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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 December 2007 (06.12.2007)

PCT

(10) International Publication Number
WO 2007/139529 A1

(51) International Patent Classification:
G06F 12/08 (2006.01)

(21) International Application Number: PCT/US2005/018252

(22) International Filing Date: 24 May 2005 (24.05.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/574,040 24 May 2004 (24.05.2004) US
11/135,838 23 May 2005 (23.05.2005) US

(71) Applicant *(for all designated States except US)*: **SUN MICROSYSTEMS, INC.** [US/US]; 4120 Network Circle, Santa Clara, CA 95054 (US).

(72) Inventors: **JACOBSON, Quinn, A.**; 598 Sunnymount Ave., Sunnyvale, CA 94087 (US). **CHAUDHRY, Shailender**; 255 King Street #530, San Francisco, CA 94107 (US).

(74) Agents: **GUNNISON, Forest** et al.; Gunnison, McKay & Hodgson, L.L.P., 1900 Garden Road, Suite 220, Monterey, CA 93940 (US).

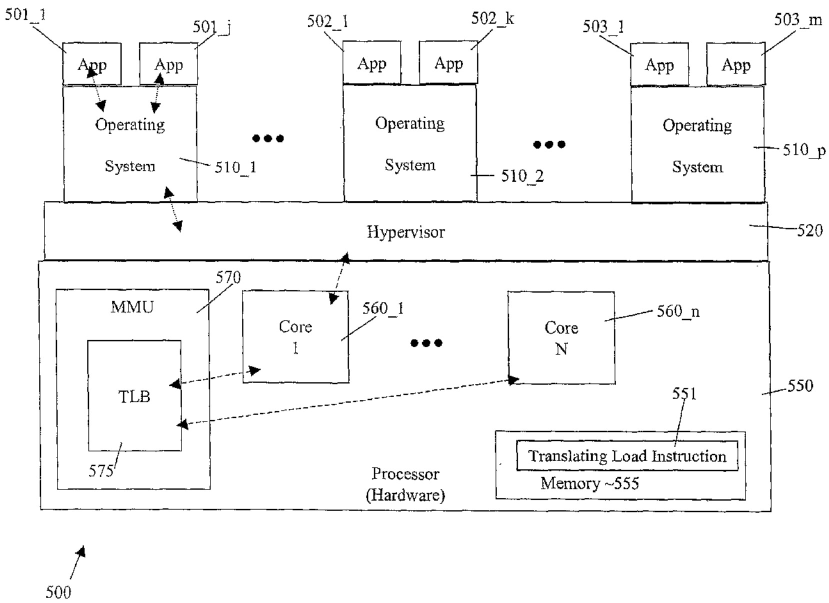
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK,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



(57) Abstract: A system (500), which includes a processor (550) that includes a plurality of cores (560_1, 560_n), generates an address translation when there is a miss in a translation lookaside buffer (TLB) (575). A hypervisor (520) utilizes a translating load instruction (551) that upon execution on processor (550) generates a data portion of a TLB entry. Execution of translating load instruction (551) utilizes information from a real-to-physical address translation table entry and information provided in the call to the translating load instruction (551) to synthesize the data portion of the new virtual-to-physical translation table entry.

WO 2007/139529 A1

Translating Loads for Accelerating Virtualized Partition

RELATED APPLICATION

[0001] This application claims the benefit of U.S. Provisional Application No. 60/574,040 filed May 24, 2004 entitled "Translating Loads for Accelerating Virtualized Partition" and naming Quinn A. Jacobson and Shailender Chaudhry as inventors, which is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] The present invention relates generally to memory management in computer systems, and more particularly to accelerating address translations following a translation lookaside buffer miss.

Description of Related Art

[0003] Virtual memory and the addresses used to locate information in the virtual memory is an old concept. Historically, virtual addresses were used to provide a large memory space to applications by a processor. The processor converted the virtual addresses to a physical address. To reduce the overhead in mapping a virtual address to a physical address, a translation lookaside buffer 100 (Fig. 1) was used.

[0004] Translation lookaside buffer 100 was a cache for holding recently used mappings from virtual addresses to physical addresses. Typically, a virtual address had two parts an offset and a virtual page identifier. The offset was the same for both the virtual address and the physical address.

[0005] Thus, the virtual address in Fig. 1 that was presented to translation lookaside buffer (TLB) 100 was the virtual page identifier. Translation lookaside buffer 100 checked to see if the virtual page identifier was stored in the cache and if it was, TLB 100 returned the physical address, which was the base address of the page in physical memory. However, if TLB 100 did not contain the virtual page identifier, a more detailed mapping was required using stored memory mapping tables and the TLB was updated as appropriate.

[0006] This approach was sufficient for a single operating system handling multiple applications, e.g., contexts. However, in main frames a further abstraction was introduced, e.g., virtual hardware.

[0007] As illustrated in Fig. 2, a plurality of operating systems 210_1, 210_2, e.g., different instances of the same operating system, or alternatively different operations systems, used a hardware processor 250. Each operating system supported a plurality of applications, e.g., operating system 210_1 supported applications 201_1, 201_2 and operating system 210_2 supported applications 202_1, 202_2.

[0008] In system 200, the hardware is logically partitioned. Logical partitioning allows multiple copies of a single operating system (OS) or multiple heterogeneous operating systems to simultaneously run on a single data processing system platform. A logical partition, within which an operating system image runs, is assigned a non-overlapping sub-set of the platform's resources. These platform allocable resources include one or more architecturally distinct processors with their interrupt management area, regions of system memory, and I/O adapter bus slots.

[0009] Hypervisor 210, typically implemented as firmware, performed a number of functions and services for operating systems 210_1, 210_2 to create and enforce the logical partitions. Hypervisor 210 owned all system resources and provided an abstraction layer through which device access and control was arbitrated.

[0010] Hypervisor 210 and firmware handled the mapping of memory, CPUs and adapters for each logical partition. Applications were generally unaware of where the partition's memory was located, which CPUs had been assigned, or which adapters were in use.

[0011] Each application had it owns virtual address space. The operating system associated with a particular application converted a virtual address to a real address. As far as the operating system was concerned the real address started at zero and went to a predetermined maximum value. Hypervisor 220 managed the physical memory addresses.

[0012] Fig. 3 is a conceptual illustration of a two-part translation lookaside buffer 300 that could be used in the translation from a virtual address to a physical address. A first table 310 includes mappings from virtual addresses to real addresses and a second table 320 includes mappings from real addresses to physical addresses.

[0013] However, TLB 300 required serialization and so typically, the logic equivalent of tables 310, 320 was implemented. In TLB 400, a first portion 401 was a field with a value that identified whether the address was a virtual address or a real address, a second portion 402 included either the virtual address or the read address, and a third portion 403 contained the mapping to the corresponding physical address. Thus, with TLB 400, it was possible to go directly from a virtual address to the corresponding physical address, or alternatively from a real address to the corresponding physical address.

[0014] An area of emphasis has been on how to minimize the penalty when there is a miss in TLB 400 for a virtual address. Two translations are required; one from the virtual to the real address; and one from the real address to the physical address. The second translation is the one that has received most of the attention.

[0015] Either the operating system, or the hypervisor using the operating system state can perform the virtual-to-real address translation. Typically, to do the translation from the real address to physical address translation base and bounds table, a coarse grain translation has been used.

SUMMARY OF THE INVENTION

[0016] In one embodiment of the present invention, a method generates a virtual-to-real-to-physical address translation more rapidly than the prior art methods. In particular, the real-to-physical address portion of the translation is accelerated using a translating load operation, which in turn accelerates virtualized partitions.

[0017] In one example, hardware determines whether a translation lookaside buffer miss occurred for a virtual address. If a miss occurred, a translating load instruction is executed. Successful execution of the translating load instruction generates a new data portion for a virtual-to-physical address translation table entry.

[0018] The execution of the translating load instruction searches a table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with the virtual address. If such a real-to-physical address translation table entry is found, the execution of the translating load then determines whether permissions and size information in the real-to-physical address translation table entry are compatible with, e.g., are equal to, or a superset of, permissions and size information associated with the real address. If the permissions and size information in the real-to-physical address translation table entry are equal to, or a superset of, permissions and size information associated with the real address, the execution of the translating load instruction creates the new data portion for the virtual-to-physical address translation table entry.

[0019] In one embodiment, the table of translation table entries is a hardware translation lookaside buffer. In another embodiment, the table of translation table entries is a translating load table.

[0020] In one embodiment, a processor includes a hardware table of translation table entries including real-to-physical translation table entries and a translating load instruction stored in a memory wherein execution of the translating load instruction generates a method as described above. A system includes this processor and a memory coupled to the processor. In one embodiment, the system is a stand-alone computer system, and in another embodiment, the system is a client-server system.

[0021] In still another embodiment, a structure includes:

means for determining whether a translation lookaside buffer miss occurred for a virtual address;

means for executing a translating load instruction, following the determining finding the translation lookaside buffer miss occurred, to generate a new data portion for a virtual-to-physical address translation table entry;

means for searching a table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with the virtual address;

means for determining, following the means for searching finding the real-to-physical address translation table entry, whether permissions and size information in the real-to-physical address translation table entry are compatible with permissions and size information associated with the real address; and

means for creating the new data portion for the virtual-to-physical address translation table entry following the means for determining finding that the permissions and size information in the real-to-physical translation table entry are compatible with the permissions and size information associated with the real address.

BRIEF DESCRIPTION OF THE DRAWINGS

[0022] Fig. 1 is a block diagram of a prior art translation lookaside buffer used for mapping a virtual address to a physical address.

[0023] Fig. 2 is a block diagram of a prior art computing platform that utilized logical partitions and so had virtual addresses, real addresses, and physical addresses.

[0024] Fig. 3 is a block diagram of a prior art translation lookaside buffer used for mapping a virtual address to a real address and the real address to a physical address.

[0025] Fig. 4 is a block diagram of a prior art translation lookaside buffer used for mapping a virtual address to a physical address and for mapping a real address to a physical address.

[0026] Fig. 5 is a block diagram of a computing platform that includes a translating load instruction, and means for executing the translating load instruction, according to one embodiment of the present invention.

[0027] Fig. 6 is a process flow diagram for a method of generating a virtual-to-physical address translation table entry for the computing platform of Fig. 5, according to one embodiment of the invention.

[0028] Fig. 7A is an illustration of a memory structure that is a virtual-to-real address translation table entry used in one embodiment of the present invention.

[0029] Fig. 7B is an illustration of a memory structure that is a data portion of the virtual-to-real address translation table entry of Fig. 7A and is passed as an argument in the translating load instruction in one embodiment of the present invention.

[0030] Fig. 8 is a diagram of translation table entries in a translation lookaside buffer in one embodiment of the present invention.

[0031] Fig. 9 is a more detailed diagram of the translations table entries of Fig. 8, according to one embodiment of the present invention.

[0032] Fig. 10 is a more detailed illustration of a memory structure that is a virtual-to-real address translation table entry used in one embodiment of the present invention.

