

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

LZLABS GMBH,
Petitioner,

v.

INTERNATIONAL BUSINESS MACHINES CORPORATION,
Patent Owner.

IPR2023-00274
Patent 8,713,289 B2

Before KEN B. BARRETT, MICHELLE N. WORMMEESTER, and
ARTHUR M. PESLAK, *Administrative Patent Judges*.

WORMMEESTER, *Administrative Patent Judge*.

DECISION
Granting Institution of *Inter Partes* Review
35 U.S.C. § 314

I. INTRODUCTION

LzLabs GmbH (“Petitioner”) filed a Petition (Paper 1, “Pet.”) requesting *inter partes* review of claims 1–4, 6, 10–12, 16, and 17 of U.S. Patent No. 8,713,289 B2 (Ex. 1001, “the ’289 patent”). International Business Machines Corporation (“Patent Owner”) did not file a preliminary response. We have authority under 35 U.S.C. § 314 and 37 C.F.R. § 42.4(a). Under 35 U.S.C. § 314(a), an *inter partes* review may not be instituted “unless . . . there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” For the reasons that follow, we institute an *inter partes* review as to all the challenged claims of the ’289 patent based on all the grounds presented.

II. BACKGROUND

A. *Related Proceedings*

Petitioner identifies one related federal district court case: *International Business Machines Corp. v. LzLabs GmbH*, No. 6:22-cv-00299 (W.D. Tex.). Pet. 1. Petitioner also identifies one related *inter partes* review proceeding.

B. *The ’289 Patent*

The ’289 patent describes emulating computer system architectures, where sequences of instructions produce valid condition code settings without the use of branching instructions from the target architecture. Ex. 1001, 1:8–14. The ’289 patent explains that executing various operations may result in the generation of condition codes, which are bits of data indicating the outcome status of instruction execution. *Id.* at 1:18–23.

The '289 patent provides “[a] classic example of an instruction which produces condition code changes upon execution,” namely, “the compare instruction which sets a condition code to ‘zero’ if the operands are equal, to ‘one’ if the first operand is strictly less than the second operand and to ‘two’ if the first operand is strictly greater than the second operand.” *Id.* at 1:29–34. The '289 patent explains that, “[i]n conventional approaches to the emulation of a compare instruction, the result is the construction of a sequence of instructions, which include three branch instructions.” *Id.* at 2:7–10. According to the '289 patent, branch instructions are not desirable because they may impede overall processing speed and waste computer resources. *Id.* at 2:13–32.

Thus, the '289 patent provides for the generation of “a sequence of target machine instructions which together operate to directly calculate target machine condition codes from carry, sign and overflow codes without the use of branch instructions from the target machine.” Ex. 1001, 3:12–18. To illustrate, Table II of the '289 patent is reproduced below.

TABLE II

[1]	sub_set_carry		rC, rA, rB	
[2]	set_bit_on_zero		rX, rC	
[3]	set_bit_on_not_zero		rC, rC	
[4]	add_to_carry_immed		rC, rC, 0	
[5]	sub		rC, rC, rX	

Instruction	rA > rB			rA = rB			rA < rB		
	Register Contents			Register Contents			Register Contents		
	rC	rX	Carry	rC	rX	Carry	rC	rX	Carry
[1]	rC > 0	N/A	1	rC = 0	N/A	1	rC < 0	N/A	0
[2]	rC > 0	0	1	rC = 0	1	1	rC < 0	0	0
[3]	1	0	1	0	1	1	1	0	0
[4]	2	0	N/A	1	1	N/A	1	0	N/A
[5]	2	0	N/A	0	1	N/A	1	0	N/A

Table II shows an example of a sequence of instructions for emulating condition code settings of a subject source instruction, namely, the z/Architecture Instruction called the Compare Logical operation. *Id.* at 5:28–35. The designations rA and rB represent the first and second operands of the z/Architecture instruction evaluated into a register. *Id.* at 5:20–21. The designation rX represents a temporary register used to hold intermediate results. *Id.* at 5:24–25. The designation rC represents the register that will hold the condition code value at the end of the sequence. *Id.* at 5:26–27.

As shown in Table II, after executing instruction [1] (sub__set__carry) in the target machine, the CPU carry bit is set to “1” where $rA \geq rB$, or to “0” where $rA < rB$. *Id.* at 5:59–61, 6:21–23. The value of rC accounts for the result of the subtraction, where the entries $rC > 0$, $rC = 0$, and $rC < 0$ indicate the resulting condition. *Id.* at 6:23–26. After executing instruction [2] (set_bit_on_zero), rC and the carry bit remain the same, but rX is set to “1” if rA and rB are the same based on the contents of rC. *Id.* at 6:27–32. After executing instruction [3] (set_bit_on_not_zero), the carry bit remains the same, but rC is set to “1” where rC is not zero. *Id.* at 6:33–37. The ’289 patent explains that, “at this point, rX is set up to provide discrimination information distinguishing equality from inequality,” and “this occurs outside of (that is, apart from) both rC and the CPU carry bit.” *Id.* at 6:37–58.

