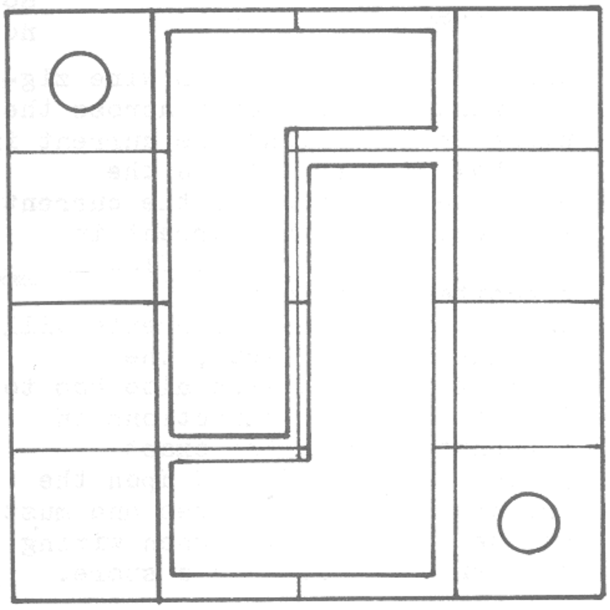
ACCN AMATEUR COMPUTER CLUB NEWSLETTER VOL 1 ISS 4 NOV 1973

THE L GAME

for two players. from S.Thompson



16 squares on the board (4x4).
2 L shaped pieces each occupying 4 squares.
2 neutral pieces each occupying 1 square.
Starting position as shown on right;

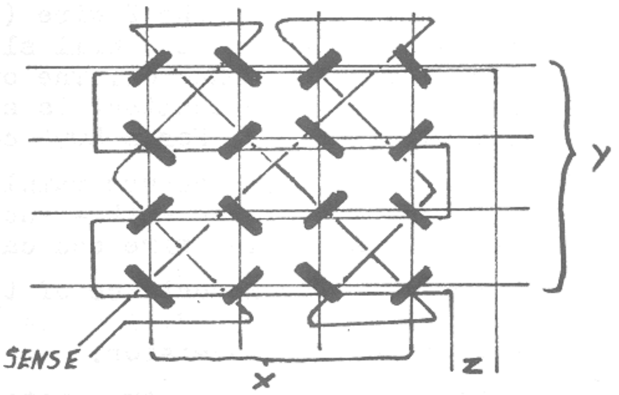
The object of the game is to force the opponent into a position from which he cannot move. Each player takes a turn to move his L piece to a new position. The L piece may be rotated or turned right over. So long as at least one square has been changed a new position exists.

After the L piece is moved the player may - if he wishes - move ONE of the neutral pieces. Clearly any two pieces must not occupy one square at the same time.

For a computer to play it must know these rules and recognise the playing pieces. It could develop its talents by storing the moves of each game played, whether lost or won. After the first few games it should then have sufficient data to rely less on random moves. Anyone clever enough to write a suitable program ?

CORES FOR STORES PART 3

3D co-incedent current core stores are wired for one plane per bit. Thus a 4096 word store - 8 bits per word - would have 8 planes, or layers, of cores, each plane containing 64 x 64 cores. Each core will have 4 wires through it; the X,Y & Z drive wires and the sense wire.

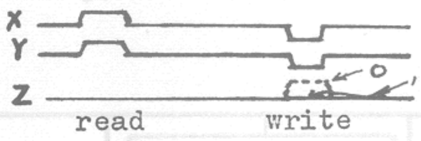


The wiring on an individual plane is usually as shown in the example on the right, although the sense wire is sometimes wired to a differrent pattern.
The reason for wiring the sense wire in such a complicated way is, of course, to reduce the stray coupling to it from currents in the X,Y & Z wires.

