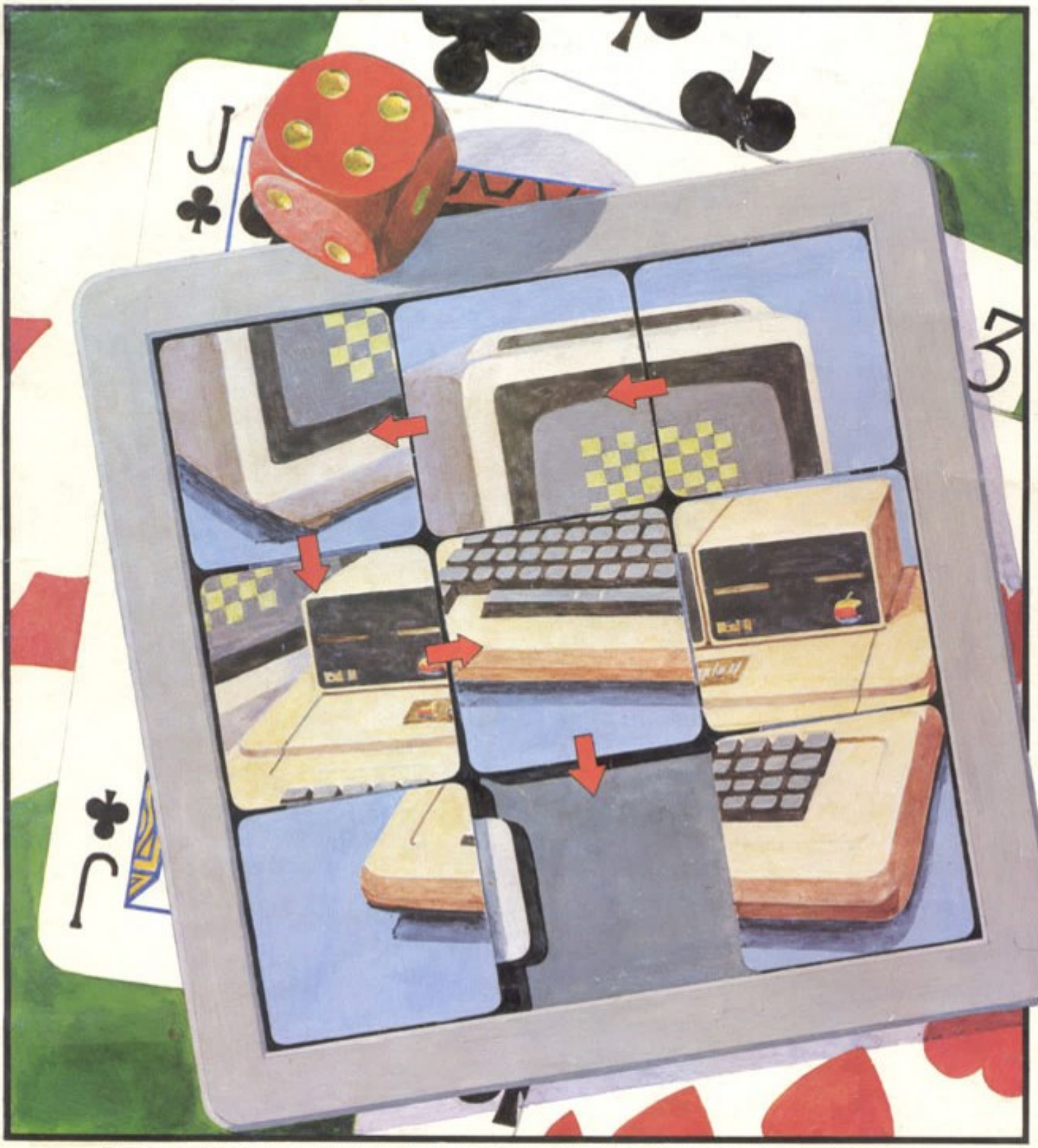
AUSTRALIAN Personal Computer

A GUIDE TO BUSINESS SOFTWARE

AUSTRALIA'S FIRST MICRO MAGAZINE

MAY 1980 $2.20



ARTIFICIAL INTELLIGENCE WITHOUT TEARS

David Levy series on game theory

APPLE II/APPLE II PLUS COMPUTERS come complete with 12K ROM, Cassette Interface, 8 port I/O facilities, tutorial manual, language reference manual, system and hardware reference manual, introductory software manual, leads, connectors, games paddles and introductory software.

APPLE II PLUS 16K RAM . . . $1395.00 *
APPLE II PLUS 32K RAM . . . $1503.00 *
APPLE II PLUS 48K RAM . . . $1611.00 *

There's never been a better time to buy an Apple® II.

apple pascal

The lowest priced, highest powered Pascal system on the market.

Our high-level, full feature Language System consists of a plug-in 16K RAM language card, five diskettes containing Pascal as well as Integer BASIC and Applesoft extended BASIC, plus seven manuals documenting the three languages.

The beauty of this Language System is that it speeds up execution and helps cut unwieldy software development jobs down to size. Also, because the languages are on diskette, loaded into RAM, you can quickly and economically take advantage of upgrades and new languages as they're introduced.

Apple's Pascal language takes full advantage of Apple high resolution and colour graphics, analog input and sound generation capabilities. It turns the Apple into the lowest priced, highest powered Pascal system on the market. With Pascal, programs can be written, debugged and executed in just one-third the time required for equivalent BASIC programs. With just one-third the memory.

On top of that, Pascal is easy to understand, elegant and able to handle advanced applications. It allows one programmer to pick up where another left off with minimal chance of foul up.

Because Apple uses UCSD Pascal.™ you get a complete software system: Editor, Assembler, Compiler, and File Handler. And because we adhere to the standard, your programs run on any UCSD Pascal system with minimum conversion. Which is really something an enthusiast can get enthusiastic about.

Apple Language System **NOW AVAILABLE @ ONLY $495.00***

Requires 48K Apple II Computer and Disk II & Controller.

The GRAPHICS TABLET is an image input device to enter pictorial information directly into the Apple II Computer. eg from:
- maps and photographs
- logic diagrams and schematics
- histograms
- architectural drawings — fine art

THE GRAPHICS TABLET
THE GRAPHICS TABLET
THE GRAPHICS TABLET
THE GRAPHICS TABLET
THE GRAPHICS TABLET
THE GRAPHICS TABLET
THE GRA

Powerful software provides comprehensive set of functions selected with stylus from menu. Graphics Tablet complete with manual, interface and software . . . **Only $874.00***

Send now for our latest free book catalogue

Now over 150 editions available: Byte Books, Compusoft, dilithium Press, Hayden, Matrix, McGraw Hill, Osbourne, Prentice Hall, Howard W. Sams, Scelbi, Sybex, University Software.

FOR FURTHER DETAILS AND INFORMATION WRITE OR CALL US

Computerland®

in Melbourne

Grd. Floor, 555 Collins Street, MELBOURNE. VIC. 3000.
Phone (03) 62 6737 (03) 62 5581 Telex AA37007.

*Prices do not include Sales Tax

Prices are correct as at time of publication and may be subject to change.

CONTENTS

Volume 1 No.1 May 1980.

Cover Illustration
Ingram Pinn

Published by Australian Microcomputer Journal,
P.O. Box 115, Carlton,
Victoria, 3053.
Telephone: (03) 82 5783.
Telex: AA30333.
P.O. Box 250,
North Sydney, N.S.W. 2060.

APC welcomes all unsolicited material (written, photographic & illustrative) and, although no guarantee can be given as to its safe return, reasonable care and attention will be exercised.

Guidelines for Contributors
APC welcomes articles of interest. Don't be put off if your style of writing is "under developed" . . . true worth lies in the content, and shaping features comes naturally to us! Manuscripts should not exceed 3,000 words and authors are asked to use triple spaced lines with a wide left-hand margin; diagrams, listings and/or photographs should be included wherever possible. Please enclose a stamped, self-addressed envelope if you would like your article returned.

Because of the foregoing, it is necessary to add that the views expressed in articles we publish are not necessarily those of *Australian Personal Computer*. Overall, however, the magazine will try to represent a balanced, though independent viewpoint. Finally, before submitting an article, please check it through thoroughly for legibility and accuracy.

Subscription Rates:
Australia $24.00 for 12 issues, New Zealand $30.00 for 12 issues (surface mail). Prices include postage and packing. Supplies to specialist shops can be arranged by negotiation direct with the publishers.

WHY WE CHOSE COMMODORE

With so many Microcomputers on the market, the choice was as difficult for us as it is for you. We wanted a machine with versatility, yet capable of specialization for individual needs and affordable.

This is what we found:—

THE PET

A very versatile Personal Computer with 8K bytes of memory that can be easily updated to 40K. Ideal for education, hobbyists, small business applications, etc. This compact machine, like its bigger brothers, can be carried and used anywhere. It operates on ordinary 240V power and is easily interfaced to a variety of other equipment. The software is written in BASIC, an easy to learn and simple to program language. A large selection of software is readily available. The machine itself, has a TV screen, calculator type keyboard and integral cassette unit. The PET requires next to no servicing yet the design is remarkably simple to facilitate any repairs.

THE CBM

A choice of two machines, either 16K or 32K, again with easily upgraded memory. Both have a typewriter keyboard, 1000 character TV screen, can be attached to an external cassette unit or, for a full business system, a dual drive floppy disk and printer. The CBM is ideal for business applications. Software readily available includes a General/Debtors/Creditors Ledgers Package, a Word Processor Package with capabilities usually only found on more expensive systems, Inventory, Stock Control, Job Costing and many more are now under development. Again, the CBM can be interfaced to other equipment and is simply designed to facilitate servicing.

BUT it is impossible to tell the full story in this small space. Now to our next point.

ABOUT US

We are professional Microcomputer Systems Designers specializing in the Commodore Product range. Our technical expertise consists of more than 10 years DP experience at high levels on mainframes, minis and micros. We can design and program systems to your personal requirements and offer you full hardware support. Should you want continuous stationery, cassettes, floppy disks, we can supply them. Like our product, we are a versatile team, eager to assist you in any way we can.

WHERE ARE WE?
Due to increased business, we have moved to larger premises at:—

4th Floor,
561 Bourke Street,
Melbourne, 3000.
(Next to AMP Building)

TELEPHONE: 614 1433, 614 1551

B.S. MICROCOMP FOR

- SOFTWARE
- HARDWARE
- SERVICE
- SUPPLIES

MICROCOMP
MICROCOMPUTER SYSTEMS DESIGNERS

bankcard
welcome here

Editor
Sean Howard

Editor (U.K.)
Bruce Sawford

Technical Editor
David Tebbutt

Consultants
John Coll,
Mike Dennis,
Miriam Cosic,
Michael James,
David Hebditch,
Sheridan Williams,
Dr Adrian Stokes,
Dr Stephen Castell.

Promotions Manager
Marie Pirotta

Promotions Assistant
Josephine McKenna

Production Manager
Michael Thomas

Typesetting & Paste-up
Pam Typesetting

Printed by
Nunawading Press,
Victoria.

EDITORIAL

The advent of computers in the 1940's evoked awe and curiosity, in much the same manner as H.G. Wells' tales had done forty-five years previously. During their subsequent development, computers were confined to staid occupations in defence departments, university laboratories and large corporations. Research into the broader applications of computers was not encouraged. And understandably so, for the production price plunge in the 70's was as unforeseen as the invention of the silicon chip.
Today, computers no longer require a team of white-coat, spectacled professors to supervise their operations – the microcomputer has entered the home. The new generation of "non-professional" users has introduced computers into many areas. We see the expanding use of microcomputers assisting in education, medicine and small businesses. Applications in dedicated functions, such as the heart of security systems and traffic signal control, are also increasing and diversifying.
Yet, despite this revolution in hardware, software has not enjoyed such impressive development – there is frequently a shortage of the quality software required in these new fields.
Perhaps the most exciting aspect of the widening use of microcomputers is the possibility of reducing this deficiency. The number of computers now being used by interested amateurs and people of unrelated disciplines will lead to radical approaches in the development of software. Already there are movements in this direction, in the legal and educational spheres for example. Australians should recognize the potential for innovative software and strive to establish an international reputation in this field.
Sean Howard.

An Invitation

You are out there doing interesting things, developing new software and hardware devices, setting up clubs and societies – perhaps even getting concerned about the likely implications of the coming technological revolution. Where better to broadcast and share the wealth of your knowledge, ideas and experience than in the pages of Australia's first microcomputer journal?
If you are concerned about style of presentation – read our 'guidelines for contributors', if you are unsure about content – there are very few restrictions. Obviously, though, an unusual contribution has more chance of being published than, say, a program for 'Mastermind'.
We'd be particularly interested to hear from people who have installed a business computer in their own business, from people who have hooked-up something unusual – perhaps through an analog interface, from people who wish to 'soapbox' their views and from people still at school or university. Also, if you want to tell us something but prefer not to go into print – fine – just write in anyway because we'd love to hear your views, good or bad. Remember, this is very much *your* magazine and its future direction should, and will, reflect the views and desires of its readership.
We look forward to hearing from you.
The Editors

TRS80* MEANS BUSINESS

IMPROVE TRS–80 PERFORMANCE WITH NEW DOS+

Enjoy the wizardry of your TRS-80 to its fullest: maximize and expand all of its magical capabilities with the new DOS+. Just look at the capabilities you can evoke.

Modifications, corrections, and enhancements to Radio Shack's TRS DOS 2.1.

- A Basic **REF**erence command for variables and numbers.
- A built in keyboard-debounce routine.
- A print screen option under DOS or BASIC to your line printer. Simply press JKL keys.
- Execution of DOS commands while in BASIC.
- Apparat's own **SUPERZAP,** a Hex dump utility to examine or modify disk or memory locations.
- A super-fast machine language **DISASSEMBLER** program.
- An improved **DISKDUMP** program.
- **Level 1 ROM** relocated in Level II RAM.
- A super-fast machine language **RENUM** program executable under BASIC.
- New copy commands for backup, allows you to copy from drive to drive keeping the same filespec.
- New BASIC scrolling and invocation commands and more.
- Modified **EDITOR ASSEMBLER** with Disk I/O and new cross reference feature.
- A LOAD MODULE for transferring machine tapes to disk.
- A **DIRCHECK** program to test a directory & list/display the contents in alphabetical order with extensions.

You don't have to pull a rabbit out of a hat . . . Send $99 for the Apparat NEW DOS+ (on diskette).
See all the magic unfold before your eyes.

INTRODUCTORY SPECIAL

NEW DOS+ and the very **Best of Apparat's** disk utility programs $99 (a $250 value). 40 Track version $110. (**NEW DOS 35** $49– **NEW DOS 40** $60).

TRS-80 SOLUTION

HARDWARE

3 speed modification – a very simple kit; only 4 connections, easily reversible	$29.95
Blank computer cassettes – leadless, guaranteed no dropout	$1.80
5 - $9.00 10 – $16.00 100 –	$100.00
Reverse video	$31.00
Archbold speed up	$35.00

SOFTWARE

Sargon 1	$25.00
Sargon (version 2)	$37.50
Microchess "still the best seller" Level 1 and Level 2 4K	$24.00
Business Package 6 cassettes – Level 1 4K "6 pack"	$29.00
Golf and Crossout "2 programmes"	$9.00
Game Playing with Basic "TRS Apple Pet" Tape 1, Tape 2, Tape 3 each	$12.50
10 pin Bowling "4 zones and 9 angles" TRS80 L1 4K and L2 16K	$9.00
Santa Paravia "Become the ruler of a Medieval City" Graphics TRS80 L1 and 2	$9.00
General Mathematics TRS80 L1 and L2 Apple 2	$12.50
Space Trek IV "Trade or Wage War"	$9.00
Oil Tycoon	$9.00
Packer	$29.95
Gammon Challenger	$17.40
Codebreaker "Sound"	$14.00
Stock Market	$12.00
Special Delivery	$99.00
Complex Matrix Maths. Pet TRS80 1 or 2 – Apple 2	$12.50
Checker King	$27.00
Flight	$9.00
Airmail Pilot	$9.00
Utility 1 and 2 each	$9.00
Backgammon/Keno	$9.00
Teacher	$12.00
Ham Package	$9.00
Azimuth Finder	$14.00
Music Tutor	$12.00
Star Trek "Acorn"	$12.00
Stock Market	$12.00
Bandito	$12.00
German "Disk"	$23.00
French	$23.00
Italian	$23.00
CCA DMS "Disk 32K"	$95.00
Stimulating Simulations	$21.00
Electric Paint Brush	$21.00
Time Trek "W/Sound"	$21.00
Level 3 Basic	$52.00
Typing Tutor	$17.00
Editor/Assembler	$34.00
Adventure "Original 32K"	$35.00
Fortran	$102.00
Spooler	$21.00
my MATH	$77.00
Basic Compiler	$202.00
Taipan	$12.00
Alien Invasion	$14.00
System Savers	$17.40
Atesm "Terminal"	$35.00
Print to L Print etc.	$9.00
Structured Basic Translator	$35.00
Cassette Label Maker	$16.00
Disassembler	$19.95

PROGRAMMING AIDS

Combination Coding/CRT Forms In 50s – 2 pads	$8.50
Flow chart sheets 280mm x 406mm In 50's – 2 pads	$10.00
Giant Video Display Pads – 5 pads	$37.50

SUNDRIES

Diskettes "Scotch" 5¼ s.s. 1	$7.00
The Best – Don't risk your data 10 for	$60.00
with cheaper disks 100 for	$550.00
Combination QSL Album & Log Book	$9.95
Binders for CB Action	$6.50
Binders for Amateur Radio Action	$6.50

TRS80 Business Systems
(Model 1 – Level 2 Osborne)
General Ledger
Accounts Payable
Accounts Receivable
Invoicing and Inventory

$99.00 each on DISKETTE

DEFOREST SOFTWARE

QUALITY QSL PTY. LTD.
26 Station Street, Nunawading Vic. Australia (03) 877 6946
Post included at no charge, however we suggest certification or registration.
DISCOUNTS: Schools less 10% — Trade Enquiries welcome.

THE TRS-80 MODEL II PROGRAMS

General Ledger/Cash Journal: handles up to 7000 transactions on 500 different user-defined accounts. It keeps track of them by month, quarter and year, makes comparisons to the prior year, and does departmentalization.

Accounts Payable/Purchase Order: generates the purchase order and posts the item to payable when the goods are received. Invoice-linked, it calculates and prints checks and aged ledger reports and links fully to the general ledger.

Accounts Receivable/Invoicing: keeps track of billed and unbilled invoices, open and closed items, aging and service charge calculation. It prints statements, links to the general ledger, and can work within either an invoice-linked or balance-forward accounting system.

Payroll/Job Costing: computes regular, overtime and piecework pay, keeps employee files, figures taxes and deductions, prints checks, journal, 941-A and W-2 forms, and breaks out individual job costs.

These Model II programs are completely customtailored, which explains their $249.95 price. Before we'll send you a disk, you have to fill out a detailed questionnaire that tells us your precise business requirements. Then we send you the disk, all the instructions you need.

Please send me the custom questionnaires for the following $249.95 Model II Programs.

- ☐ General Ledger/Cash Journal
- ☐ Accounts Payable/Purchase Order
- ☐ Payroll/Job Costing
- ☐ Accounts Receivable/Invoicing

☐ Please send me information on the TRS-80 Model I programs at $99.95 each

Your name . . . . . .
Company name . . . . . .
Address . . . . . .
City/State/Postcode . . . . . .

SPECIAL! SUPER "NEW DOS/80" now available $149.00

ADVENTURES

1–8 on Tape $16.00 each, 1 on DISK $19.00
2 on DISK $29.00, 3 on DISK $39.00

LIGHT PEN TRS-80 LEVEL II

BYPASS THE KEYBOARD
INTERACT WITH THE SCREEN DIRECTLY

APPLICATIONS

- o Education
- o Business
- o Games
- o Home

NO ASSEMBLY NECESSARY
READY TO PLUG IN
COMPLETE INSTRUCTIONS AND DEMONSTRATION PROGRAM PROVIDED

TRS-80 PEN Includes
Demo-Game Cassette $19.00, Kit Form – $14.00

DUPLICATE SYSTEM TAPES WITH "CLONE"

This machine language program makes duplicate copies of ANY tape written for Level II. They may be SYSTEM tapes (continuous or not) or data lists. It is not necessary to know the file name or where it loads in memory, and there is no chance of system co-residency. The file name, entry point, and every byte (in ASCII format) are displayed on the video screen. Data may be modified before copy is produced.
CLONE..... $18.00

Analog Input for Pet

Edible Electronics announces the Australian release of the AIM161, a 16 channel analog to digital converter designed to work with most microcomputers. The AIM161 forms the heart of the DAM Systems, designed to produce low cost measurement and control input/output modules for small computers. The modules are capable of digital input sensing, analog input sensing, digital output control, and analog output control.

The AIM161 is connected to the host computer through the computer's 8 bit input and 8 bit output ports.

Presently the AIM161 can only be interfaced to the Pet, although similar instruments will soon be available for other microcomputers.

tails from Edible Electronics, PO Box 1053, North Richmond, Victoria, 3121. Telephone: (03) 41 5708.

1980 Home Computer Show

This month sees the fourth Home Computer Show in Australia, being held at the Westco Pavilion, Sydney Showgrounds on May 22–25.

The success of the three previous Shows, one in Sydney and two in Melbourne, amply demonstrate the growing interest in microcomputers here in Australia.

The increasing number of small businesses and schools employing microcomputers suggest that this year's Sydney and Melbourne Shows will be well attended. According to the newly appointed 1980 Home Computer Show director, John Kennedy, "The 1980 Home Computer Shows apart from reflecting the current status of microcomputer equipment and technology will also offer a valuable insight into future directions. The 1980 Show will coincide with the expansion of the educational market as secondary and tertiary institutions plan their immediate purchases of equipment. We will also recognize that the microcomputer area is a stepping stone for many young people preparing for careers as computer professionals."

For further inforamtion please contact John Kennedy, 443 Little Collins Street, Melbourne, 3000, telephone (03) 67 1377; or Carol Dugan, 39 Mirrabooka Street, Bilgola Plateau, N.S.W. 2107, telephone (02) 918 8174.

Sinclair Surprise Cheapie

Clive Sinclair, the inventor of the pocket television ("for deep pockets", as Electronics Weekly once commented) has done it at last.

Exactly what he has done is to produce a personal micro with keyboard and TV interface and BASIC for under $160 in kit form, and I for one don't know how. Its only apparent drawback is the fact that it uses a touch sensitive keyboard. This is great for preventing spilled coffee or beer from getting into the switches and no doubt it keeps costs down that little bit extra, but it does make touch typing impossible; that requires the operator to keep the fingertips on eight of the keys all the time.

Apart from that quibble, it's obviously non-standard in terms of getting programs off other people's cassettes. Sinclair has hinted that his BASIC is greatly compressed and uses a quarter of the memory that a competing BASIC would use – both for itself and for programs that run under it. This saves money, but since nobody else has done it, their programs will need to be rekeyed and slightly modified to run on the Sinclair machine.

Don't expect to see the machine here for quite a while. Presently the manufacturers are having difficulty meeting demand with thousands of orders already placed. When it does arrive, however, I expect it to retail for around $300 fully assembled.

New Tele-ray Terminal

Anderson Digital Equipment has just released the latest Teleray Terminal, called "The Model 12T". Unlike previous models, the designers have given more consideration to the appearance of this new terminal.

It has a 48 line (3840 character) display memory with a 24 line rolling display window. Its 7500 character function memory can be shared by up to 32 programmable function keys. Text, forms or control sequences can be stored in this memory. The terminal is able to "PARK" excess lines (up to 90) in the function memory for later processing, should more than 48 lines of editing workspace be needed.

The Model 12T is a smart terminal including the following editing features: insert/delete a line/ character plus V dim, blink, inverse-video and underline, which can be programmed in any combination. It also has a secure feature should it be necessary to enter unseen data such as "Pass-Word" switchable baud rates (50–9600) block/character made and full duplex options are provided as standard.

By using the function memory, forms can be designed and stored, then dumped to disc for future retrieval via the 2120 Teledisk.

The unit price is expected to be approximately $1995. Currently delivery is 90 days from order, but A.D.E. expect to have stock of these terminals by August.

Further details from Anderson Digital Equipment Pty Ltd, PO Box 322, Mount Waverley, 3149 (543 2077) or PO Box 294, Ryde, 2113 (808 1444).

A.D.E.'s new Teleray terminal.

National Power Down

The American chip maker National Semiconductor has been tackling one of the most irritating problems of microcomputers – the fact that they burn up a lot of power.

A six transistor radio can run for days off a small nine-volt battery. A 7000 transistor Motorola 6800 doesn't use proportionally more, but a system using it plus a manageable amount of memory, plus a useful amount of amplifying circuitry (to make sure that the words it wants to print will reach the printer) will kill that battery in an hour.

National Semiconductor reckons that the people who have tackled this problem in the past have got it all wrong; instead it has produced the new NSC 800 micro. Not just a new computer, but a brand new chip.

What makes the NSC 800 important is that it's not in fact a brand new chip at all. It's a copy of Intel's 8085, plus a copy of Zilog's Z80-two of the most commonly used chips in the micro business. It will run any software written for those micros, without change. But it uses 4% of the power.

In the past, the only way to get a computer to be this

miserly in its use of electricity, was to build it on a special chip with two layers of impurity – both P-channel and N-channel complementing each other on the silicon (hence its name, Complementary Metal Oxide Semiconductor, or CMOS). Normal fast micros are built in N-channel, or NMOS technology – and usually, CMOS micros are a great deal slower in operation.

National's trick has taken place on two levels. First, it has found a way to speed up CMOS, and second, it has developed that faster CMOS so that the transistors are much smaller, and much closer together. In part, the first point is the result of the second.

It works like this: the transistors on a silicon chip operate by pumping electrons into or out of one of the three elements of any transistor . . . collector, base, and emitter. The exact details vary with the type of transistor you have built, but enough electrons in (or out, depending on type) can turn a piece of silicon from a good conductor into a rotten conductor of electricity; when the current flowing between the other two elements stops, the transistor has switched.

One can say that the bigger the transistor, the longer it takes to pump enough electrons in, to switch it. If you pump harder – that is, increase the power – the thing gets hotter, but it does speed up.

Very, very crudely, the effect of shrinking the length of a chip by x per cent can improve the speed or power – whichever you design for – by a factor in proportion to x^3; that's because it affects not just the area, but the volume, of the transistor.

Until now, the big problem of CMOS has been that although it overcomes the limitations of speed power on NMOS, is spreads itself over a lot more silicon. In doing so it causes more capacitance effects and increases the likelihood of a chip falling on a crystalling discontinuity in the silicon wafer. The capacitance effects slow the chip down because capacitors take time to fill up with electrons; the flaws reduce the number of good chips on a wafer, and therefore increase the cost of building and testing.

A CMOS micro small enough to be economically viable was not much of a computer, as any user of an Intersil 6100 or an RCA Cosmac will have to admit. Since those devices were originally launched, however, the art of squeezing circuitry on to chips has improved. National's latest improvement involves putting an extra layer of polysilicon on the chip – it calls the process double polysilicon or P^2CMOS – and this allows it to build a very complex chip indeed. Like the Zilog Z80, it runs all the 8080 instructions and also unlike the Z80, it runs the 8085 instructions and interrupts. A small difference, but quite crucial to somebody with an 8085 program that goes beyond what the 8085 can do.

It also contains quite a lot of extra circuitry for controlling memory, and input/output, which means that three of National's chips will do quite a lot of useful computing – where the Z80 would need twice as many.

Those three chips, unobtainable today, carry a $115 price tag. A 40% price cut next year and a 30% cut the year after will still make it more expensive than Z80 or 8085, but not killingly so.

What makes it only a dream for the average home builder is the fact that, while the main chips will be available by the end of this year, there are many support chips that will only be available in standard CMOS (rather slower) or perhaps, even only NMOS. One such chip is the EPROM, the ultra-violet eraseable, reprogrammable permanent memory. You can mix them, certainly, but one EPROM will chew up ten times the power you just saved by switching to P^2CMOS.

Silicon Burns

Due to a recent fire, Silicon Valley's shop in Bridge Road, Richmond, Victoria, has been forced to close. As there are no plans to reopen the shop, Abacus Computer Store, also of Bridge Road, has been appointed as an agent for Silicon Valley products. All stock previously available from the Silicon Valley shop, including

components, kits and the Texas Instruments range of calculators, will now be sold from Abacus Computer Store.

Abacus also announce that they are now stocking the Compucolor II range.

For more information, please contact Abacus Computer Store, 512 Bridge Road, Richmond, Victoria. Phone: (03) 429 5844.

Softwear for the 2650

MicroByte, a business specializing in the production of software for the 2650 microprocessor, has announced its first product releases. Four products are available: an Audio Cassette Operating System, Assembler, Disassembler and Microword BASIC.

For more information, write to MicroByte, PO Box 274, Belconnen, ACT, 2616.

Believe It Or Not Department

The first microprocessor, Intel's 4004, was a very simple device by any of today's standards; indeed, only by courtesy was it a processor at all. Most of the essential functions of a processor chip were on secondary, support chips. Even up to the point of the 8080, this was still true.

Now that a chip is capacious enough to carry things as powerful as Motorola's 68000, Data General has gone back to the 8080 approach – in order to squeeze its enormous Eclipse "mini" on to a single chip. The only acceptable reason for calling the Eclipse a mini is that Data General is a minicomputer company. IBM makes several smaller machines and calls them "mainframes".

The single chip that is the central processor of the Eclipse does not have room for all the instructions that can normally be added by microcode. Nonetheless, the DG microEclipse is an awesome bit of silicon, driving two megabytes of memory. I mention this machine in a "personal computer" publication just to put a little perspective on the future.

The Japanese have all announced 256 Kbit RAM chips now, and in two years, these will be available. A useable system with the microEclipse and 2 Megabytes of store will occupy 20 pieces of silicon. . . the size of Clive Sinclair's ZX80. Scared?

Data Decor

As the demand for desk top computers continues to grow so there is an expanding market for suitable furniture to accommodate these units and their peripherals.

Data Decor was established in February with this demand in mind. Stephen Decker & Stuart Macdonald now produce a range of modular computer workstations, designed to specific requirements.

Systems can be coordinated in a range of finishes to suit business, laboratory and educational locations.

Details from:
Stuart Macdonald,
1A Fenwick Street, Kew,
Victoria, 3101.
Phone: (03) 861 9702.

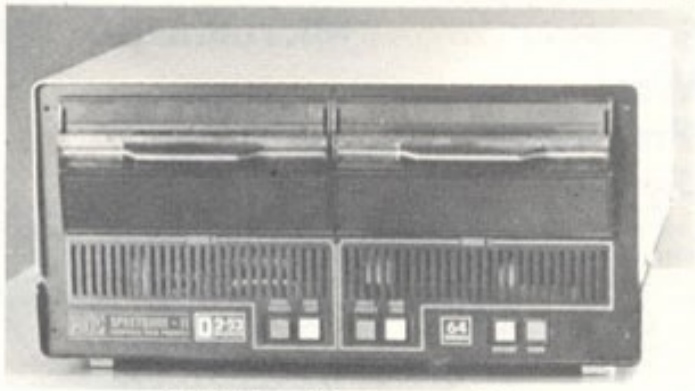
Aust.Designs Impress US

An Australian designed and built general purpose minicomputer will be installed in a United States' engineering plant.

The machine is the Spectrum-II D, a dual-sided double density floppy disc computer designed and manufactured by D.D. Webster Electronics Pty Ltd of Bayswater, Victoria. The company has been making Spectrums for two and a half years and has more than 120 installations. The small business orientated computer range is based on Digital Equipment Corporation's LSI-11 processor, but is 75% Australian in design and content.

The Spectrum will be installed at Dynatron Inc., in Wallingford, Connecticut, a company which manufactures energy conservation systems, employing 45 people. It will administer Dynatron's bookkeeping using software created by another Australian company, Tritechnic Pty Ltd of South Yarra.

The Chief Executives of the respective companies, Mr. David Webster and Mr. Ray Moloney, will supervise the installation, which they believe will be the first Australian computing system to operate on a U.S. business site. The system in Connecticut will not only provide a comprehensive computing facility for Dynatron, but will serve as a test and referral site for future U.S. sales.



