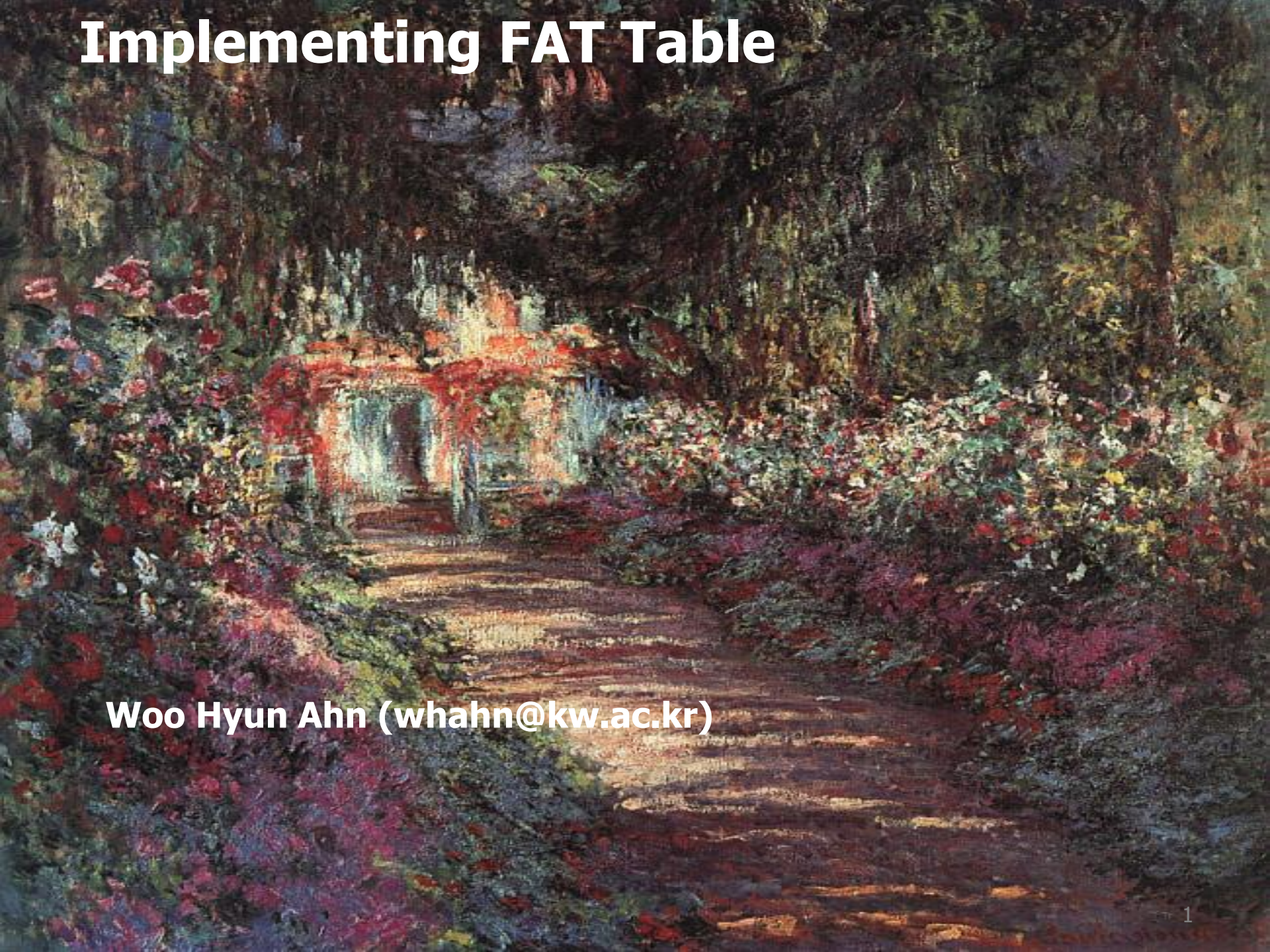
Implementing FAT Table

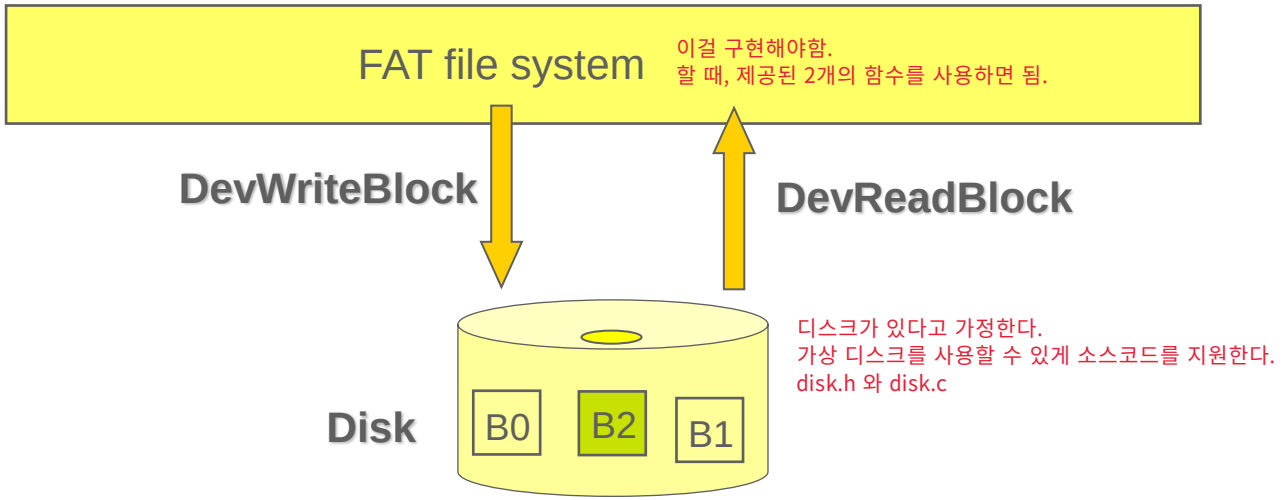


Woo Hyun Ahn (whahn@kw.ac.kr)

The Relation of File System & Disk

FAT Table을 구현해보자!

