ACC NEWSLETTER VOL 2 ISS 5 DEC 74

DEC DEKKO

ACC meeting on January 25 2.00pm

DIGITAL EQUIPMENT Co. (8th floor)
FOUNTAIN HOUSE
BUTTS CENTRE
READING, BERKS

DEC have arranged for a number of mini's to be available for hands-on use, and a number of staff will be on hand to talk informally about hardware, software interfaces, easy terms for purchasers ! (better leave your cheque book at home).

If you have specific questions you would like to ask DEC, or any particular equipment you would like to see, drop a line to Mike Lord (7 Dordells, Basildon, Essex) by 18th.

FOR SECURITY REASONS ONLY ACC MEMBERS WITH MEMBERSHIP CARDS WILL BE ADMITTED.

As DEC are going to a lot of trouble in laying on this show we would urge all ACC members who are able to come along.

Anyone wanting or willing to share transport to Reading get in touch with Mike Lord (0268 3040 x275 until 5.30)

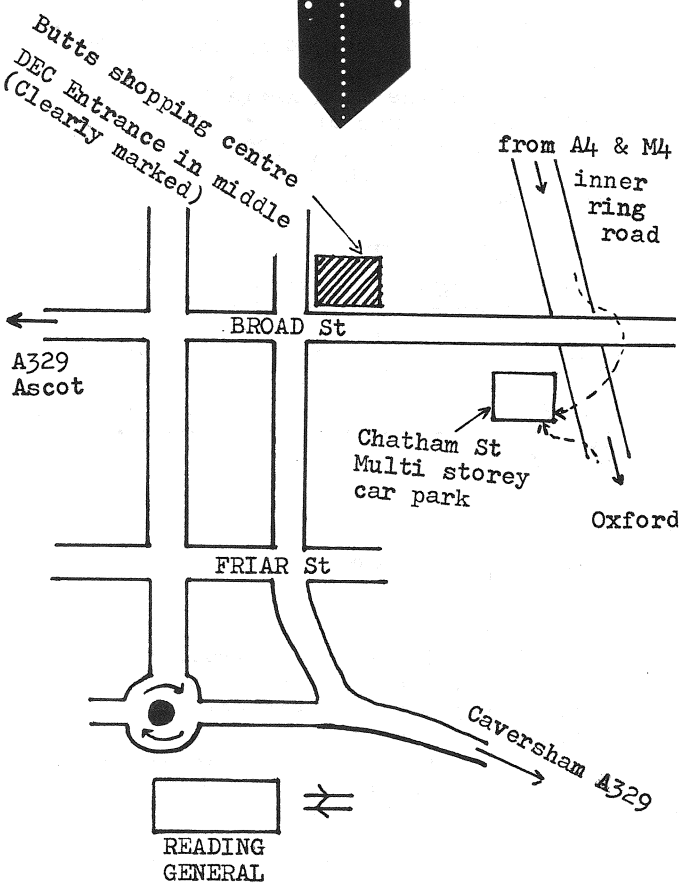
DEC's products include;

PDP-8 & PDP-11 range of minis.

System 10

MPS microprocessor modules

Various computer peripherals



Butts Centre about
½ mile from station

0·25p PER BIT

INTEL have introduced a range of 4096 x 1 bit dynamic random access MOS memory I/c. Difference between units in the range is speed. Slowest (and cheapest) is the 2107A-8. This has a cycle time better than luS, access time of 420 nS and costs £9.93 +VAT for 1 - 24, £7.72 + VAT for 25-100.

Supplies are +5V, +12V & -5V.

All inputs & outputs except for the Chip Enable input are TTL compatible.

As it is a dynamic memory (data is stored as charge on capacitors) it must be 'refreshed' by performing ane read cycle on each of 64 particular addresses every two milliseconds.

Although it is intended for use in large parallel memories - one chip per bit per 4K words - it could be used as a cheap shift register replacement to give a serial memory of say 512 eight bit words per chip.

Available from our old friends Walmore Electronics.

Data is on its way to us and will be featured in the next issue of the newsletter.

