12. PAD (Problem Analysis Diagram) によるプログラムの設計および作成

日立製作所　二村良彦, 川合敏雄
Yoshihiko Futamura　Toshio Kawai
中央研究所　堀越彌, 提正義
Hisasi Horikosi　Masayoshi Tsutsumi

目　次

1. はじめに　PADは，設計，作成，検査，保守といったプログラムの全開発過程において使用する図面です．私達はPADを，Problem Analysis Diagram と，野心的な意味をこめて呼んでいます．しかし，PADは，プログラムの分析用の Program Analysis Diagram だとか，PASCALを2次元的に書いた PASCAL Diagram にすぎないという人もいます．いずれの意見も，それぞれPADの特徴の一面を表わしています．

2. PADとは　PADは，プログラムの論理を記述する木構造の図面（木図面）です．それは，任意の機種（プログラマブル電卓から大型計算機まで）に対する，どんな種類（OS, アプリケーション等）のプログラムを開発する時にも使用できます．フローチャートとそれに対応するPADの例を，図5，4および図6に示しました．PADがどんな木図面かは，それ等を参照すると理解できると思います．

PADは，HIPOとコンパチブルです（図1）．更にPADは，PDL等のシュード・コードのための論理図として使え，PAD→PDL変換は，機械的手順によりできます（図2）．PADの書き方は簡単なので．例えば図2のPADとPDLの対応を見ることで理解できます．（詳細は，文献1), 2)を参照）

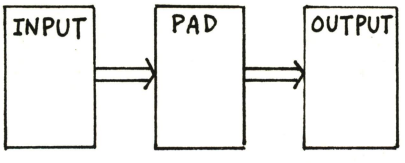
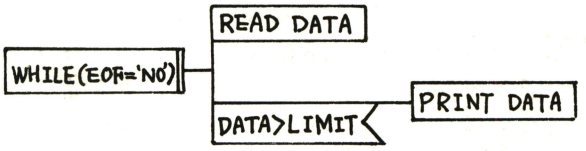


図1：PADはHIPOの処理部に書ける．




```
DO WHILE   (EOF = 'NO')
   READ DATA
   IF DATA>LIMIT
      PRINT DATA
   ENDIF
ENDDO
```

図2：PADは機械的手順によりPDL等のシュード・コードやプログラム言語に変換できる．

3. 何故構造化プログラム言語より読み易いか　「百聞は一見にしかず」という通り，人間が物事を理解する場合に，パターン認識能力が果たす役割は大きい。Dijkstra等により提唱されてきた構造化プログラミングの思想は「プログラムの論理をわかり易くスッキリと書こう。なるべく単純明快な構造を持ったプログラムを書こう」ということでした。構造化プログラミングにより確かにプログラムの論理は以前よりも読み易くなりました。しかし，論理構造が言語（即ち，言葉）で書かれている以上は，プログラムを理解する際に人間の論理的思考能力だけに頼らざるを得ない。人間のパタン認識能力を活用するためには，プログラムの論理構造を目に見えるようにする必要がある。構造化プログラミングによりスッキリとされたプログラムの論理構造を，目に見えるようにし，プログラムを理解する際に人間のパターン認識能力も活用できるようにしたものがPADです（図3）。

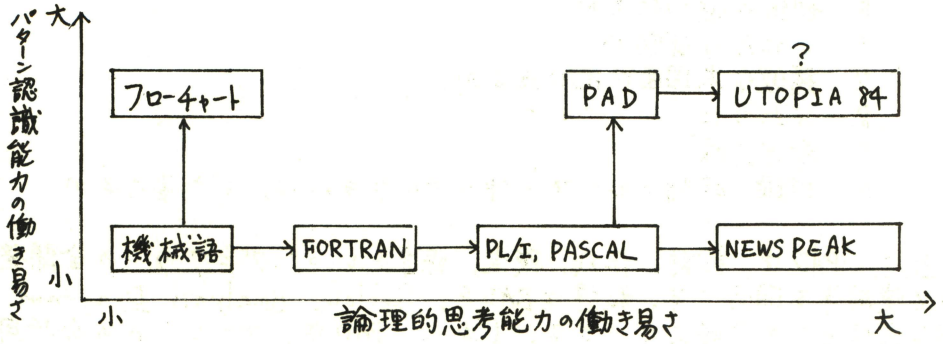


図3：　PADは人間のパターン認識能力に訴える。

図2のPADとPDLとを比較すると，図3の意味がより明らかになります。従って，PADを使ってプログラムを作成したり，PADで書かれたアルゴリズムを読むと，その論理構造は頭に入り易い。即ち，プログラムの設計やレビューが従来よりし易くなります。

