

Case IPR2023-00274
U.S. Patent No. 8,713,289 B2

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

LZLABS GMBH

Petitioner

v.

INTERNATIONAL BUSINESS MACHINES CORP.

Patent Owner

Case IPR2023-00274

U.S. Patent No. 6,785,289 B2

INTERNATIONAL BUSINESS MACHINES CORPORATION'S
PATENT OWNER'S SURREPLY

Mail Stop Patent Board
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION	1
II. UNREBUTTED CONTENTIONS	2
III. PETITIONER’S NEW CLAIM CONSTRUCTION ARGUMENT IS UNTIMELY AND THEREFORE IMPROPER	2
IV. PETITIONER RELIES ON ATTORNEY ARGUMENT IGNORING TESTIMONY FROM BOTH PARTIES’ EXPERTS.....	6
V. PETITIONER MISREPRESENTS THE INTRINSIC RECORD	8
A. “Target Machine Condition Code” in the Claims	12
B. “Target Machine Condition Code” in the Specification	12
C. “Target Machine Condition Code” in the Prosecution History	14
VI. LZLABS IGNORES LODERER’S EXPRESS TEACHINGS AND ITS OWN EXPERT’S UNDERSTANDING OF THOSE TEACHINGS.....	16
VII. EXTRINSIC EVIDENCE IS CONSISTENT WITH INTRINSIC RECORD	16
VIII. LZLABS FAILS TO SHOW THAT IBM’S IPR POSITION IS INCONSISTENT WITH ITS LITIGATION POSITIONS.....	23
IX. LZLABS FAILS TO SHOW THAT IBM’S ARGUMENT IS INCONSISTENT WITH THE CLAIM LANGUAGE AND THE INTRINSIC RECORD	24
X. CONCLUSION.....	25

EXHIBIT LIST

Exhibit No.	DESCRIPTION
2001	Declaration of Richard M. Goodin
2002	Curriculum Vitae of Richard M. Goodin
2003	Transcript of Mr. Jestice's 9/13/2023 Deposition
2004	Hennessy, John L., et al. <i>Computer Architecture: A Quantitative Approach</i> . India, Elsevier Science, 2006 (excerpt)
2005	Pentium™ Processor User's Manual, Volume 3: Architecture and Programming Manual, 1994

All emphasis added, unless specified otherwise.

All annotations have been added, unless specified otherwise.

All objections in deposition transcripts have been removed.

I. INTRODUCTION

The Petition and Reply have failed to demonstrate the unpatentability of the challenged claims. As IBM's expert explained and *LzLabs*' expert admitted, Loderer discloses a target machine that does *not* provide "condition codes."

In its Reply, LzLabs ignores the record and the inevitable defeat of the Petition. Instead, it resorts to a claim construction argument related to "target machine condition code" based solely on attorney argument. The Court should reject LzLabs "target machine condition code" argument for two reasons.

First, LzLabs' "target machine condition code" argument is untimely. In the Petition, LzLabs took the position that *no* claim term requires construction, and its expert, Mr. Jestice, took the same position in his declaration. Moreover, construction of the term "target machine condition code" was not an issue discussed in the Institution Decision or the POR. Thus, LzLabs waived any argument that "target machine condition code" required construction, and its argument in the Reply is improper.

Second, LzLabs' "target machine condition code" argument is based on its misapprehension of IBM's arguments. Specifically, LzLabs characterizes IBM's argument as contingent on storing condition codes in a "special type of storage." Reply, 26. To the contrary, as explained below, IBM's position is that the target

machine provides a “condition code;” not whether condition code values are, after being generated, stored in a particular location.

Unable to dispute that Loderer expressly discloses that its target machine does not provide “condition codes,” LzLabs ignores the record and mischaracterizes IBM’s arguments to confuse the issues.

II. UNREBUTTED CONTENTIONS

LzLabs does not dispute that both parties’ experts agree Loderer discloses a target machine that does not provide condition codes. *See* EX2003, 89:4-7; EX2001, ¶57; POR, 35. LzLabs also does not dispute that the Petition maps the claimed “target machine condition code” to Loderer’s “bit pattern BM” (stored in register REG). *See* Petition, 34; EX1003, ¶111; EX2003, 99:19-22. And LzLabs does not dispute that both parties’ experts agree Loderer’s bit pattern is *not* a condition code. *See* EX2003, 86:5-87:2; EX2001, ¶90; POR, 39.