ICL NEW RANGE

Nigel Clark

Designated the 2900 series, the machines that have been officially released are the 2970 and 2980. They can be compared with other manufacturers' equipment as in Fig 1. ICL claims however that the data throughput is three times that of comparatively priced systems. A 2970 starts at over £1M.

Each machine is a Network Computer embodying features of CDC's STAR, IBM 370/195 and Burroughs B7700. The components of the network and their features are given below.

ICL	IBM	CDC	UNIVAC	BURRO'S
2980	370/168	175	1110	B7700
2970	370/158	174	1108	B6700

Fig 1

NETWORK COMPONENTS

1. Semi-conductor main store

The main store is interfaced by Store Multiple Access Controllers (SMACS) that direct instructions to any of the available processors and data through the Store Access Controllers (SAC).

The store is divided into 1K byte pages. Software however is divided into variable sized segments which are mapped into any available (though not necessarily contiguous) free pages by the SMAC's.

SMAC's provide an automatic overlay system. Only segments actually required for execution are kept in main store, new segments being brought into store as required. This method of virtual storage completely eliminates the need for programming the overlaying system. The main store is further refined by the provision of a hardware driven stack vital for compiling and exxcuting programs written in high level languages.

2. Store Access Controller (SAC)

This controller handles all the store accesses required by the peripheral controllers, leaving the Order Code Processors (OCP) to execute program instructions at high speed.

3. Order Code Processor (OCP)

The processors derive much of their performance from the use of two advanced techniques; pipelining and slave stores.

Pipelining allows the execution of several instructions simultaneously while slave stores ensure that the pipeline is provided with high speed access to the instructions and operands it requires.

4. 1900 Series or System 4 Processor / Emulator

One of these processors may be necessary for users upgrading from either of these systems. The 2900 cannot handle programs written in either 1900 PLAN or System 4 Usercode.

	2980		2970	
FEATURES	BASIC	ENH'NCED	BASIC	ENH'NCED
STORE	1M	8M	768K	6M
CYCLE	600nS / 16 BYTES		500nS / 8 BYTES	
SMAC	2	4	1	3
SAC	1	2	1	2
OCP	1	2	1	2
EMUL'R	-	1	-	1

Fig 2

ADAPTABLE NETWORK

A network computer allows the users maximum flexibility in handling the work load. Scientific users with a high processing element in the workload can have 2 OCP's whereas commercial users supporting large networks of on-line systems can have 2 SAC's. The available configurations can be made up of multiple components, the limits being shown in Fig 2. Whatever components are used in the network can be linked to each other.

A typical network is shown in Fig 3 If a component malfunctions, it can be remotely isolated and the system will degrade itself gracefully. This ensures a high resiliance to system breakdown and increases 'up-time' by 5 - 6%

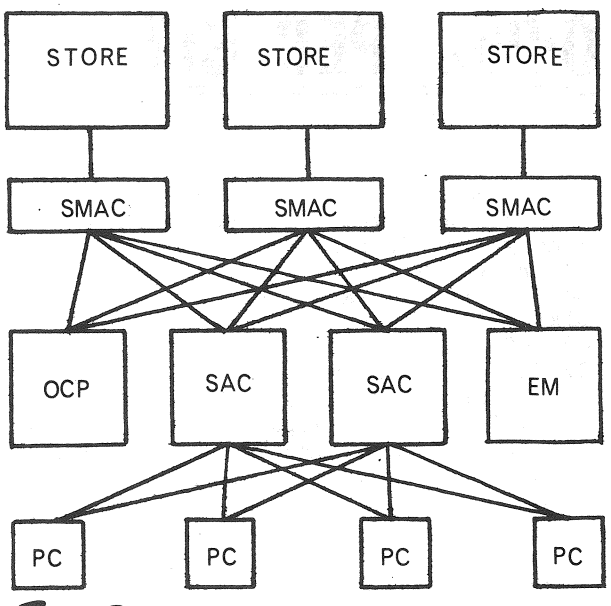


Fig 3

PERIPHERALS

All peripherals are connected to the Store Access Controllers via one of a number of peripheral controllers. Each 2980/2970 can support up to 16/8 such PC's.