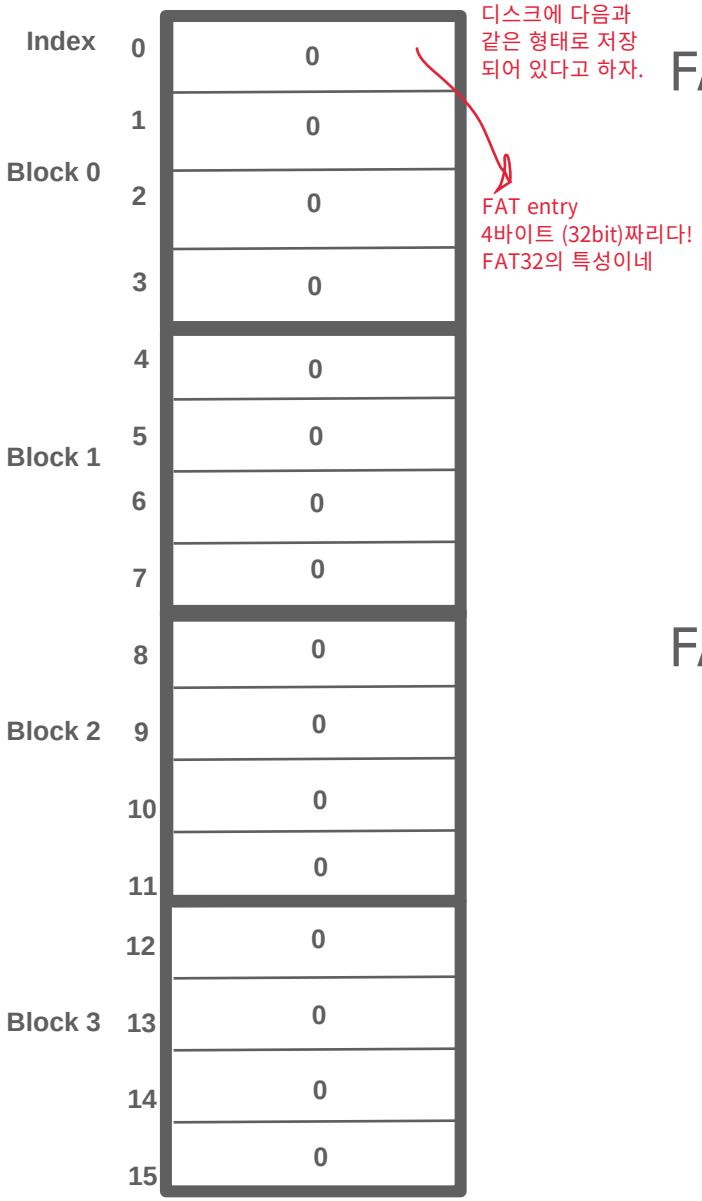
logical block ?
physical block ?
FAT table ?
이 뭔지 이해를 해야 과제 수행 가능



- Two functions is given for read/writing a block from/to disk.
  - DevReadBlock(int blkno, char* pBuf);
  - DevWriteBlock(int blkno, char* pBuf);

특정 디스크 블럭 번호에 블록 데이터를 읽거나 쓸 수 있는
과제에서 제공해주는 API

FAT Table



FAT table(FAT32) 은 디스크에 저장되며, block 들로 구성된다.

> FAT Entry 개수 = Block size / 4 bytes

> 그림에서 Block size = 16, 실제 크기는 헤더 파일에 정의됨

> 그림에서 Block 개수는 4, 실제 개수는 헤더 파일에 정의됨

fat.h 헤더파일이 제공될텐데
그 안에 블럭 사이즈가 정의가 되어있고
이걸 써서 과제를 하면 됨.

FAT entry

> 0 → 비어 있음을 의미함

> -1 → 마지막 블록을 나타냄

FatInit

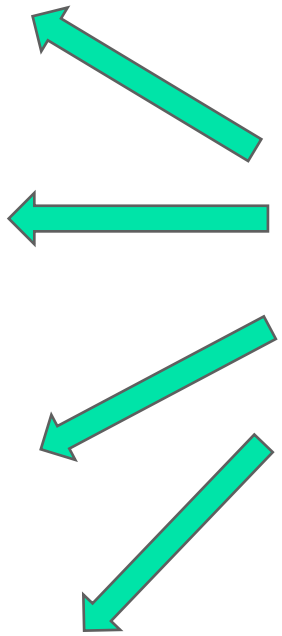
Index		
Block 0	0	0
	1	0
	2	0
	3	0
Block 1	4	0
	5	0
	6	0
	7	0
Block 2	8	0
	9	0
	10	0
	11	0
Block 3	12	0
	13	0
	14	0
	15	0

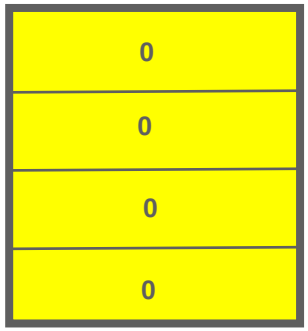
Void FatInit(void)

> FAT table을 0으로 채워서 초기화

FAT를 구성하는 모든 엔트리를 0으로 초기화 한다.

> 블록 크기의 메모리를 할당 받은 후 0으로 채우고 디스크로 저장하면 끝.





(1) Block 크기의 메모리 할당하고 0으로 채운다

0으로 채운 블럭을 만들고

(2) DevWriteBlock 함수를 통해 메모리를 Block 0부터 3까지 저장한다.

