Winter 1989


Mark M. Friedman

Follow this and additional works at: https://repository.law.uic.edu/jitpl

Part of the Computer Law Commons, Internet Law Commons, Privacy Law Commons, and the Science and Technology Law Commons

Recommended Citation

https://repository.law.uic.edu/jitpl/vol9/iss1/1

This Article is brought to you for free and open access by UIC Law Open Access Repository. It has been accepted for inclusion in The John Marshall Journal of Information Technology & Privacy Law by an authorized administrator of UIC Law Open Access Repository. For more information, please contact repository@jmls.edu.
COPYRIGHTING MACHINE LANGUAGE COMPUTER SOFTWARE—THE CASE AGAINST

By MARK M. FRIEDMAN*

Table of Contents

I. INTRODUCTION ............................................ 1
II. COMPUTER SOFTWARE TECHNOLOGY AND ITS PROTECTION ............................................ 3
   A. HISTORY .................................................. 3
   B. TECHNOLOGY ............................................ 4
   C. LEGAL PROTECTION ...................................... 5
III. COPYRIGHT INFRINGEMENT OF COMPUTER SOFTWARE ................................................ 7
IV. THE COPYRIGHTABILITY OF COMPUTER SOFTWARE 18
V. PROPOSALS FOR CHANGE ................................ 28
VI. CONCLUSION ............................................. 35

I. INTRODUCTION

Computer technology plays an increasingly important role in our lives. The computer industry has been growing at a phenomenal rate,1 paced in recent years by explosive growth in the software segment.2 The increasing importance of software has stimulated considerable competition, both legitimate and illegitimate. While the creation of original software products is a highly labor-intensive process, literal copying is exceptionally easy. The unscrupulous copier faces only minuscule bar-

* Author Mark M. Friedman attends University of Houston Law Center. This Article was awarded first place in the 1988 Center for Computer Law National Writing Competition.


2. In 1983, revenue from software sales increased at a rate 2.5 times the increase in hardware sales. Standard and Poor, Computer Hardware Industry, STANDARD AND POOR'S IND. SURVEYS (1984).
rers. As a result, software piracy has reached epidemic proportions, robbing the industry of billions of dollars in potential revenues. While software pirates limit themselves by producing a relatively limited number of copies for distribution to friends or to a small number of paying clients, in the aggregate, such piracy adversely impacts the software industry.

Cloning, a related yet distinct phenomenon, has emerged lately. To create a clone, a would-be competitor reproduces a successful software program with the aim of openly marketing the new program to the public. The reproduction is carried out after studying the functioning of the original and without literally copying it. A clone is not the product of literal copying. The competitor does not in any way use the original computer code. He simply purchases a software package at retail and studies those aspects of the program which are apparent to any legitimate user. The competitor then creates a new computer code which emulates the functioning of the original program, usually with some changes and enhancements. While recreating software in this manner requires great skill and a significant investment of time, it is considerably easier than developing original software.

Computer software currently enjoys several layers of legal and extralegal protection. The need to preserve incentives for developers of original software in the face of the relative ease of both literal and non-literal copying may motivate this protection. Copyright law is one form of protection. The 1980 amendments to the 1976 Copyright Act, and arguably the 1976 Act itself, afford copyright protection to software. Thus, courts have applied copyright law to computer software for the last decade, but with highly inconsistent results. This yielded a series of conflicting standards of copyright infringement. It also led to dislocations which threaten to skew the social bargain between software developers and society at large. This social bargain is the very underpinning of copyright law. Consequently, its distortion threatens to undermine copyright law as applied to non-software works.

This Article suggests that courts must struggle to apply copyright law to software, because copyright law is inherently inappropriate for software protection. The Copyright Act should be amended to exclude computer software in machine language form from its scope. Such exclusion will not leave software unprotected, since other means of protection, primarily trade secrecy law, are available. The alternatives to

3. Software programs worth hundreds or even thousands of dollars can easily be copied onto blank diskettes which can be purchased for a few dollars.
copyright protection more adequately balance the interests of rewarding software developers for their efforts while ensuring vigorous competition.

Part II of this Article provides a brief history of the computer software industry and a thumbnail sketch of software technology and its legal protection. Part III highlights some of the difficulties the courts encountered in applying copyright infringement standards to computer software cases. It also summarizes certain proposals for changes to the copyright standards. Part IV draws the conclusion that the underlying reason for the judicial uncertainty is that copyright law is inherently inapplicable to machine language software. This is because (1) such software is utilitarian in nature, and (2) it does not meet the constitutional requirement for public disclosure. Part V explores alternatives to copyright protection, ruling out patent protection and sui generis legislation. The Article concludes that machine language software is best protected under trade secrecy law. Trade secrecy affords a level and scope of protection which is most consistent with preserving incentives and encouraging competition.

II. COMPUTER SOFTWARE TECHNOLOGY AND ITS PROTECTION

To understand copyright law as it applies to computer software, the reader should be somewhat familiar with computer software technology. This section gives a brief sketch of the history of computer software, an overview of software technology, and an introductory overview of the legal protection of computer programs.

A. History

Commercial development of computer technology has occurred almost entirely in the post-World War II era. The early computers consisted entirely of hardware. Their circuits were physically configured ("hardwired") to carry out a specific task. Changing the task involved physically rewiring the computer's internals, a process both difficult and time consuming. By contrast, modern computers are truly universal machines able to perform a virtually endless set of tasks. While a


To date computer technology has passed through several basic technological phases. The four generations of computer hardware are based on vacuum tubes, transistors, printed circuits, and finally integrated circuits. NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 9-11 (1979) [hereinafter CONTU Report].
modern computer’s hardware configuration is largely fixed, the computer can obtain its instructions from user-written sets of instructions known as computer programs or software.\(^8\) Substituting software for much of the original hardware immeasurably increased the usefulness of computers and made them the all-pervasive technological workhorses of today.

Until fairly recently, software was developed exclusively by hardware manufacturers for distribution as part of the overall computer system. The present software development industry was born when the above practice, known as bundling, came to an end in the late 1960’s.\(^9\) Any person with the appropriate level of expertise could enter the software business with a relatively small capital investment. Opening the software industry to pervasive competition led to explosive growth in the software market and is largely responsible for the computer industry’s phenomenal growth.

B. Technology

Creating software is a multi-step process. At the outset, the software designer identifies objectives and selects a general approach. Then, he plans the program’s structure and organization, often using flow charts or word descriptions. Next, he transforms, or codes, the program structure and algorithms into computer language.\(^10\) Most programmers code using one of the high level computer languages. Computer instructions written in these languages are known as source code. High level computer languages consist of English-like commands which are easily comprehensible to programmers but which the computer cannot use directly. For the computer to use the software, the source code must be converted to machine readable form, variously known as machine code or object code, which is incomprehensible to the programmer. Source code is automatically converted into object code through the use of specialized computer programs called compilers or interpreters. Virtually all mass distribution, currently sold software is distributed in the form of machine or object code.\(^11\)

8. “A computer program is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.” 17 U.S.C. § 101 (1980).
10. At various stages in the software development, the programmer must test the program to insure that it operates properly and is error free. This debugging usually is more time consuming than the actual writing of source code. Conley and Bryan, A Unifying Theory of the Litigation of Computer Software Copyright Cases, 63 N.C. L. REV. 563, 566-67 (1985).
11. Some software packages are copy protected, or technologically locked, to make it
Software can be divided into three basic classes. Most familiar to the general public are application programs. Application programs can carry out particular, often complex, tasks such as word processing or database management.

A second class of software consists of operating systems. Unlike application programs, operating systems are largely invisible to the user. They control the housekeeping tasks of the computer, allocating memory as needed and routing signals from the keyboard and from application programs to the proper internal computer destinations. Operating systems interface between the application programs and the computer hardware, making it possible for a computer to accommodate a variety of application programs.

The third class of software is closely related to hardware. Computer hardware consists of a series of integrated circuits known as chips. Chips are highly complex, yet remarkably miniaturized, electrical circuits that perform a variety of basic functions. These functions are marshalled by the application programs, through the operating system, to achieve such complex tasks as word processing. While the integrated circuits, which make up the computer, are essentially hardware elements, chip designers usually allow for some flexibility by designing the capability of obtaining certain basic functions from a set of coded instructions into the chip. These instructions, called microcode, comprise the third basic type of software. Microcode consists of software instructions, yet it is used to change the basic operation of hardware elements. Hence, microcode falls somewhere between software and hardware. This ambivalence is reflected in the term firmware, which is often used to refer to microcode.

C. Legal Protection

Copyright protection is sanctioned by the Constitution. Congress passed the first copyright statute in 1790, but the last thorough revision was in 1976. The requirements for copyrightable works are considerably less rigorous than those for patentable works. To be

difficult for a user to create unauthorized copies. However, even software which is not copy protected is in compiled form which prevents the user from inspecting or making changes to the code.

13. “The Congress shall have Power . . . . To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries.” U.S. CONST. art. I, § 8.
copyrightable, a work must be original. Hence, it must have been independently developed rather than copied, and it must be fixed in some medium. The copyright law provides the author of a qualifying work with certain rights, such as the right to reproduce, to prepare derivative works, and to distribute copies.

Computer programs have enjoyed copyright protection for a number of years despite lingering doubts about the ability to copyright certain forms of software. In 1964, the Register of Copyright first accepted, albeit reluctantly, software for registration. Yet, somehow, Congress construed the Copyright Office's grudging acceptance as a wholehearted endorsement of the copyrightability of software. This misconception led to the enactment of laws formalizing the copyright-

16. "Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or a device." Id. § 102(a).

