電子計算機研究専門委員会資料

パラメトロン電子計算機 PC-1 について

高橋 秀俊・後藤 英一
和田 英一・相馬　嵩
石橋 善弘・中川 圭介
(東京大学理学部)

1958年9月25日
社団法人 電気通信学会

1 PC-1 の構造について

(東京大学理学部)
高橋 秀俊
後藤 英一
相馬 嵩

§1 概説

PC-1 (Parametron Computer No.1)は, 東大理学部高橋研究室において組立てられた万能型科学計算用パラメトロン計算機であり, 総数4,200ヶのパラメトロンを用い, 昭和32年9月に製作を開始し, 昭和33年3月26日一応計算が出来るようになった. その後乗除算回路等の追加, 各部の精密調整を経て現在に至っている. パラメトロン, 励振機, 入出力装置等の資材は東洋曹達工業, 国際電信電話会社, パラメトロン研究所, 日本電子測器等より借用したものを用い, 又測定機器には電電公社電気通信研究所より借用したもの及び朝日科学奨励金により購入したものを使用した. 組立てには延約300人日の人員を要し, パラメトロン回路の配線は富士通信機でリレーの配線に経験ある工員が行い, 又磁心記憶装置の組立て及び調整には東京電気化学工業の研究所員が当った. PC-1を完成し得たのは, ひとえにこの各方面よりの御援助の賜であり, ここに厚く御礼申上げる.

PC-1に使用したパラメトロンは現在では旧型に属し, 製作時期が異り, 特性が不揃いなものをよせ集め, その上大学の研究室の手細工でその回路の全部を組立てた等の悪条件にもかかわらず, 現在では, 真空管を使用する励振機と入出力装置以外には, 故障は皆無と云ってよい. 特に交流方式磁心記憶装置の安定で, かつ信頼出来る事は特筆に値し, その一つの理由としては記憶装置に41本と云う比較的多数の真空管を使用してはいるが, その主要な部分は真空管が1本切断しても支障なく動作すると云う誤りの訂正出来る符号を応用した回路方式を採用した点があげられる.

PC-1の性能は別表に見られる通り, 内部2進法, 並列型固定小数点の計算機で, 現在日本では最高速の計算機であり, 又我が国の大学では唯一の計算の出来る電子計算機でもある.

PC-1の運転は現在1日9～12時間であり, その中5時間は理学部間の各種の数値計算に使用し, 残りの時間を我々の研究室でプログラムの研究等に使用している. 又10月からは1週10時間位を学生の実習用にあてる予定である.

パラメトロンを使用する事の他にPC-1の特徴としては次の諸点があげられる. まず, PC-1の高速記憶には, 交流方式磁心記憶装置を採用した. この方式は当研究室の創案になり, 東京電気化学工業(TDK)の手によって本方式に適する優れた磁性材料が見出された事及び誤りの訂正出来る符号を応用した選択回路の利用により, 安定度及び信頼度が極めてよい装置を作るのに成功した. この装置の一つの特長としては, 比較的小型である点も見逃せない. この記憶装置に関しては32年秋の電子計算機の記憶装置に関するシンポジウム及び本年5月, 電気四学会連合大会にも報告した.

次にPC-1の演算回路には桁上げ数字を分離して取扱う数値表示方式を採用し, 又制御回路には, 現在行われている計算と次に行うべき計算とに関する制御を同時に行う制御方式を採用した. これによって, パラメトロン計算機の計算速度はこれらの方式を採用しなかった場合の2乃至3倍に向上し, しかもパラメトロンの所要個数はあまり変りないのである. 又, これらの桁上げ分離演算及び同時制御方式は最近外国でも計算速度向上の為に種々研究されていて, パラメトロンに限らず, 如何なる種類の素子を用いる電子計算機にも, 殆んどそのまま適用する事が出来るのである.

我々は現在, PC-1の経験と成果に基づく, 更に高性能な機械PC-2の製作を計画中である.

PC-2の演算及び記憶方式はPC-1と本質的な差異はないが, 第1表の様に記憶容量を増し, 高速入出力装置を設け, 浮動小数点計算装置とアドレス変更装置を附加する等の改良を行いたいと考えている.

第1表　　PC-1 および PC-2 の性能比較

	PC-1 (昭和33年7月25日現在)	PC-2 (予定)
パラメトロン	4200ケ	約8000ケ
数　値	1語2進36桁	1語2進48桁
命　令	1命令2進18桁 1アドレス	1命令2進24桁 1アドレス
計算速度		
加減算	400μs	160μs
乗　算	4.4ms	1ms
除　算	16ms	8ms
浮動小数点計算装置	なし	有
アドレス変更装置	なし	有
高速記憶装置		
交流方式磁心記憶装置	256語	1024語
入出力装置	印刷電信機器	印刷電信機器 光電リーダ磁気テープ

		等の増設可能
消費電力	3KVA	約5KVA
床面積	2坪	2坪
(入出力を含まず)		

§2 構成と動作

PC-1の構成は第1図のブロックダイアグラムに示されている. 励振電源, 出力の直流増巾と記憶装置の駆動用増巾機との他は全部パラメトロンによって構成され, 手動スイッチと印刷電信機等の入力装置中の機械的接点では, パラメトロンの発振高周波(1.1MC)が直接開閉される.

mpレジスタを除外すれば, PC-1は並列方式のプログラム内蔵型計算機としては必要最小限のレジスターのみで構成されている. これは並列計算機で使用素子数を減ずるための常套手段であろう. mpレジスターは同時制御(待合せ制御またはAdvanced Control)のためにある. PC-1は単番地方式の計算機であるから, その基本的な制御機能は時間的に見れば, 番地計数器の内容n → n番地の命令と番地, I_n, m_nを読む. → m_n番地の内容〔m_n〕を読む. →〔m_n〕を演算回路へ送りI_mを実行 → 番地計数器の内容をn+1に増す‥‥‥ という操作を反復することになる. この操作を順次に行わずに同時に実行させて計算速度を向上させようというのが同時制御である.

PC-1ではいま演算回路でn番地の命令I_n (数値〔m_n〕) を実行しているとするとmpレジスターには次の命令I_{n+1}が読み出されて待っていて, 番地計数器には次の次の命令の番地n+2が出来て待っている. I_nが完了すると同時に次の命令に使用する数値〔m_{n+1}〕を読んで直ちに命令I_{n+1}の計算を開始し, I_{n+1}を実行している間に次の命令I_{n+2}をmpレジスターに用意し, 番地計数器はn+3になる. この制御法により, 加減算, 論理演算等の速い命令はパラメトロンの4変調周期(これを4τと表す)で実行可能となった. この方法は, アキュムレータの内容を記憶装置に入れる記憶命令は, これがその直後の命令を変更する場合にだけ不都合を生ずるが, 原理的にはこの特別な場合を検出して, 訂正を行う同時制御も可能である. しかしPC-1では回路を簡易化する為に記憶命令の場合には記憶が終ってから次の命令を読むようにした. この為8τを要する. この同時制御法の採用により, 乗除算以外の"速い"命令は, これを採用しない場合よりも2乃至3倍速くなったと思われるが一方これを行う為に 増加したパラメトロンの個数は200個位であって全体の個数の5%に満たないのである. PC-2ではこの同時制御を更に高度化し, 演算装置で乗, 除, 開平, 浮動小数点演算の様に時間のかかる命令を実行している間にアドレス変更用Bレジスターを利用しプログラムの反復回数, サブルーチンの呼出し等の操作を独立に実行させる予定であり, この為には, PC-1のmpレジスターに相当する待合せ用レジスターを少くとも2組設ける必要がある.