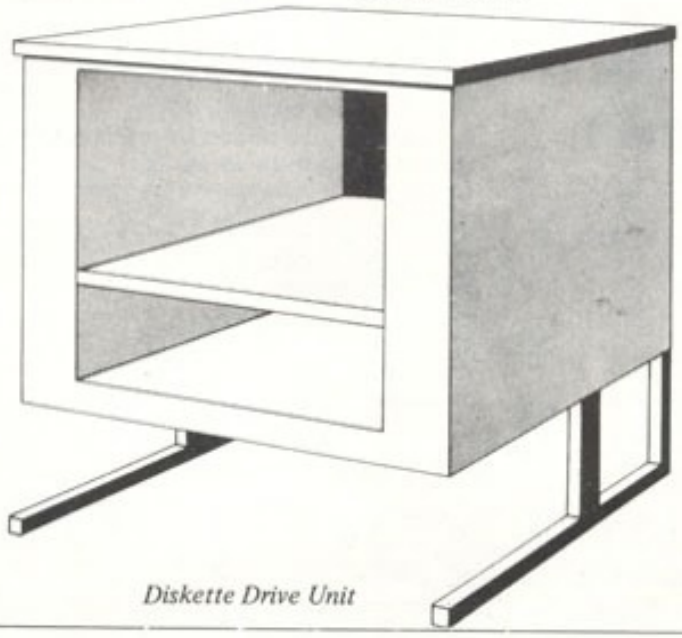
D.D. Webster's Spectrum – 11D.

Following completion of the installation, a second Spectrum-11 D will be taken to New York to be demonstrated at the Rockefeller Centre. This promotion will include the demonstration of an Australian multi-language student programming system, MONECS (Monash Educational Computing System), which was developed by the Monash University Computer Centre. The MONECS software uses a minimum of hardware: a Spectrum, card reader and printer. This system can run 200 jobs per hour and cater for the needs of up to 500 students taking introductory computer courses.

Accounting packages by Australian software house King-Smith and Associates will also be shown at the Centre.

Acorns from Cottage

Cottage Computers will be supporting and marketing the Acorn computer range of microsystems and modules from England. The Acorn system is based on Eurocards and uses the popular 6502 microprocessor (as used in Pet, Apple, Kim, etc.) The range extends from microcontrollers up to System 3 with floppy disc drives. A feature of the range is the VDU card which will generate Teletext characters and has full colour capability. Soon to be released is the Pascal package to run on the 6809 CPU board which plugs straight into the system.
Further details and demonstrations are available from the Cottage Computer shop at 386 Queens Parade, Fitzroy North, Victoria, Telephone: (03) 481 1975.



Diskette Drive Unit

NEWSPRINT

Sorcerer Word Processor System

Dick Smith Electronics has released a complete Word Processor system based on the Sorcerer computer. The cassette based system costs around $5,500 and a full scale disc based system, $6,600.

The Sorcerer becomes a ready-to-go word processor by plugging in a special 'ROM PAC' cartridge. Because the Sorcerer is a complete computer, it will be particularly interesting to small businesses feeling the need for a computer and word processor – but unable to justify buying both.

The ROM PAC WP system includes comprehensive editing and insertion functions and a macro-programming facility which makes light work of repetitive jobs such as servicing mailing lists.

As part of the Sorcerer WP system, Dick Smith is stocking a daisy wheel printer. This produces quality printing at about three times the speed of a golfball typewriter on any stationery up to 380mm wide.

Individual word processor ROM PAC's can be purchased for $275 (incl. S/T).

Further details can be obtained from any Dick Smith retail store.

Microline 80

The Logic Shop announces the release of the OKI Data Microline 80. This is a 9 x 7 dot matrix impact printer, which uses readily available stationery. The Microline 80 incorporates the standard 8" pin feed or roll holder which accepts a normal telex roll. The optional tractor feed allows the use of continuous form feed, 3" to 8.5" wide.

The Microline 80 has a character set of 96 ASCII and 96 graphics characters. It is software selectable to print 5, 10, or 16.5 characters per inch (giving a maximum of 132 characters per line), as well as 6 or 8 lines per inch.

Interfaces are microprocessor controlled. Standard models include Centronics-compatible parallel, and RS232C serial versions. The printer is plug compatible to most mini and micro computer systems.

The manufacturers claim that the print head life is 200,000,000 characters and the ribbon is good for 2,000,000 characters.

The Microline 80 costs $895.00 (plus S/T where applicable), with a tractor feed option being $75.00

The Microline 80 is distributed by Anderson Digital Equipment, and is available from The Logic Shop, 212 High Street, Prahran, (03) 51 1950, (Melbourne) or 91 Regent Street, Chippendale (02) 699 4910 (Sydney).

Debtors System

IMS Computer Systems (the microcomputer division of Integrity Management Services Pty Ltd), a Melbourne based software house, has released version 3.2 of its Debtors system.

The new system complements the existing stand alone debtors and the fully integrated Debtors/Invoicing/Stock Control/Sales Analysis system.

Version 3.2 is a stand alone debtors with full invoicing facilities. It is primarily designed for organizations that require sophisticated invoicing but do not have a "stock" problem. Typical organizations include professional services such as accountants and solicitors.

Users of V3.2 will be able to produce statements and trial balances automatically. The invoicing and debtors stationery include the facility for the user to nominate column headings which suit the particular business or profession. It is understood that the IMS package is the only microcomputer software offering this facility. Stationery is available "off the shelf" and may be purchased in small quantities.

The new package is available for a wide range of microcomputers which utilize the CP/M operating system. Cost of V3.2 is $500 and includes system disc and operator manual. Simple debtors is $300 and fully integrated version is $1,000.

Further information can be obtained from IMS Melbourne.

From Tannery to Computer

Anderson Digital Equipment recently moved into new premises at 31 Kate Street, Kedron, a Brisbane suburb. The site now occupied by ADE in Queensland was once an old tannery.

The complex comprises showroom, reception area, general office, a large workshop area, classroom, storeroom, as well as the usual public amenities.

The total area is 4000 sq. ft. and of that 1000 sq. ft. is devoted to an instructional facility.

Contact:
Anderson Digital Equipment.

Challenger Interface

An interface for the Challenger IP and Superboard II has been released in Australia. The unit permits control of a wide variety of equipment under BASIC commands.

Features of the board include a five tone sound generator, five TTL compatible switching outputs (for control of lights, relays, etc), two relays for non TTL compatible devices and LED's to monitor status of all outputs.

The unit is available in kit form or fully assembled. Further details are available from Looky Video, 418 Bridge Road, Richmond, Victoria. Telephone: (03) 429 5674.

BENCH TEST

COMPUCOLOR II

The review of the Compucolor II marks the inauguration of the APC Benchtest. Though check-outs of microcomputers are nothing new, so far no Australian monthly magazines have made any consistent attempt at standardisation of testing routines and procedures. Results, even those excellently conceived, have rarely been tabulated for sensible comparison. Over the next few months, we hope that the name "Benchtest" will become synonymous with the structured, reliable and meaningful investigation of hardware equipment.

The Compucolor II is a fairly recent addition to the currently available range of personal computers. Being a late arrival has allowed it to benefit from careful research into the desires of the American hobby market, where it is already the fourth most popular microcomputer (after Tandy, Pet & Apple). A useful starting place for a review of this machine would be to speculate what might have been the results of this research.

"At the cheap end, volume production of well-packaged units ensures Commodore and Tandy an unassailable market lead, while the top end – where the hobby market merges into the small business and science-lab field – is too fragmented to offer substantial sales on a single product. The middle of the range (e.g. Apple) offers the sophistication of graphics and/or colour. Hobbyists generally upgrade their BASIC and exchange their cassettes for disc drives as soon as they can, so that anything pitched at Apple prices which includes a powerful BASIC, disc drive and colour video is in with a chance".

It seems that the Compucolor II was produced around this philosophy of design.

Hardware

The Compucolor II looks very much like a colour television set with a separate keyboard. The minifloppy disc drive is located to the right of the screen and the microcomputer itself is located underneath the CRT. The system tested was built for American mains but came with a 240 volt transformer rated for a line frequency of 50 Hz.

A great deal of attention has obviously gone into the design of the CRT display (all reference material describes this as 13" – my tape measure made it just 12"). There are eight bright colours which can be selected by program or from the keyboard. They are fully utilized by the operating system (prompts for filing are yellow, BASIC is green, errors are red). In character mode there are sixty four characters per line and either thirty two or sixteen (double size) lines per page. It has 128 x 128 point graphics and as the display is memory mapped, there are the usual cursor control facilities as well as the ability to store the full screen on disc.

There are three choices of keyboard. The standard has seventy one keys including AUTO (that brings up a disc directory), repeat, CPU reset, escape, break and control. It requires two keystrokes for "plotting" (a term that covers character height, flashing and colour as well as its obvious meaning). The extended keyboard with 101 keys has a special pad for colour and commonly used commands, as well as a numeric keypad. The deluxe keyboard with 117 keys can access all the plotting facilities with one keystroke. Since all possible plotting options can be called on the standard keyboard there are two (and sometimes three) ways of accessing most commands on the extended and deluxe keyboards.

BENCH TEST

Graphics

The Compucolor II has colour graphics capabilities accessable through the keyboard and under program control. Rather than having just the elementary graphic functions (move from point to point and draw from point to point), Compucolor provides a wider range of facilities. For instance:

1 colour: The eight colours – red, blue, green, yellow, white, black, magenta and cyan (light blue) – can be used in any combination for both background and foreground. Any colour can be intensified.

2 cursor: The cursor, which is an "equals" sign, can be moved around the screen and homed. It can be made invisible, that is, the same colour as the background (normally it is a different colour from both the foreground and the background). When editing a line under cursor control the back arrow causes characters to disappear, while the forward arrow causes them to reappear.

3 characters: All special characters are constructed from a 2 by 4 matrix of small blocks. There are sixty four special characters of which thirty two can be user defined while the rest are predefined. All characters can be displayed in single or double height (width remaining the same).

4 flashing: Any display or part of display can be made to flash.

5 graphics: Individual points can be plotted on a 128 by 128 point grid. An optimal path can be drawn between two points.

6 bar graphs: A bar can be drawn either vertically or horizontally in one instruction.

Playing with the graphics on the keyboard is quite straightforward but incorporating graphics into a program is a different matter (on *any* system). Compucolor avoids the sheer drudgery of point to point work by providing the powerful routines detailed above.

Disc drives

The 5¼" minifloppy disc drive is made by Siemens. Each side of the disc may be used but as there is only one read/write head per drive then, equally, only one side of each disc may be on line at any one time.

Each disc has a formatted capacity of 51.2K bytes per side and it is possible to attach an extra drive making a maximum of two.

Compucolor claim an average access time of 400ms and a transfer rate of 76.8K *Bits* per second.

It is interesting to note that there is no disc controller on Compucolor II. All data transfers take place through an eight bit parallel port, one byte at a time, tying up the MPU in the process.

Basic

The 16K BASIC interpreter is stored in ROM. Compucolor Corporation named it DISK BASIC 8001 to emphasise its disc filing facilities. It bears a strong resemblance to Micro-soft Extended BASIC and is quite powerful.

Numbers are in the range 10^{-38} to 10^{38}. Variable names can be of any length but, unfortunately, only the first two characters are recognized. The system has a comprehensive set of error messages, also only two letters long. The interpreter is quite particular when it comes to what spacing it will accept. However after imposing this spacing convention on the programmer, the interpreter itself does not follow it, listing programs in a different and slightly peculiar format! The screen editing facilities cannot be easily used to re-edit a BASIC program.

The main BASIC reserved words are listed in chart 1. Note the following:

DIM Defines multi-dimensional arrays. Subscripting starts at 0.
GET Used for file processing.
OUT Outputs a byte to another port.
PLOT Multifunctional graphics facility.
PUT Used for file processing.
FRE Number of free bytes left.
LIST Does a full listing or from any specified line. Pressing "BREAK" followed by return will enable stepping through a program screen by screen.
CALL Used to interface with 8080 routines.

Note also that there is no re-initiating command, requiring the programmer to boot the BASIC to clear the memory, thereby losing any colour and character size settings.

The Compucolor II – disc drive and colour video.

File handling

Incorporated in the systems software supplied with the Compucolor II is a set of routines known generally as FCS (for File Control System); it has much in common with PDP 11 software. The prompts (in yellow) are FCS> while the file naming format is Name, Ext;Version.

The main FCS commands are listed in chart 2. They may be used under FCS or within the BASIC system.

In addition there is an extensive set of mnemonic error messages. One annoying feature of the sharing of commands between BASIC and FCS is that BASIC requires quotes around the filename while FCS does not.

The DISK BASIC 8001 also offers extensive random access file-handling through FCS with instructions including creating, opening and closing files, inserting and amending records and error trapping. Unfortunately these commands are all concentrated into a single routine and are called via single character parameters (much like the PLOT instruction).

There is also a very convenient disc accessing facility that allows easy execution of stored programs. When the

Already the fourth largest seller in the United States, the Compucolor II retails in Australia for around $2100.

AUTO key is pressed, if there is a program called MENU. BAS on the disc it will be loaded and executed. The manual explains how this program can be written to display a numbered list of the other programs on the disc, any one of which can be run by selecting the appropriate number. After the requested program has been executed, control returns to MENU. BAS.

Business potential

The price range of the Compucolor II places it in the small business class and although the bright colours might prove to distract, it could be argued that they may usefully be used to convey information (red debts, green credits etc.).

The disc system provides the greatest limitations for using the Compucolor as a business machine. Only two discs can be on line at the same time, and these may only have 51Kch each. For example, a disc would hold just 200 records if each were 256ch long, or 500 if each were 100 characters in length – not an unreasonable demand for a product record, for example. The lack of DMA (direct memory access) probably rules out Compucolor for any serious business application.

Compucolor Corporation did not design this system to interface easily with add-on products currently available. It would be extremely difficult to upgrade the disc system because of its non standard design and a maximum capacity of 102K bytes on line would be easy to outgrow.

Education

The Compucolor II keyboard is a bit flimsy for normal school use and the availability of all the controls on the keyboard is probably unwise in an educational environment. A key which disabled CPU reset and Auto (disc load) would make it a more viable school system.

However, the Compucolor II offers many features considered highly desirable, if not essential, in a good teaching aid. The distinct colours, the large letters and (rudimentary) graphics are invaluable for demonstrations, and the packaging makes it more portable than an overhead projector. It could be argued that a graphics capability with better resolution should be traded for the colour but this would depend both on the level of pupil and the amount of time available for development.

When considering the microcomputer as a system for teaching programming and computer science, it must be noted that 16K BASIC is as extensive as could be desired and the filing system is excellent for teaching data processing as well as storing pupils' programs. An 8080 assembler can be purchased for HSC level computer science which, although not the best assembler on the market, is adequate for teaching low level programming. There are no other language translators available at this time.

Graphical exercises (a major non-mathematical application) are useful for teaching problem solving to pupils. The PLOT instruction, the heart of the graphics system, is so obscurely parameterized that the mastery of it is a difficult skill. More importantly, it constitutes a hindrance to the clear-out programming style towards which the pupil should be steered.

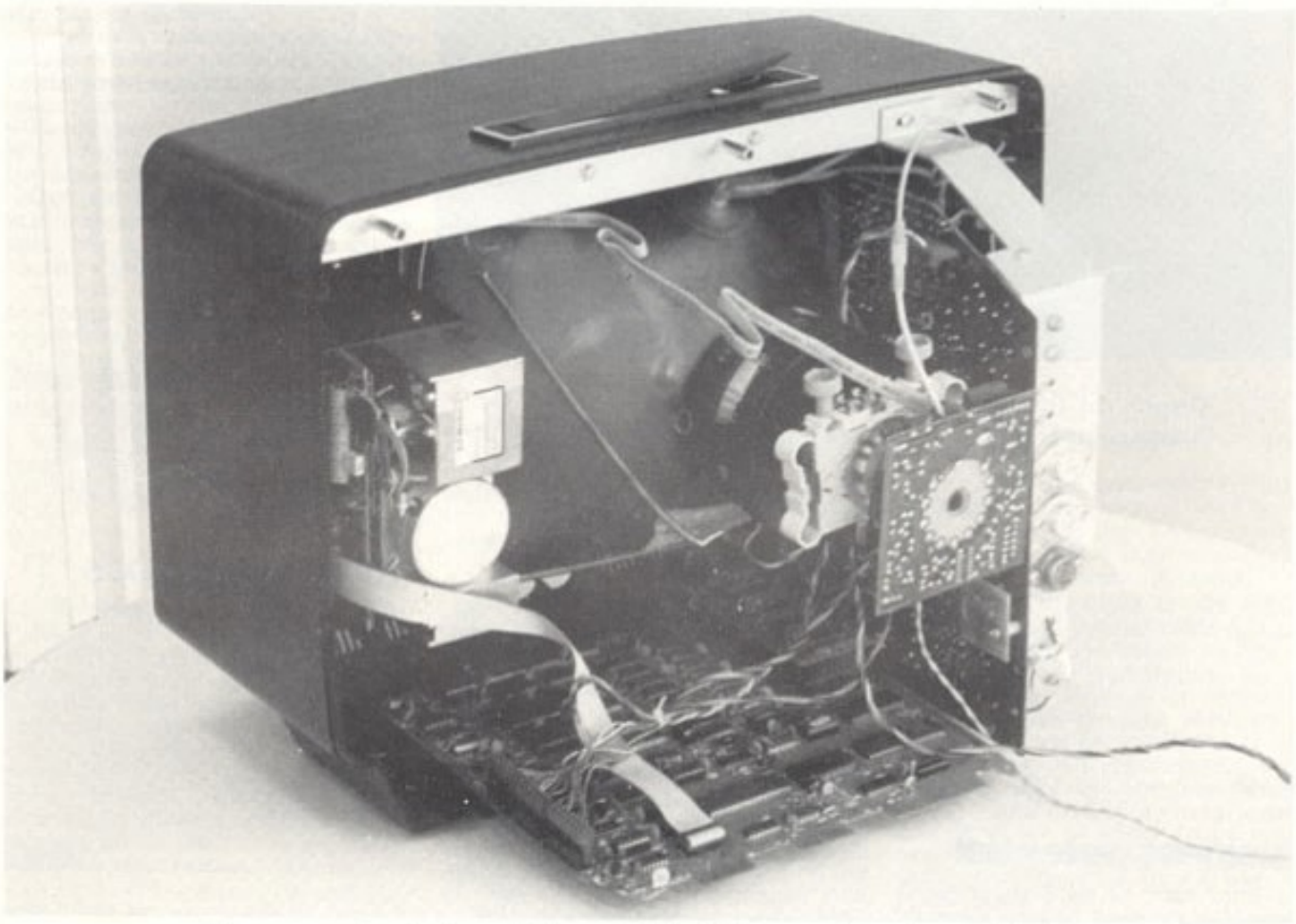
Home and games

Whilst it is not the purpose of this review to study packages in detail, I could not resist having a go at some of the offerings from Compucolor.

I found that the home accounting and housekeeping packages were very American in both spelling and content. It is unlikely that these will be greatly used in Australia. This, of course, means that there is plenty of scope for developing your own, or for someone to develop and market some Australian versions.

Some of the games are really spectacular with, as you might expect, lovely graphics. Unfortunately my joy

Compucolor's big brother: the Intercolor.



The processor board plays neighbours with the CRT . . . we detected no overheating.

at finding these games was marred by the fact that bugs creep into some of them. In fact on one called Concentration (in which you have to remember pairs of numbers) I am sure that Compucolor was deliberately cheating! From time to time it sneakily changed the numbers.

Expandability

There is limited scope for plug-in expansion. At present it consists of a second disc drive (which is needed if the discs are to be copied), an RS232 printer or terminal, music synthesizer, and memory up to 32K. The Compucolor II can, itself, be used as a terminal via the RS232 port. With some technical knowledge an S100 motherboard could be connected to the 50 pin bus, thus allowing all the S100 options.

Technical data

CPU:	8080A. 2MHz
Memory:	8 - 32K dynamic RAM
Keyboard:	70, 101, or 117 key versions
Screen:	13"(?). 8 colours. 16 or 32 lines of 64 characters. 128x128 point graphics
Cassette	N/A
Disc Drives:	Up to 2 drives. 1 head per drive. 5¼" discs. 51.2ch per side.
Printer:	N/A
Bus:	50 pin. (Enough signals to attach an S100 motherboard)
Ports:	RS232. Compucolor II may be used as a terminal via this port. Selectable baud rate.
System Software:	File Control System (FCS)
Languages:	16K Disc BASIC 8001 in ROM. 8080 Assembler.

At a glance

FIRST IMPRESSIONS	
Looks	***
Setting up	*****
Ease of use	****
HIGH LEVEL LANGUAGES	
Basic	****
Cobol	N/A
Fortran	N/A
Pascal	N/A
Other	N/A
System Software	****
PACKAGES	
Business	N/A
Education	**
Home	***
Games	****
PERFORMANCE	
Processor	**
Cassette	N/A
Disc	***
Peripherals	N/A
EXPANDABILITY	
Memory	**
Cassettes	N/A
Discs	**
Bus	**
COMPATABILITY	
Hardware	**
Software	***
DOCUMENTATION	****
VALUE FOR MONEY	***

*****	excellent
****	v. good
***	good
**	fair
*	poor

Documentation

The Compucolor II Programming and Reference Manual is a comprehensive and clearly written document. It contains a thorough description of the BASIC system (including the graphics), a guide to the file control system together with the associated lists of commands and error messages, and the specifications of the major hardware components. Although there are chapters entitled "Essentials for Simple Programming" and "Beginning to Program", this book is not a suitable text for the novice wishing to learn how to program.

The chapter on graphics is not so clearly presented . . . this may be due to the clumsy graphics system.

Memory map

0000	Restarts & tables
0100	ROM BASIC
2100	FCS & CRT ROM
4000	Future space
6000	High speed refresh RAM
7000	Slow speed refresh RAM
8000	Scratch pad RAM
8200	Scratch BASIC RAM
8300	
FFFF	Work space RAM

Prices

At the time of writing the basic Compucolor II with 8K RAM had a list price of $2095 (exclusive of sales tax). The 16K and 32K versions cost $2395 and $2695 respectively. The extended keyboard was $175 extra while the deluxe keyboard cost $260 more than the standard version. The second disc drive was $595. Therefore, the most expensive system (32K deluxe keyboard, 2 disc drives) is $3550.

The programming manual was included in the price and the maintenance manual cost $50. There are some games, an 8080 assembler and finance, utility, engineering, and educational programs available on discs from $25.

Conclusion

At $2095, the Compucolor II offers an extensive keyboard, a 13"(?) eight color CRT (128 x 128), 16K Basic in ROM, 8K RAM (expandable to 32K) and one (or two) slow minifloppy drives. If your needs fall within those criteria, you will find the system good value for money.

Benchmark comparisons

	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8
Compucolor II	2.0	10.9	22.4	23.9	25.7	38.7	55.2	10.8
Sorcerer	1.8	10.0	20.7	22.2	24.3	37.6	53.7	9.6
Apple II	1.3	8.5	16.0	17.8	19.1	28.6	44.8	10.7
Tandy TRS 80	2.5	18.0	34.5	39.0	45.0	67.0	109.0	—
PET	1.7	9.9	18.4	20.4	21.0	32.5	50.9	12.3

BASIC

CHART 1

Statements:	CLEAR	DATA	DIM	END	FILE
	FOR	GET	GOSUB	GOTO	IF
	INPUT	NEXT	ON	OUT	PLOT
	PEEK	POKE	PUT	PRINT	READ
	REM	RESTORE	STEP	THEN	TO
	WAIT				
Commands:	CONT	LIST	LOAD	RUN	SAVE
Mathematics:	ABS	ATN	CALL	EXP	INT
	INP	LOG	POS	RND	SGN
	SPC	SQR	TAB	TAN	
String handling:	ASC	CHR$	LEFT	LEN	MID
	RIGHT	STR	VAL		
Disc handling:	See FCS				

FCS

CHART 2

Logical blocks:		READ	WRITE		
File:	COPY	DELETE	LOAD	RENAME	RUN
	SAVE				
Diskette:	DEVICE	DIRECTORY	DUPLICATE	INITIALISE	

BENCHMARKS

'The proof of the pudding is in the eating' and it is the eventual performance of the hardware/software combination which is of real interest. Apart from the cost aspect one is interested to know what facilities are offered and how fast the system is. Particularly with large scale simulations, speed is a vital factor. The following benchmark programs provide an indication of the speed at which a computer system can execute a particular set of routines. As such, they should be treated with caution. They say nothing about other facilities which may be offered – for example string handling.

The first seven benchmarks were used in a series of tests carried out in the United States and published in an article in the June 1977 issue of *Kilobaud*. The eighth benchmark has been introduced to test the transcendental functions of the various interpreters. Unusually poor performance on this benchmark is a clear indication of the use of poor algorithms and is more a reflection on the programmer than on the machine.

```
BM1   300 PRINT 'S'
      400 FOR K=1 TO 1000
      500 NEXT K
      700 PRINT 'E'
      800 END

BM2   300 PRINT 'S'
      400 K=0
      500 K=K+1
      600 IF K<1000 THEN 500
      700 PRINT 'E'
      800 END

BM3   300 PRINT 'S'
      400 K=0
      500 K=K+1
      510 A=K/K*K+K-K
      600 IF K<1000 THEN 500
      700 PRINT 'E'
      800 END

BM4   300 PRINT 'S'
      400 K=0
      500 K=K+1
      510 A=K/2*3+4-5
      600 IF K<1000 THEN 500
      700 PRINT 'E'
      800 END

BM5   300 PRINT 'S'
      400 K=0
      500 K=K+1
      510 A=K/2*3+4-5
      520 GOSUB 820
      600 IF K<1000 THEN 500
      700 PRINT 'E'
      800 END
      820 RETURN

BM6   300 PRINT 'S'
      400 K=0
      430 DIM M(5)
      500 K=K+1
      510 A=K/2*3+4-5
      520 GOSUB 820
      530 FOR L=1 TO 5
      540 NEXT L
      600 IF K<1000 THEN 500
      700 PRINT 'E'
      800 END
      820 RETURN

BM7   300 PRINT 'S'
      400 K=0
      430 DIM M(5)
      500 K=K+1
      510 A=K/2*3+4-5
      520 GOSUB 820
      530 FOR L=1 TO 5
      535 M(L)=A
      540 NEXT L
      600 IF K<1000 THEN 500
      700 PRINT 'E'
      800 END
      820 RETURN

BM8   300 PRINT 'S'
      400 K=0
      500 K=K+1
      530 A=K↑2
      540 B=LOG(K)
      550 C=SIN(K)
      600 IF K<100 THEN 500
      700 PRINT 'E'
      800 END
```

COMPUTER ANSWERS

Every month in APC, Sheridan Williams will assist readers with their hardware, software and systems difficulties. Some questions he will deal with himself, other enquiries will be directed towards members of his consultancy panel.

Vague on vectors

I'm feeling confused. Could someone please tell me, what are I/O vectors?

I/O or input/output vectors are generally related to a method of 'patching' programs into a system. Most programs make frequent access to I/O devices and these are usually done using separate I/O driver subroutines. Therefore, if I have to change (relocate) the address of my I/O routines due to re-arrangement of a program, for example, then I will also have to go through the entire program altering the 'called' addresses whenever a call to I/O is made — very time consuming without an assembler. Also, due to hardware differences, my driver routines may be different from yours so transportability is a problem. The solution is to pass all I/O calls via I/O vectors. These can be conveniently located at the start of the program thus:—

```
0000            C3 09 00
   Jump to start
0003 Charin     C3 XX XX
   Jump to input sub-routine
         (Z-80 code)
0006 Charout    C3 XX XX
   Jump to output sub-routine
0009 Start      XX XX First
   instruction of program
```

It will return to the correct point in the program automatically. Should I have to alter the address of my I/O subroutines then I only need to change the address at 0003 and 0006. Similarly, if you want to patch the program to suit your own hardware environment then all you need do is insert the relevant address of your I/O drivers, in aforesaid, 0003 and 0006. Simple!

Mike Dennis

Bank-selectable

I've seen reference to bank-selectable memory boards — what is 'bank-selectable'?

'Bank-selectable' memory boards are usually found on S-100 systems (though that is not a pre-requisite). Most 8-bit micros have a limited addressing capability of 64K bytes. Bank-selecting allows you to extend that limit — usually by a factor of 8. This system is particularly suited to Z80 or 8080 based systems that generate special I/O control signals. Each bank selectable memory board has an associated port and a selection switch for Banks 0-7. If the board is to respond to Banks 1 & 2, for example, then switch positions 1 & 2 would be closed. When the appropriate data word is outputted to the necessary port, the data is latched into the port and used to decode the switch settings — thus enabling that board for a selected bank. Outputting a different data word to the port can effectively enable and disable bank selectable boards all under software control.

Mike Dennis

Random Confusion

What is the point of 'random access' files? If the files are random, how do you know where each record is stored?

I think the reason that you are confused is because of the word *random*. I prefer the term 'Direct access' to 'Random access'; the two terms are synonymous. I can only imagine that the term *random* access was coined because it does not matter in which order you access the records in the file. I much prefer to think of the file as a *direct* access file because you can access any record directly without first having to read all the previous records.

Your question about how do you find a record — this is answered fairly simply now. You only need a way of linking the 'key field' in the record to the disc address. This is known as the 'randomising algorithm' (there's that word again). The disc operating system usually takes care of the track and sector numbers, and all you have to do is work out the relative address (relative to the start of the file and record length). An example would be if you had a file of part numbers. For certain goods you could make your part numbers run from 0001 to 9999 say, and hence part number 1234 would be found at record number 1234.

Problems arise where the key field is a name. Where on a file of 26000 people would you place SMITH. Well, if the file is fairly well balanced, one idea is to start each letter of the alphabet at intervals of 1000 records, and each second letter in the name could start at 1000/26 intervals. Hence Smith would be placed at a record calculated by 19x1000+13x38=19494 (S=19th letter, M=13th). This is just one of many ideas, although obviously it can be wasteful of space.

Sheridan Williams

Plotting Lissajous

How do I plot Lissajous figures on my micro? I have seen them done but have no idea how to program them myself. Do I need a great knowledge of mathematics and physics?

Lissajous figures are nothing more than two mutually acting simple harmonic motions. An example might be the pattern formed when a pendulum swings in two planes (not just backwards and forwards, but from side to side as well), and has sand pouring out of the pendulum's bob. The trace made by the sand on the floor will be a Lissajous figure. They are fairly simple to plot provided that you don't want them plotted on a teleprinter. If you have direct cursor addressing on a VDU then you will find the task easy. Here is a program for the Research Machines 380Z, plus suitable mods for other machines like Apple and PET.

```
10 INPUT"SCREEN WIDTH"
;W
20 INPUT"SCREEN
HEIGHT";H
30 W=W/2: H=H/2
40 INPUT"TWO PARA-
METERS",A,B
50 GRAPH 1
60 FOR T=0 TO 9999 STEP
0.01
70 X=W*SIN(A*T)+W
80 Y=H*COS(B*T)+H
90 PLOT X,Y,2
100 NEXT
```

For the Apple change 50 HGR and 90 PLOTX,Y. For machines like the PET with no plotting command you will have to calculate the screen address and use 90 POKE V+2*W*Y+X,46 where V is the screen base (top left hand corner) address, and the ASCII code for a dot is 46. I won't do any more for you as half the fun is making the program work; please, no letters saying that the program doesn't work — make it work!

Lissajous figures can be stated parametrically as x=sin at, y=cos bt; and t can have any value (although it is convenient to use the values in line 60 above). Values of *a* and *b* will give differing forms of pattern, choose simple small integer values to begin with. Have fun.

Please send in more scientific questions as I'd like to include at least one per month.

Sheridan Williams

Operating systems

I have heard the name CP/M used a lot in connection with operating systems. What is an operating system, perhaps you could enlighten me as to its purpose.

As long as a computer comprises the CPU and VDU only, there is very little point in having an operating system. Once the computer becomes the centre of a computer system i.e. surrounded by peripherals such as magnetic tape, magnetic disc, printer, and maybe even paper tape and punched card input/output, it then becomes increasingly useful to have an operating system.

An operating system is a group of programs designed to increase the productivity of a computer system. Some of the programs may decrease the amount of idle time, especially on a multi-user system and others reduce the amount of programming that needs to be done by a computer user. Strictly speaking there is an 'Executive' program that resides in store calling other parts of the operating system as and when necessary, from disc.

As you have asked specifically about CP/M I will use this as an example. CP/M will reside on disc and parts of it may be called in when required. There will be a number of system commands and here are a few of them: ERA will erase specified files, DIR lists filenames present on the directory, REN renames specified files, TYPE will type the contents of the specified file on screen or printer. There are also what are called 'transient' commands. These can be extended by the user but several are supplied with the CP/M package, these include: STAT which lists statistical data about free space on disc, PIP which allows transfer operations between peripherals, DUMP will dump the contents of a file in hexadecimal etc.

