1. virtual address 공간의 크기는 physical address 공간의 크기와 동일하지 않아도 된다.

2. virtual address에서 page offset의 크기는 physical address의 page offset 크기와 동일해야 한다

3. 시스템에 추가적인 메모리 자원을 추가한다고 해서 TLB hit가 일어나는 비율이 증가하지 않는다

4. CPU가 어떤 메모리 주소에 접근할 때 TLB miss가 일어났다고 page fault가 일어나는 것은 아니다

5. 어떤 자원에 대해 mutual exclusion을 지원하는 것이 해당자원에 대해 deadlock발생을 줄여주지 않는다

6. lock 을 옳게 구현하기 위해서 mutual exclusion, progress, bounded waiting 이 지켜져야 한다

7. Critical section 은 프로그램에서 보안적으로 중요한 정보를 담고있는 데이터 영역이 아니다

    a. HDD에서 random access time은 rotational latency가 가장 큰 영향을 미치지 않는다

8. Directory는 파일의 한 종류로 어떤 경로에 존재하는 파일 목록을 가진다

9. Resource allocation group-raph에서 cycle이 존재하는 상태가 항상 데드락을 발생하는 것은 아니다

10. 부팅 프로세스동안 시스템이 초기 bootstrapping 정보를 얻어오는 곳은 boot block 이다

11. copy on write
    - 메모리에 대한 write가 안전하게 수행될 수 있도록 지원하는 메커니즘이다 X
    - fork를 통해 새로운 프로세스를 만들 때 효율적으로 수행할 수 있다 O
    - COW를 위해서 각 virtual page 마다 매핑 개수가 관리되어야 한다 O
    - COW로 인해 메모리 낭비가 일어날 수 있다 X

12. synchronization
    - 스핀락은 블로킹 방식으로 구현된 락 메커니즘이다 X
    - 카운팅 세마포어는 이용 가능한 자원 개수가 복수인 경우 사용한다 O
    - 피터슨 알고리즘은 세개 이상의 태스크에 대해 적용 가능하다 X
    - 테스트 앤 셋은 하드웨어가 제공하는 아토믹 인스트럭션이다 O

13. page replacement
    - LRU알고리즘은 구현이 어려워서 LRU approximation 알고리즘을 이용한다 O
    - 사용하는 페이지 교체 알고리즘에 따라 TLB hit 비율이 달라질 수 있다 X

- FIFO를 사용하는 경우 메모리가 많이 주어졌을 때 오히려 성능하락이 일어날 수 있다 O
- 세컨드찬스 알고리즘은 PTE의 reference bit를 이용하는 알고리즘이다 O

14. HDD disk scheduling
    - SSTF는 seek distance를 줄이는데 효과적이다 O
    - FCFS는 늦게 들어온 i/o요청부터 처리한다 X
    - SCAN은 starvation을 일으킬 수 있으며 C-SCAN으로 해결한다 X
    - disk는 여러 i/o 요청을 병렬적으로 처리할 수 있다 X

15. memory management
    - virtual page마다 다른 protection 설정을 하는 것이 가능하다 O
    - Fixed partition 으로 external fragmentation이 일어날 수 있다 X
    - internal fragment는 paging 으로 해결이 가능하다 X
    - hierarchical page table 은 메모리 액세스를 빠르게 하기 위한 구조이다 X

16. demand paging
    - MMU가 page fault를 일으키며 OS의 page fault handler에 의해 다뤄진다 O
    - pure demand paging을 사용할 경우 어떠한 버추얼 페이지도 메모리 프레임에 할당되지 않은 상태로 프로세스가 시작된다 O
    - page fault는 항상 swap out 된 버추얼 페이지에 대한 접근 때문에 발생한다 X
    - demand paging은 process의 locality 특성을 이용한 것이다 O

17. 2개의 쓰레드 동기화 락 코드

```
int interested[2] = {FALSE, FALSE}
void acquire(int my_id){
        int other = 1 - my_id
        interested[my_id] = TRUE
        while(interested[other]
}
void release(int my_id){
        interested[my_id] = FALSE
}
```

