SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECT312 – EMBEDDED SYSTEM DESIGN

III YEAR/ VI SEMESTER

UNIT 4 : EMBEDDED OPERATING SYSTEM AND MODELING

TOPIC 4.5: Embedded File Systems

Embedded File Systems

➢**Embedded File System (EFS)** is a proprietary file system used on NOR flash devices

Basic features are

Memory Organization :optimized for maximum performance

Allocation Information : reduced to a minimum, allowing small data overhead

File Names & Content : stored in fragments of variable size which provide optimal file access times

Memory Organization

- A NOR flash device memory array is physically divided into sectors or blocks
- The File System Component designates them as blocks
- Typically, a block's size is 64 kB which is also the smallest erasable unit. Blocks can be further divided, down to memory cells
- The memory cell size depends on the device architecture and is 8- (byte), 16- (half word) or 32-bit wide (word)
- The memory cell architecture also defines smallest programmable unit, which must be maximum 32-bit for use with the Embedded File System.

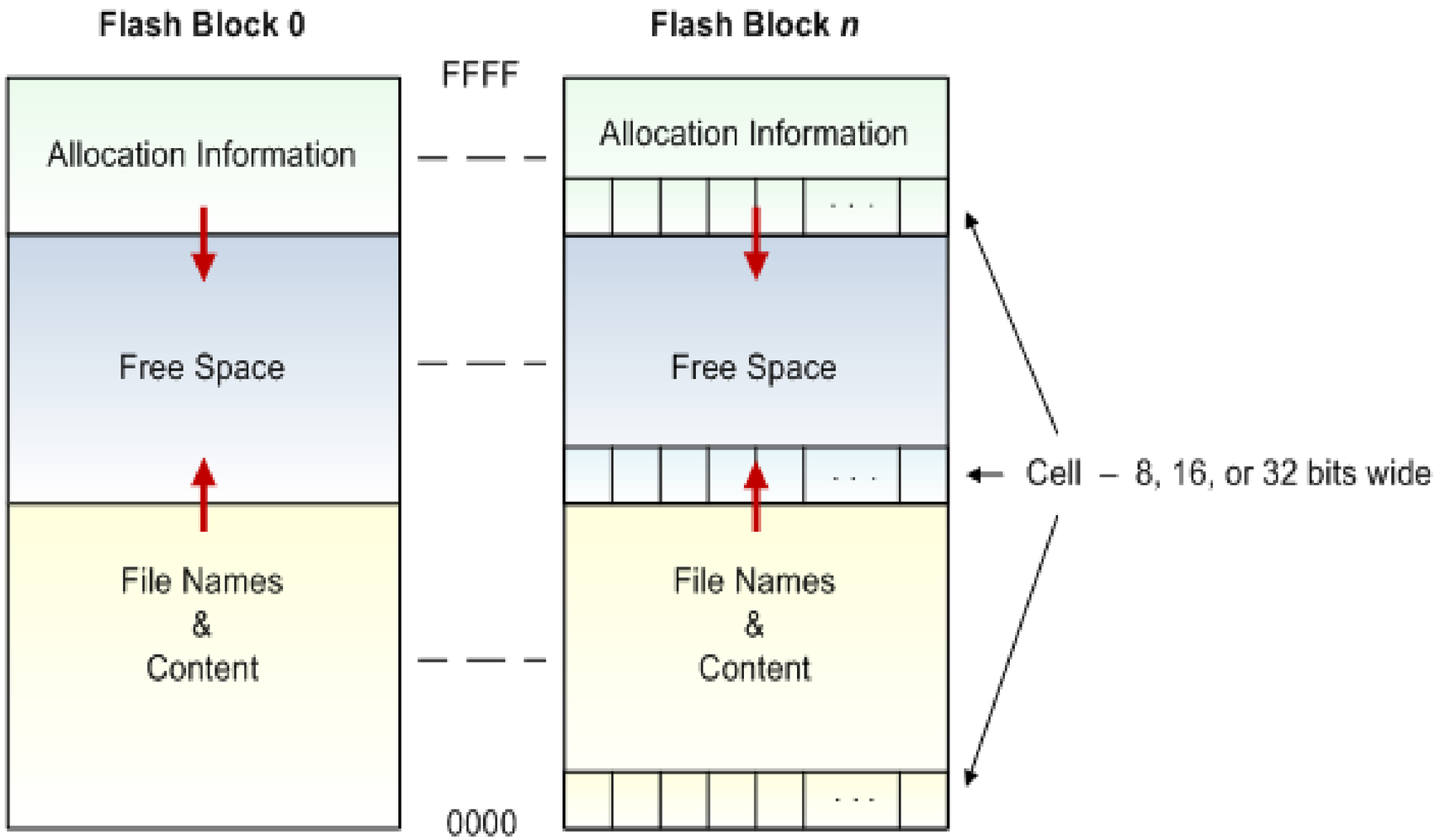
Embedded File System organizes each block into three regions

