Title: MANAGING FLASH MEMORY IN COMPUTER SYSTEMS

Abstract: Embodiments of a circuit are described. This circuit includes an instruction fetch unit to fetch instructions to be executed which are associated with one or more virtual addresses, a translation lookaside buffer (TLB), which includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell.
MANAGING FLASH MEMORY IN COMPUTER SYSTEMS

Inventors: Steven Woo, Brian Tsang, William Ng, and Ian Shaeffer

TECHNICAL FIELD

[001] The present embodiments relate to computer systems. More specifically, the present embodiments relate to circuits and methods for managing Flash memory in computer systems.

BRIEF DESCRIPTION OF THE FIGURES

[002] FIG. 1 is a block diagram illustrating an embodiment of a computer system.
[003] FIG. 2 is a graph illustrating memory and processor trends.
[004] FIG. 3 is a block diagram illustrating an embodiment of a memory device.
[005] FIG. 4A is a flow chart illustrating an embodiment of a process for performing a conversion.
[006] FIG. 4B is a flow chart illustrating an embodiment of a process for performing a conversion.
[007] FIG. 5A is a block diagram illustrating an embodiment of a computer system.
[008] FIG. 5B is a block diagram illustrating an embodiment of a computer system.
[009] FIG. 5C is a block diagram illustrating an embodiment of a computer system.
[010] FIG. 6A is a block diagram illustrating an embodiment of a Translation Lookaside Buffer (TLB) or a page table.
[011] FIG. 6B is a block diagram illustrating an embodiment of an entry in a TLB or a page table.
[012] FIG. 7 is a flow chart illustrating an embodiment of a process for exclusive caching.
[013] FIG. 8 is a flow chart illustrating an embodiment of a process for inclusive caching.
[014] FIG. 9 is a block diagram illustrating an embodiment of a computer system.
[015] FIG. 10 is a block diagram illustrating an embodiment of a system.
[016] Table 1 provides characteristics of memory devices in a memory hierarchy.
[017] Table 2 provides characteristics of memory devices in a memory hierarchy.

[018] Note that like reference numerals refer to corresponding parts throughout the drawings.

**DETAILED DESCRIPTION**

[019] The following description is presented to enable any person skilled in the art to make and use the disclosed embodiments, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present description. Thus, the present description is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

[020] Embodiments of a circuit, an integrated circuit that includes the circuit (such as a processor), a memory controller that includes the circuit, a system (such as a computer system) that includes the integrated circuit and/or the memory controller, an operating system, a compiler, an application program, and/or a technique for managing a memory hierarchy are described. In this system, a memory (such as Flash memory, phase-change memory, or another memory technology) having a storage cell with a retention time that decreases as operations are performed on the storage cell is included in a memory hierarchy. This memory hierarchy may include different types of memory devices having a range of latency, communication bandwidth, and cost per bit of storage. Moreover, the memory may have characteristics (such as latency, communication bandwidth, and cost per bit of storage) that are between characteristics of two other types of memory devices in the memory hierarchy (such as dynamic random access memory or DRAM and a hard disk drive or HDD).

[021] In some embodiments of the system Flash memory is used to augment DRAM in main memory. Moreover, DRAM may be used as a cache for data stored in the Flash memory. In this way, accessing or programming of the Flash memory may be reduced. Note that movement of data between DRAM and the Flash memory may be based on a variety of policies or protocols, including: exclusive caching, inclusive caching, and/or a hybrid approach. Additionally, DRAM may communicate to Flash memory or to the HDD (thereby bypassing Flash memory) when necessary to remedy contention or endurance issues associated with the Flash memory.
Moreover, data movement policies may be managed by: the integrated circuit (such as the processor), the memory controller, an operating system, an application program (for example, a compiler may configure the application program to manage data movement in the memory hierarchy). In some embodiments, a component in the system includes or stores a Translation Lookaside Buffer (TLB) and/or a page table (which may be stored in DRAM), either of which may convert virtual addresses into physical addresses, which include entries for physical addresses that are dedicated to DRAM as well as entries for physical addresses that are dedicated to the memory.

By including the memory in the memory hierarchy, the following embodiments may allow the memory capacity in the system to be increased, thereby providing increased performance and/or a lower ratio of the cost of memory to performance. Moreover, by increasing the memory capacity, a wider range of features than would otherwise be possible (because of cost constraints) may be included in the system.

Embodiments of the circuit, the integrated circuit, the memory controller, the system, the operating system, the application program, the compiler, and/or the technique may be used in or in conjunction with systems that include different types of memory, such as: volatile memory, non-volatile memory, DRAM, static random access memory (SRAM), read-only memory (ROM), erasable programmable read-only memory (EPROM), electrically erasable programmable read-only memory (EEPROM), Flash memory (such as NAND Flash memory or NOR Flash memory), solid-state memory, and/or another type of memory (such as phase-change memory). Moreover, for a given type of memory, these techniques may be used with different memory technologies or technology generations (which may use different power supply voltages). For example, these techniques may be used in systems that include: extreme data rate (XDR), double-data rate (DDR), graphics double-data rate (GDDR) and/or synchronous DRAM, such as: DDR2, DDR3, DDRx, GDDR1, GDDR3, GDDR5, and/or mobile DDR.

Consequently, systems and/or components that use these techniques may be included in a wide variety of applications, such as: memory systems, memory modules, operating systems, application programs, compilers, desktop or laptop computers, computer systems (such as servers and/or workstations), hand-held or portable devices (such as personal digital assistants and/or cellular telephones), set-top boxes, home networks, and/or video-game devices. For example, a storage device (such as a memory module) that includes Flash memory may be included in computer main memory. Moreover, one or more of these embodiments may be included in a communication system, such as: serial or parallel links,
metropolitan area networks (such as WiMax), local area networks (LANs), wireless local area networks (WLANs), personal area networks (PANs), and/or wireless personal area networks (WPANs).

[026] While a variety of low-cost memory technologies may utilize the techniques described below, in the discussion that follows the inclusion of Flash memory in the memory hierarchy of a computer system is used as an illustrative example.

[027] We now describe embodiments of a system, such as a computer system, that include a memory hierarchy. FIG. 1 presents a block diagram illustrating an embodiment of a computer system 100. This computer system includes one or more processors (or processor cores) 110 that are coupled to additional components by signal lines (or a communication bus) 116. These additional components may include: a memory controller 118, DRAM 120 (which is sometimes referred to as main memory), an optional graphics processor 122, an input/output (I/O) controller 124, and/or one or more hard disk drives (HDDs) 126. Moreover, the one or more processors 110 may include one or more memory caches, such as L1 cache 112 and L2 cache 114. Note that the one or more caches, DRAM 120, and the one or more HDDs 126 constitute a memory hierarchy. Relative characteristics of the different types of memory devices in this memory hierarchy are summarized in Table 1.

