伝統的複式簿記とコンピューター

―― 電子計算機複式簿記における仕訳帳の廃止 ――

藤田芳夫

はじめに*

電子計算機にふさわしい「新らしい簿記」の一つとして行列簿記 (matrix bookkeeping) が現われているが,(1) 簿記に対する電子計算機の応用は何も行列簿記に限られるわけではない。また,電子計算機簿記を考える場合,実践的な応用ということになれば,電子計算機を用いる情報処理システムの一環として考えなければならない。しかし,経営管理情報システム (MIS, Management Information System) の一環として電子計算機簿記を考える場合であっても,電子計算機複式簿記の基本理論は存在するのであって,経

---

* 本稿の執筆に際し,一橋大学名誉教授片野一郎博士,小樽商科大学教授石河英夫,同古瀬大六,同麻田四郎,同竹内清の諸先生から有益な批評と激励をいただいた,ここに記して感謝します。

(1) 行列簿記の現時点での意義を最初に主張したのは Richard Mattesich, "Towards A General and Axiomatic Foundation of Accounting," Accounting Research, Vol. 8, No. 4, pp. 328～355. であり,彼の所説はその著 Accounting and Analytical Methods, Measurement and Projection of Income and Wealth in the Micro and Macro-Economy, Richard D. Irwin, 1964 にくわしい。また行列簿記と数学との関連は J. G. Kemeny, A. Schleifer, Jr., J. L. Snell, G. L. Thompson, Finite Mathematics, with Business Applications, Prentice-Hall, 1962 が最初であろう。

以後,行列簿記に関するかなり多くの論文が現われているが省略する。わが国では越村信三郎著,「行列簿記のすすめ」,日本経済新聞社,1967年が単行本の最初であろう。そして簿記理論ないし簿記学説的な位置づけを与えようとされた文献に高寺貞男著,「簿記の一般理論――勘定簿記から行列簿記へ」,ミネルヴァ書房,1967年がある。

なお,筆者は行列簿記の実践的意義は電子計算機と結合する場合に発揮されると考えるが,この立場から行列簿記のプログラミングを考えたものとして拙稿「行列簿記とコンピューター」,雑誌「実務会計」,1966年10月,11月,12月号がある。

営管理情報システムであるがゆえに複式簿記の理論が消滅するとか，神秘化されるということはありえない。逆に簿記を含む経営管理情報システムを確立するためには，電子計算機を使用する場合の複式簿記の基本理論が要求されることにならざるをえない。(2)

この点を明らかにするため，以下，本稿ではルカ・パチオリ以来五百年の歴史に耐えてきたT字形勘定形式による伝統的複式簿記を電子計算機で遂行する場合の論理を分析したい。

ただし，本稿は前提条件としてフオートランを使用している。すなわち，プログラムが機械（電子計算機）とは一応無関係な独立性を保持しうるためである。(3)

1. 電子計算機簿記における勘定体系の展開

手記複式簿記 (manual double-entry bookkeeping) の骨格をなしているものは現金勘定，資本金勘定，売上勘定，仕入勘定といったような勘定が，それぞれバラバラなものではなく，一個の勘定体系 (accounts system) を

---

(2) たとえば，日本会計研究学会昭和42年度会計教育特別委員会の報告「会計教育と EDP」は次のようにのべている。

「トータル・インフォメーション・システムが実現する場合……それ自体固有な領域を占めてきた会計システムは，もはや独自な領域をもつものとして明確な形で認識することが困難となるであろう。」，と（同委員会，報告書，昭43年5月，p. 15）。

たしかに，同委員会が指摘しているように，トータル・インフォーメイション・システムないし MIS との関連において，会計というものを積極的ないし消極的に解する立場の相違という問題はおこりうるであろう。しかし，最も重要なことは，複式簿記がまったく無用なものになるのか，それとも形態の変化はあるとしても依然として必要なものかどうか，という点である。

この点について，複式簿記の形態は種々な変化をうけるとしても，手記複式簿記の基幹部分である勘定体系は依然として必要であり，かくして複式簿記は情報システムがどのように発展しても残る，というのが筆者の考えである。

(3) プログラミング言語として FORTRAN を使用することは，実は完全に機械から自由に (machine independent) なることではない。本稿は筆者が使用した機械 OKITAC 5090 H によってプログラムを実行しているので，たとえば森口繁一著「FORTRAN IV 入門」，東京大学出版会，1965年または同氏著「JIS FORTRAN 入門」，(上) 東京大学出版会，1968年と多少異なる点がある。

なしていることである。

しかもこの勘定体系に属する各勘定は借方と貸方の二つの計算区分が設定され，取引はその性質（仕訳）に応じて各勘定の借方，貸方に記入される。[4] 手記簿記システムの勘定を一見すると，各勘定には取引の分解結果（仕訳要素）が発生順に記録されていることがわかる。しかし勘定体系を投下原価数量（Invested Cost Figure）の計算体系と見る場合，各勘定に取引が発生順に記録されてゆくということ自体は手記簿記システムにおいても本質的な事柄ではない。むしろ，各勘定について，借方合計額と貸方合計額という二つの合計額が算出されるということが勘定体系にとって重要なのである。取引の発生順の記録（chronological record）は手記簿記システムにおいても仕訳帳（仕訳日記帳）の任務である。

以上のように考えると，投下原価数量の計算体系としての伝統的複式簿記を電子計算機により遂行する場合，各勘定の借方合計額と貸方合計額さえ算出できればよく，その内訳ないし発生順の記録はなくてもよいことになる。

そこで，T字型勘定をもつ伝統的複式簿記を，そのまま電子計算機化しようとすれば，N個の必要な勘定に対し，勘定の数の二倍，すなわち2N個の記憶場所を用意すれば，一つの勘定体系を主記憶装置のなかに展開することができる。

いま，決算のために必要な勘定である（閉鎖）残高勘定と損益勘定を除き，経常的に使用する勘定が現金勘定，売掛金勘定，備品勘定，買掛金勘定，資本金勘定，売上勘定，仕入勘定，給料勘定，営業費勘定の九個の勘定であるとしよう。

---

(4) 筆者はかつて単式簿記と複式簿記の相異について次のように指摘しておいた。「複式簿記は単式簿記が持っている簿記の財産管理機能を，計算と計算の照合という形で一箇の「完成した統一体」に結晶させ，計算結果に対しきわめて強いアカウンタビリティを持たせたものである」，と（小樽商科大学「商学討究」Vol. 16, No. 1, 1965年6月 p. 38）。この強いアカウンタビリティを保持するために借方と貸方の二つの計算区分が必要になるのである。

		第1列	第2列	第3列	第4列	第5列
現金勘定	借方	CASH 1	A 1	A11	A(1)	A(1, 1)
〃	貸方	CASH 2	A 2	A12	A(2)	A(1, 2)
売掛金勘定	借方	UKAKE 1	A 3	A21	A(3)	A(2, 1)
〃	貸方	UKAKE 2	A 4	A22	A(4)	A(2, 2)
備品勘定	借方	BIHIN 1	A 5	A31	A(5)	A(3, 1)
〃	貸方	BIHIN 2	A 6	A32	A(6)	A(3, 2)
買掛金勘定	借方	KKAKE 1	A 7	A41	A(7)	A(4, 1)
〃	貸方	KKAKE 2	A 8	A42	A(8)	A(4, 2)
資本金勘定	借方	CTAL 1	A 9	A51	A(9)	A(5, 1)
〃	貸方	CTAL 2	A10	A52	A(10)	A(5, 2)
売上勘定	借方	URI 1	A11	A61	A(11)	A(6, 1)
〃	貸方	URI 2	A12	A62	A(12)	A(6, 2)
仕入勘定	借方	SHIRE 1	A13	A71	A(13)	A(7, 1)
〃	貸方	SHIRE 2	A14	A72	A(14)	A(7, 2)
給料勘定	借方	SALRY 1	A15	A81	A(15)	A(8, 1)
〃	貸方	SALRY 2	A16	A82	A(16)	A(8, 2)
営業費勘定	借方	EXPSE 1	A17	A91	A(17)	A(9, 1)
〃	貸方	EXPSE 2	A18	A92	A(18)	A(9, 2)

（図表1－1）　**勘定体系の展開方法**

勘定体系を主記憶装置のなかに展開する方法には（図表1－1）に示すようにいくつもの方法がある。しかし，本稿では同図の第五列に示した方法，すなわち二次元の配列を利用する方法によることにする。

この二次元配列 A（9，2）を用いて勘定体系を展開すると言うことは（図表1－2）に示すように主記憶装置のなかに9行×2列の場所をとり，この場所全体を仮りに A（account matrix の先頭字 a をとった）と名付け，各勘定を縦軸（行）にそって配置し，各勘定の借方と貸方を横軸（列）にそつて定義してやることである。

行数＼列数		借方 1	貸方 2
現金勘定	1	A(1, 1)	A(1, 2)
売掛金勘定	2	A(2, 1)	A(2, 2)
備品勘定	3	A(3, 1)	A(3, 2)
買掛金勘定	4	A(4, 1)	A(4, 2)
資本金勘定	5	A(5, 1)	A(5, 2)
売上勘定	6	A(6, 1)	A(6, 2)
仕入勘定	7	A(7, 1)	A(7, 2)
給料勘定	8	A(8, 1)	A(8, 2)
営業費勘定	9	A(9, 1)	A(9, 2)

（図表1－2）　**二次元配列A(9, 2)による勘定体系の展開**

複式簿記の勘定はそれ自体が計算単位であるが，パチオリ以来，複式

簿記はこの一つの計算単位である勘定をプラスとマイナス，ないし積極と消極という二元的構造を持つものとして取扱ってきた。そして各勘定の純残高は取引の結果たえず変化するが，この各勘定の純残高は必要に応じて算出されればよいのである。したがって，各勘定は決算により内容を更新されるまで増加するだけで減少しない二つの変数（variable）によって構成されている。

上述のように二次元の勘定配列を利用すると，各勘定の借方は A(i, 1) で示され，貸方は A(i, 2) で示されることになる。

また，電子計算機は簿記のためだけでなく，いろいろな仕事に使用される。したがって，主記憶装置のうちAという名前をつけた9行2列計18ワードの場所をあらかじめクリヤーしておく方が安全である。そこで，これまで述べた勘定体系の展開をプログラムすれば

```
      DIMENSION A(9, 2)
      Do 1  J=1, 2
      Do 1  I=1, 9
    1 A(I, J)=0.0
```

というプログラムを書けばよい。[5]

2. 仕訳，元帳転記および試算表の作成

勘定体系が出来上ると，次に取引を仕訳し，各勘定に転記し，一定期間後に試算表を作成するのが伝統的な複式簿記の方法である。電子計算機簿記の場合も，大体これと同じであるが，いろいろと異なる点も出てくる。

第一に，手記複式簿記システムでは予め厳密に勘定体系を決定しておかなくても簿記，会計の常識に従ってどんどん仕訳して行き，不足する勘定が出

(5) タイプ宣言をしないで勘定配列Aを使用することは，勘定配列Aを実数型として使用することを意味している。以下，本稿では複式簿記の金額的数値を実数型で処理することにする。勿論，本稿の範囲内での処理に関するかぎり整数型で処理しても結果は同じである。

てくれば随時必要な勘定を追加してやることができる。電子計算機を使用する場合でも勘定を追加できないわけではないが，そのためにはプログラムを修正するか，勘定を正しく追加するための用意をしておかなければならない。

第二に，電子計算機簿記では手記複式簿記システムの場合とは異なり，二重分類が正しく行なわれたか否かを自働的に検証する手段としての試算表の意義は完全に失なわれる。しかし，他面では試算表は勘定記入の正確性を検証するという機能のほかに勘定一覧表という重要な機能をもっているから，電子計算機簿記でも，試算表を軽視することなく，逆に電子計算機システムと人間を結びつけるものとして重要な意味を持っている点に注意しなければならない[(6)]。というより，本稿のような方法で勘定体系を展開すると元帳自体がたえず勘定一覧表といえる形でしか存在しないことになる。試算表を作成するということは，それを人間の目でみえるようにラインプリンターから書出すことにすぎない。

