(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2007 (29.11.2007)

PCT

(10) International Publication Number
WO 2007/134648 A1

(51) International Patent Classification:
G06F 11/34 (2006.01) *G06F 9/30* (2006.01)

(21) International Application Number:
PCT/EP2006/069992

(22) International Filing Date:
20 December 2006 (20.12.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/437,220 19 May 2006 (19.05.2006) US

(71) Applicant *(for all designated States except US)*: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(71) Applicant *(for MG only)*: **IBM UNITED KINGDOM LIMITED** [GB/GB]; PO Box 41, Portsmouth Hampshire PO6 3AU (GB).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **GREINER, Dan** [US/US]; 1615 Tiffany Way, San Jose, California 95125-5021 (US).

(74) Agent: **WILLIAMS, Julian, David**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester Hampshire SO21 2JN (GB).

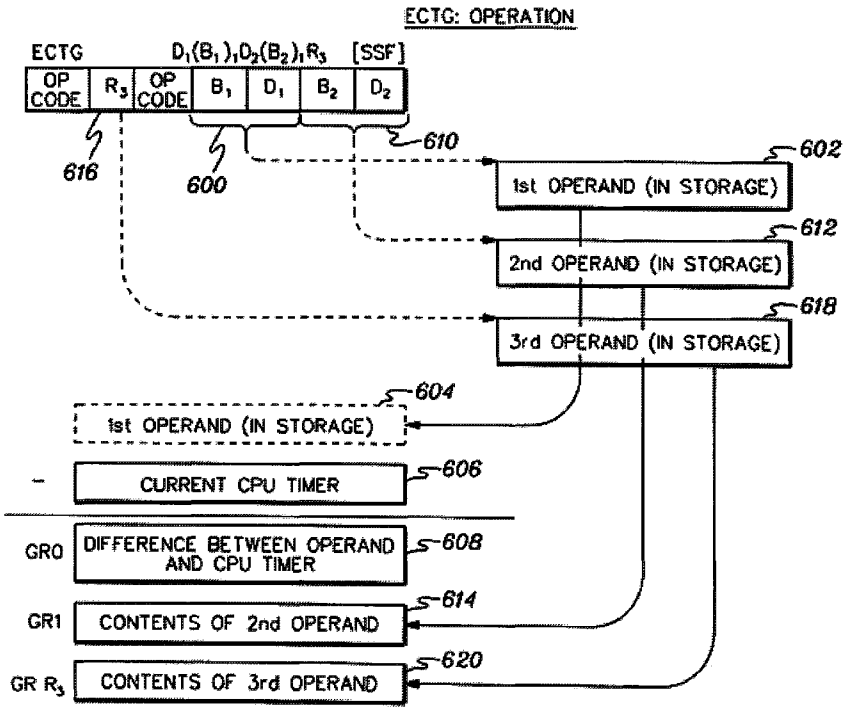
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** EXTRACT CPU TIME FACILITY



(57) **Abstract:** An efficient facility for determining resource usage, such as a processor time used by tasks. The determination is performed on behalf of user applications that do not require a call to operating system services. The facility includes an instruction that determines elapsed time and reports it to the user as a single unit of operation.

WO 2007/134648 A1

EXTRACT CPU TIME FACILITY

Technical Field

This invention relates, in general, to processing within a processing environment, and in particular, to a facility to efficiently determine resource usage of tasks.

Background of the Invention

The determination of resource usage is critical for many aspects of processing, including code refinement, billing, etc. One resource for which utilization is determined is processor time. In the z/Architecture, offered by International Business Machines Corporation, a timer is provided that measures elapsed central processing unit (CPU) time and causes an interruption when a specified amount of time has elapsed.

This timer is set by a Set CPU Timer (SPT) control instruction, and the contents of the timer are inspected via a Store CPU Time (STPT) control instruction. Both of these instructions are privileged instructions to ensure the accuracy of the time, and as such are not usable by problem-state programs (i.e., user programs).

In addition to the above, the z/OS® operating system, offered by International Business Machines Corporation, also provides a service routine referred to as TIMEUSED, which is available to problem-state programs. A program or operation calls the service to determine the amount of CPU time a piece of code (e.g., task) has used. The TIMEUSED service routine computes the elapsed time, adds the accumulated time, and returns the value to the program. The calculations of the TIMEUSED routine must be performed while being disabled for interruptions, since any interruption could adversely effect the results by manipulating the CPU timer or the accumulator.

The TIMEUSED service routine is linked to via program call and program return instructions. This routine disables for interruptions, obtains and releases a CPU lock, establishes a recovery environment, calculates the elapsed time, and re-enables after having completed its work, all of which takes hundreds of CPU cycles. When attempting to measure

a small fragment of code, the overhead of the TIMEUSED service routine can severely perturb what is being measured.

Summary of the Invention

Based on the foregoing, a need exists for a facility to efficiently determine resource usage, such as elapsed CPU time of a task. In particular, a need exists for a facility that efficiently determines resource usage of tasks without calling operating system services. A need exists for the ability of a user to efficiently determine resource usage.

The shortcomings of the prior art are overcome and additional advantages are provided through the provision of a method as claimed in claim 1.

System and computer program products corresponding to the above-summarized method, as well as one or more instructions, are also described and claimed herein.

Additional features and advantages are realized through the techniques of the present invention. Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention.

