United States District Court,
N.D. California, San Jose Division.

NAZOMI COMMUNICATIONS, INC,
Plaintiff.
v.
ARM HOLDINGS, PLC, et. al,
Defendants.

No. C 02-2521 JF (RS)

Sept. 6, 2006.

Thomas J. Friel, Jr., Cooley Godward LLP, San Francisco, CA, Clifford Liu, Jeffrey S. Karr, Sayuri K. Sharper, Cooley Godward LLP, Bernard C. Shek, Skadden Arps Slate Meagher & Flom, Brandon D. Baum, Mayer Brown Rowe & Maw LLP, Palo Alto, CA, for Plaintiff.

Andrew Y. Piatnicia, Howrey LLP, East Palo Alto, CA, Ethan B. Andelman, Howrey LLP, San Francisco, CA, for Defendants.

ORDER CONSTRUING CLAIM TERM "INSTRUCTION" AS USED IN UNITED STATES PATENT NO. 6,332,215 B1

JEREMY FOGEL**, District Judge.**

The Court construes the claim term "instruction" as used in United States Patent No. 6,332,215 B1 ("the '215 patent") as follows:

I. BACKGROUND

Plaintiff Nazomi Communications, Inc. ("Nazomi") owns the '215 patent, which claims a system that speeds up the processing of Java bytecodes. Java is a programming language developed by Sun Microsystems. In conventional programming, the source code of a program is put through a compiler that is associated with a particular type of system. The compiler translates the source code into machine code, or processor instructions, that will run only on that type of system. For example, if the source code is compiled on an Intel-based system, the resulting processor instructions (machine code) will run only on an Intel-based system. The instructions created for a particular system are said to be "native" to that system's processor, meaning that the processor is able to "decode" the instructions and carry out the desired functions. If a user wishes to run the same program on another system, the user must go back to the original source code and recompile the program for the new system.

Java works differently-instead of generating machine code for a particular type of system, the Java compiler generates "bytecodes," that is, instructions that are not specific to any processor. Thus, a Java program can

run on any processor if an appropriate "bytecode interpreter" is used. A bytecode interpreter translates Java bytecodes into native instructions for a particular processor. There are difficulties in performing this translation, because Java bytecodes are stored in "stack-based" memory, whereas most processors use "register-based" memory. Stack-based memory stores information on a last-in, first-out basis. "This approach is analogous to a stack of papers in an inbox. To access a paper at the bottom of the stack, a reader must first remove all of the papers above it." Nazomi Communications, Inc. v.. Arm Holdings, PLC, 403 F.3d 1364, 1366 (Fed.Cir.2005). In contrast, register-based memory stores and retrieves data "according to the exact location of each data item, much like an arrangement of post office boxes ... the reader simply identifies and finds the 'box' that contains the desired data, which can be instantly retrieved." *Id*.

Translation of Java bytecodes into a form usable by a register-based system, i.e., into native instructions, may be performed by hardware, software, or a combination of the two. One prior art approach was to use software to create a Java Virtual Machine ("JVM") capable of performing the necessary translation. However, this approach slowed overall execution speed. Other prior art approaches attempted to speed up the JVM by use of special interpreters that required significant additional memory and/or power. The system claimed by the '215 patent uses a hardware accelerator to provide a faster, lower-cost way to translate Java bytecodes into native instructions. FN1

FN1. The '215 patent states that the term "Java" is intended to cover successor programming languages or other programming languages that use generic instructions such as bytecodes. For ease of discussion, this order refers only to "Java."

Nazomi brought the instant patent infringement action against Defendants ARM Holdings, PLC, ARM Limited and ARM, Inc. (collectively "ARM") on May 23, 2002. Nazomi alleged that two lines of ARM's products infringe the '215 patent, the Revision 2 Jazelle line and the Revision 3 Jazelle line. ARM moved for partial summary judgment as to the Revision 3 Jazelle line, contending that while they process Java bytecodes at least in part in hardware, the Revision 3 products do *not* translate Java bytecodes into native instructions as disclosed by the '215 patent. ARM contended that the Revision 3 products in essence are specialized processors for which Java bytecodes *are* the native instructions, such that no translation is required.

