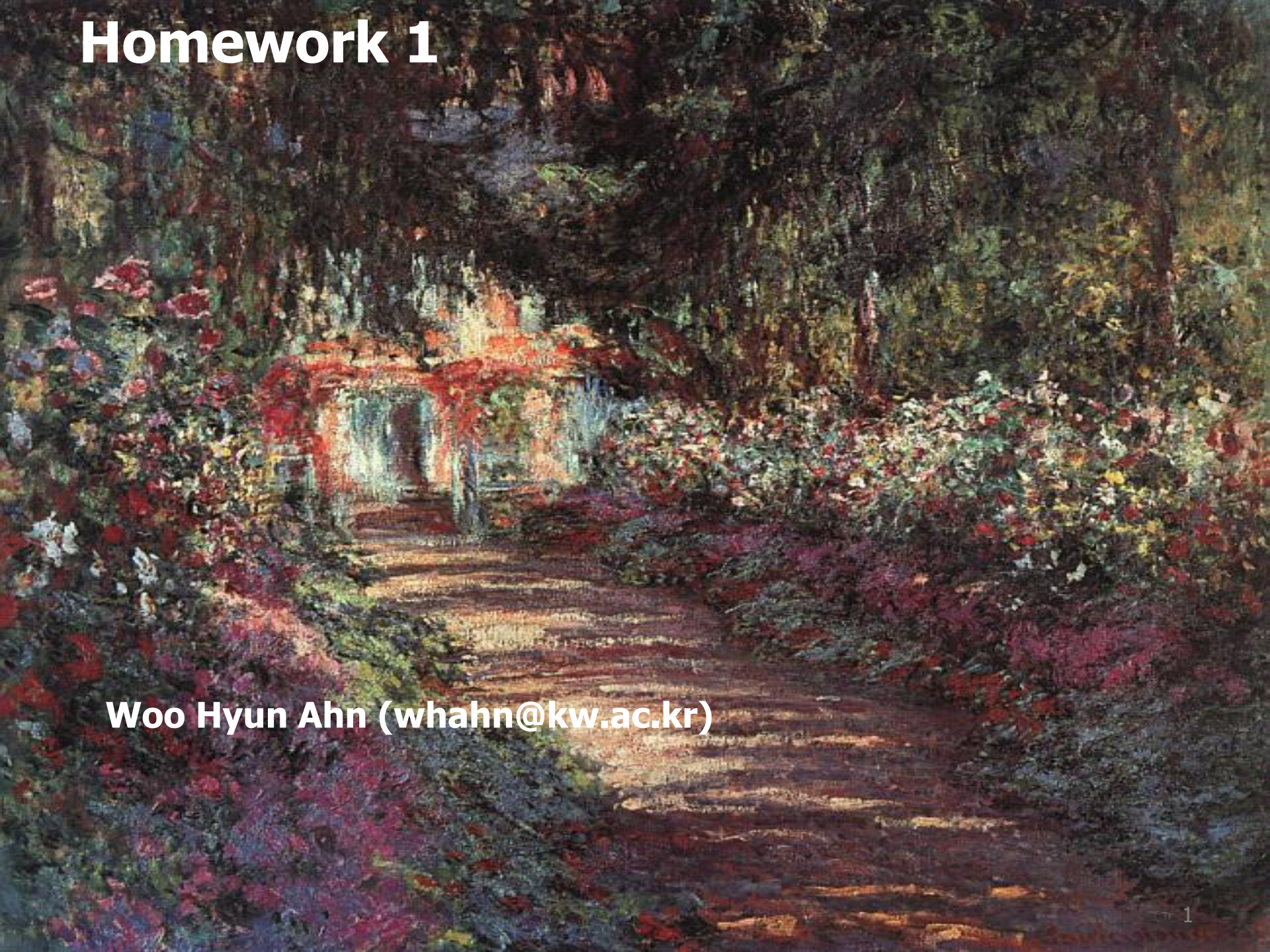
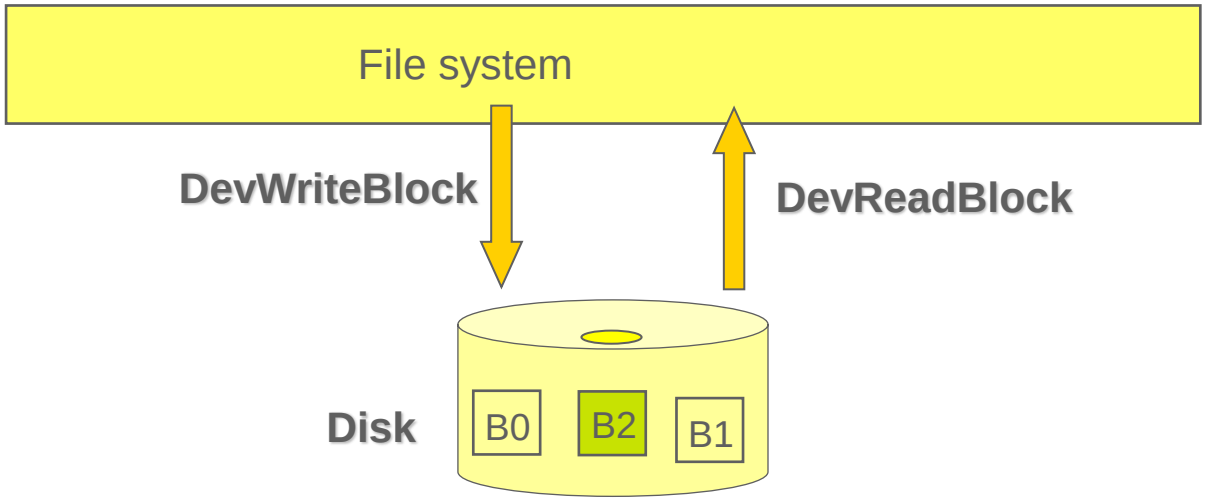
# Homework 1



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

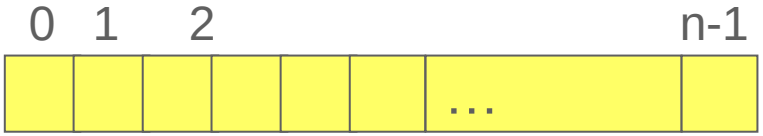
# The Relation of File System & Disk




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

# Free Space Management – Bit Map

- Bit map
  - Free space is represented as a bit map or bit vector.

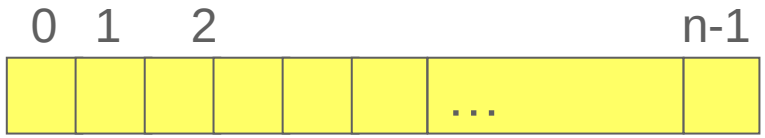



$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block } i \text{ free} \\ 1 \Rightarrow \text{block } i \text{ allocated} \end{cases}$$

- Example
  - Blocks 2,3,4,5,8,9,10,12,13,17,25,26, and 27 are allocated.
  - Bit map: 00111100111111000110000001111100000
- Implementation
  - Bit map is cached in memory to reduce disk accesses.
- Advantage
  - Simplicity
  - Performance – many computers supply bit-manipulation instructions that can be used effectively for that purpose.