Executing instruction [4] (add_to_carry_immed) involves carrying out the operation $rC + \text{CPU carry bit} + 0$, where rC is set to “2” if $rA > rB$, or to “1” if $rA = rB$ or $rA < rB$. Ex. 1001, 6:59–67. The ’289 patent points out that, at this point, “there is provided an indication in rC for which the case

$r_A > r_B$ is distinguished from the other two cases ($r_A = r_B$ and $r_A < r_B$).” *Id.* at 6:67–7:2.

Instruction [5] (sub) is the last step, where r_X is subtracted from r_C to distinguish the case of $r_A = r_B$ from the case of $r_A < r_B$. Ex. 1001, 7:3–8. Thus, at the end of the instruction sequence, $r_C = 2$ if $r_A > r_B$, $r_C = 1$ if $r_A < r_B$, and $r_C = 0$ if $r_A = r_B$. *Id.* at 7:8–10.

C. Illustrative Claim

Petitioner challenges claims 1–4, 6, 10–12, 16, and 17 of the ’289 patent, where claims 1, 11, and 16 are independent. Claim 1, reproduced below, is illustrative of the claims under challenge.

1. A computer program product for emulating computer instructions from a source machine to produce sequences of instructions on a target machine, said computer program product comprising:

a non-transitory computer readable storage medium readable by a processor and storing instructions for execution by the processor for performing a method comprising:

obtaining by the target machine a computer instruction from the source machine, said source machine having a different architecture from said target machine; and

generating a sequence of target machine instructions which together operate to derive an encoding for a target machine condition code for the computer instruction, wherein the sequence of target machine instructions provides distinguishing information to distinguish between a plurality of possible outcomes for the target machine condition code, and directly calculates the target machine condition code without using branch instructions, the directly calculating comprising:

determining an intermediate condition code value, the intermediate condition code value being a provisional value for the target machine condition code and subject to change based on the distinguishing information; and

determining, based on the intermediate condition code value and based on the distinguishing information, the target machine condition code, wherein at least part of said distinguishing information is separate from the intermediate condition code value.

D. Asserted Grounds of Unpatentability

Petitioner challenges claims 1–4, 6, 10–12, 16, and 17 of the '289 patent on the following two grounds. Pet. 10, 18–76.

Claims Challenged	35 U.S.C. § ¹	References/Basis
1, 4, 6, 10, 11, 16	103	Loderer ²
1–4, 6, 10–12, 16, 17	103	Loderer, PoO, ³ Kane, ⁴ knowledge of a person of ordinary skill in the art

In support of its arguments, Petitioner relies on a Declaration of Ian Jestice (Exhibit 1003).

¹ The Leahy-Smith America Invents Act (“AIA”), Pub. L. No. 112-29, 125 Stat. 284 (2011), amended 35 U.S.C. § 103, effective March 16, 2013. Because the application from which the '289 patent issued was filed before this date, the pre-AIA version of § 103 applies.

² Loderer, EP Publ’n No. 0 843 256 A2, published May 20, 1998 (translated by Sun IP, May 11, 2022) (Ex. 1005).

³ IBM, *Enterprise Systems Architecture/390 Principles of Operation*, SA22-7201-08 (9th ed., June 2003) (Ex. 1008).

⁴ Gerry Kane, MIPS RISC ARCHITECTURE (1989) (Ex. 1009).

III. DISCUSSION

A. Claim Construction

In an *inter partes* review proceeding, we construe a claim of a patent “using the same claim construction standard that would be used to construe the claim in a civil action under 35 U.S.C. 282(b).” *See* 37 C.F.R.

§ 42.100(b). That standard involves construing claims in accordance with the ordinary and customary meaning of such claims as would have been understood by one of ordinary skill in the art and the prosecution history pertaining to the patent. *See id.*; *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312–14 (Fed. Cir. 2005) (en banc).

Petitioner does not propose any claim constructions. *See* Pet. 17. Instead, Petitioner asserts that “terms should be given ‘ordinary and customary meaning’ to those of skill in the art.” *Id.* For purposes of this Decision, we conclude that no claim term requires express interpretation to resolve any controversy in this proceeding. *See Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999).