Construction of the term "instruction" was the primary focus of ARM's summary judgment motion. FN2 Modern computers typically have a central processing unit ("CPU") that includes circuitry to fetch, decode and execute each operation specified by the instructions provided to it. The ' 215 patent describes this circuitry as part of a multi-stage pipeline: the "fetch" stage retrieves each successive instruction; the instruction then is passed to the "decode" stage, which recognizes each instruction and generates a control signal indicating that a particular operation or function is to be performed; and the "execute logic" stage, consisting of adders, shifters and multipliers, performs the operation or function. *See* ' 215 patent, col. 5, ll. 12-17 and fig.3. ARM argued that as used in the ' 215 patent, an instruction is a command provided to the CPU as input to the decode stage. ARM distinguished these instructions provided prior to the decode stage from the "control signals" generated by the decoder. In opposition to the motion, Nazomi argued that an instruction is *any* command that specifies or causes performance of an operation or function, including the control signals generated by the processor's decoder.

FN2. The Court permitted the parties to present their arguments as to the appropriate construction of the term "instruction" in the context of ARM's motion for partial summary judgment; the Court did not hold a

claim construction hearing prior to ruling on ARM's motion.

The Court did not wholly adopt ARM's proposed construction, but concluded that the '215 patent discloses "a hardware unit or subunit that converts stack-based instructions into the register-based instructions prior to the processing of those instructions by the processor in the so-called 'decode stage.' " Based upon this construction, the Court granted ARM's motion for partial summary judgment. The parties then stipulated to Nazomi's dismissal of its claims of infringement by ARM's Revision 2 Jazelle products and to ARM's dismissal of its counterclaim for declaratory judgment.

On April 11, 2005, the Federal Circuit vacated this Court's grant of partial summary judgment for ARM and remanded for further proceedings after concluding that this Court had failed to construe the term instruction in sufficient detail for appellate review. This Court set a claim construction hearing and solicited briefing from the parties. The parties have confined their arguments to the term "instruction," but have reserved their rights to brief and argue other terms in the event construction of the term "instruction" does not result in disposition of the litigation.

II. LEGAL STANDARD

Claim construction analysis begins with the words of the claim. Nystrom v. Trex Co., Inc., 424 F.3d 1136, 1142 (Fed.Cir.2005). A particular claim term generally is given its ordinary and customary meaning, that is, "the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention." Phillips v. AWH Corp., 415 F.3d 1303, 1312-13 (Fed.Cir.2005). In determining how one of ordinary skill in the art would define a claim term, the Court looks to " 'the words of the claims themselves, the remainder of the specification, the prosecution history, and extrinsic evidence concerning relevant scientific principles, the meaning of technical terms, and the state of the art.' " Id. at 1314 (quoting Innova/Pure Water, Inc. v. Safari Water Filtration Systems, Inc., 381 F.3d 1111, 1116 (Fed.Cir.2004)). The Court therefore does not determine the meaning that one skilled in the art would attribute to a particular term if he or she were operating in a vacuum, but rather the meaning that one skilled in the art would find if he or she were to view the claim term in light of the entire intrinsic record. Nystrom, 424 F.3d at 1142.

The specification may reveal that the inventor has given a claim term a special meaning that differs from the meaning it otherwise would possess. Phillips, 415 F.3d at 1316. "In such cases, the inventor's lexicography governs." *Id*. Alternatively, "the specification may reveal an intentional disclaimer, or disavowal, of claim scope by the inventor." *Id*. The inventor's intentions as revealed in the specification are dispositive. *Id*. The specification is considered "the single best guide to the meaning of a disputed term." Phillips, 415 F.3d at 1315 (quoting Vitronics Corp. v. Conceptronic, Inc., 90 F.3d 1576, 1582 (Fed.Cir.1996)).

The prosecution history likewise provides guidance as to how the Patent and Trademark Office and the inventor understood the invention, and whether the inventor limited the invention during the course of prosecution. Phillips, 415 F.3d at 1317.

As noted above, extrinsic evidence such as technical dictionaries and expert testimony may be useful to educate the Court as to the field if the invention generally and the likely meaning that would be attributed to a particular claim term by one of ordinary skill in the art. *See* Phillips, 415 F.3d at 1318. However, "conclusory, unsupported assertions by experts as to the definition of a claim term are not useful." *Id*. Moreover, technical dictionaries and treatises may provide multiple definitions so expansive as to extend

patent protection beyond what properly should be afforded. *Id*. at 1321. For these and other reasons, extrinsic evidence is considered to be a less reliable guide to claim construction than are the patent and its prosecution history. *Id*. at 1318.