CP/M will use a couple of tracks on the disc, but you would expect it to, as it *is* a program.

Sheridan Williams

DICK gives you more – without compromising

NOW EVEN BETTER!

Sorcerer Mk II

As part of their on-going research and development, Exidy have made some changes giving the Sorcerer on-board capability of up to 48K RAM – plus greater reliability. You still get the interchangeable ROM PAC™ feature, high resolution graphics of 122,880 pixels (nearly twice that of any rival computer), both upper and lower case characters, graphics, I/O ports as standard, S-100 expansion capability, 2 cassette player interface plus many more outstanding features.

Mk II

16K RAM

$1,395.00

Terms are available to approved buyers through BFC Finance

FANTASTIC VALUE

P&P $5.50 Cat. X-3001

Micropolis™

X-3205 **$1350**

X-3208 **$750**

If you have a Sorcerer with an S-100 expansion unit we can offer you the best mini-floppy disc facility: the state-of-the-art Micropolis™ 1043/1023 quad density system. You can have from one to four drives, giving up to 1260K bytes of storage at incredibly low cost. This storage is achieved by packing no less than 77 tracks each of 16 storage sectors onto each disc, not the usual 35 or 40 tracks with only 10 sectors. Start your system with the 1043/mod 2. which comes complete with inbuilt power supply and disc controller board for the S-100 expansion. Extend the system by adding up to 3 of the 1023/mod 2. add-on drives. The 1043/mod 2. also includes user manual, cables, two discs (134mm) with MDOS and extended BASIC, an assembler, editor, debugger and other utilities.

1043/mod 2. drive with controller Cat. X-3205

1023/mod 2. add-on drive Cat. X-3208

P&P $5.50 ea.

ROM PACs

Word Processor PAC™

Add to your Sorcerer for a system that is more powerful than other available systems. It includes instruction manual and features auto text wrap, auto checking drastic commands, powerful search functions auto commands and macro programming, single key commands plus much more.

P&P $5.50

Cat. X-3085 .... **$275**

Development PAC™

Turns your Sorcerer into a dedicated development tool. Includes instructions and features designer's debugging tool, text editor, Z80 assembler, linking loader, I/O driver routines.

Cat. X-3090 .... **$139.50**

P&P $5.50

Cassette Recorder

COMPUTER QUALITY

This National cassette recorder has the quality to store and load data with accuracy. Get 1st class results from your Sorcerer with this unit. AC or DC operation.

Cat. A-4095

P&P $4

$79.95

system for your dollar
on quality!

Exidy S-100 Expansion Unit

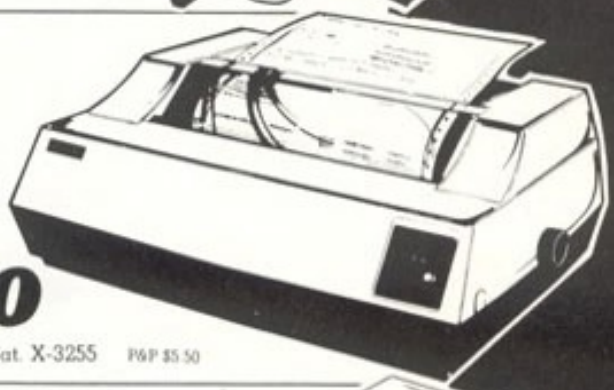
The Exidy S-100 expansion unit opens the way to almost unlimited expansion of your Sorcerer system, because more than 100 manufacturers world wide produce plug-in PCB card modules designed for the S-100 bus system. There are S-100 plug-ins available for almost every conceivable avenue of system expansion. Memory cards, floppy disc and hard disc controllers, EPROM programmers, special I/O interfaces and power control cards, colour video graphics controllers, D-to-A and A-to-D converters, music and speech synthesis cards, even speech recognition cards plus more being announced all the time. Supplied complete with inbuilt power supply, ribbon cable and connector.

Use other manufacturers peripherals with your SORCERER

$499.00 P&P $5.50 Cat. X-3010

ECONOMICAL PRINTER

The C.ITOH model 8300P dot matrix printer is a high performance unit that incorporates the latest micro-processor technology. This unit is priced below many printers offered by other companies but if you want a no-nonsense printer that can churn out the full 96-character ASCII at a brisk 125 characters per second on standard fan-fold paper, then you can't do better. Character spacing of 80, 40 or 132 columns – software selectable with inbuilt 80 byte character buffer and self testing string generation. Interface 7-bit parallel Centronics type, this is available as Cat. X-3112 @ $49.50.



$970 Cat. X-3255 P&P $5.50

Universal Monitor
WHY PAY MORE?

Suits virtually all systems including The Sorcerer, Tandy TRS-80, Apple, System 80 etc. Compare our prices to theirs! Large screen (30cm) with jitter free, distortion free characters. Simple connection plus dual power – 240V AC or 12V DC.

$149.50 Cat. X-1196 P&P $5.50

TANDY OWNERS – DON'T MISS OUT

This superb Dick Smith Mini Disc Drive provides 228% the storage capacity of the Tandy drives yet costs far less! Check these features – ● 133.4mm minifloppy diskette ● 40 tracks instead of 35 – 14% more storage per side ● soft sectored ● dual sensors for reading on both sides – doubles the storage capacity ● recording density (inside track) 2768bpi ● 100K bytes per side ● drive number is switch selectable ● use as first drive or add-on drives for TRS-80. (Save $1000 plus on a 4 drive system compared to the January 1980 Tandy price).

$379 Cat. X-3230 P&P $5.50

Power Supply

Cat. X-3234 **$60.00** P&P $4

Especially suited to the Dick Smith mini disc drive. The unit has dual voltage and will power TWO drives.

Connecting Cable

Cat. X-3232 **$49.50** P&P $3

34 way daisy-chain cable for controlling 4 disc drives to your TRS-80.

DICK SMITH ELECTRONICS

NSW	125 York Street	SYDNEY	Ph 290 3377
	147 Hume Highway	CHULLORA	Ph 642 8922
	162 Pacific Highway	GORE HILL	Ph 439 5311
	30 Grose Street	PARRAMATTA	Ph 683 1133
	396 Lane Cove Rd	NORTH RYDE	Ph 888 3200
	263 Keira Street	WOLLONGONG	Ph 28 3800
	613 Princes Highway	BLAKEHURST	(Opening Soon)
VIC	399 Lonsdale Street	MELBOURNE	Ph 67 9834
	656 Bridge Road	RICHMOND	Ph 428 1614
QLD	166 Logan Road	BURANDA	Ph 391 6233
	842 Gympie Road	CHERMSIDE	Ph 59 6970
ACT	96 Gladstone Street	FYSHWICK	Ph 80 4944
SA	60 Wright Street	ADELAIDE	Ph 212 1962
WA	414 William Street	PERTH	Ph 328 6944

bankcard welcome here

SHOPS OPEN 9AM to 5.30PM (Saturday: 9am till 12 noon) BRISBANE: Half hour earlier. ANY TERMS OFFERED ARE TO APPROVED APPLICANTS ONLY

RE-SELLERS OF DICK SMITH PRODUCTS IN MOST AREAS OF AUSTRALIA.

MAIL ORDER CENTRE: PO Box 321, NORTH RYDE NSW 2113. Ph 888 3200. PACK & POST EXTRA.

Prices correct and stock available at time of going to press. To avoid disappointment please visit nearest store for stock situation.

DSE747

COMPUTER GAMES

Chess master, David Levy, begins a series of articles on the principles behind programming computers to play games.

Games are fun, but some games are more fun than others, depending on your taste. It's long been recognized that the type of mind required to play good chess, bridge, backgammon or poker, is also likely to be adept at solving crossword puzzles and writing computer programs. Hence it is hardly surprising that many programmers derive enormous satisfaction from programming intelligent games.

In this series of articles I shall discuss the principles of programming a computer to play games, placing special emphasis on the particular problems posed by running these programs on a micro. My aim will be to acquaint the reader with the techniques of games programming so that (s)he will have the confidence and ability to program any intelligent game for a personal computer. Although I shall use a limited number of games in my examples, the same general principles can be applied to any game in which the computer competes against the user or users.

The series will be divided into three parts. The first part will cover all the general principles, giving examples and suggesting interesting programming tasks for the more enthusiastic reader who wishes to test his understanding of a particular topic. In part two I shall discuss some specific games in more detail and describe what work has been done in these areas so that the reader who is interested in a particular game need not re-invent the wheel. I shall also invite readers to write to me with their questions and ideas, and I shall publish the most interesting letters together with my comments (though I regret that no personal replies can be given). The third part of the series will begin when the most interesting games have already been discussed in detail, and it will be possible for me to devote most of each article to the readers' forum.

I very much hope that these articles will be interesting and informative for all of you who are 'into', or would like to be into, computer games.

Input/output

I/O on a personal computer is often largely a matter of taste, though certain points are worth bearing in mind when writing a game playing program:

1 The output should be easy to follow. You may not think this important, and many programmers take the attitude that if they can understand their output nothing else matters; but how about someone else? If you want to show your program off to a friend it will be so much better received if the output is clear, concise and unambiguous. Remember to output any information that may be helpful, for example in a chess program you should always announce check, checkmate and stalemate. These little touches take hardly any extra effort, and they make your program that much more attractive to another user.

2 If you want to use neat graphics or printout, plan the layout carefully, taking into consideration all possibilities. It's not much use having your bridge program display pretty pictures of the cards if one day you discover that when you are dealt ten cards in a suit only nine of them will fit onto one line and your whole display is messed up.

3 Ensure that the user can easily see whose turn it is to play, and what the

last 'move' was. It can be infuriating to leave the computer for a minute or two and then return to find that the program has moved but you do not know what it has done.

4 Make it easy for the user to enter a move and to clear an incorrect move entry.

5 Ensure that the program will reject an illegal, impossible or ambiguous move, or any entry that does not conform to your simple input rules.

One-person games

A one-person game does not involve an opponent. You play against a microcosm of the forces of nature and if you make a mistake it may be possible to recover, and then go on to win. Solving a problem or a puzzle is a good example of a one-person game — when you get near to a solution there is no-one to oppose you by suddenly making the problem more difficult. It may seem at first glance that patience games are one-person games, but in fact many patience games do not permit the player any freedom of choice, so the 'game' has no real interest. Once the cards are cut the player either will or will not finish the game, and all of his decisions are made for him by the rules.

A well-known one-person game, is the 8-puzzle, in which a 3 x 3 array of tiles contains the numbers 1 to 8 and an empty space. (The numbers are sometimes replaced by letters.) The player shuffles the tiles and then tries to reach some target position by successively moving tiles into the empty space. For example:

STARTING CONFIGURATION

3		8
2	5	7
1	4	6

TARGET CONFIGURATION

1	2	3
4		5
6	7	8

Here the task is simple, and one way in which the target can be reached from the starting configuration is by moving the tiles in the following order: 3,2,1,4, 6,7,8,3,2,1,4,6,7,8,5. With other starting and target configurations the task may be more difficult, and for those who find the 8-puzzle too simple there is always the 15-puzzle, in which a 4 x 4 array has fifteen tiles and an empty space; then there's the 24-puzzle, the 35-puzzle and the $(n^2 - 1)$-puzzle. In fact there is no reason, other than tradition, why the puzzles need to be square.

Heuristics and Algorithms

The 8-puzzle is an excellent example of the type of problem that lends itself to solution by heuristic means. Before describing how we should set about programming games of this type, it would be as well to distinguish between the terms 'heuristic' and 'algorithm', which are often misunderstood.

An *algorithm* is a technique for solving a problem (the problem may be finding the best move in some game) if a solution exists. If there is no solution to the problem the algorithm should determine this fact. Thus, an algorithm always works, otherwise it is not an algorithm.

Most interesting games do not have an algorithmic solution, at least in the practical sense. Of course there is an algorithm for finding the perfect move in a game of chess — simply examine every possible move for both sides until one player is mated or a draw is established — but since the total number of chess games is greater than the number of atoms in the universe, this algorithm would take somewhat slow in practice. In contrast, however, there does exist a useful algorithm for the interesting game of Nim. Nim is played with a number of piles of objects, often matches, and with various numbers of objects in each pile. The players move alternately, and to make a move a player must remove, from one and only one pile, any number of objects he chooses — from one object to the whole pile. The player who removes the last object loses the game. (In another version of the game the player who takes the last object is the winner.)

In order to win at Nim one need only know the following algorithm, and a few exceptional cases: *If the number of objects in each pile is expressed in binary, and each binary column of numbers is added in decimal (without carrying numbers), then if the decimal totals are all even or zero then the person who is next to move is in a losing position.* Here is an example.

			binary
Pile A:	1 1 1 1 1 1 1	= 7 matches =	1 1 1
Pile B:	1 1 1 1 1	= 5 matches =	1 0 1
Pile C:	1 1 1	= 3 matches =	1 1
Pile D:	1	= 1 match =	1
		totals:	2 2 4

All three totals are even so whoever moves next will lose, provided that his opponent plays correctly.

There are some obvious exceptions to the rule. For example if piles A, B, C and D each have one match then the player who moves next will win, and the same is true of a position in which there's only one pile of matches, provided that there are at least two matches in this pile.

The existence of this algorithm does not detract from the interest of the game since its implementation is somewhat difficult for a human being, unless the number of piles and the number of matches in each pile is small. But for a computer program the task is trivial. The program considers each move that it can make, taking one match from pile A, two matches from pile A, and so on, and it evaluates each of the resulting positions until it finds one where the decimal totals of the binary columns are all even or zero, whereupon it makes the move leading to that particular solution. Once a candidate move has been rejected it may be thrown away, so RAM is required only for the current situation, the move or decision currently under consideration, and workspace for the binary/decimal calculations. The program tries each move from the current position, and if a move is found to be unsuccessful it is 'unmade', and the next move tried. In this way it is not even necessary to store both the current position and the candidate position — the program can switch to and fro between them by making and unmaking moves, a technique which is useful for saving RAM in a highly restricted memory environment.

One trick to remember for Nim, or any other game with an algorithmic method of play, is this. Should the program find itself in a theoretically losing position, as might happen at the start of the game, it should make the move that leaves its opponent with the most complex decision. In this way the opponent is more likely to make a mistake. In Nim I would suggest that if your program is in a losing position it should remove one match from the largest pile.

A *heuristic* method of solving a problem relies on commonsense techniques for getting closer and closer to the solution, until the solution is actually within sight. A heuristic is therefore a rule of thumb — it will usually help us to find a solution to the problem, but it is not guaranteed to do so. In situations where a heuristic does work, it will often find the solution much faster than any algorithmic method, though some heuristics, for best results, are often employed in conjunction with an algorithm. A frequently used device which makes use of heuristics is the *tree*, and we shall now examine a method of solving the 8-puzzle by use of a tree and a simple heuristic.

Let us return to the starting configuration on figure 1. We always refer to the starting configuration, or the point from which the program must move, as the *root* of our tree. Before we can decide which move might be best we must know which moves are possible, i.e. in accordance with the rules of the game. A list of these moves is usually supplied by a subroutine called a *legal move generator*, which may be extremely complex, as in chess, or very simple, as in the 8-puzzle. It is not difficult to see that in our starting configuration there are three tiles which may be moved, 3,5 and 8. Our legal move generator would determine these moves by examining the elements of the 3 x 3 array which are horizontally or vertically adjacent to the empty space, and there are many simple methods for doing so. We might, for example, store all the legal moves in a table. If we number the elements of the array table thus:

1 2 3
4 5 6
7 8 9

our table of moves might look like this:

vacant	*moves*
1	2,4
2	1,3,5
3	2,6
4	1,5,7

etcetera

so that by knowing which element in the array was vacant the program could immediately list the legal moves. This type of approach is called *table-driven move generation*. It is often the fastest way to generate the moves but for some games it consumes too much program memory for it to be a feasible proposition.

Having generated the moves 3,5 and 8 from our starting configuration, we can now begin to see the tree grow.

The *branches* of the tree are the moves (m_1 m_2 m_3) that can be made from the root of the tree. We may denote the root position by P_0, the position arising after making the move m_1 is P_1; after making the move m_2 it is P_2, and after m_3 it is P_3. These positions are represented on the tree by *nodes*.

The program now looks to see if it has solved the problem, and if it had done so it will output the move leading

C.I.S.A.
159 KENT STREET, SYDNEY

The one-stop TRS-80 Microcomputer Shop Providing a total service to TRS-80* users.

(* Tandy Trade-Mark)

HARDWARE

Having inexplicable system crashes?
Plug-in our line-filter and get main-frame stability $ 55.00

Automatic telephone dialler/silent burglar alarm
Requires no hardware modifications
Software driver included . . . . . $ 59.50

Light-pen-complete with demonstration software and full programming instructions.
Compare OUR price! . . . . . $ 19.50

Lower-case modification for Electric Pencil
Sends us your keyboard by COMET
Fitted and 90 day guarantee . . . . . $ 49.95

Video stabiliser crystal
Stabilise your video output to professional standards
Crystal and instructions only . . . . . $ 19.95
Fitted and guaranteed (send video and keyboard) $ 39.00

16K upgrade kits with instructions (250ns chips). $ 85.00
We install (send keyboard or interface) $105.00

RS-232 printer-driver unit including software patch (includes 20 MA loop for teletype) . . . . . $ 54.75

CISA Data-debugger. This is manufactured in Australia and we believe it to be one of our best products. Plugs between cassette and keyboard. No modifications required. Reads and re-records tapes that are normally unreadable by any other method. . . . . $ 57.50

Cassette cabinet. Protect your valuables in this strong handsome cabinet. Holds 36 cassettes. . . . $ 27.75

Digital cassettes – top quality

C10s	1 – $1.65	10 – $14.85
C20s	1 – $1.85	10 – $16.65
C30s	1 – $1.99	10 – $18.50

5¼" diskettes –
1 – $4.95 10 – $45.00

Prime quality diskettes for business and commercial applications and for that critical data.
1 – $8.50 10 – $77.50
All tracks guaranteed to load!

We stock a complete range of printer ribbons including black Anadex. Full list in our catalogue.

We will shortly have an extensive range of computer paper, labels and pre-printed forms for the enthusiast and the business user – watch this space!

BOOKS

We believe that we have the best and most comprehensive range of titles of interest to the TRS-80 owner/user. Over 200 titles presently in stock.

Highly Recommended

BASIC and the Personal Computer – Dwyer and Critchfield . . . . . $ 14.95
Programming in Pascal – Grogono . . . . . $ 11.95
The Little Book of BASIC Style – Wereson . . . . $ 7.25
Problem Solving and Structured Programming in BASIC – Hoffman & Friedman . . . . . $ 12.95
Z80 Microcomputer Handbook – Bardin . . . . . $ 11.95
Microprogramming and Software Development – Duncan . . . . . $ 34.95
Introduction to the TRS-80 Computer – Zabinski . . . . . $ 14.95
Microcomputer Primer – Waite . . . . . $ 10.75
60 Challenging Programs with BASIC Solutions – Spencer . . . . . $ 8.50
Sargon – A Computer Chess Game – Spracklen . . . . . $ 20.00
Game-Playing with BASIC – Spencer . . . . . $ 11.00
BASIC Wordbook for Beginning Programmers – Schomann . . . . . $ 8.00
Calculators and Computers – A Source Book of Activities $ 10.95
Microcomputers and the Three Rs – Doerr $ 10.00
S100 Bus Handbook – Bursky . . . . . $ 10.00
Microprocessor Data Manual . . . . . $ 10.00
Small Computer Systems Handbook – Libes . . . $ 11.50
Pascal with Style – Legard and Nagin . . . . . $ 8.50
Basic BASIC – Coan . . . . . $ 12.00
Advanced BASIC – Coan . . . . . $ 12.00
Z80 and 8080 Assembly Language Programming – Spracklen . . . . . $ 10.00

SOFTWARE

EXTENDED BASIC including printer-driven routine, disc only. Single-key addressing of practically all BASIC words, plus repeat key. Many other features. Send us a copy of your DOS and we will patch it. . . . . $ 9.95

INFINITE BASIC from RACET . . . . . $ 29.95
INFINITE BUSINESS (requires Infinite Basic) . . $ 17.97
REMODEL . . . . . $ 24.95
REMODEL + PROLOAD . . . . . $ 34.95
GSF . . . . . $ 24.95
DOSORT . . . . . $ 34.95
COPSYS (Copy system tapes) . . . . . $ 34.95

All RACET COMPUTES professional standard system programs – full details and description in our catalogue.

NEWDOS from APPARAT
35 track . . . . . $ 99.00
40 track $110.00

This DOS is the finest we have seen for a microcomputer. Has features and facilities previously restricted to much larger systems.

TARANTO and ASSOCIATES
General Ledger $155.00
Accounts Payable $155.00
Accounts Receivable $155.00
Print Invoice . . . . . $ 99.00
Complete Business System $500.00

This is professional quality software and really provides a superb business system for the TRS-80. Software maintenance is available to purchasers of the complete system at $15.00 per hour, to tailor the programs to your specific needs.

GAMES

Sargon $25.00
Sargon II $37.50
Game Playing With Basic 3 Tapes $12.50
Biocurve $12.50
Keynote (Micromusic) $12.50
Strategy Games (5) . . . . . $ 7.95
Space Games (4) . . . . . $ 7.95
Adventureland $14.95
Ecology Simulations (32K and Disc) $24.95
Sketchmode $15.00
Gridiron $16.50
Batter Up $14.00
Worg (4 space games) . . . . . $ 9.95

We are the Australian agents for many first-class U.S. software houses – send for our full listing.

LATE SPECIAL!

PASCAL now available (Cassette I/O only). $49.50

We regularly conduct a series of evening periods of small group tuition from 7.30 p.m. onwards in BASIC with specific emphasis on commercial applications.
These sessions are conducted in a friendly and helpful atmosphere and are designed to give you a complete mastery of BASIC techniques up to and including all aspects of disc I/O and file handling.
We will teach you tricks even Tandy don't know about!
All stationery and workbooks are provided. Textbooks available at an extra charge.
Fee for 10 periods (up to 3 hours each) $150.00
Special private or group tuition on any aspect of BASIC or commercial/business applications of microcomputers available on request.
Tuition includes assignment work if required – and assessment.
A certificate of competance is awarded on successful completion of a tuition series.

REMEMBER – for all your TRS-80 needs – phone, write or call

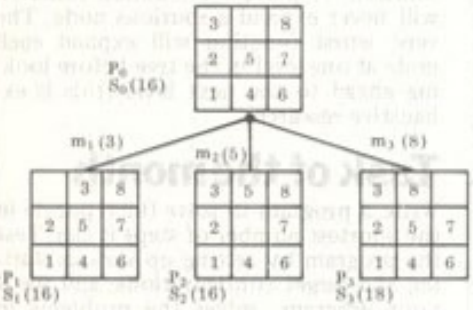
CISA

159 KENT STREET,
SYDNEY 2000
(opposite IBM building)
Phone: (02) 241 1813

We are currently preparing a correspondence course for out-of-town TRS-80 users. Details will be available shortly, shortly.

We carry the full range of MICRO-80 newsletters and recorded software plus a range of popular micro-computing journals.

to the solution, followed by a statement to the effect that the game is over and it has found a solution in however many moves, which are then listed. If it has not solved the problem the program might then like to know how close each of its moves has come to providing a solution, in which case it must evaluate each of the resulting positions. This is done with a device known as an *evaluation function* (or scoring function), which supplies a numerical score that represents nearness to or distance from a solution.



A simple evaluation function for the 8-puzzle can be programmed by counting how many vertical and horizontal places each tile is away from its target location, and summing them. This use of the so-called 'Manhattan Distance' is quite common in the computer solution of similar problems. If we examine our starting configuration we can see that:
the 3 is two places away from target
the 8 is two places away from target
the 2 is two places away from target (1 horizontally, 1 vertically)
the 5 is one place away from target
the 7,1,4 and 6 are all two places away,
and the empty space (do not forget it) is one place away.

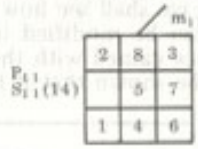
So the total of the Manhattan Distances is (2 x 1) + (7 x 2) = 16, and this is the score, S_0, which is associated with position P_0.

Counting the Manhattan Distances in P_1, P_2, and P_3 we get:
$S_1 = 16$
$S_2 = 16$
$S_3 = 18$
(Note that when a solution is found, S will be zero.)

So on the basis of our evaluation function it looks as though moves m_1 and m_2 are likely to lead to a faster solution than m_3, since positions P_1 and P_2 seem nearer the target position than does P_3. And this is where the story really begins.

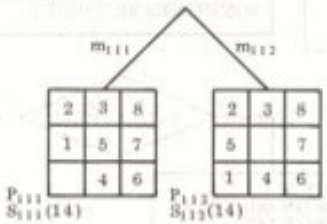
An obvious, though tedious, algorithmic solution to this problem is to look at each of the positions P_1, P_2 and P_3, then generate all the legal moves from each of these positions — look at the newly resulting positions, then generate all the moves from these positions, and so on, until one of the positions is found to be the target (i.e. its score S, the sum of the Manhattan Distances, will be zero). Eventually, this method (which is called *exhaustive search*) will find a solution, that is so long as the program does not run out of RAM. But by using a simple heuristic we can head the program in the right direction, and hopefully a solution will be found sooner than if the exhaustive search algorithm were used.

We have seen that when we expand the node P_0, of the three new positions that appear on the tree, P_1 and P_2 appear to be more promising than P_3. It is clearly logical to expand the more promising nodes before the less promising ones, so at first we should neglect P_3 and concentrate on P_1 or P_2. Since they are of equal apparent merit, the program may choose between them at random. Let us assume that it chooses to expand P_1, from which it will generate the moves of the 2 tile and the 3 tile. Since the 3 tile was moved on the previous turn, and the program is intelligent enough to know that it does not want to go back to where it has just come from, the only move (m_{11}) that the program needs to consider seriously is the move of the 2 tile, which would lead to the following position:



which we denote by P_{11}, and which has a score (S_{11}) of 14.

The best position now on the tree, i.e. the position closest to the target configuration, is P_{11}, since its score of 14 is lower than the scores of all the other nodes. So remembering not to allow the retrograde move of the 2 tile, the program now expands position P_{11}, and the choice is to move the 1 tile or the 5 tile, giving rise to the following position:



Once again we have a tie, two 'best' positions with scores of 14, and so the program again makes an arbitrary choice.

This process continues until a solution is found. It is easy to see that the method can hardly fail to be substantially faster than the exhaustive search process described earlier. The tree is grown intelligently, rather than in a dumb-ox manner, and better use is made of the available memory. With the exhaustive search process the computer's memory will, unless a solution is found, be filled at a stage when a very large proportion of the nodes on the tree are not of any real merit. With the heuristic approach, when memory is exhausted we at least know that most of the memory has not been wasted on unlikely moves, and we can use the best sequence of moves found so far.

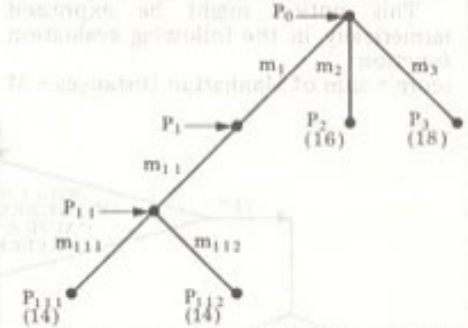
What to do when Memory is Exhausted

Working with a personal computer inevitably poses memory constraints on a different scale from those encountered when writing for a large machine. How can the programmer combat this problem when examining large trees in an attempt to solve a one-person game? I shall describe two approaches to this particular problem:

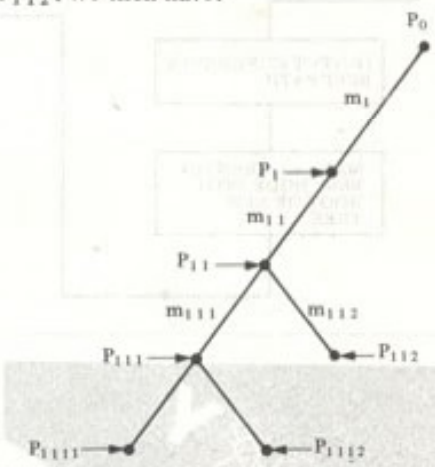
(1) Follow a path through the tree to the best position found so far and output the moves on this path. Then make this 'best position' into the root of a new tree and start again.

(2) More intelligently, when memory becomes full, delete the currently 'worst position found so far' and use the newly scrubbed bytes to store the next position that the programme generates. If this process is continued for long enough, either a solution will be found or the tree will eventually have two paths, each path having no offshoots. When that happens the program must choose the best of the paths, and make the terminal position on this path into the root of the new tree, remembering to output all the moves on the path leading to this position.

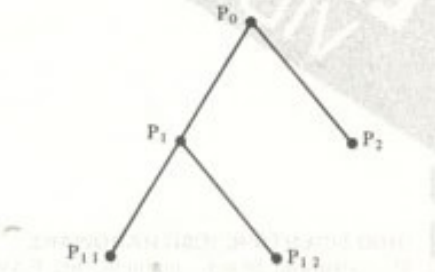
For example, our tree generated for the 8-puzzle now looks like this:



If memory is now full the program would delete m_3 (and P_3), to make room for the successor position produced when it expands P_{111}, or P_{112}. Let us assume that both m_2 (P_2) and m_3 (P_3) are deleted, to make way for P_{111} and P_{112}. We then have:



and the program can now output the moves m_1 and m_{11}, making position P_{11} the root of a new tree.



The new P_0 is the old P_{11}
The new P_1 is the old P_{111}
The new P_2 is the old P_{112}
The new P_{11} is the old P_{1111}
The new P_{12} is the old P_{1112}
And thus the search for a solution continues.

The shortest solution

In most games it is sufficient to win, but there may be reasons why one wishes to win as quickly as possible. For one-person games there exist

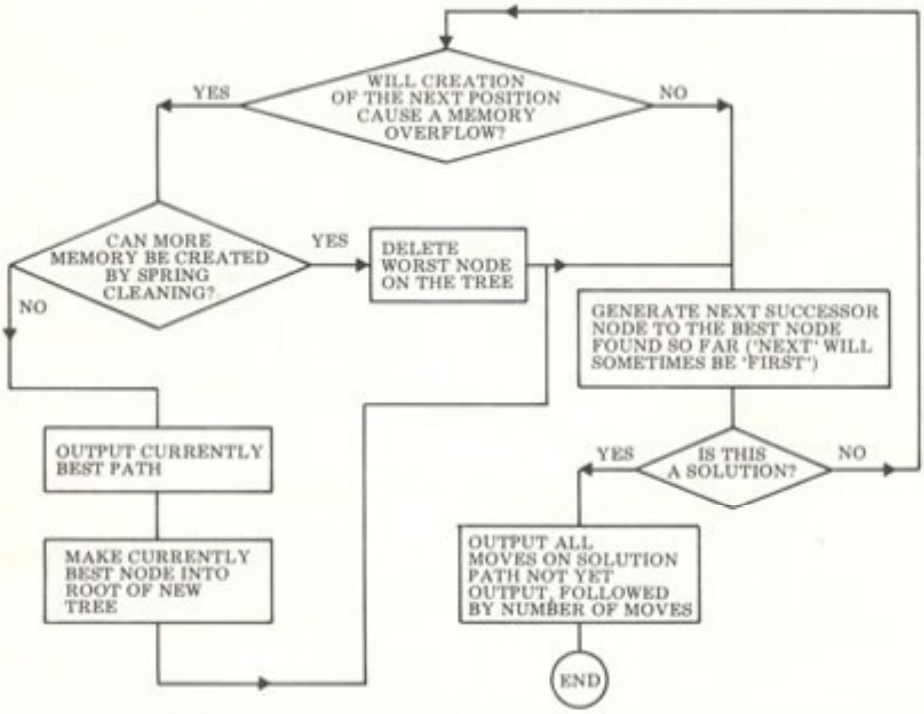
various refinements on this method of tree searching which are likely to produce such a result.

The underlying philosophy in the search for a speedy solution is the notion that it is not only important how near (or far) you are from victory, it also matters how many moves it took you to get there. With the 8-puzzle, for example, a ten move sequence leading to a position with score 12, may not be so likely to lead to a short solution as a two move sequence leading to a score of 13 — perhaps in the next eight moves it will be possible to improve on the 13 by more than 1, thereby finding a shorter route to the solution.