Brief Description of the Drawings

One or more aspects of the present invention are particularly pointed out and distinctly claimed as examples in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the invention are apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

FIG. 1 depicts one embodiment of a processing environment incorporating and using one or more aspects of the present invention;

FIG. 2 depicts one embodiment of the logic associated with determining resource usage, in accordance with an aspect of the present invention;

FIG. 3 depicts one example of a format of an Extract CPU Time instruction used in accordance with an aspect of the present invention;

FIG. 4a depicts one embodiment of the fields of general register 0 used by the Extract CPU Time instruction of one aspect of the present invention;

FIG. 4b depicts one embodiment of the fields of general register 1 employed by the Extract CPU Time instruction of one aspect of the present invention;

FIG. 5 depicts one embodiment of the logic associated with executing the Extract CPU Time instruction, in accordance with an aspect of the present invention;

FIG. 6 is a pictorial representation of the operations of the Extract CPU Time instruction of one aspect of the present invention; and

FIG. 7 depicts one example of a computer program product incorporating one or more aspects of the present invention.

Best Mode for Carrying Out the Invention

In accordance with an aspect of the present invention, a facility is provided to efficiently determine resource usage of tasks executing within a processing environment. In one example, a facility is provided in which a user (e.g., user code, user application, user program, etc.) can accurately measure the processor time required to execute a particular code fragment (referred to herein as a task). This facility determines the elapsed processor time without significant overhead that has skewed such measurements, such as overhead associated with using an operating system service to determine the elapsed time.

One embodiment of a processing environment incorporating and using one or more aspects of the present invention is described with reference to FIG. 1. Processing environment 100 is, for instance, a multi-processing environment including a plurality of processors 102 (e.g., central processing units (CPUs)), a memory 104 (e.g., main memory) and one or more input/output (I/O) devices 106 coupled to one another via, for example, one or more buses 108 or other connections.

As one example, each processor 102 is a an IBM System Z™ server, offered by International Business Machines Corporation, Armonk, New York, and one or more of the processors execute an operating system, such as z/OS®, also offered by International Business Machines

Corporation. (IBM and z/OS are registered trademarks of International Business Machines Corporation, Armonk, New York, USA. Other names used herein may be registered trademarks, trademarks or product names of International Business Machines Corporation or other companies.)

Processing within the processing environment is facilitated by the provision of a facility that enables the determination of resource usage, such as elapsed processor (e.g., CPU) time, without requiring the call of an operating system service or without using privileged instructions or operations.

One embodiment of the logic associated with determining resource usage is described with reference to FIG. 2. This logic is executed by a processing unit of the processing environment, in response to, for instance, a request by a user application (e.g., non-privileged code). The resource usage is determined for a task absent a call to an operating system service and without using privileged operations, STEP 200.

Initially, a current value of a counter used to track the resource for the task, such as time used, is determined by reading the counter value, as instructed by the logic, STEP 202. The determined value is then subtracted from a saved value, which is, for instance, the value of the counter when it was started, STEP 204. In one example, the counter decrements as the resource is used by the task and the current value of the counter is read, in response to the request. The result of the subtraction represents the amount of resource used by the task for this time interval, which is defined by the beginning and ending values of the counter, STEP 206.

In one example, the operations used to determine resource usage are performed by an instruction. As a specific example, an instruction is provided to determine an amount of processor time used by a task. The instruction can be implemented in many architectures and may be emulated. As examples, the instruction is executed in hardware by a processor; or by emulation of an instruction set that includes this instruction, by software executing on a processing unit having a different native instruction set. In one particular example, the instruction is implemented in the z/Architecture, offered by International Business Machines Corporation, and is referred to herein as an Extract CPU Time (ECTG) instruction.

An Extract CPU Time instruction 300 (FIG. 3) is a non-privileged instruction, and includes, for instance, an operation code 302a, 302b designating the Extract CPU Time instruction; a general register 304, the contents of which specify a third operand used by the instruction; a base register 306, which may be any of sixteen general purpose registers of the processing unit and includes a portion of an address of a first operand in storage used by the instruction; a displacement value 308, which is, for instance, an unsigned 12 bit binary number added to the contents of register 306 to provide the address of the first operand in storage; a base register 310, which again is any of the sixteen general purpose registers in the processing unit and includes a portion of an address of a second operand in storage used by the instruction; and a displacement value 312, which is added to the contents of register 310 to provide the address of the second operand in storage for the instruction.

In addition to the registers described above, the Extract CPU Time instruction also implicitly uses two general registers that do not have to be encoded in the instruction, but are used by the instruction. These registers include general register 0 and general register 1.

General register 0 (400, FIG. 4a) includes, for instance, the elapsed time since last dispatch of the task 402. It is the difference resulting from subtracting the value of the current CPU timer from the first operand, the contents of which include the value of the CPU timer at task dispatch.

General register 1 (410; FIG. 4b) includes, for instance, a value of the task time accumulator when the task was dispatched 412. This is the contents of the second operand of the instruction.

Although examples of registers are described above, each of the registers may include more, less or different information. Further, each may include additional data not necessarily needed in one or more aspects of the present invention. The specific location within the registers for the information is implementation and/or architecture dependent.

