(54) Title: A DEVICE DRIVER FOR ACCESSING COMPUTER FILES
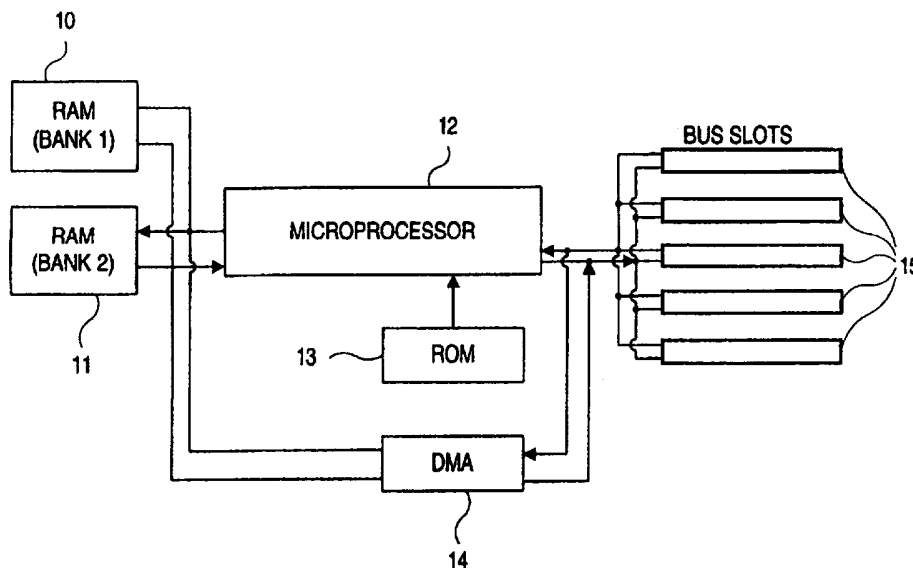
(57) Abstract

A computer device driver for accessing compressed files held in archives in a memory device, the device driver comprising: means for reading a compressed file from an archive in the memory device, decompressing the file in RAM and retaining the decompressed file in RAM in whole or in part so that operations to the memory device can be performed on the decompressed file by the operating system without having first to write the decompressed file to the memory device. The device driver also comprises means for accessing the file in said RAM and changing the contents of the file; and for returning the file to the archive in said memory device.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav	TM	Turkmenistan
BF	Burkina Faso	GR	Greece		Republic of Macedonia	TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's	NZ	New Zealand		
CM	Cameroon		Republic of Korea	PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

1

A DEVICE DRIVER FOR ACCESSING COMPUTER FILES

FIELD OF INVENTION

   The invention relates generally to device drivers in computer systems with memory devices. In particular, the invention relates to device drivers which provide for

5 accessing and updating compressed files on disk storage devices, particularly in personal computer systems.

SUMMARY OF THE INVENTION

   It is an object of the invention to provide for the fast and efficient accessing and updating of compressed files stored on storage devices.

10   A further object of the invention is to provide a system which displays archives as folders and provides for the fast accessing and updating of files on storage devices.

   Still another object of the invention is to provide a system in which virtual folders are accessible by an operating system.

   Yet another object of the invention is to provide a system which accesses

15 compressed files in memory for a user in a simple and understandable manner and which uses a minimal number of steps in performing the access.

   In accordance with the present invention, the foregoing objectives are realized by a computer device driver for interfacing with an operating system and for accessing compressed files held in archives in a memory device, said device driver comprising:

20 means for reading a compressed file from an archive in said memory device, decompressing said file in RAM and retaining the decompressed file in RAM in whole or in part so that operations can be performed on the decompressed file by the operating system without having first to write the decompressed file to the storage device; means for accessing said file in said RAM and changing the contents of said file; and means for

25 returning said file to said archive in said storage device.

BRIEF DESCRIPTION OF THE DRAWINGS

   FIG. 1 is a block diagram of a computer system using a device driver according to principles of the present invention;

2

FIG. 2 is a block diagram of the software hierarchy according to principles of the present invention;

FIG. 3 is a block diagram describing the operation of the device driver according to principles of the present invention;

FIG. 4 is a flowchart illustrating the operation of the device driver according to principles of the present invention; and

FIG. 5 is a flowchart illustrating the operation of the device driver according to principles of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Several illustrative embodiments of a device driver are described below as they might be implemented to provide for improved methods of accessing compressed files. It will of course be appreciated that in the development of any actual implementation, numerous implementation-specific decisions must be made to achieve the developers' specific goals and subgoals which will vary from one implementation to another. Moreover, it will be appreciated that such a development effort might be complex and time-consuming, but would nevertheless be a routine undertaking of device engineering for those of ordinary skill having the benefit of this disclosure.