[0033] Fig. 11 is a more detailed illustration of a memory structure that is translation table entry used in a translation lookaside buffer and in a translating load table according to one embodiment of the present invention.

[0034] Fig. 12 is a block diagram of a computing platform that includes a translating load instruction, and means for executing the translating load instruction and a translating load table used in the execution, according to one embodiment of the present invention.

[0035] Fig. 13 is a diagram of a system and of systems that utilize the processor with the translating load instruction according to various embodiments of the present invention.

[0036] In the drawings and the detailed description, elements with the same reference numeral are the same or equivalent elements. Also, for three digit reference numerals, the first digit of the reference numeral is the figure number in which the corresponding element first appears. For four digit reference numerals, the first two digits of the reference numeral are the figure number in which the corresponding element first appears.

DETAILED DESCRIPTION

[0037] In one embodiment of the present invention, a system 500, which includes a processor 550 that includes a plurality of cores 560_1 to 560_n, generates an address translation more efficiently than prior art processors when there is a miss in a translation lookaside buffer (TLB) 575. Hypervisor 520 utilizes a translating load instruction that upon execution on processor 550 generates a data portion of a TLB entry. Execution of the translating load instruction utilizes information from a real-to-physical address translation table entry and information provided in the call to the translating load instruction to synthesize the data portion of a new virtual-to-physical translation table entry.

[0038] In this embodiment, system 500 is partitioned into virtual processors. A processor assignment, e.g., one of the plurality of cores 560_1 to 560_n, with respect to an execution environment for a computing application defines the virtual processor for that computing application.

[0039] In this embodiment, system 500 includes a plurality of applications 501_1 to 501_j, 502_1 to 502_k, and 503_1 to 503_m, a plurality of operating systems 510_1, 510_2 to 510_p, a hypervisor 520, and a processor 550 that in turn includes a plurality of cores 560_1 to 560_n. A memory management unit 570 of processor 550 includes a translation lookaside buffer 575.

[0040] All virtual processors in processor 550 share translation lookaside buffer 575. To assist in defining the execution environment partition identifiers and context identifiers are used in one embodiment. Partition identifiers and context identifiers are coordinated across hypervisor code and all supervisor code respectively across all virtual processors within processor 550.

[0041] Translation lookaside buffer 575 provides virtual-to-physical address translations and real-to-physical address translations. All the virtual processors for virtual-to-physical address translations and for real-to-physical address translations use hardware-based translation lookaside buffer 575.

[0042] In one embodiment, when a virtual address translation is needed, hardware in processor 550 searches TLB 575 for an appropriate translation table entry. If such an entry is not found, a TLB miss occurs.

[0043] Upon a TLB miss, processing transfers from TLB miss check operation 695 (Fig. 6) to a get virtual address to real address translation table entry operation 601. In one embodiment, MMU 570 also includes a Translation Storage Buffer (TSB), which is a translation table in memory. The TSB contains one-level mapping information for virtual addresses to real addresses. Hardware in processor 550 looks up the TSB when a translation cannot be found in TLB 575. A TSB entry is called a Translation Table Entry, or TTE.

[0044] Use of a TSB is illustrative only and is not intended to limit the invention to this specific embodiment. Either an operating system, or hypervisor 520 using the state of the operating system can generate a virtual to real address translation using for example a page table.

[0045] Fig. 7A illustrates one embodiment of a TTE 700 for a virtual-to-real address translation. TTE 700 has two portions: a tag portion 710 and data portion 720. Data portion 720 includes a real address field 725 that contains a real address that is associated with the virtual address in tag portion 710, and permissions and size field 726 that contains permissions and size information associated with the real address.

[0046] Upon completion of operation 601, a TTE 700 is obtained for the virtual address for which there was a TLB miss. Hypervisor 520 initiates execution of a translating load instruction 551 that is stored in a memory 555, e.g., in firmware. Data portion 720 (Fig. 7B) of TTE 700 is passed in the call to translating load instruction 551.

[0047] Execution of translating load instruction 551 results in performance of method 600. Specifically, a search table for real address in translation table entry (TTE) data portion operation 602 is provided with data portion 720 (Fig. 7B) that includes a real address, in real address field 725, which is associated with, e.g., corresponds to, the virtual address for which the TLB miss occurred. With this real address, operation 602 searches, for example, TLB 575A (Fig. 8) for a real-to-physical address translation table entry with a translation from the real address in data portion 720 to a physical address.

[0048] In this example, TLB 575A includes a plurality of translation table entries 801 to 803. Fig. 9 is a more detailed illustration of one embodiment of a translation table entry 900 that is representative of each of the plurality of translation table entries 801 to 803.

[0049] Translation table entry 900 includes a tag portion 910 and a data portion 920. In this example, tag portion 910, sometimes called tag 910, includes a real field 928 and an address field 925. A value in real field 928 indicates whether translation table entry 900 is for a real-to-physical address translation, or alternatively for a virtual-to-physical address translation and consequently identifies the type of address in address field 925. Data portion 920 includes a physical address field 927 that includes a physical address associated with the address in field 925, and a permissions and size field 926 that contains information associated with the physical address in field 927. Upon completion, operation 602 transfers processing to real-to-physical address translation table entry found check operation 603 (Fig. 6).

[0050] If a real-to-physical address translation table entry was not found for real address 725, check operation 603 transfers processing to error operation 604. Error operation 604 returns a predefined value, e.g., zero, which indicates that method 600 was not successful for the virtual address that resulted in the TLB miss.

[0051] If a real-to-physical address translation table entry was found for real address 725, e.g., entry 802 (Fig. 8), check operation 603 transfers processing to physical address in TTE appropriate check operation 605. Check operation 605 compares the permissions and size information in field 726 of data portion 720 of virtual-to-physical address TTE 700 with the permissions and size information in field 820 of real-to-physical address TTE 802.

[0052] If the permissions and size information in field 826 are equal to, or a superset of, the permissions and size information of field 726, check operation 605 transfers processing to create data portion of translation table entry operation 607, and otherwise to error operation 606.

[0053] If processing transferred to error operation, the information characterizing the memory associated with the physical address indicated that the memory was not appropriate for use with the virtual address. Accordingly, error operation 604 returns a value, e.g., zero, which indicates that method 600 was not successful for the virtual address that resulted in the TLB miss.

[0054] In create data portion of TTE operation 607, a translation table entry data portion for a real-to-physical address translation table entry is generated. In one embodiment, permissions and size information in field 726 is used with the physical address in TTE 802 to form a new data portion of a virtual-to-physical address TTE. This new data portion is stored and the translating load instruction completes.

[0055] The tag for the new virtual-to-physical address TTE is known and is the information associated with the original virtual address for which the TLB miss occurred. The known tag for the new virtual-to-physical address TTE along with the new data portion from operation 607 are written to TLB 575A in put TTE in TLB operation 608 to form a virtual-to-physical address translation table entry.

[0056] The above embodiments may be used with any processor having a TLB, and the capability to implement the operations described. The use of a translating load instruction to initiate method 600 is illustrative only and is not intended to limit the invention to initiating method 600 via execution of only a load instruction.

[0057] In one embodiment, the call to the translating load instruction is of the form:

TL [Source Reg] [Destination Reg] where,

Source register holds the data portion of a virtual-to-real address TTE; and

a data portion for a new virtual-to-physical address TTE is returned in the destination register if execution if successful and otherwise a predefined value is loaded in the destination register.

Thus, in this embodiment, the destination register is tested upon completion of operation 607 and if the destination register does not contain the predefined value, operation 608 is performed. If the destination register does contain the predefined value, the virtual-to-physical address translation can be performed using the prior art methods for example.

[0058] Fig. 10 illustrates a more detailed embodiment of a virtual-to-real address TTE 1000 with a tag portion 710A and a data portion 720A. In this embodiment, data portion 720A is used as the source in the translating load instruction TL. Table 1 is one embodiment of definitions of the fields in TTE 1100 and the data contained in the fields. In column **Bit,** a T is used to denote the tag portion and a D is used to denote the data portion. In the Field Names, bold is used to denote the characters used as the reference numeral for that field in Fig. 10.

TABLE 1

Bit	Field	Description
T-63:48	Context	If R is 1, this entry maps a real address to a physical address and the context information is ignored. If R is 0, this entry maps a virtual address to a real address.
T-47:42		Reserved
T-41:00	Virtual Address	Virtual Address Tag. The virtual page number. Bits 21 through 13 are not maintained in the tag because these bits index the minimally sized, direct mapped TSB of 512 entries.
D-63	**V**alid	If this valid bit is set to 1, this TTE is a valid entry
D-62	**N**on-**F**aulting Only	If this non-faulting only bit is set to 1, this TLB entry is intended to match only non-faulting address space identifiers
D-61:N+1	**SW**	Software usable bits
D-N:13	**R**eal **A**ddress	This field provides support

Bit	Field	Description
		for 47 bits of real address. For page sizes larger than 8 KB, the lower order address bits below the page size are ignored.
D-12	**I**nvert **E**ndianness	
D-11	Side **E**ffect	If this side effect bit is set, speculative loads trap for addresses within the page. Non-cacheable memory addresses, other than block loads and stores are strongly ordered against other side effect bit accesses and non-cacheable stores are not merged. This bit should be set for pages that map I/O devices having side effects. The side effect bit does not prevent normal instruction prefetching. This bit has no effect for instruction fetches. The side effect bit does not force non-cacheable access. It is expected, but not required that the CP and CV bits are cleared to zero along with the side effect bit. If both the CP and CV bits are set to one along with the side effect bit, the

Bit	Field	Description
		result is undefined. The side effect bit and the NFO bit are mutually exclusive: both bits should never be set in any TTE.
D-10:9	Cacheable Physical and Cacheable Virtual	These two bits are passed to the cache memory sub-system on any access and determine the cacheability of that access as follows: If CP is set to 1, the mapped data or instructions may be cached in any physically indexed cache; If CP and CV are both set to 1, the mapped data or instructions may be cached in any physically or virtually index cache; and If CP is cleared to 0, the contents of the mapped page are non-cacheable.
D-8	Privileged	If this privileged bit is set to one, this mapping only matches in the TLB if the processor is in the privileged mode.
D-7	EXecute	If this execute bit is set to one, instructions may be fetched and executed from

Bit	Field	Description
		this page.
D-6	**W**ritable	If this writable bit is set to one, data mapped by this page may be written to.
D-5	**R**eadable	If this readable bit is set to one, data mapped by this page may be read from.
D-4	SW	Software usable bit
D-3:	--	Reserved
D-2:0	Page **S**ize	000=8KB, 001=64KB, 010=512KB, 011=4MB, 100=32MB, 101=256MB, 110=2GB, 111=16GB

[0059] Fig. 11 illustrates one embodiment of a TTE 1200 for TLB 575A with a tag portion 910A and a data portion 920A. Table 2 is one embodiment of definitions of the fields in TTE 1100 and the data contained in the fields. In column **Bit,** a UT is used to denote an upper part of the tag portion; a LT is used denote a lower part of the tag portions; and a D is used to denote the data portion. In the Field Names, bold is used to denote the characters used as the reference numeral for that field in Fig. 11.

