PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 11/00 — A1

(11) International Publication Number: WO 97/16786

(43) International Publication Date: 9 May 1997 (09.05.97)

(21) International Application Number: PCT/FI96/00572

(22) International Filing Date: 29 October 1996 (29.10.96)

(30) Priority Data:
955187 30 October 1995 (30.10.95) FI

(71) Applicant *(for all designated States except US)*: NOKIA TELECOMMUNICATIONS OY [FI/FI]; Upseerinkatu 1, FIN-02600 Espoo (FI).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: RINNE, Jarmo [FI/FI]; Nikkarinkuja 2 C 47, FIN-02600 Espoo (FI). PASANEN, Kari [FI/FI]; Palokatu 6 A, FIN-44100 Äänekoski (FI).

(74) Agent: OY KOLSTER AB; Iso Roobertinkatu 23, P.O. Box 148, FIN-00121 Helsinki (FI).

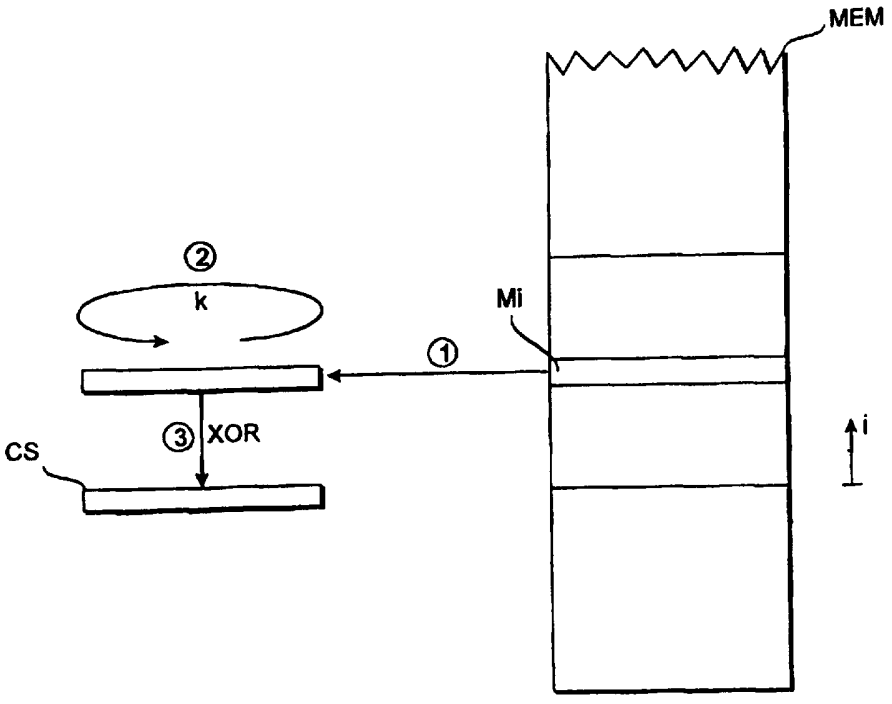
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.
In English translation (filed in Finnish).

(54) Title: DETECTING MEMORY PROBLEMS IN COMPUTERS

(57) Abstract

In process controlling computers, such as switching exchanges, faults may prove to be very expensive. Particularly hard to solve are software defects. If problems occur, it is important that maintenance can be directed at the correct target as soon as possible. A threat is constituted by vandals and hackers. If the option for maintaining from elsewhere is incorporated in computers, the danger for misuse consequently increases. Prior art techniques for detecting memory problems do not necessarily reveal defects occurring at regular intervals caused by the method of calculation. Memory defects will occur at regular intervals e.g., if an address line of a memory circuit becomes faulty. A second drawback of the prior art is that the checksum has to be calculated anew if the contents of just one storage location change. The method of detecting memory problems according to the invention is "regularly irregular" so that it detects in a reliable manner an uncontrolled change in any storage area (MA). Furthermore, the checksum (CS) calculated according to the method may be updated upon changes of an element in a storage area (MA) so that the updating requires much less calculation than would calculating the checksum over the entire storage area (MA) being monitored. The method is applicable in detecting hardware and software defects and in expediting the restart of a computer.



