US 20070186028A2

(19) **United States**

(12) **Patent Application Publication**

Kissell

(10) **Pub. No.: US 2007/0186028 A2**

(43) **Pub. Date: Aug. 9, 2007**

REPUBLICATION

(54) **SYNCHRONIZED STORAGE PROVIDING MULTIPLE SYNCHRONIZATION SEMANTICS**

(75) Inventor: **Kevin Kissell**, Le Bar sur Loup (FR)

Correspondence Address:
HUFFMAN LAW GROUP, P.C.
1900 MESA AVE.
COLORADO SPRINGS, CO 80906 (US)

(73) Assignee: **MIPS Technologies, Inc.**, Mountain View, CA (US)

(21) Appl. No.: **10/954,988**

(22) Filed: **Sep. 30, 2004**

Prior Publication Data

(65) US 2005/0251613 A1 Nov. 10, 2005

Related U.S. Application Data

(63) Continuation-in-part of application No. 10/929,342, filed on Aug. 27, 2004, which is a continuation-in-part of application No. 10/929,102, filed on Aug. 27, 2004, which is a continuation-in-part of application No. 10/928,746, filed on Aug. 27, 2004, which is a continuation-in-part of application No. 10/929,097, filed on Aug. 27, 2007, which is a continuation-in-part of application No. 10/684,350, filed on Oct. 10, 2003, which is a continuation-in-part of application No. 10/684,348, filed on Oct. 10, 2003, which is a continuation-in-part of application No. 10/684,350, filed on Oct. 10, 2003, which is a continuation-in-part of application No. 10/684,348, filed on Oct. 10, 2003, which is a continuation-in-part of application No. 10/684,350, filed on Oct. 10, 2003, which is a continuation-in-part of application No. 10/684,348, filed on Oct. 10, 2003, which is a continuation-in-part of application No. 10/684,350, filed on Oct. 10, 2003, which is a continuation-in-part of application No. 10/684,348, filed on Oct. 10, 2003, which is a continuation-in-part of application No. 10/684,350, filed on Oct. 10, 2003, which is a continuation-in-part of application No. 10/684,348, filed on Oct. 10, 2003.

(60) Provisional application No. 60/499,180, filed on Aug. 28, 2003, now expired. Provisional application No. 60/502,358, filed on Sep. 12, 2003, now expired. Provisional application No. 60/502,359, filed on Sep. 12, 2003, now expired. Provisional application No. 60/499,180, filed on Aug. 28, 2003, now expired. Provisional application No. 60/502,358, filed on Sep. 12, 2003, now expired. Provisional application No. 60/502,359, filed on Sep. 12, 2003, now expired. Provisional application No. 60/499,180, filed on Aug. 28, 2003, now expired. Provisional application No. 60/502,358, filed on Sep. 12, 2003, now expired. Provisional application No. 60/502,359, filed on Sep. 12, 2003, now expired. Provisional application No. 60/499,180, filed on Aug. 28, 2003, now expired. Provisional application No. 60/502,358, filed on Sep. 12, 2003, now expired. Provisional application No. 60/502,359, filed on Sep. 12, 2003, now expired. Provisional application No. 60/499,180, filed on Aug. 28, 2003, now expired. Provisional application No. 60/502,358, filed on Sep. 12, 2003, now expired. Provisional application No. 60/502,359, filed on Sep. 12, 2003, now expired. Provisional application No. 60/499,180, filed on Aug. 28, 2003, now expired. Provisional application No. 60/502,358, filed on Sep. 12, 2003, now expired. Provisional application No. 60/502,359, filed on Sep. 12, 2003, now expired.

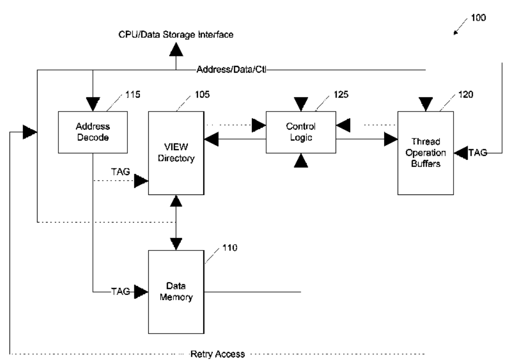
Publication Classification

(51) **Int. Cl.**
G11C 5/00 (2006.01)

(52) **U.S. Cl.** **711/1**

(57) **ABSTRACT**

A shared resource access control system having a gating storage responsive to a plurality of controls with each of the controls derived from an instruction context identifying the shared resource, the gating storage including a plurality of sets of access method functions with each set of access method functions including a first access method function and a second access method function with the gating storage producing a particular one access method function from a particular one set responsive to the controls; and a controller, coupled to the gating storage, for controlling access to the shared resource using the particular one access method function.



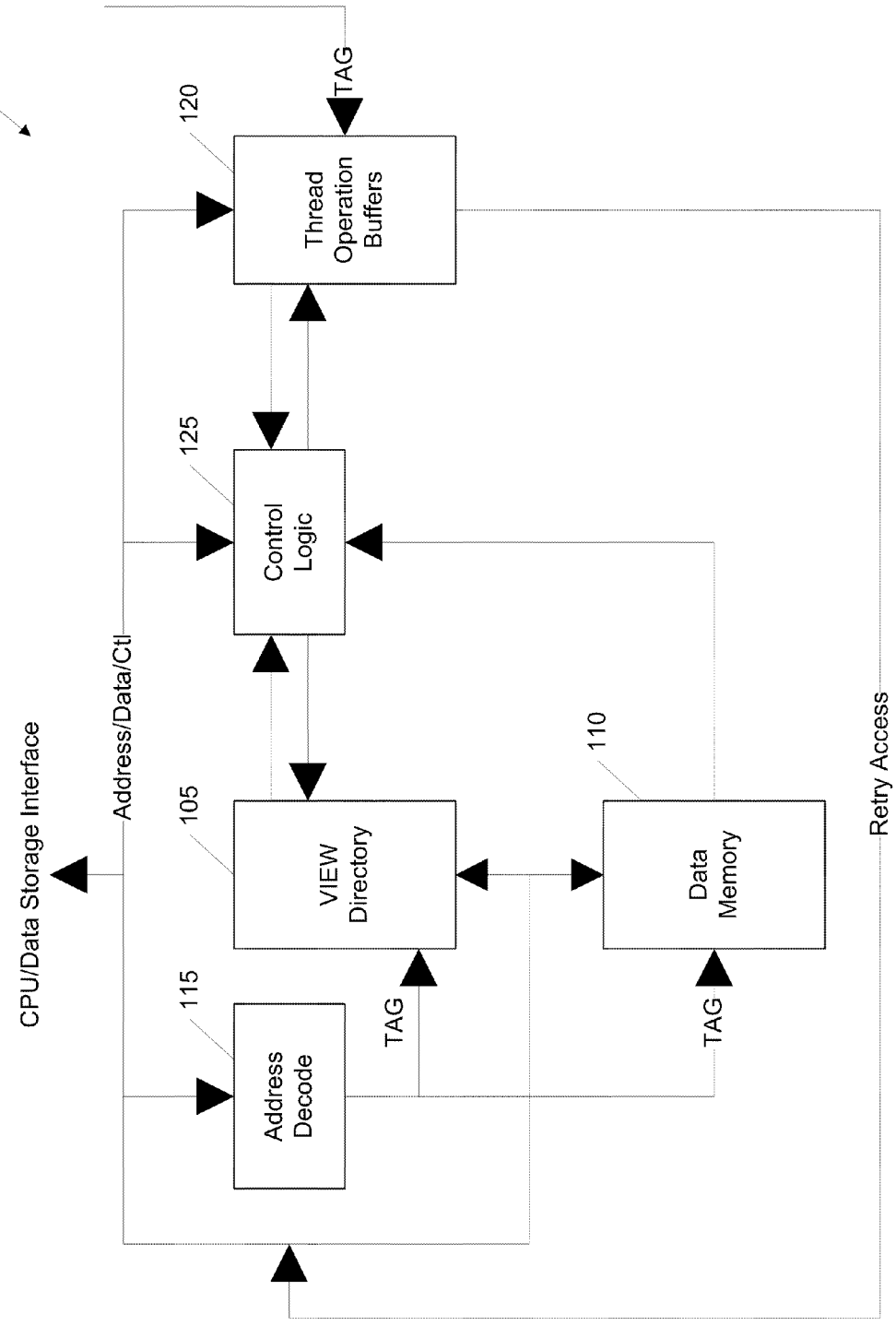

100
TAG
120
Thread Operation Buffers
125
Control Logic
Address/Data/Ctl
CPU/Data Storage Interface
105
VIEW Directory
110
Data Memory
115
Address Decode
TAG
TAG
Retry Access