The general architecture of the computer system that is one embodiment of this invention is shown in FIG. 1. The computer main memory or RAM can be divided into banks 10 and 11 and contain data, intermediate results as well as programs stored by the microprocessor 12. The microprocessor 12 can also execute instructions stored in ROM 13. The microprocessor 12 can be, for example, a i486 or Pentium processor manufactured by Intel Corporation. The ROM contains the Basic Input Output System (BIOS) which is a set of programs and data that are executed by the microprocessor at power-up. The BIOS also provides various support routines for input/output access and control. Also shown is a direct memory access (DMA) unit 14 which provides for the direct transfer of data from an outside source (connected at bus 15) and the RAM.

FIG. 2 illustrates the various levels of software present in a preferred embodiment of the invention. At the top level is an application program 20, at a lower level is an operating system 21, and at an even lower level is BIOS 22. The operating system 21 is

3

layered as well. Its center is occupied by the Kernel which comprises at least the

routines for memory management, executing task switches, and handling critical errors.

Device drivers run in a layer on top of the Kernel and supply functions such as data and

file management, character input, hardware device interfaces, and drive access. The

5 operating system can also use device drivers to bypass BIOS and access devices directly.

Turning next to FIGs. 3, 4, and 5 a preferred embodiment of a device driver

program embodying the present invention begins by receiving a file system request from

the operating system at step 301. The file system request contains information indicating

the type of request to be performed, the identity of the file on which the operation is to be

10 performed, and any other information needed to perform a successful operation. The

operation specified in the file system request may be an operation to enumerate the

contents of a folder, perform file system operations which do not modify the contents of

a file, or perform some type of operation that modifies the contents of a particular file.

At step 302, the driver determines if the file system request is a request to

15 enumerate the contents of a folder. If the answer at step 302 is affirmative, then at step

303, the driver checks to see if the enumeration points to an archive file. The term

"archive" is defined to mean a file which contains compressed files or other folders

(which themselves contain files and folders). For example, in a file system where

compressed files are denoted by using a .ZIP suffix, step 303 checks to see if a file with a

20 .ZIP suffix is the subject of the enumeration.

If the answer to step 303 is affirmative, then at step 304 the device driver creates,

in RAM, a virtual folder that represents the archive file. A virtual folder appears to user

programs and the operating system as a folder containing files and folders but is not

actually stored as a folder in the underlying storage device. The device driver then

25 augments the enumerated request with information describing the virtual folder and

passes this information to the operating system. This information includes file attributes

and a time and date stamp. By virtual, it is meant that the folder exists in RAM only and

is, therefore, temporary since upon program completion, the folder disappears. At this

point, nothing is contained inside the virtual folder.

30 If the answer at step 303 is negative, then the driver checks to see if the

information requested in the file system request is a description of the contents of a

4

virtual folder at step 305. If the answer to step 305 is affirmative, then the contents of

the virtual folder are revealed at step 306. By "revealing", it is meant that the contents

are made available to the operating system. These contents are contents of the archive

that the virtual folder represents. If the answer to step 305 is negative, then the request is

5   passed to the operating system at step 307.

If step 302 determines that a folder enumeration is not requested, then the driver

determines at step 308 whether a file system operation is requested. A file system

operation includes operations such as a "rename", "copy", "move", "delete", or "get

attributes" operations. If the operation requested is a file system operation, then the

10   driver continues at step 310 by checking to see if the requested operation is for a file

contained in a virtual folder. If the file is contained in a virtual folder, then the driver

continues with step 312 where it emulates (performs) the file system operation request.

Step 312 is expanded in the flowchart of FIG. 4 and discussed in greater detail below.

If the answer at step 310 is negative, then at step 312 the request is passed to the

15   operating system. In this case, a file would not be in a virtual folder and the operating

system can perform the operation using its normal procedures.

If the answer at step 308 is negative, then at step 309 the driver checks to see if

the request is for a file operation. By a file operation, it is meant "read", "write", "open"

or "close" file operations. If the answer is negative, then at step 307, the request is

20   passed onto the operating system. If the answer is affirmative, then at step 311, the

system checks to see if the file is in a virtual folder. If it is not, the driver, passes the

request to the operating system at step 307. However, if it is, at step 313 the driver

emulates the file request operation. Step 313 is expanded in the flowchart of FIG. 5 and

discussed in greater detail below.

25   Turning next to FIG. 4, the emulation of a file system operation begins at step

401 when the driver checks to see if the central directory of the archive has been loaded

into RAM. The central directory is an index of files and can be represented, for example,

by a conventional tree structure. If the directory has not been loaded, then at step 402 the

driver loads the central directory structure. The driver then continues with step 403.