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.

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

DETECTING MEMORY PROBLEMS IN COMPUTERS

The present invention relates to checking the integrity of memory contents in computers.

Figure 1 shows the parts of a computer that are essential as far as the invention is concerned. At the initial stage, after power switch-on, the contents of the main memory RAM are in an undefined state. The initial loading of a conventional computer takes place so that power switch-on generates an initial signal INIT which makes e.g. a processor CPU jump into a predetermined address. This memory has, in a Read Only Memory ROM, an initial loading program the execution of which comprises loading the basic parts of the operating system (OS) from a predetermined location in a disk drive HD. The OS basic parts thus loaded contain more advanced loading programs, which load parts of the OS not only from the locations predetermined for the HD in the ROM but from elsewhere as well. These more advanced loading programs, in turn, load other software modules etc.

The software modules are not necessarily located at contiguous areas on the disk drive HD. Upon reading from the HD, it is often the case that information located at different areas of the HD must be read in succession. A suggestive guideline value is that it is possible to read approximately 1 MB/s of contiguous areas from a disk drive, but in practice the reading rate is no more than one half of this. To take an example, loading 30 MB of software may take approximately a minute.

In process controlling computers, such as switching exchanges, faults may prove to be very expensive. Particularly hard to solve are software defects. Hardware defects can often be detected by making comparisons to another apparatus, known to be in a proper working order. Software defects cannot usually be detected in this manner as there usually exists no reference software known to be operating properly. In many cases, the further drawback is often encountered that on the basis of the symptoms it is often impossible to quickly deduct whether the fault is in the hardware or in the software. Uncontrolled changes in a specific storage area may result from either one of the reasons. If problems occur, it is important that maintenance can be directed at the correct target as soon as possible.

A threat is constituted by vandals and "hackers". If the option for

remote maintenance is incorporated in computers, the danger for misuse consequently increases.

In order to detect memory problems, several different solutions have been developed. Prior art knows a so-called parity bit by means of which it is possible to have an even number of 1-bits in the storage location (or, if desired, an odd number). As is well known, the use of parity does not detect a simultaneous change of state by an even number of bits in a storage location.

Another commonly used tool is a Cyclic Redundancy Check, CRC. The CRC sum is a useful tool in detecting errors in serial mode data, such as in disk drives and data links. Relating to matrix data, the CRC sum is faced with two problems. The first of these is that CRC sums cannot be used to reliably detect defects that occur at such intervals that result from the method of calculating the CRC sum. The second problem is associated with the fact that the CRC sum has to be calculated anew if the contents of just one storage location change.

Resulting from the above, it is an object of the present invention to develop a method and apparatus by means of which memory problems of computers can be detected and the aforementioned problems solved. On the one hand, the invention is based on a method which, based on experience gathered up to date, is extensive enough to detect all conceivable uncontrolled changes in the memory, and on the other hand, on employing this method in the various operations of the computer.

The method and the system according to the invention first of all have the advantage of reliably detecting memory problems. This in turn leads to the second advantage, which is that the restart of a computer can be expedited by omitting some operations that in prior art systems are carried out "just in case".

In the following, the invention will be described in greater detail by means of its preferred embodiments, with reference to the accompanying drawings in which:

Figure 1 shows the parts of a computer that are essential as far as the invention is concerned;

Figure 2 illustrates using a checksum for monitoring memory contents;

Figure 3 is a flow chart illustration for maintaining a checksum; and

Figure 4 illustrates calculating a checksum in accordance with the invention.

