CORE Metadata, citation and similar papers at core.ac.uk

Provided by Santa Clara University School of Law

Santa Clara High Technology Law Journal

Volume 3 | Issue 1 Article 1

January 1987

NEC v. Intel: A Challenge to the Developing Law of Copyright in the Protection of Computer Programs Editors'

F. Thomas Dunlap Jr.

Follow this and additional works at: http://digitalcommons.law.scu.edu/chtlj

Part of the Law Commons

Recommended Citation

F. Thomas Dunlap Jr., *NEC v. Intel: A Challenge to the Developing Law of Copyright in the Protection of Computer Programs Editors'*, 3 Santa Clara High Tech. L.J. 3 (1987).
Available at: http://digitalcommons.law.scu.edu/chtlj/vol3/iss1/1

This Case Note is brought to you for free and open access by the Journals at Santa Clara Law Digital Commons. It has been accepted for inclusion in Santa Clara High Technology Law Journal by an authorized administrator of Santa Clara Law Digital Commons. For more information, please contact sculawlibrarian@gmail.com.

NEC V. INTEL: A CHALLENGE TO THE DEVELOPING LAW OF COPYRIGHT IN THE PROTECTION OF COMPUTER PROGRAMS

F. Thomas Dunlap, Jr.[*]

I. INTRODUCTION

Since Congress established that computer programs would be protected by the copyright laws, the courts have been faced with the challenge of pacing the development of the law with the burgeoning technologies of computer science. Fortunately for continuing innovation and investment, the courts have proven equal to the task. Having conquered the intricacies of source code, object code, ROM, PROM and EPROM, the court system recently found itself examining yet another facet of technology when, in December of 1984, NEC[1] filed suit against Intel Corporation for declaratory judgment challenging the copyrightability of microcode (or microprograms, a term used interchangeably with microcode).

The microprocessor is the driver of the modern day personal computer, and microcode is the master that directs the microprocessor. Microprograms interpret and convert high level instructions into microinstructions which the computer can understand. The microcode itself is a computer program which can be stored in a variety of media. The Intel microcode is embedded in Read Only Memory (ROM) directly on the microprocessor. The use of microcode enhances design efficiency, allowing microprocessor operations to be directed by computer programs which are readily changeable, rather than by hardware which can be costly and time consuming to change or correct.

Although the suit which is the subject of this article was filed in December, 1984, the story begins in April of 1978 when Intel introduced its 8086/88 microprocessors. They were Intel's first 16-bit microprocessors and the first to utilize microprograms. NEC copied those products, including the copyrighted microprograms

Copyright © 1987 F. Thomas Dunlap, Jr. All Rights Reserved.

* F. Thomas Dunlap, Jr. is General Counsel for Intel Corporation.

1. The original plaintiff was Nec Electronics, Inc., a U.S. corporation. NEC Corporation, a Japanese corporation was joined.

(bit-for-bit), and sold them as the uPD 8086 and uPD 8088 microprocessors. When Intel complained about NEC's use of Intel's copyrighted microprograms, the claim was settled without litigation by a license agreement dated February 23, 1983.

The present suit arose out of a decision by NEC to attempt to create its own series of microprocessors, the V-series, that did not use Intel's microprograms (and thereby to avoid payment of royalties). Intel's claim is that, while NEC succeeded in creating the surface appearance of change in the V-series microprograms, they, in fact, were copied and derived from Intel's microprograms.

NEC's complaint for declaratory judgment sets out a series of "fire and fall back" positions. First NEC alleged that microprograms were not copyrightable. However, in the event that it was wrong on that issue, it alleged that if Intel had copyrights in its microprograms, the copyrights had been forfeited because some Intel licensees had failed to mark copyright notice on licensed products. In the event that a court should find that NEC was wrong on both those issues, NEC alleged that its microprograms did not infringe Intel's copyrights, or that if they did infringe, the infringement was required because there is only one way to write the microprograms to perform the task which NEC desired to perform. And finally — even if a court would find that microprograms are copyrightable, that Intel's microprograms were validly copyrighted, and that NEC copied the microprograms, although there was no necessity to do so — NEC reserved two more fallback positions: either NEC was licensed to copy the microprograms or Intel has misused its copyrights so that they cannot be asserted against NEC.

Intel counterclaimed seeking an injunction prohibiting NEC from infringing its copyrights any further, damages, and costs of suit. After a two month bench trial before Judge William A. Ingram of the U.S. District Court, Northern District of California, the court ruled that "defendant and counterclaimant Intel Corporation has good, valid and existing copyright on its 8086/8088 microcode." The issues of damages and unfair competition were each bifurcated for separate trials. The court has not yet ruled upon the infringement issue.

As the court's partial decision reflects, the major issues in the case can be grouped into three main categories:

(1) Is microcode copyrightable? The court has ruled that it is.

(2) Did Intel possess a valid copyright on its microprograms? The court has ruled that Intel has valid and existing copyrights.

