(54) Title: SOFTWARE ANTI-VIRUS FACILITY

(57) Abstract

A virus-resistant disk has a "hidden partition" in which anti-virus software is stored; the hidden partition not only shields the software from many viruses, but provides storage space that does not reduce the disk's formatted or advertised capacity. The disk includes software to cause the computer to execute the anti-virus software. The invention provides a hidden partition by utilizing storage space on the disk that is not reflected in the size and geometry information stored on the disk, e.g., in the BIOS Parameter Block.

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		

SOFTWARE ANTI-VIRUS FACILITY

<u>COPYRIGHT NOTICE</u>

<u>FIELD OF THE INVENTION</u>

10

The invention relates to detecting, identifying and removing computer viruses

from components of a computer, and more particularly to a startup (bootstrap) disk

that stores anti-virus software and facilitates the execution of this software at

bootstrap time, i.e., before the computer begins executing an ultimate operating

15   system (e.g., MS-DOS).

<u>BACKGROUND OF THE INVENTION</u>

A computer virus is software that is executed by a computer without the

knowledge or authorization of the computer's user and that causes unauthorized and

unwanted changes to components of the computer or to information stored on the

20   computer. For example, some viruses alter or destroy data stored on disk, scramble

characters on video display screens, display misleading messages to users, and

consume computer or network resources thereby preventing users from performing

desired tasks. A virus usually attempts to propagate itself to other computers by

making copies of itself on any accessible diskettes or hard disks (collectively "disks")

25   or other non-volatile memory such as "flash" read-only memory (ROM). A virus

"attack" herein means any change made to a computer component by a virus, such as

a change to stored information or the making of a propagation copy of the virus.

Viruses typically propagate by opportunistically copying themselves to

virus. If a user transports an infected disk to a second computer and the second computer executes the virus, the virus then attempts to infect disks on the second computer, and so on. Viruses generally employ one of two techniques to cause a subsequent execution of themselves. Some viruses attach themselves to application

5 programs that are stored on a disk. When a user runs the infected program, the virus also executes. Most viruses, however, are "bootstrap-time viruses" that replace the startup (bootstrap) program located on the infected disk with a program that causes the computer to execute the virus if the disk is subsequently used as a startup disk on this or another computer. Once the virus executes, it arranges for itself to remain in

10 the memory of the computer after bootstrap, but causes the computer to execute the normal bootstrap program so as to mask its presence. Following completion of bootstrap, the virus remains stored on the computer and capable of causing further mischief.

Bootstrap programs execute as part of a bootstrap sequence initiated by the

15 application of power or a reset signal to a computer. During this sequence, the computer performs a power-on self-test ("POST"), then locates a bootstrap program on a disk and then executes the bootstrap program. The bootstrap program is always stored at a characteristic, fixed location (the "boot block") on the disk. Generally, the bootstrap program readies the computer for normal functioning by causing it to load

20 and execute an operating system, such as MS-DOS, OS/2, NetWare, UNIX, or Windows-NT, although the bootstrap program can also cause the computer to execute one or more other programs prior to executing the operating system. Accordingly, as used herein, the term "bootstrap" includes the time and steps taken between the application of power or reset signal and the execution of the last program

25 prior to the operating system; and "operating system" is software that manages a computer's resources (e.g., disks and memory) and provides low-level services (e.g., I/O, timer, memory management, interprogram communication) to application programs. An "application program" is not part of an operating system and can only execute under the control of an operating system.

30 To overcome the problems created by viruses, practitioners have developed a variety of "anti-virus" programs that both detect and remove known viruses. Anti-virus software searches for "signatures", including characteristic behaviors, of viruses and

include Command Software Systems F-PROT, IBM AntiVirus, and Sophos Sweep. However, and quite problematically, bootstrap-time viruses can interfere with the operation of prior art anti-virus software. In addition, the presence of an operating system can obscure the presence of a virus in the memory of a computer.

5   As noted previously, a bootstrap program is stored at a characteristic disk location or address. For "addressing" purposes, disks are divided into surfaces, tracks, and sectors. The "formatted capacity", in bytes, of a disk (also known as the "advertised capacity") equals the product of the number of: surfaces, tracks per surface, sectors per track, and bytes per sector of the disk. A hard disk can be further 10 divided into one or more logical "partitions", each partition being treated as a separate disk. Generally, the first sector of a diskette and the first sector of each partition of a hard disk contains a disk descriptor block which contains size and geometry information about the disk, such as the number of sectors per track. "BIOS Parameter Block" herein refers to the area on the disk where this information is characteristically 15 stored. The following Table 1 lists the relevant fields of the BIOS Parameter Block. "Conventional storage capacity" of a disk herein means the formatted capacity of the disk as reflected by information in the BIOS Parameter Block of the disk.

Size (bytes)	Field
2	Number of bytes per sector
2	Total number of sectors in volume (logical partition)
2	Number of sectors per track
2	Number of surfaces (heads)
2	Number of entries in root directory

Table 1 - Selected Fields of the BIOS Parameter Block

20   Computers can include storage devices other than conventional disks. For reasons of design or in an emergency, the computer can be booted from a non-disk device such as a flash ROM, removable bubble random-access memory (RAM), an optical storage device such as a CD-ROM, or a tape drive. These devices, too, can be infected with viruses.

It is therefore an objective of the invention to provide a startup storage medium that causes the computer to automatically execute anti-virus software each time the computer starts from the storage medium, i.e., during bootstrap, so as to detect bootstrap-time viruses before or after they have executed and
5   implanted themselves in the system.

It is a further objective to provide a storage medium that stores the anti-virus software without reducing the amount of conventional storage capacity of the storage medium.

It is a further objective to provide a storage medium that stores the anti-
10   virus software so as to make it inaccessible to many viruses.

It is a further objective to provide a mechanism to detect and repair virus-inflicted damage to the anti-virus software.

It is a yet further objective to provide a virus-tolerant storage medium that can withstand an attack by a virus without incurring damage to information
15   stored thereon.

Other objectives will, in part, be obvious and will, in part, appear hereinafter. The invention accordingly comprises an article of manufacture possessing the features and properties exemplified in the constructions described herein and the several steps and the relation of one or more of such
20   steps with respect to the others and the apparatus embodying the features of construction, combination of elements and the arrangement of parts which are adapted to effect such steps, all as exemplified in the following detailed description, and the scope of the invention will be indicated in the claims.

25   **SUMMARY OF THE INVENTION**

The invention provides a storage medium on which is stored anti-virus software and/or software designed to detect and repair damage to stored information, as well as instructions to cause a computer to execute the software whenever the disk is used to start the computer. The invention may be broadly
30   applied to a variety of storage media amenable to selective actuation by a

user—that is, designation by the user as a temporary or permanent "system" device from which the computer can boot. Most commonly, computers boot from a disk: the permanently installed hard drive or a selectably introduced diskette, either of which contains the necessary bootstrap program. A user may,

5 however, designate a non-disk device such as a flash ROM, removable bubble RAM, an optical storage device such as a CD-ROM, or a tape drive as the system device that the computer accesses during startup. Another alternative is a non-local storage medium, i.e., one not intimately associated with the computer, such as a network disk configured to operate as a shared resource

10 among the computers connected to the network (where the network may be a local-area network, a wide-area network or even the Internet). Any of these storage devices may be selected or designated by a user in the sense of configuring the computer to look to this device for the bootstrap program.