TABLE 2

Bit	Field	Description
UT-63	Real	If R is 1, this entry maps a real address to a physical address and the context information is ignored. If R is 0, this entry maps a virtual address to a real address.

Bit	Field	Description
UT-62:40	--	--
UT-39:32	Partition	8-bit Partition ID used for all translation matches
UT-31:25	--	--
UT-24	Select	1-bit Select field to determine which virtual processor context value to compare TLB Entry context against for a primary/secondary context translation. For nucleus context translations, the Context Field should be zero and the value of Select should be set to 00. For a real address to physical address translation, Context Field should be zero and the value of Select should be set to 0. 0 = Compare context against primary/secondary context. 1 = Compare context against primary/secondary shared context.
UT-23:16	-	-
UT-15:0	Context	A 16-bit context identifier associated with this TLB entry. For a real address to physical address translation this field should be set to zero, otherwise the behavior

Bit	Field	Description
		of the match is undefined.
LT-63:13	Address	If R=1, Real Address Tag. If R=0, Virtual Address Tag. For page sizes larger than 8 KB, the appropriate lower order bits are ignored for tag compares and reads to these lower order bits are undefined.
LT-12:00	--	--
D-63	Valid	If this valid bit is set to 1, this TLB entry is a valid entry.
D-62	**N**on-**F**aulting **O**nly	If this non-faulting only bit is set to 1, this TLB entry is intended to match only loads using the non-faulting ASIs.
D-62	**PIO** address space	If this PIO address space bit is set to 1, this TLB entry corresponds to the PIO address space. If this PIO address space bit is set to 0, this TLB entry corresponds to the memory address space.
D-60:47	--	--
D-46:13	Physical Address	This physical address field provides support for 47 bits of physical address. For page sizes larger than 8 KB, the lower order address bits below the page size are ignored.

Bit	Field	Description
D-12	Invert Endianness	
D-11	Side Effect	If this side effect bit is set, speculative loads trap for addresses within the page. Non-cacheable memory addresses, other than block loads and stores are ordered against other side effect bit accesses and non-cacheable stores are not merged. This bit should be set for pages that map I/O devices having side effects. The side effect bit does not prevent normal instruction prefetching. This bit has no effect for instruction fetches. The side effect bit does not force non-cacheable access. It is expected, but not required that the CP and CV bits are cleared to zero along with the side effect bit. If both the CP and CV bits are set to one along with the side effect bit, the result is undefined. The side effect bit and the NFO bit are mutually exclusive: both bits should

Bit	Field	Description
		never be set in any TLB entry.
D-10	Cacheable Physical	The state of this bit determines the cacheability of the access as follows: IF the cacheable physical bit is set to 1, the mapped data or instructions may be cached in any of the processor caches, because in this embodiment all caches in the processor are physically-indexed physically tagged.
D-9	Cacheable Virtual	This bit is hardwired to zero, because in this embodiment all caches in the processor are physical. The bit is read as a zero and is write ignore.
D-8	Privileged	If this privileged bit is set to one, this mapping only matches in the TLB if the processor is in the privileged mode.
D-7	EXecute	If this execute bit is set to one, instructions may be fetched and executed from this page.
D-6	Writable	If this execute bit is set to one, data mapped by this page may be written to.
D-5	Readable	If this execute bit is set to one, data mapped by this page may be read from.

Bit	Field	Description
D-4:3	--	--
D-2:0	Page Size	000=8KB, 001=64KB, 010=512KB, 011=4MB, 100=32MB, 101=256MB, 110=2GB, 111=16GB

[0060] In this embodiment, shared TLB 575A has 8K entries (2^13) and supports page sizes from 8 KB through 16 GB. TLB 575A is used for both virtual address to physical address translation and real address to physical address translation. In one embodiment, TLB 575A is banked into four physical banks. The TLB is 16-way set associative. Thus, each bank has 2048 entries organized as 128 sets.

[0061] For accessing TLB entries a set of three buffer registers are used. These three registers hold the TLB entries upper tag portion, lower tag portion and data portion, respectively. There is a TLB access operation that reads a TLB entry into the three buffer registers and a TLB access operation that writes the two tag registers plus the data of the store into a TLB entry.

[0062] A small, per micro-core table of TLB entries, called a translating load table 1275(Fig. 12), is used to accelerate the two step translation required for virtual-to-real-to-physical translation, as illustrated in Fig. 6 in one embodiment. Either the full TLB or a smaller dedicated table can be used in method 600. Here, a smaller table is used to achieve better performance. In one embodiment, translating load table 1275 is a small fully associative table of sixteen entries containing TLB entries for real-to-physical address translation. The table is only used for translating load operations.

[0063] The execution of the translating load instruction performs a lookup in the special translating load table in operation 602. The 64-bit address used for this load operation is interpreted as being the TTE data portion of a virtual-to-real TTE. The TLB is accessed in operation 602 to look for a corresponding TLB entry that performs the real-to-physical address translation.

[0064] For performing the TLB lookup, the upper two bits, bits 63 and 62, of data portion 702A are masked to zero to form a real address. This real address is sent to the TLB in accordance with a Real Address Page Size Register. If a TLB miss occurs the Real-to-Physical translating load returns the value zero. If a TLB hit is found, e.g., TTE 1100 has the correct real address in the tag, the following set of checks are performed in operation 605:

```
Address.Sz <= TLBentry.Sz
Address.R <= TLBentry.R
Address.W <= TLBentry.W
Address.X <= TLBentry.X
Address.CP == TLBentry.CP
Address.E == TLBentry.E
Address.IE == TLBentry.IE
```

[0065] Here, "Address" refers to the data portion 720A of TTE 1000 (Fig. 10), and "TLBentry" refers to data portion 920A of TTE 1100 (Fig. 11). The letter after a period is the reference numeral in Fig. 10 or Fig. 11. Thus, the size, the readability, the writeablity, the execution ability, Endiannes, side effect, and cacheabliity are checked. If the real-to-physical TTE has properties that are equal to or a superset of those in the virtual-to-real address TTE the data portion for a new virtual-to-physical address TTE is returned in operation 607, as described above.

[0066] In one embodiment, processor 550A is included in a hardware configuration 1310 like a personal computer or workstation. In this embodiment, the applications and operating system(s) are included in memory 1312 or a memory coupled to processor 550A via the Internet for example. Hardware configuration 1310 includes, but is not limited to, an I/O interface 1314, a display 1316, a keyboard 1314, and a mouse 1318.

[0067] However, in another embodiment, system 1310 is part of a client-server computer system 1300. In this embodiment, server system 1380 includes a processor 500B as well as a display 1381, memory 1384, and a network interface 1383.

[0068] For either a client-server computer system 1300 or a stand-alone computer system 1310, memory 1312 typically includes both volatile memory, such as main memory, and non-volatile memory, such as hard disk drives. While memory 1312 is illustrated as a unified structure in Fig. 13, this should not be interpreted as requiring that all memory in memory 1312 is at the same physical location. All or part of memory 1312 can be in a different physical location than processor 550A.

[0069] More specifically, processor 550, in one embodiment, can be included in a portable computer, a workstation, a server computer, or any other device. Similarly, in another embodiment, system 1300 can be comprised of multiple different computers, wireless devices, server computers, or any desired combination of these devices that are interconnected to perform the operations, as described herein.

[0070] Herein, a computer program product comprises a medium configured to store or transport computer readable code or in which computer readable code for a method is stored. Some examples of computer program products are CD-ROM discs, ROM cards, floppy discs, magnetic tapes, computer hard drives, servers on a network and signals transmitted over a network representing computer readable program code.

[0071] Herein, a computer memory refers to a volatile memory, a non-volatile memory, or a combination of the two. Similarly, a computer input unit 1316 and a display unit 1315 refer to the features providing the required functionality to input the information described herein, and to display the information described herein, respectively, in any one of the aforementioned or equivalent devices.

[0072] In view of this disclosure, the translating load functionality can be implemented in a wide variety of computer system configurations using an operating system and computer programming language of interest to the user.

[0073] While the translating load hereinbefore has been explained in connection with one embodiment thereof, those skilled in the art will readily recognize that modifications can be made to this embodiment without departing from the spirit and scope of the present invention.

[0074] For example, in one embodiment, a structure includes:

means for determining whether a translation lookaside buffer miss occurred for a virtual address;

means for executing a translating load instruction, following the determining finding the translation lookaside buffer miss occurred, to generate a new data portion for a virtual-to-physical address translation table entry;

means for searching a table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with the virtual address;

means for determining, following the means for searching finding the real-to-physical address translation table entry, whether permissions and size information in the real-to-physical address translation table entry are compatible with permissions and size information associated with the real address; and

means for creating the new data portion for the virtual-to-physical address translation table entry following the means for determining finding that the permissions and size information in the real-to-physical translation table entry are compatible with the permissions and size information associated with the real address.

[0075] In still yet another embodiment, a structure includes:

means for determining whether a translation lookaside buffer miss occurred for a virtual address;

means for finding, following the means for determining finding the translation lookaside buffer miss occurred, a virtual-to-real address translation table entry for the virtual address wherein the virtual-to-real address translation table entry comprises a data portion including (i) the real address; and (ii) permissions and size information;

means for executing a translating load instruction having the data portion as an argument to generate a new data portion for a virtual-to-physical address translation table entry; and

means for using the new data portion in a virtual-to-physical address translation table entry for the virtual address.

[0076] This disclosure provides exemplary embodiments of the present invention. The scope of the present invention is not limited by these exemplary embodiments. Numerous variations, whether explicitly provided for by the specification or implied by the specification, may be implemented by one of skill in the art in view of this disclosure.

CLAIMS

We claim:

1. A computer-based method comprising:

determining whether a translation lookaside buffer miss occurred for a virtual address; and

executing a translating load instruction, following the determining finding the translation lookaside buffer miss occurred, to generate a new data portion for a virtual-to-physical address translation table entry.

2. The computer-based method of Claim 1 wherein the executing the translating load instruction further comprises:

searching a table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with the virtual address.

3. The computer-based method of Claim 2 wherein the executing the translating load instruction further comprises:

determining, following the searching finding the real-to-physical address translation table entry, whether permissions and size information in the real-to-physical address translation table entry are compatible with permissions and size information associated with the real address.

4. The computer-based method of Claim 3 wherein the executing the translating load instruction further comprises:

creating the new data portion for the virtual-to-physical address translation table entry following the determining finding that the permissions and size information in the real-to-physical translation table entry are compatible with the permissions and size information associated with the real address.

5. The computer-based method of Claim 2 wherein the table is a hardware translation lookaside buffer.

6. The computer-based method of Claim 2 wherein the table is a translating load table.

7. A computer-based method comprising:

searching a table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with the virtual address for which a translation lookaside buffer miss occurred; and

determining, following the searching finding the real-to-physical address translation table entry, whether permissions and size information in the real-to-physical translation table entry are compatible with permissions and size information associated with the real address.