§3 2進法並列演算回路

PC-1の演算回路は, 第1図に見られる様にアキュムレータAccと2個の補助レジスタ, Mと Rから成り, 各レジスタはそれぞれ36bitの容量になっている. これらの演算レジスタ の各桁は第4図の様な構成になっている. ここでI相はレジスタへの入力選択ゲート, II相は中間段階, III相は出力を選出する. 図中ではII相に入れる定数入力と, Ｉ相に 入れるゲート信号の結線は省略したが, 実際には, ゲートに情報を通さない場合には ++または--の2倍強度の定数がゲート信号として与えられ, 又情報を通すゲートには 打消し合う2つの定数+-のゲート信号が印加される. Mレジスタには自己保持(循環記憶) の外, Rレジスタ又はAccレジスタ, 又は記憶装置(mpレジスタ)からの情報を入れる

事が 出来る. Rレジスタには自己保持をMレジスタからの情報を入れる機能の他に右に1桁 または左に1桁シフトすることができ, このシフトはAccと連続して71桁の2進数としても 実行することも可能である. Accレジスタは3数A', S', C'の和を部分和A^* と桁上げCの2数の和すなわち, A'+S'+C'=A^*+2C とする2進加算回路と 入力の3数A', S', C'を選択するゲート(相I)とより成る. Accの各桁の出力 A^*は3個のパラメトロンS, A, Cの合成であるが加算回路の性質上この 三者がS=A≠Cとなっている3倍強度の出力を選出することは決して起らないのでA^* はあたかも1個のパラメトロンの出力の如くに考えてよいのである. ただし このためにA^*に関係するゲートA_r, A_I, A_s, MAは5入力パラメトロンとなる. このAccでは桁上げ数字CはゲートC_r により1τに1桁しか進まないのでこのままでは桁上げを処理するのに36τを要すること になるが, 後に説明する高速桁上げ回路により3τで如何なる場合にも桁上げを処理 することが出来るようになっている. しかし桁上げの処理は, Accの結果を記憶装置へ いまうというような場合にだけ必要なのであり, Acc中に中間結果を入れておくには 桁上げ数字をそのまま残しておいてA^*+2Cの形でAccの内容を表示するように してもよい.

PC-1ではAccの内容をこの2数の和の形で表示したままで右または左へ1桁Shiftも 実行できるように桁上げCにもシフト用のゲートを設けた. 原理的にはAccの内容を 記憶装置にしまう命令が来るまでこの和表示A^*+2Cを継続するようにすれば 加減算命令は1τで実行可能となるのであるが, これでは制御回路が複雑になるので, PC-1では乗除算の中間結果にのみ和表示A^*+2Cを用い, 各演算命令の 完了前に必ず桁上げを処理している.

第5図は高速桁上げ回路の説明図である. この回路はa)が基礎となる. a)の回路は (x_2, y_2)が(0, 0)または(1, 1)ならば(z, z')は(0, 0)または (1, 1)となる. すなわち(x_2, y_2)では次の桁への桁上げが 定まる場合には(z, z')は(x_2, y_2)で定まる. 次に (x_2, y_2)が(1, 0)または(0, 1)ならば(z, z')は(x_1, y_1)で定まる. すなわち(x_2, y_2) では次の桁への桁上げが定まらない場合にはその前の桁から定まる. この基本回路 a)を組合わせると3入力のパラメトロンn段の回路で2^n桁の桁上げまで一斉に 作ることができ, b)は8桁の例である. 2^6=64> 36であるからPC-1のでは 6段すなわち2τの遅れで桁上げを一斉に得られ, この高速桁上げ回路には約300個の パラメトロンが使用されている. この回路によりPC-1では桁上げを完全に処理する 加減算命令が4τで実行可能となった. すなわち1τ目でGate SM(減算では1の補数を SMから入れ最終の桁へ桁上げ1を補って真補数にする)により和表示の結果 A^*+2Cを得, 2τと3τ目は高速桁上げ回路中の後れを待ち4τ目に桁上げ ゲートC_a により一斉に処理するのである.

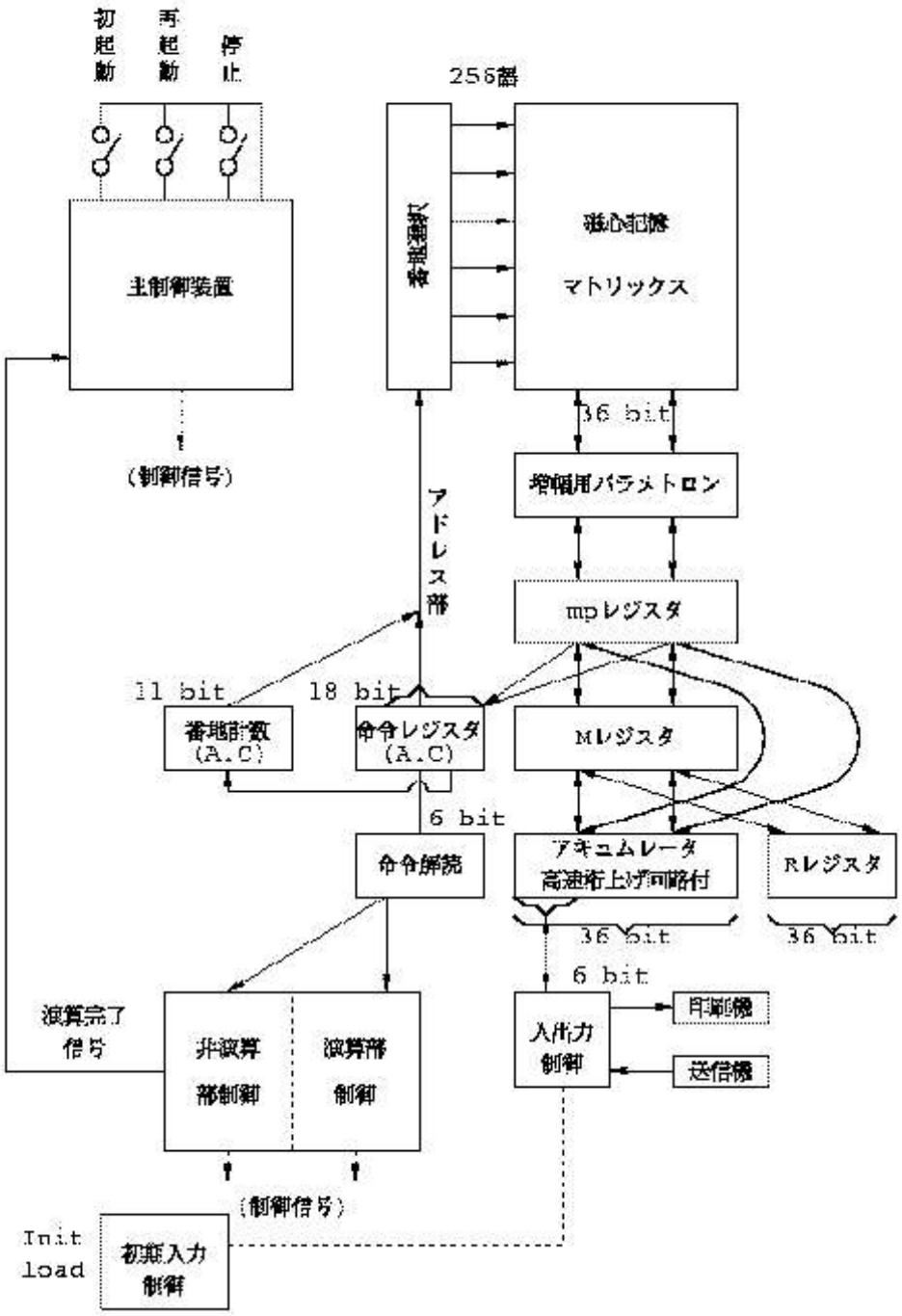
乗算は被乗数をMに入れ, 乗数をRに入れ, しかる後にAcc和表示A^*+2Cで 表示される部分値とRとを同時に右にシフトしながらMの内容を加算するか零を加算 するかして1桁を1τ, 36桁を36τで乗算し, 1τで乗数の補正用加減算を行い しかる後3τで桁上げを完全に処理する. 乗算中のAccの機能は, 梯子型高速乗算器 と同一であるが, 桁上げ処理が速く出来る点が著しい特徴である. 上の説明中の 40τの他に最初にレジスタの内容を入れ換える時間も必要なためPC-1では36桁の 乗算命令は44τを要する.