Alternatively, the invention provides a storage medium having a "hidden

15 partition" in which the anti-virus and/or repair software is stored; the hidden partition not only shields the software from many viruses, but provides storage space that does not reduce the medium's formatted or advertised capacity. Once again, the concept of a partition is not confined to disks, but applies to the various storage media discussed above. The storage medium of the present

20 invention preferably also includes software to cause the computer to execute the anti-virus software. The invention provides a hidden partition by utilizing space on the storage medium that is not reflected in the size and geometry information stored on the medium, e.g., in the BIOS Parameter Block. Heretofore, no reason existed for a virus to assume storage space existed on a disk beyond that

25 reflected by the BIOS Parameter Block or to access such space because the BIOS Parameter Block is the standard mechanism for identifying the configuration and storage capacity of a disk.

A "conventional program" herein means a program that either relies on the accuracy of the BIOS Parameter Block to access the storage medium or

30 accesses the medium via an operating system that so relies. Most viruses are conventional programs. Advantageously, the invention utilizes the hidden

6

partition to make programs and information stored in the hidden partition inaccessible to most existing viruses. In addition, storing programs and information in the hidden partition does not reduce the amount of available space on the storage medium, as reflected by the BIOS Parameter Block.

5   The invention provides the hidden partition by taking advantage of unutilized storage capacity on the storage medium. For example, diskettes commonly have gaps between the sectors of a track. In one embodiment, the invention provides a diskette in which at least some of the gaps have been reduced in size, thereby bringing closer together the sectors of the track and

10   leaving enough room for one or more additional sectors on the track. Not all tracks need have the same number of additional sectors. The additional sectors on all the tracks collectively form the hidden partition. Information in the BIOS Parameter Block does not reflect the existence of the additional sectors.

        Hard disks generally have unutilized storage capacity in the form of spare

15   sectors and/or spare tracks, the existence of which is not reflected by information in the BIOS Parameter Block. A hard disk lacking such spares generally can be formatted to contain them.

        The invention provides a storage medium on which a special bootstrap program is stored in the boot block and, if the storage medium functions at

20   startup, the normal bootstrap program is stored elsewhere on the medium. The special bootstrap program causes the computer to execute the software stored in the hidden partition and then, upon completion of that software (and if the storage medium functions at startup), to execute the normal bootstrap program, which causes the computer to execute the operating system. A source listing of

25   an exemplary special bootstrap program appears later in this specification.

        Alternatively, the storage medium also contains a bootstrap-time operating system ("BTOS") and the special bootstrap program causes the computer to execute the BTOS. The BTOS causes the computer to execute one or more anti-virus, disk-maintenance (described below), and/or other programs

30   stored in the hidden partition and then, upon completion of the programs (and if the storage medium functions during startup), to execute the normal bootstrap

program. The normal bootstrap program causes the computer to perform an otherwise-normal startup, including executing the operating system (e.g., MS-DOS) with which the user can interact and which provides services to application programs. The operating system invoked by the normal bootstrap

5   program is herein referred to as the "ultimate operating system" to distinguish it from the BTOS. "Post-bootstrap" herein means the time and steps taken after execution of the normal bootstrap program.

The BTOS facilitates the development and use of programs that must be executed before an ultimate operating system is executed. Executing before the

10 ultimate operating system provides advantages to anti-virus software because the software can take advantage of conditions within the computer that only exist during bootstrap and do not exist after the computer begins executing the ultimate operating system. For example, an "interrupt vector" contains the address of a routine that is associated with the vector and that handles requests

15 for service from a program or device. Whenever the program or device (e.g., a hard disk) requests service ("generates an interrupt"), the computer uses the address in the vector to pass control to the service routine. During bootstrap, the disk vector initially points to a well-known routine within the Basic Input/Output System (BIOS), but the ultimate operating system changes the

20 vector to point to a routine within itself. Many viruses modify the disk vector to point to themselves, but it is difficult for an anti-virus application program to distinguish a disk vector that points to a virus from one that points to a routine within an ultimate operating system; thus it is easier for anti-virus software to detect the presence of a virus by examining the disk vector before the computer

25 begins executing an ultimate operating system. The BTOS provides additional advantages in that developers of programs that must be executed before an ultimate operating system are relieved of developing special mechanisms that alter normal startup operations in order to invoke the programs; in addition, the BTOS provides low-level services to the programs and so simplifies them, and

30 relieves users from concerning themselves with the details of invoking potentially several bootstrap-time programs.

Software that accesses the hidden partition is written with the knowledge that the additional sectors exist. The software accesses the additional sectors via program calls to the BIOS or BTOS rather than to the ultimate operating system because the BIOS and BTOS allow access to arbitrary addresses on the

5   disk regardless of the contents of the BIOS Parameter Block and the ultimate operating system does not.

Alternatively, the storage medium also contains an application program that causes the computer to execute the anti-virus, disk-maintenance and/or other software that is stored in the hidden partition. This application program is

10   not a conventional program because it is capable of accessing the hidden partition without relying on the BIOS Parameter Block or using an operating system that so relies. Such a program allows the user to run the anti-virus software even if a virus has attacked the special startup program and thereby prevented execution of the anti-virus software at bootstrap time. The anti-virus

15   program examines the special startup program located in the boot block and repairs any damage done to it so it will function properly on subsequent bootstraps.

In case a virus attacks a storage medium having the anti-virus capabilities described above, the invention enables the disk to withstand the attack without

20   incurring damage to files stored on the medium. The invention provides a virus-tolerant storage medium that prevents an operating system from placing files in sectors that are known to be attacked by several viruses, thereby shielding the files from attack by the viruses. An operating system uses a file system to manage storage space on a storage medium (typically a disk) and create,

25   delete, catalog and facilitate access to files on the storage medium. On a disk, storage space is allocated in units of "clusters" (contiguous groups of sectors). All clusters on a disk contain the same number of sectors and each cluster has a "cluster number" and a corresponding entry in a file system allocation table, e.g., the File Allocation Table or "FAT" in MS-DOS. The file system "chains" together

30   clusters to allocate space to a file. Each FAT entry records whether the corresponding cluster is available, bad, or in use by a file. If it is in use, the

9

entry contains the cluster number of the next cluster in the chain for the file or, if it is the last cluster of the file, a "last-cluster" flag. The following Table 2 lists the relevant field of a FAT entry. A "directory" catalogs files stored on a disk. For each file, a directory stores the file's attributes, e.g., read-only, and correlates

5   the file's name and extension to its first cluster number. Directories can be organized into a hierarchy starting at a "root" directory. The following Table 3 lists the relevant fields of a directory entry. Disk-maintenance software, such as CHKDSK, SCANDISK and SpinRite, uses well-known techniques to detect and repair damage to information, such as the FAT and directory, stored by the file

10   system and to overcome recording errors, such as signals that are recorded off-center on a track. The invention provides a storage medium having entries in its file system allocation table that prevent the allocation to any file of sectors that are known to be attacked by viruses. For example, the entries can mark the sectors as being "bad" or already in use.