One embodiment of the logic associated with the Extract CPU Time instruction is described with reference to FIG. 5. As one example, this instruction is executed by a processor of the processing environment on behalf of a non-privileged user application (e.g., in problem state) that requests the operation as it relates to a particular task. The Extract CPU Time

instruction is a non-privileged instruction that does not invoke an operating system service. It does, however, assume in this embodiment, that the CPU timer (e.g., counter, register, etc.) is set when a task is dispatched. In one example, the timer is set by a Set CPU Timer (STP) instruction, which is a privileged instruction described in z/Architecture: Principles of Operation, IBM® Publication No. SA22-7832-04, September 2005, which is incorporated herein by reference in its entirety. It may also be set by any other means. The timer is set to a given value which represents a specified time slice for execution of the task (e.g., 10-12 ms).

In response to executing the Extract CPU Time instruction, the current value of the CPU timer is determined, STEP 500. For instance, the timer decrements as the processor processes the task, and in response to executing the Extract CPU Time instruction, the value of the timer, at that time, is observed. This includes, for instance, reading the register that holds the timer. In one embodiment, the value of the timer can be extracted at any time, including prior to the end of the time slice provided for the task and without waiting for an interruption of the timer.

The current value of the CPU timer is then subtracted from the first operand of the instruction, STEP 502. The first operand represents the value of the CPU timer at the time the task was dispatched. For example, when a task is dispatched, the CPU timer is set to a chosen value (e.g., 10-12 ms) and that value is stored in storage (e.g., PSDATSAV). Thus, PSADTSAV – current CPU Timer = elapsed processor time since last dispatch of the task. This value is placed in general register 0, STEP 504.

In addition to the above, additional information is also extracted, in one embodiment, STEP 506. As one example, the second operand of the instruction is placed unchanged in general register 1. The second operand includes, for instance, an address of a task control block (e.g., TCBTTUSD) that maintains the previously used amount of total CPU time for the task. By extracting and placing this information in general register 1, the user application is able to determine the total amount of processor time used thus far, by adding the results of general register 0 and general register 1.

Also, in one embodiment, information at the third operand location of the instruction replaces the contents of general register R_3. This information includes various types of

information, including but not limited to, flags designating information important or desired for the task, a scaling factor usable in adjusting the processor time for billing purposes, as well as other types of information.

A pictorial representation of the operations is depicted in FIG. 6. B_1D_1 (600) reference a first operand in storage 602. Subtracted from the contents of the first operand 604 is the current value of the CPU timer 606. The difference is stored in general register 0 (608). B_2D_2 (610) reference a second operand in storage 612, the contents of which are placed unchanged in general register 1 (614). Additionally, R_3 (616) references a third operand in storage 618, the contents of which are placed unchanged in general register R_3 (620).

In one embodiment, the above operations all occur within the same unit of operation, without the possibility of being interrupted. By performing these operations atomically, the values retain their meanings.

Described in detail above is a facility to efficiently determine resource usage without the overhead associated with costly operating system services and/or without using privileged operations. In particular, an Extract CPU Time facility is described that enables the efficient determination of the amount of CPU time consumed, without the costly overhead of calling an operating system service and/or without issuing Program Call and/or Program Return instructions. This facility enables an application program to accurately measure the CPU time required to execute a particular code fragment without the significant overhead that has traditionally skewed such measurements. The measurements are useful in many aspects, including, but not limited to, fine tuning of application code and billing. The facility advantageously enables an application program to efficiently determine the amount of task time used at any given moment, and not just at the end of a time slice. This allows the program to effectively determine instruction timings in the microsecond or nanosecond range without having to wait until milliseconds have elapsed.

One or more aspects of the present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has therein, for instance, computer readable program code means or logic (e.g., instructions, code, commands, etc.) to provide and facilitate the capabilities of the present

invention. The article of manufacture can be included as a part of a computer system or sold separately.

One example of an article of manufacture or a computer program product incorporating one or more aspects of the present invention is described with reference to FIG. 7. A computer program product 700 includes, for instance, one or more computer usable media 702 to store computer readable program code means or logic 704 thereon to provide and facilitate one or more aspects of the present invention. The medium can be an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system (or apparatus or device) or a propagation medium. Examples of a computer-readable medium include a semiconductor or solid state memory, magnetic tape, a removable computer diskette, a random access memory (RAM), a read-only memory (ROM), a rigid magnetic disk and an optical disk. Examples of optical disks include compact disk-read only memory (CD-ROM), compact disk-read/write (CD-R/W) and DVD.

A sequence of program instructions or a logical assembly of one or more interrelated modules defined by one or more computer readable program code means or logic direct the performance of one or more aspects of the present invention.

Although one or more examples have been provided herein, these are only examples. Many variations are possible without departing from the spirit of the present invention. For instance, processing environments other than the example provided herein may include and/or benefit from one or more aspects of the present invention. As an example, one or more processors can be other than IBM System Z™ processors and/or execute operating systems other than z/OS®. Further, the environment need not be based on the z/Architecture, but instead, can be based on other architectures, offered by, for instance, Intel, Sun Microsystems, as well as others. Yet further, the instruction can include other registers or entities other than registers to designate information. Further, different data and/or positioning within the registers and/or entities are possible. Still further, the timer can be other than counters or registers. Any mechanism can be used to determine resource usage. The term “timer” is meant to include a broad spectrum of mechanisms, including, but not limited to, counters and registers. Further, although in the embodiments herein, the timer

decrements, in other embodiments, it may increment and/or follow some pattern. Many other variations exist.

Moreover, an environment may include an emulator (e.g., software or other emulation mechanisms), in which a particular architecture or subset thereof is emulated. In such an environment, one or more emulation functions of the emulator can implement one or more aspects of the present invention, even though a computer executing the emulator may have a different architecture than the capabilities being emulated. As one example, in emulation mode, the specific instruction or operation being emulated is decoded, and an appropriate emulation function is built to implement the individual instruction or operation.

