A.C.C NEWSLETTER VOL 3 ISS 2 JUNE 75

Pattern consists of six separate sub-patterns, each of which is based on an equilateral triangle, fitted together to form a hexagon.

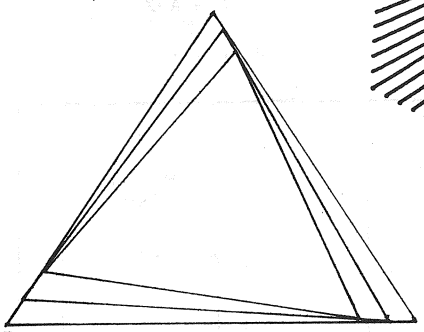
Each triangle is formed by marking three points on the edges of the triangle a constant small distance from the corners rotating in the same direction then joining these points with straight lines to form another but slightly smaller triangle.

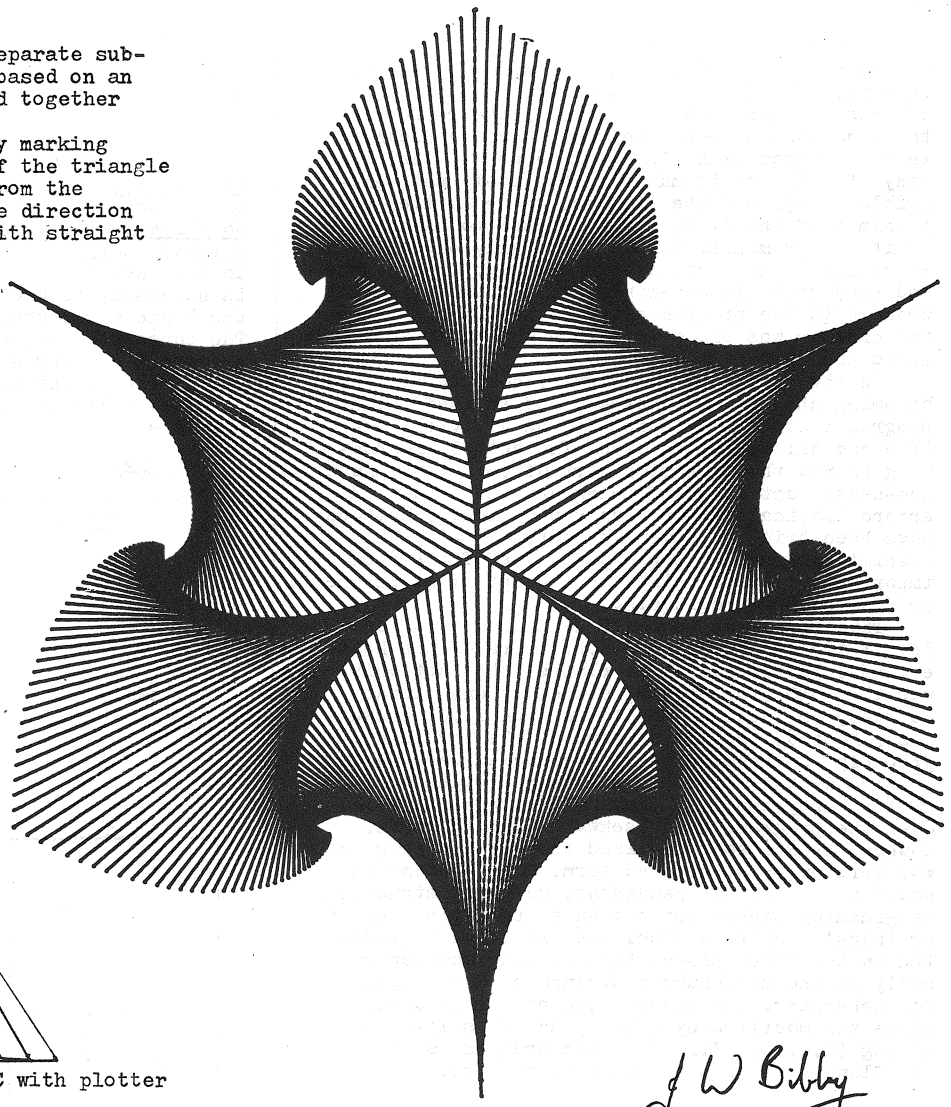
This procedure is repeated until the final triangle is small enough.

Adjacent triangles in pattern rotate in opp-osite directions.

Variations can be formed by not drawing 1 or 2 of the lines or by adding more patterns together.

Additional variations may be possible by starting with triangles other than equilateral and not arranging for them to be adjacent.





