1996 年 夏のプログラミングシンポジウム
コンピューティングの歴史

1996-07-23
@ 箱根 静雲荘

パラメトロン計算機 PC-1 – 回路設計と方式設計 –

富士通研究所
和田英一
(wada@u-tokyo.ac.jp)

はじめに

東大物理学教室の高橋研で自作したパラメトロン計算機 PC-1 のプログラムまわりについてはいろいろな機会に書いたり [1][2] 話したりしたので, 今回は PC-1 のハードまわりの話しがしたい. PC-1 は 1958 年 3 月 26 日に完成, 高橋研だけでなく, 他の研究室にも公開して使って貰っていた [3]. 1964 年の 5 月祭で, 学生が PC-1 の電源を借りにきたので, 電源をはずした機会に, 保守, 運転を中止した. 高橋先生のご著書など [4][5] にも書いてあることだが, 東大で作ろうとしていた TAC(Todai Automatic Computer) の計画が進んでいたころ, 高橋研でも自分の計算機がほしいということで, さまざまな検討をしていた. 私が高橋研に入った時分, 実験室にまだ転がっていたのが機械電子式という試作機の一部で, そういうものも考えていたらしいが, 1954 年に当時大学院生の後藤英一さんが, パラメータ励振を利用した計算素子を考えだし, それがパラメトロンの発明になった [6].

PC-1 の概要を以下に示す.

パラメトロン計算機 PC-1 概要

論理素子: パラメトロン 4200 個 励振周波数 2.3MHz, 発振 (共振) 周波数 1.15MHz,
クロック周波数 15KHz

記憶装置: 18 ビット短語 512 語 2 周波磁気コア

演算装置: 36 ビットレジスタ 3 個 (アキュムレータ, R レジスタ, メモリーレジスタ)

命令語長: 18 ビット 命令種類 27

演算時間: 加減算 4 τ, 乗算 26 τ (短語) 44 τ (長語), 除算 161 τ, 書き込み 8 τ,
無条件ジャンプ 4 τ, 条件ジャンプ 8 τ
(1 τはクロック周波数の逆数 = 67 μ秒)

入力装置: 光電紙テープリーダ

出力装置: テレタイプ

運転開始: 1958 年 3 月 26 日

運転終了: 1964 年 5 月

回路素子パラメトロン

パラメータ励振というのは, 周期 T(周波数 $f = 1/T$) の共振回路があったとき, この周期を $T/2$ (周波数 $2f$) で変化させると共振回路の発振が大きくなる現象である. 周期を変化させるとは, C と L でできた共振回路では, 周期 T は $T = 2\pi\sqrt{LC}$ だから, L か C の値を周期 $T/2$ で変化させればよい.

パラメータ励振でよく知られているのはぶらんこである. 支点から重心までの長さ l の振り子の振動周期は $T = 2\pi\sqrt{l/g}$ である. 周期を決める要素はここでは l だから, l の長さを, 振り子が 1 往復するあいだに 2 回伸縮させればよい. ぶらんこを漕ぐ子どもは, 2 回立ったりしゃがんだりするのである.

ロンドンのサイエンスミュージアムに行くと, 上野の科学博物館とおなじで, Foucault 振子が動いているが, これが Peppart 振子になっている. 振り子の支点に仕掛けがあり, 振り子が 1 往復する間に糸を 2 回上げ下げする. これで振り子はいつまでも揺れている.

さて, 子どもが 2 回立ったりしゃがんだりしたとき, ぶらんこは前に行ってから後ろへ戻るか, 後ろへ行ってから前へ戻るか, 揺れ方に両方の可能性がある. いずれかのフェーズで揺れていると, 子どもが漕いでいる限り, いつまでもそのフェーズで揺れ続ける. このフェーズは子どもが漕ぎ出した時の小さな揺れのフェーズで決まってしまう. Figure 1 にこの辺の事情を示す. 灰色の早い振動が励振である. おなじ励振に対し, 破線または 1 点鎖線のような共振が可能であり, どちらかのフェーズが 0, もう一方のフェーズが 1 を表すと考える.

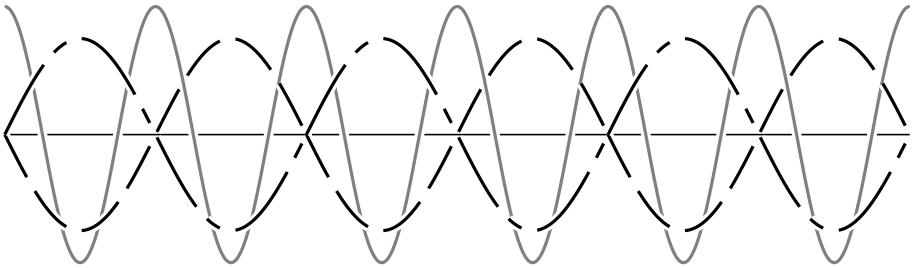


Figure 1

パラメトロン 1 個は Figure 2 のようなものである. 中央の 2 個のトランスのように書いてあるのはフェライトのコアで, 右側の巻線とコンデンサとで共振回路になっている. 左側の巻線には直流でバイアスした高周波が流れ, フェライトの透磁率が周期的に変化し, したがってインダクタンス, 共振周波数も周期的に変化し, パラメータ励振が起きる. フェライトコアが 2 個あって, 励振線が互いに逆に巻いてあるのは, 励振が共振に影響を与えないようにするためである.