<table>
<thead>
<tr>
<th>Component</th>
<th>Latency (CPU clock cycles)</th>
<th>Bandwidth</th>
<th>Cost</th>
<th>Access Size (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registers</td>
<td>1</td>
<td>Extremely High</td>
<td>Extremely High</td>
<td>4-8</td>
</tr>
<tr>
<td>L1 Cache</td>
<td>1-2</td>
<td>Very High</td>
<td>Very High</td>
<td>16-32</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>4-12</td>
<td>High</td>
<td>High</td>
<td>64-128</td>
</tr>
<tr>
<td>DRAM</td>
<td>100-1000</td>
<td>Medium</td>
<td>Medium</td>
<td>32-64 (per module)</td>
</tr>
<tr>
<td>HDD</td>
<td>&gt; 15,000,000</td>
<td>Very Low</td>
<td>Very Low</td>
<td>4096</td>
</tr>
</tbody>
</table>

Table 1

[028] Typically, data is moved between different memory devices or levels in the memory hierarchy based, at least in part, on the relative characteristics of the memory devices. In existing computer systems, cache lines (typically 64 or 128 bytes of data) may be moved between the one or more caches and DRAM 120 using hardware in the one or more processors 110. For example, 'old' cache lines (such as those least likely to be used) may be moved from one of the caches to DRAM 120 when more room is needed in one of the caches.

[029] Additionally, operating-system pages (typically 4096 bytes of data) may be moved between DRAM 120 and the one or more HDDs 126 using software (typically, in the
operating system) and/or hardware. Existing operating systems typically use so-called 'demand paging' to bring pages from one or more HDDs 126 to DRAM 120. (However, some application programs may perform so-called 'pre-paging' or pre-fetching of pages.) Note that 'old' pages (such as pages that are least likely to be used) may be moved from DRAM 120 to one or more HDDs 126 when main-memory capacity is exceeded.

[030] In many computer systems (such as servers), performance may be a strong function of the memory capacity. Moreover, inadequate memory capacity may result in frequent HDD accesses and, thus, lower performance because of the higher latency and lower bandwidth of HDDs versus DRAM. For example, when an existing blade-server computer system was tested using a mix of application programs that are representative of current workloads, the run time increased from around an hour to 141 hours when there was insufficient memory capacity. Unfortunately, simply increasing the memory capacity is not always an option because of: cost constraints, power dissipation, a limited number of memory slots for DRAM, and/or processor trends.

[031] In particular, as processors continue to evolve to include a growing number of processor cores, the demand for memory capacity may continue to grow. However, it may become more challenging to provide this memory capacity. This is shown in FIG. 2, which presents a graph 200 illustrating memory and processor trends. In particular, the number of cores per processor is doubling every 1.5-2 years while the DRAM density is doubling every 3 years. Consequently, because each core has an associated memory-capacity requirement, main-memory capacity needs are growing much faster than memory density.

[032] Flash memory, which has a low cost per bit, as well as reduced power consumption in many modes of operation, may provide additional cost-effective memory capacity when used in main memory (e.g., to augment DRAM) in the memory hierarchy in a computer system. Moreover, by including Flash memory in main memory, HDD accesses may be reduced, thereby improving the performance of the computer system. Table 2 provides relative characteristics of memory devices in such a memory hierarchy.

<table>
<thead>
<tr>
<th>Component</th>
<th>Latency (CPU clock cycles)</th>
<th>Bandwidth</th>
<th>Cost</th>
<th>Access Size (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registers</td>
<td>1</td>
<td>Extremely High</td>
<td>Extremely High</td>
<td>4-8</td>
</tr>
<tr>
<td>L1 Cache</td>
<td>1-2</td>
<td>Very High</td>
<td>Very High</td>
<td>16-32</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>4-12</td>
<td>High</td>
<td>High</td>
<td>64-128</td>
</tr>
<tr>
<td>DRAM</td>
<td>100-1000</td>
<td>Medium</td>
<td>Medium</td>
<td>32-64</td>
</tr>
</tbody>
</table>
Table 2

<table>
<thead>
<tr>
<th>NAND Flash Read</th>
<th>15,000</th>
<th>Low</th>
<th>Low</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND Flash Erase</td>
<td>6,000,000</td>
<td>Low</td>
<td>Low</td>
<td>4096</td>
</tr>
<tr>
<td>NAND Flash Program</td>
<td>2,700,000</td>
<td>Low</td>
<td>Low</td>
<td>4096</td>
</tr>
<tr>
<td>HDD</td>
<td>&gt; 15,000,000</td>
<td>Very Low</td>
<td>Very Low</td>
<td>4096</td>
</tr>
</tbody>
</table>

Note that Flash memory bridges the large performance gap between HDD and DRAM. Compared to DRAM, Flash memory has: higher latency, lower bandwidth, and lower cost per bit. However, Flash memory differs from many other types of memory devices in that storage cells are erased before they are reprogrammed. This erase operation is usually performed on at least a group of storage cells, such as: a page of storage cells, a string of storage cells, and/or a block of storage cells. Consequently, erasing storage cells can be a time-consuming process. In particular, when a block of Flash memory is being erased, it ties up the entire bank of memory until the operation is complete, preventing other data from being read or written. Therefore, the time scales or latencies associated with Flash memory accesses are typically asymmetrical, with erase and program operations taking much longer than read operations.

Based on these characteristics, the movement of data between Flash memory and DRAM may be managed to increase performance and/or the ratio of the cost of memory to performance. For example, as described further below, DRAM may be used as a cache for Flash memory, and the computer system may avoid accessing Flash memory as much as possible (such as when the only alternative is to access an HDD). Moreover, as noted previously, a variety of data-movement policies and protocols may be selectively used, such as: exclusive caching (which is described further below with reference to FIG. 7), inclusive caching (which is described further below with reference to FIG. 8), and/or hybrid caching (such as, sometimes inclusive caching and sometimes exclusive caching). These data-movement policies and protocols may be selected and/or dynamically adapted based on the data type and usage characteristics.

We now describe embodiments of Flash memory devices. FIG. 3 presents a block diagram illustrating an embodiment of a memory device 300. This device includes a substrate 310, which may be p-type or n-type. Regions on the substrate 310 are doped (for example, using diffusion or implantation) to be a source 312 and a drain 314 in a field-effect
transistor. Moreover, the source 312 and the drain 314 may be p-type or n-type. Thus, the field-effect transistor may be PMOS or NMOS. Note that the source 312 and the drain 314 regions define a channel 316 having a voltage-dependent transconductance. In an illustrative embodiment, the memory device 300 is a NAND or NOR Flash memory device, with a p-type substrate and an n-type source and drain (e.g., an NMOS field-effect transistor).

[036] Memory device 300 includes a floating-gate insulator 318 and a floating gate 320 deposited above a surface of the substrate 310. As discussed below, the floating gate 320 may be used to store charge associated with information that is stored in the memory device 300. Note that the stored charge may correspond to binary information or multi-level information. Moreover, the memory device 300 includes a control-gate insulator 322 and a control gate 324 deposited above the floating gate 320.

[037] During operation, voltages are applied between the substrate 310 and the control gate 324 using terminals 326. In particular, during a program operation a large positive voltage may be applied to terminal 326-1 of a previously erased memory device 300 (see below) and charge carriers (such as electrons) may be attracted from the channel 316 towards the floating gate 320 and may traverse the floating-gate insulator 318. These charge carriers may be stored on the floating gate 320. In an exemplary embodiment, the charge carriers traverse an energy barrier associated with the floating-gate insulator 318 by hot-electron injection for a NOR-connected memory or field-assisted tunneling (which is henceforth referred to as Fowler-Nordheim tunneling) for a NAND-connected memory. Moreover, after the charge is stored, the terminal 326-1 may be set to zero volts or some other level such that charge flow no longer occurs through the memory device 300.

