Volume 3, Spring Issue, 1990

NEC v. INTEL: BREAKING NEW GROUND IN THE LAW OF COPYRIGHT

Jorge Contreras, Laura Handley,* and Terrence Yang**

INTRODUCTION

The status of copyright protection for computer programs has long been in a state of confusion. In *NEC Corp. v. Intel Corp.*,[1] the U.S. District Court for the Northern District of California shed some light on three previously unresolved issues in this murky and continually evolving area of copyright. The court ruled that: (1) microcode embedded in certain Intel microprocessors constituted copyrightable material; (2) reverse engineering of the microcode did not infringe the microcode copyright; and (3) independent "clean room" development of similar microcode was persuasive evidence of non-infringement.

The execution of a computer program within a computer involves a number of different, operational levels.[2] An applications programmer may write a program to solve a problem in a high-level problem-oriented language containing familiar words, variables, and operators. Examples of high-level languages include BASIC, C, FORTRAN, COBOL, and Pascal. However, high-level languages cannot be implemented as such by a computer, which is controlled by the operation of digital circuits. Before instructions can be executed, a program must undergo a series of transformations that enable it to operate the computer's digital circuitry. The first step in this transformation may involve translation of the program by a compiler into an assembly-level program. Assembly languages generally reflect the internal organization and operation of the computer more than higher-level languages do, but are still incapable of directly controlling the computer.

The assembly-level program is then translated into long strings of binary numbers known as machine language, or object code, which is in turn manipulated by the computer's microcode. The microcode is a body of binary instructions that breaks higher-level instructions down to

---

* Harvard Law School Class of 1991. The authors wish to thank Douglas Derwin of Skjerven, Morrill, MacPherson, Franklin & Friel for providing valuable assistance.

1. No. C-84-20799, slip op. (N.D. Cal. 1989) (1989 WL 67434). An earlier decision was vacated when the District Judge recused himself after it was shown that he held shares of Intel stock. NEC Corp. v. Intel Corp., 645 F. Supp. 590 (N.D. Cal. 1986), *vacated sub nom.* NEC Corp. v. U.S. Dist. for N. Dist. Cal., 835 F.2d 1546 (9th Cir. 1988).

2. For a discussion of contemporary multilevel machines, see A. TANNENBAUM, STRUCTURED COMPUTER ORGANIZATION (2d ed. 1984).

the fundamental series of signals necessary to control the computer's circuits. Digital circuits, which are activated by high or low voltages, respond to the ones and zeros of the microcode. These microcode instructions move data in specific memory registers as dictated by the object code program.

Each microprocessor has its own set of microcode instructions kept in an area of memory called the "control store."[3] Unlike high-level language instructions,which are written anew for each applications program, microcode instructions reside permanently or semi-permanently within the microprocessor. Although each line of microcode generally comprises nothing more than a string of thirty-two zeros and ones, microcode designers often represent microcode functions using notation resembling assembly language notation (and called Micro Assembly Language or "MAL").[4] Therefore, microcode designers, claiming an analogy to higher-level programs, have registered their microprograms as "computer programs" with the Copyright Office.

I. *NEC v. INTEL*

A. Facts

The dispute in *NEC v. Intel* concerned NEC's alleged infringement of Intel's 8086 and 8088 microprocessor copyrights. On April 28, 1976, NEC entered into a patent cross-license agreement with Intel.[5] The license entitled either company to "make, use or sell" products based on the other's semiconductor patents. In 1979, NEC began to produce and sell Intel's 8086/88 series microprocessors.[6] While it manufactured the Intel chips, NEC used the 8086/88 designs to create a set of its own Intel-compatible microprocessors, the NEC V20 and V30 microprocessors. Such "hardware copying" was implicitly allowed under the license. However, a software engineer at NEC (Hiroaki Kaneko) involved in developing the V20 and V30 microprograms studied not only the

3. *Id.* at 126–34. The Intel microcode was stored in Read-Only Memory ("ROM") on the 8086 and 8088 microprocessors. NEC's Supplemented and Annotated Findings of Fact and Conclusions of Law (Copyrightability, Infringement, License, and Misuse) at 11, *NEC v. Intel* (No. C-84-20799) [hereinafter NEC Findings of Fact].