B. Obviousness over Loderer

Petitioner asserts that claims 1, 4, 6, 10, 11, and 16 of the ’289 patent would have been obvious over Loderer. Pet. 18–48. For the reasons explained below, we determine that Petitioner has demonstrated a reasonable likelihood of prevailing on its asserted ground.

Before addressing Petitioner’s arguments, we start with an overview of the asserted prior art, Loderer.

1. Overview of Loderer

Loderer (Ex. 1005) is an English-language translation of European Patent Office Publication No. 0 843 256 A2 (Ex. 1006), which was published in the German language. Ex. 1005, 8 (page providing certification of accuracy of translation).

Loderer describes “a method of simulating a condition code (CC) when porting hardware-specific program code (PCU) from a source hardware (M1) to a target hardware (M2) where no hardware support of a condition code (CC) is provided.” Ex. 1005, code (57). Figure 1 of Loderer is reproduced below.

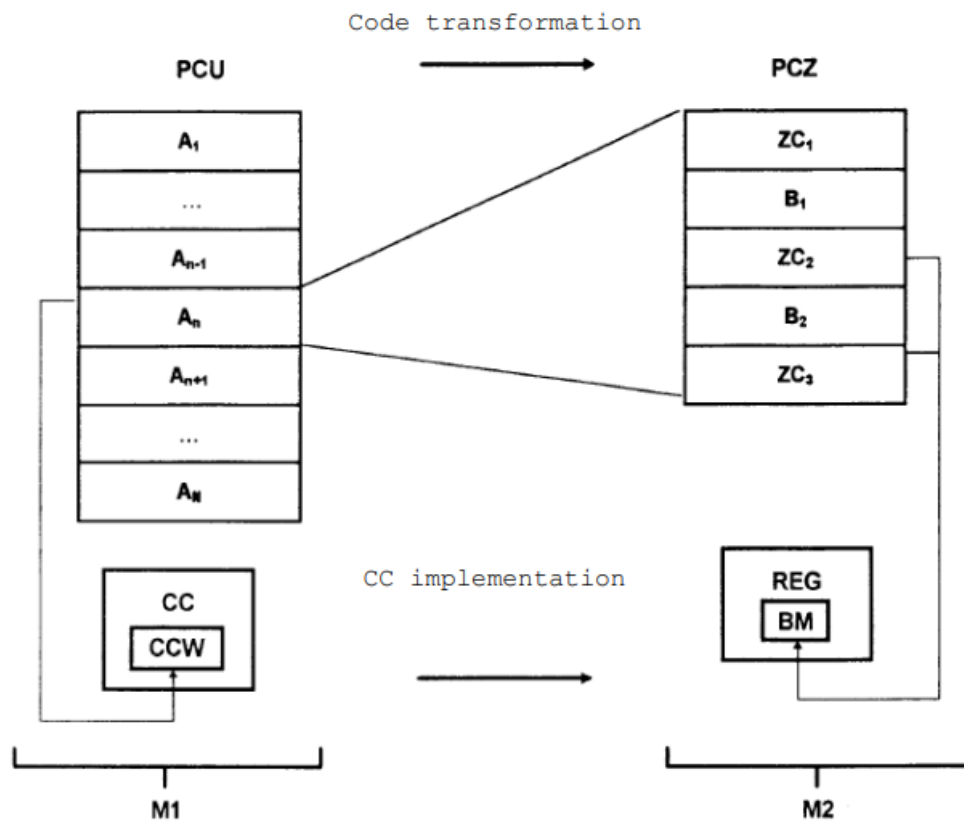


FIG. 1

Figure 1 of Loderer is a schematic diagram showing the basic principle of Loderer's method. *Id.* at col. 4, ¶ 7. Program code PCU, which includes

individual instructions A_1 through A_N , is to be ported from source hardware M1 to target hardware M2. *Id.* In a code transformation, program code PCU is converted into program code PCZ for target hardware M2, and each of instructions A_1 through A_N is implemented as “a command (or a semantically equivalent command sequence)” in program code PCZ. *Id.* at col. 4, ¶ 8. For example, instruction A_n in program code PCU is implemented as a command sequence in program code PCZ. *Id.* The command sequence includes two commands B_1 and B_2 for the semantic representation of instruction A_n and additional code having three subsections ZC_1 , ZC_2 , and ZC_3 . *Id.*

In one embodiment of Loderer, source hardware M1 includes an IBM/390 computer system, and target hardware M2 includes a Mips R4400 RISC machine. Ex. 1005, col. 5, ¶¶ 2–3. Various comparison instructions, which compare first operand OP1 with second operand OP2, are available on source hardware M1. *Id.* at col. 5, ¶ 3. To illustrate an example, Figure 2 of Loderer is reproduced below.