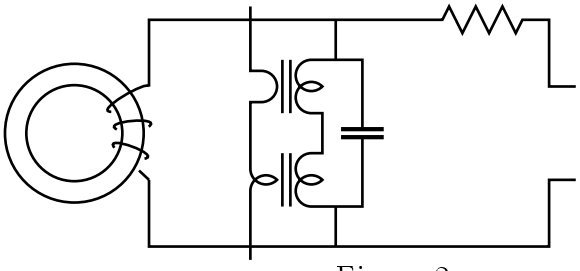


Figure 2

図の左端にあるトランスは情報を伝達するためのものである．前述のように励振の続く限り共振のフェーズは変わらず, それでは面白いことは出来ないので, 3 個のパラメトロンを I 相, II 相, III 相に分け, まず I 相だけ励振する．しばらくしてから, I 相の共振フェーズの情報をトランスで II 相に伝え, II 相のパラメトロンの共振のフェーズを制御する．II 相の励振で II 相が共振を始めたら, I 相の励振は止める．またしばらくしてから, II 相の共振フェーズの情報をトランスで III 相に伝え, III 相のパラメトロンの共振のフェーズを制御する．III 相の励振で III 相が共振を始めたら, II 相の励振は止める．III 相の次は I 相へ伝える．その様子を Figure 3 に示す．これを 3 拍励振といい, 励振をかけたり止めたりの回数を変調周波数とかクロック周波数という.

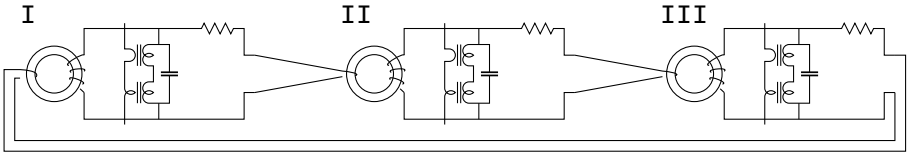


Figure 3

あるパラメトロンの入力トランスに前相の 3 個のパラメトロンの出力を入れると, このパラメトロンの共振フェーズは入力パラメトロンのフェーズの多数決で決まる．つまり入力が 0, 0, 0 または 0, 0, 1 なら共振は 0 となり, 0, 1, 1 または 1, 1, 1 なら共振は 1 となる．前相の出力を否定して入力したければ, トランスの通し方を反対にすればよい．Figure 4 には 3 個のパラメトロンから入力を貰うパラメトロンが描いてあるが, 一番下からは否定で貰う図になっている.

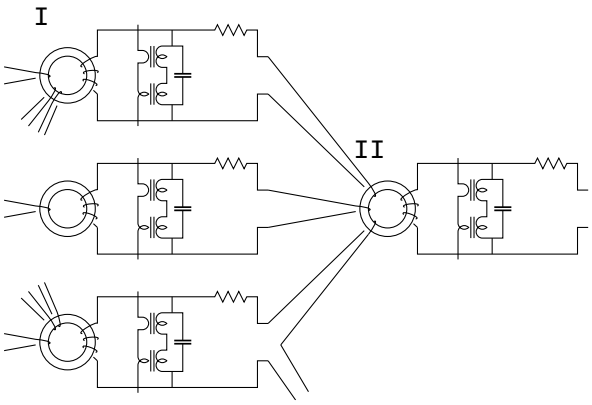


Figure 4

Table 1

x	y	z	$[x, y, z]$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

多数決と否定があれば論理関数を実現するのは簡単である．3 変数 x, y, z の多数決関数は $[x, y, z]$ のように書くことがある．(Table 1 参照) 通常の論理式で書くと

$$[x, y, z] = (y \wedge z) \vee (z \wedge x) \vee (x \wedge y)$$

となる．3 入力のうち 1 入力が常に 0 なら, 残りの 2 入力が共に 1 の時のみ出力は 1 となるので, 論理積となり, 1 入力が常に 1 なら, 残りの 2 入力のいずれかが 1 の時に出力は 1 となるので, 論理和になる.

$$x \wedge y = [x, y, 0] \qquad x \supset y = \neg x \vee y = [\overline{x}, y, 1]$$
$$x \vee y = [x, y, 1] \qquad x \subset y = x \vee \neg y = [x, \overline{y}, 1]$$

回路図は Figure 5 のように書く. 常に 0, 1 のような定数パラメトロンは入力として表示すると煩わしく, Figure 5 のようにパラメトロンを示す円内に, 0 なら−, 1 なら + を書くことにしている. 同図左下は $\neg x \vee y$ の回路である. 結線上の横棒は否定を示す.

2 入力の排他的論理和や 3 入力の全加算器のように, 単調でない関数は多数決 1 段ではできす, Figure 5 右のように 2 段で構成する. 全加算器について, 途中の段のパラメトロン a, b, c と最終出力 s の真偽値は Table 2 の通りである. 詳しくは文献 [7][8] を参照してほしい.

