Michigan Journal of International Law

Volume 8 | Issue 1

1987

The Limitations on the Protection of Program Works Under Japanese Copyright Law

Dennis S. Karjala
Arizona State University

The Limitations on the Protection of Program Works Under Japanese Copyright Law

Dennis S. Karjala*

I. Introduction

In June 1985 the Japanese Diet adopted amendments to the Japanese Copyright Act expressly extending copyright protection to computer programs.[1] The Act now provides detailed statutory provisions governing certain aspects of program modification, reproduction, authorship, and use.[2] Most importantly, the Act defines the term "program" and includes "program works" in the list of works expressly protected by copyright.[3] It limits, however, the scope of protection in

* Professor of Law, Arizona State University, Tempe, Arizona. Visiting Research Scholar and Japan Foundation Fellow, Tokyo University Faculty of Law, Tokyo, Japan, 1985-1986. The manuscript for this article was received in February 1986.

1. Copyright Act, Law No. 48 of 1970, *as amended by* Law No. 62 of 1985 [hereinafter Act]. For an English translation of the Act prior to the recent amendments, see Agency for Cultural Affairs, Copyright System in Japan (1983). Except for the recent amendments, all statutory quotations contained in this article are taken from this source. All translations of the recent amendments are by the author.

2. Act, *supra* note 1, arts. 15(2), 20(2)(iii), 47*bis*(2), 113(2). Article 15(2) now limits the moral right to preserve the integrity of a program work, and article 47*bis*(2) limits the economic rights of reproduction and adaptation. New article 15(2) governs authorship of programs produced in the course of employment. Article 113(2) limits commercial use of protected program works under certain circumstances. For general English-language discussions of the amendments to include computer programs as copyrightable works under the Japanese Copyright Act, see Doi, in this volume; Karjala, *Protection of Computer Programs under Japanese Copyright Law*, 4 Eur. Intell. Prop. J. 105 (1986); Robertson, *Copyright Protection for Computer Programs: The New Amendments*, E. Asian Executive Rep., July 1985, at 8; Sugiyama & Kosins, *The Japanese Solution to the Protection of Computer Programs*, 4 Law & Tech. Press 11 (September 1985); Sugiyama & Kosins, *An Amendment to Protect Computer Programs in Japan*, 2 Int'l Rev. of Computers, Tech. & L. 42 (1985); Takaishi, *The Perspective from Japan on Software Protection*, 1985 Int'l B.A. Sec. Bus. L. Conf. 1; Torii, *Legal Protection of Computer Software in Japan*, AIPPI Journal 150 (December 1985). For a discussion of the debate in Japan over whether to protect programs by amendment of the copyright statute or by adoption of a *sui generis* program protection scheme, see Doi, in this Volume; Karjala, *Lessons from the Computer Software Protection Debate in Japan*, 1984 Ariz. St. L.J. 53. *See also* Nakayama, *The Japan-U.S. Dispute over Software Protection*, 6 Econ. Eye 24 (1985).

3. Act, *supra* note 1, arts. 2(1)(x*bis*), 10(1)(ix). In most Japanese statutes, the various articles, paragraphs, subparagraphs, clauses and so forth are normally distinguished in references to them by

program works by providing that such protection does not extend to program languages, rules, or algorithms used in making such works.[4]

These express statutory limitations on the scope of protection in program works have important implications for a number of basic problems in computer software protection. One such problem is the scope of protection offered under Japanese law to operating systems software and microcode. A prohibition against copying any part of operating systems software or microcode could result in a *de facto* monopoly in the underlying machine or device akin to a patent. Moreover, this monopoly would be available without any showing of the nonobvious technological advance necessary for patentability and would extend well beyond the period of protection afforded under patent law. If it were decided, as a matter of policy, that copyright protection in operating systems and microcode should be limited to avoid device monopolies, the question is whether the statutory language of the Act is amenable to an interpretation that effects this limitation without removing the incentive to create such software.

A second problem involves the degree to which a programmer can make a reference to an existing copyrighted applications program in writing a new program designed to perform the same function. If protection of a program were to cover its general structure in addition to its detailed coding, a second programmer could not adopt that structure in creating her program. Such a protective rule is unrealistic in the light of the ways programmers actually work and, if enforced, would hamper technological progress, because programmers, like engineers of any other type, build and make improvements on a base of existing technology. The idea/expression dichotomy of traditional copyright law has long served to permit engineers to examine the copyrighted works of others to extract the technological ideas contained in these works for use in creating new products. In applying copyright protection to programs, the question is whether the Act can be interpreted in a way that permits technological advance without destroying applications program protection altogether.

A third problem is whether printing out or otherwise making a copy of a copyrighted program for purposes of reverse engineering, that is, for purposes of extracting its ideas for use in creating a new program (without duplicating its expression or using the copy in a computer), is an infringement of the copyright. In many cases, human inspection of the program is impossible without such copying. Consequently, if copying for this purpose were impermissible, all of the programming ideas contained in the program would be protected along with the programmer's original expression. This would extend the scope of copyright

different Chinese characters (*kanji*), so that the above citation, for example, translated literally from the Japanese, would read, "article 2, paragraph 1, subparagraph 10–2." This article follows the simpler style used by Professor Doi, in which paragraphs are shown by Arabic numerals in parentheses and subparagraphs by lower case Roman numerals in parentheses. *See* T. Doi, The Intellectual Property Law of Japan, ch. 7 (1980). References beyond subparagraphs are rare.

4. *See infra* text accompanying note 12.

protection well beyond its most important traditional boundary, that of expression of an idea rather than the idea itself.

This article examines these problems in the light of the program language, rule, and algorithm limitations on program protection under the Japanese Copyright Act. Section II sets forth the relevant statutory language, and Sections III and IV apply the program language and rule limitations to operating systems software and microcode. Section V considers the scope of protection under Japanese law in applications programs under the algorithm limitation on program protection. Finally, Section VI takes up the problem of whether copying for purposes of reverse engineering can be justified under the Act.

II. The Amendments to the Copyright Act

The Japanese Copyright Act protects "works,"[5] which are defined as follows:

> "work" means a production in which thoughts or sentiments are expressed in a creative way and which falls within the literary, scientific, artistic, or musical domain.[6]

The Act goes on to list various nonexclusive examples of works, such as literary works, musical works, and cinematographic works,[7] and the recent amendments include "program works" in this list.[8] The term "program" is defined in the Act as follows:

> "program" means a combination of instructions causing a computer to function so as to achieve a certain result.[9]

This definition may be incomplete,[10] but except when applied to microcode[11] it seems unlikely to cause too many problems as a practical matter. Consequently, a

5. Act, *supra* note 1, art. 6. The Japanese term is *chosakubutsu,* which may be translated more literally as "work of authorship" to emphasize that it is a term of art signifying those products of human endeavor that are eligible for copyright protection.