# Free Space Management – Integer Vector

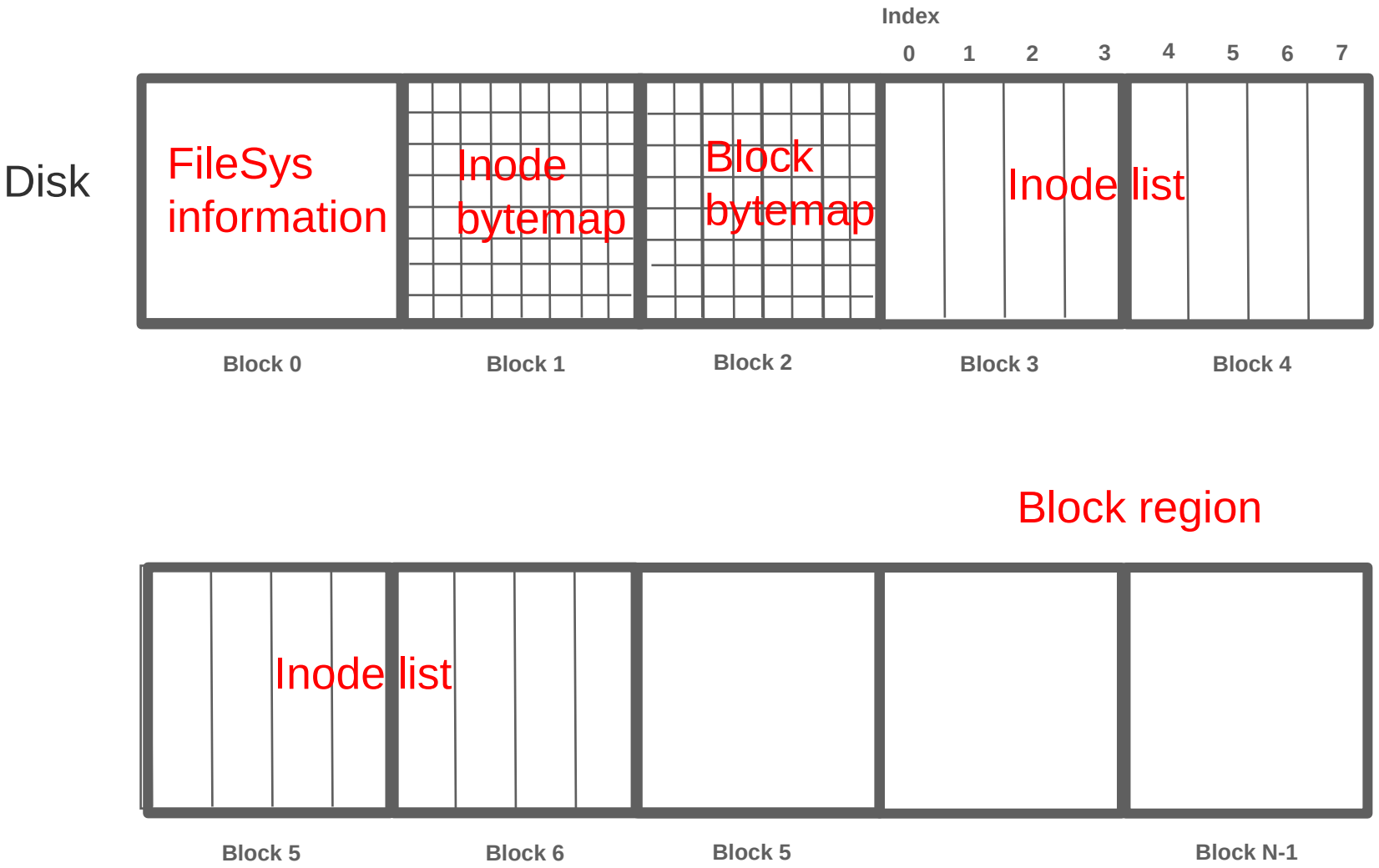
- Integer vector
  - 빈 공간을 표시하기 위해 bitmap 대신 byte vector, 즉 bytemap사용함.
  - 취지: bit map 연산이 어려워서 좀더 쉬운 bytemap 로 구현함.



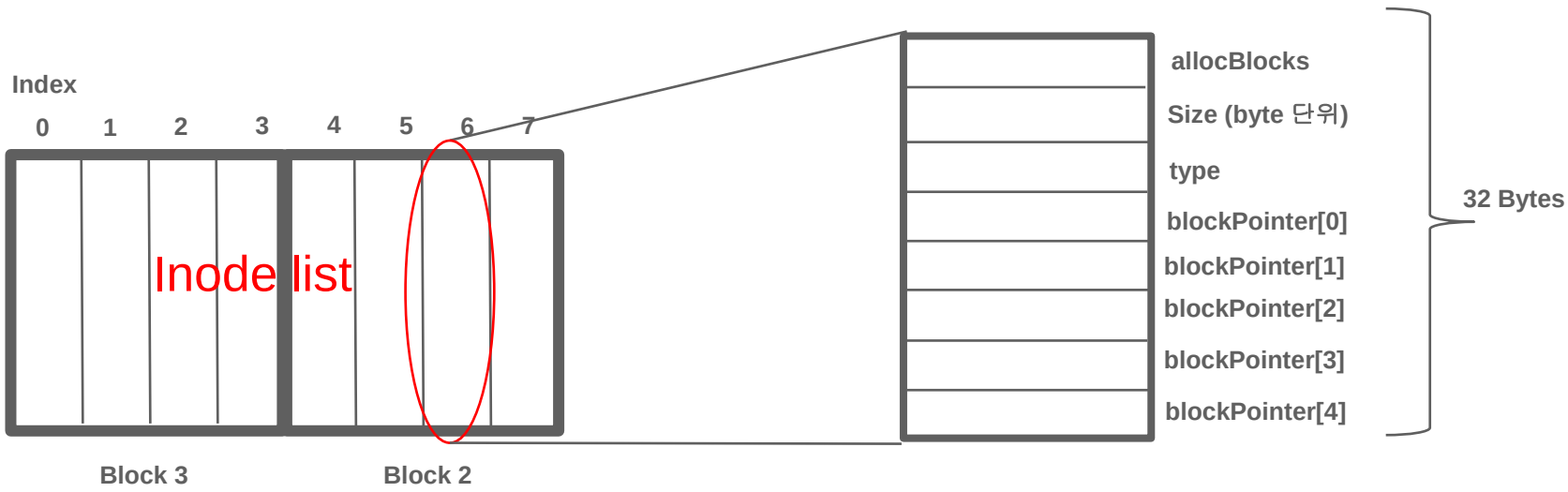

byte[*i*] = 0 ⇒ block *i* free
1 ⇒ block *i* allocated

- 예제
  - Blocks 2,3,4,5,8,9,10,12,13,17,25,26, and 27 할당됨.
  - byte : 00111100111111000110000001111000000