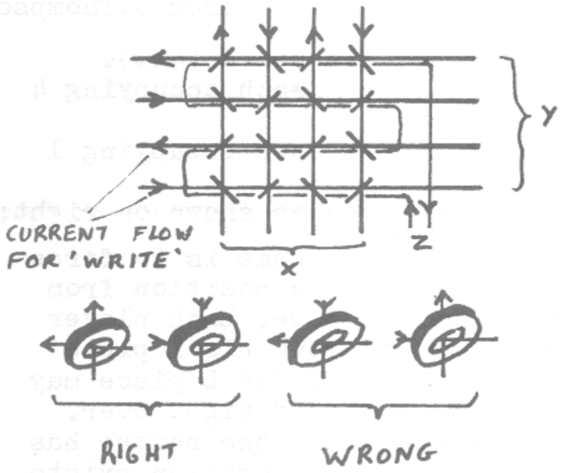
Corresponding X wires in each plane will be wired in series, as will the Y wires. One Z wire is provided for each plane and threads all of the cores in that plane - as does the sense wire. For a 4096 x 8 store we would have 64 X wires, 64 Y wires and 8 Z wires (also 8 sense wires, but we're not concerned with these yet). Passing the correct amount of current through one X wire and one Y wire will therefore select one core from each plane - thus selecting one word of n bits (n planes).

This is OK for a 'read', where we want to attempt to switch all the cores of one word to a '0' (so that we can see whether they were in a '1' state), but when we reverse the X&Y currents to write information back into the store we only want to affect the cores on those planes corresponding to a '1' in the word we are writing into the store.

The Z wire is, of course, used to do this. If at the same time as the selected X & Y wires are energised a current is passed through a Z wire equal in value to the current in the Y wire but in the opposite direction, it will cancel the effect of the Y current so the core in that particular plane will not be switched.

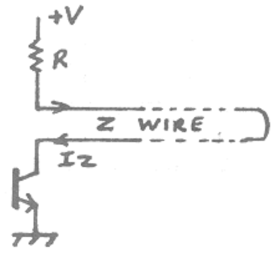


Note that because the Z wire zig-zags backwards and forwards across the plane then, so that the current in it shall always flow in the opposite direction to the current in the Y wire, the current in alternate Y wires must flow in opposite directions.
So that the X and Y currents will combine their effects, the current in the X wire also has to flow in opposite directions in alternate wires, the exact directions will depend upon the orientation of the cores and must be checked carefully when wiring the drive circuits to a store.



Z wire drivers :

These are easy. We just have to push a fairly well defined current through each Z wire during the 'write' time IF we want to write a '0' for that particular bit.



If we turn the transistor on hard so that it saturates, its collector-emitter voltage will be less than say 0.5 V so the current is defined by V and R.

$$I_z = \frac{V - V_{ce\,sat}}{R + R_z}$$ where Rz is the resistance of the Z wire and is usually about 1 ohm.

Problems :

a) The inductance of the Z wire (and remember that it has many ferrite cores threaded on it) will slow down the rise of current when the transistor is turned on. The only solution in to make V, and so R, large so that the time constant is short. (the time constant for an L-R circuit is L/R). For a 4096 core plane, V should be at least 10 V.

b) Dissipation. This occurs mainly in the resistor and is a fact of life which must be lived with - incedentally, wire wound resistors having many turns of fine wire can cause problems due to their inductance.