Each PC is micr-programmed and therefore range-independent.

1. Sectored File Controller

Each SFC handles up to 4 of the fast fixed head discs. Each disc has a capacity of 6M bytes and an average access time of 5.2 mS. These devices are primarily used to support the 2900's virtual storage system.

2. Disc File Controller

Each DFC can handle up to 16 of the exchangeable disc store units. The stores range up to 200M bytes with an average access time of 38.3 mS.

3. General Peripheral Controller

GPC's control all the serial devices (CR,TR,LP etc) and also the 2900's operating stations and repeaters.

4. Communications Controllers

The basic version is the Communications Link Controller (CLC) which handles up to 8 Network Interface Modules (NIM's) Each NIM interfaces up to 16 channels of local or remote terminals. The CLC can be enhanced on-site to become a Communications Network Processor (2903 ?) which can be supported by local fixed and excgangeable disc stores and all manner of serial I/O devices.

SOFTWARE

2900 machines are byte oriented with machine code instructions of 16 & 32 bits. No other details are known of machine level languages. All software has been written in S3, an Algol 68 type language.

Compilers have been written to handle Algol, Fortran, Basic & Cobol only. This restriction has been deliberately imposed by ICL who have no intention of releasing their low level compilers. Nor is it possible to examine the existing software and deduce these low level languages. The machines incorporate a software protection device in the hardware structure. It is known as the Access Control Register (ACR) and it only allows software at one level of privelege to access data at its own or a lower level of privelege. There are 16 levels in the ACR, system software uses the 10 most priveleged levels, users the lowest 6.

APLLICATIONS

Application packages already available cover financial, commercial & mathematical topics. Also available is a Data Management System with a powerful data description language providing logical file mapping of physical files. For the user this means that changes in the data base only require recompilation of applications programs for the DMS to continue.

MICROPROCESSOR KITS ◆ ◆ ◆

Walmore Electronics Ltd. (of 11 - 15 Betterton St., London WC2H 9BS) can now supply 4 kits of INTEL I/C's each of which provides the basis for a microprocessor system (in each case a handful of simple TTL I/Cs, lamps, switches, power supply etc would be required to complete the job). The total cost of each kit is about ½ of the cost of the individual components when bought in small quantities.

MCS4 KIT A £54 + VAT

Comprises one each of the following devices;

4004 four bit CPU. 46 instructions including binary & decimal arithmetic. 10.8 uS instruction cycle time.

4002 320 bit RAM (arranged as four registers each containing twenty-four bit words) and four bit output port.

4003 10 bit serial in/parallel out, serial out shift register used to extend the number of output lines from the micro-processor.

4702A 256 word 8 bit electrically reprogrammable ROM (erased by exposure to ultraviolet light)

4008 & 4009 memory & I/O interface set.

MCS8 KIT A £134.50 + VAT

1 off 8008 12.5uS 8 bit CPU.

8 off 8102 1024 word one bit static RAM.

1 off 8205 high speed 1 out of 8 binary decoder.

1 off 8702A-4 256 word 8 bit electrically reprogrammable ROM.

8 of 8212 eight bit input/output port.

MCS80 KIT A £342 + VAT

1 off 8080 2.0uS 8 bit CPU.

8 off 8107A 4096 word one bit dynamic RAM.

1 off 8210 Bipolar to MOS driver.

8 off 8212 eight bit input/output port.

1 off 8702A 256 word 8 bit electrically reprogrammable ROM.

MCS80 KIT B £268 + VAT

1 off 8080 2.0uS 8 bit CPU.

8 off 8212 eight bit input/output port.

8 off 8102-2 1024 word one bit static RAM.

1 off 8702A 256 word eight bit electrically reprogrammable ROM.

Unfortunately Walmore are only supplying very sketchy data, hardly more than that given above, unless you actually buy one.

LETTERS

RECURSION

I should like to air some of my views on recursion as it is a subject that has caused much controversy among programmers.

