EXHIBIT I

14.100 220 GP

2315

Express Mail Label # EP460963872US Date of Deposit 8/29/95 #13

I hereby certify that this paper is being deposited with the United States Postal Service using "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: "Commissioner of Patents and Trademarks, Washington, DC 20231."

Signed [illegible] Stuart T. Auvinen, Reg. No. 36,435 Date Signed 8/29/95

10/13

EXPEDITED PROCEDURE - AF
PATENT
RM-1

9/12/95

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of	
Blomgren et al.	Examiner: V. Vu
Serial No. 08/179,926	Group Art Unit: 2315
Filed: 1/11/94	
For: Dual-Instruction-Set Architecture CPU with Hidden Software Emulation Mode	

VU 9-13-95

BRIEF ON APPEAL UNDER 37 C.F.R. § 1.192

Hon. Commissioner of Patents and Trademarks
Box AF
Washington, DC 20231

Sir:

This is an appeal from Examiner's final rejection of claims 1-20, contained in the third office action mailed 4/5/95. The notice of appeal was filed, together with an authorization to charge the fee to my deposit account, on 7/3/95.

070 DF 09/06/95 08179926 1 220 140.00 CK

This appeal brief is being filed within 2 months of the notice of appeal. Attached is a check for the fee under 37 C.F.R. § 1.17(f), (fee code 220) for filing the Brief on Appeal of $ **140.00**. Small entity status has previously been established in this application.

X The commissioner is hereby authorized to charge payment of any patent application processing fees under 37 C.F.R. § 1.17 associated with this communication or credit any overpayment to deposit account No. **01-2950**.

STATUS OF THE CLAIMS

Claims 1-20 are pending and on appeal.

STATUS OF AMENDMENTS AFTER FINAL REJECTION

An amendment after final rejection was filed on 5/22/95 and was entered by the Examiner as noted in his Advisory Action of 6/6/95. The Examiner indicated that the amendment did not overcome the rejections.

SUMMARY OF THE INVENTION

The invention is a dual-instruction-set processor (CPU). A dual-instruction-set CPU is able to execute x86 CISC (complex instruction set computer) code or PowerPC™ RISC (reduced instruction set computer) code. (abstract) While current CPUs typically execute just one instruction set, the invention allows execution of two instruction sets. Additional hardware is minimized by sharing a single execution unit but having two instruction decoders. Hardware cost and complexity is further minimized as only a *subset* of the x86 instruction set is decoded while the entire PowerPC™ instruction set is decoded.

Claim 1 Summary

Claim 1 recites a first and a second instruction decoder (36) for decoding instructions from a first and a second instruction set. Only a subset of instructions from the second instruction set is decoded. A select means (46) selects either the decoded instruction from the first decoder or from the second decoder. An execute means (48) executes the decoded instruction selected by the select means. Thus the execute means can execute both first and second instructions provided by the select means.

Technology Tutorial — Figure 2, Appendix 2

Appellant 's Figure 2 (reproduced in Appendix 2) shows that instructions are fetched and supplied to a RISC instruction decoder (RISC ID 36) and a CISC instruction decoder (CISC ID 36). Either the decoded RISC instruction or the decoded CISC instruction is selected by mux 46 and output to execute unit 48. Note that both RISC and CISC instructions are executed by the execute unit 48. While the entire RISC instruction set is decoded and executed, only a subset of the CISC instruction set are decoded and executed.

A mode register 38 contains a RISC/CISC bit (R/C) which causes mux 46 to select either the decoded RISC instruction or the decoded CISC instruction. A miss from translation-lookaside buffer (TLB) 52 or an undecodable/unknown CISC instruction 40 can cause mode control 42 to switch mode register 38 to RISC mode from CISC mode.

Subject of Dependent Claims 2-13

Claim 2 recites that the single instruction fetch buffer 32 feeds instructions to both the RISC and CISC decoders 36. Rather than duplicate the entire pipeline, the instruction fetcher, execute unit, and TLB are shared among both instruction sets. Only the CISC decode logic is added. The mode register is claimed in claim 3; the mode control in claim 4. An undecodable instruction switching the mode register is the subject of claim 5. A TLB miss switching the mode control and thus the instruction set decoded is the subject of claim 6. In claim 7 a handler routine of first instructions is executed when the mode control is signaled by an undecodable second instruction or a TLB miss.

In Claim 8 an exception from the execute unit causes the mode control to change to decoding the first instruction set. Claim 9 recites that all references to main memory from the second instruction set are translated by the TLB, but claim 10 reveals that only first instructions can load the TLB. In claim 11 extended first instructions modify TLB entries. In claim 12 first instructions are decoded following a reset, while claim 12 recites that extended instructions are decoded only when the mode control is signaled to change to the first instruction decoding, or immediately following a reset.

What are Separate Instruction Sets ?

Since two separate instruction sets are used, a fetched instruction word could output two different but seemingly valid decoded instructions. The RISC decoder and the CISC decoder could each output a seemingly valid decoded instruction. This is not true for a single instruction set or extensions to a single instruction set. The 386 and 486 instruction sets do not meet claim limitations for separate instruction sets since the 486 set is a mere extension of the 386 set, having most opcodes in common.