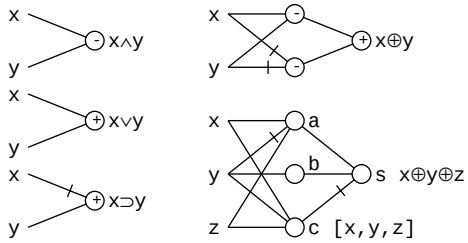


Figure 5

Table 2

y	x	z	$\overline{y}$	c	a	b	$\overline{c}$	s
0	0	0	1	0	0	0	1	0
0	0	1	1	0	1	0	1	1
1	0	0	0	0	0	1	1	1
1	0	1	0	1	0	1	0	0
0	1	0	1	0	1	0	1	1
0	1	1	1	1	1	0	0	0
1	1	0	0	1	0	1	0	0
1	1	1	0	1	1	1	0	1

全加算器には定数入力がない. これを自己双対関数という. つまり上の表で 0 と 1 を交換すると表を下から読むのと同じになる. もう一つの有用な自己双対なものは高速桁上げ回路であり, Figure 6 に示す.

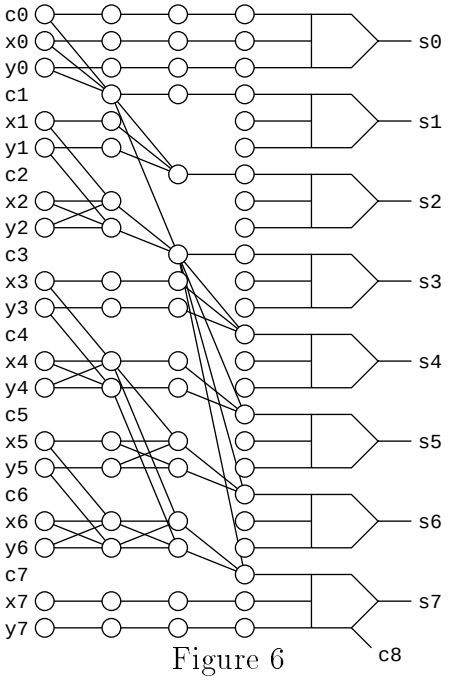


Figure 6

この図で右に縦にならんでいる 5 角形は全加算器である. 図は 8 ビットの 2 進数 x, y の各ビットを下の桁から $x_0, x_1, \ldots, x_7, y_0, y_1, \ldots, y_7$ として足すもので, また end around carry c_0 の入力もある. 普通には x_0, y_0, c_0 を全加算し, 部分和 s_0 と次段への繰り上がり c_1 が得られ, 次にその c_1 と x_1, y_1 を全加算し, 部分和 s_1 と c_2 が得られ, ... と続く. これでは n 桁の加算に n ステップかかる. それに対し, 高速桁上げ回路では, まず $log_2 n$ ステップですべての桁への繰り上げを求め, しかる後に一斉に全加算するのである. c_1, c_2 はそれぞれ $[c_0, x_0, y_0]$, $[c_1, x_1, y_1]$ で 2 ステップかかる. 次の c_3 だが, x_2, y_2 が共に 0, または共に 1 なら 0 または 1 になる. x_2, y_2 が 0, 1 または 1, 0 のときは, x_1, y_1 が共に 0, または共に 1 なら 0 または 1 になる. x_1, y_1 が 0, 1 または 1, 0 なら c_1 の 0 または 1 による. 上の方の桁も同様に考えることができる.

記憶装置

パラメトロンに比べ, あまり話題に上らないが PC-1 のメモリー回路やアドレス選択回路も非常に独特なものであった.

Figure 7 にコアメモリーを構成する 1 個のコアを示す. 36 ビットのメモリーレジスタが左方にあり, 語中の各ビットの共振フェーズの電流が横線を流れる. また, 上方にはアドレス選択回路があり, 選択された語に相当する縦線に共振周波数 f の半分, $f/2$ の選択電流が流れる. 選択電流のないコアは対称な磁場をかけるので, コアの磁化には変化がないが, 選択電流のある語は Figure 7 の下の図にあるように非対称な磁場を生じるため, コアはどちらかに磁化される.