6. Act, *supra* note 1, art. 2(1)(i).

7. *Id.* art. 10(1)(i), (ii), (vii).

8. *Id.* art. 10(1)(ix).

9. *Id.* art. 2(1)(x*bis*).

10. A leading Japanese commentator, for example, has pointed out that computer programs include statements as well as instructions. Takaishi, *supra* note 2, at 20.

11. For a discussion of whether microcode or microprograms constitute "programs" under this definition, see *infra* text accompanying note 40.

It seems clear under this definition that object code is a "program" and can constitute a copyrightable work if the other conditions of copyrightability (e.g., creativity) are satisfied. This would normally be the case, for example, for programs written directly in machine language. Object code created from higher level source languages through compilers or interpreters would still seem to be programs under the Japanese definition, but as they are created mechanically from source code, they may lack the creativity necessary for copyrightable works. This is unlikely to be a practical problem, however, because even before the recent amendments several Japanese lower courts held, in the video game context, that object code so created is a copy of the source code and indirectly protected through computer program in which thought is expressed in a creative way would be a program work eligible for protection.

Article 10(3) of the amended Act contains explicit limitations on the scope of protection in program works. Article 10(3) provides that:

> With respect to works mentioned in paragraph 1(ix) [program works], protection under this Act does not extend to program languages, rules or algorithms used in making such works. For this purpose, the meanings of these terms are set forth in the following subparagraphs.
>
> (i) "Program language" means characters or other symbols, or their organization, used to express a program.
>
> (ii) "Rule" means a special convention concerning the use of a program language in a specific program.
>
> (iii) "Algorithm" means a method for combining computer instructions in a program.[12]

These limitations can be viewed as codifying for program works the traditional idea/expression dichotomy in copyright law by not extending copyright protection to "ideas" used in making a program that constitute a program language, rule, or algorithm. The scope of copyright protection for computer programs under Japanese copyright law depends in many crucial respects on the interpretation of these limitations. The following sections of this article apply these limitations to the problems discussed in the introduction: the protection of operating systems and microcode, reliance on the overall structure of an existing program in creating a new program that achieves the same function, and reverse engineering of programs.

III. Protection of Operating Software

A. *Review of Computer Technology*

An understanding of the basic concepts and terminology of computer technology is vital in developing schemes for its legal protection. A brief summary of

---

the source code copyright. Judgment of Dec 6, 1982, Japan (Dist. Ct., Tokyo), 1060 Hanji 18; judgment of Mar. 30, 1983, Japan (Dist. Ct., Yokohama), 1081 Hanji 125; judgment of Jan. 26, 1984, Japan (Dist. Ct., Osaka) 1106 Hanji 134; *see also* Takaishi, *supra* note 2.

It also seems likely that operating system programs will be considered "programs" within the meaning of this definition. However, the extent that such programs are protected under Japanese copyright law remains a major issue. *See infra* text accompanying notes 13–27.

12. Act, *supra* note 1, art. 10(3). The drafters of the recent amendments were not very precise either in choosing names for the limitations they created or in formulating their definitions, and a variety of translations have already found their way into the literature. Most commentators seem to agree that "program language" or "programming language" is appropriate for the first limitation (*puroguramu gengo*), but the "rules" (*kiyaku*) have also been translated as "codes," Cultural Affairs Agency, Japan Ministry of Education, Reference Materials Concerning Proposed Bill to Amend Parts of the Copyright Law 3 (Hodgens & Kakinuki trans. 1985) (unpublished work on file with the author), and as "syntax," Robertson, *supra* note 2, at 8. The

some of the fundamental elements is useful in setting the stage for the policy discussion.[13] For the purpose of interpreting the program language and rule limitations on program protection under the Japanese Copyright Act, it is important to understand that programs known as operating systems create a language in which application programs can be written.

As is well known, computer technology is divided roughly into "hardware" and "software." The hardware of any computer consists of memory units for storing instructions and information, input/output devices like video screens and printers, and digital arithmetic and logic circuitry. The logic circuitry is the heart of the computer and performs all of its "thinking" and "information processing."[14]

Although it is theoretically possible to build arithmetic and logic circuits of arbitrary complexity, the important thing to understand about actual logic circuitry is that it is *very* elementary. Instructions that add two binary numbers together, compare two binary numbers for identity or nonidentity, and store a binary word at or retrieve a binary word from a designated memory location are typical of the instructions that are available from the hardware. The elementary instructions available will vary with the machine and, taken together, are known as the "hardware instruction set" for the machine. A typical personal computer will provide a hardware instruction set containing from a few score to a few hundred elementary instructions.

The computer functions by executing sequentially, in accordance with an internal clock, lists of instructions from its instruction set that are stored by the programmer at appropriate locations in memory. The hardware instruction set defines an elementary language, usually referred to as "machine language," in which programs (lists of instructions) can be written. To write a complex program, like one that does word processing, using only machine language can be a mind-boggling task. At each step in the program, the programmer would have to know not only the current state of the computer and what he or she wants to do next but exactly where in memory every piece of data is stored so that it could be properly addressed when needed. Consequently, special programs, called operat-

majority of commentators seem to agree that the third limitation (*kaihoo*) means "algorithms," but literally it means "methods of solution," and at least one set of observers has translated it as "solutions," Hodgens & Kakinuki trans., *supra*.

13. For an introduction to the basic digital circuitry used in modern computers and its operation, see Holton, *The Large-Scale Integration of Microelectronic Circuits*, SCI. AM., Sept. 1977, at 82.

14. An electronic logic circuit takes binary electronic inputs, that is, sets of physical signals like voltages or currents that are either high or low, and produces a specific binary electronic output for each possible input. These circuits can be constructed so that they are in a one-to-one correspondence with statements and operations in propositional logic. For example, one operation of propositional logic is the "not" operation, in which an element Q is changed to its negative Q. This can be mimicked by an electronic circuit that produces 0 (low voltage or current) output whenever a 1 (high voltage or current) is placed at the input, and a 1 at the output whenever a 0 is placed at the input. Similar circuits can be built that mimic operations of ordinary arithmetic, like addition.

ing systems, are designed to relieve the programmer of these most mundane of computer housekeeping chores.

An operating system thus permits the programmer to develop programs at a higher level of abstraction than is possible working directly in the machine language defined by the basic hardware instruction set. In fact, operating systems themselves can be considered as multilevel sets of programs, each of which creates a language in which the next higher level program is written. The highest level permits the programmer to write programs in a more nearly human language like Basic, Pascal, or Fortran.[15] Thus, a given operating system will define its own instruction set in which application programs are written, but many more instructions may be possible than are available directly from the hardware; moreover, both the instructions and their execution can be quite complex, invoking many of the basic hardware instructions and many cycles of the computer's internal clock.