Method claims 14-17 Summary

Independent claim 14 is directed to a method for processing instructions from two separate instruction sets. Independent claim 15 is directed to a method for processing instructions from a CISC and a RISC instruction set in which all CISC instructions are executable, either directly by the execute unit or by emulation mode with RISC instructions.

Claim 16 adds that the emulation routine uses RISC instructions and extended instructions, while claim 17 recites that memory references generated by the CISC instructions are translated under control of a translator routine of RISC instructions which load a TLB.

Apparatus Claims 18-20 Summary

Independent claim 18 is directed to a microprocessor for executing RISC and CISC instructions using both RISC and CISC instruction decoders and an enable means to enable one of the instruction decoders.

While the entire RISC instruction set is decoded, only a *subset* of the more complex CISC instruction set is decoded. Dependent claim 19 recites that the undecoded CISC instructions are emulated by an emulation mode, while the mode register indicates RISC, CISC, or emulation mode.

ISSUES

The issue is whether the obviousness rejections are proper. In particular, the Portanova reference was cited by the Examiner as 'suggesting' but 'not explicitly teaching' claim limitations. Appellant has shown that this 'suggestion' does not exist, and cannot possibly exist, because the Portanova reference explicitly teaches away from such a 'suggestion'. Any modification of Portanova using this 'suggestion' destroys the purpose explicitly stated by Portanova.

Under 35 USC § 103, claims 1-5, 14-16, and 18-20 were rejected as obvious over Portanova et al. (US Pat. No. 4,992,934) in view of Onishi (U.S. Patent No. 3,764,988). Claims 6-13 and 17 were rejected under 35 USC § 103 as obvious over Portanova in view of Onishi as set forth for claims 1-5, and further in view of Bullions, III, et al (U.S. Patent No. 4,456,954).

GROUPING OF CLAIMS

The claims do not stand or fall together.

Although the claims each differ in scope from one another, Appellant has grouped several claims together for administrative efficiency.

All claims rely on the basic argument present below for independent claims 1, 14, 15, and 18. Additional arguments are presented for dependent claims 6-13 and 17 which further include a TLB. Other specific arguments are presented at the end of the argument section for other claims. Thus the claims are grouped as:

A.) Claims 1, 2, 14 (with additional arguments for claim 2)
B.) Claims 3, 4, 18, 19 (with additional arguments for claim 19)
C.) Claims 6, 7-13, 17 (with additional arguments for several of claims 7-13)
D.) Claims 5, 15, 16, 20 (with additional arguments for claim 16)

Group A claims the basic structure of the dual-instruction-set decoders and select for the single execute unit, with only a subset of the CISC instruction set decodable. Group B further contains the mode register with the instruction-set-indicating bit. Group C further includes the TLB. Group D specifies that undecodable CISC instructions cause the instruction set decoded to switch.

ARGUMENTS

All claims were rejected using the Portanova reference as showing both RISC and CISC hardware execution. The other references, Onishi and Bullions, execute only one instruction set and are used as secondary references.

Portanova is a RISC processor that emulates or simulates CISC instructions by replacing each CISC instruction with a routine of several RISC instructions. The claimed invention is not an emulator but directly executes in hardware both RISC and CISC instructions.

Portanova is a CISC emulator while the invention is not an emulator but instead executes a subset of CISC instructions directly in hardware. This major difference is brushed aside with such statements as "software and hardware implementations are indeed interchangeable."(third action, page 5, lines 3-4) The new term "hardware emulation" has been coined by the Examiner (second action, page 5, line 11) to further blur the distinction between hardware execution and software emulation. Simple statements that software and hardware are interchangeable belittle the complex art of microprocessor design. Such simple statements fail to teach or suggest the claimed structure of a single RISC and CISC execute unit but separate decoders.

Simulator vs. Actual Hardware Execution — an Analogy

An analogy highlights the absurdity of this position: Airplane pilots learn how to fly a plane in trainers called aircraft simulators. These are large boxes with realistic-appearing controls that the pilot operates. The box may be supported by hydraulics to simulate rocking movements of the airplane. The pilot appears to be flying the real airplane. However, there is one big difference — if the pilot crashes the simulator, he opens the door and walks out. If he crashes the airplane, he's dead.

Portanova is a simulator. The invention is not a simulator. Portanova may appear similar to the invention, even being called a 'dual-instruction-set processor', but Portanova doesn't execute CISC instructions. He emulates them much as the flight simulator emulates aircraft flight though the rocking motion of the box. The invention actually executes instructions much as an actual airplane moves through the air.

The invention actually executes in hardware two entirely *independent* instruction sets. The inventors have realized that only frequently-used instructions from the second instruction set need to be decoded and executed in hardware since the less-frequently-used second instructions can be emulated if needed. Thus only a subset of the second instructions need to be hardware-executed while all of the first instructions are hardware-executed.

Portanova teaches a RISC processor used to emulate any one of several existing CISC architectures. None of the CISC instructions are hardware-executed. A CISC design methodology is disclosed whereby a RISC is designed and fabricated and whereby RISC emulation code is written concurrently with design and fabrication and also subsequent to fabrication. (Abstract)

Identification of Points of Agreement

Appellant believes that both parties agree that Portanova does not explicitly teach hardware execution of both RISC and CISC instruction sets. Portanova emulates all CISC instructions with routines of RISC instructions. Examiner acknowledges that "Portanova explicitly teaches an exemplary system that employs software implementation of CISC emulation in which each guest CISC instruction is emulated by a series of host RISC instructions." (page 4, lines 3-5, third action) Appellant understands this statement to mean that Portanova explicitly teaches only RISC hardware execution and only CISC software emulation.