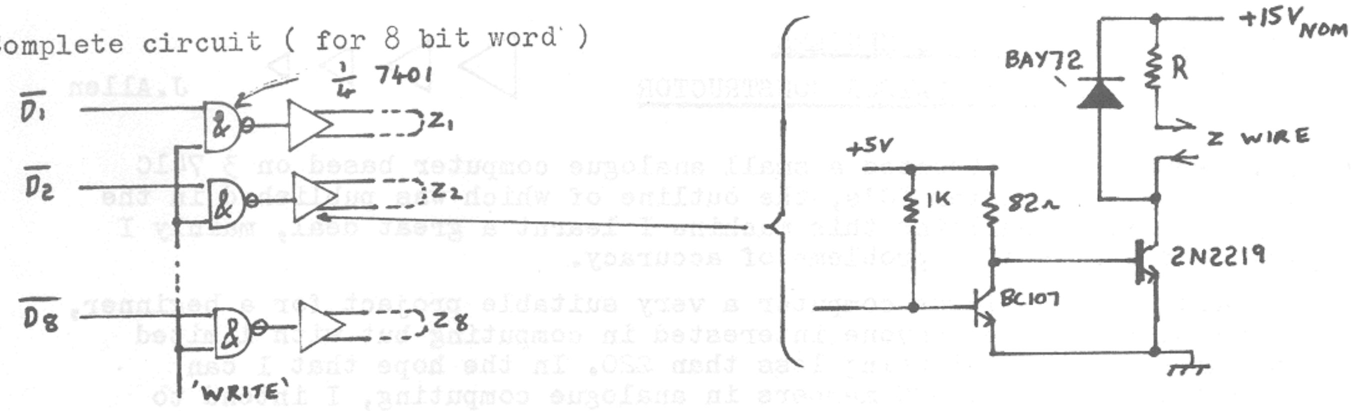
c) Overshoot. The inductance of the Z wire will cause a 'ring' when the transistor turns off. A diode clamp should be added to prevent damage to the transistor.

d) Switching times. The transistor will take a finite time to turn on and off. This can be embarassing unless a high speed transistor is used and the drive circuits carefully designed.

e) Current pulses. Each Z wire driver will be switching a current of the order of ½ A in less than one microsecond. This will give troublesome voltage spikes on power supply leads unless great care is taken with layout and decoupling.

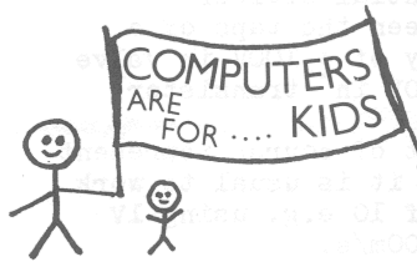
+ SUPPORT THE ACCN +
THE NEWSLETTER DEPENDS ON YOU
MEMBERS, DATA AND ARTICLES WANTED

Complete circuit (for 8 bit word)



Data to be written is presented to the 7401 open collector NAND gates in inverted form -'low'to write a 'I' and 'high' to write a '0' -on leads $\overline{D_1}$ to $\overline{D_8}$ (the ‾ 's denote the inverted form). Current will then flow in the selected Z wires when the 'write' lead is 1 ('high'). Other semiconductors than those shown can be used, the only critical requirements are reasonably high switching speeds (better than $\frac{1}{4}$ uS) and sufficient current carrying capacity.

To avoid noise problems a massive earth plane should be used - easy if you are making your own PCB, if not then 'Cirkit' sell adhesive backed copper sheet, - 100uF solid tantalum and 0.01 uF ceramic capacitors should be liberally used to decouple the 15V supply. and the Z wire connections to the core stack should be made with tightly twisted pairs.



is the motto of the PEOPLE'S COMPUTER COMPANY, a newspaper published 5 times a school year. It is about 'People, computers & a programming language called BASIC'.

Printed in a free style format replete with stars and wierd graphics, it is fun to read (and instructive). A must for teachers or anyone who dares think computers could be fun.

$ 5 for one school year(either 1973/4 or back issues from the 1972/3 school year, please specify which) from " People's Computer Company, c/o DYMAX, PO Box 310, Manlo Park, Ca. 94025 USA

+ +

By contrast another American publication ; POPULAR COMPUTING is very formal. Almost entirely concerned with numerical calculation methods (in one issue a third of one page is devoted to printing $(1+\sqrt{2})^{1024}$ to 840 decimal places).
Published monthly from Box 272, Calabasas, California 91302
Subscription is $ 16.50 for 12 issues.