In an emulation environment, a host computer includes, for instance, a memory to store instructions and data; an instruction fetch unit to fetch instructions from memory and to optionally, provide local buffering for the fetched instruction; an instruction decode unit to receive the instruction fetch unit and to determine the type of instructions that have been fetched; and an instruction execution unit to execute the instructions. Execution may include loading data into a register for memory; storing data back to memory from a register; or performing some type of arithmetic or logical operation, as determined by the decode unit. In one example, each unit is implemented in software. For instance, the operations being performed by the units are implemented as one or more subroutines within emulator software.

Further, a data processing system suitable for storing and/or executing program code is usable that includes at least one processor coupled directly or indirectly to memory elements through a system bus. The memory elements include, for instance, local memory employed during actual execution of the program code, bulk storage, and cache memory which provide temporary storage of at least some program code in order to reduce the number of times code must be retrieved from bulk storage during execution.

Input/Output or I/O devices (including, but not limited to, keyboards, displays, pointing devices, etc.) can be coupled to the system either directly or through intervening I/O controllers. Network adapters may also be coupled to the system to enable the data processing system to become coupled to other data processing systems or remote printers or

storage devices through intervening private or public networks. Modems, cable modems and Ethernet cards are just a few of the available types of network adapters.

As used herein, the term "operand" not only includes and/or refers to operands of an instruction, but also other operands, as well as parameters or arguments passed between functions of programs, or any other data that is passed between entities. Further, a task includes any portion of code, including an entire application or program or any portion thereof.

The capabilities of one or more aspects of the present invention can be implemented in software, firmware, hardware or some combination thereof. At least one program storage device readable by a machine embodying at least one program of instructions executable by the machine to perform the capabilities of the present invention can be provided.

The flow diagrams depicted herein are just examples. There may be many variations to these diagrams or the steps (or operations) described therein without departing from the spirit of the invention. For instance, the steps may be performed in a differing order, or steps may be added, deleted or modified. All of these variations are considered a part of the claimed invention.

Claims

1. A method of determining central processing unit time usage of tasks of a processing environment, said method comprising:

selecting a task of the processing environment for which usage of central processing unit time is to be determined; and

determining an amount of the central processing unti time used by the task within a particular time interval, by

determining a current value of a timer set for the task; and

subtracting said current value from a saved value to determine an amount of elapsed processor time used by the task during the particular time interval.

2. The method of claim 1, wherein said determining occurs at least at a time prior to an end of a time slice provided for the task.

3. The method of claim 1, further comprising adding the amount of elapsed processor time to an accumulated value for the task to determine a total amount of processor time used by the task thus far.

4. The method of claim 3, further comprising extracting the accumulated value, wherein the determining the amount of elapsed processor time and the extracting the accumulated value occur as a single unit of operation.

5. The method of claim 1, wherein said determining is performed using an instruction.

6. The method of claim 5, wherein said instruction enables extraction of additional information.

7. The method of claim 6, wherein the extraction of additional information and the determining occur as a single, uninterruptible unit of operation.

8. The method of claim 1, wherein the particular time interval is defined by a starting time of a timer associated with the task and an ending time of the timer.

9. The method of claim 1, wherein said method further comprises extracting additional information relating to the task, wherein the determining the amount of elapsed processor time and the extracting occur as a single unit of operation.

10. A system comprising means adapted for carrying out all the steps of the method according to any preceding method claim.

11. A computer program comprising instructions for carrying out all the steps of the method according to any preceding method claim, when said computer program is executed on a computer system.

