United States District Court,
N.D. California.

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

No. C02-02521-JF

**Sept. 30, 2003.**

Thomas J. Friel, Jr., Brandon D. Baum, Clifford Liu, Cooley Godward LLP, San Francisco, CA, Bernard C. Shek, Skadden Arps Slate Meagher & Flom, Palo Alto, CA, for Plaintiff.

Andrew Y. Piatnicia, Darren J. Gold, Howrey Simon Arnold & White, LLP, Menlo Park, CA, Ethan B. Andelman, Howrey Simon Arnold & White, San Francisco, CA, for Defendants.

# ORDER GRANTING MOTION FOR PARTIAL SUMMARY JUDGMENT OF NON-INFRINGEMENT

FOGEL**, J.**

The motion of Defendants (collectively "ARM") for partial summary judgment of non-infringement was heard on February 10, 2003. The Court withheld ruling on the motion pending a claims construction hearing. For the reasons discussed below, the Court will grant the motion.

## I. BACKGROUND

### A. General Concepts

In broad terms, computer programming is a process whereby one or more human beings instruct a computer to perform certain functions to achieve a desired result. Programmers typically write programs in so-called "high-level" computer languages that have defined vocabulary and rules of syntax, and which are capable of being implemented on a broad range of hardware platforms, without regard to how the central processing unit in each of those platforms is designed and functions.

Ordinarily, high-level languages use a subset of human vocabulary, such that a person knowledgeable in the basics of computer programming and in the conventions of a particular high-level language can "read" a program-that is to say he or she can examine the program and determine what the program is instructing the computer to do. A program written in a high-level language typically is referred to as "source code."

Computers cannot respond to high-level language commands directly-the source code first must be translated or converted into machine-readable form. FN1 In current practice, the first step in transforming a program written in a high-level language to commands that can be processed by the computer often is to run the high-level language program through a "compiler," which itself is a computer program, to convert the program into "object code" or "machine code." FN2

FN1. In this portion of this order, terms such as "translate," "convert," and "instructions" are used in a general sense for purposes of background and should not be construed as attempting to give meaning to any of the terms used in the patent claims or specification. Additionally, while the Court intends this general description of the concepts at issue to be reasonably accurate, the Court understands that persons skilled in the art might have different or more precise ways of describing the concepts. The Court does not believe any such differences would be material to the legal analysis.

FN2. As an alternative to using a compiler, source code can be translated to machine code with an interpreter-which interprets one line of source code at a time as the program runs. Interpreters and compilers have different performance advantages and disadvantages. Another alternative is for the human programmer to work in an "assembly language," a more laborious process because assembly languages are further removed from human language, but one that can result in certain performance advantages. An assembler then is used to convert the program to machine language.

Machine code generally is processor-specific. That is to say, after compilation, a program will run only on the particular type of processor for which the compiler has been designed. Thus, each type of processor requires a different compiler for each high-level language to be utilized.

Compilers, therefore, translate instructions from high-level languages that are human-readable and that can be used on many different types of processors into instructions that are "native" to a particular processor. The transformation process, however, does not stop there. A processor then must "decode" the native instruction to carry out the desired function.

It is common to think of computers processors as "thinking" in binary code-that at the heart of the processing, all the processor knows or understands are ones and zeros. It is true that all computer programs ultimately must be reduced to what can be *represented* as a series of ones and zeros, but even that so-called "machine language" is a construct. The ultimate processing or computation is carried out through relative differences in voltage at points in the circuitry-a relatively high voltage point *represents* a one, and a relatively low voltage point *represents* a zero. Thus, while the process of a programming a computer often begins with instructions that are expressed in a near-human language, those instructions ultimately must be transformed to a dynamic pattern of high and low voltage points within a silicon chip.