(3) Did NEC infringe Intel's copyright? This issue is not yet decided.

Each of these issues is considered separately below in the context of Intel's position and the proof at trial.

II. Copyright Protection for Computer Programs

The fundamental issue of the copyrightability of computer programs has been well-settled by statute and by case law. Under the 1980 amendment to the Copyright Act, copyright protection was explicitly affirmed with respect to computer programs, which were defined as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[2] The case law has uniformly affirmed this protection for a variety of computer programs, irrespective of their code level or embodiment.[3] As the district court in *Formula* emphasized:

> It is crystal-clear that CONTU recommended that *all* computer programs, fixed in any method and performing any function, be included within copyright protection. There likewise can be no doubt that Congress accepted that recommendation and embodied it in the 1980 amendments to the copyright law.[4]

Since it is clearly established that computer programs in general are protected under copyright, the task becomes one of demonstrating that microcode meets the legal definition of a "computer program."

A. *The Copyrightability of Microcode: As a Computer Program*

Microprogramming, like other forms of computer programming, enables a vast range of expressions to be adapted to the computer. As with other computer programs, writing microprograms is a creative process, usually beginning with a higher or English-like language which is then translated into a machine-readable language of 1's and 0's. Put in more conventional computer terms, microprograms are written in source code and translated into object code. As the court in the *Intel* case stated in its Findings of Fact, "the methodology employed in the creation of microcode is indis-

2. 17 U.S.C § 101.
3. *See, eg.*, Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983); Apple Computer Int'l, Inc. v. Formula Int'l, Inc., 725 F.2d 521 (9th Cir. 1984).
4. Apple Computer, Inc. v. Formula Int'l, Inc., 562 F. Supp. 775, 781 (C.D. Cal 1983) (emphasis by the court); *aff'd*, 725 F.2d 521 (9th Cir. 1984).

tinguishable from that employed in the creation of any computer program."

That finding is in large part based on the testimony of Hiroaki Kaneko, the engineer who wrote the NEC microcode:

> Q. . . . Can you explain to the Court what microcode assembly refers to?
> A. Microprograms, as I have been referring to them up until now, are source level microprograms and in order to incorporate these source programs to actual chips, you have to transfer them into binary patterns. And these binary patterns are what we call object code. And the program that produces this object code is what is called a microassembl[er].
> [Tr. 1890:2-10 (Kaneko)]
>
> * * * * * * *
>
> Q. You create the source code, then the TACO assembler generates object code from the source code; right?
> A. Yes.
> [Tr. 2058:23-2059:1 (Kaneko)]

The debate does not end at this point, however. Once created, microcode can be stored in a variety of media, such as paper, floppy disks or semiconductor chips. Intel's microprograms were embedded in ROM on the microprocessor chip, as are many computer programs. NEC argued that due in part to this form of inscription, the microprograms become a part of the hardware of the machine and as such are subject to patent but not copyright protection. The court in this dispute, however, followed the clear ruling in *Apple Computer, Inc. v. Franklin Computer Corp.*,[5] that a computer program, even when embedded in a ROM chip, is copyrightable. The court emphasized that the storage modality of a program does not change the nature of the program.

Other NEC attempts to justify isolating microprograms from other computer programs similarly failed. For example, NEC argued that microcode was distinguishable from other computer programs by the fact that often microcode is not accessible to the user, but rather encoded in the chip before shipment to the customer. As the court noted, however, some forms of microcode can be manipulated, or even written, by the user.[6] By ruling in favor of copyrightability, the court implied that user accessibility is not a determinant

5. Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1251 (3rd Cir. 1983).

6. Although the 8086 microcode is not normally "accessible" to the user, it is possible to customize the microcode to suit a particular user's needs.

of copyrightability. Had the court ruled that accessibility was a barrier to copyrightability, the industry would have found itself with two classifications of microcodes: those which are accessible to the user and protected, and those which are not protected simply because they are inaccessible to the user. The unworkability of such a distinction is obvious.

Finally, protection of microcode was attacked on the ground that microcode performs a different function than do other computer programs. Microcode adds one step into the process of translating from quasi-English (the source code) into 0s and 1s (the object code). However, this does not mean that a non-program function has been added, but rather that a traditional function has been subdivided and improved. Where there is no microcode, other computer programs are converted directly into electrical control signals and are utilized in the computer to implement the macroinstructions (instructions that are available to the user); where there is microcode, macroinstructions are interpreted by the microprograms to lower level instructions. Microprograms thus bring about the end result in the same manner as conventional programs. As the Intel-NEC court concluded in its Partial Findings, "the control signals generated in a nonmicroprogrammed computer perform the same kind of tasks generated in a microprogrammed computer." Mr. Kaneko provided some of the supporting testimony:

> Q. In the 8086/8088, there are some macroinstructions which directly control the hardware?
>
> A. Yes. I am aware that such macroinstructions do exist.
>
> Q. So, in the 8086, the control structure sometimes is directed by the microcode and sometimes by the macrocode; is that right?
>
> A. In the sense that macroinstructions can directly control hardware. In that sense, sometimes macroinstructions and sometimes microinstructions are used for control.
>
> [Tr. 2071:25-2072:10 (Kaneko)]
>
> * * * * * * *
>
> Q. In the computer microprocessor without microprogramming, the control structure operates upon commands from the macroinstruction, doesn't it?
>
> . . . .
>
> A. There is no mistake in the fact that the macroinstructions will be the trigger to the output, yes.
>
> [Tr. 2070:11-13; 2071:11-13 (Kaneko)]

In sum, microcode cannot be excluded from the generic classification of computer programs simply because of its advantages of

ROM storage or increased efficiency of translation. In the instant case, as the court ruled, the Intel microprograms are computer programs within the meaning of the Copyright Act and entitled to all the protections extended to computer programs under the Act.

B. *The Copyrightability of Microcode: As a Literary Work*

Even without the explicit addition of the computer program copyright protection amendment, microcode is entitled to copyright protection as a literary work because the Copyright Act provides protection for original works of authorship "expressed in words, numbers, or other . . . numerical symbols or indicia."[7] Microcode indeed can be written in words, numbers or other numerical symbols. Citing to the *Apple* cases, the court agreed in its Partial Findings that Intel's microprograms are "copyrightable literary works expressed in words, numbers or other numerical symbols or indicia."

The court's findings carry import far beyond this specific set of Intel microprograms. This initial decision should bring some peace of mind to the large segment of the industry which has invested time, money and trust in developing microcode under the assumption that copyright law would continue to protect its computer program products.

The marking and infringement issues in the NEC-Intel dispute are by nature far more fact-dependent than the copyrightability issue. These issues are important to the protection of the 8086 microcode but they do not have the precedental value of the ruling that microcode is copyrightable.

III. Intel Maintained the Validity of Its Copyright

NEC contended that, in the event microprograms were found to be copyrightable, Intel had lost whatever copyrights it had. Under the Copyright Act, the owner of a copyright is given certain responsibilities as well as rights in the works. The owner must add copyright notice to his work upon publication or, if this is inadvertently omitted, must make reasonable efforts to add notice to the product in the chain of distribution.[8] Failure to do so may result in forfeiture of the owner's copyright protection.

NEC chose to attack Intel on precisely this issue: Whether Intel had properly marked its product with copyright notice and, if

7. 17 U.S.C. § 101.
8. 17 U.S.C. §§ 401, 405(a)(2).

not, whether Intel had made reasonable efforts to correct any deficiencies in its marking. Of all the issues at trial, this question was the least specific to computer technology and involved a factual investigation tracing through various contracts and histories.

The court in the Partial Findings summarized the rather lengthy history of Intel's and NEC's experiences with marking the Intel microprograms as follows: Intel had "always marked with copyright notice the 8086 products marketed and distributed directly by it . . . [and at] all relevant times had a policy requiring licensees of the 8086 product to mark with copyright notice."

However, not all of Intel's licensees had such a stellar record of marking the products with copyright notice. Of Intel's twelve licensees, nine had agreements containing express marking provisions. As the court found, the fact that three licensees did not have express marking provisions was "the result of inadvertence and mistake, and not the result of a lack of policy or a deliberate nonenforcement of policy."

NEC was one of the three licensees without the express marking provision. When Intel discovered that NEC, as well as the two other licensees, was not marking its product, Intel not only sought an immediate marking agreement but also audited all licensees. Intel also took the remedial step of sending several thousand copyright stickers to major distributors of NEC's products in the U.S.A. Intel's vice president David House testified to these issues, asserting Intel's policy to mark all 8086 products and describing Intel's efforts to obtain marking by its licensees. As a result of Intel's efforts, as the court noted in the Partial Findings, just one-tenth of Intel's microprograms were distributed without marking. Furthermore, Intel had also registered its microprograms within five years after the "publication without notice" by its licensees. As a finding of law, the court held that Intel's efforts to add notice to the unmarked products had been reasonable and entitled Intel to be excused for the omission pursuant to 17 U.S.C. Section 405(a)(2). In light of the evidence that Intel itself had marked and had made reasonable efforts to obtain marks for its licensees' products and had obtained timely certificates of copyright registration, the court acknowledged in the Partial Findings that the Intel microprograms were "protected by good and valid copyright which has not been forfeited." The focus of the inquiry therefore shifted to ascertaining the degree to which NEC infringed upon that copyright.