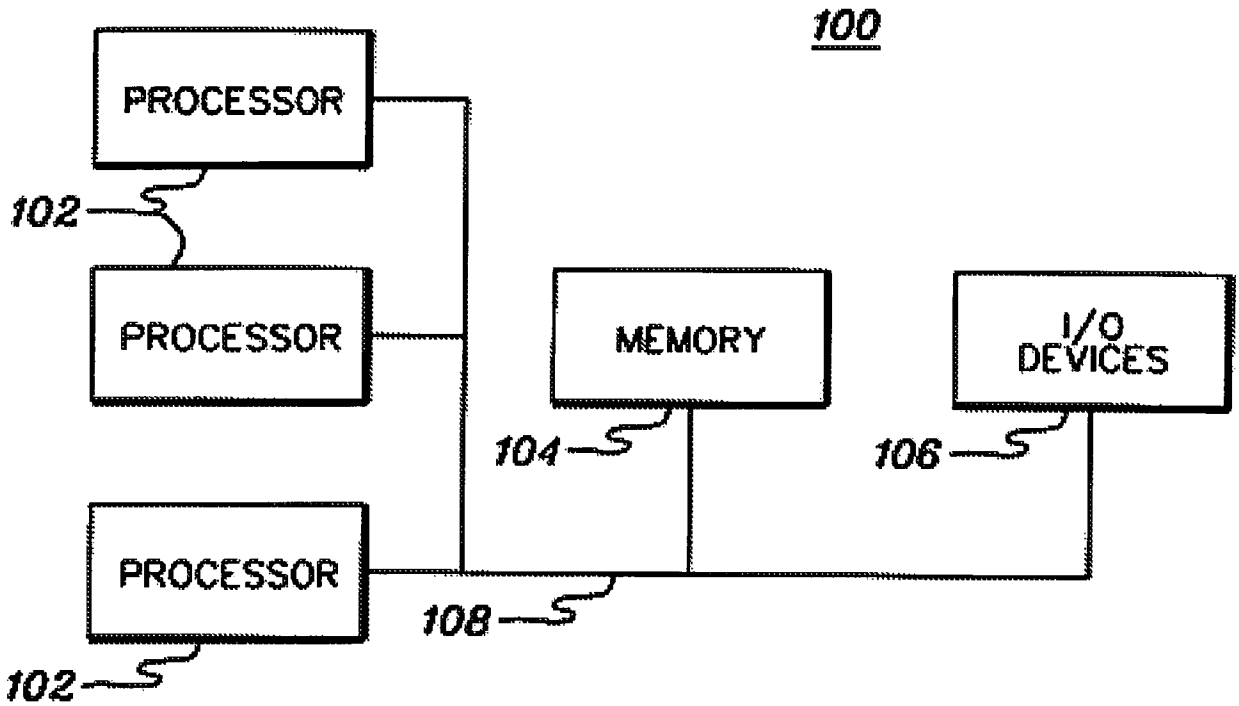

100
PROCESSOR
102
PROCESSOR
MEMORY
I/O DEVICES
104
106
PROCESSOR
108
102

fig. 1

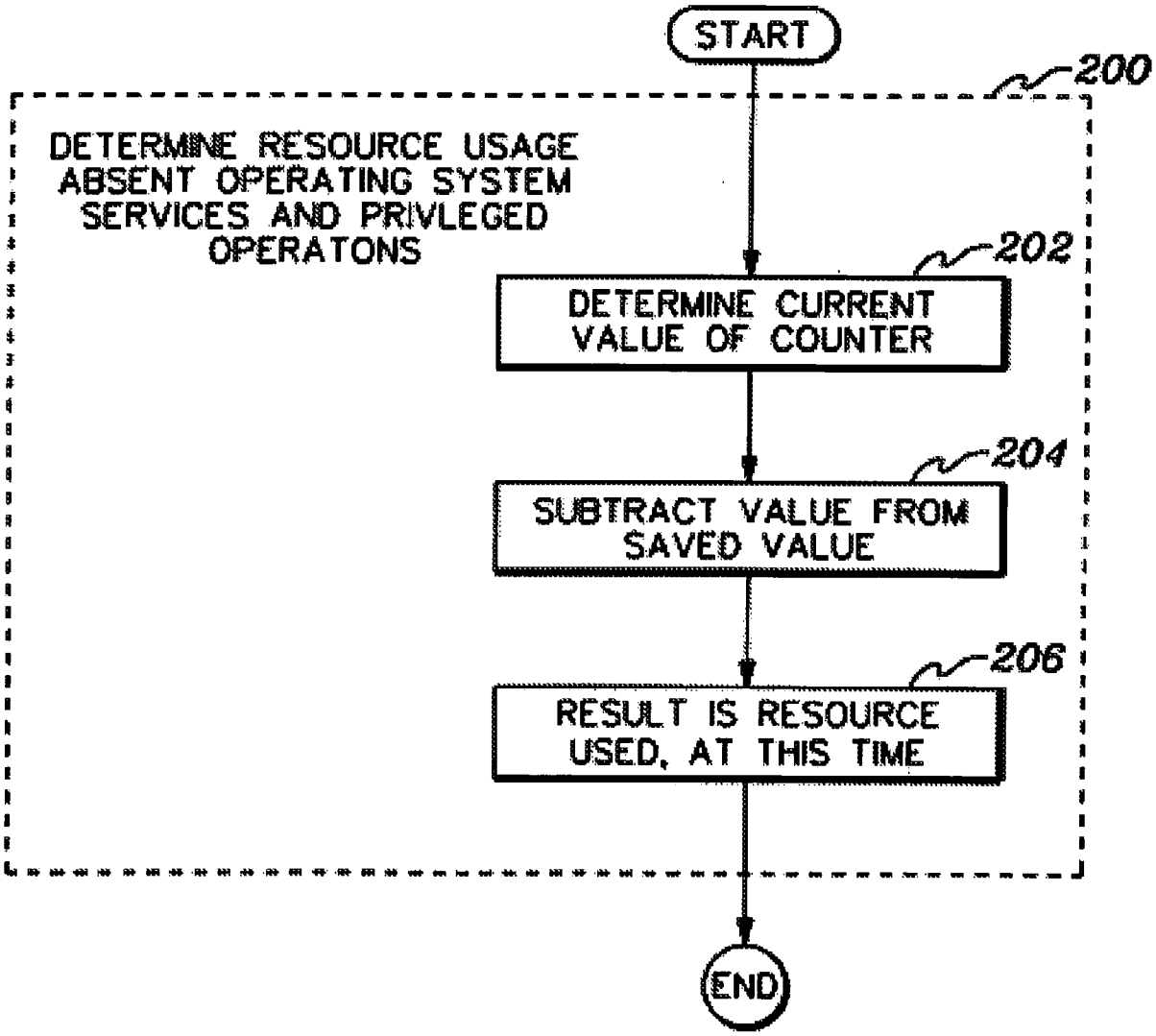

START
200
DETERMINE RESOURCE USAGE ABSENT OPERATING SYSTEM SERVICES AND PRIVLEGED OPERATONS
202
DETERMINE CURRENT VALUE OF COUNTER
204
SUBTRACT VALUE FROM SAVED VALUE
206
RESULT IS RESOURCE USED, AT THIS TIME
END