そして，手記簿記システムに対する電子計算機（複式）簿記の最大の特色は仕訳帳が不必要になることである。さらに場合によっては仕訳そのものも人間が行なう必要はなくなるという点である。ただし，このような会計処理の自働化ないし非人間化は EDP 化した複式簿記のプログラムの中にどの程度の検査検証機能を持たせてあるか（試算表および仕訳帳の廃止について），また取引そのものについてどの程度の判断機能を持たせてあるか（人間による仕訳の廃止について）によって決まる問題である。この後者の例としては拙稿「電子計算機による販売管理会計[(7)]」で若干ふれたので，本稿では前者，すなわち電子計算機複式簿記における仕訳帳の廃止という点に力点を置き，

(6) 試算表が仕訳帳から元帳への転記の正確性を検査する機能の外に，勘定一覧表としての意味を持つことは従来から指摘されてきたが，会計機械化の進展に対応して転記の正確性を検査する機能は不必要になることが片野一郎博士によって指摘されている。cf. 片野一郎博士著，「簿記精説」，同文館，1962年版，p. 369。

(7) 拙稿，「電子計算機による販売管理会計」，小樽商科大学「商学討究」，Vol. 18, No. 3, 1968年2月，pp. 89～136。

その論理を明らかにしよう。

2－1　単純取引の仕訳，元帳転記，試算表

電子計算機複式簿記の一般理論として仕訳帳が廃止される論理を明らかにする前に，まずすべての取引が単純取引のみから成る場合の処理方法を明らかにしておこう。

いま取引例として下記の10個の取引がある。

1　現金135,000円をもって営業を開始した。
2　商品80,000円を掛仕入れした。
3　商品45,000円を掛売した（但し売価）。
4　商品40,000円を現金で仕入れた。
5　商品50,000円を現金売した（但し売価）。
6　売掛金39,000円を回収した。
7　買掛金58,000円を現金で支払った。
8　備品28,000円を購入した。
9　給料5,000円を支払った。
10　営業費3,000円を支払った。

この取引例を処理するために必要な勘定体系は（図表1－2）に示したように展開される。したがって電子計算機簿記において仕訳をどうするかということを次に明らかにしよう。

上述の取引例のうち取引No.1は，通常の仕訳によれば

（借）現金　135,000　　（貸）資本金　135,000……(1)

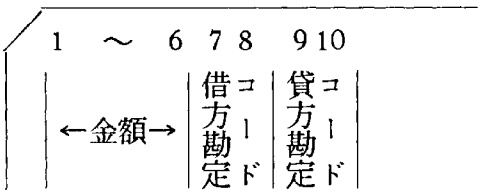
である。

この仕訳をみれば明らかなように，単純取引の仕訳では金額を二度書く必要はない。そこで，

135000，1，5

という表現で仕訳を行なうことができる。すなわち最初の135000は取引金額であり，二番目の1は（図表1－2）を見れば明らかな通り現金勘定の位置を示す行数であり，三番目の5は資本金勘定の行数である。

この例題では金額は6桁以下であるから，仕訳カードを次のように設計することにする。こうすると，前記の一〇個の取引は次のように仕訳される。



1)　135000　1　5
2)　80000　7　4
3)　45000　2　6
4)　40000　7　1
5)　50000　1　6
6)　39000　1　2
7)　58000　4　1
8)　28000　3　1
9)　5000　8　1
10)　3000　9　1

そこで，この仕訳カードを読込んで元帳に転記する，すなわち勘定配列Aに加算してやるためには

```
    2 READ(5, 101) TA, ID, IC
  101 FORMAT(F6.0, 2I2)
      IF(IA. GE. 999999.0) GŌ TŌ 3
      A(ID, 1)=A(ID, 1)+TA
      A(IC, 2)=A(IC, 2)+TA
      GŌ TŌ 2
```

というプログラムでよいことになる。

この部分プログラムの三行目で取引金額 TA(transaction amount の二つの頭文字をつづって名前とした）と六桁の9を比較したのは，仕訳カード・デックの最後にデータ・カードの終りを示すカードとしてこの場合許されるけれども，まず起りえない取引金額を使用したからである。

四行目の A(ID, 1)=A(ID, 1)+TA というステートメントは(1)式に示した伝統的仕訳が意味する借方勘定への転記を行なうためであり，五行目の A(IC, 2)=A(IC, 2)+TA というステートメントは同様に貸方勘定への転記を行なうものである。

仕訳を元帳に転記するという最も基本的な複式簿記の作業は実はこれで完了したことになる。しかし，このままでは人間の目に見えないし，また，電子計算機は他の仕事にも使用しなければならないから，ラインプリンターで

印刷するか磁気テープに記録するか，あるいはパンチ・カードにパンチ・アウトしておかなければならない。

いま，仕訳を転記し，内容が新らしくなった勘定体系（元帳）を試算表の形で，取り出すことにしよう。この場合，伝統的な合計試算表の形に慣れているわれわれにとって合計額の記入のない試算表は不自然である。そこで，転記が正確に行なわれたか否かを自動的に検証するための試算表，したがってまたそのための合計額の記入という意味ではなく，元帳を肉眼に見える形にし，また慣習に反しないため合計額が記入してある，そうした意味での試算表を作成するプログラムを考えてみることにする。

```
3 DO 4 J=1, 2
  DO 4 I=1, 9
  A(10, J)=A(10, J)+A(I, J)
4 CONTINUE
```

まず，合計額を記入する場所を勘定配列Aの最下行に追加することにする。こうすると借方合計額は左掲の DO ループの第一回目の処理で A(10, 1) に算出され，貸方合計額は第二回目の処理で A(10, 2) に算出される。

```
  1       DIMENSION A(10, 2), NAME(10)
  2       DO 1 J=1, 2
  3       DO 1 I=1, 10
  4     1 A(I, J)=0.0
  5       READ(5, 201) NAME
  6   201 FORMAT(10A7)
  7     2 READ(5, 101) TA, ID, IC
 10   101 FORMAT(F6.0, 2I2)
 11       IF(TA. GE. 999999.0) GO TO 3
 12       A(ID, 1)=A(ID, 1)+TA
 13       A(IC, 2)=A(IC, 2)+TA
 14       GO TO 2
 15     3 DO 4 J=1, 2
 16       DO 4 I=1, 9
 17     4 A(10, J)=A(10, J)+A(I, J)
 20       WRITE(6, 102)(A(I, 1), NAME(I), A(I, 2), I=1, 10)
 21   102 FORMAT(1H0, 40X, F10.0, 5X, A7, 3X, F10.0)
 22       STOP
 23       END
```

（図表２－１－１）　**複式簿記の最も単純なプログラム**

次にこの試算表をラインプリンターで印刷する命令は

```
      WRITE(6, 102) (A(I, 1), NAME(I), A(I, 2), I=1, 10)
  102 FORMAT(1H0, 40X, F10.0, 5X, A7, 3X, F10.0)
```

でよいことになる。

```
GENKIN  KAKEURIBIHIN          KAKEGAI       SHIHON  URIAGE
SHIIRE  KYURYO  EIGYOHI                             TOTAL
  135000  1  5
   80000  7  4
   45000  2  6
   40000  7  1
   50000  1  6
   39000  1  2
   58000  4  1
   28000  3  1
    5000  8  1
    3000  9  1
  999999
```

（図表２－１－２）　**データ・カード**

189000	GENKIN	154000
40000	KAKEURI	39000
68000	BIHIN	0
58000	KAKEGAI	60000
0	SHIHON	135000
0	URIAGE	95000
120000	SHIIRE	0
5000	KYURYO	0
3000	EIGYOHI	0
483000	TOTAL	483000

（図表２－１－３）
アウト・プットされた試算表

以上に説明した電子計算機複式簿記の最も単純なプログラムをまとめて示すと（図表２－１－１）のようになり，前記の例題についてのデータ・カードは（図表２－１－２）に，そして，このデータを（図表２－１－１）によって処理した結果は（図表２－１－３）のようになる。

２－２　複合取引と貸借平均検査

前節までの説明では借方金額と貸方金額の一致を検査する伝統的複式簿記のいわゆる自働検証機能についてほとんどふれるところがなかった。以下，この点を明らかにしてみよう。

単純取引だけからなる場合の処理プログラムを示した（図表２－１－１）により演算した結果，もし貸借一致の原則が破られるような事態がおこるとすれば，それはどのような原因によって発生するだろうか。

まず第一に考えられることは電子計算機の故障によって貸借が一致しない場合である。しかし，現在の電子計算機は何重にも安全装置が組込まれているから，機械的な故障にまったく気付かないで貸借不一致という結果に直面し狼狽するということはまず考えられない。

第二に考えられる原因は仕訳カードを作成する場合の勘定コードのミス・パンチである。すなわち，前節までの方法によれば

取引金額	借方勘定コード	貸方勘定コード

のようにパンチしたのであるから，金額は一度だけパンチしてある。したがって，手記簿記システムのように借方金額をたとえば2,000円としながら貸方金額を200円とするような誤りは絶対に発生しない。

しかし，借方勘定コードか貸方勘定コードか，いずれか一方でもパンチし忘れると，取引はたとえば次のようにパンチされ，これが読込まれる結果，変なことがおこる。(8)

1００	1	0
金額	借方勘定コード	貸方勘定コード

これと同じ種類の誤りは勘定コードに含まれていないコードを間違ってパンチした場合にも発生する。したがって，前節の例であれば，勘定コードは1から9までしかないのであるから，仕訳カードから読込んだ勘定コードが1から9までの範囲に含まれているか否かを検査してやればよいことになる。なお，この勘定コードの検査については第三章で詳細に扱うことにする。

---

(8) 配列要素 A(i, j) の添字 i または j がゼロになってはならない。その理由は配列 A の原点，すなわちすべての添字が1である A(1, 1) から番地の割付けが行なわれるからである。したがって，A(0, 0)，A(0, j)，A(i, 0) というような配列要素は取扱えないのである。

また第三に，勘定コードは正しくパンチしてあるが，金額パンチを落した場合にはどうなるであろうか。この場合，複式簿記の貸借平均原則に反するエラーにはならない。しかし，複式簿記の勘定体系に記録計算されている金額と，仕訳カードに転換される以前の原資料体系との間の不一致をさけることはできない。そこで，取引金額のパンチが脱落している場合にはエラー・メッセージを出すようにしておかなければならない。(9)

さて，このような配慮をし，プログラムを改善したとしても，複合取引が現われる場合にどう対処したらよいであろうか。すなわち，前節までの説明では仕訳の結果，借方および貸方に勘定が一つだけあらわれる単純取引に限られていた。したがって，貸借平均という複式簿記の特質にあまり注意を払う必要がなかった。しかし，仕訳の結果，借方または貸方のいずれか一方に勘定が二つ以上現われる場合，あるいは借方と貸方の両方に複数の勘定が現われるいわゆる複合取引の場合でも正しく処理できなければ，電子計算機による複式簿記として一般性を主張することはできない。

前例で第一の取引が「現金 55,000 円と備品 80,000 円を元入れして営業を開始した」という形に改められれば，通常の仕訳では

（借）現　金　55,000　　（貸）資本金　135,000
　　備　品　80,000

となる。

これを単純取引に分解するには

a）現　金　55,000　　資本金　55,000

b）備　品　80,000　　資本金　80,000

---

(9) 仕訳カードの取引金額について，そのパンチが脱落している場合のほか，取引金額自体が正しいか否か，さらに内容的に正当であるかどうかという問題がおきる。しかし，こうした点の検査をどおすべきかという問題は本稿では取扱わない。換言すれば，本稿で扱う問題は仕訳のいわば形式検査 (format verification) であって，これは取引そのものの形式検査の一部にすぎないし，ましてや取引の内容検査 (reasonableness verification) はまったく問題にならないのである。後者の問題は，実は EDP 会計システムを具体化する段階での問題であり，仕訳帳の廃止の問題ではなく，人間による仕訳の廃止の問題である。

のように考えるのが素直な方法である。また，現金式伝票の場合のように

a′）現　金 135,000　　資本金 135,000

b′）備　品　80,000　　現　金　80,000

というように分解することもできる。

いずれにせよ複合取引を単純取引に分解すると前節の仕訳カードのように

取　引　金　額	借方勘定コード	貸方勘定コード

という形に変形することができる。

しかし，かような方法を用いると，現金式伝票システムの場合に端的に現われるように振替伝票やその他の追加的方法が必要になる。そうしないと仕訳自体から取引の実態を把握することが困難になるからである。