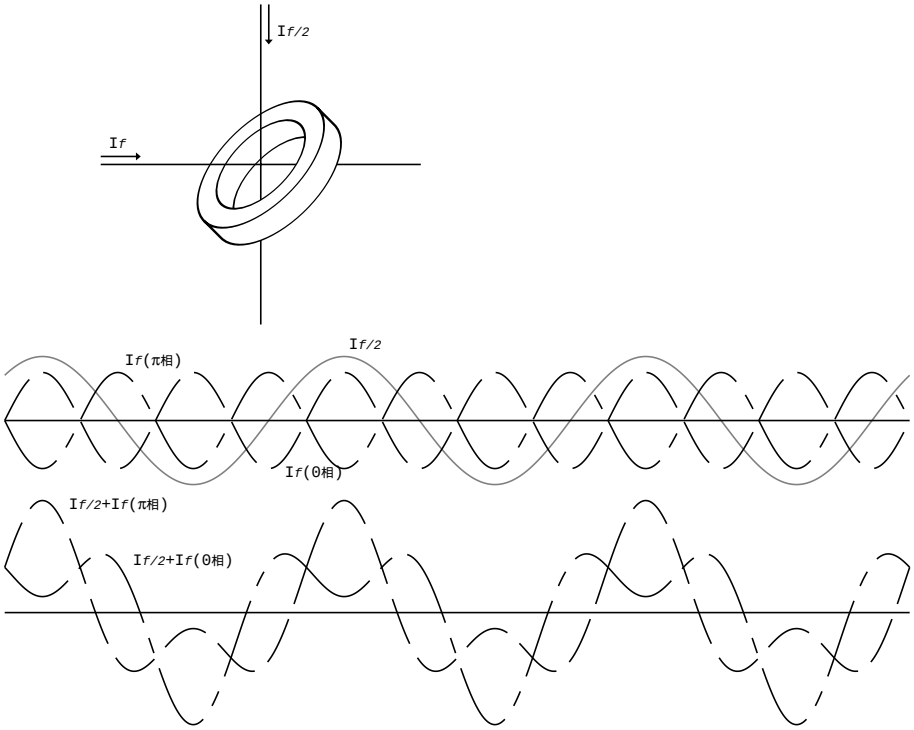


Figure 7

読み出しの原理は Figure 8 の通り. 読み出したい語のコアに, 選択電流を流す. ヒステリシスカーブで, 磁化が逆転しない程度の電流とする. すると 0 を記憶しているコアと, 1 を記憶しているコアとで, 図に示すようなヒステリシスカーブを描き, しかも磁化は不変のため, 非破壊読み出しができる. ヒステリシスカーブを回るために読み出し線に誘起される電圧は記憶内容により, フェーズが 0 かπの 2 次高調波 (周波数 f) を持つので, これをパラメトロン共振の種にすれば, 読み出せるのである. このように f と $f/2$ を使うので 2 周波メモリーという.

アドレス選択の $f/2$ 電流は多少大きめな $f/2$ パラメトロンの発振を利用した. PC-1 のメモリーは 36 ビット長語× 256 語で構成してあり, $f/2$ パラメトロンも 256 個用意した. 256 個のうちか

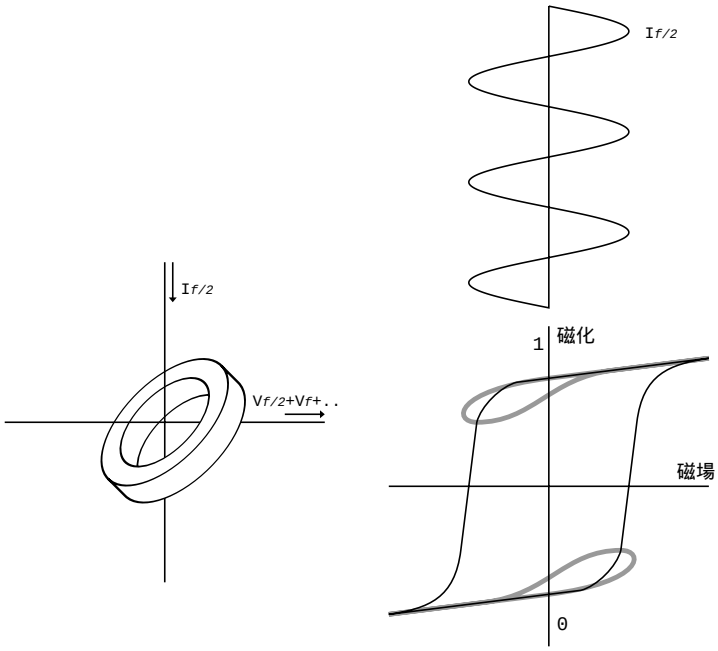


Figure 8

ら 1 個を選ぶには通常は 8 ビットの選択回路を使用するのだが, PC-1 の語選択の概略は次のようになっていた.

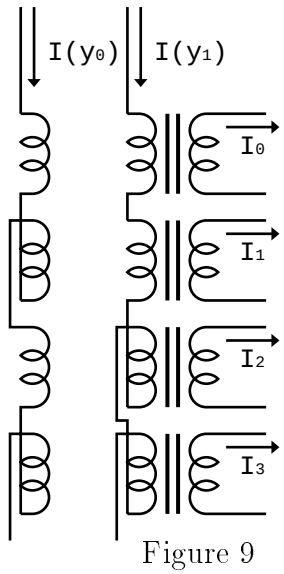


Figure 9

Table 3

駆動電流		出力電流			
$I(y_0)$	$I(y_1)$	I_0	I_1	I_2	I_3
$-I$	$-I$	$-2I$	0	0	$+2I$
$+I$	$-I$	0	$-2I$	$+2I$	0
$-I$	$+I$	0	$+2I$	$-2I$	0
$+I$	$+I$	$+2I$	0	0	$-2I$

Table 4

0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	0	1	1	0	0	1	1	1	1	0	0	1	1	0	0	1
0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
0	1	1	0	1	1	0	0	1	1	0	1	1	0	1	1	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0	1	1	1	0	0	1	1	0	1	1	0	0	1	1	0	0
0	1	0	1	1	0	1	0	1	0	1	0	1	0	0	1	0