First I must state my interest in programming. I work in a University, on Compilers & Operating systems (COBOL programmers please don't stop here !) Consequently I have been involved in some large software projects, all written in high level languages (mostly BCPL) and I should find programming a difficult task without the control structures (loops, recursive procedures, switches etc.) of a good language. Recursion is just one of the many features one expects to find in high level languages and which are an aid to producing concise, well-structured and easily read programms. These are features which ease the programmers' task and make it more likely that he will produce a correct program.

Secondly we must be sure of the meaning of the term 'recursion'. It was originally used in mathematics to denote the class of functions that were defined in terms of themselves;

eg the Factorial Function

f(n) = if n≤1 then 1 else f(n-1).n

the Bessel Functions

$$J_{n+1}(x) = \frac{2n}{x} J_n(x) - J_{n-1}(x)$$

the Ackermann Function

```
A(m,n) = if m=0 then n+1 else
         if n=0 then A(m-1,1) else
         A(m-1,A(m,n-1))
```

(try computing this for m = n = 4)

This form of functional definition is very elegant and is used in many branches of mathematics (Numerical Analysis Logic, Graph Theory as well as Recursive Function Theory).

With the advent of programming languages like Algol 60, recursion has come to mean the definition of computing processes in terms of themselves. (There is a distinction between functions and computing processes, but I shall not go into it here). We can now write programs to compute recursive functions that look like their mathematical counterparts.

```
eg  real procedure J(n,x) ;
    integer n ; real x ;
    J := (2X(n-1)/x) X J(n-1,x)-J(n-2,x)
```

This similarity has obvious benefits for program correctness.

There is often confusion between the terms 'recursive' and 'reentrant'. 'recursive' is used to describe the form of definition of a program, whereas 'reentrant' is a term used to describe a dynamic property of the program code. This confusion arises as reentrant code is almost invariably used to implement recursive programs.

Lastly some uses of recursion; as I have mentioned, many mathematical functions are defined recursively, and a recursive program is the obvious (though not always the most efficient) way to compute these.

There are many forms of data that have an inherently recursive nature eg bracketted arithmetic expressions, tree structures, lists etc., and it is difficult to deal with these without recursion. A couple of often neglected uses for recursion are in counting and re-ordering. As example here is a program that reads a word (terminated by a full-stop) then prints it backwards and counts the number of letters it contains

```
let  ReverseWord [ ] = valof
§ let Char = Next [In] ;
  test Char = '.'
  then resultis 0
  else § let n = ReverseWord [ ] +1
         Out [Char]
         resultis  n
       §
§
```

This is written in BCPL which might require some explanation.

The first line defines a function called ReverseWord. The paragraph brackets and are equivalent to Algol begin and end and define a block. The use of valof and resultis means that the value of an expression occurring after a resultis is returned as the value of the function.

The second line defines a variable Char which takes the value of the next character on the input stream.

If the character is a full-stop then we return from the function with the value 0, otherwise we reverse the remainder of the word, output the current character and then return with the number of letters already output.

Out is a routine that outputs a single character.

Finally an example taken from our Operating System. It is a routine that will output a positive integer ;

```
let  OutP[n] be
§ if n> 9 do OutP[n/10]
  Out['0' + n rem 10]
§
```

'rem' gives remainder on integer division. This routine is shorter and more efficient (in our hardware) than any non-recursive version and moreover is independant of the maximum size of n.

I hope that some of these examples will have demonstrated the use of recursion as a programming technique.

Any comments (rude or otherwise) will be gladly answered.

R.I.Cowderoy.

COSTS & AIMS

I have been browsing through the past copies of the ACC newsletter over the past few days and, well to put it bluntly, I've been hit by the bug!

I have decided that I would like to buy a digital machine. The thing is, I only wish it was that simple - just to get up and buy one. I'm afraid I really haven't got a clue as to how to go about getting one or what I can expect to pay. It's really not a matter of what sort of machine I would like, its what I can afford .Can you help me? I'm completely in the dark about what is available, where, and what the cost would be.