したがって，根本的な問題は複合取引を処理する場合，どうしても単純取引に分解しなければならないかどうかという問題である。実はパチオリ以来の慣行的複式簿記システムの主流は，複合取引を単純取引に分解しない方法として成立しているのである。換言すれば，追加的技法ないし二次的な制約条件を考える必要のない，出来るだけ一般性をもった体系として慣行的複式簿記システムは成立している。この点を明らかにするため，以下，若干の説明を補足してみよう。

前例のように借方科目または貸方科目のいずれか一つが複数で他は単数である場合，単純取引に分解するには単数の科目の金額を分解すればよい。ところが，もし借方科目も複数で貸方科目も複数である場合にはどうすればよいであろうか。

たとえば，取得原価 @ ¥ 95.00 の一時的所有の有価証券二千株を 単価 ¥ 100.00 で売却し，売却手数料 ¥ 3,000 と有価証券取引税 ¥ 200 を差引き，手取金¥ 196,800 を当座預金とした，という場合に，仕訳は次のようにいくつもの方法が可能である。

1）当　座　預　金　196,800　　有　価　証　券　190,000

　　有価証券売却益　6,800

2)	当座預金	196,800	有価証券	190,000
	支払手数料	3,000	有価証券売却益	10,000
	租税公課	200		
3)	当座預金	196,800	有価証券売却	200,000
	支払手数料	3,000		
	租税公課	200		
	売却有価証券原価	190,000	有価証券	190,000

会計学的立場からすれば第三法が最もインフォーマティヴであり，取引が内包する情報を最も詳細に記録することができる。これに対し，第二法および第一法は取引の内包している資料を完全に示すことにならない。したがって，会計学的立場からすれば仕訳方法についてその優劣を論ずることができるし，当然そうしなければならないであろう。しかし，狭義の簿記理論の立場からすれば，これら仕訳の諸方法のうち，どれを選択すべきかを一概に断定することはできない。むしろ，いずれの方法が採用されようとも正確な処理が出来ることこそ肝要である。

かように考えると，われわれの目下の問題は上記三つの方法のうちどれが会計学的にすぐれているかと言うことではなく，単純取引に分解する際，どのような差異が生ずるかという点から考察しなければならない。

第一法の仕訳も第三法の仕訳も容易に単純取引に分解できる。ところが，第二法の仕訳は前二者と同じようには分解できない。もし，第二法を次のように分解すれば，かなり妥当性をもつように見える。

(2－a)	当座預金	190,000	有価証券	190,000
(2－b)	当座預金	6,800	有価証券売却益	6,800
(2－c)	支払手数料	3,000	有価証券売却益	3,000
(2－d)	租税公課	200	有価証券売却益	200

しかし，同じ第二法を次のように分解することもできるはずである。

(2－a′)	当座預金	10,000		有価証券売却益	10,000
(2－b′)	当座預金	186,800		有価証券	186,800
(2－c′)	支払手数料	3,000		有価証券	3,000
(2－d′)	租税公課	200		有価証券	200

この二つの分解方法の優劣を論ずることは，現在のわれわれにとり意味がない。何故なら簿記の一般理論は個々の企業がどのような会計処理規程を採用しようとも，あるいは記録手続の具体的な規定がどうあろうとも，そうした二次的制約や特殊性から自由な一般性を持たなければならないからである。かように考えると，上述のように複合取引を単純取引に分解する立場ではなく，複合取引を単純取引に分解せず，複合取引は複合取引のまま処理するという立場に立たなければならなくなる。即ち，パチオリ以来の慣行的複式簿記の主流をなしてきた方法の長所を生かしながら処理する方法を明らかにしなければならない。

複合取引を単純取引に分解しないで処理する方法は慣行的二帳簿制における普通仕訳帳の方法を反省してみればよい。左図に示してある通常の仕訳帳において借方金額と貸方金額を二度記入することは，単純取引の場合，実は不必要である。借方金額欄と貸方金額欄が同時に設定してある根本的理由は複合取引を処理するためである。前記有価証券の例でいえば，第一法から第三法にいたる仕訳方法のうち，いづれの方法を採用するにせよ借方金額の合計 20 万円と貸方金額の合計 20 万円を背後で算出し，両者が一致するか否かを検査しているのである。これが表面に現われたものが，実は仕訳帳の各頁の最下行における次頁繰越額の算出である。もし，すべての仕訳が単純取引だけからなるか，あ

普通仕訳帳

日付	摘要	元丁	借方金額	貸方金額

るいは複合取引は必ず単純取引に分解するというのであれば，仕訳帳に借方金額欄と貸方金額欄の二つを設定する必要はない。

したがって，電子計算機複式簿記においても，複合取引をそのまま処理するためには，元帳（すなわち勘定体系が展開してある配列 A（10，2））へ転記する前に，一つの複合取引の仕訳が貸借平均しているかどうかを検査してやればよい。

かように考へると複合取引の仕訳形式は次のようになる。すなわち，複合取引の借方要素にはそれが複合取引に属することを示すコードを貸方に与えてやり，複合取引の貸方要素にも同じく複合取引に属することを示すコードを借方に示してやるのである。換言すれば，実質的には複合取引を単純取引に分解しないのであるが，表面的には前節までの単純取引と同じ形に変形してやるのである。前記有価証券の例で言えば

当座預金	196,800 ……	複合取引の借方要素
支払手数料	3,000 ……	複合取引の借方要素
租税公課	200 ……	複合取引の借方要素
有価証券	190,000 ……	複合取引の貸方要素
有価証券売却益	10,000 ……	複合取引の貸方要素

この仕訳を

196,800，1，99，n
3,000，24，99，n
200，29，99，n
190,000，99，15，n
10,000，99，20，n

のように変形するのである。

一番左の数字は言うまでもなく金額であり，二番目の数字は借方勘定のコード・ナンバーか複合取引の貸方要素であることを示す特殊なコード・ナンバー99である。三番目の数字は複合取引の借方要素であることを示す99か

貸方科目のコード・ナンバーである。そして右端の数字nは取引ナンバーである。一つの取引に属するすべての仕訳カードを読終ったか否かを検査するため，実は右端の取引ナンバーが必要になる。これは普通仕訳帳でいえば，一つの仕訳を終るたびに摘要欄に引く横線と同じものである。

かように変形した複合取引の仕訳を直ちに勘定配列Aに転記しないで，複合取引の借方要素はその金額を配列DAに，その借方勘定コードは配列IDXに一時記憶し，貸方要素は金額を配列CAに，勘定コードを配列ICYに一時記憶しておく，そして複合取引のすべての仕訳要素を読終ったならば，借方要素，貸方要素のうち一つでも金額パンチを脱落したものがないかどうかを確かめ，この脱落がなければ借方合計額と貸方合計額とが一致するか否かを検査してやる。一致すれば勘定配列Aに転記し，一致しなければ仕訳に間違いがあるわけであるから，その旨ラインプリンターから書き出してやればよい。金額パンチに脱落がある場合にも，複合取引全体を書き出してやることは言うまでもない。

2－3　複合取引の貸借平均検査を含む電子計算機複式簿記のプログラム

以上のように考えると複合取引の貸借金額が一致するかどうかの検査を含む電子計算機複式簿記の処理手続は（図表2－3－1）のフローチャートのように示すことができる。そして，このフローチャートをコーディングすると（図表2－3－2）のようにFORTRANによる複式簿記のプログラムが出来上る。

以下，このフローチャートを中心にしてプログラムの説明をしよう。なお，具体的な取引例は（図表2－3－3）にデータ・カードとして示してあるので参照されたい。

（図表2－3－1）として示したフローチャートの基本的な考え方は，単純取引の場合（VI）の読込ボックスで読込まれた後，（X－1）ボックスで複合取引の借方要素でないことを判定され，（XI－1）ボックスで複合取引の貸

方要素でもない，つまり単純取引であることが結論され，(XII) ボックスで勘定体系を展開してある配列Aの該当場所（すなわち該当勘定）に転記されるのである。

そして，この転記の際，金額パンチが脱落しているか否かを判定し((XII−1) ボックス)，脱落しておればエラー・サブルーチンを呼び，それから次の仕訳カードの読込に移るのである。なお，これだけでは検査は充分とは言えない。何故なら複合取引であるにもかかわらず勘定コードのペンチを間違えて単純取引と誤認されるおそれがあるからである。しかし，この勘定コードの検査については次章で詳論することにする。

複合取引であれば，仕訳カードは (X−1) ボックスまたは (XI−1) ボックスで複合取引の借方要素であるか貸方要素であるかを判定され，借方要素であれば (X−2) ボックスで，貸方要素であれば (XI−2) ボックスで取引金額と勘定コードを配列 DA と IDX または配列 CA と ICY に一時記憶する。そして次に読込まれる仕訳カードが同じ複合取引に属するか否かを (IX) ボックスで検査する。もし，一つの複合取引の仕訳カードを全部読込んだならば (XIV−1) ボックスで複合取引の仕訳カードの中に一枚も金額パンチの脱落がなく (NZ=.TRUE.)，且つ貸借合計額が一致する (HSUM1=HSUM2) 場合にのみ (XV−1) ボックスで複合取引を一時記憶しておいた場所から勘定配列Aに転記するのである。

以上が（図表２－３－１）のフローチャートおよび（図表２－３－２）のプログラムの基本的な構造である。複式簿記を電子計算機によって遂行する場合，単純取引と複合取引がどのような順序で発生しても適切に処理できなければならない。（図表２－３－１）に示したフローチャートを複雑にしている原因は実はこの処理を適切に行なうためである。以下，この点を明らかにしてみよう。

単純取引と複合取引の組合せは

1)　仕訳カードを最初に読込む場合

2）　二番目の仕訳カードから最後の仕訳カードに至る部分の場合

3）　仕訳カードの処理を終る場合

の三つの場合について，組合せを明らかにしなければならない。

(1)　はじめて仕訳カードを読込む場合

いま単純取引を S，複合取引を P とすると，初めて仕訳カードを読込むとき，一枚のカードは単純取引である場合か，複合取引であるかの二つしかない。単純取引の場合，(III) ボックスで NUMB2 をゼロにしてあるから (VIII－1) の出口は YES であり，(VIII－2) で NUMB2 は NUMB に訂正される。したがって，(IX) における比較結果は当然 YES の出口から出る。そして (X－1)，(XI－1) を通り，(XII) ボックスで配列 A に転記される。

最初の仕訳カードか複合取引に属する場合には (X－2) か (XI－2) のいずれかで一時記憶され，同一複合取引に属する二枚目以下の仕訳カードが次々に読込まれることになる。そして同じ複合取引に属する仕訳要素はすべて一時記憶場所に貯蔵される。そして，単純取引であれ，複合取引であれ，取引ナンバーの異なるものが読込まれたことが (IX) ボックスで発見されると検査の後，勘定配列に転記される。

(2)　二番目の取引の仕訳カードから最後の取引まで

この場合は次表に示すように四つの組合せが考えられる。上述の最初の取引が複合取引であった場合に引続いて (5) 番の〔PS〕の場合を考えてみよう。この場合，NUMB2 には古い複合取引の取引ナンバーが入っているから，(IX) ボックスの判定結果は NO になる。そこで貸借平均検査をし，勘定配列に転記をしてやることは前述のとおりであるが，(XV－2) ボックスにおいて再び複合取引が出現する場合にそなえて N, M, HSUM1, HSUM2 および NZ(1) と NZ(2) をリセットし，(XIII－2) でいま読込ん

n番目 / n＋1番目	S	P
S	SS…(3)	PS…(5)
P	SP…(4)	PP…(6)

だばかりの単純取引の取引ナンバーを NUMB2 に入れかえておくのである。〔SS〕,〔SP〕,〔PP〕の場合については読者みずからの検討にゆだねたい。

ここで注意すべき点は，取引ナンバーを記憶するために NUMB だけでなく NUMB2 を設定し，各仕訳カードから読込まれる取引ナンバー (NUMB) と比較したこと。また複合取引の処理のため (V) ボックスであらかじめ N, M, HSUM1, HSUM2, および NZ(1), NZ(2) を開始状態にセットしただけでなく，(XV−1) で複合取引の転記が終了すると再びそれらをリセットし直した点である。また，第三に複合取引の貸借平均検査の結果，エラーとなったとき，エラー・メッセージを印刷した後，直ちに次のカードを読込んではならぬ点である。なぜなら次の取引が単純取引にせよ複合取引にせよ，次の仕訳カードの一枚目はすでに読込まれているからである。

