US 20060224857A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2006/0224857 A1

O'Connor et al. (43) Pub. Date: Oct. 5, 2006

(54) LOCKING ENTRIES INTO TRANSLATION LOOKASIDE BUFFERS

(76) Inventors: **Dennis M. O'Connor**, Chandler, AZ (US); **Stephen J. Strazdus**, Chandler, AZ (US)

Correspondence Address:
TROP PRUNER & HU, PC
1616 S. VOSS ROAD, SUITE 750
HOUSTON, TX 77057-2631 (US)

(21) Appl. No.: **11/092,432**

(22) Filed: **Mar. 29, 2005**

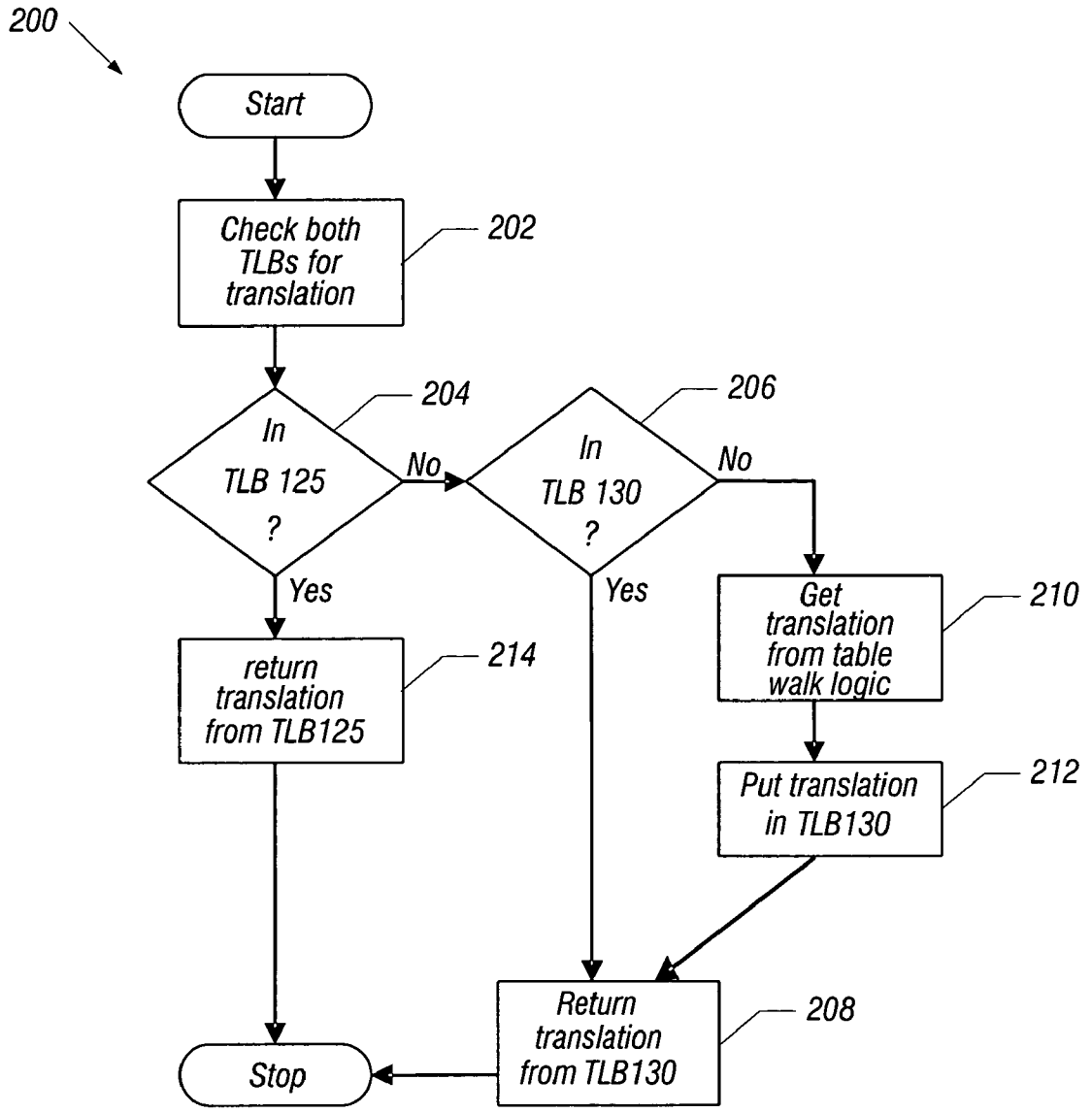
Publication Classification

(51) **Int. Cl.**
G06F 12/00 (2006.01)