4. For example, the microcode instruction needed to add a number stored in register A to a number stored in the accumulator register ("AC") and to store the result in AC might be 1000 0000 0001 0001 0001 0010 0000 0000 or, in MAL, AC = A + AC. *See* A. TANNENBAUM, *supra* note 2, at 142–43.

5. *See* NEC Findings of Fact, *supra* note 3, at 3.

6. *Id.* at 4.

licensed hardware designs, but also the disassembled and listed[7] Intel 8086/88 microcode.[8]

The resulting V20/30 microcode bore a number of similarities to the Intel code. Both sets of microcode used the same "patch" to overcome a hardware "bug,"[9] both used the same memory registers in the same order in their RESET sequences,[10] and both handled internal errors in the same idiosyncratic way.[11] The presence of these and other similarities in the two microcodes prompted Intel to charge NEC with violation of its 8086/88 microcode copyrights. NEC brought suit seeking a declaratory judgment that the Intel microcode was either invalid or not infringed by NEC. Intel counterclaimed, alleging copyright infringement.

B. Copyrightability, Notice, and Forfeiture

In an opinion delivered by Senior District Judge Gray, the court first found that Intel's microcode was proper subject matter for copyright protection. The court interpreted the microcode as "a series of instructions," bringing it under the definition of "computer program" established by the Copyright Act Revision of 1980.[12] Thus, the microcode could merit copyright protection as a literary work. The court then found that the Intel microcode satisfied the two requirements for copyrightability of a literary work: that the microcode be fixed in a tangible medium of expression and that it contain at least a modicum of creativity.[13]

In the next section of the opinion, however, the court found that Intel had forfeited its initially valid copyrights by failing to place copyright notification on the distributed chips.[14] Collectively, NEC and other

---

7. Disassembly is the process of translating low-level machine language code into a higher-level form. Listing is the process of displaying such code in a medium readable by a human, such as a paper printout or computer display.

8. *NEC v. Intel*, slip op. at 24. This method of product development, whereby employees of one company determine how another company's product works in order to develop a compatible product of their own, is called "reverse engineering." *See* C. SHERMAN, H. SANDISON, M. GUREN, COMPUTER SOFTWARE PROTECTION LAW, 206–14 (1989) [hereinafter C. SHERMAN].

9. *NEC v. Intel*, slip op. at 25.

10. *Id.* at 28.

11. *Id.* at 33.

12. *See* 17 U.S.C. § 101 (1988).

13. *NEC v. Intel*, slip op. at 3. *See* 17 U.S.C. § 102 (a) (1988).

14. The relevant portion of 17 U.S.C. § 401(a) was amended by the Berne Convention Implementation Act of 1988 to read: "a notice of copyright . . . *may* be placed on publicly distributed copies." Berne Convention Implementation Act of 1988, Pub. L. No. 100-568, § 7(a), 102 Stat. 2853, 2857 (1988) (emphasis added). The Berne convention, ratified in the United states on March 19, 1989, eliminated notice requirements for copyrighted material.

Judge Gray, writing in 1989, did not mention the pending Berne Amendments to the

manufacturers of the 8086 and 8088 neglected to mark nearly three million of the twenty eight million distributed microprocessors (representing 10.6% of the total chips produced). Although Intel marked all of the chips it manufactured, the court determined that its failure to add notice to the chips made by NEC and others demonstrated a lack of concern for the copyrights. Thus, the court found that Intel had forfeited its once-valid copyrights.[15]

C. Infringement

The court went on to find that NEC's actions did not constitute infringement of the Intel microcode copyrights, independent of their validity or forfeiture. The court divided its infringement analysis into four parts: substantial similarity, copying, constraints, and idea versus expression. First, emphasizing the importance of "not . . . los[ing] sight of the forest for the trees,"[16] the court determined that the V20/30 microcode "as a whole" was not substantially similar to the Intel 8086/88 microcode.[17] Although some of the shorter NEC microroutines (segments of microcode) were quite similar to their Intel counterparts, those microroutines involved "simple, straightforward operations in which close similarity in approach [was] not surprising."[18] Further, none of the ninety microroutines were identical to the Intel version, and many were substantially different. Thus, the V20/30 microcode would not be recognized by an "ordinary observer" as having been taken from the 8086/88 microcode. Judge Gray found this conclusion nearly dispositive of the infringement issue.[19]