There are a few glimmers of a guide in the past newsletters. In Vol 1 Iss 3 Mr. Aslett writes that he has 3 803b's for an outlay of around £400 - about £130 for each system. If this is typical then I think there is hope for me - I could afford about £100. Is there hope?

While on this point we have never seen an article in the newsletter to say what sort of outlay will buy what sort of system- or is it that there is no way of telling ? I appreciate that it will vary widely but surely an indication could be given of what type of equipment is coming on to the market and what an average outlay would be.

There was a call for suggestions on these lines in Vol 1 Iss 4. Maybe another cry for help could be put in a future issue ? The request produced a reply from Mr P Turner who names a source in Surrey which I intend to follow up.

On a different point, looking at the most recent issue and comparing it with the first issue it looks as though the club is coming on great. The new newsletter looks altogether more professional and well organised. The series on cores for stores was, I thought, a great idea and I for one found it very interesting. Is there a similar series planned for future issues or do we lack authors ?

I found the last issue very interesting the 8008 seems a very appealing device as a basis for amateur machines, it is a pity that reduced prices could not be obtained for members. This brings me to my last point;

A few more months will bring us to the end of the clubs second financial year. Surely we should be looking for an aim for the club, some function apart from the newsletter for the club to provide. The idea of trying to obtain components cheaply must surely be quite high on the list and must be appealing to most of us. I have no idea what sort of state the club's finances are in, although I am certain they are well looked after, but how about holding a competition of some kind to raise funds and interest ?

G Simpson.

* * * Thanks for the letter and the nice comments about the newsletter. You have put your finger on a number of points that the ACC Committee have been ruminating about for some time now.

First - costs. You have stirred me to action & the result will be an article in the next issue. To prevent it being too myopic a view I would like to hear other members' reactions also.

Second - yes we do lack authors - especially on software topics so those of you who understand Modular Structured GOTOless Recursive (or is it re-entrant) Macro High Level programmins please let us all into the secret.

Finally - an aim or function for the Club.

Cheap components is always a worthwhile aim and there seem to be three possibilities;

a) For the Club to buy stocks in reasonably large quantities to get low prices. This has been discussed by the Committee & we decided that it is not feasible - very large amounts of money would be needed and it would benefit only those members who are actively engaged in construction.
b) To try & persuade manufacturers & distributors to give a discount to club members. This we will try.
c) In some way to pool members individual purchases - so that if say 3 members each wanted to buy 35 TTL packages they could combine their orders to qualify for the 100 + quantity discount level. Seems a good idea but I'm not sure how it could be worked - any ideas anyone ?

One idea kicked around at a recent committee meeting was that the club could design & build some piece of equipment. The various stages of design etc. would be detailed in the newsletter & would possibly be of interest to most members. Any comments ?

Incedentally - the clubs finances are quite good despite inflation - mainly because the membership has risen this year (now around 270) instead of falling slightly as had been expected.

mike lord * * * * * *

AUTOMATIC SENTENCES

I have a fully debugged BASIC program which generates sentences at random according to a syntax table which is fed to the program as data. It will also, given a sentence and a syntax table, produce an analysis record of the sentence; this technique has applications in compiler building.

I will supply copies to ACC members on receipt of a stamped addressed (9x6 inch minimum size) envelope.

A.Fisher
Nenadd Davies Bryan , Penbryn,
University College of Wales,
Aberystwyth.

FOR SALE

IBM golfball typewriter with electric keyboard etc & many spares and manuals.

Core Stack 48K (I think !) plus details

Card Reader (80 column) mechanically complete less electronics but with manuals etc.

REQUIRED

Minicomputer. Built /nearly built / working or not. PDP or any system considered.

D.V.GOADBY
5 Queens Rd., Hinckley,Leics LE10 1ED
Hinkley (0455) 36621

BOOKLIST

The books listed below have been suggested by various members, thanks to them for their help.

COMPUTER STRUCTURES : READINGS & EXAMPLES

C.Gordon Bell & A.Newell
McGraw-Hill Inc 1971 650pp £7-90

..... a case study approach to cover 40 distinct computer types, and to provide a taxonomical framework for analysing the 1000 different computers now extant. Of the 40 computers described 10 are based on unpublished sources .....