Figure 9 のようなトランスで, $I(y_0)$ と $I(y_1)$ にパラメトロンの共振のように, 振幅, 周波数は同じだが, フェーズが 0 相かπ相の電流が流れたとする. それを $+I$, $-I$ で表す. それに対し出力電流は Table 3 のようになるであろう. パラメトロンには, 目いっぱい励振をかけられた時には発振

し, 励振がある程度以下だと発振しないという性質があり, 選択回路用のパラメトロンを出力側で励振すると, ±2I のものだけ発振する. いまは 2I と 0 だが, 入力本数が増えて J 本になると, 最大励振は JI なのに対し, 次は $(J-2)$I で, 識別が苦しくなる.
入力が 4 ビットの場合, Tabel 4 のような Hamming コードを使ったとすると, 最大が 7I に対し, 次点は ±I となるので, 識別は容易である. PC-1 ではこのように誤り訂正符号を利用することにし, アドレス選択信号から符号変換を経て 18 回線で増幅され, トランス回路で識別可能なレベルの励振電流を作って選択用パラメトロンの特定の 1 個を共振させている.
PC-1 に見学者が来ると, 高橋先生は 18 本の真空管の 1, 2 本を抜いて見せ, それでも PC-1 が誤動作しないことで, このアドレス選択方式の利点を説明されていた.

PC-1 の方式設計

PC-1 の方式設計は原則的には Cambridge 大学, Compter Laboratory の EDSAC のそれを見習った [9]. しかし, 独自の哲学で設計した部分もある. 最初にアドレス方式, レジスタ構成を説明する.

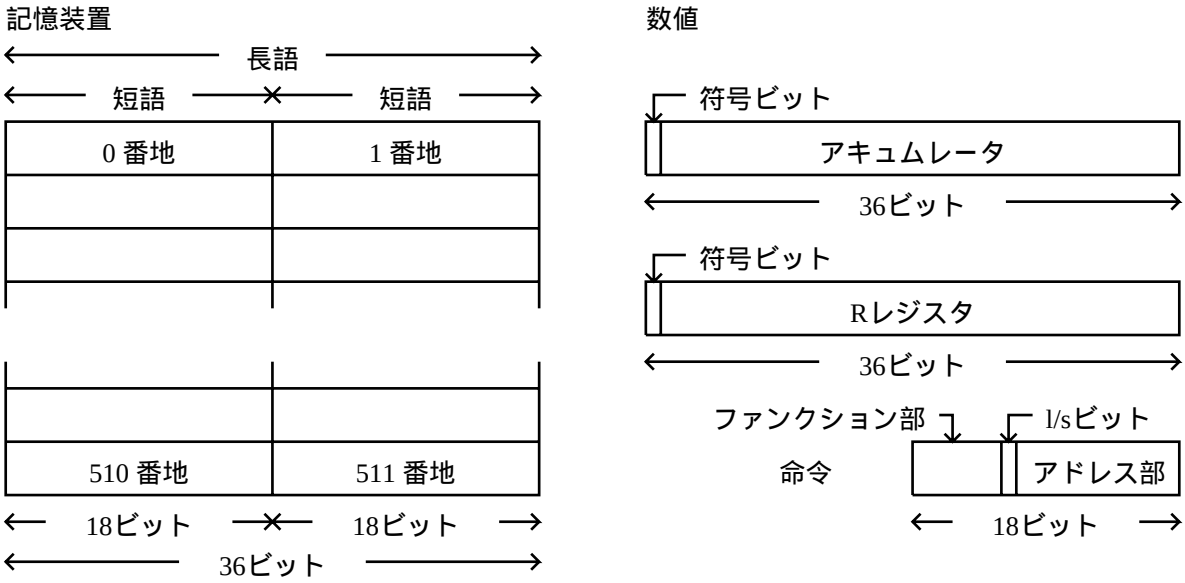


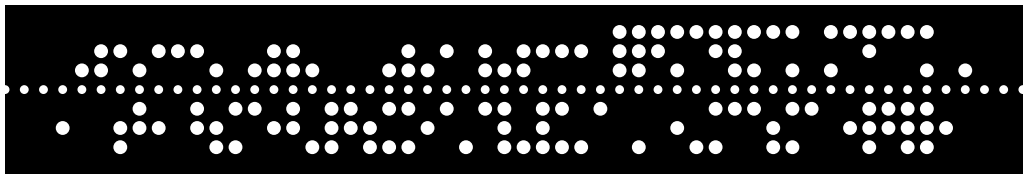
Figure 10

Figure 10 の左が記憶装置で, 図に示すように 18 ビットの短語で 512 語ある. EDSAC と同様に $2n$ と $2n+1$ 番地の短語は 2 語合わさって長語としても使える. 命令語は短語だが, 数値語は短語, 長語のいずれでもよい. 長語を指定する時は命令語のアドレスは偶数番地にする他, 命令語のなかの l/s ビットを 1 にしておく. 命令によっては l/s ビットは長語 / 短語の区別以外の目的に使うこともある.