除算にはNon-restoring法を用い, 途中の結果には和表示A^*+2Cを もちい最後の剰余が正になるように補正を行っている. この補正は整数論的な計算や 多重精度の計算の便のためである. 除算は

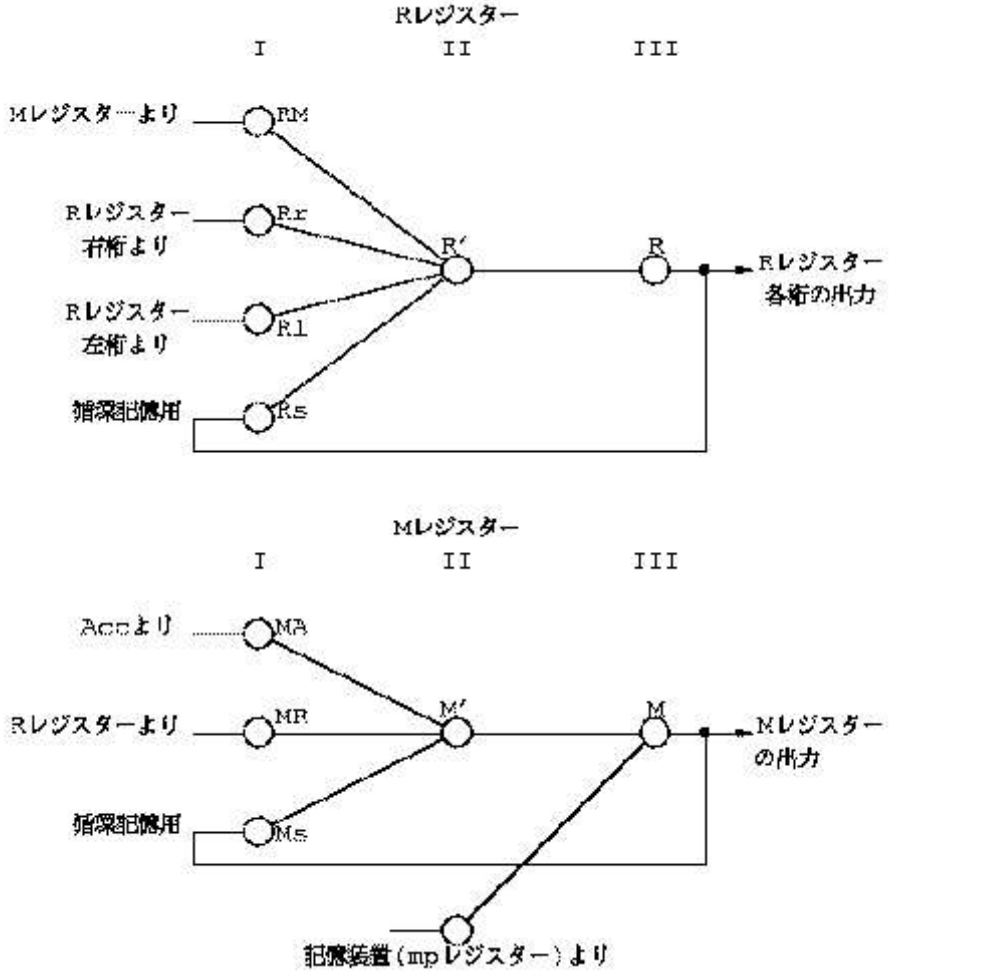
Accの符号を判定する必要があるために1桁 当たり4τを要し, またこの判定はAccの最上桁への桁上り数字を高速桁上げ回路で 調べて行っている.

第2図のAccは更に論理計算(命令BとC)にも使用され, Bは桁上げを止めることにより1τ で, またCは桁上げ入力が0の加算を行い, Cに現れた出力をC_sに戻し ゲートによりAとSを同時に0にするという操作により2τで実行される. 更に第4図 の3個のレジスタにシフトの機能があるかまたはRレジスタに2桁一挙にシフトする ゲートがあれば除算と同一時間で開平が出来るのであって, PC-2でh後の方法を採用 する予定である.

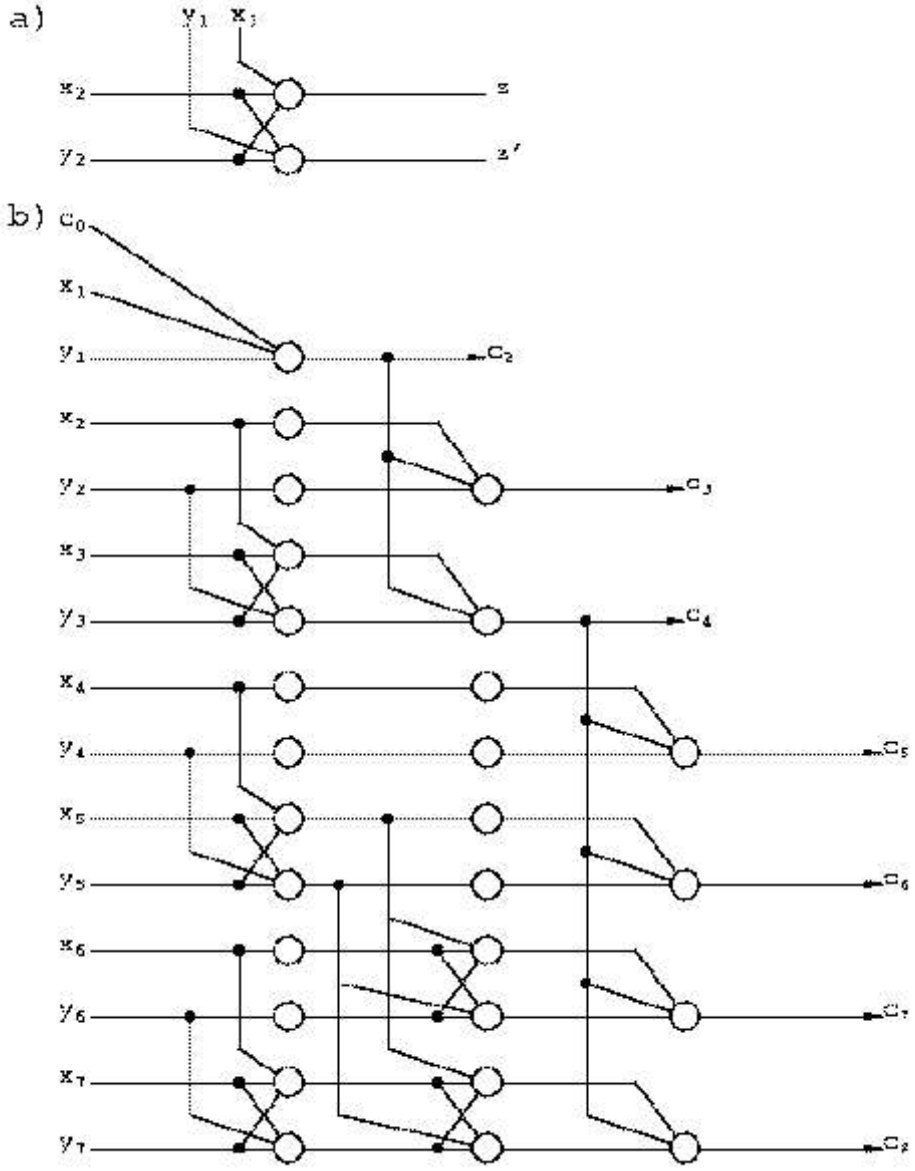
以上述べたようにPC-1のAccは3入力加算器と入力ゲートを組合せたものであり, ゲート の操作により, 和表示の並列加減算とシフト, 論理演算, 梯子型高速乗算, 除算と 万能の機能を持っていて, しかもパラメトロンの所要個数は比較的少ない. このような 次第でPC-1のAccの構成はかなり合理的なものに思われ, PC-2にも略同様なものを 使用する予定である.


