UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 95/001,548 | 02/17/2011 | 6910205 | 13557.112021 | 1709 |

25226 7590 08/19/2011
MORRISON & FOERSTER LLP
755 PAGE MILL RD
PALO ALTO, CA 94304-1018

| EXAMINER | |
|---|---|
| KISS, ERIC B | |
| **ART UNIT** | **PAPER NUMBER** |
| 3992 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 08/19/2011 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patents and Trademark Office
P.O.Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**DO NOT USE IN PALM PRINTER**

THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS
KING & SPALDING
1180 PEACHTREE STREET, NE
ATLANTA, GA 30309-3521

Date:



**Transmittal of Communication to Third Party Requester**
**Inter Partes Reexamination**

REEXAMINATION CONTROL NO. : 95001548
PATENT NO. : 6910205
TECHNOLOGY CENTER : 3999
ART UNIT : 3992

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified Reexamination proceeding. 37 CFR 1.903.

Prior to the filing of a Notice of Appeal, each time the patent owner responds to this communication, the third party requester of the inter partes reexamination may once file written comments within a period of 30 days from the date of service of the patent owner's response. This 30-day time period is statutory (35 U.S.C. 314(b)(2)), and, as such, it cannot be extended. See also 37 CFR 1.947.

If an ex parte reexamination has been merged with the inter partes reexamination, no responsive submission by any ex parte third party requester is permitted.

All correspondence relating to this inter partes reexamination proceeding should be directed to the Central Reexamination Unit at the mail, FAX, or hand-carry addresses given at the end of the communication enclosed with this transmittal.

| Transmittal of Communication to Third Party Requester Inter Partes Reexamination | Control No. | Patent Under Reexamination | |
|---|---|---|---|
| | 95/001,548 | 6910205 | |
| | Examiner | Art Unit | |
| | ERIC B. KISS | 3992 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --*

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above-identified reexamination proceeding. 37 CFR 1.903.

Prior to the filing of a Notice of Appeal, each time the patent owner responds to this communication, the third party requester of the *inter partes* reexamination may once file written comments within a period of 30 days from the date of service of the patent owner's response. This 30-day time period is statutory (35 U.S.C. 314(b)(2)), and, as such, it cannot be extended. See also 37 CFR 1.947.

If an *ex parte* reexamination has been merged with the *inter partes* reexamination, no responsive submission by any *ex parte* third party requester is permitted.

**All correspondence** relating to this inter partes reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of the communication enclosed with this transmittal.

| *OFFICE ACTION IN* INTER PARTES *REEXAMINATION* | Control No. | Patent Under Reexamination | |
|---|---|---|---|
| | 95/001,548 | 6910205 | |
| | Examiner | Art Unit | |
| | ERIC B. KISS | 3992 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --*

Responsive to the communication(s) filed by:
Patent Owner on _____
Third Party(ies) on _____

**RESPONSE TIMES ARE SET TO EXPIRE AS FOLLOWS:**

*For Patent Owner's Response*:
2 MONTH(S) from the mailing date of this action. 37 CFR 1.945. EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.956.
*For Third Party Requester's Comments on the Patent Owner Response:*
30 DAYS from the date of service of any patent owner's response. 37 CFR 1.947. NO EXTENSIONS OF TIME ARE PERMITTED. 35 U.S.C. 314(b)(2).

**All correspondence** relating to this inter partes reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of this Office action.

This action is not an Action Closing Prosecution under 37 CFR 1.949, nor is it a Right of Appeal Notice under 37 CFR 1.953.

**PART I. THE FOLLOWING ATTACHMENT(S) ARE PART OF THIS ACTION:**

1. ☐ Notice of References Cited by Examiner, PTO-892
2. ☒ Information Disclosure Citation, PTO/SB/08
3. ☐ _____

**PART II. SUMMARY OF ACTION:**

1a. ☒ Claims 1-4 and 8 are subject to reexamination.
1b. ☒ Claims 5-7 and 9-14 are not subject to reexamination.
2. ☐ Claims _____ have been canceled.
3. ☐ Claims _____ are confirmed. [Unamended patent claims]
4. ☐ Claims _____ are patentable. [Amended or new claims]
5. ☒ Claims 1-4 and 8 are rejected.
6. ☐ Claims _____ are objected to.
7. ☐ The drawings filed on _____ ☐ are acceptable ☐ are not acceptable.
8. ☐ The drawing correction request filed on _____ is: ☐ approved. ☐ disapproved.
9. ☐ Acknowledgment is made of the claim for priority under 35 U.S.C. 119 (a)-(d). The certified copy has:
☐ been received. ☐ not been received. ☐ been filed in Application/Control No 95001548.
10. ☐ Other _____

## DETAILED ACTION

Claims 1-4 and 8 of United States Patent 6,910,205 are under reexamination.