III. PETITIONER’S NEW CLAIM CONSTRUCTION ARGUMENT IS UNTIMELY AND THEREFORE IMPROPER

In the Petition, “Petitioner submit[ed] terms should be given ‘ordinary and customary meaning’ to those of skill in the art. 37 C.F.R. §42.100(b). Petitioner reserves the opportunity to respond to constructions offered by PO or adopted by the Board.” Petition, 17. In the Institution Decision, the Board noted that “[f]or purposes of this Decision, we conclude that no claim term requires express

interpretation to resolve any controversy in this proceeding.” Institution Decision (“DI”), 7. In the POR, “IBM adopt[ed] the Board’s application of the plain and ordinary meaning of the claim terms.” POR, 26.

In the Reply, Petitioner advances a new claim construction position for the term “target machine *condition code*” as “encoded values, stored as bits in a register, corresponding to *condition codes* that would have been generated from executing instructions on the source machine.” Reply, 1.

But the term “condition code” is used in LzLabs’ proposed construction of “target machine condition code.” Reply, 1. The qualifier “target machine” in “target machine condition code” merely specifies the location of the “condition code,” e.g., in the target hardware instead of the source hardware. That is, the qualifier “target machine” neither provides special characteristics to nor defines the “condition code” in the “target machine.” Courts have repeatedly criticized using the claim term within the proposed construction because (1) such use is circular and does not help to clarify the meaning of the term itself; and (2) such use is proof

that the claim term does not require construction and be given plain and ordinary meaning.¹

¹ See, *Care N' Care Ins. Co., Inc. v. Integrated Claims Sys., LLC*, CBM2015-00063, Paper 23, 8 (P.T.A.B. July 28, 2016) (“We see no reason to interpret ‘fields’ as ‘computer software fields.’ We agree with Petitioners that ***Patent Owner’s proposed construction is circular insofar as the construction includes the entire term being construed.***”); *Triplay, Inc. v. Whatsapp, Inc.*, 2016 WL 3574012, at *15 (D. Del. June 30, 2016), report and recommendation adopted, 2016 WL 6778215 (D. Del. Nov. 15, 2016) (“TriPlay ended up with a proposed construction that includes the very term to be construed ..., ***a circular result that is disfavored.***”); *Paice LLC v. Hyundai Motor Co.*, 2014 WL 3725652, at *11 (D. Md. July 24, 2014) (finding Defendants’ use of the term ‘motor(s)’ in proposed construction for the disputed term ‘motor’ highlighted that the proposed construction failed to provide clarification of the disputed term to the jury.); *Phoenix Licensing, L.L.C. v. AAA Life Ins. Co.*, 2015 WL 1813456, at *22 (E.D. Tex. Apr. 20, 2015) (“Defendants’ construction includes a form of the disputed term in their proposed construction. ***This confirms that the term ‘offer[ing]’ is not confusing or ambiguous.*** Therefore, the term will be given its plain and ordinary meaning.”); *Nike, Inc. v. Lululemon*

In addition, LzLabs' proposed construction is untimely and therefore improper. As the Trial Practice Guide ("TPD") notes "the Board will permit the petitioner, in its reply brief, *to address issues discussed in the institution decision.*" TPG, 73. But the construction of the term "target machine condition code" was not an issue discussed in the DI or the POR.

Additionally, both parties' experts agree that "target machine condition code" requires no construction. Specifically, Petitioner's expert stated that "the claim terms of the '289 patent *do not require construction to address the issues presented.* In my analyses of the '289 patent, I have applied *the ordinary meaning* of the claim language as it would have been understood to a POSA in 2007 *in view of the patent specification and its prosecution history.*" EX1003, ¶74. *See also*, EX2001, ¶32 ("I have applied the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art in my analysis provided herein.").

Unable to rebut both parties' experts testimony that Loderer does not disclose a "target machine condition code" under the plain and ordinary meaning of the term (*see* Section II), LzLabs argues for the first time in the Reply that the

USA Inc., 2023 WL 9003708, at *2 (S.D.N.Y. Dec. 28, 2023) ("[L]ululemon's proposed construction includes the words in the disputed claim term ... which is telltale evidence that those terms do not require construction.")

“target machine condition code” should be construed as “encoded values, stored as bits in a register, corresponding to condition codes that would have been generated from executing instructions on the source machine.” But “Petitioner may not submit new evidence or argument in reply that it could have presented earlier, e.g. to make out a prima facie case of unpatentability.” TPG, 73. “In an IPR, the petitioner has the burden *from the onset* to show with particularity why the patent it challenges is unpatentable.” *Harmonic Inc. v. Avid Tech., Inc.*, 815 F.3d 1356, 1363 (Fed. Cir. 2016) (citing 35 U.S.C. § 312(a)(3) (requiring IPR petitions to identify “with particularity ... the evidence that supports the grounds for the challenge to each claim”)). LzLabs should have presented the particular construction in the Petition. Accordingly, LzLabs’ proposed construction is improper.