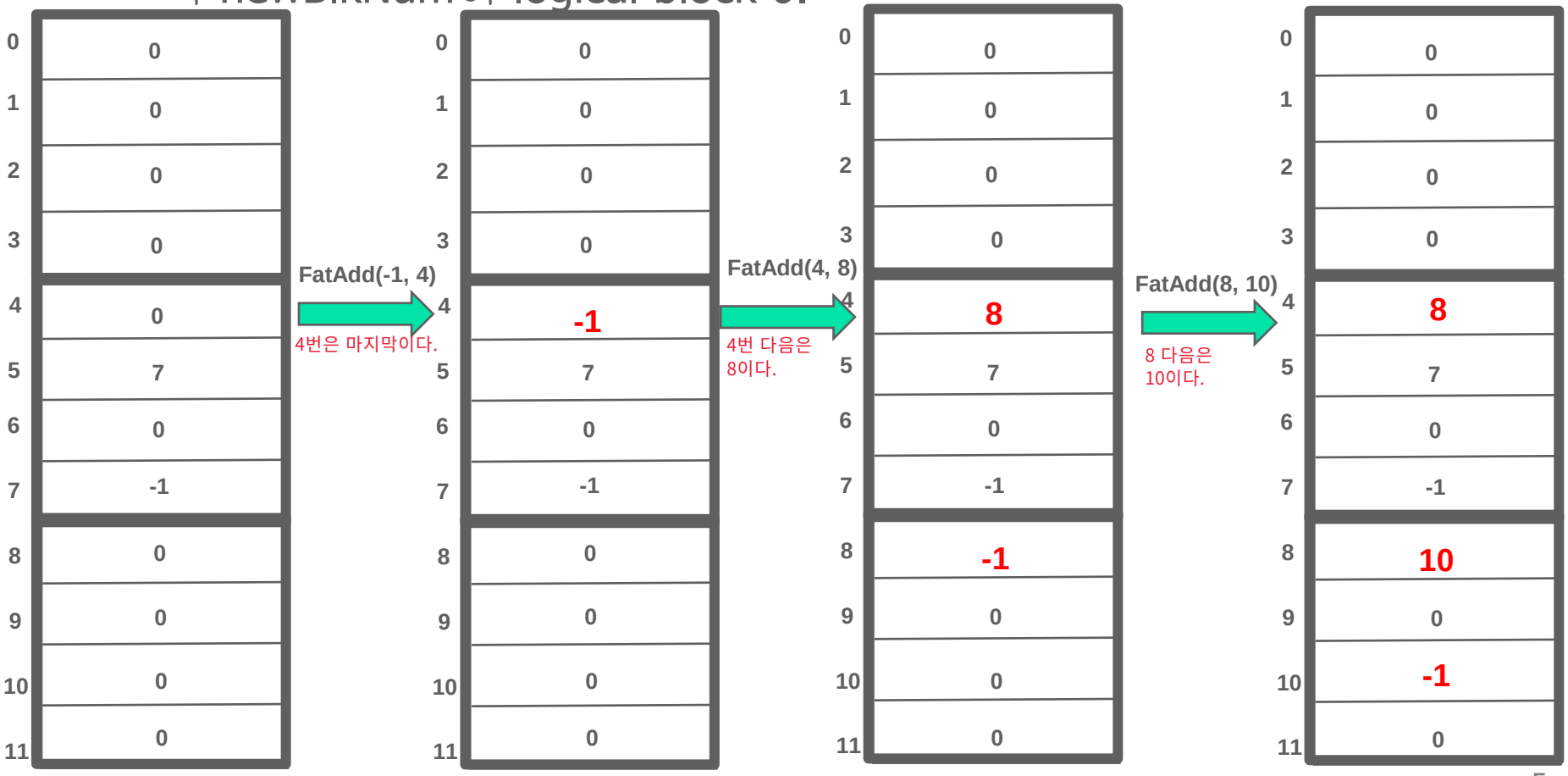
함수로 모든 블럭에 집어넣는다. 함수의 첫번째 인자는 블럭 번호를 말함.

FatAdd

Void FatAdd(int lastBlkNum, int newBlkNum)

> lastBlkNum의 다음 블록을 newBlkNum으로 지정함.

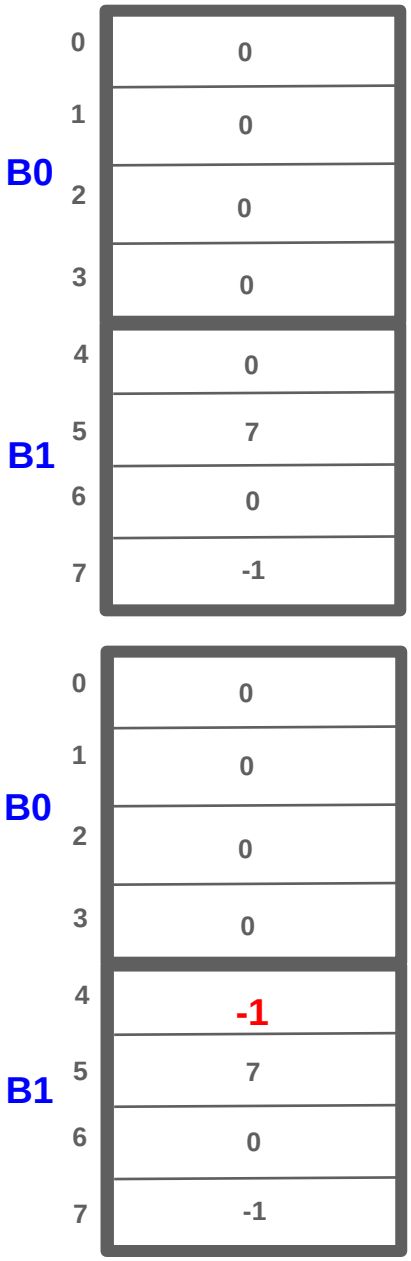
> lastBlkNum이 -1이면 빈 파일에 logical Block 0을 추가한다고 가정.
즉 newBlkNum이 logical block 0.



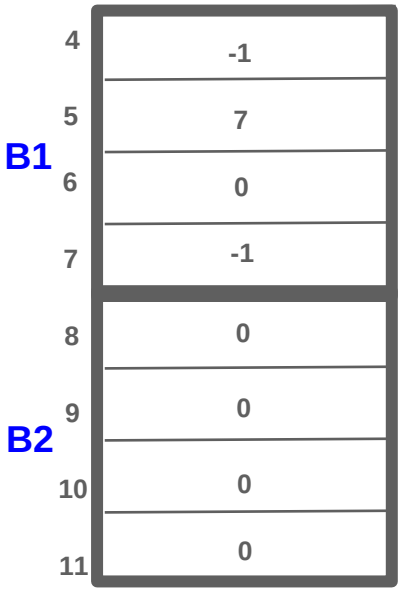
FatAdd(-1, 4)

> Index가 지정하는 FAT entry를 포함한 블록을 디스크에서 읽어야 한다.

> FAT entry 변경 후 다시 디스크로 저장해야 한다.

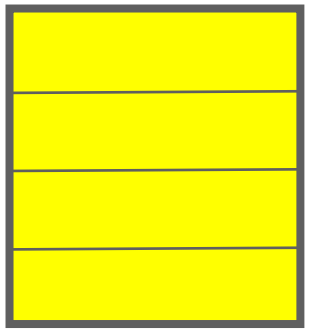


FatAdd (4, 8)

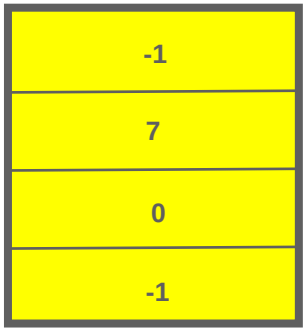


(2) Index 4가 있는 Block 번호를 계산한다. Block 1이다.