15

Size	Field
3 nibbles (for 12-bit FAT) 2 bytes (for 16-bit FAT) 4 bytes (for 32-bit FAT)	Next cluster number or flag

Table 2 - Selected Field of a FAT (File Allocation Table) Entry

Size (bytes)	Field
8	File Name
3	File Extension
1	File Attributes (e.g. read-only)
2 (4 for 32-bit FAT)	First Cluster Number

Table 3 - Selected Fields of a Directory Entry

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing discussion will be understood more readily from the following detailed description of the invention, when taken in conjunction with the accompanying drawings, in which:

5   Fig. 1 is a diagram of the sectors of one surface of a diskette that has a hidden partition according to the invention;

Fig. 2 is a diagram of the sectors of one surface of a hard disk having a hidden partition according to the invention; and

Fig. 3 is a flowchart illustrating a bootstrap sequence involving a disk
10   provided by the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

The following discussion illustrates application of the invention to a storage medium in the form of a disk or diskette, it being understood that other storage media (e.g., flash ROM, removable bubble RAM, an optical storage
15   device such as a CD-ROM, or a tape drive) may be substituted therefor and are considered within the scope of the invention. Refer first to Fig. 1, which shows generally at 100 a surface of a diskette according to the invention, with representative tracks 102a-d and 104a-c. Each track has sectors, such as sector 106. On a prior art high-density 1.44 MB (formatted capacity) diskette,
20   each track contains 18 sectors (numbered 1-18). Tracks 102a-d each contain 18 sectors, as in the prior art. As a diskette spins, the gaps between sectors provide an opportunity for circuitry within the diskette drive to synchronize itself with the carrier and data signals recorded on the diskette. The size of the gap between the sectors of tracks 102a-d is represented by angle 108. The size of
25   the gap, represented by angle 110, between the sectors of tracks 104a-c is smaller than the size of the gap 108 between the sectors of tracks 102a-d, leaving room for, preferably, two additional sectors per track without negatively impacting the ability of the circuitry to synchronize itself. The gap can be further reduced, leaving room for three additional sectors per track, but with current

technology such small gaps negatively impact the ability of the circuitry to synchronize itself. The additional sectors in track 104c are sector number "19" (112a in Fig. 1) and sector number "20" (112b in Fig. 1). Collectively, all the additional sectors on the diskette, such as sectors 112a-f, make up the hidden

5 partition 114. (Fig. 1 assumes that only tracks 104a-c have additional sectors.)

Not all tracks need contain additional sectors, thus making it difficult for a virus to detect the existence of the hidden partition 114 and the location of the tracks 104a-c that contain additional sectors. The manufacturer of diskettes can vary, from diskette to diskette, which tracks have additional sectors. Preferably,

10 all tracks contain two additional sectors, providing a total of 160KB in the hidden partition of an 80-track diskette. Advantageously, these two additional sectors at the end of each track provide a rotational delay between the last conventional sector (e.g., sector number "18") of a track and the first sector of the next track. This delay allows the diskette drive enough time to reposition the

15 head on the next track before the first sector of the next track passes under the head, thereby speeding up access to data on the diskette. Without this delay, the first sector of the next track would pass under the head before the diskette drive could reposition the head and, once the drive repositions the head, the drive would incur a time penalty of nearly one revolution before it could read the

20 first sector in the next track.

Table 1 lists the fields of the BIOS Parameter Block that are relevant to the invention. The BIOS Parameter Block located on diskette contains the value "18" in the "Number of sectors per track" field. The "Total number of sectors in volume (logical partition)" field contains the product of 18, the number of tracks

25 per surface and the number of surfaces on the diskette. Thus, the BIOS Parameter Block located on the diskette hides the additional sectors 112a-f, therefore making the hidden partition 114 inaccessible to conventional programs.

Fig. 2 shows generally at 200 a surface of a hard disk according to the

30 invention, including representative tracks 202a-e and representative sectors, such as sector 204. If the hard disk does not have spare sectors and/or spare

tracks, or if it is not desirable to utilize these spares to form a hidden partition, the hard disk can be formatted so that the BIOS Parameter Block on the disk does not reflect the existence of selected sectors. For example, by reducing the "Number of sectors per track" field in the BIOS Parameter Block by two and

5   reducing the "Total number of sectors in volume (logical partition)" field by the product of two, the number of tracks per surface and the number of surfaces on the disk, sector number "681" and sector number "682" in each track 202a-e, i.e., sectors 206a-j, become part of the hidden partition 208, and therefore inaccessible to conventional programs. However, I prefer to create a hidden

10   partition on the last sectors of a disk by simply reducing the "Total number of sectors in volume (logical partition)" field in the BIOS Parameter Block because this enables the creation of a hidden partition of arbitrary size.

Fig. 3 is a flowchart illustrating a bootstrap sequence involving a disk provided by the invention. Upon application of power or a reset signal, the

15   computer starts at step 300 performs a power-on self-test ("POST") at step 301 and executes the special bootstrap program at step 302.

At decision step 304, the special bootstrap program ascertains whether the computer was started from a diskette or from a hard disk. If the computer was started from a diskette, at step 306 the special startup program warns the

20   user of the risk of propagating a virus inherent in starting the computer from a diskette. The risk exists because a diskette that was previously inserted in another computer may have been infected by a virus running on the other computer, even if the diskette was not used to start the other computer.

At step 308 the special startup program performs an integrity test on itself

25   to ascertain whether it has been damaged, e.g., by a virus attack, and announces the results of the self-test.

At step 310 the special startup program causes the computer to execute the BTOS, which is stored in the hidden partition 114 or 208. At step 312 the BTOS checks for indication of the existence of viruses in the computer and

30   announces the results of the check. For example, the BTOS examines the disk vector, as described earlier. At step 314 the BTOS then causes the computer to

load and execute virus removal and integrity check software, which is stored in the hidden partition 114 or 208.

At decision step 315, if all detected viruses have been removed, the software checks at decision step 316 whether the special bootstrap program in the boot block had been attacked. If so, at step 317 the software copies one of several backup copies of the special bootstrap program, which are stored in the hidden partition 114 or 208, to the boot block.

In addition to being capable of detecting the existence of viruses in the computer, the BTOS is an operating system and can provide services to application programs ("compliant programs") written to make program calls to it. At step 318 the BTOS causes the computer to search the hidden partition for compliant programs, such as third-party anti-virus, disk-maintenance or other software, and to load and execute any such programs it finds. Alternatively, the BTOS can search for compliant programs on portions of the disk in addition to the hidden partition, but I prefer to store all such software on the hidden partition so it does not reduce the conventional storage capacity of the disk and to make it inaccessible to conventional viruses.

At step 320 the BTOS restores the memory and registers of the computer to an "initialization" state, i.e., substantially the state that existed immediately after the POST. This state is well known, particularly to practitioners who write BIOS programs, and the step is necessary to enable the normal bootstrap program to execute correctly. For example, when a virus modifies the disk vector, the virus stores within itself the original contents of the vector so the virus can pass control to the disk handling routine after performing its mischief. When the anti-virus software detects such a modified vector, it examines the virus, retrieves the original contents of the vector and restores the vector's original contents. The location at which a virus stores the original contents can be ascertained by "disassembling" the virus. Disassembly is a well-known process of converting object code to source code.