(3) 最終カードの場合

最終カードは金額欄に六個の9をパンチしてある。したがって，最終カードを読込んだ場合，(VII) ボックスから (XVI) ボックスに進むが，このとき仕訳カードの組合せは最後の取引が単純取引であるか複合取引であるか，いずれかである。

最後の取引が単純取引であれば，この単純取引はすでに配列Aに転記済であるから，直ちに試算表を書き出せばよい。しかし，最後の取引が複合取引であれば，他の場合と同じように貸借平均検査をしてやらなければならない。このような理由で，この部分のフローチャートは（図表2－3－1）の左側のようになるのである。

次に，（図表2－3－2）に示したプログラムについて若干説明を加えよう。

このプログラムではサブルーチンを多く用いたため，COMMON 宣言を用いたこと。また金額パンチの脱落を検査する目的で二ワードの配列 NZ を用い，これに LOGICAL 宣言をしていることである。

第三に注意すべき点は配列 DA, IDX, CA, ICY の大きさをそれぞれ三ワ

ードにしている点である。これを本例では三ワードとれば充分だからで，必らずしも一般的ではない。複合取引を仕訳した場合の借方要素と貸方要素の一時記憶場所としては一〇ワード宛とっておけばまず充分ではなかろうか。

次にサブルーチン PSET について要点を説明しよう。このサブルーチンの N は複合取引の借方要素の数をカウントする場所であり，HSUM1 は借方要素の合計額を，そして NZ(1) は借方要素の中に一つでも金額パンチの脱落があれば，その内容を FALSE（偽）にするためである。こうしたパラメーターをいわば出発点の状態におくためのサブルーチンがこの PSET である。M, HSUM2 および NZ(2) は貸方要素について上述したことを行なうためである。

サブルーチン TRANS は複合取引の仕訳要素を一時記憶のための場所に複写し，借方と貸方の合計額を算出し，また金額パンチが脱落していないかどうかを NZ に判定しておくのである。

このサブルーチン TRANS で注意すべき点は複合取引の借方要素の数 N と貸方要素の数 M とについてである。複合取引の場合，借方要素にせよ貸方要素にせよ，少なくとも一個は出現する。したがって N と M は絶対にゼロに等しくない。すなわち，複合取引では

$$N \neq 0,\ M \neq 0$$

である。これに対し単純取引では

$$N = 0,\ M = 0$$

である。したがって（図表 2 − 3 − 1）の（XIII−1）および（XVI）で行なう判定は N とゼロまたは M とゼロを比較してはならないのである。何故なら，複合取引の借方要素（または貸方要素）を示す仕訳カードが完全に脱落した場合，エラー処理が適切にできなくなるからである。

複合取引の借方要素または貸方要素が完全に脱落した場合を含めて考えると，次表に示すように

	M=0	M≠0
N=0	① N=0, M=0	③ N=0, M≠0
N≠0	② N≠0, M=0	④ N≠0, M≠0

① N=0, M=0 のとき単純取引で

② N≠0, M=0 のとき複合取引の貸方要素が脱落したエラーであり

③ N=0, M≠0 のとき複合取引の借方要素が脱落したエラーであり

④ N≠0, M≠0 のとき借方または貸方の完全な脱落のない複合取引である。

したがって，(XIII−1) および (XVI) の判定は IF(N+M. NE. 0) GŌTŌ …… としなければならない。

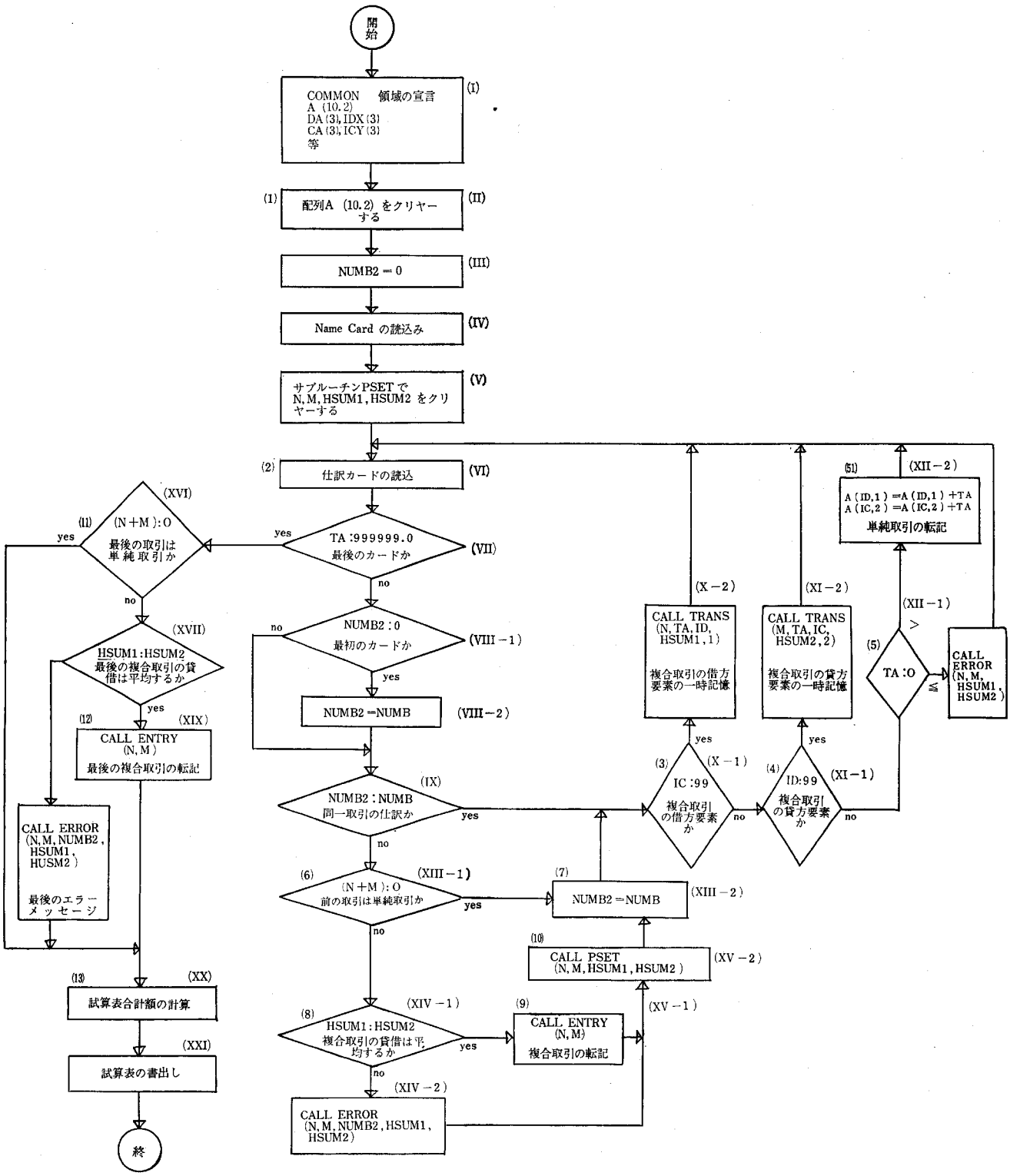
このようにすると，複合取引を処理するプログラムは手記複式簿記システムの場合と同様，ほぼ完全な検証能力を持ちうる。しかし，この場合でも借方と貸方がともに複数の取引で借方要素と貸方要素が一枚づつ脱落し，しかもその金額が等しい場合やその他の場合には手記簿記システムと同じくこのエラーを検出することはできない。

転記のためのサブルーチンは本稿の範囲内ではあまりにも単純であるから説明を省略する。

エラー・メッセージをラインプリンターから書き出すためのサブルーチン ERROR は単純取引について一箇所，複合取引については二箇所でコールされている。この ERROR サブルーチンで注意すべき点はサブルーチン TRANS の説明の最後に述べた点に関係する。すなわち，エラーの発生は N≠0, M≠0 でしかも貸借合計が一致しないときだけでなく，N=0, M≠0 のときおよび N≠0, M=0 のときのように借方要素または貸方要素が完全に脱落している場合にもエラーが発生する。したがって，この二つの場合にも正しくエラー・メッセージが出なければならない。

ERROR サブルーチンの9行目，17行目をみれば明らかな通り，DO ループを用いてエラーと判定された複合取引を書出している。したがって，もしエラー・サブルーチンを次のように構成すると

SUBRŌUTINE ERRŌR(……)



（図表２－３－１）　**複式簿記のフローチャート（試算表の作成まで）**

（各ボックス右肩のローマ数字は（図表２－５－３）のプログラムの右側の番号に該当し，左肩の数字はプログラムのステートメント・レーベルである。）

COMMON ……

WRITE (6, ×××) NUMB2

WRITE (6, ×××+1) (DA(I), IDX(I), I=1, N), HSUM1

WRITE (6, ×××+2) (CA(I), ICY(I), I=1, M), HSUM2

借方要素または貸方要素が完全に脱落した場合，脱落した要素の書出し命令のターミナル・パラメーターがゼロとなり矛盾がおきる（I=1, 0 という事態になる)。そこで（図表２－３－２）に示してある ERROR サブルーチンのように N および M がゼロか否かを検査し，それに応じて書き出すように修正しなければならないのである。

```
NO      SOURCE STATEMENT
 1      COMMON A(10, 2), DA(3), IDX(3), CA(3), ICY(3), NZ(2)          1
 2      COMMON ID, IC
 3      DIMENSION NAME(10)
 4      LOGICAL NZ
 5      DO 1 J=1, 2
 6      DO 1 I=1, 10                                                  2
 7    1 A(I, J)=0.0
10      NUMB2=0                                                       3
11      READ(5, 102)NAME                                              4
12  102 FORMAT(10A7)
13      CALL PSET(N, M, HSUMI, HSUM2)                                 5
14    2 READ(5, 101)TA, ID, IC, NUMB                                  6
15  101 FORMAT(F6.0, 2I2, I3)
16      IF(TA. GE. 999999.0) GO TO 11                                 7
17      IF(NUMB2. EQ. 0) NUMB2=NUMB                                   8
20      IF(NUMB2. NE. NUMB) GO TO 6                                   9
21    3 IF(IC. NE. 99) GO TO 4                                        10
22      CALL TRANS(N, TA, ID, HSUM1, 1)
23      GO TO 2
24    4 IF(ID. NE. 99) GO TO 5                                        11
25      CALL TRANS(M, TA, IC, HSUM2, 2)
26      GO TO 2
27    5 IF(TA. GT. 0.0)GO TO 51                                       12
30      CALL ERROR(N, M, NUMB2, HSUM1, HSUM2)
```

（図表２－３－２）（その１）

電子計算機複式簿記のプログラム（STEP 2 仕訳の計算上の検査を含む）

```
31        GO TO 2
32     51 A(ID, 1)=A(ID, 1)+TA
33        A(IC, 2)=A(IC, 2)+TA
34        GO TO 2
35      6 IF(N+M. NE. 0) GO TO 8                                    13
36      7 NUMB2=NUMB
37        GO TO 3
40      8 IF(N*M. NE. 0. AND. NZ(1). AND. NZ(2). AND.
                                    (HSUM1. EQ. HSUM2)) GO TO 9    14
41        CALL ERROR(N, M, NUMB2, HSUM1, HSUM2)
42        GO TO 10
43      9 CALL ENTRY(N, M)                                          15
44     10 CALL PSET(N, M, HSUM1, HSUM2)
45        GO TO 7
46     11 IF(N+M. EQ. 0) GO TO 13                                   16
47        IF(N*M. NE. 0. AND. NZ(1). AND. NZ(2). AND.
                                    (HSUM1. EQ. HSUM2)) GO TO 12   17
50        CALL ERROR(N, M, NUMB2, HSUM1, HSUM2)                    18
51        GO TO 13
52     12 CALL ENTRY(N, M)                                         19
53     13 DO 14 J=1, 2                                             20
54        DO 14 I=1, 9
55     14 A(10, J)=A(10, J)+A(I, J)
56        WRITE(6, 103) (A(I. 1), NAME(I), A(I, 2), I=1, 10)
57    103 FORMAT(1H0, 40X, F10.0, 5X, A7, 3X, F10.0)
60        STOP
61        END
62C
63        SUBROUTINE PSET(N, M, HSUM1, HSUM2)
64        COMMON A(10, 2), DA(3), IDX(3), CA(3), ICY(3), NZ(2)
65        LOGICAL NZ
66        N=0
67        M=0
70        HSUM1=0.0
71        HSUM2=0.0
72        NZ(1)=.TRUE.
73        NZ(2)=.TRUE.
74        RETURN
75        END
76C
77        SUBROUTINE TRANS(NM, TA, ICD, HSUM12, K)
```