Comparison result	CCW	BM
OP1 equal to OP2	0	1000
OP1 < OP2	1	0100
OP1 > OP2	2	0010
Overflow	3	0001

FIG. 2

Figure 2 of Loderer shows the value assignment for the condition code in a comparison command. *Id.* at col. 4, ¶ 4. As shown in Figure 2, bit values BM are associated with values CCW of condition code CC when simulating the condition code on a target hardware. *Id.* at col. 5, ¶ 3. Bit

pattern BM is set to 1000 when operands OP 1 and OP2 are equal, to 0100 when operand OP 1 is less than operand OP2, to 0100 when operand OP 1 is greater than operand OP2, and to 0001 when there is a numeric overflow.

Id.

To further illustrate, Figure 4 of Loderer is reproduced below.

Command sequence in PCZ		a = 0	a > 0	a < 0
ZC ₁	Set BM to '0100'	BM = 0100	BM = 0100	BM = 0100
B1	If a = 0, set t ₁ = 1, otherwise t ₁ = 0	t ₁ = 1	t ₁ = 0	t ₁ = 0
ZC ₂	BM << t ₁	BM = 1000	BM = 0100	BM = 0100
B2	If a > 0, set t ₂ = 1, otherwise t ₂ = 0	t ₂ = 0	t ₂ = 1	t ₂ = 0
ZC ₃	BM >> t ₂	BM = 1000	BM = 0010	BM = 0100

FIG. 4

Figure 4 of Loderer shows a specific example of simulating an instruction for a source hardware on a target hardware. Ex. 1005, col. 4, ¶¶ 4, 6. In particular, Figure 4 shows a command sequence generated in program code PCZ for target hardware M2. *Id.* at col. 6, ¶ 2. At the first step in subsection ZC₁, bit pattern BM is set to 0100, where OP 1 < OP2 (i.e., a < 0), and therefore is associated with condition code value CCW = 1. *Id.* at col. 6, ¶ 3. Command B1 checks whether both operands OP 1 and OP2 are equal, and then sets register variable t₁ to 1 for equality or to 0 for inequality. *Id.* at col. 6, ¶ 4. At the next step in subsection ZC₂, shift operator << shifts bit pattern BM to the left by the number of bit positions (0 or 1) stored in register variable t₁. *Id.* Command B2 checks whether operand OP 1 is greater than operand OP2, and then sets register variable t₂ to 1 if a > 0 or to 0 if a < 0. *Id.* at col. 6, ¶ 5. At the final step in subsection ZC₃, shift operator >> shifts bit pattern BM to the right by the number of bit positions (0 or 1) stored in register variable t₂. *Id.* Loderer explains,

The bit pattern BM resulting in this step is uniquely associated with the value CCW of the condition code CC occurring after execution of the original comparison instruction on the source hardware M1 and can be evaluated by subsequent instructions in the program code PCZ instead of the original condition code CC of source hardware M1.

Id.

Loderer also notes that its method does not use branching commands. Ex. 1005, code (57), col. 2, ¶ 1.

2. Analysis

Turning to the claims, we address independent claims 1, 11, and 16 together. Claim 1 is directed to a “computer program product for emulating computer instructions from a source machine to produce sequences of instructions on a target machine.” Claim 11 is directed to a “computer system for emulating computer instructions from a source machine to produce sequences of instructions on a target machine.” Claim 16 is directed to a “method for emulating computer instructions from a source machine to produce sequences of instructions on a target machine.” Claims 11 and 16 recite similar limitations as claim 1. Petitioner relies primarily on its discussion of claim 1 for claims 11 and 16. Pet. 44–48. Accordingly, we focus on the language of claim 1 in particular. Our analysis of claim 1 applies to claims 11 and 16.

Claim 1 recites various limitations that Petitioner designates as limitations 1[a2] and 1[b] through 1[f]. We address these limitations in turn.

a. Limitation 1[a2]: “non-transitory computer readable storage medium”

Claim 1 recites “a non-transitory computer readable storage medium readable by a processor and storing instructions for execution by the processor for performing a method.” For this limitation, which Petitioner designates as limitation 1[a2], Petitioner directs us to where Loderer teaches,

When porting the program code of the source hardware, any instruction potentially influencing the value of the condition code is transformed by being simulated in the program code for the target hardware by means of a command generating a result value in the course of the execution of the program and without using branching commands.