As a second example, the anti-virus software recovers memory utilized by a virus when it removes the virus. Viruses often mask their presence by

reducing the apparent amount of memory on the computer by an amount equal to their size (including buffers, etc.), e.g., by reducing a field in the BIOS Data Area that contains the number of kilobytes of memory that exist below memory address 640KB. Generally, an ultimate operating system examines the BIOS

5   Data Area when it begins executing and configures itself accordingly. Application program anti-virus software removes a virus <u>after</u> the ultimate operating system begins executing, thus after the ultimate operating system has examined the BIOS Data Area, so despite having removed the virus the application program anti-virus software does not enable the ultimate operating

10   *system to utilize the memory previously utilized by the virus. Anti-virus software provided by the invention executes <u>before</u> the ultimate operating system so, after removing a virus, it increases the field in the BIOS Data Area by the amount of memory recovered, thereby enabling the ultimate operating system to utilize the recovered memory.

15   At step 322 the BTOS causes the computer to execute the normal bootstrap program and then finishes at step 324. The normal bootstrap program causes the computer to execute the ultimate operating system.

        Several viruses are known to attack certain sectors and certain clusters of a disk. In case a virus attacks a disk having the anti-virus capabilities just

20   described, these sectors and clusters are preferably allocated to a "dummy" file, thereby preventing their being allocated to any other files or used by directories, thus preventing these viruses from inflicting any damage on the files or directories. Some viruses attacked the two sectors of a disk that generally contain the last two sectors of the root directory. Reducing the size of the root

25   directory by two sectors (see Table 1, "Number of entries in root directory" field of the BIOS Parameter Block) prevents this type of attack on the root directory. Preferably, the dummy file includes the first two sectors removed from the root directory, i.e., the first two clusters on the disk after the root directory, and the cluster comprising the last track on the last surface. (Assuming the cluster size

30   is equal to one.) The dummy file, named "ViToler8.(c)", has an entry in the directory and entries, corresponding to the above-listed sectors and clusters, in

the FAT. The directory entry marks the file as "hidden" in the MS-DOS sense. MS-DOS does not list hidden files when it provides a directory listing of files, so users will not generally know the file exists. While users can remove the hidden attribute from the file, marking the file hidden reduces the possibility that a user

5   will delete the file and thereby make the virus-targeted clusters and sectors available for allocation to other files.

While I prefer to use a hidden partition to store the anti-virus, disk-maintenance and other software and the BTOS, it is also possible to utilize a disk with no hidden partition and store any of these within the conventional

10   storage capacity of the disk. In such a case, the special bootstrap program causes the computer to execute the anti-virus software or BTOS and then, if the disk is a startup disk, to execute the normal bootstrap program. When the anti-virus, disk-maintenance, or other software or the BTOS is stored within the conventional storage capacity of the disk, its files should be marked as hidden in

15   the MS-DOS sense to reduce the possibility that a user will delete them or that a virus will attack them.

It will therefore be seen that I have developed a virus-tolerant and virus-resistant disk and method of executing software prior to an ultimate operating system, which can be utilized with a variety of anti-virus, disk-maintenance, and other

20   software to address a range of computer viruses. The terms and expressions employed herein are used as terms of description and not of limitation, and there is no intention, in the use of such terms and expressions, of excluding any equivalents of the features shown and described or portions thereof, but it is recognized that various modifications are possible within the scope of the

25   invention claimed.

PROGRAM-LISTING APPENDIX

```
    --
    * * * * * * * * * * * * * * * * * *
    * FILE TUNING.INC *
    * * * * * * * * * * * * * * * * * *

    ; TUNING.INC CREATED BY MAKEFILE
    DQSector    EQU    016H
    DQCylinder  EQU    002H
    RETRY       EQU    05H
```

```
* * * * * * * * * * * * * * * *
*  FILE BOOT.INC  *
* * * * * * * * * * * * * * * *

;CORE.INC (tm)
;Copyright(c)1995 OverByte Corporation
;-------------------------------------------------------

FARFUN    MACRO    n
n&_funptr          LABEL    DWORD
n&_offset          LABEL    WORD
          DW       OFFSET n
          DW       0
          ENDM


;-------------------------------------------------------
; The SLACK macro calculates the space used up by the
program
; relative to the space it is permitted to use up.
Ideally
; this should be of size 0.
;-------------------------------------------------------
TELLSIZE          MACRO    `COUNT,RELATIVE,MAX,MSG
IFDEF MODULENAME
%OUT    %(MODULENAME) &COUNT(dec) RELATIVE than MAX(hex)
MSG
ELSE
%OUT    CODE &COUNT(dec) RELATIVE than MAX(hex) MSG
ENDIF
          ENDM


SLACK    MACRO    COUNT,MAX,MSG,MN
          LOCAL    diff
IF2
IF ( &COUNT LT 0 ) OR ( &COUNT GT MAX )
diff      EQU      ( 65535 - &COUNT + 1 )
          TELLSIZE %(diff),larger,MAX,MSG
.ERR
ELSE
          TELLSIZE COUNT,smaller,MAX,MSG
ENDIF
ENDIF
          ENDM

* * * * * * * * * * * * * * * *
*  FILE: BOOT.ASM  *
* * * * * * * * * * * * * * * *

;BOOT (tm)
```

17

```
;Copyright(c)1995 OverByte Corporation
;---------------------------------------------------------

;MODULENAME EQU "BOOT"

INCLUDE BOOT.INC
INCLUDE TUNING.INC

STACK    SEGMENT PARA stack 'STACK'
STACK    ENDS
BOOT     SEGMENT BYTE PUBLIC 'CODE'
         ASSUME  CS:BOOT
         ORG     0
begin:
         JMP SHORT simple
         NOP
         DB       "OverByte"        ; OEM
BPB      DW       200H              ; Bytes per sector
         DB       1                 ; Sectors per cluster
         DW       1                 ; Reserved sectors
         DB       2                 ; Number of FATs
         DW       0E0H              ; Number of ROOT DIR
entries
         DW       0B40H             ; Number of sectors on
diskette
         DB       0F0H              ; Media Descriptor Byte
         DW       9                 ; Number of sectors per
FAT
         DW       12H               ; Number of sectors per
track
         DW       2                 ; Number of heads
         DD       0                 ; Number of hidden
sectors
EBPB     DD       0
         DB       0
         DB       0
         DB       0
         DD       0
         DB       "OverBoot tm"     ; Volume label
         DB       "FAT12    "       ; File system type
         DW       0                 ; Leave 2 bytes unused
         ; 64 BYTE mark
simple: MOV      BX,CS              ; BOOT is always loaded
at
                                    ; CS:7C00
         ADD      BX,780H           ; Where at least 1K is
free
; to be used
         MOV      SS,BX             ; at CS:7800 through
CS:7C00
         MOV      SP,400H           ; Because the stack will
be put
```

