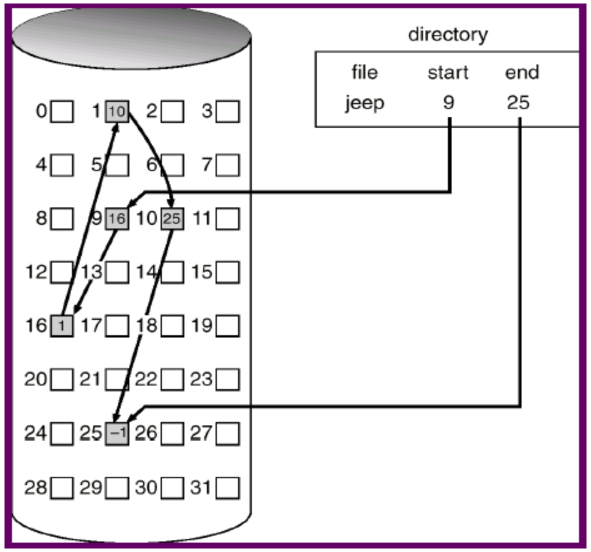
- If user is jack, he is the owner. According to the ACL, he can have read, write and execution permission.
- If a user is not jack, but is in the group for students, he can have write permission.
- Other account can only have execution permission

# Q10



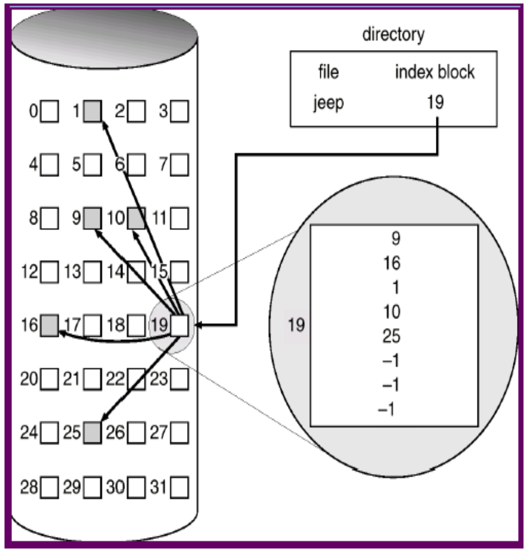
directory

| file | start | length |
|---|---|---|
| count | 0 | 2 |
| tr | 14 | 3 |
| mail | 19 | 6 |
| list | 28 | 4 |
| f | 6 | 2 |



directory

| file | start | end |
|---|---|---|
| jeep | 9 | 25 |



directory

| file | index block |
|---|---|
| jeep | 19 |

- Database should use index allocation, because it needs better performance for random access
- File system can use linked allocation, because sequential access.
- Swap should use contiguous, before its size is fixed.

# Q11

- SSTF: 250, 120, 10, 740, 900, 2190, 2400
- LOOK:  250, 740, 900, 2190, 2400, 120, 10
- C-LOOK: 250, 740, 900, 2190, 2400, 10, 120,
- The request wait is no uniform in LOOK, because it revisit the cylinder at two end within shorter time period.

# Q12

(a)RAID0 improve throughput by reading data in parallel

(b)RAID improve reliability. When disk failure occurs, it can read from replicated data.

(c) RAID3 needs to reconstruct a block from all disks and require disk synchronization.

(d) Balance the load

## Q13

- Port-mapped I/O:
  - use port address to communicate with device directly
  - Require i/o instruction
  - Less efficient
- Memory-mapped I/O:
  - Reserve a memory space for a device
  - Communicate with device through memory access
  - More data content can be read/write to a device at once

# Q13

- Poll I/O:
  - Processor has to continuously check the device status register to detect I/O completion
  - Could waste CPU cycle during long I/O delay
- Interrupt I/O:
  - Device notify processor its I/O completion by sending an interrupt signal
  - Interrupt is a costly operation

## Q13

- Programmed I/O
  - I/O is controlled by the program though CPU
  - I/O causes additional overhead to CPU
- DMA:
  - I/O is controlled by another processor (i.e. DAM controller)
  - It combines memory-mapped and interrupt approach
  - CPU is only interrupted at the beginning and at the end of an I/O operation

**14.** Consider a byte-addressable computer system with a 16-bit virtual address, total physical memory size 4KB, page size is 128 Bytes, the number of segments per process is 128. Given the following segment table, page table and a 16 bits hexadecimal logical address "0312", complete the address translation diagram below.