8. The computer-based method of Claim 7 further comprising:

creating a new data portion for a virtual-to-physical address translation table entry following the determining finding that the permissions and size information in the real-to-physical translation table entry are compatible with the permissions and size information associated with the real address.

9. The computer-based method of Claim 7 wherein the creating further comprises:

using a physical address from the real-to-physical address translation table entry, and using the permissions and size information associated with the real address to create the new data portion.

10. The computer-based method of Claim 7 wherein the table is a hardware translation lookaside buffer.

11. The computer-based method of Claim 7 wherein the table is a translating load table.

12. A computer-based method comprising:

determining whether a translation lookaside buffer miss occurred for a virtual address;

finding, following the determining finding the translation lookaside buffer miss occurred, a virtual-to-real address translation table entry for the virtual address wherein the virtual-to-real address translation table entry comprises a data portion including (i) a real address; and (ii) permissions and size information;

executing a translating load instruction having the data portion as an argument to generate a new data portion for a virtual-to-physical address translation table entry; and

using the new data portion in a virtual-to-physical address translation table entry for the virtual address.

13. The computer-based method of Claim 12 wherein the executing the translating load instruction further comprises:

searching a table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with the virtual address.

14. The computer-based method of Claim 13 wherein the executing the translating load instruction further comprises:

determining, following the searching finding the real-to-physical address translation table entry, whether permissions and size information in the real-to-physical address translation table entry are compatible with permissions and size information associated with the real address associated with the virtual address.

15. The computer-based method of Claim 14 wherein the executing the translating load instruction further comprises:

creating the new data portion for the virtual-to-physical address translation table entry following the determining finding that the permissions and size information in the real-to-physical translation table entry are compatible with the permissions and size information associated with the real address associated with the virtual address.

16. A processor comprising:

a hardware table of translation table entries including real-to-physical translation table entries; and

a translating load instruction stored in a memory wherein execution of the translating load instruction generates a method comprising:

searching the hardware table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with a virtual address for which a translation lookaside buffer miss occurred.

17. The processor of Claim 16 wherein the method further comprises:

determining, following the searching finding the real-to-physical address translation table entry, whether permissions and size information in the real-to-physical address translation table entry are compatible with permissions and size information associated with the real address.

18. The processor of Claim 17 wherein the method further comprises:

creating the new data portion for a virtual-to-physical address translation table entry following the determining finding that the permissions and size information in the real-to-physical translation table entry are compatible with the permissions and size information associated with the real address.

19. The processor of Claim 18 wherein the creating further comprises:

using a physical address from the real-to-physical address translation table entry, and using the permissions and size information associated with the real address to create the new data portion.

20. A system comprising:

a memory; and

a processor coupled to the memory wherein the processor further comprises:

a hardware table of translation table entries including real-to-physical translation table entries;

a translating load instruction stored in a memory wherein execution of the translating load instruction generates a method comprising:

searching the hardware table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with a virtual address for which a translation lookaside buffer miss occurred.

21. The system of Claim 20 wherein the method further comprises:

determining, following the searching finding the real-to-physical address translation table entry, whether permissions and size information in the real-to-physical address translation table entry are compatible with permissions and size information associated with the real address.

22. The system of Claim 21 wherein the method further comprises:

creating the new data portion for a virtual-to-physical address translation table entry following the determining finding that the permissions and size information in the real-to-physical translation table entry are compatible with the permissions and size information associated with the real address.

23. The system of Claim 22 wherein the creating further comprises:

using a physical address from the real-to-physical address translation table entry, and using the permissions and size information associated with the real address to create the new data portion.

24. A structure comprising:

means for determining whether a translation lookaside buffer miss occurred for a virtual address; and

means for executing a translating load instruction, following the determining finding the translation lookaside buffer miss occurred, to generate a new data portion for a virtual-to-physical address translation table entry.

25. The structure of Claim 24 wherein the means for executing the translating load instruction further comprises:

means for searching a table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with the virtual address.

26. The structure of Claim 25 wherein the means for executing the translating load instruction further comprises:

means for determining, following the means for searching finding the real-to-physical address translation table entry, whether permissions and size information in the real-to-physical address translation table entry are compatible with permissions and size information associated with the real address.

27. The structure of Claim 26 wherein the means for executing the translating load instruction further comprises:

means for creating the new data portion for the virtual-to-physical address translation table entry following the means for determining finding that the permissions and size information in the real-to-physical translation table entry are compatible with the permissions and size information associated with the real address.

28. A structure comprising:

means for searching a table of translation lookaside buffer entries for a real-to-physical address translation table entry including a real address associated with the virtual address for which a translation lookaside buffer miss occurred; and

means for determining, following the means for searching finding the real-to-physical address translation table entry, whether permissions and size information in the real-to-physical translation table entry are compatible with permissions and size information associated with the real address.

29. The structure of Claim 28 further comprising:

means for creating a new data portion for a virtual-to-physical address translation table entry following the means for determining finding that the permissions and size information in the real-to-physical translation table entry are compatible with the permissions and size information associated with the real address.

30. A structure comprising:

means for determining whether a translation lookaside buffer miss occurred for a virtual address;

means for finding, following the means for determining finding the translation lookaside buffer miss occurred, a virtual-to-real address translation table entry for the virtual address wherein the virtual-to-real address translation table entry comprises a data portion including (i) the real address; and (ii) permissions and size information;

means for executing a translating load instruction having the data portion as an argument to generate a new data portion for a virtual-to-physical address translation table entry; and

means for using the new data portion in a virtual-to-physical address translation table entry for the virtual address.

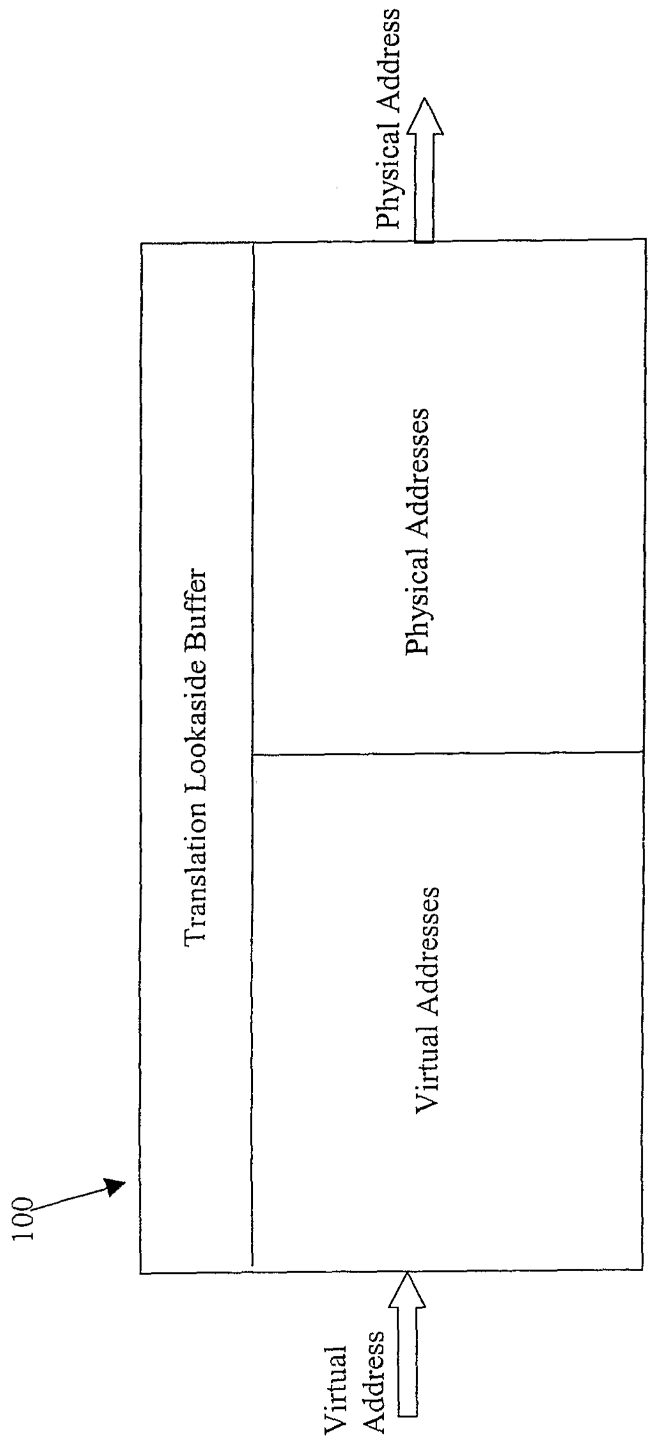


FIG. 1
(Prior Art)

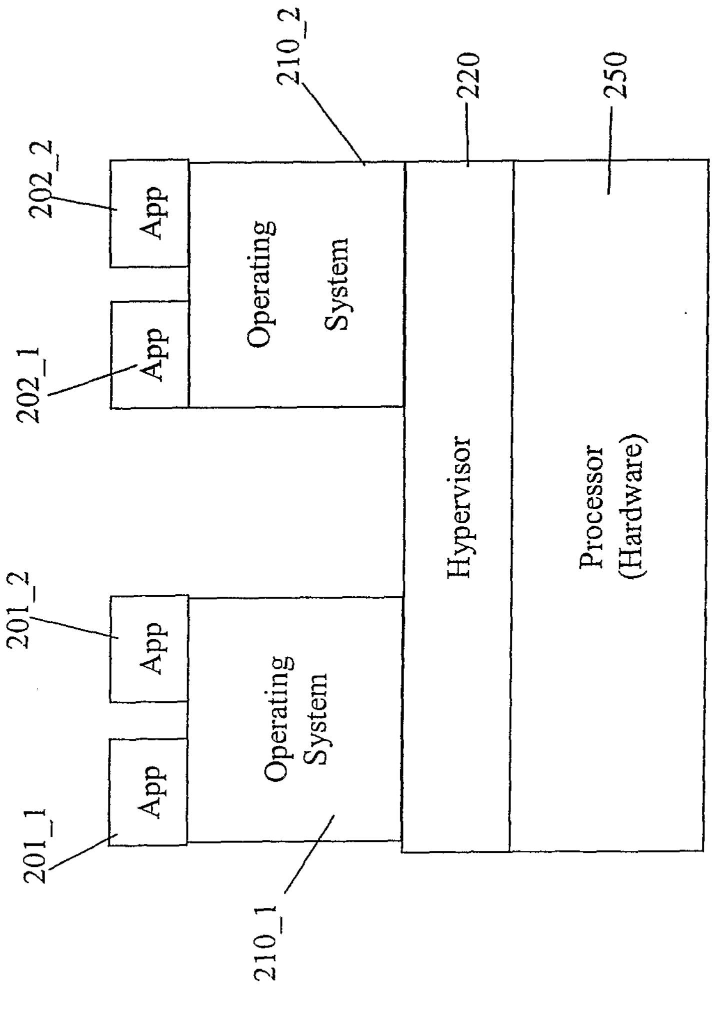


FIG. 2
(Prior Art)