Regarding the flow chart of Figure 2, the checksum is calculated over a specific storage area, it is compared to the previous checksum calculated, and in case the checksums differ, an error message will be given. In the flow chart of Figure 2, the significant matter is the manner the calculating the checksum is initiated. According to one embodiment of the invention, the calculation takes place at regular intervals. According to another embodiment, the checksum is calculated in association with specific events, such as switching on a computer, modifying programs, etc.

The flow chart of Figure 3 illustrates how writing into memory takes place. In case the contents of a storage area change, the checksum will be corrected by (i) deleting the portion of the changed storage area from the previous checksum, and (ii) by adding, to the checksum thus changed, a portion corresponding to the new contents of the changed storage area.

The simplicity of the flow chart in Figure 3 ensues from that the prior art methods for calculating a checksum cannot be used for deleting the previous portion of a changed storage area from a calculated checksum and replacing it with a portion corresponding to the new contents of the same storage area. Next, referring to Figure 4, a new method according to the present invention will be described for calculating a checksum. Out of the total memory MEM of a computer, the checksum is to be calculated over a storage area MA being monitored. The width M of the checksum (CS) is arbitrary, but a suitable value for the width M equals the word length W of the computer. Prior to the calculation, the checksum is given an initial value, e.g. 0. The calculation takes place so that each element M_i (where i denotes an index for the element within the storage area), having M bits, in a storage area MA is summed bitwise to the checksum CS (by an Exclusive OR operation) so that, prior to summing, the element M_i is rotated k bits to the right (in the direction of the least significant bit). The number k of the rotations is obtained from:

$$k = \sum_{j=0}^{\infty} INT(i / M^j) \qquad (1)$$

where INT denotes an integer. In practice, the summing does not have to be continued forever but only up to the highest value of j at which M^j is lower than or equal to the element index i (or the highest conceivable index value).

The same result can be achieved by the following fast method. The elements of the storage area MA are summed bitwise (Exclusive OR) to the checksum CS. After each summing, the checksum is rotated one bit to the left (in the direction of the most significant bit). The rotating is repeated if i+1 is divisible by M, and again if i+1 is divisible by M^2, etc, until all the conceivable integer powers of M have been gone through. Once all the elements have been summed, the checksum is rotated to the right as many times as it was rotated to the left during the summing.

A noteworthy advantage of this method is that the checksum thus calculated may easily be updated. The updating takes place e.g. as follows:

(1) Summing the data of the changed element to the old data of the element.

(2) Deducting, on the basis of the index of the changed element, how many rotations were made in all at this element when calculating the original checksum.

(3) Rotating the sum to the right to the same extent.

(4) Summing the rotated sum thus obtained to the original checksum.

The result thus obtained is a new checksum updated with regard to the changed storage area. If a storage area having an arbitrary size changes, the checksum can be updated by carrying out just two summings per element. Thus, the new checksum does not have to be calculated over the entire storage area, which would be the case with the cyclic redundancy check (CRC).

The number of rotations at the changed element may be calculated by dividing the element index i by conceivable integer powers of M and by summing the integers of the quotients according to equation (1).

Referring to Table 1, an exemplary calculation of a checksum according to the method of the invention will now be described. For reasons of simplicity, it is assumed that the storage area comprises 16 locations that can be addressed by a 4-bit index. In this example, the data of the storage locations also has a width of 4 bits. Also, the initial value of the checksum is

set to zero. All these assumptions have been made in order to illustrate the invention, and they do not restrict it in any way.

The contents 1001 of the storage location 0 are summed bitwise to the initial value 0000, and as 0+1 is not divisible by the width (4) of the checksum, the sum is rotated once to the left. The new checksum obtained is 0011. To this sum is added the contents 1011 of the storage location 1, and the sum thus obtained is rotated once to the left because 1+1 is not divisible by the width of the ckecksum. After this, the new checksum is 0001. The process continues is this manner up until the entire storage area has been gone through. At storage locations 3, 7 and 11, the checksum is rotated twice because these indexes incremented by one are divisible by the width of the checksum. At storage location 15, there will be three rotations as 15+1 is divisible by both the width of the checksum and its quadrature. At all other storage locations, the checksum is only rotated once. Lastly, the checksum is rotated 21 times in the opposite direction, i.e. to the right. The final checksum obtained is 0111. The step-by-step progress of the calculation is illustrated in Table 1.