FIGURE_1

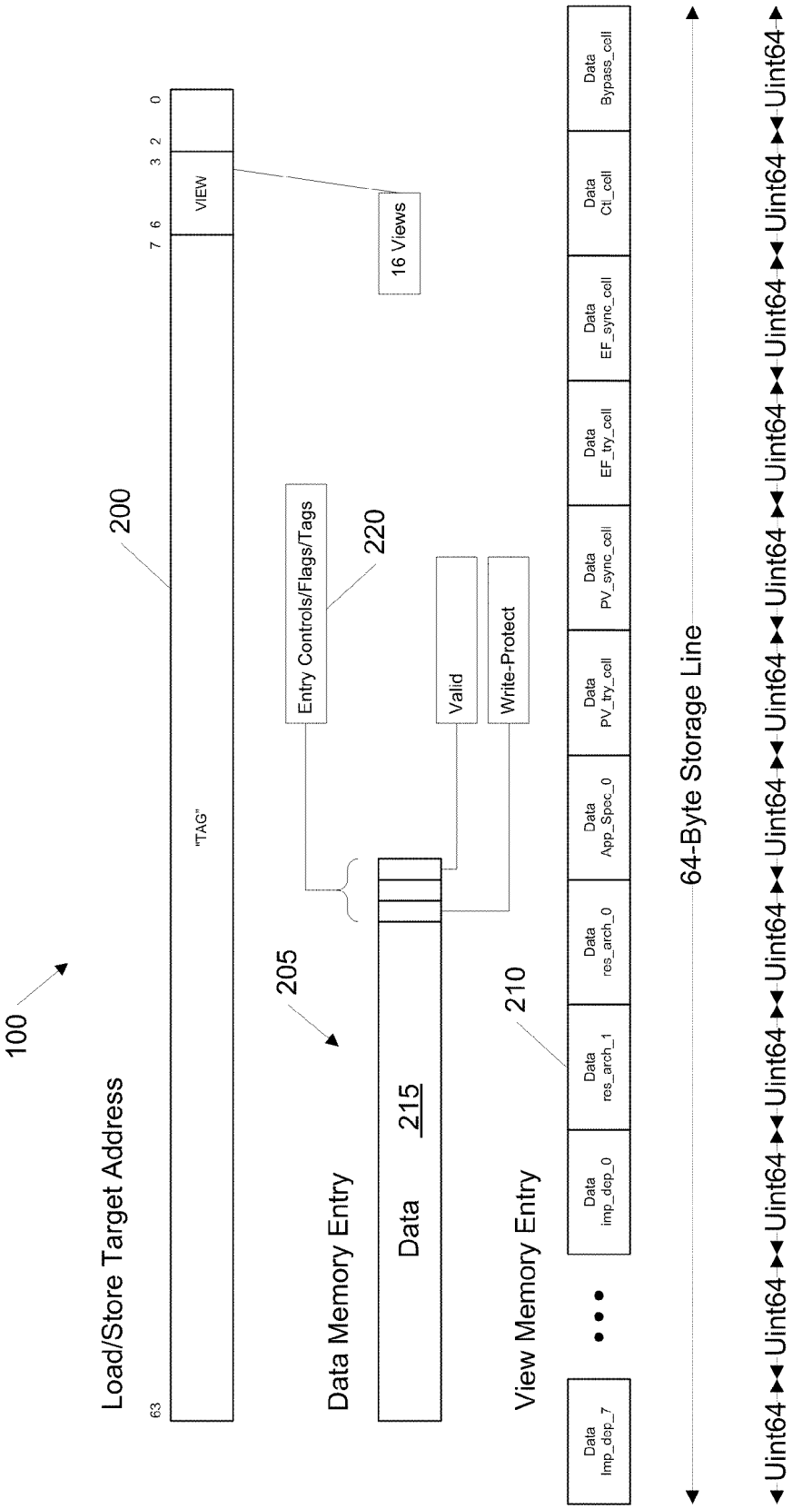

100
Load/Store Target Address
63
"TAG"
7 6
VIEW
3 2
0
200
16 Views
Entry Controls/Flags/Tags
220
205
Data Memory Entry
Data 215
Valid
Write-Protect
210
View Memory Entry
Data imp_dep_7
Data imp_dep_0
Data res_arch_1
Data res_arch_0
Data App_Spec_0
Data PV_try_cell
Data PV_sync_cell
Data EF_try_cell
Data EF_sync_cell
Data Ctl_cell
Data Bypass_cell
64-Byte Storage Line
Uint64
Uint64
Uint64
Uint64
Uint64
Uint64
Uint64
Uint64

FIGURE_2

SYNCHRONIZED STORAGE PROVIDING MULTIPLE SYNCHRONIZATION SEMANTICS

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation-in-part (CIP) of the following co-pending Non-Provisional U.S. patent applications, which are hereby expressly incorporated by reference in their entireties for all purposes:

Ser. No. (Docket No.)	Filing Date	Title
10/929,342 MIPS.0189.01US	27 Aug. 2004	INTEGRATED MECHANISM FOR SUSPENSION AND DEALLOCATION OF COMPUTATIONAL THREADS OF EXECUTION IN A PROCESSOR
10/929,102 MIPS.0193.00US	27 Aug. 2004	MECHANISMS FOR DYNAMIC CONFIGURATION OF VIRTUAL PROCESSOR RESOURCES
10/928,746 MIPS.0192.00US	27 Aug. 2004	APPARATUS, METHOD, AND INSTRUCTION FOR INITIATION OF CONCURRENT INSTRUCTION STREAMS IN A MULTITHREADING MICROPROCESSOR
10/929,097 MIPS.0194.00US	27 Aug. 2004	MECHANISMS FOR SOFTWARE MANAGEMENT OF MULTIPLE COMPUTATIONAL CONTEXTS

[0002] This application is a continuation-in-part (CIP) of the following co-pending Non-Provisional U.S. patent applications, which are hereby expressly incorporated by reference in their entireties for all purposes:

Ser. No. (Docket No.)	Filing Date	Title
10/684,350 MIPS.0188.01US	10 Oct. 2003	MECHANISMS FOR ASSURING QUALITY OF SERVICE FOR PROGRAMS EXECUTING ONA MULTITHREADED PROCESSOR
10/684,348 MIPS.0189.00US	10 Oct. 2003	INTEGRATED MECHANISM FOR SUSPENSION AND DEALLOCATION OF COMPUTATIONAL THREADS OF EXECUTION INA PROCESSOR

[0003] Each of the applications identified in Paragraph [001] is a continuation-in-part (CIP) of each of the following co-pending Non-Provisional U.S. patent applications, which are hereby expressly incorporated by reference in their entireties for all purposes:

Ser. No. (Docket No.)	Filing Date	Title
10/684,350 MIPS.0188.01US	10 Oct. 2003	MECHANISMS FOR ASSURING QUALITY OF SERVICE FOR PROGRAMS EXECUTING ONA MULTITHREADED PROCESSOR
10/684,348 MIPS.0189.00US	10 Oct. 2003	INTEGRATED MECHANISM FOR SUSPENSION AND DEALLOCATION OF COMPUTATIONAL THREADS OF EXECUTION INA PROCESSOR

[0004] Each of the co-pending Non-Provisional U.S. patent applications identified in Paragraph [001] and Paragraph [002] above claim the benefit of the following U.S. Provisional Applications, which are hereby expressly incorporated by reference in their entireties for all purposes:

Ser. No. (Docket No.)	Filing Date	Title
60/499,180 MIPS.0188.00US	28 Aug. 2003	MULTITHREADING APPLICATION SPECIFIC EXTENSION
60/502,358 MIPS.0188.02US	12 Sep. 2003	MULTITHREADING APPLICATION SPECIFIC EXTENSION TO A PROCESSOR ARCHITECTURE
60/502,359 MIPS.0188.03US	12 Sep. 2003	MULTITHREADING APPLICATION SPECIFIC EXTENSION TO A PROCESSOR ARCHITECTURE

[0005] This application is related to the following Non-Provisional U.S. patent applications:

Ser. No. (Docket No.) (Client Ref.)	Filing Date	Title
10/955,231 MIPS.0196.00US	30 Sep. 2004	A SMART MEMORY BASED SYNCHRONIZATION CONTROLLER FOR A MULTI-THREADED MULTIPROCESSOR SOC

[0006] All of the above-referenced related patent applications and priority patent applications are hereby expressly incorporated by reference in their entireties for all purposes.

FIELD OF THE INVENTION

[0007] The invention relates generally to multiprocessing systems and more specifically to multiple thread synchronization activities on one or more processing elements (real, virtual, or otherwise).

BACKGROUND OF THE INVENTION

[0008] Multiprocessing systems continue to become increasingly important in computing systems for many applications, including general purpose processing systems and embedded control systems. In the design of such multiprocessing systems, an important architectural consideration is scalability. In other words, as more hardware resources are added to a particular implementation the machine should produce higher performance. Not only do embedded implementations require increased processing power, many also require the seemingly contradictory attribute of providing low power consumption. In the context of these requirements, particularly for the embedded market, solutions are implemented as "Systems on Chip" or "SoC." The assignee of the present application, MIPS Technologies, Inc., offers a broad range of solutions for such SoC multiprocessing systems.