Identification of Points of Disagreement

Examiner believes that Portanova *suggests* hardware execution of CISC instructions in an otherwise RISC processor. This suggestion appears somewhere in col 29-30. Appellant has been unable to find this suggestion and has requested that the Examiner specifically point out what he is relying on.

Portanova Cannot Suggest CISC Hardware Execution on His RISC CPU

Appellant asserts that the CISC embodiments of col. 29-30 are nothing more than prior-art CISC-only architectures that Portanova can emulate. Indeed, these CISC architectures are labeled "prior-art" on the corresponding Figures 9-12 because they are nothing more than old processors decoding and executing just one instruction set. Portanova clearly is discussing prior-art design approaches:

> "Using current methods, the designing of a CISC could take several different forms. A first approach would be to implement all instructions using single level control.". (col. 29, lines 62-65)

"Using current methods" is a clear statement that what Portanova is discussing is prior art, a "current" method or methodology. Portanova has invented a new method or methodology. Indeed, this new methodology is a "third aspect" of his invention (col 29, line 60). He first discusses current (prior-art for Portanova) methods or design approaches that have been used for Zilog's Z8000, Motorola's 68000, DEC's VAX, and IBM's S/370 (col 29, line 64 to col 30, line 38). He finishes by discussing IBM's S/370, which he states is a "prior art design using an approach..." (col 30, line 29-30). Then he leaves these prior-art methodologies and discusses the "third aspect" of his invention, a new "design method" (col 30, line 39-40, 48).

Cited Section Simply Describes Old CISC Processors

These well-known CISC architectures are presented in the "Design Methodology" section at the end of the Portanova reference to show that Portanova's RISC processor can be used to implement *any* of these well-known CISC architectures:

> The **design method** disclosed herein applies to **any number** of **CISC instruction sets** including MIL-STD-1750, VAX, NEBULA, etc. The approach is to first build a single-level control (hardwired) using RISC design philosophy. In so doing, the designer attempts to maximize execution of the RISC (hardwired) instruction set. (col 30, lines 48-54 emphasis added)

The fact that one RISC processor can implement so many different CISC architectures indicates that CISC-specific hardware is not used. If CISC hardware was used on Portanova's processor, then this CISC hardware has to be different for each of the CISC architectures. A CISC VAX cannot execute Motorola 68000 CISC instructions since they are entirely different instruction sets, with different encoding of opcodes. An instruction decoder for the VAX instruction set could not decode 68000 instructions. *Each* CISC architecture requires a *different* instruction decoder. This decoder is part of the hardware on the processor.

Does Portanova have four different CISC decoders for each of the Z8000, 68000, VAX, and System/370 CISC instruction sets ? Of course not. Each CISC instruction set is decoded by selecting software emulation routines. These routines can be re-written for each of the different CISC instruction sets. Since these routines are software, and not hardware, the design time is reduced.

Portanova's Advantage is Faster Design Time Than Prior-Art CISC

Indeed, Portanova asserts that it is faster to use his methodology than to build a CISC-only processor. His methodology is to first design a RISC-only processor, then to write emulation code to implement the CISC architecture. Since writing computer code is faster than designing and fabricating hardware, Portanova's design methodology is faster that designing a CISC processor:

> **The rationale** for **taking this approach** is the **RISC design time** is much, **much less** than the **CISC design time.** For example, it is known in the art that the Fairchild F9450, MD281 took longer than **three years** to develop. Using the present approach, the MIL-STD-1750 RISC emulator took **less than one year** with only one trip to the silicon factory needed to achieve certification. (col 30, lines 58-64, emphasis added)

Proposed Modification Destroys Portanova's Purpose

If Portanova added one or more CISC decoders to his RISC-only hardware, then his design time increases. This destroys the rationale for his design approach that he explicitly states as quoted above at col. 30, line 58. A modification to a reference cannot be made if that modification destroys the intended function or purpose of the reference.

Portanova requires the modification of adding a CISC hardware decode unit at a bare minimum. Adding this CISC hardware decoder and its control logic significantly increases design time. If the design time for the CISC decoder were only 1/3 of the total design time, then adding just the CISC decoder adds one year to the design time – doubling the design time from one year for RISC-only, to two years.

Not only does design time increase, but adding the CISC decoder also destroys Portanova's purpose of a design method that applies to *any number* of CISC instruction sets. Once the CISC decoder is added to the RISC hardware, then the hardware is specific to just **one** CISC instruction set. Portanova's processor would no longer be able to emulate *any number* of CISC instruction sets.

Thus adding a CISC decoder to Portanova destroys his purpose of reduced design time and also his purpose of emulating any number of CISC instruction sets.

The Federal Circuit has consistently held that when a § 103 rejection is based upon a modification of a reference that destroys the intent, purpose or function of the invention disclosed in the reference, such a proposed modification cannot be properly made. In Re Gordon, 733, F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984). Portanova's intent is clearly stated as reducing design time (col. 4, lines 3-13, col. 7, lines 37-65). Another intent is to emulate any number of CISC instruction sets (col 30, lines 48-50). Both intents are destroyed by the modification of adding a CISC hardware decoder as proposed in the rejections.