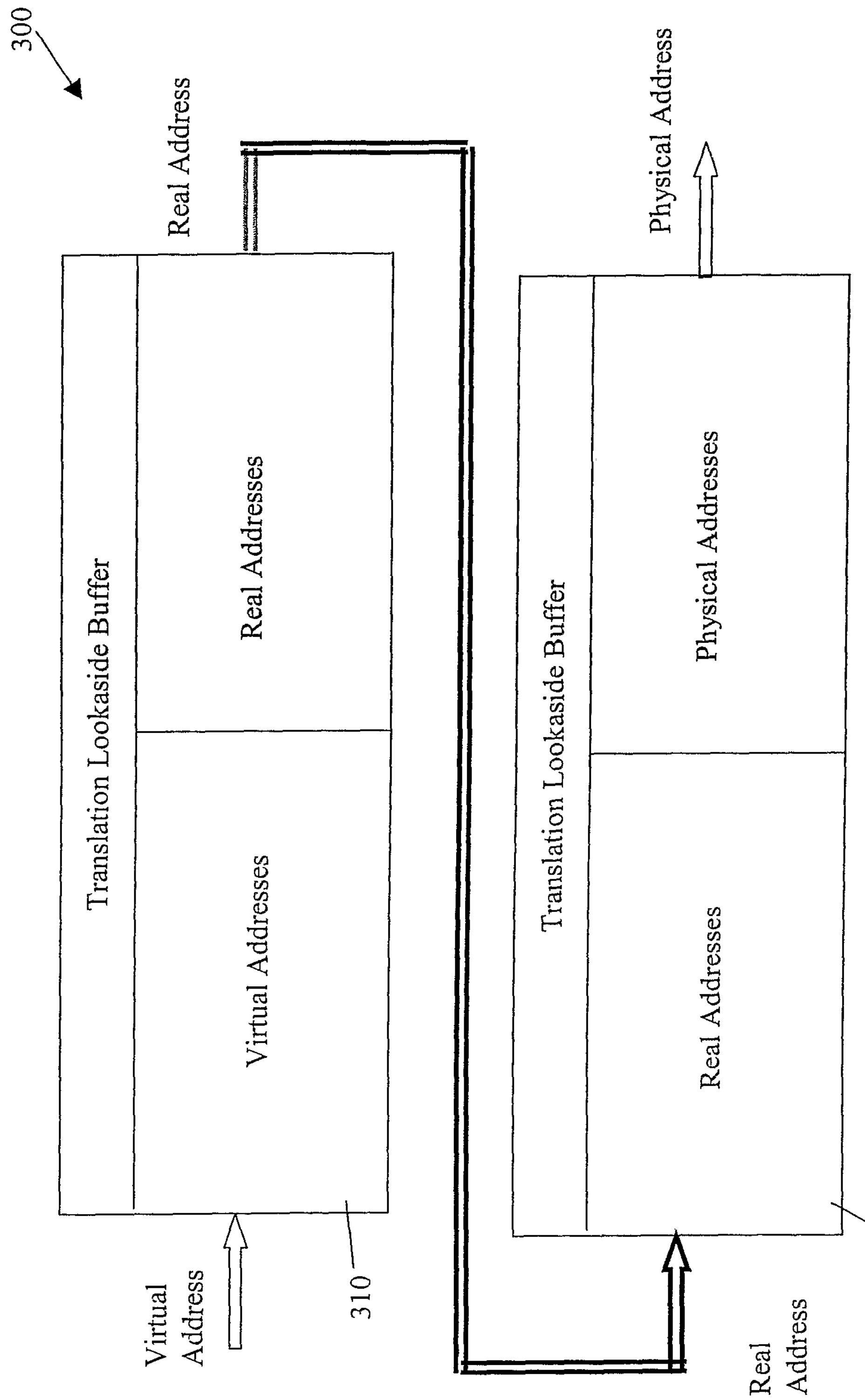


FIG. 3
(Prior Art)

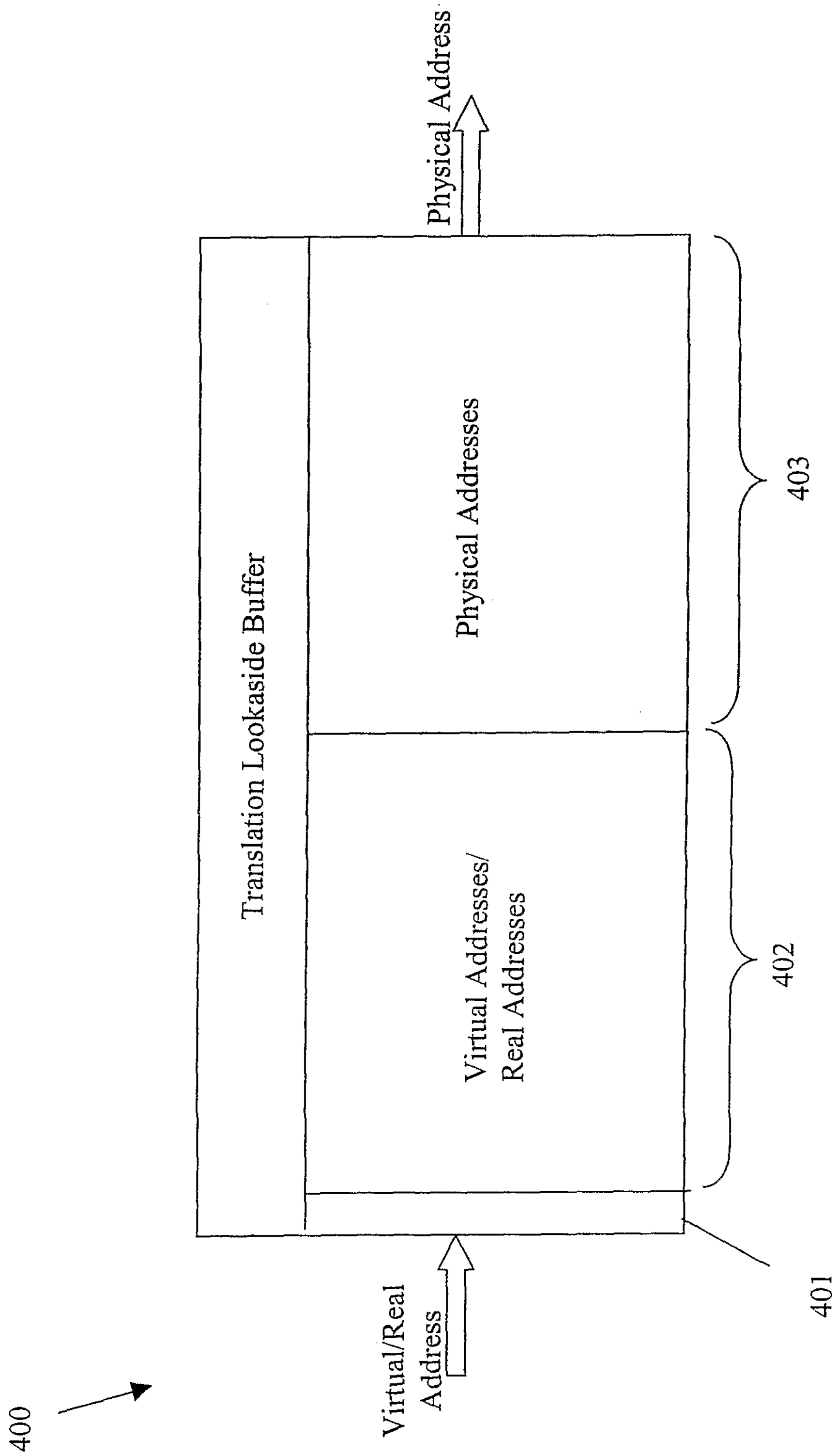


FIG. 4
(Prior Art)