(52) **U.S. Cl.** **711/202**

(57) **ABSTRACT**

Two translation lookaside buffers may be provided for simpler operation in some embodiments. A hardware managed lookaside buffer may handle traditional operations. A software managed lookaside buffer may be particularly involved in locking particular translations. As a result, the software's job is made simpler since it has a relatively simpler, software managed translation lookaside buffer to manage for locking translations.



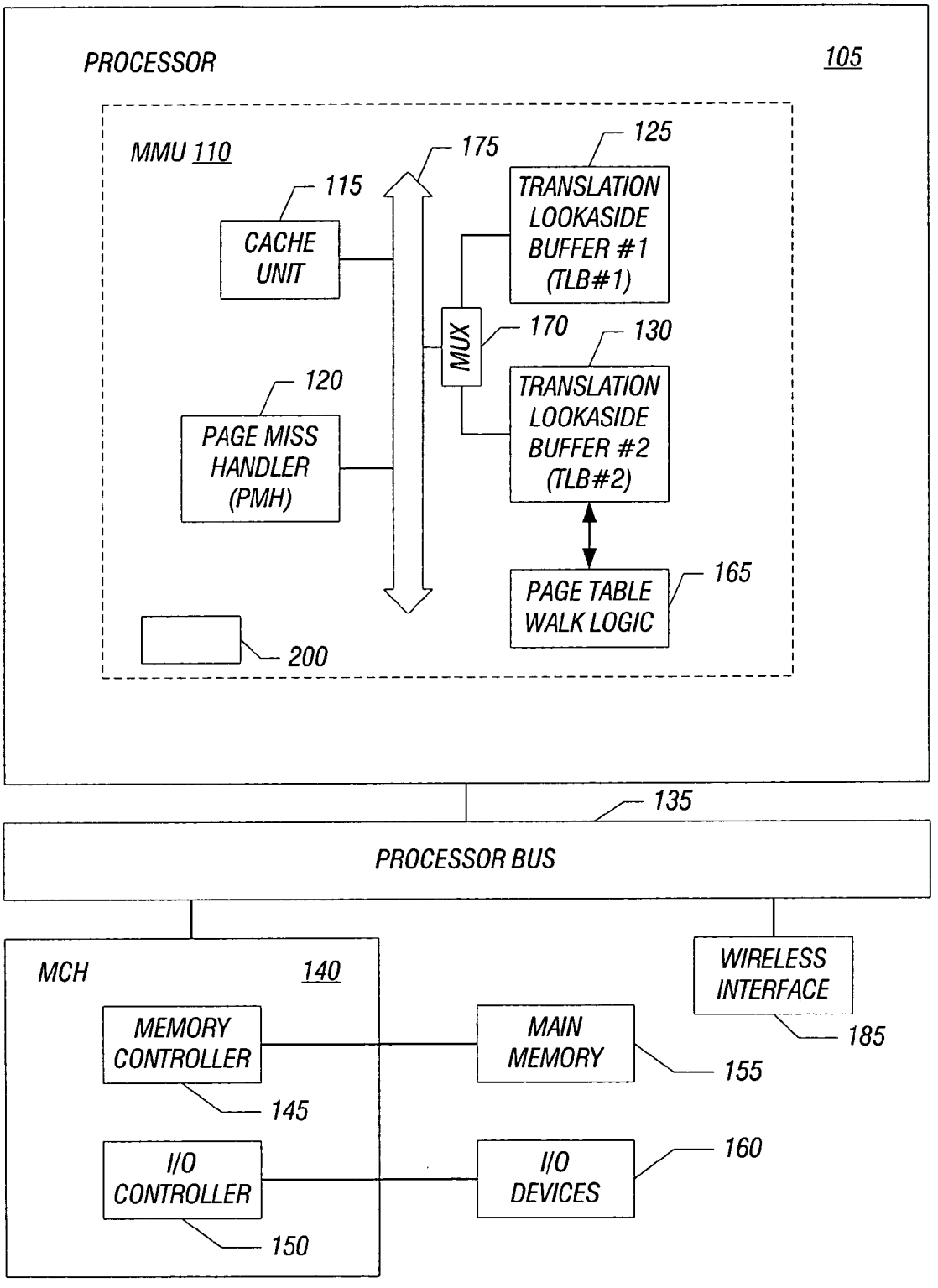

PROCESSOR
105
MMU 110
175
125
115
CACHE UNIT
TRANSLATION LOOKASIDE BUFFER #1 (TLB#1)
MUX
170
130
120
PAGE MISS HANDLER (PMH)
TRANSLATION LOOKASIDE BUFFER #2 (TLB#2)
165
PAGE TABLE WALK LOGIC
200
135
PROCESSOR BUS
MCH
140
WIRELESS INTERFACE
185
MEMORY CONTROLLER
MAIN MEMORY
155
145
I/O CONTROLLER
I/O DEVICES
160
150