Commentators generally agree that "originality" should mean only that the work owes its origin to the author, i.e., is independently created and not copied from other works. . . . Originality sufficient for copyright protection exists if the "author" has introduced any element of novelty as contrasted with the material previously known to him.


The Act defines fixation as follows.

A work is "fixed" in a tangible medium of expression when its embodiment in a copy or phonorecord, by or under the authority of the author, is sufficiently permanent or stable to permit it to be perceived, reproduced, or otherwise communicated for a period of more than transitory duration. A work consisting of sounds, images, or both, that are being transmitted, is "fixed" for purposes of this title . . . if a fixation of the work is being made simultaneously with its transmission.


18. The registrability of computer programs involves two basic questions: (1) whether the program as such is the "writing of an author" . . . and (2) whether a reproduction of the program in a form actually used to operate or to be "read" by a machine is a "copy" that can be accepted for copyright registration. Both of these are doubtful questions. However, in accordance with its policy of resolving doubtful issues in favor of registrations whenever possible, the Copyright Office will consider registration for a computer program [to be protected by copyright laws] if:

1. The elements of assembling, selecting, arranging, editing, and literary expression that went into the compilation of the program are sufficient to constitute original authorship.

2. The program has been published, with the required copyright notice; that is, "copies" (i.e., reproductions of the program in a form perceptible or capable of being made perceptible to the human eye) bearing the notice have been distributed or made available to the public.

3. The copies deposited for registration consist of or include reproductions in a language intelligible to human beings. If the only publication was in a form that cannot be perceived visually or read, something more (e.g., a print-out of the entire program) would also have to be deposited.

ability of software and is indirectly responsible for the confusion evidenced in the courts in recent years. In 1980, on the recommendation of the congressionally appointed Commission On New Technological Uses of Copyrighted Works ("CONTU"),19 Congress amended the 1976 Act to establish unambiguously the copyrightability of computer software.20

While both the CONTU report and the 1980 amendments left unresolved which aspects of software were protectible, recent court decisions extend copyright protection to an increasing array of software formats. Courts apply copyright protection to source code, machine or object code, and object code embedded internally on semiconductor chips.21 Judicial decisions also extend protection to operating systems.22 Microcode was also recently awarded copyright protection.23

III. COPYRIGHT INFRINGEMENT OF COMPUTER SOFTWARE

Recent judicial copyright infringement decisions highlight the inappropriateness of protecting machine language computer software with copyright law. Infringement can be proved by showing that the plaintiff owned a valid copyright and the defendant copied the protected work.24 When available, copying may be shown by direct evidence. When evidence of direct copying is unavailable, as in most cases, copying can be established by showing that the defendant had access to the copyrighted work and that the allegedly infringing work is substantially similar to the original.25

Judge Learned Hand expressed the essence of the substantial similarity test in Nichols v. Universal Pictures Corp.:26

Upon any work . . . a greater number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of

19. CONTU Report, supra note 7, at 1.
22. Apple Computer, 714 F.2d at 1251.
25. Sid & Marty Krofft, 562 F.2d at 1162. But see Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1232 n.23 (3d Cir. 1986), cert. denied, — U.S. —, 107 S.Ct. 877 (1987) ("[E]ven the showing of substantial similarity is not dispositive, for it is still open to the alleged infringer to prove that his work is an original creation . . . . The cause of the substantial similarity—legitimate or not—is a question of fact.").
26. 45 F.2d 119, 121 (2d Cir. 1930).
what the [work] is about . . . but there is a point in this series of ab-
stractions where they are no longer protected, since otherwise the [au-
thor] could prevent the use of his "ideas," to which, apart from their
expression, his property is never extended.

Like numerous other legal concepts, substantial similarity lacks mathe-
matical precision. As Judge Hand himself noted, "[n]obody has ever
been able to fix that boundary, and nobody ever can."27

A modern description of the substantial similarity test can be found
in Sid & Marty Krofft.28 The court described a bifurcated test consist-
ing of two separate inquiry phases. In the extrinsic phase of the in-
quiry, the factfinder determines, with the aid of expert testimony,
whether the ideas of the two works are similar. If the works contain
similar ideas, the factfinder then inquires whether the expressions of
the two works are substantially similar. In this intrinsic phase of the
inquiry, the factfinder uses an ordinary reasonable person (or average
reasonable reader or spectator) standard and not expert testimony.29

In traditional copyright infringement suits, substantial similarity is
determined by viewing the disputed works through the eyes of an ordi-
nary observer without the aid of expert testimony.30 However, applying
the ordinary observer standard to computer software cases presents
many problems.31 Most juries and judges perceive computer technology
as complex and mysterious. The average lay person is unfamiliar with
computers. Even if the test utilized a computer-literate observer stan-
dard, it is unhelpful to the decisionmaker since the most important pro-
tectible element in a computer software copyright is not the visual
output, which an ordinary observer would see while using the program,
but rather the machine language code. In the form of machine lan-

27. Id.
28. See supra note 25 and accompanying text.
29. Sid & Marty Krofft, 562 F.2d at 1164. A problem may arise when the factfinder is
the same person in both phases of the test. In this case, "that person has been exposed to
expert evidence in the first step, yet she or he is supposed to ignore or 'forget' that evi-
dence in analyzing the problem under the second step. Especially in complex cases, we
doubt that the 'forgetting' can be effective when the expert testimony is essential to even
the most fundamental understanding of the objects in question." Whelan, 797 F.2d at
1232-33.
30. Nichols, 45 F.2d at 123. One of the goals of copyright law is to protect the origina-
tor of copyrightable material from those who would gain an unfair economic advantage by
improperly using his material. Allowing the ordinary observer to decide whether or not
two works are substantially similar places the decisions in the hands of the type of person
who would make the all important economic decision in the market place. Expert testi-
mony would merely confuse the issue by highlighting obscure similarities and differences
which may be totally irrelevant to the ultimate consumer.
31. The ordinary observer test has proven "one of the most difficult questions in
copyright law, and one which is the least susceptible of helpful generalization." M. Ni-
mer, supra note 24, § 13.03[A].
guage code, the protectible expression is incomprehensible not only to the ordinary person, i.e., the judge and the jury, but also to computer experts. Perhaps because they sensed this difficulty, several courts eliminated the ordinary observer requirement for computer software cases. One commentator proposed that the test be modified by using an "iterative" substantial similarity test which would substitute expert testimony for the ordinary observer test. Several courts adopted this modified substantial similarity test in software infringement cases.

Difficulties in applying the substantial similarity test to software are not limited to the inadequacy of the ordinary observer standard. A more fundamental difficulty exists. Under the traditional substantial similarity test, copyright infringement is not limited to literal and exact reproduction of the original work. The copying does not need to be comprehensive to be infringing. Rather, substantial similarity is found where the generalized ideas in the two works are the same and the expression is substantially similar. Taken at face value, a computer program infringes if it contains the same generalized idea as some pre-existing software and is expressed in substantially the same way.

While some courts accept this conclusion, other courts and many com-

---

33. Id. at 1285-88.
35. "It is of course essential to any protection of literary property ... that the right cannot be limited literally to the text, else a plagiarist would escape by immaterial variations." Nichols, 45 F.2d at 121. See also Berkic v. Crichton, 761 F.2d 1289, 1292-93 (9th Cir. 1985) (sequence of events in literary work can be copyrightable); West Publishing Co. v. Mead Data Cent., Inc., 616 F. Supp. 1571, 1576-77 (D. Minn. 1985) (arrangement and pagination of West reporters is copyrightable); Meredith Corp. v. Harper & Row, Publishers, Inc., 378 F. Supp. 686, 689, 690 (S.D.N.Y. 1974) (creating a psychology textbook by working from a detailed outline of another book infringes that work's copyright).
36. Nimmer identifies two ways in which a work might be substantially similar to another: comprehensive nonliteral similarity and fragmented literal similarity. Comprehensive nonliteral similarity occurs when there is "a similarity not just as to a particular line or paragraph or other minor segment, but [when] the fundamental essence or structure of one work is duplicated in another." Fragmented literal similarity occurs when there is occasional, but not complete, word-for-word similarity. M. NIMMER, supra note 24, §§ 13.03[A]-13.20.1-.2. "The question of the substantiality of the similarity is ... a question of fact; the piracy of even a quantitatively small fragment ("a rose by any other name would smell as sweet") may be qualitatively substantial." Roy Export Co. Establishment v. Columbia Broadcasting Sys., Inc., 503 F. Supp. 1137, 1145 (S.D.N.Y. 1980), aff'd, 672 F.2d 1095 (2d Cir. 1982), cert. denied, 459 U.S. 826 (1982).
37. Sid & Marty Krofft, 562 F.2d at 1164.
39. See infra notes 47-57 and accompanying text.
mentators find it very troubling. The application of the substantial similarity test to software is so problematic that many courts decline to use the test, creating instead ad hoc tests which they find more comfortable. The judicial confusion and uncertainty that result from the unclear tests have a highly deleterious effect on the computer industry.

Synercom Technology, Inc. v. University Computing Co. represents one of the earliest software copyright infringement decisions. Synercom marketed a highly successful structural engineering analysis software package. University Computing, wishing to market its own software to accomplish the same purpose, independently developed a competing program. To compete more effectively, the new software had to be compatible with the market-leading Synercom program. University Computing achieved this compatibility by adopting the Synercom input format. This format was adopted by structuring the program so that the sequence and format of its user inputs would be identical to those of the Synercom program. Therefore, it was possible for a user of the new software to run input files used with the Synercom program without altering the data. Synercom claimed that University Computing's use of the identical input format infringed its software copyright. The court disagreed, stating that form and expression constitute copyrightable expression in literary and artistic works but not in software.

