Gottschalk v. Benson, 409 U.S. 63 (1972)を読む

2022 年 7 月 25 日
弁護士　渡邉　明彦

はじめに

前回の「ディーア判決を読む」の回でも紹介したが、米国で、自然界の法則を利用する発明に、特許適格性があるかという問題を巡っては、その判断の指針を示している Alice/Mayo ルールが存在している。執筆者自身の当面の関心は、***Mayo v. Prometheus***, 566 U.S. 66 (2012)を理解することにあるが、これらの判決の中で引用される先例に、ベンソン判決 ***Gottschalk v. Benson***, 409 U.S. 63 (1972)、フルック判決 ***Parker v. Flook***, 437 U.S. 584 (1978)、そして前回検討したディーア判決 ***Diamond v. Diehr***, 450 U.S. 175 (1981)がある。これら 3 つの判決は、trilogy of cases（判例の三部作）になるが、なによりも先ず、内容を理解していきたい。

Syllabus

GOTTSCHALK, ACTING COMMISSIONER OF PATENTS *v.* BENSON ET AL.

CERTIORARI TO THE UNITED STATES COURT OF CUSTOMS AND PATENT APPEALS

No. 71–485. Argued October 16, 1972—Decided November 20, 1972

Respondents' method for converting numerical information from binary-coded decimal numbers into pure binary numbers, for use in programming conventional general-purpose digital computers is merely a series of mathematical calculations or mental steps and does not constitute a patentable "process" within the meaning of the Patent Act, 35 U. S. C. § 100 (b). Pp. 64–73.

— C. C. P. A. (Pat.) —, 441 F. 2d 682, reversed.

DOUGLAS, J., delivered the opinion of the Court, in which all Members joined except STEWART, BLACKMUN, and POWELL, JJ., who took no part in the consideration or decision of the case.

Richard B. Stone argued the cause for petitioner. With him on the briefs were *Solicitor General Griswold, Assistant Attorney General Kauper, Acting Assistant Attorney General Comegys, Howard E. Shapiro, Richard H.*

1. 出願された特許のクレーム(Benson-Talbot クレーム)

In 1963, Bell Labs filed an application on behalf of its employees, Benson and Talbot, entitled "Conversion of Numerical Information." It was for a method for converting from binary-code decimal (BCD) numbers to pure binary numbers. The Patent Office Board of Patent Appeals and Interferences had affirmed the examiner's rejection of the application, and the company appealed. At the time there were only two claims remaining:

http://digital-law-online.info/lpdi1.0/treatise61.html

今回検討する特許出願は、ベル研究所の職員 Benson 氏と Talbot 氏の発明にかかる、ベル研究所の 1963 年の出願である。標題は「数値情報の変換」。拒絶査定、不服申立、不成立審決、そして審決取消訴訟に進んだ段階で、「2 つのクレームだけが残っていた。」

ここでは、この 2 つのクレームの内容を紹介する。原文は、末尾の付録に掲載している。

【請求項 8】

「2 進化 10 進数（binary coded decimal）形式の信号を純粋な 2 進形式に変換する方法（method）であって、次のステップから構成されるもの：

(1)　2 進化 10 進数の信号を再入可能（reentrant）なシフトレジスタにストアする、

(2)　該レジスタの第 2 番目の位に 2 進数「1」が来るまで、該信号を 3 桁以上右にシフトさせる、

(3)　該レジスタの第 2 番目の位にある該 2 進数「1」をマスキングする、

(4)　該レジスタの第 1 番目の位に 2 進数「1」を加える、

(5)　該信号を左側に 2 ビットだけシフトさせる、

(6)　該第 1 番目の位に「1」を加える、

(7)　該レジスタの第 2 ビット目に次の 2 進数「1」が来るように準備するため、該信号を 3 桁以上右にシフトさせる。」

【請求項 13】

「2 進化 10 進数表現（binary coded decimal number representation）を純粋な 2 進数表現（binary number representations）に変換するデータ処理方法（method）であって、次のステップから構成されるもの

(1)　2 進数「0」または「1」について、最上位の 10 進数表現の各 2 進数「1」の位を、最下位 2 進数の位から、探す、

(2)　2 進数「0」が検知されたら、該最上位の 10 進数表現の次の最下位 2 進数の位について、ステップ(1)を繰り返す、

(3)　2 進数「1」が検知されたら、次に上位の 10 進数表現の第(i+1)および第(1+3)番目に下位の 2 進数の位に、2 進数「1」を加える、

(4)　該最上位の 10 進表現の 2 進数の位が尽きたら、前のステップ(1)から(3)を実行して変更されたとおりの、次の下位の 10 進数表現についてステップ(1)から(3)を繰り返す、そして、