Second, the court determined that none of the NEC microroutines had been directly copied from Intel's microcode. Intel identified several compelling similarities between its microcode and the V20/30 microcode, and argued that these similarities constituted evidence of "slavish copying." The court, however, proposed two rationales for not viewing

---

copyright act. His strict adherence to the language of the pre-Berne § 401 seems harsh in light of Congressional intent to repeal this provision. However, it is possible that Judge Gray chose to rely strictly on the language of the 1976 Copyright Act, in order to avoid the possibility of reversal on this ground and to ensure the permanence of his other holdings. Although Judge Gray's holdings on reverse engineering, infringement, and clean rooms may be widely debated, the alternative finding of forfeiture made the prospect of ruling for Intel on appeal unlikely. Ultimately no appeal was filed.

15. *NEC v. Intel*, slip op. at 18.
16. *Id.* at 20.
17. *Id.* at 21.
18. *Id.*
19. *Id.* at 22.

these similarities as evidence of copying: (1) even if the first version of the V20/30 microcode ("Rev. 0") resembled Intel's microcode very closely, the only version of NEC's code which should have been compared to Intel's was the final "Rev. 2" version; and (2) reverse engineering of the 8086/88 microcode was permissible so long as the final V20/30 microcode was not otherwise copied from, or substantially similar to, the 8086/88 code. NEC's reverse engineering included conversion of Intel's microcode on the chip into a form easily read by humans. In contrast to prior decisions,[20] the court ruled that such unauthorized use of copyrighted material does not constitute infringement.[21]

The court approached the infringement issue in a third way through an analysis of the constraints placed on the NEC microcode by its hardware, the chip architecture, and the need for compatibility.[22] In order for NEC's hardware license from Intel to be of value, NEC had to develop microcode specifically tailored to the Intel 8086/88 hardware. The court relied heavily on evidence NEC presented that compared a "clean room"[23] program with both the V20/30 and Intel 8086/88 microcode. NEC hired an independent engineer (Gary Davidian) to develop a set of microcode for the V20/30 without access to any other microcode. Because Davidian's version of the microcode was similar in many regards to both the Intel and NEC microcodes, the court found it likely that those similarities were dictated not by copying of Intel's microcode, but rather by functional constraints of the hardware, the architecture, and the need for 8086/88 compatibility.[24]

Fourth, the court classified a number of Intel's shorter, simpler microroutines as unprotectable "ideas" rather than protectable "expressions" because of the limited number of ways those microroutines could be

20. *See, e.g.*, Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983) (holding that defendant Franklin's desire to achieve total compatibility, i.e. functionality, is irrelevant in determining whether the copyright on plaintiff's operating system has been infringed).

21. The court's permissive approach to reverse engineering may, in fact, be an unspoken or indirect admission that reverse engineering constitutes a "fair" and thus permissible "use" of copyrighted material.

22. Aspects of the V20/30 microprocessor dictated certain requirements for any microcode implemented thereon. For example, any microcode for the V20/30 required a "patch" to overcome the hardware bug the V20/30 inherited from the 8086/88. *NEC v. Intel*, slip op. at 25.

23. A "clean room" program is software developed by programmers who do not have access to the software they are trying to imitate. If programmers have no access to a copyrighted work, they cannot copy it.

24. *NEC v. Intel*, slip op. at 36.

expressed.[25] However, Judge Gray never stated precisely which aspects of the microcode were idea and which were expression. Thus, he failed to define the critical distinction between idea and expression which determines infringement.

II. ANALYSIS OF THE COURT'S DECISION

Copyright law is designed to protect only expression and not the underlying idea of a program.[26] Indeed, the Constitution mandates that the author's or inventor's proprietary interests are secondary to the benefit which society derives from "progress of science and the useful arts."[27] Therefore, Congress allows authors limited monopolies through copyright in order to make their ideas free to the public.