[038] Similarly, during an erase operation the positive voltage may be applied to terminal 326-2, and the charge stored on the floating gate 320 may be attracted toward the substrate 310 and may traverse the floating-gate insulator 318. In this way, the information stored on the memory device 300 may be erased. Moreover, once the charge on the floating gate 320 is removed, the terminal 326-2 may be set to zero volts or some other level such that charge flow no longer occurs through the memory device 300.

[039] During a read operation, a smaller positive voltage may be applied to terminal 326-1. In addition, a voltage may be applied between the source 312 and the drain 314 so that the transconductance of the memory device 300 may be determined or measured. Note that the transconductance is dependent on the stored charge on the floating gate 320, which allows the information stored on the memory device 300 to be determined. In NAND Flash memory embodiments, a group of memory devices, such as the memory device 300, are
coupled in series. In these embodiments, neighboring memory devices are pass gates while
the memory device 300 is read. Moreover, in NOR Flash memory embodiments a group of
memory devices, such as the memory device 300, are connected in parallel. In these
embodiments, each memory device 300 may be individually selected for reading.

[040] Note that the data retention time of memory device 300 is not infinite
however, as the charge leaking from floating gate 320 within the Flash memory device
eventually results in data loss. Furthermore, independent of the charge leakage problem, over
many program/erase cycles, transport of the charge across the floating-gate insulator 318 can
produce defects in the floating-gate insulator 318. These defects can lead to current-leakage
paths between the floating gate 320 and the substrate 310. This stress-induced leakage
current can degrade the stored charge and, thus, the stored information (for example, by
degradng the detectable difference between logic '1' and '0' levels), thereby reducing the
data retention time associated with the memory device 300, a phenomenon commonly
referred to as 'wear out.' In particular, for a small number of program/erase cycles, the
retention time can be many years. However, as the number of program/erase cycles
increases, retention times progressively decrease due to charge leakage from the floating gate
320. Note that the maximum number of program/erase cycles a given memory device, such
as a Flash memory device, can endure and still meet an acceptable data retention time is
commonly referred to as the 'endurance' of the memory device.

[041] Moreover, the defects in the floating-gate insulator 318 can eventually cause
failure of the memory device 300, because the floating gate 320 is no longer well insulated
from the substrate 310, e.g., the retention time may be too small to allow the stored
information to be reliably recovered. Thus, the memory device 300 can eventually become
volatile.

[042] Another reliability characteristic of some Flash memory devices is read
disturb. As storage cells (such as pages in a string) are read, data in other storage cells may
be gradually disturbed. Eventually, this can result in a read failure. Moreover, read disturb is
worse as the number of program/erase cycles increases. For example, a typical current Flash
memory device may be able to read 100,000 times to the same string before failing when the
memory device is new and 10,000 times before failing after the memory device has 10,000
program/erase cycles.

[043] Consequently, in addition to basing protocols or policies on differences in the
performance characteristics of Flash memory and DRAM (such as much larger write latency
than read latency), because of issues such as endurance and/or read disturb it may be
advantageous to only access Flash memory in a computer system when needed (such as when the alternative is accessing an HDD). Moreover, when Flash memory is accessed, read operations may be emphasized over write operations (which may be performed infrequently). Additionally, as described below, Flash memory in the memory hierarchy in the computer system may be intentionally bypassed (for example, data may be written from DRAM to an HDD rather than to Flash memory) because of endurance issues.

[044] In the preceding discussion, as an illustration memory device 300 has been described as a NAND Flash memory device. As noted previously, in other embodiments memory device 300 may be a NOR Flash memory device. In these embodiments, charge may be stored on the floating-gate layer 320 through hot-electron injection.

[045] Note that the memory device 300 may include fewer components or additional components. Moreover, two or more components in the memory device 300 may be combined into a single component and/or the position of one or more components may be changed. In some embodiments, the memory device 300 is included in one or more integrated circuits on one or more semiconductor die.

[046] We now describe embodiments of processes for communicating and executing commands in a memory hierarchy that includes Flash memory. FIG. 4A presents a flow chart illustrating an embodiment of a process 400 for performing a conversion, which may be performed by a device (which may be implemented in hardware and/or in software). During operation, the device receives a virtual address (410). Next, the device converts the virtual address to a physical address using a translation lookaside buffer (TLB) (412), which includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell (such as Flash memory). Note that the DRAM and the memory may be included in a memory hierarchy.

[047] FIG. 4B presents a flow chart illustrating an embodiment of a process 420 for performing a conversion, which may be performed by a device (such as a processor). During operation, the device receives an instruction which is associated with one or more virtual addresses (430). Next, the device executes the instruction, where executing the instruction includes accessing a translation lookaside buffer (TLB) that facilitates converting virtual addresses into physical addresses (432). Note that the TLB includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time
that decreases as operations are performed on the storage cell (such as Flash memory). Moreover, the DRAM and the memory may be included in a memory hierarchy. Also note that the TLB acts as a cache for a larger page table that resides in main memory.

[048] In some embodiments of the processes 400 (FIG. 4A) and 420 there may be fewer or additional operations. Moreover, two or more operations can be combined into a single operation, and/or a position of one or more operations may be changed.

[049] We now describe embodiments of a computer system that includes a memory hierarchy with DRAM and Flash memory. In this memory hierarchy, Flash memory and DRAM are managed together to form a combined DRAM-and-Flash-memory main memory. In some embodiments, DRAM is used as a cache for Flash memory.

[050] This combination of DRAM and Flash memory may allow maximum leverage of existing infrastructure (processors, operating systems, and/or compilers), which may ease adoption of this memory hierarchy. Consequently, in some embodiments an operating system and/or an application program may manage data movement within the memory hierarchy, such as between DRAM and Flash memory. For example, the operating system and/or the application program may include or generate a data structure, such as a page table, that includes a virtual-to-physical-address conversion for DRAM and for Flash memory that allows the operating system and/or the application program to determine where data is located. Moreover, as described further below with reference to FIGs. 6A and 6B, in some embodiments there may be separate entries or separate subsets of the page table that are dedicated to addresses associated with DRAM and addresses associated with Flash memory. Note that in some embodiments a compiler may implement or generate such a conversion when the application program is converted into code that can be executed by a processor in the computer system.

[051] Additionally and/or separately, in some embodiments hardware is used to manage data movement within the memory hierarchy, such as between DRAM and Flash memory. For example, one or more processors (or processor cores) and/or a memory controller may include one or more circuits and/or may implement a TLB. This TLB may include a virtual-to-physical-address conversion for DRAM and for Flash memory that allows the one or more processors and/or the memory controller to determine where data is located (for example, Flash memory addresses may be cached in the TLB). For example, the TLB may facilitate converting and lookup techniques, such as: tag, index, and offset. Moreover, as described further below with reference to FIGs. 6A and 6B, in some embodiments there may be separate entries or separate subsets of the TLB that are dedicated
to addresses associated with DRAM and addresses associated with Flash memory. Note that
entries from a page table are cached in the TLB.

[052] Consequently, data coherence and consistency in the memory hierarchy as
well as the movement of data (for example, the virtual-address-to-physical-address
conversion) may be managed using software (such as the operating system, the application
program, and/or the compiler) and/or hardware (such as the one or more processors and/or
the memory controller). This is shown in FIGs. 5A-5C, which illustrate several embodiments
of computer systems.