Table 1

Address	Data	Old CS	Bit sum	New CS	Rota-tions
0000	1001	0000	1001	0011	1
0001	1011	0011	1000	0001	1
0010	1101	0001	1100	1001	1
0011	0010	1001	1011	1110	2
0100	1011	1110	0101	1010	1
0101	1010	1010	0000	0000	1
0110	0111	0000	0111	1110	1
0111	0010	1110	1100	0011	2
1000	0000	0011	0011	0110	1
1001	0001	0110	0111	1110	1
1010	0111	1110	1001	0011	1
1011	0101	0011	0110	1001	2
1100	1111	1001	0110	1100	1
1101	0011	1100	1111	1111	1
1110	1110	1111	0001	0010	1
1111	0101	0010	0111	1011	3
		1011		1101	-21

Next, it is assumed that the contents of e.g. the storage location 0101 change from the value 1010 to 0110. The bitwise sum of the old and new contents will be 1100. On the basis of index 0101 (=5) the bitwise sum 1100 is rotated 5+1=6 times to the right, whereby the result obtained is 0011. When this is summed to the old checksum, the new checksum obtained is 1110. The same result is also obtained by repeating, from the beginning, the procedures discussed in association with Table 1, but with the contents of the storage location 0101 having the new value, in this case 0110.

The checksum can most advantageously be calculated by a computer program. For this purpose, the following program, in the form of pseudocode resembling Pascal, can be utilized.

```
SUM := initial value;
for I := 0 to N-1 do
    SUM := SUM xor X(I);
    SUM := SUM rol 1;
    if (I + 1) mod M = 0 then
        SUM := SUM rol 1;
        if (I + 1) mod (M*M) = 0 then
            SUM := SUM rol 1;
            if (I + 1) mod (M*M*M) = 0 then
                SUM := SUM rol 1;
                ...
                end;
            end;
        end;
    end;
end;
N1 := N+N / M+N / (M*M)+N / (M*M*M)+N ...;
SUM := SUM ror N1;
end;
```

In the above pseudocode, N denotes the size of the storage area being monitored, xor denotes the bitwise sum (Exclusive OR), mod signifies a modulo operator (remainder), rol 1 means a rotate left, and ror 1

correspondingly a rotate right.

The following pseudocode may be used to update the checksum:

```
SUM := initial value;
for I := 0 to N-1 do
    I1 := I+I / M+I / (M*M)+I / (M*M*M)+I ...;
    SUM := SUM xor (X(I) ror I1);
end;
```

It is obvious that no checksum is able to detect all memory problems. There is an apparent reason for this: as the information contents of the checksum CS are much smaller than that of the storage location MA over which the ckecksum was calculated, the storage location MA inevitably has several possible contents producing the same checksum CS. In practice, however, the defects are detected reliably. An uncontrolled change of the storage location without a concurrent change in the checksum would require a simultaneous change of several bits in several addresses of the storage area according to a specific mathematical regularity. Although practical memory problems often do manifest mathematical regularity, this regularity is not of such nature that memory problems would remain undetected by the method of the invention. In this respect, it could be said that calculating the checksum in accordance with the invention is "regularly irregular".

The method of the invention for calculating a checksum can be applied e.g. as follows: a dedicated checksum is assigned to each storage area being monitored. Upon changing the contents of the storage area, the checksum is also changed accordingly. The integrity of the checksum is monitored e.g. at specific intervals or in association with specific events. Upon detecting that the checksum does not match, the defect is at least reported and other measures required are additionally taken.