Pet. 24 (quoting Ex. 1005, col. 2, ¶ 1). Petitioner asserts that an ordinarily skilled artisan “would understand this disclosure of program instructions running on the target machine discloses and suggests the program is stored in a non-transitory computer readable storage medium readable by the processor of the target machine.” *Id.* at 24–25. Petitioner also asserts that “Loderer discloses that the emulator, running on a MIPS processor, may simulate the generation of condition codes for emulation of IBM/390 hardware,” and that an ordinarily skilled artisan “would understand that a simulator emulating a complex instruction set computer (CISC) processor, such as one that implements the IBM/390 architecture, on a reduced instruction set computer (RISC) processor such as a MIPS processor, uses software read from non-transitory memory.” *Id.* at 25. Petitioner relies on the declaration testimony of Mr. Jestice. *Id.* at 24–25 (citing Ex. 1003 ¶¶ 89–91).

Based on Petitioner’s argument and evidence, we are persuaded that Petitioner has shown sufficiently for purposes of this Decision that Loderer

teaches the non-transitory computer readable storage medium of limitation 1[a2].

b. Limitation 1[b]: “obtaining . . . a computer instruction”

In connection with the recited method of limitation 1[a2], claim 1 recites the step of “obtaining by the target machine a computer instruction from the source machine, said source machine having a different architecture from said target machine.” For this limitation, which Petitioner designates as limitation 1[b], Petitioner asserts that “Loderer discloses a source machine having architecture M1 from which the instructions originate, and a target machine having different architecture M2 that is simulating execution of the source program.” Pet. 25. As support, Petitioner directs us to where Loderer teaches that its “invention relates to a method for simulating a condition code when porting hardware-specific program code from a source hardware to a target hardware, in which no hardware support of a condition code is provided.” *Id.* (quoting Ex. 1005, col. 1, ¶ 1); *see also id.* at 26 (citing Ex. 1005, col. 1, ¶ 3). Petitioner further directs us to Loderer’s Figure 1, and asserts that “[i]nstructions A₁ . . . A_N from the [source] machine having architecture M1 are emulated on the [target] machine having architecture M2, where each instruction A_n is replaced with a sequence of instructions.” *Id.* at 26 (citing Ex. 1005, col. 4, ¶ 7, Fig. 1). Petitioner contends that “[t]he translation steps in Loderer occur on the target machine . . . , confirming the target machine obtains the source machine instructions because the target machine (e.g., MIPS RISC machine) is simulating the execution of the IBM/390 instructions of the source machine.” *Id.* at 27 (citing Ex. 1005, col. 2, ¶¶ 1–2, col. 4, ¶ 8). Petitioner

relies on the declaration testimony of Mr. Jestice. *Id.* at 25–27 (citing Ex. 1003 ¶¶ 92–96).

Based on Petitioner’s argument and evidence, we are persuaded that Petitioner has shown sufficiently for purposes of this Decision that Loderer teaches the obtaining step of limitation 1[b].

c. Limitation 1[c]: “generating a sequence of target machine instructions”

Claim 1 further recites the step of “generating a sequence of target machine instructions which together operate to derive an encoding for a target machine condition code for the computer instruction, wherein the sequence of target machine instructions provides distinguishing information to distinguish between a plurality of possible outcomes for the target machine condition code.” For this limitation, which Petitioner designates as limitation 1[c], Petitioner provides an annotated version of Figure 1 of Loderer, which is reproduced below. Pet. 28.

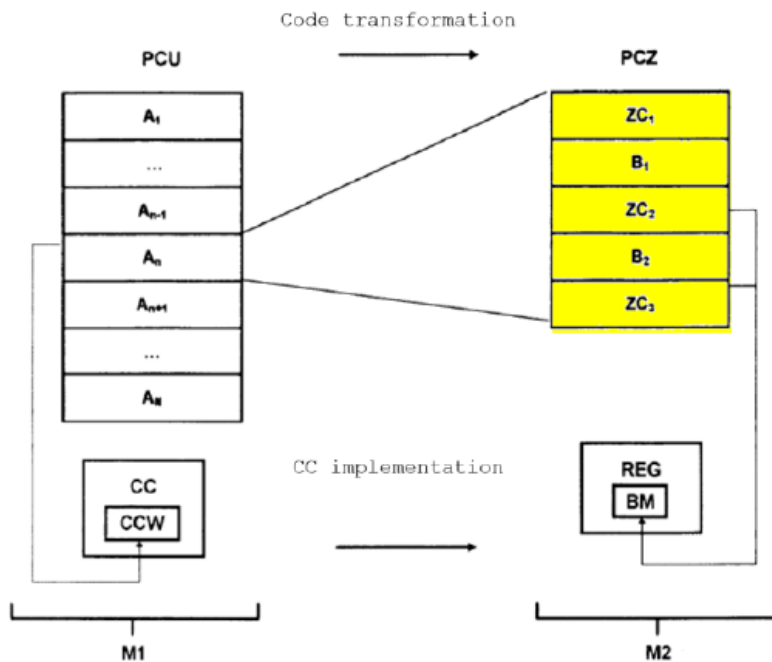


FIG. 1

Figure 1 of Loderer, as annotated by Petitioner, is a schematic diagram showing the basic principle of Loderer's method. *See* Ex. 1005, col. 4, ¶ 7. Petitioner asserts that "Loderer's Fig. 1 depicts a sequence of target machine instructions (yellow) . . . for the source instruction A_n ." Pet. 28 (citing Ex. 1005, col. 4, ¶ 8).