[053] In particular, FIG. 5A presents a block diagram illustrating an embodiment of
a computer system 500. In this system, a TLB 512 is included in at least one of the one or
more processors 110 and a page table 514 is included or stored in DRAM 120. As noted
previously, data movement (such as the movement of one or more pages) between DRAM
120 and Flash memory 510 may be managed using hardware (such as the one or more
processors 110) and/or software (such as the operating system). For example, interrupt
handlers may manage the movement of data between DRAM 120 and Flash memory 510.
Note that data movement in the memory hierarchy may be facilitated using a virtual-address-
to-physical-address conversion in TLB 512 and/or in page table 514.

[054] In computer system 500, memory controller 118 is coupled to DRAM 120 and
Flash memory 510 via separate signal lines (or memory buses) 516. However, in some
embodiments, such as computer system 530 in FIG. 5B, the memory controller 118 is
coupled to DRAM 120 and Flash memory 510 via a common signal line 516-3.

[055] Moreover, TLB 512 may reside in the one or more processors 110 or between
the one or more processors 110 and main memory (DRAM 120 and Flash memory 510). For
example, as shown in computer system 560 in FIG. 5C, TLB 512 may be included or may
reside in memory controller 118.

[056] Note that the computer systems 500 (FIG. 5A), 530 (FIG. 5B), and/or 560 may
include fewer components or additional components. For example, the one or more
processors 110 and/or the memory controller 118 may monitor the number of write
operations (and more generally, the number of write operations and the number of read
operations) to one or more storage cells in Flash memory 510, thereby allowing the
endurance and/or read disturb to be monitored. Alternatively, a separate component may
monitor the write and/or read operations to the Flash memory 510. Using this information,
communication with Flash memory 510 can be managed.
In some embodiments, I/O controller 124 is coupled to memory controller 118 (as opposed to signal lines 116) and/or optional graphics processor 122 is coupled to memory controller 118 or I/O controller 124 (as opposed to signal lines 116). Moreover, in some embodiments the functionality of the memory controller 118 is implemented in at least one of the processors 110. Consequently, in some embodiments there may not be a memory controller 118.

In some embodiments, cache tags are stored in a variety of locations in the computer system, including: the one or more processors 110, the memory controller 118, and/or in one or more buffers on one or more memory modules in DRAM 120. Additionally, in some embodiments computer systems 500 (FIG. 5A), 530 (FIG. 5B), and/or 560 are configured to support hot swapping of Flash memory or DRAM modules, thereby allowing the memory capacity and/or the memory mix to be dynamically changed without first shutting down the computer system. Note that two or more components in these computer systems may be combined into a single component and/or the position of one or more components may be changed.

We now describe embodiments of a TLB and/or a page table for use in a memory hierarchy that includes DRAM and Flash memory. FIG. 6A presents a block diagram illustrating an embodiment 600 of a Translation Lookaside Buffer (TLB) or a page table. This TLB or page table may include a pointer hierarchy of entries 610, which reference other entries in the TLB or page table.

FIG. 6B presents a block diagram illustrating an embodiment 650 of an entry 610-16 in the TLB or page table. This entry may include information associated with one or more pages of data in DRAM and information associated with one or more pages of data in Flash memory. For example, the DRAM information may include: status A 660, protection A 662, other 664, and conversion 666. Similarly, the Flash-memory information may include: status B 668, protection B 670, other 672, and conversion 674. For example, 'status' may indicate if a conversion is valid (is data in DRAM and/or in Flash memory); 'protection' may indicate attributes such as read only (e.g., only the kernel may modify this data); 'other' may indicate if the data has been recently used or other attributes that can be used to adjust performance; and 'conversion' may include a virtual-address-to-a-physical-address conversion.

In some embodiments, DRAM information is used if data is in DRAM, and Flash-memory information is used if the data is in Flash memory. Additionally, in some embodiments, when one or more pages are in Flash memory but not in DRAM, a
'conversion' entry may be set to look like the one or more pages are on swap device. In these embodiments, a swap handler can check to see if the one or more pages should be allocated to or from the Flash memory (such as to an HDD).

[062] Moreover, in some embodiments the addresses space in the memory hierarchy may be increased when Flash memory is included. Consequently, there may be separate addresses associated with Flash memory and separate addresses associated with DRAM. Alternatively, addresses may be allocated as needed, in which case the addresses space (and thus the TLB or the page table) may have a variable size.

[063] Note that in some embodiments of FIGs. 6A and 6B there may be fewer or additional components. Moreover, two or more components can be combined into a single component, and/or a position of one or more components may be changed. For example, the TLB and/or the page table may support different pages sizes. Additionally, in some embodiments one TLB and/or page table is used for Flash memory and another TLB and/or page table is used for DRAM. Note that in embodiments with inclusive caching, where data is in DRAM and Flash, the TLB and/or the corresponding page table entry may indicate that the virtual address converts to physical addresses in both the DRAM and the Flash memory.

[064] We now describe embodiments of data-movement policies and protocols in computer systems that include Flash memory and DRAM as main memory in a memory hierarchy. These policies and/or protocols may be used to determine when to move data to and/or from the Flash memory. As noted previously, based on the performance characteristics of Flash memory, as well as due to issues such as endurance and read disturb, it may be advantageous to only access Flash memory in a computer system when needed (such as when the alternative is accessing an HDD). Moreover, when Flash memory is accessed, read operations may be emphasized over write operations (which may be performed infrequently). In some embodiments, different policies may be used during read operations versus write operations to Flash memory. For example, different data sizes or granularities may be used when reading from Flash memory as opposed to when writing to Flash memory to account for the different access characteristics (or latencies) of these operations in Flash memory devices. Additionally, Flash memory in the memory hierarchy in the computer system may be intentionally bypassed (for example, data may be written from DRAM to an HDD rather than to Flash memory) because of endurance issues. Note that the data-movement policies and protocols may be selected and/or dynamically adapted based on the data type and usage characteristics.
[065] In some embodiments, DRAM is used as a cache for Flash memory. In general, data that is most likely to be used in the future may be stored in DRAM. However, even though it is hard to predict, a priori, which data will next be accessed at any point in time, a good predictor of such near-future accesses may be the most-recent past behavior. Consequently, data that has been accessed most recently and/or most frequently may be stored in DRAM, because there is a good chance that it will be used again in the near future.

[066] As noted previously, caching policies may include: exclusive caching, inclusive caching, and/or a hybrid caching approach. During exclusive caching, data is stored in either DRAM or Flash memory, but not both. This is shown in FIG. 7, which presents a flow chart illustrating an embodiment of a process 700 for exclusive caching, which may be performed using hardware or software (for example, using a device, such as a processor). During this process, the device receives data (710). Based on a storage decision 712, the data is stored in Flash memory (714) or is stored in DRAM (716). Subsequently, data may be migrated (718) from DRAM to Flash memory or from Flash memory to DRAM.

[067] Storage decision (712) and/or the migration of data (718) may be based on a variety of factors, including the data type and/or data usage. For example, the migration of data into DRAM may be a demand basis (such as when a processor needs the data) or may use pre-fetching techniques. This data may include operating-system drivers and/or a login portion of a shell. Note that when data is brought into DRAM on a demand basis, it can be brought in either from Flash memory or from an HDD into DRAM (for example, by bypassing the Flash memory). Additionally, data may be migrated out of DRAM and into Flash memory when it is determined that this data is unlikely to be accessed again in the near future. In this case, the data can also be migrated from DRAM directly into an HDD (for example, by bypassing the Flash memory). As noted previously, this bypassing may occur when there are too many write operations to the Flash memory and/or when endurance becomes an issue for the Flash memory.