IV. PETITIONER RELIES ON ATTORNEY ARGUMENT IGNORING TESTIMONY FROM BOTH PARTIES’ EXPERTS

Both parties’ experts agree that “target machine condition code” requires no construction. LzLabs’ expert stated that “the claim terms of the ’289 patent *do not require construction to address the issues presented*. In my analyses of the ’289 patent, I have applied *the ordinary meaning* of the claim language as it would have been understood to a POSA in 2007 *in view of the patent specification and its prosecution history*.” EX1003, ¶74. IBM’s expert also stated that he has

“applied the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art...” EX2001, 32.

Applying the ordinary meaning of the term “target machine condition code” LzLabs’ expert, Mr. Jestice, testified that Loderer discloses *a target machine with no condition codes*:

Q. ... The invention described in Loderer relates to *a target hardware that is not providing a condition code, correct?*

A. *That’s correct.*

EX2003, 89:4-7.

Mr. Jestice also mapped the claimed “target machine condition code” to Loderer’s “bit pattern” BM stored in a general-purpose register:

Q. So it is your opinion that *the target machine condition code in Loderer is the bit pattern BM*, correct?

A. Right.

EX2003, 99:19-22.

But Mr. Jestice admitted that Loderer’s *bit pattern is not a condition code*:

Q. Loderer uses the term “condition code on the source hardware,” correct?

A. Yes.

Q. Loderer also uses the term “a bit pattern on the target hardware,” correct?

A. Yes.

...

Q. So you're saying because the bit pattern in the target architecture uses a single bit for its condition, Loderer chooses to call it a bit pattern instead of a condition code?

A. *It isn't a condition code. It's a bit pattern.*

EX2003, 86:5-87:2. IBM's expert agreed. *See* EX2001, ¶¶57, 62.

Instead of rebutting the expert testimony of record, LzLabs advances attorney argument about the scope of a "target machine condition code" without any testimonial evidence for support. Although LzLabs had the opportunity to submit an expert Reply Declaration, they chose not to have their expert support their untimely argument and undergo another cross-examination. Indeed, LzLabs was so keen to keep expert testimony out of the record that they did not even depose IBM's expert. Regardless, as explained below LzLabs' arguments fail.

V. PETITIONER MISREPRESENTS THE INTRINSIC RECORD

LzLabs argues that "[t]he disclosed technique of the '289 patent results in generating and storing an encoding of the condition code setting in register *rC*, *a general-purpose register on the target machine*." Reply, 4. LzLabs is mistaken. LzLabs' made-up attorney argument is not supported by the evidence. Tellingly, the Reply does not point to the Specification to support LzLabs' contention that *rC* is a "general-purpose register." There is no such disclosure. Instead, the Specification discloses that an instruction executed in the target machine *sets one*

or more target CPU flag bits and the result is placed in register rC. Accordingly, register rC holds the result of one or more target CPU flag bits. EX1001, 7:51-55 (“*[A]n instruction ...in the target machine’s architecture is executed* which mimics the instruction present in the source instruction stream in a manner *which sets one or more target CPU flag bits and which places a result in a storage location (first location, such as rC above)...*”).

A POSITA would have understood there is a distinction between a “general-purpose register” and a “condition *code* register,” and the trade-offs of using one instead of the other. For example, when discussing Petitioner’s EX1009 (MIPS RISC Architecture), Mr. Jestice, LzLabs’ expert, testified:

Q. The third sentence [of EX1009 at page 43] reads “This approach greatly simplifies handling of the instruction pipeline and eases many compiler tasks but does require that programs explicitly check for conditions such as overflow and carry.” Did I read that correctly?

A. You did, yes.

Q. Is it your understanding that this section describes *the benefits of using general-purpose registers instead of condition code registers?*

A. *That’s what this article says, yes -- or this book says.*

EX2003, 51:13-52:1.

But more importantly, IBM’s argument is not contingent on where the condition codes are stored *after* being generated. IBM’s argument is contingent on

whether the target machine provides, e.g., generates, condition codes in the first place. Consistent with the claims, which require a “target machine condition code” and with the Specification, which discloses that “an instruction ... in the target machine architecture ... *sets one or more target CPU flag bits*,” IBM’s argument is that Loderer does not satisfy the claimed “target machine condition code,” because it discloses a target machine that does not provide condition codes. POR, 26-34.