If the central directory is loaded, then at step 403 the driver checks to see if the request is valid, i.e., that the file exists in the archive. If the request is not valid, then the driver posts an error message to the operating system at step 404.

If the path is valid, the driver next determines at step 405 whether the request is valid, i.e., whether all the correct parameters are present in the request. For example, for a copy operation whether a source and destination information are included. If the request is not valid, then at step 406 the driver reports an error to the operating system.

If the request is valid, then at step 407, the driver uses the central directory information to emulate a request. In other words, the central directory is modified to account for the differences due to the addition and deletion of files. For example, in a delete operation, the file is removed from the archive and the central directory is changed to reflect the deletion.

If the operation is a copy or a move, then the driver proceeds to FIG. 5. After executing the steps enumerated in FIG. 5, the driver then returns to the flowchart of FIG. 4 at step 408.

At step 408, the driver determines whether the central directory information has been changed. If it has not changed, then, at step 409 the procedure ends by returning to the operating system without updating the central directory. On the other hand, if the central directory structure has changed, then at step 410 the driver updates the archive central directory and the files affected in the archive.

Turning next to the flowchart of FIG. 5, emulation of a file operation begins when the driver determines whether the central directory has been loaded from the disk at step 501. As described above, the central directory is an indexed listing of the contents of the archive. The directory can be represented, for example, by a tree structure. If the directory is not loaded from disk, then at step 502 the driver loads the directory from the hard drive and then control proceeds to step 503. If the driver determines that the directory is loaded, then the driver also continues by executing step 503.

If the central directory has been loaded, then at step 503 the driver checks to see if the request is valid, i.e., that the file exists in the archive. If the request is not valid, then the driver posts an error message to the operating system at step 504.

6

If the path is valid, the driver next determines at step 505 whether the request is valid, i.e., whether all the correct parameters are present in the request. For example, for an open operation whether the source file is already open for writing by another process. If the request is not valid, then at step 506 the driver reports an error to the operating system.

At step 507, the driver determines whether the operation is an open operation, that is, a request to open a file and obtain a handle for it. If the operation is an open operation, at step 512 the driver creates data structures in RAM necessary for opening the file. Next, at step 517 the driver returns a file handle which refers to the file in the archive. A handle is a unique identifier, such a number or pointer, that serves to identify the file. Upon a read operation, the driver will check whether a handle is present.

If the answer to step 507 is negative, that is the operation is not an open operation, then at step 508 the driver determines whether the operation is a read operation. If the answer is affirmative, then at step 513, the driver determines whether the file has been decompressed. If it has, at step 518 the driver provides the file information requested by the read operation. If the answer at step 513 is negative, then at step 516, the driver decompresses the file and places it in RAM before proceeding with step 518.

If the answer to step 508 is negative, that is the operation is not a read operation, then at step 509 the driver determines whether the operation is a write operation. If the answer is affirmative, then at step 514, the driver determines whether the file has been decompressed. If it has, at step 519 the driver copies the write data to memory. If the answer at step 514 is negative, then at step 516, the driver decompresses the file and places it in RAM before proceeding with step 519.

If the answer to step 509 is negative, in other words, the operation is not a write, step 510 checks to see if the operation is a close operation. If the operation is not a close, the driver returns control to the operating system at step 511 and ignores the attempted operation.

If the answer to step 510 is affirmative, then at step 515 the driver determines if the file data has changed. If no changes are detected, then at step 515 the data structures are cleaned up. By cleaning up, it is meant that the memory allocated at step 512 is

released as well as the memory allocated for the file data. On the other hand, if the file data has changed, the driver continues at step 520 where it updates the archive data file on disk. That is, it writes the data from RAM to the appropriate place in disk. Control then proceeds to step 521 where the archive central directory structure is updated. The

5   procedure concludes at step 522 where any data structures left over are cleaned up, i.e., removed so that space can be used by other programs.

As discussed above, the driver of the present invention has the advantage that the contents of an archive, i.e., its compressed files and folders, are made available throughout the operating system. That is, a user and the operating system have full

10   access to the contents of an archive through the virtual folder that represents the archive no matter what function the operating system is performing.

The driver above also is fast and efficient. This is because the virtual folder and all files that are accessed in the archive are cached in RAM. As is well known in the art, using RAM to house files is a fast and efficient method for file manipulation.

15   The driver is efficient for other reasons. The user sees a "real" file when it is in fact virtual using the above mechanism. The user can perform operations on the file directly without having to first explicitly compress and/or decompress data.

The driver also makes accessing compressed files stored in memory easy and simple for the user since the user is not required to start and execute a separate

20   decompression program.

While the present invention has been described with reference to one or more particular embodiments, those skilled in the art will recognize that many changes may be made thereto without departing from the spirit and scope of the present invention, which is set forth in the following claims.