初起動
再起動
停止
256語
主制御装置
(制御信号)
アドレス部
磁心記憶
マトリックス
36 bit
増幅用パラメトロン
mpレジスタ
11 bit
18 bit
番地計数
(A.C)
命令レジスタ
(A.C)
Mレジスタ
6 bit
命令解読
アキュムレータ
高速桁上げ回路付
Rレジスタ
36 bit
36 bit
6 bit
演算完了
信号
非演算
部制御
演算部
制御
入出力
制御
印刷機
送信機
(制御信号)
Init
load
初期入力
制御

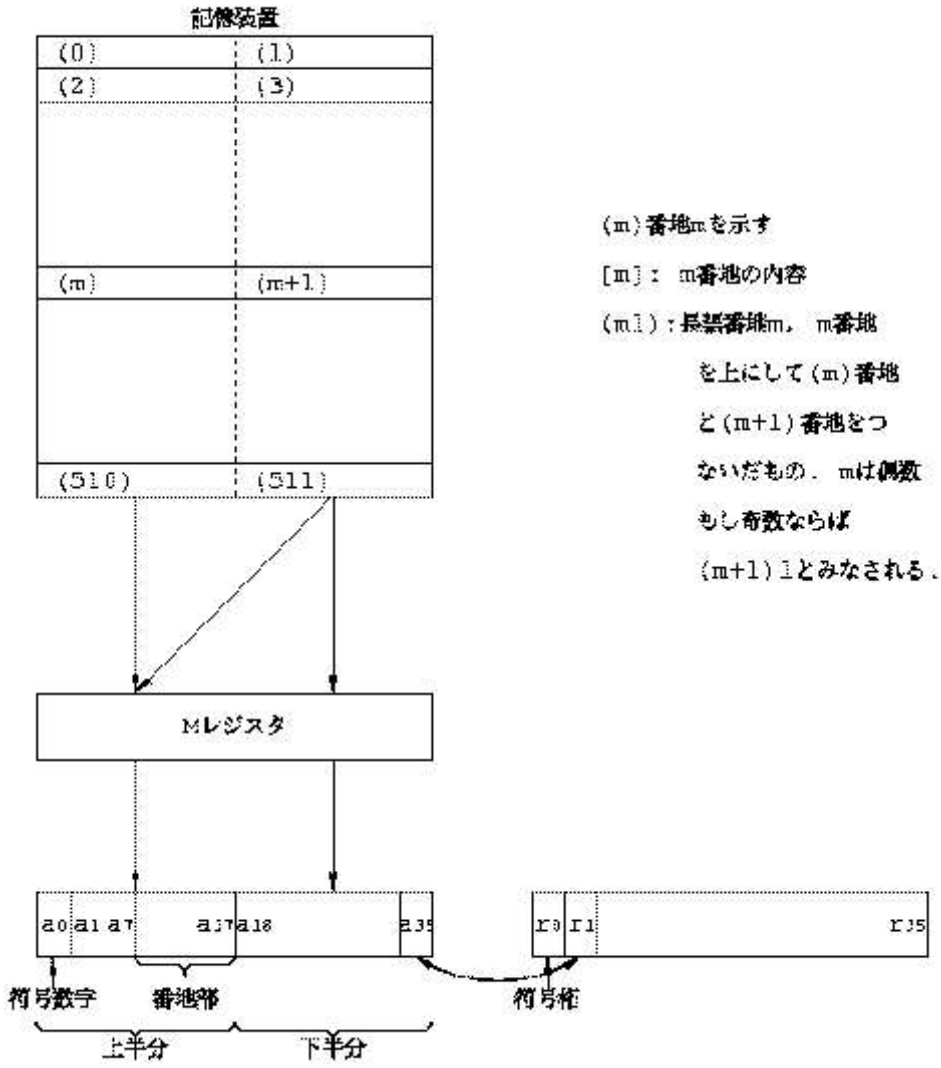
第1図


Rレジスター
I
II
III
Mレジスターより
RM
Rレジスター
右桁より
Rr
Rレジスター
左桁より
Rl
循環記憶用
Rs
R'
R
Rレジスター
各桁の出力
Mレジスター
I
II
III
Accより
MA
Rレジスターより
MR
循環記憶用
Ms
M'
M
Mレジスター
の出力
記憶装置(mpレジスター)より
Accレジスター
I
II
III
Mレジスターより
SM
Mレジスターより
(1の補数)
Accの右桁より
Ar
Accの左桁より
Al
循環記憶用
As
高速桁上げ回路より
Ca
1桁右よりの桁上げ
Cr
2桁右よりの桁上げ
Crr
Cs
S'
S
A'
A
A*
Accの出力
C'
C
桁上げ出力

第4図



第5図



第2図 レジスターの構造

第　2　表

A(Addition) 011000 (0.4msec)

al m, (ml)番地の内容をアキュムレタに加え込む.

a m, (m)番地の内容をアキュムレタの上半分に加え込む. アキュムレタの下半分は変わらない.

[注意]アキュムレタがあふれた時は2を法として正しい.

B(Binary addition) 010011 (0.4msec)

bl m, 桁上げをしないで(ml)番地の内容とアキュムレタの内容の相対応する桁について2を法とする加算を行い, 結果をアキュムレタに置く. 即ち対応する桁が一致しない桁には1を, 一致する桁には0を置く.

b m, (m)番地の内容とアキュムレタの上半分について上と同じ事を行う. アキュムレタの下半分は不変.

C(Collate) 001110 (0.4msec)

cl m, (ml)番地の内容とアキュムレタの内容とをコレートして結果をアキュムレタに置く. 即ち対応する桁が両方とも1の桁には1を, 他の場合は0を置く.

c m, (m)番地の内容とアキュムレタの上半分について上と同じ事を行う. アキュムレタの下半分は払われる.

D(Division) 010010 (161msec)

dl m, アキュムレタとRレジスタから成る長レジスタの内容を(ml)番地の内容で割り, 商をアキュムレタに, 余りをRレジスタに入れる. 余りは除数及び被除数の符号にかかわらず常に正(零を含む)になる. 故に割切れる場合には正しい答が出る. 即ち

$C(Acc)+2^{-35}\bullet C(R)=C(mL)\times C'(Acc)+2^{-35}\bullet C'(R)$

*C(")*及び*C'(")*は*(&qut)*の古い内容及び新しい内容を表わす.

除数と被除数の大きさは商が-1と1-2^{-35}の間に入るようになっていなければならない. そうでなければ"割算停止ランプ"がついて機械は止る.

[注]:0÷0で機械が止る.

-1÷(-1)は止らないけれども答は-1になる.

dm, 長レジスタの内容を(m)番地の内容で割る. 他のことはdl mと同じ.

I(Input) 001100 (0.4msec; 飛躍した場合 1.0msec 実行)

il m, 使用せず.

i m, 並列送信機用入力レジスタの内容をアキュムレタの上の6桁に移し, アキュムレタの他の桁は払われる. そして送信機のテープを一段進めて入力レジスタに読んだ文字を入れる. もし入力レジスタが空ならば命令は(m)番地へ飛ぶ.

J(Jump) 011010 (0.4msec)

jl m, (無条件飛躍) (m)番地の命令を次に行う.

j m, (サブルーチン飛躍) 未完成

K(Kconditional jump) 011110 (0.8msec)

kl m, (非負数飛躍) もしアキュムレタの内容が負でなければ次の命令は(m)番地から始める. そうでなければ順番に行う.

k m, (負数飛躍) もしアキュムレタの内容が負ならば次の命令は(m)番地から始める. そうでなければ順番に行う.

L(Left shift) 001001, (0.1×(移動の数)+0.6msec 最高13.4msec)

ll m, もしm<1024ならばアキュムレタとRレジスタをつないだ71桁の内容を m桁左に移動する. r_0の桁は変らない. Rレジスタの下からは"0"が入ってくる.

もしm≥1024ならば, 長レジスタを(2048 - m)桁右に移動する.