If order and sequence is the expression, the skilled effort is not separable, for the form, arrangement, and combination is itself the intellectual conception involved. It would follow that only to the extent the expressions involve stylistic creativity above and beyond the bare expression of sequence and arrangement, should they be protected. In the usual case sequence, choice, and arrangement have only stylistic significance, rather than constituting as they would here, the essence of the expression.

The court in Whelan reached a contrary conclusion. Jaslow Dental Laboratories hired Whelan, a computer programmer, to create software that would automate the management of Jaslow. Whelan spent time with Jaslow personnel to learn the operation of the business.

---

40. One commentator argues that several courts have given the substantial similarity test only secondary consideration with primary attention being focused on the use of the original program to create a competing product. Compare Gesmer, Developments in the Law of Computer Software Copyright Infringement, 26 JURIMETRICS J. 224, 231 (1984).
42. Id. at 1006.
43. Id. at 1008.
44. Id. at 1009.
45. Id.
46. Id. at 1014.
47. Whelan, 797 F.2d at 1222.
She then created a software system and delivered it to Jaslow. Jaslow subsequently hired a second consultant to recreate the Whelan software to run on a different and more popular computer system. When Jaslow began marketing the new system, Whelan, who had copyrighted the original system, sued Jaslow for copyright infringement. Though Jaslow had access to the source code, his software was programmed using an entirely different language than Whelan’s in order to accommodate a different computer. Therefore, the two programs were quite dissimilar in literal detail. Whelan charged, however, that while Jaslow had not literally copied her program, he did copy its overall structure and organization. As a result, the two programs were very similar in their file structure, screen outputs, and subroutine configuration. The court paid lip service to the bifurcated substantial similarity test, but neglected to carry out the two part test. Instead, the court devised its own test. It asked whether the program’s structure and organization were essential to the expression of the program’s idea as the computerized management of a dental laboratory and found the particular expression of that idea in Whelan’s program just one of many possible expressions. The court concluded that since the structure and organization were not essential to the idea, they were protectible expression under copyright law.

The Whelan court distinguished the Synercom decision by noting that the structure in question in Synercom was much simpler than the structure in Whelan. The Whelan court implied that there may have been very few ways of arranging the input formats in Synercom. However, the Synercom court went out of its way to state that order and sequence do not constitute expression. The Whelan court disagreed citing as precedent the holding in SAS Inst., Inc. v. S & H Computer Sys., Inc. However, the Whelan court’s reliance on SAS may have been misplaced since more than mere structure and organization were involved in SAS. The SAS court was heavily influenced by its finding of literal copying of code by the defendant.

Other decisions on the copyrightability of structure and organiza-
tion are equally confusing and inconsistent. *O-Co Indus. v. Hoffman*,\(^6^9\) is factually similar to *Whelan*, yet the outcome was completely different. As employees of the plaintiff, the defendants developed a computerized speech prompting system for the plaintiff. Subsequently, they developed a similar program on their own. The plaintiff sued for copyright infringement. As in *Whelan*, the new system had little or no literal similarity to the original. It was developed for a different computer system and used a different computer language. The two systems displayed substantial similarity in structure and organization.\(^6^0\) Whereas the *Whelan* court held that such similarity constituted infringement, the *O-Co* court found that only idea and not expression had been appropriated, and hence there was no infringement.\(^6^1\)

Various courts have followed and rejected the *Whelan* rationale. The recent court of appeals decision in *Plains Cotton Cooperative Assoc. v. Goodpasture Computer Service, Inc.*\(^6^2\) specifically declined to follow the *Whelan* decision. The facts were similar to those in *Whelan* and *O-Co*. A former employer sued former employees who developed a competing software package using a different computer language for copyright infringement.\(^6^3\) Again, no direct copying was alleged or even possible. Rather, the plaintiff accused the defendants of “organizational copying.”\(^6^4\) The court, in rejecting Plains’ request for a preliminary injunction, chose to follow the *Synercom* reasoning rather than that of *Whelan*.\(^6^5\)

In another recent case, *Broderbund Software, Inc. v. Unison World, Inc.*,\(^6^6\) the court, purporting to follow *Whelan*, appears to have misconstrued *Whelan’s* holding. The controversy in *Broderbund* involved the alleged infringement of Broderbund’s copyright on its Print Shop software.\(^6^7\) The court concluded that the defendant infringed the Broderbund software copyright after considering the appearance and sequence of the two software systems’ menu screens.\(^6^8\)

The court explicitly expressed a preference for the *Whelan* holding.

---

60. *Id.* at 614.
61. *Id.* at 616.
63. *Id.* at 1259.
64. *Id.* at 1260.
65. However, the court tempered its preference by noting that since the motion before it was one for a preliminary injunction, it did not feel obligated to probe the depths of the case’s merit. *Id.* at 1262.
67. Both Broderbund’s Print Shop and Unison’s Printmaster software created custom made greeting cards. The Printmaster screens were virtually identical with those of Print Shop. *Id.* at 1137.
68. *Id.* at 1133.
over that of Synercom; however, the Broderbund analysis, based largely on the menu screens, confuses the issues. The screens are an independent creation and have little to do with the underlying program. The audiovisual outputs of computer programs are properly copyrightable, independent of the software which created them. It should make no difference whether a copyrightable entity is produced with the aid of a pen, pencil, or computer. The Broderbund court apparently misinterpreted the Whelan ruling as suggesting that “copyright protection is not limited to the literal aspects of a computer program, but rather that it extends to the overall structure of a program, including its audiovisual displays.”

The court in Digital Communications Assocs. v. Softklone Distrib. Corp. adopted an opposite view. It specifically rejected the Broderbund holding. A computer program copyright does not extend to the screens. The screens are separately copyrightable.

69. Id. at 1132. The defendant’s conduct may have influenced the court. The defendant cooperated with the plaintiff on a joint project, thereby gaining ready access to the plaintiff’s operation, and subsequently abandoned the joint venture to compete with the plaintiff. Id. at 1130-31.

70. See Davidson, Common Law, Uncommon Software, 47 U. Pitt. L. Rev. 1037, 1103 (1986) (arguing that what Whelan really held is that “when the source of a functionally similar program is the code of the original [program] similarities in the external aspects of the two programs is evidence of underlying similarity in the internal aspects.”).

71. Audiovisual works are defined as works that consist of a series of related images which are intrinsically intended to be shown by the use of machines or devices such as projectors, viewers, or electronic equipment, together with accompanying sounds, if any, regardless of the nature of the material objects, such as films or tapes, in which the works are embodied. 17 U.S.C. § 101 (1982). See also Kramer, 783 F.2d at 441; Strohon, 564 F. Supp. at 749. To be copyrightable, an audiovisual work must meet copyright requirements. Thus, in Broderbund, 648 F. Supp. at 1134, unlike in Synercom, 462 F. Supp. at 1014, the screens were “aesthetically pleasing” and contained “stylistic creativity above and beyond the bare expression.”

72. Broderbund, 648 F. Supp. at 1133. The Broderbund court understandably misconstrued the Whelan holding. The district court in Whelan noted that the two programs’ screen displays were very similar, although it did add that such similarities have only probative value. “Insofar as everything that a computer does, including its screen outputs, is related to the program that operate it, there is necessarily a causal relationship between the program and the screen outputs. The screen outputs must bear some relation to the underlying programs, and therefore they have some probative value.” Whelan, 797 F.2d at 1244.

73. 659 F. Supp. 449 (N.D. Ga. 1987) (plaintiff had copyrighted the manual and the programs as well as the display screens).

74. Id. at 455.
The court in Kramer took a more perplexing stand on audiovisual outputs of computer programs. The court reasoned that since the audiovisuals can be reproduced from the program, the latter must be a copy of the former. Thus the copyright on audiovisuals protects the ROM-based program as well as the audiovisuals.76

As evident from this brief anthology of recent cases, the substantial similarity test confounds many courts. The confusion has inspired various authors to suggest alternative tests. One author replaces the substantial similarity test with a "surprising similarity" test.77 Another commentator suggests that the courts abandon altogether the substantial similarity test: "Liability should not depend on substantial similarity between the copyrighted work and a particular version of the allegedly infringing work; rather, it should depend on whether the defendant has engaged in conduct that infringes any of the plaintiff's exclusive statutory rights."78

The confusion over applying the substantial similarity test to computer software intensified on January 12, 1987 due to three related events. First, the United States Supreme Court denied the petition for certiorari in the Whelan case. The other two events were the filings of

75. ROM stands for Read Only Memory. ROM consists of instructions which reside in the computer and cannot be changed by the user, i.e., they can only be read by the computer and cannot normally be erased or overwritten. Wharton, Use and Expression: The Scope of Copyright Protection for Computer Programs, 5 COMPUTER L.J. 433, 436 (1986).
76. Kramer, 783 F.2d at 445.
77. "[T]o determine underlying similarity, find evidence which reflects a surprising, improbable number of similar discretionary choices between two programs which are supposed to share only functional similarities." Davidson, supra note 70, at 1086.
78. Conley and Bryan, supra note 10, at 608. Basing infringement on conduct would focus on the alleged infringer's behavior and motivation. A number of courts seem to have based their decisions, in part, on the parties' conduct. For example, in Whelan the defendant had access to the source code, which made the similarity of the two programs highly suspicious. Whelan, 797 F.2d at 1232. Similarly, in Broderbund the plaintiff produced evidence of direct copying. Broderbund, 648 F. Supp. at 1135. In Kramer the two programs were virtually identical. The defendant had, however, included certain cosmetic dissimilarities which the court found to be the result of a deliberate effort to make minor distinctions between the programs. Kramer, 783 F.2d at 446. The court in E.F. Johnson found and carefully described a long series of similarities which had no rational basis for existence and which convinced the court that the original program had been copied. E.F. Johnson, 623 F. Supp. at 1493-98. In Williams v. Arndt, 628 F. Supp. 571, 579 (D. Mass. 1985), the court found infringement based largely on a weighing of the relative credibilities of the opposing parties. Finally, in SAS the court found the defendant's conduct faulty. S&H had obtained and systematically used the original SAS source code in a manner contrary to the terms of the licensing agreement. SAS, 605 F. Supp. at 821. But see Note, Defining the Scope of Copyright Protection for Computer Software, 38 STAN. L. REV. 497, 515 (1986) (criticizing the "conduct" approach of Conley, preferring a structural test, as in Whelan, over a course of development tests where the conduct of the parties as well as intermediate versions of the allegedly infringing software are included in the evidence).
a pair of copyright infringement suits by Lotus Development against producers of two clones of Lotus's highly successful 1-2-3 spreadsheet program. The Lotus suits accuse Mosaic and Paperback Software, vendors of The Twin and VP-Planner, respectively, of attempting to "deliberately recreate, with only trivial variations, the 'look and feel' and user interface of Lotus 1-2-3."80