18

```
                ADD       BX,40H              ; directly underneath
CS:7C00
                PUSH      BX                  ; and execution will
proceed at
                MOV       BX,OFFSET sevenc; (CS+7C0):0
                PUSH      BX
                RETF
sevenc: PUSH    CS                          ; Operational DATA is in
this
                                            ; segment
                POP       DS
                ASSUME    DS:BOOT
                MOV       myCS,CS             ; Adjust segment so DATA
starts
                                            ; at OFFSET 0
                SUB       myCS,7C0H
                MOV       SI,OFFSET worry    ; Message asking wait
while
                                            ; DisQuick loads
                CALL      show
                CALL      fast                ; Load DisQuick
                MOV       SI,OFFSET message
                JC        badness             ; If load failed, show
Invalid
                MOV       SI,OFFSET diskok; Otherwise show OK
badness:CALL    show
                XOR       AX,AX
                INT       16H                 ; Wait for a key
                MOV       AX,CS:myCS
                OR        AX,AX               ; Check for segment
value
                JZ        reboot
                MOV       AX,4C01H            ; If run from DOS, don't
                                            ; reboot, but EXIT
                INT       21H
                INT       20H
reboot: MOV     ES,AX
                MOV       WORD PTR ES:[472H],0
                DB        0EAH,0,0,0FFH,0FFH


worry   DB      0dh,0ah,"DisQuick(tm) diskette load"
        DB      0dh,0ah,"Please wait.",0
message DB      0dh,0ah,"Invalid "
diskok  DB      "DisQuick(tm) diskette"
        DB      0dh,0ah,"Press any key",0
nomem   DB      0dh,0ah,"RAM too small",0
nodsk   DB      0dh,0ah,"DISK failed",0
noapp   DB      0dh,0ah,"Bad DisQuick",0
myNO    DB      0FFH
myCX    DW      (DQSector + ( DQCylinder SHL 8 ) )
myDX    DW      0
myIP    DW      0
myES    DW      0
```

```
myCS        DW        0

show        PROC      NEAR
            CLD
more:       LODSB
            OR        AL,AL
            JZ        done
            MOV       AH,0EH    ; Use TTY output in BIOS to show
messages
            MOV       BL,0
            INT       10H
            JMP SHORT more
done:       RET
show        ENDP


fast        PROC      NEAR
            PUSH      AX
            PUSH      BX
            PUSH      CX
            PUSH      DX
            PUSH      BP
            PUSH      SI
            PUSH      DI
            PUSH      ES

            MOV       BX,40H
            MOV       ES,BX
            MOV       BX,13H
            MOV       BX,ES:[BX]        ; Ask BIOS how much RAM
this PC
                                       ; has
            CMP       BX,1EAH          ; 512K minus a virus or
two
            JAE       memok
            MOV       SI,OFFSET nomem  ; Can't load DisQuick if
not
                                       ; enough RAM
            CALL      show
            JMP       ignore

memok:      MOV       AX,CS
            ADD       AX,0003FH
            AND       AX,0FFC0H
            ADD       AX,40H
            MOV       ES,AX            ; align load area at
next 1K
                                       ; boundary
            ASSUME    ES:NOTHING
            MOV       myES,ES

next:       MOV       CX,RETRY
again:      PUSH      CX
```

20

```
          MOV       DX,myDX          ; Using location of
current
                                     ; chunk to load
          MOV       CX,myCX
          XOR       BX,BX
          MOV       AX,202H          ; Keep loading 1K chunks
from
                                     ; hidden partition
          INT       13H

          POP       CX
          JNC       good
          PUSH      CX
          XOR       AX,AX
          XOR       DX,DX
          INT       13H
          POP       CX
          LOOP      again            ; Until all of DisQuick
is
                                     ; loaded
          MOV       SI,OFFSET nodsk
          CALL      show
          JMP SHORT ignore

good:     MOV       DL,myNO
          CMP       DL,0FFH
          JNE       count
          PUSH      BX
          XOR       BX,BX
          MOV       DX,ES:[BX+2]     ; This will be Kilobytes
in APP
use:      POP       BX
          MOV       myNO,DL
count:    MOV       DX,myDX          ; Adjust to location of
next
                                     ; chunk to load
          MOV       CX,myCX
          INC       DH
          AND       DH,1
          JNZ       head
          INC       CH
head:
          MOV       myCX,CX
          MOV       myDX,DX

          MOV       AX,ES
          ADD       AX,40H
          MOV       ES,AX
          ASSUME    ES:NOTHING

          DEC       myNO
          MOV       AL,myNO
          OR        AL,AL
```

```
          JNZ     next           ; Go fetch the next chunk

          ; Launch
          LES     BX,DWORD PTR myIP
          CMP     WORD PTR ES:[BX],002EBH
          JE      launch
          MOV     SI,OFFSET noapp
          CALL    show
          JE      ignore
launch:   PUSH    myCS
          CALL    DWORD PTR myIP  ; Execute DisQuick
          ADD     SP,2
          PUSH    CS
          POP     DS
          ASSUME  DS:BOOT
          CLC
          JMP SHORT success
ignore:   STC
success:  POP     ES
          POP     DI
          POP     SI
          POP     BP
          POP     DX
          POP     CX
          POP     BX
          POP     AX
          RET
fast      ENDP
codeend:
          ORG     1FEH
bootend:
          DW      0AA55H

IF 0
BOOTSIZE EQU ( ( OFFSET CS:bootend ) - ( OFFSET
CS:codeend ) )
SLACK %(BOOTSIZE),0200H,"from end of BOOT code to AA55"
ENDIF

BOOT      ENDS
          END     begin
--
```

22

CLAIMS

1   1.   A virus-resistant storage medium for use with a computer, the starage

2   medium being selectably actuable by a user and comprising:

3   (a)   anti-virus software stored on the storage medium and

4   configured to cause the computer to detect and remove from a

5   component thereof at least one virus; and

6   (b)   means for causing the computer to execute the anti-virus

7   software upon startup of the computer but before the computer

8   executes an ultimate operating system.

1   2.   The virus-resistant storage medium of claim 1 wherein the medium is

2   a flash ROM.

1   3.   The virus-resistant storage medium of claim 1 wherein the medium is

2   a bubble RAM.

1   4.   The virus-resistant storage medium of claim 1 wherein the medium is

2   an optical storage device.

1   5.   The virus-resistant storage medium of claim 1 wherein the medium is

2   a tape drive.

1   6.   The virus-resistant storage medium of claim 1 wherein the medium is

2   a disk.

1   7.   The virus-resistant storage medium of claim 1, wherein the execution-

2   causing means comprises bootstrap software stored on the medium,

3   execution of the bootstrap software causing the computer to execute

4   the anti-virus software prior to post-bootstrap operations, the

5   bootstrap software being executed upon startup of the computer.

1　8.　　The virus-resistant storage medium of claim 1, wherein:

2　　　　(a)　　the computer comprises memory and registers;

3　　　　(b)　　the computer has an initialization state in which the memory

4　　　　　　　　and registers of the computer accurately reflect the amount of

5　　　　　　　　memory on the computer and no interrupt vector in the

6　　　　　　　　computer contains an address of a routine within a virus; and

7　　　　(c)　　the anti-virus software is configured to cause the computer,

8　　　　　　　　after detecting and removing a virus, to enter the initialization

9　　　　　　　　state.