+ +

Also from the USA is the Amateur Computer Society. Run on similar lines to the ACC (not surprising as the ACC was in fact modelled on the ACS) it is open to all who are interested in building a digital computer that can at least perform automatic multiplication and division or is of a comparable complexity.
For membership and a subscription of at least eight issues of Vol 3 of the newsletter send $5 to; Stephen B Gray, ACS, 260 Noroton Ave., Darien, Conn. 06820. The newsletter appears every two months or so.

Note; Subsciption rates quoted for all three publications are for people living outside of the USA.

ANALOGUE COMPUTERS
For the AMATEUR CONSTRUCTOR

J.Allen

Last summer I constructed a small analogue computer based on 3 741C operational amplifier IC's, the outline of which was published in the last issue. In building this machine I learnt a great deal, mainly I admit connected with problems of accuracy.

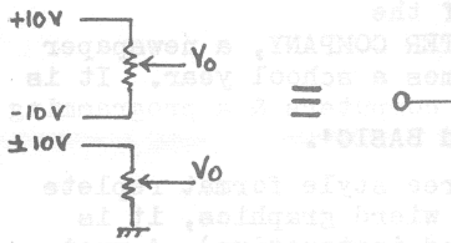
I consider an analogue computer a very suitable project for a beginner, a school society or anyone interested in computing but with limited resources - the cost being less than £20. In the hope that I can interest some other ACC members in analogue computing, I intend to give here some more general information on this machine in particular and on analogue computers in general.

Part 1 : Basic linear computing circuits

The principles of analogue computing are probably well known to a large proportion of ACC members, but since this article is aimed at those new to the field I shall first summarize the standard circuits used for computing, treating them first as ideal circuits, and then discussing the major sources of error.

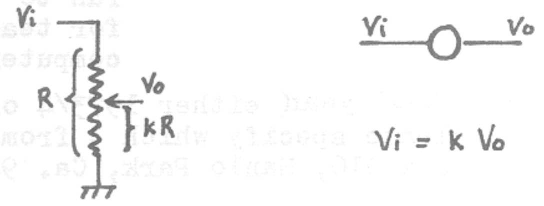
Standard Circuits

In most analogue computers the quantities are dealt with as voltages in the various parts of the machine. The first consideration must therefore be for the production of a known voltage on which we may operate. This is simply achieved by using a potential divider connected between the taps of a standard supply of ± 100V in valve models and ± 10V in transistor versions.



1 volt need not of course represent 1 unit, though it is usual to work with factors of 10 e.g. using 1V to represent 100m/s.

By a similar device, known as a computing potentiometer, we can multiply any given voltage by a positive constant of less than one ; by suitable scaling this can be extended to cover any positive constant.

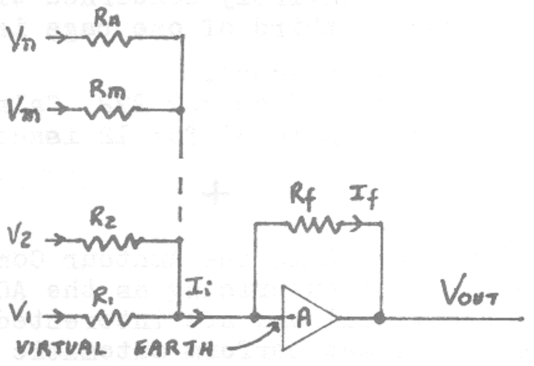


$$V_i = k V_o \quad (0<k<1)$$

We now come to the first of the active devices, in which the principal component is a high gain (about 100,000) inverting amplifier such as the 741C.

This circuit scales each of the input voltages by a constant less than zero (negative) and adds them together.

Since the output voltage of the amplifier is within 10 or so volts of zero and the gain is very large, the amplifier input voltage must be negligibly small, hence the name 'virtual earth'.