(5)　第 2 に上位の 10 進数表現がこのように処理されるまで、ステップ(1)から(4)を繰り返す。」

2. 2 進、10 進、BCD（2 進化 10 進法）について

先ず「10 進数」であるが、「数を書き表す方法（記数法）の一つで、基数を十とした表記法のこと。人間が普段最も一般的に利用している位取り記数法で、通常、アラビア数字の「0」から「9」までのすべての数字を用いて数を表現する。10 進数では桁が一つ左へ移動する毎に値の重みが十倍に、右へ移動するごとに十分の一になる。すなわち、整数の右端の桁は一（100）の位、その左は十（101）の位、その左は百（102）の位、その左は千（103）の位、といった具合に各桁の重みが決まる。（IT 用語辞典 E-Words）

本件判決の原審の判決 **Application of Benson, 441 F.2d 682** で用いられている、十進「53」を例に、考えてみよう。（「53」は、以下で引用する **amicus curiae brief** に由来するようである。）

53_{10}	
$5 \times 10^1 + 3 \times 10^0$	
10^1	10^0
5	3

次に「2 進化 10 進法」であるが、

"In true binary the decimal number 53 is represented by 110101.
The problem is to convert the intermediate BCD representation into the true binary."

10 進の 0～9 と 2 進表現の対応は、ベンソン判決、66 ページにあり、以下のとおり。

Decimal		2^3 (8)		2^2 (4)		2^1 (2)		2^0 (1)		Pure Binary
0	=	0	+	0	+	0	+	0	=	0000
1	=	0	+	0	+	0	+	2^0	=	0001
2	=	0	+	0	+	2^1	+	0	=	0010
3	=	0	+	0	+	2^1	+	2^0	=	0011
4	=	0	+	2^2	+	0	+	0	=	0100
5	=	0	+	2^2	+	0	+	2^0	=	0101
6	=	0	+	2^2	+	2^1	+	0	=	0110
7	=	0	+	2^2	+	2^1	+	2^0	=	0111
8	=	2^3	+	0	+	0	+	0	=	1000
9	=	2^3	+	0	+	0	+	2^0	=	1001
10	=	2^3	+	0	+	2^1	+	0	=	1010

53	
5	3
0101	0011

BCD 表現は、0101 0011 となる。BCD の下 4 桁は、2 進数の 0～9 と同じである。BCD を 2 進数に換算するのは、BCD⇒10 進⇒2 進と変換していくことでも、もちろんできる。

10 進⇒2 進の換算は、下のような表を準備して、

2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1
	1	1	0	1	0	1

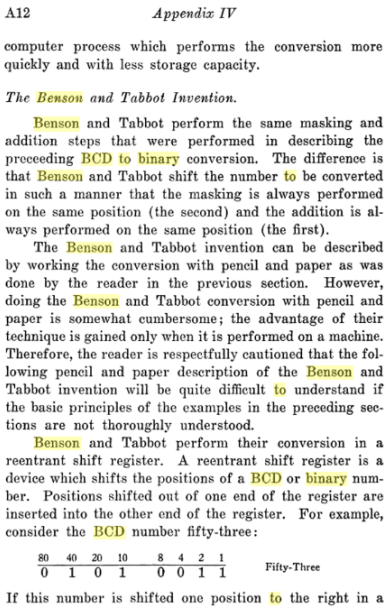
53 を超えない最大の 2 の累乗を、下の表から探す。この場合、25 である。54 から 32 を引き、21 を得る。21 を超えない最大の 2 の累乗を、下の表から探す。24 であり、21 から 16 を引き、5 を得る、を超えない最大の 2 の累乗を、下の表から探す。22 であり、5 から 4 を引き、1 を得る。1 を超えない最大の 2 の累乗を、下の表から探す。20 である。実際に「引いた」箇所に 1 を立て、他を 0 にすると、110101 という 2 進を得る。