[068] In some embodiments, a static policy is used for moving data between DRAM and Flash memory. For example, 'old' pages may be moved from DRAM to Flash memory or even to an HDD. However, in some embodiments monitoring hardware and/or software is used to actively determine when to move data between DRAM and Flash memory, and even what data to move. For example, when data is accessed by a processor, it may be moved to DRAM if it is not already there.

[069] Note that exclusively cached data in Flash memory may include read-only data (such as program code) and infrequently used data, while exclusively cached data in
DRAM may include modifiable or writeable data, as well as certain types of data (such as programmer-specified data or application-program/operating-system specific data like synchronization items or locks). For example, secure data may be stored in DRAM so that after the computer system powers down the data is lost. However, critical data or copies of critical data (such as logs of a database system) may be stored in Flash memory so that, if there is a loss of power, there is a persistent log from which to recover the database.

[070] During inclusive caching, data is stored in both DRAM and Flash memory. This is shown in FIG. 8, which presents a flow chart illustrating an embodiment of a process 800 for inclusive caching, which may be performed using hardware (for example, using a device, such as a processor). During this process, the device receives data (810). Next, this data is stored in Flash memory (812) and is stored in DRAM (814).

[071] While the data that is stored in DRAM is also stored in Flash memory in inclusive caching, the most recent version(s) of this data may be in DRAM. Moreover, DRAM can be write-back or write-through with respect to the Flash memory, and can be write-allocate or no write-allocate. In an exemplary embodiment, write-back and write-allocate are used when data is inclusively cached. Note that inclusively cached data in Flash memory and DRAM may include bursty data, such as read-only program code or data structures.

[072] In some embodiments of the processes 700 (FIG. 7) and 800 there may be fewer or additional operations. Moreover, two or more operations can be combined into a single operation, and/or a position of one or more operations may be changed.

[073] In a hybrid-caching policy or protocol, some data may be exclusively cached and some data may be inclusively cached. Note that such a policy or protocol may change over time. For example, inclusive caching may be used for program-code pages and operating-system pages, while exclusive caching may be used for application-program data. Alternatively, data may be initially inclusively cached and then may be exclusively cached, such as copy-on-write pages. For example, when such a page is to be modified, a copy may be stored in DRAM.

[074] As noted previously, because of the asymmetric nature of read and write operations in Flash memory, there may be different policies or protocols for these operations. For example, individual pages may be read, one at a time, on demand from Flash memory into DRAM. However, during a write operation, multiple pages (such as 16 pages) may be moved from DRAM into Flash memory at the same time. Note that this technique uses different access granularities for read and write operations to and from Flash memory. These
granularities may account for the difference between the read and write characteristics of Flash memory. Moreover, note that the basic unit of granularity may not be the operating-system page size (for example, the granularity could be a size between a cache-line size and the operating-system page size).

[075] Moreover, as noted previously, sometimes Flash memory may be bypassed. For example, if there are too many write operations to Flash memory (such as when too much data is being moved to Flash memory, if Flash memory is being overwhelmed, and/or if there are issues with Flash-memory endurance), some data may be written directly to an HDD or directly to DRAM (as described above in the discussion of exclusive caching).

[076] In some embodiments, data migration within the memory hierarchy (such as between Flash memory and DRAM) is facilitated by the operating system and/or the compiler. For example, the operating system and/or a compiled application program may provide hints about what is about to happen, such as when large blocks are read or written. Based on these hints, the associated data may be migrated within the memory hierarchy. Consequently, the data-movement policy or protocol may be based on these hints. Note that these hints may be implemented using processor instructions.

[077] We now describe several exemplary embodiments of data-movement policies and protocols. In one embodiment, read-only data is inclusively cached. Consequently, when a page with this data is allocated to DRAM it may be replicated to Flash memory. Moreover, when de-allocating the page from DRAM, the page is deleted from DRAM if the page is in Flash memory. Additionally, if the page is not in Flash memory, the page may be written back to Flash memory and then may be deleted from DRAM. However, if there are performance or endurance issues, Flash memory may be bypassed and the page may be written back to an HDD.

[078] In another embodiment for read/write data, this data is inclusively cached until a write operation, at which point the data may be exclusively cached in DRAM. For example, a page may be updated in DRAM and may be removed from Flash memory. Moreover, during de-allocation from DRAM, the page may be copied back to Flash memory or, if the page has been updated, Flash memory may be bypassed and the page may be stored to an HDD.

[079] In another embodiment for read/write data, this data is exclusively cached in DRAM. Moreover, during de-allocation from DRAM, the page may be copied back to Flash memory or, if the page has been updated, Flash memory may be bypassed and the page may be stored to an HDD.
In another embodiment for copy-on-write data, this data is exclusively cached in DRAM. Moreover, during de-allocation from DRAM, the page may be copied back to Flash memory or, if the page has been updated, Flash memory may be bypassed and the page may be stored to an HDD.

In one embodiment of a protocol for moving data between DRAM and Flash memory or an HDD, pages are requested to be moved into DRAM. However, if there is no space in DRAM, replacement priorities may ensure: that an old un-modified page (read-only data or read/write data) also exists in Flash memory; that an old un-modified page (read-only data or read/write data) does not exist in Flash memory; and/or that an old modified page exists in Flash memory.

In some embodiments, the determination about whether or not to copy data back to Flash memory is dynamically determined. For example, if pages are being written back to Flash memory and there is a high rate of page faults from an HDD to DRAM, it is possible that pages are being turned over. In this case, the policy may be modified to avoid writing back to Flash memory. However, if pages are not being written back to Flash memory and there is a low rate of page faults, then the policy may be changed and the pages may be written back to Flash memory.

Note that in some embodiments pages are compressed prior to being written to Flash memory. This may reduce the number of write operations performed on Flash memory.

In some embodiments, data is migrated based on a number of touches or hits within a time interval. For example, data may be migrated to DRAM any time a page is touched in Flash memory. Other triggering events for data migration may include: an absolute number of touches; some number of touches within the time interval; a direct command from operating system; and/or a hardware event.

Note that an operating system and/or one or more application programs may be included in computer systems. For example, FIG. 9 presents a block diagram illustrating an embodiment of a computer system 900. This computer system includes one or more processors 910, a communication interface 912, a user interface 914, and one or more signal lines 922 coupling these components together. Note that the one or more processing units 910 may support parallel processing and/or multi-threaded operation, the communication interface 912 may have a persistent communication connection, and the one or more signal lines 922 may constitute a communication bus. Moreover, the user interface 914 may include: a display 916, a keyboard 918, and/or a pointer 920, such as a mouse.
[086] Computer system 900 may include memory 924, which may include high speed random access memory and/or non-volatile memory. More specifically, memory 924 may include: ROM, RAM, EPROM, EEPROM, Flash memory, one or more smart cards, one or more magnetic disk storage devices or HDDs, and/or one or more optical storage devices. Memory 924 may store an operating system 926, such as SOLARIS, LINUX, UNIX, OS X, or Windows, that includes procedures (or a set of instructions) for handling various basic system services for performing hardware dependent tasks. Memory 924 may also store procedures (or a set of instructions) in a communication module 928. The communication procedures may be used for communicating with one or more computers and/or servers, including computers and/or servers that are remotely located with respect to the computer system 900.