LzLabs also alleges that “[t]he teachings disclosed in the ’289 patent are expressly applicable to *any* target architecture.” Reply, 5 (original emphasis). LzLabs points to the highlighted sentences shown below from the Specification, as allegedly providing support:

The teachings disclosed in the ’289 patent are expressly applicable to *any* target architecture. The specification states “the principles set forth herein are applicable to any source architecture and to *any target architecture*.” *Id.*, 1:47-49; *see also id.*, 3:29-32 (“[T]he principals, techniques and methods of *the present invention are not limited to any particular target machine architecture*. The two exemplar architectures discussed herein are merely the most currently ones anticipated to be of the greatest value.”).³

Reply, 5.

Conveniently, LzLabs takes these disclosures out of context. Specifically, regarding the first disclosure, the Specification recites:

While the systems and methods of the present invention are widely applicable to any emulation method *where condition codes are present*, it is particularly applicable to the emulation of the *z/Architecture*. However, the principles set forth herein are applicable to any source architecture and to any target architecture.

EX1001, 1:44-49.

Accordingly, the last sentence, which allegedly forms the basis for LzLabs' argument, merely discloses that the invention principles are not limited to the emulation of the *z/Architecture* but are applicable to other target architectures "where condition codes are present."

Regarding the second identified disclosure, the Specification recites:

The present invention provides specific guiding techniques ... to efficiently *set conditions codes* ... in an emulated binary translation environment for the *z/Architecture*. These techniques are specifically directed to situations in which the *PowerPC architecture and the Intel IA32 architectures* are employed to emulate the *z/Architecture*... However, it is noted that the principals, techniques and methods of the present invention are not limited to any particular target machine architecture. The two exemplar architectures discussed herein are merely the most currently ones anticipated to be of the greatest value.

EX1001, 3:12-34.

Accordingly, the Specification merely discloses that the invention is not limited to the *PowerPC architecture and the Intel IA32 architectures*. But even assuming *arguendo* that LzLabs' argument had any merit, it is still inconsistent with the claim language that requires "target machine condition codes." That is, the claims do not cover *any* architecture; only those architectures that provide "target machine condition codes."

A. "Target Machine Condition Code" in the Claims

LzLabs mischaracterizes IBM's argument when LzLabs argues that "the challenged claims do not speak to *where* the target machine condition code must be stored on the target machine or otherwise limit the architecture of the target machine, as PO contends." Reply, 7. IBM's argument is that the plain and ordinary meaning of "target machine condition code" is not a bit pattern in a general-purpose register, which is consistent with Mr. Jestice's testimony that a bit pattern is not a condition code. *See* EX2003, 86:5-87:2.

B. "Target Machine Condition Code" in the Specification

LzLabs argues that the "[t]arget machine condition code" of the '289 patent thus includes emulated condition code settings on the target machine that would have resulted from execution of the source machine instruction on the source machine." Reply, 9. That argument is a red herring. That the "target machine

condition code” is an emulated condition code has no bearing on whether the target machine condition code is generated or not. As the '289 Patent explains:

[I]t is the job of emulation software to accept, as input, strings of source architecture instructions and to *generate therefrom strings of instructions that, when run on the target architecture, produce the same results*. These results include the *setting of various condition codes, such as sign, carry, overflow* and various others indicating exceptions and machine states.

EX1001, 1:51-57. Accordingly, the target machine condition codes are set as a result of instructions that—when run in the target machine—emulate, e.g., produce the same results as instructions that would run on the source machine.

LzLabs also argues that “[m]oreover, the '289 patent explains that the calculated target machine condition code can be stored in locations *other than* hardware or condition code elements of the target architecture.” Reply, 10 (original emphasis). LzLabs points to the Specification at 1:57-62 for support:

It is noted that while an emulation environment preferably results in the setting of hardware or condition code elements in the target architecture, *the present invention also contemplates the situation in which condition codes are generated and stored in locations other than condition code registers in the target machine.*

Reply, 10 (citing EX1001, 1:57-62 (original emphasis)).