なお、2 進、8 進、16 進、ビット、バイトの概要と相互の関係を知るには、

	村瀬 康治 (著)『はじめて読むマシン語―ほんとうのコンピュータと出逢うために』(アスキー出版、1983 年)(「村瀬」として引用) 2 進、8 進、16 進、コンピュータアーキテクチャー、マシン語について、初心者向けの解説がある。
	Thomas C. Bartee (著), 石田晴久監訳・姫野 俊一＝他(翻訳)『コンピュータアーキテクチャと論理設計』I、II (1996/8/1、丸善)(「Bartee」として引用) BCD についての詳しい解説がある。

4. 請求項 8 について

Gottschalk v. Benson, 409 U.S. 63 (1972)事件については、現在、Google Scholar で、「BCD to Binary Benson」で検索すると、本事件で提出された **amicus curiae brief** (第三者意見書)が閲覧できる。この資料に基づいて、請求項 8 が何をクレームしている（教示している）のか検討しよう。



A12 *Appendix IV*

computer process which performs the conversion more quickly and with less storage capacity.

The Benson and Tabbot Invention.

Benson and Tabbot perform the same masking and addition steps that were performed in describing the preceeding BCD to binary conversion. The difference is that Benson and Tabbot shift the number to be converted in such a manner that the masking is always performed on the same position (the second) and the addition is always performed on the same position (the first).

The Benson and Tabbot invention can be described by working the conversion with pencil and paper as was done by the reader in the previous section. However, doing the Benson and Tabbot conversion with pencil and paper is somewhat cumbersome; the advantage of their technique is gained only when it is performed on a machine. Therefore, the reader is respectfully cautioned that the following pencil and paper description of the Benson and Tabbot invention will be quite difficult to understand if the basic principles of the examples in the preceding sections are not thoroughly understood.

Benson and Tabbot perform their conversion in a reentrant shift register. A reentrant shift register is a device which shifts the positions of a BCD or binary number. Positions shifted out of one end of the register are inserted into the other end of the register. For example, consider the BCD number fifty-three:

80	40	20	10	8	4	2	1	
0	1	0	1	0	0	1	1	Fifty-Three

If this number is shifted one position to the right in a

先ず、Benson and Talbot invention（あるいは、Benson and Talbot conversion）の発想であるが、ここで紹介されている方法は、pencil and paper でも実行できる。

The binary number nineteen is written by adding to the binary three a "1" in the position having a value of sixteen:

128	64	32	16	8	4	2	1	
0	0	0	1	0	0	1	1	"Nineteen"

The binary number forty-eight is written as the sum of thirty-two and sixteen as follows:

128	64	32	16	8	4	2	1	
0	0	1	1	0	0	0	0	"Forty-Eight"

The binary number fifty-three is written by adding a binary four and a binary one to forty-eight as follows:

128	64	32	16	8	4	2	1	
0	0	1	1	0	1	0	1	"Fifty-Three"

ここまでは、すでに検討した 10 進⇒2 進の変換である。

次に、128、64、32、16 を 80、40、20、10 に替えた表を使って、BCD への変換が紹介される。（0～9 は、2 進も BCD も同じ）

Plus a "ten":

80	40	20	10	8	4	2	1	
0	0	0	1	0	0	0	0	"Ten"

to give:

80	40	20	10	8	4	2	1	
0	0	0	1	1	0	0	1	"Nineteen"

The BCD number "fifty" is written as:

80	40	20	10	8	4	2	1	
0	1	0	1	0	0	0	0	"Fifty"

and the BCD number "fifty-three" is written by adding three to fifty to give:

80	40	20	10	8	4	2	1	
0	1	0	1	0	0	1	1	"Fifty-three"

次に BCD の 2 進への変換は、

Now consider the conversion of the BCD number ten into binary. The BCD number ten is:

80	40	20	10	8	4	2	1	
0	0	0	1	0	0	0	0	"Ten"

The reader knows that the binary number ten is written by a binary eight plus a binary two. The first step in converting the BCD number into binary is to erase, or mask the "1" in the ten's column of the BCD number. This gives:

80	40	20	10	8	4	2	1	
0	0	0	0	0	0	0	0	"zero"

The second step is to add a binary eight to give:

	80	40	20	10	8	4	2	1	
	0	0	0	0	0	0	0	0	
+	0	0	0	0	1	0	0	0	zero plus eight equals "eight"
	0	0	0	0	1	0	0	0	

The third and final step is to add the binary number two to produce:

	80	40	20	10	8	4	2	1	
	0	0	0	0	1	0	0	0	
+	0	0	0	0	0	0	1	0	Binary eight plus binary two equals binary "ten"
	0	0	0	0	1	0	1	0	