# Basic Disk Layout



# Inode list




Inode

> 파일을 대표하는 객체, 즉 파일 정보를 저장하는 객체

> 파일에 포함되는 블록의 위치를 저장한다

> 크기: 32 Bytes

Inode list

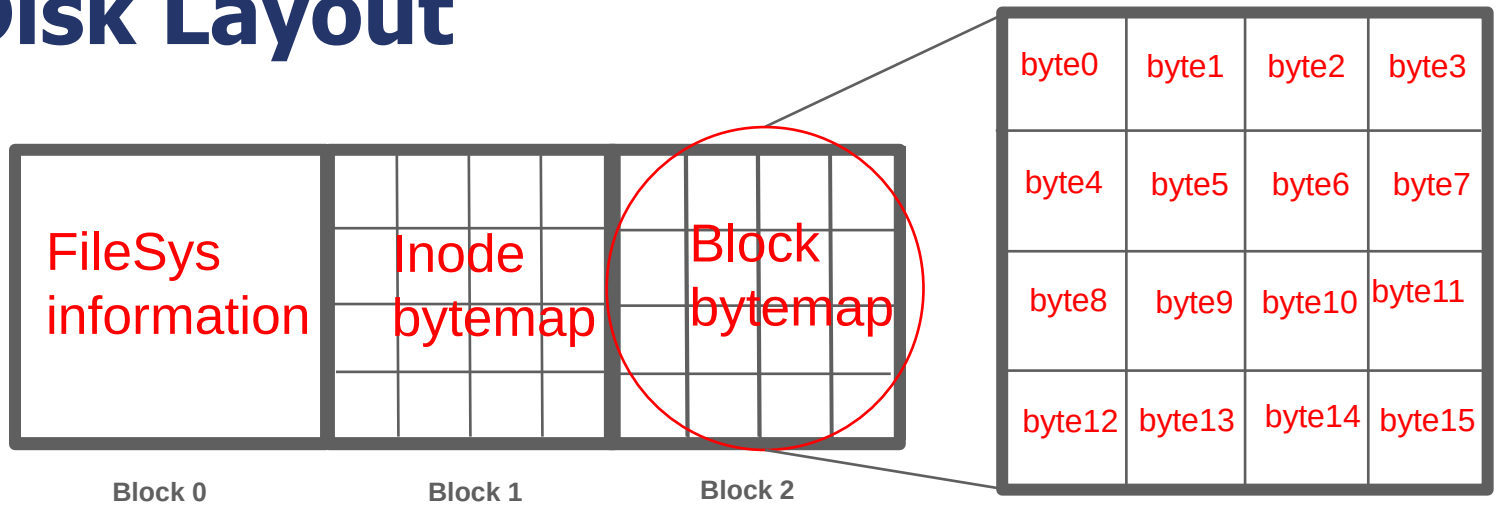
> Inode들을 관리하는 목록

> Block 당 Inode 개수 = Block size / 32 bytes = 16, block size = 512

> 전체 inode 개수 = Block 당 Inode 개수 * Block 개수

> 예로, 16(Block 당 inode 개수)*4(Block 개수) = 64개

# Basic Disk Layout




FileSys information

> 파일시스템의 정보를 관리하는 Block

> 생성된 Inode의 개수, 전체 디스크에서 할당된 Data block의 개수

Inode bytemap

> inode 상태(할당 또는 비할당)에 대한 정보를 기록

> 예로, inode 0이 할당되면, byte 0은 1로 설정. inode 10이 할당되면 Int 10이 1로 설정

Block bytemap

> block 상태(할당 또는 비할당)에 대한 정보를 기록

> 예로, block 20이 할당되면, byte 20은 1로 설정됨

# FileSysInit

Void FileSysInit(void)

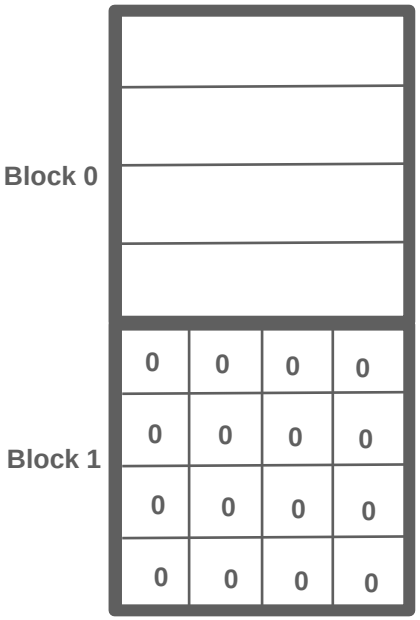
> FileSys info, inode bytemap, block bytemap, inode list를 0으로 채워서 초기화

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

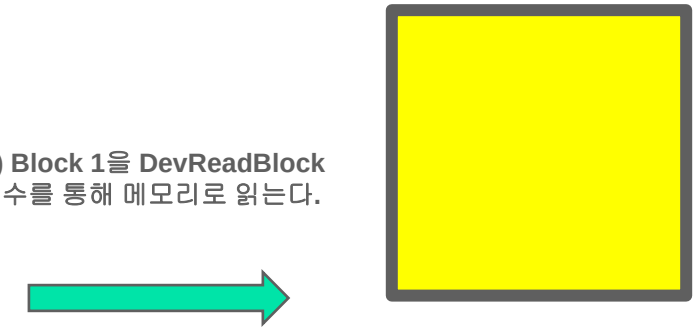
# SetInodeBytemap(2)

> Inode bytemap에서 byte 2를 1로 설정하고, 다시 디스크로 저장한다.

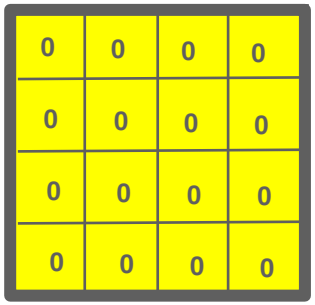
> Inode bytemap은 block 1에 저장됨



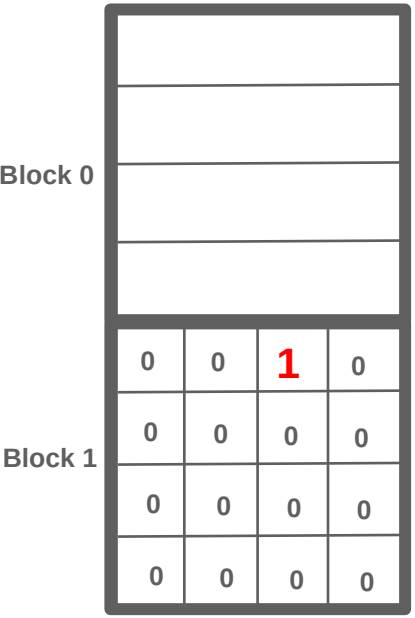

(2) Block 1을 DevReadBlock 함수를 통해 메모리로 읽는다.