[0009] In multiprocessing systems, loss in scaling efficiency may be attributed to many different issues, including long memory latencies and waits due to synchronization. The present invention addresses improvements to synchronization among threads in a multithreaded multiprocessing environment, particularly when individual threads may be active on one or more multiple processors, on a single processor but distributed among multiple thread contexts, or resident in memory (virtualized threads).

[0010] Synchronization in a multithreaded system refers to the activities and functions of such a multiplicity of threads that coordinate use of shared system resources (e.g., system memory and interface FIFOs) through variables storing "state" bits for producer/consumer communication and mutual exclusion (MUTEX) tasks. Important considerations for implementing any particular synchronization paradigm include designing and implementing structures and processes that provide for deadlock-free operation while being very efficient in terms of time, system resources, and other performance measurements.

[0011] Details regarding the MIPS processor architecture are provided in the following document, which is incorporated by reference in its entirety for all purposes: D. Sweetman, See MIPS Run, Morgan Kaufmann Publishers, Inc. (1999).

[0012] The difficulty of finding a hardware synchronization solution for a RISC processor is compounded by the nature of the RISC paradigm. A CISC paradigm is easier, in some ways, to adapt hardware resources to particular problems because the instruction set may be extended virtually without limit as instructions and operands in an instruction pipeline may be of variable length. A designer that wants to implement a special hardware synchronization instruction set is able to add new synchronization instructions easily as many CISC instruction sets already contemplate extensions to basic instruction sets. However, that solution is generally not available to designers working with RISC instruction sets. Most instructions sets are filled or nearly filled with vacancies judiciously filled after many factors are extensively considered and evaluated. What is needed is a system

for extending or enhancing existing instruction sets, with such a solution particularly useful in the RISC environment, but not exclusively useful as the CISC environment may also benefit from instruction set extension.

SUMMARY OF THE INVENTION

[0013] The present invention has been made in consideration of the above situation, and has as an object to provide a system, method, computer program product, and propagated signal which efficiently, in a specific embodiment, enables inter-thread synchronization among a plurality of threads that may be active on one or more of: multiple processors, on a single processor but distributed among multiple thread contexts, and/or resident in memory (virtualized threads) without deadlock. In a more generalized description of the preferred embodiment, a system, method, computer program, and propagated signal which efficiently enables extension of instructions and classes of instructions.

[0014] A preferred embodiment of the present invention includes a shared resource access control system having a gating storage responsive to a plurality of control bits with the control bits derived from an access reference identifying the shared resource, the gating storage including a plurality of sets of views with each set of views including a first view and a second view with the gating storage producing a particular view from a particular one set responsive to the control bits; and a controller, coupled to the gating storage, for controlling access to the shared resource using the particular one view.

[0015] Another preferred embodiment of the present invention includes a shared resource access control method, the method applying an access instruction for the data storage location to a memory system, the memory system including a plurality of data storage locations, each the data storage location associated with a set of views including a first view and a second view with the memory system producing a particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location; producing the particular one view from the particular one set of views; and controlling access to the data storage location using the particular one view.

[0016] Preferred embodiments of the present also include both a computer program product having a computer readable medium carrying program instructions for accessing a memory when executed using a computing system, the executed program instructions executing a method, as well as a propagated signal on which is carried computer-executable instructions which when executed by a computing system performs a method, the method including applying an access instruction for the data storage location to a memory system, the memory system including a plurality of data storage locations, each the data storage location associated with a set of views including a first view and a second view with the memory system producing a particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location; producing the particular one view from the particular one set of views; and controlling access to the data storage location using the particular one view.

[0017] An alternate preferred embodiment includes an apparatus for extending a load/store instruction having a target address, the apparatus including a memory system having a view associated with a data storage location identified by a tag derived from the target address, the data storage location associated with the load/store instruction, the memory system responsive to the target address to produce a particular view for the load/store instruction from the memory system; and a controller, coupled to the data storage location and to the memory system, for implementing a an load/store method for the load/store instruction using the particular view.

[0018] Other preferred embodiments include a method, and both a computer program product and a propagated signal carrying computer-executable instructions for extending an instruction using an instruction rule when executed by a computing system, the computer-executable instructions implementing a method. This method including producing, responsive to the target address, a particular view for the load/store instruction from a memory system, the memory system having a view associated with a data storage location identified by a tag derived from the target address, the data storage location associated with the load/store instruction; and implementing a load/store method for the load/store instruction using the particular view.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] FIG. **1** is a schematic block diagram of a preferred embodiment for a gating storage system; and

[0020] FIG. **2** is a schematic diagram illustrating a preferred implementation for an example of a bit assignment of the data elements shown in FIG. **4**.

DESCRIPTION OF THE PREFERRED EMBODIMENT

[0021] The present invention relates to multiple thread synchronization activities on one or more processing elements. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment and the generic principles and features described herein will be readily apparent to those skilled in the art. Thus, the present invention is not intended to be limited to the embodiment shown but is to be accorded the widest scope consistent with the principles and features described herein.

[0022] Data-driven programming models map well to multithreaded architectures. For example, threads of execution are able to read data from memory-mapped I/O FIFOs, and may be suspended for as long as it takes for a FIFO to fill, while other threads continue to execute. When the data is available, the load completes, and the incoming data may be processed directly in the load destination register without requiring any I/O interrupt service, polling, or software task scheduling.

[0023] However, many architecture models have no provision for restartably interrupting a memory operation once a memory management unit has processed it. It would thus be impossible for a thread context of a blocked thread to be used by an exception handler, or for an operating system to swap out and re-assign such a thread context. The preferred embodiment of the present invention therefore introduces a

specific implementation of a concept of "Gating Storage"—memory (or memory-like devices) that are tagged in the TLB (with extended bits or direct physical decode) as potentially requiring abort and restart of loads or stores. The abort/restart capability may require explicit support from the processor/memory interface protocols. It should be noted that in some implementations the memory is not tagged in a memory management unit; rather the memory may be direct mapped or otherwise identified.

[0024] The gating storage, as described herein, is a special case of a more generalized concept. As described herein, the gating storage may be conceptualized as a physical address subspace with special properties. In the specific examples described above and as set forth in more detail below, this gating storage serves as an Inter-Thread Communication (ITC) storage for enabling thread-to-thread and thread-to-I/O synchronization, particularly for load/store instructions. Each 64-bit location or "cell" within this gating storage space appears at multiple consecutive addresses, or "views", distinguished by "view" bits (e.g., bits [6:3] though other implementations may use more, fewer, or different bits) of the load/store target address. Each view may have distinct semantics for the same instruction. A fundamental property of the gating storage is that loads and stores may be precisely view-referenced by the load or store. Any blocked loads and stores resume execution when the actions of other threads of execution, or possibly those of external devices, result in the completion requirements being satisfied. As gating storage references, blocked ITC loads and stores can be precisely aborted and restarted by systems software.

[0025] This structure has several motivations:

[0026] 1. Issue bandwidth is a critical resource on multithreaded processors. Whereas spinning on a lock in a true multiprocessor system wastes only the issue bandwidth of the processor waiting on the resource, in a multithreaded processor, the act of polling the lock on the resource consumes issue bandwidth needed by the program thread holding the lock, and further delays the release of the resource. A thread blocked waiting on a value in gating storage consumes no issue bandwidth until the value is produced or consumed.

[0027] 2. Using hardware synchronization reduces the overhead of inter-thread control and data exchanges and makes finer grained parallel computations economical. A well-behaved algorithm running on an optimal implementation may pass values between threads at a cost of a single pipelined load or store cycle for each thread.

[0028] 3. It allows a "push model" of multiprocessor/multithread data flow to be implemented in a near-optimal way.

[0029] For example, in some views, cells within gating storage space may be "Empty" or "Full". A load from a cell that is Empty causes the thread issuing the load to be suspended until the cell is written to by a store from another thread. A store to a cell, when Full, causes the thread issuing the store to be suspended until a previous value has been consumed by a load.

[0030] Such gating storage may define independent Empty and Full conditions, rather than a single Empty/Full bit, in order to allow for FIFO buffered gating storage. In a classical Empty/Full memory configuration, Empty would simply be the negation of Full. A FIFO cannot be both Empty and Full, but it is able to be neither Empty nor Full when it contains some data, but could accept more.

[0031] It is possible that one view in an implementation is a standard empty/full synchronization construct for producers and consumers. Another view may implement classical "P/V" semaphores by blocking loads even of "full" cells when the value of the cell is zero. Other views might implement atomic semaphore "get" and "put", or fetch-and-increment or fetch-and-decrement operations without blocking, among other types, variations, and implementations of synchronization constructs.