Petitioner further asserts that the same sequence of instructions is shown in Figure 4 of Loderer. Pet. 28. To illustrate, Petitioner also provides an annotated version of Loderer's Figure 4, reproduced below. *Id.* at 29.

	Command sequence in PCZ	$a = 0$	$a > 0$	$a < 0$
ZC ₁	Set BM to '0100'	BM = 0100	BM = 0100	BM = 0100
B1	If $a = 0$, set $t_1 = 1$, otherwise $t_1 = 0$	$t_1 = 1$	$t_1 = 0$	$t_1 = 0$
ZC ₂	BM $\ll t_1$	BM = 1000	BM = 0100	BM = 0100
B2	If $a > 0$, set $t_2 = 1$, otherwise $t_2 = 0$	$t_2 = 0$	$t_2 = 1$	$t_2 = 0$
ZC ₃	BM $\gg t_2$	BM = 1000	BM = 0010	BM = 0100

FIG. 4

Loderer's Figure 4, as annotated by Petitioner, shows a specific example of simulating an instruction for a source hardware on a target hardware. *See* Ex. 1005, col. 4, ¶¶ 4, 6. According to Petitioner, Loderer's "target machine instructions illustrated in Figs. 1 and 4 (labeled 'Command sequence in PCZ' [in Figure 4]) operate together to derive a representation for the target machine condition code that is stored in register BM." Pet. 29; *see also id.* at 28 ("Loderer's Fig. 1 depicts a sequence of target machine instructions (yellow) that derive an encoding for a target machine condition code in register BM for the source instruction A_n ."). Petitioner asserts that "[t]he values of the BM register are a target encoding of the condition codes because they are a machine representation of the source condition codes that directly map to the corresponding values on the IBM/390 computer (i.e., the

source machine).” *Id.* at 29. As support, Petitioner directs us to where Loderer describes “uniquely associat[ing] each possible value (CCW) of the condition code (CC) on the source hardware (M1) with a bit pattern (BM) on the target hardware (M2),” and providing “a memory area (REG) . . . on the target hardware (M2)” in order “[t]o store the bit pattern (BM) associated with the respective current value (CCW).” Ex. 1005, code (57) (cited by Pet. 29); *see also id.* at Fig. 2 (cited by Pet. 30). Petitioner relies on the declaration testimony of Mr. Jestice. Pet. 29–30 (citing Ex. 1003 ¶¶ 99–100, 102).

Referring specifically to its annotated version of Loderer’s Figure 4, Petitioner adds that “[c]ommands B1 and B2 provide distinguishing information stored in registers t1 and t2 (blue), for use by commands ZC₁ and ZC₃.” Pet. 28; *see also id.* at 30–32. As support, Petitioner directs us to where Loderer explains that command B1 “serves to check the equality of both operands (OP1 = a, OP2 = 0),” and that “[t]he result of this first individual query is stored as the result value ‘1’ (for equality) or ‘0’ (for inequality) in the register variable t₁.” Ex. 1005, col. 6, ¶ 4 (cited by Pet. 28, 31–32). Loderer further explains that the step at subsection ZC₂ shifts bit pattern BM to the left by the number of bit positions stored in register variable t₁. *Id.* Loderer also explains that “subsequent command B2 . . . checks whether the first operand (OP1 = a) is greater than the second one (OP2 = 0),” and that “[t]he result of this second individual query is stored as result value ‘1’ (for a > 0) or ‘0’ (for a < 0) in the register variable t₂.” *Id.* at col. 6, ¶ 5 (cited by Pet. 28, 31–32); *see also id.* at Fig. 4 (“If a > 0, set t₂ = 1, otherwise t₂ = 0”). Loderer explains that the step at subsection ZC₃ shifts the bit pattern BM to the right by the number of bit positions stored in the

register variable t_2 . *Id.* at col., 6 ¶ 5. According to Petitioner, “Loderer thus discloses distinguishing information (t_1 and t_2) to distinguish between a plurality of outcomes (e.g., $a=0$, $a>1$, and $a<1$).” Pet. 32. Petitioner relies on the declaration testimony of Mr. Jestice. *Id.* at 30–32 (citing Ex. 103 ¶¶ 97–108).