Since an idea embedded in microcode is inaccessible, Judge Gray correctly concluded that reverse engineering must be permitted in order to fulfill the Congressional and Constitutional mandates. Yet, the permissibility of reverse engineering creates a need for clean room evidence, which Judge Gray admitted. However, clean rooms are generally inefficient and burdensome to the industry: they are expensive, require extensive documentation, and cause unnecessary creative redundancy. Therefore, one must re-examine the propriety of the initial premise that microcode should receive copyright protection in order to guarantee the Constitutionally mandated societal benefit.[28]

A. Breakdown of Idea and Expression

NEC attempted to direct the court's attention to fundamental functional differences between microcode and higher-level computer programs. The court dismissed these considerations as "semi-semantical."[29]

As set forth in section 102(a) of Title 17 of the U.S. Code, copyright protects "original works of authorship."[30] The scope of protection incorporates both a functional element (in "authorship") and a creative

25. *Id.* at 37.

26. 17 U.S.C. § 102(b) (1988).

27. U.S. CONST. art. I, § 8.

28. The protection of object code may present similar difficulties. Although this discussion focuses on microcode, the debate over copyrightability of computer technology has a broader context.

29. *NEC v. Intel*, slip op. at 5. Judge Gray dismissed NEC's argument that the microcode comprised "part of the computer" and was therefore not a program and not copyrightable. He reasoned somewhat circularly that because the microcode fell within the statutory definition of a computer program, it should not be considered part of the computer.

30. 17 U.S.C. § 102(a) (1988).

element (in "original works"). Thus, copyright protection may be discussed using either "creativity" or "functionality" language.[31] The "functionality" perspective seems to be the more desirable one.

Focus on the creativity of computer "writings," whether software or microcode, is consistent with copyright treatment of traditional copyright subject matter such as books. Computer "languages," whether readable by humans or only by machines, involve just as much creativity as traditional copyright subject matter.[32] However, software may be *both* symbolic and mechanical.[33] Software is created through the "engineering process of problem or project definition, followed by designing the product (the program), creating a prototype (writing the source code), testing the prototype (debugging), and ultimately realizing the marketable product."[34]

The "creativity" analysis of software is consistent with the principle that copyright does not extend to expressions solely dictated by external constraints. If a given technology imposes such stringent limitations as to allow only one means of expression, there is no creativity involved in the use of that expression. Several commentators assert that the range of

---

31 Lack of clarity on this point has led to sharp disagreement amonst commentators. For example, Samuelson, *CONTU Revisited: The Case against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L. J. 663, emphasizes the functional aspects of machine-readable programs and is critical of the method of copyright protection suggested by the Commission on New Technological Uses of Copyrighted Works ("CONTU"), which discourages disclosure of underlying ideas. *See also* OFFICE OF TECHNOLOGY ASSESSMENT, INTELLECTUAL PROPERTY PROTECTION IN AN AGE OF ELECTRONICS AND INFORMATION 78–85 (1986) [hereinafter OTA STUDY].

The functional approach, which emphasizes the importance of "idea" and acknowledges the applicability of patent protection for software, is explicitly condemned by Clapes, Lynch & Steinberg, *Silicon Epics and Binary Bards*, 34 UCLA L. REV. 1493, 1541–43 (1985) [hereinafter Clapes]. They find the views of the OTA STUDY, *supra*, and Samuelson, *supra*, to be out of step with the perceptions of both computer experts and the courts. Clapes, *supra*, at 1536–41. Their lyric, celebratory view of computer creativity and expression is cited by Judge Gray in *NEC v. Intel*, slip op. at 20.

32. *See, e.g.*, Clapes, *supra* note 31, at 1536–38. Another commentator states: "[T]he software writer, like any writer, could intellectually move in a fantasy world of his own creating; one that operated according to a metaphysic known (possibly) only to him and limited only by the constraints of the language he worked in." J. LAUTSCH, STANDARD HANDBOOK OF SOFTWARE LAW 45 (1985). Lautsch argues that an inability to understand the software writer's symbolic virtuosity is characteristic of the "typographical mind." However, in emphasizing creativity, he ignores the "functionality" question; many things which involve creativity, such as mechanical inventions, are not proper subjects of copyright.