The Lotus suits are similar to Whelan, but with one difference. In Whelan, the current high water mark in the application of copyright protection to computer software, the court based its decision on the substantial similarity of the organization and structure of the two programs. However, the court took notice of, and was probably influenced by, the fact that the defendant had access to the original program source code.81 In contrast, the defendants in the Lotus suits are not accused of having access to the 1-2-3 source code, and Lotus does not accuse the defendants of any breach of a confidential relationship. The relationship between the plaintiffs and defendants was that of a manufacturer and a retail purchaser of personal computer software.

The difference between Whelan and the Lotus suits, while ostensibly minute, is highly significant. Viewed in the context of the copyright protection of non-software works, the Lotus claim would not raise any eyebrows. With some modifications and enhancements, the two clones reproduce the 1-2-3 program. They seem to display the "concept and feel"82 of 1-2-3. Under traditional copyright law this process could easily constitute infringement. Jim Manzi, President of Lotus Development, stated that what the defendants have done is "no different than for someone to plagiarize Gone with the Wind, and then merely add a new concluding chapter."83


80. Info World, Jan. 19, 1987, at 1, col. 3. The suits allege that the defendant's products also copied "names used for various commands and functions ... and sequence." Id. at col. 4. Subsequent to filing the suits, the Copyright Office denied Lotus audiovisual copyrights on the 1-2-3 screens, because the screens consisted mostly of text, not graphics, and were not deemed unique. Info World, Mar. 9, 1987, at 1, col. 1.

In a related suit, Visicalc, the originator of the computer spreadsheet concept, filed suit against Lotus seeking $100 million for theft of technology and infringement of copyright of screens and images. Info World, Apr. 13, 1987, at 6, col. 1.

81. Whelan, 797 F.2d at 1232.

82. The term "concept and feel" first arose in Roth Greeting Cards v. United Card Co., 429 F.2d 1106, 1110 (9th Cir. 1970).

83. Info World, Mar. 30, 1987, at 8, col. 1. However, to Adam Osborne, President of defendant Paperback Software, Lotus' action is "a naked grab by the current industry leader to prevent legitimate price and performance competition." Id. at col. 4.
The Lotus suits are critically important, because should Lotus prevail, copyright protection will be extended for the first time to abstract functions, algorithms, business practices, and processes.\(^8\) However, this result would be inconsistent with sound public policy. The public policy reason for protecting computer software is to provide developers with proper incentives to produce software without, at the same time, giving them a stranglehold on the market.\(^5\) The social bargain between software developers and society at large must not be allowed to become unbalanced.\(^6\) An extremely loose standard for finding substantial similarity between programs will inhibit industry innovation and competition.\(^7\)

Computer software deserves a certain degree of protection. By nature, the actual writing of software code is relatively easy compared with the development of the overall structure, logic, and algorithms including the debugging, documentation, and maintenance.\(^8\) It would be grossly unfair to the original developer and would greatly inhibit software development to allow the literal copying and subsequent sale of software. A programmer should, however, be free to study a piece of software and to incorporate any ideas he may glean from normal usage into his own work.\(^9\)

\(^8\) The harder case than Whelan Associates would be where the defendant never looked at the code of the original program but nevertheless mimicked the user interface. No infringement of the underlying program should be found in such a case. Otherwise, copyright would be extended to protect more than the manner in which the program was written; it would then protect some of the abstract functions, algorithms, business practices, and process the program is automating.” Davidson, supra note 70, at 1103 (footnote omitted).

\(^5\) Whelan, 797 F.2d at 1237.

\(^6\) We must take care to guard against two extremes equally prejudicial: the one, that men of ability, who have employed their time for the service of the community, may not be deprived of their just merits, and the reward for their ingenuity and labor; the other, that the would may not be deprived of improvements, nor the progress of the arts retarded.

Whelan, 797 F.2d at 1235 n.27 (quoting Sayre v. Moore, 102 Eng. Rep. 138, 140 n.6 (1785)).

\(^7\) Because the computer software industry progresses by a “stepping-stone improvement process, with each innovation building on past innovations to produce an improved product,” too loose an application of the substantial similarity test could in effect require software engineers to “start from scratch” in order to achieve technological progress. Note, supra note 32, at 1291. Also, the courts must be careful not to protect material already in the public domain. “If the underlying work is itself protected by copyright, then the copyright in the derivative work or collection will neither nullify nor extend the protection accorded to the underlying work. If the underlying work is in the public domain, a copyright in the derivative work will not render the underlying work protectible.” M. Nimmer, supra note 24, § 3.04.

\(^8\) Whelan, 797 F.2d at 1231.

\(^9\) “[C]opyright protection for programs does not threaten to block the use of ideas or program language previously developed by others when that use is necessary to achieve a certain result. When other language is available, programmers are free to read copy-
If at all, most software programs improve the state of the art only by tiny increments. Progress in the computer field is evolutionary rather than revolutionary. The over-protection of software forces subsequent developers to reinvent the wheel each time they wish to make an incremental improvement in the technology. Over-protection breeds inefficiencies and discourages innovation while it induces technological stagnation. A programmer wishing to improve existing software is unlikely to undertake to create the original software. He knows that it will consume an inordinate amount of time to recreate, in an unrecognizable form, a system which already exists. He is also aware that even after painstakingly creating the new system, he is unlikely to sell enough copies to recoup his investment. Additionally, members of the user community will be reluctant to abandon their hard-earned skills in using the market's leading software to relearn new ways of accomplishing the same tasks with the new software. Compatibility is often crucial for the commercial success of new software because many users have large investments, not only in skills and expertise, but also in customized software developed using the commercial software. A potential user is unlikely to be interested in new software if it is incompatible with the market leader.

...righted programs and use the ideas embodied in them in preparing their own works.” CONTU Report, supra note 7, at 20. See also Affiliated Hosp. Prods., Inc. v. Merdel Game Mfg. Co., 513 F.2d 1183 (2d Cir. 1975) (no copyright infringement where the defendant did not copy game book verbatim and made “a good faith attempt to improve upon and clarify the presentation of the rules”).

Dan Bricklin, co-creator of Visicalc, the forerunner of Lotus 1-2-3, stated that most developers “are aghast at borrowing code; they wouldn’t consider doing that. In general, software advances by evolution. There are revolutionary steps but lots of things are done through evolution. It occurs whenever someone has to rewrite from scratch and that person’s ego gets involved. You never copy exactly; you always embellish because of your ego. That’s how we get the genetic mutation that makes evolution. Then you test the product in the marketplace to see if that mutation is good enough.” PC WEEK, May 1987, at 26. Recreating a program in a different computer language “requires substantial imagination, creativity, independent thought, and exercise of discretion, and the resulting program can in no way be said to be merely a copy or version of the problem statement. The program and the statement are so different, both in physical characteristics and in intended purpose, that they are really two different expressions of the same idea, rather than to different versions of the same expression.” Synercom, 462 F. Supp. at 1013 n.5.

90. The Synercom court makes this point by analogy with the familiar figure-H pattern for automobile gear shifts. Several other convenient configurations can be imagined. The original pattern selection was largely arbitrary but has long since become a standard. Use of such a standard is socially desirable, because it reduces the amount of restraining required. The pattern cannot be copyrighted, but descriptions of the pattern, like those found in car manuals, including photographs, are copyrightable. Synercom, 462 F. Supp. at 1013. According to Dan Bricklin, if Lotus should win, every software interface will have to be different. “The restraining costs alone are going to be in the billions of dollars.” Info World, Jan. 19, 1987, at 8, col. 1.

91. The defendant in Synercom designed its competing program to achieve compati-
While compatibility is desirable, literal software copying to achieve compatibility cannot be condoned.\(^{92}\) Compatibility can, however, be achieved without literal copying. Examples of compatible computer products that incrementally improve the technology of existing products abound.\(^{93}\) Extending copyright protection to the software's structure and organization, or look and feel, would greatly inhibit compatibility and standardization in the computer software industry. This extension also risks the smothering of an industry with tremendous impact on virtually every aspect of modern life. While the software industry would, if uninhibited, progress in a series of tiny technological steps building on previous knowledge and expertise, an ill-considered judicial decision could stifle significant innovation in the industry and create a pernicious atmosphere. In such an atmosphere, in the wry words of one observer, "[i]f you are going to see farther than others, make sure you're not standing on the shoulders of any giants."\(^{94}\)

IV. THE COPYRIGHTABILITY OF COMPUTER SOFTWARE

Applying copyright law to machine language software\(^{95}\) is inappropriate and poses the danger of corrupting and eroding longstanding copyright principles.\(^{96}\) Two fundamental constitutional difficulties arise in using copyright law to protect machine readable software. First, such software is a utilitarian work, not the constitutionally mandated "Writings."\(^{97}\) Furthermore, machine language code does not communicate to human beings, rendering moot the disclosure requirement which is part of the social contract forming the underlying constitutional basis for copyright protection—the promotion of "Science and Useful Arts."\(^{98}\)

An understanding of how machine language software became copyrightable is useful in analyzing the appropriateness of applying copy-

---

\(^{92}\) See SAS, 605 F. Supp. at 826. (By appropriating the SAS code, S&H avoided the cost involved with false starts. The benefits are not unlike those accruing from stealing the architectural plans of a building and proceeding to build according to the plans; see also E.F. Johnson, 623 F. Supp. at 1503 ("[T]he mere fact that defendant set out with the objective of creating an LTR-compatible radio does not . . . excuse its copying of plaintiff's code.").