10 を 8＋4 に分解して、10_{10} である 0001 0000_{BCD} から 0000 1010_2 を得る方法を紹介している。

最後に、「53_{10}」をバイナリに変換する方法を紹介している。

As a final example of converting BCD to binary numbers by pencil and paper, consider the conversion of the BCD number fifty-three:

80	40	20	10	8	4	2	1	
0	1	0	1	0	0	1	1	"Fifty-Three"

First, the reader knows that three is the same in BCD and in binary so he will leave unchanged the "1's" in the

「10」の位の「1」を erase または mask する。10 を分解した「8」と「2」を加える。

first two positions. The next "1" is in the "tens" position. The reader will erase, or mask, this "1" to give:

80	40	20	10	8	4	2	1
0	1	0	0	0	0	1	1

The reader will replace this BCD ten with a binary ten by first adding a binary eight and then adding a binary two as follows. Adding the binary eight gives:

80	40	20	10	8	4	2	1	
0	1	0	0	0	0	1	1	
+ 0	0	0	0	1	0	0	0	
0	1	0	0	1	0	1	1	The number being converted plus binary eight equals a number intermediate BCD and binary.

Next, the reader will add binary two as follows:

80	40	20	10	8	4	2	1	
0	1	0	0	1	0	1	1	
+ 0	0	0	0	0	0	1	0	
0	1	0	0	1	1	0	1	Number intermediate BCD and binary plus binary two equals new number intermediate BCD and binary

Proceeding to the left, the reader will next convert the "1" in the position having a value of forty from BCD to binary. First, he will erase, or mask, the "1" from this column to give

80	40	20	10	8	4	2	1	
0	0	0	0	1	1	0	1	Number intermediate BCD and binary

He will replace this BCD forty with a binary 32 plus a binary eight. First, adding binary thirty-two gives:

128	64	32	16	8	4	2	1	
0	0	0	0	1	1	0	1	
+ 0	0	1	0	0	0	0	0	
0	0	1	0	1	1	0	1	Number intermediate BCD and binary plus binary thirty-two equals new number intermediate BCD and binary

「53」の中の「40」を erase または mask する。分解した「32」と「8」を加える。

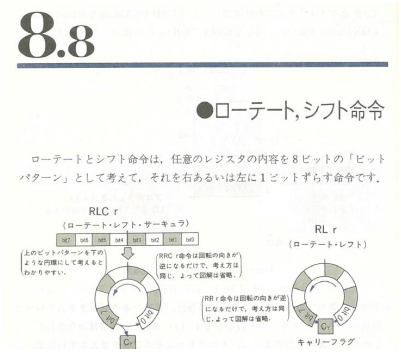
このようにして、2 進化 10 進数を 2 進数に変換する。

この解説によると、以上の内容と同じことを、Benson and Talbot invention は教示していることになる。「既知の 2 進化 10 進数の 2 進数の変換が、時間とメモリーをたくさん浪費する」ことから、シフトレジスタを利用して、マスク、シフト、1 の加算という演算を利用して達成使用とするものであると。

例は、同じ「53_{10}」である。「53_{10}」は 0101 0011 であった。
右に、位 3 つシフト（正確には「ローテート・シフト」命令）させる。

If the BCD number fifty-three is shifted three positions to the right it becomes:

0 1 1 0 1 0 1 0


8.8
●ローテート, シフト命令
ローテートとシフト命令は，任意のレジスタの内容を 8 ビットの「ビットパターン」として考えて，それを右あるいは左に 1 ビットずらす命令です。
RLC r
（ローテート・レフト・サーキュラ）
RL r
（ローテート・レフト）
キャリーフラグ

（村瀬、168 ページ）

以下、請求項 8 の（1）に該当する、「シフトレジスタ」に「ストア」するステップから始めると。

The example will be described with reference to the steps of the Benson and Tabbot claim 8. The first step is to store the binary coded decimal number fifty-three in a reentrant shift register so that the positions of this shift register are:

0 1 0 1 0 0 1 1

(2) The positions are shifted to the right by three places to produce:

0 1 1 0 1 0 1 0

(3) The binary "1" in the second position is masked out to produce:

0 1 1 0 1 0 0 0

(4) A binary "1" is added to the first position of the shift register to produce:

0 1 1 0 1 0 0 1

(5) The positions are shifted two places to the left:

1 0 1 0 0 1 0 1

(6) A "1" is added to the first position:

1 0 1 0 0 1 1 0

Thus far, Benson and Tabbot have performed the steps of masking out a "1" and replacing it with a binary eight plus a binary two. These steps were performed in the pencil and paper method of the previous section; however, in the Benson and Tabbot method, the masking was performed on the second position and both additions were made to the first position.