..... the authors offer the reader the opportunity of comparing different computers as to their capabilities and limitations ..... (quotations from fly leaf)

Machines described include;

Olivetti p101	ZEBRA
HP model 9100A	ACE
DEC PDP-8	RW 400
IBM 650 to /360	MIDAC
B 5000	NOVA
CDC 6600	etc.

DESIGN OF DIGITAL COMPUTERS
H.W.Gschwind Spronger-Verlag 1967

This book is intended as an introductory text concerned with the design of digital computers. Computer programming and operation are mentioned only where they have a direct bearing on the equipment.

PROGRAMMING SYSTEMS & LANGUAGES
S.Rosen ed McGraw Hill 1967

COMPUTERS - A PRACTICAL APPROACH
Marchant & Pegg Blackie

THEORY & DESIGN OF DIGITAL COMPUTERS
D.Lewin 1972 Nelson

DIGITAL COMPUTER SYSTEM PRINCIPLES
H.Hellerman McGraw Hill

INSTRUCTION SET FOR AN AMATEUR COMPUTER

While you can build a computer by copying a small commercial mini such as the PDP-8 or by basing your design upon one of the microprocessor chips, there is no doubt that it is more fun to design your own machine from scratch, and in doing so one can take advantage of the peculiar requirements of the amateur.

One main difference between the commercial user and the amateur is the attitude to absolute processing speed. After all the amateur isn't paying by the second for his machine time, and he doesn't want to connect large numbers of terminals into a big time sharing system or indulge in astronomical number crunching exercises. As long as the machine can say add two binary words in less than 100uS it will provide an adequate response for a single user. (in passing I've found that when programming simple games on a mini it is often better to add artificial delays into the program to give a discernable pause before the machine makes its move. For some psychological reason this gives the impression that the machine is thinking about the problem rather than reacting automatically).

On the other hand cost is of paramount importance. This implies a simple architecture and a machine language that is simple to decode (to reduce the cost of the control logic) yet which makes efficient use of expensive memory. Unfortunately microprogramming - now used to implement the control section of most commercial machines - is an expensive option for the amateur who obviously can't have special, large, ROM I/C's made for him.

Ideally the constructor should be able to start by making a 'micro-mini' with say 100 words of memory and a limited instruction set which can be built upon to -eventually- arrive at a machine comparable with a small mini.

Choice of word length is an interesting problem. Either 8 or 16 bits would seem to be a reasonable choice. An 8 bit word does have a slight cost advantage (although not nearly as much at might first be thought because all registers which could be used to hold a memory address - Program Counter, Stack Pointer, Memory Address Register etc.- must be more than 8 bits anyway and you are then forced to deal with multiple-word instructions and addresses. A 16 bit word does give a much 'cleaner' design and, all things considered, seems the better bet.

Fig 1 shows a suitable architecture. All data transfers are carried out over a single bi-directional bus. Two further buses carry address and control information. The Arithmetic Unit has two (X&Y) registers for temporary storage of operands and can be used for manipulation of program data and for the calculation of effective addresses.The X & Y

