

# 운영체제론

## 2012년 시행 5급(기술) 공채 제2차시험

응시번호 :

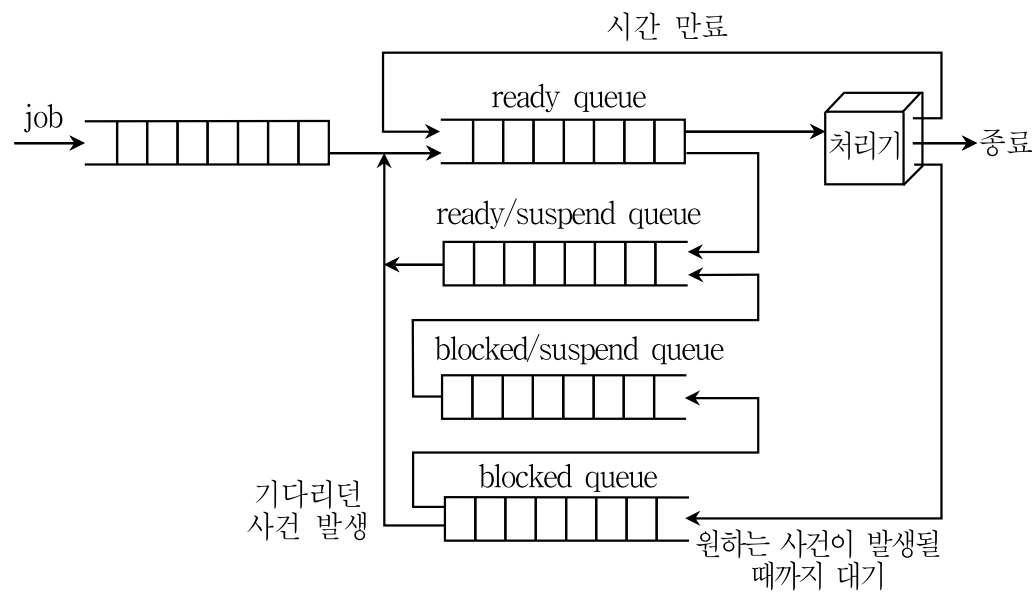
성명 :

제 1 문. 요구 페이징(demand paging)을 사용하는 가상 메모리 시스템을 이용하여 0에서 64 KB까지 16 비트의 가상 주소를 생성할 수 있는 컴퓨터가 있다고 하자. 이 컴퓨터는 물리 기억용량으로 32 KB만 가지고 있다. 다음 그림은 현재 이 컴퓨터의 MMU가 사용하는 16개의 4 KB 페이지를 갖는 페이지 테이블을 간략하게 나타낸 것이다. 만일 프로세서가 “MOV REG, 0x1000”이라는 명령을 사용한다면, 이는 레지스터 REG에 가상 메모리 주소 0x1000번지에 있는 내용을 복사하라는 명령이다. 클록 페이지 교체 알고리즘을 사용하고, 다음 교체 페이지 후보는 0번 페이지이며 index가 증가하는 순서로 조사한다고 가정한다. 다음 물음에 답하시오. (총 30점)

Index	Frame Number	Valid Bit	Reference Bit
0	010	1	1
1	001	1	1
2	110	1	0
3	000	1	0
4	100	1	0
5	011	1	0
6	000	0	0
7	000	0	0
8	010	0	0
9	101	1	0
10	000	0	0
11	111	1	0
12	000	0	0
13	100	0	0
14	111	0	0
15	000	0	0

- 1) MOV REG, 0x2336의 명령에 대하여 매핑 되는 물리주소를 16진수로 답하고, 물리주소의 변환 과정을 설명하시오. 만약 페이지 테이블이 변경된다면 페이지 테이블에 나타나는 변화를 단계별로 서술하시오. (7점)
- 2) 위의 명령 실행 후, MOV REG, 0x800C의 명령이 실행될 때, 매핑되는 물리 주소를 16진수로 답하고 변환 과정을 설명하시오. 만약 페이지 테이블이 변경된다면 페이지 테이블에 나타나는 변화를 단계별로 서술하시오. (10점)
- 3) 위의 두 명령어 실행 후의 페이지 테이블을 inverted page table로 구현을 하였을 경우를 그림으로 나타내고, 두 페이지 테이블을 비교하시오. (단, 각 페이지 테이블의 하나의 엔트리는 2 바이트라고 가정한다) (6점)
- 4) 위의 시스템에서 만약 가상 주소가 40 비트로 확장되었다고 가정하고 하나의 페이지 테이블을 한 페이지에 저장하도록 다단계 페이지 테이블을 구성하려고 한다. 다단계 페이지 테이블의 구조를 그림으로 나타내고, 하나의 페이지 테이블 엔트리가 4 바이트라고 가정하였을 경우 전체 페이지 테이블이 차지하는 공간을 계산하시오. 그리고 다단계 페이지 테이블을 사용하는 이유를 1단계 페이지 테이블의 크기와 비교하여 설명하시오. (7점)