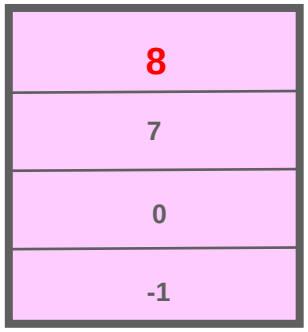
(3) Block 1을 디스크에서 메모리로 읽는다



(1) Block 크기의 메모리 할당



(3) 동작 (2) 후의 메모리 상태

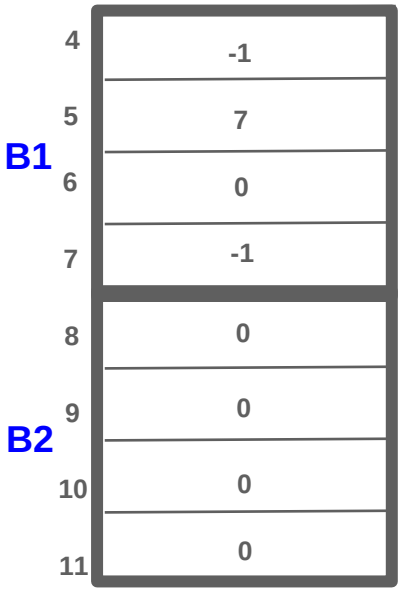


(4) Index 4의 entry에 8로 설정.
Block 4 다음 Block이 Block 8임을 의미

FatAdd(4, 8)

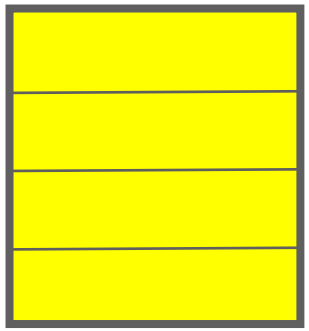
> lastBlockNum이 지정하는 FAT entry가 다른 블록에 있다면 해당 블록을 읽는다.

> 블록 읽기 후 마지막 블록임을 표시하기 위해 -1 저장

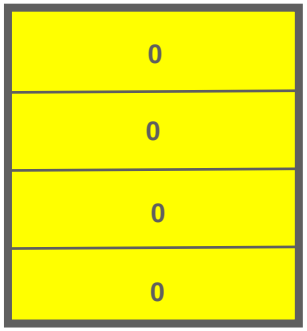


(6) Index 8가 있는 Block 번호를 계산한다. Block 2이다.

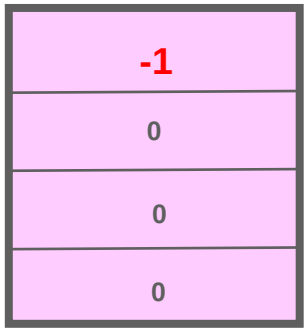
(7) Block 2을 디스크에서 읽는다



(5) Block 크기의 메모리 할당

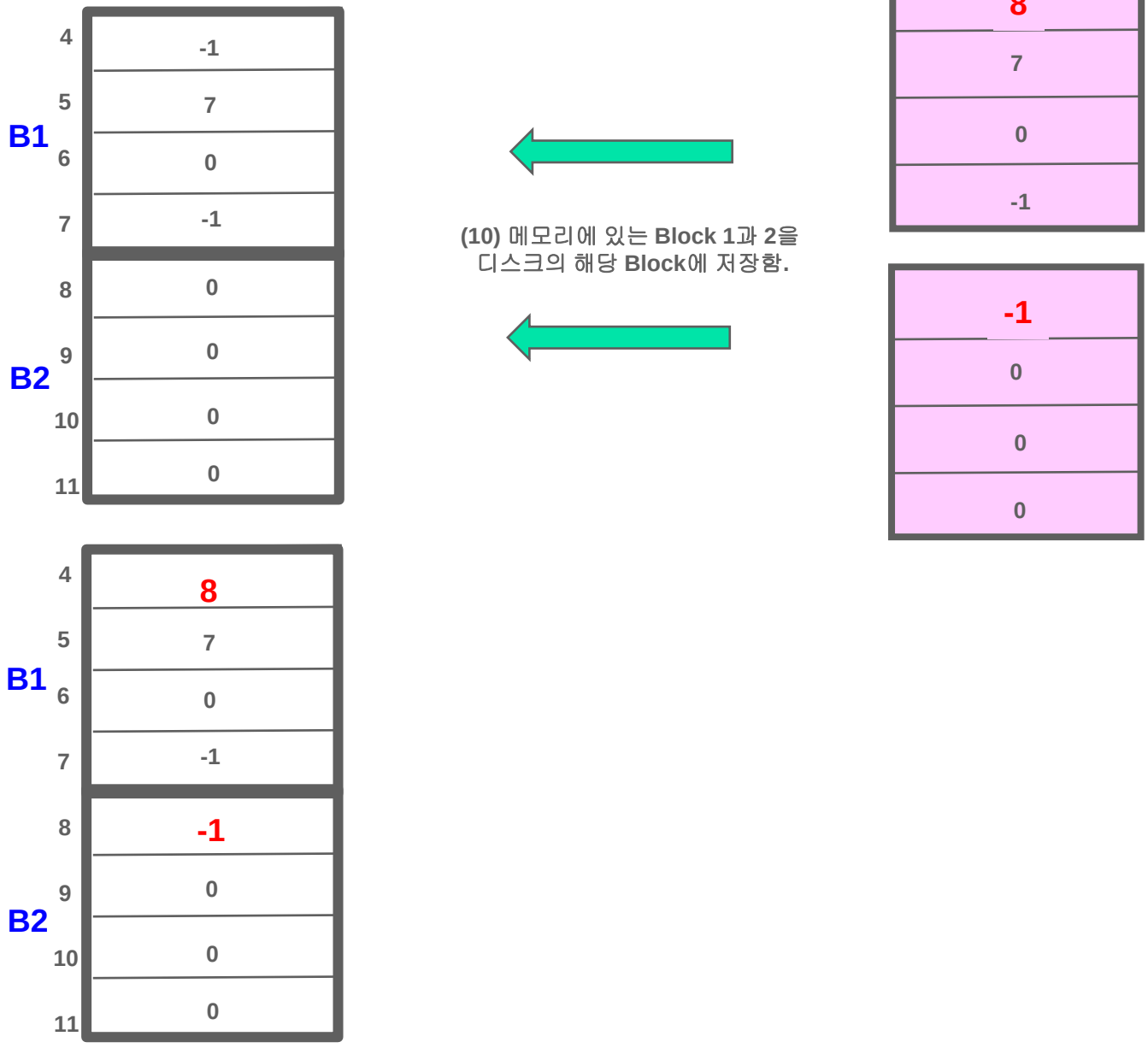


(8) 동작 (7) 후의 메모리 상태



(9) Index 8의 entry에 -1로 설정. Block 8가 마지막 블록임을 의미

FatAdd(4, 8)