IV. NEC'S INFRINGEMENT OF INTEL'S MICROPROGRAM COPYRIGHT

A. *Direct Evidence of Copying*

It was NEC's position at trial that even if microprograms were copyrightable and Intel's copyrights were not forfeited, NEC's microprograms did not infringe. Under the Copyright Act, the owner of a copyright has the exclusive right to make copies or derivative works from the copyrighted work.[9] Any person who violates this right is guilty of copyright infringement.[10] Infringement of a copyright is established by direct evidence of copying.[11] In a recent case involving infringement of computer programs and audiovisual works, the district court found that direct evidence of copying could be established by two factors: first, the use of instructions which did not make sense in the machine, and second, evidence that the same programmer had both copied the copyrighted program and "invented" his own similar program.[12] Other courts have held that copying of a computer program may be proved by use of a similar code where it is less efficient, by creation by someone with less experience than the original author of a similar work in less time, or by similarities in the structure and organization of the two programs.[13]

1. Common Errors in Both Programs

One indicator of copying is the existence of errors or unnecessary materials common to both works.[14] Errors are traditionally evidence of copying because two people rarely create the same error independently. Rather, errors are copied — usually by mistake. Suspicions were raised in this area when Intel studied NEC's microcode and found that unnecessary specifications appeared in both the Intel and NEC microprograms.

As Intel's expert put it, ". . . You see everything else lines up. . . . [T]his unnecessary operation was in Rev 'O', just like the [Intel code] 8086 . . . I can't imagine why [the NEC programmer]

9. 17 U.S.C. §§ 101, 106(1)-(3).

10. 17 U.S.C. § 501(a).

11. M. Kramer Mfg. Co. v. Andrews, 783 F.2d 421, 445 (4th Cir. 1986).

12. Broderbund Software, Inc. v. Unison World, Inc. [slip op. C85-3457, October 8, 1986 (N.D. Cal. 1986)].

13. *See, eg.*, E.F. Johnson Co. v. Uniden Corp., 623 F. Supp. 1485, 1494 (D. Minn. 1985); SAS Institute, Inc. v. S & H Computer Systems, Inc., 605 F. Supp. 816, 823 (M.D. Tenn. 1985); Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., 609 F. Supp. 1307, 1322 (E.D. Pa 1985).

14. Johnson Co. v. Uniden Corp., 623 F. Supp. at 1494.

put something in which he knew he didn't need to do unless he copied what the 8086 did."[15]

2. Documentation

Copying may also be implied where there is inadequate documentation to prove independent creation.[16] The adequacy of the documentation can be judged by comparing it to the documentation available for similar projects. In this instance, there were very few development notes maintained for the specific NEC microprograms in issue as compared to the numerous notes showing development of other microprograms by the same NEC programmer. Some of the former notes in fact had copies of the original Intel microprograms attached to the notes. Of the 88 NEC microprograms which interpreted 8086 macroinstructions, NEC produced development notes or records for fewer than 10.

3. Unusual Speed in Creating the Program

The NEC microprogram was created under unusual circumstances and with amazing speed. As one NEC programmer stated, "I myself felt that I would not be able to meet the [development] schedule." The primary programmer, Mr. Kaneko, testified that he wrote the NEC code in one-half a man-month. Kaneko came to the project with no academic training in microprogramming; his only experience with microcode was disassembling the Intel microcode for NEC. Kaneko was so inexperienced that he copied the format of Intel's instructions because as he admitted, he "did not know of any other microinstruction format that could be referred to in determining the format. . . ." In contrast, the original Intel microprogram was written by an eight year veteran of programming who spent over a month on the code, also working under a very tight schedule. He testified concerning the need to write and rewrite the code many times due to the complexity of the tasks and the technology. Such testimony should be viewed in light of the common law doctrine that copying may be inferred where there is proof of the creation of a similar work in less time than the original or by someone with less experience than the original author.[17]

4. Conduct Implying Copying

Finally, evidence of certain types of conduct can provide proof

15. [Tr. 237:1-9 (Patterson)].
16. SAS Institute, *supra* note 13, at 823.
17. Whelan, *supra* note 13 at 1322; SAS Institute, *supra* note 13, at 830.

of copying. For instance, evidence of efforts to disguise copying was accepted as proof of copying in *SAS Institute.*[18] At least three aspects of NEC's conduct imply cover-up of copying.

NEC had demonstrated a pattern of copying. As described above, NEC had copied the Intel microcode in an earlier situation. More relevant here, however, was the fact that NEC planned to copy again and began to program CMOS machine, called the 80C88, with Intel-based microprograms. As an NEC manager testified:

> Q. . . . Is it true that the NEC plan was to take the microcode from the uPD 8088 and install it in the CMOS version?
> A. Yes, that is a — that is true.
>
> . . . .
>
> Q. In short, your testimony was and is that your plan was for the CMOS version to be a bit-for-bit copy of the NMOS 8088; is that correct?
> A. To the extent of my understanding and to the extent of my recollection, that is true.
> [Tr. 2205:6-9; 2206:4-8 (Sasaki)]

* * * * * * *

When the first dispute arose over the copying of the 8086 and 8088 in the uPD 8086 and uPD 8088 (NEC's version), however, NEC pulled its employees off the 80C88 project and assigned the exact same group to a "completely new" project, the development of the V20 and V30. This project evolved to produce the microprogram at issue in the Intel-NEC trial. Yet testimony showed that this supposed change of projects became little more than a name change, at least insofar as the microcode was concerned:

> Q. You referred in your testimony to a product called the 80C88. Did that product later become the V20?
> A. As I answered earlier, some people had called this product, which was going to be named V20 later on, as 80C88 at certain point in time during 1982.
> [Depo. Tr. 64:6-10 (Kani)]

* * * * * * *

> Q. Did you stop working on the 8088 simple CMOS version during 1982?
> A. I don't think stop is a right word to use.