(図表 2 − 3 − 2)（その 2）

電子計算機複式簿記のプログラム（STEP 2 **仕訳の計算上の検査を含む**）

```
100       COMMON A(10, 2), DA(3), IDX(3), CA(3), ICY(3), NZ(2)
101       LOGICAL NZ
102       NM=NM+1
103       GO TO(1, 2), K
104     1 DA(NM)=TA
105       IDX(NM)=ICD
106       HSUM12=HSUM12+TA
107       IF(TA. LE. 0.0)NZ(1)=.FALSE.
110       RETURN
111     2 CA(NM)=TA
112       ICY(NM)=ICD
113       HSUM12=HSUM12+TA
114       IF(TA. LE. 0.0)NZ(2)=.FALSE.
115       RETURN
116       END
117C
120       SUBROUTINE ENTRY(N, M)
121       COMMON A(10, 2), DA(3), IDX(3), CA(3), ICY(3)
122       DO 1 I=1, N
123     1 A(IDX(I), 1)=A(IDX(I), 1)+DA(I)
124       DO 2 J=1, M
125     2 A(ICY(J), 2)=A(ICY(J), 2)+CA(J)
126       RETURN
127       END
130C
131       SUBROUTINE ERROR(N, M, NUMB2, HSUM1, HSUM2)
132       COMMON A(10, 2), DA(3), IDX(3), CA(3), ICY(3), NZ(2)
133       COMMON ID, IC
134       WRITE(6, 300) NUMB2
135       WRITE(6, 101)
136       IF(N+M. EQ. 0) GO TO 1
137       IF(N. EQ. 0) WRITE(6, 102)
140       IF(N. NE. 0) WRITE(6, 103) (DA(I), IDX(I), I=1, N)
141       IF(N. NE. 0) WRITE(6, 104) HSUM1
142       IF(M. EQ. 0) WRITE(6, 105)
143       IF(M. NE. 0) WRITE(6, 106) (CA(I), ICY(I), I=1, M)
144       IF(M. NE. 0) WRITE(6, 107) HSUM2
145       RETURN
146     1 WRITE(6, 301) ID, IC
147       RETURN
150   300 FORMAT(1H0, 11HSHIWAKE NO., 2X, I3)
```

（図表２－３－２）（その３）

電子計算機複式簿記のプログラム（STEP 2 仕訳の計算上の検査を含む）

```
151  101 FORMAT(1H , 19HSHIWAKE NO MACHIGAI)
152  102 FORMAT(1H , 10X, 18HKARI KATA DATURAKU)
153  103 FORMAT(1H , 15X, F7.0, 2X, I2, 2X, 2H99)
154  104 FORMAT(1H , 10X, 16HKARI KATA GOKEI. F7.0)
155  105 FORMAT(1H , 10X, 19HKASHI KATA DATURAKU)
156  106 FORMAT(1H , 15X, F7.0, 2X, 2H99, 2X, I2)
157  107 FORMAT(1H , 10X, 16HKASHI KATA GOKEI. F7.0)
160  301 FORMAT(1H0, 10X, 18HTANJUN TORIHIKI
                                  NO/1H , 10X, 16HKINGAKU DATURAKU/1
162      1H0, 19X, I2, 2X, I2)
163       RETURN
164       END
165       END OF SOURCE*
```

(図表２－３－２)（その４）

電子計算機複式簿記のプログラム（STEP 2 **仕訳の計算上の検査を含む**）

```
GENKIN KAKEURIBIHIN KAKEGAISHIHON URIAGE SHIIRE
KYURYO EIGYOHITOTAL
    95000 199  1
    40000 399  1
   13500099 5  1
    80000 799  2
    4000099 4  2
    4000099 1  2
    25000 199  3
    20000 299  3
    4500099 6  3
    40000 799  4
    2000099 4  4
    2000099 1  4
    30000 199  5
    20000 299  5
    5000099 6  5
    39000 1 2  6
    58000 4 1  7
    28000 3 1  8
     5000 8 1  9
     3000 9 1 10
   999999
```

(図表２－３－３)　**データ・カード**

(本例は前出の単純取引だけからなる例題を適宜複合取引に修正したものである。なお，アウト・プットであるエラー・メッセージおよび試算表は本稿末尾補足資料をみられたい。)

3. 電子計算機複式簿記と仕訳帳の廃止

3－1 仕訳帳の基本的職能とそのプログラム化

前章では伝統的手記複式簿記において勘定記入の正確性を自働的に検証する手段としての試算表は，電子計算機複式簿記ではその意義を完全に失なうことを指摘し，あわせて仕訳帳の記入も不必要になること，さらにまた仕訳それ自体も必らずしも人間が行なう必要はないことを示唆しておいた。本章ではこのうち，電子計算機複式簿記において仕訳帳が廃止される運命にあることを明らかにしよう。

第二章第二節で伝統的複式簿記における普通仕訳帳の存在理由として複合取引の存在をあげ，複合取引の仕訳の「計算上の正確性」を確保するために仕訳帳がどのような意義を持っているかを検討し，またこうした論理を電子計算機のプログラムとして作成した。

しかし，仕訳の「計算上の正確性」をプログラムで検査することができたとしても，それだけでは仕訳帳を廃止することはできない。何故なら，仕訳帳の職能は複合取引における借方合計額と貸方合計額の一致を検査することだけでなく，同時に転記さるべき勘定が元帳に存在することを検証する手段でもあるからである。

実は伝統的な手記複式簿記では普通仕訳帳だけでなく特殊仕訳帳をも含めて，一般に仕訳帳の摘要欄と金額欄の中間にある元丁欄がこの職能を果たしている。この元丁欄の記入が元帳のページ数で行なわれようと勘定コード番号で行なわれようと，そこには本質的なちがいはない。すなわち，元丁欄への記入は転記漏れを防止するだけでなく，転記さるべき勘定が元帳に登録されている正規の勘定であることをチェックしているのである。

すなわち，伝統的手記複式簿記の基本的形態である元帳と仕訳帳からなる二帳簿制で，仕訳帳が有つ意義は取引を発生順に記録した歴史的記録であると同時に，複合取引の処理に際し借方要素と貸方要素の仕分け「計算上の正

確性」を確保すること，単純取引と複合取引とを問わず転記さるべき勘定がその企業の採用している勘定体系の正規のメンバーであるか否かを検査すること，そして最後に転記漏れや重複転記を防止すること，という四点にあるのである。

かように考えると単純取引の仕訳にさいし，何故に重複をいとわず同一金額を借方と貸方に二重に記載するかの理由が明らかになる。もし，仕訳の計算上の正確性だけを問題にするのであれば，すでに指摘したように単純取引について金額を二度書く必要はない。したがって，仕訳帳には複合取引だけを記載すればよいことになる。にも拘らず，伝統的手記複式簿記体系において単純取引をも仕訳帳に記載してきたのは形式的に複合取引に準じて処理したためではない。これは上述したように転記さるべき勘定が勘定体系の正規のメンバーであることをチェックし，転記漏れや重複転記を防止するためには仕訳帳に記載せざるをえなかったからである。

かように仕訳帳は伝統的手記複式簿記システムの中で極めて重要な職能を果たしてきた。しかし，周知のように仕訳帳の第一の職能である取引を発生順に記録する歴史的記録という職能は，会計システムにおいて「原始証憑」が完備するにつれて仕訳帳の職能としては失なわれつつある。

また，複合取引の仕訳の「計算上の正確性」を確保するという第二の職能は，前章で明らかにしたように電子計算機複式簿記のプログラムとして組立てることが可能である。したがって，もし上述の三番目の職能，すなわち「転記の正確性」の検証が電子計算機複式簿記のプログラムの中に完全に吸収されうるならば，伝統的手記複式簿記の代表的形態であり，理論的にも最も単純化された形態である二帳簿システムの中から仕訳帳を廃止することができる。すなわち複式簿記のなかから仕訳帳がその姿を消すことになる。電子計算機のもつ計算と論理判断能力は仕訳帳を廃止し，電子計算機複式簿記では二帳簿制ではなく，その主要簿は元帳のみの一帳簿制となる。

A. C. リトルトン教授はその名著「1900年までの会計の発達」のなかで仕

訳の運命について次のように述べている。

「仕訳（journal entry）は簿記の機構の中で重要な部門をしめている。普通の言葉で表現される取引を特有の構造をもった半統計的記録（a species of technically-formed, intermediate statistical records）に転換する手段，これが仕訳である。のみならず，それはあらゆる取引に内在する二重性を明白に表示するがゆえに，複式簿記のいちぢるしい特徴を現わす点において元帳（ledger entry）をしのぐものといえよう。

仕訳の重要性は現代の簿記，少なくともアメリカの簿記においてはいくぶん低下しつつあるようである。簿記発展の過程が究極において仕訳をまったく放逐するかどうかは予断することはできない。だが，仕訳が簿記上不可欠なものでないということだけはいえる。したがって，それが簿記上まったく消滅するかも知れないということは考え得られるところである。

おもうに，仕訳は複式勘定記入（double-entry account keeping）が完成した後に簿記の構造に付加されたものであろうし，いわば，生長後のおたまじゃくしの尾のように，いつか再び簿記の構造から脱落するときがくるかもしれない」，と。(10)

このリトルトン教授の極めて示唆に富む歴史的洞察と展望を本稿の立場から分析すると，リトルトン教授は仕訳記入（journal entry）と言う言葉によって，観念上のないし論理的操作としての仕訳そのものと，帳簿としての仕訳帳への記入とを慎重に区別されていることは明らかである。しかし，簿記の実践の面でいかにして両者が分離されうるかという論理が明らかにされてないことに気がつく。しかし，これは当然のことであって，電子計算機の出現以前においては，観念上のないし思考過程における取引要素の分解と計算，すなわち取引の認識と測定ということと，認識され，測定された取引を

(10) A.C. Littleton, Accounting Evolution to 1900, American Institute Publishing Co., Inc. 1933, reissued Russell & Russell, 1966, p. 107. なお，訳文は片野一郎博士の訳文に本稿の目的のため原語を入れ，またわずかに改めた。片野一郎博士訳註，「リトルトン会計発達史」，同文館，1952年，p. 166。

元帳に転記する準備として「計算および勘定を検査する」ということとは容易に分離できないからである。

すなわち，いわゆる帳簿組織論についての通説を省みれば明らかなように，簿記の組織は仕訳帳と元帳からなる二帳簿制にその最も単純な，かつ論理的にも最も簡潔な構造を見出すのである。そしてこの二帳簿制の発展形態として一方では特殊仕訳帳による仕訳帳の分化と補助元帳の多様化する途があり，他方では仕訳帳と元帳を一体化しようとする仕訳元帳制がある。

これら二つの途が，いずれかと言えば中小企業向きの「記帳合理化」の途であるのに対し，企業が大規模になるにつれ，諸部門間の取引情報の流れ，ないし経営活動データの流れと会計的情報の流れとを出来るだけ合理的に一致させ，迅速化と同時に記帳の合理化を図る方法として伝票会計が発展した。

これらすべての発展の途について，共通していることは，仕訳帳記入(journal entry)が廃止できないということである。特殊仕訳帳が仕訳帳を廃止するものでないことはあまりにも明白である。仕訳元帳制は仕訳帳を省略して元帳のみにしようとするものであると見ることもできるが，元帳を省略するために仕訳帳を変形したものと考えても差しつかえない。いずれにしろ，仕訳元帳制の場合，仕訳帳記入は残っている。

伝票会計の場合も仕訳帳記入は省略できない。すなわち現金式伝票方式による銀行簿記の日記帳がこの典型である。これまでの手記伝票会計システムで仕訳帳記入を省略すると，勘定体系である元帳と，これに対するインプットの集合体である伝票との間の一貫性ないし相互依存性が著るしく脆弱なものになるからである。この元帳記録と原資料との一貫性を保ち，伝票会計の安定度を確保する一つの手段として，伝票会計では逆に仕訳帳（統合仕訳帳）が益々必要になるという関係になる。(11)