\(^{94}\) Id. at 166.

\(^{95}\) The term "machine language software" is used here to include object code and microcode, but not source code. Source code is, and should remain, copyrightable.

\(^{96}\) See infra notes 109-55 and accompanying text.

\(^{97}\) See U.S. CONST. art. I, § 8.

\(^{98}\) Id.
right protection to software. Of course, the Copyright Act of 1909 was silent about the copyrightability of computer software. In 1964, the Register of the Copyright Office declared that copyright registration of computer software would be permitted under the rule of doubt, provided that a copy of the software, in human readable form, is deposited with the Register. The Register had serious misgivings about the copyrightability of software, specifically whether such works represented "writings" and whether machine readable versions were copies of the original program.

The 1976 Act, while not specifically mentioning software, effected changes which ostensibly removed certain obstacles in the way of the copyright protection for software. Specifically, the requirements of publication, registration, and direct human readability were significantly modified, almost deleted. In enacting the 1976 Act, Congress left open the question of the copyrightability of computer software pending the recommendations of the Commission On the New Technological Uses of Copyrighted Works ("CONTU"). The Commission did not issue its report until 1978. Congress utilized CONTU's recommendations in 1980 to amend the Copyright Act. While the Commission claimed not to have been bound by indications that Congress favored copyright protection for software, CONTU seems to have presupposed the copyrightability of computer software.

In its report, the Commission apparently limited itself to existing forms of protection and found copyright the least inappropriate. CONTU recommended extending copyright protection to computer software, but made little attempt to define precisely what forms of

100. Id. See supra note 18.
101. CONTU Report, supra note 7, at 15.
102. Id. at 15.
103. Commissioner Hersey contended that Congress had not clearly intended to extend copyright protection to software. CONTU assumed that the 1976 Act extended copyright protection to software when, in fact, Congress was looking to CONTU to render a judgment on the subject. Id. at 31 (Hersey, Comm'r, dissenting). Although the 1980 Act included a definition of computer program, it did not explicitly state that software is copyrightable. Note, supra note 1, at 246.
104. CONTU Report, supra note 7, at 16.
105. "[I]t is clear that those who wrote the Copyright Act of 1976 . . . concur in the position that programs are copyrightable. Action by either Congress or the courts would be necessary to change this." Id. at 16.
106. Id. at 16-18.
107. The new copyright law [should] be amended: (1) to make it explicit that computer programs, to the extent that they embody an author's original creation, are proper subject matter of copyright; (2) to apply to all computer uses of copyrighted programs by the deletion of the present section 117; and (3) to ensure that rightful possessors of copies
software should be protected. The Commission preferred to leave it to the courts to make fine distinctions among various manifestations of programs.\textsuperscript{108} CONTU took the position that copyright protection would not be denied to software simply because a work featured utilitarian aspects.\textsuperscript{109} This view disregards major tenets of copyright law and is contrary to previously expressed Congressional intent.\textsuperscript{110} Extending copyright protection to utilitarian works raises serious constitutional questions. The Constitutional mandate\textsuperscript{111} to Congress calls for the establishment of two distinct forms of intellectual property rights—one for useful works (patents) and one for writings (copyright).\textsuperscript{112}

of computer programs by use or adapt these copies for their use. \textit{Id.} at 1. Commissioner Hersey dissented. His recommendation was that "[t]he Act of 1976 should be amended to make it explicit that copyright protection does not extend to a computer program in the form in which it is capable of being used to control computer operations. \textit{Id.}

108. Most infringements, at least in the immediate future, are likely to involve simply copying. In the event that future technology . . . permits future infringers to use an author's program without copying, difficult questions will arise. Should a line need to be drawn to exclude certain manifestations of programs from copyright, that line should be drawn on a case-by-case basis by the institution designed to make fine distinctions—the federal judiciary. \textit{Id.} at 22.


110. The prohibition against copyright in useful articles is a fundamental principle of our copyright laws, adhered to for the nearly 200 years of their existence. In philosophical terms, the prohibition rests on the distinction between protection for expression and nonprotection for ideas under copyright, and on the differences in scope, standards, term, and purpose of the patent and copyright systems. In pragmatic terms, the nonprotection of useful articles that do not meet the patent standards of novelty and invention represents a societal judgment that the public benefits from relatively unhampered imitative copying of non-novel useful articles.


111. See U.S. Const. art I, \S 8.

112. "It is generally accepted that the [constitutional] clause [calling for intellectual property protection] is really two provisions merged into one which can be expressed as follows: The Congress shall have the power to,

1. Promote the progress of science by securing to authors the exclusive right to their writings.

2. Promote the progress of useful arts by securing to authors the exclusive rights of their discoveries.

At the time the clause was written the word "science" meant "knowledge" or "learning." Hence the first phrase refers to copyright protection while the second refers to patent protection. Kline, \textit{Requiring an Election of Protection for Patentable/Copyrightable Computer Programs}, 6 Computer L.J. 607, 646-47 (1986). \textit{See also} Graham v. John Deers Co., 383 U.S. 1, 5 (1966) (the Supreme Court, quoting the constitutional basis for patent rights, omitted references to 'science' and 'authors' as being "[t]o promote the Progress of . . . useful Arts, by securing for limited Times to . . . Inventors the exclusive Right to their . . . Discoveries.") (Omissions in the original).

The copyrighting of useful works tends to blur the distinction between copyright and patent and to grant patent protection without requiring the protected work to meet the
The Copyright Act defines a useful work as one having an "intrins-
cic utilitarian function that is not merely to portray the appearance of
the article or to convey information."\textsuperscript{113} Machine parts are clearly not
copyrightable. Machine language software, being more closely analo-
gous to machine parts than to writings, should not be copyrightable. All
the functions contained in a specific software program could be
hardwired into the computer. Early computers were all hardware.\textsuperscript{114}
Modern computers can become various machines by the simple inser-
tion of software. The software is thus a machine part, taking the place
of the electrical wires of yesterday. Granting copyright protection to
machine language software logically and inevitably leads to the conclu-
sion that hardwired circuitry, and possibly the computer itself, is copy-
rightable.\textsuperscript{115} In his CONTU dissent, Commissioner Hersey correctly
argued that software in its machine readable form has one object—to do
work,\textsuperscript{116} and therefore a computer program in mature and usable form
is a machine control element. As such, it should not be copyrightable
on constitutional and social policy grounds.\textsuperscript{117}

Machine readable software can best be seen as a useful article
through a simple analogy. Engineering drawings used in the construc-
tion of an automobile are copyrightable, since they are works which
only convey information. The car itself is the mechanical embodiment
of the drawings, and has many uses other than to portray an appearance
or convey information. Hence, the car is a utilitarian work and there-
fore not copyrightable.\textsuperscript{118} Construction of the car without the aid of the

\textsuperscript{114} See supra notes 7-8 and accompanying text.
\textsuperscript{115} Wharton, supra note 75, at 444.
\textsuperscript{116} "In its machine-control form [the computer program] does not describe or give di-
tections for mechanical work. When activated, it does the work . . . . [T]he instructions
themselves eventually become an essential part of the machinery that produces the re-

\textsuperscript{117} "Machine language software is intrinsically utilitarian in nature and its utilitarian
aspects cannot be separated from its non-utilitarian aspects." \textit{Id.} at 27 (Hersey, Comm'r,
dissenting).
\textsuperscript{118} See Imperial Homes Corp. v. Lamont, 458 F.2d 895 (5th Cir. 1972). (The defendant
constructed houses based on actual measurements made of plaintiff's model home. The
drawings does not constitute a copyright infringement on the engineering drawings.

Similarly, software in source code format easily qualifies as a copyrightable writing, since the source code does no useful work but merely conveys information to human beings. By contrast, once the source code is compiled into machine readable form, it ceases to convey information to humans and becomes directly usable by the computer to perform work. Hence, while source code clearly deserves copyright protection, machine language code clearly does not.\textsuperscript{119} The line of demarcation between copyrightable and noncopyrightable code must fall somewhere between source and object code. A logical demarcation would render code noncopyrightable as soon as it is transformed into a form having mechanical capability.\textsuperscript{120}

It is apparent from its final report that the Commission lacked a clear conception of the functioning of computer software.\textsuperscript{121} The Com-
mission, in attempting to clarify the distinctions between software and processes and methods, stated that copyright does not protect the “electromechanical functioning of a machine.”

Hence, one may make machines perform any conceivable process, except taking another’s program. Furthermore, according to CONTU, electrical impulses may be tapped, since they represent a part of a process which is not afforded copyright protection.

