GEBZE TECHNICAL UNIVERSITY

CSE222 OPERATING SYSTEMS

HOMEWORK 3 REPORT

225008003102

SÜHA BERK KUKUK

Part 1 && Part 2:

```
struct superBlock
{
    char diskName[20];
    uint32_t diskSize;
    float blockSize;
    uint32_t numberOfEntry;
    uint32_t numberOfBlock;
    uint32_t totalByte;
    uint32_t bootSectorPosition;
    uint32_t fatTablePosition;
    uint32_t rootDirPosition;
    uint32_t dataStartPosition;

};

struct  entryDir
{
    char fileName[8]; //8 byte
    char extension[3]; // 3 byte
    uint8_t attributes; // 1 byte
    char reserved[10]; //10 byte
    uint16_t time; // 2 byte
    uint16_t date; // 2 byte
    uint16_t firstBlockNumber; // 2 byte
    uint32_t fileSize; // 4 byte
};
```

In this part superblock represents my boot sector. This structure contains information about the file system such as Bytes per sector, Sectors per cluster, Number of reserved sectors. I will keep it simple. I will put information(superblock) according to block size as in figure.

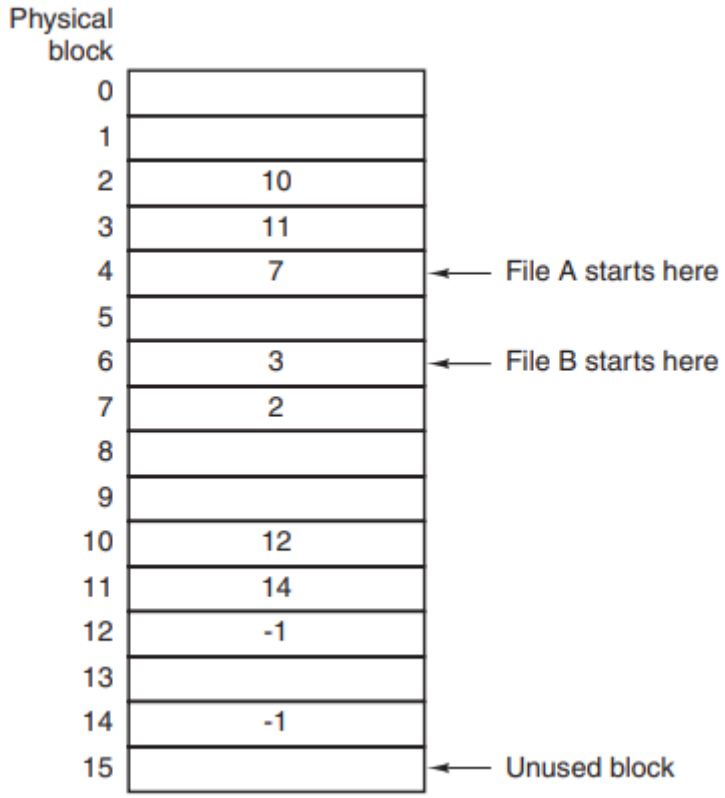
Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

Figure 4-31. Maximum partition size for different block sizes. The empty boxes represent forbidden combinations.

‘entryDir’ structure provides a structure to store information about a file or directory in a file system directory entry. The specific implementation and interpretation of the fields may vary depending on the file system used.

File Allocation Table (FAT):

I implemented FAT using array and it is like as figure below:



In the code:

```
int* fat[] = nullptr;
```

define like this.

Creta for file System. I write class for this purpose. You can see implementation in the below:

```
class fileSystem
{
    public:
        void createFile();
        int openFile();
        void initSystem();
        void printInfo();
        fileSystem(int blockSize, char *fileName);

    private:
```

```
        int  blockSize;
        char *fileName;
        superBlock disk;
};
```

The ‘fileSystem’ class encapsulates the functionalities and data needed to create, open, and manage files within a file system. It also provides methods to initialize the file system and retrieve information about it.

How we can run the code:

```
sbk@ubuntu-22:/media/sf_Vmprojects/HW3$ c++ -o out  makeFileSystem.cpp
sbk@ubuntu-22:/media/sf_Vmprojects/HW3$ ./out 4 mySystem.dat
```

Example:

```
        ****Virtual Disk Has Created!****
-----------------------------------------------------
Disk Name: mySystem.dat
Disk Size: 32MB
Block Size: 4KB
Number of Block: 4096
Boot Sector Start Address: 0
FAT Table Start Address: 57
Root Directory Start Address: 73784
Data Region Start Address: 73817
-----------------------------------------------------
```