## B. The Technology at Issue

The patent in suit, U.S. Patent No.6,332,215 B1 ("the '215 patent") relates to a programming language developed by Sun Microsystems known as Java.FN3 Java differs from the typical compiled high-level programming languages described above because it was designed to run on a different types of processors without the need for an individualized compiler for each type of processor. To that end, programs written in Java are compiled into Java "bytecodes," which defendant describes as being "akin" to machine code instructions. However, unlike machine code that has been compiled from another high-level language for use on a particular processor type, Java bytecodes typically are not "native" to the processors on which they run. Rather, processors typically have required a Java Virtual Machine ("JVM")-a software program that translates Java bytecodes into native instructions.

FN3. The patent makes clear that its claims extend not only to Java, but to any other language that may exist or be developed that functions like Java. For convenience, this order will refer only to Java.

It is not immediately obvious to the Court how first compiling Java into bytecodes and then using JVMs to translate the bytecodes into native instructions provides advantages over simply using compilers to translate source codes into native machine codes in the first instance. Nevertheless, there appears to be no dispute that advantages in portability do exist, and the Court can envision a number of reasons that might be so. It also is undisputed that there are trade-offs in performance speed; the translation step involved with JVMs slows down the processing.

The '215 patent is directed at addressing that performance issue. In simplest terms, the '215 patent discloses a solution for using *hardware* instead of or in addition to the software of a JVM to translate Java bytecodes into native instructions.

Again in broad terms, the specification and claims of the '215 patent disclose two basic embodiments. One calls for a hardware unit for translating the bytecodes that is separate from the central processing unit ("CPU"). As an alternate embodiment, the '215 patent discloses including the hardware translator as a "subunit" of the CPU. The parties dispute exactly what it means under the second embodiment for the translator to be "part" of the CPU. In essence, Defendant claims that even in the second embodiment the translator is a subunit that is physically and logically distinct from the "main part" of the CPU, even if it is on the same chip; Plaintiff argues that the claims reach more integrated designs as well.

Even prior to the issuance of the '215 patent, an alternative approach to speeding up execution of Java programs was to use dedicated or "specialized" Java processors. One such system was known as picoJava, and was described in a prior art reference listed as "Krall." As the inventors explained to the Patent and Trademark Office during prosecution of the '215 patent, "[s]pecialized Java processors do not translate Java bytecodes into instructions native to a processor. For specialized Java processors, the Java bytecodes are the native instructions, and no translation is required." While specialized Java processors apparently have their niche, such an approach is inconsistent with the cross-platform portability Java was intended to have.

The instant motion seeking a finding of non-infringement is based on defendants' contention that their "Revision 3" technology, marketed as "Jazelle," does not translate Java bytecodes into native instructions as disclosed by and claimed in the '215 patent.FN4 There is no dispute that defendant's Jazelle products process Java bytecodes at least in part in hardware, rather than through a JVM.FN5 Defendants contend, however, that Jazelle does so in a manner like that used in picoJava-the bytecodes are not translated into native instructions for the Jazelle processor; the bytecodes "are the native instructions, no translation is required." Unlike picoJava, though, Jazelle processors *also* recognize certain instructions as "native" that are *not* Java bytecodes.FN6

FN4. Defendants' motion is for partial summary judgment only, as it does not reach their Revision 2 products, which apparently were more similar in some ways to the invention claimed in the '215 patent. Defendants contend that any infringement by Revision 2 products would be de minimus, as that design was not ever sold in the United States, and was only released to one licensee for evaluation purposes.

FN5. Both the '215 patent and defendants' technology allow for some less-frequently used Java bytecodes to be processed in software. That is not at issue in this motion.

FN6. The other instruction sets that are "native" to defendants' processors are referred to as the ARM instruction set, and a subset thereof referred to as "Thumb" instructions.

Thus, in Defendants' view, Jazelle consists of nothing more than an ordinary processor that happens to be combined with an "old" technology-a specialized Java processor. Plaintiff, in contrast, sees Jazelle's processing of bytecodes in hardware as an infringement of the claims in the '215 patent, and contends that at a minimum, there are triable issues of fact as to how Jazelle operates that preclude a finding of non-infringement on this record.