4. 何故フローチャートより優れているか　PADは，プログラムの本図面ですので，フローチャートではできなかった，下記のようなことができます。

(1) プログラムの論理構造を目に見えるようにする（文献1，29頁参照）。

(2) プログラムの論理図作図上の規律を与える。

フローチャートでは，矢印と接合点を許しているため，類似の論理構造を書く場合にも，プログラマによって各人各様の記述が可能です。それに対し，PADでは，類似の論理構造を記述する場合には，どんなプログラマが書いても大体似たものになります。

例えば，私達が，PADの説明会において次のような簡単な問題を出して，受講者にPADを作成していただくと，多くの人が図4のような

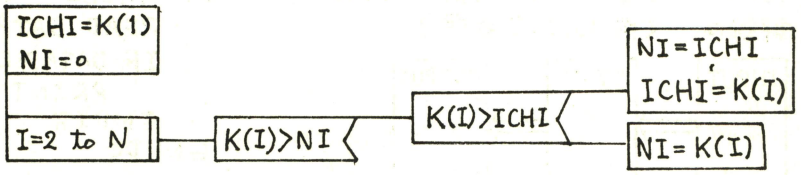


図4　ICHI-NI 問題のPAD

PADを書かれます。
[問題] 配列K(1), K(2),…,K(N)にN個の相異なる正数が入っている。このうちの最大の数(ICHI)と2番目に大きな数(NI)を求めるプログラムを作成せよ。ただし, N≧2とする。

ところが, この問題に対してフローチャートを書いていただくと, 受講者の個々の流儀により, 図5のように見かけの異なるフローが多数現われます。即ち, PADの使用により作図上の個人差を減少させることができます。

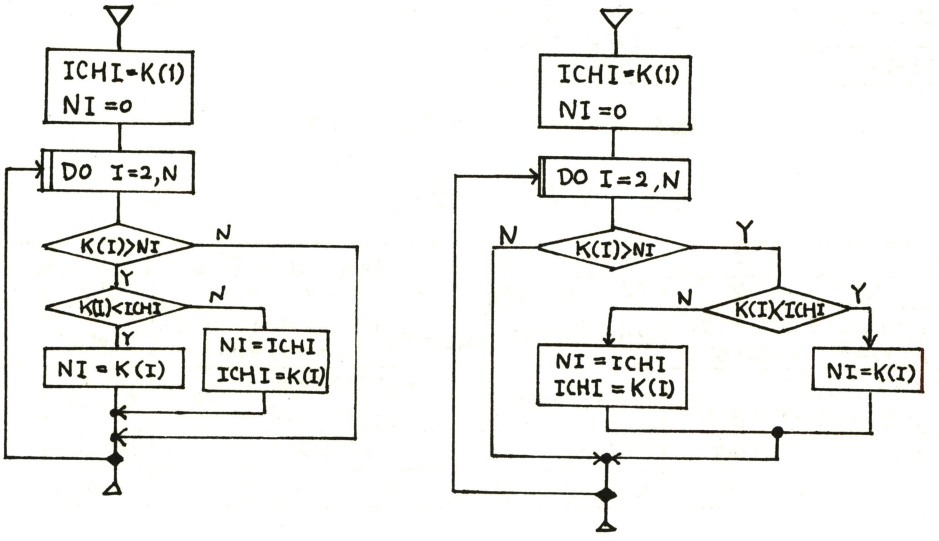


図5　ICHI-NI 問題の2つのフローチャート

(3) 初期テストの基準を明確に設定でき, 体系的な初期テストが可能になる。

PADの木の全ての葉(先端)を通るテストをすれば, プログラムの全てのステートメントは少なくとも一度はテストされます。このテストは「全ステートメント(all statements」と呼ばれるテスト基準を満足する。「全ステートメント」を満足したからといって, プログラムの正しさが保証されるわけではない。しかし, これは最低限やるべきテストです。以下に, 例題によりそのテストを体系的に行う方法を示す。