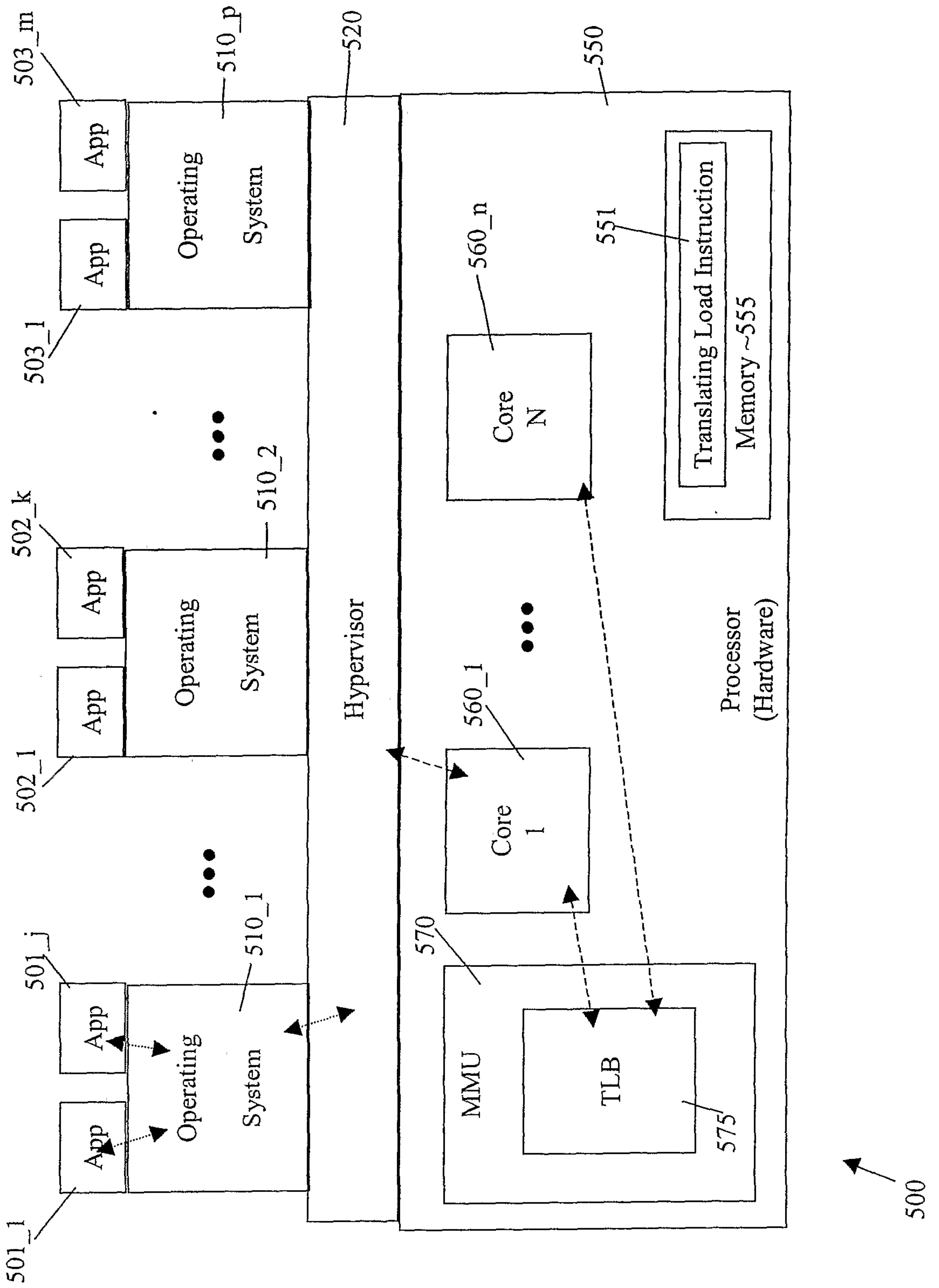
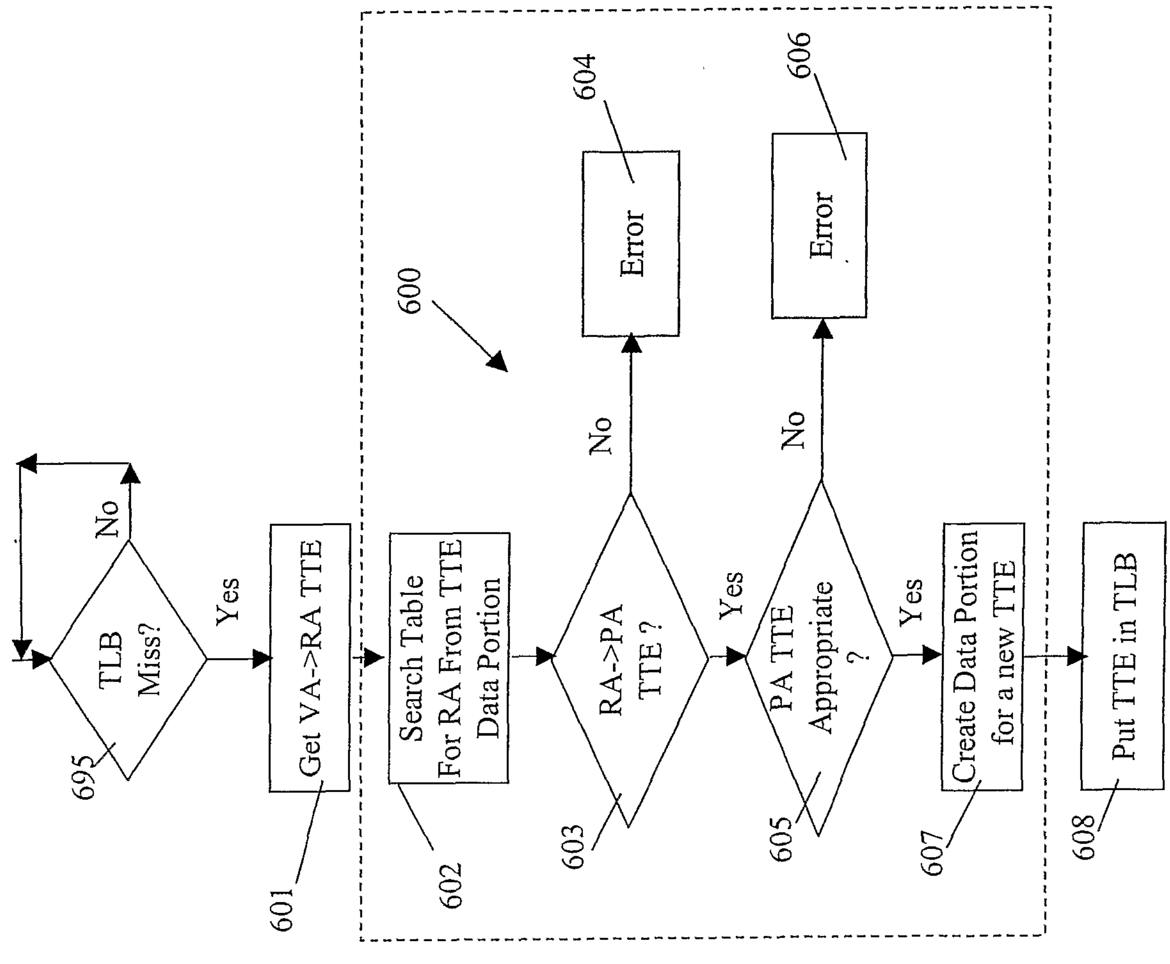

501_1
501_j
App
App
Operating System
510_1
502_1
502_k
App
App
Operating System
510_2
503_1
503_m
App
App
Operating System
510_p
Hypervisor
520
Processor (Hardware)
550
Core 1
560_1
Core N
560_n
MMU
570
TLB
575
Translating Load Instruction
551
Memory ~555
500

FIG. 5

Fig. 6


TLB Miss?
No
Yes
695
Get VA->RA TTE
601
Search Table For RA From TTE Data Portion
602
RA->PA TTE ?
603
No
Error
604
600
Yes
PA TTE Appropriate ?
605
No
Error
606
Yes
Create Data Portion for a new TTE
607
Put TTE in TLB
608