此の場合はアキュムレタの一番上の桁には"0"が入る. (論理シフト)

l m, アキュムレタの内容について上と同様の移動操作を行う.

m<1024の場合はアキュムレタの下から"0"が入る.

N(Negative load) 000110 (0.4msec)

nl m, (ml)番地の内容の符号を変えたものを, アキュムレタに入れる.

n m, (m)番地の内容の符号を逆にしたものをアキュムレタの上半分に入れる. 下半分は払われる.

O(Ooutput) 000011 (0.4msec)

ol m, o m, アキュムレタの上, 6桁を出力レジスタのうつす. そしてそのコードに相当する文字をプリントする. もし出力レジスタがつまっている時は次の命令は(m)番地から行う.

P(Positive Load) 001101 (0.4msec)

pl m, (ml)番地の内容をアキュムレタに入れる.

p m, (m)番地の内容をアキュムレタの上半分にいれる. 下半分は払われる.

Q 011101 (0.4msec)

ql m, (ml)番地の内容をRレジスタに, Rレジスタの内容をアキュムレタに移す. アキュムレタにあった内容は失われる.

q m, (m)番地の内容をRレジスタの上半分にれる. 下半分は払われる. 他はql mと同じ.

R(Right shift) 0011010 (0.1×移動桁数+0.6msec 最大13.4msec)

rl m, もしm<1024ならば, アキュムレタとRレジスタから成る長レジスタの内容を右にm桁移動する. アキュムレタの最上位の桁には符号桁の内容が補われる. もしm≥1024ならばll (2048 - m)に相当する桁移動が行われる.

r m, アキュムレタの内容のみについて上と同じ操作を行う.

S(Substraction) 010100 (0.4msec)

sl m, (ml)番地の内容をアキュムレタの内容から引く.

s m, (m)番地の内容をアキュムレタの上半分から引く.

T(Transfer to the store) 000001 (0.8msec)

tl m, アキュムレタの内容を(ml)番地にしまう. アキュムレタの内容はそのまま保たれる.

t m, アキュムレタの上半分を(m)番地にしまう. アキュムレタの内容はそのまま保たれる.

V(Vervielfältigung) 001111

vl m, アキュムレタの内容と(ml)番地の内容をかけて, Rレジスタの内容(r_1～r_{35})を結果の下の35桁に加える. そしてその答を長レジスタに入れる. (44msec)

v m, アキュムレタの内容と(m)番地の内容をかけて結果の下の35桁(a_{18}～r_{17})に相当する所にRの内容を加える. そして答をa_0～r_{17}の桁に入れる. r_{18}からr_{35}の桁は払われる. (26msec)

X 010111

xl m, 使用せず.

x m, (番地格納) アキュムレタの番地部分(a_7～a_{17}を(m)番地のそれに相当する桁にしまう. (m)番地の他の桁及びアキュムレタの内容は変らない.

Y (雑命令) 010101

フリップフロップ出力命令

yl 30, K-Tレコーダ出力用フリップフロップNo 2をリセットする. (0.4msec)

y 30, 出力用フリップフロップNo 2をセットする. (0.4msec)

yl 31, K-Tレコーダ出力用フリップフロップNo 3をリセットする. (0.4msec)

y 30, 出力用フリップフロップNo 3をセットする. (0.4msec)

Z(Zero jump) 010001 (1.1msec)

zl m, (アキュムレタ零飛躍)

もしアキュムレタの内容が零ならば次の命令は(m)番地から行う. 零でなければ順番に行う.

z m, (アドレス部零飛躍)

もしアキュムレタの番地部(a_7～a_{17})が零ならば次の命令は(m)番地から行う. 零でなければそのまま順番に行う.

設 置 予 定 命 令

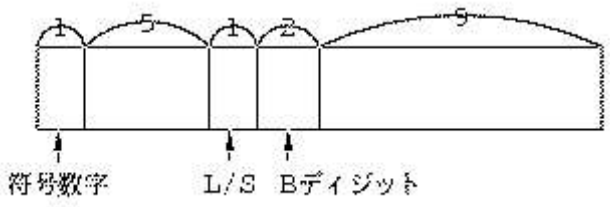
E(区切り命令) 010000 (0.4msec)

el m, もしF_L=1ならば次の命令は(m)番地から始める. そうでなければそのまま順に行う.

e m, もしF_S=1ならば次の命令は(m)番地から始める. そうでなければそのまま順に行う.

[注]F_L及びF_Sはフリップフロップで, 手動キースイッチでセットしたり, リセット出来る. スイッチが中点にある時はy-命令によってセット, リセットを行う.

B-レジスタ関係命令



F

fl m, 指定されたBレジスタに2を加えてその結果が零でなければ次の命令は(m)番地から始める. 零の場合は順に行う.

f m, 指定されたBレジスタに1を加えてその結果が零でなければ次の命令は(m)番地から始める. 零の場合は順に行う.

G

gl m, 指定されたBレジスタに数値mを入れる.

g m, 指定されたBレジスタに数値mを加える.

H

hl m, 指定されたBレジスタの内容を(m)番地の番地部にしまう. (m)番地の他の桁及びそのBレジスタの内容は変らない.

h m, (予定なし)

J

j m, 無条件に次の命令を(m+1)番地からはじめて, そのjmのあった番地に1を加えたものを(m)番地の番地部にしまう. (m)番地の他の桁及びアキュムレタの内容は変らない.

M (予定なし)

O

ol m, (m)番地の内容の上6桁を出力レジスタに移し, そのコードに相当する文字を印刷する. アキュムレタ及びRレジスタの内容は変らない.

U (アキュムレタのあふれ出し検出命令)

ul m, アキュムレタの内容を(ml)番地にしまう. もしアキュムレタが最後のPまたはN命令以来はみ出していれば機械は止る.

u m, アキュムレタの上半分を(m)番地にしまう. 他の事はul mと同じ.

Y

yl 0, アキュムレタとRレジスタから成る長レジスタをまるめて, 36桁の数にする. 即ちr_1の桁に1を加える. Rレジスタは帰零される.

y 0, アキュムレタをまるめて18桁の数にする. 即ちa_{18}の桁に1を加える. アキュムレタにはその結果がのこる.

yl 1, アキュムレタの内容とRレジスタの内容を交換する.

y 1, (予定なし)

yl 2, アキュムレタの内容の平方根をとって結果をアキュムレタに置く.

y 2, (予定なし)

yl 3, イニシャルロードのスイッチを押し右と同じ操作を行う.

y 3, (予定なし)

y(4～27)はフリップフロップセット, リセット命令.

「浮動小数点命令」演算部に関係し長語短語の区別のある命令で操作文字の次にLがあって番地が奇数の場合は浮動少数の操作とみなされる.

2 PC-1 のProgram

(東京大学理学部)
高橋 秀俊　後藤 英一
和田 英一　相馬　嵩
石橋 善弘　中川 圭介

§1 Initial Input Routine

Stored Programの計算機に, 命令や数値を入れて, 計算を始めるのに使われる手は, 相反する傾向の二つのうちのいずれかがとられる. その一つは, 充分な読み込みの 機能を計算機自身に持たせてしまうもの, もう一つは, 計算機は, 読み込みに必要な 最小限の機能しか持たず, 主としてプログラムによって読み込むものである.

わが国で最近作られた計算機では, 例えば日立製作所の計算機, HIPAC MkIは前者に 属し, また日本電気株式会社のNEAC 2201の語を読む命令にも前者の傾向が見られる. それに対して, 例えばNEAC 1101やETL Mk IVのBoot Strap方式は後者に近いものと 考えてよいであろう. 東京大学のPC-1はやはり後者に属するものであるが, Boot Strap はなく, そのかわり二進法でさん孔されたテープを0番地から順々に読み込んで, 0番地 から実行に移るInitial Loadの機能を備えている. 従って0番地から命令が始まるよう にプログラムを作り, 2進法でテープを作って, Initial Loadのキーを押せば, 計算機 に命令や数値が読み込まれ, 計算が行えるようになっている.