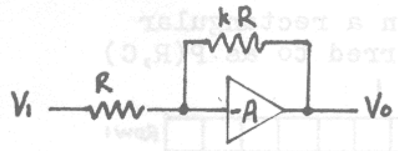


Furthermore the input of the amplifier is high impedance and may be assumed to draw no current, hence

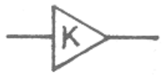
$$I_f = I_i \;; \quad -\frac{V_{out}}{R_f} = I_i = \sum_{m=0}^{n} I_m = \sum_{m=0}^{n} \frac{V_m}{R_m}$$

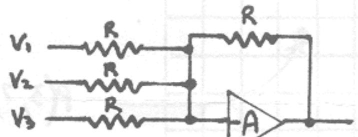
$$\text{Thus } V_{out} = -R_f \sum_{m=0}^{n} \frac{V_m}{R_m}$$

There are several common types of this general case;

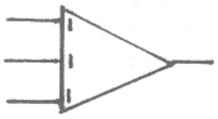


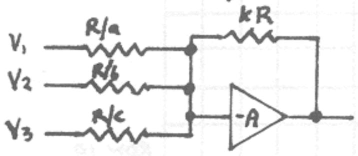
$$V_o = -k\,V_i$$



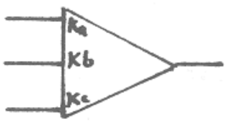


$$V_o = -(V_1 + V_2 + V_3)$$

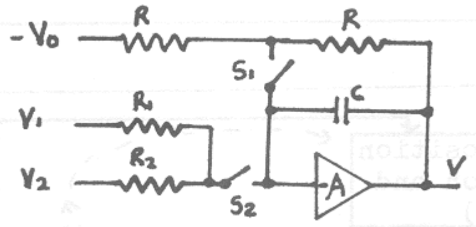


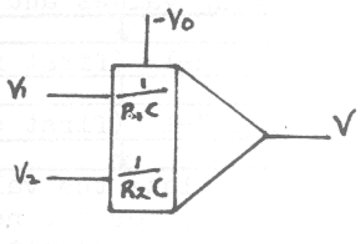


$$V_o = -k(aV_1 + bV_2 + cV_3)$$



More significantly, we can perform integration with respect to time using the following circuit;





after a time t sec ; $V = -\frac{1}{C}\int_0^t \left(\frac{V_1}{R_1} + \frac{V_2}{R_2}\right) dt + V_o$ where V_o is the value of V at time t=0

since $V = -\frac{1}{C}\int_0^t I_f \, dt$

When S1 is closed and S2 open, V becomes equal to V_o after a short time.

if then S1 is opened and S2 closed, the circuit integrates the weighted sum of the input voltages according to the equation above. If s2 is then opened after a time t seconds the value of V at time t is held indefinately.

The above circuits can be combined in the same way as the operators in an algebraic equation to solve sets of linear equations or to provide particular dynamic solutions to linear differential equations with constant coefficients in time.

Next issue ; Non linear circuits & sources of error.

The BRITISH AMATEUR ELECTRONICS CLUB is thinking of building a digital computer as a club project (probably in S.Wales). An introductory article was published in their last Newsletter. For information on the computer and the club contact;
Mr. C.Bogod
'Dickens' 26 Forrest Road,
Penarth, Glamorganshire.

GUESS WHAT ? ?
(clue - it is a North American device)