Explicit Teaching and Suggestion Lacking from Portanova

The references clearly do not teach hardware execution of two different instruction sets. The Examiner agrees that hardware execution of two instruction sets is not explicitly taught (page 7, first paragraph of action) but is suggested. However, Appellant is unable to find any language in the cited page (col 29 - col 30) of the reference containing this suggestion. In Appendix 3, Appellant discusses each of "prior-art" Figures 9, 10, 11, and 12 in the cited page, and is not able to find any such suggestion or teaching that a hardware pipeline could execute two native instruction sets. Instead these prior-art figures are simply prior art. They show existing CISC architectures that can be emulated in software. Portanova's RISC processor can be used to emulate in software these CISC instruction sets for several well-known CISC architectures, such as 68000, VAX, S/370.

It is not reasonable that a prior-art VAX combined with Portanova's RISC emulating CISC suggests a single execute unit executing both RISC and CISC. Instead Portanova clearly teaches software ***emulation*** of CISC by RISC, so the CISC VAX is implemented by software emulation.

Portanova Teaches Away - Design RISC 1st, then code CISC

Portanova clearly teaches away from hardware execution of both RISC and CISC when he teaches a two-step process:

1.) The RISC hardware is first built without regard for CISC.

2.) Then the software emulator is written for any of the CISC architectures of Figures 9-12 (col 30, lines 39-64, abstract).

Portanova clearly states this sequence:

"the designer attempts to maximize execution of the RISC (hardwired) instruction set. **Once the RISC is hardware designed** it can be sent **to the factory** for reduction to silicon. The designer **then writes the CISC** instruction **emulator** using RISC instructions, as described in the example above. (col 30, lines 52-57, emphasis added)

Hardware execution REQUIRES that CISC be considered when the hardware is designed, before being sent to the factory. Designing CISC hardware with RISC hardware requires the process of :

1.) The RISC hardware is first built with CISC hardware.

2.) A software emulator is written for instructions not in the decodable subset of CISC instructions.

This method slows down the design process since step 1 is now much more complicated. If all CISC instructions (not just a subset) are executed in hardware, then step 2 is not necessary at all. Clearly this process is not what Portanova teaches or suggests.

Identification of Basis for Suggestion Requested

If such language suggesting *hardware* execution of *two* instruction sets exist, Appellant has requested that Examiner explicitly point out <u>what figure</u> and <u>what line</u> it occurs in, rather than broadly referring to a page with four prior-art figures. This helps define the issues for appeal and ensures that Appellant and Examiner are not "discussing two completely different cases" (PTO Day 1994, pages 357-9).

Something in the Prior art must suggest the desirability of making the combination (*Uniroyal, Inc. v. Rudkin-Wiley Corp.*, 837 F.2d at 1050-51, 5 USPQ2d at 1438). The claimed invention must not be used as a blueprint. Since the suggestion does not appear in the <u>Portanova</u> reference as cited in the rejection, the suggestion must come from another source. Appellant 's claims are apparently that source. If no suggestion can be pointed out in <u>Portanova</u>, then the rejection is in error and should be overturned.

SPECIFIC RESPONSES TO STATEMENTS IN THIRD OFFICE ACTION

For completeness, the following are specific responses to specific statements made in the rejections in the third official action.

'Dual-Instruction-Set Processor' Has Different Meaning to Portanova

Examiner asserts (third action, page 4) that Appellant 's argument that <u>Portanova's</u> Figures 9-12 apply only to a single instruction set fails because <u>Portanova's</u> context is a dual-instruction set processor. However, <u>Portanova's</u> definition of dual-instruction-set processor is one that <u>emulates</u> CISC by executing RISC. Appellant 's definition of dual-instruction-set processor is one that <u>executes</u> <u>both</u> CISC and RISC. Thus the mere use of the term "dual-instruction-set processor" does not change what <u>Portanova</u> teaches or suggests, unless Appellant 's definition of dual-instruction-set processor is used as a blueprint.

Examiner believes that it is within the ordinary skill level to have a single execute unit execute both RISC and CISC instructions. Examiner also believes that it is within the ordinary skill level to modify Onishi's branch decoder to decode a second instruction set, such as adding a CISC instruction decoder to a RISC processor. These beliefs are not reasonable.

Nowhere does Portanova state that any CISC instructions can be directly executed in hardware by his RISC processor. Portanova exclusively describes CISC emulation by replacing a CISC instruction with a group of RISC instructions. Appellant asks Examiner to cite a page and line number in Portanova where such a suggestion exists.

Cited Motivation Not Relevant, Shows a Different Problem, and Points to Non-Obviousness

Examiner's motivation for using Onishi's two decoders is that it "allows the system to decode instructions more efficiently because decoding of a branch instruction usually takes longer than that of a normal instruction." (third action, page 4-5) Appellant asserts that branch instructions taking longer to decode is not a relevant motivation to having separate RISC and CISC decoders, since both decode branch instructions. RISC instructions are not decoded more efficiently because of the presence of the CISC decoder; indeed RISC instruction decoding may be *less efficient* and slower because of the additional CISC decoder tapping in to the instruction fetcher. Thus the motivation provided for combining Onishi with Portanova is irrelevant and not reasonable. The cited motivation shows Onishi to be directed to a different problem.