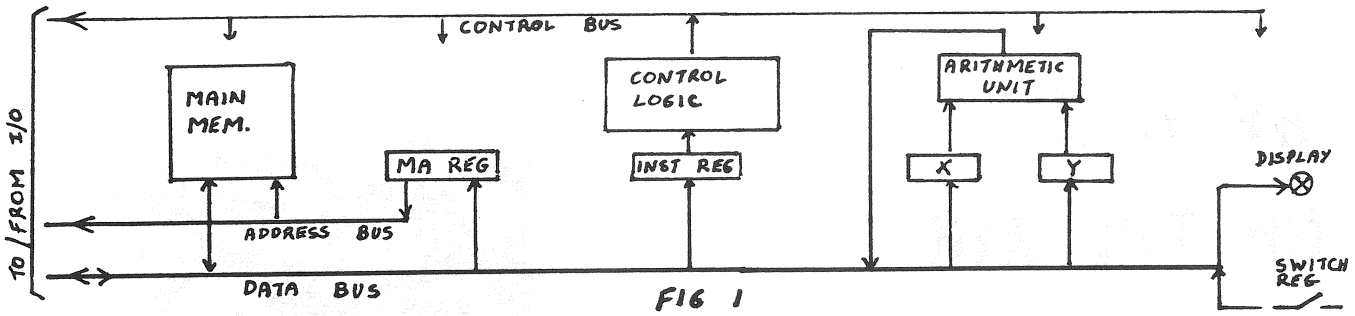


FIG 1

registers are not accessible by the machine language programmer who sees a Stack machine. The bottom few words of memory are used as a Program Counter and Stack Pointer and also provide a couple of temporary data registers used by the control section.

The instruction set & control logic are simplified by treating all I/O devices as though they are part of the main memory (I/O device data & status registers are allocated addresses which are within the machine's addressing range but which are not used by main memory).

Most instructions use a 16 bit word divided into 3 parts;
- 8 bit 'short address' Y
- 2 bits define the addressing mode
- 6 bits operation code

Data Manipulation Instructions

These include 'Two Address' type instructions such as ADD, COMPARE where one of the operands is the Top of Stack, and 'Single Address' instructions such as CLEAR, INCREMENT.

Data can be split between GLOBAL items, which are referenced by any part of the program and LOCAL or temporary variables used only within individual program segments.

The number of GLOBAL items is always very small (counting lists & arrays as one variable) and as it is extremely unlikely that the amateur will want to multi-program his machine we can designate the bottom 256 words of memory as a GLOBAL data storage area. The short address Y can therefore address any GLOBAL item directly. List and tables (and I/O devices) can be handled by storing a pointer in the GLOBAL area and using indirect addressing.

Most commercial machines use a small number of general purpose registers as temporary storage for LOCAL variables. This give a speed improvement (which we don't need) and reduces the number of address bits required which is a GOOD THING. Unfortunately keeping track of the contents and vunerability of these registers is a headache. A better approach from our point of view would be to increase the use of the Stack by allowing a limited range of Stack Relative Addressing to provide the temporary storage. This will also provide a simple way for passing data to and from subroutines.

One interesting effect of the use of stack, stack relative and global areas for data storage is that instructions which modify the program itself are not 'natural' (although possible) so there will be a tendency to write the programs as pure procedure which, the experts tell us, is another GOOD THING.

An IMMEDIATE addressing mode (data word immediately follows the instruction word) could be added to allow storage of constants within the procedure.

Program Control Instructions.

These include conditional & unconditional JUMPS and JUMP to SUBROUTINE (return address put onto stack).

In practice most JUMP instructions have a limited range (they jump over relatively few words) so we can squeeze a JUMP instruction onto one word by using Program Counter Relative addressing i.e JUMP forward (or back) Y words. This will cover most cases but we must allow for a two word JUMP instruction (target address in second word) for completeness.

JUMP to SUBROUTINE instructions, on the other hand, often transfer control between widely seperate addresses , but the number of subroutines in a program is usually limited to around 50 so we can achieve a single-word subroutine call by storing the full subroutine entry address in the GLOBAL data area and referencing it via the 8 Y bits of the JMS instruction.

Summary of Addressing Modes

data manipulation;

0:	Y	: data is in first 256 word (GLOBAL) block of memory at address Y
1:	SP+Y	: data is Y words from the top of the stack
2:	(Y)	: address of the data is in location Y (in the GLOBAL area)
3:	(SP+Y)	: address of the data is contained in the word Y words from top of stack.

program control;

0:	PC+Y	: address is formed by adding the number Y to the current contents of the program counter
1:	PC-Y	: address is formed by subtracting the number Y from the current contents of the program counter.
2:	(Y)	: address is contained in word Y of the GLOBAL area
3:	2 wd	; address is in the word immediately following the instruction.

WE WERE ON THE AIR !