제 2 문. 다음 그림은 싱글 프로세서 환경에서 일반적인 스케줄링 대기큐 구조이다. 다음 물음에 답하시오. (총 20점)



- 1) 위 그림에서 보는 바와 같이, 보통 다양한 프로세스 수행요청에 따라 시스템은 조금은 다른 스케줄링 기법을 사용한다. 이 기법들을 보면, 처리기 스케줄링은 장기(long-term, 또는 job) 스케줄링, 중기(medium-term 또는 swap) 스케줄링, 단기(short-term, 또는 CPU) 스케줄링의 3가지 유형을 가진다. 3가지 유형의 처리기 스케줄링을 프로세스 상태 전이도를 이용하여 설명하시오. (단, 여기에서 프로세스의 상태는 생성(new), 준비(ready), 수행(running), 블록(blocked), 보류중인 준비(ready/suspend), 보류중인 블록(blocked/suspend), 종료(exit)의 7가지 상태를 가질 수 있다고 가정한다) (9점)
- 2) 단기 스케줄링 방법 중에 FCFS, RR(Round Robin), SRT(Shortest Remaining Time) 방법이 있다. 이들 방법들에 대해서 스케줄링 기준, 선점/비선점 측면, 처리량, 응답시간, 문맥교환비용, 기아발생 여부를 각각 비교 기술하시오. (6점)

3) 단기 스케줄링에서 다음과 같은 프로세스 집합이 주어질 때, FCFS, RR, SRT 방법으로 스케줄링된 결과를 테이블에 표시하시오. (단, RR의 타임퀀텀은 1이며, 동시성이 있는 경우 새로 들어온 프로세스에게 우선 순위를 준다) (5점)

프로세스	도착시간	수행시간
A	0	6
B	1	4
C	4	3

알고리즘	프로세스	0	5	10	13
FCFS	A				
	B				
	C				
RR	A				
	B				
	C				
SRT	A				
	B				
	C				

제 3 문. ‘식사하는 철학자(dining philosopher)’ 문제에서 식사를 할 수 없는 상태로 무한히 대기하는 데드락에 걸릴 수 있다. 다음 코드는 식사하는 철학자 문제에 대한 해결책을 제시하고 있다. 이와 관련하여 다음 물음에 답하시오. (총 20점)

```
#define N 5
#define LEFT (i+N-1)%N
#define RIGHT (i+1)%N
#define THINKING 0
#define HUNGRY 1
#define EATING 2
typedef int semaphore;
int state[N]; semaphore mutex = 1; semaphore s[N];

void philosopher(int i) { /*1*/
    while (TRUE) { /*2*/
        think(); /*3*/
        take_forks(i); /*4*/
        eat(); /*5*/
        put_forks(i); /*6*/
    }
}

void take_forks(int i) { /*7*/
    wait(&mutex); /*8*/
    state[i] = HUNGRY; /*9*/
    test(i); /*10*/
    signal(&mutex); /*11*/
    wait(&s[i]); /*12*/
}
```

```

void put_forks(int i) {                                /*13*/
    wait(&mutex);                                       /*14*/
    state[i] = THINKING;                               /*15*/
    test(LEFT);                                         /*16*/
    test(RIGHT);                                        /*17*/
    signal(&mutex);                                     /*18*/
}

void test(int i) {                                      /*19*/
    if state[i] == HUNGRY
        && state[LEFT] != EATING
        && state[RIGHT] != EATING) {                 /*20*/
        state[i] = EATING;                             /*21*/
        signal(&s[i]);                                  /*22*/
    }
}

```

- 1) 문장 /\*12\*/의 역할이 무엇인지 설명하시오. (6점)
- 2) 만약 문장 /\*15\*/가 문장 /\*16\*/과/\*17\*/ 뒤로 옮겨졌다고 생각했을 경우 어떤 문제가 발생할 수 있는 지 명확히 지적하시오. 지적할 때, 문장 번호를 적절히 활용하면서 설명하시오. (9점)
- 3) 식사하는 철학자 문제를 해결하기 위해 다음의 방법을 고안하였다.

왼손잡이 철학자는 왼쪽 포크를 먼저 들도록 강제하고, 오른손잡이 철학자는 오른쪽 포크를 먼저 들게 한다.

왼손잡이 철학자와 오른손잡이 철학자가 각각 최소한 한 명씩 존재할 경우,  
위 방법은 테드락 문제를 해결함을 증명하십시오. (5점)

제 4 문. 실행시간 링킹(run time linking), 적재시간 링킹(load time linking), 컴파일시간 링킹(compile time linking 또는 static linking) 중에서 어느 것을 선택할지는 운영체제를 설계함에 있어서 고려해야할 중요한 이슈이다. 즉 성능 저하를 가져오지 않으면서도 한정된 자원을 효율적으로 사용할 수 있도록 설계되어야 하는 운영체제 설계의 문제이다. 링킹과 관련하여 다음 물음에 답하시오.