Onishi Merely Partitions a Decoder for a Single Instruction Set

With two instruction decoders for two different instruction sets, the same bit pattern or opcode can be decoded into two different instructions, one for RISC and the other for CISC. Both outputs can be valid operations. Thus the present invention can output valid decoded instructions from both of the decoders, and one must be selected.

Appellant 's specification explains how the same opcode, 03 hex, can be two valid operations — CISC addition or RISC trap-word-immediate:

> This same opcode, 03 hex, corresponds to a completely different instruction in the RISC instruction set. In CISC 03 hex is an addition operation, while in RISC 03 hex is TWI - trap word immediate, a control transfer instruction. Thus two separate decode blocks are necessary for the two separate instruction sets. (Spec on page 25, lines 2-6)

Onishi partitions a single decoder for a single instruction set into two decoders for different types of instructions (branches) within that one instruction set. With Onishi, one of the decoder's outputs is always invalid. The present invention uses two decoders because two separate instruction sets are decoded. Thus Onishi does not teach or suggest that a decoder for a RISC instruction set be used with a second decoder for a CISC instruction set. Certainly Onishi nowhere teaches that only a *subset* of a second, independent instruction set is decoded in a CISC decoder.

Scope of the Claims Mis-Stated by Examiner

Examiner admits that the detail of hardware implementation was not specified by either reference. However, "to the extent of the scope of the claims to design a processor capable of executing dual instructions sets where a subset of second instruction set is implemented partly with hardware, the teachings and suggestions from the applied references sufficiently meet the claim limitations." (third action, page 5).

Appellant disagrees that Appellant is merely claiming a "processor capable of executing dual instructions sets", regardless of whether the second instruction set is wholly or partially implemented in hardware. That is not what the claims state. The claims recite at least a RISC and a CISC instruction decoder, and an execute unit that receives decoded RISC and decoded CISC instructions and executes both RISC and CISC instructions. Putting a PowerPC™ Mac and a 486 PC on a desk allows execution of both a RISC and a CISC instruction set, but does not meet claim limitations of having the RISC and CISC decoders feed a single execute unit. Likewise putting Portanova's RISC CPU emulating CISC with a CISC VAX does not teach claim limitations. Also a CPU die that has a RISC pipeline in one corner and a CISC pipeline in another corner does not meet the claim limitations since decoded RISC instructions are sent to the RISC execute pipeline while decoded CISC instructions are sent to the CISC execute pipeline.

Portanova, while appearing to "execute" dual instruction sets, also fails to teach or suggest claim limitations of a RISC and a CISC decoder which feed decoded instructions to an execution unit. The fact that Portanova discusses prior-art CISC architectures such as VAX and 68000 does not mean that he suggests executing both RISC and CISC decoded instructions on the same execute unit !

Portanova specifically teaches that these prior-art CISC architectures can be emulated with his RISC processor. As the official action notes, emulation means replacing a CISC instruction with a plurality of RISC instructions. Emulation does not mean direct execution by hardware.

The third official action on page 5 has thus misstated Appellant 's claims. Even without the limitation of only a subset of the second instruction set being executed by the execute unit, the scope of the claims is narrower than merely executing dual instruction sets. The structure of two instruction-set decoders using a single execute unit is claimed but not taught or suggested by any cited reference.

'Piecemeal' Attack of References Proper

Examiner objected to Appellant 's analysis of the references, stating that "Appellant s attempt to show non-obviousness by piecemeal analysis of the references. Appellant s are reminded that one cannot show non-obviousness by attacking references individually where, as here, the rejections are based on combinations of references." (third action, page 6).

Appellant strongly disagrees. When the Examiner cites a portion of a reference as teaching a claim element, the Appellant can show that such reliance is in error. For example, if an Examiner states that reference P teaches X while reference Q teaches Y, then Appellant can properly argue that reference P does not, in fact, teach X.

In the present case, Examiner has relied on Portanova for a suggestion of hardware execution of two instruction sets. Examiner states that "Portanova, however, clearly suggests that hardware implementation of CISC emulation could have been done as an alternative approach (see col 29, line 60 - col 30, line 12)." Appellant can properly show that Examiner's reliance on the cited portion of the reference to be in error. Appellant has done this by a through analysis of this cited portion of Portanova.

Appellant may also show that it is improper to combine references, such as when a secondary reference solves a different problem (*In Re Clay*, 23 USPQ 2d 1058). Since Onishi solves the problem of branch decoding by splitting a decoder into two parts, Appellant has pointed out that the branch decoder does not solve the problem of decoding two entirely different instruction sets, such as CISC and RISC. Both of Onishi's decoders merely decode different parts of a single instruction set.

Strained Combination of References

Portanova fails to provide any detail of CISC hardware implementation, since Portanova only emulates CISC instructions with RISC instructions. Examiner has attempted to rely on brief prior-art descriptions in Portanova of well-known CISC architectures for CISC hardware execution. This attempt fails because Portanova teaches away by showing that the RISC hardware can be used to emulate ANY NUMBER of these different CISC architectures by first designing generic RISC hardware without regard to CISC, and then writing CISC emulation code containing RISC instructions. This speeds up his design time.