[例題] A(1), A(2),…,A(L) に入っているデータを, 与えられたKの値に従って, 小さい順(K=1のとき) 又は大きい順(K≠1のとき) に並べ替えるソートのためのPADを作成せよ。次に, PADに基づき,「全ステートメント」の基準を満足するテストデータの表(テストケース表: 表1参照)を作成せよ。

ソートプログラムのPADを図6に示した。PADの全ての葉の部分に10から10番おきに番号がふってある。ステートメントのない葉の部分(20, 40)にも番号がふってあることに注意して下さい。

このPADにふられた全ての番号に対応するステートメントブロックをテストする入力データを見つけるために, 表1のような表を作成する。まず, なるべく多くのブロックを通る入力データを決め, 表の項番1の入力データの欄に記入する。次に, その入力に対して期待される出力データを表に記入する。それから, 入力

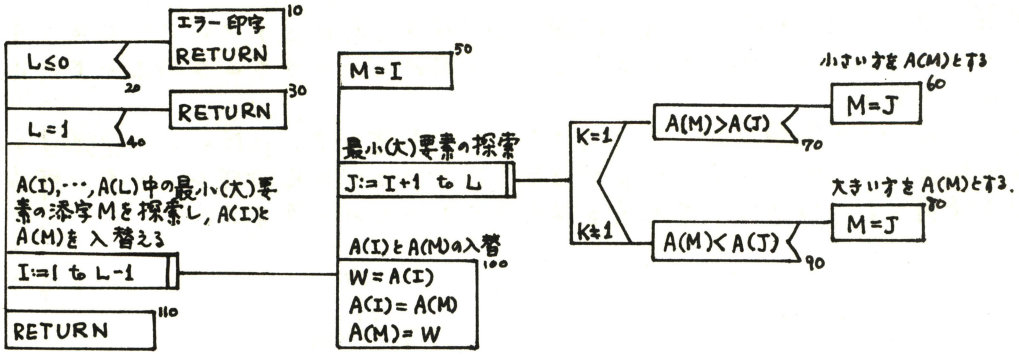


図6　ソートプログラムのPAD

表1　テストケース表

項番	テストケース		検査されるブロック	未検査ブロック	検査完了日	
	入力データ	期待される出力データ			机上	マシン上
1	K=1; L=3; A(1)=7; A(2)=9; A(3)=5	A(1)=5; A(2)=7; A(3)=9	20, 40, 50, 60, 70, 100, 110	10, 30, 80, 90	54.06.25	
2	K=2, L=3 A(1)～A(3)は同上	A(1)=9; A(2)=7, A(3)=5	20, 40, 50, 80, 90, 100, 110	10, 30	54.06.25	
3	K=1, L=1; A(1)=7	A(1)=7	20, 30	10	54.06.25	
4	K=1, L=0	エラーメッセージ	10	なし	54.06.25	

データに対してPADをトレースし(即ち，机上チェックをし)，通ったブロックの番号を「検査されるブロック」欄に記入する。残りのブロックの番号は「未検査ブロック」欄に記入する。次には，なるべく多くの未検査ブロックを通るような入力データを見つけ，同様の作業を行ない，未検査ブロックが無くなるまでこの作業を繰返す。上例の場合には，4つの入力データについてテストをすれば，「全ステートメント」のテストができることが，表1よりわかる。

「全ステートメント」は，あくまでも初期テストであるので，プログラムに要求される信頼性に従って，追加のテストを行う必要がある。

(4)機械的手順によりコーディングが可能になる。

文献1の第27頁に述べたツリーウォーク(tree walk)による方法で，PADからソースプログラムへの変換は機械的に行なえる。これにより，PADが作成された後では，コーディングの作業は無に近くなる。更に，PADとソースプログラムとの間には1-1の対応が付く。デバッグは，PADを見て行ない，PADを修正した部分は機械的手順によりソースプログラムを修正する。これにより，PADとソースプログラムとを同時に保守することが可能となる。将来，PADをソースプログラムに自動変換するPADコンパイラができれば，プログラマはPADを作り，PADをデバッグすれば良い。そうなれば，「論理図とソースプログラムの乖離」と言った，現在のフローチャートやHIPOに見られる不都合は解消できる。

5. 如何に使われるか　(図7参照)