A second way of utilizing the checksum calculation method of the invention is to expedite the restart of a computer. Conventional computers in this case load their entire software from a disk drive, a network etc. By utilizing the checksum according to the invention, it is possible to indicate which storage areas have endured free of defects as regards their contents, resulting

in that their contents do not have to be reloaded. In consequence, the start-up of a computer can be expedited by several tens of seconds compared to the prior art technology. Prior to the restart, a checksum CS may be calculated over one or more storage areas MA being monitored, and these checksums may be stored in the main storage. Some systems allow a protection for a part of the memory against later overwriting. In a memory of that type, such checksums may be stored which correspond to different program modules, because once the program module has been loaded into a memory, its contents should not be changed. Alternatively, a predetermined checksum may be associated with each program module stored in the disk drive. During loading the program module, the checksum may first be read out from the disk drive and compared to the checksum calculated over the corresponding storage area. If the checksums are identical, the program module will not have to be loaded anew from the disk drive. The checksums of different modules of software consisting of multiple parts may be stored in a common file the reading of which quickly unravels which modules have undergone the switch-on without defects and which ones should be reloaded from the disk drive.

Although the invention is explained by using computers as examples, the invention is not restricted to computers in a narrow sense but may be applied to all situations where electronic memories are used. Examples of such implementations include computer peripheral devices and communication systems, image and sound memories, data logging systems, control and process automation systems, and various kinds of automatons. It is also obvious to a person skilled in the art that upon advancements in technology the basic idea of the invention can be implemented in various ways. The invention and the embodiments thereof are not restricted to the examples described above but they may vary within the scope of the claims.

Claims

1. A method for calculating an M-bit checksum (CS) over a storage area (MA) comprising memory elements (M_i), characterized by the steps of

- giving the checksum (CS) a predetermined initial value prior to the calculation,

- calculating the checksum (CS) at each memory element (M_i) of the storage area (MA) as a bitwise sum (Exclusive OR), the first factor of the bitwise sum being formed on the basis of the element (M_i) in question and a second factor on the basis of the checksum (CS) value calculated at the preceding element (M_{i-1});

- rotating the bits of the second factor bits a predetermined number prior to calculating the sum.

2. A method as claimed in claim 1, characterized in that

- the second factor is the previous value of the checksum (CS) as such,

- the first factor is formed by rotating the bits of the element (M_i) k bits to the right (in the direction of the least significant bit), where

$$k = \sum_{j=0}^{\infty} INT(i / M^j)$$

in which INT denotes an integer, holds true for the number k of the rotations.

3. A method as claimed in claim 1, characterized in that in the method:

- the first factor is the element (M_i) as such; and

- the second factor is formed by rotating the previous value of the checksum (CS) by

-- determining, at every M's integer power M^n for which $0<M^n<i+1$ holds true, whether the element's (M_i) index incremented by one (i+1) is divisible by the M's integer power M^n in question; and

-- in response to the element (M_i) index incremented by one (i+1) being divisible by the M's integer power M^n in question, rotating the checksum (CS) one bit to the left; and

-- once all the memory elements (M_i) of the storage area (MA) have been gone through, rotating the checksum (CS) to the right as many times as it was in all rotated to the left during calculating the checksum (CS).

4. A method as claimed in claim 1 for updating the checksum (CS) upon changes in the element (M_i) of the storage area (MA), characterized by the method comprising the steps of:

- summing the new data of the changed element (M_i) bitwise to the old data of the same element (M_i);

- deducting, on the basis of the index (i) in the changed element (M_i), how many rotations in all were made at this element (M_i) when calculating the original checksum (CS);

- rotating the checksum to the right for as many bits as it was rotated to the left at this element (M_i) when calculating the original checksum;

- summing the rotated checksum (CS) thus obtained to the original checksum (CS).