Figure 10 の右は処理装置内のレジスタを示す. PC-1 で扱う数値は 2 の補数表示の 2 進固定小数で, 左端に符号ビットがあり, そのすぐ下に小数点があると考える. 演算レジスタにはアキュムレータと R レジスタがあり, 加減算, ビット演算にはアキュムレータを使う. R レジスタは小数点のところにアキュムレータの最後からつなげ, 小数部 70 ビットのレジスタとしてシフト, 乗除算に使うことができる. 図には示してないが, 記憶装置から読みだした数値はメモリーレジスタに入り, アキュムレータとメモリーレジスタの演算の結果がアキュムレータに残るというのが基本形である. 短語を読み出すとメモリレジスタの上 18 ビットに入り, 下 18 ビットはクリアされる. 命令は命令レジスタに読み出され, 先頭から 6 ビットがファンクション部, 1 ビットが l/s ビット, 11 ビットがアドレス部である. 記憶装置が 512 語なので, アドレスには 2 ビット余裕があった.
PC-1 では入出力に紙テープとテレタイプを利用していた. テレタイプのコードは 6 単位で, 欧文 5 単位テレタイプの letter には最上位ビットに 0, figure には 1 をつけたものである. letter 側の下 5 単位は ITA no.2 と同じだが, figure 側は多少違っている.

000000 Blank	010000 e	100000 Signal	110000 2
000001 t	010001 z	100001 4	110001
000010 C.R.	010010 d	100010 -	110010
000011 o	010011 b	100011 8	110011
000100 Space	010100 s	100100	110100
000101 h	010101 y	100101	110101 5
000110 n	010110 f	100110 ,	110110
000111 m	010111 x	100111 .	110111 =
001000 L.F.	011000 a	101000 +	111000 0
001001 l	011001 w	101001	111001 1
001010 r	011010 j	101010 3	111010
001011 g	011011 F.S.	101011	111011
001100 i	011100 u	101100 7	111100 6
001101 p	011101 q	101101 9	111101
001110 c	011110 k	101110	111110
001111 v	011111 L.S.	101111 :	111111 Erase

実物大の紙テープは (本稿を A4 版で出力時) 次の通りである.



abcdefghijklmnopqrstuvwxyz 0123456789 +-=,.:
Figure 11

PC-1 では市販のテレタイプをそのまま利用した. その点では EDSAC が 10 進 2 進変換が楽なようにコードバーを切り直したり, ILLIAC で 16 進数として 4 単位しか使用しなかったのとは違ってできるだけ何もしないという発想に基づいていた.

PC-1 の命令の一覧表は次の通りである. 詳細は文献 [7] の付録 命令一覧表を参照してほしい.

an al$n$	n, nL の内容をアキュムレータにたす.
bn bl$n$	n, nL の内容とアキュムレータの排他的論理和をアキュムレータにおく.
cn cl$n$	n, nL の内容とアキュムレータの論理積をアキュムレータにおく.
dn dl$n$	n, nL の内容でアキュムレータと R レジスタをわり, 商をアキュムレータに, 剰余を R レジスタにおく.
in	入力装置が停止中ならテープから 1 字読んで a_0 ~ a_5 にいれ a_6 ~ a_{35} を 0 にする. 動作中なら n へジャンプする.
jl$n$	n へジャンプする.
kn	アキュムレータの内容 <0 なら n へジャンプする.
kl$n$	アキュムレータの内容 $>=0$ なら n へジャンプする.
ln	$n < 1024$ ならアキュムレータを n ケタ左へシフト, $n >= 1024$ なら $2048 - n$ ケタ右へ論理シフト.
ll$n$	アキュムレータと R レジスタをシフトするほか ln に同じ.
nn nl$n$	アキュムレータをクリアして n, nL の内容をひく.
on	出力装置が停止中なら a_0 ~ a_5 をテレタイプに出力する. 動作中なら n へジャンプする.
pn pl$n$	アキュムレータをクリアして n, nL の内容をたす.
qn ql$n$	R レジスタの内容をアキュムレータにおき, n, nL の内容を R レジスタにおく.
rn	$n < 1024$ ならアキュムレータを n ケタ右へシフト, $n >= 1024$ なら $2048 - n$ ケタ左へシフト.
rl$n$	アキュムレータと R レジスタをシフトするほか rn に同じ.
sn sl$n$	n, nL の内容をアキュムレータからひく.
tn tl$n$	アキュムレータを n, nL に書き込む.
vn vl$n$	アキュムレータに n, nL の内容をかけ, 結果をアキュムレータと R レジスタにおく.
wn wl$n$	vn vl$n$ のほかに, はじめの R レジスタの内容を wn なら 2^{-17} 倍, wl$n$ なら 2^{-35} 倍したものを積にたす.
xn	アキュムレータの a_7 ~ a_{17} を n のアドレス部に書き込む.
zn	アキュムレータの a_7 ~ a_{17} が 0 なら n へジャンプする.
zl$n$	アキュムレータの内容 $=0$ なら n へジャンプする.