しかし, 実際にプログラムを作るときは, アドレスや数値を2進法に直さなければなら ないとか, 0番地から始まるようにしなければならないとかいうのでは, 使いにくいこと はこの上ない. それにサブルーチンを使う場合のアドレスの変更などの機能が Initial Loadにはないので, アドレスが変るプログラムは, いちいち打ち直さなければ ならない. Initial Loadは非常に機械的にしか読み込まないので, 読み込みを自在性を 与えるためには, 読み込み用のプログラムを作り, 計算機を使うことが絶対に有利で, Initial Loadは, その読み込み用プログラムその他特殊なプログラムを読み込むのに 用いることになる. 計算機のプログラムが, この読み込みプログラムを用いる場合, それとの関係で理解されなければならないことは, Hardwareの読み込み回路を持っ ている機械の場合と同じである. もちろん読み込みのプログラムはHardwareとち がって, 簡単に変えることもできるわけであるが, 計算機の存在意義は, その周囲の サブルーチンの集積(ライブラリー)にも大きく依存しているし, サブルーチンの 作り方がまた読み込みプログラムを考慮しているので簡単に変更することは 慎むべきである. また, 読み込みのプログラムを使うと, テレタイプのコード変換を プログラムに実行させることもできる. これは, テレタイプのコードバーを, わざわざ2進法のために切替える必要を除外し, 入出力装置に対する条件をゆるめる ことになる. PC-1では既存のテレタイプを改造せずに, そのまま用いることにし, コードの変換はすべて入出力のプログラムで行うことにあいた(コード表は後に 掲げる).

このような見地から, 高橋研究室では, 計算機の完成と同時に, 読み込みプログラム (Initial Input Routine)を作り, このプログラムと一緒になって働く各種の サブルーチンを用意した.

PC-1のInitial Input Routineは, いくつかあるが, R0が基本になっていて, これは Initial Loadで読み込まれるため, 2進法でさん孔されている. R0は, 10進法でさん 孔された命令および正の整数(短語)を読み込むことができる.

Initial Order R1は, R0によって読み込まれR0と一緒に働き, R0の機能に, 正負の 整数, 小数を長語として読み込む機能をつけ加える. *)

R0は, 命令を読み込む際, 相対番地を絶対番地に改め, また命令や数値に, Code Letterに従って, 種々のParameterを加えることができる. また読み込みが終った場合, 所定の位置からの計算を始めるように働くのもR0である.

R0は, その主要部分が消されない限り, 何回でも用いられる. それには, コントロール を0番地に戻せばよい. また計算が終了したとき, コントロールを, R0の適当な番地へ 移してやれば, 計算機はすぐにつぎの計算プログラムの読み込みにとりかかる.

R0で読み込む場合, テープの作り方は, Initial Load用に作るのにくらべて, はるかに 融通がきくが, それでもいくつかの点に注意しなければならない. それらの点を述べて からプログラムの例を示そう. (* この他R2, R3, R5などのInitial Input Routineがあるが, それらの機能につい ては, ここでは省略する) R0(およびR1)で読み込むテープの符号は, つぎの4種類に分けられる. これらの符号は すべてテレタイプにあるもので, それらの鍵を打つと, その符号がテープにさん孔 される.

1 Function letter, Control Symbolの直後に打たれたアルファベット. オーダーコ ードの演算の種類を直接表示する.

2 Numerals, 0から9までの数字. 命令中のアドレス, 整数, 小数などを, 10進法で 表示する.

3 コードレター, ファンクションレターとはみなされない(すなわちControl Symbol の直後にはない)アルファベット. 各コードレターには別表に示すように, それぞれ に対応する記憶位置があり, コードレターを持った命令には, 対応する記憶位置の 内容(Parameter)が, オーダーコートとアドレスに加えられる.

4 Control Symbol, Comma ","他のControl Symbolの直後にはない*) Period ".", colon ":" plus "+", minus "-" equal "="とコードレターの位置に ある改行符号. これらはその前にさん孔されているテープの内容が, どういう 種類のものであるかをInitial Input Routineに示す. (* 他の Control Symbol の直後にある"."は, R1では小数点とみなされる.)

, は命令および, 短い正の整数の区切りに使う. これによって命令が格納され, 格納 命令のアドレスが1つふえる.

: は, プログラムの最初の命令を入れる記憶位置を指定し, 相対アドレスの原点を 同時にセットするのに用いる. 例えば100:とすると, つぎの命令が100番地から 順に入る.

= は, variable code letterに対応するparameterをセットするのに用いる. 例えば, 0h=131072とすると, hパラメーターが131072になる.

+ 正の長語としての整数, 小数の区切りを示す.

- 負の長語としての整数, 小数の区切りを示す.

. オーダーコードと組み合わさって, 各種の機能を発揮する. 例えば, 命令の 読み込みが終り, N番地から命令を実行させたい場合に, jlN. の形で使用される. Control Symbolの直後にある "."は小数点をみなされるが, Control Symbolと"." との間に数字0があっても, この場合も小数点とみなされる.

Code letterとしての改行符号(LF) Initial Input Routineの中にWorking Space を払うのに用いられている. さん孔を間違った場合, Control Symbolならそれを 抹消符号で消し, それ以外にはただ改行符号を打てば, その前の語は消される.

復帰（CR), 間隔(SPACE), 下段(LTRS), 上段(FIGS)およびファンクションの位置の 改行(LF)はアルファベットとして読まれる. 空白(BLANK)は, R0, R1では無視される から, テープのどこに置いてもよい.

コードレターの働きは, まえにも述べたように, それに対応するparameterを加えるこ とにあるが, どれもが自由にかえられ, 自由に使えるわけではない. そこで, 特別な 意味を持ったコードレターについて示すと, 下のようになる.
lは, オーダーコードのロング, ショートを区別するビットをつけ加える常数で変数としてはならない.

記憶位置	コードレター	加えられる数	
57	t	tM	この位置は, R0の格納命令(transfer order)が 入っている. Mは現在格納すべきアドレス.
58	CR	jl 53	R0の命令の一部. 普通は用いない.
59	o	28	oは長語のworking spaceを指定する(0oL=28L)主としてサブルーチンの数値の 受け渡しの窓口となる. parameterは変更して差し支えない.
60	SPACE		
61	h	変数	
62	n		
63	m		
64	LF		(working space)コードレターの働きはない.
65	l	2048	
66	r	(R)	相対番地の原点が入る. R0によってセットされる.
67	g	68	68番地から87番地までのworking space'sを指定するparameter. 変更して差し支えない.
68～87			以上の外のコードレターのための場所で変数に使われる. 計算中はworking spaceとして用いられる.

§2 閉じたサブルーチンへのリンク

計算機で, ライブラリーを用意する場合, サブルーチンへとぶときの方式を約束する 必要がある. PC-1では, リンクのためにアキュムレーターを用いることにし, 従って 変数などは, 特定の場所に入れられるか, パラメーターで指定することになる. 主 ルーチンの101番地から, 200番地にあるサブルーチンへとぶ場合, 主ルーチンのその 部分は,

```
100   p100
101   jl200
102   サブルーチンから帰ってきてからの命令
```

という形をとる. 100番地のp100を格納するとき, 格納命令のアドレスが100になって いるから, それを用いると, プログラムを書くに当たって, 格納場所を知っている必要 はない. それにはコードレターtを用いるわけであるが, tのコードレターには, 格納命令tがファンクション部にあって, それも一緒に加えられてしまうため, pの 命令のファンクション部には, pからtだけ引いたもの, つまりiをとればよい. そこで

```
100   it
101   jl200
```

となる.

PC-1のInitial Input Routineでは, コードレターが充分に使えるので, サブルーチンの 格納場所は大ていの場合パラメーターで与えられる. 例えば, 200番地からprintの サブルーチンを格納するとき, pパラメーターに200を入れておけば, サブルーチンは 0p:として格納を始めればよく, サブルーチンへとぶ命令は, `it, jlp,` としてよい.