FIG. 1

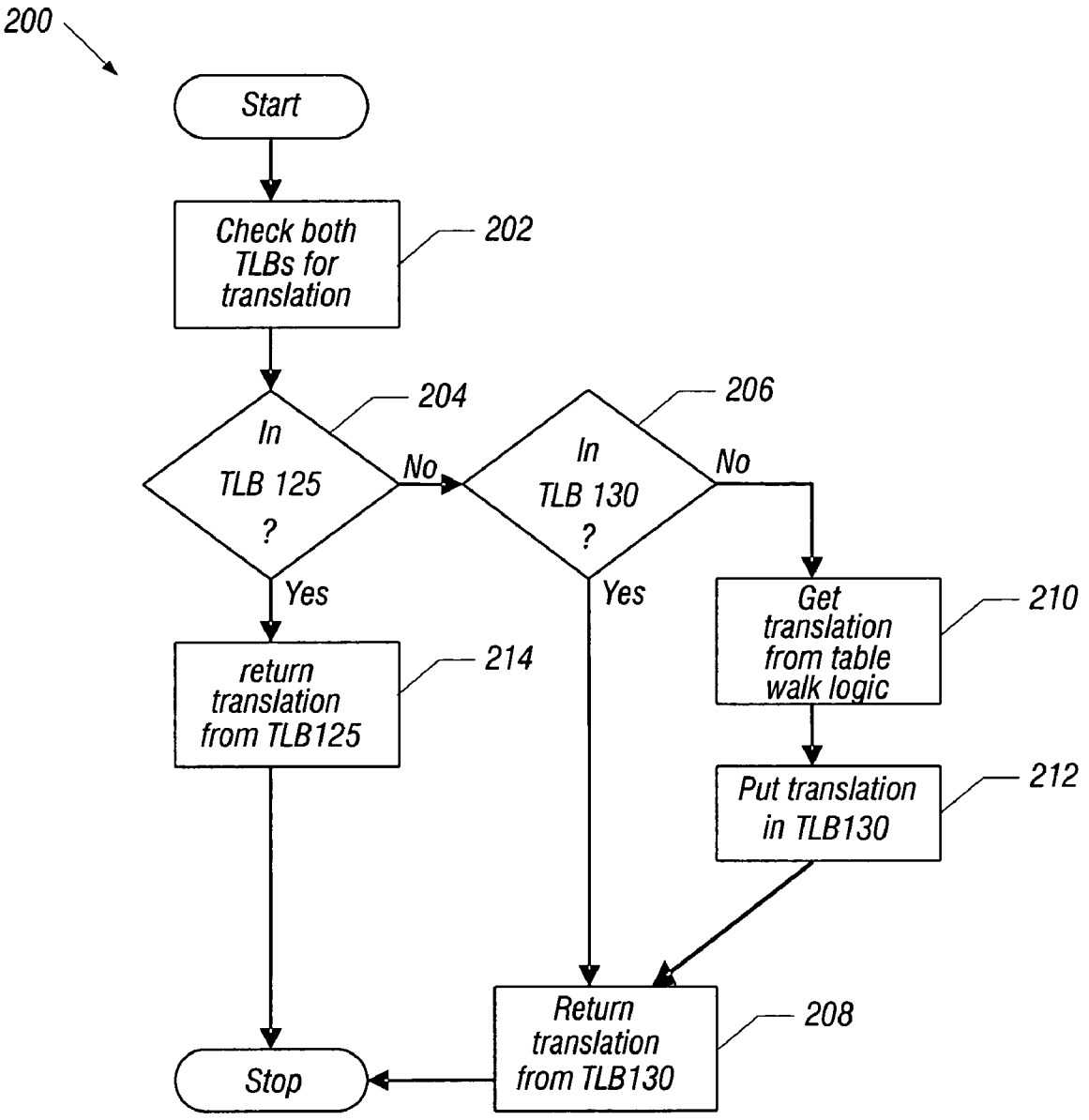

200
Start
Check both TLBs for translation
202
In TLB 125 ?
204
No
In TLB 130 ?
206
No
Yes
Yes
return translation from TLB125
214
Get translation from table walk logic
210
Put translation in TLB130
212
Return translation from TLB130
208
Stop

FIG. 2

LOCKING ENTRIES INTO TRANSLATION LOOKASIDE BUFFERS

BACKGROUND

[0001] This invention relates generally to computer systems and, particularly, to handling of memory access operations.

[0002] To facilitate memory access operations, a translation-lookaside buffer (TLB) is employed by microprocessors to provide the translation of linear addresses to physical addresses. The TLB caches linear addresses and corresponding physical addresses.

[0003] In use the TLB is initially accessed to determine whether the TLB contains the physical address corresponding to a linear address, identifying a desired memory location. If the linear address is found within the TLB, a "hit" is said to have occurred. The physical address is merely loaded out of the TLB. If the linear and physical addresses are not cached within the TLB, then a TLB "miss" is said to have occurred. In which case, a page miss handler is used to perform a page table walk to determine the physical address corresponding to the desired linear address.

[0004] TLBs allow some entries to be locked. Some performance critical translations may be locked into the TLB to ensure that the slower page table walk operation will not be triggered when one of those translations is needed. However, determining whether there was a place to lock a particular translation often involves a detailed knowledge of the TLB architecture and detailed tracking of the entries that were locked. The architecture of the TLB may limit the kinds of entries that may be locked into it.