EDSAC と違う点を少し挙げておく. EDSAC ではたす命令, ひく命令の他, アキュムレータの内容をメモリーに格納してからアキュムレータをクリアする命令と格納してもクリアしない命令があった. つまりこれで一連の演算の最後という時は, 結果をしまうとともにアキュムレータを次の演算のためにクリアするのである. 一方 PC-1 は, たす, ひくのほか, アキュムレータをクリアしてからたし込む (ポジティブロード) 命令と, クリアしてからひき去る (ネガティブロード) 命令を持っていた. その他に (クリアしない) 格納命令があったから, PC-1 ではこの種の命令は都合 5 種類あったことになり, EDSAC ではそれを 4 種類ですませていた.
かけ算もまったく違う方式で, PC-1 はアキュムレータの内容にある番地の内容をかけ, それをアキュムレータ (と R レジスタ) におく方式であり, EDSAC は別にある乗算レジスタの内容とある番地の内容をかけ, それをアキュムレータにたしたりひいたりするようになっていた. EDSAC は積和をとるのに便利であれという設計である. PC-1 では積和は苦手でも, Horner の方法による

多項式の値の計算には向いていた. PC-1 には w 命令があったが, これはかけ算でまず部分積をおくところをクリアして始めるのに対し, そのクリアをさぼるものである. したがって演算前に R レジスタにあった内容が, アキュムレータとメモリーの内容の積にたされるという (桁数には制限があるが) 一応の積和は可能であった.

わり算は EDSAC ではサブルーチンを使っていた. PC-1 でも 1958 年 3 月に稼働し始めた時は, かけ算, わり算はなく, サブルーチンが存在した. 使いながらかけ算とわり算の回路を設計し, 配線を行なった.

加減算が可能なレジスタがアキュムレータだけという演算回路では, わり算の後で, 剰余はアキュムレータに残り, アキュムレータの下にシフトでつながるレジスタに商が入るというのが標準で, ILLIAC ではそのレジスタに商ができるという意味で Q レジスタと呼んでいたが, PC-1 では, わり算の結果では商の方が重要であり, 重要な方がアキュムレータに残るべきだということで, 演算後に商と剰余の入れ換えを行なっている.

わり算では剰余の符号のとりかたに諸方式がある. 一般的なのは剰余の符号を被除数の符号にあわせるというやつで, 多くのプログラム言語でもそう規定しているようである. しかし, 多倍長の被除数をわる時は, 2 の補数の計算機では 2 回目以降に剰余をわる時からは正の商が出てほしい.

Table 5

n \ d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16
-4	-3, 3	-3, 2	-3, 1	-3, 0	-2, 3	-2, 2	-2, 1	-2, 0	-1, 3	-1, 2	-1, 1	-1, 0	0, 3	0, 2	0, 1	0, 0	1, 3	1, 2	1, 1	1, 0	2, 3	2, 2	2, 1	2, 0	3, 3	3, 2	3, 1	3, 0	* -4, 3	* -4, 2	* -4, 1	* -4, 0
-3		-4, 2	-4, 1	-4, 0	-3, 2	-3, 1	-3, 0	-2, 2	-2, 1	-2, 0	-1, 2	-1, 1	-1, 0	0, 2	0, 1	0, 0	1, 2	1, 1	1, 0	2, 2	2, 1	2, 0	3, 2	3, 1	3, 0	-4, 2	-4, 1	-4, 0	*	*	*	*
-2		†	†	†			-4, 1	-4, 0	-3, 1	-3, 0	-2, 1	-2, 0	-1, 1	-1, 0	0, 1	0, 0	1, 1	1, 0	2, 1	2, 0	3, 1	3, 0	-4, 1	-4, 0		*	*	*				
-1							†	†				† -4, 0	-3, 0	-2, 0	-1, 0	0, 0	1, 0	2, 0	3, 0	-4, 0			*	*								
0												†								*												
1									3, 1	3, 0	2, 1	2, 0	3, 0	2, 0	1, 0	0, 0	-1, 0	-2, 0	-3, 0	-4, 0												
2													1, 1	1, 0	0, 1	0, 0	-1, 1	-1, 0	-2, 1	-2, 0	-3, 1	-3, 0	-4, 1	-4, 0								
3					3, 2	3, 1	3, 0	2, 2	2, 1	2, 0	1, 2	1, 1	1, 0	0, 2	0, 1	0, 0	-1, 2	-1, 1	-1, 0	-2, 2	-2, 1	-3, 2	-3, 1	-3, 0	-4, 2	-4, 1	-4, 0						
d	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16