6. どれだけ有効か　新しい物が世に受け入れられるためには「フィーリングの

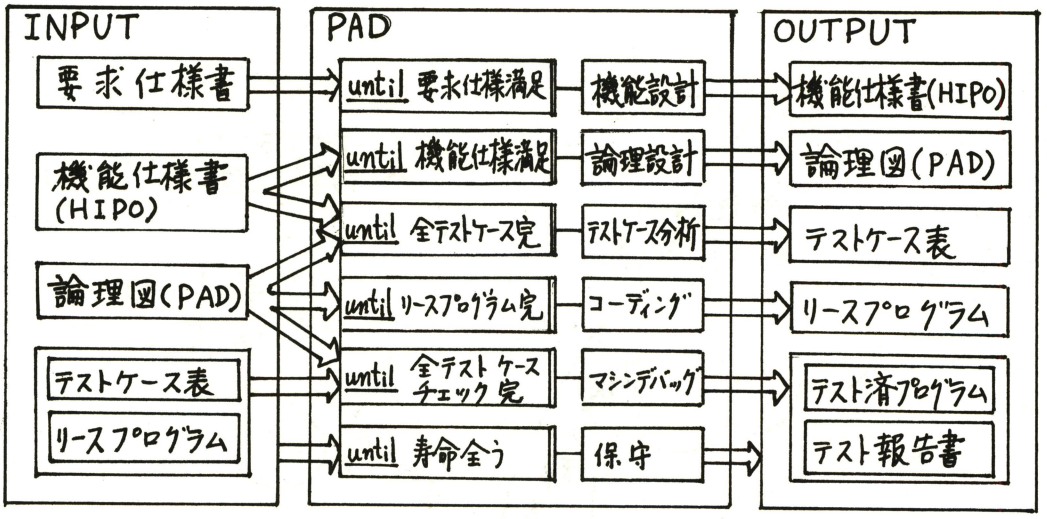


図7: プログラム開発過程におけるPADの位置付け

良さ」が大切です。そこで，まず，「PADを使ってプログラムを作成すると，それ以前と比べてどれだけプログラムの生産性が向上したと思うか」という「PADによるプログラム生産性向上感」を調べた。実務で使用するプログラムをPADを用いて開発している6チーム(12名)を選び，2週間に一度「生産性向上感」を報告していただいた。作成したプログラムの種類は，OSのスケジューラー，テキストエディタ，図形処理プログラム，数値計算用ライブラリ等の一部です。使用言語は，PL/I, FORTRAN, アセンブラ等)。表2には，プログラム開発の各フェーズにおける生産性向上感をまとめた(3ヶ月分)。表2中，1というのは，在来手法とほとんど同じという意味です。在来手法と比べてPADの方が生産性が下がったと思えば1より小さく，良くなったと思えば1より大きな数を回答していただいた。従って，「3」という数字は「今までより3倍ぐらい生産性が向上したと感じた」ということを意味する。

表2　PADによる生産性向上感の評価

開発フェーズ	評価		
	平均	最高	最低
機能設計	1.5	2	1
論理設計	1.6	3	1
テストケース分析	1.9	2.5	1.1
コーディング	2.0	3	1.3
デバッグ	2.6	3	1
修正・変更	1.4	2	1

表2の数値は，あくまでもPADユーザーの「フィーリング」を示すものです。実際に生産性がそれだけ向上したことは意味しないが，最低の評価でも「在来手法と同じ程度」ということは，PADの質の良さを示すものと考えます。表2のデータの母集団は大きくないが，PAD講習会の受講生やその他のPADユーザーの感想は，大体表2の数値と合っています。

7. 類似の木図面は他にあるか　PADと類似の目的で開発された木図面は，日本では，1978年7月の情報処理全国大会やその他で発表されています。[3,4,5] 外国ではFeretl Chart [6] やWarnier-Orr Diagram [7] が知られています。