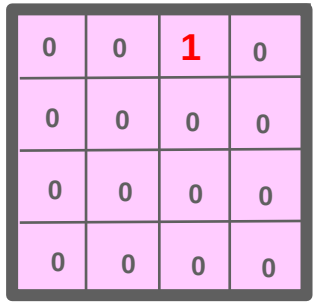
(1) Block 크기의 메모리 할당




(3) 동작 (2) 후의 메모리 상태




(6) Block 1에 DevWriteBlock 함수를 통해 저장함.

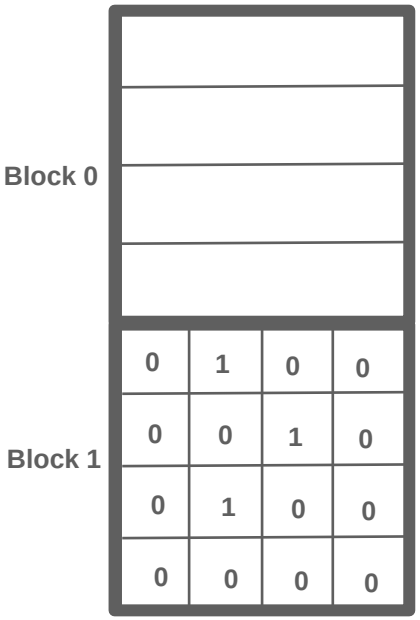



(4) byte 2를 1로 설정함

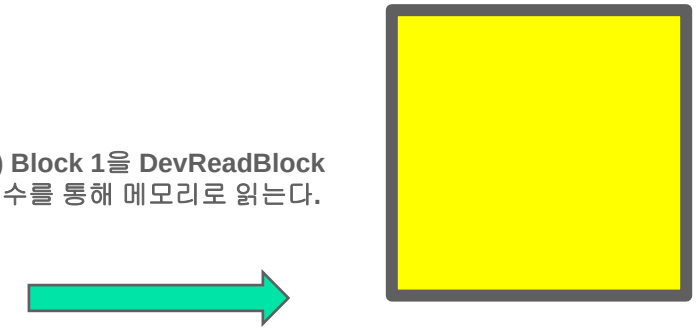
# ResetInodebytemap(6)

> Inode bytemap에서 byte 6을 0로 설정하고, 다시 디스크로 저장한다.

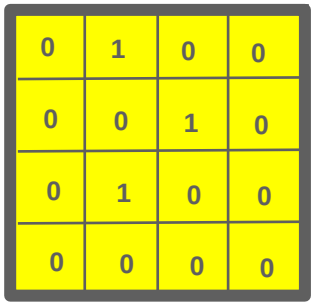
> Inode bytemap은 block 1에 저장됨



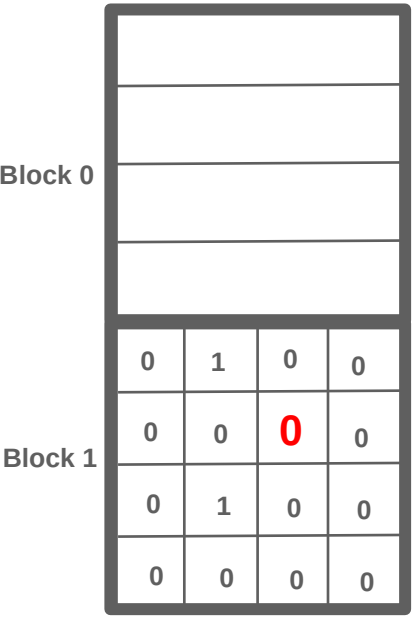

(2) Block 1을 DevReadBlock 함수를 통해 메모리로 읽는다.



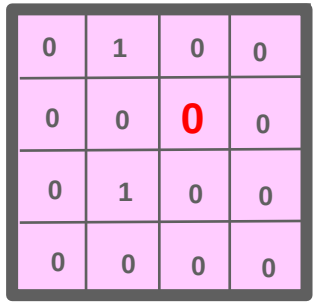
(1) Block 크기의 메모리 할당




(3) 동작 (2) 후의 메모리 상태




(6) Block 1에 DevWriteBlock 함수를 통해 저장함.




(4) Int 2를 1로 설정함

# SetBlockBytemap(3)

> Block bmtemap에서 byte 3를 1로 설정하고, 다시 디스크로 저장한다.

> Block bytemap은 block 2에 저장됨

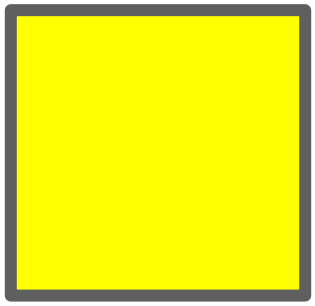
Block 1

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Block 2

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

(2) Block 2을 DevReadBlock 함수를 통해 메모리로 읽는다.



(1) Block 크기의 메모리 할당

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

(3) 동작 (2) 후의 메모리 상태

Block 1

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Block 2

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

(6) Block 2에 DevWriteBlock 함수를 통해 저장함.

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

(4) byte 3를 1로 설정함

# SetBlockBytemap(6)

> Block bytemap에서 byte 6를 1로 설정하고, 다시 디스크로 저장한다.

- Block bytemap은 block 2에 저장됨

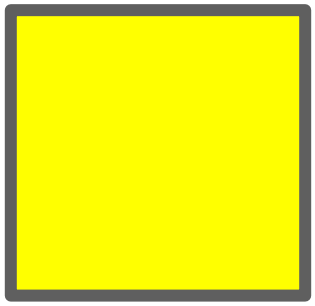
Block 1

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Block 2

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

(2) Block 2을 DevReadBlock 함수를 통해 메모리로 읽는다.



(1) Block 크기의 메모리 할당

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

(3) 동작 (2) 후의 메모리 상태

Block 1

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Block 2

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | **1** | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

(6) Block 2에 DevWriteBlock 함수를 통해 저장함.