§3 簡単なプログラムの例

1. テープに3(101010)とy(010101)と交互にさん孔するプログラム. 印刷電信機の同期 を調整するためのもので, これを116番地から入れる. プログラムは116番地から 始まる.

```
     | 116:     |
116  | n120,    |
117  | o117,    |
118  | b121,    |
119  | jl117,   |
120  | f,       |
121  | 258048,  |
     | jl116.   |
```

テープは上のような形になる. f(120番地)はファンクション部が010110で, これの 補数は101010, つまり3である. これを印刷して, 121番地の数とbinary sumを とれば, (121番地の数は, ファンクション部が111111)アキュムレーターの ファンクション部は010101, つまりyになる. これを印刷して, 再びbinary sum をとるとまた3のコードができる. これを相対番地で表せば

```
     116:
0    n4r,
1    o1r,
2    b5r,
3    jl1r,
4    f,
5    258048,
     jlr.
```

となる. 以下の","は省略する.

2.x=0.01 (0.01) 0.10 で√xを8桁求めるプログラム
一列に 五つずつ印刷して二列に表すようにする. これには, 平方根を求める サブルーチンQと, 小数を任意の桁数だけ印刷するサブルーチンP2とを用いる. Qは, Q番地(qパラメーターで指定)から始まり, 12番地を必要とする, 閉じた サブルーチンで, 0gLの平方根を)0oLにおく(0gLは, gが68なら, 68, 69番地を 表わし, 0oLは, oが28なら28, 29番地を表わす). P2は, P番地(pパラメーターで 指定)から始まり36番地を必要とする. 0oLの数をH桁(hパラメーターで指定) 印刷するが, 数のまえに小数点を, 数のあとに符号を印刷する. nパラメーターは, サブルーチンによって使用される.

これだけわかれば, 主ルーチンはつぎのように作ればよい. (Q, P2のサブルーチン の詳細は, 後述する.)

```
    0t:          まえの命令をしまったつづきからしまう.
                 相対番地の原点をセットする.
 0  p33  26より
 1  o1r
 2  o2r
 3  l2
 4  o4r
 5  t27r 25より 4 × 復帰のコードを(カウンター)しまう.
 6  q2           Rレジスターを払う.
 7  p2g          100x を持ってくる.
 8  s3g
 9  z40          100x が10になっていたらR0へ戻る.
10  a4g
11  t2g
12  d5g          100xをxにする.
13  tlg          平方根をとる数を0gLへしまう.
14  it
15  jlq          平方根のルーチンへ.
16  plo
17  al6g         丸めの定数を加える.
18  tlo
19  p27          間隔
20  o20r
21  it
22  jlp          印刷のルーチンへ.
23  p27r
24  s33          カウンターを調べる.
25  kl5r         5回未満なら5へ.
26  jlr
27  (  ) 5による カウンター
    t2g.         相対番地のアドレスを変えずに, 2gから格納する.
70  0
71  10
72  11
73  100
74
75  .000000005+  8桁で丸める定数.
    jlr.          プログラム開始の命令.
```

テープはつぎのようにつなげる. 0p=116, 0h=8 P2の写し 0t:0q=0r, Qの写し 主ルーチン P2とQとの間にある, {¥tt 0t:0g=0r,} は, P2のつづきにQを入れ, Qパラメーターを セットする命令である.

§4 Initial Input Routine R0, R1

```
記憶位置 命令
   0  al 28  38から
   1  jl 49           10進2進変換
   2  0               定数
   3  0 ( 0) 34による Working space(アドレス部分のみ)
   4  a  64  25から   パラメーターにファンクションを加える
   5  t  64  45から   ファンクションをしまう
```

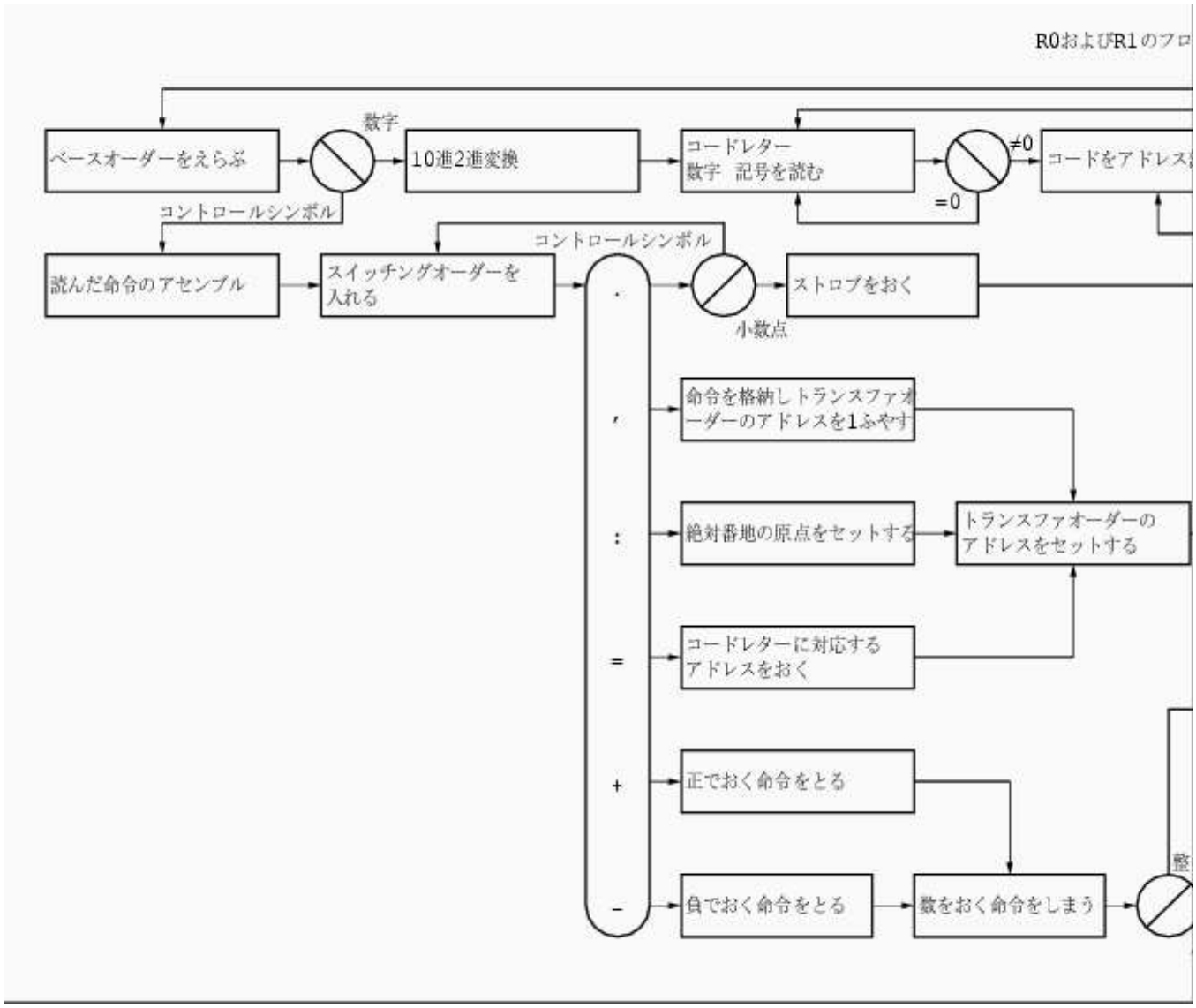
```
  6  t   6  7,52から コードレター，数字，記号を読む
  7  zl  6           BLANKなら6へ
  8  rl 12  46から   テレタイプコードをアドレス部へ移し，RLレジスターの左端を払う
  9  kl 20           コードレターならとぶ
 10  a  33          セレクションのアドレスを作る
 11  x  12          アドレスを12へ入れて，ベースオーダーをセレクトする
 12  p ( 0)
 13  kl 34          ≧0(数字)なら34へ
 14  a  47          ベースオーダー，スイッチングオーダーにする
 15  t  18  89から   オーダーを18へ入れる
 16  p  29          今読んだアドレス部と，ファンクション パラメーター部を
 17  a  64          アキュムレーターにおく．スイッチングオーダー(スイッチング
 18 (    ) 15から   オーダーは ,.:=+-を読んだときそれぞれjl57，jl88，x66，
                    p24，jl94，jl96 になる
 19  jl 55
 20  a  33  9から    セレクションアドレスを作る
 21  x  24          アドレスを24に入れる
 22  s   4
 23  z  40          コードレターがLFかどうか調べる
 24  p (  ) 21による パラメーターをアキュムレーターにおいて
 25  jl  4  (4)     4へとぶ
 26 *s  84  (-)     ベースオーダー *はd0が1であることを示す
 27  SP  8  (8)     ベースオーダー ファンクション部に間隔のコード
 28 (    ) 41，51
 29 (    ) 106による Working space
 30 *s  45  (,)     ベースオーダー
 31 *s  76  (.)        〃
 32 *s  82  (+)        〃
 33  CR 56          セレクションのアドレスと作るための定数，ファンクション部に復帰のコード
 34  x   3  13から   今読んだコードの2進数を3のアドレス部にいれる．
 35  pl 28
 36  rl  7  (7)      前の結果をアキュムレータの入れて
 37  ll  9  (9)      10進2進変換を行う
 38  jl  0
 39 *el 54          ベースオーダー
 40  0   2  (2)23,56,113から
 41  tl 28  91から   64と28,29番地を払う．またストロブを64と28番地に入れる．
 42  t  64
 43  i  43  44から   ファンクションレター，数字，記号を読む．
 44  zl 43          BLANKなら43へ
 45  kl  5  (5)      ファンクションなら5へ
 46  jl  8
 47 *nl 12          ベースオーダー
 48  r   0  (0)      10×2^-5
 49  l   1  (1)1から
 50  al  2          10進2進変換
 51  tl 28
 52  jl  6  (6)
 53  p  57  58から   トランスファーオーダーのアドレスをふやし
 54  a  49
 55  x  57  19から   トランスファーオーダーのアドレスをセットする．
 56  jl 40
 57  t (  ) (t)18から 55,110による トランスファーオーダー
 58  jl 53  (CR)
 59     28  (o)
 60 (    ) (SPACE)
 61 (    ) (h)       パラメーターの格納場所
 62 (    ) (n)
 63 (    ) (m)
 64 (    ) (LF)5,42による working space
 65   2048  (l)
 66 (    0) (r)18による  相対番地の原点
 67     68  (g)

 88   zl 90  18から
 89   jl 15
 90   p  92  88から
 91   jl 41
 92   156416 =1.193359375
 93   nl 28
 94   p 100  18から
 95   jl 9r
 96   p  93  18から
 97   t 111  95から
 98   p  64
 99   kl107
100   pl 28
101   kl105
102   v  48  104から
103   ll  5
```

```
104   k 102
105   dl114  101から
106   tl 28
107   p  57  99から
108   x 112
109   a  40
110   x  57
111  (     ) 97による
112   tl(  ) 108による
113   jl 40
114   38146
115   254976
```