References Lack Claim Elements

Examiner also cannot rely on these brief prior-art descriptions because they fail to disclose the structure recited in Appellant 's claims. For example, having two instructions decoders (RISC and CISC) but only one execution unit is claimed by the elements of claim 18. Portanova does not disclose a CISC instruction decoder nor an execution unit capable of executing both RISC and CISC instructions. Onishi is brought in as showing two instruction decoders, but this fails because Onishi's decoder is merely a branch decoder, and not a decoder for a second (CISC) instruction set. Onishi simply partitions or divides a single decoder for a single instructions set into a branch decoder portion and a non-branch decoder portion.

Onishi, like Portanova, completely lacks any teaching or suggestion of an

"execution unit, coupled to said RISC instruction decode means and said CISC instruction decode means, for executing instructions belonging to said RISC instruction set and instructions belonging to said CISC instruction set,

whereby instructions from said RISC instruction set and instructions from said CISC instruction set can be executed by said execution unit." (claim 18, emphasis added).

'Design Choice' Is Repackaged 'Obvious To Try' Standard

Tanenbaum, Lee et al., and Agnew et al. were added in the third action to further strain the combination of references. Examiner is using Tanenbaum to suggest that software or hardware is merely a design choice. The two newly-cited references are used as "further evidences of hardware implementation as opposed to software implementation". However, neither reference teaches or suggests the hardware claimed by Appellant , such as the execute unit receiving decoded RISC and decoded CISC instructions. A "design choice" is another way saying "obvious to try". "Design Choice" or "Obvious to Try" is not prima facie obviousness.

'Design choice' is the cited motivation on page 5, line 3 of the third action, and again in the second office action on page 7, line 15 and page 5, line 5:

> Thus, it would have been an obvious **engineering design choice** to one of ordinary skill in the art at the time of the invention to utilize both software and hardware implementation to emulate CISC instructions on a RISC computer. (emphasis added)

Design choice is another way of stating 'obvious to try'. The designer has the 'choice'. What is the 'choice' ? It is experimentation. When several alternative approaches exist, the Examiner is not free to choose the more obscure choices when the prior art clearly points toward another 'choice'. In this case, the prior art as a whole and Portanova in particular clearly teach software emulation, not the invention. Certainly the structure of the invention — dual instruction-set decoders using a shared execution unit — is nowhere taught or suggested in the prior art.

What are the 'Obvious' Design Choices ?

The 'obvious' way to execute both RISC and CISC instruction sets is to duplicate the processor core. One core executes RISC, while the other core executes CISC. Each core has its own instruction fetcher, decoder, execute unit, and TLB. This is similar to having a co-processor for the second instruction set.

In contrast to this 'obvious' way, the invention does not duplicate the entire core. Rather than duplicate the entire core, the instruction fetcher, execute unit, and TLB are shared among both instruction sets. Only the CISC decode logic needs to be added.

Another approach used commercially is to emulate the CISC instruction set by replacing or translating CISC instructions into RISC instructions. This is what the Portanova reference does. This approach can require little or no additional hardware, although the performance is poor, as each CISC instruction is replaced by dozens or hundreds of RISC instructions. In contrast, the invention can directly decode and execute the simpler CISC instructions, although more complex CISC instructions are emulated by replacing them with groups or routines of RISC instructions.

The invention taught by the Appellant differs from both of these more obvious approaches by directly executing both RISC and CISC instructions in a shared execute unit. The Appellant 's disclosure is being used as a blueprint to reject his own claims. What the prior art as a whole fairly teaches and suggests is to use a co-processor and duplicate all of the hardware, or to emulate the entire CISC instruction set.

The prior art itself in no way suggests adding just the second decoder but not the rest of the core. The prior art nowhere suggests executing some of the CISC instructions but not all of them on a RISC/CISC processor. The Examiner has impermissibly used the hindsight gained from Appellant 's disclosure to make major, unsuggested modifications to the prior art to reject Appellant 's claims.

The prior art as a whole fairly teaches and suggests either of the two approaches described above. No actual suggestion in the prior art references exists pointing toward the claimed invention. Other combinations are possible, and this undue experimentation has been cloaked as an "engineering design choice." Something in the prior art must suggest the desirability of making the combination. *Uniroyal, Inc. v. Rudkin-Wiley Corp.*, 837 F.2d 1044, 5 USPQ2d 1434, 1438 (CAFC 1988). If the prior art provides no teaching, suggestion, or incentive supporting the combination proposed by the Examiner, then the rejection is in error and must be reversed. *In Re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (CAFC 1990). The claimed invention must not be used as a blueprint.

Invention does not merely Duplicate Hardware — Only Decoders

The remarks in paragraph 16 of the second office action refer to <u>Onishi</u>'s two decoders as "clear evidence of a system employing partially duplicated hardware resources". It was also stated that whether the emulation unit is integrated or separate is a design of choice.

Appellant 's invention allows both instruction sets to be executed on a single execution unit, eliminating duplicated hardware for execution. Thus duplicated hardware resources are not needed for the execute unit, but only for the decoders. This approach is not suggested in any of the cited references and is not merely a design choice of separating units, as in a coprocessor, or integrating units.

Newly-Cited References Show Extensions of One Instruction Set

Appellant has reviewed the two newly-cited references. Lee et al. is clearly directed to an extension of a single instruction set rather than a separate second instruction set. Lee et al. teach an "assist" that is attached to the main processor via a set of busses (col 2, lines 40-45). Thus the "assist" appears to be the co-processor embodiment with a new name.