Allocation Information located on top of the block, grows in descending order and contains file allocation records.

Free Space

File Names & Content located on bottom of the block, grows in ascending order and contains file names and data.

Memory Organization



Flash Memory Organization

Allocation Information

➢Allocation information region is located on top of a block and describes the block's content

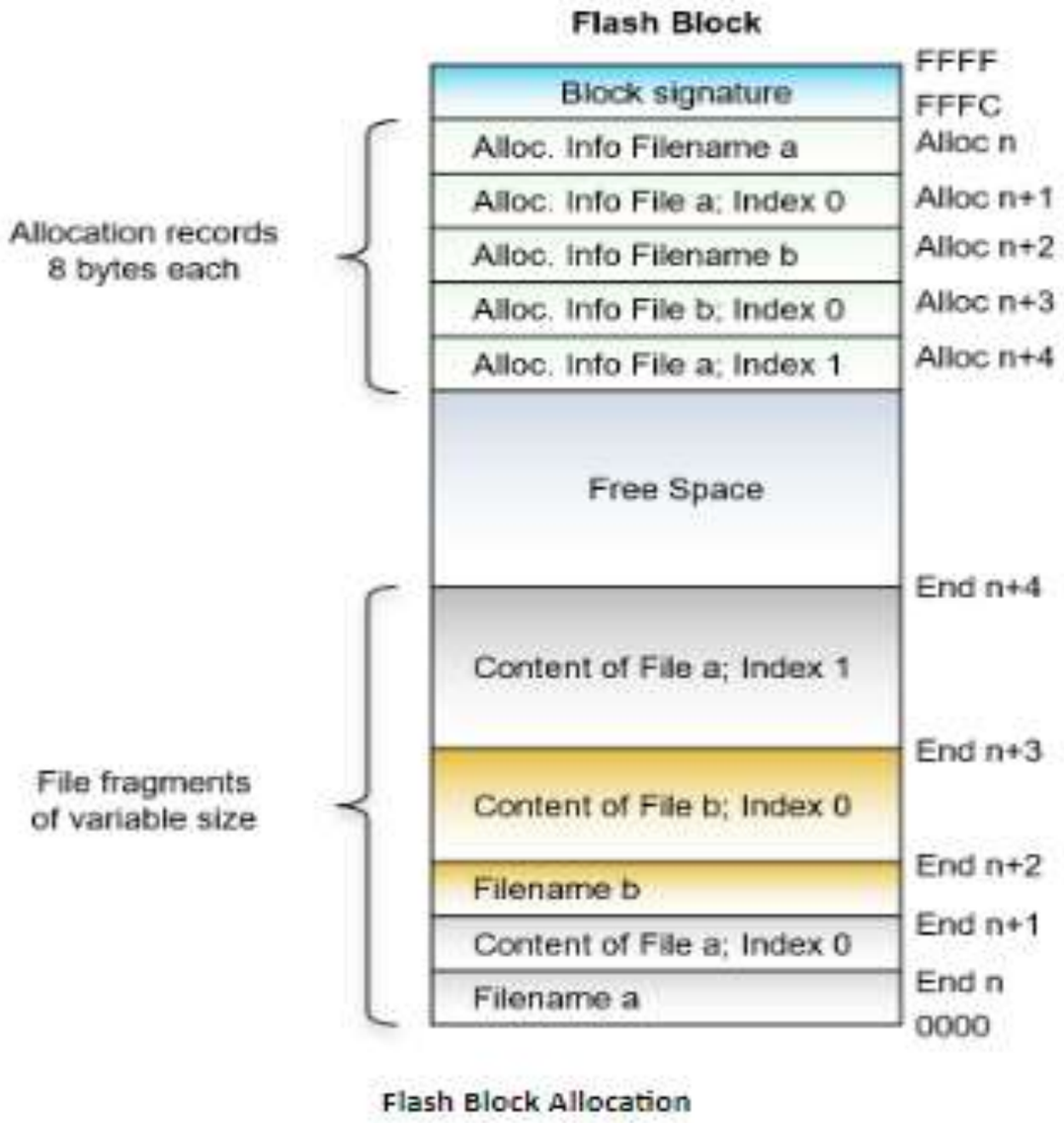
➢ It consists of block signature and file allocation information records, which are written in descending order