[0032] As discussed above, a load/store target address may designate gating storage through a direct decode or use of special TLB entries. References to virtual memory pages whose TLB entries are tagged as gating storage resolve not to standard memory, but to a store with special attributes. Each page maps a set of 1-64 64-bit storage locations, called "cells", each of which may be accessed in one of a multiplicity of ways, called "views" using standard load and store instructions. The view is encoded in the low order (and untranslated) view bits of a generated virtual address for the load/store target address. As included in the preferred embodiment of the present invention, a fundamental property of the gating storage is that it synchronizes executions streams. Loads and stores to/from a memory location in gating storage, as implemented, block until the state of the cell corresponds to the required conditions for completion in the selected view. A blocked load or store may be precisely aborted when necessary, and restarted by the controlling operating system when appropriate.

[0033] Each cell of the gating storage has Empty and Full Boolean states associated with it. The cell views are then defined as follows.

TABLE I

CELL VIEW DEFINITIONS

Address Bits [6:3] View BIT Value	Gating Storage Behavior
2#0000	Bypass: Loads/Stores do not block, and do not affect Empty/Full
2#0001	Control: Read or Write of Status/Control Information. The bit layout described below is guaranteed valid only when the Control view is referenced via LW/SW or LD/SD instructions Data Bits — Meaning 0 — When set, cell is Empty and will block on an attempt to load as synchronized storage

TABLE I-continued

CELL VIEW DEFINITIONS

Address Bits [6:3] View BIT Value	Gating Storage Behavior
	1 When set, cell is Full and will block on an attempt to store as synchronized storage
	15:2 Reserved for future architectural definition
	63:16 Implementation Dependent State
2#0010	Empty/Full Synchronized view. Loads cause the issuing thread to block when cell is Empty, and set the Empty state on returning the last available load value. Stores block when the cell is Full, and set the Full state on accepting the last possible store value. Minimally, a cell contains a single value, such that the Empty bit is the complement of the Full bit.
2#0011	Empty/Full Storage "Try" view. Loads return a value of zero when cell is Empty, regardless of actual data contained. Otherwise Load behavior is same as in Empty/Full Synchronized view. Normal Stores to Full locations through the E/F Try view fail silently to update the contents of the cell, rather than block the thread. SC (Store Conditional) instructions referencing the E/F Try view indicate success or failure based on whether the ITC store succeeds or fails.
2#0100	P/V Synchronized view. Loads and stores do not modify the Empty and Full bits, both of which should be cleared as part of cell initialization for P/V semaphore use. Loads return the current cell data value when the value is non-zero, and cause an atomic post-decrement of the value. When the cell value is zero, loads block until the cell takes a non-zero value. Stores cause an atomic increment of the cell value, up to a maximal value at which they saturate. The width of the incremented/decremented field within the ITC cell need not be the full 32 or 64-bit width of the cell. It preferably, however, implements at least 15 bits of unsigned value
2#0101	P/V Storage "Try" view. Loads and stores do not modify the Empty and Full bits, both of which should be cleared as part of cell initialization for P/V semaphore use. Loads return the current cell data value, even when zero. When the load value is non-zero, an atomic post-decrement is performed of the value. Stores cause a saturating atomic increment of the cell value, as described for the P/V Synchronized view, and cannot fail
2#0110	Architecturally Reserved View 0
2#0111	Architecturally Reserved View 1
2#1000	Application or Implementation Specific View 0
. . .	Application or Implementation Specific View X
2#1111	Application or Implementation Specific View 7

[0034] Each storage cell could thus be described by the C structure:

```
struct {
    uint64 bypass_cell
    uint64 ctl_cell
    uint64 ef_sync_cell;
    uint64 ef_try_cell;
    uint64 pv_synch_cell;
    uint64 pv_try_cell;
    uint64 res_arch[2]
    uint64 imp_dep[8]
} ITC_cell;
```

[0035] where all sixteen of the elements reference the same sixty-four bits of underlying storage data. References to this storage may have access types of less than sixty-four bits (e.g. SW/LW, SH/LH SB/LB), with the same Empty/Full protocol being enforced on a per-access basis. Store/Load pairs of the same data type to a given ITC address will always reference the same data, but the byte and halfword ordering within words, and the word ordering within 64-bit doublewords, may be implementation and endianness-dependent, i.e. a SW followed by a LB from the same ITC address is not guaranteed to be portable. While the design of ITC storage allows references to be expressed in terms of C language constructs, compiler optimizations may generate sequences that break ITC protocols, and great care must be taken if ITC is directly referenced as "memory" in a high-level language.

[0036] Systems that do not support 64-bit loads and stores need not implement all 64 bits of each cell as storage. When only 32 bits of storage are instantiated per cell, it must be visible in the least significant 32-bit word of each view, regardless of the endianness of the processor, while the results of referencing the most significant 32-bits of each view are implementation-dependent. Ignoring the 22 bit of the address on each access can satisfy this requirement. In this way a C language cast from a unit **64** to a unit **32** reference will acquire the data on both big-endian and little-endian CPU configurations. When more than 32 bits of Control view information are required in a 32-bit ITC store, the additional control bits should be referenced using one of the implementation-dependent views. Empty and Full bits are distinct so that decoupled multi-entry data buffers, such as FIFOs can be mapped into ITC storage.

[0037] The gating storage may be saved and restored by copying the {bypass_cell, ctl_cell} pair to and from general storage. While the full data width, 64 or 32 bits, of bypass-_cell must be preserved, strictly speaking, only the least significant bits of the ef_state need to be manipulated. In the case of multi-entry data buffers (e.g. FIFOs), each cell must be read using an Empty/Full view until the Control view shows the cell to be Empty to drain the buffer on a copy. The FIFO state can then be restored by performing a series of Empty/Full stores to an equivalent FIFO cell starting in an Empty state. Implementations may provide depth counters in the implementation-specific bits of the Control view to optimize this process. Software must ensure that no other accesses are made to ITC cells during the save and restore processes.

[0038] The "physical address space" of gating storage may be made global across all VPEs and processors in a multi-processor system as shown and described above, such that a thread is able to synchronize on a cell on a different VPE from the one on which it is executing. Global gating storage addresses could be derived from a CPUNum field of an EBase register of each VPE. CPUNum includes ten bits that correspond to the ten significant bits of storage address into the gating storage. Processors or cores designed for uniprocessor applications need not export a physical interface to the gating storage, and may treat the gating storage as a processor-internal resource.

[0039] FIG. **1** is a schematic block diagram of a preferred embodiment for a gating storage system **100** including a view directory **105**. View directory **105** contains the view as discussed above for load/store instructions implementing synchronization methods through use of system **100**. A data memory **110** includes an associated data block for any gating storage data. An address decoder **115** generates the tag, from an instruction or an instruction operand, as an address offset of a block within a page. An instruction that potentially references the gating storage space causes view directory **105** to be cycled at the tag generated from the operand of the instruction. Any gating storage operation is aborted when decoder **115** resolves the virtual address translation to non-gating system **100**. When the instruction is a load/store instruction, the operand is a target address within gating storage system **100**, the operand-identified function is one of the views (e.g., EF Synchronization View), and when the instruction operation is not blocked, i.e. a load from a non-Empty cell or a store to a non-Full cell, a data transfer is performed relative to data memory **110** on a subsequent cycle. Otherwise, the instruction is recorded in a thread operation buffer **120** and the thread is suspended. Changes in gating storage system **100** that may affect suspended threads are retried. For example, each change of an empty/full state of a location in data memory **110** causes the affected address to be broadcast to thread operation buffers **120**, where it is compared against the addresses of blocked operations. Those operations that become unblocked are retried against the new tag state, and when they succeed, the content of the particular thread operation buffer **120** is de-allocated and the associated thread unblocked by completion of the GS instruction operation. A control logic function **125** arbitrates between retries and new requests.

[0040] This style of implementation allows a gating storage system "hit" where data memory **110** is already in the state desired, to have the same timing as a cache hit, but it presupposes tight integration with an instruction generation source, for example with a processor core. Less closely coupled implementations of gating storage system **100**, where the gating storage block is instantiated more like a scratchpad RAM or an I/O device supporting a gating storage protocol, would be less core-intrusive, but may also stall the pipeline even on a "hit".

[0041] FIG. **2** is a schematic diagram illustrating a preferred implementation for a bit assignment of the data elements shown in FIG. **1**. For example, when the applicable instruction is a load/store instruction targeting a memory location within data memory **110**, the instruction operand includes a target address **200**. In a preferred embodiment address **200** is sixty-four bits long, with a subset of bits of low order, e.g., untranslated bits [**6**:**3**] of the target address and a subset of bits of high order used as the tag or index. Other implementations may vary some, all, or none of these values. As a generalization, an operand (i.e., a target address) of a load/store instruction into gating storage **100** is sometimes referred to herein as an access reference. Controls derived from this access reference include the tag, and the view, though in other implementations, the access reference and the controls may have different constructions and/or configurations from these preferred implementations.

[0042] View directory **105** is a special memory for views, also referred to herein as access method functions. An entry of directory **105** includes a view memory entry **210**. An entry of memory **110**, also referred to herein as a data storage location, holds data at an address derived from target address **200** and may include entry controls/flags/tags **220**. As shown in FIG. **2**, each entry **205** includes data **215** and one or more entry controls (e.g., bits or flags) **220**. Entry controls may include constructs such as a write-protect bit, a validity bit, control flag bit(s) discussed above that may modify operation of control logic **125** (e.g., a number of times to spin prior to an abort/exception), and/or other tags bits.