8. おわりに　私達はPADを開発し，普及を図ってきた。これまでに，ソフト開発の辛苦を実際に体験された多くの方に，PADの良さを深く御理解いただいてい

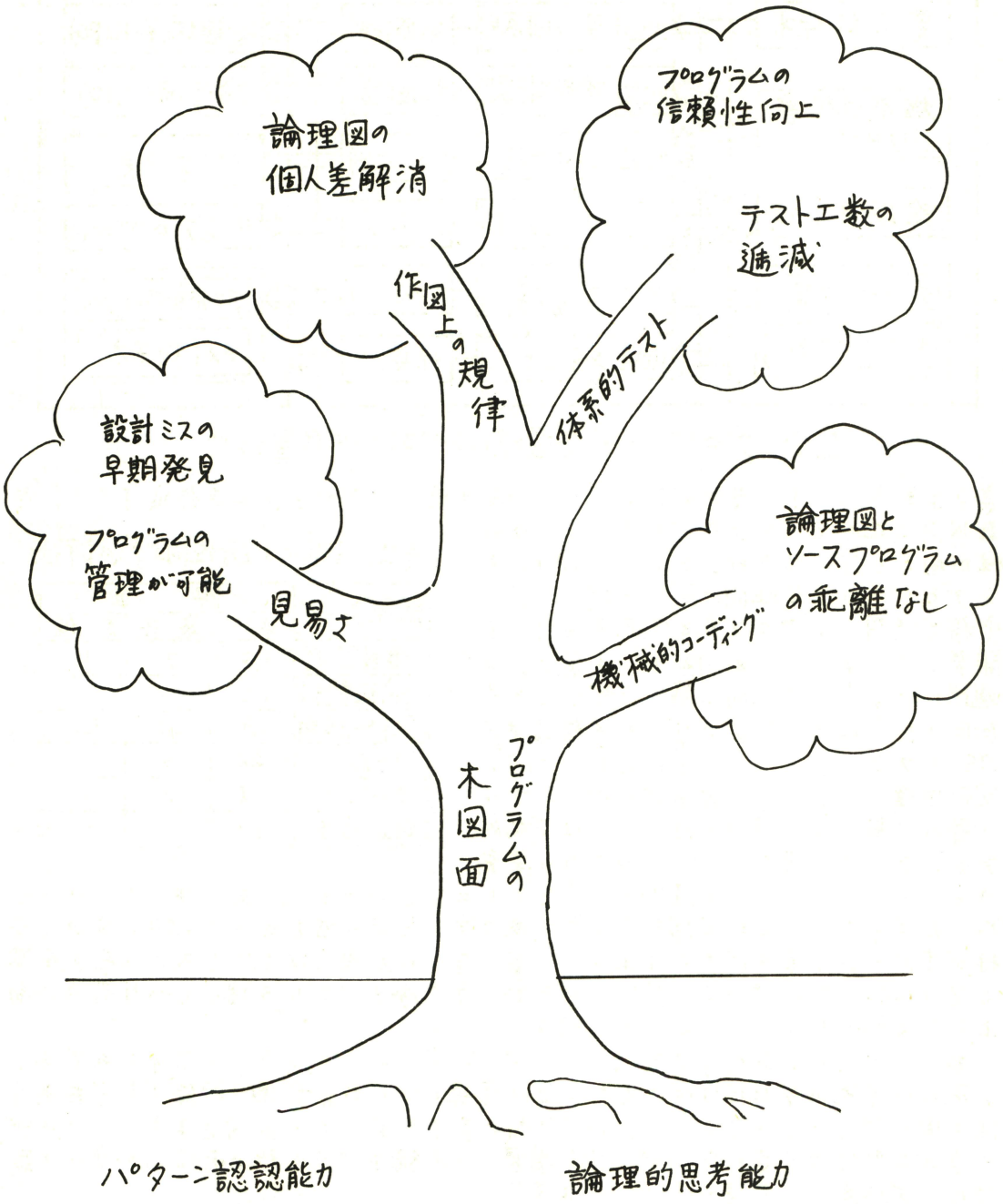


図8：プログラムの木図面の効果

ます。プログラム作成法に関する新しい技術は文献8にあるように，なかなか受入れられないのが普通です。その割には，PADは受入れられるスピードが速いように感じています。国の内外で多くの研究者が，プログラム論理図の木構造化を研究されてますので，プログラムの木図面が広く普及する日は遠くないと思っています（図8）。

PADの開発と普及には日立中研の津田順司主任研究員，小沢時典主任研究員，桑原裕計算センター長，高橋栄主任研究員および日立京浜工学専門学院の藤田勝彦助教授他，多くの方々が心から協力して下さった。これ等の方々の御理解と御協力がなければ，PADのような新しいプログラム作成技法が日の目を見る機会はなかったと思います。

9. 参考文献

1) 二村，川合，"PADによるプログラムの論理設計"，学習コンピュータ12月号，54年11月，PP24-29

2) 二村，"問題分析図（PAD）によるプログラムの改良"，電子通信学会総合全国大会予稿，昭和55年3月

3) 夜久，二木，"フローチャートの木構造型記法" 電信学会AL研究会，AL78-47，1978

4) 大原，野島，前田，"フローチャートの木構造化の一提案"，電子通信学会総合全国大会予稿1499，昭和54年3月.