J W Bibby

Program for HP 2000F TS BASIC with plotter

```
CPLOT

5   REM- GIVES +1 IF EVEN -1 IF ODD
10  DEF FNK(K)=(K=2*INT(K/2))*2-1
15  REM- LIMITS OF PLOTTER
20  DEF FNP(X)=1 MAX X MIN 9999
25  REM- S=SIZE OF SUB-PATTERN
          B9=NO OF LINES TO A SUB-PATTERN
30  INPUT S,B9
35  REM- COORDINATES OF CENTRE OF PATTERN
40  INPUT X9,Y9
45  REM- VALUE OF PYE
50  P1=ATN(1)*4
60  P2=P1/180
65  REM- PRINT SIX SUB-PATTERNS
70  FOR K=1 TO 6
80  PRINT "PLTL"
85  REM- ANGLE OF SUB-PATTERN
90  O=K*60-30
100  B=B9
105  REM- CENTRE OF SUB-PATTERN
110  X=X9+S*SIN(O*P2)
120  Y=Y9+S*COS(O*P2)
125  REM- ANTI OR CLOCKWISE
130  O1=(O-60*FNK(K))*P2
140  O2=(O+60*FNK(K))*P2
145  REM- 3 CORNER CORDINATES
150  X1=X+S*SIN(O1)
160  Y1=Y+S*COS(O1)
170  X2=X+S*SIN(O2)
180  Y2=Y+S*COS(O2)
190  X3=X9
200  Y3=Y9
205  REM- WILL IT FIT ONTO PLOTTER ?
210  IF X1#FNP(X1) OR Y1#FNP(Y1) THEN 370
220  IF X2#FNP(X2) OR Y2#FNP(Y2) THEN 370
230  IF X3#FNP(X3) OR Y3#FNP(Y3) THEN 370
240  GOSUB 400
245  REM-MOVE ALL 3 POINTS BY INCREMENT
       TOWARDS OTHER POINTS IN ROTATION
250  X0=X2-X1
260  X2=X2+(X3-X2)/B
270  X3=X3+(X1-X3)/B
280  X1=X1+X0/B
290  Y0=Y2-Y1
300  Y2=Y2+(Y3-Y2)/B
310  Y3=Y3+(Y1-Y3)/B
320  Y1=Y1+Y0/B
325  REM- CHANGE INCREMENT
330  GOSUB 400
340  B=B-B/B9
345  REM- SMALL ENOUGH YET ?
350  Z0=SQR(Y0*Y0+X0*X0)
360  IF Z0>S/10 THEN 250
370  PRINT "PLTT"
380  NEXT K
385  REM- GO AND INPUT CENTRE OF NEXT PATTERN
390  GOTO 40
395  REM- PRINT ROUTINE
400  PRINT  USING 440;X1,Y1
410  PRINT  USING 440;X2,Y2
420  PRINT  USING 440;X3,Y3
430  RETURN
440  IMAGE DDDDX,DDDD
450  END
```

STRUCTURED PROGRAMMING

P.J.Rodman

At one time, when computers were few and far between, and much slower than their modern counterparts, programmers would design their programs to be as fast as possible. This usually resulted in the program being so complicated, and using so many 'tricks' as to make it completely unintelligible to all but the creator, and sometimes even to him ! If the creator then wished to make a few additions or modifications he usually ended up rewriting large parts of the program. This consumed many valuable man-hours, and if anything went wrong with the program after it had been written the creator was the only person who could understand and correct it.

In the present day, however, computer time is becoming cheaper and cheaper, and an inefficient program costs little more than an efficient one if a comparison can be made. The actual programming is now the expensive part of computing. Most man-hours lost in programming are the result of errors (logical rather than syntactical). Methods have been tried to save man-hours by using special programming techniques, one of the most important of these being the concept of structured or disciplined programming.

This technique seems to date from a famous letter (1) by Professor E.W.Dijkstra of the Netherlands. In the letter he claimed that not only was the GOTO , or branch, statement unnecessary, but was, in fact, harmful! (in high level languages). Most of the programmers of the day (and probably the readers of this article) were most puzzled by this, as they had been brought up writing programs with many GOTO's, branching all over the program, both backwards and forwards. However it was soon realised that if the program was written in structured form, GOTO statements would become almost redundant. By using structured programming techniques not only are logical errors or 'bugs' easier to find, but are usually prevented in the first place, thus cutting down enormously on the man-hours and computer time needed for debugging. The program can be easily understood and modified by others, and is self-documenting (i.e. the 'flow' of the program is simple and flowcharts should not be necessary).

The basic concept of structured programming is that only 3 basic control structures are used in the program, namely;

sequential statements (e.g. simple replacements)

conditional statements (e.g. if then else in Algol)

a simple loop structure (e.g. do while in Algol)

These are illustrated in flowchart form in Fig 1.

The rectangular blocks represent sequential statements or one of the other two structures. Thus the structures are 'recursively' defined and can be nested. For example the structure shown in Fig 2 is perfectly valid, and consists of only the 3 basic structures.

Certain points should be noted about such structures. Firstly it can be seen stat the basic structures, and in fact any composite structure, have only one entry point and one exit point. This means that if it is possible to write a program using only the 3 basic structures, then it can be divided into sections, or modules, each of which has one entry and one exit. This not only means that the program can be written in modular form (perhaps with different programmers writing different blocks),but error detection (and prevention) is greatly simplified, as there is no more than one branch out of a module. Putting it another way once control enters a module it can only leave through one point. Therefore tracing an error back through the program to find its cause is a relatively simple process.

Secondly it should be noted that GOTO's become unnecessary, and that there are no backward branches (eliminating most of the notorious infinite loops).

The conditional statement corresponds to the if then else statement in Algol (similarly in PL/1, COBOL etc.). The loop structure corresponds to the do while clause in Algol, IF ELSE in COBOL etc. Unfortunately, languages such as FORTRAN & BASIC do not have these constructs (at present) and it is necessary to use the GOTO statement. However the basic structures can be built up using GOTO's. The concept of structured programming will be preserved as long as the range of the GOTO is within the structure.

The following two conditional statements are equivalent;

Algol	FORTRAN
if A < B then A := B	IF(A.LT.B)GO TO 10
else A := C;	A = C
	GO TO 20
	10 A = B
	20 . . . .

Similarly for the loop structure;

Algol	FORTRAN
while A < 100 do a:=A+X;	10 IF(A.GE.100)GO TO 20
	A = A+X
	GO TO 10
	20 . . . .