III. DISCUSSION

The '215 patent contains seventy claims, including independent claims 1, 33, 36, 39, 69 and 70. Claim 1 is representative, and is set forth in full as follows with the disputed term "instructions" highlighted in bold:

1. A system comprising:

a central processing unit having a register file, the central processing unit adapted to execute register-based **instructions;** and
a hardware unit associated with the central processing unit, the hardware unit adapted to convert stack-based **instructions** into register-based **instructions,** wherein a portion of the operand stack is stored in the register file of the central processing unit and wherein the hardware unit is adapted to produce at least one of overflow or underflow indications for the portion of the operand stack stored in the register file, wherein the hardware unit is adapted to swap parts of the operand stack in and out of the register file from a memory, the system including an indication of the depth of the portion of operand stack, wherein a overflow or underflow produces an operand transfer between the register file in the central processing unit and memory.

The essence of the patented invention is a hardware accelerator that quickly and efficiently translates Java bytecodes (stack-based instructions) into native instructions (register-based instructions) for a CPU. *See* '215 patent, col. 2, ll. 3-6. Claim 1 describes this invention in a fairly straightforward manner, specifying a system comprising (1) a CPU that executes register-based instructions and (2) an associated hardware unit that converts stack-based instructions into register-based instructions. The parties dispute the scope of the term "instruction."
ARM proposes the following construction: "an element of an instruction set architecture, which specifies the interface between computer programs and a processor, that element having an operation code and zero or more operand specifiers, such that the processor can recognize the instruction and perform the specified operation." As the Court understands it, the instruction set architecture ("ISA") of a particular system is the universe of native instructions recognizable at the input of the CPU's decode stage. An immediate problem with ARM's proposed construction presents itself, because Claim 1 refers not only to *register-based* instructions, which would be elements of the system's ISA, but also to *stack-based instructions,* which would not be recognizable to the CPU given the explicit statement that the CPU executes register-based instructions. This definitional glitch notwithstanding, the thrust of ARM's contention clearly is that the "instructions" described by the claims are *inputs* to the CPU prior to the decode stage.

Nazomi proposes a much broader construction: "a command that specifies or causes an operation or function to be performed." Under Nazomi's proposed construction, the "instructions" described by the claims could be either inputs to the CPU prior to the decode stage *or* outputs of the CPU, i.e., the control signals that cause the specified operation or function to be performed. Nazomi refers to these post-decode control signals as "low-level instructions" or "microinstructions."

A. Claim Language

Applying the relevant legal standards, the Court first must determine the meaning that one of ordinary skill in the art would give the term "instruction" as used in the patent claims. The parties agree that a person of

ordinary skill in the art would have a Bachelor of Science or equivalent degree in electrical engineering, computer engineering or computer science, and two or three years of experience working with computer architecture. Nazomi initially expressed concern that ARM was attempting to limit the person's work experience to programming or other experience with the external architecture of CPUs; however, ARM has clarified that it agrees with Nazomi that one of ordinary skill in the art would be knowledgeable regarding the internal architecture of CPUs as well. ARM initially argued that one of ordinary skill in the art would have at least one year of experience with Java. However, in light of Nazomi's objection to this requirement, ARM has agreed that for purposes of construing the term "instruction" Java experience is not required. Accordingly, the parties are in agreement as to the qualifications of one of ordinary skill in the art.

Predictably, both parties present extrinsic evidence to support their arguments as to how the above-described person of ordinary skill in the art would understand the term "instruction." For example, Nazomi's expert, Dr. Yalamanchili, states that "instruction" is a general term that is used to identify commands at multiple levels of the computer architecture. Yalamanchili Decl. para.para. 25, 27;Yalamanchili Suppl. Decl. para.para. 5, 8. Dr. Yalamanchili states that qualifiers such as "micro" are used before the term "instruction" (as in "microinstruction") to distinguish instructions at different levels of abstraction. Id. at para. 8. ARM attacks Dr. Yalamanchili's opinion, citing deposition testimony that when teaching a class on computer basics at the Georgia Institute of Technology, Dr. Yalamanchili distinguished between "instructions" as input to the CPU's decoder and "microinstructions" as output of the decoder. Yalamanchili Depo. 111:7-112:7, 169:18-21. ARM also cites to various texts in support of its position that the term "instruction" generally refers to an element of the ISA.