5. A method as claimed in any one of the previous claims, characterized in that the method further comprises the steps of:

- calculating a checksum (CS) over at least one storage area;

- comparing the checksum (CS) calculated over the storage area (MA) in question to a checksum calculated over the same storage area (MA) at an earlier stage;

- in response to a mismatch between the checksums calculated at different times over the same storage area (MA), generating a signal indicating this difference.

6. A method as claimed in claim 5, characterized in that the checksum calculated over the storage area (MA) in question is compared periodically at predetermined times to a checksum (CS) calculated at an earlier stage over the same storage area (MA).

7. A method for a fast restart of a computer comprising a disk drive, characterized by the steps of:

- calculating a checksum (CS) over at least one storage area;

- restarting the computer;

- calculating a checksum (CS) over at least one of such storage areas (MA) over which a checksum (CS) has been calculated prior to the restart;

- comparing the checksum (CS) calculated after the restart to a checksum (CS) calculated over the same storage area (MA) prior to the restart;

- in response to a mismatch between the checksum calculated after the restart and the checksum calculated prior to the restart, loading the data corresponding to the storage area in question from the disk drive.

8. A method as claimed in claim 7, characterized in that the value of the checksum (CS) calculated prior to the restart is stored in a main storage.

9. A method as claimed in claim 8, characterized in that the value of the checksum (CS) calculated prior to the restart is stored in a write-protected main storage.

10. A method as claimed in claim 7, characterized in that the value of the checksum (CS) calculated prior to the restart is stored on a hard disk.