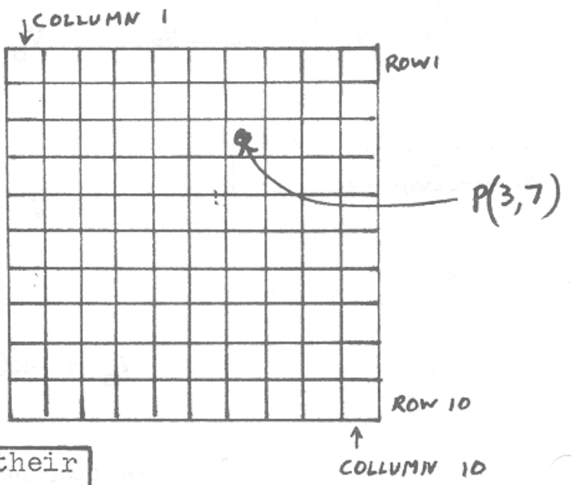
P5

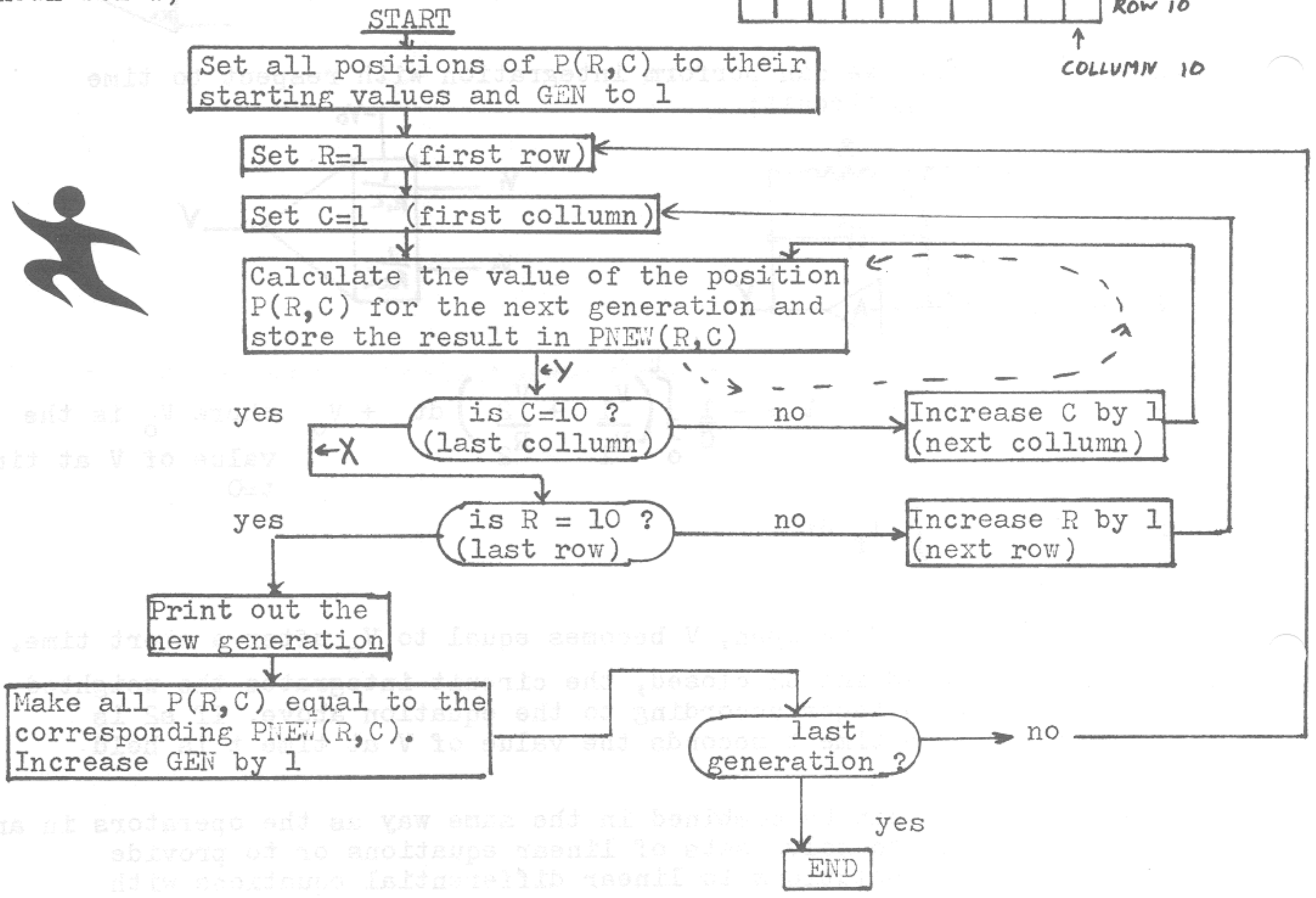
PLAYING THE GAME OF L!FE !!!!!!!