fig. 2

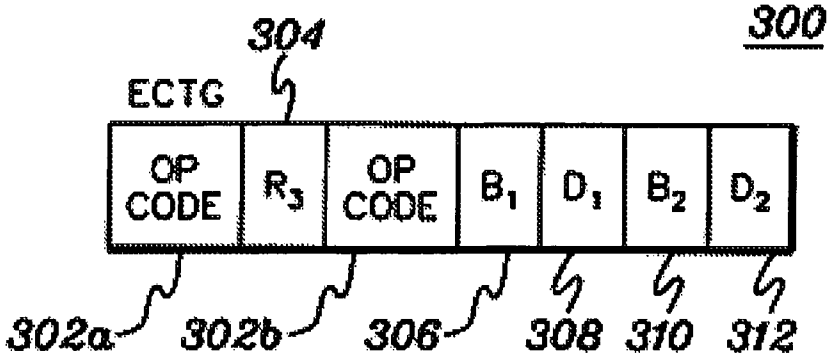


fig. 3

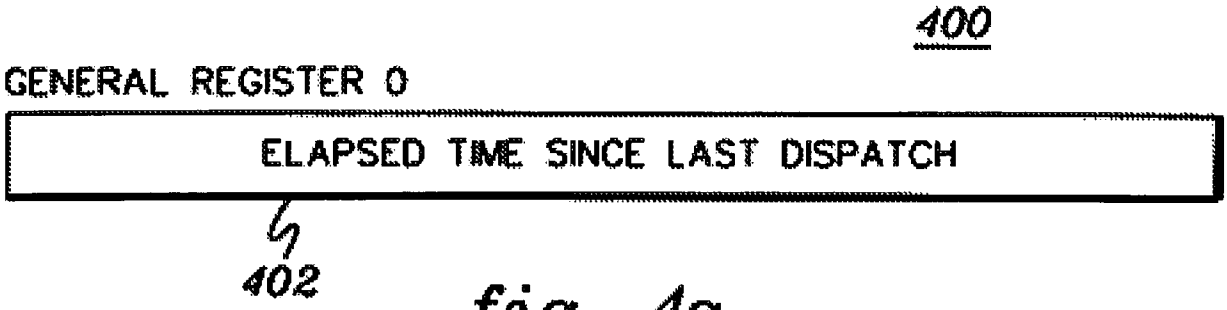


fig. 4a

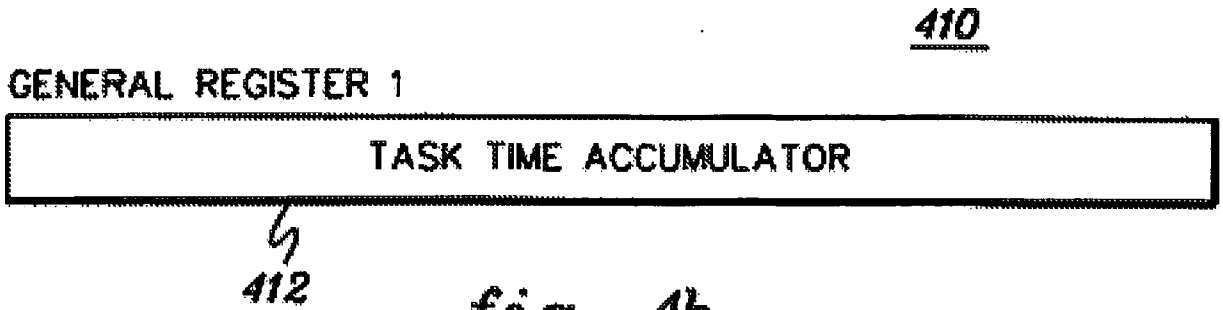


fig. 4b

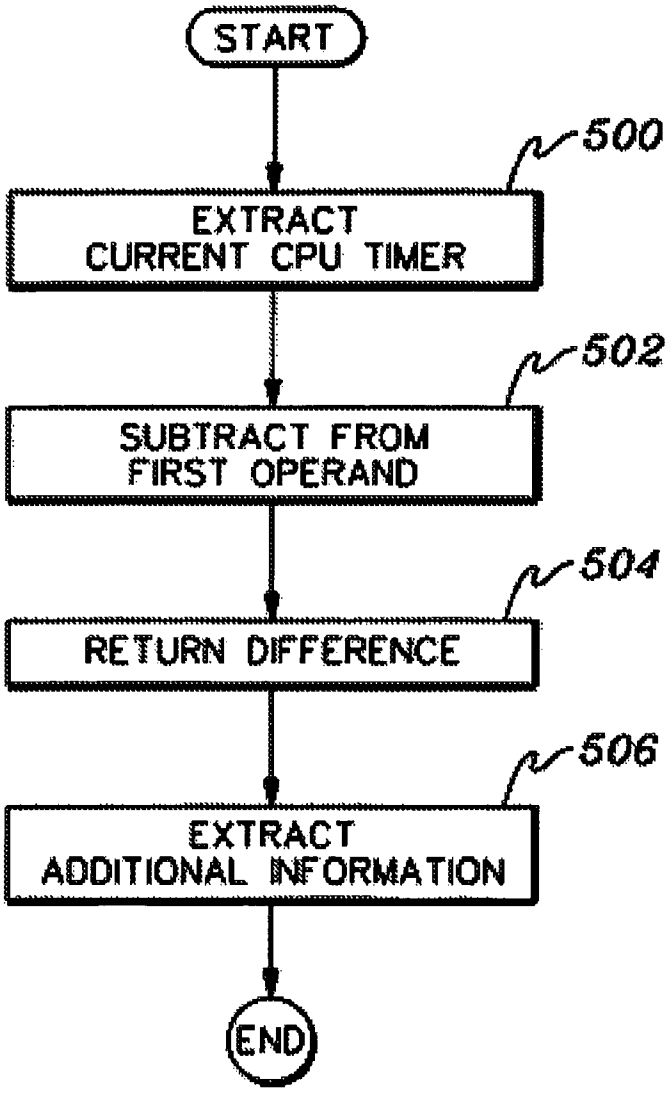

START
500
EXTRACT CURRENT CPU TIMER
502
SUBTRACT FROM FIRST OPERAND
504
RETURN DIFFERENCE
506
EXTRACT ADDITIONAL INFORMATION
END

fig. 5