[0043] Each entry **210** includes a multiplicity of eight-byte views, any particular one of which is selectable by the value of the view bits [**6**:**3**] of the instruction operand. Further discussion of these views appears below, however for now it is sufficient to understand that each of the views is used to alter/enhance/modify the affect of the operand and/or the affect of an instruction upon its operand, or the method by which a processor operates upon the instruction and the instruction operand. In the example set forth herein, the instruction is a load/store instruction, the instruction operand is a memory location decoded into gating storage **100** such that sixteen views are available to redefine some aspect of the operation of the load/store instruction relative to this memory location. Specifically in the preferred embodiment, the views define possible synchronization constructs, functions, or methods that may be used accessing the particular memory location, such as using an Empty/Full primitive or a P/V semaphore. An inter-thread communication control unit (ITU), e.g., control logic **125** or the ITU as described in the incorporated patent application, accesses a memory location consistent with the desired synchronization construct selected by the appropriate view. In other cases, the constructs, functions, or methods maybe other than synchronization-related for load/store instructions. In some cases, other instructions may be processed through a memory system having access method functions applied dependent upon an associated operand.

[0044] Gating storage is an attribute of memory which may optionally be supported by processors implementing embodiments of the present invention. The user-mode load/store semantics of gating storage are identical with those of normal memory, except that completion of the operation may be blocked for unbounded periods of time. The distinguishing feature of gating storage is that outstanding load or store operations can be aborted and restarted. Preferably it is a TLB-mediated property of a virtual page whether or not a location is treated as gating storage (though other mechanisms may be implemented to identify gating storage locations).

[0045] When a load or store operation is performed on gating storage, no instructions beyond the load/store in program order are allowed to alter software-visible states of the system until a load result or store confirmation is returned from storage. In the event that an exception is taken using the thread context of an instruction stream which is blocked on a load/store to gating storage, or in the event where such a thread is halted by setting a ThreadStatus.H bit of the associated thread context, the pending load/store operation is aborted.

[0046] When a load or store is aborted, the abort is signaled to the storage subsystem, such that the operation unambiguously either completes or is abandoned without any side effects. When a load operation is abandoned, any hardware interlocks on the load dependence are released, so that the destination register may be used as an operand source, with its preload value.

[0047] After an aborted and abandoned load/store, a program counter as seen by the exception program counter register and the branch delay state as seen by a Cause.BD bit are set so as that an execution of an exception return (ERET) by the instruction stream associated by the thread context, or a clearing of the thread context halted state, causes a re-issue of the gating load/store. Gating storage accesses are never cached, and multiple stores to a gating storage address are never merged by a processor.

[0048] While the preceding description provides a complete description of a specific implementation of a gating storage for inter-thread communication in the synchronization of load/store instructions, the present invention has a broader implementation as well. In the more generalized case, gating storage provides a simple and efficient mechanism to extend an instruction set (particularly advantageous to processors implementing RISC instruction sets). This aspect of the present invention uses an operand of an instruction to modify, enhance, substitute, or otherwise affect an instruction using hardware features. In the load/store example used throughout this discussion, the gating storage adds multiple load/store commands to the basic instruction set, each of the added commands a variant of the basic command but including a hardware-managed instruction that implements a wide variety of synchronization constructs in the process of completing a load or a store. Normal loads/stores are still available by not including an operand within the gating storage. However, by modifying an instruction by use of special memory having special instruction functions/methods triggered from the operand may be implemented to extend many types of instructions in many different ways.

[0049] The inherently restricted number and complexity of instruction operand encodings in a "RISC" instruction set is augmented by adding computational semantics to basic instructions (e.g., RISC storage access instructions such as loads and stores), by using an instruction context (e.g., a portion of the storage address of the load or store as an opcode extension) to express a calculation or control function to be executed. This provides that an instruction may have a default instruction method and one or more variations that are implemented responsive to the instruction context. The preferred embodiment of the present invention described above provides for a standard load/store instruction to be extended using specifically chosen synchronization functions to be used instead of the standard instruction method when the target address is a data storage location in the gating storage. The preferred embodiment implements many "flavors" of the alternate synchronizing instruction method through the views (which may be referred to as access method functions) that tune a particular synchronizing load/store using the desired synchronization method.

[0050] A problem addressed by this "extension" aspect of the present invention is that some implementations of MIPS Technologies, Inc. processors required a range of synchronizing operations that could not reasonably be directly encoded in an extension to the MIPS32/MIPS64 instruction set. The present invention uses a memory-like space for designated interthread communication storage (ITC) that allowed a potentially very large number of synchronized, shared variables. A number of operations were to be available for each location: synchronized loads/stores, semaphore operations, bypass accesses to data and control information, etc., and it was desirable for synchronized loads and stores to be available for the full range of memory data types supported by the MIPS32/MIPS64 architecture: byte, halfword, word, and doubleword.

[0051] Rather than invent new instructions that perform distinct operations on the memory address expressed to the instruction, the preferred embodiment treats loads and stores to the designated ITC memory space as "load-plus-operation" and "store-plus-operation" instructions, where the "operation" is determined by decoding of a subset of the bits of the effective address of the load or store instruction. In the case of the preferred embodiment, this has evolved from using a pair of bits (**2****4** and **2****3**) as a four-element opcode space, performing Empty/Full synchronization, "forcing", "bypass", and "control" operations on the ITC variable referenced by the higher-order address bits, to the current scheme where four bits, **2****6** through **2****3**, are used to create a 16-opcode space, in which the system defines "bypass", "control", Empty/Full synchronization, Empty/Full "try" operations, Blocking semaphore "P" and "V" operations, and Semaphore "try" operations. This is but one example of how an instruction may be extended using a context of the instruction to determine an applicable instruction method to be used. Other extensions are possible to load/store instructions, other extensions are possible for other instructions, particularly those having an associated operand. However, other instructions may be extended by using some other contextual information to differentiate between instances in which a default instruction method is to be used and when an alternate instruction method. In the present context, instruction methods are the procedures implemented by a processor in executing an instruction. The extension aspect of the present invention provides for a different set of procedures to be used when executing the

same instruction when a context of the instruction indicates that a different implementation should be used.

[0052] The invention described in this application may, of course, be embodied in hardware; e.g., within or coupled to a Central Processing Unit ("CPU"), microprocessor, microcontroller, System on Chip ("SOC"), or any other programmable device. Additionally, embodiments may be embodied in software (e.g., computer readable code, program code, instructions and/or data disposed in any form, such as source, object or machine language) disposed, for example, in a computer usable (e.g., readable) medium configured to store the software. Such software enables the function, fabrication, modeling, simulation, description and/or testing of the apparatus and processes described herein. For example, this can be accomplished through the use of general programming languages (e.g., C, C++), GDSII databases, hardware description languages (HDL) including Verilog HDL, VHDL, AHDL (Altera HDL) and so on, or other available programs, databases, and/or circuit (i.e., schematic) capture tools. Such software can be disposed in any known computer usable medium including semiconductor, magnetic disk, optical disc (e.g., CD-ROM, DVD-ROM, etc.) and as a computer data signal embodied in a computer usable (e.g., readable) transmission medium (e.g., carrier wave or any other medium including digital, optical, or analog-based medium). As such, the software can be transmitted over communication networks including the Internet and intranets. Embodiments of the invention embodied in software may be included in a semiconductor intellectual property core (e.g., embodied in HDL) and transformed to hardware in the production of integrated circuits. Additionally, implementations of the present invention may be embodied as a combination of hardware and software.

[0053] In the description herein, numerous specific details are provided, such as examples of components and/or methods, to provide a thorough understanding of embodiments of the present invention. One skilled in the relevant art will recognize, however, that an embodiment of the invention can be practiced without one or more of the specific details, or with other apparatus, systems, assemblies, methods, components, materials, parts, and/or the like. In other instances, well-known structures, materials, or operations are not specifically shown or described in detail to avoid obscuring aspects of embodiments of the present invention.

[0054] A "computer-readable medium" for purposes of embodiments of the present invention may be any medium that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, system or device. The computer readable medium can be, by way of example only but not by limitation, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, system, device, propagation medium, or computer memory.

[0055] A "processor" or "process" includes any human, hardware and/or software system, mechanism or component that processes data, signals or other information. A processor may include a system with a general-purpose central processing unit, multiple processing units, dedicated circuitry for achieving functionality, or other systems. Processing need not be limited to a geographic location, or have temporal limitations. For example, a processor may perform its functions in "real time,""offline," in a "batch mode," etc. Portions of processing may be performed at different times and at different locations, by different (or the same) processing systems.

