

THIS OPINION WAS NOT WRITTEN FOR PUBLICATION

The opinion in support of the decision being entered today (1) was not written for publication in a law journal and (2) is not binding precedent of the Board.

Paper No. 25

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte YAYOI ABE, SHINICHIRO SUZUKI,
and YOICHIRO TAKEUCHI

Appeal No. 1997-2212
Application No. 08/233,387

HEARD: March 6, 2000

Before THOMAS, RUGGIERO, and BARRY, Administrative Patent
Judges.

BARRY, Administrative Patent Judge.

DECISION ON APPEAL

This is a decision on the appeal under 35 U.S.C. § 134
from the final rejection of claims 1-9 and 11-19. We reverse.

BACKGROUND

During production of a microprocessing architecture, a new microprocessor and software for the microprocessor are sometimes developed concurrently. In developing the software, an executable program for the microprocessor must be debugged. When the time comes to debug the program, if the microprocessor is not yet complete, the program cannot be debugged on the microprocessor. Instead, a simulator is used to debug the program.

A simulator for an existing architecture cannot be used, as is, to debug the program. A conventional simulator must be modified or a new simulator must be developed. Neither task is simple. When several new architectures are developed, and selecting one architecture to be used is necessary, moreover, simulators corresponding to each of the new architectures must be prepared. Furthermore, execution of simulation is time-consuming.

The invention at issue in this appeal enables a executable machine code developed for a new microprocessing

architecture to be debugged on an existing microprocessor. The machine code is first converted to high-level source code that is architecture-independent. The source code is then compiled and linked to produce an executable load module for the existing microprocessor. When the load module is executed by the existing microprocessor, it performs the same operations that the new microprocessor will perform, thereby debugging the micro-processor-dependent executable code on the existing microprocessor.

Claim 1, which is representative for our purposes, follows:

1. A converting method for converting an architecture of a program, comprising:

a first step of compiling a first high-level language source program for a computer of a first architecture, thereby producing a machine program for a computer of a second architecture;

a second step of decompiling the machine program, thereby producing a second high-level language source program which does not depend on any architecture; and

a third step of compiling and linking the second high-level language source program, thereby producing a first executable load module.

The reference relied on in rejecting the claims follows:

Robinson et al. 5,307,504 Apr. 26, 1994
(Robinson) (filing Mar. 7,
1991).

Claims 1-9 and 11-19 stand rejected under 35 U.S.C. § 103 as obvious over Robinson. Rather than repeat the arguments of the appellants or examiner in toto, we refer the reader to the briefs and answer for the respective details thereof.

OPINION

In reaching our decision in this appeal, we considered the subject matter on appeal and the rejection advanced by the examiner. Furthermore, we duly considered the arguments and evidence of the appellants and examiner. After considering the totality of the record, we are persuaded that the examiner erred in rejecting claims 1-9 and 11-19. Accordingly, we reverse.

We begin by noting the following principles from In re Rijckaert, 9 F.3d 1531, 1532, 28 USPQ2d 1955, 1956 (Fed. Cir. 1993).

In rejecting claims under 35 U.S.C. Section 103, the examiner bears the initial burden of presenting a

prima facie case of obviousness. In re Oetiker, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992). Only if that burden is met, does the burden of coming forward with evidence or argument shift to the applicant. Id. "A prima facie case of obviousness is established when the teachings from the prior art itself would appear to have suggested the claimed subject matter to a person of ordinary skill in the art." In re Bell, 991 F.2d 781, 782, 26 USPQ2d 1529, 1531 (Fed. Cir. 1993) (quoting In re Rinehart, 531 F.2d 1048, 1051, 189 USPQ 143, 147 (CCPA 1976)). If the examiner fails to establish a prima facie case, the rejection is improper and will be overturned. In re Fine, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988).

With these in mind, we analyze the examiner's rejection.

The examiner's rejection is based on the following premise.

As shown in Fig. 1, a program 10 is written in X instruction set employed in producing an executable form of the program 10. The X instruction is compiled and linked by a computer system, for example VAX, according to its instruction code. The first instruction set is compiled and linked (block 16, 18). The machine code is translated to other code through code translator (block 32) or through direct translation path (26) as disclosed in Column 3, lines 37-46, Column 6, lines 1-16. The translated code is recompiled and properly executed to guarantee preservation of X instructions or platform. These instruction are executed by computer 20 (Fig. 1).

Robinson also disclose [sic] prior art teaching of high level [sic] programs migration such as

FORTRAN into machine codes, and its structure conserve during compiling and decompiling (see Col. 3, lines 37-42). (Examiner's Answer at 3.)

The appellants' argue, "Robinson does not teach decompiling at all, rather Robinson teaches only translating/assembling."

(Reply Br. at 4.)

"[W]hen interpreting a claim, words of the claim are generally given their ordinary and accustomed meaning, unless it appears from the specification or the file history that they were used differently by the inventor." In re Paulsen, 30 F.3d 1475, 1480, 31 USPQ2d 1671, 1674 (Fed. Cir. 1994) (citing Carroll Touch, Inc. v. Electro Mechanical Sys., Inc., 15 F.3d 1573, 1577, 27 USPQ2d 1836, 1839 (Fed. Cir. 1993)). Here, claims 1-9 and 11-19 each specifies in pertinent part the following limitations: "decompiling the machine program, thereby producing a second high-level language source program which does not depend on any architecture" Because neither the specification nor the file history defines the term "decompiling" nor suggests that the appellants sought to

assign a meaning to the term different from its ordinary and accustomed meaning, that is the meaning we must give it.