それには剰余の符号は除数の符号と合わせるのが良い. しかし PC-1 では, 負の除数で多倍長をわることはそんなにないだろうと, 剰余の符号が常に正となるように決めた. ここはもうひと踏ん張りし, 剰余と除数の符号をあわせるべきであったとの反省がある. Table 5 に剰余と除数の符号を合わせると, 商と剰余がどうなるかの概略を示した. (n は被除数, d は除数)
シフトは EDSAC の命令では判じものだが, アドレス解読の回路は簡単にできるのだから, アドレスが n なら n ビットシフトというわかりやすい方式にした. シフト命令の実行時間は固定値 $+n$ クロックなので, これを利用して時間を計ることができた.
シフトの仲間ではないが q 命令というのがあった. なぜ q かという質問にだれかが quick shift だといったので, シフトで思い出した. これはそもそもは e だのπだのの計算で多倍長わり算をやる場合に便利だというのでつけたのだが, R レジスタの活用には重要であった.
ビット演算はビット毎の論理積 (c 命令) とビット毎の排他的論理和 (b 命令) をもっていた. 論理和はこの二つの命令があれば, $a \vee b = a \wedge b \oplus a \oplus b$ でできるからいらないという理由である. PDP-8 という命令数の少ない計算機があって, それにはビット毎演算は論理積しかなく, 排他的論理和は 2 数の和から論理積で用意した繰り上がりを引いて得るというのであったが, そこまではしなくてすんでいる.
ジャンプ命令は無条件ジャンプ, 正負, 零の条件ジャンプがある. EDSAC では正負の条件ジャンプだけだから, ハッカー向きだが, PC-1 ではそういう無理はしない. また, アドレス部の零ジャンプはアドレス部の書き込み命令 (x 命令) とともにループ制御に威力があった.
入出力は EDSAC がメモリーの語との出し入れなのに対し, PC-1 はアキュムレータと出し入れする. さらに i, o 命令はアドレス部が不要になったため, 入出力機器の準備ができていないときには, アドレスの示す番地へジャンプする, いわゆる聖徳太子の機能がついていた. もっとも, この機能は割り込みがついてから, 割り込み処理ルーチンの中で初めて実用になった.
前にあげた命令一覧表にはないが, PC-1 にはフリップフロップがあり, y 命令でセット, リセット, e 命令でフリップフロップの条件ジャンプができた. このフリップフロップは割り込み禁止用に使ったが, 割り込みが生じると自動的にセットされ, 割り込み処理ルーチンの出口で, プログラムでリセットした. このフリップフロップにスピーカを接続し, 音楽をならしたのも楽しい思い出である.
最後に割り込みのことを述べよう. PC- が完成してほぼ 1 年使い, ライブラリも増えてきたころ, 高橋先生が走行中のプログラムに外から制御が出来ないのは面白くない, といい出された. 先述のフリップフロップは手動でもセットできたが, 入出力機器その他で制御できれば便利だということであった. そこで早速割り込み回路を設計した. 当面テレタイプが受信 OK になると, 割り込みが発生し, さらなる割り込みをフリップフロップで禁止し, 次に実行するはずの命令の番地を 510 番地へ格納し, 511 番地へ実行を移す. 511 番地には割り込み処理ルーチンへのジャンプ命令がおいてある, というものであった. さっそくサイクリックバッファを共用するかたちで, 主ルーチンと割り込み処理ルーチンの強調プログラムを書き, マルチプログラミングの世界初とも思わ

れる実験を行なった.
割り込みが出来ればトラップ (割だしとも呼んだ) へ考えは及び, メモリープロテクションのアイディアもでたが, これはアイディアだけで, PC-1 への組み込みは行なわなかった [10].

おわりに

岩波情報科学辞典を眺めていたら「PC-1」の項目があり, その最後の方に「しかし, 初期のわが国の計算機技術の発展にこの計算機が果した役割は大きい.」と書いてあった. 結果的にはそうかもしれないが, PC-1 の関係者は新しいことを次々と試み, 楽しくて仕方がないという状態であった. もう 38 年も前のことであるから回路設計からプログラムライブラリの整備まで, すべて自分たちの手でやらなければならず, またやれた時代であった.
PC-1 の稼働をやめた 1964 年は IBM が System 360 を発表している. それからしばらくは大型計算機の時代が続き, また DEC の小型機も登場はしていきたが, まだ PC-1 時代のように自由に楽しく使える程ではなかった. やっと最近, ワークステーションやパソコンが出回るようになって, PC-1 時代の再来のような気がするが, 進歩の度合はものすごく, やはり神代というべき時代だったかもしれない.
PC-1 が元気だったころ, 3 月 26 日には PC-1 の誕生パーティーをやっていた. 最近では 5 の倍数年目の 3 月 26 日にほそぼそと集まって昔話をしたりしている.

参考文献

[1] 和田英一:「PC-1 のイニシァル・オーダー R0」, bit Vol.4, No.12, pp.57–69

[2] 情報処理学会歴史特別委員会編:「日本のコンピュータの歴史」, オーム社 1985 年

[3] 日本物理学会:「電子計算機 – 使い方と応用 – 電子計算機講習会テキスト」 1959 年

[4] 高橋秀俊:「電子計算機の誕生」中公新書 273 中央公論社 1972 年

[5] 高橋秀俊:「コンピュータへの道」文芸春秋 1979 年

[6] 「後藤英一 パラメトロンから量子コンピュータまで」月刊アスキースペシャルインタビュー 2, Ascii, Vol.17, No.6(1993 年 6 月号)

[7] 高橋秀俊 編:「パラメトロン計算機」岩波書店 1968 年

[8] 高橋秀俊: 計算機械 II, 岩波講座 現代応用数学 B.14-a.II 1958 年 岩波書店

[9] Wilkes, M.V. etal:"The Preparation of Programs for an Electronic Digital Computer", Addison-Wesley 1951 年

[10] 和田英一:「モニタシステム」, 情報処理, Vol 3, No. 9, pp 267–277, 1962 年