This notion might be expressed numerically in the following evaluation function:

score = sum of Manhattan Distances + M

where M is the number of moves needed to reach this position. Whether or not this expression is the best method of relating the score to effort invested and achievement realised, can only be determined by trial and error. Perhaps M should be replaced by ½M or by 2M, or some other function of M. Playing around with the evaluation function in this way, changing the terms in the function, is one of the delights of game playing programming. When you hit upon a really good evaluation function and you see the program's performance improve dramatically as a result, there is a feeling of exhilaration, rather like watching your child crawl for the first time. In a later article we shall see how evaluation functions can be modified in the light of experience gained with the program, and it will be shown that it is even possible for the program itself to learn from its mistakes and modify its own evaluation routine!



Flow chart

A generalised global flow chart for the search of a one-person game tree is given below. Remember that the most creative part of the work lies in finding a good evaluation function, and the performance of your function can be measured by the number of spurious nodes that are expanded *en route* to a solution. A perfect evaluation function will never expand a spurious node. The very worst function will expand each node at one level in the tree before looking ahead to the next level (this is exhaustive research).

Task of the month

Write a program to solve the 8-puzzle in the shortest number of steps it can. Test the program by setting up various starting and target configurations, and see if your program solves the problems in fewer steps than you do. (Probably neither you, nor your program, will be as fast as Bobby Fischer, who can solve these puzzles with phenomenal rapidity.) When trying the problems yourself remember not to cheat — if you move a tile and then change your mind and move it back, add two to your count.

Bibliography

Nilsson, N.J: *Searching Problem-Solving and Game-Playing trees for minimal cost solutions.* Proceedings IFIP Conference 1968, vol. 2, pp. 1556-1562.

Schofield, P.D.A: *Complete solution of the 'Eight-Puzzle'.* Machine Intelligence 1 (Ed. Collins, N.L. and Michie, D.), Oliver & Boyd, 1967, pp. 125-133.

Slagle, J., and Bursky, P: *Experiments with a Multipurpose, Theorem-Proving Heuristic Program.* Journal Association Computing Machinery, vol. 15, no. 1, pp. 85-99, January 1968.

EDUCATION REPORT

by Miriam Cosic

The Victorian Education Department's aim is to get computing facilities into every high school in the State. So far more than seventy high schools have their own computers. A further sixty-odd have access to larger systems at tertiary institutions and other schools through batching, on line or being lent equipment.

Maribyrnong High School, for example, has a Wang 200. Representatives from a number of schools in the area bring in cards for batch processing. One way or another, some 54% of Victorian schools are into computer education.

The Secondary Computer Education Committee (SCEC) was set up by the Secondary Division of the Education Department in response to enquiries from principals, teachers and school councils about setting up facilities. It includes members from various disciplines to give a rounded approach; and makes recommendations to the Director of Secondary Education on matters concerning computer education. These include allocation of funding which becomes available, advising on equipment for purchase and approving courses for teachers for which study leave may be available. It assists in curriculum development and offers support services to schools.

The Committee consists of twenty members. Five, including the chairman, are from the Board of Inspectors – Secondary Division. Three of the members, two from high schools and one from Central Regional Office, act in the capacity of State Consultants. The rest of the committee is made up of representatives from high schools plus a Science Faculty research officer.

In Tasmania, South Australia and Western Australia, full-time computer consultants are attached to the education departments. South Australia has set up a central computer centre, with special couriers transporting cards between schools and its batch processing system. It also has several microcomputers available for loan.

The Elizabeth Centre in Hobart has a group of specialists attached and provides a central computer system and advisory body for computer education. NSW, on the other hand seems to have little central control.

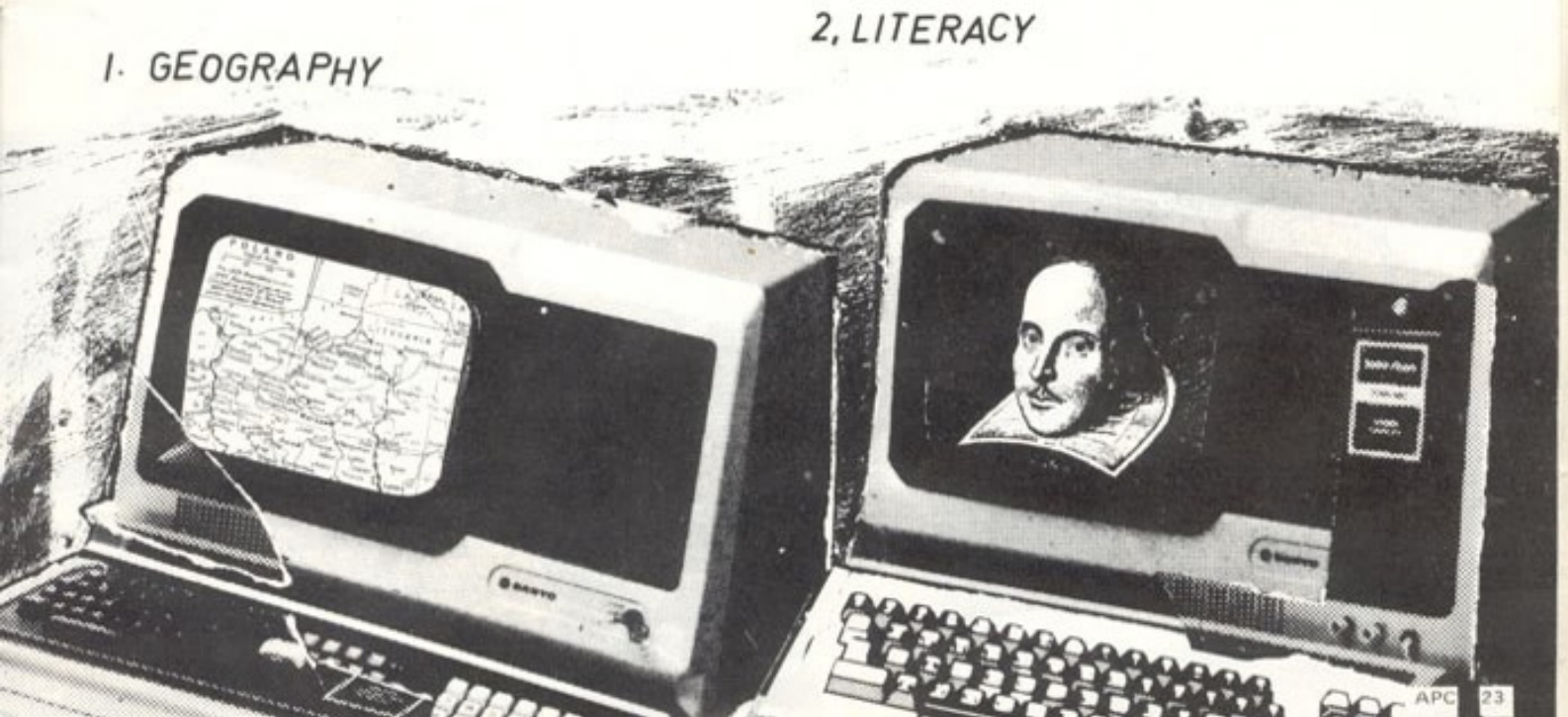
Mr. Graeme Inchley, Chairman of the SCEC, has said that he would like to see a central system in Victoria eventually, serving regional centres to which high schools can hook up. But for the moment these ideas will have to wait for policy development and funding.

The Secondary Division of the Education Department is recommending that schools set up their own systems. It has no policy on size, but the funding situation dictates that computer size be limited to what schools can afford with little assistance from the government. So schools are looking at total systems for under $5,000 – microprocessors.

This, Mr. Inchley points out, is not necessarily long term. Perhaps three to five years, until policy changes provide money for more powerful systems. Full in-house minis, allowing more than one terminal for a class of twenty, would be ideal – but logistics and funding dictate.

At the moment, limited funds are available to schools who tender a submission. However, central funds are not specifically allocated for computer equipment in schools.

The fact that schools have to raise their own finance will differentiate what schools in different socio-economic areas can have. Some schools cannot afford a computer at all; others have to decide on what may be more pressing claims on their budgets. Some private schools have had very sophisticated teaching systems working for some time.



Schools have a free choice on how to spend their money. A letter from the Director General's Computer Policy Committee advised on the configuration of hardware to buy, and suggested, that the Education Department be contacted for further advice. No products were endorsed.

A Departmental circular suggests that a "sound school's computer system" should have the configuration of a central processing unit, keyboard, visual display unit/s, printer, disc storage unit, card reader and adequate documentation. It should be usable by beginner teachers and students as well as the experienced; and have the following capabilities: use of BASIC with the capacity for alternative languages, legible display, text editing, a high resolution graphics display, a capacity for future expansion and interfacing with other equipment and for linking into external display units without undue extra expense.

The most popular system in use in schools at the moment is the 48K memory Apple II. At November last year, thirty-eight schools had opted for this system, with about twenty using the Tandy TRS80.

The Apple sells for around $1,600 and requires a colour television and a cassette as auxiliaries. The lower level TRS80 has a 4K or 16K memory, with a black and white video display monitor, and a cassette recorder which stores user programs, all at around $700. More suitably, the higher level, with a more powerful BASIC, costs about $1,200.

By the time a line printer is added for about $1,200 and a card reader for about $900, it is easy to see that schools will be straining their budgets and that government assistance could be speeded up. The ability to take VDUs is important to the system, so that as funds become available student access can be increased without disrupting the system.

In Victoria, it is not Education Department policy to endorse a particular product. However, in N.S.W., after a complex tendering process, a recommendation was made. In that State, the developed 32K Apple II and the small B & S Minimap were recommended.

Mr. Inchley has suggested that compatible systems be bought at different schools. This may overcome the real problem of teachers changing schools after having learned computing while setting up their own systems and having difficulties in adapting to a new one.

The familiarity of teachers with computers is something which has to be developed. Computer education is becoming part of teacher education courses and the State College at Melbourne is organising a post-graduate course in the area. But most teachers are already established in their fields so the SCEC is trying to sponsor in-service training schemes to familiarise those teachers who are already qualified.

In 1979, the Travelling Computer Roadshow, run by the Secondary Maths Committee as a schools-based in service, visited about fifty schools. It demonstrated microcomputers and informed teachers about the equipment, about software capabilities and the ways to establish facilities. It also gave some teachers first time hands on experience of computers.

This year, the SCEC will be taking over the Show. They hope the entire staff of the schools they visit will take part, and that it will widen the scope and understanding of all disciplines.

According to a circular explaining the SCEC, "although most schools were aware of the mathematical and administrative uses, there was little awareness of the wider and more valuable applications to commerce, art, the social sciences, etc."

In fact, according to Mr. Inchley, application is almost entirely restricted to the mathematics area. Lack of highly developed software and general conservatism have slowed the movement of schools' computers into administration or use as a teaching aid in other disciplines.

There is a real need for educational software. The SCEC is setting up an interim software library to make software of suitable quality available to schools. Currently, a lot of it is being written by teachers – some of it very good, some not so good.

Computer education is divided into two main areas. A computer awareness course is being introduced in the lower years. It is a general study of the history of computers, what they are, their uses, social implications and effect on the individual in society, with a bit of superficial programming thrown in.

At this level, the shortage of computer facilities could be solved by sharing among schools. This would have to be organized among the schools themselves, and so far the idea has not taken off. But moving the practical part of the computer awareness course from school to school would give students important hands on experience, whereas batching puts the interaction emphasis on programming. Where batching does have to be used, it is recommended that the turn around time does not exceed two days.

At the top end of the school, years 11 and 12 computer education go into the more technical areas of programming, logic gates, etc., as well as the social implications of computers and how they are applied in industry.

The SCES has released a syllabus for schools which gives guidance for a full year 11 Computer Science Course. It includes the history and description of computers, flow charting, introduction to BASIC, peripherals, DP applications, microprocessors and social implications.

A small proportion of students use the higher levels of computer education, so the main thrust of development has been directed at the awareness courses. This may change as the Victorian Institute of Secondary Education has accredited HSC Computer Science for assessment and certification as a Group 1 subject in 1981.

Group 1 subjects are examined externally and so can be accredited to university entrance assessment. There has been a hitch in the assessment of the practical part of the course; but once this has been ironed out, it should not be too long until the tertiary institutions and VUAC formally accept Computer Science counting towards entry.

There are no formal requirements for the course, although it is appreciated that students with a maths back-

3, NUMERACY

4, TRUANCY

Illustration by George Snow

ground would handle the logic and symbol manipulation of algorithmic problem solving and programming more easily. It is intended that the subject complement any combination of subjects and is not designed strictly as a part of a maths/science course.

The course has three main objectives. Students are expected to complete the course knowing broadly how computers work. They will be taught to devise, test, code, document and validate simple algorithms for problem solving. They should also end up understanding the information manipulation potential of computers.

To achieve this, the course will aim at providing students with an insight into logical processes and problem solving with algorithms. It will develop the ability to recognize these problems and the skill to devise algorithms; and will familiarise students with the concepts of data structures and file processing, and methods of using hardware and software components. A programming language will be taught and the translation of algorithms into that language.

Formally, the syllabus contains a core of twenty weeks and optional units over ten weeks. The core contains six sections titled: Computer Structure, Algorithms, Programming Language, Data Structure Manipulation, Input/Output Devices and File Manipulation. Students can choose two out of the four optional units of Social Implications of Computers, Hardware, System-Case Studies and Visits to Installations. This will be contained in four periods of forty-five minutes each, per week.

35% of the marks will go to practical core work, and 35% to an examination, both of which will be externally assessed. The practical work must be passed for a pass in the subject. The rest of the marks are taken up by internally assessed optional unit material.

The universities are questioning whether the practical work will be adequately externally assessed. This is the hold up with VUAC acceptance of the subject.

The VISE recommendation on the matter is for a moderating panel, consisting of teachers from regions designated by VISE as a convenor, to assess the students' work. While the class teacher is present at the panel meetings to answer queries on the students' work, he or she does not take part in the marking directly.

The problems seem easily surmounted and no one expects the universities not to accept the subject.

Once this happens, the ramifications could really make the prospects for computer education in schools look up. More emphasis will be placed on the junior awareness courses. With the increased status of HSC Computer Science, and depending on a good showing in enrolments, government funding will have to come through. And with increased government spending, computer professionals will increasingly turn their eyes to the market for good educational software, which will probably spin off into teaching aid development for other disciplines.

Victorian education will be moving into the computer age.

SYSTEMS

APC's Benchtest and In Store provide a comprehensive guide to hardware procurable in Australia. To provide a balanced coverage, Systems will take a different business application each issue and report on some of the software packages available around it. Dave Tebbutt and Mike Knight take up the explanation.

Perhaps before looking at the fine detail of our approach for the future we should examine the reasons for introducing Systems.

You've probably seen or heard business packages described in glowing terms. They are said to be complete, comprehensive or total. Sometimes they are not described in any terms at all; sometimes they are described in terms which only the writer understands. Somehow the prospective buyer must decide from this morass of inadequate information, which packages to consider buying.

Nor do the problems end there. Having selected a few possible packages, the potential buyer needs to know quite a lot more before making any final decision.

Is it well documented, for example? We can barely believe some of the apologies produced in the name of documentation. It can be inadequate in a number of ways. First of all it may simply not exist. . .not even instructions for operating the machine! Secondly, yes, it may exist, but in such a form as to be totally unintelligible to mere mortals — not to mention the prospective buyer/user. Thirdly, it may exist, but only in parts. The missing sections are usually the ones you need when you're burning the midnight oil and all the 'experts' are fast asleep in bed.

An exaggeration? In many cases we think not, although we have to point out that some companies do produce quite excellent documentation.

And here's something else to think about — bugs. What are bugs? ...well, in common parlance, they are errors existing in the application package which cause it to go wrong from time to time. Of course, ideally, one would like any problems resolved on the spot — time, after all, can be expensive. Here the difficulty may be that the firm from which you bought the package no longer exists. Perhaps (more likely) they aren't too interested, or don't have the staff to tackle any bugs. Again we don't want to paint an unduly miserable and pessimistic picture, but these are very serious matters and they need to be considered before any money is exchanged for software. For the businessman it could mean his business crashing down alongside the programs.

Okay, enough of the horror stories, time now to take a look at some constructive action.

Each month when we report on a particular application area, the feature will be divided into the following sections:
Objectives
Tasks and volumes
Evaluations
Comparisons
New products
Let's look at each of these in turn.

Objectives

In this section we shall define the objectives of the application. We shall also describe the application and explain any relationship with other applications. Failure to be very clear about objectives will lead any investigation to likely failure.

Taking 'payroll' as an example, we might describe the overall objective as 'to pay employees the amount due *on time* and to meet statutory requirements'. Then we might describe the application as follows:
1 Capturing information upon which payment will be based.
2 Using this information to calculate net payment.
3 Maintaining records of payments to each employee.
4 Producing appropriate documentation for company, employee and government records.

Finally, we might define the relationship with other applications as: 'information gathering — possibly the product of production hours recording. The payroll application will almost certainly create "transactions" for the accounting function'.

Tasks and volumes

In this section we shall select, say, three packages and match them against the tasks to be performed. Staying with our payroll example, we might say something like this:

"Not only will this give a guide to three particular packages, it will also offer a framework against

Tasks:	Package	A	B	C
Create employee records		●	●	●
Delete (suspend) leavers			●	
Maintain existing records		●	●	●
Build up to gross		●		
Gross to nett			●	●
Print payslips		●	●	●
payroll		●	●	●
cash analysis			●	
cheques or credit transfers		●		
bank reconciliation		●		
Update employee records		●	●	●
Produce accounting transactions		●		
etc.				
Maximum Volume/sizes:				
Employee records		400	250	600
Record size		180ch	360ch	200ch

which to measure other packages of your choosing."

Evaluations

In this section we shall again focus attention on the selected packages. This part of the feature will be written as a structured narrative, describing each package in turn. The main elements are as follows.

Availability
Documentation
System content
System maintenance
Costs
Hardware required
Support and training
User comments

Availability covers number of suppliers, their distribution and whether the product is available 'off the shelf'.

Documentation describes the scope, content and quality of the manuals and guides supplied.

System content will describe the programs involved in the package, their functions and certain aspects of their operation. For example, it may be that each program, on conclusion, automatically loads the next in sequence. On the other hand, there may be a need for a lot of disc or tape changing during the operation. We will try to give a picture of what will be involved in the day to day running of the system.

System maintenance. We shall be interested in whether the system has been designed to be changed easily. Examples which spring to mind are tax rates and discount terms. We shall also see if customisation is easy. Some packages are written with 'hooks' to enable customised routines to be added fairly simply. The language used is also important here. Finally, we shall check out who you have to go to to have these changes made.

Costs need little explanation. We shall give the costs for various versions of the package and, if applicable, the cost of any maintenance agreements.

Hardware required. We shall describe the different hardware configurations and relate these to the volumes which can be handled by each. We shall also give a guide to the hardware costs.

Support and training. If either of these areas are neglected, it's likely that you'll end up very disappointed with your new system. Training should, at the very least, teach you how to operate the system. Support is the on-going advice and guidance you will get from the supplier. It also covers their response to any problem you may encounter — a hardware fault, a software fault or perhaps an accident such as over-writing some important files. We shall assess the services offered.

User comments. We shall contact users of each system and summarise their opinions and experience of the package.

Comparisons

This section comprises a straightforward comparison chart showing all the packages notified to APC, for the application in question. Each will be evaluated against the criteria discussed in this article. Because we cannot do an in-depth analysis of every package, this information will be based on that made available by the suppliers. If the publicity documentation fails to mention something, we shall not make assumptions and the column shall be marked N/A — not applicable.

New products

Finally, and quite separately to the above, we shall provide information on any packages newly notified, for application areas already covered.

We're sure that this structured approach to package evaluation will help readers in the selection of their business software. There are a lot of good and reliable suppliers of these products in the field, all of whom will give sound advice. But this series of articles, as much as anything, should help clarify your own thoughts on what can be a rather tricky subject.

If you would like to tell the world about your system, be it a standard package or custom-built, then please get in touch — other people's successes (and failures) may offer invaluable information to businessmen working in similar areas.

BUSINESS COMPUTING

For the first time the progress of technology makes it possible to enjoy the benefits of a computer in a small business environment for about five thousand dollars.
Or is that really true?
The answer is "yes, but . . ."
The purpose of this article by Rodnay Zaks is to justify the "yes" and to describe the "but".

The essential deficiency of microcomputer systems is not at the hardware level but at the software level. This has always been the case ever since computers were introduced, and history has consistently repeated itself every time a new generation of hardware was introduced. The necessary hardware to process efficiently a number of business applications can indeed be purchased for $5,000 to $25,000. However, software is only slowly becoming available. Naturally many trade-offs exist in function of the capabilities one wishes to acquire, and these will be studied.

Therefore, the classical applications of computers in business will first be reviewed, in order to define the processing capabilities required to achieve specific goals. In order for the businessman to make a reasonable choice of a computer system, it is imperative that he understands the trade-offs between the various solutions available today as there is no "best". The choice can be somewhat compared to the selection of a new car or of a new complex machine in function of a specific intended application. There is no general-purpose choice fit for all applications.

Understanding the hardware required and the hardware available is a relatively simple matter. The more complex and difficult problem is understanding the software capabilities required. This is where a large majority of persons purchasing a business system make mistakes. These mistakes are generally more costly than hardware ones. Typically software investment in a system will quickly become the dominant one.

An inadequate system will limit the possible growth of the capabilities of the system, and possibly of the business. A transition to a different system might be costly and disruptive. For these reasons, the reader is strongly encouraged to study and understand the software concepts as well as the hardware ones that will be presented.

Applications of Computers in Business

Every business needs primarily to maintain a number of files. The best known files are: accounts receivable, accounts payable, inventory, general ledger. Additional files which are usually desirable are: personnel, customers list, mailing list, back-orders lists, sales list, vendors list, cash situation, company property, and more.

These lists are managed either by hand (typically by a bookkeeper), or with the help of electro-mechanical devices, or by computer, or by a combination of the above.

In addition to maintaining files, every business applies specific processing techniques to each of them. For example, a payroll program will process the personnel file and generate payroll reports, as well as print cheques. A tax program will process the sales reports and the personnel files to produce the required tax reports. A transaction procedure program will manage updates of specific files, and changes, or entry, of new data. A typical example is a new sale: the transaction program will utilize the inventory file, supplier file, customer file, and perhaps others. It will update them, and print reports.

Similarly an incoming shipment procedure will handle shipments coming in and will enter them in the inventory file, check for back orders, and add entries to the accounts payable list.

Any payment received will update the accounts receivable list and the cash situation list.

In addition to the main programs a number of additional programs must be available in order to produce useful reports. These additional facilities required will be described in more detail in the text following.

It is important to note that the principle is quite simple:

1- Files must be created and maintained.

2- Programs should be available to provide the interface between the user and the files, and supply the required processing functions.

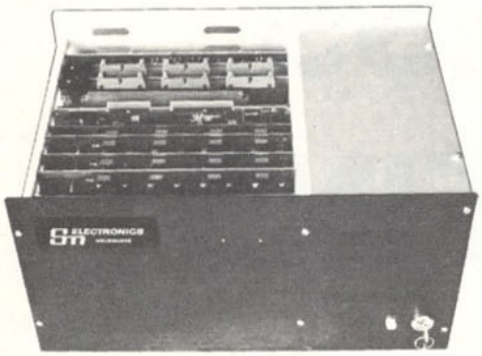
Unfortunately in a real business system, this is only part of the processing required. In fact, in most businesses, the direct maintenance of a single file is reasonably simple. The bulk of the processing required is due to the simultaneous cross-referencing and automatic updating of multiple files.

Let us look at an example. An order is received by mail. It will be processed by the transaction manager program. The sale will be entered in the sales file for the day. A complex sequence of events now unfolds. As a result of this entry, the name of the customer will be added to the customers list automatically. In addition his name will probably be coded in function of the purchase he has made or of the amount of the purchase, or of his job position. In addition, his name will be checked for credit information before the order is processed. Provided the sale is not "vetoed" by the credit manager program, the next step is to honour the order. The saleable inventory file will now be checked for the availability of the items ordered. In this example three items are ordered: A, B and C. A and B are in stock. C is not.

As a result, an invoice to the customer is generated, a shipping list and a back order are generated. The back order is added to the back order list. In our example, item B is available in stock. The inventory list is structured with a special field which specified the re-order level. The re-order level of item B is four. As a further result of this transaction, a back-order or re-order will also be generated for item B for a standard quantity of 25 items (the number 25 was specified in the inventory file). The address of the

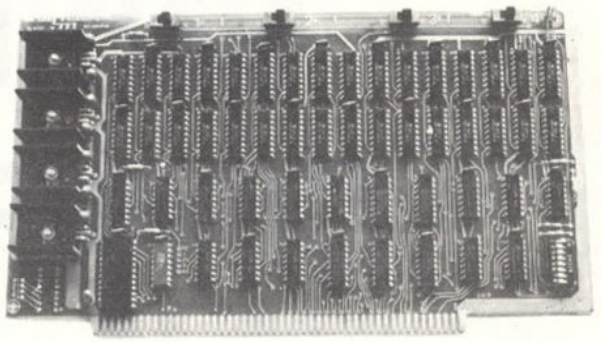
'THE S100 BUS STOP'™

S-100 & 6800 CHASSIS



11 slot backplane, fully card guided. 15 amp power supply, fan, key switch, bench mount, rack mount, annodised aluminium. 5 edge connectors standard. S-100 Bench Kit **$345.** S-100 Rack Kit **$306.** 6800 Bench Kit **$370.** 6800 Rack Kit **$330.** Assembled prices add **$100.**

ETI 642 S-100 16K STATIC RAM



Features:- 2114 low power static RAM's, 4K addressing, 4K write protect, bank select, wait state gen., plated through hole, solder resist mask, 300 or 450 nS speed, ETI 642.
Kit **$315.** Ass. **$380.** Add **$32** for 300 nS.

2708/2716 EPROM CARD
Features:- holds up to 16 2708 or 2716 (single supply) EPROMS, on board wait state gen. Unused locations may be blanked. Plated through holes, solder resist mask.
PRICE:- Kit $115. Ass $155.

EPROM PROGRAMMING CARD
Features:- ability to programme triple supply 2708's and single supply 2508, 2716, 2732 etc. Zif. Socket. On board 26V generator. Port driven.
Price: Kit $205 Ass. $255.

Z-80 CPU CARD
Features:- 4 MHz operation, power on jump, wait state generators, provision for on board 1K EPROM, front panel socket for reset, and data lines etc.
Price:- Kit $156. Ass $196.

Z-80 SINGLE BOARD COMPUTER
Features:- 4 MHz operation, 1K static RAM, 8K/16K EPROM, serial/parallel ports, power on jump, timer, vectored interrupts, software selectable buad rates. With 2716 EPROM. Price: Kit $360 Ass. $440

80 X 24 VIDEO DISPLAY CARD.
Features:- on board Z-80 and CRT 5027 controller chips, parallel keyboard interface, 2708 driver chip, and 2708 character generator chip, special effects and extended character set available.
Price: Kit $380 Ass. $450

64 X 16 VIDEO DISPLAY CARD
Features:- memory mapped 1K board, with reverse video and cursor control. RCA video connector, plated through holes and solder resist mask.
PRICE:- Kit $155. Ass. $180.

FLOPPY DISK CONTROLLER CARD
Features:- single density, mini or full size disk drives with FD 1771 controller chip, can be interrupt driven, syncs with CPU in data transfer, Shugart/Remex compatabile.
Price: Kit $258 Ass. $308

DD FLOPPY DISK CONTROLLER CARD
Features:- controls mini and full size, single/double sided single/double density and all combinations of each. Crystal locked, PLL data recovery, Shugart/Remex compatible software (CP/M / SDOS) for above controllers available.
Price: Kit $360 Ass. $420

STANDARD EXTENDER CARD
Features:- double sided f/glass board, numbered test points reflow soldered.
Price:- Kit $33. Ass. $48

WIRE WRAP CARD — PLATED THRU
Features:- double side f/glass board, ground plane and supply rails run on both sides, 3M type connector patterns on top of board, provisions for regulators on all rails, holes are on .3" pitch, by .1" pitch.
Price: Bare board $38.50

6800 PRODUCTS
6800 Extender Board **$33.** 6800 11-slot backplane **$36.** 6800 11-slot chassis, rack mount **$330.** 6800 Extender Terminator Board, **Kit $80. Ass $105.**

EPROMS AND RAM CHIPS
2708 450nS guaranteed **$12.** 2716 450nS single supply ex-stock **$47.50.** Hitachi 2114 low power 450nS **$7.50.** Hitachi 2114 low power 300nS **$8.50.**

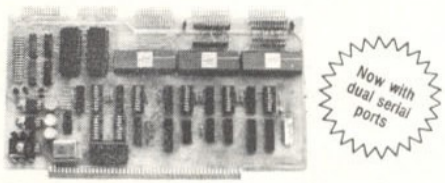
DISK DRIVES
Shugart SA400 **$410.** Shugart SA801 **$710.** Remex 8-inch double sided **$795.**
Shugart SA850 —$1040

DUAL 8" DRIVE PACKAGE
Features:- contains dual 8" single or double sided disk drives either Remex or Shugart. Inbuilt power supply, cooling fan, modular construction, keyswitch, fused on mains, all aluminium 19" rack mount (10-½" high).
Price:- single sided $1750. Double sided $1950.

EPROM SOFTWARE
1. Z-80 monitor in 2708 EPROM, has 16 functions, three versions available to drive TTY, TTY/VDU, KBD/VDU. **Price $25. 2.** ETI 640 video driver EPROM, makes the memory mapped video card look like a terminal, has XY cursor addressing, home clear screen. **Price $25.**
3. 6.25K Basic interpreter, in seven 2708 EPROMS, has trig functions, dimensions, command level input ability. EPROM resident at OC000 hex. **Price $180.**
4. Disk control EPROMS, contain I/O routines to handle our disk controller with CP/M, 2 EPROM set with second EPROM containing inbuilt video driver and I/O routines for all external devices like printers, terminals etc. **Price $50.**
Customised version available (I/O and relocation) for an additional charge depending on the programme.

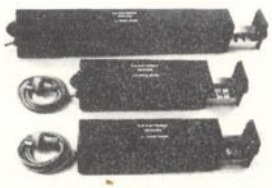
DISK SOFTWARE
CP/M version 1.4, customised for our controller $145 CBASICII **$100.** Wordmaster, word processing package **$140.** TEX writer, letter and text formatter **$50.** CP/M user group library (33 vols) at **$12** per vol . RAM Diagnostic, reports errors and likely causes **$25.** Available on 8" and 5-¼" single or double density. Above prices are for 8" disks.

S-100 I/O PORT BOARD



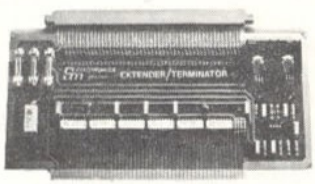
DUAL SERIAL I/O CARD Features:- dual independantly controlled serial ports with TTY and RS232 outputs and inputs. Nine programmable parallel ports, crystal controlled baud rates fully buffered and address decoded. Plated through holes & solder resist mask.
Price:- Kit **$189.** Ass. **$225.**

UV EPROM ERASER



New product range. Model LEE/T 15W tube, 120 min timer, up to 40 EPROMS will erase in 10/15 mins. Model MEE/T 8W tube 120 min timer, up to 10 EPROMS will erase in 20/30 mins. Model MEE is same as MEE/T but with no timer. All erasers are fully assembled and have a safety switch. LEE/T **$105.** MEE/T **$93.50.** MEE **$74.**

S-100 EXTENDER/TERMINATOR



EXTENDER TERMINATOR CARD — features:- true active termination of the bus with inbuilt extender connector on top of board, fused rails to extended board. Test points numbered, solder resist, plated through. Price:- Kit **$70.** Ass. **$90.**

SM ELECTRONICS
MELBOURNE
Ph (03) 842-3666
Trading Hours: 10am-6pm Mon to Fri.
1096 Doncaster Rd, Doncaster East, Vic 3109.
PO Box 19, Doncaster East, 3109. Telex AA37213.
DEALER: Canberra – 81-5011, Sydney – 661-9237.

Send 60c in stamps for COMPUTER PRINTOUT CATALOGUE for more details.

ALL PRODUCTS AUSTRALIAN MADE AND EX STOCK (ALMOST).
DEALER ENQUIRIES WELCOME.
Prices and specs. subject to change without notice.

All prices tax free, for retail prices add 15 percent.

bankcard welcome here
Give name, number, expiry date and signature for mail order sales.

vendor is obtained from the vendors file by using the vendor number as an index to the list.