---

(11) 会計記録は財務会計的にみても，管理会計的にみても，まず第一にアカウンタビリティを明らかにしなければならない。このため帳簿組織は一，記帳の正確性 *

取引の「認識と測定」と取引を元帳に転記する準備としての「計算および勘定」の検査とが容易に分離できないというのはこおいう意味である。

以下，本章では「転記の正確性」を確保するため，前章で述べた仕訳の計算上の正確性を検査するプログラムに「勘定の正確性」を検査するプログラムをどのように組み込めばよいかを分析しよう。すなわち，仕訳帳を存在させる第三および第四の理由を失なわせる電子計算機複式簿記のプログラムとは如何なる構造をもっているかを明らかにしよう。

なお，本稿では仕訳カードを前提にして，仕訳帳を廃止する条件を問題にしているのであるから，仕訳そのもの，換言すれば取引の認識と測定は依然として人間の仕事であるとしているのである。しかし，取引の認識と測定があらゆる場合に人間個有の領域として残ると主張しているのではない点に注意されたい。特定の領域における取引を認識し測定する過程とその論理を電子計算機のプログラムとして作成することができ，かつこのプログラムによって処理することが経済的である場合には取引の認識と測定も人間の手から奪われることは明らかである。ここまでくれば，仕訳と仕訳帳（あるいは取引の認識・測定と仕訳を転記するための準備作業としての検査）を区別することなく，人間による仕訳全体が廃止されることになる。すなわち，リトルトン教授の歴史的予見をさらにこえた進化が電子計算機により実現されることになる。

＊ のほかに，二，証憑の真実性，さらに三，取引の正当性という条件が要求される。

伝票会計で仕訳帳記入が必要な理由は多数の小紙片に分散している金額の合計計算上の正確性を検算によりチェックするとともに，仕訳帳の摘要欄に処理済伝票の番号を記入し，転記を受けた元帳勘定の金額と伝票さらにそのもとにある証憑との間の関係を一義的に固定するためである。アカウンタビリティという最も基本的な要請から，証憑，会計伝票，仕訳帳，元帳の間の一貫性ないし一義性が要求されるのである。

片野一郎博士は現金式伝票会計を採用しているわが国の「大銀行における日記帳は純粋総合仕訳帳である」，とされる（片野一郎博士著，「前掲書」，p. 277）。さらに貸借式仕訳伝票システムでも「総合仕訳帳」を正しく位置づけることが会計実務合理化のキイポイントであるとされている。（同博士，前掲書，p. 277）

3－2　勘定コード検査の考え方

転記の正確性を確保するためには仕訳カードにパンチしてある勘定コードが一つの会計システムが採用している正規の勘定体系に含まれるか否かを検査してやればよい。この場合，次の三つの場合が考えられる。

1．第二章で使用した例のように現金勘定から営業費勘定に至るすべての勘定が一連番号を持ち，途中に欠番がない場合である。この場合，勘定コードの検査は至って簡単であるので説明を省略する。

2．勘定コードの検査を複雑にする次の段階は勘定コードとして正の整数を使用してはいるが，途中に欠番がある場合である。この場合，勘定コード検査という立場からみると二つに区別しなければならない。

2－a．その第一は勘定コードに欠番があると共に勘定配列 A(I, J) には丁度その欠番に対応する番地が未使用のまま保存されている場合である。

2－b．これに対し，第二の場合は勘定コードに欠番があり，かつ勘定配列には欠番に対応する余裕がない場合である。この場合，表面上，勘定コードは整数型をしていても，勘定配列に転記するためには何のたしにもならない。勘定コード (i) はその勘定が勘定配列 A(I, J) の何行目に含まれるかを示さないからである。

そこで，勘定コードが整数型であっても，使用してない余裕番号を除き，使用している番号だけを含む勘定コードの配列をメイン・メモリーの内部に用意し，仕訳カード上の勘定コードが勘定コード配列の何番目のコードと一致するかをしらべ，その際計算したカウント数を媒介として勘定配列に転記してやらねばならない。

3．かように勘定配列 A(I, J) が余裕の行を含まず，勘定コードに欠番がある場合には，勘定コードがたとえ整数型であっても実質的には英数字コードで表現されている場合と同じである。

電子計算機には数字，アルファベット，特殊文字，片仮名等を処理するため，これらの文字や記号が電子計算機の内部でどのように表現されるかを約

束しているコード表がある。したがって，このコード表に忠実に従えば英数字コードで表現されている勘定コードでもその順序や欠番の有無を議論できないわけではない。しかし，人間がこのコード表に制約されるのは不便で不自然である。そこで英数字コードを使用する場合には欠番ないし余裕コードの有無は問題にしないことにする。

以上のような前提に立って考えると，結局勘定コードとして英数字コードを使用し，しかも勘定配列 A(I, J) の中に余裕行を含んでいる場合でも処理できるプログラムが最も一般的であることになる。そこで，以下，本稿ではこの場合の勘定コード検査の論理を明らかにすることにする。

このため，使用する勘定体系を（図表３－２－１）のように定める。

さて，このように定義すると，前章で使用した勘定配列 A(10, 2) は A(15, 2) に，勘定名配列 NAME(10) は NAME(15) に修正しなければならないのは勿論，このほかに英数字勘定コードを記憶しておく配列 ICODE(15) が必要になる。また仕訳カード上のデータを一時記憶するための場所として配列 DA(3), CA(3), IDX(3), ICY(3) が必要であったが，この外に勘定

	勘定配列Aにおける行数	勘定コード
現金勘定	1	A 1
売掛金勘定	2	A 2
備品勘定	3	A 3
（余裕）	4	（コードなし＝ブランクのまま）
買掛金勘定	5	L 1
（余裕）	6	（コードなし＝ブランクのまま）
資本金勘定	7	C 1
売上勘定	8	R 1
仕入勘定	9	E 1
給料勘定	10	E 2
営業費勘定	11	E 3
（余裕）	12	（コードなし＝ブランクのまま）
（余裕）	13	（〃）
（余裕）	14	（〃）
合計算出場所	15	（〃）

（図表３－２－１）　**勘定配列と勘定コードの関係**

コードが勘定コード配列の何番目に出てきたかを記憶しておくための場所 NDA(3) と NCA(3) が必要になる。また前章では NZ(2) を論理値としてリザーヴしておいたが，この他に借方要素および貸方要素のコードが正しいか否かを検査した結果を記憶しておく場所 CODE1(3), CODE2(3) が必要になる。

IDX, ICY の他に NDA, NCA を追加したのは英数字コードでは IDX, ICY に一時記憶される勘定コードが転記のために役立たないので，勘定コードをチェックした DO ループのカウント数 I をそれぞれの仕訳要素について記憶しておくためである。

CODE1 と CODE2 を追加した理由は次のとおりである。前章のプログラムではエラーの検出は計算上の仕訳間違いについてであったが，本章で追加しようとしているのは勘定コードにおける仕訳間違の検出である。たとえば複合取引の仕訳のうち，ただ一つの勘定コードがエラーであっても，その仕訳全体を転記することなく取出し，かつ，どの勘定コードがエラーであるかを示さねばならない。なぜなら，勘定の数が増加すると人間がエラー・メッセージの中からどれがエラー・コードで，どれが正しいコードであるかを判断するのは間違いやすく不能率だからである。したがって CODE1 と CODE 2 は論理値であることを宣言しておかねばならない。

なお，このように COMMON 宣言をしておくべきものが多くなると，TABLE CONSTANT を修正する必要がでてくるので，以下，本稿では COMMON 宣言を次のようなまとめた形を採用することにする。すなわち

```
COMMON  A(15, 2), NZ(2), ICODE(15)
COMMON  DCA(3, 2), IXY(3, 2, 2), CODE(3, 2)
```

ここに，

```
DCA(3, 1)  ……DA(3)
DCA(3, 2)  ……CA(3)
IXY(3, 1, 1)……NDA(3)←NDA
```

IXY(3, 2, 1)……NCA(3)←NCA

IXY(3, 1, 2)……IDX(3)←IDX

IXY(3, 2, 2)……ICY(3)←ICY

CODE(3, 1) ……CODE1(3)

CODE(3, 2) ……CODE2(3)

である。

3－3　仕訳の「計算正確性」検査プログラムと「転記正確性」検査プログラムの結合

前章では複合取引の借方要素は金額を DA に勘定コードを IDX に，貸方要素はそれぞれ CA と ICY に一時記憶し，金額のパンチもれがあるかないか，および貸借会計が一致するか否かを検査したのであるが，本章では上述の考え方を基本としてこうした金額的正確性のほかに，それぞれの仕訳要素がもっている勘定コード自体の検査を行なうのである。

ところで，上述したように勘定コードが英数字コードをとる場合を処理しようとすると（図表 2－3－2）に示したプログラムのうち次の諸点を修正しなければならない。

その第一点は 14，15 行にある仕訳カードの読込命令とその FORMAT を

2 READ(5, 101) TA, ID, IC, NUMB

101 FORMAT(F6.0, 2A2, I3)

と修正しなければならない。またこの修正に対応して 21 行と 24 行の比較を

3 IF(IC. NE. MARK99) GŌ TŌ 4

4 IF(ID. NE. MARK99) GŌ TŌ 5

というように修正しなければならない。当然 11 行目の READ 命令も

READ(5, 102) NAME

READ(5, 102) ICODE

READ(5, 102) MARK99

として，99を英数字モードで MARK99 という場所に読込んでおかなければならない。

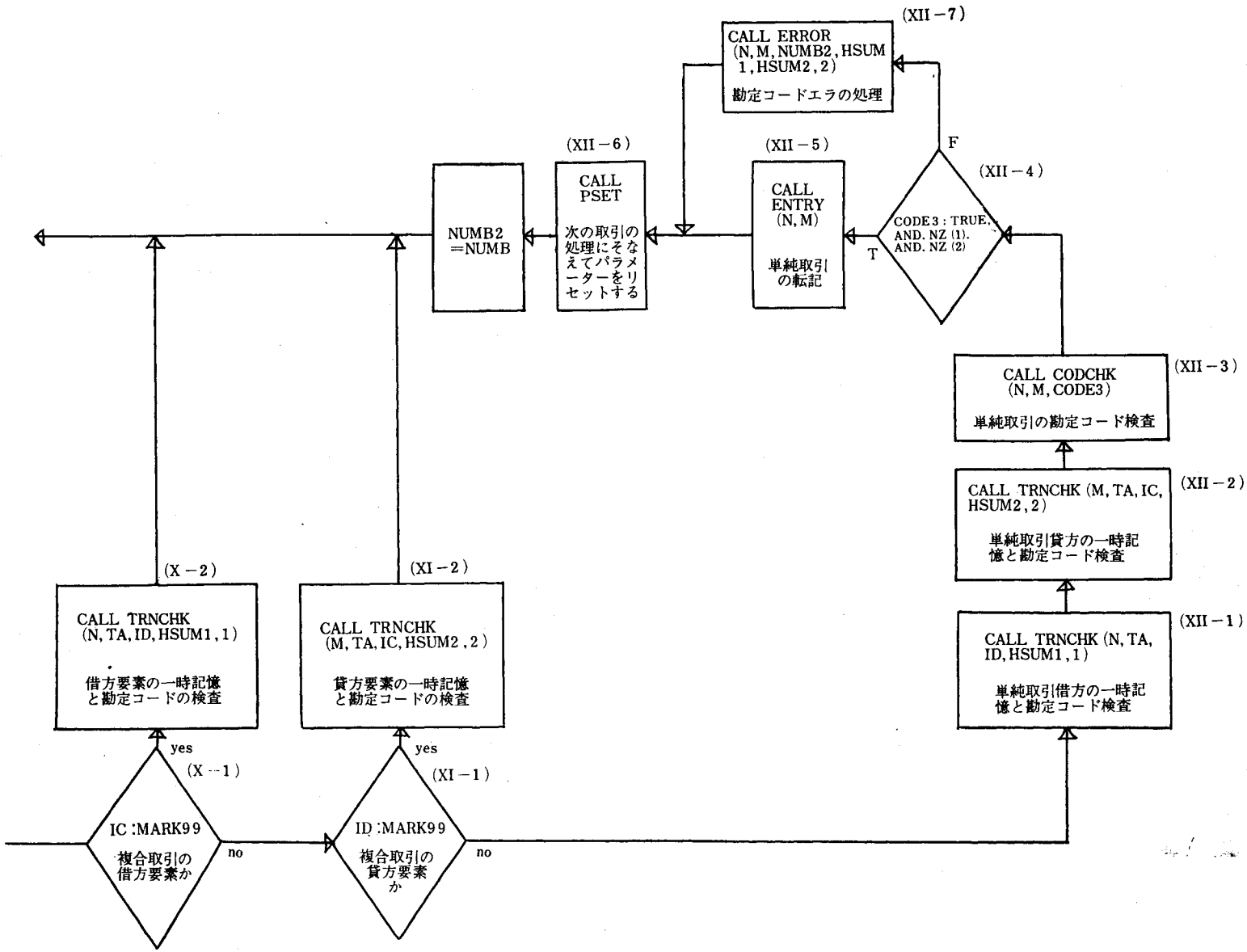
修正すべき第二点は前章のサブルーチン TRANS である。前章では一時記憶場所に転記し，金額パンチの脱落を検査しただけであったが本章では更に勘定コードを一つ宛検査し，その正否の判定結果を論理値として CODE に貯蔵しておく，したがって，サブルーチンの名称も TRNCHK と変更する。また，こうして検査した取引の勘定コードのうち一つでもエラーがあるか否かを調らべるサブルーチンを CODCHK と名付ける。こうすると，前章の（図表2－3－1）に示したフローチャートの関係部分は（図表3－3－1），（図表3－3－2）および（図表3－3－3）のように修正される。