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | **1** | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

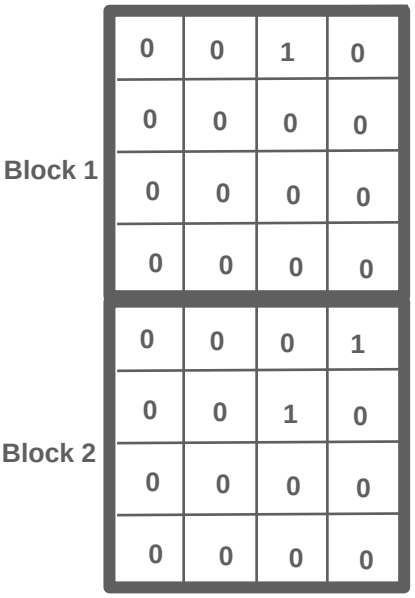
(4) byte 6를 1로 설정함

**Map 변경하기 전에, Block을 디스크에서 읽는 이유?**
**>** 디스크에 있는 원본 **Block** 없이 설정해서 저장하면 원본 **Block**의 데이터가 손실된다.

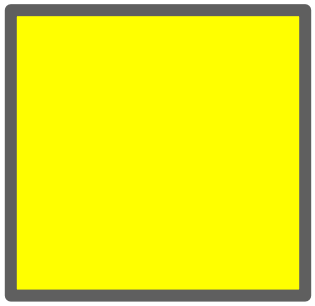
# ResetBlockBytemap(6)

> Block bytemap에서 byte 6를 0로 설정하고, 다시 디스크로 저장한다.

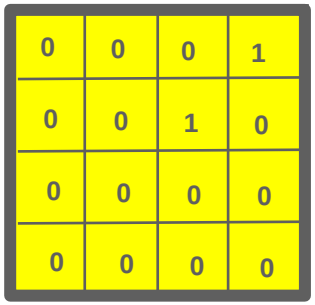
> Block bytemap은 block 2에 저장됨



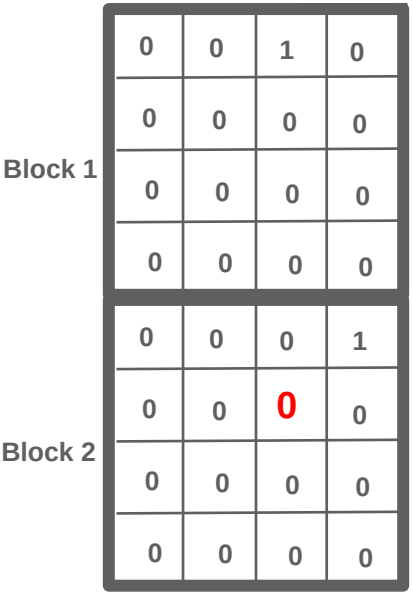

(2) Block 2을 DevReadBlock 함수를 통해 메모리로 읽는다.



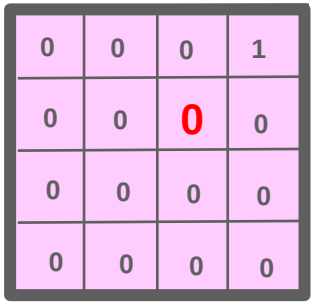
(1) Block 크기의 메모리 할당




(3) 동작 (2) 후의 메모리 상태




(6) Block 2에 DevWriteBlock 함수를 통해 저장함.