18. SAS Institute, *supra* note 13, at 822-823; *accord,* M. Kramer Mfg., *supra* note 11, at 446.

Q. What happened to the effort that was being devoted to the simple CMOS 8088 version?
A. It was, I think, useful for doing my work myself in developing V20 and V30.
[Depo. Tr. (Vol. I) 55:26-56:4 (Koike)]

* * * * * * *

Q. When, as far as you know, was work on the simple CMOS version of the 8088 stopped?
A. I don't remember it accurately when the work for CMOS version of 8088 stopped, but I think it was the latter part of 1982.
Q. Was that effort redirected to the V20, V30?
. . . .
THE WITNESS: When you said the effort in your question, I don't understand the exact content of the effort, but it is true that the people who were working on CMOS version of 8088 worked on V20 and V30.
[Depo. Tr. 51:10-15; 52:3-6 (Nohara)]

Secondly, NEC's microprogram is run on hardware which is very similar to Intel's.

One NEC manager testified:

Q. The basic hardware, the basic architecture, that you are starting with and adding a dual bus to, adding a loop counter too, is basic 8086/8088?
A. I[t] used as a base the internal hardware of the 8088.
[Tr. 2282:18-22 (Iwasaki)]

* * * * * * * *

A second NEC manager testified that he was aware of the risk in copying Intel's hardware:

Q. . . . Is it true that you told Dr. Kani that he needs to plan a microprocessor with completely different architecture than the 8088/86?
A. Yes, it is exactly as stated.
[Tr. 2220:18-22 (Sasaki)]

* * * * * * * *

Q. You understood at the time when you issued your instruction to Dr. Kani that if NEC used the same architecture as INTEL used there was a substantial risk that the microcode would be the same or similar as that used by INTEL. Isn't that right?
A. Yes, I did think that if the same architecture is used there will be a high probability that the similarity, that there will be similarity between the two, yes.
[Tr. 2222:1-9 (Sasaki)]

Ironically, NEC does not gain full advantage from the additional hardware it did implement. As NEC's expert testified, NEC lost the benefit of its augmented architecture because of the choices made in the microprograms. For example, NEC's choice of entry points in the microprograms results in a loss of efficiency; not coincidentally, NEC's microprograms mimic most of the entry points in Intel's microprograms.

Finally, NEC's behavior is suspect in that its witnesses cannot outright deny copying the Intel microprograms. The same NEC programmer who disassembled the Intel microprograms for one project later specified the instructions for the NEC V20 and V30. When questioned closely about whether he had copied any instructions from the disassembled Intel microprograms during the NEC development period, he could not remember his actions:

> Q. And from all the possibilities that you had to write reset, you wrote a sequence that was almost identical to Intel's; didn't you?
>
> A. In terms of the end result, yes, it is something close.
>
> Q. Mr. Kaneko, is there any possibility that while you were generating the REV.O V20 reset sequence, you referred to the disassembled code?
>
> A. I also do not recall whether or not I looked at the character string or strings at that time.
>
> Q. Is there any possibility that while generating the REV.O sequence for XLAT, you referred to the disassembled code?
>
> . . . .
>
> A. With respect to that also, during that particular time, I do not recall whether I referred to the character string.
>
> Q. My question in both instances was not whether you recall, but whether there is a possibility that you referred to the code.
>
> A. It is not [zero], no.
>
> Q. In any event, as you explained yesterday, you knew you weren't supposed to look at the disassembled code. Can you tell us today, in all honesty, that you didn't do so?
>
> A. I do not recall whether any reference had been made or not. [Tr. 2150:11-23; 2151:5-18 (Kaneko)]

NEC's expert carefully refrained from learning exactly what NEC's engineer Mr. Kaneko did when he wrote the microprograms:

> Q. Are those similarities enough, Dr. Frieder, to suggest to you the possibility that the Rev "O" code was copied from the disassembled uPD 8086 code?
>
> A. I don't know if it was copied.