Co-processors execute in hardware extensions of a single instruction set. A co-processor is a separate (*2nd*) execution facility which supports a subset of a single instruction set, while the invention is a *single* execution facility which supports a multiplicity of instruction sets.

Agnew et al. also teach a single instruction set that is partitioned into two or more subsets, each possibly implemented in a different chip or emulated by software. His Figure 3 again shows a co-processor embodiment connected to a bus.

In contrast, Appellant 's claim 1 recites "two separate instruction sets", and claim 18 recites a RISC and a CISC instruction set. Claim 1 clearly disallows a mere extension of a single instruction set by stating: "said first encoding of instructions independent from said second encoding of instructions". Mere extensions of instruction sets must have dependent encodings since otherwise one opcode could be used for two instructions. However, two separate instruction sets, such as RISC and CISC, have one opcode with two different instructions. (Specification, page 25, lines 2-6.)

Examiner appears to be using these three new references to show that Portanova's RISC hardware can be modified to execute native CISC instructions. However, these newly-cited references only show separate co-processors that would have separate execute units. Also, these references show mere extensions to a single instruction set, such as for floating point instructions. Thus, even if these references were used as secondary references in a non-final action, these references do not teach or suggest a single execute unit that executes both RISC and CISC instructions.

OTHER CLAIMS NOT OBVIOUS & RECITE ADDITIONAL ELEMENTS

Claim 2 recites that the single instruction fetch buffer (32 of Figure 2, Appendix 2) feeds instructions to both the RISC and CISC decoders 36. Rather than duplicate the entire pipeline, the instruction fetcher, execute unit, and TLB are shared among both instruction sets. Only the CISC decode logic needs to be added. It is not obvious that just the decoder be duplicated while the instruction fetch buffer and execution units not be duplicated. The obvious approach is to duplicate the entire pipeline.

No Reference Cited Having Mode Register With RISC-CISC bit

Some elements of dependent claims have not been shown in any of the references. For example, claim 3 claims a "mode register means, coupled to the select means, for indicating an instruction set to be decoded and executed." Examiner has not cited any reference with such a mode register. Independent claim 18 likewise recites a "mode register means for indicating a current operating mode of said microprocessor".

Instead, claims 3 and 4 were rejected as "obvious to one skilled in the art to utilize an execution mode register for indicating the execution of native and non-native instructions." (second action, page 4) No prior art was cited for the mode register with the bit indicating the instruction set, nor was any motivation given other than 'obvious'. Claim 18 was rejected 'for the same rationales' (second action, page 5).

Claim 19 recites that the mode register means indicates RISC mode, CISC mode, or an emulation mode. Nowhere has the Examiner shown a teaching or suggestion for such a mode register indicating one of RISC, CISC, or emulation modes.

Claims 5, 15 and 20 indicate that the instruction set is changed when an *undecodable* instruction is signaled by the second instruction set (CISC) decoder. Since some, but not all, CISC instructions are decoded by the CISC decoder, the CISC decoder signals the mode control to change to emulation mode (claims 15, 20) or first-instruction-set (RISC) mode when an undecodable CISC instruction is encountered (claim 5).

Portanova does not teach changing from CISC mode to RISC mode or emulation mode. Apparently his RISC processor is always emulating CISC instructions with RISC instructions and thus has no need to change to decoding CISC instructions. Indeed, Portanova never executes or decodes CISC instructions but merely emulates them. There is simply no teaching in Portanova as relied on by the rejection of claim 5 (second action, page 4-5).

Bullions Terminology Differs from Standard Usage - Claims 6-13, 17

Since Portanova and Onishi do not teach a TLB, the Bullions reference is cited for claims 6-13, 17 for teaching a TLB having translations for both host and guest instructions, and for teaching that a miss in the TLB also triggers a change of execution mode.

Bullions teaches a CISC system that emulates guest architectures on a native architecture. Each level of architecture is capable of using virtual addressing with dynamic address translation. (col 1, lines 8-25). All operating systems described by

Bullions are well-known CISC architectures (col 1, lines 29, 50). Thus Bullions does not teach processing both RISC and CISC instructions, but merely teaches emulating "guest" CISC architectures on a native CISC machine.

Bullions uses the word "architecture" to mean something other than "instruction set". His summary and claims refer to guest "programs" but not to guest "instructions" from a different "instruction set". A guest program does not necessarily use a different instruction set. Indeed, the "plural levels of architecture" referred to "involve plural levels of address translation." (col 5, lines 24-29). These plural architectures refer to address translation architectures, not to instruction set architectures. Bullions clearly states that "different architectures may use different size addresses, e.g. one architecture may use 24 bit addresses while another architecture may use 31 bit addresses." However, all operating systems described by Bullions use the same instruction set.

For example, the 286 microprocessor uses 24-bit addressing while the 386 uses 32-bit addressing. Yet both the 286 and 386 execute the same CISC x86 instruction set.

Claim 6 recites that the TLB provide "an indication to said mode control means to change said instruction set decoded to said first instruction set when no translation is found in said TLB". Bullions does not teach that another instruction set is decoded. Bullions teaches that the architecture "mode" is changed on a TLB miss, causing a native program rather than a guest program to execute. However, these programs are from the same instruction set, not different instruction sets.