8

CLAIMS:

1.   A computer device driver for interfacing with an operating system and for accessing compressed files held in archives in a storage device, said device driver comprising:

        means for reading a compressed file from an archive in said memory device,
5   decompressing said file in RAM and retaining the decompressed file in RAM in whole or in part so that operations can be performed on the decompressed file by the operating system without having first to write the decompressed file to the storage device,

        means for accessing said file in said RAM and changing the contents of said file; and

10   means for returning said file to said archive in said storage device.

2.   The driver of claim 1 further including means for compressing said file.

3.   The driver of claim 1 wherein said means for accessing and changing the contents of said file is responsive to a request from an operating system.

4.   The driver of claim 3 wherein said request from said operating system is
15 selected from the set of requests for reading, writing, opening, and closing said file.

5.   The driver of claim 3 wherein said request from said operating system is selected from the set of requests to renaming, copying, moving, deleting, or obtaining the attributes of said file.

6.   The driver of claim 1 wherein said file in said RAM is accessible to an
20 operating system and user programs.

7.   The driver of claim 1 wherein said driver is part of an operating system.

8.   A computer device driver for interfacing with an operating system and application programs, and for accessing compressed files held in archives in a memory device, said device driver comprising:

means for creating virtual folders or directories for archives to represent archives as folders throughout all levels of the operating system and application programs; and

means for representing files and folders contained in archives as files and folders in virtual folders corresponding to said archives.

9.   A computer device driver for interfacing with an operating system and accessing compressed files held in archives in a memory device said device driver comprising:

means for creating virtual folders or directories, said virtual folders and directories representing archives and directories on said memory device and accessible to all levels of said operating system;

means for representing files and folders contained in said archives as files and folders in the virtual folders and directories corresponding to said archives;

means for updating said folders and said memory device after an operation from said operating system on said files and virtual folders;

means for selectively supplying said operating system with information from said virtual folders; and

means for updating and returning said files and folders to said archive in said memory device.

10.   The driver of claim 9 wherein said virtual folder is stored in RAM.

11.   The driver of claim 9 further comprising means for receiving a request from said operating system, said request from said operating system selected from the set of requests for reading, writing, opening, and closing said file.

12.   The driver of claim 9 further comprising means for receiving a request from said operating system, said request from said operating system selected from the set of requests for renaming, copying, moving, deleting, or obtaining the attributes of said file.

13.   A computer device driver for interfacing with an operating system and for accessing compressed files held in archives in a memory device, said compressed files accessible using a directory structure, said device driver comprising:

means for creating virtual folders or directories in RAM, said virtual folders and

5   directories representing archives on said memory device and accessible by an operating system;

means for representing files and folders contained in said archives as files and folders in the virtual folders corresponding to said archives;

means for reading a compressed file from said archive in said memory device,

10   decompressing said file in RAM, and retaining the decompressed file in RAM in whole or in part so that operations can be performed on the decompressed file by the operating system without having first to write the decompressed file to the memory device;

means for accessing said file in said RAM and changing the contents of said file;

means responsive to accessing said files for adjusting said virtual directory

15   structure; and

means for returning said file to said archive in said memory device.

14.   A device driver for accessing computer files held in archives on a memory device and selectively passing information from said files to an operating system, said device driver comprising:

20   means for identifying and determining an operation requested to be performed by said operating system on a particular compressed file on said memory device;

means for extracting said particular compressed file from an archive on said memory device, decompressing said particular file, and placing said file in a RAM for a time sufficient to perform said operation;

25   means for supplying said operating system with information from said compressed file in said memory device as needed by said operating system;

means for changing the contents of said file; and

means for returning said file to said archive in said memory device.

11

15.   A computer system comprising:

an operating system;

permanent memory for permanently storing archives, said archives containing files and folders;

5   temporary memory for temporarily storing files, folders, or directories;

a device driver means comprising:

means for creating virtual folders or directories in said temporary memory, said virtual folders and directories representing archives and directories in said permanent memory and accessible by said operating system;

10   means for representing files and folders contained in said archives as files and folders in the virtual folders and directories corresponding to said archives;

means for updating said virtual directories and said permanent memory directories after an operation on said files and folders;

means for selectively supplying said operating system with information from said

15   files; and

means for returning said file to said archive in said permanent memory.

16.   The system of claim 15 wherein said permanent memory comprises a plurality of PROMs.

17.   The system of claim 15 wherein said temporary memory comprises a

20   plurality of RAM banks.