B. Policy Problems in Protecting Operating Systems Under Copyright Law

As briefly outlined above, an operating system essentially defines a language in which programs can be written. Because an operating system is designed to work with the particular hardware instruction set available with a given computer, different hardware usually requires different operating systems. This means that application programs written for a particular operating system generally will not work with that of a different computer. Moreover, the housekeeping chores carried out by an operating system can be done in a variety of ways, so more than one operating system is possible even for the same computer. Again, application programs written for one operating system generally will not be compatible with another, even though both operating systems can be used on the same computer.

Especially in the microcomputer area, however, many different computer manufacturers use the same basic microprocessing chips, that is, the same basic hardware, which in turn permits the use of the same operating system with different computers. Even when the underlying hardware is different, the creator of an operating system may seek to provide the user with the same instruction set defined by an operating system designed for another computer. In either case, the goal is the compatibility of application programs with the second computer. And

15. For an excellent discussion of these basic technological concepts, especially the various language levels created by operating software, see Sprowl, *Proprietary Rights in Programmed Computers: Looking Beyond the Hardware/Software Distinction for More Meaningful Ways of Characterizing Proprietary Interests in Digital Logic Systems*, 1983 ARIZ. ST. L.J. 785. For most legal purposes, the multilevel structure of operating systems can be ignored and the operating system regarded as a single set of programs serving as the interface between the hardware and the higher level language in which applications programs are written.

in both cases the problem is the extent to which copying all or some parts of the first operating system is necessary in order to achieve such compatibility.

The importance of compatibility can be seen by imagining a single computer manufacturer which is the first to enter the market and which achieves a dominant position. Many third-party programmers will write application software, such as word processing, video games, or statistical analysis programs, for that computer. A competitor's hardware or operating system may be much better in some technological sense, but except for true aficionados the competing computer is worthless without application software. Any competitor who seeks to enter the market must therefore either be compatible with the dominant manufacturer or develop huge amounts of application software independently. Compatibility is the obvious choice in most cases. Compatibility is also of considerable importance to consumers, because to the extent it exists they can shop for hardware by price and technological quality, without worrying about whether the application programs to which they have become accustomed will run on a particular computer.

Since it costs a great deal of money to develop an operating system for a given computer, it is manifestly unfair to permit a competitor to enter the fray by simply duplicating such a system electronically and selling it along with competing hardware. No operating system developer could ever recover development costs if that were permitted. Undoubtedly, thoughts like this underlie the Third Circuit's decision in *Apple Computer, Inc. v. Franklin Computer Corp.*,[16] in which the court held that operating systems are copyrightable under United States law. The court further stated that, provided the general functions performed by an operating system can be carried out in a number of ways, copying for the purpose of achieving compatibility would not be permitted. To the extent that copying is necessary for compatibility, however, the result is that Apple holds a monopoly position in machines that run Apple-compatible application programs. Further, Apple has this monopoly for a period much longer than would be available if Apple could patent its operating system and without meeting any of the more stringent requirements for patentability.

The Japanese are very much aware of the compatibility problems generated by recognizing a copyright in operating systems. This is at the heart of the dispute between IBM and Fujitsu, some aspects which have been discussed in the press.[17] By the time a Japanese court is asked to decide an issue of operating system copyrightability or the scope of protection provided by copyright, one should expect a greater understanding of the underlying economic and technological issues than was shown by the court in *Franklin*, especially as Japanese computer

16. 714 F.2d 1241, 1253 (3rd Cir. 1983), *cert. denied,* 464 U.S. 1033 (1984).

17. *E.g., The Software Dispute between IBM (U.S.) and Fujitsu—Is This the Fate of Compatible Systems?,* Asahi Shinbun, Oct. 23, 1985, at 9 (with subtitles "Unavoidable Copying, the Vagueness of the Concept of Copyright"); *"Copyright" and Fight—IBM (U.S.) Sues Fujitsu, id.,* Oct. 16, 1985, at p.4 (both in Japanese, titles translated by the author).

manufacturers are in a position analogous to Franklin's with respect to IBM in the worldwide markets. The next section discusses some approaches that Japanese courts might take in limiting copyright protection of operating software.

C. Protection of Operating Systems Under Japanese Copyright Law

1. Are operating systems "programs"?

Early commentators on the recent amendments to the Copyright Act agree that operating software falls within the general definition of "program," given above, and is thus within the purview of copyright law.[18] Certainly an operating system is a set of instructions that causes computer hardware to function in a certain way, but whether it causes a "computer" to achieve "a certain result"[19] seems, at the least, open to question.

As a technological matter, it is perfectly reasonable to view the hardware and the operating system together as a single general purpose machine—the "computer"—that achieves particular results (for example, statistical analysis) when coupled with particular application programs. The operating system alone, constituting only a part of the computer, does not cause the computer to achieve any particular result. When the hardware and the operating system together are thus viewed as the "computer," only application programs qualify as protected works under the statutory definition.

Because there is nothing outside of copyright that would protect operating software, viewing operating systems simply as a part of the computer would leave them completely unprotected. Consequently, notwithstanding that the technology and the statutory language can support this harsh result, we should expect the courts to go beyond such technical construction. If policy so dictates, this result could be avoided by viewing the hardware alone as constituting the "computer." Under this view, operating systems would be programs within the meaning of the Act, in that they are sets of instructions enabling the computer to accept application programs written for that operating system. An independently produced operating system will usually meet the other requirements for copyrightability, such as creativity, and would therefore also qualify as a program work under the Act.

2. Are operating systems "program languages"?

Even if an operating system is regarded as a program work under the Act, copyright protection will be denied under article 10(3)(i) to the extent that operat-

18. Abe, *Protection of Computer Programs under Copyright*, 843 JURISTO 38, 40 (1985); Bandoo, *The Copyright Law Amendments—Clarifying the Protection of Computer Programs*, 334 N.B.L. 18, 20 (1985) (both in Japanese, titles translated by the author).

19. Actually, achieving "a certain result" in the definition of a program, *see supra* text accompanying note 9, may be a mistranslation. The Japanese language definition in article 10(1)(ix) refers not to achieving *tokutei no kekka*, a "specific result," but rather *ichi no kekka*, which literally means "one result."

ing systems are considered "program languages." The question is, what constitutes a program language within the meaning of this statutory provision?

High level languages such as Basic, Fortran, and Cobol, are not programs within the new Japanese definition because they do not cause a computer to do anything, let alone achieve a particular result. Therefore, although Japanese commentators seem to agree that the exception for program languages covers high level languages,[20] there is in fact no need to except them from the scope of coverage in program works.[21] Thus, if such languages were the only ones covered by Article 10(3)(i), the limitation would be without meaningful content.