The following is a distillate of various programs, suggestions etc. contributed by several people. Thanks to them all and may their loops never be endless.

THE BASIC SOLUTION says that the cells can exist on a rectangular (say 10 x 10) matrix where each position can be referred to as P(R,C) where R & C are the row and collumn numbers respectively. Each position contains a 1 for a live cell, 0 otherwise. By examining each position in turn we can create a new matrix PNEW(R,C). When we have finished determining what the new generation will be we update P(R,C) by making each point on it equal to the corresponding point in PNEW(R,C) then start again.

This gives us the simple flowchart shown below;

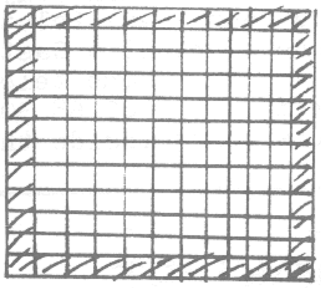



START
Set all positions of P(R,C) to their starting values and GEN to 1
Set R=1 (first row)
Set C=1 (first collumn)
Calculate the value of the position P(R,C) for the next generation and store the result in PNEW(R,C)
Y
is C=10 ? (last collumn)
yes
X
no
Increase C by 1 (next collumn)
is R = 10 ? (last row)
yes
no
Increase R by 1 (next row)
Print out the new generation
Make all P(R,C) equal to the corresponding PNEW(R,C). Increase GEN by 1
last generation ?
no
yes
END

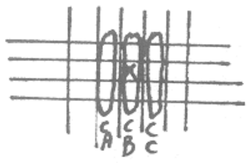
POINTS TO PONDER ; Speed of execution is the most important factor. The hints given below should generally improve the performance of a program, but their exact effects will depend upon the language in which the program is written and also upon the type of computer used to run it.

PRINT OUT in all decent machines is bufferred, so that a PRINT instruction in a program doesn't cause the machine to halt while the printer does its work, instead it puts the character(s) in the buffer so the printer can extract them at its leisure while the processor is doing other useful work. This is fine as long as you don't fill the buffer which on minicomputers rarely exceeds one printed line in length, and is sometimes as little as one character. In these cases the program could be speeded up by putting the printout at the end of every line (X in the flowchart) or even at every new position (Y in the flowchart).

BEWARE THE EDGES. The loop shown dotted is traversed most frequently - once per position per generation. Attempts to speed up the program are therefore likely to have most effect if they are in this area.
In trying to calculate the value of a position for the next generation you will have to put in time wasting IF statements so that the calculations are done correctly for the rows and collumns around the edges of the matrix.
These complications can be overcome if you make the matrix larger than it otherwise need be by adding an extra dummy row or collumn to each edge. These dummy positions contain 'O's and are never updated. They merely ensure that all of the positions in the true - inner - matrix have the full complement of 8 neighbours each.



CUT DOWN THE CALCULATION. The calculation for the next generation of any position is given by; PNEW(R,C) = 0 if N is greater than 3
= 1 if N = 3
= 1 if N= 2 and P(R,C) = 1
= 0 otherwise
where N is the number of live neighbours
= P(R-1,C-1) + P(R-1,C) + P(R-1,C+1) + P(R,C-1) + P(R,C+1) + P(R+1,C-1) +P(R+1,C) + P(R+1,C+1)
To save having to work this out each time we can say that
N = CA + CB + CC - P(R,C) where;
CA = P(R-1,C-1) + P(R,C-1) + P(R+1,C-1)
CB = P(R-1,C) + P(R,C) + P(R+1,C)
CC = P(R-1,C+1) + P(R,C+1) + P(R+1,C+1)



Then, having calculated CA,CB & CC for one position, when we look at the next position to the right we only have to make CA = the old value of CB, CB = the old value of CC and calculate a new CC.