33. Davidson, *Protecting Computer Software: A Comprehensive Analysis*, 23 JURIMETRICS J. 337, 339 (1983).

34. *Id.* at 340. Davidson ultimately supports the copyrightability of computer programs, based on their symbolic content. The symbolic content makes such programs similar to works written in a spoken language.

software expression is so broad that it will never be so constrained.[35] However, microcode is much more dependent on external constraints than higher-level languages. The expression of microcode may frequently be dictated by chip microarchitecture[36] and necessary design requirements, as noted by Judge Gray in *NEC v. Intel*.[37]

The alternative and more desirable analytical perspective focuses on functionality in determining whether a work should be copyrightable. Computer technology is best viewed not as distinct hardware, firmware, and software, but as a functional continuum.[38] All computer technology reduces to the implementation of sequential logic functions and derives from similar creative thought processes. As described by Davidson, all computer technology is a blend of: (1) engineering and problem solving; and (2) symbolic representation.[39] Since computer programs, especially microcode, are functionally equivalent to hardware devices, it initially seems unreasonable to give copyright protection to such programs. However, computer programs are undeniably associated with written expression.[40]

Unlike traditional literary works, a computer program gains its value

---

35. *See, e.g.*, Clapes, *supra* note 31, at 153; J. LAUTSCH, *supra* note 32, at 33; *but see NEC v. Intel*, slip op. at 36.

36. Steinberg, *NEC v. Intel: The Battle over Copyright Protection for Microcode*, 27 JURIMETRICS J. 173, 182–83 (1987). ("The bulk of micro programs represent groups of microinstructions whose number and order of sequence is dictated solely by the microprocessor hardware. The programmer designing such a program can exercise no discretion in choosing the particular sequence of instructions to be performed by the desired task.")

37. *NEC v. Intel*, slip op. at 37. The issue turns in part on the recurrent problem of determining what exactly the distinction is between an idea and a copyrightable expression. Steinberg demonstrates how difficult this determination is by emphasizing that a microcode is functionally continuous with the chip microarchitecture, and that microcode "expression" can be shifted into the hardware, yet still comprise substantial nonliteral similarity. Steinberg, *supra* note 36, at 191.

38. *See generally* J. LAUTSCH, *supra* note 32, at 27–56 for a coherent and informative overview of computer technology. The continuity of hardware and microcode is dramatically demonstrated by the conclusion of Steinberg, *supra* note 36, at 193–94, that microcode is best protected by shifting its functions back into the chip circuitry and gaining protection under the Semiconductor Chip Protection Act of 1984, 17 U.S.C. §§ 901–14 (1988) [hereinafter Chip Act].

39. *See* Davidson, *supra* note 33, at 342. *See also supra* notes 3–5 and accompanying text.

40. Steinberg, *supra* note 36, at 191, predicted that if the court in *NEC v. Intel* did not fully appreciate the software continuum, it would overemphasize written similarities and fail to see substantial nonliteral similarities which could result from shifting minor code expressions back into the circuitry. The court might also fail to appreciate that the circuitry in effect can perform steps simultaneously. Steinberg suggested that copyright protection should be determined by looking to similarity of the programs when they are being processed, rather than to similarities in fixed form.

from its underlying "ideas," not the style or "expression" of its author.[41] Its "expression" in line-by-line code is often secondary.[42] The truly "creative" aspect of computer programs occurs at a level of expression that is beyond the traditional "literary" scope of the Copyright Act. Consequently, overemphasis of the "creative," which is to say, literary, aspects of software authorship permits inequitable results. An unscrupulous programmer can make merely minor changes in the "expression," thereby overcoming the creator's copyright protection while still appropriating everything of value in a program.

B. Reverse Engineering

Because the underlying idea of machine readable computer code is inaccessible when embodied in a microchip, copyright protection of the code contradicts one of its own central precepts, namely, that protection of expression is for the purpose of encouraging ideas to enter the public domain. A computer programmer wishing to access the unprotected idea of a microprogram can only do so by making a copy: that is, by decompiling the encoded instructions and reproducing the work. Such a reproduction generally constitutes an infringement of the copyright under section 106 of Title 17 of the U.S. Code.[43] Section 106(1) prohibits all unauthorized "copies" of a copyrighted work. Thus, a genuine issue exists as to whether NEC's initial reverse engineering product, Rev. 0, is a prohibited copy. Yet Judge Gray did not address the issue of whether

41. J. LAUTSCH, *supra* note 32, at 32–33. Of course, value is also derived from a particular programmer's "tricks" which may serve to make a program faster and more efficient. However, this observation emphasizes the ambiguity which results from the difficulty in defining idea versus expression. Structure, sequence, and organization may be considered to be either idea or expression, depending on the level of abstraction. *Compare* Whelan Assoc. v. Jaslow Dental Laboratories, Inc., 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987) (structure, sequence, and organization are copyrightable expression) *with* Plains Cotton Coop. v. Goodpasture Co. Serv., 807 F.2d 1256 (5th Cir. 1987) (structure, sequence, and organization are not copyrightable).