18.   The system of claim 15 further comprising an application program.

19.   A method for accessing compressed files held in archives in a storage device , said method comprising the steps of:

reading a compressed file from an archive in said memory device, decompressing

25   said file in RAM and retaining the decompressed file in RAM in whole or in part so that operations can be performed on the decompressed file by the operating system without having first to write the decompressed file to the storage device,

accessing said file in said RAM and changing the contents of said file; and

returning said file to said archive in said storage device.

20. A method for accessing compressed files held in archives in a memory device, said compressed files accessible using a directory structure, said method comprising the steps of:

creating virtual folders or directories in RAM, said virtual folders and directories representing archives on said memory device and accessible to an operating system;

representing files and folders contained in said archives as files and folders in the virtual folders corresponding to said archives;

reading a compressed file from said archive in said memory device, decompressing said file in RAM, and retaining the decompressed file in RAM in whole or in part so that operations to the memory device can be performed on the decompressed file by the operating system without having first to write the decompressed file to the memory device;

accessing said file in said RAM and changing the contents of said file;

adjusting said virtual directory structure; and

returning said file to said archive in said memory device.

21. A method for accessing computer files held in archives on a memory device and selectively passing information from said files to an operating system, said method comprising the steps of:

identifying and determining an operation requested to be performed by said operating system on a particular compressed file on said memory device;

extracting said particular compressed file from an archive on said memory device, decompressing said particular file, and placing said file in RAM in whole or in part for a time sufficient to perform said operation;

supplying said operating system with information from said compressed file in said memory device as needed by said operating system;

changing the contents of said file; and

returning said file to said archive in said memory device.

22.   A computer device driver for interfacing with an operating system and for accessing compressed files held in archives in a storage device , said device driver comprising:

means for identifying and determining an operation requested to be performed by
5   said operating system on a particular compressed file on said storage device;

means for creating virtual folders or directories in RAM, said virtual folders and directories representing archives on said memory device and accessible to an operating system;

means for representing files and folders contained in said archives as files and
10   folders in the virtual folders corresponding to said archives;

means for reading a compressed file from an archive in said memory device, decompressing said file in RAM and retaining the decompressed file in RAM in whole or in part so that operations can be performed on the decompressed file by the operating system without having first to write the decompressed file to the storage device,

15   means for copying, moving, renaming, deleting, or getting the attributes of said file;

means for reading, writing, opening or closing said file;

means responsive to copying, moving, renaming, or deleting of a file and adjusting the virtual directory structure;

20   means for supplying said operating system with information from said compressed file in said memory device as needed by said operating system; and