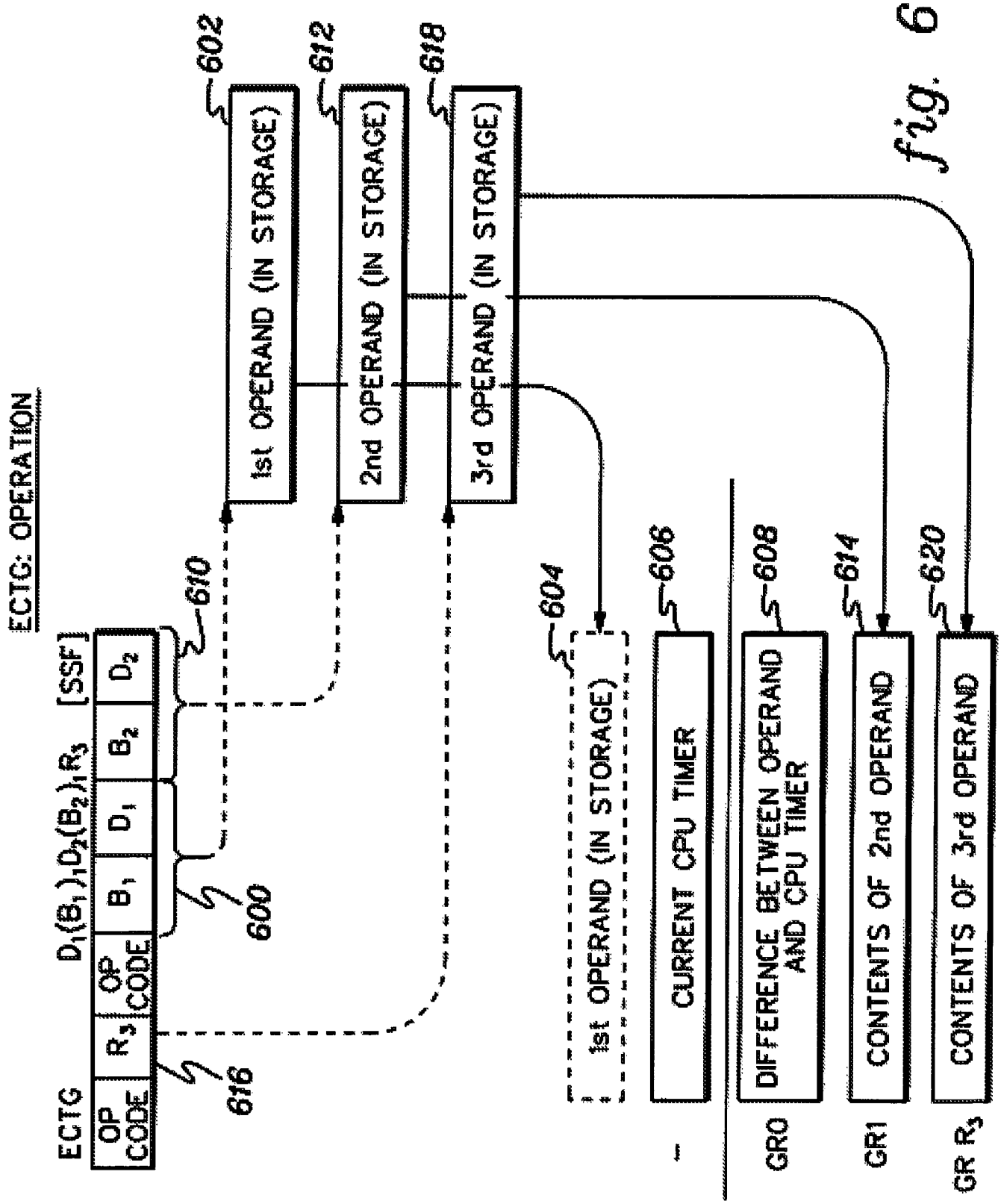

ECTG: OPERATION
ECTG
OP CODE
R3
OP CODE
D1(B1),D2(B2),R3 [SSF]
B1
D1
B2
D2
600
610
616
602
612
618
1st OPERAND (IN STORAGE)
2nd OPERAND (IN STORAGE)
3rd OPERAND (IN STORAGE)
604
1st OPERAND (IN STORAGE)
606
CURRENT CPU TIMER
608
DIFFERENCE BETWEEN OPERAND AND CPU TIMER
614
CONTENTS OF 2nd OPERAND
620
CONTENTS OF 3rd OPERAND
GR0
GR1
GR R3