[0005] Thus, there is a need for better ways to lock entries in translation lookaside buffers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] **FIG. 1** is a block diagram of one embodiment of the present invention; and

[0007] **FIG. 2** is a flow chart for the embodiment shown in **FIG. 1** in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

[0008] A computer system, shown in **FIG. 1**, includes a processor **105** coupled to a processor bus **135**. The processor **105** may be a general purpose microprocessor, a complex instruction set computer, a reduced instruction set computer, a very long instruction word, or a hybrid architecture, to mention a few examples. In one embodiment, the processor **105** is an out-of-order processor capable of performing operations either out of order or speculatively. However, the present invention is applicable to any type of processor, including out-of-order and in-order processors.

[0009] Also coupled to the processor bus **135** is a memory controller hub (MCH) **140**. The MCH **140** includes a memory controller **145** and an input/output (I/O) controller **150**. In the illustrated embodiment, a main memory **155** is coupled to the processor bus **135** through the MCH **140**. The processor **105** generates instructions (also referred to herein as micro-operations or "micro-ops"), such as memory loads, stores, and pre-fetches. The micro-ops are, in general, in a sequence that may differ from the sequence in which the instructions appear within a computer program. Micro-ops which involve memory accesses, such as memory loads, stores, and pre-fetches, are executed by a memory management unit (MMU) **110**.