This simple sales transaction has required the use of five files within our system and of several processing programs. For specific business, it might even be necessary to update, check, or modify additional files, or perform additional processing functions. It should be clear from this example that, in order to be truly useful, a business system must provide ways to access, modify and process conveniently a variety of files. In addition it must provide a mechanism for performing all the required functions automatically, not manually.

Unfortunately, it will be seen that the majority of so-called business systems available today, using microcomputers, do not perform such a complete service. They provide usually single file management and do not automate completely the complete transaction process. Much has to be done "by hand".

Word Processing

"Word processing" refers to computerized typewriter operation, where the user can easily change, modify, or format text. It requires an "editor" program, a standard facility of traditional computers. The cost of the processor has become so small that it can be dedicated to a function such as word processing so that "stand-alone" word processors are multiplying. The majority use a Selectric or similar typewriter. By contrast, business systems offer the option of displays or multi-terminals.

Using a Business Computerized System

Let us use now an in-house microcomputer for a simple transaction. We will specify the type of program, and our choices in response to choices or questions appearing on the screen of the CRT terminal.

Initially, the system displays a "menu". A "menu" is simply a multiple-choice question. The question asked by the system is stressed by one or more "prompt characters" (here, ". ."), designed to indicate that the microcomputer is waiting for an answer.

Fig. 1 A "MENU"

The "business program" has been selected. The system should load it automatically from the disc. A directory of options appears again.

Fig. 2 THE BUSINESS "SUBMENU"

We specify the "accounts receivable". At this point, the system may request that a new diskette be inserted. Let us assume not, and proceed.

Fig. 3
THE ACCOUNTS RECEIVABLE FILE

We specify a new sale, and the system will request all data needed to record the transaction, generate an invoice, and later update all related files such as bank, accounts receivable, inventory, customer list. The dialogue becomes now highly interactive with the system requesting all necessary data.

Fig. 4 ENTERING A NEW SALE

Fig. 5 SALE ENTRY, CONTINUED

Fig. 6 SALE ENTRY, END

The transaction is now completed.

The mode of interaction with the system should now be clear. The program asks all necessary questions, enforcing a discipline. In addition, we will see that it should also check the validity of data being entered (no gross errors). Finally it should automatically print invoices, and later update all related files.

Let us now examine in more detail the actual requirements.

Business System Requirements

The requirements of a business system will be analysed here in terms of the essential files that must be maintained and of the essential processing functions which must be performed.

Accounts Receivable:

This is essentially the file which contains a copy of all invoices generated by the system. Naturally the file does not contain the actual copy, but the minimum amount of information that it is possible to store, which allows the system to actually generate a complete invoice. Typically, it will store the date of the transaction, the name and address of the customer, shipment point, sales information such as salesman, how shipped, when shipped, and specific details of the items sold. It may not be necessary to store all the information which appears in a usual invoice within this accounts receivable file. If a sales file exists, all this information is stored there, to be accessed frequently and processed efficiently.

Efficient information processing by any computer requires that all elements within a file be of equal length. For this reason, all files which are processed often, or by complex programs, use fixed length entries or "blocks". An accounts receivable file can be structured in that manner. Fixed fields can be allocated to essential information such as date, name, amount due, transaction or customer code, invoice number. The presence of the invoice number allows the user of the system to access the remainder of this information in the sales list or in the invoice file. In computer jargon, the presence of a number used to access information stored elsewhere is called a pointer. The invoice number is a pointer to the actual invoice. In business jargon, this is part of the audit trial.

The accounts receivable file must be distinguished from the accounts receivable program. The accounts receivable file is simply the list of accounts. The advantages or disadvantages of its format are easy to evaluate by the business user. A typical requirement is that it contain, in an easily accessible way, all the fields that the business user requires frequently.

The accounts receivable program is responsible for manipulating this file, updating it, and generating the required report. It must also generate specialised reports such as the printing of accounts older than 30, 45, 60 or 90 days (this is called "aging"). This program can be even responsible for generating automatically reminder notices. However, the remainder notification program may be a separate program. In this case the accounts receivable program would be used for generating a file of overdue accounts. This file would then be used in turn by the reminder notice program in order to generate personalised reminders to all customers listed in the overdue file. Whether to separate functions into individual programs or integrate them within a single program has little impact on the value of this sytem. It is largely a matter of programming convenience for the system designer. The important point is that all the facilities be available.

Accounts Payable

The accounts payable file is essentially a list of all bills or invoices received by the business. Typically, whenever an "OK to pay" order has been entered, the accounts payable manager program will automatically print payment cheques for the goods received. Typically, the cheque will be printed either at a specified date, or else at a programmed date such as thirty days after receipt of invoice. (A good cheque printing program should also check that the cash balance in the bank account is sufficient to cover the expenditures!)

Inventory

There is no optimal inventory file, as inventory information is different depending on specific business needs. For this reason, most general purpose inventories files will carry a large number of categories. Not all categories will be used by the business. The unavailability of some categories can be felt to be a drawback by some users. The availability of too many categories on the other hand, means that a significant amount of space is wasted in the system. This translates into a relatively smaller number of items that may be entered in the inventory. However, with the ever decreasing costs of memory, the clear trade-off now is to provide as many categories as possible, for most types of businesses, even if some of them are never going to be used. It should be remembered that the size of the inventory file is limited by the physical storage available, such as the size of diskette.

Typical information which may be included in an inventory file is the following:

CODE – ITEM NO. – ITEM DESCRIPTION – STORAGE LOCATION – NUMBER AVAILABLE – VENDOR NUMBER – FILLING PRICE – PURCHASE PRICE – LAST SALE DATE – MINIMUM QUANTITY FOR RE-ORDER.

64 to 128 bytes at a minimum must be provided for such an entry. Using such a format, 1800 to 3600 items may be stored in a diskette.

The inventory control program must provide many functions. It must provide generalised inventory management facilities:

- complete inventory maintenance, including automatic updates of any category of information within the file
- sales order entry
- purchase order entry
- sales history
- automated backorders
- list of quantity, class, cost, vendor, item no., date of sale
- minimum quantity search
- selective update
- activity reports
- inventory lists in functions of combinations of criteria.

As a rough indication, a minimal inventory management, written in BASIC will require 10K words of memory (for all practical purposes a "word" is a "byte" here, in the case of 8-bit microprocessors). A more general program will easily require 90K or more. Since the central memory of a microprocessor is never larger than 64K, an overlay technique is used, so that such large BASIC programs can be run on a smaller main memory. An overlay consists in executing one part of the program, then bringing in the memory an additional part of the program and overwriting a no-longer-required segment of the previous one which had been installed in the main memory, and so on. The complete BASIC program is therefore never resident in the memory in one piece. Pieces of it are brought into the central memory as needed. Naturally this reduces the efficiency of the processing. However, if the overlays are cleverly written, the impact on efficiency is reasonably small.

Update

It is important to note once more, that, technically, update on an inventory file can all be performed by hand. The user can examine the list of items in the inventory and modify any of the entries such as the unit cost. However the real value of the computer system is again in automating the updating of identical information in many files. Therefore a comprehensive business system should automatically update the inventory file, whenever relevant information is changed somewhere else. For example, should the unit cost of the product be changed, it should be updated automatically in the inventory file as well as in any other file where it might reside.

To be concluded in the next issue.

BEGIN . . .

A Simple Outline of Computing

. . . END

Brian Darling

With the continuing downward trend in the cost of computer hardware, many people are now taking their first serious look at computers. But confronted with RAM and ROM, BYTE and BASIC they simply do not know where to start. There is a danger that some will be so intimidated by the difficulties before them that they will give up, which would be a pity. This article sets out to show the complete beginner the outline of computing, leaving him to fill in the details himself. Using a conceptual model, the operation of the computer is explained and a program is devised to read in, add together and print the sum of, two numbers.

The Computer

A computer consists of five main units. Three of these: store, arithmetic unit and control unit, together form the central processing unit (C.P.U.). The fourth, is an input device – which for our purpose could most conveniently be a keyboard, similar to that on an electric typewriter. The last is an output device – which could be a printer, but on a personal computer is more likely to be a television screen.

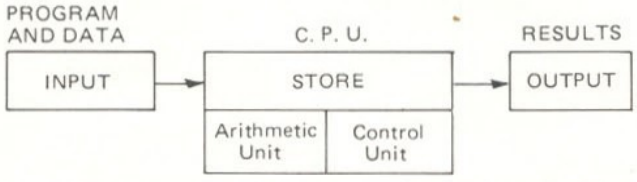


Figure 1. Notice that input goes into store and that output is taken from store.

Figure 1 shows the five units.

The function of the input and output devices should be self-evident. The arithmetic unit, as its name suggests, performs the arithmetic and is quite similar to an electronic calculator. Store – which can be imagined as a set of numbered pigeon-holes – is used to store both the data and the program. The fifth unit, control, controls the overall operation of the computer and ensures that the program is executed one step at a time.

Rather surprisingly, this incredibly simple model will allow you to understand how the most complex programs are run.

The Program

A computer program is a list of instructions which cause a computer to carry out some task. Possibly to guide a space craft to the moon, but more likely to perform some rather mundane clerical job.

As an illustration of what programming involves we will show you how to write a program to add two numbers together. We will assume for the moment that the two numbers are already in store – which, you may remember, we visualize as a set of pigeon-holes. In order to add the two numbers, they must be transferred to the arithmetic unit.

The program consists of just three instructions. The first transfers one of the numbers from store to the arithmetic unit. The second transfers the other number to the arithmetic unit, adding it to the first. The last instruction transfers the sum of the numbers back to store. This can be rather confusing at first but it should become clearer when we describe how the program is executed.

The three line program is shown below:

```
LOAD 103
ADD 127
STORE 107
```

The three numbers refer to three locations in store. 103 and 127 hold the numbers that are to be added and 107 will be used to store the result. The locations used are quite arbitrary and any others could be used, providing we first load into them the numbers we wish to add.

Be sure that you understand that it is the contents of store locations 103 and 127 that are added, not the numbers 103 and 127 themselves.

Execution of the Program

Before we can run the program we must enter it, together with the data, into store. This can be done by typing it on the keyboard. We will put the three lines into locations 001, 002 and 003. The program adds together the contents of storage locations 103 and 127, so we must enter into these locations the numbers, or data, that we wish to add. Say, 5 into 103 and 9 into 127.

Execution of the program is controlled by the control unit which has to fetch each instruction from store and execute it. This is known as the fetch/execute cycle. Part of the control unit is a counter which ensures that instructions are executed in the correct sequence. This unit is called the sequence control register (S.C.R.) or program counter.

The first instruction is fetched from location 001 and executed. It is:

LOAD 103

so the contents of 103 (5) are loaded into the arithmetic unit.

The second instruction is fetched from location 002 and executed. It is:

ADD 127

so the contents of location 127 (9) are added to the contents of the arithmetic unit (5) making the new contents of the arithmetic unit 14.

The third instruction is fetched from location 003 and executed. It is:

STORE 107

which transfers the contents of the arithmetic unit to store location 107.

Although this program is very simple it does illustrate how data is held in store and how it can be transferred to the arithmetic unit to be added, multiplied etc.

Machine Language

The above program is written in assembly language and a special program, called an assembler, is used to translate it into a form that the computer can recognize, called machine language. For engineering reasons computers use binary arithmetic, which is a kind of arithmetic employing just two symbols: 0 and 1. Machine language instructions consist of groups of eight binary digits, for example, the ADD instruction for one computer we know of is 00000010.

A binary digit is called a bit and eight bits together form a byte. Personal computers generally use a word length of one byte.

Machine language can be quite confusing to the beginner. But don't let this put you off, as your first attempts at programming will almost certainly be in a language called BASIC which we will describe shortly and you should find this much easier. Actually, machine language is not so difficult as it looks and you may get to quite like it later.

High Level Language

You may have been surprised at the amount of work involved in adding a couple of numbers together and, in fact, there is an easier way. High level languages allow instructions to be written in a form quite close to English. The three line program used above can be condensed to a single line in BASIC – which is the most widely used language for personal computers.

The BASIC instruction is shown below:

LET C = A + B

The biggest advantage of using a high level language is that it is no longer necessary to keep track of the store locations used. A program called an interpreter translates each BASIC instruction into machine language and also allocates storage locations to each of the letters A, B and C – called variables – which hold the numbers to be added.

Some computers utilize a compiler instead of an interpreter. An interpreter translates BASIC into machine

language during program execution whereas a compiler must translate the entire BASIC program *prior* to execution.

We did not show the input or the print instructions in the assembly language program, but we show them below using BASIC.

```
INPUT A, B
LET C = A + B
PRINT C
```

When the three line BASIC program has been entered through the keyboard it can be executed by typing the command RUN. (This may be slightly different on some small computers). The computer will print a question mark – or display it on the screen – and the user enters a value for A, it prints a second question mark and a value is entered for B. Almost instantly, the value of C will be printed.

It should be clear by now that BASIC makes programming very much easier.

Micro Programming

It is possible on some computers to go down to an even lower level than machine language, called micro programming. Although we described the C.P.U. as being made up of three units, it is more accurately described as three groups of units. Micro programming can be used to open and close doors –metaphorically speaking– and cause a series of 1s and 0s to pass between the various units to achieve the operations required. As you might imagine, it is a fairly complicated process and is not much used. But for some purposes it does result in extremely efficient programs.

Backing Store

Even on the largest computers the main store is rarely large enough to hold all the data that the user needs to store. For this reason, magnetic tapes and magnetic discs – called backing store – are used to provide extra storage capacity.

Most people will be familiar with computer tape units, as whenever a computer is shown in a T.V. play, the tape units are most prominent.

Data is stored serially on magnetic tape, which means that if data has to be read from several locations, the tape will have to be continually rewound. Consequently, the time taken to locate and read a particular piece of data – the access time – is quite long.

Magnetic discs store information randomly; that is, any piece of data can be accessed immediately, in contrast to tape which often has to be wound through most of its length to locate some item. The disc spins at high speed and the read/write head can be moved from the edge to the centre, to locate a particular piece of data, in much the same way that you can choose to play a particular track of an L.P. gramophone record. As a result, the access time for discs is much faster than for tape. But both are much slower than the computers main store.

ROM & RAM

The letters RAM stand for random access memory. Random, is not used in its usual sense, but rather, it means that any location can be accessed as required. The computer main store is constructed from RAM and each location is identified by a number, referred to as its address.

ROM stands for read only memory. The interpreter of a personal computer is held in ROM. When a computer is switched off the main store is emptied but ROM retains data even without the power on, so the interpreter will still be there next time you want to use it.

If you have a lively mind, this article will have raised many more questions than it has answered. But by now you should have a pretty good idea of the framework of computing and thus find it considerably easier to fill in the details.

Finally, do not allow your present lack of knowledge to deter you from pursuing computing, either as a hobby or for business purposes. You will find that learning about the subject is easier and more interesting than you ever imagined.

Feedback

INTERRUPT

Interrupt is the place in APC where readers can unburden their grievances and air controversial views. New subjects are always welcome; the 'right to reply' shall be wielded at the discretion of the Editor.

A chance for the disabled

It is rare for a day to go past without one coming across some item of headline news concerned with people being seriously injured. When injuries on the roads are counted along with the many disabling conditions that exist — like multiple sclerosis — it's not difficult to see that a large number of people, including young children, suffer by being severely physically handicapped.

For many, living and working relatively normally can be achieved by their using the aids available (coupled with a stubborn personality) but for many others life is currently very empty and frustrating. However, the microcomputer, that awful technology which is to cause mass unemployment and other problems, will revolutionise the world of aids for the disabled and provide a wealth of new opportunities associated with severe disability.

It is in the area of being able to solve many problems with one box that the microcomputer becomes a most valuable aid.

Disabled people in specially adapted cars or with guide dogs are a familiar sight. The mobility problem is easy to see, but what of other, less obvious difficulties? Most people take the ability to scribble on paper or read for granted. However, an inability to manipulate papers or turn pages makes such simple everyday tasks next to impossible. No scribbling facilities are available for many disabled people who have to rely on good memories and adapted typewriting equipment for note taking and writing. It is surprising that, as yet, no cheap and reliable page turner has been developed to assist with reading. The problems of the extremely disabled are highlighted when watching a severely spastic child who probably cannot communicate, cannot manipulate toys, cannot play games and cannot explore the surrounding world, although input to the brain is unimpaired. Life for such children is very frustrating and for their parents, both heartbreaking and trying. The list of problems for the severely disabled is endless and so is the list of expensive attempted solutions.

A wide range of aids are presently available ranging from fat handled spoons for those with poor grip to complex print reading machines for the blind. A severely disabled person very soon accumulates a house full of gadgetry, each item being very useful to assist with one particular problem. It is in the area of being able to solve many problems with one box that the microcomputer becomes a most valuable aid. However, giving all disabled people a boxed microcomputer and a terminal would not solve the problems and certainly would not be a popular policy for a somewhat 'thrifty' government.

There are basically two approaches to using microcomputer technology for aiding the disabled. Firstly, available microcomputer based equipment can be adapted to suit a particular disability and secondly, purpose built systems can be produced. Both approaches to building computer based aids have different applications and market. Neither approach produces inexpensive aids which means that, at present, it is often difficult for an unemployed disabled person to improve his/her quality of life.

Adapting existing computer based equipment for use by a severely disabled person is a skilled job requiring understanding of the problems associated with a particular disability. Many users, given the appropriate hardware, are only too willing to produce their own software. Usually, the biggest problem is to find a suitable terminal device which can be operated by people with very little movement. Special switching systems or breath tubes may also have to be interfaced and the necessary software written. There are currently disabled people successfully running small businesses from home using their personal computer for letter writing, accounting, filing etc. This, of course, is not an unusual application for a boxed microcomputer system — the difference, in this case, is that it enables someone to work who might be otherwise unemployed and probably very bored.

Microcomputer based, purpose-built aids for the disabled are also appearing increasingly. These aids are usually designed with a particular set of problems in mind. For example, machines that will scan printed text and give a spoken word read-out are currently being produced by two companies in America, a significant advance for those who cannot read braille or blind people who need to read material that has not been transcribed into braille. A portable braille word processor has also been developed, called the VersaBraille. Development of aids is not limited to America, work is also progressing rapidly all around the world. In Britain, Medelec have just launched a communicator for stroke victims — SPLINK, and Ferranti are manufacturing a microcomputer based personal information and control system — MAVIS, which is currently undergoing field trials. A word-store terminal, in which depression of a single key causes a whole word or phrase to be generated, called MATE, has been developed at Essex University.

So what of the future? As microcomputer technology advances it will become increasingly valuable to the disabled. One only needs to read books like Chris Evans' The Mighty Micro and use a little imagination to see what computer-based aids could appear soon.

Julia Howlett.

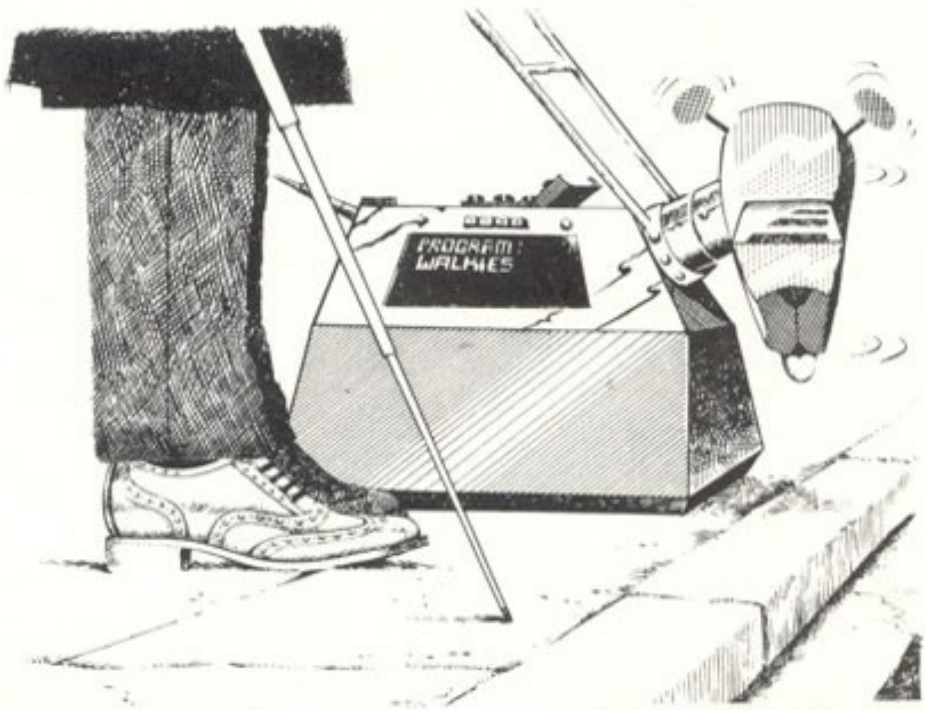


Illustration by Paul Simmons

NEW 10 PART PASCAL SERIES

THE COMPLETE PASCAL

BY SUE EISENBACH AND CHRIS SADLER

CHAPTER 1 WHY PASCAL?

How many people who program picked "their language" by consulting unbiased experts, reading books, and then making a rational choice before they began? Most of us learned our first language because it was the one available at the time. Tied up with learning a first language is learning to mechanize problem-solving, and hence to control a powerful, and captivating, machine — the computer. There's little wonder that some programmers form an emotional loyalty to their language which makes the learning of a new one almost an infidelity.

Because there are a great variety of them available, and because a second language is much easier to learn than the first one, it seems short-sighted to stick passionately to one that was so accidently selected. As designers have gained from past mistakes, so the structure of programming languages has evolved and, while one doesn't want to devote a great deal of time to digesting some fantastic syntactic edifice too big and complicated to run on a real machine, there are several modern languages which may amply repay the trouble taken to learn them. We hope that the potted history that follows will convince you that Pascal, in particular, is worth looking into.

The early machines had no formal languages; they were programmed in machine code. It soon became clear that this procedure was unnecessarily tedious and that the speed and accuracy of the computer itself could be employed to overcome the problem of slow, error-prone transcription. Assembly languages were designed therefore, to allow programmers to use mnemonic codes rather than numbers, and an *assembler* translated these programs into machine code for execution. Using an assembly language, the programmer still had the same control over the computer as with machine code but avoided having to remember and key in long and indistinguishable number codes. Unfortunately an assembly language must be designed for a particular processor so that a program written on one model will not, in general, execute on another.

The second generation of machines proved so much faster than the first that it became no longer at all necessary for the programmer to have complete control over every processor cycle. General-purpose, high-level languages came into being as a consequence. If all processes within a program could be reduced or adapted to a limited number of fixed sequences of machine code instructions, then each such fixed sequence could be represented by a single statement in the high-level language. The problem of the high-level language designer was, therefore, to specify the set of sequences which would effectively cater for the range of problems he wished to solve. The variety of languages developed at this time (as typified by FORTRAN, COBOL and ALGOL 60) aptly demonstrates the differing approaches adopted by their designers. Learning to program came to involve learning the vocabulary of a (human-oriented) language, of which, each word corresponded to a particular operation, or set of operations performed by the computer. Programming became faster to learn and easier to do (although not necessarily easier to do well!) In addition, a program written in a high-level language should, in theory, execute on any machine possessing the appropriate *compiler* (a program that generates machine code sequences from high-level language statements).

Of these early high-level languages, FORTRAN, still the most popular "scientific" language, was closest in style to the actual machine processes. This had two results — firstly, it was relatively straightforward to produce an efficient, fast FORTRAN compiler for a wide range of machines, and secondly, FORTRAN programmers were compelled to structure their algorithms in machine-oriented terms, so that the actual programs were also efficient, and fast. It is these features, its machine-efficiency and wide availability, that account for FORTRAN's continued popularity.

The other two major languages of this period made greater concessions to the human programmer at the expense of a degree of machine efficiency. COBOL, the "business" language, is written in English-like code, has a wide range of data storage and handling facilities, but, compared to FORTRAN is relatively limited in its (mathematical) processing power. ALGOL 60, the "academic" language, was designed, on the other hand, so that the mathematical specification of the problem (the algorithm) could be coded as naturally as possible by the programmer (mathematician) — this criterion has led to the class of "structured" languages, about which more later. ALGOL 60, however, had very limited data-manipulation facilities and neglected to incorporate general Imput/Output routines, thus making ALGOL programs almost completely machine-dependent.

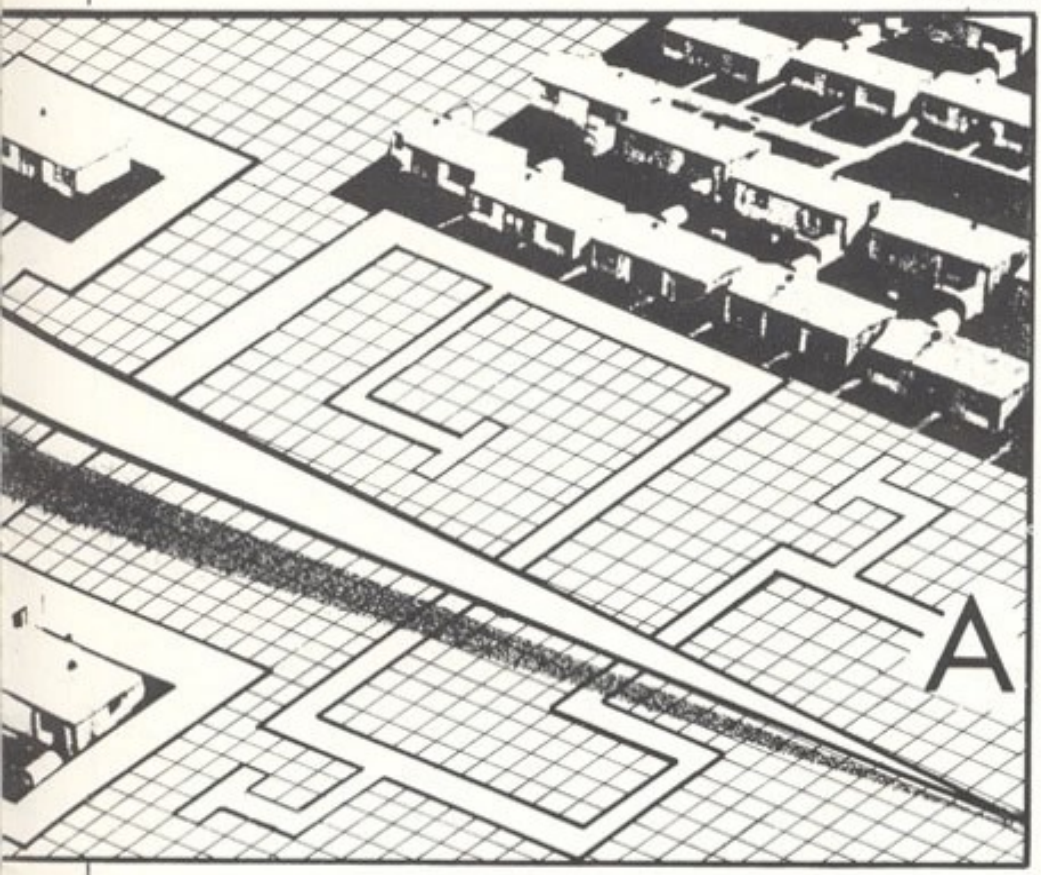
Taken together these languages contain features which can be said to epitomize a well-designed, general purpose high-level language. These are:

1. Algorithms should be expressible in a natural way.
2. A variety of means of storing and handling different types of data should be available.
3. The language should compile and execute efficiently on a variety of machines.
4. I/O instructions should be general and easy to use.
5. The language should be self-consistent and coherent so that it is easy to learn and open to standardization and hence program-probability.

The mid-sixties brought the second generation of high-level languages. Although in some ways these languages represented an improvement over the originals, none of them adequately took account of all of the five points detailed above. PL/1, the "IBM language", attempted to incorporate all known facilities of all the languages into one monolithic creation. However the entire might of IBM could not persuade programmers to drop their FORTRAN or COBOL in favour of such a large, ungainly language which required an immense effort to learn, and which produced large, ungainly programs.

BASIC, the "peoples' language", represents a different tack. It was designed as a teaching language for students who would normally have been taught FORTRAN. It took advantage of the interactive facilities which by then had become available, eliminating the need to learn about compilers, loaders, linkers and editors and allowing the students to concentrate purely on the programming. Although the BASIC syntax is closer to English than

Muscle Films



FORTRAN, and the Input/Output is much simpler, BASIC betrays its parentage in its data handling facilities and in the algorithmic style it imposes on its adherents. While no *standardized* BASIC exists (unlike almost all other languages) variations on the BASIC theme (from 2K Tiny BASIC to monstrous 32K Commercial BASIC *compilers)* abound throughout the micro-world. It is worth noting that very few programs will convert from one system to another without a degree of fiddling about between the different (and often idiosyncratic) dialects.

The mid-sixties also saw the convening of the working party of the International Federation for Information Processing (IFIP) who had previously produced ALGOL 60. This group felt that the algorithmic approach of ALGOL 60 was correct and that its limited impact had been due to its restricted data-handling facilities and non-existent general I/O statements. Their final report on ALGOL 68 appeared in early 1969. ALGOL 68 was designed to be as comprehensive as PL/1 in that the failings of ALGOL 60 had been more than made up, but the compactness of the ALGOL 60 design formalism gave it a greater degree of coherence over PL/1. It is still a vast and complex language which is difficult to learn and even more difficult to implement on the huge machines it requires. Also, as a European language, it was slow to be implemented on hardware designed, manufactured and primarily marketed in the USA.

Niklaus Wirth, a member of the IFIP working party, watched his colleagues painstakingly constructing their heavyweight language with a certain measure of disquiet. He looked for one which would not only incorporate the strength of previous languages but which could also be implemented on any reasonably-sized computer (like FORTRAN). He (rightly) predicted that ALGOL 68 would not fall into this category and so designed his own language, Pascal, as an ALGOL-like language in machine-realizable form (i.e. easy to compile and not too big).

Actually, Pascal didn't catch on straight away either, even though it is implementable on a wide variety of machines and contains the major features of ALGOL 68. However, today, with the onslaught of micro-computers (requiring smallish, efficient compilers) and with the demand amongst professional programmers for "structured" programming languages, Pascal, the "five-star" language is coming into its own. (See Table 1)

①	ALGORITHMIC STYLE	RICH DATA STRUCTURES	MACHINE EFFICIENT	I/O SIMPLE	PORTABLE AND COHERENT
ASSEMBLY LANGUAGE			●		
FORTRAN			●		●
COBOL		●		●	
ALGOL 60	●		●		
PL/2	●	●		●	
BASIC				●	
ALGOL 68	●	●		●	
PASCAL	●	●	●	●	●

It may seem as though the unblinking row of "stars" against Pascal betrays a fair amount of bias; the real point, however, is that Pascal was *designed* with just these principles in mind. Of course there have been other languages designed on completely different lines (take, for instance, APL), but if the construction philosophy is to be based on the five points that have been established above, then Pascal must certainly be worth a closer look. While this survey could in no way be described as being fully comprehensive, we have included the major *micro* languages for comparison.

Pascal nuts and bolts

Any algorithm or computer program consists of two complementary elements — a description of the *"actions"* which are to be performed, and a description of the *data* with which the actions are to be performed.

Clearly, any programming language must supply one with the means of expressing these two descriptions and the better the language, the more elegant, efficient, understandable (readable) and error-free the programs of a given programmer are likely to be. The aim of Pascal is to provide a set of statements (action) and data types which are at the same time comprehensive (to cover all eventualities) and succinct (to convey the meaning without getting too complicated). Anyone familiar with another language will recognise that the actions and data types described can usually be accomplished in that language too, so that the comparison is not so much about *what* they can do but more *how* elegantly and efficiently they do it.

Data types

In machine code, the big patterns in a location may be interpreted in any way the programmer chooses. In high-level languages, the designer must decide what interpretations the programmer will be permitted to employ. In BASIC, for instance, information is held as REAL numbers (with or without a decimal point) or STRINGS of characters. These are known as data types. Pascal has four data types:

BOOLEAN — has values: TRUE or FALSE

INTEGER — has values: an implementable subset of the whole numbers.

REAL — has values: an implementable subset of floating point real numbers (REALS usually occupy two stage locations in contrast to INTEGERS which are only allocated one)

CHAR — has values: a single alphabetic, numeric or special character.

Pascal also offers the facility of defining a new data type if the above 4 seem restrictive. For instance:

TYPE SUIT = (HEART, DIAMOND, CLUB, SPADE) defines SUIT as a data type which can have any "value" as specified in parenthesis.

likewise:

TYPE DAY = 1. . .31 allows only one of the define values, and no other, to be assigned of a variable of the DAY type.