4
5
6
7
B1
-1
7
0
-1
8
9
10
11
B2
0
0
0
0
(10) 메모리에 있는 Block 1과 2을
디스크의 해당 Block에 저장함.
8
7
0
-1
-1
0
0
0
4
5
6
7
B1
8
7
0
-1
8
9
10
11
B2
-1
0
0
0

FatGetBlockNum

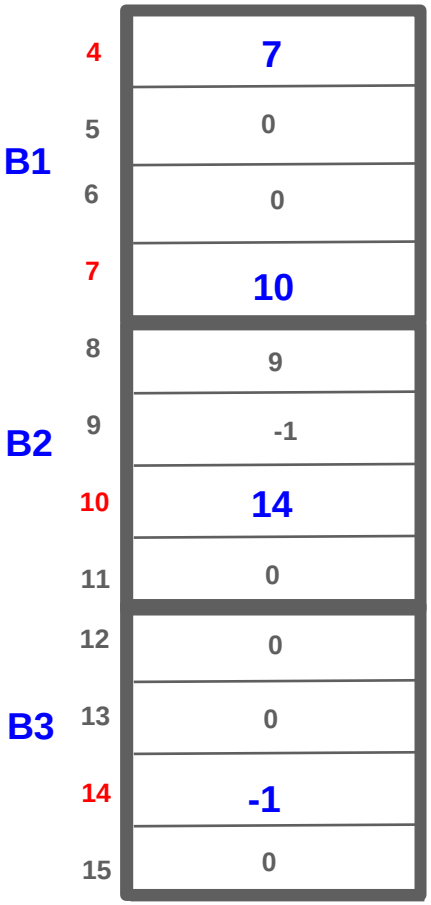
int FatGetBlockNum(int firstBlock, int LogicalBlkNum)

로지컬 블럭 넘버

> Logical block number로 physical block number를 획득

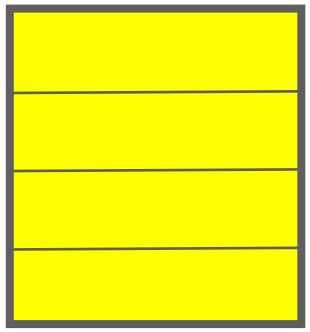
> firstBlock에서 LogicalBlkNum의 physical block number를 획득

FatGetBlockNum(4, 3)

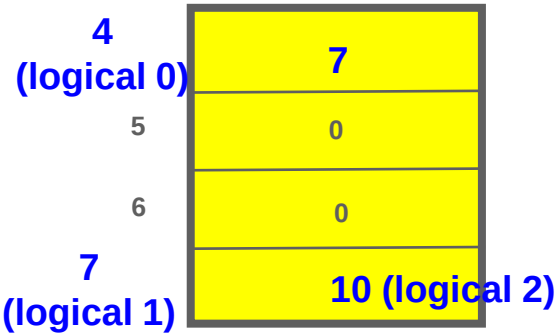


(2) Index 4이 있는 Block 번호를 계산한다. Block 1이다.

(3) Block 1을 디스크에서 읽는다

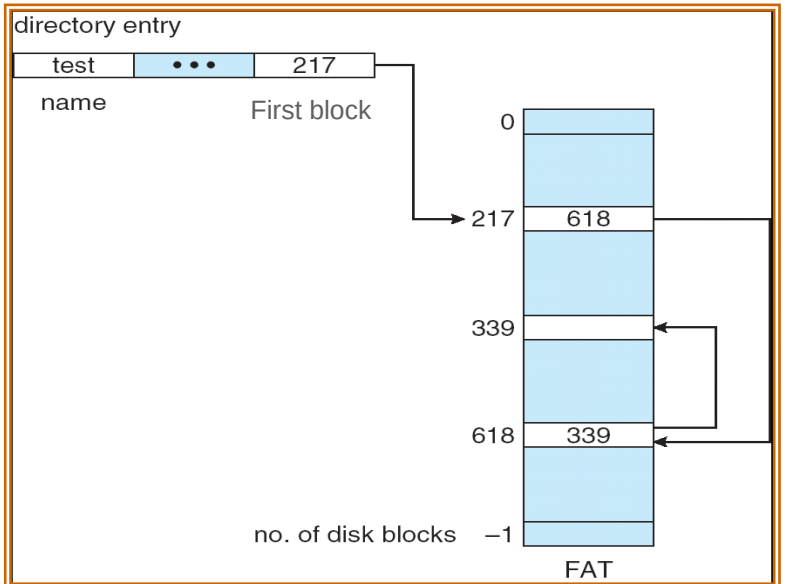


(1) Block 크기의 메모리 할당



(4) 동작 (3) 후의 메모리 상태

(5) Logical block 2는 Block 10이고, 다음 block 위치 확인하기 위해 index 10을 확인

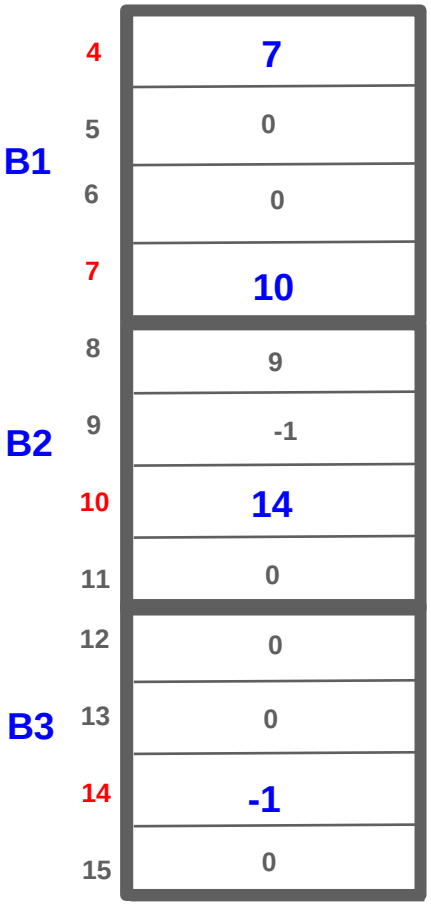


FatGetBlockNum

FatGetBlockNum(4, 3)