[0056] Reference throughout this specification to "one embodiment", "an embodiment", or "a specific embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention and not necessarily in all embodiments. Thus, respective appearances of the phrases "in one embodiment", "in an embodiment", or "in a specific embodiment" in various places throughout this specification are not necessarily referring to the same embodiment. Furthermore, the particular features, structures, or characteristics of any specific embodiment of the present invention may be combined in any suitable manner with one or more other embodiments. It is to be understood that other variations and modifications of the embodiments of the present invention described and illustrated herein are possible in light of the teachings herein and are to be considered as part of the spirit and scope of the present invention.

[0057] Embodiments of the invention may be implemented by using a programmed general purpose digital computer, by using application specific integrated circuits, programmable logic devices, field programmable gate arrays, optical, chemical, biological, quantum or nanoengineered systems, components and mechanisms may be used. In general, the functions of the present invention may be achieved by any means as is known in the art. Distributed, or networked systems, components and circuits may be used. Communication, or transfer, of data may be wired, wireless, or by any other means.

[0058] It will also be appreciated that one or more of the elements depicted in the drawings/figures may also be implemented in a more separated or integrated manner, or even removed or rendered as inoperable in certain cases, as is useful in accordance with a particular application. It is also within the spirit and scope of the present invention to implement a program or code that may be stored in a machine-readable medium or transmitted using a carrier wave to permit a computer to perform any of the methods described above.

[0059] Additionally, any signal arrows in the drawings/Figures should be considered only as exemplary, and not limiting, unless otherwise specifically noted. Furthermore, the term "or" as used herein is generally intended to mean "and/or" unless otherwise indicated. Combinations of components or steps will also be considered as being noted, where terminology is foreseen as rendering the ability to separate or combine is unclear.

[0060] As used in the description herein and throughout the claims that follow, "a", "an", and "the" includes plural references unless the context clearly dictates otherwise. Also, as used in the description herein and throughout the claims that follow, the meaning of "in" includes "in" and "on" unless the context clearly dictates otherwise.

[0061] The foregoing description of illustrated embodiments of the present invention, including what is described in the Abstract, is not intended to be exhaustive or to limit the invention to the precise forms disclosed herein. While specific embodiments of, and examples for, the invention are

described herein for illustrative purposes only, various equivalent modifications are possible within the spirit and scope of the present invention, as those skilled in the relevant art will recognize and appreciate. As indicated, these modifications may be made to the present invention in light of the foregoing description of illustrated embodiments of the present invention and are to be included within the spirit and scope of the present invention.

[0062] Thus, while the present invention has been described herein with reference to particular embodiments thereof, a latitude of modification, various changes and substitutions are intended in the foregoing disclosures, and it will be appreciated that in some instances some features of embodiments of the invention will be employed without a corresponding use of other features without departing from the scope and spirit of the invention as set forth. Therefore, many modifications may be made to adapt a particular situation or material to the essential scope and spirit of the present invention. It is intended that the invention not be limited to the particular terms used in following claims and/or to the particular embodiment disclosed as the best mode contemplated for carrying out this invention, but that the invention will include any and all embodiments and equivalents falling within the scope of the appended claims.

[0063] The above-described arrangements of apparatus and methods are merely illustrative of applications of the principles of this invention and many other embodiments and modifications may be made without departing from the spirit and scope of the invention as defined in the claims.

[0064] These and other novel aspects of the present invention will be apparent to those of ordinary skill in the art upon review of the drawings and the remaining portions of the specification. Therefore, the scope of the invention is to be determined solely by the appended claims.

What is claimed is:

1. A data storage location access control system, comprising:

a memory system including a plurality of data storage locations, each said data storage location associated with a set of views including a first view and a second view with said memory system producing a particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location; and

a controller, coupled to said memory system, for controlling access to the data storage location using said particular one view.

2. The data storage location access control system of claim 1 wherein said address is an operand of a load/store instruction for the data storage location.

3. The data storage location access control system of claim 1 wherein said memory system is coupled to a plurality of virtual processing elements (VPEs).

4. The data storage location access control system of claim 1 wherein said set of control bits includes a first control subset derived from a number N of high order bits of said memory address and a second control subset derived from a number M of untranslated bits of said memory address.

5. The data storage location access control system of claim 1 wherein said first view includes one or more Empty/Full synchronization primitive state bits and said second view includes one or more P/V semaphore synchronization primitive state bits.

6. The data storage location access control system of claim 4 wherein first view includes one or more Empty/Full synchronization primitive state bits and said second view includes one or more P/V semaphore synchronization primitive state bits.

7. The data storage location access control system of claim 6 wherein said particular one set of views is selected responsive to said first control subset and wherein said particular one view is selected responsive to said second control subset.

8. A data storage location access control method, comprising:

a) applying an access instruction for the data storage location to a memory system, said memory system including a plurality of data storage locations, each said data storage location associated with a set of views including a first view and a second view with said memory system producing a particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location;

b) producing said particular one view from said particular one set of views; and

c) controlling access to the data storage location using said particular one view.

9. The data storage location access control method of claim 8 wherein said address is an operand of a load/store instruction for the data storage location.

10. The data storage location access control method of claim 8 wherein said memory system is coupled to a plurality of virtual processing elements (VPEs).

11. The data storage location access control method of claim 8 wherein said set of control bits includes a first control subset derived from a number N of high order bits of said address and a second control subset derived from a number M of untranslated bits of said memory address.

12. The data storage location access control method of claim 8 wherein said first view includes one or more Empty/Full synchronization primitive state bits and said second view includes one or more P/V semaphore synchronization primitive state bits.

13. The data storage location access control method of claim 11 wherein said first view includes one or more Empty/Full synchronization primitive state bits and said second view includes one or more P/V semaphore synchronization primitive state bits.

14. The data storage location access control method of claim 13 further comprising selecting, responsive to said first control subset, said particular one set; and selecting, responsive to said second control subset, said particular one view.

15. A computer program product comprising a computer readable medium carrying program instructions for accessing a data storage location when executed using a computing system, the executed program instructions executing a method, the method comprising:

a) applying an access instruction for the data storage location to a memory system, said memory system including a plurality of data storage locations, each said data storage location associated with a set of views including a first view and a second view with said memory system producing a particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location;

b) producing said particular one view from said particular one set of views; and

c) controlling access to the data storage location using said particular one view.

16. The computer program product of claim 15 wherein said address is an operand of a load/store instruction for the data storage location.

17. The computer program product of claim 15 wherein said memory system is coupled to a plurality of virtual processing elements (VPEs).

18. The computer program product of claim 15 wherein said set of control bits includes a first control subset derived from a number N of high order bits of said memory address and a second control subset derived from a number M of untranslated bits of said memory address.

19. The computer program product of claim 15 wherein said first view includes one or more Empty/Full synchronization primitive state bits and said second view includes one or more P/V semaphore synchronization primitive state bits.

20. The computer program product of claim 18 wherein said first view includes one or more Empty/Full synchronization primitive state bits and said second view includes one or more P/V semaphore synchronization primitive state bits.

21. The computer program product of claim 20 further comprising selecting, responsive to said first control subset, said particular one set; and selecting, responsive to said second control subset, said particular one view.

22. A propagated signal on which is carried computer-executable instructions which when executed by a computing system performs a method, the method comprising:

a) applying an access instruction for the data storage location to a memory system, said memory system including a plurality of data storage locations, each said data storage location associated with a set of views including a first view and a second view with said memory system producing a particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location;

b) producing said particular one view from said particular one set of views; and

c) controlling access to the data storage location using said particular one view.

23. The propagated signal of claim 22 wherein said address is an operand of a load/store instruction for the data storage location.

24. The propagated signal of claim 22 wherein said memory system is coupled to a plurality of virtual processing elements (VPEs).

25. The propagated signal of claim 22 wherein said set of control bits includes a first control subset derived from a number N of high order bits of said memory address and a second control subset derived from a number M of untranslated bits of said memory address.

26. The propagated signal of claim 22 wherein said first view includes one or more Empty/Full synchronization primitive state bits and said second view includes one or more P/V semaphore synchronization primitive state bits.

27. The propagated signal of claim 25 wherein said first view includes one or more Empty/Full synchronization primitive state bits and said second view includes one or more P/V semaphore synchronization primitive state bits.

28. The propagated signal of claim 27 further comprising selecting, responsive to said first control subset, said particular one set; and selecting, responsive to said second control subset, said particular one view.

29. A data storage location access control apparatus, comprising:

means for applying an access instruction for the data storage location to a memory system, said memory system including a plurality of data storage locations, each said data storage location associated with a set of views including a first view and a second view with said memory system producing a particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location;

means for producing said particular one view from said particular one set of views; and

means for controlling access to the data storage location using said particular one view

30. A method for controlling access to a data storage location, comprising:

a) retrieving a particular one view from a memory system including a plurality of data storage locations, each said data storage location associated with a set of views including one or more views with said memory system producing said particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location; and

b) accessing the data storage location using said particular one view.

31. A computer program product comprising a computer readable medium carrying program instructions for accessing a data storage location when executed using a computing system, the executed program instructions executing a method, the method comprising:

a) retrieving a particular one view from a memory system including a plurality of data storage locations, each said data storage location associated with a set of views including one or more views with said memory system producing said particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location; and

b) accessing the data storage location using said particular one view.