さらに、

(7) The number is shifted to the right by three places to produce:

1 1 0 1 0 1 0 0

At this point there is a "0", not a "1" in the second position of the reentrant shift register. Therefore, the number is shifted to the right one position at a time until there is a "1" in the second position. Shifting to the right by one position gives:

0 1 1 0 1 0 1 0

Since there is a "1" present in the second position, this "1" is masked out to produce:

0 1 1 0 1 0 0 0

そして、

A "1" is added to the first position to produce:

0 1 1 0 1 0 0 1

(5) The number is shifted two places to the left to produce:

1 0 1 0 0 1 0 1

(6) A "1" is added to the first position to produce:

1 0 1 0 0 1 1 0

At this point, the Benson and Tabbot technique has completed the masking of the "1" in the "forty" BCD position and its replacement with a binary "thirty-two" and a binary "eight".

In order to complete the process, the number is shifted to the right by three places to give:

1 1 0 1 0 1 0 0

A test is made to determine if there is a "1" in the second bit position and since there is not, the number is shifted to the right by one position to produce:

0 1 1 0 1 0 1 0

Finally, the number is shifted to the right by one more bit position to complete the conversion:

128	64	32	16	8	4	2	1
0	0	1	1	0	1	0	1

Fifty-Three

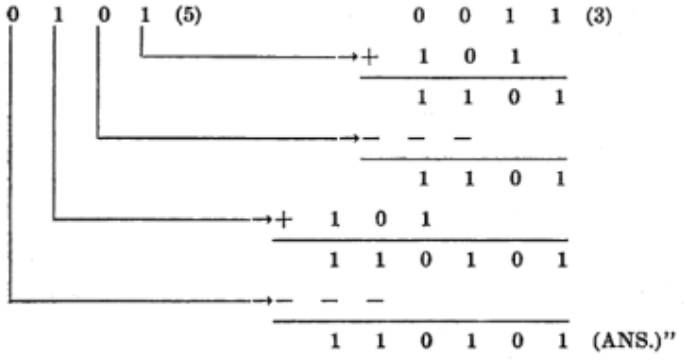
以上で、請求項８に書かれている手順は、「紙と鉛筆」による変換のアイデアに等価であることが分かった。

6. 請求項 13 について

被上告人が、米国特許庁に提出した説明によると、「手で」実行でき

Respondents submitted the following statement to the Patent Office to illustrate a manual way to carry out their method (claim 13):

"Finally, the method represented by these claims can also be carried out by hand, the shifting and adding operations being manual. The example used in the specification can be done by hand as follows:



であり、

(2) Since decimal $53=(5\times10^1)+(3\times1)$, binary coded decimal 53, which uses the decimal system of place values, must be the binary equivalent of 5 multiplied by $10^1{}_d$* plus the binary equivalent of 3 multiplied by 1_d, or $([0101\times10^1{}_d]+[0011\times1_d])$. When the multipliers $10^1{}_d$ and 1_d are converted to binary numbers we have $([0101\times1010]+[0011\times0001])$.

各位ごとに 2 進数表記の「重み」と「10」と「1」を掛け、和をとる趣旨であるとのことである。

Applying this process to the multiplication of the decimal number 10 by decimal 5, which is equivalent

to multiplying binary 1010 by binary 0101, the following steps are taken:

```
  1010
×0101
------
  1010
 0000
1010
000
------
110010.
```

Since 3 x 1=3 it is unnecessary to perform this step. Instead, the binary equivalent of 3 (0011) is simply added directly to the binary equivalent of 50 (110010) computed above:

```
  110010
+   0011
--------
  110101.
```

ただし、以上のような各位の重みに基数（の累乗）を掛けるという解釈とは別に、請求項 13 については、請求項 8 を一般化したものであるとの解釈も多い。

なお、Bartee、249 ページ、「2 進の乗算」は

「乗数の数字が 1 ならば、被乗数を単にコピーする部分積をつくる」
「乗数の数字が 0 ならば、部分積は 0 である」
「部分積をつくるごとに、1 桁だけ前の部分積の左にシフトする」

7. ベンソン判決の理由付け

Samuelson 教授は、「*Benson* Court did not clearly articulate the rationale for its decision」とされている一方、Chisum教授は、「*Benson ruled that a presumptively novel algorithm on number conversion useful for computer programming was unpatentable.*」と、ベンソン判決を評価しておられる。

他方で、ベンソン判決自体は、

Mackay Co. v. Radio Corp., 306 U. S. 86, 94, that "[w]hile a scientific truth, or the mathematical expression of it, is not a patentable invention, a novel and useful structure created with the aid of knowledge of scientific truth may be."