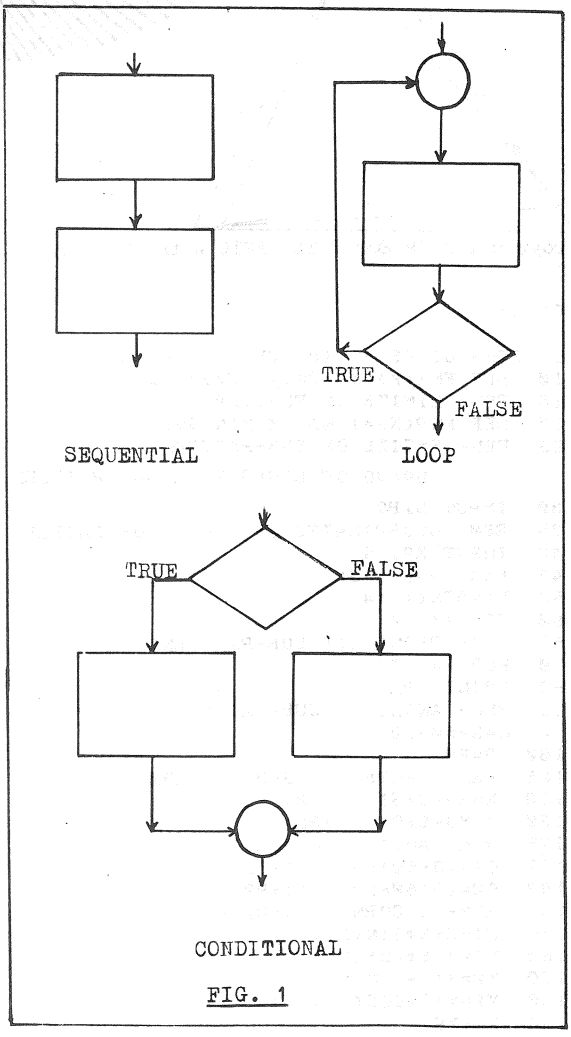


FIG. 1

It is also interesting to note that the FORTRAN IF and DO statements are special cases of the basic structures, e.g.

```
IF(A.LT.B) A = B+1
```
has the structure shown in Fig 3a, and

```
DO 20 I = 1,N
. . . . .
20 CONTINUE
```
has the structure shown in Fig 3b.

Finally it should be noted that it can be proved that all programs can be written using only the above 3 structures (2). It might be a good idea to remove the GOTO ststement altogether from languages such as Algol or COBOL !

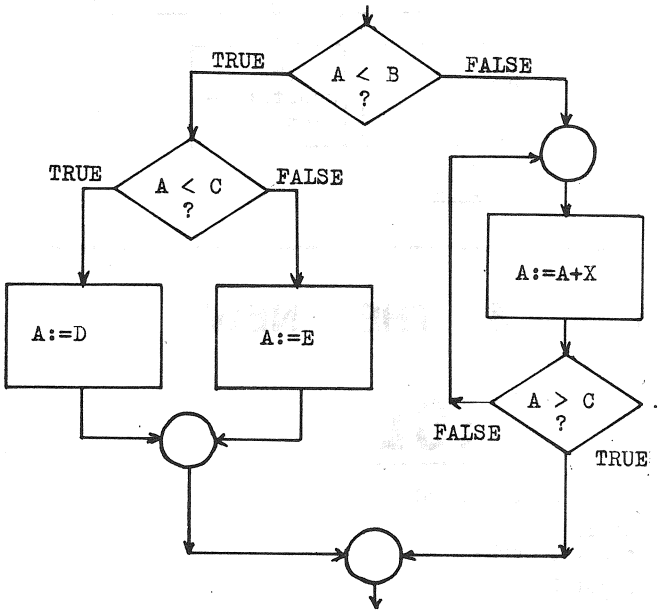


FIG. 2

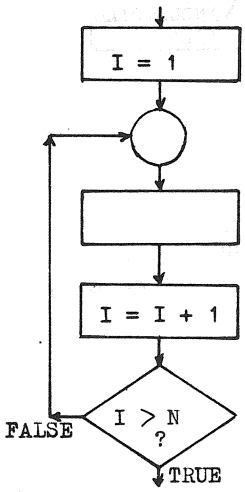


FIG. 3b

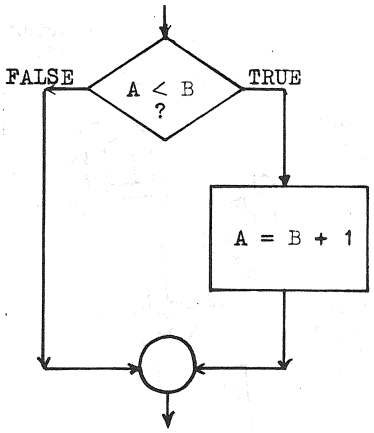


FIG. 3a

Obviously, removing GOTO statements from a program does not necessarily structure it, and to structure an unstructured program is not really worth the effort involved. It is therefore best to start from scratch and use structured techniques throughout the writing of the program. Meaningful and well-placed comments are 'worth their weight in gold' but too many, or badly placed comments only tend to confuse anyone else reading the program.

One should indent nested *if* and *do* *while* loops to make the program more readable.

Meaningful variable names seem to be quite rare in most programs. Even in Algol programs, where up to 30 (or more) characters can be used, one often finds statements such as;

```
if WS < 0 then TS := DS + WS +SIT;
```
which could have been bore descriptively written as;

```
if warehousestock < 0 then totalstocks:=
 depotstocks + warehousestocks + stockintransit;
```

Another important concept of structured programming, used by a large number of experienced programmers, is that of 'top down programming'. Inexperienced programmers tend to attack a programming problem from the 'bottom up'. That is they design the subroutines first and the main program is written to use the subroutines. Even worse they try to write their program as one huge main segment with as few subroutines as possible. This last approach means that for very long programs one can get *do* loops anf *if* statements ,the range of which stretches over several pages. This makes for tedious page turning to follow the program logic.