## I. REFERENCES CITED IN THE REQUEST

The request cites the following patents and printed publications:

1. L. Peter Deutsch et al., *Efficient Implementation of the Smalltalk-80 System*, Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, pp. 297-302, 1984 (hereinafter, "Deutsch");

2. David Wakeling, *A Throw-Away Compiler for a Lazy Functional Language*, Fuji International Workshop on Functional and Logic Programming, pp. 287-300, 1995 (hereinafter, "Wakeling");

3. Brian T. Lewis et al., *Clarity MCode: A Retargetable Intermediate Representation for Compilation*, ACM, IR '95, 1/95, San Francisco, California, USA, pp. 119-128, 1995 (hereinafter, "Lewis");

4. Paul Tarau et al., *The Power of Partial Translation: An Experiment with the C-Ification of Binary Prolog*, ACM Symposium on Applied Computing, pp. 152-156, 1995 (hereinafter, "Tarau");

5. Frank Yellin, *The JIT Compiler API*, The JIT Compiler API, October 4, 1996, pp. 1-23 (hereinafter, "Yellin");

6. U.S. Patent 6,081,665 (Nilsen et al.);

7. U.S. Patent 5,842,017 (Hookway et al.);

8. Peter Magnusson, *Partial Translation*, Swedish Institute of Computer Science Technical Report (T93.5), Oct. 1993 (hereinafter, "Magnusson"); and

9. U.S. Patent 5,768,593 (Walters et al.).

## II. INFORMATION DISCLOSURE STATEMENT

Consideration by the examiner of the information submitted in an information disclosure statement means that the examiner will consider the documents in the same manner the party filing the information citation has explained the content and relevance of the information. The initials of the examiner placed adjacent to the citations on the form PTO/SB/08A and 08B or its equivalent, without an indication to the contrary in the record, do not signify that the information has been considered by the examiner any further than to the extent noted above. See MPEP §§ 609.05(b), 2256, and 2656.

The information disclosure statement filed on April 27, 2011, has been given due consideration. Documents with fail to constitute prior art patents or printed publications have been lined through on the Form PTO/SB/08 so as not to be published on the reexamination certificate, but have been considered to the extent noted above. Documents that have previously cited and considered have also been lined through.

## III. REJECTIONS PROPOSED IN THE REQUEST

Within the scope of this reexamination proceeding, the request proposes the following rejections (Request for *Inter Partes* Reexamination, pp. 37-40):

1. Proposed: Claims 1-4 and 8 are unpatentable under 35 U.S.C. § 102(b) as being anticipated by *Tarau*, (*id.* at 37 (citing the claim chart in Exhibit 14));

2. Proposed: Claims 1-4 and 8 are unpatentable under 35 U.S.C. § 102(b) as being anticipated by *Magnusson*, (*id.* at 38 (citing the claim chart in Exhibit 18));

3. Proposed: Claims 1-4 and 8 are unpatentable under 35 U.S.C. § 103(a) as being obvious over the Walters '593 patent in view of *Tarau*, (*id.* at 39 (citing the claim chart in Exhibit 19)); and

4. Proposed: Claims 1-4 and 8 are unpatentable under 35 U.S.C. § 103(a) as being obvious over the Walters '593 patent in view of Magnusson, (*id.* at 40 (citing the claim chart in Exhibit 19)).