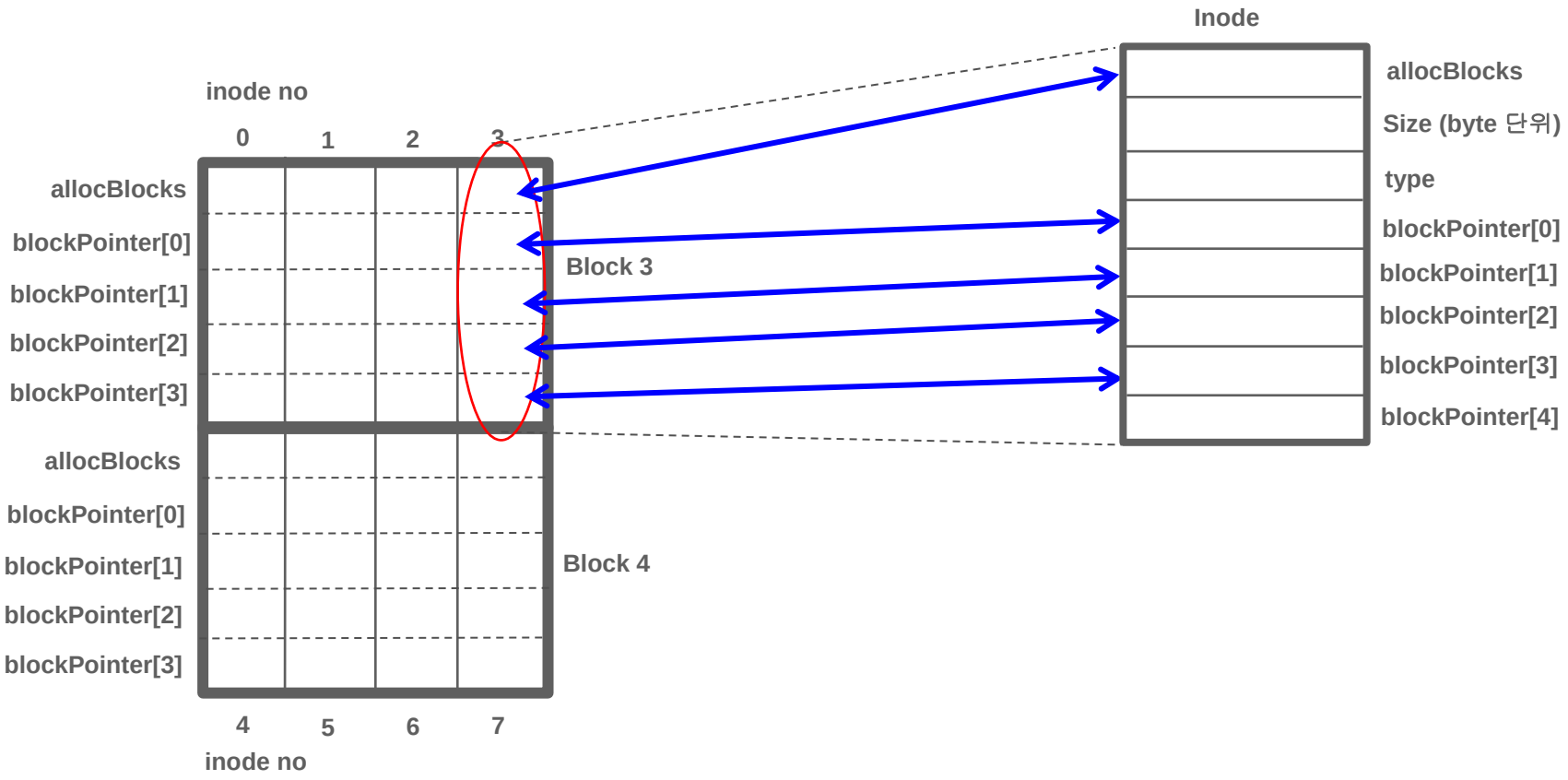



(4) byte 6를 0로 설정함

# Assumption of Inode list

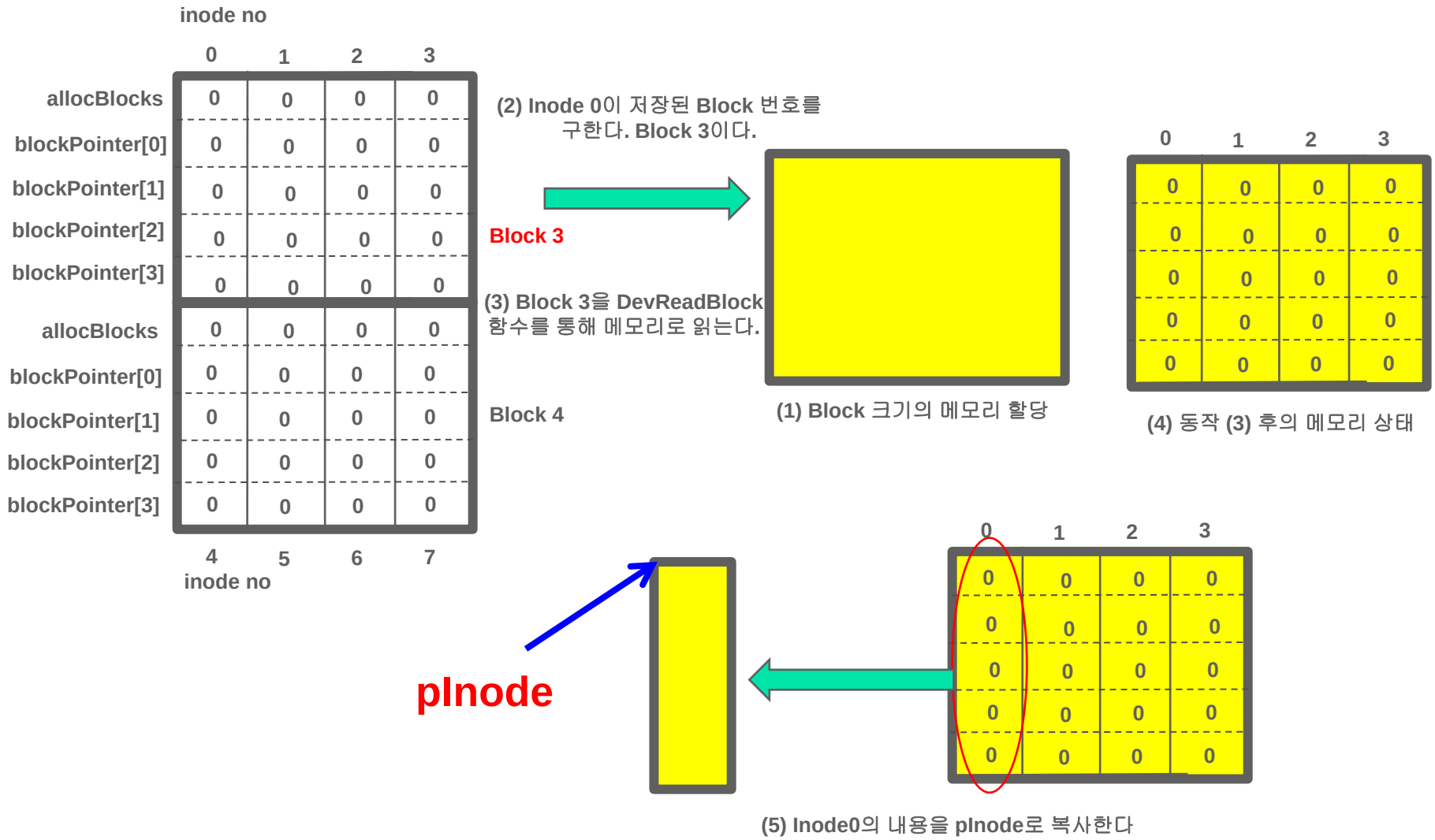
> Block 당 4개의 inode가 있다고 가정한다.
→ 단, 과제에서는 Block 당 16개 inode가 저장됨

> 왼쪽 그림에서 size, type, blockPointer[4]를 표시하지 않았음. 하지만 구현 시 고려해야함



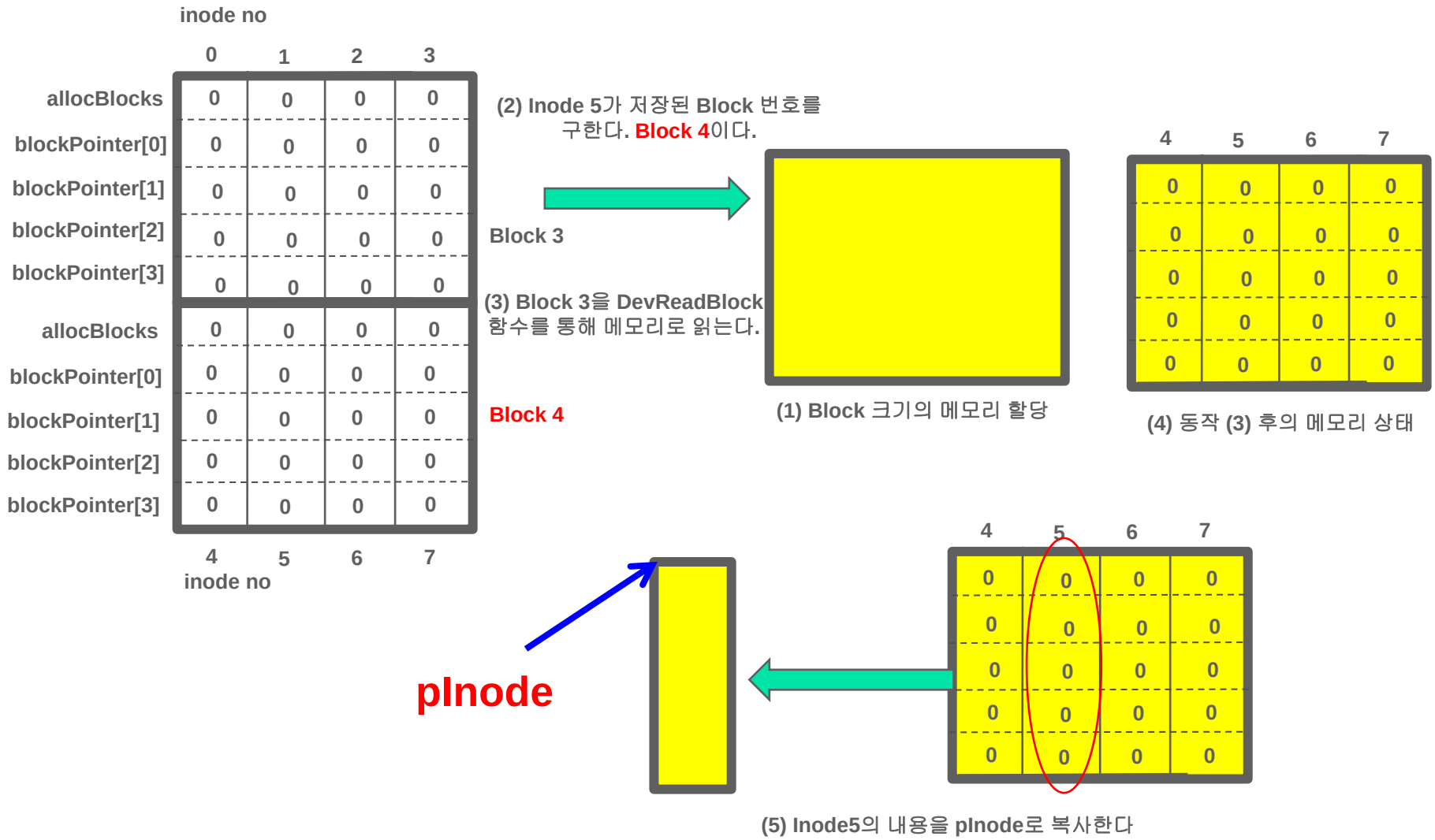
# GetInode(0, pInode)