Again, Petitioner conveniently ignores the Specification's full disclosure. That the condition code values can be stored in locations other than "condition code registers in the target machine" does not suggest that "condition codes" are not generated in the target machine. Petitioner's quote expressly states "***condition codes are generated*** and stored in locations other than condition code registers in the target machine." That is, ***after the condition codes are generated in the target machine***, e.g., in CPU flag bits or in a condition code register, they can be stored in other registers, e.g., maybe for accessing them or manipulating them from a more convenient location. Specifically, the '289 Patent discloses that "***an instruction ... in the target machine's architecture is executed*** which mimics the instruction present in the source instruction stream in a manner ***which sets one or more target CPU flag bits and which places a result in a storage location (first location, such as rC above)...***". EX1001, 7:51-55.

C. "Target Machine Condition Code" in the Prosecution History

LzLabs points to IBM's statements during prosecution and argues that "according to PO during prosecution, a 'target machine condition code' is bits of data representing a value that is calculated by a sequence of target machine instructions that may be stored in a ***general-purpose register, such as register rC.***" Reply, 11-12 (quoting from EX1004, 83-84). Again, LzLabs mischaracterizes the record. The prosecution history statements do not characterize "register rC" as a

“general-purpose register.” *See* EX1004, 83-84. LzLabs repeatedly characterizes register rC as a general-purpose register but cannot point anywhere in the record for support.

LzLabs also argues:

PO, however, overlooks that the term “target machine condition codes” was part of the original claims (EX1004, 405) and that the amendments to independent claim 9 the POR discusses (POR, 13-15) did not introduce the term “target machine condition code” into the claims. Rather, these amendments merely clarified that “the condition code” referenced in the latter part of the claim refers to the same “target machine condition code” already recited in the pending claim.

Reply, 12-13, n.5. LzLabs mischaracterizes the POR. The POR did not argue that the amendments to independent claim 9 introduced the term “target machine condition code” into the claims. Rather the POR stated that “IBM amended independent claim 9 (and others) *to clarify the requirement that the claimed ‘condition code’ is a ‘target machine condition code.’*” POR, 14. This statement is consistent with LzLabs’ characterization. *See* Reply, 12-13, n.5 (“[T]hese amendments merely *clarified* that ‘the condition code’...”). This argument is another attempt by LzLabs to confuse the issues.

VI. LZLABS IGNORES LODERER’S EXPRESS TEACHINGS AND ITS OWN EXPERT’S UNDERSTANDING OF THOSE TEACHINGS

Pages 13-17 of the Reply provide an overview of Loderer similar to the discussion of Loderer in the Petition. *Compare* Reply, 13-17 with Petition, 10-13, 18-22. However, LzLabs continues to ignore Loderer’s express teaching that the target machine in Loderer does not provide “condition codes.” *See* EX1005, 2 (“The object of the invention is ... to specify a method for simulating a condition code when porting hardware-specific program code from a source hardware to *a target hardware that is not providing for condition code...*”). Even LzLabs’ own expert understood Loderer to disclose a target machine that does not provide condition codes:

Q. ... The invention described in Loderer relates to *a target hardware that is not providing a condition code*, correct?

A. *That’s correct.*

EX2003, 89:4-7. LzLabs has yet to address its own expert’s admissions.

VII. EXTRINSIC EVIDENCE IS CONSISTENT WITH INTRINSIC RECORD

LzLabs criticizes the POR for “us[ing] extrinsic evidence to limit ‘target machine condition code’ in a manner that is inconsistent with the intrinsic record and inconsistent with PO’s infringement allegations in parallel litigation between

the parties.” Reply, 17. The Table below shows IBM’s exhibit list in this proceeding.

<u>EXHIBIT LIST</u>	
Exhibit No.	DESCRIPTION
2001	Declaration of Richard M. Goodin
2002	Curriculum Vitae of Richard M. Goodin
2003	Transcript of Mr. Jestice’s 9/13/2023 Deposition
2004	Hennessy, John L., et al. <i>Computer Architecture: A Quantitative Approach</i> . India, Elsevier Science, 2006 (excerpt)
2005	Pentium™ Processor User’s Manual, Volume 3: Architecture and Programming Manual, 1994

POR, iii. Specifically, LzLabs complains about EX2004 and EX2005. EX2004 is a seminal textbook on Computer Architecture by John Hennessy and David Patterson , John L., professors of Electrical Engineering and Computer Science at Stanford and Berkeley, respectively. EX2005 is a user manual for an Intel Architecture processor (Pentium™ processor), which is explicitly referred to as an exemplary target architecture in the ’289 Patent Specification. *See* EX1001, 3:25-26. LzLabs does not dispute the evidence in Exhibits 2004 and 2005.