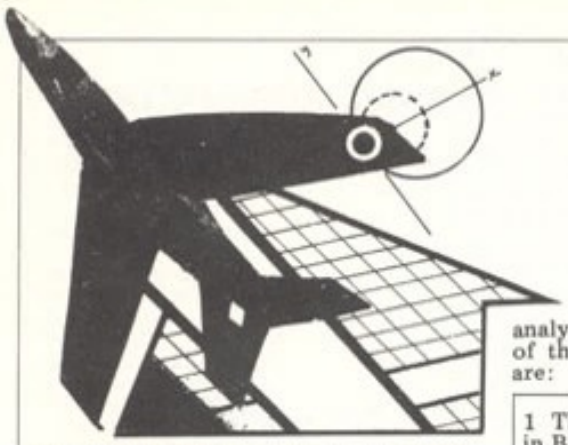
Similarly:

TYPE STRING = ARRAY (1. . .15) OF CHAR defines a 15-element BASIC-like character string.

while:

TYPE LONGSTRING = ARRAY (1. . . 80) OF CHAR would be suitable to describe a punched-card format record.

Incidently, the Pascal rule for the naming of objects (e.g. types, variables, constants, procedures and functions) is that the name should begin with a letter of the alphabet and may be arbitrarily long. Compilers for small machines may only recognise the first eight characters (which should therefore be unique) but



this is no reason to restrict the name-size to eight characters. The more self-explanatory a name can be chosen to be, the less actual explanation is required elsewhere in the program.

In addition to its data types, Pascal allows for the creation of certain data *structures*, namely ARRAYs, RECORDs, SETs and FILEs. An ARRAY is a data structure containing two or more elements of the same type. These are particularly useful in certain mathematical algorithms. The commercial counterpart is a RECORD consisting of two or more "fields" which may be of any type. SETs are a special case of ARRAYs while FILEs provide the means whereby data in any form can be extracted from or introduced to the program from (external) mass storage. Pascal enables the programmer to pack data to maximum density by enabling the specification of variable ranges and limited range data-types.

Most programmers agree that any computable problem can eventually be divided and subdivided and thence analysed (and so programmed) in terms of three fundamental operations. These are:

1 The Sequence	
in BASIC	in PASCAL
LET A=B+C	SUM:= NEXTONE
or just A=B+C	+ SUM;
and GOSUB 4900	and
	DEALCARDS or
	DEALCARDS
	(FIRSTPLAYER)
2 The Loop	
in BASIC	in PASCAL
FOR I = 1 to 10	a) WHILE Condition DO action,
NEXT I	b) REPEAT action UNTIL condition
	c) FOR I:=M TO N DO action
	FOR I:=M DOWN TO N DO action
3 The Conditional	
in BASIC	in PASCAL
a) IF condition THEN Action	a) IF condition THEN Action ELSE alternative
b) ON K GO TO 100, 200, 300	b) CASE N OF A : ACTION 1; B : ACTION 2; etc. end

Control structures

There are obviously other constructs which could be employed, and, on the other hand, any loop construct could be replaced by an equivalent conditional. So why does Pascal seem to have 3 different forms of loop? The answer appears to lie in the delicate balance between supplying the programmer with enough techniques to do his work elegantly and adequately while keeping the language simple enough to operate it efficiently and effectively. Pascal is, at the same time, comprehensive as a problem solver, and succinct as a language.

Of course, this brief description doesn't illustrate Pascal any more than a dictionary describes the English language. What is needed are some examples used in actual programs, and also some practice in formulating the problems in the "Pascal way". That is just what will be offered in the next 9 issues of APC — a systematic exposition of Pascal with a host of illustrative examples and some exercises every month. As few readers are likely to have Pascal up and running on their machines, we are willing to look at exercises you would like to submit. The most interesting programs may even find their way into APC.

IN STORE

AUSTRALIA'S MOST UP-TO-DATE BUYER'S GUIDE
FOR MICROCOMPUTERS

Month by month, every effort will be made to keep
In Store up-to-date and accurate.
And that means APC will always be happy to hear from its readers
of any errors, and additions that seem worthy of inclusion.

LIST OF ABREVIATIONS

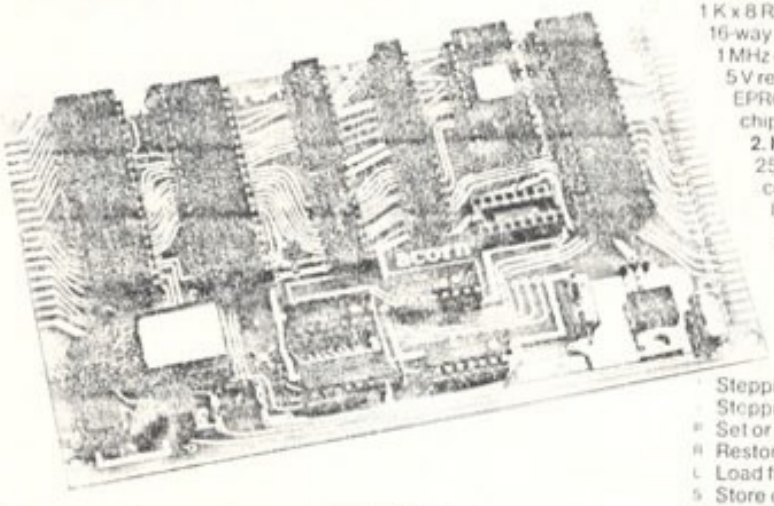
A – Assembler
A/D – Analog to Digital
B/W – Black and White
C – Cassette
cps – Characters per second
Doc – Documentation
E – Extensive
Ed – Editor
Ex – Extended
F/D – Floppy Disc
H – Hardware
I – Introductory
I/O – Input / Output
int – Interface
K/B – Keyboard
M/A – Macroassembler
N/A – Not Available
N/P – Numeric Pad
O/S – Operating System
P/P – Parallel Port
RAM – Random Access Memory
ROM – Read Only Memory
res – Resolution
S – Software
S/P – Serial Port
T/E – Text Editor
U – Utility
VDU – Video Display Unit

All prices shown are exclusive of sales tax, except where indicated by an asterix.

Software items listed in *italics* are not included in the basic price of the equipment.

Name of Machine	Main Distributor & Phone No	Hardware	Software	Doc	Price	Comments
Apple II plus	Computerland (03) 62 5581 (02) 29 3753	16-48K RAM: 6502: colour VDU int.: 81/O slots: games paddles: option 5¼" F/D (116K) and 11 MB disc	O/S: BASIC: *Pascal:* games	E	$1395	280x192 high res colour graphics: Applesoft BASIC in 12K ROM
Century	Abacus Computer Store (03) 429 5844	C100, 48-64K RAM: Z80: 12" VDU: 2x5¼" F/D (2x143K): 112 cps printer: RS232 port: S100 bus: C200 includes 2xF/D (2x315K): hard disc: 4xRS232: 2xP/P	*COBOL: FORTRAN: BASIC*	I	C100– $4950: C200– $5400	Also available: C300
Challenger IP	Systems Automation (02) 439 6477	4-32K RAM: 6502: C int: 24x32 VDU int: RS232 port: option – dual 5¼" F/D (140K)	O/S: BASIC: A: games	I	$448	8K microsoft BASIC in ROM: expansion board available
Challenger 4	Systems Automation (02) 439 6477	8-48K RAM: 6502: colour 32x64 VDU int: RS232 port: P/P: option – 6502C microprocessor; dual 5¼" F/D (140K)	BASIC: *Pascal*	I	$871	BASIC in 8K ROM
Compucolor II	Anderson Digital Equipment (03) 543 2077	8-32K RAM: 8086: 13", 32x64 8 colour VDU: single 5¼" F/D (51K): RS232 port	ExBASIC(ROM):*A*	I	$2095	16K model, $2395: 32K $2695: maintenance manual available
Cromenco System 2, System Z2H, System 3	–	64-512K RAM: Z80A: System 2, dual 5¼" F/D (346K): System Z2H, also Winchester disc (11MB): System 3, 8" dual 1MB: S/P: P/P	CDOS: *BASIC: COBOL: FORTRAN: M/A: ExBASIC: Structured BASIC*	E	System2 $3990 System Z2H $9650 System 3, $6750	All Systems expandable to multi user (2-7 users) $2880–$8825
Exidy Sorcerer Model II	Dick Smith Electronics (02)888 3200	8-48K RAM: Z80: 30x64 VDU int.: RS232 Port: P/P: S100 bus:extra C int.	O/S: ExBASIC (ROM): *M/DOS: CP/M*	I	$1295*	High res graphics capability: 16K version $1395*: 32K version $1525*: 48K version $1655*: User programmable character set

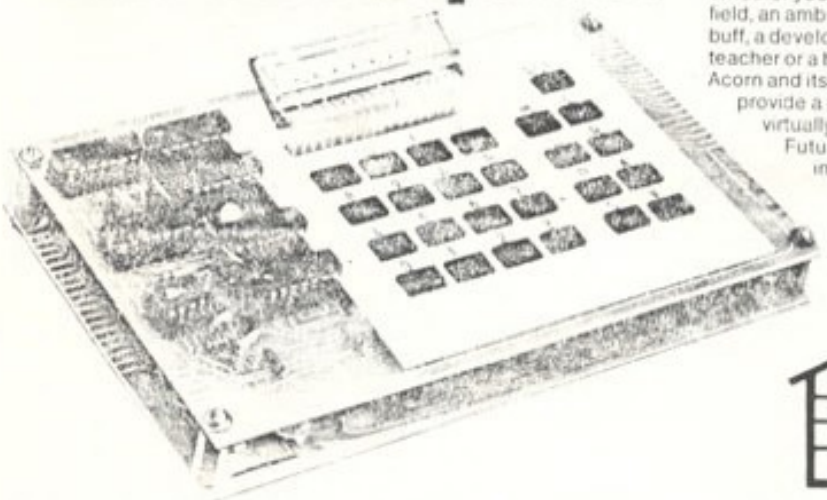
Introducing Acorn



A professional MPU card

Designed as a general purpose industrial controller based on the 6502 MPU, this card is complemented by a matching Eurocard hex keyboard and CUTS standard cassette interface, to create the new...

Acorn Microcomputer



This compact stand-alone micro-computer is based on standard Eurocard modules, and employs the highly popular 6502 MPU (as used in APPLE, PET, KIM, etc). Throughout, the design philosophy has been to provide full expandability, versatility and economy. Take a look at the full specification, and see how Acorn meets your requirements

Acorn technical specification

The Acorn consists of two single Eurocards:

1. MPU card
6502 microprocessor
512 x 8 ACORN monitor
1 K x 8 RAM
16-way I/O with 128 bytes of RAM
1 MHz crystal
5 V regulator, sockets for 2 K EPROM and second RAM I/O chip

2. Keyboard card
25 click-keys (16 hex, 9 control)
8 digit, 7 segment display
CUTS standard crystal controlled tape interface circuitry

Keyboard Instructions:
M Memory Inspect/Change (remembers last address used)
Stepping up through memory
Stepping down through memory
P Set or clear break point
R Restore from break
L Load from tape
S Store on tape
G Go (recalls last address used)
RST Reset

Compact, easy to use Acorn Monitor includes the following features:

- System program
- Set of sub-routines for use in programming
- Powerful de-bugging facility displays all internal registers
- Tape load and store routines

Acorn – with real expandability!

The standard Acorn is fully expandable to 65 K of memory and the Acorn bus is available on the 64-way edge-connector. Whether you're a beginner in the field, an ambitious home computer buff, a development engineer, a teacher or a businessman, the Acorn and its family of modules will provide a practical solution in virtually every situation.

Future expansion for Acorn includes the following software and hardware:

Software
Basic interpreter, assembler, dis-assembler, editor, TTY and disk operating system

Hardware
Memory-mapped VDU system (with upper and lower case ascii graphics and hardware scroll) floppy disk controller for 5¼ in and 7 in disks, a memory card with 8 K bytes of static RAM (2716) and 4 K bytes of EPROM (2114), a PROM programmer (for all types of PROM usable on ACORN a full ascii keyboard, a backboard for the ACORN bus, and a Eurocard racking system.

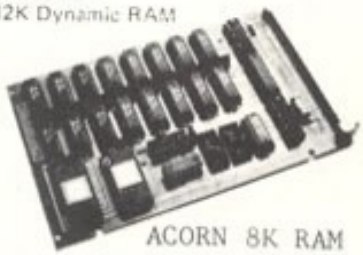
Acorn Operating Manual

With Acorn, you'll receive an operating manual that covers computing in full, from first principles of binary arithmetic, to effecient hex programming with the 6502 instruction set. The manual also includes a listing of the monitor programs and the instruction set, and other useful tabulations; plus a selection of 12 interesting and educative program samples.

Prices start from $110...Come in for a demonstration of ACORN Systems.
M-F 0930-1730h
Sat 1000-1300h

ACORN VDU

The present range includes:
6502 CPU/Single board controller
Hex keypad/CUTS interface
8K + 8K static memory card
Colour VDU interface
Universal Interface
Analogue/digital + Digital/analogue
Floppy disc controller
6809 C.P.U.
32K Dynamic RAM



ACORN 8K RAM

Cottage Computers

386 QUEENS PARADE,
FITZROY NORTH VIC. 3068
Tel: (03) 481 1975

bankcard welcome here

IN STORE

Name of Machine	Main Distributor & Phone No	Hardware	Software	Doc	Price	Comments
HP-85	Hewlett Packard Australia (03)89 6351	16-32K RAM: N/A: 5", 16x32 B/W VDU: C(200K): 64 cps printer: RS232 port: 4 x P/P	BASIC	S	$3550	Full dot matrix graphics: N/P: compact portable unit
IPS-100	Microprocessor Applications (03) 754 5108	32-896K RAM: 8085: 2 RS232 ports: S100 bus: dual 5¼" F/D (630K)	O/S: Ex BASIC: Ed: A: *CP/M: CBASIC: FORTRAN: COBOL*	E	$3750	
Microengine	Daneva Control (03)598 9207	64K RAM: MCP 1600: 2 x RS232 ports: 2xP/P: Options – dual 5¼" F/D (Single or dble density); 8" F/D (single or dble density)	BASIC: Pascal: File Manager: U	E	$2995	Also available as board
North Star Horizon	Melbourne's Byte Shop (03) 568 4022	32-64K RAM: Z80A: 5¼" F/D (170K): 2xS/P: 1P/P: optional – VDU ($1350); Quad density F/D	DOS: BASIC: *COBOL: FORTRAN: Pascal: CP/M: M/A*	E	$2695	
Pet 2001	Hanimex (02) 938 0400	8-32K RAM: 6502: C: 9" 25x40 B/W VDU: extra C Int: IEEE488 port	O/S: BASIC: *A*	I	8K$1199 16K$1859 32K$2249	Options – dual 5¼" F/D (353K), $2329: $109 for disc operating ROM
Sord M100 ACE III	Alliance Digital Corporation (02) 436 1600	48K RAM: Z80: 24x64, 12" VDU: RS232 ports: 2x5¼" F/D (2x143K): S100 bus: 2 octave speaker: A/D Conv.: option – 8 colour graphic controller ($1450)	O/S: Ex*BASIC FORTRAN*	I	$4500	M100 ACE IV – 8 colour graphics controller incl.
Sord M223	Alliance Digital Corporation (02) 436 1600	64K RAM: Z80: 12", 24x80 VDU: 2xRS232 port: S100 bus: 5¼" F/D (350K)	O/S: *ExBASIC: FORTRAN: COBOL*	I	$7500	
TRS-80 Level 1	Tandy Electronics (02) 638 6633	4-16K RAM: Z80: C: 12", 16x64 B/W VDU	BASIC: Games: A	I	$699*	BASIC in 4K ROM: upgradable to Level 2
TRS-80 Level 2	Tandy Electronics (02) 638 6633	4-48K RAM: Z80: C:12", 16x64 B/W VDU: RS232 port: P/P	BASIC: *M/A: FORTRAN: COBOL*		$879*	16K machine includes N/P: 4-16K upgrade $320*: ($250* without N/P): max. config. $1169*: option – single 5¼" F/D (78K), (max of 4)
Vector Graphics System B	AJ & JW Dicker (02) 524 5639	64K RAM: Z80: Dual 5¼" F/D (630K): 12", 24x80 B/W VDU: S/P: 2xP/P	DOS: BASIC: A: CP/M: Ed	E	$6350	Graphics and numeric pad.
Versatile 4	Microprocessor Applications (03) 754 5108	32-56K RAM: 8085: 9", 24x80 B/W VDU: dual 5¼" F/D (630K): S100 bus: 2xRS232	MBASIC: MDOS (including T/E and A): *Version 4 MDOS AND BASIC: CP/M*	E	$5692	

SINGLE BOARDS

Acorn	Cottage Computers (03) 481 1975	1-8K RAM: 6502: EPROM socket: Hex K/B: C int: 8 digit LED display: up to 16 ports: options – Eurocard 64 way connector; VDU card; Full K/B card	½K monitor: *BASIC*	S& H	System1 $285: SystemII $1224: SystemIII (incl.a5¼" F/D), $2763	Universal interface card available.
Aim 65	Dwell Pty Ltd (02) 487 3111	1-4K RAM: 6502: 8K ROM: full K/B: 20 character LED display: 20 character thermal printer: Cx2 int: 1 P/P	8K monitor in ROM: *A: BASIC*	E	$525*	Case available $75*
SBC100	Microtrix (03) 718 2581	1K RAM: Z80: 8K ROM: S100 bus: 1S/P: 1P/P	1K monitor: DOS in ROM	E	$299	Also available assembled $374
Superboard	Systems Automation (02) 439 6477	4-32K RAM: 6502: 10K ROM: full K/B: 24x32 VDU Int: C int: options– RS232; dual 5¼" F/D (140K)	BASIC: games	I	$360	BASIC in 8K ROM

BUZZWORDS

A

A/D (Analog to Digital)
Process by which information is transformed from an analog, or continuous, form to a digital, or discrete form.

Access Time
The time required to locate and make available information in a storage location.

Accumulation
One or more registers associated with the ALU where sums and other arithmetical and logical results of the ALU may be temporarily stored and/or manipulated.

Address
A coded instruction designating the location of data or program segments in storage. The address may refer to storage in registers or memories or both. The address code itself may be stored so that a location may contain the address of data rather than the data itself.

ALGOL – ALGOrithmetic Language
A language designed to facilitate the implementation of algorithms.

Algorithm
A defined set of rules processes which lead to the solution of a specific problem within a finite number of operations.

Alphanumeric
Referring to a set of characters consisting of both numbers and letters.

ALU-Arithmetic and Logic Unit
The ALU is one of three essential components of a microprocessor, the other two being the registers and the control block. The ALU performs various forms of addition and subtraction; and logical operations such as ANDing and ORing.

Analog Signals
Electrical representations of continuous physical variables such as pressure, temperature or humidity.

Arithmetic Instruction
An instruction causing the microprocessor to perform a mathematical operation such as addition or ANDing.

Artificial Intelligence
The ability of a computer to learn from experience, perform advanced logical operations and other tasks characteristic of human intelligence.

ASCII – American Standard Code for Information Interchange
A standardized code for character transmission using seven binary digits. The resulting binary patterns represent 128 alphanumeric characters.

Assembler Program
Translates man-readable source statements (mnemonics) into machine understandable object code, and assigns memory locations for variables and constants.

Assembly Language
A machine orientated language. Normally the program is written as a series of source statements using mnemonic symbols that suggest the definition of the instruction and is then translated into machine language.

Audio Magnetic Tape Storage Unit
A unit which uses audio cassettes to store audio tones as representations of computer data and programs.

B

BASIC – Beginners All Purpose Symbolic Instruction Code
The most popular programming language for microcomputers.

Baudot Code
A specific code using five bits to represent alphanumeric characters.

BCD - Binary Coded Decimal
A specific code using four bits to represent the numeric characters 0 to 9. Not all possible binary patterns (totalling 16) are utilized.

Binary
Carrying only two possible alternatives, for example: '1' or '0'.

Binary Code
A code employing only binary characters.

Binary to Hexadecimal Conversion
The process by which binary representations of numerical values are converted to equivalent hexadecimal representations.

Bit
Derived from the term binary digit. It constitutes the smallest unit of information having possible values of '1' or '0'.

Branch
This refers to the capabilities of a microprocessor to modify the function or program sequence. Such modification may depend on the actual content of the data being processed at any given instant.

Baud Rate
A measure of data flow. The number of signal elements per second based on the duration of the shortest element. When each element carries one bit, the Baud rate is numerically equal to bits per second (bps).

Breakpoint
A program point indicated by a breakpoint flag which invites interruption to give the user the opportunity to check the program before continuing to its completion.

Buffer
A circuit inserted between other circuit elements to prevent interactions, to match impedances, to supply additional drive capability, or to delay rate of information flow. Buffers may be inverting or non-inverting.

Bus Driver
An IC which is added to the data bus system to facilitate proper drive to the CPU when several memories are tied to the data bus line. These are necessary because of capacitive loading which slows down the data transfer rate and prevents proper time sequencing of microprocessing operations.

Bus System
A network of paths inside the microprocessor which facilitate data flow. The important buses in a microprocessor are identified as Data Bus, Address Bus, and Control Bus.

Byte
A pre-determined number of consecutive bits treated as an entity, e.g. 4-bit or 8-bit words.

C

Chip
An integrated circuit (more strictly, the silicon chip upon which a monolithic circuit may be built).

Clock
The master oscillator which generates the timing impulses for a computer's central processor.

COBOL – COmmon Business Orientated Language
An international standard program language intended for commercial use.

Compare
A computer function which examines one number in relation to another and commonly sets switches according to whether the first is greater than, equal to or less than, the other. It is usual to follow the compare instruction with a conditional jump, dependent on the result of the comparison.

Compiler
A program which translates another program from high level language into machine language prior to the execution of that program.

Control Bus
The electrical route taken by signals from the control logic in a computer to, e.g., fetch the next program instruction from memory.

CORAL – Computer On-Line Real-time Applications Language.
This is a development of ALGOL. CORAL 66 was originally developed by the RRE for military projects, but has since been adapted for commercial applications.

Core
(Ferrite core memory). An early form of random access memory composed of tiny magnetic rings.

CPU – Central Processing Unit
The main part of a computer, containing immediate access storage, arithmetic and logical units and special register groups. In effect it is those parts of a computer other than input, output and peripheral devices and, in some cases, also excluding immediate access storage areas which may be provided in separate modules.

Cross-Assembler
A program assembler used on a computer other than the one on which the object program is to be run.

CRT – Cathode Ray Tube
A vacuum tube which utilizes an electron beam to fluoresce a phosphor covered screen, an example being the screen of a television set.

Cursor
The screen character (typically an underlining dash) in a video display which indicates where the next character is due to appear.

D

D/A (Digital to Analog)
Process by which information is transformed from a digital, or discrete, form to an analog or continuous form.

Data Bus
The bus, or wiring that carries data between different parts of a computer system.

Debug
To trace and correct errors in software or hardware so that a program will function as expected.

Dedicated
A microprocessor system that has been specifically programmed for a single application such as weight measurement or traffic light control.

Digital Magnetic Tape Storage Unit
A unit which uses magnetic tape to store binary representations of computer data and programs.

Disc Crash
The destruction or damage to a magnetic disc as a result of direct contact between the surface of the disc and the read/write head.

Disc Memory Unit
A form of data storage, similar in function to magnetic recording tape, but in the form of a disc which is rotated at high speed, so that access to any part of storage can be achieved quickly by a movable read/write head. Discs are also characterized by a high rate of data transfer.

Diskette
See 'Floppy Disc Unit'.

DMA – Direct Memory Access
A method of gaining access to memory elements to achieve data transfer without CPU involvement. The process is controlled by devices other than the CPU and permits direct transfer of data between memory and these devices.

E

EAROM – Electrically Alterable Read Only Memory
A form of ROM wherein the contents stored may be altered by an appropriate electrical current.

Editor Program
Software which facilitates examination and alteration of text contained in the memory of the computer.

EPROM – Erasable Programmable Read Only Memory
A PROM whose contents may be erased (usually under exposure to ultraviolet light) and reprogrammed.

Even Parity
A convention for checking data after transmission: an even number of ones is expected in each group of bits transmitted.

Execution Time
The time taken by a computer to perform an instruction (such as add); usually measured in clock cycles.

Exclusive "OR" Operator
A logical operator (Boolean algebra) which has the property that, if P and Q are two statements, the statement P*Q (where the asterisk is an exclusive OR operator) is true if either P or Q, but not both, is true and is false if P and Q are both false or both true.

PUZZLES

LEISURE LINES

With J. J. Clessa

We hope to bring to you each month a page of diversions, puzzles and competitions. There will be chances to win prizes and although "Leisure Lines" will primarily be concerned with puzzles, we expect from time to time to digress a little and maybe even tell the occasional joke.

Quickie

This month's quickie – no prizes, no answers given . . . a 2-digit number, read from left to right, is 4½ times as large as the same number read from right to left. What is the number?

Prize Puzzle 1a

This, the first of this month's prize puzzles, is designed to test your logic and reasoning ability. A prize goes to the sender of the first correct entry out of the bag. On the Sydney to Melbourne air service are three passengers, named Brown, Jones and Smith. By coincidence the pilot, co-pilot and cabin steward on the aircraft are also named Brown, Jones and Smith – but not necessarily respectively.

1. Passenger Jones earns $10,400 per annum.

2. The cabin steward lives midway between Sydney and Melbourne.

3. Smith, the crew member, is married to the co-pilot's sister.

4. The passenger with the same last name as the cabin steward lives in Melbourne.

5. The passenger who lives nearest the cabin steward earns exactly three times as much per week as the cabin steward.

6. Passenger Brown lives in Sydney.

What is the pilot's name?

Prize Puzzle 1b

You are permitted to solve this by any means at your disposal – including the offer of bribes to J.J. Clessa (who's address will NOT be given. Ed).

A Pythagorean triangle is a right angled triangle who's sides are an exact number of units in length – and of course, it conforms to the famous rule:

"The square of the hypotenuse is equal to the sum of the squares on the other two sides . . ." Probably the best known Pythagorean triangle has sides 3, 4 and 5 units, and, of course, $5^2 = 3^2 + 4^2$.

The area of the triangle is obtained by halving the product of the two smaller sides: ½ (3x4) = 12 units. The perimeter of the triangle is the sum of the three sides: 3 + 4 + 5 = 12 units.

(Sorry to go through all that, for those who already knew it, but I wanted to be sure that everyone starts equal).

I'd like you to find the smallest area Pythagorean triangle who's perimeter is a perfect square and who's area is a perfect cube.

Please send answers in a separate envelope for each puzzle, addressed to: Puzzle (1a or 1b), Australian Personal Computer, P.O. Box 250, North Sydney, 2060. Winners will be notified by post and results, plus solutions, will be published in the next-but-one edition of APC. The Editor reserves the right to make final decisions on all matters.

Prizes

Alright, you asked for it, this month the "Silly Prize" for each competition will be 2.5 kg of Cadbury's Dairy Milk Chocolate.

Before leaving, I'd like to stress that "Leisure Lines" is designed to be a two-way thing. I'd welcome any offerings of puzzles, jokes, ideas, etc. All contributions will be gratefully received and acknowledged if used.

COMPUTER LANGUAGES

The developments and improvements that have occurred during the evolution of programming languages can be interpreted as responses to the needs of the programmer. Increasingly sophisticated languages have provided him with much needed tools and techniques for the solution of ever more difficult and complex problems. The aims of this article are to describe the development of computer languages, to give the flavour of some of them and to examine the effects of language developments on personal computing.

Introduction

The process of solving a problem with the aid of a computer can be divided into three parts:

i) specify the problem,
ii) find a method to solve the problem, and
iii) communicate the method to the computer.

The first step seems obvious. However, on occasion, superb programs have been written to provide the solution to a problem which, on further investigation, turned out to be the wrong problem. The second step is to find or devise a suitable algorithm. In a business environment, the first two steps are, essentially, what is known as systems analysis. Step three relates to the subject of this article, for the way to communicate a solution method to a computer is to express it in a language that the computer can understand. This is, in essence, what a computer programmer does (he may, of course, carry out the work involved in steps one and two as well).

The aims of this article are, then, to review the development of computer languages, to illustrate features of some of the languages and their applications, and, finally, to examine the impact of computer languages on personal computing.

Machine Code

All computers, whether large mainframe computers or microcomputers, can only store, and operate on, binary digits. Data and instructions are stored and understood by the computer as patterns of binary digits.

Also, each computer has its own set of instructions, with one instruction for each action that the computer can perform. Thus, a computer, fundamentally, can only obey an instruction that belongs to its instruction set, and it can only understand an instruction when it is expressed as a binary pattern. Another way of expressing the same ideas is to say that a computer can only understand its own machine code. A segment of a machine code program for an eight-bit computer is listed as Program 1.

It is clear that, while machine code is suitable for computers, it is far from ideal for people. We will return to the machine code program to consider its meaning shortly.

```
00001100
00101111
00100111
10110100
```

Program 1. Machine code program.

Programmers who had to write machine code programs naturally found it impossible to develop their programs from scratch as sequences of binary patterns, since mistakes were easy to make but hard to detect. They soon developed the habit of assigning each instruction a short mnemonic, such as LDA for LoaD a number into the Accumulator and STA for STore the contents of the Accumulator. Only at the final stage of loading the programs into the machine were the mnemonics replaced by their binary codes. In due course, it was realised that this coding chore could be undertaken by the computer itself and so assembly codes originated. A short assembly code program is given as Program 2.

```
LDA  12
ADD  15
ADD  7
STA  20
```

Program 2. Assembly code program.

This program finds the sum of three numbers, previously stored in the locations with addresses 12, 15 and 7, by loading the contents of location 12 into the accumulator, adding the contents of location 15 to it and adding the contents of location 7 to that: the sum is stored in location 20 so that it is preserved and the accumulator is available for other computations. The program has a degree of generality, for it can compute the sum of any three numbers that are stored in locations 12, 15 and 7.

The assembly code instructions of Program 2 are each equivalent to one machine code instruction. This is true of most of the instructions in any assembly code, although facilities available with some assembly codes, most notably macro-processors, make certain instructions equivalent to several machine code instructions. Assembly code instructions cannot be executed directly, but must first be translated to machine code. This translation is performed by a special program called an assembler: its operation is represented diagrammatically in Figure 1. To illustrate the basic principles of operation of an assembler, Program 2 can be assembled manually to provide the equivalent machine code program with the aid of the code in Table 1. To translate the instruction LDA 12 to machine code the mnemonic LDA is replaced by its code 000 and the decimal address 12 is replaced by its (five digit) binary equivalent 01100 to give the eight (binary) digit pattern 00001100. The same process applied to the other instructions converts Program 2 to Program 1.

mnemonic	code
LDA	000
ADD	001
SUB	010
STA	011
JUN	100
JEQ	101
EOR	110
STOP	111

Table 1. Three digit code for eight operations.

Trade Off

When machine code instructions are represented by patterns of eight binary digits, there is a trade-off

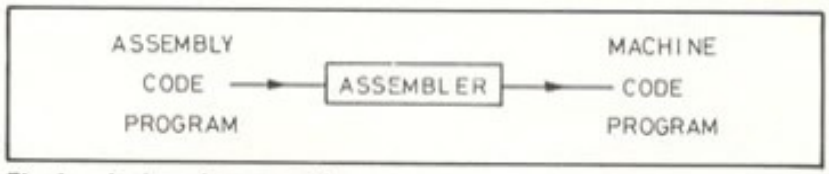


Fig. 1. Action of an assembler.

between the number of operations that can be represented and the number of locations that can be addressed. In the example just discussed, assigning three digits for the operation code allows $2^3 = 8$ operations and $2^5 = 32$ different addresses to be represented. If two digit operation codes are allocated then only $2^2 = 4$ operations can be represented (see, for example, Table 2) but $2^6 = 64$ addresses can be accessed. It should be clear that representing instructions by patterns of eight binary digits imposes considerable restrictions. These restrictions do not apply to computers with longer word lengths. It is probably necessary to use two (8-bit) words to represent instructions in an eight-bit machine.

mnemonic	*code*
LDA	00
ADD	01
SUB	10
STA	11

Table 2. Two digit code for four operations.

Arithmetic: ADC, SBC
Logical: AND, EOR, ORA
Load and Store: LDA, LDX, LDY, STA, STX, STY
Jump and Branch: BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS, JMP, JSR, RTS
Increment and Decrement: DEC, DEX, DEY, INC, INX, INY
Transfer: TAX, TAY, TSX, TXA, TXS, TYA
Shift and Rotate: ASL, LSR, ROL, ROR
Compare: CMP, CPX, CPY
and fifteen others, including NOP

Table 3. Instruction set of the 6502 microprocessor.