One should rather start by writing the main program first, such that it is divided into 'macro' statements or blocks, viz. subroutine or procedure calls. One would then proceed to attack each of the subroutines called by the main program in the same manner, and so on, until it is no longer possible to call another subprogram. This is the bottom level of the program and the program is finished once the subroutines have been written. The program now forms a nested, tree-like structure (e.g. Fig 4).

There are several points to be noted about this mode of programming;

(a) All of the blocks are written using structured programming techniques.

(b) Once the main program is written it can be tested by replacing the first level of subroutines by 'program stubs'. These are dummy subroutines which return 'contrived' results designed to aid the debugging of the main program. Once the main program is error-free the first level of subroutines can be written and a second level of program stubs can be used to debug them, and so on, until the bottom level is reached.
This is an extremely efficient way to minimise errors in the development stages.

(c) Many may argue that this is not efficient programming, and there are many unnecessary subroutine calls. This may be so, but the program was probably written in half the time, and had half as many errors as the 'efficient' version Sceptics may find the following quotation of interest (3):
"This reverence for 'machine efficiency' is an interesting phenomenon. Look out of your window at the parking lot. Are there 300 cars there ? Do you realise that that represents about one million dollars of hardware that gets used maybe 40 minutes of the day, and nobody writes memos to the director of transportation, complaining about 'wasted machine time"

Finally, for those who have struggled through the article so far, the following example (represented by Fig 4) may clarify the reader's mind. It shows the first few levels of a program to solve a system of simultaneous linear equations: $\underline{A}\underline{x} = B$. The method used is $\underline{x} = A^{-1}B$. The program is written in an Algol-like language, but should be quite understandable to non-Algol programmers.

REFERENCES:

(1) E.W.Dijkstra "GO TO statement considered harmful", Communications of the ACM. Vol 11 No 3 (March 1968) p147

(2) C.Bohm and G.Jacopini "Flow Diagrams, Turing Machines and Languages with only two Formation Rules". Communications of the ACM. May 1966 p 366.

(3) C.C.Foster, Computer Architecture, Van Nostrand Reinhold, 1970 p166

```
begin
comment  main program;
initialise;
readcoefficientmatrix(A);
readconstants(B);
solve(x);
printresults;
end;

procedure solve(x);
begin;
ainverse := inverse(A);
x := multiply(ainverse,B);
end;

procedure inverse(A);
begin;
gauss;
backsubstitution;
end;
```

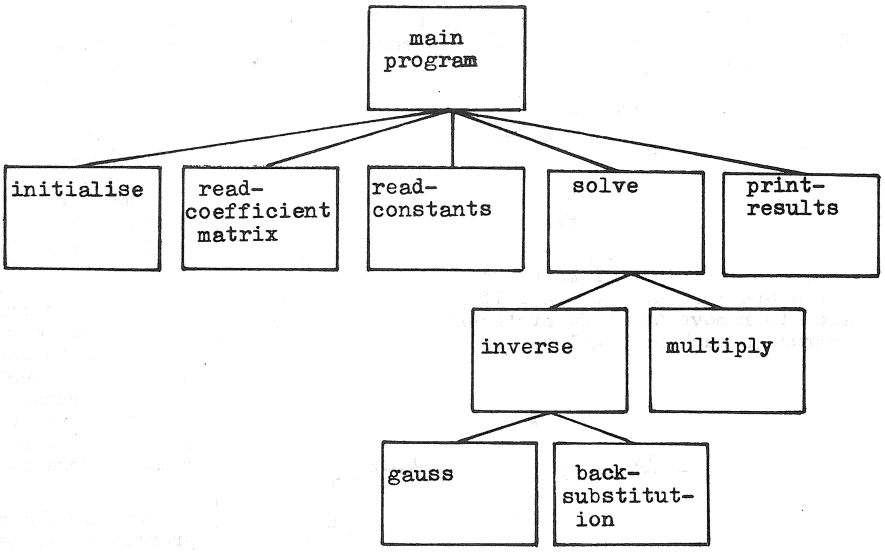


FIG. 4

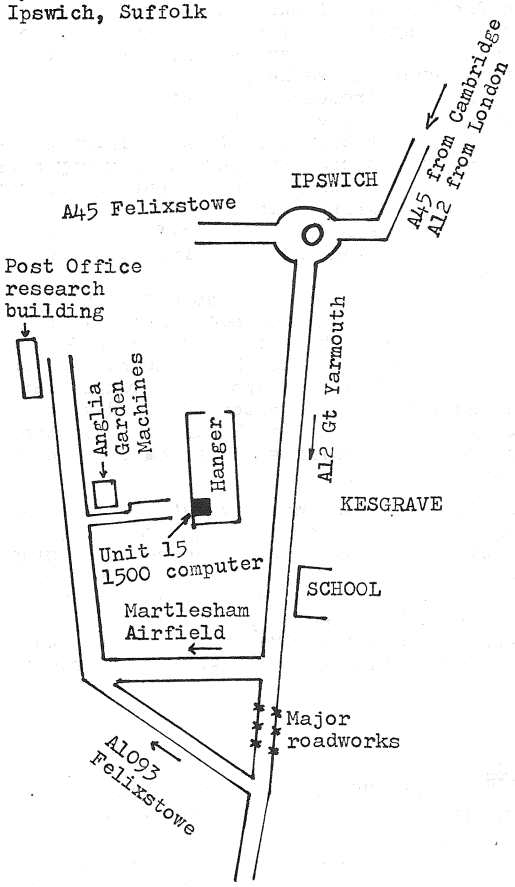
THE OLD —— MEETINGS —— & THE NEW

1500 COMPUTER CENTRE

See Jon Aslet's ICL 1500

Saturday July ~~12~~ 19 2.30 pm

15 Martlesham Heath
Ipswich, Suffolk



Configuration;