BBC Look North viewers were treated to an unusual sight on Thursday 12 Dec when a film of my Elliott 803B computer installation was shown on the regional news programme.

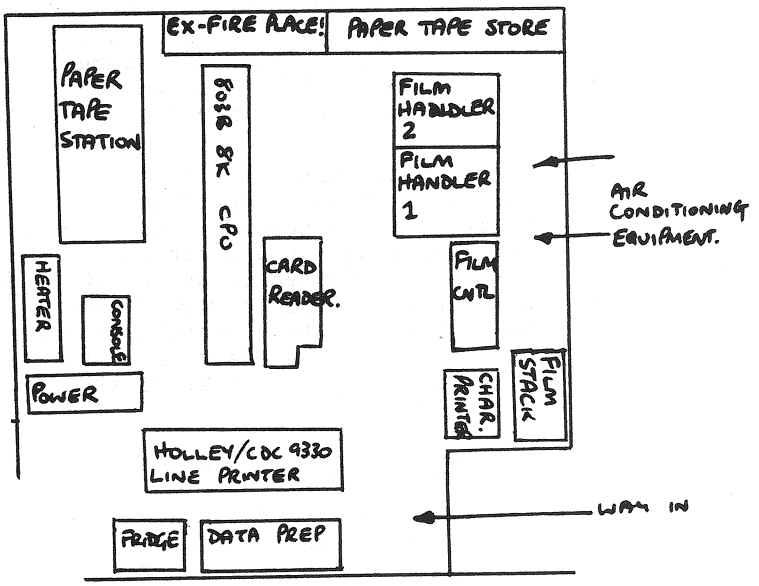
The film was 'shot' on the previous afternoon and the picture sums up the astonishment of the reporter & camera man in finding so much equipment in so small a space. The poor cameraman had considerable difficulties filming and lighting the various shots and the whole sequence of some 1 minute took the three man crew 3 hours to get in the can!

The computer finished the sequence, not by blowing up or walking into the sunset, but by gently playing the carol 'Oh Come All Ye Faithful'. The music

capability of the machine was the one thing which really mystified the crew and it took ½ hour to convince the sound man that the computer was playing the tune rather than a hidden tape recorder.

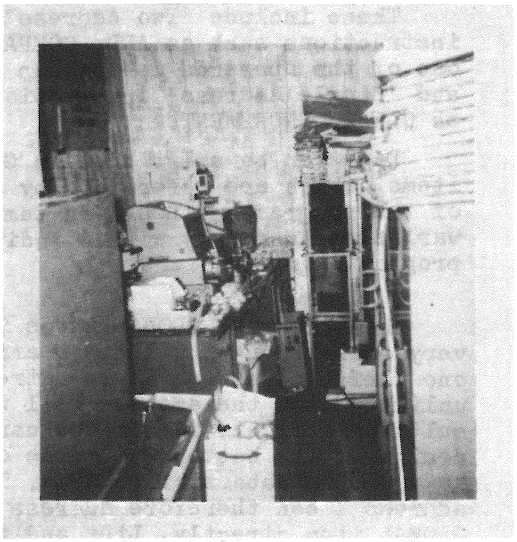
The other photo shows part of the installation (paper tape station and console)

J.Aslett



FLOOR PLAN OF THE 803B COMPUTER CENTRE LOCATED ON THE GROUND FLOOR OF 2 PARK STREET HUDDERSFIELD.

ROOM SIZE 13'x15'



ARTICLES WANTED

(Especially on software topics)

DON'T FORGET

Support the ACC meeting at DEC

COST IS THE MOST IMPORTANT FACTOR IN THE DESIGN OF ANY HIGH LEVEL LANGUAGE

Therefore;

a) If you don't like a feature then eliminate it on the grounds that it will cost too much in the compiler.

b) If you like the feature argue for it because it saves costs in the compiler.

PERHAPS & MAYBE STATEMENTS ARE DIFFICULT TO COMPILE

AMATEUR COMPUTER CLUB NEWSLETTER

Vol 2 Iss 5 Dec 1974

7 Dordells, Basildon, Essex

95mm