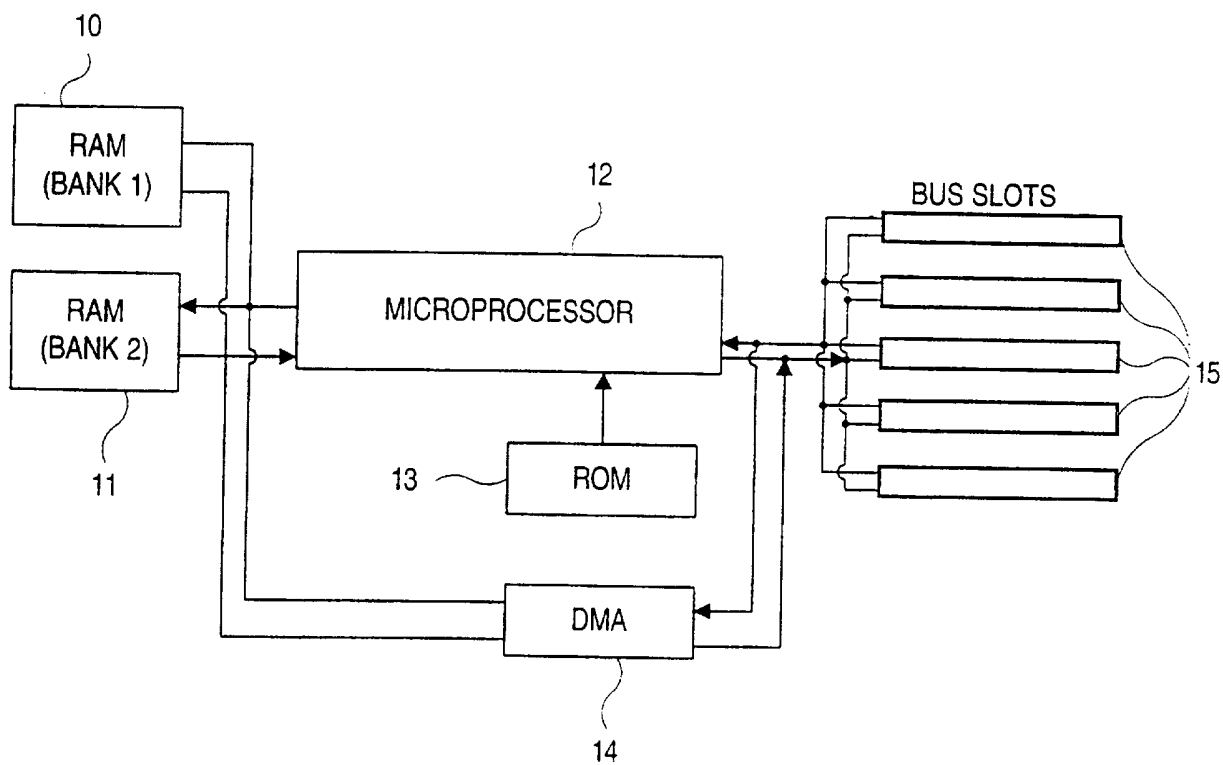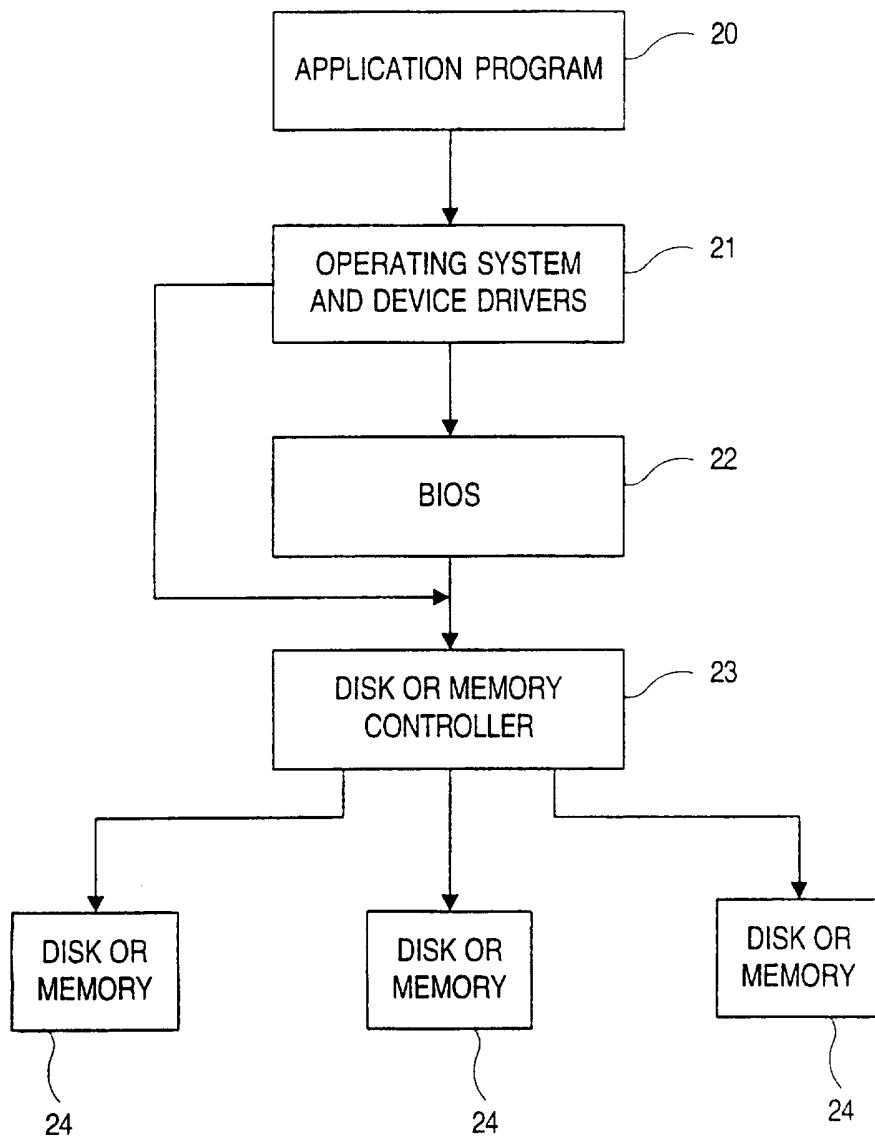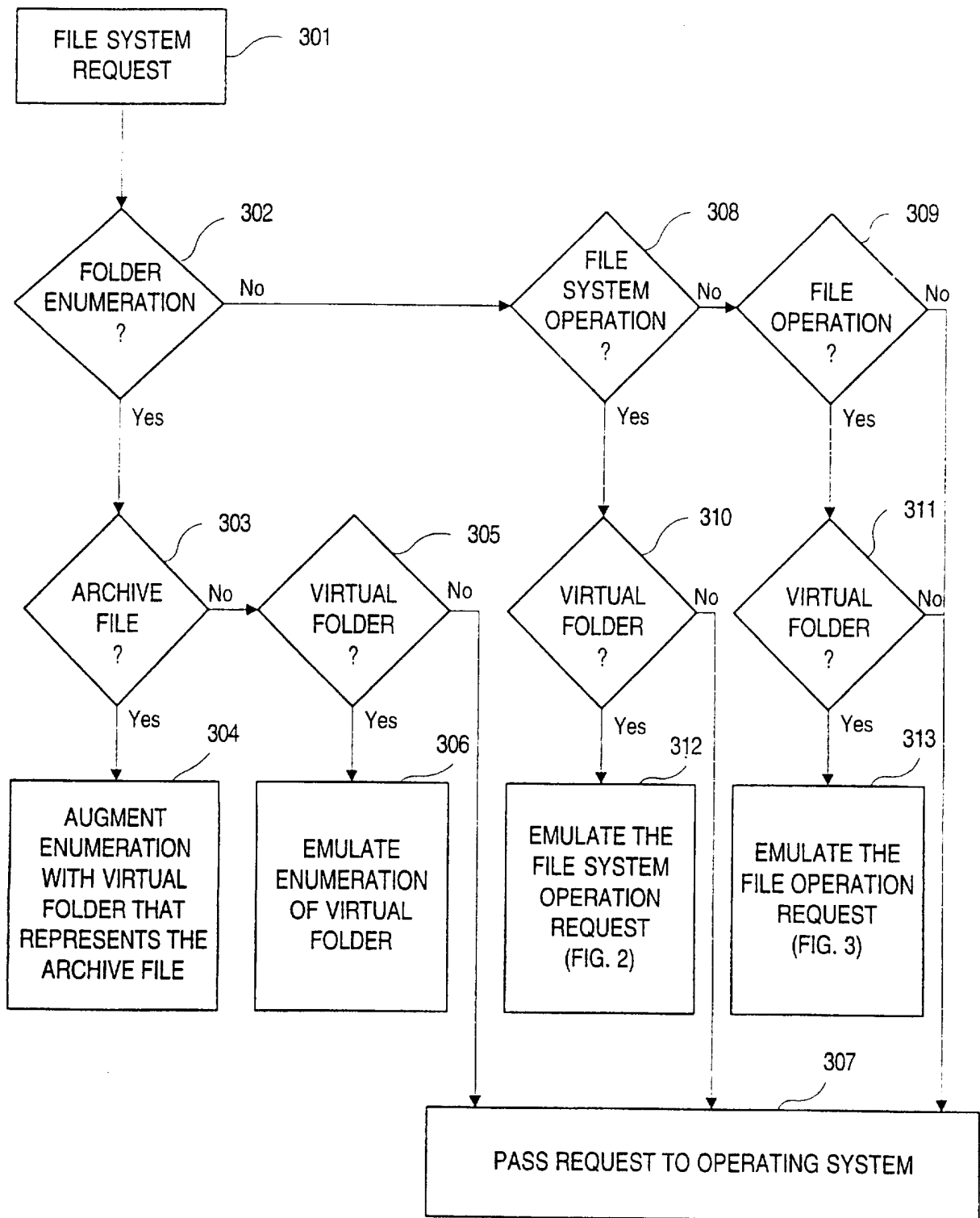