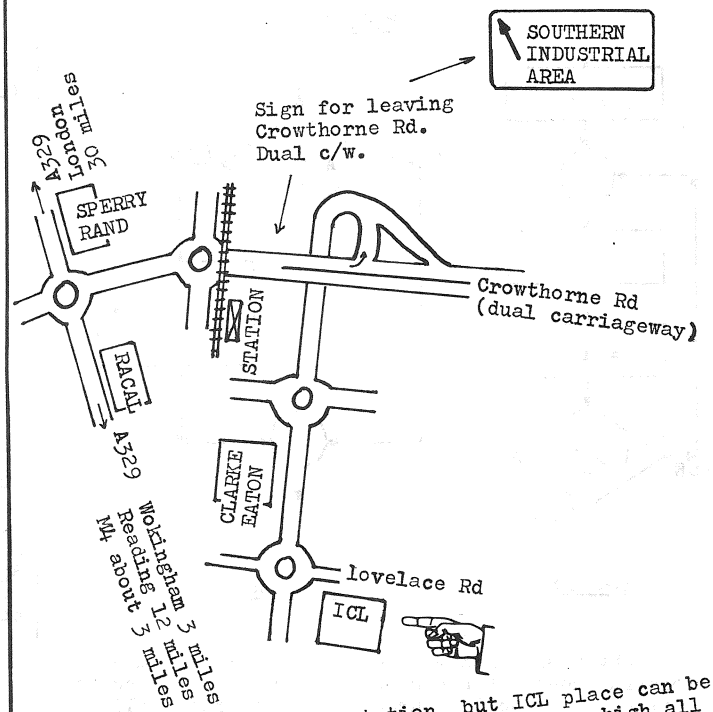
ICL 1500 7 bit processor with 20/40 K core
8 decks 10kC
1000 cps ptr
1000 lpm printer
10 cps monitor printer
Software, FORTRAN, COBOL, RPG etc.

ICL

Visit to ICL, All welcome

Lovelace Rd,
Bracknell, Berks

9 August 1975 2.30 pm



About one mile from station, but ICL place can be seen from long distance, some 12 floors high all on its own with large sign at top !

ACC COMMITTEE MEETING

Immediately following the Open Day at Jon Aslet's 1500 Computer Centre on July ~~12~~ 19

LETTERS

I have been meaning to write for several months and have finally managed to put pen to paper as it were.

Firstly I would like to recommend International Electronics Unlimited. I sent off an order via a friend who was visiting the States and half expected to hear nothing more but three weeks later the parcel arrived, registered post and with no duty to pay. Further they will accept ACCESS credit card payment solving the problem of sending over US dollars.

Next may I recommend 101 BASIC COMPUTER GAMES edited by David H Ahl and published by DEC. It lists in BASIC all the well known computer games and some less well known ones. (£3 from 'Software Services Dept., Digital Equipment Co. Ltd., Fountain House, Butts Centre, Reading RG1 7QN; ed)

I am engaged in a project to rebuild, using integrated circuits, the Digital Electronic Universal Calculating Engine, DEUCE, as built in the mid fifties by the then English Electric Company. The DEUCE was a direct descendant of the Pilot ACE built at the National Physical Laboratory and now preserved in the London Science Museum. The purpose of the project is to exactly duplicate the workings of DEUCE in order that the features of its design will not be lost. I have available a very comprehensive library of documents relating to the DEUCE thanks to the kindness of Professor Gilles of Glasgow University and Fraser Duncan of Bristol University, my former tutor, who first suggested this project. If anyone is interested in any aspect of such a project or has any comments to make I should be glad to hear from them.

H.Pufal
17C Stuart House, Burns Rd., North Carbrian, Cumbernauld G67 2AN

PUNCH

" Tape Punching Assembly 9 punch unit for 1" tape. 8 trip coils thought to be 90V DC. containing 7 ballraces,levers, contacts etc. £3 Wt 2kg "

I thought this might interest some members, Its from the catalogue of;

K.R.Whiston, New Mills, Stockport SK12 4PT

I haven't seen them but Whiston's shop is often good value.

Secondly Lock Distribution, Neville Street, Oldham OL9 6LF sent me the data on the MOSTEK MK 5065P 8 bit microprocessor. Price quoted for one off was £52. It looks a promising bit of hardware with 51 basic instructions and ability to address 32k x 8 bits of memory.

Tony Cassera

K.Whiston, New Mills, Stockport SK12 4PT are selling some surplus Flexowriter punches (8 hole) at £3 + 75p p&p (VAT included), catalogue No 3009. You require motor with electromagnetic clutch and driving electronics (90V coils).The one I have seems to be in good condition.

Part of my work involves development of numeric algorithms for use on our NOVA 1220 mini. I should be pleased to attempt to provide ACC members with any assistance they may require on these topics.

C.Doran
89 Lennard Rd., Penge, London SE20 7LY

DISPLAY

I am very interested in displays from computers and processors such as Video Display Units and I would be grateful if anyone could help me by providing some data or information on practical units or perhaps simple humble CRO ideas for data display

M.Hewitt
Freemen's Hall, Castle Leazes, Newcastle on Tyne

HELP

I am trying to determine if there are any IBM 1620 machines in existence in the UK. As far as I am aware, the 1620 was produced between 1956 and 1961, and came with various options including disks and core sizes up to 60K characters. This model is technically vary interesting because its arithmetic was organised around a variable field decimal philosophy. Thus arithmetic operations were performed by a table look-up method, with each digit being stored in successive locations, and it was possible to operate on two numbers each up to 30,000 digits long with just one machine instruction.