➢ Each file has at least one record associated with it. Multiple records belong to files with content and to fragmented files

➢A file is fragmented when it is modified or its content size exceeds a single block size and must be stored across several blocks

➢Several small files are stored into a single block

Allocation Information



Flash Block Allocation

Allocation Information

Block Signature

Each block contains a signature, consisting of 4 bytes and determines if block is:
empty i.e. erased
used but more data can be written into it
used temporarily during defragmentation
full and cannot be written anymore (only erased)

Allocation Information Record

The file allocation information record consists of 8 bytes and has the following components

end is the end address of the file fragment.
fileID is the file identification number and is associated with the file name.
index is the file fragment ordering number, which starts at 0 for each file.

```
struct falloc {
uint32_t end;
uint16_t fileID;
uint16_t index;
};
```

Allocation Information

The file allocation information is written when

- The file is opened for writing
- The file is closed
- The file is flushed and file fragment is not yet defined by the allocation information record
- The block is full and there is no more free space

File Names & Content

File Names & Content

➢The file names & content region is located at the bottom of a block and is fully defined through the file allocation information records

➢ It consists of file names and file content, which can both be fragmented

➢The first file fragment always starts at the beginning of a block (at offset 0) and is written in ascending order

File Names

➢In the Embedded File System, a file name consists of maximum 31 characters

➢Directories are not supported, therefore any file name which contains a directory separator character, such as slash (/) or backslash(\), is rejected as invalid. Other characters are allowed

.

File Names & Content

File Content

- Since file fragments are of variable size, create big file fragments in order to reduce the total number of file fragments and make the best use of a block

- Writing or appending small amounts of data to a file is not optimal, since such an approach creates a large number of allocation information records

- They consume free space and the required processing time results in a slow file access time

- When the file content is modified, the old file content is invalidated and a new file fragment is allocated

- a block is erased when all the data stored within has been invalidated

File Names & Content

Limitations

The following restrictions are applicable to the EFS:

- Maximum file name length is limited to 31 characters
- Minimum block size should be 512 bytes or more
- Directories or folders are not supported
- Multiple active file handles per file are allowed only for files opened in read mode
- Seeking (**fseek**) within files works only for files opened in read mode
- File update modes (r+, w+, a+) (**fopen**) are not supported
- Timestamp information is not supported for a file
- Drive partitions are not supported
- The EFS is not compatible with the FAT file system and cannot be used with a USB mass storage device.

THANK YOU