## IV. CLAIM REJECTIONS §§ 102(B) AND 103(A)

### A. Relevant statutes

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

### B. Tarau

**The proposed rejection of claim 8 under 35 U.S.C. 102(b) as being anticipated by *Tarau* is adopted.**

The following claim chart shows the correspondence between the '205 patent claim and pertinent teachings of Tarau.

| **'205 Patent** | **Tarau** |
|---|---|
| 8. In a computer system, a method for increasing the execution speed of virtual machine instructions, the method comprising: | "We describe a new language translation framework (partial translation) and the implementation of one of its instances: the C-ification of Binary Prolog.<br><br>"Partial C-ification is a translation framework which compiles sequences of emulator instructions down to native code (on top of a C compiler). In the case of logic programming languages, their complex control structure, some large instructions, and the management of the symbol table are left to the emulator while the native code chunks will deal with relatively long sequences of simple instructions.<br><br>"The technique can be seen as an automatic specialization with respect to a given program of the traditional instruction folding techniques used to speed-up emulators." Tarau at 152. |
| inputting virtual machine instructions for a function; | *Prolog code (source code) is converted to interpreted WAM instructions (virtual machine instructions for the Warren Abstract Machine or the simplified BinWAM).*<br><br>"By a straightforward modification of the emulator written in C we replace some contiguous sequences of interpreted WAM instructions by lightweight C-routines (C-chunks) which get called from a slightly differently compiled version of the emulator, linked together with them." Tarau at 153.<br><br>"We refer the reader to [11] for a more formal description of the binarization transformation and to [4] for its relationship to the Warren Abstract Machine (WAM) - the abstract engine used as a basis of most modern Prolog implementations. With 'AND-continuations' reified as an extra argument of each goal, we manage to have only one call on the right side (b/1 in the previous example). This allows the use of a simplified, yet powerful reduced instruction set WAM-like execution model (BinWAM), as described in [8, 9, 10, 12]. The main difference at code generation level between the WAM and the BinWAM is the call-return mechanism for the former versus the no-return, continuation passing execution for the latter." Tarau at 153. |

| | |
|---|---|
| | *Tarau further discloses an example Prolog clause chosen for translation wherein the clause may be considered a function (defined as, "A software routine (also called a subroutine, procedure, member, and method)"; see col. 4, lines 18-19 of the '205 patent):*<br><br>"Prolog version<br><br>"We have chosen the following clause containing some duplicate symbols and numbers:<br><br>treesize(tree(Left,Right),S0,S) :-<br>add(S0,i ,S1),<br>treesize(Left,S1,S2),<br>treesize(Right,S2,S).<br><br>"Binarized version<br><br>"After binarization, our example becomes:<br><br>treesize(tree(Left,Right),S0,S,C) :-<br>add(S0,1,S1,<br>treesize(Left,S1,S2,<br>treesize(Right,S2,S,C))).<br><br>"Hence, every clause gets an extra argument that carries the continuation, i.e., the part of the program that is still to be executed. Before add/4 can be started, its continuation consisting of the nested treesize/4 terms is created on the heap.<br><br>"It is precisely the creation of this continuation, and the preparation of the arguments for add/4 that will be subject to C-ification." Tarau at 153.<br><br>"From this binary form we obtain the following WAM-code: |

| | |
|---|---|
| | ```<br>[1] clause_? treesize 4<br>[2] firstarg_? tree/2 11<br>[3] get_structure tree/2 var(1)  % tree/2<br>[4] unify_variable  var(5)  % Left<br>[5] unify_variable  var(6)  % Right<br>[6] get_variable arg(2) var(1)  % S0<br>[7] c_chunk_begin 15 var(7)<br>[8] put_structure treesize/4 var(8) % make continuation<br>[9] write_value  var(5)  % Left<br>[10] write_variable  var(5)  % S1<br>[11] write_variable  var(9)  % S2<br>[12] write_variable  var(10)<br>[13] push_structure treesize/4 var(10)<br>[14] write_value  var(6)  % Right<br>[15] write_value  var(9)  % S2<br>[16] write_value  var(3)  % S<br>[17] write_value  var(4)  % C<br>[18] put_constant arg(2) 1  % 1<br>[19] put_value arg(3) var(5)  % S1<br>[20] put_value arg(4) var(8)  % move continuation<br>[21] c_chunk_end 15 var(7)  %  to its register<br>[22] execute_? add 4<br>```<br><br>Tarau at 153. |
| compiling a portion of the function into at least one native machine instruction so that the function includes both virtual and native machine instruction; | "Note the presence of c_chunk_begin and c_chunk_end in the WAM-code. The C-routine will be generated from the sequence of instructions between them." Tarau at 153.<br><br>"To be able to call a C-routine from the emulator we have to know its address. Unfortunately, the linker is the only one that knows the eventual address of a C-routine. A simple and fully portable technique to plug the address of a C-routine into the byte code is to C-ify the byte-code of the emulator into a huge C array of records, containing the symbolic address of the C-chunks. After compilation, and linking with the emulator, the linker will automatically resolve all the missing addresses and generate warnings for the missing C-routines. The result will be a stand alone Prolog application (see Figure 2)." Tarau at 153.<br><br>*The C-routine is compiled into native machine code:*<br><br>"It is interesting to take a look at the actual assembly (Sparc, Solaris 2.x, gcc -O2) listing, which shows clearly that our objective to have high quality code has been attained with minimal effort (H is in %o0, regs is in %ol, and P is in %o2).<br><br>"EMULATOR DATA STRUCTURE:<br>....<br>.word xx_399 |