The era of decimal computers is now a long time gone, but should not be forgotten, and thus I am eager to restore one of these machines. If any readers know of any remaining 1620's I would be happy to hear from them.

Brian Anderson
Smith, Kline & French Laboratories, Herts

WANTED

Flexowriter or similar device, preferably with paper tape reader and punch. Any condition considered.

Bill Hughes Tel. 0376 512469
47 Collingwood Road, Witham, Essex CM8 2DZ

Anyone willing to sell me a complete set of the Algol 60 ICL Students Manual ?

P.Rutherford
Rush Common House, Abingdon, Oxon OX14 2AJ

SALE!

8K x 18 Mullard Core (type AW3805). This consists of a core stack capable of operating at 1uS cycle time, in a metal case with sockets for connections to the core. Complete with manufacturers data etc. £65. Also Vero Card Frame, type 3C/4UC1/D2W1/MR4/L2G2/.5 new and in manufacturer's packing. Originally intended to be used with the above.£12.50 or £70 the pair

M.Reeve
6 Limes Ave N.Finchley London N12 8QN

Two 64 x 64 Mullard memory planes for sale.

M.Davison
26 Forester Ave, Bathwick, Bath, Avon

LIBRARY

An ACC library is to be set up for members' use. It will cover both hardware & software documentation. If you have any suitable manuals or literature which you would like to donate to the ACC library I would be grateful if you would contact be at the address below.

We are also trying to assemble a collection of games for general use, if you have games or other amusing programs which you would like other people to have please send the source either on cards or paper tape, again to me at the address below.

In the next issue of the newsletter we will be publishing a catalogue of the documentation and games in the ACC library.

Fred Doherty-Bullock
99 Overstrand Mansions
Prince of Wales Drive
London SW11

THE WEENY-BITTER

PROGRESS REPORT

Seem to be getting on well, main problem is expandability. How to design something which is useful at £50 but allows for future growth in one way or another. This problem shows up particularly in devising addressing schemes; an 8 bit word is OK for a 256 word machine as you can address any word in memory with 8 bits, whereas more bits (if only for 'page' or 'mode' are required for a larger store - and these bits are wasted in the small machine. We have to choose between an efficient 256 word machine which can't grow and a less efficient one which can. One idea briefly discussed at one ACC meeting was to design an efficient 256 word machine which could later be connected onto a larger memory and used as a micro-programmed CPU to generate a more powerful instruction set.

An efficient micro machine could have a fairly powerful set of one word instructions to save memory by using 3 or 4 bits to refer to registers or as some form of short address.

Alternatively, it might be worth going to a 12 or even 16 bit word length serial machine. This would allow expansion to 4 or 64k words of memory without introducing the complication of multiple word addresses, which complicate the control section of the processor quite a lot, especially if sophisticated addressing modes are used.

Anyway, to get down to the nitty-gritty of it all, I've summarised the discussion so far below;

PERIPHERALS

"It seems to me that it is wise to build as many 'useful' peripherals as possible before actually committing yourself to a CPU . . . eg some sort of decimal display . . add an A/D converter . . a more complex control system in order to operate correctly & this is where the computer 'finally' comes in " (P Maddison)

" Are we building a calculator or a domestic robot ? . . a game & calculations machine needs an (alpha)numeric keyboard & LED display of several digits. A 'domestic' robot needs a clock & relays to switch domestic appliances" (S Thompson)

"Let's, fairly early on, define the CPU-peripheral interface well enough so work on CPU & peripherals can proceed in parallel" (ACC meeting)

COST

"I would have thought that £50 was a little optimistic . . 256 words only a serious design limitation " (J Howells)

"Bi-Pak sell MSI TTL in 'reject' paks of 5 for 54p or so. In my experience 30% are completely useless, 30% only part functional but I can't find anything wrong with the other 40%. My remarks apply in particular to 7489, 7483, 74198, 74191. Bi-Pak advertise a smaller range of 'rejects' than they actually supply" (P Davies)

"some of my collegues have been surprised at the £50 . . . incredible TTL & CMOS offer in 'Electronics Today' June 75" (S Thompson)

"Lock Distribution are selling Signetics 2606 RAM, 256x4, 750nS @ £3 1 off." (M Anderson)

DESIGNS

A) J Bibby

ACC System 1

Why do you need indirect addressing when you can address 65K of store directly? as I'm sure that no member building his own could afford to buy even 65K of store !

Why restrict IO to reg 0. If they were treated as memory and a double handshake system used they would not need their own instruction set, the machine would be simpler to build and it would work with any speed of store or peripheral.

Why waste time in IDL mode, why not ARM and DISARM interrupt enables. When an interrupt occurs it will cause jump to a (hardwired address) pre-specified address and perform 16 instructions before it automatically recurses from where it left off unless it performs a jump instruction as one of the 16. This will mean having an interrupt program counter register of 4 bits probably on the interrupt board. The interrupt instructions could be a simple routine or instructions to save the return address and contents of the registers and to jump to a longer subroutine.

