

3주차 실습 보고서

실습 주제

- MicroC OS O(1) scheduler과 관련 data structure 활용
 - ready list에서 우선순위가 가장 높은 번호 선택하기

성공 실패 여부

- 성공

원인 분석

1. 서로 다른 번호인지 체크

```
rand_num = random(64);
head = rand_num >> 3;
tail = rand_num & 0x07;
if(myRdyTbl[head] & myMapTbl[tail]) {
    continue;
}
```

1. myRdyTbl[head]

head는 rand_num의 group index.

즉, myRdyTbl[head]는 rand_num이 속한 group value.

2. 랜덤으로 생성한 rand_num에 대해 & 0x07 을 하면

rand_num : 0 0 1 0 0 1 0 0 & 0 0 0 0 0 1 1 1 == 0 0 0 0 0 1 0 0 = 0x04

rand_num의 하위 3bit만을 추출한다.

즉, rand_num의 group value 에서의 index 값 획득.

3. myMapTbl[0x04]값은 0x04번째 index가 1이고 나머지가 전부 0인 8bit값.

```
myRdyTbl[0x04] == 00010000
```

즉, rand_num가 속한 group value 획득.

4. if `myRdyTbl[head] == myMapTbl[0x04]` 이면 기존에 rand_num이 있었다는 의미
ex) `myRdyTbl[head] == 00010010` , `myMapTbl[0x04] == 00010000`

2. myMapTbl, myRdyTbl 값에 입력

```
else {  
    *myRdyGrp |= myMapTbl[head];  
    myRdyTbl[head] |= myMapTbl[tail];  
    return rand_num;  
}
```

1. `myRdyGrp`은 특정 그룹에 어떠한 값이 있음을 나타내는 bit.
`myRdyGrp`에서 rand_num group 을 나타내는 값을 1로 만들어 줌.
2. `myRdyTbl[head]` 는 rand_num의 group value.
group value중 rand_num의 index자리를 1로 변경.

3. 최솟값 찾기

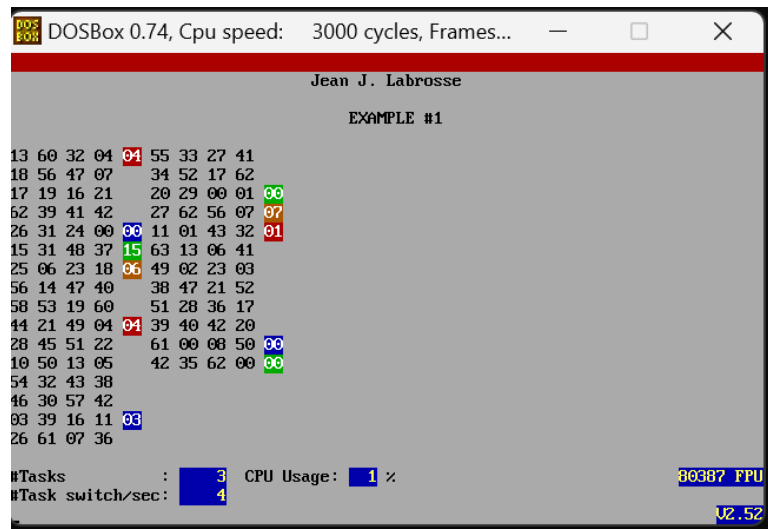
```
n = myUnMapTbl[myRdyGrp];  
m = myUnMapTbl[myRdyTbl[n]];  
min = (n << 3) + m;
```

1. `myUnMapTbl[myRdyGrp]`
 - 모든 group 중 가장 작은 그룹의 index
2. `myUnMapTbl[myRdyTbl[n]]`
 - 1에서 구한 그룹 중 가장 작은 수의 offset
3. $(n << 3)$
 - 한 그룹은 8개씩 이므로, 그룹으로부터 숫자를 얻기 위해 8을 곱한다

4. +m

- 2에서 구한 offset만큼 더하면, 가장 작은 값을 얻을 수 있다.

결과



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frames...
Jean J. Labrosse
EXAMPLE #1
13 60 32 04 04 55 33 27 41
18 56 47 07 34 52 17 62
17 19 16 21 20 29 00 01 00
62 39 41 42 27 62 56 07 07
26 31 24 00 00 11 01 43 32 01
15 31 48 37 15 63 13 06 41
25 06 23 18 06 49 02 23 03
56 14 47 40 38 47 21 52
58 53 19 60 51 28 36 17
44 21 49 04 04 39 40 42 20
28 45 51 22 61 00 08 50 00
10 50 13 05 42 35 62 00 00
54 32 43 38
46 30 57 42
03 39 16 11 03
26 61 07 36

#Tasks : 3 CPU Usage: 1 %
#Task switch/sec: 4
80387 FPU
02.52
```

5. 결과 분석

의도한 대로 4번의 0을 출력하고 종료된다.