5) 佐藤，浅見，"フローチャートの階層的表現のための一提案"，情報処理学会第20回全国大会 2I-9，昭和54年

6) O. Ferstl，"Flowcharting by Stepwise Retinement"，SIGPLAN Notices Vol 13，No.1，January 1978，PP34-42

7) Peter A. Verdegraal，Alan Steward Goodman，"The Warnier-Orr Diagram"，Digest of Papers，COMPCON 79，IEEE Catalog No.79，CH1393-BC，pp301-306

8) 宮本，"プログラム生産の場からソフトウエア工学に望む事柄"，電気四学会連合大会，講演論文集，昭和54年10月

（付録は次頁）

10. 付録 複雑なループを持つプログラムは，どう書けるか．

島内教授の「システムプログラムの実際」（サイエンス社，昭和47年）154頁の掛算プログラムを図9のようなPADで書直してみると，このプログラムの構造は，基本的には図10のPADと同じです。これにより，図9は図10のプログラムの最適化になっていることが理解できる．

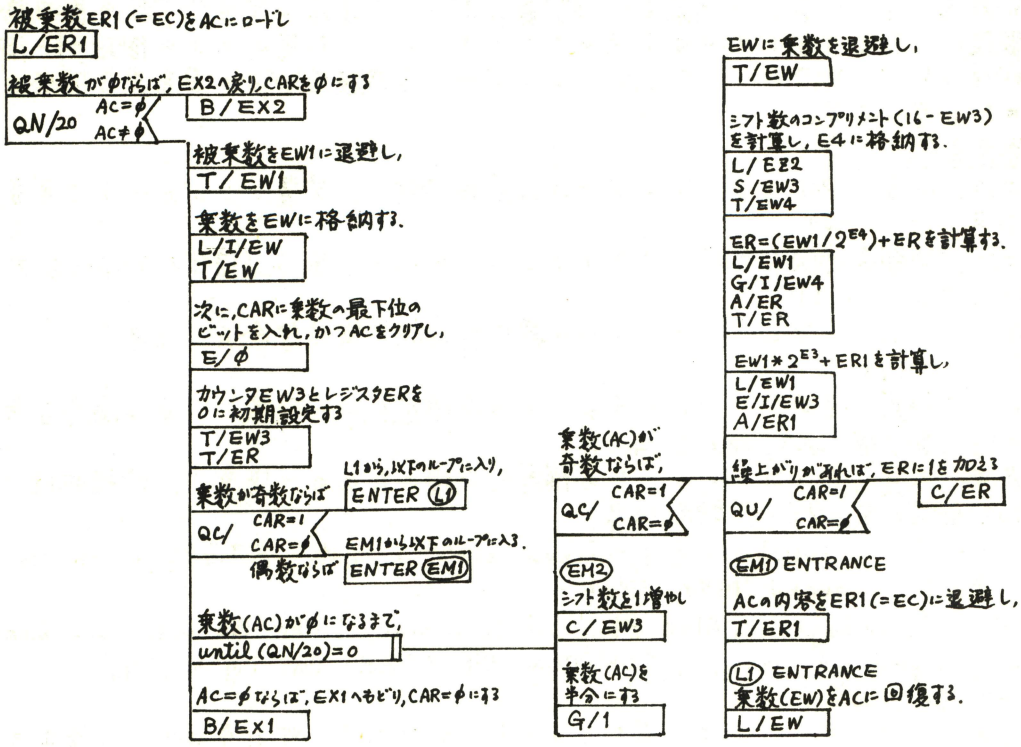


図9：島内教授の掛算プログラムのPAD化

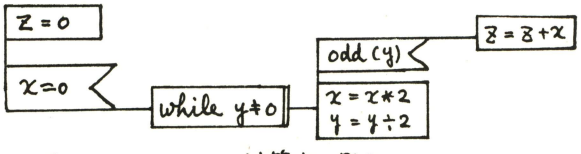


図10：z=x*yを計算するPAD