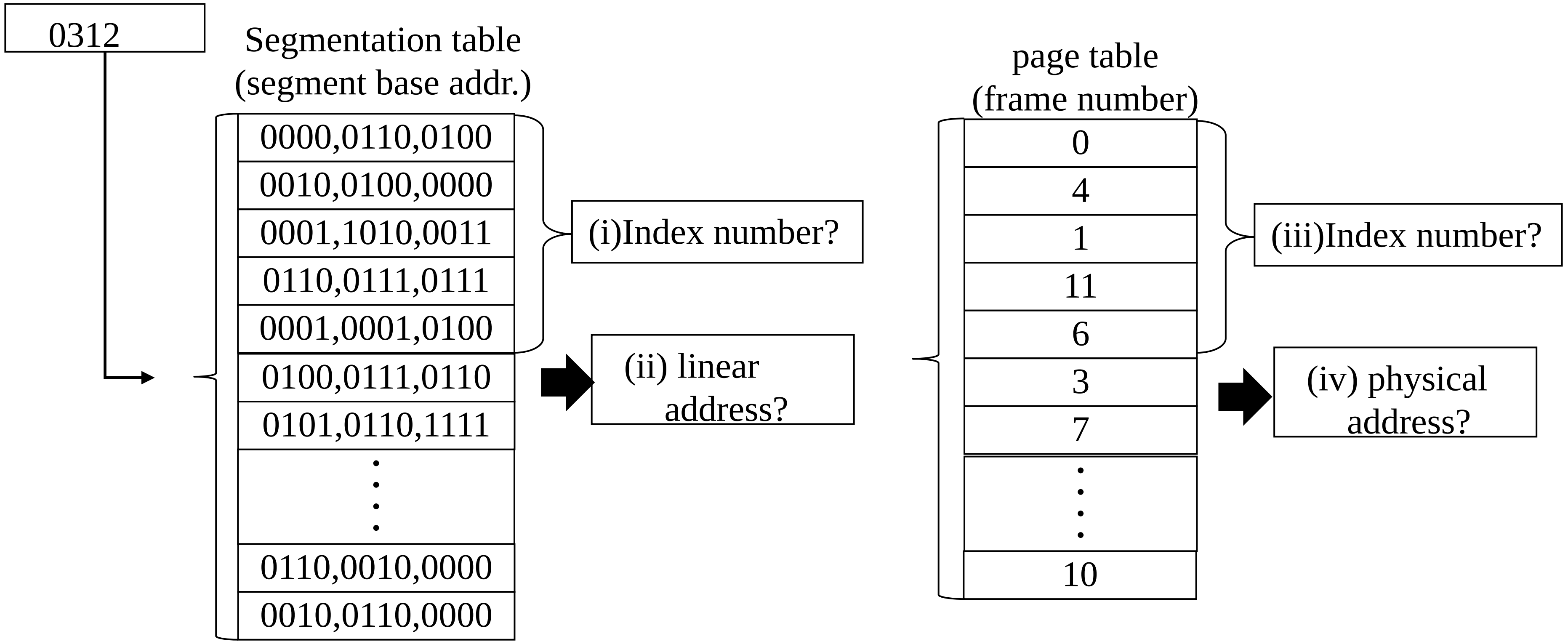
(i) (2%) segment number = 7bits ➔ logical address = 0000,0011,0001,0010

➔ segment number = 0000001 = 1

(i) (2%) linear address: 0010,0100,0000 + 1,0001,0010 = 0011,0101,0010

(ii) (2%) page table index: page offset = 7 bits ➔ page number = 5 bits ➔ 00110 = 6

(iii) (2%) physical address: 0011,1000,0000 + 100,0000 = 0011,1100,0000


0312
Segmentation table (segment base addr.)
0000,0110,0100
0010,0100,0000
0001,1010,0011
0110,0111,0111
0001,0001,0100
0100,0111,0110
0101,0110,1111
0110,0010,0000
0010,0110,0000
(i)Index number?
(ii) linear address?
page table (frame number)
0
4
1
11
6
3
7
10
(iii)Index number?
(iv) physical address?