Based on Petitioner’s argument and evidence, we are persuaded that Petitioner has shown sufficiently for purposes of this Decision that Loderer teaches the generating step of limitation 1[c].

d. Limitation 1[d]: “directly calculates the target machine condition code”

In connection with the generating step of limitation 1[c], claim 1 requires that the recited sequence of target machine instructions “directly calculates the target machine condition code without using branch instructions.” For this “directly calculating” limitation, which Petitioner designates as limitation 1[d], Petitioner directs us to where Loderer teaches that “any instruction potentially influencing the value of the condition code is transformed by being simulated in the program code for the target hardware by means of a command generating a result value in the course of the execution of the program **and without using branching commands.**” Pet. 33 (quoting Ex. 1005, col. 2, ¶ 1 (emphasis added by Petitioner)). Petitioner also directs us to its annotated version of Loderer’s Figure 4, which is reproduced again below. *Id.* at 34.

	Command sequence in PCZ	a = 0	a > 0	a < 0
ZC ₁	Set BM to '0100'	BM = 0100	BM = 0100	BM = 0100
B1	If a = 0, set t ₁ = 1, otherwise t ₁ = 0	t ₁ = 1	t ₁ = 0	t ₁ = 0
ZC ₂	BM << t ₁	BM = 1000	BM = 0100	BM = 0100
B2	If a > 0, set t ₂ = 1, otherwise t ₂ = 0	t ₂ = 0	t ₂ = 1	t ₂ = 0
ZC ₃	BM >> t ₂	BM = 1000	BM = 0010	BM = 0100

FIG. 4

As mentioned above, this version of Figure 4 shows a specific example of simulating an instruction for a source hardware on a target hardware. *See* Ex. 1005, col. 4, ¶¶ 4, 6. Petitioner asserts that Loderer's Figure 4 "shows instruction ZC₁ sets an initial intermediate setting of BM=0100, and that, depending on the values of the distinguishing information generated by commands B1 and B2 and stored in registers t₁ and t₂ (blue), the shift instructions ZC_[2] and ZC_[3] directly calculate the values of the condition codes without branches." Pet. 34 (citing Ex. 1005, col. 6, ¶¶ 2–5). Petitioner relies on the declaration testimony of Mr. Jestice. *Id.* (citing Ex. 1003 ¶¶ 109–112).

Based on Petitioner's argument and evidence, we are persuaded that Petitioner has shown sufficiently for purposes of this Decision that Loderer teaches the subject matter of limitation 1[d].

e. Limitation 1[e]: "determining an intermediate condition code value"

In connection with limitation 1[d], claim 1 requires that "directly calculating" comprises "determining an intermediate condition code value, the intermediate condition code value being a provisional value for the target machine condition code and subject to change based on the distinguishing information." For this limitation, which Petitioner designates as

limitation 1[e], Petitioner provides another annotated version of Figure 4 of Loderer, which is reproduced below. Pet. 35.

Command sequence in PCZ		a = 0	a > 0	a < 0
ZC ₁	Set BM to '0100'	BM = 0100	BM = 0100	BM = 0100
B1	If a = 0, set t ₁ = 1, otherwise t ₁ = 0	t ₁ = 1	t ₁ = 0	t ₁ = 0
ZC ₂	BM << t ₁	BM = 1000	BM = 0100	BM = 0100
B2	If a > 0, set t ₂ = 1, otherwise t ₂ = 0	t ₂ = 0	t ₂ = 1	t ₂ = 0
ZC ₃	BM >> t ₂	BM = 1000	BM = 0010	BM = 0100

FIG. 4

Figure 4 of Loderer, as annotated by Petitioner, shows a specific example of simulating an instruction for a source hardware on a target hardware. *See* Ex. 1005, col. 4, ¶¶ 4, 6. Petitioner asserts that “[t]he first intermediate condition code value, BM=0100, is set by ZC₁, the first command in Figure 4.” Pet. 35 (citing Ex. 1005, col. 6, ¶ 3). Petitioner further asserts that “[t]his provisional value in the BM register is subject to change in later instructions (e.g., ZC₂ and ZC₃) based on the distinguishing information stored in t₁ and t₂.” *Id.* As an example, Petitioner asserts that “ZC₂ may change the first provisional value in register BM to another value by shifting BM based on the value of t₁ (distinguishing information),” where “[t]he value of t₁ distinguishes between the case of a=0 and the cases in which a is not equal to zero.” *Id.* (citing Ex. 1005, col. 6, ¶ 4). More specifically, “[i]f t₁=1, ZC₂ shifts the bits in BM to the left by one, thereby distinguishing the a=0 case.” *Id.* (citing Ex. 1005, col. 6, ¶ 4).