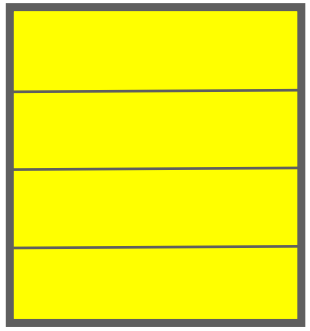
> Index 10의 FAT entry를 확인하기 위해 Block 1과 다른 Block 2를 읽음

> Index 10의 FAT entry에 logical block 3의 block number(14)를 획득함.

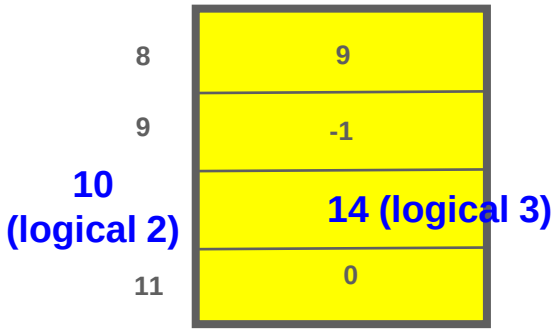


(7) Index 10이 있는 Block 번호를 계산한다. Block 2이다.

(8) Block 2을 디스크에서 읽는다



(6) Block 크기의 메모리 할당



(9) 동작(8) 후의 메모리 상태

(10) Index 10에서 logical block num 3의 Physical block num이 14임을 확인함

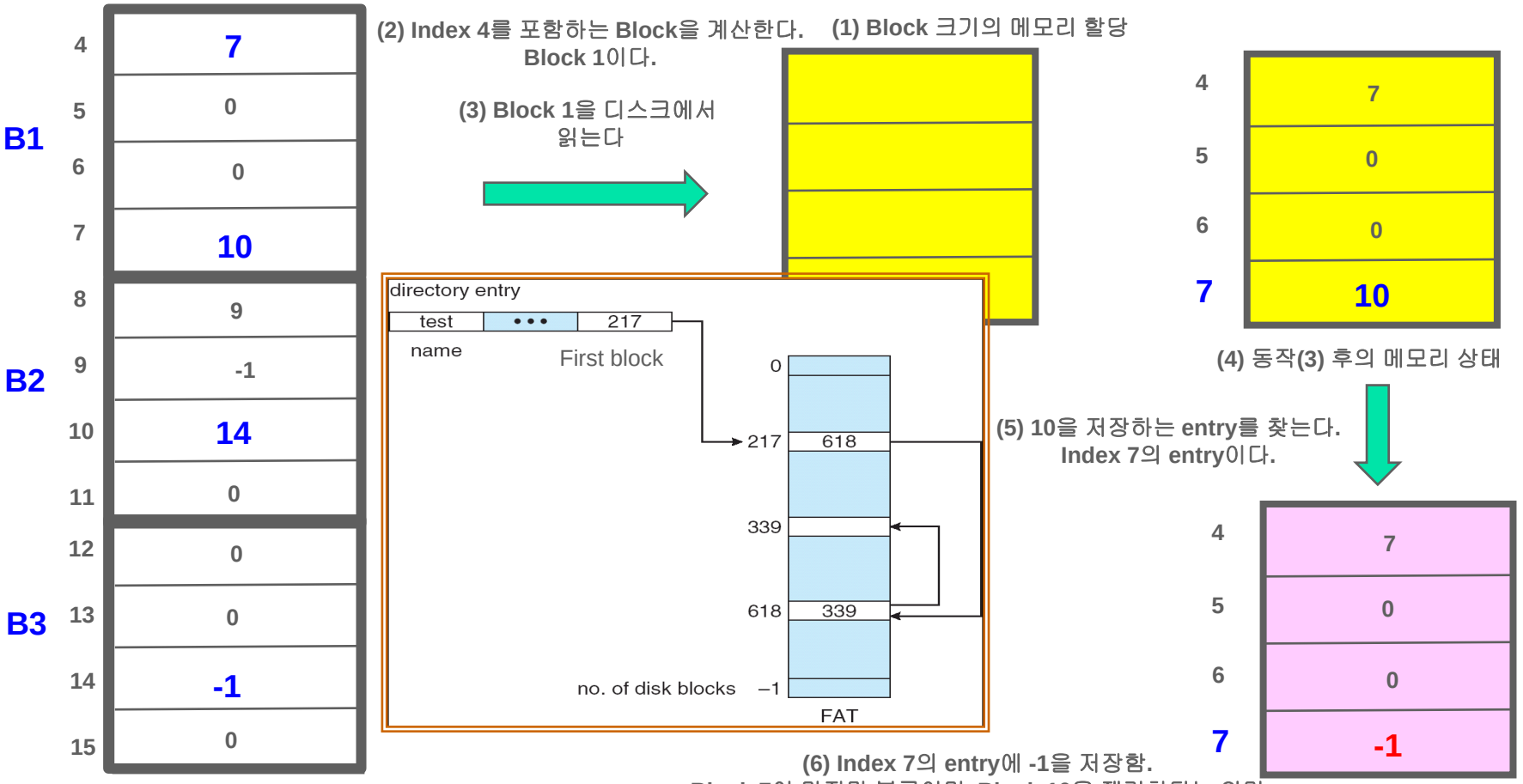
FatRemove

int FatRemove(int firstBlock, int startBlock)