32. A propagated signal on which is carried computer-executable instructions which when executed by a computing system performs a method, the method comprising:

a) retrieving a particular one view from a memory system including a plurality of data storage locations, each said data storage location associated with a set of views including one or more views with said memory system producing said particular one view from a particular one set of views associated with the data storage location responsive to a set of control bits derived from an address identifying the data storage location; and

b) accessing the data storage location using said particular one view.

33. An apparatus for extending a load/store instruction having a target address, the apparatus comprising:

a memory system having a view associated with a data storage location identified by a tag derived from the target address, said data storage location associated with the load/store instruction, said memory system responsive to the target address to produce a particular view for the load/store instruction from said memory system; and

a controller, coupled to said data storage location and to said memory system, for implementing a an load/store method for the load/store instruction using said particular view.

34. The apparatus of claim 33 wherein said target address is an operand of the load/store instruction for the data storage location.

35. The apparatus of claim 33 wherein said particular view facilitates said controller in using said load/store method for the load/store instruction to said data storage location.

36. The apparatus of claim 33 wherein said memory system associates both a first view and a second view with said data storage location and wherein said memory system is responsive to the target address to select one of said first view and said second view as said particular view.

37. The apparatus of claim 36 wherein each said view includes one or more data structures for communicating one or more synchronization primitives state bits.

38. A method for extending a load/store instruction having a target address, the method comprising:

a) producing, responsive to the target address, a particular view for the load/store instruction from a memory system, said memory system having a view associated with a data storage location identified by a tag derived from the target address, said data storage location associated with the load/store instruction; and

b) implementing a load/store method for the load/store instruction using said particular view.

39. The method of claim 38 wherein said target address is an operand of the load/store instruction for the data storage location.

40. The method of claim 38 wherein said particular view facilitates said controller to implement said load/store method for said load/store instruction to said data storage location.

41. The method of claim 38 further comprising controlling, responsive to said particular view, access of said data storage location by the load/store instruction using said load/store method.

42. The method of claim 41 wherein said data storage location is a memory shared among a plurality of concurrent processes and said load/store method is a synchronization method for synchronizing accesses to said data storage location by said plurality of concurrent processes.

43. The method of claim 38 further wherein said view associated with said data storage location is a first view and wherein said memory system has a second view associated with said data storage location.

44. The method of claim 43 further comprising selecting, responsive to a set of control bits derived from the target address, one of said first and second views as said particular view.

45. A computer program product comprising a computer readable medium carrying program instructions for extending a load/store instruction having a target address when executed using a computing system, the executed program instructions executing a method, the method comprising:

a) producing, responsive to the target address, a particular view for the load/store instruction from a memory system, said memory system having a view associated with a data storage location identified by a tag derived from the target address, said data storage location associated with the load/store instruction; and

b) implementing a load/store method for the load/store instruction using said particular view.

46. The computer program product of claim 45 wherein said target address is an operand of the load/store instruction for the data storage location.

47. The computer program product of claim 45 wherein said particular view facilitates implementation of said load/store method for said load/store instruction to said data storage location.

48. The computer program product of claim 45 further comprising controlling, responsive to said particular view, access of said data storage location by the load/store instruction using said load/store method.

49. The computer program product of claim 48 wherein said data storage location is a memory shared among a plurality of concurrent processes and said load/store method is a synchronization method for synchronizing accesses to said data storage location by said plurality of concurrent processes.

50. The computer program product of claim 45 wherein said view associated with said data storage location is a first view and wherein said memory system has a second view associated with said data storage location.

51. A propagated signal on which is carried computer-executable instructions which when executed by a computing system performs a method, the method comprising:

a) producing, responsive to the target address, a particular view for the load/store instruction from a memory system, said memory system having a view associated with a data storage location identified by a tag derived from the target address, said data storage location associated with the load/store instruction; and

b) implementing a load/store method for the load/store instruction using said particular view.

52. The propagated signal of claim 51 wherein said target address is an operand of the load/store instruction for the data storage location.

53. The propagated signal of claim 51 wherein said particular view facilitates implementation of said load/store method for said load/store instruction to said data storage location.

54. The propagated signal of claim 51 further comprising controlling, responsive to said particular view, access of said data storage location by the load/store instruction using said load/store method.

55. The propagated signal of claim 54 wherein said data storage location is a memory shared among a plurality of concurrent processes and said load/store method is a synchronization method for synchronizing accesses to said data storage location by said plurality of concurrent processes.

56. The propagated signal of claim 51 wherein said view associated with said data storage location is a first view and wherein said memory system has a second view associated with said data storage location.

57. A shared resource access control system, comprising:

a storage structure responsive to a plurality of control references with each said control reference derived from an access reference identifying the shared resource, said storage structure including a plurality of sets of access method functions at least one set associated with each of a plurality of shared resources, with each said set of access method functions including a first access method function and a second access method function with said storage structure producing a particular one access method function from a particular one set associated with the shared resource responsive to said control references; and

a controller, coupled to said storage structure, for controlling access to the shared resource using said particular one access method function.

58. A shared resource access control method, comprising:

a) applying an access instruction for a shared resource identified by an access reference to a storage structure that includes said shared resource, said storage structure responsive to a plurality of control references with each said control reference derived from said access reference, said storage structure including a plurality of sets of access method functions at least one set associated with each of a plurality of shared resources, with each said set of access method functions including a first access method function and a second access method function;

b) producing a particular one access method function from a particular one set associated with the shared resource responsive to said control references; and

c) controlling access to the shared resource using said particular one access rule function.

59. A computer program product comprising a computer readable medium carrying program instructions for accessing a shared resource when executed using a computing system, the executed program instructions executing a method, the method comprising:

a) applying an access instruction for a shared resource identified by an access reference to a storage structure that includes said shared resource, said storage structure responsive to a plurality of control references with each said control reference derived from said access reference, said storage structure including a plurality of sets of access method functions at least one set associated with each of a plurality of shared resources, with each said set of access method functions including a first access method function and a second access method function;

b) producing a particular one access method function from a particular one set associated with the shared resource responsive to said control references; and

c) controlling access to the shared resource using said particular one access rule function.

60. A propagated signal on which is carried computer-executable instructions which when executed by a computing system performs a method, the method comprising:

a) applying an access instruction for a shared resource identified by an access reference to a storage structure that includes said shared resource, said storage structure responsive to a plurality of control references with each said control reference derived from said access reference, said storage structure including a plurality of sets of access method functions at least one set associated with each of a plurality of shared resources, with each said set of access method functions including a first access method function and a second access method function;

b) producing a particular one access method function from a particular one set associated with the shared resource responsive to said control references; and

c) controlling access to the shared resource using said particular one access rule function.

61. A method for selectively extending an instruction having a default instruction method, the method comprising:

a) determining whether the instruction is to be extended through use of an alternate instruction method for the instruction responsive to a context of the instruction wherein said alternate instruction method differs from the default instruction method; and

b) using said alternate instruction method for the instruction when the instruction is to be extended.

62. The method of claim 61 wherein said context includes an operand associated with the instruction.

63. The method of claim 62 wherein said alternate instruction method uses an instruction method function.

64. The method of **63** wherein said instruction method function includes one or more state bits stored in a data structure associated with a resource used in said alternate instruction method.

65. The method of claim 62 wherein the instruction is a memory access instruction with a target address included in said operand, the instruction providing for the default instruction method to access a first memory location identified by said operand, wherein said alternate instruction method includes implementation of a synchronizing memory access with respect to a second memory location identified by said operand; and wherein a particular instruction method to be implemented is determined by said operand.

66. The method of claim 64 wherein the instruction is a memory access instruction with a target address included in said operand and said resource is a memory storage location in a memory system identified by said target address, the instruction providing for the default instruction method to access a first memory location, wherein said alternate instruction method includes implementation of a synchronizing memory access with respect to a second memory location identified by said operand; and wherein a particular instruction method to be implemented is determined by said

operand with said operand selecting said one or more state bits from said data structure when said particular instruction method is said alternate instruction method.

67. The method of claim 61 wherein said alternate instruction method includes a plurality of alternate implementations for each said context wherein a particular one alternate implementation is selected for said alternate instruction method responsive to said context.

68. The method of claim 67 wherein said context includes an operand associated with the instruction and said particular one alternate implementation is selected for said alternate instruction method responsive to said operand.

69. The method of claim 68 wherein said alternate implementations use a plurality of alternate instruction method functions, at least one function for each implementation.

70. The method of **69** wherein each said instruction method function for each context includes one or more state bits stored in a common data structure associated with said operand and wherein a particular set of said one or more state bits is selected from said data structure responsive to said operand.

71. The method of claim 67 wherein the instruction is a memory access instruction with a target address included in said operand, the instruction providing for the default instruction method to access a first memory location identified by said operand, wherein said alternate instruction method includes different implementations of a synchronizing memory access with respect to a second memory location identified by said operand; and wherein a particular instruction method to be implemented is determined by said operand.