1　9.　　A virus-resistant storage medium for use with a computer, the storage

2　　　　medium having a conventional storage capacity and comprising:

3　　　　(a)　　a hidden partition comprising storage space inaccessible to

4　　　　　　　　conventional programs and not reducing the conventional

5　　　　　　　　storage capacity;

6　　　　(b)　　means for describing the conventional storage capacity but not

7　　　　　　　　describing the hidden partition, thereby making the hidden

8　　　　　　　　partition inaccessible to conventional programs;

9　　　　(c)　　anti-virus software stored in the hidden partition and

10　　　　　　　　configured to cause the computer to detect and remove from a

11　　　　　　　　component thereof at least one virus; and

12　　　　(d)　　means for causing the computer to execute the anti-virus

13　　　　　　　　software.

1　10.　The virus-resistant storage medium of claim 9, wherein the execution-

2　　　　causing means comprises bootstrap software stored on the medium,

3　　　　execution of the bootstrap software upon startup of the computer

4　　　　causing the computer to execute the anti-virus software prior to post-

5　　　　bootstrap operations.

1   11.   The virus-resistant storage medium of claim 9, wherein the anti-virus

2   software comprises at least one anti-virus program and the execution-

3   causing means comprises:

4   (d1)   bootstrap software stored on the storage medium; and

5   (d2)   a bootstrap-time operating system stored in the hidden

6   partition for causing the computer, upon startup, to

7   automatically execute the bootstrap software, said bootstrap-

8   software execution causing the computer to execute the

9   bootstrap-time operating system, said operating-system

10   execution causing the computer to execute the at least one

11   anti-virus program prior to post-bootstrap operations.

1   12.   The virus-resistant storage medium of claim 11 further comprising:

2   (e)   maintenance software stored in the hidden partition for causing

3   the computer to detect and repair damage to information

4   stored on a component thereof; and wherein

5   (f)   the bootstrap-time operating system causes the computer to

6   execute the maintenance software prior to post-bootstrap

7   operations.

1   13.   The virus-resistant medium defined in claim 9, wherein the execution-

2   causing means comprises an application program stored on the

3   medium.

1   14.   The virus-resistant medium defined in claim 9, wherein the medium

2   has a plurality of tracks on each of at least one surface and the

3   hidden partition comprises storage space on fewer than all the tracks.

25

1 15. The virus-resistant medium defined in claim 9, wherein the disk has a
2   plurality of tracks on each of at least one surface and the hidden
3   partition comprises storage space on all the tracks.

1 16. The virus-resistant medium defined in claim 9, wherein:
2   (a) the computer comprises memory and registers;
3   (b) the computer has an initialization state in which the memory
4   and registers of the computer accurately reflect the amount of
5   memory on the computer and no interrupt vector in the
6   computer contains an address of a routine within a virus; and
7   (c) the anti-virus software is configured to cause the computer,
8   after detecting and removing a virus, to enter the initialization
9   state.

1 17. A storage medium for use with a computer, the storage medium being
2   selectably actuable by a user and comprising:
3   (a) maintenance software for causing the computer to detect and
4   repair damage to information stored on a component thereof;
5   and
6   (b) means for causing the computer to execute the software upon
7   startup of the computer but before the computer executes an
8   ultimate operating system.

1 18. A storage medium for use with a computer, the storage medium
2   having a conventional storage capacity and comprising:
3   (a) a hidden partition comprising storage space inaccessible to
4   conventional programs and not reducing the conventional
5   storage capacity;
6   (b) means for describing the conventional storage capacity but not
7   describing the hidden partition, thereby making the hidden
8   partition inaccessible to conventional programs;

9   (c)   maintenance software stored in the hidden partition for causing

10   the computer to detect and repair damage to information

11   stored on a component thereof; and

12   (d)   means for causing the computer to execute the software.

1 19. A method of operating a computer to avoid virus infection, the method

2   comprising:

3   (a)   providing a storage medium selectably actuable by a user;

4   (b)   storing, on the storage medium, anti-virus software configured

5   to cause the computer to detect and remove from a component

6   thereof at least one virus; and

7   (c)   for causing the computer to execute the anti-virus software

8   upon startup of the computer but before the computer executes

9   an ultimate operating system.

1 20. The method of claim 21 wherein the medium is a flash ROM.

1 21. The method of claim 21 wherein the medium is a bubble RAM.

1 22. The method of claim 21 wherein the medium is an optical storage

2   device.

1 23. The method of claim 21 wherein the medium is a tape drive.

1 24. The method of claim 21 wherein the medium is a network disk.

1 25. A method of causing a computer to execute at least one program

2   upon startup of the computer but before the computer executes an

3   ultimate operating system, the method comprising the steps:

4   (a)   providing a startup disk comprising a hidden partition

5   comprising storage space inaccessible to conventional

6   programs and not reducing conventional storage capacity of

7   the disk;

8   (b)   providing a bootstrap-time operating system stored in the

9   hidden partition;

10   (c)   causing the computer to execute the bootstrap-time operating

11   system upon startup of the computer but before the computer

12   executes the ultimate operating system;

13   (d)   causing the computer to execute the at least one program

14   under the control of the bootstrap-time operating system; and

15   (e)   upon completion of the at least one program, executing the

16   ultimate operating system.

1   26.   The method defined in claim 25, wherein the at least one program is

2   anti-virus software stored in the hidden partition and configured to

3   cause the computer to detect and remove from a component thereof

4   at least one virus.

1   27.   The method defined in claim 26, wherein:

2   (a)   the computer comprises memory and registers;

3   (b)   the computer has an initialization state in which the memory

4   and registers of the computer accurately reflect the amount of

5   memory on the computer and no interrupt vector in the

6   computer contains an address of a routine within a virus; and

7   (c)   the anti-virus software is configured to cause the computer,

8   after detecting and removing a virus, enter the initialization

9   state.

1   28.   The method defined in claim 27, wherein the at least one program is

2   disk-maintenance software stored in the hidden partition and

3   configured to cause the computer to detect and repair damage to

4   information stored on a component thereof.

1   29.   A virus-tolerant disk having sectors and a file system allocation table

2   for allocating the sectors to files, the virus-tolerant disk comprising:

3   (a)   a predetermined subset of the sectors, comprising sectors

4   attacked by at least one virus; and

5   (b)   entries, corresponding to the predetermined sectors, in the file

6   system allocation table for making the predetermined sectors

7   unavailable for allocation to any file, the entries being made

8   prior to an attack by a virus.

1   30.   A virus-resistant storage medium for use with a computer, the storage

2   medium having a conventional storage capacity and comprising:

3   (a)   a hidden partition comprising storage space inaccessible to

4   conventional programs and not reducing the conventional

5   storage capacity;

6   (b)   means for describing the conventional storage capacity but not

7   describing the hidden partition, thereby making the hidden

8   partition inaccessible to conventional programs;

9   (c)   anti-virus software stored in the hidden partition and

10   configured to cause the computer to detect and remove from a

11   component thereof at least one virus; and

12   (d)   means for causing the computer to execute the anti-virus

13   software.

1   31.   The virus-resistant storage medium of claim 30, wherein the

2   execution-causing means comprises bootstrap software stored on the

3   disk, execution of the bootstrap software upon startup of the

4   computer causing the computer to execute the anti-virus software

5   prior to post-bootstrap operations.