> Inode 0를 디스크에서 읽어서 pInode가 지정하는 메모리 공간으로 저장함



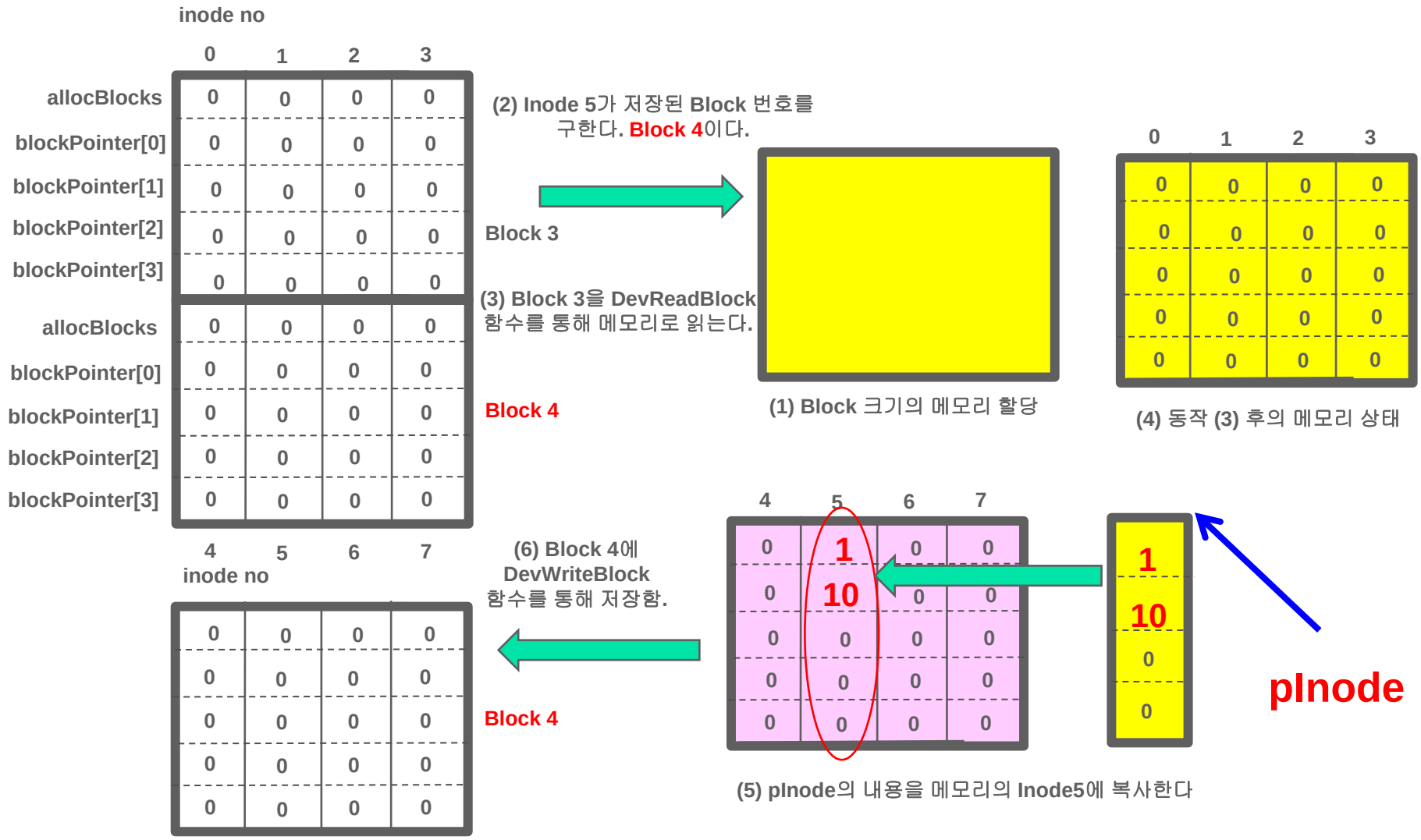
# GetInode(5, pInode)

> Inode 5를 디스크에서 읽어서 pInode로 저장함



# PutInode(5, pInode)

> pInode에 지정하는 메모리 공간에 저장된 Inode 5의 내용을 디스크에 저장한다.