(총 15점)

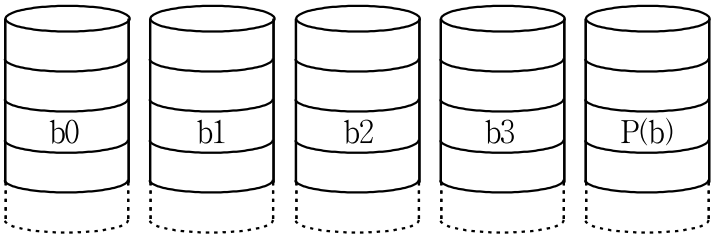
- 1) 만일 어떤 프로그램의 크기가 2 MByte이고, 라이브러리 모듈은 각각 1 MByte이며 이 프로그램은 실행 상황에 따라 최대 4개의 라이브러리 모듈을 링킹하여 사용한다고 가정한다. 이 때, 1 MByte를 로드하는데 걸리는 시간은 1초이며 로더 프로그램을 호출하여 실행하는데 소요되는 시간은 0.1초라고 한다. 아래 표에 필요 디스크 저장 공간과 지연 시간을 계산해 넣으시오. (10점)

링킹방법	응용프로그램을 저장하는데 필요한 디스크 공간(MB)	응용프로그램 기동을 위한 소요 시간	모듈 사용을 위한 실행 지연 시간 (4개 모듈 사용)	모듈 사용을 위한 실행 지연 시간 (2개 모듈 사용)	모듈 사용을 위한 실행 지연 시간 (1개 모듈 사용)
static					
load time					
run time					

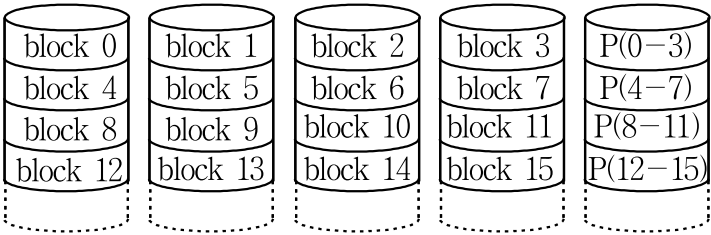
- 2) 위 결과로부터 미루어 어느 링킹 방법을 채택하는 것이 가장 적절하겠는가? 자원 소모의 면을 고려하여 그 이유를 기술하시오. (5점)

제 5 문. 다중 디스크를 사용하여 다양한 방식으로 데이터를 구성하고 신뢰성을 향상 시키기 위한 방안으로 RAID(Redundant Array of Independent Disks) 시스템이 널리 사용되고 있다. 다음 그림의 RAID level 3부터 level 6까지의 구성도를 참고하여 다음 물음에 답하시오.

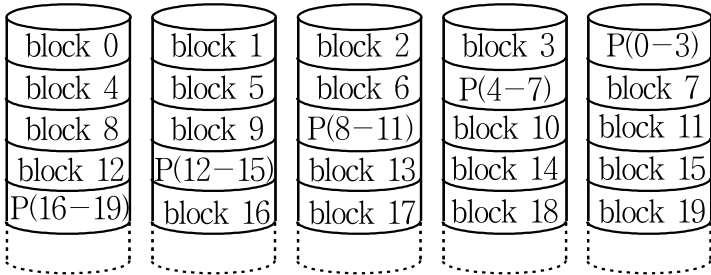
(총 15점)



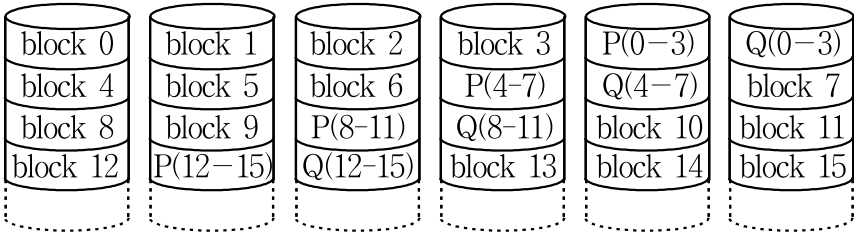
(a) RAID 3



(b) RAID 4



(c) RAID 5



(d) RAID 6

- 1) RAID level 3에서 RAID level 6까지의 특징을 각각 설명하시오. (8점)
- 2) RAID level 3에서 드라이브 실패가 발생하면 패리티 드라이브가 접근되고 나머지 장치들로부터 데이터가 재구성된다. X0부터 X3에 데이터를 저장하고 X4가 패리티 디스크인 다섯 개의 드라이브를 가진 디스크 배열이 있다고 가정할 때 i번째 비트의 패리티는 다음과 같이 계산된다.  $\oplus$ 는 배타적-OR 함수(exclusive-OR function)이다.

$$X4(i) = X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i)$$

- 만약 드라이브 X1이 실패했다고 하면 어떻게 복구할 수 있는지를 위의 식을 응용하여 설명하시오. (4점)
- 3) RAID level 4와 RAID level 5를 비교하였을 때, 병목현상(bottleneck)의 관점에서 성능의 우위를 기술하시오.(3점)

## 행정안전부 시험출제과장