1   32.   The virus-resistant storage medium of claim 30, wherein the anti-virus

2   software comprises at least one anti-virus program and the execution-

3   causing means comprises:

4   (d1)   bootstrap software stored on the storage medium; and

5   (d2)   a bootstrap-time operating system stored in the hidden

6   partition for causing the computer, upon startup, to

7   automatically execute the bootstrap software, said bootstrap-

8   software execution causing the computer to execute the

9   bootstrap-time operating system, said operating-system

10   execution causing the computer to execute the at least one
11   anti-virus program prior to post-bootstrap operations.

1   33.   The virus-resistant storage medium of claim 32 further comprising:
2   (e)   maintenance software stored in the hidden partition for
3   causing the computer to detect and repair damage to
4   information stored on a component thereof; and wherein
5   (f)   the bootstrap-time operating system causes the computer to
6   execute the maintenance software prior to post-bootstrap
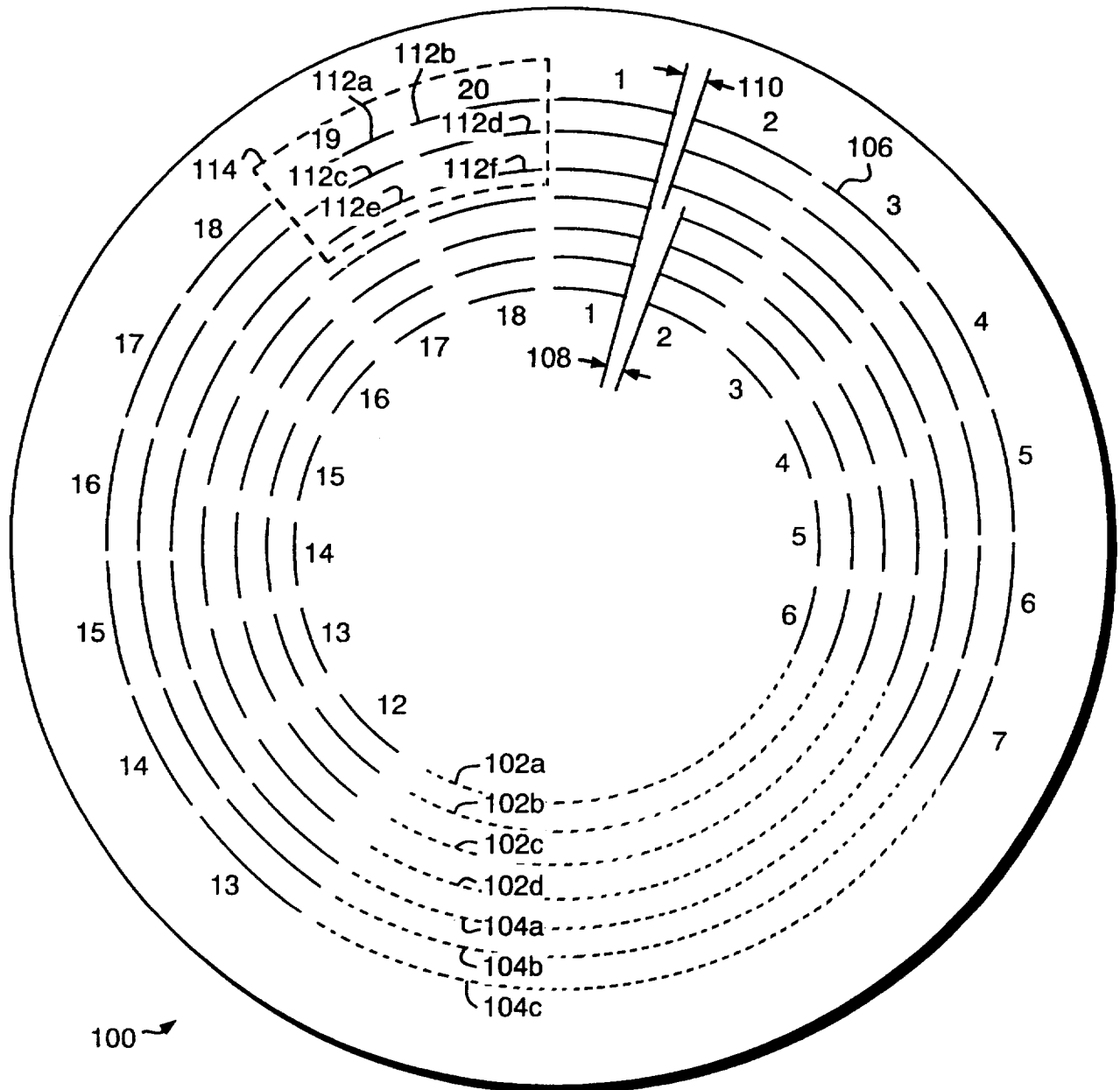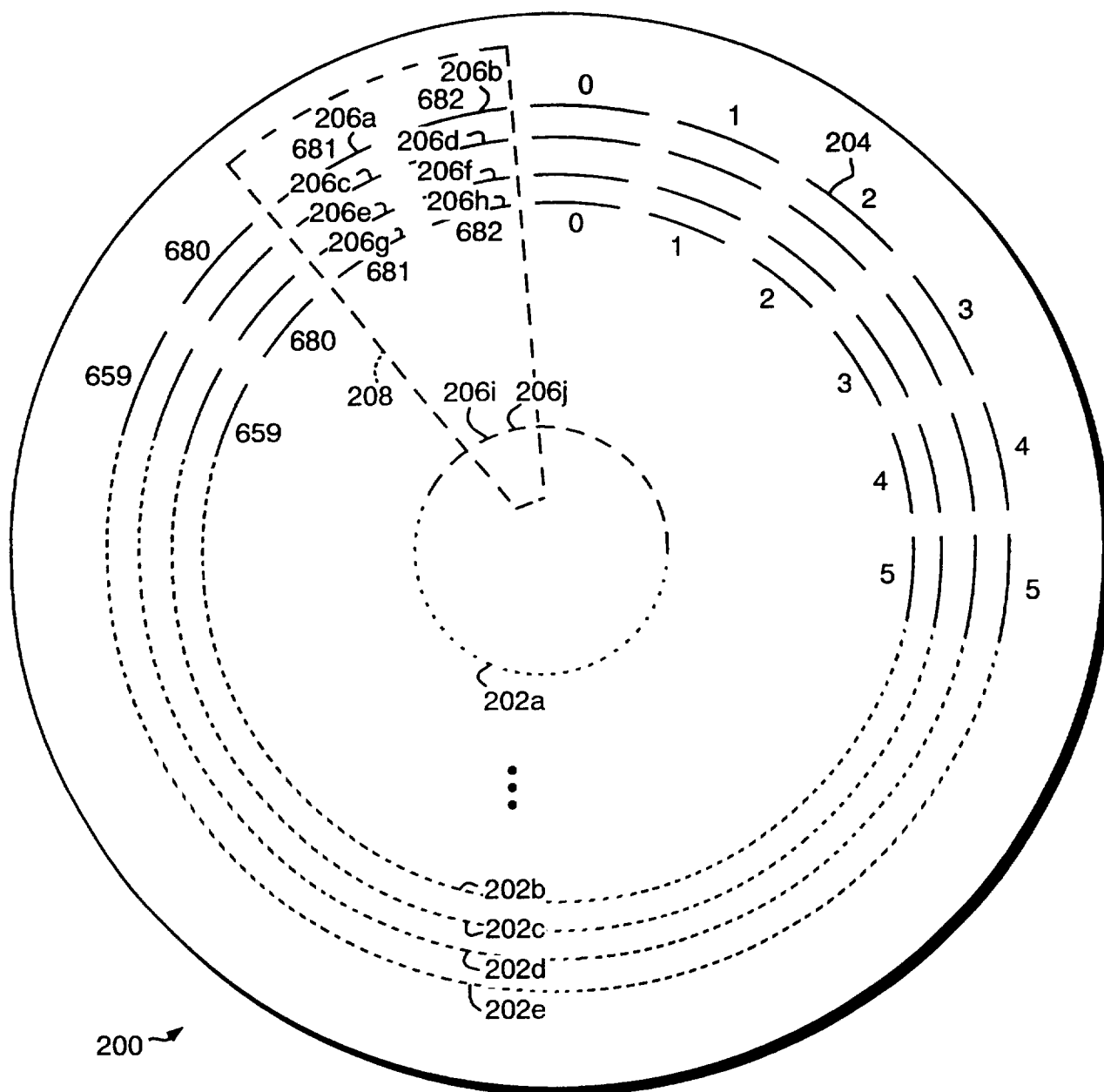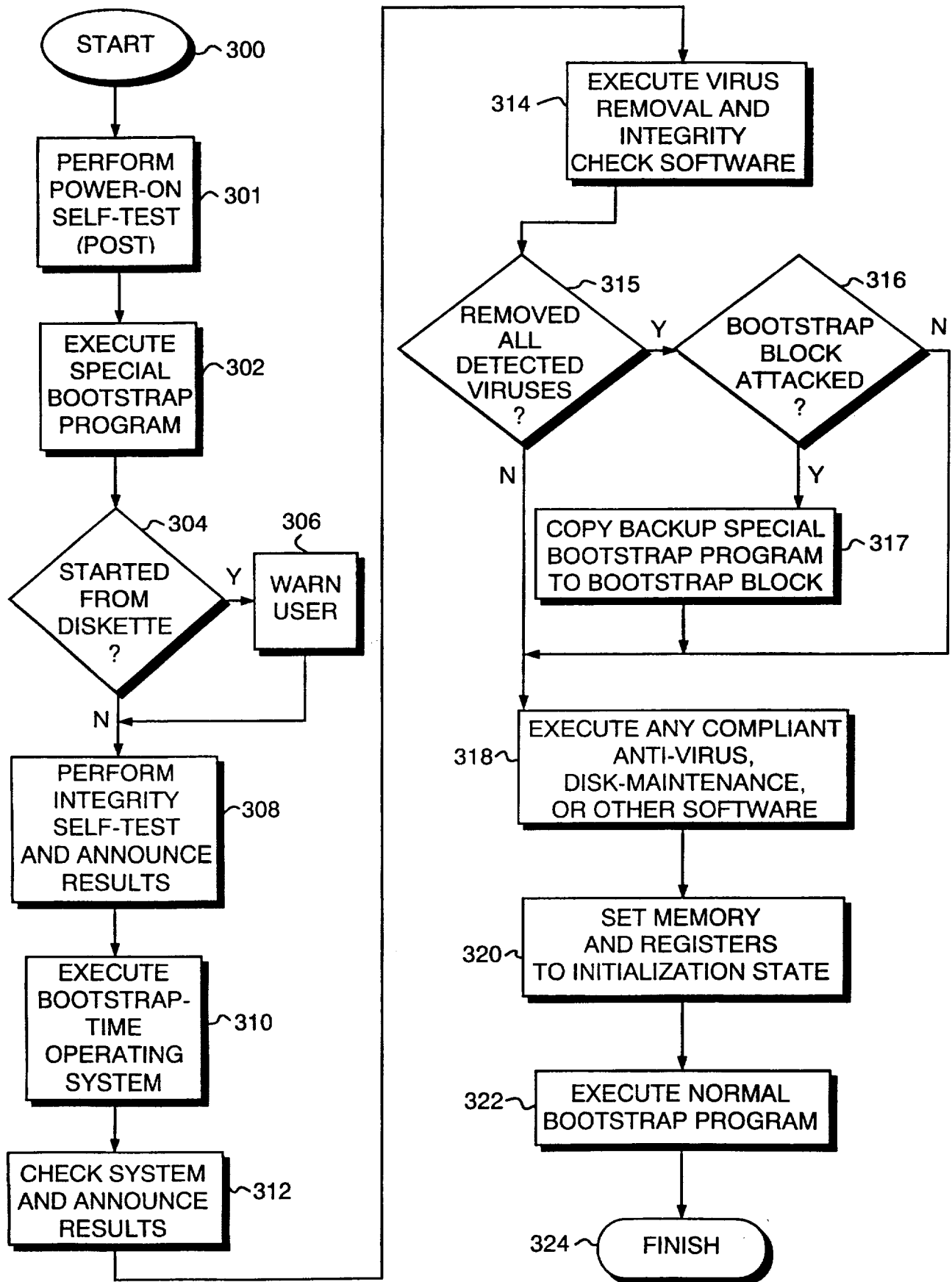7   operations.