Based on Petitioner’s argument and evidence, we are persuaded that Petitioner has shown sufficiently for purposes of this Decision that Loderer teaches the determining step of limitation 1[e].

f. Limitation 1[f]: “determining . . . the target machine condition code”

Lastly, claim 1 requires that “directly calculating” further comprises “determining, based on the intermediate condition code value and based on the distinguishing information, the target machine condition code, wherein at least part of said distinguishing information is separate from the intermediate condition code value.” For this limitation, which Petitioner designates as limitation 1[f], Petitioner cross-references its analysis for limitations 1[c] and 1[d], and asserts that “Loderer discloses calculating a target machine condition code (e.g., BM) based on the intermediate condition code value (e.g., intermediate value stored in BM), and the distinguishing information (e.g., values stored in t_1 and t_2).” Pet. 37 (citing Ex. 1005, col. 6, ¶¶ 2–5). Petitioner additionally asserts that “Loderer states that the condition code is stored in its own ‘separate register’ that ‘is provided on the target hardware,’” and that “[t]he distinguishing information (t_1 and t_2)[] is stored in ‘two register variables.’” *Id.* at 38 (quoting Ex. 1005, col. 2, ¶ 2, col. 6, ¶ 2; citing Figs. 1, 4). Thus, Petitioner contends, “the distinguishing information stored in registers t_1 and t_2 is separate from the intermediate condition code value in BM.” *Id.* Petitioner relies on the declaration testimony of Mr. Jestice. *Id.* (citing Ex. 1003 ¶¶ 118–121).

Based on Petitioner’s argument and evidence, we are persuaded that Petitioner has shown sufficiently for purposes of this Decision that Loderer teaches the determining step of limitation 1[f].

3. Summary

In view of the foregoing, we determine that Petitioner has demonstrated a reasonable likelihood of showing that independent claims 1,

11, and 16 would have been obvious over Loderer. Claims 4, 6, and 10 depend from claim 1. Having reviewed Petitioner’s arguments asserting that claims 4, 6, and 10 would have been obvious over Loderer, (*see* Pet. 38–44), we also determine that Petitioner has demonstrated a reasonable likelihood of prevailing on its assertion as to these dependent claims.

C. Obviousness over Loderer, PoO, Kane, and the Knowledge of a Person of Ordinary Skill in the Art

Petitioner asserts that claims 1–4, 6, 10–12, 16, and 17 of the ’289 patent would have been obvious over Loderer, PoO, Kane, and the knowledge of a person of ordinary skill in the art. Pet. 48–76.

As discussed above, Petitioner has already demonstrated a reasonable likelihood that it will prevail on its asserted ground of obviousness over Loderer. *See supra* Part III.B. Accordingly, we institute trial on all challenged claims and all asserted grounds of unpatentability, including Petitioner’s asserted ground of obviousness over Loderer, PoO, Kane, and the knowledge of a person of ordinary skill in the art. *See SAS Inst. Inc. v. Iancu*, 138 S. Ct. 1348, 1353 (2018) (“The agency cannot curate the claims at issue but must decide them all.”); 37 C.F.R. § 42.108(a) (“When instituting . . . review, the Board will authorize the review to proceed on all of the challenged claims and on all grounds of unpatentability asserted for each claim.”).

IV. CONCLUSION

For the foregoing reasons, we determine that Petitioner has demonstrated on the record before us a reasonable likelihood that it would

prevail in showing at least one of the challenged claims of the '289 patent is unpatentable. We institute trial on all challenged claims and all asserted grounds of unpatentability. *See SAS Inst.*, 138 S. Ct. at 1353. We have not, however, made a final determination with respect to the unpatentability of any of the challenged claims of the '289 patent.

V. ORDER

For the reasons given, it is

ORDERED that *inter partes* review is instituted as to all challenged claims of the '289 patent, namely, claims 1–4, 6, 10–12, 16, and 17, based on all grounds presented in the Petition; and

FURTHER ORDERED that, pursuant to 35 U.S.C. § 314(c) and 37 C.F.R. § 42.4, notice is hereby given of the institution of a trial; the trial will commence on the entry date of this Decision.

IPR2023-00274
Patent 8,713,289 B2

PETITIONER:

Joseph Gray
jgray@sgbfirm.com

Christopher Ryan
Chris.ryan@ryaniplaw.com

Tecuan Flores
tflores@sgbfirm.com

PATENT OWNER:

Theodoros Konstantakopoulos
tkonstantakopoulos@desmaraisllp.com

Yung-Hoon Ha
yha@desmaraisllp.com