| | |
|---|---|
| | .byte 6<br>.byte 0<br>.half 4<br>.word .LLC993<br>...<br><br>C_CHUNK :<br>xx_399:<br>st %o0, [%o1+32]<br>ld [%o2] ,%g2<br>st %,g2, [%o0]<br>ld [%o1+20] ,%g3<br>.....<br>ld [%o1+32] ,%g2<br>st %g3,[%ol+12]<br>retl<br>st %g2,[%o1+16]<br><br>"It can be seen that the mapping to a sequence of load-store instructions with precomputed offsets gives efficient code, which can be reorganized quite freely by super-scalar schedulers." Tarau at 154. |
| representing said at least one native machine instruction with a new virtual machine instruction that is executed after the compiling of the function. | *A C-call is generated in the P-code section of the emulated program. The final argument list of the C-call (containing the resolved address of the C-routine) is built once by the loader of the emulator. The C-call is a new virtual machine instruction that is executed (by the emulator) after compilation of the function:*<br><br>"EMULATOR DATA STRUCTURE:<br>.....<br>(63,7,0, (void *)xx_399},<br>(6,0,4,"treesize"}, /* argument of xx_399 */<br>.....<br><br>"All Prolog symbols are internalized by the emulator. The Prolog symbols that are needed by the C-chunk are put in the argument list of the C-call on a constant offset from the current P-pointer (here only one argument). So, the chunk can access them with a single load operation at no extra cost, and an optimizing compiler can take full advantage of this information and generate the most efficient code for a particular sequence. All this is similar to argument passing in threaded code [1]. |

| | |
|---|---|
| | "There is no need for the C-chunk to directly access the Prolog symbol table. The final argument list of the C-call is built once by the loader of the emulator." Tarau at 154. |

**The proposed rejection of claims 1-4 under 35 U.S.C. 102(b) as being anticipated by *Tarau* is not adopted.**

Although Tarau does disclose generating a new virtual machine instruction that represents or references one or more native instructions that can be executed instead of said first virtual machine instruction (see Tarau as applied above in the rejection of claim 8), the generation does not take place at runtime as required by claims 1-4. Instead, the instructions are generated and compiled and linked with the emulator code to produce a new executable emulator. See Tarau at 152.

### C. Magnusson

**The proposed rejection of claims 1-4 and 8 under 35 U.S.C. 102(b) as being anticipated by Magnusson is adopted.**

The following claim chart shows the correspondence between the '205 patent claims and pertinent teachings of Magnusson.