The POR also relies on extrinsic evidence that LzLabs introduced into the record, namely EX1005, EX1008, EX1009, and EX1024. LzLabs applies a double standard by relying on those exhibits to support their arguments but complains when disclosures from the same exhibits contradict LzLabs’ positions.

LzLabs also criticizes IBM for “sidestepping claim construction entirely in the POR.” Reply 18. But both parties, both parties’ experts, and the Board agreed that no construction is necessary to resolve the controversy. Petition, 17; POR, 26; EX1002, ¶74; EX2001, ¶32; DI, 7. Specifically, there is no need for claim construction to resolve the controversy, since Loderer expressly discloses—and both parties’ experts agree—that Loderer’s target machine does not provide “target machine condition codes”—regardless of any construction.

Besides, it is LzLabs who sidestepped claim construction entirely in the Petition and now proposes an untimely claim construction, when neither the Institution Decision nor the POR raised such an issue.

LzLabs also argues that “[a]ccording to PO, a condition code value that is directly calculated without branch instructions but stored in a ‘condition register’ or general-purpose register at the target machine cannot, by definition, be a ‘target machine condition code’ of the challenged claims.” Reply, 18. LzLabs again mischaracterizes IBM’s argument, which focuses on the calculation of “target machine condition codes”—not where they are stored. That a MIPS architecture, i.e., the Loderer target architecture, stores a bit pattern in a “condition register” or a “general-purpose register” further supports the fact that MIPS architectures do not provide “condition codes,” which is confirmed by Loderer’s disclosure (EX1005, 2 (“[S]imulating a condition code when porting hardware-specific program code

from a source hardware *target hardware that is not providing for condition code...*)), the Petition (Petition, 49 (“[T]ransforming hardware-specific program code with condition codes *to program code without condition codes...*”)), Mr. Jestice’s declaration (EX1002, ¶53 (“[S]imulating the execution of the mainframe code *on processors that do not support the condition codes...*”)), and from MIPS architecture literature (EX1009, 43).

LzLabs also argues:

PO’s argument is directly contrary to the teachings of the ’289 patent, which as explained above include storing “target machine condition codes” in general-purpose register rC. Unlike the challenged claims, other claims of the ’289 patent require the presence of “target central processing unit (CPU) **flag bits.**” EX1001, Claims 9, 15, and 20.

Reply, 19, n. 7.

First, as discussed above, LzLabs points to no support its contention that rC is a “general-purpose register.” Second, that argument is a red herring. As explained above in Section V, IBM’s argument relates to whether the target machine provides a “condition code;” not whether condition code values are, after being generated, stored in a particular location. As the Specification of the ’289 Patent explains “an instruction ...in the target machine’s architecture ... *sets one or more target CPU flag bits* and which *places a result in a storage location (first location, such as rC above)...*”). EX1001, 7:51-55.

LzLabs also argues:

PO contends Loderer does not disclose a “target machine condition code” because “the target architecture contemplated in Loderer is a MIPS architecture, which, as shown in Figure B.16 of EX2004, implements a ‘condition register’ instead of ‘condition codes.’” POR, 18-19.

Reply 22. For clarification purposes, IBM’s argument is that the MIPS architecture—which is the target architecture disclosed in Loderer—does not provide “condition codes,” which is consistent with Loderer’s disclosure—as well as MIPS architecture literature. *See* EX1005, 4 (“[A] target hardware (e.g. a Mips R4400 RISC machine).”); *id.*, 2 (“[A] target hardware that is not providing for condition code...”); EX1009, 43 (“The [MIPS RISC] R20002 architecture also provides no condition code register containing status bits such as Carry and Overflow.”).

LzLabs also argues that “PO offers no explanation why Loderer’s BM does not satisfy the ’289 patent’s broad description of condition codes as ‘several bits of data to indicate the outcome status of instruction execution.’”) Reply, 23, n.11 (citing EX1001, 1:18-23). LzLabs takes a sentence from the ’289 Patent Specification out of context to present an argument that amounts to a logical fallacy. Condition codes, e.g., CPU flag bits (EX1001, 7:54), can be several bits of data that indicate the outcome status of instruction execution, like “sign, carry,

overflow” (EX1001, 1:56), but not *any* several bits of data that indicate the outcome status of instruction execution are condition codes. Besides, LzLabs’ argument is contradicted by Loderer which distinguishes its “bit pattern” from “condition codes,” (EX1005, 2 (“[C]ondition code on the source hardware...”); *i.d.*, (“[A] bit pattern on the target hardware.”)) and even by LzLabs’ expert who testified that a “bit pattern” is not a “condition code.” *See* EX2003, 86:5-87:2. If LzLabs’ argument had any merit, neither Loderer’s disclosure nor Mr. Jestice would need to distinguish the two.