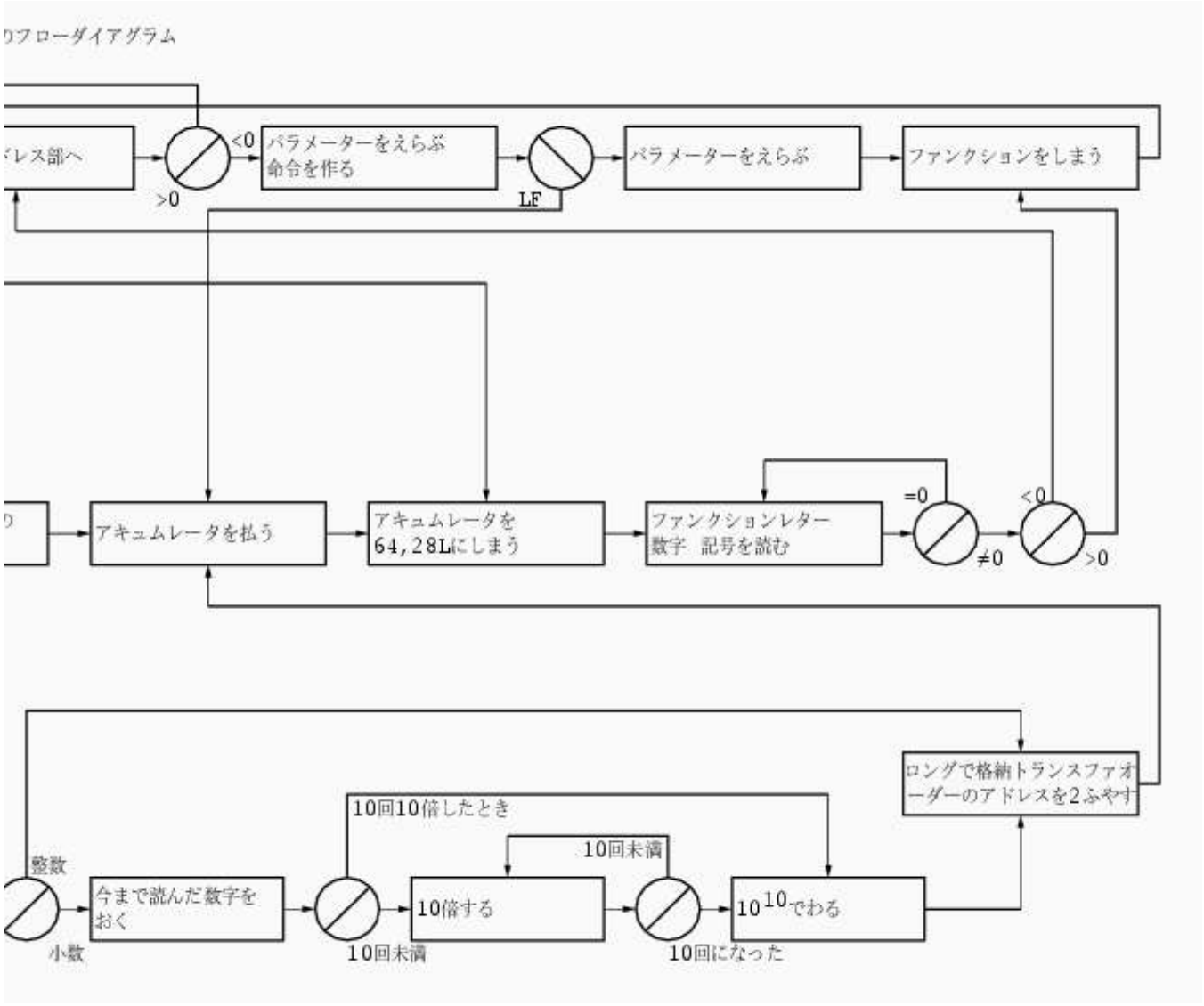
```
          Q        平方根を求めるサブルーチン
        0q:
  0   a  40
  1   x  8r
  2   pl  g
  3   tl  o
  4   nl  o
  5   vl  o
  6   al  g
  7   rl  1
  8   zl(  )
  9   dl  o
 10   al  o
 11   jl 3r
```

印刷電信機のコード表

a	011000	i	001100	q	011101	y	010101	,	100110	2	110000
b	010011	j	011010	r	001010	z	010001	.	100111	3	101010
c	001110	k	011110	s	010100	復帰(CR)	000010	:	101111	4	100001
d	010010	l	001001	t	000001	間隔(SP)	000100	=	110111	5	110101
e	010000	m	000111	u	011100	改行(LF)	001000	+	101000	6	111100
f	010110	n	000110	v	001111	上段(FIGS)	011011	-	100010	7	101100
g	001011	o	000011	w	011010	下段(LTRS)	011111	0	111000	8	100011
h	000101	p	001101	x	010111	信号(BELL)	100000	1	111001	9	101101



R0およびR1のフローダイアグラム(左半分)



R0およびR1のフローダイアグラム(右半分)