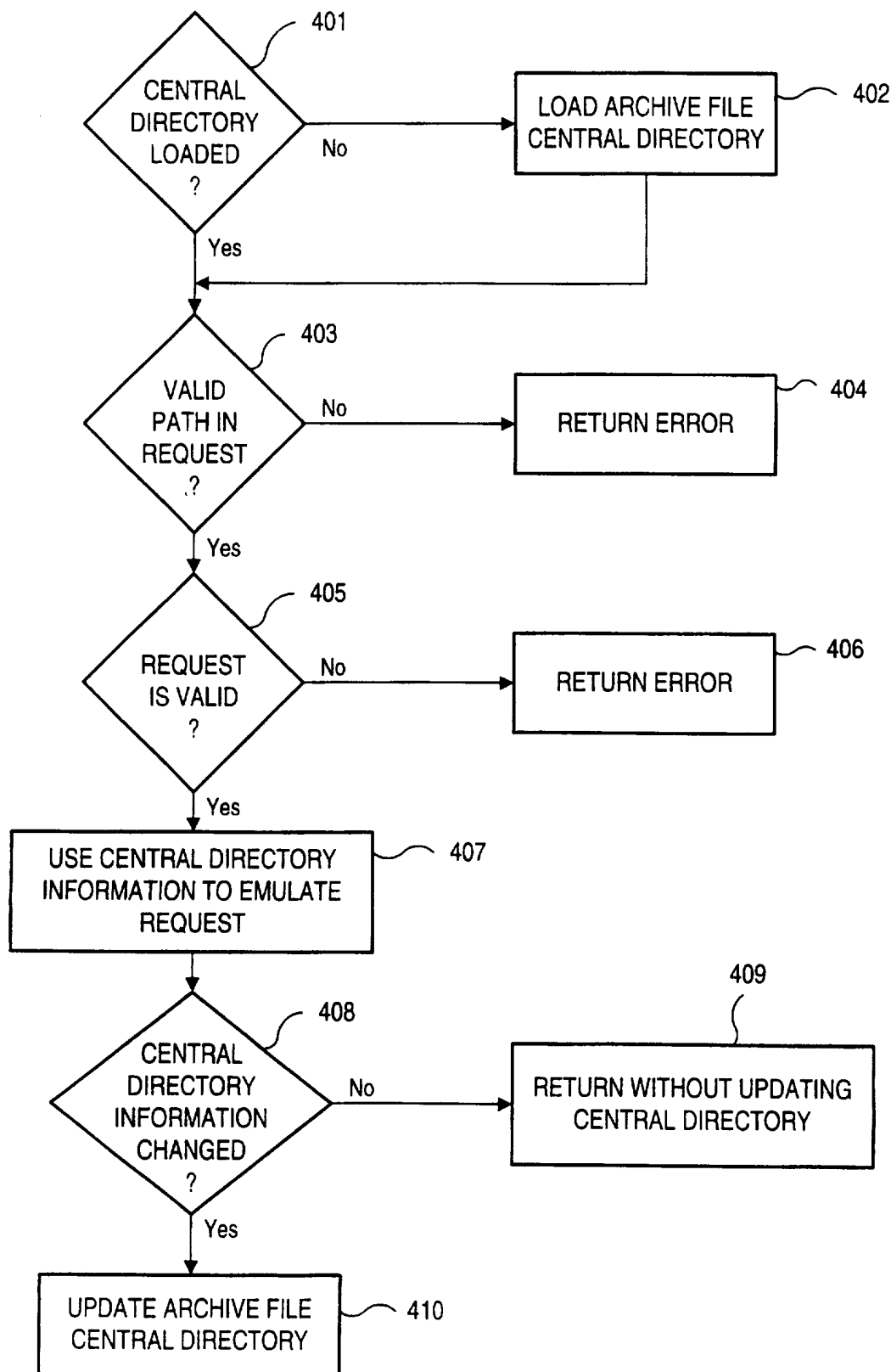means for returning said file to said archive in said memory device.