JMS could be a pseudo instruction;

load reg with program counter
jump to start of subroutine
add jump +1 to reg
save as last instruction of subroutine.
.....
jump +1 + contents of program counter

Suggestion for a 16 bit 8+8 simple processor

16 bit word accessing up to 7K

JMP	jump unconditional	01N
AND	mask A with (N)	10N
INC	increment (N)	11N
LDA	(N) A	02N
STA	(A) N	04N
LIA	(N+I) A	03N
SIA	(A) N + I	05N
LDI	(N) I	13N
LDM	(N) M	12N
STI	(I) N	15N
STM	(M) N	14N
ADD	add (N) to A	06N
SUB	subtract (N) from A	07N
MUL	multiply M by (N)	16N
DIV	divide AM by (N)	17N
ADI	add N to A	00ON
HLT	stop	000000
NOP	no operation	002000

001 skip instructions
002 housekeeping instructions
003 " and also clear K
004 shift instructions
005 rotate instructions
007 inter register instructions
007 interrupt instructions

A = the accumulator register	16 bit
M = the multiplier register	16 bit
I = the index register	16 bit
C = the control sequencer	12 bit
N = a core store address	12 bit
K = the carry store	1 bit
X = a switch register	16 bit

Front Panel Controls

Reset Obey x Resume Start Stop
Single instr/cont

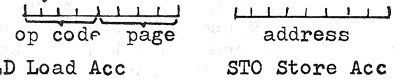
Core store up to 6K, peripherals on the same highway all with locations 7000 to 7777

B) M Anderson

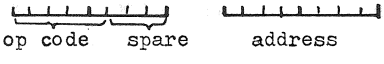
For a small memory of ¼ to 4 K, I think that the instruction set suggested by Mr. Reeve would be rather inefficient, each instruction would occupy more memory space than necessary, and would contain redundant bits. Also, the provision of so many general purpose registers would make the machine expensive.

My own suggestion would be a machine with one accumulator, and an 8 bit word length. Memory size would be up to 4K bytes and instructions one or two bytes long.

I) Load, Store & Arithmetic instructions

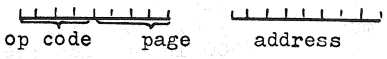


LD Load Acc STO Store Acc

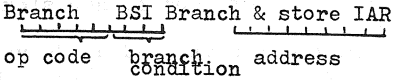


ADD Add to Acc AND And with Acc
SUB Sub from Acc OR OR with Acc
ADC Add with carry EOR Exclusive OR

II) Branch Instructions



B Branch BSI Branch & store IAR (PC)



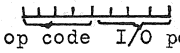
BC Branch on condition

Condition Codes;

Bit 1 0 Branch on condidion false
       1 "   "   "   true

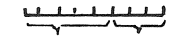
Bits 2&3 00 Acc negative
          01 Acc zero
          10 Acc odd
          11 Carry bit on

III) Input / Output



IN input to Acc
OUT output from Acc

IV) Miscellaneous



HLT Halt
SLA Shift Acc left through carry bit
SRA Shift Acc right through carry bit

When an instruction does not contain a page field, the current page is assumed . (ie bits 8-11 of the IAR (or PC) form the page address)

C) R Kirkby
    A Cheap Amateur Computer

This design is largely inspired by the article 'Instruction Set For An Amateur Computer' in last December's Newsletter. I have received other inspirations, but from so many sources that I won't bother to enumerate them, I'm too frightened of leaving someone out.

The machine is a single accumulator 8 bit word system and is intended to be as cheap as possible, and at the same time available for expansion. All of the present instructions require two words, so uprating to 16 bit words should be fairly simple. The instruction set is at present fairly small, but could be eleborated upon. Instructions are at present executed by means of simple microprogramming which is slow but I can envisage a quick change to 'handshaking' in a crude way, which is fast. It should not be too dear either.

M/C Instruction Set

If input and output devices are disguised as dummy store locations, a working instruction set can be devised of only three classes of instruction;

Arithmetic where the accumulator and specified data are operated on in suitable (requested) manner, and the result placed in the accumulator. ie ACCUMULATOR := ACCUMULATOR * OTHER DATA

SET where the accumulator and specified data are compared in a suitable (requested) manner, and the result of the comparison used to affect the state of a one bit register.

STORE IF where if the one bit register mentioned above is in one state the instruction is ignored and if in the other state the word contained in the accumulator is placed in the specified store location. The one bit register may be affected during this instruction of required .

It seemed to me that creating cummy store would be more costly than including an 'input' and an 'output' instruction pair, so;

INPUT IF where the input device's word is placed in the specified store location.

OUTPUT IF where the word in the accumulator is read to the output device.

Most decisions are taken and used only once, so for convenience the STORE IF instruction resets the one bit decision store to the 'store' state whatever its previous state. Because of this the SET instruction sets the register to the 'do not store' state if the test is unsuccesful, but, to enable a little more flexibility in programming if successful the register is left alone. Several variations on this are possible.

I amgreatly taken with the 74181 as an arithmetic unit and decision maker, mainly due to its present low price (1.98). It allows over 30 varieties of logical and arithmetic operations, and all six comparisons, to be performed more or less automatically, and at rather less than the cost of separate adders and comparators. Therefore I have moulded the actual instruction set around the 74181 instruction set.

INSTRUCTION SET

I wish to keep the use of 8 bits of instruction as suggested in December (ie dividing the 16 bit word into 8 bits of address, two bits of address mode control and six bits of instruction) to allow for subsequent expansion. The 74181 has six bits of input that could be conceived as defining its function, to whit 4 bits stating function, 1 bit defining mode- either arithmetic or logical- and 1 bit of last carry input. This would use up all of my six bits available, except for two details; some of the functions thus produced are duplications of others (could be useful at a later date to add wired-in multiplication for example), and when the device is used in logical mode the carry in bits are superfluous. Using one instruction word bit for each input to define the operation in ACCUMULATOR:= ACCUMULATOR * OTHER DATA say allocating the bits thus;

Instr reg bit 0 - mode definition
" " " 1 carry in bit
" " " 2 function bit So
" " " 3 " " S_1
" " " 4 " " S_2
" " " 5 " " S_3
" " " 6&7 not used at this time

then if IR bit 0 = 1 (so that a logical function is defined) then IR bit 1 could be used to tell the difference between an 'arithmetic' instruction and one of the others. Say if IR bit 0 = 1 and IR bit 1 = 0 then the instruction is 'arithmetic', if IR bit 0 = 1 and IR bit 1 = 1 then the instruction is something else. The 'something else' could be one of 16 choices - more than enough for even the most exotic of this type of machine (I can only think of 13 varieties of instruction on a good day)

IR bits						
0	1	2	3	4	5	
0	x	x	x	x	x	Arithmetic operation
1	0	x	x	x	x	Logical operatio
1	1	0	x	x	x	SET if
1	1	1	0	0	0	STORE IF
1	1	1	0	0	1	INPUT IF
1	1	1	0	1	0	OUTPUT IF

TO BE CONTINUED . . . .