A typical instruction set, that of the MOS Technology 6502 microprocessor (which is used in KIM and PET), is listed in Table 3. It is worth stressing that only two of the 56 instructions are arithmetic, implying quite clearly that the microprocessor is not primarily intended for 'number crunching'.

Auto Codes

The next development in computer languages was the emergence of autocodes. The early, relatively primitive, autocodes have instructions of the form

A = A + B
X = A + C

These two instructions cause the values of the variables A, B and C to be added and assigned to the variable X. The introduction of variable names relieves the programmer of address management chores by passing this task to the translation program. Primitive autocode instructions of this type are equivalent to three assembly code instructions – a load instruction and a store instruction – so that the programmer now handles a larger amount of computation per instruction than with an assembly code.

However, the task of adding three numbers still cannot be achieved with a single instruction. More sophisticated autocodes were developed, including Mercury Autocode, Extended Mercury Autocode and Atlas Autocode.

Each of these languages is specific to a particular machine, but the latter, at least, deserves to be classified as a high-level language, in the sense that its instructions can describe a large amount of computation, being, in appearance, very similar to those of Algol. The development of autocodes overlaps the emergence of the so-called high-level languages. The most widely used ones, in scientific applications, are FORTRAN, BASIC and Algol 60. In each of these languages three numbers can be added together by executing a single instruction. The instructions are:

FORTRAN:- X = A + B + C
BASIC:- LET X = A + B + C
Algol:- x : = a + b + c

The trend of these developments was to throw an increasing amount of the mundane management chores onto the translation program, thereby relieving the programmer and absolving him from the need to understand the detailed workings of the computer he was using. Also, single instructions describe increasingly large amounts of computation, implying that they are equivalent to increasing numbers of machine code instructions. This gives the programmer a better chance of comprehending his entire program; in a phrase, of seeing the wood rather than the trees.

The translator for a high-level language program is called a compiler. Its operation is represented diagrammatically in Figure 2. Clearly, a compiler performs a much more complex task than an assembler. A compiler translates a complete high-level language program to a complete machine code program prior to its execution. A program to translate a high-level language for execution one line at a time is called an interpreter. Although the use of an interpreter increases program execution times, it makes possible interactive working.

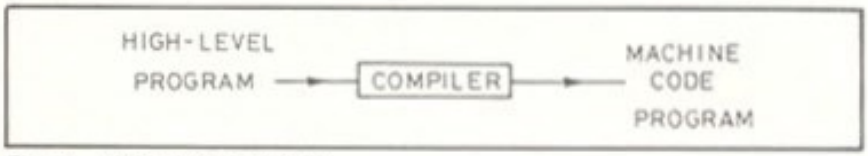


Fig. 2. Action of a compiler.

High-level Languages

FORTRAN was the first of the high-level languages, emerging from IBM in 1957. Algol 60 was formally defined by a report dated 1960, and implementations followed that date. These two languages were intended for scientific and engineering applications. BASIC, which was intended as a teaching language that would be easy to learn and to use, was devised at Dartmouth College, USA, in the early 1960s. Meanwhile, COBOL had arrived as a high-level language for business applications.

The widespread use of high-level languages, and, of course, their compilers, gave rise to the 'virtual machine concept'. Broadly, this is the idea that a computer can seem to understand FORTRAN (or, indeed, any other language) when the appropriate compiler is stored in its memory. Programs written in a high-level language can then be automatically translated to machine code and run by the computer in a way that need not concern the high-level language programmer at all. Thus, by using the stored program facility to store a translation program, a computer appears to be something it really is not.

These languages, while by no means the only ones, are typical of the early high-level languages. The remainder of this section is devoted to an account and comparison of some of their features.

The features to be considered are:-

i) *Input and output*

Input, in the form of a value for a variable, A, can be obtained by the following instructions in FORTRAN, BASIC and Algol, respectively:

```
READ (1,100) A      (FORTRAN)
100 FORMAT (F10.2)
INPUT A   (BASIC)
a: = read   (Algol)
```

The FORTRAN instruction reads a number from the input device assigned to channel 1. The 100 refers to the format which specifies the form of the number and its precise location, on, for example, a data card if the input device is a card reader. The BASIC instruction is for interactive input from a keyboard and is obviously much easier to use. The Algol instruction is from ICL Algol and is probably unique to that implementation. The one language feature not defined in the Algol 60 report was input/output, so that the form of these instructions is left to the discretion of the language implementer and almost certainly varies from implementation to implementation. Output instructions in these languages resemble input instructions and illustrations are given below.

ii) *Conditionals*

The forms of the conditional instructions in the respective languages are illustrated by:

```
IF (A.GT.2.5) S = A + B . . .
FORTRAN
IF A > 2.5 THEN S = A + B
. . . BASIC
if a > 2.5 then s: = a + b else
s: = a – b . . . Algol
```

The instructions all have the general form:
'IF condition THEN conditional instruction', but with variations. During program execution, the conditional instruction is executed only when the condition is true. The FORTRAN instruction is clumsy and the use of the same symbol as a separator and a decimal point is confusing. The BASIC form is more readable and the inclusion of THEN is very helpful. However, the Algol instruction with the 'else' at the end is much more powerful.

iii) *Repetition*

One of the strengths of the computer is the ease with which it can perform actions repetitively. For this reason languages must contain features that facilitate the handling of repetition. The facilities for repetition in the various languages are illustrated by short programs to print out the integers from 1 to 20. These programs also illustrate the output instructions.

```
      DO 50 I = 1, 20
      WRITE (6,101) I
101   FORMAT (I3)
50    CONTINUE          Fortran
```

```
      FOR I = 1 TO 20
      PRINT I
      NEXT I            Basic
```

```
      for i: = 1 step 1 until 20 do
      begin   print (i, 2, 0) end     Algol
```

The shorthand for repetition in FORTRAN and BASIC is similar, although the need in FORTRAN to identify the end of the loop with a number can be irksome.

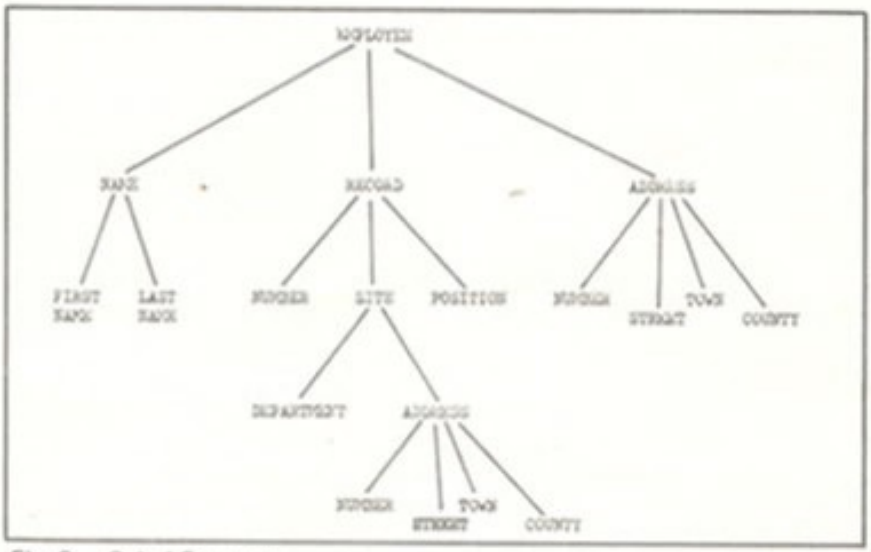


Fig. 3. Cobol Structure.

iv) *Data structures*
Arrays are almost the only facilities provided for structuring data in the early high-level scientific languages. These structures are well suited for handling vectors and matrices. The instructions in the respective languages for reserving storage space for a one-dimensional array are:-

DIMENSION V(10) (FORTRAN)
DIM V(10) (BASIC)
real array v [1:10] (Algol)

The FORTRAN instruction reserves ten locations for variables named V(1) to V(10), while, in most dialects, the BASIC instruction reserves eleven locations for V(O) to V(10). The Algol instruction is explicit.

COBOL, which is intended for use in commercial applications, has instructions of a different appearance from those of the scientific languages. Only simple arithmetic facilities are necessary, the major facilities are provided for reading and updating file records and filling in forms. The language is intended to be readable. The COBOL instruction:

MOVE X TO Y

is equivalent to the FORTRAN instruction Y = X, while

ADD BALANCE TO OLDTOTAL GIVING NEWTOTAL is equivalent to NEWTOTAL – OLDTOTAL+ BALANCE. COBOL has more sophisticated data structures than the scientific languages and can support structures such as the one illustrated in Figure 3. Selection of an item in such a structure is achieved by: LAST NAME IN NAME IN EMPLOYEE

Arrays of such structures can also be used, for example

EMPLOYEE (1:25).

Structured Programming

The next stage in language development was the emergence of languages to permit structured programming. The need for such languages emerged because programmers were experiencing intense difficulty in the development of long programs. The difficulties arose largely because language facilities were not available to permit the management of a natural flow of control in large programs, so that programmers were forced to use clumsy and unsuitable constructs.

The fundamental concepts of structured programming follow from the idea that the structure of a program should reflect the structure of the problem solution method. If programs can be written in this way they are automatically readable. Thus there is no real need to add comments to a program to make it readable. More importantly, programs are made easier to debug because the logical flow of the solution method is preserved directly by the program itself. Properly structured programs are also easier to develop and test because, as illustrated in Figure 4, they can be divided into modules, each of which has only one entry and exit point, that can be developed and tested independently.

The shortcomings of the early high-level languages arise from the fact that they do not provide features to make possible the writing of large, properly structured programs. The languages that are intended for structured programming include PASCAL and Algol 68. PL/1 also has facilities for structured programming, although it was conceived as a general purpose language, combining the features of both FORTRAN and COBOL while showing the influence of Algol 60. PASCAL was intended as a teaching language that would demonstrate programming as a systematic discipline and which could be efficiently implemented in a compact manner. The first operations compiler became available in 1970. Algol 68 was defined by a formal report dated 1968; it is a general purpose language developed from and updating Algol 60.

It is rather difficult to properly illustrate structured programming without examining long programs. However, the examples and programs discussed in the remainder of this section might give some flavour to it. The BASIC program listed as Program 3 can be said to be a very bad one!

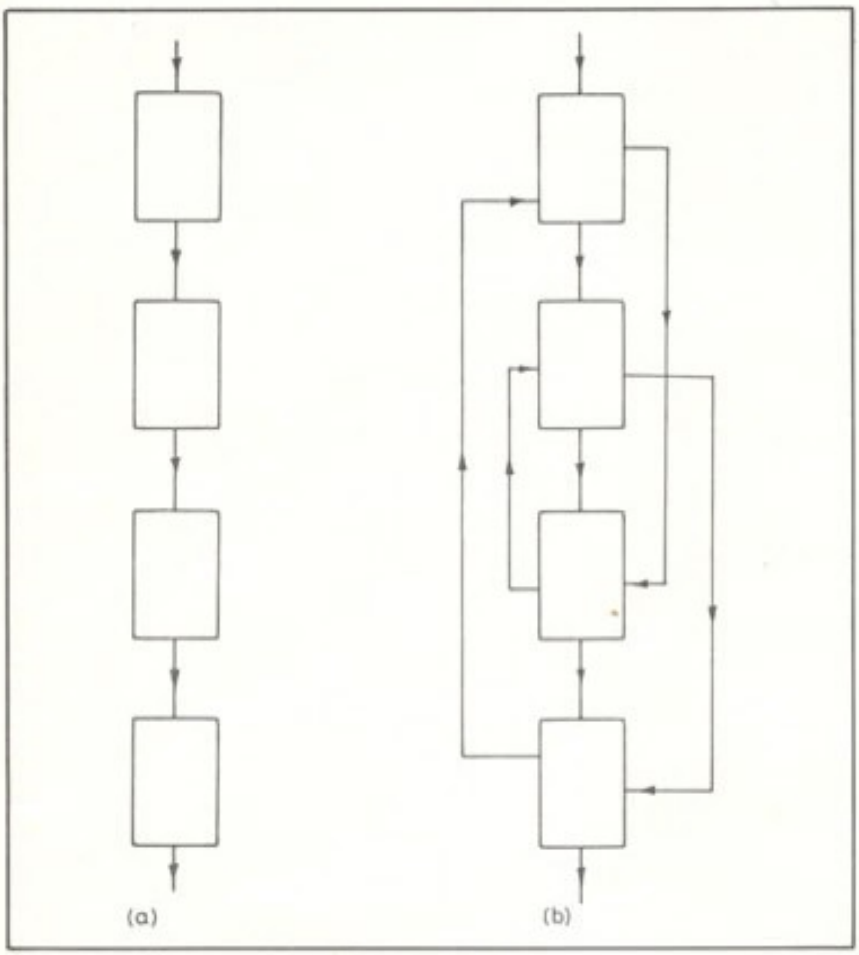


Fig. 4. Good (a) and bad (b) program structures.

```
10 A = 1
20 B = 2
30 GOTO 60
40 PRINT C
50 STOP
60 C = A + B
70 GOTO 40
```

Program 3. A 'spaghetti' program.

A short examination shows that the same computational process is described by Program 4. Program 3 is a 'spaghetti' program that can be reduced to a linear program as in Program 4. (Imagine holding the ends of a piece of spaghetti and pulling until it untangles!)

```
10 A = 1
20 B = 2
30 C = A + B
40 PRINT C
50 STOP
```

Program 4. Linear program equivalent to Program 3.

Now, when programming in one of the early high-level languages it is not always possible to write a linear program to express a linear solution method. Typically, the GOTO construct has to be used in a manner that destroys the linearity. (This is equivalent to pulling the ends of the spaghetti and finding that a knot has been produced.)

To illustrate how programs in an appropriate language can reflect a solution method better than another language, we consider two problems and present programs for their solution in PASCAL and in BASIC. The first problem is:–

Read numbers from data until the first negative number is encountered. Then print out its position and stop.

The natural solution method is to repeatedly read numbers and count them until a negative number is read, and then to print out the value of the counter before stopping. This method can be expressed directly in PASCAL as in Program 5.

```
i: = 0
repeat read (n); i: = i + 1 until n <0
writeln (i)
```

Program 5. Pascal program for problem 1.

A BASIC program to do the same thing is given as Program 6 and is a little contrived.

Now consider a second problem which concerns the calculation of rail fares:-

Given the full fare, length of stay and age of the passenger, compute the actual fare to be paid.

```
10 I = 0
20 INPUT N
30 I = I + 1
40 IF N <0 GOTO 60
50 GOTO 20
60 PRINT I
```

Program 6. Basic program for problem 1.

Reductions on the full rail fare for a journey are given according to both the length of stay and the age of the passenger as shown in the following tables:-

Length of stay (days)	Discount on full fare
1	50%
2 – 14	30%
15 or more	0

Age (years)	Actual Fare
under 3	free
3 – 13	half discounted fare
over 14	discounted fare

A PASCAL program for this which captures the natural solution method is given as Program 7. The variables ff, af and fare are used to hold the values of the full fare, adult fare and the fare to be paid, respectively. It is left as an exercise for the reader to attempt the same thing in BASIC or FORTRAN.

```
read      ff      ,      stay      age     ;
if stay           = 1    then      af:=    0.5*ff
                         else if stay      <15 then af: = 0.7*ff
                                               else af: = ff
if age            <3     then fare: =      0
                         else if age       <14 then fare: = 0.5*af
                                               else fare: = af
writeln (fare);
```

Program 7. Pascal program for problem 2.
(is the multiplication symbol)*

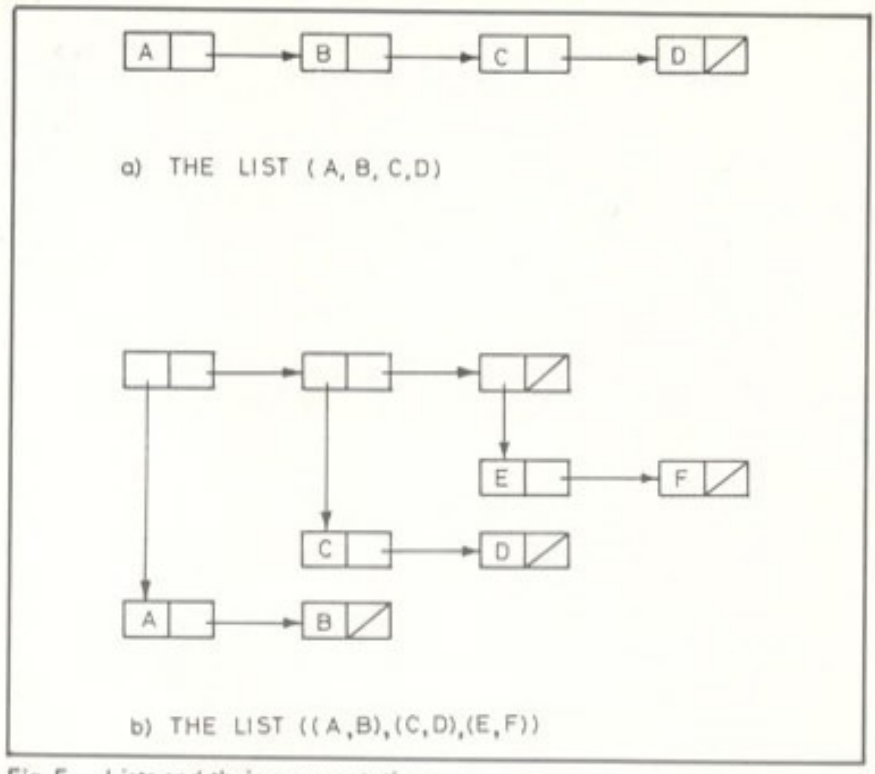


Fig. 5. Lists and their representations.

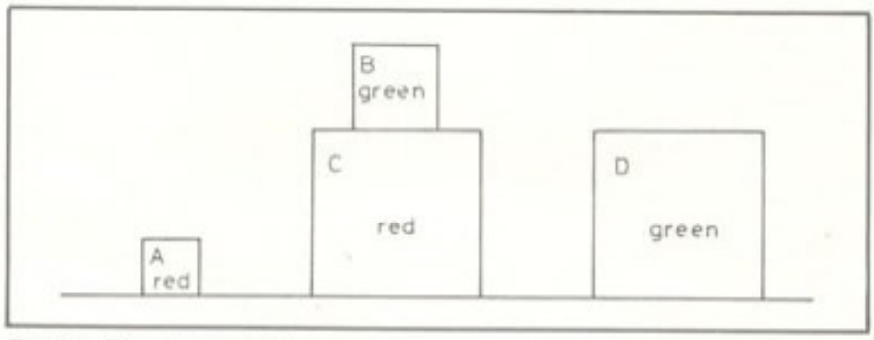


Fig. 6. The blocks world.

Special Purpose Languages

All the languages mentioned to this point are intended for either scientific or commercial application. However, other languages have been developed for particular special purposes. These include languages designed for manipulating particular data structures, most notably for manipulating strings and for processing lists. Snobol is the major language designed principally for string manipulation. It has facilities that include pattern matching and pattern replacement. A typical Snobol instruction is

STR "ED" :S (FOUND)

The effect of this instruction is to scan the string named STR for the pattern ED and if the search is successful to jump to the instruction labelled FOUND. The original string manipulation language is LISP. A typical LISP instruction is

((EQ (CAR LAT) A) (CDR LAT))

The applications of string manipulation and list processing include linguistic analysis, compiler writing, computer assisted instruction and artificial intelligence.

Two examples may show, in a limited way, how list processing can

be used, but first the meaning of a list must be explained.

A list of the four elements A, B, C and D can be represented by (A, B, C, D) and the way in which it is stored is represented in Figure 5a. The figure shows that each element of the list is stored with a pointer to the next element, and this is the way that the elements of a list are linked. A more complex list is represented in Figure 5b. The first example is drawn from a system developed by Terry Winograd for the machine comprehension of natural language.

The system manipulates a representation of a set of coloured blocks, such as the one illustrated in Figure 6. The system can understand and obey instructions expressed in natural language such as "put the small red block on the large green block". We consider only how the machine can determine what is meant by "the small red block". Given a list of all the blocks, (A, B, C, D), a list of the red blocks, (A, C), a list of the green blocks, (B, D), a list of the small blocks, (A, B), and a list of the large blocks, (C, D), then the small red block must be a block that is in both the list of small blocks and the list of red blocks. Thus, the small red block is block A.

It is worth noting that operations on sets are being implemented by using lists, for if the list of blocks is regarded as a description of a set, the lists of blocks with a particular property are subsets and the problem of determining what is meant by the small red block becomes that of finding the intersection of two subsets.

The next example illustrates how lists and list processing can be of value in enabling a robot to automatically determine the actions necessary for it to achieve a specified objective. Consider a robot in the rather simple situation illustrated in Figure 7 which is set the task of filling the bucket. The situation in the diagram can be represented by a list (known as the world list) thus:-

((ROBOT, IN, ROOM 2) (TAP, IN, ROOM 2) (BUCKET, IN, ROOM 1) (BUCKET, EMPTY))

The situation on successful completion of the task is represented by this world list:—

((ROBOT, IN, ROOM 2) (TAP, IN, ROOM 2) (BUCKET, IN, ROOM 2) (BUCKET, FULL))

Now, each action that the robot can take alters the situation in some way and can be represented by a corresponding change in the world list.

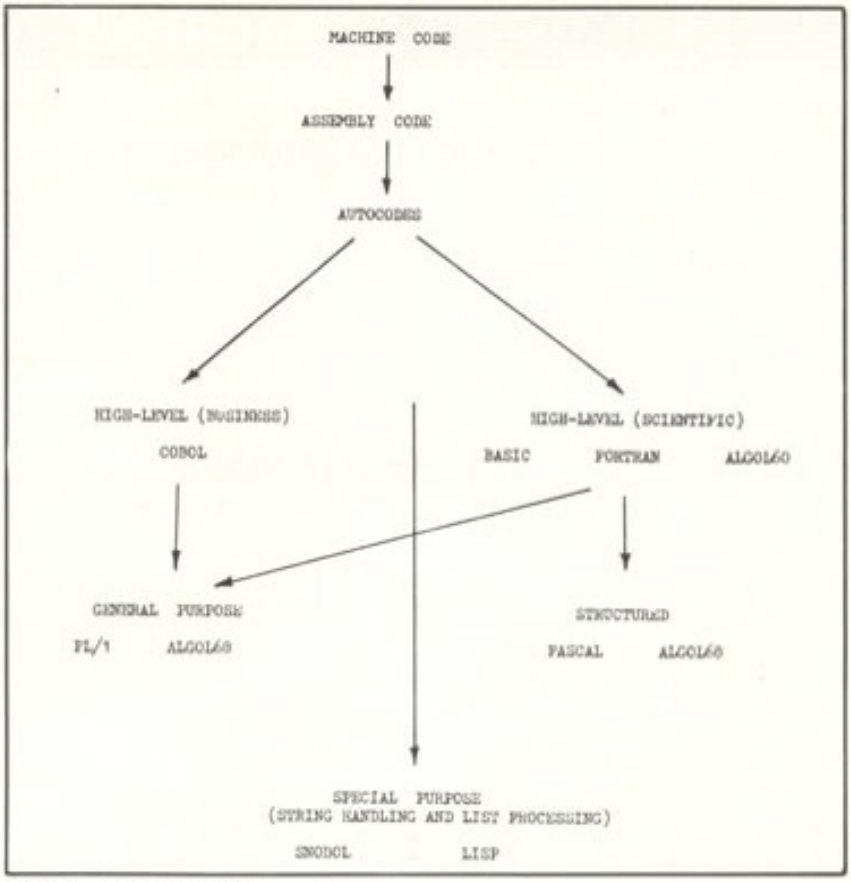


Fig. 8. Language tree.

To achieve its task, the robot must determine an appropriate sequence of actions. This can be done by determining a valid sequence of changes to the initial world list which alters it to the world list representing the situation on completion of the task.

Impact on Personal Computing

Figure 8 shows, in the form of a tree, the lines of development connecting the languages mentioned in this article. With regard to the languages that have made an impact on personal computing, assembly codes are not really in the running, since an assembly code is specific to a particular micro-processor making it difficult for one to achieve any general impact. The language that has made the greatest impact to date is BASIC, and the most likely language to achieve general acceptance in the near future is PASCAL.

It is interesting that both were designed as teaching languages. Their general acceptance probably stems from the fact that their principal aims fit the requirements of personal computing almost perfectly. In particular, BASIC is easy to learn and PASCAL is structured and capable of compact implementation. For both languages, interpreters seem to be more suitable than compilers for personal computing because the ability to interact with a program seems, in general, to be more desirable than attaining the shortest possible times for program execution.

As for the future, the possibility of natural language input, at least in a restricted form, has been touched on. Since a speech recognition unit is currently available for the TRS80, perhaps programming will soon be possible by addressing the computer in almost ordinary English.

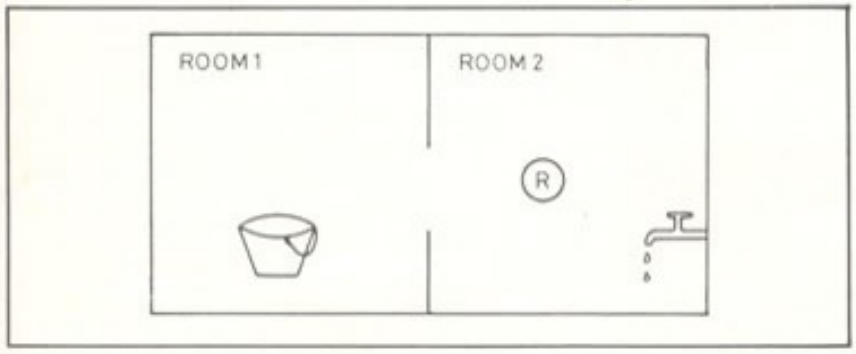


Fig. 7. A robot world.

BIBLIOGRAPHY

1. 'An introduction to the study of programming languages', D. W. Barron, Cambridge University Press, 1977. 2. 'A comparative study of programming languages', B. Higman, Macdonald and Jane's, 1967. Both books give more details of the languages mentioned in this article and of many other besides. 3. 'Pascal user manual and report', K. Jensen and N. Wirth, Springer-Verlag, 1975. The PASCAL report and a tutorial style user manual. 4. 'Illustrating *Basic', D. Alcock, Cambridge University Press, 1977. An entertaining introduction to BASIC. 5. 'List processing', J. M. Foster, Macdonald and Jane's 1967. A nice introduction to list processing and its applications. 6. 'Artificial Intelligence and Natural Man', Margaret Boden, Harvester Press, 1977. A description of artificial intelligence work that stresses its human relevance and includes an account of Winograd's blocks program.

Submitting programs to APC

Having written and thoroughly tested your original program (be it an application, game, or useful subroutine) send it to us along with a suitable explanation. In order of preference we would like your programs submitted as a clear, dark listing on plain paper; on cassette or disc; clearly, accurately typed; or, clearly, accurately handwritten.

We pay the sender of any listing published at least $20, and often much more, depending on the size and quality of the contribution. If the program is too large or complex for the "Programs" section we will sometimes publish it as a feature in the magazine.

For the sake of balance, how about some of the owners of less popular machines pitching in as well? Post your submissions to APC Programs, PO Box 115, Carlton, Victoria, 3053. We look forward to hearing from you.

Efficient character storage

by David Tucker

In computing it is very often desirable to pack as many characters as possible into a given area of memory.

The usual method of holding text is to encode the text into 8 bit ASCII and store one character per byte. Thus there are 256 possible different ASCII codes that can be stored in any particular byte.

Very often however the only characters of interest are A to Z, 0 to 9, space and perhaps a couple of punctuation marks.

It so happens that if we take a set of 40 characters we can encode them in such a way that three characters can be stored in 16 bits (2 bytes), thus increasing the storage capacity of a given area of memory by 30%.

We do this by treating the three characters as if they were digits of a number to the base 40.

Suppose a=1, b=2, c=3

Then ABC = 3x40x40+2x40+1 = 4881

The number 4881 could be stored in a 16 bit word.

The maximum value would be 39x40x40+39x40+39 = 63999.

This is less than 65535 (the maximum 16 bit number).

The two subroutines presented here are written in Z80 assembler code although the principles outlined can be applied to any machine.

The ENCODE subroutine takes a three byte ASCII string whose address is in the BC register pair and returns the encoded RADIX-40 characters in the HL register pair.

The DECODE subroutine takes the three RADIX-40 characters in the HL register pair and returns a three byte ASCII string starting at the address contained in BC.

How it works:

The ENCODE subroutine first clears the HL register pair then calls the subroutine FIND which locates the character pointed to by BC and returns its number from the end of the TABLE in register A. BC is then incremented to point to the next character.

DE is loaded with 1600 (40x40) and the subroutine MULT is then called. This subroutine multiplies the contents of register A and adds the result to register pair HL.

The process is repeated for the other two characters with DE set to 40 then 1.

The three characters have now been encoded.

```
0D00                0010          ORG  &0D00
                    0020 ;BC POINTS TO A
                    0030 ; 3 CHARACTER STRING
                    0040 ; ON RETURN
                    0050 ; HL CONTAINS RADIX-40
0D00 210000         0060 ENCODE LD   HL,0 ;CLEAR HL
0D03 CD1F0D         0070        CALL FIND ;1ST CHARACTER
0D06 114006         0080        LD   DE,1600 ;40*40
0D09 CD2F0D         0090        CALL MULT ;DE*A
0D0C CD1F0D         0100        CALL FIND ; 2ND CHARACTER
0D0F 112800         0110        LD   DE,40
0D12 CD2F0D         0120        CALL MULT
0D15 CD1F0D         0130        CALL FIND ; 3RD CHARACTER
0D18 110100         0140        LD   DE,1
0D1B CD2F0D         0150        CALL MULT
0D1E C9             0160        RET  ;FINISH
                    0170 ;
0D1F 0A             0180 FIND   LD   A,(BC) ;GET CHARACTER
0D20 03             0190        INC  BC ;POINT TO NEXT
0D21 C5             0200        PUSH BC ;SAVE
0D22 E5             0210        PUSH HL ; ON STACK
0D23 012800         0220        LD   BC,40 ;TABLE LENGTH
0D26 215F0D         0230        LD   HL,TABLE
0D29 EDB1           0240        CPIR ;FIND CHARACTER
0D2B 79             0250        LD   A,C ;GET NUMBER
0D2C E1             0260        POP  HL
0D2D C1             0270        POP  BC
0D2E C9             0280        RET
                    0290 ;
0D2F A7             0300 MULT   AND  A ;SET CONDITION
                    0310 ;  CODES
0D30 C8             0320        RET  Z ; FINISH IF 0
0D31 19             0330        ADD  HL,DE
0D32 3D             0340        DEC  A ;DEC COUNT
0D33 18FA           0350        JR   MULT ;ROUND WE GO
                    0360 ;HL CONTAINS RADIX-40
                    0370 ; BC POINTS TO RESULT BUFFER
0D35 114006         0400 DECODE LD   DE,1600 ;40*40
0D38 CD4B0D         0410        CALL TRANS ;1ST CHARACTER
0D3B 112800         0420        LD   DE,40
0D3E CD4B0D         0430        CALL TRANS ;2ND
0D41 110100         0440        LD   DE,1
0D44 CD4B0D         0450        CALL TRANS ;3RD
0D47 C9             0460        RET  ;FINISH
                    0470
                    0480
0D48 AF             0490 TRANS  XOR  A ;CLR A
0D49 ED52           0500 TRANS1 SBC  HL,DE ;SUBTRACT
0D4B 3803           0510        JR   C DONE ;JUMP IF OVERFLOW
0D4D 3C             0520        INC  A ;INC COUNTER
0D4E 18F9           0530        JR   TRANS1 ;ROUND WE GO
0D50 19             0540 DONE   ADD  HL,DE ;RECOVER VALUE
0D51 E5             0550        PUSH HL
0D52 21870D         0560        LD   HL,TABEND ;END OF TABLE
0D55 5F             0570        LD   E,A ;GET A IN DE
0D56 1600           0580        LD   D,0
0D58 ED52           0590        SBC  HL,DE ;GET ADDRESS
0D5A 7E             0600        LD   A,(HL): GET CHARACTER
0D5B 02             0610        LD   (BC),A ;STORE
0D5C 03             0620        INC  BC ;POINT TO NEXT POSITION
0D5D E1             0630        POP  HL
0D5E C9             0640        RET
```

The DECODE subroutine performs the inverse operation to ENCODE.