> startBlock부터 마지막 블록까지 제거. firstBlock은 파일 시작을 지정

> Return value: 제거된 블록 개수

FatRemove(4, 10):Index 10의 entry를 찾은 후, 뒤따르는 entry들에 0 저장

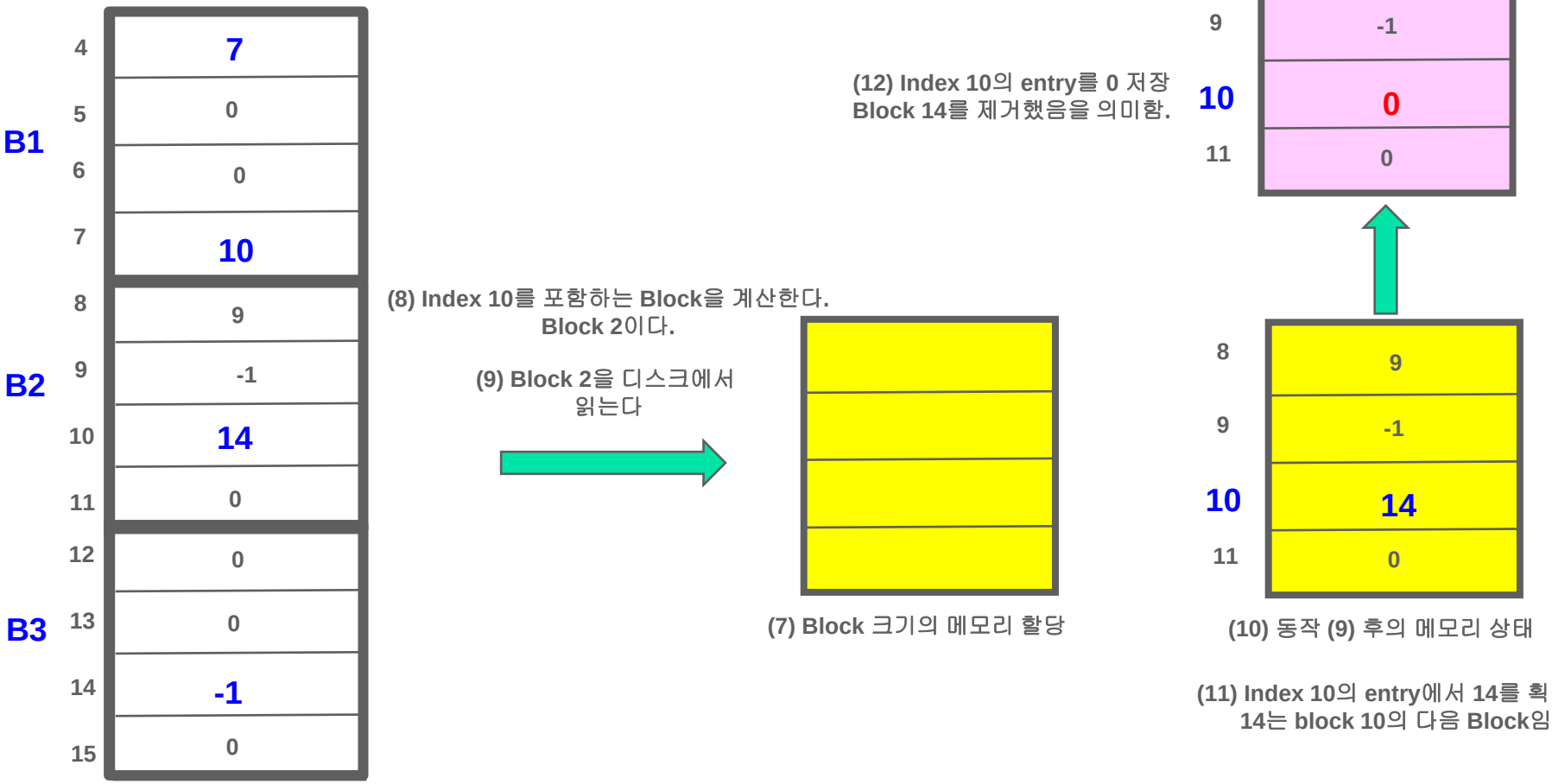


FatRemove

FatRemove(4, 10)

> index 10 entry를 entry를 0으로 채운다.

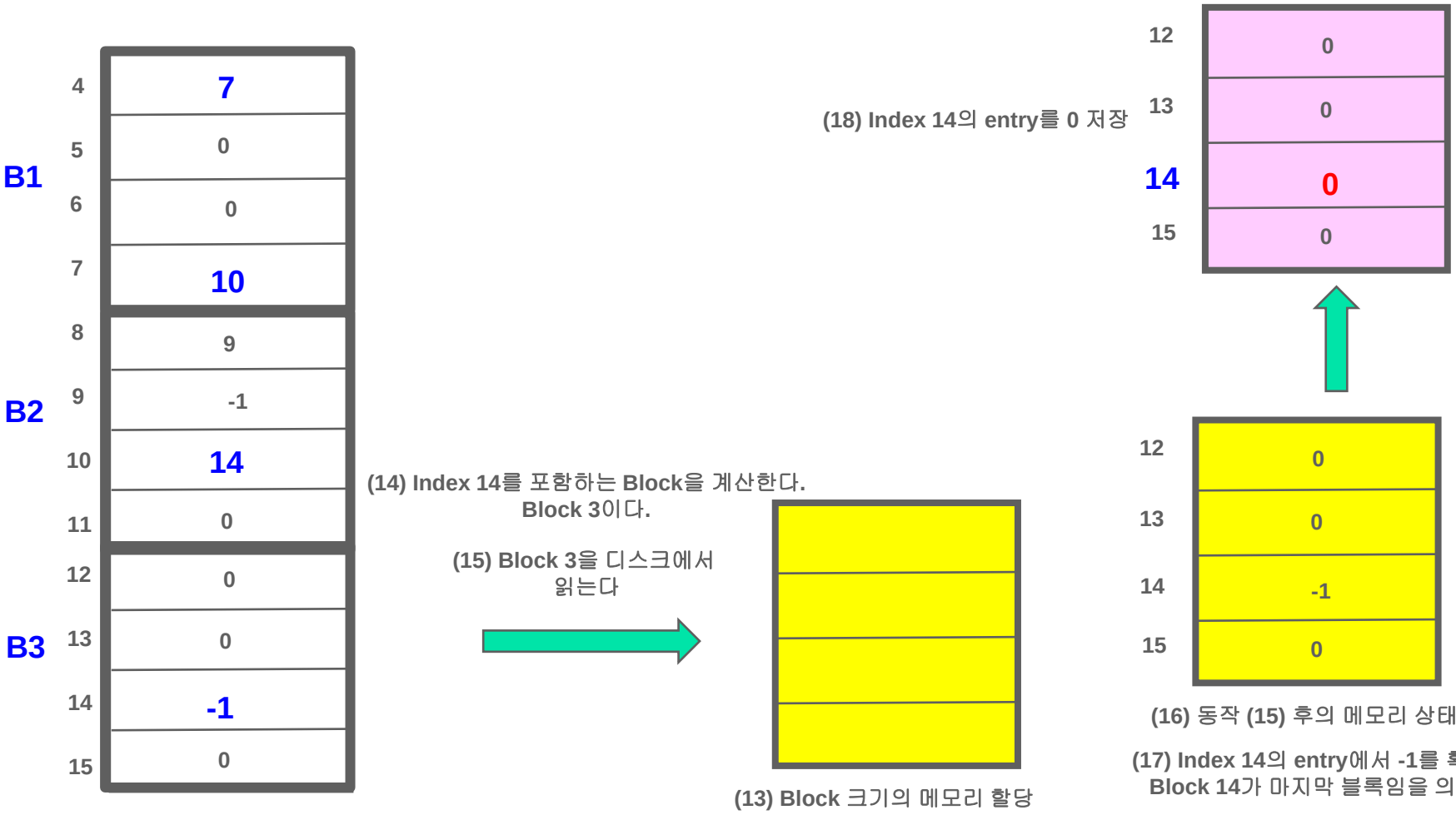
> 단, 다음 블록을 확인하기 위해 entry의 값(14)을 획득한다.