| '205 Patent | Magnusson |
|---|---|
| 1. In a computer system, a method for increasing the execution speed of virtual machine instructions at runtime, the method comprising: | "Traditional simulation of a target architecture by interpreting object code can be improved by translating the object code to an intermediate format. This approach is called interpretive translation. Despite a substantial performance improvement over traditional interpretation, a large part of the overhead is unnecessary. An alternative approach is block translation, where one or more simulated instructions are translated to directly executable code. This approach has several drawbacks.<br><br>"We discuss the problems with block translation, analyse the overhead of interpretive translation, and describe a hybrid approach--partial translation--that combines the benefits of both approaches. Partial translation implements an intermediate format that supports the addition of run-time generated code whenever appropriate. The performance limit (slowdown) of interpretive translation is around 15, and real implementations have achieved 20-30. Partial translation will perform considerably better. Finally, we present results from an aggressive implementation of interpretive translation, and results from a proof-of-concept implementation of partial translation." Magnusson at 3. |
| receiving first virtual machine instruction; | "Figure 4 illustrates how of [sic] interpreting an intermediate format can be combined with direct execution of generated code. Assume that we wish to translate a sequence M88100 instructions to native SPARC code. The intermediate code is in a 64-bit format, where the first 32 bits points to the code that simulates the corresponding instruction. The second 32 bits contain parameters for the instruction. Each 88k instruction is translated to this format.<br><br>"When execution of the program reaches the first instruction in the block, the service routine being jumped to is actually a run-time generated block of SPARC code. When this block has completed, |

| | |
|---|---|
| | it will dispatch the first instruction after the block. 'Dispatch' here means reading the intermediate format and jumping to the pointer containing the first 32 bits." Magnusson at 8.<br><br>*The "intermediate code" instructions of Magnusson are machine instructions for a software emulated M88100 microprocessor and correspond to the claimed "virtual machine instructions" (consistent with the explicit definition in col. 4, lines 10-12 of the '205 patent).*<br><br>*The intermediate code instructions are read, i.e., received, at run-time by the partial translation system and interpreted or translated.*<br><br>*The first instruction in a block of intermediate code chosen to be translated corresponds to the claimed "first virtual machine instruction." Examples include the "r7 = [r5]" instruction on p. 8 of Magnusson and the "or.u r31,r0,16416" instruction on p. 13 of Magnusson.* |
| generating, at runtime, a new virtual machine instruction that represents or references one or more native instructions that can be executed instead of said first virtual machine instruction; | "There are two ways of combining the threaded code model with block translation. Either the internal format includes a direct pointer to the compiled block, or we introduce a new instruction, TRANSLATED. This instruction takes as a parameter a pointer to the translated block, and handles generic entry/exit issues. In the former approach, the previous instruction dispatches the decoded block directly (by jumping to it), and entry/exit code needs to be compiled into every block. Both approaches have the advantage that re-entry needs no special checks." Magnusson at 9. |
| and executing said new virtual machine instruction instead of said first virtual machine instruction. | "When execution of the program reaches the first instruction in the block, the service routine being jumped to is actually a run-time generated block of SPARC code." Magnusson at 8. |
| 2. The method of claim 1, further comprising overwriting a selected virtual machine instruction with a new virtual machine instruction, the new virtual machine instruction specifying execution of the at least one native machine | "There are two ways of combining the threaded code model with block translation. Either the internal format includes a direct pointer to the compiled block, or we introduce a new instruction, TRANSLATED. This instruction takes as a parameter a pointer to the translated block, and handles generic entry/exit issues. In the former approach, the previous instruction dispatches the decoded block directly (by jumping to it), and entry/exit code needs to be compiled into every block. Both approaches have the advantage that re-entry needs no special checks." Magnusson at 9. |

| | |
|---|---|
| instruction. | "When execution of the program reaches the first instruction in the block, the service routine being jumped to is actually a run-time generated block of SPARC code." Magnusson at 8. |
| 3. The method of claim 2, wherein the new virtual machine instruction includes a pointer to the at least one native machine instruction. | "Either the internal format includes a direct pointer to the compiled block, or we introduce a new instruction, TRANSLATED. This instruction takes as a parameter a pointer to the translated block, and handles generic entry/exit issues." Magnusson at 9. |
| 4. The method of claim 2, further comprising storing the selected virtual machine instruction before it is overwritten. | "During translation of the block, we estimate that the block will take 8 cycles if executed (in this example, one cycle per instruction except memory accesses which take two). We first check if there are 8 or more cycles left, otherwise we abort to exit 0. Exit 0 dispatches the first instruction of the block as if it had been an interpreted instruction. It does this by loading the corresponding rOP value and branching to the interpretation code. Of course, it knows exactly where to branch to. The overhead of unsuccessfully attempting to dispatch the block is 5 instructions, of which two are branches and none are memory accesses." Magnusson at 11. |
| 8. In a computer system, a method for increasing the execution speed of virtual machine instructions, the method comprising: | "Traditional simulation of a target architecture by interpreting object code can be improved by translating the object code to an intermediate format. This approach is called interpretive translation. Despite a substantial performance improvement over traditional interpretation, a large part of the overhead is unnecessary. An alternative approach is block translation, where one or more simulated instructions are translated to directly executable code. This approach has several drawbacks.<br><br>"We discuss the problems with block translation, analyse the overhead of interpretive translation, and describe a hybrid approach--partial translation--that combines the benefits of both approaches. Partial translation implements an intermediate format that supports the addition of run-time generated code whenever appropriate. The performance limit (slowdown) of interpretive translation is around 15, and real implementations have achieved 20-30. Partial translation will perform considerably better. Finally, we present results from an aggressive implementation of interpretive translation, and results from a proof-of-concept |