42. The final coding "can often be routinely entrusted to a beginning programmer." Clapes, *supra* note 31, at 1544. Thus, the most visible "literary" aspect, when viewed from Lautsch's "typographic mindset," *see supra* note 32, may be contributed by an anonymous programmer.

43. 17 U.S.C. § 106 (1988). *See* Hubco Data Products Corp. v. Management Assistance, Inc., 219 U.S.P.Q. 450 (D. Idaho 1983) (Defendant's production of a written printout of the unscrambled object code met the statutory definition of "copy," and was infringing to the extent that it was an unauthorized reproduction). The relief provided to computer users by the CONTU amendments does not create an exception for disassembly. *See* 17 U.S.C. § 117 (1988).

NEC's Rev. 0 infringed the Intel microcode.[44] Instead, he focused solely on NEC's final product, Rev. 2. As a result, Judge Gray's explicit approval of reverse engineering techniques may itself tolerate infringement by the initial decompiled program (in this case, Rev. 0).

Moreover, unrestrained reverse engineering is harmful to the computer software industry.[45] Innovators require returns on their research and development costs. However, if programs may be reverse engineered too rapidly, innovators may not have time to recover their costs, let alone earn minimal profits.[46]

C. The Costs of Clean Rooms

Another problem inherent in Judge Gray's approach is his unquestioning acceptance of NEC's clean room evidence. *NEC v. Intel* represents the first successful defensive use of clean room procedures to refute an allegation of copying.[47] The holding may create a willingness

---

44. The court did not explicitly address the issue; it appears that the issue was not pleaded by Intel's attorneys. To authorize the copy implicit in Rev. 0, Judge Gray may have considered Rev. 0 to constitute a "fair use" of the copyrighted material. If so, his holding would be a considerable expansion of the fair use doctrine, which generally permits copies of protected works to be made for non-commercial or academic purposes. *See* C. SHERMAN, *supra* note 8, § 210.5.

45. The Chip Act, *supra* note 38, tries to differentiate between "legitimate" reverse engineering and outright piracy by imposing an originality requirement. The precise scope of originality in this context has not yet been defined. *See* C. SHERMAN, *supra* note 8, § 509.4(C)(2).

46. *See* Mennell, *Computer Software Protection*, 39 STAN. L. REV. 1329 *passim* (1987). Mennell provides an in-depth economic analysis of the computer technology market, and is critical of the approaches taken to date by Congress and the courts. *See also* Spivock, *Does Form Follow Function? The Idea/Expression Dichotomy in Copyright Protection of Computer Software*, 35 UCLA L. REV. 723 (1986). A full discussion of the economic ramifications of microcode copyright are beyond the scope of this Recent Development; however, one brief criticism of Mennell's analysis is in order. It is not clear whether the social costs associated with *foreign* technology piracy are fully accounted for (NEC is the largest computer manufacturer in Japan). Factors to reconsider would include the reduced ability of U.S. companies to compete, stagnation of their research and development efforts, and the social costs resulting from loss of American jobs. These factors were a motivating force behind the Chip Act, which makes extension of protection to foreign works discretionary. *See* C. SHERMAN, *supra* note 8, § 501.3(b). *See also* Steinberg, *supra* note 36, at 175 (predicting that resolution of the *NEC v. Intel* case wil affect the national economy, and will govern the long-term availability, price, quality, and form of computer technology).