A final significant point is that Java bytecodes are so-called stack-based instructions, whereas machine code complied from other high-level languages typically is register-based. The distinction between stack-based and register-based instructions involves whether data is accessed from the "top" of the "stack" (stack-based) or whether any item in a data stack can be accessed directly through use of a "pointer" (registered-based). The claims of the '215 patent generally refer to the process of converting Java byte codes for processing as being a conversion from stack-based instructions into register-based instructions.

## II. DISCUSSION

### A. Infringement

Defendants would have the Court resolve this motion by construing the claim term "instruction" as meaning only an "external command that tells a computer processor to perform a discrete task" and that is "recognized at the interface of a given processor." Thus, Defendants argue, the '215 patent requires that Java bytecodes be translated in hardware into "native" instructions *before* reaching the "decode" portion of a processor. Defendants contend that although the decoder of Jazelle processor in one sense further interprets Java bytecodes, the output from the decoder are "control signals," not "instructions." Pointing to its arguments that one embodiment of the '215 permits integrates the Java interpreting hardware with the CPU, Plaintiff argues that Defendant is drawing an arbitrary line, and that the "control signals" emerging from the decoder could as easily be characterized as "low level instructions" that have been translated in hardware, as claimed by the '215 patent.

The Court concludes that Defendants' attempt to resolve the question through a narrow construction of the term "instruction" is perhaps overly simplistic, but that Defendant's fundamental position is sound. As described above, there is a something of a continuum from instructions written in a near-human language to the point where computations are carried out in the relationships between voltage levels in different addresses of a silicon chip. At least on this record, the Court is not persuaded that term "instruction" necessarily is used by persons reasonably skilled in the art is such a narrow sense as to require that it be given the precise construction proposed by defendant.

Nonetheless, more generally the Court can and will construe the terms of the patent claims as calling for a hardware unit or subunit that converts stack-based instructions into register-based instructions prior to the processing of those instructions by the processor in the so-called "decode stage." In so doing, the Court is *not* reading into the claims a limitation from the specification. Rather, the Court is construing the patent, as it must, in light of the specification. Both the specification and the prosecution history reveal that the prior art-picoJava and other specialized Java processors-implement a hardware solution for processing Java bytecodes. It follows necessarily that the claims of the patent, to be valid, must reach a different type of hardware solution, and that the solution of the prior art does not infringe.

Here, although Plaintiff does not rely on any particular proposed construction language, in essence it is proposing a construction that would provide no basis for distinguishing the prior art. Plaintiff has articulated arguments as to why Jazelle could be seen to infringe the claims of the '215 patent in the sense that Plaintiff believes those the claims should be understood. Plaintiff has not shown, however, how picoJava or other specialized Java processors would not also infringe under such an interpretation of the claims. The Court therefore concludes that Revision 3 does not infringe the '215 patent either literally or under the doctrine of equivalents, and that no reasonable trier of fact could find to the contrary.FN7

FN7. The Court notes Plaintiff's objections to the foundational basis of the evidence submitted by Defendants as to how Jazelle operates. The Court is satisfied from the evidentiary record as a whole, including the evidence submitted by Plaintiff itself, that there are no material disputes as to how Jazelle operates that would affect the analysis.

### B. Remaining issues

Still pending before the Court are the parties respective's requests for claim construction, heard on April 22, 2003. In light of defendant's representation to the Court that substantially less is at stake with respect to any infringement issues by revision 2, and in view of the conclusions of this order, the Court recognizes that the scope of what remains to be construed may have changed. Accordingly, the Court declines to issue a further claims construction order pending further input from the parties as to what terms, if any, still warrant construction. To that end, the parties shall appear for a further case management conference at the time and date specified below.

## II. ORDER

Defendants' motion for partial summary judgment is GRANTED. The parties shall appear for a further case management conference on November 3, 2003 at 10:30 am.

N.D.Cal.,2003.
Nazomi Communications, Inc. v. Arm Holdings PLC