| | |
|---|---|
| | implementation of partial translation." Magnusson at 3. |
| inputting virtual machine instructions for a function; | "Figure 4 illustrates how of [sic] interpreting an intermediate format can be combined with direct execution of generated code. Assume that we wish to translate a sequence M88100 instructions to native SPARC code. The intermediate code is in a 64-bit format, where the first 32 bits points to the code that simulates the corresponding instruction. The second 32 bits contain parameters for the instruction. Each 88k instruction is translated to this format.<br><br>"When execution of the program reaches the first instruction in the block, the service routine being jumped to is actually a run-time generated block of SPARC code. When this block has completed, it will dispatch the first instruction after the block. 'Dispatch' here means reading the intermediate format and jumping to the pointer containing the first 32 bits." Magnusson at 8.<br><br>*The "intermediate code" instructions of Magnusson are machine instructions for a software emulated M88100 microprocessor and correspond to the claimed "virtual machine instructions" (consistent with the explicit definition in col. 4, lines 10-12 of the '205 patent).*<br><br>*The intermediate code instructions are read, i.e., inputted, at run-time by the partial translation system and interpreted or translated.*<br><br>*Magnusson further discloses an example block of intermediate code chosen for translation wherein the block comprises a portion of a function (defined as, "A software routine (also called a subroutine, procedure, member, and method)"; see col. 4, lines 18-19 of the '205 patent):*<br><br>"Consider the following example code:<br><br>label: r7 = [r5] ; read next word from source<br>[r9] = r7 ; store it to destination<br>r5 = r5 + 4 ; increment source pointer<br>r9 = r9 + 4 ; increment destination pointer<br>cmp r13,r9 ; finished?<br>br ne,<label> ; if not, continue<br><br>"The example code is written in pseudo-assembler, since the discussion is processor independent. The example code above might occur inside a block copy routine. Each iteration would take 120-180 instructions in a simulator based on interpretive |

| | |
|---|---|
| | translation, i.e. assuming a slowdown of 20-30." Magnusson at 8-9. |
| compiling a portion of the function into at least one native machine instruction so that the function includes both virtual and native machine instruction; | *In the block copy routine example, Magnusson at 8-12, the translated block of code is native machine code, but in the event of certain runtime errors, the simulator can default to dispatching and interpreting the intermediate code instead, and therefore, the function includes both virtual and native machine instructions:*<br><br>"The third check done upon entry is to assert that the instruction pipeline looks the same as the translation routine thought it would. The code generated in the translated block assumes a particular content in the pipeline. This is not likely to change, but this check is necessary for sake of correctness. It could be optimised away by dispatching this block of code directly only by other translated blocks, and checking the instruction pipeline only in the TRANSLATED pseudo-instruction. If this check fails, then the instructions are interpreted instead, so we branch to exit 0." Magnusson at 11. |
| representing said at least one native machine instruction with a new virtual machine instruction that is executed after the compiling of the function. | "There are two ways of combining the threaded code model with block translation. Either the internal format includes a direct pointer to the compiled block, or we introduce a new instruction, TRANSLATED. This instruction takes as a parameter a pointer to the translated block, and handles generic entry/exit issues. In the former approach, the previous instruction dispatches the decoded block directly (by jumping to it), and entry/exit code needs to be compiled into every block. Both approaches have the advantage that re-entry needs no special checks." Magnusson at 9.<br><br>"When execution of the program reaches the first instruction in the block, the service routine being jumped to is actually a run-time generated block of SPARC code." Magnusson at 8. |