An operating system, however, does in fact define a language that permits programmers to work at higher levels of abstraction; it creates a set of instructions in which application programs can be written.[22] This set of instructions defined by the operating system is a program language, because it is through the organization of those instructions that one creates a program. Thus, the operating system itself could be regarded as a program language. After all, the program language limitation should be given some operative content.

The problem with treating an operating system as a program language under Japanese law, however, is that *all* legal protection for operating systems would be lost. It is not difficult to imagine the wrath of the United States government at any development that permitted Japanese computer manufacturers to steal a United States manufacturer's operating software, and with trade friction already severe, such a development is hardly likely.

There is, in fact, a more limited interpretation that still gives meaningful content to the program language limitation on the scope of copyright protection in program works. Under this interpretation, copyright law would protect an operating system as a program work, but the scope of protection would not extend to the program language defined by the operating system. When appropriately stored in a computer, the operating system determines what combinations of symbols the machine will accept to achieve a particular result. These combinations of symbols are a program language, and protection in a program work does not extend to such a language. If a competitor can create different operating system programs that accept precisely the same combinations of symbols and process them in exactly the same way, he has created the same "program lan-

20. Abe, *supra* note 18, at 41; Bandoo, *supra* note 18, at 20; Takaishi, *Protection of Computer Programs—The Copyright Amendments and Future Topics,* 850 Jurisuto 34, 36 (1985) (in Japanese, title translated by the author). Mr. Takaishi expressly argues that compilers and assemblers, which one can consider part of the computer's operating system, are program works and that loose references to them as "languages" should not limit the degree of protection afforded them.

21. Article 10(3) only limits protection in "program works;" it is not a general exception from copyright coverage. Therefore, it is difficult to maintain that article 10(3)(i)'s exception for "program languages" only restates the general principle that languages are not protected by copyright. If a particular program language could somehow fall within the scope of another type of copyrightable work, article 10(3)(i) would not deny or limit protection.

22. *See supra* text accompanying notes 13–15.

guage" but with different operating system programs. The exception for program languages means that the creation of a different operating system that deliberately reproduces the program language defined by the first system does not infringe the copyright in the first system. This analysis gives substantive meaning to the program language exception without completely removing protection from operating system programs.

An important corollary follows from this analysis: there would be no infringement of the copyright in an operating system to the extent that copying is necessary to achieve compatibility with application programs written by third parties. This follows naturally from the nature of operating systems and the program language limitation on copyright protection in program works. Third-party programmers write their programs in the program language defined by the original operating system. A fully compatible competitor machine must accept *all* such programs, which means it must accept an identical instruction set and process each instruction in the same way as the original machine.[23] Depending on the particular hardware and operating system involved, it may be that the number of ways to arrange to accept particular combinations of symbols for identical processing is limited. To the extent that is the case, the copyright in the original operating system should not prevent copying of those aspects of the operating system programs, because to do so would extend protection to the language defined by those programs, which is expressly excluded from protection under article 10(3)(i).

Consequently, a reasonable approach to the "program language" limitation would be to treat operating systems as protected programs but to permit copying to the extent necessary for application program compatibility, that is, to the extent necessary to create the same "program language." This approach would insure that any competitor seeking to market a compatible computer would have to put at least as much work into the creation of his operating system as the original manufacturer.[24] He can be asked to produce the voluminous "paper trail"

23. This is not intended to suggest that the detailed information flow inside the computer must be the same for each instruction. Rather, it means that the processing must *appear* identical from the point of view of the outside programmer. To the extent each instruction in a program is executed completely independently of every other, simply providing the same instruction set to the programmer should insure that her program will run on either computer, because she will be using only allowable instructions for each one. If, however, the programmer is very sophisticated and makes use, for example, of the time it takes a particular computer to execute particular instructions in choosing the order in which instructions are included in the program, the program may not run on a second computer that does not execute those same instructions in the same amount of time.

24. It is possible that even more work may be involved. The original creator of an operating system can make a number of arbitrary choices (for example, the location in memory at which a particular subroutine begins) that, once made, become necessary features of any compatible system. These decisions, which are inexpensive initially, become very time consuming, and therefore costly, to trace for the purpose of achieving compatibility. *See* Karjala, *Lessons from the Computer Software Protection Debate in Japan,* 53, 63–64 n.35.

that necessarily accompanies the development of an operating system and to justify, on the ground of necessity for compatibility, any exact duplications. This means that the competitor will not have an unfair economic advantage over the original creator of the operating system, while at the same time insuring that the original creator does not maintain a monopoly over third-party software written for his computer for the long period of copyright protection.

3. Are operating systems "rules"?

Of the three limitations on the scope of copyright protection in program works provided for in article 10(3), perhaps the most difficult to understand is that for "rules." As set forth above,[25] a rule is defined to be a special convention concerning the use of a program language in a specific program. A representative of the Cultural Affairs Agency, which administers the Copyright Act, has explained the meaning of "rules" as follows:

> In making a program, in addition to the conventions applicable to a program language, it is sometimes necessary to follow specific conventions for the purpose of using the program in connection with a different program in the same computer or with a program in another computer through the medium of communication circuits. All these conventions are included within the term "rules."[26]

A leading Japanese commentator has interpreted this explanation to mean "interface" conventions but still finds the meaning to be somewhat unclear.[27] Particularly troublesome is that, of the three limitations on protection in article 10(3), only "rules" are restricted by reference to use in a "specific program" rather than in programs generally.

With respect to operating systems, one possible interpretation giving meaningful content to the "rules" limitation would be to treat operating systems as unprotected to the extent they must be copied to achieve compatibility. An operating system is, in fact, the interface between an application programmer and the underlying hardware. As discussed in the previous subsection, the operating system defines a language for writing programs, and in that sense it defines a set of special conventions concerning the use of that language. Compatibility means following these same conventions in linking the operating system to third-party application programs. If these conventions are unprotected as "rules," copying them would not infringe any copyright in the original operating system programs.

Interpreting operating systems as creating conventional "rules" linking them to third-party application programs adds little if anything to the analysis of the "program language" limitation discussed above. Such an interpretation is therefore unnecessary in achieving the social policy goal of permitting copying of operating systems as necessary for compatibility. Moreover, the set of special

25. *See supra* text accompanying note 12.
26. Bandoo, *supra* note 18, at 20 (translated by the author).
27. Takaishi, *supra* note 20, at 36.

conventions defined by an operating system is general and is used in creating all application programs. Thus, this interpretation essentially ignores the restriction of "rules" to conventions concerning program language use in a specific program. Consequently, the "rules" limitation on protection as a program work does not seem to play a pivotal role in determining the scope of copyright protection in operating systems.