[087] Memory 924 may also include one or more application programs 930 (of sets of instructions) and/or page table 932. Instructions in the application programs 930 in the memory 924 may be implemented in: a high-level procedural language, an object-oriented programming language, and/or in an assembly or machine language. The programming language may be compiled or interpreted, e.g., configurable or configured to be executed by the one or more processing units 910, using compiler 934. This compiler may configure one or more of the application programs 930 to implement a virtual-address-to-physical-address conversion, such as that contained in page table 932. Alternatively or additionally, operating system 926 may implement the virtual-address-to-physical-address conversion.

[088] Computer system 900 may include fewer components or additional components. Moreover, two or more components can be combined into a single component, and/or a position of one or more components may be changed. In some embodiments, the functionality of the computer system 900 may be implemented more in hardware and less in software, or less in hardware and more in software, as is known in the art.

[089] Although the computer system 900 is illustrated as having a number of discrete items, FIG. 9 is intended to be a functional description of the various features that may be present in the computer system 900 rather than as a structural schematic of the embodiments described herein. In practice, and as recognized by those of ordinary skill in the art, the functions of the computer system 900 may be distributed over a large number of servers or computers, with various groups of the servers or computers performing particular subsets of the functions. In some embodiments, some or all of the functionality of the computer system 900 may be implemented in one or more application specific integrated circuits (ASICs) and/or one or more digital signal processors (DSPs).
[090] Devices and circuits described herein may be implemented using computer aided design tools available in the art, and embodied by computer-readable files containing software descriptions of such circuits. These software descriptions may be: at behavioral, register transfer, logic component, transistor and layout geometry level descriptions. Moreover, the software descriptions may be stored on storage media or communicated by carrier waves.

[091] Data formats in which such descriptions may be implemented include, but are not limited to: formats supporting behavioral languages like C, formats supporting register transfer level RTL languages like Verilog and VHDL, formats supporting geometry description languages (such as GDSII, GDSIII, GDSIV, CIF, and MEBES), and other suitable formats and languages. Moreover, data transfers of such files on machine-readable media may be done electronically over the diverse media on the Internet or, for example, via email. Note that physical files may be implemented on machine-readable media such as: 4 mm magnetic tape, 8 mm magnetic tape, 3-1/2 inch floppy media, CDs, DVDs, and so on.

[092] FIG. 10 presents a block diagram illustrating an embodiment of a system 1000 that stores such computer-readable files. This system may include at least one data processor or central processing unit (CPU) 1010, memory 1024 and one or more signal lines or communication busses 1022 for coupling these components to one another. Memory 1024 may include high-speed random access memory and/or non-volatile memory, such as: ROM, RAM, EPROM, EEPROM, Flash memory, one or more smart cards, one or more magnetic disk storage devices or HDDs, and/or one or more optical storage devices.

[093] Memory 1024 may store a circuit compiler 1026 (such as computer program that generates a description of a circuit based on descriptions of portions of the circuit) and circuit descriptions 1028. Circuit descriptions 1028 may include descriptions for the circuits, or a subset of the circuits discussed above with respect to FIGs. 1-8. In particular, circuit descriptions 1028 may include circuit descriptions of: one or more processors 1030 (or sets of instructions), one or more memory controllers 1032, one or more circuits 1034, one or more storage cells 1036, and/or one or more TLBs 1038. For example, a TLB may include a virtual-address-to-physical-address conversion for DRAM and Flash memory in a memory hierarchy in a computer system, such as computer system 900 in FIG. 9. Alternatively, a first TLB may include a virtual-address-to-physical-address conversion for DRAM in the computer system and a second TLB may include a virtual-address-to-physical-address conversion for Flash memory in the computer system.
[094] In some embodiments, system 1000 includes fewer or additional components. Moreover, two or more components can be combined into a single component, and/or a position of one or more components may be changed.

[095] In some embodiments, a circuit includes an instruction fetch unit to fetch instructions to be executed which are associated with one or more virtual addresses, a translation lookaside buffer (TLB), and an execution unit to execute the instructions. This TLB converts virtual addresses into physical addresses. Note that the TLB includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell. Moreover, the DRAM and the memory may be included in a memory hierarchy.

[096] In some embodiments, the memory includes Flash memory. Moreover, the circuit may include a processor.

[097] In some embodiments, the circuit is coupled to the DRAM and the memory via a shared communication channel. This shared communication channel may include a memory bus. However, in some embodiments the circuit is coupled to the DRAM via a communication channel and is coupled to the memory via another communication channel.

[098] Note that the memory hierarchy may include different types of memory devices having a range of latency, communication bandwidth, and cost per bit of storage.

[099] In some embodiments, where inclusive caching is used, in which data is stored in both DRAM and the memory, the TLB includes entries associated with DRAM and entries associated with the memory for the data.

[100] In some embodiments, the TLB caches entries for a page table, which can be stored in the DRAM. Moreover, the TLB may facilitate exclusive caching of data in the memory; inclusive caching of data in the memory and the DRAM; and/or hybrid caching of data in the memory and the DRAM, where hybrid caching involves exclusive caching of first data in the memory and inclusive caching of second data in the DRAM and the memory based on a hybrid-caching policy. Additionally, the TLB may facilitate direct communication of data from one type of memory device in the memory hierarchy to the memory.

[101] In some embodiments the circuit is included in an integrated circuit.

[102] Another embodiment provides a first computer system that includes a memory and a processor. This processor may include the instruction fetch unit, the TLB, and the execution unit.
Another embodiment provides a computer-program product that includes a computer-readable storage medium containing information which specifies the design of at least a portion of the integrated circuit (such as the processor) that contains a TLB.

Another embodiment provides a compiler configured to generate instructions for the processor.

Another embodiment provides a second computer system that includes a memory, another processor, and the TLB. This computer system may include an operating system and/or an application program.

Another embodiment provides another circuit that includes the TLB. This circuit may be included in another integrated circuit. For example, the integrated circuit may include a memory controller.

Another embodiment provides a third computer system that includes a memory, the other processor, and the other integrated circuit.

Another embodiment provides a method for performing a conversion, which may be performed by a device (which may be implemented in hardware and/or in software). During operation, the device receives a virtual address. Next, the device converts the virtual address to a physical address using a translation lookaside buffer (TLB), which includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell. Note that the DRAM and the memory are included in a memory hierarchy.

Another embodiment provides a method for performing a conversion, which may be performed by a device (such as the processor). During operation, the device receives an instruction. Next, the device executes the instruction, where executing the instruction includes accessing a translation lookaside buffer (TLB) that facilitates converting virtual addresses into physical addresses. Note that the TLB includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell. Moreover, the DRAM and the memory may be included in a memory hierarchy.

The foregoing descriptions of embodiments have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the present description to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. Additionally, the above disclosure is not
intended to limit the present description. The scope of the present description is defined by the appended claims.
What is claimed is:

1. An integrated circuit, comprising:
   an instruction fetch unit to fetch instructions to be executed, wherein the instructions are associated with one or more virtual addresses;
   a translation lookaside buffer (TLB) to convert the virtual addresses into physical addresses, wherein the TLB includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell; and
   an execution unit to execute the instructions.

2. The integrated circuit of claim 1, wherein the memory includes Flash memory.

3. The integrated circuit of claim 1, wherein the integrated circuit includes a processor.

4. The integrated circuit of claim 1, wherein the integrated circuit is to be coupled to the DRAM and the memory via a shared communication channel.