The register pair DE is first loaded with 1600 then the subroutine TRANS is called. This subroutine performs an integer division of HL by DE leaving the result in register A and the remainder in HL. It then locates the relevant character in the TABLE and transfers it to the location pointed at by BC and finally increments BC. The whole process is repeated with DE set to 40 then 1.

```
                        0650 ;
                        0660 ;
0D5F 20414243 0670 TABLE  DEFM / ABC/
0D63 44454647 0680        DEFM /DEFG/
0D67 48494A4B 0690        DEFM /HIJK/
0D6B 4C4D4E4F 0700        DEFM /LMNO/
0D6F 50515253 0710        DEFM /PQRS/
0D73 54555657 0720        DEFM /TUVW/
0D77 58595A31 0730        DEFM /XYZ1/
0D7B 32333435 0740        DEFM /2345/
0D7F 36373839 0750        DEFM /6789/
0D83 302E2C3F 0760        DEFM /0.,?/
0D87 (X)      0770 TABEND NOP
```

Challenge from Space

by Peter Dillon.

He writes: "The following program is written in Level II BASIC and will fit comfortably into 4K of memory if the REM statements are removed.

The game involves alien craft bombarding the surface of a planet; and in retaliation, the home base launching missiles as the three rows of aliens move across the screen. The base is protected by a series of barriers. However, these are slowly eroded during the attack by the alien missiles.

As the aliens move closer to the planet surface, so their missiles become more difficult to evade. (Hint: try to destroy the lowest row of aliens first).

Three controls are provided for movement of the base: key 4 and key 6 move it to the left and right respectively, and key 5 halts any movement and simultaneously launches a missile. Any number of missiles may be fired against the aliens, although limitations could easily be placed on this by introducing a missile counter into line 240 which gives a value of 1 to the variable M, indicating that a missile is in flight.

There are five home bases, although again, this can be altered. Line 410 increases the value of the variable B by 1 each time a base is destroyed and checks to see if B equals five.

I suggest you increase the amount of string space cleared in line 20 to at least 1500 bytes if you have a 16K machine as this results in faster execution of the subroutine to remove aliens.

There is a short delay immediately after you type RUN during which strings for barriers, aliens, missiles etc. are defined. At the end of the game, pressing the letter B will re-run the program."

```
10 REM CHALLENGE FROM SPACE
20 CLS:CLEAR900:DEFSTR A:DEFINT D,N,P,X:PC=15360:PG=926:HM=701:NR=3
30 PRINTCHR$(23):PRINT@270,"CHALLENGE FROM":PRINT@534,"SPACE";
40 REM DEFINES GUNS, MISSILES AND DESTRUCTION CHARACTERS
50 BD$=CHR$(191):AM=CHR$(91):AG="  "+CHR$(184)+CHR$(191)+CHR$(180)+"  ":AN=CHR$(
92):FORX1=1TO7:AD=AD+CHR$(153):NEXT
60 REM DEFINES CHALLENGERS
70 FORX1=1TO3:FORX2=1TO5:A(X1,X2)=CHR$(152)+CHR$(172)+CHR$(156)+CHR$(164):NEXTX2
,X1
80 FORX1=1TO3:FORX2=1TO5:A(X1,0)=A(X1,0)+"  "+A(X1,X2)+"     ":NEXTX2,X1
90 REM DEFINES LANDSCAPE
100 FORX1=1TO63:AL=AL+CHR$(RND(62)+129):NEXT
110 REM DEFINES RND OBSTACLES
120 FORX1=1TO2:FORX2=0TO59 STEP10
130 FORX3=0TO4:AB(X1)=AB(X1)+CHR$(RND(63)+128):NEXTX3
140 AB(X1)=AB(X1)+"      ":NEXTX2,X1
150 REM DRAWS LANDSCAPE AND OBSTACLES
160 CLS:PRINT@768,AB(1);:PRINT@832,AB(2);:PRINT@960,AL;
170 PRINT@36,"STRENGTH OF ALIEN FORCE  15";
180 REM
190 REM ACTIVE PART OF PROGRAM BEGINS HERE
200 REM
210 FORPI=258TO650STEP2:FORX6=1TO3
220 A=INKEY$:IFA()"" THEN S=2*(VAL(A)-5)
230 IF A()"5" THEN IF M=1 THEN250  ELSE270
240 PRINT@PM," ";:PM=PG-125:M=1:PRINT@PM,AM;:GOTO270
250 IFPM(PITHENM=0:PRINT@PM," "; ELSEPM=PM-64:IFPEEK (PM+PC)()32 OR PEEK(PM+PC+64
)=92 THEN M=0: GOSUB330 ELSEPRINT@PM,AM;
260 PRINT@PM+64," ";
270 IFN()1 THEN PN=64*INT(PI/64)-HM+PG:N=1:DM=(PG-PN)/32:PRINT@PN-DM,AN;:PRINT@P
N,AN;:PRINT@PN+DM,AN;:GOTO290
280 PRINT@PN-DM," ";:PRINT@PN," ";:PRINT@PN+DM," ";:PN=PN+64:IFPN(757THENPRINT@P
N-DM,AN;:PRINT@PN,AN;:PRINT@PN+DM,AN; ELSEIFPN)895ORPEEK(PN-DM+PC)()32OR PEEK(PN
+PC)()32ORPEEK(PN+DM+PC)()32THENGOSUB400ELSEPRINT@PN-DM,AN;:PRINT@PN,AN;:
PRINT@PN+DM,AN;
290 PG=PG+S:IF PG(896 OR PG)954 THENPG=PG-S ELSEPRINT@PG,AG;
300 PRINT@PI+64*(X6-1),A(X6,0);
310 NEXTX6,PI
320 REM SUBROUTINE TO REMOVE INVADERS
330 IFPEEK(PM+PC)=92 OR PEEK(PM+PC+64)=92 THEN M=0:N=0:PRINT@PN," ";:PRINT@PN-DM
," ";:PRINT@PN+DM," ";:RETURN
340 PRINT@PM-2,AD;:RA=INT((PM-PI)/64)+1:PA=INT((PM-(RA-1)*64-PI)/11)+1:IFRA)3 TH
EN370 ELSEA(RA,PA)="     ":NI=NI+1:PRINT@60,15-NI;:IF NI=15 THEN380
350 A(RA,0)="  "+A(RA,1)+"       "+A(RA,2)+"       "+A(RA,3)+"       "+A(RA,4)+"
      "+A(RA,5)+"     "
360 IF A(NR,0)=STRING$(55," ")THEN HM=HM+64:NR=NR-1
370 PRINT@PM-2,"          ";:RETURN
380 CLS:PRINTCHR$(23):PRINT@324,"INVASION FORCE DESTROYED";:PRINT@456,"YOU ARE V
ICTORIOUS !!!";:GOTO440
390 REM ROUTINE TO CHECK IF BASE HAS BEEN DESTROYED
400 IF(PN)=PG+2ANDPN(PG+5) OR (PN-DM)=PG+2ANDPN-DM(PG+5) OR (PN+DM)=PG+2ANDPN+DM
(PG+5) THENPRINT@PG-1,AD;:GOTO410  ELSEN=0:PRINT@PN-DM,BD$;:PRINT@PN,BD$;:PRINT@
PN+DM,BD$;:PRINT@PN-DM,"  ";:PRINT@PN,"  ";:PRINT@PN+DM,"  ";:RETURN
410 N=0:PRINT@PG-1,"          ";:B=B+1:PRINT@66,B;:IFB=1 THENPRINT"BASE HAS BEEN D
ESTROYED...FOUR REMAIN FUNCTIONAL.";ELSE IFB(5 THENPRINT"BASES HAVE BEEN DESTROY
ED.";ELSE430
420 FORX1=1TO800:NEXT:PRINT@66,STRING$(52," ");:RETURN
430 CLS:PRINTCHR$(23):PRINT@260,"INVASION COMPLETE";:PRINT@390,"ALL DEFENCE BASE
S DESTROYED.";
440 ZZ$=INKEY$:IFZZ$="B" THEN20  ELSE440
```

Fox and Hounds for TRS80

L. J. Aston

This program generates a very nice draught board, upon which are placed four hounds and one fox. The object is to corner the fox. Full instructions are contained in the listing. It is fairly easy to corner the fox with four hounds, difficult with three and, I think, impossible with two. The program requires 6K of memory and runs on a TRS80 level II.

```
10 CLS:PRINT:PRINT:PRINT:PRINTCHR$(23)
20 PRINT"        FOX AND HOUNDS
30 PRINT:PRINT"     LIKE INSTRUCTIONS(Y/N)":GOTO40
40 A$=INKEY$:IFA$=""THEN 40 ELSEIFA$="Y"GOSUB 800
50 DIML(9,9)
60 Z=0:CLS:PRINTCHR$(23)
70 FORC=1TO8STEP2
80 L(1,C)=1:L(1,C+1)=2:NEXTC
90 FORC=1TO8STEP2
100 L(2,C)=4:L(2,C+1)=1
110 L(3,C+1)=4:L(3,C)=1
120 L(4,C)=4:L(4,C+1)=1
130 L(5,C)=1:L(5,C+1)=4
140 L(6,C)=4:L(6,C+1)=1
150 L(7,C)=1:L(7,C+1)=4
160 L(8,C)=4:L(8,C+1)=1:NEXTC
170 L(8,5)=3:X=8:Y=5:A=X:B=Y
180 GOSUB 820
190 PRINT@14,"C D L U M N
200 PRINT"      1 2 3 4 5 6 7 8"
210 PRINT"    "+STRING$(16,176)
220 FORL=1TO8:PRINT"R"L;CHR$(178);
230 FORC=1TO8
240 IFL(L,C)=1PRINTCHR$(191)+CHR$(191);
250 IFL(L,C)=2PRINT"H ";
260 IFL(L,C)=3PRINT"F ";
270 IFL(L,C)=4PRINT"  ";
280 NEXTC
290 PRINTCHR$(149)+" R"L:NEXTL
300 PRINT"    "+STRING$(16,131)
310 PRINT"      1 2 3 4 5 6 7 8
320 IFX=1THEN790
330 IFZ=1THEN770
340 PRINT@832,"  YOUR MOVE   -  ROW? COLUMN?
350 GOSUB1110:FORI=1TO200:NEXT
360 PRINT"  -TO ?";:GOSUB1150 :FOR I=1TO200:NEXT
370 IFT(FPRINT@960,"YOU CANNOT MOVE BACKWARDS";
380 IFT(FGOTO340
390 IFY-F()1GOTO 420
400 IFL(F,F1)()2GOTO 420
410 IFL(T,T1)=4ANDL(F,F1)=2GOTO440
420 PRINT@960,"INVALID MOVE"+CHR$(31);
430 GOTO340
440 L(T,T1)=2:L(F,F1)=4:D=0
450 K=INT(RND(0)*2)
460 IFK)2THEN 450
470 IF K =1 THEN 490
480 IFK=0THEN550
490 D=D+1:X=A-1:Y=B-1
500 IFL(X,Y)=4THENL(X,Y)=3
510 IFL(X,Y)()3ANDD()2THEN550
520 IFL(X,Y)()3ANDD=2GOTO 610
530 IFL(X,Y)=3THENL(X+1,Y+1)=4
540 IFL(X,Y)=3THEN180
550 X=A-1:Y=B+1:D=D+1
560 IFL(X,Y)=4THENL(X,Y)=3
570 IFL(X,Y)()3ANDD(2THEN490
580 IFL(X,Y)()3ANDD=2THEN610
590 IFL(X,Y)=3THENL(X+1,Y-1)=4
600 IFL(X,Y)=3THEN180
610 IFK=0THEN630
620 IF K=1THEN700
630 D=D+1:X=A+1:Y=B+1
640 IFL(X,Y)=4THENL(X,Y)=3
650 IFL(X,Y)()3ANDD(4THEN700
660 IFL(X,Y)=3THENL(X-1,Y-1)=4
670 IFL(X,Y)=3THEN180
680 IFL(X,Y)()3THENZ=1
690 GOTO180
700 Y=B-1:X=A+1:D=D+1
710 IFL(X,Y)=4THENL(X,Y)=3
720 IFL(X,Y)()3ANDD(4THEN630
730 IFL(X,Y)=3THENL(X-1,Y+1)=4
740 IFL(X,Y)=3THEN180
750 IFL(X,Y)()3THENZ=1
760 GOTO180
770 PRINT"YOU WIN-LIKE ANOTHER GAME(Y/N)";
780 GOTO800
790 PRINT"I WIN-LIKE TO TRY AGAIN(Y/N)":GOTO800
800 A$=INKEY$:IFA$=""THEN 800 ELSEIFA$="Y"THEN52
810 END
820 IFA=XTHEN870
830 IFZ=1CLS:PRINTCHR$(23)"                    I CAN'T MOVE
840 IFZ=1THEN870
850 CLS:PRINTCHR$(23):PRINT@960,"I MOVED FROM "1A1","1
B"TO "1X1","1Y1
860 A=X:B=Y:FORI=1TO920:NEXT
870 RETURN
880 CLS:PRINTCHR$(23):PRINT:PRINT"YOU ARE THE HOUNDS
890 PRINT"YOU HAVE FOUR PLAYERS(H)
900 PRINT"YOU MAY ONLY MOVE FORWARD.
910 PRINT:PRINT"THE COMPUTER IS THE FOX(F)
920 PRINT"AND MOVES FORWARDS OR BACKWARDS"
930 PRINT:PRINT"IF YOU TRAP THE FOX YOU WIN
940 PRINT"IF THE FOX GETS TO ROW 1
950 PRINT"THE COMPUTER WINS":PRINT
960 INPUT"PRESS ENTER TO  CONTINUE";A$
970 CLS:PRINTCHR$(23)
980 PRINT"WHEN IT IS YOUR TURN.
990 PRINT"TYPE IN THE ROW NUMBER
1000 PRINT"AND COLUMN NUMBER OF
1010 PRINT"THE MAN TO BE MOVED
1020 PRINT:PRINT"THEN WHEN  ASKED '-TO?'
1030 PRINT"TYPE IN ROW NUMBER AND
1040 PRINT"COLUMN NUMBER TO WHICH
1050 PRINT"YOU ARE MOVING.
1060 PRINT"ONLY THE BLACK SQUARES
1070 PRINT"MAY BE USED.
1080 PRINT"   GOOD LUCK !!!
1090 PRINT:INPUT"PRESS ENTER TO CONTINUE";A$
1100 CLS:PRINTCHR$(23):RETURN
1110 A$=INKEY$:IFA$=""THEN1110
1120 F=VAL(A$):PRINTF1;",";
1130 B$=INKEY$:IFB$=""THEN1130
1140 F1=VAL(B$):PRINTF1;:RETURN
1150 A$=INKEY$:IFA$=""THEN1150
1160 T=VAL(A$):PRINTT;",";
1170 B$=INKEY$:IFB$=""THEN1170
1180 T1=VAL(B$):PRINTT1;:RETURN
```

PETs and tanks

Kevin Jones (13) has, with the aid of his father, produced a Tank Battle simulation for the PET. He tells us that it will work in a 4K PET if all the REMs are removed and it will work in both new and old ROM PETS. The game takes place across a minefield, with the additional hazard of walls to negotiate. The game is for two players, each equipped with a tank, and the first to score ten points wins. A point is scored by hitting your opponent's tank with a missile.Each player has 9 controls arranged in a 3 by 3 square. Though the game was written for a PET it should be fairly easy to adapt. The PET's instruction POKE 32768+40*Y+X,Z is equivalent to PLOT X,Y,Z on other machines.

```
5 REM
15 REM *** TANKS ***
25 REM KJR JONES 25/10/79 ***
35 REM
40 PRINT"■TANKS" PRINT
50 PRINT"THE OBJECT OF THE GAME IS TO SCORE"
60 PRINT "TEN POINTS"
70 PRINT "YOU MAY SCORE A POINT IN TWO WAYS. BY"
80 PRINT "SHOOTING YOUR OPPONENT'S TANK OR IF YOU"
90 PRINT "STEP ON A MINE ●. A POINT IS SCORED FOR"
100 PRINT "HIM. A MISSILE WILL FLY OVER A MINE"
110 PRINT "AND DISINTEGRATE ON HITTING AN OUTER"
120 PRINT "WALL. IF IT HITS A BARRIER INSIDE THE"
130 PRINT "BOARD AREA. IT WILL HALF DESTROY IT. OR"
140 PRINT "THE NEXT HIT IT WILL RUIN IT TOTALLY."
160 PRINT "EACH PLAYER HAS NINE CONTROLS AS SHOWN"
170 PRINT "   E R T                7 8 9"
180 PRINT "   D F G                4 5 6"
190 PRINT "   C V B                1 2 3"
200 PRINT "TO MOVE 1 SQUARE IN ANY DIRECTION PRESS"
210 PRINT "THE KEY IN THAT DIRECTION FROM YOUR"
220 PRINT "CENTRE KEY. THE CENTRE BUTTON ITSELF"
230 PRINT "FIRES A MISSILE IN THE DIRECTION OF"
240 PRINT "YOUR LAST MOVE."
250 PRINT "THE LEFT TANK IS SHOWN AS ●."
260 PRINT "THE RIGHT TANK IS SHOWN AS O."
270 PRINT "PRESS ANY KEY TO START " A$=""
280 GET A$ IF A$="" THEN 280
290 GOSUB 900 PRINT"■"
300 FOR X=0 TO 39
310 POKE 32768+X,227 POKE 33568+X,228
320 NEXT X
330 FOR Y=1 TO 19
340 POKE 32768+40*Y,229 POKE 32807+40*Y,231
350 NEXT Y
360 FOR Y=1 TO 19
370 FOR X=1 TO 38
380 R=RND(1)*10
390 IF R>9.3 AND R<9.75 THEN POKE 32768+40*Y+X,160
400 IF R>9.75 THEN POKE 32768+40*Y+X,90
410 NEXT X:NEXT Y
420 LV=10:LH=3:RV=10:RH=36
430 M=32
440 POKE 33171,81 POKE 33204,87
445 REM *** SETS UP BOARD ***
450 POKE 33661,SL+48:POKE 33675,SR+48
460 IF SL=10 OR SR=10 THEN 840
470 A$=""
480 GET A$:IF A$="" THEN 480
490 IF A$="C" OR A$="1" THEN R=-1:D=1
500 IF A$="V" OR A$="2" THEN R=0:D=1
510 IF A$="B" OR A$="3" THEN R=1:D=1
520 IF A$="D" OR A$="4" THEN R=-1:D=0
530 IF A$="F" OR A$="5" THEN 680
540 IF A$="G" OR A$="6" THEN R=1:D=0
550 IF A$="E" OR A$="7" THEN R=-1:D=-1
560 IF A$="R" OR A$="8" THEN R=0:D=-1
570 IF A$="T" OR A$="9" THEN R=1:D=-1
580 IF R=0 AND D=0 THEN 470
585 REM *** ACCEPTS CONTROL ***
590 IF ASC(A$)<60 THEN RR=R:RD=D:V=RV:H=RH:P=87:S=SL:M=RM
600 IF ASC(A$)>60 THEN LR=R:LD=D:V=LV:H=LH:P=81:S=SR:M=LM
604 C=32768+H+40*V
608 N=32768+H+R+40*(V+D)
610 IF PEEK(N)>80 AND PEEK(N)<90 THEN 450
620 IF PEEK(N)=32 THEN POKE C,M:M=32:POKE N,P
630 IF PEEK(N)=90 THEN POKE C,M:M=90:POKE N,P:GOSUB 790:S=S+1
640 H=H+R:V=V+D
650 IF ASC(A$)<60 THEN RV=V:RH=H:SL=S:RM=M
660 IF ASC(A$)>60 THEN LV=V:LH=H:SR=S:LM=M
670 GOTO 450
675 REM *** TANK MOVEMENT ***
680 IF A$="5" THEN V=RV:H=RH:R=RR:D=RD:O=87
690 IF A$="F" THEN V=LV:H=LH:R=LR:D=LD:O=81
700 C=32768+H+40*V
710 N=32768+H+R+40*(V+D)
720 IF PEEK(N)<32 THEN POKE C,O:POKE N,46:H=H+R:V=V+D:O=32:GOTO 700
730 IF PEEK(N)=102 THEN POKE C,O:GOSUB 790:POKE N,32:GOTO 450
740 IF PEEK(N)=160 THEN POKE C,O:GOSUB 790:POKE N,102:GOTO 450
750 IF PEEK(N)=90 THEN POKE C,O:POKE N,46:H=H+R:V=V+D:O=90:GOTO 700
760 IF PEEK(N)=81 THEN SR=SR+1:POKE C,O:GOSUB 790:GOTO 450
770 IF PEEK(N)=87 THEN SL=SL+1:POKE C,O:GOSUB 790:GOTO 450
780 IF PEEK(N)>200 THEN POKE C,O:GOTO 450
785 REM *** FIRING ROUTINE ***
790 Z1=PEEK(N+1):Z2=PEEK(N-1):Z3=PEEK(N+40):Z4=PEEK(N-40)
800 POKE N+1,42:POKE N-1,42:POKE N+40,42
810 FOR X=0 TO 50:NEXT X
820 POKE N+1,Z1:POKE N-1,Z2:POKE N+40,Z3:POKE N-40,Z4
830 RETURN
835 REM *** EXPLOSION EFFECT ***
840 IF SL=10 THEN A$="LEFT"
850 IF SR=10 THEN A$="RIGHT"
860 PRINT"■■■THE GAME WAS WON BY THE ";A$;" PLAYER."
870 INPUT "■■■ANOTHER GAME";A$
880 IF LEFT$(A$,1)="Y" THEN GOSUB 900:GOTO 290
890 END
900 A$="":R=0:D=0:SL=0:SR=0:LM=32:RM=32:RETURN
```

String routines

by Michael Parr

These routines were designed to run on an Altair system but are intended for any Microsoft-type system — eg Tandy, PET etc.

String Changing

A common operation when working with character strings is to change part of a string, leaving the rest unaltered. For example, to change "COMPUTOR" to "COMPUTER" the operation can be specified as replacing "TO" by "TE". If we were imprecise, and just altered "O" to "E" then the result would be "CEMPUTER".

Some versions of BASIC have a statement of the form:

CHANGE F$ TO T$ IN L$

which automatically does the replacing. One may write:

```
10 L$ = "COMPUTOR"
20 CHANGE "TO" TO "TE" IN L$
30 PRINT L$
```

where COMPUTER is printed.

This is fine, but the commonly available Microsoft BASIC does not include such a statement. However, do not despair. Fig 1 gives the listing of an equivalent subroutine. The calling sequence must set L$ to the string to be changed, F$ to the section of L$ to be changed, and T$ to the new version of F$. As an example, the above operation is performed by:

```
10 L$ = "COMPUTOR"
20 F$ = "TO" : T$ = "TE"
30 GOSUB 1200
40 PRINT L$
```

Note:

a. It is possible to delete characters by setting T$ to a zero length string, thus:
20 F$ = "A" : T$="" : GOSUB 1200
would remove every letter "A" from L$.

b. If F$ is not found in L$, the subroutine does not change L$. However, an error may result if an attempt is made to extend L$ beyond the maximum possible length (usually 255).

The subroutine has a variety of uses:

a. The addition of some ten lines results in a simple file editor (fig 2), which has proved useful in converting programs written for different BASIC systems, which may use for example "instead of", and may need an argument for RND, i.e. RND(1).

b. A word processing system requires the facility to alter all occurences of a word to a different word. By the inclusion of spaces in F$, one can ensure that complete words are selected for alteration, as opposed to parts of a word.

c. The routine has been used as the heart of a simple macro-processor, taking up some 80 lines of BASIC.

An INSTR Routine

Frustrated Pet users will have realised that, though their BASIC includes LEFT$, RIGHT$, and MID$, the INSTR function (which locates the position of a substring within a string) is missing. Fortunately fig 3 lists a subroutine which exactly simulates the Altair INSTR function. It has been intentionally written in "simple" BASIC to aid implementation on a range of systems.

The routine takes F as the starting position of the search, and examines L$ for an occurrence of F$. The position is set in P8, and is zero if F$ is not found.

To produce the effect of:

1230 P8=INSTR(F, L$, F$)

use:

1230 GOSUB 2000

(1)

```
1200 REM * * CHANGE F$ TO T$ IN L$ * *
1210 REM USES S$,L1,F,P8
1220 F=1
1230 P8=INSTR(F,L$,F$)
1240 IF P8=0 THEN RETURN
1250 S$=L$
1260 L1=LEN(L$)
1270 IF L$=F$ THEN L$=T$:RETURN
1280 IF P8+LEN(F$)=L1+1 THEN L$=LEFT$(L$,P8-1)+T$:GOTO 1310
1290 IF P8=1 THEN L$=T$+RIGHT$(L$,L1-LEN(F$)):GOTO 1310
1300 L$=LEFT$(L$,P8-1)+T$+RIGHT$(S$,L1-P8-LEN(F$)+1)
1310 F=P8+LEN(T$)
1320 GOTO 1230
1330 REM SUB END
```

(2)

```
10 REM FILE EDITOR
20 CLEAR 500 ' ENSURE ENOUGH STRING SPACE (NOT NEEDED ON PET)
30 INPUT"WHICH FILE TO EDIT",A$
40 INPUT"NEW FILE";B$
50 REM OPEN "I"NPUT AND "O"UTPUT FILE
60 OPEN"I",1,A$ : OPEN"O",2,B$
70 LINEINPUT"ALTER ALL ";F$
80 LINEINPUT "TO ";T$
90 REM EDIT ALL FILE
100 IF EOF(1) THEN END
110 LINEINPUT£1,L$ : GOSUB 1200
120 PRINT£2,L$ : GOTO 100
```

(3)

```
2000 REM INSTR(F,L$,F$) : RESULT IS P8
2010 REM F9 USED
2020 F9=INT(F)
2030 IF F9>0 THEN 2050
2040 PRINT"ILLEGAL CALL OF INSTR ": STOP
2050 IF LEN(L$)=0 THEN 2120
2060 IF LEN(F$)=0 THEN 2120
2070 IF LEN(L$)<LEN(F$) THEN 2120
2080 IF F9>LEN(L$)-LEN(F$)+1 THEN 2120
2090 FOR P8=F9 TO LEN(L$)-LEN(F$)+1
2100 IF MID$(L$,P8,LEN(F$)) = F$ THEN 2130
2110 NEXT P8
2120 P8=0
2130 RETURN
```

TRS 80 Disable

by David Eyles.

For those of you who hold your programming abilities in high esteem, this short routine should relieve some of your concern when lending cassettes.

Its incorporation into programs prevents their listing, saving on tape and breaking of program execution (by either the 'BREAK' or (shift) @ key). However, these commands are still available — if you know the code.

The routine should be keyed in, run, and saved on tape... "It won't save!", I hear you say. Try CSAVE "X" (shift) C.

Similarly LIST, 'BREAK' and (shift) @ have been disabled, but can be accessed by typing LIST (shift) L and (shift) B. Unfortunately (shift) @ is lost.

If the routine is loaded from cassette (either by itself or as a part of a program), no listing is possible, even using LIST (shift) L. To utilize the codes, the routine must first be run.

```
1 FORA=32512TO32582:READB:POKEA,B:NEXT:POKE16405,155:POKE16406,88:POKE16407,127:POKE17121,1:POKE17122,2:POKE16425,2
2 DATA205,227,3,254,98,40,21,254,96,40,13,254,105,40,16,254,99,40,15,254,13,40,27,254,1,192,175,201,62,1,201,62,13,201
  ,229,33,205,88,62,255
3 DATA119,35,119,225,24,241,197,213,229,33,52,27,17,232,55,6,4,205,76,127,33,75,27,17,233,55,6,4,205,76,127,225,209,193,
  193,24,211,26,190,192,19,35,16,249,201,225,225,229,197,195,224,6
```

Next Month

COMPUTER GAMES
"And Then There Were Two". David Levy guides the programmer through the difficulties of adding a second player to a game.

BENCHTEST
Next month the Cromemco System Three will be put through its paces by Sue Eisenbach – another thorough dissection.

SYSTEMS
The life blood of most businesses is their cash flow. In fact many small businesses go bankrupt because they find it impossible to get their customers – particularly the large ones – to pay their accounts on time. Not surprisingly we have chosen the Sales Ledger as the first subject to be covered in our regular business software feature.

EDUCATION REPORT
We continue our look at micros and the education system with an article by John Skelton.

THE COMPLETE PASCAL
The fundamentals are considered in Part 2 of our special series.

BUSINESS COMPUTING II
The conclusion of our overview examining the role of micros in the business environment. Mailing lists, customised software and possible pitfalls in purchasing software packages are discussed.

FASTER MORE EFFICIENT PROGRAMS
Save time, memory and effort . . . the secrets are revealed.

PSYCHIC PERCEPTION – OR DECEPTION
ESP – the ultimate interface? Next month's cover story.

BUZZWORDS
Another mouthful of computer jargon

PLUS REGULAR FEATURES:
Newsprint, Computer Answers, In Store, Leisure Lines, Programs, Interrupt.

ADVERTISERS INDEX

People's Pascal

$29·95

At last your TRS-80* can run Pascal too! The Chung/Yuen "tiny" Pascal is fully implemented for Level II TRS-80*, 16K and up. You no longer need to be left out of the growing group of Pascal users, because People's Pascal gives you everything you need to write structured Pascal programs:

• tiny Pascal compiler • complete text editor for writing your programs • complete tiny Pascal monitor • sample Pascal programs • users manual (TRS-80 Computing issue 1:4)

People's Pascal is both a powerful, structured language and "CPU expeditor". People's Pascal programs execute at least four times faster than Basic, and often eight-times faster! Special functions open up the complete graphic capability of TRS-80*. You now have the means to write those dazzling, impressive, high-speed graphics programs that are great for games, plotting, statistics, etc.

For the serious computerist, side two of People's Pascal II (tape 6) contains a larger compiler and complete source to the compiler, written in Pascal. This means you can re-compile the compiler, making changes, adding features, etc. (but this will take at least 36K RAM and a solid knowledge of programming).

With the complete People's Pascal operating system, you can save and load both source (Pascal) programs, and compiled programs, to or from cassette tape. This means that once you have de-bugged a program, you can save the P-code (compiled program) and thereafter, to run the program, you need only load the super-fast P-code.

Here is a partial list of People's Pascal features: recursive procedure/functions • for (loop) • case if/then/else • one-dimensional arrays • write • read constant • repeat/until(loop) • "peek & poke" • plot (graphics for TRS-80*).

MS MICROSOFTWARE

Please send mail order to P.O. Box 119, Essendon, Victoria, 3040.
$29.95 includes postage within Australia

* Trade Mark of Tandy Corporation

The Challenger 4

Whichever way you look at it, no other computer offers so much for so little, and in colour!

Minifloppy – two can be added.

Full 53 key, keyboard which in polled mode can recognise up to 8 simultaneously depressed keys.

RF shielded aluminium case with 2 step baked on enamel finish.

Inside: real time clock and countdown timer 24K memory 6502A processor

Solid oiled walnut sides.

Control line interfaces, parallel lines for home security, accessory BUS, parallel I/O lines or other TTL signals.

Joystick sockets – add realism to games.

Colour adjustment.

Video Display 64 × 32

Sound outputs, programmable tone generators – 8 bit companding digital to analog convertors.

RF output for control of AC appliances.

Printer RS232 interface

Numeric key pad slots.

Cassette sockets.

You'd have to go a long way to get better value in a computer. It has execution speed that really separates the computers from the toys. We think the Challenger 4 is way ahead of anything you've seen so far, for a wide variety of uses including business, personal, educational and games, as well as a real-time operating system, word processor and a data base management system.

The Challenger 4 has a 2MHz 6502 processor, and if that's not fast enough we can supply the GT option with the 6502C processor, and 120 nanosecond memory which averages over one million instructions per second.

A real time clock and count down timer, a 64 x 32 display in 16 colours, including 8K memory in the cassette version, 24K for the minifloppy. A BUS structure allows easy plug in of extra memory or many more OHIO boards. The BUS means modularity. If you bought your vintage C2-4 in 1977 we can change the boards at a much lower cost than a new computer.

For the best surprise of all ask our opposition if they can provide all these facilities. When they can't, ask us!

Special offer with this advertisement only — bring it along with you when you visit your dealer and obtain $20 discount off your CHALLENGER 4 purchase.

TCG

TOMORROWS TECHNOLOGY TODAY

OHIO SCIENTIFIC

TCG Ohio Scientific Pty. Ltd. 31 Hume Street, Crows Nest, N.S.W. 2065 Phone: (02) 439-6477