As the Federal Circuit recently noted, "there is a virtually unbounded universe of potential extrinsic evidence of some marginal relevance that could be brought to bear on any claim construction question. In the course of litigation, each party will naturally choose the pieces of extrinsic evidence most favorable to its cause, leaving the court with the considerable task of filtering the useful extrinsic evidence from the fluff." Phillips, 415 F.3d at 1318. The conflicting evidence submitted by the parties is not particularly useful in aiding the Court to determine how one of ordinary skill in the art would understand the term "instruction." Based upon all of the evidence in the record, this Court concludes that, at least in the abstract, one of ordinary skill in the art might understand the term "instruction" to apply to multiple levels of computer architecture. The *relevant* question, however, is how a person of ordinary skill in the art would understand the term "instruction" in the context of *this patent*. *See* Nystrom, 424 F.3d at 1142 (holding that "[t]he person of ordinary skill in the art views the claim term in light of the entire intrinsic record").

The problem addressed by the patent is the slow and inefficient manner in which the prior art translated Java bytecodes into native instructions that can be executed by a CPU. *See* '215 patent, col. 1, ll. 38-67. The essence of the patented invention is a hardware accelerator that speeds up this translation. *See* id., col. 2, ll. 1-7. *All* of the claims describe a hardware unit that translates stack-based instructions, e.g., Java bytecodes, into register-based instructions. Given this context, the Court concludes that a person of ordinary skill in the art would understand the term "instructions" to refer to Java bytecodes or the native instructions to which those bytecodes are translated. *See* Nystrom, 424 F.3d at 1145 (holding that the background section of the specification frames the invention in context and can implicitly limit claim terms); Irdeto Access, Inc. v. Echostar Satellite Corp., 383 F.3d 1295, 1300 (Fed.Cir.2004) ("Even when guidance is not provided in explicit definitional format, the specification may define claim terms by implication such that the meaning may be found in or ascertained by a reading of the patent documents") (citations omitted). This translation, as described by the specification, necessarily occurs upstream of the CPU's decode stage-otherwise, the CPU would not be able to process the Java bytecodes. *See* '215 patent, col. 1, ll. 24-38.

Nazomi points to claim 30, "wherein the central processing unit includes an execution unit to execute the register-based instructions." Nazomi also points to claim 32, "wherein register-based instructions cause the manipulation of the register file," pointing out that the register file provides input to the execute logic stage. Read literally then, these claims would appear to indicate that "instructions" exist *after* the decode stage. Such a reading, however, would not make sense. It appears clear from a reading of the entire specification that the register-based instructions referred to by the claims are native instructions input at the beginning of the pipeline, i.e., at the instruction fetch stage. Such instructions do not exist in the same form after the decode stage. While it acknowledges that the language of claims 30 and 32 could be confusing, the Court concludes that the most likely explanation is that the inventor was using the term "instruction" imprecisely in these two instances. Certainly, when native instructions are processed in the pipeline they "cause" a downstream effect, e.g., manipulation of the register file. The Court concludes that the inventor must have been referring to this downstream effect rather than indicating that native instructions exist after the decode stage.

B. Specification

The foregoing construction is consistent with the specification, which consistently uses the term "instruction" to refer either to Java bytecodes or to native processor instructions. The specification never indicates that the term "instruction" means the control signals that are the output of the decode stage and never uses the phrase "microinstruction" or "low level instruction." Nazomi points to the following language:

The CPU is divided into pipeline stages including the instruction fetch 26a, instruction decode 26b, execute logic 26c, memory access logic 26d, and writeback logic 26e. The execute logic 26c executes the native instructions and thus can determine whether a branch instruction is taken and issue the "branch taken" signal.