**D. The Walters '593 patent in view of Tarau**

**The proposed rejection of claims 1-4 and 8 under 35 U.S.C. 103(a) as being obvious over the Walters '593 patent in view of *Tarau* is not adopted.**

The Supreme Court in KSR noted that the analysis supporting a rejection under 35 U.S.C. 103 should be made explicit. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007). The Court quoting *In re Kahn*, 441 F.3d 977, 988, 78 USPQ2d 1329, 1336 (Fed. Cir. 2006), stated that "[R]ejections on obviousness cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness.'" *Id.*

The request relies exclusively on the Claim Chart attached as Exhibit 19 to provide the factual basis supporting the proposed rejection, (Request at 39). However, the claim chart contains conclusory statements that certain claim features would have been obvious "based on the separate disclosures of each of the following prior art references, which individually disclose this limitation in the context of the subject matter of the Walters '593 patent," without providing a legally tenable rationale for combining the cited teachings under § 103. (Exhibit 19 at 4, 6, and 16-17 (citing the claim charts of Exhibits 11 through 18).)

**E. The Walters '593 patent in view of Magnusson**

**The proposed rejection of claims 1-4 and 8 under 35 U.S.C. 103(a) as being obvious over the Walters '593 patent in view of *Magnusson* is not adopted.**

The request relies exclusively on the Claim Chart attached as Exhibit 19 to provide the factual basis supporting the proposed rejection, (Request at 40). However, the claim chart contains conclusory statements that certain claim features would have been obvious "based on the separate disclosures of each of the following prior art references, which individually disclose this limitation in the context of the subject matter of the Walters '593 patent," without providing

a legally tenable rationale for combining the cited teachings under § 103. (Exhibit 19 at 4, 6, and 16-17 (citing the claim charts of Exhibits 11 through 18).)

## V. CONCLUSION

Because the requester did not request reexamination of claims 5-7 and 9-14 and did not assert the existence of a substantial new question of patentability (SNQ) for such claims, claims 5-7 and 9-14 will not be reexamined. *See Sony Computer Entertainment America Inc. v. Dudas*, 85 USPQ2d 1594 (E.D. Va 2006) ("[W]hile the PTO in its discretion may review claims for which *inter partes* review was not requested, nothing in the statute compels it to do so.").

The patent owner is reminded of the continuing responsibility under 37 CFR 1.565(a) to apprise the Office of any litigation activity, or other prior or concurrent proceeding, involving Patent No. 6,910,205 throughout the course of this reexamination proceeding. The third party requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP §§ 2207, 2282 and 2286.

Any paper filed with the USPTO, *i.e.*, any submission made, by either the patent owner or the third party requester must be served on every other party in the reexamination proceeding, including any other third party requester that is part of the proceeding due to merger of the reexamination proceedings. As proof of service, the party submitting the paper to the Office must attach a Certificate of Service to the paper, which sets forth the name and address of the party served and the method of service. Papers filed without the required Certificate of Service may be denied consideration. 37 CFR 1.903; MPEP 2666.06.

In order to ensure full consideration of any amendments, affidavits or declarations, or other documents as evidence of patentability, such documents must be submitted in response to this Office action. Submissions after the next Office action, which is intended to be an Action

Closing Prosecution (ACP), will be governed by 37 CFR 1.116(b) and (d), which will be strictly enforced.

**All** correspondence relating to this *inter partes* reexamination proceeding should be directed:

By Mail to: Mail Stop *Inter Partes* Reexam
Attn: Central Reexamination Unit
Commissioner for Patents
United States Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

By FAX to: (571) 273-9900
Central Reexamination Unit

By EFS: Registered users may submit via the electronic filing system EFS-WEB, at https://efs.uspto.gov/efile/myportal/efs-registered

By hand: Customer Service Window
Randolph Building
401 Dulany St.
Alexandria, VA 22314

Any inquiry concerning this communication or earlier communications from the examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

/Eric B. Kiss/
Primary Examiner, Art Unit 3992

Conferees:

/Mary Steelman/
Primary Examiner CRU 3992