次にサブルーチン TRNCHK の内容を少し説明しよう。本章では前章に組立てたサブルーチン TRANS の下に勘定コードの検査部分を接続すればよい。すなわち，

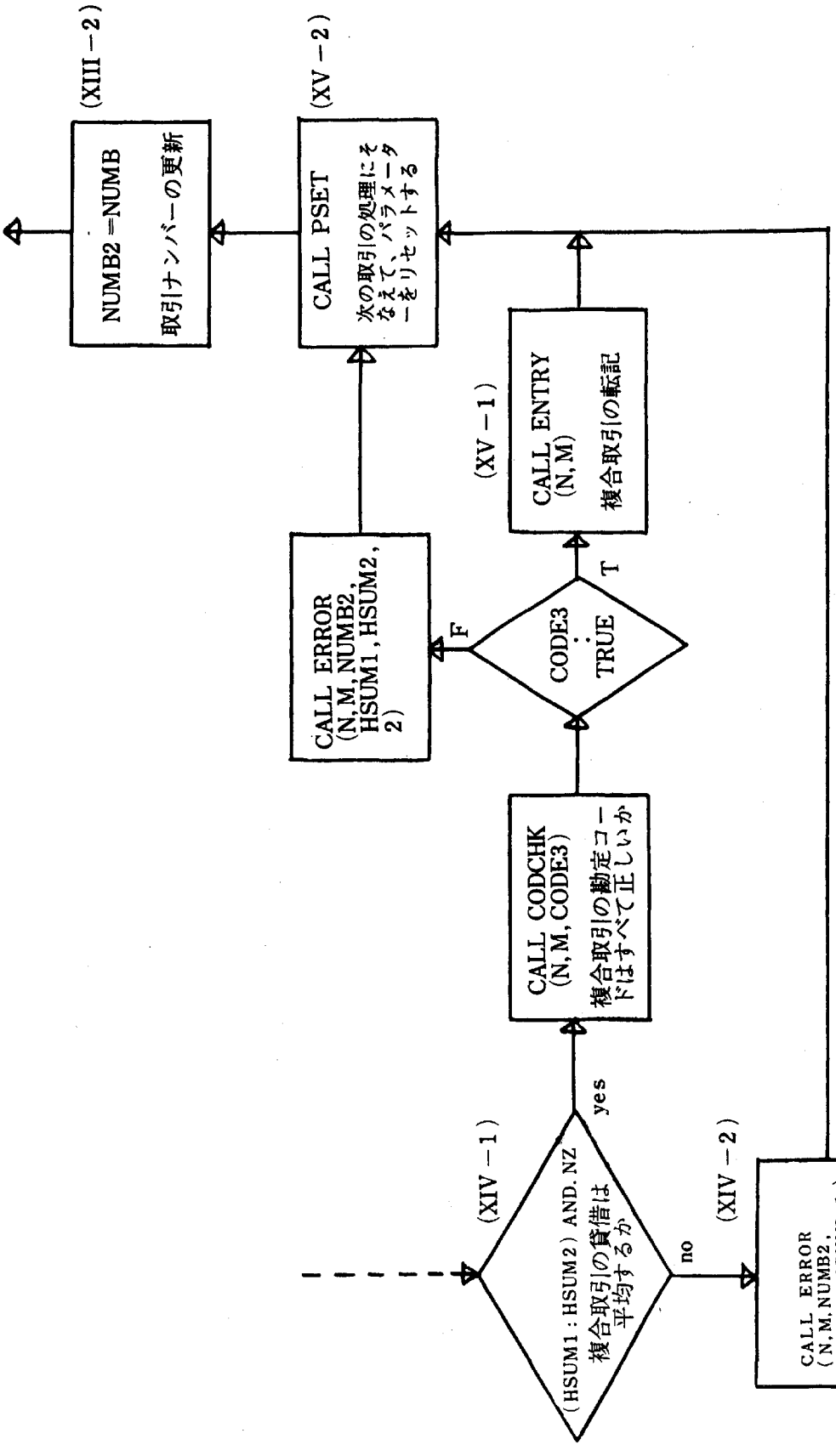
```
      IF(IXY(NM, JA, 2) .NE. ICODE(IA)) GO TO 2
1     CODE(NM, JA)=.FALSE.
      IXY(NM, JA, 1)=0
      RETURN
2     DO 3 J=1, IA
      IF(J. EQ. IA) GO TO 1
      IF(ICODE(J) .NE. IXY(NM, JA, 2)) GO TO 3
      CODE(NM, JA)=.TRUE.
      IXY(NM, JA, 1)=J
      RETURN
3     CONTINUE
```

を接続すればよいことになる。

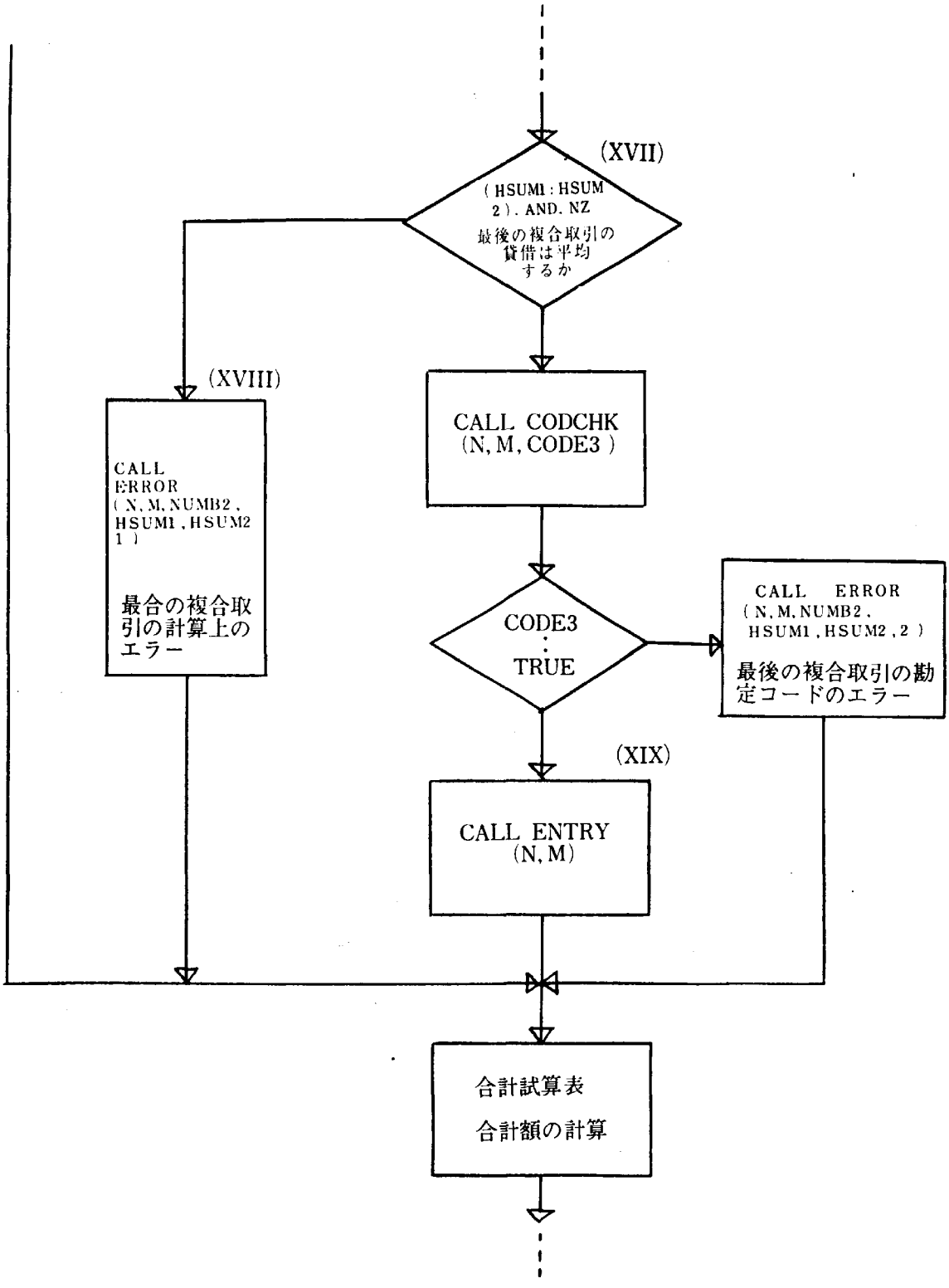
サブルーチンTRNCHKを上述のように作成すると単純取引と複合取引とを問わず仕訳の一時記憶が終了したとき，借方合計額と貸方合計額が算出済



（図表３－３－１）　**取引要素を一時記憶する部分の修正**



（図表３－３－２）　**複合取引を転記する部分の修正**



（図表３－３－３）　**最後の複合取引を検査する部分の修正**

になっているだけでなく，仕訳要素のうちに金額パンチの脱落したものがあるか否か，すべての勘定コードが正規の勘定体系のメンバーであるか否かについての情報が完全に集められていることになる。また転記すべき勘定配列の行数は借方要素については IXY(I, 1, 1) に，貸方要素については IXY(I, 2, 1) に記入されていることになる。

したがって，単純取引の場合には TRNCHK が終了した後に，複合取引の場合には別の取引に属する仕訳カードが読込まれたとき，借方合計と貸方合計の一致および NZ の検査を行なうだけでなく一つの仕訳の勘定コードが全部真であるか否かを検査しなければならない。このためのサブルーチンを前述のように CODCHK と名付ける。

仕訳の借方要素の個数 N と貸方要素の個数 M はサブルーチン TRNCHK で計算してあるので，基本的に

```
      DO ⓐ I=1, N
      IF(CODE(I, 1)) GO TO ⓐ
      GO TO ⓑ
ⓐ CONTINUE
```

とし，ⓑで全体としてこの仕訳は「偽」であることを示し，ⓐの次で「真」となるようにしてやればよい。この全体として真か偽を判断するために論理変教 CODE3 を使用したのである（(図表３－３－４）プログラムのサブルーチン CODCHK 参照のこと)。

かように考えると，(図表２－３－１)に示したフローチャートおよび(図表３－３－１)，(図表３－３－２)，(図表３－３－３）に示したフローチャートは大巾に単純化することができる。何故なら単純取引でも複合取引でも，仕訳を一時記憶して，検査を行なわねばならず，そのため，丁度，伝統的な複式簿記の二帳簿制で単純取引であると複合取引であるとを問わず，すべて仕訳帳に記入したのと同じことになるからである。

このような結論にもとづいてコーデイングしたものが（図表３－３－４）

に示してあるプログラムである。このプログラムのメイン・プログラムを見れば明らかなように，きわめて単純なものになっている。しかし，このためには実は仕訳の計算上のエラーおよび勘定コードのエラーについての総合的な検査を行なうサブルーチン TEST を追加しなければならない。このサブルーチン TEST の論理構造を説明するのが本来であるが，既に本稿の紙数もかなり大きくなったので，別稿にゆずることにする。