FatRemove

FatRemove(4, 10)

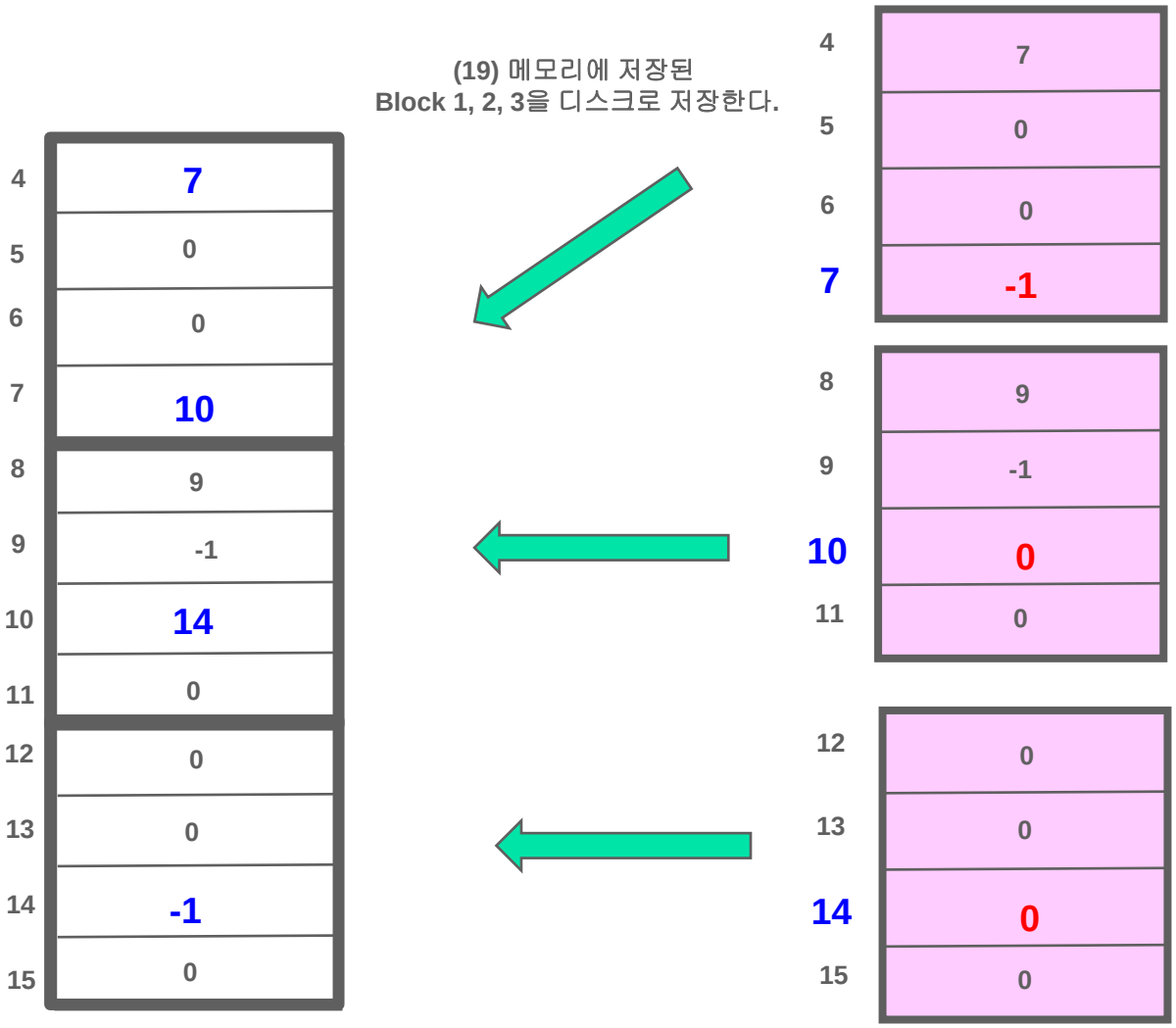
> Index 14의 entry를 찾아서 0으로 채운다.



FatRemove

FatRemove(4, 10)

> 메모리에서 변경된 블록들을 디스크로 저장한다.



유의 사항

- 구현 해야할 함수들
  - FatInit, FatAdd, FatRemove, FatGetBlockNumber

- 구현 해야할 파일들
  - fat.c 구현. fat.c에 임의의 헤더 파일(예, temp.h 등)을 include 가능
  - fat.h, disk.c, disk.h 제공됨. 수정 불가(수정하면 0점 처리)
  - fat.h에는 구현 해야할 function prototype 선언함.
  - main.c에서 fat.h를 include해서 testcase가 제공되기 전에 각자 테스트함
  - 향후 testcase를 포함하는 main.c을 제공할 계획

main.c

```
#include <stdio.h>
#include "fat.h"

main(void)
{
  int blkno;
  FatInit();
  FatAdd(-1,4);
  FatAdd(4, 6);
  FatAdd(6, 10);
  blkno = FatGetBlockNum(4, 1);
  ...
}
```

fat.c

```
#include "fat.h"

Void FatInit(void)
{
  int pMem = malloc(...);
  ...
}
Void FatAdd(int lastBlkNum, int
newBlkNum)
{
  ... // implement this func.
}
...
```