4. Summary

As a policy matter, there are good reasons for limiting the scope of copyright protection in operating systems software when to do otherwise would create a *de facto* monopoly over application programs created for a particular computer. On the other hand, it would be plainly unfair and a long-term social loss (because of the resulting disincentive to produce such software) to deny protection to operating systems completely. When the case arises in Japan, a Japanese court will surely be sensitive to the latter concern. Whether it will ignore the former and follow the United States' lead in prohibiting almost completely any copying of operating systems programs is difficult to predict.

The Japanese Copyright Act, however, is amenable to an interpretation that permits demonstrably necessary copying for achieving compatibility while still requiring competitors to make a significant investment in developing their competing operating software. That interpretation treats an operating system as defining a program language; where such program language (which is not protected) cannot be exactly reproduced except by copying portions of the original operating system, the language and the operating system merge. To that extent, article 10(3)(i) denies protection in the operating system programs.

IV. Protection of Microcode

A dispute between two of the world's largest manufacturers of semiconductors, one American and one Japanese, centers on the scope of copyright protection in microcode.[28] The dispute is important to microcomputer users as well as to the companies involved, because it involves an attempt by the Japanese company NEC to market a microprocessor compatible with Intel's 8088 and 8086 microprocessors, which constitute the heart of the IBM PC and other compatible computers. The NEC microprocessor is said to be able to run all programs written for the Intel chip, with extra advantages in terms of speed and versatility.

28. NEC Corp. v. Intel Corp., No. C-84-20799-WAI (N.D. Cal. filed Feb. 14, 1985). Brief descriptions of this dispute are given in Duffy, *Intel Court Bid to Bar NEC Chips Could Cost Users Strong Product*, PC Week, Mar. 5, 1985, at 8; Wirbel, *Intel Countersuit Hits NEC over Copyright on MPUs*, Electronic News, Mar. 4, 1985, at 1. Pursuant to partial findings of fact and conclusions of law issued Sept. 22, 1986, the *Intel* court concluded that Intel had valid copyrights in its microcode. NEC Corp. v. Intel Corp., 645 F. Supp. 590 (1986). Whether NEC infringed those copyrights remains for decision at a later date.

If true, a manufacturer using the NEC chip could make a fully IBM compatible computer (provided he can create an operating system that does not violate IBM's operating system copyrights) with additional features thrown in. Yet, if NEC's chip is found to infringe Intel's copyright in the chip microcode, Intel could bar NEC completely from marketing the latter's chip.[29] Understanding the issues involved in how and whether to protect microcode again requires a short detour into the technology.

A. *What is Microcode?*[30]

The above discussion of operating software divided computer technology into two parts, namely, hardware and software, and then further divided software into application programs and operating system programs.[31] It was briefly pointed out that operating systems software exists at several levels as far as engineers are concerned, but that present legal analysis permits all levels generally to be treated together as the intermediary between the application programmer and the hardware. An understanding of microcode requires that the discussion of hardware similarly be broken down into two levels.

What we have previously treated as hardware presents the programmer with an elementary set of instructions, such as transferring a binary datum or instruction from one place to another or adding two binary numbers together. That elementary instruction set, and the speed with which its instructions are executed, define the basic characteristics of the machine. A change in the elementary instruction set results in essentially a different computer.

In the early days of computers, the elementary instruction set was "hardwired;" each instruction was directly "executed" by physical electronic circuitry, in that a given binary signal at the inputs of the circuitry immediately gave the binary output corresponding to that instruction. When computers are constructed in this way, it is rather difficult to change the basic instruction set, which is why the elements of the technology at that level and below have come to be referred to as "hardware," while those above are called "software."

Hardwiring each instruction of a complicated instruction set, however, can

29. Copyright law has no general provision for compulsory licensing. Historically, copyright law has been intimately tied up with artistic expression, and it has always been felt that an artist should have complete control over distribution of his or her work. The inappropriateness of this attitude in application to programs, in which esthetics are wholly irrelevant except for the nontraditional notion (under copyright) of machine efficiency, was an important basis for the insistence of the Ministry of International Trade and Industry (MITI) on a separate Program Rights Law for programs. The proposed MITI statute, which was rejected in favor of following the world-wide trend of protecting programs under copyright law, would have included a compulsory licensing provision for programs under certain circumstances. *See* Karjala, *supra* note 24, at 68–69, 77–79.

30. For an introduction to the concepts involved in microcode and microprogramming, see Patterson, *Microprogramming*, Sci. Am., March 1983, at 50.

31. *See supra* text accompanying notes 13–15.

become expensive and unwieldy. A computer functions sequentially under the ticking of an internal clock, with control signals being sent out during each clock cycle to open and close the switches necessary to bring into operation the digital circuitry desired in that cycle. This control function becomes very complex when the instruction set is large. Moreover, different instructions often make use of similar digital logic at various points, and it is wasteful to build the same circuit many times over if a single circuit could be used in each instruction that calls for the logical operations it represents. Microprogramming, or microcode, permits the use of simpler hardwired logic circuitry in sequential combination to permit the execution of more complicated instructions.[32]

As an example, oversimplified for clarity, consider an instruction that adds two numbers A and B and calls the result C. To execute this instruction, the computer must sequentially bring A out of memory and put it at one input of an adding circuit, bring B out of memory and put it at the other input of the same adding circuit, execute the addition operation, and finally place the result in the desired memory location.[33] While it is possible to build a single circuit that will accomplish this in one operation, three simpler operations can do it more tidily, namely, a "Load" operation that brings data out of memory, an "Add" operation that effects the addition of two numbers placed at the input terminals of an adding circuit, and a "Store" operation that places data in memory. Then the instruction "Add A, B, C" (meaning "Fetch the numbers stored at memory locations A and B, add them together, and store the result at memory location C") can be carried out in four steps: Load A, Load B, Add, Store C.

In this example, only the Load, Add, and Store functions need be hardwired. When the microprocessing chip receives the instruction "Add A, B, C," a "microprogram" stored inside the chip then calls upon the Load, Add, and Store functions sequentially to carry it out. From the point of view of the outside programmer, the chip has the complicated instruction "Add A, B, C," as part of its instruction set, and the programmer is indifferent to the fact that the "hardware" consists of simpler circuitry used sequentially.[34] In the same way, many of

32. Although the discussion below treats microcode as being encapsulated in a microprocessing chip, there is no necessary connection between "microcode" and microprocessors or microcomputers. Microcode can be written to control the functions of any computer, regardless of its size or power. In practice, however, it seems to be most commonly used in microprocessing chips, or microprocessors, which constitute the central processing units of so-called microcomputers, and the terms "microcode" and "microprogram" have become rather firmly established jargon. The "micro" prefix in these latter two terms actually refers to instructions carried inside the computer that determine the most basic instruction set available to programmers and not normally accessible by them.

33. The computer must be "informed" of the memory addresses of A and B as well as the address in which C is to be stored when calculated. The machine language instruction supplies this information in the form of digital electronic signals that connect the circuit paths appropriately between the computer's memory and its logic circuitry.