以上により，仕訳の計算上の正確性だけでなく「転記の正確性」を保証しうる電子計算機複式簿記プログラムの主要な点を指摘したことになる。この他に説明を補足すべき点があるとすれば，サブルーチン ENTRY を修正した点とサブルーチン ERROR の修正点とであろう。しかし，前者の修正は極めて簡単であるから説明を省略する。サブルーチン ERROR で説明を補足すべき点は勘定コードについて，間違っているものについてはその右側に「F」を，正しいものについては「T」を明示するようにプログラムを作成してある点である。

以上により複式簿記の仕訳が与えられるならば，その計算上の正確性と転記の正確性を保証してくれる電子計算機複式簿記のプログラムが完成したことになる。これをまとめて示すと（図表３－３－４）のようになる。このプログラムにより，電子計算機簿記では仕訳帳および転記の正確性を検査する手段としての試算表は複式簿記の構造のなかから存在理由を失なうことになる。そして複式簿記のいわゆる主要簿は元帳だけとなり，電子計算機複式簿記は二帳簿制ではなく一帳簿制となる。ただし，前述したように，仕訳それ自体が人間の手を必要としなくなるためには更に他の条件が必要である。しかし，それにしても電子計算機の出現により，簿記はパチオリ以来，最も大きな衝激をうけ，最大の変革に直面していることは以上の分析によって疑いもなく明らかである。

```
      COMMON A(15, 2), NZ(2), ICODE(15)
      DIMENSION NAME(15)
      LOGICAL NZ
      IA=15
      IB=IA-1
      DO 1 J=1, 2
      DO 1 I=1, IA
    1 A(I, J)=0.0
      NUMB2=0
      READ(5, 102) NAME
      READ(5, 102) ICODE
      READ(5, 102) MARK99
  102 FORMAT(10A7)
      CALL PSET(N, M, HSUM1, HSUM2)
    2 READ(5, 101) TA, ID, IC, NUMB
  101 FORMAT(F6.0, 2A2, I3)
      IF(TA. GE. 999999.0) GO TO 7
      IF(NUMB2. EQ. 0) NUMB2=NUMB
      IF(NUMB2. NE. NUMB) GO TO 6
    3 IF(IC. NE. MARK99) GO TO 4
      CALL TRNCHK(IA, N, TA, ID, HSUM1, 1)
      GO TO 2
    4 IF(ID. NE. MARK99) GO TO 5
      CALL TRNCHK(IA, M, TA, IC, HSUM2, 2)
      GO TO 2
    5 CALL TRNCHK(IA, N, TA, ID, HSUM1, 1)
      CALL TRNCHK(IA, M, TA, IC, HSUM2, 2)
      GO TO 2
    6 CALL TEST(N, M, NUMB2, HSUM1, HSUM2)
      NUMB2=NUMB
      GO TO 3
    7 CALL TEST(N, M, NUMB2, HSUM1, HSUM2)
      DO 8 J=1, 2
      DO 8 I=1, IB
    8 A(IA, J)=A(IA, J)+A(I, J)
      WRITE(6, 103) (A(I, 1), NAME(I), A(I, 2), I=1, IA)
  103 FORMAT(1H0, 40X, F10.0, 5X, A7, 3X, F10.0)
      STOP
      END
50C
```

（図表３－３－４）（その１）

電子計算機複式簿記のプログラム　——試算表作成まで——

```
 51      SUBROUTINE PSET(N, M, HSUM1, HSUM2)
 52      COMMON A(15, 2), NZ(2)
 53      LOGICAL NZ
 54      N=0
 55      M=0
 56      HSUM1=0.0
 57      HSUM2=0.0
 60      NZ(1)=.TRUE.
 61      NZ(2)=.TRUE.
 62      RETURN
 63      END
64C
 65      SUBROUTINE TRNCHK(IA, NM, TA, ICD, HSUM12, JA)
 66      COMMON A(15, 2), NZ(2)
 67      COMMON DCA(3, 2), IXY(3, 2, 2), CODE(3, 2)
 70      LOGICAL NZ, CODE
 71      NM=NM+1
 72      DCA(NM, JA)=TA
 73      IXY(NM, JA, 2)=ICD
 74      HSUM12=HSUM12+TA
 75      IF(TA. LE. 0.0) NZ(JA)=.FALSE.
 76      IF(IXY(NM, JA, 2). NE. ICODE(IA)) GO TO 2
 77    1 CODE(NM, JA)=.FALSE.
100      IXY(NM, JA, 1)=0
101      RETURN
102    2 DO 3 J=1, IA
103      IF(J. EQ. IA) GO TO 1
104      IF(ICODE(J). NE. IXY(NM, JA, 2)) GO TO 3
105      CODE(NM, JA)=.TRUE.
106      IXY(NM, JA, 1)=J
107      RETURN
110    3 CONTINUE
111      END
112C
113      SUBROUTINE TEST(N, M, NUMB2, HSUM1, HSUM2)
114      COMMONA(15, 2), NZ(2)
115      LOGICAL CODE3, NZ, TESTS
116      CALL CODCHK(N, M, CODE3)
117      TESTS=.FALSE.
120      IF(ABS(HSUM1-HSUM2). LT. 0. 1. AND. NZ(1). AND. NZ(2))
                                                  TESTS=.TRUE.
```

(図表3－3－4)　(その2)

電子計算機複式簿記のプログラム　——試算表作成まで——

```
121       IF(TESTS) GO TO 1
122       CALL ERROR(N, M, MUMB2, HSUM1, HSUM2, 1)
123     1 IF(CODE3) GO TO 2
124       CALL ERROR(N, M, NUMB2, HSUM1, HSUM2, 2)
125       GO TO 3
126     2 IF(TESTS) CALL ENTRY(N, M)
127     3 CALL PSET(N, M, HSUM1, HSUM2)
130       RETURN
131       END
132C
133       SUBROUTINE ENTRY(N, M)
134       COMMON A(15, 2), NZ(2), ICODE(15)
135       COMMON DCA(3, 2), IXY(3, 2, 2)
136       DO 1 I=1, N
137     1 A(IXY(I, 1, 1), 1)=A(IXY(I, 1, 1),1)+DCA(I, 1)
140       DO 2 J=1, M
141     2 A(IXY(J, 2, 1), 2)=A(IXY(J, 2, 1), 2)+DCA(J, 2)
142       RETURN
143       END
144C
145       SUBROUTINE CODCHK(N, M, CODE3)
146       COMMON A(15, 2), NZ(2), ICODE(15)
147       COMMON DCA(3, 2), IXY(3, 2, 2,), CODE(3, 2)
150       LOGICAL CODE, CODE3
151       DO 1 I=1, N
152       IF(CODE(I, 1)) GO TO 1
153       GO TO 3
154     1 CONTINUE
155       DO 2 I=1, M
156       IF(CODE(I, 2)) GO TO 2
157       GO TO 3
160     2 CONTINUE
161       CODE3=.TRUE.
162       RETURN
163     3 CODE3=.FALSE.
164       RETURN
165       END
166C
167       SUBROUTINE ERROR(N, M, NUMB2, HSUM1, HSUM2, K)
170       COMMON A(15, 2), NZ(2), ICODE(15)
```

（図表３－３－４）（その３）

電子計算機複式簿記のプログラム　――試算表作成まで――

```
171       COMMON DCA(3, 2), IXY(3, 2, 2), CODE(3, 2)
172       LOGICAL CODE, NZ
173       WRITE(6, 300) NUMB2
174       GO TO(11, 12), K
175    11 WRITE(6, 101)
176       IF((N. EQ. 1). AND. (M. EQ. 1). AND. (.NOT. NZ(1)).
                                              AND. (.NOT. NZ(2))) GO TO 2
177     1 IF(N. EQ. 0) WRITE(6, 102)
200       IF(N. NE. 0) WRITE(6, 103) (DCA(I, 1), IXY(I, 1, 2), I=1, N)
201       IF(N. NE. 0) WRITE(6, 104) HSUM1
202       IF(M. EQ. 0) WRITE(6, 105)
203       IF(M. NE. 0) WRITE(6, 106) (DCA(I, 2), IXY(I, 2, 2), I=1, M)
204       IF(M. NE. 0) WRITE(6, 107) HSUM2
205       RETURN
206     2 IF(.NOT. ((HSUM1. EQ. 0. 0). AND. (HSUM2. EQ. 0. 0))) GO TO 1
207       WRITE(6, 301) (IXY(1, J, 2), J=1, 2)
210       RETURN
211    12 WRITE(6, 201)
212       IF(N. EQ. 0) WRITE(6, 102)
213       IF(N. NE. 0) WRITE(6, 202) (DCA(I, 1), IXY(I, 1, 2), CODE(I, 1), I=1, N)
214       IF(M. EQ. 0) WRITE(6, 105)
215       IF(M. NE. 0) WRITE(6, 203) (DCA(I, 2), IXY(I, 2, 2), CODE(I, 2), I=1, M)
216       RETURN
217   300 FORMAT(1H0, 11HSHIWAKE NO., 2X, I3)
220   101 FORMAT(1H , 19HSHIWAKE NO MACHIGAI)
221   102 FORMAT(1H , 10X, 18HKARI KATA DATURAKU)
222   103 FORMAT(1H , 15X, F7.0, 2X, A2, 2X, 2H99)
223   104 FORMAT(1H , 10X, 16HKARI KATA GOKEI, F7.0)
224   105 FORMAT(1H , 10X, 19HKASHI KATA DATURAKU)
225   106 FORMAT(1H , 15X, F7.0, 2X, 2H99, 2X, A2)
226   107 FORMAT(1H , 10X, 16HKASHI KATA GOKEI, F7.0)
227   301 FORMAT(1H0, 6X, 18HTANJUN TORIHIKI NO/ , 7X,
                                          16HKINGAKU DATURAKU, 2X, A2
230       1, 2X, A2)
231   201 FORMAT(1H , 22HKANJO CODE NO MACHIGAI)
232   202 FORMAT(1H , 10X, 8HKARIKATA/(1H , 15X, F7.0, 2X, A2, 3X, L1))
233   203 FORMAT(1H , 10X, 10HKASHI KATA/(1H , 15X, F7.0, 2X, A2, 3X, L1))
234       END
235       END OF SOURCE*
```

(図表３－３－４)　(その４・完)

電子計算機複式簿記のプログラム　――試算表作成まで――

```
GENKIN  URIKAKEBIHIN          KAIKAKE      SHIHON  URIAGE
SHIIRE  KYURYO  EIGYOHI                            TOTAL
A1      A2      A3            L1           C1      R1      E1      E2
E3
99
 95000A199   1
 40000A399   1
13500099C1   1
 80000E199   2
 4000099A1   2
 4000099L1   2
 25000A199   3
 20000A299   3
 4500099R1   3
 40000E199   4
 2000099L1   4
 2000099A1   4
 30000A199   5
 20000A299   5
 5000099R1   5
 39000A1A2   6
 58000L1A1   7
 28000A3A1   8
  5000E2A1   9
  3000E3A1  10
 95000X199  21
 40000X299  21
13500099C1  21
 88000E199  31
 4000099A1  31
 4000099L1  31
999999
```

（補足資料－１）（図表３－３－４）にしめした電子計算機複式簿記プログラムで処理するために本稿の例を修正した場合のデータ・カード

SHIWAKE NO. 21

KANJO CODE NO MACHIGAI

KARI KATA

95000 X1 F

40000 X2 F

KASHI KATA

135000 C1 T

SHIWAKE NO. 31

SHIWAKE NO MACHIGAI

88000 E1 99

KARI KATA GOKEI 88000

40000 99 A1

40000 99 L1

KASHI KATA GOKEI 80000

189000	GENKIN	154000
40000	URIKAKE	39000
68000	BIHIN	0
0		0
58000	KAIKAKE	60000
0		0
0	SHIHON	135000
0	URIAGE	95000
120000	SHIIRE	0
5000	KYURYO	0
3000	EIGYOHI	0
0		0
0		0
0		0
483000	TOTAL	483000

（補足資料－2） **前記（補足資料－1）のデータ・カードを（図表3－3－4）に示したプログラムで処理した場合のアウト・プット**