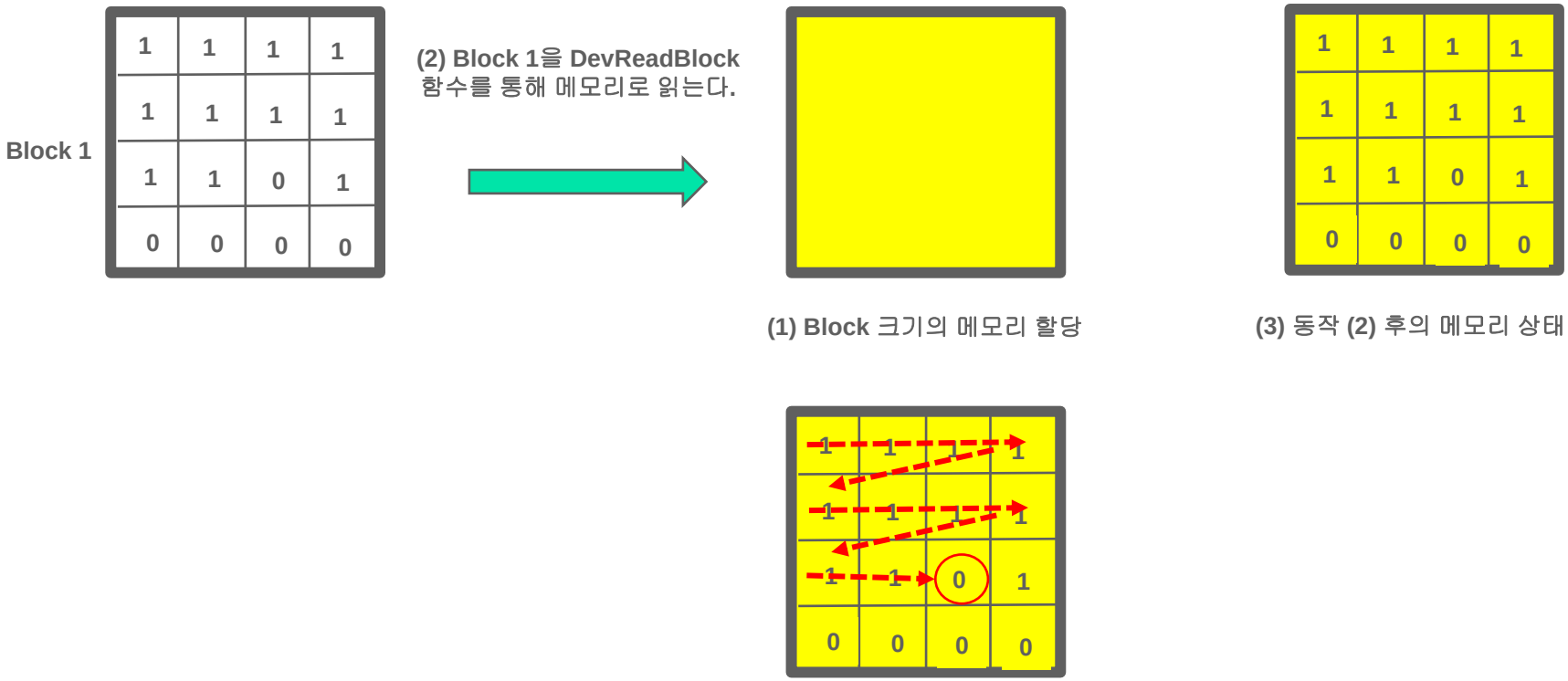


# GetFreeInodeNum

int GetFreeInodeNum(void)

> 할당되지 않은 inode 번호를 획득한다.

> Inode bytemap (block1)에서 First fit searching 방법 사용함

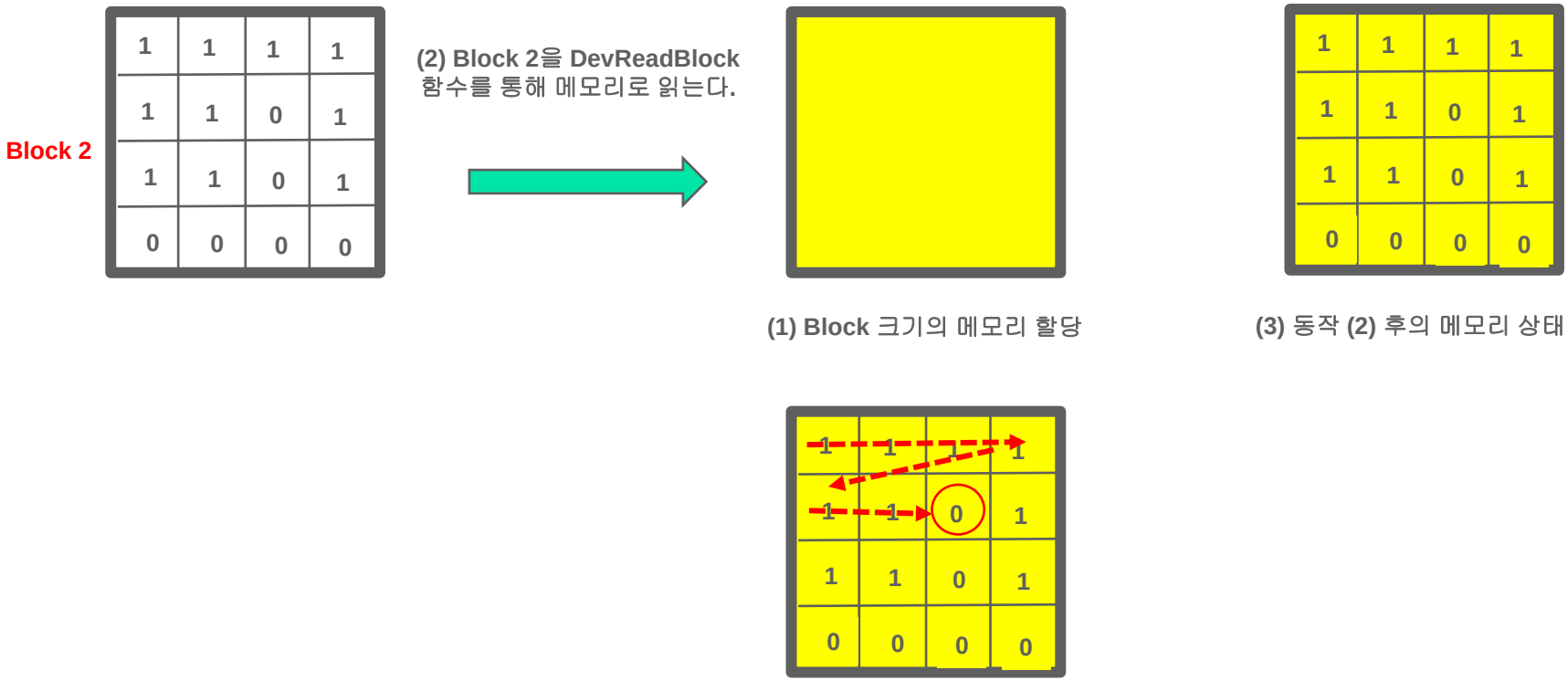


# GetFreeBlockNum

int GetFreeBlockNum(void)

> 할당되지 않은 Block 번호를 획득한다.

> Block bytemap (block2)에서 First fit searching 방법 사용함



# Implementation Scope

- FileSysInit, SetInodeMap, ResetInodeMap,SetBlockMap, ResetBlockMap, PutInode, GetInode, GetFreeInodeNum, GetFreeBlockNum

main.c

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

Main(void)
{
  int blknum, inodenum;
  Inode inode, *pInode;
  FileSysInit();
  inodenum = GetFreeInodeNum();

  SetInodeBytemap(inodenum);
  blknum = GetFreeBlockNum();
  SetBlockBytemap(blknum);
  inode.size = 0;
  inode.allocBlks = 0;
  ...
  PutInode(inodenum, &inode);
  pInode = malloc(sizeof(Inode));
  GetInode(inodenum, pInode);
}
```

fs.c

```
#include "fs.h"

Void SetInodeBytemap(int inodenum)
{
  int pMem = malloc(...);
  ...
}
Void PutInode(int inodenum, Inode*
pInode)
{
  ... // implement this func.
}
...
```