72. The method of claim 70 wherein the instruction is a memory access instruction with a target address included in said operand and said resource is a memory storage location in a memory system identified by said target address, the instruction providing for the default instruction method to access a first memory location, wherein said alternate instruction method includes different implementation of a synchronizing memory access with respect to a second memory location identified by said operand; and wherein a particular instruction method to be implemented is determined by said operand with said operand selecting said one or more state bits of said particular set from said data structure when said particular instruction method is said alternate instruction method.

73. A computer program product comprising a computer readable medium carrying program instructions for selectively extending an instruction having a default instruction method when executed using a computing system, the executed program instructions executing a method, the method comprising:

a) determining whether the instruction is to be extended through use of an alternate instruction method for the instruction responsive to a context of the instruction wherein said alternate instruction method differs from the default instruction method; and

b) using said alternate instruction method for the instruction when the instruction is to be extended.

74. A propagated signal on which is carried computer-executable instructions which when executed by a computing system performs a method, the method comprising:

a) determining whether the instruction is to be extended through use of an alternate instruction method for the instruction responsive to a context of the instruction wherein said alternate instruction method differs from the default instruction method; and

b) using said alternate instruction method for the instruction when the instruction is to be extended.

75. An apparatus, comprising:

means for determining whether the instruction is to be extended through use of an alternate instruction method for the instruction responsive to a context of the instruction wherein said alternate instruction method differs from the default instruction method; and

means for using said alternate instruction method for the instruction when the instruction is to be extended.

76. An apparatus for selectively extending an instruction having a default instruction method, comprising:

a context evaluator for determining whether the instruction is to be extended through use of an alternate instruction method for the instruction responsive to a context of the instruction wherein said alternate instruction method differs from the default instruction method; and

a controller using said alternate instruction method for the instruction when the instruction is to be extended.

77. A method for selectively extending a memory load/store instruction having a default instruction method, the method comprising:

a) determining whether the instruction is to be extended through use of a synchronizing instruction method applied to the instruction responsive to an operand of the instruction, said operand including a target address identifying a target memory location in a memory system, said target memory location different for the default instruction method and said synchronizing instruction method with at least of a portion of said target address selecting a particular set of one or more synchronization primitive data bits from a data structure associated with said target memory location appropriate for the desired synchronization instruction method to be applied; and

b) using said synchronizing instruction method for the instruction when the instruction is to be extended so that said particular set of synchronization primitive data bits influence the memory load/store instruction relative to said target memory location wherein the default instruction method is not influenced by any synchronization primitive data bits.

78. A computer program product comprising a computer readable medium carrying program instructions for selectively extending an instruction having a default instruction method when executed using a computing system, the executed program instructions executing a method, the method comprising:

a) determining whether the instruction is to be extended through use of a synchronizing instruction method applied to the instruction responsive to an operand of the instruction, said operand including a target address identifying a target memory location in a memory system, said target memory location different for the default instruction method and said synchronizing instruction method with at least of a portion of said

target address selecting a particular set of one or more synchronization primitive data bits from a data structure associated with said target memory location appropriate for the desired synchronization instruction method to be applied; and

b) using said synchronizing instruction method for the instruction when the instruction is to be extended so that said particular set of synchronization primitive data bits influence the memory load/store instruction relative to said target memory location wherein the default instruction method is not influenced by any synchronization primitive data bits.

79. A propagated signal on which is carried computer-executable instructions which when executed by a computing system performs a method, the method comprising:

a) determining whether the instruction is to be extended through use of a synchronizing instruction method applied to the instruction responsive to an operand of the instruction, said operand including a target address identifying a target memory location in a memory system, said target memory location different for the default instruction method and said synchronizing instruction method with at least of a portion of said target address selecting a particular set of one or more synchronization primitive data bits from a data structure associated with said target memory location appropriate for the desired synchronization instruction method to be applied; and

b) using said synchronizing instruction method for the instruction when the instruction is to be extended so that said particular set of synchronization primitive data bits influence the memory load/store instruction relative to said target memory location wherein the default instruction method is not influenced by any synchronization primitive data bits.

80. An apparatus, comprising:

means for determining whether the instruction is to be extended through use of a synchronizing instruction method applied to the instruction responsive to an operand of the instruction, said operand including a target address identifying a target memory location in a memory system, said target memory location different for the default instruction method and said synchronizing instruction method with at least of a portion of said target address selecting a particular set of one or more synchronization primitive data bits from a data structure associated with said target memory location appropriate for the desired synchronization instruction method to be applied; and

means for using said synchronizing instruction method for the instruction when the instruction is to be extended so that said particular set of synchronization primitive data bits influence the memory load/store instruction relative to said target memory location wherein the default instruction method is not influenced by any synchronization primitive data bits.

81. An apparatus for selectively extending an instruction having a default instruction method, comprising:

a context evaluator for determining whether the instruction is to be extended through use of a synchronizing instruction method applied to the instruction responsive to an operand of the instruction, said operand including a target address identifying a target memory location in a memory system, said target memory location different for the default instruction method and said synchronizing instruction method with at least of a portion of said target address selecting a particular set of one or more synchronization primitive data bits from a data structure associated with said target memory location appropriate for the desired synchronization instruction method to be applied; and

a controller using said synchronizing instruction method for the instruction when the instruction is to be extended so that said particular set of synchronization primitive data bits influence the memory load/store instruction relative to said target memory location wherein the default instruction method is not influenced by any synchronization primitive data bits.

82. A method for selectively extending a load/store instruction having a default load/store method, the method comprising:

a) determining whether the load/store instruction is to be extended through use of a synchronizing load/store method for the load/store instruction responsive to a target address of the load/store instruction wherein said synchronizing load/store method differs from the default load/store method; and

b) using said synchronizing load/store method for the load/store instruction when the load/store instruction is to be extended.

83. A computer program product comprising a computer readable medium carrying program instructions for selectively extending an instruction having a default instruction method when executed using a computing system, the executed program instructions executing a method, the method comprising:

a) determining whether the load/store instruction is to be extended through use of a synchronizing load/store method for the load/store instruction responsive to a target address of the load/store instruction wherein said synchronizing load/store method differs from the default load/store method; and

b) using said synchronizing load/store method for the load/store instruction when the load/store instruction is to be extended.

84. A propagated signal on which is carried computer-executable instructions which when executed by a computing system performs a method, the method comprising:

a) determining whether the load/store instruction is to be extended through use of a synchronizing load/store method for the load/store instruction responsive to a target address of the load/store instruction wherein said synchronizing load/store method differs from the default load/store method; and

b) using said synchronizing load/store method for the load/store instruction when the load/store instruction is to be extended.

85. An apparatus for selectively extending an instruction having a default instruction method, comprising:

a context evaluator for determining whether the load/store instruction is to be extended through use of a synchro-

nizing load/store method for the load/store instruction responsive to a target address of the load/store instruction wherein said synchronizing load/store method differs from the default load/store method; and

a controller using said synchronizing load/store method for the load/store instruction when the load/store instruction is to be extended.

86. An apparatus for selectively extending an instruction having a default instruction method, comprising:

means for determining whether the load/store instruction is to be extended through use of a synchronizing load/store method for the load/store instruction responsive to a target address of the load/store instruction wherein said synchronizing load/store method differs from the default load/store method; and

means for using said synchronizing load/store method for the load/store instruction when the load/store instruction is to be extended.

87. A data storage system responsive to a load/store instruction having a target address, comprising:

a data memory having a plurality of data storage locations, said data memory identifying a particular data storage location responsive to the target address; and

a view directory storing a view associated with each data storage location of said plurality of storage locations, said view directory identifying a particular one view responsive to the target address.

88. A data storage method responsive to a load/store instruction having a target address, the method comprising:

a) identifying, responsive to the target address, a particular data storage location from a data memory having a plurality of data storage locations; and

b) identifying, responsive to the target address, a particular one view from a view directory storing a view associated with each data storage location of said plurality of storage locations.

89. A computer program product comprising a computer readable medium carrying program instructions for producing data responsive to a load/store instruction having a target address when executed using a computing system, the executed program instructions executing a method, the method comprising:

a) identifying, responsive to the target address, a particular data storage location from a data memory having a plurality of data storage locations; and

b) identifying, responsive to the target address, a particular one view from a view directory storing a view associated with each data storage location of said plurality of storage locations.

90. A propagated signal on which is carried computer-executable instructions which when executed by a computing system performs a method, the method comprising:

a) identifying, responsive to the target address, a particular data storage location from a data memory having a plurality of data storage locations; and

b) identifying, responsive to the target address, a particular one view from a view directory storing a view associated with each data storage location of said plurality of storage locations.

91. A data storage apparatus responsive to a load/store instruction having a target address, comprising:

means for identifying, responsive to the target address, a particular data storage location from a data memory having a plurality of data storage locations; and

means for identifying, responsive to the target address, a particular one view from a view directory storing a view associated with each data storage location of said plurality of storage locations.

* * * * *