STRING OUT THE MATRIX. instead of storing P and PNEW as 2 dimensional matrices, use one dimensional storage. P(A) can be found much more quickly by the computer than P(A,B).

HOPEFULLY by the next issue of the ACCN I will have been able to try some of these ideas out in practice and will print a REAL WORKING PROGRAM.

IN THE MEANTIME for anyone who wants to try LIFE on a VDU may I suggest the following idea ; the state of the cells are displayed frequently to avoid flicker on the display. Between each display update according to the rules of the game not all of the positions but only a portion, selected at random. At the same time introduce a number of spontaneous random 'births' and 'Deaths'. Now allow the player to have control over the success of the 'organism' by allowing him to adjust the number of the random births & deaths as the game progresses.

WANTACOMPUTA ?

No problem. Just offer to relieve the owner of his burden. Who knows, he might even pay you to take it away.
The advert reproduced here came from a recent edition of one of the professional computing magazines.

We will
DISMANTLE AND DISPOSE OF YOUR COMPUTER
as scrap and spares
Satisfied clients include insurance companies, banks, local authorities, private and public companies.
NO CHARGE IN MOST CASES
and Partners

On the same subject, we've had requests from some enthusiasts living outside London and the SE for help in finding out where they can buy second-hand computer/digital equipment. Any suggestions ?

IT IS MEET

that we meet to discuss the future direction of the ACC, elect officers etc as mentioned in issue 3.
Mr J.Creutzberg has kindly arranged that we can have the use of a room at the South Bank Polytechnic from 7.30pm on Thursday December 13.
The South Bank Polytechnic (the old Borough Poly) is on Borough Rd. at the Elephant & Castle. Room No details can be obtained from Mr Creutzberg - home telephone 674 1205 , work 928 8989 ext 2205 - or from myself (02 68 3040 ext 275) - and will also be posted on the notice board inside the main entrance. See you there.

BOOKLIST

COMPUTER ARCHITECTURE C.C.Foster 225 pages
Van Nostrand Reinhold Computer Science Series

As the title indicates, this book describes the hardware design of digital computers - an acquaintance with machine language programming is desirable. Two small machines , BLUE and INDIGO, are described in some detail. BLUE (the machines are named after the colour of their cabinets) is a very basic machine with 4K words, 16 instructions and a common bus system.It would probably be possible to build the machine from the information given, although whether anyone would actually want to is another question, as the author says, it has "none of the 'goody features' present in most current machines eg indirect addressing, index registers, interrupts etc."

A very good book for the hardware enthusiast.

More Macdonald Computer Monographs

PARLEZ VOUS CORAL, BASIC, TELCOMP, Algol, PL/1, FOCAL, PAL, RPG, 803 Autocode, FORTRAN IV, PFOCAL, COBOL etc ?. Or maybe you prefer to express yourself in flowcharts.
This is a real problem when we want to print a program in ACCN. My own preference is to use BASIC - partly I admit because I have access to a machine that will run BASIC, but mainly because I think it is a widely known language & is sufficiently simple that anyone can easily translate a BASIC program into his own preferred language. Any comments ?

THE AMATEUR COMPUTER CLUB
IS OPEN TO ANYONE INTERESTED IN THE DESIGN, CONSTRUCTION OR PROGRAMMING OF COMPUTERS AS A HOBBY. FOR MEMBERSHIP OF THE ACC AND A SUBSCRIPTION TO VOL.I OF THE NEWSLETTER SEND 50p - 75p FOR OVERSEAS MEMBERS - VOL I WILL COMPRISE AT LEAST 5 ISSUES PRODUCED AT 2 TO 3 MONTH INTERVALS.

180 MEMBERS

M LORD 7 DORDELLS
BASILDON ESSEX
SS 15 - 5BZ