47. Prior to *NEC v. Intel*, two cases addressed the persuasiveness of clean room evidence. In Pearl Sys. v. Competition Electronics, 8 U.S.P.Q.2d 1520 (S.D. Fla. 1988), clean room evidence was used to show that similarities between plaintiff's and defendant's programs were not dictated by functional constraints, but were due to copying. In SAS Inst. v. S & H Computer Sys., 605 F. Supp. 816 (M.D. Tenn. 1985), the court found defendant's clean room to be inadequately insulated from access to plaintiff's program.

on the part of courts to accept clean room evidence of noninfringement. Until now, clean room development efforts have not typically been initiated until after the inception of copyright infringement litigation, as in *NEC v. Intel*.[48] Such "made for litigation" clean rooms are primarily evidentiary tools. The second type of clean room is a standard operating procedure ("SOP") clean room. Companies may try to avert litigation entirely by routinely developing compatible software in clean rooms.[49]

Companies in the business of producing compatible software run a high risk of litigation if they develop their software without clean rooms. If litigation ensues, they may be forced to incur the expense of a made-for-litigation clean room. As a result, it is cheaper to install clean room techniques prior to development, both in order to forestall litigation and to avoid the duplication of effort involved in developing a program and then developing it again in a clean room. Therefore, a trend toward greater use of clean rooms in the software industry should be expected as a result of decisions like *NEC v. Intel*.

Yet, SOP clean rooms would be a major burden on the software industry. Most hard-felt will be the record-keeping burden: In *NEC v. Intel*, the clean room documentation was "many thousand[s] of pages" long.[50] It included "every single piece of paper which Mr. Davidian saw," plus records of all his written and electronic communications.[51] Such completeness demands a large investment of time and resources.

The manpower burden is also quite large. Clean rooms require at least three groups of people: a specification team, a design team, and a coordination team.[52] These groups must all work together to develop the same program that one design team could have developed absent clean room requirements. Moreover, companies developing more than one program at a time will be required to maintain numerous clean rooms simultaneously. Elaborate precautions may have to be created to prevent developers from talking to one another about sensitive programs.

It may become increasingly difficult to find programmers with the

---

48. *See also Pearl Sys.*, 8 U.S.P.Q.2d at 1520 (where plaintiff used clean room evidence to demonstrate a different way to design a program).

49. A number of IBM PC clone manufacturers have begun to use clean room procedures in developing Basic Input/Output Software for their computers. So far, IBM has brought no cases against these manufacturers to litigation. *See* Davidson, *Reverse Engineering Computer Software Under Copyright Law: The IBM PC BIOS*, in OWNING SCIENTIFIC AND TECHNICAL INFORMATION 148 (V. Weil and J. Snapper eds. 1989).

50. NEC's Post-Trial Brief at 36, *NEC v. Intel* (No. C-84-20799) [hereinafter NEC Post-Trial Brief].

51. *Id.* at 33.

52. Derwin, *Licensing Software Created Under Clean Room Conditions*, in COMPUTER SOFTWARE 1989: PROTECTION AND MARKETING 439, 447 (M. Goldberg ed. 1989).

requisite programming skills who have never been exposed to subject programs.[53] There may ultimately be a shortage of qualified "neutral" clean room programmers. The alternative is an industry of enforced ignorance and tunnel vision. Programmers may intentionally avoid learning about competitors' products so that they will remain qualified to work in clean rooms developing clones of those products. Such a development could also stigmatize and render unemployable qualified programmers with wide-ranging experience.

The burden of maintaining clean rooms extends to every sector of the software industry. Purchasers and licensees of clean room software will eventually require warranties of clean room procedures in order to avoid their own liability for infringement. Auditing clean room documentation to ensure that proper procedures were observed could likewise become a massive financial and human drain on prospective buyers and sellers of software.[54]

The most obvious bearers of clean room costs are companies in the business of producing compatible software ("copiers"). The economic rationale for burdening copiers with the clean room cost is as follows: In order to duplicate the function of a copyrighted program without liability, a copier will have to install elaborate and expensive clean room procedures. This requirement forces copiers to pay development costs closer to those expended by "innovators." Such a disparity seemingly rewards innovators and forces copiers to pay a premium to benefit from that innovation. If copiers cease to find the duplication of programs profitable, they may decide to stop copying and perhaps they will rechannel their resources into more innovative efforts.