Rubber-Tip Pencil Co. v. Howard, 20 Wall. 498, 507. "A principle, in the abstract, is a fundamental truth; an original cause; a motive; these cannot be patented, as no one can claim in either of them an exclusive right."

Le Roy v. *Tatham,* 14 How. 156, 175. Phenomena of nature, though just discovered, mental processes, and abstract intellectual concepts are not patentable, as they are the basic tools of scientific and technological work.

O'Reilly v. Morse, 15 How. 62, "If this claim can be maintained, it matters not by what process or machinery the result is accomplished. … But yet, if it is covered by this patent, the inventor could not use it, nor the public have the benefit of it, without the permission of this patentee."

ベンソン判決の理由付けは、

It is conceded that one may not patent an idea. But in practical effect that would be the result if the formula for converting BCD numerals to pure binary numerals were patented in this case. The mathematical formula involved here has no substantial practical application except in connection with a digital computer, which means that if the judgment below is affirmed, the patent would wholly pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself.

8. 結語

請求項で開示されている内容を具体的に検討すると、①当該「方法」は、「紙と鉛筆」あるいは「手」で実現できること、②開示されている方法は、そのままデジタルコンピューターで実行できる「プログラミング」にまで達しておらず、概要を記述したものであること、③利用するリソースは、広く使用されているデジタルコンピューターの「シフトレジスタ」であったり、「ローテート・シフト」命令であったりすること、④BCD⇒2 進数変換それ自体が特別な用途をもたない（特定されていない）かはともかく、必要な場合は、誰でも利用できるものでないといけないこと、⑤BCD⇒2 進数変換の方法は、抽象的には分かっていたこと、等を考慮すると、請求項 8、請求項 13（これは、とくに、位別に 2 進表示された「重み」と 2 進表示された「基数」を、掛けるものである）の特許適格性を否定したのは、当然のことのように思われる。

ただし、ベンソン判決は「アルゴリズムそのものは特許にならない」と判示したとか、「一連の数字を他の数字に変換するだけのコンピュータプログラムに特許を与えれば、ソフトウエア

が依拠する数学の公式を先占させることになる」から特許を無効と判断したとか、「Benson 事件で述べられたアルゴリズム（2 進化 10 進数を 2 進数表現に変換すること）は、抽象的アイデアである」とかの評価は当たらないであろう。

（終わり）

APPENDIX TO OPINION OF THE COURT

Claim 8 reads:

"The method of converting signals from binary coded decimal form into binary which comprises the steps of

"(1) storing the binary coded decimal signals in a reentrant shift register,

"(2) shifting the signals to the right by at least three places, until there is a binary '1' in the second position of said register,

"(3) masking out said binary '1' in said second position of said register,

"(4) adding a binary '1' to the first position of said register,

"(5) shifting the signals to the left by two positions,

"(6) adding a '1' to said first position, and

"(7) shifting the signals to the right by at least three positions in preparation for a succeeding binary '1' in the second position of said register."

Claim 13 reads:

"A data processing method for converting binary coded decimal number representations into binary number representations comprising the steps of

"(1) testing each binary digit position '1,' beginning with the least significant binary digit position, of the most significant decimal digit representation for a binary '0' or a binary '1';

"(2) if a binary '0' is detected, repeating step (1) for the next least significant binary digit position of said most significant decimal digit representation;

"(3) if a binary '1' is detected, adding a binary '1' at the (i+1)th and (i+3)th least significant binary digit positions of the next lesser significant decimal digit representation, and repeating step (1) for the next least significant binary digit position of said most significant decimal digit representation;

"(4) upon exhausting the binary digit positions of said most significant decimal digit representation, repeating steps (1) through (3) for the next lesser significant decimal digit representation as modified by the previous execution of steps (1) through (3); and

"(5) repeating steps (1) through (4) until the second least significant decimal digit representation has been so processed."