5. The integrated circuit of claim 4, wherein the shared communication channel includes a memory bus.

6. The integrated circuit of claim 1, wherein the integrated circuit is to be coupled to the DRAM via a communication channel and is to be coupled to the memory via another communication channel.

7. The integrated circuit of claim 1, wherein DRAM and the memory exist within a memory hierarchy which includes different types of memory devices having a range of latency, communication bandwidth, and cost per bit of storage.

8. The integrated circuit of claim 1, wherein the TLB caches entries for a page table, which can be stored in the DRAM.

9. The integrated circuit of claim 1, wherein the TLB facilitates exclusive caching of data in the memory.

10. The integrated circuit of claim 1, wherein the TLB facilitates inclusive caching of data in the memory and the DRAM.
11. The integrated circuit of claim 1, wherein the TLB facilitates hybrid caching of data in the memory and the DRAM; and
   wherein hybrid caching involves exclusive caching of first data in the memory and inclusive caching of second data in the DRAM and the memory based on a hybrid-caching policy.

12. The integrated circuit of claim 1, wherein DRAM and the memory exist within a memory hierarchy; and
   wherein the TLB facilitates direct communication of data from one type of memory device in the memory hierarchy to the memory.

13. The integrated circuit of claim 1, wherein, for inclusive caching in which data is stored in both DRAM and the memory, the TLB is to include entries associated with DRAM and entries associated with the memory for the data.

14. A computer system, comprising:
   a memory;
   a processor, wherein the processor includes:
       an instruction fetch unit to fetch instructions to be executed, wherein the instructions are associated with one or more virtual addresses;
       a translation lookaside buffer (TLB) to convert the virtual addresses to physical addresses, wherein the TLB includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell; and
       an execution unit to execute the instructions.

15. The computer system of claim 14, wherein the memory includes Flash memory.

16. The computer system of claim 14, wherein the processor is coupled to the DRAM and the memory via a shared communication channel.

17. The computer system of claim 16, wherein the shared communication channel includes a memory bus.

18. The computer system of claim 14, wherein the processor is coupled to the DRAM via a communication channel and is coupled to the memory via another communication channel.
19. The computer system of claim 14, wherein DRAM and the memory exist within a memory hierarchy which includes different types of memory devices having a range of latency, communication bandwidth, and cost per bit of storage.

20. The computer system of claim 14, wherein the TLB caches entries for a page table, which can be stored in the DRAM.

21. The computer system of claim 14, wherein the TLB facilitates exclusive caching of data in the memory.

22. The computer system of claim 14, wherein the TLB facilitates inclusive caching of data in the memory and the DRAM.

23. The computer system of claim 14, wherein the TLB facilitates hybrid caching of data in the memory and the DRAM; and wherein hybrid caching involves exclusive caching of first data in the memory and inclusive caching of second data in the DRAM and the memory based on a hybrid-caching policy.

24. The computer system of claim 14, wherein DRAM and the memory exist within a memory hierarchy; and wherein the TLB facilitates direct communication of data from one type of memory device in the memory hierarchy to the memory.

25. The computer system of claim 14, wherein, for inclusive caching in which data is stored in both DRAM and the memory, the TLB is to include entries associated with DRAM and entries associated with the memory for the data.

26. A computer-program product comprising a computer-readable storage medium containing information which specifies the design of at least a portion of an integrated circuit, at least the portion of the integrated circuit including:

   a translation lookaside buffer (TLB) to convert virtual addresses to physical addresses including entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell.

27. The computer-program product of claim 26, wherein the memory includes Flash memory.
28. The computer-program product of claim 26, wherein DRAM and the memory exist within a memory hierarchy which includes different types of memory devices having a range of latency, communication bandwidth, and cost per bit of storage.

29. The computer-program product of claim 26, wherein the TLB caches entries from a page table, which can be stored in the DRAM.

30. The computer-program product of claim 26, wherein the TLB facilitates exclusive caching of data in the memory.

31. The computer-program product of claim 26, wherein the TLB facilitates inclusive caching of data in the memory and the DRAM.

32. The computer-program product of claim 26, wherein the TLB facilitates hybrid caching of data in the memory and the DRAM; and wherein hybrid caching involves exclusive caching of first data in the memory and inclusive caching of second data in the DRAM and the memory based on a hybrid-caching policy.

33. The computer-program product of claim 26, wherein DRAM and the memory exist within a memory hierarchy; and wherein the TLB facilitates direct communication of data from one type of memory device in the memory hierarchy to the memory.

34. The computer-program product of claim 26, wherein, for inclusive caching in which data is stored in both DRAM and the memory, the TLB is to include entries associated with DRAM and entries associated with the memory for the data.

35. A computer system, comprising:
   a processor;
   memory; and
   a translation lookaside buffer (TLB) to convert virtual addresses associated with instructions to be executed by the processor to physical addresses including entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a storage cell in the memory having a retention time that decreases as operations are performed on the storage cell.
36. The computer system of claim 35, wherein the computer system includes an operating system.

37. The computer system of claim 35, wherein the computer system includes an application program.

38. The computer system of claim 35, wherein the memory includes Flash memory.

39. The computer system of claim 35, wherein the processor is coupled to the DRAM and the memory via a shared communication channel.

40. The computer system of claim 39, wherein the shared communication channel includes a memory bus.

41. The computer system of claim 35, wherein the processor is coupled to the DRAM via a communication channel and is coupled to the memory via another communication channel.

42. The computer system of claim 35, wherein DRAM and the memory exist within a memory hierarchy which includes different types of memory devices having a range of latency, communication bandwidth, and cost per bit of storage.

43. The computer system of claim 35, wherein the TLB facilitates exclusive caching of data in the memory.

44. The computer system of claim 35, wherein the TLB facilitates inclusive caching of data in the memory and the DRAM.

45. The computer system of claim 35, wherein the TLB facilitates hybrid caching of data in the memory and the DRAM; and

   wherein hybrid caching involves exclusive caching of first data in the memory and inclusive caching of second data in the DRAM and the memory based on a hybrid-caching policy.

46. The computer system of claim 35, wherein DRAM and the memory exist within a memory hierarchy; and

   wherein the TLB facilitates direct communication of data from one type of memory device in the memory hierarchy to the memory.
47. The computer system of claim 35, wherein, for inclusive caching in which data is stored in both DRAM and the memory, the TLB is to include entries associated with DRAM and entries associated with the memory for the data.

48. An integrated circuit, comprising a translation lookaside buffer (TLB) to convert virtual addresses to physical addresses, wherein the TLB includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell.

49. The integrated circuit of claim 48, wherein the memory includes Flash memory.

50. The integrated circuit of claim 48, wherein the integrated circuit includes a memory controller.

51. The integrated circuit of claim 48, wherein the integrated circuit is to be coupled to the DRAM and the memory via a shared communication channel.

52. The integrated circuit of claim 51, wherein the shared communication channel includes a memory bus.

53. The integrated circuit of claim 48, wherein the integrated circuit is to be coupled to the DRAM via a communication channel and is to be coupled to the memory via another communication channel.

54. The integrated circuit of claim 48, wherein DRAM and the memory exist within a memory hierarchy which includes different types of memory devices having a range of latency, communication bandwidth, and cost per bit of storage.

55. The integrated circuit of claim 48, wherein the TLB caches entries for a page table, which can be stored in the DRAM.