fig. 6

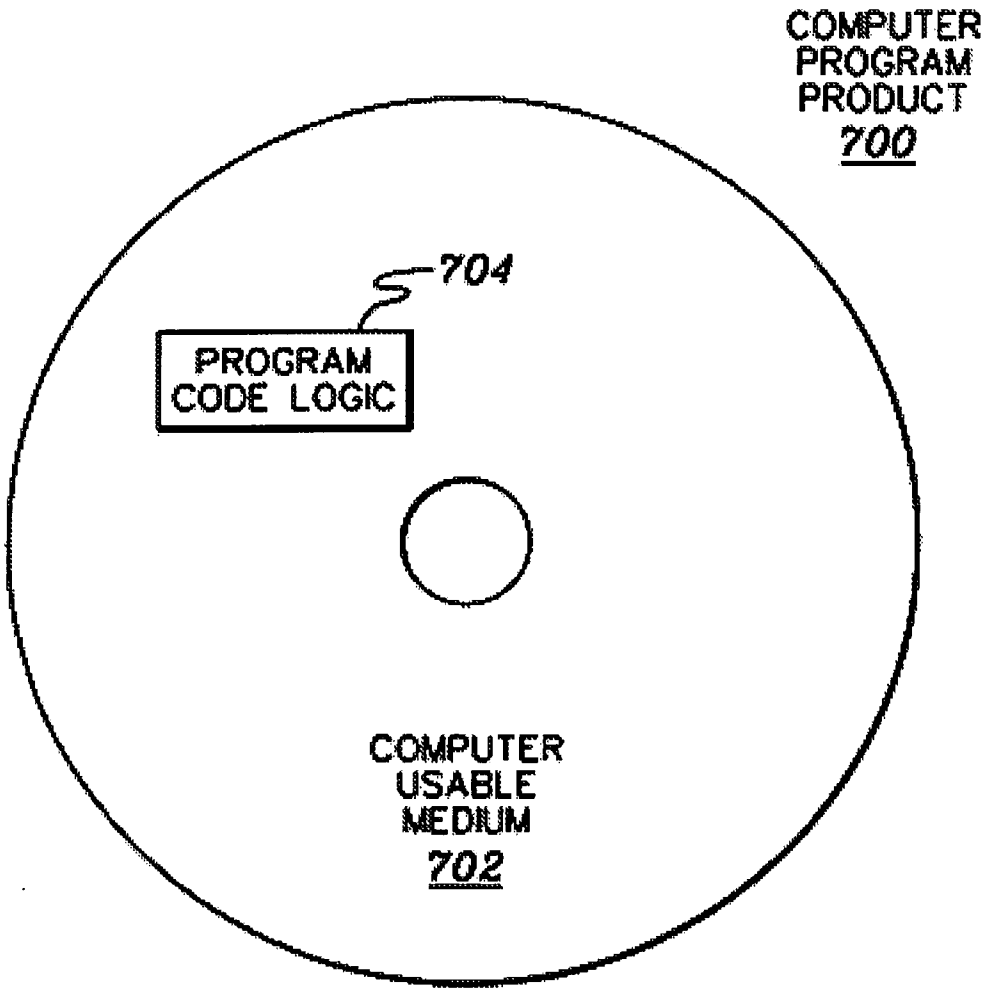

COMPUTER
PROGRAM
PRODUCT
700
704
PROGRAM
CODE LOGIC
COMPUTER
USABLE
MEDIUM
702

fig. 7

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/069992

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F11/34 G06F9/30

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)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	US 6 385 637 B1 (PETERS MICHAEL [US] ET AL) 7 May 2002 (2002-05-07) abstract; figure 2 column 6, line 59 - column 10, line 23	1-3,5-8, 10,11 4,9
X A	US 6 016 466 A (GUINTHER THOMAS [US] ET AL) 18 January 2000 (2000-01-18) abstract; figures 13,15 column 18, line 48 - column 22, line 23	1-3,5-8, 10,11 4,9
X A	WO 00/72143 A (BULL HN INFORMATION SYST [US]) 30 November 2000 (2000-11-30) page 5, line 13 - page 10, line 16	1-3,5-8, 10,11 4,9
A	EP 0 953 908 A (SUN MICROSYSTEMS INC [US]) 3 November 1999 (1999-11-03) abstract paragraph [0019] - paragraph [0032]	1-11

-/--

[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" earlier 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
23 August 2007	31/08/2007

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Bozas, Ioannis

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/069992

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	ZAGHA M ET AL: "Performance Analysis Using the MIPS R10000 Performance Counters" SUPERCOMPUTING, 1996. PROCEEDINGS OF THE 1996 ACM/IEEE CONFERENCE ON PITTSBURGH, PA, USA 01-01 JAN. 1996, PISCATAWAY, NJ, USA,IEEE, 1 January 1996 (1996-01-01), pages 16-16, XP010779838 ISBN: 0-89791-854-1 the whole document	1-11

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2006/069992

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6385637 B1	07-05-2002	NONE	
US 6016466 A	18-01-2000	NONE	
WO 0072143 A	30-11-2000	EP 1292888 A1	19-03-2003
		US 6247170 B1	12-06-2001
EP 0953908 A	03-11-1999	JP 2000040022 A	08-02-2000
		US 2002099760 A1	25-07-2002