FIG. 1

FIG. 2

```
                    ┌─────────────┐
                    │ FILE SYSTEM │───── 301
                    │   REQUEST   │
                    └─────────────┘
                          ┊
                          ▼
                   ◇ 302
            ╱ FOLDER ╲
           ╱ ENUMERATION ╲   No
           ╲     ?      ╱ ──────────────────────►
            ╲         ╱
                │ Yes
                ▼
          ◇ 303              ◇ 305
       ╱ ARCHIVE ╲  No   ╱ VIRTUAL ╲  No
      ╱   FILE    ╲ ───► ╱  FOLDER  ╲ ───►
      ╲     ?     ╱      ╲    ?     ╱
       ╲        ╱          ╲      ╱
          │ Yes                │ Yes
          ▼      304           ▼       306
   ┌──────────────┐      ┌──────────────┐
   │  AUGMENT     │      │   EMULATE    │
   │ ENUMERATION  │      │ ENUMERATION  │
   │ WITH VIRTUAL │      │ OF VIRTUAL   │
   │ FOLDER THAT  │      │   FOLDER     │
   │ REPRESENTS THE│     └──────────────┘
   │ ARCHIVE FILE │
   └──────────────┘
```

FILE SYSTEM REQUEST — 301

FOLDER ENUMERATION ? — 302 No

ARCHIVE FILE ? — 303 No   Yes

VIRTUAL FOLDER ? — 305 No   Yes

AUGMENT ENUMERATION WITH VIRTUAL FOLDER THAT REPRESENTS THE ARCHIVE FILE — 304

EMULATE ENUMERATION OF VIRTUAL FOLDER — 306

FILE SYSTEM OPERATION ? — 308 No   Yes

FILE OPERATION ? — 309 No   Yes

VIRTUAL FOLDER ? — 310 No   Yes

VIRTUAL FOLDER ? — 311 No   Yes

EMULATE THE FILE SYSTEM OPERATION REQUEST (FIG. 2) — 312

EMULATE THE FILE OPERATION REQUEST (FIG. 3) — 313

PASS REQUEST TO OPERATING SYSTEM — 307

FIG. 3

FIG. 4

FIG. 5