56. The integrated circuit of claim 48, wherein the TLB facilitates exclusive caching of data in the memory.

57. The integrated circuit of claim 48, wherein the TLB facilitates inclusive caching of data in the memory and the DRAM.

58. The integrated circuit of claim 48, wherein the TLB facilitates hybrid caching of data in the memory and the DRAM; and
wherein hybrid caching involves exclusive caching of first data in the memory and inclusive caching of second data in the DRAM and the memory based on a hybrid-caching policy.

59. The integrated circuit of claim 48, wherein DRAM and the memory exist within a memory hierarchy; and

wherein the TLB facilitates direct communication of data from one type of memory device in the memory hierarchy to the memory.

60. The integrated circuit of claim 48, wherein, for inclusive caching in which data is stored in both DRAM and the memory, the TLB is to include entries associated with DRAM and entries associated with the memory for the data.

61. A computer system, comprising:

a processor;

a memory; and

integrated circuit including a translation lookaside buffer (TLB) to convert virtual addresses associated with instructions to be executed by the processor to physical addresses including entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a storage cell in the memory having a retention time that decreases as operations are performed on the storage cell.

62. The computer system of claim 61, wherein the memory includes Flash memory.

63. The computer system of claim 61, wherein the processor is coupled to the DRAM and the memory via a shared communication channel.

64. The computer system of claim 63, wherein the shared communication channel includes a memory bus.

65. The computer system of claim 61, wherein the processor is coupled to the DRAM via a communication channel and is coupled to the memory via another communication channel.

66. The computer system of claim 61, wherein the integrated circuit includes a memory controller.

67. The computer system of claim 61, wherein DRAM and the memory exist within a memory hierarchy which includes different types of memory devices having a range of latency, communication bandwidth, and cost per bit of storage.
68. The computer system of claim 61, wherein the TLB includes entries for a page table, which can be stored in the DRAM.

69. The computer system of claim 61, wherein the TLB facilitates exclusive caching of data in the memory.

70. The computer system of claim 61, wherein the TLB facilitates inclusive caching of data in the memory and the DRAM.

71. The computer system of claim 61, wherein the TLB facilitates hybrid caching of data in the memory and the DRAM; and

wherein hybrid caching involves exclusive caching of first data in the memory and inclusive caching of second data in the DRAM and the memory based on a hybrid-caching policy.

72. The computer system of claim 61, wherein DRAM and the memory exist within a memory hierarchy; and

wherein the TLB facilitates direct communication of data from one type of memory device in the memory hierarchy to the memory.

73. The computer system of claim 61, wherein, for inclusive caching in which data is stored in both DRAM and the memory, the TLB is to include entries associated with DRAM and entries associated with the memory for the data.

74. A compiler configured to generate instructions for a processor with a translation lookaside buffer (TLB) that supports two types of entries, wherein the TLB is configured to:

convert virtual addresses to physical addresses including entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell.

75. The compiler of claim 74, wherein the memory includes Flash memory.

76. The compiler of claim 74, wherein DRAM and the memory exist within a memory hierarchy which includes different types of memory devices having a range of latency, communication bandwidth, and cost per bit of storage.

77. The compiler of claim 74, wherein the TLB includes entries for a page table, which can be stored in the DRAM.
78. The compiler of claim 74, wherein the TLB facilitates exclusive caching of data in the memory.

79. The compiler of claim 74, wherein the TLB facilitates inclusive caching of data in the memory and the DRAM.

80. The compiler of claim 74, wherein the TLB facilitates hybrid caching of data in the memory and the DRAM; and

wherein hybrid caching involves exclusive caching of first data in the memory and inclusive caching of second data in the DRAM and the memory based on a hybrid-caching policy.

81. The compiler of claim 74, wherein DRAM and the memory exist within a memory hierarchy; and

wherein the TLB facilitates direct communication of data from one type of memory device in the memory hierarchy to the memory.

82. A method for performing a conversion, comprising:

receiving a virtual address; and

converting the virtual address to a physical address using a translation lookaside buffer (TLB), which includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell.

83. A method, comprising:

receiving an instruction, wherein the instructions are associated with one or more virtual addresses; and

executing the instruction, wherein executing the instruction includes accessing a translation lookaside buffer (TLB) that facilitates converting virtual addresses into physical addresses;

wherein the TLB includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a memory having a storage cell with a retention time that decreases as operations are performed on the storage cell.

84. A computer system, comprising:
a memory means for computing, wherein the means includes:

an instruction fetch unit to fetch instructions to be executed, wherein the instructions are associated with one or more virtual addresses;

a translation lookaside buffer (TLB) to convert virtual addresses associated with instructions to be executed by the processor to physical addresses, wherein the TLB includes entries for physical addresses that are dedicated to dynamic random access memory (DRAM) and entries for physical addresses that are dedicated to a storage cell in the memory having a retention time that decreases as operations are performed on the storage cell; and

an execution unit to execute the instructions.
FIG. 1
FIG. 2

CORES OR MEMORY DENSITY

PROCESSOR-CORE TREND

MEMORY-DENSITY TREND

TIME

200
FIG. 3
RECEIVE A VIRTUAL ADDRESS

CONVERT THE VIRTUAL ADDRESS TO A PHYSICAL ADDRESS USING A TRANSLATION LOOKASIDE BUFFER (TLB), WHICH INCLUDES ENTRIES FOR PHYSICAL ADDRESSES THAT ARE DEDICATED TO DYNAMIC RANDOM ACCESS MEMORY (DRAM) AND ENTRIES FOR PHYSICAL ADDRESSES THAT ARE DEDICATED TO A MEMORY HAVING A STORAGE CELL WITH A RETENTION TIME THAT DECREASES AS OPERATIONS ARE PERFORMED ON THE STORAGE CELL.

FIG. 4A
RECEIVE AN INSTRUCTION WHICH IS ASSOCIATED WITH ONE OR MORE VIRTUAL ADDRESSES

EXECUTE THE INSTRUCTION BY ACCESSING A TRANSLATION LOOKASIDE BUFFER (TLB) THAT FACILITATES CONVERTING VIRTUAL ADDRESSES INTO PHYSICAL ADDRESSES, WHERE THE TLB INCLUDES ENTRIES FOR PHYSICAL ADDRESSES THAT ARE DEDICATED TO DYNAMIC RANDOM ACCESS MEMORY (DRAM) AND ENTRIES FOR PHYSICAL ADDRESSES THAT ARE DEDICATED TO A MEMORY HAVING A STORAGE CELL WITH A RETENTION TIME THAT DECREASES AS OPERATIONS ARE PERFORMED ON THE STORAGE CELL.

FIG. 4B
FIG. 5C
FIG. 8

STORE DATA IN DRAM 814

RECEIVE DATA 810

STORE DATA IN FLASH MEMORY 812

800
FIG. 9
FIG. 10
A. CLASSIFICATION OF SUBJECT MATTER

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>US 5 802 554 A (CACERES RAMON [US] ET AL) 1 September 1998 (1998-09-01) column 2, line 5 - line 12 column 3, line 7 - line 19 column 3, line 52 - line 62 figures 1-4</td>
<td>1-84</td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of Box C.

See patent family annex.

Date of the actual completion of the international search
21 November 2008

Date of mailing of the international search report
04/12/2008

Name and mailing address of the ISA/
European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel: (+31-70) 340-20 40
Fax: (+31-70) 340-3016

Authorized officer
Knutsson, Frederic
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
</table>