It is thus improper to replace Bullions architecture or program with the word "instruction", as his programs and architectures refer to different address translation architectures and not to different instruction sets. Bullions teaches guest *programs* and guest *architectures*, but not guest *instructions* from a different instruction set.

Claim 17 recites translating memory references generated by CISC instructions that are directly executed, where the translation of memory references is controlled by a software translator routine comprised of RISC instructions. Bullions fails to teach that RISC instructions are used in a software translator routine while some CISC instructions are directly executed. Thus claim 17 cannot be obvious in view of Bullions and the other references.

Subject of Dependent Claims 7-13, 16 Not Cited or Obvious

In claim 7 a handler routine of *first* instructions is executed when the mode control is signaled by an undecodable *second* instruction or a TLB miss. No reference teaches or suggests that a handler routine of instructions from a first instruction set be fetched from main memory and executed when a signal is received from a decoder for a second instruction set or from a TLB. No handler routine has even been cited in the references. Certainly a handler routine from one instruction set being called by a decoder for *another* instruction set is not obvious.

In Claim 8 an exception from the execute unit causes the mode control to change to decoding the first instruction set. Bullions was cited for teaching switching the execution mode in response to an interrupt at col. 13, lines 18-62. This cited text simply teaches calling a control program (CP) when an interrupt is received. No instruction set switch occurs.

Claim 9 recites that all references to main memory from the second instruction set are translated by the TLB, but claim 10 reveals that only *first* instructions can load the TLB. No specific rejection for claims 9-10 have been given by the Examiner. The cited references fail to teach or suggest that a *first* instruction set is used to *load* a TLB which translates main memory references generated by a *second* instruction set. This is not obvious nor trivial. Examiner should cite all claimed elements or allow the claim if unable to do so.

In claim 11 extended first instructions modify TLB entries. Bullions is cited as teaching "a special instruction to initiate software routine emulation and reload the TLB (see col. 12, lines 63-67)." The cited text teaches a "start interpretive execution (SIE) instruction. This instruction simply initiates guest virtual machine (VM) operation. Nothing is said here about reloading the TLB. Claim 11 recites that the "address translation entries in said TLB are modified only by said extended instructions."

In claim 12 first instructions are decoded following a reset, while Claim 13 further recites that these extended instructions are decoded only when the mode control is signaled to change instruction sets, or immediately following a reset. Examiner has rejected claims 12-13 as merely "obvious to one skilled in the art to reset the system execution mode to a normal operation in response to a system reset signal." Examiner is obliged to cite references showing these claim limitations. Indeed, the normal mode of operation could be considered the mode the user executes his programs in, which is not used after a reset. Instead of a 'normal' mode, an operating system may be activated after a reset.

Claim 16 adds that the emulation routine uses RISC instructions and extended instructions. The extended instructions use undefined opcodes in the RISC instruction set. Examiner has not cited any reference teaching that undefined RISC opcodes be used for extended instructions. Indeed, no rejection was given for claims 14-16, 18-20 except for being rejected "for the same rationales".

Many other claims were summarily rejected with no attempt at an element-by-element analysis. If these features are truly conventional, then the Examiner is obliged to cite references for these features or submit an affidavit. M.P.E.P. § 706.02(a). Clearly the scope and subject of these claims differ from the scope of other claims, and the Appellant deserves to have these claims examined too. M.P.E.P. §§ 904, 904.02.

CONCLUSION

The Portanova reference does not explicitly teach hardware execution of both CISC and RISC instruction sets in a single processor. Portanova explicitly teaches emulation of CISC using RISC instructions. CISC instructions are never directly executed in Portanova's hardware. No reference teaches that only a *subset* of a second, independent CISC instruction set is decoded while an entire first (RISC) instruction set is decoded.

No valid 'suggestion' of hardware execution of CISC instructions by Portanova is possible since Portanova explicitly teaches away from CISC hardware being designed with the RISC hardware, stating:

> "the designer attempts to maximize execution of the RISC (hardwired) instruction set. **Once the RISC is hardware designed** it can be **sent** to the **factory** for reduction to silicon. The designer ***then*** **writes** the **CISC** instruction **emulator** using RISC instructions, as described in the example above. (col 30, lines 52-57, emphasis added)

Portanova designs the RISC hardware and sends the design out for silicon fabrication, ***then*** writes CISC emulation code.

The purpose of Portanova is destroyed by any modification for hardware execution of CISC instructions, since additional CISC hardware must be included in the hardware design, increasing the design time that Portanova claims to reduce.

Indeed, Appellant's believe that they have significantly advanced the state of the art by their invention which efficiently uses a single execute pipeline but two instruction decoders. Only a subset of the second instruction set need be decoded and directly executed. Others will waste precious CPU die area by having two separate, complete CISC and RISC execute pipelines. Others will decode the entire CISC instruction set.

Some (such as Portanova) will forfeit performance by emulating one instruction set. Appellant 's invention is efficient and the public would benefit by its disclosure.

For the foregoing reasons, Appellant submits that the rejection of claims 1-20 is in error and should be reversed on appeal.

Stuart T. Auvinen
429 26th Avenue
Santa Cruz, CA 95062

(408) 476-5506
(408) 477-0703 Fax

Respectfully Submitted,

[illegible]

Stuart T. Auvinen
Agent for Appellant
Reg. No. 36,435