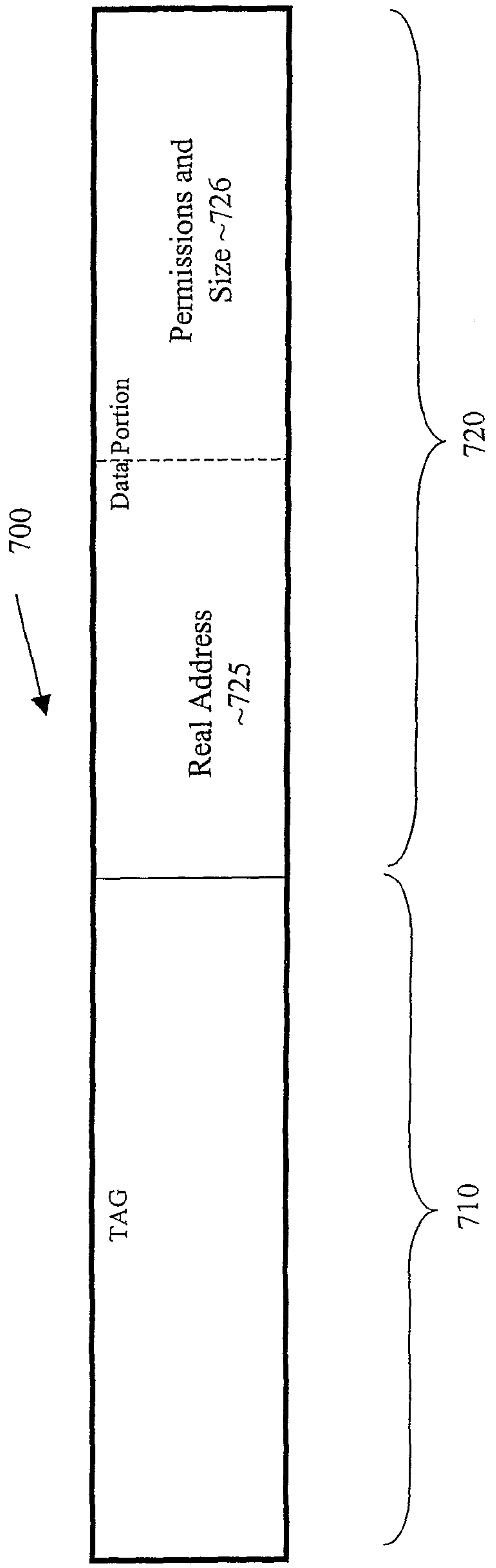

700
Permissions and Size ~726
Data Portion
Real Address ~725
TAG
720
710

Fig. 7A

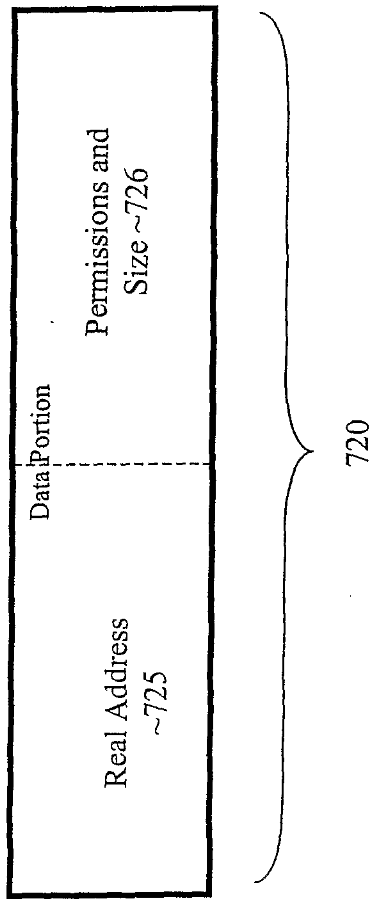

Permissions and Size ~726
Data Portion
Real Address ~725
720

Fig. 7B

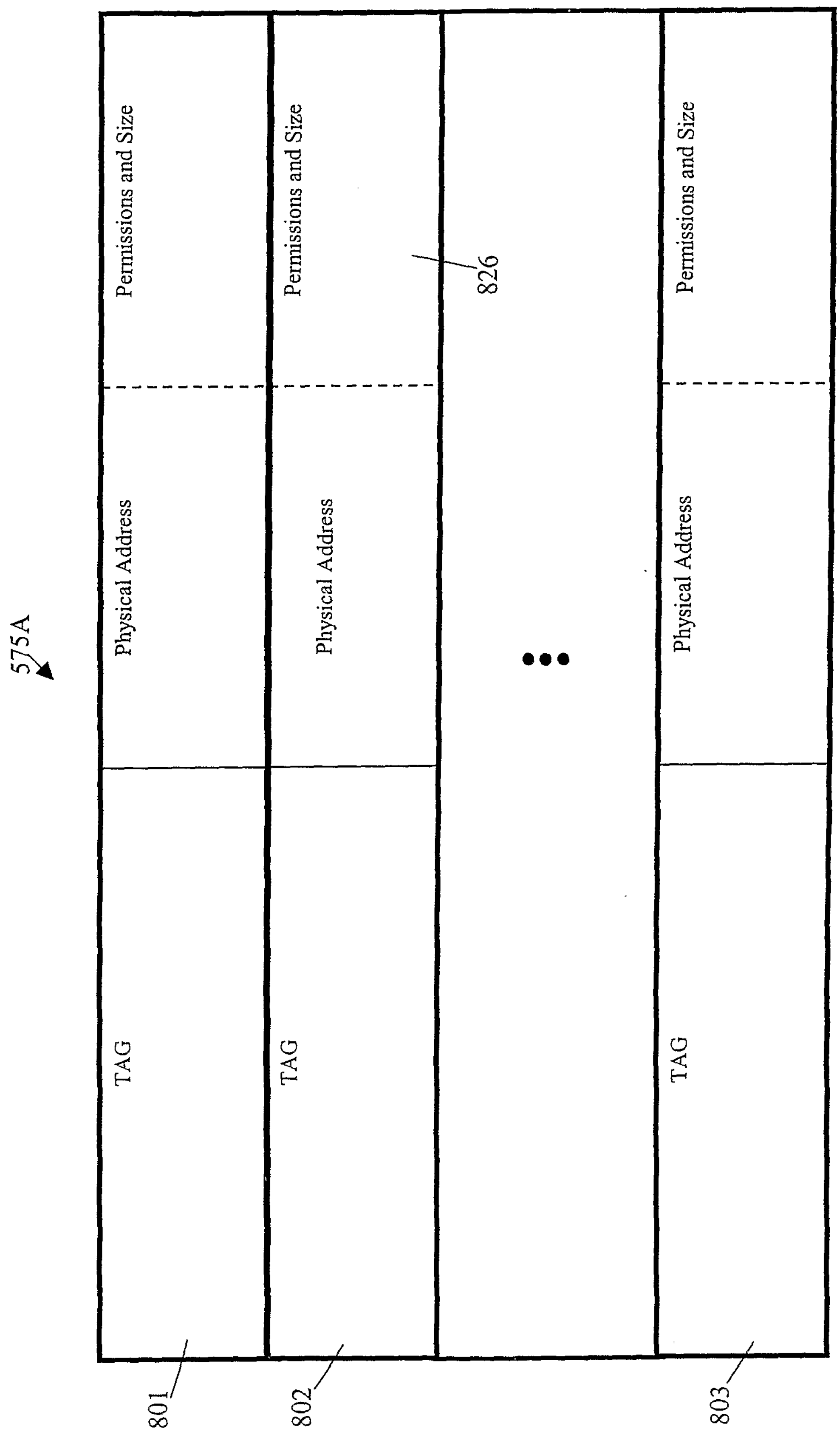

575A
Permissions and Size
Physical Address
TAG
Permissions and Size
Physical Address
TAG
826
Permissions and Size
Physical Address
TAG
801
802
803

FIG. 8

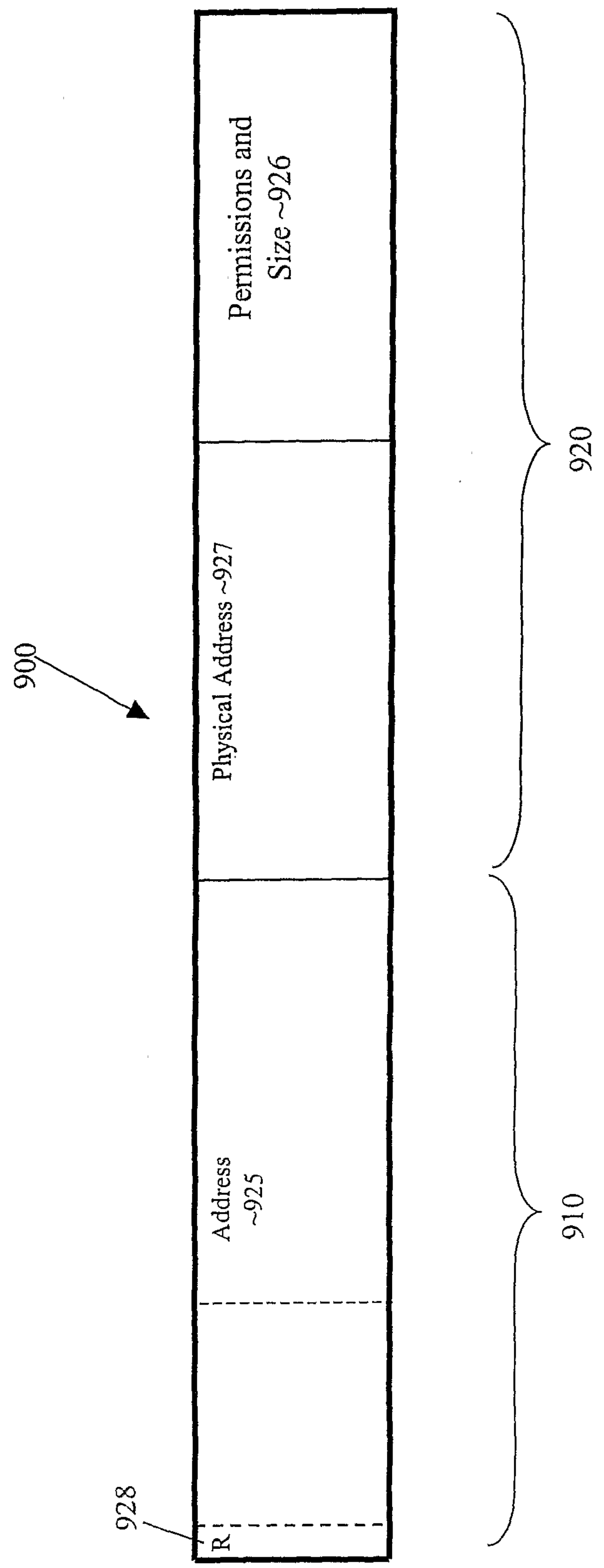

Permissions and Size ~926
Physical Address ~927
Address ~925
R
928
900
920
910

Fig. 9

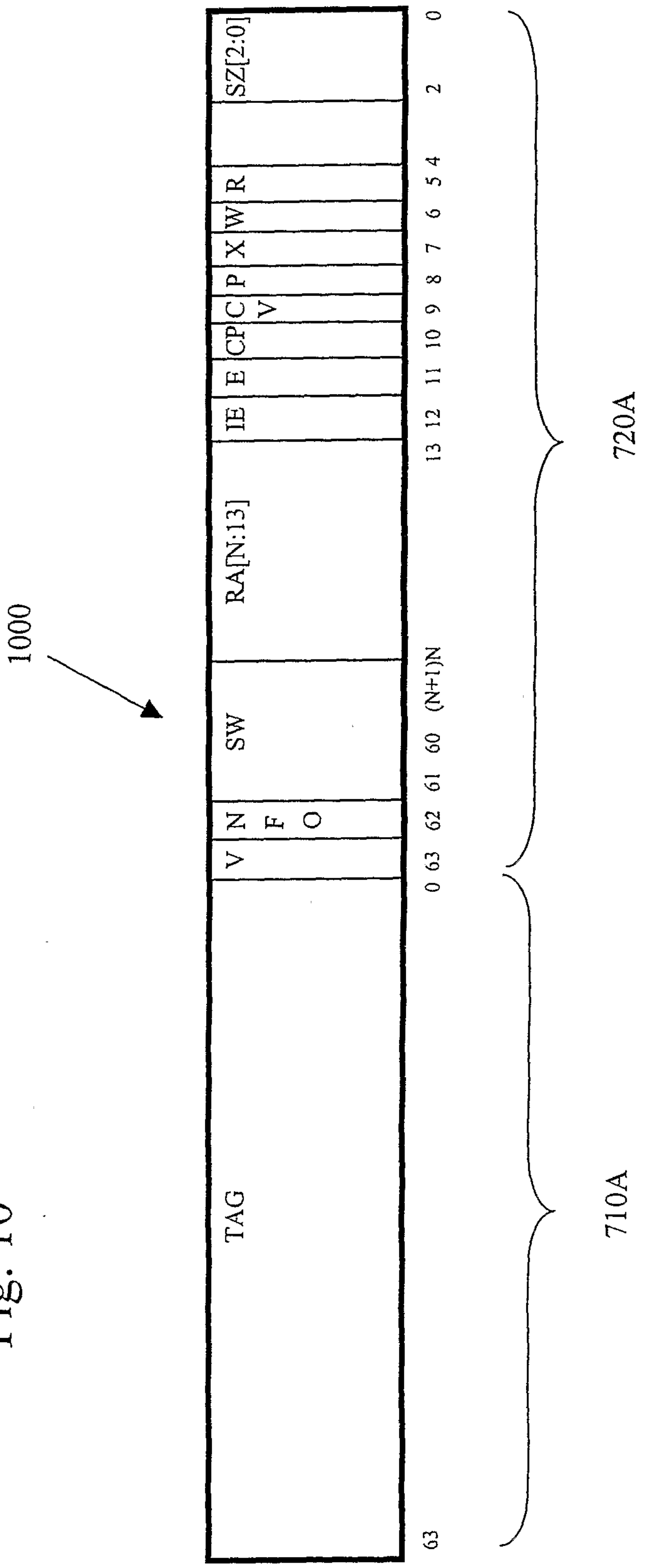
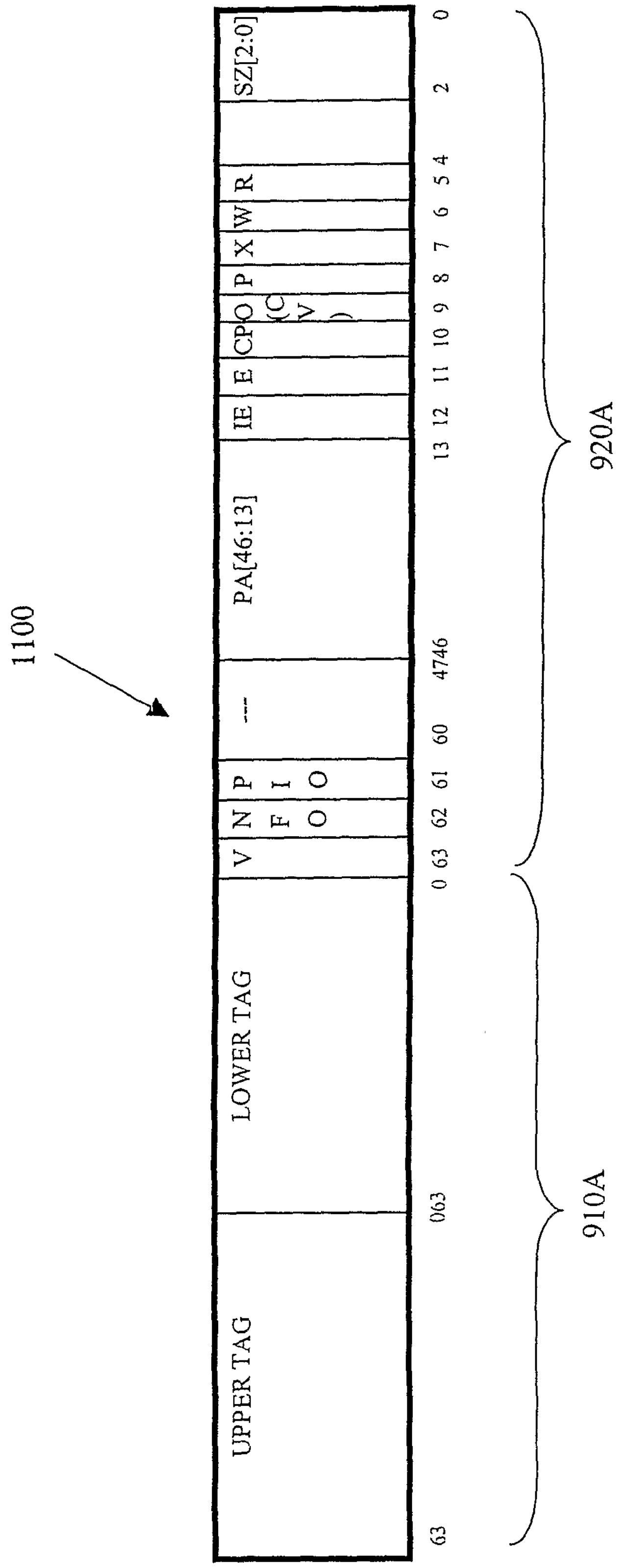