[0010] The MMU **110** includes, among other things, a cache unit **115**, a page miss handler (PMH) **120**, a software managed translation lookaside buffer **125**, a hardware managed translation lookaside buffer **130** with a page table walk logic **165**, the buffers **125**, **130** coupled to a central processing unit (CPU) bus **175** by a multiplexer **170**. The cache unit **115** may comprise a first level (L0) cache memory and a second level (L1) cache memory. The L0 and L1 cache memories may be integrated into a single device. Alternatively, the L1 cache memory may be coupled to the processor **105** by a shared bus (not shown).

[0011] The main memory **155** and the cache unit **115** store sequences of instructions and data that are executed by the processor **105**. In one embodiment, the main memory **155** includes a dynamic random access memory (DRAM); however, the main memory **155** may have other configurations as well.

[0012] Additional devices may also be coupled to the memory controller hub **140**, such as multiple main memory devices or a wireless interface **185**. The interface **185** may be a dipole antenna for example, to enable radio frequency communications. The memory controller **145** coordinates data transfer to and from the main memory **155** at the request of the processor **105** and/or I/O devices **160**. Data and/or sequences of instructions, executed by the processor **105**, may be retrieved from the main memory **155**, the cache unit **115**, or other storage devices. A computer system has been described in terms of a single processor; however, multiple processors may be coupled to the processor bus **135**.

[0013] In operation, the TLB **125** maintains a mapping of address translations between linear addresses and corresponding physical addresses. When a memory access type micro-op is loaded into an execution pipeline, it is intercepted by TLB **125**, which performs a lookup to determine whether its internal cache lines contain the physical address corresponding to the linear address of the micro-op. If the address translation is found therein, i.e., if a hit occurs, TLB **125** re-dispatches a micro-op, updated to include the physical address. If a miss occurs, TLB **125** notifies the hardware managed translation lookaside buffer **130**. If a hit occurs, the TLB **130** re-dispatches a micro-op, updated to include the physical address. If a miss occurs, the TLB **130** notifies the PMH **120** that a page walk must be performed to determine the physical address corresponding to the linear address of the micro-op. The page table walk may be performed by hardware, microcode, or other types of software.

[0014] Initially, a micro-op containing, among other things, information relating to the type of instruction and a sequence number is generated and loaded in a pipeline. Although the processor **105** generates a wide variety of micro-ops, only memory access micro-ops, such as pre-fetch, load, and stored, are handled here.

[0015] In one embodiment, software **220**, executed by the processor **105** and stored, for example, within the MMU **110**, begins as shown in **FIG. 2**, by checking both TLBs **125** and **130** for a translation (block **202**). If the translation is in

TLB **125** as determined in diamond **204**, the translation is returned from TLB **125** as indicated a block **214**.

[**0016**] Otherwise, a check at diamond **206** determines whether the translation is in TLB **130**. If so, the translation is returned from TLB **130** as indicated in block **208**. If the translation is not found in either TLB, the translation must be obtained from a page table walk logic as indicated in block **210**. The translation is then put in the TLB **130**, as indicated in block **212**, and returned from TLB **130** as indicated in block **208**.

[**0017**] The sequence of steps shown in **FIG. 2** may also be implemented by hardware or microcode.

[**0018**] TLB **130** is managed primarily by hardware, using any number of known hardware-resident algorithms to decide which translations to put, replace, or invalidate within itself. Software commands to manage TLB **130** may also exist, but may not provide the ability to lock entries into TLB **130** in some embodiments.

[**0019**] TLB **125** is managed entirely via software commands that add and remove entries. TLB **125** can have translations locked into it. The software managed TLB **125** may not use page table walk logic **165**. The page table walk logic is only handled by the TLB **130**.

[**0020**] In one implementation, TLB **130** may be a 128-entry, 4-way set associative cache. TLB **125** may be an 8-entry, fully-associative cache in one embodiment of the present invention. The TLBs **125** and **130** may also handle different ranges of page sizes.

[**0021**] When a request for translation is passed to the TLBs **125** and **130**, both TLBs may be consulted in parallel or sequentially. TLB **125**, the software managed TLB, takes precedence in one embodiment. If the requested translation is not found in either TLB, then the hardware managed TLB **130** sends a request to the page table walk logic **165**, caches the result according to its hardware resident replacement policies, and returns the translation to the requester. With a processor **105**, having elevated security modes, managing the software managed TLB **125** may be restricted to that mode, while less privileged modes may be allowed to manage the hardware managed TLB **130**.