34. The programmer is indifferent in the sense that precisely the same instruction is available to him through microprogramming that could have been made available by hardwiring, which means his

the other instructions available to the outside programmer will be executed by hardware circuitry called to duty sequentially in accordance with microcode stored inside the chip. Thus, the microcode determines the instruction set offered by the chip.

B. Policy Problems in Protecting Microcode

Although microprograms are usually rather simple, consisting often of only a few steps, there is no difference in principle between a microprogram and any other kind of program. Even in practice, the only difference is that the microprogram is usually stored inside the microprocessing chip itself and is usually inaccessible to outside programmers. Nevertheless, protecting microcode under copyright law raises some serious policy questions.

Consider a semiconductor manufacturer that develops a microprocessor used in a very popular computer. The microprocessor contains microcode that determines the most basic instruction set available to programmers.[35] Once a large amount of software is available for the computer, any computer manufacturer offering a competing computer will have severe marketing difficulties unless his computer is compatible with the existing software.

A first and vital step in achieving such compatibility is that his computer offer precisely the same basic instruction set and that each instruction in the set be carried out in exactly the same number of computer clock cycles.[36] One way to do this, of course, is to use the same microprocessing chip if it is available. A second way is to use a different chip that "looks" identical to the outside programmer, in that it presents the same instruction set and executes each instruction in the same amount of time. If, however, a copyright in the original chip's

program will run regardless of which way the computer is actually constructed. Because microprogramming calls on simple functions sequentially, however, the execution of a complicated instruction with microprogramming must step through a number of cycles of the internal clock and can therefore take longer. Consequently, if all other things were equal, a programmer might prefer a hardwired machine because his program would be executed faster. Of course, all other things are not equal, which is why so much microprogramming is being used today.

35. In this case, the programmers will be those who create the computer's operating system, which in turn defines the higher level instruction set in which most application programs will be written.

36. As in the case of operating system compatibility, see *supra* note 23, this somewhat oversimplifies the compatibility problem. Obviously, a chip can be compatible if it includes the first chip's instruction set as a subset of its own instruction set, and probably if it executes all instructions in that set faster than the first chip, rather than at exactly the same speed. Moreover, there are undoubtedly many specific cases in which clever computer engineers can achieve a high degree of compatibility, perhaps 100 percent compatibility, even when some instructions are executed more slowly than those of the first chip. The key point for purposes of this article is that there may be technological constraints on achieving compatibility that require some degree of copying of the microcode. Execution time seems a likely candidate for such a constraint and is one that can be understood generally in terms of the brief introduction to the technology provided in this article. In a particular case, the trier of fact must consider the evidence from both sides concerning the nature of such constraints, if any.

microcode effectively renders it impossible for a competing manufacturer to make a different but compatible chip, compatibility could only be achieved by using the original chip. This in turn would give the original chip manufacturer a complete monopoly over a vital piece of that particular machine, and not just for the 10 years given under the recently enacted Semiconductor Chip Protection Act[37] or the seventeen years available if the chip were patented, but for the much longer period provided by copyright law.

Recognizing a copyright in the microcode could, in fact, result in such a monopoly because it may be impossible to achieve identical performance without at least some copying. The reason is that microcode is very basic. There may be many ways to write microcode so as to achieve the same instruction set, but there may be at most only a few ways to write it so that each of, say, 100 to 200 instructions in the chip's instruction set are executed in a way that presents exactly the same appearance to the outside programmer. If the performance of the chip, as seen by the outside programmer, is not identical, 100 percent compatibility cannot be achieved.

C. Protection of Microcode Under Japanese Copyright Law

As discussed above,[38] the use of microcode achieves at the most basic level of a computer what the operating system achieves at higher levels: it determines the instruction set available to the next level of programmer (in this case, the creator of the operating system). It is clear that the instruction set itself is not protected under Japanese copyright law, pursuant to the "program language" exception of article 10(3)(i).[39]

Nevertheless, the microcode defining the instruction set is different from the instruction set itself, just as the operating system programs are different from the higher level instruction set that they create. Consequently, we must ask whether microcode is a "program work" under Japanese law and, if so, whether protection in such a work is limited by any of the exceptions of article 10(3).

1. Is microcode a "program work"?

In deciding whether microcode is a program work under Japanese copyright law, the first question is whether it constitutes a "program" within the meaning of the definition in article 2(1)(x-ii).[40] Under that definition, a program is a set of instructions causing a "computer" to achieve a certain result. Because the microcode essentially defines the computer, in the sense that if the microcode is changed, the computer itself is a different machine, microcode might be viewed as not causing a computer to do anything but rather as causing a hardwired set of

37. 17 U.S.C. §§ 901–914 (Supp. III 1985).
38. *See supra* text accompanying notes 31–34.
39. *See supra* text accompanying note 12.
40. *See supra* text accompanying note 9.

logic circuits to become a computer. This interpretation would remove microcode from the scope of copyright protection altogether.

Even if microcode is considered a "program" under the Japanese definition, whether it constitutes a "program work" still hinges on article 2(1)'s definition of copyrightable work,[41] which requires that the microcode contain creatively expressed thought or feeling. This creativity requirement is generally considered to be more or less equivalent to the "originality" requirement of United States (U.S.) law, which requires no more than that the work owe its origin to the author.[42] The case law in Japan, however, is sparse and something more may be required as a matter of policy when a potential monopoly over technology is at stake.

As discussed above, most microprograms consist of relatively few steps. It may be that, given the hardwired circuits available, achieving a particular instruction in the instruction set is either obvious or achievable in only one way or both. In either case, any competent microprogrammer asked to produce the desired instruction would come up with the same microcode to effect it. Under U.S. law, idea and expression are said to merge when the idea is so simple that it can be expressed in only a limited number of ways.[43] Japanese law applies the same concept to deny creativity when idea and expression merge.[44] Consequently, it seems both sound policy and consistent with copyright principles to conclude that microprograms are not "program works" under Japanese law. This interpretation, too, would remove microcode from the scope of copyright protection altogether.

2. *Is microcode a "program language" or "rule"?*

If a microcode should survive the analysis outlined above and be held to fall within the "program" and "program work" definitions, it still remains to decide whether protection is limited by article 10(3).[45] The analysis of microcode under the "program language" and "rule" limitations on copyright protection in program works under Japanese law is quite similar to that given above for operating systems. Microcode determines the instruction set available to the outside programmer and consequently creates a program language.[46] Although the microcode itself is not a program language, to the extent that copying the microcode is necessary in making a compatible chip, a copyright in the microcode results in a monopoly on the program language it creates. Because copyright protection in a

41. *See supra* text accompanying notes 5–8.

42. *E.g.*, Alfred Bell & Co. v. Catalda Fine Arts, Inc., 191 F.2d 99 (2d Cir. 1951).