However, this rationale ignores the copyright protection already afforded innovative programs. If copiers duplicate only unprotected ideas (functionality) and not protected expression (implementation), then they infringe no copyright and their "copying" is permissible.[55] Imposing a clean room cost on the copiers of ideas expands the power of the copyright holder beyond the intent of Congress. A software designer forced to incur SOP clean room costs is actually penalized to the benefit of the copyright holder and to the detriment of consumers who might otherwise have profited from the competitive exploitation of an unprotected idea.

Moreover, innovators, too, suffer from SOP clean rooms. It is generally impossible to categorize real software developers as either

53. In *NEC v. Intel*, Davidian never had access to any microcodes created by either Intel or NEC, nor did he have access to any microcode implementing the 8086 instruction set. NEC Post-Trial Brief, *supra* note 50, at 33.

54. *See* Derwin, *supra* note 52, at 439.

55. *NEC v. Intel*, slip op. at 38.

innovators or copiers. In *NEC v. Intel*, NEC was not a mere "slavish" copyist.[56] NEC's V20/30 microcode improved on Intel's microcode in many ways. For example, the NEC clean room microcode made twenty-nine uses of the dual bus in the V-series hardware. The Intel 8086/88 had only a single bus, so its microcode made no such uses.[57] If clean rooms were standard practice, Intel would have to resort to a clean room to implement dual bus capabilities in its next generation of microcode. Then NEC might again have to resort to a clean room to avoid incorporating information from Intel's further improvement.

The clean room benefit to Intel, the so-called innovator, therefore expires after the first round of development. Then Intel, too, becomes subject to the high costs of the clean room procedure. And since nothing is completely new, all innovators will be driven almost immediately to the use of clean rooms if future courts make clean room evidence necessary by following the lead of *NEC v. Intel*.

A regime of SOP clean rooms hurts society as a whole. The clean room cost places a burden on all software producers. As a result, less money will be available for the development of programs. The software market in general will become less efficient and less productive. A regime of SOP clean rooms will also lower the quality of programs. Clean rooms limit the information with which programmers can work, resulting in an overall loss of programming effectiveness.[58] Programmers will be unable to learn from the mistakes of their predecessors.[59]

In addition, the clean room process creates a general duplication of effort. Not only does the innovator have to develop its microcode from scratch, but every other company wishing to avoid infringement must, likewise, start from scratch. Such a system wastes programmers' time that could be used to create new products.

---

56. *Id.* at 25.

57. *Id.* at 32.

58. The text of any copyrighted work is of public record. The court in *NEC v. Intel* ruled that even the inspection of a copyrighted work to get ideas for a new work is permissible, thereby permitting widespread use of programming information. However, the court's clean room holding inadvertently created a precedent which may severely limit the information programmers can access.

59. This result should be expected in an SOP clean room regime despite the affirmation of reverse engineering in *NEC v. Intel*. Clean room programmers by definition do not engage in reverse engineering.

CONCLUSION

The current status of copyright protection for computer microcode and software in general is inappropriate at worst and improvident at best. A *sui generis* method of protection for computer technology may be a desirable alternative. Such *sui generis* protection currently exists, but is limited to the designs of photographic "mask works" necessary for the production of semiconductor chips.[60] Since the microcode is so closely tied to chip architecture,[61] it seems rational to make their protection co-extensive under the Chip Act.[62] Similar treatment of operating systems and even applications software is also conceivable.

The Constitution requires Congress to maximize technological progress through its protection of intellectual property. However, the current regime of computer software protection may actually inhibit progress and harm society. The copyright protection of microcode may promote either excessive reverse engineering or widespread use of costly clean rooms. These possible consequences of *NEC v. Intel* strongly suggest that a new system of computer technology protection may be in order.

---

60. Chip Act, *supra* note 38. As in the Copyright Act, 17 U.S.C. § 102 (1988), the work must be "fixed" in the chip and must be "original." However, the degree of originality that the Chip Act requires is significantly greater than the de minimis requirement in the Copyright Act.

61. *See supra* notes 37–40 and accompanying text.

62. The Chip Act provides protection for only ten years. "Given the fact that the effect of any chip can be implemented through the software and that any software can be embodied in a chip, the different terms of protection [for software and microchips] poses an interesting legal problem." M. GEMIGNANI, COMPUTER LAW § 40:35, at supp. 311 (1985 & Supp. 1989).