FIG. 1

FIG. 2

FIG. 3

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US97/02345

A.	CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 11/00
US CL :395/186

According to International Patent Classification (IPC) or to both national classification and IPC

B.	FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/185.01, 186; 364/286.4, 286.5; 380/4,25

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Super Mater (User's Manual)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, DIALOG

C.	DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X ---- Y	HITECH BUSINESS SOFTWARE INC (1992), selected section Chapter 3 and 7	1,7, 8, 17,19, 29 ---------------- 2-6,20-24
Y	US 5,359,659 A (ROSENTHAL) 25 October 1994, see abstract; figure 3; col. 7, line 35 and et seq.	1-33
Y	US 5,398,196 A (CHAMBERS) 14 March 1995, col. 1, line 65 to col. 4, line 14	1-33
Y	US 5,440,723 A (ARNOLD et al.) 08 August 1995, see the abstract; col. 2, line 44 to col. 3, line 12	1-33

X	Further documents are listed in the continuation of Box C.	☐	See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)		
		"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 APRIL 1997	02 JUN 1997

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	ALBERT DECADY
Facsimile No. (703) 305-3230	Telephone No. (703) 308-3900

Form PCT/ISA/210 (second sheet)(July 1992)*

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,421,006 A (JABLON et al.) 30 May 1995, see the abstract; see figures 9 and 10; also col. 7, line 55 to col. 9, line 54	1-33
Y	US 4,975,950 A (LENTZ) 04 December 1990, see the abstract; also figures 1and 2	1-33
X --- Y	US 5,086,502 A (MALCOLM) 04 February 1992, see the abstract; tables 1-4	29 ------- 1-28, 30-33
Y	US 5,121,345 A (LENTZ) 09 June 1992, see the abstract; also figures 1 and 2; col. 2, lines 46-58	1-33
A	US 5,414,833 A (HERSHEY et al.) 09 May 1995, see the abstract	1-33
A	US 5,483,649 A (KUZNETSOV et al.) 09 January 1996, see the abstract; figure 8	1-33
A	US 5,144,660 A (ROSE) 01 September 1992, see col. 1, line 55 to col. 2, line 17	1-33