Fig. 1

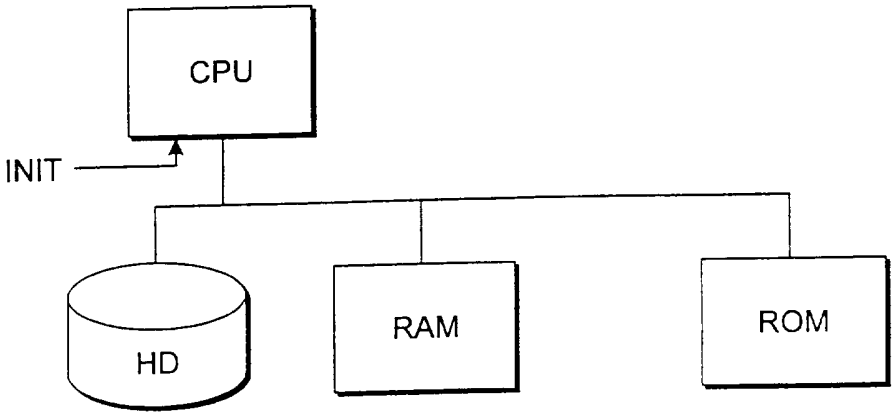


Fig. 4

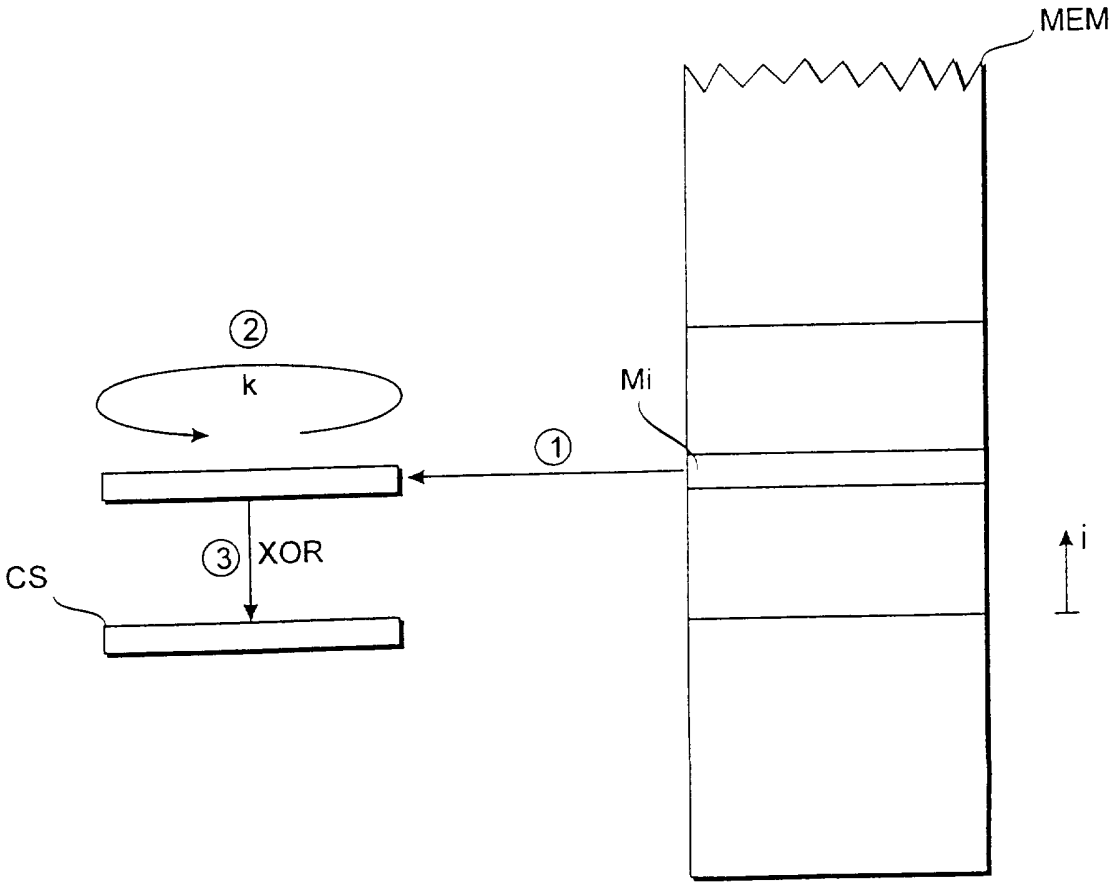


Fig. 2

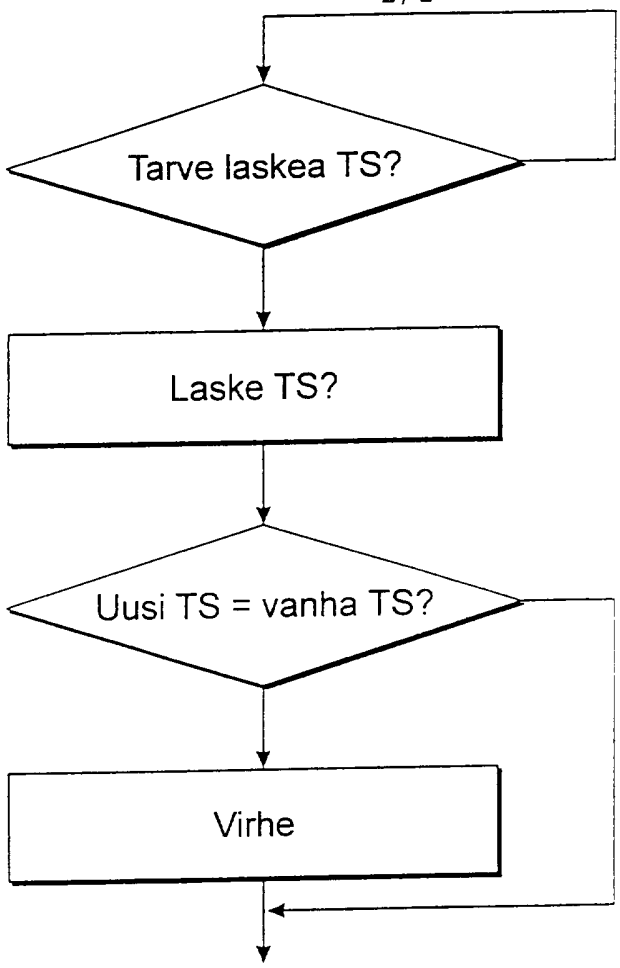
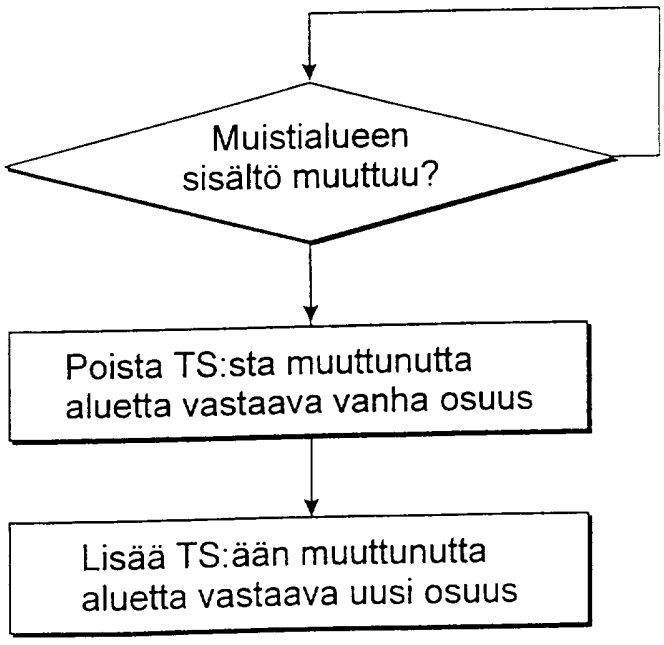

Tarve laskea TS?
Laske TS?
Uusi TS = vanha TS?
Virhe

Fig. 3


Muistialueen sisältö muuttuu?
Poista TS:sta muuttunutta aluetta vastaava vanha osuus
Lisää TS:ään muuttunutta aluetta vastaava uusi osuus

INTERNATIONAL SEARCH REPORT

International application No.
PCT/FI 96/00572

A. CLASSIFICATION OF SUBJECT MATTER

IPC6: G06F 11/00
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)

IPC6: G06F

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

SE,DK,FI,NO classes as above

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

EDOC JAPIO WPIL

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 4355390 A (F.W. HELLWIG ET AL), 19 October 1982 (19.10.82), column 2, line 12 - line 41	7
Y	US 4843633 A (B.J. MENICH ET AL), 27 June 1989 (27.06.89), column 2, line 60 - line 65	7
A	US 5007053 A (B.R.IYER ET AL), 9 April 1991 (09.04.91)	1-6
A	US 4849978 A (Y.DISHON ET AL.), 18 July 1989 (18.07.89)	1-6

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

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" erlier document but published on or after the international filing date
"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)
"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

"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
"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
"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
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 March 1997	24 -03- 1997
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Jan Silfverling Telephone No. +46 8 782 25 00

INTERNATIONAL SEARCH REPORT

International application No.
PCT/FI 96/00572

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 4433388 A (D.W.D.OOSTERBAAN), 21 February 1984 (21.02.84) -- --------	1-6

INTERNATIONAL SEARCH REPORT

Information on patent family members

04/03/97

International application No.
PCT/FI 96/00572

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4355390 A	19/10/82	DE 2939461 A,C	09/04/81
US 4843633 A	27/06/89	US 4730187 A	08/03/88
US 5007053 A	09/04/91	EP 0371243 A	06/06/90
		JP 2194457 A	01/08/90
US 4849978 A	18/07/89	EP 0297507 A	04/01/89
		JP 1021652 A	25/01/89
		JP 1798330 C	12/11/93
		JP 5004699 B	20/01/93
US 4433388 A	21/02/84	NONE	