43. Morrissey v. Procter & Gamble Co., 379 F.2d 675 (1st Cir. 1967).

44. *E.g.*, N. Nakayama, The Legal Protection of Software 39 (1986) (in Japanese). Professor Nakayama expressly applies the merger of idea and expression concept to question the copyrightability of microcode.

45. *See supra* text accompanying note 12.

46. *See supra* text accompanying notes 32–34.

program work does not extend to such languages, it should not cover the microcode itself to the extent that copying is necessary to create the language.

It also may be possible to deny copyright protection to microcode as a "rule" under article 10(3)(ii), at least to the extent it must be copied to achieve a compatible interface. Just as the operating system creates the interface between the hardware and the application programmer, microcode similarly creates the interface between the hardwired circuitry and the operating system programmer. Moreover, to the extent the microcode uniquely defines that interface, the argument is stronger that it is excepted from protection as a "rule" than in the case of operating systems, because the microcode is the link between the circuitry and a *specific* program, namely, the operating system.

3. *Summary*

The statutory language provides four solid possibilities for denying or limiting copyright protection of microcode under Japanese copyright law: microcode may not be a "program" because it is only a part of a computer, or it may not be a "program work" because it lacks creativity. Either of these two interpretations would result in a complete denial of copyright protection. Alternatively, microcode might be protected in general, but protection may be limited to the extent necessary to achieve compatibility with a competing chip, pursuant to either the "program language" or "rule" limitations of article 10(3). Avoiding a technological monopoly created by copyright law provides a policy basis for adopting some or all of these approaches, especially the latter two, which would permit copying only to the extent that a necessity for compatibility can be shown.

A Japanese court will surely be aware of these policy arguments by the time the question arises in a dispute. It therefore seems likely that protection of microcode under Japanese copyright law will either be denied outright or severely limited. Chip manufacturers would then have to rely on the various chip protection statutes recently adopted in both America and Japan, which seems appropriate in that those statutes were adopted for the specific purpose of protecting chip technology.[47]

V. The Scope of Copyright Protection in an Application Program

Both Japanese and American copyright law protect the expression of a work and not the ideas themselves.[48] Suppose a programmer has a copy of the source code of a popular application program and wants to create a competing program

47. Act Concerning the Circuit Layout of Semiconductor Integrated Circuits, May 31, 1985 (Japan); Semiconductor Chip Protection Act of 1984, 17 U.S.C. §§ 901–914 (Supp. III 1985).

48. *E.g.*, Mazer v. Stein, 347 U.S. 201, 217–18 (1954); judgment of Feb. 10, 1984, Japan (District Court, Tokyo), 1111 Hanji 134 (identical game rules expressed in different language not a copyright violation).

that accomplishes the same result. To what extent can he borrow from the original program or, in other words, what constitutes the idea, as opposed to the expression, in a program?

Two recent American cases exemplify the problem concretely. Both *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*[49] and *SAS Institute, Inc. v. S & H Computer Systems, Inc.*[50] involved the defendants' rewriting a complicated application program in a different source code language so that it would run on a different make of computer. The courts in both cases found this activity to be a copyright violation and in so doing raised a number of difficult issues that are necessarily beyond the scope of this short essay.[51] Nevertheless, a brief discussion of these cases may shed some light on how cases of this type should be decided if they were to arise under Japanese law.

In *SAS*, the court found that the defendant "extensively and systematically" used plaintiff's source code in preparing its competing product and also found a number of instances of pure slavish copying, in that there were places in which the name "SAS" had either been replaced with "S & H" or removed entirely, with the result that statements became ungrammatical or incoherent.[52] Nevertheless, the plaintiff's expert was apparently able to come up with only 44 code lines that he could say for certain were copied, out of a total of some 186,000 lines. In response to the defendant's argument that so few similarities could not be considered "substantial," the court responded that "to the extent that it represents copying of the organization and structural details of SAS, such copying pervades the entire S & H product."[53] In finding a copyright violation, the opinion seems to be saying that the organization and structure of a program are protected expression and not unprotected ideas.

The holding of *Whelan* on this point is even clearer. The *Whelan* district court expressly states that converting from one program language to another is not comparable to translating a book from English to French; in fact, a literal translation was virtually impossible. Nevertheless, in finding a violation, the court said that in order to convert a program from one source language to another, one must "study the manner in which the information flows sequentially from one function to another. Once this is fully understood, one may copy this exact manner of operation for use in a computer that responds to commands written in a different source code language." Thus, mimicking the *manner of operation* of a computer results in a copyright violation.

Protecting the overall structure and organization of an application program can

49. 609 F. Supp. 1307 (E.D. Pa. 1985), *aff'd*, 797 F.2d 1222 (3d Cir. 1986).

50. 605 F. Supp. 816 (M.D. Tenn. 1985).

51. For a detailed analysis of these cases, see Karjala, *Chosakukenhoo no Moto ni okeru Konpyuuta Puroguramu no Hogo Hani (The Scope of Protection in a Computer Program under Copyright Law)*, 4 Hoo to Konpyuuta 92 (Law and Computers)(1986)(in Japanese).

52. *Id.* at 822–23.

53. *Id.* at 830.

result in long-term copyright protection of technological innovation. The overall structure and information flow of a particular program are determined by flow charts created in the program design stage. As software technology develops, some of these general structures are likely to emerge as more efficient than others. In a word processing program, for example, it may become clear that a particular approach to handling footnotes is better in some technological sense than the others that have been tried. If that approach is considered part of the protected expression in a program, its original discoverer will enjoy a very lengthy monopoly in this optimal aspect of the technology. Copyright would then have ventured deeply into an area traditionally reserved for patent law.

For policy reasons, therefore, the overall structure and composition of a program should be considered "idea" rather than "expression." Whether a Japanese court would accept this argument on its own is open to question, especially in view of the United States precedent going the other way. It seems clear, however, that *Whelan*, and quite possibly *SAS*, would come out differently under Japanese law, because the "algorithm" limitation under article 10(3)(iii) on protection in a program work leads to this interpretation without reliance on policy. An algorithm, or "method of solution," is "a method for combining instructions in a program," and it is difficult to think of the overall structure, derived from the flow chart,[54] as anything else. That structure tells the programmer how to combine the various instructions in the program without specifying the specific instructions themselves.

Consequently, the algorithm limitation would seem to deny protection under Japanese copyright law to the overall structure of a program. This would call for a different result on the facts of *Whelan*, and, at the least, would require a court in a case like *SAS* to inquire more carefully into precisely what was copied and whether the facts justify a conclusion that expression and not just algorithmic ideas were taken.