LzLabs finally argues that “[t]he challenged claims recite deriving ‘an *encoding* for a target machine condition code,’ and—like Loderer—the ’289 specification discloses bit patterns stored in a register representing emulated IBM architecture condition codes.” Reply, 23, n. 11. This is another red herring. All this disclosure suggests is that the condition code bits can be encoded such that they can be represented in an efficient manner, e.g., by two bits instead of three bits. For example, in the ESA/390 system, two bits of the Program-Status-Word correspond to the condition codes.

Program-Status-Word Format

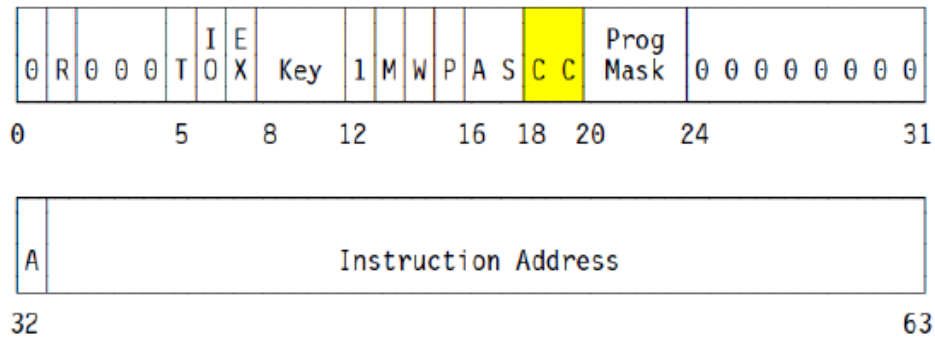


Figure 4-2. PSW Format

EX1008, 115.

However, the result of a comparison operation can result in three different—and mutually exclusive—outcomes. EX1008, 865 (“When the comparison ends, the condition code is set to 0, 1, or 2, depending on whether the first operand is equal to, less than, or greater than the second operand, respectively.”). Instead of allocating three condition codes (flag bits) that each would be set in case of each of the outcome, for efficiency ESA/390 encodes the condition codes into two bits, wherein the encoded values “0” (binary “00”), “1” (binary “01”), and “2” (binary “10”) corresponds to the three different outcomes. The fact that the claims recite “encoding” of target machine condition codes” supports—not negates—that condition codes are generated in the target machine.

VIII. LZLABS FAILS TO SHOW THAT IBM'S IPR POSITION IS INCONSISTENT WITH ITS LITIGATION POSITIONS

LzLabs points to four sentences taken out of context from IBM's complaint and preliminary infringement contentions (Reply, 24-25) to argue that these documents refer to the "target machine condition code" as "a representation of the source ISA condition code." LzLabs concludes that this "interpretation and mapping of 'target machine condition code' for infringement are thus consistent with Petitioner's mapping of 'target machine condition code' in Loderer." Reply, 25.

But LzLabs provides no basis or support for this conclusion. For example, LzLabs does not provide any evidence that the "representation of the source ISA condition code" is a "bit pattern" stored in a general-purpose register. Neither does LzLabs argue that the "target machine" in IBM's complaint is a target architecture that does not provide "condition codes." Rather, as the complaint notes "[t]o perform this emulation, the SDM first obtains, by *the 'target machine' with an x86 architecture*, a computer instruction that is intended for a 'source machine' having a z/Architecture." EX1027, ¶143. Accordingly, the "target machine" in the accused products is a target machine that provides condition codes. *See* EX2004, 8. Additionally, the complaint also notes that "target instructions would provide

information distinguishing between different IBM mainframe condition codes to ensure that *the correct condition code is generated.*” EX1027, ¶143.

IX. LZLABS FAILS TO SHOW THAT IBM’S ARGUMENT IS INCONSISTENT WITH THE CLAIM LANGUAGE AND THE INTRINSIC RECORD

LzLabs argues that “[t]he challenged claims do not limit the architecture of the target machine or require storing the code in a special type of storage.” Once again, LzLabs mischaracterizes IBM’s argument. IBM’s argument is not specific to a particular architecture, e.g., x86 architecture, nor is contingent on storing the code in a special type of storage. IBM’s argument is that a prior art reference that discloses a target machine that does not provide a “target machine condition code” cannot satisfy the claims that require a “target machine condition code.” POR, 26-33.