Q. I took your deposition on May 17, and at page 585, I asked you this question and you gave this answer, lines 5 to 8:
QUESTION: Is this enough similarity to suggest to you that there is a possibility that the right-hand column is copied from the left-hand column?
ANSWER: Yes.
You would give that same answer today, would you sir?
A. Sure. I said I don't know. I don't know if it was copied.
Q. You don't for a moment suppose that Mr. Kaneko just remembered the uPD 8086 code by heart, do you?
A. I don't know what Mr. Kaneko did.
[Tr. 1582:14-1583:10 (Frieder)]

* * * * * * * *

Q. . . . I'm asking you whether or not that [Ex. AGT] doesn't suggest to you, the expert investigator, the possibility that Mr. Kaneko used his disassembled code to write that Rev. O version of his INTSEQ sequence?
A. He may have, he may not have. I don't know.
[Tr. 1716:17-21 (Frieder)]

* * * * * * * *

Q. In fact you weren't asked to decide or consider whether Mr. Kaneko used the disassembled source code and you don't know at all if he used it or not?
A. I don't know if he used it or not.
[Tr. 1803:1-4 (Frieder)]

* * * * * * * *

Q. . . . Isn't is correct and fair to say that you really don't know whether or not Mr. Kaneko copied sequences in the INTEL microcode?
. . . .
THE WITNESS: In all legal sense I don't know if Mr. Kaneko copied.
[Tr. 1803:13-15; 1803:25-1804:1 (Frieder)]

B. *Indirect Proof of Copying*

Copying can be proved either by direct evidence or by indirect evidence showing access and substantial similarity. As the federal district court in *Broderbund* said: "In the interest of creating a comprehensive record, the Court deems it desirable also to undertake a circumstantial analysis of copying, i.e., a determination of whether defendant had access to the copyrighted work and whether there exists 'substantial similarity' between the copyrighted work and defendant's work."[19]

Proof of access and substantial similarity is an established means for demonstrating copying. The Ninth Circuit has stated: "Copying is ordinarily established indirectly. The plaintiff demonstrates that the defendant had access to the copyrighted items, and that the defendant's product is substantially similar to plaintiffs' work. Once this is done, the burden shifts to the defendant to prove his work was not copied, but independently created."[20]

1. Access to Copyrighted Work

An admission of access is an important element when considering other proofs of copying. Where substantial similarity of the works is offered as indirect proof of copying, an admission of access will reduce the quantum of proof required.[21] NEC had in fact copied Intel's microcode bit-for-bit for an earlier NEC machine, the uPD, without any authorization from Intel and access to the copyrighted work was not contested in this dispute.

2. Substantial Similarity

The traditional test for substantial similarity is whether the work is recognized by an ordinary observer as having been taken from the copyrighted work.[22] This test has undergone some modifications in recent years. As the court pointed out in *Broderbund,* the Ninth Circuit adopted a two-step test for determining substantial similarity — an "extrinsic test" consisting of a side-by-side comparison and expert testimony and an "intrinsic test" reflecting the ordinary person's response. The district court in *Broderbund* believed that a new test integrating both lay and expert testimony was the "way of the future" (as shown by its use in the Third Circuit). In *Intel,* the court was presented with all the elements necessary: expert testimony, expert side-by-side comparisons, and the materials from which to draw a conclusion.

Many of NEC's microprograms, including several key sequences, are substantially similar to Intel's corresponding microprograms. Dr. David Patterson, Intel's computer expert, testified that at least 54 out of the 88 sequences in NEC's code were substan-

19. Broderbund, *supra* note 12, at 16.
20. Kamar Int'l. Inc. v. Russ Berrie & Co., 657 F.2d 1059, 1062 (9th Cir. 1981).
21. Sid & Marty Kroft Television Productions, Inc. v. McDonald's Corp., 562 F.2d 1157, 1172 (9th Cir. 1977).
22. Transgo, Inc. v. Ajax Transmission Parts Corp., 768 F.2d 1001, 1018 (9th Cir. 1985), *cert. denied,* 106 S. Ct. 802 (1986).

tially similar. NEC's expert witness, Dr. Gideon Frieder, agreed that "very many" were in fact very similar:

> Q. . . . Now, as I understand it, Dr. Frieder, you did not [sic] note that there were some places where the V20 and V30 [and] Intel code were identical. Correct?
> A. Yes.
> Q. And there were some [places] that you noted where the codes were very close, very similar, correct?
> A. Yes.
> . . . .
> I analyzed all of them. I analyzed all of them. I classified all of them. I considered *very many* of them *very similar,* very similar and completely constrained (emphasis supplied).
> [Tr. 1728:13-19; 1733:13-16; 1788:2-7 (Frieder)]

The high degree of similarity between the Intel and NEC codes is even more suspect when one considers that five years had passed between the times that Intel and NEC had each written their microprograms. In trial, Intel's programmer explained the effect of this time difference: "In the five years between '77 and '82, the semiconductor technology had advanced in many ways. The dies could be safely manufactured physically bigger and transistors were smaller and many more could fit on the die . . . In my opinion . . . not only would the program itself have been different, but I would have gone so far back up the change of the design, the design change, that I certainly would have changed the language in which the program was written . . . ."[23]

C. *Was the Similarity in Microprograms Required?*

Given that a great many similarities exist between Intel's and NEC's microprograms, the issue becomes whether there was any legal justification for these similarities. NEC argued that the similarities were required due to the constraints on programming imposed by the hardware. Intel, in turn, questioned both NEC's choice of hardware and whether the similarities must necessarily exist even given the same hardware.