'215 patent, col. 5, ll.12-17. Again, Nazomi is trying to demonstrate that "instructions" exist *after* the decode stage and in fact are executed by the execute logic stage. Such a literal reading would make no sense here, however; according to the background section of the specification, native instructions are the result of a compiler or Java translator and are *inputs* to the CPU's decode stage. *See* '215 patent, col. 1, ll. 13-45. Taken in context, the stray references cited by Nazomi must be shorthand for the fact that the execute logic stage carries out the operations specified by the native instructions.

Nazomi also makes reference to the following language in the specification indicating that the hardware accelerator "can be incorporated into" the CPU:

Although the Java hardware accelerator is shown in FIG. 1 as separate from the central processing unit, the Java hardware accelerator can be incorporated into a central processing unit. In that case, the central processing unit has a Java hardware accelerator subunit to translate Java bytecode into the native instructions operated on by the main portion of the CPU.

'215 patent, col. 3, ll. 35-41. Nazomi argues that if the hardware unit is incorporated into the CPU, its output cannot be *input* into the CPU. ARM does not dispute that the hardware unit may be incorporated into the CPU, but argues that such incorporation does not change the *logical* relationship between the hardware unit and the CPU's pipeline, meaning that the hardware unit must be upstream of the pipeline.

Nazomi argues that the reference to "incorporation" into the CPU refers to *logical* incorporation, that is, insertion of the hardware accelerator into the pipeline such that the register-based instructions that are the output of the accelerator would not be input to the decode stage. There does not appear to be any support for this argument in the specification itself, which gives no indication that the translated Java bytecodes enter the pipeline at anyplace other than the instruction fetch stage. Nazomi submits a second declaration of the inventor, Mr. Patel, providing several figures showing how the hardware accelerator *could* be integrated into the CPU pipeline. None of these figures appears in the patent specification, however.

The Court notes that the specification references an "embedded solution in which the hardware accelerator is positioned on the same chip as the existing CPU design." '215 patent, col. 6, ll. 56-59. Arguably the reference to "incorporation" refers to this "embedded" solution, in which the hardware accelerator is merely on the same chip as the CPU. Even if the reference to "incorporation" means physical incorporation into the CPU, however, the Court concludes that the specification gives no indication that the logical relationship between the hardware unit's output and the pipeline would be changed. In other words, there is no indication that the hardware unit ever is located anywhere other than upstream of the decode stage. FN3

FN3. Nazomi devotes extensive argument to its position that the presence of the hardware unit on the same chip as the CPU, or within the CPU but upstream of the pipeline, would not satisfy the requirement of an "integrated embodiment." The patent does not use the phrase "integrated embodiment" or even the term "integrated." The term used by the patent is "incorporated." Nazomi appears to have transmuted the term "incorporated" into "integrated." The patent does not use the term "integrated."

C. Prosecution History

The Federal Circuit's remand order suggests that the Court consider the prior art references by Krall, et al. and Dickol, et al. Nazomi, 403 F.3d at 1369. Both the Krall article and the Dickol patent use the term "instruction" to refer to Java bytecodes or native instructions; neither uses the term "instruction" to refer to the control signals that are the output of the decode stage.

Nazomi cites other prior art references, Coon and Trembley. Neither were discussed by the examiner. Nazomi nonetheless argues that these references use the term "instructions" broadly to encompass multiple levels of computer architecture. As it noted above, the Court is prepared to accept the proposition that, in the abstract, the term "instruction" could have the very broad meaning attributed to it by Nazomi. The question, however, is the meaning of the term as used in *this* patent. Because the prosecution history does not disclose any discussion of Coon or Trembley, these references are not useful to this inquiry.

D. Appropriate Construction

As the Federal Circuit noted, and as is described above, "[i]n this patent, it appears that the inventor defined 'instructions' in an indirect manner. Specifically, the specification refers primarily to what the instructions do and where they may do it." Nazomi, 403 F.3d at 1369. Consistent with the manner in which the inventor defined the term, the Court construes the term "instruction" to mean either a stack-based instruction that is to be translated into a register-based instruction, or a register-based instructions that is input to the CPU pipeline. In either case the "instruction" must be upstream of the decode stage of the CPU pipeline. As used in the claims of the patent, "instruction" cannot mean the control signals that are the output of the decode stage.

IV. ORDER

The term "instruction" is construed as set forth herein.

N.D.Cal.,2006.
Nazomi Communications, Inc. v. ARM Holdings, PLC

Produced by Sans Paper, LLC.