LzLabs also criticizes IBM for “relying on a table from an extrinsic textbook to distinguish Loderer.” Reply, 26. Tellingly, LzLabs does not criticize or rebut the evidence in the table from EX2004 (reproduced below), which confirms that the MIPS architecture—the target architecture in Loderer—does not implement condition codes. LzLabs merely criticizes the reliance on the evidence in the table. But the evidence—that MIPS architecture does not implement condition codes—is consistent with Loderer’s disclosure. See EX1005, 4 (“[A] target hardware (e. g. a

Mips R4400 RISC machine).”); *id.*, 2 (“[A] target hardware that is not providing for condition code...”).

Name	Examples	How condition is tested	Advantages	Disadvantages
Condition code (CC)	80x86, ARM, PowerPC, SPARC, SuperH	Tests special bits set by ALU operations, possibly under program control.	Sometimes condition is set for free.	CC is extra state. Condition codes constrain the ordering of instructions since they pass information from one instruction to a branch.
Condition register	Alpha, MIPS	Tests arbitrary register with the result of a comparison.	Simple.	Uses up a register.
Compare and branch	PA-RISC, VAX	Compare is part of the branch. Often compare is limited to subset.	One instruction rather than two for a branch.	May be too much work per instruction for pipelined execution.

Figure B.16 The major methods for evaluating branch conditions, their advantages, and their disadvantages. Although condition codes can be set by ALU operations that are needed for other purposes, measurements on programs show that this rarely happens. The major implementation problems with condition codes arise when the condition code is set by a large or haphazardly chosen subset of the instructions, rather than being controlled by a bit in the instruction. Computers with compare and branch often limit the set of compares and use a condition register for more complex compares. Often, different techniques are used for branches based on floating-point comparison versus those based on integer comparison. This dichotomy is reasonable since the number of branches that depend on floating-point comparisons is much smaller than the number depending on integer comparisons.

EX2004, 8.

X. CONCLUSION

For the reasons set forth above, LzLabs has failed to demonstrate the unpatentability of the challenged claims.

Case IPR2023-00274
U.S. Patent No. 8,713,289 B2

Dated: February 2, 2024

Respectfully submitted,

/Theodoros Konstantakopoulos/
Theodoros Konstantakopoulos
(Reg. No. 74,155)
tkonstantakopoulos@desmaraisllp.com
DESMARAIS LLP
230 Park Ave
New York, NY 10169
212-808-1060 (telephone)
212-351-3401 (facsimile)

Counsel for Patent Owner
International Business Machines Corp.

CERTIFICATION UNDER 37 C.F.R. § 42.24(d)

I hereby certify that this Patent Owner's Surreply has 4,797 words as counted by the word-processing system used to prepare this document, in compliance with 37 C.F.R. §42.24(c)(4).

Dated: February 2, 2024

Respectfully submitted,

/Theodoros Konstantakopoulos/
Theodoros Konstantakopoulos
(Reg. No. 74,155)
tkonstantakopoulos@desmaraisllp.com
DESMARAIS LLP
230 Park Ave
New York, NY 10169
212-808-1060 (telephone)
212-351-3401 (facsimile)

Counsel for Patent Owner
International Business Machines Corp.

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. § 42.6(e), the undersigned certifies that a complete copy of Patent Owner's Surreply was served on counsel of record for Petitioner by filing this document through P-TACTS and by sending this document via electronic mail to the following addresses:

Joseph D. Gray (Reg. No. 61,803)
Slayden Grubert Beard PLLC
401 Congress Ave., Suite 1650
Austin, Texas 78701
Tel. 512-402-3550
jgray@sgbfirm.com

Chris V. Ryan (Reg. No. 54,759)
RyanIPLaw, PC
1508 Bay Hill Drive
Austin, TX 78701-4078
chris.ryan@ryaniplaw.com

Tecuan Flores (Reg. No. 77,668)
Slayden Grubert Beard PLLC
401 Congress Ave., Suite 1650
Austin, Texas 78701
Tel. 512-402-3550
tflores@sgbfirm.com

litigation@sgbfirm.com

Dated: February 2, 2024

Respectfully submitted,

/Theodoros Konstantakopoulos/
Theodoros Konstantakopoulos
(Reg. No. 74,155)
tkonstantakopoulos@desmaraisllp.com
DESMARAIS LLP
230 Park Ave
New York, NY 10169
212-808-1060 (telephone)
212-351-3401 (facsimile)

*Counsel for Patent Owner
International Business Machines Corp.*