    - lock 동작은 작동하지 않는다
    - mutual exclusion은 만족하지만 progress를 만족하지 않는다

18. 데드락 발생

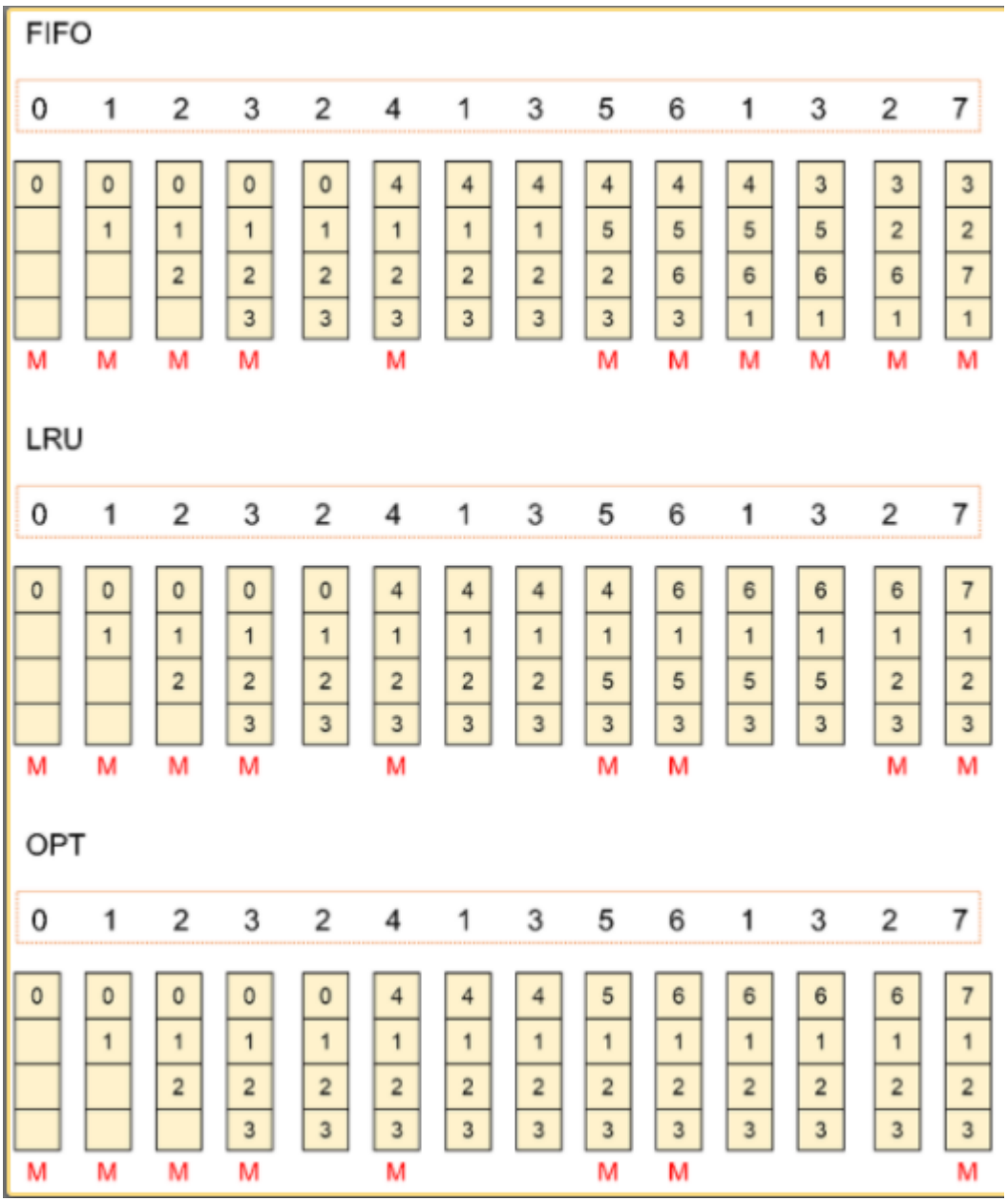
	Allocation				Max			
	A	B	C	D	A	B	C	D
P1	0	1	1	4	5	3	2	4
P2	2	2	0	3	2	4	3	3
P3	0	0	3	4	2	6	5	6
P4	1	0	1	0	2	2	3	2
P5	4	1	1	0	7	3	2	2

Available resources			
A	B	C	D
1	2	2	2

자원 ABCD 프로세스 P 1 2 3 4 5

- 데드락 발생 안함
- p4 2232
- p2 3425
- p5 7535
- p1 7645
- p3 7679

19. 8개의 virtual pages로 이루어진 virtual address 공간과 4개의 page frame 으로 이루어진 physical address 공간이 있다. 01232413561327 virtual page에 접근한다 몇 번의 page fault 일어나는가



20. 4 KB 크기의 block을 가지는 Unix 파일시스템이 있다 “/user/doc/test.txt”에 접근하기 위해 하드디스크에 접근하는 과정을 써라 하드디스크에 몇 번의 read operation 이 필요한가 “/user/doc/test.txt”는 40KB inode는 12개 direct block entry 가짐
    - inode / read 1
    - directory / read 2
    - inode user read 3
    - directory user read 4
    - inode doc read 5
    - directory doc read 6
    - inode test.txt read 7
    - test.txt의 10개 다이렉트 블록 read, read operation 한 번으로 각 블록 읽음

8-17

21. 34 bit virtual address 공간을 hierarchical page table 로 관리, virtual page = 16KB, Page table entry = 16byte
    - 전체 virtual address 공간의 크기 = 2^34 16GB
    - 34비트 버추얼 어드레스 중 페이지 오프셋 사용하는 크기 = 14bit
    - 하나의 페이지 테이블이 갖는 페이지 테이블 엔트리 개수 = 2^10 1024개
    - 사용중인 hierarchical page table 레벨 = 2 level page table