The trial and appellate court decisions in Apple Computer, Inc. v. Franklin Computer Corp. offer an interesting exposition of the utility issue. Apple, a large microcomputer manufacturer, sued Franklin, a fledgling producer of Apple-compatible computers, alleging copyright infringement of portions of Apple’s operating system. The operating system resided in machine code on chips in the Apple computer. In its defense, Franklin challenged the validity of Apple’s copyright, relying largely on the claim that the software in question was utilitarian in nature and thus noncopyrightable. The district court agreed with Franklin and refused to issue a preliminary injunction, thus allowing Franklin to continue marketing its computers. The appellate court overruled, holding that Franklin’s interpretation of Baker v. Selden to the effect that utilitarian works could not be copyrighted, had been overruled by Mazer v. Stein and rejected by CONTU. Furthermore, the appellate court found that computer programs are not machine parts but are machine instructions. The court saw no difference between instructions in a program and instructions in a manual.

Professor of Law, Harvard Law School
7) Robert Wedgeworth, Executive Director, American Library Association
8) Alice E. Wilcox, Director, Minnesota Interlibrary Telecommunications exchange Representing the public:
9) George D. Cary, retired Register of Copyrights
10) Stanley H. Fuld, retired Chief Judge of the State of New York and the New York Court of Appeals
11) Rhoda H. Karpatkin, Executive Director, Consumers Union
12) Melville B. Nimmer, Professor of Law, University of California at Los Angeles Law School
CONTU Report, supra note 7, at 4.
122. CONTU Report, supra note 7, at 20.
123. Id.
124. Id. at 22.
126. Apple II, 714 F.2d at 1243.
127. Id. at 1251.
129. 101 U.S. 99 (1879).
131. CONTU Report, supra note 7, at 21.
132. Apple II, 714 F.2d at 1251.
133. The court stated:
Since it is only the instructions which are protected, a “process” is no more involved because the instructions in an operating system program may be used to activate the operation of the computer than it would be if instructions were writ-
In conclusion, the court stated that if other code can be written that will accomplish the same purpose as the disputed programs, then the programs constitute expression rather than idea and are protectible under copyright law.\(^{134}\)

Apart from the utilitarian issue, machine language software raises the issue of disclosure of the copyrighted work. Humans cannot read machine language code. This should render such code noncopyrightable,\(^{135}\) since copyright legitimately protects only writings.

The term "writing" has been given a broad and ever expanding meaning.\(^{136}\) The CONTU majority also subscribed to the human readability requirement, but may have misunderstood the technology involved.\(^{137}\) Similarly, other commentators have followed this line of

---

134. Id. at 1253. See also Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d 521, 524 (9th Cir. 1984) (software's utility should not affect copyrightability).

135. Id. at 1228. See also E.F. Johnson, supra note 7, at 14.


137. The Commission considered the machine language version of a program to be a copy of the program. "But a program, when keyed or run into a computer, is transformed
reasoning.138 The Commission wrote that program instructions “may be read, understood, and followed by a human being.”139 Source code meets this description, but machine code decidedly does not.140 While CONTU stressed the human readability of copyrighted works, it simultaneously treated software as writings, thus dispensing with the fundamental requirement of disclosure.141 The copyrighting of a work lacking human readability troubled Commissioner Melville Nimmer.142 Nimmer realized that the extensive scope of protection offered by the majority might in the future prove over-restrictive and suggested instead that copyright protection be limited to software which was itself capable of producing copyrightable works.143 Although Commissioner Nimmer reluctantly concurred in the CONTU majority opinion, Commissioner Hersey dissented. Hersey contended that the “writing” requirement mandated that software not only act to communicate rather than do work, but that the communication be to humans, not to machines.144

It is quite possible that CONTU may have perceived machine language software to be human-readable, since it was ostensibly possible, with the aid of a device, to make the software yield its expression in human readable form.145 The Commission drew an analogy with videotape which when passed over a magnetized head, causes electrical currents to flow and regenerates the expression into a form which humans can perceive.146 The analogy, however, is inappropriate. While the videotape machine converts the information embedded in the videotape into a human readable expression, which is a precise recreation of the

by a compiler program into a purely machine state. The term copy is meaningless for the reason that in this transformation the means of expression of the original work become totally irrelevant. All that matters is the program’s functional use.” Id. at 32 (Hersey, Comm’r, dissenting).

138. Wharton, supra note 75, at 440.
139. CONTU Report, supra note 7, at 10.
141. CONTU Report, supra note 7, at 10, 20, 22.
142. In reluctantly concurring in the Commission’s recommendations, Commissioner Nimmer pointed out that the majority offered no rationale for its decision to copyright software. Nimmer warned that this action raised serious constitutional questions in that it risks turning the copyright law into a general misappropriation law. Id. at 26 (Nimmer, Comm’r, concurring).
143. Id. What seems to have swayed Commissioner Nimmer to concur with CONTU is the perceived urgency of providing formalized protection. Id. at 26-27.
144. Id. at 28-29 (Hersey, Comm’r, dissenting). Accord JS & A, 480 F. Supp. at 1068; Apple I, 545 F. Supp. at 821.
145. A similar misconception can be seen in Williams where the court noted that ROM-resident object code programs may be indirectly perceived by the user “with the aid of a machine or device.” Williams, 685 F.2d at 873.
146. CONTU Report, supra note 7, at 10.
The computer does not render the machine code readable by a human. The screens displayed by those software programs which display output are not an expression of the machine code. Rather, they are the result of the work the computer system performed.\footnote{147}

The failure to distinguish between the computer program and the audiovisual displays produced by many, but not all,\footnote{148} programs has greatly contributed to the confusion. The copyrightable work which, for purposes of copyright law, must be human-readable is the actual code, not the product or result of implementing the code on a computer. Computer program outputs are largely independent of the underlying code. Vastly different programs can create very similar outputs.\footnote{149} Conversely, a programmer can quickly alter an identical copy of a software program to produce vastly dissimilar screen or other outputs. While the audiovisual displays may superficially resemble displays of copyrightable expression, they in fact express none of the computer program instructions which are the intended subject matter of software copyright protection.\footnote{150}

The relative independence of computer software and program outputs poses a fundamental problem not encountered in previous technologies.\footnote{151} An integral part of the social contract underpinning patent and copyright laws is the idea that the inventor or author must disclose his work to the public in exchange for some degree of legally sanctioned monopoly. Such disclosure is a fundamental constitutional require-

\footnote{147. The functions of computer programs are fundamentally and absolutely different in nature from those of sound recordings, motion pictures, or videotapes. Recordings, films, and videotape produce for the human ear and/or eye the sounds and images that were fed into them and so are simply media for transmitting the means of expression of the writings of their authors. The direct product of a sound recording, when it is put in a record player, is the sound of music—the writing of the author in its audible form. Of film, it is a combination of picture and sound—the writing of the author in its visible and audible forms. Of videotape, the same. But the direct product of a computer program is a series of electronic impulses which operate a computer; the 'writing' of the author is spent in the labor of the machine. The first three communicate with human beings. The computer program communicates, if at all, only with a machine.  

\textit{Id.} at 29 (Hersey, Comm'r, dissenting).

\footnote{148. For example, operating systems produce no visible output.}

\footnote{149. \textit{Stern Electronics, Inc. v. Kaufman, 669 F.2d 852, 855 (2d Cir. 1982).}}

\footnote{150. For the most part, the typical user of computer software does not care to see the detailed instructions which comprise the program. The user is simply interested in the efficient functioning of the program.}

\footnote{151. The problem did not come up before the advent of computer software, since humans could perceive the expression of all previously copyrightable materials. Although neither the Constitution nor the Copyright Act explicitly calls for disclosure of copyrighted works, disclosure has always been an implied requirement of copyright law. \textit{See Baker, 101 U.S. at 103.}}
ment. Unlike trade secrecy law, copyright law is designed to encourage communication. However, currently nearly all copyrighted software is distributed in machine code form which has virtually no disclosure value. Paradoxically, a programmer can publish a copyrightable work and at the same time keep it secret. Some commentators are concerned that without disclosure copyright may become the primary means of enforcing trade secrecy.

The purpose of copyright law is to encourage the creation of copyrightable works by protecting the copyright owner from competition in exchange for his disclosure of the copyrighted material. Computer software is an undeniably useful technology and should be protected in some way. But is copyright the proper protection? The computer software industry is itself uncertain. While copyright protection is far less costly to obtain and maintain than either patent or trade secrecy protection, the computer software industry has been and continues to

---

152. Maer, 347 U.S. at 219 ("The economic philosophy behind the clause empowering Congress to grant patents and copyrights is the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare through the talents of authors and inventors in 'Science and the useful Arts.' ").

153. See D. Hofstadter, Godel, Escher, Bach: An Eternal Golden Braid 290 (1980). A machine language program consists of a highly complex array of millions of electrical impulses. With the aid of decompilers or disassemblers, it is possible to make some rudimentary sense of a program in machine language form. However, more often it is more manageable to simply start by observing the functioning of the program and reprogramming the functions in source code.

154. Ralph S. Brown, the noted copyright scholar, stated that "[t]he notion of secret copyrights is abominable." N.Y. Times, July 5, 1983, at 29, col. 3. However, the deposit rules make it quite easy to register software for copyright protection without disclosing anything of value to the public. For example, an author may seek "special relief" which is readily granted. Under such relief, the deposit requirements are satisfied provided the deposit is made in one of the following ways:

1. Deposit of the first and last 25 pages of the object code and any 10 pages of source code.
2. Deposit of the first and last pages of source code.
3. Deposit of the first and the last 25 pages of source code with the blocking out of any portion, so long as approximately 50% of the source code remains unblocked.


156. SAS, 605 F. Supp. at 818; see also Hubco, 219 U.S.P.Q. (BNA) 450 (relief was sought for both copyright infringement and trade secrecy misappropriation). Copyright law does offer the advantage of access to the federal courts which have exclusive jurisdiction. 28 U.S.C. § 1338 (1982). Allowing copyright law to protect trade secrets transforms copyright law into a general misappropriation law. See also Note, Copyright Protection of Computer Program Object Code, 96 Harv. L. Rev. 1723, 1741 (1983).