Those skilled in the art would have understood that a decompiler is "[a] program that takes ... machine code and attempts to generate high-level source code from it"

Microsoft Press Computer Dictionary 114 (2d ed. 1994) (copy attached). In view of this understanding, the limitations recite translating executable machine code into higher-level source code that is architecture-independent.

The examiner fails to show a teaching or suggestion of the claimed limitations. "Obviousness may not be established using hindsight or in view of the teachings or suggestions of the inventor." Para-Ordnance Mfg. v. SGS Importers Int'l, 73 F.3d 1085, 1087, 37 USPQ2d 1237, 1239 (Fed. Cir. 1995), cert. denied, 519 U.S. 822 (1996) (citing W.L. Gore & Assocs., Inc. v. Garlock, Inc., 721 F.2d 1540, 1551, 1553, 220 USPQ 303, 311, 312-13 (Fed. Cir. 1983), cert. denied, 469 U.S. 851 (1984)). The mere fact that prior art may be modified as proposed by an examiner does not make the modification obvious unless the prior art suggested the desirability thereof. In

re Fritch, 972 F.2d 1260, 1266, 23 USPQ2d 1780, 1784 (Fed. Cir. 1992); In re Gordon, 733 F.2d 900, 902, 221 USPQ 1125, 1127 (Fed. Cir. 1984).

Here, as background to his invention, Robinson mentions "migrating programs written in a high level language such as FORTRAN" Col. 3, ll. 39-41. Such migration employs "[r]ecompiling or recoding" Id. at l. 38. The examiner has not shown that either of these operations teaches or would have suggested decompiling executable machine code into higher-level source code that is architecture-independent. To the contrary, recompiling comprises "compil[ing] a program again, usually because of changes that needed to be made in the source code" Microsoft Press Computer Dictionary at 333 (copy attached). Because coding comprises "generating source code in the language(s) of the programmer's choice," id. at 78, recoding is generating source code in at least one language of the programmer's choice again.

In describing his invention, Robinson teaches translating higher-level source code into executable machine code that is

architecture-dependent. "As shown in FIG. 1, an application program 10, written in source code," col. 5, ll. 43-44, is the higher-level source code. "As [also] shown in FIG. 1, the application program 10 can be migrated to the Y executable code 22 in either an indirect path 24 or a direct path 26." Col. 6, ll. 1-3. The examiner has not shown that either of these techniques teaches or would have suggested decompiling executable machine code into higher-level source code that is architecture-independent.

Direct migration comprises compiling and linking the higher-level source code "with the use of a Y compiler 28 and a Y linker 30." Id. at ll. 4-5. The compiling and linking produce a "resultant Y executable code ... designated by the reference numeral 22B." Id. at ll. 5-6. The Y executable machine code is architecture-dependent. Specifically, the executable machine code "employ[s] a Y instruction set to which the hardware architecture of the Y computer system 20 is adapted." Col. 5, ll. 53-55. "[A]s specifically indicated for illustrative purposes in FIG. 1, ... the Y system can employ a reduced instruction set architecture called the RISC

architecture within the Digital Equipment Corporation." Id.
at ll. 61-66. The RISC architecture is "embodied in equipment
made by Digital Equipment Corporation" Id. at ll. 66-68.

Indirect migration involves compiling and linking the
higher-level source code "by means of an X compiler 16 and an
X linker 18." Col. 6, ll. 16-17. The compiling and linking
produce "X executable code 14 which can run on the X computer
system 12." Id. at ll. 18-19. The X executable machine code
is architecture-dependent. Specifically, the X executable
machine code "can be a RISC instruction set. For example, as
specifically indicated for illustrative purposes in FIG. 1,
the X
system can employ the VAX® architecture" Col. 5, ll. 59-
63. The RISC architectures is "embodied in equipment made by
Digital Equipment Corporation" Id. at ll. 66-68.
Following, compiling and linking, the X executable machine
code is translated "into the corresponding Y executable
application code

designated by the reference numeral 22A." Col. 6, ll. 21-23.
As explained regarding direct migration, the Y executable
machine code is architecture-dependent.

For the foregoing reasons, we are not persuaded that
teachings from the prior art would appear to have suggested
the claimed limitation of decompiling, i.e., translating
executable machine code into higher-level source code that is
architecture-independent. The examiner impermissibly relies
on the appellants' teachings or suggestions; he has not
established a prima facie case of obviousness. Therefore, we
reverse the rejection of claims 1-9 and 11-19 under 35 U.S.C.
§ 103.

CONCLUSION

To summarize, the rejection of claims 1-9 and 11-19 under
35 U.S.C. § 103 is reversed.

REVERSED

JAMES D. THOMAS)	
Administrative Patent Judge)	
)	
)	
)	
)	BOARD OF PATENT
JOSEPH F. RUGGIERO)	APPEALS
Administrative Patent Judge)	AND
)	INTERFERENCES
)	
)	
)	
LANCE LEONARD BARRY)	
Administrative Patent Judge)	

Appeal No. 1997-2212
Application No. 08/233,387

Page 13

OBLON, SPIVAK, MCCLELLAND,
MAIER & NEUSTADT
1755 Jefferson Davis Highway
4th Floor
Arlington, VA 22202

Appeal No. 1997-2212
Application No. 08/233,387

Page 14