VI. Reverse Engineering of Programs

If one accepts the above conclusion that program structure is not protected by Japanese copyright law, it follows that such structure constitutes an idea in the public domain. Such ideas, of course, may be used freely by others. The question then arises of how one can extract that idea from a program that is distributed in a form that cannot be read or studied by human beings. Does the act of converting such a program into a form that can be read and studied by human beings constitute an illegal act of copying?

The easiest, but probably fairly uncommon, example arises when source code is distributed on a magnetic tape or disk. But even when the program is dis-

54. The flow chart itself, of course, is independently a copyrightable work other than a "program work." The exceptions under the Act, *supra* note 1, art. 10(3) limit only the protection afforded to program works, so they do not affect any copyright in the flow chart.

tributed in object code form, others may desire to reproduce it for the purpose of working backwards to develop the source code or at least an approximation of the source code. Is it illegal, for the purpose of studying the program, to reproduce the program itself, on the screen of a monitor or to print it out on paper? The court in *SAS* held that it was.[55]

At first glance, this result seems obvious. After all, copyright law is aimed specifically at copying. This shows one of the conceptual difficulties with using copyright law to protect programs. If one makes a copy of a book, two people can read the book at the same time while the author receives royalties for only one. But programs are not designed for human consumption; rather, they constitute the technology for *using* computers. A simple printout of a program cannot be used in a computer, because computers do not accept input in that form. In making such a printout without more (such as reinputting the program into a second computer), there is no danger that the program will be used in two computers while its creator receives the royalty on only a single sale. Moreover, if such activity is prohibited by copyright law, there is no way for other programmers to extract the programming ideas in the program, notwithstanding that they are theoretically in the public domain. The result is that copyright protection extends to the ideas in a program as well as to their expression.

Is it possible to interpret Japanese law to obtain the result that copying for the purpose of extracting the freely usable ideas in a program is not a copyright violation?[56] Given the express denial of protection to algorithms, one might expect a Japanese court to try to reach a result permitting the free use of algorithms in practice as well as in principle, but it will not be easy under the current language of the Copyright Act. Article 47-2(1) now permits reproducing or adapting programs to the extent deemed necessary to use them in a computer.[57] There is an obvious negative implication that reproducing for any other purpose is infringement and reproducing for the purpose of studying the program structure is clearly not necessary to use the program in a computer. Consequently,

55. 605 F. Supp. at 828–29.

56. In the United States, such activity may well be a "fair use" of the program, notwithstanding the decision to the contrary in *SAS*. In E.F. Johnson Co. v. Uniden Corp. of America, 623 F. Supp. 1485 (D. Minn. 1985), for example, on facts similar to *SAS*, both parties admitted that "dumping" and analyzing code is standard industry practice. Under the Japanese Copyright Act, *supra* note 1, however, the "fair use" provisions are expressly provided for in the statute (arts. 30–50) and are narrow in their coverage. Moreover, recently adopted art. 47-2(1), discussed in the text below, is an explicit fair use provision applying to program works. In the face of so many express statutory limitations on the scope of copyright, it is unusual for Japanese courts to create further limitations on their own.

57. Act, *supra* note 1, art. 47-2(1) provides:

> The owner of a copy of a program work may copy or adapt such copy (including copies of derivative works made pursuant hereto) to the extent deemed necessary for him to use the work in a computer. Provided, however, that this rule shall not apply when the provisions of Article 113(2) [providing that the knowing use of illegally made copies is, under certain circumstances, a copyright violation] apply to the use of the copy in operating a computer.

such activity will be allowable only if, somehow, it can be deemed not to be "reproduction."

Article 2(1)(xv) of the Copyright Act defines "reproduction" as "the reproduction in a tangible form by means of printing, photography, polygraphy, sound or visual recording or otherwise."[58] This definition was adopted before the recent amendments to include programs as copyrightable works and on its face certainly seems to cover the printout of a program. However, the reason for having programs in the first place is not to read them but to use them in a computer. If there were a way to guarantee that a program copy could never be used in a computer, program makers would not legitimately care how many copies were made, because people would still have to buy the program in order to use it. And the printout of a program *cannot* be used in a computer. Of course, program makers have never wanted their programs copied even for the purpose of extracting the ideas contained in them, but that only evinces a desire for more protection than copyright law provides. Copyright does not protect ideas and never has (except perhaps until now). Consequently, it would make sense to interpret article 2(1)(xv), in the case of program works, to mean reproduction in a form usable in a computer, excluding a reproduction comprehensible only to humans. Whether a Japanese court will be this creative, however, is another question entirely.

VII. Conclusion

The recent amendments to the Japanese Copyright Act constitute a detailed statutory approach to problems of computer program protection. By setting specific limitations on the scope of copyright protection in program works, the Act appears to define the line between protected "expression" and unprotected "ideas" in computer programs. These limitations on copyright protection are potentially of great significance to a number of important problems arising out of the decision to protect programs under copyright law.

First, the analysis developed in this article concludes that, while computer operating systems should be treated generally as program works under the Japanese Copyright Act, the statutory limitation that copyright protection in program works does not extend to program languages implies that operating system programs can be copied to the extent necessary to insure compatibility with application programs written for that operating system. To do otherwise would extend copyright protection to the program language defined by the operating system, contravening the express statutory limitation. This conclusion insures that the first creator of a popular operating system, or the commonly used operating

58. The requirement that the reproduction be in a tangible form might permit reproduction of the program on the screen of a monitor. This may be enough to permit reverse engineering of programs, although it would seem to be much more difficult to study a program in that form than from a printout. In any event, technicalities of this type should not be determinative of the fundamental question of whether programs can be copied for the purpose of study.

system for a popular computer, does not achieve a *de facto* monopoly on third-party software written for that system.

Second, there are sufficient grounds under the analysis presented above for concluding that microcode is not protected at all under the Act. Moreover, even if it is protected as a program work, it can probably be copied to the extent necessary to design and build digital logic circuitry presenting an identical "hardware" appearance to the outside programmer. Not to permit such copying would have the effect of giving copyright protection to the program language, that is, the chip's instruction set, defined and created by the chip's microcode. Such protection for program languages is explicitly denied by the Act. It is also important as a policy matter to permit such copying to insure that the first manufacturer of the microprocessing chip for a popular computer does not achieve a long-term *de facto* technological monopoly over the supply of chips for that machine.

Third, the protection of program works does not extend to algorithms used in their production. This implies that, under Japanese law, the organization and structure of a program are in the public domain. This conclusion, too, is supported by policy considerations. Programs are the technology that allows users to operate computers, and program organization and structure can contain innovative technological ideas leading to higher efficiencies and other improvements. Unless protected by patent law, such technological ideas should be freely usable, so that program technology can develop as freely as any other area of technology. Unfortunately, these policy notions are not fully implemented under the Japanese Copyright Act, because the Act does not appear to authorize the copying of programs for the limited purpose of studying them to extract the organizational and structural ideas expressed therein.