158. Obtaining copyright protection requires the author to (1) display the copyright no-
be reluctant to rely exclusively or even primarily on copyright for software protection. Surprisingly, for all its legislative sanction, copyright protection is used sparingly in the industry.

When CONTU issued its recommendations in 1978, it was satisfied that extending copyright protection to software was not likely to grant excessive monopoly power to software authors and would have an insignificant effect on the price of software products. CONTU realized that the industry was changing rapidly and urged that its recommendations be reviewed periodically to assess the impact on prices, competition, and cultural values. The time to reassess the extension of copyright protection to machine language software has arrived. A reassessment would reveal that copyright protection or machine language computer software is inappropriate. Furthermore, since other means, both legal and technological, exist for protecting software, copyright protection is also unnecessary.

V. PROPOSALS FOR CHANGE

Computer software should be protected, but only to an appropri-
ate extent. An appropriate method of protection would satisfy certain requirements. The method should proscribe unauthorized copying without inhibiting use or blocking development, and it should not give the author of the original program too much economic power.\textsuperscript{166} While safeguarding the author's investment, the protection should allow competitors to analyze the design and incorporate its concepts into their own works.\textsuperscript{167} Historically, Congress has expanded the scope of copyright protection to cover new types of works. Once bestowed on a class of works, copyright protection has never been withdrawn.\textsuperscript{168} To foster the software industry's growth, this trend should now be interrupted by withdrawing copyright protection from machine language software.

Copyright protection of machine language software is unnecessary as well as inappropriate. Withdrawing copyright protection from machine language software will not leave programmers at the mercy of unscrupulous competitors, since other means, both technological and legal, can protect such software.\textsuperscript{169}

One obvious option is to follow the current course taken by most courts by adjusting copyright law on an ad hoc basis to accommodate the fundamentally different situation presented by software. This approach is undesirable for two reasons. First, it consumes an inordinate amount of judicial resources and yields inconsistent results. Second, the excessive litigation is unfair to smaller industry members, since it can easily force them out of business by compelling them to defend an infringement suit.\textsuperscript{170} Inconsistent judicial results coupled with the threat of financially devastating litigation makes it very difficult for software developers to function efficiently and prevents potential developers from competitively entering the industry. Society pays for this diminished competition through higher consumer prices and technological stagnation.

Most commentators agree that patent protection is also not the answer.\textsuperscript{171} The case law is inconclusive. However, because of the substantial threshold requirements of novelty, particularly nonobviousness, patent law would be appropriate for only a minute fraction of all software products.\textsuperscript{172} Further, it takes too long, relative to the short

\begin{itemize}
  \item \textsuperscript{166} CONTU Report, \textit{supra} note 7, at 12.
  \item \textsuperscript{167} Shih, \textit{supra} note 112, at 137.
  \item \textsuperscript{168} CONTU Report, \textit{supra} note 7, at 15.
  \item \textsuperscript{169} Among the legal devices which may be used are contract, nondisclosure, trade secret, common law misappropriation, patent, etc. \textit{Id.} at 30 (Hersey, Comm'r, dissenting).
  \item See \textit{supra} note 166.
  \item \textsuperscript{170} Info World, Jan. 19, 1987, at 8, col. 1.
  \item \textsuperscript{171} See \textit{infra} note 166.
  \item \textsuperscript{172} By one estimate, fewer than one percent of all software systems are patentable. Davidson, \textit{Protecting Computer Software: A Comprehensible Analysis}, 1983 ARIZ. ST. L.J.
useful life of typical software products, to secure patent protection.\textsuperscript{173}

Creating a new body of intellectual property law designed specifically to accommodate software might inject a dose of certainty into software protection.\textsuperscript{174} Besides augmenting certainty in the software industry, such \textit{sui generis} legislation could arrest the distortion of copyright law as applied to non-software works. Such distortion has occurred as courts attempt to squeeze machine language software into the copyright law mold.

A \textit{sui generis} law was developed recently to protect semiconductor chips.\textsuperscript{175} The preliminary reviews are mixed. Integrated circuits form the core of computer hardware. While chip design requires a great deal of expertise and time, reverse engineering an existing commercial chip is fairly inexpensive.\textsuperscript{176} The 1984 Chip Act was enacted to provide chip manufacturers with some protection from exact duplication. While the Senate Committee recommended that a modified copyright law accommodate chip technology, the House Committee's recommendation for a \textit{sui generis} form of protection prevailed.\textsuperscript{177}

Opponents have two main objections to copyright law protecting chips. First, chips are useful articles and therefore ineligible for copyright protection. Second, even if chips were a proper subject matter for copyright purposes, the chip's protectible expression is inseparable from

\textsuperscript{611, 647} (1983). To be patentable, software must meet the standards of "usefulness," "novelty," and "nonobviousness." Patent statutes defined nonobviousness as follows. "A patent may not be obtained . . . if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains." \textit{35} U.S.C. \textit{\$} 103 (1987). "While a mathematic or functional principle incorporated in a program may satisfy these requirements, the software itself is patentable only if the manner in which the principle is utilized is also new and useful." Note, \textit{supra} note 1, at 239. Computer software may or may not be \textit{per se} patentable. The Patent Office at one time declared that "computer programming \textit{per se} . . . shall not be patentable." \textit{Examination of Patent Applications on Computer Programs}, 33 Fed. Reg. 15, 610 (1968). However, subsequent case law, though far from conclusive, raised the possibility that software may be patentable. See \textit{Diamond v. Diehr}, 450 U.S. 175 (1981); \textit{Parker v. Flook}, 437 U.S. 584 (1978); \textit{Gottschalt v. Benson}, 409 U.S. 63 (1972).

\textsuperscript{173} The patent application process takes an average of two years, not including the time required to draft the patent application. In comparison, the commercial lifespan of most computer programs is under three years. \textit{Id.} at 258.

\textsuperscript{174} Davidson, \textit{supra} note 70, at 1072.


\textsuperscript{176} This process is usually done by chemically stripping successive layers and photographing the exposed layers. Then, the photographs can be used as a blueprint of the chip.

\textsuperscript{177} \textit{See Note, supra} note 119, at 253 ("[T]he Act provides neither appropriate nor adequate protection for chips and that protection should come, if at all, from \textit{sui generis} law completely independent from copyright law.").
its idea, making chips unprotectible under copyright law. The same arguments can be made for denying the machine language software copyright protection.

Commentators also criticize the Chip Act on constitutional and pragmatic grounds. The Chip Act mixes elements of copyright and patent law. Scholars object to mixing these elements, because it creates an unconstitutional blurring of patent and copyright law. Beyond this fundamental difficulty, the Chip Act has been vehemently criticized because it fails to provide guidance as to when a chip infringes the protection of a predecessor chip. It improperly sidesteps the very problem it was designed to resolve. Indeed, an infringement standard is conspicuously missing from the Chip Act. Thus, it is unlikely that courts applying the Chip Act will find it easier to determine infringement in chips than to determine whether two software programs are substantially similar. Like the Chip Act, sui generis law, which is modelled largely on copyright law, cannot properly protect machine language software for the same reasons that copyright law is not effective.

While the software industry would prefer to retain copyright protection, it adamantly refuses to permit public disclosure of its software source code. Nondisclosure defies longstanding copyright principles. The software industry is unlikely to agree with a system that would require code disclosure. As a result, trade secrecy has been and continues to be the industry's overwhelming favorite form of protection. CONTU considered trade secrecy as the primary method of protection and rejected it due to its inherent "secrecy." CONTU preferred copyright protection because of its openness. However, the current software copyright system, with its provisions for limited disclosure, preserves secrecy to no lesser a degree than does trade secrecy.

178. Id. at 276-77. As with software, patent protection for chips has little use, since chips do not usually satisfy the novelty and nonobviousness standards. Unlike software, trade secrecy is totally ineffective for chip protection, since the secrets embedded in chips can easily be perceived by reverse engineering.

179. “[T]he Act offers copyright protection to utilitarian chips and grants unprecedented monopoly rights, similar to patents, without strict standards of review.” Id. at 292.


181. See supra general discussion of copyright law.


183. CONTU Report, supra note 7, at 17-18.
A comparison of copyright and trade secrecy law, as applied to software, reveals some important advantages to trade secrecy. The establishment of trade secrecy requires fewer formalities than copyright registration. No registration is required, and protection is available immediately. More importantly, unlike copyright, trade secrecy protection extends to the algorithms embodied in the software. The industry has found copyright protection unsatisfactory primarily because it does not protect the software's most valuable aspects. While copyright protects the expression, including the actual source code, the industry would also like to protect the program's design and logic, including the concepts, techniques, methods, and processes. Trade secrecy protects all of the above aspects.