[**0022**] In some embodiments of the present invention, the complexity inherent in locking translations in hardware managed TLBs may be avoided. Determining whether there was a place to lock a particular translation in a hardware managed TLB often involves detailed knowledge of the TLB architecture and detailed tracking of the entries that were locked. The architecture of hardware managed TLBs may also limit the kind of entries that can be locked into hardware managed TLBS. For example, only 4-kilobyte pages may be locked in some cases.

[**0023**] In some embodiments of the present invention, the hardware complexity issues may be lessened by using two translation lookaside buffers, coupled together by a simple mechanism. Thus, the software's job of locking translations may be made simpler, in some embodiments, because it has a more flexible translation lookaside buffer to manage without having to be concerned about interaction with the hardware managed translation lookaside buffer.

[**0024**] While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

What is claimed is:

1. A method comprising:

providing two translation lookaside buffers for one microprocessor.

2. The method of claim 1 including providing a software and a hardware managed translation lookaside buffer for one microprocessor.

3. The method of claim 2 including providing a page table walk logic handled by said hardware managed translation lookaside buffer.

4. The method of claim 3 wherein, when a request for a translation from a linear to physical address is received, checking the software managed translation lookaside buffer to determine whether or not the translation is resident therein before checking said hardware managed translation lookaside buffer.

5. The method of claim 4 including checking said hardware managed translation lookaside buffer for said translation if said translation is not in said software managed translation lookaside buffer.

6. The method of claim 5 including, if said translation is not in said hardware managed translation lookaside buffer, obtaining the translation from a page table walk logic.

7. The method of claim 6 including returning the translation from the page table walk logic to said hardware managed translation lookaside buffer.

8. The method of claim 2 including locking translations in said software managed translation lookaside buffer.

9. The method of claim 2 including managing the software managed translation lookaside buffer in a restricted mode.

10. An article storing instructions that, if executed, enable a processor-based system to:

search in a software managed translation lookaside buffer for a translation from a linear to a physical address and, if the translation is not resident in said software managed translation lookaside buffer, look for said translation in a hardware managed translation lookaside buffer.

11. The article of claim 10 further storing instructions that, if executed, enable a processor-based system to obtain translations from a table walk logic exclusively via a hardware managed translation lookaside buffer instead of said software managed translation lookaside buffer.

12. The article of claim 10 further storing instructions that, if executed, enable a processor-based system to lock translations in said software managed translation lookaside buffer.

13. The article of claim 10 further storing instructions that, if executed, enable a processor-based system to manage the software managed translation lookaside buffer in a restricted mode.

14. The article of claim 10 further storing instructions that, if executed, enable a processor-based system to return a translation from a page table walk logic to said hardware managed translation lookaside buffer.

15. A system comprising:

a processor comprising a pair of translation lookaside buffers, only one of said buffers having page table walk logic; and

a wireless interface coupled to said processor.

16. The system of claim 15 wherein said wireless interface includes a dipole antenna.

17. The system of claim 15 wherein one of said buffers is a software managed translation lookaside buffer and the other one is a hardware managed translation lookaside buffer that is coupled to a page table walk logic.

18. The system of claim 17 wherein said software managed translation lookaside buffer is consulted first and only if a translation is not in said software managed translation lookaside buffer is said hardware managed translation lookaside buffer checked for the translation.

19. The system of claim 18 including a page table walk logic coupled exclusively to said hardware managed translation lookaside buffer.

20. The system of claim 19 wherein the software managed translation lookaside buffer to determine whether or not the translation is resident therein before said hardware managed translation lookaside buffer is checked for said translation.

21. The system of claim 20 wherein said page table walk logic to return a translation from the page table walk logic to said hardware managed translation lookaside buffer.

22. The system of claim 17, said processor to lock translations exclusively in said software managed translation lookaside buffer.

23. The system of claim 17 wherein said software managed translation lookaside buffer is managed in a restricted mode.

* * * * *