1. The Inapplicability of the Merger Doctrine

NEC's arguments in this area sought to lay a foundation for a finding of merger. The merger doctrine states that copyright protection does not extend to works which encompass an idea which

23. [Tr. 464:25-465:15; 466:1-12 (McKevitt)].

can only be expressed in one way. The idea and the expression are in that instance "merged," and since copyright does not protect an *idea* from being copied, but only the specific expression of the idea, such works are not protected.

The courts have not readily applied the merger doctrine to computer program cases, however, but have sought to determine whether other variations on the suspect programs were possible. As the Third Circuit stated, "If other programs can be written or created which perform the same function as [this] operating system program, then that program is an expression of the idea and hence copyrightable."[24] Another court has declared, "Even in the case of simple statistical calculations, there is room for variation. . . ."[25] The Third Circuit has refined the statement: "When there are various means of achieving the desired purpose, then the particular means chosen is not necessary to the purpose; hence, there is expression, not idea."[26] Thus, in the NEC situation, the court must determine whether there were other ways in which NEC could have written microcode to direct the computer to perform certain instructions. It was Intel's position that NEC had *many* choices in building a compatible microprocessor. NEC could have implemented the 8086 instruction set entirely in hardware, could have used both microprograms and hardware but selected different hardware or could have written original microprograms.

a. NEC could have used different hardware

Although NEC argued that microcode similarities were required by the hardware and instruction set, Intel contended that the similarity in hardware was a result of NEC's choice, not necessity. As NEC's expert Dr. Gideon Frieder agreed, NEC could have designed a microprocessor that executed the Intel instruction set with microcode radically different from Intel's:

> Q. Is it true that NEC could have designed a microprocessor that executed the 8086 instruction set with radically different microcode than that employed by Intel's 8086. That was entirely possible, wasn't it?
>
> A. Yes.
>
> [Tr. 1523:19-23 (Frieder)]

* * * * * * *

24. Apple v. Franklin, *supra* note 5, at 1253.
25. SAS Institute, *supra* note 13, at 825.
26. Whelan, *supra* note 13, at 1323.

> Q. Last week, you recall that we agreed a microprogrammer could design totally different hardware and totally different microcode to implement and still be compatible with a given macroinstruction set.
> A. That is correct.
> [Tr. 1633:11-20 (Frieder)]

NEC's primary programmer agreed this was true as well. Even NEC's counsel stated in trial that "NEC has already acknowledged that there were other hardwares that could have been used."

Intel's expert witness Dr. Patterson traced the path that NEC probably followed in developing its microarchitecture. As he explained:

> Well, assuming that the 8086, you decided to be compatible with the 8086 macroinstruction set, the first decision point is whether you are going to implement the microprocessor all in hardware or in a combination of hardware and microcodes . . . Now, Mr.Kaneko testified that NEC adopted the Intel microinstruction format even though there was no necessity to do so. Also, the microinstruction set is very similar to Intel's. . . . One of the most important decisions in the design of microarchitecture is how to decode macroinstructions. There are many ways to do this . . . In 1977, [Intel's] Mr. McKevitt decided to use a matching instruction decoder and, in 1982, [NEC's] Mr. Kaneko decided also to use a matching instruction decoder, although that wasn't necessary. . . . I think it would be fair to say that NEC adopted the Intel microarchitecture.[27]

Dr. Patterson further testified that NEC could have used different hardware, a different microinstruction format, a different microinstruction set, a different decoder, a different ROM size, and still be software compatible with the Intel 8086, as Intel itself did with two later microprocessors, the 80286 and 80386. As he stated: "In my view, it is that [NEC] started at the disassembled code, provided a microarchitecture that would allow them to refer to this disassembled code and copy the microprogram.[28]

b. NEC Could Have Designed Different Microcode

In addition to expert testimony that the Intel and NEC microcode did not have to be, but in fact were, similar, experts for both sides also demonstrated that several different microprograms could be written as alternatives to the Intel microprograms, even if

27. [Tr. 2776:11-17; 2777:18-2778:13; 2778-23:2779:7; 2782:15-2783:4 (Patterson)].
28. [Tr. 2843:16-2844:10 (Patterson)].

written for the same hardware elements common to Intel's and NEC's microprocessors. Among the comparisons which demonstrate that substantial similarity between the Intel and NEC microprograms was the result of copying are the following:

> (1) There are at least 90 ways to reorder the microinstructions to accomplish the RESET function in the Intel microprocessors; about one-fourth of those would operate faster than the original written by Intel's microprogrammer. In the original NEC version of the RESET sequence, the order is exactly the same as the order chosen by Intel. Many ways were also demonstrated to write the reset sequence for the NEC machine; at least a dozen of the rewritten sequences were faster than the microprogram used by NEC and there were many more that could have been used.
> (2) Numerous ways were demonstrated to write the common interrupt procedure. The NEC microprograms implementing interrupts were substantially similar to the Intel microcode for interrupts. One NEC sequence utilized a unique code appearing in the Intel sequence — this was an Intel "patch" specifically written to program around a hardware bug in the Intel machines. NEC argues that once the hardware bug was copied, the patch was the obvious way to program around the adopted bug.
> (3) The NEC microcode utilizes 217 out of 288 entry points and 66 out of 76 instruction groupings used by the Intel code, thus sacrificing improvement and speed which could otherwise have been realized by the hardware changes in the NEC microprocessor. As NEC's Kaneko testified, many different choices of entry points and groupings were available to NEC.

c. *The Merger Doctrine Does Not Apply*

Such similarities cannot be rationalized as being the only way to write the program or to express the idea behind the program. It is not the idea behind the program which Intel seeks to protect, but the programmer's individual expression of that idea. Here, as Dr. Patterson testified, "the structure, organization and ordering of the microinstructions in the Intel microprograms constituted the expression of the microprogrammer reflected in those microprograms and not merely the idea or function which inspired the programs." The evidence demonstrated many possible modes of expression.

V. NEC's "Last Ditch" Arguments

All else failing (i.e., microprograms are copyrightable; Intel's copyrights are preserved; NEC's microprograms are copies of Intel's; and NEC was not constrained to copy) NEC argues two additional points. NEC contends that Intel licensed NEC for products such as the V-series, and that Intel misused its copyrights.

The license argument has been fairly well disposed of by candid testimony from NEC's in-house counsel, Robert Hinckley:

> Q. Well, your company and you have taken positions on that, have you not, to the extent that the V-series is a new generation of product and not covered, not subject to the '83 license. Isn't that true?
> A. Our position was that the V-series was, the microcode in the V-series was independently developed and that the V-series, itself, is a proprietary microprocessor and, as such, does not require the license.
> [Tr. 861:10-17 (Hinckley)]

Intel's response to the misuse arguments was testimony showing that its suspicions of copying were well-founded. For instance, Steven Domenik, an Intel engineer with some skill in the Japanese language, testified to a conversation in Japanese between two NEC employees:

> Q. And did you have a discussion at that meeting concerning the V20 and V30 products which you had been looking at?
> A. Yes.
> Q. And tell us as best you can recall, concerning that discussion, what was said by Mr. Yamamoto and Mr. Suzuki?
> A. We discussed the suspicion that I had arrived at . . . that the microcode had been produced by copying . . . .
> . . . .
> I raised that explanation of how one might do that by saying: And I think that is how you copied the 8086 microcode.
> . . . .
> Q. I want you to recite the conversation as you understand it and the words used by Mr. Yamamoto and Mr. Suzuki.
> . . . .
> THE WITNESS: I heard Mr. Yamamoto say 'so desu ka', which I understood to mean, 'is it like that,' with reference to the microcode.
> Mr. Suzuki, by his facial expression, his general demeanor and his answer in Japanese, which was 'so desu,' I interpreted his acknowledgment that he had used roughly the procedure I had outlined.
> [Tr. 774:11-20; 774:25-775:2; 777:16-778:4 (Dominek)]

These issues, along with infringement, remain to be adjudicated.

V. Conclusion

This case is about the specific microprograms embedded in the Intel 8086/8088 microprocessors. However, the issue of the copyrightability of microcode has much broader implications. The Court's partial decision that microcode is copyrightable has far reaching implications to the computer industry in general. Today's 32-bit microprocessors require an investment of hundreds of millions of dollars to develop. The court's partial decision gives additional protection to these large research and development investments because it is now clear that it is illegal to simply pirate the microcode to quickly come to market with a competing product. The decision puts potential copiers on notice that if they want to compete with the Intel 80286, 80386, the National 32,000 or the Motorola 68020, they must write their own microcode. The remaining issues are important to Intel with respect to the 8086 microprograms, but in the fast-paced semiconductor industry, these products are already being overtaken by new offerings.

Nevertheless the unresolved infringement issue remains one of the focal points of the trial. It is critical that the court not be persuaded that purported constraints allegedly imposed by NEC's choice of hardware justify a decision to copy microcode. As Jack Brown, counsel for Intel, summarized, "it is inconceivable that the copyright law would allow a competing manufacturer to bootstrap itself into a right to copy Intel's copyrighted microprograms simply by adopting the microarchitecture or microinstruction format or allocation of memory space."

To condone NEC's actions would invite a copier to disassemble a machine and its code, construct similar architecture in his own machine and then claim that, given that particular machine, he had no choice but to use substantially similar microprograms. Creators of original programs would find their copyrights completely circumvented. The incentive to produce microprograms would evaporate, and with it, a valuable source of productivity.