Trade secrecy is defined in a variety of ways. While software is a proper subject matter for trade secrecy protection, applying trade secrecy protection to computer software has certain potential problems. One potential difficulty is the lack of uniform trade secrecy law among the states. Perhaps Congress, under its Commerce Clause power, should enact a uniform federal trade secrecy legislation. The law could serve to create a federal cause of action for computer program misappropriation—a Federal Software Trade Secrecy Act. However, congressional legislation is unnecessary to apply trade secrecy to software. Trade secrecy can effectively protect software even in the absence of such federal action. Trade secrecy laws in various states have similar major provisions. Finally, software vendors can include a choice of law provision in their license agreements which will unequivocally stipulate

---

184. Note, supra note 38, at 265.
185. Note, supra note 1, at 240; see also MacGrady, supra note 182, at 1051.
186. See Restatement (First) of Torts § 757 comment b (1934); MacGrady, supra note 175, at 1051, 1080; Bender, supra note 9, at 428; R. Nimmer, Law of Computer Technology § 3.06, 3-14, 17 (accord E.F. Johnson, 623 F. Supp. at 1051 n.17, 18, 19, 52 (1985)); J. McCarthy, Trademarks & Unfair Competition, § 10:25 322 (1973); Note, supra note 1, at 238.
187. Davidson, supra note 172, at 719.
188. See CONTU Report, supra note 7, at 17; Bender, supra note 9, at 447; Davidson, supra note 172, at 718-19. Employee mobility in the software industry is notoriously high. This may result in trade secrecy litigation when a valued employee leaves one company to join a competitor which then begins to market a product similar to one which the original employer had sold. Id. at 728. See also Goodpasture, 807 F.2d at 1256, Q-Co, 625 F. Supp. at 608 (weighing the difference between general skills of employees and trade secrets of the employer).
190. This law would be a type of sui generis legislation, but not based on intellectual property law. Davidson, supra note 172, at 746.
which state's law will control in case of conflicts.\(^\text{191}\)

Widespread distribution of software packages presents the most serious difficulty with trade secrecy protection of software.\(^\text{192}\) Trade secrecy does not protect information which has been disclosed to others through publication. The courts have not definitively resolved whether a software developer's sale of thousands of copies of a software program on the national market constitutes publication. The case law is sparse and inconclusive.\(^\text{193}\) However, most commentators argue that mass distribution of software in object code format is a distribution under secrecy conditions which does not constitute publication; therefore, trade secrecy protection is not negated.\(^\text{194}\) Distribution of software solely in object code form ensures actual and legal secrecy of the programs regardless of the extent of such distribution.\(^\text{195}\)

Unlike patent and copyright law, trade secrecy law has the potential to afford precisely the right level of software protection and to preserve the incentive to create for programmers. At the same time, trade secrecy law can avoid giving programmers so much protection that competition is stifled and programmers are allowed to reap inordinately large profits at the public's expense. Trade secrecy clearly encourages competition by allowing programmers to reproduce existing software, as

\(^{191}\) MacGrady, supra note 182, at 1077.

\(^{192}\) See Note, supra note 38, at 264; Note, supra note 1, at 240; Davidson, supra note 172, at 727.

\(^{193}\) See generally Technicon Medical Information Sys. Corp. v. Green Bay Packaging, Inc., 687 F.2d 1032 (7th Cir. 1982), cert. denied, 459 U.S. 1106 (1983) (trade secrecy preserved despite distribution of manuals with copyright marks); Management Science of Am., Inc. v. Cyborg Sys., Inc., 6 C.L.S.R. 921 (N.D. Ill. 1978) (no infirmity arising by virtue of mere proliferation); Videotronics, Inc. v. Bend Elecs., 564 F. Supp. 1471, 1476 (D. Nev. 1983) ("where such a computer program is made readily available to the public ... its contents may not be deemed a trade secret unless access to it is actually treated as a secret"). See also J & K Computer Sys., Inc. v. Parrish, 642 P.2d 732, 735 (Utah 1982) ("[t]hat a few of the plaintiff's customers had access to the program does not prevent the program from being classified as a trade secret where the plaintiff was attempting to keep the secret ... ").

\(^{194}\) Distributors desiring to retain a claim of trade secrecy enhance the force of their claim if distribution is in a form that tends to conceal or reduce access to the underlying secret. In the context of software, for example, if the secret resides in the underlying algorithm, distribution in object code, rather than source code form, is desirable since discovery of the underlying design is more difficult. R. Nimmer, supra note 186, at 3-18.

See also Davidson, supra note 172, at 726-27 (A solution to the problem has been to "market software by license only, requiring the licensee to sign a restrictive covenant acknowledging the trade secret status of the software, acknowledging restrictions on use, transfer or disclosure of the software, and requiring the licensee to take certain protective measures to preserve the confidentiality of the trade secret."); MacGrady, supra note 182, at 1063 ("[s]ecrecy will not be destroyed by the wide distribution of computer programs if they are distributed in object form only.").

\(^{195}\) MacGrady, supra note 182, at 1054-59.
long as they do so without misappropriating—without the benefit of seeing the source code or other confidential materials. However, does trade secrecy preserve sufficient incentive for the original developer who has spent much time and effort to create an innovative program?

A programmer can reproduce existing software at a considerably reduced cost. He avoids organizing a complex system and suffering through the false starts and experimentation inherent in the development of original software. He does, however, need to create a working program, which involves a considerable effort in coding and debugging. In recreating a program, the programmer is unlikely to reproduce the exact original program. He is virtually certain to make changes and enhancements, usually resulting in a better product. Because of his lower costs, he can undersell the original manufacturer. The critical question is, will the new social bargain leave the original manufacturer with sufficient incentive to ensure the continued production of new and original software?

Any software protection scheme must afford developers of valuable software the opportunity to recoup their investment. Fortunately, because of the technology’s inherently short lifespan, this goal can readily be accomplished without imposing extraneous legal constraints. The average lifespan of mass marketed software is extremely short, with a typical elapsed time of one year between the introduction of a popular original package and the issuance of an upgrade. Creators of original software will have the benefit of lead time, since contract provisions and trade secrecy law can ban direct copying and misappropriation.

---

196. Upgrades are enhancements to existing software which are issued periodically by the developer in order to improve the capabilities and performance of the original program.

197. The lead-time gained by an innovator who is first to market a good product, creates special profits. Indeed, those who are first enjoy a form of product monopoly that lasts until they are copied. (Footnote omitted.) Moreover, copying is seldom immediate; generally, the more sophisticated or involved [the product], the longer it will take to copy. (Footnote omitted.) More fundamentally, it is competition that provides much of the impetus behind innovation.

Note, supra note 119, at 288. Robert Noyce, one of the inventors of the chip and a founder of Intel Corp., speaking of chips, stated that “[a] year’s advantage in introducing a new product or new process can give a company a 25 percent cost advantage over competing companies; conversely, a year’s lag puts a company at a significant disadvantage with respect to its competitor.” Noyce, Microelectronics, 237 Sci. AM. 62, 68 (Sept. 1977).

The software manufacturer can take steps to prevent rote copying and other trade secret misappropriation. Under trade secrecy law the source code will not be registered, but the audiovisual screens may be copyrightable if they qualify. Stern, 669 F.2d at 855. Where available, this should provide effective protection against a competitor who copies outright. Requiring software developers to preserve for an appropriate period of time a ‘paper trail’ of their software development process also prevents literal copying. This could be used as evidence in any potential trade secrecy litigation. A legitimate researcher creates such a paper trail of his progress, including earlier versions of the pro-
Several months will elapse before a competitor is able to bring out a competing product, because it takes a certain amount of time for competitors to realize which programs are popular enough to warrant a competing product and to reproduce the original program. Once on the market, the new product will require additional time to become known and to gain market share. A year or more will elapse from introduction of the original product, and the original manufacturer will be introducing a product upgrade. If the competitor had simply reproduced the original software without making significant enhancements, his product will now be competing with a superior (albeit probably more costly) upgraded original product. If, however, the competitor made valuable enhancements, the original manufacturer will be forced to make significant modifications and/or to reduce the price of the new products. In any case, the lead time will preserve ample incentive for original developers by affording them a reasonable chance to recoup their investment, while the absence of copyright protection will give competitors the ability to compete freely. Such free competition will inevitably lead to enhanced social welfare through reduced consumer costs and increased software availability and usage. Thus, both the original developer and his competitors will be spurred to create reasonably priced yet significantly enhanced products.198

VI. CONCLUSION

The fundamental inappropriateness of copyright law to protect machine language software is the underlying cause of the confusion reflected in recent software copyright infringement court decisions. According copyright protection to inherently utilitarian machine language software programs violates the constitutional intent of keeping patent

gram, while a literal copier does not. Wilf, A Chip Off the Old Block: Copyright Law and the Semiconductor Chip Protection Act, 7 COMPUTER L.J. 245, 249 (1986).

A literal copier will be unable to compete with the original developer in user support, and will, of course, be unable to upgrade his product. Further protection can be obtained by including in the program code certain fingerprints or house marks—pieces or redundant or superfluous code whose only function is to serve as proof of literal copying. See Davidson, supra note 166, at 732; Bender, supra note 9, at 459. Such extraneous code and similar proof of copying may be enough to show misappropriation, providing that the proper level of secrecy and a confidential relationship have been established. R. Nimmer, supra note 179, at 3-53. In fact, recent court decisions have looked at such evidence of literal copying and may have based their decisions on such conduct to a greater extent than they would admit. See Kramer, 783 F.2d at 446 (hidden legend included in the original program was duplicated in the infringing program); SAS, 605 F. Supp. at 821 (redundant functionless code included); E.F. Johnson, 623 F. Supp. at 1493-97 (infringing program contained numerous instances of such "coincidental" similarities).

198. As a direct result, software vendors will compete among themselves for after-sale service and user support, usable manuals, and training as well as for programs.
and copyright law strictly separate and dangerously skews the social bargain between society and software developers. Application of copyright protection to machine language software creates uncertainty and discomfort in the computer software industry. This, in turn, stifles progress, reduces competition, and raises consumer prices. Collaterally, it distorts copyright law as applied to non-software works.

Copyright protection of machine language software is inappropriate and unnecessary. Adequate forms of protection, especially trade secrecy law, are already available. In particular, using trade secrecy law, as the software industry does, but without the unwarranted interference of copyright law is recommended. This will help effect an appropriate balance in the social contract between software developers and society at large. Trade secrecy provides incentives for developers to create more technically advanced software and promotes healthy competition in the industry.