SZ[2:0]
R
W
X
P
C V
CP
E
IE
RA[N:13]
SW
N F O
V
TAG
0
2
4
5
6
7
8
9
10
11
12
13
(N+1)N
60
61
62
63
0
63
720A
710A
1000

Fig. 10

Fig. 11


1100
SZ[2:0]
R
W
X
P
IE
E
PA[46:13]
V
N
P
FIFO
LOWER TAG
UPPER TAG
920A
910A

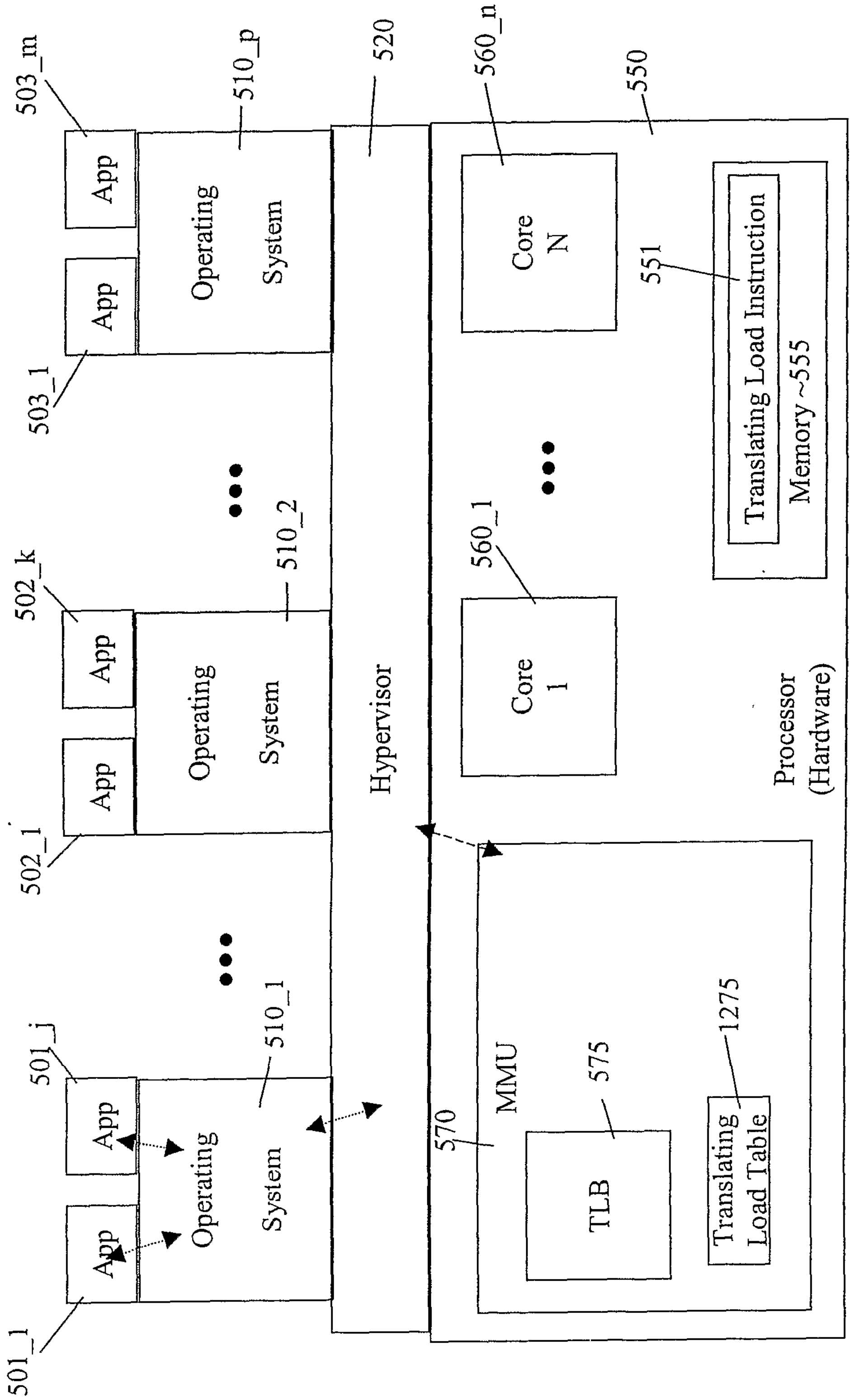

503_m
503_1
App
App
Operating System
510_p
502_k
502_1
App
App
Operating System
510_2
501_j
501_1
App
App
Operating System
510_1
Hypervisor
520
560_n
Core N
560_1
Core 1
550
551
Translating Load Instruction
Memory ~555
Processor (Hardware)
MMU
570
575
TLB
1275
Translating Load Table

FIG. 12

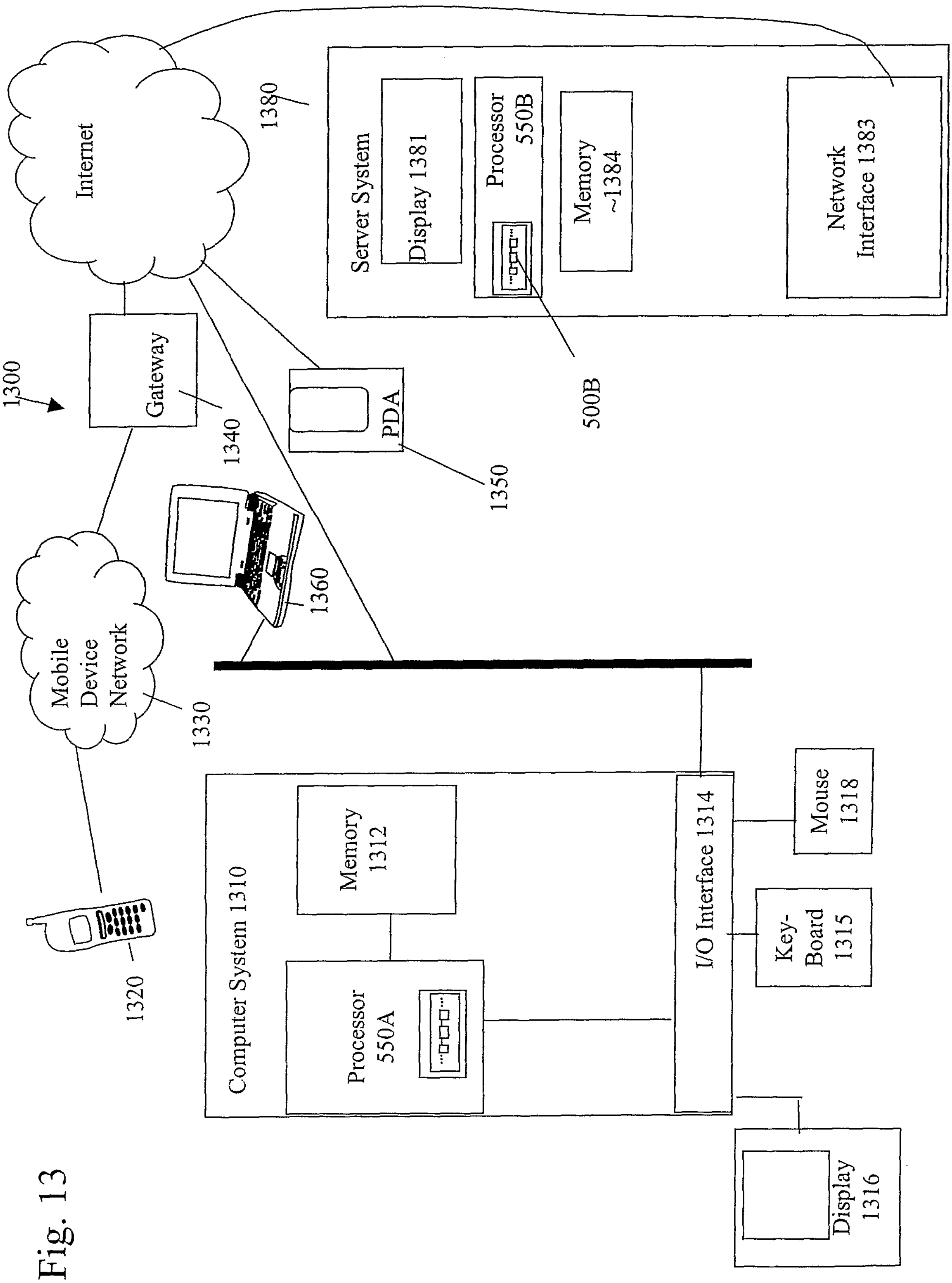

Internet
1380
Server System
Display 1381
Processor
550B
Memory
~1384
Network
Interface 1383
500B
1300
Gateway
1340
PDA
1350
1360
Mobile
Device
Network
1330
1320
Computer System 1310
Memory
1312
Processor
550A
I/O Interface 1314
Mouse
1318
Key-
Board
1315
Display
1316

Fig. 13

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US05/18252

A. CLASSIFICATION OF SUBJECT MATTER

IPC: **GO6F 12/08**

USPC: 711/207

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. : 711/202, 203, 206, 207

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 practicable, search terms used)
USPAT, PGPUB, USOCR, FPRS, EPO, JPO, IBM TDB, Derwent

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2002/0062434 A1 (CHAUVEL et al.) 23 May 2002, entire document.	1-30
A	US 2002/0065989 A1 (CHAUVEL et al.) 30 May 2002, entire document.	1-30
A	US 2005/0188175 A1 (CHAUVEL et al.) 25 August 2005, entire document.	1-30
A	US 2005/0188176 A1 (CHAUVEL et al.) 25 August 2005, entire document.	1-30
A	US 6,742,103 B2 (CHAUVEL et al.) 25 May 2004, entire document.	1-30
A	US 6,742,104 B2 (CHAUVEL et al.) 25 May 2004, entire document.	1-30

☐ Further documents are listed in the continuation of Box C. ☐ 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 application or patent 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 04 September 2006 (04.09.2006)	Date of mailing of the international search report 31 OCT 2006
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner of Patents P.O. Box 1450 Alexandria, Virginia 22313-1450 Facsimile No. (571) 273-3201	Authorized officer Reginald Bragdon Telephone No. 571-272-2100