These two pages can, of course, contain only a fraction of the ideas, suggestions, comments received over the last two months - thanks too all those who have contributed - and keep it going, we're still sticking too our schedule and aim to define the machine language & basic hardware in the next newsletter.

?

HIGHLIGHTS FROM THE QUESTIONNAIRE RESULTS

Taken from a preliminary analysis of 98 forms (208 sent out, about 110 eventually returned)

Age: 44% 18 to 25, 42% 26 to 45

Occupation: A very wide spread, 38% classified themselves as Academic or Student, 17% were 'computer engineering' or 'programming'.

What do you expect to gain from ACC membership ?: 54% gave 'education' as their first choice, 29% said 'enjoyment'

What do you find of most interest in the newsletter ?: First choices were 52% hardware, 24% software.

91% said that they were interested in the ACC low cost processor, 32% even said that they would build it.

47% said that they would be prepared to contribute to the paper design of a sophisticated hypothetical machine.

Still trying to find a way of interpreting the results of the last question (How good is the ACC valued as a score between 0 and a maximum of 9 for the following subscription rates) but the raw data is;

percentage replies for each category;

score:	0	1	2	3	4	5	6	7	8	9
£1.00	7	2	1	0	2	4	1	5	9	69
£1.50	7	0	1	1	0	5	7	8	21	49
£2.00	9	0	0	3	3	9	5	20	16	34
£2.50	10	2	3	5	6	10	13	16	16	17
£3.00	16	8	4	6	5	15	10	13	13	8
£3.50	28	6	8	4	14	8	12	7	10	2
£4.00	35	8	6	17	9	7	4	9	2	2
£4.50	46	4	21	5	5	5	5	5	2	1
£5.00	53	17	3	7	3	8	3	2	1	2

note that in each case results are percentage of total number of questionnaires returned.

Following typing up the questionnaire sheets for input to my 1500 at Martlesham I award Nigel Freestone my prize of 1 hour free computer time for providing the best filled in and neatly done form. The Booby Prize of 10 hours free computer time goes to G.B.Lane who confused me and the computer two times with a mislaidout address causing the format to blow out. (Sorry - only joking, my thanks to all those who returned their forms even from Germany and South Africa and I hope to extract the most useful information for the club committee to help provide the type of club you want.)

J Aslett

TIME

HOW TO GO ABOUT GETTING IT

If ACC members or others get together and form into a group of about 10 and go to the local college with computer facilities and enroll themselves as a course (appointing one of their number as the tutor) and for the small charge of a night class the computer can be used by the course members as much as they like.

SORRY

List of ACC committee members published in V3 I1 should have read;

I.Richardson
T.Jones
M.Reeve
R.Kirkby

BIGGIES

Control Data has announced a mass storage system with a maximum capacity of 16000 million bytes, equivalent to about 6400 large tape reels, makes our 256 word processor look rather sick.

An array processor developed at University College, London, uses 9216 interlinked microprocessor cells to process complex black & white images.

CAN YOU READ THIS ??

Trying an experiment this month to get more information onto the same number of sheets - the original artwork for this issue was on A3 sheets, giving a linear reduction by /2 , so in other words these 8 pages started out as 16. Let me know if you find the small print troublesome.

mike lord

BOOKLIST

COMPUTERS, COMMUNICATIONS & SOCIETY
M.Laver £3.50 Oxford University Press

ROBOTICS
J.F.Young 1973 £6.00 Newnes-Butterworth

INTRODUCTION TO COMPUTER LOGIC
Nagle, Carrol & Irwin £9.90 Prentice/Hall

INFORMATION-LOSSLESS AUTOMATA OF FINITE ORDER
A A Kurmit £9.80 John Wiley

COMPUTERS, CHESS AND LONG-RANGE PLANNING
M.Botvinnik, translated by A.Brown
1970 Springer-Verlag Inc

Intel price cuts

Extensive price reductions have been made to most of INTEL's product range.

Examples;

microcomputer kits;

MCS-4A	£42.70
MCS-40A	£45.00
MCS-8A	£61.80
MCS-80A	£175.00
MCS-80B	£103.00

microcumputer chips;

C4004	4 bit CPU	£13.48
C4040	4 bit CPU	£14.23
C8008	8 bit CPU (20uS cycle)	£20.22
C8008-1	8 bit CPU (12.5uS cycle)	£24.71
C8080	8 bit CPU (2uS cycle)	£74.00

MOS memories;

P1103	1024x1 dynamic RAM	£6.74
C1402A	Quad 256 bit dynamic SR	£5.07
P2101	256x4 static RAM (1uS)	£3.04
P2102	1024x1 static RAM (1uS)	£3.04
C2107b-6	4096x1 dynamic RAM (.35uS)	£10.80

one-off recommended retail prices w/o VAT

from; **Rapid Recall Limited**
9 Betterton Street Drury Lane
London WC2H 9BS. Tel: 01-379 6741.

AMATEUR COMPUTER CLUB NEWSLETTER
Vol 3 Iss 2 June '75

M.Lord
7 Dordells, Basildon, Essex
tel; 0268 411125 (home)
0268 3040 x 117 (work)